Broadcast Psi-calculi

Palle Raabjerg Johannes Åman Pohjola Tjark Weber

March 17, 2025

Abstract

We provide an Isabelle/HOL-Nominal formalisation of the definitions, theorems and proofs in the paper *Broadcast Psi-calculi with an Application to Wireless Protocols* by Borgström et al., which extends the Psi-calculi framework with primitives for broadcast communication in order to model wireless protocols.

1 Introduction

We provide an Isabelle/HOL-Nominal formalisation of the definitions, theorems and proofs in the paper *Broadcast Psi-calculi with an Application to Wireless Protocols* [4, 5], which extends the Psi-calculi framework [2, 3, 1] with primitives for broadcast communication in order to model wireless protocols.

The file `Broadcast_Thms.thy` contains a collection of the relevant definitions and theorems, with comments relating them directly to the paper.

2 Formalisation

theory *Broadcast-Chain*
  imports *Psi-Calculi.Chain*
begin

lemma *pair-perm-fresh-contr*:
  fixes *a*::*'a* **and** *b*::*'a*
  assumes
    at: *at TYPE*(*'a*)
  and
    prems: *b* ♯ *pi* (*a*, *b*) ∈ *set pi*
  shows *False*
  using *prems*
  by(*induct pi*)
    (*auto simp add*: *supp-list-cons fresh-list-nil supp-prod at-supp*[*OF at*]
                fresh-list-cons fresh-prod at-fresh[*OF at*])

lemma *pair-perm-fresh-contr′*:
fixes *a*::*′a* **and** *b*::*′a*
assumes
at: *at TYPE*(*′a*)
and
prems: *a* ♯ *pi* (*a*, *b*) ∈ *set pi*
shows *False*
using *prems*
by(*induct pi*)
(*auto simp add*: *supp-list-cons fresh-list-nil supp-prod at-supp*[*OF at*]
fresh-list-cons fresh-prod at-fresh[*OF at*])

lemma *list-set-supp*:
fixes *l* :: (*′d*::*fs-name*) *list*
shows *supp* (*set l*) = (*supp l* :: *name set*)
proof(*induct l*)
case *Nil* **then show** *?case*
by (*simp add*: *supp-list-nil supp-set-empty*)
next
case (*Cons x xs*) **then show** *?case*
using *at-name-inst fs-name-inst pt-list-set-supp pt-name-inst* **by** *blast*
qed

lemma *name-set-supp*:
assumes *finite a*
shows *supp a* = (*a*::*name set*)
using *assms*
by(*rule at-fin-set-supp*[*OF at-name-inst*])

lemma *supp-idem*:
fixes *l* :: (*′d*::*fs-name*)
shows *supp*((*supp l*)::*name set*) = (*supp*(*l*)::*name set*)
proof −
have *f*: *finite*((*supp l*)::*name set*)
by *finite-guess*
show *?thesis*
by(*rule name-set-supp*[*OF f*])
qed

lemma *fresh-supp*:
fixes *a* :: *name*
and *X* :: (*′d*::*fs-name*)
shows *a* ♯ ((*supp X*)::*name set*) = *a* ♯ *X*
by(*simp add*: *fresh-def supp-idem*)

lemma *fresh-chain-supp*:
fixes *A* :: *name list*
and *X* :: (*′d*::*fs-name*)
shows *A* ♯∗ ((*supp X*)::*name set*) = *A* ♯∗ *X*

unfolding *fresh-star-def*
by(*simp add*: *fresh-supp*)

lemma *fresh-chain-fin-union*:
fixes X::($'d$::*fs-name set*)
and Y::($'d$::*fs-name set*)
and A::*name list*
assumes *f1*: *finite* X
and *f2*: *finite* Y
shows $A\sharp*(X \cup Y) = (A\sharp*X \wedge A\sharp*Y)$
unfolding *fresh-star-def*
apply(*subst fresh-fin-union*[*OF pt-name-inst*, *OF at-name-inst*, *OF fs-name-inst*, *OF f1*, *OF f2*])
by *blast*

lemma *fresh-subset*:
fixes S :: *name set*
and S' :: *name set*
and a :: *name*
assumes $a \sharp S$
and $S' \subseteq S$
and *finite* S

shows $a \sharp S'$
proof −
have *finite* S' **using** ‹$S' \subseteq S$› ‹*finite* S›
by(*rule Finite-Set.finite-subset*)
with *assms* **show** *?thesis*
by(*auto simp add*: *fresh-def supp-subset Chain.name-list-supp name-set-supp*)
qed

lemma *fresh-subset'*:
fixes S :: $'d$::*fs-name set*
and S' :: $'d$::*fs-name set*
and a :: *name*
assumes $a \sharp S$
and $S' \subseteq S$
and *finite* S

shows $a \sharp S'$
proof −
have *finite* S' **using** ‹$S' \subseteq S$› ‹*finite* S›
by(*rule Finite-Set.finite-subset*)
have *supp* $S' \subseteq$ ((*supp* S)::*name set*)
apply(*rule Chain.supp-subset*)
by *fact+*
with *assms* **show** *?thesis*
unfolding *fresh-def*
by *auto*

qed

lemma *fresh-star-subset'*:
fixes S :: *'d::fs-name set*
and S' :: *'d::fs-name set*
and A :: *name list*
assumes $A \sharp* S$
and $S' \subseteq S$
and *finite* S

shows $A \sharp* S'$
using *assms*
unfolding *fresh-star-def*
by(*auto simp add*: *fresh-subset'*)

lemma *fresh-star-subset*:
fixes S :: *name set*
and S' :: *name set*
and A :: *name list*
assumes $A \sharp* S$
and $S' \subseteq S$
and *finite* S

shows $A \sharp* S'$
using *assms*
unfolding *fresh-star-def*
by(*auto simp add*: *fresh-subset*)

lemma *times-set-fresh*:
fixes a :: *name*
and S :: *name list*
and S' :: *name list*
assumes $a \sharp$ *set* S
and $a \sharp$ *set* S'
shows $a \sharp$ *set* $S \times$ *set* S'
using *assms*
proof(*cases* S)
case *Nil* **then show** *?thesis* **by**(*simp add*: *fresh-set-empty*)
next
case (*Cons s Svec*) **then show** *?thesis*
proof(*cases* S')
case *Nil* **then show** *?thesis* **by**(*simp add*: *fresh-set-empty*)
next
case (*Cons* s' S') **then show** *?thesis*
using ‹$S = s \# Svec$› *assms*
apply(*subst fresh-cart-prod*[*OF pt-name-inst*, *OF pt-name-inst*, *OF fs-name-inst*, *OF fs-name-inst*, *OF at-name-inst*])
by(*simp*+)
qed

qed

lemma *times-set-fresh-star*:
  fixes A :: *name list*
    and S :: *name list*
    and S' :: *name list*
  assumes $A \sharp* set\ S$
    and $A \sharp* set\ S'$
  shows $A \sharp* (set\ S \times set\ S')$
  using *assms*
  unfolding *fresh-star-def*
proof(*induct A*)
  case *Nil* **show** *?case* **by** *simp*
next
  case(*Cons a A*)
  have $a \sharp set\ S$ **and** $a \sharp set\ S'$ **using** *Cons* **by** *simp+*
  then have $a \sharp (set\ S \times set\ S')$ **by**(*rule times-set-fresh*)
  have $\forall\ x \in set\ A.\ (x \sharp set\ S)$ **and** $\forall\ x \in set\ A.\ (x \sharp set\ S')$ **using** *Cons*
    by *simp+*
  then have $\forall x \in set\ A.\ x \sharp set\ S \times set\ S'$ **using** *Cons* **by** *simp*
  then show *?case* **using** $\langle a \sharp (set\ S \times set\ S')\rangle$
    by *simp*
qed

lemma *supp-list-set*:
  fixes *M*::*'d*::*fs-name list*
  shows (*supp M*) = ((*supp*(*set M*))::*name set*)
proof(*induct M*)
  case *Nil* **then show** *?case* **by**(*simp add*: *supp-set-empty supp-list-nil*)
next
  case (*Cons m M*)
 have *lhs*: (*supp* (*m* # *M*)::*name set*) = *supp m* ∪ *supp M* **by**(*simp add*: *supp-list-cons*)
  have *rhs*: (*supp* (*set* (*m* # *M*))::*name set*) = *supp m* ∪ *supp M*
  proof −
    have *supp* (*set* (*m* # *M*)) = (*supp* (*set* [*m*] ∪ *set M*)::*name set*)
      by *simp*
    moreover have ... = *supp* (*set* [*m*]) ∪ *supp*(*set M*)
    apply(*rule supp-fin-union*[*OF pt-name-inst*, *OF at-name-inst*, *OF fs-name-inst*])
      by *simp+*
    moreover have ... = *supp m* ∪ *supp M*
      using *calculation*(*1*) *calculation*(*2*) *lhs list-set-supp* **by** *blast*
    ultimately show *?thesis*
      by *simp*
  qed
  show *?case*
    unfolding *lhs rhs*
    by(*rule refl*)
qed

lemma *fresh-list-set*:
fixes M::$'d$::*fs-name list*
and A::*name list*
shows $A \sharp* set\ M = A \sharp* M$
unfolding *fresh-star-def fresh-def supp-list-set*
by(*rule refl*)

lemma *permSupp*:
fixes Ψ :: *name prm*
and Ψ' :: $'d$::*fs-name*

shows (*supp*($\Psi \cdot \Psi'$)::*name set*) $\subseteq$ ((*supp* Ψ) $\cup$ (*supp* Ψ'))
proof −
{
fix x::*name*
let *?P* = $\lambda y.\ ([(x,\ y)] \cdot \Psi) \cdot ([(x,\ y)] \cdot \Psi') \neq \Psi \cdot \Psi'$
let *?Q* = $\lambda y\ \Psi.\ ([(x,\ y)] \cdot \Psi) \neq \Psi$
assume *finite* {*y*. *?Q y* Ψ'}
moreover assume *finite* {*y*. *?Q y* Ψ}
and *infinite* {*b*. $[(x,\ b)] \cdot \Psi \cdot \Psi' \neq \Psi \cdot \Psi'$}
from ‹*infinite* {*b*. $[(x,\ b)] \cdot \Psi \cdot \Psi' \neq \Psi \cdot \Psi'$}› **have** *infinite* {*y*. *?P*(*y*)}
by(*subst cp1*[*symmetric*, *OF cp-pt-inst*, *OF pt-name-inst*, *OF at-name-inst*])
then have *infinite*({*y*. *?P*(*y*)} − {*y*. *?Q y* Ψ})
using ‹*finite* {*y*. *?Q y* Ψ'}› ‹*finite* {*y*. *?Q y* Ψ}›
by − (*rule Diff-infinite-finite*)
ultimately have *infinite*(({*y*. *?P*(*y*)} − {*y*. *?Q y* Ψ}) − {*y*. *?Q y* Ψ'})
by(*rule Diff-infinite-finite*)
then have *infinite*({*y*. *?P*(*y*) ∧ ¬(*?Q y* Ψ) ∧ ¬ (*?Q y* Ψ')}) **by**(*simp add*: *set-diff-eq*)
moreover have {*y*. *?P*(*y*) ∧ ¬(*?Q y* Ψ) ∧ ¬ (*?Q y* Ψ')} = {}
by *auto*
ultimately have *infinite* {}
by − (*drule Infinite-cong*, *auto*)
then have *False* **by** *simp*
}
from *this* **show** *?thesis*
by(*force simp add*: *supp-def*)
qed

end
theory *Broadcast-Frame*
imports *Psi-Calculi.Frame*
begin

locale *assertionAux* = *Frame.assertionAux SCompose SImp SBottom SChanEq*
for *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*)

+
fixes *SOutCon* :: *'a::fs-name* ⇒ *'a* ⇒ *'c* (‹- ≼ -› [*80*, *80*] *80*)
and *SInCon* :: *'a::fs-name* ⇒ *'a* ⇒ *'c* (‹- ≽ -› [*80*, *80*] *80*)

assumes *statEqvt'''*[*eqvt*]: ⋀*p*::*name prm*. *p* ∙ (*M* ≼ *N*) = (*p* ∙ *M*) ≼ (*p* ∙ *N*)
and *statEqvt''''*[*eqvt*]: ⋀*p*::*name prm*. *p* ∙ (*M* ≽ *N*) = (*p* ∙ *M*) ≽ (*p* ∙ *N*)

begin

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

shows (*supp*(*M* ≽ *N*)::*name set*) ⊆ ((*supp M*) ∪ (*supp N*))
proof −
{
fix *x*::*name*
let *?P* = λ*y*. ([(*x*, *y*)] ∙ *M*) ≽ [(*x*, *y*)] ∙ *N* ≠ *M* ≽ *N*
let *?Q* = λ*y M*. ([(*x*, *y*)] ∙ *M*) ≠ *M*
assume *finite* {*y*. *?Q y N*}
moreover assume *finite* {*y*. *?Q y M*} **and** *infinite* {*y*. *?P*(*y*)}
then have *infinite*({*y*. *?P*(*y*)} − {*y*. *?Q y M*}) **by**(*rule Diff-infinite-finite*)
ultimately have *infinite*(({*y*. *?P*(*y*)} − {*y*. *?Q y M*}) − {*y*. *?Q y N*}) **by**(*rule Diff-infinite-finite*)
then have *infinite*({*y*. *?P*(*y*) ∧ ¬(*?Q y M*) ∧ ¬ (*?Q y N*)}) **by**(*simp add*: *set-diff-eq*)
moreover have {*y*. *?P*(*y*) ∧ ¬(*?Q y M*) ∧ ¬ (*?Q y N*)} = {} **by** *auto*
ultimately have *infinite* {} **by**(*blast dest*: *Infinite-cong*)
then have *False* **by** *simp*
}
then show *?thesis* **by**(*auto simp add*: *eqvts supp-def*)
qed

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

shows (*supp*(*M* ≼ *N*)::*name set*) ⊆ ((*supp M*) ∪ (*supp N*))
proof −
{
fix *x*::*name*
let *?P* = λ*y*. ([(*x*, *y*)] ∙ *M*) ≼ [(*x*, *y*)] ∙ *N* ≠ *M* ≼ *N*
let *?Q* = λ*y M*. ([(*x*, *y*)] ∙ *M*) ≠ *M*
assume *finite* {*y*. *?Q y N*}
moreover assume *finite* {*y*. *?Q y M*} **and** *infinite* {*y*. *?P*(*y*)}
then have *infinite*({*y*. *?P*(*y*)} − {*y*. *?Q y M*}) **by**(*rule Diff-infinite-finite*)
ultimately have *infinite*(({*y*. *?P*(*y*)} − {*y*. *?Q y M*}) − {*y*. *?Q y N*}) **by**(*rule Diff-infinite-finite*)
then have *infinite*({*y*. *?P*(*y*) ∧ ¬(*?Q y M*) ∧ ¬ (*?Q y N*)}) **by**(*simp add*:

set-diff-eq)
  moreover have $\{y.\ ?P(y) \wedge \neg(?Q\ y\ M) \wedge \neg\ (?Q\ y\ N)\} = \{\}$ **by** *auto*
  ultimately have *infinite* $\{\}$ **by**(*blast dest*: *Infinite-cong*)
  then have *False* **by** *simp*
 }
 then show *?thesis* **by** (*auto simp add*: *eqvts supp-def*)
qed

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

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

shows $x \sharp M \succeq N$
 using *assms chanInConSupp*
 by(*auto simp add*: *fresh-def*)

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

shows $[\![xvec \sharp* M;\ xvec \sharp* N]\!] \Longrightarrow xvec \sharp* (M \succeq N)$
 and $[\![Xs \sharp* M;\ Xs \sharp* N]\!] \Longrightarrow Xs \sharp* (M \succeq N)$
 by(*auto simp add*: *fresh-star-def*)

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

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

shows $x \sharp M \preceq N$
 using *assms chanOutConSupp*
 by(*auto simp add*: *fresh-def*)

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

shows $[\![xvec \sharp* M;\ xvec \sharp* N]\!] \Longrightarrow xvec \sharp* (M \preceq N)$

and $[\![Xs \sharp* M;\ Xs \sharp* N]\!] \Longrightarrow Xs \sharp* (M \preceq N)$
by(*auto simp add*: *fresh-star-def*)

lemma *chanOutConClosed*:
fixes $\Psi :: 'b$
and $M :: 'a$
and $N :: 'a$
and $p :: name\ prm$

assumes $\Psi \vdash M \preceq N$

shows $(p \cdot \Psi) \vdash (p \cdot M) \preceq (p \cdot N)$
proof −
from ‹$\Psi \vdash M \preceq N$› **have** $(p \cdot \Psi) \vdash p \cdot (M \preceq N)$
by(*rule statClosed*)
then show *?thesis* **by**(*auto simp add*: *eqvts*)
qed

lemma *chanInConClosed*:
fixes $\Psi :: 'b$
and $M :: 'a$
and $N :: 'a$
and $p :: name\ prm$

assumes $\Psi \vdash M \succeq N$

shows $(p \cdot \Psi) \vdash (p \cdot M) \succeq (p \cdot N)$
proof −
from ‹$\Psi \vdash M \succeq N$› **have** $(p \cdot \Psi) \vdash p \cdot (M \succeq N)$
by(*rule statClosed*)
then show *?thesis* **by**(*auto simp add*: *eqvts*)
qed

end

locale *assertion* = *assertionAux SCompose SImp SBottom SChanEq SOutCon SInCon* + *assertion 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*
and *SOutCon* :: *'a*::*fs-name* ⇒ *'a* ⇒ *'c*
and *SInCon* :: *'a*::*fs-name* ⇒ *'a* ⇒ *'c* +

assumes *chanOutConSupp*: *SImp* Ψ (*SOutCon M N*) ⟹ (((*supp N*)::*name set*) ⊆ ((*supp M*)::*name set*))
and *chanInConSupp*: *SImp* Ψ (*SInCon N M*) ⟹ (((*supp N*)::*name set*) ⊆ ((*supp M*)::*name set*))

begin

notation *SOutCon* (‹- $\preceq$ -› [*90*, *90*] *90*)
notation *SInCon* (‹- $\succeq$ -› [*90*, *90*] *90*)

end

end
theory *Semantics*
imports *Broadcast-Chain Broadcast-Frame*
begin

This file is a (heavily modified) variant of the theory *Psi_Calculi.Semantics* from [1]. The nominal datatypes *('a,'b,'c) residual* and *'a action* have been extended with constructors for broadcast input and output. This leads to a different semantics.

nominal-datatype ('*a*, '*b*, '*c*) *boundOutput* =
BOut '*a*::*fs-name* ('*a*, '*b*::*fs-name*, '*c*::*fs-name*) *psi* (‹- $\prec''$ -› [*110*, *110*] *110*)
| *BStep* «*name*» ('*a*, '*b*, '*c*) *boundOutput* (‹($\langle\!|\nu$-$|\!\rangle$-› [*110*, *110*] *110*)

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

abbreviation
BOresChainJudge (‹($\langle\!|\nu$*-$|\!\rangle$-› [*80*, *80*] *80*) **where** $\langle\!|\nu$*xvec$|\!\rangle$*B* ≡ *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* ∙ ($\langle\!|\nu$*xvec$|\!\rangle$*B*) = $\langle\!|\nu$*(*perm* ∙ *xvec*)$|\!\rangle$(*perm* ∙ *B*)
by(*induct xvec*) *auto*

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 ($\langle\!|\nu$*xvec$|\!\rangle$*N* $\prec'$ *P* = *N'* $\prec'$ *P'*) = (*xvec* = [] ∧ *N* = *N'* ∧ *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 $((\!|\nu{*}xvec|\!)B = (\!|\nu{*}xvec|\!)B') = (B = B')$
by(*induct xvec*) (*auto simp add*: *boundOutput.inject alpha*)

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 $(xvec \sharp* (N \prec' P)) = ((xvec \sharp* N) \wedge (xvec \sharp* P))$
by(*auto simp add*: *fresh-star-def*)

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

shows $(x \sharp (\!|\nu{*}xvec|\!)B)) = (x \in set\ xvec \vee x \sharp B)$
by (*induct xvec*) (*simp-all add*: *abs-fresh*)

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 $Xs \sharp* ((\!|\nu{*}xvec|\!)B) = (\forall x{\in}Xs.\ x \in set\ xvec \vee x \sharp B)$
and $yvec \sharp* ((\!|\nu{*}xvec|\!)B) = (\forall x{\in}(set\ yvec).\ x \in set\ xvec \vee x \sharp B)$
and $Xs \sharp* ((\!|\nu x|\!)B) = Xs \sharp* [x].B$
and $xvec \sharp* ((\!|\nu x|\!)B) = xvec \sharp* [x].B$
by(*simp add*: *fresh-star-def boundOutputFresh*)+

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

shows $(supp((\!|\nu{*}xvec|\!)B)::name\ set) = (supp\ B) - (supp\ xvec)$
by(*induct xvec*)
(*auto simp add*: *boundOutput.supp supp-list-nil supp-list-cons abs-supp supp-atm*)

lemma *boundOutputFreshSimps*[*simp*]:
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 $Xs \sharp* xvec \Longrightarrow (Xs \sharp* (\langle\!|\nu* xvec|\!\rangle B)) = (Xs \sharp* B)$
and $yvec \sharp* xvec \Longrightarrow yvec \sharp* (\langle\!|\nu* xvec|\!\rangle B) = yvec \sharp* B$
and $xvec \sharp* (\langle\!|\nu* xvec|\!\rangle B)$
and $x \sharp xvec \Longrightarrow x \sharp \langle\!|\nu* xvec|\!\rangle B = x \sharp B$
apply(*simp add*: *boundOutputFreshSet*) **apply**(*force simp add*: *fresh-star-def name-list-supp fresh-def*)
apply(*simp add*: *boundOutputFreshSet*) **apply**(*force simp add*: *fresh-star-def name-list-supp fresh-def*)
apply(*simp add*: *boundOutputFreshSet*)
by(*simp add*: *BOresChainSupp fresh-def*)

lemma *boundOutputChainAlpha*:
fixes p :: *name prm*
and *xvec* :: *name list*
and B :: ('*a*::*fs-name*, '*b*::*fs-name*, '*c*::*fs-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 −
note *pt-name-inst at-name-inst S*
moreover from ‹$(set\ xvec) \subseteq (set\ yvec)$› **have** *set xvec* $\sharp* (\langle\!|\nu* yvec|\!\rangle B)$
by(*force simp add*: *boundOutputFreshSet*)
moreover from *xvecFreshB* ‹$(set\ xvec) \subseteq (set\ yvec)$› **have** *set* $(p \cdot xvec) \sharp* (\langle\!|\nu* yvec|\!\rangle B)$
by (*simp add*: *boundOutputFreshSet*) (*simp add*: *fresh-star-def*)
ultimately have $(\langle\!|\nu* yvec|\!\rangle B) = p \cdot (\langle\!|\nu* yvec|\!\rangle B)$
by (*rule pt-freshs-freshs* [*symmetric*])
then show *?thesis* **by**(*simp add*: *eqvts*)
qed

lemma *boundOutputChainAlpha'*:
fixes p :: *name prm*
and *xvec* :: *name list*
and B :: ('*a*::*fs-name*, '*b*::*fs-name*, '*c*::*fs-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 −

note *pt-name-inst at-name-inst S* ‹*yvec* ♯* (⦇*ν***zvec*⦈*B*)›
moreover from *xvecFreshB* **have** *set* (*xvec*) ♯* (⦇*ν***zvec*⦈*B*)
by (*simp add*: *boundOutputFreshSet*) (*simp add*: *fresh-star-def*)
ultimately have (⦇*ν***zvec*⦈*B*) = *p* ∙ (⦇*ν***zvec*⦈*B*)
by(*auto intro*: *pt-freshs-freshs* [*symmetric*])
then show *?thesis* **by**(*simp add*: *eqvts*)
qed

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* ∙ *xvec*) ♯* *M*
and (*p* ∙ *xvec*) ♯* *P*
and *set p* ⊆ *set xvec* × *set* (*p* ∙ *xvec*)
and (*set xvec*) ⊆ (*set yvec*)

shows (⦇*ν***yvec*⦈*M* ≺' *P*) = (⦇*ν**(*p* ∙ *yvec*)⦈(*p* ∙ *M*) ≺' (*p* ∙ *P*))
using *assms*
by(*subst boundOutputChainAlpha*) *auto*

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* ♯ *N*
and *y* ♯ *P*
and *x* ∈ (*set xvec*)

shows ⦇*ν***xvec*⦈*N* ≺' *P* = ⦇*ν**([(*x*, *y*)] ∙ *xvec*)⦈([(*x* ,*y*)] ∙ *N*) ≺' ([(*x*, *y*)] ∙ *P*)
proof(*cases x*=*y*)
assume *x*=*y*
then show *?thesis* **by** *simp*
next
assume *x* ≠ *y*
with *assms* **show** *?thesis*
by(*auto simp add*: *calc-atm intro*: *boundOutputChainAlpha''*[**where** *xvec*=[*x*]])
qed

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 $(\!|\nu x|\!)B = (\!|\nu y|\!)([(x, y)] \cdot B)$
using *assms*
by(*auto simp add*: *boundOutput.inject alpha fresh-left calc-atm*)

lemma *boundOutputEqFresh*:
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 $x \sharp B$

shows $y \sharp C$
using *assms*
by(*auto simp add*: *boundOutput.inject alpha fresh-left calc-atm*)

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 $y \in supp\ C$
using *assms*
apply(*clarsimp simp add*: *boundOutput.inject alpha fresh-left calc-atm*)
apply(*drule pt-set-bij2*[**where** *pi*=[(*x*, *y*)], *OF pt-name-inst*, *OF at-name-inst*])
by(*auto simp add*: *eqvts calc-atm*)

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 −
obtain n **where** $n = length\ xvec$ **by** *auto*

```
  with assms show ?thesis
  proof(induct n arbitrary: xvec yvec B B')
    case(0 xvec yvec B B')
    have Eq: (|ν∗xvec|)B = (|ν∗yvec|)B' by fact
    from ‹0 = length xvec› have xvec = [] by auto
    moreover with ‹length xvec = length yvec› have yvec = []
      by(cases yvec) auto
    ultimately show ?case using Eq
      by(simp add: boundOutput.inject)
  next
    case(Suc n xvec yvec B B')
    from ‹Suc n = length xvec›
    obtain x xvec' where xvec = x#xvec' and length xvec' = n
      by(cases xvec) auto
   from ‹(|ν∗xvec|)B = (|ν∗yvec|)B'› ‹xvec = x # xvec'› ‹length xvec = length yvec›
    obtain y yvec' where (|ν∗(x#xvec')|)B = (|ν∗(y#yvec')|)B'
      and yvec = y#yvec' and length xvec' = length yvec'
      by(cases yvec) auto
    then have EQ: (|νx|)((|ν∗xvec'|)B) = (|νy|)((|ν∗yvec'|)B')
      by simp
    from ‹xvec = x#xvec'› ‹yvec=y#yvec'› ‹xvec ♯∗ yvec›
    have x ≠ y and xvec' ♯∗ yvec' and x ♯ yvec' and y ♯ xvec'
      by auto
   have IH: ⋀xvec yvec B B'. [[(|ν∗xvec|)(B::('a::fs-name, 'b::fs-name, 'c::fs-name)
boundOutput) = (|ν∗yvec|)B'; xvec ♯∗ yvec; length xvec = length yvec; n = length
xvec]] ⟹ ∃p. (set p) ⊆ (set xvec) × (set yvec) ∧ distinctPerm p ∧  B = p • B' ∧
set(map fst p) ⊆ supp B ∧ xvec ♯∗ B' ∧ yvec ♯∗ B
      by fact
    from EQ ‹x ≠ y› have EQ': (|ν∗xvec'|)B = ([(x, y)] • ((|ν∗yvec'|)B'))
      and xFreshB': x ♯ ((|ν∗yvec'|)B')
      and yFreshB: y ♯ ((|ν∗xvec'|)B)
      by(metis boundOutput.inject alpha)+
    from xFreshB' ‹x ♯ yvec'› have x ♯ B'
      by(auto simp add: boundOutputFresh) (simp add: fresh-def name-list-supp)+
    from yFreshB ‹y ♯ xvec'› have y ♯ B
      by(auto simp add: boundOutputFresh) (simp add: fresh-def name-list-supp)+
    show ?case
    proof(cases x ♯ (|ν∗xvec'|)B)
      assume xFreshB: x ♯ (|ν∗xvec'|)B
      with EQ have yFreshB': y ♯ (|ν∗yvec'|)B'
        by(rule boundOutputEqFresh)
      with xFreshB' EQ' have (|ν∗xvec'|)B = (|ν∗yvec'|)B'
        by(simp)
      with ‹xvec' ♯∗ yvec'› ‹length xvec' = length yvec'› ‹length xvec' = n› IH
       obtain p where S: (set p) ⊆ (set xvec') × (set yvec') and distinctPerm p
and B = p • B'
        and set(map fst p) ⊆ supp B and xvec' ♯∗ B'  and yvec' ♯∗ B
        by blast
      from S have (set p) ⊆ set(x#xvec') × set(y#yvec') by auto
```

moreover note ‹*xvec* = *x*#*xvec*′› ‹*yvec*=*y*#*yvec*′› ‹*distinctPerm p*› ‹*B* = *p* ∙ *B*′›
‹*xvec*′ ♯* *B*′› ‹*x* ♯ *B*′› ‹*x* ♯ *B*′› ‹*yvec*′ ♯* *B*› ‹*y* ♯ *B*› ‹*set*(*map fst p*) ⊆ *supp B*›

ultimately show *?case* **by** *auto*
next
assume ¬(*x* ♯ (⦇*ν*∗*xvec*′⦈*B*)
then have *xSuppB*: *x* ∈ *supp*(⦇*ν*∗*xvec*′⦈*B*)
by(*simp add*: *fresh-def*)
with *EQ* **have** *ySuppB*′: *y* ∈ *supp* (⦇*ν*∗*yvec*′⦈*B*′)
by(*rule boundOutputEqSupp*)
then have *y* ♯ *yvec*′
by(*induct yvec*′) (*auto simp add*: *boundOutput.supp abs-supp*)
with ‹*x* ♯ *yvec*′› *EQ*′ **have** ⦇*ν*∗*xvec*′⦈*B* = ⦇*ν*∗*yvec*′⦈([(*x*, *y*)] ∙ *B*′)
by(*simp add*: *eqvts*)
with ‹*xvec*′ ♯* *yvec*′› ‹*length xvec*′ = *length yvec*′› ‹*length xvec*′ = *n*› *IH*
obtain *p* **where** *S*: (*set p*) ⊆ (*set xvec*′) × (*set yvec*′) **and** *distinctPerm p* **and** *B* = *p* ∙ [(*x*, *y*)] ∙ *B*′
and *set*(*map fst p*) ⊆ *supp B* **and** *xvec*′ ♯* ([(*x*, *y*)] ∙ *B*′) **and** *yvec*′ ♯* *B*
by *blast*

from *xSuppB* **have** *x* ♯ *xvec*′
by(*induct xvec*′) (*auto simp add*: *boundOutput.supp abs-supp*)
with ‹*x* ♯ *yvec*′› ‹*y* ♯ *xvec*′› ‹*y* ♯ *yvec*′› *S* **have** *x* ♯ *p* **and** *y* ♯ *p*
apply(*induct p*)
by(*auto simp add*: *name-list-supp*) (*auto simp add*: *fresh-def*)
from *S* **have** (*set* ((*x*, *y*)#*p*)) ⊆ (*set*(*x*#*xvec*′)) × (*set*(*y*#*yvec*′))
by *force*
moreover from ‹*x* ≠ *y*› ‹*x* ♯ *p*› ‹*y* ♯ *p*› *S* ‹*distinctPerm p*›
have *distinctPerm*((*x*,*y*)#*p*) **by** *simp*
moreover from ‹*B* = *p* ∙ [(*x*, *y*)] ∙ *B*′› ‹*x* ♯ *p*› ‹*y* ♯ *p*› **have** *B* = [(*x*, *y*)] ∙ *p* ∙ *B*′
by(*subst perm-compose*) *simp*
then have *B* = ((*x*, *y*)#*p*) ∙ *B*′ **by** *simp*
moreover from ‹*xvec*′ ♯* ([(*x*, *y*)] ∙ *B*′)› **have** ([(*x*, *y*)] ∙ *xvec*′) ♯* ([(*x*, *y*)] ∙ [(*x*, *y*)] ∙ *B*′)
by(*simp only*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹*x* ♯ *xvec*′› ‹*y* ♯ *xvec*′› ‹*x* ♯ *B*′› **have** (*x*#*xvec*′) ♯* *B*′ **by** *simp*
moreover from ‹*y* ♯ *B*› ‹*yvec*′ ♯* *B*› **have** (*y*#*yvec*′) ♯* *B* **by** *simp*
moreover from ‹*set*(*map fst p*) ⊆ *supp B*› *xSuppB* ‹*x* ♯ *xvec*′›
have *set*(*map fst* ((*x*, *y*)#*p*)) ⊆ *supp B*
by(*simp add*: *BOresChainSupp*)
ultimately show *?case* **using** ‹*xvec*=*x*#*xvec*′› ‹*yvec*=*y*#*yvec*′›
by *metis*
qed
qed
qed

lemma *boundOutputChainEqLength*:
  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::fs-name*
    and *Q*   :: (*'a*, *'b::fs-name*, *'c::fs-name*) *psi*

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

shows *length xvec* = *length yvec*
proof −
  obtain *n* **where** *n* = *length xvec* **by** *auto*
  with *assms* **show** *?thesis*
  proof(*induct n arbitrary*: *xvec yvec M P N Q*)
    case(*0 xvec yvec M P N Q*)
    from ‹*0* = *length xvec*› **have** *xvec* = [] **by** *auto*
    moreover with ‹$(\!|\nu*xvec|\!)M \prec' P = (\!|\nu*yvec|\!)N \prec' Q$› **have** *yvec* = []
      by(*cases yvec*) *auto*
    ultimately show *?case* **by** *simp*
  next
    case(*Suc n xvec yvec M P N Q*)
    from ‹*Suc n* = *length xvec*›
    obtain *x xvec'* **where** *xvec* = *x*#*xvec'* **and** *length xvec'* = *n*
      by(*cases xvec*) *auto*
    from ‹$(\!|\nu*xvec|\!)M \prec' P = (\!|\nu*yvec|\!)N \prec' Q$› ‹*xvec* = *x* # *xvec'*›
    obtain *y yvec'* **where** $(\!|\nu*(x\#xvec')|\!)M \prec' P = (\!|\nu*(y\#yvec')|\!)N \prec' Q$
      and *yvec* = *y*#*yvec'*
      by(*cases yvec*) *auto*
    then have *EQ*: $(\!|\nu x|\!)((\!|\nu*xvec'|\!)M \prec' P) = (\!|\nu y|\!)((\!|\nu*yvec'|\!)N \prec' Q)$
      by *simp*
    have *IH*: $\bigwedge$*xvec yvec M P N Q*. $[\![(\!|\nu*xvec|\!)M \prec' P = (\!|\nu*yvec|\!)N \prec' (Q::('a, 'b, 'c)\ psi);\ n = length\ xvec]\!] \Longrightarrow$ *length xvec* = *length yvec*
      by *fact*
    show *?case*
    proof(*cases x* = *y*)
      assume *x* = *y*
      with *EQ* **have** $(\!|\nu*xvec'|\!)M \prec' P = (\!|\nu*yvec'|\!)N \prec' Q$
        by(*simp add*: *alpha boundOutput.inject*)
      with *IH* ‹*length xvec'* = *n*› **have** *length xvec'* = *length yvec'*
        by *blast*
      with ‹*xvec* = *x*#*xvec'*› ‹*yvec*=*y*#*yvec'*›
      show *?case* **by** *simp*
    next
      assume *x* ≠ *y*
      with *EQ* **have** $(\!|\nu*xvec'|\!)M \prec' P = [(x, y)] \cdot (\!|\nu*yvec'|\!)N \prec' Q$
        by(*simp add*: *alpha boundOutput.inject*)
      then have $(\!|\nu*xvec'|\!)M \prec' P = (\!|\nu*([(x, y)] \cdot yvec')|\!)([(x, y)] \cdot N) \prec' ([(x, y)] \cdot Q)$

```
      by(simp add: eqvts)
    with IH ‹length xvec' = n› have length xvec' = length ([(x, y)] • yvec')
      by blast
    then have length xvec' = length yvec'
      by simp
    with ‹xvec = x#xvec'› ‹yvec=y#yvec'›
    show ?case by simp
  qed
 qed
qed

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 (⦇ν*xvec⦈)M ≺' P = (⦇ν*yvec⦈)N ≺' Q
  and   xvec ♯* yvec

shows ∃ p. (set p) ⊆ (set xvec) × set (yvec) ∧ distinctPerm p ∧  M = p • N ∧  P
= p • Q ∧ xvec ♯* N ∧ xvec ♯* Q ∧ yvec ♯* M ∧ yvec ♯* P
  using assms boundOutputChainEq boundOutputChainEqLength by fastforce

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 (⦇ν*xvec⦈)M ≺' P = (⦇ν*yvec⦈)N ≺' Q
  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 and N = p • M and Q = p • P and xvec ♯* N and xvec ♯* Q
and (p • xvec) ♯* M and (p • xvec) ♯* P
proof −

  assume ⋀p. ⟦set p ⊆ set xvec × set (p • xvec); distinctPerm p; yvec = p • xvec;
N = p • M; Q = p • P; xvec ♯* N; xvec ♯* Q; (p • xvec) ♯* M; (p • xvec) ♯* P⟧
⟹ thesis

  moreover obtain n where n = length xvec by auto
```

with *assms* **have** $\exists p.\ (set\ p) \subseteq (set\ xvec) \times set\ (yvec) \wedge distinctPerm\ p \wedge\ yvec = p \cdot xvec \wedge N = p \cdot M \wedge Q = p \cdot P \wedge xvec \sharp* N \wedge xvec \sharp* Q \wedge (p \cdot xvec) \sharp* M \wedge (p \cdot xvec) \sharp* P$
proof(*induct n arbitrary*: *xvec yvec M P N Q*)
case(*0 xvec yvec M P N Q*)
have *Eq*: $(\!|\nu*xvec|\!)M \prec' P = (\!|\nu*yvec|\!)N \prec' Q$ **by** *fact*
from ‹*0* = *length xvec*› **have** *xvec* = [] **by** *auto*
moreover with *Eq* **have** *yvec* = []
by(*cases yvec*) *auto*
ultimately show *?case* **using** *Eq*
by(*simp add*: *boundOutput.inject*)
next
case(*Suc n xvec yvec M P N Q*)
from ‹*Suc n* = *length xvec*›
obtain *x xvec'* **where** *xvec* = *x*#*xvec'* **and** *length xvec'* = *n*
by(*cases xvec*) *auto*
from ‹$(\!|\nu*xvec|\!)M \prec' P = (\!|\nu*yvec|\!)N \prec' Q$› ‹*xvec* = *x* # *xvec'*›
obtain *y yvec'* **where** $(\!|\nu*(x\#xvec')|\!)M \prec' P = (\!|\nu*(y\#yvec')|\!)N \prec' Q$
and *yvec* = *y*#*yvec'*
by(*cases yvec*) *auto*
then have *EQ*: $(\!|\nu x|\!)((\!|\nu*xvec'|\!)M \prec' P) = (\!|\nu y|\!)((\!|\nu*yvec'|\!)N \prec' Q)$
by *simp*
from ‹*xvec* = *x*#*xvec'*› ‹*yvec*=*y*#*yvec'*› ‹*xvec* $\sharp*$ *yvec*›
have $x \neq y$ **and** *xvec'* $\sharp*$ *yvec'* **and** $x \sharp yvec'$ **and** $y \sharp xvec'$
by *auto*
from ‹*distinct xvec*› ‹*distinct yvec*› ‹*xvec*=*x*#*xvec'*› ‹*yvec*=*y*#*yvec'*› **have** $x \sharp xvec'$ **and** $y \sharp yvec'$ **and** *distinct xvec'* **and** *distinct yvec'*
by *simp+*
have *IH*: $\bigwedge xvec\ yvec\ M\ P\ N\ Q.\ [\![(\!|\nu*xvec|\!)(M::'a) \prec' (P::('a, 'b, 'c)\ psi) = (\!|\nu*yvec|\!)N \prec' Q; xvec \sharp* yvec; distinct\ xvec; distinct\ yvec; n = length\ xvec]\!] \Longrightarrow \exists p.\ (set\ p) \subseteq (set\ xvec) \times (set\ yvec) \wedge distinctPerm\ p \wedge\ yvec = p \cdot xvec \wedge N = p \cdot M \wedge Q = p \cdot P \wedge xvec \sharp* N \wedge xvec \sharp* Q \wedge (p \cdot xvec) \sharp* M \wedge (p \cdot xvec) \sharp* P$
by *fact*
from *EQ* ‹$x \neq y$› ‹$x \sharp yvec'$› ‹$y \sharp yvec'$› ‹$y \sharp xvec'$› ‹$x \sharp xvec'$› **have** $(\!|\nu*xvec'|\!)M \prec' P = (\!|\nu*yvec'|\!)([(x, y)] \cdot N) \prec' ([(x, y)] \cdot Q)$ **and** $x \sharp N$ **and** $x \sharp Q$ **and** $y \sharp M$ **and** $y \sharp P$
apply −
apply(*simp add*: *boundOutput.inject alpha eqvts*)
apply(*simp add*: *boundOutput.inject alpha eqvts*)
apply(*simp add*: *boundOutput.inject alpha eqvts*)
by(*simp add*: *boundOutput.inject alpha' eqvts*)+
with ‹*xvec'* $\sharp*$ *yvec'*› ‹*distinct xvec'*› ‹*distinct yvec'*› ‹*length xvec'* = *n*› *IH*
obtain *p* **where** *S*: (*set p*) $\subseteq$ (*set xvec'*) $\times$ (*set yvec'*) **and** *distinctPerm p* **and** *yvec'* = *p* $\cdot$ *xvec'* **and** $([(x, y)] \cdot N) = p \cdot M$ **and** $([(x, y)] \cdot Q) = p \cdot P$ **and** *xvec'* $\sharp*$ $([(x, y)] \cdot N)$ **and** *xvec'* $\sharp*$ $([(x, y)] \cdot Q)$ **and** *yvec'* $\sharp*$ *M* **and** *yvec'* $\sharp*$ *P*
by *metis*
from *S* **have** $set((x, y)\#p) \subseteq set(x\#xvec') \times set(y\#yvec')$ **by** *auto*
moreover from ‹$x \sharp xvec'$› ‹$x \sharp yvec'$› ‹$y \sharp xvec'$› ‹$y \sharp yvec'$› *S* **have** $x \sharp p$ **and** $y \sharp p$

apply(*induct p*)
by(*auto simp add*: *fresh-prod name-list-supp*) (*auto simp add*: *fresh-def*)

with *S* ‹*distinctPerm p*› ‹*x* ≠ *y*› **have** *distinctPerm*((*x*, *y*)#*p*) **by** *auto*
moreover from ‹*yvec'* = *p* ∙ *xvec'*› ‹*x* ♯ *p*› ‹*y* ♯ *p*› ‹*x* ♯ *xvec'*› ‹*y* ♯ *xvec'*› **have** (*y*#*yvec'*) = ((*x*, *y*)#*p*) ∙ (*x*#*xvec'*)
by(*simp add*: *eqvts calc-atm perm-compose freshChainSimps*)
moreover from ‹([(*x*, *y*)] ∙ *N*) = *p* ∙ *M*›
have ([(*x*, *y*)] ∙ [(*x*, *y*)] ∙ *N*) = [(*x*, *y*)] ∙ *p* ∙ *M*
by(*simp add*: *pt-bij*)
then have *N* = ((*x*, *y*)#*p*) ∙ *M* **by** *simp*
moreover from ‹([(*x*, *y*)] ∙ *Q*) = *p* ∙ *P*›
have ([(*x*, *y*)] ∙ [(*x*, *y*)] ∙ *Q*) = [(*x*, *y*)] ∙ *p* ∙ *P*
by(*simp add*: *pt-bij*)
then have *Q* = ((*x*, *y*)#*p*) ∙ *P* **by** *simp*
moreover from ‹*xvec'* ♯* ([(*x*, *y*)] ∙ *N*)› **have** ([(*x*, *y*)] ∙ *xvec'*) ♯* ([(*x*, *y*)] ∙ [(*x*, *y*)] ∙ *N*)
by(*subst fresh-star-bij*)
with ‹*x* ♯ *xvec'*› ‹*y* ♯ *xvec'*› **have** *xvec'* ♯* *N* **by** *simp*
with ‹*x* ♯ *N*› **have** (*x*#*xvec'*) ♯* *N* **by** *simp*
moreover from ‹*xvec'* ♯* ([(*x*, *y*)] ∙ *Q*)› **have** ([(*x*, *y*)] ∙ *xvec'*) ♯* ([(*x*, *y*)] ∙ [(*x*, *y*)] ∙ *Q*)
by(*subst fresh-star-bij*)
with ‹*x* ♯ *xvec'*› ‹*y* ♯ *xvec'*› **have** *xvec'* ♯* *Q* **by** *simp*
with ‹*x* ♯ *Q*› **have** (*x*#*xvec'*) ♯* *Q* **by** *simp*
moreover from ‹*y* ♯ *M*› ‹*yvec'* ♯* *M*› **have** (*y*#*yvec'*) ♯* *M* **by** *simp*
moreover from ‹*y* ♯ *P*› ‹*yvec'* ♯* *P*› **have** (*y*#*yvec'*) ♯* *P* **by** *simp*
ultimately show *?case* **using** ‹*xvec*=*x*#*xvec'*› ‹*yvec*=*y*#*yvec'*›
by *metis*
qed
ultimately show *?thesis* **by** *blast*
qed

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*: (|ν*x*|)((|ν*xvec*|)*M* ≺' *P*) = (|ν*y*|)((|ν*yvec*|)*N* ≺' *Q*)
and *x* ≠ *y*
and *x* ♯ *yvec*
and *x* ♯ *xvec*
and *y* ♯ *xvec*
and *y* ♯ *yvec*

and $xvec \sharp* yvec$
and $x \in supp\ M$

shows $y \in supp\ N$
proof −
from Eq ‹$x \neq y$› ‹$x \sharp yvec$› ‹$y \sharp yvec$› **have** $(\!|\nu* xvec|\!)M \prec' P = (\!|\nu* yvec|\!)([(x, y)] \cdot N) \prec' ([(x, y)] \cdot Q)$
by(*simp add*: *boundOutput.inject alpha eqvts*)
then obtain p **where** S: $set\ p \subseteq set\ xvec \times set\ yvec$ **and** $M = p \cdot [(x, y)] \cdot N$ **and** $distinctPerm\ p$ **using** ‹$xvec \sharp* yvec$›
by(*blast dest*: *boundOutputChainEq'*)
with ‹$x \in supp\ M$› **have** $x \in supp(p \cdot [(x, y)] \cdot N)$ **by** *simp*
then have $(p \cdot x) \in p \cdot supp(p \cdot [(x, y)] \cdot N)$
by(*simp add*: *pt-set-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$x \sharp xvec$› ‹$x \sharp yvec$› S ‹$distinctPerm\ p$› **have** $x \in supp([(x, y)] \cdot N)$
by(*simp add*: *eqvts*)
then have $([(x, y)] \cdot x) \in ([(x, y)] \cdot (supp([(x, y)] \cdot N)))$
by(*simp add*: *pt-set-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$x \neq y$› **show** *?thesis* **by**(*simp add*: *calc-atm eqvts*)
qed

lemma *boundOutputChainOpenIH*:
fixes $xvec :: name\ list$
and $x :: name$
and $B :: ('a::fs\text{-}name, 'b::fs\text{-}name, 'c::fs\text{-}name)\ boundOutput$
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 $(\!|\nu* xvec|\!)B = (\!|\nu* yvec|\!)([(x, y)] \cdot B')$
using *assms*
proof(*induct n==length xvec arbitrary*: *xvec yvec y B' rule*: *nat.induct*)
case(*zero xvec yvec y B'*)
have $0 = length\ xvec$ **and** $length\ xvec = length\ yvec$ **by** *fact+*
moreover have $(\!|\nu* xvec|\!)(\!|\nu x|\!)B = (\!|\nu* yvec|\!)(\!|\nu y|\!)B'$ **by** *fact*
ultimately show *?case* **by**(*auto simp add*: *boundOutput.inject alpha*)
next
case(*Suc n xvec yvec y B'*)
have L: $length\ xvec = length\ yvec$ **and** $Suc\ n = length\ xvec$ **by** *fact+*
then obtain $x'\ xvec'\ y'\ yvec'$ **where** xEq: $xvec = x'\#xvec'$ **and** yEq: $yvec = y'\#yvec'$
and L': $length\ xvec' = length\ yvec'$
by(*cases xvec*, *auto*, *cases yvec*, *auto*)

have *xFreshB'*: $x \sharp B'$ **by** *fact*
have $x \sharp xvec$ **and** $x \sharp yvec$ **by** *fact+*
with *xEq yEq* **have** *xineqx'*: $x \neq x'$ **and** *xFreshxvec'*: $x \sharp xvec'$
and *xineqy'*: $x \neq y'$ **and** *xFreshyvec'*: $x \sharp yvec'$
by *simp+*
have $(\!|\nu{*}xvec|\!)(\!|\nu x|\!)B = (\!|\nu{*}yvec|\!)(\!|\nu y|\!)B'$ **by** *fact*
with *xEq yEq* **have** *Eq*: $(\!|\nu x'|\!)((\!|\nu{*}xvec'|\!)(\!|\nu x|\!)B) = (\!|\nu y'|\!)((\!|\nu{*}yvec'|\!)(\!|\nu y|\!)B')$ **by** *simp*
have *IH*: $\bigwedge xvec\ yvec\ y\ B'.$
$[\![n = length\ xvec;\ (\!|\nu{*}xvec|\!)(\!|\nu x|\!)B = (\!|\nu{*}yvec|\!)(\!|\nu y|\!)B';\ length\ xvec = length\ yvec;\ x \sharp B';\ x \sharp xvec;\ x \sharp yvec]\!]$
$\Longrightarrow (\!|\nu{*}xvec|\!)B = (\!|\nu{*}yvec|\!)([(x,\ y)] \cdot B')$ **by** *fact*
have *Suc n = length xvec* **by** *fact*
with *xEq* **have** *L''*: *n = length xvec'* **by** *simp*
have $(\!|\nu x'|\!)((\!|\nu{*}xvec'|\!)B) = (\!|\nu y'|\!)((\!|\nu{*}yvec'|\!)([(x,\ y)] \cdot B'))$
proof(*cases x'=y'*)
assume *x'eqy'*: $x' = y'$
with *Eq* **have** $(\!|\nu{*}xvec'|\!)(\!|\nu x|\!)B = (\!|\nu{*}yvec'|\!)(\!|\nu y|\!)B'$ **by**(*simp add*: *boundOutput.inject alpha*)
then have $(\!|\nu{*}xvec'|\!)B = (\!|\nu{*}yvec'|\!)([(x,\ y)] \cdot B')$ **using** *L' xFreshB' xFreshxvec' xFreshyvec' L''* **by**(*metis IH*)
with *x'eqy'* **show** *?thesis* **by**(*simp add*: *boundOutput.inject alpha*)
next
assume *x'ineqy'*: $x' \neq y'$
with *Eq* **have** *Eq'*: $(\!|\nu{*}xvec'|\!)(\!|\nu x|\!)B = (\!|\nu{*}([(x',\ y')] \cdot yvec')|\!)(\!|\nu([(x',\ y')] \cdot y)|\!)([(x',\ y')] \cdot B')$
and *x'FreshB'*: $x' \sharp (\!|\nu{*}yvec'|\!)(\!|\nu y|\!)B'$
by(*simp add*: *boundOutput.inject alpha eqvts*)+
from *L'* **have** *length xvec' = length* $([(x',\ y')] \cdot yvec')$ **by** *simp*
moreover from *xineqx' xineqy' xFreshB'* **have** $x \sharp [(x',\ y')] \cdot B'$ **by**(*simp add*: *fresh-left calc-atm*)
moreover from *xineqx' xineqy' xFreshyvec'* **have** $x \sharp [(x',\ y')] \cdot yvec'$ **by**(*simp add*: *fresh-left calc-atm*)
ultimately have $(\!|\nu{*}xvec'|\!)B = (\!|\nu{*}([(x',\ y')] \cdot yvec')|\!)([(x,\ ([(x',\ y')] \cdot y))] \cdot [(x',\ y')] \cdot B')$ **using** *Eq' xFreshxvec' L''*
by(*metis IH*)
moreover from *x'FreshB'* **have** $x' \sharp (\!|\nu{*}yvec'|\!)([(x,\ y)] \cdot B')$
proof(*cases* $x' \sharp yvec'$)
assume $x' \sharp yvec'$
with *x'FreshB'* **have** *x'FreshB'*: $x' \sharp (\!|\nu y|\!)B'$
by(*simp add*: *fresh-def BOresChainSupp*)
show *?thesis*
proof(*cases x'=y*)
assume *x'eqy*: $x' = y$
show *?thesis*
proof(*cases x=y*)
assume *x=y*
with *xFreshB' x'eqy* **show** *?thesis* **by**(*simp add*: *BOresChainSupp fresh-def*)
next

```
        assume x ≠ y
        with ‹x ♯ B'› have y ♯ [(x, y)] • B' by(simp add: fresh-left calc-atm)
        with x'eqy show ?thesis by(simp add: BOresChainSupp fresh-def)
       qed
      next
       assume x'ineqy: x' ≠ y
       with x'FreshB' have x' ♯ B' by(simp add: abs-fresh)
      with xineqx' x'ineqy have x' ♯ ([(x, y)] • B') by(simp add: fresh-left calc-atm)
       then show ?thesis by(simp add: BOresChainSupp fresh-def)
      qed
    next
      assume ¬x' ♯ yvec'
      then show ?thesis by(simp add: BOresChainSupp fresh-def)
    qed
    ultimately show ?thesis using x'ineqy' xineqx' xineqy'
      apply(simp add: boundOutput.inject alpha eqvts)
      apply(subst perm-compose[of [(x', y')]])
      by(simp add: calc-atm)
  qed
  with xEq yEq show ?case by simp
qed

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 (⌊ν*xvec⌋)M ≺' P = (⌊ν*yvec⌋)N ≺' (Q ∥ R)
  and   xvec ♯* R
  and   yvec ♯* R

obtains T where P = T ∥ R and (⌊ν*xvec⌋)M ≺' T = (⌊ν*yvec⌋)N ≺' Q
proof -
  assume ⋀T. ⟦P = T ∥ R; (⌊ν*xvec⌋)M ≺' T = (⌊ν*yvec⌋)N ≺' Q⟧ ⟹ thesis
  moreover obtain n where n = length xvec by auto
  with assms have ∃ T. P = T ∥ R ∧ (⌊ν*xvec⌋)M ≺' T = (⌊ν*yvec⌋)N ≺' Q
  proof(induct n arbitrary: xvec yvec M N P Q R)
    case(0 xvec yvec M N P Q R)
    have Eq: (⌊ν*xvec⌋)M ≺' P = (⌊ν*yvec⌋)N ≺' (Q ∥ R) by fact
    from ‹0 = length xvec› have xvec = [] by auto
    moreover with Eq have yvec = []
      by(cases yvec) auto
    ultimately show ?case using Eq
      by(simp add: boundOutput.inject)
  next
```

case(*Suc n xvec yvec M N P Q R*)
from ‹*Suc n = length xvec*›
obtain *x xvec′* **where** *xvec = x#xvec′* **and** *length xvec′ = n*
by(*cases xvec*) *auto*
from ‹(|*ν***xvec*|)*M* ≺′ *P* = (|*ν***yvec*|)*N* ≺′ (*Q* ∥ *R*)› ‹*xvec = x # xvec′*›
obtain *y yvec′* **where** (|*ν**(*x*#*xvec′*)|)*M* ≺′ *P* = (|*ν**(*y*#*yvec′*)|)*N* ≺′ (*Q* ∥ *R*)
and *yvec = y#yvec′*
by(*cases yvec*) *auto*
then have *EQ*: (|*νx*|)((|*ν***xvec′*|)*M* ≺′ *P*) = (|*νy*|)((|*ν***yvec′*|)*N* ≺′ (*Q* ∥ *R*))
by *simp*
from ‹*xvec* ♯* *R*› ‹*yvec* ♯* *R*› ‹*xvec = x#xvec′*› ‹*yvec = y#yvec′*›
have *x* ♯ *R* **and** *xvec′* ♯* *R* **and** *y* ♯ *R* **and** *yvec′* ♯* *R* **by** *auto*
have *IH*: ⋀*xvec yvec M N P Q R*. ⟦(|*ν***xvec*|)*M* ≺′ (*P*::('*a*, '*b*, '*c*) *psi*) = (|*ν***yvec*|)*N* ≺′ (*Q* ∥ *R*); *xvec* ♯* *R*; *yvec* ♯* *R*; *n = length xvec*⟧ ⟹ ∃ *T*. *P* = *T* ∥ *R* ∧ (|*ν***xvec*|)*M* ≺′ *T* = (|*ν***yvec*|)*N* ≺′ *Q*
by *fact*
show *?case*
proof(*cases x = y*)
assume *x = y*
with *EQ* **have** (|*ν***xvec′*|)*M* ≺′ *P* = (|*ν***yvec′*|)*N* ≺′ (*Q* ∥ *R*)
by(*simp add*: *boundOutput.inject alpha*)
with ‹*xvec′* ♯* *R*› ‹*yvec′* ♯* *R*› ‹*length xvec′ = n*›
obtain *T* **where** *P* = *T* ∥ *R* **and** (|*ν***xvec′*|)*M* ≺′ *T* = (|*ν***yvec′*|)*N* ≺′ *Q*
by(*auto dest*: *IH*)
with ‹*xvec*=*x*#*xvec′*› ‹*yvec*=*y*#*yvec′*› ‹*x*=*y*› **show** *?case*
by(*force simp add*: *boundOutput.inject alpha*)
next
assume *x* ≠ *y*
with *EQ* ‹*x* ♯ *R*› ‹*y* ♯ *R*›
have (|*ν***xvec′*|)*M* ≺′ *P* = (|*ν**([(*x*, *y*)] ∙ *yvec′*)|)([(*x*, *y*)] ∙ *N*) ≺′ (([(*x*, *y*)] ∙ *Q*) ∥ *R*)
and *xFreshQR*: *x* ♯ (|*ν***yvec′*|)*N* ≺′ (*Q* ∥ *R*)
by(*simp add*: *boundOutput.inject alpha eqvts*)+
moreover from ‹*yvec′* ♯* *R*› **have** ([(*x*, *y*)] ∙ *yvec′*) ♯* ([(*x*, *y*)] ∙ *R*)
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹*x* ♯ *R*› ‹*y* ♯ *R*› **have** ([(*x*, *y*)] ∙ *yvec′*) ♯* *R* **by** *simp*
moreover note ‹*xvec′* ♯* *R*› ‹*length xvec′ = n*›
ultimately obtain *T* **where** *P* = *T* ∥ *R* **and** *A*: (|*ν***xvec′*|)*M* ≺′ *T* = (|*ν**([(*x*, *y*)] ∙ *yvec′*)|)([(*x*, *y*)] ∙ *N*) ≺′ ([(*x*, *y*)] ∙ *Q*)
by(*auto dest*: *IH*)

from *A* **have** (|*νx*|)((|*ν***xvec′*|)*M* ≺′ *T*) = (|*νx*|)((|*ν**([(*x*, *y*)] ∙ *yvec′*)|)([(*x*, *y*)] ∙ *N*) ≺′ ([(*x*, *y*)] ∙ *Q*))
by(*simp add*: *boundOutput.inject alpha*)
moreover from *xFreshQR* **have** *x* ♯ (|*ν***yvec′*|)*N* ≺′ *Q*
by(*force simp add*: *boundOutputFresh*)
ultimately show *?thesis* **using** ‹*P* = *T* ∥ *R*› ‹*xvec*=*x*#*xvec′*› ‹*yvec*=*y*#*yvec′*› *xFreshQR*
by(*force simp add*: *alphaBoundOutput name-swap eqvts*)

```
    qed
  qed
  ultimately show ?thesis
    by blast
qed

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 (|ν*xvec|)M ≺' P = (|ν*yvec|)N ≺' (Q ∥ R)
  and   xvec ♯* yvec

obtains T p where set p ⊆ set xvec × set yvec and P = T ∥ (p ∙ R) and
(|ν*xvec|)M ≺' T = (|ν*yvec|)N ≺' Q
proof −
  assume ⋀p T. [[set p ⊆ set xvec × set yvec; P = T ∥ (p ∙ R); (|ν*xvec|)M ≺' T
= (|ν*yvec|)N ≺' Q]] ⟹ thesis
  moreover obtain n where n = length xvec by auto
   with assms have ∃p T. set p ⊆ set xvec × set yvec ∧ P = T ∥ (p ∙ R) ∧
(|ν*xvec|)M ≺' T = (|ν*yvec|)N ≺' Q
  proof(induct n arbitrary: xvec yvec M N P Q R)
    case(0 xvec yvec M N P Q R)
    have Eq: (|ν*xvec|)M ≺' P = (|ν*yvec|)N ≺' (Q ∥ R) by fact
    from ‹0 = length xvec› have xvec = [] by auto
    moreover with Eq have yvec = []
      by(cases yvec) auto
    ultimately show ?case using Eq
      by(simp add: boundOutput.inject)
  next
    case(Suc n xvec yvec M N P Q R)
    from ‹Suc n = length xvec›
    obtain x xvec' where xvec = x#xvec' and length xvec' = n
      by(cases xvec) auto
    from ‹(|ν*xvec|)M ≺' P = (|ν*yvec|)N ≺' (Q ∥ R)› ‹xvec = x # xvec'›
    obtain y yvec' where (|ν*(x#xvec')|)M ≺' P = (|ν*(y#yvec')|)N ≺' (Q ∥ R)
      and yvec = y#yvec'
      by(cases yvec) auto
    then have Eq: (|νx|)((|ν*xvec'|)M ≺' P) = (|νy|)((|ν*yvec'|)N ≺' (Q ∥ R))
      by simp
     from ‹xvec = x#xvec'› ‹yvec=y#yvec'› ‹xvec ♯* yvec› have x ≠ y and x ♯
yvec' and y ♯ xvec' and xvec' ♯* yvec'
      by auto
    from Eq ‹x ≠ y› have Eq': (|ν*xvec'|)M ≺' P = [(x, y)] ∙ (|ν*yvec'|)N ≺' (Q ∥
```

R)
and *xFreshQR*: $x \sharp (\!|\nu * yvec'|\!) N \prec' (Q \parallel R)$
by(*simp add*: *boundOutput.inject alpha*)+
have *IH*: $\bigwedge xvec\ yvec\ M\ N\ P\ Q\ R.\ [\![(\!|\nu * xvec|\!) M \prec' (P::('a, 'b, 'c)\ psi) = (\!|\nu * yvec|\!) N \prec' (Q \parallel R);\ xvec \sharp* yvec;\ n = length\ xvec]\!] \Longrightarrow \exists p\ T.\ set\ p \subseteq set\ xvec \times set\ yvec \wedge P = T \parallel (p \cdot R) \wedge (\!|\nu * xvec|\!) M \prec' T = (\!|\nu * yvec|\!) N \prec' Q$
by *fact*
show *?case*
proof(*cases* $x \sharp (\!|\nu * xvec'|\!) M \prec' P$)
assume $x \sharp (\!|\nu * xvec'|\!) M \prec' P$
with *Eq* **have** *yFreshQR*: $y \sharp (\!|\nu * yvec'|\!) N \prec' (Q \parallel R)$
by(*rule boundOutputEqFresh*)
with *Eq'* *xFreshQR* **have** $(\!|\nu * xvec'|\!) M \prec' P = (\!|\nu * yvec'|\!) N \prec' (Q \parallel R)$
by *simp*
with ‹$xvec' \sharp* yvec'$› ‹$length\ xvec' = n$›
obtain $p\ T$ **where** *S*: $set\ p \subseteq set\ xvec' \times set\ yvec'$ **and** $P = T \parallel (p \cdot R)$ **and** *A*: $(\!|\nu * xvec'|\!) M \prec' T = (\!|\nu * yvec'|\!) N \prec' Q$
by(*auto dest*: *IH*)
from *yFreshQR xFreshQR* **have** *yFreshQ*: $y \sharp (\!|\nu * yvec'|\!) N \prec' Q$ **and** *xFreshQ*: $x \sharp (\!|\nu * yvec'|\!) N \prec' Q$
by(*force simp add*: *BOresChainSupp fresh-def boundOutput.supp psi.supp*)+
then have $(\!|\nu x|\!)((\!|\nu * yvec'|\!) N \prec' Q) = (\!|\nu y|\!)((\!|\nu * yvec'|\!) N \prec' Q)$ **by** (*subst alphaBoundOutput*) *simp*+
with *A* **have** $(\!|\nu x|\!)((\!|\nu * xvec'|\!) M \prec' T) = (\!|\nu y|\!)((\!|\nu * yvec'|\!) N \prec' Q)$ **by** *simp*
with ‹$xvec = x \# xvec'$› ‹$yvec = y \# yvec'$› *S* ‹$P = T \parallel (p \cdot R)$› **show** *?case*
by *auto*
next
assume $\neg(x \sharp (\!|\nu * xvec'|\!) M \prec' P)$
then have $x \in supp((\!|\nu * xvec'|\!) M \prec' P)$ **by**(*simp add*: *fresh-def*)
with *Eq* **have** $y \in supp((\!|\nu * yvec'|\!) N \prec' (Q \parallel R))$
by(*rule boundOutputEqSupp*)
then have $y \sharp yvec'$ **by**(*simp add*: *BOresChainSupp fresh-def*)
with *Eq'* ‹$x \sharp yvec'$› **have** $(\!|\nu * xvec'|\!) M \prec' P = (\!|\nu * yvec'|\!)([(x, y)] \cdot N) \prec' (([(x, y)] \cdot Q) \parallel ([(x, y)] \cdot R))$
by(*simp add*: *eqvts*)
moreover note ‹$xvec' \sharp* yvec'$› ‹$length\ xvec' = n$›
ultimately obtain $p\ T$ **where** *S*: $set\ p \subseteq set\ xvec' \times set\ yvec'$ **and** $P = T \parallel (p \cdot [(x, y)] \cdot R)$ **and** *A*: $(\!|\nu * xvec'|\!) M \prec' T = (\!|\nu * yvec'|\!)([(x, y)] \cdot N) \prec' ([(x, y)] \cdot Q)$
by(*auto dest*: *IH*)

from *S* **have** $set(p@[(x, y)]) \subseteq set(x \# xvec') \times set(y \# yvec')$ **by** *auto*
moreover from ‹$P = T \parallel (p \cdot [(x, y)] \cdot R)$› **have** $P = T \parallel ((p\ @\ [(x, y)]) \cdot R)$
by(*simp add*: *pt2*[*OF pt-name-inst*])
moreover from *xFreshQR* **have** *xFreshQ*: $x \sharp (\!|\nu * yvec'|\!) N \prec' Q$
by(*force simp add*: *BOresChainSupp fresh-def boundOutput.supp psi.supp*)+
with ‹$x \sharp yvec'$› ‹$y \sharp yvec'$› ‹$x \neq y$› **have** $y \sharp (\!|\nu * yvec'|\!)([(x, y)] \cdot N) \prec' ([(x, y)] \cdot Q)$

```
      by(simp add: fresh-left calc-atm)
    with ‹x ♯ yvec'› ‹y ♯ yvec'› have (|νx|)((|ν*yvec'|)([(x, y)] • N) ≺' ([(x, y)] •
Q)) = (|νy|)((|ν*yvec'|)N ≺' Q)
      by(subst alphaBoundOutput) (assumption | simp add: eqvts)+
    with A have (|νx|)((|ν*xvec'|)M ≺' T) = (|νy|)((|ν*yvec'|)N ≺' Q) by simp
    ultimately show ?thesis using ‹xvec=x#xvec'› ‹yvec=y#yvec'›
      by − (rule exI[where x=p@[(x, y)]], force)
  qed
 qed
 ultimately show ?thesis
  by blast
qed

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 (|ν*xvec|)M ≺' P = (|ν*yvec|)N ≺' (Q ∥ R)
 and    xvec ♯* Q
 and    yvec ♯* Q

obtains T where P = Q ∥ T and (|ν*xvec|)M ≺' T = (|ν*yvec|)N ≺' R
proof −
 assume ⋀T. [[P = Q ∥ T; (|ν*xvec|)M ≺' T = (|ν*yvec|)N ≺' R]] ⟹ thesis
 moreover obtain n where n = length xvec by auto
 with assms have ∃ T. P = Q ∥ T ∧ (|ν*xvec|)M ≺' T = (|ν*yvec|)N ≺' R
 proof(induct n arbitrary: xvec yvec M N P Q R)
  case(0 xvec yvec M N P Q R)
  have Eq: (|ν*xvec|)M ≺' P = (|ν*yvec|)N ≺' (Q ∥ R) by fact
  from ‹0 = length xvec› have xvec = [] by auto
  moreover with Eq have yvec = []
   by(cases yvec) auto
  ultimately show ?case using Eq
   by(simp add: boundOutput.inject)
 next
  case(Suc n xvec yvec M N P Q R)
  from ‹Suc n = length xvec›
  obtain x xvec' where xvec = x#xvec' and length xvec' = n
   by(cases xvec) auto
  from ‹(|ν*xvec|)M ≺' P = (|ν*yvec|)N ≺' (Q ∥ R)› ‹xvec = x # xvec'›
  obtain y yvec' where (|ν*(x#xvec')|)M ≺' P = (|ν*(y#yvec')|)N ≺' (Q ∥ R)
   and yvec = y#yvec'
   by(cases yvec) auto
  then have EQ: (|νx|)((|ν*xvec'|)M ≺' P) = (|νy|)((|ν*yvec'|)N ≺' (Q ∥ R))
```

by *simp*
from ‹*xvec* ♯* *Q*› ‹*yvec* ♯* *Q*› ‹*xvec* = *x*#*xvec'*› ‹*yvec* = *y*#*yvec'*›
have *x* ♯ *Q* **and** *xvec'* ♯* *Q* **and** *y* ♯ *Q* **and** *yvec'* ♯* *Q* **by** *auto*
have *IH*: ⋀*xvec yvec M N P Q R*. ⟦(|*ν**xvec*|)*M* ≺' (*P*::('*a*, '*b*, '*c*) *psi*) = (|*ν**yvec*|)*N* ≺' (*Q* ∥ *R*); *xvec* ♯* *Q*; *yvec* ♯* *Q*; *n* = *length xvec*⟧ ⟹ ∃ *T*. *P* = *Q* ∥ *T* ∧ (|*ν**xvec*|)*M* ≺' *T* = (|*ν**yvec*|)*N* ≺' *R*
by *fact*
show *?case*
proof(*cases x* = *y*)
assume *x* = *y*
with *EQ* **have** (|*ν**xvec'*|)*M* ≺' *P* = (|*ν**yvec'*|)*N* ≺' (*Q* ∥ *R*)
by(*simp add*: *boundOutput.inject alpha*)
with ‹*xvec'* ♯* *Q*› ‹*yvec'* ♯* *Q*› ‹*length xvec'* = *n*›
obtain *T* **where** *P* = *Q* ∥ *T* **and** (|*ν**xvec'*|)*M* ≺' *T* = (|*ν**yvec'*|)*N* ≺' *R*
by(*auto dest*: *IH*)
with ‹*xvec*=*x*#*xvec'*› ‹*yvec*=*y*#*yvec'*› ‹*x*=*y*› **show** *?case*
by(*force simp add*: *boundOutput.inject alpha*)
next
assume *x* ≠ *y*
with *EQ* ‹*x* ♯ *Q*› ‹*y* ♯ *Q*›
have (|*ν**xvec'*|)*M* ≺' *P* = (|*ν**([(*x*, *y*)] ∙ *yvec'*)|)([(*x*, *y*)] ∙ *N*) ≺' (*Q* ∥ ([(*x*, *y*)] ∙ *R*))
and *xFreshQR*: *x* ♯ (|*ν**yvec'*|)*N* ≺' (*Q* ∥ *R*)
by(*simp add*: *boundOutput.inject alpha eqvts*)+
moreover from ‹*yvec'* ♯* *Q*› **have** ([(*x*, *y*)] ∙ *yvec'*) ♯* ([(*x*, *y*)] ∙ *Q*)
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹*x* ♯ *Q*› ‹*y* ♯ *Q*› **have** ([(*x*, *y*)] ∙ *yvec'*) ♯* *Q* **by** *simp*
moreover note ‹*xvec'* ♯* *Q*› ‹*length xvec'* = *n*›
ultimately obtain *T* **where** *P* = *Q* ∥ *T* **and** *A*: (|*ν**xvec'*|)*M* ≺' *T* = (|*ν**([(*x*, *y*)] ∙ *yvec'*)|)([(*x*, *y*)] ∙ *N*) ≺' ([(*x*, *y*)] ∙ *R*)
by(*auto dest*: *IH*)

from *A* **have** (|*ν**x*|)((|*ν**xvec'*|)*M* ≺' *T*) = (|*ν**x*|)((|*ν**([(*x*, *y*)] ∙ *yvec'*)|)([(*x*, *y*)] ∙ *N*) ≺' ([(*x*, *y*)] ∙ *R*))
by(*simp add*: *boundOutput.inject alpha*)
moreover from *xFreshQR* **have** *x* ♯ (|*ν**yvec'*|)*N* ≺' *R*
by(*force simp add*: *boundOutputFresh*)
ultimately show *?thesis* **using** ‹*P* = *Q* ∥ *T*› ‹*xvec*=*x*#*xvec'*› ‹*yvec*=*y*#*yvec'*› *xFreshQR*
by(*force simp add*: *alphaBoundOutput name-swap eqvts*)
qed
qed
ultimately show *?thesis*
by *blast*
qed

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 −
assume $\bigwedge p\ T.\ [\![$*set* $p \subseteq$ *set xvec* $\times$ *set yvec*; $P = (p \cdot Q) \parallel T$; $(\!|\nu{*}xvec|\!)M \prec' T = (\!|\nu{*}yvec|\!)N \prec' R]\!] \Longrightarrow$ *thesis*
moreover obtain n **where** $n =$ *length xvec* **by** *auto*
with *assms* **have** $\exists p\ T.$ *set* $p \subseteq$ *set xvec* $\times$ *set yvec* $\wedge$ $P = (p \cdot Q) \parallel T \wedge (\!|\nu{*}xvec|\!)M \prec' T = (\!|\nu{*}yvec|\!)N \prec' R$
proof(*induct n arbitrary*: *xvec yvec M N P Q R*)
case(*0 xvec yvec M N P Q R*)
have *Eq*: $(\!|\nu{*}xvec|\!)M \prec' P = (\!|\nu{*}yvec|\!)N \prec' (Q \parallel R)$ **by** *fact*
from ‹*0* = *length xvec*› **have** *xvec* = [] **by** *auto*
moreover with *Eq* **have** *yvec* = []
by(*cases yvec*) *auto*
ultimately show *?case* **using** *Eq*
by(*simp add*: *boundOutput.inject*)
next
case(*Suc n xvec yvec M N P Q R*)
from ‹*Suc n* = *length xvec*›
obtain *x xvec'* **where** *xvec* = *x*#*xvec'* **and** *length xvec'* = *n*
by(*cases xvec*) *auto*
from ‹$(\!|\nu{*}xvec|\!)M \prec' P = (\!|\nu{*}yvec|\!)N \prec' (Q \parallel R)$› ‹*xvec* = *x* # *xvec'*›
obtain *y yvec'* **where** $(\!|\nu{*}(x\#xvec')|\!)M \prec' P = (\!|\nu{*}(y\#yvec')|\!)N \prec' (Q \parallel R)$
and *yvec* = *y*#*yvec'*
by(*cases yvec*) *auto*
then have *Eq*: $(\!|\nu x|\!)((\!|\nu{*}xvec'|\!)M \prec' P) = (\!|\nu y|\!)((\!|\nu{*}yvec'|\!)N \prec' (Q \parallel R))$
by *simp*
from ‹*xvec* = *x*#*xvec'*› ‹*yvec*=*y*#*yvec'*› ‹*xvec* $\sharp*$ *yvec*› **have** $x \neq y$ **and** $x \sharp yvec'$ **and** $y \sharp xvec'$ **and** *xvec'* $\sharp*$ *yvec'*
by *auto*
from *Eq* ‹$x \neq y$› **have** *Eq'*: $(\!|\nu{*}xvec'|\!)M \prec' P = [(x, y)] \cdot (\!|\nu{*}yvec'|\!)N \prec' (Q \parallel R)$
and *xFreshQR*: $x \sharp (\!|\nu{*}yvec'|\!)N \prec' (Q \parallel R)$
by(*simp add*: *boundOutput.inject alpha*)+
have *IH*: $\bigwedge$*xvec yvec M N P Q R.* $[\![(\!|\nu{*}xvec|\!)M \prec' (P::('a, 'b, 'c)\ psi) = (\!|\nu{*}yvec|\!)N \prec' (Q \parallel R);$ *xvec* $\sharp*$ *yvec*; $n =$ *length xvec*$]\!] \Longrightarrow \exists p\ T.$ *set* $p \subseteq$ *set xvec* $\times$ *set yvec* $\wedge$ $P = (p \cdot Q) \parallel T \wedge (\!|\nu{*}xvec|\!)M \prec' T = (\!|\nu{*}yvec|\!)N \prec' R$
by *fact*
show *?case*
proof(*cases* $x \sharp (\!|\nu{*}xvec'|\!)M \prec' P$)

assume $x \sharp (\!|\nu{*}xvec'|\!)M \prec' P$
with *Eq* **have** *yFreshQR*: $y \sharp (\!|\nu{*}yvec'|\!)N \prec' (Q \parallel R)$
by(*rule boundOutputEqFresh*)
with *Eq'* *xFreshQR* **have** $(\!|\nu{*}xvec'|\!)M \prec' P = (\!|\nu{*}yvec'|\!)N \prec' (Q \parallel R)$
by *simp*
with ‹*xvec'* $\sharp*$ *yvec'*› ‹*length xvec'* = *n*›
obtain *p T* **where** *S*: *set p* ⊆ *set xvec'* × *set yvec'* **and** $P = (p \cdot Q) \parallel T$ **and** *A*: $(\!|\nu{*}xvec'|\!)M \prec' T = (\!|\nu{*}yvec'|\!)N \prec' R$
by(*auto dest*: *IH*)
from *yFreshQR xFreshQR* **have** *yFreshR*: $y \sharp (\!|\nu{*}yvec'|\!)N \prec' R$ **and** *xFreshQ*: $x \sharp (\!|\nu{*}yvec'|\!)N \prec' R$
by(*force simp add*: *BOresChainSupp fresh-def boundOutput.supp psi.supp*)+
then have $(\!|\nu x|\!)((\!|\nu{*}yvec'|\!)N \prec' R) = (\!|\nu y|\!)((\!|\nu{*}yvec'|\!)N \prec' R)$ **by** (*subst alphaBoundOutput*) *simp*+
with *A* **have** $(\!|\nu x|\!)((\!|\nu{*}xvec'|\!)M \prec' T) = (\!|\nu y|\!)((\!|\nu{*}yvec'|\!)N \prec' R)$ **by** *simp*
with ‹*xvec*=*x*#*xvec'*› ‹*yvec*=*y*#*yvec'*› *S* ‹*P* = (*p* · *Q*) ∥ *T*› **show** *?case*
by *auto*
next
assume $\neg(x \sharp (\!|\nu{*}xvec'|\!)M \prec' P)$
then have $x \in supp((\!|\nu{*}xvec'|\!)M \prec' P)$ **by**(*simp add*: *fresh-def*)
with *Eq* **have** $y \in supp((\!|\nu{*}yvec'|\!)N \prec' (Q \parallel R))$
by(*rule boundOutputEqSupp*)
then have $y \sharp yvec'$ **by**(*simp add*: *BOresChainSupp fresh-def*)
with *Eq'* ‹*x* $\sharp$ *yvec'*› **have** $(\!|\nu{*}xvec'|\!)M \prec' P = (\!|\nu{*}yvec'|\!)([(x, y)] \cdot N) \prec' (([(x, y)] \cdot Q) \parallel ([(x, y)] \cdot R))$
by(*simp add*: *eqvts*)
moreover note ‹*xvec'* $\sharp*$ *yvec'*› ‹*length xvec'* = *n*›
ultimately obtain *p T* **where** *S*: *set p* ⊆ *set xvec'* × *set yvec'* **and** $P = (p \cdot [(x, y)] \cdot Q) \parallel T$ **and** *A*: $(\!|\nu{*}xvec'|\!)M \prec' T = (\!|\nu{*}yvec'|\!)([(x, y)] \cdot N) \prec' ([(x, y)] \cdot R)$
by(*auto dest*: *IH*)

from *S* **have** *set*(*p*@[(*x*, *y*)]) ⊆ *set*(*x*#*xvec'*) × *set*(*y*#*yvec'*) **by** *auto*
moreover from ‹*P* = (*p* · [(*x*, *y*)] · *Q*) ∥ *T*› **have** $P = ((p \mathbin{@} [(x, y)]) \cdot Q) \parallel T$
by(*simp add*: *pt2*[*OF pt-name-inst*])
moreover from *xFreshQR* **have** *xFreshR*: $x \sharp (\!|\nu{*}yvec'|\!)N \prec' R$
by(*force simp add*: *BOresChainSupp fresh-def boundOutput.supp psi.supp*)+
with ‹*x* $\sharp$ *yvec'*› ‹*y* $\sharp$ *yvec'*› ‹*x* ≠ *y*› **have** $y \sharp (\!|\nu{*}yvec'|\!)([(x, y)] \cdot N) \prec' ([(x, y)] \cdot R)$
by(*simp add*: *fresh-left calc-atm*)
with ‹*x* $\sharp$ *yvec'*› ‹*y* $\sharp$ *yvec'*› **have** $(\!|\nu x|\!)((\!|\nu{*}yvec'|\!)([(x, y)] \cdot N) \prec' ([(x, y)] \cdot R)) = (\!|\nu y|\!)((\!|\nu{*}yvec'|\!)N \prec' R)$
by(*subst alphaBoundOutput*) (*assumption* | *simp add*: *eqvts*)+
with *A* **have** $(\!|\nu x|\!)((\!|\nu{*}xvec'|\!)M \prec' T) = (\!|\nu y|\!)((\!|\nu{*}yvec'|\!)N \prec' R)$ **by** *simp*
ultimately show *?thesis* **using** ‹*xvec*=*x*#*xvec'*› ‹*yvec*=*y*#*yvec'*›
by(*force intro*!: *exI*[**where** *x*=*p*@[(*x*, *y*)]])
qed
qed

ultimately show *?thesis*
by *blast*
qed

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

shows $(\!|\nu*(xvec@yvec)|\!)B = (\!|\nu*xvec|\!)((\!|\nu*yvec|\!)B)$
by(*induct xvec*) *auto*

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 ∃ *yvec1 y yvec2*. *yvec* = *yvec1*@*y*#*yvec2* ∧ *length xvec1* = *length yvec1* ∧ *length xvec2* = *length yvec2*
apply(*rule exI*[**where** *x*=*take* (*length xvec1*) *yvec*])
apply(*rule exI*[**where** *x*=*yvec* ! *length xvec1*])
apply(*rule exI*[**where** *x*=*drop* (*length xvec1*+*1*) *yvec*])
using *assms* **by**(*auto simp add*: *id-take-nth-drop*)

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 −
assume *Ass*: ⋀*yvec1 y yvec2*.
⟦*yvec* = *yvec1* @ *y* # *yvec2*; *length xvec1* = *length yvec1*; *length xvec2* = *length yvec2*;
⦇*ν*∗(*xvec1* @ *xvec2*)⦈*M* ≺′ *P* = ⦇*ν*∗(*yvec1* @ *yvec2*)⦈([(*x*, *y*)] ∙ *N*) ≺′ ([(*x*, *y*)] ∙ *Q*)⟧
⟹ *thesis*
from *Eq* **have** *length*(*xvec1*@*x*#*xvec2*) = *length yvec* **by**(*rule boundOutputChainEqLength*)
then obtain *yvec1 y yvec2* **where** *A*: *yvec* = *yvec1*@*y*#*yvec2* **and** *length xvec1* = *length yvec1*
and *length xvec2* = *length yvec2*
by(*metis openInjectAux sym*)

from ‹*distinct yvec*› *A* **have** *y* ♯ *yvec2* **by** *simp*
from *A* ‹*x* ♯ *yvec*› **have** *x* ♯ *yvec2* **and** *x* ♯ *yvec1* **by** *simp*+
with *Eq* ‹*length xvec1* = *length yvec1*› ‹*x* ♯ *N*› ‹*x* ♯ *Q*› ‹*y* ♯ *yvec2*› ‹*x* ♯ *xvec1*› *A*
have ⦇*ν*∗(*xvec1*@*xvec2*)⦈*M* ≺′ *P* = ⦇*ν*∗(*yvec1*@*yvec2*)⦈([(*x*, *y*)] ∙ *N*) ≺′ ([(*x*, *y*)] ∙ *Q*)
by(*force dest*: *boundOutputChainOpenIH simp add*: *boundOutputApp BOresChainSupp fresh-def boundOutput.supp eqvts*)
with ‹*length xvec1* = *length yvec1*› ‹*length xvec2* = *length yvec2*› *A Ass* **show** *?thesis*
by *blast*
qed

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*: ⦇*ν*∗(*xvec1*@*x*#*xvec2*)⦈*M* ≺′ *P* = ⦇*ν*∗*yvec*⦈*N* ≺′ *Q*
and *x* ♯ *xvec1*
and *x* ♯ *yvec*
and *x* ♯ *N*
and *x* ♯ *Q*

obtains *yvec1 y yvec2* **where** *yvec*=*yvec1*@*y*#*yvec2* **and** *length xvec1* = *length yvec1* **and** *length xvec2* = *length yvec2*
and ⦇*ν*∗(*xvec1*@*xvec2*)⦈*M* ≺′ *P* = ⦇*ν*∗(*yvec1*@[(*x*, *y*)] ∙ *yvec2*)⦈([(*x*, *y*)] ∙ *N*) ≺′ ([(*x*, *y*)] ∙ *Q*)
proof −
assume *Ass*: ⋀*yvec1 y yvec2*.
⟦*yvec* = *yvec1* @ *y* # *yvec2*; *length xvec1* = *length yvec1*; *length xvec2* =

length yvec2;
(⦇*ν**(*xvec1* @ *xvec2*)⦈*M* ≺′ *P* = ⦇*ν**(*yvec1* @ ([(*x*, *y*)] ∙ *yvec2*))⦈([(*x*, *y*)] ∙ *N*) ≺′ ([(*x*, *y*)] ∙ *Q*)⟧
⟹ *thesis*
from *Eq* **have** *length*(*xvec1*@*x*#*xvec2*) = *length yvec* **by**(*rule boundOutputChainEqLength*)
then obtain *yvec1 y yvec2* **where** *A*: *yvec* = *yvec1*@*y*#*yvec2* **and** *length xvec1* = *length yvec1*
and *length xvec2* = *length yvec2*
by(*metis openInjectAux sym*)

from *A* ‹*x* ♯ *yvec*› **have** *x* ♯ *yvec2* **and** *x* ♯ *yvec1* **by** *simp+*
with *Eq* ‹*length xvec1* = *length yvec1*› ‹*x* ♯ *N*› ‹*x* ♯ *Q*› ‹*x* ♯ *xvec1*› *A*
have ⦇*ν**(*xvec1*@*xvec2*)⦈*M* ≺′ *P* = ⦇*ν**(*yvec1*@([(*x*, *y*)] ∙ *yvec2*))⦈([(*x*, *y*)] ∙ *N*) ≺′ ([(*x*, *y*)] ∙ *Q*)
by(*force dest*: *boundOutputChainOpenIH simp add*: *boundOutputApp BOresChainSupp fresh-def boundOutput.supp eqvts*)
with ‹*length xvec1* = *length yvec1*› ‹*length xvec2* = *length yvec2*› *A Ass* **show** *?thesis*
by *blast*
qed

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 ⦇*ν***xvec*⦈*M* ≺′ *P* = ⦇*ν***yvec*⦈*N* ≺′ ⦇*νz*⦈*Q*
and *z* ♯ *xvec*
and *z* ♯ *yvec*

obtains *R* **where** *P* = ⦇*νz*⦈*R* **and** ⦇*ν***xvec*⦈*M* ≺′ *R* = ⦇*ν***yvec*⦈*N* ≺′ *Q*
proof −
assume ⋀*R*. ⟦*P* = ⦇*νz*⦈*R*; ⦇*ν***xvec*⦈*M* ≺′ *R* = ⦇*ν***yvec*⦈*N* ≺′ *Q*⟧ ⟹ *thesis*
moreover obtain *n* **where** *n* = *length xvec* **by** *auto*
with *assms* **have** ∃ *R*. *P* = ⦇*νz*⦈*R* ∧ ⦇*ν***xvec*⦈*M* ≺′ *R* = ⦇*ν***yvec*⦈*N* ≺′ *Q*
proof(*induct n arbitrary*: *xvec yvec M N P Q z*)
case(*0 xvec yvec M N P Q z*)
have *Eq*: ⦇*ν***xvec*⦈*M* ≺′ *P* = ⦇*ν***yvec*⦈*N* ≺′ ⦇*νz*⦈*Q* **by** *fact*
from ‹*0* = *length xvec*› **have** *xvec* = [] **by** *auto*
moreover with *Eq* **have** *yvec* = []
by(*cases yvec*) *auto*
ultimately show *?case* **using** *Eq*
by(*simp add*: *boundOutput.inject*)
next
case(*Suc n xvec yvec M N P Q z*)

from ‹*Suc n = length xvec*›
obtain *x xvec'* **where** *xvec = x#xvec'* **and** *length xvec' = n*
by(*cases xvec*) *auto*
from ‹$(\!|\nu{*}xvec|\!)M \prec' P = (\!|\nu{*}yvec|\!)N \prec' ((\!|\nu z|\!)Q)$› ‹*xvec = x # xvec'*›
obtain *y yvec'* **where** $(\!|\nu{*}(x\#xvec')|\!)M \prec' P = (\!|\nu{*}(y\#yvec')|\!)N \prec' (\!|\nu z|\!)Q$
and *yvec = y#yvec'*
by(*cases yvec*) *auto*
then have *EQ*: $(\!|\nu x|\!)((\!|\nu{*}xvec'|\!)M \prec' P) = (\!|\nu y|\!)((\!|\nu{*}yvec'|\!)N \prec' (\!|\nu z|\!)Q)$
by *simp*
from ‹*z ♯ xvec*› ‹*z ♯ yvec*› ‹*xvec = x#xvec'*› ‹*yvec = y#yvec'*›
have *z ≠ x* **and** *z ≠ y* **and** *z ♯ xvec'* **and** *z ♯ yvec'*
by *simp+*
have *IH*: $\bigwedge xvec\ yvec\ M\ N\ P\ Q\ z.\ [\![(\!|\nu{*}xvec|\!)M \prec' (P::('a, 'b, 'c)\ psi) = (\!|\nu{*}yvec|\!)N \prec' (\!|\nu z|\!)Q;\ z \sharp xvec;\ z \sharp yvec;\ n = length\ xvec]\!] \Longrightarrow \exists R.\ P = (\!|\nu z|\!)R \wedge (\!|\nu{*}xvec|\!)M \prec' R = (\!|\nu{*}yvec|\!)N \prec' Q$
by *fact*
show *?case*
proof(*cases x = y*)
assume *x = y*
with *EQ* **have** $(\!|\nu{*}xvec'|\!)M \prec' P = (\!|\nu{*}yvec'|\!)N \prec' (\!|\nu z|\!)Q$
by(*simp add*: *boundOutput.inject alpha*)
with ‹*z ♯ xvec'*› ‹*z ♯ yvec'*› ‹*length xvec' = n*›
obtain *R* **where** *P = (|νz|)R* **and** $(\!|\nu{*}xvec'|\!)M \prec' R = (\!|\nu{*}yvec'|\!)N \prec' Q$
by(*auto dest*: *IH*)
with ‹*xvec=x#xvec'*› ‹*yvec=y#yvec'*› ‹*x=y*› **show** *?case*
by(*force simp add*: *boundOutput.inject alpha*)
next
assume *x ≠ y*
with *EQ* ‹*z ≠ x*› ‹*z ≠ y*›
have $(\!|\nu{*}xvec'|\!)M \prec' P = (\!|\nu{*}([(x,\ y)] \cdot yvec')|\!)([(x,\ y)] \cdot N) \prec' (\!|\nu z|\!)([(x,\ y)] \cdot Q)$
and *xFreshzQ*: *x ♯ (|νz|)Q*
by(*simp add*: *boundOutput.inject alpha eqvts*)+
moreover from ‹*z ≠ x*› ‹*z ≠ y*› ‹*z ♯ yvec'*› ‹*x ≠ y*› **have** $z \sharp ([(x,\ y)] \cdot yvec')$
by(*simp add*: *fresh-left calc-atm*)
moreover note ‹*z ♯ xvec'*› ‹*length xvec' = n*›
ultimately obtain *R* **where** *P = (|νz|)R* **and** *A*: $(\!|\nu{*}xvec'|\!)M \prec' R = (\!|\nu{*}([(x,\ y)] \cdot yvec')|\!)([(x,\ y)] \cdot N) \prec' ([(x,\ y)] \cdot Q)$
by(*auto dest*: *IH*)

from *A* **have** $(\!|\nu x|\!)((\!|\nu{*}xvec'|\!)M \prec' R) = (\!|\nu x|\!)((\!|\nu{*}([(x,\ y)] \cdot yvec')|\!)([(x,\ y)] \cdot N) \prec' ([(x,\ y)] \cdot Q))$
by(*simp add*: *boundOutput.inject alpha*)
moreover from *xFreshzQ* ‹*z ≠ x*› **have** *x ♯ (|νy*yvec'|)N ≺' Q*
by(*simp add*: *boundOutputFresh abs-fresh*)
ultimately show *?thesis* **using** ‹*P = (|νz|)R*› ‹*xvec=x#xvec'*› ‹*yvec=y#yvec'*› *xFreshzQ*
by(*force simp add*: *alphaBoundOutput name-swap eqvts*)
qed

```
  qed
  ultimately show ?thesis
    by blast
qed

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

nominal-datatype 'a action = In 'a::fs-name 'a          (‹-(⦇-⦈)› [90, 90] 90)
  | BrIn 'a::fs-name 'a                                 (‹¿-(⦇-⦈)› [90, 90] 90)
  | Out 'a::fs-name name list 'a                        (‹-(⦇ν*-⦈)⟨-⟩› [90, 90, 90] 90)
  | BrOut '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(⦇N⦈)) = []
  | bn (M(⦇ν*xvec⦈)⟨N⟩) = xvec
  | bn (¡M(⦇ν*xvec⦈)⟨N⟩) = xvec
  | bn (τ) = []
  by(rule TrueI)+

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

shows (p ∙ bn α) = bn(p ∙ α)
  by(nominal-induct α rule: action.strong-induct) auto

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(⦇N⦈)) ≺ P = RBrIn M N P
  | M(⦇ν*xvec⦈)⟨N⟩ ≺ P = ROut M ((⦇ν*xvec⦈)(N ≺' P))
  | (¡M(⦇ν*xvec⦈)⟨N⟩) ≺ P = RBrOut M ((⦇ν*xvec⦈)(N ≺' P))
  | τ ≺ P = (RTau P)
  by(rule TrueI)+

nominal-primrec subject :: ('a::fs-name) action ⇒ 'a option
  where
    subject (M(⦇N⦈)) = Some M
  | subject (¿M(⦇N⦈)) = Some M
  | subject (M(⦇ν*xvec⦈)⟨N⟩) = Some M
```

```
  | subject (¡M(⦇ν∗xvec⦈)⟨N⟩) = Some M
  | subject (τ) = None
  by(rule TrueI)+

nominal-primrec object :: ('a::fs-name) action ⇒ 'a option
  where
    object (M(⦇N⦈)) = Some N
  | object (¿M(⦇N⦈)) = Some N
  | object (M(⦇ν∗xvec⦈)⟨N⟩) = Some N
  | object (¡M(⦇ν∗xvec⦈)⟨N⟩) = Some N
  | object (τ) = None
  by(rule TrueI)+

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
  by(auto simp add: fresh-star-def fresh-some fresh-none)

lemmas [simp] = fresh-some fresh-none

lemma actionFresh[simp]:
  fixes x :: name
    and α :: ('a::fs-name) action

shows (x ♯ α)  = (x ♯ (subject α) ∧ x ♯ (bn α) ∧ x ♯ (object α))
  by(nominal-induct α rule: action.strong-induct) auto

lemma actionFreshChain[simp]:
  fixes X    :: name set
    and α    :: ('a::fs-name) action
    and xvec :: name list

shows (X ♯∗ α) = (X ♯∗ (subject α) ∧ X ♯∗ (bn α) ∧ X ♯∗ (object α))
  and (xvec ♯∗ α) = (xvec ♯∗ (subject α) ∧ xvec ♯∗ (bn α) ∧ xvec ♯∗ (object α))
  by(auto simp add: fresh-star-def)

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

shows (p ∙ subject α) = subject(p ∙ α)
  by(nominal-induct α rule: action.strong-induct) auto

lemma okjectEqvt[eqvt]:
```

fixes p :: *name prm*
and α :: ('*a*::*fs-name*) *action*

shows (p ∙ *object* α) = *object*(p ∙ α)
by(*nominal-induct* α *rule*: *action.strong-induct*) *auto*

lemma *create-residualEqvt*[*eqvt*]:
fixes p :: *name prm*
and α :: ('*a*::*fs-name*) *action*
and P :: ('*a*, '*b*::*fs-name*, '*c*::*fs-name*) *psi*

shows (p ∙ (α ≺ P)) = (p ∙ α) ≺ (p ∙ P)
by(*nominal-induct* α *rule*: *action.strong-induct*)
(*auto simp add*: *eqvts*)

lemma *residualFresh*:
fixes x :: *name*
and α :: '*a*::*fs-name action*
and P :: ('*a*, '*b*::*fs-name*, '*c*::*fs-name*) *psi*

shows (x ♯ (α ≺ P)) = (x ♯ (*subject* α) ∧ (x ∈ (*set*(*bn*(α))) ∨ (x ♯ *object*(α) ∧ x ♯ P)))
by(*nominal-induct* α *rule*: *action.strong-induct*)
(*auto simp add*: *fresh-some fresh-none boundOutputFresh*)

lemma *residualFresh2*[*simp*]:
fixes x :: *name*
and α :: ('*a*::*fs-name*) *action*
and P :: ('*a*, '*b*::*fs-name*, '*c*::*fs-name*) *psi*

assumes x ♯ α
and x ♯ P

shows x ♯ α ≺ P
using *assms*
by(*nominal-induct* α *rule*: *action.strong-induct*) *auto*

lemma *residualFreshChain2*[*simp*]:
fixes *xvec* :: *name list*
and X :: *name set*
and α :: ('*a*::*fs-name*) *action*
and P :: ('*a*, '*b*::*fs-name*, '*c*::*fs-name*) *psi*

shows ⟦*xvec* ♯* α; *xvec* ♯* P⟧ ⟹ *xvec* ♯* (α ≺ P)
and ⟦X ♯* α; X ♯* P⟧ ⟹ X ♯* (α ≺ P)
by(*auto simp add*: *fresh-star-def*)

lemma *residualFreshSimp*[*simp*]:
fixes x :: *name*

and $M :: 'a$::*fs-name*
and $N :: 'a$
and $P :: ('a, 'b$::*fs-name*, $'c$::*fs-name*) *psi*

shows $x \sharp (M(\!|N|\!) \prec P) = (x \sharp M \wedge x \sharp N \wedge x \sharp P)$
and $x \sharp (¿\ M(\!|N|\!) \prec P) = (x \sharp M \wedge x \sharp N \wedge x \sharp P)$
and $x \sharp (M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P) = (x \sharp M \wedge x \sharp ((\!|\nu{*}xvec|\!)(N \prec' P)))$
and $x \sharp (¡M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P) = (x \sharp M \wedge x \sharp ((\!|\nu{*}xvec|\!)(N \prec' P)))$
and $x \sharp (\tau \prec P) = (x \sharp P)$
by(*auto simp add*: *residualFresh*)

lemma *residualInject'*:

shows $(\alpha \prec P = RIn\ M\ N\ Q) = (P = Q \wedge \alpha = M(\!|N|\!))$
and $(\alpha \prec P = RBrIn\ M\ N\ Q) = (P = Q \wedge \alpha = ¿M(\!|N|\!))$
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 $(\alpha \prec P = RBrOut\ M\ B) = (\exists xvec\ N.\ \alpha = ¡M(\!|\nu{*}xvec|\!)\langle N\rangle \wedge B = (\!|\nu{*}xvec|\!)(N \prec' P))$
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 $(RBrIn\ M\ N\ Q = \alpha \prec P) = (P = Q \wedge \alpha = ¿M(\!|N|\!))$
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 $(RBrOut\ M\ B = \alpha \prec P) = (\exists xvec\ N.\ \alpha = ¡M(\!|\nu{*}xvec|\!)\langle N\rangle \wedge B = (\!|\nu{*}xvec|\!)(N \prec' P))$
and $(RTau\ Q = \alpha \prec P) = (\alpha = \tau \wedge P = Q)$
proof −
show $(\alpha \prec P = RIn\ M\ N\ Q) = (P = Q \wedge \alpha = M(\!|N|\!))$
by(*nominal-induct* α *rule*: *action.strong-induct*)
(*auto simp add*: *residual.inject action.inject*)
next
show $(\alpha \prec P = RBrIn\ M\ N\ Q) = (P = Q \wedge \alpha = ¿M(\!|N|\!))$
by(*nominal-induct* α *rule*: *action.strong-induct*)
(*auto simp add*: *residual.inject action.inject*)
next
show $(\alpha \prec P = ROut\ M\ B) = (\exists xvec\ N.\ \alpha = M(\!|\nu{*}xvec|\!)\langle N\rangle \wedge B = (\!|\nu{*}xvec|\!)(N \prec' P))$
by(*nominal-induct* α *rule*: *action.strong-induct*)
(*auto simp add*: *residual.inject action.inject*)
next
show $(\alpha \prec P = RBrOut\ M\ B) = (\exists xvec\ N.\ \alpha = ¡M(\!|\nu{*}xvec|\!)\langle N\rangle \wedge B = (\!|\nu{*}xvec|\!)(N \prec' P))$
by(*nominal-induct* α *rule*: *action.strong-induct*)
(*auto simp add*: *residual.inject action.inject*)
next
show $(\alpha \prec P = RTau\ Q) = (\alpha = \tau \wedge P = Q)$
by(*nominal-induct* α *rule*: *action.strong-induct*)

(*auto simp add*: *residual.inject action.inject*)
next
show (*RIn M N Q* = $\alpha \prec P$) = ($P = Q \wedge \alpha = M(\!|N|\!)$)
by(*nominal-induct* α *rule*: *action.strong-induct*)
(*auto simp add*: *residual.inject action.inject*)
next
show (*RBrIn M N Q* = $\alpha \prec P$) = ($P = Q \wedge \alpha = ¿M(\!|N|\!)$)
by(*nominal-induct* α *rule*: *action.strong-induct*)
(*auto simp add*: *residual.inject action.inject*)
next
show (*ROut M B* = $\alpha \prec P$) = ($\exists$ *xvec N*. $\alpha = M(\!|\nu{*}xvec|\!)\langle N\rangle \wedge B = (\!|\nu{*}xvec|\!)(N \prec' P)$)
by(*nominal-induct* α *rule*: *action.strong-induct*)
(*auto simp add*: *residual.inject action.inject*)
next
show (*RBrOut M B* = $\alpha \prec P$) = ($\exists$ *xvec N*. $\alpha = ¡M(\!|\nu{*}xvec|\!)\langle N\rangle \wedge B = (\!|\nu{*}xvec|\!)(N \prec' P)$)
by(*nominal-induct* α *rule*: *action.strong-induct*)
(*auto simp add*: *residual.inject action.inject*)
next
show (*RTau Q* = $\alpha \prec P$) = ($\alpha = \tau \wedge P = Q$)
by(*nominal-induct* α *rule*: *action.strong-induct*)
(*auto simp add*: *residual.inject action.inject*)
qed

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

shows *xvec* $\sharp*$ ($M(\!|N|\!) \prec P$) = (*xvec* $\sharp*$ *M* $\wedge$ *xvec* $\sharp*$ *N* $\wedge$ *xvec* $\sharp*$ *P*)
and *xvec* $\sharp*$ ($¿M(\!|N|\!) \prec P$) = (*xvec* $\sharp*$ *M* $\wedge$ *xvec* $\sharp*$ *N* $\wedge$ *xvec* $\sharp*$ *P*)
and *xvec* $\sharp*$ ($M(\!|\nu{*}yvec|\!)\langle N\rangle \prec P$) = (*xvec* $\sharp*$ *M* $\wedge$ *xvec* $\sharp*$ ($(\!|\nu{*}yvec|\!)(N \prec' P)$))
and *xvec* $\sharp*$ ($¡M(\!|\nu{*}yvec|\!)\langle N\rangle \prec P$) = (*xvec* $\sharp*$ *M* $\wedge$ *xvec* $\sharp*$ ($(\!|\nu{*}yvec|\!)(N \prec' P)$))
and *xvec* $\sharp*$ ($\tau \prec P$) = (*xvec* $\sharp*$ *P*)
and *X* $\sharp*$ ($M(\!|N|\!) \prec P$) = (*X* $\sharp*$ *M* $\wedge$ *X* $\sharp*$ *N* $\wedge$ *X* $\sharp*$ *P*)
and *X* $\sharp*$ ($¿M(\!|N|\!) \prec P$) = (*X* $\sharp*$ *M* $\wedge$ *X* $\sharp*$ *N* $\wedge$ *X* $\sharp*$ *P*)
and *X* $\sharp*$ ($M(\!|\nu{*}yvec|\!)\langle N\rangle \prec P$) = (*X* $\sharp*$ *M* $\wedge$ *X* $\sharp*$ ($(\!|\nu{*}yvec|\!)(N \prec' P)$))
and *X* $\sharp*$ ($¡M(\!|\nu{*}yvec|\!)\langle N\rangle \prec P$) = (*X* $\sharp*$ *M* $\wedge$ *X* $\sharp*$ ($(\!|\nu{*}yvec|\!)(N \prec' P)$))
and *X* $\sharp*$ ($\tau \prec P$) = (*X* $\sharp*$ *P*)
by(*auto simp add*: *fresh-star-def*)

lemma *residualFreshChainSimp2*[*simp*]:
fixes *xvec* :: *name list*
and *X* :: *name set*
and *M* :: *'a*::*fs-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 $xvec \sharp* (RBrIn\ M\ N\ P) = (xvec \sharp* M \wedge xvec \sharp* N \wedge xvec \sharp* P)$
and $xvec \sharp* (ROut\ M\ B) = (xvec \sharp* M \wedge xvec \sharp* B)$
and $xvec \sharp* (RBrOut\ M\ B) = (xvec \sharp* M \wedge xvec \sharp* B)$
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 $X \sharp* (RBrIn\ M\ N\ P) = (X \sharp* M \wedge X \sharp* N \wedge X \sharp* P)$
and $X \sharp* (ROut\ M\ B) = (X \sharp* M \wedge X \sharp* B)$
and $X \sharp* (RBrOut\ M\ B) = (X \sharp* M \wedge X \sharp* B)$
and $X \sharp* (RTau\ P) = (X \sharp* P)$
by(*auto simp add*: *fresh-star-def*)

lemma *freshResidual3*[*dest*]:
fixes x :: $name$
and α :: ($'a$::$fs\text{-}name$) $action$
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** $x \sharp P$
using *assms*
by(*nominal-induct rule*: *action.strong-induct*) *auto*

lemma *freshResidualChain3*[*dest*]:
fixes $xvec$ :: $name\ list$
and α :: ($'a$::$fs\text{-}name$) $action$
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** $xvec \sharp* P$
using *assms*
by(*nominal-induct rule*: *action.strong-induct*) *auto*

lemma *freshResidual4*[*dest*]:
fixes x :: $name$
and α :: ($'a$::$fs\text{-}name$) $action$
and P :: ($'a$, $'b$::$fs\text{-}name$, $'c$::$fs\text{-}name$) psi

assumes $x \sharp \alpha \prec P$

shows $x \sharp subject\ \alpha$
using *assms*

by(*nominal-induct rule*: *action.strong-induct*) *auto*

lemma *freshResidualChain4*[*dest*]:
fixes *xvec* :: *name list*
and α :: ('*a*::*fs-name*) *action*
and *P* :: ('*a*, '*b*::*fs-name*, '*c*::*fs-name*) *psi*

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

shows *xvec* $\sharp*$ *subject* α
using *assms*
by(*nominal-induct rule*: *action.strong-induct*) *auto*

lemma *alphaOutputResidual*:
fixes *M* :: '*a*::*fs-name*
and *xvec* :: *name list*
and *N* :: '*a*
and *P* :: ('*a*, '*b*::*fs-name*, '*c*::*fs-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 $M\langle\!|\nu*yvec|\!\rangle\langle N\rangle \prec P = M\langle\!|\nu*(p \cdot yvec)|\!\rangle\langle(p \cdot N)\rangle \prec (p \cdot P)$
and ¡$M\langle\!|\nu*yvec|\!\rangle\langle N\rangle \prec P$ = ¡$M\langle\!|\nu*(p \cdot yvec)|\!\rangle\langle(p \cdot N)\rangle \prec (p \cdot P)$
using *assms*
by(*simp add*: *boundOutputChainAlpha''*)+

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

lemma *residualInject''*:

assumes *bn* α = *bn* β

shows ($\alpha \prec P = \beta \prec Q$) = ($\alpha = \beta \wedge P = Q$)
using *assms*
by(*nominal-induct* α *rule*: *action.strong-induct*)
(*force simp add*: *residual.inject create-residual.simps residualInject' action.inject boundOutput.inject*)+

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

lemma *bnFreshResidual*[*simp*]:
fixes α :: ('*a*::*fs-name*) *action*

shows (*bn* α) $\sharp*$ ($\alpha \prec P$) = *bn* α $\sharp*$ (*subject* α)

by(*nominal-induct* α *rule*: *action.strong-induct*)
(*auto simp add*: *residualFresh fresh-some fresh-star-def*)

lemma *actionCases*[*case-names cInput cBrInput cOutput cBrOutput cTau*]:
fixes α :: (*'a*::*fs-name*) *action*

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

shows *Prop*
using *assms*
by(*nominal-induct* α *rule*: *action.strong-induct*) *auto*

lemma *actionPar1Dest*:
fixes α :: (*'a*::*fs-name*) *action*
and P :: (*'a*, *'b*::*fs-name*, *'c*::*fs-name*) *psi*
and β :: (*'a*::*fs-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* α $\sharp*$ *bn* β

obtains $T\ p$ **where** *set* $p \subseteq$ *set*(*bn* α) $\times$ *set*(*bn* β) **and** $P = T \parallel (p \cdot R)$ **and** $\alpha \prec T = \beta \prec Q$
using *assms*
by(*cases rule*: *actionCases*[**where** α=α])
(*force simp add*: *residualInject dest*: *boundOutputPar1Dest'*)+

lemma *actionPar2Dest*:
fixes α :: (*'a*::*fs-name*) *action*
and P :: (*'a*, *'b*::*fs-name*, *'c*::*fs-name*) *psi*
and β :: (*'a*::*fs-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* α $\sharp*$ *bn* β

obtains $T\ p$ **where** *set* $p \subseteq$ *set*(*bn* α) $\times$ *set*(*bn* β) **and** $P = (p \cdot Q) \parallel T$ **and** $\alpha \prec T = \beta \prec R$
using *assms*
by(*cases rule*: *actionCases*[**where** α=α])
(*force simp add*: *simp add*: *residualInject dest*: *boundOutputPar2Dest'*)+

lemma *actionScopeDest*:

fixes α :: ($'a$::*fs-name*) *action*
and P :: ($'a$, $'b$::*fs-name*, $'c$::*fs-name*) *psi*
fixes β :: ($'a$::*fs-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** $\alpha \prec R = \beta \prec Q$
using *assms boundOutputScopeDest*
by(*cases rule*: *actionCases*[**where** α=α]) (*force simp add*: *residualInject*)+

lemma *emptyFreshName*:
fixes x :: *name*
and M :: $'a$::*fs-name*

assumes *supp* M = ({}::*name set*)

shows $x \sharp M$
using *assms*
by(*auto simp add*: *fresh-def*)

lemma *emptyFresh*:
fixes *xvec* :: *name list*
and M :: $'a$::*fs-name*

assumes *supp* M = ({}::*name set*)

shows *xvec* $\sharp*$ M
using *assms* **by** (*induct xvec*, *auto simp add*: *emptyFreshName*)

lemma *permEmptyEq*:
fixes p :: *name prm*
and M :: $'a$::*fs-name*

assumes *suppE*: *supp* M = ({}::*name set*)

shows $(p \cdot M) = M$
proof(*induct p*)
case *Nil*
then show *?case* **by** *simp*
next
case(*Cons a p*)
have $p \cdot M = M$ **by**(*rule Cons*)
then have $([a] \cdot p \cdot M) = [a] \cdot M$ **by** *simp*
then have $((a\#p) \cdot M) = [a] \cdot M$
by(*simp add*: *pt2*[*OF pt-name-inst*, *symmetric*])

then show *?case* **using** *suppE perm-fresh-fresh*
by(*cases a*) (*simp add*: *fresh-def*)
qed

abbreviation
outputJudge (‹-⟨-⟩› [*110*, *110*] *110*) **where** $M\langle N\rangle \equiv M(\!|\nu*([])|\!)\langle N\rangle$

abbreviation
brOutputJudge (‹¡-⟨-⟩› [*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′ SOutCon′ SInCon′
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*
and *SOutCon′* :: *′a* ⇒ *′a* ⇒ *′c*
and *SInCon′* :: *′a* ⇒ *′a* ⇒ *′c*
begin
notation *SCompose′* (**infixr** ‹⊗› *90*)
notation *SImp′* (‹- ⊢ -› [*85*, *85*] *85*)
notation *FrameImp* (‹- $\vdash_F$ -› [*85*, *85*] *85*)
abbreviation
FBottomJudge (‹$\bot_F$› *90*) **where** $\bot_F \equiv (FAssert\ SBottom')$
notation *SChanEq′* (‹- ↔ -› [*90*, *90*] *90*)
notation *SOutCon′* (‹- ≺ -› [*90*, *90*] *90*)
notation *SInCon′* (‹- ≻ -› [*90*, *90*] *90*)
notation *substTerm* (‹-[-::=-]› [*100*, *100*, *100*] *100*)
notation *subs* (‹-[-::=-]› [*100*, *100*, *100*] *100*)
notation *AssertionStatEq* (‹- ≃ -› [*80*, *80*] *80*)
notation *FrameStatEq* (‹- $\simeq_F$ -› [*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* ↔ *K*; *distinct xvec*; *set xvec* ⊆ *supp N*; *xvec* ♯* *Tvec*;
length xvec = *length Tvec*;
xvec ♯* Ψ; *xvec* ♯* *M*; *xvec* ♯* *K*⟧ ⟹ Ψ ▷ $M(\!|\lambda*xvec\ N|\!).P$ ⟼ $K(\!|(N[xvec::=Tvec])|\!)$ ≺ $P[xvec::=Tvec]$
| *cBrInput*:⟦Ψ ⊢ *K* ≻ *M*; *distinct xvec*; *set xvec* ⊆ *supp N*; *xvec* ♯* *Tvec*;
length xvec = *length Tvec*;

$xvec \sharp* \Psi$; $xvec \sharp* M$; $xvec \sharp* K]\!] \Longrightarrow \Psi \rhd M(\!|\lambda * xvec\ N|\!).P \longmapsto ¿K(\!|(N[xvec ::= Tvec])|\!) \prec P[xvec ::= Tvec]$

| *Output*: $[\![\Psi \vdash M \leftrightarrow K]\!] \Longrightarrow \Psi \rhd M\langle N\rangle.P \longmapsto K\langle N\rangle \prec P$

| *BrOutput*: $[\![\Psi \vdash M \preceq K]\!] \Longrightarrow \Psi \rhd M\langle N\rangle.P \longmapsto ¡K\langle N\rangle \prec P$

| *Case*: $[\![\Psi \rhd P \longmapsto Rs; (\varphi, P) \in set\ Cs; \Psi \vdash \varphi; guarded\ P]\!] \Longrightarrow \Psi \rhd Cases\ Cs \longmapsto Rs$

| *cPar1*: $[\![(\Psi \otimes \Psi_Q) \rhd P \longmapsto \alpha \prec P'; extractFrame\ Q = \langle A_Q, \Psi_Q\rangle; distinct\ A_Q;$
$A_Q \sharp* P; A_Q \sharp* Q; A_Q \sharp* \Psi; A_Q \sharp* \alpha; A_Q \sharp* P'; distinct(bn\ \alpha);$
$bn\ \alpha \sharp* \Psi; bn\ \alpha \sharp* \Psi_Q; bn\ \alpha \sharp* Q; bn\ \alpha \sharp* P; bn\ \alpha \sharp* (subject\ \alpha)]\!] \Longrightarrow$
$\Psi \rhd P \parallel Q \longmapsto \alpha \prec (P' \parallel Q)$

| *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*: $[\![\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P'; extractFrame\ P = \langle A_P, \Psi_P\rangle; distinct\ A_P;$
$\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|N|\!) \prec Q'; extractFrame\ Q = \langle A_Q, \Psi_Q\rangle; distinct\ A_Q;$
$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K;$
$A_P \sharp* \Psi; A_P \sharp* \Psi_Q; A_P \sharp* P; A_P \sharp* M; A_P \sharp* N; A_P \sharp* P';$
$A_P \sharp* Q; A_P \sharp* Q'; A_P \sharp* A_Q; A_P \sharp* xvec;$
$A_Q \sharp* \Psi; A_Q \sharp* \Psi_P; A_Q \sharp* P; A_Q \sharp* N; A_Q \sharp* P';$
$A_Q \sharp* Q; A_Q \sharp* K; A_Q \sharp* Q'; A_Q \sharp* xvec; distinct\ xvec;$
$xvec \sharp* \Psi; xvec \sharp* \Psi_P; xvec \sharp* \Psi_Q; xvec \sharp* P; xvec \sharp* M;$
$xvec \sharp* Q; xvec \sharp* K]\!] \Longrightarrow$
$\Psi \rhd P \parallel Q \longmapsto \tau \prec (\!|\nu * xvec|\!)(P' \parallel Q')$

| *cBrMerge*: $[\![\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'; extractFrame\ P = \langle A_P, \Psi_P\rangle; distinct\ A_P;$
$\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'; extractFrame\ Q = \langle A_Q, \Psi_Q\rangle; distinct\ A_Q;$
$A_P \sharp* \Psi; A_P \sharp* \Psi_Q; A_P \sharp* P; A_P \sharp* N; A_P \sharp* P';$
$A_P \sharp* Q; A_P \sharp* Q'; A_P \sharp* A_Q;$
$A_P \sharp* M; A_Q \sharp* M;$
$A_Q \sharp* \Psi; A_Q \sharp* \Psi_P; A_Q \sharp* P; A_Q \sharp* N; A_Q \sharp* P';$
$A_Q \sharp* Q; A_Q \sharp* Q']\!] \Longrightarrow$
$\Psi \rhd P \parallel Q \longmapsto ¿M(\!|N|\!) \prec (P' \parallel Q')$

| *cBrComm1*: $[\![\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$; *extractFrame* $P = \langle A_P, \Psi_P\rangle$; *distinct* A_P;
$\Psi \otimes \Psi_P \rhd Q \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec Q'$; *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$; *distinct* A_Q;
$A_P \sharp* \Psi$; $A_P \sharp* \Psi_Q$; $A_P \sharp* P$; $A_P \sharp* N$; $A_P \sharp* P'$;
$A_P \sharp* Q$; $A_P \sharp* Q'$; $A_P \sharp* A_Q$; $A_P \sharp* xvec$;
$A_P \sharp* M$; $A_Q \sharp* M$;
$A_Q \sharp* \Psi$; $A_Q \sharp* \Psi_P$; $A_Q \sharp* P$; $A_Q \sharp* N$; $A_Q \sharp* P'$;
$A_Q \sharp* Q$; $A_Q \sharp* Q'$; $A_Q \sharp* xvec$; *distinct* $xvec$;
$xvec \sharp* \Psi$; $xvec \sharp* \Psi_P$; $xvec \sharp* \Psi_Q$; $xvec \sharp* P$;
$xvec \sharp* Q$; $xvec \sharp* M]\!] \Longrightarrow$
$\Psi \rhd P \parallel Q \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec (P' \parallel Q')$
| *cBrComm2*: $[\![\Psi \otimes \Psi_Q \rhd P \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$; *extractFrame* $P = \langle A_P, \Psi_P\rangle$; *distinct* A_P;
$\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$; *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$; *distinct* A_Q;
$A_P \sharp* \Psi$; $A_P \sharp* \Psi_Q$; $A_P \sharp* P$; $A_P \sharp* N$; $A_P \sharp* P'$;
$A_P \sharp* Q$; $A_P \sharp* Q'$; $A_P \sharp* A_Q$; $A_P \sharp* xvec$;
$A_P \sharp* M$; $A_Q \sharp* M$;
$A_Q \sharp* \Psi$; $A_Q \sharp* \Psi_P$; $A_Q \sharp* P$; $A_Q \sharp* N$; $A_Q \sharp* P'$;
$A_Q \sharp* Q$; $A_Q \sharp* Q'$; $A_Q \sharp* xvec$; *distinct* $xvec$;
$xvec \sharp* \Psi$; $xvec \sharp* \Psi_P$; $xvec \sharp* \Psi_Q$; $xvec \sharp* P$;
$xvec \sharp* Q$; $xvec \sharp* M]\!] \Longrightarrow$
$\Psi \rhd P \parallel Q \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec (P' \parallel Q')$
| *cBrClose*: $[\![\Psi \rhd P \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$;
$x \in supp\ M$;
distinct $xvec$; $xvec \sharp* \Psi$; $xvec \sharp* P$;
$xvec \sharp* M$;
$x \sharp \Psi$; $x \sharp xvec]\!] \Longrightarrow$
$\Psi \rhd (\!|\nu x|\!)P \longmapsto \tau \prec (\!|\nu x|\!)((\!|\nu * xvec|\!)P')$
| *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* $yvec$;
$xvec \sharp* \Psi$; $xvec \sharp* P$; $xvec \sharp* M$; $xvec \sharp* yvec$; $yvec \sharp* \Psi$; $yvec \sharp* P$; $yvec \sharp* M]\!] \Longrightarrow$
$\Psi \rhd (\!|\nu x|\!)P \longmapsto M(\!|\nu * (xvec@x\#yvec)|\!)\langle N\rangle \prec P'$
| *cBrOpen*: $[\![\Psi \rhd P \longmapsto ¡M(\!|\nu * (xvec@yvec)|\!)\langle N\rangle \prec P'$; $x \in supp\ N$; $x \sharp xvec$; $x \sharp yvec$; $x \sharp M$; $x \sharp \Psi$;
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 $(\langle\text{-} \longmapsto \text{-}\rangle\ [50, 50]\ 50)$ **where** $P \longmapsto Rs \equiv \mathbf{1} \rhd P \longmapsto Rs$

equivariance *env.semantics*

nominal-inductive2 *env.semantics*
avoids *cInput*: *set xvec*
| *cBrInput*: *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*
| *cBrMerge*: *set* $A_P \cup$ *set* A_Q
| *cBrComm1*: *set* $A_P \cup$ *set* $A_Q \cup$ *set xvec*
| *cBrComm2*: *set* $A_P \cup$ *set* $A_Q \cup$ *set xvec*
| *cBrClose*: $\{x\} \cup$ *set xvec*
| *cOpen*: $\{x\} \cup$ *set xvec* $\cup$ *set yvec*
| *cBrOpen*: $\{x\} \cup$ *set xvec* $\cup$ *set yvec*
| *cScope*: $\{x\} \cup set(bn\ \alpha)$
apply −
apply(*force intro*: *substTerm.subst4Chain subst4Chain simp add*: *abs-fresh residualFresh*)+
apply(*force intro*: *substTerm.subst4Chain subst4Chain simp add*: *abs-fresh residualFresh boundOutputFresh boundOutputFreshSet fresh-star-def resChainFresh*)+
done

lemma *nilTrans1*:
fixes Ψ :: *'b*
and M :: *'a*
and *xvec* :: *name list*
and N :: *'a*
and P :: (*'a*, *'b*, *'c*) *psi*

assumes $\Psi \rhd \mathbf{0} \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P$

shows *False*
using *assms*
apply −
by (*ind-cases* $\Psi \rhd \mathbf{0} \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P$)

lemma *nilTrans1'*:
fixes Ψ :: *'b*
and M :: *'a*
and *xvec* :: *name list*
and N :: *'a*
and P :: (*'a*, *'b*, *'c*) *psi*

assumes $\Psi \rhd \mathbf{0} \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec P$

shows *False*

using *assms*
apply −
by (*ind-cases* $\Psi \rhd \mathbf{0} \longmapsto ¡M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P$)

lemma *nilTrans2*:
fixes Ψ :: $'b$
and *Rs* :: ($'a$, $'b$, $'c$) *residual*

assumes $\Psi \rhd \mathbf{0} \longmapsto Rs$

shows *False*
using *assms*
apply(*cases rule*: *semantics.cases*)
by(*auto simp add*: *residualInject*)+

lemma *nilTrans3*:
fixes Ψ :: $'b$
and M :: $'a$
and M' :: $'a$
and *xvec* :: *name list*
and *yvec* :: *name list*
and N :: $'a$
and N' :: $'a$
and P :: ($'a$, $'b$, $'c$) *psi*
and P' :: ($'a$, $'b$, $'c$) *psi*

assumes $\Psi \rhd M(\!|\lambda{*}xvec\ N|\!).P \longmapsto M'(\!|\nu{*}yvec|\!)\langle N'\rangle \prec P'$

shows *False*
using *assms*
apply −
by(*ind-cases* $\Psi \rhd M(\!|\lambda{*}xvec\ N|\!).P \longmapsto M'(\!|\nu{*}yvec|\!)\langle N'\rangle \prec P'$) (*auto simp add*: *residualInject*)

lemma *nilTrans3'*:
fixes Ψ :: $'b$
and M :: $'a$
and M' :: $'a$
and *xvec* :: *name list*
and *yvec* :: *name list*
and N :: $'a$
and N' :: $'a$
and P :: ($'a$, $'b$, $'c$) *psi*
and P' :: ($'a$, $'b$, $'c$) *psi*

assumes $\Psi \rhd M(\!|\lambda{*}xvec\ N|\!).P \longmapsto ¡M'(\!|\nu{*}yvec|\!)\langle N'\rangle \prec P'$

shows *False*
using *assms*

apply −
by(*ind-cases* $\Psi \rhd M(\!|\lambda * xvec\ N|\!).P \longmapsto ¡M'(\!|\nu * yvec|\!)\langle N'\rangle \prec P'$) (*auto simp add*: *residualInject*)

lemma *nilTrans4*:
fixes Ψ :: $'b$
and *Rs* :: ($'a$, $'b$, $'c$) *residual*

assumes $\Psi \rhd M(\!|\lambda * xvec\ N|\!).P \longmapsto\!\!\!\!\!\!\tau\ \prec P'$

shows *False*
using *assms*
apply(*cases rule*: *semantics.cases*)
by(*auto simp add*: *residualInject*)+

lemma *nilTrans5*:
fixes Ψ :: $'b$
fixes Ψ' :: $'b$
and *M* :: $'a$
and *xvec* :: *name list*
and *N* :: $'a$
and *P* :: ($'a$, $'b$, $'c$) *psi*

assumes $\Psi \rhd \{\!|\Psi'|\!\} \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P$

shows *False*
using *assms*
apply −
by(*ind-cases* $\Psi \rhd \{\!|\Psi'|\!\} \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P$)

lemma *nilTrans5'*:
fixes Ψ :: $'b$
fixes Ψ' :: $'b$
and *M* :: $'a$
and *xvec* :: *name list*
and *N* :: $'a$
and *P* :: ($'a$, $'b$, $'c$) *psi*

assumes $\Psi \rhd \{\!|\Psi'|\!\} \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec P$

shows *False*
using *assms*
apply −
by(*ind-cases* $\Psi \rhd \{\!|\Psi'|\!\} \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec P$)

lemma *nilTrans6*:
fixes Ψ :: $'b$
and *Rs* :: ($'a$, $'b$, $'c$) *residual*

assumes $\Psi \rhd \{\!|\Psi'|\!\} \longmapsto Rs$

shows *False*
using *assms*
apply(*cases rule*: *semantics.cases*)
by(*auto simp add*: *residualInject*)+

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 $\Psi \rhd M(\!|\lambda * xvec\ N|\!).P \longmapsto ¡K(\!|\nu * yvec|\!)\langle N'\rangle \prec P' \Longrightarrow$ *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 $\Psi \rhd M\langle N\rangle.P \longmapsto ¿K(\!|N'|\!) \prec P' \Longrightarrow$ *False*
and $\Psi \rhd M\langle N\rangle.P \longmapsto \tau \prec P' \Longrightarrow$ *False*
and $\Psi \rhd \{\!|\Psi'|\!\} \longmapsto Rs \Longrightarrow$ *False*
apply −
apply(*rule nilTrans2*)
apply *assumption*
apply(*cases rule*: *semantics.cases*) **apply**(*force simp add*: *residualInject*)+
apply(*cases rule*: *semantics.cases*) **apply**(*force simp add*: *residualInject*)+
apply(*rule nilTrans4*)
apply *assumption*
apply(*cases rule*: *semantics.cases*) **apply**(*force simp add*: *residualInject*)+
apply(*cases rule*: *semantics.cases*) **apply**(*force simp add*: *residualInject*)+
apply(*cases rule*: *semantics.cases*) **apply**(*force simp add*: *residualInject*)+
apply(*rule nilTrans6*)
by *assumption*

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** $bn(p \cdot \alpha)\ \sharp*\ P$
using *assms*
proof(*nominal-induct* α *rule*: *action.strong-induct*)
case(*In M N*)
then show *?case* **by**(*simp add*: *residualInject*)
next
case(*BrIn M N*)
then show *?case* **by**(*simp add*: *residualInject*)
next
case(*Out M xvec N*)
then show *?case*
using *boundOutputChainEq''* **by**(*force simp add*: *residualInject*)
next
case(*BrOut M xvec N*)
then show *?case*
using *boundOutputChainEq''* **by**(*force simp add*: *residualInject*)
next
case *Tau*
then show *?case* **by**(*simp add*: *residualInject*)
qed

lemma *semanticsInduct*[*consumes 3*, *case-names cAlpha cInput cBrInput cOutput cBrOutput cCase cPar1 cPar2 cComm1 cComm2 cBrMerge cBrComm1 cBrComm2 cBrClose cOpen cBrOpen cScope cBang*]:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)\ psi$
and α :: $'a\ action$
and P' :: $('a, 'b, 'c)\ psi$
and *Prop* :: $'f$::*fs-name* $\Rightarrow$ $'b \Rightarrow ('a, 'b, 'c)\ psi \Rightarrow$
$'a\ action \Rightarrow ('a, 'b, 'c)\ psi \Rightarrow bool$
and C :: $'f$::*fs-name*

assumes $\Psi \rhd P \longmapsto \alpha \prec P'$
and $bn\ \alpha\ \sharp*\ (subject\ \alpha)$
and $distinct(bn\ \alpha)$
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 *rBrInput*:$\bigwedge \Psi\ K\ M\ xvec\ N\ Tvec\ P\ C.$
$[\![\Psi \vdash K \succeq M;\ distinct\ xvec;\ set\ xvec \subseteq supp\ N;$
$length\ xvec = length\ Tvec;\ xvec \sharp* \Psi;$
$xvec \sharp* M;\ xvec \sharp* K;\ xvec \sharp* C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (M(\!|\lambda* xvec\ N|\!).P)$
$(¿K(\!|(N[xvec::=Tvec])|\!))\ (P[xvec::=Tvec])$

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 *rBrOutput*:$\bigwedge \Psi\ M\ K\ N\ P\ C.\ [\![\Psi \vdash M \preceq K]\!] \Longrightarrow Prop\ C\ \Psi\ (M\langle N\rangle.P)$
$(¡K\langle N\rangle)\ P$

and *rCase*: $\bigwedge \Psi\ P\ \alpha\ P'\ \varphi\ Cs\ C.\ [\![\Psi \rhd P \longmapsto \alpha \prec P';\ \bigwedge C.\ Prop\ C\ \Psi\ P\ \alpha\ P';$
$(\varphi,\ P) \in set\ Cs;\ \Psi \vdash \varphi;\ guarded\ P]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (Cases\ Cs)\ \alpha\ P'$

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$$\rrbracket \Longrightarrow$
Prop $C\ \Psi\ (P \parallel Q)\ (\tau)\ (\langle\!|\nu* xvec|\!\rangle(P' \parallel Q'))$

and *rComm2*: $\bigwedge \Psi\ \Psi_Q\ P\ M\ xvec\ N\ P'\ A_P\ \Psi_P\ Q\ K\ Q'\ A_Q\ C.$
$\llbracket \Psi \otimes \Psi_Q \rhd P \longmapsto M\langle\!|\nu* xvec|\!\rangle\langle N\rangle \prec P'$; $\bigwedge C.$ *Prop* $C\ (\Psi \otimes \Psi_Q)\ P\ (M\langle\!|\nu* xvec|\!\rangle\langle N\rangle)\ P'$;
extractFrame $P = \langle A_P, \Psi_P\rangle$; *distinct* A_P;
$\Psi \otimes \Psi_P \rhd Q \longmapsto K\langle\!|N|\!\rangle \prec Q'$; $\bigwedge C.$ *Prop* $C\ (\Psi \otimes \Psi_P)\ Q\ (K\langle\!|N|\!\rangle)\ Q'$;
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$$\rrbracket \Longrightarrow$
Prop $C\ \Psi\ (P \parallel Q)\ (\tau)\ (\langle\!|\nu* xvec|\!\rangle(P' \parallel Q'))$

and *rBrMerge*: $\bigwedge \Psi\ \Psi_Q\ P\ M\ N\ P'\ A_P\ \Psi_P\ Q\ Q'\ A_Q\ C.$
$\llbracket \Psi \otimes \Psi_Q \rhd P \longmapsto ¿M\langle\!|N|\!\rangle \prec P'$; $\bigwedge C.$ *Prop* $C\ (\Psi \otimes \Psi_Q)\ P\ (¿M\langle\!|N|\!\rangle)\ P'$;
extractFrame $P = \langle A_P, \Psi_P\rangle$; *distinct* A_P;
$\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M\langle\!|N|\!\rangle \prec Q'$; $\bigwedge C.$ *Prop* $C\ (\Psi \otimes \Psi_P)\ Q\ (¿M\langle\!|N|\!\rangle)\ Q'$;
extractFrame $Q = \langle A_Q, \Psi_Q\rangle$; *distinct* A_Q;
$A_P \sharp* \Psi$; $A_P \sharp* \Psi_Q$; $A_P \sharp* P$; $A_P \sharp* N$; $A_P \sharp* P'$;
$A_P \sharp* Q$; $A_P \sharp* Q'$; $A_P \sharp* A_Q$; $A_P \sharp* M$; $A_Q \sharp* M$;
$A_Q \sharp* \Psi$; $A_Q \sharp* \Psi_P$; $A_Q \sharp* P$; $A_Q \sharp* N$; $A_Q \sharp* P'$;
$A_Q \sharp* Q$; $A_Q \sharp* Q'$; $A_P \sharp* C$; $A_Q \sharp* C$$\rrbracket \Longrightarrow$
Prop $C\ \Psi\ (P \parallel Q)\ (¿M\langle\!|N|\!\rangle)\ (P' \parallel Q')$

and *rBrComm1*: $\bigwedge \Psi\ \Psi_Q\ P\ M\ N\ P'\ A_P\ \Psi_P\ Q\ xvec\ Q'\ A_Q\ C.$
$\llbracket \Psi \otimes \Psi_Q \rhd P \longmapsto ¿M\langle\!|N|\!\rangle \prec P'$; $\bigwedge C.$ *Prop* $C\ (\Psi \otimes \Psi_Q)\ P\ (¿M\langle\!|N|\!\rangle)\ P'$;
extractFrame $P = \langle A_P, \Psi_P\rangle$; *distinct* A_P;
$\Psi \otimes \Psi_P \rhd Q \longmapsto ¡M\langle\!|\nu* xvec|\!\rangle\langle N\rangle \prec Q'$; $\bigwedge C.$ *Prop* $C\ (\Psi \otimes \Psi_P)\ Q\ (¡M\langle\!|\nu* xvec|\!\rangle\langle N\rangle)\ Q'$;
extractFrame $Q = \langle A_Q, \Psi_Q\rangle$; *distinct* A_Q;
$A_P \sharp* \Psi$; $A_P \sharp* \Psi_Q$; $A_P \sharp* P$; $A_P \sharp* N$; $A_P \sharp* P'$;
$A_P \sharp* Q$; $A_P \sharp* Q'$; $A_P \sharp* A_Q$; $A_P \sharp* xvec$; $A_Q \sharp* \Psi$; $A_Q \sharp* \Psi_P$;
$A_Q \sharp* P$; $A_Q \sharp* N$; $A_Q \sharp* P'$; $A_Q \sharp* Q$; $A_Q \sharp* Q'$; *distinct xvec*;
$A_P \sharp* M$; $A_Q \sharp* M$; $xvec \sharp* M$;
$A_Q \sharp* xvec$; $xvec \sharp* \Psi$; $xvec \sharp* \Psi_P$; $xvec \sharp* \Psi_Q$; $xvec \sharp* P$;
$xvec \sharp* Q$; $A_P \sharp* C$; $A_Q \sharp* C$; $xvec \sharp* C$$\rrbracket \Longrightarrow$
Prop $C\ \Psi\ (P \parallel Q)\ (¡M\langle\!|\nu* xvec|\!\rangle\langle N\rangle)\ (P' \parallel Q')$

and *rBrComm2*: $\bigwedge \Psi\ \Psi_Q\ P\ M\ xvec\ N\ P'\ A_P\ \Psi_P\ Q\ Q'\ A_Q\ C.$
$\llbracket \Psi \otimes \Psi_Q \rhd P \longmapsto ¡M\langle\!|\nu* xvec|\!\rangle\langle N\rangle \prec P'$; $\bigwedge C.$ *Prop* $C\ (\Psi \otimes \Psi_Q)\ P\ (¡M\langle\!|\nu* xvec|\!\rangle\langle N\rangle)\ P'$;

$extractFrame\ P = \langle A_P, \Psi_P\rangle$; $distinct\ A_P$;
$\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$; $\bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_P)\ Q\ (¿M(\!|N|\!))\ Q'$;
$extractFrame\ Q = \langle A_Q, \Psi_Q\rangle$; $distinct\ A_Q$;
$A_P \sharp* \Psi$; $A_P \sharp* \Psi_Q$; $A_P \sharp* P$; $A_P \sharp* N$; $A_P \sharp* P'$;
$A_P \sharp* Q$; $A_P \sharp* Q'$; $A_P \sharp* A_Q$; $A_P \sharp* xvec$; $A_Q \sharp* \Psi$; $A_Q \sharp* \Psi_P$;
$A_Q \sharp* P$; $A_Q \sharp* N$; $A_Q \sharp* P'$; $A_Q \sharp* Q$; $A_Q \sharp* Q'$; $distinct\ xvec$;
$A_P \sharp* M$; $A_Q \sharp* M$; $xvec \sharp* M$;
$A_Q \sharp* xvec$; $xvec \sharp* \Psi$; $xvec \sharp* \Psi_P$; $xvec \sharp* \Psi_Q$; $xvec \sharp* P$;
$xvec \sharp* Q$; $A_P \sharp* C$; $A_Q \sharp* C$; $xvec \sharp* C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (P \parallel Q)\ (¡M(\!|\nu*xvec|\!)\langle N\rangle)\ (P' \parallel Q')$

and *rBrClose*: $\bigwedge \Psi\ P\ M\ xvec\ N\ P'\ x\ C.$
$[\![\Psi \rhd P \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec P'$;
$\bigwedge C.\ Prop\ C\ \Psi\ P\ (¡M(\!|\nu*xvec|\!)\langle N\rangle)\ P'$;
$x \in supp\ M$;
$distinct\ xvec$; $xvec \sharp* \Psi$; $xvec \sharp* P$;
$xvec \sharp* M$;
$x \sharp \Psi$; $x \sharp xvec]\!] \Longrightarrow$
$Prop\ C\ \Psi\ ((\!|\nu x|\!)P)\ (\tau)\ ((\!|\nu x|\!)((\!|\nu*xvec|\!)P'))$

and *rOpen*: $\bigwedge \Psi\ P\ M\ xvec\ yvec\ N\ P'\ x\ C.$
$[\![\Psi \rhd P \longmapsto M(\!|\nu*(xvec@yvec)|\!)\langle N\rangle \prec P'$; $x \in supp\ N$; $\bigwedge C.\ Prop\ C\ \Psi\ P\ (M(\!|\nu*(xvec@yvec)|\!)\langle N\rangle)\ P'$;
$x \sharp \Psi$; $x \sharp M$; $x \sharp xvec$; $x \sharp yvec$; $xvec \sharp* \Psi$; $xvec \sharp* P$; $xvec \sharp* M$; $distinct\ xvec$; $distinct\ yvec$;
$yvec \sharp* \Psi$; $yvec \sharp* P$; $yvec \sharp* M$; $yvec \sharp* C$; $x \sharp C$; $xvec \sharp* C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ ((\!|\nu x|\!)P)\ (M(\!|\nu*(xvec@x\#yvec)|\!)\langle N\rangle)\ P'$

and *rBrOpen*: $\bigwedge \Psi\ P\ M\ xvec\ yvec\ N\ P'\ x\ C.$
$[\![\Psi \rhd P \longmapsto ¡M(\!|\nu*(xvec@yvec)|\!)\langle N\rangle \prec P'$; $x \in supp\ N$; $\bigwedge C.\ Prop\ C\ \Psi\ P\ (¡M(\!|\nu*(xvec@yvec)|\!)\langle N\rangle)\ P'$;
$x \sharp \Psi$; $x \sharp M$; $x \sharp xvec$; $x \sharp yvec$; $xvec \sharp* \Psi$; $xvec \sharp* P$; $xvec \sharp* M$; $distinct\ xvec$; $distinct\ yvec$;
$yvec \sharp* \Psi$; $yvec \sharp* P$; $yvec \sharp* M$; $yvec \sharp* C$; $x \sharp C$; $xvec \sharp* C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ ((\!|\nu x|\!)P)\ (¡M(\!|\nu*(xvec@x\#yvec)|\!)\langle N\rangle)\ P'$

and *rScope*: $\bigwedge \Psi\ P\ \alpha\ P'\ x\ C.$
$[\![\Psi \rhd P \longmapsto \alpha \prec P'$; $\bigwedge C.\ Prop\ C\ \Psi\ P\ \alpha\ P'$;
$x \sharp \Psi$; $x \sharp \alpha$; $bn\ \alpha \sharp* \Psi$;
$bn\ \alpha \sharp* P$; $bn\ \alpha \sharp* (subject\ \alpha)$; $x \sharp C$; $bn\ \alpha \sharp* C$; $distinct(bn\ \alpha)]\!] \Longrightarrow$
$Prop\ C\ \Psi\ ((\!|\nu x|\!)P)\ \alpha\ ((\!|\nu x|\!)P')$

and *rBang*: $\bigwedge \Psi\ P\ \alpha\ P'\ C.$
$[\![\Psi \rhd P \parallel !P \longmapsto \alpha \prec P'$; $guarded\ P$; $\bigwedge C.\ Prop\ C\ \Psi\ (P \parallel !P)\ \alpha\ P']\!] \Longrightarrow$
$Prop\ C\ \Psi\ (!P)\ \alpha\ P'$

shows $Prop\ C\ \Psi\ P\ \alpha\ P'$
using $\langle \Psi \rhd P \longmapsto \alpha \prec P'\rangle\ \langle bn\ \alpha \sharp* (subject\ \alpha)\rangle\ \langle distinct(bn\ \alpha)\rangle$
proof(*nominal-induct x3*==$\alpha \prec P'$ *avoiding*: α C *arbitrary*: P' *rule*: *semantics.strong-induct*)
case(*cInput* Ψ M K *xvec* N *Tvec* P α C P')

then show *?case* **by**(*force intro*: *rInput simp add*: *residualInject*)
next
case(*cBrInput* Ψ M K *xvec* N *Tvec* P α C P')
then show *?case*
by(*force simp add*: *rBrInput residualInject*)
next
case(*Output* Ψ M K N P α C P')
then show *?case* **by**(*force intro*: *rOutput simp add*: *residualInject*)
next
case(*BrOutput* Ψ M K N P α C P')
then show *?case* **by**(*force intro*: *rBrOutput simp add*: *residualInject*)
next
case(*Case* Ψ P φ *Cs* α C P')
then show *?case* **by**(*auto intro*: *rCase*)
next
case(*cPar1* Ψ Ψ_Q P α P' Q A_Q α' C P'')
note ‹$\alpha \prec (P' \parallel Q) = \alpha' \prec P''$›
moreover from ‹*bn* $\alpha \sharp* \alpha'$› **have** *bn* $\alpha \sharp* (bn\ \alpha')$ **by** *auto*
moreover note ‹*distinct* (*bn* α)› ‹*distinct*(*bn* α')›
moreover from ‹*bn* $\alpha \sharp*$ *subject* α› ‹*bn* $\alpha' \sharp*$ *subject* α'›
have *bn* $\alpha \sharp* (\alpha \prec P' \parallel Q)$ **and** *bn* $\alpha' \sharp* (\alpha' \prec P'')$ **by** *simp+*
ultimately obtain p **where** S: (*set* p) $\subseteq$ (*set*(*bn* α)) $\times$ (*set*(*bn*($p \cdot \alpha$))) **and** *distinctPerm* p
and αEq: $\alpha' = p \cdot \alpha$ **and** $P'eq$: $P'' = p \cdot (P' \parallel Q)$ **and** (*bn*($p \cdot \alpha$)) $\sharp* \alpha$
and (*bn*($p \cdot \alpha$)) $\sharp* (P' \parallel Q)$
by(*rule residualEq*)

note ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto \alpha \prec P'$› ‹*extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$› ‹*distinct* A_Q›
moreover from ‹*bn* $\alpha \sharp*$ *subject* α› ‹*distinct*(*bn* α)›
have $\bigwedge C$. *Prop* C $(\Psi \otimes \Psi_Q)$ P α P' **by**(*metis cPar1*)
moreover note ‹$A_Q \sharp* P$› ‹$A_Q \sharp* Q$› ‹$A_Q \sharp* \Psi$› ‹$A_Q \sharp* \alpha$› ‹$A_Q \sharp* P'$› ‹$A_Q \sharp* C$›
‹*bn* $\alpha \sharp* Q$› ‹*distinct*(*bn* α)› ‹*bn* $\alpha \sharp* \Psi$› ‹*bn* $\alpha \sharp* \Psi_Q$› ‹*bn* $\alpha \sharp* P$› ‹*bn* $\alpha \sharp*$ *subject* α› ‹*bn* $\alpha \sharp* C$›
ultimately have *Prop* C Ψ $(P \parallel Q)$ α $(P' \parallel Q)$
by(*metis rPar1*)

with ‹*bn* $\alpha \sharp* \Psi$› ‹*bn* $\alpha \sharp* P$› ‹*bn* $\alpha \sharp* Q$› ‹*bn* $\alpha \sharp*$ *subject* α› ‹*bn* $\alpha \sharp* C$› ‹*bn* $\alpha \sharp*$ *bn* α'› S ‹*distinctPerm* p› ‹*bn*($p \cdot \alpha$) $\sharp* \alpha$› ‹*bn*($p \cdot \alpha$) $\sharp* (P' \parallel Q)$› ‹$A_Q \sharp* C$›
have *Prop* C Ψ $(P \parallel Q)$ $(p \cdot \alpha)$ $(p \cdot (P' \parallel Q))$
by $-$ (*rule rAlpha*, *auto*)
with αEq $P'eq$ ‹*distinctPerm* p› **show** *?case* **by** *simp*
next
case(*cPar2* Ψ Ψ_P Q α Q' P A_P α' C Q'')
note ‹$\alpha \prec (P \parallel Q') = \alpha' \prec Q''$›
moreover from ‹*bn* $\alpha \sharp* \alpha'$› **have** *bn* $\alpha \sharp* (bn\ \alpha')$ **by** *auto*
moreover note ‹*distinct* (*bn* α)› ‹*distinct*(*bn* α')›
moreover from ‹*bn* $\alpha \sharp*$ *subject* α› ‹*bn* $\alpha' \sharp*$ *subject* α'›
have *bn* $\alpha \sharp* (\alpha \prec P \parallel Q')$ **and** *bn* $\alpha' \sharp* (\alpha' \prec Q'')$ **by** *simp+*

ultimately obtain p **where** S: $(set\ p) \subseteq (set(bn\ \alpha)) \times (set(bn(p \cdot \alpha)))$ **and** *distinctPerm p*
and αEq: $\alpha' = p \cdot \alpha$ **and** $Q'eq$: $Q'' = p \cdot (P \parallel Q')$ **and** $(bn(p \cdot \alpha)) \sharp* \alpha$
and $(bn(p \cdot \alpha)) \sharp* (P \parallel Q')$
by(*rule residualEq*)

note $\langle \Psi \otimes \Psi_P \rhd Q \longmapsto \alpha \prec Q' \rangle$ $\langle extractFrame\ P = \langle A_P, \Psi_P \rangle \rangle$ $\langle distinct\ A_P \rangle$
moreover from $\langle bn\ \alpha \sharp* subject\ \alpha \rangle$ $\langle distinct(bn\ \alpha) \rangle$
have $\bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_P)\ Q\ \alpha\ Q'$ **by**(*auto intro*: *cPar2*)

moreover note $\langle A_P \sharp* P \rangle$ $\langle A_P \sharp* Q \rangle$ $\langle A_P \sharp* \Psi \rangle$ $\langle A_P \sharp* \alpha \rangle$ $\langle A_P \sharp* Q' \rangle$ $\langle A_P \sharp* C \rangle$
$\langle bn\ \alpha \sharp* Q \rangle$ $\langle distinct(bn\ \alpha) \rangle$ $\langle bn\ \alpha \sharp* \Psi \rangle$ $\langle bn\ \alpha \sharp* \Psi_P \rangle$ $\langle bn\ \alpha \sharp* P \rangle$ $\langle bn\ \alpha \sharp* subject\ \alpha \rangle$ $\langle bn\ \alpha \sharp* C \rangle$
ultimately have $Prop\ C\ \Psi\ (P \parallel Q)\ \alpha\ (P \parallel Q')$
by(*metis rPar2*)
with $\langle bn\ \alpha \sharp* \Psi \rangle$ $\langle bn\ \alpha \sharp* P \rangle$ $\langle bn\ \alpha \sharp* Q \rangle$ $\langle bn\ \alpha \sharp* subject\ \alpha \rangle$ $\langle bn\ \alpha \sharp* C \rangle$ $\langle bn\ \alpha \sharp* (bn\ \alpha') \rangle$ S $\langle distinctPerm\ p \rangle$ $\langle bn(p \cdot \alpha) \sharp* \alpha \rangle$ $\langle bn(p \cdot \alpha) \sharp* (P \parallel Q') \rangle$
have $Prop\ C\ \Psi\ (P \parallel Q)\ (p \cdot \alpha)\ (p \cdot (P \parallel Q'))$
by $-$ (*rule rAlpha*, *auto*)
with αEq $Q'eq$ $\langle distinctPerm\ p \rangle$ **show** *?case* **by** *simp*
next
case(*cComm1* Ψ Ψ_Q P M N P' A_P Ψ_P Q K *xvec* Q' A_Q α C P'')
then have $Prop\ C\ \Psi\ (P \parallel Q)\ (\tau)\ ((\!|\nu * xvec|\!)(P' \parallel Q'))$
by(*fastforce intro*: *rComm1*)
then show *?case* **using** $\langle \tau \prec (\!|\nu * xvec|\!)(P' \parallel Q') = \alpha \prec P'' \rangle$
by(*simp add*: *residualInject*)
next
case(*cComm2* Ψ Ψ_Q P M *xvec* N P' A_P Ψ_P Q K Q' A_Q α C P'')
then have $Prop\ C\ \Psi\ (P \parallel Q)\ (\tau)\ ((\!|\nu * xvec|\!)(P' \parallel Q'))$
by(*fastforce intro*: *rComm2*)
then show *?case* **using** $\langle \tau \prec (\!|\nu * xvec|\!)(P' \parallel Q') = \alpha \prec P'' \rangle$
by(*simp add*: *residualInject*)
next
case (*cBrMerge* Ψ Ψ_Q P M N P' A_P Ψ_P Q Q' A_Q α C P'')
then show *?case* **by**(*simp add*: *rBrMerge residualInject*)
next
case(*cBrComm1* Ψ Ψ_Q P M N P' A_P Ψ_P Q *xvec* Q' A_Q α C P'')
from *cBrComm1* $\langle A_P \sharp* M \rangle$ $\langle A_Q \sharp* M \rangle$ $\langle xvec \sharp* M \rangle$ **have** $Prop\ C\ \Psi\ (P \parallel Q)\ (¡M(\!|\nu * xvec|\!)\langle N \rangle)\ (P' \parallel Q')$
by(*fastforce intro*: *rBrComm1*)

note $\langle ¡M(\!|\nu * xvec|\!)\langle N \rangle \prec P' \parallel Q' = \alpha \prec P'' \rangle$
moreover from $\langle xvec \sharp* \alpha \rangle$ **have** $bn\ (¡M(\!|\nu * xvec|\!)\langle N \rangle) \sharp* (bn\ \alpha)$ **by** *simp*
moreover from $\langle distinct\ xvec \rangle$ **have** $distinct\ (bn\ (¡M(\!|\nu * xvec|\!)\langle N \rangle))$ **by** *simp*
moreover note $\langle distinct\ (bn\ \alpha) \rangle$
moreover from $\langle xvec \sharp* M \rangle$ **have** $bn\ (¡M(\!|\nu * xvec|\!)\langle N \rangle) \sharp* (¡M(\!|\nu * xvec|\!)\langle N \rangle \prec P' \parallel Q')$ **by** *simp*
moreover from $\langle bn\ \alpha \sharp* subject\ \alpha \rangle$ **have** $bn\ \alpha \sharp* (\alpha \prec P'')$ **by** *simp*

ultimately obtain p **where** S: $(set\ p) \subseteq (set(bn\ (¡M(\!|\nu{*}xvec|\!)\langle N\rangle)))) \times (set(bn(p \cdot (¡M(\!|\nu{*}xvec|\!)\langle N\rangle))))$ **and** *distinctPerm p*
and αEq: $\alpha = p \cdot (¡M(\!|\nu{*}xvec|\!)\langle N\rangle)$ **and** $P'eq$: $P'' = p \cdot (P' \parallel Q')$ **and** $(bn(p \cdot (¡M(\!|\nu{*}xvec|\!)\langle N\rangle))) \sharp* (¡M(\!|\nu{*}xvec|\!)\langle N\rangle)$
and $(bn(p \cdot (¡M(\!|\nu{*}xvec|\!)\langle N\rangle))) \sharp* (P' \parallel Q')$
by(*rule residualEq*)

from ‹*xvec* $\sharp*$ Ψ› **have** *bn* $(¡M(\!|\nu{*}xvec|\!)\langle N\rangle) \sharp* \Psi$ **by** *simp*
moreover from ‹*xvec* $\sharp*$ P› ‹*xvec* $\sharp*$ Q› **have** *bn* $(¡M(\!|\nu{*}xvec|\!)\langle N\rangle) \sharp* (P \parallel Q)$ **by** *simp*
moreover from ‹*xvec* $\sharp*$ M› **have** *bn* $(¡M(\!|\nu{*}xvec|\!)\langle N\rangle) \sharp*$ *subject* $(¡M(\!|\nu{*}xvec|\!)\langle N\rangle)$ **by** *simp*
moreover from ‹*xvec* $\sharp*$ C› **have** *bn* $(¡M(\!|\nu{*}xvec|\!)\langle N\rangle) \sharp* C$ **by** *simp*
moreover from ‹$(bn(p \cdot (¡M(\!|\nu{*}xvec|\!)\langle N\rangle))) \sharp* (¡M(\!|\nu{*}xvec|\!)\langle N\rangle)$› **have** *bn* $(¡M(\!|\nu{*}xvec|\!)\langle N\rangle) \sharp*$ *bn* $(p \cdot (¡M(\!|\nu{*}xvec|\!)\langle N\rangle))$ **by** *simp*
moreover note ‹$(set\ p) \subseteq (set(bn\ (¡M(\!|\nu{*}xvec|\!)\langle N\rangle)))) \times (set(bn(p \cdot (¡M(\!|\nu{*}xvec|\!)\langle N\rangle))))$›
moreover note ‹*distinctPerm p*›
moreover note ‹$(bn(p \cdot (¡M(\!|\nu{*}xvec|\!)\langle N\rangle))) \sharp* (¡M(\!|\nu{*}xvec|\!)\langle N\rangle)$›
moreover note ‹$(bn(p \cdot (¡M(\!|\nu{*}xvec|\!)\langle N\rangle))) \sharp* (P' \parallel Q')$›
moreover note ‹*Prop* $C\ \Psi\ (P \parallel Q)\ (¡M(\!|\nu{*}xvec|\!)\langle N\rangle)\ (P' \parallel Q')$›

ultimately have *propEqvt*: *Prop* $C\ \Psi\ (P \parallel Q)\ (p \cdot (¡M(\!|\nu{*}xvec|\!)\langle N\rangle))\ (p \cdot (P' \parallel Q'))$ **by**(*rule rAlpha*)

then show *?case* **by** (*simp add*: $\alpha Eq\ P'eq$ *propEqvt*)
next
case(*cBrComm2* $\Psi\ \Psi_Q\ P\ M\ xvec\ N\ P'\ A_P\ \Psi_P\ Q\ Q'\ A_Q\ \alpha\ C\ P''$)
from *cBrComm2* ‹$A_P \sharp* M$› ‹$A_Q \sharp* M$› ‹*xvec* $\sharp*$ M› **have** *Prop* $C\ \Psi\ (P \parallel Q)\ (¡M(\!|\nu{*}xvec|\!)\langle N\rangle)\ (P' \parallel Q')$
by(*fastforce intro*: *rBrComm2*)

note ‹$¡M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P' \parallel Q' = \alpha \prec P''$›
moreover from ‹*xvec* $\sharp*$ α› **have** *bn* $(¡M(\!|\nu{*}xvec|\!)\langle N\rangle) \sharp* (bn\ \alpha)$ **by** *simp*
moreover from ‹*distinct xvec*› **have** *distinct* $(bn\ (¡M(\!|\nu{*}xvec|\!)\langle N\rangle))$ **by** *simp*
moreover note ‹*distinct* $(bn\ \alpha)$›
moreover from ‹*xvec* $\sharp*$ M› **have** *bn* $(¡M(\!|\nu{*}xvec|\!)\langle N\rangle) \sharp* (¡M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P' \parallel Q')$ **by** *simp*
moreover from ‹*bn* $\alpha \sharp*$ *subject* α› **have** *bn* $\alpha \sharp* (\alpha \prec P'')$ **by** *simp*
ultimately obtain p **where** S: $(set\ p) \subseteq (set(bn\ (¡M(\!|\nu{*}xvec|\!)\langle N\rangle)))) \times (set(bn(p \cdot (¡M(\!|\nu{*}xvec|\!)\langle N\rangle))))$ **and** *distinctPerm p*
and αEq: $\alpha = p \cdot (¡M(\!|\nu{*}xvec|\!)\langle N\rangle)$ **and** $P'eq$: $P'' = p \cdot (P' \parallel Q')$ **and** $(bn(p \cdot (¡M(\!|\nu{*}xvec|\!)\langle N\rangle))) \sharp* (¡M(\!|\nu{*}xvec|\!)\langle N\rangle)$
and $(bn(p \cdot (¡M(\!|\nu{*}xvec|\!)\langle N\rangle))) \sharp* (P' \parallel Q')$
by(*rule residualEq*)

from ‹*xvec* $\sharp*$ Ψ› **have** *bn* $(¡M(\!|\nu{*}xvec|\!)\langle N\rangle) \sharp* \Psi$ **by** *simp*
moreover from ‹*xvec* $\sharp*$ P› ‹*xvec* $\sharp*$ Q› **have** *bn* $(¡M(\!|\nu{*}xvec|\!)\langle N\rangle) \sharp* (P \parallel Q)$ **by** *simp*
moreover from ‹*xvec* $\sharp*$ M› **have** *bn* $(¡M(\!|\nu{*}xvec|\!)\langle N\rangle) \sharp*$ *subject* $(¡M(\!|\nu{*}xvec|\!)\langle N\rangle)$

by *simp*
moreover from ‹*xvec* ♯* *C*› **have** *bn* (¡*M*(⦇*ν*∗*xvec*⦈)⟨*N*⟩) ♯* *C* **by** *simp*
moreover from ‹(*bn*(*p* ∙ (¡*M*(⦇*ν*∗*xvec*⦈)⟨*N*⟩))) ♯* (¡*M*(⦇*ν*∗*xvec*⦈)⟨*N*⟩)› **have** *bn* (¡*M*(⦇*ν*∗*xvec*⦈)⟨*N*⟩) ♯* *bn* (*p* ∙ (¡*M*(⦇*ν*∗*xvec*⦈)⟨*N*⟩)) **by** *simp*
moreover note ‹(*set p*) ⊆ (*set*(*bn* (¡*M*(⦇*ν*∗*xvec*⦈)⟨*N*⟩))) × (*set*(*bn*(*p* ∙ (¡*M*(⦇*ν*∗*xvec*⦈)⟨*N*⟩))))›
moreover note ‹*distinctPerm p*›
moreover note ‹(*bn*(*p* ∙ (¡*M*(⦇*ν*∗*xvec*⦈)⟨*N*⟩))) ♯* (¡*M*(⦇*ν*∗*xvec*⦈)⟨*N*⟩)›
moreover note ‹(*bn*(*p* ∙ (¡*M*(⦇*ν*∗*xvec*⦈)⟨*N*⟩))) ♯* (*P*′ ∥ *Q*′)›
moreover note ‹*Prop C* Ψ (*P* ∥ *Q*) (¡*M*(⦇*ν*∗*xvec*⦈)⟨*N*⟩) (*P*′ ∥ *Q*′)›

ultimately have *propEqvt*: *Prop C* Ψ (*P* ∥ *Q*) (*p* ∙ (¡*M*(⦇*ν*∗*xvec*⦈)⟨*N*⟩)) (*p* ∙ (*P*′ ∥ *Q*′)) **by**(*rule rAlpha*)

then show *?case* **by** (*simp add*: *αEq P*′*eq propEqvt*)
next
case (*cBrClose* Ψ *P M xvec N P*′ *x* α *C P*′′)
then have *Prop C* Ψ (⦇*νx*⦈*P*) (τ) (⦇*νx*⦈(⦇*ν*∗*xvec*⦈*P*′))
by(*fastforce intro*: *rBrClose*)
then show *?case* **using** ‹τ ≺ ⦇*νx*⦈(⦇*ν*∗*xvec*⦈*P*′) = α ≺ *P*′′›
by(*simp add*: *residualInject*)
next
case(*cOpen* Ψ *P M xvec yvec N P*′ *x* α *C P*′′)
note ‹*M*⦇*ν*∗(*xvec*@*x*#*yvec*)⦈⟨*N*⟩ ≺ *P*′ = α ≺ *P*′′›
moreover from ‹*xvec* ♯* α› ‹*x* ♯ α› ‹*yvec* ♯* α› **have** (*xvec*@*x*#*yvec*) ♯* (*bn* α)
by *auto*
moreover from ‹*xvec* ♯* *yvec*› ‹*x* ♯ *xvec*› ‹*x* ♯ *yvec*› ‹*distinct xvec*› ‹*distinct yvec*›
have *distinct*(*xvec*@*x*#*yvec*)
by(*auto simp add*: *fresh-star-def*) (*simp add*: *fresh-def name-list-supp*)
moreover note ‹*distinct*(*bn* α)›
moreover from ‹*xvec* ♯* *M*› ‹*x* ♯ *M*› ‹*yvec* ♯* *M*› **have** (*xvec*@*x*#*yvec*) ♯* *M* **by** *auto*
then have (*xvec*@*x*#*yvec*) ♯* (*M*⦇*ν*∗(*xvec*@*x*#*yvec*)⦈⟨*N*⟩ ≺ *P*′) **by** *auto*
moreover from ‹*bn* α ♯* *subject* α› **have** *bn* α ♯* (α ≺ *P*′′) **by** *simp*
ultimately obtain *p* **where** *S*: (*set p*) ⊆ (*set*(*xvec*@*x*#*yvec*)) × (*set*(*p* ∙ (*xvec*@*x*#*yvec*))) **and** *distinctPerm p*
and *αeq*: α = (*p* ∙ *M*)⦇*ν*∗(*p* ∙ (*xvec*@*x*#*yvec*))⦈⟨(*p* ∙ *N*)⟩ **and** *P*′*eq*: *P*′′ = (*p* ∙ *P*′)
and *A*: (*xvec*@*x*#*yvec*) ♯* ((*p* ∙ *M*)⦇*ν*∗(*p* ∙ (*xvec*@*x*#*yvec*))⦈⟨(*p* ∙ *N*)⟩)
and *B*: (*p* ∙ (*xvec*@*x*#*yvec*)) ♯* (*M*⦇*ν*∗(*xvec*@*x*#*yvec*)⦈⟨*N*⟩)
and *C*: (*p* ∙ (*xvec*@*x*#*yvec*)) ♯* *P*′
by − (*rule residualEq*, (*assumption* | *simp*)+)
note ‹Ψ ▷ *P* ⟼*M*⦇*ν*∗(*xvec*@*yvec*)⦈⟨*N*⟩ ≺ *P*′› ‹*x* ∈ (*supp N*)›

moreover {
fix *C*
from ‹*xvec* ♯* *M*› ‹*yvec* ♯* *M*› **have** (*xvec*@*yvec*) ♯* *M* **by** *simp*
moreover from ‹*distinct xvec*› ‹*distinct yvec*› ‹*xvec* ♯* *yvec*› **have** *distinct*(*xvec*@*yvec*)
by *auto* (*simp add*: *fresh-star-def name-list-supp fresh-def*)
ultimately have *Prop C* Ψ *P* (*M*⦇*ν*∗(*xvec*@*yvec*)⦈⟨*N*⟩) *P*′ **by**(*fastforce intro*:

cOpen)
}

moreover note ‹$x \sharp \Psi$› ‹$x \sharp M$› ‹$x \sharp xvec$› ‹$x \sharp yvec$› ‹$xvec \sharp* \Psi$› ‹$xvec \sharp* P$› ‹$xvec \sharp* M$›
‹$yvec \sharp* \Psi$› ‹$yvec \sharp* P$› ‹$yvec \sharp* M$› ‹$yvec \sharp* C$› ‹$x \sharp C$› ‹$xvec \sharp* C$› ‹*distinct xvec*› ‹*distinct yvec*›
ultimately have *Prop C* Ψ $((\!|\nu x|\!)P)$ $(M(\!|\nu*(xvec@x\#yvec)|\!)\langle N\rangle)$ P'
by(*metis rOpen*)

with ‹$xvec \sharp* \Psi$› ‹$yvec \sharp* \Psi$› ‹$xvec \sharp* P$› ‹$yvec \sharp* P$› ‹$xvec \sharp* M$› ‹$yvec \sharp* M$›
‹$yvec \sharp* C$› *S* ‹*distinctPerm p*› ‹$x \sharp C$› ‹$xvec \sharp* C$›
‹$x \sharp \Psi$› ‹$x \sharp M$› ‹$x \sharp xvec$› ‹$x \sharp yvec$› *A B C*
have *Prop C* Ψ $((\!|\nu x|\!)P)$ $(p \cdot (M(\!|\nu*(xvec@x\#yvec)|\!)\langle N\rangle))$ $(p \cdot P')$
apply −
apply(*rule rAlpha*[**where** α=$M(\!|\nu*(xvec@x\#yvec)|\!)\langle N\rangle$];*clarsimp*)
by (*meson abs-fresh*(*1*) *abs-fresh-list-star' freshChainAppend freshSets*(*5*) *psiFreshVec*(*5*))
with αeq $P'eq$ **show** *?case* **by** *simp*
next
case(*cBrOpen* Ψ *P M xvec yvec N P' x* α *C P''*)
note ‹$¡M(\!|\nu*(xvec@x\#yvec)|\!)\langle N\rangle \prec P' = \alpha \prec P''$›
moreover from ‹$xvec \sharp* \alpha$› ‹$x \sharp \alpha$› ‹$yvec \sharp* \alpha$› **have** $(xvec@x\#yvec) \sharp* (bn\ \alpha)$
by *auto*
moreover from ‹$xvec \sharp* yvec$› ‹$x \sharp xvec$› ‹$x \sharp yvec$› ‹*distinct xvec*› ‹*distinct yvec*›
have *distinct*$(xvec@x\#yvec)$
by(*clarsimp simp add*: *fresh-star-def*; *safe*; *simp add*: *fresh-def name-list-supp*)
moreover note ‹*distinct*$(bn\ \alpha)$›
moreover from ‹$xvec \sharp* M$› ‹$x \sharp M$› ‹$yvec \sharp* M$› **have** $(xvec@x\#yvec) \sharp* M$
by *auto*
then have $(xvec@x\#yvec) \sharp* (¡M(\!|\nu*(xvec@x\#yvec)|\!)\langle N\rangle \prec P')$ **by** *auto*
moreover from ‹$bn\ \alpha \sharp* subject\ \alpha$› **have** $bn\ \alpha \sharp* (\alpha \prec P'')$ **by** *simp*
ultimately obtain *p* **where** *S*: $(set\ p) \subseteq (set(xvec@x\#yvec)) \times (set(p \cdot (xvec@x\#yvec)))$
and *distinctPerm p*
and αeq: $\alpha = ¡(p \cdot M)(\!|\nu*(p \cdot (xvec@x\#yvec))|\!)\langle(p \cdot N)\rangle$ **and** $P'eq$: $P'' = (p \cdot P')$
and *A*: $(xvec@x\#yvec) \sharp* (¡(p \cdot M)(\!|\nu*(p \cdot (xvec@x\#yvec))|\!)\langle(p \cdot N)\rangle)$
and *B*: $(p \cdot (xvec@x\#yvec)) \sharp* (¡M(\!|\nu*(xvec@x\#yvec)|\!)\langle N\rangle)$
and *C*: $(p \cdot (xvec@x\#yvec)) \sharp* P'$
apply −
by(*rule residualEq*) (*assumption*|*simp*)+
note ‹$\Psi \rhd P \longmapsto ¡M(\!|\nu*(xvec@yvec)|\!)\langle N\rangle \prec P'$› ‹$x \in (supp\ N)$›

moreover {
fix *C*
from ‹$xvec \sharp* M$› ‹$yvec \sharp* M$› **have** $(xvec@yvec) \sharp* M$ **by** *simp*
moreover from ‹*distinct xvec*› ‹*distinct yvec*› ‹$xvec \sharp* yvec$› **have** *distinct*$(xvec@yvec)$
by *auto* (*simp add*: *fresh-star-def name-list-supp fresh-def*)
ultimately have *Prop C* Ψ *P* $(¡M(\!|\nu*(xvec@yvec)|\!)\langle N\rangle)$ P' **by**(*fastforce intro*: *cBrOpen*)

}

moreover note ‹$x \sharp \Psi$› ‹$x \sharp M$› ‹$x \sharp xvec$› ‹$x \sharp yvec$› ‹$xvec \sharp* \Psi$› ‹$xvec \sharp* P$› ‹$xvec \sharp* M$›
‹$yvec \sharp* \Psi$› ‹$yvec \sharp* P$› ‹$yvec \sharp* M$› ‹$yvec \sharp* C$› ‹$x \sharp C$› ‹$xvec \sharp* C$› ‹*distinct* $xvec$› ‹*distinct* $yvec$›
ultimately have *Prop* $C\ \Psi\ (\langle\!|\nu x|\!\rangle P)\ (¡M\langle\!|\nu*(xvec@x\#yvec)|\!\rangle\langle N\rangle)\ P'$
by(*metis rBrOpen*)

with ‹$xvec \sharp* \Psi$› ‹$yvec \sharp* \Psi$› ‹$xvec \sharp* P$› ‹$yvec \sharp* P$› ‹$xvec \sharp* M$› ‹$yvec \sharp* M$›
‹$yvec \sharp* C$› S ‹*distinctPerm* p› ‹$x \sharp C$› ‹$xvec \sharp* C$›
‹$x \sharp \Psi$› ‹$x \sharp M$› ‹$x \sharp xvec$› ‹$x \sharp yvec$› $A\ B\ C$
have *Prop* $C\ \Psi\ (\langle\!|\nu x|\!\rangle P)\ (p \cdot (¡M\langle\!|\nu*(xvec@x\#yvec)|\!\rangle\langle N\rangle))\ (p \cdot P')$
apply −
apply(*rule rAlpha*[**where** α=$¡M\langle\!|\nu*(xvec@x\#yvec)|\!\rangle\langle N\rangle$]; *clarsimp*)
by (*meson abs-fresh*(1) *abs-fresh-list-star*′ *freshChainAppend freshSets*(5) *psiFreshVec*(5))
with $\alpha eq\ P'eq$ **show** *?case* **by** *simp*
next
case(*cScope* $\Psi\ P\ \alpha\ P'\ x\ \alpha'\ C\ P''$)
note ‹$\alpha \prec (\langle\!|\nu x|\!\rangle P') = \alpha' \prec P''$›
moreover from ‹$bn\ \alpha \sharp* \alpha'$› **have** $bn\ \alpha \sharp* (bn\ \alpha')$ **by** *auto*
moreover note ‹*distinct* $(bn\ \alpha)$› ‹*distinct*$(bn\ \alpha')$›
moreover from ‹$bn\ \alpha \sharp*$ *subject* α› ‹$bn\ \alpha' \sharp*$ *subject* α'›
have $bn\ \alpha \sharp* (\alpha \prec (\langle\!|\nu x|\!\rangle P')$ **and** $bn\ \alpha' \sharp* (\alpha' \prec P'')$ **by** *simp*+
ultimately obtain p **where** S: $(set\ p) \subseteq (set(bn\ \alpha)) \times (set(bn(p \cdot \alpha)))$ **and** *distinctPerm p*
and αEq: $\alpha' = p \cdot \alpha$ **and** $P'eq$: $P'' = p \cdot (\langle\!|\nu x|\!\rangle P')$ **and** $(bn(p \cdot \alpha)) \sharp* \alpha$
and $(bn(p \cdot \alpha)) \sharp* (\langle\!|\nu x|\!\rangle P')$
by(*rule residualEq*)

note ‹$\Psi \rhd P \longmapsto \alpha \prec P'$›
moreover from ‹$bn\ \alpha \sharp*$ *subject* α› ‹*distinct*$(bn\ \alpha)$›
have $\bigwedge C.$ *Prop* $C\ \Psi\ P\ \alpha\ P'$ **by**(*fastforce intro*: *cScope*)

moreover note ‹$x \sharp \Psi$› ‹$x \sharp \alpha$› ‹$bn\ \alpha \sharp* \Psi$› ‹$bn\ \alpha \sharp* P$› ‹$bn\ \alpha \sharp*$ *subject* α›
‹$x \sharp C$› ‹$bn\ \alpha \sharp* C$› ‹*distinct*$(bn\ \alpha)$›
ultimately have *Prop* $C\ \Psi\ (\langle\!|\nu x|\!\rangle P)\ \alpha\ (\langle\!|\nu x|\!\rangle P')$
by(*rule rScope*)
with ‹$bn\ \alpha \sharp* \Psi$› ‹$bn\ \alpha \sharp* P$› ‹$x \sharp \alpha$› ‹$bn\ \alpha \sharp*$ *subject* α› ‹$bn\ \alpha \sharp* C$› ‹$bn\ \alpha \sharp* (bn\ \alpha')$› S ‹*distinctPerm* p› ‹$bn(p \cdot \alpha) \sharp* \alpha$› ‹$bn(p \cdot \alpha) \sharp* (\langle\!|\nu x|\!\rangle P')$›
have *Prop* $C\ \Psi\ (\langle\!|\nu x|\!\rangle P)\ (p \cdot \alpha)\ (p \cdot (\langle\!|\nu x|\!\rangle P'))$
by(*fastforce intro*: *rAlpha*)
with $\alpha Eq\ P'eq$ ‹*distinctPerm* p› **show** *?case* **by** *simp*
next
case(*Bang* $\Psi\ P\ \alpha\ C\ P'$)
then show *?case* **by**(*fastforce intro*: *rBang*)
qed

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* :: $'f$::*fs-name* $\Rightarrow$ $'b \Rightarrow ('a, 'b, 'c)$ *psi* $\Rightarrow$
$'a \Rightarrow ('a, 'b, 'c)$ *boundOutput* $\Rightarrow$ *bool*
and C :: $'f$::*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*: $\bigwedge \Psi\ P\ M\ B\ \varphi\ Cs\ C.$
$[\![\Psi \rhd P \longmapsto (ROut\ M\ B); \bigwedge C.\ Prop\ C\ \Psi\ P\ M\ B; (\varphi, P) \in set\ Cs; \Psi \vdash \varphi; guarded\ P]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (Cases\ Cs)\ M\ B$
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* Ψ *P M B*
using ‹Ψ ▷ *P* ⟼(*ROut M B*)›
proof(*nominal-induct* Ψ *P Rs*==(*ROut M B*) *avoiding*: *C arbitrary*: *B rule*: *semantics.strong-induct*)
case(*cInput* Ψ *M K xvec N Tvec P C*)
then show *?case* **by**(*simp add*: *residualInject*)
next
case *cBrInput*
then show *?case* **by**(*simp add*: *residualInject*)
next
case(*Output* Ψ *M K N P C*)
then show *?case* **by**(*force simp add*: *residualInject intro*: *rOutput*)
next
case(*BrOutput* Ψ *M N P C*)
then show *?case* **by**(*simp add*: *residualInject*)
next
case(*Case* Ψ *P* φ *Cs C B*)
then show *?case* **by**(*force intro*: *rCase*)
next
case(*cPar1* Ψ Ψ_Q *P* α *P′ Q* A_Q *C*)
then show *?case* **by**(*force intro*: *rPar1 simp add*: *residualInject*)
next
case(*cPar2* Ψ Ψ_P *Q* α *Q′ P* A_P *C*)
then show *?case* **by**(*force intro*: *rPar2 simp add*: *residualInject*)
next
case *cComm1*
then show *?case* **by**(*simp add*: *residualInject*)
next
case *cComm2*
then show *?case* **by**(*simp add*: *residualInject*)
next
case *cBrMerge*
then show *?case* **by**(*simp add*: *residualInject*)
next
case (*cBrComm1* Ψ Ψ_Q *P M N P′* A_P Ψ_P *Q xvec Q′* A_Q *C B*)
then show *?case* **by**(*simp add*: *residualInject*)
next
case (*cBrComm2* Ψ Ψ_Q *P M xvec N P′* A_P Ψ_P *Q Q′* A_Q *C B*)
then show *?case* **by**(*simp add*: *residualInject*)
next
case(*cBrClose* Ψ *P M xvec N P′ C B*)
then show *?case* **by**(*simp add*: *residualInject*)
next
case(*cOpen* Ψ *P M xvec yvec N P′ x C B*)
then show *?case* **by**(*force intro*: *rOpen simp add*: *residualInject*)
next
case *cBrOpen*
then show *?case* **by**(*simp add*: *residualInject*)
next

case(*cScope* Ψ P M α P' x C)
then show *?case* **by**(*force intro*: *rScope simp add*: *residualInject*)
next
case(*Bang* Ψ P C B)
then show *?case* **by**(*force intro*: *rBang*)
qed

lemma *brOutputInduct*[*consumes 1*, *case-names cBrOutput cCase cPar1 cPar2 cBrComm1 cBrComm2 cBrOpen cScope cBang*]:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and M :: $'a$
and B :: $('a, 'b, 'c)$ *boundOutput*
and *Prop* :: $'f$::*fs-name* $\Rightarrow$ $'b \Rightarrow ('a, 'b, 'c)$ *psi* $\Rightarrow$
$'a \Rightarrow ('a, 'b, 'c)$ *boundOutput* $\Rightarrow$ *bool*
and C :: $'f$::*fs-name*

assumes $\Psi \rhd P \longmapsto RBrOut\ M\ B$
and *rBrOutput*: $\bigwedge \Psi\ M\ K\ N\ P\ C.\ [\![\Psi \vdash M \preceq K]\!] \Longrightarrow Prop\ C\ \Psi\ (M\langle N\rangle.P)\ K\ (N \prec' P)$
and *rCase*: $\bigwedge \Psi\ P\ M\ B\ \varphi\ Cs\ C.$
$[\![\Psi \rhd P \longmapsto (RBrOut\ M\ B);\ \bigwedge C.\ Prop\ C\ \Psi\ P\ M\ B;\ (\varphi, P) \in set\ Cs;\ \Psi \vdash \varphi;\ guarded\ P]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (Cases\ Cs)\ M\ B$
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 *rBrComm1*: $\bigwedge \Psi\ \Psi_Q\ P\ M\ N\ P'\ A_P\ \Psi_P\ Q\ xvec\ Q'\ A_Q\ C.$
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P';$
$extractFrame\ P = \langle A_P, \Psi_P\rangle;\ distinct\ A_P;$
$\Psi \otimes \Psi_P \rhd Q \longmapsto ¡M(\!|\nu{*}xvec|\!)\langle N\rangle \prec Q';\ \bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_P)\ Q\ M\ ((\!|\nu{*}xvec|\!)N \prec' Q');$
$extractFrame\ Q = \langle A_Q, \Psi_Q\rangle;\ distinct\ A_Q;$
$A_P \sharp* \Psi;\ A_P \sharp* \Psi_Q;\ A_P \sharp* P;\ A_P \sharp* N;\ A_P \sharp* P';$
$A_P \sharp* Q;\ A_P \sharp* Q';\ A_P \sharp* A_Q;\ A_P \sharp* xvec;\ A_Q \sharp* \Psi;\ A_Q \sharp* \Psi_P;$
$A_Q \sharp* P;\ A_Q \sharp* N;\ A_Q \sharp* P';\ A_Q \sharp* Q;\ A_Q \sharp* Q';\ distinct\ xvec;$

$A_P \sharp* M$; $A_Q \sharp* M$; $xvec \sharp* M$;
$A_Q \sharp* xvec$; $xvec \sharp* \Psi$; $xvec \sharp* \Psi_P$; $xvec \sharp* \Psi_Q$; $xvec \sharp* P$;
$xvec \sharp* Q$; $A_P \sharp* C$; $A_Q \sharp* C$; $xvec \sharp* C\rrbracket \Longrightarrow$
Prop C Ψ $(P \parallel Q)$ M $((\!|\nu* xvec|\!)N \prec' (P' \parallel Q'))$

and *rBrComm2*: $\bigwedge \Psi\ \Psi_Q\ P\ M\ xvec\ N\ P'\ A_P\ \Psi_P\ Q\ Q'\ A_Q\ C.$
$\llbracket \Psi \otimes \Psi_Q \rhd P \longmapsto ¡M(\!|\nu* xvec|\!)\langle N\rangle \prec P'$; $\bigwedge C.$ *Prop* C $(\Psi \otimes \Psi_Q)$ P M $((\!|\nu* xvec|\!)N \prec' P')$;
extractFrame $P = \langle A_P, \Psi_P\rangle$; *distinct* A_P;
$\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$;
extractFrame $Q = \langle A_Q, \Psi_Q\rangle$; *distinct* A_Q;
$A_P \sharp* \Psi$; $A_P \sharp* \Psi_Q$; $A_P \sharp* P$; $A_P \sharp* N$; $A_P \sharp* P'$;
$A_P \sharp* Q$; $A_P \sharp* Q'$; $A_P \sharp* A_Q$; $A_P \sharp* xvec$; $A_Q \sharp* \Psi$; $A_Q \sharp* \Psi_P$;
$A_Q \sharp* P$; $A_Q \sharp* N$; $A_Q \sharp* P'$; $A_Q \sharp* Q$; $A_Q \sharp* Q'$; *distinct* $xvec$;
$A_P \sharp* M$; $A_Q \sharp* M$; $xvec \sharp* M$;
$A_Q \sharp* xvec$; $xvec \sharp* \Psi$; $xvec \sharp* \Psi_P$; $xvec \sharp* \Psi_Q$; $xvec \sharp* P$;
$xvec \sharp* Q$; $A_P \sharp* C$; $A_Q \sharp* C$; $xvec \sharp* C\rrbracket \Longrightarrow$
Prop C Ψ $(P \parallel Q)$ M $((\!|\nu* xvec|\!)N \prec' (P' \parallel Q'))$

and *rBrOpen*: $\bigwedge \Psi\ P\ M\ xvec\ yvec\ N\ P'\ x\ C.$
$\llbracket \Psi \rhd P \longmapsto ¡M(\!|\nu*(xvec@yvec)|\!)\langle N\rangle \prec P'$; $x \in supp\ N$; $\bigwedge C.$ *Prop* C Ψ P M $((\!|\nu*(xvec@yvec)|\!)N \prec' P')$;
$x \sharp \Psi$; $x \sharp M$; $x \sharp xvec$; $x \sharp yvec$; $xvec \sharp* \Psi$; $xvec \sharp* P$; $xvec \sharp* M$;
$xvec \sharp* yvec$; $yvec \sharp* \Psi$; $yvec \sharp* P$; $yvec \sharp* M$; $yvec \sharp* C$; $x \sharp C$;
$xvec \sharp* C\rrbracket \Longrightarrow$
Prop C Ψ $((\!|\nu x|\!)P)$ M $((\!|\nu*(xvec@x\#yvec)|\!)N \prec' P')$

and *rScope*: $\bigwedge \Psi\ P\ M\ xvec\ N\ P'\ x\ C.$
$\llbracket \Psi \rhd P \longmapsto ¡M(\!|\nu* xvec|\!)\langle N\rangle \prec P'$; $\bigwedge C.$ *Prop* C Ψ P M $((\!|\nu* xvec|\!)N \prec' P')$;
$x \sharp \Psi$; $x \sharp M$; $x \sharp xvec$; $x \sharp N$; $xvec \sharp* \Psi$; $xvec \sharp* P$; $xvec \sharp* M$;
$x \sharp C$; $xvec \sharp* C\rrbracket \Longrightarrow$
Prop C Ψ $((\!|\nu x|\!)P)$ M $((\!|\nu* xvec|\!)N \prec' (\!|\nu x|\!)P')$

and *rBang*: $\bigwedge \Psi\ P\ M\ B\ C.$
$\llbracket \Psi \rhd P \parallel !P \longmapsto (RBrOut\ M\ B)$; *guarded* P; $\bigwedge C.$ *Prop* C Ψ $(P \parallel !P)$ M $B\rrbracket \Longrightarrow$
Prop C Ψ $(!P)$ M B

shows *Prop* C Ψ P M B

using ‹$\Psi \rhd P \longmapsto (RBrOut\ M\ B)$›

proof(*nominal-induct* Ψ P *Rs*==$(RBrOut\ M\ B)$ *avoiding*: C *arbitrary*: B *rule*: *semantics.strong-induct*)

case(*cInput* Ψ M K $xvec$ N $Tvec$ P C)

then show *?case* **by**(*simp add*: *residualInject*)

next

case *cBrInput*

then show *?case* **by**(*simp add*: *residualInject*)

next

case(*Output* Ψ M K N P C)

then show *?case* **by**(*simp add*: *residualInject*)

next

case(*BrOutput* Ψ M N P C)

then show *?case* **by**(*auto simp add*: *residualInject intro*: *rBrOutput*)

```
next
  case(Case Ψ P φ Cs C B)
  then show ?case by(force intro: rCase)
next
  case(cPar1 Ψ Ψ_Q P α P' Q A_Q C)
  then show ?case by(force intro: rPar1 simp add: residualInject)
next
  case(cPar2 Ψ Ψ_P Q α Q' P A_P C)
  then show ?case by(force intro: rPar2 simp add: residualInject)
next
  case cComm1
  then show ?case by(simp add: residualInject)
next
  case cComm2
  then show ?case by(simp add: residualInject)
next
  case cBrMerge
  then show ?case by(simp add: residualInject)
next
  case (cBrComm1 Ψ Ψ_Q P M N P' A_P Ψ_P Q xvec Q' A_Q C B)
  then show ?case by(force intro: rBrComm1 simp add: residualInject)
next
  case (cBrComm2 Ψ Ψ_Q P M xvec N P' A_P Ψ_P Q Q' A_Q C B)
  then show ?case by(force intro: rBrComm2 simp add: residualInject)
next
  case(cBrClose Ψ P M xvec N P' C B)
  then show ?case by(simp add: residualInject)
next
  case(cOpen Ψ P M xvec yvec N P' x C B)
  then show ?case by(simp add: residualInject)
next
  case(cBrOpen Ψ P M xvec yvec N P' x C B)
  then show ?case by(force intro: rBrOpen simp add: residualInject)
next
  case(cScope Ψ P M α P' x C)
  then show ?case by(force intro: rScope simp add: residualInject)
next
  case(Bang Ψ P C B)
  then show ?case by(force intro: rBang)
qed

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 $\Psi \rhd P \longmapsto\!\!\!\!\rightarrow \alpha \prec P'$
and $bn\ \alpha \sharp* subject\ \alpha$
and $distinct(bn\ \alpha)$
and $y \in set(bn\ \alpha)$

shows $y \in supp(object\ \alpha)$
using *assms*
proof(*nominal-induct avoiding*: P' *arbitrary*: y *rule*: *semanticsInduct*)
case(*cAlpha* $\Psi\ P\ \alpha\ P'\ p\ P''\ y$)
from ‹$y \in set(bn(p \cdot \alpha))$› **have** $(p \cdot y) \in (p \cdot set(bn(p \cdot \alpha)))$
by(*rule pt-set-bij2*[*OF pt-name-inst*, *OF at-name-inst*])
then have $(p \cdot y) \in set(bn\ \alpha)$ **using** ‹*distinctPerm p*›
by(*simp add*: *eqvts*)
then have $(p \cdot y) \in supp(object\ \alpha)$ **by**(*rule cAlpha*)
then have $(p \cdot p \cdot y) \in (p \cdot supp(object\ \alpha))$
by(*rule pt-set-bij2*[*OF pt-name-inst*, *OF at-name-inst*])
then show *?case* **using** ‹*distinctPerm p*›
by(*simp add*: *eqvts*)
next
case *cInput*
then show *?case* **by** *simp*
next
case *cBrInput*
then show *?case* **by** *simp*
next
case *cOutput*
then show *?case* **by** *simp*
next
case *cBrOutput*
then show *?case* **by** *simp*
next
case *cCase*
then show *?case* **by** *simp*
next
case *cPar1*
then show *?case* **by** *simp*
next
case *cPar2*
then show *?case* **by** *simp*
next
case *cComm1*
then show *?case* **by** *simp*
next
case *cComm2*
then show *?case* **by** *simp*
next
case *cBrMerge*
then show *?case* **by** *simp*

next
case *cBrComm1*
then show *?case* **by** *simp*
next
case *cBrComm2*
then show *?case* **by** *simp*
next
case *cBrClose*
then show *?case* **by** *simp*
next
case *cOpen*
then show *?case* **by**(*auto simp add*: *supp-list-cons supp-list-append supp-atm supp-some*)
next
case *cBrOpen*
then show *?case* **by**(*auto simp add*: *supp-list-cons supp-list-append supp-atm supp-some*)
next
case *cScope*
then show *?case* **by** *simp*
next
case *cBang*
then show *?case* **by** *simp*
qed

lemma *alphaBoundOutputChain'*:
fixes *yvec* :: *name list*
and *xvec* :: *name list*
and *B* :: (*'a*, *'b*, *'c*) *boundOutput*

assumes *length xvec* = *length yvec*
and *yvec* ♯* *B*
and *yvec* ♯* *xvec*
and *distinct yvec*

shows (|*ν***xvec*|)*B* = (|*ν***yvec*|)([*xvec yvec*] •$_v$ *B*)
using *assms*
proof(*induct rule*: *composePermInduct*)
case *cBase*
show *?case* **by** *simp*
next
case(*cStep x xvec y yvec*)
then show *?case*
by (*auto simp add*: *alphaBoundOutput*[*of y*] *eqvts*)
qed

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 ♯* N
  and   yvec ♯* P
  and   yvec ♯* xvec
  and   distinct yvec

shows (⦇ν*xvec⦈)(N ≺' P) = (⦇ν*yvec⦈)(([xvec yvec] ∙v N) ≺' ([xvec yvec] ∙v P))
proof −
  from assms have (⦇ν*xvec⦈)(N ≺' P) = (⦇ν*yvec⦈)([xvec yvec] ∙v (N ≺' P))
    by(simp add: alphaBoundOutputChain')
  then show ?thesis by simp
qed

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 α ≺ P = β ≺ Q
  and   distinct(bn α)
  and   ⋀x. x ∈ set(bn α) ⟹ x ∈ supp(object α)
  and   bn α ♯* bn β
  and   bn α ♯* (object β)
  and   bn α ♯* Q

shows distinct(bn β)
  using assms
proof −
  {
    fix xvec M yvec N
    assume Eq: (⦇ν*xvec⦈)N ≺' P = (⦇ν*yvec⦈)M ≺' Q
    assume distinct xvec and xvec ♯* M and xvec ♯* yvec and xvec ♯* Q
    assume Mem: ⋀x. x ∈ set xvec ⟹ x ∈ (supp N)
    have distinct yvec
    proof −
      from Eq have length xvec = length yvec
        by(rule boundOutputChainEqLength)
       with Eq ‹distinct xvec› ‹xvec ♯* yvec› ‹xvec ♯* M› ‹xvec ♯* Q› Mem show
?thesis
      proof(induct n==length xvec arbitrary: xvec yvec M Q rule: nat.induct)
        case(zero xvec yvec M Q)
        then show ?case by simp
      next
```

case(*Suc n xvec yvec M Q*)
have *L*: *length xvec* = *length yvec* **and** *Suc n* = *length xvec* **by** *fact*+
then obtain *x xvec′ y yvec′* **where** *xEq*: *xvec* = *x*#*xvec′* **and** *yEq*: *yvec* = *y*#*yvec′*
and *L′*: *length xvec′* = *length yvec′*
by(*cases xvec*, *auto*, *cases yvec*, *auto*)
have *xvecFreshyvec*: *xvec* ♯* *yvec* **and** *xvecDist*: *distinct xvec* **by** *fact*+
with *xEq yEq* **have** *xineqy*: *x* ≠ *y* **and** *xvec′Freshyvec′*: *xvec′* ♯* *yvec′*
and *xvec′Dist*: *distinct xvec′* **and** *xFreshxvec′*: *x* ♯ *xvec′*
and *xFreshyvec′*: *x* ♯ *yvec′* **and** *yFreshxvec′*: *y* ♯ *xvec′*
by *auto*
have *Eq*: (|*ν***xvec*|)*N* ≺′ *P* = (|*ν***yvec*|)*M* ≺′ *Q* **by** *fact*
with *xEq yEq xineqy* **have** *Eq′*: (|*ν***xvec′*|)*N* ≺′ *P* = (|*ν**([(*x*, *y*)] ∙ *yvec′*)|)([(*x*, *y*)] ∙ *M*) ≺′ ([(*x*, *y*)] ∙ *Q*)
by(*simp add*: *boundOutput.inject alpha eqvts*)
moreover have *Mem*:⋀*x*. *x* ∈ *set xvec* ⟹ *x* ∈ *supp N* **by** *fact*
with *xEq* **have** ⋀*x*. *x* ∈ *set xvec′* ⟹ *x* ∈ *supp N* **by** *simp*
moreover have *xvec* ♯* *M* **by** *fact*
with *xEq xFreshxvec′ yFreshxvec′* **have** *xvec′* ♯* ([(*x*, *y*)] ∙ *M*) **by** *simp*
moreover have *xvecFreshQ*: *xvec* ♯* *Q* **by** *fact*
with *xEq xFreshxvec′ yFreshxvec′* **have** *xvec′* ♯* ([(*x*, *y*)] ∙ *Q*) **by** *simp*
moreover have *Suc n* = *length xvec* **by** *fact*
with *xEq* **have** *n* = *length xvec′* **by** *simp*
moreover from *xvec′Freshyvec′ xFreshxvec′ yFreshxvec′* **have** *xvec′* ♯* ([(*x*, *y*)] ∙ *yvec′*)
by *simp*
moreover from *L′* **have** *length xvec′* = *length*([(*x*, *y*)] ∙ *yvec′*) **by** *simp*
ultimately have *distinct*([(*x*, *y*)] ∙ *yvec′*) **using** *xvec′Dist*
apply −
apply(*rule Suc*)
by(*assumption* | *simp*)+
then have *distinct yvec′* **by** *simp*
from *Mem xEq* **have** *xSuppN*: *x* ∈ *supp N* **by** *simp*
from *L* ‹*distinct xvec*› ‹*xvec* ♯* *yvec*› ‹*xvec* ♯* *M*› ‹*xvec* ♯* *Q*›
have (|*ν***yvec*|)*M* ≺′ *Q* = (|*ν***xvec*|)([*yvec xvec*] ∙*v* *M*) ≺′ ([*yvec xvec*] ∙*v* *Q*)
by(*simp add*: *alphaBoundOutputChain″*)
with *Eq* **have** *N* = [*yvec xvec*] ∙*v* *M* **by** *simp*
with *xEq yEq* **have** *N* = [(*y*, *x*)] ∙ [*yvec′ xvec′*] ∙*v* *M*
by *simp*
with *xSuppN* **have** *ySuppM*: *y* ∈ *supp*([*yvec′ xvec′*] ∙*v* *M*)
by(*force simp add*: *calc-atm eqvts name-swap*
dest: *pt-set-bij2*[**where** *pi*=[(*x*, *y*)],*OF pt-name-inst*, *OF at-name-inst*])
have *y* ♯ *yvec′*
proof −
{
assume *y* ∈ *supp yvec′*
then have *y* ∈ *set yvec′*
by(*induct yvec′*) (*auto simp add*: *supp-list-nil supp-list-cons supp-atm*)
moreover from ‹*xvec* ♯* *M*› *xEq xFreshxvec′* **have** *xvec′* ♯* *M* **by** *simp*

```
        ultimately have y ♯ [yvec' xvec'] ·v M using L' xvec'Freshyvec' xvec'Dist
          by(force intro: freshChainPerm)
         with ySuppM have False by(simp add: fresh-def)
        }
        then show ?thesis
         by(simp add: fresh-def, rule notI)
       qed
       with ‹distinct yvec'› yEq show ?case by simp
      qed
    qed
  } note res = this
  show ?thesis
    apply(rule actionCases[where α=α])
    using assms res
    by(auto simp add: residualInject supp-some)
qed

lemma boundOutputDistinct:
  fixes Ψ   :: 'b
    and P   :: ('a, 'b, 'c) psi
    and α   :: 'a action
    and P'  :: ('a, 'b, 'c) psi

assumes Ψ ▷ P ⟼α ≺ P'

shows distinct(bn α)
  using assms
proof(nominal-induct Ψ P x3==α ≺ P' avoiding: α P' rule: semantics.strong-induct)
  case cPar1
  then show ?case
    by(force intro: alphaDistinct boundOutputBindObject)
next
  case cPar2
  then show ?case
    by(force intro: alphaDistinct boundOutputBindObject)
next
  case (cBrComm1 Ψ Ψ_Q P M N P' A_P Ψ_P Q xvec Q' A_Q α P'')
  note ‹Ψ ⊗ Ψ_P ▷ Q ⟼ ¡M⦇ν*xvec⦈⟨N⟩ ≺ Q'›
 moreover from ‹xvec ♯* M› have bn (¡M⦇ν*xvec⦈⟨N⟩) ♯* subject (¡M⦇ν*xvec⦈⟨N⟩)
by simp
  moreover from ‹distinct xvec› have distinct (bn (¡M⦇ν*xvec⦈⟨N⟩)) by simp
   ultimately have someX: ⋀ x. x ∈ set (bn (¡M⦇ν*xvec⦈⟨N⟩)) ⟹ x ∈ supp
(object (¡M⦇ν*xvec⦈⟨N⟩))
    by (drule boundOutputBindObject) (assumption)

  note ‹¡M⦇ν*xvec⦈⟨N⟩ ≺ P' ‖ Q' = α ≺ P''›
  moreover from ‹distinct xvec› have distinct (bn (¡M⦇ν*xvec⦈⟨N⟩)) by simp
   moreover note ‹⋀x. x ∈ set (bn (¡M⦇ν*xvec⦈⟨N⟩)) ⟹ x ∈ supp (object
(¡M⦇ν*xvec⦈⟨N⟩))›
```

moreover from ‹*xvec* ♯* *α*› **have** *bn* (¡*M*(⦇*ν***xvec*⦈)⟨*N*⟩) ♯* *bn α* **by** *simp*
moreover from ‹*xvec* ♯* *α*› **have** *bn* (¡*M*(⦇*ν***xvec*⦈)⟨*N*⟩) ♯* *object α* **by** *simp*
moreover from ‹*xvec* ♯* *P*′′› **have** *bn* (¡*M*(⦇*ν***xvec*⦈)⟨*N*⟩) ♯* *P*′′ **by** *simp*
ultimately show *?case*
by(*rule alphaDistinct*)
next
case (*cBrComm2* Ψ Ψ*Q P M xvec N P*′ *AP* Ψ*P Q Q*′ *AQ α P*′′)
note ‹Ψ ⊗ Ψ*Q* ▷ *P* ⟼ ¡*M*(⦇*ν***xvec*⦈)⟨*N*⟩ ≺ *P*′›
moreover from ‹*xvec* ♯* *M*› **have** *bn* (¡*M*(⦇*ν***xvec*⦈)⟨*N*⟩) ♯* *subject* (¡*M*(⦇*ν***xvec*⦈)⟨*N*⟩) **by** *simp*
moreover from ‹*distinct xvec*› **have** *distinct* (*bn* (¡*M*(⦇*ν***xvec*⦈)⟨*N*⟩)) **by** *simp*
ultimately have *someX*: ⋀ *x*. *x* ∈ *set* (*bn* (¡*M*(⦇*ν***xvec*⦈)⟨*N*⟩)) ⟹ *x* ∈ *supp* (*object* (¡*M*(⦇*ν***xvec*⦈)⟨*N*⟩))
by (*drule boundOutputBindObject*) (*assumption*)

note ‹¡*M*(⦇*ν***xvec*⦈)⟨*N*⟩ ≺ *P*′ ‖ *Q*′ = *α* ≺ *P*′′›
moreover from ‹*distinct xvec*› **have** *distinct* (*bn* (¡*M*(⦇*ν***xvec*⦈)⟨*N*⟩)) **by** *simp*
moreover note ‹⋀*x*. *x* ∈ *set* (*bn* (¡*M*(⦇*ν***xvec*⦈)⟨*N*⟩)) ⟹ *x* ∈ *supp* (*object* (¡*M*(⦇*ν***xvec*⦈)⟨*N*⟩))›
moreover from ‹*xvec* ♯* *α*› **have** *bn* (¡*M*(⦇*ν***xvec*⦈)⟨*N*⟩) ♯* *bn α* **by** *simp*
moreover from ‹*xvec* ♯* *α*› **have** *bn* (¡*M*(⦇*ν***xvec*⦈)⟨*N*⟩) ♯* *object α* **by** *simp*
moreover from ‹*xvec* ♯* *P*′′› **have** *bn* (¡*M*(⦇*ν***xvec*⦈)⟨*N*⟩) ♯* *P*′′ **by** *simp*
ultimately show *?case*
by(*rule alphaDistinct*)
next
case(*cBrClose* Ψ *P M xvec N P*′ *α P*′′)
then show *?case* **by**(*simp add*: *residualInject*)
next
case(*cOpen* Ψ *P M xvec yvec N P*′ *x α P*′′)
note ‹*M*(⦇*ν**(*xvec*@*x*#*yvec*)⦈)⟨*N*⟩ ≺ *P*′ = *α* ≺ *P*′′›
moreover from ‹*xvec* ♯* *yvec*› ‹*x* ♯ *xvec*› ‹*x* ♯ *yvec*› ‹*distinct xvec*› ‹*distinct yvec*›
have *distinct*(*bn*(*M*(⦇*ν**(*xvec*@*x*#*yvec*)⦈)⟨*N*⟩))
by *auto* (*simp add*: *fresh-star-def fresh-def name-list-supp*)
moreover {
fix *y*
from ‹Ψ ▷ *P* ⟼*M*(⦇*ν**(*xvec*@*yvec*)⦈)⟨*N*⟩ ≺ *P*′› ‹*x* ∈ *supp N*› ‹*x* ♯ *xvec*› ‹*x* ♯ *yvec*› ‹*x* ♯ *M*› ‹*x* ♯ Ψ› ‹*distinct xvec*› ‹*distinct yvec*› ‹*xvec* ♯* Ψ› ‹*xvec* ♯* *P*› ‹*xvec* ♯* *M*› ‹*xvec* ♯* *yvec*› ‹*yvec* ♯* Ψ› ‹*yvec* ♯* *P*› ‹*yvec* ♯* *M*›
have Ψ ▷ (⦇*ν**x*⦈)*P* ⟼*M*(⦇*ν**(*xvec*@*x*#*yvec*)⦈)⟨*N*⟩ ≺ *P*′ **by**(*rule semantics.cOpen*)
moreover moreover from ‹*xvec* ♯* *M*› ‹*x* ♯ *M*› ‹*yvec* ♯* *M*›
have *bn*(*M*(⦇*ν**(*xvec*@*x*#*yvec*)⦈)⟨*N*⟩) ♯* (*subject*(*M*(⦇*ν**(*xvec*@*x*#*yvec*)⦈)⟨*N*⟩))
by *simp*
moreover note ‹*distinct*(*bn*(*M*(⦇*ν**(*xvec*@*x*#*yvec*)⦈)⟨*N*⟩))›
moreover assume *y* ∈ *set*(*bn*(*M*(⦇*ν**(*xvec*@*x*#*yvec*)⦈)⟨*N*⟩))

ultimately have *y* ∈ *supp*(*object*(*M*(⦇*ν**(*xvec*@*x*#*yvec*)⦈)⟨*N*⟩))
by(*metis boundOutputBindObject*)
}
moreover from ‹*xvec* ♯* *α*› ‹*x* ♯ *α*› ‹*yvec* ♯* *α*›

have $bn(M(\!|\nu*(xvec@x\#yvec)|\!)\langle N\rangle)\ \sharp*\ bn\ \alpha$ **and** $bn(M(\!|\nu*(xvec@x\#yvec)|\!)\langle N\rangle)$
$\sharp*\ object\ \alpha$ **by** *simp+*
moreover from ‹$xvec\ \sharp*\ P''$› ‹$x\ \sharp\ P''$› ‹$yvec\ \sharp*\ P''$›
have $bn(M(\!|\nu*(xvec@x\#yvec)|\!)\langle N\rangle)\ \sharp*\ P''$ **by** *simp*
ultimately show *?case* **by**(*rule alphaDistinct*)
next
case(*cBrOpen* $\Psi\ P\ M\ xvec\ yvec\ N\ P'\ x\ \alpha\ P''$)
note ‹$¡M(\!|\nu*(xvec@x\#yvec)|\!)\langle N\rangle \prec P' = \alpha \prec P''$›
moreover from ‹$xvec\ \sharp*\ yvec$› ‹$x\ \sharp\ xvec$› ‹$x\ \sharp\ yvec$› ‹*distinct xvec*› ‹*distinct yvec*›
have *distinct*($bn(¡M(\!|\nu*(xvec@x\#yvec)|\!)\langle N\rangle)$)
by *auto* (*simp add*: *fresh-star-def fresh-def name-list-supp*)
moreover {
fix *y*
from ‹$\Psi \rhd P \longmapsto ¡M(\!|\nu*(xvec@yvec)|\!)\langle N\rangle \prec P'$› ‹$x \in supp\ N$› ‹$x\ \sharp\ xvec$› ‹$x\ \sharp$
$yvec$› ‹$x\ \sharp\ M$› ‹$x\ \sharp\ \Psi$› ‹*distinct xvec*› ‹*distinct yvec*› ‹$xvec\ \sharp*\ \Psi$› ‹$xvec\ \sharp*\ P$› ‹$xvec$
$\sharp*\ M$› ‹$xvec\ \sharp*\ yvec$› ‹$yvec\ \sharp*\ \Psi$› ‹$yvec\ \sharp*\ P$› ‹$yvec\ \sharp*\ M$›
have $\Psi \rhd (\!|\nu x|\!)P \longmapsto ¡M(\!|\nu*(xvec@x\#yvec)|\!)\langle N\rangle \prec P'$ **by**(*rule semantics.cBrOpen*)
moreover moreover from ‹$xvec\ \sharp*\ M$› ‹$x\ \sharp\ M$› ‹$yvec\ \sharp*\ M$›
have $bn(¡M(\!|\nu*(xvec@x\#yvec)|\!)\langle N\rangle)\ \sharp*\ (subject(¡M(\!|\nu*(xvec@x\#yvec)|\!)\langle N\rangle))$
by *simp*
moreover note ‹$distinct(bn(¡M(\!|\nu*(xvec@x\#yvec)|\!)\langle N\rangle))$›
moreover assume $y \in set(bn(¡M(\!|\nu*(xvec@x\#yvec)|\!)\langle N\rangle))$

ultimately have $y \in supp(object(¡M(\!|\nu*(xvec@x\#yvec)|\!)\langle N\rangle))$
by(*metis boundOutputBindObject*)
}
moreover from ‹$xvec\ \sharp*\ \alpha$› ‹$x\ \sharp\ \alpha$› ‹$yvec\ \sharp*\ \alpha$›
have $bn(¡M(\!|\nu*(xvec@x\#yvec)|\!)\langle N\rangle)\ \sharp*\ bn\ \alpha$ **and** $bn(¡M(\!|\nu*(xvec@x\#yvec)|\!)\langle N\rangle)$
$\sharp*\ object\ \alpha$ **by** *simp+*
moreover from ‹$xvec\ \sharp*\ P''$› ‹$x\ \sharp\ P''$› ‹$yvec\ \sharp*\ P''$›
have $bn(¡M(\!|\nu*(xvec@x\#yvec)|\!)\langle N\rangle)\ \sharp*\ P''$ **by** *simp*
ultimately show *?case* **by**(*rule alphaDistinct*)
next
case *cScope*
then show *?case*
by − (*rule alphaDistinct*, *auto intro*: *boundOutputBindObject*)
qed (*simp-all add*: *residualInject*)

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*

using *assms*
by(*nominal-induct* Ψ $P{=}{=}M(\!|\lambda{*}xvec\ N|\!).P$ *Rs avoiding*: *xvec N P rule*: *semantics.strong-induct*)
(*auto simp add*: *psi.inject intro*: *alphaInputDistinct*)

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* :: $'f$::*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 :: $'f$::*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* $C\ \Psi\ P\ M\ xvec\ N\ P'$; $(\varphi, P) \in$ *set 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 $\Psi\ P\ M$ $(xvec@yvec)$ $N\ P'$;

$x \sharp \Psi$; $x \sharp M$; $x \sharp xvec$; $x \sharp yvec$; $xvec \sharp* \Psi$; $xvec \sharp* P$; $xvec \sharp* M$; $yvec \sharp* \Psi$; $yvec \sharp* P$; $yvec \sharp* M$; $yvec \sharp* C$; $x \sharp C$; $xvec \sharp* C\rrbracket \Longrightarrow$
Prop $C\ \Psi\ (\langle\!|\nu x|\!\rangle P)\ M\ (xvec@x\#yvec)\ N\ P'$

and *rScope*: $\bigwedge \Psi\ P\ M\ xvec\ N\ P'\ x\ C$.
$\llbracket \Psi \rhd P \longmapsto M\langle\!|\nu * xvec|\!\rangle\langle N\rangle \prec P'$; $\bigwedge C$. *Prop* $C\ \Psi\ P\ M\ xvec\ N\ P'$; $x \sharp \Psi$; $x \sharp M$; $x \sharp xvec$; $x \sharp N$; $xvec \sharp* \Psi$; $xvec \sharp* P$; $xvec \sharp* M$; $x \sharp C$; $xvec \sharp* C\rrbracket \Longrightarrow$
Prop $C\ \Psi\ (\langle\!|\nu x|\!\rangle P)\ M\ xvec\ N\ (\langle\!|\nu x|\!\rangle P')$

and *rBang*: $\bigwedge \Psi\ P\ M\ xvec\ N\ P'\ C$.
$\llbracket \Psi \rhd P \parallel\ !P \longmapsto M\langle\!|\nu * xvec|\!\rangle\langle N\rangle \prec P'$; *guarded* P; $\bigwedge C$. *Prop* $C\ \Psi\ (P \parallel\ !P)\ M\ xvec\ N\ P'\rrbracket \Longrightarrow$
Prop $C\ \Psi\ (!P)\ M\ xvec\ N\ P'$

shows *Prop* $C\ \Psi\ P\ M\ xvec\ N\ P'$
proof −
note ‹$\Psi \rhd P \longmapsto M\langle\!|\nu * xvec|\!\rangle\langle N\rangle \prec P'$›
moreover from ‹$xvec \sharp* M$› **have** $bn(M\langle\!|\nu * xvec|\!\rangle\langle N\rangle) \sharp* subject(M\langle\!|\nu * xvec|\!\rangle\langle N\rangle)$ **by** *simp*
moreover from ‹$\Psi \rhd P \longmapsto M\langle\!|\nu * xvec|\!\rangle\langle N\rangle \prec P'$› **have** $distinct(bn(M\langle\!|\nu * xvec|\!\rangle\langle N\rangle))$
by(*rule boundOutputDistinct*)
ultimately show *?thesis*
proof(*nominal-induct* $\Psi\ P\ \alpha == M\langle\!|\nu * xvec|\!\rangle\langle N\rangle\ P'$ *avoiding*: C *arbitrary*: $M\ xvec\ N$ *rule*: *semanticsInduct*)
case(*cAlpha* $\Psi\ P\ \alpha\ P'\ p\ C\ M\ xvec\ N$)
from ‹$(p \cdot \alpha) = M\langle\!|\nu * xvec|\!\rangle\langle N\rangle$› **have** $(p \cdot p \cdot \alpha) = p \cdot (M\langle\!|\nu * xvec|\!\rangle\langle N\rangle)$
by(*simp add*: *fresh-bij*)
with ‹*distinctPerm p*› **have** *A*: $\alpha = (p \cdot M)\langle\!|\nu * (p \cdot xvec)|\!\rangle\langle(p \cdot N)\rangle$
by(*simp add*: *eqvts*)
with ‹$bn\ \alpha \sharp* \Psi$› ‹$bn\ \alpha \sharp* P$› ‹$bn\ \alpha \sharp* subject\ \alpha$ › ‹$bn\ \alpha \sharp* C$› ‹$bn\ \alpha \sharp* bn(p \cdot \alpha)$› ‹*distinctPerm p*›
have $(p \cdot xvec) \sharp* \Psi$ **and** $(p \cdot xvec) \sharp* P$ **and** $(p \cdot xvec) \sharp* (p \cdot M)$ **and** $(p \cdot xvec) \sharp* C$ **and** $(p \cdot xvec) \sharp* (p \cdot p \cdot xvec)$
by *auto*
moreover from *A* ‹$set\ p \subseteq set(bn\ \alpha) \times set(bn(p \cdot \alpha))$› ‹*distinctPerm p*›
have *S*: $set\ p \subseteq set(p \cdot xvec) \times set(p \cdot p \cdot xvec)$ **by** *simp*
moreover note ‹*distinctPerm p*›
moreover from *A* ‹$bn(p \cdot \alpha) \sharp* \alpha$› ‹$bn(p \cdot \alpha) \sharp* P'$›
have $(p \cdot p \cdot xvec) \sharp* (p \cdot N)$ **and** $(p \cdot p \cdot xvec) \sharp* P'$ **by** *simp+*
moreover from *A* **have** *Prop* $C\ \Psi\ P\ (p \cdot M)\ (p \cdot xvec)\ (p \cdot N)\ P'$
by(*rule cAlpha*)
ultimately have *Prop* $C\ \Psi\ P\ (p \cdot M)\ (p \cdot p \cdot xvec)\ (p \cdot p \cdot N)\ (p \cdot P')$
by(*rule rAlpha*)
moreover from *A* ‹$bn\ \alpha \sharp* subject\ \alpha$› **have** $(p \cdot xvec) \sharp* (p \cdot M)$ **by** *simp*
then have $xvec \sharp* M$ **by**(*simp add*: *fresh-star-bij*)
from *A* ‹$bn(p \cdot \alpha) \sharp* \alpha$› ‹*distinctPerm p*› **have** $xvec \sharp* (p \cdot M)$ **by** *simp*
then have $(p \cdot xvec) \sharp* (p \cdot p \cdot M)$ **by**(*simp add*: *fresh-star-bij*)
with ‹*distinctPerm p*› **have** $(p \cdot xvec) \sharp* M$ **by** *simp*
with ‹$xvec \sharp* M$› *S* ‹*distinctPerm p*› **have** $(p \cdot M) = M$ **by** *simp*
ultimately show *?case* **using** *S* ‹*distinctPerm p*› **by** *simp*
next

```
  case cInput
  then show ?case by(simp add: residualInject)
next
  case cBrInput
  then show ?case by simp
next
  case cOutput
  then show ?case by(force dest: rOutput simp add: action.inject)
next
  case cBrOutput
  then show ?case by simp
next
  case cCase
  then show ?case by(force intro: rCase)
next
  case cPar1
  then show ?case by(force intro: rPar1)
next
  case cPar2
  then show ?case by(force intro: rPar2)
next
  case cComm1
  then show ?case by(simp add: action.inject)
next
  case cComm2
  then show ?case by(simp add: action.inject)
next
  case cBrMerge
  then show ?case by(simp add: action.inject)
next
  case cBrComm1
  then show ?case by simp
next
  case cBrComm2
  then show ?case by simp
next
  case cBrClose
  then show ?case by simp
next
  case cOpen
  then show ?case by(auto intro: rOpen simp add: action.inject)
next
  case cBrOpen
  then show ?case by simp
next
  case cScope
  then show ?case by(auto intro: rScope)
next
  case cBang
```

then show *?case* **by**(*auto intro*: *rBang*)
qed
qed

lemma *brOutputInduct'*[*consumes 2*, *case-names cAlpha cBrOutput cCase cPar1 cPar2 cBrComm1 cBrComm2 cBrOpen 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* :: *'f*::*fs-name* ⇒ *'b* ⇒ (*'a*, *'b*, *'c*) *psi* ⇒
'a ⇒ *name list* ⇒ *'a* ⇒ (*'a*, *'b*, *'c*) *psi* ⇒ *bool*
and C :: *'f*::*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* $\sharp*$ Ψ; *xvec* $\sharp*$ P; *xvec* $\sharp*$ M; *xvec* $\sharp*$ C; *xvec* $\sharp*$ $(p \cdot xvec)$;
set $p \subseteq$ *set xvec* × *set*$(p \cdot xvec)$; *distinctPerm* p;
$(p \cdot xvec)$ $\sharp*$ N; $(p \cdot xvec)$ $\sharp*$ P'; *Prop* C Ψ P M *xvec* N P'⟧ ⟹
Prop C Ψ P M $(p \cdot xvec)$ $(p \cdot N)$ $(p \cdot P')$
and *rBrOutput*: $\bigwedge \Psi\ M\ K\ N\ P\ C.$ ⟦$\Psi \vdash M \preceq K$⟧ ⟹ *Prop* C Ψ $(M\langle N\rangle.P)$ 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 Ψ P M *xvec* N P'; $(\varphi, P) \in$ *set Cs*; $\Psi \vdash \varphi$; *guarded* P⟧ ⟹
Prop C Ψ (*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 $\sharp*$ Ψ; A_Q $\sharp*$ M;
A_Q $\sharp*$ *xvec*; A_Q $\sharp*$ N; A_Q $\sharp*$ P'; A_Q $\sharp*$ C; *xvec* $\sharp*$ Q;
xvec $\sharp*$ Ψ; *xvec* $\sharp*$ Ψ_Q; *xvec* $\sharp*$ P; *xvec* $\sharp*$ M; *xvec* $\sharp*$ C⟧ ⟹
Prop C Ψ $(P \parallel Q)$ 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 $\sharp*$ Ψ; A_P $\sharp*$ M;
A_P $\sharp*$ *xvec*; A_P $\sharp*$ N; A_P $\sharp*$ Q'; A_P $\sharp*$ C; *xvec* $\sharp*$ Q;
xvec $\sharp*$ Ψ; *xvec* $\sharp*$ Ψ_P; *xvec* $\sharp*$ P; *xvec* $\sharp*$ M; *xvec* $\sharp*$ C⟧ ⟹
Prop C Ψ $(P \parallel Q)$ M *xvec* N $(P \parallel Q')$
and *rBrComm1*: $\bigwedge \Psi\ \Psi_Q\ P\ M\ N\ P'\ A_P\ \Psi_P\ Q\ xvec\ Q'\ A_Q\ C.$
⟦$\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$;
extractFrame $P = \langle A_P, \Psi_P\rangle$; *distinct* A_P;
$\Psi \otimes \Psi_P \rhd Q \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec Q'$; $\bigwedge C.$ *Prop* C $(\Psi \otimes \Psi_P)$

$Q\ M\ xvec\ N\ Q'$;
extractFrame $Q = \langle A_Q, \Psi_Q\rangle$; *distinct* A_Q;
$A_P \sharp* \Psi$; $A_P \sharp* \Psi_Q$; $A_P \sharp* P$; $A_P \sharp* N$; $A_P \sharp* P'$;
$A_P \sharp* Q$; $A_P \sharp* Q'$; $A_P \sharp* A_Q$; $A_P \sharp* xvec$; $A_Q \sharp* \Psi$; $A_Q \sharp* \Psi_P$;
$A_Q \sharp* P$; $A_Q \sharp* N$; $A_Q \sharp* P'$; $A_Q \sharp* Q$; $A_Q \sharp* Q'$; *distinct* $xvec$;
$A_P \sharp* M$; $A_Q \sharp* M$; $xvec \sharp* M$;
$A_Q \sharp* xvec$; $xvec \sharp* \Psi$; $xvec \sharp* \Psi_P$; $xvec \sharp* \Psi_Q$; $xvec \sharp* P$;
$xvec \sharp* Q$; $A_P \sharp* C$; $A_Q \sharp* C$; $xvec \sharp* C\rrbracket \Longrightarrow$
Prop $C\ \Psi\ (P \parallel Q)\ M\ xvec\ N\ (P' \parallel Q')$

and *rBrComm2*: $\bigwedge \Psi\ \Psi_Q\ P\ M\ xvec\ N\ P'\ A_P\ \Psi_P\ Q\ Q'\ A_Q\ C$.
$\llbracket \Psi \otimes \Psi_Q \rhd P \longmapsto ¡M(\!|\nu* xvec|\!)\langle N\rangle \prec P'$; $\bigwedge C$. *Prop* $C\ (\Psi \otimes \Psi_Q)\ P\ M\ xvec\ N\ P'$;
extractFrame $P = \langle A_P, \Psi_P\rangle$; *distinct* A_P;
$\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$;
extractFrame $Q = \langle A_Q, \Psi_Q\rangle$; *distinct* A_Q;
$A_P \sharp* \Psi$; $A_P \sharp* \Psi_Q$; $A_P \sharp* P$; $A_P \sharp* N$; $A_P \sharp* P'$;
$A_P \sharp* Q$; $A_P \sharp* Q'$; $A_P \sharp* A_Q$; $A_P \sharp* xvec$; $A_Q \sharp* \Psi$; $A_Q \sharp* \Psi_P$;
$A_Q \sharp* P$; $A_Q \sharp* N$; $A_Q \sharp* P'$; $A_Q \sharp* Q$; $A_Q \sharp* Q'$; *distinct* $xvec$;
$A_P \sharp* M$; $A_Q \sharp* M$; $xvec \sharp* M$;
$A_Q \sharp* xvec$; $xvec \sharp* \Psi$; $xvec \sharp* \Psi_P$; $xvec \sharp* \Psi_Q$; $xvec \sharp* P$;
$xvec \sharp* Q$; $A_P \sharp* C$; $A_Q \sharp* C$; $xvec \sharp* C\rrbracket \Longrightarrow$
Prop $C\ \Psi\ (P \parallel Q)\ M\ xvec\ N\ (P' \parallel Q')$

and *rOpen*: $\bigwedge \Psi\ P\ M\ xvec\ yvec\ N\ P'\ x\ C$.
$\llbracket \Psi \rhd P \longmapsto ¡M(\!|\nu*(xvec@yvec)|\!)\langle N\rangle \prec P'$; $x \in supp\ N$; $\bigwedge C$. *Prop* $C\ \Psi\ P\ M\ (xvec@yvec)\ N\ P'$;
$x \sharp \Psi$; $x \sharp M$; $x \sharp xvec$; $x \sharp yvec$; $xvec \sharp* \Psi$; $xvec \sharp* P$; $xvec \sharp* M$;
$yvec \sharp* \Psi$; $yvec \sharp* P$; $yvec \sharp* M$; $yvec \sharp* C$; $x \sharp C$; $xvec \sharp* C\rrbracket \Longrightarrow$
Prop $C\ \Psi\ ((\!|\nu x|\!)P)\ M\ (xvec@x\#yvec)\ N\ P'$

and *rScope*: $\bigwedge \Psi\ P\ M\ xvec\ N\ P'\ x\ C$.
$\llbracket \Psi \rhd P \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec P'$; $\bigwedge C$. *Prop* $C\ \Psi\ P\ M\ xvec\ N\ P'$;
$x \sharp \Psi$; $x \sharp M$; $x \sharp xvec$; $x \sharp N$; $xvec \sharp* \Psi$;
$xvec \sharp* P$; $xvec \sharp* M$; $x \sharp C$; $xvec \sharp* C\rrbracket \Longrightarrow$
Prop $C\ \Psi\ ((\!|\nu x|\!)P)\ M\ xvec\ N\ ((\!|\nu x|\!)P')$

and *rBang*: $\bigwedge \Psi\ P\ M\ xvec\ N\ P'\ C$.
$\llbracket \Psi \rhd P \parallel !P \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec P'$; *guarded* P; $\bigwedge C$. *Prop* $C\ \Psi\ (P \parallel !P)\ M\ xvec\ N\ P'\rrbracket \Longrightarrow$
Prop $C\ \Psi\ (!P)\ M\ xvec\ N\ P'$

shows *Prop* $C\ \Psi\ P\ M\ xvec\ N\ P'$
proof −
note $\langle \Psi \rhd P \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec P'\rangle$
moreover from $\langle xvec \sharp* M\rangle$ **have** $bn(¡M(\!|\nu*xvec|\!)\langle N\rangle) \sharp* subject(¡M(\!|\nu*xvec|\!)\langle N\rangle)$
by *simp*
moreover from $\langle \Psi \rhd P \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec P'\rangle$ **have** $distinct(bn(¡M(\!|\nu*xvec|\!)\langle N\rangle))$
by(*rule boundOutputDistinct*)
ultimately show *?thesis*
proof(*nominal-induct* $\Psi\ P\ \alpha == ¡M(\!|\nu*xvec|\!)\langle N\rangle\ P'$ *avoiding*: C *arbitrary*: $M\ xvec\ N$ *rule*: *semanticsInduct*)
case(*cAlpha* $\Psi\ P\ \alpha\ P'\ p\ C\ M\ xvec\ N$)
from $\langle (p \cdot \alpha) = ¡M(\!|\nu*xvec|\!)\langle N\rangle\rangle$ **have** $(p \cdot p \cdot \alpha) = p \cdot (¡M(\!|\nu*xvec|\!)\langle N\rangle)$

```
    by(simp add: fresh-bij)
  with ‹distinctPerm p› have A: α = ¡(p • M)⦇ν*(p • xvec)⦈⟨(p • N)⟩
    by(simp add: eqvts)
  with ‹bn α ♯* Ψ› ‹bn α ♯* P› ‹bn α ♯* subject α › ‹bn α ♯* C› ‹bn α ♯* bn(p
• α)› ‹distinctPerm p›
  have (p • xvec) ♯* Ψ and (p • xvec) ♯* P and (p • xvec) ♯* (p • M) and (p
• xvec) ♯* C and (p • xvec) ♯* (p • p • xvec)
    by auto
  moreover from A ‹set p ⊆ set(bn α) × set(bn(p • α))› ‹distinctPerm p›
  have S: set p ⊆ set(p • xvec) × set(p • p • xvec) by simp
  moreover note ‹distinctPerm p›
  moreover from A ‹bn(p • α) ♯* α› ‹bn(p • α) ♯* P'›
  have (p • p • xvec) ♯* (p • N) and (p • p • xvec) ♯* P' by simp+
  moreover from A have Prop C Ψ P (p • M) (p • xvec) (p • N) P'
    by(rule cAlpha)
  ultimately have Prop C Ψ P (p • M) (p • p • xvec) (p • p • N) (p • P')
    by(rule rAlpha)
  moreover from A ‹bn α ♯* subject α› have (p • xvec) ♯* (p • M) by simp
  then have xvec ♯* M by(simp add: fresh-star-bij)
  from A ‹bn(p • α) ♯* α› ‹distinctPerm p› have xvec ♯* (p • M) by simp
  then have (p • xvec) ♯* (p • p • M) by(simp add: fresh-star-bij)
  with ‹distinctPerm p› have (p • xvec) ♯* M by simp
  with ‹xvec ♯* M› S ‹distinctPerm p› have (p • M) = M by simp
  ultimately show ?case using S ‹distinctPerm p› by simp
next
  case cInput
  then show ?case by(simp add: residualInject)
next
  case cBrInput
  then show ?case by simp
next
  case cOutput
  then show ?case by simp
next
  case cBrOutput
  then show ?case by(simp add: rBrOutput action.inject)
next
  case cCase
  then show ?case by(force intro: rCase)
next
  case cPar1
  then show ?case by(force intro: rPar1)
next
  case cPar2
  then show ?case by(force intro: rPar2)
next
  case cComm1
  then show ?case by(simp add: action.inject)
next
```

```
    case cComm2
    then show ?case by(simp add: action.inject)
  next
    case cBrMerge
    then show ?case by(simp add: action.inject)
  next
    case cBrComm1
    then show ?case
      by(auto intro: rBrComm1 simp add: action.inject)
  next
    case cBrComm2
    then show ?case
      by(auto intro: rBrComm2 simp add: action.inject)
  next
    case cBrClose
    then show ?case by simp
  next
    case cOpen
    then show ?case by simp
  next
    case cBrOpen
    then show ?case by(auto intro: rOpen simp add: action.inject)
  next
    case cScope
    then show ?case by(auto intro: rScope)
  next
    case cBang
    then show ?case by(auto intro: rBang)
  qed
qed

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 :: 'f::fs-name ⇒ 'b ⇒ ('a, 'b, 'c) psi ⇒
                'a ⇒ 'a ⇒ ('a, 'b, 'c) psi ⇒ bool
    and C    :: 'f::fs-name

assumes Trans: Ψ ▷ P ⟼M⦇N⦈ ≺ P'
  and   rInput: ⋀Ψ M K xvec N Tvec P C.
                  ⟦Ψ ⊢ M ↔ K; distinct xvec; set xvec ⊆ supp N;
                   length xvec = length Tvec; xvec ♯* Ψ;
                   xvec ♯* M; xvec ♯* K; xvec ♯* C⟧ ⟹
                   Prop C Ψ (M⦇λ*xvec N⦈.P)
                            K (N[xvec::=Tvec]) (P[xvec::=Tvec])
```

and *rCase*: $\bigwedge \Psi\ P\ M\ N\ P'\ \varphi\ Cs\ C.\ [\![\Psi \rhd P \longmapsto M(\!|N|\!) \prec P';\ \bigwedge C.\ Prop\ C\ \Psi\ P\ M\ N\ P';\ (\varphi,\ P) \in set\ Cs;\ \Psi \vdash \varphi;\ guarded\ P]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (Cases\ Cs)\ M\ N\ P'$
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 $Prop\ C\ \Psi\ P\ M\ N\ P'$
using *Trans*
proof(*nominal-induct* $\Psi\ P\ Rs==M(\!|N|\!) \prec P'$ *avoiding*: C *arbitrary*: P' *rule*: *semantics.strong-induct*)
case(*cInput* $\Psi\ M\ K\ xvec\ N\ Tvec\ P\ C$)
then show *?case*
by(*force intro*: *rInput simp add*: *residualInject action.inject*)
next
case(*cBrInput* $\Psi\ M\ K\ xvec\ N\ Tvec\ P\ C$)
then show *?case*
by (*simp add*: *residualInject*)
next
case(*Output* $\Psi\ M\ K\ N\ P\ C$)
then show *?case* **by**(*simp add*: *residualInject*)
next
case *BrOutput*
then show *?case* **by**(*simp add*: *residualInject*)
next
case(*Case* $\Psi\ P\ \varphi\ CS\ C\ P'$)
then show *?case* **by**(*force intro*: *rCase*)
next
case(*cPar1* $\Psi\ \Psi_Q\ P\ \alpha\ P'\ Q\ A_Q\ C\ P''$)
then show *?case* **by**(*force intro*: *rPar1 simp add*: *residualInject*)
next
case(*cPar2* $\Psi\ \Psi P\ Q\ \alpha\ Q'\ xvec\ P\ C\ Q''$)

then show *?case* **by**(*force intro*: *rPar2 simp add*: *residualInject*)
next
case(*cComm1* Ψ ΨQ P M N P' *xvec* ΨP Q K *zvec* Q' *yvec* C PQ)
then show *?case* **by**(*simp add*: *residualInject*)
next
case(*cComm2* Ψ ΨQ P M *zvec* N P' *xvec* ΨP Q K *yvec* Q' C PQ)
then show *?case* **by**(*simp add*: *residualInject*)
next
case *cBrMerge*
then show *?case* **by**(*simp add*: *residualInject*)
next
case *cBrComm1*
then show *?case* **by**(*simp add*: *residualInject*)
next
case *cBrComm2*
then show *?case* **by**(*simp add*: *residualInject*)
next
case(*cBrClose* Ψ P M *xvec* N P' C P'')
then show *?case* **by**(*simp add*: *residualInject*)
next
case(*cOpen* Ψ P M *xvec* N P' x *yvec* C P'')
then show *?case* **by**(*simp add*: *residualInject*)
next
case(*cBrOpen* Ψ P M *xvec* N P' x *yvec* C P'')
then show *?case* **by**(*simp add*: *residualInject*)
next
case(*cScope* Ψ P α P' x C P'')
then show *?case* **by**(*force intro*: *rScope simp add*: *residualInject*)
next
case(*Bang* Ψ P C P')
then show *?case* **by**(*force intro*: *rBang*)
qed

lemma *brInputInduct*[*consumes 1*, *case-names cBrInput cCase cPar1 cPar2 cBrMerge 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* :: *'f*::*fs-name* ⇒ *'b* ⇒ (*'a*, *'b*, *'c*) *psi* ⇒
'a ⇒ *'a* ⇒ (*'a*, *'b*, *'c*) *psi* ⇒ *bool*
and C :: *'f*::*fs-name*

assumes *Trans*: $\Psi \rhd P \longmapsto ¿M\langle\!|N|\!\rangle \prec P'$
and *rBrInput*: $\bigwedge \Psi\ K\ M\ xvec\ N\ Tvec\ P\ C.$
$[\![\Psi \vdash K \succeq M;\ distinct\ xvec;\ set\ xvec \subseteq supp\ N;$
$length\ xvec = length\ Tvec;\ xvec \sharp* \Psi;$
$xvec \sharp* M;\ xvec \sharp* K;\ xvec \sharp* C]\!] \Longrightarrow$

$Prop\ C\ \Psi\ (M(\!|\lambda{*}xvec\ N|\!).P)$
$K\ (N[xvec::=Tvec])\ (P[xvec::=Tvec])$

and *rCase*: $\bigwedge \Psi\ P\ M\ N\ P'\ \varphi\ Cs\ C.\ [\![\Psi \rhd P \longmapsto ¿M(\!|N|\!) \prec P';\ \bigwedge C.\ Prop\ C\ \Psi\ P\ M\ N\ P';\ (\varphi, P) \in set\ Cs;\ \Psi \vdash \varphi;\ guarded\ P]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (Cases\ Cs)\ M\ N\ P'$

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 *rBrMerge*: $\bigwedge \Psi\ \Psi_Q\ P\ M\ N\ P'\ A_P\ \Psi_P\ Q\ Q'\ A_Q\ C.$
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P';\ \bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_Q)\ P\ M\ N\ P';$
$extractFrame\ P = \langle A_P, \Psi_P\rangle;\ distinct\ A_P;$
$\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q';\ \bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_P)\ Q\ M\ N\ Q';$
$extractFrame\ Q = \langle A_Q, \Psi_Q\rangle;\ distinct\ A_Q;$
$A_P \sharp* \Psi;\ A_P \sharp* \Psi_Q;\ A_P \sharp* P;\ A_P \sharp* N;\ A_P \sharp* P';$
$A_P \sharp* Q;\ A_P \sharp* Q';\ A_P \sharp* A_Q;\ A_P \sharp* M;\ A_Q \sharp* M;$
$A_Q \sharp* \Psi;\ A_Q \sharp* \Psi_P;\ A_Q \sharp* P;\ A_Q \sharp* N;\ A_Q \sharp* P';$
$A_Q \sharp* Q;\ A_Q \sharp* Q';\ A_P \sharp* C;\ A_Q \sharp* C;$
$A_P \sharp* M;\ A_Q \sharp* M]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (P \parallel Q)\ M\ N\ (P' \parallel Q')$

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 $Prop\ C\ \Psi\ P\ M\ N\ P'$

using *Trans*

proof(*nominal-induct* $\Psi\ P\ Rs==¿M(\!|N|\!) \prec P'$ *avoiding*: C *arbitrary*: P' *rule*: *semantics.strong-induct*)

case(*cInput* $\Psi\ M\ K\ xvec\ N\ Tvec\ P\ C$)

then show *?case* **by** (*simp add*: *residualInject*)

next

case(*cBrInput* $\Psi\ K\ M\ xvec\ N\ Tvec\ P\ C$)

then show *?case*

by(*auto intro*: *rBrInput simp add*: *residualInject action.inject*)

next
case(*Output* Ψ M K N P C)
then show *?case* **by**(*simp add*: *residualInject*)
next
case *BrOutput*
then show *?case* **by**(*simp add*: *residualInject*)
next
case(*Case* Ψ P φ CS C P')
then show *?case* **by**(*force intro*: *rCase*)
next
case(*cPar1* Ψ Ψ_Q P α P' Q A_Q C P'')
then show *?case* **by**(*force intro*: *rPar1 simp add*: *residualInject*)
next
case(*cPar2* Ψ ΨP Q α Q' *xvec* P C Q'')
then show *?case* **by**(*force intro*: *rPar2 simp add*: *residualInject*)
next
case(*cComm1* Ψ ΨQ P M N P' *xvec* ΨP Q K *zvec* Q' *yvec* C PQ)
then show *?case* **by**(*simp add*: *residualInject*)
next
case(*cComm2* Ψ ΨQ P M *zvec* N P' *xvec* ΨP Q K *yvec* Q' C PQ)
then show *?case* **by**(*simp add*: *residualInject*)
next
case *cBrMerge*
then show *?case* **by**(*auto intro*: *rBrMerge simp add*: *residualInject action.inject*)
next
case *cBrComm1*
then show *?case* **by**(*simp add*: *residualInject*)
next
case *cBrComm2*
then show *?case* **by**(*simp add*: *residualInject*)
next
case(*cBrClose* Ψ P M *xvec* N P' C P'')
then show *?case* **by**(*simp add*: *residualInject*)
next
case(*cOpen* Ψ P M *xvec* N P' x *yvec* C P'')
then show *?case* **by**(*simp add*: *residualInject*)
next
case(*cBrOpen* Ψ P M *xvec* N P' x *yvec* C P'')
then show *?case* **by**(*simp add*: *residualInject*)
next
case(*cScope* Ψ P α P' x C P'')
then show *?case* **by**(*force intro*: *rScope simp add*: *residualInject*)
next
case(*Bang* Ψ P C P')
then show *?case* **by**(*force intro*: *rBang*)
qed

lemma *tauInduct*[*consumes 1*, *case-names cCase cPar1 cPar2 cComm1 cComm2 cBrClose cScope cBang*]:

fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and Rs :: $('a, 'b, 'c)$ *residual*
and *Prop* :: $'f$::*fs-name* $\Rightarrow$ $'b$ $\Rightarrow$ $('a, 'b, 'c)$ *psi* $\Rightarrow$
$('a, 'b, 'c)$ *psi* $\Rightarrow$ *bool*
and C :: $'f$::*fs-name*

assumes *Trans*: $\Psi \rhd P \longmapsto \tau \prec P'$
and *rCase*: $\bigwedge \Psi\ P\ P'\ \varphi\ Cs\ C.\ [\![\Psi \rhd P \longmapsto \tau \prec P';\ \bigwedge C.\ Prop\ C\ \Psi\ P\ P';\ (\varphi, P) \in set\ Cs;\ \Psi \vdash \varphi;\ guarded\ P]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (Cases\ Cs)\ P'$
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*$\rrbracket \Longrightarrow$
Prop C Ψ (*P* $\parallel$ *Q*) ($\langle\!|\nu*xvec|\!\rangle$(*P'* $\parallel$ *Q'*))
and *rBrClose*: $\bigwedge\Psi$ *P M xvec N P'* A_P Ψ_P *x C*.
$\llbracket\Psi \rhd P \longmapsto {\downarrow}M\langle\!|\nu*xvec|\!\rangle\langle N\rangle \prec P'$;
x $\in$ *supp M*;
distinct xvec; *xvec* $\sharp*$ Ψ; *xvec* $\sharp*$ *P*;
xvec $\sharp*$ *M*;
x $\sharp$ Ψ; *x* $\sharp$ *xvec*$\rrbracket \Longrightarrow$
Prop C Ψ ($\langle\!|\nu x|\!\rangle$*P*) ($\langle\!|\nu x|\!\rangle$($\langle\!|\nu*xvec|\!\rangle$*P'*))
and *rScope*: $\bigwedge\Psi$ *P P' x C*.
$\llbracket\Psi \rhd P \longmapsto\tau \prec P'$; $\bigwedge$*C*. *Prop C* Ψ *P P'*; *x* $\sharp$ Ψ; *x* $\sharp$ *C*$\rrbracket \Longrightarrow$
Prop C Ψ ($\langle\!|\nu x|\!\rangle$*P*) ($\langle\!|\nu x|\!\rangle$*P'*)
and *rBang*: $\bigwedge\Psi$ *P P' C*.
$\llbracket\Psi \rhd P \parallel !P \longmapsto\tau \prec P'$; *guarded P*; $\bigwedge$*C*. *Prop C* Ψ (*P* $\parallel$!*P*) *P'*$\rrbracket$
$\Longrightarrow$ *Prop C* Ψ (!*P*) *P'*
shows *Prop C* Ψ *P P'*
using *Trans*
proof(*nominal-induct* Ψ *P Rs*==$\tau \prec P'$ *avoiding*: *C arbitrary*: *P'* *rule*: *semantics.strong-induct*)
case(*cInput M K xvec N Tvec P C*)
then show *?case* **by**(*simp add*: *residualInject*)
next
case *cBrInput*
then show *?case* **by**(*simp add*: *residualInject*)
next
case(*Output* Ψ *M K N P C*)
then show *?case* **by**(*simp add*: *residualInject*)
next
case *BrOutput*
then show *?case* **by**(*simp add*: *residualInject*)
next
case(*Case* Ψ *P* φ *Cs C P'*)
then show *?case* **by**(*force intro*: *rCase simp add*: *residualInject*)
next
case(*cPar1* Ψ Ψ_Q *P* α *P'* A_Q *Q C P''*)
then show *?case* **by**(*force intro*: *rPar1 simp add*: *residualInject*)
next
case(*cPar2* Ψ Ψ_P *Q* α *Q'* A_P *P C Q''*)
then show *?case* **by**(*force intro*: *rPar2 simp add*: *residualInject*)
next
case(*cComm1* Ψ Ψ_Q *P M N P'* A_P Ψ_P *Q K xvec Q'* A_Q *C PQ*)
then show *?case* **by**(*force intro*: *rComm1 simp add*: *residualInject*)
next
case(*cComm2* Ψ Ψ_Q *P M xvec N P'* A_P Ψ*P Q'* A_Q *C PQ*)
then show *?case* **by**(*force intro*: *rComm2 simp add*: *residualInject*)
next
case *cBrMerge*
then show *?case* **by**(*simp add*: *residualInject*)
next

case *cBrComm1*
then show *?case* **by**(*simp add*: *residualInject*)
next
case *cBrComm2*
then show *?case* **by**(*simp add*: *residualInject*)
next
case *cBrClose*
then show *?case* **by**(*force intro*: *rBrClose simp add*: *residualInject*)
next
case(*cOpen* Ψ *P M xvec N P' x yvec C P''*)
then show *?case* **by**(*simp add*: *residualInject*)
next
case(*cBrOpen* Ψ *P M xvec N P' x yvec C P''*)
then show *?case* **by**(*simp add*: *residualInject*)
next
case(*cScope* Ψ *P* α *P' x C P''*)
then show *?case* **by**(*force intro*: *rScope simp add*: *residualInject*)
next
case(*Bang* Ψ *P C P'*)
then show *?case* **by**(*force intro*: *rBang simp add*: *residualInject*)
qed

lemma *semanticsFrameInduct*[*consumes 3*, *case-names cAlpha cInput cBrInput cOutput cBrOutput cCase cPar1 cPar2 cComm1 cComm2 cBrMerge cBrComm1 cBrComm2 cBrClose cOpen cBrOpen 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* :: *'f*::*fs-name* $\Rightarrow$ *'b* $\Rightarrow$ (*'a*, *'b*, *'c*) *psi* $\Rightarrow$
(*'a*, *'b*, *'c*) *residual* $\Rightarrow$ *name list* $\Rightarrow$ *'b* $\Rightarrow$ *bool*
and *C* :: *'f*::*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 Ψ *P Rs* A_P $\Psi_P$$]\!] \Longrightarrow$ *Prop C* Ψ *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* $\sharp*$ Ψ;
xvec $\sharp*$ *M*; *xvec* $\sharp*$ *K*; *xvec* $\sharp*$ *C*$]\!] \Longrightarrow$
Prop C Ψ $(M(\!|\lambda* xvec\ N|\!).P)$
$(K(\!|(N[xvec::=Tvec])|\!) \prec (P[xvec::=Tvec]))$ ([]) (**1**)
and *rBrInput*: $\bigwedge \Psi\ M\ K\ xvec\ N\ Tvec\ P\ C.$

$[\![\Psi \vdash K \succeq M;\ \mathit{distinct}\ \mathit{xvec};\ \mathit{set}\ \mathit{xvec} \subseteq \mathit{supp}\ N;$
$\mathit{length}\ \mathit{xvec} = \mathit{length}\ \mathit{Tvec};\ \mathit{xvec} \sharp* \Psi;$
$\mathit{xvec} \sharp* M;\ \mathit{xvec} \sharp* K;\ \mathit{xvec} \sharp* C]\!] \Longrightarrow$
$\mathit{Prop}\ C\ \Psi\ (M(\!|\lambda*\mathit{xvec}\ N|\!).P)$
$(¿K(\!|(N[\mathit{xvec}::=\mathit{Tvec}])|\!) \prec (P[\mathit{xvec}::=\mathit{Tvec}]))\ ([])\ (\mathbf{1})$

and *rOutput*: $\bigwedge \Psi\ M\ K\ N\ P\ C.\ \Psi \vdash M \leftrightarrow K \Longrightarrow \mathit{Prop}\ C\ \Psi\ (M\langle N\rangle.P)\ (K\langle N\rangle \prec P)\ ([])\ (\mathbf{1})$

and *rBrOutput*: $\bigwedge \Psi\ M\ K\ N\ P\ C.\ \Psi \vdash M \preceq K \Longrightarrow \mathit{Prop}\ C\ \Psi\ (M\langle N\rangle.P)\ (¡K\langle N\rangle \prec P)\ ([])\ (\mathbf{1})$

and *rCase*: $\bigwedge \Psi\ P\ \mathit{Rs}\ \varphi\ \mathit{Cs}\ A_P\ \Psi_P\ C.\ [\![\Psi \rhd P \longmapsto \mathit{Rs};\ \mathit{extractFrame}\ P = \langle A_P, \Psi_P\rangle;\ \mathit{distinct}\ A_P;\ \bigwedge C.\ \mathit{Prop}\ C\ \Psi\ P\ \mathit{Rs}\ A_P\ \Psi_P;$
$(\varphi, P) \in \mathit{set}\ \mathit{Cs};\ \Psi \vdash \varphi;\ \mathit{guarded}\ P;\ \Psi_P \simeq \mathbf{1};$
$(\mathit{supp}\ \Psi_P) = (\{\}::\mathit{name}\ \mathit{set});$
$A_P \sharp* \Psi;\ A_P \sharp* P;\ A_P \sharp* \mathit{Rs};\ A_P \sharp* C]\!] \Longrightarrow$
$\mathit{Prop}\ C\ \Psi\ (\mathit{Cases}\ \mathit{Cs})\ \mathit{Rs}\ ([])\ (\mathbf{1})$

and *rPar1*: $\bigwedge \Psi\ \Psi_Q\ P\ \alpha\ P'\ A_Q\ Q\ A_P\ \Psi_P\ C.$
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto \alpha \prec P';$
$\mathit{extractFrame}\ P = \langle A_P, \Psi_P\rangle;\ \mathit{distinct}\ A_P;$
$\mathit{extractFrame}\ Q = \langle A_Q, \Psi_Q\rangle;\ \mathit{distinct}\ A_Q;$
$\bigwedge C.\ \mathit{Prop}\ C\ (\Psi \otimes \Psi_Q)\ P\ (\alpha \prec P')\ A_P\ \Psi_P;\ \mathit{distinct}(\mathit{bn}\ \alpha);$
$A_P \sharp* P;\ A_P \sharp* Q;\ A_P \sharp* \Psi;\ A_P \sharp* \alpha;\ A_P \sharp* P';\ A_P \sharp* A_Q;\ A_P \sharp* \Psi_Q;$
$A_Q \sharp* P;\ A_Q \sharp* Q;\ A_Q \sharp* \Psi;\ A_Q \sharp* \alpha;\ A_Q \sharp* P';\ A_Q \sharp* \Psi_P;$
$\mathit{bn}\ \alpha \sharp* \Psi;\ \mathit{bn}\ \alpha \sharp* P;\ \mathit{bn}\ \alpha \sharp* Q;\ \mathit{bn}\ \alpha \sharp* \mathit{subject}\ \alpha;\ \mathit{bn}\ \alpha \sharp* \Psi_P;$
$\mathit{bn}\ \alpha \sharp* \Psi_Q;$
$A_P \sharp* C;\ A_Q \sharp* C;\ \mathit{bn}\ \alpha \sharp* C]\!] \Longrightarrow$
$\mathit{Prop}\ C\ \Psi\ (P \parallel Q)\ (\alpha \prec (P' \parallel Q))\ (A_P @ A_Q)\ (\Psi_P \otimes \Psi_Q)$

and *rPar2*: $\bigwedge \Psi\ \Psi_P\ Q\ \alpha\ Q'\ A_P\ P\ A_Q\ \Psi_Q\ C.$
$[\![\Psi \otimes \Psi_P \rhd Q \longmapsto \alpha \prec Q';$
$\mathit{extractFrame}\ P = \langle A_P, \Psi_P\rangle;\ \mathit{distinct}\ A_P;$
$\mathit{extractFrame}\ Q = \langle A_Q, \Psi_Q\rangle;\ \mathit{distinct}\ A_Q;$
$\bigwedge C.\ \mathit{Prop}\ C\ (\Psi \otimes \Psi_P)\ Q\ (\alpha \prec Q')\ A_Q\ \Psi_Q;\ \mathit{distinct}(\mathit{bn}\ \alpha);$
$A_P \sharp* P;\ A_P \sharp* Q;\ A_P \sharp* \Psi;\ A_P \sharp* \alpha;\ A_P \sharp* Q';\ A_P \sharp* A_Q;\ A_P \sharp* \Psi_Q;$
$A_Q \sharp* P;\ A_Q \sharp* Q;\ A_Q \sharp* \Psi;\ A_Q \sharp* \alpha;\ A_Q \sharp* Q';\ A_Q \sharp* \Psi_P;$
$\mathit{bn}\ \alpha \sharp* \Psi;\ \mathit{bn}\ \alpha \sharp* P;\ \mathit{bn}\ \alpha \sharp* Q;\ \mathit{bn}\ \alpha \sharp* \mathit{subject}\ \alpha;\ \mathit{bn}\ \alpha \sharp* \Psi_P;$
$\mathit{bn}\ \alpha \sharp* \Psi_Q;$
$A_P \sharp* C;\ A_Q \sharp* C;\ \mathit{bn}\ \alpha \sharp* C]\!] \Longrightarrow$
$\mathit{Prop}\ C\ \Psi\ (P \parallel Q)\ (\alpha \prec (P \parallel Q'))\ (A_P @ A_Q)\ (\Psi_P \otimes \Psi_Q)$

and *rComm1*: $\bigwedge \Psi\ \Psi_Q\ P\ M\ N\ P'\ A_P\ \Psi_P\ Q\ K\ \mathit{xvec}\ Q'\ A_Q\ C.$
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|N|\!) \prec P';\ \mathit{extractFrame}\ P = \langle A_P, \Psi_P\rangle;$
$\mathit{distinct}\ A_P;$
$\bigwedge C.\ \mathit{Prop}\ C\ (\Psi \otimes \Psi_Q)\ P\ ((M(\!|N|\!)) \prec P')\ A_P\ \Psi_P;$
$\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|\nu*\mathit{xvec}|\!)\langle N\rangle \prec Q';\ \mathit{extractFrame}\ Q = \langle A_Q, \Psi_Q\rangle;\ \mathit{distinct}\ A_Q;$
$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K;$
$\bigwedge C.\ \mathit{Prop}\ C\ (\Psi \otimes \Psi_P)\ Q\ (K(\!|\nu*\mathit{xvec}|\!)\langle N\rangle \prec Q')\ A_Q\ \Psi_Q;\ \mathit{distinct}\ \mathit{xvec};$
$A_P \sharp* \Psi;\ A_P \sharp* \Psi_Q;\ A_P \sharp* P;\ A_P \sharp* M;\ A_P \sharp* N;\ A_P \sharp* P';$

$A_P \sharp* Q$; $A_P \sharp* Q'$; $A_P \sharp* A_Q$; $A_P \sharp* xvec$; $A_Q \sharp* \Psi$; $A_Q \sharp* \Psi_P$; $A_Q \sharp* P$; $A_Q \sharp* N$; $A_Q \sharp* P'$; $A_Q \sharp* Q$; $A_Q \sharp* K$; $A_Q \sharp* Q'$; $A_Q \sharp* xvec$; $xvec \sharp* \Psi$; $xvec \sharp* \Psi_P$; $xvec \sharp* \Psi_Q$; $xvec \sharp* P$; $xvec \sharp* M$; $xvec \sharp* Q$; $xvec \sharp* K$; $A_P \sharp* C$; $A_Q \sharp* C$; $xvec \sharp* C$⟧ ⟹
Prop C Ψ $(P \parallel Q)$ $(\tau \prec (\!|\nu* xvec|\!)(P' \parallel Q'))$ $(A_P @ A_Q)$ $(\Psi_P \otimes \Psi_Q)$

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 Ψ_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 Ψ_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 Ψ $(P \parallel Q)$ $(\tau \prec (\!|\nu* xvec|\!)(P' \parallel Q'))$ $(A_P @ A_Q)$ $(\Psi_P \otimes \Psi_Q)$

and *rBrMerge*: $\bigwedge \Psi\ \Psi_Q\ P\ M\ N\ P'\ A_P\ \Psi_P\ Q\ Q'\ A_Q\ C.$
⟦$\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$; $\bigwedge C.$ *Prop* C $(\Psi \otimes \Psi_Q)$ P $(¿M(\!|N|\!) \prec P')$ A_P Ψ_P; *extractFrame* $P = \langle A_P, \Psi_P\rangle$; *distinct* A_P; $\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$; $\bigwedge C.$ *Prop* C $(\Psi \otimes \Psi_P)$ Q $(¿M(\!|N|\!) \prec Q')$ A_Q Ψ_Q; *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$; *distinct* A_Q; $A_P \sharp* \Psi$; $A_P \sharp* \Psi_Q$; $A_P \sharp* P$; $A_P \sharp* N$; $A_P \sharp* P'$; $A_P \sharp* Q$; $A_P \sharp* Q'$; $A_P \sharp* A_Q$; $A_P \sharp* M$; $A_Q \sharp* M$; $A_Q \sharp* \Psi$; $A_Q \sharp* \Psi_P$; $A_Q \sharp* P$; $A_Q \sharp* N$; $A_Q \sharp* P'$; $A_Q \sharp* Q$; $A_Q \sharp* Q'$; $A_P \sharp* C$; $A_Q \sharp* C$; $A_P \sharp* M$; $A_Q \sharp* M$⟧ ⟹
Prop C Ψ $(P \parallel Q)$ $(¿M(\!|N|\!) \prec (P' \parallel Q'))$ $(A_P @ A_Q)$ $(\Psi_P \otimes \Psi_Q)$

and *rBrComm1*:$\bigwedge \Psi\ \Psi_Q\ P\ M\ N\ P'\ A_P\ \Psi_P\ Q\ xvec\ Q'\ A_Q\ C.$
⟦$\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$; *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 Ψ_P; $\Psi \otimes \Psi_P \rhd Q \longmapsto ¡M(\!|\nu* xvec|\!)\langle N\rangle \prec Q'$; *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$; *distinct* A_Q; $\bigwedge C.$ *Prop* C $(\Psi \otimes \Psi_P)$ Q $(¡M(\!|\nu* xvec|\!)\langle N\rangle \prec Q')$ A_Q Ψ_Q; *distinct* $xvec$; $A_P \sharp* \Psi$; $A_P \sharp* \Psi_Q$; $A_P \sharp* P$; $A_P \sharp* N$; $A_P \sharp* P'$; $A_P \sharp* Q$; $A_P \sharp* Q'$; $A_P \sharp* A_Q$; $A_P \sharp* xvec$; $A_Q \sharp* \Psi$; $A_Q \sharp* \Psi_P$; $A_Q \sharp* P$; $A_Q \sharp* N$; $A_Q \sharp* P'$; $A_Q \sharp* Q$; $A_Q \sharp* Q'$; $A_Q \sharp* xvec$; $xvec \sharp* \Psi$; $xvec \sharp* \Psi_P$; $xvec \sharp* \Psi_Q$; $xvec \sharp* P$; $xvec \sharp* Q$; $A_P \sharp* C$; $A_Q \sharp* C$; $xvec \sharp* C$; $A_P \sharp* M$; $A_Q \sharp* M$; $xvec \sharp* M$⟧ ⟹
Prop C Ψ $(P \parallel Q)$ $(¡M(\!|\nu* xvec|\!)\langle N\rangle \prec (P' \parallel Q'))$ $(A_P @ A_Q)$ $(\Psi_P$

$\otimes\ \Psi_Q)$
and *rBrComm2*:$\bigwedge \Psi\ \Psi_Q\ P\ M\ xvec\ N\ P'\ A_P\ \Psi_P\ Q\ Q'\ A_Q\ C.$
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto ¡M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P';\ extractFrame\ P = \langle A_P,$
$\Psi_P\rangle;\ distinct\ A_P;$
$\bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_Q)\ P\ (¡M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P')\ A_P\ \Psi_P;$
$\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q';\ extractFrame\ Q = \langle A_Q,\ \Psi_Q\rangle;$
$distinct\ A_Q;$
$\bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_P)\ Q\ (¿M(\!|N|\!) \prec Q')\ A_Q\ \Psi_Q;\ distinct\ xvec;$
$A_P \sharp{*}\ \Psi;\ A_P \sharp{*}\ \Psi_Q;\ A_P \sharp{*}\ P;\ A_P \sharp{*}\ N;\ A_P \sharp{*}\ P';$
$A_P \sharp{*}\ Q;\ A_P \sharp{*}\ Q';\ A_P \sharp{*}\ A_Q;\ A_P \sharp{*}\ xvec;\ A_Q \sharp{*}\ \Psi;\ A_Q \sharp{*}\ \Psi_P;$
$A_Q \sharp{*}\ P;\ A_Q \sharp{*}\ N;\ A_Q \sharp{*}\ P';\ A_Q \sharp{*}\ Q;\ A_Q \sharp{*}\ Q';$
$A_Q \sharp{*}\ xvec;\ xvec \sharp{*}\ \Psi;\ xvec \sharp{*}\ \Psi_P;\ xvec \sharp{*}\ \Psi_Q;\ xvec \sharp{*}\ P;$
$xvec \sharp{*}\ Q;\ A_P \sharp{*}\ C;\ A_Q \sharp{*}\ C;\ xvec \sharp{*}\ C;$
$A_P \sharp{*}\ M;\ A_Q \sharp{*}\ M;\ xvec \sharp{*}\ M]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (P \parallel Q)\ (¡M(\!|\nu{*}xvec|\!)\langle N\rangle \prec (P' \parallel Q'))\ (A_P@A_Q)\ (\Psi_P$
$\otimes\ \Psi_Q)$
and *rBrClose*: $\bigwedge \Psi\ P\ M\ xvec\ N\ P'\ A_P\ \Psi_P\ x\ C.$
$[\![\Psi \rhd P \longmapsto ¡M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P';$
$x \in supp\ M;$
$\bigwedge C.\ Prop\ C\ \Psi\ P\ (¡M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P')\ A_P\ \Psi_P;$
$extractFrame\ P = \langle A_P,\ \Psi_P\rangle;\ distinct\ A_P;$
$A_P \sharp{*}\ \Psi;\ A_P \sharp{*}\ P;\ A_P \sharp{*}\ M;\ A_P \sharp{*}\ N;\ A_P \sharp{*}\ P';\ A_P \sharp{*}\ xvec;$
$distinct\ xvec;\ xvec \sharp{*}\ \Psi;\ xvec \sharp{*}\ \Psi_P;\ xvec \sharp{*}\ P;$
$xvec \sharp{*}\ M;$
$x \sharp \Psi;\ x \sharp xvec;\ x \sharp A_P;$
$A_P \sharp{*}\ C;\ xvec \sharp{*}\ C;\ x \sharp C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ ((\!|\nu x|\!)P)\ (\tau \prec ((\!|\nu x|\!)((\!|\nu{*}xvec|\!)P')))\ (x\#A_P)\ \Psi_P$
and *rOpen*: $\bigwedge \Psi\ P\ M\ xvec\ yvec\ N\ P'\ x\ A_P\ \Psi_P\ C.$
$[\![\Psi \rhd P \longmapsto M(\!|\nu{*}(xvec@yvec)|\!)\langle N\rangle \prec P';\ extractFrame\ P = \langle A_P,$
$\Psi_P\rangle;\ distinct\ A_P;$
$\bigwedge C.\ Prop\ C\ \Psi\ P\ (M(\!|\nu{*}(xvec@yvec)|\!)\langle N\rangle \prec P')\ A_P\ \Psi_P;\ x \in supp$
$N;\ x \sharp \Psi;\ x \sharp M;$
$x \sharp A_P;\ x \sharp xvec;\ x \sharp yvec;\ A_P \sharp{*}\ \Psi;\ A_P \sharp{*}\ P;\ A_P \sharp{*}\ M;\ A_P \sharp{*}$
$N;\ A_P \sharp{*}\ P';$
$A_P \sharp{*}\ xvec;\ A_P \sharp{*}\ yvec;\ xvec \sharp{*}\ yvec;\ distinct\ xvec;\ distinct\ yvec;$
$xvec \sharp{*}\ \Psi;\ xvec \sharp{*}\ P;\ xvec \sharp{*}\ M;\ xvec \sharp{*}\ \Psi_P;\ yvec \sharp{*}\ \Psi_P;$
$yvec \sharp{*}\ \Psi;\ yvec \sharp{*}\ P;\ yvec \sharp{*}\ M;\ A_P \sharp{*}\ C;\ x \sharp C;\ xvec \sharp{*}\ C;\ yvec$
$\sharp{*}\ C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ ((\!|\nu x|\!)P)\ (M(\!|\nu{*}(xvec@x\#yvec)|\!)\langle N\rangle \prec P')\ (x\#A_P)\ \Psi_P$
and *rBrOpen*: $\bigwedge \Psi\ P\ M\ xvec\ yvec\ N\ P'\ x\ A_P\ \Psi_P\ C.$
$[\![\Psi \rhd P \longmapsto ¡M(\!|\nu{*}(xvec@yvec)|\!)\langle N\rangle \prec P';\ extractFrame\ P = \langle A_P,$
$\Psi_P\rangle;\ distinct\ A_P;$
$\bigwedge C.\ Prop\ C\ \Psi\ P\ (¡M(\!|\nu{*}(xvec@yvec)|\!)\langle N\rangle \prec P')\ A_P\ \Psi_P;\ x \in$
$supp\ N;\ x \sharp \Psi;\ x \sharp M;$
$x \sharp A_P;\ x \sharp xvec;\ x \sharp yvec;\ A_P \sharp{*}\ \Psi;\ A_P \sharp{*}\ P;\ A_P \sharp{*}\ M;\ A_P \sharp{*}$
$N;\ A_P \sharp{*}\ P';$
$A_P \sharp{*}\ xvec;\ A_P \sharp{*}\ yvec;\ xvec \sharp{*}\ yvec;\ distinct\ xvec;\ distinct\ yvec;$
$xvec \sharp{*}\ \Psi;\ xvec \sharp{*}\ P;\ xvec \sharp{*}\ M;\ xvec \sharp{*}\ \Psi_P;\ yvec \sharp{*}\ \Psi_P;$
$yvec \sharp{*}\ \Psi;\ yvec \sharp{*}\ P;\ yvec \sharp{*}\ M;\ A_P \sharp{*}\ C;\ x \sharp C;\ xvec \sharp{*}\ C;\ yvec$

$\sharp*\ C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ ((\!|\nu x|\!)P)\ (¡M(\!|\nu*(xvec@x\#yvec)|\!)\langle N\rangle \prec P')\ (x\#A_P)\ \Psi_P$
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*$(bn\ \alpha)$;
$bn\ \alpha \sharp* \Psi$; $bn\ \alpha \sharp* P$; $bn\ \alpha \sharp* subject\ \alpha$; $bn\ \alpha \sharp* \Psi_P$;
$A_P \sharp* C$; $x \sharp C$; $bn\ \alpha \sharp* C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ ((\!|\nu x|\!)P)\ (\alpha \prec ((\!|\nu x|\!)P'))\ (x\#A_P)\ \Psi_P$
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* Ψ_P = ({}::*name set*);
$A_P \sharp* \Psi$; $A_P \sharp* P$; $A_P \sharp* Rs$; $A_P \sharp* C]\!] \Longrightarrow Prop\ C\ \Psi\ (!P)\ Rs$ ([]) (**1**)
shows $Prop\ C\ \Psi\ P\ Rs\ A_P\ \Psi_P$
using *Trans FrP* ‹*distinct* A_P›
proof(*nominal-induct avoiding*: $A_P\ \Psi_P\ C$ *rule*: *semantics.strong-induct*)
case(*cInput* $\Psi\ M\ K\ xvec\ N\ Tvec\ P\ A_P\ \Psi_P\ C$)
from ‹*extractFrame* $(M(\!|\lambda*xvec\ N|\!).P) = \langle A_P, \Psi_P\rangle$›
have $A_P = []$ **and** $\Psi_P = \mathbf{1}$
by *auto*
with ‹$\Psi \vdash M \leftrightarrow K$› ‹*distinct xvec*› ‹*set xvec* $\subseteq$ *supp* N› ‹*length xvec* = *length Tvec*›
‹*xvec* $\sharp*\ \Psi$› ‹*xvec* $\sharp*\ M$› ‹*xvec* $\sharp*\ K$› ‹*xvec* $\sharp*\ C$›
show *?case* **by**(*blast intro*: *rInput*)
next
case(*cBrInput* $\Psi\ K\ M\ xvec\ N\ Tvec\ P\ A_P\ \Psi_P\ C$)
from ‹*extractFrame* $(M(\!|\lambda*xvec\ N|\!).P) = \langle A_P, \Psi_P\rangle$›
have $A_P = []$ **and** $\Psi_P = \mathbf{1}$
by *auto*
with ‹$\Psi \vdash K \succeq M$› ‹*distinct xvec*› ‹*set xvec* $\subseteq$ *supp* N› ‹*length xvec* = *length Tvec*›
‹*xvec* $\sharp*\ \Psi$› ‹*xvec* $\sharp*\ M$› ‹*xvec* $\sharp*\ K$› ‹*xvec* $\sharp*\ C$›
show *?case* **by**(*blast intro*: *rBrInput*)
next
case(*Output* $\Psi\ M\ K\ N\ P\ A_P\ \Psi_P$)
from ‹*extractFrame* $(M\langle N\rangle.P) = \langle A_P, \Psi_P\rangle$›
have $A_P = []$ **and** $\Psi_P = \mathbf{1}$
by *auto*
with ‹$\Psi \vdash M \leftrightarrow K$› **show** *?case*
by(*blast intro*: *rOutput*)
next
case(*BrOutput* $\Psi\ M\ K\ N\ P\ A_P\ \Psi_P$)
from ‹*extractFrame* $(M\langle N\rangle.P) = \langle A_P, \Psi_P\rangle$›
have $A_P = []$ **and** $\Psi_P = \mathbf{1}$
by *auto*

with ‹$\Psi \vdash M \preceq K$› **show** *?case*
by(*blast intro*: *rBrOutput*)
next
case(*Case* Ψ P *Rs* φ *Cs* A_{cP} Ψ_{cP} C)
obtain A_P Ψ_P **where** *FrP*: *extractFrame* $P = \langle A_P, \Psi_P\rangle$ **and** *distinct* A_P
and $A_P \sharp* (\Psi, P, Rs, C)$
by(*rule freshFrame*)
then have $A_P \sharp* \Psi$ **and** $A_P \sharp* P$ **and** $A_P \sharp* Rs$ **and** $A_P \sharp* C$
by *simp*+
note ‹$\Psi \rhd P \longmapsto Rs$› *FrP* ‹*distinct* A_P›
moreover from *FrP* ‹*distinct* A_P› ‹$\bigwedge A_P\ \Psi_P\ C.\ [\![$*extractFrame* $P = \langle A_P, \Psi_P\rangle$; *distinct* $A_P]\!] \Longrightarrow$ *Prop* C Ψ P *Rs* A_P Ψ_P›
have $\bigwedge C.$ *Prop* C Ψ P *Rs* A_P Ψ_P **by** *simp*
moreover note ‹$(\varphi, P) \in$ *set Cs*› ‹$\Psi \vdash \varphi$› ‹*guarded* P›
moreover from ‹*guarded* P› *FrP* **have** $\Psi_P \simeq \mathbf{1}$ **and** *supp* $\Psi_P = (\{\}::$*name set*$)$
by(*metis guardedStatEq*)+
moreover note ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_P \sharp* Rs$› ‹$A_P \sharp* C$›
ultimately have *Prop* C Ψ (*Cases Cs*) *Rs* ([]) (**1**)
by(*rule rCase*)
then show *?case* **using** ‹*extractFrame*(*Cases Cs*) $= \langle A_{cP}, \Psi_{cP}\rangle$› **by** *simp*
next
case(*cPar1* Ψ Ψ_Q P α P' Q A_Q A_{PQ} Ψ_{PQ} C)
obtain A_P Ψ_P **where** *FrP*: *extractFrame* $P = \langle A_P, \Psi_P\rangle$ **and** *distinct* A_P
$A_P \sharp* (P, Q, \Psi, \alpha, P', A_Q, A_{PQ}, C, \Psi_Q)$
by(*rule freshFrame*)
then have $A_P \sharp* P$ **and** $A_P \sharp* Q$ **and** $A_P \sharp* \Psi$ **and** $A_P \sharp* \alpha$ **and** $A_P \sharp* P'$
and $A_P \sharp* A_Q$ **and** $A_P \sharp* A_{PQ}$ **and** $A_P \sharp* C$ **and** $A_P \sharp* \Psi_Q$
by *simp*+

have *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$ **by** *fact*

from ‹$A_Q \sharp* P$› ‹$A_P \sharp* A_Q$› *FrP* **have** $A_Q \sharp* \Psi_P$
by(*force dest*: *extractFrameFreshChain*)

from ‹*bn* $\alpha \sharp* P$› ‹$A_P \sharp* \alpha$› *FrP* **have** *bn* $\alpha \sharp* \Psi_P$
by(*force dest*: *extractFrameFreshChain*)

from ‹*extractFrame*$(P \parallel Q) = \langle A_{PQ}, \Psi_{PQ}\rangle$› *FrP FrQ* ‹$A_P \sharp* A_Q$› ‹$A_P \sharp* \Psi_Q$›
‹$A_Q \sharp* \Psi_P$›
have $\langle (A_P @ A_Q), \Psi_P \otimes \Psi_Q\rangle = \langle A_{PQ}, \Psi_{PQ}\rangle$
by *simp*
moreover from ‹*distinct* A_P› ‹*distinct* A_Q› ‹$A_P \sharp* A_Q$› **have** *distinct*$(A_P @ A_Q)$
by(*auto simp add*: *fresh-star-def fresh-def name-list-supp*)
ultimately obtain p **where** S: *set* $p \subseteq$ *set*$(A_P @ A_Q) \times$ *set*$((p \cdot A_P) @ (p \cdot A_Q))$
and *distinctPerm* p
and Ψeq: $\Psi_{PQ} = p \cdot (\Psi_P \otimes \Psi_Q)$ **and** *Aeq*: $A_{PQ} = (p \cdot A_P) @ (p \cdot A_Q)$
using ‹$A_P \sharp* A_{PQ}$› ‹$A_Q \sharp* A_{PQ}$› ‹*distinct* A_{PQ}›
apply −
apply(*rule frameChainEq'*)

by (*assumption* | *simp add*: *eqvts*)+
note ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto \alpha \prec P'$› *FrP* ‹*distinct* A_P› *FrQ* ‹*distinct* A_Q›

moreover from *FrP* ‹*distinct* A_P› ‹$\bigwedge A_P\ \Psi_P\ C.\ [\![\mathit{extractFrame}\ P = \langle A_P, \Psi_P\rangle; \mathit{distinct}\ A_P]\!] \Longrightarrow \mathit{Prop}\ C\ (\Psi \otimes \Psi_Q)\ P\ (\alpha \prec P')\ A_P\ \Psi_P$›
have $\bigwedge C.\ \mathit{Prop}\ C\ (\Psi \otimes \Psi_Q)\ P\ (\alpha \prec P')\ A_P\ \Psi_P$ **by** *simp*

moreover note ‹$A_P \sharp* P$› ‹$A_P \sharp* Q$› ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* \alpha$› ‹$A_P \sharp* P'$› ‹$A_P \sharp* A_Q$› ‹$A_P \sharp* \Psi_Q$›
‹$A_Q \sharp* P$› ‹$A_Q \sharp* Q$› ‹$A_Q \sharp* \Psi$› ‹$A_Q \sharp* \alpha$› ‹$A_Q \sharp* P'$› ‹$A_Q \sharp* \Psi_P$› ‹*distinct*(*bn* α)›
‹*bn* $\alpha \sharp* \Psi$› ‹*bn* $\alpha \sharp* P$› ‹*bn* $\alpha \sharp* Q$› ‹*bn* $\alpha \sharp*$ *subject* α› ‹*bn* $\alpha \sharp* \Psi_P$› ‹*bn* $\alpha \sharp* \Psi_Q$›
‹$A_P \sharp* C$› ‹$A_Q \sharp* C$› ‹*bn* $\alpha \sharp* C$›
ultimately have *Prop* $C\ \Psi\ (P \parallel Q)\ (\alpha \prec (P' \parallel Q))\ (A_P @ A_Q)\ (\Psi_P \otimes \Psi_Q)$
by(*metis rPar1*)
with ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_P \sharp* Q$› ‹$A_P \sharp* \alpha$› ‹$A_P \sharp* P'$› ‹$A_P \sharp* A_{PQ}$› ‹$A_P \sharp* C$›
‹$A_Q \sharp* \Psi$› ‹$A_Q \sharp* P$› ‹$A_Q \sharp* Q$› ‹$A_Q \sharp* \alpha$› ‹$A_Q \sharp* P'$› ‹$A_Q \sharp* A_{PQ}$› ‹$A_Q \sharp* C$›
S ‹*distinctPerm p*› *Aeq*
have *Prop* $C\ \Psi\ (P \parallel Q)\ (\alpha \prec (P' \parallel Q))\ (p \cdot (A_P @ A_Q))\ (p \cdot (\Psi_P \otimes \Psi_Q))$
apply −
apply(*rule rAlpha*)
by(*assumption* | *simp add*: *eqvts*)+
with Ψ*eq Aeq* **show** *?case* **by**(*simp add*: *eqvts*)
next
case(*cPar2* $\Psi\ \Psi_P\ Q\ \alpha\ Q'\ P\ A_P\ A_{PQ}\ \Psi_{PQ}\ C$)
obtain $A_Q\ \Psi_Q$ **where** *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$ **and** *distinct* A_Q
$A_Q \sharp* (P, Q, \Psi, \alpha, Q', A_P, A_{PQ}, C, \Psi_P)$
by(*rule freshFrame*)
then have $A_Q \sharp* P$ **and** $A_Q \sharp* Q$ **and** $A_Q \sharp* \Psi$ **and** $A_Q \sharp* \alpha$ **and** $A_Q \sharp* Q'$ **and** $A_Q \sharp* A_P$ **and** $A_Q \sharp* A_{PQ}$ **and** $A_Q \sharp* C$ **and** $A_Q \sharp* \Psi_P$
by *simp*+

from ‹$A_Q \sharp* A_P$› **have** $A_P \sharp* A_Q$ **by** *simp*
have *FrP*: *extractFrame* $P = \langle A_P, \Psi_P\rangle$ **by** *fact*

from ‹$A_P \sharp* Q$› ‹$A_Q \sharp* A_P$› *FrQ* **have** $A_P \sharp* \Psi_Q$
by(*force dest*: *extractFrameFreshChain*)
from ‹*bn* $\alpha \sharp* Q$› ‹$A_Q \sharp* \alpha$› *FrQ* **have** *bn* $\alpha \sharp* \Psi_Q$
by(*force dest*: *extractFrameFreshChain*)

from ‹*extractFrame*$(P \parallel Q) = \langle A_{PQ}, \Psi_{PQ}\rangle$› *FrP FrQ* ‹$A_P \sharp* A_Q$› ‹$A_P \sharp* \Psi_Q$› ‹$A_Q \sharp* \Psi_P$›
have $\langle (A_P @ A_Q), \Psi_P \otimes \Psi_Q\rangle = \langle A_{PQ}, \Psi_{PQ}\rangle$
by *simp*
moreover from ‹*distinct* A_P› ‹*distinct* A_Q› ‹$A_P \sharp* A_Q$› **have** *distinct*$(A_P @ A_Q)$
by(*auto simp add*: *fresh-star-def fresh-def name-list-supp*)

ultimately obtain p **where** S: $(set\ p \subseteq (set(A_P@A_Q)) \times (set\ A_{PQ}))$ **and** *distinctPerm p*
and Ψeq: $\Psi_{PQ} = p \cdot (\Psi_P \otimes \Psi_Q)$ **and** Aeq: $A_{PQ} = ((p \cdot A_P)@(p \cdot A_Q))$
using $\langle A_P \sharp* A_{PQ}\rangle$ $\langle A_Q \sharp* A_{PQ}\rangle$ $\langle distinct\ A_{PQ}\rangle$
apply $-$
apply(*rule frameChainEq'*)
by(*assumption* | *simp add*: *eqvts*)+

note $\langle \Psi \otimes \Psi_P \rhd Q \longmapsto \alpha \prec Q'\rangle$ *FrP* $\langle distinct\ A_P\rangle$ *FrQ* $\langle distinct\ A_Q\rangle$

moreover from *FrQ* $\langle distinct\ A_Q\rangle$ $\langle \bigwedge A_Q\ \Psi_Q\ C.\ [\![extractFrame\ Q = \langle A_Q, \Psi_Q\rangle; distinct\ A_Q]\!] \Longrightarrow Prop\ C\ (\Psi \otimes \Psi_P)\ Q\ (\alpha \prec Q')\ A_Q\ \Psi_Q\rangle$
have $\bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_P)\ Q\ (\alpha \prec Q')\ A_Q\ \Psi_Q$ **by** *simp*

moreover note $\langle A_P \sharp* P\rangle$ $\langle A_P \sharp* Q\rangle$ $\langle A_P \sharp* \Psi\rangle$ $\langle A_P \sharp* \alpha\rangle$ $\langle A_P \sharp* Q'\rangle$ $\langle A_P \sharp* A_Q\rangle$ $\langle A_P \sharp* \Psi_Q\rangle$
$\langle A_Q \sharp* P\rangle$ $\langle A_Q \sharp* Q\rangle$ $\langle A_Q \sharp* \Psi\rangle$ $\langle A_Q \sharp* \alpha\rangle$ $\langle A_Q \sharp* Q'\rangle$ $\langle A_Q \sharp* \Psi_P\rangle$ $\langle distinct(bn\ \alpha)\rangle$
$\langle bn\ \alpha \sharp* \Psi\rangle$ $\langle bn\ \alpha \sharp* P\rangle$ $\langle bn\ \alpha \sharp* Q\rangle$ $\langle bn\ \alpha \sharp* subject\ \alpha\rangle$ $\langle bn\ \alpha \sharp* \Psi_P\rangle$ $\langle bn\ \alpha \sharp* \Psi_Q\rangle$
$\langle A_P \sharp* C\rangle$ $\langle A_Q \sharp* C\rangle$ $\langle bn\ \alpha \sharp* C\rangle$
ultimately have $Prop\ C\ \Psi\ (P \parallel Q)\ (\alpha \prec (P \parallel Q'))\ (A_P@A_Q)\ (\Psi_P \otimes \Psi_Q)$
by(*metis rPar2*)

with $\langle A_P \sharp* \Psi\rangle$ $\langle A_P \sharp* P\rangle$ $\langle A_P \sharp* Q\rangle$ $\langle A_P \sharp* \alpha\rangle$ $\langle A_P \sharp* Q'\rangle$ $\langle A_P \sharp* A_{PQ}\rangle$ $\langle A_P \sharp* C\rangle$
$\langle A_Q \sharp* \Psi\rangle$ $\langle A_Q \sharp* P\rangle$ $\langle A_Q \sharp* Q\rangle$ $\langle A_Q \sharp* \alpha\rangle$ $\langle A_Q \sharp* Q'\rangle$ $\langle A_Q \sharp* A_{PQ}\rangle$ $\langle A_Q \sharp* C\rangle$
S $\langle distinctPerm\ p\rangle$ Aeq
have $Prop\ C\ \Psi\ (P \parallel Q)\ (\alpha \prec (P \parallel Q'))\ (p \cdot (A_P@A_Q))\ (p \cdot (\Psi_P \otimes \Psi_Q))$
apply $-$
apply(*rule rAlpha*)
by(*assumption* | *simp add*: *eqvts*)+
with Ψeq Aeq **show** *?case* **by**(*simp add*: *eqvts*)
next
case(*cComm1* Ψ Ψ_Q P M N P' A_P Ψ_P Q K *xvec* Q' A_Q A_{PQ} Ψ_{PQ} C)
from $\langle distinct\ A_P\rangle$ $\langle distinct\ A_Q\rangle$ $\langle A_P \sharp* A_Q\rangle$ **have** $distinct(A_P@A_Q)$
by(*auto simp add*: *fresh-star-def fresh-def name-list-supp*)
from *cComm1* **have** $Prop\ C\ \Psi\ (P \parallel Q)\ (\tau \prec (\!|\nu*xvec|\!)(P' \parallel Q'))\ (A_P@A_Q)\ (\Psi_P \otimes \Psi_Q)$
by(*metis rComm1*)
moreover from $\langle extractFrame(P \parallel Q) = \langle A_{PQ}, \Psi_{PQ}\rangle\rangle$ $\langle extractFrame\ P = \langle A_P, \Psi_P\rangle\rangle$ $\langle extractFrame\ Q = \langle A_Q, \Psi_Q\rangle\rangle$
$\langle A_P \sharp* A_Q\rangle$ $\langle A_P \sharp* \Psi_Q\rangle$ $\langle A_Q \sharp* \Psi_P\rangle$
have $\langle (A_P@A_Q), (\Psi_P \otimes \Psi_Q)\rangle = \langle A_{PQ}, \Psi_{PQ}\rangle$
by *simp*
with $\langle A_P \sharp* A_{PQ}\rangle$ $\langle A_Q \sharp* A_{PQ}\rangle$ $\langle distinct(A_P@A_Q)\rangle$ $\langle distinct\ A_{PQ}\rangle$
obtain p **where** S: $(set\ p \subseteq (set(A_P@A_Q)) \times (set\ A_{PQ}))$ **and** *distinctPerm p*
and Ψeq: $\Psi_{PQ} = p \cdot (\Psi_P \otimes \Psi_Q)$ **and** Aeq: $A_{PQ} = p \cdot (A_P@A_Q)$

apply −
apply(*rule frameChainEq′*)
by(*assumption* | *simp*)+
moreover note ‹$A_P \sharp* \Psi$› ‹$A_Q \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_Q \sharp* P$› ‹$A_P \sharp* Q$› ‹$A_Q \sharp* Q$› ‹$A_P \sharp* xvec$›
‹$A_Q \sharp* xvec$› ‹$A_P \sharp* P'$› ‹$A_Q \sharp* P'$› ‹$A_P \sharp* Q'$› ‹$A_Q \sharp* Q'$› ‹$A_P \sharp* A_{PQ}$› ‹$A_Q \sharp* A_{PQ}$›
‹$A_P \sharp* C$› ‹$A_Q \sharp* C$›
ultimately have *Prop* $C\ \Psi\ (P \parallel Q)\ (\tau \prec (\!|\nu * xvec|\!)(P' \parallel Q'))\ (p \cdot (A_P @ A_Q))\ (p \cdot (\Psi_P \otimes \Psi_Q))$
by(*fastforce simp add*: *rAlpha*)
with $\Psi eq\ Aeq$ **show** *?case* **by** *simp*
next
case(*cComm2* $\Psi\ \Psi_Q\ P\ M\ xvec\ N\ P'\ A_P\ \Psi_P\ Q\ K\ Q'\ A_Q\ A_{PQ}\ \Psi_{PQ}\ C$)
from ‹*distinct* A_P› ‹*distinct* A_Q› ‹$A_P \sharp* A_Q$› **have** *distinct*$(A_P @ A_Q)$
by(*auto simp add*: *fresh-star-def fresh-def name-list-supp*)
from *cComm2* **have** *Prop* $C\ \Psi\ (P \parallel Q)\ (\tau \prec (\!|\nu * xvec|\!)(P' \parallel Q'))\ (A_P @ A_Q)\ (\Psi_P \otimes \Psi_Q)$
by(*metis rComm2*)
moreover from ‹*extractFrame*$(P \parallel Q) = \langle A_{PQ}, \Psi_{PQ}\rangle$› ‹*extractFrame* $P = \langle A_P, \Psi_P\rangle$› ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$›
‹$A_P \sharp* A_Q$› ‹$A_P \sharp* \Psi_Q$› ‹$A_Q \sharp* \Psi_P$›
have $\langle (A_P @ A_Q), (\Psi_P \otimes \Psi_Q)\rangle = \langle A_{PQ}, \Psi_{PQ}\rangle$
by *simp*
with ‹$A_P \sharp* A_{PQ}$› ‹$A_Q \sharp* A_{PQ}$› ‹*distinct*$(A_P @ A_Q)$› ‹*distinct* A_{PQ}›
obtain p **where** S: $(set\ p \subseteq (set(A_P @ A_Q)) \times (set\ A_{PQ}))$ **and** *distinctPerm* p
and Ψeq: $\Psi_{PQ} = p \cdot (\Psi_P \otimes \Psi_Q)$ **and** Aeq: $A_{PQ} = p \cdot (A_P @ A_Q)$
apply −
apply(*rule frameChainEq′*)
by(*assumption* | *simp*)+
moreover note ‹$A_P \sharp* \Psi$› ‹$A_Q \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_Q \sharp* P$› ‹$A_P \sharp* Q$› ‹$A_Q \sharp* Q$› ‹$A_P \sharp* xvec$›
‹$A_Q \sharp* xvec$› ‹$A_P \sharp* P'$› ‹$A_Q \sharp* P'$› ‹$A_P \sharp* Q'$› ‹$A_Q \sharp* Q'$› ‹$A_P \sharp* A_{PQ}$› ‹$A_Q \sharp* A_{PQ}$›
‹$A_P \sharp* C$› ‹$A_Q \sharp* C$›
ultimately have *Prop* $C\ \Psi\ (P \parallel Q)\ (\tau \prec (\!|\nu * xvec|\!)(P' \parallel Q'))\ (p \cdot (A_P @ A_Q))\ (p \cdot (\Psi_P \otimes \Psi_Q))$
by(*fastforce intro*: *rAlpha*)
with $\Psi eq\ Aeq$ **show** *?case* **by** *simp*
next
case(*cBrMerge* $\Psi\ \Psi_Q\ P\ M\ N\ P'\ A_P\ \Psi_P\ Q\ Q'\ A_Q\ A_{PQ}\ \Psi_{PQ}\ C$)
from ‹*distinct* A_P› ‹*distinct* A_Q› ‹$A_P \sharp* A_Q$› **have** *distinct*$(A_P @ A_Q)$
by(*auto simp add*: *fresh-star-def fresh-def name-list-supp*)
from *cBrMerge* **have** *Prop* $C\ \Psi\ (P \parallel Q)\ (¿M(\!|N|\!) \prec (P' \parallel Q'))\ (A_P @ A_Q)\ (\Psi_P \otimes \Psi_Q)$
by(*fastforce intro*!: *rBrMerge*)
moreover from ‹*extractFrame*$(P \parallel Q) = \langle A_{PQ}, \Psi_{PQ}\rangle$› ‹*extractFrame* $P = \langle A_P, \Psi_P\rangle$› ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$›
‹$A_P \sharp* A_Q$› ‹$A_P \sharp* \Psi_Q$› ‹$A_Q \sharp* \Psi_P$›

have $\langle(A_P@A_Q), (\Psi_P \otimes \Psi_Q)\rangle = \langle A_{PQ}, \Psi_{PQ}\rangle$
by *simp*
with ‹$A_P \sharp* A_{PQ}$› ‹$A_Q \sharp* A_{PQ}$› ‹*distinct*$(A_P@A_Q)$› ‹*distinct* A_{PQ}›
obtain p **where** S: (*set* $p \subseteq$ (*set*$(A_P@A_Q)) \times$ (*set* A_{PQ})) **and** *distinctPerm* p
and Ψeq: $\Psi_{PQ} = p \cdot (\Psi_P \otimes \Psi_Q)$ **and** Aeq: $A_{PQ} = p \cdot (A_P@A_Q)$
apply −
apply(*rule frameChainEq′*)
by(*assumption* | *simp*)+
moreover note ‹$A_P \sharp* \Psi$› ‹$A_Q \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_Q \sharp* P$› ‹$A_P \sharp* Q$› ‹$A_Q \sharp* Q$›
‹$A_P \sharp* P'$› ‹$A_Q \sharp* P'$› ‹$A_P \sharp* Q'$› ‹$A_Q \sharp* Q'$› ‹$A_P \sharp* A_{PQ}$› ‹$A_Q \sharp* A_{PQ}$›
‹$A_P \sharp* C$› ‹$A_Q \sharp* C$› ‹$A_P \sharp* M$› ‹$A_Q \sharp* M$› ‹$A_P \sharp* N$› ‹$A_Q \sharp* N$›
ultimately have *Prop* $C\ \Psi\ (P \parallel Q)$ ($¿M⦇N⦈ \prec (P' \parallel Q')$) $(p \cdot (A_P@A_Q))$ $(p \cdot (\Psi_P \otimes \Psi_Q))$
by(*fastforce intro*: *rAlpha*)
with $\Psi eq\ Aeq$ **show** *?case* **by** *simp*
next
case(*cBrComm1* $\Psi\ \Psi_Q\ P\ M\ N\ P'\ A_P\ \Psi_P\ Q\ xvec\ Q'\ A_Q\ A_{PQ}\ \Psi_{PQ}\ C$)
from ‹*distinct* A_P› ‹*distinct* A_Q› ‹$A_P \sharp* A_Q$› **have** *distinct*$(A_P@A_Q)$
by(*auto simp add*: *fresh-star-def fresh-def name-list-supp*)
from *cBrComm1* **have** *Prop* $C\ \Psi\ (P \parallel Q)$ ($¡M⦇\nu*xvec⦈\langle N\rangle \prec (P' \parallel Q')$) $(A_P@A_Q)$ $(\Psi_P \otimes \Psi_Q)$
by(*metis rBrComm1*)
moreover from ‹*extractFrame*$(P \parallel Q) = \langle A_{PQ}, \Psi_{PQ}\rangle$› ‹*extractFrame* $P = \langle A_P, \Psi_P\rangle$› ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$›
‹$A_P \sharp* A_Q$› ‹$A_P \sharp* \Psi_Q$› ‹$A_Q \sharp* \Psi_P$›
have $\langle(A_P@A_Q), (\Psi_P \otimes \Psi_Q)\rangle = \langle A_{PQ}, \Psi_{PQ}\rangle$
by *simp*
with ‹$A_P \sharp* A_{PQ}$› ‹$A_Q \sharp* A_{PQ}$› ‹*distinct*$(A_P@A_Q)$› ‹*distinct* A_{PQ}›
obtain p **where** S: (*set* $p \subseteq$ (*set*$(A_P@A_Q)) \times$ (*set* A_{PQ})) **and** *distinctPerm* p
and Ψeq: $\Psi_{PQ} = p \cdot (\Psi_P \otimes \Psi_Q)$ **and** Aeq: $A_{PQ} = p \cdot (A_P@A_Q)$
apply −
apply(*rule frameChainEq′*)
by(*assumption* | *simp*)+
moreover note ‹$A_P \sharp* \Psi$› ‹$A_Q \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_Q \sharp* P$› ‹$A_P \sharp* Q$› ‹$A_Q \sharp* Q$› ‹$A_P \sharp* xvec$›
‹$A_Q \sharp* xvec$› ‹$A_P \sharp* P'$› ‹$A_Q \sharp* P'$› ‹$A_P \sharp* Q'$› ‹$A_Q \sharp* Q'$› ‹$A_P \sharp* A_{PQ}$› ‹$A_Q \sharp* A_{PQ}$›
‹$A_P \sharp* C$› ‹$A_Q \sharp* C$› ‹$A_P \sharp* M$› ‹$A_Q \sharp* M$› ‹$A_P \sharp* N$› ‹$A_Q \sharp* N$›
ultimately have *Prop* $C\ \Psi\ (P \parallel Q)$ ($¡M⦇\nu*xvec⦈\langle N\rangle \prec (P' \parallel Q')$) $(p \cdot (A_P@A_Q))$ $(p \cdot (\Psi_P \otimes \Psi_Q))$
by(*fastforce intro*: *rAlpha*)
with $\Psi eq\ Aeq$ **show** *?case* **by** *simp*
next
case(*cBrComm2* $\Psi\ \Psi_Q\ P\ M\ xvec\ N\ P'\ A_P\ \Psi_P\ Q\ Q'\ A_Q\ A_{PQ}\ \Psi_{PQ}\ C$)
from ‹*distinct* A_P› ‹*distinct* A_Q› ‹$A_P \sharp* A_Q$› **have** *distinct*$(A_P@A_Q)$
by(*auto simp add*: *fresh-star-def fresh-def name-list-supp*)
from *cBrComm2* **have** *Prop* $C\ \Psi\ (P \parallel Q)$ ($¡M⦇\nu*xvec⦈\langle N\rangle \prec (P' \parallel Q')$) $(A_P@A_Q)$ $(\Psi_P \otimes \Psi_Q)$

by(*metis rBrComm2*)
moreover from ‹*extractFrame*$(P \parallel Q) = \langle A_{PQ}, \Psi_{PQ}\rangle$› ‹*extractFrame* $P = \langle A_P, \Psi_P\rangle$› ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$›
‹$A_P \sharp* A_Q$› ‹$A_P \sharp* \Psi_Q$› ‹$A_Q \sharp* \Psi_P$›
have $\langle(A_P@A_Q), (\Psi_P \otimes \Psi_Q)\rangle = \langle A_{PQ}, \Psi_{PQ}\rangle$
by *simp*
with ‹$A_P \sharp* A_{PQ}$› ‹$A_Q \sharp* A_{PQ}$› ‹*distinct*$(A_P@A_Q)$› ‹*distinct* A_{PQ}›
obtain p **where** S: (*set* $p \subseteq$ (*set*$(A_P@A_Q)) \times$ (*set* A_{PQ})) **and** *distinctPerm* p
and Ψeq: $\Psi_{PQ} = p \cdot (\Psi_P \otimes \Psi_Q)$ **and** Aeq: $A_{PQ} = p \cdot (A_P@A_Q)$
apply −
apply(*rule frameChainEq′*)
by(*assumption* | *simp*)+
moreover note ‹$A_P \sharp* \Psi$› ‹$A_Q \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_Q \sharp* P$› ‹$A_P \sharp* Q$› ‹$A_Q \sharp* Q$› ‹$A_P \sharp* xvec$›
‹$A_Q \sharp* xvec$› ‹$A_P \sharp* P'$› ‹$A_Q \sharp* P'$› ‹$A_P \sharp* Q'$› ‹$A_Q \sharp* Q'$› ‹$A_P \sharp* A_{PQ}$› ‹$A_Q \sharp* A_{PQ}$›
‹$A_P \sharp* C$› ‹$A_Q \sharp* C$› ‹$A_P \sharp* M$› ‹$A_Q \sharp* M$› ‹$A_P \sharp* N$› ‹$A_Q \sharp* N$›
ultimately have *Prop* $C\ \Psi\ (P \parallel Q)$ $(¡M(\!|\nu*xvec|\!)\langle N\rangle \prec (P' \parallel Q'))$ $(p \cdot (A_P@A_Q))$ $(p \cdot (\Psi_P \otimes \Psi_Q))$
by(*fastforce intro*: *rAlpha*)
with Ψeq Aeq **show** *?case* **by** *simp*
next
case(*cBrClose* Ψ P M *xvec* N P' x $A_P{}'$ $\Psi_P{}'$ C)
obtain A_P Ψ_P **where** *FrP*: *extractFrame* $P = \langle A_P, \Psi_P\rangle$ **and** *distinct* A_P
and $A_P \sharp* (\Psi, P, M, xvec, N, P', A_P{}', \Psi_P{}', C, x)$
by(*rule freshFrame*)
then have $A_P \sharp* \Psi$ **and** $A_P \sharp* P$ **and** $A_P \sharp* M$ **and** $A_P \sharp* xvec$ **and** $A_P \sharp* N$ **and** $A_P \sharp* P'$
and $A_P \sharp* A_P{}'$ **and** $A_P \sharp* \Psi_P{}'$ **and** $A_P \sharp* C$ **and** $x \sharp A_P$
by *simp*+
from *FrP* ‹$A_P \sharp* xvec$› ‹$xvec \sharp* P$› **have** $xvec \sharp* \Psi_P$
by(*force dest*: *extractFrameFreshChain*)
from ‹$A_P \sharp* xvec$› ‹$A_P \sharp* P'$› ‹$x \sharp A_P$›
have $A_P \sharp* (\tau \prec (\!|\nu x|\!)((\!|\nu*xvec|\!)P'))$ **by** *simp*
from ‹*extractFrame* $P = \langle A_P, \Psi_P\rangle$› ‹*extractFrame* $((\!|\nu x|\!)P) = \langle A_P{}', \Psi_P{}'\rangle$›
have $\langle(x\#A_P), \Psi_P\rangle = \langle A_P{}', \Psi_P{}'\rangle$ **by** *simp*
with ‹$A_P \sharp* A_P{}'$› ‹$x \sharp A_P{}'$› ‹$x \sharp A_P$› ‹*distinct* A_P› ‹*distinct* $A_P{}'$›
obtain p **where** S: (*set* $p \subseteq$ (*set* $(x\#A_P)) \times$ (*set* $A_P{}'$)) **and** *distinctPerm* p
and Ψeq: $\Psi_P{}' = p \cdot \Psi_P$ **and** Aeq: $A_P{}' = p \cdot (x\#A_P)$
apply −
apply(*rule frameChainEq′*)
by(*assumption* | *simp*)+

from ‹$x \sharp A_P{}'$› Aeq **have** $x \sharp (p \cdot (x\#A_P))$ **by** *simp*
moreover from S Aeq ‹*distinct* $A_P{}'$› ‹$x \sharp A_P{}'$› ‹$A_P \sharp* A_P{}'$› **have** $(p \cdot x) \sharp A_P$
by *simp*
from *cBrClose FrP* ‹*distinct* A_P› ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_P \sharp* M$› ‹$A_P \sharp* xvec$› ‹$A_P \sharp* N$› ‹$A_P \sharp* P'$›
‹$A_P \sharp* A_P{}'$› ‹$A_P \sharp* \Psi_P{}'$› ‹$A_P \sharp* C$› ‹$x \sharp A_P$› ‹$xvec \sharp* \Psi_P$› ‹$A_P \sharp* C$› ‹*xvec*

$\sharp*\ C\rangle\ \langle x\ \sharp\ C\rangle$
have *Prop C* Ψ $((\!|\nu x|\!)P)$ $(\tau \prec (\!|\nu x|\!)((\!|\nu{*}xvec|\!)P'))$ $(x\#A_P)$ Ψ_P
by(*force intro*: *rBrClose*)

moreover from *Aeq* $\langle A_P\ \sharp*\ A_P'\rangle$ **have** $A_P\ \sharp*\ (p \cdot A_P)$ **by** *simp*
moreover from *Aeq* $\langle(set\ p \subseteq (set\ (x\#A_P)) \times (set\ A_P'))\rangle$
have $(set\ p \subseteq (set\ (x\#A_P)) \times (set\ (p \cdot (x\#A_P))))$ **by** *simp*
moreover from $\langle A_P\ \sharp*\ P\rangle\ \langle x\ \sharp\ A_P\rangle$ **have** $A_P\ \sharp*\ ((\!|\nu x|\!)P)$ **by** *simp*
moreover from *S* $\langle x\ \sharp\ A_P'\rangle$ **have** $p \cdot x \neq x$
using *Aeq* **by** *fastforce*
moreover from $\langle x\ \sharp\ A_P'\rangle$ *Aeq* **have** $x\ \sharp\ p \cdot A_P$ **by** *simp*
moreover note $\langle A_P\ \sharp*\ \Psi\rangle\ \langle A_P\ \sharp*\ P\rangle\ \langle A_P\ \sharp*\ (\tau \prec (\!|\nu x|\!)((\!|\nu{*}xvec|\!)P'))\rangle$
$\langle A_P\ \sharp*\ C\rangle\ \langle distinctPerm\ p\rangle$
$\langle x\ \sharp\ \Psi\rangle\ \langle x\ \sharp\ C\rangle\ \langle(p \cdot x)\ \sharp\ A_P\rangle$
ultimately
have *Prop C* Ψ $((\!|\nu x|\!)P)$ $(\tau \prec (\!|\nu x|\!)((\!|\nu{*}xvec|\!)P'))$ $(p \cdot (x\#A_P))$ $(p \cdot \Psi_P)$
by(*fastforce intro*!: *rAlpha simp add*: *abs-fresh*)
with Ψeq *Aeq* **show** *?case* **by** *simp*
next
case(*cOpen* Ψ *P M xvec yvec N P' x* A_{xP} Ψ_{xP} *C*)
obtain A_P Ψ_P **where** *FrP*: *extractFrame P* $= \langle A_P, \Psi_P\rangle$ **and** *distinct* A_P
and $A_P\ \sharp*\ (\Psi, P, M, xvec, yvec, N, P', A_{xP}, \Psi_{xP}, C, x)$
by(*rule freshFrame*)
then have $A_P\ \sharp*\ \Psi$ **and** $A_P\ \sharp*\ P$ **and** $A_P\ \sharp*\ M$ **and** $A_P\ \sharp*\ xvec$**and** $A_P\ \sharp*$ *yvec* **and** $A_P\ \sharp*\ N$ **and** $A_P\ \sharp*\ P'$
and $A_P\ \sharp*\ A_{xP}$ **and** $A_P\ \sharp*\ \Psi_{xP}$ **and** $A_P\ \sharp*\ C$ **and** $x\ \sharp\ A_P$
by *simp+*

from $\langle xvec\ \sharp*\ P\rangle\ \langle A_P\ \sharp*\ xvec\rangle$ *FrP* **have** $xvec\ \sharp*\ \Psi_P$
by(*force dest*: *extractFrameFreshChain*)
from $\langle yvec\ \sharp*\ P\rangle\ \langle A_P\ \sharp*\ yvec\rangle$ *FrP* **have** $yvec\ \sharp*\ \Psi_P$
by(*force dest*: *extractFrameFreshChain*)

from $\langle extractFrame((\!|\nu x|\!)P) = \langle A_{xP}, \Psi_{xP}\rangle\rangle$ *FrP*
have $\langle(x\#A_P), \Psi_P\rangle = \langle A_{xP}, \Psi_{xP}\rangle$
by *simp*
moreover from $\langle x\ \sharp\ A_P\rangle\ \langle distinct\ A_P\rangle$ **have** $distinct(x\#A_P)$ **by** *simp*
ultimately obtain *p* **where** *S*: *set p* $\subseteq$ *set* $(x\#A_P) \times$ *set* $(p \cdot (x\#A_P))$ **and** *distinctPerm p*
and Ψeq: $\Psi_{xP} = p \cdot \Psi_P$ **and** *Aeq*: $A_{xP} = (p \cdot x)\#(p \cdot A_P)$
using $\langle A_P\ \sharp*\ A_{xP}\rangle\langle x\ \sharp\ A_{xP}\rangle\ \langle distinct\ A_{xP}\rangle$
apply $-$
apply(*rule frameChainEq'*)
by(*assumption* | *simp*)+

note $\langle\Psi \rhd P \longmapsto M(\!|\nu{*}(xvec@yvec)|\!)\langle N\rangle \prec P'\rangle$ *FrP* $\langle distinct\ A_P\rangle$
moreover from *FrP* $\langle distinct\ A_P\rangle\ \langle\bigwedge A_P\ \Psi_P\ C.\ [\![extractFrame\ P = \langle A_P, \Psi_P\rangle;$ *distinct* $A_P]\!] \Longrightarrow Prop\ C\ \Psi\ P\ (M(\!|\nu{*}(xvec@yvec)|\!)\langle N\rangle \prec P')\ A_P\ \Psi_P\rangle$
have $\bigwedge C.\ Prop\ C\ \Psi\ P\ (M(\!|\nu{*}(xvec@yvec)|\!)\langle N\rangle \prec P')\ A_P\ \Psi_P$ **by** *simp*

moreover note ‹$x \sharp \Psi$› ‹$x \sharp M$› ‹$x \sharp xvec$› ‹$x \sharp yvec$› ‹$x \in supp\ N$› ‹$x \sharp A_P$› ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_P \sharp* M$› ‹$A_P \sharp* xvec$› ‹$A_P \sharp* yvec$› ‹$A_P \sharp* N$› ‹$A_P \sharp* P'$›
‹$xvec \sharp* \Psi$› ‹$xvec \sharp* P$› ‹$xvec \sharp* M$› ‹$xvec \sharp* \Psi_P$› ‹$yvec \sharp* \Psi$› ‹$yvec \sharp* P$› ‹$yvec \sharp* M$› ‹$yvec \sharp* \Psi_P$›
‹$A_P \sharp* C$› ‹$x \sharp C$› ‹$xvec \sharp* C$› ‹$yvec \sharp* C$› ‹$xvec \sharp* yvec$› ‹*distinct xvec*› ‹*distinct yvec*›
ultimately have *Prop* $C\ \Psi\ (\langle\!|\nu x|\!\rangle P)\ (M\langle\!|\nu*(xvec@x\#yvec)|\!\rangle\langle N\rangle \prec P')\ (x\#A_P)\ \Psi_P$
by(*metis rOpen*)

with ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_P \sharp* M$› ‹$A_P \sharp* xvec$› ‹$A_P \sharp* yvec$› ‹$A_P \sharp* N$› ‹$A_P \sharp* P'$› ‹$A_P \sharp* A_{xP}$› ‹$A_P \sharp* C$› ‹$x \sharp A_{xP}$› ‹$A_P \sharp* A_{xP}$› ‹$x \sharp A_P$›
‹$x \sharp \Psi$› ‹$x \sharp M$› ‹$x \sharp C$› ‹$x \sharp xvec$› ‹$x \sharp yvec$› *Aeq*
S ‹*distinctPerm p*›
have *Prop* $C\ \Psi\ (\langle\!|\nu x|\!\rangle P)\ (M\langle\!|\nu*(xvec@x\#yvec)|\!\rangle\langle N\rangle \prec P')\ (p \cdot (x\#A_P))\ (p \cdot \Psi_P)$
apply −
apply(*rule rAlpha*[**where** $A_P{=}x\#A_P$])
by(*assumption* | *simp add*: *abs-fresh fresh-star-def boundOutputFresh*)+
with Ψ*eq Aeq* **show** *?case* **by**(*simp add*: *eqvts*)
next
case(*cBrOpen* Ψ *P M xvec yvec N P' x* A_{xP} Ψ_{xP} *C*)
obtain $A_P\ \Psi_P$ **where** *FrP*: *extractFrame P* = $\langle A_P, \Psi_P\rangle$ **and** *distinct* A_P
and $A_P \sharp* (\Psi, P, M, xvec, yvec, N, P', A_{xP}, \Psi_{xP}, C, x)$
by(*rule freshFrame*)
then have $A_P \sharp* \Psi$ **and** $A_P \sharp* P$ **and** $A_P \sharp* M$ **and** $A_P \sharp* xvec$**and** $A_P \sharp* yvec$ **and** $A_P \sharp* N$ **and** $A_P \sharp* P'$
and $A_P \sharp* A_{xP}$ **and** $A_P \sharp* \Psi_{xP}$ **and** $A_P \sharp* C$ **and** $x \sharp A_P$
by *simp*+

from ‹$xvec \sharp* P$› ‹$A_P \sharp* xvec$› *FrP* **have** $xvec \sharp* \Psi_P$
by(*force dest*: *extractFrameFreshChain*)
from ‹$yvec \sharp* P$› ‹$A_P \sharp* yvec$› *FrP* **have** $yvec \sharp* \Psi_P$
by(*force dest*: *extractFrameFreshChain*)

from ‹*extractFrame*$(\langle\!|\nu x|\!\rangle P) = \langle A_{xP}, \Psi_{xP}\rangle$› *FrP*
have $\langle(x\#A_P), \Psi_P\rangle = \langle A_{xP}, \Psi_{xP}\rangle$
by *simp*
moreover from ‹$x \sharp A_P$› ‹*distinct* A_P› **have** *distinct*$(x\#A_P)$ **by** *simp*
ultimately obtain *p* **where** *S*: *set p* $\subseteq$ *set* $(x\#A_P) \times$ *set* $(p \cdot (x\#A_P))$ **and** *distinctPerm p*
and Ψ*eq*: $\Psi_{xP} = p \cdot \Psi_P$ **and** *Aeq*: $A_{xP} = (p \cdot x)\#(p \cdot A_P)$
using ‹$A_P \sharp* A_{xP}$›‹$x \sharp A_{xP}$› ‹*distinct* A_{xP}›
apply −
apply(*rule frameChainEq'*)
by(*assumption* | *simp*)+

note ‹$\Psi \rhd P \longmapsto ¡M\langle\!|\nu*(xvec@yvec)|\!\rangle\langle N\rangle \prec P'$› *FrP* ‹*distinct* A_P›

moreover from *FrP* ‹*distinct* A_P› ‹$\bigwedge A_P\ \Psi_P\ C.\ [\![extractFrame\ P = \langle A_P, \Psi_P\rangle; distinct\ A_P]\!] \Longrightarrow Prop\ C\ \Psi\ P\ (¡M(\!|\nu*(xvec@yvec)|\!)\langle N\rangle \prec P')\ A_P\ \Psi_P$›
have $\bigwedge C.\ Prop\ C\ \Psi\ P\ (¡M(\!|\nu*(xvec@yvec)|\!)\langle N\rangle \prec P')\ A_P\ \Psi_P$ **by** *simp*
moreover note ‹$x \sharp \Psi$› ‹$x \sharp M$› ‹$x \sharp xvec$› ‹$x \sharp yvec$› ‹$x \in supp\ N$› ‹$x \sharp A_P$› ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_P \sharp* M$› ‹$A_P \sharp* xvec$› ‹$A_P \sharp* yvec$› ‹$A_P \sharp* N$› ‹$A_P \sharp* P'$›
‹$xvec \sharp* \Psi$› ‹$xvec \sharp* P$› ‹$xvec \sharp* M$› ‹$xvec \sharp* \Psi_P$› ‹$yvec \sharp* \Psi$› ‹$yvec \sharp* P$› ‹$yvec \sharp* M$› ‹$yvec \sharp* \Psi_P$›
‹$A_P \sharp* C$› ‹$x \sharp C$› ‹$xvec \sharp* C$› ‹$yvec \sharp* C$› ‹$xvec \sharp* yvec$› ‹*distinct xvec*› ‹*distinct yvec*›
ultimately have $Prop\ C\ \Psi\ (\!|\nu x|\!)P)\ (¡M(\!|\nu*(xvec@x\#yvec)|\!)\langle N\rangle \prec P')\ (x\#A_P)\ \Psi_P$
by(*metis rBrOpen*)

with ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_P \sharp* M$› ‹$A_P \sharp* xvec$› ‹$A_P \sharp* yvec$› ‹$A_P \sharp* N$› ‹$A_P \sharp* P'$› ‹$A_P \sharp* A_{xP}$› ‹$A_P \sharp* C$› ‹$x \sharp A_{xP}$› ‹$A_P \sharp* A_{xP}$› ‹$x \sharp A_P$›
‹$x \sharp \Psi$› ‹$x \sharp M$› ‹$x \sharp C$› ‹$x \sharp xvec$› ‹$x \sharp yvec$› *Aeq*
S ‹*distinctPerm p*›
have $Prop\ C\ \Psi\ (\!|\nu x|\!)P)\ (¡M(\!|\nu*(xvec@x\#yvec)|\!)\langle N\rangle \prec P')\ (p \cdot (x\#A_P))\ (p \cdot \Psi_P)$
apply −
apply(*rule rAlpha*[**where** $A_P{=}x\#A_P$])
by(*assumption* | *simp add*: *abs-fresh fresh-star-def boundOutputFresh*)+
with Ψeq *Aeq* **show** *?case* **by**(*simp add*: *eqvts*)
next
case(*cScope* $\Psi\ P\ \alpha\ P'\ x\ A_{xP}\ \Psi_{xP}\ C$)
obtain $A_P\ \Psi_P$ **where** *FrP*: $extractFrame\ P = \langle A_P, \Psi_P\rangle$ **and** $distinct\ A_P$
and $A_P \sharp* (\Psi, P, \alpha, P', A_{xP}, \Psi_{xP}, C, x)$
by(*rule freshFrame*)
then have $A_P \sharp* \Psi$ **and** $A_P \sharp* P$ **and** $A_P \sharp* \alpha$ **and** $A_P \sharp* P'$
and $A_P \sharp* A_{xP}$ **and** $A_P \sharp* \Psi_{xP}$ **and** $A_P \sharp* C$ **and** $x \sharp A_P$
by *simp*+

from ‹$bn\ \alpha \sharp* P$› ‹$A_P \sharp* \alpha$› *FrP* **have** $bn\ \alpha \sharp* \Psi_P$
by(*force dest*: *extractFrameFreshChain*)

from ‹$extractFrame(\!|\nu x|\!)P) = \langle A_{xP}, \Psi_{xP}\rangle$› *FrP*
have $\langle(x\#A_P), \Psi_P\rangle = \langle A_{xP}, \Psi_{xP}\rangle$
by *simp*
moreover from ‹$x \sharp A_P$› ‹$distinct\ A_P$› **have** $distinct(x\#A_P)$ **by** *simp*
ultimately obtain *p* **where** *S*: $set\ p \subseteq set\ (x\#A_P) \times set\ (p \cdot (x\#A_P))$ **and** *distinctPerm p*
and Ψeq: $\Psi_{xP} = p \cdot \Psi_P$ **and** *Aeq*: $A_{xP} = (p \cdot x)\#(p \cdot A_P)$
using ‹$A_P \sharp* A_{xP}$›‹$x \sharp A_{xP}$› ‹$distinct\ A_{xP}$›
apply −
apply(*rule frameChainEq'*)
by(*assumption* | *simp*)+

note ‹$\Psi \rhd P \longmapsto \alpha \prec P'$› *FrP* ‹*distinct* A_P›

moreover from *FrP* ‹*distinct* A_P› ‹$\bigwedge A_P\ \Psi_P\ C.\ [\![$*extractFrame* $P = \langle A_P, \Psi_P\rangle$; *distinct* $A_P]\!] \Longrightarrow$ *Prop* $C\ \Psi\ P\ (\alpha \prec P')\ A_P\ \Psi_P$›
have $\bigwedge C.$ *Prop* $C\ \Psi\ P\ (\alpha \prec P')\ A_P\ \Psi_P$ **by** *simp*
moreover note ‹$x \sharp \Psi$› ‹$x \sharp \alpha$› ‹$x \sharp A_P$› ‹$A_P \sharp^* \Psi$› ‹$A_P \sharp^* P$› ‹$A_P \sharp^* \alpha$› ‹$A_P \sharp^* P'$› ‹*distinct*$(bn\ \alpha)$›
‹$bn\ \alpha \sharp^* \Psi$› ‹$bn\ \alpha \sharp^* P$› ‹$bn\ \alpha \sharp^*$ *subject* α› ‹$bn\ \alpha \sharp^* \Psi_P$› ‹$A_P \sharp^* C$› ‹$x \sharp C$› ‹$bn\ \alpha \sharp^* C$›
ultimately have *Prop* $C\ \Psi\ (\langle\!(\nu x)\!\rangle P)\ (\alpha \prec (\langle\!(\nu x)\!\rangle P'))\ (x\#A_P)\ \Psi_P$
by(*metis rScope*)

with ‹$A_P \sharp^* \Psi$› ‹$A_P \sharp^* P$› ‹$A_P \sharp^* \alpha$› ‹$A_P \sharp^* P'$› ‹$A_P \sharp^* A_{xP}$› ‹$A_P \sharp^* C$› ‹$x \sharp A_{xP}$› ‹$A_P \sharp^* A_{xP}$› ‹$x \sharp A_P$›
‹$x \sharp \Psi$› ‹$x \sharp \alpha$› ‹$x \sharp C$› *Aeq*
S ‹*distinctPerm p*›
have *Prop* $C\ \Psi\ (\langle\!(\nu x)\!\rangle P)\ (\alpha \prec (\langle\!(\nu x)\!\rangle P'))\ (p \cdot (x\#A_P))\ (p \cdot \Psi_P)$
apply −
apply(*rule rAlpha*[**where** A_P=$x\#A_P$])
by(*assumption* | *simp add*: *abs-fresh fresh-star-def*)+
with *Ψeq Aeq* **show** *?case* **by**(*simp add*: *eqvts*)
next
case(*Bang* $\Psi\ P\ Rs\ A_{bP}\ \Psi_{bP}\ C$)

obtain $A_P\ \Psi_P$ **where** *FrP*: *extractFrame* $P = \langle A_P, \Psi_P\rangle$ **and** *distinct* A_P
and $A_P \sharp^* (\Psi, P, Rs, C)$
by(*rule freshFrame*)
then have $A_P \sharp^* \Psi$ **and** $A_P \sharp^* P$ **and** $A_P \sharp^* Rs$ **and** $A_P \sharp^* C$
by *simp*+

note ‹$\Psi \rhd P \parallel\ !P \longmapsto Rs$› ‹*guarded P*› *FrP* ‹*distinct* A_P›
moreover from *FrP* **have** *extractFrame* $(P \parallel\ !P) = \langle A_P, \Psi_P \otimes \mathbf{1}\rangle$
by *simp*
with ‹*distinct* A_P› ‹$\bigwedge A_P\ \Psi_P\ C.\ [\![$*extractFrame* $(P \parallel\ !P) = \langle A_P, \Psi_P\rangle$; *distinct* $A_P]\!] \Longrightarrow$ *Prop* $C\ \Psi\ (P \parallel\ !P)\ Rs\ A_P\ \Psi_P$›
have $\bigwedge C.$ *Prop* $C\ \Psi\ (P \parallel\ !P)\ Rs\ A_P\ (\Psi_P \otimes \mathbf{1})$ **by** *simp*
moreover from ‹*guarded P*› *FrP* **have** $\Psi_P \simeq \mathbf{1}$ **and** *supp* $\Psi_P = (\{\}$::*name set*) **by**(*metis guardedStatEq*)+
moreover note ‹$A_P \sharp^* \Psi$› ‹$A_P \sharp^* P$› ‹$A_P \sharp^* Rs$› ‹$A_P \sharp^* C$›
ultimately have *Prop* $C\ \Psi\ (!P)\ Rs\ ([])\ (\mathbf{1})$
by(*rule rBang*)
then show *?case* **using** ‹*extractFrame*$(!P) = \langle A_{bP}, \Psi_{bP}\rangle$› **by** *simp*
qed

lemma *semanticsFrameInduct'*[*consumes 5*, *case-names cAlpha cFrameAlpha cInput cBrInput cOutput cBrOutput cCase cPar1 cPar2 cComm1 cComm2 cBrMerge cBrComm1 cBrComm2 cBrClose cOpen cBrOpen 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* :: *'f*::*fs-name* ⇒ *'b* ⇒ (*'a*, *'b*, *'c*) *psi* ⇒ *'a action* ⇒
(*'a*, *'b*, *'c*) *psi* ⇒ *name list* ⇒ *'b* ⇒ *bool*
and *C* :: *'f*::*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* α)
and *rAlpha*: $\bigwedge \Psi\ P\ \alpha\ P'\ p\ A_P\ \Psi_P\ C.$ ⟦*bn* $\alpha \sharp* \Psi$; *bn* $\alpha \sharp* P$; *bn* $\alpha \sharp* subject\ \alpha$; *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* $C\ \Psi\ P\ \alpha\ P'\ A_P\ \Psi_P$⟧ ⟹
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 $C\ \Psi\ P\ \alpha\ P'\ A_P\ \Psi_P$⟧ ⟹ *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* $\sharp* C$⟧ ⟹
Prop $C\ \Psi\ (M(\!|\lambda* xvec\ N|\!).P)$
$(K(\!|(N[xvec ::= Tvec])|\!))\ (P[xvec ::= Tvec])\ ([])\ (\mathbf{1})$
and *rBrInput*: $\bigwedge \Psi\ M\ K\ xvec\ N\ Tvec\ P\ C.$
⟦$\Psi \vdash K \succeq M$; *distinct xvec*; *set xvec* $\subseteq$ *supp* N;
length xvec = *length Tvec*; *xvec* $\sharp* \Psi$;
xvec $\sharp* M$; *xvec* $\sharp* K$; *xvec* $\sharp* C$⟧ ⟹
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 *rBrOutput*: $\bigwedge \Psi\ M\ K\ N\ P\ C.\ \Psi \vdash M \preceq K \Longrightarrow Prop\ C\ \Psi\ (M\langle N \rangle.P)\ (¡K\langle N \rangle)\ P\ ([])\ (\mathbf{1})$
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* $C\ \Psi\ P\ \alpha\ P'\ A_P\ \Psi_P$;
$(\varphi, P) \in set\ Cs$; $\Psi \vdash \varphi$; *guarded* P; $\Psi_P \simeq \mathbf{1}$; $(supp\ \Psi_P) = (\{\}::name\ set)$;
$A_P \sharp* \Psi$; $A_P \sharp* P$; $A_P \sharp* \alpha$; $A_P \sharp* P'$; $A_P \sharp* C$⟧ ⟹ *Prop* $C\ \Psi\ (Cases\ Cs)\ \alpha\ P'\ ([])\ (\mathbf{1})$
and *rPar1*: $\bigwedge \Psi\ \Psi_Q\ P\ \alpha\ P'\ A_Q\ Q\ A_P\ \Psi_P\ C.$
⟦$\Psi \otimes \Psi_Q \rhd P \longmapsto \alpha \prec P'$;

$$
\begin{array}{l}
\quad\quad\quad\quad \mathit{extractFrame}\ P = \langle A_P,\ \Psi_P\rangle;\ \mathit{distinct}\ A_P;\\
\quad\quad\quad\quad \mathit{extractFrame}\ Q = \langle A_Q,\ \Psi_Q\rangle;\ \mathit{distinct}\ A_Q;\\
\quad\quad\quad\quad \bigwedge C.\ \mathit{Prop}\ C\ (\Psi \otimes \Psi_Q)\ P\ \alpha\ P'\ A_P\ \Psi_P;\\
\quad\quad\quad\quad 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;\\
\quad\quad\quad\quad 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;\\
\quad\quad\quad\quad \mathit{bn}\ \alpha \sharp* \Psi;\ \mathit{bn}\ \alpha \sharp* P;\ \mathit{bn}\ \alpha \sharp* Q;\ \mathit{bn}\ \alpha \sharp* \mathit{subject}\ \alpha;\ \mathit{bn}\ \alpha \sharp* \Psi_P;\ \mathit{bn}\ \alpha \sharp* \Psi_Q;\\
\quad\quad\quad\quad A_P \sharp* C;\ A_Q \sharp* C;\ \mathit{bn}\ \alpha \sharp* C[\!] \Longrightarrow\\
\quad\quad\quad\quad \mathit{Prop}\ C\ \Psi\ (P \parallel Q)\ \alpha\ (P' \parallel Q)\ (A_P @ A_Q)\ (\Psi_P \otimes \Psi_Q)\\
\textbf{and}\quad \mathit{rPar2}\colon \bigwedge \Psi\ \Psi_P\ Q\ \alpha\ Q'\ A_P\ P\ A_Q\ \Psi_Q\ C.\\
\quad\quad\quad\quad [\![\Psi \otimes \Psi_P \rhd Q \longmapsto \alpha \prec Q';\\
\quad\quad\quad\quad \mathit{extractFrame}\ P = \langle A_P,\ \Psi_P\rangle;\ \mathit{distinct}\ A_P;\\
\quad\quad\quad\quad \mathit{extractFrame}\ Q = \langle A_Q,\ \Psi_Q\rangle;\ \mathit{distinct}\ A_Q;\\
\quad\quad\quad\quad \bigwedge C.\ \mathit{Prop}\ C\ (\Psi \otimes \Psi_P)\ Q\ \alpha\ Q'\ A_Q\ \Psi_Q;\\
\quad\quad\quad\quad 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;\\
\quad\quad\quad\quad 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;\\
\quad\quad\quad\quad \mathit{bn}\ \alpha \sharp* \Psi;\ \mathit{bn}\ \alpha \sharp* P;\ \mathit{bn}\ \alpha \sharp* Q;\ \mathit{bn}\ \alpha \sharp* \mathit{subject}\ \alpha;\ \mathit{bn}\ \alpha \sharp* \Psi_P;\ \mathit{bn}\ \alpha \sharp* \Psi_Q;\\
\quad\quad\quad\quad A_P \sharp* C;\ A_Q \sharp* C;\ \mathit{bn}\ \alpha \sharp* C[\!] \Longrightarrow\\
\quad\quad\quad\quad \mathit{Prop}\ C\ \Psi\ (P \parallel Q)\ \alpha\ (P \parallel Q')\ (A_P @ A_Q)\ (\Psi_P \otimes \Psi_Q)\\
\textbf{and}\quad \mathit{rComm1}\colon \bigwedge \Psi\ \Psi_Q\ P\ M\ N\ P'\ A_P\ \Psi_P\ Q\ K\ \mathit{xvec}\ Q'\ A_Q\ C.\\
\quad\quad\quad\quad\quad [\![\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|N|\!) \prec P';\ \mathit{extractFrame}\ P = \langle A_P,\ \Psi_P\rangle;\ \mathit{distinct}\ A_P;\\
\quad\quad\quad\quad \bigwedge C.\ \mathit{Prop}\ C\ (\Psi \otimes \Psi_Q)\ P\ (M(\!|N|\!))\ P'\ A_P\ \Psi_P;\\
\quad\quad\quad\quad\quad \Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|\nu{*}\mathit{xvec}|\!)\langle N\rangle \prec Q';\ \mathit{extractFrame}\ Q = \langle A_Q,\ \Psi_Q\rangle;\ \mathit{distinct}\ A_Q;\\
\quad\quad\quad\quad \Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K;\ \mathit{distinct}\ \mathit{xvec};\\
\quad\quad\quad\quad \bigwedge C.\ \mathit{Prop}\ C\ (\Psi \otimes \Psi_P)\ Q\ (K(\!|\nu{*}\mathit{xvec}|\!)\langle N\rangle)\ Q'\ A_Q\ \Psi_Q;\\
\quad\quad\quad\quad 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';\\
\quad\quad\quad\quad A_P \sharp* Q;\ A_P \sharp* Q';\ A_P \sharp* A_Q;\ A_P \sharp* \mathit{xvec};\ A_Q \sharp* \Psi;\ A_Q \sharp* \Psi_P;\\
\quad\quad\quad\quad A_Q \sharp* P;\ A_Q \sharp* N;\ A_Q \sharp* P';\ A_Q \sharp* Q;\ A_Q \sharp* K;\ A_Q \sharp* Q';\\
\quad\quad\quad\quad A_Q \sharp* \mathit{xvec};\ \mathit{xvec} \sharp* \Psi;\ \mathit{xvec} \sharp* \Psi_P;\ \mathit{xvec} \sharp* \Psi_Q;\ \mathit{xvec} \sharp* P;\ \mathit{xvec} \sharp* M;\\
\quad\quad\quad\quad\quad \mathit{xvec} \sharp* Q;\ \mathit{xvec} \sharp* K;\ A_P \sharp* C;\ A_Q \sharp* C;\ \mathit{xvec} \sharp* C[\!] \Longrightarrow\\
\quad\quad\quad\quad \mathit{Prop}\ C\ \Psi\ (P \parallel Q)\ (\tau)\ ((\!|\nu{*}\mathit{xvec}|\!)(P' \parallel Q'))\ (A_P @ A_Q)\ (\Psi_P \otimes \Psi_Q)\\
\textbf{and}\quad \mathit{rComm2}\colon \bigwedge \Psi\ \Psi_Q\ P\ M\ \mathit{xvec}\ N\ P'\ A_P\ \Psi_P\ Q\ K\ Q'\ A_Q\ C.\\
\quad\quad\quad\quad\quad [\![\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|\nu{*}\mathit{xvec}|\!)\langle N\rangle \prec P';\ \mathit{extractFrame}\ P = \langle A_P,\ \Psi_P\rangle;\ \mathit{distinct}\ A_P;\\
\quad\quad\quad\quad \bigwedge C.\ \mathit{Prop}\ C\ (\Psi \otimes \Psi_Q)\ P\ (M(\!|\nu{*}\mathit{xvec}|\!)\langle N\rangle)\ P'\ A_P\ \Psi_P;\\
\quad\quad\quad\quad\quad \Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|N|\!) \prec Q';\ \mathit{extractFrame}\ Q = \langle A_Q,\ \Psi_Q\rangle;\ \mathit{distinct}\ A_Q;\\
\quad\quad\quad\quad \bigwedge C.\ \mathit{Prop}\ C\ (\Psi \otimes \Psi_P)\ Q\ (K(\!|N|\!))\ Q'\ A_Q\ \Psi_Q;\\
\quad\quad\quad\quad \Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K;\ \mathit{distinct}\ \mathit{xvec};\\
\quad\quad\quad\quad 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';\\
\quad\quad\quad\quad A_P \sharp* Q;\ A_P \sharp* Q';\ A_P \sharp* A_Q;\ A_P \sharp* \mathit{xvec};\ A_Q \sharp* \Psi;\ A_Q \sharp* \Psi_P;\\
\quad\quad\quad\quad A_Q \sharp* P;\ A_Q \sharp* N;\ A_Q \sharp* P';\ A_Q \sharp* Q;\ A_Q \sharp* K;\ A_Q \sharp* Q';\\
\quad\quad\quad\quad A_Q \sharp* \mathit{xvec};\ \mathit{xvec} \sharp* \Psi;\ \mathit{xvec} \sharp* \Psi_P;\ \mathit{xvec} \sharp* \Psi_Q;\ \mathit{xvec} \sharp* P;\ \mathit{xvec} \sharp*
\end{array}
$$

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 *rBrMerge*: $\bigwedge \Psi\ \Psi_Q\ P\ M\ N\ P'\ A_P\ \Psi_P\ Q\ Q'\ A_Q\ C.$
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P';\ \bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_Q)\ P\ (¿M(\!|N|\!))\ P'\ A_P\ \Psi_P;$
$extractFrame\ P = \langle A_P, \Psi_P \rangle$; $distinct\ A_P$;
$\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q';\ \bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_P)\ Q\ (¿M(\!|N|\!))\ Q'\ A_Q\ \Psi_Q;$
$extractFrame\ Q = \langle A_Q, \Psi_Q \rangle$; $distinct\ A_Q$;
$A_P \sharp* \Psi$; $A_P \sharp* \Psi_Q$; $A_P \sharp* P$; $A_P \sharp* N$; $A_P \sharp* P'$;
$A_P \sharp* Q$; $A_P \sharp* Q'$; $A_P \sharp* A_Q$; $A_P \sharp* M$; $A_Q \sharp* M$;
$A_Q \sharp* \Psi$; $A_Q \sharp* \Psi_P$; $A_Q \sharp* P$; $A_Q \sharp* N$; $A_Q \sharp* P'$;
$A_Q \sharp* Q$; $A_Q \sharp* Q'$; $A_P \sharp* C$; $A_Q \sharp* C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (P \parallel Q)\ (¿M(\!|N|\!))\ (P' \parallel Q')\ (A_P @ A_Q)\ (\Psi_P \otimes \Psi_Q)$

and *rBrComm1*: $\bigwedge \Psi\ \Psi_Q\ P\ M\ N\ P'\ A_P\ \Psi_P\ Q\ xvec\ Q'\ A_Q\ C.$
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P';\ extractFrame\ P = \langle A_P, \Psi_P \rangle;$
$distinct\ A_P;$
$\bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_Q)\ P\ (¿M(\!|N|\!))\ P'\ A_P\ \Psi_P;$
$\Psi \otimes \Psi_P \rhd Q \longmapsto ¡M(\!|\nu{*}xvec|\!)\langle N \rangle \prec Q';\ extractFrame\ Q = \langle A_Q, \Psi_Q \rangle;$ $distinct\ A_Q$;
$distinct\ xvec$;
$\bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_P)\ Q\ (¡M(\!|\nu{*}xvec|\!)\langle N \rangle)\ Q'\ A_Q\ \Psi_Q;$
$A_P \sharp* \Psi$; $A_P \sharp* \Psi_Q$; $A_P \sharp* P$; $A_P \sharp* N$; $A_P \sharp* P'$;
$A_P \sharp* Q$; $A_P \sharp* Q'$; $A_P \sharp* A_Q$; $A_P \sharp* xvec$; $A_Q \sharp* \Psi$; $A_Q \sharp* \Psi_P$;
$A_Q \sharp* P$; $A_Q \sharp* N$; $A_Q \sharp* P'$; $A_Q \sharp* Q$; $A_Q \sharp* Q'$;
$A_Q \sharp* xvec$; $xvec \sharp* \Psi$; $xvec \sharp* \Psi_P$; $xvec \sharp* \Psi_Q$; $xvec \sharp* P$;
$xvec \sharp* Q$; $A_P \sharp* C$; $A_Q \sharp* C$; $xvec \sharp* C$;
$A_P \sharp* M$; $A_Q \sharp* M$; $xvec \sharp* M]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (P \parallel Q)\ (¡M(\!|\nu{*}xvec|\!)\langle N \rangle)\ (P' \parallel Q')\ (A_P @ A_Q)\ (\Psi_P \otimes \Psi_Q)$

and *rBrComm2*: $\bigwedge \Psi\ \Psi_Q\ P\ M\ xvec\ N\ P'\ A_P\ \Psi_P\ Q\ Q'\ A_Q\ C.$
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto ¡M(\!|\nu{*}xvec|\!)\langle N \rangle \prec P';\ extractFrame\ P = \langle A_P, \Psi_P \rangle;$ $distinct\ A_P;$
$\bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_Q)\ P\ (¡M(\!|\nu{*}xvec|\!)\langle N \rangle)\ P'\ A_P\ \Psi_P;$
$\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q';\ extractFrame\ Q = \langle A_Q, \Psi_Q \rangle;$ $distinct\ A_Q$;
$\bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_P)\ Q\ (¿M(\!|N|\!))\ Q'\ A_Q\ \Psi_Q;$
$distinct\ xvec$;
$A_P \sharp* \Psi$; $A_P \sharp* \Psi_Q$; $A_P \sharp* P$; $A_P \sharp* N$; $A_P \sharp* P'$;
$A_P \sharp* Q$; $A_P \sharp* Q'$; $A_P \sharp* A_Q$; $A_P \sharp* xvec$; $A_Q \sharp* \Psi$; $A_Q \sharp* \Psi_P$;
$A_Q \sharp* P$; $A_Q \sharp* N$; $A_Q \sharp* P'$; $A_Q \sharp* Q$; $A_Q \sharp* Q'$;
$A_Q \sharp* xvec$; $xvec \sharp* \Psi$; $xvec \sharp* \Psi_P$; $xvec \sharp* \Psi_Q$; $xvec \sharp* P$;
$xvec \sharp* Q$; $A_P \sharp* C$; $A_Q \sharp* C$; $xvec \sharp* C$;
$A_P \sharp* M$; $A_Q \sharp* M$; $xvec \sharp* M]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (P \parallel Q)\ (¡M(\!|\nu{*}xvec|\!)\langle N \rangle)\ (P' \parallel Q')\ (A_P @ A_Q)\ (\Psi_P \otimes \Psi_Q)$

and *rBrClose*: $\bigwedge \Psi\ P\ M\ xvec\ N\ P'\ A_P\ \Psi_P\ x\ C.$
$[\![\Psi \rhd P \longmapsto ¡M(\!|\nu{*}xvec|\!)\langle N \rangle \prec P';$

$x \in supp\ M$;
$\bigwedge C.\ Prop\ C\ \Psi\ P\ (¡M(\!|\nu{*}xvec|\!)\langle N\rangle)\ P'\ A_P\ \Psi_P$;
$extractFrame\ P = \langle A_P, \Psi_P\rangle$; $distinct\ A_P$;
$A_P \sharp* \Psi$; $A_P \sharp* P$; $A_P \sharp* M$; $A_P \sharp* N$; $A_P \sharp* P'$; $A_P \sharp* xvec$;
$distinct\ xvec$; $xvec \sharp* \Psi$; $xvec \sharp* \Psi_P$; $xvec \sharp* P$;
$xvec \sharp* M$;
$x \sharp \Psi$; $x \sharp xvec$; $x \sharp A_P$;
$A_P \sharp* C$; $xvec \sharp* C$; $x \sharp C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ ((\!|\nu x|\!)P)\ (\tau)\ ((\!|\nu x|\!)(\!|\nu{*}xvec|\!)P'))\ (x\#A_P)\ \Psi_P$

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 *rBrOpen*: $\bigwedge \Psi\ P\ M\ xvec\ yvec\ N\ P'\ x\ A_P\ \Psi_P\ y\ C.$
$[\![\Psi \rhd P \longmapsto ¡M(\!|\nu{*}(xvec@yvec)|\!)\langle N\rangle \prec P'$; $extractFrame\ P = \langle A_P, \Psi_P\rangle$; $distinct\ A_P$;
$\bigwedge C.\ Prop\ C\ \Psi\ P\ (¡M(\!|\nu{*}(xvec@yvec)|\!)\langle N\rangle)\ P'\ A_P\ \Psi_P$; $x \in supp\ N$; $x \sharp \Psi$; $x \sharp M$;
$x \sharp A_P$; $x \sharp xvec$; $x \sharp yvec$; $A_P \sharp* \Psi$; $A_P \sharp* P$; $A_P \sharp* M$; $A_P \sharp* N$; $A_P \sharp* P'$;
$A_P \sharp* xvec$; $A_P \sharp* yvec$; $xvec \sharp* yvec$; $distinct\ xvec$; $distinct\ yvec$;
$xvec \sharp* \Psi$; $xvec \sharp* P$; $xvec \sharp* M$; $xvec \sharp* \Psi_P$;
$yvec \sharp* \Psi$; $yvec \sharp* P$; $yvec \sharp* M$; $A_P \sharp* C$; $x \sharp C$; $xvec \sharp* C$; $yvec \sharp* C$;
$y \neq x$; $y \sharp \Psi$; $y \sharp P$; $y \sharp M$; $y \sharp xvec$; $y \sharp yvec$; $y \sharp N$; $y \sharp P'$; $y \sharp A_P$; $y \sharp \Psi_P$; $y \sharp C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ ((\!|\nu x|\!)P)\ (¡M(\!|\nu{*}(xvec@y\#yvec)|\!)\langle([(x,\ y)] \cdot N)\rangle)\ ([(x, y)] \cdot P')\ (x\#A_P)\ \Psi_P$

and *rScope*: $\bigwedge \Psi\ P\ \alpha\ P'\ x\ A_P\ \Psi_P\ C.$
$[\![\Psi \rhd P \longmapsto \alpha \prec P'$; $extractFrame\ P = \langle A_P, \Psi_P\rangle$; $distinct\ A_P$;
$\bigwedge C.\ Prop\ C\ \Psi\ P\ \alpha\ P'\ A_P\ \Psi_P$;
$x \sharp \Psi$; $x \sharp \alpha$; $x \sharp A_P$; $A_P \sharp* \Psi$; $A_P \sharp* P$;
$A_P \sharp* \alpha$; $A_P \sharp* P'$;
$bn\ \alpha \sharp* \Psi$; $bn\ \alpha \sharp* P$; $bn\ \alpha \sharp* subject\ \alpha$; $bn\ \alpha \sharp* \Psi_P$;
$A_P \sharp* C$; $x \sharp C$; $bn\ \alpha \sharp* C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ ((\!|\nu x|\!)P)\ \alpha\ ((\!|\nu x|\!)P')\ (x\#A_P)\ \Psi_P$

and *rBang*: $\bigwedge \Psi\ P\ \alpha\ P'\ A_P\ \Psi_P\ C.$
$[\![\Psi \rhd P \parallel\ !P \longmapsto \alpha \prec P'$; $guarded\ P$; $extractFrame\ P = \langle A_P, \Psi_P\rangle$;

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* $C\ \Psi\ P\ \alpha\ P'\ A_P\ \Psi_P$
using *Trans FrP* ‹*distinct* A_P› ‹*bn* $\alpha \sharp*$ *subject* α› ‹*distinct*(*bn* α)›
proof(*nominal-induct* $\Psi\ P\ Rs==\alpha \prec P'\ A_P\ \Psi_P$ *avoiding*: $C\ \alpha\ P'$ *rule*: *semanticsFrameInduct*)
case *cAlpha*
then show *?case* **using** *rFrameAlpha*
by *auto*
next
case *cInput*
then show *?case* **using** *rInput*
by(*auto simp add*: *residualInject*)
next
case *cBrInput*
then show *?case* **using** *rBrInput*
by(*auto simp add*: *residualInject*)
next
case *cOutput*
then show *?case* **using** *rOutput*
by(*auto simp add*: *residualInject*)
next
case *cBrOutput*
then show *?case* **using** *rBrOutput*
by(*auto simp add*: *residualInject*)
next
case *cCase*
then show *?case* **using** *rCase*
by(*auto simp add*: *residualInject*)
next
case(*cPar1* $\Psi\ \Psi_Q\ P\ \alpha\ P'\ A_Q\ Q\ A_P\ \Psi_P\ C\ \alpha'\ P''$)
note ‹$\alpha \prec (P' \parallel Q) = \alpha' \prec P''$›
moreover from ‹*bn* $\alpha \sharp* \alpha'$› **have** *bn* $\alpha \sharp* (bn\ \alpha')$ **by** *auto*
moreover note ‹*distinct* (*bn* α)› ‹*distinct*(*bn* α')›
moreover from ‹*bn* $\alpha \sharp*$ *subject* α› ‹*bn* $\alpha' \sharp*$ *subject* α'›
have *bn* $\alpha \sharp* (\alpha \prec P' \parallel Q)$ **and** *bn* $\alpha' \sharp* (\alpha' \prec P'')$ **by** *simp+*
ultimately obtain p **where** S: $(set\ p) \subseteq (set(bn\ \alpha)) \times (set(bn(p \cdot \alpha)))$ **and** *distinctPerm* p
and αEq: $\alpha' = p \cdot \alpha$ **and** $P'eq$: $P'' = p \cdot (P' \parallel Q)$ **and** $(bn(p \cdot \alpha)) \sharp* \alpha$
and $(bn(p \cdot \alpha)) \sharp* (P' \parallel Q)$
by(*rule residualEq*)

note ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto \alpha \prec P'$› ‹*extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$› ‹*distinct* A_Q›
moreover from ‹*bn* $\alpha \sharp*$ *subject* α› ‹*distinct*(*bn* α)› ‹$A_P \sharp* \alpha$›
have $\bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_Q)\ P\ \alpha\ P'\ A_P\ \Psi_P$ **by**(*fastforce intro*: *cPar1*)

moreover note ‹$A_Q \sharp* P$› ‹$A_Q \sharp* Q$› ‹$A_Q \sharp* \Psi$› ‹$A_Q \sharp* \alpha$› ‹$A_Q \sharp* P'$› ‹$A_Q \sharp* C$› ‹$A_P \sharp* P$› ‹$A_P \sharp* Q$› ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* \alpha$› ‹$A_P \sharp* P'$› ‹$A_P \sharp* C$›
‹$bn\ \alpha \sharp* Q$› ‹*distinct*$(bn\ \alpha)$› ‹$bn\ \alpha \sharp* \Psi$› ‹$bn\ \alpha \sharp* \Psi_Q$› ‹$bn\ \alpha \sharp* P$› ‹$bn\ \alpha \sharp*$ *subject* α› ‹$bn\ \alpha \sharp* C$›
‹*extractFrame* $P = \langle A_P, \Psi_P\rangle$› ‹*distinct* A_P› ‹$A_P \sharp* A_Q$› ‹$A_P \sharp* \Psi_Q$› ‹$A_Q \sharp* \Psi_P$› ‹$bn\ \alpha \sharp* \Psi_P$›
ultimately have *Prop* $C\ \Psi\ (P \parallel Q)\ \alpha\ (P' \parallel Q)\ (A_P @ A_Q)\ (\Psi_P \otimes \Psi_Q)$
by(*metis rPar1*)
with ‹$bn\ \alpha \sharp* \Psi$› ‹$bn\ \alpha \sharp* P$› ‹$bn\ \alpha \sharp* Q$› ‹$bn\ \alpha \sharp*$ *subject* α› ‹$bn\ \alpha \sharp* C$› ‹$bn\ \alpha \sharp* (bn\ \alpha')$› S ‹*distinctPerm* p› ‹$bn(p \cdot \alpha) \sharp* \alpha$› ‹$bn(p \cdot \alpha) \sharp* (P' \parallel Q)$› ‹$bn\ \alpha \sharp* \Psi_P$› ‹$bn\ \alpha \sharp* \Psi_Q$› ‹$A_P \sharp* \alpha$› ‹$A_Q \sharp* \alpha$› ‹$A_P \sharp* \alpha'$› ‹$A_Q \sharp* \alpha'$› αEq ‹$bn\ \alpha \sharp* \Psi_P$› ‹$bn\ \alpha \sharp* \alpha'$› ‹$A_P \sharp* \Psi$› ‹$A_Q \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_Q \sharp* P$› ‹$A_P \sharp* Q$› ‹$A_Q \sharp* Q$› ‹$A_P \sharp* P'$› ‹$A_Q \sharp* P'$› ‹$A_P \sharp* C$› ‹$A_Q \sharp* C$›
have *Prop* $C\ \Psi\ (P \parallel Q)\ (p \cdot \alpha)\ (p \cdot (P' \parallel Q))\ (A_P @ A_Q)\ (\Psi_P \otimes \Psi_Q)$
by(*fastforce intro*!: *rAlpha*)
with αEq $P'eq$ ‹*distinctPerm* p› **show** *?case* **by** *simp*
next
case(*cPar2* $\Psi\ \Psi_P\ Q\ \alpha\ Q'\ A_P\ P\ A_Q\ \Psi_Q\ C\ \alpha'\ Q''$)
note ‹$\alpha \prec (P \parallel Q') = \alpha' \prec Q''$›
moreover from ‹$bn\ \alpha \sharp* \alpha'$› **have** $bn\ \alpha \sharp* (bn\ \alpha')$ **by** *auto*
moreover note ‹*distinct* $(bn\ \alpha)$› ‹*distinct*$(bn\ \alpha')$›
moreover from ‹$bn\ \alpha \sharp*$ *subject* α› ‹$bn\ \alpha' \sharp*$ *subject* α'›
have $bn\ \alpha \sharp* (\alpha \prec P \parallel Q')$ **and** $bn\ \alpha' \sharp* (\alpha' \prec Q'')$ **by** *simp+*
ultimately obtain p **where** S: $(set\ p) \subseteq (set(bn\ \alpha)) \times (set(bn(p \cdot \alpha)))$ **and** *distinctPerm p*
and αEq: $\alpha' = p \cdot \alpha$ **and** $Q'eq$: $Q'' = p \cdot (P \parallel Q')$ **and** $(bn(p \cdot \alpha)) \sharp* \alpha$
and $(bn(p \cdot \alpha)) \sharp* (P \parallel Q')$
by(*rule residualEq*)

note ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto \alpha \prec Q'$› ‹*extractFrame* $P = \langle A_P, \Psi_P\rangle$› ‹*distinct* A_P›
moreover from ‹$bn\ \alpha \sharp*$ *subject* α› ‹*distinct*$(bn\ \alpha)$› ‹$A_Q \sharp* \alpha$›
have $\bigwedge C.$ *Prop* $C\ (\Psi \otimes \Psi_P)\ Q\ \alpha\ Q'\ A_Q\ \Psi_Q$ **by**(*fastforce intro*!: *cPar2*)

moreover note ‹$A_Q \sharp* P$› ‹$A_Q \sharp* Q$› ‹$A_Q \sharp* \Psi$› ‹$A_Q \sharp* \alpha$› ‹$A_Q \sharp* Q'$› ‹$A_Q \sharp* C$› ‹$A_P \sharp* P$› ‹$A_P \sharp* Q$› ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* \alpha$› ‹$A_P \sharp* Q'$› ‹$A_P \sharp* C$›
‹$bn\ \alpha \sharp* Q$› ‹*distinct*$(bn\ \alpha)$› ‹$bn\ \alpha \sharp* \Psi$› ‹$bn\ \alpha \sharp* \Psi_Q$› ‹$bn\ \alpha \sharp* P$› ‹$bn\ \alpha \sharp*$ *subject* α› ‹$bn\ \alpha \sharp* C$›
‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› ‹*distinct* A_Q› ‹$A_P \sharp* A_Q$› ‹$A_P \sharp* \Psi_Q$› ‹$A_Q \sharp* \Psi_P$› ‹$bn\ \alpha \sharp* \Psi_P$›
ultimately have *Prop* $C\ \Psi\ (P \parallel Q)\ \alpha\ (P \parallel Q')\ (A_P @ A_Q)\ (\Psi_P \otimes \Psi_Q)$
by(*fastforce intro*!: *rPar2*)
with ‹$bn\ \alpha \sharp* \Psi$› ‹$bn\ \alpha \sharp* P$› ‹$bn\ \alpha \sharp* Q$› ‹$bn\ \alpha \sharp*$ *subject* α› ‹$bn\ \alpha \sharp* C$› ‹$bn\ \alpha \sharp* (bn\ \alpha')$› S ‹*distinctPerm* p› ‹$bn(p \cdot \alpha) \sharp* \alpha$› ‹$bn(p \cdot \alpha) \sharp* (P \parallel Q')$› ‹$bn\ \alpha \sharp* \Psi_P$› ‹$bn\ \alpha \sharp* \Psi_Q$› ‹$A_P \sharp* \alpha$› ‹$A_Q \sharp* \alpha$› ‹$A_P \sharp* \alpha'$› ‹$A_Q \sharp* \alpha'$› αEq ‹$bn\ \alpha \sharp* \alpha'$› ‹$A_P \sharp* \Psi$› ‹$A_Q \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_Q \sharp* P$› ‹$A_P \sharp* Q$› ‹$A_Q \sharp* Q$› ‹$A_P \sharp* Q'$› ‹$A_Q \sharp* Q'$› ‹$A_P \sharp* C$› ‹$A_Q \sharp* C$›
have *Prop* $C\ \Psi\ (P \parallel Q)\ (p \cdot \alpha)\ (p \cdot (P \parallel Q'))\ (A_P @ A_Q)\ (\Psi_P \otimes \Psi_Q)$
by(*fastforce intro*!: *rAlpha*)
with αEq $Q'eq$ ‹*distinctPerm* p› **show** *?case* **by** *simp*

next
case(*cComm1* Ψ Ψ_Q P M N P' A_P Ψ_P Q K *xvec* Q' A_Q C α P'')
then show *?case* **using** *rComm1*
apply −
apply(*drule meta-spec*[**where** *x*=$M(\!|N|\!)$])
apply(*drule meta-spec*[**where** *x*=$K(\!|\nu{*}xvec|\!)\langle N\rangle$])
by(*auto simp add*: *residualInject*)
next
case(*cComm2* Ψ Ψ_Q P M *xvec* N P' A_P Ψ_P Q K Q' A_Q C α Q'')
then show *?case* **using** *rComm2*
apply −
apply(*drule meta-spec*[**where** *x*=$M(\!|\nu{*}xvec|\!)\langle N\rangle$])
apply(*drule meta-spec*[**where** *x*=$K(\!|N|\!)$])
by(*auto simp add*: *residualInject*)
next
case(*cBrMerge* Ψ Ψ_Q P M N P' A_P Ψ_P Q Q' A_Q C α P'')
then show *?case* **using** *rBrMerge*
apply −
apply(*drule meta-spec*[**where** *x*=¿$M(\!|N|\!)$])
apply(*drule meta-spec*[**where** *x*=¿$M(\!|N|\!)$])
by(*auto simp add*: *residualInject*)
next
case(*cBrComm1* Ψ Ψ_Q P M N P' A_P Ψ_P Q *xvec* Q' A_Q C α P'')
have *bn* (¿$M(\!|N|\!)$) $\sharp*$ *subject* (¿$M(\!|N|\!)$) **by** *simp*
moreover have *distinct* (*bn* (¿$M(\!|N|\!)$)) **by** *simp*
moreover have ¿$M(\!|N|\!) \prec P' =$ ¿$M(\!|N|\!) \prec P'$ **by** *simp*
moreover note *cBrComm1*
ultimately have *inProp*: $\bigwedge C.$ *Prop* C $(\Psi \otimes \Psi_Q)$ P (¿$M(\!|N|\!)$) P' A_P Ψ_P **by** *simp*

note ‹*xvec* $\sharp*$ M› ‹*distinct xvec*› *cBrComm1*
then have *outProp*: $\bigwedge C.$ *Prop* C $(\Psi \otimes \Psi_P)$ Q (¡$M(\!|\nu{*}xvec|\!)\langle N\rangle$) Q' A_Q Ψ_Q **by** *simp*

note *inProp outProp cBrComm1*
then have *bigProp*: *Prop* C Ψ $(P \parallel Q)$ (¡$M(\!|\nu{*}xvec|\!)\langle N\rangle$) $(P' \parallel Q')$ $(A_P \mathbin{@} A_Q)$ $(\Psi_P \otimes \Psi_Q)$ **by** (*simp add*: *rBrComm1*)

note ‹¡$M(\!|\nu{*}xvec|\!)\langle N\rangle \prec (P' \parallel Q') = \alpha \prec P''$›
moreover from ‹*xvec* $\sharp*$ α› **have** *bn* (¡$M(\!|\nu{*}xvec|\!)\langle N\rangle$) $\sharp*$ (*bn* α) **by** *simp*
moreover from ‹*distinct xvec*› **have** *distinct* (*bn* (¡$M(\!|\nu{*}xvec|\!)\langle N\rangle$)) **by** *simp*
moreover note ‹*distinct*(*bn* α)›
moreover from ‹*xvec* $\sharp*$ M› ‹*bn* α $\sharp*$ *subject* α›
have *bn* (¡$M(\!|\nu{*}xvec|\!)\langle N\rangle$) $\sharp*$ (¡$M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P' \parallel Q'$) **and** *bn* α $\sharp*$ $(\alpha \prec P'')$ **by** *simp+*
ultimately obtain p **where** S: $(set\ p) \subseteq (set(bn\ (¡M(\!|\nu{*}xvec|\!)\langle N\rangle))) \times (set(bn(p \cdot (¡M(\!|\nu{*}xvec|\!)\langle N\rangle))))$ **and** *distinctPerm* p
and αEq: $\alpha = p \cdot (¡M(\!|\nu{*}xvec|\!)\langle N\rangle)$ **and** $P'eq$: $P'' = p \cdot (P' \parallel Q')$
and $(bn(p \cdot (¡M(\!|\nu{*}xvec|\!)\langle N\rangle))) \sharp* (¡M(\!|\nu{*}xvec|\!)\langle N\rangle)$ **and** $(bn(p \cdot (¡M(\!|\nu{*}xvec|\!)\langle N\rangle)))$

$\sharp*\ (P' \parallel Q')$
by(*rule residualEq*) *simp*

from ‹*xvec* $\sharp*$ Ψ› **have** *bn* (¡M⟪$\nu*xvec$⟫⟨N⟩) $\sharp*$ Ψ **by** *simp*
moreover from ‹*xvec* $\sharp*$ P› ‹*xvec* $\sharp*$ Q› **have** *bn* (¡M⟪$\nu*xvec$⟫⟨N⟩) $\sharp*$ $(P \parallel Q)$ **by** *simp*
moreover note ‹*xvec* $\sharp*$ M›
moreover from ‹*xvec* $\sharp*$ Ψ_P› ‹*xvec* $\sharp*$ Ψ_Q› **have** *bn* (¡M⟪$\nu*xvec$⟫⟨N⟩) $\sharp*$ $(\Psi_P \otimes \Psi_Q)$ **by** *auto*
moreover from ‹*xvec* $\sharp*$ C› **have** *bn* (¡M⟪$\nu*xvec$⟫⟨N⟩) $\sharp*$ C **by** *simp*
moreover from ‹*xvec* $\sharp*$ α› αEq **have** *bn* (¡M⟪$\nu*xvec$⟫⟨N⟩) $\sharp*$ $(p \cdot$ (¡M⟪$\nu*xvec$⟫⟨N⟩)) **by** *simp*
moreover from ‹A_P $\sharp*$ Ψ› ‹A_Q $\sharp*$ Ψ› **have** $(A_P$ @ $A_Q)$ $\sharp*$ Ψ **by** *simp*
moreover from ‹A_P $\sharp*$ P› ‹A_Q $\sharp*$ Q› ‹A_P $\sharp*$ Q› ‹A_Q $\sharp*$ P› **have** $(A_P$ @ $A_Q)$ $\sharp*$ $(P \parallel Q)$ **by** *simp*
moreover from ‹A_P $\sharp*$ α› ‹A_Q $\sharp*$ α› **have** $(A_P$ @ $A_Q)$ $\sharp*$ α **by** *simp*
moreover from ‹A_P $\sharp*$ P'› ‹A_Q $\sharp*$ Q'› ‹A_P $\sharp*$ Q'› ‹A_Q $\sharp*$ P'› **have** $(A_P$ @ $A_Q)$ $\sharp*$ $(P' \parallel Q')$ **by** *simp*
moreover from ‹A_P $\sharp*$ C› ‹A_Q $\sharp*$ C› **have** $(A_P$ @ $A_Q)$ $\sharp*$ C **by** *simp*
moreover note S ‹*distinctPerm* p› ‹(*bn*($p \cdot$ (¡M⟪$\nu*xvec$⟫⟨N⟩))) $\sharp*$ (¡M⟪$\nu*xvec$⟫⟨N⟩)›
‹(*bn*($p \cdot$ (¡M⟪$\nu*xvec$⟫⟨N⟩))) $\sharp*$ $(P' \parallel Q')$› *bigProp*
‹A_P $\sharp*$ M› ‹A_Q $\sharp*$ M› ‹A_P $\sharp*$ *xvec*› ‹A_Q $\sharp*$ *xvec*› ‹A_P $\sharp*$ N› ‹A_Q $\sharp*$ N›

ultimately have *Prop* C Ψ $(P \parallel Q)$ $(p \cdot$ (¡M⟪$\nu*xvec$⟫⟨N⟩)) $(p \cdot (P' \parallel Q'))$ $(A_P$ @ $A_Q)$ $(\Psi_P \otimes \Psi_Q)$
by(*fastforce intro*!: *rAlpha*)
then show *?case* **using** αEq $P'eq$ **by** *simp*
next
case(*cBrComm2* Ψ Ψ_Q P M *xvec* N P' A_P Ψ_P Q Q' A_Q C α Q'')
have *bn* (¿M⟪N⟫) $\sharp*$ *subject* (¿M⟪N⟫) **by** *simp*
moreover have *distinct* (*bn* (¿M⟪N⟫)) **by** *simp*
moreover have ¿M⟪N⟫ $\prec$ Q' = ¿M⟪N⟫ $\prec$ Q' **by** *simp*
moreover note *cBrComm2*
ultimately have *inProp*: $\bigwedge C$. *Prop* C $(\Psi \otimes \Psi_P)$ Q (¿M⟪N⟫) Q' A_Q Ψ_Q **by** *simp*

from ‹*xvec* $\sharp*$ M› **have** *bn* (¡M⟪$\nu*xvec$⟫⟨N⟩) $\sharp*$ *subject* (¡M⟪$\nu*xvec$⟫⟨N⟩) **by** *simp*
moreover from ‹*distinct xvec*› **have** *distinct* (*bn* (¡M⟪$\nu*xvec$⟫⟨N⟩)) **by** *simp*
moreover have ¡M⟪$\nu*xvec$⟫⟨N⟩ $\prec$ P' = ¡M⟪$\nu*xvec$⟫⟨N⟩ $\prec$ P' **by** *simp*
moreover note *cBrComm2*
ultimately have *outProp*: $\bigwedge C$. *Prop* C $(\Psi \otimes \Psi_Q)$ P (¡M⟪$\nu*xvec$⟫⟨N⟩) P' A_P Ψ_P **by** *simp*

note *inProp outProp cBrComm2*
then have *bigProp*: *Prop* C Ψ $(P \parallel Q)$ (¡M⟪$\nu*xvec$⟫⟨N⟩) $(P' \parallel Q')$ $(A_P$ @ $A_Q)$ $(\Psi_P \otimes \Psi_Q)$ **by** (*simp add*: *rBrComm2*)

note ‹¡M⟪$\nu*xvec$⟫⟨N⟩ $\prec$ $(P' \parallel Q') = \alpha \prec Q''$›

moreover from ‹$xvec \sharp* \alpha$› **have** $bn\ (¡M(\!|\nu*xvec|\!)\langle N\rangle) \sharp* (bn\ \alpha)$ **by** *simp*
moreover note ‹$distinct\ (bn\ (¡M(\!|\nu*xvec|\!)\langle N\rangle))$› ‹$distinct(bn\ \alpha)$›
moreover from ‹$bn\ (¡M(\!|\nu*xvec|\!)\langle N\rangle) \sharp* subject\ (¡M(\!|\nu*xvec|\!)\langle N\rangle)$› ‹$bn\ \alpha \sharp* subject\ \alpha$›
have $bn\ (¡M(\!|\nu*xvec|\!)\langle N\rangle) \sharp* (¡M(\!|\nu*xvec|\!)\langle N\rangle \prec P' \parallel Q')$ **and** $bn\ \alpha \sharp* (\alpha \prec Q'')$ **by** *simp+*
ultimately obtain p **where** S: $(set\ p) \subseteq (set(bn\ (¡M(\!|\nu*xvec|\!)\langle N\rangle))) \times (set(bn(p \cdot (¡M(\!|\nu*xvec|\!)\langle N\rangle))))$ **and** *distinctPerm p*
and αEq: $\alpha = p \cdot (¡M(\!|\nu*xvec|\!)\langle N\rangle)$ **and** $P'eq$: $Q'' = p \cdot (P' \parallel Q')$ **and** $(bn(p \cdot (¡M(\!|\nu*xvec|\!)\langle N\rangle))) \sharp* (¡M(\!|\nu*xvec|\!)\langle N\rangle)$
and $(bn(p \cdot (¡M(\!|\nu*xvec|\!)\langle N\rangle))) \sharp* (P' \parallel Q')$
by(*rule residualEq*)

from ‹$xvec \sharp* \Psi$› **have** $bn\ (¡M(\!|\nu*xvec|\!)\langle N\rangle) \sharp* \Psi$ **by** *simp*
moreover from ‹$xvec \sharp* P$› ‹$xvec \sharp* Q$› **have** $bn\ (¡M(\!|\nu*xvec|\!)\langle N\rangle) \sharp* (P \parallel Q)$ **by** *simp*
moreover note ‹$bn\ (¡M(\!|\nu*xvec|\!)\langle N\rangle) \sharp* subject\ (¡M(\!|\nu*xvec|\!)\langle N\rangle)$›
moreover from ‹$xvec \sharp* \Psi_P$› ‹$xvec \sharp* \Psi_Q$› **have** $bn\ (¡M(\!|\nu*xvec|\!)\langle N\rangle) \sharp* (\Psi_P \otimes \Psi_Q)$ **by** *auto*
moreover from ‹$xvec \sharp* C$› **have** $bn\ (¡M(\!|\nu*xvec|\!)\langle N\rangle) \sharp* C$ **by** *simp*
moreover from ‹$xvec \sharp* \alpha$› αEq **have** $bn\ (¡M(\!|\nu*xvec|\!)\langle N\rangle) \sharp* (p \cdot (¡M(\!|\nu*xvec|\!)\langle N\rangle))$ **by** *simp*
moreover from ‹$A_P \sharp* \Psi$› ‹$A_Q \sharp* \Psi$› **have** $(A_P \mathbin{@} A_Q) \sharp* \Psi$ **by** *simp*
moreover from ‹$A_P \sharp* P$› ‹$A_Q \sharp* Q$› ‹$A_P \sharp* Q$› ‹$A_Q \sharp* P$› **have** $(A_P \mathbin{@} A_Q) \sharp* (P \parallel Q)$ **by** *simp*
moreover from ‹$A_P \sharp* \alpha$› ‹$A_Q \sharp* \alpha$› **have** $(A_P \mathbin{@} A_Q) \sharp* \alpha$ **by** *simp*
moreover from ‹$A_P \sharp* P'$› ‹$A_Q \sharp* Q'$› ‹$A_P \sharp* Q'$› ‹$A_Q \sharp* P'$› **have** $(A_P \mathbin{@} A_Q) \sharp* (P' \parallel Q')$ **by** *simp*
moreover from ‹$A_P \sharp* C$› ‹$A_Q \sharp* C$› **have** $(A_P \mathbin{@} A_Q) \sharp* C$ **by** *simp*
moreover note S ‹*distinctPerm p*› ‹$(bn(p \cdot (¡M(\!|\nu*xvec|\!)\langle N\rangle))) \sharp* (¡M(\!|\nu*xvec|\!)\langle N\rangle)$›
‹$(bn(p \cdot (¡M(\!|\nu*xvec|\!)\langle N\rangle))) \sharp* (P' \parallel Q')$› *bigProp*
‹$A_P \sharp* M$› ‹$A_Q \sharp* M$› ‹$A_P \sharp* xvec$› ‹$A_Q \sharp* xvec$› ‹$A_P \sharp* N$› ‹$A_Q \sharp* N$›

ultimately have $Prop\ C\ \Psi\ (P \parallel Q)\ (p \cdot (¡M(\!|\nu*xvec|\!)\langle N\rangle))\ (p \cdot (P' \parallel Q'))\ (A_P \mathbin{@} A_Q)\ (\Psi_P \otimes \Psi_Q)$
by(*fastforce intro*!: *rAlpha*)
then show *?case* **using** αEq $P'eq$ **by** *simp*
next
case(*cBrClose* $\Psi\ P\ M\ xvec\ N\ P'\ A_P\ \Psi_P\ x\ C\ \alpha\ P''$)
note ‹$\tau \prec (\!|\nu x|\!)((\!|\nu*xvec|\!)P') = \alpha \prec P''$›
moreover have $bn\ (\tau) \sharp* (bn\ \alpha)$ **by** *simp*
moreover have $distinct\ (bn\ (\tau))$ **by** *simp*
moreover note ‹$distinct\ (bn\ \alpha)$›
moreover have $(bn\ (\tau) \sharp* (\tau \prec (\!|\nu x|\!)((\!|\nu*xvec|\!)P')))$ **by** *simp*
moreover from ‹$bn\ \alpha \sharp* subject\ \alpha$› **have** $bn\ \alpha \sharp* (\alpha \prec P'')$ **by** *simp*
ultimately obtain p **where** S: $(set\ p) \subseteq (set(bn\ (\tau))) \times (set(bn(p \cdot (\tau))))$
and αEq: $\alpha = p \cdot (\tau)$ **and** $P'eq$: $P'' = p \cdot ((\!|\nu x|\!)((\!|\nu*xvec|\!)P'))$
and $bn\ (\tau) \sharp* \alpha$ **and** $bn\ (\tau) \sharp* P''$
and $(bn(p \cdot (\tau))) \sharp* (\tau)$ **and** $(bn(p \cdot (\tau))) \sharp* ((\!|\nu x|\!)((\!|\nu*xvec|\!)P'))$

by(*rule residualEq*) *simp*
moreover from *cBrClose* **have** $\bigwedge C.\ Prop\ C\ \Psi\ P\ (¡M(\!|\nu{*}xvec|\!)\langle N\rangle)\ P'\ A_P\ \Psi_P$ **by** *simp*
moreover with *cBrClose* **have** $Prop\ C\ \Psi\ ((\!|\nu x|\!)P)\ (\tau)\ ((\!|\nu x|\!)((\!|\nu{*}xvec|\!)P'))\ (x\#A_P)\ \Psi_P$
by(*simp add*: *rBrClose*)
with *S* **have** $Prop\ C\ \Psi\ ((\!|\nu x|\!)P)\ (p \cdot \tau)\ (p \cdot (\!|\nu x|\!)((\!|\nu{*}xvec|\!)P'))\ (x\#A_P)\ \Psi_P$ **by** *simp*
then show *?case* **using** $\alpha Eq\ P'eq$
by *simp*
next
case(*cOpen* $\Psi\ P\ M\ xvec\ yvec\ N\ P'\ x\ A_P\ \Psi_P\ C\ \alpha\ P''$)
note $\langle M(\!|\nu{*}(xvec@x\#yvec)|\!)\langle N\rangle \prec P' = \alpha \prec P''\rangle$
moreover from $\langle xvec\ \sharp{*}\ \alpha\rangle\ \langle x\ \sharp\ \alpha\rangle\ \langle yvec\ \sharp{*}\ \alpha\rangle$ **have** $(xvec@x\#yvec)\ \sharp{*}\ (bn\ \alpha)$
by *auto*
moreover from $\langle xvec\ \sharp{*}\ yvec\rangle\ \langle x\ \sharp\ xvec\rangle\ \langle x\ \sharp\ yvec\rangle\ \langle distinct\ xvec\rangle\ \langle distinct\ yvec\rangle$
have $distinct(xvec@x\#yvec)$
by(*auto simp add*: *fresh-star-def*) (*simp add*: *fresh-def name-list-supp*)
moreover note $\langle distinct(bn\ \alpha)\rangle$
moreover from $\langle xvec\ \sharp{*}\ M\rangle\ \langle x\ \sharp\ M\rangle\ \langle yvec\ \sharp{*}\ M\rangle$ **have** $(xvec@x\#yvec)\ \sharp{*}\ M$ **by** *auto*
then have $(xvec@x\#yvec)\ \sharp{*}\ (M(\!|\nu{*}(xvec@x\#yvec)|\!)\langle N\rangle \prec P')$ **by** *auto*
moreover from $\langle bn\ \alpha\ \sharp{*}\ subject\ \alpha\rangle$ **have** $bn\ \alpha\ \sharp{*}\ (\alpha \prec P'')$ **by** *simp*
ultimately obtain *p* **where** *S*: $(set\ p) \subseteq (set(xvec@x\#yvec)) \times (set(p \cdot (xvec@x\#yvec)))$ **and** *distinctPerm p*
and αeq: $\alpha = (p \cdot M)(\!|\nu{*}(p \cdot (xvec@x\#yvec))|\!)\langle(p \cdot N)\rangle$ **and** $P'eq$: $P'' = (p \cdot P')$
and *A*: $(xvec@x\#yvec)\ \sharp{*}\ ((p \cdot M)(\!|\nu{*}(p \cdot (xvec@x\#yvec))|\!)\langle(p \cdot N)\rangle)$
and *B*: $(p \cdot (xvec@x\#yvec))\ \sharp{*}\ (M(\!|\nu{*}(xvec@x\#yvec)|\!)\langle N\rangle)$
and *C*: $(p \cdot (xvec@x\#yvec))\ \sharp{*}\ P'$
apply −
apply(*rule residualEq*)
by(*assumption* | *simp*)+

note $\langle \Psi \rhd P \longmapsto M(\!|\nu{*}(xvec@yvec)|\!)\langle N\rangle \prec P'\rangle\ \langle x \in (supp\ N)\rangle$

moreover {
fix *C*
from $\langle xvec\ \sharp{*}\ M\rangle\ \langle yvec\ \sharp{*}\ M\rangle$ **have** $(xvec@yvec)\ \sharp{*}\ M$ **by** *simp*
moreover from $\langle distinct\ xvec\rangle\ \langle distinct\ yvec\rangle\ \langle xvec\ \sharp{*}\ yvec\rangle$ **have** $distinct(xvec@yvec)$
by (*auto simp add*: *fresh-star-def name-list-supp fresh-def*)
ultimately have $Prop\ C\ \Psi\ P\ (M(\!|\nu{*}(xvec@yvec)|\!)\langle N\rangle)\ P'\ A_P\ \Psi_P$ **using** $\langle A_P\ \sharp{*}\ xvec\rangle\ \langle A_P\ \sharp{*}\ yvec\rangle\ \langle A_P\ \sharp{*}\ M\rangle\ \langle A_P\ \sharp{*}\ N\rangle$
by(*fastforce intro*!: *cOpen*)
}
moreover obtain *y*::*name* **where** $y\ \sharp\ \Psi$ **and** $y \neq x$ **and** $y\ \sharp\ P$ **and** $y\ \sharp\ xvec$ **and** $y\ \sharp\ yvec$ **and** $y\ \sharp\ \alpha$ **and** $y\ \sharp\ P'$ **and** $y\ \sharp\ A_P$ **and** $y\ \sharp\ \Psi_P$ **and** $y\ \sharp\ M$ **and** $y\ \sharp\ N$ **and** $y\ \sharp\ C$ **and** $y\ \sharp\ p$
by(*generate-fresh name*) *auto*

moreover note ‹$x \sharp \Psi$› ‹$x \sharp M$› ‹$x \sharp xvec$› ‹$x \sharp yvec$› ‹$xvec \sharp* \Psi$› ‹$xvec \sharp* P$› ‹$xvec \sharp* M$›
‹$yvec \sharp* \Psi$› ‹$yvec \sharp* P$› ‹$yvec \sharp* M$› ‹$yvec \sharp* C$› ‹$x \sharp C$› ‹$xvec \sharp* C$› ‹*distinct* $xvec$› ‹*distinct* $yvec$›
‹*extractFrame* $P = \langle A_P, \Psi_P \rangle$› ‹*distinct* A_P› ‹$x \sharp A_P$› ‹$xvec \sharp* yvec$› ‹$xvec \sharp* \Psi_P$›
‹$A_P \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_P \sharp* M$› ‹$A_P \sharp* xvec$› ‹$A_P \sharp* yvec$› ‹$A_P \sharp* N$› ‹$A_P \sharp* P'$› ‹$A_P \sharp* C$›
ultimately have *Prop* $C\ \Psi\ (\langle\!|\nu x|\!\rangle P)\ (M\langle\!|\nu*(xvec@y\#yvec)|\!\rangle\langle([(x,\ y)] \cdot N)\rangle)\ ([(x,\ y)] \cdot P')\ (x\#A_P)\ \Psi_P$
by(*metis rOpen*)
moreover have $(([(x,\ y)] \cdot p) \cdot [(x,\ y)] \cdot M) = [(x,\ y)] \cdot p \cdot M$
by(*subst perm-compose*[*symmetric*]) *simp*
with ‹$y \sharp M$› ‹$x \sharp \alpha$› αeq ‹$y \sharp p$› ‹$x \sharp M$› **have** D: $(([(x,\ y)] \cdot p) \cdot M) = p \cdot M$
by(*auto simp add*: *eqvts freshChainSimps*)
moreover have $(([(x,\ y)] \cdot p) \cdot [(x,\ y)] \cdot xvec) = [(x,\ y)] \cdot p \cdot xvec$
by(*subst perm-compose*[*symmetric*]) *simp*
with ‹$y \sharp xvec$› ‹$x \sharp \alpha$› αeq ‹$y \sharp p$› ‹$x \sharp xvec$› **have** E: $(([(x,\ y)] \cdot p) \cdot xvec) = p \cdot xvec$
by(*auto simp add*: *eqvts freshChainSimps*)
moreover have $(([(x,\ y)] \cdot p) \cdot [(x,\ y)] \cdot yvec) = [(x,\ y)] \cdot p \cdot yvec$
by(*subst perm-compose*[*symmetric*]) *simp*
with ‹$y \sharp yvec$› ‹$x \sharp \alpha$› αeq ‹$y \sharp p$› ‹$x \sharp yvec$› **have** F: $(([(x,\ y)] \cdot p) \cdot yvec) = p \cdot yvec$
by(*auto simp add*: *eqvts freshChainSimps*)
moreover have $(([(x,\ y)] \cdot p) \cdot [(x,\ y)] \cdot x) = [(x,\ y)] \cdot p \cdot x$
by(*subst perm-compose*[*symmetric*]) *simp*
with ‹$y \neq x$› ‹$y \sharp p$› **have** G: $(([(x,\ y)] \cdot p) \cdot y) = p \cdot x$
apply(*simp add*: *freshChainSimps calc-atm*)
apply(*subgoal-tac* $y \neq p \cdot x$)
apply(*clarsimp*)
using $A\ \alpha eq$
apply(*simp add*: *eqvts*)
apply(*subst fresh-atm*[*symmetric*])
apply(*simp only*: *freshChainSimps*)
by *simp*
moreover have $(([(x,\ y)] \cdot p) \cdot [(x,\ y)] \cdot N) = [(x,\ y)] \cdot p \cdot N$
by(*subst perm-compose*[*symmetric*]) *simp*
with ‹$y \sharp N$› ‹$x \sharp \alpha$› ‹$y \sharp p$› αeq **have** H: $(([(x,\ y)] \cdot p) \cdot [(x,\ y)] \cdot N) = p \cdot N$
by(*auto simp add*: *eqvts freshChainSimps*)
moreover have $(([(x,\ y)] \cdot p) \cdot [(x,\ y)] \cdot P') = [(x,\ y)] \cdot p \cdot P'$
by(*subst perm-compose*[*symmetric*]) *simp*
with ‹$y \sharp P'$› ‹$x \sharp P''$› ‹$y \sharp p$› $P'eq$ **have** I: $(([(x,\ y)] \cdot p) \cdot [(x,\ y)] \cdot P') = p \cdot P'$
by(*auto simp add*: *eqvts freshChainSimps*)
from ‹$y \sharp p$› ‹$y \neq x$› **have** $y \neq p \cdot x$
apply(*subst fresh-atm*[*symmetric*])
apply(*simp only*: *freshChainSimps*)
by *simp*

moreover from S **have** $([(x,\ y)] \cdot set\ p) \subseteq [(x,\ y)] \cdot (set(xvec@x\#yvec) \times set(p \cdot (xvec@x\#yvec)))$
by(*simp*)
with ‹$y \neq p \cdot x$› ‹$(([(x,\ y)] \cdot p) \cdot y) = p \cdot x$› ‹$x \sharp xvec$› ‹$y \sharp xvec$› ‹$x \sharp yvec$› ‹$y \sharp yvec$› ‹$y \sharp p$› ‹$x \sharp \alpha$› αeq **have**
$set([(x,\ y)] \cdot p) \subseteq set(xvec@y\#yvec) \times set(([(x,\ y)] \cdot p) \cdot (xvec@y\#yvec))$
by(*simp add*: *eqvts calc-atm perm-compose*)
moreover note ‹$xvec \sharp* \Psi$› ‹$yvec \sharp* \Psi$› ‹$xvec \sharp* P$› ‹$yvec \sharp* P$› ‹$xvec \sharp* M$› ‹$yvec \sharp* M$›
‹$yvec \sharp* C$› S ‹$distinctPerm\ p$› ‹$x \sharp C$› ‹$xvec \sharp* C$› ‹$xvec \sharp* \Psi_P$› ‹$yvec \sharp* \Psi_P$› ‹$x \sharp \Psi$›
‹$A_P \sharp* xvec$› ‹$x \sharp A_P$› ‹$A_P \sharp* yvec$› ‹$A_P \sharp* M$› ‹$x \sharp xvec$› ‹$x \sharp yvec$› ‹$x \sharp M$› ‹$x \sharp A_P$› ‹$A_P \sharp* N$›
$A\ B\ C\ \alpha eq$ ‹$A_P \sharp* \alpha$› ‹$y \sharp \Psi$› ‹$y \neq x$› ‹$y \sharp P$› ‹$y \sharp M$› ‹$y \sharp \Psi_P$› ‹$y \sharp C$› ‹$xvec \sharp* \alpha$› ‹$x \sharp \alpha$› ‹$yvec \sharp* \alpha$› ‹$y \sharp \alpha$› ‹$A_P \sharp* P$› ‹$A_P \sharp* \Psi$› ‹$y \sharp A_P$› ‹$y \sharp N$› ‹$A_P \sharp* P'$› ‹$y \sharp P'$› ‹$A_P \sharp* C$› $P'eq$
ultimately have $Prop\ C\ \Psi\ (\langle\!|\nu x|\!\rangle P)\ (([(x,\ y)] \cdot p) \cdot (M\langle\!|\nu*(xvec@y\#yvec)|\!\rangle\langle([(x,\ y)] \cdot N)\rangle))\ (([(x,\ y)] \cdot p) \cdot [(x,\ y)] \cdot P')\ (x\#A_P)\ \Psi_P$
apply −
apply(*rule rAlpha*[**where** α=$M\langle\!|\nu*(xvec@y\#yvec)|\!\rangle\langle([(x,\ y)] \cdot N)\rangle$])
apply(*assumption* | *simp*)+
apply(*simp add*: *eqvts*)
apply(*assumption* | *simp add*: *abs-fresh*)+
apply(*simp add*: *fresh-left calc-atm*)
apply(*assumption* | *simp*)+
apply(*simp add*: *fresh-left calc-atm*)
apply(*assumption* | *simp*)+
by(*simp add*: *eqvts fresh-left*)+
with $\alpha eq\ P'eq\ D\ E\ F\ G\ H\ I$ **show** *?case*
by(*simp add*: *eqvts*)
next
case(*cBrOpen* $\Psi\ P\ M\ xvec\ yvec\ N\ P'\ x\ A_P\ \Psi_P\ C\ \alpha\ P''$)
note ‹$¡M\langle\!|\nu*(xvec@x\#yvec)|\!\rangle\langle N\rangle \prec P' = \alpha \prec P''$›
moreover from ‹$xvec \sharp* \alpha$› ‹$x \sharp \alpha$› ‹$yvec \sharp* \alpha$› **have** $(xvec@x\#yvec) \sharp* (bn\ \alpha)$
by *auto*
moreover from ‹$xvec \sharp* yvec$› ‹$x \sharp xvec$› ‹$x \sharp yvec$› ‹*distinct xvec*› ‹*distinct yvec*›
have $distinct(xvec@x\#yvec)$
by(*auto simp add*: *fresh-star-def*) (*simp add*: *fresh-def name-list-supp*)
moreover note ‹$distinct(bn\ \alpha)$›
moreover from ‹$xvec \sharp* M$› ‹$x \sharp M$› ‹$yvec \sharp* M$› **have** $(xvec@x\#yvec) \sharp* M$ **by** *auto*
then have $(xvec@x\#yvec) \sharp* (¡M\langle\!|\nu*(xvec@x\#yvec)|\!\rangle\langle N\rangle \prec P')$ **by** *auto*
moreover from ‹$bn\ \alpha \sharp* subject\ \alpha$› **have** $bn\ \alpha \sharp* (\alpha \prec P'')$ **by** *simp*
ultimately obtain p **where** S: $(set\ p) \subseteq (set(xvec@x\#yvec)) \times (set(p \cdot (xvec@x\#yvec)))$ **and** $distinctPerm\ p$
and αeq: $\alpha = ¡(p \cdot M)\langle\!|\nu*(p \cdot (xvec@x\#yvec))|\!\rangle\langle(p \cdot N)\rangle$ **and** $P'eq$: $P'' = (p \cdot P')$
and A: $(xvec@x\#yvec) \sharp* (¡(p \cdot M)\langle\!|\nu*(p \cdot (xvec@x\#yvec))|\!\rangle\langle(p \cdot N)\rangle)$
and B: $(p \cdot (xvec@x\#yvec)) \sharp* (¡M\langle\!|\nu*(xvec@x\#yvec)|\!\rangle\langle N\rangle)$

and C: $(p \cdot (xvec@x\#yvec)) \sharp* P'$
apply $-$
apply(*rule residualEq*)
by(*assumption* | *simp*)+

note $\langle \Psi \rhd P \longmapsto ¡M(\!|\nu*(xvec@yvec)|\!)\langle N\rangle \prec P'\rangle$ $\langle x \in (supp\ N)\rangle$

moreover {
fix C
from $\langle xvec \sharp* M\rangle$ $\langle yvec \sharp* M\rangle$ **have** $(xvec@yvec) \sharp* M$ **by** *simp*
moreover from $\langle distinct\ xvec\rangle$ $\langle distinct\ yvec\rangle$ $\langle xvec \sharp* yvec\rangle$ **have** $distinct(xvec@yvec)$
by *auto* (*simp add*: *fresh-star-def name-list-supp fresh-def*)
ultimately have *Prop* $C\ \Psi\ P\ (¡M(\!|\nu*(xvec@yvec)|\!)\langle N\rangle)\ P'\ A_P\ \Psi_P$ **using** $\langle A_P \sharp* xvec\rangle$ $\langle A_P \sharp* yvec\rangle$ $\langle A_P \sharp* M\rangle$ $\langle A_P \sharp* N\rangle$
by(*fastforce intro*!: *cBrOpen*)
}
moreover obtain y::*name* **where** $y \sharp \Psi$ **and** $y \neq x$ **and** $y \sharp P$ **and** $y \sharp xvec$ **and** $y \sharp yvec$ **and** $y \sharp \alpha$ **and** $y \sharp P'$ **and** $y \sharp A_P$ **and** $y \sharp \Psi_P$ **and** $y \sharp M$ **and** $y \sharp N$ **and** $y \sharp C$ **and** $y \sharp p$
by(*generate-fresh name*) *auto*
moreover note $\langle x \sharp \Psi\rangle$ $\langle x \sharp M\rangle$ $\langle x \sharp xvec\rangle$ $\langle x \sharp yvec\rangle$ $\langle xvec \sharp* \Psi\rangle$ $\langle xvec \sharp* P\rangle$ $\langle xvec \sharp* M\rangle$
$\langle yvec \sharp* \Psi\rangle$ $\langle yvec \sharp* P\rangle$ $\langle yvec \sharp* M\rangle$ $\langle yvec \sharp* C\rangle$ $\langle x \sharp C\rangle$ $\langle xvec \sharp* C\rangle$ $\langle distinct\ xvec\rangle$ $\langle distinct\ yvec\rangle$
$\langle extractFrame\ P = \langle A_P, \Psi_P\rangle\rangle$ $\langle distinct\ A_P\rangle$ $\langle x \sharp A_P\rangle$ $\langle xvec \sharp* yvec\rangle$ $\langle xvec \sharp* \Psi_P\rangle$
$\langle A_P \sharp* \Psi\rangle$ $\langle A_P \sharp* P\rangle$ $\langle A_P \sharp* M\rangle$ $\langle A_P \sharp* xvec\rangle$ $\langle A_P \sharp* yvec\rangle$ $\langle A_P \sharp* N\rangle$ $\langle A_P \sharp* P'\rangle$ $\langle A_P \sharp* C\rangle$
ultimately have *Prop* $C\ \Psi\ (\!|\nu x|\!)P)\ (¡M(\!|\nu*(xvec@y\#yvec)|\!)\langle([(x,\ y)] \cdot N)\rangle)\ ([(x,\ y)] \cdot P')\ (x\#A_P)\ \Psi_P$
by(*metis rBrOpen*)
moreover have $(([(x,\ y)] \cdot p) \cdot [(x,\ y)] \cdot M) = [(x,\ y)] \cdot p \cdot M$
by(*subst perm-compose*[*symmetric*]) *simp*
with $\langle y \sharp M\rangle$ $\langle x \sharp \alpha\rangle$ αeq $\langle y \sharp p\rangle$ $\langle x \sharp M\rangle$ **have** D: $(([(x,\ y)] \cdot p) \cdot M) = p \cdot M$
by(*auto simp add*: *eqvts freshChainSimps*)
moreover have $(([(x,\ y)] \cdot p) \cdot [(x,\ y)] \cdot xvec) = [(x,\ y)] \cdot p \cdot xvec$
by(*subst perm-compose*[*symmetric*]) *simp*
with $\langle y \sharp xvec\rangle$ $\langle x \sharp \alpha\rangle$ αeq $\langle y \sharp p\rangle$ $\langle x \sharp xvec\rangle$ **have** E: $(([(x,\ y)] \cdot p) \cdot xvec) = p \cdot xvec$
by(*auto simp add*: *eqvts freshChainSimps*)
moreover have $(([(x,\ y)] \cdot p) \cdot [(x,\ y)] \cdot yvec) = [(x,\ y)] \cdot p \cdot yvec$
by(*subst perm-compose*[*symmetric*]) *simp*
with $\langle y \sharp yvec\rangle$ $\langle x \sharp \alpha\rangle$ αeq $\langle y \sharp p\rangle$ $\langle x \sharp yvec\rangle$ **have** F: $(([(x,\ y)] \cdot p) \cdot yvec) = p \cdot yvec$
by(*auto simp add*: *eqvts freshChainSimps*)
moreover have $(([(x,\ y)] \cdot p) \cdot [(x,\ y)] \cdot x) = [(x,\ y)] \cdot p \cdot x$
by(*subst perm-compose*[*symmetric*]) *simp*
with $\langle y \neq x\rangle$ $\langle y \sharp p\rangle$ **have** G: $(([(x,\ y)] \cdot p) \cdot y) = p \cdot x$
apply(*simp add*: *freshChainSimps calc-atm*)

apply(*subgoal-tac* $y \neq p \cdot x$)
apply(*clarsimp*)
using A αeq
apply(*simp add*: *eqvts*)
apply(*subst fresh-atm*[*symmetric*])
apply(*simp only*: *freshChainSimps*)
by *simp*
moreover have $(([(x, y)] \cdot p) \cdot [(x, y)] \cdot N) = [(x, y)] \cdot p \cdot N$
by(*subst perm-compose*[*symmetric*]) *simp*
with ‹$y \sharp N$› ‹$x \sharp \alpha$› ‹$y \sharp p$› αeq **have** H: $(([(x, y)] \cdot p) \cdot [(x, y)] \cdot N) = p \cdot N$
by(*auto simp add*: *eqvts freshChainSimps*)
moreover have $(([(x, y)] \cdot p) \cdot [(x, y)] \cdot P') = [(x, y)] \cdot p \cdot P'$
by(*subst perm-compose*[*symmetric*]) *simp*
with ‹$y \sharp P'$› ‹$x \sharp P''$› ‹$y \sharp p$› $P'eq$ **have** I: $(([(x, y)] \cdot p) \cdot [(x, y)] \cdot P') = p \cdot P'$
by(*auto simp add*: *eqvts freshChainSimps*)
from ‹$y \sharp p$› ‹$y \neq x$› **have** $y \neq p \cdot x$
apply(*subst fresh-atm*[*symmetric*])
apply(*simp only*: *freshChainSimps*)
by *simp*
moreover from S **have** $([(x, y)] \cdot set\ p) \subseteq [(x, y)] \cdot (set(xvec@x\#yvec) \times set(p \cdot (xvec@x\#yvec)))$
by(*simp*)
with ‹$y \neq p \cdot x$› ‹$(([(x, y)] \cdot p) \cdot y) = p \cdot x$› ‹$x \sharp xvec$› ‹$y \sharp xvec$› ‹$x \sharp yvec$› ‹$y \sharp yvec$› ‹$y \sharp p$› ‹$x \sharp \alpha$› αeq **have**
$set([(x, y)] \cdot p) \subseteq set(xvec@y\#yvec) \times set(([(x, y)] \cdot p) \cdot (xvec@y\#yvec))$
by(*simp add*: *eqvts calc-atm perm-compose*)
moreover note ‹$xvec \sharp* \Psi$› ‹$yvec \sharp* \Psi$› ‹$xvec \sharp* P$› ‹$yvec \sharp* P$› ‹$xvec \sharp* M$› ‹$yvec \sharp* M$›
‹$yvec \sharp* C$› S ‹$distinctPerm\ p$› ‹$x \sharp C$› ‹$xvec \sharp* C$› ‹$xvec \sharp* \Psi_P$› ‹$yvec \sharp* \Psi_P$› ‹$x \sharp \Psi$›
‹$A_P \sharp* xvec$› ‹$x \sharp A_P$› ‹$A_P \sharp* yvec$› ‹$A_P \sharp* M$› ‹$x \sharp xvec$› ‹$x \sharp yvec$› ‹$x \sharp M$› ‹$x \sharp A_P$› ‹$A_P \sharp* N$›
A B C αeq ‹$A_P \sharp* \alpha$› ‹$y \sharp \Psi$› ‹$y \neq x$› ‹$y \sharp P$› ‹$y \sharp M$› ‹$y \sharp \Psi_P$› ‹$y \sharp C$› ‹$xvec \sharp* \alpha$› ‹$x \sharp \alpha$› ‹$yvec \sharp* \alpha$› ‹$y \sharp \alpha$› ‹$A_P \sharp* P$› ‹$A_P \sharp* \Psi$› ‹$y \sharp A_P$› ‹$y \sharp N$› ‹$A_P \sharp* P'$› ‹$y \sharp P'$› ‹$A_P \sharp* C$› $P'eq$
ultimately have $Prop\ C\ \Psi\ (([\nu x])P)\ (([(x, y)] \cdot p) \cdot ([M]\!\!\langle \nu*(xvec@y\#yvec)\rangle\!\!\rangle\langle([(x, y)] \cdot N)\rangle))\ (([(x, y)] \cdot p) \cdot [(x, y)] \cdot P')\ (x\#A_P)\ \Psi_P$
apply −
apply(*rule rAlpha*[**where** α=$[M]\!\!\langle \nu*(xvec@y\#yvec)\rangle\!\!\rangle\langle([(x, y)] \cdot N)\rangle$])
apply(*assumption* | *simp*)+
apply(*simp add*: *eqvts*)
apply(*assumption* | *simp add*: *abs-fresh*)+
apply(*simp add*: *fresh-left calc-atm*)
apply(*assumption* | *simp*)+
apply(*simp add*: *fresh-left calc-atm*)
apply(*assumption* | *simp*)+
by(*simp add*: *eqvts fresh-left*)+
with αeq $P'eq$ D E F G H I **show** *?case*

by(*simp add*: *eqvts*)
next
case(*cScope* Ψ P α P' x A_P Ψ_P C α' P'')
note ‹$\alpha \prec (\langle\!|\nu x|\!\rangle P') = \alpha' \prec P''$›
moreover from ‹*bn* α $\sharp*$ α'› **have** *bn* α $\sharp*$ (*bn* α') **by** *auto*
moreover note ‹*distinct* (*bn* α)› ‹*distinct*(*bn* α')›
moreover from ‹*bn* α $\sharp*$ *subject* α› ‹*bn* α' $\sharp*$ *subject* α'›
have *bn* α $\sharp*$ ($\alpha \prec (\langle\!|\nu x|\!\rangle P')$ **and** *bn* α' $\sharp*$ ($\alpha' \prec P''$) **by** *simp+*
ultimately obtain p **where** S: (*set* p) $\subseteq$ (*set*(*bn* α)) $\times$ (*set*(*bn*($p \cdot \alpha$))) **and** *distinctPerm* p
and αEq: $\alpha' = p \cdot \alpha$ **and** $P'eq$: $P'' = p \cdot (\langle\!|\nu x|\!\rangle P')$ **and** (*bn*($p \cdot \alpha$)) $\sharp*$ α
and (*bn*($p \cdot \alpha$)) $\sharp*$ ($\langle\!|\nu x|\!\rangle P'$)
by(*rule residualEq*)

note ‹$\Psi \rhd P \longmapsto \alpha \prec P'$›
moreover from ‹*bn* α $\sharp*$ *subject* α› ‹*distinct*(*bn* α)›
have $\bigwedge C$. *Prop* C Ψ P α P' A_P Ψ_P **by**(*fastforce intro*!: *cScope*)

moreover note ‹$x \sharp \Psi$› ‹$x \sharp \alpha$› ‹*bn* α $\sharp*$ Ψ› ‹*bn* α $\sharp*$ P› ‹*bn* α $\sharp*$ *subject* α› ‹*bn* α $\sharp*$ Ψ_P›
‹$x \sharp C$› ‹*bn* α $\sharp*$ C› ‹*distinct*(*bn* α)› ‹*extractFrame* $P = \langle A_P, \Psi_P\rangle$›
‹*distinct* A_P› ‹$x \sharp A_P$› ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_P \sharp* \alpha$› ‹$A_P \sharp* P'$› ‹$A_P \sharp* C$›
ultimately have *Prop* C Ψ ($\langle\!|\nu x|\!\rangle P$) α ($\langle\!|\nu x|\!\rangle P'$) ($x \# A_P$) Ψ_P
by(*metis rScope*)
with ‹*bn* α $\sharp*$ Ψ› ‹*bn* α $\sharp*$ P› ‹$x \sharp \alpha$› ‹*bn* α $\sharp*$ *subject* α› ‹*bn* α $\sharp*$ C› ‹*bn* α $\sharp*$ (*bn* α')› S ‹*distinctPerm* p› ‹*bn*($p \cdot \alpha$) $\sharp*$ α› ‹*bn*($p \cdot \alpha$) $\sharp*$ ($\langle\!|\nu x|\!\rangle P'$)› ‹$A_P \sharp* \alpha$› ‹$A_P \sharp* \alpha'$› αEq ‹$x \sharp \alpha'$› ‹*bn* α $\sharp*$ Ψ_P› ‹*bn* α $\sharp*$ α'› ‹$x \sharp \Psi$› ‹$A_P \sharp* \Psi$› ‹$x \sharp A_P$› ‹$A_P \sharp* P$› ‹$A_P \sharp* P'$› ‹$x \sharp C$› ‹$A_P \sharp* C$›
have *Prop* C Ψ ($\langle\!|\nu x|\!\rangle P$) ($p \cdot \alpha$) ($p \cdot (\langle\!|\nu x|\!\rangle P')$) ($x \# A_P$) Ψ_P
by(*fastforce intro*!: *rAlpha simp add*: *abs-fresh*)
with αEq $P'eq$ ‹*distinctPerm* p› **show** *?case* **by** *simp*
next
case(*cBang* Ψ P A_P Ψ_P C α P')
then show *?case* **by**(*fastforce intro*!: *rBang*)
qed

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* :: *'f*::*fs-name* $\Rightarrow$ *'b* $\Rightarrow$ (*'a*, *'b*, *'c*) *psi* $\Rightarrow$
'a $\Rightarrow$ *'a* $\Rightarrow$ (*'a*, *'b*, *'c*) *psi* $\Rightarrow$ *name list* $\Rightarrow$ *'b* $\Rightarrow$ *bool*
and C :: *'f*::*fs-name*

assumes *Trans*: $\Psi \rhd P \longmapsto M\langle\!|N|\!\rangle \prec P'$
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*: $\bigwedge \Psi\ M\ K\ xvec\ N\ Tvec\ P\ C.$
$[\![\Psi \vdash M \leftrightarrow K;\ distinct\ xvec;\ set\ xvec \subseteq supp\ N;$
$length\ xvec = length\ Tvec;\ xvec \sharp* \Psi;$
$xvec \sharp* M;\ xvec \sharp* K;\ xvec \sharp* C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (M(\!|\lambda* xvec\ N|\!).P)$
$K\ (N[xvec::=Tvec])\ (P[xvec::=Tvec])\ ([])\ (\mathbf{1})$

and *rCase*: $\bigwedge \Psi\ P\ M\ N\ P'\ \varphi\ Cs\ A_P\ \Psi_P\ C.\ [\![\Psi \rhd P \longmapsto M(\!|N|\!) \prec P';\ extractFrame\ P = \langle A_P, \Psi_P\rangle;\ distinct\ A_P;\ \bigwedge C.\ Prop\ C\ \Psi\ P\ M\ N\ P'\ A_P\ \Psi_P;$
$(\varphi, P) \in set\ Cs;\ \Psi \vdash \varphi;\ guarded\ P;\ \Psi_P \simeq \mathbf{1};$
$(supp\ \Psi_P) = (\{\}::name\ set);$
$A_P \sharp* \Psi;\ A_P \sharp* P;\ A_P \sharp* M;\ A_P \sharp* N;\ A_P \sharp* P';\ A_P \sharp* C]\!] \Longrightarrow Prop\ C\ \Psi\ (Cases\ Cs)\ M\ N\ P'\ ([])\ (\mathbf{1})$

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* Ψ *P M N P′* A_P Ψ_P
using *Trans FrP* ‹*distinct* A_P›
proof(*nominal-induct* Ψ *P Rs*==*M*(|*N*|) $\prec$ *P′* A_P Ψ_P *avoiding*: *C arbitrary*: *P′ rule*: *semanticsFrameInduct*)
case *cAlpha*
then show *?case* **by** (*simp add*: *rAlpha*)
next
case *cInput*
then show *?case* **by**(*auto simp add*: *rInput residualInject*)
next
case *cBrInput*
then show *?case* **by**(*simp add*: *residualInject*)
next
case *cOutput*
then show *?case* **by**(*simp add*: *residualInject*)
next
case *cBrOutput*
then show *?case* **by**(*simp add*: *residualInject*)
next
case *cCase*
then show *?case* **by**(*simp add*: *rCase residualInject*)
next
case *cPar1*
then show *?case* **by**(*auto simp add*: *rPar1 residualInject*)
next
case *cPar2*
then show *?case* **by**(*auto simp add*: *rPar2 residualInject*)
next
case *cComm1*
then show *?case* **by**(*simp add*: *residualInject*)
next
case *cComm2*
then show *?case* **by**(*simp add*: *residualInject*)
next
case *cBrMerge*
then show *?case* **by**(*simp add*: *residualInject*)
next
case *cBrComm1*
then show *?case* **by**(*simp add*: *residualInject*)
next
case *cBrComm2*
then show *?case* **by**(*simp add*: *residualInject*)
next
case *cBrClose*
then show *?case* **by**(*simp add*: *residualInject*)

next
case *cOpen*
then show *?case* **by**(*simp add*: *residualInject*)
next
case *cBrOpen*
then show *?case* **by**(*simp add*: *residualInject*)
next
case *cScope*
then show *?case* **by**(*auto simp add*: *rScope residualInject*)
next
case *cBang*
then show *?case* **by**(*simp add*: *rBang residualInject*)
qed

lemma *brinputFrameInduct*[*consumes 3*, *case-names cAlpha cBrInput cCase cPar1 cPar2 cBrMerge 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* :: $'f$::*fs-name* $\Rightarrow$ $'b \Rightarrow ('a, 'b, 'c)$ *psi* $\Rightarrow$
$'a \Rightarrow 'a \Rightarrow ('a, 'b, 'c)$ *psi* $\Rightarrow$ *name list* $\Rightarrow 'b \Rightarrow$ *bool*
and C :: $'f$::*fs-name*

assumes *Trans*: $\Psi \rhd P \longmapsto ¿ M(\!|N|\!) \prec P'$
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$$]\!] \Longrightarrow$ *Prop* $C\ \Psi\ P\ M\ N\ P'\ (p \cdot A_P)\ (p \cdot \Psi_P)$
and *rBrInput*: $\bigwedge \Psi\ M\ K\ xvec\ N\ Tvec\ P\ C$.
$[\![\Psi \vdash K \succeq M$; *distinct xvec*; *set xvec* $\subseteq$ *supp* N;
length xvec = *length Tvec*; *xvec* $\sharp*$ Ψ;
xvec $\sharp*$ M; *xvec* $\sharp*$ K; *xvec* $\sharp*$ $C$$]\!] \Longrightarrow$
Prop $C\ \Psi\ (M(\!|\lambda* xvec\ N|\!).P)$
$K\ (N[xvec ::= Tvec])\ (P[xvec ::= Tvec])\ ([])\ (\mathbf{1})$
and *rCase*: $\bigwedge \Psi\ P\ M\ N\ P'\ \varphi\ Cs\ A_P\ \Psi_P\ C$. $[\![\Psi \rhd P \longmapsto ¿ M(\!|N|\!) \prec P'$; *extractFrame* $P = \langle A_P, \Psi_P \rangle$; *distinct* A_P; $\bigwedge C$. *Prop* $C\ \Psi\ P\ M\ N\ P'\ A_P\ \Psi_P$;
$(\varphi, P) \in$ *set* 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 *rBrMerge*: $\bigwedge \Psi\ \Psi_Q\ P\ M\ N\ P'\ A_P\ \Psi_P\ Q\ Q'\ A_Q\ C.$
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P';\ \bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_Q)\ P\ M\ N\ P'\ A_P\ \Psi_P;$
$extractFrame\ P = \langle A_P, \Psi_P\rangle;\ distinct\ A_P;$
$\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q';\ \bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_P)\ Q\ M\ N\ Q'\ A_Q\ \Psi_Q;$
$extractFrame\ Q = \langle A_Q, \Psi_Q\rangle;\ distinct\ A_Q;$
$A_P \sharp* \Psi;\ A_P \sharp* \Psi_Q;\ A_P \sharp* P;\ A_P \sharp* N;\ A_P \sharp* P';$
$A_P \sharp* Q;\ A_P \sharp* Q';\ A_P \sharp* A_Q;\ A_P \sharp* M;\ A_Q \sharp* M;$
$A_Q \sharp* \Psi;\ A_Q \sharp* \Psi_P;\ A_Q \sharp* P;\ A_Q \sharp* N;\ A_Q \sharp* P';$
$A_Q \sharp* Q;\ A_Q \sharp* Q';\ A_P \sharp* C;\ A_Q \sharp* C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (P \parallel Q)\ M\ N\ (P' \parallel Q')\ (A_P @ A_Q)\ (\Psi_P \otimes \Psi_Q)$

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 $Prop\ C\ \Psi\ P\ M\ N\ P'\ A_P\ \Psi_P$

using *Trans FrP* ‹*distinct* A_P›

proof(*nominal-induct* $\Psi\ P\ Rs==¿M(\!|N|\!) \prec P'\ A_P\ \Psi_P$ *avoiding*: *C arbitrary*: P' *rule*: *semanticsFrameInduct*)

case *cAlpha*

then show *?case* **by** (*simp add*: *rAlpha*)
next
case *cInput*
then show *?case* **by**(*simp add*: *residualInject*)
next
case *cBrInput*
then show *?case* **by**(*auto simp add*: *rBrInput residualInject*)
next
case *cOutput*
then show *?case* **by**(*simp add*: *residualInject*)
next
case *cBrOutput*
then show *?case* **by**(*simp add*: *residualInject*)
next
case *cCase*
then show *?case* **by**(*simp add*: *rCase residualInject*)
next
case *cPar1*
then show *?case* **by**(*auto simp add*: *rPar1 residualInject*)
next
case *cPar2*
then show *?case* **by**(*auto simp add*: *rPar2 residualInject*)
next
case *cComm1*
then show *?case* **by**(*simp add*: *residualInject*)
next
case *cComm2*
then show *?case* **by**(*simp add*: *residualInject*)
next
case *cBrMerge*
then show *?case* **by**(*auto simp add*: *rBrMerge residualInject*)
next
case *cBrComm1*
then show *?case* **by**(*simp add*: *residualInject*)
next
case *cBrComm2*
then show *?case* **by**(*simp add*: *residualInject*)
next
case *cBrClose*
then show *?case* **by**(*simp add*: *residualInject*)
next
case *cOpen*
then show *?case* **by**(*simp add*: *residualInject*)
next
case *cBrOpen*
then show *?case* **by**(*simp add*: *residualInject*)
next
case *cScope*
then show *?case* **by**(*auto simp add*: *rScope residualInject*)

next
case *cBang*
then show *?case* **by**(*simp add*: *rBang residualInject*)
qed

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* :: *'f*::*fs-name* ⇒ *'b* ⇒ (*'a*, *'b*, *'c*) *psi* ⇒
'a ⇒ (*'a*, *'b*, *'c*) *boundOutput* ⇒ *name list* ⇒ *'b* ⇒ *bool*
and C :: *'f*::*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*: $\bigwedge \Psi\ P\ M\ B\ \varphi\ Cs\ A_P\ \Psi_P\ C.\ [\![\Psi \rhd P \longmapsto (ROut\ M\ B);\ extractFrame\ P = \langle A_P, \Psi_P\rangle;\ distinct\ A_P;\ \bigwedge C.\ Prop\ C\ \Psi\ P\ M\ B\ A_P\ \Psi_P;$
$(\varphi, P) \in set\ Cs;\ \Psi \vdash \varphi;\ guarded\ P;\ \Psi_P \simeq \mathbf{1};\ (supp\ \Psi_P) = (\{\}::name\ set);$
$A_P \sharp* \Psi;\ A_P \sharp* P;\ A_P \sharp* M;\ A_P \sharp* B;\ A_P \sharp* C]\!] \Longrightarrow Prop\ C\ \Psi\ (Cases\ Cs)\ M\ B\ ([])\ (\mathbf{1})$
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 Ψ_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 Ψ $(P \parallel Q)$ 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 Ψ_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 Ψ P M $((\!|\nu * (xvec @ yvec)|\!)N \prec' P')$ A_P Ψ_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 Ψ $((\!|\nu x|\!)P)$ M $((\!|\nu * (xvec @ x \# yvec)|\!)N \prec' P')$ $(x \# A_P)$ Ψ_P

and *rScope*: $\bigwedge \Psi$ P M *xvec* N P' x A_P Ψ_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 Ψ P M $((\!|\nu * xvec|\!)N \prec' P')$ A_P Ψ_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 Ψ $((\!|\nu x|\!)P)$ M $((\!|\nu * xvec|\!)N \prec' ((\!|\nu x|\!)P'))$ $(x \# A_P)$ Ψ_P

and *rBang*: $\bigwedge \Psi$ P M B A_P Ψ_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 Ψ $(P \parallel !P)$ 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 Ψ $(!P)$ M B $([])$ $(\mathbf{1})$

shows *Prop* C Ψ P M B A_P Ψ_P
proof −
{
fix B
assume $\Psi \rhd P \longmapsto ROut\ M\ B$
then have *Prop* C Ψ P M B A_P Ψ_P **using** *FrP* ‹*distinct* A_P›
proof(*nominal-induct* Ψ P *Rs*==*ROut* M B A_P Ψ_P *avoiding*: C *arbitrary*: B *rule*: *semanticsFrameInduct*)
case *cAlpha*
then show *?case* **by**(*auto intro*: *rAlpha*)
next

```
  case cInput
  then show ?case by(simp add: residualInject)
next
  case cBrInput
  then show ?case by(simp add: residualInject)
next
  case cOutput
  then show ?case by(force intro: rOutput simp add: residualInject)
next
  case cBrOutput
  then show ?case by(simp add: residualInject)
next
  case cCase
  then show ?case by(force intro: rCase simp add: residualInject)
next
  case cPar1
  then show ?case
    by(auto intro!: rPar1 simp add: residualInject)
next
  case cPar2
  then show ?case
    by(auto intro!: rPar2 simp add: residualInject)
next
  case cComm1
  then show ?case by(simp add: residualInject)
next
  case cComm2
  then show ?case by(simp add: residualInject)
next
  case cBrMerge
  then show ?case by(simp add: residualInject)
next
  case cBrComm1
  then show ?case by(simp add: residualInject)
next
  case cBrComm2
  then show ?case by(simp add: residualInject)
next
  case cBrClose
  then show ?case by(simp add: residualInject)
next
  case cOpen
  then show ?case by(auto intro: rOpen simp add: residualInject)
next
  case cBrOpen
  then show ?case by(simp add: residualInject)
next
  case cScope
  then show ?case by(force intro: rScope simp add: residualInject)
```

next
case *cBang*
then show *?case* **by**(*force intro*: *rBang simp add*: *residualInject*)
qed
}
with *Trans* **show** *?thesis* **by**(*simp add*: *residualInject*)
qed

lemma *broutputFrameInduct*[*consumes 3*, *case-names cAlpha cBrOutput cCase cPar1 cPar2 cBrComm1 cBrComm2 cBrOpen 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* :: *'f*::*fs-name* ⇒ *'b* ⇒ (*'a*, *'b*, *'c*) *psi* ⇒
'a ⇒ (*'a*, *'b*, *'c*) *boundOutput* ⇒ *name list* ⇒ *'b* ⇒ *bool*
and C :: *'f*::*fs-name*

assumes *Trans*: $\Psi \rhd P \longmapsto RBrOut\ M\ B$
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 *rBrOutput*: $\bigwedge \Psi\ M\ K\ N\ P\ C.\ \Psi \vdash M \preceq K \Longrightarrow Prop\ C\ \Psi\ (M\langle N\rangle.P)\ K\ (N \prec' P)\ ([])\ (\mathbf{1})$
and *rCase*: $\bigwedge \Psi\ P\ M\ B\ \varphi\ Cs\ A_P\ \Psi_P\ C.\ [\![\Psi \rhd P \longmapsto (RBrOut\ M\ B);\ extractFrame\ P = \langle A_P, \Psi_P\rangle;\ distinct\ A_P;\ \bigwedge C.\ Prop\ C\ \Psi\ P\ M\ B\ A_P\ \Psi_P;$
$(\varphi, P) \in set\ Cs;\ \Psi \vdash \varphi;\ guarded\ P;\ \Psi_P \simeq \mathbf{1};\ (supp\ \Psi_P) = (\{\}::name\ set);$
$A_P \sharp* \Psi;\ A_P \sharp* P;\ A_P \sharp* M;\ A_P \sharp* B;\ A_P \sharp* C]\!] \Longrightarrow Prop\ C\ \Psi\ (Cases\ Cs)\ M\ B\ ([])\ (\mathbf{1})$
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 *rBrComm1*: $\bigwedge \Psi\ \Psi_Q\ P\ M\ N\ P'\ A_P\ \Psi_P\ Q\ xvec\ Q'\ A_Q\ C.$
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P';\ extractFrame\ P = \langle A_P, \Psi_P\rangle;\ distinct\ A_P;$
$\Psi \otimes \Psi_P \rhd Q \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec Q';\ extractFrame\ Q = \langle A_Q, \Psi_Q\rangle;\ distinct\ A_Q;$
$distinct\ xvec;$
$\bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_P)\ Q\ M\ ((\!|\nu * xvec|\!)N \prec' Q')\ A_Q\ \Psi_Q;$
$A_P \sharp * \Psi;\ A_P \sharp * \Psi_Q;\ A_P \sharp * P;\ A_P \sharp * N;\ A_P \sharp * P';$
$A_P \sharp * Q;\ A_P \sharp * Q';\ A_P \sharp * A_Q;\ A_P \sharp * xvec;\ A_Q \sharp * \Psi;\ A_Q \sharp * \Psi_P;$
$A_Q \sharp * P;\ A_Q \sharp * N;\ A_Q \sharp * P';\ A_Q \sharp * Q;\ A_Q \sharp * Q';$
$A_Q \sharp * xvec;\ xvec \sharp * \Psi;\ xvec \sharp * \Psi_P;\ xvec \sharp * \Psi_Q;\ xvec \sharp * P;$
$xvec \sharp * Q;\ A_P \sharp * C;\ A_Q \sharp * C;\ xvec \sharp * C;$
$A_P \sharp * M;\ A_Q \sharp * M;\ xvec \sharp * M]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (P \parallel Q)\ M\ ((\!|\nu * xvec|\!)N \prec' (P' \parallel Q'))\ (A_P @ A_Q)\ (\Psi_P \otimes \Psi_Q)$

and *rBrComm2*: $\bigwedge \Psi\ \Psi_Q\ P\ M\ xvec\ N\ P'\ A_P\ \Psi_P\ Q\ Q'\ A_Q\ C.$
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec P';\ extractFrame\ P = \langle A_P, \Psi_P\rangle;\ distinct\ A_P;$
$\bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_Q)\ P\ M\ ((\!|\nu * xvec|\!)N \prec' P')\ A_P\ \Psi_P;$
$\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q';\ extractFrame\ Q = \langle A_Q, \Psi_Q\rangle;\ distinct\ A_Q;$
$distinct\ xvec;$
$A_P \sharp * \Psi;\ A_P \sharp * \Psi_Q;\ A_P \sharp * P;\ A_P \sharp * N;\ A_P \sharp * P';$
$A_P \sharp * Q;\ A_P \sharp * Q';\ A_P \sharp * A_Q;\ A_P \sharp * xvec;\ A_Q \sharp * \Psi;\ A_Q \sharp * \Psi_P;$
$A_Q \sharp * P;\ A_Q \sharp * N;\ A_Q \sharp * P';\ A_Q \sharp * Q;\ A_Q \sharp * Q';$
$A_Q \sharp * xvec;\ xvec \sharp * \Psi;\ xvec \sharp * \Psi_P;\ xvec \sharp * \Psi_Q;\ xvec \sharp * P;$
$xvec \sharp * Q;\ A_P \sharp * C;\ A_Q \sharp * C;\ xvec \sharp * C;$
$A_P \sharp * M;\ A_Q \sharp * M;\ xvec \sharp * M]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (P \parallel Q)\ M\ ((\!|\nu * xvec|\!)N \prec' (P' \parallel Q'))\ (A_P @ A_Q)\ (\Psi_P \otimes \Psi_Q)$

and *rBrOpen*: $\bigwedge \Psi\ P\ M\ xvec\ yvec\ N\ P'\ x\ A_P\ \Psi_P\ C.$
$[\![\Psi \rhd P \longmapsto ¡M(\!|\nu * (xvec @ yvec)|\!)\langle N\rangle \prec P';\ extractFrame\ P = \langle A_P, \Psi_P\rangle;\ distinct\ A_P;$
$\bigwedge C.\ Prop\ C\ \Psi\ P\ M\ ((\!|\nu * (xvec @ yvec)|\!)N \prec' P')\ A_P\ \Psi_P;\ x \in supp\ N;\ x \sharp \Psi;\ x \sharp M;$
$x \sharp A_P;\ x \sharp xvec;\ x \sharp yvec;\ A_P \sharp * \Psi;\ A_P \sharp * P;\ A_P \sharp * M;\ A_P \sharp *$

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*: $\bigwedge \Psi\ P\ M\ B\ A_P\ \Psi_P\ C.$
$[\![\Psi \rhd P \parallel\ !P \longmapsto RBrOut\ M\ B$; $guarded\ P$; $extractFrame\ P = \langle A_P, \Psi_P\rangle$; $distinct\ A_P$;
$\bigwedge C.\ Prop\ C\ \Psi\ (P \parallel\ !P)\ M\ B\ A_P\ (\Psi_P \otimes \mathbf{1})$; $\Psi_P \simeq \mathbf{1}$; $supp\ \Psi_P = (\{\}::name\ set)$;
$A_P \sharp* \Psi$; $A_P \sharp* P$; $A_P \sharp* M$; $A_P \sharp* C]\!] \Longrightarrow Prop\ C\ \Psi\ (!P)\ M\ B\ ([])\ (\mathbf{1})$

shows $Prop\ C\ \Psi\ P\ M\ B\ A_P\ \Psi_P$
proof −
{
fix B
assume $\Psi \rhd P \longmapsto RBrOut\ M\ B$
then have $Prop\ C\ \Psi\ P\ M\ B\ A_P\ \Psi_P$ **using** $FrP\ \langle distinct\ A_P\rangle$
proof(*nominal-induct* $\Psi\ P\ Rs==RBrOut\ M\ B\ A_P\ \Psi_P$ *avoiding*: C *arbitrary*: B *rule*: *semanticsFrameInduct*)
case *cAlpha*
then show *?case* **by**(*auto intro*: *rAlpha*)
next
case *cInput*
then show *?case* **by**(*simp add*: *residualInject*)
next
case *cBrInput*
then show *?case* **by**(*simp add*: *residualInject*)
next
case *cOutput*
then show *?case* **by**(*simp add*: *residualInject*)
next
case *cBrOutput*
then show *?case* **by**(*force intro*: *rBrOutput simp add*: *residualInject*)
next
case *cCase*
then show *?case* **by**(*force intro*: *rCase simp add*: *residualInject*)
next
case *cPar1*

```
    then show ?case by(auto intro!: rPar1 simp add: residualInject)
  next
    case cPar2
    then show ?case by(auto intro!: rPar2 simp add: residualInject)
  next
    case cComm1
    then show ?case by(simp add: residualInject)
  next
    case cComm2
    then show ?case by(simp add: residualInject)
  next
    case cBrMerge
    then show ?case by(simp add: residualInject)
  next
    case cBrComm1
    then show ?case by(auto intro: rBrComm1 simp add: residualInject)
  next
    case cBrComm2
    then show ?case by(auto intro: rBrComm2 simp add: residualInject)
  next
    case cBrClose
    then show ?case by(simp add: residualInject)
  next
    case cOpen
    then show ?case by(simp add: residualInject)
  next
    case cBrOpen
    then show ?case by(auto intro: rBrOpen simp add: residualInject)
  next
    case cScope
    then show ?case by(force intro: rScope simp add: residualInject)
  next
    case cBang
    then show ?case by(force intro: rBang simp add: residualInject)
  qed
 }
 with Trans show ?thesis by(simp add: residualInject)
qed

lemma tauFrameInduct[consumes 3, case-names cAlpha cCase cPar1 cPar2 cComm1
cComm2 cBrClose cScope cBang]:
 fixes Ψ    :: 'b
   and P    :: ('a, 'b, 'c) psi
   and P'   :: ('a, 'b, 'c) psi
   and Prop :: 'f::fs-name ⇒ 'b ⇒ ('a, 'b, 'c) psi ⇒
               ('a, 'b, 'c) psi ⇒ name list ⇒ 'b ⇒ bool
   and C    :: 'f::fs-name

assumes Trans: Ψ ▷ P ⟼τ ≺ P'
```

and *FrP*: $\mathit{extractFrame}\ P = \langle A_P, \Psi_P\rangle$
and $\mathit{distinct}\ A_P$
and *rAlpha*: $\bigwedge \Psi\ P\ P'\ A_P\ \Psi_P\ p\ C.\ [\![A_P \sharp* \Psi;\ A_P \sharp* P;\ A_P \sharp* P';\ A_P \sharp* (p \cdot A_P);\ A_P \sharp* C;$
$\mathit{set}\ p \subseteq \mathit{set}\ A_P \times \mathit{set}\ (p \cdot A_P);\ \mathit{distinctPerm}\ p;$
$\mathit{Prop}\ C\ \Psi\ P\ P'\ A_P\ \Psi_P]\!] \Longrightarrow \mathit{Prop}\ C\ \Psi\ P\ P'\ (p \cdot A_P)\ (p \cdot \Psi_P)$
and *rCase*: $\bigwedge \Psi\ P\ P'\ \varphi\ \mathit{Cs}\ A_P\ \Psi_P\ C.\ [\![\Psi \rhd P \longmapsto \tau \prec P';\ \mathit{extractFrame}\ P = \langle A_P, \Psi_P\rangle;\ \mathit{distinct}\ A_P;\ \bigwedge C.\ \mathit{Prop}\ C\ \Psi\ P\ P'\ A_P\ \Psi_P;$
$(\varphi, P) \in \mathit{set}\ \mathit{Cs};\ \Psi \vdash \varphi;\ \mathit{guarded}\ P;\ \ \Psi_P \simeq \mathbf{1};\ (\mathit{supp}\ \Psi_P) = (\{\}::\mathit{name}\ \mathit{set});$
$A_P \sharp* \Psi;\ A_P \sharp* P;\ A_P \sharp* P';\ A_P \sharp* C]\!] \Longrightarrow \mathit{Prop}\ C\ \Psi\ (\mathit{Cases}\ \mathit{Cs})\ P'\ ([])\ (\mathbf{1})$
and *rPar1*: $\bigwedge \Psi\ \Psi_Q\ P\ P'\ A_Q\ Q\ A_P\ \Psi_P\ C.$
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto \tau \prec P';$
$\mathit{extractFrame}\ P = \langle A_P, \Psi_P\rangle;\ \mathit{distinct}\ A_P;$
$\mathit{extractFrame}\ Q = \langle A_Q, \Psi_Q\rangle;\ \mathit{distinct}\ A_Q;$
$\bigwedge C.\ \mathit{Prop}\ C\ (\Psi \otimes \Psi_Q)\ P\ P'\ A_P\ \Psi_P;$
$A_P \sharp* P;\ A_P \sharp* Q;\ A_P \sharp* \Psi;\ A_P \sharp* P';\ A_P \sharp* A_Q;\ A_P \sharp* \Psi_Q;$
$A_Q \sharp* P;\ A_Q \sharp* Q;\ A_Q \sharp* \Psi;\ A_Q \sharp* P';\ A_Q \sharp* \Psi_P;$
$A_P \sharp* C;\ A_Q \sharp* C]\!] \Longrightarrow$
$\mathit{Prop}\ C\ \Psi\ (P \parallel Q)\ (P' \parallel Q)\ (A_P @ A_Q)\ (\Psi_P \otimes \Psi_Q)$
and *rPar2*: $\bigwedge \Psi\ \Psi_P\ Q\ Q'\ A_P\ P\ A_Q\ \Psi_Q\ C.$
$[\![\Psi \otimes \Psi_P \rhd Q \longmapsto \tau \prec Q';$
$\mathit{extractFrame}\ P = \langle A_P, \Psi_P\rangle;\ \mathit{distinct}\ A_P;$
$\mathit{extractFrame}\ Q = \langle A_Q, \Psi_Q\rangle;\ \mathit{distinct}\ A_Q;$
$\bigwedge C.\ \mathit{Prop}\ C\ (\Psi \otimes \Psi_P)\ Q\ Q'\ A_Q\ \Psi_Q;$
$A_P \sharp* P;\ A_P \sharp* Q;\ A_P \sharp* \Psi;\ A_P \sharp* Q';\ A_P \sharp* A_Q;\ A_P \sharp* \Psi_Q;$
$A_Q \sharp* P;\ A_Q \sharp* Q;\ A_Q \sharp* \Psi;\ A_Q \sharp* Q';\ A_Q \sharp* \Psi_P;$
$A_P \sharp* C;\ A_Q \sharp* C]\!] \Longrightarrow$
$\mathit{Prop}\ C\ \Psi\ (P \parallel Q)\ (P \parallel Q')\ (A_P @ A_Q)\ (\Psi_P \otimes \Psi_Q)$
and *rComm1*: $\bigwedge \Psi\ \Psi_Q\ P\ M\ N\ P'\ A_P\ \Psi_P\ Q\ K\ \mathit{xvec}\ Q'\ A_Q\ C.$
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|N|\!) \prec P';\ \mathit{extractFrame}\ P = \langle A_P, \Psi_P\rangle;\ \mathit{distinct}\ A_P;$
$\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|\nu*\mathit{xvec}|\!)\langle N\rangle \prec Q';\ \mathit{extractFrame}\ Q = \langle A_Q, \Psi_Q\rangle;\ \mathit{distinct}\ A_Q;$
$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K;\ \mathit{distinct}\ \mathit{xvec};$
$A_P \sharp* \Psi;\ A_P \sharp* \Psi_Q;\ A_P \sharp* P;\ A_P \sharp* M;\ A_P \sharp* N;\ A_P \sharp* P';$
$A_P \sharp* Q;\ A_P \sharp* Q';\ A_P \sharp* A_Q;\ A_P \sharp* \mathit{xvec};\ A_Q \sharp* \Psi;\ A_Q \sharp* \Psi_P;$
$A_Q \sharp* P;\ A_Q \sharp* N;\ A_Q \sharp* P';\ A_Q \sharp* Q;\ A_Q \sharp* K;\ A_Q \sharp* Q';$
$A_Q \sharp* \mathit{xvec};\ \mathit{xvec} \sharp* \Psi;\ \mathit{xvec} \sharp* \Psi_P;\ \mathit{xvec} \sharp* \Psi_Q;\ \mathit{xvec} \sharp* P;\ \mathit{xvec} \sharp* M;$
$\mathit{xvec} \sharp* Q;\ \mathit{xvec} \sharp* K;\ A_P \sharp* C;\ A_Q \sharp* C;\ \mathit{xvec} \sharp* C]\!] \Longrightarrow$
$\mathit{Prop}\ C\ \Psi\ (P \parallel Q)\ ((\!|\nu*\mathit{xvec}|\!)(P' \parallel Q'))\ (A_P @ A_Q)\ (\Psi_P \otimes \Psi_Q)$
and *rComm2*: $\bigwedge \Psi\ \Psi_Q\ P\ M\ \mathit{xvec}\ N\ P'\ A_P\ \Psi_P\ Q\ K\ Q'\ A_Q\ C.$
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|\nu*\mathit{xvec}|\!)\langle N\rangle \prec P';\ \mathit{extractFrame}\ P = \langle A_P, \Psi_P\rangle;\ \mathit{distinct}\ A_P;$
$\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|N|\!) \prec Q';\ \mathit{extractFrame}\ Q = \langle A_Q, \Psi_Q\rangle;\ \mathit{distinct}\ A_Q;$

$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K$; *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 Ψ $(P \parallel Q)$ $((\!|\nu*xvec|\!)(P' \parallel Q'))$ $(A_P @ A_Q)$ $(\Psi_P \otimes \Psi_Q)$

and *rBrClose*: $\bigwedge \Psi$ P M *xvec* N P' A_P Ψ_P x C.
$\llbracket \Psi \rhd P \longmapsto {¡}M(\!|\nu*xvec|\!)\langle N\rangle \prec P'$;
$x \in$ *supp* M;
extractFrame $P = \langle A_P, \Psi_P\rangle$; *distinct* A_P;
$A_P \sharp* \Psi$; $A_P \sharp* P$; $A_P \sharp* M$; $A_P \sharp* N$; $A_P \sharp* P'$; $A_P \sharp* xvec$;
distinct xvec; $xvec \sharp* \Psi$; $xvec \sharp* \Psi_P$; $xvec \sharp* P$;
$xvec \sharp* M$;
$x \sharp \Psi$; $x \sharp xvec$; $x \sharp A_P$;
$A_P \sharp* C$; $xvec \sharp* C$; $x \sharp C\rrbracket \Longrightarrow$
Prop C Ψ $((\!|\nu x|\!)P)$ $((\!|\nu x|\!)((\!|\nu*xvec|\!)P'))$ $(x\#A_P)$ Ψ_P

and *rScope*: $\bigwedge \Psi$ P P' x A_P Ψ_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 Ψ P P' A_P Ψ_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 Ψ $((\!|\nu x|\!)P)$ $((\!|\nu x|\!)P')$ $(x\#A_P)$ Ψ_P

and *rBang*: $\bigwedge \Psi$ P P' A_P Ψ_P C.
$\llbracket \Psi \rhd P \parallel !P \longmapsto \tau \prec P'$; *guarded* P; *extractFrame* $P = \langle A_P, \Psi_P\rangle$; *distinct* A_P;
$\bigwedge C$. *Prop* C Ψ $(P \parallel !P)$ P' A_P $(\Psi_P \otimes \mathbf{1})$; $\Psi_P \simeq \mathbf{1}$; *supp* Ψ_P = ({}::*name set*);
$A_P \sharp* \Psi$; $A_P \sharp* P$; $A_P \sharp* P'$; $A_P \sharp* C\rrbracket \Longrightarrow$ *Prop* C Ψ $(!P)$ P' ([]) (**1**)

shows *Prop* C Ψ P P' A_P Ψ_P
using *Trans FrP* ‹*distinct* A_P›
proof(*nominal-induct* Ψ P *Rs*==$\tau \prec P'$ A_P Ψ_P *avoiding*: C *arbitrary*: P' *rule*: *semanticsFrameInduct*)
case *cAlpha*
then show *?case* **by**(*force intro*: *rAlpha simp add*: *residualInject*)
next
case *cInput*
then show *?case* **by**(*simp add*: *residualInject*)
next
case *cBrInput*
then show *?case* **by**(*simp add*: *residualInject*)
next
case *cOutput*
then show *?case* **by**(*simp add*: *residualInject*)
next
case *cBrOutput*
then show *?case* **by**(*simp add*: *residualInject*)

```
next
  case cCase
  then show ?case by(force intro: rCase simp add: residualInject)
next
  case cPar1
  then show ?case by(force intro: rPar1 simp add: residualInject)
next
  case cPar2
  then show ?case by(force intro: rPar2 simp add: residualInject)
next
  case cComm1
  then show ?case by(force intro: rComm1 simp add: residualInject)
next
  case cComm2
  then show ?case by(force intro: rComm2 simp add: residualInject)
next
  case cBrMerge
  then show ?case by(simp add: residualInject)
next
  case cBrComm1
  then show ?case by(simp add: residualInject)
next
  case cBrComm2
  then show ?case by(simp add: residualInject)
next
  case cBrClose
  then show ?case by(force intro: rBrClose simp add: residualInject)
next
  case cOpen
  then show ?case by(simp add: residualInject)
next
  case cBrOpen
  then show ?case by(simp add: residualInject)
next
  case cScope
  then show ?case by(force intro: rScope simp add: residualInject)
next
  case cBang
  then show ?case by(force intro: rBang simp add: residualInject)
qed

lemma inputFreshDerivative:
  fixes Ψ  :: '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 −
have $bn(M(\!|N|\!)) \sharp* subject(M(\!|N|\!))$ **and** $distinct(bn(M(\!|N|\!)))$ **by** *simp+*
with ‹$\Psi \rhd P \longmapsto M(\!|N|\!) \prec P'$› **show** *?thesis* **using** ‹$x \sharp P$› ‹$x \sharp N$›
proof(*nominal-induct* Ψ P α==$M(\!|N|\!)$ P' *avoiding*: x *rule*: *semanticsInduct*)
case(*cAlpha* Ψ P α P' p x)
then show *?case* **by** *simp*
next
case(*cInput* Ψ M' K *xvec* N' *Tvec* P x)
from ‹$K(\!|(N'[xvec::=Tvec])|\!) = M(\!|N|\!)$› **have** $M = K$ **and** *NeqN'*: $N = N'[xvec::=Tvec]$ **by**(*simp add*: *action.inject*)+
note ‹*length xvec* = *length Tvec*› ‹*distinct xvec*› **then**
moreover have $x \sharp Tvec$ **using** ‹*set xvec* ⊆ *supp* N'› ‹$x \sharp N$› *NeqN'*
by(*blast intro*: *substTerm.subst3*)
moreover from ‹*xvec* $\sharp*$ x› ‹$x \sharp M'(\!|\lambda*xvec\ N'|\!).P$›
have $x \sharp P$ **by**(*simp add*: *inputChainFresh*) (*simp add*: *name-list-supp fresh-def*)
ultimately show *?case* **using** ‹*xvec* $\sharp*$ x› **by** *auto*
next
case *cBrInput*
then show *?case* **by** *simp*
next
case(*cOutput* Ψ M K N P x)
then show *?case* **by** *simp*
next
case *cBrOutput*
then show *?case* **by** *simp*
next
case(*cCase* Ψ P P' φ *Cs* x)
then show *?case* **by**(*induct Cs*, *auto*)
next
case(*cPar1* Ψ Ψ_Q P P' *xvec* Q x)
then show *?case* **by** *simp*
next
case(*cPar2* Ψ Ψ_P Q Q' *xvec* P x)
then show *?case* **by** *simp*
next
case(*cComm1* Ψ Ψ_Q P M N P' A_P Ψ_P Q K *xvec* Q' A_Q x)
then show *?case* **by** *simp*
next
case(*cComm2* Ψ Ψ_Q P M *xwec* N P' A_P Ψ_P Q K Q' A_Q x)
then show *?case* **by** *simp*
next
case *cBrMerge*
then show *?case* **by** *simp*
next

```
    case cBrComm1
    then show ?case by simp
  next
    case cBrComm2
    then show ?case by simp
  next
    case cBrClose
    then show ?case by simp
  next
    case(cOpen Ψ P M xvec yvec N P' x y)
    then show ?case by simp
  next
    case(cBrOpen Ψ P M xvec yvec N P' x y)
    then show ?case by simp
  next
    case(cScope Ψ P P' x y)
    then show ?case by(simp add: abs-fresh)
  next
    case(cBang Ψ P P' x)
    then show ?case by simp
  qed
qed

lemma brinputFreshDerivative:
  fixes Ψ  :: 'b
    and P  :: ('a, 'b, 'c) psi
    and M  :: 'a
    and N  :: 'a
    and P' :: ('a, 'b, 'c) psi
    and x  :: name

assumes Ψ ▷ P ⟼¿M⦇N⦈ ≺ P'
  and   x ♯ P
  and   x ♯ N

shows x ♯ P'
proof −
  have bn(¿M⦇N⦈) ♯* subject(¿M⦇N⦈) and distinct(bn(¿M⦇N⦈)) by simp+
  with ‹Ψ ▷ P ⟼¿M⦇N⦈ ≺ P'› show ?thesis using ‹x ♯ P› ‹x ♯ N›
  proof(nominal-induct Ψ P α==¿M⦇N⦈ P' avoiding: x rule: semanticsInduct)
    case(cAlpha Ψ P α P' p x)
    then show ?case by simp
  next
    case(cInput Ψ M' K xvec N' Tvec P x)
    then show ?case by simp
  next
    case(cBrInput Ψ M' K xvec N' Tvec P x)
     from ‹¿M'⦇(N'[xvec::=Tvec])⦈ = ¿M⦇N⦈› have M' = M and NeqN': N =
N'[xvec::=Tvec] by(simp add: action.inject)+
```

note ‹*length xvec* = *length Tvec*› ‹*distinct xvec*› **then**
moreover have $x \sharp Tvec$ **using** ‹*set xvec* ⊆ *supp N'*› ‹$x \sharp N$› *NeqN'*
by(*blast intro*: *substTerm.subst3*)
moreover from ‹*xvec* $\sharp*$ *x*› ‹$x \sharp K(\!|\lambda*xvec\ N'|\!).P$›
have $x \sharp P$ **by**(*simp add*: *inputChainFresh*) (*simp add*: *name-list-supp fresh-def*)
ultimately show *?case* **using** ‹*xvec* $\sharp*$ *x*› **by** *auto*
next
case(*cOutput* Ψ *M K N P x*)
then show *?case* **by** *simp*
next
case *cBrOutput*
then show *?case* **by** *simp*
next
case(*cCase* Ψ *P P'* φ *Cs x*)
then show *?case* **by**(*induct Cs*, *auto*)
next
case(*cPar1* Ψ Ψ_Q *P P' xvec Q x*)
then show *?case* **by** *simp*
next
case(*cPar2* Ψ Ψ_P *Q Q' xvec P x*)
then show *?case* **by** *simp*
next
case(*cComm1* Ψ Ψ_Q *P M N P'* A_P Ψ_P *Q K xvec Q'* A_Q *x*)
then show *?case* **by** *simp*
next
case(*cComm2* Ψ Ψ_Q *P M xwec N P'* A_P Ψ_P *Q K Q'* A_Q *x*)
then show *?case* **by** *simp*
next
case *cBrMerge*
then show *?case* **by** *simp*
next
case *cBrComm1*
then show *?case* **by** *simp*
next
case *cBrComm2*
then show *?case* **by** *simp*
next
case *cBrClose*
then show *?case* **by** *simp*
next
case(*cOpen* Ψ *P M xvec yvec N P' x y*)
then show *?case* **by** *simp*
next
case(*cBrOpen* Ψ *P M xvec yvec N P' x y*)
then show *?case* **by** *simp*
next
case(*cScope* Ψ *P P' x y*)
then show *?case* **by**(*simp add*: *abs-fresh*)
next

```
    case(cBang Ψ P P' x)
    then show ?case by simp
  qed
qed

lemma inputFreshChainDerivative:
  fixes Ψ    :: 'b
    and P    :: ('a, 'b, 'c) psi
    and M    :: 'a
    and N    :: 'a
    and P'   :: ('a, 'b, 'c) psi
    and xvec :: name list

assumes Ψ ▷ P ⟼M⦇N⦈ ≺ P'
  and   xvec ♯* P
  and   xvec ♯* N

shows xvec ♯* P'
  using assms
  by(induct xvec)
    (auto intro: inputFreshDerivative)

lemma brinputFreshChainDerivative:
  fixes Ψ    :: 'b
    and P    :: ('a, 'b, 'c) psi
    and M    :: 'a
    and N    :: 'a
    and P'   :: ('a, 'b, 'c) psi
    and xvec :: name list

assumes Ψ ▷ P ⟼¿M⦇N⦈ ≺ P'
  and   xvec ♯* P
  and   xvec ♯* N

shows xvec ♯* P'
  using assms
  by(induct xvec)
    (auto intro: brinputFreshDerivative)

lemma outputFreshDerivativeN:
  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 x    :: name

assumes Ψ ▷ P ⟼M⦇ν*xvec⦈⟨N⟩ ≺ P'
```

and $xvec \sharp* M$
and $distinct\ xvec$
and $x \sharp P$
and $x \sharp xvec$

shows $x \sharp N$
proof −
note $\langle \Psi \rhd P \longmapsto M(\!|\nu* xvec|\!)\langle N\rangle \prec P'\rangle$
moreover from $\langle xvec \sharp* M\rangle$ **have** $bn(M(\!|\nu* xvec|\!)\langle N\rangle) \sharp* subject(M(\!|\nu* xvec|\!)\langle N\rangle)$ **by** *simp*
moreover from $\langle distinct\ xvec\rangle$ **have** $distinct(bn(M(\!|\nu* xvec|\!)\langle N\rangle))$ **by** *simp*
ultimately show *freshN*: $x \sharp N$ **using** $\langle x \sharp P\rangle\ \langle x \sharp xvec\rangle$
proof(*nominal-induct* $\Psi\ P\ \alpha == M(\!|\nu* xvec|\!)\langle N\rangle\ P'$ *avoiding*: *x arbitrary*: *M xvec N rule*: *semanticsInduct*)
case(*cAlpha* $\Psi\ P\ \alpha\ P'\ p\ x\ M\ xvec\ N$)
have *S*: $set\ p \subseteq set(bn\ \alpha) \times set(bn(p \cdot \alpha))$ **by** *fact*
from $\langle (p \cdot \alpha) = M(\!|\nu* xvec|\!)\langle N\rangle\rangle$ **have** $(p \cdot p \cdot \alpha) = p \cdot (M(\!|\nu* xvec|\!)\langle N\rangle)$ **by**(*simp add*: *fresh-star-bij*)
with $\langle distinctPerm\ p\rangle$ **have** $\alpha = (p \cdot M)(\!|\nu*(p \cdot xvec)|\!)\langle (p \cdot N)\rangle$ **by** *simp*
moreover from $\langle (p \cdot \alpha) = M(\!|\nu* xvec|\!)\langle N\rangle\rangle\ \langle x \sharp xvec\rangle$ **have** $x \sharp (bn(p \cdot \alpha))$ **by** *simp*
with $\langle (bn\ \alpha) \sharp* x\rangle\ \langle x \sharp xvec\rangle\ S$ **have** $x \sharp (p \cdot xvec)$
by(*fastforce dest*: *pt-fresh-bij1*[*OF pt-name-inst*, *OF at-name-inst*, **where** *pi*=*p* **and** *x*=*xvec*])
ultimately have $x \sharp (p \cdot N)$ **using** $\langle x \sharp P\rangle$ **by**(*metis cAlpha*)
then have $(p \cdot x) \sharp (p \cdot p \cdot N)$ **by**(*simp add*: *pt-fresh-bij1*[*OF pt-name-inst*, *OF at-name-inst*])
with $\langle distinctPerm\ p\rangle\ \langle bn(\alpha) \sharp* x\rangle\ \langle x \sharp (bn(p \cdot \alpha))\rangle S$ **show** *?case* **by** *simp*
next
case *cInput*
then show *?case* **by** *simp*
next
case *cBrInput*
then show *?case* **by** *simp*
next
case *cOutput*
then show *?case* **by**(*simp add*: *action.inject*)
next
case *cBrOutput*
then show *?case* **by**(*simp add*: *action.inject*)
next
case (*cCase* $\Psi\ P\ P'\ \varphi\ Cs\ x\ M\ xvec\ N$)
then show *?case* **by**(*auto simp add*: *action.inject dest*: *memFresh*)
next
case *cPar1*
then show *?case* **by** *simp*
next
case *cPar2*
then show *?case* **by** *simp*

```
  next
    case cComm1
    then show ?case by simp
  next
    case cComm2
    then show ?case by simp
  next
    case cBrMerge
    then show ?case by simp
  next
    case cBrComm1
    then show ?case by simp
  next
    case cBrComm2
    then show ?case by simp
  next
    case cBrClose
    then show ?case by simp
  next
    case(cOpen Ψ P M xvec yvec N P' x y M' zvec N')
   from ‹M⦇νx(xvec@x#yvec)⦈⟨N⟩ = M'⦇νx(zvec)⦈⟨N'⟩› have zvec = xvec@x#yvec
and N = N'
      by(simp add: action.inject)+
    from ‹y ♯ ⦇νx⦈P› ‹x ♯ y› have y ♯ P by(simp add: abs-fresh)
    moreover from ‹y ♯ zvec› ‹zvec = xvec@x#yvec›have y ♯ (xvec@yvec)
      by simp
    ultimately have y ♯ N by(fastforce intro!: cOpen)
    with ‹N = N'› show ?case by simp
  next
    case cBrOpen
    then show ?case by simp
  next
    case cScope
    then show ?case by(auto simp add: abs-fresh)
  next
    case cBang
    then show ?case by simp
  qed
qed

lemma broutputFreshDerivativeN:
  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 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 $x \sharp N$
proof −
note ‹$\Psi \rhd P \longmapsto ¡M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P'$›
moreover from ‹$xvec \sharp* M$› **have** $bn(¡M(\!|\nu{*}xvec|\!)\langle N\rangle) \sharp* subject(¡M(\!|\nu{*}xvec|\!)\langle N\rangle)$ **by** *simp*
moreover from ‹*distinct xvec*› **have** $distinct(bn(¡M(\!|\nu{*}xvec|\!)\langle N\rangle))$ **by** *simp*
ultimately show *freshN*: $x \sharp N$ **using** ‹$x \sharp P$› ‹$x \sharp xvec$›
proof(*nominal-induct* Ψ P α==$¡M(\!|\nu{*}xvec|\!)\langle N\rangle$ P' *avoiding*: x *arbitrary*: M $xvec$ N *rule*: *semanticsInduct*)
case(*cAlpha* Ψ P α P' p x M $xvec$ N)
have *S*: *set* $p \subseteq set(bn\ \alpha) \times set(bn(p \cdot \alpha))$ **by** *fact*
from ‹$(p \cdot \alpha) = ¡M(\!|\nu{*}xvec|\!)\langle N\rangle$› **have** $(p \cdot p \cdot \alpha) = p \cdot (¡M(\!|\nu{*}xvec|\!)\langle N\rangle)$ **by**(*simp add*: *fresh-star-bij*)
with ‹*distinctPerm p*› **have** $\alpha = ¡(p \cdot M)(\!|\nu{*}(p \cdot xvec)|\!)\langle(p \cdot N)\rangle$ **by** *simp*
moreover from ‹$(p \cdot \alpha) = ¡M(\!|\nu{*}xvec|\!)\langle N\rangle$› ‹$x \sharp xvec$› **have** $x \sharp (bn(p \cdot \alpha))$ **by** *simp*
with ‹$(bn\ \alpha) \sharp* x$› ‹$x \sharp xvec$› *S* **have** $x \sharp (p \cdot xvec)$
by(*fastforce dest*: *pt-fresh-bij1*[*OF pt-name-inst*, *OF at-name-inst*, **where** *pi*=*p* **and** *x*=*xvec*])
ultimately have $x \sharp (p \cdot N)$ **using** ‹$x \sharp P$› **by**(*metis cAlpha*)
then have $(p \cdot x) \sharp (p \cdot p \cdot N)$ **by**(*simp add*: *pt-fresh-bij1*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹*distinctPerm p*› ‹$bn(\alpha) \sharp* x$› ‹$x \sharp (bn(p \cdot \alpha))$›*S* **show** *?case* **by** *simp*
next
case *cInput*
then show *?case* **by** *simp*
next
case *cBrInput*
then show *?case* **by** *simp*
next
case *cOutput*
then show *?case* **by**(*simp add*: *action.inject*)
next
case *cBrOutput*
then show *?case* **by**(*simp add*: *action.inject*)
next
case *cCase*
then show *?case* **by**(*auto simp add*: *action.inject dest*: *memFresh*)
next
case *cPar1*
then show *?case* **by** *simp*
next
case *cPar2*

```
    then show ?case by simp
  next
    case cComm1
    then show ?case by simp
  next
    case cComm2
    then show ?case by simp
  next
    case cBrMerge
    then show ?case by simp
  next
    case cBrComm1
    then show ?case by simp
  next
    case cBrComm2
    then show ?case by simp
  next
    case cBrClose
    then show ?case by simp
  next
    case(cOpen Ψ P M xvec yvec N P' x y M' zvec N')
    then show ?case by simp
  next
    case(cBrOpen Ψ P M xvec yvec N P' x y M' zvec N')
   from ‹¡M⦇ν∗(xvec@x#yvec)⦈⟨N⟩ = ¡M'⦇ν∗zvec⦈⟨N'⟩› have zvec = xvec@x#yvec
and N = N'
      by(simp add: action.inject)+
    from ‹y ♯ ⦇νx⦈P› ‹x ♯ y› have y ♯ P by(simp add: abs-fresh)
    moreover from ‹y ♯ zvec› ‹zvec = xvec@x#yvec›have y ♯ (xvec@yvec)
      by simp
    ultimately have y ♯ N by(fastforce intro!: cBrOpen)
    with ‹N = N'› show ?case by simp
  next
    case cScope
    then show ?case by(auto simp add: abs-fresh)
  next
    case cBang
    then show ?case by simp
  qed
qed

lemma outputFreshDerivativeP:
  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 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 $x \sharp P'$
proof −
note ‹$\Psi \rhd P \longmapsto M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P'$›
moreover from ‹$xvec \sharp* M$› **have** $bn(M(\!|\nu{*}xvec|\!)\langle N\rangle) \sharp* subject(M(\!|\nu{*}xvec|\!)\langle N\rangle)$ **by** *simp*
moreover from ‹*distinct xvec*› **have** $distinct(bn(M(\!|\nu{*}xvec|\!)\langle N\rangle))$ **by** *simp*
ultimately show $x \sharp P'$ **using** ‹$x \sharp P$› ‹$x \sharp xvec$›
proof(*nominal-induct* Ψ P α==$M(\!|\nu{*}xvec|\!)\langle N\rangle$ P' *avoiding*: x *arbitrary*: M *xvec* N *rule*: *semanticsInduct*)
case(*cAlpha* Ψ P α P' p x M *xvec* N)
have S: $set\ p \subseteq set(bn\ \alpha) \times set(bn(p \cdot \alpha))$ **by** *fact*
from ‹$(p \cdot \alpha) = M(\!|\nu{*}xvec|\!)\langle N\rangle$› **have** $(p \cdot p \cdot \alpha) = p \cdot (M(\!|\nu{*}xvec|\!)\langle N\rangle)$ **by**(*simp add*: *fresh-star-bij*)
with ‹*distinctPerm p*› **have** $\alpha = (p \cdot M)(\!|\nu{*}(p \cdot xvec)|\!)\langle(p \cdot N)\rangle$ **by** *simp*
moreover from ‹$(p \cdot \alpha) = M(\!|\nu{*}xvec|\!)\langle N\rangle$› ‹$x \sharp xvec$› **have** $x \sharp (bn(p \cdot \alpha))$ **by** *simp*
with ‹$(bn\ \alpha) \sharp* x$› ‹$x \sharp xvec$› S **have** $x \sharp (p \cdot xvec)$
by(*fastforce dest*: *pt-fresh-bij1*[*OF pt-name-inst*, *OF at-name-inst*, **where** *pi*=p **and** x=*xvec*])
ultimately have $x \sharp P'$ **using** ‹$x \sharp P$› **by**(*metis cAlpha*)
then have $(p \cdot x) \sharp (p \cdot P')$ **by**(*simp add*: *pt-fresh-bij1*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹*distinctPerm p*› ‹$bn(\alpha) \sharp* x$› ‹$x \sharp (bn(p \cdot \alpha))$›$S$ **show** *?case* **by** *simp*
next
case *cInput*
then show *?case* **by** *simp*
next
case *cBrInput*
then show *?case* **by** *simp*
next
case *cOutput*
then show *?case* **by**(*simp add*: *action.inject*)
next
case *cBrOutput*
then show *?case* **by**(*simp add*: *action.inject*)
next
case *cCase*
then show *?case* **by**(*auto simp add*: *action.inject dest*: *memFresh*)
next
case *cPar1*
then show *?case* **by** *simp*
next

```
    case cPar2
    then show ?case by simp
  next
    case cComm1
    then show ?case by simp
  next
    case cComm2
    then show ?case by simp
  next
    case cBrMerge
    then show ?case by simp
  next
    case cBrComm1
    then show ?case by simp
  next
    case cBrComm2
    then show ?case by simp
  next
    case cBrClose
    then show ?case by simp
  next
    case(cOpen Ψ P M xvec yvec N P' x y M' zvec N')
   from ‹M⦇νx*(xvec@x#yvec)⦈⟨N⟩ = M'⦇νx*zvec⦈⟨N'⟩› have zvec = xvec@x#yvec
      by(simp add: action.inject)
    from ‹y ♯ ⦇νx⦈P› ‹x ♯ y›  have y ♯ P by(simp add: abs-fresh)
    moreover from ‹y ♯ zvec› ‹zvec = xvec@x#yvec›have y ♯ (xvec@yvec)
      by simp
    ultimately show y ♯ P'
      by(fastforce intro!: cOpen)
  next
    case cBrOpen
    then show ?case by simp
  next
    case cScope
    then show ?case by(auto simp add: abs-fresh)
  next
    case cBang
    then show ?case by simp
  qed
qed

lemma broutputFreshDerivativeP:
  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 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 $x \sharp P'$
proof −
note ‹$\Psi \rhd P \longmapsto ¡M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P'$›
moreover from ‹$xvec \sharp* M$› **have** $bn(¡M(\!|\nu{*}xvec|\!)\langle N\rangle) \sharp* subject(¡M(\!|\nu{*}xvec|\!)\langle N\rangle)$ **by** *simp*
moreover from ‹*distinct xvec*› **have** $distinct(bn(¡M(\!|\nu{*}xvec|\!)\langle N\rangle))$ **by** *simp*
ultimately show $x \sharp P'$ **using** ‹$x \sharp P$› ‹$x \sharp xvec$›
proof(*nominal-induct* Ψ P α==$¡M(\!|\nu{*}xvec|\!)\langle N\rangle$ P' *avoiding*: *x arbitrary*: *M xvec N rule*: *semanticsInduct*)
case(*cAlpha* Ψ P α P' p x M *xvec* N)
have S: *set* $p \subseteq set(bn\ \alpha) \times set(bn(p \cdot \alpha))$ **by** *fact*
from ‹$(p \cdot \alpha) = ¡M(\!|\nu{*}xvec|\!)\langle N\rangle$› **have** $(p \cdot p \cdot \alpha) = p \cdot (¡M(\!|\nu{*}xvec|\!)\langle N\rangle)$ **by**(*simp add*: *fresh-star-bij*)
with ‹*distinctPerm p*› **have** $\alpha = ¡(p \cdot M)(\!|\nu{*}(p \cdot xvec)|\!)\langle(p \cdot N)\rangle$ **by** *simp*
moreover from ‹$(p \cdot \alpha) = ¡M(\!|\nu{*}xvec|\!)\langle N\rangle$› ‹$x \sharp xvec$› **have** $x \sharp (bn(p \cdot \alpha))$ **by** *simp*
with ‹$(bn\ \alpha) \sharp* x$› ‹$x \sharp xvec$› S **have** $x \sharp (p \cdot xvec)$
by(*fastforce dest*: *pt-fresh-bij1*[*OF pt-name-inst*, *OF at-name-inst*, **where** *pi*=*p* **and** *x*=*xvec*])
ultimately have $x \sharp P'$ **using** ‹$x \sharp P$› **by**(*metis cAlpha*)
then have $(p \cdot x) \sharp (p \cdot P')$ **by**(*simp add*: *pt-fresh-bij1*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹*distinctPerm p*› ‹$bn(\alpha) \sharp* x$› ‹$x \sharp (bn(p \cdot \alpha))$›$S$ **show** *?case* **by** *simp*
next
case *cInput*
then show *?case* **by** *simp*
next
case *cBrInput*
then show *?case* **by** *simp*
next
case *cOutput*
then show *?case* **by**(*simp add*: *action.inject*)
next
case *cBrOutput*
then show *?case* **by**(*simp add*: *action.inject*)
next
case *cCase*
then show *?case* **by**(*auto simp add*: *action.inject dest*: *memFresh*)
next
case *cPar1*
then show *?case* **by** *simp*
next

case *cPar2*
then show *?case* **by** *simp*
next
case *cComm1*
then show *?case* **by** *simp*
next
case *cComm2*
then show *?case* **by** *simp*
next
case *cBrMerge*
then show *?case* **by** *simp*
next
case (*cBrComm1* Ψ Ψ_Q P M N P' A_P Ψ_P Q *xvec* Q' A_Q x M' *zvec* N')
from ‹$x \sharp (P \parallel Q)$› **have** $x \sharp P$ **and** $x \sharp Q$ **by** *simp+*

from ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec Q'$› ‹*xvec* $\sharp*$ M› ‹*distinct xvec*› ‹$x \sharp Q$› ‹*xvec* $\sharp*$ x›
have $x \sharp N$ **by**(*simp add*: *broutputFreshDerivativeN*)

with ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$› ‹$x \sharp P$› **have** $x \sharp P'$ **by**(*simp add*: *brinputFreshDerivative*)

then show *?case* **using** *cBrComm1* **by** *simp*
next
case (*cBrComm2* Ψ Ψ_Q P M *xvec* N P' A_P Ψ_P Q Q' A_Q x M' *zvec* N')
from ‹$x \sharp (P \parallel Q)$› **have** $x \sharp P$ **and** $x \sharp Q$ **by** *simp+*

from ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$› ‹*xvec* $\sharp*$ M› ‹*distinct xvec*› ‹$x \sharp P$› ‹*xvec* $\sharp*$ x›
have $x \sharp N$ **by**(*simp add*: *broutputFreshDerivativeN*)

with ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$› ‹$x \sharp Q$› **have** $x \sharp Q'$ **by**(*simp add*: *brinputFreshDerivative*)

then show *?case* **using** *cBrComm2* **by** *simp*
next
case *cBrClose*
then show *?case* **by** *simp*
next
case *cOpen*
then show *?case* **by** *simp*
next
case(*cBrOpen* Ψ P M *xvec yvec* N P' x y M' *zvec* N')
from ‹$¡M(\!|\nu * (xvec @ x \# yvec)|\!)\langle N\rangle = ¡M'(\!|\nu * zvec|\!)\langle N'\rangle$› **have** $zvec = xvec @ x \# yvec$
by(*simp add*: *action.inject*)
from ‹$y \sharp (\!|\nu x|\!)P$› ‹$x \sharp y$› **have** $y \sharp P$ **by**(*simp add*: *abs-fresh*)
moreover from ‹$y \sharp zvec$› ‹$zvec = xvec @ x \# yvec$›**have** $y \sharp (xvec @ yvec)$
by *simp*
ultimately show $y \sharp P'$

```
    by(fastforce intro: cBrOpen)
  next
    case cScope
    then show ?case by(auto simp add: abs-fresh)
  next
    case cBang
    then show ?case by simp
  qed
qed

lemma outputFreshDerivative:
  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 x    :: name

assumes Ψ ▷ P ⟼M(⦇ν*xvec⦈)⟨N⟩ ≺ P'
  and   xvec ♯* M
  and   distinct xvec
  and   x ♯ P
  and   x ♯ xvec

shows   x ♯ N
  and   x ♯ P'
  using assms
  by(auto simp add: outputFreshDerivativeN outputFreshDerivativeP)

lemma broutputFreshDerivative:
  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 x    :: name

assumes Ψ ▷ P ⟼¡M(⦇ν*xvec⦈)⟨N⟩ ≺ P'
  and   xvec ♯* M
  and   distinct xvec
  and   x ♯ P
  and   x ♯ xvec

shows   x ♯ N
  and   x ♯ P'
  using assms
  by(auto simp add: broutputFreshDerivativeN broutputFreshDerivativeP)
```

lemma *outputFreshChainDerivative*:
  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 *yvec* :: *name list*

assumes $\Psi \rhd P \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$
  and *xvec* $\sharp*$ M
  and *distinct xvec*
  and *yvec* $\sharp*$ P
  and *yvec* $\sharp*$ *xvec*

shows *yvec* $\sharp*$ N
  and *yvec* $\sharp*$ P'
  using *assms*
  by(*induct yvec*) (*auto intro*: *outputFreshDerivative*)

lemma *broutputFreshChainDerivative*:
  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 *yvec* :: *name list*

assumes $\Psi \rhd P \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$
  and *xvec* $\sharp*$ M
  and *distinct xvec*
  and *yvec* $\sharp*$ P
  and *yvec* $\sharp*$ *xvec*

shows *yvec* $\sharp*$ N
  and *yvec* $\sharp*$ P'
  using *assms*
  by(*induct yvec*) (*auto intro*: *broutputFreshDerivative*)

lemma *tauFreshDerivative*:
  fixes Ψ :: $'b$
    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 −
have $bn(\tau) \sharp* subject(\tau)$ **and** $distinct(bn(\tau))$ **by** *simp+*
with $\langle \Psi \rhd P \longmapsto \tau \prec P' \rangle$ **show** *?thesis* **using** $\langle x \sharp P \rangle$
proof(*nominal-induct* Ψ P $\alpha==(\tau::('a\ action))$ P' *avoiding*: *x rule*: *semanticsInduct*)
case *cAlpha*
then show *?case* **by** *simp*
next
case *cInput*
then show *?case* **by** *simp*
next
case *cBrInput*
then show *?case* **by** *simp*
next
case *cOutput*
then show *?case* **by** *simp*
next
case *cBrOutput*
then show *?case* **by** *simp*
next
case *cCase*
then show *?case* **by**(*auto dest*: *memFresh*)
next
case *cPar1*
then show *?case* **by** *simp*
next
case *cPar2*
then show *?case* **by** *simp*
next
case *cComm1*
then show *?case*
by(*auto dest*: *inputFreshDerivative outputFreshDerivative simp add*: *resChainFresh*)
next
case *cComm2*
then show *?case*
by(*auto dest*: *inputFreshDerivative outputFreshDerivative simp add*: *resChainFresh*)
next
case *cBrMerge*
then show *?case* **by** *simp*
next
case *cBrComm1*
then show *?case* **by** *simp*
next
case *cBrComm2*
then show *?case* **by** *simp*

next
case *cBrClose*
then show *?case*
by(*auto dest*: *brinputFreshDerivative broutputFreshDerivative simp add*: *resChainFresh abs-fresh*)
next
case *cOpen*
then show *?case* **by** *simp*
next
case *cBrOpen*
then show *?case* **by** *simp*
next
case *cScope*
then show *?case* **by**(*simp add*: *abs-fresh*)
next
case *cBang*
then show *?case* **by** *simp*
qed
qed

lemma *tauFreshChainDerivative*:
fixes Ψ :: $'b$
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'
using *assms*
by(*induct xvec*) (*auto intro*: *tauFreshDerivative*)

lemma *freeFreshDerivative*:
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 *bn* α $\sharp*$ *subject* α
and *distinct*(*bn* α)
and $x \sharp \alpha$
and $x \sharp P$

shows $x \sharp P'$

using *assms*
apply −
by(*rule actionCases*[**where** α=α])
(*auto intro*: *inputFreshDerivative brinputFreshDerivative*
tauFreshDerivative
outputFreshDerivative broutputFreshDerivative)

lemma *freeFreshChainDerivative*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and α :: $'a$ *action*
and P' :: $('a, 'b, 'c)$ *psi*
and *xvec* :: *name list*

assumes $\Psi \rhd P \longmapsto \alpha \prec P'$
and *bn* α $\sharp*$ *subject* α
and *distinct*(*bn* α)
and *xvec* $\sharp*$ P
and *xvec* $\sharp*$ α

shows *xvec* $\sharp*$ P'
using *assms*
by(*auto intro*: *freeFreshDerivative simp add*: *fresh-star-def*)

lemma *Input*:
fixes Ψ :: $'b$
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 −
obtain p **where** *xvecFreshPsi*: $((p::name\ prm) \cdot (xvec::name\ list)) \sharp* \Psi$
and *xvecFreshM*: $(p \cdot xvec) \sharp* M$
and *xvecFreshN*: $(p \cdot xvec) \sharp* N$
and *xvecFreshK*: $(p \cdot xvec) \sharp* K$
and *xvecFreshTvec*: $(p \cdot xvec) \sharp* Tvec$
and *xvecFreshP*: $(p \cdot xvec) \sharp* P$
and S: $(set\ p) \subseteq (set\ xvec) \times (set(p \cdot xvec))$
and *dp*: *distinctPerm p*
apply −
by(*rule name-list-avoiding*[**where** *xvec*=*xvec* **and** c=(Ψ, M, K, N, P, *Tvec*)])

```
    (auto simp add: eqvts fresh-star-prod)
  note ‹Ψ ⊢ M ↔ K›
  moreover from ‹distinct xvec› have distinct(p • xvec)
    by simp
  moreover from ‹(set xvec) ⊆ (supp N)› have (p • (set xvec)) ⊆ (p • (supp N))
    by simp
  then have set(p • xvec) ⊆ supp(p • N)
    by(simp add: eqvts)
  moreover from ‹length xvec = length Tvec› have length(p • xvec) = length Tvec
    by simp
  ultimately have Ψ ▷ M⦇λ*(p • xvec) (p • N)⦈.(p • P) ⟼K⦇(p • N)[(p •
xvec)::=Tvec]⦈ ≺ (p • P)[(p • xvec)::=Tvec]
    using xvecFreshPsi xvecFreshM xvecFreshK xvecFreshTvec
    by(metis cInput)
  then show ?thesis using xvecFreshN xvecFreshP S ‹length xvec = length Tvec›
dp
    by(auto simp add: inputChainAlpha′ substTerm.renaming renaming)
qed

lemma BrInput:
  fixes Ψ    :: 'b
    and M    :: 'a
    and K    :: 'a
    and xvec :: name list
    and N    :: 'a
    and Tvec :: 'a list

assumes Ψ ⊢ K ≻ M
  and   distinct xvec
  and   set xvec ⊆ supp N
  and   length xvec = length Tvec

shows Ψ ▷ M⦇λ*xvec N⦈.P ⟼¿K⦇N[xvec::=Tvec]⦈ ≺ P[xvec::=Tvec]
proof −
  obtain p where xvecFreshPsi: ((p::name prm) • (xvec::name list)) ♯* Ψ
    and xvecFreshM: (p • xvec) ♯* M
    and xvecFreshN: (p • xvec) ♯* N
    and xvecFreshK: (p • xvec) ♯* K
    and xvecFreshTvec: (p • xvec) ♯* Tvec
    and xvecFreshP: (p • xvec) ♯* P
    and S: (set p) ⊆ (set xvec) × (set(p • xvec))
    and dp: distinctPerm p
    apply −
    by(rule name-list-avoiding[where xvec=xvec and c=(Ψ, M, K, N, P, Tvec)])
      (auto simp add: eqvts fresh-star-prod)
  note ‹Ψ ⊢ K ≻ M›
  moreover from ‹distinct xvec› have distinct(p • xvec)
    by simp
  moreover from ‹(set xvec) ⊆ (supp N)› have (p • (set xvec)) ⊆ (p • (supp N))
```

by *simp*
then have $set(p \cdot xvec) \subseteq supp(p \cdot N)$
by(*simp add*: *eqvts*)
moreover from ‹*length xvec* = *length Tvec*› **have** $length(p \cdot xvec) = length\ Tvec$
by *simp*
ultimately have $\Psi \rhd M(\!|\lambda*(p \cdot xvec)\ (p \cdot N)|\!).(p \cdot P) \longmapsto ¿K(\!|(p \cdot N)[(p \cdot xvec)::=Tvec]|\!) \prec (p \cdot P)[(p \cdot xvec)::=Tvec]$
using *xvecFreshPsi xvecFreshM xvecFreshK xvecFreshTvec*
by(*metis cBrInput*)
then show *?thesis* **using** *xvecFreshN xvecFreshP S* ‹*length xvec* = *length Tvec*› *dp*
by(*auto simp add*: *inputChainAlpha′ substTerm.renaming renaming*)
qed

lemma *residualAlpha*:
fixes p :: *name prm*
and α :: *'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 $\alpha \prec P = (p \cdot \alpha) \prec (p \cdot P)$
using *assms*
apply −
apply(*rule actionCases*[**where** α=α])
apply(*simp only*: *eqvts bn.simps*)
apply *simp*
apply(*simp only*: *eqvts bn.simps*)
apply *simp*
apply *simp*
apply(*simp add*: *boundOutputChainAlpha″ residualInject*)
apply *simp*
apply(*simp add*: *boundOutputChainAlpha″ residualInject*)
by *simp*

lemma *Par1*:
fixes Ψ :: *'b*
and Ψ_Q :: *'b*
and P :: (*'a*, *'b*, *'c*) *psi*
and α :: *'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 = ⟨A_Q, Ψ_Q⟩
  and   bn α ♯* Q
  and   A_Q ♯* Ψ
  and   A_Q ♯* P
  and   A_Q ♯* α

shows Ψ ▷ P ∥ Q ⟼α ≺ (P' ∥ Q)
proof −
  {
    fix Ψ   :: 'b
      and Ψ_Q  :: 'b
      and P   :: ('a, 'b, 'c) psi
      and α   :: 'a action
      and P'  :: ('a, 'b, 'c) psi
      and A_Q  :: name list
      and Q   :: ('a, 'b, 'c) psi

    assume Ψ ⊗ Ψ_Q ▷ P ⟼α ≺ P'
      and   extractFrame Q = ⟨A_Q, Ψ_Q⟩
      and   bn α ♯* Q
      and   bn α ♯* subject α
      and   A_Q ♯* Ψ
      and   A_Q ♯* P
      and   A_Q ♯* α
      and   distinct A_Q

    have  Ψ ▷ P ∥ Q ⟼α ≺ (P' ∥ Q)
    proof −
      from ‹Ψ ⊗ Ψ_Q ▷ P ⟼α ≺ P'› have distinct(bn α) by(rule boundOutput-
Distinct)
      obtain q::name prm where bn(q · α) ♯* Ψ and bn(q · α) ♯* P and bn(q ·
α) ♯* Q and bn(q · α) ♯* α
        and bn(q · α) ♯* A_Q and bn(q · α) ♯* P' and bn(q · α) ♯* Ψ_Q
        and Sq: (set q) ⊆ (set (bn α)) × (set(bn(q · α)))
        apply −
        by(rule name-list-avoiding[where xvec=bn α and c=(Ψ, P, Q, α, A_Q, Ψ_Q,
P')]) (auto simp add: eqvts)
      obtain p::name prm where (p · A_Q) ♯* Ψ and (p · A_Q) ♯* P and (p · A_Q)
♯* Q and (p · A_Q) ♯* α
        and (p · A_Q) ♯* α and (p · A_Q) ♯* (q · α) and (p · A_Q) ♯* P'
        and (p · A_Q) ♯* (q · P') and (p · A_Q) ♯* Ψ_Q and Sp: (set p) ⊆ (set A_Q)
× (set(p · A_Q))
        apply −
        by(rule name-list-avoiding[where xvec=A_Q and c=(Ψ, P, Q, α, bn α, q ·
α, P', (q · P'), Ψ_Q)]) auto
      from ‹distinct(bn α)› have distinct(bn(q · α))
        by − (rule actionCases[where α=α], auto simp add: eqvts)
      from ‹A_Q ♯* α› ‹bn(q · α) ♯* A_Q› Sq have A_Q ♯* (q · α)
        apply −
```

apply(*rule actionCases*[**where** α=α])
apply(*simp only*: *bn.simps eqvts*, *simp*)
apply(*simp only*: *bn.simps eqvts*, *simp*)
apply(*simp add*: *freshChainSimps*)
apply(*simp add*: *freshChainSimps*)
by *simp*
from ‹*bn* α $\sharp*$ *subject* α› **have** $(q \cdot (bn\ \alpha))\ \sharp*\ (q \cdot (subject\ \alpha))$
by(*simp add*: *fresh-star-bij*)
then have $bn(q \cdot \alpha)\ \sharp*\ subject(q \cdot \alpha)$ **by**(*simp add*: *eqvts*)
from ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto \alpha \prec P'$› ‹$bn(q \cdot \alpha)\ \sharp*\ \alpha$› ‹$bn(q \cdot \alpha)\ \sharp*\ P'$› ‹*bn* α $\sharp*$ (*subject* α)› *Sq*
have *Trans*: $\Psi \otimes \Psi_Q \rhd P \longmapsto (q \cdot \alpha) \prec (q \cdot P')$
by(*force simp add*: *residualAlpha*)
then have $A_Q\ \sharp*\ (q \cdot P')$ **using** ‹$bn(q \cdot \alpha)\ \sharp*\ subject(q \cdot \alpha)$› ‹*distinct*(*bn*(*q* · α))› ‹$A_Q\ \sharp*\ P$› ‹$A_Q\ \sharp*\ (q \cdot \alpha)$›
by(*auto intro*: *freeFreshChainDerivative*)
from *Trans* **have** $(p \cdot (\Psi \otimes \Psi_Q)) \rhd (p \cdot P) \longmapsto p \cdot ((q \cdot \alpha) \prec (q \cdot P'))$
by(*rule semantics.eqvt*)
with ‹$A_Q\ \sharp*\ \Psi$› ‹$A_Q\ \sharp*\ P$› ‹$A_Q\ \sharp*\ (q \cdot \alpha)$› ‹$(p \cdot A_Q)\ \sharp*\ (q \cdot \alpha)$› ‹$A_Q\ \sharp*\ (q \cdot P')$›
‹$(p \cdot A_Q)\ \sharp*\ \Psi$› ‹$(p \cdot A_Q)\ \sharp*\ P$› ‹$(p \cdot A_Q)\ \sharp*\ (q \cdot P')$› *Sp*
have $\Psi \otimes (p \cdot \Psi_Q) \rhd P \longmapsto (q \cdot \alpha) \prec (q \cdot P')$ **by**(*simp add*: *eqvts*)
moreover from ‹*extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$› ‹$(p \cdot A_Q)\ \sharp*\ \Psi_Q$› *Sp* **have** *extractFrame* $Q = \langle (p \cdot A_Q), (p \cdot \Psi_Q) \rangle$
by(*simp add*: *frameChainAlpha'* *eqvts*)
moreover from ‹$(bn(q \cdot \alpha))\ \sharp*\ \Psi_Q$› ‹$(bn(q \cdot \alpha))\ \sharp*\ A_Q$› ‹$(p \cdot A_Q)\ \sharp*\ (q \cdot \alpha)$› *Sp*
have $(bn(q \cdot \alpha))\ \sharp*\ (p \cdot \Psi_Q)$
by(*simp add*: *freshAlphaPerm*)
moreover from ‹*distinct* A_Q› **have** $distinct(p \cdot A_Q)$ **by** *simp*
ultimately have $\Psi \rhd P \parallel Q \longmapsto (q \cdot \alpha) \prec ((q \cdot P') \parallel Q)$
using ‹$(p \cdot A_Q)\ \sharp*\ P$› ‹$(p \cdot A_Q)\ \sharp*\ Q$› ‹$(p \cdot A_Q)\ \sharp*\ \Psi$› ‹$(p \cdot A_Q)\ \sharp*\ (q \cdot \alpha)$›
‹$(p \cdot A_Q)\ \sharp*\ (q \cdot P')$› ‹$(bn(q \cdot \alpha))\ \sharp*\ \Psi$› ‹$(bn(q \cdot \alpha))\ \sharp*\ Q$› ‹$(bn(q \cdot \alpha))\ \sharp*\ P$›
‹$(bn(q \cdot \alpha))\ \sharp*\ (subject\ (q \cdot \alpha))$› ‹$distinct(bn(q \cdot \alpha))$›
by(*metis cPar1*)

then show *?thesis* **using** ‹$bn(q \cdot \alpha)\ \sharp*\ \alpha$› ‹$bn(q \cdot \alpha)\ \sharp*\ P'$› ‹*bn* α $\sharp*$ *subject* α› ‹$bn(q \cdot \alpha)\ \sharp*\ Q$› ‹*bn* α $\sharp*$ *Q*› *Sq*
by(*force simp add*: *residualAlpha*)
qed
}
note *Goal* = *this*
from ‹*extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$› ‹$A_Q\ \sharp*\ \Psi$› ‹$A_Q\ \sharp*\ P$› ‹$A_Q\ \sharp*\ \alpha$›
obtain A_Q' **where** *FrQ*: *extractFrame* $Q = \langle A_Q', \Psi_Q \rangle$ **and** *distinct* A_Q' **and** $A_Q'\ \sharp*\ \Psi$ **and** $A_Q'\ \sharp*\ P$ **and** $A_Q'\ \sharp*\ \alpha$
apply −
by(*rule distinctFrame*[**where** *C*=(Ψ, *P*, α)]) *auto*
show *?thesis*

proof(*induct rule*: *actionCases*[**where** α=α])
case(*cInput M N*)
from *Trans FrQ* ‹$A_Q{'} \sharp* \Psi$› ‹$A_Q{'} \sharp* P$› ‹$A_Q{'} \sharp* \alpha$› ‹*distinct* $A_Q{'}$› ‹*bn* $\alpha \sharp* Q$›
show *?case* **using** ‹$\alpha = M(\!|N|\!)$› **by**(*force intro*: *Goal*)
next
case(*cBrInput M N*)
from *Trans FrQ* ‹$A_Q{'} \sharp* \Psi$› ‹$A_Q{'} \sharp* P$› ‹$A_Q{'} \sharp* \alpha$› ‹*distinct* $A_Q{'}$› ‹*bn* $\alpha \sharp* Q$›
show *?case* **using** ‹$\alpha = ¿M(\!|N|\!)$› **by**(*force intro*: *Goal*)
next
case *cTau*
from *Trans FrQ* ‹$A_Q{'} \sharp* \Psi$› ‹$A_Q{'} \sharp* P$› ‹$A_Q{'} \sharp* \alpha$› ‹*distinct* $A_Q{'}$› ‹*bn* $\alpha \sharp* Q$›
show *?case* **using** ‹$\alpha = \tau$› **by**(*force intro*: *Goal*)
next
case(*cOutput M xvec N*)
from ‹$\alpha = M(\!|\nu*xvec|\!)\langle N\rangle$› ‹$A_Q{'} \sharp* \alpha$› ‹*bn* $\alpha \sharp* Q$› **have** *xvec* $\sharp*$ $A_Q{'}$ **and** *xvec* $\sharp*$ *Q*
by *simp+*
obtain *p* **where** $(p \cdot xvec) \sharp* N$ **and** $(p \cdot xvec) \sharp* P'$ **and** $(p \cdot xvec) \sharp* Q$
and $(p \cdot xvec) \sharp* M$ **and** $(p \cdot xvec) \sharp* A_Q{'}$
and *S*: *set p* $\subseteq$ *set xvec* $\times$ *set*$(p \cdot xvec)$
apply −
by(*rule name-list-avoiding*[**where** *xvec*=*xvec* **and** *c*=(*N*, *P'*, *Q*, *M*, $A_Q{'}$)]) *auto*
from *Trans* ‹α=$M(\!|\nu*xvec|\!)\langle N\rangle$› **have** $\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|\nu*xvec|\!)\langle N\rangle \prec P'$ **by** *simp*
with ‹$(p \cdot xvec) \sharp* N$› ‹$(p \cdot xvec) \sharp* P'$› *S*
have $\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|\nu*(p \cdot xvec)|\!)\langle(p \cdot N)\rangle \prec (p \cdot P')$
by(*simp add*: *boundOutputChainAlpha''* *create-residual.simps*)
moreover from ‹*xvec* $\sharp*$ $A_Q{'}$› ‹$(p \cdot xvec) \sharp* A_Q{'}$› ‹$A_Q{'} \sharp* \alpha$› *S*
have $A_Q{'} \sharp* (p \cdot \alpha)$ **by**(*simp add*: *freshChainSimps del*: *actionFreshChain*)
ultimately have $\Psi \rhd P \parallel Q \longmapsto M(\!|\nu*(p \cdot xvec)|\!)\langle(p \cdot N)\rangle \prec (p \cdot P') \parallel Q$
using *FrQ* ‹$A_Q{'} \sharp* \Psi$› ‹$A_Q{'} \sharp* P$› ‹*distinct* $A_Q{'}$› ‹$(p \cdot xvec) \sharp* Q$› ‹$A_Q{'} \sharp* \alpha$› ‹$(p \cdot xvec) \sharp* M$› ‹$\alpha = M(\!|\nu*xvec|\!)\langle N\rangle$›
by(*force intro*: *Goal*)
with ‹$(p \cdot xvec) \sharp* N$› ‹$(p \cdot xvec) \sharp* P'$› ‹$(p \cdot xvec) \sharp* Q$› ‹*xvec* $\sharp*$ *Q*› *S* ‹$\alpha = M(\!|\nu*xvec|\!)\langle N\rangle$›
show *?case*
by(*simp add*: *boundOutputChainAlpha''* *eqvts* *create-residual.simps*)
next
case(*cBrOutput M xvec N*)
from ‹$\alpha = ¡M(\!|\nu*xvec|\!)\langle N\rangle$› ‹$A_Q{'} \sharp* \alpha$› ‹*bn* $\alpha \sharp* Q$› **have** *xvec* $\sharp*$ $A_Q{'}$ **and** *xvec* $\sharp*$ *Q*
by *simp+*
obtain *p* **where** $(p \cdot xvec) \sharp* N$ **and** $(p \cdot xvec) \sharp* P'$ **and** $(p \cdot xvec) \sharp* Q$
and $(p \cdot xvec) \sharp* M$ **and** $(p \cdot xvec) \sharp* A_Q{'}$
and *S*: *set p* $\subseteq$ *set xvec* $\times$ *set*$(p \cdot xvec)$
by(*rule name-list-avoiding*[**where** *xvec*=*xvec* **and** *c*=(*N*, *P'*, *Q*, *M*, $A_Q{'}$)]) *auto*
from *Trans* ‹α=$¡M(\!|\nu*xvec|\!)\langle N\rangle$› **have** $\Psi \otimes \Psi_Q \rhd P \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec P'$

by *simp*
with ‹$(p \cdot xvec) \sharp* N$› ‹$(p \cdot xvec) \sharp* P'$› S
have $\Psi \otimes \Psi_Q \rhd P \longmapsto ¡M(\!|\nu*(p \cdot xvec)|\!)\langle(p \cdot N)\rangle \prec (p \cdot P')$
by(*simp add*: *boundOutputChainAlpha''* *create-residual.simps*)
moreover from ‹$xvec \sharp* A_Q{}'$› ‹$(p \cdot xvec) \sharp* A_Q{}'$› ‹$A_Q{}' \sharp* \alpha$› S
have $A_Q{}' \sharp* (p \cdot \alpha)$ **by**(*simp add*: *freshChainSimps del*: *actionFreshChain*)
ultimately have $\Psi \rhd P \parallel Q \longmapsto ¡M(\!|\nu*(p \cdot xvec)|\!)\langle(p \cdot N)\rangle \prec (p \cdot P') \parallel Q$
using *FrQ* ‹$A_Q{}' \sharp* \Psi$› ‹$A_Q{}' \sharp* P$› ‹*distinct* $A_Q{}'$› ‹$(p \cdot xvec) \sharp* Q$› ‹$A_Q{}' \sharp* \alpha$›
‹$(p \cdot xvec) \sharp* M$› ‹$\alpha = ¡M(\!|\nu*xvec|\!)\langle N\rangle$›
by(*force intro*: *Goal*)
with ‹$(p \cdot xvec) \sharp* N$› ‹$(p \cdot xvec) \sharp* P'$› ‹$(p \cdot xvec) \sharp* Q$› ‹$xvec \sharp* Q$› S ‹$\alpha = ¡M(\!|\nu*xvec|\!)\langle N\rangle$›
show *?case*
by(*simp add*: *boundOutputChainAlpha''* *eqvts* *create-residual.simps*)
qed
qed

lemma *Par2*:
fixes Ψ :: *'b*
and Ψ_P :: *'b*
and Q :: (*'a*, *'b*, *'c*) *psi*
and α :: *'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 −
{
fix Ψ :: *'b*
and Ψ_P :: *'b*
and Q :: (*'a*, *'b*, *'c*) *psi*
and α :: *'a action*
and Q' :: (*'a*, *'b*, *'c*) *psi*
and A_P :: *name list*
and P :: (*'a*, *'b*, *'c*) *psi*

assume $\Psi \otimes \Psi_P \rhd Q \longmapsto \alpha \prec Q'$
and *extractFrame* $P = \langle A_P, \Psi_P\rangle$
and $bn\ \alpha \sharp* P$
and $bn\ \alpha \sharp* subject\ \alpha$
and $A_P \sharp* \Psi$

and $A_P \sharp* Q$
and $A_P \sharp* \alpha$
and *distinct* A_P

have $\Psi \rhd P \parallel Q \longmapsto \alpha \prec (P \parallel Q')$
proof −
from ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto \alpha \prec Q'$› **have** *distinct*(*bn* α) **by**(*rule boundOutput-Distinct*)
obtain *q*::*name prm* **where** $bn(q \cdot \alpha) \sharp* \Psi$ **and** $bn(q \cdot \alpha) \sharp* P$ **and** $bn(q \cdot \alpha) \sharp* Q$ **and** $bn(q \cdot \alpha) \sharp* \alpha$
and $bn(q \cdot \alpha) \sharp* A_P$ **and** $bn(q \cdot \alpha) \sharp* Q'$ **and** $bn(q \cdot \alpha) \sharp* \Psi_P$
and *Sq*: (*set q*) ⊆ (*set* (*bn* α)) × (*set*($bn(q \cdot \alpha)$))
by(*rule name-list-avoiding*[**where** *xvec*=*bn* α **and** *c*=(Ψ, *P*, *Q*, α, A_P, Ψ_P, Q')]) (*auto simp add*: *eqvts*)
obtain *p*::*name prm* **where** $(p \cdot A_P) \sharp* \Psi$ **and** $(p \cdot A_P) \sharp* P$ **and** $(p \cdot A_P) \sharp* Q$ **and** $(p \cdot A_P) \sharp* \alpha$
and $(p \cdot A_P) \sharp* \alpha$ **and** $(p \cdot A_P) \sharp* (q \cdot \alpha)$ **and** $(p \cdot A_P) \sharp* Q'$
and $(p \cdot A_P) \sharp* (q \cdot Q')$ **and** $(p \cdot A_P) \sharp* \Psi_P$
and *Sp*: (*set p*) ⊆ (*set* A_P) × (*set*($p \cdot A_P$))
by(*rule name-list-avoiding*[**where** *xvec*=A_P **and** *c*=(Ψ, *P*, *Q*, α, $q \cdot \alpha$, Q', $(q \cdot Q')$, Ψ_P)]) *auto*
from ‹*distinct*(*bn* α)› **have** *distinct*($bn(q \cdot \alpha)$)
apply −
by(*rule actionCases*[**where** α=α]) (*auto simp add*: *eqvts*)
from ‹$A_P \sharp* \alpha$› ‹$bn(q \cdot \alpha) \sharp* A_P$› *Sq* **have** $A_P \sharp* (q \cdot \alpha)$
apply −
apply(*rule actionCases*[**where** α=α])
apply(*simp only*: *bn.simps eqvts*, *simp*)
apply(*simp only*: *bn.simps eqvts*, *simp*)
apply(*simp add*: *freshChainSimps*)
apply(*simp add*: *freshChainSimps*)
by *simp*
from ‹*bn* $\alpha \sharp*$ *subject* α› **have** $(q \cdot (bn\ \alpha)) \sharp* (q \cdot (subject\ \alpha))$
by(*simp add*: *fresh-star-bij*)
then have $bn(q \cdot \alpha) \sharp*$ *subject*($q \cdot \alpha$) **by**(*simp add*: *eqvts*)
from ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto \alpha \prec Q'$› ‹$bn(q \cdot \alpha) \sharp* \alpha$› ‹$bn(q \cdot \alpha) \sharp* Q'$› ‹*bn* $\alpha \sharp*$ (*subject* α)› *Sq*
have *Trans*: $\Psi \otimes \Psi_P \rhd Q \longmapsto (q \cdot \alpha) \prec (q \cdot Q')$
by(*force simp add*: *residualAlpha*)
then have $A_P \sharp* (q \cdot Q')$ **using** ‹$bn(q \cdot \alpha) \sharp*$ *subject*($q \cdot \alpha$)› ‹*distinct*($bn(q \cdot \alpha)$)› ‹$A_P \sharp* Q$› ‹$A_P \sharp* (q \cdot \alpha)$›
by(*auto intro*: *freeFreshChainDerivative*)
from *Trans* **have** $(p \cdot (\Psi \otimes \Psi_P)) \rhd (p \cdot Q) \longmapsto p \cdot ((q \cdot \alpha) \prec (q \cdot Q'))$
by(*rule semantics.eqvt*)
with ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* Q$› ‹$A_P \sharp* (q \cdot \alpha)$› ‹$(p \cdot A_P) \sharp* (q \cdot \alpha)$› ‹$A_P \sharp* (q \cdot Q')$›
‹$(p \cdot A_P) \sharp* \Psi$› ‹$(p \cdot A_P) \sharp* Q$› ‹$(p \cdot A_P) \sharp* (q \cdot Q')$› *Sp*
have $\Psi \otimes (p \cdot \Psi_P) \rhd Q \longmapsto (q \cdot \alpha) \prec (q \cdot Q')$ **by**(*simp add*: *eqvts*)
moreover from ‹*extractFrame* $P = \langle A_P, \Psi_P \rangle$› ‹$(p \cdot A_P) \sharp* \Psi_P$› *Sp* **have**

extractFrame $P = \langle(p \cdot A_P), (p \cdot \Psi_P)\rangle$
by(*simp add*: *frameChainAlpha' eqvts*)
moreover from ‹$(bn(q \cdot \alpha))$ $\sharp*$ Ψ_P› ‹$(bn(q \cdot \alpha))$ $\sharp*$ A_P› ‹$(p \cdot A_P)$ $\sharp*$ $(q \cdot \alpha)$› *Sp*
have $(bn(q \cdot \alpha))$ $\sharp*$ $(p \cdot \Psi_P)$
by(*simp add*: *freshAlphaPerm*)
moreover from ‹*distinct* A_P› **have** *distinct*$(p \cdot A_P)$ **by** *simp*
ultimately have $\Psi \rhd P \parallel Q \longmapsto(q \cdot \alpha) \prec (P \parallel (q \cdot Q'))$
using ‹$(p \cdot A_P)$ $\sharp*$ P› ‹$(p \cdot A_P)$ $\sharp*$ Q› ‹$(p \cdot A_P)$ $\sharp*$ Ψ› ‹$(p \cdot A_P)$ $\sharp*$ $(q \cdot \alpha)$›
‹$(p \cdot A_P)$ $\sharp*$ $(q \cdot Q')$› ‹$(bn(q \cdot \alpha))$ $\sharp*$ Ψ› ‹$(bn(q \cdot \alpha))$ $\sharp*$ Q› ‹$(bn(q \cdot \alpha))$ $\sharp*$ P›
‹$(bn(q \cdot \alpha))$ $\sharp*$ $(subject\ (q \cdot \alpha))$› ‹*distinct*$(bn(q \cdot \alpha))$›
by(*metis cPar2*)

then show *?thesis* **using** ‹$bn(q \cdot \alpha)$ $\sharp*$ α› ‹$bn(q \cdot \alpha)$ $\sharp*$ Q'› ‹$bn\ \alpha$ $\sharp*$ *subject* α› ‹$bn(q \cdot \alpha)$ $\sharp*$ P› ‹$bn\ \alpha$ $\sharp*$ P› *Sq*
by(*force simp add*: *residualAlpha*)
qed
}
note *Goal* = *this*
from ‹*extractFrame* $P = \langle A_P, \Psi_P\rangle$› ‹$A_P$ $\sharp*$ Ψ› ‹A_P $\sharp*$ Q› ‹A_P $\sharp*$ α›
obtain A_P' **where** *FrP*: *extractFrame* $P = \langle A_P', \Psi_P\rangle$ **and** *distinct* A_P' **and** A_P' $\sharp*$ Ψ **and** A_P' $\sharp*$ Q **and** A_P' $\sharp*$ α
apply −
by(*rule distinctFrame*[**where** C=(Ψ, Q, α)]) *auto*
show *?thesis*
proof(*induct rule*: *actionCases*[**where** α=α])
case(*cInput M N*)
from *Trans FrP* ‹A_P' $\sharp*$ Ψ› ‹A_P' $\sharp*$ Q› ‹A_P' $\sharp*$ α› ‹*distinct* A_P'› ‹$bn\ \alpha$ $\sharp*$ P›
show *?case* **using** ‹$\alpha = M(\!|N|\!)$› **by**(*force intro*: *Goal*)
next
case(*cBrInput M N*)
from *Trans FrP* ‹A_P' $\sharp*$ Ψ› ‹A_P' $\sharp*$ Q› ‹A_P' $\sharp*$ α› ‹*distinct* A_P'› ‹$bn\ \alpha$ $\sharp*$ P›
show *?case* **using** ‹$\alpha = ¿M(\!|N|\!)$› **by**(*force intro*: *Goal*)
next
case *cTau*
from *Trans FrP* ‹A_P' $\sharp*$ Ψ› ‹A_P' $\sharp*$ Q› ‹A_P' $\sharp*$ α› ‹*distinct* A_P'› ‹$bn\ \alpha$ $\sharp*$ P›
show *?case* **using** ‹$\alpha = \tau$› **by**(*force intro*: *Goal*)
next
case(*cOutput M xvec N*)
from ‹$\alpha = M(\!|\nu*xvec|\!)\langle N\rangle$› ‹$A_P'$ $\sharp*$ α› ‹$bn\ \alpha$ $\sharp*$ P› **have** *xvec* $\sharp*$ A_P' **and** *xvec* $\sharp*$ P
by *simp+*
obtain p **where** $(p \cdot xvec)$ $\sharp*$ N **and** $(p \cdot xvec)$ $\sharp*$ Q' **and** $(p \cdot xvec)$ $\sharp*$ P
and $(p \cdot xvec)$ $\sharp*$ M **and** $(p \cdot xvec)$ $\sharp*$ A_P'
and S: *set* $p \subseteq$ *set xvec* $\times$ *set*$(p \cdot xvec)$
by(*rule name-list-avoiding*[**where** *xvec*=*xvec* **and** c=(N, Q', P, M, A_P')]) *auto*
from *Trans* ‹α=$M(\!|\nu*xvec|\!)\langle N\rangle$› **have** $\Psi \otimes \Psi_P \rhd Q \longmapsto M(\!|\nu*xvec|\!)\langle N\rangle \prec Q'$

by *simp*
with ‹$(p \cdot xvec)\ \sharp*\ N$› ‹$(p \cdot xvec)\ \sharp*\ Q'$› S
have $\Psi \otimes \Psi_P \rhd Q \longmapsto M\langle\!|\nu*(p \cdot xvec)|\!\rangle\langle(p \cdot N)\rangle \prec (p \cdot Q')$
by(*simp add*: *boundOutputChainAlpha''* *create-residual.simps*)
moreover from ‹$xvec\ \sharp*\ A_P'$› ‹$(p \cdot xvec)\ \sharp*\ A_P'$› ‹$A_P'\ \sharp*\ \alpha$› S
have $A_P'\ \sharp*\ (p \cdot \alpha)$ **by**(*simp add*: *freshChainSimps del*: *actionFreshChain*)
ultimately have $\Psi \rhd P \parallel Q \longmapsto M\langle\!|\nu*(p \cdot xvec)|\!\rangle\langle(p \cdot N)\rangle \prec P \parallel (p \cdot Q')$
using *FrP* ‹$A_P'\ \sharp*\ \Psi$› ‹$A_P'\ \sharp*\ Q$› ‹*distinct* A_P'› ‹$(p \cdot xvec)\ \sharp*\ P$› ‹$A_P'\ \sharp*\ \alpha$›
‹$(p \cdot xvec)\ \sharp*\ M$› ‹$\alpha = M\langle\!|\nu*xvec|\!\rangle\langle N\rangle$›
by(*force intro*: *Goal*)
with ‹$(p \cdot xvec)\ \sharp*\ N$› ‹$(p \cdot xvec)\ \sharp*\ Q'$› ‹$(p \cdot xvec)\ \sharp*\ P$› ‹$xvec\ \sharp*\ P$› S ‹$\alpha = M\langle\!|\nu*xvec|\!\rangle\langle N\rangle$›
show *?case*
by(*simp add*: *boundOutputChainAlpha''* *eqvts* *create-residual.simps*)
next
case(*cBrOutput M xvec N*)
from ‹$\alpha = ¡M\langle\!|\nu*xvec|\!\rangle\langle N\rangle$› ‹$A_P'\ \sharp*\ \alpha$› ‹$bn\ \alpha\ \sharp*\ P$› **have** $xvec\ \sharp*\ A_P'$ **and** $xvec\ \sharp*\ P$
by *simp+*
obtain p **where** $(p \cdot xvec)\ \sharp*\ N$ **and** $(p \cdot xvec)\ \sharp*\ Q'$ **and** $(p \cdot xvec)\ \sharp*\ P$
and $(p \cdot xvec)\ \sharp*\ M$ **and** $(p \cdot xvec)\ \sharp*\ A_P'$
and S: *set* $p \subseteq$ *set* $xvec \times$ *set*$(p \cdot xvec)$
by(*rule name-list-avoiding*[**where** *xvec*=*xvec* **and** c=(N, Q', P, M, A_P')])
auto
from *Trans* ‹α=$¡M\langle\!|\nu*xvec|\!\rangle\langle N\rangle$› **have** $\Psi \otimes \Psi_P \rhd Q \longmapsto ¡M\langle\!|\nu*xvec|\!\rangle\langle N\rangle \prec Q'$
by *simp*
with ‹$(p \cdot xvec)\ \sharp*\ N$› ‹$(p \cdot xvec)\ \sharp*\ Q'$› S
have $\Psi \otimes \Psi_P \rhd Q \longmapsto ¡M\langle\!|\nu*(p \cdot xvec)|\!\rangle\langle(p \cdot N)\rangle \prec (p \cdot Q')$
by(*simp add*: *boundOutputChainAlpha''* *create-residual.simps*)
moreover from ‹$xvec\ \sharp*\ A_P'$› ‹$(p \cdot xvec)\ \sharp*\ A_P'$› ‹$A_P'\ \sharp*\ \alpha$› S
have $A_P'\ \sharp*\ (p \cdot \alpha)$ **by**(*simp add*: *freshChainSimps del*: *actionFreshChain*)
ultimately have $\Psi \rhd P \parallel Q \longmapsto ¡M\langle\!|\nu*(p \cdot xvec)|\!\rangle\langle(p \cdot N)\rangle \prec P \parallel (p \cdot Q')$
using *FrP* ‹$A_P'\ \sharp*\ \Psi$› ‹$A_P'\ \sharp*\ Q$› ‹*distinct* A_P'› ‹$(p \cdot xvec)\ \sharp*\ P$› ‹$A_P'\ \sharp*\ \alpha$›
‹$(p \cdot xvec)\ \sharp*\ M$› ‹$\alpha = ¡M\langle\!|\nu*xvec|\!\rangle\langle N\rangle$›
by(*force intro*: *Goal*)
with ‹$(p \cdot xvec)\ \sharp*\ N$› ‹$(p \cdot xvec)\ \sharp*\ Q'$› ‹$(p \cdot xvec)\ \sharp*\ P$› ‹$xvec\ \sharp*\ P$› S ‹$\alpha = ¡M\langle\!|\nu*xvec|\!\rangle\langle N\rangle$›
show *?case*
by(*simp add*: *boundOutputChainAlpha''* *eqvts* *create-residual.simps*)
qed
qed

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 −
from *Trans* **have** *distinct*$(xvec@yvec)$ **by**(*force dest*: *boundOutputDistinct*)
then have $xvec \sharp* yvec$ **by**(*induct xvec*) *auto*

obtain p **where** $(p \cdot yvec) \sharp* \Psi$ **and** $(p \cdot yvec) \sharp* P$ **and** $(p \cdot yvec) \sharp* M$
and $(p \cdot yvec) \sharp* yvec$ **and** $(p \cdot yvec) \sharp* N$ **and** $(p \cdot yvec) \sharp* P'$
and $x \sharp (p \cdot yvec)$ **and** $(p \cdot yvec) \sharp* xvec$
and *Sp*: $(set\ p) \subseteq (set\ yvec) \times (set(p \cdot yvec))$
by(*rule name-list-avoiding*[**where** *xvec*=*yvec* **and** *c*=(Ψ, P, M, *xvec*, *yvec*, N, P', x)])
(*auto simp add*: *eqvts fresh-star-prod*)
obtain q **where** $(q \cdot xvec) \sharp* \Psi$ **and** $(q \cdot xvec) \sharp* P$ **and** $(q \cdot xvec) \sharp* M$
and $(q \cdot xvec) \sharp* xvec$ **and** $(q \cdot xvec) \sharp* N$ **and** $(q \cdot xvec) \sharp* P'$
and $x \sharp (q \cdot xvec)$ **and** $(q \cdot xvec) \sharp* yvec$
and $(q \cdot xvec) \sharp* p$ **and** $(q \cdot xvec) \sharp* (p \cdot yvec)$
and *Sq*: $(set\ q) \subseteq (set\ xvec) \times (set(q \cdot xvec))$
by(*rule name-list-avoiding*[**where** *xvec*=*xvec* **and** *c*=(Ψ, P, M, *xvec*, *yvec*, $p \cdot yvec$, N, P', x, p)])
(*auto simp add*: *eqvts fresh-star-prod*)

note $\langle\Psi \rhd P \longmapsto M(\!|\nu*(xvec@yvec)|\!)\langle N\rangle \prec P'\rangle$
moreover from $\langle(p \cdot yvec) \sharp* N\rangle$ $\langle(q \cdot xvec) \sharp* N\rangle$ $\langle xvec \sharp* yvec\rangle$ $\langle(q \cdot xvec) \sharp* yvec\rangle$ $\langle(q \cdot xvec) \sharp* (p \cdot yvec)\rangle$ $\langle(p \cdot yvec) \sharp* xvec\rangle$ *Sp Sq*
have $((p@q) \cdot (xvec\ @\ yvec)) \sharp* N$
apply(*simp only*: *eqvts*)
apply(*simp only*: *pt2*[*OF pt-name-inst*])
by *simp*
moreover from $\langle(p \cdot yvec) \sharp* P'\rangle$ $\langle(q \cdot xvec) \sharp* P'\rangle$ $\langle xvec \sharp* yvec\rangle$ $\langle(q \cdot xvec) \sharp* yvec\rangle$ $\langle(q \cdot xvec) \sharp* (p \cdot yvec)\rangle$ $\langle(p \cdot yvec) \sharp* xvec\rangle$ *Sp Sq*
have $((p@q) \cdot (xvec\ @\ yvec)) \sharp* P'$ **by**(*simp del*: *freshAlphaPerm add*: *eqvts pt2*[*OF pt-name-inst*])
moreover from *Sp Sq* $\langle xvec \sharp* yvec\rangle$ $\langle(q \cdot xvec) \sharp* yvec\rangle$ $\langle(q \cdot xvec) \sharp* (p \cdot yvec)\rangle$ $\langle(p \cdot yvec) \sharp* xvec\rangle$
have *Spq*: $set(p@q) \subseteq set(xvec@yvec) \times set((p@q) \cdot (xvec@yvec))$
by(*simp add*: *pt2*[*OF pt-name-inst*] *eqvts*) *blast*
ultimately have $\Psi \rhd P \longmapsto M(\!|\nu*((p@q) \cdot (xvec@yvec))|\!)\langle((p@q) \cdot N)\rangle \prec ((p@q) \cdot P')$
apply(*simp add*: *create-residual.simps*)

by(*erule rev-mp*) (*subst boundOutputChainAlpha*, *auto*)

with *Sp Sq* ‹*xvec* ♯* *yvec*› ‹(*q* · *xvec*) ♯* *yvec*› ‹(*q* · *xvec*) ♯* (*p* · *yvec*)› ‹(*p* · *yvec*) ♯* *xvec*›
have Ψ ▷ *P* ⟼*M*⦇*ν*∗((*q* · *xvec*)@(*p* · *yvec*))⦈⟨((*p*@*q*) · *N*)⟩ ≺ ((*p*@*q*) · *P*′)
by(*simp add*: *eqvts pt2*[*OF pt-name-inst*] *del*: *freshAlphaPerm*)
moreover from ‹*x* ∈ *supp N*› **have** ((*p*@*q*) · *x*) ∈ (*p*@*q*) · (*supp N*)
by(*simp add*: *pt-set-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹*x* ♯ *xvec*› ‹*x* ♯ *yvec*› ‹*x* ♯ (*q* · *xvec*)› ‹*x* ♯ (*p* · *yvec*)› *Sp Sq*
have *x* ∈ *supp*((*p*@*q*)· *N*) **by**(*simp add*: *eqvts pt2*[*OF pt-name-inst*])
moreover from ‹*distinct*(*xvec*@*yvec*)› **have** *distinct*(*q* · *xvec*) **and** *distinct*(*p* · *yvec*)
by *auto*
moreover note ‹*x* ♯ (*q* · *xvec*)› ‹*x* ♯ (*p* · *yvec*)› ‹*x* ♯ *M*› ‹*x* ♯ Ψ›
‹(*q* · *xvec*) ♯* Ψ› ‹(*q* · *xvec*) ♯* *P*› ‹(*q* · *xvec*) ♯* *M*› ‹(*q* · *xvec*) ♯* (*p* · *yvec*)›
‹(*p* · *yvec*) ♯* Ψ› ‹(*p* · *yvec*) ♯* *P*› ‹(*p* · *yvec*) ♯* *M*› ‹*distinct*(*q* · *xvec*)›
ultimately have Ψ ▷ ⦇*νx*⦈*P* ⟼*M*⦇*ν*∗((*q* · *xvec*)@*x*#(*p* · *yvec*))⦈⟨((*p*@*q*) · *N*)⟩ ≺ ((*p*@*q*) · *P*′)
by(*metis cOpen*)
with ‹*x* ♯ *xvec*› ‹*x* ♯ *yvec*› ‹*x* ♯ (*q* · *xvec*)› ‹*x* ♯ (*p* · *yvec*)›
‹*xvec* ♯* *yvec*› ‹(*q* · *xvec*) ♯* *yvec*› ‹(*q* · *xvec*) ♯* (*p* · *yvec*)› ‹(*p* · *yvec*) ♯* *xvec*› *Sp Sq*
have Ψ ▷ ⦇*νx*⦈*P* ⟼*M*⦇*ν*∗((*p*@*q*) · (*xvec*@*x*#*yvec*))⦈⟨((*p*@*q*) · *N*)⟩ ≺ ((*p*@*q*) · *P*′)
by(*simp add*: *eqvts pt2*[*OF pt-name-inst*] *del*: *freshAlphaPerm*)
then show *?thesis* **using** ‹((*p*@*q*) · (*xvec* @ *yvec*)) ♯* *N*› ‹((*p*@*q*) · (*xvec* @ *yvec*)) ♯* *P*′› *Spq*
apply(*simp add*: *create-residual.simps*)
by(*erule rev-mp*) (*subst boundOutputChainAlpha*, *auto*)
qed

lemma *BrOpen*:
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*: Ψ ▷ *P* ⟼¡*M*⦇*ν*∗(*xvec*@*yvec*)⦈⟨*N*⟩ ≺ *P*′
and *x* ∈ *supp N*
and *x* ♯ Ψ
and *x* ♯ *M*
and *x* ♯ *xvec*
and *x* ♯ *yvec*

shows Ψ ▷ ⦇*νx*⦈*P* ⟼¡*M*⦇*ν*∗(*xvec*@*x*#*yvec*)⦈⟨*N*⟩ ≺ *P*′

proof −
from *Trans* **have** *distinct*(*xvec*@*yvec*) **by**(*force dest*: *boundOutputDistinct*)
then have *xvec* ♯* *yvec* **by**(*induct xvec*) *auto*

obtain *p* **where** (*p* ∙ *yvec*) ♯* Ψ **and** (*p* ∙ *yvec*) ♯* *P* **and** (*p* ∙ *yvec*) ♯* *M*
and (*p* ∙ *yvec*) ♯* *yvec* **and** (*p* ∙ *yvec*) ♯* *N* **and** (*p* ∙ *yvec*) ♯* *P′*
and *x* ♯ (*p* ∙ *yvec*) **and** (*p* ∙ *yvec*) ♯* *xvec*
and *Sp*: (*set p*) ⊆ (*set yvec*) × (*set*(*p* ∙ *yvec*))
by(*rule name-list-avoiding*[**where** *xvec*=*yvec* **and** *c*=(Ψ, *P*, *M*, *xvec*, *yvec*, *N*, *P′*, *x*)])
(*auto simp add*: *eqvts fresh-star-prod*)
obtain *q* **where** (*q* ∙ *xvec*) ♯* Ψ **and** (*q* ∙ *xvec*) ♯* *P* **and** (*q* ∙ *xvec*) ♯* *M*
and (*q* ∙ *xvec*) ♯* *xvec* **and** (*q* ∙ *xvec*) ♯* *N* **and** (*q* ∙ *xvec*) ♯* *P′*
and *x* ♯ (*q* ∙ *xvec*) **and** (*q* ∙ *xvec*) ♯* *yvec*
and (*q* ∙ *xvec*) ♯* *p* **and** (*q* ∙ *xvec*) ♯* (*p* ∙ *yvec*)
and *Sq*: (*set q*) ⊆ (*set xvec*) × (*set*(*q* ∙ *xvec*))
by(*rule name-list-avoiding*[**where** *xvec*=*xvec* **and** *c*=(Ψ, *P*, *M*, *xvec*, *yvec*, *p* ∙ *yvec*, *N*, *P′*, *x*, *p*)])
(*auto simp add*: *eqvts fresh-star-prod*)

note ‹Ψ ▷ *P* ⟼¡*M*⦇*ν**(*xvec*@*yvec*)⦈⟨*N*⟩ ≺ *P′*›
moreover from ‹(*p* ∙ *yvec*) ♯* *N*› ‹(*q* ∙ *xvec*) ♯* *N*› ‹*xvec* ♯* *yvec*› ‹(*q* ∙ *xvec*) ♯* *yvec*› ‹(*q* ∙ *xvec*) ♯* (*p* ∙ *yvec*)› ‹(*p* ∙ *yvec*) ♯* *xvec*› *Sp Sq*
have ((*p*@*q*) ∙ (*xvec* @ *yvec*)) ♯* *N*
apply(*simp only*: *eqvts*)
apply(*simp only*: *pt2*[*OF pt-name-inst*])
by *simp*
moreover from ‹(*p* ∙ *yvec*) ♯* *P′*› ‹(*q* ∙ *xvec*) ♯* *P′*› ‹*xvec* ♯* *yvec*› ‹(*q* ∙ *xvec*) ♯* *yvec*› ‹(*q* ∙ *xvec*) ♯* (*p* ∙ *yvec*)› ‹(*p* ∙ *yvec*) ♯* *xvec*› *Sp Sq*
have ((*p*@*q*) ∙ (*xvec* @ *yvec*)) ♯* *P′* **by**(*simp del*: *freshAlphaPerm add*: *eqvts pt2*[*OF pt-name-inst*])
moreover from *Sp Sq* ‹*xvec* ♯* *yvec*› ‹(*q* ∙ *xvec*) ♯* *yvec*› ‹(*q* ∙ *xvec*) ♯* (*p* ∙ *yvec*)› ‹(*p* ∙ *yvec*) ♯* *xvec*›
have *Spq*: *set*(*p*@*q*) ⊆ *set*(*xvec*@*yvec*) × *set*((*p*@*q*) ∙ (*xvec*@*yvec*))
by(*simp add*: *pt2*[*OF pt-name-inst*] *eqvts*) *blast*
ultimately have Ψ ▷ *P* ⟼¡*M*⦇*ν**((*p*@*q*) ∙ (*xvec*@*yvec*))⦈⟨((*p*@*q*) ∙ *N*)⟩ ≺ ((*p*@*q*) ∙ *P′*)
apply(*simp add*: *create-residual.simps*)
by(*erule rev-mp*) (*subst boundOutputChainAlpha*, *auto*)

with *Sp Sq* ‹*xvec* ♯* *yvec*› ‹(*q* ∙ *xvec*) ♯* *yvec*› ‹(*q* ∙ *xvec*) ♯* (*p* ∙ *yvec*)› ‹(*p* ∙ *yvec*) ♯* *xvec*›
have Ψ ▷ *P* ⟼¡*M*⦇*ν**((*q* ∙ *xvec*)@(*p* ∙ *yvec*))⦈⟨((*p*@*q*) ∙ *N*)⟩ ≺ ((*p*@*q*) ∙ *P′*)
by(*simp add*: *eqvts pt2*[*OF pt-name-inst*] *del*: *freshAlphaPerm*)
moreover from ‹*x* ∈ *supp N*› **have** ((*p*@*q*) ∙ *x*) ∈ (*p*@*q*) ∙ (*supp N*)
by(*simp add*: *pt-set-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹*x* ♯ *xvec*› ‹*x* ♯ *yvec*› ‹*x* ♯ (*q* ∙ *xvec*)› ‹*x* ♯ (*p* ∙ *yvec*)› *Sp Sq*
have *x* ∈ *supp*((*p*@*q*)∙ *N*) **by**(*simp add*: *eqvts pt2*[*OF pt-name-inst*])
moreover from ‹*distinct*(*xvec*@*yvec*)› **have** *distinct*(*q* ∙ *xvec*) **and** *distinct*(*p* ∙

yvec)
by *auto*
moreover note ‹$x \sharp (q \cdot xvec)$› ‹$x \sharp (p \cdot yvec)$› ‹$x \sharp M$› ‹$x \sharp \Psi$›
‹$(q \cdot xvec) \sharp* \Psi$› ‹$(q \cdot xvec) \sharp* P$› ‹$(q \cdot xvec) \sharp* M$› ‹$(q \cdot xvec) \sharp* (p \cdot yvec)$›
‹$(p \cdot yvec) \sharp* \Psi$› ‹$(p \cdot yvec) \sharp* P$› ‹$(p \cdot yvec) \sharp* M$› ‹$distinct(q \cdot xvec)$›
ultimately have $\Psi \rhd (\!|\nu x|\!)P \longmapsto ¡M (\!|\nu*((q \cdot xvec)@x\#(p \cdot yvec))|\!) \langle((p@q) \cdot N)\rangle \prec ((p@q) \cdot P')$
by(*metis cBrOpen*)
with ‹$x \sharp xvec$› ‹$x \sharp yvec$› ‹$x \sharp (q \cdot xvec)$› ‹$x \sharp (p \cdot yvec)$›
‹$xvec \sharp* yvec$› ‹$(q \cdot xvec) \sharp* yvec$› ‹$(q \cdot xvec) \sharp* (p \cdot yvec)$› ‹$(p \cdot yvec) \sharp* xvec$›
Sp Sq
have $\Psi \rhd (\!|\nu x|\!)P \longmapsto ¡M (\!|\nu*((p@q) \cdot (xvec@x\#yvec))|\!) \langle((p@q) \cdot N)\rangle \prec ((p@q) \cdot P')$
by(*simp add*: *eqvts pt2*[*OF pt-name-inst*] *del*: *freshAlphaPerm*)
then show *?thesis* **using** ‹$((p@q) \cdot (xvec \ @ \ yvec)) \sharp* N$› ‹$((p@q) \cdot (xvec \ @ \ yvec)) \sharp* P'$› *Spq*
apply(*simp add*: *create-residual.simps*)
by(*erule rev-mp*) (*subst boundOutputChainAlpha*, *auto*)
qed

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 −
{
fix Ψ P M *xvec* N P' x

assume $\Psi \rhd P \longmapsto M (\!|\nu*xvec|\!) \langle N\rangle \prec P'$
and $(x::name) \sharp \Psi$
and $x \sharp M$
and $x \sharp xvec$
and $x \sharp N$

obtain $p::name\ prm$ **where** $(p \cdot xvec) \sharp* \Psi$ **and** $(p \cdot xvec) \sharp* P$ **and** $(p \cdot xvec) \sharp* M$ **and** $(p \cdot xvec) \sharp* xvec$
and $(p \cdot xvec) \sharp* N$ **and** $(p \cdot xvec) \sharp* P'$ **and** $x \sharp (p \cdot xvec)$
and S: $(set\ p) \subseteq (set\ xvec) \times (set(p \cdot xvec))$
by(*rule name-list-avoiding*[**where** *xvec*=*xvec* **and** c=(Ψ, P, M, *xvec*, N, P', x)])
(*auto simp add*: *eqvts fresh-star-prod*)

from ‹$\Psi \rhd P \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$› ‹$(p \cdot xvec) \sharp* N$› ‹$(p \cdot xvec) \sharp* P'$› S
have $\Psi \rhd P \longmapsto M(\!|\nu * (p \cdot xvec)|\!)\langle (p \cdot N)\rangle \prec (p \cdot P')$
by(*simp add*: *boundOutputChainAlpha''* *create-residual.simps*)
moreover then have *distinct*$(p \cdot xvec)$ **by**(*force dest*: *boundOutputDistinct*)
moreover note ‹$x \sharp \Psi$› ‹$x \sharp M$› ‹$x \sharp (p \cdot xvec)$›
moreover from ‹$x \sharp xvec$› ‹$x \sharp p \cdot xvec$› ‹$x \sharp N$› S **have** $x \sharp (p \cdot N)$
by(*simp add*: *fresh-left del*: *freshAlphaSwap*)
ultimately have $\Psi \rhd (\!|\nu x|\!)P \longmapsto M(\!|\nu * (p \cdot xvec)|\!)\langle (p \cdot N)\rangle \prec (\!|\nu x|\!)(p \cdot P')$ **using** ‹$(p \cdot xvec) \sharp* \Psi$› ‹$(p \cdot xvec) \sharp* P$› ‹$(p \cdot xvec) \sharp* M$›
by(*force intro*: *cScope*)
moreover from ‹$x \sharp xvec$› ‹$x \sharp p \cdot xvec$› S **have** $p \cdot x = x$ **by** *simp*
ultimately have $\Psi \rhd (\!|\nu x|\!)P \longmapsto M(\!|\nu * (p \cdot xvec)|\!)\langle (p \cdot N)\rangle \prec (p \cdot ((\!|\nu x|\!)P'))$ **by** *simp*
moreover from ‹$(p \cdot xvec) \sharp* P'$› ‹$x \sharp xvec$› ‹$x \sharp (p \cdot xvec)$› **have** $(p \cdot xvec) \sharp* (\!|\nu x|\!)P'$
by(*simp add*: *abs-fresh-star*)
ultimately have $\Psi \rhd (\!|\nu x|\!)P \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec (\!|\nu x|\!)P'$ **using** ‹$(p \cdot xvec) \sharp* N$› S
by(*simp add*: *boundOutputChainAlpha''* *create-residual.simps*)
}
then have

outputCase: $\bigwedge \Psi\ P\ M\ xvec\ N\ P'\ x.$
$[\![\Psi \rhd P \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P';$
$(x::name) \sharp \Psi;$
$x \sharp M;$
$x \sharp xvec;$
$x \sharp N]\!] \Longrightarrow$
$\Psi \rhd (\!|\nu x|\!)P \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec (\!|\nu x|\!)P'$ **by** *simp*

{
fix $\Psi\ P\ M\ xvec\ N\ P'\ x$

assume $\Psi \rhd P \longmapsto ¡ M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$
and $(x::name) \sharp \Psi$
and $x \sharp M$
and $x \sharp xvec$
and $x \sharp N$

obtain $p::name\ prm$ **where** $(p \cdot xvec) \sharp* \Psi$ **and** $(p \cdot xvec) \sharp* P$ **and** $(p \cdot xvec) \sharp* M$ **and** $(p \cdot xvec) \sharp* xvec$
and $(p \cdot xvec) \sharp* N$ **and** $(p \cdot xvec) \sharp* P'$ **and** $x \sharp (p \cdot xvec)$
and S: $(set\ p) \subseteq (set\ xvec) \times (set(p \cdot xvec))$
by(*rule name-list-avoiding*[**where** *xvec*=*xvec* **and** *c*=(Ψ, P, M, *xvec*, N, P', x)])
(*auto simp add*: *eqvts fresh-star-prod*)
from ‹$\Psi \rhd P \longmapsto ¡ M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$› ‹$(p \cdot xvec) \sharp* N$› ‹$(p \cdot xvec) \sharp* P'$› S
have $\Psi \rhd P \longmapsto ¡ M(\!|\nu * (p \cdot xvec)|\!)\langle (p \cdot N)\rangle \prec (p \cdot P')$
by(*simp add*: *boundOutputChainAlpha''* *create-residual.simps*)

moreover then have *distinct*$(p \cdot xvec)$ **by**(*force dest*: *boundOutputDistinct*)
moreover note $\langle x \sharp \Psi\rangle$ $\langle x \sharp M\rangle$ $\langle x \sharp (p \cdot xvec)\rangle$
moreover from $\langle x \sharp xvec\rangle$ $\langle x \sharp p \cdot xvec\rangle$ $\langle x \sharp N\rangle$ S **have** $x \sharp (p \cdot N)$
by(*simp add*: *fresh-left del*: *freshAlphaSwap*)
ultimately have $\Psi \rhd (\!|\nu x|\!)P \longmapsto\!\!¡ M(\!|\nu*(p \cdot xvec)|\!)\langle(p \cdot N)\rangle \prec (\!|\nu x|\!)(p \cdot P')$
using $\langle(p \cdot xvec) \sharp* \Psi\rangle$ $\langle(p \cdot xvec) \sharp* P\rangle$ $\langle(p \cdot xvec) \sharp* M\rangle$
by(*force simp add*: *cScope*)
moreover from $\langle x \sharp xvec\rangle$ $\langle x \sharp p \cdot xvec\rangle$ S **have** $p \cdot x = x$ **by** *simp*
ultimately have $\Psi \rhd (\!|\nu x|\!)P \longmapsto\!\!¡ M(\!|\nu*(p \cdot xvec)|\!)\langle(p \cdot N)\rangle \prec (p \cdot ((\!|\nu x|\!)P'))$
by *simp*
moreover from $\langle(p \cdot xvec) \sharp* P'\rangle$ $\langle x \sharp xvec\rangle$ $\langle x \sharp (p \cdot xvec)\rangle$ **have** $(p \cdot xvec) \sharp* (\!|\nu x|\!)P'$
by(*simp add*: *abs-fresh-star*)
ultimately have $\Psi \rhd (\!|\nu x|\!)P \longmapsto\!\!¡ M(\!|\nu*xvec|\!)\langle N\rangle \prec (\!|\nu x|\!)P'$ **using** $\langle(p \cdot xvec) \sharp* N\rangle$ S
by(*simp add*: *boundOutputChainAlpha''* *create-residual.simps*)
}
then have

broutputCase: $\bigwedge \Psi\ P\ M\ xvec\ N\ P'\ x.$
$[\![\Psi \rhd P \longmapsto\!\!¡ M(\!|\nu*xvec|\!)\langle N\rangle \prec P';$
$(x::name) \sharp \Psi;$
$x \sharp M;$
$x \sharp xvec;$
$x \sharp N]\!] \Longrightarrow$
$\Psi \rhd (\!|\nu x|\!)P \longmapsto\!\!¡ M(\!|\nu*xvec|\!)\langle N\rangle \prec (\!|\nu x|\!)P'$ **by** *simp*

show *?thesis*
proof(*induct rule*: *actionCases*[**where** $\alpha=\alpha$])
case(*cInput M N*)
with *assms* **show** *?case* **by**(*force intro*: *cScope*)
next
case(*cBrInput M N*)
with *assms* **show** *?case* **by**(*force intro*: *cScope*)
next
case(*cOutput M xvec N*)
with *assms* **show** *?case* **by**(*force intro*: *outputCase*)
next
case(*cBrOutput M xvec N*)
with *assms* **show** *?case* **by**(*force intro*: *broutputCase*)
next
case *cTau*
with *assms* **show** *?case* **by**(*force intro*: *cScope*)
qed
qed

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 $x \sharp \Psi$
and $x \sharp P$
and $y \sharp \Psi$
and $y \sharp P$

shows $([(x, y)] \cdot \Psi) \rhd P \longmapsto ([(x, y)] \cdot M)(\!|N|\!) \prec P'$
using *assms*
proof(*nominal-induct avoiding*: *x y rule*: *inputInduct*)
case(*cInput* Ψ M K *xvec* N *Tvec* P x y)
from ‹$x \sharp M(\!|\lambda* xvec\ N|\!).P$› **have** $x \sharp M$ **by** *simp*
from ‹$y \sharp M(\!|\lambda* xvec\ N|\!).P$› **have** $y \sharp M$ **by** *simp*
from ‹$\Psi \vdash M \leftrightarrow K$› **have** $([(x, y)] \cdot \Psi) \vdash ([(x, y)] \cdot M) \leftrightarrow ([(x, y)] \cdot K)$
by(*rule chanEqClosed*)
with ‹$x \sharp M$› ‹$y \sharp M$› **have** $([(x, y)] \cdot \Psi) \vdash M \leftrightarrow ([(x, y)] \cdot K)$
by(*simp*)
then show *?case* **using** ‹*distinct xvec*› ‹*set xvec* ⊆ *supp N*› ‹*length xvec* = *length Tvec*›
by(*rule Input*)
next
case(*cCase* Ψ P M N P' φ *Cs* x y)
from ‹$x \sharp$ *Cases Cs*› ‹$y \sharp$ *Cases Cs*› ‹$(\varphi, P) \in$ *set Cs*› **have** $x \sharp \varphi$ **and** $x \sharp P$ **and** $y \sharp \varphi$ **and** $y \sharp P$
by(*auto dest*: *memFresh*)
from ‹$x \sharp P$› ‹$y \sharp P$› **have** $([(x\ ,y)] \cdot \Psi) \rhd P \longmapsto ([(x, y)] \cdot M)(\!|N|\!) \prec P'$ **by**(*rule cCase*)
moreover note ‹$(\varphi, P) \in$ *set Cs*›
moreover from ‹$\Psi \vdash \varphi$› **have** $([(x, y)] \cdot \Psi) \vdash ([(x, y)] \cdot \varphi)$ **by**(*rule statClosed*)
with ‹$x \sharp \varphi$› ‹$y \sharp \varphi$› **have** $([(x, y)] \cdot \Psi) \vdash \varphi$ **by** *simp*
ultimately show *?case* **using** ‹*guarded P*› **by**(*rule Case*)
next
case(*cPar1* Ψ Ψ_Q P M N P' A_Q Q x y)
from ‹$x \sharp P \parallel Q$› **have** $x \sharp P$ **and** $x \sharp Q$ **by** *simp+*
from ‹$y \sharp P \parallel Q$› **have** $y \sharp P$ **and** $y \sharp Q$ **by** *simp+*
from ‹$x \sharp P$› ‹$y \sharp P$› ‹$\bigwedge x\ y.\ [\![x \sharp P; y \sharp P]\!] \Longrightarrow ([(x, y)] \cdot (\Psi \otimes \Psi_Q)) \rhd P \longmapsto ([(x, y)] \cdot M)(\!|N|\!) \prec P'$›
have $([(x, y)] \cdot \Psi) \otimes ([(x, y)] \cdot \Psi_Q) \rhd P \longmapsto ([(x, y)] \cdot M)(\!|N|\!) \prec P'$
by(*simp add*: *eqvts*)

moreover from ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› **have** $([(x, y)] \cdot (\mathit{extractFrame}\ Q)) = ([(x, y)] \cdot \langle A_Q, \Psi_Q\rangle)$
by *simp*
with ‹$A_Q \sharp* x$› ‹$x \sharp Q$› ‹$A_Q \sharp* y$› ‹$y \sharp Q$› **have** $\langle A_Q, ([(x, y)] \cdot \Psi_Q)\rangle = \mathit{extractFrame}\ Q$
by(*simp add*: *eqvts*)

moreover from ‹$A_Q \sharp* \Psi$› **have** $([(x,\ y)] \cdot A_Q) \sharp* ([(x,\ y)] \cdot \Psi)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$A_Q \sharp* x$› ‹$A_Q \sharp* y$› **have** $A_Q \sharp* ([(x,\ y)] \cdot \Psi)$ **by** *simp*
moreover from ‹$A_Q \sharp* M$› **have** $([(x,\ y)] \cdot A_Q) \sharp* ([(x,\ y)] \cdot M)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$A_Q \sharp* x$› ‹$A_Q \sharp* y$› **have** $A_Q \sharp* ([(x,\ y)] \cdot M)$ **by** *simp*
ultimately show *?case* **using** ‹$A_Q \sharp* P$› ‹$A_Q \sharp* N$›
by(*force intro*!: *Par1*)
next
case(*cPar2* Ψ Ψ_P Q M N Q' A_P P x y)
from ‹$x \sharp P \parallel Q$› **have** $x \sharp P$ **and** $x \sharp Q$ **by** *simp+*
from ‹$y \sharp P \parallel Q$› **have** $y \sharp P$ **and** $y \sharp Q$ **by** *simp+*
from ‹$x \sharp Q$› ‹$y \sharp Q$› ‹$\bigwedge x\ y.\ [\![x \sharp Q;\ y \sharp Q]\!] \Longrightarrow ([(x,\ y)] \cdot (\Psi \otimes \Psi_P)) \rhd Q \longmapsto ([(x,\ y)] \cdot M)(\!|N|\!) \prec Q'$›
have $([(x,\ y)] \cdot \Psi) \otimes ([(x,\ y)] \cdot \Psi_P) \rhd Q \longmapsto ([(x,\ y)] \cdot M)(\!|N|\!) \prec Q'$
by(*simp add*: *eqvts*)

moreover from ‹$extractFrame\ P = \langle A_P,\ \Psi_P \rangle$› **have** $([(x,\ y)] \cdot (extractFrame\ P)) = ([(x,\ y)] \cdot \langle A_P,\ \Psi_P \rangle)$
by *simp*
with ‹$A_P \sharp* x$› ‹$x \sharp P$› ‹$A_P \sharp* y$› ‹$y \sharp P$› **have** $\langle A_P,\ ([(x,\ y)] \cdot \Psi_P) \rangle = extractFrame\ P$
by(*simp add*: *eqvts*)
moreover from ‹$A_P \sharp* \Psi$› **have** $([(x,\ y)] \cdot A_P) \sharp* ([(x,\ y)] \cdot \Psi)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$A_P \sharp* x$› ‹$A_P \sharp* y$› **have** $A_P \sharp* ([(x,\ y)] \cdot \Psi)$ **by** *simp*
moreover from ‹$A_P \sharp* M$› **have** $([(x,\ y)] \cdot A_P) \sharp* ([(x,\ y)] \cdot M)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$A_P \sharp* x$› ‹$A_P \sharp* y$› **have** $A_P \sharp* ([(x,\ y)] \cdot M)$ **by** *simp*
ultimately show *?case* **using** ‹$A_P \sharp* Q$› ‹$A_P \sharp* N$›
by(*force intro*: *Par2*)
next
case(*cScope* Ψ P M N P' z x y)
from ‹$x \sharp (\!|\nu z|\!) P$› ‹$z \sharp x$› **have** $x \sharp P$ **by**(*simp add*: *abs-fresh*)
from ‹$y \sharp (\!|\nu z|\!) P$› ‹$z \sharp y$› **have** $y \sharp P$ **by**(*simp add*: *abs-fresh*)
from ‹$x \sharp P$› ‹$y \sharp P$› ‹$\bigwedge x\ y.\ [\![x \sharp P;\ y \sharp P]\!] \Longrightarrow ([(x,\ y)] \cdot \Psi) \rhd P \longmapsto ([(x,\ y)] \cdot M)(\!|N|\!) \prec P'$›
have $([(x,\ y)] \cdot \Psi) \rhd P \longmapsto ([(x,\ y)] \cdot M)(\!|N|\!) \prec P'$ **by** *simp*
moreover with ‹$z \sharp \Psi$› **have** $([(x,\ y)] \cdot z) \sharp [(x,\ y)] \cdot \Psi$
by(*simp add*: *pt-fresh-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$z \sharp x$› ‹$z \sharp y$› **have** $z \sharp [(x,\ y)] \cdot \Psi$ **by** *simp*
moreover with ‹$z \sharp M$› **have** $([(x,\ y)] \cdot z) \sharp [(x,\ y)] \cdot M$
by(*simp add*: *pt-fresh-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$z \sharp x$› ‹$z \sharp y$› **have** $z \sharp [(x,\ y)] \cdot M$ **by** *simp*
ultimately show *?case* **using** ‹$z \sharp N$›
by(*force intro*!: *Scope*)
next
case(*cBang* Ψ P M N P' x y)
then show *?case* **by**(*force intro*: *Bang*)

qed

lemma *brinputSwapFrameSubject*:
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 $([(x, y)] \cdot \Psi) \rhd P \longmapsto ¿([(x, y)] \cdot M)(\!|N|\!) \prec P'$
using *assms*
proof(*nominal-induct avoiding*: *x y rule*: *brInputInduct*)
case(*cBrInput* Ψ *K M xvec N Tvec P x y*)
from ‹$x \sharp M(\!|\lambda* xvec\ N|\!).P$› **have** $x \sharp M$ **by** *simp*
from ‹$y \sharp M(\!|\lambda* xvec\ N|\!).P$› **have** $y \sharp M$ **by** *simp*
from ‹$\Psi \vdash K \succeq M$› **have** $([(x, y)] \cdot \Psi) \vdash ([(x, y)] \cdot K) \succeq ([(x, y)] \cdot M)$
by(*rule chanInConClosed*)
with ‹$x \sharp M$› ‹$y \sharp M$› **have** $([(x, y)] \cdot \Psi) \vdash ([(x, y)] \cdot K) \succeq M$
by(*simp*)
then show *?case* **using** ‹*distinct xvec*› ‹*set xvec* ⊆ *supp N*› ‹*length xvec* = *length Tvec*›
by(*rule BrInput*)
next
case(*cCase* Ψ *P M N P'* φ *Cs x y*)
from ‹$x \sharp$ *Cases Cs*› ‹$y \sharp$ *Cases Cs*› ‹$(\varphi, P) \in$ *set Cs*› **have** $x \sharp \varphi$ **and** $x \sharp P$ **and** $y \sharp \varphi$ **and** $y \sharp P$
by(*auto dest*: *memFresh*)
from ‹$x \sharp P$› ‹$y \sharp P$› **have** $([(x\ ,y)] \cdot \Psi) \rhd P \longmapsto ¿([(x, y)] \cdot M)(\!|N|\!) \prec P'$ **by**(*rule cCase*)
moreover note ‹$(\varphi, P) \in$ *set Cs*›
moreover from ‹$\Psi \vdash \varphi$› **have** $([(x, y)] \cdot \Psi) \vdash ([(x, y)] \cdot \varphi)$ **by**(*rule statClosed*)
with ‹$x \sharp \varphi$› ‹$y \sharp \varphi$› **have** $([(x, y)] \cdot \Psi) \vdash \varphi$ **by** *simp*
ultimately show *?case* **using** ‹*guarded P*› **by**(*rule Case*)
next
case(*cPar1* Ψ Ψ_Q *P M N P'* A_Q *Q x y*)
from ‹$x \sharp P \parallel Q$› **have** $x \sharp P$ **and** $x \sharp Q$ **by** *simp+*
from ‹$y \sharp P \parallel Q$› **have** $y \sharp P$ **and** $y \sharp Q$ **by** *simp+*
from ‹$x \sharp P$› ‹$y \sharp P$› ‹$\bigwedge x\ y.\ [\![x \sharp P;\ y \sharp P]\!] \Longrightarrow ([(x, y)] \cdot (\Psi \otimes \Psi_Q)) \rhd P \longmapsto ¿([(x, y)] \cdot M)(\!|N|\!) \prec P'$›
have $([(x, y)] \cdot \Psi) \otimes ([(x, y)] \cdot \Psi_Q) \rhd P \longmapsto ¿([(x, y)] \cdot M)(\!|N|\!) \prec P'$
by(*simp add*: *eqvts*)

moreover from ‹*extractFrame Q* = $\langle A_Q, \Psi_Q \rangle$› **have** $([(x, y)] \cdot$ (*extractFrame*

$Q)) = ([(x,\ y)] \cdot \langle A_Q,\ \Psi_Q\rangle)$
by *simp*
with ‹$A_Q \sharp* x$› ‹$x \sharp Q$› ‹$A_Q \sharp* y$› ‹$y \sharp Q$› **have** $\langle A_Q,\ ([(x,\ y)] \cdot \Psi_Q)\rangle =$ *extractFrame Q*
by(*simp add*: *eqvts*)
moreover from ‹$A_Q \sharp* \Psi$› **have** $([(x,\ y)] \cdot A_Q) \sharp* ([(x,\ y)] \cdot \Psi)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$A_Q \sharp* x$› ‹$A_Q \sharp* y$› **have** $A_Q \sharp* ([(x,\ y)] \cdot \Psi)$ **by** *simp*
moreover from ‹$A_Q \sharp* M$› **have** $([(x,\ y)] \cdot A_Q) \sharp* ([(x,\ y)] \cdot M)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$A_Q \sharp* x$› ‹$A_Q \sharp* y$› **have** $A_Q \sharp* ([(x,\ y)] \cdot M)$ **by** *simp*
ultimately show *?case* **using** ‹$A_Q \sharp* P$› ‹$A_Q \sharp* N$›
by(*force intro*!: *Par1*)
next
case(*cPar2* $\Psi\ \Psi_P\ Q\ M\ N\ Q'\ A_P\ P\ x\ y$)
from ‹$x \sharp P \parallel Q$› **have** $x \sharp P$ **and** $x \sharp Q$ **by** *simp+*
from ‹$y \sharp P \parallel Q$› **have** $y \sharp P$ **and** $y \sharp Q$ **by** *simp+*
from ‹$x \sharp Q$› ‹$y \sharp Q$› ‹$\bigwedge x\ y.\ [\![x \sharp Q;\ y \sharp Q]\!] \Longrightarrow ([(x,\ y)] \cdot (\Psi \otimes \Psi_P)) \rhd Q \longmapsto ¿([(x,\ y)] \cdot M)(\!|N|\!) \prec Q'$›
have $([(x,\ y)] \cdot \Psi) \otimes ([(x,\ y)] \cdot \Psi_P) \rhd Q \longmapsto ¿([(x,\ y)] \cdot M)(\!|N|\!) \prec Q'$
by(*simp add*: *eqvts*)

moreover from ‹*extractFrame* $P = \langle A_P,\ \Psi_P\rangle$› **have** $([(x,\ y)] \cdot$ (*extractFrame* $P)) = ([(x,\ y)] \cdot \langle A_P,\ \Psi_P\rangle)$
by *simp*
with ‹$A_P \sharp* x$› ‹$x \sharp P$› ‹$A_P \sharp* y$› ‹$y \sharp P$› **have** $\langle A_P,\ ([(x,\ y)] \cdot \Psi_P)\rangle =$ *extractFrame P*
by(*simp add*: *eqvts*)
moreover from ‹$A_P \sharp* \Psi$› **have** $([(x,\ y)] \cdot A_P) \sharp* ([(x,\ y)] \cdot \Psi)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$A_P \sharp* x$› ‹$A_P \sharp* y$› **have** $A_P \sharp* ([(x,\ y)] \cdot \Psi)$ **by** *simp*
moreover from ‹$A_P \sharp* M$› **have** $([(x,\ y)] \cdot A_P) \sharp* ([(x,\ y)] \cdot M)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$A_P \sharp* x$› ‹$A_P \sharp* y$› **have** $A_P \sharp* ([(x,\ y)] \cdot M)$ **by** *simp*
ultimately show *?case* **using** ‹$A_P \sharp* Q$› ‹$A_P \sharp* N$›
by(*force intro*!: *Par2*)
next
case (*cBrMerge* $\Psi\ \Psi_Q\ P\ M\ N\ P'\ A_P\ \Psi_P\ Q\ Q'\ A_Q\ x\ y$)
from ‹$x \sharp P \parallel Q$› **have** $x \sharp P$ **and** $x \sharp Q$ **by** *simp+*
from ‹$y \sharp P \parallel Q$› **have** $y \sharp P$ **and** $y \sharp Q$ **by** *simp+*

from ‹*extractFrame* $P = \langle A_P,\ \Psi_P\rangle$› **have** $([(x,\ y)] \cdot$ (*extractFrame* $P)) = ([(x,\ y)] \cdot \langle A_P,\ \Psi_P\rangle)$
by *simp*
with ‹$A_P \sharp* x$› ‹$x \sharp P$› ‹$A_P \sharp* y$› ‹$y \sharp P$› **have** *extractFrame* $P = \langle A_P,\ ([(x,\ y)] \cdot \Psi_P)\rangle$
by(*simp add*: *eqvts*)

from ‹*extractFrame* $Q = \langle A_Q,\ \Psi_Q\rangle$› **have** $([(x,\ y)] \cdot$ (*extractFrame* $Q)) = ([(x,$

$y)] \cdot \langle A_Q, \Psi_Q \rangle$)
  by *simp*
 with ‹$A_Q \sharp* x$› ‹$x \sharp Q$› ‹$A_Q \sharp* y$› ‹$y \sharp Q$› **have** *extractFrame* $Q = \langle A_Q, ([(x, y)] \cdot \Psi_Q) \rangle$
  by(*simp add*: *eqvts*)

 from ‹$x \sharp P$› ‹$y \sharp P$› ‹$\bigwedge x\ y.\ [\![x \sharp P; y \sharp P]\!] \Longrightarrow ([(x, y)] \cdot (\Psi \otimes \Psi_Q)) \rhd P \longmapsto ¿([(x, y)] \cdot M)\langle\!|N|\!\rangle \prec P'$›
 have $([(x, y)] \cdot \Psi) \otimes ([(x, y)] \cdot \Psi_Q) \rhd P \longmapsto ¿([(x, y)] \cdot M)\langle\!|N|\!\rangle \prec P'$
  by(*simp add*: *eqvts*)
 moreover from ‹$x \sharp Q$› ‹$y \sharp Q$› ‹$\bigwedge x\ y.\ [\![x \sharp Q; y \sharp Q]\!] \Longrightarrow ([(x, y)] \cdot (\Psi \otimes \Psi_P)) \rhd Q \longmapsto ¿([(x, y)] \cdot M)\langle\!|N|\!\rangle \prec Q'$›
 have $([(x, y)] \cdot \Psi) \otimes ([(x, y)] \cdot \Psi_P) \rhd Q \longmapsto ¿([(x, y)] \cdot M)\langle\!|N|\!\rangle \prec Q'$
  by(*simp add*: *eqvts*)
 moreover note ‹*extractFrame* $P = \langle A_P, ([(x, y)] \cdot \Psi_P) \rangle$› ‹*extractFrame* $Q = \langle A_Q, ([(x, y)] \cdot \Psi_Q) \rangle$›

 moreover from ‹$A_P \sharp* \Psi$› **have** $([(x, y)] \cdot A_P) \sharp* ([(x, y)] \cdot \Psi)$
  by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
 with ‹$A_P \sharp* x$› ‹$A_P \sharp* y$› **have** $A_P \sharp* ([(x, y)] \cdot \Psi)$ **by** *simp*
 moreover from ‹$A_P \sharp* \Psi_Q$› **have** $([(x, y)] \cdot A_P) \sharp* ([(x, y)] \cdot \Psi_Q)$
  by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
 with ‹$A_P \sharp* x$› ‹$A_P \sharp* y$› **have** $A_P \sharp* ([(x, y)] \cdot \Psi_Q)$ **by** *simp*
 moreover from ‹$A_P \sharp* M$› **have** $([(x, y)] \cdot A_P) \sharp* ([(x, y)] \cdot M)$
  by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
 with ‹$A_P \sharp* x$› ‹$A_P \sharp* y$› **have** $A_P \sharp* ([(x, y)] \cdot M)$ **by** *simp*

 moreover from ‹$A_Q \sharp* \Psi$› **have** $([(x, y)] \cdot A_Q) \sharp* ([(x, y)] \cdot \Psi)$
  by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
 with ‹$A_Q \sharp* x$› ‹$A_Q \sharp* y$› **have** $A_Q \sharp* ([(x, y)] \cdot \Psi)$ **by** *simp*
 moreover from ‹$A_Q \sharp* \Psi_P$› **have** $([(x, y)] \cdot A_Q) \sharp* ([(x, y)] \cdot \Psi_P)$
  by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
 with ‹$A_Q \sharp* x$› ‹$A_Q \sharp* y$› **have** $A_Q \sharp* ([(x, y)] \cdot \Psi_P)$ **by** *simp*
 moreover from ‹$A_Q \sharp* M$› **have** $([(x, y)] \cdot A_Q) \sharp* ([(x, y)] \cdot M)$
  by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
 with ‹$A_Q \sharp* x$› ‹$A_Q \sharp* y$› **have** $A_Q \sharp* ([(x, y)] \cdot M)$ **by** *simp*

 moreover note ‹*distinct* A_P› ‹*distinct* A_Q›
  ‹$A_P \sharp* P$› ‹$A_P \sharp* N$› ‹$A_P \sharp* P'$› ‹$A_P \sharp* Q$› ‹$A_P \sharp* Q'$› ‹$A_P \sharp* A_Q$›
  ‹$A_Q \sharp* P$› ‹$A_Q \sharp* N$› ‹$A_Q \sharp* P'$› ‹$A_Q \sharp* Q$› ‹$A_Q \sharp* Q'$›
 ultimately show *?case*
  by(*force intro*!: *semantics.cBrMerge*)
next
 case(*cScope* Ψ P M N P' z x y)
 from ‹$x \sharp (\!|\nu z|\!)P$› ‹$z \sharp x$› **have** $x \sharp P$ **by**(*simp add*: *abs-fresh*)
 from ‹$y \sharp (\!|\nu z|\!)P$› ‹$z \sharp y$› **have** $y \sharp P$ **by**(*simp add*: *abs-fresh*)
 from ‹$x \sharp P$› ‹$y \sharp P$› ‹$\bigwedge x\ y.\ [\![x \sharp P; y \sharp P]\!] \Longrightarrow ([(x, y)] \cdot \Psi) \rhd P \longmapsto ¿([(x, y)] \cdot M)\langle\!|N|\!\rangle \prec P'$›
 have $([(x, y)] \cdot \Psi) \rhd P \longmapsto ¿([(x, y)] \cdot M)\langle\!|N|\!\rangle \prec P'$ **by** *simp*

```
  moreover with ‹z ♯ Ψ› have ([(x, y)] • z) ♯ [(x, y)] • Ψ
    by(simp add: pt-fresh-bij[OF pt-name-inst, OF at-name-inst])
  with ‹z ♯ x› ‹z ♯ y› have z ♯ [(x, y)] • Ψ by simp
  moreover with ‹z ♯ M› have ([(x, y)] • z) ♯ [(x, y)] • M
    by(simp add: pt-fresh-bij[OF pt-name-inst, OF at-name-inst])
  with ‹z ♯ x› ‹z ♯ y› have z ♯ [(x, y)] • M by simp
  ultimately show ?case using ‹z ♯ N›
    by(force intro: Scope)
next
  case(cBang Ψ P M N P' x y)
  then show ?case by(force intro: Bang)
qed

lemma inputPermFrameSubject:
  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 Xs :: name set
    and Ys :: name set

assumes Ψ ▷ P ⟼M(⦇N⦈) ≺ P'
  and   S: set p ⊆ Xs × Ys
  and   Xs ♯* P
  and   Ys ♯* P

shows (p • Ψ) ▷ P ⟼ (p • M)(⦇N⦈) ≺ P'
  using S
proof(induct p)
  case Nil
  from ‹Ψ ▷ P ⟼M(⦇N⦈) ≺ P'›
  show ?case by simp
next
  case(Cons a p)
  from ‹set(a#p) ⊆ Xs × Ys› have set p ⊆ Xs × Ys by auto
  with ‹set p ⊆ Xs × Ys ⟹ (p • Ψ) ▷ P ⟼ (p • M)(⦇N⦈) ≺ P'›
  have Trans: (p • Ψ) ▷ P ⟼ (p • M)(⦇N⦈) ≺ P' by simp
  from ‹set(a#p) ⊆ Xs × Ys› show ?case
  proof(cases a)
    case (Pair x y)
    then have x ∈ Xs and y ∈ Ys
      using ‹set(a#p) ⊆ Xs × Ys› by auto
    with ‹Xs ♯* P› ‹Ys ♯* P› have x ♯ P and y ♯ P
      by(auto simp add: fresh-star-def)
    with Trans have ([(x, y)] • p • Ψ) ▷ P ⟼ ([(x, y)] • p • M)(⦇N⦈) ≺ P'
      by(rule inputSwapFrameSubject)
    then show ?thesis
```

```
    using Pair by simp
  qed
qed

lemma brinputPermFrameSubject:
  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 Xs :: name set
    and Ys :: name set

assumes Ψ ▷ P ⟼¿M(⦇N⦈) ≺ P'
  and   S: set p ⊆ Xs × Ys
  and   Xs ♯* P
  and   Ys ♯* P

shows (p • Ψ) ▷ P ⟼ ¿(p • M)(⦇N⦈) ≺ P'
  using S
proof(induct p)
  case Nil
  from ‹Ψ ▷ P ⟼¿M(⦇N⦈) ≺ P'›
  show ?case by simp
next
  case(Cons a p)
  from ‹set(a#p) ⊆ Xs × Ys› have set p ⊆ Xs × Ys by auto
  with ‹set p ⊆ Xs × Ys ⟹ (p • Ψ) ▷ P ⟼ ¿(p • M)(⦇N⦈) ≺ P'›
  have Trans: (p • Ψ) ▷ P ⟼ ¿(p • M)(⦇N⦈) ≺ P' by simp
  from ‹set(a#p) ⊆ Xs × Ys› show ?case
  proof(cases a)
    case (Pair x y)
    then have x ∈ Xs and y ∈ Ys
      using ‹set(a#p) ⊆ Xs × Ys› by auto
    with ‹Xs ♯* P› ‹Ys ♯* P› have x ♯ P and y ♯ P
      by(auto simp add: fresh-star-def)
    with Trans have ([(x, y)] • p • Ψ) ▷ P ⟼ ¿([(x, y)] • p • M)(⦇N⦈) ≺ P'
      by(rule brinputSwapFrameSubject)
    then show ?thesis
      using Pair by simp
  qed
qed

lemma inputSwapSubject:
  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 $y \sharp P$
and $x \sharp \Psi$
and $y \sharp \Psi$

shows $\Psi \rhd P \longmapsto ([(x, y)] \cdot M)(\!|N|\!) \prec P'$
proof −
from ‹$\Psi \rhd P \longmapsto M(\!|N|\!) \prec P'$› ‹$x \sharp P$› ‹$y \sharp P$›
have $([(x, y)] \cdot \Psi) \rhd P \longmapsto([(x, y)] \cdot M)(\!|N|\!) \prec P'$
by(*rule inputSwapFrameSubject*)
with ‹$x \sharp \Psi$› ‹$y \sharp \Psi$› **show** *?thesis*
by *simp*
qed

lemma *brinputSwapSubject*:
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 $y \sharp P$
and $x \sharp \Psi$
and $y \sharp \Psi$

shows $\Psi \rhd P \longmapsto ¿([(x, y)] \cdot M)(\!|N|\!) \prec P'$
proof −
from ‹$\Psi \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$› ‹$x \sharp P$› ‹$y \sharp P$›
have $([(x, y)] \cdot \Psi) \rhd P \longmapsto ¿([(x, y)] \cdot M)(\!|N|\!) \prec P'$
by(*rule brinputSwapFrameSubject*)
with ‹$x \sharp \Psi$› ‹$y \sharp \Psi$› **show** *?thesis*
by *simp*
qed

lemma *inputPermSubject*:
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 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 −
from ‹$\Psi \rhd P \longmapsto M(\!|N|\!) \prec P'$› S ‹$Xs \sharp* P$› ‹$Ys \sharp* P$›
have $(p \cdot \Psi) \rhd P \longmapsto (p \cdot M)(\!|N|\!) \prec P'$
by(*rule inputPermFrameSubject*)
with ‹$Xs \sharp* \Psi$› ‹$Ys \sharp* \Psi$› S **show** *?thesis*
by *simp*
qed

lemma *brinputPermSubject*:
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 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 −
from ‹$\Psi \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$› S ‹$Xs \sharp* P$› ‹$Ys \sharp* P$›
have $(p \cdot \Psi) \rhd P \longmapsto ¿(p \cdot M)(\!|N|\!) \prec P'$
by(*rule brinputPermFrameSubject*)
with ‹$Xs \sharp* \Psi$› ‹$Ys \sharp* \Psi$› S **show** *?thesis*
by *simp*
qed

lemma *inputSwapFrame*:
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 Ψ ▷ P ⟼M⦇N⦈ ≺ P'
  and   x ♯ P
  and   y ♯ P
  and   x ♯ M
  and   y ♯ M

shows ([(x, y)] • Ψ) ▷ P ⟼ M⦇N⦈ ≺ P'
proof −
  from ‹Ψ ▷ P ⟼M⦇N⦈ ≺ P'› ‹x ♯ P› ‹y ♯ P›
  have ([(x, y)] • Ψ) ▷ P ⟼([(x, y)] • M)⦇N⦈ ≺ P'
    by(rule inputSwapFrameSubject)
  with ‹x ♯ M› ‹y ♯ M› show ?thesis
    by simp
qed

lemma brinputSwapFrame:
  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 Ψ ▷ P ⟼¿M⦇N⦈ ≺ P'
  and   x ♯ P
  and   y ♯ P
  and   x ♯ M
  and   y ♯ M

shows ([(x, y)] • Ψ) ▷ P ⟼ ¿M⦇N⦈ ≺ P'
proof −
  from ‹Ψ ▷ P ⟼¿M⦇N⦈ ≺ P'› ‹x ♯ P› ‹y ♯ P›
  have ([(x, y)] • Ψ) ▷ P ⟼¿([(x, y)] • M)⦇N⦈ ≺ P'
    by(rule brinputSwapFrameSubject)
  with ‹x ♯ M› ‹y ♯ M› show ?thesis
    by simp
qed

lemma inputPermFrame:
  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 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 −
from ‹$\Psi \rhd P \longmapsto M(\!|N|\!) \prec P'$› S ‹$Xs \sharp* P$› ‹$Ys \sharp* P$›
have $(p \cdot \Psi) \rhd P \longmapsto (p \cdot M)(\!|N|\!) \prec P'$
by(*rule inputPermFrameSubject*)
with ‹$Xs \sharp* M$› ‹$Ys \sharp* M$› S **show** *?thesis*
by *simp*
qed

lemma *brinputPermFrame*:
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 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 −
from ‹$\Psi \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$› S ‹$Xs \sharp* P$› ‹$Ys \sharp* P$›
have $(p \cdot \Psi) \rhd P \longmapsto ¿(p \cdot M)(\!|N|\!) \prec P'$
by(*rule brinputPermFrameSubject*)
with ‹$Xs \sharp* M$› ‹$Ys \sharp* M$› S **show** *?thesis*
by *simp*
qed

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 −
from ‹$\Psi \rhd P \longmapsto M(\!|N|\!) \prec P'$› ‹*set p* $\subseteq$ (*set xvec*) $\times$ (*set* ($p \cdot$ *xvec*))› ‹*xvec* $\sharp*$ P› ‹($p \cdot$ *xvec*) $\sharp*$ P›
have ($p \cdot \Psi$) $\rhd$ $P \longmapsto (p \cdot M)(\!|N|\!) \prec P'$ **by** − (*rule inputPermFrameSubject*, *auto*)
then have ($p \cdot p \cdot \Psi$) $\rhd$ ($p \cdot P$) $\longmapsto (p \cdot ((p \cdot M)(\!|N|\!) \prec P'))$ **by**(*rule eqvts*)
with ‹*distinctPerm p*› ‹*xvec* $\sharp*$ P› ‹($p \cdot$ *xvec*) $\sharp*$ P› ‹*set p* $\subseteq$ (*set xvec*) $\times$ (*set* (p $\cdot$ *xvec*))›
show *?thesis* **by**(*simp add*: *eqvts*)
qed

lemma *brinputAlpha*:
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 −
from ‹$\Psi \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$› ‹*set p* $\subseteq$ (*set xvec*) $\times$ (*set* ($p \cdot$ *xvec*))› ‹*xvec* $\sharp*$ P› ‹($p \cdot$ *xvec*) $\sharp*$ P›
have ($p \cdot \Psi$) $\rhd$ $P \longmapsto ¿(p \cdot M)(\!|N|\!) \prec P'$ **by** − (*rule brinputPermFrameSubject*, *auto*)
then have ($p \cdot p \cdot \Psi$) $\rhd$ ($p \cdot P$) $\longmapsto$ ($p \cdot (¿(p \cdot M)(\!|N|\!) \prec P')$) **by**(*rule eqvts*)
with ‹*distinctPerm p*› ‹*xvec* $\sharp*$ P› ‹($p \cdot$ *xvec*) $\sharp*$ P› ‹*set p* $\subseteq$ (*set xvec*) $\times$ (*set* (p

$\cdot$ *xvec*))›
show *?thesis* **by**(*simp add*: *eqvts*)
qed

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 $x \sharp \Psi_F$
using *assms*
by(*simp add*: *frameResChainFresh*) (*simp add*: *fresh-def name-list-supp*)

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 $([(x, y)] \cdot \Psi) \rhd P \longmapsto ([(x, y)] \cdot M)(\!|\nu * xvec|\!)\langle N\rangle \prec P'$
using *assms*
proof(*nominal-induct avoiding*: *x y rule*: *outputInduct'*)
case *cAlpha*
then show *?case* **by**(*simp add*: *create-residual.simps boundOutputChainAlpha''*)
next
case(*cOutput* Ψ M K N P x y)
from ‹$x \sharp M\langle N\rangle.P$› **have** $x \sharp M$ **by** *simp*
from ‹$y \sharp M\langle N\rangle.P$› **have** $y \sharp M$ **by** *simp*
from ‹$\Psi \vdash M \leftrightarrow K$› **have** $([(x, y)] \cdot \Psi) \vdash ([(x, y)] \cdot M) \leftrightarrow ([(x, y)] \cdot K)$
by(*rule chanEqClosed*)
with ‹$x \sharp M$› ‹$y \sharp M$› **have** $([(x, y)] \cdot \Psi) \vdash M \leftrightarrow ([(x, y)] \cdot K)$
by(*simp*)
then show *?case* **by**(*rule Output*)
next
case(*cCase* Ψ P M *xvec* N P' φ *Cs* x y)
from ‹$x \sharp$ *Cases Cs*› ‹$y \sharp$ *Cases Cs*› ‹$(\varphi, P) \in$ *set Cs*› **have** $x \sharp \varphi$ **and** $x \sharp P$ **and** $y \sharp \varphi$ **and** $y \sharp P$
by(*auto dest*: *memFresh*)

from ‹$x \sharp P$› ‹$y \sharp P$› **have** $([(x\ ,y)] \cdot \Psi) \rhd P \longmapsto([(x,\ y)] \cdot M)(\!|\nu * xvec|\!)\langle N\rangle \prec P'$ **by**(*rule cCase*)
moreover note ‹$(\varphi,\ P) \in$ *set Cs*›
moreover from ‹$\Psi \vdash \varphi$›
have $([(x,\ y)] \cdot \Psi) \vdash ([(x,\ y)] \cdot \varphi)$ **by**(*rule statClosed*)
with ‹$x \sharp \varphi$› ‹$y \sharp \varphi$›
have $([(x,\ y)] \cdot \Psi) \vdash \varphi$ **by** *simp*
ultimately show *?case* **using** ‹*guarded P*› **by**(*rule Case*)
next
case(*cPar1* $\Psi\ \Psi_Q\ P\ M\ xvec\ N\ P'\ A_Q\ Q\ x\ y$)
from ‹$x \sharp P \parallel Q$›
have $x \sharp P$ **and** $x \sharp Q$ **by** *simp+*
from ‹$y \sharp P \parallel Q$›
have $y \sharp P$ **and** $y \sharp Q$ **by** *simp+*
from ‹$x \sharp P$› ‹$y \sharp P$› ‹$\bigwedge x\ y.\ [\![x \sharp P;\ y \sharp P]\!] \Longrightarrow ([(x,\ y)] \cdot (\Psi \otimes \Psi_Q)) \rhd P \longmapsto([(x,\ y)] \cdot M)(\!|\nu * xvec|\!)\langle N\rangle \prec P'$›
have $([(x,\ y)] \cdot \Psi) \otimes ([(x,\ y)] \cdot \Psi_Q) \rhd P \longmapsto([(x,\ y)] \cdot M)(\!|\nu * xvec|\!)\langle N\rangle \prec P'$
by(*simp add*: *eqvts*)

moreover from ‹$extractFrame\ Q = \langle A_Q,\ \Psi_Q\rangle$›
have $([(x,\ y)] \cdot \langle A_Q,\ \Psi_Q\rangle) = ([(x,\ y)] \cdot (extractFrame\ Q))$
by *simp*
with ‹$A_Q \sharp* x$› ‹$x \sharp Q$› ‹$A_Q \sharp* y$› ‹$y \sharp Q$›
have $\langle A_Q,\ ([(x,\ y)] \cdot \Psi_Q)\rangle = extractFrame\ Q$
by(*simp add*: *eqvts*)
moreover from ‹$A_Q \sharp* \Psi$›
have $([(x,\ y)] \cdot A_Q) \sharp* ([(x,\ y)] \cdot \Psi)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$A_Q \sharp* x$› ‹$A_Q \sharp* y$›
have $A_Q \sharp* ([(x,\ y)] \cdot \Psi)$ **by** *simp*
moreover from ‹$A_Q \sharp* M$›
have $([(x,\ y)] \cdot A_Q) \sharp* ([(x,\ y)] \cdot M)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$A_Q \sharp* x$› ‹$A_Q \sharp* y$›
have $A_Q \sharp* ([(x,\ y)] \cdot M)$ **by** *simp*
ultimately show *?case* **using** ‹$A_Q \sharp* P$› ‹$A_Q \sharp* N$› ‹$xvec \sharp* Q$› ‹$A_Q \sharp* xvec$›
by(*force intro*: *Par1*)
next
case(*cPar2* $\Psi\ \Psi_P\ Q\ M\ xvec\ N\ Q'\ A_P\ P\ x\ y$)
from ‹$x \sharp P \parallel Q$›
have $x \sharp P$ **and** $x \sharp Q$ **by** *simp+*
from ‹$y \sharp P \parallel Q$›
have $y \sharp P$ **and** $y \sharp Q$ **by** *simp+*
from ‹$x \sharp Q$› ‹$y \sharp Q$› ‹$\bigwedge x\ y.\ [\![x \sharp Q;\ y \sharp Q]\!] \Longrightarrow ([(x,\ y)] \cdot (\Psi \otimes \Psi_P)) \rhd Q \longmapsto([(x,\ y)] \cdot M)(\!|\nu * xvec|\!)\langle N\rangle \prec Q'$›
have $([(x,\ y)] \cdot \Psi) \otimes ([(x,\ y)] \cdot \Psi_P) \rhd Q \longmapsto([(x,\ y)] \cdot M)(\!|\nu * xvec|\!)\langle N\rangle \prec Q'$
by(*simp add*: *eqvts*)

moreover from ‹$extractFrame\ P = \langle A_P,\ \Psi_P\rangle$›
have $([(x,\ y)] \cdot \langle A_P,\ \Psi_P\rangle) = ([(x,\ y)] \cdot (extractFrame\ P))$
by *simp*
with ‹$A_P \sharp* x$› ‹$x \sharp P$› ‹$A_P \sharp* y$› ‹$y \sharp P$›
have $\langle A_P,\ ([(x,\ y)] \cdot \Psi_P)\rangle = extractFrame\ P$
by(*simp add*: *eqvts*)
moreover from ‹$A_P \sharp* \Psi$›
have $([(x,\ y)] \cdot A_P) \sharp* ([(x,\ y)] \cdot \Psi)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$A_P \sharp* x$› ‹$A_P \sharp* y$›
have $A_P \sharp* ([(x,\ y)] \cdot \Psi)$ **by** *simp*
moreover from ‹$A_P \sharp* M$›
have $([(x,\ y)] \cdot A_P) \sharp* ([(x,\ y)] \cdot M)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])

with ‹A_P ♯* x› ‹A_P ♯* y› **have** A_P ♯* ([(x, y)] ∙ M) **by** *simp*
ultimately show *?case* **using** ‹A_P ♯* Q› ‹A_P ♯* N› ‹$xvec$ ♯* P› ‹A_P ♯* $xvec$›
by(*force intro*: *Par2*)
next
case(*cOpen* Ψ P M $xvec$ $yvec$ N P' z x y)
from ‹x ♯ (|νz|)P› ‹z ♯ x› **have** x ♯ P **by**(*simp add*: *abs-fresh*)
from ‹y ♯ (|νz|)P› ‹z ♯ y› **have** y ♯ P **by**(*simp add*: *abs-fresh*)
from ‹x ♯ P› ‹y ♯ P› ‹⋀x y. [[x ♯ P; y ♯ P]] ⟹ ([(x, y)] ∙ Ψ) ▷ P ⟼([(x, y)] ∙ M)(|ν*($xvec$@$yvec$)|)⟨N⟩ ≺ P'›
have ([(x, y)] ∙ Ψ) ▷ P ⟼([(x, y)] ∙ M)(|ν*($xvec$@$yvec$)|)⟨N⟩ ≺ P' **by** *simp*
moreover with ‹z ♯ Ψ› **have** ([(x, y)] ∙ z) ♯ [(x, y)] ∙ Ψ
by(*simp add*: *pt-fresh-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹z ♯ x› ‹z ♯ y› **have** z ♯ [(x, y)] ∙ Ψ **by** *simp*
moreover with ‹z ♯ M› **have** ([(x, y)] ∙ z) ♯ [(x, y)] ∙ M
by(*simp add*: *pt-fresh-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹z ♯ x› ‹z ♯ y› **have** z ♯ [(x, y)] ∙ M **by** *simp*
ultimately show *?case* **using** ‹z ∈ *supp* N› ‹z ♯ $xvec$› ‹z ♯ $yvec$›
by(*force intro*!: *Open*)
next
case(*cScope* Ψ P M $xvec$ N P' z x y)
from ‹x ♯ (|νz|)P› ‹z ♯ x› **have** x ♯ P **by**(*simp add*: *abs-fresh*)
from ‹y ♯ (|νz|)P› ‹z ♯ y› **have** y ♯ P **by**(*simp add*: *abs-fresh*)
from ‹x ♯ P› ‹y ♯ P› ‹⋀x y. [[x ♯ P; y ♯ P]] ⟹ ([(x, y)] ∙ Ψ) ▷ P ⟼([(x, y)] ∙ M)(|ν*$xvec$|)⟨N⟩ ≺ P'›
have ([(x, y)] ∙ Ψ) ▷ P ⟼([(x, y)] ∙ M)(|ν*$xvec$|)⟨N⟩ ≺ P' **by** *simp*
moreover with ‹z ♯ Ψ› **have** ([(x, y)] ∙ z) ♯ [(x, y)] ∙ Ψ
by(*simp add*: *pt-fresh-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹z ♯ x› ‹z ♯ y› **have** z ♯ [(x, y)] ∙ Ψ **by** *simp*
moreover with ‹z ♯ M› **have** ([(x, y)] ∙ z) ♯ [(x, y)] ∙ M
by(*simp add*: *pt-fresh-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹z ♯ x› ‹z ♯ y› **have** z ♯ [(x, y)] ∙ M **by** *simp*
ultimately show *?case* **using** ‹z ♯ N› ‹z ♯ $xvec$›
by(*force intro*!: *Scope*)
next
case(*cBang* Ψ P M B x y)
then show *?case* **by**(*force intro*: *Bang*)
qed

lemma *broutputSwapFrameSubject*:
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 Ψ ▷ P ⟼¡M(|ν*$xvec$|)⟨N⟩ ≺ P'
and $xvec$ ♯* M

and $x \sharp P$
and $y \sharp P$

shows $([(x, y)] \cdot \Psi) \rhd P \longmapsto ¡([(x, y)] \cdot M)(\!|\nu * xvec|\!)\langle N\rangle \prec P'$
using *assms*
proof(*nominal-induct avoiding*: *x y rule*: *brOutputInduct'*)
case *cAlpha*
then show *?case* **by**(*simp add*: *create-residual.simps boundOutputChainAlpha''*)
next
case(*cBrOutput* Ψ *M K N P x y*)
from ‹$x \sharp M\langle N\rangle.P$› **have** $x \sharp M$ **by** *simp*
from ‹$y \sharp M\langle N\rangle.P$› **have** $y \sharp M$ **by** *simp*
from ‹$\Psi \vdash M \preceq K$› **have** $([(x, y)] \cdot \Psi) \vdash ([(x, y)] \cdot M) \preceq ([(x, y)] \cdot K)$
by(*rule chanOutConClosed*)
with ‹$x \sharp M$› ‹$y \sharp M$› **have** $([(x, y)] \cdot \Psi) \vdash M \preceq ([(x, y)] \cdot K)$
by(*simp*)
then show *?case* **by**(*rule BrOutput*)
next
case(*cCase* Ψ *P M xvec N P'* φ *Cs x y*)
from ‹$x \sharp$ *Cases Cs*› ‹$y \sharp$ *Cases Cs*› ‹$(\varphi, P) \in$ *set Cs*› **have** $x \sharp \varphi$ **and** $x \sharp P$ **and** $y \sharp \varphi$ **and** $y \sharp P$
by(*auto dest*: *memFresh*)
from ‹$x \sharp P$› ‹$y \sharp P$› **have** $([(x ,y)] \cdot \Psi) \rhd P \longmapsto ¡([(x, y)] \cdot M)(\!|\nu * xvec|\!)\langle N\rangle \prec P'$ **by**(*rule cCase*)
moreover note ‹$(\varphi, P) \in$ *set Cs*›
moreover from ‹$\Psi \vdash \varphi$› **have** $([(x, y)] \cdot \Psi) \vdash ([(x, y)] \cdot \varphi)$ **by**(*rule statClosed*)
with ‹$x \sharp \varphi$› ‹$y \sharp \varphi$› **have** $([(x, y)] \cdot \Psi) \vdash \varphi$ **by** *simp*
ultimately show *?case* **using** ‹*guarded P*› **by**(*rule Case*)
next
case(*cPar1* Ψ Ψ_Q *P M xvec N P'* A_Q *Q x y*)
from ‹$x \sharp P \parallel Q$› **have** $x \sharp P$ **and** $x \sharp Q$ **by** *simp+*
from ‹$y \sharp P \parallel Q$› **have** $y \sharp P$ **and** $y \sharp Q$ **by** *simp+*
from ‹$x \sharp P$› ‹$y \sharp P$› ‹$\bigwedge x\ y.\ [\![x \sharp P;\ y \sharp P]\!] \Longrightarrow ([(x, y)] \cdot (\Psi \otimes \Psi_Q)) \rhd P \longmapsto ¡([(x, y)] \cdot M)(\!|\nu * xvec|\!)\langle N\rangle \prec P'$›
have $([(x, y)] \cdot \Psi) \otimes ([(x, y)] \cdot \Psi_Q) \rhd P \longmapsto ¡([(x, y)] \cdot M)(\!|\nu * xvec|\!)\langle N\rangle \prec P'$
by(*simp add*: *eqvts*)

moreover from ‹*extractFrame Q* = $\langle A_Q, \Psi_Q\rangle$› **have** $([(x, y)] \cdot \langle A_Q, \Psi_Q\rangle) = ([(x, y)] \cdot (\mathit{extractFrame}\ Q))$
by *simp*
with ‹$A_Q \sharp* x$› ‹$x \sharp Q$› ‹$A_Q \sharp* y$› ‹$y \sharp Q$› **have** $\langle A_Q, ([(x, y)] \cdot \Psi_Q)\rangle =$ *extractFrame Q*
by(*simp add*: *eqvts*)
moreover from ‹$A_Q \sharp* \Psi$› **have** $([(x, y)] \cdot A_Q) \sharp* ([(x, y)] \cdot \Psi)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$A_Q \sharp* x$› ‹$A_Q \sharp* y$› **have** $A_Q \sharp* ([(x, y)] \cdot \Psi)$ **by** *simp*
moreover from ‹$A_Q \sharp* M$› **have** $([(x, y)] \cdot A_Q) \sharp* ([(x, y)] \cdot M)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$A_Q \sharp* x$› ‹$A_Q \sharp* y$› **have** $A_Q \sharp* ([(x, y)] \cdot M)$ **by** *simp*

ultimately show *?case* **using** ‹A_Q ♯* P› ‹A_Q ♯* N› ‹$xvec$ ♯* Q› ‹A_Q ♯* $xvec$›
by(*force intro*: *Par1*)
next
case(*cPar2* Ψ Ψ_P Q M $xvec$ N Q' A_P P x y)
from ‹x ♯ $P \parallel Q$› **have** x ♯ P **and** x ♯ Q **by** *simp+*
from ‹y ♯ $P \parallel Q$› **have** y ♯ P **and** y ♯ Q **by** *simp+*
from ‹x ♯ Q› ‹y ♯ Q› ‹$\bigwedge x\ y.\ [\![x \sharp Q; y \sharp Q]\!] \Longrightarrow ([(x, y)] \cdot (\Psi \otimes \Psi_P)) \rhd Q \longmapsto ¡([(x, y)] \cdot M)(\!|\nu * xvec|\!)\langle N\rangle \prec Q'$›
have $([(x, y)] \cdot \Psi) \otimes ([(x, y)] \cdot \Psi_P) \rhd Q \longmapsto ¡([(x, y)] \cdot M)(\!|\nu * xvec|\!)\langle N\rangle \prec Q'$
by(*simp add*: *eqvts*)

moreover from ‹*extractFrame* $P = \langle A_P, \Psi_P\rangle$› **have** $([(x, y)] \cdot \langle A_P, \Psi_P\rangle) = ([(x, y)] \cdot (extractFrame\ P))$
by *simp*
with ‹A_P ♯* x› ‹x ♯ P› ‹A_P ♯* y› ‹y ♯ P› **have** $\langle A_P, ([(x, y)] \cdot \Psi_P)\rangle = extractFrame\ P$
by(*simp add*: *eqvts*)
moreover from ‹A_P ♯* Ψ› **have** $([(x, y)] \cdot A_P)$ ♯* $([(x, y)] \cdot \Psi)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹A_P ♯* x› ‹A_P ♯* y› **have** A_P ♯* $([(x, y)] \cdot \Psi)$ **by** *simp*
moreover from ‹A_P ♯* M› **have** $([(x, y)] \cdot A_P)$ ♯* $([(x, y)] \cdot M)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹A_P ♯* x› ‹A_P ♯* y› **have** A_P ♯* $([(x, y)] \cdot M)$ **by** *simp*
ultimately show *?case* **using** ‹A_P ♯* Q› ‹A_P ♯* N› ‹$xvec$ ♯* P› ‹A_P ♯* $xvec$›
by(*force intro*: *Par2*)
next
case(*cBrComm1* Ψ Ψ_Q P M N P' A_P Ψ_P Q $xvec$ Q' A_Q x y)
from ‹x ♯ $P \parallel Q$› **have** x ♯ P **and** x ♯ Q **by** *simp+*
from ‹y ♯ $P \parallel Q$› **have** y ♯ P **and** y ♯ Q **by** *simp+*

from ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$› ‹$x$ ♯ P› ‹y ♯ P›
have $([(x, y)] \cdot (\Psi \otimes \Psi_Q)) \rhd P \longmapsto ¿([(x, y)] \cdot M)(\!|N|\!) \prec P'$
by(*rule brinputSwapFrameSubject*)
then have *permIn*: $(([(x, y)] \cdot \Psi) \otimes ([(x, y)] \cdot \Psi_Q)) \rhd P \longmapsto ¿([(x, y)] \cdot M)(\!|N|\!) \prec P'$
by(*simp add*: *eqvts*)
moreover from ‹x ♯ Q› ‹y ♯ Q› ‹$\bigwedge x\ y.\ [\![x \sharp Q; y \sharp Q]\!] \Longrightarrow ([(x, y)] \cdot (\Psi \otimes \Psi_P)) \rhd Q \longmapsto ¡([(x, y)] \cdot M)(\!|\nu * xvec|\!)\langle N\rangle \prec Q'$›
have *permOut*: $([(x, y)] \cdot \Psi) \otimes ([(x, y)] \cdot \Psi_P) \rhd Q \longmapsto ¡([(x, y)] \cdot M)(\!|\nu * xvec|\!)\langle N\rangle \prec Q'$
by(*simp add*: *eqvts*)

moreover from ‹*extractFrame* $P = \langle A_P, \Psi_P\rangle$› **have** $([(x, y)] \cdot \langle A_P, \Psi_P\rangle) = ([(x, y)] \cdot (extractFrame\ P))$
by *simp*
with ‹A_P ♯* x› ‹x ♯ P› ‹A_P ♯* y› ‹y ♯ P› **have** $extractFrame\ P = \langle A_P, ([(x, y)] \cdot \Psi_P)\rangle$
by(*simp add*: *eqvts*)
moreover from ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› **have** $([(x, y)] \cdot \langle A_Q, \Psi_Q\rangle) =$

$([(x,\ y)] \cdot (extractFrame\ Q))$
by *simp*
with ‹A_Q ♯* x› ‹x ♯ Q› ‹A_Q ♯* y› ‹y ♯ Q› **have** *extractFrame* $Q = \langle A_Q, ([(x, y)] \cdot \Psi_Q)\rangle$
by(*simp add*: *eqvts*)
moreover from ‹A_P ♯* Ψ› **have** $([(x,\ y)] \cdot A_P)$ ♯* $([(x,\ y)] \cdot \Psi)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹A_P ♯* x› ‹A_P ♯* y› **have** A_P ♯* $([(x,\ y)] \cdot \Psi)$ **by** *simp*
moreover from ‹A_Q ♯* Ψ› **have** $([(x,\ y)] \cdot A_Q)$ ♯* $([(x,\ y)] \cdot \Psi)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹A_Q ♯* x› ‹A_Q ♯* y› **have** A_Q ♯* $([(x,\ y)] \cdot \Psi)$ **by** *simp*
moreover from ‹A_P ♯* Ψ_Q› **have** $([(x,\ y)] \cdot A_P)$ ♯* $([(x,\ y)] \cdot \Psi_Q)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹A_P ♯* x› ‹A_P ♯* y› **have** A_P ♯* $([(x,\ y)] \cdot \Psi_Q)$ **by** *simp*
moreover from ‹A_Q ♯* Ψ_P› **have** $([(x,\ y)] \cdot A_Q)$ ♯* $([(x,\ y)] \cdot \Psi_P)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹A_Q ♯* x› ‹A_Q ♯* y› **have** A_Q ♯* $([(x,\ y)] \cdot \Psi_P)$ **by** *simp*
moreover from ‹A_P ♯* M› **have** $([(x,\ y)] \cdot A_P)$ ♯* $([(x,\ y)] \cdot M)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹A_P ♯* x› ‹A_P ♯* y› **have** A_P ♯* $([(x,\ y)] \cdot M)$ **by** *simp*
moreover from ‹A_Q ♯* M› **have** $([(x,\ y)] \cdot A_Q)$ ♯* $([(x,\ y)] \cdot M)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹A_Q ♯* x› ‹A_Q ♯* y› **have** A_Q ♯* $([(x,\ y)] \cdot M)$ **by** *simp*
moreover from ‹*xvec* ♯* Ψ› **have** $([(x,\ y)] \cdot xvec)$ ♯* $([(x,\ y)] \cdot \Psi)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹*xvec* ♯* x› ‹*xvec* ♯* y› **have** *xvec* ♯* $([(x,\ y)] \cdot \Psi)$ **by** *simp*
moreover from ‹*xvec* ♯* Ψ_Q› **have** $([(x,\ y)] \cdot xvec)$ ♯* $([(x,\ y)] \cdot \Psi_Q)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹*xvec* ♯* x› ‹*xvec* ♯* y› **have** *xvec* ♯* $([(x,\ y)] \cdot \Psi_Q)$ **by** *simp*
moreover from ‹*xvec* ♯* Ψ_P› **have** $([(x,\ y)] \cdot xvec)$ ♯* $([(x,\ y)] \cdot \Psi_P)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹*xvec* ♯* x› ‹*xvec* ♯* y› **have** *xvec* ♯* $([(x,\ y)] \cdot \Psi_P)$ **by** *simp*
moreover from ‹*xvec* ♯* M› **have** $([(x,\ y)] \cdot xvec)$ ♯* $([(x,\ y)] \cdot M)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹*xvec* ♯* x› ‹*xvec* ♯* y› **have** *xvec* ♯* $([(x,\ y)] \cdot M)$ **by** *simp*

moreover note ‹*distinct* A_P› ‹*distinct* A_Q› ‹A_P ♯* P› ‹A_P ♯* N› ‹A_P ♯* P'› ‹A_P ♯* Q› ‹A_P ♯* Q'› ‹A_P ♯* A_Q›
‹A_P ♯* *xvec*› ‹A_Q ♯* P› ‹A_Q ♯* N› ‹A_Q ♯* P'› ‹A_Q ♯* Q› ‹A_Q ♯* Q'› ‹A_Q ♯* *xvec*›
‹*distinct xvec*› ‹*xvec* ♯* P› ‹*xvec* ♯* Q›
ultimately show *?case*
by(*simp add*: *semantics.cBrComm1*)
next
case(*cBrComm2* Ψ Ψ_Q P M *xvec* N P' A_P Ψ_P Q Q' A_Q x y)
from ‹x ♯ $P \parallel Q$› **have** x ♯ P **and** x ♯ Q **by** *simp*+
from ‹y ♯ $P \parallel Q$› **have** y ♯ P **and** y ♯ Q **by** *simp*+

from ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$› ‹$x$ ♯ Q› ‹y ♯ Q›

have $([(x,\ y)] \cdot (\Psi \otimes \Psi_P)) \rhd Q \longmapsto ¿([(x,\ y)] \cdot M)(\!|N|\!) \prec Q'$
by(*rule brinputSwapFrameSubject*)
then have *permIn*: $(([(x,\ y)] \cdot \Psi) \otimes ([(x,\ y)] \cdot \Psi_P)) \rhd Q \longmapsto ¿([(x,\ y)] \cdot M)(\!|N|\!) \prec Q'$
by(*simp add*: *eqvts*)
moreover from ‹$x \sharp P$› ‹$y \sharp P$› ‹$\bigwedge x\ y.\ [\![x \sharp P;\ y \sharp P]\!] \Longrightarrow ([(x,\ y)] \cdot (\Psi \otimes \Psi_Q)) \rhd P \longmapsto ¡([(x,\ y)] \cdot M)(\!|\nu * xvec|\!)\langle N\rangle \prec P'$›
have *permOut*: $([(x,\ y)] \cdot \Psi) \otimes ([(x,\ y)] \cdot \Psi_Q) \rhd P \longmapsto ¡([(x,\ y)] \cdot M)(\!|\nu * xvec|\!)\langle N\rangle \prec P'$
by(*simp add*: *eqvts*)

moreover from ‹*extractFrame* $P = \langle A_P,\ \Psi_P\rangle$› **have** $([(x,\ y)] \cdot \langle A_P,\ \Psi_P\rangle) = ([(x,\ y)] \cdot (\mathit{extractFrame}\ P))$
by *simp*
with ‹$A_P \sharp * x$› ‹$x \sharp P$› ‹$A_P \sharp * y$› ‹$y \sharp P$› **have** *extractFrame* $P = \langle A_P,\ ([(x,\ y)] \cdot \Psi_P)\rangle$
by(*simp add*: *eqvts*)
moreover from ‹*extractFrame* $Q = \langle A_Q,\ \Psi_Q\rangle$› **have** $([(x,\ y)] \cdot \langle A_Q,\ \Psi_Q\rangle) = ([(x,\ y)] \cdot (\mathit{extractFrame}\ Q))$
by *simp*
with ‹$A_Q \sharp * x$› ‹$x \sharp Q$› ‹$A_Q \sharp * y$› ‹$y \sharp Q$› **have** *extractFrame* $Q = \langle A_Q,\ ([(x,\ y)] \cdot \Psi_Q)\rangle$
by(*simp add*: *eqvts*)
moreover from ‹$A_P \sharp * \Psi$› **have** $([(x,\ y)] \cdot A_P) \sharp * ([(x,\ y)] \cdot \Psi)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$A_P \sharp * x$› ‹$A_P \sharp * y$› **have** $A_P \sharp * ([(x,\ y)] \cdot \Psi)$ **by** *simp*
moreover from ‹$A_Q \sharp * \Psi$› **have** $([(x,\ y)] \cdot A_Q) \sharp * ([(x,\ y)] \cdot \Psi)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$A_Q \sharp * x$› ‹$A_Q \sharp * y$› **have** $A_Q \sharp * ([(x,\ y)] \cdot \Psi)$ **by** *simp*
moreover from ‹$A_P \sharp * \Psi_Q$› **have** $([(x,\ y)] \cdot A_P) \sharp * ([(x,\ y)] \cdot \Psi_Q)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$A_P \sharp * x$› ‹$A_P \sharp * y$› **have** $A_P \sharp * ([(x,\ y)] \cdot \Psi_Q)$ **by** *simp*
moreover from ‹$A_Q \sharp * \Psi_P$› **have** $([(x,\ y)] \cdot A_Q) \sharp * ([(x,\ y)] \cdot \Psi_P)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$A_Q \sharp * x$› ‹$A_Q \sharp * y$› **have** $A_Q \sharp * ([(x,\ y)] \cdot \Psi_P)$ **by** *simp*
moreover from ‹$A_P \sharp * M$› **have** $([(x,\ y)] \cdot A_P) \sharp * ([(x,\ y)] \cdot M)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$A_P \sharp * x$› ‹$A_P \sharp * y$› **have** $A_P \sharp * ([(x,\ y)] \cdot M)$ **by** *simp*
moreover from ‹$A_Q \sharp * M$› **have** $([(x,\ y)] \cdot A_Q) \sharp * ([(x,\ y)] \cdot M)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$A_Q \sharp * x$› ‹$A_Q \sharp * y$› **have** $A_Q \sharp * ([(x,\ y)] \cdot M)$ **by** *simp*
moreover from ‹*xvec* $\sharp * \Psi$› **have** $(([(x,\ y)] \cdot xvec) \sharp * ([(x,\ y)] \cdot \Psi)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹*xvec* $\sharp * x$› ‹*xvec* $\sharp * y$› **have** *xvec* $\sharp * ([(x,\ y)] \cdot \Psi)$ **by** *simp*
moreover from ‹*xvec* $\sharp * \Psi_Q$› **have** $([(x,\ y)] \cdot xvec) \sharp * ([(x,\ y)] \cdot \Psi_Q)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹*xvec* $\sharp * x$› ‹*xvec* $\sharp * y$› **have** *xvec* $\sharp * ([(x,\ y)] \cdot \Psi_Q)$ **by** *simp*
moreover from ‹*xvec* $\sharp * \Psi_P$› **have** $([(x,\ y)] \cdot xvec) \sharp * ([(x,\ y)] \cdot \Psi_P)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])

with ‹*xvec* ♯* *x*› ‹*xvec* ♯* *y*› **have** *xvec* ♯* ([(*x*, *y*)] ∙ Ψ$_P$) **by** *simp*
moreover from ‹*xvec* ♯* *M*› **have** ([(*x*, *y*)] ∙ *xvec*) ♯* ([(*x*, *y*)] ∙ *M*)
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹*xvec* ♯* *x*› ‹*xvec* ♯* *y*› **have** *xvec* ♯* ([(*x*, *y*)] ∙ *M*) **by** *simp*

moreover note ‹*distinct* A_P› ‹*distinct* A_Q› ‹A_P ♯* *P*› ‹A_P ♯* *N*› ‹A_P ♯* *P′*› ‹A_P ♯* *Q*› ‹A_P ♯* *Q′*› ‹A_P ♯* A_Q›
‹A_P ♯* *xvec*› ‹A_Q ♯* *P*› ‹A_Q ♯* *N*› ‹A_Q ♯* *P′*› ‹A_Q ♯* *Q*› ‹A_Q ♯* *Q′*› ‹A_Q ♯* *xvec*›
‹*distinct xvec*› ‹*xvec* ♯* *P*› ‹*xvec* ♯* *Q*›
ultimately show *?case*
by(*simp add*: *semantics.cBrComm2*)
next
case(*cBrOpen* Ψ *P M xvec yvec N P′ z x y*)
from ‹*x* ♯ (⦇ν*z*⦈)*P*› ‹*z* ♯ *x*› **have** *x* ♯ *P* **by**(*simp add*: *abs-fresh*)
from ‹*y* ♯ (⦇ν*z*⦈)*P*› ‹*z* ♯ *y*› **have** *y* ♯ *P* **by**(*simp add*: *abs-fresh*)
from ‹*x* ♯ *P*› ‹*y* ♯ *P*› ‹⋀*x y*. ⟦*x* ♯ *P*; *y* ♯ *P*⟧ ⟹ ([(*x*, *y*)] ∙ Ψ) ▷ *P* ⟼¡([(*x*, *y*)] ∙ *M*)⦇ν*(*xvec*@*yvec*)⦈⟨*N*⟩ ≺ *P′*›
have ([(*x*, *y*)] ∙ Ψ) ▷ *P* ⟼¡([(*x*, *y*)] ∙ *M*)⦇ν*(*xvec*@*yvec*)⦈⟨*N*⟩ ≺ *P′* **by** *simp*
moreover with ‹*z* ♯ Ψ› **have** ([(*x*, *y*)] ∙ *z*) ♯ [(*x*, *y*)] ∙ Ψ
by(*simp add*: *pt-fresh-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹*z* ♯ *x*› ‹*z* ♯ *y*› **have** *z* ♯ [(*x*, *y*)] ∙ Ψ **by** *simp*
moreover with ‹*z* ♯ *M*› **have** ([(*x*, *y*)] ∙ *z*) ♯ [(*x*, *y*)] ∙ *M*
by(*simp add*: *pt-fresh-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹*z* ♯ *x*› ‹*z* ♯ *y*› **have** *z* ♯ [(*x*, *y*)] ∙ *M* **by** *simp*
ultimately show *?case* **using** ‹*z* ∈ *supp N*› ‹*z* ♯ *xvec*› ‹*z* ♯ *yvec*›
by(*force intro*: *BrOpen*)
next
case(*cScope* Ψ *P M xvec N P′ z x y*)
from ‹*x* ♯ (⦇ν*z*⦈)*P*› ‹*z* ♯ *x*› **have** *x* ♯ *P* **by**(*simp add*: *abs-fresh*)
from ‹*y* ♯ (⦇ν*z*⦈)*P*› ‹*z* ♯ *y*› **have** *y* ♯ *P* **by**(*simp add*: *abs-fresh*)
from ‹*x* ♯ *P*› ‹*y* ♯ *P*› ‹⋀*x y*. ⟦*x* ♯ *P*; *y* ♯ *P*⟧ ⟹ ([(*x*, *y*)] ∙ Ψ) ▷ *P* ⟼¡([(*x*, *y*)] ∙ *M*)⦇ν*xvec⦈⟨*N*⟩ ≺ *P′*›
have ([(*x*, *y*)] ∙ Ψ) ▷ *P* ⟼¡([(*x*, *y*)] ∙ *M*)⦇ν*xvec⦈⟨*N*⟩ ≺ *P′* **by** *simp*
moreover with ‹*z* ♯ Ψ› **have** ([(*x*, *y*)] ∙ *z*) ♯ [(*x*, *y*)] ∙ Ψ
by(*simp add*: *pt-fresh-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹*z* ♯ *x*› ‹*z* ♯ *y*› **have** *z* ♯ [(*x*, *y*)] ∙ Ψ **by** *simp*
moreover with ‹*z* ♯ *M*› **have** ([(*x*, *y*)] ∙ *z*) ♯ [(*x*, *y*)] ∙ *M*
by(*simp add*: *pt-fresh-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹*z* ♯ *x*› ‹*z* ♯ *y*› **have** *z* ♯ [(*x*, *y*)] ∙ *M* **by** *simp*
ultimately show *?case* **using** ‹*z* ♯ *N*› ‹*z* ♯ *xvec*›
by(*force intro*: *Scope*)
next
case(*cBang* Ψ *P M B x y*)
then show *?case* **by**(*force intro*: *Bang*)
qed

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 Ψ ▷ P ⟼M(⦇ν*xvec⦈)⟨N⟩ ≺ P'
  and   S: set p ⊆ set yvec × set zvec
  and   yvec ♯* P
  and   zvec ♯* P

shows (p · Ψ) ▷ P ⟼(p · M)(⦇ν*xvec⦈)⟨N⟩ ≺ P'
proof −
  {
    fix xvec N P' Xs YS
    assume Ψ ▷ P ⟼M(⦇ν*xvec⦈)⟨N⟩ ≺ P' and xvec ♯* M and xvec ♯* yvec and
xvec ♯* zvec
    have (p · Ψ) ▷ P ⟼(p · M)(⦇ν*xvec⦈)⟨N⟩ ≺ P' using S
    proof(induct p)
      case Nil
      from ‹Ψ ▷ P ⟼M(⦇ν*xvec⦈)⟨N⟩ ≺ P'›
      show ?case by simp
    next
      case(Cons a p)
      from ‹set(a#p) ⊆ set yvec × set zvec› have set p ⊆ set yvec × set zvec by
auto
      then have Trans: (p · Ψ) ▷ P ⟼(p · M)(⦇ν*xvec⦈)⟨N⟩ ≺ P' by(rule Cons)
      show ?case
      proof(cases a)
        case (Pair x y)
        note Trans
        moreover from ‹xvec ♯* yvec› ‹xvec ♯* zvec› ‹set p ⊆ set yvec × set zvec›
‹xvec ♯* M› have xvec ♯* (p · M)
          by(simp add: freshChainSimps)
        moreover have x ∈ set yvec and y ∈ set zvec
          using ‹set (a # p) ⊆ set yvec × set zvec› Pair by auto
        with ‹yvec ♯* P› ‹zvec ♯* P› have x ♯ P and y ♯ P
          by(auto simp add: fresh-star-def)
        ultimately have ([(x, y)] · p · Ψ) ▷ P ⟼([(x, y)] · p · M)(⦇ν*xvec⦈)⟨N⟩
≺ P'
          by(rule outputSwapFrameSubject)
        then show ?thesis
          using Pair by simp
      qed
    qed
  }
```

note *Goal* = *this*
obtain *q*::*name prm* **where** $(q \cdot xvec) \sharp* yvec$ **and** $(q \cdot xvec) \sharp* zvec$ **and** $(q \cdot xvec) \sharp* xvec$
and $(q \cdot xvec) \sharp* N$ **and** $(q \cdot xvec) \sharp* P'$ **and** $(q \cdot xvec) \sharp* M$
and *Sq*: $(set\ q) \subseteq (set\ xvec) \times (set(q \cdot xvec))$
by(*rule name-list-avoiding*[**where** *xvec*=*xvec* **and** *c*=(P, *xvec*, *yvec*, *zvec*, N, M, P')]) *auto*
with $\langle \Psi \rhd P \longmapsto M(\!|\nu* xvec|\!)\langle N\rangle \prec P'\rangle$ **have** $\Psi \rhd P \longmapsto M(\!|\nu*(q \cdot xvec)|\!)\langle(q \cdot N)\rangle \prec (q \cdot P')$
by(*simp add*: *boundOutputChainAlpha''* *residualInject*)
then have $(p \cdot \Psi) \rhd P \longmapsto (p \cdot M)(\!|\nu*(q \cdot xvec)|\!)\langle(q \cdot N)\rangle \prec (q \cdot P')$
using $\langle(q \cdot xvec) \sharp* M\rangle$ $\langle(q \cdot xvec) \sharp* yvec\rangle$ $\langle(q \cdot xvec) \sharp* zvec\rangle$
by(*rule Goal*)
with $\langle(q \cdot xvec) \sharp* N\rangle$ $\langle(q \cdot xvec) \sharp* P'\rangle$ *Sq* **show** *?thesis*
by(*simp add*: *boundOutputChainAlpha''* *residualInject*)
qed

lemma *broutputPermFrameSubject*:
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 −
{
fix *xvec* N P' *Xs* *YS*
assume $\Psi \rhd P \longmapsto ¡ M(\!|\nu* xvec|\!)\langle N\rangle \prec P'$ **and** $xvec \sharp* M$ **and** $xvec \sharp* yvec$ **and** $xvec \sharp* zvec$
have $(p \cdot \Psi) \rhd P \longmapsto ¡ (p \cdot M)(\!|\nu* xvec|\!)\langle N\rangle \prec P'$ **using** *S*
proof(*induct p*)
case *Nil*
from $\langle \Psi \rhd P \longmapsto ¡ M(\!|\nu* xvec|\!)\langle N\rangle \prec P'\rangle$
show *?case* **by** *simp*
next
case(*Cons a p*)
from $\langle set(a\#p) \subseteq set\ yvec \times set\ zvec\rangle$ **have** $set\ p \subseteq set\ yvec \times set\ zvec$ **by** *auto*
then have *Trans*: $(p \cdot \Psi) \rhd P \longmapsto ¡ (p \cdot M)(\!|\nu* xvec|\!)\langle N\rangle \prec P'$ **by**(*rule Cons*)

```
      show ?case
      proof(cases a)
        case (Pair x y)
        note Trans
        moreover from ‹xvec ♯* yvec› ‹xvec ♯* zvec› ‹set p ⊆ set yvec × set zvec›
‹xvec ♯* M› have xvec ♯* (p • M)
          by(simp add: freshChainSimps)
        moreover have x ∈ set yvec and y ∈ set zvec
          using ‹set (a # p) ⊆ set yvec × set zvec› Pair by auto
        with ‹yvec ♯* P› ‹zvec ♯* P› have x ♯ P and y ♯ P
          by(auto simp add: fresh-star-def)
        ultimately have ([(x, y)] • p • Ψ) ▷ P ⟼¡([(x, y)] • p • M)⦇ν*xvec⦈⟨N⟩
≺ P'
          by(rule broutputSwapFrameSubject)
        then show ?thesis
          using Pair by simp
      qed
    qed
  }
  note Goal = this
  obtain q::name prm where (q • xvec) ♯* yvec and (q • xvec) ♯* zvec and (q •
xvec) ♯* xvec
    and (q • xvec) ♯* N and (q • xvec) ♯* P' and (q • xvec) ♯* M
    and Sq: (set q) ⊆ (set xvec) × (set(q • xvec))
     by(rule name-list-avoiding[where xvec=xvec and c=(P, xvec, yvec, zvec, N,
M, P')]) auto
  with ‹Ψ ▷ P ⟼¡M⦇ν*xvec⦈⟨N⟩ ≺ P'› have Ψ ▷ P ⟼¡M⦇ν*(q • xvec)⦈⟨(q •
N)⟩ ≺ (q • P')
    by(simp add: boundOutputChainAlpha'' residualInject)
  then have (p • Ψ) ▷ P ⟼¡(p • M)⦇ν*(q • xvec)⦈⟨(q • N)⟩ ≺ (q • P')
    using ‹(q • xvec) ♯* M› ‹(q • xvec) ♯* yvec› ‹(q • xvec) ♯* zvec›
    by(rule Goal)
  with ‹(q • xvec) ♯* N› ‹(q • xvec) ♯* P'› Sq show ?thesis
    by(simp add: boundOutputChainAlpha'' residualInject)
qed

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 Ψ ▷ P ⟼M⦇ν*xvec⦈⟨N⟩ ≺ P'
  and   xvec ♯* M
  and   x ♯ P
  and   y ♯ P
  and   x ♯ Ψ
```

and $y \sharp \Psi$

shows $\Psi \rhd P \longmapsto([(x,\ y)] \cdot M)(\!|\nu{*}xvec|\!)\langle N\rangle \prec P'$
proof −
from ‹$\Psi \rhd P \longmapsto M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P'$› ‹$xvec \sharp{*}\ M$› ‹$x \sharp P$› ‹$y \sharp P$›
have $([(x,\ y)] \cdot \Psi) \rhd P \longmapsto([(x,\ y)] \cdot M)(\!|\nu{*}xvec|\!)\langle N\rangle \prec P'$
by(*rule outputSwapFrameSubject*)
with ‹$x \sharp \Psi$› ‹$y \sharp \Psi$› **show** *?thesis*
by *simp*
qed

lemma *broutputSwapSubject*:
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 −
from ‹$\Psi \rhd P \longmapsto¡M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P'$› ‹$xvec \sharp{*}\ M$› ‹$x \sharp P$› ‹$y \sharp P$›
have $([(x,\ y)] \cdot \Psi) \rhd P \longmapsto¡([(x,\ y)] \cdot M)(\!|\nu{*}xvec|\!)\langle N\rangle \prec P'$
by(*rule broutputSwapFrameSubject*)
with ‹$x \sharp \Psi$› ‹$y \sharp \Psi$› **show** *?thesis*
by *simp*
qed

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 −
from *assms* **have** $(p \cdot \Psi) \rhd P \longmapsto (p \cdot M)(\!|\nu * xvec|\!)\langle N\rangle \prec P'$
by(*metis outputPermFrameSubject*)
with S ‹*yvec* $\sharp*$ Ψ› ‹*zvec* $\sharp*$ Ψ› **show** *?thesis*
by *simp*
qed

lemma *broutputPermSubject*:
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 −
from *assms* **have** $(p \cdot \Psi) \rhd P \longmapsto ¡(p \cdot M)(\!|\nu * xvec|\!)\langle N\rangle \prec P'$
by(*metis broutputPermFrameSubject*)
with S ‹*yvec* $\sharp*$ Ψ› ‹*zvec* $\sharp*$ Ψ› **show** *?thesis*
by *simp*
qed

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 −
  from ‹$\Psi \rhd P \longmapsto M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P'$› ‹$xvec \sharp* M$› ‹$x \sharp P$› ‹$y \sharp P$›
  have $([(x,\ y)] \cdot \Psi) \rhd P \longmapsto([(x,\ y)] \cdot M)(\!|\nu{*}xvec|\!)\langle N\rangle \prec P'$
    by(*rule outputSwapFrameSubject*)
  with ‹$x \sharp M$› ‹$y \sharp M$› **show** *?thesis*
    by *simp*
qed

lemma *broutputSwapFrame*:
  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 $\Psi \rhd P \longmapsto\!\mathbin{¡} M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P'$
  and $xvec \sharp* M$
  and $x \sharp P$
  and $y \sharp P$
  and $x \sharp M$
  and $y \sharp M$

shows $([(x,\ y)] \cdot \Psi) \rhd P \longmapsto\!\mathbin{¡} M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P'$
proof −
  from ‹$\Psi \rhd P \longmapsto\!\mathbin{¡} M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P'$› ‹$xvec \sharp* M$› ‹$x \sharp P$› ‹$y \sharp P$›
  have $([(x,\ y)] \cdot \Psi) \rhd P \longmapsto\!\mathbin{¡}([(x,\ y)] \cdot M)(\!|\nu{*}xvec|\!)\langle N\rangle \prec P'$
    by(*rule broutputSwapFrameSubject*)
  with ‹$x \sharp M$› ‹$y \sharp M$› **show** *?thesis*
    by *simp*
qed

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 −
from *assms* **have** $(p \cdot \Psi) \rhd P \longmapsto (p \cdot M)(\!|\nu * xvec|\!)\langle N\rangle \prec P'$
by(*metis outputPermFrameSubject*)
with S ‹*yvec* $\sharp*$ M› ‹*zvec* $\sharp*$ M› **show** *?thesis*
by *simp*
qed

lemma *broutputPermFrame*:
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 −
from *assms* **have** $(p \cdot \Psi) \rhd P \longmapsto ¡(p \cdot M)(\!|\nu * xvec|\!)\langle N\rangle \prec P'$
by(*metis broutputPermFrameSubject*)
with S ‹*yvec* $\sharp*$ M› ‹*zvec* $\sharp*$ M› **show** *?thesis*
by *simp*
qed

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 −
{
fix 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*

assume $\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|N|\!) \prec P'$
and *extractFrame* $P = \langle A_P, \Psi_P\rangle$
and *distinct* A_P
and $\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|\nu * xvec|\!)\langle N\rangle \prec Q'$
and *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$
and *distinct* A_Q
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 $xvec \sharp* P$

have $\Psi \rhd P \parallel Q \longmapsto\tau \prec (\!|\nu * xvec|\!)(P' \parallel Q')$
proof −

obtain r::*name prm* **where** $(r \cdot xvec) \sharp* \Psi$ **and** $(r \cdot xvec) \sharp* P$ **and** $(r \cdot xvec) \sharp* Q$ **and** $(r \cdot xvec) \sharp* M$
and $(r \cdot xvec) \sharp* K$ **and** $(r \cdot xvec) \sharp* N$ **and** $(r \cdot xvec) \sharp* A_P$ **and** $(r \cdot xvec) \sharp* A_Q$
and $(r \cdot xvec) \sharp* P'$ **and** $(r \cdot xvec) \sharp* Q'$ **and** $(r \cdot xvec) \sharp* \Psi_P$ **and** $(r \cdot xvec) \sharp* \Psi_Q$
and *Sr*: $(set\ r) \subseteq (set\ xvec) \times (set(r \cdot xvec))$ **and** *distinctPerm r*
by(*rule name-list-avoiding*[**where** *xvec*=*xvec* **and** *c*=(Ψ, P, Q, M, K, N, A_P, A_Q, Ψ_P, Ψ_Q, P', Q')])
(*auto simp add*: *eqvts fresh-star-prod*)
obtain q::*name prm* **where** $(q \cdot A_Q) \sharp* \Psi$ **and** $(q \cdot A_Q) \sharp* P$ **and** $(q \cdot A_Q) \sharp* Q$ **and** $(q \cdot A_Q) \sharp* K$
and $(q \cdot A_Q) \sharp* (r \cdot N)$ **and** $(q \cdot A_Q) \sharp* (r \cdot xvec)$ **and** $(q \cdot A_Q) \sharp* (r \cdot Q')$
and $(q \cdot A_Q) \sharp* (r \cdot P')$ **and** $(q \cdot A_Q) \sharp* \Psi_P$ **and** $(q \cdot A_Q) \sharp* A_P$ **and** $(q \cdot A_Q) \sharp* \Psi_Q$
and *Sq*: $set\ q \subseteq set\ A_Q \times set(q \cdot A_Q)$
by(*rule name-list-avoiding*[**where** *xvec*=A_Q **and** *c*=(Ψ, P, Q, K, $r \cdot N$, $r \cdot xvec$, Ψ_Q, A_P, Ψ_P, $r \cdot Q'$, $r \cdot P'$)])
(*auto simp add*: *eqvts fresh-star-prod*)
obtain p::*name prm* **where** $(p \cdot A_P) \sharp* \Psi$ **and** $(p \cdot A_P) \sharp* P$ **and** $(p \cdot A_P) \sharp* Q$ **and** $(p \cdot A_P) \sharp* M$
and $(p \cdot A_P) \sharp* (r \cdot N)$ **and** $(p \cdot A_P) \sharp* (r \cdot xvec)$ **and** $(p \cdot A_P) \sharp* (r \cdot Q')$
and $(p \cdot A_P) \sharp* (r \cdot P')$ **and** $(p \cdot A_P) \sharp* \Psi_P$ **and** $(p \cdot A_P) \sharp* \Psi_Q$ **and** $(p \cdot A_P) \sharp* A_Q$
and $(p \cdot A_P) \sharp* (q \cdot A_Q)$ **and** *Sp*: $(set\ p) \subseteq (set\ A_P) \times (set(p \cdot A_P))$
by(*rule name-list-avoiding*[**where** *xvec*=A_P **and** *c*=(Ψ, P, Q, M, $r \cdot N$, $r \cdot xvec$, A_Q, $q \cdot A_Q$, Ψ_Q, Ψ_P, $r \cdot Q'$, $r \cdot P'$)])
(*auto simp add*: *eqvts fresh-star-prod*)
have *FrP*: *extractFrame* $P = \langle A_P, \Psi_P \rangle$ **by** *fact*
have *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$ **by** *fact*

from ‹$A_P \sharp* Q$› *FrQ* ‹$A_P \sharp* A_Q$› **have** $A_P \sharp* \Psi_Q$
by(*force dest*: *extractFrameFreshChain*)
from ‹$A_Q \sharp* P$› *FrP* ‹$A_P \sharp* A_Q$› **have** $A_Q \sharp* \Psi_P$
by(*force dest*: *extractFrameFreshChain*)
from ‹$(r \cdot xvec) \sharp* A_P$› ‹$(p \cdot A_P) \sharp* (r \cdot xvec)$› ‹$(r \cdot xvec) \sharp* A_P$› *Sp* **have** $(r \cdot xvec) \sharp* (p \cdot A_P)$
by(*simp add*: *freshChainSimps*)

from ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|N|\!) \prec P'$› *Sr* ‹*distinctPerm r*› ‹*xvec* $\sharp*$ P› ‹$(r \cdot xvec) \sharp* P$›
have $\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|(r \cdot N)|\!) \prec (r \cdot P')$
by(*rule inputAlpha*)
then have $(q \cdot (\Psi \otimes \Psi_Q)) \rhd P \longmapsto (q \cdot M)(\!|(r \cdot N)|\!) \prec (r \cdot P')$ **using** *Sq* ‹$A_Q \sharp* P$› ‹$(q \cdot A_Q) \sharp* P$›
by − (*rule inputPermFrameSubject*, (*assumption* | *simp*)+)

then have *PTrans*: $\Psi \otimes (q \cdot \Psi_Q) \rhd P \longmapsto (q \cdot M)(\!|(r \cdot N)|\!) \prec (r \cdot P')$ **using** *Sq* $\langle A_Q \sharp* \Psi\rangle$ $\langle (q \cdot A_Q) \sharp* \Psi\rangle$
by(*simp add*: *eqvts*)

moreover from $\langle$*extractFrame* $P = \langle A_P, \Psi_P\rangle\rangle$ *Sp* $\langle (p \cdot A_P) \sharp* \Psi_P\rangle$
have *FrP*: *extractFrame* $P = \langle (p \cdot A_P), (p \cdot \Psi_P)\rangle$
by(*simp add*: *frameChainAlpha*)
moreover from $\langle$*distinct* $A_P\rangle$ **have** *distinct*$(p \cdot A_P)$ **by** *simp*

moreover from $\langle \Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|\nu* xvec|\!)\langle N\rangle \prec Q'\rangle$ *Sr* $\langle (r \cdot xvec) \sharp* N\rangle$ $\langle (r \cdot xvec) \sharp* Q'\rangle$
have $\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|\nu*(r \cdot xvec)|\!)\langle (r \cdot N)\rangle \prec (r \cdot Q')$
by(*simp add*: *boundOutputChainAlpha''* *create-residual.simps*)
then have $(p \cdot (\Psi \otimes \Psi_P)) \rhd Q \longmapsto (p \cdot K)(\!|\nu*(r \cdot xvec)|\!)\langle (r \cdot N)\rangle \prec (r \cdot Q')$ **using** *Sp* $\langle A_P \sharp* Q\rangle$ $\langle (p \cdot A_P) \sharp* Q\rangle$ $\langle (r \cdot xvec) \sharp* K\rangle$ $\langle (r \cdot xvec) \sharp* A_P\rangle$ $\langle (r \cdot xvec) \sharp* (p \cdot A_P)\rangle$
by(*fastforce intro*: *outputPermFrameSubject*)
then have *QTrans*: $\Psi \otimes (p \cdot \Psi_P) \rhd Q \longmapsto (p \cdot K)(\!|\nu*(r \cdot xvec)|\!)\langle (r \cdot N)\rangle \prec (r \cdot Q')$ **using** *Sp* $\langle A_P \sharp* \Psi\rangle$ $\langle (p \cdot A_P) \sharp* \Psi\rangle$
by(*simp add*: *eqvts*)
moreover then have *distinct*$(r \cdot xvec)$ **by**(*force dest*: *boundOutputDistinct*)
moreover from $\langle$*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle\rangle$ *Sq* $\langle (q \cdot A_Q) \sharp* \Psi_Q\rangle$
have *FrQ*: *extractFrame* $Q = \langle (q \cdot A_Q), (q \cdot \Psi_Q)\rangle$
by(*simp add*: *frameChainAlpha*)
moreover from $\langle$*distinct* $A_Q\rangle$ **have** *distinct*$(q \cdot A_Q)$ **by** *simp*

moreover from $\langle \Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K\rangle$ **have** $(p \cdot q \cdot (\Psi \otimes \Psi_P \otimes \Psi_Q)) \vdash (p \cdot q \cdot M) \leftrightarrow (p \cdot q \cdot K)$
by(*metis chanEqClosed*)
with $\langle A_P \sharp* \Psi\rangle$ $\langle (p \cdot A_P) \sharp* \Psi\rangle$ $\langle A_Q \sharp* \Psi\rangle$ $\langle (q \cdot A_Q) \sharp* \Psi\rangle$ $\langle A_Q \sharp* \Psi_P\rangle$ $\langle (q \cdot A_Q) \sharp* \Psi_P\rangle$
$\langle A_P \sharp* \Psi_Q\rangle$ $\langle (p \cdot A_P) \sharp* \Psi_Q\rangle$ $\langle A_P \sharp* M\rangle$ $\langle (p \cdot A_P) \sharp* M\rangle$ $\langle (q \cdot A_Q) \sharp* A_P\rangle$ $\langle (p \cdot A_P) \sharp* (q \cdot A_Q)\rangle$
$\langle A_Q \sharp* K\rangle$ $\langle (q \cdot A_Q) \sharp* K\rangle$ $\langle A_P \sharp* A_Q\rangle$ $\langle (p \cdot A_P) \sharp* A_Q\rangle$ *Sp Sq*
have $\Psi \otimes (p \cdot \Psi_P) \otimes (q \cdot \Psi_Q) \vdash (q \cdot M) \leftrightarrow (p \cdot K)$ **by**(*simp add*: *eqvts freshChainSimps*)
moreover note $\langle (p \cdot A_P) \sharp* \Psi\rangle$
moreover from $\langle (p \cdot A_P) \sharp* A_Q\rangle$ $\langle (p \cdot A_P) \sharp* (q \cdot A_Q)\rangle$ $\langle (p \cdot A_P) \sharp* \Psi_Q\rangle$ *Sq* **have** $(p \cdot A_P) \sharp* (q \cdot \Psi_Q)$
by(*simp add*: *freshChainSimps*)
moreover note $\langle (p \cdot A_P) \sharp* P\rangle$
moreover from $\langle (p \cdot A_P) \sharp* A_Q\rangle$ $\langle (p \cdot A_P) \sharp* (q \cdot A_Q)\rangle$ $\langle (p \cdot A_P) \sharp* M\rangle$ *Sq* **have** $(p \cdot A_P) \sharp* (q \cdot M)$
by(*simp add*: *freshChainSimps*)
moreover note $\langle (p \cdot A_P) \sharp* (r \cdot N)\rangle$ $\langle (p \cdot A_P) \sharp* (r \cdot P')\rangle$ $\langle (p \cdot A_P) \sharp* Q\rangle$ $\langle (p \cdot A_P) \sharp* (r \cdot Q')\rangle$ $\langle (p \cdot A_P) \sharp* (q \cdot A_Q)\rangle$
$\langle (p \cdot A_P) \sharp* (r \cdot xvec)\rangle$ $\langle (q \cdot A_Q) \sharp* \Psi\rangle$
moreover from $\langle (q \cdot A_Q) \sharp* A_P\rangle$ $\langle (p \cdot A_P) \sharp* A_Q\rangle$ $\langle (q \cdot A_Q) \sharp* \Psi_P\rangle$ $\langle (p \cdot A_P) \sharp* (q \cdot A_Q)\rangle$ *Sp Sq* **have** $(q \cdot A_Q) \sharp* (p \cdot \Psi_P)$

by(*simp add*: *freshChainSimps*)
moreover note ‹$(q \cdot A_Q) \sharp* P$› ‹$(q \cdot A_Q) \sharp* (r \cdot N)$›‹$(q \cdot A_Q) \sharp* (r \cdot P')$› ‹$(q \cdot A_Q) \sharp* Q$›
moreover from ‹$(q \cdot A_Q) \sharp* A_P$› ‹$(p \cdot A_P) \sharp* A_Q$› ‹$(q \cdot A_Q) \sharp* K$› ‹$(p \cdot A_P) \sharp* (q \cdot A_Q)$› *Sp Sq* **have** $(q \cdot A_Q) \sharp* (p \cdot K)$
by(*simp add*: *freshChainSimps*)
moreover note ‹$(q \cdot A_Q) \sharp* (r \cdot Q')$› ‹$(q \cdot A_Q) \sharp* (r \cdot xvec)$› ‹$(r \cdot xvec) \sharp* \Psi$›
moreover from ‹$(r \cdot xvec) \sharp* A_P$› ‹$(p \cdot A_P) \sharp* (r \cdot xvec)$› ‹$(r \cdot xvec) \sharp* \Psi_P$› *Sp* **have** $(r \cdot xvec) \sharp* (p \cdot \Psi_P)$
by(*simp add*: *freshChainSimps*)
moreover from ‹$(r \cdot xvec) \sharp* A_Q$› ‹$(q \cdot A_Q) \sharp* (r \cdot xvec)$› ‹$(r \cdot xvec) \sharp* \Psi_Q$› *Sq* **have** $(r \cdot xvec) \sharp* (q \cdot \Psi_Q)$
by(*simp add*: *freshChainSimps*)
moreover note ‹$(r \cdot xvec) \sharp* P$›
moreover from ‹$(r \cdot xvec) \sharp* A_Q$› ‹$(q \cdot A_Q) \sharp* (r \cdot xvec)$› ‹$(r \cdot xvec) \sharp* M$› *Sq* **have** $(r \cdot xvec) \sharp* (q \cdot M)$
by(*simp add*: *freshChainSimps*)
moreover note ‹$(r \cdot xvec) \sharp* Q$›
moreover from ‹$(r \cdot xvec) \sharp* A_P$› ‹$(p \cdot A_P) \sharp* (r \cdot xvec)$› ‹$(r \cdot xvec) \sharp* K$› *Sp* **have** $(r \cdot xvec) \sharp* (p \cdot K)$
by(*simp add*: *freshChainSimps*)
ultimately have $\Psi \rhd P \parallel Q \longmapsto \tau \prec (\!|\nu*(r \cdot xvec)|\!)((r \cdot P') \parallel (r \cdot Q'))$
by − (*rule cComm1*)
with ‹$(r \cdot xvec) \sharp* P'$› ‹$(r \cdot xvec) \sharp* Q'$› *Sr*
show *?thesis*
by(*subst resChainAlpha*) *auto*
qed
}
note *Goal* = *this*
note ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|N|\!) \prec P'$› ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|\nu*xvec|\!)\langle N\rangle \prec Q'$› ‹$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K$›
moreover from ‹*extractFrame* $P = \langle A_P, \Psi_P\rangle$› ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_P \sharp* Q$› ‹$A_P \sharp* M$› ‹$A_P \sharp* A_Q$›
obtain A_P' **where** *extractFrame* $P = \langle A_P', \Psi_P\rangle$ **and** *distinct* A_P' **and** $A_P' \sharp* \Psi$ **and** $A_P' \sharp* P$ **and** $A_P' \sharp* Q$ **and** $A_P' \sharp* M$ **and** $A_P' \sharp* A_Q$
by − (*rule distinctFrame*[**where** C=(Ψ, P, Q, M, A_Q)], *auto*)
moreover from ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› ‹$A_Q \sharp* \Psi$› ‹$A_Q \sharp* P$› ‹$A_Q \sharp* Q$› ‹$A_Q \sharp* K$› ‹$A_P' \sharp* A_Q$›
obtain A_Q' **where** *extractFrame* $Q = \langle A_Q', \Psi_Q\rangle$ **and** *distinct* A_Q' **and** $A_Q' \sharp* \Psi$ **and** $A_Q' \sharp* P$ **and** $A_Q' \sharp* Q$ **and** $A_Q' \sharp* K$ **and** $A_P' \sharp* A_Q'$
by − (*rule distinctFrame*[**where** C=(Ψ, P, Q, K, A_P')], *auto*)
ultimately show *?thesis* **using** ‹*xvec* $\sharp* P$›
by(*metis Goal*)
qed

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 −
{
fix 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*

assume $\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$
and *extractFrame* $P = \langle A_P, \Psi_P\rangle$
and *distinct* A_P
and $\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|N|\!) \prec Q'$

and $extractFrame\ Q = \langle A_Q, \Psi_Q\rangle$
and $distinct\ A_Q$
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 $xvec \sharp* Q$

have $\Psi \rhd P \parallel Q \longmapsto\tau \prec (\!|\nu* xvec|\!)(P' \parallel Q')$
proof −

obtain r::$name\ prm$ **where** $(r \cdot xvec) \sharp* \Psi$ **and** $(r \cdot xvec) \sharp* P$ **and** $(r \cdot xvec) \sharp* Q$ **and** $(r \cdot xvec) \sharp* M$
and $(r \cdot xvec) \sharp* K$ **and** $(r \cdot xvec) \sharp* N$ **and** $(r \cdot xvec) \sharp* A_P$ **and** $(r \cdot xvec) \sharp* A_Q$
and $(r \cdot xvec) \sharp* P'$ **and** $(r \cdot xvec) \sharp* Q'$ **and** $(r \cdot xvec) \sharp* \Psi_P$ **and** $(r \cdot xvec) \sharp* \Psi_Q$
and Sr: $(set\ r) \subseteq (set\ xvec) \times (set(r \cdot xvec))$ **and** $distinctPerm\ r$
by(*rule name-list-avoiding*[**where** $xvec$=$xvec$ **and** c=$(\Psi, P, Q, M, K, N, A_P, A_Q, \Psi_P, \Psi_Q, P', Q')$])
(*auto simp add*: *eqvts fresh-star-prod*)
obtain q::$name\ prm$ **where** $(q \cdot A_Q) \sharp* \Psi$ **and** $(q \cdot A_Q) \sharp* P$ **and** $(q \cdot A_Q) \sharp* Q$ **and** $(q \cdot A_Q) \sharp* K$
and $(q \cdot A_Q) \sharp* (r \cdot N)$ **and** $(q \cdot A_Q) \sharp* (r \cdot xvec)$ **and** $(q \cdot A_Q) \sharp* (r \cdot Q')$
and $(q \cdot A_Q) \sharp* (r \cdot P')$ **and** $(q \cdot A_Q) \sharp* \Psi_P$ **and** $(q \cdot A_Q) \sharp* A_P$ **and** $(q \cdot A_Q) \sharp* \Psi_Q$
and Sq: $set\ q \subseteq set\ A_Q \times set(q \cdot A_Q)$
by(*rule name-list-avoiding*[**where** $xvec$=A_Q **and** c=$(\Psi, P, Q, K, r \cdot N, r \cdot xvec, \Psi_Q, A_P, \Psi_P, r \cdot Q', r \cdot P')$])
(*auto simp add*: *eqvts fresh-star-prod*)
obtain p::$name\ prm$ **where** $(p \cdot A_P) \sharp* \Psi$ **and** $(p \cdot A_P) \sharp* P$ **and** $(p \cdot A_P) \sharp* Q$ **and** $(p \cdot A_P) \sharp* M$
and $(p \cdot A_P) \sharp* (r \cdot N)$ **and** $(p \cdot A_P) \sharp* (r \cdot xvec)$ **and** $(p \cdot A_P) \sharp* (r \cdot Q')$
and $(p \cdot A_P) \sharp* (r \cdot P')$ **and** $(p \cdot A_P) \sharp* \Psi_P$ **and** $(p \cdot A_P) \sharp* \Psi_Q$ **and** $(p \cdot A_P) \sharp* A_Q$
and $(p \cdot A_P) \sharp* (q \cdot A_Q)$ **and** Sp: $(set\ p) \subseteq (set\ A_P) \times (set(p \cdot A_P))$
by(*rule name-list-avoiding*[**where** $xvec$=A_P **and** c=$(\Psi, P, Q, M, r \cdot N, r \cdot xvec, A_Q, q \cdot A_Q, \Psi_Q, \Psi_P, r \cdot Q', r \cdot P')$])
(*auto simp add*: *eqvts fresh-star-prod*)

have FrP: $extractFrame\ P = \langle A_P, \Psi_P\rangle$ **by** *fact*
have FrQ: $extractFrame\ Q = \langle A_Q, \Psi_Q\rangle$ **by** *fact*

from ‹A_P ♯* Q› FrQ ‹A_P ♯* A_Q› **have** A_P ♯* Ψ_Q
by(*auto dest*: *extractFrameFreshChain*)
from ‹A_Q ♯* P› FrP ‹A_P ♯* A_Q› **have** A_Q ♯* Ψ_P
by(*auto dest*: *extractFrameFreshChain*)

from ‹$(r \cdot xvec)$ ♯* A_Q› ‹$(q \cdot A_Q)$ ♯* $(r \cdot xvec)$› ‹$(r \cdot xvec)$ ♯* A_Q› Sq **have** $(r \cdot xvec)$ ♯* $(q \cdot A_Q)$
by(*simp add*: *freshChainSimps*)

from ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$› Sr ‹$(r \cdot xvec)$ ♯* N› ‹$(r \cdot xvec)$ ♯* P'›
have $\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|\nu * (r \cdot xvec)|\!)\langle (r \cdot N)\rangle \prec (r \cdot P')$
by(*simp add*: *boundOutputChainAlpha''* *create-residual.simps*)
then have $(q \cdot (\Psi \otimes \Psi_Q)) \rhd P \longmapsto (q \cdot M)(\!|\nu * (r \cdot xvec)|\!)\langle (r \cdot N)\rangle \prec (r \cdot P')$ **using** Sq ‹A_Q ♯* P› ‹$(q \cdot A_Q)$ ♯* P› ‹$(r \cdot xvec)$ ♯* M› ‹$(r \cdot xvec)$ ♯* A_Q› ‹$(r \cdot xvec)$ ♯* $(q \cdot A_Q)$›
by(*fastforce intro*: *outputPermFrameSubject*)
then have *PTrans*: $\Psi \otimes (q \cdot \Psi_Q) \rhd P \longmapsto (q \cdot M)(\!|\nu * (r \cdot xvec)|\!)\langle (r \cdot N)\rangle \prec (r \cdot P')$ **using** Sq ‹A_Q ♯* Ψ› ‹$(q \cdot A_Q)$ ♯* Ψ›
by(*simp add*: *eqvts*)
moreover then have *distinct*$(r \cdot xvec)$ **by**(*force dest*: *boundOutputDistinct*)

moreover from ‹*extractFrame* $P = \langle A_P, \Psi_P\rangle$› Sp ‹$(p \cdot A_P)$ ♯* Ψ_P›
have *FrP*: *extractFrame* $P = \langle (p \cdot A_P), (p \cdot \Psi_P)\rangle$
by(*simp add*: *frameChainAlpha*)
moreover from ‹*distinct* A_P› **have** *distinct*$(p \cdot A_P)$ **by** *simp*

moreover from ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|N|\!) \prec Q'$› Sr ‹*distinctPerm* r› ‹$xvec$ ♯* Q› ‹$(r \cdot xvec)$ ♯* Q›
have $\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|(r \cdot N)|\!) \prec (r \cdot Q')$
by(*rule inputAlpha*)
then have $(p \cdot (\Psi \otimes \Psi_P)) \rhd Q \longmapsto (p \cdot K)(\!|(r \cdot N)|\!) \prec (r \cdot Q')$ **using** Sp ‹A_P ♯* Q› ‹$(p \cdot A_P)$ ♯* Q›
by − (*rule inputPermFrameSubject*, (*assumption* | *simp*)+)
then have *QTrans*: $\Psi \otimes (p \cdot \Psi_P) \rhd Q \longmapsto (p \cdot K)(\!|(r \cdot N)|\!) \prec (r \cdot Q')$ **using** Sp ‹A_P ♯* Ψ› ‹$(p \cdot A_P)$ ♯* Ψ›
by(*simp add*: *eqvts*)

moreover from ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› Sq ‹$(q \cdot A_Q)$ ♯* Ψ_Q›
have *FrQ*: *extractFrame* $Q = \langle (q \cdot A_Q), (q \cdot \Psi_Q)\rangle$
by(*simp add*: *frameChainAlpha*)
moreover from ‹*distinct* A_Q› **have** *distinct*$(q \cdot A_Q)$ **by** *simp*

moreover from ‹$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K$› **have** $(p \cdot q \cdot (\Psi \otimes \Psi_P \otimes \Psi_Q)) \vdash (p \cdot q \cdot M) \leftrightarrow (p \cdot q \cdot K)$
by(*metis chanEqClosed*)
with ‹A_P ♯* Ψ› ‹$(p \cdot A_P)$ ♯* Ψ› ‹A_Q ♯* Ψ› ‹$(q \cdot A_Q)$ ♯* Ψ› ‹A_Q ♯* Ψ_P› ‹$(q \cdot A_Q)$ ♯* Ψ_P›
‹A_P ♯* Ψ_Q› ‹$(p \cdot A_P)$ ♯* Ψ_Q› ‹A_P ♯* M› ‹$(p \cdot A_P)$ ♯* M› ‹$(q \cdot A_Q)$ ♯* A_P›

$\langle(p \cdot A_P) \sharp* (q \cdot A_Q)\rangle$
$\langle A_Q \sharp* K\rangle$ $\langle(q \cdot A_Q) \sharp* K\rangle$ $\langle A_P \sharp* A_Q\rangle$ $\langle(p \cdot A_P) \sharp* A_Q\rangle$ *Sp Sq*
have $\Psi \otimes (p \cdot \Psi_P) \otimes (q \cdot \Psi_Q) \vdash (q \cdot M) \leftrightarrow (p \cdot K)$
by(*simp add*: *eqvts freshChainSimps*)
moreover note $\langle(p \cdot A_P) \sharp* \Psi\rangle$
moreover from $\langle(p \cdot A_P) \sharp* A_Q\rangle$ $\langle(p \cdot A_P) \sharp* (q \cdot A_Q)\rangle$ $\langle(p \cdot A_P) \sharp* \Psi_Q\rangle$ *Sq* **have** $(p \cdot A_P) \sharp* (q \cdot \Psi_Q)$
by(*simp add*: *freshChainSimps*)
moreover note $\langle(p \cdot A_P) \sharp* P\rangle$
moreover from $\langle(p \cdot A_P) \sharp* A_Q\rangle$ $\langle(p \cdot A_P) \sharp* (q \cdot A_Q)\rangle$ $\langle(p \cdot A_P) \sharp* M\rangle$ *Sq* **have** $(p \cdot A_P) \sharp* (q \cdot M)$
by(*simp add*: *freshChainSimps*)
moreover note $\langle(p \cdot A_P) \sharp* (r \cdot N)\rangle$ $\langle(p \cdot A_P) \sharp* (r \cdot P')\rangle$ $\langle(p \cdot A_P) \sharp* Q\rangle$ $\langle(p \cdot A_P) \sharp* (r \cdot Q')\rangle$ $\langle(p \cdot A_P) \sharp* (q \cdot A_Q)\rangle$
$\langle(p \cdot A_P) \sharp* (r \cdot xvec)\rangle$ $\langle(q \cdot A_Q) \sharp* \Psi\rangle$
moreover from $\langle(q \cdot A_Q) \sharp* A_P\rangle$ $\langle(p \cdot A_P) \sharp* A_Q\rangle$ $\langle(q \cdot A_Q) \sharp* \Psi_P\rangle$ $\langle(p \cdot A_P) \sharp* (q \cdot A_Q)\rangle$ *Sp Sq* **have** $(q \cdot A_Q) \sharp* (p \cdot \Psi_P)$
by(*simp add*: *freshChainSimps*)
moreover note $\langle(q \cdot A_Q) \sharp* P\rangle$ $\langle(q \cdot A_Q) \sharp* (r \cdot N)\rangle$$\langle(q \cdot A_Q) \sharp* (r \cdot P')\rangle$ $\langle(q \cdot A_Q) \sharp* Q\rangle$
moreover from $\langle(q \cdot A_Q) \sharp* A_P\rangle$ $\langle(p \cdot A_P) \sharp* A_Q\rangle$ $\langle(q \cdot A_Q) \sharp* K\rangle$ $\langle(p \cdot A_P) \sharp* (q \cdot A_Q)\rangle$ *Sp Sq* **have** $(q \cdot A_Q) \sharp* (p \cdot K)$
by(*simp add*: *freshChainSimps*)
moreover note $\langle(q \cdot A_Q) \sharp* (r \cdot Q')\rangle$ $\langle(q \cdot A_Q) \sharp* (r \cdot xvec)\rangle$ $\langle(r \cdot xvec) \sharp* \Psi\rangle$
moreover from $\langle(r \cdot xvec) \sharp* A_P\rangle$ $\langle(p \cdot A_P) \sharp* (r \cdot xvec)\rangle$ $\langle(r \cdot xvec) \sharp* \Psi_P\rangle$ *Sp* **have** $(r \cdot xvec) \sharp* (p \cdot \Psi_P)$
by(*simp add*: *freshChainSimps*)
moreover from $\langle(r \cdot xvec) \sharp* A_Q\rangle$ $\langle(q \cdot A_Q) \sharp* (r \cdot xvec)\rangle$ $\langle(r \cdot xvec) \sharp* \Psi_Q\rangle$ *Sq* **have** $(r \cdot xvec) \sharp* (q \cdot \Psi_Q)$
by(*simp add*: *freshChainSimps*)
moreover note $\langle(r \cdot xvec) \sharp* P\rangle$
moreover from $\langle(r \cdot xvec) \sharp* A_Q\rangle$ $\langle(q \cdot A_Q) \sharp* (r \cdot xvec)\rangle$ $\langle(r \cdot xvec) \sharp* M\rangle$ *Sq* **have** $(r \cdot xvec) \sharp* (q \cdot M)$
by(*simp add*: *freshChainSimps*)
moreover note $\langle(r \cdot xvec) \sharp* Q\rangle$
moreover from $\langle(r \cdot xvec) \sharp* A_P\rangle$ $\langle(p \cdot A_P) \sharp* (r \cdot xvec)\rangle$ $\langle(r \cdot xvec) \sharp* K\rangle$ *Sp* **have** $(r \cdot xvec) \sharp* (p \cdot K)$
by(*simp add*: *freshChainSimps*)
ultimately have $\Psi \rhd P \parallel Q \longmapsto \tau \prec (\!|\nu*(r \cdot xvec)|\!)((r \cdot P') \parallel (r \cdot Q'))$
by $-$ (*rule cComm2*)
with $\langle(r \cdot xvec) \sharp* P'\rangle$ $\langle(r \cdot xvec) \sharp* Q'\rangle$ *Sr*
show *?thesis*
by(*subst resChainAlpha*) *auto*
qed
}
note *Goal* = *this*
note $\langle\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|\nu*xvec|\!)\langle N\rangle \prec P'\rangle$ $\langle\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|N|\!) \prec Q'\rangle$ $\langle\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K\rangle$

moreover from ‹*extractFrame* $P = \langle A_P, \Psi_P \rangle$› ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_P \sharp* Q$› ‹$A_P \sharp* M$› ‹$A_P \sharp* A_Q$›
obtain $A_P{'}$ **where** *extractFrame* $P = \langle A_P{'}, \Psi_P \rangle$ **and** *distinct* $A_P{'}$ **and** $A_P{'} \sharp* \Psi$ **and** $A_P{'} \sharp* P$ **and** $A_P{'} \sharp* Q$ **and** $A_P{'} \sharp* M$ **and** $A_P{'} \sharp* A_Q$
by − (*rule distinctFrame*[**where** C=(Ψ, P, Q, M, A_Q)], *auto*)
moreover from ‹*extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$› ‹$A_Q \sharp* \Psi$› ‹$A_Q \sharp* P$› ‹$A_Q \sharp* Q$› ‹$A_Q \sharp* K$› ‹$A_P{'} \sharp* A_Q$›
obtain $A_Q{'}$ **where** *extractFrame* $Q = \langle A_Q{'}, \Psi_Q \rangle$ **and** *distinct* $A_Q{'}$ **and** $A_Q{'} \sharp* \Psi$ **and** $A_Q{'} \sharp* P$ **and** $A_Q{'} \sharp* Q$ **and** $A_Q{'} \sharp* K$ **and** $A_P{'} \sharp* A_Q{'}$
by − (*rule distinctFrame*[**where** C=(Ψ, P, Q, K, $A_P{'}$)], *auto*)
ultimately show *?thesis* **using** ‹*xvec* $\sharp* Q$›
by(*metis Goal*)
qed

lemma *BrMerge*:
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 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 $\Psi \otimes \Psi_P \rhd Q \longmapsto ¿ M(\!|N|\!) \prec Q'$
and *extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$
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 $A_Q \sharp* M$

shows $\Psi \rhd P \parallel Q \longmapsto ¿ M(\!|N|\!) \prec (P' \parallel Q')$
proof −
{
fix 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 Q' :: $('a, 'b, 'c)$ *psi*
and A_Q :: *name list*
and *svec* :: *name list*

assume $\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$
and *extractFrame* $P = \langle A_P, \Psi_P \rangle$
and *distinct* A_P
and $\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$
and *extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$
and *distinct* A_Q
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 $A_Q \sharp* M$

have $\Psi \rhd P \parallel Q \longmapsto ¿M(\!|N|\!) \prec (P' \parallel Q')$
proof −
obtain q::*name prm* **where** $(q \cdot (A_Q$::*name list*$)) \sharp* \Psi$ **and** $(q \cdot A_Q) \sharp* P$
and $(q \cdot A_Q) \sharp* Q$ **and** $(q \cdot A_Q) \sharp* M$
and $(q \cdot A_Q) \sharp* \Psi_P$ **and** $(q \cdot A_Q) \sharp* A_P$ **and** $(q \cdot A_Q) \sharp* \Psi_Q$
and $(q \cdot A_Q) \sharp* N$ **and** $(q \cdot A_Q) \sharp* P'$ **and** $(q \cdot A_Q) \sharp* Q'$
and *Sq*: *set* $q \subseteq$ *set* $A_Q \times$ *set*$(q \cdot A_Q)$
and *distinctPerm* q
by(*rule name-list-avoiding*[**where** c=(Ψ, P, M, N, P', Q', Q, Ψ_Q, A_P, Ψ_P)])
(*auto simp add*: *eqvts fresh-star-prod*)
obtain p::*name prm* **where** $(p \cdot A_P) \sharp* \Psi$ **and** $(p \cdot A_P) \sharp* P$
and $(p \cdot A_P) \sharp* Q$ **and** $(p \cdot A_P) \sharp* M$
and $(p \cdot A_P) \sharp* \Psi_P$ **and** $(p \cdot A_P) \sharp* \Psi_Q$ **and** $(p \cdot A_P) \sharp* A_Q$
and $(p \cdot A_P) \sharp* N$ **and** $(p \cdot A_P) \sharp* P'$ **and** $(p \cdot A_P) \sharp* Q'$
and *Sp*: *set* $p \subseteq$ *set* $A_P \times$ *set*$(p \cdot A_P)$
and $(p \cdot A_P) \sharp* (q \cdot A_Q)$
by(*rule name-list-avoiding*[**where** c=(Ψ, P, N, P', Q', Q, M, A_Q, $q \cdot A_Q$, Ψ_Q, Ψ_P)])
(*auto simp add*: *eqvts fresh-star-prod*)

have *FrP*: *extractFrame* $P = \langle A_P, \Psi_P \rangle$ **by** *fact*
have *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$ **by** *fact*

from ‹$A_P \sharp* Q$› *FrQ* ‹$A_P \sharp* A_Q$› **have** $A_P \sharp* \Psi_Q$

by(*force dest*: *extractFrameFreshChain*)
from ‹A_Q ♯* P› *FrP* ‹A_P ♯* A_Q› **have** A_Q ♯* Ψ_P
by(*force dest*: *extractFrameFreshChain*)

from *Sq* ‹A_Q ♯* M› ‹$(q \cdot A_Q)$ ♯* M›
have $(q \cdot M) = M$
by *simp*
from *Sp* ‹A_P ♯* M› ‹$(p \cdot A_P)$ ♯* M›
have $(p \cdot M) = M$
by *simp*

from ‹*distinct* A_P› **have** *distinct*$(p \cdot A_P)$ **by** *simp*
moreover from ‹*distinct* A_Q› **have** *distinct*$(q \cdot A_Q)$ **by** *simp*
moreover from ‹*extractFrame* $P = \langle A_P, \Psi_P\rangle$› *Sp* ‹$(p \cdot A_P)$ ♯* Ψ_P›
have *FrP*: *extractFrame* $P = \langle(p \cdot A_P), (p \cdot \Psi_P)\rangle$
by(*simp add*: *frameChainAlpha*)
moreover from ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› *Sq* ‹$(q \cdot A_Q)$ ♯* Ψ_Q›
have *FrQ*: *extractFrame* $Q = \langle(q \cdot A_Q), (q \cdot \Psi_Q)\rangle$
by(*simp add*: *frameChainAlpha*)

moreover have $(q \cdot (\Psi \otimes \Psi_Q)) \rhd P \longmapsto ¿(q \cdot M)(\!|N|\!) \prec P'$ **using** *Sq* ‹A_Q ♯* P› ‹$(q \cdot A_Q)$ ♯* P› ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$›
by − (*rule brinputPermFrameSubject*, (*assumption* | *simp*)+)
then have $(\Psi \otimes (q \cdot \Psi_Q)) \rhd P \longmapsto ¿(q \cdot M)(\!|N|\!) \prec P'$ **using** *Sq* ‹A_Q ♯* Ψ› ‹$(q \cdot A_Q)$ ♯* Ψ›
by(*simp add*: *eqvts*)
with ‹$(q \cdot M) = M$› **have** *PTrans*: $(\Psi \otimes (q \cdot \Psi_Q)) \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$
by *simp*

moreover have $(p \cdot (\Psi \otimes \Psi_P)) \rhd Q \longmapsto ¿(p \cdot M)(\!|N|\!) \prec Q'$ **using** *Sp* ‹A_P ♯* Q› ‹$(p \cdot A_P)$ ♯* Q› ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$›
by − (*rule brinputPermFrameSubject*, (*assumption* | *simp*)+)
then have $(\Psi \otimes (p \cdot \Psi_P)) \rhd Q \longmapsto ¿(p \cdot M)(\!|N|\!) \prec Q'$ **using** *Sp* ‹A_P ♯* Ψ› ‹$(p \cdot A_P)$ ♯* Ψ›
by(*simp add*: *eqvts*)
with ‹$(p \cdot M) = M$› **have** *PTrans*: $(\Psi \otimes (p \cdot \Psi_P)) \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$
by *simp*
moreover from ‹$(p \cdot A_P)$ ♯* A_Q› ‹$(p \cdot A_P)$ ♯* $(q \cdot A_Q)$› ‹$(p \cdot A_P)$ ♯* Ψ_Q› *Sq* **have** $(p \cdot A_P)$ ♯* $(q \cdot \Psi_Q)$
by(*simp add*: *freshChainSimps*)
moreover from ‹$(q \cdot A_Q)$ ♯* A_P› ‹$(p \cdot A_P)$ ♯* A_Q› ‹$(q \cdot A_Q)$ ♯* Ψ_P› ‹$(p \cdot A_P)$ ♯* $(q \cdot A_Q)$› *Sp Sq* **have** $(q \cdot A_Q)$ ♯* $(p \cdot \Psi_P)$
by(*simp add*: *freshChainSimps*)

moreover note
‹$(p \cdot A_P)$ ♯* Ψ› ‹$(p \cdot A_P)$ ♯* M›
‹$(p \cdot A_P)$ ♯* P› ‹$(p \cdot A_P)$ ♯* N› ‹$(p \cdot A_P)$ ♯* P'›
‹$(p \cdot A_P)$ ♯* Q› ‹$(p \cdot A_P)$ ♯* Q'› ‹$(p \cdot A_P)$ ♯* $(q \cdot A_Q)$›
‹$(q \cdot A_Q)$ ♯* Ψ› ‹$(q \cdot A_Q)$ ♯* M›

$\langle(q \cdot A_Q) \sharp* P\rangle$ $\langle(q \cdot A_Q) \sharp* N\rangle$ $\langle(q \cdot A_Q) \sharp* P'\rangle$
$\langle(q \cdot A_Q) \sharp* Q\rangle$ $\langle(q \cdot A_Q) \sharp* Q'\rangle$
ultimately show *?thesis*
by(*simp add*: *cBrMerge*)
qed
}
note *Goal* = *this*

note $\langle\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'\rangle$ $\langle\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'\rangle$
moreover from $\langle$*extractFrame* $P = \langle A_P, \Psi_P\rangle\rangle$ $\langle A_P \sharp* \Psi\rangle$ $\langle A_P \sharp* P\rangle$ $\langle A_P \sharp* Q\rangle$ $\langle A_P \sharp* M\rangle$ $\langle A_P \sharp* A_Q\rangle$
obtain A_P' **where** *extractFrame* $P = \langle A_P', \Psi_P\rangle$ **and** *distinct* A_P' **and** $A_P' \sharp* \Psi$ **and** $A_P' \sharp* P$ **and** $A_P' \sharp* Q$ **and** $A_P' \sharp* M$ **and** $A_P' \sharp* A_Q$
by − (*rule distinctFrame*[**where** C=(Ψ, P, Q, M, A_Q)], *auto*)
moreover from $\langle$*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle\rangle$ $\langle A_Q \sharp* \Psi\rangle$ $\langle A_Q \sharp* P\rangle$ $\langle A_Q \sharp* Q\rangle$ $\langle A_Q \sharp* M\rangle$ $\langle A_P' \sharp* A_Q\rangle$
obtain A_Q' **where** *extractFrame* $Q = \langle A_Q', \Psi_Q\rangle$ **and** *distinct* A_Q' **and** $A_Q' \sharp* \Psi$ **and** $A_Q' \sharp* P$ **and** $A_Q' \sharp* Q$ **and** $A_Q' \sharp* M$ **and** $A_P' \sharp* A_Q'$
by − (*rule distinctFrame*[**where** C=(Ψ, P, Q, M, A_P')], *auto*)
ultimately show *?thesis*
by(*metis Goal*)
qed

lemma *BrComm1*:
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 *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 $\Psi \otimes \Psi_P \rhd Q \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec Q'$
and *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$
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 $A_Q \sharp* M$
and $xvec \sharp* P$

shows $\Psi \rhd P \parallel Q \longmapsto ¡M(\!|\nu* xvec|\!)\langle N\rangle \prec (P' \parallel Q')$
proof −
{
fix 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 *xvec* :: *name list*
and Q' :: $('a, 'b, 'c)$ *psi*
and A_Q :: *name list*

assume $\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$
and *extractFrame* $P = \langle A_P, \Psi_P\rangle$
and *distinct* A_P
and $\Psi \otimes \Psi_P \rhd Q \longmapsto ¡M(\!|\nu* xvec|\!)\langle N\rangle \prec Q'$
and *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$
and *distinct* A_Q
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 $A_Q \sharp* M$
and $xvec \sharp* P$

have $\Psi \rhd P \parallel Q \longmapsto ¡M(\!|\nu* xvec|\!)\langle N\rangle \prec (P' \parallel Q')$
proof −
obtain r::*name prm* **where** $(r \cdot xvec) \sharp* \Psi$ **and** $(r \cdot xvec) \sharp* P$ **and** $(r \cdot xvec) \sharp* Q$ **and** $(r \cdot xvec) \sharp* M$
and $(r \cdot xvec) \sharp* N$ **and** $(r \cdot xvec) \sharp* A_P$ **and** $(r \cdot xvec) \sharp* A_Q$
and $(r \cdot xvec) \sharp* P'$ **and** $(r \cdot xvec) \sharp* Q'$ **and** $(r \cdot xvec) \sharp* \Psi_P$ **and** $(r \cdot xvec) \sharp* \Psi_Q$
and *Sr*: $(set\ r) \subseteq (set\ xvec) \times (set(r \cdot xvec))$ **and** *distinctPerm r*
by(*rule name-list-avoiding*[**where** *xvec*=*xvec* **and** *c*=(Ψ, P, Q, M, N, A_P, A_Q, Ψ_P, Ψ_Q, P', Q')])
(*auto simp add*: *eqvts fresh-star-prod*)
obtain q::*name prm* **where** $(q \cdot A_Q) \sharp* \Psi$ **and** $(q \cdot A_Q) \sharp* P$ **and** $(q \cdot A_Q) \sharp* Q$ **and** $(q \cdot A_Q) \sharp* M$

and $(q \cdot A_Q) \sharp* (r \cdot N)$ **and** $(q \cdot A_Q) \sharp* (r \cdot xvec)$ **and** $(q \cdot A_Q) \sharp* (r \cdot Q')$
and $(q \cdot A_Q) \sharp* (r \cdot P')$ **and** $(q \cdot A_Q) \sharp* \Psi_P$ **and** $(q \cdot A_Q) \sharp* A_P$ **and** $(q \cdot A_Q) \sharp* \Psi_Q$
and *Sq*: *set q* $\subseteq$ *set* $A_Q \times set(q \cdot A_Q)$
by(*rule name-list-avoiding*[**where** *xvec*=A_Q **and** *c*=(Ψ, *P*, *Q*, *M*, $r \cdot N$, $r \cdot xvec$, Ψ_Q, A_P, Ψ_P, $r \cdot Q'$, $r \cdot P'$)])
(*auto simp add*: *eqvts fresh-star-prod*)
obtain *p*::*name prm* **where** $(p \cdot A_P) \sharp* \Psi$ **and** $(p \cdot A_P) \sharp* P$ **and** $(p \cdot A_P) \sharp* Q$ **and** $(p \cdot A_P) \sharp* M$
and $(p \cdot A_P) \sharp* (r \cdot N)$ **and** $(p \cdot A_P) \sharp* (r \cdot xvec)$ **and** $(p \cdot A_P) \sharp* (r \cdot Q')$
and $(p \cdot A_P) \sharp* (r \cdot P')$ **and** $(p \cdot A_P) \sharp* \Psi_P$ **and** $(p \cdot A_P) \sharp* \Psi_Q$ **and** $(p \cdot A_P) \sharp* A_Q$
and $(p \cdot A_P) \sharp* (q \cdot A_Q)$ **and** *Sp*: (*set p*) $\subseteq$ (*set* A_P) $\times$ (*set*$(p \cdot A_P)$)
by(*rule name-list-avoiding*[**where** *xvec*=A_P **and** *c*=(Ψ, *P*, *Q*, *M*, $r \cdot N$, $r \cdot xvec$, A_Q, $q \cdot A_Q$, Ψ_Q, Ψ_P, $r \cdot Q'$, $r \cdot P'$)])
(*auto simp add*: *eqvts fresh-star-prod*)

have *FrP*: *extractFrame P* = $\langle A_P, \Psi_P \rangle$ **by** *fact*
have *FrQ*: *extractFrame Q* = $\langle A_Q, \Psi_Q \rangle$ **by** *fact*

from *Sp* ‹$A_P \sharp* M$› ‹$(p \cdot A_P) \sharp* M$›
have $(p \cdot M) = M$
by *simp*
from *Sq* ‹$A_Q \sharp* M$› ‹$(q \cdot A_Q) \sharp* M$›
have $(q \cdot M) = M$
by *simp*

from ‹$A_P \sharp* Q$› *FrQ* ‹$A_P \sharp* A_Q$› **have** $A_P \sharp* \Psi_Q$
by(*auto dest*: *extractFrameFreshChain*)
from ‹$A_Q \sharp* P$› *FrP* ‹$A_P \sharp* A_Q$› **have** $A_Q \sharp* \Psi_P$
by(*auto dest*: *extractFrameFreshChain*)
from ‹$(r \cdot xvec) \sharp* A_P$› ‹$(p \cdot A_P) \sharp* (r \cdot xvec)$› ‹$(r \cdot xvec) \sharp* A_P$› *Sp* **have** $(r \cdot xvec) \sharp* (p \cdot A_P)$
by(*simp add*: *freshChainSimps*)

from ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$› *Sr* ‹*distinctPerm r*› ‹*xvec* $\sharp*$ *P*› ‹$(r \cdot xvec) \sharp* P$›
have $\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|(r \cdot N)|\!) \prec (r \cdot P')$
by(*rule brinputAlpha*)
then have $(q \cdot (\Psi \otimes \Psi_Q)) \rhd P \longmapsto ¿(q \cdot M)(\!|(r \cdot N)|\!) \prec (r \cdot P')$ **using** *Sq* ‹$A_Q \sharp* P$› ‹$(q \cdot A_Q) \sharp* P$›
by − (*rule brinputPermFrameSubject*, (*assumption* | *simp*)+)
then have $\Psi \otimes (q \cdot \Psi_Q) \rhd P \longmapsto ¿(q \cdot M)(\!|(r \cdot N)|\!) \prec (r \cdot P')$ **using** *Sq* ‹$A_Q \sharp* \Psi$› ‹$(q \cdot A_Q) \sharp* \Psi$›
by(*simp add*: *eqvts*)
with ‹$(q \cdot M) = M$› **have** *PTrans*: $\Psi \otimes (q \cdot \Psi_Q) \rhd P \longmapsto ¿M(\!|(r \cdot N)|\!) \prec (r \cdot P')$ **by** *simp*

moreover from ‹*extractFrame P* = $\langle A_P, \Psi_P \rangle$› *Sp* ‹$(p \cdot A_P) \sharp* \Psi_P$›

have *FrP*: $\mathit{extractFrame}\ P = \langle(p \cdot A_P), (p \cdot \Psi_P)\rangle$
by(*simp add*: *frameChainAlpha*)
moreover from ‹$\mathit{distinct}\ A_P$› **have** $\mathit{distinct}(p \cdot A_P)$ **by** *simp*

moreover from ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec Q'$› *Sr* ‹$(r \cdot xvec) \sharp* N$› ‹$(r \cdot xvec) \sharp* Q'$›
have $\Psi \otimes \Psi_P \rhd Q \longmapsto ¡M(\!|\nu * (r \cdot xvec)|\!)\langle(r \cdot N)\rangle \prec (r \cdot Q')$
by(*simp add*: *boundOutputChainAlpha''* *create-residual.simps*)
then have $(p \cdot (\Psi \otimes \Psi_P)) \rhd Q \longmapsto ¡(p \cdot M)(\!|\nu * (r \cdot xvec)|\!)\langle(r \cdot N)\rangle \prec (r \cdot Q')$ **using** *Sp* ‹$A_P \sharp* Q$› ‹$(p \cdot A_P) \sharp* Q$› ‹$(r \cdot xvec) \sharp* M$› ‹$(r \cdot xvec) \sharp* A_P$› ‹$(r \cdot xvec) \sharp* (p \cdot A_P)$›
by(*fastforce intro*: *broutputPermFrameSubject*)
then have $\Psi \otimes (p \cdot \Psi_P) \rhd Q \longmapsto ¡(p \cdot M)(\!|\nu * (r \cdot xvec)|\!)\langle(r \cdot N)\rangle \prec (r \cdot Q')$ **using** *Sp* ‹$A_P \sharp* \Psi$› ‹$(p \cdot A_P) \sharp* \Psi$›
by(*simp add*: *eqvts*)
with ‹$(p \cdot M) = M$› **have** *QTrans*: $\Psi \otimes (p \cdot \Psi_P) \rhd Q \longmapsto ¡M(\!|\nu * (r \cdot xvec)|\!)\langle(r \cdot N)\rangle \prec (r \cdot Q')$ **by** *simp*
moreover then have $\mathit{distinct}(r \cdot xvec)$ **by**(*force dest*: *boundOutputDistinct*)
moreover from ‹$\mathit{extractFrame}\ Q = \langle A_Q, \Psi_Q\rangle$› *Sq* ‹$(q \cdot A_Q) \sharp* \Psi_Q$›
have *FrQ*: $\mathit{extractFrame}\ Q = \langle(q \cdot A_Q), (q \cdot \Psi_Q)\rangle$
by(*simp add*: *frameChainAlpha*)
moreover from ‹$\mathit{distinct}\ A_Q$› **have** $\mathit{distinct}(q \cdot A_Q)$ **by** *simp*

moreover note ‹$(p \cdot A_P) \sharp* \Psi$›
moreover from ‹$(p \cdot A_P) \sharp* A_Q$› ‹$(p \cdot A_P) \sharp* (q \cdot A_Q)$› ‹$(p \cdot A_P) \sharp* \Psi_Q$› *Sq* **have** $(p \cdot A_P) \sharp* (q \cdot \Psi_Q)$
by(*simp add*: *freshChainSimps*)
moreover note ‹$(p \cdot A_P) \sharp* P$› ‹$(p \cdot A_P) \sharp* M$›
moreover note ‹$(p \cdot A_P) \sharp* (r \cdot N)$› ‹$(p \cdot A_P) \sharp* (r \cdot P')$› ‹$(p \cdot A_P) \sharp* Q$› ‹$(p \cdot A_P) \sharp* (r \cdot Q')$› ‹$(p \cdot A_P) \sharp* (q \cdot A_Q)$›
‹$(p \cdot A_P) \sharp* (r \cdot xvec)$› ‹$(q \cdot A_Q) \sharp* \Psi$›
moreover from ‹$(q \cdot A_Q) \sharp* A_P$› ‹$(p \cdot A_P) \sharp* A_Q$› ‹$(q \cdot A_Q) \sharp* \Psi_P$› ‹$(p \cdot A_P) \sharp* (q \cdot A_Q)$› *Sp Sq* **have** $(q \cdot A_Q) \sharp* (p \cdot \Psi_P)$
by(*simp add*: *freshChainSimps*)
moreover note ‹$(q \cdot A_Q) \sharp* P$› ‹$(q \cdot A_Q) \sharp* (r \cdot N)$›‹$(q \cdot A_Q) \sharp* (r \cdot P')$› ‹$(q \cdot A_Q) \sharp* Q$› ‹$(q \cdot A_Q) \sharp* M$›
moreover note ‹$(q \cdot A_Q) \sharp* (r \cdot Q')$› ‹$(q \cdot A_Q) \sharp* (r \cdot xvec)$› ‹$(r \cdot xvec) \sharp* \Psi$›
moreover from ‹$(r \cdot xvec) \sharp* A_P$› ‹$(p \cdot A_P) \sharp* (r \cdot xvec)$› ‹$(r \cdot xvec) \sharp* \Psi_P$› *Sp* **have** $(r \cdot xvec) \sharp* (p \cdot \Psi_P)$
by(*simp add*: *freshChainSimps*)
moreover from ‹$(r \cdot xvec) \sharp* A_Q$› ‹$(q \cdot A_Q) \sharp* (r \cdot xvec)$› ‹$(r \cdot xvec) \sharp* \Psi_Q$› *Sq* **have** $(r \cdot xvec) \sharp* (q \cdot \Psi_Q)$
by(*simp add*: *freshChainSimps*)
moreover note ‹$(r \cdot xvec) \sharp* P$› ‹$(r \cdot xvec) \sharp* M$›
moreover note ‹$(r \cdot xvec) \sharp* Q$› ‹$(r \cdot xvec) \sharp* M$›
ultimately have $\Psi \rhd P \parallel Q \longmapsto ¡M(\!|\nu * (r \cdot xvec)|\!)\langle(r \cdot N)\rangle \prec ((r \cdot P') \parallel (r \cdot Q'))$
by − (*rule cBrComm1*)

then have *permuted*: $\Psi \rhd P \parallel Q \longmapsto ¡M(\!|\nu*(r \cdot xvec)|\!)\langle(r \cdot N)\rangle \prec (r \cdot (P' \parallel Q'))$ **by** *simp*
note $\langle(r \cdot xvec) \sharp* N\rangle$
moreover from $\langle(r \cdot xvec) \sharp* P'\rangle$ $\langle(r \cdot xvec) \sharp* Q'\rangle$
have $(r \cdot xvec) \sharp* (P' \parallel Q')$ **by** *simp*
moreover note *Sr*
moreover have *set xvec* $\subseteq$ *set xvec* **by** *simp*
ultimately have $(\!|\nu*xvec|\!)N \prec' (P' \parallel Q') = (\!|\nu*(r \cdot xvec)|\!)(r \cdot N) \prec' (r \cdot (P' \parallel Q'))$
by(*rule boundOutputChainAlpha''*)
then have $¡M(\!|\nu*xvec|\!)\langle N\rangle \prec (P' \parallel Q') = ¡M(\!|\nu*(r \cdot xvec)|\!)\langle(r \cdot N)\rangle \prec (r \cdot (P' \parallel Q'))$
by(*simp only*: *create-residual.simps*)
with *permuted* **show** *?thesis*
by *simp*
qed
}
note *Goal* = *this*

note $\langle\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'\rangle$ $\langle\Psi \otimes \Psi_P \rhd Q \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec Q'\rangle$
moreover from $\langle$*extractFrame* $P = \langle A_P, \Psi_P\rangle\rangle$ $\langle A_P \sharp* \Psi\rangle$ $\langle A_P \sharp* P\rangle$ $\langle A_P \sharp* Q\rangle$ $\langle A_P \sharp* M\rangle$ $\langle A_P \sharp* A_Q\rangle$
obtain A_P' **where** *extractFrame* $P = \langle A_P', \Psi_P\rangle$ **and** *distinct* A_P' **and** $A_P' \sharp* \Psi$ **and** $A_P' \sharp* P$ **and** $A_P' \sharp* Q$ **and** $A_P' \sharp* M$ **and** $A_P' \sharp* A_Q$
by − (*rule distinctFrame*[**where** C=(Ψ, P, Q, M, A_Q)], *auto*)
moreover from $\langle$*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle\rangle$ $\langle A_Q \sharp* \Psi\rangle$ $\langle A_Q \sharp* P\rangle$ $\langle A_Q \sharp* Q\rangle$ $\langle A_Q \sharp* M\rangle$ $\langle A_P' \sharp* A_Q\rangle$
obtain A_Q' **where** *extractFrame* $Q = \langle A_Q', \Psi_Q\rangle$ **and** *distinct* A_Q' **and** $A_Q' \sharp* \Psi$ **and** $A_Q' \sharp* P$ **and** $A_Q' \sharp* Q$ **and** $A_Q' \sharp* M$ **and** $A_P' \sharp* A_Q'$
by − (*rule distinctFrame*[**where** C=(Ψ, P, Q, M, A_P')], *auto*)
ultimately show *?thesis* **using** $\langle xvec \sharp* P\rangle$
by(*metis Goal*)
qed

lemma *BrComm2*:
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 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 = ⟨A_P, Ψ_P⟩
  and   Ψ ⊗ Ψ_P ▷ Q ⟼¡M(⦇N⦈) ≺ Q'
  and   extractFrame Q = ⟨A_Q, Ψ_Q⟩
  and   A_P ♯* Ψ
  and   A_P ♯* P
  and   A_P ♯* Q
  and   A_P ♯* M
  and   A_P ♯* A_Q
  and   A_Q ♯* Ψ
  and   A_Q ♯* P
  and   A_Q ♯* Q
  and   A_Q ♯* M
  and   xvec ♯* Q

shows Ψ ▷ P ∥ Q ⟼¡M(⦇ν*xvec⦈)⟨N⟩ ≺ (P' ∥ Q')
proof −
  {
    fix 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 Q'   :: ('a, 'b, 'c) psi
      and A_Q  :: name list

    assume Ψ ⊗ Ψ_Q ▷ P ⟼¡M(⦇ν*xvec⦈)⟨N⟩ ≺ P'
      and  extractFrame P = ⟨A_P, Ψ_P⟩
      and  distinct A_P
      and  Ψ ⊗ Ψ_P ▷ Q ⟼¡M(⦇N⦈) ≺ Q'
      and  extractFrame Q = ⟨A_Q, Ψ_Q⟩
      and  distinct A_Q
      and  A_P ♯* Ψ
      and  A_P ♯* P
      and  A_P ♯* Q
      and  A_P ♯* M
      and  A_P ♯* A_Q
      and  A_Q ♯* Ψ
      and  A_Q ♯* P
      and  A_Q ♯* Q
      and  A_Q ♯* M
      and  xvec ♯* Q

    have Ψ ▷ P ∥ Q ⟼¡M(⦇ν*xvec⦈)⟨N⟩ ≺ (P' ∥ Q')
    proof −
```

obtain r::*name prm* **where** $(r \cdot xvec) \sharp* \Psi$ **and** $(r \cdot xvec) \sharp* P$ **and** $(r \cdot xvec) \sharp* Q$ **and** $(r \cdot xvec) \sharp* M$
and $(r \cdot xvec) \sharp* M$ **and** $(r \cdot xvec) \sharp* N$ **and** $(r \cdot xvec) \sharp* A_P$ **and** $(r \cdot xvec) \sharp* A_Q$
and $(r \cdot xvec) \sharp* P'$ **and** $(r \cdot xvec) \sharp* Q'$ **and** $(r \cdot xvec) \sharp* \Psi_P$ **and** $(r \cdot xvec) \sharp* \Psi_Q$
and *Sr*: $(set\ r) \subseteq (set\ xvec) \times (set(r \cdot xvec))$ **and** *distinctPerm r*
by(*rule name-list-avoiding*[**where** *xvec*=*xvec* **and** *c*=(Ψ, P, Q, M, N, A_P, A_Q, Ψ_P, Ψ_Q, P', Q')])
(*auto simp add*: *eqvts fresh-star-prod*)

obtain q::*name prm* **where** $(q \cdot A_Q) \sharp* \Psi$ **and** $(q \cdot A_Q) \sharp* P$ **and** $(q \cdot A_Q) \sharp* Q$ **and** $(q \cdot A_Q) \sharp* M$
and $(q \cdot A_Q) \sharp* (r \cdot N)$ **and** $(q \cdot A_Q) \sharp* (r \cdot xvec)$ **and** $(q \cdot A_Q) \sharp* (r \cdot Q')$
and $(q \cdot A_Q) \sharp* (r \cdot P')$ **and** $(q \cdot A_Q) \sharp* \Psi_P$ **and** $(q \cdot A_Q) \sharp* A_P$ **and** $(q \cdot A_Q) \sharp* \Psi_Q$
and *Sq*: $set\ q \subseteq set\ A_Q \times set(q \cdot A_Q)$
by(*rule name-list-avoiding*[**where** *xvec*=A_Q **and** *c*=(Ψ, P, Q, M, $r \cdot N$, $r \cdot xvec$, Ψ_Q, A_P, Ψ_P, $r \cdot Q'$, $r \cdot P'$)])
(*auto simp add*: *eqvts fresh-star-prod*)

obtain p::*name prm* **where** $(p \cdot A_P) \sharp* \Psi$ **and** $(p \cdot A_P) \sharp* P$ **and** $(p \cdot A_P) \sharp* Q$ **and** $(p \cdot A_P) \sharp* M$
and $(p \cdot A_P) \sharp* (r \cdot N)$ **and** $(p \cdot A_P) \sharp* (r \cdot xvec)$ **and** $(p \cdot A_P) \sharp* (r \cdot Q')$
and $(p \cdot A_P) \sharp* (r \cdot P')$ **and** $(p \cdot A_P) \sharp* \Psi_P$ **and** $(p \cdot A_P) \sharp* \Psi_Q$ **and** $(p \cdot A_P) \sharp* A_Q$
and $(p \cdot A_P) \sharp* (q \cdot A_Q)$ **and** *Sp*: $(set\ p) \subseteq (set\ A_P) \times (set(p \cdot A_P))$
by(*rule name-list-avoiding*[**where** *xvec*=A_P **and** *c*=(Ψ, P, Q, M, $r \cdot N$, $r \cdot xvec$, A_Q, $q \cdot A_Q$, Ψ_Q, Ψ_P, $r \cdot Q'$, $r \cdot P'$)])
(*auto simp add*: *eqvts fresh-star-prod*)

have *FrP*: *extractFrame* $P = \langle A_P, \Psi_P \rangle$ **by** *fact*
have *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$ **by** *fact*

from *Sp* ‹$A_P \sharp* M$› ‹$(p \cdot A_P) \sharp* M$›
have $(p \cdot M) = M$
by *simp*
from *Sq* ‹$A_Q \sharp* M$› ‹$(q \cdot A_Q) \sharp* M$›
have $(q \cdot M) = M$
by *simp*

from ‹$A_P \sharp* Q$› *FrQ* ‹$A_P \sharp* A_Q$› **have** $A_P \sharp* \Psi_Q$
by(*auto dest*: *extractFrameFreshChain*)
from ‹$A_Q \sharp* P$› *FrP* ‹$A_P \sharp* A_Q$› **have** $A_Q \sharp* \Psi_P$
by(*auto dest*: *extractFrameFreshChain*)
from ‹$(r \cdot xvec) \sharp* A_Q$› ‹$(q \cdot A_Q) \sharp* (r \cdot xvec)$› ‹$(r \cdot xvec) \sharp* A_Q$› *Sq* **have** $(r \cdot xvec) \sharp* (q \cdot A_Q)$
by(*simp add*: *freshChainSimps*)

from ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto ¡M(\!|\nu* xvec|\!)\langle N \rangle \prec P'$› *Sr* ‹$(r \cdot xvec) \sharp* N$› ‹$(r \cdot xvec) \sharp* P'$›

have $\Psi \otimes \Psi_Q \rhd P \longmapsto ¡M(\!|\nu*(r \cdot xvec)|\!)\langle(r \cdot N)\rangle \prec (r \cdot P')$
by(*simp add*: *boundOutputChainAlpha''* *create-residual.simps*)
then have $(q \cdot (\Psi \otimes \Psi_Q)) \rhd P \longmapsto ¡(q \cdot M)(\!|\nu*(r \cdot xvec)|\!)\langle(r \cdot N)\rangle \prec (r \cdot P')$ **using** *Sq* ‹$A_Q \sharp* P$› ‹$(q \cdot A_Q) \sharp* P$› ‹$(r \cdot xvec) \sharp* M$› ‹$(r \cdot xvec) \sharp* A_Q$› ‹$(r \cdot xvec) \sharp* (q \cdot A_Q)$›
by(*fastforce intro*: *broutputPermFrameSubject*)
then have $\Psi \otimes (q \cdot \Psi_Q) \rhd P \longmapsto ¡(q \cdot M)(\!|\nu*(r \cdot xvec)|\!)\langle(r \cdot N)\rangle \prec (r \cdot P')$ **using** *Sq* ‹$A_Q \sharp* \Psi$› ‹$(q \cdot A_Q) \sharp* \Psi$›
by(*simp add*: *eqvts*)
with ‹$(q \cdot M) = M$› **have** *PTrans*: $\Psi \otimes (q \cdot \Psi_Q) \rhd P \longmapsto ¡M(\!|\nu*(r \cdot xvec)|\!)\langle(r \cdot N)\rangle \prec (r \cdot P')$ **by** *simp*
moreover then have *distinct*$(r \cdot xvec)$ **by**(*force dest*: *boundOutputDistinct*)

moreover from ‹*extractFrame* $P = \langle A_P, \Psi_P\rangle$› *Sp* ‹$(p \cdot A_P) \sharp* \Psi_P$›
have *FrP*: *extractFrame* $P = \langle(p \cdot A_P), (p \cdot \Psi_P)\rangle$
by(*simp add*: *frameChainAlpha*)
moreover from ‹*distinct* A_P› **have** *distinct*$(p \cdot A_P)$ **by** *simp*

moreover from ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$› *Sr* ‹*distinctPerm r*› ‹*xvec* $\sharp* Q$› ‹$(r \cdot xvec) \sharp* Q$›
have $\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|(r \cdot N)|\!) \prec (r \cdot Q')$
by(*rule brinputAlpha*)
then have $(p \cdot (\Psi \otimes \Psi_P)) \rhd Q \longmapsto ¿(p \cdot M)(\!|(r \cdot N)|\!) \prec (r \cdot Q')$ **using** *Sp* ‹$A_P \sharp* Q$› ‹$(p \cdot A_P) \sharp* Q$›
by $-$ (*rule brinputPermFrameSubject*, (*assumption* | *simp*)+)
then have *QTrans*: $\Psi \otimes (p \cdot \Psi_P) \rhd Q \longmapsto ¿(p \cdot M)(\!|(r \cdot N)|\!) \prec (r \cdot Q')$ **using** *Sp* ‹$A_P \sharp* \Psi$› ‹$(p \cdot A_P) \sharp* \Psi$›
by(*simp add*: *eqvts*)
with ‹$(p \cdot M) = M$› **have** $\Psi \otimes (p \cdot \Psi_P) \rhd Q \longmapsto ¿M(\!|(r \cdot N)|\!) \prec (r \cdot Q')$ **by** *simp*

moreover from ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› *Sq* ‹$(q \cdot A_Q) \sharp* \Psi_Q$›
have *FrQ*: *extractFrame* $Q = \langle(q \cdot A_Q), (q \cdot \Psi_Q)\rangle$
by(*simp add*: *frameChainAlpha*)
moreover from ‹*distinct* A_Q› **have** *distinct*$(q \cdot A_Q)$ **by** *simp*

moreover note ‹$(p \cdot A_P) \sharp* \Psi$›
moreover from ‹$(p \cdot A_P) \sharp* A_Q$› ‹$(p \cdot A_P) \sharp* (q \cdot A_Q)$› ‹$(p \cdot A_P) \sharp* \Psi_Q$› *Sq* **have** $(p \cdot A_P) \sharp* (q \cdot \Psi_Q)$
by(*simp add*: *freshChainSimps*)
moreover note ‹$(p \cdot A_P) \sharp* P$› ‹$(p \cdot A_P) \sharp* M$›
moreover note ‹$(p \cdot A_P) \sharp* (r \cdot N)$› ‹$(p \cdot A_P) \sharp* (r \cdot P')$› ‹$(p \cdot A_P) \sharp* Q$› ‹$(p \cdot A_P) \sharp* (r \cdot Q')$› ‹$(p \cdot A_P) \sharp* (q \cdot A_Q)$›
‹$(p \cdot A_P) \sharp* (r \cdot xvec)$› ‹$(q \cdot A_Q) \sharp* \Psi$›
moreover from ‹$(q \cdot A_Q) \sharp* A_P$› ‹$(p \cdot A_P) \sharp* A_Q$› ‹$(q \cdot A_Q) \sharp* \Psi_P$› ‹$(p \cdot A_P) \sharp* (q \cdot A_Q)$› *Sp Sq* **have** $(q \cdot A_Q) \sharp* (p \cdot \Psi_P)$
by(*simp add*: *freshChainSimps*)
moreover note ‹$(q \cdot A_Q) \sharp* P$› ‹$(q \cdot A_Q) \sharp* (r \cdot N)$›‹$(q \cdot A_Q) \sharp* (r \cdot P')$› ‹$(q \cdot A_Q) \sharp* Q$› ‹$(q \cdot A_Q) \sharp* M$›

moreover note ‹$(q \cdot A_Q) \sharp* (r \cdot Q')$› ‹$(q \cdot A_Q) \sharp* (r \cdot xvec)$› ‹$(r \cdot xvec) \sharp* \Psi$›
moreover from ‹$(r \cdot xvec) \sharp* A_P$› ‹$(p \cdot A_P) \sharp* (r \cdot xvec)$› ‹$(r \cdot xvec) \sharp* \Psi_P$› *Sp* **have** $(r \cdot xvec) \sharp* (p \cdot \Psi_P)$
by(*simp add*: *freshChainSimps*)
moreover from ‹$(r \cdot xvec) \sharp* A_Q$› ‹$(q \cdot A_Q) \sharp* (r \cdot xvec)$› ‹$(r \cdot xvec) \sharp* \Psi_Q$› *Sq* **have** $(r \cdot xvec) \sharp* (q \cdot \Psi_Q)$
by(*simp add*: *freshChainSimps*)
moreover note ‹$(r \cdot xvec) \sharp* P$› ‹$(r \cdot xvec) \sharp* M$›
moreover note ‹$(r \cdot xvec) \sharp* Q$› ‹$(r \cdot xvec) \sharp* M$›
ultimately have $\Psi \rhd P \parallel Q \longmapsto ¡M(\!|\nu*(r \cdot xvec)|\!)\langle(r \cdot N)\rangle \prec ((r \cdot P') \parallel (r \cdot Q'))$
by − (*rule cBrComm2*)
then have *permuted*: $\Psi \rhd P \parallel Q \longmapsto ¡M(\!|\nu*(r \cdot xvec)|\!)\langle(r \cdot N)\rangle \prec (r \cdot (P' \parallel Q'))$ **by** *simp*
note ‹$(r \cdot xvec) \sharp* N$›
moreover from ‹$(r \cdot xvec) \sharp* P'$› ‹$(r \cdot xvec) \sharp* Q'$›
have $(r \cdot xvec) \sharp* (P' \parallel Q')$ **by** *simp*
moreover note *Sr*
moreover have *set xvec* ⊆ *set xvec* **by** *simp*
ultimately have $(\!|\nu*xvec|\!)N \prec' (P' \parallel Q') = (\!|\nu*(r \cdot xvec)|\!)(r \cdot N) \prec' (r \cdot (P' \parallel Q'))$
by(*rule boundOutputChainAlpha''*)
then have $¡M(\!|\nu*xvec|\!)\langle N\rangle \prec (P' \parallel Q') = ¡M(\!|\nu*(r \cdot xvec)|\!)\langle(r \cdot N)\rangle \prec (r \cdot (P' \parallel Q'))$
by(*simp only*: *create-residual.simps*)
with *permuted* **show** *?thesis*
by *simp*
qed
}
note *Goal* = *this*

note ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec P'$› ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$›
moreover from ‹*extractFrame* $P = \langle A_P, \Psi_P\rangle$› ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_P \sharp* Q$› ‹$A_P \sharp* M$› ‹$A_P \sharp* A_Q$›
obtain A_P' **where** *extractFrame* $P = \langle A_P', \Psi_P\rangle$ **and** *distinct* A_P' **and** $A_P' \sharp* \Psi$ **and** $A_P' \sharp* P$ **and** $A_P' \sharp* Q$ **and** $A_P' \sharp* M$ **and** $A_P' \sharp* A_Q$
by − (*rule distinctFrame*[**where** *C*=(Ψ, *P*, *Q*, *M*, A_Q)], *auto*)
moreover from ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› ‹$A_Q \sharp* \Psi$› ‹$A_Q \sharp* P$› ‹$A_Q \sharp* Q$› ‹$A_Q \sharp* M$› ‹$A_P' \sharp* A_Q$›
obtain A_Q' **where** *extractFrame* $Q = \langle A_Q', \Psi_Q\rangle$ **and** *distinct* A_Q' **and** $A_Q' \sharp* \Psi$ **and** $A_Q' \sharp* P$ **and** $A_Q' \sharp* Q$ **and** $A_Q' \sharp* M$ **and** $A_P' \sharp* A_Q'$
by − (*rule distinctFrame*[**where** *C*=(Ψ, *P*, *Q*, *M*, A_P')], *auto*)
ultimately show *?thesis* **using** ‹*xvec* $\sharp*$ *Q*›
by(*metis Goal*)
qed

lemma *BrClose*:
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 x :: *name*

assumes $\Psi \rhd P \longmapsto ¡M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P'$
and $x \in supp\ M$
and $x \sharp \Psi$

shows $\Psi \rhd (\!|\nu x|\!)P \longmapsto \tau \prec (\!|\nu x|\!)((\!|\nu{*}xvec|\!)P')$
proof −
obtain p **where** *xvecFreshPsi*: $((p::name\ prm) \cdot (xvec::name\ list)) \sharp* \Psi$
and *xvecFreshM*: $(p \cdot xvec) \sharp* M$
and *xvecFreshN*: $(p \cdot xvec) \sharp* N$
and *xvecFreshP*: $(p \cdot xvec) \sharp* P$
and *xvecFreshP*: $(p \cdot xvec) \sharp* P'$
and *xvecFrechx*: $(p \cdot xvec) \sharp* x$
and S: $(set\ p) \subseteq (set\ xvec) \times (set(p \cdot xvec))$
and *dp*: *distinctPerm p*
by(*rule name-list-avoiding*[**where** *xvec*=*xvec* **and** c=(Ψ, M, N, P, P', x)])
(*auto simp add*: *eqvts fresh-star-prod*)

obtain y::*name* **where** $y \sharp P$ **and** $y \sharp xvec$ **and** $y \neq x$ **and** $y \sharp N$
and $y \sharp (p \cdot xvec)$ **and** $y \sharp (p \cdot P')$
and $y \sharp M$ **and** $y \sharp \Psi$ **and** $y \sharp P'$
by(*generate-fresh name*) (*auto simp add*: *freshChainSimps*)

from ‹$y \sharp (p \cdot xvec)$› ‹$y \sharp (p \cdot P')$›
have *yFreshRes*: $y \sharp ((\!|\nu{*}(p \cdot xvec)|\!)(p \cdot P'))$
by(*simp add*: *resChainFresh*)

from ‹$\Psi \rhd P \longmapsto ¡M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P'$› S
‹$(p \cdot xvec) \sharp* N$› ‹$(p \cdot xvec) \sharp* P'$›
have $\Psi \rhd P \longmapsto ¡M(\!|\nu{*}(p \cdot xvec)|\!)\langle(p \cdot N)\rangle \prec (p \cdot P')$
by(*simp add*: *alphaOutputResidual*)

then have $[(x, y)] \cdot (\Psi \rhd P \longmapsto ¡M(\!|\nu{*}(p \cdot xvec)|\!)\langle(p \cdot N)\rangle \prec (p \cdot P'))$
by *simp*
with ‹$(p \cdot xvec) \sharp* x$› ‹$y \sharp (p \cdot xvec)$›
‹$x \sharp \Psi$› ‹$y \sharp \Psi$›
have *pretrans*: $\Psi \rhd ([(x, y)] \cdot P) \longmapsto ¡([(x, y)] \cdot M)(\!|\nu{*}(p \cdot xvec)|\!)\langle([(x, y)] \cdot (p$
$\cdot N))\rangle \prec ([(x, y)] \cdot (p \cdot P'))$
by(*simp add*: *eqvts*)

moreover from ‹$x \in supp\ M$› ‹$y \sharp M$›
have $y \in supp\ ([(x, y)] \cdot M)$
by (*metis fresh-bij fresh-def swap-simps*)

moreover from *pretrans*
have *distinct* $(p \cdot xvec)$
by(*force dest*: *boundOutputDistinct*)

moreover note $\langle(p \cdot xvec) \sharp* \Psi\rangle$
moreover from $\langle(p \cdot xvec) \sharp* P\rangle$ $\langle(p \cdot xvec) \sharp* x\rangle$ $\langle y \sharp (p \cdot xvec)\rangle$
have $(p \cdot xvec) \sharp* ([(x, y)] \cdot P)$ **by** *simp*
moreover from $\langle(p \cdot xvec) \sharp* M\rangle$ $\langle(p \cdot xvec) \sharp* x\rangle$ $\langle y \sharp (p \cdot xvec)\rangle$
have $(p \cdot xvec) \sharp* ([(x, y)] \cdot M)$ **by** *simp*
moreover note $\langle y \sharp \Psi\rangle$ $\langle y \sharp (p \cdot xvec)\rangle$

ultimately have $\Psi \rhd (\!|\nu y|\!)([(x, y)] \cdot P) \longmapsto \tau \prec (\!|\nu y|\!)((\!|\nu*(p \cdot xvec)|\!)([(x, y)] \cdot (p \cdot P')))$
by(*rule cBrClose*)
with $\langle(p \cdot xvec) \sharp* x\rangle$ $\langle y \sharp (p \cdot xvec)\rangle$
have $\Psi \rhd (\!|\nu y|\!)([(x, y)] \cdot P) \longmapsto \tau \prec (\!|\nu y|\!)([(x, y)] \cdot ((\!|\nu*(p \cdot xvec)|\!)(p \cdot P')))$
by(*simp add*: *eqvts*)
with *yFreshRes* $\langle y \sharp P\rangle$
have $\Psi \rhd (\!|\nu x|\!)P \longmapsto \tau \prec (\!|\nu x|\!)((\!|\nu*(p \cdot xvec)|\!)(p \cdot P'))$
by(*simp add*: *alphaRes*)

with $\langle(p \cdot xvec) \sharp* P'\rangle$ *S*
show *?thesis*
by(*simp add*: *resChainAlpha*)
qed

lemma *semanticsCasesAux*[*consumes 1*, *case-names cInput cBrInput cOutput cBrOutput cCase cPar1 cPar2 cComm1 cComm2 cBrMerge cBrComm1 cBrComm2 cBrClose cOpen cBrOpen cScope cBang*]:
fixes *cP* :: $('a, 'b, 'c)$ *psi*
and *cRs* :: $('a, 'b, 'c)$ *residual*
and *C* :: $'f$::*fs-name*
and *x* :: *name*

assumes $\Psi \rhd cP \longmapsto cRs$
and *rInput*: $\bigwedge M\ K\ xvec\ N\ Tvec\ P.\ [\![cP = M(\!|\lambda* xvec\ N|\!).P;\ cRs = K(\!|(N[xvec::=Tvec])|\!) \prec P[xvec::=Tvec];$
$\Psi \vdash M \leftrightarrow K;\ distinct\ xvec;\ set\ xvec \subseteq supp\ N;$
$length\ xvec = length\ Tvec;$
$xvec \sharp* Tvec;\ xvec \sharp* \Psi;\ xvec \sharp* M;\ xvec \sharp* K;$
$xvec \sharp* C]\!] \Longrightarrow Prop$
and *rBrInput*: $\bigwedge M\ K\ xvec\ N\ Tvec\ P.\ [\![cP = M(\!|\lambda* xvec\ N|\!).P;\ cRs = ¿K(\!|(N[xvec::=Tvec])|\!) \prec P[xvec::=Tvec];$
$\Psi \vdash K \succeq M;\ distinct\ xvec;\ set\ xvec \subseteq supp\ N;$
$length\ xvec = length\ Tvec;$
$xvec \sharp* Tvec;\ xvec \sharp* \Psi;\ xvec \sharp* M;\ xvec \sharp* K;$
$xvec \sharp* C]\!] \Longrightarrow Prop$
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 *rBrOutput*: $\bigwedge M\ K\ N\ P.\ [\![cP = M\langle N\rangle.P;\ cRs = ¡K\langle N\rangle \prec P;\ \Psi \vdash M \preceq K]\!] \Longrightarrow$ *Prop*
and *rCase*: $\bigwedge Cs\ P\ \varphi.\ [\![cP = Cases\ Cs;\ \Psi \rhd P \longmapsto cRs;\ (\varphi, P) \in set\ Cs;\ \Psi \vdash \varphi;\ guarded\ P]\!] \Longrightarrow$ *Prop*

and *rPar1*: $\bigwedge \Psi_Q\ P\ \alpha\ P'\ Q\ A_Q.\ [\![cP = P \parallel Q;\ cRs = \alpha \prec (P' \parallel Q);$
$(\Psi \otimes \Psi_Q) \rhd P \longmapsto (\alpha \prec P');\ extractFrame\ Q = \langle A_Q, \Psi_Q\rangle;\ distinct\ A_Q;$
$A_Q \sharp* P;\ A_Q \sharp* Q;\ A_Q \sharp* \Psi;\ A_Q \sharp* \alpha;\ A_Q \sharp* C;\ A_Q \sharp* P';\ bn\ \alpha \sharp* \Psi;\ bn\ \alpha \sharp* \Psi_Q;$
$bn\ \alpha \sharp* Q;\ bn\ \alpha \sharp* P;\ bn\ \alpha \sharp* subject\ \alpha;\ bn\ \alpha \sharp* C;\ distinct(bn\ \alpha)]\!] \Longrightarrow$
Prop
and *rPar2*: $\bigwedge \Psi_P\ Q\ \alpha\ Q'\ P\ A_P.\ [\![cP = P \parallel Q;\ cRs = \alpha \prec (P \parallel Q');$
$(\Psi \otimes \Psi_P) \rhd Q \longmapsto \alpha \prec Q';\ extractFrame\ P = \langle A_P, \Psi_P\rangle;\ distinct\ A_P;$
$A_P \sharp* P;\ A_P \sharp* Q;\ A_P \sharp* \Psi;\ A_P \sharp* \alpha;\ A_P \sharp* C;$
$A_P \sharp* Q';\ bn\ \alpha \sharp* \Psi;\ bn\ \alpha \sharp* \Psi_P;\ bn\ \alpha \sharp* P;\ bn\ \alpha \sharp* Q;\ bn\ \alpha \sharp* subject\ \alpha;\ bn\ \alpha \sharp* C;\ distinct(bn\ \alpha)]\!] \Longrightarrow Prop$
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*$\rrbracket \Longrightarrow$ *Prop*

and *rBrMerge*: $\bigwedge \Psi_Q\ P\ M\ N\ P'\ A_P\ \Psi_P\ Q\ Q'\ A_Q$.
$\llbracket cP = (P \parallel Q)$; $cRs = ¿M(\!|N|\!) \prec (P' \parallel Q')$;
$\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$; *extractFrame* $P = \langle A_P, \Psi_P \rangle$; *distinct* A_P;
$\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$; *extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$; *distinct* A_Q;
$A_P \sharp* \Psi$; $A_P \sharp* \Psi_Q$; $A_P \sharp* P$; $A_P \sharp* N$; $A_P \sharp* P'$;
$A_P \sharp* Q$; $A_P \sharp* Q'$; $A_P \sharp* M$; $A_P \sharp* A_Q$;
$A_Q \sharp* \Psi$; $A_Q \sharp* \Psi_P$; $A_Q \sharp* P$; $A_Q \sharp* N$; $A_Q \sharp* P'$;
$A_Q \sharp* Q$; $A_Q \sharp* Q'$; $A_Q \sharp* M$; $A_P \sharp* C$; $A_Q \sharp* C\rrbracket \Longrightarrow$ *Prop*

and *rBrComm1*: $\bigwedge \Psi_Q\ P\ M\ N\ P'\ A_P\ \Psi_P\ Q\ xvec\ Q'\ A_Q$.
$\llbracket cP = P \parallel Q$; $cRs = ¡M(\!|\nu * xvec|\!)\langle N \rangle \prec (P' \parallel Q')$;
$\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$; *extractFrame* $P = \langle A_P, \Psi_P \rangle$; *distinct* A_P;
$\Psi \otimes \Psi_P \rhd Q \longmapsto ¡M(\!|\nu * xvec|\!)\langle N \rangle \prec Q'$; *extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$; *distinct* A_Q;
$A_P \sharp* \Psi$; $A_P \sharp* \Psi_Q$; $A_P \sharp* P$; $A_P \sharp* N$;
$A_P \sharp* P'$; $A_P \sharp* Q$; $A_P \sharp* Q'$; $A_P \sharp* M$; $A_P \sharp* A_Q$; $A_P \sharp* xvec$; $A_Q \sharp* \Psi$; $A_Q \sharp* \Psi_P$;
$A_Q \sharp* P$; $A_Q \sharp* N$; $A_Q \sharp* P'$; $A_Q \sharp* M$; $A_Q \sharp* Q$; $A_Q \sharp* Q'$; $A_Q \sharp* xvec$;
$xvec \sharp* \Psi$; $xvec \sharp* \Psi_P$; $xvec \sharp* \Psi_Q$; $xvec \sharp* P$; $xvec \sharp* Q$; $xvec \sharp* M$;
$A_P \sharp* C$; $A_Q \sharp* C$; $xvec \sharp* C$; *distinct xvec*$\rrbracket \Longrightarrow$ *Prop*

and *rBrComm2*: $\bigwedge \Psi_Q\ P\ M\ xvec\ N\ P'\ A_P\ \Psi_P\ Q\ Q'\ A_Q$.
$\llbracket cP = P \parallel Q$; $cRs = ¡M(\!|\nu * xvec|\!)\langle N \rangle \prec (P' \parallel Q')$;
$\Psi \otimes \Psi_Q \rhd P \longmapsto ¡M(\!|\nu * xvec|\!)\langle N \rangle \prec P'$; *extractFrame* $P = \langle A_P, \Psi_P \rangle$; *distinct* A_P;
$\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$; *extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$; *distinct* A_Q;
$A_P \sharp* \Psi$; $A_P \sharp* \Psi_Q$; $A_P \sharp* P$; $A_P \sharp* N$;
$A_P \sharp* P'$; $A_P \sharp* Q$; $A_P \sharp* Q'$; $A_P \sharp* M$; $A_P \sharp* A_Q$; $A_P \sharp* xvec$; $A_Q \sharp* \Psi$; $A_Q \sharp* \Psi_P$;
$A_Q \sharp* P$; $A_Q \sharp* N$; $A_Q \sharp* P'$; $A_Q \sharp* M$; $A_Q \sharp* Q$; $A_Q \sharp* Q'$; $A_Q \sharp* xvec$;
$xvec \sharp* \Psi$; $xvec \sharp* \Psi_P$; $xvec \sharp* \Psi_Q$; $xvec \sharp* P$; $xvec \sharp* Q$; $xvec \sharp* M$;
$A_P \sharp* C$; $A_Q \sharp* C$; $xvec \sharp* C$; *distinct xvec*$\rrbracket \Longrightarrow$ *Prop*

and *rBrClose*: $\bigwedge P\ M\ xvec\ N\ P'\ x$.
$\llbracket cP = ((\!|\nu x|\!)P)$; $cRs = \tau \prec (\!|\nu x|\!)((\!|\nu * xvec|\!)P')$;
$x \in supp\ M$;
$\Psi \rhd P \longmapsto ¡M(\!|\nu * xvec|\!)\langle N \rangle \prec P'$;
distinct xvec; $xvec \sharp* \Psi$; $xvec \sharp* P$;
$xvec \sharp* M$;
$x \sharp \Psi$; $x \sharp xvec$;
$xvec \sharp* C$; $x \sharp C\rrbracket \Longrightarrow$ *Prop*

and *rOpen*: $\bigwedge P\ M\ xvec\ yvec\ N\ P'\ x$.
$\llbracket cP = (\!|\nu x|\!)P$; $cRs = M(\!|\nu * (xvec @ x \# yvec)|\!)\langle N \rangle \prec P'$;

$\Psi \rhd P \longmapsto M(\!|\nu*(xvec@yvec)|\!)\langle N\rangle \prec P'$; $x \in supp\ N$; $x \sharp xvec$; $x \sharp yvec$; $x \sharp M$; $x \sharp \Psi$; *distinct xvec*; *distinct yvec*;
$xvec \sharp* \Psi$; $xvec \sharp* P$; $xvec \sharp* M$; $xvec \sharp* yvec$; $yvec \sharp* \Psi$; $yvec \sharp* P$; $yvec \sharp* M$; $xvec \sharp* C$; $x \sharp C$; $yvec \sharp* C\rrbracket \Longrightarrow$
Prop

and *rBrOpen*: $\bigwedge P\ M\ xvec\ yvec\ N\ P'\ x.$
$\llbracket cP = (\!|\nu x|\!)P$; $cRs = ¡M(\!|\nu*(xvec@x\#yvec)|\!)\langle N\rangle \prec P'$;
$\Psi \rhd P \longmapsto ¡M(\!|\nu*(xvec@yvec)|\!)\langle N\rangle \prec P'$; $x \in supp\ N$; $x \sharp xvec$; $x \sharp yvec$; $x \sharp M$; $x \sharp \Psi$; *distinct xvec*; *distinct yvec*;
$xvec \sharp* \Psi$; $xvec \sharp* P$; $xvec \sharp* M$; $xvec \sharp* yvec$; $yvec \sharp* \Psi$; $yvec \sharp* P$; $yvec \sharp* M$; $xvec \sharp* C$; $x \sharp C$; $yvec \sharp* C\rrbracket \Longrightarrow$
Prop

and *rScope*: $\bigwedge P\ \alpha\ P'\ x.\ \llbracket cP = (\!|\nu x|\!)P$; $cRs = \alpha \prec (\!|\nu x|\!)P'$;
$\Psi \rhd P \longmapsto \alpha \prec P'$; $x \sharp \Psi$; $x \sharp \alpha$; $x \sharp C$; $bn\ \alpha \sharp* \Psi$; $bn\ \alpha \sharp* P$; $bn\ \alpha \sharp* subject\ \alpha$; $bn\ \alpha \sharp* C$; $distinct(bn\ \alpha)\rrbracket \Longrightarrow Prop$

and *rBang*: $\bigwedge P.\ \llbracket cP = !P$; $\Psi \rhd P \parallel !P \longmapsto cRs$; *guarded P*$\rrbracket \Longrightarrow Prop$

shows *Prop*

using ‹$\Psi \rhd cP \longmapsto cRs$›

proof(*cases rule*: *semantics.cases*)

case(*cInput M K xvec N Tvec P*)

obtain *p*::*name prm* **where** $(p \cdot xvec) \sharp* \Psi$ **and** $(p \cdot xvec) \sharp* M$ **and** $(p \cdot xvec) \sharp* N$ **and** $(p \cdot xvec) \sharp* K$
and $(p \cdot xvec) \sharp* Tvec$ **and** $(p \cdot xvec) \sharp* P$ **and** $(p \cdot xvec) \sharp* C$
and *S*: $(set\ p) \subseteq (set\ xvec) \times (set(p \cdot xvec))$ **and** *distinctPerm p*
by(*rule name-list-avoiding*[**where** *xvec*=*xvec* **and** *c*=(Ψ, *M*, *K*, *N*, *P*, *C*, *Tvec*)])
(*auto simp add*: *eqvts fresh-star-prod*)

from ‹$cP = M(\!|\lambda*xvec\ N|\!).P$› ‹$(p \cdot xvec) \sharp* N$› ‹$(p \cdot xvec) \sharp* P$› *S*

have $cP = M(\!|\lambda*(p \cdot xvec)\ (p \cdot N)|\!).(p \cdot P)$
by(*simp add*: *inputChainAlpha'*)

moreover from ‹$cRs = K(\!|(N[xvec::=Tvec])|\!) \prec P[xvec::=Tvec]$› ‹$(p \cdot xvec) \sharp* N$› ‹$(p \cdot xvec) \sharp* P$› *S* ‹*length xvec* = *length Tvec*› ‹*distinctPerm p*›

have $cRs = K(\!|((p \cdot N)[(p \cdot xvec)::=Tvec])|\!) \prec (p \cdot P)[(p \cdot xvec)::=Tvec]$
by(*auto simp add*: *substTerm.renaming renaming residualInject*)

moreover note ‹$\Psi \vdash M \leftrightarrow K$›

moreover from ‹*distinct xvec*› **have** $distinct(p \cdot xvec)$
by *simp*

moreover from ‹$(set\ xvec) \subseteq (supp\ N)$› **have** $(p \cdot (set\ xvec)) \subseteq (p \cdot (supp\ N))$
by *simp*

then have $set(p \cdot xvec) \subseteq supp(p \cdot N)$
by(*simp add*: *eqvts*)

moreover from ‹*length xvec* = *length Tvec*› **have** $length(p \cdot xvec) = length\ Tvec$
by *simp*

ultimately show *?thesis* **using** ‹$(p \cdot xvec) \sharp* Tvec$› ‹$(p \cdot xvec) \sharp* \Psi$› ‹$(p \cdot xvec) \sharp* M$› ‹$(p \cdot xvec) \sharp* K$›
‹$(p \cdot xvec) \sharp* C$›
by(*rule rInput*)

next

case(*cBrInput K M xvec N Tvec P*)
obtain *p*::*name prm* **where** $(p \cdot xvec) \sharp* \Psi$ **and** $(p \cdot xvec) \sharp* M$ **and** $(p \cdot xvec) \sharp* N$ **and** $(p \cdot xvec) \sharp* K$
and $(p \cdot xvec) \sharp* Tvec$ **and** $(p \cdot xvec) \sharp* P$ **and** $(p \cdot xvec) \sharp* C$
and *S*: $(set\ p) \subseteq (set\ xvec) \times (set(p \cdot xvec))$ **and** *distinctPerm p*
by(*rule name-list-avoiding*[**where** *xvec*=*xvec* **and** *c*=(Ψ, *M*, *K*, *N*, *P*, *C*, *Tvec*)])
(*auto simp add*: *eqvts fresh-star-prod*)
from ‹$cP = M\langle\!\langle \lambda* xvec\ N \rangle\!\rangle.P$› ‹$(p \cdot xvec) \sharp* N$› ‹$(p \cdot xvec) \sharp* P$› *S*
have $cP = M\langle\!\langle \lambda*(p \cdot xvec)\ (p \cdot N) \rangle\!\rangle.(p \cdot P)$
by(*simp add*: *inputChainAlpha'*)
moreover from ‹$cRs = ¿K\langle\!\langle (N[xvec::=Tvec]) \rangle\!\rangle \prec P[xvec::=Tvec]$› ‹$(p \cdot xvec) \sharp* N$› ‹$(p \cdot xvec) \sharp* P$› *S* ‹*length xvec* = *length Tvec*› ‹*distinctPerm p*›
have $cRs = ¿K\langle\!\langle ((p \cdot N)[(p \cdot xvec)::=Tvec]) \rangle\!\rangle \prec (p \cdot P)[(p \cdot xvec)::=Tvec]$
by(*auto simp add*: *substTerm.renaming renaming residualInject*)

moreover note ‹$\Psi \vdash K \succeq M$›
moreover from ‹*distinct xvec*› **have** $distinct(p \cdot xvec)$
by *simp*
moreover from ‹$(set\ xvec) \subseteq (supp\ N)$›
have $(p \cdot (set\ xvec)) \subseteq (p \cdot (supp\ N))$
by *simp*
then have $set(p \cdot xvec) \subseteq supp(p \cdot N)$
by(*simp add*: *eqvts*)
moreover from ‹*length xvec* = *length Tvec*› **have** $length(p \cdot xvec) = length\ Tvec$
by *simp*
ultimately show *?thesis* **using** ‹$(p \cdot xvec) \sharp* Tvec$› ‹$(p \cdot xvec) \sharp* \Psi$› ‹$(p \cdot xvec) \sharp* M$› ‹$(p \cdot xvec) \sharp* K$›
‹$(p \cdot xvec) \sharp* C$›
by(*simp add*: *rBrInput*)
next
case(*Output M K N P*)
then show *?thesis* **by**(*rule rOutput*)
next
case(*BrOutput M K N P*)
then show *?thesis* **by**(*rule rBrOutput*)
next
case(*Case P* φ *Cs*)
then show *?thesis* **by**(*rule rCase*)
next
case(*cPar1* Ψ_Q *P* α *P'* *Q* A_Q)
obtain *q*::*name prm* **where** $(bn(q \cdot \alpha)) \sharp* \Psi$ **and** $(bn(q \cdot \alpha)) \sharp* P$ **and** $(bn(q \cdot \alpha)) \sharp* Q$
and $(bn(q \cdot \alpha)) \sharp* \alpha$ **and** $(bn(q \cdot \alpha)) \sharp* A_Q$ **and** $(bn(q \cdot \alpha)) \sharp* P'$ **and** $(bn(q \cdot \alpha)) \sharp* \Psi_Q$
and *distinctPerm q*
and $(bn(q \cdot \alpha)) \sharp* C$ **and** *Sq*: $(set\ q) \subseteq set(bn\ \alpha) \times (set(bn(q \cdot \alpha)))$
by(*rule name-list-avoiding*[**where** *xvec*=*bn* α **and** *c*=(Ψ, *P*, *Q*, α, A_Q, Ψ_Q, *P'*, *C*)]) (*auto simp add*: *eqvts*)
obtain *p*::*name prm* **where** $(p \cdot A_Q) \sharp* \Psi$ **and** $(p \cdot A_Q) \sharp* P$ **and** $(p \cdot A_Q) \sharp*$

Q
and $(p \cdot A_Q) \sharp* \alpha$ **and** $(p \cdot A_Q) \sharp* (q \cdot \alpha)$ **and** $(p \cdot A_Q) \sharp* P'$
and $(p \cdot A_Q) \sharp* (q \cdot P')$ **and** $(p \cdot A_Q) \sharp* \Psi_Q$ **and** $(p \cdot A_Q) \sharp* C$
and *Sp*: $(set\ p) \subseteq (set\ A_Q) \times (set(p \cdot A_Q))$ **and** *distinctPerm p*
by(*rule name-list-avoiding*[**where** *xvec*=A_Q **and** c=(Ψ, P, Q, α, $q \cdot \alpha$, P', ($q \cdot P'$), Ψ_Q, C)]) *auto*
from ‹$A_Q \sharp* \alpha$› ‹$bn(q \cdot \alpha) \sharp* A_Q$› *Sq* ‹*distinctPerm q*› **have** $A_Q \sharp* (q \cdot \alpha)$
by(*subst fresh-star-bij*[*symmetric*, *of* - - q]) (*simp add*: *eqvts*)
from ‹$bn\ \alpha \sharp* subject\ \alpha$› ‹*distinctPerm q*› **have** $bn(q \cdot \alpha) \sharp* subject(q \cdot \alpha)$
by(*subst fresh-star-bij*[*symmetric*, *of* - - q]) (*simp add*: *eqvts*)
from ‹*distinct*($bn\ \alpha$)› ‹*distinctPerm q*› **have** *distinct*($bn(q \cdot \alpha)$)
by(*subst distinctClosed*[*symmetric*, *of* - q]) (*simp add*: *eqvts*)
note ‹$cP = P \parallel Q$›

moreover from ‹$cRs = \alpha \prec (P' \parallel Q)$› ‹$bn\ \alpha \sharp* subject\ \alpha$› ‹$(bn(q \cdot \alpha)) \sharp* \alpha$› ‹$(bn(q \cdot \alpha)) \sharp* P'$› ‹$(bn(q \cdot \alpha)) \sharp* Q$› ‹$bn\ \alpha \sharp* Q$› *Sq*
have $cRs = (q \cdot \alpha) \prec (q \cdot P') \parallel Q$
by(*force simp add*: *residualAlpha*)
moreover from ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto \alpha \prec P'$› ‹$bn\ \alpha \sharp* subject\ \alpha$› ‹$(bn(q \cdot \alpha)) \sharp* \alpha$› ‹$(bn(q \cdot \alpha)) \sharp* P'$› *Sq*
have *Trans*: $\Psi \otimes \Psi_Q \rhd P \longmapsto (q \cdot \alpha) \prec (q \cdot P')$
by(*force simp add*: *residualAlpha*)
then have $A_Q \sharp* (q \cdot P')$ **using** ‹$bn(q \cdot \alpha) \sharp* subject(q \cdot \alpha)$› ‹*distinct*($bn(q \cdot \alpha)$)› ‹$A_Q \sharp* P$› ‹$A_Q \sharp* (q \cdot \alpha)$›
by(*force dest*: *freeFreshChainDerivative*)

from *Trans* **have** $(p \cdot (\Psi \otimes \Psi_Q)) \rhd (p \cdot P) \longmapsto p \cdot ((q \cdot \alpha) \prec (q \cdot P'))$
by(*rule semantics.eqvt*)
with ‹$A_Q \sharp* \Psi$› ‹$A_Q \sharp* P$› ‹$A_Q \sharp* (q \cdot \alpha)$› ‹$A_Q \sharp* (q \cdot P')$› ‹$(bn(q \cdot \alpha)) \sharp* A_Q$› ‹$(p \cdot A_Q) \sharp* (q \cdot \alpha)$›
‹$(p \cdot A_Q) \sharp* \Psi$› ‹$(p \cdot A_Q) \sharp* P$› ‹$(p \cdot A_Q) \sharp* (q \cdot P')$› *Sp*
have $\Psi \otimes (p \cdot \Psi_Q) \rhd P \longmapsto (q \cdot \alpha) \prec (q \cdot P')$ **by**(*simp add*: *eqvts*)
moreover from ‹*extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$› ‹$(p \cdot A_Q) \sharp* \Psi_Q$› *Sp* **have** *extractFrame* $Q = \langle (p \cdot A_Q), (p \cdot \Psi_Q) \rangle$
by(*simp add*: *frameChainAlpha'* *eqvts*)
moreover from ‹$(bn(q \cdot \alpha)) \sharp* \Psi_Q$› ‹$(bn(q \cdot \alpha)) \sharp* A_Q$› ‹$(p \cdot A_Q) \sharp* (q \cdot \alpha)$› *Sp* **have** $(bn(q \cdot \alpha)) \sharp* (p \cdot \Psi_Q)$
by(*simp add*: *freshAlphaPerm*)
moreover from ‹*distinct* A_Q› **have** *distinct*($p \cdot A_Q$) **by** *simp*
ultimately show *?thesis*
using ‹$(p \cdot A_Q) \sharp* P$› ‹$(p \cdot A_Q) \sharp* Q$› ‹$(p \cdot A_Q) \sharp* \Psi$› ‹$(p \cdot A_Q) \sharp* (q \cdot \alpha)$›
‹$(p \cdot A_Q) \sharp* (q \cdot P')$› ‹$(bn(q \cdot \alpha)) \sharp* \Psi$› ‹$(bn(q \cdot \alpha)) \sharp* Q$› ‹$(bn(q \cdot \alpha)) \sharp* P$›
‹$(bn(q \cdot \alpha)) \sharp* C$› ‹$(p \cdot A_Q) \sharp* C$› ‹$bn\ (q \cdot \alpha) \sharp* subject\ (q \cdot \alpha)$› ‹*distinct*($bn(q \cdot \alpha)$)›
by(*metis rPar1*)
next
case(*cPar2* Ψ_P Q α Q' P A_P)
obtain q::*name prm* **where** $(bn(q \cdot \alpha)) \sharp* \Psi$ **and** $(bn(q \cdot \alpha)) \sharp* P$ **and** $(bn(q \cdot \alpha)) \sharp* Q$

and $(bn(q \cdot \alpha)) \sharp* \alpha$ **and** $(bn(q \cdot \alpha)) \sharp* A_P$ **and** $(bn(q \cdot \alpha)) \sharp* Q'$ **and** $(bn(q \cdot \alpha)) \sharp* \Psi_P$
and *distinctPerm q*
and $(bn(q \cdot \alpha)) \sharp* C$ **and** *Sq*: $(set\ q) \subseteq set(bn\ \alpha) \times (set(bn(q \cdot \alpha)))$
by (*rule name-list-avoiding*[**where** *xvec*=$bn\ \alpha$ **and** *c*=$(\Psi, P, Q, \alpha, A_P, \Psi_P, Q', C)$]) (*auto simp add*: *eqvts*)
obtain *p*::*name prm* **where** $(p \cdot A_P) \sharp* \Psi$ **and** $(p \cdot A_P) \sharp* P$ **and** $(p \cdot A_P) \sharp* Q$
and $(p \cdot A_P) \sharp* \alpha$ **and** $(p \cdot A_P) \sharp* (q \cdot \alpha)$ **and** $(p \cdot A_P) \sharp* Q'$
and $(p \cdot A_P) \sharp* (q \cdot Q')$ **and** $(p \cdot A_P) \sharp* \Psi_P$ **and** $(p \cdot A_P) \sharp* C$
and *Sp*: $(set\ p) \subseteq (set\ A_P) \times (set(p \cdot A_P))$ **and** *distinctPerm p*
by(*rule name-list-avoiding*[**where** *xvec*=A_P **and** *c*=$(\Psi, P, Q, \alpha, q \cdot \alpha, Q', (q \cdot Q'), \Psi_P, C)$]) *auto*
from ‹$A_P \sharp* \alpha$› ‹$bn(q \cdot \alpha) \sharp* A_P$› *Sq* ‹*distinctPerm q*› **have** $A_P \sharp* (q \cdot \alpha)$
by(*subst fresh-star-bij*[*symmetric*, *of* - - *q*]) (*simp add*: *eqvts*)
from ‹$bn\ \alpha \sharp* subject\ \alpha$› ‹*distinctPerm q*› **have** $bn(q \cdot \alpha) \sharp* subject(q \cdot \alpha)$
by(*subst fresh-star-bij*[*symmetric*, *of* - - *q*]) (*simp add*: *eqvts*)
from ‹$distinct(bn\ \alpha)$› ‹*distinctPerm q*› **have** $distinct(bn(q \cdot \alpha))$
by(*subst distinctClosed*[*symmetric*, *of* - *q*]) (*simp add*: *eqvts*)
note ‹$cP = P \parallel Q$›

moreover from ‹$cRs = \alpha \prec (P \parallel Q')$› ‹$bn\ \alpha \sharp* subject\ \alpha$› ‹$(bn(q \cdot \alpha)) \sharp* \alpha$› ‹$(bn(q \cdot \alpha)) \sharp* Q'$› ‹$(bn(q \cdot \alpha)) \sharp* P$› ‹$bn\ \alpha \sharp* P$› *Sq*
have $cRs = (q \cdot \alpha) \prec P \parallel (q \cdot Q')$
by(*force simp add*: *residualAlpha*)
moreover from ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto \alpha \prec Q'$› ‹$bn\ \alpha \sharp* subject\ \alpha$› ‹$(bn(q \cdot \alpha)) \sharp* \alpha$› ‹$(bn(q \cdot \alpha)) \sharp* Q'$› *Sq*
have *Trans*: $\Psi \otimes \Psi_P \rhd Q \longmapsto (q \cdot \alpha) \prec (q \cdot Q')$
by(*force simp add*: *residualAlpha*)
then have $A_P \sharp* (q \cdot Q')$ **using** ‹$bn(q \cdot \alpha) \sharp* subject(q \cdot \alpha)$› ‹$distinct(bn(q \cdot \alpha))$› ‹$A_P \sharp* Q$› ‹$A_P \sharp* (q \cdot \alpha)$›
by(*auto dest*: *freeFreshChainDerivative*)

from *Trans* **have** $(p \cdot (\Psi \otimes \Psi_P)) \rhd (p \cdot Q) \longmapsto p \cdot ((q \cdot \alpha) \prec (q \cdot Q'))$
by(*rule semantics.eqvt*)
with ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* Q$› ‹$A_P \sharp* (q \cdot \alpha)$› ‹$A_P \sharp* (q \cdot Q')$› ‹$(bn(q \cdot \alpha)) \sharp* A_P$› ‹$(p \cdot A_P) \sharp* (q \cdot \alpha)$›
‹$(p \cdot A_P) \sharp* \Psi$› ‹$(p \cdot A_P) \sharp* Q$› ‹$(p \cdot A_P) \sharp* (q \cdot Q')$› *Sp*
have $\Psi \otimes (p \cdot \Psi_P) \rhd Q \longmapsto (q \cdot \alpha) \prec (q \cdot Q')$ **by**(*simp add*: *eqvts*)
moreover from ‹$extractFrame\ P = \langle A_P, \Psi_P \rangle$› ‹$(p \cdot A_P) \sharp* \Psi_P$› *Sp* **have** $extractFrame\ P = \langle (p \cdot A_P), (p \cdot \Psi_P) \rangle$
by(*simp add*: *frameChainAlpha' eqvts*)
moreover from ‹$(bn(q \cdot \alpha)) \sharp* \Psi_P$› ‹$(bn(q \cdot \alpha)) \sharp* A_P$› ‹$(p \cdot A_P) \sharp* (q \cdot \alpha)$› *Sp* **have** $(bn(q \cdot \alpha)) \sharp* (p \cdot \Psi_P)$
by(*simp add*: *freshAlphaPerm*)
moreover from ‹*distinct* A_P› **have** $distinct(p \cdot A_P)$ **by** *simp*
ultimately show *?thesis*
using ‹$(p \cdot A_P) \sharp* P$› ‹$(p \cdot A_P) \sharp* Q$› ‹$(p \cdot A_P) \sharp* \Psi$› ‹$(p \cdot A_P) \sharp* (q \cdot \alpha)$›
‹$(p \cdot A_P) \sharp* (q \cdot Q')$› ‹$(bn(q \cdot \alpha)) \sharp* \Psi$› ‹$(bn(q \cdot \alpha)) \sharp* Q$› ‹$(bn(q \cdot \alpha)) \sharp* P$›

‹(*bn*(*q* ∙ α)) ♯* *C*› ‹(*p* ∙ *A*$_P$) ♯* *C*› ‹*bn* (*q* ∙ α) ♯* *subject* (*q* ∙ α)› ‹*distinct*(*bn*(*q* ∙ α))›
by(*metis rPar2*)
next
case(*cComm1* Ψ$_Q$ *P M N P′ A*$_P$ Ψ$_P$ *Q K xvec Q′ A*$_Q$)
obtain *r*::*name prm* **where** (*r* ∙ *xvec*) ♯* Ψ **and** (*r* ∙ *xvec*) ♯* *P* **and** (*r* ∙ *xvec*) ♯* *Q* **and** (*r* ∙ *xvec*) ♯* *M*
and (*r* ∙ *xvec*) ♯* *K* **and** (*r* ∙ *xvec*) ♯* *N* **and** (*r* ∙ *xvec*) ♯* *A*$_P$ **and** (*r* ∙ *xvec*) ♯* *A*$_Q$
and (*r* ∙ *xvec*) ♯* *P′* **and** (*r* ∙ *xvec*) ♯* *Q′* **and** (*r* ∙ *xvec*) ♯* Ψ$_P$ **and** (*r* ∙ *xvec*) ♯* Ψ$_Q$
and (*r* ∙ *xvec*) ♯* *C* **and** *Sr*: (*set r*) ⊆ (*set xvec*) × (*set*(*r* ∙ *xvec*)) **and** *distinctPerm r*
by(*rule name-list-avoiding*[**where** *xvec*=*xvec* **and** *c*=(Ψ, *P*, *Q*, *M*, *K*, *N*, *A*$_P$, *A*$_Q$, Ψ$_P$, Ψ$_Q$, *P′*, *Q′*, *C*)])
(*auto simp add*: *eqvts*)

obtain *q*::*name prm* **where** (*q* ∙ *A*$_Q$) ♯* Ψ **and** (*q* ∙ *A*$_Q$) ♯* *P* **and** (*q* ∙ *A*$_Q$) ♯* *Q* **and** (*q* ∙ *A*$_Q$) ♯* *K*
and (*q* ∙ *A*$_Q$) ♯* *N* **and** (*q* ∙ *A*$_Q$) ♯* *xvec* **and** (*q* ∙ *A*$_Q$) ♯* *Q′* **and** (*q* ∙ *A*$_Q$) ♯* *P′*
and (*q* ∙ *A*$_Q$) ♯* Ψ$_P$ **and** (*q* ∙ *A*$_Q$) ♯* *A*$_P$ **and** (*q* ∙ *A*$_Q$) ♯* Ψ$_Q$ **and** (*q* ∙ *A*$_Q$) ♯* (*r* ∙ *xvec*)
and (*q* ∙ *A*$_Q$) ♯* *C* **and** *Sq*: (*set q*) ⊆ (*set A*$_Q$) × (*set*(*q* ∙ *A*$_Q$))
by(*rule name-list-avoiding*[**where** *xvec*=*A*$_Q$ **and** *c*=(Ψ, *P*, *Q*, *K*, *N*, *xvec*, *r* ∙ *xvec*, Ψ$_Q$, *A*$_P$, Ψ$_P$, *Q′*, *P′*, *C*)]) *clarsimp*

obtain *p*::*name prm* **where** (*p* ∙ *A*$_P$) ♯* Ψ **and** (*p* ∙ *A*$_P$) ♯* *P* **and** (*p* ∙ *A*$_P$) ♯* *Q* **and** (*p* ∙ *A*$_P$) ♯* *M*
and (*p* ∙ *A*$_P$) ♯* *N* **and** (*p* ∙ *A*$_P$) ♯* *xvec* **and** (*p* ∙ *A*$_P$) ♯* *Q′* **and** (*p* ∙ *A*$_P$) ♯* *A*$_Q$
and (*p* ∙ *A*$_P$) ♯* *P′* **and** (*p* ∙ *A*$_P$) ♯* Ψ$_P$ **and** (*p* ∙ *A*$_P$) ♯* Ψ$_Q$ **and** (*p* ∙ *A*$_P$) ♯* (*q* ∙ *A*$_Q$)
and (*p* ∙ *A*$_P$) ♯* *C* **and** (*p* ∙ *A*$_P$) ♯* (*r* ∙ *xvec*) **and** *Sp*: (*set p*) ⊆ (*set A*$_P$) × (*set*(*p* ∙ *A*$_P$))
by(*rule name-list-avoiding*[**where** *xvec*=*A*$_P$ **and** *c*=(Ψ, *P*, *Q*, *M*, *N*, *xvec*, *r* ∙ *xvec*, *A*$_Q$, *q* ∙ *A*$_Q$, Ψ$_Q$, Ψ$_P$, *Q′*, *P′*, *C*)])
(*auto simp add*: *eqvts fresh-star-prod*)

have *FrP*: *extractFrame P* = ⟨*A*$_P$, Ψ$_P$⟩ **by** *fact*
have *FrQ*: *extractFrame Q* = ⟨*A*$_Q$, Ψ$_Q$⟩ **by** *fact*

from ‹*A*$_P$ ♯* *Q*› *FrQ* ‹*A*$_P$ ♯* *A*$_Q$› **have** *A*$_P$ ♯* Ψ$_Q$
by(*auto dest*: *extractFrameFreshChain*)
from ‹*A*$_Q$ ♯* *P*› *FrP* ‹*A*$_P$ ♯* *A*$_Q$› **have** *A*$_Q$ ♯* Ψ$_P$
by(*auto dest*: *extractFrameFreshChain*)
note ‹*cP* = *P* ∥ *Q*›
moreover from ‹(*r* ∙ *xvec*) ♯* *P′*› ‹(*r* ∙ *xvec*) ♯* *Q′*› **have** (*r* ∙ *xvec*) ♯* (*P′* ∥ *Q′*)

by *simp*
with ‹$cRs = \tau \prec (\!|\nu * xvec|\!)(P' \parallel Q')$› ‹$(r \cdot xvec) \sharp* N$› *Sr*
have $cRs = \tau \prec (\!|\nu * (r \cdot xvec)|\!)(r \cdot (P' \parallel Q'))$ **by**(*simp add*: *resChainAlpha residualInject*)
then have $cRs = \tau \prec (\!|\nu * (r \cdot xvec)|\!)((r \cdot P') \parallel (r \cdot Q'))$ **by** *simp*

moreover from ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|N|\!) \prec P'$› *Sr* ‹*distinctPerm r*› ‹$xvec \sharp* P$› ‹$(r \cdot xvec) \sharp* P$›
have $\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|(r \cdot N)|\!) \prec (r \cdot P')$
by(*rule inputAlpha*)
then have $(q \cdot (\Psi \otimes \Psi_Q)) \rhd P \longmapsto (q \cdot M)(\!|(r \cdot N)|\!) \prec (r \cdot P')$ **using** *Sq* ‹$A_Q \sharp* P$› ‹$(q \cdot A_Q) \sharp* P$›
by − (*rule inputPermFrameSubject*, (*assumption* | *simp*)+)
then have *PTrans*: $\Psi \otimes (q \cdot \Psi_Q) \rhd P \longmapsto (q \cdot M)(\!|(r \cdot N)|\!) \prec (r \cdot P')$ **using** *Sq* ‹$A_Q \sharp* \Psi$› ‹$(q \cdot A_Q) \sharp* \Psi$›
by(*simp add*: *eqvts*)

moreover from ‹*extractFrame* $P = \langle A_P, \Psi_P \rangle$› *Sp* ‹$(p \cdot A_P) \sharp* \Psi_P$›
have *FrP*: *extractFrame* $P = \langle (p \cdot A_P), (p \cdot \Psi_P) \rangle$
by(*simp add*: *frameChainAlpha*)
moreover from ‹*distinct* A_P› **have** *distinct*$(p \cdot A_P)$ **by** *simp*

moreover from ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|\nu * xvec|\!)\langle N \rangle \prec Q'$› *Sr* ‹$(r \cdot xvec) \sharp* N$› ‹$(r \cdot xvec) \sharp* Q'$›
have $\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|\nu * (r \cdot xvec)|\!)\langle (r \cdot N) \rangle \prec (r \cdot Q')$
by(*simp add*: *boundOutputChainAlpha''* *residualInject*)
then have $(p \cdot (\Psi \otimes \Psi_P)) \rhd Q \longmapsto (p \cdot K)(\!|\nu * (r \cdot xvec)|\!)\langle (r \cdot N) \rangle \prec (r \cdot Q')$ **using** *Sp* ‹$A_P \sharp* Q$› ‹$(p \cdot A_P) \sharp* Q$› ‹$(r \cdot xvec) \sharp* K$›‹$(p \cdot A_P) \sharp* (r \cdot xvec)$› ‹$(r \cdot xvec) \sharp* A_P$›
by(*fastforce intro*: *outputPermFrameSubject*)
then have *QTrans*: $\Psi \otimes (p \cdot \Psi_P) \rhd Q \longmapsto (p \cdot K)(\!|\nu * (r \cdot xvec)|\!)\langle (r \cdot N) \rangle \prec (r \cdot Q')$ **using** *Sp* ‹$A_P \sharp* \Psi$› ‹$(p \cdot A_P) \sharp* \Psi$›
by(*simp add*: *eqvts*)

moreover from ‹*extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$› *Sq* ‹$(q \cdot A_Q) \sharp* \Psi_Q$›
have *FrQ*: *extractFrame* $Q = \langle (q \cdot A_Q), (q \cdot \Psi_Q) \rangle$
by(*simp add*: *frameChainAlpha*)
moreover from ‹*distinct* A_Q› **have** *distinct*$(q \cdot A_Q)$ **by** *simp*

moreover from ‹$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K$› **have** $(p \cdot q \cdot (\Psi \otimes \Psi_P \otimes \Psi_Q)) \vdash (p \cdot q \cdot M) \leftrightarrow (p \cdot q \cdot K)$
by(*metis chanEqClosed*)
with ‹$A_P \sharp* \Psi$› ‹$(p \cdot A_P) \sharp* \Psi$› ‹$A_Q \sharp* \Psi$› ‹$(q \cdot A_Q) \sharp* \Psi$› ‹$A_Q \sharp* \Psi_P$› ‹$(q \cdot A_Q) \sharp* \Psi_P$›
‹$A_P \sharp* \Psi_Q$› ‹$(p \cdot A_P) \sharp* \Psi_Q$› ‹$A_P \sharp* M$› ‹$(p \cdot A_P) \sharp* M$› ‹$(q \cdot A_Q) \sharp* A_P$› ‹$(p \cdot A_P) \sharp* (q \cdot A_Q)$›
‹$A_Q \sharp* K$› ‹$(q \cdot A_Q) \sharp* K$› ‹$A_P \sharp* A_Q$› ‹$(p \cdot A_P) \sharp* A_Q$› *Sp Sq*
have $\Psi \otimes (p \cdot \Psi_P) \otimes (q \cdot \Psi_Q) \vdash (q \cdot M) \leftrightarrow (p \cdot K)$
by(*simp add*: *eqvts freshChainSimps*)

moreover note $\langle (p \cdot A_P) \sharp* \Psi \rangle$
moreover from $\langle (p \cdot A_P) \sharp* A_Q \rangle$ $\langle (p \cdot A_P) \sharp* (q \cdot A_Q) \rangle$ $\langle (p \cdot A_P) \sharp* \Psi_Q \rangle$ *Sq* **have** $(p \cdot A_P) \sharp* (q \cdot \Psi_Q)$
by(*simp add*: *freshChainSimps*)
moreover note $\langle (p \cdot A_P) \sharp* P \rangle$
moreover from $\langle (p \cdot A_P) \sharp* A_Q \rangle$ $\langle (p \cdot A_P) \sharp* (q \cdot A_Q) \rangle$ $\langle (p \cdot A_P) \sharp* M \rangle$ *Sq* **have** $(p \cdot A_P) \sharp* (q \cdot M)$
by(*simp add*: *freshChainSimps*)
moreover from $\langle (p \cdot A_P) \sharp* xvec \rangle$ $\langle (p \cdot A_P) \sharp* (r \cdot xvec) \rangle$ $\langle (p \cdot A_P) \sharp* N \rangle$ *Sr* **have** $(p \cdot A_P) \sharp* (r \cdot N)$
by(*simp add*: *freshChainSimps*)
moreover from $\langle (p \cdot A_P) \sharp* xvec \rangle$ $\langle (p \cdot A_P) \sharp* (r \cdot xvec) \rangle$ $\langle (p \cdot A_P) \sharp* P' \rangle$ *Sr* **have** $(p \cdot A_P) \sharp* (r \cdot P')$
by(*simp add*: *freshChainSimps*)
moreover note $\langle (p \cdot A_P) \sharp* Q \rangle$
moreover from $\langle (p \cdot A_P) \sharp* xvec \rangle$ $\langle (p \cdot A_P) \sharp* (r \cdot xvec) \rangle$ $\langle (p \cdot A_P) \sharp* Q' \rangle$ *Sr* **have** $(p \cdot A_P) \sharp* (r \cdot Q')$
by(*simp add*: *freshChainSimps*)
moreover note $\langle (p \cdot A_P) \sharp* (q \cdot A_Q) \rangle$ $\langle (p \cdot A_P) \sharp* (r \cdot xvec) \rangle$ $\langle (q \cdot A_Q) \sharp* \Psi \rangle$
moreover from $\langle (q \cdot A_Q) \sharp* A_P \rangle$ $\langle (p \cdot A_P) \sharp* A_Q \rangle$ $\langle (q \cdot A_Q) \sharp* \Psi_P \rangle$ $\langle (p \cdot A_P) \sharp* (q \cdot A_Q) \rangle$ *Sp Sq* **have** $(q \cdot A_Q) \sharp* (p \cdot \Psi_P)$
by(*simp add*: *freshChainSimps*)
moreover note $\langle (q \cdot A_Q) \sharp* P \rangle$
moreover from $\langle (q \cdot A_Q) \sharp* A_P \rangle$ $\langle (p \cdot A_P) \sharp* A_Q \rangle$ $\langle (q \cdot A_Q) \sharp* K \rangle$ $\langle (p \cdot A_P) \sharp* (q \cdot A_Q) \rangle$ *Sp Sq* **have** $(q \cdot A_Q) \sharp* (p \cdot K)$
by(*simp add*: *freshChainSimps*)
moreover from $\langle (q \cdot A_Q) \sharp* xvec \rangle$ $\langle (q \cdot A_Q) \sharp* (r \cdot xvec) \rangle$ $\langle (q \cdot A_Q) \sharp* N \rangle$ *Sr* **have** $(q \cdot A_Q) \sharp* (r \cdot N)$
by(*simp add*: *freshChainSimps*)
moreover from $\langle (q \cdot A_Q) \sharp* xvec \rangle$ $\langle (q \cdot A_Q) \sharp* (r \cdot xvec) \rangle$ $\langle (q \cdot A_Q) \sharp* P' \rangle$ *Sr* **have** $(q \cdot A_Q) \sharp* (r \cdot P')$
by(*simp add*: *freshChainSimps*)
moreover note $\langle (q \cdot A_Q) \sharp* Q \rangle$
moreover from $\langle (q \cdot A_Q) \sharp* xvec \rangle$ $\langle (q \cdot A_Q) \sharp* (r \cdot xvec) \rangle$ $\langle (q \cdot A_Q) \sharp* Q' \rangle$ *Sr* **have** $(q \cdot A_Q) \sharp* (r \cdot Q')$
by(*simp add*: *freshChainSimps*)
moreover note $\langle (q \cdot A_Q) \sharp* (r \cdot xvec) \rangle$ $\langle (r \cdot xvec) \sharp* \Psi \rangle$
moreover from $\langle (r \cdot xvec) \sharp* A_P \rangle$ $\langle (p \cdot A_P) \sharp* (r \cdot xvec) \rangle$ $\langle (r \cdot xvec) \sharp* \Psi_P \rangle$ *Sp* **have** $(r \cdot xvec) \sharp* (p \cdot \Psi_P)$
by(*simp add*: *freshChainSimps*)
moreover from $\langle (r \cdot xvec) \sharp* A_Q \rangle$ $\langle (q \cdot A_Q) \sharp* (r \cdot xvec) \rangle$ $\langle (r \cdot xvec) \sharp* \Psi_Q \rangle$ *Sq* **have** $(r \cdot xvec) \sharp* (q \cdot \Psi_Q)$
by(*simp add*: *freshChainSimps*)
moreover note $\langle (r \cdot xvec) \sharp* P \rangle$
moreover from $\langle (r \cdot xvec) \sharp* A_Q \rangle$ $\langle (q \cdot A_Q) \sharp* (r \cdot xvec) \rangle$ $\langle (r \cdot xvec) \sharp* M \rangle$ *Sq* **have** $(r \cdot xvec) \sharp* (q \cdot M)$
by(*simp add*: *freshChainSimps*)
moreover note $\langle (r \cdot xvec) \sharp* Q \rangle$
moreover from $\langle (r \cdot xvec) \sharp* A_P \rangle$ $\langle (p \cdot A_P) \sharp* (r \cdot xvec) \rangle$ $\langle (r \cdot xvec) \sharp* K \rangle$ *Sp*

have $(r \cdot xvec) \sharp* (p \cdot K)$
by(*simp add*: *freshChainSimps*)
moreover note $\langle (p \cdot A_P) \sharp* C\rangle$ $\langle (q \cdot A_Q) \sharp* C\rangle$ $\langle (r \cdot xvec) \sharp* C\rangle$
moreover from ‹*distinct xvec*› **have** *distinct*$(r \cdot xvec)$ **by** *simp*
ultimately show *?thesis* **by**(*rule rComm1*)
next
case(*cComm2* Ψ_Q P M *xvec* N P' A_P Ψ_P Q K Q' A_Q)
obtain r::*name prm* **where** $(r \cdot xvec) \sharp* \Psi$ **and** $(r \cdot xvec) \sharp* P$ **and** $(r \cdot xvec) \sharp* Q$ **and** $(r \cdot xvec) \sharp* M$
and $(r \cdot xvec) \sharp* K$ **and** $(r \cdot xvec) \sharp* N$ **and** $(r \cdot xvec) \sharp* A_P$ **and** $(r \cdot xvec) \sharp* A_Q$
and $(r \cdot xvec) \sharp* P'$ **and** $(r \cdot xvec) \sharp* Q'$ **and** $(r \cdot xvec) \sharp* \Psi_P$ **and** $(r \cdot xvec) \sharp* \Psi_Q$
and $(r \cdot xvec) \sharp* C$ **and** *Sr*: $(set\ r) \subseteq (set\ xvec) \times (set(r \cdot xvec))$ **and** *distinctPerm r*
by(*rule name-list-avoiding*[**where** *xvec*=*xvec* **and** *c*=(Ψ, P, Q, M, K, N, A_P, A_Q, Ψ_P, Ψ_Q, P', Q', C)])
(*auto simp add*: *eqvts*)

obtain q::*name prm* **where** $(q \cdot A_Q) \sharp* \Psi$ **and** $(q \cdot A_Q) \sharp* P$ **and** $(q \cdot A_Q) \sharp* Q$ **and** $(q \cdot A_Q) \sharp* K$
and $(q \cdot A_Q) \sharp* N$ **and** $(q \cdot A_Q) \sharp* xvec$ **and** $(q \cdot A_Q) \sharp* Q'$ **and** $(q \cdot A_Q) \sharp* P'$
and $(q \cdot A_Q) \sharp* \Psi_P$ **and** $(q \cdot A_Q) \sharp* A_P$ **and** $(q \cdot A_Q) \sharp* \Psi_Q$ **and** $(q \cdot A_Q) \sharp* (r \cdot xvec)$
and $(q \cdot A_Q) \sharp* C$ **and** *Sq*: $(set\ q) \subseteq (set\ A_Q) \times (set(q \cdot A_Q))$
by(*rule name-list-avoiding*[**where** *xvec*=A_Q **and** *c*=(Ψ, P, Q, K, N, *xvec*, $r \cdot xvec$, Ψ_Q, A_P, Ψ_P, Q', P', C)]) *clarsimp*

obtain p::*name prm* **where** $(p \cdot A_P) \sharp* \Psi$ **and** $(p \cdot A_P) \sharp* P$ **and** $(p \cdot A_P) \sharp* Q$ **and** $(p \cdot A_P) \sharp* M$
and $(p \cdot A_P) \sharp* N$ **and** $(p \cdot A_P) \sharp* xvec$ **and** $(p \cdot A_P) \sharp* Q'$ **and** $(p \cdot A_P) \sharp* A_Q$
and $(p \cdot A_P) \sharp* P'$ **and** $(p \cdot A_P) \sharp* \Psi_P$ **and** $(p \cdot A_P) \sharp* \Psi_Q$ **and** $(p \cdot A_P) \sharp* (q \cdot A_Q)$
and $(p \cdot A_P) \sharp* C$ **and** $(p \cdot A_P) \sharp* (r \cdot xvec)$ **and** *Sp*: $(set\ p) \subseteq (set\ A_P) \times (set(p \cdot A_P))$
by(*rule name-list-avoiding*[**where** *xvec*=A_P **and** *c*=(Ψ, P, Q, M, N, *xvec*, $r \cdot xvec$, A_Q, $q \cdot A_Q$, Ψ_Q, Ψ_P, Q', P', C)])
(*auto simp add*: *eqvts fresh-star-prod*)

have *FrP*: *extractFrame* $P = \langle A_P, \Psi_P\rangle$ **by** *fact*
have *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$ **by** *fact*

from $\langle A_P \sharp* Q\rangle$ *FrQ* $\langle A_P \sharp* A_Q\rangle$ **have** $A_P \sharp* \Psi_Q$
by(*auto dest*: *extractFrameFreshChain*)
from $\langle A_Q \sharp* P\rangle$ *FrP* $\langle A_P \sharp* A_Q\rangle$ **have** $A_Q \sharp* \Psi_P$
by(*auto dest*: *extractFrameFreshChain*)

note ‹$cP = P \parallel Q$›
moreover from ‹$(r \cdot xvec) \sharp* P'$› ‹$(r \cdot xvec) \sharp* Q'$› **have** $(r \cdot xvec) \sharp* (P' \parallel Q')$
by *simp*
with ‹$cRs = \tau \prec (\!|\nu* xvec|\!)(P' \parallel Q')$› ‹$(r \cdot xvec) \sharp* N$› *Sr*
have $cRs = \tau \prec (\!|\nu*(r \cdot xvec)|\!)(r \cdot (P' \parallel Q'))$ **by**(*simp add*: *resChainAlpha residualInject*)
then have $cRs = \tau \prec (\!|\nu*(r \cdot xvec)|\!)((r \cdot P') \parallel (r \cdot Q'))$
by *simp*

moreover from ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|\nu* xvec|\!)\langle N\rangle \prec P'$› *Sr* ‹$(r \cdot xvec) \sharp* N$› ‹$(r \cdot xvec) \sharp* P'$›
have $\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|\nu*(r \cdot xvec)|\!)\langle(r \cdot N)\rangle \prec (r \cdot P')$ **by**(*simp add*: *boundOutputChainAlpha''* *residualInject*)
then have $(q \cdot (\Psi \otimes \Psi_Q)) \rhd P \longmapsto (q \cdot M)(\!|\nu*(r \cdot xvec)|\!)\langle(r \cdot N)\rangle \prec (r \cdot P')$ **using** *Sq* ‹$A_Q \sharp* P$› ‹$(q \cdot A_Q) \sharp* P$› ‹$(r \cdot xvec) \sharp* M$› ‹$(r \cdot xvec) \sharp* A_Q$› ‹$(q \cdot A_Q) \sharp* (r \cdot xvec)$›
by(*fastforce intro*: *outputPermFrameSubject*)
then have *PTrans*: $\Psi \otimes (q \cdot \Psi_Q) \rhd P \longmapsto (q \cdot M)(\!|\nu*(r \cdot xvec)|\!)\langle(r \cdot N)\rangle \prec (r \cdot P')$ **using** *Sq* ‹$A_Q \sharp* \Psi$› ‹$(q \cdot A_Q) \sharp* \Psi$›
by(*simp add*: *eqvts*)

moreover from ‹*extractFrame* $P = \langle A_P, \Psi_P\rangle$› *Sp* ‹$(p \cdot A_P) \sharp* \Psi_P$›
have *FrP*: *extractFrame* $P = \langle(p \cdot A_P), (p \cdot \Psi_P)\rangle$
by(*simp add*: *frameChainAlpha*)
moreover from ‹*distinct* A_P› **have** *distinct*$(p \cdot A_P)$ **by** *simp*

moreover from ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|N|\!) \prec Q'$› *Sr* ‹*distinctPerm* r› ‹*xvec* $\sharp*$ Q› ‹$(r \cdot xvec) \sharp* Q$›
have $\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|(r \cdot N)|\!) \prec (r \cdot Q')$ **by**(*rule inputAlpha*)
then have $(p \cdot (\Psi \otimes \Psi_P)) \rhd Q \longmapsto (p \cdot K)(\!|(r \cdot N)|\!) \prec (r \cdot Q')$ **using** *Sp* ‹$A_P \sharp* Q$› ‹$(p \cdot A_P) \sharp* Q$›
by − (*rule inputPermFrameSubject*, (*assumption* | *simp*)+)
then have *QTrans*: $\Psi \otimes (p \cdot \Psi_P) \rhd Q \longmapsto (p \cdot K)(\!|(r \cdot N)|\!) \prec (r \cdot Q')$ **using** *Sp* ‹$A_P \sharp* \Psi$› ‹$(p \cdot A_P) \sharp* \Psi$›
by(*simp add*: *eqvts*)

moreover from ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› *Sq* ‹$(q \cdot A_Q) \sharp* \Psi_Q$›
have *FrQ*: *extractFrame* $Q = \langle(q \cdot A_Q), (q \cdot \Psi_Q)\rangle$
by(*simp add*: *frameChainAlpha*)
moreover from ‹*distinct* A_Q› **have** *distinct*$(q \cdot A_Q)$ **by** *simp*

moreover from ‹$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K$› **have** $(p \cdot q \cdot (\Psi \otimes \Psi_P \otimes \Psi_Q)) \vdash (p \cdot q \cdot M) \leftrightarrow (p \cdot q \cdot K)$
by(*metis chanEqClosed*)
with ‹$A_P \sharp* \Psi$› ‹$(p \cdot A_P) \sharp* \Psi$› ‹$A_Q \sharp* \Psi$› ‹$(q \cdot A_Q) \sharp* \Psi$› ‹$A_Q \sharp* \Psi_P$› ‹$(q \cdot A_Q) \sharp* \Psi_P$›
‹$A_P \sharp* \Psi_Q$› ‹$(p \cdot A_P) \sharp* \Psi_Q$› ‹$A_P \sharp* M$› ‹$(p \cdot A_P) \sharp* M$› ‹$(q \cdot A_Q) \sharp* A_P$› ‹$(p \cdot A_P) \sharp* (q \cdot A_Q)$›

$\langle A_Q \sharp* K\rangle$ $\langle (q \cdot A_Q) \sharp* K\rangle$ $\langle A_P \sharp* A_Q\rangle$ $\langle (p \cdot A_P) \sharp* A_Q\rangle$ *Sp Sq*
have $\Psi \otimes (p \cdot \Psi_P) \otimes (q \cdot \Psi_Q) \vdash (q \cdot M) \leftrightarrow (p \cdot K)$
by(*simp add*: *eqvts freshChainSimps*)
moreover note $\langle (p \cdot A_P) \sharp* \Psi\rangle$
moreover from $\langle (p \cdot A_P) \sharp* A_Q\rangle$ $\langle (p \cdot A_P) \sharp* (q \cdot A_Q)\rangle$ $\langle (p \cdot A_P) \sharp* \Psi_Q\rangle$ *Sq*
have $(p \cdot A_P) \sharp* (q \cdot \Psi_Q)$
by(*simp add*: *freshChainSimps*)
moreover note $\langle (p \cdot A_P) \sharp* P\rangle$
moreover from $\langle (p \cdot A_P) \sharp* A_Q\rangle$ $\langle (p \cdot A_P) \sharp* (q \cdot A_Q)\rangle$ $\langle (p \cdot A_P) \sharp* M\rangle$ *Sq*
have $(p \cdot A_P) \sharp* (q \cdot M)$
by(*simp add*: *freshChainSimps*)
moreover from $\langle (p \cdot A_P) \sharp* xvec\rangle$ $\langle (p \cdot A_P) \sharp* (r \cdot xvec)\rangle$ $\langle (p \cdot A_P) \sharp* N\rangle$ *Sr*
have $(p \cdot A_P) \sharp* (r \cdot N)$
by(*simp add*: *freshChainSimps*)
moreover from $\langle (p \cdot A_P) \sharp* xvec\rangle$ $\langle (p \cdot A_P) \sharp* (r \cdot xvec)\rangle$ $\langle (p \cdot A_P) \sharp* P'\rangle$ *Sr*
have $(p \cdot A_P) \sharp* (r \cdot P')$
by(*simp add*: *freshChainSimps*)
moreover note $\langle (p \cdot A_P) \sharp* Q\rangle$
moreover from $\langle (p \cdot A_P) \sharp* xvec\rangle$ $\langle (p \cdot A_P) \sharp* (r \cdot xvec)\rangle$ $\langle (p \cdot A_P) \sharp* Q'\rangle$ *Sr*
have $(p \cdot A_P) \sharp* (r \cdot Q')$
by(*simp add*: *freshChainSimps*)
moreover note $\langle (p \cdot A_P) \sharp* (q \cdot A_Q)\rangle$ $\langle (p \cdot A_P) \sharp* (r \cdot xvec)\rangle$ $\langle (q \cdot A_Q) \sharp* \Psi\rangle$
moreover from $\langle (q \cdot A_Q) \sharp* A_P\rangle$ $\langle (p \cdot A_P) \sharp* A_Q\rangle$ $\langle (q \cdot A_Q) \sharp* \Psi_P\rangle$ $\langle (p \cdot A_P) \sharp* (q \cdot A_Q)\rangle$ *Sp Sq* **have** $(q \cdot A_Q) \sharp* (p \cdot \Psi_P)$
by(*simp add*: *freshChainSimps*)
moreover note $\langle (q \cdot A_Q) \sharp* P\rangle$
moreover from $\langle (q \cdot A_Q) \sharp* A_P\rangle$ $\langle (p \cdot A_P) \sharp* A_Q\rangle$ $\langle (q \cdot A_Q) \sharp* K\rangle$ $\langle (p \cdot A_P) \sharp* (q \cdot A_Q)\rangle$ *Sp Sq* **have** $(q \cdot A_Q) \sharp* (p \cdot K)$
by(*simp add*: *freshChainSimps*)
moreover from $\langle (q \cdot A_Q) \sharp* xvec\rangle$ $\langle (q \cdot A_Q) \sharp* (r \cdot xvec)\rangle$ $\langle (q \cdot A_Q) \sharp* N\rangle$ *Sr*
have $(q \cdot A_Q) \sharp* (r \cdot N)$
by(*simp add*: *freshChainSimps*)
moreover from $\langle (q \cdot A_Q) \sharp* xvec\rangle$ $\langle (q \cdot A_Q) \sharp* (r \cdot xvec)\rangle$ $\langle (q \cdot A_Q) \sharp* P'\rangle$ *Sr*
have $(q \cdot A_Q) \sharp* (r \cdot P')$
by(*simp add*: *freshChainSimps*)
moreover note $\langle (q \cdot A_Q) \sharp* Q\rangle$
moreover from $\langle (q \cdot A_Q) \sharp* xvec\rangle$ $\langle (q \cdot A_Q) \sharp* (r \cdot xvec)\rangle$ $\langle (q \cdot A_Q) \sharp* Q'\rangle$ *Sr*
have $(q \cdot A_Q) \sharp* (r \cdot Q')$
by(*simp add*: *freshChainSimps*)
moreover note $\langle (q \cdot A_Q) \sharp* (r \cdot xvec)\rangle$ $\langle (r \cdot xvec) \sharp* \Psi\rangle$
moreover from $\langle (r \cdot xvec) \sharp* A_P\rangle$ $\langle (p \cdot A_P) \sharp* (r \cdot xvec)\rangle$ $\langle (r \cdot xvec) \sharp* \Psi_P\rangle$ *Sp* **have** $(r \cdot xvec) \sharp* (p \cdot \Psi_P)$
by(*simp add*: *freshChainSimps*)
moreover from $\langle (r \cdot xvec) \sharp* A_Q\rangle$ $\langle (q \cdot A_Q) \sharp* (r \cdot xvec)\rangle$ $\langle (r \cdot xvec) \sharp* \Psi_Q\rangle$ *Sq* **have** $(r \cdot xvec) \sharp* (q \cdot \Psi_Q)$
by(*simp add*: *freshChainSimps*)
moreover note $\langle (r \cdot xvec) \sharp* P\rangle$
moreover from $\langle (r \cdot xvec) \sharp* A_Q\rangle$ $\langle (q \cdot A_Q) \sharp* (r \cdot xvec)\rangle$ $\langle (r \cdot xvec) \sharp* M\rangle$ *Sq*
have $(r \cdot xvec) \sharp* (q \cdot M)$

by(*simp add*: *freshChainSimps*)
moreover note ‹$(r \cdot xvec) \sharp* Q$›
moreover from ‹$(r \cdot xvec) \sharp* A_P$› ‹$(p \cdot A_P) \sharp* (r \cdot xvec)$› ‹$(r \cdot xvec) \sharp* K$› *Sp*
have $(r \cdot xvec) \sharp* (p \cdot K)$
by(*simp add*: *freshChainSimps*)
moreover note ‹$(p \cdot A_P) \sharp* C$› ‹$(q \cdot A_Q) \sharp* C$› ‹$(r \cdot xvec) \sharp* C$›
moreover from ‹*distinct xvec*› **have** *distinct*$(r \cdot xvec)$ **by** *simp*
ultimately show *?thesis* **by**(*rule rComm2*)
next
case(*cBrMerge* Ψ_Q P M N P' A_P Ψ_P Q Q' A_Q)
obtain q::*name prm* **where** $(q \cdot A_Q) \sharp* \Psi$ **and** $(q \cdot A_Q) \sharp* P$
and $(q \cdot A_Q) \sharp* Q$ **and** $(q \cdot A_Q) \sharp* M$
and $(q \cdot A_Q) \sharp* \Psi_P$ **and** $(q \cdot A_Q) \sharp* A_P$ **and** $(q \cdot A_Q) \sharp* \Psi_Q$
and $(q \cdot A_Q) \sharp* N$ **and** $(q \cdot A_Q) \sharp* P'$ **and** $(q \cdot A_Q) \sharp* Q'$
and $(q \cdot A_Q) \sharp* C$
and *Sq*: *set* $q \subseteq$ *set* $A_Q \times$ *set*$(q \cdot A_Q)$
by(*rule name-list-avoiding*[**where** c=(Ψ, P, N, M, P', Q', Q, Ψ_Q, A_P, Ψ_P, C)])
(*auto simp add*: *eqvts fresh-star-prod emptyFresh*)
obtain p::*name prm* **where** $(p \cdot A_P) \sharp* \Psi$ **and** $(p \cdot A_P) \sharp* P$
and $(p \cdot A_P) \sharp* Q$ **and** $(p \cdot A_P) \sharp* M$
and $(p \cdot A_P) \sharp* \Psi_P$ **and** $(p \cdot A_P) \sharp* \Psi_Q$ **and** $(p \cdot A_P) \sharp* A_Q$
and $(p \cdot A_P) \sharp* N$ **and** $(p \cdot A_P) \sharp* P'$ **and** $(p \cdot A_P) \sharp* Q'$
and $(p \cdot A_P) \sharp* C$
and *Sp*: *set* $p \subseteq$ *set* $A_P \times$ *set*$(p \cdot A_P)$
and $(p \cdot A_P) \sharp* (q \cdot A_Q)$
by(*rule name-list-avoiding*[**where** c=(Ψ, P, N, P', Q', Q, M, A_Q, $q \cdot A_Q$, Ψ_Q, Ψ_P, C)])
(*auto simp add*: *eqvts fresh-star-prod*)

have *FrP*: *extractFrame* $P = \langle A_P, \Psi_P \rangle$ **by** *fact*
have *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$ **by** *fact*

from ‹$A_P \sharp* Q$› *FrQ* ‹$A_P \sharp* A_Q$› **have** $A_P \sharp* \Psi_Q$
by(*auto dest*: *extractFrameFreshChain*)
from ‹$A_Q \sharp* P$› *FrP* ‹$A_P \sharp* A_Q$› **have** $A_Q \sharp* \Psi_P$
by(*auto dest*: *extractFrameFreshChain*)

from *Sp* ‹$A_P \sharp* M$› ‹$(p \cdot A_P) \sharp* M$›
have $(p \cdot M) = M$
by *simp*
from *Sq* ‹$A_Q \sharp* M$› ‹$(q \cdot A_Q) \sharp* M$›
have $(q \cdot M) = M$
by *simp*

note ‹$cP = P \parallel Q$› ‹$cRs = ¿M⦇N⦈ \prec (P' \parallel Q')$›
moreover from ‹*distinct* A_P› **have** *distinct*$(p \cdot A_P)$ **by** *simp*
moreover from ‹*distinct* A_Q› **have** *distinct*$(q \cdot A_Q)$ **by** *simp*
moreover from ‹*extractFrame* $P = \langle A_P, \Psi_P \rangle$› *Sp* ‹$(p \cdot A_P) \sharp* \Psi_P$›

have *FrP*: *extractFrame* $P = \langle(p \cdot A_P), (p \cdot \Psi_P)\rangle$
by(*simp add*: *frameChainAlpha*)
moreover from ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› *Sq* ‹$(q \cdot A_Q) \sharp* \Psi_Q$›
have *FrQ*: *extractFrame* $Q = \langle(q \cdot A_Q), (q \cdot \Psi_Q)\rangle$
by(*simp add*: *frameChainAlpha*)

moreover have $(q \cdot (\Psi \otimes \Psi_Q)) \rhd P \longmapsto¿(q \cdot M)(\!|N|\!) \prec P'$ **using** *Sq* ‹$A_Q \sharp* P$› ‹$(q \cdot A_Q) \sharp* P$› ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto¿M(\!|N|\!) \prec P'$›
by − (*rule brinputPermFrameSubject*, (*assumption* | *simp*)+)
then have $(\Psi \otimes (q \cdot \Psi_Q)) \rhd P \longmapsto¿(q \cdot M)(\!|N|\!) \prec P'$ **using** *Sq* ‹$A_Q \sharp* \Psi$› ‹$(q \cdot A_Q) \sharp* \Psi$›
by(*simp add*: *eqvts*)
with ‹$(q \cdot M) = M$› **have** *PTrans*: $(\Psi \otimes (q \cdot \Psi_Q)) \rhd P \longmapsto¿M(\!|N|\!) \prec P'$
by *simp*

moreover have $(p \cdot (\Psi \otimes \Psi_P)) \rhd Q \longmapsto¿(p \cdot M)(\!|N|\!) \prec Q'$ **using** *Sp* ‹$A_P \sharp* Q$› ‹$(p \cdot A_P) \sharp* Q$› ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto¿M(\!|N|\!) \prec Q'$›
by − (*rule brinputPermFrameSubject*, (*assumption* | *simp*)+)
then have $(\Psi \otimes (p \cdot \Psi_P)) \rhd Q \longmapsto¿(p \cdot M)(\!|N|\!) \prec Q'$ **using** *Sp* ‹$A_P \sharp* \Psi$› ‹$(p \cdot A_P) \sharp* \Psi$›
by(*simp add*: *eqvts*)
with ‹$(p \cdot M) = M$› **have** *PTrans*: $(\Psi \otimes (p \cdot \Psi_P)) \rhd Q \longmapsto¿M(\!|N|\!) \prec Q'$
by *simp*
moreover from ‹$(p \cdot A_P) \sharp* A_Q$› ‹$(p \cdot A_P) \sharp* (q \cdot A_Q)$› ‹$(p \cdot A_P) \sharp* \Psi_Q$› *Sq* **have** $(p \cdot A_P) \sharp* (q \cdot \Psi_Q)$
by(*simp add*: *freshChainSimps*)
moreover from ‹$(q \cdot A_Q) \sharp* A_P$› ‹$(p \cdot A_P) \sharp* A_Q$› ‹$(q \cdot A_Q) \sharp* \Psi_P$› ‹$(p \cdot A_P) \sharp* (q \cdot A_Q)$› *Sp Sq* **have** $(q \cdot A_Q) \sharp* (p \cdot \Psi_P)$
by(*simp add*: *freshChainSimps*)

moreover note
‹$(p \cdot A_P) \sharp* \Psi$›
‹$(p \cdot A_P) \sharp* P$› ‹$(p \cdot A_P) \sharp* N$› ‹$(p \cdot A_P) \sharp* P'$› ‹$(p \cdot A_P) \sharp* M$›
‹$(p \cdot A_P) \sharp* Q$› ‹$(p \cdot A_P) \sharp* Q'$› ‹$(p \cdot A_P) \sharp* (q \cdot A_Q)$› ‹$(q \cdot A_Q) \sharp* \Psi$›
‹$(q \cdot A_Q) \sharp* P$› ‹$(q \cdot A_Q) \sharp* N$› ‹$(q \cdot A_Q) \sharp* P'$› ‹$(q \cdot A_Q) \sharp* M$›
‹$(q \cdot A_Q) \sharp* Q$› ‹$(q \cdot A_Q) \sharp* Q'$› ‹$(p \cdot A_P) \sharp* C$› ‹$(q \cdot A_Q) \sharp* C$›
ultimately show *?thesis*
by(*auto simp add*: *rBrMerge*)
next
case(*cBrComm1* Ψ_Q P M N P' A_P Ψ_P Q *xvec* Q' A_Q)
obtain *r*::*name prm* **where** $(r \cdot xvec) \sharp* \Psi$ **and** $(r \cdot xvec) \sharp* P$ **and** $(r \cdot xvec) \sharp* Q$ **and** $(r \cdot xvec) \sharp* M$
and $(r \cdot xvec) \sharp* N$ **and** $(r \cdot xvec) \sharp* A_P$ **and** $(r \cdot xvec) \sharp* A_Q$
and $(r \cdot xvec) \sharp* P'$ **and** $(r \cdot xvec) \sharp* Q'$ **and** $(r \cdot xvec) \sharp* \Psi_P$ **and** $(r \cdot xvec) \sharp* \Psi_Q$
and $(r \cdot xvec) \sharp* C$ **and** *Sr*: $(set\ r) \subseteq (set\ xvec) \times (set(r \cdot xvec))$ **and** *distinctPerm r*
by(*rule name-list-avoiding*[**where** *xvec*=*xvec* **and** *c*=(Ψ, P, Q, M, N, A_P, A_Q, Ψ_P, Ψ_Q, P', Q', C)])

(*auto simp add*: *eqvts*)

obtain *q*::*name prm* **where** $(q \cdot A_Q) \sharp* \Psi$ **and** $(q \cdot A_Q) \sharp* P$ **and** $(q \cdot A_Q) \sharp* Q$ **and** $(q \cdot A_Q) \sharp* M$
and $(q \cdot A_Q) \sharp* N$ **and** $(q \cdot A_Q) \sharp* xvec$ **and** $(q \cdot A_Q) \sharp* Q'$ **and** $(q \cdot A_Q) \sharp* P'$
and $(q \cdot A_Q) \sharp* \Psi_P$ **and** $(q \cdot A_Q) \sharp* A_P$ **and** $(q \cdot A_Q) \sharp* \Psi_Q$ **and** $(q \cdot A_Q) \sharp* (r \cdot xvec)$
and $(q \cdot A_Q) \sharp* C$ **and** *Sq*: $(set\ q) \subseteq (set\ A_Q) \times (set(q \cdot A_Q))$
by(*rule name-list-avoiding*[**where** $xvec{=}A_Q$ **and** $c{=}(\Psi, P, Q, M, N, xvec, r \cdot xvec, \Psi_Q, A_P, \Psi_P, Q', P', C)$]) *clarsimp*

obtain *p*::*name prm* **where** $(p \cdot A_P) \sharp* \Psi$ **and** $(p \cdot A_P) \sharp* P$ **and** $(p \cdot A_P) \sharp* Q$ **and** $(p \cdot A_P) \sharp* M$
and $(p \cdot A_P) \sharp* N$ **and** $(p \cdot A_P) \sharp* xvec$ **and** $(p \cdot A_P) \sharp* Q'$ **and** $(p \cdot A_P) \sharp* A_Q$
and $(p \cdot A_P) \sharp* P'$ **and** $(p \cdot A_P) \sharp* \Psi_P$ **and** $(p \cdot A_P) \sharp* \Psi_Q$ **and** $(p \cdot A_P) \sharp* (q \cdot A_Q)$
and $(p \cdot A_P) \sharp* C$ **and** $(p \cdot A_P) \sharp* (r \cdot xvec)$ **and** *Sp*: $(set\ p) \subseteq (set\ A_P) \times (set(p \cdot A_P))$
by(*rule name-list-avoiding*[**where** $xvec{=}A_P$ **and** $c{=}(\Psi, P, Q, M, N, xvec, r \cdot xvec, A_Q, q \cdot A_Q, \Psi_Q, \Psi_P, Q', P', C)$])
(*auto simp add*: *eqvts fresh-star-prod*)

from *Sp* ‹$A_P \sharp* M$› ‹$(p \cdot A_P) \sharp* M$›
have $(p \cdot M) = M$
by *simp*
from *Sq* ‹$A_Q \sharp* M$› ‹$(q \cdot A_Q) \sharp* M$›
have $(q \cdot M) = M$
by *simp*
from *Sr* ‹$xvec \sharp* M$› ‹$(r \cdot xvec) \sharp* M$›
have $(r \cdot M) = M$
by *simp*

have *FrP*: $extractFrame\ P = \langle A_P, \Psi_P \rangle$ **by** *fact*
have *FrQ*: $extractFrame\ Q = \langle A_Q, \Psi_Q \rangle$ **by** *fact*

from ‹$A_P \sharp* Q$› *FrQ* ‹$A_P \sharp* A_Q$› **have** $A_P \sharp* \Psi_Q$
by(*auto dest*: *extractFrameFreshChain*)
from ‹$A_Q \sharp* P$› *FrP* ‹$A_P \sharp* A_Q$› **have** $A_Q \sharp* \Psi_P$
by(*auto dest*: *extractFrameFreshChain*)

note ‹$cP = P \parallel Q$›
moreover from ‹$(r \cdot xvec) \sharp* P'$› ‹$(r \cdot xvec) \sharp* Q'$› **have** $(r \cdot xvec) \sharp* (P' \parallel Q')$
by *simp*
with ‹$cRs = ¡M⦇\nu{*}xvec⦈\langle N \rangle \prec (P' \parallel Q')$› ‹$(r \cdot xvec) \sharp* N$› ‹$(r \cdot xvec) \sharp* P'$› ‹$(r \cdot xvec) \sharp* Q'$› ‹$xvec \sharp* M$› ‹$(r \cdot xvec) \sharp* M$› *Sr*
have $cRs = (r \cdot (¡M⦇\nu{*}xvec⦈\langle N \rangle)) \prec (r \cdot (P' \parallel Q'))$

by (*simp add*: *residualAlpha*)
with ‹$(r \cdot M) = M$› **have** *cRs* = ¡$M(\!|\nu*(r \cdot xvec)|\!)\langle(r \cdot N)\rangle \prec ((r \cdot P') \parallel (r \cdot Q'))$ **by** *simp*

moreover from ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$› *Sr* ‹*distinctPerm r*› ‹*xvec* ♯* *P*› ‹$(r \cdot xvec)$ ♯* *P*›
have $\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|(r \cdot N)|\!) \prec (r \cdot P')$
by(*rule brinputAlpha*)
then have $(q \cdot (\Psi \otimes \Psi_Q)) \rhd P \longmapsto ¿(q \cdot M)(\!|(r \cdot N)|\!) \prec (r \cdot P')$ **using** *Sq* ‹A_Q ♯* *P*› ‹$(q \cdot A_Q)$ ♯* *P*›
by − (*rule brinputPermFrameSubject*, (*assumption* | *simp*)+)
then have *PTrans*: $\Psi \otimes (q \cdot \Psi_Q) \rhd P \longmapsto ¿M(\!|(r \cdot N)|\!) \prec (r \cdot P')$ **using** *Sq* ‹A_Q ♯* Ψ› ‹$(q \cdot A_Q)$ ♯* Ψ› ‹$(q \cdot M) = M$›
by(*simp add*: *eqvts*)

moreover from ‹*extractFrame* $P = \langle A_P, \Psi_P\rangle$› *Sp* ‹$(p \cdot A_P)$ ♯* Ψ_P›
have *FrP*: *extractFrame* $P = \langle(p \cdot A_P), (p \cdot \Psi_P)\rangle$
by(*simp add*: *frameChainAlpha*)
moreover from ‹*distinct* A_P› **have** *distinct*$(p \cdot A_P)$ **by** *simp*

moreover from ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec Q'$› *Sr* ‹$(r \cdot xvec)$ ♯* N› ‹$(r \cdot xvec)$ ♯* Q'›
have $\Psi \otimes \Psi_P \rhd Q \longmapsto ¡M(\!|\nu*(r \cdot xvec)|\!)\langle(r \cdot N)\rangle \prec (r \cdot Q')$
by(*simp add*: *boundOutputChainAlpha*'' *residualInject*)
then have $(p \cdot (\Psi \otimes \Psi_P)) \rhd Q \longmapsto ¡(p \cdot M)(\!|\nu*(r \cdot xvec)|\!)\langle(r \cdot N)\rangle \prec (r \cdot Q')$ **using** *Sp* ‹A_P ♯* Q› ‹$(p \cdot A_P)$ ♯* Q› ‹$(r \cdot xvec)$ ♯* M› ‹$(p \cdot A_P)$ ♯* $(r \cdot xvec)$› ‹$(r \cdot xvec)$ ♯* A_P›
by(*fastforce intro*: *broutputPermFrameSubject*)
then have *QTrans*: $\Psi \otimes (p \cdot \Psi_P) \rhd Q \longmapsto ¡M(\!|\nu*(r \cdot xvec)|\!)\langle(r \cdot N)\rangle \prec (r \cdot Q')$ **using** *Sp* ‹A_P ♯* Ψ› ‹$(p \cdot A_P)$ ♯* Ψ› ‹$(p \cdot M) = M$›
by(*simp add*: *eqvts*)

moreover from ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› *Sq* ‹$(q \cdot A_Q)$ ♯* Ψ_Q›
have *FrQ*: *extractFrame* $Q = \langle(q \cdot A_Q), (q \cdot \Psi_Q)\rangle$
by(*simp add*: *frameChainAlpha*)
moreover from ‹*distinct* A_Q› **have** *distinct*$(q \cdot A_Q)$ **by** *simp*

moreover note ‹$(p \cdot A_P)$ ♯* Ψ›
moreover from ‹$(p \cdot A_P)$ ♯* A_Q› ‹$(p \cdot A_P)$ ♯* $(q \cdot A_Q)$› ‹$(p \cdot A_P)$ ♯* Ψ_Q› *Sq* **have** $(p \cdot A_P)$ ♯* $(q \cdot \Psi_Q)$
by(*simp add*: *freshChainSimps*)
moreover note ‹$(p \cdot A_P)$ ♯* P›
moreover from ‹$(p \cdot A_P)$ ♯* *xvec*› ‹$(p \cdot A_P)$ ♯* $(r \cdot xvec)$› ‹$(p \cdot A_P)$ ♯* N› *Sr* **have** $(p \cdot A_P)$ ♯* $(r \cdot N)$
by(*simp add*: *freshChainSimps*)
moreover from ‹$(p \cdot A_P)$ ♯* *xvec*› ‹$(p \cdot A_P)$ ♯* $(r \cdot xvec)$› ‹$(p \cdot A_P)$ ♯* P'› *Sr* **have** $(p \cdot A_P)$ ♯* $(r \cdot P')$
by(*simp add*: *freshChainSimps*)
moreover note ‹$(p \cdot A_P)$ ♯* Q›

moreover from ‹$(p \cdot A_P) \sharp* xvec$› ‹$(p \cdot A_P) \sharp* (r \cdot xvec)$› ‹$(p \cdot A_P) \sharp* Q'$› *Sr* **have** $(p \cdot A_P) \sharp* (r \cdot Q')$
by(*simp add*: *freshChainSimps*)
moreover note ‹$(p \cdot A_P) \sharp* (q \cdot A_Q)$› ‹$(p \cdot A_P) \sharp* (r \cdot xvec)$› ‹$(q \cdot A_Q) \sharp* \Psi$›
moreover from ‹$(q \cdot A_Q) \sharp* A_P$› ‹$(p \cdot A_P) \sharp* A_Q$› ‹$(q \cdot A_Q) \sharp* \Psi_P$› ‹$(p \cdot A_P) \sharp* (q \cdot A_Q)$› *Sp Sq* **have** $(q \cdot A_Q) \sharp* (p \cdot \Psi_P)$
by(*simp add*: *freshChainSimps*)
moreover note ‹$(q \cdot A_Q) \sharp* P$›
moreover from ‹$(q \cdot A_Q) \sharp* xvec$› ‹$(q \cdot A_Q) \sharp* (r \cdot xvec)$› ‹$(q \cdot A_Q) \sharp* N$› *Sr* **have** $(q \cdot A_Q) \sharp* (r \cdot N)$
by(*simp add*: *freshChainSimps*)
moreover from ‹$(q \cdot A_Q) \sharp* xvec$› ‹$(q \cdot A_Q) \sharp* (r \cdot xvec)$› ‹$(q \cdot A_Q) \sharp* P'$› *Sr* **have** $(q \cdot A_Q) \sharp* (r \cdot P')$
by(*simp add*: *freshChainSimps*)
moreover note ‹$(q \cdot A_Q) \sharp* Q$›
moreover from ‹$(q \cdot A_Q) \sharp* xvec$› ‹$(q \cdot A_Q) \sharp* (r \cdot xvec)$› ‹$(q \cdot A_Q) \sharp* Q'$› *Sr* **have** $(q \cdot A_Q) \sharp* (r \cdot Q')$
by(*simp add*: *freshChainSimps*)
moreover note ‹$(q \cdot A_Q) \sharp* (r \cdot xvec)$› ‹$(r \cdot xvec) \sharp* \Psi$›
moreover from ‹$(r \cdot xvec) \sharp* A_P$› ‹$(p \cdot A_P) \sharp* (r \cdot xvec)$› ‹$(r \cdot xvec) \sharp* \Psi_P$› *Sp* **have** $(r \cdot xvec) \sharp* (p \cdot \Psi_P)$
by(*simp add*: *freshChainSimps*)
moreover from ‹$(r \cdot xvec) \sharp* A_Q$› ‹$(q \cdot A_Q) \sharp* (r \cdot xvec)$› ‹$(r \cdot xvec) \sharp* \Psi_Q$› *Sq* **have** $(r \cdot xvec) \sharp* (q \cdot \Psi_Q)$
by(*simp add*: *freshChainSimps*)
moreover note ‹$(p \cdot A_P) \sharp* M$› ‹$(q \cdot A_Q) \sharp* M$›
moreover note ‹$(r \cdot xvec) \sharp* P$›
moreover note ‹$(r \cdot xvec) \sharp* Q$› ‹$(r \cdot xvec) \sharp* M$›
moreover note ‹$(p \cdot A_P) \sharp* C$› ‹$(q \cdot A_Q) \sharp* C$› ‹$(r \cdot xvec) \sharp* C$›
moreover from ‹*distinct xvec*› **have** *distinct*$(r \cdot xvec)$ **by** *simp*
ultimately show *?thesis* **by**(*simp add*: *rBrComm1*)
next
case(*cBrComm2* Ψ_Q P M *xvec* N P' A_P Ψ_P Q Q' A_Q)
obtain *r*::*name prm* **where** $(r \cdot xvec) \sharp* \Psi$ **and** $(r \cdot xvec) \sharp* P$ **and** $(r \cdot xvec) \sharp* Q$ **and** $(r \cdot xvec) \sharp* M$
and $(r \cdot xvec) \sharp* N$ **and** $(r \cdot xvec) \sharp* A_P$ **and** $(r \cdot xvec) \sharp* A_Q$
and $(r \cdot xvec) \sharp* P'$ **and** $(r \cdot xvec) \sharp* Q'$ **and** $(r \cdot xvec) \sharp* \Psi_P$ **and** $(r \cdot xvec) \sharp* \Psi_Q$
and $(r \cdot xvec) \sharp* C$ **and** *Sr*: $(set\ r) \subseteq (set\ xvec) \times (set(r \cdot xvec))$ **and** *distinctPerm r*
by(*rule name-list-avoiding*[**where** *xvec*=*xvec* **and** *c*=(Ψ, P, Q, M, N, A_P, A_Q, Ψ_P, Ψ_Q, P', Q', C)])
(*auto simp add*: *eqvts*)

obtain *q*::*name prm* **where** $(q \cdot A_Q) \sharp* \Psi$ **and** $(q \cdot A_Q) \sharp* P$ **and** $(q \cdot A_Q) \sharp* Q$ **and** $(q \cdot A_Q) \sharp* M$
and $(q \cdot A_Q) \sharp* N$ **and** $(q \cdot A_Q) \sharp* xvec$ **and** $(q \cdot A_Q) \sharp* Q'$ **and** $(q \cdot A_Q) \sharp* P'$
and $(q \cdot A_Q) \sharp* \Psi_P$ **and** $(q \cdot A_Q) \sharp* A_P$ **and** $(q \cdot A_Q) \sharp* \Psi_Q$ **and** $(q \cdot A_Q)$

$\sharp*$ $(r \cdot xvec)$
and $(q \cdot A_Q) \sharp* C$ **and** *Sq*: $(set\ q) \subseteq (set\ A_Q) \times (set(q \cdot A_Q))$
by(*rule name-list-avoiding*[**where** $xvec{=}A_Q$ **and** $c{=}(\Psi, P, Q, N, M, xvec, r \cdot xvec, \Psi_Q, A_P, \Psi_P, Q', P', C)$]) *clarsimp*

obtain p::*name prm* **where** $(p \cdot A_P) \sharp* \Psi$ **and** $(p \cdot A_P) \sharp* P$ **and** $(p \cdot A_P) \sharp* Q$ **and** $(p \cdot A_P) \sharp* M$
and $(p \cdot A_P) \sharp* N$ **and** $(p \cdot A_P) \sharp* xvec$ **and** $(p \cdot A_P) \sharp* Q'$ **and** $(p \cdot A_P) \sharp* A_Q$
and $(p \cdot A_P) \sharp* P'$ **and** $(p \cdot A_P) \sharp* \Psi_P$ **and** $(p \cdot A_P) \sharp* \Psi_Q$ **and** $(p \cdot A_P) \sharp* (q \cdot A_Q)$
and $(p \cdot A_P) \sharp* C$ **and** $(p \cdot A_P) \sharp* (r \cdot xvec)$ **and** *Sp*: $(set\ p) \subseteq (set\ A_P) \times (set(p \cdot A_P))$
by(*rule name-list-avoiding*[**where** $xvec{=}A_P$ **and** $c{=}(\Psi, P, Q, M, N, xvec, r \cdot xvec, A_Q, q \cdot A_Q, \Psi_Q, \Psi_P, Q', P', C)$])
(*auto simp add*: *eqvts fresh-star-prod*)

from *Sp* ‹$A_P \sharp* M$› ‹$(p \cdot A_P) \sharp* M$›
have $(p \cdot M) = M$
by *simp*
from *Sq* ‹$A_Q \sharp* M$› ‹$(q \cdot A_Q) \sharp* M$›
have $(q \cdot M) = M$
by *simp*
from *Sr* ‹$xvec \sharp* M$› ‹$(r \cdot xvec) \sharp* M$›
have $(r \cdot M) = M$
by *simp*

have *FrP*: $extractFrame\ P = \langle A_P, \Psi_P \rangle$ **by** *fact*
have *FrQ*: $extractFrame\ Q = \langle A_Q, \Psi_Q \rangle$ **by** *fact*

from ‹$A_P \sharp* Q$› *FrQ* ‹$A_P \sharp* A_Q$› **have** $A_P \sharp* \Psi_Q$
by(*auto dest*: *extractFrameFreshChain*)
from ‹$A_Q \sharp* P$› *FrP* ‹$A_P \sharp* A_Q$› **have** $A_Q \sharp* \Psi_P$
by(*auto dest*: *extractFrameFreshChain*)

note ‹$cP = P \parallel Q$›
moreover from ‹$(r \cdot xvec) \sharp* P'$› ‹$(r \cdot xvec) \sharp* Q'$› **have** $(r \cdot xvec) \sharp* (P' \parallel Q')$
by *simp*
with ‹$cRs = ¡M(\!|\nu{*}xvec|\!)\langle N\rangle \prec (P' \parallel Q')$› ‹$(r \cdot xvec) \sharp* N$› ‹$(r \cdot xvec) \sharp* P'$› ‹$(r \cdot xvec) \sharp* Q'$› ‹$xvec \sharp* M$› ‹$(r \cdot xvec) \sharp* M$› *Sr*
have $cRs = (r \cdot (¡M(\!|\nu{*}xvec|\!)\langle N\rangle)) \prec (r \cdot (P' \parallel Q'))$
by (*simp add*: *residualAlpha*)
with ‹$(r \cdot M) = M$› **have** $cRs = ¡M(\!|\nu{*}(r \cdot xvec)|\!)\langle (r \cdot N)\rangle \prec ((r \cdot P') \parallel (r \cdot Q'))$ **by** *simp*

moreover from ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto ¡M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P'$› *Sr* ‹$(r \cdot xvec) \sharp* N$› ‹$(r \cdot xvec) \sharp* P'$›
have $\Psi \otimes \Psi_Q \rhd P \longmapsto ¡M(\!|\nu{*}(r \cdot xvec)|\!)\langle (r \cdot N)\rangle \prec (r \cdot P')$ **by**(*simp add*:

boundOutputChainAlpha'' *residualInject*)
then have $(q \cdot (\Psi \otimes \Psi_Q)) \rhd P \longmapsto ¡(q \cdot M)(\!|\nu*(r \cdot xvec)|\!)\langle(r \cdot N)\rangle \prec (r \cdot P')$ **using** *Sq* ‹$A_Q \sharp* P$› ‹$(q \cdot A_Q) \sharp* P$› ‹$(r \cdot xvec) \sharp* M$› ‹$(r \cdot xvec) \sharp* A_Q$› ‹$(q \cdot A_Q) \sharp* (r \cdot xvec)$›
by(*fastforce intro*: *broutputPermFrameSubject*)
then have *PTrans*: $\Psi \otimes (q \cdot \Psi_Q) \rhd P \longmapsto ¡M(\!|\nu*(r \cdot xvec)|\!)\langle(r \cdot N)\rangle \prec (r \cdot P')$ **using** *Sq* ‹$A_Q \sharp* \Psi$› ‹$(q \cdot A_Q) \sharp* \Psi$› ‹$(q \cdot M) = M$›
by(*simp add*: *eqvts*)

moreover from ‹*extractFrame* $P = \langle A_P, \Psi_P\rangle$› *Sp* ‹$(p \cdot A_P) \sharp* \Psi_P$›
have *FrP*: *extractFrame* $P = \langle(p \cdot A_P), (p \cdot \Psi_P)\rangle$
by(*simp add*: *frameChainAlpha*)
moreover from ‹*distinct* A_P› **have** *distinct*$(p \cdot A_P)$ **by** *simp*

moreover from ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$› *Sr* ‹*distinctPerm* r› ‹$xvec \sharp* Q$› ‹$(r \cdot xvec) \sharp* Q$›
have $\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|(r \cdot N)|\!) \prec (r \cdot Q')$ **by**(*rule brinputAlpha*)
then have $(p \cdot (\Psi \otimes \Psi_P)) \rhd Q \longmapsto ¿(p \cdot M)(\!|(r \cdot N)|\!) \prec (r \cdot Q')$ **using** *Sp* ‹$A_P \sharp* Q$› ‹$(p \cdot A_P) \sharp* Q$›
by − (*rule brinputPermFrameSubject*, (*assumption* | *simp*)+)
then have *QTrans*: $\Psi \otimes (p \cdot \Psi_P) \rhd Q \longmapsto ¿M(\!|(r \cdot N)|\!) \prec (r \cdot Q')$ **using** *Sp* ‹$A_P \sharp* \Psi$› ‹$(p \cdot A_P) \sharp* \Psi$› ‹$(p \cdot M) = M$›
by(*simp add*: *eqvts*)

moreover from ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› *Sq* ‹$(q \cdot A_Q) \sharp* \Psi_Q$›
have *FrQ*: *extractFrame* $Q = \langle(q \cdot A_Q), (q \cdot \Psi_Q)\rangle$
by(*simp add*: *frameChainAlpha*)
moreover from ‹*distinct* A_Q› **have** *distinct*$(q \cdot A_Q)$ **by** *simp*

moreover note ‹$(p \cdot A_P) \sharp* \Psi$›
moreover from ‹$(p \cdot A_P) \sharp* A_Q$› ‹$(p \cdot A_P) \sharp* (q \cdot A_Q)$› ‹$(p \cdot A_P) \sharp* \Psi_Q$› *Sq* **have** $(p \cdot A_P) \sharp* (q \cdot \Psi_Q)$
by(*simp add*: *freshChainSimps*)
moreover note ‹$(p \cdot A_P) \sharp* P$›
moreover from ‹$(p \cdot A_P) \sharp* xvec$› ‹$(p \cdot A_P) \sharp* (r \cdot xvec)$› ‹$(p \cdot A_P) \sharp* N$› *Sr* **have** $(p \cdot A_P) \sharp* (r \cdot N)$
by(*simp add*: *freshChainSimps*)
moreover from ‹$(p \cdot A_P) \sharp* xvec$› ‹$(p \cdot A_P) \sharp* (r \cdot xvec)$› ‹$(p \cdot A_P) \sharp* P'$› *Sr* **have** $(p \cdot A_P) \sharp* (r \cdot P')$
by(*simp add*: *freshChainSimps*)
moreover note ‹$(p \cdot A_P) \sharp* Q$›
moreover from ‹$(p \cdot A_P) \sharp* xvec$› ‹$(p \cdot A_P) \sharp* (r \cdot xvec)$› ‹$(p \cdot A_P) \sharp* Q'$› *Sr* **have** $(p \cdot A_P) \sharp* (r \cdot Q')$
by(*simp add*: *freshChainSimps*)
moreover note ‹$(p \cdot A_P) \sharp* (q \cdot A_Q)$› ‹$(p \cdot A_P) \sharp* (r \cdot xvec)$› ‹$(q \cdot A_Q) \sharp* \Psi$›
moreover from ‹$(q \cdot A_Q) \sharp* A_P$› ‹$(p \cdot A_P) \sharp* A_Q$› ‹$(q \cdot A_Q) \sharp* \Psi_P$› ‹$(p \cdot A_P) \sharp* (q \cdot A_Q)$› *Sp Sq* **have** $(q \cdot A_Q) \sharp* (p \cdot \Psi_P)$
by(*simp add*: *freshChainSimps*)
moreover note ‹$(q \cdot A_Q) \sharp* P$›

moreover from ‹$(q \cdot A_Q) \sharp* xvec$› ‹$(q \cdot A_Q) \sharp* (r \cdot xvec)$› ‹$(q \cdot A_Q) \sharp* N$› *Sr* **have** $(q \cdot A_Q) \sharp* (r \cdot N)$
by(*simp add*: *freshChainSimps*)
moreover from ‹$(q \cdot A_Q) \sharp* xvec$› ‹$(q \cdot A_Q) \sharp* (r \cdot xvec)$› ‹$(q \cdot A_Q) \sharp* P'$› *Sr* **have** $(q \cdot A_Q) \sharp* (r \cdot P')$
by(*simp add*: *freshChainSimps*)
moreover note ‹$(q \cdot A_Q) \sharp* Q$›
moreover from ‹$(q \cdot A_Q) \sharp* xvec$› ‹$(q \cdot A_Q) \sharp* (r \cdot xvec)$› ‹$(q \cdot A_Q) \sharp* Q'$› *Sr* **have** $(q \cdot A_Q) \sharp* (r \cdot Q')$
by(*simp add*: *freshChainSimps*)
moreover note ‹$(q \cdot A_Q) \sharp* (r \cdot xvec)$› ‹$(r \cdot xvec) \sharp* \Psi$›
moreover from ‹$(r \cdot xvec) \sharp* A_P$› ‹$(p \cdot A_P) \sharp* (r \cdot xvec)$› ‹$(r \cdot xvec) \sharp* \Psi_P$› *Sp* **have** $(r \cdot xvec) \sharp* (p \cdot \Psi_P)$
by(*simp add*: *freshChainSimps*)
moreover from ‹$(r \cdot xvec) \sharp* A_Q$› ‹$(q \cdot A_Q) \sharp* (r \cdot xvec)$› ‹$(r \cdot xvec) \sharp* \Psi_Q$› *Sq* **have** $(r \cdot xvec) \sharp* (q \cdot \Psi_Q)$
by(*simp add*: *freshChainSimps*)
moreover note ‹$(p \cdot A_P) \sharp* M$› ‹$(q \cdot A_Q) \sharp* M$›
moreover note ‹$(r \cdot xvec) \sharp* P$›
moreover note ‹$(r \cdot xvec) \sharp* Q$› ‹$(r \cdot xvec) \sharp* M$›
moreover note ‹$(p \cdot A_P) \sharp* C$› ‹$(q \cdot A_Q) \sharp* C$› ‹$(r \cdot xvec) \sharp* C$›
moreover from ‹*distinct xvec*› **have** *distinct*$(r \cdot xvec)$ **by** *simp*
ultimately show *?thesis* **by**(*simp add*: *rBrComm2*)
next
case(*cBrClose P M xvec N P' x*)
obtain *r*::*name prm* **where** $(r \cdot xvec) \sharp* \Psi$ **and** $(r \cdot xvec) \sharp* P$ **and** $(r \cdot xvec) \sharp* M$
and $(r \cdot xvec) \sharp* N$ **and** $(r \cdot xvec) \sharp* P'$ **and** $(r \cdot xvec) \sharp* x$
and $(r \cdot xvec) \sharp* C$ **and** *Sr*: $(set\ r) \subseteq (set\ xvec) \times (set(r \cdot xvec))$ **and** *distinctPerm r*
by(*rule name-list-avoiding*[**where** *xvec*=*xvec* **and** *c*=(Ψ, *P*, *M*, *N*, *P'*, *x*, *C*)])
(*auto simp add*: *eqvts*)
obtain *y*::*name* **where** $y \sharp P$ **and** $y \sharp C$ **and** $y \sharp xvec$ **and** $y \neq x$ **and** $y \sharp N$
and $y \sharp (r \cdot xvec)$ **and** $y \sharp r$ **and** $y \sharp M$ **and** $y \sharp \Psi$
and $y \sharp P'$ **and** $y \sharp (r \cdot P')$ **and** $y \sharp (r \cdot N)$
by(*generate-fresh name*) (*auto simp add*: *freshChainSimps*)
from ‹$cP = (\!|\nu x|\!)P$› ‹$y \sharp P$› **have** *cP-perm*: $cP = (\!|\nu y|\!)([(x, y)] \cdot P)$ **by**(*simp add*: *alphaRes*)
from ‹$cRs = \tau \prec (\!|\nu x|\!)((\!|\nu * xvec|\!)P')$› ‹$(r \cdot xvec) \sharp* P'$› *Sr*
have $cRs = \tau \prec (\!|\nu x|\!)((\!|\nu * (r \cdot xvec)|\!)(r \cdot P'))$ **by**(*simp add*: *resChainAlpha*)
moreover from ‹$y \sharp (r \cdot P')$› **have** $y \sharp ((\!|\nu * (r \cdot xvec)|\!)(r \cdot P'))$ **by**(*simp add*: *resChainFresh*)
ultimately have $cRs = \tau \prec (\!|\nu y|\!)([(x, y)] \cdot ((\!|\nu * (r \cdot xvec)|\!)(r \cdot P')))$ **by**(*simp add*: *alphaRes*)
with ‹$(r \cdot xvec) \sharp* x$› ‹$y \sharp (r \cdot xvec)$›
have *cRs-perm*: $cRs = \tau \prec (\!|\nu y|\!)((\!|\nu * (r \cdot xvec)|\!)([(x, y)] \cdot (r \cdot P')))$ **by**(*simp add*: *eqvts*)

from ‹$x \sharp xvec$› ‹$(r \cdot xvec) \sharp* x$› *Sr* **have** $r \cdot x = x$ **by** *simp*

from ‹$\Psi \rhd P \longmapsto ¡M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P'$› ‹$(r \cdot xvec) \sharp{*} N$› ‹$(r \cdot xvec) \sharp{*} P'$›
‹*set* $r \subseteq$ *set* $xvec \times$ *set* $(r \cdot xvec)$›
have $\Psi \rhd P \longmapsto ¡M(\!|\nu{*}(r \cdot xvec)|\!)\langle(r \cdot N)\rangle \prec (r \cdot P')$
by(*simp add*: *boundOutputChainAlpha''* *create-residual.simps*)
then have $[(x, y)] \cdot (\Psi \rhd P \longmapsto ¡M(\!|\nu{*}(r \cdot xvec)|\!)\langle(r \cdot N)\rangle \prec (r \cdot P'))$
by(*simp add*: *perm-bool*)
with ‹$x \sharp \Psi$› ‹$y \sharp \Psi$› ‹$(r \cdot xvec) \sharp{*} x$› ‹$y \sharp (r \cdot xvec)$›
‹$y \sharp (r \cdot N)$›
have *trans-perm*: $\Psi \rhd ([(x, y)] \cdot P) \longmapsto ¡([(x, y)] \cdot M)(\!|\nu{*}(r \cdot xvec)|\!)\langle([(x, y)] \cdot (r \cdot N))\rangle \prec ([(x, y)] \cdot (r \cdot P'))$
by(*auto simp add*: *eqvts*)

note *cP-perm cRs-perm*
moreover from ‹$x \in$ *supp* M› **have** $y \in$ *supp* $([(x, y)] \cdot M)$
by (*metis fresh-bij fresh-def swap-simps*)
moreover note *trans-perm*
moreover from ‹*distinct xvec*› ‹*distinctPerm r*› **have** *distinct* $(r \cdot xvec)$
by *simp*
moreover note ‹$(r \cdot xvec) \sharp{*} \Psi$›
moreover from ‹$(r \cdot xvec) \sharp{*} x$› ‹$y \sharp (r \cdot xvec)$› ‹$(r \cdot xvec) \sharp{*} P$›
have $(r \cdot xvec) \sharp{*} ([(x, y)] \cdot P)$ **by** *simp*
moreover from ‹$(r \cdot xvec) \sharp{*} x$› ‹$y \sharp (r \cdot xvec)$› ‹$(r \cdot xvec) \sharp{*} M$›
have $(r \cdot xvec) \sharp{*} ([(x, y)] \cdot M)$ **by** *simp*
moreover note ‹$y \sharp \Psi$› ‹$y \sharp (r \cdot xvec)$›
‹$(r \cdot xvec) \sharp{*} C$› ‹$y \sharp C$›
ultimately show *?thesis*
by(*rule rBrClose*)
next
case(*cOpen P M xvec yvec N P' x*)
from ‹$\Psi \rhd P \longmapsto M(\!|\nu{*}(xvec@yvec)|\!)\langle N\rangle \prec P'$› **have** *distinct*(*xvec@yvec*)
by(*force dest*: *boundOutputDistinct*)
then have *xvec* $\sharp{*}$ *yvec* **by**(*induct xvec*) *auto*
obtain *p* **where** $(p \cdot yvec) \sharp{*} \Psi$ **and** $(p \cdot yvec) \sharp{*} P$ **and** $(p \cdot yvec) \sharp{*} M$
and $(p \cdot yvec) \sharp{*} yvec$ **and** $(p \cdot yvec) \sharp{*} N$ **and** $(p \cdot yvec) \sharp{*} P'$
and $x \sharp (p \cdot yvec)$ **and** $(p \cdot yvec) \sharp{*} xvec$
and $(p \cdot yvec) \sharp{*} C$ **and** *Sp*: $(set\ p) \subseteq (set\ yvec) \times (set(p \cdot yvec))$
by(*rule name-list-avoiding*[**where** *xvec*=*yvec* **and** *c*=(Ψ, *P*, *M*, *xvec*, *yvec*, *N*, *P'*, *x*, *C*)])
(*auto simp add*: *eqvts fresh-star-prod*)
obtain *q* **where** $(q \cdot xvec) \sharp{*} \Psi$ **and** $(q \cdot xvec) \sharp{*} P$ **and** $(q \cdot xvec) \sharp{*} M$
and $(q \cdot xvec) \sharp{*} xvec$ **and** $(q \cdot xvec) \sharp{*} N$ **and** $(q \cdot xvec) \sharp{*} P'$
and $x \sharp (q \cdot xvec)$ **and** $(q \cdot xvec) \sharp{*} yvec$
and $(q \cdot xvec) \sharp{*} p$ **and** $(q \cdot xvec) \sharp{*} (p \cdot yvec)$
and $(q \cdot xvec) \sharp{*} C$ **and** *Sq*: $(set\ q) \subseteq (set\ xvec) \times (set(q \cdot xvec))$
by(*rule name-list-avoiding*[**where** *xvec*=*xvec* **and** *c*=(Ψ, *P*, *M*, *xvec*, *yvec*, $p \cdot yvec$, *N*, *P'*, *x*, *p*, *C*)])
(*auto simp add*: *eqvts fresh-star-prod*)
obtain *y*::*name* **where** $y \sharp P$ **and** $y \sharp C$ **and** $y \sharp xvec$ **and** $y \sharp yvec$ **and** $y \neq x$

and $y \sharp N$
  and $y \sharp (q \cdot xvec)$ **and** $y \sharp (p \cdot yvec)$ **and** $y \sharp M$ **and** $y \sharp \Psi$ **and** $y \sharp P'$
  by(*generate-fresh name*) (*auto simp add*: *freshChainSimps*)
 from ‹$cP = (\!|\nu x|\!)P$› ‹$y \sharp P$› **have** $cP = (\!|\nu y|\!)([(x, y)] \cdot P)$ **by**(*simp add*: *alphaRes*)
 moreover have $cRs = M(\!|\nu*((q \cdot xvec)@y\#(p \cdot yvec))|\!)\langle((q@(x, y)\#p) \cdot N)\rangle \prec ((q@(x, y)\#p) \cdot P')$
 proof −
  note ‹$cRs = M(\!|\nu*(xvec@x\#yvec)|\!)\langle N\rangle \prec P'$›
  moreover have $(\!|\nu*(xvec@x\#yvec)|\!)N \prec' P' = (\!|\nu*xvec|\!)((\!|\nu x|\!)((\!|\nu*yvec|\!)N \prec' P'))$ **by**(*simp add*: *boundOutputApp*)
  moreover from ‹$(p \cdot yvec) \sharp* N$› ‹$(p \cdot yvec) \sharp* P'$› *Sp* **have** $\ldots = (\!|\nu*xvec|\!)((\!|\nu x|\!)((\!|\nu*(p \cdot yvec)|\!)(p \cdot N) \prec' (p \cdot P')))$
    by(*simp add*: *boundOutputChainAlpha''*)
  moreover with ‹$y \sharp N$› ‹$y \sharp P'$› ‹$y \sharp (p \cdot yvec)$› ‹$y \sharp yvec$› ‹$x \sharp yvec$› ‹$x \sharp (p \cdot yvec)$› *Sp*
  moreover have $\ldots = (\!|\nu*xvec|\!)((\!|\nu y|\!)((\!|\nu*(p \cdot yvec)|\!)(([(x, y)] \cdot p \cdot N) \prec' ([(x, y)] \cdot p \cdot P'))))$
    by(*subst alphaBoundOutput*[**where** $y{=}y$]) (*simp add*: *freshChainSimps eqvts*)+
  moreover then have $\ldots = (\!|\nu*xvec|\!)((\!|\nu y|\!)((\!|\nu*(p \cdot yvec)|\!)((((x, y)\#p) \cdot N) \prec' (((x, y)\#p) \cdot P'))))$
    by *simp*
  moreover from ‹$(q \cdot xvec) \sharp* N$› ‹$(q \cdot xvec) \sharp* P'$› ‹$xvec \sharp* yvec$› ‹$(p \cdot yvec) \sharp* xvec$› ‹$(q \cdot xvec) \sharp* yvec$› ‹$(q \cdot xvec) \sharp* (p \cdot yvec)$›
    ‹$y \sharp xvec$› ‹$y \sharp (q \cdot xvec)$› ‹$x \sharp xvec$› ‹$x \sharp (q \cdot xvec)$› *Sp Sq*
  have $\ldots = (\!|\nu*(q \cdot xvec)|\!)((\!|\nu y|\!)((\!|\nu*(p \cdot yvec)|\!)((q \cdot ((x, y)\#p) \cdot N) \prec' (q \cdot ((x, y)\#p) \cdot P'))))$
  apply(*subst boundOutputChainAlpha*[**where** $p{=}q$ **and** $xvec{=}xvec$ **and** $yvec{=}xvec$])
    defer
    apply *assumption*
   apply *simp*
  apply(*simp add*: *eqvts*)
  apply(*simp add*: *eqvts*)
  apply(*simp add*: *boundOutputFreshSet*(*4*))
  apply(*rule conjI*)
   apply(*simp add*: *freshChainSimps*)
  apply(*simp add*: *freshChainSimps*)
  done
  moreover then have $\ldots = (\!|\nu*(q \cdot xvec@y\#(p \cdot yvec))|\!)((q@(x, y)\#p) \cdot N) \prec' ((q@(x, y)\#p) \cdot P')$
    by(*simp only*: *pt2*[*OF pt-name-inst*] *boundOutputApp BOresChain.simps*)
  ultimately show *?thesis*
    by(*simp only*: *residualInject*)
 qed
 moreover have $\Psi \rhd ([(x, y)] \cdot P) \longmapsto M(\!|\nu*((q \cdot xvec)@(p \cdot yvec))|\!)\langle((q@(x, y)\#p) \cdot N)\rangle \prec ((q@(x, y)\#p) \cdot P')$
 proof −
  note‹$\Psi \rhd P \longmapsto M(\!|\nu*(xvec@yvec)|\!)\langle N\rangle \prec P'$›
  moreover from ‹$(p \cdot yvec) \sharp* N$› ‹$(q \cdot xvec) \sharp* N$› ‹$xvec \sharp* yvec$› ‹$(q \cdot xvec) \sharp* yvec$› ‹$(q \cdot xvec) \sharp* (p \cdot yvec)$› ‹$(p \cdot yvec) \sharp* xvec$› *Sp Sq*

have $((q@p) \cdot (xvec\ @\ yvec))\ \sharp*\ N$ **apply**(*simp only*: *eqvts*) **apply**(*simp only*: *pt2*[*OF pt-name-inst*])
by *simp*
moreover from ‹$(p \cdot yvec)\ \sharp*\ P'$› ‹$(q \cdot xvec)\ \sharp*\ P'$› ‹$xvec\ \sharp*\ yvec$› ‹$(q \cdot xvec)\ \sharp*\ yvec$› ‹$(q \cdot xvec)\ \sharp*\ (p \cdot yvec)$› ‹$(p \cdot yvec)\ \sharp*\ xvec$› *Sp Sq*
have $((q@p) \cdot (xvec\ @\ yvec))\ \sharp*\ P'$ **by**(*simp del*: *freshAlphaPerm add*: *eqvts pt2*[*OF pt-name-inst*])
moreover from *Sp Sq* ‹$xvec\ \sharp*\ yvec$› ‹$(q \cdot xvec)\ \sharp*\ yvec$› ‹$(q \cdot xvec)\ \sharp*\ (p \cdot yvec)$› ‹$(p \cdot yvec)\ \sharp*\ xvec$›
have *Spq*: $set(q@p) \subseteq set(xvec@yvec) \times set((q@p) \cdot (xvec@yvec))$
by(*simp add*: *pt2*[*OF pt-name-inst*] *eqvts*) *blast*
ultimately have $\Psi \rhd P \longmapsto M(\!|\nu*((q@p) \cdot (xvec@yvec))|\!)\langle((q@p) \cdot N)\rangle \prec ((q@p) \cdot P')$
apply(*simp only*: *residualInject*)
by(*erule rev-mp*) (*subst boundOutputChainAlpha*, *auto*)
with *Sp Sq* ‹$xvec\ \sharp*\ yvec$› ‹$(q \cdot xvec)\ \sharp*\ yvec$› ‹$(q \cdot xvec)\ \sharp*\ (p \cdot yvec)$› ‹$(p \cdot yvec)\ \sharp*\ xvec$›
have $\Psi \rhd P \longmapsto M(\!|\nu*((q \cdot xvec)@(p \cdot yvec))|\!)\langle((q@p) \cdot N)\rangle \prec ((q@p) \cdot P')$
by(*simp add*: *eqvts pt2*[*OF pt-name-inst*] *del*: *freshAlphaPerm*)
then have $([(x, y)] \cdot \Psi) \rhd ([(x, y)] \cdot P) \longmapsto [(x, y)] \cdot (M(\!|\nu*((q \cdot xvec)@(p \cdot yvec))|\!)\langle((q@p) \cdot N)\rangle \prec ((q@p) \cdot P'))$
by(*rule semantics.eqvt*)
with ‹$x\ \sharp\ \Psi$› ‹$y\ \sharp\ \Psi$› ‹$x\ \sharp\ M$› ‹$y\ \sharp\ M$› ‹$x\ \sharp\ xvec$› ‹$y\ \sharp\ xvec$› ‹$x\ \sharp\ (q \cdot xvec)$› ‹$y\ \sharp\ (q \cdot xvec)$› ›‹$x\ \sharp\ yvec$› ‹$y\ \sharp\ yvec$› ‹$x\ \sharp\ (p \cdot yvec)$› ‹$y\ \sharp\ (p \cdot yvec)$› *Sp Sq*
show *?thesis*
apply(*simp add*: *eqvts pt2*[*OF pt-name-inst*])
by(*subst perm-compose*[*of q*], *simp*)+
qed
moreover from ‹$x \in supp\ N$› **have** $((q@(x, y)\#p) \cdot x) \in ((q@(x, y)\#p) \cdot (supp\ N))$
by(*simp add*: *pt-set-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$x\ \sharp\ xvec$› ‹$x\ \sharp\ yvec$› ‹$x\ \sharp\ (q \cdot xvec)$› ‹$x\ \sharp\ (p \cdot yvec)$› ‹$y\ \sharp\ xvec$› ‹$y\ \sharp\ (q \cdot xvec)$› *Sp Sq*
have $y \in supp((q@(x, y)\#p) \cdot N)$ **by**(*simp add*: *pt2*[*OF pt-name-inst*] *calc-atm eqvts*)
moreover from ‹*distinct xvec*› **have** *distinct*$(q \cdot xvec)$ **by** *simp*
moreover from ‹*distinct yvec*› **have** *distinct*$(p \cdot yvec)$ **by** *simp*
moreover note ‹$x\ \sharp\ (q \cdot xvec)$› ‹$x\ \sharp\ (p \cdot yvec)$› ‹$x\ \sharp\ M$› ‹$x\ \sharp\ \Psi$›
‹$(q \cdot xvec)\ \sharp*\ \Psi$› ‹$(q \cdot xvec)\ \sharp*\ P$› ‹$(q \cdot xvec)\ \sharp*\ M$› ‹$(q \cdot xvec)\ \sharp*\ (p \cdot yvec)$›
‹$(p \cdot yvec)\ \sharp*\ \Psi$› ‹$(p \cdot yvec)\ \sharp*\ P$› ‹$(p \cdot yvec)\ \sharp*\ M$› ‹$y\ \sharp\ (q \cdot xvec)$› ‹$y\ \sharp\ (p \cdot yvec)$› ‹$y\ \sharp\ M$› ‹$y\ \sharp\ C$› ‹$y\ \sharp\ \Psi$›
‹$(p \cdot yvec)\ \sharp*\ C$› ‹$(q \cdot xvec)\ \sharp*\ C$›
ultimately show *Prop* **by** − (*rule rOpen*, (*assumption* | *simp*)+)
next
case(*cBrOpen P M xvec yvec N P' x*)
from ‹$\Psi \rhd P \longmapsto ¡M(\!|\nu*(xvec@yvec)|\!)\langle N\rangle \prec P'$› **have** *distinct*$(xvec@yvec)$ **by**(*force dest*: *boundOutputDistinct*)
then have *xvec* $\sharp*$ *yvec* **by**(*induct xvec*) *auto*
obtain *p* **where** $(p \cdot yvec)\ \sharp*\ \Psi$ **and** $(p \cdot yvec)\ \sharp*\ P$ **and** $(p \cdot yvec)\ \sharp*\ M$

and $(p \cdot yvec) \sharp* yvec$ **and** $(p \cdot yvec) \sharp* N$ **and** $(p \cdot yvec) \sharp* P'$
and $x \sharp (p \cdot yvec)$ **and** $(p \cdot yvec) \sharp* xvec$
and $(p \cdot yvec) \sharp* C$ **and** *Sp*: $(set\ p) \subseteq (set\ yvec) \times (set(p \cdot yvec))$
by(*rule name-list-avoiding*[**where** *xvec*=*yvec* **and** *c*=(Ψ, P, M, *xvec*, *yvec*, N, P', x, C)])
(*auto simp add*: *eqvts fresh-star-prod*)
obtain q **where** $(q \cdot xvec) \sharp* \Psi$ **and** $(q \cdot xvec) \sharp* P$ **and** $(q \cdot xvec) \sharp* M$
and $(q \cdot xvec) \sharp* xvec$ **and** $(q \cdot xvec) \sharp* N$ **and** $(q \cdot xvec) \sharp* P'$
and $x \sharp (q \cdot xvec)$ **and** $(q \cdot xvec) \sharp* yvec$
and $(q \cdot xvec) \sharp* p$ **and** $(q \cdot xvec) \sharp* (p \cdot yvec)$
and $(q \cdot xvec) \sharp* C$ **and** *Sq*: $(set\ q) \subseteq (set\ xvec) \times (set(q \cdot xvec))$
by(*rule name-list-avoiding*[**where** *xvec*=*xvec* **and** *c*=(Ψ, P, M, *xvec*, *yvec*, $p \cdot yvec$, N, P', x, p, C)])
(*auto simp add*: *eqvts fresh-star-prod*)
obtain *y*::*name* **where** $y \sharp P$ **and** $y \sharp C$ **and** $y \sharp xvec$ **and** $y \sharp yvec$ **and** $y \neq x$ **and** $y \sharp N$
and $y \sharp (q \cdot xvec)$ **and** $y \sharp (p \cdot yvec)$ **and** $y \sharp M$ **and** $y \sharp \Psi$ **and** $y \sharp P'$
by(*generate-fresh name*) (*auto simp add*: *freshChainSimps*)
from ‹$cP = (\!|\nu x|\!)P$› ‹$y \sharp P$› **have** $cP = (\!|\nu y|\!)([(x, y)] \cdot P)$ **by**(*simp add*: *alphaRes*)
moreover have $cRs = ¡M(\!|\nu*((q \cdot xvec)@y\#(p \cdot yvec))|\!)\langle((q@(x, y)\#p) \cdot N)\rangle \prec ((q@(x, y)\#p) \cdot P')$
proof −
note ‹$cRs = ¡M(\!|\nu*(xvec@x\#yvec)|\!)\langle N\rangle \prec P'$›
moreover have $(\!|\nu*(xvec@x\#yvec)|\!)N \prec' P' = (\!|\nu*xvec|\!)((\!|\nu x|\!)((\!|\nu*yvec|\!)N \prec' P'))$ **by**(*simp add*: *boundOutputApp*)
moreover from ‹$(p \cdot yvec) \sharp* N$› ‹$(p \cdot yvec) \sharp* P'$› *Sp* **have** $\ldots = (\!|\nu*xvec|\!)((\!|\nu x|\!)((\!|\nu*(p \cdot yvec)|\!)(p \cdot N) \prec' (p \cdot P')))$
by(*simp add*: *boundOutputChainAlpha''*)
moreover with ‹$y \sharp N$› ‹$y \sharp P'$› ‹$y \sharp (p \cdot yvec)$› ‹$y \sharp yvec$› ‹$x \sharp yvec$› ‹$x \sharp (p \cdot yvec)$› *Sp*
moreover have $\ldots = (\!|\nu*xvec|\!)((\!|\nu y|\!)((\!|\nu*(p \cdot yvec)|\!)(([(x, y)] \cdot p \cdot N) \prec' ([(x, y)] \cdot p \cdot P'))))$
by(*subst alphaBoundOutput*[**where** *y*=*y*]) (*simp add*: *freshChainSimps eqvts*)+
moreover then have $\ldots = (\!|\nu*xvec|\!)((\!|\nu y|\!)((\!|\nu*(p \cdot yvec)|\!)((((x, y)\#p) \cdot N) \prec' (((x, y)\#p) \cdot P'))))$
by *simp*
moreover from ‹$(q \cdot xvec) \sharp* N$› ‹$(q \cdot xvec) \sharp* P'$› ‹$xvec \sharp* yvec$› ‹$(p \cdot yvec) \sharp* xvec$› ‹$(q \cdot xvec) \sharp* yvec$› ‹$(q \cdot xvec) \sharp* (p \cdot yvec)$›
‹$y \sharp xvec$› ‹$y \sharp (q \cdot xvec)$› ‹$x \sharp xvec$› ‹$x \sharp (q \cdot xvec)$› *Sp Sq*
have $\ldots = (\!|\nu*(q \cdot xvec)|\!)((\!|\nu y|\!)((\!|\nu*(p \cdot yvec)|\!)((q \cdot ((x, y)\#p) \cdot N) \prec' (q \cdot ((x, y)\#p) \cdot P'))))$
apply(*subst boundOutputChainAlpha*[**where** p=q **and** *xvec*=*xvec* **and** *yvec*=*xvec*])
defer
apply *assumption*
apply *simp*
apply(*simp add*: *eqvts*)
apply(*simp add*: *eqvts*)
apply(*simp add*: *boundOutputFreshSet*(*4*))
apply(*rule conjI*)

apply(*simp add*: *freshChainSimps*)
apply(*simp add*: *freshChainSimps*)
done
moreover then have ... = $(\!|\nu*(q \cdot xvec@y\#(p \cdot yvec))|\!)((q@(x,\ y)\#p) \cdot N) \prec' ((q@(x,\ y)\#p) \cdot P')$
by(*simp only*: *pt2*[*OF pt-name-inst*] *boundOutputApp BOresChain.simps*)
ultimately show *?thesis*
by(*simp only*: *residualInject*)
qed
moreover have $\Psi \rhd ([(x,\ y)] \cdot P) \longmapsto ¡M(\!|\nu*((q \cdot xvec)@(p \cdot yvec))|\!)\langle((q@(x,\ y)\#p) \cdot N)\rangle \prec ((q@(x,\ y)\#p) \cdot P')$
proof −
note‹$\Psi \rhd P \longmapsto ¡M(\!|\nu*(xvec@yvec)|\!)\langle N\rangle \prec P'$›
moreover from ‹$(p \cdot yvec) \sharp* N$› ‹$(q \cdot xvec) \sharp* N$› ‹$xvec \sharp* yvec$› ‹$(q \cdot xvec) \sharp* yvec$› ‹$(q \cdot xvec) \sharp* (p \cdot yvec)$› ‹$(p \cdot yvec) \sharp* xvec$› *Sp Sq*
have $((q@p) \cdot (xvec\ @\ yvec)) \sharp* N$ **apply**(*simp only*: *eqvts*) **apply**(*simp only*: *pt2*[*OF pt-name-inst*])
by *simp*
moreover from ‹$(p \cdot yvec) \sharp* P'$› ‹$(q \cdot xvec) \sharp* P'$› ‹$xvec \sharp* yvec$› ‹$(q \cdot xvec) \sharp* yvec$› ‹$(q \cdot xvec) \sharp* (p \cdot yvec)$› ‹$(p \cdot yvec) \sharp* xvec$› *Sp Sq*
have $((q@p) \cdot (xvec\ @\ yvec)) \sharp* P'$ **by**(*simp del*: *freshAlphaPerm add*: *eqvts pt2*[*OF pt-name-inst*])
moreover from *Sp Sq* ‹$xvec \sharp* yvec$› ‹$(q \cdot xvec) \sharp* yvec$› ‹$(q \cdot xvec) \sharp* (p \cdot yvec)$› ‹$(p \cdot yvec) \sharp* xvec$›
have *Spq*: $set(q@p) \subseteq set(xvec@yvec) \times set((q@p) \cdot (xvec@yvec))$
by(*simp add*: *pt2*[*OF pt-name-inst*] *eqvts*) *blast*
ultimately have $\Psi \rhd P \longmapsto ¡M(\!|\nu*((q@p) \cdot (xvec@yvec))|\!)\langle((q@p) \cdot N)\rangle \prec ((q@p) \cdot P')$
apply(*simp only*: *residualInject*)
by(*erule rev-mp*) (*subst boundOutputChainAlpha*, *auto*)
with *Sp Sq* ‹$xvec \sharp* yvec$› ‹$(q \cdot xvec) \sharp* yvec$› ‹$(q \cdot xvec) \sharp* (p \cdot yvec)$› ‹$(p \cdot yvec) \sharp* xvec$›
have $\Psi \rhd P \longmapsto ¡M(\!|\nu*((q \cdot xvec)@(p \cdot yvec))|\!)\langle((q@p) \cdot N)\rangle \prec ((q@p) \cdot P')$
by(*simp add*: *eqvts pt2*[*OF pt-name-inst*] *del*: *freshAlphaPerm*)
then have $([(x,\ y)] \cdot \Psi) \rhd ([(x,\ y)] \cdot P) \longmapsto [(x,\ y)] \cdot (¡M(\!|\nu*((q \cdot xvec)@(p \cdot yvec))|\!)\langle((q@p) \cdot N)\rangle \prec ((q@p) \cdot P'))$
by(*rule semantics.eqvt*)
with ‹$x \sharp \Psi$› ‹$y \sharp \Psi$› ‹$x \sharp M$› ‹$y \sharp M$› ‹$x \sharp xvec$› ‹$y \sharp xvec$› ‹$x \sharp (q \cdot xvec)$› ‹$y \sharp (q \cdot xvec)$› ›‹$x \sharp yvec$› ‹$y \sharp yvec$› ‹$x \sharp (p \cdot yvec)$› ‹$y \sharp (p \cdot yvec)$› *Sp Sq*
show *?thesis*
apply(*simp add*: *eqvts pt2*[*OF pt-name-inst*])
by(*subst perm-compose*[*of q*], *simp*)+
qed
moreover from ‹$x \in supp\ N$› **have** $((q@(x,\ y)\#p) \cdot x) \in ((q@(x,\ y)\#p) \cdot (supp\ N))$
by(*simp add*: *pt-set-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$x \sharp xvec$› ‹$x \sharp yvec$› ‹$x \sharp (q \cdot xvec)$› ‹$x \sharp (p \cdot yvec)$› ‹$y \sharp xvec$› ‹$y \sharp (q \cdot xvec)$› *Sp Sq*
have $y \in supp((q@(x,\ y)\#p) \cdot N)$ **by**(*simp add*: *pt2*[*OF pt-name-inst*] *calc-atm*

eqvts)
moreover from ‹*distinct xvec*› **have** *distinct*(*q* ∙ *xvec*) **by** *simp*
moreover from ‹*distinct yvec*› **have** *distinct*(*p* ∙ *yvec*) **by** *simp*
moreover note ‹*x* ♯ (*q* ∙ *xvec*)› ‹*x* ♯ (*p* ∙ *yvec*)› ‹*x* ♯ *M*› ‹*x* ♯ Ψ›
‹(*q* ∙ *xvec*) ♯* Ψ› ‹(*q* ∙ *xvec*) ♯* *P*› ‹(*q* ∙ *xvec*) ♯* *M*› ‹(*q* ∙ *xvec*) ♯* (*p* ∙ *yvec*)›
‹(*p* ∙ *yvec*) ♯* Ψ› ‹(*p* ∙ *yvec*) ♯* *P*› ‹(*p* ∙ *yvec*) ♯* *M*› ‹*y* ♯ (*q* ∙ *xvec*)› ‹*y* ♯ (*p* ∙ *yvec*)› ‹*y* ♯ *M*› ‹*y* ♯ *C*› ‹*y* ♯ Ψ›
‹(*p* ∙ *yvec*) ♯* *C*› ‹(*q* ∙ *xvec*) ♯* *C*›
ultimately show *Prop* **by** − (*rule rBrOpen*, (*assumption* | *simp*)+)
next
case(*cScope P* α *P*′ *x*)
obtain *p*::*name prm* **where** (*bn*(*p* ∙ α)) ♯* Ψ **and** (*bn*(*p* ∙ α)) ♯* *P*
and (*bn*(*p* ∙ α)) ♯* α **and** (*bn*(*p* ∙ α)) ♯* *P*′ **and** *x* ♯ *bn*(*p* ∙ α)
and *distinctPerm p*
and (*bn*(*p* ∙ α)) ♯* *C* **and** *Sp*: (*set p*) ⊆ *set*(*bn* α) × (*set*(*bn*(*p* ∙ α)))
by(*rule name-list-avoiding*[**where** *xvec*=*bn* α **and** *c*=(Ψ, *P*, α, *x*, *P*′, *C*)])
(*auto simp add*: *eqvts*)
obtain *y*::*name* **where** *y* ♯ Ψ **and** *y* ♯ *P* **and** *y* ♯ (*p* ∙ *P*′) **and** *y* ♯ (*p* ∙ α) **and** *y* ♯ *C*
by(*generate-fresh name*) (*auto simp add*: *freshChainSimps simp del*: *actionFresh*)
from ‹*bn* α ♯* *subject* α› ‹*distinctPerm p*› **have** *bn*(*p* ∙ α) ♯* *subject*(*p* ∙ α)
by(*subst fresh-star-bij*[*symmetric*, *of* - - *p*]) (*simp add*: *eqvts*)
from ‹*distinct*(*bn* α)› ‹*distinctPerm p*› **have** *distinct*(*bn*(*p* ∙ α))
by(*subst distinctClosed*[*symmetric*, *of* - *p*]) (*simp add*: *eqvts*)
from ‹*x* ♯ α› ‹*x* ♯ (*bn*(*p* ∙ α))› ‹*distinctPerm p*› *Sp* **have** *x* ♯ (*p* ∙ α)
by(*subst fresh-bij*[*symmetric*, *of* - - *p*]) (*simp add*: *eqvts freshChainSimps*)

from ‹*cP* = ⟪ν*x*⟫*P*› ‹*y* ♯ *P*› **have** *cP* = ⟪ν*y*⟫([(*x*, *y*)] ∙ *P*) **by**(*simp add*: *alphaRes*)
moreover from ‹*cRs* = α ≺ ⟪ν*x*⟫*P*′› ‹*bn* α ♯* *subject* α› ‹(*bn*(*p* ∙ α)) ♯* α› ‹*x* ♯ *bn*(*p* ∙ α)› ‹(*bn*(*p* ∙ α)) ♯* *P*′› ‹*x* ♯ α› *Sp*
have *cRs* = (*p* ∙ α) ≺ ⟪ν*x*⟫(*p* ∙ *P*′)
by(*force simp add*: *residualAlpha*)
with ‹*y* ♯ (*p* ∙ *P*′)› **have** *cRs* = (*p* ∙ α) ≺ ⟪ν*y*⟫([(*x*, *y*)] ∙ *p* ∙ *P*′)
by(*simp add*: *alphaRes*)
moreover from ‹Ψ ▷ *P* ⟼α ≺ *P*′› ‹*bn* α ♯* *subject* α› ‹(*bn*(*p* ∙ α)) ♯* α› ‹(*bn*(*p* ∙ α)) ♯* *P*′› *Sp*
have Ψ ▷ *P* ⟼(*p* ∙ α) ≺ (*p* ∙ *P*′) **by**(*force simp add*: *residualAlpha*)
then have([(*x*, *y*)] ∙ Ψ) ▷ ([(*x*, *y*)] ∙ *P*) ⟼[(*x*, *y*)] ∙ ((*p* ∙ α) ≺ (*p* ∙ *P*′))
by(*rule eqvts*)
with ‹*x* ♯ Ψ› ‹*y* ♯ Ψ› ‹*y* ♯ (*p* ∙ α)› ‹*x* ♯ (*p* ∙ α)› *Sp* ‹*distinctPerm p*›
have Ψ ▷ ([(*x*, *y*)] ∙ *P*) ⟼(*p* ∙ α) ≺ ([(*x*, *y*)] ∙ *p* ∙ *P*′)
by(*simp add*: *eqvts*)
moreover from ‹*bn*(*p* ∙ α) ♯* *P*› ‹*y* ♯ (*p* ∙ α)› ‹*y* ♯ *P*› **have** *bn*(*p* ∙ α) ♯* ([(*x*, *y*)] ∙ *P*)
by(*auto simp add*: *fresh-star-def fresh-left calc-atm*) (*simp add*: *fresh-def name-list-supp*)
moreover from ‹*distinct*(*bn* α)› **have** *distinct*(*p* ∙ *bn* α) **by** *simp*
then have *distinct*(*bn*(*p* ∙ α)) **by**(*simp add*: *eqvts*)
ultimately show *?thesis*

using ‹$y \sharp \Psi$› ‹$y \sharp (p \cdot \alpha)$› ‹$y \sharp C$› ‹$bn(p \cdot \alpha) \sharp* \Psi$› ‹$bn(p \cdot \alpha) \sharp* subject(p \cdot \alpha)$› ‹$bn(p \cdot \alpha) \sharp* C$›
by(*metis rScope*)
next
case(*Bang P*)
then show *?thesis* **by**(*metis rBang*)
qed

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*
apply(*finite-guess*)+
apply(*rule TrueI*)+
by(*fresh-guess add*: *fresh-nat*)+

nominal-primrec *boundOutputLength* :: ('*a*, '*b*, '*c*) *boundOutput* ⇒ *nat*
where
boundOutputLength (*BOut M P*) = *0*
| *boundOutputLength* (*BStep x B*) = (*boundOutputLength B*) + *1*
apply(*finite-guess*)+
apply(*rule TrueI*)+
by(*fresh-guess add*: *fresh-nat*)+

nominal-primrec *residualLength* :: ('*a*, '*b*, '*c*) *residual* ⇒ *nat*
where
residualLength (*RIn M N P*) = *0*
| *residualLength* (*RBrIn M N P*) = *0*
| *residualLength* (*ROut M B*) = *boundOutputLength B*
| *residualLength* (*RBrOut M B*) = *boundOutputLength B*
| *residualLength* (*RTau P*) = *0*
by(*rule TrueI*)+

lemma *inputLengthProc*[*simp*]:
shows *inputLength*($M(\!|\lambda * xvec\ N|\!).P$) = *length xvec*
by(*induct xvec*) *auto*

lemma *boundOutputLengthSimp*[*simp*]:
shows *residualLength*($M(\!|\nu * xvec|\!)\langle N\rangle \prec P$) = *length xvec*
and *residualLength*(¡$M(\!|\nu * xvec|\!)\langle N\rangle \prec P$) = *length xvec*
by(*induct xvec*) (*auto simp add*: *residualInject*)

lemma *boundOuputLengthSimp2*[*simp*]:
shows *residualLength*($\alpha \prec P$) = *length*(*bn* α)
by(*nominal-induct* α *rule*: *action.strong-induct*, *auto*) (*auto simp add*: *residual-Inject*)

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 −
assume ⋀*p*. ⟦*set p* ⊆ *set xvec* × *set* (*p* ∙ *xvec*); *distinctPerm p*; *yvec* = *p* ∙ *xvec*⟧ ⟹ *thesis*
moreover obtain *n* **where** *n* = *length xvec* **by** *auto*
with *assms* **have** ∃ *p*. (*set p*) ⊆ (*set xvec*) × *set* (*yvec*) ∧ *distinctPerm p* ∧ *yvec* = *p* ∙ *xvec*
proof(*induct n arbitrary*: *xvec yvec*)
case(*0 xvec yvec*)
then show *?case* **by** *simp*
next
case(*Suc n xvec yvec*)
from ‹*Suc n* = *length xvec*›
obtain *x xvec'* **where** *xvec* = *x*#*xvec'* **and** *length xvec'* = *n*
by(*cases xvec*) *auto*
from ‹*length xvec* = *length yvec*› ‹*xvec* = *x* # *xvec'*›
obtain *y yvec'* **where** *length xvec'* = *length yvec'* **and** *yvec* = *y*#*yvec'*
by(*cases yvec*) *auto*
from ‹*xvec* = *x*#*xvec'*› ‹*yvec*=*y*#*yvec'*› ‹*xvec* ♯* *yvec*›
have *x* ≠ *y* **and** *xvec'* ♯* *yvec'* **and** *x* ♯ *yvec'* **and** *y* ♯ *xvec'*
by(*auto simp add*: *fresh-list-cons*)
from ‹*distinct xvec*› ‹*distinct yvec*› ‹*xvec*=*x*#*xvec'*› ‹*yvec*=*y*#*yvec'*› **have** *x* ♯

xvec' **and** *y* ♯ *yvec'* **and** *distinct xvec'* **and** *distinct yvec'*
by *simp+*
from ‹*Suc n = length xvec*› ‹*xvec=x#xvec'*› **have** *n = length xvec'* **by** *simp*
with ‹*length xvec' = length yvec'*› ‹*xvec' ♯* yvec'*› ‹*distinct xvec'*› ‹*distinct yvec'*›
obtain *p* **where** *S*: *set p ⊆ set xvec' × set yvec'* **and** *distinctPerm p* **and** *yvec'* = *p* ∙ *xvec'*
by − (*drule Suc*,*auto*)
from *S* **have** *set*((*x*, *y*)#*p*) ⊆ *set*(*x*#*xvec'*) × *set*(*y*#*yvec'*) **by** *auto*
moreover from ‹*x* ♯ *xvec'*› ‹*x* ♯ *yvec'*› ‹*y* ♯ *xvec'*› ‹*y* ♯ *yvec'*› *S* **have** *x* ♯ *p* **and** *y* ♯ *p*
apply(*induct p*)
by(*clarsimp simp add*: *fresh-list-nil fresh-list-cons fresh-prod name-list-supp*; *force simp add*: *fresh-def*)+

with *S* ‹*distinctPerm p*› ‹*x* ≠ *y*› **have** *distinctPerm*((*x*, *y*)#*p*) **by** *auto*
moreover from ‹*yvec' = p* ∙ *xvec'*› ‹*x* ♯ *p*› ‹*y* ♯ *p*› ‹*x* ♯ *xvec'*› ‹*y* ♯ *xvec'*› **have** (*y*#*yvec'*) = ((*x*, *y*)#*p*) ∙ (*x*#*xvec'*)
by(*simp add*: *calc-atm freshChainSimps*)
ultimately show *?case* **using** ‹*xvec=x#xvec'*› ‹*yvec=y#yvec'*›
by *blast*
qed
ultimately show *?thesis* **by** *blast*
qed

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

shows (*set xvec* ∩ *set yvec* = {}) = *xvec ♯** *yvec*
by(*auto simp add*: *fresh-star-def name-list-supp fresh-def*)

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*
using *assms*
by(*induct xvec arbitrary*: *yvec y*) *auto*

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 −
assume ⋀*xvec1 x xvec2*.
⟦*xvec = xvec1 @ x # xvec2*; *length xvec1 = length yvec*;
length xvec2 = length zvec⟧
⟹ *thesis*
moreover from *assms* **have** ∃*xvec1 x xvec2*. *xvec=xvec1@x#xvec2* ∧ *length xvec1 = length yvec* ∧ *length xvec2 = length zvec*
apply −
apply(*rule exI*[**where** *x=take* (*length yvec*) *xvec*])
apply(*rule exI*[**where** *x=hd*(*drop* (*length yvec*) *xvec*)])
apply(*rule exI*[**where** *x=tl*(*drop* (*length yvec*) *xvec*)])
by *auto*
ultimately show *?thesis* **by** *blast*
qed

lemma *semanticsCases*[*consumes 19*, *case-names cInput cBrInput cOutput cBrOutput cCase cPar1 cPar2 cComm1 cComm2 cBrMerge cBrComm1 cBrComm2 cBrClose cOpen cBrOpen cScope cBang*]:
fixes Ψ :: *'b*
and *cP* :: (*'a*, *'b*, *'c*) *psi*
and *cRs* :: (*'a*, *'b*, *'c*) *residual*
and *C* :: *'f*::*fs-name*
and *x1* :: *name*
and *x2* :: *name*
and *x3* :: *name*
and *x4* :: *name*
and *xvec1* :: *name list*
and *xvec2* :: *name list*
and *xvec3* :: *name list*
and *xvec4* :: *name list*
and *xvec5* :: *name list*
and *xvec6* :: *name list*
and *xvec7* :: *name list*
and *xvec8* :: *name list*
and *xvec9* :: *name list*

assumes Ψ ▷ *cP* ⟼ *cRs*
and *length xvec1 = inputLength cP* **and** *distinct xvec1*
and *length xvec6 = inputLength cP* **and** *distinct xvec6*
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 *length xvec7 = residualLength cRs* **and** *distinct xvec7*
and *length xvec8 = residualLength cRs* **and** *distinct xvec8*
and *length xvec9 = residualLength cRs* **and** *distinct xvec9*
and *rInput*: ⋀*M K N Tvec P*. (⟦*xvec1* ♯* Ψ; *xvec1* ♯* *cP*; *xvec1* ♯* *cRs*⟧ ⟹

$cP = M(\!|\lambda{*}xvec1\ N|\!).P \wedge \ \ cRs = K(\!|(N[xvec1::=Tvec])|\!) \prec P[xvec1::=Tvec] \wedge$
$\Psi \vdash M \leftrightarrow K \wedge distinct\ xvec1 \wedge set\ xvec1 \subseteq$
$supp\ N \wedge length\ xvec1{=}length\ Tvec \wedge$
$xvec1\ \sharp{*}\ Tvec \wedge xvec1\ \sharp{*}\ \Psi \wedge xvec1\ \sharp{*}\ M \wedge$
$xvec1\ \sharp{*}\ K) \Longrightarrow Prop$
and $rBrInput$: $\bigwedge M\ K\ N\ Tvec\ P.\ ([\![xvec6\ \sharp{*}\ \Psi;\ xvec6\ \sharp{*}\ cP;\ xvec6\ \sharp{*}\ cRs]\!] \Longrightarrow$
$cP = M(\!|\lambda{*}xvec6\ N|\!).P \wedge \ \ cRs = ¿K(\!|(N[xvec6::=Tvec])|\!) \prec P[xvec6::=Tvec] \wedge$
$\Psi \vdash K \succeq M \wedge distinct\ xvec6 \wedge set\ xvec6 \subseteq$
$supp\ N \wedge length\ xvec6{=}length\ Tvec \wedge$
$xvec6\ \sharp{*}\ Tvec \wedge xvec6\ \sharp{*}\ \Psi \wedge xvec6\ \sharp{*}\ M \wedge$
$xvec6\ \sharp{*}\ K) \Longrightarrow Prop$
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 $rBrOutput$: $\bigwedge M\ K\ N\ P.\ [\![cP = M\langle N\rangle.P;\ cRs = ¡K\langle N\rangle \prec P;\ \Psi \vdash M \preceq$
$K]\!] \Longrightarrow Prop$
and $rCase$: $\bigwedge Cs\ P\ \varphi.\ [\![cP = Cases\ Cs;\ \Psi \rhd P \longmapsto cRs;\ (\varphi,\ P) \in set\ Cs;\ \Psi \vdash$
$\varphi;\ guarded\ P]\!] \Longrightarrow Prop$
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 $rComm1$: $\bigwedge \Psi_Q\ P\ M\ N\ P'\ A_P\ \Psi_P\ Q\ K\ xvec\ Q'\ A_Q.$
$[\![cP = P \parallel Q;\ cRs = \tau \prec (\!|\nu{*}xvec|\!)P' \parallel Q';$
$\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|N|\!) \prec P';\ extractFrame\ P = \langle A_P,\ \Psi_P\rangle;$
$distinct\ A_P;$
$\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|\nu{*}xvec|\!)\langle N\rangle \prec Q';\ extractFrame\ Q = \langle A_Q,$
$\Psi_Q\rangle;\ distinct\ A_Q;$
$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K;\ A_P\ \sharp{*}\ \Psi;\ A_P\ \sharp{*}\ \Psi_Q;\ A_P\ \sharp{*}\ P;\ A_P\ \sharp{*}$
$M;\ A_P\ \sharp{*}\ N;$
$A_P\ \sharp{*}\ P';\ A_P\ \sharp{*}\ Q;\ A_P\ \sharp{*}\ Q';\ A_P\ \sharp{*}\ A_Q;\ A_P\ \sharp{*}\ xvec;\ A_Q\ \sharp{*}\ \Psi;$
$A_Q\ \sharp{*}\ \Psi_P;$
$A_Q\ \sharp{*}\ P;\ A_Q\ \sharp{*}\ K;\ A_Q\ \sharp{*}\ N;\ A_Q\ \sharp{*}\ P';\ A_Q\ \sharp{*}\ Q;\ A_Q\ \sharp{*}\ Q';\ A_Q$
$\sharp{*}\ xvec;$
$xvec\ \sharp{*}\ \Psi;\ xvec\ \sharp{*}\ \Psi_P;\ xvec\ \sharp{*}\ \Psi_Q;\ xvec\ \sharp{*}\ P;\ xvec\ \sharp{*}\ M;\ xvec\ \sharp{*}$
$Q;$
$xvec\ \sharp{*}\ K;\ A_P\ \sharp{*}\ C;\ A_Q\ \sharp{*}\ C;\ xvec\ \sharp{*}\ C;\ distinct\ xvec]\!] \Longrightarrow Prop$
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 *rBrMerge*: $\bigwedge \Psi_Q\ P\ M\ N\ P'\ A_P\ \Psi_P\ Q\ Q'\ A_Q$.
$[\![cP = (P \parallel Q)$; $cRs = ¿M(\!|N|\!) \prec (P' \parallel Q')$;
$\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$; *extractFrame* $P = \langle A_P, \Psi_P\rangle$;
distinct A_P;
$\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$; *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$;
distinct A_Q;
$A_P \sharp\!* \Psi$; $A_P \sharp\!* \Psi_Q$; $A_P \sharp\!* P$; $A_P \sharp\!* N$; $A_P \sharp\!* P'$;
$A_P \sharp\!* Q$; $A_P \sharp\!* Q'$; $A_P \sharp\!* A_Q$; $A_P \sharp\!* M$; $A_Q \sharp\!* M$;
$A_Q \sharp\!* \Psi$; $A_Q \sharp\!* \Psi_P$; $A_Q \sharp\!* P$; $A_Q \sharp\!* N$; $A_Q \sharp\!* P'$;
$A_Q \sharp\!* Q$; $A_Q \sharp\!* Q'$; $A_P \sharp\!* C$; $A_Q \sharp\!* C$$]\!] \Longrightarrow$ *Prop*

and *rBrComm1*: $\bigwedge \Psi_Q\ P\ M\ N\ P'\ A_P\ \Psi_P\ Q\ Q'\ A_Q$.
$([\![xvec7 \sharp\!* \Psi$; $xvec7 \sharp\!* cP$; $xvec7 \sharp\!* cRs$$]\!] \Longrightarrow$
$cP = P \parallel Q \wedge cRs = ¡M(\!|\nu{*}xvec7|\!)\langle N\rangle \prec (P' \parallel Q') \wedge$
$\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P' \wedge$ *extractFrame* $P = \langle A_P, \Psi_P\rangle \wedge$
distinct $A_P \wedge$
$\Psi \otimes \Psi_P \rhd Q \longmapsto ¡M(\!|\nu{*}xvec7|\!)\langle N\rangle \prec Q' \wedge$ *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle \wedge$ *distinct* $A_Q \wedge$
$A_P \sharp\!* \Psi \wedge A_P \sharp\!* \Psi_Q \wedge A_P \sharp\!* P \wedge A_P \sharp\!* N \wedge$
$A_P \sharp\!* P' \wedge A_P \sharp\!* Q \wedge A_P \sharp\!* Q' \wedge A_P \sharp\!* A_Q \wedge A_P \sharp\!* xvec7 \wedge A_Q \sharp\!* \Psi \wedge A_Q \sharp\!* \Psi_P \wedge$
$A_Q \sharp\!* P \wedge A_Q \sharp\!* N \wedge A_Q \sharp\!* P' \wedge A_Q \sharp\!* Q \wedge A_Q \sharp\!* Q' \wedge A_Q \sharp\!* xvec7 \wedge$
$xvec7 \sharp\!* \Psi \wedge xvec7 \sharp\!* \Psi_P \wedge xvec7 \sharp\!* \Psi_Q \wedge xvec7 \sharp\!* P \wedge xvec7 \sharp\!* Q \wedge$
$A_P \sharp\!* M \wedge A_Q \sharp\!* M \wedge xvec7 \sharp\!* M \wedge$
$A_P \sharp\!* C \wedge A_Q \sharp\!* C \wedge$ *distinct* $xvec7) \Longrightarrow$ *Prop*

and *rBrComm2*: $\bigwedge \Psi_Q\ P\ M\ N\ P'\ A_P\ \Psi_P\ Q\ Q'\ A_Q$.
$([\![xvec8 \sharp\!* \Psi$; $xvec8 \sharp\!* cP$; $xvec8 \sharp\!* cRs$$]\!] \Longrightarrow$
$cP = P \parallel Q \wedge cRs = ¡M(\!|\nu{*}xvec8|\!)\langle N\rangle \prec (P' \parallel Q') \wedge$
$\Psi \otimes \Psi_Q \rhd P \longmapsto ¡M(\!|\nu{*}xvec8|\!)\langle N\rangle \prec P' \wedge$ *extractFrame* $P = \langle A_P, \Psi_P\rangle \wedge$ *distinct* $A_P \wedge$
$\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q' \wedge$ *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle \wedge$ *distinct* $A_Q \wedge$
$A_P \sharp\!* \Psi \wedge A_P \sharp\!* \Psi_Q \wedge A_P \sharp\!* P \wedge A_P \sharp\!* N \wedge$
$A_P \sharp\!* P' \wedge A_P \sharp\!* Q \wedge A_P \sharp\!* Q' \wedge A_P \sharp\!* A_Q \wedge A_P \sharp\!* xvec8 \wedge A_Q \sharp\!* \Psi \wedge A_Q \sharp\!* \Psi_P \wedge$
$A_Q \sharp\!* P \wedge A_Q \sharp\!* N \wedge A_Q \sharp\!* P' \wedge A_Q \sharp\!* Q \wedge A_Q \sharp\!* Q' \wedge A_Q \sharp\!*$

$xvec8 \wedge$
$xvec8 \sharp* \Psi \wedge xvec8 \sharp* \Psi_P \wedge xvec8 \sharp* \Psi_Q \wedge xvec8 \sharp* P \wedge xvec8 \sharp* Q \wedge$
$A_P \sharp* M \wedge A_Q \sharp* M \wedge xvec8 \sharp* M \wedge$
$A_P \sharp* C \wedge A_Q \sharp* C \wedge distinct\ xvec8) \Longrightarrow Prop$
and *rBrClose*: $\bigwedge P\ M\ N\ xvec\ P'.$
$([\![x3 \sharp \Psi; x3 \sharp cP; x3 \sharp cRs]\!] \Longrightarrow$
$cP = (\!|\nu x3|\!)P \wedge cRs = \tau \prec (\!|\nu x3|\!)((\!|\nu* xvec|\!)P') \wedge$
$x3 \in supp\ M \wedge$
$\Psi \rhd P \longmapsto {¡}M(\!|\nu* xvec|\!)\langle N\rangle \prec P' \wedge$
$distinct\ xvec \wedge xvec \sharp* \Psi \wedge xvec \sharp* P \wedge$
$xvec \sharp* M \wedge xvec \sharp* C \wedge$
$x3 \sharp \Psi \wedge x3 \sharp xvec) \Longrightarrow Prop$
and *rOpen*: $\bigwedge P\ M\ xvec\ y\ yvec\ N\ P'.$
$([\![xvec4 \sharp* \Psi; xvec4 \sharp* cP; xvec4 \sharp* cRs; x1 \sharp \Psi; x1 \sharp cP; x1 \sharp cRs; x1 \sharp xvec4]\!] \Longrightarrow$
$cP = (\!|\nu x1|\!)P \wedge cRs = M(\!|\nu*(xvec@x1\#yvec)|\!)\langle N\rangle \prec P' \wedge xvec4{=}xvec@y\#yvec \wedge$
$\Psi \rhd P \longmapsto M(\!|\nu*(xvec@yvec)|\!)\langle N\rangle \prec P' \wedge x1 \in supp\ N \wedge x1 \sharp xvec \wedge x1 \sharp yvec \wedge$
$distinct\ xvec \wedge distinct\ yvec \wedge xvec \sharp* \Psi \wedge xvec \sharp* P \wedge xvec \sharp* M \wedge xvec \sharp* yvec \wedge$
$yvec \sharp* \Psi \wedge yvec \sharp* P \wedge yvec \sharp* M) \Longrightarrow Prop$
and *rBrOpen*: $\bigwedge P\ M\ xvec\ y\ yvec\ N\ P'.$
$([\![xvec9 \sharp* \Psi; xvec9 \sharp* cP; xvec9 \sharp* cRs; x4 \sharp \Psi; x4 \sharp cP; x4 \sharp cRs; x4 \sharp xvec9]\!] \Longrightarrow$
$cP = (\!|\nu x4|\!)P \wedge cRs = {¡}M(\!|\nu*(xvec@x4\#yvec)|\!)\langle N\rangle \prec P' \wedge xvec9{=}xvec@y\#yvec \wedge$
$\Psi \rhd P \longmapsto {¡}M(\!|\nu*(xvec@yvec)|\!)\langle N\rangle \prec P' \wedge x4 \in supp\ N \wedge x4 \sharp xvec \wedge x4 \sharp yvec \wedge$
$distinct\ xvec \wedge distinct\ yvec \wedge xvec \sharp* \Psi \wedge xvec \sharp* P \wedge xvec \sharp* M \wedge xvec \sharp* yvec \wedge$
$yvec \sharp* \Psi \wedge yvec \sharp* P \wedge yvec \sharp* M) \Longrightarrow Prop$
and *rScope*: $\bigwedge P\ \alpha\ P'.$ $([\![xvec5 \sharp* \Psi; xvec5 \sharp* cP; xvec5 \sharp* cRs; x2 \sharp \Psi; x2 \sharp cP; x2 \sharp cRs; x2 \sharp xvec5]\!] \Longrightarrow$
$cP = (\!|\nu x2|\!)P \wedge cRs = \alpha \prec (\!|\nu x2|\!)P' \wedge\ xvec5 = bn\ \alpha \wedge$
$\Psi \rhd P \longmapsto \alpha \prec P' \wedge x2 \sharp \Psi \wedge x2 \sharp \alpha \wedge bn\ \alpha \sharp* subject\ \alpha \wedge distinct(bn\ \alpha)) \Longrightarrow Prop$
and *rBang*: $\bigwedge P.$ $[\![cP = !P;$
$\Psi \rhd P \parallel !P \longmapsto cRs; guarded\ P]\!] \Longrightarrow Prop$
shows *Prop*
using ‹$\Psi \rhd cP \longmapsto cRs$›
proof(*cases rule*: *semanticsCasesAux*[**where** *C*=(*xvec1*, *xvec2*, *xvec3*, *xvec4*, *xvec5*, *xvec6*, *xvec7*, *xvec8*, *xvec9*, *x1*, *x2*, *x3*, *x4*, *cP*, *cRs*, *C*)])
case(*cInput M K xvec N Tvec P*)
have *B*: $cP = M(\!|\lambda* xvec\ N|\!).P$ **and** *C*: $cRs = K(\!|(N[xvec::=Tvec])|\!) \prec (P[xvec::=Tvec])$
by *fact+*
from ‹$xvec \sharp* (xvec1, xvec2, xvec3, xvec4, xvec5, xvec6, xvec7, xvec8, xvec9, x1, x2, x3, x4, cP, cRs, C)$› **have** $xvec \sharp* xvec1$ **by** *simp*

from ‹*length xvec1 = inputLength cP*› *B* **have** *length xvec1 = length xvec*
by *simp*
then obtain *p* **where** *S*: *set p* ⊆ *set xvec* × *set*(*p* · *xvec*) **and** *distinctPerm p* **and** *xvec1 = p* · *xvec*
using ‹*xvec* ♯* *xvec1*› ‹*distinct xvec*› ‹*distinct xvec1*›
by − (*rule constructPerm*[**where** *xvec*=*xvec* **and** *yvec*=*xvec1*], *auto*)
show *?thesis*
proof(*rule rInput*[**where** *M*=*M* **and** *K*=*K* **and** *N = p* · *N* **and** *Tvec*=*Tvec* **and** *P*=*p* · *P*])
assume *xvec1* ♯* Ψ **and** *xvec1* ♯* *cP* **and** *xvec1* ♯* *cRs*
from *B* ‹*xvec* ♯* *xvec1*› ‹*xvec1* ♯* *cP*› **have** *xvec1* ♯* *N* **and** *xvec1* ♯* *P*
by(*auto simp add*: *fresh-star-def inputChainFresh name-list-supp*) (*auto simp add*: *fresh-def*)
moreover from ‹*cP = M*(|λ*xvec N|).*P*› *S* ‹*xvec1* ♯* *N*› ‹*xvec1* ♯* *P*› ‹*xvec1 = p* · *xvec*›
have *cP = M*(|λ*xvec1* (*p* · *N*)|).(*p* · *P*)
apply *simp*
by(*subst inputChainAlpha*) *auto*
moreover from ‹*cRs = K*(|(*N*[*xvec*::=*Tvec*])|) ≺ *P*[*xvec*::=*Tvec*]› *S* ‹*xvec1* ♯* *N*› ‹*xvec1* ♯* *P*› ‹*xvec1 = p* · *xvec*› ‹*length xvec = length Tvec*› ‹*distinctPerm p*›
have *cRs = K*(|((*p* · *N*)[*xvec1*::=*Tvec*])|) ≺ (*p* · *P*)[*xvec1*::=*Tvec*]
by(*simp add*: *renaming substTerm.renaming*)
moreover note ‹Ψ ⊢ *M* ↔ *K*›
moreover from ‹*distinct xvec*› ‹*xvec1 = p* · *xvec*› **have** *distinct xvec1* **by** *simp*
moreover from ‹*set xvec* ⊆ *supp N*› **have** (*p* · *set xvec*) ⊆ (*p* · (*supp N*))
by(*simp add*: *eqvts*)
with ‹*xvec1 = p* · *xvec*› **have** *set xvec1* ⊆ *supp*(*p* · *N*) **by**(*simp add*: *eqvts*)
moreover from ‹*length xvec = length Tvec*› ‹*xvec1 = p* · *xvec*› **have** *length xvec1 = length Tvec*
by *simp*

moreover from ‹*xvec1* ♯* *cRs*› *C* ‹*length xvec = length Tvec*› ‹*distinct xvec*› ‹*set xvec* ⊆ *supp N*›
have (*set xvec1*) ♯* *Tvec*
by − (*rule substTerm.subst3Chain*[**where** *T*=*N*], *auto*)
then have *xvec1* ♯* *Tvec* **by** *simp*
moreover from ‹*xvec* ♯* *Tvec*› **have** (*p* · *xvec*) ♯* (*p* · *Tvec*) **by**(*simp add*: *fresh-star-bij*)
with *S* ‹*xvec* ♯* *Tvec*› ‹*xvec1* ♯* *Tvec*› ‹*xvec1 = p* · *xvec*› **have** *xvec1* ♯* *Tvec* **by** *simp*
moreover note ‹*xvec1* ♯* Ψ›
moreover from ‹*xvec* ♯* *M*› **have** (*p* · *xvec*) ♯* (*p* · *M*) **by**(*simp add*: *fresh-star-bij*)
with *S* ‹*xvec* ♯* *M*› ‹*xvec1* ♯* *cP*› *B* ‹*xvec1 = p* · *xvec*› **have** *xvec1* ♯* *M* **by** *simp*
moreover from ‹*xvec* ♯* *K*› **have** (*p* · *xvec*) ♯* (*p* · *K*) **by**(*simp add*: *fresh-star-bij*)
with *S* ‹*xvec* ♯* *K*› ‹*xvec1* ♯* *cRs*› *C* ‹*xvec1 = p* · *xvec*› **have** *xvec1* ♯* *K* **by**

simp
ultimately show $cP = M(\!|\lambda * xvec1\ p \cdot N|\!).p \cdot P \wedge cRs = K(\!|(p \cdot N)[xvec1::=Tvec]|\!) \prec (p \cdot P)[xvec1::=Tvec] \wedge \Psi \vdash M \leftrightarrow K \wedge distinct\ xvec1 \wedge set\ xvec1 \subseteq supp\ (p \cdot N) \wedge length\ xvec1 = length\ Tvec \wedge xvec1 \sharp * Tvec \wedge xvec1 \sharp * \Psi \wedge xvec1 \sharp * M \wedge xvec1 \sharp * K$
by *blast*
qed
next
case(*cBrInput M K xvec N Tvec P*)
have *B*: $cP = M(\!|\lambda * xvec\ N|\!).P$ **and** *C*: $cRs = ¿K(\!|(N[xvec::=Tvec])|\!) \prec (P[xvec::=Tvec])$
by *fact+*
from ‹*xvec* $\sharp *$ (*xvec1*, *xvec2*, *xvec3*, *xvec4*, *xvec5*, *xvec6*, *xvec7*, *xvec8*, *xvec9*, *x1*, *x2*, *x3*, *x4*, *cP*, *cRs*, *C*)› **have** *xvec* $\sharp *$ *xvec6* **by** *simp*

from ‹*length xvec6* = *inputLength cP*› *B* **have** *length xvec6* = *length xvec*
by *simp*
then obtain *p* **where** *S*: *set p* $\subseteq$ *set xvec* $\times$ *set*(*p* $\cdot$ *xvec*) **and** *distinctPerm p* **and** *xvec6* = *p* $\cdot$ *xvec*
using ‹*xvec* $\sharp *$ *xvec6*› ‹*distinct xvec*› ‹*distinct xvec6*›
by − (*rule constructPerm*[**where** *xvec*=*xvec* **and** *yvec*=*xvec6*], *auto*)
show *?thesis*
proof(*rule rBrInput*[**where** *M*=*M* **and** *K*=*K* **and** *N* = *p* $\cdot$ *N* **and** *Tvec*=*Tvec* **and** *P*=*p* $\cdot$ *P*])
assume *xvec6* $\sharp *$ Ψ **and** *xvec6* $\sharp *$ *cP* **and** *xvec6* $\sharp *$ *cRs*
from *B* ‹*xvec* $\sharp *$ *xvec6*› ‹*xvec6* $\sharp *$ *cP*› **have** *xvec6* $\sharp *$ *N* **and** *xvec6* $\sharp *$ *P*
by(*auto simp add*: *fresh-star-def inputChainFresh name-list-supp*) (*auto simp add*: *fresh-def*)

moreover from ‹*cP* = *M*(|$\lambda *$*xvec N*|).*P*› *S* ‹*xvec6* $\sharp *$ *N*› ‹*xvec6* $\sharp *$ *P*› ‹*xvec6* = *p* $\cdot$ *xvec*›
have *cP* = *M*(|$\lambda *$*xvec6* (*p* $\cdot$ *N*)|).(*p* $\cdot$ *P*)
apply *simp*
by(*subst inputChainAlpha*) *auto*
moreover from ‹*cRs* = ¿*K*(|(*N*[*xvec*::=*Tvec*])|) $\prec$ *P*[*xvec*::=*Tvec*]› *S* ‹*xvec6* $\sharp *$ *N*› ‹*xvec6* $\sharp *$ *P*› ‹*xvec6* = *p* $\cdot$ *xvec*› ‹*length xvec* = *length Tvec*› ‹*distinctPerm p*›
have *cRs* = ¿*K*(|((*p* $\cdot$ *N*)[*xvec6*::=*Tvec*])|) $\prec$ (*p* $\cdot$ *P*)[*xvec6*::=*Tvec*]
by(*simp add*: *renaming substTerm.renaming*)
moreover note ‹$\Psi \vdash K \succeq M$›
moreover from ‹*distinct xvec*› ‹*xvec6* = *p* $\cdot$ *xvec*› **have** *distinct xvec6* **by** *simp*
moreover from ‹*set xvec* $\subseteq$ *supp N*› **have** (*p* $\cdot$ *set xvec*) $\subseteq$ (*p* $\cdot$ (*supp N*))
by(*simp add*: *eqvts*)
with ‹*xvec6* = *p* $\cdot$ *xvec*› **have** *set xvec6* $\subseteq$ *supp*(*p* $\cdot$ *N*) **by**(*simp add*: *eqvts*)
moreover from ‹*length xvec* = *length Tvec*› ‹*xvec6* = *p* $\cdot$ *xvec*› **have** *length xvec6* = *length Tvec*
by *simp*

moreover from ‹*xvec6* $\sharp *$ *cRs*› *C* ‹*length xvec* = *length Tvec*› ‹*distinct xvec*› ‹*set xvec* $\subseteq$ *supp N*›

have (*set xvec6*) $\sharp*$ *Tvec*
by − (*rule substTerm.subst3Chain*[**where** *T*=*N*], *auto*)
then have *xvec6* $\sharp*$ *Tvec* **by** *simp*
moreover from ‹*xvec* $\sharp*$ *Tvec*› **have** (*p* · *xvec*) $\sharp*$ (*p* · *Tvec*) **by**(*simp add*: *fresh-star-bij*)
with *S* ‹*xvec* $\sharp*$ *Tvec*› ‹*xvec6* $\sharp*$ *Tvec*› ‹*xvec6* = *p* · *xvec*› **have** *xvec6* $\sharp*$ *Tvec* **by** *simp*
moreover note ‹*xvec6* $\sharp*$ Ψ›
moreover from ‹*xvec* $\sharp*$ *M*› **have** (*p* · *xvec*) $\sharp*$ (*p* · *M*) **by**(*simp add*: *fresh-star-bij*)
with *S* ‹*xvec* $\sharp*$ *M*› ‹*xvec6* $\sharp*$ *cP*› *B* ‹*xvec6* = *p* · *xvec*› **have** *xvec6* $\sharp*$ *M* **by** *simp*
moreover from ‹*xvec* $\sharp*$ *K*› **have** (*p* · *xvec*) $\sharp*$ (*p* · *K*) **by**(*simp add*: *fresh-star-bij*)
with *S* ‹*xvec* $\sharp*$ *K*› ‹*xvec6* $\sharp*$ *cRs*› *C* ‹*xvec6* = *p* · *xvec*› **have** *xvec6* $\sharp*$ *K* **by** *simp*
ultimately show *cP* = *M*⟪λ*xvec6 *p* · *N*⟫.*p* · *P* ∧
cRs = ¿*K*⟪(*p* · *N*)[*xvec6*::=*Tvec*]⟫ ≺ (*p* · *P*)[*xvec6*::=*Tvec*] ∧
Ψ ⊢ *K* ↔ *M* ∧ *distinct xvec6* ∧ *set xvec6* ⊆ *supp* (*p* · *N*) ∧ *length xvec6* = *length Tvec* ∧
xvec6 $\sharp*$ *Tvec* ∧ *xvec6* $\sharp*$ Ψ ∧ *xvec6* $\sharp*$ *M* ∧ *xvec6* $\sharp*$ *K* **by** *blast*
qed
next
case(*cOutput M K N P*)
then show *?thesis* **by**(*rule rOutput*)
next
case(*cBrOutput M N P*)
then show *?thesis* **by**(*rule rBrOutput*)
next
case(*cCase Cs P φ*)
then show *?thesis* **by**(*rule rCase*)
next
case(*cPar1* Ψ_Q *P* α *P′ Q* A_Q)
have *B*: *cP* = *P* ∥ *Q* **and** *C*: *cRs* = α ≺ *P′* ∥ *Q*
by *fact+*
from ‹*bn* α $\sharp*$ (*xvec1*, *xvec2*, *xvec3*, *xvec4*, *xvec5*, *xvec6*, *xvec7*, *xvec8*, *xvec9*, *x1*, *x2*, *x3*, *x4*, *cP*, *cRs*, *C*)› **have** *bn* α $\sharp*$ *xvec2* **by** *simp*
from ‹A_Q $\sharp*$ (*xvec1*, *xvec2*, *xvec3*, *xvec4*, *xvec5*, *xvec6*, *xvec7*, *xvec8*, *xvec9*, *x1*, *x2*, *x3*, *x4*, *cP*, *cRs*, *C*)› **have** A_Q $\sharp*$ *xvec2* **and** A_Q $\sharp*$ *C* **by** *simp+*

from ‹*length xvec2* = *residualLength cRs*› *C* **have** *length xvec2* = *length*(*bn* α)
by *simp*
then obtain *p* **where** *S*: *set p* ⊆ *set*(*bn* α) × *set*(*bn*(*p* · α)) **and** *distinctPerm p* **and** *xvec2*= *bn*(*p* · α)
using ‹*bn* α $\sharp*$ *xvec2*› ‹*distinct*(*bn* α)› ‹*distinct xvec2*›
by − (*rule constructPerm*[**where** *xvec*=*bn* α **and** *yvec*=*xvec2*], *auto simp add*: *eqvts*)
show *?thesis*
proof(*rule rPar1*[**where** *P*=*P* **and** *Q*=*Q* **and** α=*p* · α **and** *P′*=*p* · *P′* **and**

A_Q=A_Q **and** Ψ_Q=Ψ_Q])
assume *xvec2* $\sharp*$ Ψ **and** *xvec2* $\sharp*$ *cP* **and** *xvec2* $\sharp*$ *cRs*
note ‹*cP* = *P* ∥ *Q*›
moreover from *C S* ‹*bn* α $\sharp*$ *xvec2*› ‹*xvec2* $\sharp*$ *cRs*› ‹*xvec2* = *bn*(*p* ∙ α)› ‹*bn* α $\sharp*$ *subject* α› ‹*xvec2* $\sharp*$ *cP*› ‹*bn* α $\sharp*$ *Q*›
have *cRs* = (*p* ∙ α) ≺ (*p* ∙ *P'*) ∥ *Q*
apply *clarsimp*
by(*subst residualAlpha*[**where** *p*=*p*]) *auto*
moreover note ‹*xvec2* = *bn*(*p* ∙ α)›
moreover from ‹Ψ ⊗ Ψ_Q ▷ *P* ⟼α ≺ *P'*› *S B C S* ‹*bn* α $\sharp*$ *xvec2*› ‹*xvec2* $\sharp*$ *cRs*› ‹*xvec2* = *bn*(*p* ∙ α)› ‹*bn* α $\sharp*$ *subject* α› ‹*xvec2* $\sharp*$ *cP*›
have Ψ ⊗ Ψ_Q ▷ *P* ⟼(*p* ∙ α) ≺ (*p* ∙ *P'*)
by(*subst residualAlpha*[*symmetric*]) *auto*
moreover note ‹*extractFrame Q* = ⟨A_Q, Ψ_Q⟩› ‹*distinct* A_Q› ‹A_Q $\sharp*$ *P*› ‹A_Q $\sharp*$ *Q*› ‹A_Q $\sharp*$ Ψ› ‹A_Q $\sharp*$ α›
moreover from ‹A_Q $\sharp*$ α› ‹A_Q $\sharp*$ *xvec2*› *S* ‹*xvec2* = *bn*(*p* ∙ α)› ‹*distinctPerm p*› **have** A_Q $\sharp*$ (*p* ∙ α)
by(*subst fresh-star-bij*[*symmetric*, **where** *pi*=*p*]) *simp*
moreover from ‹A_Q $\sharp*$ *P'*› ‹A_Q $\sharp*$ α› ‹A_Q $\sharp*$ *xvec2*› *S* ‹*xvec2* = *bn*(*p* ∙ α)› ‹*distinctPerm p*› **have** A_Q $\sharp*$ (*p* ∙ *P'*)
by(*subst fresh-star-bij*[*symmetric*, **where** *pi*=*p*]) *simp*
moreover note ‹A_Q $\sharp*$ *C*›
ultimately show *cP* = *P* ∥ *Q* ∧ *cRs* = (*p* ∙ α) ≺ (*p* ∙ *P'*) ∥ *Q* ∧ *xvec2* = *bn* (*p* ∙ α) ∧
Ψ ⊗ Ψ_Q ▷ *P* ⟼ (*p* ∙ α) ≺ *p* ∙ *P'* ∧ *extractFrame Q* = ⟨A_Q, Ψ_Q⟩ ∧ *distinct* A_Q ∧ A_Q $\sharp*$ *P* ∧
A_Q $\sharp*$ *Q* ∧ A_Q $\sharp*$ Ψ ∧ A_Q $\sharp*$ (*p* ∙ α) ∧ A_Q $\sharp*$ (*p* ∙ *P'*) ∧ A_Q $\sharp*$ *C* **by** *blast*
qed
next
case(*cPar2* Ψ_P *Q* α *Q'* *P* A_P)
have *B*: *cP* = *P* ∥ *Q* **and** *C*: *cRs* = α ≺ *P* ∥ *Q'*
by *fact+*
from ‹*bn* α $\sharp*$ (*xvec1*, *xvec2*, *xvec3*, *xvec4*, *xvec5*, *xvec6*, *xvec7*, *xvec8*, *xvec9*, *x1*, *x2*, *x3*, *x4*, *cP*, *cRs*, *C*)› **have** *bn* α $\sharp*$ *xvec3* **by** *simp*
from ‹A_P $\sharp*$ (*xvec1*, *xvec2*, *xvec3*, *xvec4*, *xvec5*, *xvec6*, *xvec7*, *xvec8*, *xvec9*, *x1*, *x2*, *x3*, *x4*, *cP*, *cRs*, *C*)› **have** A_P $\sharp*$ *xvec3* **and** A_P $\sharp*$ *C* **by** *simp+*

from ‹*length xvec3* = *residualLength cRs*› *C* **have** *length xvec3* = *length*(*bn* α)
by *simp*
then obtain *p* **where** *S*: *set p* ⊆ *set*(*bn* α) × *set*(*bn*(*p* ∙ α)) **and** *distinctPerm p* **and** *xvec3* = *bn*(*p* ∙ α)
using ‹*bn* α $\sharp*$ *xvec3*› ‹*distinct*(*bn* α)› ‹*distinct xvec3*›
by − (*rule constructPerm*[**where** *xvec*=*bn* α **and** *yvec*=*xvec3*], *auto simp add*: *eqvts*)
show *?thesis*
proof(*rule rPar2*[**where** *P*=*P* **and** *Q*=*Q* **and** α=*p* ∙ α **and** *Q'*=*p* ∙ *Q'* **and** A_P=A_P **and** Ψ_P=Ψ_P])
assume *xvec3* $\sharp*$ Ψ **and** *xvec3* $\sharp*$ *cP* **and** *xvec3* $\sharp*$ *cRs*
note ‹*cP* = *P* ∥ *Q*›

moreover from B C S ‹$bn\ \alpha\ \sharp*\ xvec3$› ‹$xvec3\ \sharp*\ cRs$› ‹$xvec3 = bn(p \cdot \alpha)$› ‹$bn\ \alpha\ \sharp*\ subject\ \alpha$› ‹$xvec3\ \sharp*\ cP$› ‹$bn\ \alpha\ \sharp*\ P$›
have $cRs = (p \cdot \alpha) \prec P \parallel (p \cdot Q')$
apply *clarsimp*
by(*subst residualAlpha*[**where** p=p]) *auto*
moreover note ‹$xvec3 = bn(p \cdot \alpha)$›
moreover from ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto \alpha \prec Q'$› S B C S ‹$bn\ \alpha\ \sharp*\ xvec3$› ‹$xvec3\ \sharp*\ cRs$› ‹$xvec3 = bn(p \cdot \alpha)$› ‹$bn\ \alpha\ \sharp*\ subject\ \alpha$› ‹$xvec3\ \sharp*\ cP$›
have $\Psi \otimes \Psi_P \rhd Q \longmapsto (p \cdot \alpha) \prec (p \cdot Q')$
by(*subst residualAlpha*[*symmetric*]) *auto*
moreover note ‹*extractFrame* $P = \langle A_P, \Psi_P \rangle$› ‹*distinct* A_P› ‹$A_P\ \sharp*\ P$› ‹$A_P\ \sharp*\ Q$› ‹$A_P\ \sharp*\ \Psi$› ‹$A_P\ \sharp*\ \alpha$›
moreover from ‹$A_P\ \sharp*\ \alpha$› ‹$A_P\ \sharp*\ xvec3$› S ‹$xvec3 = bn(p \cdot \alpha)$› ‹*distinctPerm* p› **have** $A_P\ \sharp*\ (p \cdot \alpha)$
by(*subst fresh-star-bij*[*symmetric*, **where** pi=p]) *simp*
moreover from ‹$A_P\ \sharp*\ Q'$› ‹$A_P\ \sharp*\ \alpha$› ‹$A_P\ \sharp*\ xvec3$› S ‹$xvec3 = bn(p \cdot \alpha)$› ‹*distinctPerm* p› **have** $A_P\ \sharp*\ (p \cdot Q')$
by(*subst fresh-star-bij*[*symmetric*, **where** pi=p]) *simp*
moreover note ‹$A_P\ \sharp*\ C$›
ultimately show $cP = P \parallel Q \wedge cRs = (p \cdot \alpha) \prec P \parallel (p \cdot Q') \wedge xvec3 = bn(p \cdot \alpha) \wedge$
$\Psi \otimes \Psi_P \rhd Q \longmapsto (p \cdot \alpha) \prec p \cdot Q' \wedge$
extractFrame $P = \langle A_P, \Psi_P \rangle \wedge$ *distinct* $A_P \wedge A_P\ \sharp*\ P \wedge A_P\ \sharp*\ Q \wedge A_P\ \sharp*\ \Psi \wedge A_P\ \sharp*\ (p \cdot \alpha) \wedge$
$A_P\ \sharp*\ (p \cdot Q') \wedge A_P\ \sharp*\ C$ **by** *blast*
qed
next
case(*cComm1* Ψ_Q P M N P' A_P Ψ_P Q K *xvec* Q' A_Q)
then show *?thesis* **by** − (*rule rComm1*[**where** P=P **and** Q=Q], (*assumption* | *simp*)+)
next
case(*cComm2* Ψ_Q P M *xvec* N P' A_P Ψ_P Q K Q' A_Q)
then show *?thesis* **by** − (*rule rComm2*[**where** P=P **and** Q=Q], (*assumption* | *simp*)+)
next
case(*cBrMerge* Ψ_Q P M N P' A_P Ψ_P Q Q' A_Q)
then show *?thesis* **by** − (*rule rBrMerge*[**where** P=P **and** Q=Q], (*assumption* | *simp*)+)
next
case(*cBrComm1* Ψ_Q P M N P' A_P Ψ_P Q *xvec* Q' A_Q)
have B: $cP = P \parallel Q$ **and** C: $cRs = ¡M(\!|\nu * xvec|\!)\langle N \rangle \prec P' \parallel Q'$
by *fact*+
from ‹$xvec\ \sharp*\ (xvec1, xvec2, xvec3, xvec4, xvec5, xvec6, xvec7, xvec8, xvec9, x1, x2, x3, x4, cP, cRs, C)$› **have** $xvec\ \sharp*\ xvec7$ **and** $xvec\ \sharp*\ C$ **by** *simp*+
from ‹$A_P\ \sharp*\ (xvec1, xvec2, xvec3, xvec4, xvec5, xvec6, xvec7, xvec8, xvec9, x1, x2, x3, x4, cP, cRs, C)$› **have** $A_P\ \sharp*\ xvec7$ **and** $A_P\ \sharp*\ C$ **by** *simp*+
from ‹$A_Q\ \sharp*\ (xvec1, xvec2, xvec3, xvec4, xvec5, xvec6, xvec7, xvec8, xvec9, x1, x2, x3, x4, cP, cRs, C)$› **have** $A_Q\ \sharp*\ xvec7$ **and** $A_Q\ \sharp*\ C$ **by** *simp*+

from ‹*length xvec7* = *residualLength cRs*› *C* **have** *length xvec7* = *length xvec*
by *simp*
then obtain *p* **where** *S*: *set p* ⊆ *set xvec* × *set* (*p* ∙ *xvec*) **and** *distinctPerm p* **and** *xvec7* = *p* ∙ *xvec*
using ‹*xvec* ♯* *xvec7*› ‹*distinct xvec*› ‹*distinct xvec7*›
by − (*rule constructPerm*[**where** *xvec*=*xvec* **and** *yvec*=*xvec7*], *auto simp add*: *eqvts*)
show *?thesis*
proof(*rule rBrComm1*[**where** *P*=*P* **and** *Q*=*Q* **and** *P'*=*p* ∙ *P'*
and *Q'*=*p* ∙ *Q'* **and** *N*=*p* ∙ *N* **and** A_P=A_P **and** Ψ_P=Ψ_P **and** A_Q=A_Q **and** Ψ_Q=Ψ_Q **and** *M*=*M*])
assume *xvec7* ♯* Ψ **and** *xvec7* ♯* *cP* **and** *xvec7* ♯* *cRs*
from ‹A_Q ♯* *xvec7*› ‹*xvec7* = *p* ∙ *xvec*›
have A_Q ♯* (*p* ∙ *xvec*) **by** *simp*
from ‹A_P ♯* *xvec7*› ‹*xvec7* = *p* ∙ *xvec*›
have A_P ♯* (*p* ∙ *xvec*) **by** *simp*
from ‹*xvec* ♯* *M*› **have** (*p* ∙ *xvec*) ♯* (*p* ∙ *M*) **by**(*simp add*: *fresh-star-bij*)
with *S* ‹*xvec* ♯* *M*› ‹*xvec7* ♯* *cRs*› *C* ‹*xvec7* = *p* ∙ *xvec*› **have** *xvec7* ♯* *M* **by** *simp*

note ‹*cP* = *P* ∥ *Q*›
moreover from *C S* ‹*xvec* ♯* *xvec7*› ‹*xvec7* ♯* *cRs*› ‹*xvec7* = *p* ∙ *xvec*› ‹*xvec* ♯* *M*› ‹*xvec7* ♯* *cP*›
have *cRs* = ¡*M*(⦇*ν***xvec7*⦈)⟨(*p* ∙ *N*)⟩ ≺ (*p* ∙ *P'*) ∥ (*p* ∙ *Q'*)
apply *clarsimp*
by(*subst residualAlpha*[**where** *p*=*p*]) *simp*+

moreover note ‹*xvec7* = *p* ∙ *xvec*›
moreover from ‹Ψ ⊗ Ψ_Q ▷ *P* ⟼ ¿*M*(⦇*N*⦈) ≺ *P'*› *S B* ‹*distinctPerm p*› ‹*xvec* ♯* *xvec7*› ‹*xvec7* = *p* ∙ *xvec*› ‹*xvec* ♯* *P*› ‹*xvec7* ♯* *cP*›
have Ψ ⊗ Ψ_Q ▷ *P* ⟼ ¿*M*(⦇(*p* ∙ *N*)⦈) ≺ *p* ∙ *P'*
by(*simp add*: *brinputAlpha*)

moreover from ‹Ψ ⊗ Ψ_P ▷ *Q* ⟼ ¡*M*(⦇*ν***xvec*⦈)⟨*N*⟩ ≺ *Q'*› *S B C* ‹*xvec* ♯* *xvec7*› ‹*xvec7* ♯* *cRs*› ‹*xvec7* = *p* ∙ *xvec*› ‹*xvec* ♯* *M*› ‹*xvec7* ♯* *cP*› ‹*xvec7* ♯* *M*›
have Ψ ⊗ Ψ_P ▷ *Q* ⟼ ¡*M*(⦇*ν***xvec7*⦈)⟨(*p* ∙ *N*)⟩ ≺ *p* ∙ *Q'*
by(*auto simp add*: *residualAlpha*)

moreover note
‹*extractFrame P* = ⟨A_P, Ψ_P⟩› ‹*distinct* A_P› ‹*extractFrame Q* = ⟨A_Q, Ψ_Q⟩› ‹*distinct* A_Q›
‹A_P ♯* Ψ› ‹A_P ♯* Ψ_Q› ‹A_P ♯* *P*› ‹A_P ♯* *Q*› ‹A_P ♯* A_Q› ‹A_P ♯* *xvec7*›
‹A_Q ♯* Ψ› ‹A_Q ♯* Ψ_P› ‹A_Q ♯* *P*› ‹A_Q ♯* *Q*› ‹A_Q ♯* *xvec7*›

moreover from *S* ‹A_P ♯* *xvec*› ‹A_P ♯* (*p* ∙ *xvec*)› ‹A_P ♯* *N*›
have A_P ♯* (*p* ∙ *N*)
by(*simp add*: *freshChainSimps*)
moreover from *S* ‹A_P ♯* *xvec*› ‹A_P ♯* (*p* ∙ *xvec*)› ‹A_P ♯* *P'*›
have A_P ♯* (*p* ∙ *P'*)

by(*simp add*: *freshChainSimps*)
moreover from S ‹$A_P \sharp* xvec$› ‹$A_P \sharp* (p \cdot xvec)$› ‹$A_P \sharp* Q'$›
have $A_P \sharp* (p \cdot Q')$
by(*simp add*: *freshChainSimps*)
moreover from S ‹$A_Q \sharp* xvec$› ‹$A_Q \sharp* (p \cdot xvec)$› ‹$A_Q \sharp* N$›
have $A_Q \sharp* (p \cdot N)$
by(*simp add*: *freshChainSimps*)
moreover from S ‹$A_Q \sharp* xvec$› ‹$A_Q \sharp* (p \cdot xvec)$› ‹$A_Q \sharp* P'$›
have $A_Q \sharp* (p \cdot P')$
by(*simp add*: *freshChainSimps*)
moreover from S ‹$A_Q \sharp* xvec$› ‹$A_Q \sharp* (p \cdot xvec)$› ‹$A_Q \sharp* Q'$›
have $A_Q \sharp* (p \cdot Q')$
by(*simp add*: *freshChainSimps*)

moreover note ‹$xvec\mathcal{7} \sharp* \Psi$›

moreover from ‹$xvec\mathcal{7} \sharp* cP$› ‹$cP = P \parallel Q$› ‹*extractFrame* $P = \langle A_P, \Psi_P\rangle$›
‹$A_P \sharp* xvec\mathcal{7}$›
have $xvec\mathcal{7} \sharp* \Psi_P$
by *simp* (*metis extractFrameFreshChain freshFrameDest*)

moreover from ‹$xvec\mathcal{7} \sharp* cP$› ‹$cP = P \parallel Q$› ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$›
‹$A_Q \sharp* xvec\mathcal{7}$›
have $xvec\mathcal{7} \sharp* \Psi_Q$
by *simp* (*metis extractFrameFreshChain freshFrameDest*)

moreover from ‹$xvec\mathcal{7} \sharp* cP$› ‹$cP = P \parallel Q$› **have** $xvec\mathcal{7} \sharp* P$ **by** *simp*
moreover from ‹$xvec\mathcal{7} \sharp* cP$› ‹$cP = P \parallel Q$› **have** $xvec\mathcal{7} \sharp* Q$ **by** *simp*

moreover note ‹$A_P \sharp* M$› ‹$A_Q \sharp* M$›

moreover note ‹$xvec\mathcal{7} \sharp* M$›

moreover note ‹$A_P \sharp* C$› ‹$A_Q \sharp* C$› ‹*distinct* $xvec\mathcal{7}$›

ultimately show $cP = P \parallel Q \wedge cRs = ¡M⦇\nu*xvec\mathcal{7}⦈\langle(p \cdot N)\rangle \prec (p \cdot P') \parallel (p \cdot Q') \wedge$
$\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M⦇(p \cdot N)⦈ \prec p \cdot P' \wedge$ *extractFrame* $P = \langle A_P, \Psi_P\rangle \wedge$ *distinct* $A_P \wedge$
$\Psi \otimes \Psi_P \rhd Q \longmapsto ¡M⦇\nu*xvec\mathcal{7}⦈\langle(p \cdot N)\rangle \prec p \cdot Q' \wedge$ *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle \wedge$ *distinct* $A_Q \wedge$
$A_P \sharp* \Psi \wedge A_P \sharp* \Psi_Q \wedge A_P \sharp* P \wedge A_P \sharp* (p \cdot N) \wedge A_P \sharp* (p \cdot P') \wedge A_P \sharp* Q \wedge A_P \sharp* (p \cdot Q') \wedge$
$A_P \sharp* A_Q \wedge A_P \sharp* xvec\mathcal{7} \wedge A_Q \sharp* \Psi \wedge A_Q \sharp* \Psi_P \wedge A_Q \sharp* P \wedge A_Q \sharp* (p \cdot N) \wedge A_Q \sharp* (p \cdot P') \wedge$
$A_Q \sharp* Q \wedge A_Q \sharp* (p \cdot Q') \wedge A_Q \sharp* xvec\mathcal{7} \wedge xvec\mathcal{7} \sharp* \Psi \wedge xvec\mathcal{7} \sharp* \Psi_P \wedge xvec\mathcal{7} \sharp* \Psi_Q \wedge xvec\mathcal{7} \sharp* P \wedge$
$xvec\mathcal{7} \sharp* Q \wedge A_P \sharp* M \wedge A_Q \sharp* M \wedge xvec\mathcal{7} \sharp* M \wedge A_P \sharp* C \wedge A_Q \sharp* C \wedge$ *distinct* $xvec\mathcal{7}$ **by** *blast*

qed
next
case(*cBrComm2* Ψ_Q P M *xvec* N P' A_P Ψ_P Q Q' A_Q)
have *B*: $cP = P \parallel Q$ **and** *C*: $cRs = ¡M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P' \parallel Q'$
by *fact+*
from ‹*xvec* ♯* (*xvec1*, *xvec2*, *xvec3*, *xvec4*, *xvec5*, *xvec6*, *xvec7*, *xvec8*, *xvec9*, *x1*, *x2*, *x3*, *x4*, *cP*, *cRs*, *C*)› **have** *xvec* ♯* *xvec8* **and** *xvec* ♯* *C* **by** *simp+*
from ‹A_P ♯* (*xvec1*, *xvec2*, *xvec3*, *xvec4*, *xvec5*, *xvec6*, *xvec7*, *xvec8*, *xvec9*, *x1*, *x2*, *x3*, *x4*, *cP*, *cRs*, *C*)› **have** A_P ♯* *xvec8* **and** A_P ♯* *C* **by** *simp+*
from ‹A_Q ♯* (*xvec1*, *xvec2*, *xvec3*, *xvec4*, *xvec5*, *xvec6*, *xvec7*, *xvec8*, *xvec9*, *x1*, *x2*, *x3*, *x4*, *cP*, *cRs*, *C*)› **have** A_Q ♯* *xvec8* **and** A_Q ♯* *C* **by** *simp+*

from ‹*length xvec8* = *residualLength cRs*› *C* **have** *length xvec8* = *length xvec*
by *simp*
then obtain *p* **where** *S*: *set p* ⊆ *set xvec* × *set* (*p* · *xvec*) **and** *distinctPerm p* **and** *xvec8* = *p* · *xvec*
using ‹*xvec* ♯* *xvec8*› ‹*distinct xvec*› ‹*distinct xvec8*›
by − (*rule constructPerm*[**where** *xvec*=*xvec* **and** *yvec*=*xvec8*], *auto simp add*: *eqvts*)
show *?thesis*
proof(*rule rBrComm2*[**where** P=P **and** Q=Q **and** P'=p · P'
and Q'=p · Q' **and** N=p · N **and** A_P=A_P **and** Ψ_P=Ψ_P **and** A_Q=A_Q **and** Ψ_Q=Ψ_Q **and** M=M])
assume *xvec8* ♯* Ψ **and** *xvec8* ♯* *cP* **and** *xvec8* ♯* *cRs*
from ‹A_Q ♯* *xvec8*› ‹*xvec8* = *p* · *xvec*›
have A_Q ♯* (*p* · *xvec*) **by** *simp*
from ‹A_P ♯* *xvec8*› ‹*xvec8* = *p* · *xvec*›
have A_P ♯* (*p* · *xvec*) **by** *simp*
from ‹*xvec* ♯* *M*› **have** (*p* · *xvec*) ♯* (*p* · *M*) **by**(*simp add*: *fresh-star-bij*)
with *S* ‹*xvec* ♯* *M*› ‹*xvec8* ♯* *cRs*› *C* ‹*xvec8* = *p* · *xvec*› **have** *xvec8* ♯* *M* **by** *simp*

note ‹*cP* = *P* ∥ *Q*›
moreover from *C S* ‹*xvec* ♯* *xvec8*› ‹*xvec8* ♯* *cRs*› ‹*xvec8* = *p* · *xvec*› ‹*xvec* ♯* *M*› ‹*xvec8* ♯* *cP*›
have *cRs* = ¡$M(\!|\nu{*}xvec8|\!)\langle(p \cdot N)\rangle \prec (p \cdot P') \parallel (p \cdot Q')$
apply *clarsimp*
by(*subst residualAlpha*[**where** *p*=*p*]) *simp+*

moreover note ‹*xvec8* = *p* · *xvec*›
moreover from ‹Ψ ⊗ Ψ_P ▷ $Q \longmapsto$ ¿$M(\!|N|\!) \prec Q'$› *S B* ‹*distinctPerm p*› ‹*xvec* ♯* *xvec8*› ‹*xvec8* = *p* · *xvec*› ‹*xvec* ♯* *Q*› ‹*xvec8* ♯* *cP*›
have Ψ ⊗ Ψ_P ▷ $Q \longmapsto$ ¿$M(\!|(p \cdot N)|\!) \prec p \cdot Q'$
by(*simp add*: *brinputAlpha*)

moreover from ‹Ψ ⊗ Ψ_Q ▷ $P \longmapsto$ ¡$M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P'$› *S B C* ‹*xvec* ♯* *xvec8*› ‹*xvec8* ♯* *cRs*› ‹*xvec8* = *p* · *xvec*› ‹*xvec* ♯* *M*› ‹*xvec8* ♯* *cP*› ‹*xvec8* ♯* *M*›
have Ψ ⊗ Ψ_Q ▷ $P \longmapsto$ ¡$M(\!|\nu{*}xvec8|\!)\langle(p \cdot N)\rangle \prec p \cdot P'$
by(*auto simp add*: *residualAlpha*)

moreover note
‹*extractFrame* $P = \langle A_P, \Psi_P \rangle$› ‹*distinct* A_P› ‹*extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$› ‹*distinct* A_Q›
‹$A_P \sharp* \Psi$› ‹$A_P \sharp* \Psi_Q$› ‹$A_P \sharp* P$› ‹$A_P \sharp* Q$› ‹$A_P \sharp* A_Q$› ‹$A_P \sharp* xvec8$›
‹$A_Q \sharp* \Psi$› ‹$A_Q \sharp* \Psi_P$› ‹$A_Q \sharp* P$› ‹$A_Q \sharp* Q$› ‹$A_Q \sharp* xvec8$›

moreover from S ‹$A_P \sharp* xvec$› ‹$A_P \sharp* (p \cdot xvec)$› ‹$A_P \sharp* N$›
have $A_P \sharp* (p \cdot N)$
by(*simp add*: *freshChainSimps*)
moreover from S ‹$A_P \sharp* xvec$› ‹$A_P \sharp* (p \cdot xvec)$› ‹$A_P \sharp* P'$›
have $A_P \sharp* (p \cdot P')$
by(*simp add*: *freshChainSimps*)
moreover from S ‹$A_P \sharp* xvec$› ‹$A_P \sharp* (p \cdot xvec)$› ‹$A_P \sharp* Q'$›
have $A_P \sharp* (p \cdot Q')$
by(*simp add*: *freshChainSimps*)
moreover from S ‹$A_Q \sharp* xvec$› ‹$A_Q \sharp* (p \cdot xvec)$› ‹$A_Q \sharp* N$›
have $A_Q \sharp* (p \cdot N)$
by(*simp add*: *freshChainSimps*)
moreover from S ‹$A_Q \sharp* xvec$› ‹$A_Q \sharp* (p \cdot xvec)$› ‹$A_Q \sharp* P'$›
have $A_Q \sharp* (p \cdot P')$
by(*simp add*: *freshChainSimps*)
moreover from S ‹$A_Q \sharp* xvec$› ‹$A_Q \sharp* (p \cdot xvec)$› ‹$A_Q \sharp* Q'$›
have $A_Q \sharp* (p \cdot Q')$
by(*simp add*: *freshChainSimps*)

moreover note ‹*xvec8* $\sharp* \Psi$›

moreover from ‹*xvec8* $\sharp* cP$› ‹$cP = P \parallel Q$› ‹*extractFrame* $P = \langle A_P, \Psi_P \rangle$› ‹$A_P \sharp* xvec8$›
have *xvec8* $\sharp* \Psi_P$
by *simp* (*metis extractFrameFreshChain freshFrameDest*)

moreover from ‹*xvec8* $\sharp* cP$› ‹$cP = P \parallel Q$› ‹*extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$› ‹$A_Q \sharp* xvec8$›
have *xvec8* $\sharp* \Psi_Q$
by *simp* (*metis extractFrameFreshChain freshFrameDest*)

moreover from ‹*xvec8* $\sharp* cP$› ‹$cP = P \parallel Q$› **have** *xvec8* $\sharp* P$ **by** *simp*
moreover from ‹*xvec8* $\sharp* cP$› ‹$cP = P \parallel Q$› **have** *xvec8* $\sharp* Q$ **by** *simp*

moreover note ‹$A_P \sharp* M$› ‹$A_Q \sharp* M$›

moreover note ‹*xvec8* $\sharp* M$›

moreover note ‹$A_P \sharp* C$› ‹$A_Q \sharp* C$› ‹*distinct xvec8*›

ultimately show $cP = P \parallel Q \wedge cRs = ¡M⦇\nu*xvec8⦈\langle(p \cdot N)\rangle \prec (p \cdot P') \parallel (p \cdot Q') \wedge$

$\Psi \otimes \Psi_Q \rhd P \longmapsto ¡M(\!|\nu{*}xvec8|\!)\langle(p \cdot N)\rangle \prec p \cdot P' \wedge$ *extractFrame* $P = \langle A_P, \Psi_P\rangle \wedge$ *distinct* $A_P \wedge$

$\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|(p \cdot N)|\!) \prec p \cdot Q' \wedge$ *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle \wedge$ *distinct* $A_Q \wedge$

$A_P \sharp* \Psi \wedge A_P \sharp* \Psi_Q \wedge A_P \sharp* P \wedge A_P \sharp* (p \cdot N) \wedge A_P \sharp* (p \cdot P') \wedge A_P \sharp* Q \wedge A_P \sharp* (p \cdot Q') \wedge$

$A_P \sharp* A_Q \wedge A_P \sharp* xvec8 \wedge A_Q \sharp* \Psi \wedge A_Q \sharp* \Psi_P \wedge A_Q \sharp* P \wedge A_Q \sharp* (p \cdot N) \wedge A_Q \sharp* (p \cdot P') \wedge$

$A_Q \sharp* Q \wedge A_Q \sharp* (p \cdot Q') \wedge A_Q \sharp* xvec8 \wedge xvec8 \sharp* \Psi \wedge xvec8 \sharp* \Psi_P \wedge xvec8 \sharp* \Psi_Q \wedge xvec8 \sharp* P \wedge$

$xvec8 \sharp* Q \wedge A_P \sharp* M \wedge A_Q \sharp* M \wedge xvec8 \sharp* M \wedge A_P \sharp* C \wedge A_Q \sharp* C \wedge$ *distinct xvec8* **by** *blast*

qed

next

case(*cBrClose P M xvec N P' x*)

have *B*: $cP = (\!|\nu x|\!)P$ **and** *C*: $cRs = \tau \prec (\!|\nu x|\!)((\!|\nu{*}xvec|\!)P')$

by *fact+*

from ‹$x \sharp (xvec1, xvec2, xvec3, xvec4, xvec5, xvec6, xvec7, xvec8, xvec9, x1, x2, x3, x4, cP, cRs, C)$› **have** $x \sharp cP$ **and** $x \sharp cRs$ **and** $x \neq x3$ **by** *simp+*

from ‹$xvec \sharp* (xvec1, xvec2, xvec3, xvec4, xvec5, xvec6, xvec7, xvec8, xvec9, x1, x2, x3, x4, cP, cRs, C)$› **have** $xvec \sharp* cP$ **and** $xvec \sharp* cRs$ **and** $x3 \sharp xvec$ **and** $xvec \sharp* x3$ **by** *simp+*

obtain *r*::*name prm* **where** $(r \cdot xvec) \sharp* \Psi$ **and** $(r \cdot xvec) \sharp* P$ **and** $(r \cdot xvec) \sharp* M$

and $(r \cdot xvec) \sharp* N$ **and** $(r \cdot xvec) \sharp* P'$ **and** $(r \cdot xvec) \sharp* x$

and $(r \cdot xvec) \sharp* C$ **and** $(r \cdot xvec) \sharp* x3$

and $(r \cdot xvec) \sharp* ([(x, x3)] \cdot P)$ **and** $(r \cdot xvec) \sharp* ([(x, x3)] \cdot M)$

and *Sr*: $(set\ r) \subseteq (set\ xvec) \times (set(r \cdot xvec))$ **and** *distinctPerm r*

by(*rule name-list-avoiding*[**where** *xvec*=*xvec* **and** *c*=(Ψ, *P*, *M*, *N*, *P'*, *x*, *x3*, ([(*x*, *x3*)] · *P*), ([(*x*, *x3*)] · *P*), *C*)])

(*auto simp add*: *eqvts*)

from ‹$(r \cdot xvec) \sharp* x3$› **have** $x3 \sharp (r \cdot xvec)$ **by** *simp*

show *?thesis*

proof(*rule rBrClose*[**where** *P*=[(*x*, *x3*)] · *P* **and** *M*=[(*x*, *x3*)] · *M* **and** *N*=[(*x*, *x3*)] · (*r* · *N*) **and** *xvec*=(*r* · *xvec*) **and** *P'*=[(*x*, *x3*)] · (*r* · *P'*)])

assume $x3 \sharp \Psi$ **and** $x3 \sharp cP$ **and** $x3 \sharp cRs$

with ‹$x3 \sharp xvec$› **have** $x3 \notin set\ xvec$ **by** *simp*

from ‹$x3 \sharp cRs$› *C* **have** $x3 \sharp (\tau \prec (\!|\nu x|\!)((\!|\nu{*}xvec|\!)P'))$ **by** *simp*

then have $x3 \sharp ((\!|\nu x|\!)((\!|\nu{*}xvec|\!)P'))$ **by** *simp*

with ‹$x \neq x3$› **have** $x3 \sharp ((\!|\nu{*}xvec|\!)P')$ **by**(*simp add*: *psi.fresh abs-fresh*)

then have $(x3 \in set\ xvec) \vee (x3 \sharp P')$ **by**(*simp add*: *resChainFresh*)

with ‹$x3 \notin set\ xvec$› **have** $x3 \sharp P'$ **by** *blast*

with ‹$x3 \sharp xvec$› ‹$x3 \sharp (r \cdot xvec)$› *Sr* **have** $x3 \sharp (r \cdot P')$

by(*simp add*: *freshChainSimps*)

from ‹$cP = (\!|\nu x|\!)P$› ‹$x3 \sharp cP$› ‹$x \neq x3$› **have** *cP-perm*: $cP = (\!|\nu x3|\!)([(x, x3)] \cdot P)$
by(*simp add*: *alphaRes abs-fresh*)
from ‹$(r \cdot xvec) \sharp* P'$› *Sr C*
have $cRs = \tau \prec (\!|\nu x|\!)((\!|\nu*(r \cdot xvec)|\!)(r \cdot P'))$
by(*simp add*: *resChainAlpha*)
moreover from ‹$x3 \sharp (r \cdot P')$›
have $x3 \sharp (\!|\nu*(r \cdot xvec)|\!)(r \cdot P')$
by(*simp add*: *resChainFresh*)
ultimately have $cRs = \tau \prec (\!|\nu x3|\!)([(x, x3)] \cdot (\!|\nu*(r \cdot xvec)|\!)(r \cdot P'))$ **by**(*simp add*: *alphaRes*)
with ‹$(r \cdot xvec) \sharp* x$› ‹$(r \cdot xvec) \sharp* x3$›
have *cRs-perm*: $cRs = \tau \prec (\!|\nu x3|\!)((\!|\nu*(r \cdot xvec)|\!)[(x, x3)] \cdot (r \cdot P'))$
by(*simp add*: *eqvts*)

from ‹$x \in supp\ M$›
have *supp-inc-perm*: $x3 \in supp\ ([(x, x3)] \cdot M)$
by (*metis fresh-bij fresh-def swap-simps*)

from ‹$x \sharp xvec$› ‹$(r \cdot xvec) \sharp* x$› *Sr* **have** $r \cdot x = x$ **by** *simp*

from ‹$\Psi \rhd P \longmapsto ¡M(\!|\nu* xvec|\!)\langle N\rangle \prec P'$› ‹$(r \cdot xvec) \sharp* N$› ‹$(r \cdot xvec) \sharp* P'$› *Sr*
have $\Psi \rhd P \longmapsto ¡M(\!|\nu*(r \cdot xvec)|\!)\langle(r \cdot N)\rangle \prec (r \cdot P')$
by(*simp add*: *boundOutputChainAlpha''* *create-residual.simps*)
then have $[(x, x3)] \cdot (\Psi \rhd P \longmapsto ¡M(\!|\nu*(r \cdot xvec)|\!)\langle(r \cdot N)\rangle \prec (r \cdot P'))$
by(*simp add*: *perm-bool*)
with ‹$x \sharp \Psi$› ‹$x3 \sharp \Psi$› ‹$(r \cdot xvec) \sharp* x$› ‹$x3 \sharp (r \cdot xvec)$›
have *trans-perm*: $\Psi \rhd ([(x, x3)] \cdot P) \longmapsto ¡([(x, x3)] \cdot M)(\!|\nu*(r \cdot xvec)|\!)\langle([(x, x3)] \cdot (r \cdot N))\rangle \prec ([(x, x3)] \cdot (r \cdot P'))$
by(*auto simp add*: *eqvts*)

from ‹*distinctPerm r*› ‹*distinct xvec*›
have *distinct-perm*: *distinct* $(r \cdot xvec)$ **by** *simp*

note *cP-perm cRs-perm supp-inc-perm trans-perm distinct-perm*
‹$(r \cdot xvec) \sharp* \Psi$› ‹$(r \cdot xvec) \sharp* ([(x, x3)] \cdot P)$› ‹$(r \cdot xvec) \sharp* ([(x, x3)] \cdot M)$›
‹$(r \cdot xvec) \sharp* C$› ‹$x3 \sharp \Psi$› ‹$x3 \sharp (r \cdot xvec)$›

then show $cP = (\!|\nu x3|\!)([(x, x3)] \cdot P) \wedge cRs = \tau \prec (\!|\nu x3|\!)((\!|\nu* r \cdot xvec|\!)[(x, x3)] \cdot r \cdot P') \wedge$
$x3 \in supp\ ([(x, x3)] \cdot M) \wedge$
$\Psi \rhd [(x, x3)] \cdot P \longmapsto ¡([(x, x3)] \cdot M)(\!|\nu*(r \cdot xvec)|\!)\langle([(x, x3)] \cdot r \cdot N)\rangle \prec [(x, x3)] \cdot r \cdot P' \wedge$
$distinct\ (r \cdot xvec) \wedge (r \cdot xvec) \sharp* \Psi \wedge (r \cdot xvec) \sharp* ([(x, x3)] \cdot P) \wedge$
$(r \cdot xvec) \sharp* ([(x, x3)] \cdot M) \wedge (r \cdot xvec) \sharp* C \wedge x3 \sharp \Psi \wedge x3 \sharp r \cdot xvec$
by *simp*
qed

next
case(*cOpen P M xvec yvec N P′ x*)
have *B*: *cP* = ⟪*νx*⟫*P* **and** *C*: *cRs* = *M*⟪*ν*∗(*xvec*@*x*#*yvec*)⟫⟨*N*⟩ ≺ *P′*
by *fact*+
from ‹*xvec* ♯∗ (*xvec1*, *xvec2*, *xvec3*, *xvec4*, *xvec5*, *xvec6*, *xvec7*, *xvec8*, *xvec9*, *x1*, *x2*, *x3*, *x4*, *cP*, *cRs*, *C*)› **have** *xvec* ♯∗ *xvec4* **and** *xvec* ♯∗ *cP* **and** *xvec* ♯∗ *cRs* **and** *x1* ♯ *xvec* **by** *simp*+
from ‹*x* ♯ (*xvec1*, *xvec2*, *xvec3*, *xvec4*, *xvec5*, *xvec6*, *xvec7*, *xvec8*, *xvec9*, *x1*, *x2*, *x3*, *x4*, *cP*, *cRs*, *C*)› **have** *x* ♯ *xvec4* **and** *x* ♯ *cP* **and** *x* ♯ *cRs* **and** *x* ≠ *x1* **by** *simp*+
from ‹*yvec* ♯∗ (*xvec1*, *xvec2*, *xvec3*, *xvec4*, *xvec5*, *xvec6*, *xvec7*, *xvec8*, *xvec9*, *x1*, *x2*, *x3*, *x4*, *cP*, *cRs*, *C*)› **have** *yvec* ♯∗ *xvec4* **and** *yvec* ♯∗ *cP* **and** *yvec* ♯∗ *cRs* **and** *x1* ♯ *yvec* **by** *simp*+

from ‹*xvec* ♯∗ *cRs*› ‹*x* ♯ *cRs*› ‹*yvec* ♯∗ *cRs*› *C* **have** (*xvec*@*x*#*yvec*) ♯∗ *M* **by** *simp*
from ‹*xvec* ♯∗ Ψ› ‹*x* ♯ Ψ› ‹*yvec* ♯∗ Ψ› **have** (*xvec*@*x*#*yvec*) ♯∗ Ψ **by** *simp*
from ‹*length xvec4* = *residualLength cRs*› *C* **obtain** *xvec′ y yvec′* **where** *D*: *xvec4* = *xvec′*@*y*#*yvec′* **and** *length xvec′* = *length xvec* **and** *length yvec′* = *length yvec*
by(*auto intro*: *lengthAux2*)
with ‹*distinct xvec*› ‹*distinct yvec*› ‹*x* ♯ *xvec*› ‹*x* ♯ *yvec*› ‹*xvec* ♯∗ *yvec*› ‹*xvec* ♯∗ *xvec4*› ‹*yvec* ♯∗ *xvec4*› ‹*x* ♯ *xvec4*› ‹*distinct xvec4*›
have *distinct xvec′* **and** *distinct yvec′* **and** *xvec′* ♯∗ *yvec′* **and** *x* ≠ *y* **and** *y* ♯ *xvec′* **and** *y* ♯ *yvec′*
and *x* ♯ *xvec′* **and** *x* ♯ *yvec′* **and** *y* ♯ *xvec* **and** *y* ♯ *yvec* **and** *xvec* ♯∗ *xvec′* **and** *yvec* ♯∗ *yvec′*
by *auto*
from ‹*length xvec′* = *length xvec*› ‹*xvec* ♯∗ *xvec′*› ‹*distinct xvec*› ‹*distinct xvec′*›
obtain *p* **where** *Sp*: *set p* ⊆ *set xvec* × *set*(*p* ∙ *xvec*) **and** *distinctPerm p* **and** *E*: *xvec′* = *p* ∙ *xvec*
by(*metis constructPerm*)
from ‹*length yvec′* = *length yvec*› ‹*yvec* ♯∗ *yvec′*› ‹*distinct yvec*› ‹*distinct yvec′*›
obtain *q* **where** *Sq*: *set q* ⊆ *set yvec* × *set*(*q* ∙ *yvec*) **and** *distinctPerm q* **and** *F*: *yvec′* = *q* ∙ *yvec*
by(*metis constructPerm*)

show *?thesis*
proof(*rule rOpen*[**where** *P*=([(*x*, *x1*)] ∙ *P*) **and** *xvec*=*p* ∙ *xvec* **and** *y*=*y* **and** *yvec*=*q* ∙ *yvec* **and** *N*=(*p*@(*x1*, *x*)#*q*) ∙ *N* **and** *P′*=(*p*@(*x1*, *x*)#*q*) ∙ *P′* **and** *M*=*M*])
assume *xvec4* ♯∗ Ψ **and** *xvec4* ♯∗ *cP* **and** *xvec4* ♯∗ *cRs* **and** *x1* ♯ Ψ **and** *x1* ♯ *cP* **and** *x1* ♯ *cRs* **and** *x1* ♯ *xvec4*
from ‹*xvec* ♯∗ *xvec4*› ‹*x* ♯ *xvec4*› ‹*x1* ♯ *xvec4*› ‹*yvec* ♯∗ *xvec4*› *D E F*
have *x* ≠ *y* **and** *x1* ≠ *y* **and** *x1* ♯ *p* ∙ *xvec* **and** *x1* ♯ *q* ∙ *yvec* **by** *simp*+
from ‹*xvec4* ♯∗ *cRs*› ‹*x1* ♯ *cRs*› *C* **have** *xvec4* ♯∗ *M* **and** *x1* ♯ *M* **by** *simp*+
moreover from ‹*cP* = ⟪*νx*⟫*P*› ‹*x* ♯ *cP*› ‹*x* ≠ *x1*› **have** ([(*x*, *x1*)] ∙ *cP*) = [(*x*, *x1*)] ∙ ⟪*νx*⟫*P*
by *simp*

with ‹$x \sharp cP$› ‹$x1 \sharp cP$› **have** $cP = (\!|\nu x1|\!)([(x, x1)] \cdot P)$ **by**(*simp add*: *eqvts calc-atm*)
moreover from C **have** $((p@(x1, x)\#q) \cdot cRs) = (p@(x1, x)\#q) \cdot (M(\!|\nu*(xvec@x\#yvec)|\!)\langle N\rangle \prec P')$ **by**(*simp add*: *fresh-star-bij*)
with *Sp Sq* ‹$xvec4 \sharp* cRs$› *D E F* ‹$xvec \sharp* cRs$› ‹$x \sharp cRs$› ‹$yvec \sharp* cRs$› ‹$xvec4 \sharp* M$› ‹$(xvec@x\#yvec) \sharp* M$› ‹$xvec \sharp* xvec4$› ‹$x \sharp xvec4$› ‹$yvec \sharp* xvec4$› ‹$xvec \sharp* yvec$› ‹$x \sharp xvec$› ‹$x \sharp yvec$› ‹$y \sharp xvec'$› ‹$y \sharp yvec'$› ‹$xvec' \sharp* yvec'$› ‹$x1 \sharp xvec$› ‹$x1 \sharp yvec$› ‹$x1 \neq y$› ‹$x1 \sharp xvec4$› ‹$x1 \sharp cRs$› ‹$x1 \sharp cRs$› ‹$x \neq x1$› ‹$x1 \sharp M$›
have $cRs = M(\!|\nu*((p \cdot xvec)@x1\#(q \cdot yvec))|\!)\langle((p@(x1, x)\#q) \cdot N)\rangle \prec ((p@(x1, x)\#q) \cdot P')$
by(*simp add*: *eqvts pt2*[*OF pt-name-inst*] *calc-atm*)
moreover from *D E F* **have** $xvec4 = (p \cdot xvec)@y\#(q \cdot yvec)$ **by** *simp*
moreover from ‹$\Psi \rhd P \longmapsto M(\!|\nu*(xvec@yvec)|\!)\langle N\rangle \prec P'$› **have** $((p@(x1, x)\#q) \cdot \Psi) \rhd ((p@(x1, x)\#q) \cdot P) \longmapsto ((p@(x1, x)\#q) \cdot (M(\!|\nu*(xvec@yvec)|\!)\langle N\rangle \prec P'))$
by(*intro eqvts*)
with *Sp Sq B C D E F* ‹$xvec4 \sharp* \Psi$› ‹$(xvec@x\#yvec) \sharp* \Psi$› ‹$xvec4 \sharp* cRs$› ‹$x \sharp xvec4$› *C D* ‹$x \sharp cRs$› ‹$yvec \sharp* cRs$› ‹$xvec4 \sharp* M$› ‹$(xvec@x\#yvec) \sharp* M$› ‹$x \sharp M$› ‹$x1 \sharp cRs$› ‹$x \neq x1$› ‹$x1 \sharp xvec$› ‹$x1 \sharp yvec$› ‹$xvec \sharp* xvec4$› ‹$yvec \sharp* xvec4$› ‹$x1 \sharp xvec4$› ‹$x \sharp xvec$› ‹$x \sharp yvec$› ‹$x1 \sharp \Psi$› ‹$xvec4 \sharp* cP$› ‹$xvec \sharp* P$› ‹$yvec \sharp* P$› ‹$xvec' \sharp* yvec'$› ‹$x1 \sharp xvec4$› ‹$xvec4 \sharp* cP$› ‹$yvec \sharp* xvec4$› ‹$xvec \sharp* xvec4$› ‹$x \neq x1$› ‹$xvec \sharp* yvec$›
have $\Psi \rhd ([(x, x1)] \cdot P) \longmapsto M(\!|\nu*((p \cdot xvec)@(q \cdot yvec))|\!)\langle((p@(x1, x)\#q) \cdot N)\rangle \prec ((p@(x1, x)\#q) \cdot P')$
by(*simp add*: *eqvts pt-fresh-bij*[*OF pt-name-inst*, *OF at-name-inst*] *pt2*[*OF pt-name-inst*] *name-swap*)

moreover from ‹$x \in supp\ N$› **have** $((p@(x1, x)\#q) \cdot x) \in ((p@(x1, x)\#q) \cdot supp\ N)$
by(*simp add*: *pt-set-bij*[*OF pt-name-inst*, *OF at-name-inst*])
then have $x1 \in supp((p@(x1, x)\#q) \cdot N)$
using ‹$x \sharp xvec$› ‹$x \sharp yvec$› ‹$x1 \sharp xvec$› ‹$x1 \sharp yvec$› ‹$x \sharp xvec4$› ‹$x1 \sharp xvec4$› ‹$xvec \sharp* xvec4$› ‹$yvec \sharp* xvec4$› ‹$xvec' \sharp* yvec'$› *D E F Sp Sq* ‹$x \neq x1$›
by(*simp add*: *eqvts pt2*[*OF pt-name-inst*] *calc-atm*)
moreover from ‹$x1 \sharp xvec4$› *D E F* **have** $x1 \sharp (p \cdot xvec)$ **and** $x1 \sharp (q \cdot yvec)$ **by** *simp+*
moreover from ‹*distinct xvec'*› ‹*distinct yvec'*› *E F* **have** $distinct(p \cdot xvec)$ **and** $distinct(q \cdot yvec)$ **by** *simp+*
moreover from ‹$xvec' \sharp* yvec'$› *E F* **have** $(p \cdot xvec) \sharp* (q \cdot yvec)$ **by** *auto*
moreover from ‹$xvec \sharp* \Psi$› **have** $(p \cdot xvec) \sharp* (p \cdot \Psi)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with *Sp D E* ‹$xvec4 \sharp* \Psi$› ‹$xvec \sharp* \Psi$› **have** $(p \cdot xvec) \sharp* \Psi$ **by**(*simp add*: *eqvts*)
moreover from ‹$yvec \sharp* \Psi$› **have** $(p \cdot yvec) \sharp* (p \cdot \Psi)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with *Sq D F* ‹$xvec4 \sharp* \Psi$› ‹$yvec \sharp* \Psi$› **have** $(q \cdot yvec) \sharp* \Psi$ **by**(*simp add*: *eqvts*)
moreover from ‹$xvec4 \sharp* cP$› ‹$x \sharp xvec4$› ‹$x1 \sharp xvec4$› *B D E F* **have** $(p \cdot xvec) \sharp* ([(x, x1)] \cdot P)$ **and** $(q \cdot yvec) \sharp* ([(x, x1)] \cdot P)$

by *simp+*
moreover from ‹*xvec4* ♯* *M*› *C D E F* **have** (*p* ∙ *xvec*) ♯* *M* **and** (*q* ∙ *yvec*) ♯* *M* **by** *simp+*
ultimately show *cP* = (|ν*x1*|)([(*x*, *x1*)] ∙ *P*) ∧
cRs = *M*(|ν*(*p* ∙ *xvec* @ *x1* # *q* ∙ *yvec*)|)⟨((*p* @ (*x1*, *x*) # *q*) ∙ *N*)⟩ ≺ (*p* @ (*x1*, *x*) # *q*) ∙ *P*′ ∧
xvec4 = *p* ∙ *xvec* @ *y* # *q* ∙ *yvec* ∧
Ψ ⊳ [(*x*, *x1*)] ∙ *P* ⟼ *M*(|ν*(*p* ∙ *xvec* @ *q* ∙ *yvec*)|)⟨((*p* @ (*x1*, *x*) # *q*) ∙ *N*)⟩ ≺ (*p* @ (*x1*, *x*) # *q*) ∙ *P*′ ∧
x1 ∈ *supp* ((*p* @ (*x1*, *x*) # *q*) ∙ *N*) ∧
x1 ♯ *p* ∙ *xvec* ∧
x1 ♯ *q* ∙ *yvec* ∧
distinct (*p* ∙ *xvec*) ∧
distinct (*q* ∙ *yvec*) ∧
(*p* ∙ *xvec*) ♯* Ψ ∧
(*p* ∙ *xvec*) ♯* ([(*x*, *x1*)] ∙ *P*) ∧
(*p* ∙ *xvec*) ♯* *M* ∧
(*p* ∙ *xvec*) ♯* (*q* ∙ *yvec*) ∧
(*q* ∙ *yvec*) ♯* Ψ ∧
(*q* ∙ *yvec*) ♯* ([(*x*, *x1*)] ∙ *P*) ∧
(*q* ∙ *yvec*) ♯* *M*
by *blast*
qed
next
case(*cBrOpen P M xvec yvec N P′ x*)
have *B*: *cP* = (|ν*x*|)*P* **and** *C*: *cRs* = ¡*M*(|ν*(*xvec*@*x*#*yvec*)|)⟨*N*⟩ ≺ *P*′
by *fact+*
from ‹*xvec* ♯* (*xvec1*, *xvec2*, *xvec3*, *xvec4*, *xvec5*, *xvec6*, *xvec7*, *xvec8*, *xvec9*, *x1*, *x2*, *x3*, *x4*, *cP*, *cRs*, *C*)› **have** *xvec* ♯* *xvec9* **and** *xvec* ♯* *cP* **and** *xvec* ♯* *cRs* **and** *x4* ♯ *xvec* **by** *simp+*
from ‹*x* ♯ (*xvec1*, *xvec2*, *xvec3*, *xvec4*, *xvec5*, *xvec6*, *xvec7*, *xvec8*, *xvec9*, *x1*, *x2*, *x3*, *x4*, *cP*, *cRs*, *C*)› **have** *x* ♯ *xvec9* **and** *x* ♯ *cP* **and** *x* ♯ *cRs* **and** *x* ≠ *x4* **by** *simp+*
from ‹*yvec* ♯* (*xvec1*, *xvec2*, *xvec3*, *xvec4*, *xvec5*, *xvec6*, *xvec7*, *xvec8*, *xvec9*, *x1*, *x2*, *x3*, *x4*, *cP*, *cRs*, *C*)› **have** *yvec* ♯* *xvec9* **and** *yvec* ♯* *cP* **and** *yvec* ♯* *cRs* **and** *x4* ♯ *yvec* **by** *simp+*

from ‹*xvec* ♯* *cRs*› ‹*x* ♯ *cRs*› ‹*yvec* ♯* *cRs*› *C* **have** (*xvec*@*x*#*yvec*) ♯* *M* **by** *simp*
from ‹*xvec* ♯* Ψ› ‹*x* ♯ Ψ› ‹*yvec* ♯* Ψ› **have** (*xvec*@*x*#*yvec*) ♯* Ψ **by** *simp*
from ‹*length xvec9* = *residualLength cRs*› *C* **obtain** *xvec′ y yvec′* **where** *D*: *xvec9* = *xvec′*@*y*#*yvec′* **and** *length xvec′* = *length xvec* **and** *length yvec′* = *length yvec*
by(*auto intro*: *lengthAux2*)
with ‹*distinct xvec*› ‹*distinct yvec*› ‹*x* ♯ *xvec*› ‹*x* ♯ *yvec*› ‹*xvec* ♯* *yvec*› ‹*xvec* ♯* *xvec9*› ‹*yvec* ♯* *xvec9*› ‹*x* ♯ *xvec9*› ‹*distinct xvec9*›
have *distinct xvec′* **and** *distinct yvec′* **and** *xvec′* ♯* *yvec′* **and** *x* ≠ *y* **and** *y* ♯ *xvec′* **and** *y* ♯ *yvec′*
and *x* ♯ *xvec′* **and** *x* ♯ *yvec′* **and** *y* ♯ *xvec* **and** *y* ♯ *yvec* **and** *xvec* ♯* *xvec′* **and**

yvec $\sharp*$ *yvec'*
by *auto*
from ‹*length xvec'* = *length xvec*› ‹*xvec* $\sharp*$ *xvec'*› ‹*distinct xvec*› ‹*distinct xvec'*›
obtain *p* **where** *Sp*: *set p* $\subseteq$ *set xvec* $\times$ *set*(*p* $\cdot$ *xvec*) **and** *distinctPerm p* **and** *E*: *xvec'* = *p* $\cdot$ *xvec*
by(*metis constructPerm*)
from ‹*length yvec'* = *length yvec*› ‹*yvec* $\sharp*$ *yvec'*› ‹*distinct yvec*› ‹*distinct yvec'*›
obtain *q* **where** *Sq*: *set q* $\subseteq$ *set yvec* $\times$ *set*(*q* $\cdot$ *yvec*) **and** *distinctPerm q* **and** *F*: *yvec'* = *q* $\cdot$ *yvec*
by(*metis constructPerm*)

show *?thesis*
proof(*rule rBrOpen*[**where** *P*=([(*x*, *x4*)] $\cdot$ *P*) **and** *xvec*=*p* $\cdot$ *xvec* **and** *y*=*y* **and** *yvec*=*q* $\cdot$ *yvec* **and** *N*=(*p*@(*x4*, *x*)#*q*) $\cdot$ *N* **and** *P'*=(*p*@(*x4*, *x*)#*q*) $\cdot$ *P'* **and** *M*=*M*])
assume *xvec9* $\sharp*$ Ψ **and** *xvec9* $\sharp*$ *cP* **and** *xvec9* $\sharp*$ *cRs* **and** *x4* $\sharp$ Ψ **and** *x4* $\sharp$ *cP* **and** *x4* $\sharp$ *cRs* **and** *x4* $\sharp$ *xvec9*
from ‹*xvec* $\sharp*$ *xvec9*› ‹*x* $\sharp$ *xvec9*› ‹*x4* $\sharp$ *xvec9*› ‹*yvec* $\sharp*$ *xvec9*› *D E F*
have *x* $\neq$ *y* **and** *x4* $\neq$ *y* **and** *x4* $\sharp$ *p* $\cdot$ *xvec* **and** *x4* $\sharp$ *q* $\cdot$ *yvec* **by** *simp+*
from ‹*xvec9* $\sharp*$ *cRs*› ‹*x4* $\sharp$ *cRs*› *C* **have** *xvec9* $\sharp*$ *M* **and** *x4* $\sharp$ *M* **by** *simp+*
moreover from ‹*cP* = $(\!|\nu x|\!)P$› ‹*x* $\sharp$ *cP*› ‹*x* $\neq$ *x4*› **have** ([(*x*, *x4*)] $\cdot$ *cP*) = [(*x*, *x4*)] $\cdot$ $(\!|\nu x|\!)P$
by *simp*
with ‹*x* $\sharp$ *cP*› ‹*x4* $\sharp$ *cP*› **have** *cP* = $(\!|\nu x4|\!)$([(*x*, *x4*)] $\cdot$ *P*) **by**(*simp add*: *eqvts calc-atm*)
moreover from *C* **have** ((*p*@(*x4*, *x*)#*q*) $\cdot$ *cRs*) = (*p*@(*x4*, *x*)#*q*) $\cdot$ (¡*M*$(\!|\nu*(xvec@x\#yvec)|\!)\langle N\rangle \prec P'$) **by**(*simp add*: *fresh-star-bij*)
with *Sp Sq* ‹*xvec9* $\sharp*$ *cRs*› *D E F* ‹*xvec* $\sharp*$ *cRs*› ‹*x* $\sharp$ *cRs*› ‹*yvec* $\sharp*$ *cRs*› ‹*xvec9* $\sharp*$ *M*› ‹(*xvec@x*#*yvec*) $\sharp*$ *M*› ‹*xvec* $\sharp*$ *xvec9*› ‹*x* $\sharp$ *xvec9*› ‹*yvec* $\sharp*$ *xvec9*› ‹*xvec* $\sharp*$ *yvec*› ‹*x* $\sharp$ *xvec*› ‹*x* $\sharp$ *yvec*› ‹*y* $\sharp$ *xvec'*› ‹*y* $\sharp$ *yvec'*› ‹*xvec'* $\sharp*$ *yvec'*› ‹*x4* $\sharp$ *xvec*› ‹*x4* $\sharp$ *yvec*› ‹*x4* $\neq$ *y*› ‹*x4* $\sharp$ *xvec9*› ‹*x4* $\sharp$ *cRs*› ‹*x4* $\sharp$ *cRs*› ‹*x* $\neq$ *x4*› ‹*x4* $\sharp$ *M*›
have *cRs* = ¡*M*$(\!|\nu*((p \cdot xvec)@x4\#(q \cdot yvec))|\!)\langle((p@(x4, x)\#q) \cdot N)\rangle \prec ((p@(x4, x)\#q) \cdot P')$
by(*simp add*: *eqvts pt2*[*OF pt-name-inst*] *calc-atm*)
moreover from *D E F* **have** *xvec9* = (*p* $\cdot$ *xvec*)@*y*#(*q* $\cdot$ *yvec*) **by** *simp*
moreover from ‹$\Psi \rhd P \longmapsto$¡*M*$(\!|\nu*(xvec@yvec)|\!)\langle N\rangle \prec P'$› **have** ((*p*@(*x4*, *x*)#*q*) $\cdot$ Ψ) $\rhd$ ((*p*@(*x4*, *x*)#*q*) $\cdot$ *P*) $\longmapsto$((*p*@(*x4*, *x*)#*q*) $\cdot$ (¡*M*$(\!|\nu*(xvec@yvec)|\!)\langle N\rangle \prec P'$))
by(*intro eqvts*)
with *Sp Sq B C D E F* ‹*xvec9* $\sharp*$ Ψ› ‹(*xvec@x*#*yvec*) $\sharp*$ Ψ› ‹*xvec9* $\sharp*$ *cRs*› ‹*x* $\sharp$ *xvec9*› *C D* ‹*x* $\sharp$ *cRs*› ‹*yvec* $\sharp*$ *cRs*› ‹*xvec9* $\sharp*$ *M*› ‹(*xvec@x*#*yvec*) $\sharp*$ *M*› ‹*x* $\sharp$ *M*› ‹*x4* $\sharp$ *cRs*› ‹*x* $\neq$ *x4*› ‹*x4* $\sharp$ *xvec*› ‹*x4* $\sharp$ *yvec*› ‹*xvec* $\sharp*$ *xvec9*› ‹*yvec* $\sharp*$ *xvec9*› ‹*x4* $\sharp$ *xvec9*› ‹*x* $\sharp$ *xvec*› ‹*x* $\sharp$ *yvec*› ‹*x4* $\sharp$ Ψ› ‹*xvec9* $\sharp*$ *cP*› ‹*xvec* $\sharp*$ *P*› ‹*yvec* $\sharp*$ *P*› ‹*xvec'* $\sharp*$ *yvec'*› ‹*x4* $\sharp$ *xvec9*› ‹*xvec9* $\sharp*$ *cP*› ‹*yvec* $\sharp*$ *xvec9*› ‹*xvec* $\sharp*$ *xvec9*› ‹*x* $\neq$ *x4*› ‹*xvec* $\sharp*$ *yvec*›
have $\Psi \rhd ([(x, x4)] \cdot P) \longmapsto$¡*M*$(\!|\nu*((p \cdot xvec)@(q \cdot yvec))|\!)\langle((p@(x4, x)\#q) \cdot N)\rangle \prec ((p@(x4, x)\#q) \cdot P')$
by(*simp add*: *eqvts pt-fresh-bij*[*OF pt-name-inst*, *OF at-name-inst*] *pt2*[*OF pt-name-inst*] *name-swap*)

moreover from ‹$x \in supp\ N$› **have** $((p@(x4,\ x)\#q) \cdot x) \in ((p@(x4,\ x)\#q) \cdot supp\ N)$
by(*simp add*: *pt-set-bij*[*OF pt-name-inst*, *OF at-name-inst*])
then have $x4 \in supp((p@(x4,\ x)\#q) \cdot N)$
using ‹$x \sharp xvec$› ‹$x \sharp yvec$› ‹$x4 \sharp xvec$› ‹$x4 \sharp yvec$› ‹$x \sharp xvec9$› ‹$x4 \sharp xvec9$› ‹$xvec \sharp* xvec9$› ‹$yvec \sharp* xvec9$› ‹$xvec' \sharp* yvec'$› *D E F Sp Sq* ‹$x \neq x4$›
by(*simp add*: *eqvts pt2*[*OF pt-name-inst*] *calc-atm*)
moreover from ‹$x4 \sharp xvec9$› *D E F* **have** $x4 \sharp (p \cdot xvec)$ **and** $x4 \sharp (q \cdot yvec)$ **by** *simp+*
moreover from ‹*distinct xvec'*› ‹*distinct yvec'*› *E F* **have** $distinct(p \cdot xvec)$ **and** $distinct(q \cdot yvec)$ **by** *simp+*
moreover from ‹$xvec' \sharp* yvec'$› *E F* **have** $(p \cdot xvec) \sharp* (q \cdot yvec)$ **by** *auto*
moreover from ‹$xvec \sharp* \Psi$› **have** $(p \cdot xvec) \sharp* (p \cdot \Psi)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with *Sp D E* ‹$xvec9 \sharp* \Psi$› ‹$xvec \sharp* \Psi$› **have** $(p \cdot xvec) \sharp* \Psi$ **by**(*simp add*: *eqvts*)
moreover from ‹$yvec \sharp* \Psi$› **have** $(p \cdot yvec) \sharp* (p \cdot \Psi)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with *Sq D F* ‹$xvec9 \sharp* \Psi$› ‹$yvec \sharp* \Psi$› **have** $(q \cdot yvec) \sharp* \Psi$ **by**(*simp add*: *eqvts*)
moreover from ‹$xvec9 \sharp* cP$› ‹$x \sharp xvec9$› ‹$x4 \sharp xvec9$› *B D E F* **have** $(p \cdot xvec) \sharp* ([(x,\ x4)] \cdot P)$ **and** $(q \cdot yvec) \sharp* ([(x,\ x4)] \cdot P)$
by *simp+*
moreover from ‹$xvec9 \sharp* M$› *C D E F* **have** $(p \cdot xvec) \sharp* M$ **and** $(q \cdot yvec) \sharp* M$ **by** *simp+*
ultimately show $cP = (\!|\nu x4|\!)([(x,\ x4)] \cdot P) \wedge$
$cRs = ¡M(\!|\nu*(p \cdot xvec\ @\ x4\ \#\ q \cdot yvec)|\!)\langle((p\ @\ (x4,\ x)\ \#\ q) \cdot N)\rangle \prec (p\ @\ (x4,\ x)\ \#\ q) \cdot P' \wedge$
$xvec9 = p \cdot xvec\ @\ y\ \#\ q \cdot yvec \wedge$
$\Psi \rhd [(x,\ x4)] \cdot P \longmapsto ¡M(\!|\nu*(p \cdot xvec\ @\ q \cdot yvec)|\!)\langle((p\ @\ (x4,\ x)\ \#\ q) \cdot N)\rangle \prec (p\ @\ (x4,\ x)\ \#\ q) \cdot P' \wedge$
$x4 \in supp\ ((p\ @\ (x4,\ x)\ \#\ q) \cdot N) \wedge$
$x4 \sharp p \cdot xvec \wedge$
$x4 \sharp q \cdot yvec \wedge$
$distinct\ (p \cdot xvec) \wedge$
$distinct\ (q \cdot yvec) \wedge$
$(p \cdot xvec) \sharp* \Psi \wedge$
$(p \cdot xvec) \sharp* ([(x,\ x4)] \cdot P) \wedge$
$(p \cdot xvec) \sharp* M \wedge$
$(p \cdot xvec) \sharp* (q \cdot yvec) \wedge$
$(q \cdot yvec) \sharp* \Psi \wedge$
$(q \cdot yvec) \sharp* ([(x,\ x4)] \cdot P) \wedge$
$(q \cdot yvec) \sharp* M$
by *blast*
qed
next
case(*cScope P* α *P' x*)
have *B*: $cP = (\!|\nu x|\!)P$ **and** *C*: $cRs = \alpha \prec (\!|\nu x|\!)P'$

by *fact+*
from ‹*bn* α $\sharp*$ (*xvec1*, *xvec2*, *xvec3*, *xvec4*, *xvec5*, *xvec6*, *xvec7*, *xvec8*, *xvec9*, *x1*, *x2*, *x3*, *x4*, *cP*, *cRs*, *C*)› **have** *bn* α $\sharp*$ *xvec5* **and** *x2* $\sharp$ *bn* α **by** *simp+*
from ‹*x* $\sharp$ (*xvec1*, *xvec2*, *xvec3*, *xvec4*, *xvec5*, *xvec6*, *xvec7*, *xvec8*, *xvec9*, *x1*, *x2*, *x3*, *x4*, *cP*, *cRs*, *C*)› **have** *x* $\sharp$ *xvec5* **and** *x* $\neq$ *x2* **and** *x* $\sharp$ *cRs* **by** *simp+*

from ‹*length xvec5* = *residualLength cRs*› *C* **have** *length xvec5* = *length*(*bn* α)
by *simp*
then obtain *p* **where** *S*: *set p* $\subseteq$ *set*(*bn* α) $\times$ *set*(*bn*(*p* $\cdot$ α)) **and** *distinctPerm p* **and** *xvec5*= *bn*(*p* $\cdot$ α)
using ‹*bn* α $\sharp*$ *xvec5*› ‹*distinct*(*bn* α)› ‹*distinct xvec5*›
by − (*rule constructPerm*[**where** *xvec*=*bn* α **and** *yvec*=*xvec5*], *auto simp add*: *eqvts*)
show *?thesis*
proof(*rule rScope*[**where** *P*=[(*x*, *x2*)] $\cdot$ *P* **and** α=[(*x*, *x2*)] $\cdot$ *p* $\cdot$ α **and** *P′*=[(*x*, *x2*)] $\cdot$ *p* $\cdot$ *P′*])
assume *xvec5* $\sharp*$ Ψ **and** *xvec5* $\sharp*$ *cP* **and** *xvec5* $\sharp*$ *cRs* **and** *x2* $\sharp$ Ψ **and** *x2* $\sharp$ *cP* **and** *x2* $\sharp$ *cRs* **and** *x2* $\sharp$ *xvec5*
from ‹*x2* $\sharp$ *cRs*› *C* ‹*x2* $\sharp$ *bn* α› ‹*x* $\neq$ *x2*› **have** *x2* $\sharp$ α **and** *x2* $\sharp$ *P′* **by**(*auto simp add*: *abs-fresh*)
moreover from ‹*cP* = (ν*x*)*P*› ‹*x2* $\sharp$ *cP*› ‹*x* $\neq$ *x2*› **have** *cP* = (ν*x2*)([(*x*, *x2*)] $\cdot$ *P*)
by(*simp add*: *alphaRes abs-fresh*)
moreover from *B C S* ‹*bn* α $\sharp*$ *xvec5*› ‹*xvec5* $\sharp*$ *cRs*› ‹*xvec5* = *bn*(*p* $\cdot$ α)› ‹*bn* α $\sharp*$ *subject* α› ‹*xvec5* $\sharp*$ *cP*› ‹*x* $\sharp$ α› ‹*x* $\sharp$ *xvec5*›
have *cRs* = (*p* $\cdot$ α) $\prec$ (ν*x*)(*p* $\cdot$ *P′*)
apply *clarsimp*
by(*subst residualAlpha*[**where** *p*=*p*] *alphaRes*) (*auto simp del*: *actionFresh*)
then have ([(*x*, *x2*)] $\cdot$ *cRs*) = [(*x*, *x2*)] $\cdot$ ((*p* $\cdot$ α) $\prec$ (ν*x*)(*p* $\cdot$ *P′*))
by *simp*
with ‹*x2* $\sharp$ *cRs*› ‹*x* $\sharp$ *cRs*› **have** *cRs* = ([(*x*, *x2*)] $\cdot$ *p* $\cdot$ α) $\prec$ (ν*x2*)([(*x*, *x2*)] $\cdot$ *p* $\cdot$ *P′*)
by(*simp add*: *eqvts calc-atm*)
moreover from ‹*xvec5*= *bn*(*p* $\cdot$ α)› **have** ([(*x*, *x2*)] $\cdot$ *xvec5*) = ([(*x*, *x2*)] $\cdot$ *bn*(*p* $\cdot$ α))
by *simp*
with ‹*x* $\sharp$ *xvec5*› ‹*x2* $\sharp$ *xvec5*› **have** *xvec5* = *bn*([(*x*, *x2*)] $\cdot$ *p* $\cdot$ α)
by(*simp add*: *eqvts*)
moreover from ‹Ψ $\rhd$ *P* $\longmapsto$$\alpha$ $\prec$ *P′*› *S B C S* ‹*bn* α $\sharp*$ *xvec5*› ‹*xvec5* $\sharp*$ *cRs*› ‹*xvec5* = *bn*(*p* $\cdot$ α)› ‹*bn* α $\sharp*$ *subject* α› ‹*xvec5* $\sharp*$ *cP*› ‹*x* $\sharp$ *xvec5*›
have Ψ $\rhd$ *P* $\longmapsto$(*p* $\cdot$ α) $\prec$ (*p* $\cdot$ *P′*)
by(*subst residualAlpha*[*symmetric*]) *auto*
then have ([(*x*, *x2*)] $\cdot$ Ψ) $\rhd$ ([(*x*, *x2*)] $\cdot$ *P*) $\longmapsto$([(*x*, *x2*)] $\cdot$ ((*p* $\cdot$ α) $\prec$ (*p* $\cdot$ *P′*)))
by(*rule eqvt*)
with ‹*x* $\sharp$ Ψ› ‹*x2* $\sharp$ Ψ› **have** Ψ $\rhd$ ([(*x*, *x2*)] $\cdot$ *P*) $\longmapsto$([(*x*, *x2*)] $\cdot$ *p* $\cdot$ α) $\prec$ ([(*x*, *x2*)] $\cdot$ *p* $\cdot$ *P′*)
by(*simp add*: *eqvts*)
moreover note ‹*x2* $\sharp$ Ψ›
moreover from ‹*x* $\sharp$ α› ‹*x2* $\sharp$ α› ‹*x* $\sharp$ *xvec5*› ‹*x2* $\sharp$ *xvec5*› *S* ‹*x* $\neq$ *x2*› ‹*xvec5*

```
= bn(p · α)› have x2 ♯ [(x, x2)] · p · α
    apply(subgoal-tac x ♯ p ∧ x2 ♯ p)
     apply(simp add: perm-compose freshChainSimps del: actionFresh)
    by(auto dest: freshAlphaSwap)
  moreover from ‹bn α ♯* subject α› have ([(x, x2)] · p · (bn α)) ♯* ([(x, x2)]
· p · (subject α))
    by(simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst])
  then have bn([(x, x2)] · p · α) ♯* subject([(x, x2)] · p · α)
    by(simp add: eqvts)
  moreover from ‹distinct(bn α)› have distinct([(x, x2)] · p · (bn α)) by simp
  then have distinct(bn([(x, x2)] · p · α)) by(simp add: eqvts)
  ultimately show cP = (⦇νx2⦈)([(x, x2)] · P) ∧
    cRs = ([(x, x2)] · p · α) ≺ (⦇νx2⦈)([(x, x2)] · p · P′) ∧
    xvec5 = bn ([(x, x2)] · p · α) ∧
    Ψ ▷ [(x, x2)] · P ⟼ ([(x, x2)] · p · α) ≺ [(x, x2)] · p · P′ ∧
    x2 ♯ Ψ ∧
    x2 ♯ [(x, x2)] · p · α ∧
    bn ([(x, x2)] · p · α) ♯* subject ([(x, x2)] · p · α) ∧
    distinct (bn ([(x, x2)] · p · α)) by blast
 qed
next
 case(cBang P)
 then show ?thesis by(auto intro: rBang)
qed

lemma resResidEq:
 fixes xvec :: name list
   and P    :: ('a, 'b, 'c) psi
   and Q    :: ('a, 'b, 'c) psi
   and M    :: 'a
   and N    :: 'a
   and N′   :: 'a

assumes ¡M(⦇ν*xvec⦈)⟨N⟩ ≺ P = ¡M(⦇ν*xvec⦈)⟨N′⟩ ≺ Q
 and    xvec ♯* M

shows ¡M(⦇ν*xvec⦈)⟨N⟩ = ¡M(⦇ν*xvec⦈)⟨N′⟩
 using assms
proof(induct xvec)
 case Nil
 then show ?case by(simp add: residualInject)
next
 case (Cons x xvec)
 from ‹(x # xvec) ♯* M›
 have x ♯ M and xvec ♯* M by simp+
 from ‹¡M(⦇ν*(x # xvec)⦈)⟨N⟩ ≺ P = ¡M(⦇ν*(x # xvec)⦈)⟨N′⟩ ≺ Q› ‹x ♯ M›
 have ¡M(⦇ν*(xvec)⦈)⟨N⟩ ≺ P = ¡M(⦇ν*(xvec)⦈)⟨N′⟩ ≺ Q
   by (metis action.inject(4) assms(1) bn.simps(4) residualInject′′)
 then have ¡M(⦇ν*xvec⦈)⟨N⟩ = ¡M(⦇ν*xvec⦈)⟨N′⟩ using ‹xvec ♯* M›
```

by(*rule Cons*(*1*))
then show *?case*
by(*simp add*: *action.inject*)
qed

lemma *parCases*[*consumes 5*, *case-names cPar1 cPar2 cComm1 cComm2 cBrMerge cBrComm1 cBrComm2*]:
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 :: *'f*::*fs-name*
and M :: *'a*
and N :: *'a*

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*: $\bigwedge P'\ A_Q\ \Psi_Q.\ [\![\Psi \otimes \Psi_Q \rhd P \longmapsto \alpha \prec P';\ extractFrame\ Q = \langle A_Q, \Psi_Q\rangle;\ distinct\ A_Q;$
$A_Q \sharp^* \Psi;\ A_Q \sharp^* P;\ A_Q \sharp^* Q;\ A_Q \sharp^* \alpha;\ A_Q \sharp^* P';\ A_Q \sharp^* C]\!] \Longrightarrow Prop\ \alpha\ (P' \parallel Q)$
and *rPar2*: $\bigwedge Q'\ A_P\ \Psi_P.\ [\![\Psi \otimes \Psi_P \rhd Q \longmapsto \alpha \prec Q';\ extractFrame\ P = \langle A_P, \Psi_P\rangle;\ distinct\ A_P;$
$A_P \sharp^* \Psi;\ A_P \sharp^* P;\ A_P \sharp^* Q;\ A_P \sharp^* \alpha;\ A_P \sharp^* Q';\ A_P \sharp^* C]\!] \Longrightarrow Prop\ \alpha\ (P \parallel Q')$
and *rComm1*: $\bigwedge \Psi_Q\ M\ N\ P'\ A_P\ \Psi_P\ K\ xvec\ Q'\ A_Q.$
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|N|\!) \prec P';\ extractFrame\ P = \langle A_P, \Psi_P\rangle;\ distinct\ A_P;$
$\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|\nu * xvec|\!)\langle N\rangle \prec Q';\ extractFrame\ Q = \langle A_Q, \Psi_Q\rangle;\ distinct\ A_Q;$
$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K;\ distinct\ xvec;\ \alpha = \tau;$
$A_P \sharp^* \Psi;\ A_P \sharp^* \Psi_Q;\ A_P \sharp^* P;\ A_P \sharp^* M;\ A_P \sharp^* N;\ A_P \sharp^* P';\ A_P \sharp^* Q;\ A_P \sharp^* xvec;\ A_P \sharp^* Q';\ A_P \sharp^* A_Q;\ A_P \sharp^* C;$
$A_Q \sharp^* \Psi;\ A_Q \sharp^* \Psi_P;\ A_Q \sharp^* P;\ A_Q \sharp^* K;\ A_Q \sharp^* N;\ A_Q \sharp^* P';\ A_Q \sharp^* Q;\ A_Q \sharp^* xvec;\ A_Q \sharp^* Q';\ A_Q \sharp^* C;$
$xvec \sharp^* \Psi;\ xvec \sharp^* \Psi_P;\ xvec \sharp^* P;\ xvec \sharp^* M;\ xvec \sharp^* K;\ xvec \sharp^* Q;\ xvec \sharp^* \Psi_Q;\ xvec \sharp^* C]\!] \Longrightarrow$
$Prop\ (\tau)\ ((\!|\nu * xvec|\!)(P' \parallel Q'))$
and *rComm2*: $\bigwedge \Psi_Q\ M\ xvec\ N\ P'\ A_P\ \Psi_P\ K\ Q'\ A_Q.$
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P';\ extractFrame\ P = \langle A_P, \Psi_P\rangle;\ distinct\ A_P;$
$\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|N|\!) \prec Q';\ extractFrame\ Q = \langle A_Q, \Psi_Q\rangle;\ distinct\ A_Q;$
$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K;\ distinct\ xvec;\ \alpha = \tau;$
$A_P \sharp^* \Psi;\ A_P \sharp^* \Psi_Q;\ A_P \sharp^* P;\ A_P \sharp^* M;\ A_P \sharp^* N;\ A_P \sharp^* P';\ A_P \sharp^* Q;\ A_P \sharp^* xvec;\ A_P \sharp^* Q';\ A_P \sharp^* A_Q;\ A_P \sharp^* C;$
$A_Q \sharp^* \Psi;\ A_Q \sharp^* \Psi_P;\ A_Q \sharp^* P;\ A_Q \sharp^* K;\ A_Q \sharp^* N;\ A_Q \sharp^* P';\ A_Q \sharp^*$

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 *rBrMerge*: $\bigwedge \Psi_Q\ M\ N\ P'\ A_P\ \Psi_P\ Q'\ A_Q$.
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto {¿}M(\!|N|\!) \prec P'$; *extractFrame* $P = \langle A_P, \Psi_P\rangle$; *distinct* A_P;
$\Psi \otimes \Psi_P \rhd Q \longmapsto {¿}M(\!|N|\!) \prec Q'$; *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$; *distinct* A_Q;
$A_P \sharp* \Psi$; $A_P \sharp* \Psi_Q$; $A_P \sharp* P$; $A_P \sharp* N$; $A_P \sharp* P'$;
$A_P \sharp* Q$; $A_P \sharp* Q'$; $A_P \sharp* A_Q$; $A_P \sharp* M$; $A_Q \sharp* M$;
$A_Q \sharp* \Psi$; $A_Q \sharp* \Psi_P$; $A_Q \sharp* P$; $A_Q \sharp* N$; $A_Q \sharp* P'$;
$A_Q \sharp* Q$; $A_Q \sharp* Q'$; $A_P \sharp* C$; $A_Q \sharp* C$; $\alpha = {¿}M(\!|N|\!)]\!] \Longrightarrow$
Prop (${¿}M(\!|N|\!)$) ($P' \parallel Q'$)

and *rBrComm1*: $\bigwedge \Psi_Q\ M\ N\ P'\ A_P\ \Psi_P\ xvec\ Q'\ A_Q$.
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto {¿}M(\!|N|\!) \prec P'$; *extractFrame* $P = \langle A_P, \Psi_P\rangle$; *distinct* A_P;
$\Psi \otimes \Psi_P \rhd Q \longmapsto {¡}M(\!|\nu{*}xvec|\!)\langle N\rangle \prec Q'$; *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$; *distinct* A_Q;
distinct *xvec*;
$A_P \sharp* \Psi$; $A_P \sharp* \Psi_Q$; $A_P \sharp* P$; $A_P \sharp* N$; $A_P \sharp* P'$; $A_P \sharp* Q$; $A_P \sharp* xvec$; $A_P \sharp* Q'$; $A_P \sharp* A_Q$; $A_P \sharp* C$;
$A_Q \sharp* \Psi$; $A_Q \sharp* \Psi_P$; $A_Q \sharp* P$; $A_Q \sharp* N$; $A_Q \sharp* P'$; $A_Q \sharp* Q$; $A_Q \sharp* xvec$; $A_Q \sharp* Q'$; $A_Q \sharp* C$;
$A_P \sharp* M$; $A_Q \sharp* M$; $xvec \sharp* M$; ${¡}M(\!|\nu{*}xvec|\!)\langle N\rangle = \alpha$;
$xvec \sharp* \Psi$; $xvec \sharp* \Psi_P$; $xvec \sharp* P$; $xvec \sharp* Q$; $xvec \sharp* \Psi_Q]\!] \Longrightarrow$
Prop (${¡}M(\!|\nu{*}xvec|\!)\langle N\rangle$) ($P' \parallel Q'$)

and *rBrComm2*: $\bigwedge \Psi_Q\ M\ xvec\ N\ P'\ A_P\ \Psi_P\ Q'\ A_Q$.
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto {¡}M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P'$; *extractFrame* $P = \langle A_P, \Psi_P\rangle$; *distinct* A_P;
$\Psi \otimes \Psi_P \rhd Q \longmapsto {¿}M(\!|N|\!) \prec Q'$; *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$; *distinct* A_Q;
distinct *xvec*;
$A_P \sharp* \Psi$; $A_P \sharp* \Psi_Q$; $A_P \sharp* P$; $A_P \sharp* N$; $A_P \sharp* P'$; $A_P \sharp* Q$; $A_P \sharp* xvec$; $A_P \sharp* Q'$; $A_P \sharp* A_Q$; $A_P \sharp* C$;
$A_Q \sharp* \Psi$; $A_Q \sharp* \Psi_P$; $A_Q \sharp* P$; $A_Q \sharp* N$; $A_Q \sharp* P'$; $A_Q \sharp* Q$; $A_Q \sharp* xvec$; $A_Q \sharp* Q'$; $A_Q \sharp* C$;
$A_P \sharp* M$; $A_Q \sharp* M$; $xvec \sharp* M$; ${¡}M(\!|\nu{*}xvec|\!)\langle N\rangle = \alpha$;
$xvec \sharp* \Psi$; $xvec \sharp* \Psi_P$; $xvec \sharp* P$; $xvec \sharp* Q$; $xvec \sharp* \Psi_Q]\!] \Longrightarrow$
Prop (${¡}M(\!|\nu{*}xvec|\!)\langle N\rangle$) ($P' \parallel Q'$)

shows *Prop* α T
proof −
from *Trans* **have** *distinct*(*bn* α) **by**(*auto dest*: *boundOutputDistinct*)
have *length*(*bn* α) = *residualLength*($\alpha \prec T$) **by** *simp*
note *Trans*
moreover have *length* [] = *inputLength*($P \parallel Q$) **and** *distinct* []
by(*auto simp add*: *inputLength-inputLength′-inputLength″.simps*)
moreover have *length* [] = *inputLength*($P \parallel Q$) **and** *distinct* []

by(*auto simp add*: *inputLength-inputLength'-inputLength''.simps*)
moreover note ‹*length*(*bn* α) = *residualLength*($\alpha \prec T$)› ‹*distinct*(*bn* α)›
moreover note ‹*length*(*bn* α) = *residualLength*($\alpha \prec T$)› ‹*distinct*(*bn* α)›
moreover note ‹*length*(*bn* α) = *residualLength*($\alpha \prec T$)› ‹*distinct*(*bn* α)›
moreover note ‹*length*(*bn* α) = *residualLength*($\alpha \prec T$)› ‹*distinct*(*bn* α)›
moreover note ‹*length*(*bn* α) = *residualLength*($\alpha \prec T$)› ‹*distinct*(*bn* α)›
moreover note ‹*length*(*bn* α) = *residualLength*($\alpha \prec T$)› ‹*distinct*(*bn* α)›
moreover note ‹*length*(*bn* α) = *residualLength*($\alpha \prec T$)› ‹*distinct*(*bn* α)›
moreover obtain *x*::*name* **where** $x \sharp \Psi$ **and** $x \sharp P$ **and** $x \sharp Q$ **and** $x \sharp \alpha$ **and** $x \sharp T$
by(*generate-fresh name*) *auto*
ultimately show *?thesis* **using** ‹*bn* $\alpha \sharp* \Psi$› ‹*bn* $\alpha \sharp* P$› ‹*bn* $\alpha \sharp* Q$› ‹*bn* $\alpha \sharp*$ *subject* α›
proof(*cases rule*: *semanticsCases*[*of* - - - - - - - - - - - - - *C x x x x*])
case *cInput*
then show *?thesis*
by(*simp add*: *residualInject*)
next
case *cBrInput*
then show *?thesis*
by(*simp add*: *residualInject*)
next
case *cOutput*
then show *?thesis*
by(*simp add*: *residualInject*)
next
case *cBrOutput*
then show *?thesis*
by(*simp add*: *residualInject*)
next
case *cCase*
then show *?thesis*
by(*simp add*: *residualInject*)
next
case (*cPar1* Ψ_Q *P* α' *P'* *Q* A_Q)
then show *?thesis* **using** *assms*
by(*force simp add*: *psi.inject residualInject residualInject' intro*: *rPar1*)
next
case (*cPar2* Ψ_P *Q* α' *Q'* *P* A_P)
then show *?thesis* **using** *assms*
by(*force simp add*: *psi.inject residualInject residualInject' intro*: *rPar1*)
next
case *cComm1*
then show *?thesis* **using** *assms*
by(*force simp add*: *psi.inject residualInject residualInject' intro*: *rComm1*)
next
case *cComm2*
then show *?thesis* **using** *assms*
by(*force simp add*: *psi.inject residualInject residualInject' intro*: *rComm2*)

next
case *cBrMerge*
then show *?thesis* **using** *assms*
by(*force simp add*: *psi.inject residualInject residualInject′ intro*: *rBrMerge*)
next
case (*cBrComm1* Ψ_Q *P1 M N P′* A_P Ψ_P *Q1 Q′* A_Q)
note ‹*bn* α $\sharp*$ Ψ›
moreover from ‹*bn* α $\sharp*$ *P*› ‹*bn* α $\sharp*$ *Q*› **have** *bn* α $\sharp*$ (*P* ∥ *Q*) **by** *simp*
moreover from ‹*bn* α $\sharp*$ *subject* α› **have** *bn* α $\sharp*$ (α ≺ *T*) **by** *simp*
ultimately have *all*:
P ∥ *Q* = *P1* ∥ *Q1* ∧
α ≺ *T* = ¡*M*⦇ν∗*bn* α⦈⟨*N*⟩ ≺ *P′* ∥ *Q′* ∧
Ψ ⊗ Ψ_Q ⊳ *P1* ⟼ ¿*M*⦇*N*⦈ ≺ *P′* ∧
extractFrame P1 = ⟨A_P, Ψ_P⟩ ∧
distinct A_P ∧
Ψ ⊗ Ψ_P ⊳ *Q1* ⟼ ¡*M*⦇ν∗*bn* α⦈⟨*N*⟩ ≺ *Q′* ∧
extractFrame Q1 = ⟨A_Q, Ψ_Q⟩ ∧
distinct A_Q ∧
A_P $\sharp*$ Ψ ∧
A_P $\sharp*$ Ψ_Q ∧
A_P $\sharp*$ *P1* ∧
A_P $\sharp*$ *N* ∧
A_P $\sharp*$ *P′* ∧
A_P $\sharp*$ *Q1* ∧
A_P $\sharp*$ *Q′* ∧
A_P $\sharp*$ A_Q ∧
A_P $\sharp*$ *bn* α ∧
A_Q $\sharp*$ Ψ ∧
A_Q $\sharp*$ Ψ_P ∧
A_Q $\sharp*$ *P1* ∧
A_Q $\sharp*$ *N* ∧
A_Q $\sharp*$ *P′* ∧
A_Q $\sharp*$ *Q1* ∧
A_Q $\sharp*$ *Q′* ∧
A_Q $\sharp*$ *bn* α ∧
bn α $\sharp*$ Ψ ∧
bn α $\sharp*$ Ψ_P ∧
bn α $\sharp*$ Ψ_Q ∧
bn α $\sharp*$ *P1* ∧
bn α $\sharp*$ *Q1* ∧
A_P $\sharp*$ *M* ∧ A_Q $\sharp*$ *M* ∧ *bn* α $\sharp*$ *M* ∧ A_P $\sharp*$ *C* ∧ A_Q $\sharp*$ *C* ∧ *distinct* (*bn* α)
by(*rule cBrComm1*(*1*))

from *all* **have** *bn* α $\sharp*$ *M* **and** α ≺ *T* = ¡*M*⦇ν∗*bn* α⦈⟨*N*⟩ ≺ *P′* ∥ *Q′*
by *simp*+

from ‹α ≺ *T* = ¡*M*⦇ν∗*bn* α⦈⟨*N*⟩ ≺ *P′* ∥ *Q′*› **have** α ≺ *T* = *RBrOut M* (⦇ν∗*bn* α⦈*N* ≺′ (*P′* ∥ *Q′*))
by(*simp add*: *residualInject*)

then obtain *xvec* N' **where** $\alpha = ¡M(\!|\nu{*}xvec|\!)\langle N'\rangle$
by(*auto simp add*: *residualInject*)
then have *bn* α = *xvec* **by** *simp*
from ‹$\alpha = ¡M(\!|\nu{*}xvec|\!)\langle N'\rangle$› ‹$\alpha \prec T = ¡M(\!|\nu{*}bn\ \alpha|\!)\langle N\rangle \prec P' \parallel Q'$› **have** *resEq*: $¡M(\!|\nu{*}xvec|\!)\langle N'\rangle \prec T = ¡M(\!|\nu{*}bn\ \alpha|\!)\langle N\rangle \prec P' \parallel Q'$
by *simp*
then have $¡M(\!|\nu{*}bn\ \alpha|\!)\langle N'\rangle \prec T = ¡M(\!|\nu{*}bn\ \alpha|\!)\langle N\rangle \prec P' \parallel Q'$ **using** ‹*bn* α = *xvec*›
by *simp*
then have $¡M(\!|\nu{*}bn\ \alpha|\!)\langle N'\rangle = ¡M(\!|\nu{*}bn\ \alpha|\!)\langle N\rangle$ **using** ‹*bn* $\alpha \sharp* M$›
by(*rule resResidEq*)
with ‹$\alpha = ¡M(\!|\nu{*}xvec|\!)\langle N'\rangle$› **have** $\alpha = ¡M(\!|\nu{*}xvec|\!)\langle N\rangle$ **by** *simp*

moreover from *all* **have** $P \parallel Q = P1 \parallel Q1$
and $\alpha \prec T = ¡M(\!|\nu{*}bn\ \alpha|\!)\langle N\rangle \prec P' \parallel Q'$
and $\Psi \otimes \Psi_Q \rhd P1 \longmapsto ¿M(\!|N|\!) \prec P'$
and *extractFrame* $P1 = \langle A_P, \Psi_P\rangle$
and *distinct* A_P
and $\Psi \otimes \Psi_P \rhd Q1 \longmapsto ¡M(\!|\nu{*}bn\ \alpha|\!)\langle N\rangle \prec Q'$
and *extractFrame* $Q1 = \langle A_Q, \Psi_Q\rangle$
and *distinct* A_Q
and $A_P \sharp* \Psi$
and $A_P \sharp* \Psi_Q$
and $A_P \sharp* P1$
and $A_P \sharp* N$
and $A_P \sharp* P'$
and $A_P \sharp* Q1$
and $A_P \sharp* Q'$
and $A_P \sharp* A_Q$
and $A_P \sharp* bn\ \alpha$
and $A_Q \sharp* \Psi$
and $A_Q \sharp* \Psi_P$
and $A_Q \sharp* P1$
and $A_Q \sharp* N$
and $A_Q \sharp* P'$
and $A_Q \sharp* Q1$
and $A_Q \sharp* Q'$
and $A_Q \sharp* bn\ \alpha$
and $bn\ \alpha \sharp* \Psi$
and $bn\ \alpha \sharp* \Psi_P$
and $bn\ \alpha \sharp* \Psi_Q$
and $bn\ \alpha \sharp* P1$
and $bn\ \alpha \sharp* Q1$
and $A_P \sharp* M$
and $A_Q \sharp* M$
and $bn\ \alpha \sharp* M$
and $A_P \sharp* C$
and $A_Q \sharp* C$

and *distinct* $(bn\ \alpha)$
by *auto*

moreover then have $P = P1$ **and** $Q = Q1$
by(*auto simp add*: *psi.inject*)

ultimately have *Prop* $(¡M(\!|\nu{*}(bn\ \alpha)|\!)\langle N\rangle)\ (P' \parallel Q')$
by(*force intro*: *rBrComm1*)
then show *?thesis* **using** $\langle \alpha \prec T = ¡M(\!|\nu{*}bn\ \alpha|\!)\langle N\rangle \prec P' \parallel Q'\rangle[symmetric]$
by(*force simp add*: *residualInject*)
next
case (*cBrComm2* Ψ_Q *P1 M N P'* A_P Ψ_P *Q1 Q'* A_Q)
note $\langle bn\ \alpha \sharp{*}\ \Psi\rangle$
moreover from $\langle bn\ \alpha \sharp{*}\ P\rangle$ $\langle bn\ \alpha \sharp{*}\ Q\rangle$ **have** $bn\ \alpha \sharp{*}\ (P \parallel Q)$ **by** *simp*
moreover from $\langle bn\ \alpha \sharp{*}\ subject\ \alpha\rangle$ **have** $bn\ \alpha \sharp{*}\ (\alpha \prec T)$ **by** *simp*
ultimately have *all*:
$P \parallel Q = P1 \parallel Q1 \wedge$
$\alpha \prec T = ¡M(\!|\nu{*}bn\ \alpha|\!)\langle N\rangle \prec P' \parallel Q' \wedge$
$\Psi \otimes \Psi_Q \rhd P1 \longmapsto ¡M(\!|\nu{*}bn\ \alpha|\!)\langle N\rangle \prec P' \wedge$
extractFrame P1 $= \langle A_P, \Psi_P\rangle \wedge$
distinct $A_P \wedge$
$\Psi \otimes \Psi_P \rhd Q1 \longmapsto ¿M(\!|N|\!) \prec Q' \wedge$
extractFrame Q1 $= \langle A_Q, \Psi_Q\rangle \wedge$
distinct $A_Q \wedge$
$A_P \sharp{*}\ \Psi \wedge$
$A_P \sharp{*}\ \Psi_Q \wedge$
$A_P \sharp{*}\ P1 \wedge$
$A_P \sharp{*}\ N \wedge$
$A_P \sharp{*}\ P' \wedge$
$A_P \sharp{*}\ Q1 \wedge$
$A_P \sharp{*}\ Q' \wedge$
$A_P \sharp{*}\ A_Q \wedge$
$A_P \sharp{*}\ bn\ \alpha \wedge$
$A_Q \sharp{*}\ \Psi \wedge$
$A_Q \sharp{*}\ \Psi_P \wedge$
$A_Q \sharp{*}\ P1 \wedge$
$A_Q \sharp{*}\ N \wedge$
$A_Q \sharp{*}\ P' \wedge$
$A_Q \sharp{*}\ Q1 \wedge$
$A_Q \sharp{*}\ Q' \wedge$
$A_Q \sharp{*}\ bn\ \alpha \wedge$
$bn\ \alpha \sharp{*}\ \Psi \wedge$
$bn\ \alpha \sharp{*}\ \Psi_P \wedge$
$bn\ \alpha \sharp{*}\ \Psi_Q \wedge$
$bn\ \alpha \sharp{*}\ P1 \wedge$
$bn\ \alpha \sharp{*}\ Q1 \wedge$
$A_P \sharp{*}\ M \wedge A_Q \sharp{*}\ M \wedge bn\ \alpha \sharp{*}\ M \wedge A_P \sharp{*}\ C \wedge A_Q \sharp{*}\ C \wedge$ *distinct* $(bn\ \alpha)$
by(*rule cBrComm2*(*1*))

from *all* **have** $bn\ \alpha \sharp* M$ **and** $\alpha \prec T = ¡M(\!|\nu{*}bn\ \alpha|\!)\langle N\rangle \prec P' \parallel Q'$
by *simp+*

from $\langle \alpha \prec T = ¡M(\!|\nu{*}bn\ \alpha|\!)\langle N\rangle \prec P' \parallel Q'\rangle$ **have** $\alpha \prec T = RBrOut\ M\ ((\!|\nu{*}bn\ \alpha|\!)N \prec' (P' \parallel Q'))$
by(*simp add*: *residualInject*)

then obtain *xvec* N' **where** $\alpha = ¡M(\!|\nu{*}xvec|\!)\langle N'\rangle$
by(*auto simp add*: *residualInject*)
then have $bn\ \alpha = xvec$ **by** *simp*
from $\langle \alpha = ¡M(\!|\nu{*}xvec|\!)\langle N'\rangle\rangle$ $\langle \alpha \prec T = ¡M(\!|\nu{*}bn\ \alpha|\!)\langle N\rangle \prec P' \parallel Q'\rangle$ **have** *resEq*: $¡M(\!|\nu{*}xvec|\!)\langle N'\rangle \prec T = ¡M(\!|\nu{*}bn\ \alpha|\!)\langle N\rangle \prec P' \parallel Q'$
by *simp*
then have $¡M(\!|\nu{*}bn\ \alpha|\!)\langle N'\rangle \prec T = ¡M(\!|\nu{*}bn\ \alpha|\!)\langle N\rangle \prec P' \parallel Q'$ **using** $\langle bn\ \alpha = xvec\rangle$
by *simp*
then have $¡M(\!|\nu{*}bn\ \alpha|\!)\langle N'\rangle = ¡M(\!|\nu{*}bn\ \alpha|\!)\langle N\rangle$ **using** $\langle bn\ \alpha \sharp* M\rangle$
by(*rule resResidEq*)
with $\langle \alpha = ¡M(\!|\nu{*}xvec|\!)\langle N'\rangle\rangle$ **have** $\alpha = ¡M(\!|\nu{*}xvec|\!)\langle N\rangle$ **by** *simp*
moreover from *all* **have** $P \parallel Q = P1 \parallel Q1$
and $\alpha \prec T = ¡M(\!|\nu{*}bn\ \alpha|\!)\langle N\rangle \prec P' \parallel Q'$
and $\Psi \otimes \Psi_Q \rhd P1 \longmapsto ¡M(\!|\nu{*}bn\ \alpha|\!)\langle N\rangle \prec P'$
and $extractFrame\ P1 = \langle A_P, \Psi_P\rangle$
and $distinct\ A_P$
and $\Psi \otimes \Psi_P \rhd Q1 \longmapsto ¿M(\!|N|\!) \prec Q'$
and $extractFrame\ Q1 = \langle A_Q, \Psi_Q\rangle$
and $distinct\ A_Q$
and $A_P \sharp* \Psi$
and $A_P \sharp* \Psi_Q$
and $A_P \sharp* P1$
and $A_P \sharp* N$
and $A_P \sharp* P'$
and $A_P \sharp* Q1$
and $A_P \sharp* Q'$
and $A_P \sharp* A_Q$
and $A_P \sharp* bn\ \alpha$
and $A_Q \sharp* \Psi$
and $A_Q \sharp* \Psi_P$
and $A_Q \sharp* P1$
and $A_Q \sharp* N$
and $A_Q \sharp* P'$
and $A_Q \sharp* Q1$
and $A_Q \sharp* Q'$
and $A_Q \sharp* bn\ \alpha$
and $bn\ \alpha \sharp* \Psi$
and $bn\ \alpha \sharp* \Psi_P$
and $bn\ \alpha \sharp* \Psi_Q$
and $bn\ \alpha \sharp* P1$
and $bn\ \alpha \sharp* Q1$

and $A_P \sharp* M$
and $A_Q \sharp* M$
and $bn\ \alpha \sharp* M$
and $A_P \sharp* C$
and $A_Q \sharp* C$
and *distinct* $(bn\ \alpha)$
by *auto*

moreover then have $P = P1$ **and** $Q = Q1$
by(*auto simp add*: *psi.inject*)

ultimately have *Prop* $(¡M(\!|\nu*(bn\ \alpha)|\!)\langle N\rangle)\ (P' \parallel Q')$
by(*force intro*: *rBrComm2*)
then show *?thesis* **using** $\langle\alpha \prec T = ¡M(\!|\nu*bn\ \alpha|\!)\langle N\rangle \prec P' \parallel Q'\rangle[symmetric]$
by(*force simp add*: *residualInject*)
next
case *cBrClose*
then show *?thesis* **using** $\langle x \sharp \Psi\rangle\ \langle x \sharp P\rangle\ \langle x \sharp Q\rangle\ \langle x \sharp \alpha\rangle\ \langle x \sharp T\rangle$
by(*simp add*: *residualInject*)
next
case *cOpen*
then show *?thesis* **using** *assms* $\langle x \sharp \Psi\rangle\ \langle x \sharp P\rangle\ \langle x \sharp Q\rangle\ \langle x \sharp \alpha\rangle\ \langle x \sharp T\rangle$
by(*simp add*: *residualInject*)
next
case *cBrOpen*
then show *?thesis* **using** *assms* $\langle x \sharp \Psi\rangle\ \langle x \sharp P\rangle\ \langle x \sharp Q\rangle\ \langle x \sharp \alpha\rangle\ \langle x \sharp T\rangle$
by(*simp add*: *residualInject*)
next
case *cScope*
then show *?thesis* **using** *assms* $\langle x \sharp \Psi\rangle\ \langle x \sharp P\rangle\ \langle x \sharp Q\rangle\ \langle x \sharp \alpha\rangle\ \langle x \sharp T\rangle$
by(*simp add*: *residualInject*)
next
case *cBang*
then show *?thesis*
by(*simp add*: *residualInject*)
qed
qed

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 $C :: 'f::fs\text{-}name$

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;
$A_Q \sharp* \Psi$; $A_Q \sharp* P$; $A_Q \sharp* Q$; $A_Q \sharp* M$; $A_Q \sharp* N$; $A_Q \sharp* C\rrbracket$
$\Longrightarrow$ *Prop* $(P' \parallel Q)$
and *rPar2*: $\bigwedge Q'\ A_P\ \Psi_P.\ \llbracket \Psi \otimes \Psi_P \rhd Q \longmapsto M(\!|N|\!) \prec Q'$; *extractFrame* $P = \langle A_P, \Psi_P\rangle$; *distinct* A_P;
$A_P \sharp* \Psi$; $A_P \sharp* P$; $A_P \sharp* Q$; $A_P \sharp* M$; $A_P \sharp* N$; $A_P \sharp* C\rrbracket$
$\Longrightarrow$ *Prop* $(P \parallel Q')$
shows *Prop* R
proof −
from *Trans* **obtain** α **where** $\Psi \rhd P \parallel Q \longmapsto \alpha \prec R$ **and** *bn* $\alpha \sharp* \Psi$ **and** *bn* $\alpha \sharp* P$ **and** *bn* $\alpha \sharp* Q$ **and** *bn* $\alpha \sharp*$ *subject* α **and** $\alpha = M(\!|N|\!)$ **by** *auto*
then show *?thesis* **using** *rPar1 rPar2*
by(*induct rule*: *parCases*) (*auto simp add*: *residualInject*)
qed

lemma *parBrInputCases*[*consumes 1*, *case-names cPar1 cPar2 cBrMerge*]:
fixes Ψ :: *'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 :: *'f::fs-name*

assumes *Trans*: $\Psi \rhd P \parallel Q \longmapsto ¿M(\!|N|\!) \prec R$
and *rPar1*: $\bigwedge P'\ A_Q\ \Psi_Q.\ \llbracket \Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$; *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$; *distinct* A_Q;
$A_Q \sharp* \Psi$; $A_Q \sharp* P$; $A_Q \sharp* Q$; $A_Q \sharp* M$; $A_Q \sharp* N$; $A_Q \sharp* C\rrbracket$
$\Longrightarrow$ *Prop* $(P' \parallel Q)$
and *rPar2*: $\bigwedge Q'\ A_P\ \Psi_P.\ \llbracket \Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$; *extractFrame* $P = \langle A_P, \Psi_P\rangle$; *distinct* A_P;
$A_P \sharp* \Psi$; $A_P \sharp* P$; $A_P \sharp* Q$; $A_P \sharp* M$; $A_P \sharp* N$; $A_P \sharp* C\rrbracket$
$\Longrightarrow$ *Prop* $(P \parallel Q')$
and *rBrMerge*: $\bigwedge \Psi_Q\ P'\ A_P\ \Psi_P\ Q'\ A_Q.$
$\llbracket \Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$; *extractFrame* $P = \langle A_P, \Psi_P\rangle$; *distinct* A_P;
$\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$; *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$; *distinct* A_Q;
$A_P \sharp* \Psi$; $A_P \sharp* \Psi_Q$; $A_P \sharp* P$; $A_P \sharp* N$; $A_P \sharp* P'$;
$A_P \sharp* Q$; $A_P \sharp* Q'$; $A_P \sharp* A_Q$; $A_P \sharp* M$; $A_Q \sharp* M$;
$A_Q \sharp* \Psi$; $A_Q \sharp* \Psi_P$; $A_Q \sharp* P$; $A_Q \sharp* N$; $A_Q \sharp* P'$;
$A_Q \sharp* Q$; $A_Q \sharp* Q'$; $A_P \sharp* C$; $A_Q \sharp* C\rrbracket \Longrightarrow$
Prop $(P' \parallel Q')$
shows *Prop* R
proof −
from *Trans* **obtain** α **where** $\Psi \rhd P \parallel Q \longmapsto \alpha \prec R$ **and** *bn* $\alpha \sharp* \Psi$ **and** *bn* $\alpha \sharp* P$ **and** *bn* $\alpha \sharp* Q$ **and** *bn* $\alpha \sharp*$ *subject* α **and** $\alpha = ¿M(\!|N|\!)$ **by** *auto*
then show *?thesis* **using** *rPar1 rPar2 rBrMerge*
by(*induct rule*: *parCases*) (*auto simp add*: *residualInject action.inject*)

qed

lemma *parOutputCases*[*consumes 5*, *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 R :: $('a, 'b, 'c)$ *psi*
and C :: $'f$::*fs-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 −
from *Trans* **have** *distinct xvec* **by**(*auto dest*: *boundOutputDistinct*)
obtain α **where** $\alpha = M(\!|\nu * xvec|\!)\langle N\rangle$ **by** *simp*
with *Trans* ‹*xvec* $\sharp*$ Ψ› ‹*xvec* $\sharp*$ P› ‹*xvec* $\sharp*$ Q› ‹*xvec* $\sharp*$ M›
have $\Psi \rhd P \parallel Q \longmapsto \alpha \prec R$ **and** *bn* $\alpha \sharp* \Psi$ **and** *bn* $\alpha \sharp* P$ **and** *bn* $\alpha \sharp* Q$ *bn* $\alpha \sharp*$ *subject* α
by *simp*+
then show *?thesis* **using** ‹$\alpha = M(\!|\nu * xvec|\!)\langle N\rangle$› *rPar1 rPar2* ‹*distinct xvec*›
by(*induct rule*: *parCases*[**where** $C = (xvec, C)$]) (*auto simp add*: *residualInject*)
qed

lemma *parBrOutputCases*[*consumes 5*, *case-names cPar1 cPar2 cBrComm1 cBrComm2*]:
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 R :: $('a, 'b, 'c)$ *psi*
and C :: $'f$::*fs-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*: $\bigwedge Q'\ A_P\ \Psi_P.\ [\![\Psi \otimes \Psi_P \rhd Q \longmapsto ¡M(\!|\nu* xvec|\!)\langle N\rangle \prec Q';\ extractFrame\ P = \langle A_P, \Psi_P\rangle;\ distinct\ A_P;$
$A_P \sharp* \Psi;\ A_P \sharp* P;\ A_P \sharp* Q;\ A_P \sharp* M;\ A_P \sharp* xvec;\ A_P \sharp* N;\ A_P \sharp* C;\ A_P \sharp* xvec;\ distinct\ xvec]\!] \Longrightarrow Prop\ (P \parallel Q')$
and *rBrComm1*: $\bigwedge \Psi_Q\ P'\ A_P\ \Psi_P\ Q'\ A_Q.$
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P';\ extractFrame\ P = \langle A_P, \Psi_P\rangle;\ distinct\ A_P;$
$\Psi \otimes \Psi_P \rhd Q \longmapsto ¡M(\!|\nu* xvec|\!)\langle N\rangle \prec Q';\ extractFrame\ Q = \langle A_Q, \Psi_Q\rangle;\ distinct\ A_Q;$
$distinct\ xvec;$
$A_P \sharp* \Psi;\ A_P \sharp* \Psi_Q;\ A_P \sharp* P;\ A_P \sharp* N;\ A_P \sharp* P';\ A_P \sharp* Q;\ A_P \sharp* xvec;\ A_P \sharp* Q';\ A_P \sharp* A_Q;\ A_P \sharp* C;$
$A_Q \sharp* \Psi;\ A_Q \sharp* \Psi_P;\ A_Q \sharp* P;\ A_Q \sharp* N;\ A_Q \sharp* P';\ A_Q \sharp* Q;\ A_Q \sharp* xvec;\ A_Q \sharp* Q';\ A_Q \sharp* C;$
$A_P \sharp* M;\ A_Q \sharp* M;\ xvec \sharp* M;$
$xvec \sharp* \Psi;\ xvec \sharp* \Psi_P;\ xvec \sharp* P;\ xvec \sharp* Q;\ xvec \sharp* \Psi_Q]\!] \Longrightarrow$
$Prop\ (P' \parallel Q')$
and *rBrComm2*: $\bigwedge \Psi_Q\ P'\ A_P\ \Psi_P\ Q'\ A_Q.$
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto ¡M(\!|\nu* xvec|\!)\langle N\rangle \prec P';\ extractFrame\ P = \langle A_P, \Psi_P\rangle;\ distinct\ A_P;$
$\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q';\ extractFrame\ Q = \langle A_Q, \Psi_Q\rangle;\ distinct\ A_Q;$
$distinct\ xvec;$
$A_P \sharp* \Psi;\ A_P \sharp* \Psi_Q;\ A_P \sharp* P;\ A_P \sharp* N;\ A_P \sharp* P';\ A_P \sharp* Q;\ A_P \sharp* xvec;\ A_P \sharp* Q';\ A_P \sharp* A_Q;\ A_P \sharp* C;$
$A_Q \sharp* \Psi;\ A_Q \sharp* \Psi_P;\ A_Q \sharp* P;\ A_Q \sharp* N;\ A_Q \sharp* P';\ A_Q \sharp* Q;\ A_Q \sharp* xvec;\ A_Q \sharp* Q';\ A_Q \sharp* C;$
$A_P \sharp* M;\ A_Q \sharp* M;\ xvec \sharp* M;$
$xvec \sharp* \Psi;\ xvec \sharp* \Psi_P;\ xvec \sharp* P;\ xvec \sharp* Q;\ xvec \sharp* \Psi_Q]\!] \Longrightarrow$
$Prop\ (P' \parallel Q')$
shows *Prop R*
proof −
from *Trans* **have** *distinct xvec* **by**(*auto dest*: *boundOutputDistinct*)
obtain α **where** α=¡$M(\!|\nu* xvec|\!)\langle N\rangle$ **by** *simp*
with *Trans* ‹$xvec \sharp* \Psi$› ‹$xvec \sharp* P$› ‹$xvec \sharp* Q$› ‹$xvec \sharp* M$›
have $\Psi \rhd P \parallel Q \longmapsto \alpha \prec R$ **and** $bn\ \alpha \sharp* \Psi$ **and** $bn\ \alpha \sharp* P$ **and** $bn\ \alpha \sharp* Q$ $bn\ \alpha \sharp* subject\ \alpha$
by *simp*+
then show *?thesis* **using** ‹α=¡$M(\!|\nu* xvec|\!)\langle N\rangle$› *rPar1 rPar2 rBrComm1 rBrComm2* ‹*distinct xvec*›
by(*induct rule*: *parCases*[**where** C=($xvec$, C)]) (*auto simp add*: *residualInject*

action.inject)
qed

lemma *theEqvt*[*eqvt-force*]:
fixes p :: *name prm*
and α :: $'a$ *action*

assumes $\alpha \neq \tau$

shows $(p \cdot the(subject\ \alpha)) = the(p \cdot (subject\ \alpha))$
using *assms*
by(*induct rule*: *actionCases*[**where** α=α]) *auto*

lemma *theSubjectFresh*[*simp*]:
fixes α :: $'a$ *action*
and x :: *name*

assumes $\alpha \neq \tau$

shows $x \sharp the(subject\ \alpha) = x \sharp subject\ \alpha$
using *assms*
by(*cases rule*: *actionCases*) *auto*

lemma *theSubjectFreshChain*[*simp*]:
fixes α :: $'a$ *action*
and *xvec* :: *name list*

assumes $\alpha \neq \tau$

shows $xvec \sharp* the(subject\ \alpha) = xvec \sharp* subject\ \alpha$
using *assms*
by(*cases rule*: *actionCases*) *auto*

lemma *inputObtainPrefix*:
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 N :: $'a$
and K :: $'a$
and B :: *name list*

assumes $\Psi \rhd P \longmapsto K(\!|N|\!) \prec P'$
and $extractFrame\ P = \langle A_P, \Psi_P \rangle$
and *distinct* A_P
and $B \sharp* P$
and $A_P \sharp* \Psi$
and $A_P \sharp* B$

and $A_P \sharp* P$
and $A_P \sharp* K$

obtains M **where** $\Psi \otimes \Psi_P \vdash K \leftrightarrow M$ **and** $B \sharp* M$
using *assms*
proof(*nominal-induct avoiding*: B *arbitrary*: *thesis rule*: *inputFrameInduct*)
case(*cAlpha* Ψ P K N P' A_P Ψ_P p B)
then obtain M **where** *subjEq*: $\Psi \otimes \Psi_P \vdash K \leftrightarrow M$ **and** $B \sharp* M$
by(*auto intro*: *cAlpha*)
from ‹$\Psi \otimes \Psi_P \vdash K \leftrightarrow M$›
have $p \cdot (\Psi \otimes \Psi_P \vdash K \leftrightarrow M)$ **by** *simp*
with ‹*set* $p \subseteq$ *set* $A_P \times$ *set* $(p \cdot A_P)$›
‹$A_P \sharp* \Psi$› ‹$(p \cdot A_P) \sharp* \Psi$› ‹$A_P \sharp* K$› ‹$(p \cdot A_P) \sharp* K$›
have *permEq*: $\Psi \otimes (p \cdot \Psi_P) \vdash K \leftrightarrow (p \cdot M)$ **by**(*simp add*: *eqvts*)
from ‹$B \sharp* M$› **have** $p \cdot (B \sharp* M)$ **by** *simp*
with ‹*set* $p \subseteq$ *set* $A_P \times$ *set* $(p \cdot A_P)$›
‹$A_P \sharp* B$› ‹$(p \cdot A_P) \sharp* B$›
have *permFresh*: $B \sharp* (p \cdot M)$ **by**(*simp add*: *eqvts*)

show *?case* **using** *cAlpha permEq permFresh*
by *auto*
next
case(*cInput* Ψ M K *xvec* N *Tvec* P B)
from ‹$\Psi \vdash M \leftrightarrow K$› **have** $\Psi \otimes \mathbf{1} \vdash M \leftrightarrow K$
by(*blast intro*: *statEqEnt AssertionStatEqSym*[*OF Identity*])
then have $\Psi \otimes \mathbf{1} \vdash K \leftrightarrow M$ **by**(*rule chanEqSym*)
moreover from ‹$B \sharp* (M(\!|\lambda* xvec\ N|\!).P)$› **have** $B \sharp* M$ **by** *simp*
ultimately show *?case* **by**(*auto intro*: *cInput*)
next
case(*cCase* Ψ P K N P' φ *Cs* A_P Ψ_P B)
then obtain M **where** $\Psi \otimes \Psi_P \vdash K \leftrightarrow M$ **and** $B \sharp* M$
by $-$ (*rule cCase*, *auto dest*: *memFreshChain*)
with ‹$\Psi_P \simeq \mathbf{1}$› **show** *?case* **by**(*blast intro*: *cCase statEqEnt compositionSym Identity*)
next
case(*cPar1* Ψ Ψ_Q P K N P' A_Q Q A_P Ψ_P B)
then obtain M **where** $(\Psi \otimes \Psi_Q) \otimes \Psi_P \vdash K \leftrightarrow M$ **and** $B \sharp* M$
by (*metis freshCompChain*(*1*) *psiFreshVec*(*4*))
then show *?case*
by(*metis cPar1 statEqEnt Associativity Commutativity AssertionStatEqTrans Composition*)
next
case(*cPar2* Ψ Ψ_P Q K N Q' A_P P A_Q Ψ_Q B)
then obtain M **where** $(\Psi \otimes \Psi_P) \otimes \Psi_Q \vdash K \leftrightarrow M$ **and** $B \sharp* M$
by $-$ (*rule cPar2*, *auto*)
then show *?case* **by**(*metis cPar2 statEqEnt Associativity*)
next
case(*cScope* Ψ P K N P' x A_P Ψ_P B)
then obtain M **where** $\Psi \otimes \Psi_P \vdash K \leftrightarrow M$ **and** $B \sharp* M$

```
    by − (rule cScope, auto)
  then show ?case by(auto intro: cScope)
next
  case(cBang Ψ P K N P' A_P Ψ_P B)
  then obtain M where Ψ ⊗ Ψ_P ⊗ 1 ⊢ K ↔ M and B ♯* M
    by − (rule cBang, auto)
  with ‹Ψ_P ≃ 1› show ?case by(metis cBang statEqEnt compositionSym Identity)
qed

lemma outputObtainPrefix:
  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 N  :: 'a
    and K  :: 'a
    and xvec :: name list
    and B  :: name list

assumes Ψ ▷ P ⟼ ROut K (⦇ν*xvec⦈)N ≺' P')
  and   extractFrame P = ⟨A_P, Ψ_P⟩
  and   distinct A_P
  and   xvec ♯* K
  and   distinct xvec
  and   B ♯* P
  and   A_P ♯* Ψ
  and   A_P ♯* B
  and   A_P ♯* P
  and   A_P ♯* K

obtains M where Ψ ⊗ Ψ_P ⊢ K ↔ M and B ♯* M
  using assms
proof(nominal-induct avoiding: B xvec arbitrary: thesis rule: outputFrameInduct)
  case(cAlpha Ψ P K A_P Ψ_P p α B xvec)
  then obtain M where subjEq: Ψ ⊗ Ψ_P ⊢ K ↔ M and B ♯* M
    by(auto intro: cAlpha)

  from ‹Ψ ⊗ Ψ_P ⊢ K ↔ M›
  have p • (Ψ ⊗ Ψ_P ⊢ K ↔ M) by simp

  with ‹set p ⊆ set A_P × set (p • A_P)›
    ‹A_P ♯* Ψ› ‹(p • A_P) ♯* Ψ› ‹A_P ♯* K› ‹(p • A_P) ♯* K›
  have permEq: Ψ ⊗ (p • Ψ_P) ⊢ K ↔ (p • M) by(simp add: eqvts)

  from ‹B ♯* M› have p • (B ♯* M) by simp
  with ‹set p ⊆ set A_P × set (p • A_P)›
    ‹A_P ♯* B› ‹(p • A_P) ♯* B›
  have permFresh: B ♯* (p • M) by(simp add: eqvts)
```

show *?case* **using** *cAlpha permEq permFresh*
by *auto*
next
case(*cOutput* Ψ M K N P B *xvec*)
from ‹$\Psi \vdash M \leftrightarrow K$› **have** $\Psi \otimes \mathbf{1} \vdash M \leftrightarrow K$
by(*blast intro*: *statEqEnt AssertionStatEqSym*[*OF Identity*])
then have $\Psi \otimes \mathbf{1} \vdash K \leftrightarrow M$
by(*rule chanEqSym*)
moreover from ‹$B \sharp* (M\langle N\rangle.P)$› **have** $B \sharp* M$ **by** *simp*
ultimately show *?case* **by**(*auto intro*: *cOutput*)
next
case(*cCase* Ψ P K P' φ *Cs* A_P Ψ_P B *xvec*)
then obtain M **where** $\Psi \otimes \Psi_P \vdash K \leftrightarrow M$ **and** $B \sharp* M$
by − (*rule cCase*, *auto dest*: *memFreshChain*)
with ‹$\Psi_P \simeq \mathbf{1}$› **show** *?case* **by**(*blast intro*: *cCase statEqEnt compositionSym Identity*)
next
case(*cPar1* Ψ Ψ_Q P K *yvec* N P' A_Q Q A_P Ψ_P B *xvec*)
then obtain M **where** $(\Psi \otimes \Psi_Q) \otimes \Psi_P \vdash K \leftrightarrow M$ **and** $B \sharp* M$
by (*metis freshCompChain*(*1*) *psiFreshVec*(*4*))
then show *?case* **by**(*metis cPar1 statEqEnt Associativity Commutativity AssertionStatEqTrans Composition*)
next
case(*cPar2* Ψ Ψ_P Q K *yvec* N Q' A_P P A_Q Ψ_Q B *xvec*)
then obtain M **where** $(\Psi \otimes \Psi_P) \otimes \Psi_Q \vdash K \leftrightarrow M$ **and** $B \sharp* M$
by (*metis freshCompChain*(*1*) *psiFreshVec*(*4*))
then show *?case* **by**(*metis cPar2 statEqEnt Associativity*)
next
case(*cOpen* Ψ P M *zvec yvec* N P' x A_P Ψ_P B *xvec*)
then obtain K **where** $\Psi \otimes \Psi_P \vdash M \leftrightarrow K$ **and** $B \sharp* K$
by (*metis abs-fresh-list-star'* *psiFreshVec*(*5*))
then show *?case* **by**(*auto intro*: *cOpen*)
next
case(*cScope* Ψ P K *yvec* N P' x A_P Ψ_P B *xvec*)
then obtain M **where** $\Psi \otimes \Psi_P \vdash K \leftrightarrow M$ **and** $B \sharp* M$
by (*metis abs-fresh-list-star'* *psiFreshVec*(*5*))
then show *?case* **by**(*auto intro*: *cScope*)
next
case(*cBang* Ψ P K P' A_P Ψ_P B *xvec*)
then obtain M **where** $\Psi \otimes \Psi_P \otimes \mathbf{1} \vdash K \leftrightarrow M$ **and** $B \sharp* M$
by − (*rule cBang*, *auto*)
with ‹$\Psi_P \simeq \mathbf{1}$› **show** *?case* **by**(*metis cBang statEqEnt compositionSym Identity*)
qed

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 $\Psi \rhd P \longmapsto K(\!|N|\!) \prec P'$
using *assms*
proof(*nominal-induct avoiding*: K *rule*: *inputFrameInduct*)
case(*cAlpha* Ψ P M N P' A_P Ψ_P p K)
have S: *set* $p \subseteq$ *set* $A_P \times$ *set* $(p \cdot A_P)$ **by** *fact*
from ‹$\Psi \otimes (p \cdot \Psi_P) \vdash M \leftrightarrow K$› **have** $(p \cdot (\Psi \otimes (p \cdot \Psi_P))) \vdash (p \cdot M) \leftrightarrow (p \cdot K)$
by(*rule chanEqClosed*)
with S ‹*distinctPerm* p› ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* M$› ‹$A_P \sharp* K$› ‹$(p \cdot A_P) \sharp* \Psi$› ‹$(p \cdot A_P) \sharp* M$› ‹$(p \cdot A_P) \sharp* K$›
have $\Psi \otimes \Psi_P \vdash M \leftrightarrow K$ **by**(*simp add*: *eqvts*)
with ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_P \sharp* M$› ‹$A_P \sharp* K$›
‹$[\![\Psi \otimes \Psi_P \vdash M \leftrightarrow K;\ A_P \sharp* \Psi;\ A_P \sharp* P;\ A_P \sharp* M;\ A_P \sharp* K]\!] \Longrightarrow \Psi \rhd P \longmapsto K(\!|N|\!) \prec P'$›
show *?case* **by** *blast*
next
case(*cInput* Ψ M K *xvec* N *Tvec* P K')
from ‹$\Psi \otimes \mathbf{1} \vdash K \leftrightarrow K'$› **have** $\Psi \vdash K \leftrightarrow K'$
by(*blast intro*: *statEqEnt Identity*)
with ‹$\Psi \vdash M \leftrightarrow K$› **have** $\Psi \vdash M \leftrightarrow K'$
by(*rule chanEqTrans*)
then show *?case* **using** ‹*distinct xvec*› ‹*set xvec* $\subseteq$ *supp* N› ‹*length xvec* = *length Tvec*›
by(*rule Input*)
next
case(*cCase* Ψ P M N P' φ *Cs* A_P Ψ_P K)
from ‹$\Psi \otimes \mathbf{1} \vdash M \leftrightarrow K$› ‹$\Psi_P \simeq \mathbf{1}$› **have** $\Psi \otimes \Psi_P \vdash M \leftrightarrow K$
by(*blast intro*: *statEqEnt Identity compositionSym AssertionStatEqSym*)
with ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_P \sharp* M$› ‹$A_P \sharp* K$›
‹$\bigwedge K.\ [\![\Psi \otimes \Psi_P \vdash M \leftrightarrow K;\ A_P \sharp* \Psi;\ A_P \sharp* P;\ A_P \sharp* M;\ A_P \sharp* K]\!] \Longrightarrow \Psi \rhd P \longmapsto K(\!|N|\!) \prec P'$›
have $\Psi \rhd P \longmapsto K(\!|N|\!) \prec P'$ **by** *force*
then show *?case* **using** ‹$(\varphi, P) \in$ *set Cs*› ‹$\Psi \vdash \varphi$› ‹*guarded* P› **by**(*rule Case*)
next
case(*cPar1* Ψ Ψ_Q P M N P' A_Q Q A_P Ψ_P K)

from ‹$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K$› **have** $(\Psi \otimes \Psi_Q) \otimes \Psi_P \vdash M \leftrightarrow K$
by(*metis statEqEnt Associativity Composition AssertionStatEqTrans Commutativity*)
with ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* \Psi_Q$› ‹$A_P \sharp* P$› ‹$A_P \sharp* M$› ‹$A_P \sharp* K$›
‹$\bigwedge K.\ [\![(\Psi \otimes \Psi_Q) \otimes \Psi_P \vdash M \leftrightarrow K;\ A_P \sharp* (\Psi \otimes \Psi_Q);\ A_P \sharp* P;\ A_P \sharp* M;\ A_P \sharp* K]\!] \Longrightarrow \Psi \otimes \Psi_Q \rhd P \longmapsto K(\!|N|\!) \prec P'$›
have $\Psi \otimes \Psi_Q \rhd P \longmapsto K(\!|N|\!) \prec P'$ **by** *force*
then show *?case* **using** ‹*extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$› ‹$A_Q \sharp* \Psi$› ‹$A_Q \sharp* P$› ‹$A_Q \sharp* K$› ‹$A_Q \sharp* N$›
by(*auto intro*: *Par1*)
next
case(*cPar2* Ψ Ψ_P Q M N Q' A_P P A_Q Ψ_Q K)
from ‹$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K$› **have** $(\Psi \otimes \Psi_P) \otimes \Psi_Q \vdash M \leftrightarrow K$
by(*rule statEqEnt*[*OF AssertionStatEqSym*[*OF Associativity*]])
with ‹$A_Q \sharp* \Psi$› ‹$A_Q \sharp* \Psi_P$› ‹$A_Q \sharp* Q$› ‹$A_Q \sharp* M$› ‹$A_Q \sharp* K$›
‹$\bigwedge K.\ [\![(\Psi \otimes \Psi_P) \otimes \Psi_Q \vdash M \leftrightarrow K;\ A_Q \sharp* (\Psi \otimes \Psi_P);\ A_Q \sharp* Q;\ A_Q \sharp* M;\ A_Q \sharp* K]\!] \Longrightarrow \Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|N|\!) \prec Q'$›
have $\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|N|\!) \prec Q'$ **by** *force*
then show *?case* **using** ‹*extractFrame* $P = \langle A_P, \Psi_P \rangle$› ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* Q$› ‹$A_P \sharp* K$› ‹$A_P \sharp* N$›
by(*auto intro*: *Par2*)
next
case(*cScope* Ψ P M N P' x A_P Ψ_P)
then have $\Psi \rhd P \longmapsto K(\!|N|\!) \prec P'$ **by** *force*
with ‹$x \sharp \Psi$› ‹$x \sharp K$› ‹$x \sharp N$› **show** *?case*
by(*auto intro*: *Scope*)
next
case(*cBang* Ψ P M N P' A_P Ψ_P K)
from ‹$\Psi \otimes \mathbf{1} \vdash M \leftrightarrow K$› ‹$\Psi_P \simeq \mathbf{1}$› **have** $\Psi \otimes \Psi_P \otimes \mathbf{1} \vdash M \leftrightarrow K$
by(*blast intro*: *statEqEnt Identity compositionSym AssertionStatEqSym*)
with ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_P \sharp* M$› ‹$A_P \sharp* K$›
‹$\bigwedge K.\ [\![\Psi \otimes \Psi_P \otimes \mathbf{1} \vdash M \leftrightarrow K;\ A_P \sharp* \Psi;\ A_P \sharp* (P \parallel !P);\ A_P \sharp* M;\ A_P \sharp* K]\!] \Longrightarrow \Psi \rhd P \parallel !P \longmapsto K(\!|N|\!) \prec P'$›
have $\Psi \rhd P \parallel !P \longmapsto K(\!|N|\!) \prec P'$ **by** *force*
then show *?case* **using** ‹*guarded* P› **by**(*rule Bang*)
qed

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 $\Psi \rhd P \longmapsto K(\!|\nu* xvec|\!)\langle N\rangle \prec P'$
using *assms* **unfolding** *residualInject*
proof(*nominal-induct avoiding*: *K rule*: *outputFrameInduct*)
case(*cAlpha* $\Psi\ P\ M\ A_P\ \Psi_P\ p\ B\ K$)
have *S*: $set\ p \subseteq set\ A_P \times set(p \cdot A_P)$ **by** *fact*
from ‹$\Psi \otimes (p \cdot \Psi_P) \vdash M \leftrightarrow K$› **have** $(p \cdot (\Psi \otimes (p \cdot \Psi_P))) \vdash (p \cdot M) \leftrightarrow (p \cdot K)$
by(*rule chanEqClosed*)
with *S* ‹*distinctPerm p*› ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* M$› ‹$A_P \sharp* K$› ‹$(p \cdot A_P) \sharp* \Psi$› ‹$(p \cdot A_P) \sharp* M$› ‹$(p \cdot A_P) \sharp* K$›
have $\Psi \otimes \Psi_P \vdash M \leftrightarrow K$ **by**(*simp add*: *eqvts*)
with ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_P \sharp* M$› ‹$A_P \sharp* K$›
show *?case* **by**(*blast intro*: *cAlpha*)
next
case(*cOutput* $\Psi\ M\ K\ N\ P\ K'$)
from ‹$\Psi \otimes \mathbf{1} \vdash K \leftrightarrow K'$› **have** $\Psi \vdash K \leftrightarrow K'$
by(*blast intro*: *statEqEnt Identity*)
with ‹$\Psi \vdash M \leftrightarrow K$› **have** $\Psi \vdash M \leftrightarrow K'$
by(*rule chanEqTrans*)
then show *?case* **using** *Output* **by**(*force simp add*: *residualInject*)
next
case(*cCase* $\Psi\ P\ M\ B\ \varphi\ Cs\ A_P\ \Psi_P\ K$)
from ‹$\Psi \otimes \mathbf{1} \vdash M \leftrightarrow K$› ‹$\Psi_P \simeq \mathbf{1}$› **have** $\Psi \otimes \Psi_P \vdash M \leftrightarrow K$
by(*blast intro*: *statEqEnt Identity compositionSym AssertionStatEqSym*)
with ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_P \sharp* M$› ‹$A_P \sharp* K$›
‹$\bigwedge K.\ [\![\Psi \otimes \Psi_P \vdash M \leftrightarrow K;\ A_P \sharp* \Psi;\ A_P \sharp* P;\ A_P \sharp* M;\ A_P \sharp* K]\!] \Longrightarrow \Psi \rhd P \longmapsto (ROut\ K\ B)$›
have $\Psi \rhd P \longmapsto ROut\ K\ B$ **by** *force*
then show *?case* **using** ‹$(\varphi, P) \in set\ Cs$› ‹$\Psi \vdash \varphi$› ‹*guarded P*› **by**(*rule Case*)
next
case(*cPar1* $\Psi\ \Psi_Q\ P\ M\ xvec\ N\ P'\ A_Q\ Q\ A_P\ \Psi_P\ K$)
from ‹$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K$› **have** $(\Psi \otimes \Psi_Q) \otimes \Psi_P \vdash M \leftrightarrow K$
by(*metis statEqEnt Associativity Composition AssertionStatEqTrans Commutativity*)
with ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* \Psi_Q$› ‹$A_P \sharp* P$› ‹$A_P \sharp* M$› ‹$A_P \sharp* K$›
‹$\bigwedge K.\ [\![(\Psi \otimes \Psi_Q) \otimes \Psi_P \vdash M \leftrightarrow K;\ A_P \sharp* (\Psi \otimes \Psi_Q);\ A_P \sharp* P;\ A_P \sharp* M;\ A_P \sharp* K]\!] \Longrightarrow \Psi \otimes \Psi_Q \rhd P \longmapsto (ROut\ K\ ((\!|\nu* xvec|\!)N \prec' P'))$›
have $\Psi \otimes \Psi_Q \rhd P \longmapsto K(\!|\nu* xvec|\!)\langle N\rangle \prec P'$ **by**(*force simp add*: *residualInject*)
then show *?case* **using** ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› ‹$xvec \sharp* Q$› ‹$A_Q \sharp* \Psi$› ‹$A_Q \sharp* P$› ‹$A_Q \sharp* K$› ‹$A_Q \sharp* xvec$› ‹$A_Q \sharp* N$› *Par1*[**where** $\alpha{=}K(\!|\nu* xvec|\!)\langle N\rangle$]
by(*auto simp add*: *residualInject*)
next

case(*cPar2* Ψ Ψ_P Q M *xvec* N Q' A_P P A_Q Ψ_Q K)
from ‹$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K$› **have** $(\Psi \otimes \Psi_P) \otimes \Psi_Q \vdash M \leftrightarrow K$
by(*rule statEqEnt*[*OF AssertionStatEqSym*[*OF Associativity*]])
with ‹$A_Q \sharp* \Psi$› ‹$A_Q \sharp* \Psi_P$› ‹$A_Q \sharp* Q$› ‹$A_Q \sharp* M$› ‹$A_Q \sharp* K$›
‹$\bigwedge K.\ [\![(\Psi \otimes \Psi_P) \otimes \Psi_Q \vdash M \leftrightarrow K;\ A_Q \sharp* (\Psi \otimes \Psi_P);\ A_Q \sharp* Q;\ A_Q \sharp* M;\ A_Q \sharp* K]\!] \Longrightarrow \Psi \otimes \Psi_P \rhd Q \longmapsto ROut\ K\ (\langle\!|\nu * xvec|\!\rangle N \prec' Q')$›
have $\Psi \otimes \Psi_P \rhd Q \longmapsto ROut\ K\ (\langle\!|\nu * xvec|\!\rangle N \prec' Q')$ **by** *force*
then show *?case* **using** ‹*extractFrame* $P = \langle A_P, \Psi_P\rangle$› ‹*xvec* $\sharp* P$› ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* Q$› ‹$A_P \sharp* K$› ‹$A_P \sharp* xvec$› ‹$A_P \sharp* N$› *Par2*[**where** $\alpha = K\langle\!|\nu * xvec|\!\rangle\langle N\rangle$]
by(*auto simp add*: *residualInject*)
next
case(*cOpen* Ψ P M *xvec* *yvec* N P' x A_P Ψ_P)
then have $\Psi \rhd P \longmapsto K\langle\!|\nu * (xvec @ yvec)|\!\rangle\langle N\rangle \prec P'$ **by**(*force simp add*: *residualInject*)
with ‹$x \in supp\ N$› ‹$x \sharp \Psi$› ‹$x \sharp K$› ‹$x \sharp xvec$› ‹$x \sharp yvec$› *Open* **show** *?case*
by(*auto simp add*: *residualInject*)
next
case(*cScope* Ψ P M *xvec* N P' x A_P Ψ_P)
then have $\Psi \rhd P \longmapsto K\langle\!|\nu * xvec|\!\rangle\langle N\rangle \prec P'$ **by**(*force simp add*: *residualInject*)
with ‹$x \sharp \Psi$› ‹$x \sharp K$› ‹$x \sharp xvec$› ‹$x \sharp N$› *Scope*[**where** $\alpha = K\langle\!|\nu * xvec|\!\rangle\langle N\rangle$] **show** *?case*
by(*auto simp add*: *residualInject*)
next
case(*cBang* Ψ P M B A_P Ψ_P K)
from ‹$\Psi \otimes \mathbf{1} \vdash M \leftrightarrow K$› ‹$\Psi_P \simeq \mathbf{1}$› **have** $\Psi \otimes \Psi_P \otimes \mathbf{1} \vdash M \leftrightarrow K$
by(*blast intro*: *statEqEnt Identity compositionSym AssertionStatEqSym*)
with ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_P \sharp* M$› ‹$A_P \sharp* K$›
‹$\bigwedge K.\ [\![\Psi \otimes \Psi_P \otimes \mathbf{1} \vdash M \leftrightarrow K;\ A_P \sharp* \Psi;\ A_P \sharp* (P \parallel\ !P);\ A_P \sharp* M;\ A_P \sharp* K]\!] \Longrightarrow \Psi \rhd P \parallel\ !P \longmapsto ROut\ K\ B$›
have $\Psi \rhd P \parallel\ !P \longmapsto ROut\ K\ B$ **by** *force*
then show *?case* **using** ‹*guarded* P› **by**(*rule Bang*)
qed

lemma *parCasesSubject*[*consumes* 7, *case-names cPar1 cPar2 cComm1 cComm2 cBrMerge cBrComm1 cBrComm2*]:
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 :: *'f*::*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* $\alpha \sharp* subject\ \alpha$
and *yvec* $\sharp* P$

and $yvec \sharp* Q$

and *rPar1*: $\bigwedge P'\ A_Q\ \Psi_Q.\ [\![\Psi \otimes \Psi_Q \rhd P \longmapsto \alpha \prec P';\ extractFrame\ Q = \langle A_Q, \Psi_Q\rangle;\ distinct\ A_Q;$
$A_Q \sharp* \Psi;\ A_Q \sharp* P;\ A_Q \sharp* \alpha;\ A_Q \sharp* C]\!] \Longrightarrow Prop\ \alpha\ (P' \parallel Q)$

and *rPar2*: $\bigwedge Q'\ A_P\ \Psi_P.\ [\![\Psi \otimes \Psi_P \rhd Q \longmapsto \alpha \prec Q';\ extractFrame\ P = \langle A_P, \Psi_P\rangle;\ distinct\ A_P;$
$A_P \sharp* \Psi;\ A_P \sharp* Q;\ A_P \sharp* \alpha;\ A_P \sharp* C]\!] \Longrightarrow Prop\ \alpha\ (P \parallel Q')$

and *rComm1*: $\bigwedge \Psi_Q\ M\ N\ P'\ A_P\ \Psi_P\ K\ xvec\ Q'\ A_Q.$
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|N|\!) \prec P';\ extractFrame\ P = \langle A_P, \Psi_P\rangle;\ distinct\ A_P;$
$\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|\nu*xvec|\!)\langle N\rangle \prec Q';\ extractFrame\ Q = \langle A_Q, \Psi_Q\rangle;$
$distinct\ A_Q;$
$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K;\ yvec \sharp* M;\ yvec \sharp* K;\ distinct\ xvec;\ \alpha = \tau;$
$A_P \sharp* \Psi;\ A_P \sharp* \Psi_Q;\ A_P \sharp* P;\ A_P \sharp* M;\ A_P \sharp* N;\ A_P \sharp* P';\ A_P \sharp* Q;\ A_P \sharp* xvec;\ A_P \sharp* Q';\ A_P \sharp* A_Q;\ A_P \sharp* C;$
$A_Q \sharp* \Psi;\ A_Q \sharp* \Psi_P;\ A_Q \sharp* P;\ A_Q \sharp* K;\ A_Q \sharp* N;\ A_Q \sharp* P';\ A_Q \sharp* Q;\ A_Q \sharp* xvec;\ A_Q \sharp* Q';\ A_Q \sharp* C;$
$xvec \sharp* \Psi;\ xvec \sharp* \Psi_P;\ xvec \sharp* P;\ xvec \sharp* M;\ xvec \sharp* K;\ xvec \sharp* Q;\ xvec \sharp* \Psi_Q;\ xvec \sharp* C]\!] \Longrightarrow$
$Prop\ (\tau)\ ((\!|\nu*xvec|\!)(P' \parallel Q'))$

and *rComm2*: $\bigwedge \Psi_Q\ M\ xvec\ N\ P'\ A_P\ \Psi_P\ K\ Q'\ A_Q.$
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|\nu*xvec|\!)\langle N\rangle \prec P';\ extractFrame\ P = \langle A_P, \Psi_P\rangle;\ distinct\ A_P;$
$\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|N|\!) \prec Q';\ extractFrame\ Q = \langle A_Q, \Psi_Q\rangle;\ distinct\ A_Q;$
$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K;\ yvec \sharp* M;\ yvec \sharp* K;\ distinct\ xvec;\ \alpha = \tau;$
$A_P \sharp* \Psi;\ A_P \sharp* \Psi_Q;\ A_P \sharp* P;\ A_P \sharp* M;\ A_P \sharp* N;\ A_P \sharp* P';\ A_P \sharp* Q;\ A_P \sharp* xvec;\ A_P \sharp* Q';\ A_P \sharp* A_Q;\ A_P \sharp* C;$
$A_Q \sharp* \Psi;\ A_Q \sharp* \Psi_P;\ A_Q \sharp* P;\ A_Q \sharp* K;\ A_Q \sharp* N;\ A_Q \sharp* P';\ A_Q \sharp* Q;\ A_Q \sharp* xvec;\ A_Q \sharp* Q';\ A_Q \sharp* C;$
$xvec \sharp* \Psi;\ xvec \sharp* \Psi_P;\ xvec \sharp* P;\ xvec \sharp* M;\ xvec \sharp* K;\ xvec \sharp* Q;\ xvec \sharp* \Psi_Q;\ xvec \sharp* C]\!] \Longrightarrow$
$Prop\ (\tau)\ ((\!|\nu*xvec|\!)(P' \parallel Q'))$

and *rBrMerge*: $\bigwedge \Psi_Q\ M\ N\ P'\ A_P\ \Psi_P\ Q'\ A_Q.$
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P';\ extractFrame\ P = \langle A_P, \Psi_P\rangle;\ distinct\ A_P;$
$\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q';\ extractFrame\ Q = \langle A_Q, \Psi_Q\rangle;\ distinct\ A_Q;$
$A_P \sharp* \Psi;\ A_P \sharp* \Psi_Q;\ A_P \sharp* P;\ A_P \sharp* N;\ A_P \sharp* P';$
$A_P \sharp* Q;\ A_P \sharp* Q';\ A_P \sharp* A_Q;\ A_P \sharp* M;\ A_Q \sharp* M;$
$A_Q \sharp* \Psi;\ A_Q \sharp* \Psi_P;\ A_Q \sharp* P;\ A_Q \sharp* N;\ A_Q \sharp* P';$
$A_Q \sharp* Q;\ A_Q \sharp* Q';\ A_P \sharp* C;\ A_Q \sharp* C;\ \alpha = ¿M(\!|N|\!)]\!] \Longrightarrow$
$Prop\ (¿M(\!|N|\!))\ (P' \parallel Q')$

and *rBrComm1*: $\bigwedge \Psi_Q\ M\ N\ P'\ A_P\ \Psi_P\ xvec\ Q'\ A_Q.$
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P';\ extractFrame\ P = \langle A_P, \Psi_P\rangle;\ distinct\ A_P;$
$\Psi \otimes \Psi_P \rhd Q \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec Q';\ extractFrame\ Q = \langle A_Q, \Psi_Q\rangle;\ distinct\ A_Q;$
$distinct\ xvec;$
$A_P \sharp* \Psi;\ A_P \sharp* \Psi_Q;\ A_P \sharp* P;\ A_P \sharp* N;\ A_P \sharp* P';\ A_P \sharp* Q;\ A_P \sharp* xvec;\ A_P \sharp* Q';\ A_P \sharp* A_Q;\ A_P \sharp* C;$

$A_Q \sharp* \Psi$; $A_Q \sharp* \Psi_P$; $A_Q \sharp* P$; $A_Q \sharp* N$; $A_Q \sharp* P'$; $A_Q \sharp* Q$; $A_Q \sharp* xvec$; $A_Q \sharp* Q'$; $A_Q \sharp* C$;
$A_P \sharp* M$; $A_Q \sharp* M$; $xvec \sharp* M$; ¡$M(\!|\nu*xvec|\!)\langle N\rangle = \alpha$;
$xvec \sharp* \Psi$; $xvec \sharp* \Psi_P$; $xvec \sharp* P$; $xvec \sharp* Q$; $xvec \sharp* \Psi_Q$$]\!] \Longrightarrow$
Prop (¡$M(\!|\nu*xvec|\!)\langle N\rangle$) ($P' \parallel Q'$)

and *rBrComm2*: $\bigwedge \Psi_Q\ M\ xvec\ N\ P'\ A_P\ \Psi_P\ Q'\ A_Q$.
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto$¡$M(\!|\nu*xvec|\!)\langle N\rangle \prec P'$; *extractFrame* $P = \langle A_P, \Psi_P\rangle$; *distinct* A_P;
$\Psi \otimes \Psi_P \rhd Q \longmapsto$¿$M(\!|N|\!) \prec Q'$; *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$; *distinct* A_Q;
distinct xvec;
$A_P \sharp* \Psi$; $A_P \sharp* \Psi_Q$; $A_P \sharp* P$; $A_P \sharp* N$; $A_P \sharp* P'$; $A_P \sharp* Q$; $A_P \sharp* xvec$; $A_P \sharp* Q'$; $A_P \sharp* A_Q$; $A_P \sharp* C$;
$A_Q \sharp* \Psi$; $A_Q \sharp* \Psi_P$; $A_Q \sharp* P$; $A_Q \sharp* N$; $A_Q \sharp* P'$; $A_Q \sharp* Q$; $A_Q \sharp* xvec$; $A_Q \sharp* Q'$; $A_Q \sharp* C$;
$A_P \sharp* M$; $A_Q \sharp* M$; $xvec \sharp* M$; ¡$M(\!|\nu*xvec|\!)\langle N\rangle = \alpha$;
$xvec \sharp* \Psi$; $xvec \sharp* \Psi_P$; $xvec \sharp* P$; $xvec \sharp* Q$; $xvec \sharp* \Psi_Q$$]\!] \Longrightarrow$
Prop (¡$M(\!|\nu*xvec|\!)\langle N\rangle$) ($P' \parallel Q'$)

shows *Prop* α R
using *Trans* ‹*bn* $\alpha \sharp* \Psi$› ‹*bn* $\alpha \sharp* P$› ‹*bn* $\alpha \sharp* Q$› ‹*bn* $\alpha \sharp*$ *subject* α›
proof(*induct rule*: *parCases*[**where** C=(C, *yvec*)])
case(*cPar1* $P'\ A_Q\ \Psi_Q$)
then show *?case* **by**(*auto intro*: *rPar1*)
next
case(*cPar2* $Q'\ A_P\ \Psi_P$)
then show *?case* **by**(*auto intro*: *rPar2*)
next
case(*cComm1* $\Psi_Q\ M\ N\ P'\ A_P\ \Psi_P\ K\ xvec\ Q'\ A_Q$)
from ‹$A_P \sharp* (C,$ *yvec*)› ‹$A_Q \sharp* (C,$ *yvec*)› ‹*xvec* $\sharp* (C,$ *yvec*)›
have $A_P \sharp* C$ **and** $A_Q \sharp* C$ **and** $xvec \sharp* C$ **and** $A_P \sharp*$ *yvec* **and** $A_Q \sharp*$ *yvec* **and** $xvec \sharp*$ *yvec*
by *simp*+

have *FrP*: *extractFrame* $P = \langle A_P, \Psi_P\rangle$ **and** *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$
and *MeqK*: $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K$ **by** *fact*+

from ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|N|\!) \prec P'$› *FrP* ‹*distinct* A_P› ‹$A_P \sharp* P$› ‹$A_Q \sharp* P$› ‹*yvec* $\sharp* P$› ‹$A_P \sharp* \Psi$›
‹$A_P \sharp* A_Q$› ‹$A_P \sharp*$ *yvec*› ‹$A_P \sharp*$ *xvec*› ‹$A_P \sharp* P$› ‹$A_P \sharp* M$› ‹*xvec* $\sharp* P$› ‹$A_P \sharp* \Psi_Q$›
obtain M' **where** *MeqM'*: $(\Psi \otimes \Psi_Q) \otimes \Psi_P \vdash M \leftrightarrow M'$ **and** $xvec \sharp* M'$ **and** *yvec* $\sharp* M'$ **and** $A_Q \sharp* M'$
by − (*rule inputObtainPrefix*[**where** B=*xvec*@*yvec*@A_Q], (*assumption* | *force*)+)
from ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|\nu*xvec|\!)\langle N\rangle \prec Q'$› **have** $\Psi \otimes \Psi_P \rhd Q \longmapsto$ *ROut* K $(\!|\nu*xvec|\!)N \prec' Q'$)
by(*simp add*: *residualInject*)
with *FrQ* ‹*distinct* A_Q› ‹$A_P \sharp* Q$› ‹$A_Q \sharp* Q$› ‹*yvec* $\sharp* Q$› ‹$A_Q \sharp* \Psi$›
‹$A_P \sharp* A_Q$› ‹$A_Q \sharp*$ *yvec*› ‹$A_Q \sharp*$ *xvec*› ‹$A_Q \sharp* Q$› ‹$A_Q \sharp* K$› ‹*xvec* $\sharp* Q$› ‹A_Q

$\sharp*\ \Psi_P\rangle$ $\langle xvec\ \sharp*\ K\rangle$ $\langle distinct\ xvec\rangle$
obtain K' **where** $KeqK'$: $(\Psi \otimes \Psi_P) \otimes \Psi_Q \vdash K \leftrightarrow K'$ **and** $xvec\ \sharp*\ K'$ **and** $yvec\ \sharp*\ K'$ **and** $A_P\ \sharp*\ K'$
by − (*rule outputObtainPrefix*[**where** B=$xvec@yvec@A_P$], (*assumption* | *force* | *metis freshChainSym*)+)

from *MeqK KeqK'* **have** $(\Psi \otimes \Psi_Q) \otimes \Psi_P \vdash M \leftrightarrow K'$
by(*metis statEqEnt Associativity Commutativity Composition chanEqTrans*)
with $\langle\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|N|\!) \prec P'\rangle$ *FrP* $\langle distinct\ A_P\rangle$
have $\Psi \otimes \Psi_Q \rhd P \longmapsto K'(\!|N|\!) \prec P'$ **using** $\langle A_P\ \sharp*\ \Psi\rangle$ $\langle A_P\ \sharp*\ \Psi_Q\rangle$ $\langle A_P\ \sharp*\ P\rangle$ $\langle A_P\ \sharp*\ M\rangle$ $\langle A_P\ \sharp*\ K'\rangle$
by − (*rule inputRenameSubject*, (*assumption* | *force*)+)
moreover note *FrP* $\langle distinct\ A_P\rangle$
moreover from *MeqK MeqM'* **have** $(\Psi \otimes \Psi_P) \otimes \Psi_Q \vdash K \leftrightarrow M'$
by(*metis statEqEnt Associativity Commutativity Composition chanEqTrans chanEqSym*)
with $\langle\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|\nu*xvec|\!)\langle N\rangle \prec Q'\rangle$ *FrQ* $\langle distinct\ A_Q\rangle$
have $\Psi \otimes \Psi_P \rhd Q \longmapsto M'(\!|\nu*xvec|\!)\langle N\rangle \prec Q'$ **using** $\langle A_Q\ \sharp*\ \Psi\rangle$ $\langle A_Q\ \sharp*\ \Psi_P\rangle$ $\langle A_Q\ \sharp*\ Q\rangle$ $\langle A_Q\ \sharp*\ K\rangle$ $\langle A_Q\ \sharp*\ M'\rangle$
by − (*rule outputRenameSubject*, (*assumption* | *force*)+)
moreover note *FrQ* $\langle distinct\ A_Q\rangle$
moreover from *MeqM' KeqK' MeqK* **have** $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash K' \leftrightarrow M'$
by(*metis statEqEnt Associativity Commutativity Composition chanEqTrans chanEqSym*)
moreover note $\langle A_P\ \sharp*\ \Psi\rangle$ $\langle A_P\ \sharp*\ \Psi_Q\rangle$ $\langle A_P\ \sharp*\ P\rangle$ $\langle A_P\ \sharp*\ K'\rangle$ $\langle A_P\ \sharp*\ N\rangle$ $\langle A_P\ \sharp*\ P'\rangle$ $\langle A_P\ \sharp*\ Q\rangle$ $\langle A_P\ \sharp*\ xvec\rangle$ $\langle A_P\ \sharp*\ Q'\rangle$ $\langle A_P\ \sharp*\ A_Q\rangle$ $\langle A_P\ \sharp*\ C\rangle$
$\langle A_Q\ \sharp*\ \Psi\rangle$ $\langle A_Q\ \sharp*\ \Psi_P\rangle$ $\langle A_Q\ \sharp*\ Q\rangle$ $\langle A_Q\ \sharp*\ M'\rangle$ $\langle A_Q\ \sharp*\ N\rangle$ $\langle A_Q\ \sharp*\ Q'\rangle$ $\langle A_Q\ \sharp*\ P\rangle$ $\langle A_Q\ \sharp*\ xvec\rangle$ $\langle A_Q\ \sharp*\ P'\rangle$ $\langle A_Q\ \sharp*\ C\rangle$ $\langle\alpha = \tau\rangle$
$\langle xvec\ \sharp*\ \Psi\rangle$ $\langle xvec\ \sharp*\ \Psi_P\rangle$ $\langle xvec\ \sharp*\ P\rangle$ $\langle xvec\ \sharp*\ M'\rangle$ $\langle xvec\ \sharp*\ K'\rangle$ $\langle xvec\ \sharp*\ Q\rangle$ $\langle xvec\ \sharp*\ \Psi_Q\rangle$ $\langle xvec\ \sharp*\ C\rangle$ $\langle yvec\ \sharp*\ M'\rangle$ $\langle yvec\ \sharp*\ K'\rangle$ $\langle distinct\ xvec\rangle$
ultimately show *?case*
by(*metis rComm1*)
next
case(*cComm2* Ψ_Q M *xvec* N P' A_P Ψ_P K Q' A_Q)
from $\langle A_P\ \sharp*\ (C,\ yvec)\rangle$ $\langle A_Q\ \sharp*\ (C,\ yvec)\rangle$ $\langle xvec\ \sharp*\ (C,\ yvec)\rangle$
have $A_P\ \sharp*\ C$ **and** $A_Q\ \sharp*\ C$ **and** $xvec\ \sharp*\ C$ **and** $A_P\ \sharp*\ yvec$ **and** $A_Q\ \sharp*\ yvec$ **and** $xvec\ \sharp*\ yvec$
by *simp*+

have *FrP*: $extractFrame\ P = \langle A_P,\ \Psi_P\rangle$ **and** *FrQ*: $extractFrame\ Q = \langle A_Q,\ \Psi_Q\rangle$
and *MeqK*: $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K$ **by** *fact*+

from $\langle\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|\nu*xvec|\!)\langle N\rangle \prec P'\rangle$ **have** $\Psi \otimes \Psi_Q \rhd P \longmapsto ROut\ M\ (\!|\nu*xvec|\!)N \prec' P'$)
by(*simp add*: *residualInject*)
with *FrP* $\langle distinct\ A_P\rangle$ $\langle A_P\ \sharp*\ P\rangle$ $\langle A_Q\ \sharp*\ P\rangle$ $\langle yvec\ \sharp*\ P\rangle$ $\langle A_P\ \sharp*\ \Psi\rangle$
$\langle A_P\ \sharp*\ A_Q\rangle$ $\langle A_P\ \sharp*\ yvec\rangle$ $\langle A_P\ \sharp*\ xvec\rangle$ $\langle A_P\ \sharp*\ P\rangle$ $\langle A_P\ \sharp*\ M\rangle$ $\langle xvec\ \sharp*\ P\rangle$ $\langle A_P\ \sharp*\ \Psi_Q\rangle$ $\langle xvec\ \sharp*\ M\rangle$ $\langle distinct\ xvec\rangle$
obtain M' **where** $MeqM'$: $(\Psi \otimes \Psi_Q) \otimes \Psi_P \vdash M \leftrightarrow M'$ **and** $xvec\ \sharp*\ M'$ **and**

yvec $\sharp*$ M' **and** A_Q $\sharp*$ M'
by − (*rule outputObtainPrefix*[**where** *B*=*xvec*@*yvec*@A_Q], (*assumption* | *force*)+)
from ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|N|\!) \prec Q'$› *FrQ* ‹*distinct* A_Q› ‹A_P $\sharp*$ Q› ‹A_Q $\sharp*$ Q› ‹*yvec* $\sharp*$ Q› ‹A_Q $\sharp*$ Ψ›
‹A_P $\sharp*$ A_Q› ‹A_Q $\sharp*$ *yvec*› ‹A_Q $\sharp*$ *xvec*› ‹A_Q $\sharp*$ Q› ‹A_Q $\sharp*$ K› ‹*xvec* $\sharp*$ Q› ‹A_Q $\sharp*$ Ψ_P›
obtain K' **where** *KeqK'*: $(\Psi \otimes \Psi_P) \otimes \Psi_Q \vdash K \leftrightarrow K'$ **and** *xvec* $\sharp*$ K' **and** *yvec* $\sharp*$ K' **and** A_P $\sharp*$ K'
by − (*rule inputObtainPrefix*[**where** *B*=*xvec*@*yvec*@A_P], (*assumption* | *force* | *metis freshChainSym*)+)

from *MeqK KeqK'* **have** $(\Psi \otimes \Psi_Q) \otimes \Psi_P \vdash M \leftrightarrow K'$
by(*metis statEqEnt Associativity Commutativity Composition chanEqTrans*)
with ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|\nu*xvec|\!)\langle N\rangle \prec P'$› *FrP* ‹*distinct* A_P›
have $\Psi \otimes \Psi_Q \rhd P \longmapsto K'(\!|\nu*xvec|\!)\langle N\rangle \prec P'$ **using** ‹A_P $\sharp*$ Ψ› ‹A_P $\sharp*$ Ψ_Q› ‹A_P $\sharp*$ P› ‹A_P $\sharp*$ M› ‹A_P $\sharp*$ K'›
by − (*rule outputRenameSubject*, (*assumption* | *force*)+)
moreover note *FrP* ‹*distinct* A_P›
moreover from *MeqK MeqM'* **have** $(\Psi \otimes \Psi_P) \otimes \Psi_Q \vdash K \leftrightarrow M'$
by(*metis statEqEnt Associativity Commutativity Composition chanEqTrans chanEqSym*)
with ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|N|\!) \prec Q'$› *FrQ* ‹*distinct* A_Q›
have $\Psi \otimes \Psi_P \rhd Q \longmapsto M'(\!|N|\!) \prec Q'$ **using** ‹A_Q $\sharp*$ Ψ› ‹A_Q $\sharp*$ Ψ_P› ‹A_Q $\sharp*$ Q› ‹A_Q $\sharp*$ K› ‹A_Q $\sharp*$ M'›
by − (*rule inputRenameSubject*, (*assumption* | *force*)+)
moreover note *FrQ* ‹*distinct* A_Q›
moreover from *MeqM' KeqK' MeqK* **have** $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash K' \leftrightarrow M'$
by(*metis statEqEnt Associativity Commutativity Composition chanEqTrans chanEqSym*)
moreover note ‹A_P $\sharp*$ Ψ› ‹A_P $\sharp*$ Ψ_Q› ‹A_P $\sharp*$ P› ‹A_P $\sharp*$ K'› ‹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 $\sharp*$ Ψ› ‹A_Q $\sharp*$ Ψ_P› ‹A_Q $\sharp*$ Q› ‹A_Q $\sharp*$ M'› ‹A_Q $\sharp*$ N› ‹A_Q $\sharp*$ Q'› ‹A_Q $\sharp*$ P› ‹A_Q $\sharp*$ *xvec*› ‹A_Q $\sharp*$ P'› ‹A_Q $\sharp*$ C› ‹$\alpha = \tau$›
‹*xvec* $\sharp*$ Ψ› ‹*xvec* $\sharp*$ Ψ_P› ‹*xvec* $\sharp*$ P› ‹*xvec* $\sharp*$ M'› ‹*xvec* $\sharp*$ K'› ‹*xvec* $\sharp*$ Q› ‹*xvec* $\sharp*$ Ψ_Q› ‹*xvec* $\sharp*$ C› ‹*yvec* $\sharp*$ M'› ‹*yvec* $\sharp*$ K'› ‹*distinct xvec*›
ultimately show *?case*
by(*metis rComm2*)
next
case *cBrMerge*
then show *?case* **by** (*simp add*: *rBrMerge*)
next
case *cBrComm1*
then show *?case* **by** (*auto intro*: *rBrComm1*)
next
case *cBrComm2*
then show *?case* **by** (*auto intro*: *rBrComm2*)
qed

lemma *inputCases*[*consumes 1*, *case-names cInput cBrInput*]:

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*: $\bigwedge K\ Tvec.\ [\![\Psi \vdash M \leftrightarrow K; set\ xvec \subseteq supp\ N; length\ xvec = length\ Tvec; distinct\ xvec]\!] \Longrightarrow Prop\ (K(\!|N[xvec::=Tvec]|\!))\ (P[xvec::=Tvec])$
and *rBrInput*: $\bigwedge K\ Tvec.\ [\![\Psi \vdash K \succeq M; set\ xvec \subseteq supp\ N; length\ xvec = length\ Tvec; distinct\ xvec]\!] \Longrightarrow Prop\ (¿K(\!|N[xvec::=Tvec]|\!))\ (P[xvec::=Tvec])$

shows $Prop\ \alpha\ P'$
proof −
{
fix $xvec\ N\ P$
assume *Trans*: $\Psi \rhd M(\!|\lambda * xvec\ N|\!).P \longmapsto \alpha \prec P'$
and $xvec \sharp * \Psi$ **and** $xvec \sharp * M$ **and** $xvec \sharp * \alpha$ **and** $xvec \sharp * P'$ **and** *distinct xvec*
and *rInput*: $\bigwedge K\ Tvec.\ [\![\Psi \vdash M \leftrightarrow K; set\ xvec \subseteq supp\ N; length\ xvec = length\ Tvec; distinct\ xvec]\!] \Longrightarrow Prop\ (K(\!|N[xvec::=Tvec]|\!))\ (P[xvec::=Tvec])$
and *rBrInput*: $\bigwedge K\ Tvec.\ [\![\Psi \vdash K \succeq M; set\ xvec \subseteq supp\ N; length\ xvec = length\ Tvec; distinct\ xvec]\!] \Longrightarrow Prop\ (¿K(\!|N[xvec::=Tvec]|\!))\ (P[xvec::=Tvec])$

from *Trans* **have** $bn\ \alpha = []$
apply −
by(*ind-cases* $\Psi \rhd M(\!|\lambda * xvec\ N|\!).P \longmapsto \alpha \prec P'$) (*auto simp add*: *residualInject*)
from *Trans* **have** $distinct(bn\ \alpha)$ **by**(*auto dest*: *boundOutputDistinct*)
have $length(bn\ \alpha) = residualLength(\alpha \prec P')$ **by** *simp*
note *Trans*
moreover have $length\ xvec = inputLength(M(\!|\lambda * xvec\ N|\!).P)$ **by** *auto*
moreover note ‹*distinct xvec*›
moreover have $length\ xvec = inputLength(M(\!|\lambda * xvec\ N|\!).P)$ **by** *auto*
moreover note ‹*distinct xvec*›
moreover note ‹$length(bn\ \alpha) = residualLength(\alpha \prec P')$› ‹$distinct(bn\ \alpha)$›
moreover note ‹$length(bn\ \alpha) = residualLength(\alpha \prec P')$› ‹$distinct(bn\ \alpha)$›
moreover note ‹$length(bn\ \alpha) = residualLength(\alpha \prec P')$› ‹$distinct(bn\ \alpha)$›
moreover note ‹$length(bn\ \alpha) = residualLength(\alpha \prec P')$› ‹$distinct(bn\ \alpha)$›
moreover note ‹$length(bn\ \alpha) = residualLength(\alpha \prec P')$› ‹$distinct(bn\ \alpha)$›
moreover note ‹$length(bn\ \alpha) = residualLength(\alpha \prec P')$› ‹$distinct(bn\ \alpha)$›
moreover note ‹$length(bn\ \alpha) = residualLength(\alpha \prec P')$› ‹$distinct(bn\ \alpha)$›
moreover obtain $x::name$ **where** $x \sharp \Psi$ **and** $x \sharp P$ **and** $x \sharp M$ **and** $x \sharp xvec$ **and** $x \sharp \alpha$ **and** $x \sharp P'$ **and** $x \sharp N$
by(*generate-fresh name*) *auto*
ultimately have $Prop\ \alpha\ P'$ **using** ‹$bn\ \alpha = []$› ‹$xvec \sharp * \Psi$›‹$xvec \sharp * M$› ‹$xvec \sharp * \alpha$› ‹$xvec \sharp * P'$›
apply(*cases rule*: *semanticsCases*[*of* - - - - - - - - - - - - - $C\ x\ x\ x\ x$])

apply(*force simp add*: *residualInject psi.inject rInput*)
apply(*force simp add*: *residualInject psi.inject rBrInput*)
by(*auto simp add*: *residualInject psi.inject inputChainFresh*)+
}
note *Goal* = *this*
moreover obtain *p* :: *name prm* **where** (*p* · *xvec*) ♯* Ψ **and** (*p* · *xvec*) ♯* *M* **and** (*p* · *xvec*) ♯* *N* **and** (*p* · *xvec*) ♯* *P*
and (*p* · *xvec*) ♯* α **and** (*p* · *xvec*) ♯* *P'* **and** *S*: *set p* ⊆ *set xvec* × *set*(*p* · *xvec*)
and *distinctPerm p*
by(*rule name-list-avoiding*[**where** *xvec*=*xvec* **and** *c*=(Ψ, *M*, *N*, *P*, α, *P'*)]) *auto*
from *Trans* ‹(*p* · *xvec*) ♯* *N*› ‹(*p* · *xvec*) ♯* *P*› *S* **have** Ψ ▷ *M*⟨λ*(*p* · *xvec*) (*p* · *N*)⟩.(*p* · *P*) ⟼α ≺ *P'*
by(*simp add*: *inputChainAlpha'*)
moreover {
fix *K Tvec*
assume Ψ ⊢ *M* ↔ *K*
moreover assume *set*(*p* · *xvec*) ⊆ *supp*(*p* · *N*)
then have (*p* · *set*(*p* · *xvec*)) ⊆ (*p* · *supp*(*p* · *N*)) **by** *simp*
with ‹*distinctPerm p*› **have** *set xvec* ⊆ *supp N* **by**(*simp add*: *eqvts*)
moreover assume *length*(*p* · *xvec*) = *length*(*Tvec*::'*a list*)
then have *length xvec* = *length Tvec* **by** *simp*
moreover assume *distinct xvec*
ultimately have *Prop* (*K*⟨*N*[*xvec*::=*Tvec*]⟩) (*P*[*xvec*::=*Tvec*])
by(*rule rInput*)
with ‹*length xvec* = *length Tvec*› *S* ‹*distinctPerm p*› ‹(*p* · *xvec*) ♯* *N*› ‹(*p* · *xvec*) ♯* *P*›
have *Prop* (*K*⟨(*p* · *N*)[(*p* · *xvec*)::=*Tvec*]⟩) ((*p* · *P*)[(*p* · *xvec*)::=*Tvec*])
by(*simp add*: *renaming substTerm.renaming*)
}
moreover {
fix *K Tvec*
assume Ψ ⊢ *K* ≻ *M*
moreover assume *set*(*p* · *xvec*) ⊆ *supp*(*p* · *N*)
then have (*p* · *set*(*p* · *xvec*)) ⊆ (*p* · *supp*(*p* · *N*)) **by** *simp*
with ‹*distinctPerm p*› **have** *set xvec* ⊆ *supp N* **by**(*simp add*: *eqvts*)
moreover assume *length*(*p* · *xvec*) = *length*(*Tvec*::'*a list*)
then have *length xvec* = *length Tvec* **by** *simp*
moreover assume *distinct xvec*
ultimately have *Prop* (¿*K*⟨*N*[*xvec*::=*Tvec*]⟩) (*P*[*xvec*::=*Tvec*])
by(*rule rBrInput*)
with ‹*length xvec* = *length Tvec*› *S* ‹*distinctPerm p*› ‹(*p* · *xvec*) ♯* *N*› ‹(*p* · *xvec*) ♯* *P*›
have *Prop* (¿*K*⟨(*p* · *N*)[(*p* · *xvec*)::=*Tvec*]⟩) ((*p* · *P*)[(*p* · *xvec*)::=*Tvec*])
by(*simp add*: *renaming substTerm.renaming*)
}
moreover from *Trans* **have** *distinct xvec* **by**(*rule inputDistinct*)
then have *distinct*(*p* · *xvec*) **by** *simp*

ultimately show *?thesis* **using** ‹$(p \bullet xvec) \sharp* \Psi$› ‹$(p \bullet xvec) \sharp* M$› ‹$(p \bullet xvec) \sharp* \alpha$› ‹$(p \bullet xvec) \sharp* P'$› ‹*distinct xvec*›
by(*metis Goal*)
qed

lemma *outputCases*[*consumes 1*, *case-names cOutput cBrOutput*]:
fixes $\Psi :: 'b$
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 $\bigwedge K.\ \Psi \vdash M \leftrightarrow K \Longrightarrow Prop\ (K\langle N\rangle)\ P$
and $\bigwedge K.\ \Psi \vdash M \preceq K \Longrightarrow Prop\ (¡K\langle N\rangle)\ P$

shows $Prop\ \alpha\ P'$
using *assms*
by(*cases rule*: *semantics.cases*) (*auto simp add*: *residualInject psi.inject*)

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*: $\bigwedge \varphi\ P.\ [\![\Psi \rhd P \longmapsto Rs;\ (\varphi, P) \in set\ Cs;\ \Psi \vdash \varphi;\ guarded\ P]\!] \Longrightarrow Prop$

shows *Prop*
using *assms*
by(*cases rule*: *semantics.cases*) (*auto simp add*: *residualInject psi.inject*)

lemma *resCases*[*consumes 7*, *case-names cOpen cBrOpen cRes cBrClose*]:
fixes $\Psi :: 'b$
and $x :: name$
and $P :: ('a, 'b, 'c)\ psi$
and $\alpha :: 'a\ action$
and $P' :: ('a, 'b, 'c)\ psi$
and $C :: 'f::fs\text{-}name$

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 $rBrOpen$: $\bigwedge M\ xvec\ yvec\ y\ N\ P'.\ [\![\Psi \rhd P \longmapsto ¡M(\!|\nu*(xvec@yvec)|\!)\langle([(x,\ y)] \cdot N)\rangle \prec ([(x,\ y)] \cdot P');\ y \in supp\ N;$
$x \sharp N;\ x \sharp P';\ x \neq y;\ y \sharp xvec;\ y \sharp yvec;\ y \sharp M;$ *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 $rScope$: $\bigwedge P'.\ [\![\Psi \rhd P \longmapsto \alpha \prec P']\!] \Longrightarrow Prop\ \alpha\ ((\!|\nu x|\!)P')$
and $rBrClose$: $\bigwedge M\ xvec\ N\ P'.$
$[\![\Psi \rhd P \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec P';$
$x \in supp\ M;$
$distinct\ xvec;\ xvec \sharp*\ \Psi;\ xvec \sharp*\ P;$
$xvec \sharp*\ M;$
$x \sharp \Psi;\ x \sharp xvec;$
$xvec \sharp*\ C]\!] \Longrightarrow Prop\ (\tau)\ ((\!|\nu x|\!)((\!|\nu*xvec|\!)P'))$
shows $Prop\ \alpha\ P'$
proof −
from *Trans* **have** $distinct(bn\ \alpha)$
by(*auto dest*: *boundOutputDistinct*)
note $facts = ‹bn\ \alpha\ \sharp*\ \Psi› ‹bn\ \alpha\ \sharp*\ P› ‹bn\ \alpha\ \sharp*\ subject\ \alpha› ‹x \sharp \Psi› ‹x \sharp \alpha› ‹x \sharp P'› ‹distinct(bn\ \alpha)›$
have $length(bn\ \alpha) = residualLength(\alpha \prec P')$ **by** *simp*
note *Trans*
moreover have $length\ [] = inputLength((\!|\nu x|\!)P)$ **and** $distinct\ []$
by(*auto simp add*: *inputLength-inputLength'-inputLength''.simps*)
moreover have $length\ [] = inputLength((\!|\nu x|\!)P)$ **and** $distinct\ []$
by(*auto simp add*: *inputLength-inputLength'-inputLength''.simps*)
moreover note $‹length(bn\ \alpha) = residualLength(\alpha \prec P')› ‹distinct(bn\ \alpha)›$
moreover note $‹length(bn\ \alpha) = residualLength(\alpha \prec P')› ‹distinct(bn\ \alpha)›$
moreover note $‹length(bn\ \alpha) = residualLength(\alpha \prec P')› ‹distinct(bn\ \alpha)›$
moreover note $‹length(bn\ \alpha) = residualLength(\alpha \prec P')› ‹distinct(bn\ \alpha)›$
moreover note $‹length(bn\ \alpha) = residualLength(\alpha \prec P')› ‹distinct(bn\ \alpha)›$
moreover note $‹length(bn\ \alpha) = residualLength(\alpha \prec P')› ‹distinct(bn\ \alpha)›$
moreover note $‹length(bn\ \alpha) = residualLength(\alpha \prec P')› ‹distinct(bn\ \alpha)›$
ultimately show *?thesis* **using** *facts*
proof(*cases rule*: *semanticsCases*[*of* - - - - - - - - - - - - - $C\ x\ x\ x\ x$])
case ($cOpen\ P\ M\ xvec\ y\ yvec\ N\ P'$)
moreover then have $y \in supp\ ([(x,\ y)] \cdot N)$ **using** *facts*
apply(*clarsimp simp add*: *psi.inject alpha abs-fresh residualInject boundOut-*

putApp boundOutput.inject eqvts)
apply(*drule pt-set-bij2*[**where** *pi*=[(*x*, *y*)], **where** *x*=*x*, *OF pt-name-inst*, *OF at-name-inst*])
by(*auto simp add*: *calc-atm eqvts fresh-def*)
ultimately show *?thesis* **using** *facts*
apply(*clarsimp simp add*: *psi.inject alpha abs-fresh residualInject boundOutputApp boundOutput.inject eqvts*)
by(*rule rOpen*) (*auto simp add*: *residualInject boundOutputApp*)
next
case (*cBrOpen P M xvec y yvec N P'*)
moreover then have *y* ∈ *supp* ([(*x*, *y*)] ∙ *N*) **using** *facts*
apply(*clarsimp simp add*: *psi.inject alpha abs-fresh residualInject boundOutputApp boundOutput.inject eqvts*)
apply(*drule pt-set-bij2*[**where** *pi*=[(*x*, *y*)], **where** *x*=*x*, *OF pt-name-inst*, *OF at-name-inst*])
by(*auto simp add*: *calc-atm eqvts fresh-def*)
ultimately show *?thesis* **using** *facts*
apply(*clarsimp simp add*: *psi.inject alpha abs-fresh residualInject boundOutputApp boundOutput.inject eqvts*)
by(*rule rBrOpen*) (*auto simp add*: *residualInject boundOutputApp*)
next
qed (*auto simp add*: *psi.inject alpha abs-fresh residualInject boundOutputApp boundOutput.inject eqvts*
intro: *rScope rBrClose*)
qed

lemma *resCases'*[*consumes 7*, *case-names cOpen cBrOpen cRes cBrClose*]:
fixes Ψ :: *'b*
and *x* :: *name*
and *P* :: (*'a*, *'b*, *'c*) *psi*
and α :: *'a action*
and *P'* :: (*'a*, *'b*, *'c*) *psi*
and *C* :: *'f*::*fs-name*

assumes *Trans*: Ψ ▷ ⦇νx⦈*P* ⟼α ≺ *P'*
and *x* ♯ Ψ
and *x* ♯ α
and *x* ♯ *P'*
and *bn* α ♯* Ψ
and *bn* α ♯* *P*
and *bn* α ♯* *subject* α
and *rOpen*: ⋀*M xvec yvec y N P'*. ⟦Ψ ▷ ([(*x*, *y*)] ∙ *P*) ⟼*M*⦇ν*(*xvec*@*yvec*)⦈⟨*N*⟩ ≺ *P'*; *y* ∈ *supp N*;
x ♯ *N*; *x* ♯ *P'*; *x* ≠ *y*; *y* ♯ *xvec*; *y* ♯ *yvec*; *y* ♯ *M*; *distinct xvec*; *distinct yvec*;
xvec ♯* Ψ; *y* ♯ Ψ; *yvec* ♯* Ψ; *xvec* ♯* *P*; *y* ♯ *P*; *yvec* ♯* *P*; *xvec* ♯* *M*; *y* ♯ *M*;
yvec ♯* *M*; *xvec* ♯* *yvec*⟧ ⟹
Prop (*M*⦇ν*(*xvec*@*y*#*yvec*)⦈⟨*N*⟩) *P'*

and *rBrOpen*: $\bigwedge M\ xvec\ yvec\ y\ N\ P'.\ [\![\Psi \rhd ([(x,\ y)] \cdot P) \longmapsto ¡M(\!|\nu*(xvec@yvec)|\!)\langle N\rangle \prec P';\ y \in supp\ N;$
$x \sharp N;\ x \sharp P';\ x \neq y;\ y \sharp xvec;\ y \sharp yvec;\ y \sharp M;$ *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*: $\bigwedge P'.\ [\![\Psi \rhd P \longmapsto \alpha \prec P']\!] \Longrightarrow Prop\ \alpha\ ((\!|\nu x|\!)P')$
and *rBrClose*: $\bigwedge M\ xvec\ N\ P'.$
$[\![\Psi \rhd P \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec P';$
$x \in supp\ M;$
distinct xvec; $xvec \sharp* \Psi;\ xvec \sharp* P;$
$xvec \sharp* M;$
$x \sharp \Psi;\ x \sharp xvec;$
$xvec \sharp* C]\!] \Longrightarrow Prop\ (\tau)\ ((\!|\nu x|\!)((\!|\nu*xvec|\!)P'))$

shows $Prop\ \alpha\ P'$
proof −
from *Trans* **have** $distinct(bn\ \alpha)$
by(*auto dest*: *boundOutputDistinct*)
note $facts = \langle bn\ \alpha \sharp* \Psi\rangle\ \langle bn\ \alpha \sharp* P\rangle\ \langle bn\ \alpha \sharp* subject\ \alpha\rangle\ \langle x \sharp \Psi\rangle\ \langle x \sharp \alpha\rangle\ \langle x \sharp P'\rangle\ \langle distinct(bn\ \alpha)\rangle$
have $length(bn\ \alpha) = residualLength(\alpha \prec P')$
by *simp*
note *Trans*
moreover have $length\ [] = inputLength((\!|\nu x|\!)P)$ **and** $distinct\ []$
by(*auto simp add*: *inputLength-inputLength'-inputLength''.simps*)
moreover have $length\ [] = inputLength((\!|\nu x|\!)P)$ **and** $distinct\ []$
by(*auto simp add*: *inputLength-inputLength'-inputLength''.simps*)
moreover note $\langle length(bn\ \alpha) = residualLength(\alpha \prec P')\rangle\ \langle distinct(bn\ \alpha)\rangle$
moreover note $\langle length(bn\ \alpha) = residualLength(\alpha \prec P')\rangle\ \langle distinct(bn\ \alpha)\rangle$
moreover note $\langle length(bn\ \alpha) = residualLength(\alpha \prec P')\rangle\ \langle distinct(bn\ \alpha)\rangle$
moreover note $\langle length(bn\ \alpha) = residualLength(\alpha \prec P')\rangle\ \langle distinct(bn\ \alpha)\rangle$
moreover note $\langle length(bn\ \alpha) = residualLength(\alpha \prec P')\rangle\ \langle distinct(bn\ \alpha)\rangle$
moreover note $\langle length(bn\ \alpha) = residualLength(\alpha \prec P')\rangle\ \langle distinct(bn\ \alpha)\rangle$
moreover note $\langle length(bn\ \alpha) = residualLength(\alpha \prec P')\rangle\ \langle distinct(bn\ \alpha)\rangle$
ultimately show *?thesis* **using** *facts*
proof(*cases rule*: *semanticsCases*[*of* - - - - - - - - - - - - - *C x x x x*])
case (*cOpen P M xvec y yvec N P'*)
moreover then have $y \in supp\ ([(x,\ y)] \cdot N)$ **using** *facts*
apply(*clarsimp simp add*: *psi.inject alpha abs-fresh residualInject boundOutputApp boundOutput.inject eqvts*)
apply(*drule pt-set-bij2*[**where** *pi*=$[(x,\ y)]$, *OF pt-name-inst*, *OF at-name-inst*])
by(*auto simp add*: *calc-atm eqvts fresh-def*)
ultimately show *?thesis* **using** *facts*
apply(*clarsimp simp add*: *psi.inject alpha abs-fresh residualInject boundOutputApp boundOutput.inject eqvts*)
apply(*rule rOpen*) — 20 subgoals

apply(*drule semantics.eqvt*[**where** *pi*=[(*x*, *y*)]])
apply(*auto simp add*: *eqvts residualInject boundOutputApp*)
done
next
case (*cBrOpen P M xvec y yvec N P'*)
moreover then have *y* ∈ *supp* ([(*x*, *y*)] ∙ *N*) **using** *facts*
apply(*clarsimp simp add*: *psi.inject alpha abs-fresh residualInject boundOutputApp boundOutput.inject eqvts*)
apply(*drule pt-set-bij2*[**where** *pi*=[(*x*, *y*)], *OF pt-name-inst*, *OF at-name-inst*])
by(*auto simp add*: *calc-atm eqvts fresh-def*)
ultimately show *?thesis* **using** *facts*
apply(*clarsimp simp add*: *psi.inject alpha abs-fresh residualInject boundOutputApp boundOutput.inject eqvts*)
apply(*rule rBrOpen*) — 20 subgoals
apply(*drule semantics.eqvt*[**where** *pi*=[(*x*, *y*)]])
apply(*auto simp add*: *eqvts residualInject boundOutputApp*)
done
qed (*auto simp add*: *psi.inject alpha abs-fresh residualInject boundOutputApp boundOutput.inject eqvts*
intro: *rScope rBrClose*)
qed

lemma *resInputCases'*[*consumes 4*, *case-names cRes*]:
fixes Ψ :: *'b*
and *x* :: *name*
and *M* :: *'a*
and *N* :: *'a*
and *P* :: (*'a*, *'b*, *'c*) *psi*
and *R* :: (*'a*, *'b*, *'c*) *psi*
and *C* :: *'f*::*fs-name*

assumes *Trans*: Ψ ▷ ⦇ν*x*⦈*P* ⟼*M*⦇*N*⦈ ≺ *R*
and *2*: *x* ♯ Ψ
and *x* ♯ (*M*⦇*N*⦈)
and *4*: *x* ♯ *R*
and *rScope*: ⋀*P'*. ⟦Ψ ▷ *P* ⟼*M*⦇*N*⦈ ≺ *P'*⟧ ⟹ *Prop* (⦇ν*x*⦈*P'*)

shows *Prop R*
proof −
from *Trans* **obtain** α **where** *1*: Ψ ▷ ⦇ν*x*⦈*P* ⟼α ≺ *R* **and** *5*: *bn* α ♯* Ψ **and** *6*: *bn* α ♯* *P* **and** *7*: *bn* α ♯* *subject* α **and** α = *M*⦇*N*⦈ **by** *auto*
from ‹*x* ♯ (*M*⦇*N*⦈)› ‹α = (*M*⦇*N*⦈)› **have** *3*: *x* ♯ α **by** *simp*
show *?thesis* **using** *1 2 3 4 5 6 7* ‹α = *M*⦇*N*⦈› *rScope*
by(*induct rule*: *resCases'*) (*auto simp add*: *residualInject*)
qed

lemma *resBrInputCases'*[*consumes 4*, *case-names cRes*]:
fixes Ψ :: *'b*
and *x* :: *name*

and M :: $'a$
and N :: $'a$
and P :: $('a, 'b, 'c)$ *psi*
and R :: $('a, 'b, 'c)$ *psi*
and C :: $'f$::*fs-name*

assumes *Trans*: $\Psi \rhd (\!|\nu x|\!)P \longmapsto ¿M(\!|N|\!) \prec R$
and *2*: $x \sharp \Psi$
and $x \sharp (¿M(\!|N|\!))$
and *4*: $x \sharp R$
and *rScope*: $\bigwedge P'.\ [\![\Psi \rhd P \longmapsto ¿M(\!|N|\!) \prec P']\!] \Longrightarrow$ *Prop* $((\!|\nu x|\!)P')$

shows *Prop* R
proof −
from *Trans* **obtain** α **where** *1*: $\Psi \rhd (\!|\nu x|\!)P \longmapsto \alpha \prec R$ **and** *5*: *bn* $\alpha \sharp* \Psi$ **and** *6*: *bn* $\alpha \sharp* P$ **and** *7*: *bn* $\alpha \sharp* $ *subject* α **and** $\alpha = ¿M(\!|N|\!)$ **by** *auto*
from $\langle x \sharp (¿M(\!|N|\!))\rangle$ $\langle \alpha = (¿M(\!|N|\!))\rangle$ **have** *3*: $x \sharp \alpha$ **by** *simp*
show *?thesis* **using** *1 2 3 4 5 6 7* $\langle \alpha = ¿M(\!|N|\!)\rangle$ *rScope*
by(*induct rule*: *resCases'*) (*auto simp add*: *residualInject*)
qed

lemma *resOutputCases'''*[*consumes 7*, *case-names cOpen cRes*]:
fixes Ψ :: $'b$
and x :: *name*
and M :: $'a$
and N :: $'a$
and P :: $('a, 'b, 'c)$ *psi*
and R :: $('a, 'b, 'c)$ *psi*
and C :: $'f$::*fs-name*

assumes *Trans*: $\Psi \rhd (\!|\nu x|\!)P \longmapsto M(\!|\nu*(zvec1\ @\ zvec2)|\!)\langle N\rangle \prec R$
and *1*: $x \sharp \Psi$
and $x \sharp (M(\!|\nu*(zvec1\ @\ zvec2)|\!)\langle N\rangle)$
and *3*: $x \sharp R$
and $(zvec1\ @\ zvec2) \sharp* \Psi$
and $(zvec1\ @\ zvec2) \sharp* P$
and $(zvec1\ @\ zvec2) \sharp* M$
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 P'
and *rScope*: $\bigwedge P'.\ [\![\Psi \rhd P \longmapsto M(\!|\nu*(zvec1\ @\ zvec2)|\!)\langle N\rangle \prec P']\!] \Longrightarrow$ *Prop* $((\!|\nu x|\!)P')$

shows *Prop* R

proof −
from *Trans* **have** *distinct* (*zvec1* @ *zvec2*) **by**(*auto dest*: *boundOutputDistinct*)
obtain α **where** $\alpha{=}M\langle\!|\nu*(zvec1\ @\ zvec2)|\!\rangle\langle N\rangle$ **by** *simp*
with *Trans* ‹(*zvec1* @ *zvec2*) $\sharp*$ Ψ› ‹(*zvec1* @ *zvec2*) $\sharp*$ P› ‹(*zvec1* @ *zvec2*) $\sharp*$ M›
have α*Trans*: $\Psi \rhd \langle\!|\nu x|\!\rangle P \longmapsto \alpha \prec R$ **and** *4*: *bn* α $\sharp*$ Ψ **and** *5*: *bn* α $\sharp*$ P **and** *6*: *bn* α $\sharp*$ *subject* α
by *simp*+
from ‹x $\sharp$ $(M\langle\!|\nu*(zvec1\ @\ zvec2)|\!\rangle\langle N\rangle)$› ‹$\alpha{=}M\langle\!|\nu*(zvec1\ @\ zvec2)|\!\rangle\langle N\rangle$› **have** *2*: x $\sharp$ α **by** *simp*
show *?thesis* **using** α*Trans 1 2 3 4 5 6* ‹$\alpha{=}M\langle\!|\nu*(zvec1\ @\ zvec2)|\!\rangle\langle N\rangle$› *rOpen rScope*
proof(*induct rule*: *resCases'*[**where** $C{=}(zvec1,\ zvec2,\ C)$])
case *cBrOpen*
then show *?case*
by(*auto simp add*: *residualInject boundOutputApp*)
next
case *cRes*
then show *?case*
by(*auto simp add*: *residualInject boundOutputApp*)
next
case *cBrClose*
then show *?case*
by(*auto simp add*: *residualInject boundOutputApp*)
next
case(*cOpen* M' *xvec yvec y* N' P')
then show *?case*
by(*auto simp add*: *residualInject boundOutputApp*)
qed
qed

lemma *resOutputCases''*[*consumes 7*, *case-names cOpen cRes*]:
fixes Ψ :: *'b*
and x :: *name*
and z :: *name*
and M :: *'a*
and N :: *'a*
and P :: (*'a*, *'b*, *'c*) *psi*
and R :: (*'a*, *'b*, *'c*) *psi*
and C :: *'f*::*fs-name*

assumes *Trans*: $\Psi \rhd \langle\!|\nu x|\!\rangle P \longmapsto M\langle\!|\nu*(zvec1@y\#zvec2)|\!\rangle\langle N\rangle \prec R$
and *1*: x $\sharp$ Ψ
and x $\sharp$ $(M\langle\!|\nu*(zvec1@y\#zvec2)|\!\rangle\langle N\rangle)$
and *3*: x $\sharp$ R
and $(zvec1@y\#zvec2)$ $\sharp*$ Ψ
and $(zvec1@y\#zvec2)$ $\sharp*$ P
and $(zvec1@y\#zvec2)$ $\sharp*$ M
and *rOpen*: $\bigwedge M\ xvec\ yvec\ y\ N\ P'.\ [\![\Psi \rhd ([(x,\ y)] \cdot P) \longmapsto M\langle\!|\nu*(xvec@yvec)|\!\rangle\langle N\rangle$

$\prec P'$; $y \in supp\ N$;
$x \sharp N$; $x \sharp P'$; $x \neq y$; $y \sharp xvec$; $y \sharp yvec$; $y \sharp M$; *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 P'
and *rScope*: $\bigwedge P'.\ [\![\Psi \rhd P \longmapsto M(\!|\nu*(zvec1@y\#zvec2)|\!)\langle N\rangle \prec P']\!] \Longrightarrow$ *Prop* $((\!|\nu x|\!)P')$

shows *Prop R*
proof −
from *Trans* **have** *distinct* $(zvec1@y\#zvec2)$ **by**(*auto dest*: *boundOutputDistinct*)
obtain α **where** $\alpha{=}M(\!|\nu*(zvec1@y\#zvec2)|\!)\langle N\rangle$ **by** *simp*
with *Trans* ‹$(zvec1@y\#zvec2) \sharp* \Psi$› ‹$(zvec1@y\#zvec2) \sharp* P$› ‹$(zvec1@y\#zvec2) \sharp* M$›
have $\alpha Trans$: $\Psi \rhd (\!|\nu x|\!)P \longmapsto \alpha \prec R$ **and** *4*: $bn\ \alpha \sharp* \Psi$ **and** *5*: $bn\ \alpha \sharp* P$ **and** *6*: $bn\ \alpha \sharp* subject\ \alpha$
by *simp+*
from ‹$x \sharp (M(\!|\nu*(zvec1@y\#zvec2)|\!)\langle N\rangle)$› ‹$\alpha{=}M(\!|\nu*(zvec1@y\#zvec2)|\!)\langle N\rangle$› **have** *2*: $x \sharp \alpha$ **by** *simp*
show *?thesis* **using** $\alpha Trans$ *1 2 3 4 5 6* ‹$\alpha{=}M(\!|\nu*(zvec1@y\#zvec2)|\!)\langle N\rangle$› *rOpen rScope*
proof(*induct rule*: *resCases'*[**where** $C{=}(zvec1, zvec2, z, C)$])
case *cBrOpen*
then show *?case*
by(*auto simp add*: *residualInject boundOutputApp*)
next
case *cRes*
then show *?case*
by(*auto simp add*: *residualInject boundOutputApp*)
next
case *cBrClose*
then show *?case*
by(*auto simp add*: *residualInject boundOutputApp*)
next
case(*cOpen* M' *xvec yvec y* N' P')
then show *?case*
by(*auto simp add*: *residualInject boundOutputApp*)
qed
qed

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 $\Psi' :: 'b$

assumes $\Psi \rhd P \longmapsto Rs$
and $\Psi \simeq \Psi'$

shows $\Psi' \rhd P \longmapsto Rs$
using *assms*
proof(*nominal-induct avoiding*: Ψ' *rule*: *semantics.strong-induct*)
case(*cInput* Ψ M K *xvec* N *Tvec* P Ψ')
from ‹$\Psi \simeq \Psi'$› ‹$\Psi \vdash M \leftrightarrow K$› **have** $\Psi' \vdash M \leftrightarrow K$
by(*simp add*: *AssertionStatImp-def AssertionStatEq-def*)
then show *?case* **using** ‹*distinct xvec*› ‹*set xvec* ⊆ *supp N*› ‹*length xvec* = *length Tvec*›
by(*rule Input*)
next
case(*cBrInput* Ψ K M *xvec* N *Tvec* P Ψ')
from ‹$\Psi \simeq \Psi'$› ‹$\Psi \vdash K \succeq M$› **have** $\Psi' \vdash K \succeq M$
by(*simp add*: *AssertionStatImp-def AssertionStatEq-def*)
then show *?case* **using** ‹*distinct xvec*› ‹*set xvec* ⊆ *supp N*› ‹*length xvec* = *length Tvec*›
by(*rule BrInput*)
next
case(*Output* Ψ M K N P Ψ')
from ‹$\Psi \simeq \Psi'$› ‹$\Psi \vdash M \leftrightarrow K$› **have** $\Psi' \vdash M \leftrightarrow K$
by(*simp add*: *AssertionStatImp-def AssertionStatEq-def*)
then show *?case* **by**(*rule semantics.Output*)
next
case(*BrOutput* Ψ M K N P Ψ')
from ‹$\Psi \simeq \Psi'$› ‹$\Psi \vdash M \preceq K$› **have** $\Psi' \vdash M \preceq K$
by(*simp add*: *AssertionStatImp-def AssertionStatEq-def*)
then show *?case* **by**(*rule semantics.BrOutput*)
next
case(*Case* Ψ P Rs φ Cs Ψ')
then have $\Psi' \rhd P \longmapsto Rs$ **by**(*intro Case*)
moreover note ‹$(\varphi, P) \in set\ Cs$›
moreover from ‹$\Psi \simeq \Psi'$› ‹$\Psi \vdash \varphi$› **have** $\Psi' \vdash \varphi$
by(*simp add*: *AssertionStatImp-def AssertionStatEq-def*)
ultimately show *?case* **using** ‹*guarded P*› **by**(*rule semantics.Case*)
next
case(*cPar1* Ψ ΨQ P α P' *xvec* Q Ψ')
then show *?case*
by(*intro Par1*) (*auto intro*: *Composition*)
next
case(*cPar2* Ψ ΨP Q α Q' *xvec* P Ψ')
then show *?case*
by(*intro Par2*) (*auto intro*: *Composition*)
next
case(*cComm1* Ψ ΨQ P M N P' *xvec* ΨP Q K *zvec* Q' *yvec* Ψ')

then show *?case*
by(*clarsimp*, *intro Comm1*) (*blast intro*: *Composition statEqEnt*)+
next
case(*cComm2* Ψ ΨQ P M *zvec* N P' *xvec* ΨP Q K Q' *yvec* Ψ')
then show *?case*
by(*clarsimp*, *intro Comm2*) (*blast intro*: *Composition statEqEnt*)+
next
case(*cBrMerge* Ψ Ψ_Q P M N P' A_P Ψ_P Q Q' A_Q Ψ')
then show *?case*
by(*clarsimp*, *intro BrMerge*) (*blast intro*: *Composition*)+
next
case(*cBrComm1* Ψ Ψ_Q P M N P' A_P Ψ_P Q *xvec* Q' A_Q Ψ')
then show *?case*
by(*clarsimp*, *intro BrComm1*) (*blast intro*: *Composition*)+
next
case(*cBrComm2* Ψ Ψ_Q P M *xvec* N P' A_P Ψ_P Q Q' A_Q Ψ')
then show *?case*
by(*clarsimp*, *intro BrComm2*) (*blast intro*: *Composition*)+
next
case(*cBrClose* Ψ P M *xvec* N P' x Ψ')
then show *?case* **by**(*force intro*: *BrClose*)
next
case(*cOpen* Ψ P M *xvec* N P' x *yvec* Ψ')
then show *?case* **by**(*force intro*: *Open*)
next
case(*cBrOpen* Ψ P M *xvec* N P' x *yvec* Ψ')
then show *?case* **by**(*force intro*: *BrOpen*)
next
case(*cScope* Ψ P α P' x Ψ')
then show *?case* **by**(*force intro*: *Scope*)
next
case(*Bang* Ψ P *Rs* Ψ')
then show *?case* **by**(*force intro*: *semantics.Bang*)
qed

lemma *brInputTermSupp*:
fixes Ψ :: *'b*::*fs-name*
and P :: (*'a*, *'b*, (*'c*::*fs-name*)) *psi*
and P' :: (*'a*, *'b*, *'c*) *psi*
and N :: *'a*::*fs-name*
and K :: *'a*::*fs-name*

assumes $\Psi \rhd P \longmapsto ¿K\langle\!|N|\!\rangle \prec P'$

shows (*supp* K) $\subseteq$ ((*supp* P)::*name set*)
using *assms*
proof(*nominal-induct rule*: *brInputInduct*)
case(*cBrInput* Ψ K M *xvec* N *Tvec* P)
from ‹$\Psi \vdash K \succeq M$› **have** (*supp* K) $\subseteq$ ((*supp* M)::*name set*)

by(*simp add*: *chanInConSupp*)
then show *?case*
by (*metis Un-commute Un-upper2 psi.supp*(*3*) *subset-trans*)
next
case(*cCase* Ψ *P M N P'* φ *Cs*)
then have *supp M* $\subseteq$ ((*supp P*)::*name set*)
by *simp*
from ‹(φ, *P*) $\in$ *set Cs*›
have {(φ, *P*)} $\subseteq$ *set Cs*
by *simp*
moreover have *finite* {(φ, *P*)} **by** *simp*
moreover have *finite* (*set Cs*) **by** *simp*
ultimately have ((*supp* {(φ, *P*)})::*name set*) $\subseteq$ ((*supp* (*set Cs*))::*name set*)
by(*simp add*: *supp-subset*)

moreover have *supp* {(φ, *P*)} = ((*supp* (φ, *P*))::*name set*)
by (*meson supp-singleton*)
moreover have *supp P* $\subseteq$ *supp* (φ, *P*)
by (*metis Un-upper2 supp-prod*)
ultimately have ((*supp P*)::*name set*) $\subseteq$ ((*supp Cs*)::*name set*)
by (*auto simp add*: *supp-list-set*)

moreover have ((*supp Cs*)::*name set*) = ((*supp* (*Cases Cs*))::*name set*)
unfolding *psi.supp*
apply(*induct rule*: *psiCases.induct*)
apply(*metis psiCase.supp*(*1*) *psiCases.simps*(*1*) *set-empty2 supp-list-nil*)
by *simp* (*metis Un-assoc psiCase.supp*(*2*) *supp-list-cons supp-prod*)

ultimately have ((*supp P*)::*name set*) $\subseteq$ ((*supp* (*Cases Cs*))::*name set*)
by *simp*

with ‹*supp M* $\subseteq$ *supp P*›
show *?case* **by** *simp*
next
case(*cPar1* Ψ Ψ_Q *P M N P'* A_Q *Q*)
then have ((*supp M*)::*name set*) $\subseteq$ ((*supp P*)::*name set*)
by *auto*
then show *?case*
by(*auto simp add*: *psi.supp*)
next
case(*cPar2* Ψ Ψ_P *Q M N Q'* A_P *P*)
then have ((*supp M*)::*name set*) $\subseteq$ ((*supp Q*)::*name set*)
by *auto*
then show *?case*
by(*auto simp add*: *psi.supp*)
next
case(*cBrMerge* Ψ Ψ_Q *P M N P'* A_P Ψ_P *Q Q'* A_Q)
then show *?case*
by(*auto simp add*: *psi.supp*)

next
case(*cScope* Ψ *P M N P′ x*)
then have ((*supp M*)::*name set*) ⊆ ((*supp P*)::*name set*)
by *simp*
then show *?case*
by(*simp add*: *psi.supp*) (*metis abs-supp cScope.hyps fresh-def insert-Diff-single subset-insert-iff*)
next
case(*cBang* Ψ *P M N P′*)
then show *?case*
by(*simp add*: *psi.supp*)
qed

lemma *brOutputTermSupp*:
fixes Ψ :: *′b*::*fs-name*
and *P* :: (*′a*, *′b*, (*′c*::*fs-name*)) *psi*
and *P′* :: (*′a*, *′b*, *′c*) *psi*
and *N* :: *′a*::*fs-name*
and *K* :: *′a*::*fs-name*
and *xvec* :: *name list*

assumes Ψ ▷ *P* ⟼ *RBrOut K* (⦇*ν*∗*xvec*⦈*N* ≺*′ P′*)

shows (*supp K*) ⊆ ((*supp P*)::*name set*)
using *assms*
proof(*nominal-induct rule*: *brOutputInduct*)
case(*cBrOutput* Ψ *M K N P*)
from ‹Ψ ⊢ *M* ≼ *K*› **have** (*supp K*) ⊆ ((*supp M*)::*name set*)
by(*simp add*: *chanOutConSupp*)
then show *?case*
by (*metis Un-commute Un-upper2 psi.supp*(*2*) *subset-iff-psubset-eq subset-trans*)
next
case(*cCase* Ψ *P M B φ Cs*)
then have *supp M* ⊆ ((*supp P*)::*name set*)
by *simp*
from ‹(*φ*, *P*) ∈ *set Cs*›
have {(*φ*, *P*)} ⊆ *set Cs*
by *simp*
moreover have *finite* {(*φ*, *P*)} **by** *simp*
moreover have *finite* (*set Cs*) **by** *simp*
ultimately have ((*supp* {(*φ*, *P*)})::*name set*) ⊆ ((*supp* (*set Cs*))::*name set*)
by(*simp add*: *supp-subset*)

moreover have *supp* {(*φ*, *P*)} = ((*supp* (*φ*, *P*))::*name set*)
by(*meson supp-singleton*)
moreover have *supp P* ⊆ *supp* (*φ*, *P*)
by (*metis Un-upper2 supp-prod*)
ultimately have ((*supp P*)::*name set*) ⊆ ((*supp Cs*)::*name set*)
by (*auto simp add*: *supp-list-set*)

```
  moreover have ((supp Cs)::name set) = ((supp (Cases Cs))::name set)
    unfolding psi.supp
    apply(induct rule: psiCases.induct)
     apply(metis psiCase.supp(1) psiCases.simps(1) set-empty2 supp-list-nil)
    by simp (metis Un-assoc psiCase.supp(2) supp-list-cons supp-prod)

  ultimately have ((supp P)::name set) ⊆ ((supp (Cases Cs))::name set)
    by simp

  with ‹supp M ⊆ supp P›
  show ?case by simp
next
  case(cPar1 Ψ Ψ_Q P M xvec N P' A_Q Q)
  then have ((supp M)::name set) ⊆ ((supp P)::name set)
    by auto
  then show ?case
    by(auto simp add: psi.supp)
next
  case(cPar2 Ψ Ψ_P Q M xvec N Q' A_P P)
  then have ((supp M)::name set) ⊆ ((supp Q)::name set)
    by auto
  then show ?case
    by(auto simp add: psi.supp)
next
  case(cBrComm1 Ψ Ψ_Q P M N P' A_P Ψ_P Q xvec Q' A_Q)
  then have ((supp M)::name set) ⊆ ((supp Q)::name set)
    by auto
  then show ?case
    by(auto simp add: psi.supp)
next
  case(cBrComm2 Ψ Ψ_Q P M xvec N P' A_P Ψ_P Q Q' A_Q)
  then have ((supp M)::name set) ⊆ ((supp P)::name set)
    by auto
  then show ?case
    by(auto simp add: psi.supp)
next
  case(cBrOpen Ψ P M xvec yvec N P' x)
  then have ((supp M)::name set) ⊆ ((supp P)::name set)
    by simp
  with ‹x ♯ M›
  show ?case
   by(simp add: psi.supp) (metis abs-supp cBrOpen.hyps fresh-def insert-Diff-single
subset-insert-iff)
next
  case(cScope Ψ P M xvec N P' x)
  then have ((supp M)::name set) ⊆ ((supp P)::name set)
    by simp
  then show ?case
```

by(*simp add*: *psi.supp*) (*metis abs-supp cScope.hyps fresh-def insert-Diff-single subset-insert-iff*)
next
case *cBang*
then show *?case*
by(*simp add*: *psi.supp*)
qed

lemma *actionPar1Dest'*:
fixes α :: ('*a*::*fs-name*) *action*
and *P* :: ('*a*, '*b*::*fs-name*, '*c*::*fs-name*) *psi*
and β :: ('*a*::*fs-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* α $\sharp*$ *R*
and *bn* β $\sharp*$ *R*

obtains *T* **where** $P = T \parallel R$ **and** $\alpha \prec T = \beta \prec Q$
using *assms*
apply(*cases rule*: *actionCases*[**where** α=α])
apply (*metis residualInject'*(*1*))
apply (*metis residualInject'*(*7*))
apply (*smt* (*z3*) *bn.simps*(*3*) *boundOutputPar1Dest create-residual.simps*(*3*) *residualInject'*(*8*))
apply (*smt* (*z3*) *bn.simps*(*4*) *boundOutputPar1Dest create-residual.simps*(*4*) *residualInject'*(*9*))
by (*metis residualInject'*(*10*))

lemma *actionPar2Dest'*:
fixes α :: ('*a*::*fs-name*) *action*
and *P* :: ('*a*, '*b*::*fs-name*, '*c*::*fs-name*) *psi*
and β :: ('*a*::*fs-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* α $\sharp*$ *Q*
and *bn* β $\sharp*$ *Q*

obtains *T* **where** $P = Q \parallel T$ **and** $\alpha \prec T = \beta \prec R$
using *assms*
apply(*cases rule*: *actionCases*[**where** α=α])
apply (*metis residualInject'*(*1*))
apply (*metis residualInject'*(*7*))
apply (*smt* (*z3*) *bn.simps*(*3*) *boundOutputPar2Dest create-residual.simps*(*3*) *residualInject'*(*8*))
apply (*smt* (*z3*) *bn.simps*(*4*) *boundOutputPar2Dest create-residual.simps*(*4*)

residualInject'(9))
by (*metis residualInject'*(10))

lemma *expandNonTauFrame*:
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 :: *'f*::*fs-name*
and C' :: *'g*::*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 $A_P \sharp* P$
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 −
assume *A*: $\bigwedge p\ \Psi'\ \Psi_P'\ A_P'$.
⟦*set* $p \subseteq set(bn\ \alpha) \times set(bn(p \cdot \alpha))$; $(p \cdot \Psi_P) \otimes \Psi' \simeq \Psi_P'$; *distinctPerm p*;
extractFrame $P' = \langle A_P', \Psi_P' \rangle$; $A_P' \sharp* P'$; $A_P' \sharp* \alpha$; $A_P' \sharp* (p \cdot \alpha)$;
$A_P' \sharp* C$; $(bn(p \cdot \alpha)) \sharp* C'$; $(bn(p \cdot \alpha)) \sharp* \alpha$; $(bn(p \cdot \alpha)) \sharp* P'$; *distinct* A_P'⟧
⟹ *thesis*

from *Trans* **have** *distinct*(*bn* α) **by**(*auto dest*: *boundOutputDistinct*)

with *Trans* ‹*bn* $\alpha \sharp* subject\ \alpha$› ‹$A_P \sharp* P$› ‹$A_P \sharp* \alpha$› **have** $A_P \sharp* P'$
by(*drule-tac freeFreshChainDerivative*) *auto*

{
fix *V* :: *'a list*
and *W* :: (*'a action*) *list*
and *X* :: *name list*

and $Y :: 'b\ list$
and $Z :: ('a, 'b, 'c)\ psi\ list$

assume $bn\ \alpha \sharp* V$ **and** $bn\ \alpha \sharp* W$ **and** $bn\ \alpha \sharp* X$ **and** $bn\ \alpha \sharp* Y$ **and** $bn\ \alpha \sharp* Z$ **and** $A_P \sharp* V$ **and** $A_P \sharp* W$ **and** $A_P \sharp* X$ **and** $A_P \sharp* Y$ **and** $A_P \sharp* Z$

with *assms* **obtain** $p\ \Psi'\ A_P'\ \Psi_P'$ **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 $A_P' \sharp* V$ **and** $A_P' \sharp* W$ **and** $A_P' \sharp* X$ **and** $A_P' \sharp* Y$ **and** $A_P' \sharp* Z$ **and** *distinct* A_P'
and $(bn(p \cdot \alpha)) \sharp* V$ **and** $(bn(p \cdot \alpha)) \sharp* W$ **and** $(bn(p \cdot \alpha)) \sharp* X$ **and** $(bn(p \cdot \alpha)) \sharp* Y$ **and** $(bn(p \cdot \alpha)) \sharp* Z$
using $\langle A_P \sharp* P' \rangle$ $\langle distinct(bn\ \alpha) \rangle$
proof(*nominal-induct* $\Psi\ P\ Rs == \alpha \prec P'\ A_P\ \Psi_P$ *avoiding*: $C\ C'\ \alpha\ P'\ V\ W\ X\ Y\ Z$ *arbitrary*: *thesis rule*: *semanticsFrameInduct*)
case(*cAlpha* $\Psi\ P\ A_P\ \Psi_P\ p\ C\ C'\ \alpha\ P'\ V\ W\ X\ Y\ Z$)
then obtain $q\ \Psi'\ A_P'\ \Psi_P'$ **where** *Sq*: $set\ q \subseteq set(bn\ \alpha) \times set(bn(q \cdot \alpha))$ **and** *PeqP'*: $(q \cdot \Psi_P) \otimes \Psi' \simeq \Psi_P'$ **and** *distinctPerm q*
and *FrP'*: *extractFrame* $P' = \langle A_P', \Psi_P' \rangle$ **and** $A_P' \sharp* P'$ **and** $A_P' \sharp* \alpha$ **and** $A_P' \sharp* (q \cdot \alpha)$
and $A_P' \sharp* C$ **and** $(bn(q \cdot \alpha)) \sharp* C'$ **and** $(bn(q \cdot \alpha)) \sharp* \alpha$ **and** $(bn(q \cdot \alpha)) \sharp* P'$
and $A_P' \sharp* V$ **and** $A_P' \sharp* W$ **and** $A_P' \sharp* X$ **and** $A_P' \sharp* Y$ **and** $A_P' \sharp* Z$ **and** *distinct* A_P'
and $(bn(q \cdot \alpha)) \sharp* V$ **and** $(bn(q \cdot \alpha)) \sharp* W$ **and** $(bn(q \cdot \alpha)) \sharp* X$ **and** $(bn(q \cdot \alpha)) \sharp* Y$ **and** $(bn(q \cdot \alpha)) \sharp* Z$
by *metis*

have *Sp*: $set\ p \subseteq set\ A_P \times set\ (p \cdot A_P)$ **by** *fact*

from *Sq* **have** $(p \cdot set\ q) \subseteq p \cdot (set(bn\ \alpha) \times set(bn(q \cdot \alpha)))$
by(*simp add*: *subsetClosed*)
then have $set(p \cdot q) \subseteq set(bn(p \cdot \alpha)) \times set(p \cdot bn(q \cdot \alpha))$
by(*simp add*: *eqvts*)
with $\langle A_P \sharp* \alpha \rangle$ $\langle (p \cdot A_P) \sharp* \alpha \rangle$ *Sp* **have** $set(p \cdot q) \subseteq set(bn\ \alpha) \times set(bn((p \cdot q) \cdot \alpha))$
by(*simp add*: *perm-compose bnEqvt*[*symmetric*])
moreover from *PeqP'* **have** $(p \cdot (q \cdot \Psi_P) \otimes \Psi') \simeq (p \cdot \Psi_P')$
by(*simp add*: *AssertionStatEqClosed*)
then have $((p \cdot q) \cdot p \cdot \Psi_P) \otimes (p \cdot \Psi') \simeq (p \cdot \Psi_P')$
apply(*subst perm-compose*[*symmetric*])
by(*simp add*: *eqvts*)
moreover from $\langle distinctPerm\ q \rangle$ **have** *distinctPerm* $(p \cdot q)$
by *simp*
moreover from $\langle (bn(q \cdot \alpha)) \sharp* C' \rangle$ **have** $(p \cdot bn(q \cdot \alpha)) \sharp* (p \cdot C')$

by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹A_P ♯* α› ‹$(p \cdot A_P)$ ♯* α› ‹A_P ♯* C'› ‹$(p \cdot A_P)$ ♯* C'› *Sp* **have** *bn*((p · q) · α) ♯* C'
by(*simp add*: *perm-compose bnEqvt*[*symmetric*])
moreover from *FrP'* **have** $(p \cdot \mathit{extractFrame}\ P') = p \cdot \langle A_P', \Psi_P' \rangle$ **by** *simp*
with ‹A_P ♯* P'› ‹$(p \cdot A_P)$ ♯* P'› *Sp* **have** *extractFrame* $P' = \langle p \cdot A_P', p \cdot \Psi_P' \rangle$
by(*simp add*: *eqvts*)
moreover from ‹A_P' ♯* P'› **have** $(p \cdot A_P')$ ♯* $(p \cdot P')$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹A_P ♯* P'› ‹$(p \cdot A_P)$ ♯* P'› *Sp* **have** $(p \cdot A_P')$ ♯* P' **by** *simp*
moreover from ‹A_P' ♯* α› **have** $(p \cdot A_P')$ ♯* $(p \cdot \alpha)$
by(*simp only*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹A_P ♯* α› ‹$(p \cdot A_P)$ ♯* α› *Sp* **have** $(p \cdot A_P')$ ♯* α **by** *simp*
moreover from ‹A_P' ♯* C› **have** $(p \cdot A_P')$ ♯* $(p \cdot C)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹A_P ♯* C› ‹$(p \cdot A_P)$ ♯* C› *Sp* **have** $(p \cdot A_P')$ ♯* C **by** *simp*
moreover from ‹$(bn(q \cdot \alpha))$ ♯* α› **have** $(p \cdot bn(q \cdot \alpha))$ ♯* $(p \cdot \alpha)$
by(*simp only*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹A_P ♯* α› ‹$(p \cdot A_P)$ ♯* α› ‹A_P ♯* α› ‹$(p \cdot A_P)$ ♯* α› *Sp* **have** *bn*((p · q) · α) ♯* α
by(*simp add*: *perm-compose eqvts*)
moreover from ‹$(bn(q \cdot \alpha))$ ♯* P'› **have** $(p \cdot bn(q \cdot \alpha))$ ♯* $(p \cdot P')$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹A_P ♯* α› ‹$(p \cdot A_P)$ ♯* α› ‹A_P ♯* P'› ‹$(p \cdot A_P)$ ♯* P'› *Sp* **have** *bn*((p · q) · α) ♯* P'
by(*simp add*: *perm-compose eqvts*)
moreover from ‹A_P' ♯* $(q \cdot \alpha)$› **have** $(p \cdot A_P')$ ♯* $(p \cdot q \cdot \alpha)$
by(*simp only*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with *Sp* ‹A_P ♯* α› ‹$(p \cdot A_P)$ ♯* α› **have** $(p \cdot A_P')$ ♯* $((p \cdot q) \cdot \alpha)$
by(*simp add*: *perm-compose*)
moreover from ‹A_P' ♯* V› **have** $(p \cdot A_P')$ ♯* $(p \cdot V)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹A_P ♯* V› ‹$(p \cdot A_P)$ ♯* V› *Sp* **have** $(p \cdot A_P')$ ♯* V **by** *simp*
moreover from ‹A_P' ♯* W› **have** $(p \cdot A_P')$ ♯* $(p \cdot W)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹A_P ♯* W› ‹$(p \cdot A_P)$ ♯* W› *Sp* **have** $(p \cdot A_P')$ ♯* W **by** *simp*
moreover from ‹A_P' ♯* X› **have** $(p \cdot A_P')$ ♯* $(p \cdot X)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹A_P ♯* X› ‹$(p \cdot A_P)$ ♯* X› *Sp* **have** $(p \cdot A_P')$ ♯* X **by** *simp*
moreover from ‹A_P' ♯* Y› **have** $(p \cdot A_P')$ ♯* $(p \cdot Y)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹A_P ♯* Y› ‹$(p \cdot A_P)$ ♯* Y› *Sp* **have** $(p \cdot A_P')$ ♯* Y **by** *simp*
moreover from ‹A_P' ♯* Z› **have** $(p \cdot A_P')$ ♯* $(p \cdot Z)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹A_P ♯* Z› ‹$(p \cdot A_P)$ ♯* Z› *Sp* **have** $(p \cdot A_P')$ ♯* Z **by** *simp*
moreover from ‹$(bn(q \cdot \alpha))$ ♯* V› **have** $(p \cdot bn(q \cdot \alpha))$ ♯* $(p \cdot V)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹A_P ♯* α› ‹$(p \cdot A_P)$ ♯* α› ‹A_P ♯* V› ‹$(p \cdot A_P)$ ♯* V› *Sp* **have** *bn*((p ·

$q) \cdot \ \alpha) \sharp* \ V$
by(*simp add*: *perm-compose eqvts*)
moreover from ‹$(bn(q \cdot \alpha)) \sharp* \ W$› **have** $(p \cdot bn(q \cdot \alpha)) \sharp* (p \cdot W)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$A_P \sharp* \alpha$› ‹$(p \cdot A_P) \sharp* \alpha$› ‹$A_P \sharp* W$› ‹$(p \cdot A_P) \sharp* W$› *Sp* **have** $bn((p \cdot q) \cdot \ \alpha) \sharp* W$
by(*simp add*: *perm-compose eqvts*)
moreover from ‹$(bn(q \cdot \alpha)) \sharp* X$› **have** $(p \cdot bn(q \cdot \alpha)) \sharp* (p \cdot X)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$A_P \sharp* \alpha$› ‹$(p \cdot A_P) \sharp* \alpha$› ‹$A_P \sharp* X$› ‹$(p \cdot A_P) \sharp* X$› *Sp* **have** $bn((p \cdot q) \cdot \ \alpha) \sharp* X$
by(*simp add*: *perm-compose eqvts*)
moreover from ‹$(bn(q \cdot \alpha)) \sharp* Y$› **have** $(p \cdot bn(q \cdot \alpha)) \sharp* (p \cdot Y)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$A_P \sharp* \alpha$› ‹$(p \cdot A_P) \sharp* \alpha$› ‹$A_P \sharp* Y$› ‹$(p \cdot A_P) \sharp* Y$› *Sp* **have** $bn((p \cdot q) \cdot \alpha) \sharp* Y$
by(*simp add*: *perm-compose eqvts*)
moreover from ‹$(bn(q \cdot \alpha)) \sharp* Z$› **have** $(p \cdot bn(q \cdot \alpha)) \sharp* (p \cdot Z)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$A_P \sharp* \alpha$› ‹$(p \cdot A_P) \sharp* \alpha$› ‹$A_P \sharp* Z$› ‹$(p \cdot A_P) \sharp* Z$› *Sp* **have** $bn((p \cdot q) \cdot \alpha) \sharp* Z$
by(*simp add*: *perm-compose eqvts*)
moreover from ‹*distinct* A_P'› **have** *distinct*$(p \cdot A_P')$ **by** *simp*
ultimately show *?case*
by(*elim cAlpha*)
next
case(*cInput* Ψ M K *xvec* N *Tvec* P C C' α P' V W X Y Z)
moreover obtain A_P Ψ_P **where** *extractFrame*$(P[xvec::=Tvec]) = \langle A_P, \Psi_P \rangle$ **and** *distinct* A_P
and $A_P \sharp* (C, P[xvec::=Tvec], \alpha, P', V, W, X, Y, Z, N)$
by(*rule freshFrame*)
moreover have $\mathbf{1} \otimes \Psi_P \simeq \Psi_P$
by(*blast intro*: *Identity Commutativity AssertionStatEqTrans*)
ultimately show *?case*
by(*intro cInput*) (*assumption* | *simp add*: *residualInject*)+
next
case(*cBrInput* Ψ M K *xvec* N *Tvec* P C C' α P' V W X Y Z)
moreover obtain A_P Ψ_P **where** *extractFrame*$(P[xvec::=Tvec]) = \langle A_P, \Psi_P \rangle$ **and** *distinct* A_P
and $A_P \sharp* (C, P[xvec::=Tvec], \alpha, P', V, W, X, Y, Z, N)$
by(*rule freshFrame*)
moreover have $\mathbf{1} \otimes \Psi_P \simeq \Psi_P$
by(*blast intro*: *Identity Commutativity AssertionStatEqTrans*)
ultimately show *?case*
by(*intro cBrInput*) (*assumption* | *simp add*: *residualInject*)+
next
case(*cOutput* Ψ M K N P C C' α P' V W X Y Z)
moreover obtain A_P Ψ_P **where** *extractFrame* $P = \langle A_P, \Psi_P \rangle$ **and** *distinct* A_P

and $A_P \sharp* (C, C', P, \alpha, N, P', V, W, X, Y, Z)$
by(*rule freshFrame*)
moreover have 1 $\otimes \Psi_P \simeq \Psi_P$
by(*blast intro*: *Identity Commutativity AssertionStatEqTrans*)
ultimately show *?case* **by**(*simp add*: *residualInject*)
next
case(*cBrOutput* Ψ *M K N P C C'* α *P' V W X Y Z*)
moreover obtain A_P Ψ_P **where** *extractFrame P* $= \langle A_P, \Psi_P \rangle$ **and** *distinct* A_P
and $A_P \sharp* (C, C', P, \alpha, N, P', V, W, X, Y, Z)$
by(*rule freshFrame*)
moreover have 1 $\otimes \Psi_P \simeq \Psi_P$
by(*blast intro*: *Identity Commutativity AssertionStatEqTrans*)
ultimately show *?case* **by**(*simp add*: *residualInject*)
next
case(*cCase* Ψ *P* φ *Cs* A_P Ψ_P *C C'* α *P' V W X Y Z*)
moreover from ‹*bn* $\alpha \sharp* (Cases\ Cs)$› ‹$(\varphi, P) \in set\ Cs$› **have** *bn* $\alpha \sharp* P$ **by**(*auto dest*: *memFreshChain*)
ultimately obtain *p* Ψ' A_P' Ψ_P' **where** *S*: *set p* $\subseteq set(bn\ \alpha) \times set(bn(p \cdot \alpha))$
and *FrP'*: *extractFrame P'* $= \langle A_P', \Psi_P' \rangle$
and *PeqP'*: $(p \cdot \Psi_P) \otimes \Psi' \simeq \Psi_P'$ **and** *distinct* A_P'
and $A_P' \sharp* C$ **and** $A_P' \sharp* P'$ **and** $A_P' \sharp* \alpha$ **and** $A_P' \sharp* (p \cdot \alpha)$
and $A_P' \sharp* V$ **and** $A_P' \sharp* W$ **and** $A_P' \sharp* X$ **and** $A_P' \sharp* Y$ **and** $A_P' \sharp* Z$
and *distinctPerm p* **and** $(bn(p \cdot \alpha)) \sharp* \alpha$ **and** $(bn(p \cdot \alpha)) \sharp* P'$
and $(bn(p \cdot \alpha)) \sharp* C'$ **and** $(bn(p \cdot \alpha)) \sharp* V$ **and** $(bn(p \cdot \alpha)) \sharp* W$ **and** $(bn(p \cdot \alpha)) \sharp* X$ **and** $(bn(p \cdot \alpha)) \sharp* Y$ **and** $(bn(p \cdot \alpha)) \sharp* Z$
apply −
apply (*rule cCase*)
apply (*assumption* | *simp* (*no-asm-use*))+
done
moreover from ‹$\Psi_P \simeq \mathbf{1}$› **have** $(p \cdot \Psi_P) \simeq (p \cdot \mathbf{1})$
by(*simp add*: *AssertionStatEqClosed*)
then have $(p \cdot \Psi_P) \simeq \mathbf{1}$ **by**(*simp add*: *permBottom*)
with *PeqP'* **have** $(\mathbf{1} \otimes \Psi') \simeq \Psi_P'$
by(*metis Identity AssertionStatEqTrans composition' Commutativity Associativity AssertionStatEqSym*)
ultimately show *?case* **using** *cCase* ‹*bn* $\alpha \sharp* P$›
by(*intro cCase*(*22*)) (*assumption* | *simp*)+
next
case(*cPar1* Ψ Ψ_Q *P* α *P'* A_Q *Q* A_P Ψ_P *C C'* α' *PQ' V W X Y Z*)
have *FrP*: *extractFrame P* $= \langle A_P, \Psi_P \rangle$ **and** *FrQ*: *extractFrame Q* $= \langle A_Q, \Psi_Q \rangle$
by *fact*+

note ‹*bn* α' $\sharp*$ *subject* α'›
moreover from ‹*bn* $\alpha' \sharp* (P \parallel Q)$› **have** *bn* $\alpha' \sharp* P$ **and** *bn* $\alpha' \sharp* Q$ **by** *simp*+
moreover with *FrP FrQ* ‹$A_P \sharp* \alpha'$› ‹$A_Q \sharp* \alpha'$› **have** *bn* $\alpha' \sharp* \Psi_P$ **and** *bn*

$\alpha' \sharp* \Psi_Q$
by(*force dest*: *extractFrameFreshChain*)+

moreover note ‹*bn* $\alpha' \sharp* V$› ‹*bn* $\alpha' \sharp* W$›
moreover from ‹*bn* $\alpha' \sharp* X$› ‹$A_Q \sharp* \alpha'$› **have** *bn* $\alpha' \sharp* (X@A_Q)$ **by** *simp*
moreover from ‹*bn* $\alpha' \sharp* Y$› ‹*bn* $\alpha' \sharp* \Psi_Q$› **have** *bn* $\alpha' \sharp* (\Psi_Q \# Y)$ **by** *simp*
moreover from ‹*bn* $\alpha' \sharp* Z$› ‹*bn* $\alpha' \sharp* Q$› **have** *bn* $\alpha' \sharp* (Q \# Z)$ **by** *simp*
moreover note ‹$A_P \sharp* V$› ‹$A_P \sharp* W$›
moreover from ‹$A_P \sharp* X$› ‹$A_P \sharp* A_Q$› **have** $A_P \sharp* (X@A_Q)$ **by** *simp*
moreover from ‹$A_P \sharp* Y$› ‹$A_P \sharp* \Psi_Q$› **have** $A_P \sharp* (\Psi_Q \# Y)$ **by** *force*
moreover from ‹$A_P \sharp* Z$› ‹$A_P \sharp* Q$› **have** $A_P \sharp* (Q \# Z)$ **by** *simp*
moreover from ‹$\alpha \prec (P' \parallel Q) = \alpha' \prec PQ'$› ‹*bn* $\alpha \sharp* Q$› ‹*bn* $\alpha' \sharp* Q$› ‹*bn* $\alpha \sharp* \alpha'$›
obtain P'' **where** *A*: $\alpha \prec P' = \alpha' \prec P''$ **and** $PQ' = P'' \parallel Q$
by(*metis actionPar1Dest'*)
moreover from ‹$A_P \sharp* PQ'$› ‹$PQ' = P'' \parallel Q$› **have** $A_P \sharp* P''$ **by** *simp*
ultimately obtain p Ψ' ${\Psi_P}'$ ${A_P}'$ **where** *S*: *set* $p \subseteq$ *set*(*bn* α') $\times$ *set* (*bn*($p \cdot \alpha'$)) **and** *PeqP'*: $((p \cdot \Psi_P) \otimes \Psi') \simeq {\Psi_P}'$
and *distinctPerm* p **and** $(bn(p \cdot \alpha')) \sharp* C'$ **and** *FrP'*: *extractFrame* $P'' = \langle {A_P}', {\Psi_P}' \rangle$
and ${A_P}' \sharp* P''$ **and** ${A_P}' \sharp* \alpha'$ ${A_P}' \sharp* (p \cdot \alpha')$ **and** ${A_P}' \sharp* C$
and $(bn(p \cdot \alpha')) \sharp* \alpha'$ **and** $(bn(p \cdot \alpha')) \sharp* P''$ **and** *distinct* ${A_P}'$
and ${A_P}' \sharp* V$ **and** ${A_P}' \sharp* W$ **and** ${A_P}' \sharp* (X \mathbin{@} A_Q)$ **and** ${A_P}' \sharp* (\Psi_Q \# Y)$
and ${A_P}' \sharp* (Q \# Z)$ **and** $(bn(p \cdot \alpha')) \sharp* V$ **and** $(bn(p \cdot \alpha')) \sharp* W$ **and** $(bn(p \cdot \alpha')) \sharp* (X \mathbin{@} A_Q)$ **and** $(bn(p \cdot \alpha')) \sharp* (\Psi_Q \# Y)$
and $(bn(p \cdot \alpha')) \sharp* (Q \# Z)$ **using** *cPar1*
by(*elim cPar1*)

then have ${A_P}' \sharp* Q$ **and** ${A_P}' \sharp* Z$ **and** ${A_P}' \sharp* A_Q$ **and** ${A_P}' \sharp* X$ **and** ${A_P}' \sharp* \Psi_Q$ **and** ${A_P}' \sharp* Y$
and $(bn(p \cdot \alpha')) \sharp* A_Q$ **and** $(bn(p \cdot \alpha')) \sharp* X$ **and** $(bn(p \cdot \alpha')) \sharp* Y$ **and** $(bn(p \cdot \alpha')) \sharp* Z$ **and** $(bn(p \cdot \alpha')) \sharp* \Psi_Q$
and $(bn(p \cdot \alpha')) \sharp* Q$
by(*simp del*: *freshChainSimps*)+

from ‹$A_Q \sharp* PQ'$› ‹$PQ' = P'' \parallel Q$› ‹${A_P}' \sharp* A_Q$› *FrP'* **have** $A_Q \sharp* {\Psi_P}'$
by(*force dest*: *extractFrameFreshChain*)
note *S*
moreover from *PeqP'* **have** $((p \cdot (\Psi_P \otimes \Psi_Q)) \otimes \Psi') \simeq {\Psi_P}' \otimes (p \cdot \Psi_Q)$
by(*simp add*: *eqvts*) (*metis Composition Associativity AssertionStatEqTrans AssertionStatEqSym Commutativity*)
with ‹$(bn(p \cdot \alpha')) \sharp* \Psi_Q$› ‹*bn* $\alpha' \sharp* \Psi_Q$› *S* **have** $((p \cdot (\Psi_P \otimes \Psi_Q)) \otimes \Psi') \simeq {\Psi_P}' \otimes \Psi_Q$
by *simp*
moreover from ‹$PQ' = P'' \parallel Q$› ‹${A_P}' \sharp* A_Q$› ‹${A_P}' \sharp* \Psi_Q$› ‹$A_Q \sharp* {\Psi_P}'$› ‹$A_Q \sharp* PQ'$› *FrP' FrQ* **have** *extractFrame* $PQ' = \langle ({A_P}'@A_Q), {\Psi_P}' \otimes \Psi_Q \rangle$
by *simp*
moreover note ‹*distinctPerm* p› ‹$(bn(p \cdot \alpha')) \sharp* C'$›
moreover from ‹${A_P}' \sharp* P''$› ‹${A_P}' \sharp* Q$› ‹$PQ' = P'' \parallel Q$› **have** ${A_P}' \sharp* PQ'$

by *simp*
moreover note ‹$A_Q \sharp* PQ'$› ‹$A_P' \sharp* \alpha'$› ‹$A_Q \sharp* \alpha'$› ‹$A_P' \sharp* C$› ‹$A_Q \sharp* C$› ‹$(bn(p \cdot \alpha')) \sharp* \alpha'$›
moreover from ‹$bn\ \alpha' \sharp* Q$› **have** $(bn(p \cdot \alpha')) \sharp* (p \cdot Q)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*] *bnEqvt*[*symmetric*])
with ‹$bn\ \alpha \sharp* Q$› ‹$(bn(p \cdot \alpha')) \sharp* Q$› S **have** $(bn(p \cdot \alpha')) \sharp* Q$ **by** *simp*
with ‹$(bn(p \cdot \alpha')) \sharp* P''$› ‹$PQ' = P'' \parallel Q$› **have** $(bn(p \cdot \alpha')) \sharp* PQ'$ **by** *simp*
moreover from ‹$A_P' \sharp* \alpha'$› ‹$A_Q \sharp* \alpha'$› **have** $(A_P'@A_Q) \sharp* \alpha'$ **by** *simp*
moreover from ‹$A_P' \sharp* C$› ‹$A_Q \sharp* C$› **have** $(A_P'@A_Q) \sharp* C$ **by** *simp*
moreover from ‹$A_P' \sharp* V$› ‹$A_Q \sharp* V$› **have** $(A_P'@A_Q) \sharp* V$ **by** *simp*
moreover from ‹$A_P' \sharp* W$› ‹$A_Q \sharp* W$› **have** $(A_P'@A_Q) \sharp* W$ **by** *simp*
moreover from ‹$A_P' \sharp* X$› ‹$A_Q \sharp* X$› **have** $(A_P'@A_Q) \sharp* X$ **by** *simp*
moreover from ‹$A_P' \sharp* Y$› ‹$A_Q \sharp* Y$› **have** $(A_P'@A_Q) \sharp* Y$ **by** *simp*
moreover from ‹$A_P' \sharp* Z$› ‹$A_Q \sharp* Z$› **have** $(A_P'@A_Q) \sharp* Z$ **by** *simp*
moreover from ‹$A_P' \sharp* PQ'$› ‹$A_Q \sharp* PQ'$› **have** $(A_P'@A_Q) \sharp* PQ'$ **by** *simp*
moreover from ‹$A_P' \sharp* \alpha'$› ‹$A_Q \sharp* \alpha'$› **have** $(A_P'@A_Q) \sharp* \alpha'$ **by** *simp*
moreover from ‹$A_Q \sharp* \alpha'$› **have** $(p \cdot A_Q) \sharp* (p \cdot \alpha')$
by(*simp only*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*] *bnEqvt*[*symmetric*])
with S ‹$A_Q \sharp* \alpha'$› ‹$bn(p \cdot \alpha') \sharp* A_Q$› **have** $A_Q \sharp* (p \cdot \alpha')$
by *simp*
with ‹$A_P' \sharp* (p \cdot \alpha')$› ‹$A_Q \sharp* \alpha'$› ‹$bn(p \cdot \alpha') \sharp* A_Q$› S **have** $(A_P'@A_Q) \sharp* (p \cdot \alpha')$
by *simp*
moreover from ‹$A_P' \sharp* A_Q$› ‹*distinct* A_P'› ‹*distinct* A_Q› **have** *distinct*$(A_P'@A_Q)$ **by** *auto*
moreover note ‹$(bn(p \cdot \alpha')) \sharp* V$› ‹$(bn(p \cdot \alpha')) \sharp* W$› ‹$(bn(p \cdot \alpha')) \sharp* X$› ‹$(bn(p \cdot \alpha')) \sharp* Y$› ‹$(bn(p \cdot \alpha')) \sharp* Z$›
ultimately show *?case* **using** *cPar1*
by *metis*
next
case(*cPar2* Ψ Ψ_P Q α Q' A_P P A_Q Ψ_Q C C' α' PQ' V W X Y Z)
have *FrP*: *extractFrame* $P = \langle A_P, \Psi_P \rangle$ **and** *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$
by *fact+*

note ‹$bn\ \alpha' \sharp* subject\ \alpha'$›
moreover from ‹$bn\ \alpha' \sharp* (P \parallel Q)$› **have** $bn\ \alpha' \sharp* Q$ **and** $bn\ \alpha' \sharp* P$ **by** *simp+*
moreover with *FrP FrQ* ‹$A_P \sharp* \alpha'$› ‹$A_Q \sharp* \alpha'$› **have** $bn\ \alpha' \sharp* \Psi_P$ **and** $bn\ \alpha' \sharp* \Psi_Q$
by(*force dest*: *extractFrameFreshChain*)+

moreover note ‹$bn\ \alpha' \sharp* V$› ‹$bn\ \alpha' \sharp* W$›
moreover from ‹$bn\ \alpha' \sharp* X$› ‹$A_P \sharp* \alpha'$› **have** $bn\ \alpha' \sharp* (X@A_P)$ **by** *simp*
moreover from ‹$bn\ \alpha' \sharp* Y$› ‹$bn\ \alpha' \sharp* \Psi_P$› **have** $bn\ \alpha' \sharp* (\Psi_P \# Y)$ **by** *simp*
moreover from ‹$bn\ \alpha' \sharp* Z$› ‹$bn\ \alpha' \sharp* P$› **have** $bn\ \alpha' \sharp* (P \# Z)$ **by** *simp*
moreover note ‹$A_Q \sharp* V$› ‹$A_Q \sharp* W$›
moreover from ‹$A_Q \sharp* X$› ‹$A_P \sharp* A_Q$› **have** $A_Q \sharp* (X@A_P)$ **by** *simp*
moreover from ‹$A_Q \sharp* Y$› ‹$A_Q \sharp* \Psi_P$› **have** $A_Q \sharp* (\Psi_P \# Y)$ **by** *force*

moreover from ‹$A_Q \sharp* Z$› ‹$A_Q \sharp* P$› **have** $A_Q \sharp* (P\#Z)$ **by** *simp*
moreover from ‹$\alpha \prec (P \parallel Q') = \alpha' \prec PQ'$› ‹$bn\ \alpha \sharp* P$› ‹$bn\ \alpha' \sharp* P$› ‹$bn\ \alpha \sharp* \alpha'$›
obtain Q'' **where** A: $\alpha \prec Q' = \alpha' \prec Q''$ **and** $PQ' = P \parallel Q''$
by(*metis actionPar2Dest'*)
moreover from ‹$A_Q \sharp* PQ'$› ‹$PQ' = P \parallel Q''$› **have** $A_Q \sharp* Q''$ **by** *simp*
ultimately obtain $p\ \Psi'\ A_Q'\ \Psi_Q'$ **where** S: *set* $p \subseteq$ *set*$(bn\ \alpha') \times$ *set* $(bn(p \cdot \alpha'))$ **and** *QeqQ'*: $((p \cdot \Psi_Q) \otimes \Psi') \simeq \Psi_Q'$
and *distinctPerm p* **and** $(bn(p \cdot \alpha')) \sharp* C'$ **and** *FrQ'*: *extractFrame* $Q'' = \langle A_Q', \Psi_Q' \rangle$
and $A_Q' \sharp* Q''$ **and** $A_Q' \sharp* \alpha'\ A_Q' \sharp* (p \cdot \alpha')$ **and** $A_Q' \sharp* C$
and $(bn(p \cdot \alpha')) \sharp* \alpha'$ **and** $(bn(p \cdot \alpha')) \sharp* Q''$ **and** *distinct* A_Q'
and $A_Q' \sharp* V$ **and** $A_Q' \sharp* W$ **and** $A_Q' \sharp* (X\ @\ A_P)$ **and** $A_Q' \sharp* (\Psi_P\#Y)$
and $A_Q' \sharp* (P\#Z)$ **and** $(bn(p \cdot \alpha')) \sharp* V$ **and** $(bn(p \cdot \alpha')) \sharp* W$ **and** $(bn(p \cdot \alpha')) \sharp* (X\ @\ A_P)$ **and** $(bn(p \cdot \alpha')) \sharp* (\Psi_P\#Y)$
and $(bn(p \cdot \alpha')) \sharp* (P\#Z)$ **using** *cPar2*
by(*elim cPar2*)

then have $A_Q' \sharp* P$ **and** $A_Q' \sharp* Z$ **and** $A_Q' \sharp* A_P$ **and** $A_Q' \sharp* X$ **and** $A_Q' \sharp* \Psi_P$ **and** $A_Q' \sharp* Y$
and $(bn(p \cdot \alpha')) \sharp* A_P$ **and** $(bn(p \cdot \alpha')) \sharp* X$ **and** $(bn(p \cdot \alpha')) \sharp* Y$ **and** $(bn(p \cdot \alpha')) \sharp* Z$ **and** $(bn(p \cdot \alpha')) \sharp* \Psi_P$
and $(bn(p \cdot \alpha')) \sharp* P$
by(*simp del*: *freshChainSimps*)+

from ‹$A_P \sharp* PQ'$› ‹$PQ' = P \parallel Q''$› ‹$A_Q' \sharp* A_P$› *FrQ'* **have** $A_P \sharp* \Psi_Q'$
by(*force dest*: *extractFrameFreshChain*)
note S
moreover from *QeqQ'* **have** $((p \cdot (\Psi_P \otimes \Psi_Q)) \otimes \Psi') \simeq (p \cdot \Psi_P) \otimes \Psi_Q'$
by(*simp add*: *eqvts*) (*metis Composition Associativity AssertionStatEqTrans AssertionStatEqSym Commutativity*)
with ‹$(bn(p \cdot \alpha')) \sharp* \Psi_P$› ‹$bn\ \alpha' \sharp* \Psi_P$› S **have** $((p \cdot (\Psi_P \otimes \Psi_Q)) \otimes \Psi') \simeq \Psi_P \otimes \Psi_Q'$
by *simp*
moreover from ‹$PQ' = P \parallel Q''$› ‹$A_Q' \sharp* A_P$› ‹$A_Q' \sharp* \Psi_P$› ‹$A_P \sharp* \Psi_Q'$› ‹$A_P \sharp* PQ'$› *FrQ' FrP* **have** *extractFrame* $PQ' = \langle (A_P @ A_Q'), \Psi_P \otimes \Psi_Q' \rangle$
by *simp*
moreover note ‹*distinctPerm p*› ‹$(bn(p \cdot \alpha')) \sharp* C'$›
moreover from ‹$A_Q' \sharp* Q''$› ‹$A_Q' \sharp* P$› ‹$PQ' = P \parallel Q''$› **have** $A_Q' \sharp* PQ'$ **by** *simp*
moreover note ‹$A_Q \sharp* PQ'$› ‹$A_Q' \sharp* \alpha'$› ‹$A_Q \sharp* \alpha'$› ‹$A_Q' \sharp* C$› ‹$A_Q \sharp* C$› ‹$(bn(p \cdot \alpha')) \sharp* \alpha'$›
moreover from ‹$bn\ \alpha' \sharp* Q$› **have** $(bn(p \cdot \alpha')) \sharp* (p \cdot Q)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*] *bnEqvt*[*symmetric*])
with ‹$bn\ \alpha \sharp* P$› ‹$(bn(p \cdot \alpha')) \sharp* P$› S **have** $(bn(p \cdot \alpha')) \sharp* P$ **by** *simp*
with ‹$(bn(p \cdot \alpha')) \sharp* Q''$› ‹$PQ' = P \parallel Q''$› **have** $(bn(p \cdot \alpha')) \sharp* PQ'$ **by** *simp*
moreover from ‹$A_Q' \sharp* \alpha'$› ‹$A_P \sharp* \alpha'$› **have** $(A_P @ A_Q') \sharp* \alpha'$ **by** *simp*
moreover from ‹$A_Q' \sharp* C$› ‹$A_P \sharp* C$› **have** $(A_P @ A_Q') \sharp* C$ **by** *simp*
moreover from ‹$A_Q' \sharp* V$› ‹$A_P \sharp* V$› **have** $(A_P @ A_Q') \sharp* V$ **by** *simp*

moreover from ‹$A_Q{}'$ ♯* W› ‹A_P ♯* W› **have** (A_P@$A_Q{}'$) ♯* W **by** *simp*
moreover from ‹$A_Q{}'$ ♯* X› ‹A_P ♯* X› **have** (A_P@$A_Q{}'$) ♯* X **by** *simp*
moreover from ‹$A_Q{}'$ ♯* Y› ‹A_P ♯* Y› **have** (A_P@$A_Q{}'$) ♯* Y **by** *simp*
moreover from ‹$A_Q{}'$ ♯* Z› ‹A_P ♯* Z› **have** (A_P@$A_Q{}'$) ♯* Z **by** *simp*
moreover from ‹$A_Q{}'$ ♯* PQ'› ‹A_P ♯* PQ'› **have** (A_P@$A_Q{}'$) ♯* PQ' **by** *simp*
moreover from ‹$A_Q{}'$ ♯* α'› ‹A_P ♯* α'› **have** (A_P@$A_Q{}'$) ♯* α' **by** *simp*
moreover from ‹A_P ♯* α'› **have** (p ∙ A_P) ♯* (p ∙ α')
by(*simp only*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*] *bnEqvt*[*symmetric*])
with S ‹A_P ♯* α'› ‹*bn*(p ∙ α') ♯* A_P› **have** A_P ♯* (p ∙ α')
by *simp*
with ‹$A_Q{}'$ ♯* (p ∙ α')› ‹A_P ♯* α'› ‹*bn*(p ∙ α') ♯* A_P› S **have** (A_P@$A_Q{}'$) ♯* (p ∙ α')
by *simp*
moreover note ‹(*bn*(p ∙ α')) ♯* V› ‹(*bn*(p ∙ α')) ♯* W› ‹(*bn*(p ∙ α')) ♯* X› ‹(*bn*(p ∙ α')) ♯* Y› ‹(*bn*(p ∙ α')) ♯* Z›
moreover from ‹$A_Q{}'$ ♯* A_P› ‹*distinct* A_P› ‹*distinct* $A_Q{}'$› **have** *distinct*(A_P@$A_Q{}'$) **by** *auto*
ultimately show *?case* **using** *cPar2*
by *metis*
next
case *cComm1*
then show *?case* **by**(*simp add*: *residualInject*)
next
case *cComm2*
then show *?case* **by**(*simp add*: *residualInject*)
next
case(*cBrMerge* Ψ Ψ_Q P M N P' A_P Ψ_P Q Q' A_Q C C' α PQ' V W X Y Z)
have *FrP*: *extractFrame* P = ⟨A_P, Ψ_P⟩ **and** *FrQ*: *extractFrame* Q = ⟨A_Q, Ψ_Q⟩
by *fact+*

have *bn* (¿M(⦇N⦈)) ♯* Q **by** *simp*
have *bn* (¿M(⦇N⦈)) ♯* Q' **by** *simp*
have *bn* (¿M(⦇N⦈)) ♯* α **by** *simp*
have *bn* (¿M(⦇N⦈)) ♯* P **by** *simp*
have *bn* (¿M(⦇N⦈)) ♯* P' **by** *simp*
have *bn* (¿M(⦇N⦈)) ♯* α **by** *simp*

from ‹¿M(⦇N⦈) ≺ P' ∥ Q' = α ≺ PQ'›
have *empty*α: *bn* α = ([]::*name list*) **and**
α = (¿M(⦇N⦈))
by(*simp add*: *residualInject*)+
then have *bn* α ♯* Q **and** *bn* α ♯* Q'
and *bn* α ♯* P **and** *bn* α ♯* P' **by** *simp+*

from ‹*bn* α = []›
have *bn* α ♯* *subject* α **and** *bn* α ♯* P **and** *bn* α ♯* Q
and *bn* α ♯* Ψ_P **and** *bn* α ♯* Ψ_Q **and** *bn* α ♯* C'
and *bn* α ♯* (X@A_Q)

and $bn\ \alpha\ \sharp*\ (\Psi_Q \# Y)$ **and** $bn\ \alpha\ \sharp*\ (Q \# Z)$ **by** *simp+*

moreover note $\langle A_P\ \sharp*\ P\rangle\ \langle A_P\ \sharp*\ \alpha\rangle\ \langle A_P\ \sharp*\ C\rangle\ \langle A_P\ \sharp*\ C'\rangle$
$\langle \alpha \neq \tau\rangle$

moreover from $\langle A_P\ \sharp*\ X\rangle\ \langle A_P\ \sharp*\ A_Q\rangle$ **have** $A_P\ \sharp*\ (X @ A_Q)$ **by** *simp*
moreover from $\langle A_P\ \sharp*\ Y\rangle\ \langle A_P\ \sharp*\ \Psi_Q\rangle$ **have** $A_P\ \sharp*\ (\Psi_Q \# Y)$ **by** *force*
moreover from $\langle A_P\ \sharp*\ Z\rangle\ \langle A_P\ \sharp*\ Q\rangle$ **have** $A_P\ \sharp*\ (Q \# Z)$ **by** *simp*
moreover from $\langle ¿M(\!|N|\!) \prec (P' \parallel Q') = \alpha \prec PQ'\rangle\ \langle bn\ (¿M(\!|N|\!))\ \sharp*\ Q'\rangle\ \langle bn\ \alpha\ \sharp*\ Q'\rangle\ \langle bn\ (¿M(\!|N|\!))\ \sharp*\ \alpha\rangle$
obtain P'' **where** *A*: $¿M(\!|N|\!) \prec P' = \alpha \prec P''$ **and** $PQ' = P'' \parallel Q'$
by(*metis actionPar1Dest'*)
moreover from $\langle A_P\ \sharp*\ PQ'\rangle\ \langle PQ' = P'' \parallel Q'\rangle$ **have** $A_P\ \sharp*\ P''$ **by** *simp*
ultimately obtain $p\ P\Psi'\ {A_P}'\ {\Psi_P}'$ **where** *Sp*: $set\ p \subseteq set(bn\ \alpha) \times set\ (bn(p \cdot \alpha))$ **and** *PeqP'*: $((p \cdot \Psi_P) \otimes P\Psi') \simeq {\Psi_P}'$
and *distinctPerm p* **and** *FrP'*: $extractFrame\ P'' = \langle {A_P}', {\Psi_P}'\rangle$
and ${A_P}'\ \sharp*\ P''$ **and** ${A_P}'\ \sharp*\ \alpha\ {A_P}'\ \sharp*\ (p \cdot \alpha)$ **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* ${A_P}'$
and ${A_P}'\ \sharp*\ V$ **and** ${A_P}'\ \sharp*\ W$ **and** ${A_P}'\ \sharp*\ (X @ A_Q)$ **and** ${A_P}'\ \sharp*\ (\Psi_Q \# Y)$
and ${A_P}'\ \sharp*\ (Q \# Z)$ **and** $(bn(p \cdot \alpha))\ \sharp*\ V$ **and** $(bn(p \cdot \alpha))\ \sharp*\ W$ **and** $(bn(p \cdot \alpha))\ \sharp*\ (X @ A_Q)$ **and** $(bn(p \cdot \alpha))\ \sharp*\ (\Psi_Q \# Y)$
and $(bn(p \cdot \alpha))\ \sharp*\ (Q \# Z)$ **using** *cBrMerge*
by(*elim cBrMerge*)

then have ${A_P}'\ \sharp*\ Q$ **and** ${A_P}'\ \sharp*\ Z$ **and** ${A_P}'\ \sharp*\ A_Q$ **and** ${A_P}'\ \sharp*\ X$ **and** ${A_P}'\ \sharp*\ \Psi_Q$ **and** ${A_P}'\ \sharp*\ Y$
and $(bn(p \cdot \alpha))\ \sharp*\ A_Q$ **and** $(bn(p \cdot \alpha))\ \sharp*\ X$ **and** $(bn(p \cdot \alpha))\ \sharp*\ Y$ **and** $(bn(p \cdot \alpha))\ \sharp*\ Z$ **and** $(bn(p \cdot \alpha))\ \sharp*\ \Psi_Q$
and $(bn(p \cdot \alpha))\ \sharp*\ Q$
by(*simp del*: *freshChainSimps*)+

from $\langle A_Q\ \sharp*\ PQ'\rangle\ \langle PQ' = (P'' \parallel Q')\rangle$ **have** $A_Q\ \sharp*\ P''$ **by** *simp*
with $\langle extractFrame\ P'' = \langle {A_P}', {\Psi_P}'\rangle\rangle\ \langle {A_P}'\ \sharp*\ A_Q\rangle$ **have** $A_Q\ \sharp*\ {\Psi_P}'$
by (*metis extractFrameFreshChain freshFrameDest*)

from $\langle bn\ \alpha = []\rangle$
have $bn\ \alpha\ \sharp*\ subject\ \alpha$ **and** $bn\ \alpha\ \sharp*\ Q$ **and** $bn\ \alpha\ \sharp*\ P''$
and $bn\ \alpha\ \sharp*\ \Psi_Q$ **and** $bn\ \alpha\ \sharp*\ {\Psi_P}'$ **and** $bn\ \alpha\ \sharp*\ C'$
and $bn\ \alpha\ \sharp*\ (X @ {A_P}')$
and $bn\ \alpha\ \sharp*\ ({\Psi_P}' \# Y)$ **and** $bn\ \alpha\ \sharp*\ (P'' \# Z)$ **by** *simp+*

moreover note $\langle {A_P}'\ \sharp*\ Q\rangle\ \langle A_Q\ \sharp*\ \alpha\rangle\ \langle A_Q\ \sharp*\ C\rangle\ \langle A_Q\ \sharp*\ C'\rangle$
$\langle \alpha \neq \tau\rangle$

moreover from $\langle A_Q\ \sharp*\ X\rangle\ \langle {A_P}'\ \sharp*\ A_Q\rangle$ **have** $A_Q\ \sharp*\ (X @ {A_P}')$ **by** *simp*
moreover from $\langle A_Q\ \sharp*\ Y\rangle\ \langle A_Q\ \sharp*\ {\Psi_P}'\rangle$ **have** $A_Q\ \sharp*\ ({\Psi_P}' \# Y)$ **by** *force*
moreover from $\langle A_Q\ \sharp*\ Z\rangle\ \langle A_Q\ \sharp*\ P''\rangle$ **have** $A_Q\ \sharp*\ (P'' \# Z)$ **by** *simp*
moreover from $\langle ¿M(\!|N|\!) \prec (P' \parallel Q') = \alpha \prec PQ'\rangle\ \langle bn\ (¿M(\!|N|\!))\ \sharp*\ P'\rangle\ \langle bn$

$\alpha \sharp* P' \rangle$ $\langle bn\ (¿M(\!|N|\!)) \sharp* \alpha \rangle$
obtain Q'' **where** A: $¿M(\!|N|\!) \prec Q' = \alpha \prec Q''$ **and** $PQ' = P' \parallel Q''$
by(*metis actionPar2Dest'*)

moreover from $\langle PQ' = P'' \parallel Q' \rangle$ $\langle PQ' = P' \parallel Q'' \rangle$
have $PQ' = P'' \parallel Q''$
by (*simp add*: *psi.inject*)

moreover from $\langle A_Q \sharp* PQ' \rangle$ $\langle PQ' = P'' \parallel Q'' \rangle$ **have** $A_Q \sharp* Q''$ **by** *simp*
ultimately obtain $q\ \Psi'\ {A_Q}'\ {\Psi_Q}'$ **where** *Sq*: *set* $q \subseteq$ *set*$(bn\ \alpha) \times$ *set* $(bn(q \cdot \alpha))$ **and** *QeqQ'*: $((q \cdot \Psi_Q) \otimes \Psi') \simeq {\Psi_Q}'$
and *distinctPerm q* **and** *FrQ'*: *extractFrame* $Q'' = \langle {A_Q}', {\Psi_Q}' \rangle$
and ${A_Q}' \sharp* Q''$ **and** ${A_Q}' \sharp* \alpha$ ${A_Q}' \sharp* (q \cdot \alpha)$ **and** ${A_Q}' \sharp* C$ **and** $(bn(q \cdot \alpha)) \sharp* C'$
and $(bn(q \cdot \alpha)) \sharp* \alpha$ **and** $(bn(q \cdot \alpha)) \sharp* Q''$ **and** *distinct* ${A_Q}'$
and ${A_Q}' \sharp* V$ **and** ${A_Q}' \sharp* W$ **and** ${A_Q}' \sharp* (X @ {A_P}')$ **and** ${A_Q}' \sharp* ({\Psi_P}' \# Y)$
and ${A_Q}' \sharp* (P'' \# Z)$ **and** $(bn(q \cdot \alpha)) \sharp* V$ **and** $(bn(q \cdot \alpha)) \sharp* W$ **and** $(bn(q \cdot \alpha)) \sharp* (X @ {A_P}')$ **and** $(bn(q \cdot \alpha)) \sharp* ({\Psi_P}' \# Y)$
and $(bn(q \cdot \alpha)) \sharp* (P'' \# Z)$ **using** *cBrMerge*
by(*elim cBrMerge*(*6*)[**where** *bb*=α]) (*rule refl* | *assumption*)+

then have ${A_Q}' \sharp* P''$ **and** ${A_Q}' \sharp* Z$ **and** ${A_Q}' \sharp* {A_P}'$ **and** ${A_Q}' \sharp* X$ **and** ${A_Q}' \sharp* {\Psi_P}'$ **and** ${A_Q}' \sharp* Y$
and $(bn(q \cdot \alpha)) \sharp* {A_P}'$ **and** $(bn(q \cdot \alpha)) \sharp* X$ **and** $(bn(q \cdot \alpha)) \sharp* Y$ **and** $(bn(q \cdot \alpha)) \sharp* Z$ **and** $(bn(q \cdot \alpha)) \sharp* {\Psi_P}'$
and $(bn(q \cdot \alpha)) \sharp* P''$
by(*simp del*: *freshChainSimps*)+

from *Sp Sq* $\langle bn\ \alpha = [] \rangle$ **have** $p = ([]::name\ prm)$ **and** $q = ([]::name\ prm)$ **and** $p = q$
by *simp*+

from $\langle {A_Q}' \sharp* P'' \rangle$ $\langle {A_Q}' \sharp* {A_P}' \rangle$ $\langle extractFrame\ P'' = \langle {A_P}', {\Psi_P}' \rangle \rangle$ **have** ${A_Q}' \sharp* {\Psi_P}'$
by (*metis extractFrameFreshChain freshFrameDest*)
from $\langle {A_P}' \sharp* \alpha \rangle$ $\langle \alpha = ¿M(\!|N|\!) \rangle$ **have** ${A_P}' \sharp* M$ **and** ${A_P}' \sharp* N$
by *simp*+

from $\langle \Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q' \rangle$ $\langle {A_P}' \sharp* Q \rangle$ $\langle {A_P}' \sharp* N \rangle$
have ${A_P}' \sharp* Q'$
by(*simp add*: *brinputFreshChainDerivative*)
with $\langle PQ' = (P'' \parallel Q') \rangle$ $\langle PQ' = (P'' \parallel Q'') \rangle$ **have** ${A_P}' \sharp* Q''$
by (*simp add*: *psi.inject*)
with *FrQ'* $\langle {A_Q}' \sharp* {A_P}' \rangle$ **have** ${A_P}' \sharp* {\Psi_Q}'$
by(*metis extractFrameFreshChain freshFrameDest*)
from $\langle {A_P}' \sharp* P'' \rangle$ $\langle {A_P}' \sharp* Q'' \rangle$ $\langle PQ' = (P'' \parallel Q'') \rangle$
have ${A_P}' \sharp* PQ'$ **by** *simp*
from $\langle {A_Q}' \sharp* P'' \rangle$ $\langle {A_Q}' \sharp* Q'' \rangle$ $\langle PQ' = (P'' \parallel Q'') \rangle$
have ${A_Q}' \sharp* PQ'$ **by** *simp*

from *PeqP'* **have** $((p \cdot (\Psi_P \otimes \Psi_Q)) \otimes P\Psi') \simeq \Psi_P{}' \otimes (p \cdot \Psi_Q)$
by(*simp add*: *eqvts*) (*metis Composition Associativity AssertionStatEqTrans AssertionStatEqSym Commutativity*)

with $\langle p = [] \rangle$ **have** $((\Psi_P \otimes \Psi_Q) \otimes P\Psi') \simeq (\Psi_P{}' \otimes \Psi_Q)$ **by** *simp*
with *QeqQ'* **have** $((q \cdot (\Psi_Q \otimes \Psi_P{}')) \otimes \Psi') \simeq \Psi_Q{}' \otimes (q \cdot \Psi_P{}')$
by(*simp add*: *eqvts*) (*metis Composition Associativity AssertionStatEqTrans AssertionStatEqSym Commutativity*)
with *PeqP'* $\langle p = [] \rangle$ $\langle q = [] \rangle$ **have** $((q \cdot (\Psi_Q \otimes ((p \cdot \Psi_P) \otimes P\Psi'))) \otimes \Psi') \simeq \Psi_Q{}' \otimes (q \cdot \Psi_P{}')$
by(*simp add*: *eqvts*) (*metis AssertionStatEqTrans Composition compositionSym*)
with $\langle p = [] \rangle$ **have** $((q \cdot (\Psi_Q \otimes (\Psi_P \otimes P\Psi'))) \otimes \Psi') \simeq \Psi_Q{}' \otimes (q \cdot \Psi_P{}')$ **by** *simp*
with $\langle q = [] \rangle$ **have** $(q \cdot (\Psi_P \otimes \Psi_Q)) \otimes (P\Psi' \otimes \Psi') \simeq \Psi_P{}' \otimes \Psi_Q{}'$
by(*simp add*: *eqvts*) (*metis AssertionStatEqTrans Commutativity Composition associativitySym*)

moreover note $\langle set\ q \subseteq set(bn\ \alpha) \times set\ (bn\ (q \cdot \alpha)) \rangle$

moreover from $\langle PQ' = P'' \parallel Q'' \rangle$ $\langle A_Q{}' \sharp* A_P{}' \rangle$ $\langle A_P{}' \sharp* \Psi_Q{}' \rangle$ $\langle A_Q{}' \sharp* \Psi_P{}' \rangle$ $\langle A_Q{}' \sharp* PQ' \rangle$ *FrP' FrQ'* **have** *extractFrame* $PQ' = \langle (A_P{}'@A_Q{}'), \Psi_P{}' \otimes \Psi_Q{}' \rangle$
by *simp*

moreover note $\langle distinctPerm\ q \rangle$

moreover from $\langle A_P{}' \sharp* PQ' \rangle$ $\langle A_Q{}' \sharp* PQ' \rangle$
have $(A_P{}'@A_Q{}') \sharp* PQ'$ **by** *simp*
moreover from $\langle A_P{}' \sharp* \alpha \rangle$ $\langle A_Q{}' \sharp* \alpha \rangle$
have $(A_P{}'@A_Q{}') \sharp* \alpha$ **by** *simp*
moreover with $\langle q = [] \rangle$
have $(A_P{}'@A_Q{}') \sharp* (q \cdot \alpha)$ **by** *simp*
moreover from $\langle A_P{}' \sharp* C \rangle$ $\langle A_Q{}' \sharp* C \rangle$
have $(A_P{}'@A_Q{}') \sharp* C$ **by** *simp*
moreover note $\langle bn\ (q \cdot \alpha) \sharp* C' \rangle$ $\langle bn\ (q \cdot \alpha) \sharp* \alpha \rangle$
moreover from $\langle bn\ \alpha = [] \rangle$
have $bn\ (q \cdot \alpha) \sharp* PQ'$
by (*metis Nominal.nil-eqvt bnEqvt freshSets*)
moreover from $\langle A_P{}' \sharp* V \rangle$ $\langle A_Q{}' \sharp* V \rangle$
have $(A_P{}'@A_Q{}') \sharp* V$ **by** *simp*
moreover from $\langle A_P{}' \sharp* W \rangle$ $\langle A_Q{}' \sharp* W \rangle$
have $(A_P{}'@A_Q{}') \sharp* W$ **by** *simp*
moreover from $\langle A_P{}' \sharp* X \rangle$ $\langle A_Q{}' \sharp* X \rangle$
have $(A_P{}'@A_Q{}') \sharp* X$ **by** *simp*
moreover from $\langle A_P{}' \sharp* Y \rangle$ $\langle A_Q{}' \sharp* Y \rangle$
have $(A_P{}'@A_Q{}') \sharp* Y$ **by** *simp*
moreover from $\langle A_P{}' \sharp* Z \rangle$ $\langle A_Q{}' \sharp* Z \rangle$

have $(A_P{}'@A_Q{}') \sharp* Z$ **by** *simp*
moreover from $\langle A_Q{}' \sharp* A_P{}' \rangle$ $\langle distinct\ A_P{}' \rangle$ $\langle distinct\ A_Q{}' \rangle$ **have** $distinct(A_P{}'@A_Q{}')$ **by** *simp*
moreover note $\langle (bn(q \cdot \alpha)) \sharp* V \rangle$ $\langle (bn(q \cdot \alpha)) \sharp* W \rangle$ $\langle (bn(q \cdot \alpha)) \sharp* X \rangle$ $\langle (bn(q \cdot \alpha)) \sharp* Y \rangle$ $\langle (bn(q \cdot \alpha)) \sharp* Z \rangle$
ultimately show *?case*
by(*intro cBrMerge*(*4 7*))
next
case(*cBrComm1* Ψ Ψ_Q P M N P' A_P Ψ_P Q *xvec* Q' A_Q C C' α PQ' V W X Y Z)
have *FrP*: *extractFrame* $P = \langle A_P, \Psi_P \rangle$ **and** *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$
by *fact+*

from $\langle xvec \sharp* \alpha \rangle$ **have** $xvec \sharp* bn\ \alpha$ **by** *simp*

from $\langle ¡M(\!|\nu*xvec|\!)\langle N \rangle \prec P' \parallel Q' = \alpha \prec PQ' \rangle$ **have** $\alpha \prec PQ' = ¡M(\!|\nu*xvec|\!)\langle N \rangle \prec P' \parallel Q'$ **by** *simp*
with $\langle xvec \sharp* (bn\ \alpha) \rangle$
obtain Q'' r **where** *rPerm*: *set* $r \subseteq$ *set* $(bn\ \alpha) \times$ *set xvec*
and $PQ' = (r \cdot P') \parallel Q''$ **and** $\alpha \prec Q'' = ¡M(\!|\nu*xvec|\!)\langle N \rangle \prec Q'$
by(*elim actionPar2Dest*) (*assumption* | *simp*)+
then have $¡M(\!|\nu*xvec|\!)\langle N \rangle \prec Q' = \alpha \prec Q''$ **by** *simp*

from $\langle A_Q \sharp* PQ' \rangle$ $\langle PQ' = (r \cdot P') \parallel Q'' \rangle$ **have** $A_Q \sharp* Q''$
by *simp*

from $\langle A_P \sharp* PQ' \rangle$ $\langle PQ' = (r \cdot P') \parallel Q'' \rangle$ **have** $A_P \sharp* Q''$
by *simp*

from $\langle bn\ \alpha \sharp* (P \parallel Q) \rangle$ $\langle A_P \sharp* \alpha \rangle$
have $bn\ \alpha \sharp* P$ **and** $A_P \sharp* bn\ \alpha$ **by** *simp+*
with $\langle extractFrame\ P = \langle A_P, \Psi_P \rangle \rangle$ **have** $bn\ \alpha \sharp* \Psi_P$
by (*metis extractFrameFreshChain freshFrameDest*)

from $\langle bn\ \alpha \sharp* (P \parallel Q) \rangle$ $\langle A_Q \sharp* \alpha \rangle$
have $bn\ \alpha \sharp* Q$ **and** $A_Q \sharp* bn\ \alpha$ **by** *simp+*
with $\langle extractFrame\ Q = \langle A_Q, \Psi_Q \rangle \rangle$ **have** $bn\ \alpha \sharp* \Psi_Q$
by (*metis extractFrameFreshChain freshFrameDest*)

from *rPerm* $\langle A_P \sharp* xvec \rangle$ $\langle xvec \sharp* \Psi_P \rangle$ $\langle A_P \sharp* bn\ \alpha \rangle$ $\langle bn\ \alpha \sharp* \Psi_P \rangle$
have $r \cdot A_P = A_P$ **and** $r \cdot \Psi_P = \Psi_P$ **by** *simp+*

have $¿M(\!|N|\!) \prec P' = ¿M(\!|N|\!) \prec P'$ **by** *simp*
moreover note $\langle A_P \sharp* P \rangle$ $\langle A_P \sharp* C \rangle$ $\langle A_P \sharp* C' \rangle$
moreover from $\langle A_P \sharp* M \rangle$ $\langle A_P \sharp* N \rangle$ **have** $A_P \sharp* (¿M(\!|N|\!))$ **by** *simp*
moreover note $\langle A_P \sharp* V \rangle$
moreover from $\langle A_P \sharp* W \rangle$ $\langle A_P \sharp* \alpha \rangle$ **have** $A_P \sharp* (\alpha \# W)$ **by** *simp*
moreover from $\langle A_P \sharp* X \rangle$ $\langle A_P \sharp* A_Q \rangle$ $\langle A_P \sharp* xvec \rangle$ **have** $A_P \sharp* (X@A_Q@xvec)$

by *simp*
moreover from ‹$A_P \sharp* Y$› ‹$A_P \sharp* \Psi_Q$› **have** $A_P \sharp* (\Psi_Q \# Y)$ **by** *force*
moreover from ‹$A_P \sharp* Z$› ‹$A_P \sharp* Q$› ‹$A_P \sharp* Q'$› ‹$A_P \sharp* Q''$› **have** $A_P \sharp* (Q\#(Q'\#(Q''\#Z)))$ **by** *simp*
moreover note ‹$A_P \sharp* P'$› ‹$A_P \sharp* Q$›

ultimately obtain p $P\Psi'$ $A_P{}'$ $\Psi_P{}'$ **where** *Sp*: *set* $p \subseteq set(bn\ (¿M(\!|N|\!)) \times set\ (bn(p \cdot (¿M(\!|N|\!))))$ **and** $((p \cdot \Psi_P) \otimes P\Psi') \simeq \Psi_P{}'$
and *distinctPerm* p **and** *FrP'*: *extractFrame* $P' = \langle A_P{}', \Psi_P{}' \rangle$
and $A_P{}' \sharp* P'$ **and** $A_P{}' \sharp* (¿M(\!|N|\!))$ $A_P{}' \sharp* (p \cdot (¿M(\!|N|\!)))$ **and** $A_P{}' \sharp* C$ **and** $(bn(p \cdot (¿M(\!|N|\!)))) \sharp* C'$
and $(bn(p \cdot (¿M(\!|N|\!)))) \sharp* (¿M(\!|N|\!))$ **and** $(bn(p \cdot (¿M(\!|N|\!)))) \sharp* P'$ **and** *distinct* $A_P{}'$
and $A_P{}' \sharp* V$ **and** $A_P{}' \sharp* (\alpha \# W)$ **and** $A_P{}' \sharp* (X\ @\ A_Q\ @\ xvec)$ **and** $A_P{}' \sharp* (\Psi_Q \# Y)$
and $A_P{}' \sharp* (Q\#(Q'\#(Q''\#Z)))$ **and** $(bn(p \cdot (¿M(\!|N|\!)))) \sharp* (\alpha \# W)$ **and** $(bn(p \cdot (¿M(\!|N|\!)))) \sharp* (X\ @\ A_Q\ @\ xvec)$ **and** $(bn(p \cdot (¿M(\!|N|\!)))) \sharp* (\Psi_Q \# Y)$
and $(bn(p \cdot (¿M(\!|N|\!)))) \sharp* (Q\#(Q'\#(Q''\#Z)))$
by(*elim cBrComm1*(*4*)) (*assumption* | *simp*)+

then have *PeqP'*: $\Psi_P \otimes P\Psi' \simeq \Psi_P{}'$
and $A_P{}' \sharp* P'$ **and** $A_P{}' \sharp* (¿M(\!|N|\!))$ **and** $A_P{}' \sharp* C$ **and** *distinct* $A_P{}'$
and $A_P{}' \sharp* Q$ **and** $A_P{}' \sharp* Q'$ **and** $A_P{}' \sharp* Q''$ **and** $A_P{}' \sharp* Z$ **and** $A_P{}' \sharp* A_Q$ **and** $A_P{}' \sharp* X$ **and** $A_P{}' \sharp* \Psi_Q$ **and** $A_P{}' \sharp* Y$
and $A_P{}' \sharp* xvec$ **and** $A_P{}' \sharp* \alpha$ **and** $A_P{}' \sharp* (subject\ \alpha)$ **and** $A_P{}' \sharp* (bn\ \alpha)$ **and** $A_P{}' \sharp* (object\ \alpha)$
and $A_P{}' \sharp* V$ **and** $A_P{}' \sharp* W$
by(*simp del*: *freshChainSimps*)+

from *rPerm* ‹$bn\ \alpha \sharp* \Psi_P$› ‹$xvec \sharp* \Psi_P$›
have $(r \cdot \Psi_P) = \Psi_P$ **by** *simp*

from *PeqP'* **have** $r \cdot (\Psi_P \otimes P\Psi' \simeq \Psi_P{}')$
by *simp*
with ‹$(r \cdot \Psi_P) = \Psi_P$›
have *rPeqP'*: $\Psi_P \otimes (r \cdot P\Psi') \simeq (r \cdot \Psi_P{}')$ **by**(*simp add*: *eqvts*)

from *rPerm* ‹$A_P{}' \sharp* (bn\ \alpha)$› ‹$A_P{}' \sharp* xvec$›
have $r \cdot A_P{}' = A_P{}'$ **by** *simp*

from *FrP'* **have** $r \cdot (extractFrame\ P' = \langle A_P{}', \Psi_P{}' \rangle)$
by *simp*

with ‹$r \cdot A_P{}' = A_P{}'$› **have** *rFrP'*: *extractFrame* $(r \cdot P') = \langle A_P{}', (r \cdot \Psi_P{}') \rangle$
by(*simp add*: *eqvts*)

from ‹$xvec \sharp* \alpha$› **have** $(bn\ \alpha) \sharp* xvec$ **by** *simp*

from ‹$(A_P\ @\ A_Q) \sharp* \alpha$› **have** $A_Q \sharp* bn\ \alpha$ **by** *simp*

from ‹A_Q ♯* M› ‹A_Q ♯* $xvec$› ‹A_Q ♯* N› **have** A_Q ♯* (¡M(⦇ν*$xvec$⦈)⟨N⟩) **by** *simp*

from ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto$ ¡M(⦇ν*$xvec$⦈)⟨N⟩ ≺ Q'› ‹$xvec$ ♯* M› ‹*distinct* $xvec$› ‹$bn\ \alpha$ ♯* Q› ‹$xvec$ ♯* $bn\ \alpha$›
have $bn\ \alpha$ ♯* Q' **by**(*simp add*: *broutputFreshChainDerivative*)

from ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto$ ¡M(⦇ν*$xvec$⦈)⟨N⟩ ≺ Q'› ‹$xvec$ ♯* M› ‹*distinct* $xvec$› ‹$bn\ \alpha$ ♯* Q› ‹$bn\ \alpha$ ♯* $xvec$›
have $bn\ \alpha$ ♯* N **by**(*simp add*: *broutputFreshChainDerivative*)

note ‹$bn\ \alpha$ ♯* *subject* α›

moreover from ‹$bn\ \alpha$ ♯* ($P \parallel Q$)› **have** $bn\ \alpha$ ♯* Q **and** $bn\ \alpha$ ♯* P **by** *simp*+
moreover from ‹$bn\ \alpha$ ♯* V› ‹$bn\ \alpha$ ♯* N› **have** $bn\ \alpha$ ♯* (N#V) **by** *simp*
moreover note ‹$bn\ \alpha$ ♯* Ψ_P› ‹$bn\ \alpha$ ♯* Ψ_Q› ‹$bn\ \alpha$ ♯* W›
moreover from ‹$bn\ \alpha$ ♯* X› ‹A_P ♯* $bn\ \alpha$› ‹$A_P{}'$ ♯* $bn\ \alpha$› ‹$bn\ \alpha$ ♯* $xvec$› **have** $bn\ \alpha$ ♯* (X@$xvec$@A_P@$A_P{}'$) **by** *simp*
moreover from ‹$bn\ \alpha$ ♯* Y› ‹$bn\ \alpha$ ♯* Ψ_P› **have** $bn\ \alpha$ ♯* (Ψ_P#Y) **by** *simp*
moreover from ‹$bn\ \alpha$ ♯* Z› ‹$bn\ \alpha$ ♯* P› ‹$bn\ \alpha$ ♯* Q› **have** $bn\ \alpha$ ♯* (P#Q#Z) **by** *simp*
moreover from ‹A_Q ♯* V› ‹A_Q ♯* N› **have** A_Q ♯* (N#V) **by** *simp*
moreover note ‹A_Q ♯* W›
moreover from ‹A_Q ♯* X› ‹A_P ♯* A_Q› ‹$A_P{}'$ ♯* A_Q› ‹A_Q ♯* $xvec$› **have** A_Q ♯* (X@$xvec$@A_P@$A_P{}'$) **by** *simp*
moreover from ‹A_Q ♯* Y› ‹A_Q ♯* Ψ_P› **have** A_Q ♯* (Ψ_P#Y) **by** *force*
moreover from ‹A_Q ♯* Z› ‹A_Q ♯* P› ‹A_Q ♯* Q› **have** A_Q ♯* (P#Q#Z) **by** *simp*

moreover note ‹¡M(⦇ν*$xvec$⦈)⟨N⟩ ≺ Q' = α ≺ Q''›
ultimately obtain q $Q\Psi'$ $A_Q{}'$ $\Psi_Q{}'$ **where** *Sq*: *set* $q \subseteq$ *set* ($bn\ \alpha$) × *set* (bn ($q \cdot \alpha$)) **and** *QeqQ'*: (($q \cdot \Psi_Q$) ⊗ $Q\Psi'$) ≃ $\Psi_Q{}'$
and *distinctPerm* q **and** $bn(q \cdot \alpha)$ ♯* C' **and** *FrQ'*: *extractFrame* Q'' = ⟨$A_Q{}'$, $\Psi_Q{}'$⟩
and $A_Q{}'$ ♯* Q'' **and** $A_Q{}'$ ♯* α **and** $A_Q{}'$ ♯* ($q \cdot \alpha$) **and** $A_Q{}'$ ♯* C
and $bn(q \cdot \alpha)$ ♯* α **and** $bn(q \cdot \alpha)$ ♯* Q'' **and** *distinct* $A_Q{}'$
and $A_Q{}'$ ♯* (N#V) **and** $A_Q{}'$ ♯* W **and** $A_Q{}'$ ♯* (X @ $xvec$ @ A_P @ $A_P{}'$) **and** $A_Q{}'$ ♯* (Ψ_P#Y)
and $A_Q{}'$ ♯* (P#Q#Z) **and** $bn(q \cdot \alpha)$ ♯* (N#V) **and** $bn(q \cdot \alpha)$ ♯* W **and** $bn(q \cdot \alpha)$ ♯* (X @ $xvec$ @ A_P @ $A_P{}'$) **and** $bn(q \cdot \alpha)$ ♯* (Ψ_P#Y)
and $bn(q \cdot \alpha)$ ♯* (P#Q#Z)
using ‹A_Q ♯* Q› ‹A_Q ♯* α› ‹A_Q ♯* C› ‹A_Q ♯* C'›
‹$bn\ \alpha$ ♯* C'› ‹$\alpha \neq \tau$› ‹A_Q ♯* Q''› ‹*distinct* ($bn\ \alpha$)›
by(*elim cBrComm1*(*8*)[**where** *b*=C **and** *ba*=C' **and** *bb*=α **and** *bc*=Q'' **and** *bf*=(X @ $xvec$ @ A_P @ $A_P{}'$) **and** *bg*=(Ψ_P#Y) **and** *bh*=(P#Q#Z)]) (*assumption* | *simp*)+
then have $A_Q{}'$ ♯* P **and** $A_Q{}'$ ♯* Z **and** $A_Q{}'$ ♯* A_P **and** $A_Q{}'$ ♯* $A_P{}'$ **and** $A_Q{}'$ ♯* X **and** $A_Q{}'$ ♯* Ψ_P **and** $A_Q{}'$ ♯* Y **and** $A_Q{}'$ ♯* Q **and** $A_Q{}'$ ♯* N
and $A_Q{}'$ ♯* $xvec$ **and** $A_Q{}'$ ♯* V **and** $A_Q{}'$ ♯* W

and $bn(q \cdot \alpha) \sharp* A_P$ **and** $bn(q \cdot \alpha) \sharp* A_P{}'$ **and** $bn(q \cdot \alpha) \sharp* X$ **and** $bn(q \cdot \alpha) \sharp* Y$ **and** $bn(q \cdot \alpha) \sharp* Z$ **and** $bn(q \cdot \alpha) \sharp* \Psi_P$
and $bn(q \cdot \alpha) \sharp* P$ **and** $bn(q \cdot \alpha) \sharp* W$ **and** $bn(q \cdot \alpha) \sharp* V$ **and** $bn(q \cdot \alpha) \sharp* N$ **and** $bn(q \cdot \alpha) \sharp* xvec$
by(*simp del*: *freshChainSimps*)+

from $\langle A_P{}' \sharp* Q'' \rangle$ $\langle A_Q{}' \sharp* A_P{}' \rangle$ *FrQ'*
have $A_P{}' \sharp* \Psi_Q{}'$
by(*metis extractFrameFreshChain freshFrameDest*)

from $\langle \Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P' \rangle$ $\langle A_Q{}' \sharp* P \rangle$ $\langle A_Q{}' \sharp* N \rangle$
have $A_Q{}' \sharp* P'$
by(*simp add*: *brinputFreshChainDerivative*)
with *FrP'* $\langle A_Q{}' \sharp* A_P{}' \rangle$ **have** $A_Q{}' \sharp* \Psi_P{}'$
by(*metis extractFrameFreshChain freshFrameDest*)

have $(\Psi_P \otimes (r \cdot P\Psi')) \otimes ((q \cdot \Psi_Q) \otimes Q\Psi') \simeq (r \cdot \Psi_P{}') \otimes \Psi_Q{}'$
by(*metis Composition' rPeqP' QeqQ'*)
then have $(\Psi_P \otimes ((r \cdot P\Psi') \otimes ((q \cdot \Psi_Q) \otimes Q\Psi'))) \simeq (r \cdot \Psi_P{}') \otimes \Psi_Q{}'$
by (*metis AssertionStatEqSym AssertionStatEqTrans Associativity*)
then have $(\Psi_P \otimes (((q \cdot \Psi_Q) \otimes Q\Psi') \otimes (r \cdot P\Psi'))) \simeq (r \cdot \Psi_P{}') \otimes \Psi_Q{}'$
by (*metis AssertionStatEqSym AssertionStatEqTrans Associativity associativitySym*)
then have $(\Psi_P \otimes ((q \cdot \Psi_Q) \otimes (Q\Psi' \otimes (r \cdot P\Psi')))) \simeq (r \cdot \Psi_P{}') \otimes \Psi_Q{}'$
by (*metis AssertionStatEqSym AssertionStatEqTrans Associativity compositionSym*)
then have $((\Psi_P \otimes (q \cdot \Psi_Q)) \otimes (Q\Psi' \otimes (r \cdot P\Psi'))) \simeq (r \cdot \Psi_P{}') \otimes \Psi_Q{}'$
by (*metis AssertionStatEqTrans Associativity*)
then have $((\Psi_P \otimes (q \cdot \Psi_Q)) \otimes ((r \cdot P\Psi') \otimes Q\Psi')) \simeq (r \cdot \Psi_P{}') \otimes \Psi_Q{}'$
by (*metis AssertionStatEqSym AssertionStatEqTrans Associativity associativitySym*)
with *Sq* $\langle bn\ \alpha \sharp* \Psi_P \rangle$ $\langle bn(q \cdot \alpha) \sharp* \Psi_P \rangle$ **have** $((q \cdot (\Psi_P \otimes \Psi_Q)) \otimes ((r \cdot P\Psi') \otimes Q\Psi')) \simeq (r \cdot \Psi_P{}') \otimes \Psi_Q{}'$
by(*simp add*: *eqvts*)

from *Sq* $\langle A_P{}' \sharp* \alpha \rangle$ $\langle bn(q \cdot \alpha) \sharp* A_P{}' \rangle$ **have** $A_P{}' \sharp* (q \cdot \alpha)$
by (*metis actionFreshChain freshChainSym freshStarChainSimps fresh-star-set-eq*)

from $\langle A_Q{}' \sharp* \alpha \rangle$ **have** $A_Q{}' \sharp* bn\ \alpha$ **by** *simp*
from *rPerm* $\langle A_P{}' \sharp* P' \rangle$ $\langle A_P{}' \sharp* xvec \rangle$ $\langle A_P{}' \sharp* bn\ \alpha \rangle$
have $A_P{}' \sharp* (r \cdot P')$
by (*metis freshAlphaPerm freshChainSym name-list-set-fresh permStarFresh*)
from *rPerm* $\langle A_Q{}' \sharp* P' \rangle$ $\langle A_Q{}' \sharp* xvec \rangle$ $\langle A_Q{}' \sharp* bn\ \alpha \rangle$
have $A_Q{}' \sharp* (r \cdot P')$
by (*metis freshAlphaPerm freshChainSym name-list-set-fresh permStarFresh*)
from *rPerm* $\langle A_Q{}' \sharp* \Psi_P{}' \rangle$ $\langle A_Q{}' \sharp* xvec \rangle$ $\langle A_Q{}' \sharp* bn\ \alpha \rangle$
have $A_Q{}' \sharp* (r \cdot \Psi_P{}')$
by (*metis freshAlphaPerm freshChainSym name-list-set-fresh permStarFresh*)

with ‹$A_P{}' \sharp* \Psi_Q{}'$› ‹$A_Q{}' \sharp* A_P{}'$› *rFrP' FrQ'*
have *extractFrame* $((r \cdot P') \parallel Q'') = \langle (A_P{}'@A_Q{}'), ((r \cdot \Psi_P{}') \otimes \Psi_Q{}') \rangle$
by *simp*
with ‹$PQ' = ((r \cdot P') \parallel Q'')$›
have *extractFrame* $PQ' = \langle (A_P{}'@A_Q{}'), ((r \cdot \Psi_P{}') \otimes \Psi_Q{}') \rangle$
by *simp*

from ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$› ‹$bn(q \cdot \alpha) \sharp* P$› ‹$bn(q \cdot \alpha) \sharp* N$›
have $bn(q \cdot \alpha) \sharp* P'$ **by**(*simp add*: *brinputFreshChainDerivative*)

with *rPerm* ‹$bn(q \cdot \alpha) \sharp* \alpha$› ‹$bn(q \cdot \alpha) \sharp* xvec$›
have $bn(q \cdot \alpha) \sharp* (r \cdot P')$
by (*metis actionFreshChain freshAlphaPerm freshChainSym name-list-set-fresh permStarFresh*)

from ‹$A_P{}' \sharp* Q''$› ‹$A_P{}' \sharp* (r \cdot P')$› ‹$PQ' = (r \cdot P') \parallel Q''$› **have** $A_P{}' \sharp* PQ'$ **by** *simp*
from ‹$A_Q{}' \sharp* Q''$› ‹$A_Q{}' \sharp* (r \cdot P')$› ‹$PQ' = (r \cdot P') \parallel Q''$› **have** $A_Q{}' \sharp* PQ'$ **by** *simp*

note ‹*set* $q \subseteq (set\ (bn\ \alpha)) \times set\ (bn(q \cdot \alpha))$›
‹$((q \cdot (\Psi_P \otimes \Psi_Q)) \otimes ((r \cdot P\Psi') \otimes Q\Psi')) \simeq (r \cdot \Psi_P{}') \otimes \Psi_Q{}'$› ‹*distinctPerm q*›
‹*extractFrame* $PQ' = \langle (A_P{}'@A_Q{}'), ((r \cdot \Psi_P{}') \otimes \Psi_Q{}') \rangle$›

moreover from ‹$A_P{}' \sharp* PQ'$› ‹$A_Q{}' \sharp* PQ'$› **have** $(A_P{}'@A_Q{}') \sharp* PQ'$ **by** *simp*
moreover from ‹$A_P{}' \sharp* \alpha$› ‹$A_Q{}' \sharp* \alpha$› **have** $(A_P{}'@A_Q{}') \sharp* \alpha$ **by** *simp*
moreover from ‹$A_P{}' \sharp* (q \cdot \alpha)$› ‹$A_Q{}' \sharp* (q \cdot \alpha)$› **have** $(A_P{}'@A_Q{}') \sharp* (q \cdot \alpha)$ **by** *simp*
moreover from ‹$A_P{}' \sharp* C$› ‹$A_Q{}' \sharp* C$› **have** $(A_P{}'@A_Q{}') \sharp* C$ **by** *simp*
moreover note ‹$bn(q \cdot \alpha) \sharp* C'$› ‹$bn(q \cdot \alpha) \sharp* \alpha$›
moreover from ‹$bn(q \cdot \alpha) \sharp* (r \cdot P')$› ‹$bn(q \cdot \alpha) \sharp* Q''$› ‹$PQ' = (r \cdot P') \parallel Q''$›
have $bn(q \cdot \alpha) \sharp* PQ'$ **by** *simp*
moreover from ‹$A_P{}' \sharp* V$› ‹$A_Q{}' \sharp* V$› **have** $(A_P{}'@A_Q{}') \sharp* V$ **by** *simp*
moreover from ‹$A_P{}' \sharp* W$› ‹$A_Q{}' \sharp* W$› **have** $(A_P{}'@A_Q{}') \sharp* W$ **by** *simp*
moreover from ‹$A_P{}' \sharp* X$› ‹$A_Q{}' \sharp* X$› **have** $(A_P{}'@A_Q{}') \sharp* X$ **by** *simp*
moreover from ‹$A_P{}' \sharp* Y$› ‹$A_Q{}' \sharp* Y$› **have** $(A_P{}'@A_Q{}') \sharp* Y$ **by** *simp*
moreover from ‹$A_P{}' \sharp* Z$› ‹$A_Q{}' \sharp* Z$› **have** $(A_P{}'@A_Q{}') \sharp* Z$ **by** *simp*
moreover from ‹*distinct* $A_P{}'$› ‹*distinct* $A_Q{}'$› ‹$A_Q{}' \sharp* A_P{}'$›
have *distinct*$(A_P{}'@A_Q{}')$ **by** *simp*
moreover note ‹$bn(q \cdot \alpha) \sharp* V$› ‹$bn(q \cdot \alpha) \sharp* W$›
‹$bn(q \cdot \alpha) \sharp* X$› ‹$bn(q \cdot \alpha) \sharp* Y$› ‹$bn(q \cdot \alpha) \sharp* Z$›
ultimately show *?case*
by(*rule cBrComm1*(*63*))
next
case(*cBrComm2* $\Psi\ \Psi_Q\ P\ M\ xvec\ N\ P'\ A_P\ \Psi_P\ Q\ Q'\ A_Q\ C\ C'\ \alpha\ PQ'\ V\ W\ X\ Y\ Z$)

have *FrP*: *extractFrame* $P = \langle A_P, \Psi_P \rangle$ **and** *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$
by *fact+*

from ‹*xvec* ♯* α› **have** *xvec* ♯* *bn* α **by** *simp*

from ‹¡M⦇ν*$xvec$⦈⟨N⟩ ≺ P' ∥ Q' = α ≺ PQ'› **have** α ≺ PQ' = ¡M⦇ν*$xvec$⦈⟨N⟩ ≺ P' ∥ Q' **by** *simp*
with ‹*xvec* ♯* (*bn* α)›
obtain P'' r **where** *rPerm*: *set* r ⊆ *set* (*bn* α) × *set xvec*
and $PQ' = P'' \parallel (r \cdot Q')$ **and** $\alpha \prec P'' =$ ¡M⦇ν*$xvec$⦈⟨N⟩ ≺ P'
by(*elim actionPar1Dest*) (*assumption* | *simp*)+
then have ¡M⦇ν*$xvec$⦈⟨N⟩ ≺ P' = α ≺ P'' **by** *simp*

from ‹A_Q ♯* PQ'› ‹$PQ' = P'' \parallel (r \cdot Q')$› **have** A_Q ♯* P''
by *simp*

from ‹A_P ♯* PQ'› ‹$PQ' = P'' \parallel (r \cdot Q')$› **have** A_P ♯* P''
by *simp*

from ‹*bn* α ♯* ($P \parallel Q$)› ‹A_P ♯* α›
have *bn* α ♯* P **and** A_P ♯* *bn* α **by** *simp+*
with ‹*extractFrame* $P = \langle A_P, \Psi_P \rangle$› **have** *bn* α ♯* Ψ_P
by (*metis extractFrameFreshChain freshFrameDest*)

from ‹*bn* α ♯* ($P \parallel Q$)› ‹A_Q ♯* α›
have *bn* α ♯* Q **and** A_Q ♯* *bn* α **by** *simp+*
with ‹*extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$› **have** *bn* α ♯* Ψ_Q
by (*metis extractFrameFreshChain freshFrameDest*)

from *rPerm* ‹A_Q ♯* *xvec*› ‹*xvec* ♯* Ψ_Q› ‹A_Q ♯* *bn* α› ‹*bn* α ♯* Ψ_Q›
have $r \cdot A_Q = A_Q$ **and** $r \cdot \Psi_Q = \Psi_Q$ **by** *simp+*

have ¿M⦇N⦈ ≺ Q' = ¿M⦇N⦈ ≺ Q' **by** *simp*
moreover note ‹A_Q ♯* P› ‹A_Q ♯* C› ‹A_Q ♯* C'›
moreover from ‹A_Q ♯* M› ‹A_Q ♯* N› **have** A_Q ♯* (¿M⦇N⦈) **by** *simp*
moreover note ‹A_Q ♯* V›
moreover from ‹A_Q ♯* W› ‹A_Q ♯* α› **have** A_Q ♯* (α#W) **by** *simp*
moreover from ‹A_Q ♯* X› ‹A_P ♯* A_Q› ‹A_Q ♯* *xvec*› **have** A_Q ♯* (X@A_P@*xvec*) **by** *simp*
moreover from ‹A_Q ♯* Y› ‹A_Q ♯* Ψ_P› **have** A_Q ♯* (Ψ_P#Y) **by** *force*
moreover from ‹A_Q ♯* Z› ‹A_Q ♯* P› ‹A_Q ♯* P'› ‹A_Q ♯* P''› **have** A_Q ♯* (P#(P'#(P''#Z))) **by** *simp*
moreover note ‹A_Q ♯* Q'› ‹A_Q ♯* Q›

ultimately obtain q $Q\Psi'$ A_Q' Ψ_Q' **where** *Sq*: *set* q ⊆ *set*(*bn* (¿M⦇N⦈)) × *set* (*bn*(q · (¿M⦇N⦈))) **and** ((q · Ψ_Q) ⊗ $Q\Psi'$) ≃ Ψ_Q'
and *distinctPerm* q **and** *FrQ'*: *extractFrame* $Q' = \langle A_Q', \Psi_Q' \rangle$
and A_Q' ♯* Q' **and** A_Q' ♯* (¿M⦇N⦈) A_Q' ♯* (q · (¿M⦇N⦈)) **and** A_Q' ♯* C

and $(bn(q \cdot (¿M(\!|N|\!))))\ \sharp*\ C'$
and $(bn(q \cdot (¿M(\!|N|\!))))\ \sharp*\ (¿M(\!|N|\!))$ **and** $(bn(q \cdot (¿M(\!|N|\!))))\ \sharp*\ Q'$ **and** *distinct* $A_Q{}'$
and $A_Q{}'\ \sharp*\ V$ **and** $A_Q{}'\ \sharp*\ (\alpha \# W)$ **and** $A_Q{}'\ \sharp*\ (X\ @\ A_P\ @\ xvec)$ **and** $A_Q{}'\ \sharp*\ (\Psi_P \# Y)$
and $A_Q{}'\ \sharp*\ (P\#(P'\#(P''\#Z)))$ **and** $(bn(q \cdot (¿M(\!|N|\!))))\ \sharp*\ (\alpha \# W)$ **and** $(bn(q \cdot (¿M(\!|N|\!))))\ \sharp*\ (X\ @\ A_P\ @\ xvec)$ **and** $(bn(q \cdot (¿M(\!|N|\!))))\ \sharp*\ (\Psi_P \# Y)$
and $(bn(q \cdot (¿M(\!|N|\!))))\ \sharp*\ (P\#(P'\#(P''\#Z)))$
by(*elim cBrComm2*(*8*)) (*assumption* | *simp*)+

then have *QeqQ'*: $\Psi_Q \otimes Q\Psi' \simeq \Psi_Q{}'$
and $A_Q{}'\ \sharp*\ Q'$ **and** $A_Q{}'\ \sharp*\ (¿M(\!|N|\!))$ **and** $A_Q{}'\ \sharp*\ C$ **and** *distinct* $A_Q{}'$
and $A_Q{}'\ \sharp*\ P$ **and** $A_Q{}'\ \sharp*\ P'$ **and** $A_Q{}'\ \sharp*\ P''$ **and** $A_Q{}'\ \sharp*\ Z$ **and** $A_Q{}'\ \sharp*\ A_P$ **and** $A_Q{}'\ \sharp*\ X$ **and** $A_Q{}'\ \sharp*\ \Psi_P$ **and** $A_Q{}'\ \sharp*\ Y$
and $A_Q{}'\ \sharp*\ xvec$ **and** $A_Q{}'\ \sharp*\ \alpha$ **and** $A_Q{}'\ \sharp*\ (subject\ \alpha)$ **and** $A_Q{}'\ \sharp*\ (bn\ \alpha)$ **and** $A_Q{}'\ \sharp*\ (object\ \alpha)$
and $A_Q{}'\ \sharp*\ V$ **and** $A_Q{}'\ \sharp*\ W$
by(*simp del*: *freshChainSimps*)+

from *rPerm* ‹$bn\ \alpha\ \sharp*\ \Psi_Q$› ‹$xvec\ \sharp*\ \Psi_Q$›
have $(r \cdot \Psi_Q) = \Psi_Q$ **by** *simp*

from *QeqQ'* **have** $r \cdot (\Psi_Q \otimes Q\Psi' \simeq \Psi_Q{}')$
by *simp*
with ‹$(r \cdot \Psi_Q) = \Psi_Q$›
have *rQeqQ'*: $\Psi_Q \otimes (r \cdot Q\Psi') \simeq (r \cdot \Psi_Q{}')$ **by**(*simp add*: *eqvts*)

from *rPerm* ‹$A_Q{}'\ \sharp*\ (bn\ \alpha)$› ‹$A_Q{}'\ \sharp*\ xvec$›
have $r \cdot A_Q{}' = A_Q{}'$ **by** *simp*

from *FrQ'* **have** $r \cdot (extractFrame\ Q' = \langle A_Q{}', \Psi_Q{}'\rangle)$
by *simp*

with ‹$r \cdot A_Q{}' = A_Q{}'$› **have** *rFrQ'*: $extractFrame\ (r \cdot Q') = \langle A_Q{}', (r \cdot \Psi_Q{}')\rangle$
by(*simp add*: *eqvts*)

from ‹$xvec\ \sharp*\ \alpha$› **have** $(bn\ \alpha)\ \sharp*\ xvec$ **by** *simp*

from ‹$(A_P\ @\ A_Q)\ \sharp*\ \alpha$› **have** $A_P\ \sharp*\ bn\ \alpha$ **by** *simp*

from ‹$A_P\ \sharp*\ M$› ‹$A_P\ \sharp*\ xvec$› ‹$A_P\ \sharp*\ N$› **have** $A_P\ \sharp*\ (¡M(\!|\nu*xvec|\!)\langle N\rangle)$ **by** *simp*
from ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec P'$› ‹$xvec\ \sharp*\ M$› ‹*distinct xvec*› ‹$bn\ \alpha\ \sharp*\ P$› ‹$xvec\ \sharp*\ bn\ \alpha$›
have $bn\ \alpha\ \sharp*\ P'$ **by**(*simp add*: *broutputFreshChainDerivative*)

from ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec P'$› ‹$xvec\ \sharp*\ M$› ‹*distinct xvec*› ‹$bn\ \alpha\ \sharp*\ P$› ‹$bn\ \alpha\ \sharp*\ xvec$›
have $bn\ \alpha\ \sharp*\ N$ **by**(*simp add*: *broutputFreshChainDerivative*)

note ‹$bn\ \alpha\ \sharp*\ subject\ \alpha$›

moreover from ‹$bn\ \alpha\ \sharp*\ (P \parallel Q)$› **have** $bn\ \alpha\ \sharp*\ Q$ **and** $bn\ \alpha\ \sharp*\ P$ **by** *simp+*
moreover from ‹$bn\ \alpha\ \sharp*\ V$› ‹$bn\ \alpha\ \sharp*\ N$› **have** $bn\ \alpha\ \sharp*\ (N\#V)$ **by** *simp*
moreover note ‹$bn\ \alpha\ \sharp*\ \Psi_P$› ‹$bn\ \alpha\ \sharp*\ \Psi_Q$› ‹$bn\ \alpha\ \sharp*\ W$›
moreover from ‹$bn\ \alpha\ \sharp*\ X$› ‹$A_Q\ \sharp*\ bn\ \alpha$› ‹$A_Q{'}\ \sharp*\ bn\ \alpha$› ‹$bn\ \alpha\ \sharp*\ xvec$› **have** $bn\ \alpha\ \sharp*\ (X@xvec@A_Q@A_Q{'})$ **by** *simp*
moreover from ‹$bn\ \alpha\ \sharp*\ Y$› ‹$bn\ \alpha\ \sharp*\ \Psi_Q$› **have** $bn\ \alpha\ \sharp*\ (\Psi_Q\#Y)$ **by** *simp*
moreover from ‹$bn\ \alpha\ \sharp*\ Z$› ‹$bn\ \alpha\ \sharp*\ P$› ‹$bn\ \alpha\ \sharp*\ Q$› **have** $bn\ \alpha\ \sharp*\ (P\#Q\#Z)$ **by** *simp*
moreover from ‹$A_P\ \sharp*\ V$› ‹$A_P\ \sharp*\ N$› **have** $A_P\ \sharp*\ (N\#V)$ **by** *simp*
moreover note ‹$A_P\ \sharp*\ W$›
moreover from ‹$A_P\ \sharp*\ X$› ‹$A_P\ \sharp*\ A_Q$› ‹$A_Q{'}\ \sharp*\ A_P$› ‹$A_P\ \sharp*\ xvec$› **have** $A_P\ \sharp*\ (X@xvec@A_Q@A_Q{'})$ **by** *simp*
moreover from ‹$A_P\ \sharp*\ Y$› ‹$A_P\ \sharp*\ \Psi_Q$› **have** $A_P\ \sharp*\ (\Psi_Q\#Y)$ **by** *force*
moreover from ‹$A_P\ \sharp*\ Z$› ‹$A_P\ \sharp*\ Q$› ‹$A_P\ \sharp*\ P$› **have** $A_P\ \sharp*\ (P\#Q\#Z)$ **by** *simp*

moreover note ‹$¡M(\!|\nu*xvec|\!)\langle N\rangle \prec P' = \alpha \prec P''$›
ultimately obtain $p\ P\Psi'\ A_P{'}\ \Psi_P{'}$ **where** *Sp*: $set\ p \subseteq set\ (bn\ \alpha) \times set\ (bn\ (p \cdot \alpha))$ **and** *PeqP'*: $((p \cdot \Psi_P) \otimes P\Psi') \simeq \Psi_P{'}$
and *distinctPerm p* **and** $bn(p \cdot \alpha)\ \sharp*\ C'$ **and** *FrP'*: $extractFrame\ P'' = \langle A_P{'}, \Psi_P{'}\rangle$
and $A_P{'}\ \sharp*\ P''$ **and** $A_P{'}\ \sharp*\ \alpha$ **and** $A_P{'}\ \sharp*\ (p \cdot \alpha)$ **and** $A_P{'}\ \sharp*\ C$
and $bn(p \cdot \alpha)\ \sharp*\ \alpha$ **and** $bn(p \cdot \alpha)\ \sharp*\ P''$ **and** $distinct\ A_P{'}$
and $A_P{'}\ \sharp*\ (N\#V)$ **and** $A_P{'}\ \sharp*\ W$ **and** $A_P{'}\ \sharp*\ (X\ @\ xvec\ @\ A_Q\ @\ A_Q{'})$ **and** $A_P{'}\ \sharp*\ (\Psi_Q\#Y)$
and $A_P{'}\ \sharp*\ (P\#Q\#Z)$ **and** $bn(p \cdot \alpha)\ \sharp*\ (N\#V)$ **and** $bn(p \cdot \alpha)\ \sharp*\ W$ **and** $bn(p \cdot \alpha)\ \sharp*\ (X\ @\ xvec\ @\ A_Q\ @\ A_Q{'})$ **and** $bn(p \cdot \alpha)\ \sharp*\ (\Psi_Q\#Y)$
and $bn(p \cdot \alpha)\ \sharp*\ (P\#Q\#Z)$
using ‹$A_P\ \sharp*\ P$› ‹$A_P\ \sharp*\ \alpha$› ‹$A_P\ \sharp*\ C$› ‹$A_P\ \sharp*\ C'$›
‹$bn\ \alpha\ \sharp*\ C'$› ‹$\alpha \neq \tau$› ‹$A_P\ \sharp*\ P''$› ‹$distinct\ (bn\ \alpha)$›
by(*elim cBrComm2*(*4*)[**where** *b*=*C* **and** *ba*=*C'* **and** *bb*=α **and** *bc*=*P''* **and** *bf*=$(X\ @\ xvec\ @\ A_Q\ @\ A_Q{'})$ **and** *bg*=$(\Psi_Q\#Y)$ **and** *bh*=$(P\#Q\#Z)$]) (*assumption* | *simp*)+
then have $A_P{'}\ \sharp*\ Q$ **and** $A_P{'}\ \sharp*\ Z$ **and** $A_P{'}\ \sharp*\ A_Q$ **and** $A_Q{'}\ \sharp*\ A_P{'}$ **and** $A_P{'}\ \sharp*\ X$ **and** $A_P{'}\ \sharp*\ \Psi_Q$ **and** $A_P{'}\ \sharp*\ Y$ **and** $A_P{'}\ \sharp*\ P$ **and** $A_P{'}\ \sharp*\ N$
and $A_P{'}\ \sharp*\ xvec$ **and** $A_P{'}\ \sharp*\ V$ **and** $A_P{'}\ \sharp*\ W$
and $bn(p \cdot \alpha)\ \sharp*\ A_Q$ **and** $bn(p \cdot \alpha)\ \sharp*\ A_Q{'}$ **and** $bn(p \cdot \alpha)\ \sharp*\ X$ **and** $bn(p \cdot \alpha)\ \sharp*\ Y$ **and** $bn(p \cdot \alpha)\ \sharp*\ Z$ **and** $bn(p \cdot \alpha)\ \sharp*\ \Psi_Q$
and $bn(p \cdot \alpha)\ \sharp*\ Q$ **and** $bn(p \cdot \alpha)\ \sharp*\ W$ **and** $bn(p \cdot \alpha)\ \sharp*\ V$ **and** $bn(p \cdot \alpha)\ \sharp*\ N$ **and** $bn(p \cdot \alpha)\ \sharp*\ xvec$
by(*simp del*: *freshChainSimps*)+

from ‹$A_Q{'}\ \sharp*\ P''$› ‹$A_Q{'}\ \sharp*\ A_P{'}$› *FrP'*
have $A_Q{'}\ \sharp*\ \Psi_P{'}$
by(*metis extractFrameFreshChain freshFrameDest*)

from ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$› ‹$A_P{}' \sharp* Q$› ‹$A_P{}' \sharp* N$›
have $A_P{}' \sharp* Q'$
by(*simp add*: *brinputFreshChainDerivative*)
with *FrQ'* ‹$A_Q{}' \sharp* A_P{}'$› **have** $A_P{}' \sharp* \Psi_Q{}'$
by(*metis extractFrameFreshChain freshFrameDest*)

have $((p \cdot \Psi_P) \otimes P\Psi') \otimes (\Psi_Q \otimes (r \cdot Q\Psi')) \simeq \Psi_P{}' \otimes (r \cdot \Psi_Q{}')$
by (*metis Composition' PeqP' rQeqQ'*)
then have $((p \cdot \Psi_P) \otimes (P\Psi' \otimes (\Psi_Q \otimes (r \cdot Q\Psi')))) \simeq \Psi_P{}' \otimes (r \cdot \Psi_Q{}')$
by (*metis AssertionStatEqSym AssertionStatEqTrans Associativity*)
then have $((p \cdot \Psi_P) \otimes ((\Psi_Q \otimes (r \cdot Q\Psi')) \otimes P\Psi')) \simeq \Psi_P{}' \otimes (r \cdot \Psi_Q{}')$
by (*metis AssertionStatEqSym AssertionStatEqTrans Associativity associativitySym*)
then have $(p \cdot \Psi_P) \otimes (\Psi_Q \otimes ((r \cdot Q\Psi') \otimes P\Psi')) \simeq \Psi_P{}' \otimes (r \cdot \Psi_Q{}')$
by (*metis AssertionStatEqSym AssertionStatEqTrans Associativity compositionSym*)
then have $((p \cdot \Psi_P) \otimes \Psi_Q) \otimes ((r \cdot Q\Psi') \otimes P\Psi') \simeq \Psi_P{}' \otimes (r \cdot \Psi_Q{}')$
by (*metis AssertionStatEqTrans Associativity*)
then have $((p \cdot \Psi_P) \otimes \Psi_Q) \otimes (P\Psi' \otimes (r \cdot Q\Psi')) \simeq \Psi_P{}' \otimes (r \cdot \Psi_Q{}')$
by (*metis AssertionStatEqSym AssertionStatEqTrans Associativity associativitySym*)
with *Sp* ‹*bn* $\alpha \sharp* \Psi_Q$› ‹*bn* $(p \cdot \alpha) \sharp* \Psi_Q$› **have** $(p \cdot (\Psi_P \otimes \Psi_Q)) \otimes (P\Psi' \otimes (r \cdot Q\Psi')) \simeq \Psi_P{}' \otimes (r \cdot \Psi_Q{}')$
by (*simp add*: *eqvts*)

from *Sp* ‹$A_Q{}' \sharp* \alpha$› ‹$bn(p \cdot \alpha) \sharp* A_Q{}'$› **have** $A_Q{}' \sharp* (p \cdot \alpha)$
by (*metis actionFreshChain freshChainSym freshStarChainSimps fresh-star-set-eq*)

from ‹$A_P{}' \sharp* \alpha$› **have** $A_P{}' \sharp* bn\ \alpha$ **by** *simp*
from *rPerm* ‹$A_Q{}' \sharp* Q'$› ‹$A_Q{}' \sharp* xvec$› ‹$A_Q{}' \sharp* bn\ \alpha$›
have $A_Q{}' \sharp* (r \cdot Q')$
by (*metis freshAlphaPerm freshChainSym name-list-set-fresh permStarFresh*)
from *rPerm* ‹$A_P{}' \sharp* Q'$› ‹$A_P{}' \sharp* xvec$› ‹$A_P{}' \sharp* bn\ \alpha$›
have $A_P{}' \sharp* (r \cdot Q')$
by (*metis freshAlphaPerm freshChainSym name-list-set-fresh permStarFresh*)
from *rPerm* ‹$A_P{}' \sharp* \Psi_Q{}'$› ‹$A_P{}' \sharp* xvec$› ‹$A_P{}' \sharp* bn\ \alpha$›
have $A_P{}' \sharp* (r \cdot \Psi_Q{}')$
by (*metis freshAlphaPerm freshChainSym name-list-set-fresh permStarFresh*)

with ‹$A_Q{}' \sharp* \Psi_P{}'$› ‹$A_Q{}' \sharp* A_P{}'$› *FrP' rFrQ'*
have *extractFrame* $(P'' \parallel (r \cdot Q')) = \langle (A_P{}' @ A_Q{}'), (\Psi_P{}' \otimes (r \cdot \Psi_Q{}'))\rangle$
by *simp*
with ‹$PQ' = (P'' \parallel (r \cdot Q'))$›
have *extractFrame* $PQ' = \langle (A_P{}' @ A_Q{}'), (\Psi_P{}' \otimes (r \cdot \Psi_Q{}'))\rangle$
by *simp*

from ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$› ‹$bn(p \cdot \alpha) \sharp* Q$› ‹$bn(p \cdot \alpha) \sharp* N$›
have $bn(p \cdot \alpha) \sharp* Q'$ **by**(*simp add*: *brinputFreshChainDerivative*)

with *rPerm* ‹$bn(p \cdot \alpha) \sharp* \alpha$› ‹$bn(p \cdot \alpha) \sharp* xvec$›
have $bn(p \cdot \alpha) \sharp* (r \cdot Q')$
by (*metis actionFreshChain freshAlphaPerm freshChainSym name-list-set-fresh permStarFresh*)

from ‹$A_P' \sharp* P''$› ‹$A_P' \sharp* (r \cdot Q')$› ‹$PQ' = P'' \parallel (r \cdot Q')$› **have** $A_P' \sharp* PQ'$ **by** *simp*
from ‹$A_Q' \sharp* P''$› ‹$A_Q' \sharp* (r \cdot Q')$› ‹$PQ' = P'' \parallel (r \cdot Q')$› **have** $A_Q' \sharp* PQ'$ **by** *simp*

note ‹*set* $p \subseteq (set\ (bn\ \alpha)) \times set\ (bn(p \cdot \alpha))$›
‹$((p \cdot (\Psi_P \otimes \Psi_Q)) \otimes (P\Psi' \otimes (r \cdot Q\Psi'))) \simeq \Psi_P' \otimes (r \cdot \Psi_Q')$› ‹*distinctPerm* p›
‹*extractFrame* $PQ' = \langle (A_P'@A_Q'), (\Psi_P' \otimes (r \cdot \Psi_Q')) \rangle$›

moreover from ‹$A_P' \sharp* PQ'$› ‹$A_Q' \sharp* PQ'$› **have** $(A_P'@A_Q') \sharp* PQ'$ **by** *simp*
moreover from ‹$A_P' \sharp* \alpha$› ‹$A_Q' \sharp* \alpha$› **have** $(A_P'@A_Q') \sharp* \alpha$ **by** *simp*
moreover from ‹$A_P' \sharp* (p \cdot \alpha)$› ‹$A_Q' \sharp* (p \cdot \alpha)$› **have** $(A_P'@A_Q') \sharp* (p \cdot \alpha)$ **by** *simp*
moreover from ‹$A_P' \sharp* C$› ‹$A_Q' \sharp* C$› **have** $(A_P'@A_Q') \sharp* C$ **by** *simp*
moreover note ‹$bn(p \cdot \alpha) \sharp* C'$› ‹$bn(p \cdot \alpha) \sharp* \alpha$›
moreover from ‹$bn(p \cdot \alpha) \sharp* (r \cdot Q')$› ‹$bn(p \cdot \alpha) \sharp* P''$› ‹$PQ' = P'' \parallel (r \cdot Q')$›
have $bn(p \cdot \alpha) \sharp* PQ'$ **by** *simp*
moreover from ‹$A_P' \sharp* V$› ‹$A_Q' \sharp* V$› **have** $(A_P'@A_Q') \sharp* V$ **by** *simp*
moreover from ‹$A_P' \sharp* W$› ‹$A_Q' \sharp* W$› **have** $(A_P'@A_Q') \sharp* W$ **by** *simp*
moreover from ‹$A_P' \sharp* X$› ‹$A_Q' \sharp* X$› **have** $(A_P'@A_Q') \sharp* X$ **by** *simp*
moreover from ‹$A_P' \sharp* Y$› ‹$A_Q' \sharp* Y$› **have** $(A_P'@A_Q') \sharp* Y$ **by** *simp*
moreover from ‹$A_P' \sharp* Z$› ‹$A_Q' \sharp* Z$› **have** $(A_P'@A_Q') \sharp* Z$ **by** *simp*
moreover from ‹*distinct* A_P'› ‹*distinct* A_Q'› ‹$A_Q' \sharp* A_P'$›
have $distinct(A_P'@A_Q')$ **by** *simp*
moreover note ‹$bn(p \cdot \alpha) \sharp* V$› ‹$bn(p \cdot \alpha) \sharp* W$›
‹$bn(p \cdot \alpha) \sharp* X$› ‹$bn(p \cdot \alpha) \sharp* Y$› ‹$bn(p \cdot \alpha) \sharp* Z$›
ultimately show *?case*
by(*rule cBrComm2*(*63*))
next
case *cBrClose*
then show *?case*
by(*simp add*: *residualInject*)
next
case(*cOpen* Ψ P M *xvec1 xvec2* N P' x A_P Ψ_P C C' α P'' V W X Y Z)
from ‹$M(\!|\nu*(xvec1@x\#xvec2)|\!)\langle N \rangle \prec P' = \alpha \prec P''$› ‹$x \sharp xvec1$› ‹$x \sharp xvec2$› ‹$x \sharp \alpha$› ‹$x \sharp P''$› ‹$distinct(bn\ \alpha)$› ‹$A_P \sharp* \alpha$› ‹$x \sharp \alpha$›
obtain *yvec1 y yvec2* N' **where** *yvecEq*: $bn\ \alpha = yvec1@y\#yvec2$ **and** $P'eqP''$: $(\!|\nu*(xvec1@xvec2)|\!)N \prec' P' = (\!|\nu*(yvec1@yvec2)|\!)([(x, y)] \cdot N') \prec' ([(x, y)] \cdot P'')$ **and** $A_P \sharp* N'$ **and** *Subj*: *subject* $\alpha = Some\ M$ **and** $x \sharp N'$ **and** αeq: $\alpha = M(\!|\nu*(yvec1@y\#yvec2)|\!)\langle N' \rangle$
apply(*cases rule*: *actionCases*[**where** α=α])

apply(*simp add*: *residualInject*)
apply(*simp add*: *residualInject*)
apply(*simp add*: *residualInject*)
apply(*metis boundOutputOpenDest*)
apply(*simp add*: *residualInject*)
by(*simp add*: *residualInject*)

note ‹$A_P \sharp* P$› ‹$A_P \sharp* M$›
moreover from *Subj yvecEq* ‹*bn* $\alpha \sharp* subject\ \alpha$› **have** *yvec1* $\sharp*$ *M yvec2* $\sharp*$ *M* **by** *simp+*
moreover from *yvecEq* ‹$A_P \sharp* \alpha$› **have** $A_P \sharp* (yvec1@yvec2)$ **by** *simp*
moreover note ‹$A_P \sharp* C$›
moreover from *yvecEq* ‹*bn* $\alpha \sharp* (\!|\nu x|\!) P$› ‹$x \sharp \alpha$› **have** $(yvec1@yvec2) \sharp* P$ **by** *simp*
moreover from *yvecEq* ‹*bn* $\alpha \sharp* C'$› ‹*bn* $\alpha \sharp* V$› ‹*bn* $\alpha \sharp* W$› ‹*bn* $\alpha \sharp* X$› ‹*bn* $\alpha \sharp* Y$› ‹*bn* $\alpha \sharp* Z$› ‹*distinct*(*bn* α)› ‹$x \sharp \alpha$›
have $(yvec1@yvec2) \sharp* C'$ **and** $(yvec1@yvec2) \sharp* V$ **and** $(yvec1@yvec2) \sharp* W$ **and** $(yvec1@yvec2) \sharp* (x\#y\#X)$ **and** $(yvec1@yvec2) \sharp* Y$ **and** $(yvec1@yvec2) \sharp* Z$
by *simp+*
moreover note ‹$A_P \sharp* V$› ‹$A_P \sharp* W$›
moreover from ‹$A_P \sharp* X$› ‹$x \sharp A_P$› ‹$A_P \sharp* \alpha$› *yvecEq* **have** $A_P \sharp* (x\#y\#X)$ **by** *simp*
moreover note ‹$A_P \sharp* Y$› ‹$A_P \sharp* Z$›
moreover from ‹$A_P \sharp* N'$› ‹$A_P \sharp* P''$› ‹$x \sharp A_P$› ‹$A_P \sharp* \alpha$› *yvecEq* **have** $A_P \sharp* ([(x, y)] \cdot N')$ **and** $A_P \sharp* ([(x, y)] \cdot P'')$
by *simp+*
moreover from *yvecEq* ‹*distinct*(*bn* α)› **have** *distinct*$(yvec1@yvec2)$ **by** *simp*
moreover from $P' eqP''$ **have** $M(\!|\nu*(xvec1@xvec2)|\!)\langle N\rangle \prec P' = M(\!|\nu*(yvec1@yvec2)|\!)\langle([(x, y)] \cdot N')\rangle \prec ([(x, y)] \cdot P'')$
by(*simp add*: *residualInject*)
ultimately obtain $p\ \Psi'\ A_P'\ \Psi_P'$ **where** *S*: *set* $p \subseteq$ *set* $(yvec1@yvec2) \times$ *set* $(p \cdot (yvec1@yvec2))$ **and** *PeqP'*: $((p \cdot \Psi_P) \otimes \Psi') \simeq \Psi_P'$
and *distinctPerm p* **and** $(p \cdot (yvec1@yvec2)) \sharp* C'$ **and** *FrP'*: *extractFrame*$([(x, y)] \cdot P'') = \langle A_P', \Psi_P'\rangle$
and $A_P' \sharp* ([(x, y)] \cdot P'')$ **and** $A_P' \sharp* ([(x, y)] \cdot N')$ **and** $A_P' \sharp* C$ **and** $(p \cdot (yvec1@yvec2)) \sharp* ([(x, y)] \cdot N')$ **and** $A_P' \sharp* M$ **and** $(p \cdot (yvec1@yvec2)) \sharp* (yvec1@yvec2)$ **and** $(p \cdot (yvec1@yvec2)) \sharp* M$ **and** *distinct* A_P'
and $(p \cdot (yvec1@yvec2)) \sharp* ([(x, y)] \cdot P'')$ **and** $(yvec1@yvec2) \sharp* A_P'$ **and** $(p \cdot (yvec1@yvec2)) \sharp* A_P'$
and $A_P' \sharp* V$ **and** $A_P' \sharp* W$ **and** $A_P' \sharp* (x\#y\#X)$ **and** $A_P' \sharp* Y$ **and** $A_P' \sharp* Z$ **and** $(p \cdot (yvec1@yvec2)) \sharp* (x\#y\#X)$
and $(p \cdot (yvec1@yvec2)) \sharp* V$ **and** $(p \cdot (yvec1@yvec2)) \sharp* W$ **and** $(p \cdot (yvec1@yvec2)) \sharp* Y$ **and** $(p \cdot (yvec1@yvec2)) \sharp* Z$ **using** ‹$A_P \sharp* C'$›
by(*elim cOpen*(*4*)[**where** *b*=*C* **and** *ba*=C' **and** *bd*=*V* **and** *be*=*W* **and** *bf*=$x\#y\#X$ **and** *bg*=*Y* **and** *bh*=*Z*]) (*assumption* | *simp*)+

from ‹$A_P' \sharp* (x\#y\#X)$› **have** $x \sharp A_P'$ **and** $y \sharp A_P'$ **and** $A_P' \sharp* X$ **by** *simp+*
from ‹$(p \cdot (yvec1@yvec2)) \sharp* (x\#y\#X)$› **have** $x \sharp (p \cdot (yvec1@yvec2))$ **and**

$y \sharp (p \cdot (yvec1@yvec2))$ **and** $(p \cdot (yvec1@yvec2)) \sharp* X$ **by** *simp+*

from ‹$x \sharp \alpha$› *yvecEq* **have** $x \sharp yvec1$ **and** $x \neq y$ **and** $x \sharp yvec2$ **by** *simp+*
from ‹*distinct*(*bn* α)› *yvecEq* **have** $yvec1 \sharp* yvec2$ **and** $y \sharp yvec1$ **and** $y \sharp yvec2$ **by** *simp+*
from ‹*bn* $\alpha \sharp* C'$› *yvecEq* **have** $yvec1 \sharp* C'$ **and** $y \sharp C'$ **and** $yvec2 \sharp* C'$ **by** *simp+*

from *S* ‹$x \sharp \alpha$› ‹$x \sharp p \cdot (yvec1@yvec2)$› *yvecEq* **have** $x \sharp p$ **by**(*intro freshAlphaSwap*) (*assumption* | *simp*)+
from *S* ‹*distinct*(*bn* α)› ‹$y \sharp p \cdot (yvec1@yvec2)$› *yvecEq* **have** $y \sharp p$ **by**(*intro freshAlphaSwap*) (*assumption* | *simp*)+

from *yvecEq S* ‹$x \sharp yvec1$› ‹$x \sharp yvec2$› ‹$y \sharp yvec1$› ‹$y \sharp yvec2$› ‹$x \sharp p \cdot (yvec1@yvec2)$› ‹$y \sharp p \cdot (yvec1@yvec2)$›
have *set* $((y, x)\#p) \subseteq set(bn\ \alpha) \times set(bn(((y, x)\#p) \cdot \alpha))$
apply(*simp add*: *bnEqvt*[*symmetric*])
by(*auto simp add*: *eqvts calc-atm*)

moreover from *PeqP'* **have** $([(y, x)] \cdot ((p \cdot \Psi_P) \otimes \Psi')) \simeq [(y, x)] \cdot \Psi_P'$
by(*simp add*: *AssertionStatEqClosed*)
then have $(((y, x)\#p) \cdot \Psi_P) \otimes ([(y, x)] \cdot \Psi') \simeq ([(y, x)] \cdot \Psi_P')$
by(*simp add*: *eqvts*)
moreover from ‹*distinctPerm p*› *S* ‹$x \neq y$› ‹$x \sharp p$› ‹$y \sharp p$› **have** *distinctPerm*$((y, x)\#p)$
by *simp*
moreover from *FrP'* **have** $([(x, y)] \cdot (extractFrame([(x, y)] \cdot P'))) = ([(x, y)] \cdot \langle A_P', \Psi_P' \rangle)$
by *simp*
with ‹$x \sharp A_P'$› ‹$y \sharp A_P'$› **have** $extractFrame\ P'' = \langle A_P', ([(y, x)] \cdot \Psi_P') \rangle$
by(*simp add*: *eqvts name-swap*)
moreover from ‹$x \sharp p$› ‹$y \sharp p$› ‹$x \sharp N'$› ‹$(p \cdot (yvec1@yvec2)) \sharp* ([(x, y)] \cdot N')$› **have** $([(y, x)] \cdot p \cdot (yvec1@yvec2)) \sharp* ([(y, x)] \cdot [(x, y)] \cdot N')$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
then have $(((y, x)\#p) \cdot (yvec1@yvec2)) \sharp* N'$ **by**(*simp add*: *name-swap*)
with ‹$x \sharp yvec1$› ‹$x \sharp yvec2$› ‹$x \neq y$› ‹$x \sharp C$› ‹$y \sharp yvec1$› ‹$y \sharp yvec2$› ‹$x \sharp p$› ‹$y \sharp p$› ‹$x \sharp N'$› *yvecEq*
have $bn(((y, x)\#p) \cdot \alpha) \sharp* N'$ **by**(*simp add*: *bnEqvt*[*symmetric*]) (*simp add*: *eqvts perm-compose calc-atm freshChainSimps*)
moreover from ‹$x \sharp p$› ‹$y \sharp p$› ‹$x \sharp N'$› ‹$(p \cdot (yvec1@yvec2)) \sharp* ([(x, y)] \cdot P'')$› **have** $([(y, x)] \cdot p \cdot (yvec1@yvec2)) \sharp* ([(y, x)] \cdot [(x, y)] \cdot P'')$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
then have $(((y, x)\#p) \cdot (yvec1@yvec2)) \sharp* P''$ **by**(*simp add*: *name-swap*)
with ‹$x \sharp yvec1$› ‹$x \sharp yvec2$› ‹$x \neq y$› ‹$y \sharp yvec1$› ‹$y \sharp yvec2$› ‹$x \sharp p$› ‹$y \sharp p$› ‹$x \sharp P''$› *yvecEq*
have $bn(((y, x)\#p) \cdot \alpha) \sharp* P''$ **by**(*simp add*: *bnEqvt*[*symmetric*]) (*simp add*: *perm-compose calc-atm eqvts freshChainSimps*)
moreover from ‹$x \sharp p$› ‹$y \sharp p$› ‹$x \sharp A_P'$› ‹$(p \cdot (yvec1@yvec2)) \sharp* A_P'$› **have** $(p \cdot (yvec1@x\#yvec2)) \sharp* A_P'$

by(*simp add*: *eqvts freshChainSimps*)
then have $([(y, x)] \cdot p \cdot (yvec1@x\#yvec2)) \sharp* ([(y, x)] \cdot A_P{}')$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with $\langle x \sharp yvec1\rangle \langle x \sharp yvec2\rangle \langle x \neq y\rangle \langle x \sharp A_P{}'\rangle \langle y \sharp yvec1\rangle \langle y \sharp yvec2\rangle \langle x \sharp p\rangle \langle y \sharp p\rangle \langle y \sharp A_P{}'\rangle$ *yvecEq*
have $bn(((y, x)\#p) \cdot \alpha) \sharp* A_P{}'$ **by**(*simp add*: *bnEqvt*[*symmetric*]) (*simp add*: *perm-compose calc-atm eqvts freshChainSimps*)
moreover from $\langle x \sharp p\rangle \langle y \sharp p\rangle \langle x \sharp C'\rangle \langle (p \cdot (yvec1@yvec2)) \sharp* C'\rangle$ **have** $(p \cdot (yvec1@x\#yvec2)) \sharp* C'$
by(*simp add*: *eqvts freshChainSimps*)
then have $([(y, x)] \cdot p \cdot (yvec1@x\#yvec2)) \sharp* ([(y, x)] \cdot C')$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with $\langle x \sharp yvec1\rangle \langle x \sharp yvec2\rangle \langle x \neq y\rangle \langle x \sharp C'\rangle \langle y \sharp yvec1\rangle \langle y \sharp yvec2\rangle \langle x \sharp p\rangle \langle y \sharp p\rangle \langle y \sharp C'\rangle$ *yvecEq*
have $bn(((y, x)\#p) \cdot \alpha) \sharp* C'$ **by**(*simp add*: *bnEqvt*[*symmetric*]) (*simp add*: *perm-compose calc-atm eqvts freshChainSimps*)
moreover from $\langle x \sharp p\rangle \langle y \sharp p\rangle \langle x \sharp X\rangle \langle (p \cdot (yvec1@yvec2)) \sharp* X\rangle$ **have** $(p \cdot (yvec1@x\#yvec2)) \sharp* X$
by(*simp add*: *eqvts freshChainSimps*)
then have $([(y, x)] \cdot p \cdot (yvec1@x\#yvec2)) \sharp* ([(y, x)] \cdot X)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with $\langle x \sharp yvec1\rangle \langle x \sharp yvec2\rangle \langle x \neq y\rangle \langle x \sharp X\rangle \langle y \sharp yvec1\rangle \langle y \sharp yvec2\rangle \langle x \sharp p\rangle \langle y \sharp p\rangle \langle bn\ \alpha \sharp* X\rangle$ *yvecEq*
have $bn(((y, x)\#p) \cdot \alpha) \sharp* X$ **by**(*simp add*: *bnEqvt*[*symmetric*]) (*simp add*: *perm-compose calc-atm eqvts freshChainSimps*)
moreover from $\langle x \sharp p\rangle \langle y \sharp p\rangle \langle x \sharp V\rangle \langle (p \cdot (yvec1@yvec2)) \sharp* V\rangle$ **have** $(p \cdot (yvec1@x\#yvec2)) \sharp* V$
by(*simp add*: *eqvts freshChainSimps*)
then have $([(y, x)] \cdot p \cdot (yvec1@x\#yvec2)) \sharp* ([(y, x)] \cdot V)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with $\langle x \sharp yvec1\rangle \langle x \sharp yvec2\rangle \langle x \neq y\rangle \langle x \sharp V\rangle \langle y \sharp yvec1\rangle \langle y \sharp yvec2\rangle \langle x \sharp p\rangle \langle y \sharp p\rangle \langle bn\ \alpha \sharp* V\rangle$ *yvecEq*
have $bn(((y, x)\#p) \cdot \alpha) \sharp* V$ **by**(*simp add*: *bnEqvt*[*symmetric*]) (*simp add*: *perm-compose calc-atm eqvts freshChainSimps*)
moreover from $\langle x \sharp p\rangle \langle y \sharp p\rangle \langle x \sharp W\rangle \langle (p \cdot (yvec1@yvec2)) \sharp* W\rangle$ **have** $(p \cdot (yvec1@x\#yvec2)) \sharp* W$
by(*simp add*: *eqvts freshChainSimps*)
then have $([(y, x)] \cdot p \cdot (yvec1@x\#yvec2)) \sharp* ([(y, x)] \cdot W)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with $\langle x \sharp yvec1\rangle \langle x \sharp yvec2\rangle \langle x \neq y\rangle \langle x \sharp W\rangle \langle y \sharp yvec1\rangle \langle y \sharp yvec2\rangle \langle x \sharp p\rangle \langle y \sharp p\rangle \langle bn\ \alpha \sharp* W\rangle$ *yvecEq*
have $bn(((y, x)\#p) \cdot \alpha) \sharp* W$ **by**(*simp add*: *bnEqvt*[*symmetric*]) (*simp add*: *perm-compose calc-atm eqvts freshChainSimps*)
moreover from $\langle x \sharp p\rangle \langle y \sharp p\rangle \langle x \sharp Y\rangle \langle (p \cdot (yvec1@yvec2)) \sharp* Y\rangle$ **have** $(p \cdot (yvec1@x\#yvec2)) \sharp* Y$
by(*simp add*: *eqvts freshChainSimps*)
then have $([(y, x)] \cdot p \cdot (yvec1@x\#yvec2)) \sharp* ([(y, x)] \cdot Y)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with $\langle x \sharp yvec1\rangle \langle x \sharp yvec2\rangle \langle x \neq y\rangle \langle x \sharp Y\rangle \langle y \sharp yvec1\rangle \langle y \sharp yvec2\rangle \langle x \sharp p\rangle$

‹$y \sharp p$› ‹$bn\ \alpha \sharp* Y$› *yvecEq*
have $bn(((y,\ x)\#p) \cdot \alpha) \sharp* Y$ **by**(*simp add*: *bnEqvt*[*symmetric*]) (*simp add*: *perm-compose calc-atm eqvts freshChainSimps*)
moreover from ‹$x \sharp p$› ‹$y \sharp p$› ‹$x \sharp Z$› ‹$(p \cdot (yvec1@yvec2)) \sharp* Z$› **have** $(p \cdot (yvec1@x\#yvec2)) \sharp* Z$
by(*simp add*: *eqvts freshChainSimps*)
then have $([(y,\ x)] \cdot p \cdot (yvec1@x\#yvec2)) \sharp* ([(y,\ x)] \cdot Z)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$x \sharp yvec1$› ‹$x \sharp yvec2$› ‹$x \neq y$› ‹$x \sharp Z$› ‹$y \sharp yvec1$› ‹$y \sharp yvec2$› ‹$x \sharp p$› ‹$y \sharp p$› ‹$bn\ \alpha \sharp* Z$› *yvecEq*
have $bn(((y,\ x)\#p) \cdot \alpha) \sharp* Z$ **by**(*simp add*: *bnEqvt*[*symmetric*]) (*simp add*: *perm-compose calc-atm eqvts freshChainSimps*)
moreover from ‹$(yvec1@yvec2) \sharp* A_P{}'$› ‹$y \sharp A_P{}'$› *yvecEq* **have** $bn\ \alpha \sharp* A_P{}'$
by *simp*
moreover from ‹$A_P{}' \sharp* ([(x,\ y)] \cdot N')$› **have** $([(x,\ y)] \cdot A_P{}') \sharp* ([(x,\ y)] \cdot [(x,\ y)] \cdot N')$
by(*simp only*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$x \sharp A_P{}'$› ‹$y \sharp A_P{}'$› **have** $A_P{}' \sharp* N'$ **by** *simp*
with ‹$A_P{}' \sharp* M$› ‹$(yvec1@yvec2) \sharp* A_P{}'$› ‹$y \sharp A_P{}'$› αeq **have** $A_P{}' \sharp* \alpha$ **by** *simp*
moreover then have $(((y,\ x)\#p) \cdot A_P{}') \sharp* (((y,\ x)\#p) \cdot \alpha)$
by(*simp only*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$x \sharp A_P{}'$› ‹$y \sharp A_P{}'$› S ‹$(yvec1@yvec2) \sharp* A_P{}'$› ‹$(p \cdot (yvec1@yvec2)) \sharp* A_P{}'$›
have $A_P{}' \sharp* (((y,\ x)\#p) \cdot \alpha)$ **by**(*simp add*: *eqvts*)
moreover from ‹$A_P{}' \sharp* ([(x,\ y)] \cdot P'')$› **have** $([(x,\ y)] \cdot A_P{}') \sharp* ([(x,\ y)] \cdot [(x,\ y)] \cdot P'')$
by(*simp only*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$x \sharp A_P{}'$› ‹$y \sharp A_P{}'$› **have** $A_P{}' \sharp* P''$ **by** *simp*
moreover from *yvecEq* αeq ‹$(p \cdot (yvec1@yvec2)) \sharp* (yvec1@yvec2)$› ‹$y \sharp p$› ‹$x \sharp \alpha$› S ‹$(p \cdot (yvec1@yvec2)) \sharp* M$›‹$(p \cdot (yvec1@yvec2)) \sharp* ([(x,\ y)] \cdot N')$› ‹$y \sharp yvec1$›‹$y \sharp yvec2$› ‹$x \sharp p$›
have $bn(((y,\ x)\#p) \cdot \alpha) \sharp* \alpha$
apply(*simp add*: *eqvts del*: *set-append*)
apply(*intro conjI*)
apply(*simp add*: *perm-compose eqvts del*: *set-append*)
apply(*simp add*: *perm-compose freshChainSimps*(6) *calc-atm eqvts del*: *set-append*)
apply(*simp add*: *perm-compose eqvts del*: *set-append*)
apply(*simp add*: *perm-compose eqvts swapStarFresh del*: *set-append*)
apply(*simp add*: *perm-compose freshChainSimps*(6) *calc-atm eqvts del*: *set-append*)
apply(*simp add*: *perm-compose freshChainSimps*(6) *calc-atm eqvts del*: *set-append*)
apply(*simp add*: *perm-compose freshChainSimps*(6) *calc-atm eqvts del*: *set-append*)
apply(*simp add*: *perm-compose freshChainSimps*(6) *swapStarFresh calc-atm eqvts del*: *set-append*)
apply(*simp add*: *perm-compose freshChainSimps*(6) *calc-atm eqvts del*:

set-append)
apply(*subst pt-fresh-star-bij*[*symmetric*, *OF pt-name-inst*, *OF at-name-inst*, **where** *pi*=[(*x*, *y*)]])
apply(*simp add*: *perm-compose freshChainSimps*(*6*) *calc-atm eqvts del*: *set-append*)
apply(*simp add*: *perm-compose freshChainSimps*(*6*) *calc-atm eqvts del*: *set-append*)
apply(*subst pt-fresh-star-bij*[*symmetric*, *OF pt-name-inst*, *OF at-name-inst*, **where** *pi*=[(*x*, *y*)]])
by(*simp add*: *perm-compose freshChainSimps*(*6*) *calc-atm eqvts del*: *set-append*)
moreover note ‹A_P' ♯* C› ‹A_P' ♯* V› ‹A_P' ♯* W› ‹A_P' ♯* X› ‹A_P' ♯* Y› ‹A_P' ♯* Z› ‹*distinct* A_P'›

ultimately show *?case*
by(*elim cOpen*)
next
case(*cBrOpen* Ψ P M *xvec1* *xvec2* N P' x A_P Ψ_P C C' α P'' V W X Y Z)
from ‹¡M⦇ν*(*xvec1*@x#*xvec2*)⦈⟨N⟩ ≺ P' = α ≺ P''› ‹x ♯ *xvec1*› ‹x ♯ *xvec2*› ‹x ♯ α› ‹x ♯ P''› ‹*distinct*(*bn* α)› ‹A_P ♯* α› ‹x ♯ α›
obtain *yvec1* y *yvec2* N' **where** *yvecEq*: *bn* α = *yvec1*@y#*yvec2* **and** P'*eqP*'': ⦇ν*(*xvec1*@*xvec2*)⦈N ≺' P' = ⦇ν*(*yvec1*@*yvec2*)⦈([(*x*, *y*)] ∙ N') ≺' ([(*x*, *y*)] ∙ P'') **and** A_P ♯* N' **and** *Subj*: *subject* α = *Some* M **and** x ♯ N' **and** α*eq*: α = ¡M⦇ν*(*yvec1*@y#*yvec2*)⦈⟨N'⟩
apply(*cases rule*: *actionCases*[**where** α=α])
apply(*simp-all add*: *residualInject*)
by (*metis boundOutputOpenDest*)

note ‹A_P ♯* P› ‹A_P ♯* M›
moreover from *Subj* *yvecEq* ‹*bn* α ♯* *subject* α› **have** *yvec1* ♯* M *yvec2* ♯* M **by** *simp*+
moreover from *yvecEq* ‹A_P ♯* α› **have** A_P ♯* (*yvec1*@*yvec2*) **by** *simp*
moreover note ‹A_P ♯* C›
moreover from *yvecEq* ‹*bn* α ♯* (⦇$\nu$$x$⦈)$P$› ‹$x$ ♯ α› **have** (*yvec1*@*yvec2*) ♯* P **by** *simp*
moreover from *yvecEq* ‹*bn* α ♯* C'› ‹*bn* α ♯* V› ‹*bn* α ♯* W› ‹*bn* α ♯* X› ‹*bn* α ♯* Y› ‹*bn* α ♯* Z› ‹*distinct*(*bn* α)› ‹x ♯ α›
have (*yvec1*@*yvec2*) ♯* C' **and** (*yvec1*@*yvec2*) ♯* V **and** (*yvec1*@*yvec2*) ♯* W **and** (*yvec1*@*yvec2*) ♯* (*x*#*y*#*X*) **and** (*yvec1*@*yvec2*) ♯* Y **and** (*yvec1*@*yvec2*) ♯* Z
by *simp*+
moreover note ‹A_P ♯* V› ‹A_P ♯* W›
moreover from ‹A_P ♯* X› ‹x ♯ A_P› ‹A_P ♯* α› *yvecEq* **have** A_P ♯* (*x*#*y*#*X*) **by** *simp*
moreover note ‹A_P ♯* Y› ‹A_P ♯* Z›
moreover from ‹A_P ♯* N'› ‹A_P ♯* P''› ‹x ♯ A_P› ‹A_P ♯* α› *yvecEq* **have** A_P ♯* ([(*x*, *y*)] ∙ N') **and** A_P ♯* ([(*x*, *y*)] ∙ P'')
by *simp*+
moreover from *yvecEq* ‹*distinct*(*bn* α)› **have** *distinct*(*yvec1*@*yvec2*) **by** *simp*
moreover from P'*eqP*'' **have** ¡M⦇ν*(*xvec1*@*xvec2*)⦈⟨N⟩ ≺ P' = ¡M⦇ν*(*yvec1*@*yvec2*)⦈⟨([(*x*,

$y)] \cdot N')\rangle \prec ([(x,\ y)] \cdot P'')$
by(*simp add*: *residualInject*)
ultimately obtain $p\ \Psi'\ A_P'\ \Psi_P'$ **where** S: *set* $p \subseteq$ *set* $(yvec1@yvec2) \times$ *set* $(p \cdot (yvec1@yvec2))$ **and** $PeqP'$: $((p \cdot \Psi_P) \otimes \Psi') \simeq \Psi_P'$
and *distinctPerm* p **and** $(p \cdot (yvec1@yvec2))\ \sharp*\ C'$ **and** FrP': *extractFrame*$([(x,\ y)] \cdot P'') = \langle A_P',\ \Psi_P'\rangle$
and $A_P'\ \sharp*\ ([(x,\ y)] \cdot P'')$ **and** $A_P'\ \sharp*\ ([(x,\ y)] \cdot N')$ **and** $A_P'\ \sharp*\ C$ **and** $(p \cdot (yvec1@yvec2))\ \sharp*\ ([(x,\ y)] \cdot N')$ **and** $A_P'\ \sharp*\ M$ **and** $(p \cdot (yvec1@yvec2))\ \sharp*\ (yvec1@yvec2)$ **and** $(p \cdot (yvec1@yvec2))\ \sharp*\ M$ **and** *distinct* A_P'
and $(p \cdot (yvec1@yvec2))\ \sharp*\ ([(x,\ y)] \cdot P'')$ **and** $(yvec1@yvec2)\ \sharp*\ A_P'$ **and** $(p \cdot (yvec1@yvec2))\ \sharp*\ A_P'$
and $A_P'\ \sharp*\ V$ **and** $A_P'\ \sharp*\ W$ **and** $A_P'\ \sharp*\ (x\#y\#X)$ **and** $A_P'\ \sharp*\ Y$ **and** $A_P'\ \sharp*\ Z$ **and** $(p \cdot (yvec1@yvec2))\ \sharp*\ (x\#y\#X)$
and $(p \cdot (yvec1@yvec2))\ \sharp*\ V$ **and** $(p \cdot (yvec1@yvec2))\ \sharp*\ W$ **and** $(p \cdot (yvec1@yvec2))\ \sharp*\ Y$ **and** $(p \cdot (yvec1@yvec2))\ \sharp*\ Z$ **using** $\langle A_P\ \sharp*\ C'\rangle$
by(*elim cBrOpen*(4)[**where** $b{=}C$ **and** $ba{=}C'$ **and** $bd{=}V$ **and** $be{=}W$ **and** $bf{=}x\#y\#X$ **and** $bg{=}Y$ **and** $bh{=}Z$]) (*assumption* | *simp*)+

from $\langle A_P'\ \sharp*\ (x\#y\#X)\rangle$ **have** $x\ \sharp\ A_P'$ **and** $y\ \sharp\ A_P'$ **and** $A_P'\ \sharp*\ X$ **by** *simp*+
from $\langle (p \cdot (yvec1@yvec2))\ \sharp*\ (x\#y\#X)\rangle$ **have** $x\ \sharp\ (p \cdot (yvec1@yvec2))$ **and** $y\ \sharp\ (p \cdot (yvec1@yvec2))$ **and** $(p \cdot (yvec1@yvec2))\ \sharp*\ X$ **by** *simp*+

from $\langle x\ \sharp\ \alpha\rangle$ *yvecEq* **have** $x\ \sharp\ yvec1$ **and** $x \neq y$ **and** $x\ \sharp\ yvec2$ **by** *simp*+
from $\langle$*distinct*$(bn\ \alpha)\rangle$ *yvecEq* **have** $yvec1\ \sharp*\ yvec2$ **and** $y\ \sharp\ yvec1$ **and** $y\ \sharp\ yvec2$ **by** *simp*+
from $\langle bn\ \alpha\ \sharp*\ C'\rangle$ *yvecEq* **have** $yvec1\ \sharp*\ C'$ **and** $y\ \sharp\ C'$ **and** $yvec2\ \sharp*\ C'$ **by** *simp*+

from S $\langle x\ \sharp\ \alpha\rangle$ $\langle x\ \sharp\ p \cdot (yvec1@yvec2)\rangle$ *yvecEq* **have** $x\ \sharp\ p$ **by**(*intro freshAlphaSwap*) (*assumption* | *simp*)+
from S $\langle$*distinct*$(bn\ \alpha)\rangle$ $\langle y\ \sharp\ p \cdot (yvec1@yvec2)\rangle$ *yvecEq* **have** $y\ \sharp\ p$ **by**(*intro freshAlphaSwap*) (*assumption* | *simp*)+

from *yvecEq* S $\langle x\ \sharp\ yvec1\rangle$ $\langle x\ \sharp\ yvec2\rangle$ $\langle y\ \sharp\ yvec1\rangle$ $\langle y\ \sharp\ yvec2\rangle$ $\langle x\ \sharp\ p \cdot (yvec1@yvec2)\rangle$ $\langle y\ \sharp\ p \cdot (yvec1@yvec2)\rangle$
have *set* $((y,\ x)\#p) \subseteq$ *set*$(bn\ \alpha) \times$ *set*$(bn(((y,\ x)\#p) \cdot \alpha))$
apply(*simp add*: *bnEqvt*[*symmetric*])
by(*auto simp add*: *eqvts calc-atm*)

moreover from *PeqP'* **have** $([(y,\ x)] \cdot ((p \cdot \Psi_P) \otimes \Psi')) \simeq [(y,\ x)] \cdot \Psi_P'$
by(*simp add*: *AssertionStatEqClosed*)
then have $(((y,\ x)\#p) \cdot \Psi_P) \otimes ([(y,\ x)] \cdot \Psi') \simeq ([(y,\ x)] \cdot \Psi_P')$
by(*simp add*: *eqvts*)
moreover from $\langle$*distinctPerm* $p\rangle$ S $\langle x \neq y\rangle$ $\langle x\ \sharp\ p\rangle$ $\langle y\ \sharp\ p\rangle$ **have** *distinctPerm*$((y,\ x)\#p)$
by *simp*
moreover from *FrP'* **have** $([(x,\ y)] \cdot (extractFrame([(x,\ y)] \cdot P''))) = ([(x,\ y)] \cdot \langle A_P',\ \Psi_P'\rangle)$
by *simp*

with ‹$x \sharp A_P{}'$› ‹$y \sharp A_P{}'$› **have** *extractFrame* $P'' = \langle A_P{}', ([(y, x)] \cdot \Psi_P{}')\rangle$
by(*simp add*: *eqvts name-swap*)
moreover from ‹$x \sharp p$› ‹$y \sharp p$› ‹$x \sharp N'$› ‹$(p \cdot (yvec1@yvec2)) \sharp* ([(x, y)] \cdot N')$› **have** $([(y, x)] \cdot p \cdot (yvec1@yvec2)) \sharp* ([(y, x)] \cdot [(x, y)] \cdot N')$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
then have $(((y, x)\#p) \cdot (yvec1@yvec2)) \sharp* N'$ **by**(*simp add*: *name-swap*)
with ‹$x \sharp yvec1$› ‹$x \sharp yvec2$› ‹$x \neq y$› ‹$x \sharp C$› ‹$y \sharp yvec1$› ‹$y \sharp yvec2$› ‹$x \sharp p$› ‹$y \sharp p$› ‹$x \sharp N'$› *yvecEq*
have $bn(((y, x)\#p) \cdot \alpha) \sharp* N'$ **by**(*simp add*: *bnEqvt*[*symmetric*]) (*simp add*: *eqvts perm-compose calc-atm freshChainSimps*)
moreover from ‹$x \sharp p$› ‹$y \sharp p$› ‹$x \sharp N'$› ‹$(p \cdot (yvec1@yvec2)) \sharp* ([(x, y)] \cdot P'')$› **have** $([(y, x)] \cdot p \cdot (yvec1@yvec2)) \sharp* ([(y, x)] \cdot [(x, y)] \cdot P'')$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
then have $(((y, x)\#p) \cdot (yvec1@yvec2)) \sharp* P''$ **by**(*simp add*: *name-swap*)
with ‹$x \sharp yvec1$› ‹$x \sharp yvec2$› ‹$x \neq y$› ‹$y \sharp yvec1$› ‹$y \sharp yvec2$› ‹$x \sharp p$› ‹$y \sharp p$› ‹$x \sharp P''$› *yvecEq*
have $bn(((y, x)\#p) \cdot \alpha) \sharp* P''$ **by**(*simp add*: *bnEqvt*[*symmetric*]) (*simp add*: *perm-compose calc-atm eqvts freshChainSimps*)
moreover from ‹$x \sharp p$› ‹$y \sharp p$› ‹$x \sharp A_P{}'$› ‹$(p \cdot (yvec1@yvec2)) \sharp* A_P{}'$› **have** $(p \cdot (yvec1@x\#yvec2)) \sharp* A_P{}'$
by(*simp add*: *eqvts freshChainSimps*)
then have $([(y, x)] \cdot p \cdot (yvec1@x\#yvec2)) \sharp* ([(y, x)] \cdot A_P{}')$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$x \sharp yvec1$› ‹$x \sharp yvec2$› ‹$x \neq y$› ‹$x \sharp A_P{}'$› ‹$y \sharp yvec1$› ‹$y \sharp yvec2$› ‹$x \sharp p$› ‹$y \sharp p$› ‹$y \sharp A_P{}'$› *yvecEq*
have $bn(((y, x)\#p) \cdot \alpha) \sharp* A_P{}'$ **by**(*simp add*: *bnEqvt*[*symmetric*]) (*simp add*: *perm-compose calc-atm eqvts freshChainSimps*)
moreover from ‹$x \sharp p$› ‹$y \sharp p$› ‹$x \sharp C'$› ‹$(p \cdot (yvec1@yvec2)) \sharp* C'$› **have** $(p \cdot (yvec1@x\#yvec2)) \sharp* C'$
by(*simp add*: *eqvts freshChainSimps*)
then have $([(y, x)] \cdot p \cdot (yvec1@x\#yvec2)) \sharp* ([(y, x)] \cdot C')$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$x \sharp yvec1$› ‹$x \sharp yvec2$› ‹$x \neq y$› ‹$x \sharp C'$› ‹$y \sharp yvec1$› ‹$y \sharp yvec2$› ‹$x \sharp p$› ‹$y \sharp p$› ‹$y \sharp C'$› *yvecEq*
have $bn(((y, x)\#p) \cdot \alpha) \sharp* C'$ **by**(*simp add*: *bnEqvt*[*symmetric*]) (*simp add*: *perm-compose calc-atm eqvts freshChainSimps*)
moreover from ‹$x \sharp p$› ‹$y \sharp p$› ‹$x \sharp X$› ‹$(p \cdot (yvec1@yvec2)) \sharp* X$› **have** $(p \cdot (yvec1@x\#yvec2)) \sharp* X$
by(*simp add*: *eqvts freshChainSimps*)
then have $([(y, x)] \cdot p \cdot (yvec1@x\#yvec2)) \sharp* ([(y, x)] \cdot X)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$x \sharp yvec1$› ‹$x \sharp yvec2$› ‹$x \neq y$› ‹$x \sharp X$› ‹$y \sharp yvec1$› ‹$y \sharp yvec2$› ‹$x \sharp p$› ‹$y \sharp p$› ‹*bn* $\alpha \sharp* X$› *yvecEq*
have $bn(((y, x)\#p) \cdot \alpha) \sharp* X$ **by**(*simp add*: *bnEqvt*[*symmetric*]) (*simp add*: *perm-compose calc-atm eqvts freshChainSimps*)
moreover from ‹$x \sharp p$› ‹$y \sharp p$› ‹$x \sharp V$› ‹$(p \cdot (yvec1@yvec2)) \sharp* V$› **have** $(p \cdot (yvec1@x\#yvec2)) \sharp* V$
by(*simp add*: *eqvts freshChainSimps*)
then have $([(y, x)] \cdot p \cdot (yvec1@x\#yvec2)) \sharp* ([(y, x)] \cdot V)$

by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$x \sharp yvec1$› ‹$x \sharp yvec2$› ‹$x \neq y$› ‹$x \sharp V$› ‹$y \sharp yvec1$› ‹$y \sharp yvec2$› ‹$x \sharp p$› ‹$y \sharp p$› ‹$bn\ \alpha \sharp* V$› *yvecEq*
have $bn(((y, x)\#p) \cdot \alpha) \sharp* V$ **by**(*simp add*: *bnEqvt*[*symmetric*]) (*simp add*: *perm-compose calc-atm eqvts freshChainSimps*)
moreover from ‹$x \sharp p$› ‹$y \sharp p$› ‹$x \sharp W$› ‹$(p \cdot (yvec1@yvec2)) \sharp* W$› **have** $(p \cdot (yvec1@x\#yvec2)) \sharp* W$
by(*simp add*: *eqvts freshChainSimps*)
then have $([(y, x)] \cdot p \cdot (yvec1@x\#yvec2)) \sharp* ([(y, x)] \cdot W)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$x \sharp yvec1$› ‹$x \sharp yvec2$› ‹$x \neq y$› ‹$x \sharp W$› ‹$y \sharp yvec1$› ‹$y \sharp yvec2$› ‹$x \sharp p$› ‹$y \sharp p$› ‹$bn\ \alpha \sharp* W$› *yvecEq*
have $bn(((y, x)\#p) \cdot \alpha) \sharp* W$ **by**(*simp add*: *bnEqvt*[*symmetric*]) (*simp add*: *perm-compose calc-atm eqvts freshChainSimps*)
moreover from ‹$x \sharp p$› ‹$y \sharp p$› ‹$x \sharp Y$› ‹$(p \cdot (yvec1@yvec2)) \sharp* Y$› **have** $(p \cdot (yvec1@x\#yvec2)) \sharp* Y$
by(*simp add*: *eqvts freshChainSimps*)
then have $([(y, x)] \cdot p \cdot (yvec1@x\#yvec2)) \sharp* ([(y, x)] \cdot Y)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$x \sharp yvec1$› ‹$x \sharp yvec2$› ‹$x \neq y$› ‹$x \sharp Y$› ‹$y \sharp yvec1$› ‹$y \sharp yvec2$› ‹$x \sharp p$› ‹$y \sharp p$› ‹$bn\ \alpha \sharp* Y$› *yvecEq*
have $bn(((y, x)\#p) \cdot \alpha) \sharp* Y$ **by**(*simp add*: *bnEqvt*[*symmetric*]) (*simp add*: *perm-compose calc-atm eqvts freshChainSimps*)
moreover from ‹$x \sharp p$› ‹$y \sharp p$› ‹$x \sharp Z$› ‹$(p \cdot (yvec1@yvec2)) \sharp* Z$› **have** $(p \cdot (yvec1@x\#yvec2)) \sharp* Z$
by(*simp add*: *eqvts freshChainSimps*)
then have $([(y, x)] \cdot p \cdot (yvec1@x\#yvec2)) \sharp* ([(y, x)] \cdot Z)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$x \sharp yvec1$› ‹$x \sharp yvec2$› ‹$x \neq y$› ‹$x \sharp Z$› ‹$y \sharp yvec1$› ‹$y \sharp yvec2$› ‹$x \sharp p$› ‹$y \sharp p$› ‹$bn\ \alpha \sharp* Z$› *yvecEq*
have $bn(((y, x)\#p) \cdot \alpha) \sharp* Z$ **by**(*simp add*: *bnEqvt*[*symmetric*]) (*simp add*: *perm-compose calc-atm eqvts freshChainSimps*)
moreover from ‹$(yvec1@yvec2) \sharp* A_P'$› ‹$y \sharp A_P'$› *yvecEq* **have** $bn\ \alpha \sharp* A_P'$
by *simp*
moreover from ‹$A_P' \sharp* ([(x, y)] \cdot N')$› **have** $([(x, y)] \cdot A_P') \sharp* ([(x, y)] \cdot [(x, y)] \cdot N')$
by(*simp only*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$x \sharp A_P'$› ‹$y \sharp A_P'$› **have** $A_P' \sharp* N'$ **by** *simp*
with ‹$A_P' \sharp* M$› ‹$(yvec1@yvec2) \sharp* A_P'$› ‹$y \sharp A_P'$› αeq **have** $A_P' \sharp* \alpha$ **by** *simp*
moreover then have $(((y, x)\#p) \cdot A_P') \sharp* (((y, x)\#p) \cdot \alpha)$
by(*simp only*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$x \sharp A_P'$› ‹$y \sharp A_P'$› *S* ‹$(yvec1@yvec2) \sharp* A_P'$› ‹$(p \cdot (yvec1@yvec2)) \sharp* A_P'$›
have $A_P' \sharp* (((y, x)\#p) \cdot \alpha)$ **by**(*simp add*: *eqvts*)
moreover from ‹$A_P' \sharp* ([(x, y)] \cdot P'')$› **have** $([(x, y)] \cdot A_P') \sharp* ([(x, y)] \cdot [(x, y)] \cdot P'')$
by(*simp only*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$x \sharp A_P'$› ‹$y \sharp A_P'$› **have** $A_P' \sharp* P''$ **by** *simp*

moreover from *yvecEq* αeq ‹$(p \cdot (yvec1@yvec2))$ $\sharp*$ $(yvec1@yvec2)$› ‹$y \sharp p$› ‹$x \sharp \alpha$› S ‹$(p \cdot (yvec1@yvec2))$ $\sharp*$ M›‹$(p \cdot (yvec1@yvec2))$ $\sharp*$ $([(x, y)] \cdot N')$› ‹$y \sharp yvec1$›‹$y \sharp yvec2$› ‹$x \sharp p$›
have $bn(((y, x)\#p) \cdot \alpha)$ $\sharp*$ α
apply(*simp add*: *eqvts del*: *set-append*)
apply(*intro conjI*)
apply(*simp add*: *perm-compose eqvts del*: *set-append*)
apply(*simp add*: *perm-compose freshChainSimps*(6) *calc-atm eqvts del*: *set-append*)
apply(*simp add*: *perm-compose eqvts del*: *set-append*)
apply(*simp add*: *perm-compose eqvts swapStarFresh del*: *set-append*)
apply(*simp add*: *perm-compose freshChainSimps*(6) *calc-atm eqvts del*: *set-append*)
apply(*simp add*: *perm-compose freshChainSimps*(6) *calc-atm eqvts del*: *set-append*)
apply(*simp add*: *perm-compose freshChainSimps*(6) *calc-atm eqvts del*: *set-append*)
apply(*simp add*: *perm-compose freshChainSimps*(6) *swapStarFresh calc-atm eqvts del*: *set-append*)
apply(*simp add*: *perm-compose freshChainSimps*(6) *calc-atm eqvts del*: *set-append*)
apply(*subst pt-fresh-star-bij*[*symmetric*, *OF pt-name-inst*, *OF at-name-inst*, **where** *pi*=[(*x*, *y*)]])
apply(*simp add*: *perm-compose freshChainSimps*(6) *calc-atm eqvts del*: *set-append*)
apply(*simp add*: *perm-compose freshChainSimps*(6) *calc-atm eqvts del*: *set-append*)
apply(*subst pt-fresh-star-bij*[*symmetric*, *OF pt-name-inst*, *OF at-name-inst*, **where** *pi*=[(*x*, *y*)]])
by(*simp add*: *perm-compose freshChainSimps*(6) *calc-atm eqvts del*: *set-append*)
moreover note ‹A_P' $\sharp*$ C› ‹A_P' $\sharp*$ V› ‹A_P' $\sharp*$ W› ‹A_P' $\sharp*$ X› ‹A_P' $\sharp*$ Y› ‹A_P' $\sharp*$ Z› ‹*distinct* A_P'›

ultimately show *?case*
by(*elim cBrOpen*)
next
case(*cScope* Ψ P α P' x A_P Ψ_P C C' α' P'' V W X Y Z)
from ‹$\alpha \prec (\!|\nu x|\!) P' = \alpha' \prec P''$› ‹$x \sharp \alpha$› ‹$x \sharp \alpha'$›
obtain P''' **where** $\alpha \prec P' = \alpha' \prec P'''$ **and** $P'' = (\!|\nu x|\!) P'''$
apply(*cases rule*: *actionCases*[**where** α=α])
apply(*simp-all add*: *residualInject*)
apply (*metis bn.simps*(3) *boundOutputScopeDest*)
by (*metis bn.simps*(4) *boundOutputScopeDest*)
then obtain p Ψ' A_P' Ψ_P' **where** S: *set* $p \subseteq$ *set*$(bn\ \alpha') \times$ *set*$(bn(p \cdot \alpha'))$ **and** *PeqP'*: $((p \cdot \Psi_P) \otimes \Psi') \simeq \Psi_P'$
and *distinctPerm* p **and** $bn(p \cdot \alpha')$ $\sharp*$ C' **and** *FrP'*: *extractFrame* $P''' = \langle A_P', \Psi_P' \rangle$
and A_P' $\sharp*$ P''' **and** A_P' $\sharp*$ α' **and** A_P' $\sharp*$ $(p \cdot \alpha')$ **and** A_P' $\sharp*$ C **and** *distinct* A_P'

and $bn(p \cdot \alpha')$ $\sharp*$ P''' **and** $A_P{}'$ $\sharp*$ V **and** $A_P{}'$ $\sharp*$ W **and** $A_P{}'$ $\sharp*$ $(x\#X)$ **and** $A_P{}'$ $\sharp*$ Y **and** $bn(p \cdot \alpha')$ $\sharp*$ α'
and $A_P{}'$ $\sharp*$ Z **and** $bn(p \cdot \alpha')$ $\sharp*$ V **and** $bn(p \cdot \alpha')$ $\sharp*$ W **and** $bn(p \cdot \alpha')$ $\sharp*$ $(x\#X)$ **and** $bn(p \cdot \alpha')$ $\sharp*$ Y
and $bn(p \cdot \alpha')$ $\sharp*$ Z **using** *cScope*
by(*elim cScope*) (*assumption* | *simp*)+
from ‹$A_P{}'$ $\sharp*$ $(x\#X)$› **have** $x \sharp A_P{}'$ **and** $A_P{}'$ $\sharp*$ X **by** *simp*+
from ‹$bn(p \cdot \alpha')$ $\sharp*$ $(x\#X)$› **have** $x \sharp bn(p \cdot \alpha')$ **and** $bn(p \cdot \alpha')$ $\sharp*$ X **by** *simp*+

note *S PeqP'* ‹*distinctPerm p*› ‹$bn(p \cdot \alpha')$ $\sharp*$ C'›
moreover from *FrP'* ‹$P'' = (\!|\nu x|\!)P'''$› **have** *extractFrame* $P'' = \langle(x\#A_P{}'), \Psi_P{}'\rangle$ **by** *simp*
moreover from ‹$A_P{}'$ $\sharp*$ P'''› ‹$P'' = (\!|\nu x|\!)P'''$› ‹$x \sharp A_P{}'$› **have** $(x\#A_P{}')$ $\sharp*$ P'' **by**(*simp add*: *abs-fresh*)
moreover from ‹$A_P{}'$ $\sharp*$ α'› ‹$A_P{}'$ $\sharp*$ C› ‹$x \sharp \alpha'$› ‹$x \sharp C$› **have** $(x\#A_P{}')$ $\sharp*$ α' **and** $(x\#A_P{}')$ $\sharp*$ C **by** *simp*+
moreover note ‹$bn(p \cdot \alpha')$ $\sharp*$ α'›
moreover from ‹$bn(p \cdot \alpha')$ $\sharp*$ P'''› ‹$P'' = (\!|\nu x|\!)P'''$› ‹$x \sharp bn(p \cdot \alpha')$› **have** $bn(p \cdot \alpha')$ $\sharp*$ P'' **by** *simp*
moreover from ‹$A_P{}'$ $\sharp*$ α'› ‹$x \sharp \alpha'$› **have** $(x\#A_P{}')$ $\sharp*$ α' **by** *simp*
moreover from ‹$A_P{}'$ $\sharp*$ $(p \cdot \alpha')$› ‹$x \sharp \alpha'$› *S* ‹$x \sharp bn(p \cdot \alpha')$› **have** $(x\#A_P{}')$ $\sharp*$ $(p \cdot \alpha')$
by(*simp add*: *subjectEqvt*[*symmetric*] *bnEqvt*[*symmetric*] *objectEqvt*[*symmetric*] *freshChainSimps*)
moreover from ‹$A_P{}'$ $\sharp*$ V› ‹$x \sharp V$› **have** $(x\#A_P{}')$ $\sharp*$ V **by** *simp*+
moreover from ‹$A_P{}'$ $\sharp*$ W› ‹$x \sharp W$› **have** $(x\#A_P{}')$ $\sharp*$ W **by** *simp*+
moreover from ‹$A_P{}'$ $\sharp*$ X› ‹$x \sharp X$› **have** $(x\#A_P{}')$ $\sharp*$ X **by** *simp*+
moreover from ‹$A_P{}'$ $\sharp*$ Y› ‹$x \sharp Y$› **have** $(x\#A_P{}')$ $\sharp*$ Y **by** *simp*+
moreover from ‹$A_P{}'$ $\sharp*$ Z› ‹$x \sharp Z$› **have** $(x\#A_P{}')$ $\sharp*$ Z **by** *simp*+
moreover note ‹$bn(p \cdot \alpha')$ $\sharp*$ V› ‹$bn(p \cdot \alpha')$ $\sharp*$ W› ‹$bn(p \cdot \alpha')$ $\sharp*$ X› ‹$bn(p \cdot \alpha')$ $\sharp*$ Y› ‹$bn(p \cdot \alpha')$ $\sharp*$ Z›
moreover from ‹*distinct* $A_P{}'$› ‹$x \sharp A_P{}'$› **have** *distinct*$(x\#A_P{}')$ **by** *simp*
ultimately show *?case* **by**(*elim cScope*)
next
case(*cBang* Ψ P A_P Ψ_P C C' α P' V W X Y Z)
then obtain p Ψ' $A_P{}'$ $\Psi_P{}'$ **where** *S*: *set* $p \subseteq set(bn\ \alpha) \times set(bn(p \cdot \alpha))$
and *FrP'*: *extractFrame* $P' = \langle A_P{}', \Psi_P{}'\rangle$
and *PeqP'*: $(p \cdot (\Psi_P \otimes \mathbf{1})) \otimes \Psi' \simeq \Psi_P{}'$
and $A_P{}'$ $\sharp*$ C **and** $A_P{}'$ $\sharp*$ P' **and** $A_P{}'$ $\sharp*$ α **and** $A_P{}'$ $\sharp*$ $(p \cdot \alpha)$
and $A_P{}'$ $\sharp*$ V **and** $A_P{}'$ $\sharp*$ W **and** $A_P{}'$ $\sharp*$ X **and** $A_P{}'$ $\sharp*$ Y **and** $A_P{}'$ $\sharp*$ Z **and** *distinct* $A_P{}'$
and *distinctPerm p* **and** $(bn(p \cdot \alpha))$ $\sharp*$ α **and** $(bn(p \cdot \alpha))$ $\sharp*$ P'
and $(bn(p \cdot \alpha))$ $\sharp*$ C' **and** $(bn(p \cdot \alpha))$ $\sharp*$ V **and** $(bn(p \cdot \alpha))$ $\sharp*$ W **and** $(bn(p \cdot \alpha))$ $\sharp*$ X **and** $(bn(p \cdot \alpha))$ $\sharp*$ Y **and** $(bn(p \cdot \alpha))$ $\sharp*$ Z
apply −
by(*rule cBang*)(*assumption* | *simp* (*no-asm-use*))+
moreover from ‹$\Psi_P \simeq \mathbf{1}$› **have** $(p \cdot \Psi_P) \simeq (p \cdot \mathbf{1})$
by(*simp add*: *AssertionStatEqClosed*)

then have $(p \cdot \Psi_P) \simeq \mathbf{1}$ **by**(*simp add*: *permBottom*)
with *PeqP'* **have** $(\mathbf{1} \otimes \Psi') \simeq \Psi_P{}'$
by(*simp add*: *eqvts permBottom*) (*metis Identity AssertionStatEqTrans composition' Commutativity Associativity AssertionStatEqSym*)
ultimately show *?case* **using** *cBang*
by (*metis permBottom*)
qed

with *A* **have** *?thesis* **by** *blast*
}
moreover have *bn* α $\sharp*$ ([]::'*a list*) **and** *bn* α $\sharp*$ ([]::('*a action*) *list*) **and** *bn* α $\sharp*$ ([]::*name list*) **and** *bn* α $\sharp*$ ([]::'*b list*) **and** *bn* α $\sharp*$ ([]::('*a*, '*b*, '*c*) *psi list*)
and A_P $\sharp*$ ([]::'*a list*) **and** A_P $\sharp*$ ([]::('*a action*) *list*) **and** A_P $\sharp*$ ([]::*name list*) **and** A_P $\sharp*$ ([]::'*b list*) **and** A_P $\sharp*$ ([]::('*a*, '*b*, '*c*) *psi list*)
by *simp+*
ultimately show *?thesis* **by** *blast*
qed

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* :: '*f*::*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 Ψ' $A_P{}'$ $\Psi_P{}'$ **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 −
assume *A*: $\bigwedge A_P{}'\ \Psi_P{}'\ \Psi'$.
$[\![$*extractFrame P'* $= \langle A_P{}', \Psi_P{}' \rangle$; $\Psi_P \otimes \Psi' \simeq \Psi_P{}'$; $A_P{}' \sharp* C$; $A_P{}' \sharp* P'$; *distinct* $A_P{}'$$]\!]$
$\Longrightarrow$ *thesis*

from ‹$\Psi \rhd P \longmapsto \tau \prec P'$› ‹$A_P \sharp* P$› **have** $A_P \sharp* P'$ **by**(*rule tauFreshChainDerivative*)

{
fix *X* :: *name list*
and *Y* :: '*b list*
and *Z* :: ('*a*, '*b*, '*c*) *psi list*

assume $A_P \sharp* X$

and $A_P \sharp* Y$
and $A_P \sharp* Z$

with *assms* ‹$A_P \sharp* P'$› **obtain** $\Psi'\ A_P'\ \Psi_P'$ **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** $A_P' \sharp* X$ **and** $A_P' \sharp* Y$ **and** $A_P' \sharp* Z$ **and** *distinct* A_P'
proof(*nominal-induct avoiding*: $C\ X\ Y\ Z$ *arbitrary*: *thesis rule*: *tauFrameInduct*)
case(*cAlpha* $\Psi\ P\ P'\ A_P\ \Psi_P\ p\ C\ X\ Y\ Z$)
then obtain $\Psi'\ A_P'\ \Psi_P'$ **where** *FrP'*: *extractFrame* $P' = \langle A_P', \Psi_P' \rangle$ **and** $\Psi_P \otimes \Psi' \simeq \Psi_P'$ **and** *distinct* A_P'
and $A_P' \sharp* C$ **and** $A_P' \sharp* P'$ **and** $A_P' \sharp* X$ **and** $A_P' \sharp* Y$ **and** $A_P' \sharp* Z$
by *metis*

have *S*: *set* $p \subseteq$ *set* $A_P \times$ *set*$(p \cdot A_P)$ **by** *fact*

from *FrP'* **have** $(p \cdot$ *extractFrame* $P') = p \cdot \langle A_P', \Psi_P' \rangle$ **by** *simp*
with ‹$A_P \sharp* P'$› ‹$(p \cdot A_P) \sharp* P'$› *S* **have** *extractFrame* $P' = \langle (p \cdot A_P'), (p \cdot \Psi_P') \rangle$ **by**(*simp add*: *eqvts*)
moreover from ‹$\Psi_P \otimes \Psi' \simeq \Psi_P'$› **have** $(p \cdot (\Psi_P \otimes \Psi')) \simeq (p \cdot \Psi_P')$ **by**(*rule AssertionStatEqClosed*)
then have $(p \cdot \Psi_P) \otimes (p \cdot \Psi') \simeq (p \cdot \Psi_P')$ **by**(*simp add*: *eqvts*)
moreover from ‹$A_P' \sharp* C$› **have** $(p \cdot A_P') \sharp* (p \cdot C)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$A_P \sharp* C$› ‹$(p \cdot A_P) \sharp* C$› *S* **have** $(p \cdot A_P') \sharp* C$ **by** *simp*
moreover from ‹$A_P' \sharp* P'$› **have** $(p \cdot A_P') \sharp* (p \cdot P')$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$A_P \sharp* P'$› ‹$(p \cdot A_P) \sharp* P'$› *S* **have** $(p \cdot A_P') \sharp* P'$ **by** *simp*
moreover from ‹$A_P' \sharp* X$› **have** $(p \cdot A_P') \sharp* (p \cdot X)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$A_P \sharp* X$› ‹$(p \cdot A_P) \sharp* X$› *S* **have** $(p \cdot A_P') \sharp* X$ **by** *simp*
moreover from ‹$A_P' \sharp* Y$› **have** $(p \cdot A_P') \sharp* (p \cdot Y)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$A_P \sharp* Y$› ‹$(p \cdot A_P) \sharp* Y$› *S* **have** $(p \cdot A_P') \sharp* Y$ **by** *simp*
moreover from ‹$A_P' \sharp* Z$› **have** $(p \cdot A_P') \sharp* (p \cdot Z)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$A_P \sharp* Z$› ‹$(p \cdot A_P) \sharp* Z$› *S* **have** $(p \cdot A_P') \sharp* Z$ **by** *simp*
moreover from ‹*distinct* A_P'› **have** *distinct*$(p \cdot A_P')$ **by** *simp*
ultimately show *?case* **by**(*rule cAlpha*)
next
case(*cCase* $\Psi\ P\ P'\ \varphi\ Cs\ A_P\ \Psi_P\ C\ B\ Y\ Z$ *thesis*)
then obtain $\Psi'\ A_P'\ \Psi_P'$ **where** *FrP'*: *extractFrame* $P' = \langle A_P', \Psi_P' \rangle$
and $\Psi_P \otimes \Psi' \simeq \Psi_P'$ **and** *distinct* A_P'
and $A_P' \sharp* C$ **and** $A_P' \sharp* P'$
and $A_P' \sharp* B$ **and** $A_P' \sharp* Y$ **and** $A_P' \sharp* Z$
apply −
by(*rule cCase*) (*assumption* | *simp* (*no-asm-use*))+
with ‹$\Psi_P \simeq \mathbf{1}$› ‹$\Psi_P \otimes \Psi' \simeq \Psi_P'$› **have** $\mathbf{1} \otimes \Psi' \simeq \Psi_P'$
by(*metis Identity AssertionStatEqTrans composition' Commutativity Associativity AssertionStatEqSym*)

then show *?case* **using** *FrP'* ‹A_P' ♯* P'› ‹A_P' ♯* C› ‹A_P' ♯* B› ‹A_P' ♯* Y› ‹A_P' ♯* Z› ‹*distinct* A_P'› **using** *cCase*
by *force*
next
case(*cPar1* Ψ Ψ_Q P P' A_Q Q A_P Ψ_P C X Y Z)
moreover from ‹A_P ♯* X› ‹A_P ♯* A_Q› ‹A_P ♯* Y› ‹A_P ♯* Ψ_Q› ‹A_P ♯* Q› ‹A_P ♯* Z›
have A_P ♯* (X@A_Q) **and** A_P ♯* (Ψ_Q#Y) **and** A_P ♯* (Q#Z)
by *simp+*
ultimately obtain Ψ' A_P' Ψ_P' **where** *FrP'*: *extractFrame* P' = ⟨A_P', Ψ_P'⟩ **and** *distinct* A_P'
and Ψ_P ⊗ Ψ' ≃ Ψ_P' **and** A_P ♯* P **and** A_P ♯* C **and** A_P' ♯* C **and** A_P' ♯* P'
and A_P' ♯* (X@A_Q) **and** A_P' ♯* (Ψ_Q#Y) **and** A_P' ♯* (Q#Z)
by *metis*

then have A_P' ♯* X **and** A_P' ♯* A_Q **and** A_P' ♯* Y **and** A_P' ♯* Ψ_Q **and** A_P' ♯* Q **and** A_P' ♯* Z
by *simp+*

from ‹A_P' ♯* A_Q› ‹A_Q ♯* P'› *FrP'* **have** A_Q ♯* Ψ_P' **by**(*force dest*: *extractFrameFreshChain*)
with ‹A_P' ♯* Ψ_Q› ‹A_P' ♯* A_Q› ‹*extractFrame* Q = ⟨A_Q, Ψ_Q⟩› *FrP'*
have *extractFrame*(P' ‖ Q) = ⟨(A_P'@A_Q), Ψ_P' ⊗ Ψ_Q⟩ **by** *simp*

moreover from ‹Ψ_P ⊗ Ψ' ≃ Ψ_P'›**have** (Ψ_P ⊗ Ψ_Q) ⊗ Ψ' ≃ Ψ_P' ⊗ Ψ_Q
by(*metis Associativity Commutativity Composition AssertionStatEqTrans AssertionStatEqSym*)

moreover from ‹A_P' ♯* C› ‹A_Q ♯* C› **have** (A_P'@A_Q) ♯* C **by** *simp*
moreover from ‹A_P' ♯* P'› ‹A_Q ♯* P'› ‹A_P' ♯* Q› ‹A_Q ♯* Q› **have** (A_P'@A_Q) ♯* (P' ‖ Q) **by** *simp*
moreover from ‹A_P' ♯* X› ‹A_Q ♯* X› **have** (A_P'@A_Q) ♯* X **by** *simp*
moreover from ‹A_P' ♯* Y› ‹A_Q ♯* Y› **have** (A_P'@A_Q) ♯* Y **by** *simp*
moreover from ‹A_P' ♯* Z› ‹A_Q ♯* Z› **have** (A_P'@A_Q) ♯* Z **by** *simp*
moreover from ‹A_P' ♯* A_Q› ‹*distinct* A_P'› ‹*distinct* A_Q› **have** *distinct*(A_P'@A_Q) **by** *simp*
ultimately show *?case* **by**(*rule cPar1*)
next
case(*cPar2* Ψ Ψ_P Q Q' A_P P A_Q Ψ_Q C X Y Z)
moreover from ‹A_Q ♯* X› ‹A_P ♯* A_Q› ‹A_Q ♯* Y› ‹A_Q ♯* Ψ_P› ‹A_Q ♯* P› ‹A_Q ♯* Z›
have A_Q ♯* (X@A_P) **and** A_Q ♯* (Ψ_P#Y) **and** A_Q ♯* (P#Z)
by(*simp add*: *freshChainSimps*)+
ultimately obtain Ψ' A_Q' Ψ_Q' **where** *FrQ'*: *extractFrame* Q' = ⟨A_Q', Ψ_Q'⟩ **and** *distinct* A_Q'
and Ψ_Q ⊗ Ψ' ≃ Ψ_Q'**and** A_Q' ♯* C **and** A_Q' ♯* Q'
and A_Q' ♯* (X@A_P) **and** A_Q' ♯* (Ψ_P#Y) **and** A_Q' ♯* (P#Z)
by *metis*

then have $A_Q{}' \sharp* X$ **and** $A_Q{}' \sharp* A_P$ **and** $A_Q{}' \sharp* Y$ **and** $A_Q{}' \sharp* \Psi_P$ **and** $A_Q{}' \sharp* P$ **and** $A_Q{}' \sharp* Z$
by *simp+*

from ‹$A_Q{}' \sharp* A_P$› ‹$A_P \sharp* Q'$› FrQ' **have** $A_P \sharp* \Psi_Q{}'$ **by**(*force dest*: *extractFrameFreshChain*)
with ‹$A_Q{}' \sharp* \Psi_P$› ‹$A_Q{}' \sharp* A_P$› ‹*extractFrame* $P = \langle A_P, \Psi_P\rangle$› FrQ'
have *extractFrame*$(P \parallel Q') = \langle(A_P@A_Q{}'), \Psi_P \otimes \Psi_Q{}'\rangle$ **by** *simp*

moreover from ‹$\Psi_Q \otimes \Psi' \simeq \Psi_Q{}'$›**have** $(\Psi_P \otimes \Psi_Q) \otimes \Psi' \simeq \Psi_P \otimes \Psi_Q{}'$
by(*metis Associativity Commutativity Composition AssertionStatEqTrans AssertionStatEqSym*)
moreover from ‹$A_P \sharp* C$› ‹$A_Q{}' \sharp* C$› **have** $(A_P@A_Q{}') \sharp* C$ **by** *simp*
moreover from ‹$A_P \sharp* P$› ‹$A_Q{}' \sharp* P$› ‹$A_P \sharp* Q'$› ‹$A_Q{}' \sharp* Q'$› **have** $(A_P@A_Q{}') \sharp* (P \parallel Q')$ **by** *simp*
moreover from ‹$A_P \sharp* X$› ‹$A_Q{}' \sharp* X$› **have** $(A_P@A_Q{}') \sharp* X$ **by** *simp*
moreover from ‹$A_P \sharp* Y$› ‹$A_Q{}' \sharp* Y$› **have** $(A_P@A_Q{}') \sharp* Y$ **by** *simp*
moreover from ‹$A_P \sharp* Z$› ‹$A_Q{}' \sharp* Z$› **have** $(A_P@A_Q{}') \sharp* Z$ **by** *simp*
moreover from ‹$A_Q{}' \sharp* A_P$› ‹*distinct* A_P› ‹*distinct* $A_Q{}'$› **have** *distinct*$(A_P@A_Q{}')$ **by** *simp*
ultimately show *?case* **by**(*rule cPar2*)
next
case(*cComm1* $\Psi\ \Psi_Q\ P\ M\ N\ P'\ A_P\ \Psi_P\ Q\ K\ xvec\ Q'\ A_Q\ C\ X\ Y\ Z$)
have *PTrans*: $\Psi \otimes \Psi_Q \rhd P \longmapsto M\langle\!|N|\!\rangle \prec P'$ **and** *QTrans*: $\Psi \otimes \Psi_P \rhd Q \longmapsto K\langle\!|\nu*xvec|\!\rangle\langle N\rangle \prec Q'$ **by** *fact+*
from *PTrans* ‹*extractFrame* $P = \langle A_P, \Psi_P\rangle$› ‹*distinct* A_P› ‹$A_P \sharp* P$› ‹$A_P \sharp* N$› ‹$A_P \sharp* M$›
‹$A_P \sharp* Q'$› ‹$A_P \sharp* C$› ‹$A_P \sharp* X$› ‹$A_P \sharp* Y$› ‹$A_P \sharp* Z$› ‹$A_P \sharp* A_Q$› ‹$A_P \sharp* xvec$›
obtain $\Psi'\ A_P{}'\ \Psi_P{}'$ **where** *FrP'*: *extractFrame* $P' = \langle A_P{}', \Psi_P{}'\rangle$ **and** *PeqP'*: $\Psi_P \otimes \Psi' \simeq \Psi_P{}'$
and $A_P{}' \sharp* Q'$ **and** $A_P{}' \sharp* C$ **and** $A_P{}' \sharp* X$ **and** $A_P{}' \sharp* Y$ **and** *distinct* $A_P{}'$
and $A_P{}' \sharp* Z$ **and** $A_P{}' \sharp* A_Q$ **and** $A_P{}' \sharp* xvec$ **and** $A_P{}' \sharp* P'$
by(*elim expandNonTauFrame*[**where** $C=(Q', C, X, Y, Z, A_Q, xvec)$ **and** $C'=(Q', C, X, Y, Z, A_Q, xvec)$]) *auto*
moreover from *QTrans* **have** *distinct xvec* **by**(*auto dest*: *boundOutputDistinct*)
from *QTrans* ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› ‹*distinct* A_Q› ‹$A_Q \sharp* P$› ‹$A_Q \sharp* xvec$› ‹$A_Q \sharp* K$› ‹$xvec \sharp* K$› ‹*distinct xvec*›
‹$A_P{}' \sharp* A_Q$› ‹$A_P{}' \sharp* xvec$› ‹$A_Q \sharp* Q$› ‹$xvec \sharp* Q$› ‹$A_Q \sharp* \Psi_P$› ‹$xvec \sharp* \Psi_P$› ‹$A_Q \sharp* N$›
‹$A_Q \sharp* C$› ‹$A_Q \sharp* X$› ‹$A_Q \sharp* Y$› ‹$A_Q \sharp* Z$› ‹$xvec \sharp* P$› ‹$xvec \sharp* C$› ‹$xvec \sharp* X$› ‹$xvec \sharp* Y$› ‹$xvec \sharp* Z$›
obtain $p\ \Psi''\ A_Q{}'\ \Psi_Q{}'$ **where** *S*: *set* $p \subseteq$ *set xvec* $\times$ *set*$(p \cdot xvec)$ **and** *QeqQ'*: $(p \cdot \Psi_Q) \otimes \Psi'' \simeq \Psi_Q{}'$ **and** *FrQ'*: *extractFrame* $Q' = \langle A_Q{}', \Psi_Q{}'\rangle$
and $A_Q{}' \sharp* A_P{}'$ **and** $A_Q{}' \sharp* C$ **and** $A_Q{}' \sharp* X$ **and** $A_Q{}' \sharp* Y$ **and** $A_Q{}' \sharp* Z$ **and** $A_Q{}' \sharp* P$ **and** $A_Q{}' \sharp* N$ **and** *distinct* $A_Q{}'$

and $(p \cdot xvec) \sharp* A_P'$ **and** $(p \cdot xvec) \sharp* C$ **and** $(p \cdot xvec) \sharp* X$ **and** $(p \cdot xvec) \sharp* Y$ **and** $(p \cdot xvec) \sharp* P$
and $(p \cdot xvec) \sharp* Z$ **and** $(p \cdot xvec) \sharp* N$ **and** $(p \cdot xvec) \sharp* \Psi_P$ **and** $(p \cdot xvec) \sharp* A_Q'$**and** $(p \cdot xvec) \sharp* Q'$
and *distinctPerm p* **and** $A_Q' \sharp* xvec$ **and** $A_Q' \sharp* Q'$
by(*elim expandNonTauFrame*[**where** $C=(P, C, X, Y, Z, A_P', \Psi_P)$ **and** $C'=(P, C, X, Y, Z, A_P', \Psi_P)$]) (*assumption* | *simp*)+

from *PTrans* $\langle A_Q' \sharp* P\rangle$ $\langle A_Q' \sharp* N\rangle$ $\langle (p \cdot xvec) \sharp* P\rangle$ $\langle (p \cdot xvec) \sharp* N\rangle$
have $A_Q' \sharp* P'$ **and** $(p \cdot xvec) \sharp* P'$ **by**(*force dest*: *inputFreshChainDerivative*)+
with *FrP'* $\langle A_Q' \sharp* A_P'\rangle$ $\langle (p \cdot xvec) \sharp* A_P'\rangle$ **have** $A_Q' \sharp* \Psi_P'$ **and** $(p \cdot xvec) \sharp* \Psi_P'$ **by**(*force dest*: *extractFrameFreshChain*)+
from *FrQ'* $\langle A_Q' \sharp* A_P'\rangle$ $\langle A_P' \sharp* Q'\rangle$ $\langle (p \cdot xvec) \sharp* A_Q'\rangle$ $\langle (p \cdot xvec) \sharp* Q'\rangle$ **have** $A_P' \sharp* \Psi_Q'$ **and** $(p \cdot xvec) \sharp* \Psi_Q'$
by(*force dest*: *extractFrameFreshChain*)+

have $extractFrame((\!|\nu*xvec|\!)(P' \parallel Q')) = \langle((p \cdot xvec)@A_P'@A_Q'), (p \cdot \Psi_P') \otimes (p \cdot \Psi_Q')\rangle$
proof −
from *FrP' FrQ'* $\langle A_P' \sharp* \Psi_Q'\rangle$ $\langle A_Q' \sharp* A_P'\rangle$ $\langle A_Q' \sharp* \Psi_P'\rangle$ **have** $extractFrame(P' \parallel Q') = \langle(A_P'@A_Q'), \Psi_P' \otimes \Psi_Q'\rangle$
by *simp*
then have $extractFrame((\!|\nu*xvec|\!)(P' \parallel Q')) = \langle(xvec@A_P'@A_Q'), \Psi_P' \otimes \Psi_Q'\rangle$
by(*induct xvec*) *auto*
moreover from $\langle (p \cdot xvec) \sharp* \Psi_P'\rangle$ $\langle (p \cdot xvec) \sharp* \Psi_Q'\rangle$ *S*
have $(\!|\nu*xvec|\!)((\!|\nu*(A_P'@A_Q')|\!)(FAssert\ (\Psi_P' \otimes \Psi_Q'))) = (\!|\nu*(p \cdot xvec)|\!)(p \cdot (\!|\nu*(A_P'@A_Q')|\!)(FAssert(\Psi_P' \otimes \Psi_Q')))$
by(*intro frameChainAlpha*) (*auto simp add*: *fresh-star-def frameResChainFresh*)
then have $(\!|\nu*xvec|\!)((\!|\nu*(A_P'@A_Q')|\!)(FAssert\ (\Psi_P' \otimes \Psi_Q'))) = (\!|\nu*(p \cdot xvec)|\!)((\!|\nu*(A_P'@A_Q')|\!)(FAssert((p \cdot \Psi_P') \otimes (p \cdot \Psi_Q'))))$
using $\langle A_P' \sharp* xvec\rangle$ $\langle (p \cdot xvec) \sharp* A_P'\rangle$ $\langle A_Q' \sharp* xvec\rangle$ $\langle (p \cdot xvec) \sharp* A_Q'\rangle$ *S*
by(*auto simp add*: *eqvts*)
ultimately show *?thesis*
by(*simp add*: *frameChainAppend*)
qed

moreover have $(\Psi_P \otimes \Psi_Q) \otimes ((p \cdot \Psi') \otimes (p \cdot \Psi'')) \simeq (p \cdot \Psi_P') \otimes (p \cdot \Psi_Q')$
proof −
have $(\Psi_P \otimes (p \cdot \Psi_Q)) \otimes (\Psi' \otimes \Psi'') \simeq (\Psi_P \otimes \Psi') \otimes ((p \cdot \Psi_Q) \otimes \Psi'')$
by(*metis Associativity Commutativity Composition AssertionStatEqTrans*)
moreover from *PeqP' QeqQ'* **have** $(\Psi_P \otimes \Psi') \otimes ((p \cdot \Psi_Q) \otimes \Psi'') \simeq \Psi_P' \otimes \Psi_Q'$
by(*metis Associativity Commutativity Composition AssertionStatEqTrans*)
ultimately have $(\Psi_P \otimes (p \cdot \Psi_Q)) \otimes (\Psi' \otimes \Psi'') \simeq \Psi_P' \otimes \Psi_Q'$
by(*metis AssertionStatEqTrans*)
then have $(p \cdot ((\Psi_P \otimes (p \cdot \Psi_Q)) \otimes (\Psi' \otimes \Psi''))) \simeq (p \cdot (\Psi_P' \otimes \Psi_Q'))$

by(*rule AssertionStatEqClosed*)
with ‹*xvec* ♯* Ψ_P› ‹(p ∙ *xvec*) ♯* Ψ_P› *S* ‹*distinctPerm p*› **show** *?thesis*
by(*simp add*: *eqvts*)
qed

moreover from ‹(p ∙ *xvec*) ♯* *C*› ‹$A_P{}'$ ♯* *C*› ‹$A_Q{}'$ ♯* *C*› **have** ((p ∙ *xvec*)@$A_P{}'$@$A_Q{}'$) ♯* *C* **by** *simp*
moreover from ‹(p ∙ *xvec*) ♯* *X*› ‹$A_P{}'$ ♯* *X*› ‹$A_Q{}'$ ♯* *X*› **have** ((p ∙ *xvec*)@$A_P{}'$@$A_Q{}'$) ♯* *X* **by** *simp*
moreover from ‹(p ∙ *xvec*) ♯* *Y*› ‹$A_P{}'$ ♯* *Y*› ‹$A_Q{}'$ ♯* *Y*› **have** ((p ∙ *xvec*)@$A_P{}'$@$A_Q{}'$) ♯* *Y* **by** *simp*
moreover from ‹(p ∙ *xvec*) ♯* *Z*› ‹$A_P{}'$ ♯* *Z*› ‹$A_Q{}'$ ♯* *Z*› **have** ((p ∙ *xvec*)@$A_P{}'$@$A_Q{}'$) ♯* *Z* **by** *simp*
moreover from ‹(p ∙ *xvec*) ♯* P'› ‹(p ∙ *xvec*) ♯* Q'› ‹$A_P{}'$ ♯* P'› ‹$A_P{}'$ ♯* Q' ›‹$A_Q{}'$ ♯* P'› ‹$A_Q{}'$ ♯* Q'›
have ((p ∙ *xvec*)@$A_P{}'$@$A_Q{}'$) ♯* ((⦇ν*xvec⦈)(P' ∥ Q')) **by**(*auto simp add*: *resChainFresh fresh-star-def*)
moreover from ‹(p ∙ *xvec*) ♯* $A_P{}'$› ‹(p ∙ *xvec*) ♯* $A_Q{}'$› ‹$A_Q{}'$ ♯* $A_P{}'$› ‹*distinct xvec*› ‹*distinct* $A_P{}'$› ‹*distinct* $A_Q{}'$›
have *distinct*((p ∙ *xvec*)@$A_P{}'$@$A_Q{}'$)
by *auto* (*simp add*: *name-list-supp fresh-star-def fresh-def*)+

ultimately show *?case* **using** *cComm1*
by *metis*
next
case(*cComm2* Ψ Ψ_Q P M *xvec* N P' A_P Ψ_P Q K Q' A_Q C X Y Z)
have *PTrans*: $\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P'$ **and** *QTrans*: $\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|N|\!) \prec Q'$ **by** *fact*+
from *PTrans* **have** *distinct xvec* **by**(*auto dest*: *boundOutputDistinct*)
from *PTrans* ‹*extractFrame* $P = \langle A_P, \Psi_P\rangle$› ‹*distinct* A_P› ‹A_P ♯* P› ‹A_P ♯* Q› ‹*xvec* ♯* Q› ‹*distinct xvec*› ‹*xvec* ♯* M›
‹A_P ♯* C› ‹A_P ♯* X› ‹A_P ♯* Y› ‹A_P ♯* Z› ‹A_P ♯* A_Q› ‹A_P ♯* *xvec*› ‹A_P ♯* Ψ_Q› ‹A_P ♯* M› ‹A_P ♯* N›
‹*xvec* ♯* P› ‹*xvec* ♯* C› ‹*xvec* ♯* X› ‹*xvec* ♯* Y› ‹*xvec* ♯* Z› ‹A_Q ♯* *xvec*› ‹*xvec* ♯* Ψ_Q›
obtain p Ψ' $A_P{}'$ $\Psi_P{}'$ **where** *S*: *set* $p \subseteq$ *set xvec* × *set*(p ∙ *xvec*)
and *FrP'*: *extractFrame* $P' = \langle A_P{}', \Psi_P{}'\rangle$ **and** *PeqP'*: (p ∙ Ψ_P) ⊗ $\Psi' \simeq \Psi_P{}'$ **and** *distinct* $A_P{}'$
and $A_P{}'$ ♯* C **and** $A_P{}'$ ♯* X **and** $A_P{}'$ ♯* Y **and** $A_P{}'$ ♯* N **and** $A_P{}'$ ♯* Q **and** (p ∙ *xvec*) ♯* Q
and $A_P{}'$ ♯* Z **and** $A_P{}'$ ♯* A_Q **and** $A_P{}'$ ♯* *xvec* **and** $A_P{}'$ ♯* P' **and** (p ∙ *xvec*) ♯* N **and** (p ∙ *xvec*) ♯* Ψ_Q
and (p ∙ *xvec*) ♯* $A_P{}'$ **and** (p ∙ *xvec*) ♯* C **and** (p ∙ *xvec*) ♯* X **and** (p ∙ *xvec*) ♯* A_Q
and (p ∙ *xvec*) ♯* Y **and** (p ∙ *xvec*) ♯* Z **and** (p ∙ *xvec*) ♯* P' **and** *distinctPerm* p
by(*elim expandNonTauFrame*[**where** C=(C, X, Y, Z, A_Q, Q, Ψ_Q) **and** C'=(C, X, Y, Z, A_Q, Q, Ψ_Q)]) (*assumption* | *simp*)+

from *QTrans* ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› ‹*distinct* A_Q› ‹A_Q ♯* Q› ‹A_Q ♯* *xvec*› ‹A_Q ♯* P'› ‹$(p \cdot xvec)$ ♯* A_Q›
‹$A_P{}'$ ♯* A_Q› ‹A_Q ♯* P› ‹A_Q ♯* Ψ_P› ‹A_Q ♯* C› ‹A_Q ♯* X› ‹A_Q ♯* Y› ‹A_Q ♯* Z› ‹A_Q ♯* N› ‹A_Q ♯* K›
obtain Ψ'' $A_Q{}'$ $\Psi_Q{}'$ **where** *QeqQ'*: $\Psi_Q \otimes \Psi'' \simeq \Psi_Q{}'$ **and** *FrQ'*: *extractFrame* $Q' = \langle A_Q{}', \Psi_Q{}'\rangle$ **and** *distinct* $A_Q{}'$
and $A_Q{}'$ ♯* *xvec* **and** $A_Q{}'$ ♯* Q' **and** $A_Q{}'$ ♯* *xvec* **and** $A_Q{}'$ ♯* P' **and** $A_Q{}'$ ♯* $(p \cdot xvec)$
and $A_Q{}'$ ♯* $A_P{}'$ **and** $A_Q{}'$ ♯* C **and** $A_Q{}'$ ♯* X **and** $A_Q{}'$ ♯* Y **and** $A_Q{}'$ ♯* Z **and** $A_Q{}'$ ♯* P
by(*elim expandNonTauFrame*[**where** C=(P, C, P', X, Y, Z, $A_P{}'$, *xvec*, $(p \cdot xvec)$, Ψ_P) **and** C'=(P, C, P', X, Y, Z, $A_P{}'$, *xvec*, $(p \cdot xvec)$, Ψ_P)]) (*assumption* | *simp*)+

from *QTrans* ‹$A_P{}'$ ♯* Q› ‹$A_P{}'$ ♯* N› ‹$(p \cdot xvec)$ ♯* Q› ‹$(p \cdot xvec)$ ♯* N›
have $A_P{}'$ ♯* Q' **and** $(p \cdot xvec)$ ♯* Q' **by**(*force dest*: *inputFreshChainDerivative*)+
with *FrQ'* ‹$A_Q{}'$ ♯* $A_P{}'$› ‹$A_Q{}'$ ♯* $(p \cdot xvec)$› **have** $A_P{}'$ ♯* $\Psi_Q{}'$ **and** $(p \cdot xvec)$ ♯* $\Psi_Q{}'$ **by**(*force dest*: *extractFrameFreshChain*)+
from *FrP'* ‹$A_Q{}'$ ♯* $A_P{}'$› ‹$A_Q{}'$ ♯* P'› ‹$(p \cdot xvec)$ ♯* $A_P{}'$› ‹$(p \cdot xvec)$ ♯* P'› **have** $A_Q{}'$ ♯* $\Psi_P{}'$ **and** $(p \cdot xvec)$ ♯* $\Psi_P{}'$
by(*force dest*: *extractFrameFreshChain*)+

have *extractFrame*($(\!|\nu*xvec|\!)(P' \parallel Q')$) = $\langle((p \cdot xvec)@A_P{}'@A_Q{}'), (p \cdot \Psi_P{}') \otimes (p \cdot \Psi_Q{}')\rangle$
proof −
from *FrP' FrQ'* ‹$A_P{}'$ ♯* $\Psi_Q{}'$› ‹$A_Q{}'$ ♯* $A_P{}'$› ‹$A_Q{}'$ ♯* $\Psi_P{}'$› **have** *extractFrame*($P' \parallel Q'$) = $\langle(A_P{}'@A_Q{}'), \Psi_P{}' \otimes \Psi_Q{}'\rangle$
by *simp*
then have *extractFrame*($(\!|\nu*xvec|\!)(P' \parallel Q')$) = $\langle(xvec@A_P{}'@A_Q{}'), \Psi_P{}' \otimes \Psi_Q{}'\rangle$
by(*induct xvec*) *auto*
moreover from ‹$(p \cdot xvec)$ ♯* $\Psi_P{}'$› ‹$(p \cdot xvec)$ ♯* $\Psi_Q{}'$› *S*
have $(\!|\nu*xvec|\!)((\!|\nu*(A_P{}'@A_Q{}')|\!)(FAssert\ (\Psi_P{}' \otimes \Psi_Q{}')))$ = $(\!|\nu*(p \cdot xvec)|\!)(p \cdot (\!|\nu*(A_P{}'@A_Q{}')|\!)(FAssert(\Psi_P{}' \otimes \Psi_Q{}')))$
by(*intro frameChainAlpha*) (*auto simp add*: *fresh-star-def frameResChainFresh*)
then have $(\!|\nu*xvec|\!)((\!|\nu*(A_P{}'@A_Q{}')|\!)(FAssert\ (\Psi_P{}' \otimes \Psi_Q{}')))$ = $(\!|\nu*(p \cdot xvec)|\!)((\!|\nu*(A_P{}'@A_Q{}')|\!)(FAssert((p \cdot \Psi_P{}') \otimes (p \cdot \Psi_Q{}'))))$
using ‹$A_P{}'$ ♯* *xvec*› ‹$(p \cdot xvec)$ ♯* $A_P{}'$› ‹$A_Q{}'$ ♯* *xvec*› ‹$A_Q{}'$ ♯* $(p \cdot xvec)$› *S*
by(*auto simp add*: *eqvts*)
ultimately show *?thesis*
by(*simp add*: *frameChainAppend*)
qed

moreover have $(\Psi_P \otimes \Psi_Q) \otimes ((p \cdot \Psi') \otimes (p \cdot \Psi'')) \simeq (p \cdot \Psi_P{}') \otimes (p \cdot \Psi_Q{}')$
proof −
have $((p \cdot \Psi_P) \otimes \Psi_Q) \otimes (\Psi' \otimes \Psi'') \simeq ((p \cdot \Psi_P) \otimes \Psi') \otimes (\Psi_Q \otimes \Psi'')$
by(*metis Associativity Commutativity Composition AssertionStatEqTrans*)

moreover from $PeqP'$ $QeqQ'$ **have** $((p \cdot \Psi_P) \otimes \Psi') \otimes (\Psi_Q \otimes \Psi'') \simeq \Psi_P{}' \otimes \Psi_Q{}'$
by(*metis Associativity Commutativity Composition AssertionStatEqTrans*)
ultimately have $((p \cdot \Psi_P) \otimes \Psi_Q) \otimes (\Psi' \otimes \Psi'') \simeq \Psi_P{}' \otimes \Psi_Q{}'$
by(*metis AssertionStatEqTrans*)
then have $(p \cdot ((p \cdot \Psi_P) \otimes \Psi_Q) \otimes (\Psi' \otimes \Psi'')) \simeq (p \cdot (\Psi_P{}' \otimes \Psi_Q{}'))$
by(*rule AssertionStatEqClosed*)
with ‹*xvec* $\sharp*$ Ψ_Q› ‹$(p \cdot xvec)$ $\sharp*$ Ψ_Q› S ‹*distinctPerm p*› **show** *?thesis*
by(*simp add*: *eqvts*)
qed

moreover from ‹$(p \cdot xvec)$ $\sharp*$ C› ‹$A_P{}'$ $\sharp*$ C› ‹$A_Q{}'$ $\sharp*$ C› **have** $((p \cdot xvec)@A_P{}'@A_Q{}')$ $\sharp*$ C **by** *simp*
moreover from ‹$(p \cdot xvec)$ $\sharp*$ X› ‹$A_P{}'$ $\sharp*$ X› ‹$A_Q{}'$ $\sharp*$ X› **have** $((p \cdot xvec)@A_P{}'@A_Q{}')$ $\sharp*$ X **by** *simp*
moreover from ‹$(p \cdot xvec)$ $\sharp*$ Y› ‹$A_P{}'$ $\sharp*$ Y› ‹$A_Q{}'$ $\sharp*$ Y› **have** $((p \cdot xvec)@A_P{}'@A_Q{}')$ $\sharp*$ Y **by** *simp*
moreover from ‹$(p \cdot xvec)$ $\sharp*$ Z› ‹$A_P{}'$ $\sharp*$ Z› ‹$A_Q{}'$ $\sharp*$ Z› **have** $((p \cdot xvec)@A_P{}'@A_Q{}')$ $\sharp*$ Z **by** *simp*
moreover from ‹$(p \cdot xvec)$ $\sharp*$ P'› ‹$(p \cdot xvec)$ $\sharp*$ Q'› ‹$A_P{}'$ $\sharp*$ P'› ‹$A_P{}'$ $\sharp*$ Q'› ‹$A_Q{}'$ $\sharp*$ P'› ‹$A_Q{}'$ $\sharp*$ Q'›
have $((p \cdot xvec)@A_P{}'@A_Q{}')$ $\sharp*$ $((\!|\nu*xvec|\!)(P' \parallel Q'))$ **by**(*auto simp add*: *resChainFresh fresh-star-def*)
moreover from ‹$(p \cdot xvec)$ $\sharp*$ $A_P{}'$› ‹ $A_Q{}'$ $\sharp*$ $(p \cdot xvec)$› ‹$A_Q{}'$ $\sharp*$ $A_P{}'$› ‹*distinct xvec*› ‹*distinct* $A_P{}'$› ‹*distinct* $A_Q{}'$›
have *distinct*$((p \cdot xvec)@A_P{}'@A_Q{}')$
by *auto* (*simp add*: *name-list-supp fresh-star-def fresh-def*)+

ultimately show *?case* **using** *cComm2* **by** *metis*
next
case(*cBrClose* Ψ P M *xvec* N P' A_P Ψ_P x C X Y Z)
from ‹$(x \# A_P)$ $\sharp*$ C› **have** A_P $\sharp*$ C **by** *simp*
from ‹$(x \# A_P)$ $\sharp*$ X› ‹x $\sharp$ A_P› **have** A_P $\sharp*$ $(x\#X)$ **by** *simp*
from ‹$(x \# A_P)$ $\sharp*$ Y› **have** A_P $\sharp*$ Y **by** *simp*
from ‹$(x \# A_P)$ $\sharp*$ Z› **have** A_P $\sharp*$ Z **by** *simp*
from ‹x $\sharp$ *xvec*› ‹*xvec* $\sharp*$ X› **have** *xvec* $\sharp*$ $(x\#X)$ **by** *simp*

have *PTrans*: $\Psi \rhd P \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec P'$ **by** *fact*+
from *PTrans* **have** *distinct xvec* **by**(*auto dest*: *boundOutputDistinct*)
from *PTrans* ‹*extractFrame* $P = \langle A_P, \Psi_P\rangle$› ‹*distinct* A_P› ‹*xvec* $\sharp*$ M› ‹A_P $\sharp*$ P› ‹A_P $\sharp*$ M› ‹A_P $\sharp*$ *xvec*› ‹A_P $\sharp*$ N› ‹A_P $\sharp*$ C›
‹A_P $\sharp*$ $(x\#X)$› ‹A_P $\sharp*$ Y› ‹A_P $\sharp*$ Z› ‹*xvec* $\sharp*$ P› ‹*xvec* $\sharp*$ C› ‹*xvec* $\sharp*$ $(x\#X)$› ‹*xvec* $\sharp*$ Y› ‹*xvec* $\sharp*$ Z›

obtain p Ψ' $A_P{}'$ $\Psi_P{}'$ **where** S: *set* $p \subseteq$ *set xvec* $\times$ *set*$(p \cdot xvec)$
and *FrP'*: *extractFrame* $P' = \langle A_P{}', \Psi_P{}'\rangle$ **and** *PeqP'*: $(p \cdot \Psi_P) \otimes \Psi' \simeq \Psi_P{}'$ **and** *distinct* $A_P{}'$
and $A_P{}'$ $\sharp*$ C **and** $A_P{}'$ $\sharp*$ $(x\#X)$ **and** $A_P{}'$ $\sharp*$ Y **and** $A_P{}'$ $\sharp*$ N
and $A_P{}'$ $\sharp*$ Z **and** $A_P{}'$ $\sharp*$ *xvec* **and** $A_P{}'$ $\sharp*$ P' **and** $(p \cdot xvec)$ $\sharp*$ *xvec* **and**

$(p \cdot xvec) \sharp* N$
and $(p \cdot xvec) \sharp* A_P{}'$ **and** $(p \cdot xvec) \sharp* C$ **and** $(p \cdot xvec) \sharp* (x\#X)$
and $(p \cdot xvec) \sharp* Y$ **and** $(p \cdot xvec) \sharp* Z$ **and** $(p \cdot xvec) \sharp* P'$ **and** *distinctPerm p*
by(*elim expandNonTauFrame*[**where** C=$(C,\ (x\#X),\ Y,\ Z)$ **and** C'=$(C,\ (x\#X),\ Y,\ Z)$]) (*assumption* | *simp*)+
then have $A_P{}' \sharp* X$ **and** $x \sharp A_P{}'$ **and** $(p \cdot xvec) \sharp* X$ **and** $x \sharp (p \cdot xvec)$ **by** *simp*+
from *FrP'* $\langle(p \cdot xvec) \sharp* A_P{}'\rangle$ $\langle(p \cdot xvec) \sharp* P'\rangle$ **have** $(p \cdot xvec) \sharp* \Psi_P{}'$
by (*metis extractFrameFreshChain freshFrameDest*)
from *S* $\langle A_P{}' \sharp* xvec\rangle$ $\langle(p \cdot xvec) \sharp* A_P{}'\rangle$ **have** $p \cdot A_P{}' = A_P{}'$ **by** *simp*

from $\langle(p \cdot xvec) \sharp* \Psi_P{}'\rangle$ *S*
have $(\!|\nu*xvec|\!)(\langle A_P{}', \Psi_P{}'\rangle) = (\!|\nu*(p \cdot xvec)|\!)(p \cdot \langle A_P{}', \Psi_P{}'\rangle)$
by(*intro frameChainAlpha*) (*auto simp add*: *fresh-star-def frameResChain-Fresh*)
then have $\langle(xvec@A_P{}'), \Psi_P{}'\rangle = (p \cdot \langle(xvec@A_P{}'), \Psi_P{}'\rangle)$
by (*metis frameChainAppend frameResChainEqvt*)
with $\langle p \cdot A_P{}' = A_P{}'\rangle$ **have** $\langle(xvec@A_P{}'), \Psi_P{}'\rangle = \langle((p \cdot xvec)@A_P{}'), (p \cdot \Psi_P{}')\rangle$
by(*simp add*: *eqvts*)
then have $\langle(x\#(xvec@A_P{}')), \Psi_P{}'\rangle = \langle(x\#((p \cdot xvec)@A_P{}')), (p \cdot \Psi_P{}')\rangle$
by *simp*

moreover from *FrP'* **have** *extractFrame* $((\!|\nu x|\!)((\!|\nu*xvec|\!)P')) = \langle(x\#(xvec@A_P{}')), \Psi_P{}'\rangle$
by (*metis extractFrameResChain extractFrame-extractFrame'-extractFrame''.simps frameChainAppend frameResChain.step*)

ultimately have *FrP'2*: *extractFrame* $((\!|\nu x|\!)((\!|\nu*xvec|\!)P')) = \langle(x\#((p \cdot xvec)@A_P{}')), (p \cdot \Psi_P{}')\rangle$
by *simp*

from *PeqP'* **have** $(p \cdot ((p \cdot \Psi_P) \otimes \Psi')) \simeq (p \cdot \Psi_P{}')$
by (*metis AssertionStatEqClosed*)
with $\langle$*distinctPerm p*$\rangle$ **have** *PeqP'2*: $\Psi_P \otimes (p \cdot \Psi') \simeq (p \cdot \Psi_P{}')$
by(*simp add*: *eqvts*)

note *FrP'2 PeqP'2*

moreover from $\langle x \sharp C\rangle$ $\langle(p \cdot xvec) \sharp* C\rangle$ $\langle A_P{}' \sharp* C\rangle$
have $(x\#((p \cdot xvec)@A_P{}')) \sharp* C$ **by** *simp*
moreover from $\langle(x \# A_P) \sharp* (\!|\nu x|\!)((\!|\nu*xvec|\!)P')\rangle$ $\langle A_P{}' \sharp* (x\#X)\rangle$ $\langle A_P{}' \sharp* P'\rangle$ $\langle A_P{}' \sharp* xvec\rangle$
$\langle x \sharp (p \cdot xvec)\rangle$ $\langle(p \cdot xvec) \sharp* xvec\rangle$ $\langle(p \cdot xvec) \sharp* P'\rangle$
have $(x\#((p \cdot xvec)@A_P{}')) \sharp* ((\!|\nu x|\!)((\!|\nu*xvec|\!)P'))$
by *simp*
moreover from $\langle x \sharp X\rangle$ $\langle(p \cdot xvec) \sharp* X\rangle$ $\langle A_P{}' \sharp* X\rangle$
have $(x\#((p \cdot xvec)@A_P{}')) \sharp* X$ **by** *simp*

moreover from ‹$x \sharp Y$› ‹$(p \cdot xvec) \sharp* Y$› ‹$A_P' \sharp* Y$›
have $(x\#((p \cdot xvec)@A_P')) \sharp* Y$ **by** *simp*
moreover from ‹$x \sharp Z$› ‹$(p \cdot xvec) \sharp* Z$› ‹$A_P' \sharp* Z$›
have $(x\#((p \cdot xvec)@A_P')) \sharp* Z$ **by** *simp*
moreover from ‹$x \sharp (p \cdot xvec)$› ‹$x \sharp A_P'$› ‹$(p \cdot xvec) \sharp* A_P'$›
‹*distinct* A_P'› ‹*distinct xvec*› ‹*distinctPerm p*›
have *distinct* $(x\#((p \cdot xvec)@A_P'))$ **by** *simp*
ultimately show *?case*
by(*rule cBrClose*)
next
case(*cScope* Ψ P P' x A_P Ψ_P C X Y Z)
then obtain Ψ' A_P' Ψ_P' **where** *FrP'*: *extractFrame* $P' = \langle A_P', \Psi_P' \rangle$ **and** *distinct* A_P'
and $\Psi_P \otimes \Psi' \simeq \Psi_P'$ **and** $A_P' \sharp* C$ **and** $A_P' \sharp* P'$
and $A_P' \sharp* (x\#X)$ **and** $A_P' \sharp* Y$ **and** $A_P' \sharp* Z$
using *cScope*(*4*)[**where** *ba*=$x\#X$] **by** (*metis freshStarAtom fresh-star-list-cons*(*1*))
from ‹$A_P' \sharp* (x\#X)$› **have** $x \sharp A_P'$ **and** $A_P' \sharp* X$ **by** *simp+*
moreover from *FrP'* **have** *extractFrame*($\langle\!|\nu x|\!\rangle P'$) = $\langle (x\#A_P'), \Psi_P' \rangle$ **by** *simp*
moreover note ‹$\Psi_P \otimes \Psi' \simeq \Psi_P'$›
moreover from ‹$x \sharp C$› ‹$A_P' \sharp* C$› **have** $(x\#A_P') \sharp* C$ **by** *simp*
moreover from ‹$A_P' \sharp* P'$› **have** $(x\#A_P') \sharp* (\langle\!|\nu x|\!\rangle P')$ **by**(*simp add*: *abs-fresh fresh-star-def*)
moreover from ‹$x \sharp X$› ‹$A_P' \sharp* X$› **have** $(x\#A_P') \sharp* X$ **by** *simp*
moreover from ‹$x \sharp Y$› ‹$A_P' \sharp* Y$› **have** $(x\#A_P') \sharp* Y$ **by** *simp*
moreover from ‹$x \sharp Z$› ‹$A_P' \sharp* Z$› **have** $(x\#A_P') \sharp* Z$ **by** *simp*
moreover from ‹$x \sharp A_P'$› ‹*distinct* A_P'› **have** *distinct*$(x\#A_P')$ **by** *simp*
ultimately show *?case* **by**(*elim cScope*)
next
case(*cBang* Ψ P P' A_P Ψ_P C B Y Z)
then obtain Ψ' A_P' Ψ_P' **where** *FrP'*: *extractFrame* $P' = \langle A_P', \Psi_P' \rangle$
and $(\Psi_P \otimes \mathbf{1}) \otimes \Psi' \simeq \Psi_P'$
and $A_P' \sharp* C$ **and** $A_P' \sharp* P'$ **and** *distinct* A_P'
and $A_P' \sharp* B$ **and** $A_P' \sharp* Y$ **and** $A_P' \sharp* Z$
using *cBang* **by** (*metis psiFreshVec*(*4*) *psiFreshVec*(*7*))
with ‹$\Psi_P \simeq \mathbf{1}$› ‹$(\Psi_P \otimes \mathbf{1}) \otimes \Psi' \simeq \Psi_P'$› **have** $\mathbf{1} \otimes \Psi' \simeq \Psi_P'$
by(*metis Identity AssertionStatEqTrans composition' Commutativity Associativity AssertionStatEqSym*)
then show *?case* **using** *FrP'* ‹$A_P' \sharp* P'$› ‹$A_P' \sharp* C$› ‹$A_P' \sharp* B$› ‹$A_P' \sharp* Y$› ‹$A_P' \sharp* Z$› ‹*distinct* A_P'›
by(*elim cBang*)
qed
with *A* **have** *?thesis* **by** *simp*
}
moreover have $A_P \sharp* ([]::name\ list)$ **and** $A_P \sharp* ([]::'b\ list)$ **and** $A_P \sharp* ([]::('a, 'b, 'c)\ psi\ list)$ **by** *simp+*
ultimately show *?thesis* **by** *blast*
qed

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 :: $'f$::*fs-name*
and C' :: $'g$::*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* α) $\times$ *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'
using *assms*
apply(*cases* α=τ)
by(*auto intro*: *expandTauFrame*[**where** C=C] *expandNonTauFrame*[**where** C=C **and** C'=C'])

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 $F \hookrightarrow_F I$
using *assms*
by(*auto simp add*: *FrameStatEq-def*) (*blast intro*: *FrameStatImpTrans*)

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 $A_F \sharp* \mathit{subject}\ \alpha$
and $A_G \sharp* \mathit{subject}\ \alpha$
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 $\Psi_G \rhd P \longmapsto \alpha \prec P'$
using *assms*
proof(*nominal-induct* Ψ_F P *Rs*==$\alpha \prec P'$ A_P Ψ_P *avoiding*: α P' Ψ_G A_F A_G *rule*: *semanticsFrameInduct*)
case(*cAlpha* Ψ_F P A_P Ψ_P p α P' Ψ_G A_F A_G)
from ‹$\langle A_F, \Psi_F \otimes (p \cdot \Psi_P)\rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes (p \cdot \Psi_P)\rangle$›
have $(p \cdot (\langle A_F, \Psi_F \otimes (p \cdot \Psi_P)\rangle)) \hookrightarrow_F (p \cdot (\langle A_G, \Psi_G \otimes (p \cdot \Psi_P)\rangle))$
by(*rule FrameStatImpClosed*)
with ‹$A_P \sharp* A_F$› ‹$(p \cdot A_P) \sharp* A_F$› ‹$A_P \sharp* \Psi_F$› ‹$(p \cdot A_P) \sharp* \Psi_F$› ‹$A_P \sharp* A_G$› ‹$(p \cdot A_P) \sharp* A_G$› ‹$A_P \sharp* \Psi_G$› ‹$(p \cdot A_P) \sharp* \Psi_G$›
‹*distinctPerm* p› ‹*set* $p \subseteq$ *set* $A_P \times$ *set* $(p \cdot A_P)$› **have** $\langle A_F, \Psi_F \otimes \Psi_P \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \rangle$
by(*simp add*: *eqvts*)
with *cAlpha* **show** *?case* **by** *force*
next
case(*cInput* Ψ_F M K *xvec* N *Tvec* P α P' Ψ_G A_F A_G)
from *cInput* **have** $A_F \sharp* K$ **and** $A_G \sharp* K$ **by**(*auto simp add*: *residualInject*)

from ‹$A_F \sharp* (M(\!|\lambda* \mathit{xvec}\ N|\!).P)$› ‹$A_G \sharp* (M(\!|\lambda* \mathit{xvec}\ N|\!).P)$› **have** $A_F \sharp* M$ **and** $A_G \sharp* M$ **by** *simp*+
from ‹$\Psi_F \vdash M \leftrightarrow K$›
have $\Psi_F \otimes \mathbf{1} \vdash M \leftrightarrow K$
by(*blast intro*: *statEqEnt Identity AssertionStatEqSym*)
with ‹$A_F \sharp* M$› ‹$A_F \sharp* K$›

have $(\langle A_F,\ \Psi_F \otimes \mathbf{1}\rangle) \vdash_F M \leftrightarrow K$
by(*force intro*: *frameImpI*)
with $\langle\langle A_F,\ \Psi_F \otimes \mathbf{1}\rangle \hookrightarrow_F \langle A_G,\ \Psi_G \otimes \mathbf{1}\rangle\rangle$
have $(\langle A_G,\ \Psi_G \otimes \mathbf{1}\rangle) \vdash_F M \leftrightarrow K$
by(*simp add*: *FrameStatEq-def FrameStatImp-def*)
with $\langle A_G \sharp* M\rangle$ $\langle A_G \sharp* K\rangle$
have $\Psi_G \otimes \mathbf{1} \vdash M \leftrightarrow K$ **by**(*force dest*: *frameImpE*)
then have $\Psi_G \vdash M \leftrightarrow K$ **by**(*blast intro*: *statEqEnt Identity*)
then show *?case* **using** ‹*distinct xvec*› ‹*set xvec* $\subseteq$ *supp N*› ‹*length xvec* = *length Tvec*› **using** *cInput Input*
by(*force simp add*: *residualInject*)
next
case(*cBrInput* Ψ_F *M K xvec N Tvec P* α *P'* Ψ_G A_F A_G)
from *cBrInput* **have** $A_F \sharp* K$ **and** $A_G \sharp* K$ **by**(*auto simp add*: *residualInject*)

from $\langle A_F \sharp* (M(\!|\lambda* xvec\ N|\!).P)\rangle$ $\langle A_G \sharp* (M(\!|\lambda* xvec\ N|\!).P)\rangle$ **have** $A_F \sharp* M$ **and** $A_G \sharp* M$ **by** *simp+*
from $\langle \Psi_F \vdash K \succeq M\rangle$
have $\Psi_F \otimes \mathbf{1} \vdash K \succeq M$
by(*blast intro*: *statEqEnt Identity AssertionStatEqSym*)
with $\langle A_F \sharp* M\rangle$ $\langle A_F \sharp* K\rangle$
have $(\langle A_F,\ \Psi_F \otimes \mathbf{1}\rangle) \vdash_F K \succeq M$
by(*force intro*: *frameImpI*)
with $\langle\langle A_F,\ \Psi_F \otimes \mathbf{1}\rangle \hookrightarrow_F \langle A_G,\ \Psi_G \otimes \mathbf{1}\rangle\rangle$
have $(\langle A_G,\ \Psi_G \otimes \mathbf{1}\rangle) \vdash_F K \succeq M$
by(*simp add*: *FrameStatEq-def FrameStatImp-def*)
with $\langle A_G \sharp* M\rangle$ $\langle A_G \sharp* K\rangle$
have $\Psi_G \otimes \mathbf{1} \vdash K \succeq M$ **by**(*force dest*: *frameImpE*)
then have $\Psi_G \vdash K \succeq M$ **by**(*blast intro*: *statEqEnt Identity*)
then show *?case* **using** ‹*distinct xvec*› ‹*set xvec* $\subseteq$ *supp N*› ‹*length xvec* = *length Tvec*› **using** *cBrInput BrInput*
by(*force simp add*: *residualInject*)
next
case(*cOutput* Ψ_F *M K N P* α *P'* Ψ_G A_F A_G)
from *cOutput* **have** $A_F \sharp* K$ **and** $A_G \sharp* K$ **by**(*auto simp add*: *residualInject*)

from $\langle A_F \sharp* (M\langle N\rangle.P)\rangle$ $\langle A_G \sharp* (M\langle N\rangle.P)\rangle$ **have** $A_F \sharp* M$ **and** $A_G \sharp* M$ **by** *simp+*
from $\langle \Psi_F \vdash M \leftrightarrow K\rangle$
have $\Psi_F \otimes \mathbf{1} \vdash M \leftrightarrow K$
by(*blast intro*: *statEqEnt Identity AssertionStatEqSym*)
with $\langle A_F \sharp* M\rangle$ $\langle A_F \sharp* K\rangle$
have $(\langle A_F,\ \Psi_F \otimes \mathbf{1}\rangle) \vdash_F M \leftrightarrow K$
by(*force intro*: *frameImpI*)
with $\langle\langle A_F,\ \Psi_F \otimes \mathbf{1}\rangle \hookrightarrow_F \langle A_G,\ \Psi_G \otimes \mathbf{1}\rangle\rangle$
have $(\langle A_G,\ \Psi_G \otimes \mathbf{1}\rangle) \vdash_F M \leftrightarrow K$
by(*simp add*: *FrameStatImp-def*)
with $\langle A_G \sharp* M\rangle$ $\langle A_G \sharp* K\rangle$
have $\Psi_G \otimes \mathbf{1} \vdash M \leftrightarrow K$ **by**(*force dest*: *frameImpE*)

then have $\Psi_G \vdash M \leftrightarrow K$ **by**(*blast intro*: *statEqEnt Identity*)
then show *?case* **using** *cOutput Output* **by**(*force simp add*: *residualInject*)
next
case(*cBrOutput* Ψ_F M K N P α P' Ψ_G A_F A_G)
from *cBrOutput* **have** $A_F \sharp* K$ **and** $A_G \sharp* K$ **by**(*auto simp add*: *residualInject*)

from ‹$A_F \sharp* (M\langle N\rangle.P)$› ‹$A_G \sharp* (M\langle N\rangle.P)$› **have** $A_F \sharp* M$ **and** $A_G \sharp* M$ **by** *simp+*
from ‹$\Psi_F \vdash M \preceq K$›
have $\Psi_F \otimes \mathbf{1} \vdash M \preceq K$
by(*blast intro*: *statEqEnt Identity AssertionStatEqSym*)
with ‹$A_F \sharp* M$› ‹$A_F \sharp* K$›
have $(\langle A_F, \Psi_F \otimes \mathbf{1}\rangle) \vdash_F M \preceq K$
by(*force intro*: *frameImpI*)
with ‹$\langle A_F, \Psi_F \otimes \mathbf{1}\rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \mathbf{1}\rangle$›
have $(\langle A_G, \Psi_G \otimes \mathbf{1}\rangle) \vdash_F M \preceq K$
by(*simp add*: *FrameStatImp-def*)
with ‹$A_G \sharp* M$› ‹$A_G \sharp* K$›
have $\Psi_G \otimes \mathbf{1} \vdash M \preceq K$ **by**(*force dest*: *frameImpE*)
then have $\Psi_G \vdash M \preceq K$ **by**(*blast intro*: *statEqEnt Identity*)
then show *?case* **using** *cBrOutput BrOutput* **by**(*force simp add*: *residualInject*)
next
case(*cCase* Ψ_F P φ *Cs* A_P Ψ_P α P' Ψ_G A_F A_G)
from ‹$\Psi_P \simeq \mathbf{1}$› **have** $\langle A_F, \Psi_F \otimes \Psi_P\rangle \simeq_F \langle A_F, \Psi_F \otimes \mathbf{1}\rangle$
by(*metis frameIntCompositionSym Identity AssertionStatEqTrans*)
moreover note ‹$\langle A_F, \Psi_F \otimes \mathbf{1}\rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \mathbf{1}\rangle$›
moreover from ‹$\Psi_P \simeq \mathbf{1}$› **have** $\langle A_G, \Psi_G \otimes \mathbf{1}\rangle \simeq_F \langle A_G, \Psi_G \otimes \Psi_P\rangle$
by(*metis frameIntCompositionSym Identity AssertionStatEqTrans AssertionStatEqSym*)
ultimately have $\langle A_F, \Psi_F \otimes \Psi_P\rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P\rangle$
by(*rule FrameStatEqImpCompose*)
with *cCase* **have** $\Psi_G \rhd P \longmapsto \alpha \prec P'$
by (*metis freshStarPair*(*2*) *memFreshChain*(*1*) *psiCasesFreshChain*(*1*) *psiFreshVec*(*3*))
moreover note ‹$(\varphi, P) \in$ *set Cs*›
moreover from ‹$A_F \sharp* (Cases\ Cs)$›‹$A_G \sharp* (Cases\ Cs)$› ‹$(\varphi, P) \in$ *set Cs*› **have** $A_F \sharp* \varphi$ **and** $A_G \sharp* \varphi$
by(*auto dest*: *memFreshChain*)
from ‹$\Psi_F \vdash \varphi$› **have** $\Psi_F \otimes \mathbf{1} \vdash \varphi$ **by**(*blast intro*: *statEqEnt Identity AssertionStatEqSym*)
with ‹$A_F \sharp* \varphi$› **have** $(\langle A_F, \Psi_F \otimes \mathbf{1}\rangle) \vdash_F \varphi$ **by**(*force intro*: *frameImpI*)
with ‹$\langle A_F, \Psi_F \otimes \mathbf{1}\rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \mathbf{1}\rangle$› **have** $(\langle A_G, \Psi_G \otimes \mathbf{1}\rangle) \vdash_F \varphi$
by(*simp add*: *FrameStatImp-def*)
with ‹$A_G \sharp* \varphi$› **have** $\Psi_G \otimes \mathbf{1} \vdash \varphi$ **by**(*force dest*: *frameImpE*)
then have $\Psi_G \vdash \varphi$ **by**(*blast intro*: *statEqEnt Identity*)
ultimately show *?case* **using** ‹*guarded P*› *cCase Case* **by**(*force intro*: *residualInject*)
next
case(*cPar1* Ψ_F Ψ_Q P α P' A_Q Q A_P Ψ_P α' PQ' Ψ_G A_F A_G)
from ‹$A_F \sharp* (P \parallel Q)$› ‹$A_G \sharp* (P \parallel Q)$› **have** $A_F \sharp* P$ **and** $A_G \sharp* P$ **and** A_F

$\sharp*$ Q **and** A_G $\sharp*$ Q
by *simp+*
have *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$ **by** *fact*
have $\langle A_F, (\Psi_F \otimes \Psi_Q) \otimes \Psi_P\rangle \simeq_F \langle A_F, \Psi_F \otimes \Psi_P \otimes \Psi_Q\rangle$
by(*metis Associativity Composition AssertionStatEqSym AssertionStatEqTrans Commutativity frameResChainPres frameNilStatEq*)
moreover note $\langle\langle A_F, \Psi_F \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q\rangle\rangle$
moreover have $\langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q\rangle \simeq_F \langle A_G, (\Psi_G \otimes \Psi_Q) \otimes \Psi_P\rangle$
by(*metis Associativity Composition AssertionStatEqSym AssertionStatEqTrans Commutativity frameResChainPres frameNilStatEq*)
ultimately have $\langle A_F, (\Psi_F \otimes \Psi_Q) \otimes \Psi_P\rangle \hookrightarrow_F \langle A_G, (\Psi_G \otimes \Psi_Q) \otimes \Psi_P\rangle$
by(*rule FrameStatEqImpCompose*)
moreover from $\langle A_F \sharp* Q\rangle$ $\langle A_G \sharp* Q\rangle$ *FrQ* $\langle A_Q \sharp* A_F\rangle$ $\langle A_Q \sharp* A_G\rangle$ **have** $A_F \sharp* \Psi_Q$ **and** $A_G \sharp* \Psi_Q$
by(*force dest*: *extractFrameFreshChain*)*+*
moreover note $\langle A_F \sharp* P\rangle$ $\langle A_G \sharp* P\rangle$ $\langle A_F \sharp*$ *subject* $\alpha'\rangle$ $\langle A_G \sharp*$ *subject* $\alpha'\rangle$ $\langle A_P \sharp* A_F\rangle$ $\langle A_P \sharp* A_G\rangle$ $\langle A_P \sharp* \Psi_F\rangle$ $\langle A_P \sharp* \Psi_Q\rangle$ $\langle A_P \sharp* A_G\rangle$ $\langle A_P \sharp* \Psi_G\rangle$
moreover from $\langle\alpha \prec P' \parallel Q = \alpha' \prec PQ'\rangle$ $\langle bn\ \alpha \sharp* \alpha'\rangle$
obtain p P'' **where** $\alpha \prec P' = \alpha' \prec P''$ **and** *set* $p \subseteq$ *set*$(bn\ \alpha') \times$ *set*$(bn\ \alpha)$ **and** $PQ' = P'' \parallel (p \cdot Q)$
apply(*drule-tac sym*)
by(*rule actionPar1Dest*) (*assumption* | *simp* | *blast dest*: *sym*)*+*
ultimately have $\Psi_G \otimes \Psi_Q \rhd P \longmapsto \alpha \prec P'$ **using** $\langle\alpha' \neq \tau\rangle$ **by**(*force intro*: *cPar1*)
then show *?case* **using** *FrQ* $\langle(bn\ \alpha) \sharp* Q\rangle$ $\langle A_Q \sharp* \Psi_G\rangle$ $\langle A_Q \sharp* P\rangle$ $\langle A_Q \sharp* \alpha\rangle$ **using** *cPar1*
by(*metis Par1*)
next
case(*cPar2* Ψ_F Ψ_P Q α Q' A_P P A_Q Ψ_Q α' PQ' Ψ_G A_F A_G)
from $\langle A_F \sharp* (P \parallel Q)\rangle$ $\langle A_G \sharp* (P \parallel Q)\rangle$ **have** $A_F \sharp* P$ **and** $A_G \sharp* P$ **and** $A_F \sharp* Q$ **and** $A_G \sharp* Q$
by *simp+*
have *FrP*: *extractFrame* $P = \langle A_P, \Psi_P\rangle$ **by** *fact*
have $\langle A_F, (\Psi_F \otimes \Psi_P) \otimes \Psi_Q\rangle \simeq_F \langle A_F, \Psi_F \otimes \Psi_P \otimes \Psi_Q\rangle$
by(*metis Associativity frameResChainPres frameNilStatEq*)
moreover note $\langle\langle A_F, \Psi_F \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q\rangle\rangle$
moreover have $\langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q\rangle \simeq_F \langle A_G, (\Psi_G \otimes \Psi_P) \otimes \Psi_Q\rangle$
by(*metis Associativity AssertionStatEqSym frameResChainPres frameNilStatEq*)
ultimately have $\langle A_F, (\Psi_F \otimes \Psi_P) \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_G, (\Psi_G \otimes \Psi_P) \otimes \Psi_Q\rangle$
by(*rule FrameStatEqImpCompose*)
moreover from $\langle A_F \sharp* P\rangle$ $\langle A_G \sharp* P\rangle$ *FrP* $\langle A_P \sharp* A_F\rangle$ $\langle A_P \sharp* A_G\rangle$ **have** $A_F \sharp* \Psi_P$ **and** $A_G \sharp* \Psi_P$
by(*force dest*: *extractFrameFreshChain*)*+*
moreover note $\langle A_F \sharp* Q\rangle$ $\langle A_G \sharp* Q\rangle$ $\langle A_F \sharp*$ *subject* $\alpha'\rangle$ $\langle A_G \sharp*$ *subject* $\alpha'\rangle$ $\langle A_Q \sharp* A_F\rangle$ $\langle A_Q \sharp* A_G\rangle$ $\langle A_Q \sharp* \Psi_F\rangle$ $\langle A_Q \sharp* \Psi_P\rangle$ $\langle A_Q \sharp* A_G\rangle$ $\langle A_Q \sharp* \Psi_G\rangle$
moreover from $\langle\alpha \prec P \parallel Q' = \alpha' \prec PQ'\rangle$ $\langle bn\ \alpha \sharp* \alpha'\rangle$
obtain p Q'' **where** $\alpha \prec Q' = \alpha' \prec Q''$ **and** *set* $p \subseteq$ *set*$(bn\ \alpha') \times$ *set*$(bn\ \alpha)$ **and** $PQ' = (p \cdot P) \parallel Q''$
apply(*drule-tac sym*)

by(*rule actionPar2Dest*) (*assumption* | *simp* | *blast dest*: *sym*)+
ultimately have $\Psi_G \otimes \Psi_P \rhd Q \longmapsto \alpha \prec Q'$ **using** $\langle \alpha' \neq \tau \rangle$ **by**(*force intro*: *cPar2*)
then show *?case* **using** *FrP* $\langle (bn\ \alpha) \sharp* P \rangle$ $\langle A_P \sharp* \Psi_G \rangle$ $\langle A_P \sharp* Q \rangle$ $\langle A_P \sharp* \alpha \rangle$ **using** *cPar2*
by(*metis Par2*)
next
case *cComm1*
then show *?case* **by**(*simp add*: *residualInject*)
next
case *cComm2*
then show *?case* **by**(*simp add*: *residualInject*)
next
case(*cBrMerge* $\Psi_F\ \Psi_Q\ P\ M\ N\ P'\ A_P\ \Psi_P\ Q\ Q'\ A_Q\ \alpha\ PQ'\ \Psi_G\ A_F\ A_G$)
from $\langle \langle A_F, \Psi_F \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q \rangle \rangle$
have $\langle A_F, (\Psi_F \otimes \Psi_Q) \otimes \Psi_P \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q \rangle$
by (*metis AssertionStatEqTrans AssertionStatEq-def Associativity FrameStatImpTrans associativitySym frameImpNilStatEq frameImpResChainPres*)
moreover have $\langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_G, (\Psi_G \otimes \Psi_Q) \otimes \Psi_P \rangle$
by (*metis AssertionStatEqTrans AssertionStatEq-def Associativity associativitySym frameImpNilStatEq frameImpResChainPres*)
ultimately have *FimpP*: $\langle A_F, (\Psi_F \otimes \Psi_Q) \otimes \Psi_P \rangle \hookrightarrow_F \langle A_G, (\Psi_G \otimes \Psi_Q) \otimes \Psi_P \rangle$
by (*metis FrameStatImpTrans*)

from $\langle ¿M(\!|N|\!) \prec P' \parallel Q' = \alpha \prec PQ' \rangle$ **have** $\alpha = ¿M(\!|N|\!)$ **and** $(P' \parallel Q') = PQ'$
by(*simp add*: *residualInject*)+

moreover note *FimpP* $\langle A_F \sharp* (P \parallel Q) \rangle$ $\langle A_G \sharp* (P \parallel Q) \rangle$ $\langle A_F \sharp* subject\ \alpha \rangle$ $\langle A_G \sharp* subject\ \alpha \rangle$
$\langle A_P \sharp* A_F \rangle$ $\langle A_P \sharp* A_G \rangle$ $\langle A_P \sharp* \Psi_F \rangle$ $\langle A_P \sharp* \Psi_G \rangle$ $\langle A_P \sharp* \Psi_Q \rangle$
ultimately have *TransP*: $\Psi_G \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$
by(*intro cBrMerge*(*2*)[**where** *bd*=A_G]) *auto*

from $\langle \langle A_F, \Psi_F \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q \rangle \rangle$
have $\langle A_F, (\Psi_F \otimes \Psi_P) \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q \rangle$
by (*metis FrameStatEq-def FrameStatImpTrans frameIntAssociativity*)
moreover have $\langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_G, (\Psi_G \otimes \Psi_P) \otimes \Psi_Q \rangle$
by (*metis FrameStatEq-def frameIntAssociativity*)
ultimately have *FimpQ*: $\langle A_F, (\Psi_F \otimes \Psi_P) \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_G, (\Psi_G \otimes \Psi_P) \otimes \Psi_Q \rangle$
by (*metis FrameStatImpTrans*)

note $\langle \alpha = ¿M(\!|N|\!) \rangle$ $\langle (P' \parallel Q') = PQ' \rangle$ *FimpQ* $\langle A_F \sharp* (P \parallel Q) \rangle$ $\langle A_G \sharp* (P \parallel Q) \rangle$ $\langle A_F \sharp* subject\ \alpha \rangle$ $\langle A_G \sharp* subject\ \alpha \rangle$
$\langle A_Q \sharp* A_F \rangle$ $\langle A_Q \sharp* A_G \rangle$ $\langle A_Q \sharp* \Psi_F \rangle$ $\langle A_Q \sharp* \Psi_G \rangle$ $\langle A_Q \sharp* \Psi_P \rangle$
then have *TransQ*: $\Psi_G \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$
by(*intro cBrMerge*(*6*)[**where** *bd*=A_G]) *auto*

have $\Psi_G \rhd P \parallel Q \longmapsto ¿M(\!|N|\!) \prec P' \parallel Q'$ **using** *TransP TransQ cBrMerge*

by(*intro BrMerge*)
with ‹¿M(|N|) ≺ P' ∥ Q' = α ≺ PQ'›
show *?case* **by** *simp*
next
case(*cBrComm1* Ψ_F Ψ_Q P M N P' A_P Ψ_P Q *xvec* Q' A_Q α PQ' Ψ_G A_F A_G)
from ‹¡M(|ν∗*xvec*|)⟨N⟩ ≺ P' ∥ Q' = α ≺ PQ'› ‹A_F ♯∗ *subject* α› ‹A_G ♯∗ *subject* α›
have A_F ♯∗ M **and** A_G ♯∗ M
by(*auto simp add*: *residualInject*)

from ‹⟨A_F, Ψ_F ⊗ Ψ_P ⊗ Ψ_Q⟩ ↪$_F$ ⟨A_G, Ψ_G ⊗ Ψ_P ⊗ Ψ_Q⟩›
have ⟨A_F, (Ψ_F ⊗ Ψ_Q) ⊗ Ψ_P⟩ ↪$_F$ ⟨A_G, Ψ_G ⊗ Ψ_P ⊗ Ψ_Q⟩
by (*metis AssertionStatEqTrans AssertionStatEq-def Associativity FrameStatImpTrans associativitySym frameImpNilStatEq frameImpResChainPres*)
moreover have ⟨A_G, Ψ_G ⊗ Ψ_P ⊗ Ψ_Q⟩ ↪$_F$ ⟨A_G, (Ψ_G ⊗ Ψ_Q) ⊗ Ψ_P⟩
by (*metis AssertionStatEqTrans AssertionStatEq-def Associativity associativitySym frameImpNilStatEq frameImpResChainPres*)
ultimately have *FimpP*: ⟨A_F, (Ψ_F ⊗ Ψ_Q) ⊗ Ψ_P⟩ ↪$_F$ ⟨A_G, (Ψ_G ⊗ Ψ_Q) ⊗ Ψ_P⟩
by (*metis FrameStatImpTrans*)

note *FimpP* ‹A_F ♯∗ (P ∥ Q)› ‹A_G ♯∗ (P ∥ Q)› ‹A_F ♯∗ M› ‹A_G ♯∗ M›
‹A_P ♯∗ A_F› ‹A_P ♯∗ A_G› ‹A_P ♯∗ Ψ_F› ‹A_P ♯∗ Ψ_G› ‹A_P ♯∗ Ψ_Q›
then have *TransP*: Ψ_G ⊗ Ψ_Q ▷ P ⟼ ¿M(|N|) ≺ P'
by (*intro cBrComm1*(*4*)[**where** *bc*=A_F **and** *bd*=A_G]) *auto*
from ‹⟨A_F, Ψ_F ⊗ Ψ_P ⊗ Ψ_Q⟩ ↪$_F$ ⟨A_G, Ψ_G ⊗ Ψ_P ⊗ Ψ_Q⟩›
have ⟨A_F, (Ψ_F ⊗ Ψ_P) ⊗ Ψ_Q⟩ ↪$_F$ ⟨A_G, Ψ_G ⊗ Ψ_P ⊗ Ψ_Q⟩
by (*metis FrameStatEq-def FrameStatImpTrans frameIntAssociativity*)
moreover have ⟨A_G, Ψ_G ⊗ Ψ_P ⊗ Ψ_Q⟩ ↪$_F$ ⟨A_G, (Ψ_G ⊗ Ψ_P) ⊗ Ψ_Q⟩
by (*metis FrameStatEq-def frameIntAssociativity*)
ultimately have *FimpQ*: ⟨A_F, (Ψ_F ⊗ Ψ_P) ⊗ Ψ_Q⟩ ↪$_F$ ⟨A_G, (Ψ_G ⊗ Ψ_P) ⊗ Ψ_Q⟩
by (*metis FrameStatImpTrans*)
note ‹*distinct xvec*› *FimpQ* ‹A_F ♯∗ (P ∥ Q)› ‹A_G ♯∗ (P ∥ Q)› ‹A_F ♯∗ M› ‹A_G ♯∗ M›
‹A_Q ♯∗ A_F› ‹A_Q ♯∗ A_G› ‹A_Q ♯∗ Ψ_F› ‹A_Q ♯∗ Ψ_G› ‹A_Q ♯∗ Ψ_P›
then have *TransQ*: Ψ_G ⊗ Ψ_P ▷ Q ⟼ ¡M(|ν∗*xvec*|)⟨N⟩ ≺ Q'
by (*simp add*: *cBrComm1.hyps*(*8*) *freshCompChain*(*1*))

from *TransP TransQ cBrComm1*
have Ψ_G ▷ P ∥ Q ⟼ ¡M(|ν∗*xvec*|)⟨N⟩ ≺ P' ∥ Q'
by(*intro BrComm1*)
with ‹¡M(|ν∗*xvec*|)⟨N⟩ ≺ P' ∥ Q' = α ≺ PQ'›
show *?case*
by *simp*
next
case(*cBrComm2* Ψ_F Ψ_Q P M *xvec* N P' A_P Ψ_P Q Q' A_Q α PQ' Ψ_G A_F A_G)
from ‹¡M(|ν∗*xvec*|)⟨N⟩ ≺ P' ∥ Q' = α ≺ PQ'› ‹A_F ♯∗ *subject* α› ‹A_G ♯∗ *subject* α›
have A_F ♯∗ M **and** A_G ♯∗ M

by(*auto simp add*: *residualInject*)

from ‹$\langle A_F, \Psi_F \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q\rangle$›
have $\langle A_F, (\Psi_F \otimes \Psi_P) \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q\rangle$
by (*metis FrameStatEq-def FrameStatImpTrans frameIntAssociativity*)
moreover have $\langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_G, (\Psi_G \otimes \Psi_P) \otimes \Psi_Q\rangle$
by (*metis FrameStatEq-def frameIntAssociativity*)
ultimately have *FimpQ*: $\langle A_F, (\Psi_F \otimes \Psi_P) \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_G, (\Psi_G \otimes \Psi_P) \otimes \Psi_Q\rangle$
by (*metis FrameStatImpTrans*)

note *FimpQ* ‹$A_F \sharp^* (P \parallel Q)$› ‹$A_G \sharp^* (P \parallel Q)$› ‹$A_F \sharp^* M$› ‹$A_G \sharp^* M$›
‹$A_Q \sharp^* A_F$› ‹$A_Q \sharp^* A_G$› ‹$A_Q \sharp^* \Psi_F$› ‹$A_Q \sharp^* \Psi_G$› ‹$A_Q \sharp^* \Psi_P$›
then have *TransQ*: $\Psi_G \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$
by(*intro cBrComm2*(*8*)[**where** *bc*=A_F **and** *bd*=A_G]) *auto*

from ‹$\langle A_F, \Psi_F \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q\rangle$›
have $\langle A_F, (\Psi_F \otimes \Psi_Q) \otimes \Psi_P\rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q\rangle$
by (*metis AssertionStatEqTrans AssertionStatEq-def Associativity FrameStatImpTrans associativitySym frameImpNilStatEq frameImpResChainPres*)
moreover have $\langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_G, (\Psi_G \otimes \Psi_Q) \otimes \Psi_P\rangle$
by (*metis AssertionStatEqTrans AssertionStatEq-def Associativity associativitySym frameImpNilStatEq frameImpResChainPres*)
ultimately have *FimpP*: $\langle A_F, (\Psi_F \otimes \Psi_Q) \otimes \Psi_P\rangle \hookrightarrow_F \langle A_G, (\Psi_G \otimes \Psi_Q) \otimes \Psi_P\rangle$
by (*metis FrameStatImpTrans*)

note ‹*distinct xvec*› *FimpP* ‹$A_F \sharp^* (P \parallel Q)$› ‹$A_G \sharp^* (P \parallel Q)$› ‹$A_F \sharp^* M$› ‹$A_G \sharp^* M$›
‹$A_P \sharp^* A_F$› ‹$A_P \sharp^* A_G$› ‹$A_P \sharp^* \Psi_F$› ‹$A_P \sharp^* \Psi_G$› ‹$A_P \sharp^* \Psi_Q$›
then have *TransP*: $\Psi_G \otimes \Psi_Q \rhd P \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$
by(*intro cBrComm2*(*4*)[**where** *bc*=A_F **and** *bd*=A_G]) *auto*

from *TransP TransQ cBrComm2*
have $\Psi_G \rhd P \parallel Q \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec P' \parallel Q'$
by(*intro BrComm2*)
with ‹$¡M(\!|\nu * xvec|\!)\langle N\rangle \prec P' \parallel Q' = \alpha \prec PQ'$›
show *?case*
by *simp*
next
case *cBrClose*
then show *?case* **by**(*simp add*: *residualInject*)
next
case(*cOpen* Ψ_F P M *xvec yvec* N P' x A_P Ψ_P α P'' Ψ_G A_F A_G)
from ‹$M(\!|\nu * (xvec\ @\ x\ \#\ yvec)|\!)\langle N\rangle \prec P' = \alpha \prec P''$› ‹$x \sharp xvec$› ‹$x \sharp yvec$› ‹$x \sharp \alpha$› ‹$x \sharp P''$› ‹*distinct*(*bn* α)›
obtain *xvec'* *x'* *yvec'* N' **where** $M(\!|\nu * (xvec @ yvec)|\!)\langle N\rangle \prec P' = M(\!|\nu * (xvec' @ yvec')|\!)\langle([(x, x')] \cdot N')\rangle \prec ([(x, x')] \cdot P'')$
and $\alpha = M(\!|\nu * (xvec' @ x' \# yvec')|\!)\langle N'\rangle$
apply(*cases rule*: *actionCases*[**where** α=α])

apply(*simp add*: *residualInject*)
apply(*simp add*: *residualInject*)
apply(*simp add*: *residualInject*)
apply(*metis boundOutputOpenDest*)
apply(*simp add*: *residualInject*)
by(*simp add*: *residualInject*)

then have $\Psi_G \rhd P \longmapsto M(\!|\nu*(xvec@yvec)|\!)\langle N\rangle \prec P'$ **using** *cOpen*
by(*intro cOpen*(*4*)[**where** *bc*=A_F **and** *bd*=A_G]) *auto*
with ‹$x \in supp\ N$› ‹$x \sharp \Psi_G$› ‹$x \sharp M$› ‹$x \sharp xvec$› ‹$x \sharp yvec$›
have $\Psi_G \rhd (\!|\nu x|\!)P \longmapsto M(\!|\nu*(xvec@x\#yvec)|\!)\langle N\rangle \prec P'$
by(*intro Open*)
then show *?case* **using** ‹$\alpha = M(\!|\nu*(xvec'@x'\#yvec')|\!)\langle N'\rangle$› ‹$M(\!|\nu*(xvec\ @\ x\ \#\ yvec)|\!)\langle N\rangle \prec P' = \alpha \prec P''$›
by *simp*
next
case(*cBrOpen* $\Psi_F\ P\ M\ xvec\ yvec\ N\ P'\ x\ A_P\ \Psi_P\ \alpha\ P''\ \Psi_G\ A_F\ A_G$)
from ‹$¡M(\!|\nu*(xvec\ @\ x\ \#\ yvec)|\!)\langle N\rangle \prec P' = \alpha \prec P''$› ‹$x \sharp xvec$› ‹$x \sharp yvec$› ‹$x \sharp \alpha$› ‹$x \sharp P''$› ‹*distinct*(*bn* α)›
obtain $xvec'\ x'\ yvec'\ N'$ **where** $¡M(\!|\nu*(xvec@yvec)|\!)\langle N\rangle \prec P' = ¡M(\!|\nu*(xvec'@yvec')|\!)\langle([(x, x')] \cdot N')\rangle \prec ([(x, x')] \cdot P'')$
and $\alpha = ¡M(\!|\nu*(xvec'@x'\#yvec')|\!)\langle N'\rangle$
apply(*cases rule*: *actionCases*[**where** α=α])
apply(*simp add*: *residualInject*)
apply(*simp add*: *residualInject*)
apply(*simp add*: *residualInject*)
apply(*simp add*: *residualInject*)
apply(*metis boundOutputOpenDest*)
by(*simp add*: *residualInject*)

then have $\Psi_G \rhd P \longmapsto ¡M(\!|\nu*(xvec@yvec)|\!)\langle N\rangle \prec P'$ **using** *cBrOpen*
by(*intro cBrOpen*) (*assumption* | *simp*)+
with ‹$x \in supp\ N$› ‹$x \sharp \Psi_G$› ‹$x \sharp M$› ‹$x \sharp xvec$› ‹$x \sharp yvec$›
have $\Psi_G \rhd (\!|\nu x|\!)P \longmapsto ¡M(\!|\nu*(xvec@x\#yvec)|\!)\langle N\rangle \prec P'$
by(*intro BrOpen*)
then show *?case* **using** ‹$\alpha = ¡M(\!|\nu*(xvec'@x'\#yvec')|\!)\langle N'\rangle$› ‹$¡M(\!|\nu*(xvec\ @\ x\ \#\ yvec)|\!)\langle N\rangle \prec P' = \alpha \prec P''$›
by *simp*
next
case(*cScope* $\Psi_F\ P\ \alpha\ P'\ x\ A_P\ \Psi_P\ \alpha'\ xP\ \Psi_G\ A_F\ A_G$)
from ‹$\alpha \prec (\!|\nu x|\!)P' = \alpha' \prec xP$› ‹$x \sharp \alpha$› ‹$x \sharp \alpha'$› **obtain** P'' **where** $xP = (\!|\nu x|\!)P''$ **and** $\alpha \prec P' = \alpha' \prec P''$
by(*drule-tac sym*) (*force intro*: *actionScopeDest*)
then have $\Psi_G \rhd P \longmapsto \alpha \prec P'$ **using** *cScope* **by** *auto*
with ‹$x \sharp \Psi_G$› ‹$x \sharp \alpha'$› ‹$\alpha \prec P' = \alpha' \prec P''$› ‹$xP = (\!|\nu x|\!)P''$› **show** *?case*
by(*metis Scope*)
next
case(*cBang* $\Psi_F\ P\ A_P\ \Psi_P\ \alpha\ P'\ \Psi_G\ A_F\ A_G$)
from ‹$\Psi_P \simeq \mathbf{1}$› **have** $\langle A_F, \Psi_F \otimes \Psi_P \otimes \mathbf{1}\rangle \simeq_F \langle A_F, \Psi_F \otimes \mathbf{1}\rangle$

by(*metis frameIntCompositionSym Identity AssertionStatEqTrans*)
moreover note ‹$\langle A_F, \Psi_F \otimes \mathbf{1}\rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \mathbf{1}\rangle$›
moreover from ‹$\Psi_P \simeq \mathbf{1}$› **have** $\langle A_G, \Psi_G \otimes \mathbf{1}\rangle \simeq_F \langle A_G, \Psi_G \otimes \Psi_P \otimes \mathbf{1}\rangle$
by(*metis frameIntCompositionSym Identity AssertionStatEqTrans AssertionStatEqSym*)
ultimately have $\langle A_F, \Psi_F \otimes \Psi_P \otimes \mathbf{1}\rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \otimes \mathbf{1}\rangle$
by(*rule FrameStatEqImpCompose*)
with *cBang* **have** $\Psi_G \rhd P \parallel !P \longmapsto \alpha \prec P'$ **by** *force*
then show *?case* **using** ‹*guarded P*› **using** *cBang* **by**(*metis Bang*)
qed

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 $\Psi_G \rhd P \longmapsto \tau \prec P'$
using *assms*
proof(*nominal-induct avoiding*: Ψ_G A_F A_G *rule*: *tauFrameInduct*)
case(*cAlpha* Ψ_F P P' A_P Ψ_P p Ψ_G A_F A_G)
from ‹$\langle A_F, \Psi_F \otimes (p \cdot \Psi_P)\rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes (p \cdot \Psi_P)\rangle$›
have $(p \cdot (\langle A_F, \Psi_F \otimes (p \cdot \Psi_P)\rangle)) \hookrightarrow_F (p \cdot (\langle A_G, \Psi_G \otimes (p \cdot \Psi_P)\rangle))$
by(*rule FrameStatImpClosed*)
with ‹$A_P \sharp* A_F$› ‹$(p \cdot A_P) \sharp* A_F$› ‹$A_P \sharp* \Psi_F$› ‹$(p \cdot A_P) \sharp* \Psi_F$› ‹$A_P \sharp* A_G$› ‹$(p \cdot A_P) \sharp* A_G$› ‹$A_P \sharp* \Psi_G$› ‹$(p \cdot A_P) \sharp* \Psi_G$›
‹*distinctPerm p*› ‹*set p* ⊆ *set* $A_P \times$ *set* $(p \cdot A_P)$› **have** $\langle A_F, \Psi_F \otimes \Psi_P\rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P\rangle$
by(*simp add*: *eqvts*)
with *cAlpha* **show** *?case* **by** *blast*
next
case(*cCase* Ψ_F P P' φ *Cs* A_P Ψ_P Ψ_G A_F A_G)
from ‹$\Psi_P \simeq \mathbf{1}$› **have** $\langle A_F, \Psi_F \otimes \Psi_P\rangle \simeq_F \langle A_F, \Psi_F \otimes \mathbf{1}\rangle$
by(*metis frameIntCompositionSym Identity AssertionStatEqTrans*)
moreover note ‹$\langle A_F, \Psi_F \otimes \mathbf{1}\rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \mathbf{1}\rangle$›
moreover from ‹$\Psi_P \simeq \mathbf{1}$› **have** $\langle A_G, \Psi_G \otimes \mathbf{1}\rangle \simeq_F \langle A_G, \Psi_G \otimes \Psi_P\rangle$

by(*metis frameIntCompositionSym Identity AssertionStatEqTrans AssertionStatEqSym*)
ultimately have $\langle A_F, \Psi_F \otimes \Psi_P\rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P\rangle$
by(*rule FrameStatEqImpCompose*)
with *cCase* **have** $\Psi_G \rhd P \longmapsto\tau \prec P'$
by (*metis freshStarPair*(*2*) *memFreshChain*(*1*) *psiCasesFreshChain*(*1*) *psiFreshVec*(*3*))
moreover note $\langle(\varphi, P) \in$ *set Cs*$\rangle$
moreover from $\langle A_F \sharp* ($*Cases Cs*$)\rangle\langle A_G \sharp* ($*Cases Cs*$)\rangle$ $\langle(\varphi, P) \in$ *set Cs*$\rangle$ **have** $A_F \sharp* \varphi$ **and** $A_G \sharp* \varphi$
by(*auto dest*: *memFreshChain*)
from $\langle\Psi_F \vdash \varphi\rangle$ **have** $\Psi_F \otimes \mathbf{1} \vdash \varphi$ **by**(*blast intro*: *statEqEnt Identity AssertionStatEqSym*)
with $\langle A_F \sharp* \varphi\rangle$ **have** $(\langle A_F, \Psi_F \otimes \mathbf{1}\rangle) \vdash_F \varphi$ **by**(*force intro*: *frameImpI*)
with $\langle\langle A_F, \Psi_F \otimes \mathbf{1}\rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \mathbf{1}\rangle\rangle$ **have** $(\langle A_G, \Psi_G \otimes \mathbf{1}\rangle) \vdash_F \varphi$
by(*simp add*: *FrameStatImp-def*)
with $\langle A_G \sharp* \varphi\rangle$ **have** $\Psi_G \otimes \mathbf{1} \vdash \varphi$ **by**(*force dest*: *frameImpE*)
then have $\Psi_G \vdash \varphi$ **by**(*blast intro*: *statEqEnt Identity*)
ultimately show *?case* **using** $\langle$*guarded P*$\rangle$ **by**(*rule Case*)
next
case(*cPar1* $\Psi_F\ \Psi_Q\ P\ P'\ A_Q\ Q\ A_P\ \Psi_P\ \Psi_G\ A_F\ A_G$)
from $\langle A_F \sharp* (P \parallel Q)\rangle$ $\langle A_G \sharp* (P \parallel Q)\rangle$ **have** $A_F \sharp* P$ **and** $A_G \sharp* P$ **and** $A_F \sharp* Q$ **and** $A_G \sharp* Q$
by *simp+*
have *IH*: $\bigwedge\Psi\ A_F\ A_G.\ [\![\langle A_F, (\Psi_F \otimes \Psi_Q) \otimes \Psi_P\rangle \hookrightarrow_F \langle A_G, \Psi \otimes \Psi_P\rangle;\ A_F \sharp* P;\ A_G \sharp* P;$
$A_P \sharp* A_F;\ A_P \sharp* A_G;\ A_P \sharp* (\Psi_F \otimes \Psi_Q);\ A_P \sharp* \Psi]\!] \Longrightarrow \Psi \rhd P \longmapsto\tau \prec P'$
by *fact*
have *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$ **by** *fact*
have $\langle A_F, (\Psi_F \otimes \Psi_Q) \otimes \Psi_P\rangle \simeq_F \langle A_F, \Psi_F \otimes \Psi_P \otimes \Psi_Q\rangle$
by(*metis Associativity Composition AssertionStatEqSym AssertionStatEqTrans Commutativity frameResChainPres frameNilStatEq*)
moreover note $\langle\langle A_F, \Psi_F \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q\rangle\rangle$
moreover have $\langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q\rangle \simeq_F \langle A_G, (\Psi_G \otimes \Psi_Q) \otimes \Psi_P\rangle$
by(*metis Associativity Composition AssertionStatEqSym AssertionStatEqTrans Commutativity frameResChainPres frameNilStatEq*)
ultimately have $\langle A_F, (\Psi_F \otimes \Psi_Q) \otimes \Psi_P\rangle \hookrightarrow_F \langle A_G, (\Psi_G \otimes \Psi_Q) \otimes \Psi_P\rangle$
by(*rule FrameStatEqImpCompose*)
moreover from $\langle A_F \sharp* Q\rangle$ $\langle A_G \sharp* Q\rangle$ *FrQ* $\langle A_Q \sharp* A_F\rangle$ $\langle A_Q \sharp* A_G\rangle$ **have** $A_F \sharp* \Psi_Q$ **and** $A_G \sharp* \Psi_Q$
by(*force dest*: *extractFrameFreshChain*)+
moreover note $\langle A_F \sharp* P\rangle$ $\langle A_G \sharp* P\rangle$ $\langle A_P \sharp* A_F\rangle$ $\langle A_P \sharp* A_G\rangle$ $\langle A_P \sharp* \Psi_F\rangle$ $\langle A_P \sharp* \Psi_Q\rangle$ $\langle A_P \sharp* A_G\rangle$ $\langle A_P \sharp* \Psi_G\rangle$
ultimately have $\Psi_G \otimes \Psi_Q \rhd P \longmapsto\tau \prec P'$
using *IH* **by** *blast*
then show *?case* **using** *FrQ* $\langle A_Q \sharp* \Psi_G\rangle$ $\langle A_Q \sharp* P\rangle$
by(*intro Par1*) *auto*
next
case(*cPar2* $\Psi_F\ \Psi_P\ Q\ Q'\ A_P\ P\ A_Q\ \Psi_Q\ \Psi_G\ A_F\ A_G$)

from ‹$A_F \sharp* (P \parallel Q)$› ‹$A_G \sharp* (P \parallel Q)$› **have** $A_F \sharp* P$ **and** $A_G \sharp* P$ **and** $A_F \sharp* Q$ **and** $A_G \sharp* Q$
by *simp+*
have *IH*: $\bigwedge \Psi\ A_F\ A_G.\ [\![\langle A_F, (\Psi_F \otimes \Psi_P) \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_G, \Psi \otimes \Psi_Q\rangle;\ A_F \sharp* Q;\ A_G \sharp* Q;$
$A_Q \sharp* A_F;\ A_Q \sharp* A_G;\ A_Q \sharp* (\Psi_F \otimes \Psi_P);\ A_Q \sharp* \Psi]\!] \Longrightarrow \Psi \rhd Q \longmapsto \tau \prec Q'$
by *fact*
have *FrP*: *extractFrame* $P = \langle A_P, \Psi_P\rangle$ **by** *fact*
have $\langle A_F, (\Psi_F \otimes \Psi_P) \otimes \Psi_Q\rangle \simeq_F \langle A_F, \Psi_F \otimes \Psi_P \otimes \Psi_Q\rangle$
by(*metis Associativity frameResChainPres frameNilStatEq*)
moreover note ‹$\langle A_F, \Psi_F \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q\rangle$›
moreover have $\langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q\rangle \simeq_F \langle A_G, (\Psi_G \otimes \Psi_P) \otimes \Psi_Q\rangle$
by(*metis Associativity AssertionStatEqSym frameResChainPres frameNilStatEq*)
ultimately have $\langle A_F, (\Psi_F \otimes \Psi_P) \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_G, (\Psi_G \otimes \Psi_P) \otimes \Psi_Q\rangle$
by(*rule FrameStatEqImpCompose*)
moreover from ‹$A_F \sharp* P$› ‹$A_G \sharp* P$› *FrP* ‹$A_P \sharp* A_F$› ‹$A_P \sharp* A_G$› **have** $A_F \sharp* \Psi_P$ **and** $A_G \sharp* \Psi_P$
by(*force dest*: *extractFrameFreshChain*)+
moreover note ‹$A_F \sharp* Q$› ‹$A_G \sharp* Q$› ‹$A_Q \sharp* A_F$› ‹$A_Q \sharp* A_G$› ‹$A_Q \sharp* \Psi_F$› ‹$A_Q \sharp* \Psi_P$› ‹$A_Q \sharp* A_G$› ‹$A_Q \sharp* \Psi_G$›
ultimately have $\Psi_G \otimes \Psi_P \rhd Q \longmapsto \tau \prec Q'$
using *IH* **by** *blast*
then show *?case* **using** *FrP* ‹$A_P \sharp* \Psi_G$› ‹$A_P \sharp* Q$›
by(*intro Par2*) *auto*
next
case(*cComm1* $\Psi_F\ \Psi_Q\ P\ M\ N\ P'\ A_P\ \Psi_P\ Q\ K\ xvec\ Q'\ A_Q\ \Psi_G\ A_F\ A_G$)
have *FimpG*: $\langle A_F, \Psi_F \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q\rangle$ **by** *fact*
from ‹$A_F \sharp* (P \parallel Q)$› ‹$A_G \sharp* (P \parallel Q)$› **have** $A_F \sharp* P$ **and** $A_G \sharp* P$ **and** $A_F \sharp* Q$ **and** $A_G \sharp* Q$
by *simp+*
have *FrP*: *extractFrame* $P = \langle A_P, \Psi_P\rangle$ **by** *fact*
have *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$ **by** *fact*
from ‹$A_F \sharp* P$› ‹$A_G \sharp* P$› *FrP* ‹$A_P \sharp* A_F$› ‹$A_P \sharp* A_G$› **have** $A_F \sharp* \Psi_P$ **and** $A_G \sharp* \Psi_P$
by(*force dest*: *extractFrameFreshChain*)+
from ‹$A_F \sharp* Q$› ‹$A_G \sharp* Q$› *FrQ* ‹$A_Q \sharp* A_F$› ‹$A_Q \sharp* A_G$› **have** $A_F \sharp* \Psi_Q$ **and** $A_G \sharp* \Psi_Q$
by(*force dest*: *extractFrameFreshChain*)+

from ‹$\Psi_F \otimes \Psi_P \rhd Q \longmapsto K\langle\!|\nu*xvec|\!\rangle\langle N\rangle \prec Q'$› **have** $\Psi_F \otimes \Psi_P \rhd Q \longmapsto ROut\ K\ (\langle\!|\nu*xvec|\!\rangle N \prec' Q')$
by(*simp add*: *residualInject*)
with *FrQ* ‹*distinct* A_Q›
obtain K' **where** *KeqK'*: $(\Psi_F \otimes \Psi_P) \otimes \Psi_Q \vdash K \leftrightarrow K'$ **and** $A_P \sharp* K'$ **and** $A_F \sharp* K'$ **and** $A_G \sharp* K'$
using ‹$A_P \sharp* Q$› ‹$A_Q \sharp* A_F$› ‹$A_Q \sharp* A_G$› ‹$A_F \sharp* Q$› ‹$A_G \sharp* Q$› ‹$A_Q \sharp* \Psi_F$› ‹$A_Q \sharp* \Psi_P$› ‹$A_P \sharp* A_Q$› ‹$A_Q \sharp* Q$› ‹$A_Q \sharp* K$› ‹*xvec* $\sharp* K$› ‹*distinct xvec*›
by(*elim outputObtainPrefix*[**where** B=A_P@A_F@A_G]) *force+*

have $\Psi_G \otimes \Psi_Q \rhd P \longmapsto K'(\!|N|\!) \prec P'$
proof −
from *KeqK'* **have** $\Psi_F \otimes (\Psi_P \otimes \Psi_Q) \vdash K \leftrightarrow K'$ **by**(*rule statEqEnt*[*OF Associativity*])
with ‹$\Psi_F \otimes (\Psi_P \otimes \Psi_Q) \vdash M \leftrightarrow K$› **have** $\Psi_F \otimes (\Psi_P \otimes \Psi_Q) \vdash M \leftrightarrow K'$
by(*rule chanEqTrans*)
then have $(\Psi_F \otimes \Psi_Q) \otimes \Psi_P \vdash M \leftrightarrow K'$
by(*metis statEqEnt AssertionStatEqSym Associativity AssertionStatEqTrans compositionSym Commutativity*)
with ‹$\Psi_F \otimes \Psi_Q \rhd P \longmapsto M(\!|N|\!) \prec P'$› *FrP* ‹*distinct* A_P›
have $\Psi_F \otimes \Psi_Q \rhd P \longmapsto K'(\!|N|\!) \prec P'$ **using** ‹$A_P \sharp* \Psi_F$› ‹$A_P \sharp* \Psi_Q$› ‹$A_P \sharp* P$› ‹$A_P \sharp* M$› ‹$A_P \sharp* K'$›
by(*force intro*: *inputRenameSubject*)
moreover have $\langle A_F, (\Psi_F \otimes \Psi_Q) \otimes \Psi_P\rangle \hookrightarrow_F \langle A_G, (\Psi_G \otimes \Psi_Q) \otimes \Psi_P\rangle$
proof −
have $\langle A_F, (\Psi_F \otimes \Psi_Q) \otimes \Psi_P\rangle \simeq_F \langle A_F, \Psi_F \otimes \Psi_P \otimes \Psi_Q\rangle$
by(*metis Associativity Composition AssertionStatEqSym AssertionStatEqTrans Commutativity frameResChainPres frameNilStatEq*)
moreover have $\langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q\rangle \simeq_F \langle A_G, (\Psi_G \otimes \Psi_Q) \otimes \Psi_P\rangle$
by(*metis Associativity Composition AssertionStatEqSym AssertionStatEqTrans Commutativity frameResChainPres frameNilStatEq*)
ultimately show *?thesis* **using** *FimpG*
by(*elim FrameStatEqImpCompose*)
qed
ultimately show *?thesis* **using** ‹$A_F \sharp* P$› ‹$A_G \sharp* P$› ‹$A_F \sharp* K'$›
‹$A_G \sharp* K'$› ‹$A_P \sharp* A_F$› ‹$A_P \sharp* A_G$› ‹$A_P \sharp* \Psi_F$› ‹$A_P \sharp* \Psi_G$› ‹$A_P \sharp* \Psi_Q$› *FrP* ‹*distinct* A_P›
by(*auto intro*: *transferNonTauFrame*)
qed

moreover from *FrP* ‹$\Psi_F \otimes \Psi_Q \rhd P \longmapsto M(\!|N|\!) \prec P'$› ‹*distinct* A_P›
obtain M' **where** *MeqM'*: $(\Psi_F \otimes \Psi_Q) \otimes \Psi_P \vdash M \leftrightarrow M'$ **and** $A_Q \sharp* M'$ **and** $A_F \sharp* M'$ **and** $A_G \sharp* M'$
using ‹$A_Q \sharp* P$› ‹$A_P \sharp* A_F$› ‹$A_P \sharp* A_G$› ‹$A_F \sharp* P$› ‹$A_G \sharp* P$› ‹$A_P \sharp* \Psi_F$› ‹$A_P \sharp* \Psi_Q$› ‹$A_P \sharp* A_Q$› ‹$A_P \sharp* P$› ‹$A_P \sharp* M$›
by(*elim inputObtainPrefix*[**where** B=A_Q@A_F@A_G]) *force*+

have $\Psi_G \otimes \Psi_P \rhd Q \longmapsto M'(\!|\nu*xvec|\!)\langle N\rangle \prec Q'$
proof −
from *MeqM'* **have** $\Psi_F \otimes (\Psi_Q \otimes \Psi_P) \vdash M \leftrightarrow M'$
by(*rule statEqEnt*[*OF Associativity*])
with ‹$\Psi_F \otimes (\Psi_P \otimes \Psi_Q) \vdash M \leftrightarrow K$› **have** $\Psi_F \otimes (\Psi_Q \otimes \Psi_P) \vdash K \leftrightarrow M'$
by(*blast intro*: *chanEqTrans chanEqSym compositionSym Commutativity statEqEnt*)
then have $(\Psi_F \otimes \Psi_P) \otimes \Psi_Q \vdash K \leftrightarrow M'$
by(*blast intro*: *statEqEnt AssertionStatEqSym Associativity AssertionStatEqTrans compositionSym Commutativity*)
with ‹$\Psi_F \otimes \Psi_P \rhd Q \longmapsto K(\!|\nu*xvec|\!)\langle N\rangle \prec Q'$› *FrQ* ‹*distinct* A_Q›
have $\Psi_F \otimes \Psi_P \rhd Q \longmapsto M'(\!|\nu*xvec|\!)\langle N\rangle \prec Q'$ **using** ‹$A_Q \sharp* \Psi_F$› ‹$A_Q \sharp* \Psi_P$›

‹A_Q ♯* Q› ‹A_Q ♯* K› ‹A_Q ♯* M'›
by(*force intro*: *outputRenameSubject*)
moreover have $\langle A_F, (\Psi_F \otimes \Psi_P) \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_G, (\Psi_G \otimes \Psi_P) \otimes \Psi_Q\rangle$
proof −
have $\langle A_F, (\Psi_F \otimes \Psi_P) \otimes \Psi_Q\rangle \simeq_F \langle A_F, \Psi_F \otimes \Psi_P \otimes \Psi_Q\rangle$
by(*metis Associativity frameResChainPres frameNilStatEq*)
moreover have $\langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q\rangle \simeq_F \langle A_G, (\Psi_G \otimes \Psi_P) \otimes \Psi_Q\rangle$
by(*metis Associativity AssertionStatEqSym frameResChainPres frameNilStatEq*)
ultimately show *?thesis* **using** *FimpG*
by(*elim FrameStatEqImpCompose*)
qed

ultimately show *?thesis* **using** ‹A_F ♯* Q› ‹A_G ♯* Q› ‹A_F ♯* M'› ‹A_G ♯* M'›
‹A_Q ♯* A_F› ‹A_Q ♯* A_G› ‹A_Q ♯* Ψ_F› ‹A_Q ♯* Ψ_G› ‹A_Q ♯* Ψ_P› *FrQ* ‹*distinct* A_Q› ‹*distinct xvec*›
by(*auto intro*: *transferNonTauFrame*)
qed

moreover have $\Psi_G \otimes \Psi_P \otimes \Psi_Q \vdash K' \leftrightarrow M'$
proof −
from *MeqM'* **have** $\Psi_F \otimes \Psi_P \otimes \Psi_Q \vdash M' \leftrightarrow M$
by(*blast intro*: *chanEqSym Associativity statEqEnt Commutativity compositionSym*)
moreover from *KeqK'* **have** $\Psi_F \otimes \Psi_P \otimes \Psi_Q \vdash K \leftrightarrow K'$
by(*blast intro*: *chanEqSym Associativity statEqEnt Commutativity compositionSym*)
ultimately have $\Psi_F \otimes \Psi_P \otimes \Psi_Q \vdash K' \leftrightarrow M'$ **using** ‹$\Psi_F \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K$›
by(*blast intro*: *chanEqSym chanEqTrans*)
then show *?thesis* **using** ‹A_F ♯* M'› ‹A_F ♯* K'› ‹A_G ♯* M'› ‹A_G ♯* K'› *FimpG*
apply(*simp add*: *FrameStatImp-def*)
apply(*erule allE*[**where** *x*=*SChanEq' K' M'*])
by(*force intro*: *frameImpI dest*: *frameImpE*)
qed

ultimately show *?case* **using** ‹A_P ♯* Ψ_G› ‹A_P ♯* P› ‹A_P ♯* Q› ‹A_P ♯* A_Q› ‹A_P ♯* K'› ‹A_Q ♯* Ψ_G› ‹A_Q ♯* P› ‹A_Q ♯* Q› ‹A_Q ♯* M'› ‹*xvec* ♯* P› *FrP FrQ*
by(*intro Comm1*) (*assumption* | *simp*)+
next
case(*cComm2* Ψ_F Ψ_Q P M *xvec* N P' A_P Ψ_P Q K Q' A_Q Ψ_G A_F A_G)
have *FimpG*: $\langle A_F, \Psi_F \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q\rangle$ **by** *fact*
from ‹A_F ♯* $(P \parallel Q)$› ‹A_G ♯* $(P \parallel Q)$› **have** A_F ♯* P **and** A_G ♯* P **and** A_F ♯* Q **and** A_G ♯* Q
by *simp*+
have *FrP*: *extractFrame* $P = \langle A_P, \Psi_P\rangle$ **by** *fact*
have *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$ **by** *fact*
from ‹A_F ♯* P› ‹A_G ♯* P› *FrP* ‹A_P ♯* A_F› ‹A_P ♯* A_G› **have** A_F ♯* Ψ_P **and**

$A_G \sharp* \Psi_P$
by(*force dest*: *extractFrameFreshChain*)+
from ‹$A_F \sharp* Q$› ‹$A_G \sharp* Q$› *FrQ* ‹$A_Q \sharp* A_F$› ‹$A_Q \sharp* A_G$› **have** $A_F \sharp* \Psi_Q$ **and** $A_G \sharp* \Psi_Q$
by(*force dest*: *extractFrameFreshChain*)+
from ‹$\Psi_F \otimes \Psi_P \rhd Q \longmapsto K(\!|N|\!) \prec Q'$› *FrQ* ‹*distinct* A_Q›
obtain K' **where** *KeqK'*: $(\Psi_F \otimes \Psi_P) \otimes \Psi_Q \vdash K \leftrightarrow K'$ **and** $A_P \sharp* K'$ **and** $A_F \sharp* K'$ **and** $A_G \sharp* K'$
using ‹$A_P \sharp* Q$› ‹$A_Q \sharp* A_F$› ‹$A_Q \sharp* A_G$› ‹$A_F \sharp* Q$› ‹$A_G \sharp* Q$› ‹$A_Q \sharp* \Psi_F$› ‹$A_Q \sharp* \Psi_P$› ‹$A_P \sharp* A_Q$› ‹$A_Q \sharp* Q$› ‹$A_Q \sharp* K$›
by(*elim inputObtainPrefix*[**where** B=A_P@A_F@A_G]) *force*+
have $\Psi_G \otimes \Psi_Q \rhd P \longmapsto K'(\!|\nu*xvec|\!)\langle N\rangle \prec P'$
proof −
from *KeqK'* **have** $\Psi_F \otimes (\Psi_P \otimes \Psi_Q) \vdash K \leftrightarrow K'$
by(*rule statEqEnt*[*OF Associativity*])
with ‹$\Psi_F \otimes (\Psi_P \otimes \Psi_Q) \vdash M \leftrightarrow K$› **have** $\Psi_F \otimes (\Psi_P \otimes \Psi_Q) \vdash M \leftrightarrow K'$
by(*rule chanEqTrans*)
then have $(\Psi_F \otimes \Psi_Q) \otimes \Psi_P \vdash M \leftrightarrow K'$
by(*metis statEqEnt AssertionStatEqSym Associativity AssertionStatEqTrans compositionSym Commutativity*)
with ‹$\Psi_F \otimes \Psi_Q \rhd P \longmapsto M(\!|\nu*xvec|\!)\langle N\rangle \prec P'$› *FrP* ‹*distinct* A_P›
have $\Psi_F \otimes \Psi_Q \rhd P \longmapsto K'(\!|\nu*xvec|\!)\langle N\rangle \prec P'$ **using** ‹$A_P \sharp* \Psi_F$› ‹$A_P \sharp* \Psi_Q$› ‹$A_P \sharp* P$› ‹$A_P \sharp* M$› ‹$A_P \sharp* K'$›
by(*force intro*: *outputRenameSubject*)
moreover have $\langle A_F, (\Psi_F \otimes \Psi_Q) \otimes \Psi_P\rangle \hookrightarrow_F \langle A_G, (\Psi_G \otimes \Psi_Q) \otimes \Psi_P\rangle$
proof −
have $\langle A_F, (\Psi_F \otimes \Psi_Q) \otimes \Psi_P\rangle \simeq_F \langle A_F, \Psi_F \otimes \Psi_P \otimes \Psi_Q\rangle$
by(*metis Associativity Composition AssertionStatEqSym AssertionStatEqTrans Commutativity frameResChainPres frameNilStatEq*)
moreover have $\langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q\rangle \simeq_F \langle A_G, (\Psi_G \otimes \Psi_Q) \otimes \Psi_P\rangle$
by(*metis Associativity Composition AssertionStatEqSym AssertionStatEqTrans Commutativity frameResChainPres frameNilStatEq*)
ultimately show *?thesis* **using** *FimpG*
by(*elim FrameStatEqImpCompose*)
qed
ultimately show *?thesis* **using** ‹$A_F \sharp* P$› ‹$A_G \sharp* P$› ‹$A_F \sharp* K'$› ‹$A_G \sharp* K'$› ‹$A_P \sharp* A_F$› ‹$A_P \sharp* A_G$› ‹$A_P \sharp* \Psi_F$› ‹$A_P \sharp* \Psi_G$› ‹$A_P \sharp* \Psi_Q$› *FrP* ‹*distinct* A_P› ‹*distinct xvec*›
by(*auto intro*: *transferNonTauFrame*)
qed

moreover from ‹$\Psi_F \otimes \Psi_Q \rhd P \longmapsto M(\!|\nu*xvec|\!)\langle N\rangle \prec P'$› **have** $\Psi_F \otimes \Psi_Q \rhd P \longmapsto ROut\ M\ ((\!|\nu*xvec|\!)N \prec' P')$
by(*simp add*: *residualInject*)
moreover with *FrP* ‹*distinct* A_P›
obtain M' **where** *MeqM'*: $(\Psi_F \otimes \Psi_Q) \otimes \Psi_P \vdash M \leftrightarrow M'$ **and** $A_Q \sharp* M'$ **and** $A_F \sharp* M'$ **and** $A_G \sharp* M'$
using ‹$A_Q \sharp* P$› ‹$A_P \sharp* A_F$› ‹$A_P \sharp* A_G$› ‹$A_F \sharp* P$› ‹$A_G \sharp* P$› ‹$A_P \sharp* \Psi_F$›

$\langle A_P \sharp* \Psi_Q\rangle$ $\langle A_P \sharp* A_Q\rangle$ $\langle A_P \sharp* P\rangle$ $\langle A_P \sharp* M\rangle$ $\langle xvec \sharp* M\rangle$ $\langle distinct\ xvec\rangle$
by(*elim outputObtainPrefix*[**where** $B{=}A_Q@A_F@A_G$]) *force*+

have $\Psi_G \otimes \Psi_P \rhd Q \longmapsto M'(\!|N|\!) \prec Q'$
proof −
from *MeqM'* **have** $\Psi_F \otimes (\Psi_Q \otimes \Psi_P) \vdash M \leftrightarrow M'$ **by**(*rule statEqEnt*[*OF Associativity*])
with $\langle \Psi_F \otimes (\Psi_P \otimes \Psi_Q) \vdash M \leftrightarrow K\rangle$ **have** $\Psi_F \otimes (\Psi_Q \otimes \Psi_P) \vdash K \leftrightarrow M'$
by(*blast intro*: *chanEqTrans chanEqSym compositionSym Commutativity statEqEnt*)
then have $(\Psi_F \otimes \Psi_P) \otimes \Psi_Q \vdash K \leftrightarrow M'$
by(*blast intro*: *statEqEnt AssertionStatEqSym Associativity AssertionStatEqTrans compositionSym Commutativity*)
with $\langle \Psi_F \otimes \Psi_P \rhd Q \longmapsto K(\!|N|\!) \prec Q'\rangle$ *FrQ* $\langle distinct\ A_Q\rangle$
have $\Psi_F \otimes \Psi_P \rhd Q \longmapsto M'(\!|N|\!) \prec Q'$ **using** $\langle A_Q \sharp* \Psi_F\rangle$ $\langle A_Q \sharp* \Psi_P\rangle$ $\langle A_Q \sharp* Q\rangle$ $\langle A_Q \sharp* K\rangle$ $\langle A_Q \sharp* M'\rangle$
by(*force intro*: *inputRenameSubject*)
moreover have $\langle A_F, (\Psi_F \otimes \Psi_P) \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_G, (\Psi_G \otimes \Psi_P) \otimes \Psi_Q\rangle$
proof −
have $\langle A_F, (\Psi_F \otimes \Psi_P) \otimes \Psi_Q\rangle \simeq_F \langle A_F, \Psi_F \otimes \Psi_P \otimes \Psi_Q\rangle$
by(*metis Associativity frameResChainPres frameNilStatEq*)
moreover have $\langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q\rangle \simeq_F \langle A_G, (\Psi_G \otimes \Psi_P) \otimes \Psi_Q\rangle$
by(*metis Associativity AssertionStatEqSym frameResChainPres frameNilStatEq*)
ultimately show *?thesis* **using** *FimpG*
by(*elim FrameStatEqImpCompose*)
qed

ultimately show *?thesis* **using** $\langle A_F \sharp* Q\rangle$ $\langle A_G \sharp* Q\rangle$ $\langle A_F \sharp* M'\rangle$ $\langle A_G \sharp* M'\rangle$ $\langle A_Q \sharp* A_F\rangle$ $\langle A_Q \sharp* A_G\rangle$ $\langle A_Q \sharp* \Psi_F\rangle$ $\langle A_Q \sharp* \Psi_G\rangle$ $\langle A_Q \sharp* \Psi_P\rangle$ *FrQ* $\langle distinct\ A_Q\rangle$ $\langle distinct\ xvec\rangle$
by(*auto intro*: *transferNonTauFrame*)
qed

moreover have $\Psi_G \otimes \Psi_P \otimes \Psi_Q \vdash K' \leftrightarrow M'$
proof −
from *MeqM'* **have** $\Psi_F \otimes \Psi_P \otimes \Psi_Q \vdash M' \leftrightarrow M$
by(*blast intro*: *chanEqSym Associativity statEqEnt Commutativity compositionSym*)
moreover from *KeqK'* **have** $\Psi_F \otimes \Psi_P \otimes \Psi_Q \vdash K \leftrightarrow K'$
by(*blast intro*: *chanEqSym Associativity statEqEnt Commutativity compositionSym*)
ultimately have $\Psi_F \otimes \Psi_P \otimes \Psi_Q \vdash K' \leftrightarrow M'$ **using** $\langle \Psi_F \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K\rangle$
by(*blast intro*: *chanEqSym chanEqTrans*)
then show *?thesis* **using** $\langle A_F \sharp* M'\rangle$ $\langle A_F \sharp* K'\rangle$ $\langle A_G \sharp* M'\rangle$ $\langle A_G \sharp* K'\rangle$ *FimpG*
apply(*simp add*: *FrameStatImp-def*)
apply(*erule allE*[**where** $x{=}SChanEq'\ K'\ M'$])

by(*force intro*: *frameImpI dest*: *frameImpE*)
qed

ultimately show *?case* **using** ‹$A_P \sharp* \Psi_G$› ‹$A_P \sharp* P$› ‹$A_P \sharp* Q$› ‹$A_P \sharp* A_Q$› ‹$A_P \sharp* K'$› ‹$A_Q \sharp* \Psi_G$› ‹$A_Q \sharp* P$› ‹$A_Q \sharp* Q$› ‹$A_Q \sharp* M'$› ‹*xvec* $\sharp*$ *Q*› *FrP FrQ*
by(*intro Comm2*) (*assumption* | *simp*)+
next
case(*cBrClose* Ψ_F P M *xvec* N P' A_P Ψ_P x Ψ_G A_F A_G)
from ‹$\Psi_F \rhd P \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$›
have *suppM*: ((*supp M*)::*name set*) ⊆ ((*supp P*)::*name set*)
by(*simp add*: *residualInject brOutputTermSupp*)

note ‹$\Psi_F \rhd P \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$› ‹*extractFrame P* = $\langle A_P, \Psi_P\rangle$›
‹*distinct* A_P› ‹*distinct xvec*› ‹$\langle A_F, \Psi_F \otimes \Psi_P\rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P\rangle$›

moreover from ‹$x \sharp A_F$› ‹$A_F \sharp* (\!|\nu x|\!)P$› ‹$x \sharp A_G$› ‹$A_G \sharp* (\!|\nu x|\!)P$›
have $A_F \sharp* P$ **and** $A_G \sharp* P$ **by** *simp*+
moreover with *suppM*
have $A_F \sharp* M$ **and** $A_G \sharp* M$
by(*auto simp add*: *fresh-star-def fresh-def*)
moreover note ‹$A_P \sharp* A_F$› ‹$A_P \sharp* A_G$› ‹$A_P \sharp* \Psi_F$› ‹$A_P \sharp* \Psi_G$›
ultimately have $\Psi_G \rhd P \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$
by(*simp add*: *transferNonTauFrame*)
with ‹$x \in$ *supp M*› ‹$x \sharp \Psi_G$›
show *?case*
by(*simp add*: *BrClose*)
next
case(*cScope* Ψ_F P P' x A_P Ψ_P Ψ_G A_F A_G)
then have $\Psi_G \rhd P \longmapsto\tau \prec P'$ **by** *auto*
with ‹$x \sharp \Psi_G$› **show** *?case*
by(*intro Scope*) *auto*
next
case(*cBang* Ψ_F P P' A_P Ψ_P Ψ_G A_F A_G)
from ‹$\Psi_P \simeq \mathbf{1}$› **have** $\langle A_F, \Psi_F \otimes \Psi_P \otimes \mathbf{1}\rangle \simeq_F \langle A_F, \Psi_F \otimes \mathbf{1}\rangle$
by(*metis frameIntCompositionSym Identity AssertionStatEqTrans*)
moreover note ‹$\langle A_F, \Psi_F \otimes \mathbf{1}\rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \mathbf{1}\rangle$›
moreover from ‹$\Psi_P \simeq \mathbf{1}$› **have** $\langle A_G, \Psi_G \otimes \mathbf{1}\rangle \simeq_F \langle A_G, \Psi_G \otimes \Psi_P \otimes \mathbf{1}\rangle$
by(*metis frameIntCompositionSym Identity AssertionStatEqTrans AssertionStatEqSym*)
ultimately have $\langle A_F, \Psi_F \otimes \Psi_P \otimes \mathbf{1}\rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \otimes \mathbf{1}\rangle$
by(*rule FrameStatEqImpCompose*)
with *cBang* **have** $\Psi_G \rhd P \parallel !P \longmapsto\tau \prec P'$ **by** *force*
then show *?case* **using** ‹*guarded P*› **by**(*rule Bang*)
qed

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 $A_F \sharp* \mathit{subject}\ \alpha$
and $A_G \sharp* \mathit{subject}\ \alpha$
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 $\Psi_G \rhd P \longmapsto \alpha \prec P'$
using *assms*
proof −
from ‹$\Psi_F \rhd P \longmapsto \alpha \prec P'$› **have** *distinct*(*bn* α) **by**(*auto dest*: *boundOutputDistinct*)
then show *?thesis* **using** *assms*
by(*cases* $\alpha = \tau$) (*auto intro*: *transferNonTauFrame transferTauFrame*)
qed

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*
using *Trans*
proof(*induct rule*: *parInputCases*[*of* - - - - - - (A_{PQ}, Ψ_{PQ})])
case(*cPar1 P' A_Q Ψ_Q*)
from ‹$A_Q \sharp* (A_{PQ}, \Psi_{PQ})$› **have** $A_Q \sharp* A_{PQ}$ **and** $A_Q \sharp* \Psi_{PQ}$ **by** *simp+*
obtain $A_P\ \Psi_P$ **where** *FrP*: *extractFrame* $P = \langle A_P, \Psi_P\rangle$ **and** *distinct* A_P $A_P \sharp* (P, Q, \Psi, M, A_Q, A_{PQ}, \Psi_Q)$
by(*rule freshFrame*)
then have $A_P \sharp* P$ **and** $A_P \sharp* Q$ **and** $A_P \sharp* \Psi$ **and** $A_P \sharp* M$ **and** $A_P \sharp* A_Q$ **and** $A_P \sharp* A_{PQ}$ **and** $A_P \sharp* \Psi_Q$
by *simp+*

have *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$ **by** *fact*

from ‹$A_Q \sharp* P$› ‹$A_P \sharp* A_Q$› *FrP* **have** $A_Q \sharp* \Psi_P$
by(*force dest*: *extractFrameFreshChain*)

from ‹*extractFrame*$(P \parallel Q) = \langle A_{PQ}, \Psi_{PQ}\rangle$› *FrP FrQ* ‹$A_P \sharp* A_Q$› ‹$A_P \sharp* \Psi_Q$› ‹$A_Q \sharp* \Psi_P$›
have $\langle (A_P@A_Q), \Psi_P \otimes \Psi_Q\rangle = \langle A_{PQ}, \Psi_{PQ}\rangle$ **by** *simp*
moreover from ‹*distinct* A_P› ‹*distinct* A_Q› ‹$A_P \sharp* A_Q$› **have** *distinct*$(A_P@A_Q)$
by(*auto simp add*: *fresh-star-def fresh-def name-list-supp*)
ultimately obtain p **where** S: *set* $p \subseteq$ *set*$(A_P@A_Q) \times$ *set*$((p \cdot A_P)@(p \cdot A_Q))$ **and** *distinctPerm p*
and Ψeq: $\Psi_{PQ} = (p \cdot \Psi_P) \otimes (p \cdot \Psi_Q)$ **and** *Aeq*: $A_{PQ} = (p \cdot A_P)@(p \cdot A_Q)$
using ‹$A_P \sharp* A_{PQ}$› ‹$A_Q \sharp* A_{PQ}$› ‹*distinct* A_{PQ}›
by(*elim frameChainEq'*) (*assumption* | *simp add*: *eqvts*)+

from ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|N|\!) \prec P'$› S ‹$A_{PQ} \sharp* P$› ‹$A_P \sharp* P$› ‹$A_Q \sharp* P$› ‹$A_{PQ} \sharp* M$› ‹$A_P \sharp* M$› ‹$A_Q \sharp* M$› *Aeq*
have $(p \cdot (\Psi \otimes \Psi_Q)) \rhd P \longmapsto M(\!|N|\!) \prec P'$
by(*elim inputPermFrame*) *auto*
with S ‹$A_{PQ} \sharp* \Psi$› ‹$A_P \sharp* \Psi$› ‹$A_Q \sharp* \Psi$› *Aeq* **have** $\Psi \otimes (p \cdot \Psi_Q) \rhd P \longmapsto M(\!|N|\!) \prec P'$
by(*simp add*: *eqvts*)
moreover from *FrP* **have** $(p \cdot$ *extractFrame* $P) = p \cdot \langle A_P, \Psi_P\rangle$ **by** *simp*
with S ‹$A_{PQ} \sharp* P$› ‹$A_P \sharp* P$› ‹$A_Q \sharp* P$› *Aeq* **have** *extractFrame* $P = \langle (p \cdot A_P), p \cdot \Psi_P\rangle$
by(*simp add*: *eqvts*)
moreover from *FrQ* **have** $(p \cdot$ *extractFrame* $Q) = p \cdot \langle A_Q, \Psi_Q\rangle$ **by** *simp*
with S ‹$A_{PQ} \sharp* Q$› ‹$A_P \sharp* Q$› ‹$A_Q \sharp* Q$› *Aeq* **have** *extractFrame* $Q = \langle (p \cdot A_Q), p \cdot \Psi_Q\rangle$

by(*simp add*: *eqvts*)
moreover from ‹*distinct* A_P› ‹*distinct* A_Q› **have** *distinct*($p \cdot A_P$) **and** *distinct*($p \cdot A_Q$)
by *simp*+
moreover from ‹$A_P \sharp* A_Q$› **have** $(p \cdot A_P) \sharp* (p \cdot A_Q)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
moreover from ‹$A_P \sharp* \Psi_Q$› **have** $(p \cdot A_P) \sharp* (p \cdot \Psi_Q)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
moreover from ‹$A_Q \sharp* \Psi_P$› **have** $(p \cdot A_Q) \sharp* (p \cdot \Psi_P)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
ultimately show *?case* **using** ‹$A_{PQ} \sharp* \Psi$› ‹$A_{PQ} \sharp* P$› ‹$A_{PQ} \sharp* Q$› ‹$A_{PQ} \sharp* M$› *Aeq* Ψ*eq*
by(*intro rPar1*) (*assumption* | *simp*)+
next
case(*cPar2* Q' A_P Ψ_P)
from ‹$A_P \sharp* (A_{PQ}, \Psi_{PQ})$› **have** $A_P \sharp* A_{PQ}$ **and** $A_P \sharp* \Psi_{PQ}$ **by** *simp*+
obtain A_Q Ψ_Q **where** *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$ **and** *distinct* A_Q $A_Q \sharp* (P, Q, \Psi, M, A_P, A_{PQ}, \Psi_P)$
by(*rule freshFrame*)
then have $A_Q \sharp* P$ **and** $A_Q \sharp* Q$ **and** $A_Q \sharp* \Psi$ **and** $A_Q \sharp* M$ **and** $A_Q \sharp* A_P$ **and** $A_Q \sharp* A_{PQ}$ **and** $A_Q \sharp* \Psi_P$
by *simp*+

have *FrP*: *extractFrame* $P = \langle A_P, \Psi_P\rangle$ **by** *fact*

from ‹$A_P \sharp* Q$› ‹$A_Q \sharp* A_P$› *FrQ* **have** $A_P \sharp* \Psi_Q$
by(*force dest*: *extractFrameFreshChain*)

from ‹*extractFrame*($P \parallel Q$) = $\langle A_{PQ}, \Psi_{PQ}\rangle$› *FrP FrQ* ‹$A_Q \sharp* A_P$› ‹$A_P \sharp* \Psi_Q$› ‹$A_Q \sharp* \Psi_P$›
have $\langle (A_P @ A_Q), \Psi_P \otimes \Psi_Q\rangle = \langle A_{PQ}, \Psi_{PQ}\rangle$ **by** *simp*
moreover from ‹*distinct* A_P› ‹*distinct* A_Q› ‹$A_Q \sharp* A_P$› **have** *distinct*($A_P @ A_Q$)
by(*auto simp add*: *fresh-star-def fresh-def name-list-supp*)
ultimately obtain p **where** S: *set* $p \subseteq$ *set*($A_P @ A_Q$) $\times$ *set*($(p \cdot A_P) @ (p \cdot A_Q)$) **and** *distinctPerm* p
and Ψ*eq*: $\Psi_{PQ} = (p \cdot \Psi_P) \otimes (p \cdot \Psi_Q)$ **and** *Aeq*: $A_{PQ} = (p \cdot A_P) @ (p \cdot A_Q)$
using ‹$A_P \sharp* A_{PQ}$› ‹$A_Q \sharp* A_{PQ}$› ‹*distinct* A_{PQ}›
by(*elim frameChainEq'*) (*assumption* | *simp add*: *eqvts*)+

from ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto M(\!|N|\!) \prec Q'$› S ‹$A_{PQ} \sharp* Q$› ‹$A_P \sharp* Q$› ‹$A_Q \sharp* Q$› ‹$A_{PQ} \sharp* M$› ‹$A_P \sharp* M$› ‹$A_Q \sharp* M$› *Aeq*
have $(p \cdot (\Psi \otimes \Psi_P)) \rhd Q \longmapsto M(\!|N|\!) \prec Q'$
by(*elim inputPermFrame*) *auto*
with S ‹$A_{PQ} \sharp* \Psi$› ‹$A_P \sharp* \Psi$› ‹$A_Q \sharp* \Psi$› *Aeq* **have** $\Psi \otimes (p \cdot \Psi_P) \rhd Q \longmapsto M(\!|N|\!) \prec Q'$
by(*simp add*: *eqvts*)
moreover from *FrP* **have** $(p \cdot$ *extractFrame* $P) = p \cdot \langle A_P, \Psi_P\rangle$ **by** *simp*
with S ‹$A_{PQ} \sharp* P$› ‹$A_P \sharp* P$› ‹$A_Q \sharp* P$› *Aeq* **have** *extractFrame* $P = \langle (p \cdot A_P), p \cdot \Psi_P\rangle$

by(*simp add*: *eqvts*)
moreover from *FrQ* **have** $(p \cdot \mathit{extractFrame}\ Q) = p \cdot \langle A_Q, \Psi_Q\rangle$ **by** *simp*
with S ‹$A_{PQ} \sharp* Q$› ‹$A_P \sharp* Q$› ‹$A_Q \sharp* Q$› *Aeq* **have** $\mathit{extractFrame}\ Q = \langle(p \cdot A_Q), p \cdot \Psi_Q\rangle$
by(*simp add*: *eqvts*)
moreover from ‹*distinct* A_P› ‹*distinct* A_Q› **have** $\mathit{distinct}(p \cdot A_P)$ **and** $\mathit{distinct}(p \cdot A_Q)$
by *simp*+
moreover from ‹$A_Q \sharp* A_P$› **have** $(p \cdot A_P) \sharp* (p \cdot A_Q)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
moreover from ‹$A_P \sharp* \Psi_Q$› **have** $(p \cdot A_P) \sharp* (p \cdot \Psi_Q)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
moreover from ‹$A_Q \sharp* \Psi_P$› **have** $(p \cdot A_Q) \sharp* (p \cdot \Psi_P)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
ultimately show *?case* **using** ‹$A_{PQ} \sharp* \Psi$› ‹$A_{PQ} \sharp* P$› ‹$A_{PQ} \sharp* Q$› ‹$A_{PQ} \sharp* M$› *Aeq* *Ψeq*
by(*intro rPar2*) (*assumption* | *simp*)+
qed

lemma *parCasesBrInputFrame*[*consumes* 7, *case-names cPar1 cPar2 cBrMerge*]:
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 $\mathit{extractFrame}(P \parallel Q) = \langle A_{PQ}, \Psi_{PQ}\rangle$
and $\mathit{distinct}\ A_{PQ}$
and $A_{PQ} \sharp* \Psi$
and $A_{PQ} \sharp* P$
and $A_{PQ} \sharp* Q$
and $A_{PQ} \sharp* M$
and *rPar1*: $\bigwedge P'\ A_P\ \Psi_P\ A_Q\ \Psi_Q.\ [\![\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M《N》 \prec P';\ \mathit{extractFrame}\ P = \langle A_P, \Psi_P\rangle;\ \mathit{extractFrame}\ Q = \langle A_Q, \Psi_Q\rangle;$
$\mathit{distinct}\ A_P;\ \mathit{distinct}\ A_Q;\ A_P \sharp* \Psi;\ A_P \sharp* P;\ A_P \sharp* Q;\ A_P \sharp* M;\ A_Q \sharp* \Psi;\ A_Q \sharp* P;\ A_Q \sharp* Q;\ A_Q \sharp* M;$
$A_P \sharp* \Psi_Q;\ A_Q \sharp* \Psi_P;\ A_P \sharp* A_Q;\ A_{PQ} = A_P @ A_Q;\ \Psi_{PQ} = \Psi_P \otimes \Psi_Q]\!] \Longrightarrow \mathit{Prop}\ (P' \parallel Q)$
and *rPar2*: $\bigwedge Q'\ A_P\ \Psi_P\ A_Q\ \Psi_Q.\ [\![\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M《N》 \prec Q';\ \mathit{extractFrame}\ P = \langle A_P, \Psi_P\rangle;\ \mathit{extractFrame}\ Q = \langle A_Q, \Psi_Q\rangle;$
$\mathit{distinct}\ A_P;\ \mathit{distinct}\ A_Q;\ A_P \sharp* \Psi;\ A_P \sharp* P;\ A_P \sharp* Q;\ A_P \sharp* M;\ A_Q \sharp* \Psi;\ A_Q \sharp* P;\ A_Q \sharp* Q;\ A_Q \sharp* M;$
$A_P \sharp* \Psi_Q;\ A_Q \sharp* \Psi_P;\ A_P \sharp* A_Q;\ A_{PQ} = A_P @ A_Q;\ \Psi_{PQ} = \Psi_P \otimes \Psi_Q]\!] \Longrightarrow \mathit{Prop}\ (P \parallel Q')$
and *rBrMerge*: $\bigwedge \Psi_Q\ P'\ A_P\ \Psi_P\ Q'\ A_Q.$

$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$; *extractFrame* $P = \langle A_P, \Psi_P\rangle$; *distinct* A_P;
$\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$; *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$; *distinct* A_Q;
$A_P \sharp* \Psi$; $A_P \sharp* \Psi_Q$; $A_P \sharp* P$;
$A_P \sharp* Q$; $A_P \sharp* A_Q$;
$A_Q \sharp* \Psi$; $A_Q \sharp* \Psi_P$; $A_Q \sharp* P$;
$A_Q \sharp* Q$;
$A_{PQ} = A_P @ A_Q$; $\Psi_{PQ} = \Psi_P \otimes \Psi_Q]\!] \Longrightarrow$
Prop $(P' \parallel Q')$

shows *Prop T*
using *Trans*
proof(*induct rule*: *parBrInputCases*[*of* - - - - - - (A_{PQ}, Ψ_{PQ})])
case(*cPar1* P' A_Q Ψ_Q)
from ‹$A_Q \sharp* (A_{PQ}, \Psi_{PQ})$› **have** $A_Q \sharp* A_{PQ}$ **and** $A_Q \sharp* \Psi_{PQ}$ **by** *simp+*
obtain A_P Ψ_P **where** *FrP*: *extractFrame* $P = \langle A_P, \Psi_P\rangle$ **and** *distinct* A_P
$A_P \sharp* (P, Q, \Psi, M, A_Q, A_{PQ}, \Psi_Q)$
by(*rule freshFrame*)
then have $A_P \sharp* P$ **and** $A_P \sharp* Q$ **and** $A_P \sharp* \Psi$ **and** $A_P \sharp* M$ **and** $A_P \sharp* A_Q$ **and** $A_P \sharp* A_{PQ}$ **and** $A_P \sharp* \Psi_Q$
by *simp+*

have *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$ **by** *fact*

from ‹$A_Q \sharp* P$› ‹$A_P \sharp* A_Q$› *FrP* **have** $A_Q \sharp* \Psi_P$
by(*force dest*: *extractFrameFreshChain*)

from ‹*extractFrame*$(P \parallel Q) = \langle A_{PQ}, \Psi_{PQ}\rangle$› *FrP FrQ* ‹$A_P \sharp* A_Q$› ‹$A_P \sharp* \Psi_Q$› ‹$A_Q \sharp* \Psi_P$›
have $\langle (A_P @ A_Q), \Psi_P \otimes \Psi_Q\rangle = \langle A_{PQ}, \Psi_{PQ}\rangle$ **by** *simp*
moreover from ‹*distinct* A_P› ‹*distinct* A_Q› ‹$A_P \sharp* A_Q$› **have** *distinct*$(A_P @ A_Q)$
by(*auto simp add*: *fresh-star-def fresh-def name-list-supp*)
ultimately obtain p **where** S: *set* $p \subseteq$ *set*$(A_P @ A_Q) \times$ *set*$((p \cdot A_P) @ (p \cdot A_Q))$ **and** *distinctPerm p*
and Ψeq: $\Psi_{PQ} = (p \cdot \Psi_P) \otimes (p \cdot \Psi_Q)$ **and** *Aeq*: $A_{PQ} = (p \cdot A_P) @ (p \cdot A_Q)$
using ‹$A_P \sharp* A_{PQ}$› ‹$A_Q \sharp* A_{PQ}$› ‹*distinct* A_{PQ}›
by(*elim frameChainEq'*) (*assumption* | *simp add*: *eqvts*)+

from ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$› S ‹$A_{PQ} \sharp* P$› ‹$A_P \sharp* P$› ‹$A_Q \sharp* P$› ‹$A_{PQ} \sharp* M$› ‹$A_P \sharp* M$› ‹$A_Q \sharp* M$› *Aeq*
have $(p \cdot (\Psi \otimes \Psi_Q)) \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$
by(*elim brinputPermFrame*) *auto*
with S ‹$A_{PQ} \sharp* \Psi$› ‹$A_P \sharp* \Psi$› ‹$A_Q \sharp* \Psi$› *Aeq* **have** $\Psi \otimes (p \cdot \Psi_Q) \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$
by(*simp add*: *eqvts*)
moreover from *FrP* **have** $(p \cdot$ *extractFrame* $P) = p \cdot \langle A_P, \Psi_P\rangle$ **by** *simp*
with S ‹$A_{PQ} \sharp* P$› ‹$A_P \sharp* P$› ‹$A_Q \sharp* P$› *Aeq* **have** *extractFrame* $P = \langle (p \cdot A_P), p \cdot \Psi_P\rangle$
by(*simp add*: *eqvts*)

moreover from *FrQ* **have** $(p \cdot \mathit{extractFrame}\ Q) = p \cdot \langle A_Q, \Psi_Q\rangle$ **by** *simp*
with S ‹$A_{PQ} \sharp* Q$› ‹$A_P \sharp* Q$› ‹$A_Q \sharp* Q$› *Aeq* **have** $\mathit{extractFrame}\ Q = \langle(p \cdot A_Q), p \cdot \Psi_Q\rangle$
by(*simp add*: *eqvts*)
moreover from ‹*distinct* A_P› ‹*distinct* A_Q› **have** $\mathit{distinct}(p \cdot A_P)$ **and** $\mathit{distinct}(p \cdot A_Q)$
by *simp+*
moreover from ‹$A_P \sharp* A_Q$› **have** $(p \cdot A_P) \sharp* (p \cdot A_Q)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
moreover from ‹$A_P \sharp* \Psi_Q$› **have** $(p \cdot A_P) \sharp* (p \cdot \Psi_Q)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
moreover from ‹$A_Q \sharp* \Psi_P$› **have** $(p \cdot A_Q) \sharp* (p \cdot \Psi_P)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
ultimately show *?case* **using** ‹$A_{PQ} \sharp* \Psi$› ‹$A_{PQ} \sharp* P$› ‹$A_{PQ} \sharp* Q$› ‹$A_{PQ} \sharp* M$› *Aeq* Ψ*eq*
by(*intro rPar1*) (*assumption* | *simp*)+
next
case(*cPar2* Q' A_P Ψ_P)
from ‹$A_P \sharp* (A_{PQ}, \Psi_{PQ})$› **have** $A_P \sharp* A_{PQ}$ **and** $A_P \sharp* \Psi_{PQ}$ **by** *simp+*
obtain A_Q Ψ_Q **where** *FrQ*: $\mathit{extractFrame}\ Q = \langle A_Q, \Psi_Q\rangle$ **and** *distinct* A_Q $A_Q \sharp* (P, Q, \Psi, M, A_P, A_{PQ}, \Psi_P)$
by(*rule freshFrame*)
then have $A_Q \sharp* P$ **and** $A_Q \sharp* Q$ **and** $A_Q \sharp* \Psi$ **and** $A_Q \sharp* M$ **and** $A_Q \sharp* A_P$ **and** $A_Q \sharp* A_{PQ}$ **and** $A_Q \sharp* \Psi_P$
by *simp+*

have *FrP*: $\mathit{extractFrame}\ P = \langle A_P, \Psi_P\rangle$ **by** *fact*

from ‹$A_P \sharp* Q$› ‹$A_Q \sharp* A_P$› *FrQ* **have** $A_P \sharp* \Psi_Q$
by(*force dest*: *extractFrameFreshChain*)

from ‹$\mathit{extractFrame}(P \parallel Q) = \langle A_{PQ}, \Psi_{PQ}\rangle$› *FrP FrQ* ‹$A_Q \sharp* A_P$› ‹$A_P \sharp* \Psi_Q$› ‹$A_Q \sharp* \Psi_P$›
have $\langle(A_P @ A_Q), \Psi_P \otimes \Psi_Q\rangle = \langle A_{PQ}, \Psi_{PQ}\rangle$ **by** *simp*
moreover from ‹*distinct* A_P› ‹*distinct* A_Q› ‹$A_Q \sharp* A_P$› **have** $\mathit{distinct}(A_P @ A_Q)$
by(*auto simp add*: *fresh-star-def fresh-def name-list-supp*)
ultimately obtain p **where** S: $\mathit{set}\ p \subseteq \mathit{set}(A_P @ A_Q) \times \mathit{set}((p \cdot A_P) @ (p \cdot A_Q))$ **and** *distinctPerm* p
and Ψ*eq*: $\Psi_{PQ} = (p \cdot \Psi_P) \otimes (p \cdot \Psi_Q)$ **and** *Aeq*: $A_{PQ} = (p \cdot A_P) @ (p \cdot A_Q)$
using ‹$A_P \sharp* A_{PQ}$› ‹$A_Q \sharp* A_{PQ}$› ‹*distinct* A_{PQ}›
by(*elim frameChainEq'*) (*assumption* | *simp add*: *eqvts*)+

from ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$› S ‹$A_{PQ} \sharp* Q$› ‹$A_P \sharp* Q$› ‹$A_Q \sharp* Q$› ‹$A_{PQ} \sharp* M$› ‹$A_P \sharp* M$› ‹$A_Q \sharp* M$› *Aeq*
have $(p \cdot (\Psi \otimes \Psi_P)) \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$
by(*elim brinputPermFrame*) *auto*
with S ‹$A_{PQ} \sharp* \Psi$› ‹$A_P \sharp* \Psi$› ‹$A_Q \sharp* \Psi$› *Aeq* **have** $\Psi \otimes (p \cdot \Psi_P) \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$
by(*simp add*: *eqvts*)

moreover from *FrP* **have** $(p \cdot \mathit{extractFrame}\ P) = p \cdot \langle A_P, \Psi_P\rangle$ **by** *simp*
with S ‹$A_{PQ} \sharp* P$› ‹$A_P \sharp* P$› ‹$A_Q \sharp* P$› *Aeq* **have** $\mathit{extractFrame}\ P = \langle(p \cdot A_P), p \cdot \Psi_P\rangle$
by(*simp add*: *eqvts*)
moreover from *FrQ* **have** $(p \cdot \mathit{extractFrame}\ Q) = p \cdot \langle A_Q, \Psi_Q\rangle$ **by** *simp*
with S ‹$A_{PQ} \sharp* Q$› ‹$A_P \sharp* Q$› ‹$A_Q \sharp* Q$› *Aeq* **have** $\mathit{extractFrame}\ Q = \langle(p \cdot A_Q), p \cdot \Psi_Q\rangle$
by(*simp add*: *eqvts*)
moreover from ‹*distinct* A_P› ‹*distinct* A_Q› **have** $\mathit{distinct}(p \cdot A_P)$ **and** $\mathit{distinct}(p \cdot A_Q)$
by *simp*+
moreover from ‹$A_Q \sharp* A_P$› **have** $(p \cdot A_P) \sharp* (p \cdot A_Q)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
moreover from ‹$A_P \sharp* \Psi_Q$› **have** $(p \cdot A_P) \sharp* (p \cdot \Psi_Q)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
moreover from ‹$A_Q \sharp* \Psi_P$› **have** $(p \cdot A_Q) \sharp* (p \cdot \Psi_P)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
ultimately show *?case* **using** ‹$A_{PQ} \sharp* \Psi$› ‹$A_{PQ} \sharp* P$› ‹$A_{PQ} \sharp* Q$› ‹$A_{PQ} \sharp* M$› *Aeq* Ψ*eq*
by(*intro rPar2*) (*assumption* | *simp*)+
next
case(*cBrMerge* Ψ_Q P' A_P Ψ_P Q' A_Q)
then have *FrP*: $\mathit{extractFrame}\ P = \langle A_P, \Psi_P\rangle$ **and** *FrQ*: $\mathit{extractFrame}\ Q = \langle A_Q, \Psi_Q\rangle$
and $A_P \sharp* A_{PQ}$ **and** $A_Q \sharp* A_{PQ}$
by *simp*+

from ‹$\mathit{extractFrame}(P \parallel Q) = \langle A_{PQ}, \Psi_{PQ}\rangle$› *FrP FrQ* ‹$A_P \sharp* A_Q$› ‹$A_P \sharp* \Psi_Q$› ‹$A_Q \sharp* \Psi_P$›
have $\langle(A_P @ A_Q), \Psi_P \otimes \Psi_Q\rangle = \langle A_{PQ}, \Psi_{PQ}\rangle$ **by** *simp*
moreover from ‹*distinct* A_P› ‹*distinct* A_Q› ‹$A_P \sharp* A_Q$› **have** $\mathit{distinct}(A_P @ A_Q)$
by(*auto simp add*: *fresh-star-def fresh-def name-list-supp*)
ultimately obtain p **where** S: $\mathit{set}\ p \subseteq \mathit{set}(A_P @ A_Q) \times \mathit{set}((p \cdot A_P) @ (p \cdot A_Q))$ **and** *distinctPerm p*
and Ψ*eq*: $\Psi_{PQ} = (p \cdot \Psi_P) \otimes (p \cdot \Psi_Q)$ **and** *Aeq*: $A_{PQ} = (p \cdot A_P) @ (p \cdot A_Q)$
using ‹$A_P \sharp* A_{PQ}$› ‹$A_Q \sharp* A_{PQ}$› ‹*distinct* A_{PQ}›
by(*elim frameChainEq'*) (*assumption* | *simp add*: *eqvts*)+

from ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$› S ‹$A_{PQ} \sharp* P$› ‹$A_P \sharp* P$› ‹$A_Q \sharp* P$› ‹$A_{PQ} \sharp* M$› ‹$A_P \sharp* M$› ‹$A_Q \sharp* M$› *Aeq*
have $(p \cdot (\Psi \otimes \Psi_Q)) \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$
by(*elim brinputPermFrame*) *auto*
with S ‹$A_{PQ} \sharp* \Psi$› ‹$A_P \sharp* \Psi$› ‹$A_Q \sharp* \Psi$› *Aeq* **have** $\Psi \otimes (p \cdot \Psi_Q) \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$
by(*simp add*: *eqvts*)

from ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$› S ‹$A_{PQ} \sharp* Q$› ‹$A_Q \sharp* Q$› ‹$A_P \sharp* Q$› ‹$A_{PQ} \sharp* M$› ‹$A_Q \sharp* M$› ‹$A_P \sharp* M$› *Aeq*
have $(p \cdot (\Psi \otimes \Psi_P)) \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$

by(*elim brinputPermFrame*) *auto*
with S ‹A_{PQ} ♯* Ψ› ‹A_P ♯* Ψ› ‹A_Q ♯* Ψ› *Aeq* **have** $\Psi \otimes (p \cdot \Psi_P) \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$
by(*simp add*: *eqvts*)

note ‹$\Psi \otimes (p \cdot \Psi_Q) \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$› ‹$\Psi \otimes (p \cdot \Psi_P) \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$›
moreover from *FrP* **have** $(p \cdot extractFrame\ P) = p \cdot \langle A_P, \Psi_P\rangle$ **by** *simp*
with S ‹A_{PQ} ♯* P› ‹A_P ♯* P› ‹A_Q ♯* P› *Aeq* **have** $extractFrame\ P = \langle(p \cdot A_P), p \cdot \Psi_P\rangle$
by(*simp add*: *eqvts*)
moreover from *FrQ* **have** $(p \cdot extractFrame\ Q) = p \cdot \langle A_Q, \Psi_Q\rangle$ **by** *simp*
with S ‹A_{PQ} ♯* Q› ‹A_P ♯* Q› ‹A_Q ♯* Q› *Aeq* **have** $extractFrame\ Q = \langle(p \cdot A_Q), p \cdot \Psi_Q\rangle$
by(*simp add*: *eqvts*)
moreover from ‹*distinct* A_P› ‹*distinct* A_Q› **have** $distinct(p \cdot A_P)$ **and** $distinct(p \cdot A_Q)$
by *simp+*
moreover from ‹A_P ♯* A_Q› **have** $(p \cdot A_P)$ ♯* $(p \cdot A_Q)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
moreover from ‹A_P ♯* Ψ_Q› **have** $(p \cdot A_P)$ ♯* $(p \cdot \Psi_Q)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
moreover from ‹A_Q ♯* Ψ_P› **have** $(p \cdot A_Q)$ ♯* $(p \cdot \Psi_P)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
ultimately show *?case* **using** ‹A_{PQ} ♯* Ψ› ‹A_{PQ} ♯* P› ‹A_{PQ} ♯* Q› ‹A_{PQ} ♯* M› *Aeq* Ψ*eq*
by(*intro rBrMerge*) *simp+*
qed

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* ♯* Ψ
and *xvec* ♯* P
and *xvec* ♯* Q
and *xvec* ♯* M
and $extractFrame(P \parallel Q) = \langle A_{PQ}, \Psi_{PQ}\rangle$
and *distinct* A_{PQ}
and A_{PQ} ♯* Ψ
and A_{PQ} ♯* P
and A_{PQ} ♯* 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
using *Trans* ‹*xvec* $\sharp*$ Ψ› ‹*xvec* $\sharp*$ P› ‹*xvec* $\sharp*$ Q› ‹*xvec* $\sharp*$ M›
proof(*induct rule*: *parOutputCases*[*of* - - - - - - - (A_{PQ}, Ψ_{PQ})])
case(*cPar1* $P'\ A_Q\ \Psi_Q$)
from ‹$A_Q \sharp* (A_{PQ}, \Psi_{PQ})$› **have** $A_Q \sharp* A_{PQ}$ **and** $A_Q \sharp* \Psi_{PQ}$ **by** *simp*+
obtain $A_P\ \Psi_P$ **where** *FrP*: *extractFrame* $P = \langle A_P, \Psi_P\rangle$ **and** *distinct* A_P $A_P \sharp* (P, Q, \Psi, M, A_Q, A_{PQ}, \Psi_Q)$
by(*rule freshFrame*)
then have $A_P \sharp* P$ **and** $A_P \sharp* Q$ **and** $A_P \sharp* \Psi$ **and** $A_P \sharp* M$ **and** $A_P \sharp* A_Q$ **and** $A_P \sharp* A_{PQ}$ **and** $A_P \sharp* \Psi_Q$
by *simp*+

have *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$ **by** *fact*

from ‹$A_Q \sharp* P$› ‹$A_P \sharp* A_Q$› *FrP* **have** $A_Q \sharp* \Psi_P$
by(*force dest*: *extractFrameFreshChain*)

from ‹*extractFrame*$(P \parallel Q) = \langle A_{PQ}, \Psi_{PQ}\rangle$› *FrP FrQ* ‹$A_P \sharp* A_Q$› ‹$A_P \sharp* \Psi_Q$› ‹$A_Q \sharp* \Psi_P$›
have $\langle (A_P @ A_Q), \Psi_P \otimes \Psi_Q\rangle = \langle A_{PQ}, \Psi_{PQ}\rangle$ **by** *simp*
moreover from ‹*distinct* A_P› ‹*distinct* A_Q› ‹$A_P \sharp* A_Q$› **have** *distinct*$(A_P @ A_Q)$
by(*auto simp add*: *fresh-star-def fresh-def name-list-supp*)
ultimately obtain p **where** S: *set* $p \subseteq$ *set*$(A_P @ A_Q) \times$ *set*$((p \cdot A_P) @ (p \cdot A_Q))$ **and** *distinctPerm* p
and Ψeq: $\Psi_{PQ} = (p \cdot \Psi_P) \otimes (p \cdot \Psi_Q)$ **and** *Aeq*: $A_{PQ} = (p \cdot A_P) @ (p \cdot A_Q)$
using ‹$A_P \sharp* A_{PQ}$› ‹$A_Q \sharp* A_{PQ}$› ‹*distinct* A_{PQ}›
by(*elim frameChainEq'*) (*assumption* | *simp add*: *eqvts*)+

from ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|\nu* xvec|\!)\langle N\rangle \prec P'$› S ‹$A_{PQ} \sharp* P$› ‹$A_P \sharp* P$› ‹$A_Q \sharp* P$› ‹$A_{PQ} \sharp* M$› ‹$A_P \sharp* M$› ‹$A_Q \sharp* M$› *Aeq*
have $(p \cdot (\Psi \otimes \Psi_Q)) \rhd P \longmapsto M(\!|\nu* xvec|\!)\langle N\rangle \prec P'$
by(*elim outputPermFrame*) (*assumption* | *simp*)+

with S ‹$A_{PQ} \sharp* \Psi$› ‹$A_P \sharp* \Psi$› ‹$A_Q \sharp* \Psi$› *Aeq* **have** $\Psi \otimes (p \cdot \Psi_Q) \rhd P \longmapsto M(\!|\nu* xvec|\!)\langle N\rangle \prec P'$
by(*simp add*: *eqvts*)

moreover from *FrP* **have** $(p \cdot \mathit{extractFrame}\ P) = p \cdot \langle A_P, \Psi_P\rangle$ **by** *simp*
with S ‹$A_{PQ} \sharp* P$› ‹$A_P \sharp* P$› ‹$A_Q \sharp* P$› *Aeq* **have** $\mathit{extractFrame}\ P = \langle(p \cdot A_P), p \cdot \Psi_P\rangle$
by(*simp add*: *eqvts*)
moreover from *FrQ* **have** $(p \cdot \mathit{extractFrame}\ Q) = p \cdot \langle A_Q, \Psi_Q\rangle$ **by** *simp*
with S ‹$A_{PQ} \sharp* Q$› ‹$A_P \sharp* Q$› ‹$A_Q \sharp* Q$› *Aeq* **have** $\mathit{extractFrame}\ Q = \langle(p \cdot A_Q), p \cdot \Psi_Q\rangle$
by(*simp add*: *eqvts*)
moreover from ‹*distinct* A_P› ‹*distinct* A_Q› **have** $\mathit{distinct}(p \cdot A_P)$ **and** $\mathit{distinct}(p \cdot A_Q)$
by *simp*+
moreover from ‹$A_P \sharp* A_Q$› **have** $(p \cdot A_P) \sharp* (p \cdot A_Q)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
moreover from ‹$A_P \sharp* \Psi_Q$› **have** $(p \cdot A_P) \sharp* (p \cdot \Psi_Q)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
moreover from ‹$A_Q \sharp* \Psi_P$› **have** $(p \cdot A_Q) \sharp* (p \cdot \Psi_P)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
ultimately show *?case* **using** ‹$A_{PQ} \sharp* \Psi$› ‹$A_{PQ} \sharp* P$› ‹$A_{PQ} \sharp* Q$› ‹$A_{PQ} \sharp* M$› *Aeq* Ψ*eq*
by(*intro rPar1*) (*assumption* | *simp*)+
next
case(*cPar2* Q' A_P Ψ_P)
from ‹$A_P \sharp* (A_{PQ}, \Psi_{PQ})$› **have** $A_P \sharp* A_{PQ}$ **and** $A_P \sharp* \Psi_{PQ}$ **by** *simp*+
obtain A_Q Ψ_Q **where** *FrQ*: $\mathit{extractFrame}\ Q = \langle A_Q, \Psi_Q\rangle$ **and** *distinct* A_Q $A_Q \sharp* (P, Q, \Psi, M, A_P, A_{PQ}, \Psi_P)$
by(*rule freshFrame*)
then have $A_Q \sharp* P$ **and** $A_Q \sharp* Q$ **and** $A_Q \sharp* \Psi$ **and** $A_Q \sharp* M$ **and** $A_Q \sharp* A_P$ **and** $A_Q \sharp* A_{PQ}$ **and** $A_Q \sharp* \Psi_P$
by *simp*+

have *FrP*: $\mathit{extractFrame}\ P = \langle A_P, \Psi_P\rangle$ **by** *fact*

from ‹$A_P \sharp* Q$› ‹$A_Q \sharp* A_P$› *FrQ* **have** $A_P \sharp* \Psi_Q$
by(*force dest*: *extractFrameFreshChain*)

from ‹$\mathit{extractFrame}(P \parallel Q) = \langle A_{PQ}, \Psi_{PQ}\rangle$› *FrP FrQ* ‹$A_Q \sharp* A_P$› ‹$A_P \sharp* \Psi_Q$› ‹$A_Q \sharp* \Psi_P$›
have $\langle(A_P @ A_Q), \Psi_P \otimes \Psi_Q\rangle = \langle A_{PQ}, \Psi_{PQ}\rangle$ **by** *simp*
moreover from ‹*distinct* A_P› ‹*distinct* A_Q› ‹$A_Q \sharp* A_P$› **have** $\mathit{distinct}(A_P @ A_Q)$
by(*auto simp add*: *fresh-star-def fresh-def name-list-supp*)
ultimately obtain p **where** S: $\mathit{set}\ p \subseteq \mathit{set}(A_P @ A_Q) \times \mathit{set}((p \cdot A_P) @ (p \cdot A_Q))$ **and** *distinctPerm* p
and Ψ*eq*: $\Psi_{PQ} = (p \cdot \Psi_P) \otimes (p \cdot \Psi_Q)$ **and** *Aeq*: $A_{PQ} = (p \cdot A_P) @ (p \cdot A_Q)$
using ‹$A_P \sharp* A_{PQ}$› ‹$A_Q \sharp* A_{PQ}$› ‹*distinct* A_{PQ}›
by(*elim frameChainEq'*) (*assumption* | *simp add*: *eqvts*)+

from ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto M\langle\!|\nu* xvec|\!\rangle\langle N\rangle \prec Q'$› S ‹$A_{PQ} \sharp* Q$› ‹$A_P \sharp* Q$› ‹$A_Q \sharp* Q$› ‹$A_{PQ} \sharp* M$› ‹$A_P \sharp* M$› ‹$A_Q \sharp* M$› *Aeq*
have $(p \cdot (\Psi \otimes \Psi_P)) \rhd Q \longmapsto M\langle\!|\nu* xvec|\!\rangle\langle N\rangle \prec Q'$

by(*elim outputPermFrame*) (*assumption* | *simp*)+
with S ‹A_{PQ} ♯* Ψ› ‹A_P ♯* Ψ› ‹A_Q ♯* Ψ› *Aeq* **have** $\Psi \otimes (p \cdot \Psi_P) \rhd Q \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec Q'$
by(*simp add*: *eqvts*)
moreover from *FrP* **have** $(p \cdot extractFrame\ P) = p \cdot \langle A_P, \Psi_P\rangle$ **by** *simp*
with S ‹A_{PQ} ♯* P› ‹A_P ♯* P› ‹A_Q ♯* P› *Aeq* **have** $extractFrame\ P = \langle (p \cdot A_P), p \cdot \Psi_P\rangle$
by(*simp add*: *eqvts*)
moreover from *FrQ* **have** $(p \cdot extractFrame\ Q) = p \cdot \langle A_Q, \Psi_Q\rangle$ **by** *simp*
with S ‹A_{PQ} ♯* Q› ‹A_P ♯* Q› ‹A_Q ♯* Q› *Aeq* **have** $extractFrame\ Q = \langle (p \cdot A_Q), p \cdot \Psi_Q\rangle$
by(*simp add*: *eqvts*)
moreover from ‹*distinct* A_P› ‹*distinct* A_Q› **have** $distinct(p \cdot A_P)$ **and** $distinct(p \cdot A_Q)$
by *simp*+
moreover from ‹A_Q ♯* A_P› **have** $(p \cdot A_P)$ ♯* $(p \cdot A_Q)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
moreover from ‹A_P ♯* Ψ_Q› **have** $(p \cdot A_P)$ ♯* $(p \cdot \Psi_Q)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
moreover from ‹A_Q ♯* Ψ_P› **have** $(p \cdot A_Q)$ ♯* $(p \cdot \Psi_P)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
ultimately show *?case* **using** ‹A_{PQ} ♯* Ψ› ‹A_{PQ} ♯* P› ‹A_{PQ} ♯* Q› ‹A_{PQ} ♯* M› *Aeq* Ψ*eq*
by(*intro rPar2*) (*assumption* | *simp*)+
qed

lemma *parCasesBrOutputFrame*[*consumes 11*, *case-names cPar1 cPar2 cBrComm1 cBrComm2*]:
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* ♯* Ψ
and *xvec* ♯* P
and *xvec* ♯* Q
and *xvec* ♯* M
and $extractFrame(P \parallel Q) = \langle A_{PQ}, \Psi_{PQ}\rangle$
and $distinct\ A_{PQ}$
and A_{PQ} ♯* Ψ
and A_{PQ} ♯* P
and A_{PQ} ♯* Q
and A_{PQ} ♯* 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* $(P \parallel Q')$

and *rBrComm1*: $\bigwedge \Psi_Q\ P'\ A_P\ \Psi_P\ Q'\ A_Q.$
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$; *extractFrame* $P = \langle A_P,\ \Psi_P\rangle$; *distinct* A_P;
$\Psi \otimes \Psi_P \rhd Q \longmapsto ¡M(\!|\nu* xvec|\!)\langle N\rangle \prec Q'$; *extractFrame* $Q = \langle A_Q,\ \Psi_Q\rangle$; *distinct* A_Q;
distinct xvec;
$A_P \sharp* \Psi$; $A_P \sharp* \Psi_Q$; $A_P \sharp* P$; $A_P \sharp* Q$; $A_P \sharp* A_Q$;
$A_Q \sharp* \Psi$; $A_Q \sharp* \Psi_P$; $A_Q \sharp* P$; $A_Q \sharp* Q$;
$A_P \sharp* M$; $A_Q \sharp* M$; *xvec* $\sharp* M$;
xvec $\sharp* \Psi$; *xvec* $\sharp* P$; *xvec* $\sharp* Q$;
$A_{PQ} = A_P@A_Q$; $\Psi_{PQ} = \Psi_P \otimes \Psi_Q]\!] \Longrightarrow$
Prop $(P' \parallel Q')$

and *rBrComm2*: $\bigwedge \Psi_Q\ P'\ A_P\ \Psi_P\ Q'\ A_Q.$
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto ¡M(\!|\nu* xvec|\!)\langle N\rangle \prec P'$; *extractFrame* $P = \langle A_P,\ \Psi_P\rangle$; *distinct* A_P;
$\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$; *extractFrame* $Q = \langle A_Q,\ \Psi_Q\rangle$; *distinct* A_Q;
distinct xvec;
$A_P \sharp* \Psi$; $A_P \sharp* \Psi_Q$; $A_P \sharp* P$; $A_P \sharp* Q$; $A_P \sharp* A_Q$;
$A_Q \sharp* \Psi$; $A_Q \sharp* \Psi_P$; $A_Q \sharp* P$; $A_Q \sharp* Q$;
$A_P \sharp* M$; $A_Q \sharp* M$; *xvec* $\sharp* M$;
xvec $\sharp* \Psi$; *xvec* $\sharp* P$; *xvec* $\sharp* Q$;
$A_{PQ} = A_P@A_Q$; $\Psi_{PQ} = \Psi_P \otimes \Psi_Q]\!] \Longrightarrow$
Prop $(P' \parallel Q')$

shows *Prop* T
using *Trans* ‹*xvec* $\sharp* \Psi$› ‹*xvec* $\sharp* P$› ‹*xvec* $\sharp* Q$› ‹*xvec* $\sharp* M$›
proof(*induct rule*: *parBrOutputCases*[*of* - - - - - - - $(A_{PQ},\ \Psi_{PQ})$])
case(*cPar1* $P'\ A_Q\ \Psi_Q$)
from ‹$A_Q \sharp* (A_{PQ},\ \Psi_{PQ})$› **have** $A_Q \sharp* A_{PQ}$ **and** $A_Q \sharp* \Psi_{PQ}$ **by** *simp+*
obtain $A_P\ \Psi_P$ **where** *FrP*: *extractFrame* $P = \langle A_P,\ \Psi_P\rangle$ **and** *distinct* A_P
$A_P \sharp* (P,\ Q,\ \Psi,\ M,\ A_Q,\ A_{PQ},\ \Psi_Q)$
by(*rule freshFrame*)
then have $A_P \sharp* P$ **and** $A_P \sharp* Q$ **and** $A_P \sharp* \Psi$ **and** $A_P \sharp* M$ **and** $A_P \sharp* A_Q$ **and** $A_P \sharp* A_{PQ}$ **and** $A_P \sharp* \Psi_Q$
by *simp+*

have *FrQ*: *extractFrame* $Q = \langle A_Q,\ \Psi_Q\rangle$ **by** *fact*

from ‹A_Q ♯* P› ‹A_P ♯* A_Q› *FrP* **have** A_Q ♯* Ψ_P
by(*force dest*: *extractFrameFreshChain*)

from ‹*extractFrame*(P ∥ Q) = ⟨A_{PQ}, Ψ_{PQ}⟩› *FrP FrQ* ‹A_P ♯* A_Q› ‹A_P ♯* Ψ_Q› ‹A_Q ♯* Ψ_P›
have ⟨(A_P@A_Q), Ψ_P ⊗ Ψ_Q⟩ = ⟨A_{PQ}, Ψ_{PQ}⟩ **by** *simp*
moreover from ‹*distinct* A_P› ‹*distinct* A_Q› ‹A_P ♯* A_Q› **have** *distinct*(A_P@A_Q)
by(*auto simp add*: *fresh-star-def fresh-def name-list-supp*)
ultimately obtain p **where** S: *set* p ⊆ *set*(A_P@A_Q) × *set*((p ∙ A_P)@(p ∙ A_Q)) **and** *distinctPerm* p
and Ψeq: Ψ_{PQ} = (p ∙ Ψ_P) ⊗ (p ∙ Ψ_Q) **and** Aeq: A_{PQ} = (p ∙ A_P)@(p ∙ A_Q)
using ‹A_P ♯* A_{PQ}› ‹A_Q ♯* A_{PQ}› ‹*distinct* A_{PQ}›
by(*elim frameChainEq'*) (*assumption* | *simp add*: *eqvts*)+

from ‹Ψ ⊗ Ψ_Q ▷ P ⟼¡M⦇ν*$xvec$⦈⟨N⟩ ≺ P'› S ‹A_{PQ} ♯* P› ‹A_P ♯* P› ‹A_Q ♯* P› ‹A_{PQ} ♯* M› ‹A_P ♯* M› ‹A_Q ♯* M› Aeq
have (p ∙ (Ψ ⊗ Ψ_Q)) ▷ P ⟼¡M⦇ν*$xvec$⦈⟨N⟩ ≺ P'
by(*elim broutputPermFrame*) (*assumption* | *simp*)+

with S ‹A_{PQ} ♯* Ψ› ‹A_P ♯* Ψ› ‹A_Q ♯* Ψ› Aeq **have** Ψ ⊗ (p ∙ Ψ_Q) ▷ P ⟼¡M⦇ν*$xvec$⦈⟨N⟩ ≺ P'
by(*simp add*: *eqvts*)
moreover from *FrP* **have** (p ∙ *extractFrame* P) = p ∙ ⟨A_P, Ψ_P⟩ **by** *simp*
with S ‹A_{PQ} ♯* P› ‹A_P ♯* P› ‹A_Q ♯* P› Aeq **have** *extractFrame* P = ⟨(p ∙ A_P), p ∙ Ψ_P⟩
by(*simp add*: *eqvts*)
moreover from *FrQ* **have** (p ∙ *extractFrame* Q) = p ∙ ⟨A_Q, Ψ_Q⟩ **by** *simp*
with S ‹A_{PQ} ♯* Q› ‹A_P ♯* Q› ‹A_Q ♯* Q› Aeq **have** *extractFrame* Q = ⟨(p ∙ A_Q), p ∙ Ψ_Q⟩
by(*simp add*: *eqvts*)
moreover from ‹*distinct* A_P› ‹*distinct* A_Q› **have** *distinct*(p ∙ A_P) **and** *distinct*(p ∙ A_Q)
by *simp*+
moreover from ‹A_P ♯* A_Q› **have** (p ∙ A_P) ♯* (p ∙ A_Q) **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
moreover from ‹A_P ♯* Ψ_Q› **have** (p ∙ A_P) ♯* (p ∙ Ψ_Q) **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
moreover from ‹A_Q ♯* Ψ_P› **have** (p ∙ A_Q) ♯* (p ∙ Ψ_P) **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
ultimately show *?case* **using** ‹A_{PQ} ♯* Ψ› ‹A_{PQ} ♯* P› ‹A_{PQ} ♯* Q› ‹A_{PQ} ♯* M› Aeq Ψeq
by(*intro rPar1*) (*assumption* | *simp*)+
next
case(*cPar2* Q' A_P Ψ_P)
from ‹A_P ♯* (A_{PQ}, Ψ_{PQ})› **have** A_P ♯* A_{PQ} **and** A_P ♯* Ψ_{PQ} **by** *simp*+
obtain A_Q Ψ_Q **where** *FrQ*: *extractFrame* Q = ⟨A_Q, Ψ_Q⟩ **and** *distinct* A_Q A_Q ♯* (P, Q, Ψ, M, A_P, A_{PQ}, Ψ_P)
by(*rule freshFrame*)
then have A_Q ♯* P **and** A_Q ♯* Q **and** A_Q ♯* Ψ **and** A_Q ♯* M **and** A_Q ♯* A_P

and $A_Q \sharp* A_{PQ}$ **and** $A_Q \sharp* \Psi_P$
by *simp+*

have *FrP*: $extractFrame\ P = \langle A_P, \Psi_P\rangle$ **by** *fact*

from $\langle A_P \sharp* Q\rangle$ $\langle A_Q \sharp* A_P\rangle$ *FrQ* **have** $A_P \sharp* \Psi_Q$
by(*force dest*: *extractFrameFreshChain*)

from $\langle extractFrame(P \parallel Q) = \langle A_{PQ}, \Psi_{PQ}\rangle\rangle$ *FrP FrQ* $\langle A_Q \sharp* A_P\rangle$ $\langle A_P \sharp* \Psi_Q\rangle$ $\langle A_Q \sharp* \Psi_P\rangle$
have $\langle(A_P@A_Q), \Psi_P \otimes \Psi_Q\rangle = \langle A_{PQ}, \Psi_{PQ}\rangle$ **by** *simp*
moreover from $\langle distinct\ A_P\rangle$ $\langle distinct\ A_Q\rangle$ $\langle A_Q \sharp* A_P\rangle$ **have** $distinct(A_P@A_Q)$
by(*auto simp add*: *fresh-star-def fresh-def name-list-supp*)
ultimately obtain *p* **where** *S*: $set\ p \subseteq set(A_P@A_Q) \times set((p \cdot A_P)@(p \cdot A_Q))$ **and** *distinctPerm p*
and Ψeq: $\Psi_{PQ} = (p \cdot \Psi_P) \otimes (p \cdot \Psi_Q)$ **and** *Aeq*: $A_{PQ} = (p \cdot A_P)@(p \cdot A_Q)$
using $\langle A_P \sharp* A_{PQ}\rangle$ $\langle A_Q \sharp* A_{PQ}\rangle$ $\langle distinct\ A_{PQ}\rangle$
by(*elim frameChainEq'*) (*assumption* | *simp add*: *eqvts*)+

from $\langle \Psi \otimes \Psi_P \rhd Q \longmapsto ¡M(\!|\nu* xvec|\!)\langle N\rangle \prec Q'\rangle$ *S* $\langle A_{PQ} \sharp* Q\rangle$ $\langle A_P \sharp* Q\rangle$ $\langle A_Q \sharp* Q\rangle$ $\langle A_{PQ} \sharp* M\rangle$ $\langle A_P \sharp* M\rangle$ $\langle A_Q \sharp* M\rangle$ *Aeq*
have $(p \cdot (\Psi \otimes \Psi_P)) \rhd Q \longmapsto ¡M(\!|\nu* xvec|\!)\langle N\rangle \prec Q'$
by(*elim broutputPermFrame*) (*assumption* | *simp*)+
with *S* $\langle A_{PQ} \sharp* \Psi\rangle$ $\langle A_P \sharp* \Psi\rangle$ $\langle A_Q \sharp* \Psi\rangle$ *Aeq* **have** $\Psi \otimes (p \cdot \Psi_P) \rhd Q \longmapsto ¡M(\!|\nu* xvec|\!)\langle N\rangle \prec Q'$
by(*simp add*: *eqvts*)
moreover from *FrP* **have** $(p \cdot extractFrame\ P) = p \cdot \langle A_P, \Psi_P\rangle$ **by** *simp*
with *S* $\langle A_{PQ} \sharp* P\rangle$ $\langle A_P \sharp* P\rangle$ $\langle A_Q \sharp* P\rangle$ *Aeq* **have** $extractFrame\ P = \langle(p \cdot A_P), p \cdot \Psi_P\rangle$
by(*simp add*: *eqvts*)
moreover from *FrQ* **have** $(p \cdot extractFrame\ Q) = p \cdot \langle A_Q, \Psi_Q\rangle$ **by** *simp*
with *S* $\langle A_{PQ} \sharp* Q\rangle$ $\langle A_P \sharp* Q\rangle$ $\langle A_Q \sharp* Q\rangle$ *Aeq* **have** $extractFrame\ Q = \langle(p \cdot A_Q), p \cdot \Psi_Q\rangle$
by(*simp add*: *eqvts*)
moreover from $\langle distinct\ A_P\rangle$ $\langle distinct\ A_Q\rangle$ **have** $distinct(p \cdot A_P)$ **and** $distinct(p \cdot A_Q)$
by *simp+*
moreover from $\langle A_Q \sharp* A_P\rangle$ **have** $(p \cdot A_P) \sharp* (p \cdot A_Q)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
moreover from $\langle A_P \sharp* \Psi_Q\rangle$ **have** $(p \cdot A_P) \sharp* (p \cdot \Psi_Q)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
moreover from $\langle A_Q \sharp* \Psi_P\rangle$ **have** $(p \cdot A_Q) \sharp* (p \cdot \Psi_P)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
ultimately show *?case* **using** $\langle A_{PQ} \sharp* \Psi\rangle$ $\langle A_{PQ} \sharp* P\rangle$ $\langle A_{PQ} \sharp* Q\rangle$ $\langle A_{PQ} \sharp* M\rangle$ *Aeq* Ψeq
by(*intro rPar2*) (*assumption* | *simp*)+
next
case(*cBrComm1* Ψ_Q P' A_P Ψ_P Q' A_Q)
then have *FrP*: $extractFrame\ P = \langle A_P, \Psi_P\rangle$ **and** *FrQ*: $extractFrame\ Q = \langle A_Q,$

$\Psi_Q\rangle$
and $A_P \sharp* A_{PQ}$ **and** $A_Q \sharp* A_{PQ}$
by *simp+*

from ‹*extractFrame*$(P \parallel Q) = \langle A_{PQ}, \Psi_{PQ}\rangle$› *FrP FrQ* ‹$A_P \sharp* A_Q$› ‹$A_P \sharp* \Psi_Q$› ‹$A_Q \sharp* \Psi_P$›
have $\langle (A_P @ A_Q), \Psi_P \otimes \Psi_Q\rangle = \langle A_{PQ}, \Psi_{PQ}\rangle$ **by** *simp*
moreover from ‹*distinct* A_P› ‹*distinct* A_Q› ‹$A_P \sharp* A_Q$› **have** *distinct*$(A_P @ A_Q)$
by(*auto simp add*: *fresh-star-def fresh-def name-list-supp*)
ultimately obtain p **where** S: *set* $p \subseteq$ *set*$(A_P @ A_Q) \times$ *set*$((p \cdot A_P) @ (p \cdot A_Q))$ **and** *distinctPerm* p
and Ψeq: $\Psi_{PQ} = (p \cdot \Psi_P) \otimes (p \cdot \Psi_Q)$ **and** Aeq: $A_{PQ} = (p \cdot A_P) @ (p \cdot A_Q)$
using ‹$A_P \sharp* A_{PQ}$› ‹$A_Q \sharp* A_{PQ}$› ‹*distinct* A_{PQ}›
by(*elim frameChainEq'*) (*assumption* | *simp add*: *eqvts*)+

from ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$› S ‹$A_{PQ} \sharp* P$› ‹$A_P \sharp* P$› ‹$A_Q \sharp* P$› ‹$A_{PQ} \sharp* M$› ‹$A_P \sharp* M$› ‹$A_Q \sharp* M$› Aeq
have $(p \cdot (\Psi \otimes \Psi_Q)) \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$
by(*elim brinputPermFrame*) (*assumption* | *simp*)+

from ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec Q'$› S ‹$A_{PQ} \sharp* Q$› ‹$A_Q \sharp* Q$› ‹$A_P \sharp* Q$› ‹$A_{PQ} \sharp* M$› ‹$A_P \sharp* M$› ‹$A_Q \sharp* M$› Aeq
have $(p \cdot (\Psi \otimes \Psi_P)) \rhd Q \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec Q'$
by(*elim broutputPermFrame*) (*assumption* | *simp*)+

from ‹$(p \cdot (\Psi \otimes \Psi_Q)) \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$› S ‹$A_{PQ} \sharp* \Psi$› ‹$A_P \sharp* \Psi$› ‹$A_Q \sharp* \Psi$› Aeq **have** $\Psi \otimes (p \cdot \Psi_Q) \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$
by(*simp add*: *eqvts*)
moreover from ‹$(p \cdot (\Psi \otimes \Psi_P)) \rhd Q \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec Q'$› S ‹$A_{PQ} \sharp* \Psi$› ‹$A_P \sharp* \Psi$› ‹$A_Q \sharp* \Psi$› Aeq **have** $\Psi \otimes (p \cdot \Psi_P) \rhd Q \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec Q'$
by(*simp add*: *eqvts*)

moreover from *FrP* **have** $(p \cdot$ *extractFrame* $P) = p \cdot \langle A_P, \Psi_P\rangle$ **by** *simp*
with S ‹$A_{PQ} \sharp* P$› ‹$A_P \sharp* P$› ‹$A_Q \sharp* P$› Aeq **have** *extractFrame* $P = \langle (p \cdot A_P), p \cdot \Psi_P\rangle$
by(*simp add*: *eqvts*)
moreover from *FrQ* **have** $(p \cdot$ *extractFrame* $Q) = p \cdot \langle A_Q, \Psi_Q\rangle$ **by** *simp*
with S ‹$A_{PQ} \sharp* Q$› ‹$A_P \sharp* Q$› ‹$A_Q \sharp* Q$› Aeq **have** *extractFrame* $Q = \langle (p \cdot A_Q), p \cdot \Psi_Q\rangle$
by(*simp add*: *eqvts*)
moreover from ‹*distinct* A_P› ‹*distinct* A_Q› **have** *distinct*$(p \cdot A_P)$ **and** *distinct*$(p \cdot A_Q)$
by *simp+*
moreover from ‹$A_P \sharp* A_Q$› **have** $(p \cdot A_P) \sharp* (p \cdot A_Q)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
moreover from ‹$A_P \sharp* \Psi_Q$› **have** $(p \cdot A_P) \sharp* (p \cdot \Psi_Q)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
moreover from ‹$A_Q \sharp* \Psi_P$› **have** $(p \cdot A_Q) \sharp* (p \cdot \Psi_P)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])

ultimately show *?case* **using** ‹*distinct xvec*› ‹*xvec* ♯* M› ‹*xvec* ♯* Ψ› ‹*xvec* ♯* P› ‹*xvec* ♯* Q› ‹A_{PQ} ♯* Ψ› ‹A_{PQ} ♯* P› ‹A_{PQ} ♯* Q› ‹A_{PQ} ♯* M› *Aeq* Ψeq
by(*intro rBrComm1*) (*assumption* | *simp*)+
next
case(*cBrComm2* Ψ_Q P' A_P Ψ_P Q' A_Q)
then have *FrP*: *extractFrame* $P = \langle A_P, \Psi_P\rangle$ **and** *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$
and A_P ♯* A_{PQ} **and** A_Q ♯* A_{PQ}
by *simp*+

from ‹*extractFrame*$(P \parallel Q) = \langle A_{PQ}, \Psi_{PQ}\rangle$› *FrP FrQ* ‹$A_P$ ♯* A_Q› ‹A_P ♯* Ψ_Q› ‹A_Q ♯* Ψ_P›
have $\langle(A_P@A_Q), \Psi_P \otimes \Psi_Q\rangle = \langle A_{PQ}, \Psi_{PQ}\rangle$ **by** *simp*
moreover from ‹*distinct* A_P› ‹*distinct* A_Q› ‹A_P ♯* A_Q› **have** *distinct*$(A_P@A_Q)$
by(*auto simp add*: *fresh-star-def fresh-def name-list-supp*)
ultimately obtain p **where** S: *set* $p \subseteq$ *set*$(A_P@A_Q) \times$ *set*$((p \cdot A_P)@(p \cdot A_Q))$ **and** *distinctPerm* p
and Ψeq: $\Psi_{PQ} = (p \cdot \Psi_P) \otimes (p \cdot \Psi_Q)$ **and** *Aeq*: $A_{PQ} = (p \cdot A_P)@(p \cdot A_Q)$
using ‹A_P ♯* A_{PQ}› ‹A_Q ♯* A_{PQ}› ‹*distinct* A_{PQ}›
by(*elim frameChainEq'*) (*assumption* | *simp add*: *eqvts*)+

from ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$› S ‹A_{PQ} ♯* P› ‹A_P ♯* P› ‹A_Q ♯* P› ‹A_{PQ} ♯* M› ‹A_P ♯* M› ‹A_Q ♯* M› *Aeq*
have $(p \cdot (\Psi \otimes \Psi_Q)) \rhd P \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$
by(*elim broutputPermFrame*) (*assumption* | *simp*)+

from ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$› S ‹A_{PQ} ♯* Q› ‹A_Q ♯* Q› ‹A_P ♯* Q› ‹A_{PQ} ♯* M› ‹A_P ♯* M› ‹A_Q ♯* M› *Aeq*
have $(p \cdot (\Psi \otimes \Psi_P)) \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$
by(*elim brinputPermFrame*) (*assumption* | *simp*)+

from ‹$(p \cdot (\Psi \otimes \Psi_Q)) \rhd P \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$› S ‹A_{PQ} ♯* Ψ› ‹A_P ♯* Ψ› ‹A_Q ♯* Ψ› *Aeq* **have** $\Psi \otimes (p \cdot \Psi_Q) \rhd P \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$
by(*simp add*: *eqvts*)
moreover from ‹$(p \cdot (\Psi \otimes \Psi_P)) \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$› S ‹A_{PQ} ♯* Ψ› ‹A_P ♯* Ψ› ‹A_Q ♯* Ψ› *Aeq* **have** $\Psi \otimes (p \cdot \Psi_P) \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$
by(*simp add*: *eqvts*)

moreover from *FrP* **have** $(p \cdot$ *extractFrame* $P) = p \cdot \langle A_P, \Psi_P\rangle$ **by** *simp*
with S ‹A_{PQ} ♯* P› ‹A_P ♯* P› ‹A_Q ♯* P› *Aeq* **have** *extractFrame* $P = \langle(p \cdot A_P), p \cdot \Psi_P\rangle$
by(*simp add*: *eqvts*)
moreover from *FrQ* **have** $(p \cdot$ *extractFrame* $Q) = p \cdot \langle A_Q, \Psi_Q\rangle$ **by** *simp*
with S ‹A_{PQ} ♯* Q› ‹A_P ♯* Q› ‹A_Q ♯* Q› *Aeq* **have** *extractFrame* $Q = \langle(p \cdot A_Q), p \cdot \Psi_Q\rangle$
by(*simp add*: *eqvts*)
moreover from ‹*distinct* A_P› ‹*distinct* A_Q› **have** *distinct*$(p \cdot A_P)$ **and** *distinct*$(p \cdot A_Q)$
by *simp*+

moreover from ‹$A_P \sharp* A_Q$› **have** $(p \cdot A_P) \sharp* (p \cdot A_Q)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
moreover from ‹$A_P \sharp* \Psi_Q$› **have** $(p \cdot A_P) \sharp* (p \cdot \Psi_Q)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
moreover from ‹$A_Q \sharp* \Psi_P$› **have** $(p \cdot A_Q) \sharp* (p \cdot \Psi_P)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
ultimately show *?case* **using** ‹*distinct xvec*› ‹*xvec* $\sharp*$ M› ‹*xvec* $\sharp*$ Ψ› ‹*xvec* $\sharp*$ P› ‹*xvec* $\sharp*$ Q› ‹$A_{PQ} \sharp* \Psi$› ‹$A_{PQ} \sharp* P$› ‹$A_{PQ} \sharp* Q$› ‹$A_{PQ} \sharp* M$› *Aeq* *Ψeq*
by(*intro rBrComm2*) (*assumption* | *simp*)+
qed

inductive *bangPred* :: ('*a*, '*b*, '*c*) *psi* ⇒ ('*a*, '*b*, '*c*) *psi* ⇒ *bool*
where
aux1: *bangPred P* (!*P*)
| *aux2*: *bangPred P* (*P* ∥ !*P*)

lemma *bangInduct*[*consumes 1*, *case-names cPar1 cPar2 cComm1 cComm2 cBrMerge cBrComm1 cBrComm2 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 Ψ ▷ !*P* ⟼ *Rs*
and *rPar1*: ⋀α *P'* *C*. ⟦Ψ ▷ *P* ⟼α ≺ *P'*; *bn* α $\sharp*$ Ψ; *bn* α $\sharp*$ *P*; *bn* α $\sharp*$ *subject* α; *bn* α $\sharp*$ *C*; *distinct*(*bn* α)⟧ ⟹ *Prop C* Ψ (*P* ∥ !*P*) (α ≺ (*P'* ∥ !*P*))
and *rPar2*: ⋀α *P'* *C*. ⟦Ψ ▷ !*P* ⟼α ≺ *P'*; *bn* α $\sharp*$ Ψ; *bn* α $\sharp*$ *P*; *bn* α $\sharp*$ *subject* α; *bn* α $\sharp*$ *C*; *distinct*(*bn* α); ⋀*C*. *Prop C* Ψ (!*P*) (α ≺ *P'*)⟧ ⟹ *Prop C* Ψ (*P* ∥ !*P*) (α ≺ (*P* ∥ *P'*))
and *rComm1*: ⋀*M N P' K xvec P''* *C*. ⟦Ψ ▷ *P* ⟼*M*(⦇*N*⦈) ≺ *P'*; Ψ ▷ !*P* ⟼*K*(⦇*ν*∗*xvec*⦈)⟨*N*⟩ ≺ *P''*; ⋀*C*. *Prop C* Ψ (!*P*) (*K*(⦇*ν*∗*xvec*⦈)⟨*N*⟩ ≺ *P''*); Ψ ⊢ *M* ↔ *K*; *xvec* $\sharp*$ Ψ; *xvec* $\sharp*$ *P*; *xvec* $\sharp*$ *M*; *xvec* $\sharp*$ *K*; *xvec* $\sharp*$ *C*; *distinct xvec*⟧ ⟹ *Prop C* Ψ (*P* ∥ !*P*) (τ ≺ (⦇*ν*∗*xvec*⦈)(*P'* ∥ *P''*))
and *rComm2*: ⋀*M xvec N P' K P''* *C*. ⟦Ψ ▷ *P* ⟼*M*(⦇*ν*∗*xvec*⦈)⟨*N*⟩ ≺ *P'*; Ψ ▷ !*P* ⟼*K*(⦇*N*⦈) ≺ *P''*; ⋀*C*. *Prop C* Ψ (!*P*) (*K*(⦇*N*⦈) ≺ *P''*); Ψ ⊢ *M* ↔ *K*; *xvec* $\sharp*$ Ψ; *xvec* $\sharp*$ *P*; *xvec* $\sharp*$ *M*; *xvec* $\sharp*$ *K*; *xvec* $\sharp*$ *C*; *distinct xvec*⟧ ⟹ *Prop C* Ψ (*P* ∥ !*P*) (τ ≺ (⦇*ν*∗*xvec*⦈)(*P'* ∥ *P''*))
and *rBrMerge*: ⋀*M N P' P''* *C*. ⟦Ψ ▷ *P* ⟼¿*M*(⦇*N*⦈) ≺ *P'*; Ψ ▷ !*P* ⟼¿*M*(⦇*N*⦈) ≺ *P''*; ⋀*C*. *Prop C* Ψ (!*P*) (¿*M*(⦇*N*⦈) ≺ *P''*)⟧ ⟹ *Prop C* Ψ (*P* ∥ !*P*) (¿*M*(⦇*N*⦈) ≺ (*P'* ∥ *P''*))
and *rBrComm1*: ⋀*M N P' xvec P''* *C*. ⟦Ψ ▷ *P* ⟼¿*M*(⦇*N*⦈) ≺ *P'*; Ψ ▷ !*P* ⟼¡*M*(⦇*ν*∗*xvec*⦈)⟨*N*⟩ ≺ *P''*; ⋀*C*. *Prop C* Ψ (!*P*) (¡*M*(⦇*ν*∗*xvec*⦈)⟨*N*⟩ ≺ *P''*); *xvec* $\sharp*$ Ψ; *xvec* $\sharp*$ *P*; *xvec* $\sharp*$ *M*; *xvec* $\sharp*$ *C*; *distinct xvec*⟧ ⟹ *Prop C* Ψ (*P* ∥ !*P*) (¡*M*(⦇*ν*∗*xvec*⦈)⟨*N*⟩ ≺ (*P'* ∥ *P''*))
and *rBrComm2*: ⋀*M N P' xvec P''* *C*. ⟦Ψ ▷ *P* ⟼¡*M*(⦇*ν*∗*xvec*⦈)⟨*N*⟩ ≺ *P'*; Ψ ▷ !*P* ⟼¿*M*(⦇*N*⦈) ≺ *P''*; ⋀*C*. *Prop C* Ψ (!*P*) (¿*M*(⦇*N*⦈) ≺ *P''*);

xvec $\sharp*$ Ψ; *xvec* $\sharp*$ *P*; *xvec* $\sharp*$ *M*; *xvec* $\sharp*$ *C*; *distinct xvec*$]\!] \Longrightarrow$ *Prop C* Ψ (*P* $\parallel$!*P*) (¡*M*(|ν*xvec*|)$\langle N\rangle \prec$ (*P'* $\parallel$ *P''*))
and *rBang*: $\bigwedge$*Rs C*. $[\![\Psi \rhd P \parallel !P \longmapsto Rs$; $\bigwedge$*C*. *Prop C* Ψ (*P* $\parallel$!*P*) *Rs*; *guarded P*$]\!] \Longrightarrow$ *Prop C* Ψ (!*P*) *Rs*
shows *Prop C* Ψ (!*P*) *Rs*
proof −
from ‹$\Psi \rhd !P \longmapsto Rs$› **have** *guarded P*
by(*nominal-induct* Ψ *P*==!*P Rs rule*: *semantics.strong-induct*) (*auto simp add*: *psi.inject*)
{
fix *Q* :: ('*a*, '*b*, '*c*) *psi*
and Ψ' :: '*b*

assume $\Psi' \rhd Q \longmapsto Rs$
and *guarded Q*
and *bangPred P Q*
and $\Psi \simeq \Psi'$

then have *Prop C* Ψ *Q Rs* **using** *rPar1 rPar1 rPar2 rPar2 rComm1 rComm2 rBrMerge rBrComm1 rBrComm2 rBang*
proof(*nominal-induct avoiding*: Ψ *C rule*: *semantics.strong-induct*)
case(*cInput* Ψ' *M K xvec N Tvec Q* Ψ *C*)
then show *?case* **by** − (*ind-cases bangPred P* (*M*(|λ*xvec N*|).*Q*))
next
case(*cBrInput* Ψ' *K M xvec N Tvec Q* Ψ *C*)
then show *?case* **by** − (*ind-cases bangPred P* (*M*(|λ*xvec N*|).*Q*))
next
case(*Output* Ψ' *M K N Q* Ψ *C*)
then show *?case* **by** − (*ind-cases bangPred P* (*M*$\langle N\rangle$.*Q*))
next
case(*BrOutput* Ψ' *M K N Q* Ψ *C*)
then show *?case* **by** − (*ind-cases bangPred P* (*M*$\langle N\rangle$.*Q*))
next
case(*Case* Ψ' *Q Rs* φ *Cs* Ψ *C*)
then show *?case* **by** − (*ind-cases bangPred P* (*Cases Cs*))
next
case(*cPar1* Ψ' Ψ_R *Q* α *P'* *R* A_R Ψ *C*)
have *rPar1*: $\bigwedge\alpha$ *P' C*. $[\![\Psi \rhd P \longmapsto\alpha \prec P'$; *bn* α $\sharp*$ Ψ; *bn* α $\sharp*$ *P*; *bn* α $\sharp*$ *subject* α; *bn* α $\sharp*$ *C*; *distinct*(*bn* α)$]\!] \Longrightarrow$ *Prop C* Ψ (*P* $\parallel$!*P*) ($\alpha \prec$ (*P'* $\parallel$!*P*))
by *fact*
from ‹*bangPred P* (*Q* $\parallel$ *R*)› **have** *Q* = *P* **and** *R* = !*P*
by − (*ind-cases bangPred P* (*Q* $\parallel$ *R*), *auto simp add*: *psi.inject*)+
from ‹*R* = !*P*› ‹*extractFrame R* = $\langle A_R, \Psi_R\rangle$› **have** A_R = [] **and** Ψ_R = **1** **by** *auto*
from ‹$\Psi' \otimes \Psi_R \rhd Q \longmapsto\alpha \prec P'$› ‹*Q* = *P*› ‹$\Psi \simeq \Psi'$› ‹$\Psi_R$ = **1**› **have** $\Psi \rhd P \longmapsto\alpha \prec P'$
by(*metis statEqTransition Identity AssertionStatEqSym*)
then have *Prop C* Ψ (*P* $\parallel$!*P*) ($\alpha \prec$ (*P'* $\parallel$!*P*)) **using** ‹*bn* α $\sharp*$ Ψ› ‹*bn* α $\sharp*$ *Q*› ‹*bn* α $\sharp*$ *subject* α› ‹*bn* α $\sharp*$ *C*› ‹*Q* = *P*› ‹*distinct*(*bn* α)›

by(*intro rPar1*) *auto*
with ‹$R = !P$› ‹$Q = P$› **show** *?case* **by** *simp*
next
case(*cPar2* Ψ' Ψ_P R α P' Q A_P Ψ C)
have *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 Ψ $(!P)$ $(\alpha \prec P')]\!] \Longrightarrow$ *Prop* C Ψ $(P \parallel !P)$ $(\alpha \prec (P \parallel P'))$
by *fact*
from ‹*bangPred* P $(Q \parallel R)$› **have** $Q = P$ **and** $R = !P$
by $-$ (*ind-cases bangPred* P $(Q \parallel R)$, *auto simp add*: *psi.inject*)+
from ‹$Q = P$› ‹*extractFrame* $Q = \langle A_P, \Psi_P\rangle$› ‹*guarded* P› **have** $\Psi_P \simeq \mathbf{1}$ **and** *supp* $\Psi_P = (\{\}::name\ set)$
by(*blast dest*: *guardedStatEq*)+
from ‹$\Psi' \otimes \Psi_P \rhd R \longmapsto\alpha \prec P'$› ‹$R = !P$› ‹$\Psi \simeq \Psi'$› ‹$\Psi_P \simeq \mathbf{1}$› **have** $\Psi \rhd !P \longmapsto\alpha \prec P'$
by(*metis statEqTransition Identity Composition Commutativity AssertionStatEqSym*)
moreover
{
fix C
have *bangPred* P $(!P)$ **by**(*rule aux1*)
moreover from ‹$\Psi \simeq \Psi'$› ‹$\Psi_P \simeq \mathbf{1}$› **have** $\Psi \simeq \Psi' \otimes \Psi_P$ **by**(*metis Composition Identity Commutativity AssertionStatEqSym AssertionStatEqTrans*)
ultimately have $\bigwedge C.$ *Prop* C Ψ $(!P)$ $(\alpha \prec P')$ **using** *cPar2* ‹$R = !P$› ‹*guarded* P› **by** *simp*
}
ultimately have *Prop* C Ψ $(P \parallel !P)$ $(\alpha \prec (P \parallel P'))$ **using** ‹*bn* $\alpha \sharp* \Psi$› ‹*bn* $\alpha \sharp* Q$› ‹*bn* $\alpha \sharp*$ *subject* α› ‹*bn* $\alpha \sharp* C$› ‹$Q = P$› ‹*distinct*(*bn* α)›
by(*elim rPar2*) *auto*
with ‹$R = !P$› ‹$Q = P$› **show** *?case* **by** *simp*
next
case(*cComm1* Ψ' Ψ_R Q M N P' A_P Ψ_P R K *xvec* P'' A_R Ψ C)
have *rComm1*: $\bigwedge M\ N\ P'\ K$ *xvec* $P''\ C.\ [\![\Psi \rhd P \longmapsto M\langle\!|N|\!\rangle \prec P'$; $\Psi \rhd !P \longmapsto K(\!|\nu*xvec|\!)\langle N\rangle \prec P''$; $\bigwedge C.$ *Prop* C Ψ $(!P)$ $(K(\!|\nu*xvec|\!)\langle N\rangle \prec P'')$; $\Psi \vdash M \leftrightarrow K$; *xvec* $\sharp* \Psi$; *xvec* $\sharp* P$; *xvec* $\sharp* M$; *xvec* $\sharp* K$; *xvec* $\sharp* C$; *distinct xvec*$]\!] \Longrightarrow$ *Prop* C Ψ $(P \parallel !P)$ $(\tau \prec (\!|\nu*xvec|\!)(P' \parallel P''))$
by *fact*
from ‹*bangPred* P $(Q \parallel R)$› **have** $Q = P$ **and** $R = !P$
by $-$ (*ind-cases bangPred* P $(Q \parallel R)$, *auto simp add*: *psi.inject*)+
from ‹$R = !P$› ‹*extractFrame* $R = \langle A_R, \Psi_R\rangle$› **have** $A_R = []$ **and** $\Psi_R = \mathbf{1}$ **by** *auto*
from ‹$\Psi' \otimes \Psi_R \rhd Q \longmapsto M\langle\!|N|\!\rangle \prec P'$› ‹$Q = P$› ‹$\Psi \simeq \Psi'$› ‹$\Psi_R = \mathbf{1}$› **have** $\Psi \rhd P \longmapsto M\langle\!|N|\!\rangle \prec P'$
by(*metis statEqTransition Identity AssertionStatEqSym*)
moreover from ‹$Q = P$› ‹*extractFrame* $Q = \langle A_P, \Psi_P\rangle$› ‹*guarded* P› **have** $\Psi_P \simeq \mathbf{1}$ **and** *supp* $\Psi_P = (\{\}::name\ set)$
by(*blast dest*: *guardedStatEq*)+

moreover from ‹$\Psi' \otimes \Psi_P \rhd R \longmapsto K(\!|\nu{*}xvec|\!)\langle N\rangle \prec P''$› ‹$R = !P$› ‹$\Psi_P \simeq \mathbf{1}$› ‹$\Psi \simeq \Psi'$› **have** $\Psi \rhd !P \longmapsto K(\!|\nu{*}xvec|\!)\langle N\rangle \prec P''$
by(*metis statEqTransition Identity Composition Commutativity AssertionStatEqSym*)
moreover
{
fix C
have *bangPred* P $(!P)$ **by**(*rule aux1*)
moreover from ‹$\Psi \simeq \Psi'$› ‹$\Psi_P \simeq \mathbf{1}$› **have** $\Psi \simeq \Psi' \otimes \Psi_P$ **by**(*metis Composition Identity Commutativity AssertionStatEqSym AssertionStatEqTrans*)
ultimately have $\bigwedge C.$ *Prop* C Ψ $(!P)$ $(K(\!|\nu{*}xvec|\!)\langle N\rangle \prec P'')$ **using** *cComm1* ‹$R = !P$› ‹*guarded* P› **by** *simp*
}
moreover from ‹$\Psi' \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K$› ‹$\Psi_P \simeq \mathbf{1}$› ‹$\Psi \simeq \Psi'$› ‹$\Psi_R = \mathbf{1}$› **have** $\Psi \vdash M \leftrightarrow K$
by(*metis statEqEnt Identity Composition Commutativity AssertionStatEqSym*)
ultimately have *Prop* C Ψ $(P \parallel !P)$ $(\tau \prec (\!|\nu{*}xvec|\!)(P' \parallel P''))$ **using** ‹*xvec* $\sharp*$ Ψ› ‹*xvec* $\sharp*$ Q› ‹*xvec* $\sharp*$ M› ‹*xvec* $\sharp*$ K› ‹*xvec* $\sharp*$ C› ‹$Q = P$› ‹*distinct xvec*›
by(*elim rComm1*[**where** $K{=}K$ **and** $M{=}M$ **and** $N{=}N$]) *auto*
with ‹$R = !P$› ‹$Q = P$› **show** *?case* **by** *simp*
next
case(*cComm2* Ψ' Ψ_R Q M *xvec* N P' A_P Ψ_P R K P'' A_R Ψ C)
have *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 Ψ $(!P)$ $(K(\!|N|\!) \prec P'')$; $\Psi \vdash M \leftrightarrow K$; *xvec* $\sharp*$ Ψ; *xvec* $\sharp*$ P; *xvec* $\sharp*$ M; *xvec* $\sharp*$ K; *xvec* $\sharp*$ C; *distinct xvec*$]\!] \Longrightarrow$ *Prop* C Ψ $(P \parallel !P)$ $(\tau \prec (\!|\nu{*}xvec|\!)(P' \parallel P''))$
by *fact*
from ‹*bangPred* P $(Q \parallel R)$› **have** $Q = P$ **and** $R = !P$
by $-$ (*ind-cases bangPred* P $(Q \parallel R)$, *auto simp add*: *psi.inject*)+
from ‹$R = !P$› ‹*extractFrame* $R = \langle A_R, \Psi_R\rangle$› **have** $A_R = []$ **and** $\Psi_R = \mathbf{1}$ **by** *auto*
from ‹$\Psi' \otimes \Psi_R \rhd Q \longmapsto M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P'$› ‹$Q = P$› ‹$\Psi \simeq \Psi'$› ‹$\Psi_R = \mathbf{1}$› **have** $\Psi \rhd P \longmapsto M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P'$
by(*metis statEqTransition Identity AssertionStatEqSym*)
moreover from ‹$Q = P$› ‹*extractFrame* $Q = \langle A_P, \Psi_P\rangle$› ‹*guarded* P› **have** $\Psi_P \simeq \mathbf{1}$ **and** *supp* $\Psi_P = (\{\}::\!name\ set)$
by(*blast dest*: *guardedStatEq*)+
moreover from ‹$\Psi' \otimes \Psi_P \rhd R \longmapsto K(\!|N|\!) \prec P''$› ‹$R = !P$› ‹$\Psi_P \simeq \mathbf{1}$› ‹$\Psi \simeq \Psi'$› **have** $\Psi \rhd !P \longmapsto K(\!|N|\!) \prec P''$
by(*metis statEqTransition Identity Composition Commutativity AssertionStatEqSym*)
moreover
{
fix C
have *bangPred* P $(!P)$ **by**(*rule aux1*)
moreover from ‹$\Psi \simeq \Psi'$› ‹$\Psi_P \simeq \mathbf{1}$› **have** $\Psi \simeq \Psi' \otimes \Psi_P$ **by**(*metis Composition Identity Commutativity AssertionStatEqSym AssertionStatEqTrans*)
ultimately have $\bigwedge C.$ *Prop* C Ψ $(!P)$ $(K(\!|N|\!) \prec P'')$ **using** *cComm2* ‹$R =$

$!P\rangle$ ‹*guarded* P› **by** *simp*
}
moreover from ‹$\Psi' \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K$› ‹$\Psi_P \simeq \mathbf{1}$› ‹$\Psi \simeq \Psi'$› ‹$\Psi_R = \mathbf{1}$› **have** $\Psi \vdash M \leftrightarrow K$
by(*metis statEqEnt Identity Composition Commutativity AssertionStatEqSym*)
ultimately have *Prop C* Ψ $(P \parallel !P)$ $(\tau \prec (\!|\nu{*}xvec|\!)(P' \parallel P''))$ **using** ‹*xvec* $\sharp*$ Ψ› ‹*xvec* $\sharp*$ Q› ‹*xvec* $\sharp*$ M› ‹*xvec* $\sharp*$ K› ‹*xvec* $\sharp*$ C› ‹$Q = P$› ‹*distinct xvec*›
by(*elim rComm2*[**where** $K{=}K$ **and** $M{=}M$ **and** $N{=}N$]) *auto*
with ‹$R = !P$› ‹$Q = P$› **show** *?case* **by** *simp*
next
case(*cBrMerge* Ψ' Ψ_R Q M N P' A_P Ψ_P R P'' A_R Ψ C)
have *rBrMerge*: $\bigwedge M\ N\ P'\ P''\ C.\ [\![\Psi \rhd P \longmapsto ¿M(\!|N|\!) \prec P';\ \Psi \rhd !P \longmapsto ¿M(\!|N|\!) \prec P'';\ \bigwedge C.\ \mathit{Prop}\ C\ \Psi\ (!P)\ (¿M(\!|N|\!) \prec P'')]\!] \Longrightarrow$
$\mathit{Prop}\ C\ \Psi\ (P \parallel !P)\ (¿M(\!|N|\!) \prec (P' \parallel P''))$
by *fact*
from ‹*bangPred* P $(Q \parallel R)$› **have** $Q = P$ **and** $R = !P$
by − (*ind-cases bangPred* P $(Q \parallel R)$, *auto simp add*: *psi.inject*)+
from ‹$R = !P$› ‹*extractFrame* $R = \langle A_R, \Psi_R\rangle$› **have** $A_R = []$ **and** $\Psi_R = \mathbf{1}$ **by** *auto*
from ‹$\Psi' \otimes \Psi_R \rhd Q \longmapsto ¿M(\!|N|\!) \prec P'$› ‹$Q = P$› ‹$\Psi \simeq \Psi'$› ‹$\Psi_R = \mathbf{1}$› **have** $\Psi \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$
by(*metis statEqTransition Identity AssertionStatEqSym*)
moreover from ‹$Q = P$› ‹*extractFrame* $Q = \langle A_P, \Psi_P\rangle$› ‹*guarded* P› **have** $\Psi_P \simeq \mathbf{1}$ **and** *supp* $\Psi_P = (\{\}::\mathit{name\ set})$
by(*blast dest*: *guardedStatEq*)+
from ‹$\Psi' \otimes \Psi_P \rhd R \longmapsto ¿M(\!|N|\!) \prec P''$› ‹$R = !P$› ‹$\Psi \simeq \Psi'$› ‹$\Psi_P \simeq \mathbf{1}$› **have** $\Psi \rhd !P \longmapsto ¿M(\!|N|\!) \prec P''$
by (*metis AssertionStatEqSym Identity compositionSym statEqTransition*)
moreover
{
fix C
have *bangPred* P $(!P)$ **by**(*rule aux1*)
moreover from ‹$\Psi \simeq \Psi'$› ‹$\Psi_P \simeq \mathbf{1}$› **have** $\Psi \simeq \Psi' \otimes \Psi_P$ **by**(*metis Composition Identity Commutativity AssertionStatEqSym AssertionStatEqTrans*)
ultimately have $\bigwedge C.\ \mathit{Prop}\ C\ \Psi\ (!P)\ (¿M(\!|N|\!) \prec P'')$ **using** *cBrMerge* ‹$R = !P$› ‹*guarded* P› **by** *simp*
}
ultimately have *Prop C* Ψ $(P \parallel !P)$ $(¿M(\!|N|\!) \prec (P' \parallel P''))$
by(*elim rBrMerge*) *auto*
with ‹$R = !P$› ‹$Q = P$› **show** *?case* **by** *simp*
next
case(*cBrComm1* Ψ' Ψ_R Q M N P' A_P Ψ_P R *xvec* P'' A_R Ψ C)
have *rBrComm1*: $\bigwedge M\ N\ P'\ xvec\ P''\ C.\ [\![\Psi \rhd P \longmapsto ¿M(\!|N|\!) \prec P';\ \Psi \rhd !P \longmapsto ¡M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P'';\ \bigwedge C.\ \mathit{Prop}\ C\ \Psi\ (!P)\ (¡M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P'');$
$xvec \sharp* \Psi;\ xvec \sharp* P;\ xvec \sharp* M;\ xvec \sharp* C;\ \mathit{distinct\ xvec}]\!] \Longrightarrow \mathit{Prop}\ C\ \Psi\ (P \parallel !P)\ (¡M(\!|\nu{*}xvec|\!)\langle N\rangle \prec (P' \parallel P''))$
by *fact*
from ‹*bangPred* P $(Q \parallel R)$› **have** $Q = P$ **and** $R = !P$

by − (*ind-cases bangPred P* (*Q* ∥ *R*), *auto simp add*: *psi.inject*)+
from ‹*R* = !*P*› ‹*extractFrame R* = ⟨A_R, Ψ_R⟩› **have** $A_R = []$ **and** $\Psi_R = \mathbf{1}$ **by** *auto*
from ‹$\Psi' \otimes \Psi_R \rhd Q \longmapsto ¡M(\!|N|\!) \prec P'$› ‹*Q* = *P*› ‹$\Psi \simeq \Psi'$› ‹$\Psi_R = \mathbf{1}$› **have** $\Psi \rhd P \longmapsto ¡M(\!|N|\!) \prec P'$
by(*metis statEqTransition Identity AssertionStatEqSym*)
moreover from ‹*Q* = *P*› ‹*extractFrame Q* = ⟨A_P, Ψ_P⟩› ‹*guarded P*› **have** $\Psi_P \simeq \mathbf{1}$ **and** *supp* Ψ_P = ({}::*name set*)
by(*blast dest*: *guardedStatEq*)+
moreover from ‹$\Psi' \otimes \Psi_P \rhd R \longmapsto ¡M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P''$› ‹*R* = !*P*› ‹$\Psi_P \simeq \mathbf{1}$› ‹$\Psi \simeq \Psi'$› **have** $\Psi \rhd !P \longmapsto ¡M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P''$
by(*metis statEqTransition Identity Composition Commutativity AssertionStatEqSym*)
moreover
{
fix *C*
have *bangPred P* (!*P*) **by**(*rule aux1*)
moreover from ‹$\Psi \simeq \Psi'$› ‹$\Psi_P \simeq \mathbf{1}$› **have** $\Psi \simeq \Psi' \otimes \Psi_P$ **by**(*metis Composition Identity Commutativity AssertionStatEqSym AssertionStatEqTrans*)
ultimately have $\bigwedge C$. *Prop C* Ψ (!*P*) (¡*M*(|$\nu{*}xvec$|)⟨*N*⟩ ≺ *P''*) **using** *cBrComm1* ‹*R* = !*P*› ‹*guarded P*› **by** *simp*
}
ultimately have *Prop C* Ψ (*P* ∥ !*P*) (¡*M*(|$\nu{*}xvec$|)⟨*N*⟩ ≺ (*P'* ∥ *P''*)) **using** ‹*xvec* ♯* Ψ› ‹*xvec* ♯* *Q*› ‹*xvec* ♯* *M*› ‹*xvec* ♯* *C*› ‹*Q* = *P*› ‹*distinct xvec*›
by(*elim rBrComm1*) *auto*
with ‹*R* = !*P*› ‹*Q* = *P*› **show** *?case* **by** *simp*
next
case(*cBrComm2* Ψ' Ψ_R *Q M xvec N P'* A_P Ψ_P *R P''* A_R Ψ *C*)
have *rBrComm2*: $\bigwedge M\ N\ P'\ xvec\ P''\ C$. ⟦$\Psi \rhd P \longmapsto ¡M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P'$; $\Psi \rhd !P \longmapsto ¡M(\!|N|\!) \prec P''$; $\bigwedge C$. *Prop C* Ψ (!*P*) (¡*M*(|*N*|) ≺ *P''*); *xvec* ♯* Ψ; *xvec* ♯* *P*; *xvec* ♯* *M*; *xvec* ♯* *C*; *distinct xvec*⟧ ⟹ *Prop C* Ψ (*P* ∥ !*P*) (¡*M*(|$\nu{*}xvec$|)⟨*N*⟩ ≺ (*P'* ∥ *P''*))
by *fact*
from ‹*bangPred P* (*Q* ∥ *R*)› **have** *Q* = *P* **and** *R* = !*P*
by − (*ind-cases bangPred P* (*Q* ∥ *R*), *auto simp add*: *psi.inject*)+
from ‹*R* = !*P*› ‹*extractFrame R* = ⟨A_R, Ψ_R⟩› **have** $A_R = []$ **and** $\Psi_R = \mathbf{1}$ **by** *auto*
from ‹$\Psi' \otimes \Psi_R \rhd Q \longmapsto ¡M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P'$› ‹*Q* = *P*› ‹$\Psi \simeq \Psi'$› ‹$\Psi_R = \mathbf{1}$› **have** $\Psi \rhd P \longmapsto ¡M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P'$
by(*metis statEqTransition Identity AssertionStatEqSym*)
moreover from ‹*Q* = *P*› ‹*extractFrame Q* = ⟨A_P, Ψ_P⟩› ‹*guarded P*› **have** $\Psi_P \simeq \mathbf{1}$ **and** *supp* Ψ_P = ({}::*name set*)
by(*blast dest*: *guardedStatEq*)+
moreover from ‹$\Psi' \otimes \Psi_P \rhd R \longmapsto ¡M(\!|N|\!) \prec P''$› ‹*R* = !*P*› ‹$\Psi_P \simeq \mathbf{1}$› ‹$\Psi \simeq \Psi'$› **have** $\Psi \rhd !P \longmapsto ¡M(\!|N|\!) \prec P''$
by(*metis statEqTransition Identity Composition Commutativity AssertionStatEqSym*)
moreover
{

```
      fix C
      have bangPred P (!P) by(rule aux1)
        moreover from ‹Ψ ≃ Ψ'› ‹Ψ_P ≃ 1› have Ψ ≃ Ψ' ⊗ Ψ_P by(metis
Composition Identity Commutativity AssertionStatEqSym AssertionStatEqTrans)
       ultimately have ⋀C. Prop C Ψ (!P) (¡M(⦇N⦈) ≺ P'') using cBrComm2
‹R = !P› ‹guarded P› by simp
     }
     ultimately have Prop C Ψ (P ∥ !P) (¡M(⦇ν*xvec⦈)⟨N⟩ ≺ (P' ∥ P'')) using
‹xvec ♯* Ψ› ‹xvec ♯* Q› ‹xvec ♯* M› ‹xvec ♯* C› ‹Q = P› ‹distinct xvec›
      by(elim rBrComm2) auto
     with ‹R = !P› ‹Q = P› show ?case by simp
   next
     case(cBrClose Ψ' Q M xvec N P' x Ψ C)
     then show ?case by − (ind-cases bangPred P (⦇νx⦈Q))
   next
     case(cOpen Ψ Q M xvec yvec N P' x C)
     then show ?case by − (ind-cases bangPred P (⦇νx⦈Q))
   next
     case(cBrOpen Ψ Q M xvec yvec N P' x C)
     then show ?case by − (ind-cases bangPred P (⦇νx⦈Q))
   next
     case(cScope Ψ Q α P' x C)
     then show ?case by − (ind-cases bangPred P (⦇νx⦈Q))
   next
     case(Bang Ψ' Q Rs Ψ C)
      have rBang: ⋀Rs C. ⟦Ψ ▷ P ∥ !P ⟼ Rs; ⋀C. Prop C Ψ (P ∥ !P) Rs;
guarded P⟧ ⟹ Prop C Ψ (!P) Rs
       by fact
     from ‹bangPred P (!Q)› have P = Q
       by − (ind-cases bangPred P (!Q), auto simp add: psi.inject)
      with ‹Ψ' ▷ Q ∥ !Q ⟼ Rs› ‹Ψ ≃ Ψ'› have Ψ ▷ P ∥ !P ⟼ Rs by(metis
statEqTransition AssertionStatEqSym)
     moreover
     {
       fix C
       have bangPred P (P ∥ !P) by(rule aux2)
       with Bang ‹P = Q› have ⋀C. Prop C Ψ (P ∥ !P) Rs by simp
     }
     moreover from ‹guarded Q› ‹P = Q› have guarded P by simp
     ultimately have Prop C Ψ (!P) Rs by(rule rBang)
     with ‹P = Q› show ?case by simp
   qed
 }
 with ‹guarded P› ‹Ψ ▷ !P ⟼ Rs›
 show ?thesis by(force intro: aux1)
qed

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'
using $\langle \Psi \rhd !P \longmapsto M(\!|N|\!) \prec P' \rangle$
by(*nominal-induct* Ψ P *Rs*==$M(\!|N|\!) \prec P'$ *arbitrary*: P' *rule*: *bangInduct*)
(*auto simp add*: *residualInject intro*: *rPar1 rPar2 rBang*)

lemma *brbangInputInduct*[*consumes 1*, *case-names cPar1 cPar2 cBrMerge 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 *rBrMerge*: $\bigwedge P'\ P''\ C.\ [\![\Psi \rhd P \longmapsto ¿M(\!|N|\!) \prec P';\ \Psi \rhd !P \longmapsto ¿M(\!|N|\!) \prec P'';\ \bigwedge C.$ *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'
using $\langle \Psi \rhd !P \longmapsto ¿M(\!|N|\!) \prec P' \rangle$
by(*nominal-induct* Ψ P *Rs*==$¿M(\!|N|\!) \prec P'$ *arbitrary*: P' *rule*: *bangInduct*)
(*auto simp add*: *residualInject intro*: *rPar1 rPar2 rBrMerge rBang*)

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*: $\bigwedge xvec\ N\ P'\ C.\ [\![\Psi \rhd P \longmapsto M(\!|\nu*xvec|\!)\langle N\rangle \prec P';\ xvec \sharp* \Psi;\ xvec \sharp* P;\ xvec \sharp* M;\ xvec \sharp* C;\ distinct\ xvec]\!] \Longrightarrow Prop\ C\ \Psi\ (P \parallel !P)\ M\ ((\!|\nu*xvec|\!)N \prec' (P' \parallel !P))$
and *rPar2*: $\bigwedge xvec\ N\ P'\ C.\ [\![\Psi \rhd !P \longmapsto M(\!|\nu*xvec|\!)\langle N\rangle \prec P';\ \bigwedge C.\ Prop\ C\ \Psi\ (!P)\ M\ ((\!|\nu*xvec|\!)N \prec' P');\ xvec \sharp* \Psi;\ xvec \sharp* P;\ xvec \sharp* M;\ xvec \sharp* C;\ distinct\ xvec]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (P \parallel !P)\ M\ ((\!|\nu*xvec|\!)N \prec' (P \parallel P'))$
and *rBang*: $\bigwedge B\ C.\ [\![\Psi \rhd P \parallel !P \longmapsto (ROut\ M\ B);\ \bigwedge C.\ Prop\ C\ \Psi\ (P \parallel !P)\ M\ B;\ guarded\ P]\!] \Longrightarrow Prop\ C\ \Psi\ (!P)\ M\ B$

shows $Prop\ C\ \Psi\ (!P)\ M\ ((\!|\nu*xvec|\!)N \prec' P')$
using $\langle\Psi \rhd !P \longmapsto M(\!|\nu*xvec|\!)\langle N\rangle \prec P'\rangle$
apply(*simp add*: *residualInject*)
by(*nominal-induct* Ψ P *Rs*==*ROut M* $((\!|\nu*xvec|\!)N \prec' P')$ *avoiding*: *C arbitrary*: *xvec N* P' *rule*: *bangInduct*)
(*force simp add*: *residualInject intro*: *rPar1 rPar2 rBang*)+

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 $Prop\ C\ \Psi\ (!P)\ P'$
using $\langle\Psi \rhd !P \longmapsto \tau \prec P'\rangle$
by(*nominal-induct* Ψ P *Rs*==$\tau \prec P'$ *avoiding*: *C arbitrary*: P' *rule*: *bangInduct*)
(*auto simp add*: *residualInject intro*: *rPar1 rPar2 rComm1 rComm2 rBang*)

lemma *bangInduct'*[*consumes 2*, *case-names cAlpha cPar1 cPar2 cComm1 cComm2*

cBrMerge cBrComm1 cBrComm2 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 *rBrMerge*: $\bigwedge M\ N\ P'\ P''\ C.\ [\![\Psi \rhd P \longmapsto ¿M(\!|N|\!) \prec P';\ \Psi \rhd\, !P \longmapsto ¿M(\!|N|\!) \prec P'';\ \bigwedge C.\ Prop\ C\ \Psi\ (!P)\ (¿M(\!|N|\!))\ P']\!] \Longrightarrow$
$Prop\ C\ \Psi\ (P \parallel\, !P)\ (¿M(\!|N|\!))\ (P' \parallel\, P'')$
and *rBrComm1*: $\bigwedge M\ N\ P'\ xvec\ P''\ C.\ [\![\Psi \rhd P \longmapsto ¿M(\!|N|\!) \prec P';\ \Psi \rhd\, !P \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec P'';\ \bigwedge C.\ Prop\ C\ \Psi\ (!P)\ (¡M(\!|\nu * xvec|\!)\langle N\rangle)\ P'';$
$xvec \sharp* \Psi;\ xvec \sharp* P;\ xvec \sharp* M;\ xvec \sharp* C;\ distinct\ xvec]\!] \Longrightarrow Prop\ C\ \Psi\ (P \parallel\, !P)\ (¡M(\!|\nu * xvec|\!)\langle N\rangle)\ (P' \parallel\, P'')$
and *rBrComm2*: $\bigwedge M\ N\ P'\ xvec\ P''\ C.\ [\![\Psi \rhd P \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec P';\ \Psi \rhd\, !P \longmapsto ¿M(\!|N|\!) \prec P'';\ \bigwedge C.\ Prop\ C\ \Psi\ (!P)\ (¿M(\!|N|\!))\ P'';$
$xvec \sharp* \Psi;\ xvec \sharp* P;\ xvec \sharp* M;\ xvec \sharp* C;\ distinct\ xvec]\!] \Longrightarrow Prop\ C\ \Psi\ (P \parallel\, !P)\ (¡M(\!|\nu * xvec|\!)\langle N\rangle)\ (P' \parallel\, P'')$
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 −
from ‹$\Psi \rhd !P \longmapsto\alpha \prec P'$› **have** *distinct*($bn\ \alpha$) **by**(*rule boundOutputDistinct*)
with ‹$\Psi \rhd !P \longmapsto\alpha \prec P'$› ‹$bn\ \alpha \sharp* subject\ \alpha$› **show** *?thesis*
proof(*nominal-induct* Ψ P $Rs==\alpha \prec P'$ *avoiding*: C α P' *rule*: *bangInduct*)
case(*cPar1* α P' C α' P'')
note ‹$\alpha \prec (P' \parallel !P) = \alpha' \prec P''$›
moreover from ‹$bn\ \alpha \sharp* \alpha'$› **have** $bn\ \alpha \sharp* bn\ \alpha'$ **by** *simp*
moreover note ‹*distinct*($bn\ \alpha$)› ‹*distinct*($bn\ \alpha'$)›
moreover from ‹$bn\ \alpha \sharp* subject\ \alpha$› **have** $bn\ \alpha \sharp* (\alpha \prec P' \parallel !P)$ **by** *simp*
moreover from ‹$bn\ \alpha' \sharp* subject\ \alpha'$› **have** $bn\ \alpha' \sharp* (\alpha' \prec P'')$ **by** *simp*
ultimately obtain p **where** S: *set* $p \subseteq$ *set*($bn\ \alpha$) $\times$ *set*($bn(p \cdot \alpha)$) **and** *distinctPerm* p **and** $\alpha' = p \cdot \alpha$
and $P'eq$: $P'' = p \cdot (P' \parallel !P)$ **and** $bn(p \cdot \alpha) \sharp* \alpha$ **and** $bn(p \cdot \alpha) \sharp* (P' \parallel !P)$
by(*elim residualEq*)

from ‹$\Psi \rhd P \longmapsto\alpha \prec P'$› ‹$bn\ \alpha \sharp* \Psi$› ‹$bn\ \alpha \sharp* P$› ‹$bn\ \alpha \sharp* subject\ \alpha$› ‹$bn\ \alpha \sharp* C$› ‹*distinct*($bn\ \alpha$)›
have *Prop* C Ψ $(P \parallel !P)$ α $(P' \parallel !P)$
by(*rule rPar1*)

with ‹$bn\ \alpha \sharp* \Psi$› ‹$bn\ \alpha \sharp* P$› ‹$bn\ \alpha \sharp* subject\ \alpha$› ‹$bn\ \alpha \sharp* C$› S ‹*distinctPerm* p› ‹$bn\ \alpha \sharp* \alpha'$› ‹$bn\ \alpha \sharp* P''$› ‹$\alpha' = (p \cdot \alpha)$› $P'eq$ ‹$bn(p \cdot \alpha) \sharp* \alpha$› ‹$bn(p \cdot \alpha) \sharp* (P' \parallel !P)$›
have *Prop* C Ψ $(P \parallel !P)$ $(p \cdot \alpha)$ $(p \cdot (P' \parallel !P))$
by(*elim rAlpha*)
with $P'eq$ ‹$\alpha' = p \cdot \alpha$› ‹*distinctPerm* p› **show** *?case* **by** *simp*
next
case(*cPar2* α P' C α' P'')
note ‹$\alpha \prec (P \parallel P') = \alpha' \prec P''$›
moreover from ‹$bn\ \alpha \sharp* \alpha'$› **have** $bn\ \alpha \sharp* bn\ \alpha'$ **by** *simp*
moreover note ‹*distinct*($bn\ \alpha$)› ‹*distinct*($bn\ \alpha'$)›
moreover from ‹$bn\ \alpha \sharp* subject\ \alpha$› **have** $bn\ \alpha \sharp* (\alpha \prec P \parallel P')$ **by** *simp*
moreover from ‹$bn\ \alpha' \sharp* subject\ \alpha'$› **have** $bn\ \alpha' \sharp* (\alpha' \prec P'')$ **by** *simp*
ultimately obtain p **where** S: *set* $p \subseteq$ *set*($bn\ \alpha$) $\times$ *set*($bn(p \cdot \alpha)$) **and** *distinctPerm* p **and** $\alpha' = p \cdot \alpha$
and $P'eq$: $P'' = p \cdot (P \parallel P')$ **and** $bn(p \cdot \alpha) \sharp* \alpha$ **and** $bn(p \cdot \alpha) \sharp* (P \parallel P')$
by(*elim residualEq*)

note ‹$\Psi \rhd !P \longmapsto\alpha \prec P'$›
moreover from ‹$bn\ \alpha \sharp* subject\ \alpha$› ‹*distinct*($bn\ \alpha$)› **have** $\bigwedge C$. *Prop* C Ψ $(!P)$ α P' **by**(*intro cPar2*) *auto*
moreover note ‹$bn\ \alpha \sharp* \Psi$› ‹$bn\ \alpha \sharp* P$› ‹$bn\ \alpha \sharp* subject\ \alpha$› ‹$bn\ \alpha \sharp* C$› ‹*distinct*($bn\ \alpha$)›
ultimately have *Prop* C Ψ $(P \parallel !P)$ α $(P \parallel P')$
by(*rule rPar2*)
with ‹$bn\ \alpha \sharp* \Psi$› ‹$bn\ \alpha \sharp* P$› ‹$bn\ \alpha \sharp* subject\ \alpha$› ‹$bn\ \alpha \sharp* C$› S ‹*distinctPerm* p› ‹$bn\ \alpha \sharp* \alpha'$› ‹$bn\ \alpha \sharp* P''$› ‹$\alpha' = (p \cdot \alpha)$› $P'eq$ ‹$bn(p \cdot \alpha) \sharp* \alpha$› ‹$bn(p \cdot \alpha) \sharp* (P \parallel P')$›
have *Prop* C Ψ $(P \parallel !P)$ $(p \cdot \alpha)$ $(p \cdot (P \parallel P'))$

by(*elim rAlpha*)
with *P'eq* ‹$\alpha' = p \cdot \alpha$› **show** *?case* **by** *simp*
next
case(*cComm1 M N P' K xvec P'' C α P'''*)
then have *Prop C Ψ* (*P* ‖ !*P*) (*τ*) ((⦇*ν*∗*xvec*⦈)(*P'* ‖ *P''*))
by(*elim rComm1*) (*assumption* | *simp*)+
then show *?case* **using** ‹$\tau \prec (\!|\nu{*}xvec|\!)(P' \parallel P'') = \alpha \prec P'''$›
by(*simp add*: *residualInject*)
next
case(*cComm2 M xvec N P' K P'' C α P'''*)
then have *Prop C Ψ* (*P* ‖ !*P*) (*τ*) ((⦇*ν*∗*xvec*⦈)(*P'* ‖ *P''*))
by(*elim rComm2*) (*assumption* | *simp*)+
then show *?case* **using** ‹$\tau \prec (\!|\nu{*}xvec|\!)(P' \parallel P'') = \alpha \prec P'''$›
by(*simp add*: *residualInject*)
next
case(*cBrMerge M N P' P'' C α P'''*)
then have *Prop C Ψ* (*P* ‖ !*P*) (¿*M*⦇*N*⦈) (*P'* ‖ *P''*)
by(*elim rBrMerge*) (*assumption* | *simp*)+
then show *?case* **using** ‹¿*M*⦇*N*⦈ ≺ *P'* ‖ *P''* = *α* ≺ *P'''*›
by(*simp add*: *residualInject*)
next
case(*cBrComm1 M N P' xvec P'' C α P'''*)
note ‹(¡*M*⦇*ν*∗*xvec*⦈⟨*N*⟩) ≺ (*P'* ‖ *P''*) = *α* ≺ *P'''*›
moreover from ‹*xvec* ♯∗ *α*› **have** *xvec* ♯∗ *bn α* **by** *simp*
moreover note ‹*distinct xvec*› ‹*distinct*(*bn α*)›
moreover from ‹*xvec* ♯∗ *M*› **have** *xvec* ♯∗ ((¡*M*⦇*ν*∗*xvec*⦈⟨*N*⟩) ≺ (*P'* ‖ *P''*)) **by** *simp*
moreover from ‹*bn α* ♯∗ *subject α*› **have** *bn α* ♯∗ (*α* ≺ *P'''*) **by** *simp*
ultimately obtain *p* **where** *S*: *set p* ⊆ *set xvec* × *set*(*p* · *xvec*) **and** *distinctPerm p* **and** *α* = *p* · (¡*M*⦇*ν*∗*xvec*⦈⟨*N*⟩)
and *P'eq*: *P'''* = *p* · (*P'* ‖ *P''*) **and** (*p* · *xvec*) ♯∗ (¡*M*⦇*ν*∗*xvec*⦈⟨*N*⟩) **and** (*p* · *xvec*) ♯∗ (*P'* ‖ *P''*)
using *residualEq*[**where** *α*=(¡*M*⦇*ν*∗*xvec*⦈⟨*N*⟩) **and** *β*=*α*] **by** (*smt* (*verit*, *best*) *Sigma-cong bn.simps*(*4*) *bnEqvt*)

from *cBrComm1* **have** *Prop C Ψ* (*P* ‖ !*P*) (¡*M*⦇*ν*∗*xvec*⦈⟨*N*⟩) (*P'* ‖ *P''*)
by(*elim rBrComm1*) (*assumption* | *simp*)+

with ‹*xvec* ♯∗ *Ψ*› ‹*xvec* ♯∗ *P*› ‹*xvec* ♯∗ *M*› ‹*xvec* ♯∗ *C*› *S* ‹*distinctPerm p*› ‹*xvec* ♯∗ *α*› ‹*xvec* ♯∗ *P'''*› ‹*α* = (*p* · (¡*M*⦇*ν*∗*xvec*⦈⟨*N*⟩))› *P'eq* ‹(*p* · *xvec*) ♯∗ (¡*M*⦇*ν*∗*xvec*⦈⟨*N*⟩)› ‹(*p* · *xvec*) ♯∗ (*P'* ‖ *P''*)›
have *Prop C Ψ* (*P* ‖ !*P*) (*p* · (¡*M*⦇*ν*∗*xvec*⦈⟨*N*⟩)) (*p* · (*P'* ‖ *P''*))
by(*intro rAlpha*) *simp*+

with *P'eq* ‹*α* = *p* · (¡*M*⦇*ν*∗*xvec*⦈⟨*N*⟩)› ‹*distinctPerm p*› **show** *?case* **by** *simp*
next
case(*cBrComm2 M N P' xvec P'' C α P'''*)
note ‹(¡*M*⦇*ν*∗*xvec*⦈⟨*N*⟩) ≺ (*P'* ‖ *P''*) = *α* ≺ *P'''*›
moreover from ‹*xvec* ♯∗ *α*› **have** *xvec* ♯∗ *bn α* **by** *simp*

moreover note ‹*distinct xvec*› ‹*distinct*(*bn* α)›
moreover from ‹*xvec* $\sharp*$ *M*› **have** *xvec* $\sharp*$ (($¡M(\!|\nu*xvec|\!)\langle N\rangle$) $\prec$ ($P' \parallel P''$)) **by** *simp*
moreover from ‹*bn* α $\sharp*$ *subject* α› **have** *bn* α $\sharp*$ ($\alpha \prec P'''$) **by** *simp*
ultimately obtain *p* **where** *S*: *set p* $\subseteq$ *set xvec* $\times$ *set*(*p* $\cdot$ *xvec*) **and** *distinctPerm p* **and** $\alpha = p \cdot (¡M(\!|\nu*xvec|\!)\langle N\rangle)$
and *P'eq*: $P''' = p \cdot (P' \parallel P'')$ **and** (*p* $\cdot$ *xvec*) $\sharp*$ ($¡M(\!|\nu*xvec|\!)\langle N\rangle$) **and** (*p* $\cdot$ *xvec*) $\sharp*$ ($P' \parallel P''$)
using *residualEq*[**where** α=($¡M(\!|\nu*xvec|\!)\langle N\rangle$) **and** β=α] **by** (*smt* (*verit*, *best*) *Sigma-cong bn.simps*(*4*) *bnEqvt*)

from *cBrComm2* **have** *Prop C* Ψ ($P \parallel !P$) ($¡M(\!|\nu*xvec|\!)\langle N\rangle$) ($P' \parallel P''$)
by(*elim rBrComm2*) (*assumption* | *simp*)+

with ‹*xvec* $\sharp*$ Ψ› ‹*xvec* $\sharp*$ *P*› ‹*xvec* $\sharp*$ *M*› ‹*xvec* $\sharp*$ *C*› *S* ‹*distinctPerm p*› ‹*xvec* $\sharp*$ α› ‹*xvec* $\sharp*$ P'''› ‹$\alpha = (p \cdot (¡M(\!|\nu*xvec|\!)\langle N\rangle))$› *P'eq* ‹(*p* $\cdot$ *xvec*) $\sharp*$ ($¡M(\!|\nu*xvec|\!)\langle N\rangle$)› ‹(*p* $\cdot$ *xvec*) $\sharp*$ ($P' \parallel P''$)›
have *Prop C* Ψ ($P \parallel !P$) (*p* $\cdot$ ($¡M(\!|\nu*xvec|\!)\langle N\rangle$)) (*p* $\cdot$ ($P' \parallel P''$))
by(*intro rAlpha*) *simp*+

with *P'eq* ‹$\alpha = p \cdot (¡M(\!|\nu*xvec|\!)\langle N\rangle)$› ‹*distinctPerm p*› **show** *?case* **by** *simp*
next
case(*cBang C* α *P'*)
then show *?case* **by**(*auto intro*: *rBang*)
qed
qed

lemma *brCommInAuxTooMuch*:
fixes Ψ :: *'b*
and Ψ_Q :: *'b*
and *R* :: (*'a*, *'b*, *'c*) *psi*
and *M* :: *'a*
and *N* :: *'a*
and *R'* :: (*'a*, *'b*, *'c*) *psi*
and A_R :: *name list*
and Ψ_R :: *'b*
and A_P :: *name list*
and Ψ_P :: *'b*
and A_Q :: *name list*

assumes *RTrans*: $\Psi \otimes \Psi_Q \rhd R \longmapsto ¿M(\!|N|\!) \prec R'$
and *FrR*: *extractFrame R* = $\langle A_R, \Psi_R\rangle$
and *distinct* A_R
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 $A_R \sharp* A_P$
and $A_R \sharp* A_Q$
and $A_R \sharp* \Psi$
and $A_R \sharp* \Psi_P$
and $A_R \sharp* \Psi_Q$

and $A_R \sharp* (\Psi \otimes \Psi_Q)$
and $A_R \sharp* R$
and $A_R \sharp* M$
and $A_P \sharp* R$
and $A_P \sharp* M$
and $A_Q \sharp* R$
and $A_Q \sharp* M$

shows $\Psi \otimes \Psi_P \rhd R \longmapsto ¿M(\!|N|\!) \prec R'$
using *assms*
proof(*nominal-induct avoiding*: $A_P\ \Psi_P\ A_Q\ \Psi_Q\ \Psi$ *rule*: *brinputFrameInduct*)
case(*cAlpha* $\Psi'\ P\ M\ N\ P'\ A_R\ \Psi_R\ p\ A_P\ \Psi_P\ A_Q\ \Psi_Q\ \Psi$)
have *S*: *set* $p \subseteq$ *set* $A_R \times$ *set* $(p \cdot A_R)$ **by** *fact*
from $\langle\langle A_Q, \Psi' \otimes (p \cdot \Psi_R)\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes (p \cdot \Psi_R)\rangle\rangle$
have $(p \cdot \langle A_Q, \Psi' \otimes (p \cdot \Psi_R)\rangle) \hookrightarrow_F (p \cdot \langle A_P, (\Psi \otimes \Psi_P) \otimes (p \cdot \Psi_R)\rangle)$
by(*rule FrameStatImpClosed*)
with $\langle A_R \sharp* A_P\rangle\ \langle(p \cdot A_R) \sharp* A_P\rangle\ \langle A_R \sharp* \Psi'\rangle\ \langle(p \cdot A_R) \sharp* \Psi'\rangle\ \langle A_R \sharp* \Psi_P\rangle\ \langle(p \cdot A_R) \sharp* \Psi_P\rangle\ \langle A_R \sharp* A_Q\rangle$
$\langle(p \cdot A_R) \sharp* A_Q\rangle\ \langle A_R \sharp* \Psi\rangle\ \langle(p \cdot A_R) \sharp* \Psi\rangle\ S\ \langle$*distinctPerm* $p\rangle$
have $\langle A_Q, \Psi' \otimes \Psi_R\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R\rangle$ **by**(*simp add*: *eqvts*)
moreover note $\langle A_R \sharp* \Psi'\rangle\ \langle A_R \sharp* \Psi\rangle\ \langle A_R \sharp* P\rangle\ \langle A_R \sharp* \Psi_P\rangle\ \langle A_R \sharp* \Psi_Q\rangle\ \langle A_R \sharp* M\rangle\ \langle A_P \sharp* P\rangle$
$\langle A_Q \sharp* P\rangle\ \langle A_P \sharp* M\rangle\ \langle A_Q \sharp* M\rangle\ \langle A_R \sharp* A_P\rangle\ \langle A_R \sharp* A_Q\rangle$
ultimately show *?case*
by(*elim cAlpha*)
next
case(*cBrInput* $\Psi'\ M\ K\ xvec\ N\ Tvec\ P\ A_P\ \Psi_P\ A_Q\ \Psi_Q\ \Psi$)
from $\langle A_P \sharp* (M(\!|\lambda*xvec\ N|\!).P)\rangle\ \langle A_Q \sharp* (M(\!|\lambda*xvec\ N|\!).P)\rangle$
have $A_P \sharp* M$ **and** $A_Q \sharp* M$ **by** *simp+*
from $\langle\Psi' \vdash K \succeq M\rangle$
have $\Psi' \otimes \mathbf{1} \vdash K \succeq M$
by(*blast intro*: *statEqEnt Identity AssertionStatEqSym*)
with $\langle A_Q \sharp* K\rangle\ \langle A_Q \sharp* M\rangle$
have $(\langle A_Q, \Psi' \otimes \mathbf{1}\rangle) \vdash_F K \succeq M$
by(*force intro*: *frameImpI*)
with $\langle\langle A_Q, \Psi' \otimes \mathbf{1}\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1}\rangle\rangle$
have $(\langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1}\rangle) \vdash_F K \succeq M$
by(*simp add*: *FrameStatImp-def*)
with $\langle A_P \sharp* K\rangle\ \langle A_P \sharp* M\rangle$ **have** $(\Psi \otimes \Psi_P) \otimes \mathbf{1} \vdash K \succeq M$
by(*force dest*: *frameImpE*)
then have $\Psi \otimes \Psi_P \vdash K \succeq M$ **by**(*blast intro*: *statEqEnt Identity*)
then show *?case* **using** $\langle$*distinct xvec*$\rangle\ \langle$*set xvec* $\subseteq$ *supp N*$\rangle\ \langle$*length xvec* $=$ *length Tvec*$\rangle$
by(*rule BrInput*)
next
case(*cCase* $\Psi'\ P\ M\ N\ P'\ \varphi\ Cs\ A_R\ \Psi_R\ A_Q\ \Psi_Q\ A_P\ \Psi_P\ \Psi$)
from $\langle A_P \sharp* (Cases\ Cs)\rangle\ \langle A_Q \sharp* (Cases\ Cs)\rangle\ \langle(\varphi, P) \in set\ Cs\rangle$
have $A_P \sharp* P$ **and** $A_Q \sharp* P$ **and** $A_P \sharp* \varphi$ **and** $A_Q \sharp* \varphi$
by(*auto dest*: *memFreshChain*)

from ‹$\Psi' \vdash \varphi$› **have** $\Psi' \otimes \mathbf{1} \vdash \varphi$ **by**(*blast intro*: *statEqEnt Identity AssertionStatEqSym*) **with** ‹$A_P \sharp* \varphi$› **have** $(\langle A_P, \Psi' \otimes \mathbf{1}\rangle) \vdash_F \varphi$ **by**(*force intro*: *frameImpI*) **with** ‹$\langle A_P, \Psi' \otimes \mathbf{1}\rangle \hookrightarrow_F \langle A_Q, (\Psi \otimes \Psi_Q) \otimes \mathbf{1}\rangle$› **have** $(\langle A_Q, (\Psi \otimes \Psi_Q) \otimes \mathbf{1}\rangle) \vdash_F \varphi$ **by**(*simp add*: *FrameStatImp-def*) **with** ‹$A_Q \sharp* \varphi$› **have** $(\Psi \otimes \Psi_Q) \otimes \mathbf{1} \vdash \varphi$ **by**(*force dest*: *frameImpE*) **then have** $\Psi \otimes \Psi_Q \vdash \varphi$ **by**(*blast intro*: *statEqEnt Identity*)

have $\langle A_P, \Psi' \otimes \Psi_R\rangle \hookrightarrow_F \langle A_Q, (\Psi \otimes \Psi_Q) \otimes \Psi_R\rangle$ **proof** − **from** ‹$\Psi_R \simeq \mathbf{1}$› **have** $\langle A_P, \Psi' \otimes \Psi_R\rangle \simeq_F \langle A_P, \Psi' \otimes \mathbf{1}\rangle$ **by**(*metis Identity Commutativity AssertionStatEqSym Composition frameResChainPres frameNilStatEq AssertionStatEqTrans*) **moreover note** ‹$\langle A_P, \Psi' \otimes \mathbf{1}\rangle \hookrightarrow_F \langle A_Q, (\Psi \otimes \Psi_Q) \otimes \mathbf{1}\rangle$› **moreover from** ‹$\Psi_R \simeq \mathbf{1}$› **have** $\langle A_Q, (\Psi \otimes \Psi_Q) \otimes \mathbf{1}\rangle \simeq_F \langle A_Q, (\Psi \otimes \Psi_Q) \otimes \Psi_R\rangle$ **by**(*metis Identity Commutativity AssertionStatEqSym Composition frameResChainPres frameNilStatEq AssertionStatEqTrans*) **ultimately show** *?thesis* **by**(*rule FrameStatEqImpCompose*) **qed** **moreover note** ‹$A_R \sharp* \Psi'$› ‹$A_R \sharp* \Psi$› ‹$A_R \sharp* P$› ‹$A_R \sharp* \Psi_Q$› ‹$A_R \sharp* M$› ‹$A_Q \sharp* P$› ‹$A_P \sharp* P$› ‹$A_Q \sharp* M$› ‹$A_P \sharp* M$› ‹$A_R \sharp* A_Q$› ‹$A_R \sharp* A_P$› **ultimately have** $\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$ **by**(*elim cCase*(*4*)) **moreover note** ‹$(\varphi, P) \in$ *set Cs*› ‹$\Psi \otimes \Psi_Q \vdash \varphi$› ‹*guarded P*› **ultimately show** *?case* **by**(*rule Case*)

next

case(*cPar1* $\Psi'\ \Psi_Q\ P\ M\ N\ P'\ A_Q\ Q\ A_P\ \Psi_P\ A_P'\ \Psi_P'\ A_Q'\ \Psi_Q'\ \Psi$) **from** ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› ‹$A_Q \sharp* A_P'$› ‹$A_P' \sharp* (P \parallel Q)$› **have** $A_P' \sharp* \Psi_Q$ **by**(*force dest*: *extractFrameFreshChain*)

have $\langle A_P', (\Psi \otimes \Psi_P') \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_P', (\Psi \otimes \Psi_P') \otimes \Psi_Q \otimes \Psi_P\rangle$ **by** (*metis Commutativity FrameStatEq-def frameIntCompositionSym*) **moreover have** $\langle A_P', (\Psi \otimes \Psi_P') \otimes \Psi_Q \otimes \Psi_P\rangle \hookrightarrow_F \langle A_P', ((\Psi \otimes \Psi_P') \otimes \Psi_Q) \otimes \Psi_P\rangle$ **by** (*metis FrameStatEq-def frameIntAssociativity*) **moreover have** $\langle A_P', ((\Psi \otimes \Psi_P') \otimes \Psi_Q) \otimes \Psi_P\rangle \hookrightarrow_F \langle A_P', ((\Psi \otimes \Psi_Q) \otimes \Psi_P') \otimes \Psi_P\rangle$ **by** (*metis FrameStatEq-def associativitySym frameIntComposition*) **ultimately have** *right*: $\langle A_P', (\Psi \otimes \Psi_P') \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_P', ((\Psi \otimes \Psi_Q) \otimes \Psi_P') \otimes \Psi_P\rangle$

by (*metis FrameStatImpTrans*)
have $\langle A_Q{}', \Psi' \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_Q{}', \Psi' \otimes \Psi_Q \otimes \Psi_P\rangle$
by (*metis AssertionStatEq-def Commutativity compositionSym frameImpNilStatEq frameImpResChainPres*)
moreover have $\langle A_Q{}', \Psi' \otimes \Psi_Q \otimes \Psi_P\rangle \hookrightarrow_F \langle A_Q{}', (\Psi' \otimes \Psi_Q) \otimes \Psi_P\rangle$
by (*metis FrameStatEq-def frameIntAssociativity*)
ultimately have *left*: $\langle A_Q{}', \Psi' \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_Q{}', (\Psi' \otimes \Psi_Q) \otimes \Psi_P\rangle$
by (*metis FrameStatImpTrans*)
from $\langle\langle A_Q{}', \Psi' \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_P{}', (\Psi \otimes \Psi_P{}') \otimes \Psi_P \otimes \Psi_Q\rangle\rangle$ *left right*
have $\langle A_Q{}', (\Psi' \otimes \Psi_Q) \otimes \Psi_P\rangle \hookrightarrow_F \langle A_P{}', ((\Psi \otimes \Psi_Q) \otimes \Psi_P{}') \otimes \Psi_P\rangle$
by (*metis AssertionStatEqSym AssertionStatEqTrans AssertionStatEq-def Associativity FrameStatImpTrans associativitySym frameImpNilStatEq frameImpResChainPres*)
moreover note $\langle A_P \sharp* \Psi'\rangle$ $\langle A_P \sharp* \Psi_Q\rangle$ $\langle A_P \sharp* \Psi\rangle$ $\langle A_P \sharp* \Psi_Q\rangle$ $\langle A_P \sharp* P\rangle$ $\langle A_P \sharp* M\rangle$
$\langle A_P \sharp* \Psi_P{}'\rangle$ $\langle A_P{}' \sharp* (P \parallel Q)\rangle$ $\langle A_Q{}' \sharp* (P \parallel Q)\rangle$ $\langle A_P{}' \sharp* \Psi_Q\rangle$ $\langle A_P{}' \sharp* M\rangle$ $\langle A_Q{}' \sharp* M\rangle$ $\langle A_P \sharp* A_P{}'\rangle$ $\langle A_P \sharp* A_Q{}'\rangle$
ultimately have $(\Psi \otimes \Psi_Q) \otimes \Psi_P{}' \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$
by(*elim cPar1*(*6*)) (*simp* | *force*)+
then have $(\Psi \otimes \Psi_P{}') \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$
by (*metis associativitySym statEqTransition*)

moreover note $\langle$*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle\rangle$ $\langle A_Q \sharp* \Psi\rangle$ $\langle A_Q \sharp* \Psi_P{}'\rangle$
$\langle A_Q \sharp* P\rangle$ $\langle A_Q \sharp* M\rangle$ $\langle A_Q \sharp* N\rangle$
ultimately show *?case*
by(*elim Par1*) (*simp* | *force*)+
next
case(*cPar2* Ψ' Ψ_P Q M N Q' A_P P A_Q Ψ_Q $A_P{}'$ $\Psi_P{}'$ $A_Q{}'$ $\Psi_Q{}'$ Ψ)
from $\langle$*extractFrame* $P = \langle A_P, \Psi_P\rangle\rangle$ $\langle A_P \sharp* A_P{}'\rangle$ $\langle A_P{}' \sharp* (P \parallel Q)\rangle$ **have** $A_P{}' \sharp* \Psi_P$
by(*force dest*: *extractFrameFreshChain*)
have $\langle A_P{}', (\Psi \otimes \Psi_P{}') \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_P{}', ((\Psi \otimes \Psi_P{}') \otimes \Psi_P) \otimes \Psi_Q\rangle$
by (*metis FrameStatEq-def frameIntAssociativity*)
moreover have $\langle A_P{}', ((\Psi \otimes \Psi_P{}') \otimes \Psi_P) \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_P{}', ((\Psi \otimes \Psi_P) \otimes \Psi_P{}') \otimes \Psi_Q\rangle$
by (*metis FrameStatEq-def associativitySym frameIntComposition*)
ultimately have *right*: $\langle A_P{}', (\Psi \otimes \Psi_P{}') \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_P{}', ((\Psi \otimes \Psi_P) \otimes \Psi_P{}') \otimes \Psi_Q\rangle$
by (*metis FrameStatImpTrans*)
moreover have *left*: $\langle A_Q{}', \Psi' \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_Q{}', (\Psi' \otimes \Psi_P) \otimes \Psi_Q\rangle$
by (*metis FrameStatEq-def frameIntAssociativity*)
from $\langle\langle A_Q{}', \Psi' \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_P{}', (\Psi \otimes \Psi_P{}') \otimes \Psi_P \otimes \Psi_Q\rangle\rangle$ *left right*
have $\langle A_Q{}', (\Psi' \otimes \Psi_P) \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_P{}', ((\Psi \otimes \Psi_P) \otimes \Psi_P{}') \otimes \Psi_Q\rangle$
by (*metis FrameStatEq-def FrameStatImpTrans frameIntAssociativity*)
moreover note $\langle A_Q \sharp* \Psi'\rangle$ $\langle A_Q \sharp* \Psi_P\rangle$ $\langle A_Q \sharp* \Psi\rangle$ $\langle A_Q \sharp* \Psi_P\rangle$ $\langle A_Q \sharp* Q\rangle$ $\langle A_Q \sharp* M\rangle$
$\langle A_Q \sharp* \Psi_P{}'\rangle$ $\langle A_P{}' \sharp* (P \parallel Q)\rangle$ $\langle A_Q{}' \sharp* (P \parallel Q)\rangle$ $\langle A_P{}' \sharp* \Psi_P\rangle$ $\langle A_P{}' \sharp* M\rangle$ $\langle A_Q{}' \sharp* M\rangle$ $\langle A_Q \sharp* A_Q{}'\rangle$ $\langle A_Q \sharp* A_P{}'\rangle$
ultimately have $(\Psi \otimes \Psi_P) \otimes \Psi_P{}' \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$

by(*elim cPar2*(*6*)) (*simp* | *force*)+
then have $(\Psi \otimes {\Psi_P}') \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$
by (*metis associativitySym statEqTransition*)
moreover note ‹*extractFrame* $P = \langle A_P, \Psi_P\rangle$› ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* {\Psi_P}'$›
‹$A_P \sharp* Q$› ‹$A_P \sharp* M$› ‹$A_P \sharp* N$›
ultimately show *?case*
by(*elim Par2*) (*simp* | *force*)+
next
case(*cBrMerge* $\Psi'\ \Psi_Q\ P\ M\ N\ P'\ A_P\ \Psi_P\ Q\ Q'\ A_Q\ {A_P}'\ \Psi_P'\ {A_Q}'\ \Psi_Q'\ \Psi$)
from ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› ‹$A_Q \sharp* {A_P}'$› ‹${A_P}' \sharp* (P \parallel Q)$› **have** ${A_P}' \sharp* \Psi_Q$
by(*force dest*: *extractFrameFreshChain*)
from ‹*extractFrame* $P = \langle A_P, \Psi_P\rangle$› ‹$A_P \sharp* {A_P}'$› ‹${A_P}' \sharp* (P \parallel Q)$› **have** ${A_P}' \sharp* \Psi_P$
by(*force dest*: *extractFrameFreshChain*)
have $\langle {A_P}', (\Psi \otimes {\Psi_P}') \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle {A_P}', (\Psi \otimes {\Psi_P}') \otimes \Psi_Q \otimes \Psi_P\rangle$
by (*metis Commutativity FrameStatEq-def frameIntCompositionSym*)
moreover have $\langle {A_P}', (\Psi \otimes {\Psi_P}') \otimes \Psi_Q \otimes \Psi_P\rangle \hookrightarrow_F \langle {A_P}', ((\Psi \otimes {\Psi_P}') \otimes \Psi_Q) \otimes \Psi_P\rangle$
by (*metis FrameStatEq-def frameIntAssociativity*)
moreover have $\langle {A_P}', ((\Psi \otimes {\Psi_P}') \otimes \Psi_Q) \otimes \Psi_P\rangle \hookrightarrow_F \langle {A_P}', ((\Psi \otimes \Psi_Q) \otimes {\Psi_P}') \otimes \Psi_P\rangle$
by (*metis FrameStatEq-def associativitySym frameIntComposition*)
ultimately have *right*: $\langle {A_P}', (\Psi \otimes {\Psi_P}') \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle {A_P}', ((\Psi \otimes \Psi_Q) \otimes {\Psi_P}') \otimes \Psi_P\rangle$
by (*metis FrameStatImpTrans*)
have $\langle {A_Q}', \Psi' \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle {A_Q}', \Psi' \otimes \Psi_Q \otimes \Psi_P\rangle$
by (*metis AssertionStatEq-def Commutativity compositionSym frameImpNilStatEq frameImpResChainPres*)
moreover have $\langle {A_Q}', \Psi' \otimes \Psi_Q \otimes \Psi_P\rangle \hookrightarrow_F \langle {A_Q}', (\Psi' \otimes \Psi_Q) \otimes \Psi_P\rangle$
by (*metis FrameStatEq-def frameIntAssociativity*)
ultimately have *left*: $\langle {A_Q}', \Psi' \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle {A_Q}', (\Psi' \otimes \Psi_Q) \otimes \Psi_P\rangle$
by (*metis FrameStatImpTrans*)
from ‹$\langle {A_Q}', \Psi' \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle {A_P}', (\Psi \otimes {\Psi_P}') \otimes \Psi_P \otimes \Psi_Q\rangle$› *left right*
have $\langle {A_Q}', (\Psi' \otimes \Psi_Q) \otimes \Psi_P\rangle \hookrightarrow_F \langle {A_P}', ((\Psi \otimes \Psi_Q) \otimes {\Psi_P}') \otimes \Psi_P\rangle$
by (*metis AssertionStatEqSym AssertionStatEqTrans AssertionStatEq-def Associativity FrameStatImpTrans associativitySym frameImpNilStatEq frameImpResChainPres*)
moreover note ‹$A_P \sharp* {\Psi_P}'$› ‹$A_P \sharp* \Psi_Q$› ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* \Psi_Q$› ‹$A_P \sharp* P$› ‹$A_P \sharp* M$›
‹$A_P \sharp* {\Psi_P}'$› ‹${A_P}' \sharp* (P \parallel Q)$› ‹${A_Q}' \sharp* (P \parallel Q)$› ‹${A_P}' \sharp* \Psi_Q$› ‹${A_P}' \sharp* M$› ‹${A_Q}' \sharp* M$› ‹$A_P \sharp* {A_P}'$› ‹$A_P \sharp* {A_Q}'$›
ultimately have $(\Psi \otimes \Psi_Q) \otimes {\Psi_P}' \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$
by(*elim cBrMerge*(*2*)) (*simp* | *force*)+
then have *Ptrans*: $(\Psi \otimes {\Psi_P}') \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$
by (*metis associativitySym statEqTransition*)

have *left2*: $\langle {A_Q}', \Psi' \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle {A_Q}', (\Psi' \otimes \Psi_P) \otimes \Psi_Q\rangle$
by (*metis FrameStatEq-def frameIntAssociativity*)

have $\langle A_P{}', (\Psi \otimes \Psi_P{}') \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_P{}', ((\Psi \otimes \Psi_P{}') \otimes \Psi_P) \otimes \Psi_Q\rangle$
by (*metis FrameStatEq-def frameIntAssociativity*)
moreover have $\langle A_P{}', ((\Psi \otimes \Psi_P{}') \otimes \Psi_P) \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_P{}', ((\Psi \otimes \Psi_P) \otimes \Psi_P{}') \otimes \Psi_Q\rangle$
by (*metis FrameStatEq-def associativitySym frameIntComposition*)
ultimately have *right2*: $\langle A_P{}', (\Psi \otimes \Psi_P{}') \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_P{}', ((\Psi \otimes \Psi_P) \otimes \Psi_P{}') \otimes \Psi_Q\rangle$
by (*metis FrameStatImpTrans*)
from $\langle\langle A_Q{}', \Psi' \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_P{}', (\Psi \otimes \Psi_P{}') \otimes \Psi_P \otimes \Psi_Q\rangle\rangle$ *left2 right2*
have $\langle A_Q{}', (\Psi' \otimes \Psi_P) \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_P{}', ((\Psi \otimes \Psi_P) \otimes \Psi_P{}') \otimes \Psi_Q\rangle$
by (*metis FrameStatEq-def FrameStatImpTrans frameIntAssociativity*)
moreover note $\langle A_Q \sharp* \Psi'\rangle$ $\langle A_Q \sharp* \Psi_P\rangle$ $\langle A_Q \sharp* \Psi\rangle$ $\langle A_Q \sharp* \Psi_P\rangle$ $\langle A_Q \sharp* Q\rangle$ $\langle A_Q \sharp* M\rangle$
$\langle A_Q \sharp* \Psi_P{}'\rangle$ $\langle A_P{}' \sharp* (P \parallel Q)\rangle$ $\langle A_Q{}' \sharp* (P \parallel Q)\rangle$ $\langle A_P{}' \sharp* \Psi_P\rangle$ $\langle A_P{}' \sharp* M\rangle$ $\langle A_Q{}' \sharp* M\rangle$ $\langle A_Q \sharp* A_Q{}'\rangle$ $\langle A_Q \sharp* A_P{}'\rangle$
ultimately have $(\Psi \otimes \Psi_P) \otimes \Psi_P{}' \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$
by(*elim cBrMerge*(*6*)) (*simp* | *force*)+
then have *Qtrans*: $(\Psi \otimes \Psi_P{}') \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$
by (*metis associativitySym statEqTransition*)

from *Ptrans* $\langle$*extractFrame* $P = \langle A_P, \Psi_P\rangle\rangle$ *Qtrans* $\langle$*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle\rangle$
$\langle A_P \sharp* \Psi\rangle$ $\langle A_P \sharp* \Psi_P{}'\rangle$ $\langle A_P \sharp* P\rangle$ $\langle A_P \sharp* Q\rangle$ $\langle A_P \sharp* M\rangle$ $\langle A_P \sharp* A_Q\rangle$ $\langle A_Q \sharp* \Psi\rangle$ $\langle A_Q \sharp* \Psi_P{}'\rangle$ $\langle A_Q \sharp* P\rangle$
$\langle A_Q \sharp* Q\rangle$ $\langle A_Q \sharp* M\rangle$
show *?case*
by(*elim BrMerge*) (*simp* | *force*)+
next
case(*cScope* Ψ' P M N P' x A_P Ψ_P $A_P{}'$ $\Psi_P{}'$ A_Q Ψ_Q Ψ)
then have $\Psi \otimes \Psi_P{}' \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$
by *simp*
with $\langle x \sharp \Psi\rangle$ $\langle x \sharp \Psi_P{}'\rangle$ $\langle x \sharp M\rangle$ $\langle x \sharp N\rangle$
show *?case*
by(*elim Scope*) (*simp* | *force*)+
next
case(*cBang* Ψ' P M N P' A_R Ψ_R A_P Ψ_P A_Q Ψ_Q Ψ)
from $\langle A_R \sharp* P\rangle$ **have** $A_R \sharp* (P \parallel !P)$
by *simp*
from $\langle A_P \sharp* !P\rangle$ **have** $A_P \sharp* (P \parallel !P)$
by *simp*
from $\langle A_Q \sharp* !P\rangle$ **have** $A_Q \sharp* (P \parallel !P)$
by *simp*

have $\langle A_Q, \Psi' \otimes \Psi_R \otimes \mathbf{1}\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R \otimes \mathbf{1}\rangle$
proof −
from $\langle \Psi_R \simeq \mathbf{1}\rangle$ **have** $\langle A_Q, \Psi' \otimes \Psi_R \otimes \mathbf{1}\rangle \simeq_F \langle A_Q, \Psi' \otimes \mathbf{1}\rangle$
by(*metis Identity Commutativity AssertionStatEqSym Composition frameResChainPres frameNilStatEq AssertionStatEqTrans*)
moreover note $\langle\langle A_Q, \Psi' \otimes \mathbf{1}\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1}\rangle\rangle$

moreover from ‹$\Psi_R \simeq \mathbf{1}$› **have** $\langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1}\rangle \simeq_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R \otimes \mathbf{1}\rangle$
by(*metis Identity Commutativity AssertionStatEqSym Composition frameResChainPres frameNilStatEq AssertionStatEqTrans*)
ultimately show *?thesis* **by**(*rule FrameStatEqImpCompose*)
qed
then have $\Psi \otimes \Psi_P \rhd P \parallel !P \longmapsto ¿M(\!|N|\!) \prec P'$
using ‹$A_R \sharp* \Psi'$› ‹$A_R \sharp* \Psi$› ‹$A_R \sharp* (P \parallel !P)$› ‹$A_R \sharp* \Psi_P$›
‹$A_P \sharp* (P \parallel !P)$› ‹$A_Q \sharp* (P \parallel !P)$› ‹$A_P \sharp* M$› ‹$A_Q \sharp* M$› ‹$A_R \sharp* M$› ‹$A_R \sharp* A_P$› ‹$A_R \sharp* A_Q$›
by(*elim cBang*(*5*))
then show *?case* **using** ‹*guarded P*›
by(*rule Bang*)
qed

lemma *brCommInAux*:
fixes Ψ :: *'b*
and Ψ_Q :: *'b*
and R :: (*'a*, *'b*, *'c*) *psi*
and M :: *'a*
and N :: *'a*
and R' :: (*'a*, *'b*, *'c*) *psi*
and A_R :: *name list*
and Ψ_R :: *'b*
and A_P :: *name list*
and Ψ_P :: *'b*
and A_Q :: *name list*

assumes *RTrans*: $\Psi \otimes \Psi_Q \rhd R \longmapsto ¿M(\!|N|\!) \prec R'$
and *FrR*: *extractFrame* $R = \langle A_R, \Psi_R\rangle$
and *distinct* A_R
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 $A_R \sharp* A_P$
and $A_R \sharp* A_Q$
and $A_R \sharp* \Psi$
and $A_R \sharp* \Psi_P$
and $A_R \sharp* \Psi_Q$
and $A_R \sharp* R$
and $A_R \sharp* M$
and $A_P \sharp* R$
and $A_P \sharp* M$
and $A_Q \sharp* R$
and $A_Q \sharp* M$

shows $\Psi \otimes \Psi_P \rhd R \longmapsto ¿M(\!|N|\!) \prec R'$
proof −
from ‹$A_R \sharp* \Psi$› ‹$A_R \sharp* \Psi_Q$›
have $A_R \sharp* (\Psi \otimes \Psi_Q)$
by *auto*

with *assms*
show *?thesis*
by(*simp add*: *brCommInAuxTooMuch*)
qed

lemma *brCommOutAuxTooMuch*:
fixes Ψ :: *'b*
and Ψ_Q :: *'b*
and R :: (*'a*, *'b*, *'c*) *psi*
and M :: *'a*
and *xvec* :: *name list*
and N :: *'a*
and R' :: (*'a*, *'b*, *'c*) *psi*
and A_R :: *name list*
and Ψ_R :: *'b*
and A_P :: *name list*
and Ψ_P :: *'b*
and A_Q :: *name list*

assumes *RTrans*: $\Psi \otimes \Psi_Q \rhd R \longmapsto RBrOut\ M\ ((\!|\nu * xvec|\!)N) \prec' R'$
and *FrR*: *extractFrame* $R = \langle A_R, \Psi_R\rangle$
and *distinct* A_R
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 $A_R \sharp* A_P$
and $A_R \sharp* A_Q$
and $A_R \sharp* \Psi$
and $A_R \sharp* \Psi_P$
and $A_R \sharp* \Psi_Q$
and $A_R \sharp* (\Psi \otimes \Psi_Q)$
and $A_R \sharp* R$
and $A_R \sharp* M$
and $A_P \sharp* R$
and $A_P \sharp* M$
and $A_Q \sharp* R$
and $A_Q \sharp* M$

shows $\Psi \otimes \Psi_P \rhd R \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec R'$
using *assms*
proof(*nominal-induct R M B*==$(\!|\nu * xvec|\!)N \prec' R'\ A_R\ \Psi_R$ *avoiding*: $\Psi\ A_P\ \Psi_P\ A_Q\ \Psi_Q\ N\ R'$ *xvec rule*: *broutputFrameInduct*)
case (*cAlpha* $\Psi'\ P\ M\ A_R\ \Psi_R\ p\ \Psi\ A_P\ \Psi_P\ A_Q\ \Psi_Q\ N\ P'$ *xvec*)
have *S*: *set* $p \subseteq$ *set* $A_R \times$ *set* $(p \cdot A_R)$ **by** *fact*
from $\langle\langle A_Q, \Psi' \otimes (p \cdot \Psi_R)\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes (p \cdot \Psi_R)\rangle\rangle$
have $(p \cdot \langle A_Q, \Psi' \otimes (p \cdot \Psi_R)\rangle) \hookrightarrow_F (p \cdot \langle A_P, (\Psi \otimes \Psi_P) \otimes (p \cdot \Psi_R)\rangle)$
by(*rule FrameStatImpClosed*)
with $\langle A_R \sharp* A_P\rangle\ \langle(p \cdot A_R) \sharp* A_P\rangle\ \langle A_R \sharp* \Psi'\rangle\ \langle(p \cdot A_R) \sharp* \Psi'\rangle\ \langle A_R \sharp* \Psi_P\rangle\ \langle(p \cdot A_R) \sharp* \Psi_P\rangle\ \langle A_R \sharp* A_Q\rangle$
$\langle(p \cdot A_R) \sharp* A_Q\rangle\ \langle A_R \sharp* \Psi\rangle\ \langle(p \cdot A_R) \sharp* \Psi\rangle\ S\ \langle distinctPerm\ p\rangle$
have $\langle A_Q, \Psi' \otimes \Psi_R\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R\rangle$ **by**(*simp add*: *eqvts*)

moreover note ‹$A_R \sharp* \Psi'$› ‹$A_R \sharp* \Psi$› ‹$A_R \sharp* P$› ‹$A_R \sharp* \Psi_P$› ‹$A_R \sharp* \Psi_Q$› ‹$A_R \sharp* M$› ‹$A_P \sharp* P$›
‹$A_Q \sharp* P$› ‹$A_P \sharp* M$› ‹$A_Q \sharp* M$› ‹$A_R \sharp* A_P$› ‹$A_R \sharp* A_Q$›
ultimately show *?case*
by(*elim cAlpha*)
next
case(*cBrOutput* Ψ' M K N P Ψ A_P Ψ_P A_Q Ψ_Q N' R' *xvec*)
from ‹$A_P \sharp* (M\langle N\rangle.P)$› ‹$A_Q \sharp* (M\langle N\rangle.P)$›
have $A_P \sharp* M$ **and** $A_Q \sharp* M$ **by** *simp+*
from ‹$\Psi' \vdash M \preceq K$›
have $\Psi' \otimes \mathbf{1} \vdash M \preceq K$
by(*blast intro*: *statEqEnt Identity AssertionStatEqSym*)
with ‹$A_Q \sharp* K$› ‹$A_Q \sharp* M$›
have $(\langle A_Q, \Psi' \otimes \mathbf{1}\rangle) \vdash_F M \preceq K$
by(*force intro*: *frameImpI*)
with ‹$\langle A_Q, \Psi' \otimes \mathbf{1}\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1}\rangle$›
have $(\langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1}\rangle) \vdash_F M \preceq K$
by(*simp add*: *FrameStatImp-def*)
with ‹$A_P \sharp* K$› ‹$A_P \sharp* M$› **have** $(\Psi \otimes \Psi_P) \otimes \mathbf{1} \vdash M \preceq K$
by(*force dest*: *frameImpE*)
then have $\Psi \otimes \Psi_P \vdash M \preceq K$ **by**(*blast intro*: *statEqEnt Identity*)
then have $\Psi \otimes \Psi_P \rhd M\langle N\rangle.P \longmapsto ¡K\langle N\rangle \prec P$
by(*rule BrOutput*)
with ‹$N \prec' P = (\!|\nu* xvec|\!) N' \prec' R'$›
show *?case*
by(*simp add*: *residualInject*)
next
case(*cCase* Ψ' R M φ *Cs* A_R Ψ_R Ψ A_P Ψ_P A_Q Ψ_Q N R' *xvec*)
from ‹$A_P \sharp* (Cases\ Cs)$› ‹$A_Q \sharp* (Cases\ Cs)$› ‹$(\varphi, R) \in set\ Cs$›
have $A_P \sharp* R$ **and** $A_Q \sharp* R$ **and** $A_P \sharp* \varphi$ **and** $A_Q \sharp* \varphi$
by(*auto dest*: *memFreshChain*)

from ‹$\Psi' \vdash \varphi$› **have** $\Psi' \otimes \mathbf{1} \vdash \varphi$ **by**(*blast intro*: *statEqEnt Identity AssertionStatEqSym*)
with ‹$A_Q \sharp* \varphi$› **have** $(\langle A_Q, \Psi' \otimes \mathbf{1}\rangle) \vdash_F \varphi$ **by**(*force intro*: *frameImpI*)
with ‹$\langle A_Q, \Psi' \otimes \mathbf{1}\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1}\rangle$› **have** $(\langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1}\rangle) \vdash_F \varphi$
by(*simp add*: *FrameStatImp-def*)
with ‹$A_P \sharp* \varphi$› **have** $(\Psi \otimes \Psi_P) \otimes \mathbf{1} \vdash \varphi$ **by**(*force dest*: *frameImpE*)
then have $\Psi \otimes \Psi_P \vdash \varphi$ **by**(*blast intro*: *statEqEnt Identity*)

have $\langle A_Q, \Psi' \otimes \Psi_R\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R\rangle$
proof −
from ‹$\Psi_R \simeq \mathbf{1}$› **have** $\langle A_Q, \Psi' \otimes \Psi_R\rangle \simeq_F \langle A_Q, \Psi' \otimes \mathbf{1}\rangle$
by(*metis Identity Commutativity AssertionStatEqSym Composition frameResChainPres frameNilStatEq AssertionStatEqTrans*)
moreover note ‹$\langle A_Q, \Psi' \otimes \mathbf{1}\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1}\rangle$›
moreover from ‹$\Psi_R \simeq \mathbf{1}$› **have** $\langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1}\rangle \simeq_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R\rangle$

by(*metis Identity Commutativity AssertionStatEqSym Composition frameResChainPres frameNilStatEq AssertionStatEqTrans*)
ultimately show *?thesis* **by**(*rule FrameStatEqImpCompose*)
qed
with ‹$A_P \sharp* R$› ‹$A_Q \sharp* R$›
have $\Psi \otimes \Psi_P \rhd R \longmapsto ¡M(\!|\nu* xvec|\!)\langle N\rangle \prec R'$ **using** *cCase*
by(*intro cCase*(*4*)) *simp*+
then show *?case* **using** ‹$(\varphi, R) \in set\ Cs$› ‹$\Psi \otimes \Psi_P \vdash \varphi$› ‹*guarded R*›
by(*rule Case*)
next
case(*cPar1* $\Psi'\ \Psi_Q\ P\ M\ xvec\ N\ P'\ A_Q\ Q\ A_P\ \Psi_P\ \Psi\ A_P'\ \Psi_P'\ A_Q'\ \Psi_Q'\ N'\ R'$ *yvec*)
from ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› ‹$A_Q \sharp* A_P'$› ‹$A_P' \sharp* (P \parallel Q)$› **have** $A_P' \sharp* \Psi_Q$
by(*force dest*: *extractFrameFreshChain*)

have $\langle A_P', (\Psi \otimes \Psi_P') \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_P', (\Psi \otimes \Psi_P') \otimes \Psi_Q \otimes \Psi_P\rangle$
by (*metis Commutativity FrameStatEq-def frameIntCompositionSym*)
moreover have $\langle A_P', (\Psi \otimes \Psi_P') \otimes \Psi_Q \otimes \Psi_P\rangle \hookrightarrow_F \langle A_P', ((\Psi \otimes \Psi_P') \otimes \Psi_Q) \otimes \Psi_P\rangle$
by (*metis FrameStatEq-def frameIntAssociativity*)
moreover have $\langle A_P', ((\Psi \otimes \Psi_P') \otimes \Psi_Q) \otimes \Psi_P\rangle \hookrightarrow_F \langle A_P', ((\Psi \otimes \Psi_Q) \otimes \Psi_P') \otimes \Psi_P\rangle$
by (*metis FrameStatEq-def associativitySym frameIntComposition*)
ultimately have *right*: $\langle A_P', (\Psi \otimes \Psi_P') \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_P', ((\Psi \otimes \Psi_Q) \otimes \Psi_P') \otimes \Psi_P\rangle$
by (*metis FrameStatImpTrans*)
have $\langle A_Q', \Psi' \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_Q', \Psi' \otimes \Psi_Q \otimes \Psi_P\rangle$
by (*metis AssertionStatEq-def Commutativity compositionSym frameImpNilStatEq frameImpResChainPres*)
moreover have $\langle A_Q', \Psi' \otimes \Psi_Q \otimes \Psi_P\rangle \hookrightarrow_F \langle A_Q', (\Psi' \otimes \Psi_Q) \otimes \Psi_P\rangle$
by (*metis FrameStatEq-def frameIntAssociativity*)
ultimately have *left*: $\langle A_Q', \Psi' \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_Q', (\Psi' \otimes \Psi_Q) \otimes \Psi_P\rangle$
by (*metis FrameStatImpTrans*)
from ‹$\langle A_Q', \Psi' \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_P', (\Psi \otimes \Psi_P') \otimes \Psi_P \otimes \Psi_Q\rangle$› *left right*
have $\langle A_Q', (\Psi' \otimes \Psi_Q) \otimes \Psi_P\rangle \hookrightarrow_F \langle A_P', ((\Psi \otimes \Psi_Q) \otimes \Psi_P') \otimes \Psi_P\rangle$
by (*metis AssertionStatEqSym AssertionStatEqTrans AssertionStatEq-def Associativity FrameStatImpTrans associativitySym frameImpNilStatEq frameImpResChainPres*)
moreover note ‹$A_P \sharp* \Psi'$› ‹$A_P \sharp* \Psi_Q$› ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* \Psi_Q$› ‹$A_P \sharp* P$› ‹$A_P \sharp* M$›
‹$A_P \sharp* \Psi_P'$› ‹$A_P' \sharp* (P \parallel Q)$› ‹$A_Q' \sharp* (P \parallel Q)$› ‹$A_P' \sharp* \Psi_Q$› ‹$A_P' \sharp* M$› ‹$A_Q' \sharp* M$› ‹$A_P \sharp* A_P'$› ‹$A_P \sharp* A_Q'$›
ultimately have $(\Psi \otimes \Psi_Q) \otimes \Psi_P' \rhd P \longmapsto ¡M(\!|\nu* xvec|\!)\langle N\rangle \prec P'$
by(*intro cPar1*(*6*)) (*simp* | *force*)+
then have $(\Psi \otimes \Psi_P') \otimes \Psi_Q \rhd P \longmapsto ¡M(\!|\nu* xvec|\!)\langle N\rangle \prec P'$
by (*metis associativitySym statEqTransition*)
moreover note ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› ‹$xvec \sharp* Q$› ‹$A_Q \sharp* \Psi$› ‹$A_Q \sharp* \Psi_P'$›

$\langle A_Q \sharp* P\rangle$ $\langle A_Q \sharp* M\rangle$ $\langle A_Q \sharp* xvec\rangle$ $\langle A_Q \sharp* N\rangle$
ultimately have $\Psi \otimes \Psi_P{}' \rhd P \parallel Q \longmapsto ¡M(\!|\nu* xvec|\!)\langle N\rangle \prec P' \parallel Q$
by(*elim Par1*) (*simp* | *force*)+
then show *?case* **using** $\langle(\!|\nu* xvec|\!)N \prec' (P' \parallel Q) = (\!|\nu* yvec|\!)N' \prec' R'\rangle$
by(*simp add*: *residualInject*)
next
case(*cPar2* Ψ' Ψ_P Q M *xvec* N Q' A_P P A_Q Ψ_Q Ψ $A_P{}'$ $\Psi_P{}'$ $A_Q{}'$ $\Psi_Q{}'$ N' R' *yvec*)
from $\langle$*extractFrame* $P = \langle A_P, \Psi_P\rangle\rangle$ $\langle A_P \sharp* A_P{}'\rangle$ $\langle A_P{}' \sharp* (P \parallel Q)\rangle$ **have** $A_P{}' \sharp* \Psi_P$
by(*force dest*: *extractFrameFreshChain*)
have $\langle A_P{}', (\Psi \otimes \Psi_P{}') \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_P{}', ((\Psi \otimes \Psi_P{}') \otimes \Psi_P) \otimes \Psi_Q\rangle$
by (*metis FrameStatEq-def frameIntAssociativity*)
moreover have $\langle A_P{}', ((\Psi \otimes \Psi_P{}') \otimes \Psi_P) \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_P{}', ((\Psi \otimes \Psi_P) \otimes \Psi_P{}') \otimes \Psi_Q\rangle$
by (*metis FrameStatEq-def associativitySym frameIntComposition*)
ultimately have *right*: $\langle A_P{}', (\Psi \otimes \Psi_P{}') \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_P{}', ((\Psi \otimes \Psi_P) \otimes \Psi_P{}') \otimes \Psi_Q\rangle$
by (*metis FrameStatImpTrans*)
moreover have *left*: $\langle A_Q{}', \Psi' \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_Q{}', (\Psi' \otimes \Psi_P) \otimes \Psi_Q\rangle$
by (*metis FrameStatEq-def frameIntAssociativity*)
from $\langle\langle A_Q{}', \Psi' \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_P{}', (\Psi \otimes \Psi_P{}') \otimes \Psi_P \otimes \Psi_Q\rangle\rangle$ *left right*
have $\langle A_Q{}', (\Psi' \otimes \Psi_P) \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_P{}', ((\Psi \otimes \Psi_P) \otimes \Psi_P{}') \otimes \Psi_Q\rangle$
by (*metis FrameStatEq-def FrameStatImpTrans frameIntAssociativity*)
moreover note $\langle A_Q \sharp* \Psi'\rangle$ $\langle A_Q \sharp* \Psi_P\rangle$ $\langle A_Q \sharp* \Psi\rangle$ $\langle A_Q \sharp* \Psi_P\rangle$ $\langle A_Q \sharp* Q\rangle$ $\langle A_Q \sharp* M\rangle$
$\langle A_Q \sharp* \Psi_P{}'\rangle$ $\langle A_P{}' \sharp* (P \parallel Q)\rangle$ $\langle A_Q{}' \sharp* (P \parallel Q)\rangle$ $\langle A_P{}' \sharp* \Psi_P\rangle$ $\langle A_P{}' \sharp* M\rangle$ $\langle A_Q{}' \sharp* M\rangle$ $\langle A_Q \sharp* A_Q{}'\rangle$ $\langle A_Q \sharp* A_P{}'\rangle$
ultimately have $(\Psi \otimes \Psi_P) \otimes \Psi_P{}' \rhd Q \longmapsto ¡M(\!|\nu* xvec|\!)\langle N\rangle \prec Q'$
by(*intro cPar2*(*6*)) (*simp* | *force*)+
then have $(\Psi \otimes \Psi_P{}') \otimes \Psi_P \rhd Q \longmapsto ¡M(\!|\nu* xvec|\!)\langle N\rangle \prec Q'$
by (*metis associativitySym statEqTransition*)
moreover note $\langle$*extractFrame* $P = \langle A_P, \Psi_P\rangle\rangle$ $\langle xvec \sharp* P\rangle$ $\langle A_P \sharp* \Psi\rangle$ $\langle A_P \sharp* \Psi_P{}'\rangle$
$\langle A_P \sharp* Q\rangle$ $\langle A_P \sharp* M\rangle$ $\langle A_P \sharp* xvec\rangle$ $\langle A_P \sharp* N\rangle$
ultimately have $\Psi \otimes \Psi_P{}' \rhd P \parallel Q \longmapsto ¡M(\!|\nu* xvec|\!)\langle N\rangle \prec P \parallel Q'$
by(*elim Par2*) (*simp* | *force*)+
then show *?case* **using** $\langle(\!|\nu* xvec|\!)N \prec' (P \parallel Q') = (\!|\nu* yvec|\!)N' \prec' R'\rangle$
by(*simp add*: *residualInject*)
next
case(*cBrComm1* Ψ' Ψ_Q P M N P' A_P Ψ_P Q *xvec* Q' A_Q Ψ $A_P{}'$ $\Psi_P{}'$ $A_Q{}'$ $\Psi_Q{}'$ N' R' *yvec*)
have *right*: $\langle A_P{}', (\Psi \otimes \Psi_P{}') \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_P{}', (\Psi_Q \otimes (\Psi \otimes \Psi_P{}')) \otimes \Psi_P\rangle$
by (*metis AssertionStatEqTrans AssertionStatEq-def Associativity Commutativity FrameStatImpTrans frameImpNilStatEq frameImpResChainPres*)
with $\langle\langle A_Q{}', \Psi' \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_P{}', (\Psi \otimes \Psi_P{}') \otimes \Psi_P \otimes \Psi_Q\rangle\rangle$
have $\langle A_Q{}', (\Psi_Q \otimes \Psi') \otimes \Psi_P\rangle \hookrightarrow_F \langle A_P{}', (\Psi_Q \otimes (\Psi \otimes \Psi_P{}')) \otimes \Psi_P\rangle$
by (*metis AssertionStatEqTrans AssertionStatEq-def Associativity Commutativity FrameStatImpTrans frameImpNilStatEq frameImpResChainPres*)

moreover from ‹$\Psi' \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$›
have $\Psi_Q \otimes \Psi' \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$
by (*metis Commutativity statEqTransition*)
moreover note ‹*extractFrame* $P = \langle A_P, \Psi_P\rangle$› ‹*distinct* A_P› ‹$A_P \sharp* A_P'$›
‹$A_P \sharp* A_Q'$› ‹$A_P \sharp* \Psi_Q$› ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* \Psi_P'$› ‹$A_P \sharp* \Psi'$› ‹$A_P \sharp* P$› ‹$A_P \sharp* M$› ‹$A_P' \sharp* (P \parallel Q)$›
‹$A_P' \sharp* M$› ‹$A_Q' \sharp* (P \parallel Q)$› ‹$A_Q' \sharp* M$›
ultimately have $\Psi_Q \otimes (\Psi \otimes \Psi_P') \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$
by(*elim brCommInAux*) (*simp* | *force*)+
then have *Ptrans*: $(\Psi \otimes \Psi_P') \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$
by (*metis Commutativity statEqTransition*)

have *right2*: $\langle A_P', (\Psi \otimes \Psi_P') \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_P', ((\Psi \otimes \Psi_P') \otimes \Psi_P) \otimes \Psi_Q\rangle$
by (*metis FrameStatEq-def frameIntAssociativity*)
with ‹$\langle A_Q', \Psi' \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_P', (\Psi \otimes \Psi_P') \otimes \Psi_P \otimes \Psi_Q\rangle$›
have $\langle A_Q', (\Psi' \otimes \Psi_P) \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_P', ((\Psi \otimes \Psi_P') \otimes \Psi_P) \otimes \Psi_Q\rangle$
by (*metis FrameStatEq-def FrameStatImpTrans frameIntAssociativity*)
then have *Qtrans*: $(\Psi \otimes \Psi_P') \otimes \Psi_P \rhd Q \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec Q'$
using ‹$A_Q \sharp* A_P'$› ‹$A_Q \sharp* A_Q'$› ‹$A_Q \sharp* \Psi$› ‹$A_Q \sharp* \Psi_P'$›
‹$A_Q \sharp* \Psi_P$› ‹$A_Q \sharp* \Psi'$› ‹$A_Q \sharp* \Psi_P$› ‹$A_Q \sharp* Q$› ‹$A_Q \sharp* M$› ‹$A_P' \sharp* (P \parallel Q)$› ‹$A_P' \sharp* M$›
‹$A_Q' \sharp* (P \parallel Q)$› ‹$A_Q' \sharp* M$›
by(*intro cBrComm1*(*8*)) (*simp* | *force*)+
from *Ptrans* ‹*extractFrame* $P = \langle A_P, \Psi_P\rangle$› *Qtrans* ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$›
‹$A_P \sharp* \Psi$› ‹$A_P \sharp* \Psi_P'$› ‹$A_P \sharp* P$› ‹$A_P \sharp* Q$›
‹$A_P \sharp* M$› ‹$A_P \sharp* A_Q$› ‹$A_Q \sharp* \Psi$› ‹$A_Q \sharp* \Psi_P'$› ‹$A_Q \sharp* P$› ‹$A_Q \sharp* Q$› ‹$A_Q \sharp* M$› ‹*xvec* $\sharp* P$›
have $\Psi \otimes \Psi_P' \rhd P \parallel Q \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec P' \parallel Q'$
by(*elim BrComm1*) (*simp* | *force*)+
then show *?case* **using** ‹$(\!|\nu*xvec|\!)N \prec' (P' \parallel Q') = (\!|\nu*yvec|\!)N' \prec' R'$›
by(*simp add*: *residualInject*)
next
case(*cBrComm2* $\Psi'\ \Psi_Q\ P\ M\ xvec\ N\ P'\ A_P\ \Psi_P\ Q\ Q'\ A_Q\ \Psi\ A_P'\ \Psi_P'\ A_Q'\ \Psi_Q'\ N'\ R'\ yvec$)
have $\langle A_P', (\Psi \otimes \Psi_P') \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_P', (\Psi_P \otimes (\Psi \otimes \Psi_P')) \otimes \Psi_Q\rangle$
by (*metis AssertionStatEqTrans AssertionStatEq-def Commutativity associativitySym frameImpNilStatEq frameImpResChainPres*)
with ‹$\langle A_Q', \Psi' \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_P', (\Psi \otimes \Psi_P') \otimes \Psi_P \otimes \Psi_Q\rangle$›
have $\langle A_Q', (\Psi_P \otimes \Psi') \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_P', (\Psi_P \otimes (\Psi \otimes \Psi_P')) \otimes \Psi_Q\rangle$
by (*metis AssertionStatEqTrans AssertionStatEq-def Commutativity FrameStatImpTrans associativitySym frameImpNilStatEq frameImpResChainPres*)
moreover from ‹$\Psi' \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$›
have $\Psi_P \otimes \Psi' \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$
by (*metis Commutativity statEqTransition*)
moreover note ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› ‹*distinct* A_Q›
‹$A_Q \sharp* A_P'$› ‹$A_Q \sharp* A_Q'$› ‹$A_Q \sharp* \Psi_P$› ‹$A_Q \sharp* \Psi$› ‹$A_Q \sharp* \Psi_P'$›
‹$A_Q \sharp* \Psi'$› ‹$A_Q \sharp* Q$› ‹$A_Q \sharp* M$› ‹$A_P' \sharp* (P \parallel Q)$› ‹$A_P' \sharp* M$› ‹$A_Q' \sharp* (P \parallel$

$Q)\rangle$ $\langle A_Q{}' \sharp* M\rangle$
ultimately have $\Psi_P \otimes (\Psi \otimes \Psi_P{}') \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$
by (*elim brCommInAux*) (*simp* | *force*)+
then have *Qtrans*: $(\Psi \otimes \Psi_P{}') \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$
by (*metis Commutativity statEqTransition*)

have $\langle A_P{}', (\Psi \otimes \Psi_P{}') \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_P{}', ((\Psi \otimes \Psi_P{}') \otimes \Psi_Q) \otimes \Psi_P\rangle$
by (*metis AssertionStatEqTrans AssertionStatEq-def Associativity associativitySym frameImpNilStatEq frameImpResChainPres*)
with $\langle\langle A_Q{}', \Psi' \otimes \Psi_P \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_P{}', (\Psi \otimes \Psi_P{}') \otimes \Psi_P \otimes \Psi_Q\rangle\rangle$
have $\langle A_Q{}', (\Psi' \otimes \Psi_Q) \otimes \Psi_P\rangle \hookrightarrow_F \langle A_P{}', ((\Psi \otimes \Psi_P{}') \otimes \Psi_Q) \otimes \Psi_P\rangle$
by (*metis AssertionStatEqTrans AssertionStatEq-def Associativity FrameStatImpTrans associativitySym frameImpNilStatEq frameImpResChainPres*)
with $\langle A_P \sharp* A_P{}'\rangle$ $\langle A_P \sharp* A_Q{}'\rangle$ $\langle A_P \sharp* \Psi\rangle$ $\langle A_P \sharp* \Psi_P{}'\rangle$ $\langle A_P \sharp* \Psi_Q\rangle$
$\langle A_P \sharp* \Psi'\rangle$ $\langle A_P \sharp* \Psi_Q\rangle$ $\langle A_P \sharp* P\rangle$ $\langle A_P \sharp* M\rangle$ $\langle A_P{}' \sharp* (P \parallel Q)\rangle$ $\langle A_P{}' \sharp* M\rangle$
$\langle A_Q{}' \sharp* (P \parallel Q)\rangle$ $\langle A_Q{}' \sharp* M\rangle$
have *Ptrans*: $(\Psi \otimes \Psi_P{}') \otimes \Psi_Q \rhd P \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec P'$
by(*intro cBrComm2*(*4*)) (*simp* | *force*)+
from *Ptrans* $\langle$*extractFrame* $P = \langle A_P, \Psi_P\rangle\rangle$ *Qtrans* $\langle$*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle\rangle$
$\langle A_P \sharp* \Psi\rangle$ $\langle A_P \sharp* \Psi_P{}'\rangle$ $\langle A_P \sharp* P\rangle$ $\langle A_P \sharp* Q\rangle$ $\langle A_P \sharp* M\rangle$ $\langle A_P \sharp* A_Q\rangle$
$\langle A_Q \sharp* \Psi\rangle$ $\langle A_Q \sharp* \Psi_P{}'\rangle$ $\langle A_Q \sharp* P\rangle$ $\langle A_Q \sharp* Q\rangle$ $\langle A_Q \sharp* M\rangle$ $\langle xvec \sharp* Q\rangle$
have $\Psi \otimes \Psi_P{}' \rhd P \parallel Q \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec P' \parallel Q'$
by(*elim BrComm2*) (*simp* | *force*)+
then show *?case* **using** $\langle(\!|\nu*xvec|\!)N \prec' (P' \parallel Q') = (\!|\nu*yvec|\!)N' \prec' R'\rangle$
by(*simp add*: *residualInject*)
next
case(*cBrOpen* Ψ' R M' *xvec yvec* N' R' x A_R Ψ_R Ψ A_P Ψ_P A_Q Ψ_Q N R'' *zvec*)
have $\Psi \otimes \Psi_P \rhd R \longmapsto ¡M'(\!|\nu*(xvec@yvec)|\!)\langle N'\rangle \prec R'$ **using** *cBrOpen*
by(*intro cBrOpen*(*4*)) (*assumption* | *simp*)+

then have $\Psi \otimes \Psi_P \rhd (\!|\nu x|\!)R \longmapsto ¡M'(\!|\nu*(xvec@x\#yvec)|\!)\langle N'\rangle \prec R'$
using $\langle x \in supp\ N'\rangle$ $\langle x \sharp \Psi\rangle$ $\langle x \sharp \Psi_P\rangle$ $\langle x \sharp M'\rangle$ $\langle x \sharp xvec\rangle$ $\langle x \sharp yvec\rangle$
by(*elim BrOpen*) (*simp* | *force*)+
with $\langle(\!|\nu*(xvec\ @\ x\ \#\ yvec)|\!)N' \prec' R' = (\!|\nu*zvec|\!)N \prec' R''\rangle$
show *?case*
by(*simp add*: *residualInject*)
next
case(*cScope* Ψ' R M' *xvec* N' R' x A_R Ψ_R Ψ A_P Ψ_P A_Q Ψ_Q N R'' *yvec*)
then have $\Psi \otimes \Psi_P \rhd R \longmapsto ¡M'(\!|\nu*xvec|\!)\langle N'\rangle \prec R'$
by(*intro cScope*(*4*)) (*simp* | *force*)+
with $\langle x \sharp \Psi\rangle$ $\langle x \sharp \Psi_P\rangle$ $\langle x \sharp M'\rangle$ $\langle x \sharp xvec\rangle$ $\langle x \sharp N'\rangle$
have $\Psi \otimes \Psi_P \rhd (\!|\nu x|\!)R \longmapsto ¡M'(\!|\nu*xvec|\!)\langle N'\rangle \prec (\!|\nu x|\!)R'$
by(*elim Scope*) (*simp* | *force*)+
then show *?case* **using** $\langle(\!|\nu*xvec|\!)N' \prec' (\!|\nu x|\!)R' = (\!|\nu*yvec|\!)N \prec' R''\rangle$
by(*simp add*: *residualInject*)
next
case(*cBang* Ψ' R M' A_R Ψ_R Ψ A_P Ψ_P A_Q Ψ_Q N R' *xvec*)
have $\langle A_Q, \Psi' \otimes \Psi_R \otimes \mathbf{1}\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R \otimes \mathbf{1}\rangle$

proof −
from ‹$\Psi_R \simeq \mathbf{1}$› **have** $\langle A_Q,\ \Psi' \otimes \Psi_R \otimes \mathbf{1}\rangle \simeq_F \langle A_Q, \Psi' \otimes \mathbf{1}\rangle$
by(*metis Identity Commutativity AssertionStatEqSym Composition frameResChainPres frameNilStatEq AssertionStatEqTrans*)
moreover note ‹$\langle A_Q, \Psi' \otimes \mathbf{1}\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1}\rangle$›
moreover from ‹$\Psi_R \simeq \mathbf{1}$› **have** $\langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1}\rangle \simeq_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R \otimes \mathbf{1}\rangle$
by(*metis Identity Commutativity AssertionStatEqSym Composition frameResChainPres frameNilStatEq AssertionStatEqTrans*)
ultimately show *?thesis* **by**(*rule FrameStatEqImpCompose*)
qed
then have $\Psi \otimes \Psi_P \rhd R \parallel\ !R \longmapsto ¡M'(\!|\nu{*}xvec|\!)\langle N\rangle \prec R'$ **using** *cBang*
by(*intro cBang*(*5*)) (*simp* | *force*)+
then show *?case* **using** ‹*guarded R*›
by(*rule Bang*)
qed

lemma *brCommOutAux*:
fixes Ψ :: *'b*
and Ψ_Q :: *'b*
and R :: (*'a*, *'b*, *'c*) *psi*
and M :: *'a*
and *xvec* :: *name list*
and N :: *'a*
and R' :: (*'a*, *'b*, *'c*) *psi*
and A_R :: *name list*
and Ψ_R :: *'b*
and A_P :: *name list*
and Ψ_P :: *'b*
and A_Q :: *name list*

assumes *RTrans*: $\Psi \otimes \Psi_Q \rhd R \longmapsto ¡M(\!|\nu{*}xvec|\!)\langle N\rangle \prec R'$
and *FrR*: *extractFrame* $R = \langle A_R, \Psi_R\rangle$
and *distinct* A_R
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 $A_R \sharp* A_P$
and $A_R \sharp* A_Q$
and $A_R \sharp* \Psi$
and $A_R \sharp* \Psi_P$
and $A_R \sharp* \Psi_Q$
and $A_R \sharp* R$
and $A_R \sharp* M$
and $A_P \sharp* R$
and $A_P \sharp* M$
and $A_Q \sharp* R$
and $A_Q \sharp* M$

shows $\Psi \otimes \Psi_P \rhd R \longmapsto ¡M(\!|\nu{*}xvec|\!)\langle N\rangle \prec R'$
proof −

from *RTrans* **have** $\Psi \otimes \Psi_Q \rhd R \longmapsto RBrOut\ M\ (\langle\!|\nu{*}xvec|\!\rangle N \prec' R')$
by(*simp add*: *residualInject*)
moreover from $\langle A_R \sharp* \Psi\rangle$ $\langle A_R \sharp* \Psi_Q\rangle$ **have** $A_R \sharp* (\Psi \otimes \Psi_Q)$
by *force*
ultimately show *?thesis* **using** *assms*
by(*elim brCommOutAuxTooMuch*)
qed

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*: $\Psi \otimes \Psi_Q \rhd R \longmapsto K\langle\!|\nu{*}xvec|\!\rangle\langle N\rangle \prec R'$
and *FrR*: $extractFrame\ R = \langle A_R, \Psi_R\rangle$
and *PTrans*: $\Psi \otimes \Psi_R \rhd P \longmapsto M\langle\!|L|\!\rangle \prec P'$
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 −
from ‹$A_R \sharp* P$› ‹$A_R \sharp* Q$› ‹$A_R \sharp* A_P$› ‹$A_R \sharp* A_Q$› *FrP FrQ* **have** $A_R \sharp* \Psi_P$ **and** $A_R \sharp* \Psi_Q$
by(*force dest*: *extractFrameFreshChain*)+
assume *Assumptions*: $\bigwedge K'.\ [\![\Psi \otimes \Psi_P \rhd R \longmapsto K'(\!|\nu*xvec|\!)\langle N\rangle \prec R';\ \Psi \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K';\ A_R \sharp* K']\!] \Longrightarrow$ *thesis*
{
fix Ψ' ::$'b$
and *zvec* ::*name list*

assume *A*: $\Psi \otimes \Psi_Q \simeq \Psi'$
assume $\Psi' \rhd R \longmapsto K(\!|\nu*xvec|\!)\langle N\rangle \prec R'$
then have $\Psi' \rhd R \longmapsto ROut\ K\ ((\!|\nu*xvec|\!)N \prec' R')$ **by**(*simp add*: *residualInject*)
moreover note *FrR* ‹*distinct* A_R› *PTrans*
moreover from ‹$\Psi' \rhd R \longmapsto K(\!|\nu*xvec|\!)\langle N\rangle \prec R'$› **have** *distinct xvec* **by**(*auto dest*: *boundOutputDistinct*)
moreover assume $\Psi' \otimes \Psi_R \vdash M \leftrightarrow K$ **and** $\langle A_Q, \Psi' \otimes \Psi_R\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R\rangle$
and $A_R \sharp* zvec$ **and** $A_P \sharp* zvec$ **and** $zvec \sharp* R$ **and** $zvec \sharp* P$
and $A_R \sharp* \Psi'$
ultimately have $\exists K'.\ \Psi \otimes \Psi_P \rhd R \longmapsto ROut\ K'\ ((\!|\nu*xvec|\!)N \prec' R') \wedge \Psi \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K' \wedge zvec \sharp* K' \wedge A_R \sharp* K'$
using *FrP* ‹$A_P \sharp* R$› ‹$A_Q \sharp* R$› ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* P$› ‹*distinct* A_P› ‹$A_R \sharp* \Psi$› ‹$A_R \sharp* P$› ‹$A_R \sharp* R$›
‹$A_P \sharp* M$› ‹$A_Q \sharp* M$› ‹$A_R \sharp* K$› ‹$A_R \sharp* A_P$› ‹$A_R \sharp* A_Q$› ‹$A_R \sharp* \Psi_P$›
proof(*nominal-induct* Ψ' *R K B*==$(\!|\nu*xvec|\!)N \prec' R'$ A_R Ψ_R *avoiding*: Ψ *P* A_P Ψ_P A_Q *zvec xvec N R' arbitrary*: *M rule*: *outputFrameInduct*)
case(*cAlpha* Ψ' *R K* A_R Ψ_R *p* Ψ *P* A_P Ψ_P A_Q *zvec xvec N R' M*)
have *S*: *set p* ⊆ *set* A_R × *set* $(p \cdot A_R)$ **by** *fact*
from ‹$\Psi' \otimes (p \cdot \Psi_R) \vdash M \leftrightarrow K$› **have** $(p \cdot (\Psi' \otimes (p \cdot \Psi_R))) \vdash (p \cdot M) \leftrightarrow (p \cdot K)$
by(*rule chanEqClosed*)
with ‹$A_R \sharp* \Psi'$› ‹$(p \cdot A_R) \sharp* \Psi'$› ‹$A_R \sharp* K$› ‹$(p \cdot A_R) \sharp* K$› *S* ‹*distinctPerm p*›
have $\Psi' \otimes \Psi_R \vdash (p \cdot M) \leftrightarrow K$ **by**(*simp add*: *eqvts*)
moreover from ‹$\Psi \otimes (p \cdot \Psi_R) \rhd P \longmapsto M(\!|L|\!) \prec P'$› *S* ‹$A_R \sharp* P$› ‹$(p \cdot A_R) \sharp* P$› **have** $(p \cdot (\Psi \otimes (p \cdot \Psi_R))) \rhd P \longmapsto (p \cdot M)(\!|L|\!) \prec P'$
by(*elim inputPermFrameSubject*) *auto*
with ‹$A_R \sharp* \Psi$› ‹$(p \cdot A_R) \sharp* \Psi$› *S* ‹*distinctPerm p*› **have** $\Psi \otimes \Psi_R \rhd P \longmapsto (p \cdot M)(\!|L|\!) \prec P'$
by(*simp add*: *eqvts*)
moreover from ‹$A_P \sharp* M$› **have** $(p \cdot A_P) \sharp* (p \cdot M)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$A_R \sharp* A_P$› ‹$(p \cdot A_R) \sharp* A_P$› *S* **have** $A_P \sharp* (p \cdot M)$ **by** *simp*
moreover from ‹$A_Q \sharp* M$› **have** $(p \cdot A_Q) \sharp* (p \cdot M)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$A_R \sharp* A_Q$› ‹$(p \cdot A_R) \sharp* A_Q$› *S* **have** $A_Q \sharp* (p \cdot M)$ **by** *simp*

moreover from ‹$\langle A_Q,\ \Psi' \otimes (p \cdot \Psi_R)\rangle \hookrightarrow_F \langle A_P,\ (\Psi \otimes \Psi_P) \otimes (p \cdot \Psi_R)\rangle$ ›
have $(p \cdot \langle A_Q,\ \Psi' \otimes (p \cdot \Psi_R)\rangle) \hookrightarrow_F (p \cdot \langle A_P,\ (\Psi \otimes \Psi_P) \otimes (p \cdot \Psi_R)\rangle)$
by(*rule FrameStatImpClosed*)
with ‹$A_R \sharp* A_P$› ‹$(p \cdot A_R) \sharp* A_P$› ‹$A_R \sharp* \Psi'$› ‹$(p \cdot A_R) \sharp* \Psi'$› ‹$A_R \sharp* \Psi_P$› ‹$(p \cdot A_R) \sharp* \Psi_P$› ‹$A_R \sharp* A_Q$›
‹$(p \cdot A_R) \sharp* A_Q$› ‹$A_R \sharp* \Psi$› ‹$(p \cdot A_R) \sharp* \Psi$› S ‹*distinctPerm p*›
have $\langle A_Q,\ \Psi' \otimes \Psi_R\rangle \hookrightarrow_F \langle A_P,\ (\Psi \otimes \Psi_P) \otimes \Psi_R\rangle$ **by**(*simp add*: *eqvts*)
ultimately obtain K' **where** $\Psi \otimes \Psi_P \rhd R \longmapsto ROut\ K'\ (\langle\!|\nu * xvec|\!\rangle N \prec' R')$ **and** $\Psi \otimes \Psi_P \otimes \Psi_R \vdash (p \cdot M) \leftrightarrow K'$ **and** $zvec \sharp* K'$ **and** $A_R \sharp* K'$
using *cAlpha*
by *metis*
from ‹$\Psi \otimes \Psi_P \rhd R \longmapsto ROut\ K'\ (\langle\!|\nu * xvec|\!\rangle N \prec' R')$› S ‹$A_R \sharp* R$› ‹$(p \cdot A_R) \sharp* R$›
have $(p \cdot (\Psi \otimes \Psi_P)) \rhd R \longmapsto (ROut\ (p \cdot K')\ (\langle\!|\nu * xvec|\!\rangle N \prec' R'))$ **using** *outputPermFrameSubject*
by(*auto simp add*: *residualInject*)
with S ‹$A_R \sharp* \Psi$› ‹$(p \cdot A_R) \sharp* \Psi$› ‹$A_R \sharp* \Psi_P$› ‹$(p \cdot A_R) \sharp* \Psi_P$› **have** $\Psi \otimes \Psi_P \rhd R \longmapsto (ROut\ (p \cdot K')\ (\langle\!|\nu * xvec|\!\rangle N \prec' R'))$
by(*simp add*: *eqvts*)
moreover from ‹$\Psi \otimes \Psi_P \otimes \Psi_R \vdash (p \cdot M) \leftrightarrow K'$› **have** $(p \cdot (\Psi \otimes \Psi_P \otimes \Psi_R)) \vdash (p \cdot p \cdot M) \leftrightarrow (p \cdot K')$
by(*rule chanEqClosed*)
with S ‹$A_R \sharp* \Psi$› ‹$(p \cdot A_R) \sharp* \Psi$› ‹$A_R \sharp* \Psi_P$› ‹$(p \cdot A_R) \sharp* \Psi_P$› ‹*distinctPerm p*› **have** $\Psi \otimes \Psi_P \otimes (p \cdot \Psi_R) \vdash M \leftrightarrow (p \cdot K')$
by(*simp add*: *eqvts*)
moreover from ‹$zvec \sharp* K'$› **have** $(p \cdot zvec) \sharp* (p \cdot K')$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$A_R \sharp* zvec$› ‹$(p \cdot A_R) \sharp* zvec$› S **have** $zvec \sharp* (p \cdot K')$ **by** *simp*
moreover from ‹$A_R \sharp* K'$› **have** $(p \cdot A_R) \sharp* (p \cdot K')$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
ultimately show *?case* **by** *blast*
next
case(*cOutput* $\Psi'\ M'\ K\ N\ R\ \Psi\ P\ A_P\ \Psi_P\ A_Q\ zvec\ xvec\ N'\ R'\ M$)
from ‹$A_P \sharp* (M'\langle N\rangle.R)$› ‹$A_Q \sharp* (M'\langle N\rangle.R)$› ‹$zvec \sharp* (M'\langle N\rangle.R)$›
have $A_P \sharp* M'$ **and** $A_Q \sharp* M'$ **and** $zvec \sharp* M'$ **by** *simp+*

from ‹$\Psi' \vdash M' \leftrightarrow K$› **have** $\Psi' \otimes \mathbf{1} \vdash M' \leftrightarrow K$ **by**(*blast intro*: *statEqEnt Identity AssertionStatEqSym*)
then have $\Psi' \otimes \mathbf{1} \vdash M' \leftrightarrow M'$ **by**(*blast intro*: *chanEqSym chanEqTrans*)
with ‹$A_Q \sharp* M'$› **have** $(\langle A_Q,\ \Psi' \otimes \mathbf{1}\rangle) \vdash_F M' \leftrightarrow M'$ **by**(*force intro*: *frameImpI*)

with ‹$\langle A_Q,\ \Psi' \otimes \mathbf{1}\rangle \hookrightarrow_F \langle A_P,\ (\Psi \otimes \Psi_P) \otimes \mathbf{1}\rangle$› **have** $(\langle A_P,\ (\Psi \otimes \Psi_P) \otimes \mathbf{1}\rangle) \vdash_F M' \leftrightarrow M'$
by(*simp add*: *FrameStatImp-def*)
with ‹$A_P \sharp* M'$› **have** $(\Psi \otimes \Psi_P) \otimes \mathbf{1} \vdash M' \leftrightarrow M'$ **by**(*force dest*: *frameImpE*)
then have $\Psi \otimes \Psi_P \vdash M' \leftrightarrow M'$ **by**(*blast intro*: *statEqEnt Identity*) **then have** $\Psi \otimes \Psi_P \rhd M'\langle N\rangle.R \longmapsto M'\langle N\rangle \prec R$
by(*rule Output*)

moreover from ‹$\Psi' \otimes \mathbf{1} \vdash M \leftrightarrow K$› ‹$\Psi' \otimes \mathbf{1} \vdash M' \leftrightarrow K$›
have $\Psi' \otimes \mathbf{1} \vdash M \leftrightarrow M'$ **by**(*metis chanEqSym chanEqTrans*)
with ‹$A_Q \sharp* M$› ‹$A_Q \sharp* M'$›
have $(\langle A_Q, \Psi' \otimes \mathbf{1}\rangle) \vdash_F M \leftrightarrow M'$
by(*force intro*: *frameImpI*)
with ‹$\langle A_Q, \Psi' \otimes \mathbf{1}\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1}\rangle$›
have $(\langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1}\rangle) \vdash_F M \leftrightarrow M'$
by(*simp add*: *FrameStatImp-def*)
with ‹$A_P \sharp* M$› ‹$A_P \sharp* M'$› **have** $(\Psi \otimes \Psi_P) \otimes \mathbf{1} \vdash M \leftrightarrow M'$
by(*force dest*: *frameImpE*)
then have $\Psi \otimes \Psi_P \otimes \mathbf{1} \vdash M \leftrightarrow M'$
by(*metis statEqEnt Associativity*)
ultimately show *?case* **using** *cOutput* **by**(*auto simp add*: *residualInject*)
next
case(*cCase* Ψ' R M' φ *Cs* A_R Ψ_R Ψ P A_P Ψ_P A_Q *zvec xvec* N R' M)
from ‹*guarded R*› ‹*extractFrame R* = $\langle A_R, \Psi_R\rangle$› **have** $\Psi_R \simeq \mathbf{1}$
by(*metis guardedStatEq*)
with ‹$\Psi' \otimes \mathbf{1} \vdash M \leftrightarrow M'$› **have** $\Psi' \otimes \Psi_R \vdash M \leftrightarrow M'$
by(*metis Identity Commutativity statEqEnt AssertionStatEqSym Composition*)
moreover have $\langle A_Q, \Psi' \otimes \Psi_R\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R\rangle$
proof −
from ‹$\Psi_R \simeq \mathbf{1}$› **have** $\langle A_Q, \ \Psi' \otimes \Psi_R\rangle \simeq_F \langle A_Q, \Psi' \otimes \mathbf{1}\rangle$
by(*metis Identity Commutativity AssertionStatEqSym Composition frameResChainPres frameNilStatEq AssertionStatEqTrans*)
moreover note ‹$\langle A_Q, \Psi' \otimes \mathbf{1}\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1}\rangle$›
moreover from ‹$\Psi_R \simeq \mathbf{1}$› **have** $\langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1}\rangle \simeq_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R\rangle$
by(*metis Identity Commutativity AssertionStatEqSym Composition frameResChainPres frameNilStatEq AssertionStatEqTrans*)
ultimately show *?thesis* **by**(*rule FrameStatEqImpCompose*)
qed
moreover from ‹$\Psi \otimes \mathbf{1} \rhd P \longmapsto M(\!|L|\!) \prec P'$› ‹$\Psi_R \simeq \mathbf{1}$›
have $\Psi \otimes \Psi_R \ \rhd P \longmapsto M(\!|L|\!) \prec P'$ **by**(*metis statEqTransition Identity Commutativity AssertionStatEqSym Composition*)
moreover from ‹*zvec* $\sharp*$ (*Cases Cs*)› ‹$A_P \sharp*$ (*Cases Cs*)› ‹$A_Q \sharp*$ (*Cases Cs*)› ‹(φ, R) $\in$ *set Cs*›
have $A_P \sharp* R$ **and** $A_Q \sharp* R$ **and** *zvec* $\sharp* R$ **and** $A_P \sharp* \varphi$ **and** $A_Q \sharp* \varphi$
by(*auto dest*: *memFreshChain*)
ultimately have $\exists K'.\ \Psi \otimes \Psi_P \rhd R \longmapsto ROut\ K'\ (\!|\nu*xvec|\!)N \prec' R') \wedge \Psi \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K' \wedge$ *zvec* $\sharp* K' \wedge A_R \sharp* K'$ **using** *cCase*
by(*intro cCase*) (*assumption* | *simp*)+
then obtain K' **where** *RTrans*: $\Psi \otimes \Psi_P \rhd R \longmapsto ROut\ K'\ (\!|\nu*xvec|\!)N \prec' R')$
and *MeqK'*: $\Psi \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K'$ **and** *zvec* $\sharp* K'$ **and** $A_R \sharp* K'$
by *metis*
note *RTrans* ‹(φ, R) $\in$ *set Cs*›
moreover from ‹$\Psi' \vdash \varphi$› **have** $\Psi' \otimes \mathbf{1} \vdash \varphi$ **by**(*blast intro*: *statEqEnt Identity AssertionStatEqSym*)

with ‹A_Q ♯* φ› **have** ($\langle A_Q, \Psi' \otimes \mathbf{1}\rangle$) $\vdash_F \varphi$ **by**(*force intro*: *frameImpI*)
with ‹$\langle A_Q, \Psi' \otimes \mathbf{1}\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1}\rangle$› **have** ($\langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1}\rangle$) $\vdash_F \varphi$
by(*simp add*: *FrameStatImp-def*)
with ‹A_P ♯* φ› **have** $(\Psi \otimes \Psi_P) \otimes \mathbf{1} \vdash \varphi$ **by**(*force dest*: *frameImpE*)
then have $\Psi \otimes \Psi_P \vdash \varphi$ **by**(*blast intro*: *statEqEnt Identity*)
ultimately have $\Psi \otimes \Psi_P \rhd$ *Cases Cs* $\longmapsto$*ROut* K' ($(\!|\nu * xvec|\!)N \prec' R'$) **using** ‹*guarded R*› **by**(*rule Case*)
moreover from *MeqK'* ‹$\Psi_R \simeq \mathbf{1}$› **have** $\Psi \otimes \Psi_P \otimes \mathbf{1} \vdash M \leftrightarrow K'$
by(*metis Identity Commutativity statEqEnt AssertionStatEqSym Composition AssertionStatEqTrans*)
ultimately show *?case* **using** ‹*zvec* ♯* *K'*›
by *auto*
next
case(*cPar1* Ψ' Ψ_{R2} R_1 M' *xvec* N' R_1' A_{R2} R_2 A_{R1} Ψ_{R1} Ψ P A_P Ψ_P A_Q *zvec yvec N R' M*)
have *FrR2*: *extractFrame* $R_2 = \langle A_{R2}, \Psi_{R2}\rangle$ **by** *fact*
from ‹$\Psi' \otimes \Psi_{R1} \otimes \Psi_{R2} \vdash M \leftrightarrow M'$› **have** $(\Psi' \otimes \Psi_{R2}) \otimes \Psi_{R1} \vdash M \leftrightarrow M'$
by(*metis statEqEnt Associativity Composition Commutativity*)
moreover have $\langle A_Q, (\Psi' \otimes \Psi_{R2}) \otimes \Psi_{R1}\rangle \hookrightarrow_F \langle A_P, ((\Psi \otimes \Psi_{R2}) \otimes \Psi_P) \otimes \Psi_{R1}\rangle$
proof −
have $\langle A_Q, (\Psi' \otimes \Psi_{R2}) \otimes \Psi_{R1}\rangle \simeq_F \langle A_Q, \Psi' \otimes \Psi_{R1} \otimes \Psi_{R2}\rangle$
by(*metis Associativity Composition Commutativity AssertionStatEqTrans AssertionStatEqSym frameNilStatEq frameResChainPres*)
moreover note ‹$\langle A_Q, \Psi' \otimes \Psi_{R1} \otimes \Psi_{R2}\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_{R1} \otimes \Psi_{R2}\rangle$›
moreover have $\langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_{R1} \otimes \Psi_{R2}\rangle \simeq_F \langle A_P, ((\Psi \otimes \Psi_{R2}) \otimes \Psi_P) \otimes \Psi_{R1}\rangle$
by(*metis Associativity Composition Commutativity AssertionStatEqTrans AssertionStatEqSym frameNilStatEq frameResChainPres*)
ultimately show *?thesis* **by**(*rule FrameStatEqImpCompose*)
qed
moreover from ‹$\Psi \otimes \Psi_{R1} \otimes \Psi_{R2} \rhd P \longmapsto M(\!|L|\!) \prec P'$› **have** $(\Psi \otimes \Psi_{R2}) \otimes \Psi_{R1} \rhd P \longmapsto M(\!|L|\!) \prec P'$
by(*metis statEqTransition Associativity Composition Commutativity*)
moreover from ‹A_{R1} ♯* A_P› ‹A_{R2} ♯* A_P› ‹A_P ♯* $(R_1 \parallel R_2)$› ‹*extractFrame* $R_1 = \langle A_{R1}, \Psi_{R1}\rangle$› *FrR2* **have** A_P ♯* Ψ_{R1} **and** A_P ♯* Ψ_{R2}
by(*force dest*: *extractFrameFreshChain*)+

moreover from ‹$(\!|\nu * xvec|\!)N' \prec' (R_1' \parallel R_2) = (\!|\nu * yvec|\!)N \prec' R'$› ‹*xvec* ♯* *yvec*›
obtain *p T* **where** $(\!|\nu * xvec|\!)N' \prec' R_1' = (\!|\nu * yvec|\!)N \prec' T$ **and** $R' = T \parallel (p \cdot R_2)$ **and** *set p* ⊆ *set yvec* × *set xvec*
apply(*drule-tac sym*)
by(*elim boundOutputPar1Dest'*) (*assumption* | *simp* | *blast dest*: *sym*)+
ultimately have $\exists K'.\ (\Psi \otimes \Psi_{R2}) \otimes \Psi_P \rhd R_1 \longmapsto$*ROut* K' ($(\!|\nu * yvec|\!)N \prec' T$) $\wedge$ $(\Psi \otimes \Psi_{R2}) \otimes \Psi_P \otimes \Psi_{R1} \vdash M \leftrightarrow K' \wedge (A_{R2}@zvec)$ ♯* $K' \wedge A_{R1}$ ♯* K' **using** *cPar1*

by(*elim cPar1*(*6*)[**where** *ba*=P **and** *bb*=A_P **and** *bd*=A_Q]) *auto*
then obtain K' **where** *RTrans*: $(\Psi \otimes \Psi_{R2}) \otimes \Psi_P \rhd R_1 \longmapsto K'\langle\!(\nu{*}xvec)\!\rangle\langle N'\rangle \prec {R_1}'$
and *MeqK'*: $(\Psi \otimes \Psi_{R2}) \otimes \Psi_P \otimes \Psi_{R1} \vdash M \leftrightarrow K'$ **and** $A_{R2} \sharp* K'$ **and** $A_{R1} \sharp* K'$ **and** *zvec* $\sharp*$ K'
using ‹$\langle\!(\nu{*}xvec)\!\rangle N' \prec' {R_1}' = \langle\!(\nu{*}yvec)\!\rangle N \prec' T$› **by**(*auto simp add*: *residualInject*)

from *RTrans* **have** $(\Psi \otimes \Psi_P) \otimes \Psi_{R2} \rhd R_1 \longmapsto K'\langle\!(\nu{*}xvec)\!\rangle\langle N'\rangle \prec {R_1}'$
by(*metis statEqTransition Associativity Composition Commutativity*)
then have $\Psi \otimes \Psi_P \rhd (R_1 \parallel R_2) \longmapsto K'\langle\!(\nu{*}xvec)\!\rangle\langle N'\rangle \prec ({R_1}' \parallel R_2)$ **using** *FrR2* ‹*xvec* $\sharp*$ R_2› ‹$A_{R2} \sharp* \Psi$› ‹$A_{R2} \sharp* \Psi_P$› ‹$A_{R2} \sharp* K'$› ‹$A_{R2} \sharp* R_1$› ‹$A_{R2} \sharp*$ *xvec*› ‹$A_{R2} \sharp* N'$›
by(*force intro*: *Par1*)
moreover from *MeqK'* **have** $\Psi \otimes \Psi_P \otimes \Psi_{R1} \otimes \Psi_{R2} \vdash M \leftrightarrow K'$
by(*metis statEqEnt Associativity Composition Commutativity*)
ultimately show *?case* **using** ‹*zvec* $\sharp*$ K'› ‹$A_{R1} \sharp* K'$› ‹$A_{R2} \sharp* K'$› ‹$\langle\!(\nu{*}xvec)\!\rangle N' \prec' ({R_1}' \parallel R_2) = \langle\!(\nu{*}yvec)\!\rangle N \prec' R'$›
by(*auto simp add*: *residualInject*)
next
case(*cPar2* Ψ' Ψ_{R1} R_2 M' *xvec* N' ${R_2}'$ A_{R1} R_1 A_{R2} Ψ_{R2} Ψ P A_P Ψ_P A_Q *zvec* *yvec* N R' M)
have *FrR1*: *extractFrame* $R_1 = \langle A_{R1}, \Psi_{R1}\rangle$ **by** *fact*
from ‹$\Psi' \otimes \Psi_{R1} \otimes \Psi_{R2} \vdash M \leftrightarrow M'$› **have** $(\Psi' \otimes \Psi_{R1}) \otimes \Psi_{R2} \vdash M \leftrightarrow M'$
by(*metis statEqEnt Associativity Composition Commutativity*)
moreover have $\langle A_Q, (\Psi' \otimes \Psi_{R1}) \otimes \Psi_{R2}\rangle \hookrightarrow_F \langle A_P, ((\Psi \otimes \Psi_{R1}) \otimes \Psi_P) \otimes \Psi_{R2}\rangle$
proof −
have $\langle A_Q, (\Psi' \otimes \Psi_{R1}) \otimes \Psi_{R2}\rangle \simeq_F \langle A_Q, \Psi' \otimes \Psi_{R1} \otimes \Psi_{R2}\rangle$
by(*metis Associativity Composition Commutativity AssertionStatEqTrans AssertionStatEqSym frameNilStatEq frameResChainPres*)
moreover note ‹$\langle A_Q, \Psi' \otimes \Psi_{R1} \otimes \Psi_{R2}\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_{R1} \otimes \Psi_{R2}\rangle$›
moreover have $\langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_{R1} \otimes \Psi_{R2}\rangle \simeq_F \langle A_P, ((\Psi \otimes \Psi_{R1}) \otimes \Psi_P) \otimes \Psi_{R2}\rangle$
by(*metis Associativity Composition Commutativity AssertionStatEqTrans AssertionStatEqSym frameNilStatEq frameResChainPres*)
ultimately show *?thesis* **by**(*rule FrameStatEqImpCompose*)
qed
moreover from ‹$\Psi \otimes \Psi_{R1} \otimes \Psi_{R2} \rhd P \longmapsto M\langle\!(L)\!\rangle \prec P'$› **have** $(\Psi \otimes \Psi_{R1}) \otimes \Psi_{R2} \rhd P \longmapsto M\langle\!(L)\!\rangle \prec P'$
by(*metis statEqTransition Associativity Composition Commutativity*)
moreover from ‹$A_{R1} \sharp* A_P$› ‹$A_{R2} \sharp* A_P$› ‹$A_P \sharp* (R_1 \parallel R_2)$› *FrR1* ‹*extractFrame* $R_2 = \langle A_{R2}, \Psi_{R2}\rangle$› **have** $A_P \sharp* \Psi_{R1}$ **and** $A_P \sharp* \Psi_{R2}$
by(*force dest*: *extractFrameFreshChain*)+
moreover from ‹$\langle\!(\nu{*}xvec)\!\rangle N' \prec' (R_1 \parallel {R_2}') = \langle\!(\nu{*}yvec)\!\rangle N \prec' R'$› ‹*xvec* $\sharp*$ *yvec*›
obtain p T **where** $\langle\!(\nu{*}xvec)\!\rangle N' \prec' {R_2}' = \langle\!(\nu{*}yvec)\!\rangle N \prec' T$ **and** $R' = (p \cdot R_1) \parallel T$ **and** *set* $p \subseteq$ *set yvec* $\times$ *set xvec*

apply(*drule-tac sym*)
by(*elim boundOutputPar2Dest'*) (*assumption* | *simp* | *blast dest*: *sym*)+
ultimately have $\exists K'.\ (\Psi \otimes \Psi_{R1}) \otimes \Psi_P \rhd R_2 \longmapsto ROut\ K'\ (\langle\!|\nu{*}yvec|\!\rangle N \prec' T) \wedge (\Psi \otimes \Psi_{R1}) \otimes \Psi_P \otimes \Psi_{R2} \vdash M \leftrightarrow K' \wedge (A_{R1}@zvec) \sharp* K' \wedge A_{R2} \sharp* K'$ **using** *cPar2*
by(*elim cPar2*(*6*)) (*assumption* | *simp* | *auto*)+
then obtain K' **where** *RTrans*: $(\Psi \otimes \Psi_{R1}) \otimes \Psi_P \rhd R_2 \longmapsto K'\langle\!|\nu{*}xvec|\!\rangle\langle N'\rangle \prec R_2'$
and *MeqK'*: $(\Psi \otimes \Psi_{R1}) \otimes \Psi_P \otimes \Psi_{R2} \vdash M \leftrightarrow K'$ **and** $A_{R1} \sharp* K'$ **and** $zvec \sharp* K'$ **and** $A_{R2} \sharp* K'$
using $\langle\langle\!|\nu{*}xvec|\!\rangle N' \prec' R_2' = \langle\!|\nu{*}yvec|\!\rangle N \prec' T\rangle$ **by**(*auto simp add*: *residualInject*)

from *RTrans* **have** $(\Psi \otimes \Psi_P) \otimes \Psi_{R1} \rhd R_2 \longmapsto K'\langle\!|\nu{*}xvec|\!\rangle\langle N'\rangle \prec R_2'$
by(*metis statEqTransition Associativity Composition Commutativity*)
then have $\Psi \otimes \Psi_P \rhd (R_1 \parallel R_2) \longmapsto K'\langle\!|\nu{*}xvec|\!\rangle\langle N'\rangle \prec (R_1 \parallel R_2')$ **using** *FrR1* $\langle xvec \sharp* R_1\rangle$ $\langle A_{R1} \sharp* \Psi\rangle$ $\langle A_{R1} \sharp* \Psi_P\rangle$ $\langle A_{R1} \sharp* K'\rangle$ $\langle A_{R1} \sharp* xvec\rangle$ $\langle A_{R1} \sharp* N'\rangle$ $\langle A_{R1} \sharp* R_2\rangle$
by(*force intro*: *Par2*)
moreover from *MeqK'* **have** $\Psi \otimes \Psi_P \otimes \Psi_{R1} \otimes \Psi_{R2} \vdash M \leftrightarrow K'$
by(*metis statEqEnt Associativity Composition Commutativity*)
ultimately show *?case* **using** $\langle zvec \sharp* K'\rangle$ $\langle A_{R1} \sharp* K'\rangle$ $\langle A_{R2} \sharp* K'\rangle$ $\langle\langle\!|\nu{*}xvec|\!\rangle N' \prec' (R_1 \parallel R_2') = \langle\!|\nu{*}yvec|\!\rangle N \prec' R'\rangle$
by(*auto simp add*: *residualInject*)
next
case(*cOpen* $\Psi'\ R\ M'\ xvec\ yvec\ N'\ R'\ x\ A_R\ \Psi_R\ \Psi\ P\ A_P\ \Psi_P\ A_Q\ zvec\ zvec2\ N\ R''\ M$)
from $\langle\langle\!|\nu{*}(xvec\ @\ x\ \#\ yvec)|\!\rangle N' \prec' R' = \langle\!|\nu{*}zvec2|\!\rangle N \prec' R''\rangle$ $\langle x \sharp xvec\rangle$ $\langle x \sharp yvec\rangle$ $\langle x \sharp zvec2\rangle$ $\langle x \sharp R''\rangle$ $\langle x \sharp N\rangle$ $\langle distinct\ zvec2\rangle$
obtain $xvec'\ x'\ yvec'$ **where** A: $\langle\!|\nu{*}(xvec@yvec)|\!\rangle N' \prec' R' = \langle\!|\nu{*}(xvec'@yvec')|\!\rangle([(x, x')] \cdot N) \prec' ([(x, x')] \cdot R'')$
and B: $zvec2 = (xvec'@x'\#yvec')$
by(*elim boundOutputOpenDest*) *auto*
then have $\exists K'.\ \Psi \otimes \Psi_P \rhd R \longmapsto ROut\ K'\ (\langle\!|\nu{*}(xvec'@yvec')|\!\rangle([(x, x')] \cdot N) \prec' ([(x, x')] \cdot R'')) \wedge \Psi \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K' \wedge (zvec) \sharp* K' \wedge A_R \sharp* K'$ **using** *cOpen*
by(*elim cOpen*(*4*)) (*assumption* | *simp*)+
then obtain K' **where** *RTrans*: $\Psi \otimes \Psi_P \rhd R \longmapsto K'\langle\!|\nu{*}(xvec@yvec)|\!\rangle\langle N'\rangle \prec R'$
and *MeqK'*: $\Psi \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K'$ **and** $zvec \sharp* K'$ **and** $A_R \sharp* K'$
using A **by**(*auto simp add*: *residualInject*)
from $\langle A_R \sharp* A_P\rangle$ $\langle A_P \sharp* (\langle\!|\nu x|\!\rangle R)\rangle$ $\langle x \sharp A_P\rangle$ $\langle extractFrame\ R = \langle A_R, \Psi_R\rangle\rangle$ **have** $A_P \sharp* \Psi_R$
by(*force dest*: *extractFrameFreshChain*)+

from $\langle\Psi \otimes \Psi_R \rhd P \longmapsto M\langle\!|L|\!\rangle \prec P'\rangle$ $\langle extractFrame\ P = \langle A_P, \Psi_P\rangle\rangle$ $\langle distinct\ A_P\rangle$ $\langle zvec \sharp* P\rangle$ $\langle A_P \sharp* \Psi_R\rangle$ $\langle x \sharp A_P\rangle$ $\langle A_P \sharp* M\rangle$ $\langle A_P \sharp* P\rangle$ $\langle A_P \sharp* zvec\rangle$ $\langle A_P \sharp* \Psi\rangle$ $\langle A_P \sharp* zvec\rangle$ $\langle A_R \sharp* P\rangle$ $\langle A_R \sharp* A_P\rangle$ $\langle x \sharp A_P\rangle$ $\langle x \sharp P\rangle$

obtain K'' **where** *MeqK''*: $(\Psi \otimes \Psi_R) \otimes \Psi_P \vdash M \leftrightarrow K''$ **and** $A_R \sharp* K''$ **and** $zvec \sharp* K''$ **and** $x \sharp K''$
by(*elim inputObtainPrefix*[**where** B=($x\#A_R@zvec$)]) (*assumption* | *simp* | *force*)+

from *MeqK''* *MeqK'* **have** *KeqK''*: $(\Psi \otimes \Psi_P) \otimes \Psi_R \vdash K' \leftrightarrow K''$
by(*metis statEqEnt Associativity Composition Commutativity chanEqSym chanEqTrans*)
with *RTrans* ‹*extractFrame* $R = \langle A_R, \Psi_R\rangle$› ‹*distinct* A_R› ‹$A_R \sharp* \Psi$› ‹$A_R \sharp* \Psi_P$› ‹$A_R \sharp* K'$› ‹$A_R \sharp* K''$› ‹$A_R \sharp* R$›
have $\Psi \otimes \Psi_P \rhd R \longmapsto K''(\!|\nu*(xvec@yvec)|\!)\langle N'\rangle \prec R'$
by(*elim outputRenameSubject*) (*assumption* | *force*)+
then have $\Psi \otimes \Psi_P \rhd (\!|\nu x|\!)R \longmapsto K''(\!|\nu*(xvec@x\#yvec)|\!)\langle N'\rangle \prec R'$
using ‹$x \sharp \Psi$› ‹$x \sharp \Psi_P$› ‹$x \sharp K''$› ‹$x \sharp xvec$› ‹ $x \sharp yvec$› ‹$x \in supp\ N'$› ‹$xvec \sharp* \Psi$› ‹$xvec \sharp* \Psi_P$› ‹$xvec \sharp* R$› ‹$x \sharp K''$›
by(*elim Open*) (*assumption* | *force*)+
moreover from *MeqK''* **have** $\Psi \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K''$
by(*metis statEqEnt Associativity Composition Commutativity*)
ultimately show *?case* **using** ‹$zvec \sharp* K''$› ‹$x \sharp K''$› ‹$A_R \sharp* K''$› B ‹$(\!|\nu*(xvec\ @\ x\ \#\ yvec)|\!)N' \prec' R' = (\!|\nu*zvec2|\!)N \prec' R''$›
by(*auto simp add*: *residualInject*)
next
case(*cScope* Ψ' R M' *xvec* N' R' x A_R Ψ_R Ψ P A_P Ψ_P A_Q *zvec* *yvec* N R'' M)
from ‹$(\!|\nu*xvec|\!)N' \prec' (\!|\nu x|\!)R' = (\!|\nu*yvec|\!)N \prec' R''$› ‹$x \sharp xvec$› ‹$x \sharp yvec$›
obtain R''' **where** $R'' = (\!|\nu x|\!)R'''$ **and** $(\!|\nu*xvec|\!)N' \prec' R' = (\!|\nu*yvec|\!)N \prec' R'''$
apply(*drule-tac sym*)
by(*metis boundOutputScopeDest*)
then have $\exists K'.\ \Psi \otimes \Psi_P \rhd R \longmapsto ROut\ K'\ ((\!|\nu*yvec|\!)N \prec' R''') \wedge \Psi \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K' \wedge zvec \sharp* K' \wedge A_R \sharp* K'$ **using** *cScope*
by(*elim cScope*(*4*)) (*assumption* | *simp*)+
then obtain K' **where** *RTrans*: $\Psi \otimes \Psi_P \rhd R \longmapsto K'(\!|\nu*xvec|\!)\langle N'\rangle \prec R'$
and *MeqK'*: $\Psi \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K'$ **and** $zvec \sharp* K'$ **and** $A_R \sharp* K'$
using ‹$(\!|\nu*xvec|\!)N' \prec' R' = (\!|\nu*yvec|\!)N \prec' R'''$›
by(*auto simp add*: *residualInject*)
from ‹$A_R \sharp* A_P$› ‹$A_P \sharp* ((\!|\nu x|\!)R)$› ‹$x \sharp A_P$› ‹*extractFrame* $R = \langle A_R, \Psi_R\rangle$› **have** $A_P \sharp* \Psi_R$
by(*force dest*: *extractFrameFreshChain*)+
from ‹$\Psi \otimes \Psi_R \rhd P \longmapsto M(\!|L|\!) \prec P'$› ‹*extractFrame* $P = \langle A_P, \Psi_P\rangle$› ‹*distinct* A_P› ‹$x \sharp P$› ‹$zvec \sharp* P$› ‹$A_P \sharp* \Psi_R$› ‹$x \sharp A_P$› ‹$A_P \sharp* M$› ‹$A_P \sharp* P$› ‹$A_P \sharp* zvec$› ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* zvec$› ‹$A_R \sharp* P$› ‹$A_R \sharp* A_P$›
obtain K'' **where** *MeqK''*: $(\Psi \otimes \Psi_R) \otimes \Psi_P \vdash M \leftrightarrow K''$ **and** $x \sharp K''$ **and** $A_R \sharp* K''$ **and** $zvec \sharp* K''$
by(*elim inputObtainPrefix*[**where** B=($x\#A_R@zvec$)]) (*assumption* | *force*)+

from *MeqK''* *MeqK'* **have** *KeqK''*: $(\Psi \otimes \Psi_P) \otimes \Psi_R \vdash K' \leftrightarrow K''$
by(*metis statEqEnt Associativity Composition Commutativity chanEqSym chanEqTrans*)

with *RTrans* ‹*extractFrame* $R = \langle A_R, \Psi_R\rangle$› ‹*distinct* A_R› ‹$A_R \sharp* \Psi$› ‹$A_R \sharp* \Psi_P$› ‹$A_R \sharp* K'$› ‹$A_R \sharp* K''$› ‹$A_R \sharp* R$›
have $\Psi \otimes \Psi_P \rhd R \longmapsto K''(\!|\nu * xvec|\!)\langle N'\rangle \prec R'$
by(*elim outputRenameSubject*) (*assumption* | *force*)+
then have $\Psi \otimes \Psi_P \rhd (\!|\nu x|\!)R \longmapsto K''(\!|\nu * xvec|\!)\langle N'\rangle \prec (\!|\nu x|\!)R'$ **using** ‹$x \sharp \Psi$› ‹$x \sharp \Psi_P$› ‹$x \sharp K''$› ‹$x \sharp xvec$› ‹$x \sharp N'$› ‹$xvec \sharp* \Psi$› ‹$xvec \sharp* \Psi_P$› ‹$xvec \sharp* R$› ‹$x \sharp K''$›
by(*elim Scope*) (*assumption* | *force*)+
moreover from *MeqK''* **have** $\Psi \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K''$
by(*metis statEqEnt Associativity Composition Commutativity*)
ultimately show *?case* **using** ‹$zvec \sharp* K''$› ‹$x \sharp K''$› ‹$A_R \sharp* K''$› ‹$(\!|\nu * xvec|\!)N' \prec' (\!|\nu x|\!)R' = (\!|\nu * yvec|\!)N \prec' R''$›
by(*auto simp add*: *residualInject*)
next
case(*cBang* Ψ' R M' A_R Ψ_R Ψ P A_P Ψ_P A_Q *zvec xvec* N R' M)
from ‹*guarded* R› ‹*extractFrame* $R = \langle A_R, \Psi_R\rangle$› **have** $\Psi_R \simeq \mathbf{1}$
by(*metis guardedStatEq*)
with ‹$\Psi' \otimes \mathbf{1} \vdash M \leftrightarrow M'$› **have** $\Psi' \otimes \Psi_R \otimes \mathbf{1} \vdash M \leftrightarrow M'$
by(*metis Identity Commutativity statEqEnt AssertionStatEqSym Composition*)
moreover have $\langle A_Q, \Psi' \otimes \Psi_R \otimes \mathbf{1}\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R \otimes \mathbf{1}\rangle$
proof −
from ‹$\Psi_R \simeq \mathbf{1}$› **have** $\langle A_Q,\ \Psi' \otimes \Psi_R \otimes \mathbf{1}\rangle \simeq_F \langle A_Q, \Psi' \otimes \mathbf{1}\rangle$
by(*metis Identity Commutativity AssertionStatEqSym Composition frameResChainPres frameNilStatEq AssertionStatEqTrans*)
moreover note ‹$\langle A_Q, \Psi' \otimes \mathbf{1}\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1}\rangle$›
moreover from ‹$\Psi_R \simeq \mathbf{1}$› **have** $\langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1}\rangle \simeq_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R \otimes \mathbf{1}\rangle$
by(*metis Identity Commutativity AssertionStatEqSym Composition frameResChainPres frameNilStatEq AssertionStatEqTrans*)
ultimately show *?thesis* **by**(*rule FrameStatEqImpCompose*)
qed
moreover from ‹$\Psi \otimes \mathbf{1} \rhd P \longmapsto M(\!|L|\!) \prec P'$› ‹$\Psi_R \simeq \mathbf{1}$›
have $\Psi \otimes \Psi_R \otimes \mathbf{1} \rhd P \longmapsto M(\!|L|\!) \prec P'$ **by**(*metis statEqTransition Identity Commutativity AssertionStatEqSym Composition*)
ultimately have $\exists K'.\ \Psi \otimes \Psi_P \rhd R \parallel\ !R \longmapsto ROut\ K'\ ((\!|\nu * xvec|\!)N \prec' R') \wedge \Psi \otimes \Psi_P \otimes \Psi_R \otimes \mathbf{1} \vdash M \leftrightarrow K' \wedge zvec \sharp* K' \wedge A_R \sharp* K'$ **using** *cBang*
by(*intro cBang*(*5*)) (*assumption* |*simp*)+
then obtain K' **where** *RTrans*: $\Psi \otimes \Psi_P \rhd R \parallel\ !R \longmapsto ROut\ K'\ ((\!|\nu * xvec|\!)N \prec' R')$
and *MeqK'*: $\Psi \otimes \Psi_P \otimes \Psi_R \otimes \mathbf{1} \vdash M \leftrightarrow K'$ **and** $zvec \sharp* K'$ **and** $A_R \sharp* K'$
by *metis*
from *RTrans* ‹*guarded* R› **have** $\Psi \otimes \Psi_P \rhd\ !R \longmapsto ROut\ K'\ ((\!|\nu * xvec|\!)N \prec' R')$ **by**(*rule Bang*)
moreover from *MeqK'* ‹$\Psi_R \simeq \mathbf{1}$› **have** $\Psi \otimes \Psi_P \otimes \mathbf{1} \vdash M \leftrightarrow K'$
by(*metis Identity Commutativity statEqEnt AssertionStatEqSym Composition AssertionStatEqTrans*)
ultimately show *?case* **using** ‹$zvec \sharp* K'$›
by *force*

qed
}
note *Goal* = *this*
have $\Psi \otimes \Psi_Q \simeq \Psi \otimes \Psi_Q$ **by** *simp*
moreover note *RTrans*
moreover from *MeqK* **have** $(\Psi \otimes \Psi_Q) \otimes \Psi_R \vdash M \leftrightarrow K$
by(*metis statEqEnt Associativity Commutativity*)
moreover note *PeqQ* $\langle A_R \sharp* \Psi\rangle$ $\langle A_R \sharp* \Psi_Q\rangle$
ultimately have $\exists K'.\ \Psi \otimes \Psi_P \rhd R \longmapsto ROut\ K'\ ((\!|\nu * xvec|\!)N \prec' R') \wedge \Psi \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K' \wedge ([]::name\ list) \sharp* K' \wedge A_R \sharp* K'$
by(*elim Goal*) (*assumption* | *force simp add*: *residualInject*)+
with *Assumptions* **show** *?thesis*
by(*force simp add*: *residualInject*)
qed

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 −
from ‹$A_R \sharp* P$› ‹$A_R \sharp* Q$› ‹$A_R \sharp* A_P$› ‹$A_R \sharp* A_Q$› *FrP FrQ* **have** $A_R \sharp* \Psi_P$ **and** $A_R \sharp* \Psi_Q$
by(*force dest*: *extractFrameFreshChain*)+
assume *Assumptions*: $\bigwedge K'.\ [\![\Psi \otimes \Psi_P \rhd R \longmapsto K'\langle\!|N|\!\rangle \prec R';\ \Psi \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K';\ A_R \sharp* K']\!] \Longrightarrow thesis$
{
fix $\Psi'::'b$
fix *zvec*::*name list*
assume $A_R \sharp* \Psi'$
assume $A_R \sharp* zvec$
assume $A_P \sharp* zvec$
assume $zvec \sharp* R$
assume $zvec \sharp* P$

assume *A*: $\Psi \otimes \Psi_Q \simeq \Psi'$
with *RTrans* **have** $\Psi' \rhd R \longmapsto K\langle\!|N|\!\rangle \prec R'$
by(*rule statEqTransition*)
moreover note *FrR* ‹*distinct* A_R›
moreover from ‹$\Psi \otimes \Psi_Q \otimes \Psi_R \vdash M \leftrightarrow K$› **have** $(\Psi \otimes \Psi_Q) \otimes \Psi_R \vdash M \leftrightarrow K$
by(*blast intro*: *statEqEnt Associativity AssertionStatEqSym*)
with *A* **have** $\Psi' \otimes \Psi_R \vdash M \leftrightarrow K$ **by**(*rule statEqEnt*[*OF Composition*])
moreover have $\langle A_Q, \Psi' \otimes \Psi_R\rangle \simeq_F \langle A_Q, (\Psi \otimes \Psi_Q) \otimes \Psi_R\rangle$ **using** *A*
by(*blast dest*: *frameIntComposition FrameStatEqTrans FrameStatEqSym*)
with *QimpP* **have** $\langle A_Q, \Psi' \otimes \Psi_R\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R\rangle$
by(*force intro*: *FrameStatEqImpCompose*)
moreover from *PTrans* **have** *distinct xvec* **by**(*auto dest*: *boundOutputDistinct*)
ultimately have $\exists K'.\ \Psi \otimes \Psi_P \rhd R \longmapsto K'\langle\!|N|\!\rangle \prec R' \wedge \Psi \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K' \wedge zvec \sharp* K' \wedge A_R \sharp* K'$
using *PTrans FrP* ‹$A_R \sharp* K$› ‹$A_R \sharp* \Psi'$› ‹$A_R \sharp* \Psi$› ‹$A_R \sharp* P$› ‹$A_R \sharp* R$› ‹$A_R \sharp* \Psi_P$› ‹$A_P \sharp* R$› ‹$A_Q \sharp* R$› ‹$A_P \sharp* \Psi$›
‹$A_P \sharp* P$› ‹$A_P \sharp* M$› ‹$A_P \sharp* zvec$› ‹$A_Q \sharp* M$› ‹$A_R \sharp* zvec$› ‹$zvec \sharp* R$› ‹$zvec \sharp* P$› ‹*distinct* A_P›
‹$A_R \sharp* A_P$› ‹$A_R \sharp* A_Q$› ‹$A_R \sharp* xvec$› ‹$xvec \sharp* M$›
proof(*nominal-induct avoiding*: Ψ P A_P Ψ_P A_Q *zvec xvec arbitrary*: *M rule*: *inputFrameInduct*)
case(*cAlpha* Ψ' *R K N R' A*$_R$ Ψ_R *p* Ψ *P A*$_P$ Ψ_P A_Q *zvec xvec M*)
have *S*: *set p* $\subseteq$ *set* $A_R \times$ *set* $(p \cdot A_R)$ **by** *fact*
from ‹$\Psi' \otimes (p \cdot \Psi_R) \vdash M \leftrightarrow K$› **have** $(p \cdot (\Psi' \otimes (p \cdot \Psi_R))) \vdash (p \cdot M) \leftrightarrow$

$(p \cdot K)$
by(*rule chanEqClosed*)
with ‹$A_R \sharp* \Psi'$› ‹$(p \cdot A_R) \sharp* \Psi'$› ‹$A_R \sharp* K$› ‹$(p \cdot A_R) \sharp* K$› S ‹*distinctPerm* p›
have $\Psi' \otimes \Psi_R \vdash (p \cdot M) \leftrightarrow K$ **by**(*simp add*: *eqvts*)
moreover from ‹$\Psi \otimes (p \cdot \Psi_R) \rhd P \longmapsto M(\!|\nu* xvec|\!)\langle N\rangle \prec P'$› S ‹$A_R \sharp* P$› ‹$(p \cdot A_R) \sharp* P$› ‹$A_R \sharp* xvec$› ‹$(p \cdot A_R) \sharp* xvec$› ‹$xvec \sharp* M$›
have $(p \cdot (\Psi \otimes (p \cdot \Psi_R))) \rhd P \longmapsto (p \cdot M)(\!|\nu* xvec|\!)\langle N\rangle \prec P'$
using *outputPermFrameSubject* **by**(*auto simp add*: *residualInject*)
with ‹$A_R \sharp* \Psi$› ‹$(p \cdot A_R) \sharp* \Psi$› S ‹*distinctPerm* p› **have** $\Psi \otimes \Psi_R \rhd P \longmapsto (p \cdot M)(\!|\nu* xvec|\!)\langle N\rangle \prec P'$
by(*simp add*: *eqvts*)
moreover from ‹$A_P \sharp* M$› **have** $(p \cdot A_P) \sharp* (p \cdot M)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$A_R \sharp* A_P$› ‹$(p \cdot A_R) \sharp* A_P$› S **have** $A_P \sharp* (p \cdot M)$ **by** *simp*
moreover from ‹$A_Q \sharp* M$› **have** $(p \cdot A_Q) \sharp* (p \cdot M)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$A_R \sharp* A_Q$› ‹$(p \cdot A_R) \sharp* A_Q$› S **have** $A_Q \sharp* (p \cdot M)$ **by** *simp*

moreover from ‹$\langle A_Q, \Psi' \otimes (p \cdot \Psi_R)\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes (p \cdot \Psi_R)\rangle$›
have $(p \cdot \langle A_Q, \Psi' \otimes (p \cdot \Psi_R)\rangle) \hookrightarrow_F (p \cdot \langle A_P, (\Psi \otimes \Psi_P) \otimes (p \cdot \Psi_R)\rangle)$
by(*rule FrameStatImpClosed*)
with ‹$A_R \sharp* A_P$› ‹$(p \cdot A_R) \sharp* A_P$› ‹$A_R \sharp* \Psi'$› ‹$(p \cdot A_R) \sharp* \Psi'$› ‹$A_R \sharp* \Psi_P$› ‹$(p \cdot A_R) \sharp* \Psi_P$› ‹$A_R \sharp* A_Q$›
‹$(p \cdot A_R) \sharp* A_Q$› ‹$A_R \sharp* \Psi$› ‹$(p \cdot A_R) \sharp* \Psi$› S ‹*distinctPerm* p›
have $\langle A_Q, \Psi' \otimes \Psi_R\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R\rangle$ **by**(*simp add*: *eqvts*)
moreover from ‹$xvec \sharp* M$› **have** $(p \cdot xvec) \sharp* (p \cdot M)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with S ‹$A_R \sharp* xvec$› ‹$(p \cdot A_R) \sharp* xvec$› **have** $xvec \sharp* (p \cdot M)$ **by** *simp*
ultimately obtain K' **where** $\Psi \otimes \Psi_P \rhd R \longmapsto K'(\!|N|\!) \prec R'$ **and** $\Psi \otimes \Psi_P \otimes \Psi_R \vdash (p \cdot M) \leftrightarrow K'$ **and** $zvec \sharp* K'$ **and** $A_R \sharp* K'$
using *cAlpha*
by(*auto simp del*: *freshChainSimps*)
from ‹$\Psi \otimes \Psi_P \rhd R \longmapsto K'(\!|N|\!) \prec R'$› S ‹$A_R \sharp* R$› ‹$(p \cdot A_R) \sharp* R$› **have** $(p \cdot (\Psi \otimes \Psi_P)) \rhd R \longmapsto (p \cdot K')(\!|N|\!) \prec R'$
by(*elim inputPermFrameSubject*) *auto*
with S ‹$A_R \sharp* \Psi$› ‹$(p \cdot A_R) \sharp* \Psi$› ‹$A_R \sharp* \Psi_P$› ‹$(p \cdot A_R) \sharp* \Psi_P$› **have** $\Psi \otimes \Psi_P \rhd R \longmapsto (p \cdot K')(\!|N|\!) \prec R'$
by(*simp add*: *eqvts*)
moreover from ‹$\Psi \otimes \Psi_P \otimes \Psi_R \vdash (p \cdot M) \leftrightarrow K'$› **have** $(p \cdot (\Psi \otimes \Psi_P \otimes \Psi_R)) \vdash (p \cdot p \cdot M) \leftrightarrow (p \cdot K')$
by(*rule chanEqClosed*)
with S ‹$A_R \sharp* \Psi$› ‹$(p \cdot A_R) \sharp* \Psi$› ‹$A_R \sharp* \Psi_P$› ‹$(p \cdot A_R) \sharp* \Psi_P$› ‹*distinctPerm* p› **have** $\Psi \otimes \Psi_P \otimes (p \cdot \Psi_R) \vdash M \leftrightarrow (p \cdot K')$
by(*simp add*: *eqvts*)
moreover from ‹$zvec \sharp* K'$› **have** $(p \cdot zvec) \sharp* (p \cdot K')$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$A_R \sharp* zvec$› ‹$(p \cdot A_R) \sharp* zvec$› S **have** $zvec \sharp* (p \cdot K')$ **by** *simp*
moreover from ‹$A_R \sharp* K'$› **have** $(p \cdot A_R) \sharp* (p \cdot K')$

by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
ultimately show *?case* **by** *blast*
next
case(*cInput* Ψ' M' K *xvec* N *Tvec* R Ψ P A_P Ψ_P A_Q *zvec yvec* M)
from ‹A_P ♯* (M'⦇λ*xvec N⦈.R)› ‹A_Q ♯* (M'⦇λ*xvec N⦈.R)› ‹*zvec* ♯* (M'⦇λ*xvec N⦈.R)›
have A_P ♯* M' **and** A_Q ♯* M' **and** *zvec* ♯* M' **by** *simp*+

from ‹$\Psi' \vdash M' \leftrightarrow K$›
have $\Psi' \otimes \mathbf{1} \vdash M' \leftrightarrow K$
by(*blast intro*: *statEqEnt Identity AssertionStatEqSym*)
then have $\Psi' \otimes \mathbf{1} \vdash M' \leftrightarrow M'$
by(*blast intro*: *chanEqSym chanEqTrans*)
with ‹A_Q ♯* M'›
have $(\langle A_Q, \Psi' \otimes \mathbf{1}\rangle) \vdash_F M' \leftrightarrow M'$
by(*force intro*: *frameImpI*)

with ‹$\langle A_Q, \Psi' \otimes \mathbf{1}\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1}\rangle$›
have $(\langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1}\rangle) \vdash_F M' \leftrightarrow M'$
by(*simp add*: *FrameStatImp-def*)
with ‹A_P ♯* M'› **have** $(\Psi \otimes \Psi_P) \otimes \mathbf{1} \vdash M' \leftrightarrow M'$
by(*force dest*: *frameImpE*)
then have $\Psi \otimes \Psi_P \vdash M' \leftrightarrow M'$ **by**(*blast intro*: *statEqEnt Identity*)
then have $\Psi \otimes \Psi_P \rhd M'$⦇λ*xvec N⦈.$R \longmapsto M'$⦇$(N[xvec::=Tvec])$⦈ $\prec R[xvec::=Tvec]$
using ‹*distinct xvec*› ‹*set xvec* ⊆ *supp* N› ‹*length xvec* = *length Tvec*›
by(*rule Input*)

moreover from ‹$\Psi' \otimes \mathbf{1} \vdash M \leftrightarrow K$› ‹$\Psi' \otimes \mathbf{1} \vdash M' \leftrightarrow K$›
have $\Psi' \otimes \mathbf{1} \vdash M \leftrightarrow M'$ **by**(*metis chanEqSym chanEqTrans*)
with ‹A_Q ♯* M› ‹A_Q ♯* M'›
have $(\langle A_Q, \Psi' \otimes \mathbf{1}\rangle) \vdash_F M \leftrightarrow M'$
by(*force intro*: *frameImpI*)
with ‹$\langle A_Q, \Psi' \otimes \mathbf{1}\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1}\rangle$ ›
have $(\langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1}\rangle) \vdash_F M \leftrightarrow M'$
by(*simp add*: *FrameStatImp-def*)
with ‹A_P ♯* M› ‹A_P ♯* M'› **have** $(\Psi \otimes \Psi_P) \otimes \mathbf{1} \vdash M \leftrightarrow M'$
by(*force dest*: *frameImpE*)
then have $\Psi \otimes \Psi_P \otimes \mathbf{1} \vdash M \leftrightarrow M'$
by(*metis statEqEnt Associativity*)
ultimately show *?case* **using** ‹*zvec* ♯* M'›
by *force*
next
case(*cCase* Ψ' R M' N R' φ *Cs* A_R Ψ_R Ψ P A_P Ψ_P A_Q *zvec xvec* M)
from ‹*guarded* R› ‹*extractFrame* $R = \langle A_R, \Psi_R\rangle$› **have** $\Psi_R \simeq \mathbf{1}$
by(*metis guardedStatEq*)
with ‹$\Psi' \otimes \mathbf{1} \vdash M \leftrightarrow M'$› **have** $\Psi' \otimes \Psi_R \vdash M \leftrightarrow M'$
by(*metis Identity Commutativity statEqEnt AssertionStatEqSym Composition*)
moreover have $\langle A_Q, \Psi' \otimes \Psi_R\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R\rangle$

proof −
from ‹$\Psi_R \simeq \mathbf{1}$› **have** $\langle A_Q,\ \Psi' \otimes \Psi_R\rangle \simeq_F \langle A_Q, \Psi' \otimes \mathbf{1}\rangle$
by(*metis Identity Commutativity AssertionStatEqSym Composition frameResChainPres frameNilStatEq AssertionStatEqTrans*)
moreover note ‹$\langle A_Q, \Psi' \otimes \mathbf{1}\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1}\rangle$›
moreover from ‹$\Psi_R \simeq \mathbf{1}$› **have** $\langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1}\rangle \simeq_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R\rangle$
by(*metis Identity Commutativity AssertionStatEqSym Composition frameResChainPres frameNilStatEq AssertionStatEqTrans*)
ultimately show *?thesis* **by**(*rule FrameStatEqImpCompose*)
qed
moreover from ‹$\Psi \otimes \mathbf{1} \rhd P \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$› ‹$\Psi_R \simeq \mathbf{1}$›
have $\Psi \otimes \Psi_R \rhd P \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$ **by**(*metis statEqTransition Identity Commutativity AssertionStatEqSym Composition*)
moreover from ‹*zvec* ♯* (*Cases Cs*)› ‹A_P ♯* (*Cases Cs*)› ‹A_Q ♯* (*Cases Cs*)› ‹(φ, R) ∈ *set Cs*›
have A_P ♯* R **and** A_Q ♯* R **and** *zvec* ♯* R **and** A_P ♯* φ **and** A_Q ♯* φ
by(*auto dest*: *memFreshChain*)
ultimately have $\exists K'.\ \Psi \otimes \Psi_P \rhd R \longmapsto K'(\!|N|\!) \prec R' \wedge \Psi \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K' \wedge$ *zvec* ♯* $K' \wedge A_R$ ♯* K' **using** *cCase*
by(*elim cCase*) (*assumption* |*simp*)+
then obtain K' **where** *RTrans*: $\Psi \otimes \Psi_P \rhd R \longmapsto K'(\!|N|\!) \prec R'$
and *MeqK'*: $\Psi \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K'$ **and** *zvec* ♯* K' **and** A_R ♯* K'
by *metis*
note *RTrans* ‹(φ, R) ∈ *set Cs*›
moreover from ‹$\Psi' \vdash \varphi$› **have** $\Psi' \otimes \mathbf{1} \vdash \varphi$ **by**(*blast intro*: *statEqEnt Identity AssertionStatEqSym*)
with ‹A_Q ♯* φ› **have** $(\langle A_Q, \Psi' \otimes \mathbf{1}\rangle) \vdash_F \varphi$ **by**(*force intro*: *frameImpI*)
with ‹$\langle A_Q, \Psi' \otimes \mathbf{1}\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1}\rangle$› **have** $(\langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1}\rangle) \vdash_F \varphi$
by(*simp add*: *FrameStatImp-def*)
with ‹A_P ♯* φ› **have** $(\Psi \otimes \Psi_P) \otimes \mathbf{1} \vdash \varphi$ **by**(*force dest*: *frameImpE*)
then have $\Psi \otimes \Psi_P \vdash \varphi$ **by**(*blast intro*: *statEqEnt Identity*)
ultimately have $\Psi \otimes \Psi_P \rhd$ *Cases Cs* $\longmapsto K'(\!|N|\!) \prec R'$ **using** ‹*guarded R*› **by**(*rule Case*)
moreover from *MeqK'* ‹$\Psi_R \simeq \mathbf{1}$› **have** $\Psi \otimes \Psi_P \otimes \mathbf{1} \vdash M \leftrightarrow K'$
by(*metis Identity Commutativity statEqEnt AssertionStatEqSym Composition AssertionStatEqTrans*)
ultimately show *?case* **using** ‹*zvec* ♯* K'›
by *force*
next
case(*cPar1* Ψ' Ψ_{R2} R_1 M' N R_1' A_{R2} R_2 A_{R1} Ψ_{R1} Ψ P A_P Ψ_P A_Q *zvec xvec* M)
have *FrR2*: *extractFrame* $R_2 = \langle A_{R2}, \Psi_{R2}\rangle$ **by** *fact*
from ‹$\Psi' \otimes \Psi_{R1} \otimes \Psi_{R2} \vdash M \leftrightarrow M'$› **have** $(\Psi' \otimes \Psi_{R2}) \otimes \Psi_{R1} \vdash M \leftrightarrow M'$
by(*metis statEqEnt Associativity Composition Commutativity*)
moreover have $\langle A_Q, (\Psi' \otimes \Psi_{R2}) \otimes \Psi_{R1}\rangle \hookrightarrow_F \langle A_P, ((\Psi \otimes \Psi_{R2}) \otimes \Psi_P) \otimes \Psi_{R1}\rangle$
proof −

have $\langle A_Q,\, (\Psi' \otimes \Psi_{R2}) \otimes \Psi_{R1}\rangle \simeq_F \langle A_Q,\, \Psi' \otimes \Psi_{R1} \otimes \Psi_{R2}\rangle$
by(*metis Associativity Composition Commutativity AssertionStatEqTrans AssertionStatEqSym frameNilStatEq frameResChainPres*)
moreover note $\langle\langle A_Q,\, \Psi' \otimes \Psi_{R1} \otimes \Psi_{R2}\rangle \hookrightarrow_F \langle A_P,\, (\Psi \otimes \Psi_P) \otimes \Psi_{R1} \otimes \Psi_{R2}\rangle\rangle$
moreover have $\langle A_P,\, (\Psi \otimes \Psi_P) \otimes \Psi_{R1} \otimes \Psi_{R2}\rangle \simeq_F \langle A_P,\, ((\Psi \otimes \Psi_{R2}) \otimes \Psi_P) \otimes \Psi_{R1}\rangle$
by(*metis Associativity Composition Commutativity AssertionStatEqTrans AssertionStatEqSym frameNilStatEq frameResChainPres*)
ultimately show *?thesis* **by**(*rule FrameStatEqImpCompose*)
qed
moreover from $\langle \Psi \otimes \Psi_{R1} \otimes \Psi_{R2} \rhd P \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P'\rangle$ **have** $(\Psi \otimes \Psi_{R2}) \otimes \Psi_{R1} \rhd P \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$
by(*metis statEqTransition Associativity Composition Commutativity*)
moreover from $\langle A_{R1} \sharp* A_P\rangle$ $\langle A_{R2} \sharp* A_P\rangle$ $\langle A_P \sharp* (R_1 \parallel R_2)\rangle$ $\langle$*extractFrame* $R_1 = \langle A_{R1},\, \Psi_{R1}\rangle\rangle$ *FrR2* **have** $A_P \sharp* \Psi_{R1}$ **and** $A_P \sharp* \Psi_{R2}$
by(*force dest*: *extractFrameFreshChain*)+
moreover note $\langle$*distinct xvec*$\rangle$

ultimately have $\exists K'.\ (\Psi \otimes \Psi_{R2}) \otimes \Psi_P \rhd R_1 \longmapsto K'(\!|N|\!) \prec {R_1}' \wedge (\Psi \otimes \Psi_{R2}) \otimes \Psi_P \otimes \Psi_{R1} \vdash M \leftrightarrow K' \wedge (A_{R2}@zvec) \sharp* K' \wedge A_{R1} \sharp* K'$ **using** *cPar1*
by(*elim cPar1*(*6*)[**where** *ba*=P **and** *bb*=A_P **and** *bd*=A_Q **and** *bf*=*xvec*]) *auto*
then obtain K' **where** *RTrans*: $(\Psi \otimes \Psi_{R2}) \otimes \Psi_P \rhd R_1 \longmapsto K'(\!|N|\!) \prec {R_1}'$ **and** *MeqK'*: $(\Psi \otimes \Psi_{R2}) \otimes \Psi_P \otimes \Psi_{R1} \vdash M \leftrightarrow K'$ **and** $A_{R2} \sharp* K'$ **and** *zvec* $\sharp* K'$ **and** $A_{R1} \sharp* K'$
by *force*

from *RTrans* **have** $(\Psi \otimes \Psi_P) \otimes \Psi_{R2} \rhd R_1 \longmapsto K'(\!|N|\!) \prec {R_1}'$
by(*metis statEqTransition Associativity Composition Commutativity*)
then have $\Psi \otimes \Psi_P \rhd (R_1 \parallel R_2) \longmapsto K'(\!|N|\!) \prec ({R_1}' \parallel R_2)$ **using** *FrR2* $\langle A_{R2} \sharp* \Psi\rangle$ $\langle A_{R2} \sharp* \Psi_P\rangle$ $\langle A_{R2} \sharp* K'\rangle$ $\langle A_{R2} \sharp* R_1\rangle$ $\langle A_{R2} \sharp* N\rangle$
by(*force intro*: *Par1*)
moreover from *MeqK'* **have** $\Psi \otimes \Psi_P \otimes \Psi_{R1} \otimes \Psi_{R2} \vdash M \leftrightarrow K'$
by(*metis statEqEnt Associativity Composition Commutativity*)
ultimately show *?case* **using** $\langle$*zvec* $\sharp* K'\rangle$ $\langle A_{R1} \sharp* K'\rangle$ $\langle A_{R2} \sharp* K'\rangle$
by *force*
next
case(*cPar2* Ψ' Ψ_{R1} R_2 M' N ${R_2}'$ A_{R1} R_1 A_{R2} Ψ_{R2} Ψ P A_P Ψ_P A_Q *zvec* *xvec* M)
have *FrR1*: *extractFrame* $R_1 = \langle A_{R1},\, \Psi_{R1}\rangle$ **by** *fact*
from $\langle \Psi' \otimes \Psi_{R1} \otimes \Psi_{R2} \vdash M \leftrightarrow M'\rangle$ **have** $(\Psi' \otimes \Psi_{R1}) \otimes \Psi_{R2} \vdash M \leftrightarrow M'$
by(*metis statEqEnt Associativity Composition Commutativity*)
moreover have $\langle A_Q,\, (\Psi' \otimes \Psi_{R1}) \otimes \Psi_{R2}\rangle \hookrightarrow_F \langle A_P,\, ((\Psi \otimes \Psi_{R1}) \otimes \Psi_P) \otimes \Psi_{R2}\rangle$
proof −
have $\langle A_Q,\, (\Psi' \otimes \Psi_{R1}) \otimes \Psi_{R2}\rangle \simeq_F \langle A_Q,\, \Psi' \otimes \Psi_{R1} \otimes \Psi_{R2}\rangle$
by(*metis Associativity Composition Commutativity AssertionStatEqTrans AssertionStatEqSym frameNilStatEq frameResChainPres*)

moreover note $\langle\langle A_Q,\ \Psi' \otimes \Psi_{R1} \otimes \Psi_{R2}\rangle \hookrightarrow_F \langle A_P,\ (\Psi \otimes \Psi_P) \otimes \Psi_{R1} \otimes \Psi_{R2}\rangle\rangle$

moreover have $\langle A_P,\ (\Psi \otimes \Psi_P) \otimes \Psi_{R1} \otimes \Psi_{R2}\rangle \simeq_F \langle A_P,\ ((\Psi \otimes \Psi_{R1}) \otimes \Psi_P) \otimes \Psi_{R2}\rangle$

by(*metis Associativity Composition Commutativity AssertionStatEqTrans AssertionStatEqSym frameNilStatEq frameResChainPres*)

ultimately show *?thesis* **by**(*rule FrameStatEqImpCompose*)

qed

moreover from $\langle\Psi \otimes \Psi_{R1} \otimes \Psi_{R2} \rhd P \longmapsto M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P'\rangle$ **have** $(\Psi \otimes \Psi_{R1}) \otimes \Psi_{R2} \rhd P \longmapsto M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P'$

by(*metis statEqTransition Associativity Composition Commutativity*)

moreover from $\langle A_{R1} \sharp{*} A_P\rangle$ $\langle A_{R2} \sharp{*} A_P\rangle$ $\langle A_P \sharp{*} (R_1 \parallel R_2)\rangle$ *FrR1* $\langle$*extractFrame* $R_2 = \langle A_{R2},\ \Psi_{R2}\rangle\rangle$ **have** $A_P \sharp{*} \Psi_{R1}$ **and** $A_P \sharp{*} \Psi_{R2}$

by(*force dest*: *extractFrameFreshChain*)+

ultimately have $\exists K'.\ (\Psi \otimes \Psi_{R1}) \otimes \Psi_P \rhd R_2 \longmapsto K'(\!|N|\!) \prec {R_2}' \wedge (\Psi \otimes \Psi_{R1}) \otimes \Psi_P \otimes \Psi_{R2} \vdash M \leftrightarrow K' \wedge (A_{R1}@zvec) \sharp{*} K' \wedge A_{R2} \sharp{*} K'$ **using** $\langle$*distinct xvec*$\rangle$ *cPar2*

by(*elim cPar2*(*6*)[**where** *ba*=P **and** *bb*=A_P **and** *bd*=A_Q **and** *bf*=*xvec*]) *auto*

then obtain K' **where** *RTrans*: $(\Psi \otimes \Psi_{R1}) \otimes \Psi_P \rhd R_2 \longmapsto K'(\!|N|\!) \prec {R_2}'$ **and** *MeqK'*: $(\Psi \otimes \Psi_{R1}) \otimes \Psi_P \otimes \Psi_{R2} \vdash M \leftrightarrow K'$ **and** $A_{R1} \sharp{*} K'$ **and** *zvec* $\sharp{*}$ K' **and** $A_{R2} \sharp{*} K'$

by *force*

from *RTrans* **have** $(\Psi \otimes \Psi_P) \otimes \Psi_{R1} \rhd R_2 \longmapsto K'(\!|N|\!) \prec {R_2}'$

by(*metis statEqTransition Associativity Composition Commutativity*)

then have $\Psi \otimes \Psi_P \rhd (R_1 \parallel R_2) \longmapsto K'(\!|N|\!) \prec (R_1 \parallel {R_2}')$ **using** *FrR1* $\langle A_{R1} \sharp{*} \Psi\rangle$ $\langle A_{R1} \sharp{*} \Psi_P\rangle$ $\langle A_{R1} \sharp{*} K'\rangle$ $\langle A_{R1} \sharp{*} R_2\rangle$ $\langle A_{R1} \sharp{*} N\rangle$

by(*force intro*: *Par2*)

moreover from *MeqK'* **have** $\Psi \otimes \Psi_P \otimes \Psi_{R1} \otimes \Psi_{R2} \vdash M \leftrightarrow K'$

by(*metis statEqEnt Associativity Composition Commutativity*)

ultimately show *?case* **using** $\langle$*zvec* $\sharp{*}$ $K'\rangle$ $\langle A_{R1} \sharp{*} K'\rangle$ $\langle A_{R2} \sharp{*} K'\rangle$

by *force*

next

case(*cScope* Ψ' R M' N R' x A_R Ψ_R Ψ P A_P Ψ_P A_Q *zvec xvec* M)

then have $\exists K'.\ \Psi \otimes \Psi_P \rhd R \longmapsto K'(\!|N|\!) \prec R' \wedge \Psi \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K' \wedge zvec \sharp{*} K' \wedge A_R \sharp{*} K'$

by(*elim cScope*(*4*)) (*assumption* | *simp del*: *freshChainSimps*)+

then obtain K' **where** *RTrans*: $\Psi \otimes \Psi_P \rhd R \longmapsto K'(\!|N|\!) \prec R'$ **and** *MeqK'*: $\Psi \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K'$ **and** *zvec* $\sharp{*}$ K' **and** $A_R \sharp{*} K'$

by *metis*

from $\langle A_R \sharp{*} A_P\rangle$ $\langle A_P \sharp{*} ((\!|\nu x|\!)R)\rangle$ $\langle x \sharp A_P\rangle$ $\langle$*extractFrame* $R = \langle A_R,\ \Psi_R\rangle\rangle$ **have** $A_P \sharp{*} \Psi_R$

by(*force dest*: *extractFrameFreshChain*)+

from $\langle\Psi \otimes \Psi_R \rhd P \longmapsto M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P'\rangle$ **have** $\Psi \otimes \Psi_R \rhd P \longmapsto \mathit{ROut}\ M\ ((\!|\nu{*}xvec|\!)N \prec' P')$ **by**(*simp add*: *residualInject*)

with $\langle$*extractFrame* $P = \langle A_P,\ \Psi_P\rangle\rangle$ $\langle$*distinct* $A_P\rangle$ $\langle x \sharp P\rangle$ $\langle$*zvec* $\sharp{*}$ $P\rangle$ $\langle A_P \sharp{*} \Psi_R\rangle$ $\langle x \sharp A_P\rangle$ $\langle A_P \sharp{*} M\rangle$ $\langle A_P \sharp{*} P\rangle$ $\langle A_P \sharp{*} zvec\rangle$ $\langle A_P \sharp{*} \Psi\rangle$ $\langle A_P \sharp{*} zvec\rangle$ $\langle A_R$

$\sharp*\ P\rangle$ ‹$A_R\ \sharp*\ A_P$› ‹*xvec* $\sharp*\ M$› ‹*distinct xvec*›
obtain K'' **where** *MeqK''*: $(\Psi \otimes \Psi_R) \otimes \Psi_P \vdash M \leftrightarrow K''$ **and** $x \sharp K''$ **and** $A_R \sharp* K''$ **and** *zvec* $\sharp* K''$
by(*elim outputObtainPrefix*[**where** B=($x\#A_R$@*zvec*)]) (*assumption* | *simp* | *force* | *metis freshChainSym*)+

from *MeqK''* *MeqK'* **have** *KeqK''*: $(\Psi \otimes \Psi_P) \otimes \Psi_R \vdash K' \leftrightarrow K''$
by(*metis statEqEnt Associativity Composition Commutativity chanEqSym chanEqTrans*)
with *RTrans* ‹*extractFrame* $R = \langle A_R, \Psi_R\rangle$› ‹*distinct* A_R› ‹$A_R \sharp* \Psi$› ‹$A_R \sharp* \Psi_P$› ‹$A_R \sharp* K'$› ‹$A_R \sharp* K''$› ‹$A_R \sharp* R$›
have $\Psi \otimes \Psi_P \rhd R \longmapsto K''(\!|N|\!) \prec R'$
by(*elim inputRenameSubject*) (*assumption* | *force*)+
then have $\Psi \otimes \Psi_P \rhd (\!|\nu x|\!)R \longmapsto K''(\!|N|\!) \prec (\!|\nu x|\!)R'$ **using** ‹$x \sharp \Psi$› ‹$x \sharp \Psi_P$› ‹$x \sharp K''$› ‹$x \sharp N$›
by(*elim Scope*) (*assumption* | *force*)+
moreover from *MeqK''* **have** $\Psi \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K''$
by(*metis statEqEnt Associativity Composition Commutativity*)
ultimately show *?case* **using** ‹*zvec* $\sharp* K''$› ‹$x \sharp K''$› ‹$A_R \sharp* K''$›
by *force*
next
case(*cBang* Ψ' R M' N R' A_R Ψ_R Ψ P A_P Ψ_P A_Q *zvec xvec* M)
from ‹*guarded* R› ‹*extractFrame* $R = \langle A_R, \Psi_R\rangle$› **have** $\Psi_R \simeq \mathbf{1}$
by(*metis guardedStatEq*)
with ‹$\Psi' \otimes \mathbf{1} \vdash M \leftrightarrow M'$› **have** $\Psi' \otimes \Psi_R \otimes \mathbf{1} \vdash M \leftrightarrow M'$
by(*metis Identity Commutativity statEqEnt AssertionStatEqSym Composition*)
moreover have $\langle A_Q, \Psi' \otimes \Psi_R \otimes \mathbf{1}\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R \otimes \mathbf{1}\rangle$
proof −
from ‹$\Psi_R \simeq \mathbf{1}$› **have** $\langle A_Q, \Psi' \otimes \Psi_R \otimes \mathbf{1}\rangle \simeq_F \langle A_Q, \Psi' \otimes \mathbf{1}\rangle$
by(*metis Identity Commutativity AssertionStatEqSym Composition frameResChainPres frameNilStatEq AssertionStatEqTrans*)
moreover note ‹$\langle A_Q, \Psi' \otimes \mathbf{1}\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1}\rangle$›
moreover from ‹$\Psi_R \simeq \mathbf{1}$› **have** $\langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1}\rangle \simeq_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R \otimes \mathbf{1}\rangle$
by(*metis Identity Commutativity AssertionStatEqSym Composition frameResChainPres frameNilStatEq AssertionStatEqTrans*)
ultimately show *?thesis* **by**(*rule FrameStatEqImpCompose*)
qed
moreover from ‹$\Psi \otimes \mathbf{1} \rhd P \longmapsto M(\!|\nu*xvec|\!)\langle N\rangle \prec P'$› ‹$\Psi_R \simeq \mathbf{1}$›
have $\Psi \otimes \Psi_R \otimes \mathbf{1} \rhd P \longmapsto M(\!|\nu*xvec|\!)\langle N\rangle \prec P'$ **by**(*metis statEqTransition Identity Commutativity AssertionStatEqSym Composition*)
ultimately have $\exists K'.\ \Psi \otimes \Psi_P \rhd R \parallel\ !R \longmapsto K'(\!|N|\!) \prec R' \wedge \Psi \otimes \Psi_P \otimes \Psi_R \otimes \mathbf{1} \vdash M \leftrightarrow K' \wedge$ *zvec* $\sharp* K' \wedge A_R \sharp* K'$ **using** *cBang*
by(*elim cBang*(*5*)) (*assumption* | *simp del*: *freshChainSimps*)+
then obtain K' **where** *RTrans*: $\Psi \otimes \Psi_P \rhd R \parallel\ !R \longmapsto K'(\!|N|\!) \prec R'$
and *MeqK'*: $\Psi \otimes \Psi_P \otimes \Psi_R \otimes \mathbf{1} \vdash M \leftrightarrow K'$ **and** *zvec* $\sharp* K'$ **and** $A_R \sharp* K'$
by *metis*
from *RTrans* ‹*guarded* R› **have** $\Psi \otimes \Psi_P \rhd\ !R \longmapsto K'(\!|N|\!) \prec R'$ **by**(*rule Bang*)

moreover from *MeqK'* ‹$\Psi_R \simeq \mathbf{1}$› **have** $\Psi \otimes \Psi_P \otimes \mathbf{1} \vdash M \leftrightarrow K'$
by(*metis Identity Commutativity statEqEnt AssertionStatEqSym Composition AssertionStatEqTrans*)
ultimately show *?case* **using** ‹*zvec* $\sharp*$ K'›
by *force*
qed
}
note *Goal* = *this*
have $\Psi \otimes \Psi_Q \simeq \Psi \otimes \Psi_Q$ **by**(*simp add*: *AssertionStatEqRefl*)
moreover from ‹$A_R \sharp* \Psi$› ‹$A_R \sharp* \Psi_Q$› **have** $A_R \sharp* (\Psi \otimes \Psi_Q)$ **by** *force*
ultimately have $\exists K'.\ \Psi \otimes \Psi_P \rhd R \longmapsto K'\langle\!|N|\!\rangle \prec R' \wedge \Psi \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K' \wedge ([]::name\ list) \sharp* K' \wedge A_R \sharp* K'$
by(*intro Goal*) (*assumption* | *force*)+
with *Assumptions* **show** *?thesis*
by *blast*
qed

end

end
theory *Simulation*
imports *Semantics*
begin

This file is a (heavily modified) variant of the theory *Psi_Calculi.Simulation* from [1].

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 :: $'d::fs\text{-}name$

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 −
{
fix $\alpha\ Q'$
assume $\Psi \rhd Q \longmapsto \alpha \prec Q'$ **and** $bn\ \alpha\ \sharp*\ \Psi$ **and** $bn\ \alpha\ \sharp*\ P$
then have $\exists\, P'.\ \Psi \rhd P \longmapsto \alpha \prec P' \wedge (\Psi,\ P',\ Q') \in Rel$
proof(*nominal-induct* α *rule*: *action.strong-induct*)
case(*In M N*)
then show *?case* **by**(*auto simp add*: *Sim*)
next
case(*BrIn M N*)
then show *?case* **by**(*auto simp add*: *Sim*)
next
case(*Out M xvec N*)
moreover {
fix *M xvec N Q'*
assume (*xvec*::*name list*) $\sharp*\ \Psi$ **and** $xvec\ \sharp*\ P$
obtain *p* **where** *xvecFreshPsi*: $((p::name\ prm) \cdot (xvec::name\ list))\ \sharp*\ \Psi$
and *xvecFreshM*: $(p \cdot xvec)\ \sharp*\ (M::'a)$
and *xvecFreshN*: $(p \cdot xvec)\ \sharp*\ (N::'a)$
and *xvecFreshP*: $(p \cdot xvec)\ \sharp*\ P$
and *xvecFreshQ*: $(p \cdot xvec)\ \sharp*\ Q$
and *xvecFreshQ'*: $(p \cdot xvec)\ \sharp*\ (Q'::('a,\ 'b,\ 'c)\ psi)$
and *xvecFreshC*: $(p \cdot xvec)\ \sharp*\ C$
and *xvecFreshxvec*: $(p \cdot xvec)\ \sharp*\ xvec$
and *S*: $(set\ p) \subseteq (set\ xvec) \times (set(p \cdot xvec))$
and *dpr*: *distinctPerm p*
by(*rule name-list-avoiding*[**where** *xvec*=*xvec* **and** *c*=(Ψ, *M*, *Q*, *N*, *P*, *Q'*, *xvec*, *C*)]) *auto*
from ‹$(p \cdot xvec)\ \sharp*\ M$› ‹*distinctPerm p*› **have** $xvec\ \sharp*\ (p \cdot M)$
by(*subst pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*, **where** *pi*=*p*, *symmetric*]) *simp*

assume *Trans*: $\Psi \rhd Q \longmapsto M(\!|\nu*xvec|\!)\langle N\rangle \prec Q'$
with *xvecFreshN xvecFreshQ' S*
have $\Psi \rhd Q \longmapsto M(\!|\nu*(p \cdot xvec)|\!)\langle(p \cdot N)\rangle \prec (p \cdot Q')$
by(*simp add*: *boundOutputChainAlpha'' residualInject*)
moreover then have *distinct*$(p \cdot xvec)$ **by**(*auto dest*: *boundOutputDistinct*)

moreover note *xvecFreshPsi xvecFreshP xvecFreshQ xvecFreshM xvecFreshC*
ultimately obtain *P'* **where** *PTrans*: $\Psi \rhd P \longmapsto M(\!|\nu*(p \cdot xvec)|\!)\langle(p \cdot N)\rangle \prec P'$
and *P'RelQ'*: $(\Psi,\ P',\ p \cdot Q') \in Rel$
using *Sim* **by** (*metis bn.simps*(*3*) *optionFreshChain*(*1*) *subject.simps*(*3*))
then have $(p \cdot \Psi) \rhd (p \cdot P) \longmapsto (p \cdot (M(\!|\nu*(p \cdot xvec)|\!)\langle(p \cdot N)\rangle \prec P'))$

```
      by(simp add: semantics.eqvt)
    with ‹xvec ♯* Ψ› xvecFreshPsi ‹xvec ♯* P› xvecFreshP S dpr
    have Ψ ▷ P ⟼(p • M)⦇ν*xvec⦈⟨N⟩ ≺ (p • P')
      by(simp add: eqvts name-set-fresh-fresh)
    with ‹xvec ♯* Ψ› xvecFreshPsi ‹xvec ♯* P› xvecFreshP S ‹xvec ♯* (p • M)›
    have Ψ ▷ P ⟼(p • p • M)⦇ν*xvec⦈⟨N⟩ ≺ (p • P')
      by(simp add: outputPermSubject)

    with dpr have Ψ ▷ P ⟼M⦇ν*xvec⦈⟨N⟩ ≺ (p • P')
      by simp

    moreover from P'RelQ' Eqvt have (p • Ψ, p • P', p • p • Q') ∈ Rel
      apply(simp add: eqvt-def eqvts)
      apply(erule ballE[where x=(Ψ, P', p • Q')])
       apply(erule allE[where x=p])
      by(auto simp add: eqvts)

    with ‹xvec ♯* Ψ› xvecFreshPsi S dpr have (Ψ, p • P', Q') ∈ Rel
      by simp
     ultimately have ∃ P'. Ψ ▷ P ⟼ M⦇ν*xvec⦈⟨N⟩ ≺ P' ∧ (Ψ, P', Q') ∈
Rel
      by blast
  }
  ultimately show ?case by force
 next
  case(BrOut M xvec N)
  moreover {
   fix M xvec N Q'
   assume (xvec::name list) ♯* Ψ and xvec ♯* P
   obtain p where xvecFreshPsi: ((p::name prm) • (xvec::name list)) ♯* Ψ
   and xvecFreshM: (p • xvec) ♯* (M::'a)
   and xvecFreshN: (p • xvec) ♯* (N::'a)
   and xvecFreshP: (p • xvec) ♯* P
   and xvecFreshQ: (p • xvec) ♯* Q
   and xvecFreshQ': (p • xvec) ♯* (Q'::('a, 'b, 'c) psi)
   and xvecFreshC: (p • xvec) ♯* C
   and xvecFreshxvec: (p • xvec) ♯* xvec
   and S: (set p) ⊆ (set xvec) × (set(p • xvec))
   and dpr: distinctPerm p
    by(rule name-list-avoiding[where xvec=xvec and c=(Ψ, M, Q, N, P, Q',
xvec, C)]) auto

   from ‹(p • xvec) ♯* M› ‹distinctPerm p› have xvec ♯* (p • M)
    by(subst pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst, where pi=p,
symmetric]) simp

   assume Trans: Ψ ▷ Q ⟼¡M⦇ν*xvec⦈⟨N⟩ ≺ Q'
   with xvecFreshN xvecFreshQ' S
   have Ψ ▷ Q ⟼¡M⦇ν*(p • xvec)⦈⟨(p • N)⟩ ≺ (p • Q')
```

```
      by(simp add: boundOutputChainAlpha'' residualInject)
    moreover then have distinct(p • xvec)
      by(auto dest: boundOutputDistinct)

  moreover note xvecFreshPsi xvecFreshP xvecFreshQ xvecFreshM xvecFreshC
   ultimately obtain P' where PTrans: Ψ ▷ P ⟼¡M⦇ν*(p • xvec)⦈⟨(p •
N)⟩ ≺ P'
    and P'RelQ': (Ψ, P', p • Q') ∈ Rel
      by(auto dest: Sim)
    then have (p • Ψ) ▷ (p • P) ⟼(p • (¡M⦇ν*(p • xvec)⦈⟨(p • N)⟩ ≺ P'))
      by(metis semantics.eqvt)
    with ‹xvec ♯* Ψ› xvecFreshPsi ‹xvec ♯* P› xvecFreshP S dpr
    have Ψ ▷ P ⟼¡(p • M)⦇ν*xvec⦈⟨N⟩ ≺ (p • P')
      by(simp add: eqvts name-set-fresh-fresh)
    with ‹xvec ♯* Ψ› xvecFreshPsi ‹xvec ♯* P› xvecFreshP S ‹xvec ♯* (p • M)›
    have Ψ ▷ P ⟼¡(p • p • M)⦇ν*xvec⦈⟨N⟩ ≺ (p • P')
      by(simp add: broutputPermSubject)

    with dpr have Ψ ▷ P ⟼¡M⦇ν*xvec⦈⟨N⟩ ≺ (p • P')
      by simp

    moreover from P'RelQ' Eqvt have (p • Ψ, p • P', p • p • Q') ∈ Rel
      apply(simp add: eqvt-def eqvts)
      apply(erule ballE[where x=(Ψ, P', p • Q')])
       apply(erule allE[where x=p])
      by(auto simp add: eqvts)

    with ‹xvec ♯* Ψ› xvecFreshPsi S dpr have (Ψ, p • P', Q') ∈ Rel
      by simp
     ultimately have ∃ P'. Ψ ▷ P ⟼ ¡M⦇ν*xvec⦈⟨N⟩ ≺ P' ∧ (Ψ, P', Q') ∈
Rel
      by blast
  }
  ultimately show ?case by force
 next
  case Tau
  then show ?case by (auto simp add: Sim)
 qed
}
then show ?thesis
 by (auto simp add: simulation-def)
qed

lemma simI2[case-names cSim]:
 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 *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 $\Psi \rhd P \rightsquigarrow[Rel]\ Q$
using *assms*
by(*auto simp add*: *simulation-def dest*: *boundOutputDistinct*)

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* $\sharp*\ \Psi$
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 $\Psi \rhd P \rightsquigarrow[Rel]\ Q$
using ‹*eqvt Rel*›
proof(*induct rule*: *simI*[**where** C=(*xvec*, C)])
case(*cSim* $\alpha\ Q'$)
from ‹$bn\ \alpha \sharp* (xvec,\ C)$› **have** $bn\ \alpha \sharp* xvec$ **and** $bn\ \alpha \sharp* C$ **by** *simp*+
obtain p::*name prm* **where** $(p \cdot xvec) \sharp* \Psi$ **and** $(p \cdot xvec) \sharp* P$ **and** $(p \cdot xvec) \sharp* Q$
and $(p \cdot xvec) \sharp* \alpha$ **and** S: *set* $p \subseteq$ *set xvec* $\times$ *set*$(p \cdot xvec)$
and *distinctPerm p*
by(*rule name-list-avoiding*[**where** c=(Ψ, P, Q, α) **and** *xvec*=*xvec*]) *auto*
show *?case*
proof(*cases rule*: *actionCases*[**where** α=α])
case(*cInput M N*)
from ‹$\Psi \rhd Q \longmapsto\alpha \prec Q'$› ‹$\alpha$=$M(\!|N|\!)$› **have** $(p \cdot \Psi) \rhd (p \cdot Q) \longmapsto (p \cdot (M(\!|N|\!) \prec Q'))$
by(*auto intro*: *semantics.eqvt*)
with ‹*xvec* $\sharp*\ \Psi$› ‹$(p \cdot xvec) \sharp*\ \Psi$› ‹*xvec* $\sharp*\ Q$› ‹$(p \cdot xvec) \sharp*\ Q$› *S*
have *QTrans*: $\Psi \rhd Q \longmapsto (p \cdot M)(\!|(p \cdot N)|\!) \prec (p \cdot Q')$
by(*simp add*: *eqvts*)
moreover from ‹$(p \cdot xvec) \sharp* \alpha$›
have $(p \cdot (p \cdot xvec)) \sharp* (p \cdot \alpha)$
by(*simp only*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹*distinctPerm p*› ‹α=$M(\!|N|\!)$›
have *xvec* $\sharp*\ (p \cdot M)$ **and** *xvec* $\sharp*\ (p \cdot N)$
by *simp*+
moreover with *QTrans* ‹*xvec* $\sharp*\ Q$›
have *xvec* $\sharp*\ (p \cdot Q')$
by(*metis inputFreshChainDerivative*)

ultimately have $\exists P'.\ \Psi \rhd P \longmapsto (p \cdot M)\langle\!|(p \cdot N)|\!\rangle \prec P' \wedge (\Psi, P', (p \cdot Q')) \in Rel$
by(*simp add*: *Sim*)
then obtain P' **where** *PTrans*: $\Psi \rhd P \longmapsto (p \cdot M)\langle\!|(p \cdot N)|\!\rangle \prec P'$ **and** $P'RelQ'$: $(\Psi, P', (p \cdot Q')) \in Rel$
by *blast*
from *PTrans* **have** $(p \cdot \Psi) \rhd (p \cdot P) \longmapsto (p \cdot ((p \cdot M)\langle\!|(p \cdot N)|\!\rangle \prec P'))$
by(*rule semantics.eqvt*)
with ‹*xvec* $\sharp*$ Ψ› ‹$(p \cdot xvec)$ $\sharp*$ Ψ› ‹*xvec* $\sharp*$ P› ‹$(p \cdot xvec)$ $\sharp*$ P› S ‹*distinctPerm* p›
have $\Psi \rhd P \longmapsto M\langle\!|N|\!\rangle \prec (p \cdot P')$ **by**(*simp add*: *eqvts*)
moreover from $P'RelQ'$ ‹*eqvt Rel*› **have** $(p \cdot \Psi, p \cdot P', p \cdot p \cdot Q') \in Rel$
by(*auto simp add*: *eqvt-def*)
with ‹*xvec* $\sharp*$ Ψ› ‹$(p \cdot xvec)$ $\sharp*$ Ψ› S ‹*distinctPerm* p›
have $(\Psi, p \cdot P', Q') \in Rel$ **by** *simp*
ultimately show *?thesis* **using** ‹α=$M\langle\!|N|\!\rangle$› **by** *blast*
next
case(*cBrInput M N*)
from ‹$\Psi \rhd Q \longmapsto \alpha \prec Q'$› ‹$\alpha$=¿$M\langle\!|N|\!\rangle$› **have** $(p \cdot \Psi) \rhd (p \cdot Q) \longmapsto (p \cdot (¿M\langle\!|N|\!\rangle \prec Q'))$
by(*auto intro*: *semantics.eqvt*)
with ‹*xvec* $\sharp*$ Ψ› ‹$(p \cdot xvec)$ $\sharp*$ Ψ› ‹*xvec* $\sharp*$ Q› ‹$(p \cdot xvec)$ $\sharp*$ Q› S
have *QTrans*: $\Psi \rhd Q \longmapsto ¿(p \cdot M)\langle\!|(p \cdot N)|\!\rangle \prec (p \cdot Q')$
by(*simp add*: *eqvts*)
moreover from ‹$(p \cdot xvec)$ $\sharp*$ α› **have** $(p \cdot (p \cdot xvec))$ $\sharp*$ $(p \cdot \alpha)$
by(*simp only*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹*distinctPerm* p› ‹α=¿$M\langle\!|N|\!\rangle$› **have** *xvec* $\sharp*$ $(p \cdot M)$ **and** *xvec* $\sharp*$ $(p \cdot N)$ **by** *simp+*
moreover with *QTrans* ‹*xvec* $\sharp*$ Q› **have** *xvec* $\sharp*$ $(p \cdot Q')$ **by**(*metis brinput-FreshChainDerivative*)
ultimately have $\exists P'.\ \Psi \rhd P \longmapsto ¿(p \cdot M)\langle\!|(p \cdot N)|\!\rangle \prec P' \wedge (\Psi, P', (p \cdot Q')) \in Rel$
by(*simp add*: *Sim*)
then obtain P' **where** *PTrans*: $\Psi \rhd P \longmapsto ¿(p \cdot M)\langle\!|(p \cdot N)|\!\rangle \prec P'$ **and** $P'RelQ'$: $(\Psi, P', (p \cdot Q')) \in Rel$
by *blast*
from *PTrans* **have** $(p \cdot \Psi) \rhd (p \cdot P) \longmapsto (p \cdot (¿(p \cdot M)\langle\!|(p \cdot N)|\!\rangle \prec P'))$
by(*rule semantics.eqvt*)
with ‹*xvec* $\sharp*$ Ψ› ‹$(p \cdot xvec)$ $\sharp*$ Ψ› ‹*xvec* $\sharp*$ P› ‹$(p \cdot xvec)$ $\sharp*$ P› S ‹*distinctPerm* p›
have $\Psi \rhd P \longmapsto ¿M\langle\!|N|\!\rangle \prec (p \cdot P')$ **by**(*simp add*: *eqvts*)
moreover from $P'RelQ'$ ‹*eqvt Rel*› **have** $(p \cdot \Psi, p \cdot P', p \cdot p \cdot Q') \in Rel$
by(*auto simp add*: *eqvt-def*)
with ‹*xvec* $\sharp*$ Ψ› ‹$(p \cdot xvec)$ $\sharp*$ Ψ› S ‹*distinctPerm* p›
have $(\Psi, p \cdot P', Q') \in Rel$ **by** *simp*
ultimately show *?thesis* **using** ‹α=¿$M\langle\!|N|\!\rangle$› **by** *blast*
next
case(*cOutput M yvec N*)
from ‹*distinct*(*bn* α)› ‹*bn* α $\sharp*$ *subject* α› ‹α=$M\langle\!|\nu*yvec|\!\rangle\langle N\rangle$› **have** *distinct*

yvec **and** *yvec* ♯* *M* **by** *simp+*
from ‹Ψ ▷ *Q* ⟼α ≺ *Q′*› ‹α=*M*⦇ν*yvec⦈⟨*N*⟩› **have** (*p* ∙ Ψ) ▷ (*p* ∙ *Q*) ⟼(*p* ∙ (*M*⦇ν*yvec⦈⟨*N*⟩ ≺ *Q′*))
by(*auto intro*: *semantics.eqvt*)
with ‹*xvec* ♯* Ψ› ‹(*p* ∙ *xvec*) ♯* Ψ› ‹*xvec* ♯* *Q*› ‹(*p* ∙ *xvec*) ♯* *Q*› *S*
have *QTrans*: Ψ ▷ *Q* ⟼(*p* ∙ *M*)⦇ν*(*p* ∙ *yvec*)⦈⟨(*p* ∙ *N*)⟩ ≺ (*p* ∙ *Q′*)
by(*simp add*: *eqvts*)
with *S* ‹*bn* α ♯* *xvec*› ‹(*p* ∙ *xvec*) ♯* α› ‹α=*M*⦇ν*yvec⦈⟨*N*⟩› **have** Ψ ▷ *Q* ⟼(*p* ∙ *M*)⦇ν*yvec⦈⟨(*p* ∙ *N*)⟩ ≺ (*p* ∙ *Q′*)
by *simp*
moreover from ‹(*p* ∙ *xvec*) ♯* α› **have** (*p* ∙ (*p* ∙ *xvec*)) ♯* (*p* ∙ α)
by(*simp only*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹*distinctPerm p*› ‹α=*M*⦇ν*yvec⦈⟨*N*⟩› **have** *xvec* ♯* (*p* ∙ *M*) **and** *xvec* ♯* (*p* ∙ *N*) **and** *xvec* ♯* (*p* ∙ *yvec*) **by** *simp+*
moreover with *QTrans* ‹*xvec* ♯* *Q*› ‹*distinct yvec*› ‹*yvec* ♯* *M*› **have** *xvec* ♯* (*p* ∙ *Q′*)
apply(*drule-tac outputFreshChainDerivative*(*2*))
by(*auto simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
moreover from ‹*yvec* ♯* *M*› *S* ‹*bn* α ♯* *xvec*› ‹(*p* ∙ *xvec*) ♯* α› ‹α=*M*⦇ν*yvec⦈⟨*N*⟩› ‹*distinctPerm p*›
have *yvec* ♯* (*p* ∙ *M*) **by**(*subst pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*, *symmetric*, **where** *pi*=*p*]) *simp*
ultimately have ∃ *P′*. Ψ ▷ *P* ⟼ (*p* ∙ *M*)⦇ν*yvec⦈⟨(*p* ∙ *N*)⟩ ≺ *P′* ∧ (Ψ, *P′*, (*p* ∙ *Q′*)) ∈ *Rel*
using ‹*bn* α ♯* Ψ› ‹*bn* α ♯* *P*› ‹*bn* α ♯* *Q*›‹*bn* α ♯* *xvec*› ‹*bn* α ♯* *C*› ‹*yvec* ♯* *M*› ‹α=*M*⦇ν*yvec⦈⟨*N*⟩› ‹*distinct yvec*›
by(*simp add*: *Sim*)
then obtain *P′* **where** *PTrans*: Ψ ▷ *P* ⟼ (*p* ∙ *M*)⦇ν*yvec⦈⟨(*p* ∙ *N*)⟩ ≺ *P′* **and** *P′RelQ′*: (Ψ, *P′*, (*p* ∙ *Q′*)) ∈ *Rel*
by *blast*
from *PTrans* **have** (*p* ∙ Ψ) ▷ (*p* ∙ *P*) ⟼ (*p* ∙ ((*p* ∙ *M*)⦇ν*yvec⦈⟨(*p* ∙ *N*)⟩ ≺ *P′*))
by(*rule semantics.eqvt*)
with ‹*xvec* ♯* Ψ› ‹(*p* ∙ *xvec*) ♯* Ψ› ‹*xvec* ♯* *P*› ‹(*p* ∙ *xvec*) ♯* *P*› *S* ‹*distinctPerm p*› ‹*bn* α ♯* *xvec*› ‹(*p* ∙ *xvec*) ♯* α› ‹α=*M*⦇ν*yvec⦈⟨*N*⟩›
have Ψ ▷ *P* ⟼ *M*⦇ν*yvec⦈⟨*N*⟩ ≺ (*p* ∙ *P′*) **by**(*simp add*: *eqvts*)
moreover from *P′RelQ′* ‹*eqvt Rel*› **have** (*p* ∙ Ψ, *p* ∙ *P′*, *p* ∙ *p* ∙ *Q′*) ∈ *Rel*
by(*auto simp add*: *eqvt-def*)
with ‹*xvec* ♯* Ψ› ‹(*p* ∙ *xvec*) ♯* Ψ› *S* ‹*distinctPerm p*›
have (Ψ, *p* ∙ *P′*, *Q′*) ∈ *Rel* **by** *simp*
ultimately show *?thesis* **using** ‹α=*M*⦇ν*yvec⦈⟨*N*⟩› **by** *blast*
next
case(*cBrOutput M yvec N*)
from ‹*distinct*(*bn* α)› ‹*bn* α ♯* *subject* α› ‹α=¡*M*⦇ν*yvec⦈⟨*N*⟩› **have** *distinct yvec* **and** *yvec* ♯* *M* **by** *simp+*
from ‹Ψ ▷ *Q* ⟼α ≺ *Q′*› ‹α=¡*M*⦇ν*yvec⦈⟨*N*⟩› **have** (*p* ∙ Ψ) ▷ (*p* ∙ *Q*) ⟼(*p* ∙ (¡*M*⦇ν*yvec⦈⟨*N*⟩ ≺ *Q′*))
by(*auto intro*: *semantics.eqvt*)
with ‹*xvec* ♯* Ψ› ‹(*p* ∙ *xvec*) ♯* Ψ› ‹*xvec* ♯* *Q*› ‹(*p* ∙ *xvec*) ♯* *Q*› *S*

have *QTrans*: $\Psi \rhd Q \longmapsto ¡(p \cdot M)(\!|\nu{*}(p \cdot yvec)|\!)\langle(p \cdot N)\rangle \prec (p \cdot Q')$
by(*simp add*: *eqvts*)
with S ‹*bn* $\alpha \sharp* xvec$› ‹$(p \cdot xvec) \sharp* \alpha$› ‹$\alpha$=¡$M(\!|\nu{*}yvec|\!)\langle N\rangle$› **have** $\Psi \rhd Q \longmapsto ¡(p \cdot M)(\!|\nu{*}yvec|\!)\langle(p \cdot N)\rangle \prec (p \cdot Q')$
by *simp*
moreover from ‹$(p \cdot xvec) \sharp* \alpha$› **have** $(p \cdot (p \cdot xvec)) \sharp* (p \cdot \alpha)$
by(*simp only*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹*distinctPerm p*› ‹α=¡$M(\!|\nu{*}yvec|\!)\langle N\rangle$› **have** $xvec \sharp* (p \cdot M)$ **and** $xvec \sharp* (p \cdot N)$ **and** $xvec \sharp* (p \cdot yvec)$ **by** *simp+*
moreover with *QTrans* ‹$xvec \sharp* Q$› ‹*distinct yvec*› ‹$yvec \sharp* M$› **have** $xvec \sharp* (p \cdot Q')$
apply(*drule-tac broutputFreshChainDerivative*(*2*))
by (*auto simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
moreover from ‹$yvec \sharp* M$› S ‹*bn* $\alpha \sharp* xvec$› ‹$(p \cdot xvec) \sharp* \alpha$› ‹$\alpha$=¡$M(\!|\nu{*}yvec|\!)\langle N\rangle$› ‹*distinctPerm p*›
have $yvec \sharp* (p \cdot M)$ **by**(*subst pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*, *symmetric*, **where** *pi*=*p*]) *simp*
ultimately have $\exists P'.\ \Psi \rhd P \longmapsto ¡(p \cdot M)(\!|\nu{*}yvec|\!)\langle(p \cdot N)\rangle \prec P' \wedge (\Psi, P', (p \cdot Q')) \in Rel$
using ‹*bn* $\alpha \sharp* \Psi$› ‹*bn* $\alpha \sharp* P$› ‹*bn* $\alpha \sharp* Q$›‹*bn* $\alpha \sharp* xvec$› ‹*bn* $\alpha \sharp* C$› ‹$yvec \sharp* M$› ‹α=¡$M(\!|\nu{*}yvec|\!)\langle N\rangle$› ‹*distinct yvec*›
by(*simp add*: *Sim*)
then obtain P' **where** *PTrans*: $\Psi \rhd P \longmapsto ¡(p \cdot M)(\!|\nu{*}yvec|\!)\langle(p \cdot N)\rangle \prec P'$ **and** *P'RelQ'*: $(\Psi, P', (p \cdot Q')) \in Rel$
by *blast*
from *PTrans* **have** $(p \cdot \Psi) \rhd (p \cdot P) \longmapsto (p \cdot (¡(p \cdot M)(\!|\nu{*}yvec|\!)\langle(p \cdot N)\rangle \prec P'))$
by(*rule semantics.eqvt*)
with ‹$xvec \sharp* \Psi$› ‹$(p \cdot xvec) \sharp* \Psi$› ‹$xvec \sharp* P$› ‹$(p \cdot xvec) \sharp* P$› S ‹*distinctPerm p*› ‹*bn* $\alpha \sharp* xvec$› ‹$(p \cdot xvec) \sharp* \alpha$› ‹$\alpha$=¡$M(\!|\nu{*}yvec|\!)\langle N\rangle$›
have $\Psi \rhd P \longmapsto ¡M(\!|\nu{*}yvec|\!)\langle N\rangle \prec (p \cdot P')$ **by**(*simp add*: *eqvts*)
moreover from *P'RelQ'* ‹*eqvt Rel*› **have** $(p \cdot \Psi, p \cdot P', p \cdot p \cdot Q') \in Rel$
by(*auto simp add*: *eqvt-def*)
with ‹$xvec \sharp* \Psi$› ‹$(p \cdot xvec) \sharp* \Psi$› S ‹*distinctPerm p*›
have $(\Psi, p \cdot P', Q') \in Rel$ **by** *simp*
ultimately show *?thesis* **using** ‹α=¡$M(\!|\nu{*}yvec|\!)\langle N\rangle$› **by** *blast*
next
case *cTau*
from ‹$\Psi \rhd Q \longmapsto \alpha \prec Q'$› ‹$\alpha = \tau$› ‹$xvec \sharp* Q$› **have** $xvec \sharp* Q'$
by(*blast dest*: *tauFreshChainDerivative*)
with ‹$\Psi \rhd Q \longmapsto \alpha \prec Q'$› ‹$\alpha = \tau$›
show *?thesis*
using *Sim* **by** *simp*
qed
qed

lemma *simIFresh*[*consumes 4*, *case-names cSim*]:
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 *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 $\Psi \rhd P \rightsquigarrow [Rel]\ Q$
using *assms*
by (*auto intro*: *simICh*ainFresh[**where** *xvec*=[*x*] **and** *C*=*C*])

lemma *simE*:
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 $\Psi \rhd P \rightsquigarrow [Rel]\ Q$

shows $\bigwedge \alpha\ Q'.\ [\![\Psi \rhd Q \longmapsto \alpha \prec Q';\ bn\ \alpha \sharp* \Psi;\ bn\ \alpha \sharp* P]\!] \Longrightarrow \exists P'.\ \Psi \rhd P \longmapsto \alpha \prec P' \wedge (\Psi, P', Q') \in Rel$
using *assms*
by(*auto simp add*: *simulation-def*)

lemma *simClosedAux*:
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 *p* :: *name prm*

assumes *EqvtRel*: *eqvt Rel*
and *PSimQ*: $\Psi \rhd P \rightsquigarrow [Rel]\ Q$

shows $(p \cdot \Psi) \rhd (p \cdot P) \rightsquigarrow [Rel]\ (p \cdot Q)$
using *EqvtRel*
proof(*induct rule*: *simI*[*of* - - - - (Ψ, *P*, *p*)])
case(*cSim* α Q')
from ‹$p \cdot \Psi \rhd p \cdot Q \longmapsto \alpha \prec Q'$›
have $(rev\ p \cdot p \cdot \Psi) \rhd (rev\ p \cdot p \cdot Q) \longmapsto (rev\ p \cdot (\alpha \prec Q'))$
by(*blast dest*: *semantics.eqvt*)
then have $\Psi \rhd Q \longmapsto (rev\ p \cdot \alpha) \prec (rev\ p \cdot Q')$
by(*simp add*: *eqvts*)

moreover with ‹$bn\ \alpha\ \sharp*\ (\Psi,\ P,\ p)$› **have** $bn\ \alpha\ \sharp*\ \Psi$ **and** $bn\ \alpha\ \sharp*\ P$ **and** $bn\ \alpha\ \sharp*\ p$ **by** *simp+*
ultimately obtain P' **where** *PTrans*: $\Psi \rhd P \longmapsto(rev\ p \cdot \alpha) \prec P'$
and *P'RelQ'*: $(\Psi,\ P',\ rev\ p \cdot Q') \in Rel$
using *PSimQ*
by(*force dest*: *simE freshChainPermSimp simp add*: *eqvts*)
from *PTrans* **have** $(p \cdot \Psi) \rhd (p \cdot P) \longmapsto(p \cdot ((rev\ p \cdot \alpha) \prec P'))$
by(*rule semantics.eqvt*)
with ‹$bn\ \alpha\ \sharp*\ p$› **have** $(p \cdot \Psi) \rhd (p \cdot P) \longmapsto\alpha \prec (p \cdot P')$
by(*simp add*: *eqvts freshChainPermSimp*)
moreover from *P'RelQ' EqvtRel* **have** $(p \cdot (\Psi,\ P',\ rev\ p \cdot Q')) \in Rel$
by(*simp only*: *eqvt-def*)
then have $(p \cdot \Psi,\ p \cdot P',\ Q') \in Rel$ **by** *simp*
ultimately show *?case* **by** *blast*
qed

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 $P \rightsquigarrow[Rel]\ Q \Longrightarrow (p \cdot P) \rightsquigarrow[Rel]\ (p \cdot Q)$
using *EqvtRel*
by(*force dest*: *simClosedAux simp add*: *permBottom*)+

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 $\Psi \rhd P \rightsquigarrow[Rel]\ P$
using *assms*
by(*auto simp add*: *simulation-def*)

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 $\Psi \rhd P \rightsquigarrow[Rel'']\ R$
using ‹*eqvt Rel''*›
proof(*induct rule*: *simI*[**where** *C*=*Q*])
case(*cSim* α R')
from *QSimR* ‹$\Psi \rhd R \longmapsto \alpha \prec R'$› ‹$(bn\ \alpha) \sharp* \Psi$› ‹$(bn\ \alpha) \sharp* Q$›
obtain Q' **where** *QTrans*: $\Psi \rhd Q \longmapsto \alpha \prec Q'$ **and** $Q'Rel'R'$: $(\Psi, Q', R') \in Rel'$
by(*blast dest*: *simE*)
from *PSimQ QTrans* ‹$bn\ \alpha \sharp* \Psi$› ‹$bn\ \alpha \sharp* P$›
obtain P' **where** *PTrans*: $\Psi \rhd P \longmapsto \alpha \prec P'$ **and** $P'RelQ'$: $(\Psi, P', Q') \in Rel$
by(*blast dest*: *simE*)
with *PTrans* $Q'Rel'R'$ $P'RelQ'$ *Set*
show *?case* **by** *blast*
qed

lemma *statEqSim*:
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 $\Psi' \rhd P \rightsquigarrow[Rel']\ Q$
using ‹*eqvt Rel'*›
proof(*induct rule*: *simI*[*of* - - - - Ψ])
case(*cSim* α Q')
from ‹$\Psi' \rhd Q \longmapsto \alpha \prec Q'$› ‹$\Psi \simeq \Psi'$›
have $\Psi \rhd Q \longmapsto \alpha \prec Q'$ **by**(*metis statEqTransition AssertionStatEqSym*)
with *PSimQ* ‹$bn\ \alpha \sharp* \Psi$› ‹$bn\ \alpha \sharp* P$›
obtain P' **where** $\Psi \rhd P \longmapsto \alpha \prec P'$ **and** $(\Psi, P', Q') \in Rel$
by(*blast dest*: *simE*)

from ‹$\Psi \rhd P \longmapsto \alpha \prec P'$› ‹$\Psi \simeq \Psi'$› **have** $\Psi' \rhd P \longmapsto \alpha \prec P'$
by(*rule statEqTransition*)
moreover from ‹$(\Psi, P', Q') \in Rel$› ‹$\Psi \simeq \Psi'$› **have** $(\Psi', P', Q') \in Rel'$
by(*rule C1*)
ultimately show *?case* **by** *blast*
qed

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 $\Psi \rhd P \rightsquigarrow[B]\ Q$
  using *assms*
  by(*simp* (*no-asm*) *add*: *simulation-def*) (*auto dest*: *simE*)

end

end
theory *Bisimulation*
  imports *Simulation*
begin

This file is a (heavily modified) variant of the theory *Psi_Calculi.Bisimulation* from [1].

context *env* **begin**

lemma *monoCoinduct*: $\bigwedge x\ y\ xa\ xb\ xc\ P\ Q\ \Psi.$
$x \leq y \Longrightarrow$
$(\Psi \rhd Q \rightsquigarrow[\{(xc, xb, xa).\ x\ xc\ xb\ xa\}]\ P) \longrightarrow$
$(\Psi \rhd Q \rightsquigarrow[\{(xb, xa, xc).\ y\ xb\ xa\ xc\}]\ P)$
  apply(*rule impI*)
  apply(*rule monotonic*)
  by(*auto dest*: *le-funE*)

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 ($\langle$- $\rhd$ - $\sim$ -$\rangle$ [70, 70, 70] 65) **where** $\Psi \rhd P \sim Q \equiv (\Psi, P, Q) \in bisim$
abbreviation
  bisimNilJudge ($\langle$- $\sim$ -$\rangle$ [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 −
have $X \cup bisim = \{(\Psi, P, Q).\ (\Psi, P, Q) \in X \vee (\Psi, P, Q) \in bisim\}$ **by** *auto*
with *assms* **show** *?thesis*
by *coinduct simp*
qed

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 −
have $X \cup bisim = \{(\Psi, P, Q).\ (\Psi, P, Q) \in X \vee (\Psi, P, Q) \in bisim\}$ **by** *auto*
with *assms* **show** *?thesis*
by *coinduct simp*
qed

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) Ψ $\wedge$
$\Psi \rhd P \rightsquigarrow[X]\ Q \wedge$
$(\forall \Psi'.\ (\Psi \otimes \Psi', P, Q) \in X) \wedge (\Psi, Q, P) \in X$

shows $(\Psi, P, Q) \in$ *bisim*
using *assms*
by(*coinduct rule*: *bisimCoinductAux*) (*blast intro*: *monotonic*)

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) Ψ
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 $-$
have $X \cup$ *bisim* $= \{(\Psi, P, Q).\ (\Psi, P, Q) \in X \vee (\Psi, P, Q) \in$ *bisim*$\}$ **by** *auto*
with *assms* **show** *?thesis*
by(*coinduct rule*: *bisimCoinduct*) (*blast intro*: *monotonic*)+
qed

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) Ψ
and $\Psi \rhd P \rightsquigarrow[$*bisim*$]\ Q$
and $(\Psi \otimes \Psi', P, Q) \in$ *bisim*
and $(\Psi, Q, P) \in$ *bisim*
using *assms*
by(*auto simp add*: *intro*: *bisim.cases*)

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) Ψ
and $\Psi \rhd P \rightsquigarrow[bisim]\ Q$
and $\forall \Psi'.\ (\Psi \otimes \Psi', P, Q) \in bisim$
and $(\Psi, Q, P) \in bisim$

shows $(\Psi, P, Q) \in bisim$
using *assms*
by(*auto intro*: *bisim.step*)

lemma *bisimReflexive*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$

shows $\Psi \rhd P \sim P$
proof −
let *?X* = $\{(\Psi, P, P) \mid \Psi\ P.\ True\}$
have $(\Psi, P, P) \in$ *?X* **by** *simp*
then show *?thesis*
by(*coinduct rule*: *bisimWeakCoinduct*, *auto intro*: *reflexive*)
qed

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 −
let *?X* = $\{(p \cdot \Psi, p \cdot P, p \cdot Q) \mid (p::name\ prm)\ \Psi\ \ P\ Q.\ \Psi \rhd P \sim Q\}$
from *PBisimQ* **have** $(p \cdot \Psi, p \cdot P, p \cdot Q) \in$ *?X* **by** *blast*
then show *?thesis*
proof(*coinduct rule*: *bisimWeakCoinduct*)
case(*cStatEq* Ψ P Q)
have $\bigwedge \Psi\ P\ Q\ (p::name\ prm).$ *insertAssertion* (*extractFrame* P) $\Psi \simeq_F$ *insertAssertion* (*extractFrame* Q) $\Psi \Longrightarrow$
insertAssertion (*extractFrame*$(p \cdot P)$) $(p \cdot \Psi) \simeq_F$ *insertAssertion* (*extractFrame*$(p \cdot Q)$) $(p \cdot \Psi)$
by(*drule FrameStatEqClosed*) (*simp add*: *eqvts*)

with ‹$(\Psi, P, Q) \in$ *?X*› **show** *?case* **by**(*blast dest*: *bisimE*)
next
case(*cSim* Ψ P Q)
{
fix $p ::$ *name prm*

```
    fix Ψ P Q
    have eqvt ?X
      apply(clarsimp simp add: eqvt-def)
      by (metis (no-types, opaque-lifting) pt-name2)
    moreover assume Ψ ▷ P ⇝[bisim] Q
    then have Ψ ▷ P ⇝[?X] Q
      apply(rule monotonic[where A=bisim], auto)
      by(rule exI[where x=[]::name prm]) auto
    ultimately have ((p::name prm) • Ψ) ▷ (p • P) ⇝[?X] (p • Q)
      by(rule simClosed)
  }
  with ‹(Ψ, P, Q) ∈ ?X› show ?case
    by(blast dest: bisimE)
 next
  case(cExt Ψ P Q Ψ')
  {
    fix p :: name prm
    fix Ψ P Q Ψ'
    assume ∀Ψ'. (Ψ ⊗ Ψ', P, Q) ∈ bisim
    then have ((p • Ψ) ⊗ Ψ', p • P, p • Q) ∈ ?X
      apply(clarsimp)
      apply(rule exI[where x=p])
      apply(rule exI[where x=Ψ ⊗ (rev p • Ψ')])
      by(auto simp add: eqvts)
  }
  with ‹(Ψ, P, Q) ∈ ?X› show ?case
    by(blast dest: bisimE)
 next
  case(cSym Ψ P Q)
  then show ?case
    by(blast dest: bisimE)
 qed
qed

lemma bisimEqvt[simp]:
 shows eqvt bisim
 by(auto simp add: eqvt-def bisimClosed)

lemma statEqBisim:
 fixes Ψ  :: 'b
   and P  :: ('a, 'b, 'c) psi
   and Q  :: ('a, 'b, 'c) psi
   and Ψ' :: 'b

assumes Ψ ▷ P ∼ Q
 and    Ψ ≃ Ψ'

shows Ψ' ▷ P ∼ Q
proof −
```

let *?X* = {(Ψ′, *P*, *Q*) | Ψ *P Q* Ψ′. Ψ ▷ *P* ∼ *Q* ∧ Ψ ≃ Ψ′}
from ‹Ψ ▷ *P* ∼ *Q*› ‹Ψ ≃ Ψ′› **have** (Ψ′, *P*, *Q*) ∈ *?X* **by** *auto*
then show *?thesis*
proof(*coinduct rule*: *bisimCoinduct*)
case(*cStatEq* Ψ′ *P Q*)
from ‹(Ψ′, *P*, *Q*) ∈ *?X*› **obtain** Ψ **where** Ψ ▷ *P* ∼ *Q* **and** Ψ ≃ Ψ′
by *auto*
from ‹Ψ ▷ *P* ∼ *Q*› **have** *PeqQ*: *insertAssertion* (*extractFrame P*) Ψ ≃$_F$ *insertAssertion* (*extractFrame Q*) Ψ
by(*rule bisimE*)

obtain A_P Ψ_P **where** *FrP*: *extractFrame P* = ⟨A_P, Ψ_P⟩ **and** A_P ♯* Ψ **and** A_P ♯* Ψ′
by(*rule freshFrame*[**where** *C*=(Ψ, Ψ′)]) *auto*
obtain A_Q Ψ_Q **where** *FrQ*: *extractFrame Q* = ⟨A_Q, Ψ_Q⟩ **and** A_Q ♯* Ψ **and** A_Q ♯* Ψ′
by(*rule freshFrame*[**where** *C*=(Ψ, Ψ′)]) *auto*

from *PeqQ FrP FrQ* ‹A_P ♯* Ψ› ‹A_Q ♯* Ψ› ‹Ψ ≃ Ψ′›
have ⟨A_P, Ψ′ ⊗ Ψ_P⟩ ≃$_F$ ⟨A_Q, Ψ′ ⊗ Ψ_Q⟩
by *simp* (*metis frameIntComposition FrameStatEqTrans FrameStatEqSym*)
with *FrP FrQ* ‹A_P ♯* Ψ′› ‹A_Q ♯* Ψ′› **show** *?case* **by** *simp*
next
case(*cSim* Ψ′ *P Q*)
from ‹(Ψ′, *P*, *Q*) ∈ *?X*› **obtain** Ψ **where** Ψ ▷ *P* ∼ *Q* **and** Ψ ≃ Ψ′
by *auto*
from ‹Ψ ▷ *P* ∼ *Q*› **have** Ψ ▷ *P* ⇝[*bisim*] *Q* **by**(*blast dest*: *bisimE*)
moreover have *eqvt ?X*
by(*auto simp add*: *eqvt-def*) (*metis bisimClosed AssertionStatEqClosed*)
then have *eqvt*(*?X* ∪ *bisim*) **by** *auto*
moreover note ‹Ψ ≃ Ψ′›
moreover have ⋀Ψ *P Q* Ψ′. ⟦Ψ ▷ *P* ∼ *Q*; Ψ ≃ Ψ′⟧ ⟹ (Ψ′, *P*, *Q*) ∈ *?X* ∪ *bisim*
by *auto*
ultimately show *?case*
by(*rule statEqSim*)
next
case(*cExt* Ψ′ *P Q* Ψ″)
from ‹(Ψ′, *P*, *Q*) ∈ *?X*› **obtain** Ψ **where** Ψ ▷ *P* ∼ *Q* **and** Ψ ≃ Ψ′
by *auto*
from ‹Ψ ▷ *P* ∼ *Q*› **have** Ψ ⊗ Ψ″ ▷ *P* ∼ *Q* **by**(*rule bisimE*)
moreover from ‹Ψ ≃ Ψ′› **have** Ψ ⊗ Ψ″ ≃ Ψ′ ⊗ Ψ″ **by**(*rule Composition*)
ultimately show *?case* **by** *blast*
next
case(*cSym* Ψ′ *P Q*)
from ‹(Ψ′, *P*, *Q*) ∈ *?X*› **obtain** Ψ **where** Ψ ▷ *P* ∼ *Q* **and** Ψ ≃ Ψ′
by *auto*
from ‹Ψ ▷ *P* ∼ *Q*› **have** Ψ ▷ *Q* ∼ *P* **by**(*rule bisimE*)
then show *?case* **using** ‹Ψ ≃ Ψ′› **by** *auto*

```
  qed
qed

lemma bisimTransitive:
  fixes Ψ :: 'b
    and P :: ('a, 'b, 'c) psi
    and Q :: ('a, 'b, 'c) psi
    and R :: ('a, 'b, 'c) psi

assumes PQ: Ψ ▷ P ∼ Q
  and   QR: Ψ ▷ Q ∼ R

shows Ψ ▷ P ∼ R
proof −
  let ?X = {(Ψ, P, R) | Ψ P Q R. Ψ ▷ P ∼ Q ∧ Ψ ▷ Q ∼ R}
  from PQ QR have (Ψ, P, R) ∈ ?X by auto
  then show ?thesis
  proof(coinduct rule: bisimCoinduct)
    case(cStatEq Ψ P R)
    then show ?case by(blast dest: bisimE FrameStatEqTrans)
  next
    case(cSim Ψ P R)
    {
      fix Ψ P Q R
      assume Ψ ▷ P ⇝[bisim] Q and Ψ ▷ Q ⇝[bisim] R
      moreover have eqvt ?X
        by(force simp add: eqvt-def dest: bisimClosed)
      with bisimEqvt have eqvt (?X ∪ bisim) by blast
      moreover have ?X ⊆ ?X ∪ bisim by auto
      ultimately have Ψ ▷ P ⇝[(?X ∪ bisim)] R
        by(force intro: transitive)
    }
    with ‹(Ψ, P, R) ∈ ?X› show ?case
      by(blast dest: bisimE)
  next
    case(cExt Ψ P R Ψ')
    then show ?case by(blast dest: bisimE)
  next
    case(cSym Ψ P R)
    then show ?case by(blast dest: bisimE)
  qed
qed

lemma weakTransitiveCoinduct[case-names cStatEq 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 × ('a, 'b, 'c) psi × ('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*) Ψ
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 −
let *?X* $= \{(\Psi,\ P,\ Q) \mid \Psi\ P\ P'\ Q'\ Q.\ \Psi \rhd P \sim P' \wedge (\Psi,\ P',\ Q') \in X \wedge \Psi \rhd Q' \sim Q\}$
from p **have** $(\Psi,\ P,\ Q) \in$ *?X*
by(*blast intro*: *bisimReflexive*)
then show *?thesis*
proof(*coinduct rule*: *bisimWeakCoinduct*)
case(*cStatEq* $\Psi\ P\ Q$)
then show *?case*
by(*blast dest*: *rStatEq bisimE FrameStatEqTrans*)
next
case(*cSim* $\Psi\ P\ Q$)
{
fix $\Psi\ P\ P'\ Q'\ Q$
assume $\Psi \rhd P \rightsquigarrow[bisim]\ P'$
moreover assume *P'RelQ'*: $(\Psi,\ P',\ Q') \in X$
then have $\Psi \rhd P' \rightsquigarrow[?X]\ Q'$ **by**(*rule rSim*)
moreover from ‹*eqvt X*› *P'RelQ'* **have** *eqvt ?X*
apply(*clarsimp simp add*: *eqvt-def*)
apply(*drule bisimClosed*)
apply(*drule bisimClosed*)
apply(*rule exI*)
apply(*rule conjI*)
apply *assumption*
apply(*rule exI*)
by *auto*
ultimately have $\Psi \rhd P \rightsquigarrow[?X]\ Q'$
by(*force intro*: *transitive dest*: *bisimTransitive*)
moreover assume $\Psi \rhd Q' \rightsquigarrow[bisim]\ Q$
ultimately have $\Psi \rhd P \rightsquigarrow[?X]\ Q$ **using** ‹*eqvt ?X*›
by(*force intro*: *transitive dest*: *bisimTransitive*)
}
with ‹$(\Psi,\ P,\ Q) \in$ *?X*› **show** *?case*
by(*blast dest*: *bisimE*)
next
case(*cExt* $\Psi\ P\ Q\ \Psi'$)

```
    then show ?case by(blast dest: bisimE intro: rExt)
  next
    case(cSym Ψ P Q)
    then show ?case by(blast dest: bisimE intro: rSym)
  qed
qed

lemma weakTransitiveCoinduct'[case-names cStatEq 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 × ('a, 'b, 'c) psi × ('a, 'b, 'c) psi) set

assumes p: (Ψ, P, Q) ∈ X
  and Eqvt: eqvt X
  and rStatEq: ⋀Ψ P Q. (Ψ, P, Q) ∈ X ⟹ insertAssertion (extractFrame P) Ψ
≃_F insertAssertion (extractFrame Q) Ψ
  and rSim: ⋀Ψ P Q. (Ψ, P, Q) ∈ X ⟹ Ψ ▷ P ⇝[({(Ψ, P, Q) | Ψ P P' Q' Q.
Ψ ▷ P ∼ P' ∧
                                                    (Ψ, P', Q') ∈ X ∧
                                                    Ψ ▷ Q' ∼ Q})] Q
  and rExt: ⋀Ψ P Q Ψ'. (Ψ, P, Q) ∈ X ⟹ (Ψ ⊗ Ψ', P, Q) ∈ X
  and rSym: ⋀Ψ P Q. (Ψ, P, Q) ∈ X ⟹
                    (Ψ, Q, P) ∈ {(Ψ, P, Q) | Ψ P P' Q' Q. Ψ ▷ P ∼ P' ∧ (Ψ, P',
Q') ∈ X ∧ Ψ ▷ Q' ∼ Q}

shows Ψ ▷ P ∼ Q
proof −
  let ?X = {(Ψ, P, Q) | Ψ P P' Q' Q. Ψ ▷ P ∼ P' ∧ (Ψ, P', Q') ∈ X ∧ Ψ ▷ Q'
∼ Q}
  from p have (Ψ, P, Q) ∈ ?X
    by(blast intro: bisimReflexive)
  then show ?thesis
  proof(coinduct rule: bisimWeakCoinduct)
    case(cStatEq Ψ P Q)
    then show ?case
      by(blast dest: rStatEq bisimE FrameStatEqTrans)
  next
    case(cSim Ψ P Q)
    {
      fix Ψ P P' Q' Q
      assume Ψ ▷ P ⇝[bisim] P'
      moreover assume P'RelQ': (Ψ, P', Q') ∈ X
      then have Ψ ▷ P' ⇝[?X] Q' by(rule rSim)
      moreover from ‹eqvt X› P'RelQ' have eqvt ?X
        apply(clarsimp simp add: eqvt-def)
        apply(drule bisimClosed)
        apply(drule bisimClosed)
```

```
      apply(rule exI)
      apply(rule conjI)
       apply assumption
      apply(rule exI)
      by auto
    ultimately have Ψ ▷ P ⇝[?X] Q'
      by(force intro: transitive dest: bisimTransitive)
    moreover assume Ψ ▷ Q' ⇝[bisim] Q
    ultimately have Ψ ▷ P ⇝[?X] Q using ‹eqvt ?X›
      by(force intro: transitive dest: bisimTransitive)
  }
  with ‹(Ψ, P, Q) ∈ ?X› show ?case
    by(blast dest: bisimE)
 next
  case(cExt Ψ P Q Ψ')
  then show ?case by(blast dest: bisimE intro: rExt)
 next
  case(cSym Ψ P Q)
  then show ?case
    apply clarsimp
    apply(drule rSym)
    apply clarsimp
    by(metis bisimTransitive bisimE(4))
 qed
qed

lemma weakTransitiveCoinduct''[case-names cStatEq 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 × ('a, 'b, 'c) psi × ('a, 'b, 'c) psi) set

assumes p: (Ψ, P, Q) ∈ X
 and Eqvt: eqvt X
 and rStatEq: ⋀Ψ P Q. (Ψ, P, Q) ∈ X ⟹ insertAssertion (extractFrame P) Ψ
≃F insertAssertion (extractFrame Q) Ψ
 and rSim: ⋀Ψ P Q. (Ψ, P, Q) ∈ X ⟹ Ψ ▷ P ⇝[({(Ψ, P, Q) | Ψ P P' Q' Q.
Ψ ▷ P ∼ P' ∧
                                                    (Ψ, P', Q') ∈ X ∧
                                                    Ψ ▷ Q' ∼ Q})] Q
 and rExt: ⋀Ψ P Q Ψ'. (Ψ, P, Q) ∈ {(Ψ, P, Q) | Ψ P P' Q' Q. Ψ ▷ P ∼ P' ∧
(Ψ, P', Q') ∈ X ∧ Ψ ▷ Q' ∼ Q} ⟹
                        (Ψ ⊗ Ψ', P, Q) ∈ {(Ψ, P, Q) | Ψ P P' Q' Q. Ψ ▷ P ∼ P' ∧
(Ψ, P', Q') ∈ X ∧ Ψ ▷ Q' ∼ Q}
 and rSym: ⋀Ψ P Q. (Ψ, P, Q) ∈ {(Ψ, P, Q) | Ψ P P' Q' Q. Ψ ▷ P ∼ P' ∧
(Ψ, P', Q') ∈ X ∧ Ψ ▷ Q' ∼ Q} ⟹
                    (Ψ, Q, P) ∈ {(Ψ, P, Q) | Ψ P P' Q' Q. Ψ ▷ P ∼ P' ∧ (Ψ, P',
Q') ∈ X ∧ Ψ ▷ Q' ∼ Q}
```

shows $\Psi \rhd P \sim Q$
proof −
let *?X* = $\{(\Psi, P, Q) \mid \Psi\ P\ P'\ Q'\ Q.\ \Psi \rhd P \sim P' \wedge (\Psi, P', Q') \in X \wedge \Psi \rhd Q' \sim Q\}$
from *p* **have** $(\Psi, P, Q) \in$ *?X*
by(*blast intro*: *bisimReflexive*)
then show *?thesis*
proof(*coinduct rule*: *bisimWeakCoinduct*)
case(*cStatEq* Ψ P Q)
then show *?case*
by(*blast dest*: *rStatEq bisimE FrameStatEqTrans*)
next
case(*cSim* Ψ P Q)
{
fix Ψ P P' Q' Q
assume $\Psi \rhd P \rightsquigarrow[bisim]\ P'$
moreover assume *P'RelQ'*: $(\Psi, P', Q') \in X$
then have $\Psi \rhd P' \rightsquigarrow[?X]\ Q'$ **by**(*rule rSim*)
moreover from ‹*eqvt X*› *P'RelQ'* **have** *eqvt ?X*
apply(*clarsimp simp add*: *eqvt-def*)
apply(*drule bisimClosed*)
apply(*drule bisimClosed*)
apply(*rule exI*)
apply(*rule conjI*)
apply *assumption*
apply(*rule exI*)
by *auto*
ultimately have $\Psi \rhd P \rightsquigarrow[?X]\ Q'$
by(*force intro*: *transitive dest*: *bisimTransitive*)
moreover assume $\Psi \rhd Q' \rightsquigarrow[bisim]\ Q$
ultimately have $\Psi \rhd P \rightsquigarrow[?X]\ Q$ **using** ‹*eqvt ?X*›
by(*force intro*: *transitive dest*: *bisimTransitive*)
}
with ‹$(\Psi, P, Q) \in$ *?X*› **show** *?case*
by(*blast dest*: *bisimE*)
next
case(*cExt* Ψ P Q Ψ')
then show *?case* **by**(*rule rExt*)
next
case(*cSym* Ψ P Q)
then show *?case* **by**(*rule rSym*)
qed
qed

lemma *transitiveCoinduct*[*case-names cStatEq 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 *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 −
from *p* **have** $(\Psi, P, Q) \in (X \cup bisim)$
by *blast*
moreover from ‹*eqvt X*› *bisimEqvt* **have** *eqvt* $(X \cup bisim)$
by *auto*
ultimately show *?thesis*
proof(*coinduct rule*: *weakTransitiveCoinduct'*)
case(*cStatEq* Ψ *P Q*)
then show *?case*
by(*blast intro*: *rStatEq dest*: *bisimE*)
next
case(*cSim* Ψ *P Q*)
then show *?case*
apply *clarsimp*
apply (*erule disjE*)
apply(*blast intro*: *rSim*)
apply(*drule bisimE*(*2*))
apply(*rule monotonic*, *simp*)
by(*force intro*: *bisimReflexive*)
next
case(*cExt* Ψ *P Q* Ψ')
then show *?case*
by(*blast dest*: *bisimE rExt*)
next
case(*cSym* Ψ *P Q*)
then show *?case* **by**(*blast dest*: *bisimE rSym intro*: *bisimReflexive*)
qed
qed

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 $-$
from p **have** $(\Psi, P, Q) \in (X \cup bisim)$
by *blast*
moreover from ‹$eqvt\ X$› *bisimEqvt* **have** $eqvt\ (X \cup bisim)$
by *auto*
ultimately show *?thesis*
proof(*coinduct rule*: *weakTransitiveCoinduct'*)
case(*cStatEq* $\Psi\ P\ Q$)
then show *?case*
by(*blast intro*: *rStatEq dest*: *bisimE*)
next
case(*cSim* $\Psi\ P\ Q$)
then show *?case*
apply $-$
apply(*cases* $(\Psi, P, Q) \in X$)
apply(*rule rSim*)
apply *simp*
apply(*clarify*)
apply(*drule bisimE*(*2*))
apply(*rule monotonic*, *simp*)
by(*force intro*: *bisimReflexive*)
next
case(*cExt* $\Psi\ P\ Q\ \Psi'$)
then show *?case*
by(*blast dest*: *bisimE rExt*)
next
case(*cSym* $\Psi\ P\ Q$)
then show *?case*
apply *clarsimp*

```
    apply (erule disjE)
     apply(drule rSym)
     apply clarsimp
    apply(rule exI[where x=Q])
    apply(simp add: bisimReflexive)
    apply(rule exI)
    by(auto intro: bisimReflexive dest: bisimE(4))
  qed
qed

lemma bisimSymmetric:
  fixes Ψ :: 'b
    and P :: ('a, 'b, 'c) psi
    and Q :: ('a, 'b, 'c) psi

assumes Ψ ⊳ P ∼ Q

shows Ψ ⊳ Q ∼ P
  using assms
  by(rule bisimE)

lemma eqvtTrans[intro]:
  assumes eqvt X

shows eqvt {(Ψ, P, Q) | Ψ P P' Q' Q. Ψ ⊳ P ∼ P' ∧ ((Ψ, P', Q') ∈ X ∨ Ψ ⊳
P' ∼ Q') ∧ Ψ ⊳ Q' ∼ Q}
  using assms
  apply(clarsimp simp add: eqvt-def)
  apply(erule disjE)
   using ballE bisimClosed apply fastforce
  by(blast dest: bisimClosed)

lemma eqvtWeakTrans[intro]:
  assumes eqvt X

shows eqvt {(Ψ, P, Q) | Ψ P P' Q' Q. Ψ ⊳ P ∼ P' ∧ (Ψ, P', Q') ∈ X ∧ Ψ ⊳ Q'
∼ Q}
  using assms
  apply(clarsimp simp add: eqvt-def)
  using ballE bisimClosed by fastforce

inductive-set
   rel-trancl :: ('b × ('a,'b,'c) psi × ('a,'b,'c) psi) set ⇒ ('b × ('a,'b,'c) psi ×
('a,'b,'c) psi) set  (‹(-⋆)› [1000] 999)
  for r :: ('b × ('a,'b,'c) psi × ('a,'b,'c) psi) set
  where
    r-into-rel-trancl [intro, Pure.intro]: (Ψ, P,Q) : r ==> (Ψ,P,Q) : r⋆
  | rel-trancl-into-rel-trancl [Pure.intro]: (Ψ,P,Q) : r⋆ ==> (Ψ,Q,R) : r ==>
(Ψ,P,R) : r⋆
```

lemma *rel-trancl-transitive*:
assumes $(\Psi,P,Q) \in Rel^\star$
and $(\Psi,Q,R) \in Rel^\star$
shows $(\Psi,P,R) \in Rel^\star$
using ‹$(\Psi,Q,R) \in Rel^\star$› ‹$(\Psi,P,Q) \in Rel^\star$›
by(*induct rule*: *rel-trancl.induct*) (*auto intro*: *rel-trancl-into-rel-trancl*)

lemma *rel-trancl-eqvt*:
assumes *eqvt X*
shows *eqvt*($X^\star$)
proof −
{
fix *p*::*name prm*
and Ψ *P Q*
assume $(\Psi,P,Q) \in X^\star$
then have $(p\cdot(\Psi,P,Q)) \in X^\star$
proof(*induct rule*: *rel-trancl.induct*)
case(*r-into-rel-trancl* Ψ *P Q*)
with ‹*eqvt X*› **have** $(p\cdot(\Psi,\ P,\ Q)) \in X$
unfolding *eqvt-def* **by** *auto*
then show *?case* **by** *auto*
next
case(*rel-trancl-into-rel-trancl* Ψ *P Q R*)
from ‹$(\Psi,\ Q,\ R) \in X$› ‹*eqvt X*› **have** $(p\cdot(\Psi,\ Q,\ R)) \in X$
unfolding *eqvt-def* **by** *auto*
then have $(p\cdot(\Psi,\ Q,\ R)) \in X^\star$
by *auto*
with ‹$p \cdot (\Psi,\ P,\ Q) \in X^\star$› **show** *?case* **by**(*auto dest*: *rel-trancl-transitive*)
qed
}
then show *?thesis* **unfolding** *eqvt-def*
by *auto*
qed

lemma *bisimStarWeakCoinduct*[*consumes 2*, *case-names cStatEq cSim cExt cSym*]:
fixes *F* :: *'b*
and *P* :: (*'a*, *'b*, *'c*) *psi*
and *Q* :: (*'a*, *'b*, *'c*) *psi*
and *X* :: (*'b* × (*'a*, *'b*, *'c*) *psi* × (*'a*, *'b*, *'c*) *psi*) *set*

assumes $(\Psi,\ P,\ Q) \in X$
and *eqvt X*
and *rStatEq*: $\bigwedge\Psi'\ R\ S.\ (\Psi',\ R,\ S) \in X \Longrightarrow$ *insertAssertion* (*extractFrame R*) $\Psi' \simeq_F$ *insertAssertion* (*extractFrame S*) Ψ'
and *rSim*: $\bigwedge\Psi'\ R\ S.\ (\Psi',\ R,\ S) \in X \Longrightarrow \Psi' \rhd R \rightsquigarrow[X^\star]\ S$
and *rExt*: $\bigwedge\Psi'\ R\ S\ \Psi''.\ (\Psi',\ R,\ S) \in X \Longrightarrow (\Psi' \otimes \Psi'',\ R,\ S) \in X$
and *rSym*: $\bigwedge\Psi'\ R\ S.\ (\Psi',\ R,\ S) \in X \Longrightarrow (\Psi',\ S,\ R) \in X$

shows $(\Psi,\ P,\ Q) \in bisim$
proof −
have $eqvt(X^\star)$ **using** ‹$eqvt\ X$›
by(*rule rel-trancl-eqvt*)
from ‹$(\Psi,\ P,\ Q) \in X$›
have $(\Psi,\ P,\ Q) \in X^\star$
by *force*
then show *?thesis*
proof(*coinduct rule*: *bisimWeakCoinduct*)
case(*cSim* $\Psi'\ R\ S$)
then show *?case*
proof(*induct rule*: *rel-trancl.induct*)
case(*r-into-rel-trancl* $\Psi\ P\ Q$)
then show *?case*
by(*rule rSim*)
next
case(*rel-trancl-into-rel-trancl* $\Psi\ P\ Q\ R$)
note ‹$\Psi \rhd P \rightsquigarrow[X^\star]\ Q$›
moreover from ‹$(\Psi,\ Q,\ R) \in X$› **have** $\Psi \rhd Q \rightsquigarrow[X^\star]\ R$
by(*rule rSim*)
moreover note ‹$eqvt(X^\star)$›
moreover have $\{(\Psi,\ P,\ R)\ |\Psi\ P\ R.\ \exists\, Q.\ (\Psi,\ P,\ Q) \in X^\star \wedge (\Psi,\ Q,\ R) \in X^\star\} \subseteq X^\star$
by(*blast intro*: *rel-trancl-transitive*)
ultimately show *?case*
using *transitive* **by** *meson*
qed
next
case ($cStatEq\ \Psi\ P\ Q$)
then show *?case*
proof(*induct rule*: *rel-trancl.induct*)
case(*r-into-rel-trancl* $\Psi\ P\ Q$)
then show *?case*
by(*rule rStatEq*)
next
case(*rel-trancl-into-rel-trancl* $\Psi\ P\ Q\ R$)
from ‹$(\Psi,\ Q,\ R) \in X$› **have** $insertAssertion\ (extractFrame\ Q)\ \Psi \simeq_F insertAssertion\ (extractFrame\ R)\ \Psi$
by(*rule rStatEq*)
with ‹$insertAssertion\ (extractFrame\ P)\ \Psi \simeq_F insertAssertion\ (extractFrame\ Q)\ \Psi$›
show *?case*
by(*rule FrameStatEqTrans*)
qed
next
case(*cExt* $\Psi\ P\ Q\ \Psi'$)
then show *?case*
proof(*induct rule*: *rel-trancl.induct*)
case(*r-into-rel-trancl* $\Psi\ P\ Q$)

```
      then have (Ψ ⊗ Ψ', P, Q) ∈ X by(rule rExt)
      then show ?case
        by force
    next
      case(rel-trancl-into-rel-trancl Ψ P Q R)
      from ‹(Ψ, Q, R) ∈ X› have (Ψ ⊗ Ψ', Q, R) ∈ X by(rule rExt)
      then have (Ψ ⊗ Ψ', Q, R) ∈ X* by force
      with ‹(Ψ ⊗ Ψ', P, Q) ∈ X*›
      show ?case
        by(rule rel-trancl-transitive)
    qed
  next
    case(cSym Ψ P Q)
    then show ?case
    proof(induct rule: rel-trancl.induct)
      case(r-into-rel-trancl Ψ P Q)
      then have (Ψ, Q, P) ∈ X by(rule rSym)
      then show ?case
        by force
    next
      case(rel-trancl-into-rel-trancl Ψ P Q R)
      from ‹(Ψ, Q, R) ∈ X› have (Ψ, R, Q) ∈ X by(rule rSym)
      then have (Ψ, R, Q) ∈ X* by force
      then show ?case using ‹(Ψ, Q, P) ∈ X*›
        by(rule rel-trancl-transitive)
    qed
  qed
qed

lemma weakTransitiveStarCoinduct[case-names cStatEq 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 × ('a, 'b, 'c) psi × ('a, 'b, 'c) psi) set

assumes p: (Ψ, P, Q) ∈ X
  and Eqvt: eqvt X
  and rStatEq: ⋀Ψ P Q. (Ψ, P, Q) ∈ X ⟹ insertAssertion (extractFrame P) Ψ
≃_F insertAssertion (extractFrame Q) Ψ
  and rSim: ⋀Ψ P Q. (Ψ, P, Q) ∈ X ⟹ Ψ ▷ P ⇝[(({(Ψ, P, Q) | Ψ P P' Q' Q.
Ψ ▷ P ∼ P' ∧
                                                (Ψ, P', Q') ∈ X ∧
                                                Ψ ▷ Q' ∼ Q})*] Q
  and rExt: ⋀Ψ P Q Ψ'. (Ψ, P, Q) ∈ X ⟹ (Ψ ⊗ Ψ', P, Q) ∈ X
  and rSym: ⋀Ψ P Q. (Ψ, P, Q) ∈ X ⟹ (Ψ, Q, P) ∈ X

shows Ψ ▷ P ∼ Q
proof −
```

let $?X = \{(\Psi, P, Q) \mid \Psi\ P\ P'\ Q'\ Q.\ \Psi \rhd P \sim P' \wedge (\Psi, P', Q') \in X \wedge \Psi \rhd Q' \sim Q\}$
from p **have** $(\Psi, P, Q) \in ?X$
by(*blast intro*: *bisimReflexive*)
moreover from ‹*eqvt X*› **have** *eqvt ?X* **by** *auto*
ultimately show *?thesis*
proof(*coinduct rule*: *bisimStarWeakCoinduct*)
case(*cStatEq* Ψ P Q)
then show *?case*
by(*blast dest*: *rStatEq bisimE FrameStatEqTrans*)
next
case(*cSim* Ψ P Q)
then obtain P' Q' **where** $\Psi \rhd P \sim P'$ **and** $(\Psi, P', Q') \in X$ **and** $\Psi \rhd Q' \sim Q$
by *blast*
then have $\Psi \rhd P \rightsquigarrow[bisim]\ P'$ **and** $\Psi \rhd Q' \rightsquigarrow[bisim]\ Q$
by(*auto dest*: *bisimE*)
{
fix Ψ P Q
and Q'::('*a*,'*b*,'*c*) *psi*
assume $\Psi \rhd P \sim Q$
and $(\Psi,Q,Q') \in ?X^\star$
note ‹$(\Psi,Q,Q') \in ?X^\star$› ‹$\Psi \rhd P \sim Q$›
then have $(\Psi,P,Q') \in ?X^\star$
proof(*induct rule*: *rel-trancl.inducts*)
case(*r-into-rel-trancl* Ψ Q Q') **then show** *?case*
by(*blast dest*: *bisimTransitive*)
next
case(*rel-trancl-into-rel-trancl* Ψ P' Q Q')
then show *?case*
by(*blast dest*: *rel-trancl-transitive*)
qed
}
note *tonic* = *this*
{
fix Ψ P Q
and Q'::('*a*,'*b*,'*c*) *psi*
assume $(\Psi,P,Q) \in ?X^\star$
and $\Psi \rhd Q \sim Q'$
then have $(\Psi,P,Q') \in ?X^\star$
proof(*induct arbitrary*: Q' *rule*: *rel-trancl.inducts*)
case(*r-into-rel-trancl* Ψ P Q) **then show** *?case*
by(*blast dest*: *bisimTransitive*)
next
case(*rel-trancl-into-rel-trancl* Ψ P P' Q)
from ‹$(\Psi, P', Q) \in ?X$› ‹$\Psi \rhd Q \sim Q'$› **have** $(\Psi, P', Q') \in ?X$
by(*blast dest*: *bisimTransitive*)
then have $(\Psi, P', Q') \in ?X^\star$
by *blast*

```
      with ‹(Ψ, P, P') ∈ ?X*›
      show ?case
        by(rule rel-trancl-transitive)
    qed
  }
  note tonic2 = this
  from ‹(Ψ, P', Q') ∈ X› have Ψ ▷ P' ⇝[?X*] Q'
    by(rule rSim)
  with ‹Ψ ▷ P ⇝[bisim] P'› have Ψ ▷ P ⇝[?X*] Q'
    apply -
    apply(rule transitive)
     apply assumption
    apply assumption
   apply(rule rel-trancl-eqvt)
   apply(rule ‹eqvt ?X›)
    by(blast intro: tonic)
  with ‹Ψ ▷ Q' ⇝[bisim] Q› show ?case
    apply -
    apply(rule transitive)
     apply assumption
    apply assumption
   apply(rule rel-trancl-eqvt)
   apply(rule ‹eqvt ?X›)
    by(blast intro: tonic2)
 next
  case(cExt Ψ P Q Ψ')
  then show ?case by(blast dest: bisimE intro: rExt)
 next
  case(cSym Ψ P Q)
  then show ?case by(blast dest: bisimE intro: rSym)
 qed
qed

lemma weakTransitiveStarCoinduct'[case-names cStatEq 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 × ('a, 'b, 'c) psi × ('a, 'b, 'c) psi) set

assumes p: (Ψ, P, Q) ∈ X
 and Eqvt: eqvt X
 and rStatEq: ⋀Ψ P Q. (Ψ, P, Q) ∈ X ⟹ insertAssertion (extractFrame P) Ψ
≃F insertAssertion (extractFrame Q) Ψ
 and rSim: ⋀Ψ P Q. (Ψ, P, Q) ∈ X ⟹ Ψ ▷ P ⇝[({(Ψ, P, Q) | Ψ P P' Q' Q.
Ψ ▷ P ∼ P' ∧
                                                    (Ψ, P', Q') ∈ X ∧
                                                    Ψ ▷ Q' ∼ Q})*] Q
 and rExt: ⋀Ψ P Q Ψ'. (Ψ, P, Q) ∈ X ⟹ (Ψ ⊗ Ψ', P, Q) ∈ X
```

```
  and rSym: ⋀Ψ P Q. (Ψ, P, Q) ∈ X ⟹
                    (Ψ, Q, P) ∈ {(Ψ, P, Q) | Ψ P P′ Q′ Q. Ψ ▷ P ∼ P′ ∧ (Ψ, P′,
Q′) ∈ X ∧ Ψ ▷ Q′ ∼ Q}

shows Ψ ▷ P ∼ Q
proof −
  let ?X = {(Ψ, P, Q) | Ψ P P′ Q′ Q. Ψ ▷ P ∼ P′ ∧ (Ψ, P′, Q′) ∈ X ∧ Ψ ▷ Q′
∼ Q}
  from p have (Ψ, P, Q) ∈ ?X
    by(blast intro: bisimReflexive)
  moreover from ‹eqvt X› have eqvt ?X by auto
  ultimately show ?thesis
  proof(coinduct rule: bisimStarWeakCoinduct)
    case(cStatEq Ψ P Q)
    then show ?case
      by(blast dest: rStatEq bisimE FrameStatEqTrans)
  next
    case(cSim Ψ P Q)
    then obtain P′ Q′ where Ψ ▷ P ∼ P′ and (Ψ, P′, Q′) ∈ X and Ψ ▷ Q′ ∼
Q
      by blast
    then have Ψ ▷ P ⇝[bisim] P′ and Ψ ▷ Q′ ⇝[bisim] Q
      by(auto dest: bisimE)
    {
      fix Ψ P Q
        and Q′::('a,'b,'c) psi
      assume Ψ ▷ P ∼ Q
        and  (Ψ,Q,Q′) ∈ ?X⋆
      note ‹(Ψ,Q,Q′) ∈ ?X⋆› ‹Ψ ▷ P ∼ Q›
      then have (Ψ,P,Q′) ∈ ?X⋆
      proof(induct rule: rel-trancl.inducts)
        case(r-into-rel-trancl Ψ Q Q′) then show ?case
          by(blast dest: bisimTransitive)
      next
        case(rel-trancl-into-rel-trancl Ψ P′ Q Q′)
        then show ?case
          by(blast dest: rel-trancl-transitive)
      qed
    }
    note tonic = this
    {
      fix Ψ P Q
        and Q′::('a,'b,'c) psi
      assume (Ψ,P,Q) ∈ ?X⋆
        and  Ψ ▷ Q ∼ Q′
      then have (Ψ,P,Q′) ∈ ?X⋆
      proof(induct arbitrary: Q′ rule: rel-trancl.inducts)
        case(r-into-rel-trancl Ψ P Q) then show ?case
          by(blast dest: bisimTransitive)
```

```
    next
      case(rel-trancl-into-rel-trancl Ψ P P' Q)
      from ‹(Ψ, P', Q) ∈ ?X› ‹Ψ ▷ Q ∼ Q'› have (Ψ, P', Q') ∈ ?X
        by(blast dest: bisimTransitive)
      then have (Ψ, P', Q') ∈ ?X*
        by blast
      with ‹(Ψ, P, P') ∈ ?X*›
      show ?case
        by(rule rel-trancl-transitive)
    qed
  }
  note tonic2 = this
  from ‹(Ψ, P', Q') ∈ X› have Ψ ▷ P' ⇝[?X*] Q'
    by(rule rSim)
  with ‹Ψ ▷ P ⇝[bisim] P'› have Ψ ▷ P ⇝[?X*] Q'
    apply −
    apply(rule transitive)
      apply assumption
     apply assumption
    apply(rule rel-trancl-eqvt)
    apply(rule ‹eqvt ?X›)
    by(blast intro: tonic)
  with ‹Ψ ▷ Q' ⇝[bisim] Q› show ?case
    apply −
    apply(rule transitive)
      apply assumption
     apply assumption
    apply(rule rel-trancl-eqvt)
    apply(rule ‹eqvt ?X›)
    by(blast intro: tonic2)
 next
  case(cExt Ψ P Q Ψ')
  then show ?case by(blast dest: bisimE intro: rExt)
 next
  case(cSym Ψ P Q)
  then show ?case
    apply clarsimp
    apply(drule rSym)
    apply clarsimp
    by(metis bisimTransitive bisimE(4))
 qed
qed

lemma transitiveStarCoinduct[case-names cStatEq 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 × ('a, 'b, 'c) psi × ('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) Ψ'
and *rSim*: $\bigwedge \Psi' R S.\ (\Psi', R, S) \in X \Longrightarrow \Psi' \rhd R \rightsquigarrow[(\{(\Psi', R, S) \mid \Psi' R R' S' S.\ \Psi' \rhd R \sim R' \wedge ((\Psi', R', S') \in X \vee \Psi' \rhd R' \sim S') \wedge \Psi' \rhd S' \sim S\})^\star]\ S$
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 −
from p **have** $(\Psi, P, Q) \in (X \cup \mathit{bisim})$
by *blast*
moreover from ‹*eqvt* X› *bisimEqvt* **have** *eqvt* $(X \cup \mathit{bisim})$
by *auto*
ultimately show *?thesis*
proof(*coinduct rule*: *weakTransitiveStarCoinduct′*)
case(*cStatEq* Ψ P Q)
then show *?case*
by(*blast intro*: *rStatEq dest*: *bisimE*)
next
case(*cSim* Ψ P Q)
then show *?case*
apply *clarsimp*
apply (*erule disjE*)
apply(*blast intro*: *rSim*)
apply(*drule bisimE*(*2*))
apply(*rule monotonic*, *simp*)
by(*force intro*: *bisimReflexive*)
next
case(*cExt* Ψ P Q Ψ')
then show *?case*
by(*blast dest*: *bisimE rExt*)
next
case(*cSym* Ψ P Q)
then show *?case* **by**(*blast dest*: *bisimE rSym intro*: *bisimReflexive*)
qed
qed

end

end
theory *Sim-Pres*

imports *Simulation*
begin

This file is a (heavily modified) variant of the theory *Psi_Calculi.Sim_Pres* from [1].

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 −
{
fix α Q'
assume $\Psi \rhd M(\!|\lambda * xvec\ N|\!).Q \longmapsto \alpha \prec Q'$
then have $\exists P'.\ \Psi \rhd M(\!|\lambda * xvec\ N|\!).P \longmapsto \alpha \prec P' \wedge (\Psi, P', Q') \in Rel$
by(*induct rule*: *inputCases*) (*auto intro*: *Input BrInput PRelQ*)
}
then show *?thesis*
by(*auto simp add*: *simulation-def residual.inject psi.inject*)
qed

lemma *outputPres*:
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*: $(\Psi, P, Q) \in Rel$

shows $\Psi \rhd M\langle N\rangle.P \rightsquigarrow[Rel]\ M\langle N\rangle.Q$
proof −
{
fix α Q'
assume $\Psi \rhd M\langle N\rangle.Q \longmapsto \alpha \prec Q'$
then have $\exists P'.\ \Psi \rhd M\langle N\rangle.P \longmapsto \alpha \prec P' \wedge (\Psi, P', Q') \in Rel$
by(*induct rule*: *outputCases*) (*auto intro*: *Output BrOutput PRelQ*)
}

then show *?thesis*
by(*auto simp add*: *simulation-def residual.inject psi.inject*)
qed

lemma *casePres*:
fixes Ψ :: *'b*
and *CsP* :: (*'c* × (*'a*, *'b*, *'c*) *psi*) *list*
and *Rel* :: (*'b* × (*'a*, *'b*, *'c*) *psi* × (*'a*, *'b*, *'c*) *psi*) *set*
and *CsQ* :: (*'c* × (*'a*, *'b*, *'c*) *psi*) *list*
and *M* :: *'a*
and *N* :: *'a*

assumes *PRelQ*: $\bigwedge\varphi\ Q.\ (\varphi, Q) \in set\ CsQ \Longrightarrow \exists P.\ (\varphi, P) \in set\ CsP \wedge guarded\ P \wedge (\Psi, P, Q) \in Rel$
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 −
{
fix $\alpha\ Q'$
assume $\Psi \rhd Cases\ CsQ \longmapsto \alpha \prec Q'$ **and** $bn\ \alpha \sharp* CsP$ **and** $bn\ \alpha \sharp* \Psi$
then have $\exists P'.\ \Psi \rhd Cases\ CsP \longmapsto \alpha \prec P' \wedge (\Psi, P', Q') \in Rel'$
proof(*induct rule*: *caseCases*)
case(*cCase* $\varphi\ Q$)
from ‹$(\varphi, Q) \in set\ CsQ$› **obtain** P **where** $(\varphi, P) \in set\ CsP$ **and** *guarded P* **and** $(\Psi, P, Q) \in Rel$
by(*metis PRelQ*)
from ‹$(\Psi, P, Q) \in Rel$› **have** $\Psi \rhd P \rightsquigarrow[Rel]\ Q$
by(*rule Sim*)
moreover from ‹$bn\ \alpha \sharp* CsP$› ‹$(\varphi, P) \in set\ CsP$› **have** $bn\ \alpha \sharp* P$
by(*auto dest*: *memFreshChain*)
moreover note ‹$\Psi \rhd Q \longmapsto \alpha \prec Q'$› ‹$bn\ \alpha \sharp* \Psi$›
ultimately obtain P' **where** *PTrans*: $\Psi \rhd P \longmapsto \alpha \prec P'$ **and** $P'RelQ'$: $(\Psi, P', Q') \in Rel$
by(*blast dest*: *simE*)
from *PTrans* ‹$(\varphi, P) \in set\ CsP$› ‹$\Psi \vdash \varphi$› ‹*guarded P*› **have** $\Psi \rhd Cases\ CsP \longmapsto \alpha \prec P'$
by(*rule Case*)
moreover from $P'RelQ'$ ‹$Rel \subseteq Rel'$› **have** $(\Psi, P', Q') \in Rel'$
by *blast*
ultimately show *?case*
by *blast*
qed
}
then show *?thesis*
by(*auto simp add*: *simulation-def residual.inject psi.inject*)
qed

lemma *resPres*:
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 x :: *name*
and *Rel'* :: (*'b* × (*'a*, *'b*, *'c*) *psi* × (*'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*: $\bigwedge \Psi' R S yvec.\ [\![(\Psi', R, S) \in Rel; yvec \sharp* \Psi']\!] \Longrightarrow (\Psi', (\!|\nu * yvec|\!)R, (\!|\nu * yvec|\!)S) \in Rel'$

shows $\Psi \rhd (\!|\nu x|\!)P \rightsquigarrow[Rel'] (\!|\nu x|\!)Q$
proof −
note ‹*eqvt Rel'*› ‹$x \sharp \Psi$›
moreover have $x \sharp (\!|\nu x|\!)P$ **and** $x \sharp (\!|\nu x|\!)Q$ **by**(*simp add*: *abs-fresh*)+
ultimately show *?thesis*
proof(*induct rule*: *simIFresh*[**where** C=()])
case(*cSim* α Q')
from ‹$bn\ \alpha \sharp* (\!|\nu x|\!)P$› ‹$bn\ \alpha \sharp* (\!|\nu x|\!)Q$› ‹$x \sharp \alpha$› **have** $bn\ \alpha \sharp* P$ **and** $bn\ \alpha \sharp* Q$ **by** *simp*+
from ‹$\Psi \rhd (\!|\nu x|\!)Q \longmapsto \alpha \prec Q'$› ‹$x \sharp \Psi$› ‹$x \sharp \alpha$› ‹$x \sharp Q'$› ‹$bn\ \alpha \sharp* \Psi$› ‹$bn\ \alpha \sharp* Q$› ‹$bn\ \alpha \sharp*$ *subject* α›
‹$bn\ \alpha \sharp* \Psi$› ‹$bn\ \alpha \sharp* P$› ‹$x \sharp \alpha$›
show *?case*
proof(*induct rule*: *resCases*[**where** C=P])
case(*cOpen M xvec1 xvec2 y N Q'*)
from ‹$bn\ (M(\!|\nu * (xvec1@y\#xvec2)|\!)\langle N\rangle) \sharp* \Psi$› **have** *xvec1* $\sharp* \Psi$ **and** $y \sharp \Psi$ **and** *xvec2* $\sharp* \Psi$ **by** *simp*+
from ‹$bn\ (M(\!|\nu * (xvec1@y\#xvec2)|\!)\langle N\rangle) \sharp* P$› **have** *xvec1* $\sharp* P$ **and** $y \sharp P$ **and** *xvec2* $\sharp* P$ **by** *simp*+
from ‹$x \sharp (M(\!|\nu * (xvec1@y\#xvec2)|\!)\langle N\rangle)$› **have** $x \sharp$ *xvec1* **and** $x \neq y$ **and** $x \sharp$ *xvec2* **and** $x \sharp M$ **by** *simp*+
from *PSimQ* ‹$\Psi \rhd Q \longmapsto M(\!|\nu * (xvec1@xvec2)|\!)\langle([(x, y)] \cdot N)\rangle \prec ([(x, y)] \cdot Q')$›
‹*xvec1* $\sharp* \Psi$› ‹*xvec2* $\sharp* \Psi$› ‹*xvec1* $\sharp* P$› ‹*xvec2* $\sharp* P$›
obtain P' **where** *PTrans*: $\Psi \rhd P \longmapsto M(\!|\nu * (xvec1@xvec2)|\!)\langle([(x, y)] \cdot N)\rangle \prec P'$ **and** $P'RelQ'$: $(\Psi, P', ([(x, y)] \cdot Q')) \in Rel$
by(*force dest*: *simE*)
from ‹$y \in supp\ N$› ‹$x \neq y$› **have** $x \in supp([(x, y)] \cdot N)$
apply −
apply(*drule pt-set-bij2*[*OF pt-name-inst*, *OF at-name-inst*, **where** pi=[(*x*, *y*)]])
by(*simp add*: *eqvts calc-atm*)
with *PTrans* ‹$x \sharp \Psi$› ‹$x \sharp M$› ‹$x \sharp$ *xvec1*› ‹$x \sharp$ *xvec2*›
have $\Psi \rhd (\!|\nu x|\!)P \longmapsto M(\!|\nu * (xvec1@x\#xvec2)|\!)\langle([(x, y)] \cdot N)\rangle \prec P'$

by(*metis Open*)
then have $([(x, y)] \cdot \Psi) \rhd ([(x, y)] \cdot (\langle\!|\nu x|\!\rangle P) \longmapsto ([(x, y)] \cdot (M\langle\!|\nu*(xvec1@x\#xvec2)|\!\rangle\langle([(x, y)] \cdot N)\rangle \prec P'))$
by(*rule eqvts*)
with ‹$x \sharp \Psi$› ‹$y \sharp \Psi$› ‹$y \sharp P$› ‹$x \sharp M$› ‹$y \sharp M$› ‹$x \sharp xvec1$› ‹$y \sharp xvec1$› ‹$x \sharp xvec2$› ‹$y \sharp xvec2$› ‹$x \neq y$›
have $\Psi \rhd (\langle\!|\nu x|\!\rangle P \longmapsto M\langle\!|\nu*(xvec1@y\#xvec2)|\!\rangle\langle N\rangle \prec ([(x, y)] \cdot P')$ **by**(*simp add*: *eqvts calc-atm alphaRes*)
moreover from $P'RelQ'$ ‹$Rel \subseteq Rel'$› ‹*eqvt Rel'*› **have** $([(y, x)] \cdot \Psi, [(y, x)] \cdot P', [(y, x)] \cdot [(x, y)] \cdot Q') \in Rel'$
by(*force simp add*: *eqvt-def*)
with ‹$x \sharp \Psi$› ‹$y \sharp \Psi$› **have** $(\Psi, [(x, y)] \cdot P', Q') \in Rel'$ **by**(*simp add*: *name-swap*)
ultimately show *?case* **by** *blast*
next
case(*cBrOpen M xvec1 xvec2 y N Q'*)
from ‹$bn\ (¡M\langle\!|\nu*(xvec1@y\#xvec2)|\!\rangle\langle N\rangle) \sharp* \Psi$› **have** $xvec1 \sharp* \Psi$ **and** $y \sharp \Psi$ **and** $xvec2 \sharp* \Psi$ **by** *simp+*
from ‹$bn\ (¡M\langle\!|\nu*(xvec1@y\#xvec2)|\!\rangle\langle N\rangle) \sharp* P$› **have** $xvec1 \sharp* P$ **and** $y \sharp P$ **and** $xvec2 \sharp* P$ **by** *simp+*
from ‹$x \sharp (¡M\langle\!|\nu*(xvec1@y\#xvec2)|\!\rangle\langle N\rangle)$›
have $x \sharp xvec1$ **and** $x \neq y$ **and** $x \sharp xvec2$ **and** $x \sharp M$ **by** *simp+*
from $PSimQ$ ‹$\Psi \rhd Q \longmapsto ¡M\langle\!|\nu*(xvec1@xvec2)|\!\rangle\langle([(x, y)] \cdot N)\rangle \prec ([(x, y)] \cdot Q')$›
‹$xvec1 \sharp* \Psi$› ‹$xvec2 \sharp* \Psi$› ‹$xvec1 \sharp* P$› ‹$xvec2 \sharp* P$›
obtain P' **where** $PTrans$: $\Psi \rhd P \longmapsto ¡M\langle\!|\nu*(xvec1@xvec2)|\!\rangle\langle([(x, y)] \cdot N)\rangle \prec P'$ **and** $P'RelQ'$: $(\Psi, P', ([(x, y)] \cdot Q')) \in Rel$
by(*force dest*: *simE*)
from ‹$y \in supp\ N$› ‹$x \neq y$› **have** $x \in supp([(x, y)] \cdot N)$
apply −
apply(*drule pt-set-bij2*[*OF pt-name-inst*, *OF at-name-inst*, **where** *pi*=[(*x*, *y*)]])
by(*simp add*: *eqvts calc-atm*)
with *PTrans* ‹$x \sharp \Psi$› ‹$x \sharp M$› ‹$x \sharp xvec1$› ‹$x \sharp xvec2$›
have $\Psi \rhd (\langle\!|\nu x|\!\rangle P \longmapsto ¡M\langle\!|\nu*(xvec1@x\#xvec2)|\!\rangle\langle([(x, y)] \cdot N)\rangle \prec P'$
by(*metis BrOpen*)
then have $([(x, y)] \cdot \Psi) \rhd ([(x, y)] \cdot (\langle\!|\nu x|\!\rangle P) \longmapsto ([(x, y)] \cdot (¡M\langle\!|\nu*(xvec1@x\#xvec2)|\!\rangle\langle([(x, y)] \cdot N)\rangle \prec P'))$
by(*rule eqvts*)
with ‹$x \sharp \Psi$› ‹$y \sharp \Psi$› ‹$y \sharp P$› ‹$x \sharp M$› ‹$y \sharp M$› ‹$x \sharp xvec1$› ‹$y \sharp xvec1$› ‹$x \sharp xvec2$› ‹$y \sharp xvec2$› ‹$x \neq y$›
have $\Psi \rhd (\langle\!|\nu x|\!\rangle P \longmapsto ¡M\langle\!|\nu*(xvec1@y\#xvec2)|\!\rangle\langle N\rangle \prec ([(x, y)] \cdot P')$ **by**(*simp add*: *eqvts calc-atm alphaRes*)
moreover from $P'RelQ'$ ‹$Rel \subseteq Rel'$› ‹*eqvt Rel'*› **have** $([(y, x)] \cdot \Psi, [(y, x)] \cdot P', [(y, x)] \cdot [(x, y)] \cdot Q') \in Rel'$
by(*force simp add*: *eqvt-def*)
with ‹$x \sharp \Psi$› ‹$y \sharp \Psi$› **have** $(\Psi, [(x, y)] \cdot P', Q') \in Rel'$ **by**(*simp add*: *name-swap*)
ultimately show *?case* **by** *blast*

next
case(*cRes* Q')
from *PSimQ* ‹$\Psi \rhd Q \longmapsto \alpha \prec Q'$› ‹*bn* $\alpha \sharp* \Psi$› ‹*bn* $\alpha \sharp* P$›
obtain P' **where** *PTrans*: $\Psi \rhd P \longmapsto \alpha \prec P'$ **and** $P'RelQ'$: $(\Psi, P', Q') \in Rel$
by(*blast dest*: *simE*)
from *PTrans* ‹$x \sharp \Psi$› ‹$x \sharp \alpha$› **have** $\Psi \rhd (\!|\nu x|\!) P \longmapsto \alpha \prec (\!|\nu x|\!) P'$
by(*rule Scope*)
moreover from $P'RelQ'$ ‹$x \sharp \Psi$› **have** $(\Psi, (\!|\nu*[x]|\!) P', (\!|\nu*[x]|\!) Q') \in Rel'$
by(*intro C1*[**where** *yvec*=[*x*]]) *simp*+
moreover then have $(\Psi, (\!|\nu x|\!) P', (\!|\nu x|\!) Q') \in Rel'$
by (*metis resChain.base resChain.step*)
ultimately show *?case* **by** *blast*
next
case(*cBrClose M xvec N Q'*)
from *PSimQ* ‹$\Psi \rhd Q \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec Q'$› ‹*xvec* $\sharp* \Psi$› ‹*xvec* $\sharp* P$›
obtain P' **where** *PTrans*: $\Psi \rhd P \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec P'$ **and** $P'RelQ'$: $(\Psi, P', Q') \in Rel$
by(*force dest*: *simE*)
with ‹$x \sharp \Psi$› ‹*xvec* $\sharp* \Psi$›
have $(x\#xvec) \sharp* \Psi$ **by** *simp*

from $P'RelQ'$
have $(\Psi, (\!|\nu*(x\#xvec)|\!) P', (\!|\nu*(x\#xvec)|\!) Q') \in Rel'$
using ‹$(x\#xvec) \sharp* \Psi$›
by(*rule C1*)
then have $(\Psi, (\!|\nu x|\!)((\!|\nu*xvec|\!) P'), (\!|\nu x|\!)((\!|\nu*xvec|\!) Q')) \in Rel'$
by *simp*

moreover from ‹$\Psi \rhd P \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec P'$› ‹$x \in supp\ M$› ‹$x \sharp \Psi$›
have $\Psi \rhd (\!|\nu x|\!) P \longmapsto \tau \prec (\!|\nu x|\!)((\!|\nu*xvec|\!) P')$
by(*rule BrClose*)
ultimately show *?case*
by *blast*
qed
qed
qed

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*: $\bigwedge \Psi' R S\ yvec.\ [\![(\Psi', R, S) \in Rel; yvec \sharp* \Psi']\!] \Longrightarrow (\Psi', (\!|\nu*yvec|\!) R,$

$(\!|\nu{*}yvec|\!)S) \in Rel$

shows $\Psi \rhd (\!|\nu{*}xvec|\!)P \rightsquigarrow[Rel]\ (\!|\nu{*}xvec|\!)Q$
using ‹*xvec* $\sharp*$ Ψ›
proof(*induct xvec*)
case *Nil*
from *PSimQ* **show** *?case* **by** *simp*
next
case(*Cons x xvec*)
from ‹(*x*#*xvec*) $\sharp*$ Ψ› **have** $x \sharp \Psi$ **and** *xvec* $\sharp*$ Ψ **by** *simp*+
from ‹*xvec* $\sharp*$ Ψ› **have** $\Psi \rhd (\!|\nu{*}xvec|\!)P \rightsquigarrow[Rel]\ (\!|\nu{*}xvec|\!)Q$ **by**(*rule Cons*)
moreover note ‹*eqvt Rel*› ‹$x \sharp \Psi$›
moreover have $Rel \subseteq Rel$ **by** *simp*
ultimately have $\Psi \rhd (\!|\nu x|\!)((\!|\nu{*}xvec|\!)P) \rightsquigarrow[Rel]\ (\!|\nu x|\!)((\!|\nu{*}xvec|\!)Q)$ **using** *C1*
by(*rule resPres*)
then show *?case* **by** *simp*
qed

lemma *parPres*:
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 R :: (*'a*, *'b*, *'c*) *psi*
and *Rel'* :: (*'b* × (*'a*, *'b*, *'c*) *psi* × (*'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 $\Psi \rhd P \parallel R \rightsquigarrow[Rel']\ Q \parallel R$
using *Eqvt'*
proof(*induct rule*: *simI*[*of* - - - - ()])
case(*cSim* α *QR*)
from ‹*bn* α $\sharp*$ ($P \parallel R$)› ‹*bn* α $\sharp*$ ($Q \parallel R$)›
have *bn* α $\sharp*$ P **and** *bn* α $\sharp*$ Q **and** *bn* α $\sharp*$ R

by *simp+*
from ‹$\Psi \rhd Q \parallel R \longmapsto \alpha \prec QR$› ‹*bn* $\alpha \sharp* \Psi$› ‹*bn* $\alpha \sharp* Q$› ‹*bn* $\alpha \sharp* R$› ‹*bn* $\alpha \sharp*$ *subject* α›
show *?case*
proof(*induct rule*: *parCases*[**where** $C = (P, R)$])
case(*cPar1* Q' A_R Ψ_R)
from ‹$A_R \sharp* (P, R)$› **have** $A_R \sharp* P$ **by** *simp*
have *FrR*: *extractFrame* $R = \langle A_R, \Psi_R \rangle$ **by** *fact*
from ‹$A_R \sharp* \alpha$› ‹*bn* $\alpha \sharp* R$› *FrR* **have** *bn* $\alpha \sharp* \Psi_R$
by(*auto dest*: *extractFrameFreshChain*)
from *FrR* ‹$A_R \sharp* \Psi$› ‹$A_R \sharp* P$› ‹$A_R \sharp* Q$› **have** $\Psi \otimes \Psi_R \rhd P \rightsquigarrow [Rel]\ Q$
by(*blast intro*: *Sim PRelQ*)
moreover have *QTrans*: $\Psi \otimes \Psi_R \rhd Q \longmapsto \alpha \prec Q'$ **by** *fact*
ultimately obtain P' **where** *PTrans*: $\Psi \otimes \Psi_R \rhd P \longmapsto \alpha \prec P'$
and $P'RelQ'$: $(\Psi \otimes \Psi_R, P', Q') \in Rel$
using ‹*bn* $\alpha \sharp* \Psi$› ‹*bn* $\alpha \sharp* \Psi_R$› ‹*bn* $\alpha \sharp* P$›
by(*force dest*: *simE*)
from *PTrans QTrans* ‹$A_R \sharp* P$› ‹$A_R \sharp* Q$› ‹$A_R \sharp* \alpha$› ‹*bn* $\alpha \sharp*$ *subject* α› ‹*distinct*(*bn* α)› **have** $A_R \sharp* P'$ **and** $A_R \sharp* Q'$
by(*blast dest*: *freeFreshChainDerivative*)+
from *PTrans* ‹*bn* $\alpha \sharp* R$› *FrR* ‹$A_R \sharp* \Psi$› ‹$A_R \sharp* P$› ‹$A_R \sharp* \alpha$› **have** $\Psi \rhd P \parallel R \longmapsto \alpha \prec (P' \parallel R)$
by(*metis Par1*)
moreover from $P'RelQ'$ *FrR* ‹$A_R \sharp* \Psi$› ‹$A_R \sharp* P'$› ‹$A_R \sharp* Q'$› **have** $(\Psi, P' \parallel R, Q' \parallel R) \in Rel'$ **by**(*rule C1*)
ultimately show *?case* **by** *blast*
next
case(*cPar2* R' A_Q Ψ_Q)
from ‹$A_Q \sharp* (P, R)$› **have** $A_Q \sharp* P$ **and** $A_Q \sharp* R$ **by** *simp+*
obtain A_P Ψ_P **where** *FrP*: *extractFrame* $P = \langle A_P, \Psi_P \rangle$ **and** $A_P \sharp* (\Psi, A_Q, \Psi_Q, \alpha, R)$
by(*rule freshFrame*)
then have $A_P \sharp* \Psi$ **and** $A_P \sharp* A_Q$ **and** $A_P \sharp* \Psi_Q$ **and** $A_P \sharp* \alpha$ **and** $A_P \sharp* R$
by *simp+*

have *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$ **by** *fact*
from ‹$A_Q \sharp* P$› *FrP* ‹$A_P \sharp* A_Q$› **have** $A_Q \sharp* \Psi_P$
by(*auto dest*: *extractFrameFreshChain*)

from *FrP FrQ* ‹*bn* $\alpha \sharp* P$› ‹*bn* $\alpha \sharp* Q$› ‹$A_P \sharp* \alpha$› ‹$A_Q \sharp* \alpha$›
have *bn* $\alpha \sharp* \Psi_P$ **and** *bn* $\alpha \sharp* \Psi_Q$
by(*force dest*: *extractFrameFreshChain*)+

obtain A_R Ψ_R **where** *FrR*: *extractFrame* $R = \langle A_R, \Psi_R \rangle$ **and** $A_R \sharp* (\Psi, P, Q, A_Q, A_P, \Psi_Q, \Psi_P, \alpha, R)$ **and** *distinct* A_R
by(*rule freshFrame*)
then have $A_R \sharp* \Psi$ **and** $A_R \sharp* P$ **and** $A_R \sharp* Q$ **and** $A_R \sharp* A_Q$ **and** $A_R \sharp* A_P$ **and** $A_R \sharp* \Psi_Q$ **and** $A_R \sharp* \Psi_P$ **and** $A_R \sharp* \alpha$ **and** $A_R \sharp* R$

by *simp+*

from ‹$A_Q \sharp* R$› *FrR* ‹$A_R \sharp* A_Q$› **have** $A_Q \sharp* \Psi_R$
by(*auto dest*: *extractFrameFreshChain*)
from ‹$A_P \sharp* R$› ‹$A_R \sharp* A_P$› *FrR* **have** $A_P \sharp* \Psi_R$
by(*auto dest*: *extractFrameFreshChain*)

have *RTrans*: $\Psi \otimes \Psi_Q \rhd R \longmapsto \alpha \prec R'$ **by** *fact*
moreover have $\langle A_Q, (\Psi \otimes \Psi_Q) \otimes \Psi_R\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R\rangle$
proof −
have $\langle A_Q, (\Psi \otimes \Psi_Q) \otimes \Psi_R\rangle \simeq_F \langle A_Q, (\Psi \otimes \Psi_R) \otimes \Psi_Q\rangle$
by(*metis frameIntAssociativity Commutativity FrameStatEqTrans frameIntCompositionSym FrameStatEqSym*)
moreover from *FrR* ‹$A_R \sharp* \Psi$› ‹$A_R \sharp* P$› ‹$A_R \sharp* Q$›
have (*insertAssertion* (*extractFrame Q*) ($\Psi \otimes \Psi_R$)) $\hookrightarrow_F$ (*insertAssertion* (*extractFrame P*) ($\Psi \otimes \Psi_R$))
by(*blast intro*: *PRelQ StatImp*)
with *FrP FrQ* ‹$A_P \sharp* \Psi$› ‹$A_Q \sharp* \Psi$› ‹$A_P \sharp* \Psi_R$› ‹$A_Q \sharp* \Psi_R$›
have $\langle A_Q, (\Psi \otimes \Psi_R) \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_R) \otimes \Psi_P\rangle$ **using** *freshCompChain* **by** *auto*
moreover have $\langle A_P, (\Psi \otimes \Psi_R) \otimes \Psi_P\rangle \simeq_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R\rangle$
by(*metis frameIntAssociativity Commutativity FrameStatEqTrans frameIntCompositionSym frameIntAssociativity*[*THEN FrameStatEqSym*])
ultimately show *?thesis*
by(*rule FrameStatEqImpCompose*)
qed

ultimately have $\Psi \otimes \Psi_P \rhd R \longmapsto \alpha \prec R'$
using ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* \Psi_Q$› ‹$A_Q \sharp* \Psi$› ‹$A_Q \sharp* \Psi_P$› ‹$A_P \sharp* R$› ‹$A_Q \sharp* R$› ‹$A_P \sharp* \alpha$› ‹$A_Q \sharp* \alpha$›
‹$A_R \sharp* A_P$› ‹$A_R \sharp* A_Q$› ‹$A_R \sharp* \Psi_P$› ‹$A_R \sharp* \Psi_Q$› ‹$A_R \sharp* \Psi$› *FrR* ‹*distinct* A_R›
by(*force intro*: *transferFrame*)
with ‹*bn* $\alpha \sharp* P$› ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* R$› ‹$A_P \sharp* \alpha$› *FrP* **have** $\Psi \rhd P \parallel R \longmapsto \alpha \prec (P \parallel R')$
by(*force intro*: *Par2*)
moreover obtain A_R' Ψ_R' **where** *extractFrame* $R' = \langle A_R', \Psi_R'\rangle$ **and** $A_R' \sharp* \Psi$ **and** $A_R' \sharp* P$ **and** $A_R' \sharp* Q$
by(*rule freshFrame*[**where** C=(Ψ, P, Q)]) *auto*

moreover from *RTrans FrR* ‹*distinct* A_R› ‹$A_R \sharp* \Psi$› ‹$A_R \sharp* P$› ‹$A_R \sharp* Q$› ‹$A_R \sharp* R$› ‹$A_R \sharp* \alpha$› ‹*bn* $\alpha \sharp* \Psi$› ‹*bn* $\alpha \sharp* P$› ‹*bn* $\alpha \sharp* Q$› ‹*bn* $\alpha \sharp* R$› ‹*bn* $\alpha \sharp*$ *subject* α› ‹*distinct*(*bn* α)›
obtain p Ψ' A_R' Ψ_R' **where** S: *set* $p \subseteq$ *set*(*bn* α) $\times$ *set*(*bn*($p \cdot \alpha$)) **and** ($p \cdot \Psi_R$) $\otimes$ $\Psi' \simeq \Psi_R'$ **and** *FrR'*: *extractFrame* $R' = \langle A_R', \Psi_R'\rangle$
and *bn*($p \cdot \alpha$) $\sharp* R$ **and** *bn*($p \cdot \alpha$) $\sharp* \Psi$ **and** *bn*($p \cdot \alpha$) $\sharp* P$ **and** *bn*($p \cdot \alpha$) $\sharp* Q$ **and** *bn*($p \cdot \alpha$) $\sharp* R$
and $A_R' \sharp* \Psi$ **and** $A_R' \sharp* P$ **and** $A_R' \sharp* Q$
apply −

apply (*rule expandFrame*[**where** C=(Ψ, P, Q, R) **and** C'=(Ψ, P, Q, R)])
apply *simp+*
done

from ‹A_R ♯* Ψ› **have** (p ∙ A_R) ♯* (p ∙ Ψ) **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹*bn* α ♯* Ψ› ‹*bn*(p ∙ α) ♯* Ψ› S **have** (p ∙ A_R) ♯* Ψ **by** *simp*
from ‹A_R ♯* P› **have** (p ∙ A_R) ♯* (p ∙ P) **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹*bn* α ♯* P› ‹*bn*(p ∙ α) ♯* P› S **have** (p ∙ A_R) ♯* P **by** *simp*
from ‹A_R ♯* Q› **have** (p ∙ A_R) ♯* (p ∙ Q) **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹*bn* α ♯* Q› ‹*bn*(p ∙ α) ♯* Q› S **have** (p ∙ A_R) ♯* Q **by** *simp*

from *FrR* **have** (p ∙ *extractFrame* R) = p ∙ ⟨A_R, Ψ_R⟩ **by** *simp*
with ‹*bn* α ♯* R› ‹*bn*(p ∙ α) ♯* R› S **have** *extractFrame* R = ⟨(p ∙ A_R), (p ∙ Ψ_R)⟩
by(*simp add*: *eqvts*)

with ‹(p ∙ A_R) ♯* Ψ› ‹(p ∙ A_R) ♯* P› ‹(p ∙ A_R) ♯* Q› **have** (Ψ ⊗ (p ∙ Ψ_R), P, Q) ∈ *Rel*
by(*metis PRelQ*)

then have ((Ψ ⊗ (p ∙ Ψ_R)) ⊗ Ψ', P, Q) ∈ *Rel* **by**(*rule Ext*)
with ‹(p ∙ Ψ_R) ⊗ Ψ' ≃ Ψ_R'› **have** (Ψ ⊗ Ψ_R', P, Q) ∈ *Rel* **by**(*blast intro*: *C3 Associativity compositionSym*)
with *FrR'* ‹A_R' ♯* Ψ› ‹A_R' ♯* P› ‹A_R' ♯* Q› **have** (Ψ, P ∥ R', Q ∥ R') ∈ *Rel'*
by(*metis C1*)
ultimately show *?case* **by** *blast*
next
case(*cComm1* Ψ_R M N Q' A_Q Ψ_Q K *xvec* R' A_R)
have *FrQ*: *extractFrame* Q = ⟨A_Q, Ψ_Q⟩ **by** *fact*
from ‹A_Q ♯* (P, R)› **have** A_Q ♯* P **and** A_Q ♯* R **by** *simp+*

have *FrR*: *extractFrame* R = ⟨A_R, Ψ_R⟩ **by** *fact*
from ‹A_R ♯* (P, R)› **have** A_R ♯* P **and** A_R ♯* R **by** *simp+*

from ‹*xvec* ♯* (P, R)› **have** *xvec* ♯* P **and** *xvec* ♯* R **by** *simp+*

obtain A_P Ψ_P **where** *FrP*: *extractFrame* P = ⟨A_P, Ψ_P⟩ **and** A_P ♯* (Ψ, A_Q, Ψ_Q, A_R, M, N, K, R, P, *xvec*) **and** *distinct* A_P
by(*rule freshFrame*)
then have A_P ♯* Ψ **and** A_P ♯* A_Q **and** A_P ♯* Ψ_Q **and** A_P ♯* M **and** A_P ♯* R
and A_P ♯* N **and** A_P ♯* K **and** A_P ♯* A_R **and** A_P ♯* P **and** A_P ♯* *xvec*
by *simp+*

have *QTrans*: Ψ ⊗ Ψ_R ▷ Q ⟼M⦇N⦈ ≺ Q' **and** *RTrans*: Ψ ⊗ Ψ_Q ▷ R ⟼K⦇ν∗*xvec*⦈⟨N⟩ ≺ R'

and *MeqK*: $\Psi \otimes \Psi_Q \otimes \Psi_R \vdash M \leftrightarrow K$ **by** *fact+*

from *FrP FrR* ‹$A_Q \sharp* P$› ‹$A_P \sharp* R$› ‹$A_R \sharp* P$› ‹$A_P \sharp* A_Q$› ‹$A_P \sharp* A_R$› ‹$A_P \sharp* xvec$› ‹$xvec \sharp* P$›
have $A_P \sharp* \Psi_R$ **and** $A_Q \sharp* \Psi_P$ **and** $A_R \sharp* \Psi_P$ **and** $xvec \sharp* \Psi_P$
by(*auto dest*!: *extractFrameFreshChain*)+

from *RTrans FrR* ‹*distinct* A_R› ‹$A_R \sharp* R$› ‹$A_R \sharp* xvec$› ‹$xvec \sharp* R$› ‹$xvec \sharp* Q$› ‹$xvec \sharp* \Psi$› ‹$xvec \sharp* \Psi_Q$› ‹$A_R \sharp* Q$›
‹$A_R \sharp* \Psi$› ‹$A_R \sharp* \Psi_Q$› ‹$xvec \sharp* K$› ‹$A_R \sharp* K$› ‹$A_R \sharp* N$› ‹$A_R \sharp* R$› ‹$xvec \sharp* R$› ‹$A_R \sharp* P$› ‹$xvec \sharp* P$› ‹$A_P \sharp* A_R$› ‹$A_P \sharp* xvec$›
‹$A_Q \sharp* A_R$› ‹$A_Q \sharp* xvec$› ‹$A_R \sharp* \Psi_P$› ‹$xvec \sharp* \Psi_P$› ‹*distinct xvec*› ‹$xvec \sharp* M$›
obtain p Ψ' A_R' Ψ_R' **where** S: *set* $p \subseteq$ *set* $xvec \times$ *set*$(p \cdot xvec)$ **and** *FrR'*: *extractFrame* $R' = \langle A_R', \Psi_R' \rangle$
and $(p \cdot \Psi_R) \otimes \Psi' \simeq \Psi_R'$ **and** $A_R' \sharp* Q$ **and** $A_R' \sharp* \Psi$ **and** $(p \cdot xvec) \sharp* \Psi$
and $(p \cdot xvec) \sharp* Q$ **and** $(p \cdot xvec) \sharp* \Psi_Q$ **and** $(p \cdot xvec) \sharp* K$ **and** $(p \cdot xvec) \sharp* R$
and $(p \cdot xvec) \sharp* P$ **and** $(p \cdot xvec) \sharp* A_P$ **and** $(p \cdot xvec) \sharp* A_Q$ **and** $(p \cdot xvec) \sharp* \Psi_P$
and $A_R' \sharp* P$ **and** $A_R' \sharp* N$
by (*subst expandFrame*[**where** C=(Ψ, Q, Ψ_Q, K, R, P, A_P, A_Q, Ψ_P) **and** C'=(Ψ, Q, Ψ_Q, K, R, P, A_P, A_Q, Ψ_P)]) *simp+*

from ‹$A_R \sharp* \Psi$› **have** $(p \cdot A_R) \sharp* (p \cdot \Psi)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$xvec \sharp* \Psi$› ‹$(p \cdot xvec) \sharp* \Psi$› S **have** $(p \cdot A_R) \sharp* \Psi$ **by** *simp*
from ‹$A_R \sharp* P$› **have** $(p \cdot A_R) \sharp* (p \cdot P)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$xvec \sharp* P$› ‹$(p \cdot xvec) \sharp* P$› S **have** $(p \cdot A_R) \sharp* P$ **by** *simp*
from ‹$A_R \sharp* Q$› **have** $(p \cdot A_R) \sharp* (p \cdot Q)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$xvec \sharp* Q$› ‹$(p \cdot xvec) \sharp* Q$› S **have** $(p \cdot A_R) \sharp* Q$ **by** *simp*
from ‹$A_R \sharp* R$› **have** $(p \cdot A_R) \sharp* (p \cdot R)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$xvec \sharp* R$› ‹$(p \cdot xvec) \sharp* R$› S **have** $(p \cdot A_R) \sharp* R$ **by** *simp*
from ‹$A_R \sharp* K$› **have** $(p \cdot A_R) \sharp* (p \cdot K)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$xvec \sharp* K$› ‹$(p \cdot xvec) \sharp* K$› S **have** $(p \cdot A_R) \sharp* K$ **by** *simp*

from ‹$A_P \sharp* xvec$› ‹$(p \cdot xvec) \sharp* A_P$› ‹$A_P \sharp* M$› S **have** $A_P \sharp* (p \cdot M)$ **by**(*simp add*: *freshChainSimps*)
from ‹$A_Q \sharp* xvec$› ‹$(p \cdot xvec) \sharp* A_Q$› ‹$A_Q \sharp* M$› S **have** $A_Q \sharp* (p \cdot M)$ **by**(*simp add*: *freshChainSimps*)
from ‹$A_P \sharp* xvec$› ‹$(p \cdot xvec) \sharp* A_P$› ‹$A_P \sharp* A_R$› S **have** $(p \cdot A_R) \sharp* A_P$ **by**(*simp add*: *freshChainSimps*)
from ‹$A_Q \sharp* xvec$› ‹$(p \cdot xvec) \sharp* A_Q$› ‹$A_Q \sharp* A_R$› S **have** $(p \cdot A_R) \sharp* A_Q$ **by**(*simp add*: *freshChainSimps*)

from *QTrans S* ‹*xvec* ♯* *Q*› ‹(*p* · *xvec*) ♯* *Q*› **have** (*p* · (Ψ ⊗ Ψ$_R$)) ▷ *Q* ⟼ (*p* · *M*)⦇*N*⦈ ≺ *Q*′
using *inputPermFrameSubject name-list-set-fresh* **by** *blast*
with ‹*xvec* ♯* Ψ› ‹(*p* · *xvec*) ♯* Ψ› *S* **have** *QTrans*: (Ψ ⊗ (*p* · Ψ$_R$)) ▷ *Q* ⟼ (*p* · *M*)⦇*N*⦈ ≺ *Q*′
by(*simp add*: *eqvts*)

from *FrR* **have** (*p* · *extractFrame R*) = *p* · ⟨*A*$_R$, Ψ$_R$⟩ **by** *simp*
with ‹*xvec* ♯* *R*› ‹(*p* · *xvec*) ♯* *R*› *S* **have** *FrR*: *extractFrame R* = ⟨(*p* · *A*$_R$), (*p* · Ψ$_R$)⟩
by(*simp add*: *eqvts*)

note *RTrans FrR*
moreover from *FrR* ‹(*p* · *A*$_R$) ♯* Ψ› ‹(*p* · *A*$_R$) ♯* *P*› ‹(*p* · *A*$_R$) ♯* *Q*› **have** Ψ ⊗ (*p* · Ψ$_R$) ▷ *P* ⇝[*Rel*] *Q*
by(*metis Sim PRelQ*)
with *QTrans* **obtain** *P*′ **where** *PTrans*: Ψ ⊗ (*p* · Ψ$_R$) ▷ *P* ⟼(*p* · *M*)⦇*N*⦈ ≺ *P*′ **and** *P*′*RelQ*′: (Ψ ⊗ (*p* · Ψ$_R$), *P*′, *Q*′) ∈ *Rel*
by(*force dest*: *simE*)
from *PTrans QTrans* ‹*A*$_R$′ ♯* *P*› ‹*A*$_R$′ ♯* *Q*› ‹*A*$_R$′ ♯* *N*› **have** *A*$_R$′ ♯* *P*′ **and** *A*$_R$′ ♯* *Q*′
by(*blast dest*: *inputFreshChainDerivative*)+

note *PTrans*
moreover from *MeqK* **have** (*p* · (Ψ ⊗ Ψ$_Q$ ⊗ Ψ$_R$)) ⊢ (*p* · *M*) ↔ (*p* · *K*) **by**(*rule chanEqClosed*)
with ‹*xvec* ♯* Ψ› ‹(*p* · *xvec*) ♯* Ψ› ‹*xvec* ♯* Ψ$_Q$› ‹(*p* · *xvec*) ♯* Ψ$_Q$› ‹*xvec* ♯* *K*› ‹(*p* · *xvec*) ♯* *K*› *S*
have *MeqK*: Ψ ⊗ Ψ$_Q$ ⊗ (*p* · Ψ$_R$) ⊢ (*p* · *M*) ↔ *K* **by**(*simp add*: *eqvts*)

moreover have ⟨*A*$_Q$, (Ψ ⊗ Ψ$_Q$) ⊗ (*p* · Ψ$_R$)⟩ ↪$_F$ ⟨*A*$_P$, (Ψ ⊗ Ψ$_P$) ⊗ (*p* · Ψ$_R$)⟩
proof −
have ⟨*A*$_P$, (Ψ ⊗ (*p* · Ψ$_R$)) ⊗ Ψ$_P$⟩ ≃$_F$ ⟨*A*$_P$, (Ψ ⊗ Ψ$_P$) ⊗ (*p* · Ψ$_R$)⟩
by(*metis frameResChainPres frameNilStatEq Commutativity AssertionStatEqTrans Composition Associativity*)
moreover from *FrR* ‹(*p* · *A*$_R$) ♯* Ψ› ‹(*p* · *A*$_R$) ♯* *P*› ‹(*p* · *A*$_R$) ♯* *Q*›
have (*insertAssertion* (*extractFrame Q*) (Ψ ⊗ (*p* · Ψ$_R$))) ↪$_F$ (*insertAssertion* (*extractFrame P*) (Ψ ⊗ (*p* · Ψ$_R$)))
by(*metis PRelQ StatImp*)
with *FrP FrQ* ‹*A*$_P$ ♯* Ψ› ‹*A*$_Q$ ♯* Ψ› ‹*A*$_P$ ♯* Ψ$_R$› ‹*A*$_Q$ ♯* Ψ$_R$› ‹*A*$_P$ ♯* *xvec*› ‹(*p* · *xvec*) ♯* *A*$_P$› ‹*A*$_Q$ ♯* *xvec*› ‹(*p* · *xvec*) ♯* *A*$_Q$› *S*
have ⟨*A*$_Q$, (Ψ ⊗ (*p* · Ψ$_R$)) ⊗ Ψ$_Q$⟩ ↪$_F$ ⟨*A*$_P$, (Ψ ⊗ (*p* · Ψ$_R$)) ⊗ Ψ$_P$⟩ **using** *freshCompChain*
by(*simp add*: *freshChainSimps*)
moreover have ⟨*A*$_Q$, (Ψ ⊗ Ψ$_Q$) ⊗ (*p* · Ψ$_R$)⟩ ≃$_F$ ⟨*A*$_Q$, (Ψ ⊗ (*p* · Ψ$_R$)) ⊗ Ψ$_Q$⟩
by(*metis frameResChainPres frameNilStatEq Commutativity AssertionStatEqTrans Composition Associativity*)
ultimately show *?thesis*

by(*metis FrameStatEqImpCompose*)
qed
moreover note *FrP FrQ* ‹*distinct* A_P›
moreover from ‹*distinct* A_R› **have** *distinct*$(p \cdot A_R)$ **by** *simp*
moreover note ‹$(p \cdot A_R) \sharp* A_P$› ‹$(p \cdot A_R) \sharp* A_Q$› ‹$(p \cdot A_R) \sharp* \Psi$› ‹$(p \cdot A_R) \sharp* P$› ‹$(p \cdot A_R) \sharp* Q$› ‹$(p \cdot A_R) \sharp* R$› ‹$(p \cdot A_R) \sharp* K$›
‹$A_P \sharp* \Psi$› ‹$A_P \sharp* R$› ‹$A_P \sharp* P$› ‹$A_P \sharp* (p \cdot M)$› ‹$A_Q \sharp* R$› ‹$A_Q \sharp* (p \cdot M)$› ‹$A_P \sharp* xvec$› ‹$xvec \sharp* P$› ‹$A_P \sharp* R$›
ultimately obtain K' **where** $\Psi \otimes \Psi_P \rhd R \longmapsto K'(\!|\nu * xvec|\!)\langle N\rangle \prec R'$ **and** $\Psi \otimes \Psi_P \otimes (p \cdot \Psi_R) \vdash (p \cdot M) \leftrightarrow K'$ **and** $(p \cdot A_R) \sharp* K'$
using *comm1Aux* **by** *blast*

with *PTrans FrP* **have** $\Psi \rhd P \parallel R \longmapsto \tau \prec (\!|\nu * xvec|\!)(P' \parallel R')$ **using** *FrR* ‹$(p \cdot A_R) \sharp* \Psi$› ‹$(p \cdot A_R) \sharp* P$› ‹$(p \cdot A_R) \sharp* R$›
‹$xvec \sharp* P$› ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_P \sharp* R$› ‹$A_P \sharp* (p \cdot M)$› ‹$(p \cdot A_R) \sharp* K'$› ‹$(p \cdot A_R) \sharp* A_P$›
by (*intro Comm1*) (*assumption* | *simp*)+

moreover from *P'RelQ'* **have** $((\Psi \otimes (p \cdot \Psi_R)) \otimes \Psi', P', Q') \in Rel$ **by**(*rule Ext*)
with ‹$(p \cdot \Psi_R) \otimes \Psi' \simeq \Psi_R{}'$› **have** $(\Psi \otimes \Psi_R{}', P', Q') \in Rel$ **by**(*metis C3 Associativity compositionSym*)
with *FrR'* ‹$A_R{}' \sharp* P'$› ‹$A_R{}' \sharp* Q'$› ‹$A_R{}' \sharp* \Psi$› **have** $(\Psi, P' \parallel R', Q' \parallel R') \in Rel'$
by(*metis C1*)
with ‹*xvec* $\sharp* \Psi$› **have** $(\Psi, (\!|\nu * xvec|\!)(P' \parallel R'), (\!|\nu * xvec|\!)(Q' \parallel R')) \in Rel'$
by(*metis C2*)
ultimately show *?case* **by** *blast*
next
case(*cComm2* Ψ_R M *xvec* N Q' A_Q Ψ_Q K R' A_R)
have *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$ **by** *fact*
from ‹$A_Q \sharp* (P, R)$› **have** $A_Q \sharp* P$ **and** $A_Q \sharp* R$ **by** *simp*+

have *FrR*: *extractFrame* $R = \langle A_R, \Psi_R\rangle$ **by** *fact*
from ‹$A_R \sharp* (P, R)$› **have** $A_R \sharp* P$ **and** $A_R \sharp* R$ **by** *simp*+

from ‹*xvec* $\sharp* (P, R)$› **have** *xvec* $\sharp* P$ **and** *xvec* $\sharp* R$ **by** *simp*+

obtain A_P Ψ_P **where** *FrP*: *extractFrame* $P = \langle A_P, \Psi_P\rangle$ **and** $A_P \sharp* (\Psi, A_Q, \Psi_Q, A_R, M, N, K, R, P, xvec)$ **and** *distinct* A_P
by(*rule freshFrame*)
then have $A_P \sharp* \Psi$ **and** $A_P \sharp* A_Q$ **and** $A_P \sharp* \Psi_Q$ **and** $A_P \sharp* M$ **and** $A_P \sharp* R$
and $A_P \sharp* N$ $A_P \sharp* K$ **and** $A_P \sharp* A_R$ **and** $A_P \sharp* P$ **and** $A_P \sharp* xvec$
by *simp*+

from *FrP FrR* ‹$A_Q \sharp* P$› ‹$A_P \sharp* R$› ‹$A_R \sharp* P$› ‹$A_P \sharp* A_Q$› ‹$A_P \sharp* A_R$› ‹$A_P \sharp* xvec$› ‹*xvec* $\sharp* P$›
have $A_P \sharp* \Psi_R$ **and** $A_Q \sharp* \Psi_P$ **and** $A_R \sharp* \Psi_P$ **and** *xvec* $\sharp* \Psi_P$

by(*auto dest*!: *extractFrameFreshChain*)+

have *QTrans*: $\Psi \otimes \Psi_R \rhd Q \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec Q'$ **by** *fact*

note $\langle \Psi \otimes \Psi_Q \rhd R \longmapsto K(\!|N|\!) \prec R' \rangle$ *FrR* $\langle \Psi \otimes \Psi_Q \otimes \Psi_R \vdash M \leftrightarrow K \rangle$
moreover from *FrR* $\langle A_R \sharp * \Psi \rangle$ $\langle A_R \sharp * P \rangle$ $\langle A_R \sharp * Q \rangle$ **have** $\Psi \otimes \Psi_R \rhd P \rightsquigarrow [Rel]\ Q$ **by**(*metis PRelQ Sim*)
with *QTrans* **obtain** P' **where** *PTrans*: $\Psi \otimes \Psi_R \rhd P \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$ **and** $P'RelQ'$: $(\Psi \otimes \Psi_R, P', Q') \in Rel$
using $\langle xvec \sharp * \Psi \rangle$ $\langle xvec \sharp * \Psi_R \rangle$ $\langle xvec \sharp * P \rangle$
by(*force dest*: *simE*)
from *PTrans QTrans* $\langle A_R \sharp * P \rangle$ $\langle A_R \sharp * Q \rangle$ $\langle A_R \sharp * xvec \rangle$ $\langle xvec \sharp * M \rangle$ $\langle distinct\ xvec \rangle$ **have** $A_R \sharp * P'$ **and** $A_R \sharp * Q'$
by(*blast dest*: *outputFreshChainDerivative*)+
note *PTrans* $\langle \Psi \otimes \Psi_Q \otimes \Psi_R \vdash M \leftrightarrow K \rangle$
moreover have $\langle A_Q, (\Psi \otimes \Psi_Q) \otimes \Psi_R\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R\rangle$
proof −
have $\langle A_P, (\Psi \otimes \Psi_R) \otimes \Psi_P\rangle \simeq_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R\rangle$
by(*metis frameResChainPres frameNilStatEq Commutativity AssertionStatEqTrans Composition Associativity*)
moreover from *FrR* $\langle A_R \sharp * \Psi \rangle$ $\langle A_R \sharp * P \rangle$ $\langle A_R \sharp * Q \rangle$
have (*insertAssertion* (*extractFrame Q*) $(\Psi \otimes \Psi_R)$) $\hookrightarrow_F$ (*insertAssertion* (*extractFrame P*) $(\Psi \otimes \Psi_R)$)
by(*metis PRelQ StatImp*)
with *FrP FrQ* $\langle A_P \sharp * \Psi \rangle$ $\langle A_Q \sharp * \Psi \rangle$ $\langle A_P \sharp * \Psi_R \rangle$ $\langle A_Q \sharp * \Psi_R \rangle$
have $\langle A_Q, (\Psi \otimes \Psi_R) \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_R) \otimes \Psi_P\rangle$ **using** *freshCompChain* **by** *simp*
moreover have $\langle A_Q, (\Psi \otimes \Psi_Q) \otimes \Psi_R\rangle \simeq_F \langle A_Q, (\Psi \otimes \Psi_R) \otimes \Psi_Q\rangle$
by(*metis frameResChainPres frameNilStatEq Commutativity AssertionStatEqTrans Composition Associativity*)
ultimately show *?thesis*
by(*metis FrameStatEqImpCompose*)
qed
moreover note *FrP FrQ* $\langle distinct\ A_P \rangle$ $\langle distinct\ A_R \rangle$
moreover from $\langle A_P \sharp * A_R \rangle$ $\langle A_Q \sharp * A_R \rangle$ **have** $A_R \sharp * A_P$ **and** $A_R \sharp * A_Q$ **by** *simp*+
moreover note $\langle A_R \sharp * \Psi \rangle$ $\langle A_R \sharp * P \rangle$ $\langle A_R \sharp * Q \rangle$ $\langle A_R \sharp * R \rangle$ $\langle A_R \sharp * K \rangle$ $\langle A_P \sharp * \Psi \rangle$ $\langle A_P \sharp * P \rangle$
$\langle A_P \sharp * R \rangle$ $\langle A_P \sharp * M \rangle$ $\langle A_Q \sharp * R \rangle$ $\langle A_Q \sharp * M \rangle$ $\langle A_R \sharp * xvec \rangle$ $\langle xvec \sharp * M \rangle$
ultimately obtain K' **where** $\Psi \otimes \Psi_P \rhd R \longmapsto K'(\!|N|\!) \prec R'$ **and** $\Psi \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K'$ **and** $A_R \sharp * K'$
using *comm2Aux* **by** *blast*

with *PTrans FrP* **have** $\Psi \rhd P \parallel R \longmapsto \tau \prec (\!|\nu * xvec|\!)(P' \parallel R')$ **using** *FrR* $\langle A_R \sharp * \Psi \rangle$ $\langle A_R \sharp * P \rangle$ $\langle A_R \sharp * R \rangle$
$\langle A_R \sharp * \Psi \rangle$ $\langle A_R \sharp * P \rangle$ $\langle A_R \sharp * R \rangle$ **and** $\langle xvec \sharp * R \rangle$ $\langle A_P \sharp * \Psi \rangle$ $\langle A_P \sharp * P \rangle$ $\langle A_P \sharp * R \rangle$ $\langle A_P \sharp * A_R \rangle$ $\langle A_P \sharp * M \rangle$ $\langle A_R \sharp * K' \rangle$
by(*force intro*: *Comm2*)

moreover from ‹$\Psi \otimes \Psi_P \rhd R \longmapsto K'\langle\!|N|\!\rangle \prec R'$› *FrR* ‹*distinct* A_R› ‹$A_R \sharp* \Psi$› ‹$A_R \sharp* R$› ‹$A_R \sharp* P'$› ‹$A_R \sharp* Q'$› ‹$A_R \sharp* N$› ‹$A_R \sharp* K'$›
obtain Ψ' ${A_R}'$ ${\Psi_R}'$ **where** *ReqR'*: $\Psi_R \otimes \Psi' \simeq {\Psi_R}'$ **and** *FrR'*: *extractFrame* $R' = \langle {A_R}', {\Psi_R}' \rangle$
and ${A_R}' \sharp* \Psi$ **and** ${A_R}' \sharp* P'$ **and** ${A_R}' \sharp* Q'$
by(*auto intro*: *expandFrame*[**where** $C{=}(\Psi, P', Q')$ **and** $C'{=}\Psi$])

from *P'RelQ'* **have** $((\Psi \otimes \Psi_R) \otimes \Psi', P', Q') \in Rel$ **by**(*rule Ext*)
with *ReqR'* **have** $(\Psi \otimes {\Psi_R}', P', Q') \in Rel$ **by**(*metis C3 Associativity compositionSym*)
with *FrR'* ‹${A_R}' \sharp* P'$› ‹${A_R}' \sharp* Q'$› ‹${A_R}' \sharp* \Psi$› **have** $(\Psi, P' \parallel R', Q' \parallel R') \in Rel'$
by(*metis C1*)
with ‹*xvec* $\sharp* \Psi$› **have** $(\Psi, (\!|\nu{*}xvec|\!)(P' \parallel R'), (\!|\nu{*}xvec|\!)(Q' \parallel R')) \in Rel'$
by(*metis C2*)
ultimately show *?case* **by** *blast*
next
case(*cBrMerge* Ψ_R M N Q' A_Q Ψ_Q R' A_R)
have *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$ **by** *fact*
from ‹$A_Q \sharp* (P, R)$› **have** $A_Q \sharp* P$ **and** $A_Q \sharp* R$ **by** *simp+*

have *FrR*: *extractFrame* $R = \langle A_R, \Psi_R \rangle$ **by** *fact*
from ‹$A_R \sharp* (P, R)$› **have** $A_R \sharp* P$ **and** $A_R \sharp* R$ **by** *simp+*

obtain A_P Ψ_P **where** *FrP*: *extractFrame* $P = \langle A_P, \Psi_P \rangle$ **and** $A_P \sharp* (\Psi, A_Q, \Psi_Q, A_R, M, N, R, P)$ **and** *distinct* A_P
by(*rule freshFrame*)
then have $A_P \sharp* \Psi$ **and** $A_P \sharp* A_Q$ **and** $A_P \sharp* \Psi_Q$ **and** $A_P \sharp* M$ **and** $A_P \sharp* R$
and $A_P \sharp* N$ **and** $A_P \sharp* A_R$ **and** $A_P \sharp* P$
by *simp+*

from *FrP FrR* ‹$A_Q \sharp* P$› ‹$A_P \sharp* R$› ‹$A_R \sharp* P$› ‹$A_P \sharp* A_Q$› ‹$A_P \sharp* A_R$›
have $A_P \sharp* \Psi_R$ **and** $A_Q \sharp* \Psi_P$ **and** $A_R \sharp* \Psi_P$
by(*auto dest*: *extractFrameFreshChain*)+

have *QTrans*: $\Psi \otimes \Psi_R \rhd Q \longmapsto ¿M\langle\!|N|\!\rangle \prec Q'$ **by** *fact*

have *RTrans*: $\Psi \otimes \Psi_Q \rhd R \longmapsto ¿M\langle\!|N|\!\rangle \prec R'$ **by** *fact*

from *FrR* ‹$A_R \sharp* \Psi$› ‹$A_R \sharp* P$› ‹$A_R \sharp* Q$› **have** $\Psi \otimes \Psi_R \rhd P \rightsquigarrow[Rel]\ Q$ **by**(*metis PRelQ Sim*)
with *QTrans* **obtain** P' **where** *PTrans*: $\Psi \otimes \Psi_R \rhd P \longmapsto ¿M\langle\!|N|\!\rangle \prec P'$ **and** *P'RelQ'*: $(\Psi \otimes \Psi_R, P', Q') \in Rel$
by(*force dest*: *simE*)
from *PTrans QTrans* ‹$A_R \sharp* P$› ‹$A_R \sharp* Q$› ‹$A_R \sharp* N$› **have** $A_R \sharp* P'$ **and** $A_R \sharp* Q'$
by(*blast dest*: *brinputFreshChainDerivative*)+

have $\langle A_Q, (\Psi \otimes \Psi_Q) \otimes \Psi_R\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R\rangle$
proof −
have $\langle A_P, (\Psi \otimes \Psi_R) \otimes \Psi_P\rangle \simeq_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R\rangle$
by(*metis frameResChainPres frameNilStatEq Commutativity AssertionStatEqTrans Composition Associativity*)
moreover from *FrR* ‹$A_R \sharp^* \Psi$› ‹$A_R \sharp^* P$› ‹$A_R \sharp^* Q$›
have (*insertAssertion* (*extractFrame Q*) ($\Psi \otimes \Psi_R$)) $\hookrightarrow_F$ (*insertAssertion* (*extractFrame P*) ($\Psi \otimes \Psi_R$))
by(*metis PRelQ StatImp*)
with *FrP FrQ* ‹$A_P \sharp^* \Psi$› ‹$A_Q \sharp^* \Psi$› ‹$A_P \sharp^* \Psi_R$› ‹$A_Q \sharp^* \Psi_R$›
have $\langle A_Q, (\Psi \otimes \Psi_R) \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_R) \otimes \Psi_P\rangle$ **using** *freshCompChain* **by** *simp*
moreover have $\langle A_Q, (\Psi \otimes \Psi_Q) \otimes \Psi_R\rangle \simeq_F \langle A_Q, (\Psi \otimes \Psi_R) \otimes \Psi_Q\rangle$
by(*metis frameResChainPres frameNilStatEq Commutativity AssertionStatEqTrans Composition Associativity*)
ultimately show *?thesis*
by(*metis FrameStatEqImpCompose*)
qed
with *RTrans* ‹*extractFrame* $R = \langle A_R, \Psi_R\rangle$› ‹*distinct* A_R› ‹$A_P \sharp^* A_R$› ‹$A_Q \sharp^* A_R$›
‹$A_R \sharp^* \Psi$› ‹$A_R \sharp^* \Psi_P$› ‹$A_R \sharp^* \Psi_Q$› ‹$A_R \sharp^* R$› ‹$A_R \sharp^* M$› ‹$A_P \sharp^* R$› ‹$A_P \sharp^* M$› ‹$A_Q \sharp^* R$› ‹$A_Q \sharp^* M$›
have $\Psi \otimes \Psi_P \rhd R \longmapsto ¿M(\!|N|\!) \prec R'$
using *brCommInAux freshChainSym* **by** *blast*

with *PTrans FrP* **have** *Transition*: $\Psi \rhd P \parallel R \longmapsto ¿M(\!|N|\!) \prec (P' \parallel R')$ **using** *FrR* ‹$A_R \sharp^* \Psi$› ‹$A_R \sharp^* P$› ‹$A_R \sharp^* R$›
‹$A_R \sharp^* \Psi$› ‹$A_R \sharp^* P$› ‹$A_R \sharp^* R$› ‹$A_P \sharp^* \Psi$› ‹$A_P \sharp^* P$› ‹$A_P \sharp^* R$› ‹$A_P \sharp^* A_R$› ‹$A_P \sharp^* M$› ‹$A_R \sharp^* M$›
by(*force intro*: *BrMerge*)

from ‹$\Psi \otimes \Psi_P \rhd R \longmapsto ¿M(\!|N|\!) \prec R'$› *FrR* ‹*distinct* A_R› ‹$A_R \sharp^* \Psi$› ‹$A_R \sharp^* R$› ‹$A_R \sharp^* P'$› ‹$A_R \sharp^* Q'$› ‹$A_R \sharp^* N$› ‹$A_R \sharp^* M$›
obtain Ψ' $A_R{}'$ $\Psi_R{}'$ **where** *ReqR'*: $\Psi_R \otimes \Psi' \simeq \Psi_R{}'$ **and** *FrR'*: *extractFrame* $R' = \langle A_R{}', \Psi_R{}'\rangle$
and $A_R{}' \sharp^* \Psi$ **and** $A_R{}' \sharp^* P'$ **and** $A_R{}' \sharp^* Q'$
by(*auto intro*: *expandFrame*[**where** C=(Ψ, P', Q') **and** C'=Ψ])

from *P'RelQ'* **have** $((\Psi \otimes \Psi_R) \otimes \Psi', P', Q') \in Rel$ **by**(*rule Ext*)
with *ReqR'* **have** $(\Psi \otimes \Psi_R{}', P', Q') \in Rel$ **by**(*metis C3 Associativity compositionSym*)
with *FrR'* ‹$A_R{}' \sharp^* P'$› ‹$A_R{}' \sharp^* Q'$› ‹$A_R{}' \sharp^* \Psi$› **have** *Relation*: (Ψ, $P' \parallel R'$, $Q' \parallel R'$) $\in$ *Rel'*
by(*metis C1*)
show *?case* **using** *Transition Relation*
by *blast*
next
case(*cBrComm1* Ψ_R M N Q' A_Q Ψ_Q *xvec* R' A_R)
have *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$ **by** *fact*

from ‹A_Q ♯* (P, R)› **have** A_Q ♯* P **and** A_Q ♯* R **by** *simp+*

have *FrR*: *extractFrame* $R = \langle A_R, \Psi_R\rangle$ **by** *fact*
from ‹A_R ♯* (P, R)› **have** A_R ♯* P **and** A_R ♯* R **by** *simp+*
from ‹¡M⟪ν*$xvec$⟫⟨N⟩ = α› **have** $xvec = bn\ \alpha$
by(*auto simp add*: *action.inject*)
from ‹$xvec = bn\ \alpha$› ‹$bn\ \alpha$ ♯* P› ‹$bn\ \alpha$ ♯* R›
have $xvec$ ♯* P **and** $xvec$ ♯* R **by** *simp+*

obtain A_P Ψ_P **where** *FrP*: *extractFrame* $P = \langle A_P, \Psi_P\rangle$ **and** A_P ♯* (Ψ, A_Q, Ψ_Q, A_R, M, N, R, P, $xvec$) **and** *distinct* A_P
by(*rule freshFrame*)
then have A_P ♯* Ψ **and** A_P ♯* A_Q **and** A_P ♯* Ψ_Q **and** A_P ♯* M **and** A_P ♯* R
and A_P ♯* N **and** A_P ♯* A_R **and** A_P ♯* P **and** A_P ♯* $xvec$
by *simp+*

from *FrP FrR* ‹A_Q ♯* P› ‹A_P ♯* R› ‹A_R ♯* P› ‹A_P ♯* A_Q› ‹A_P ♯* A_R›
have A_P ♯* Ψ_R **and** A_Q ♯* Ψ_P **and** A_R ♯* Ψ_P
by(*auto dest*: *extractFrameFreshChain*)+

from ‹A_P ♯* $xvec$› ‹$xvec$ ♯* P› *FrP* **have** $xvec$ ♯* Ψ_P
by(*auto dest*: *extractFrameFreshChain*)

have *QTrans*: $\Psi \otimes \Psi_R \rhd Q \longmapsto$¿$M$⟪$N$⟫ $\prec Q'$ **by** *fact*

have *RTrans*: $\Psi \otimes \Psi_Q \rhd R \longmapsto$ ¡M⟪ν*$xvec$⟫⟨N⟩ $\prec R'$ **by** *fact*

from *RTrans FrR* ‹*distinct* A_R› ‹A_R ♯* R› ‹A_R ♯* $xvec$› ‹$xvec$ ♯* R› ‹$xvec$ ♯* Q› ‹$xvec$ ♯* Ψ› ‹$xvec$ ♯* Ψ_Q› ‹A_R ♯* Q›
‹A_R ♯* Ψ› ‹A_R ♯* Ψ_Q› ‹A_R ♯* M› ‹A_R ♯* N› ‹A_R ♯* R› ‹$xvec$ ♯* R› ‹A_R ♯* P› ‹$xvec$ ♯* P› ‹A_P ♯* A_R› ‹A_P ♯* $xvec$›
‹A_Q ♯* A_R› ‹A_Q ♯* $xvec$› ‹A_R ♯* Ψ_P› ‹$xvec$ ♯* Ψ_P› ‹*distinct* $xvec$› ‹$xvec$ ♯* M›
obtain p Ψ' A_R' Ψ_R' **where** S: *set* $p \subseteq$ *set* $xvec \times$ *set*($p \cdot xvec$) **and** *FrR'*: *extractFrame* $R' = \langle A_R', \Psi_R'\rangle$
and $(p \cdot \Psi_R) \otimes \Psi' \simeq \Psi_R'$ **and** A_R' ♯* Q **and** A_R' ♯* Ψ **and** $(p \cdot xvec)$ ♯* Ψ
and $(p \cdot xvec)$ ♯* Q **and** $(p \cdot xvec)$ ♯* Ψ_Q **and** $(p \cdot xvec)$ ♯* R **and** $(p \cdot xvec)$ ♯* M
and $(p \cdot xvec)$ ♯* P **and** $(p \cdot xvec)$ ♯* A_P **and** $(p \cdot xvec)$ ♯* A_Q **and** $(p \cdot xvec)$ ♯* Ψ_P
and A_R' ♯* P **and** A_R' ♯* N
by(*auto intro*: *expandFrame*[**where** C=(Ψ, Q, Ψ_Q, R, P, M, A_P, A_Q, Ψ_P) **and** C'=(Ψ, Q, Ψ_Q, R, P, M, A_P, A_Q, Ψ_P)])

from ‹A_R ♯* Ψ› **have** $(p \cdot A_R)$ ♯* $(p \cdot \Psi)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$xvec$ ♯* Ψ› ‹$(p \cdot xvec)$ ♯* Ψ› S **have** $(p \cdot A_R)$ ♯* Ψ **by** *simp*
from ‹A_R ♯* Ψ_P› **have** $(p \cdot A_R)$ ♯* $(p \cdot \Psi_P)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF*

pt-name-inst, *OF at-name-inst*])
with ‹*xvec* ♯* Ψ_P› ‹(p ∙ *xvec*) ♯* Ψ_P› *S* **have** (p ∙ A_R) ♯* Ψ_P **by** *simp*
from ‹A_R ♯* Ψ_Q› **have** (p ∙ A_R) ♯* (p ∙ Ψ_Q) **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹*xvec* ♯* Ψ_Q› ‹(p ∙ *xvec*) ♯* Ψ_Q› *S* **have** (p ∙ A_R) ♯* Ψ_Q **by** *simp*
from ‹A_R ♯* *M*› **have** (p ∙ A_R) ♯* (p ∙ *M*) **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹*xvec* ♯* *M*› ‹(p ∙ *xvec*) ♯* *M*› *S* **have** (p ∙ A_R) ♯* *M* **by** *simp*
from ‹A_R ♯* *P*› **have** (p ∙ A_R) ♯* (p ∙ *P*) **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹*xvec* ♯* *P*› ‹(p ∙ *xvec*) ♯* *P*› *S* **have** (p ∙ A_R) ♯* *P* **by** *simp*
from ‹A_R ♯* *Q*› **have** (p ∙ A_R) ♯* (p ∙ *Q*) **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹*xvec* ♯* *Q*› ‹(p ∙ *xvec*) ♯* *Q*› *S* **have** (p ∙ A_R) ♯* *Q* **by** *simp*
from ‹A_R ♯* *R*› **have** (p ∙ A_R) ♯* (p ∙ *R*) **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹*xvec* ♯* *R*› ‹(p ∙ *xvec*) ♯* *R*› *S* **have** (p ∙ A_R) ♯* *R* **by** *simp*

from ‹A_P ♯* *xvec*› ‹(p ∙ *xvec*) ♯* A_P› ‹A_P ♯* *M*› *S* **have** A_P ♯* (p ∙ *M*) **by**(*simp add*: *freshChainSimps*)
from ‹A_Q ♯* *xvec*› ‹(p ∙ *xvec*) ♯* A_Q› ‹A_Q ♯* *M*› *S* **have** A_Q ♯* (p ∙ *M*) **by**(*simp add*: *freshChainSimps*)
from ‹A_P ♯* *xvec*› ‹(p ∙ *xvec*) ♯* A_P› ‹A_P ♯* A_R› *S* **have** (p ∙ A_R) ♯* A_P **by**(*simp add*: *freshChainSimps*)
from ‹A_Q ♯* *xvec*› ‹(p ∙ *xvec*) ♯* A_Q› ‹A_Q ♯* A_R› *S* **have** (p ∙ A_R) ♯* A_Q **by**(*simp add*: *freshChainSimps*)

from *QTrans S* ‹*xvec* ♯* *Q*› ‹(p ∙ *xvec*) ♯* *Q*› **have** (p ∙ (Ψ ⊗ Ψ_R)) ▷ *Q* ⟼ ¿(p ∙ *M*)⦇*N*⦈ ≺ *Q*′
using *brinputPermFrameSubject name-list-set-fresh* **by** *blast*
with ‹*xvec* ♯* Ψ› ‹(p ∙ *xvec*) ♯* Ψ› *S* **have** *QTrans*: (Ψ ⊗ (p ∙ Ψ_R)) ▷ *Q* ⟼ ¿(p ∙ *M*)⦇*N*⦈ ≺ *Q*′
by(*simp add*: *eqvts*)

from *FrR* **have** (p ∙ *extractFrame R*) = p ∙ ⟨A_R, Ψ_R⟩ **by** *simp*
with ‹*xvec* ♯* *R*› ‹(p ∙ *xvec*) ♯* *R*› *S* **have** *FrR*: *extractFrame R* = ⟨(p ∙ A_R), (p ∙ Ψ_R)⟩
by(*simp add*: *eqvts*)

from *FrR* ‹(p ∙ A_R) ♯* Ψ› ‹(p ∙ A_R) ♯* *P*› ‹(p ∙ A_R) ♯* *Q*› **have** Ψ ⊗ (p ∙ Ψ_R) ▷ *P* ⇝[*Rel*] *Q*
by(*metis Sim PRelQ*)
with *QTrans* **obtain** *P*′ **where** *PTrans*: Ψ ⊗ (p ∙ Ψ_R) ▷ *P* ⟼¿(p ∙ *M*)⦇*N*⦈ ≺ *P*′ **and** *P*′*RelQ*′: (Ψ ⊗ (p ∙ Ψ_R), *P*′, *Q*′) ∈ *Rel*
by(*force dest*: *simE*)
with *QTrans* ‹A_R′ ♯* *P*› ‹A_R′ ♯* *Q*› ‹A_R′ ♯* *N*› **have** A_R′ ♯* *P*′ **and** A_R′ ♯* *Q*′
by(*blast dest*: *brinputFreshChainDerivative*)+

have $\langle A_Q, (\Psi \otimes \Psi_Q) \otimes (p \cdot \Psi_R)\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes (p \cdot \Psi_R)\rangle$
proof −
have $\langle A_P, (\Psi \otimes (p \cdot \Psi_R)) \otimes \Psi_P\rangle \simeq_F \langle A_P, (\Psi \otimes \Psi_P) \otimes (p \cdot \Psi_R)\rangle$
by(*metis frameResChainPres frameNilStatEq Commutativity AssertionStatEqTrans Composition Associativity*)
moreover from *FrR* $\langle(p \cdot A_R) \sharp* \Psi\rangle$ $\langle(p \cdot A_R) \sharp* P\rangle$ $\langle(p \cdot A_R) \sharp* Q\rangle$
have (*insertAssertion* (*extractFrame Q*) $(\Psi \otimes (p \cdot \Psi_R))) \hookrightarrow_F$ (*insertAssertion* (*extractFrame P*) $(\Psi \otimes (p \cdot \Psi_R)))$
by(*metis PRelQ StatImp*)
with *FrP FrQ* $\langle A_P \sharp* \Psi\rangle$ $\langle A_Q \sharp* \Psi\rangle$ $\langle A_P \sharp* \Psi_R\rangle$ $\langle A_Q \sharp* \Psi_R\rangle$ $\langle A_P \sharp* xvec\rangle$ $\langle(p \cdot xvec) \sharp* A_P\rangle$ $\langle A_Q \sharp* xvec\rangle$ $\langle(p \cdot xvec) \sharp* A_Q\rangle$ *S*
have $\langle A_Q, (\Psi \otimes (p \cdot \Psi_R)) \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes (p \cdot \Psi_R)) \otimes \Psi_P\rangle$ **using** *freshCompChain*
by(*simp add*: *freshChainSimps*)
moreover have $\langle A_Q, (\Psi \otimes \Psi_Q) \otimes (p \cdot \Psi_R)\rangle \simeq_F \langle A_Q, (\Psi \otimes (p \cdot \Psi_R)) \otimes \Psi_Q\rangle$
by(*metis frameResChainPres frameNilStatEq Commutativity AssertionStatEqTrans Composition Associativity*)
ultimately show *?thesis*
by(*metis FrameStatEqImpCompose*)
qed
moreover note *RTrans FrR FrP FrQ* $\langle$*distinct* $A_P\rangle$
moreover from $\langle$*distinct* $A_R\rangle$ **have** *distinct*$(p \cdot A_R)$ **by** *simp*
moreover note $\langle(p \cdot A_R) \sharp* A_P\rangle$ $\langle(p \cdot A_R) \sharp* A_Q\rangle$ $\langle(p \cdot A_R) \sharp* \Psi\rangle$ $\langle(p \cdot A_R) \sharp* \Psi_P\rangle$ $\langle(p \cdot A_R) \sharp* \Psi_Q\rangle$ $\langle(p \cdot A_R) \sharp* M\rangle$ $\langle(p \cdot A_R) \sharp* P\rangle$ $\langle(p \cdot A_R) \sharp* Q\rangle$ $\langle(p \cdot A_R) \sharp* R\rangle$
$\langle A_P \sharp* \Psi\rangle$ $\langle A_P \sharp* R\rangle$ $\langle A_P \sharp* P\rangle$ $\langle A_P \sharp* M\rangle$ $\langle A_Q \sharp* M\rangle$ $\langle A_P \sharp* (p \cdot M)\rangle$ $\langle A_Q \sharp* R\rangle$ $\langle A_P \sharp* xvec\rangle$ $\langle xvec \sharp* P\rangle$ $\langle A_P \sharp* R\rangle$
ultimately have $\Psi \otimes \Psi_P \rhd R \longmapsto\!\!¡ M(\!|\nu*xvec|\!)\langle N\rangle \prec R'$
using *brCommOutAux* **by** *blast*
with *S* $\langle xvec \sharp* M\rangle$ $\langle(p \cdot xvec) \sharp* M\rangle$ **have** $\Psi \otimes \Psi_P \rhd R \longmapsto\!\!¡ (p \cdot M)(\!|\nu*xvec|\!)\langle N\rangle \prec R'$ **by** *simp*

with *PTrans FrP S* $\langle xvec \sharp* M\rangle$ $\langle(p \cdot xvec) \sharp* M\rangle$ $\langle(p \cdot A_R) \sharp* M\rangle$ **have** $\Psi \rhd P \parallel R \longmapsto\!\!¡ (p \cdot M)(\!|\nu*xvec|\!)\langle N\rangle \prec (P' \parallel R')$ **using** *FrR* $\langle(p \cdot A_R) \sharp* \Psi\rangle$ $\langle(p \cdot A_R) \sharp* P\rangle$ $\langle(p \cdot A_R) \sharp* R\rangle$
$\langle xvec \sharp* P\rangle$ $\langle A_P \sharp* \Psi\rangle$ $\langle A_P \sharp* P\rangle$ $\langle A_P \sharp* R\rangle$ $\langle A_P \sharp* (p \cdot M)\rangle$ $\langle(p \cdot A_R) \sharp* A_P\rangle$
by(*intro BrComm1*) (*assumption* | *simp*)+

with *S* $\langle xvec \sharp* M\rangle$ $\langle(p \cdot xvec) \sharp* M\rangle$
have *Transition*: $\Psi \rhd P \parallel R \longmapsto\!\!¡ M(\!|\nu*xvec|\!)\langle N\rangle \prec (P' \parallel R')$ **by** *simp*

from *P'RelQ'* **have** $((\Psi \otimes (p \cdot \Psi_R)) \otimes \Psi', P', Q') \in Rel$ **by**(*rule Ext*)
with $\langle(p \cdot \Psi_R) \otimes \Psi' \simeq \Psi_R'\rangle$ **have** $(\Psi \otimes \Psi_R', P', Q') \in Rel$ **by**(*metis C3 Associativity compositionSym*)
with *FrR'* $\langle A_R' \sharp* P'\rangle$ $\langle A_R' \sharp* Q'\rangle$ $\langle A_R' \sharp* \Psi\rangle$ **have** *Relation*: $(\Psi, P' \parallel R', Q' \parallel R') \in Rel'$
by(*metis C1*)

show *?case* **using** *Transition Relation*
by *blast*
next
case(*cBrComm2* Ψ_R M *xvec* N Q' A_Q Ψ_Q R' A_R)
have *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$ **by** *fact*
from ‹$A_Q \sharp* (P, R)$› **have** $A_Q \sharp* P$ **and** $A_Q \sharp* R$ **by** *simp*+

have *FrR*: *extractFrame* $R = \langle A_R, \Psi_R\rangle$ **by** *fact*
from ‹$A_R \sharp* (P, R)$› **have** $A_R \sharp* P$ **and** $A_R \sharp* R$ **by** *simp*+
from ‹¡$M(\!|\nu* xvec|\!)\langle N\rangle = \alpha$› **have** *xvec* = *bn* α
by(*auto simp add*: *action.inject*)
from ‹*xvec* = *bn* α› ‹*bn* $\alpha \sharp* P$› ‹*bn* $\alpha \sharp* R$›
have *xvec* $\sharp* P$ **and** *xvec* $\sharp* R$ **by** *simp*+

obtain A_P Ψ_P **where** *FrP*: *extractFrame* $P = \langle A_P, \Psi_P\rangle$ **and** $A_P \sharp* (\Psi, A_Q, \Psi_Q, A_R, M, N, R, P)$ **and** *distinct* A_P
by(*rule freshFrame*)
then have $A_P \sharp* \Psi$ **and** $A_P \sharp* A_Q$ **and** $A_P \sharp* \Psi_Q$ **and** $A_P \sharp* M$ **and** $A_P \sharp* R$
and $A_P \sharp* N$ **and** $A_P \sharp* A_R$ **and** $A_P \sharp* P$
by *simp*+

from *FrP FrR* ‹$A_Q \sharp* P$› ‹$A_P \sharp* R$› ‹$A_R \sharp* P$› ‹$A_P \sharp* A_Q$› ‹$A_P \sharp* A_R$›
have $A_P \sharp* \Psi_R$ **and** $A_Q \sharp* \Psi_P$ **and** $A_R \sharp* \Psi_P$
by(*auto dest*: *extractFrameFreshChain*)+

have *QTrans*: $\Psi \otimes \Psi_R \rhd Q \longmapsto$¡$M(\!|\nu* xvec|\!)\langle N\rangle \prec Q'$ **by** *fact*

have *RTrans*: $\Psi \otimes \Psi_Q \rhd R \longmapsto$¿$M(\!|N|\!) \prec R'$ **by** *fact*

from *FrR* ‹$A_R \sharp* \Psi$› ‹$A_R \sharp* P$› ‹$A_R \sharp* Q$› **have** $\Psi \otimes \Psi_R \rhd P \rightsquigarrow[Rel]\ Q$ **by**(*metis PRelQ Sim*)
from ‹$\Psi \otimes \Psi_R \rhd P \rightsquigarrow[Rel]\ Q$› *QTrans* ‹*xvec* $\sharp* \Psi$› ‹*xvec* $\sharp* \Psi_R$› ‹*xvec* $\sharp* P$›
obtain P' **where** *PTrans*: $\Psi \otimes \Psi_R \rhd P \longmapsto$¡$M(\!|\nu* xvec|\!)\langle N\rangle \prec P'$ **and** $P'RelQ'$: $(\Psi \otimes \Psi_R, P', Q') \in Rel$
by(*force dest*: *simE*)
from *PTrans QTrans* ‹$A_R \sharp* P$› ‹$A_R \sharp* Q$› ‹$A_R \sharp* N$› ‹$A_R \sharp* xvec$› ‹*xvec* $\sharp* M$› ‹*distinct xvec*› **have** $A_R \sharp* P'$ **and** $A_R \sharp* Q'$
by(*blast dest*: *broutputFreshChainDerivative*)+

have $\langle A_Q, (\Psi \otimes \Psi_Q) \otimes \Psi_R\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R\rangle$
proof −
have $\langle A_P, (\Psi \otimes \Psi_R) \otimes \Psi_P\rangle \simeq_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R\rangle$
by(*metis frameResChainPres frameNilStatEq Commutativity AssertionStatEqTrans Composition Associativity*)
moreover from *FrR* ‹$A_R \sharp* \Psi$› ‹$A_R \sharp* P$› ‹$A_R \sharp* Q$›
have (*insertAssertion* (*extractFrame* Q) $(\Psi \otimes \Psi_R)$) $\hookrightarrow_F$ (*insertAssertion* (*extractFrame* P) $(\Psi \otimes \Psi_R)$)

by(*metis PRelQ StatImp*)
with *FrP FrQ* ‹$A_P \sharp^* \Psi$› ‹$A_Q \sharp^* \Psi$› ‹$A_P \sharp^* \Psi_R$› ‹$A_Q \sharp^* \Psi_R$›
have $\langle A_Q, (\Psi \otimes \Psi_R) \otimes \Psi_Q\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_R) \otimes \Psi_P\rangle$ **using** *freshCompChain* **by** *simp*
moreover have $\langle A_Q, (\Psi \otimes \Psi_Q) \otimes \Psi_R\rangle \simeq_F \langle A_Q, (\Psi \otimes \Psi_R) \otimes \Psi_Q\rangle$
by(*metis frameResChainPres frameNilStatEq Commutativity AssertionStatEqTrans Composition Associativity*)
ultimately show *?thesis*
by(*metis FrameStatEqImpCompose*)
qed
with *RTrans* ‹*extractFrame* $R = \langle A_R, \Psi_R\rangle$› ‹*distinct* A_R› ‹$A_P \sharp^* A_R$› ‹$A_Q \sharp^* A_R$›
‹$A_R \sharp^* \Psi$› ‹$A_R \sharp^* \Psi_P$› ‹$A_R \sharp^* \Psi_Q$› ‹$A_R \sharp^* R$› ‹$A_R \sharp^* M$› ‹$A_P \sharp^* R$› ‹$A_P \sharp^* M$› ‹$A_Q \sharp^* R$› ‹$A_Q \sharp^* M$›
have $\Psi \otimes \Psi_P \rhd R \longmapsto ¡M(\!|N|\!) \prec R'$
using *brCommInAux freshChainSym* **by** *blast*

with *PTrans FrP* **have** *Transition*: $\Psi \rhd P \parallel R \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec (P' \parallel R')$ **using** *FrR* ‹$A_R \sharp^* \Psi$› ‹$A_R \sharp^* P$› ‹$A_R \sharp^* R$›
‹$A_R \sharp^* \Psi$› ‹$A_R \sharp^* P$› ‹$A_R \sharp^* R$› ‹$A_P \sharp^* \Psi$› ‹$A_P \sharp^* P$› ‹$A_P \sharp^* R$› ‹$A_P \sharp^* A_R$› ‹$A_P \sharp^* M$› ‹$A_R \sharp^* M$› ‹*xvec* $\sharp^* R$›
by(*force intro*: *BrComm2*)

from ‹$\Psi \otimes \Psi_P \rhd R \longmapsto ¡M(\!|N|\!) \prec R'$› *FrR* ‹*distinct* A_R› ‹$A_R \sharp^* \Psi$› ‹$A_R \sharp^* R$› ‹$A_R \sharp^* P'$› ‹$A_R \sharp^* Q'$› ‹$A_R \sharp^* N$› ‹$A_R \sharp^* M$›
obtain Ψ' $A_R{}'$ $\Psi_R{}'$ **where** *ReqR'*: $\Psi_R \otimes \Psi' \simeq \Psi_R{}'$ **and** *FrR'*: *extractFrame* $R' = \langle A_R{}', \Psi_R{}'\rangle$
and $A_R{}' \sharp^* \Psi$ **and** $A_R{}' \sharp^* P'$ **and** $A_R{}' \sharp^* Q'$
by(*auto intro*: *expandFrame*[**where** C=(Ψ, P', Q') **and** C'=Ψ])

from *P'RelQ'* **have** $((\Psi \otimes \Psi_R) \otimes \Psi', P', Q') \in Rel$ **by**(*rule Ext*)
with *ReqR'* **have** $(\Psi \otimes \Psi_R{}', P', Q') \in Rel$ **by**(*metis C3 Associativity compositionSym*)
with *FrR'* ‹$A_R{}' \sharp^* P'$› ‹$A_R{}' \sharp^* Q'$› ‹$A_R{}' \sharp^* \Psi$› **have** *Relation*: $(\Psi, P' \parallel R', Q' \parallel R') \in Rel'$
by(*metis C1*)
show *?case* **using** *Transition Relation*
by *blast*
qed
qed

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* × (*'a*, *'b*, *'c*) *psi* × (*'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 $\Psi \rhd R \parallel !P \rightsquigarrow [Rel']\ R \parallel !Q$
using ‹*eqvt Rel'*›
proof(*induct rule*: *simI*[*of* - - - - ()])
case(*cSim* $\alpha\ RQ'$)
from ‹$bn\ \alpha \sharp* (R \parallel !P)$› ‹$bn\ \alpha \sharp* (R \parallel !Q)$› **have** $bn\ \alpha \sharp* P$ **and** $bn\ \alpha \sharp* (!Q)$ **and** $bn\ \alpha \sharp* Q$ **and** $bn\ \alpha \sharp* R$ **by** *simp+*
from ‹$\Psi \rhd R \parallel !Q \longmapsto \alpha \prec RQ'$› ‹$bn\ \alpha \sharp* \Psi$› ‹$bn\ \alpha \sharp* R$› ‹$bn\ \alpha \sharp* !Q$› ‹$bn\ \alpha \sharp* subject\ \alpha$› **show** *?case*
proof(*induct rule*: *parCases*[**where** *C*=*P*])
case(*cPar1 R' A_Q Ψ_Q*)
from ‹$extractFrame\ (!Q) = \langle A_Q, \Psi_Q \rangle$› **have** $A_Q = []$ **and** $\Psi_Q = SBottom'$ **by** *simp+*
with ‹$\Psi \otimes \Psi_Q \rhd R \longmapsto \alpha \prec R'$› ‹$bn\ \alpha \sharp* P$› **have** $\Psi \rhd R \parallel !P \longmapsto \alpha \prec (R' \parallel !P)$

by(*intro Par1*) (*assumption* | *simp*)+
moreover from ‹(Ψ, *P*, *Q*) ∈ *Rel*› ‹*guarded P*› ‹*guarded Q*› **have** (Ψ, *R′* ‖ !*P*, *R′* ‖ !*Q*) ∈ *Rel′*
by(*rule C1*)
ultimately show *?case* **by** *blast*
next
case(*cPar2 Q′ A*$_R$ Ψ$_R$)
have *QTrans*: Ψ ⊗ Ψ$_R$ ▷ !*Q* ⟼α ≺ *Q′* **and** *FrR*: *extractFrame R* = ⟨*A*$_R$, Ψ$_R$⟩ **by** *fact*+
with ‹*bn* α ♯* *R*› ‹*A*$_R$ ♯* α› **have** *bn* α ♯* Ψ$_R$ **by**(*force dest*: *extractFrameFreshChain*)
with *QTrans* ‹(Ψ, *P*, *Q*) ∈ *Rel*› ‹*bn* α ♯* Ψ›‹*bn* α ♯* *P*› ‹*bn* α ♯* *Q*› ‹*guarded P*› ‹*bn* α ♯* *subject* α›
obtain *P′ S T* **where** *PTrans*: Ψ ⊗ Ψ$_R$ ▷ !*P* ⟼α ≺ *P′* **and** (Ψ ⊗ Ψ$_R$, *P′*, *T* ‖ !*P*) ∈ *Rel*
and (Ψ ⊗ Ψ$_R$, *Q′*, *S* ‖ !*Q*) ∈ *Rel* **and** (Ψ ⊗ Ψ$_R$, *S*, *T*) ∈ *Rel*
and *suppT*: ((*supp T*)::*name set*) ⊆ *supp P′* **and** *suppS*: ((*supp S*)::*name set*) ⊆ *supp Q′*
by(*drule-tac cSym*) (*auto dest*: *Der cExt*)
from *PTrans FrR* ‹*A*$_R$ ♯* Ψ› ‹*A*$_R$ ♯* *P*› ‹*A*$_R$ ♯* α› ‹*bn* α ♯* *R*› **have** Ψ ▷ *R* ‖ !*P* ⟼α ≺ (*R* ‖ *P′*)
by(*intro Par2*) *auto*
moreover
{
from ‹*A*$_R$ ♯* *P*› ‹*A*$_R$ ♯* (!*Q*)› ‹*A*$_R$ ♯* α› *PTrans QTrans* ‹*bn* α ♯* *subject* α› ‹*distinct*(*bn* α)› **have** *A*$_R$ ♯* *P′* **and** *A*$_R$ ♯* *Q′*
by(*force dest*: *freeFreshChainDerivative*)+
from ‹(Ψ ⊗ Ψ$_R$, *P′*, *T* ‖ !*P*) ∈ *Rel*› *FrR* ‹*A*$_R$ ♯* Ψ› ‹*A*$_R$ ♯* *P′*› ‹*A*$_R$ ♯* *P*› *suppT* **have** (Ψ, *R* ‖ *P′*, *R* ‖ (*T* ‖ !*P*)) ∈ *Rel*
by(*intro FrameParPres*) (*auto simp add*: *fresh-star-def fresh-def psi.supp*)
then have (Ψ, *R* ‖ *P′*, (*R* ‖ *T*) ‖ !*P*) ∈ *Rel* **by**(*blast intro*: *Assoc Trans*)
moreover from ‹(Ψ, *P*, *Q*) ∈ *Rel*› ‹*guarded P*› ‹*guarded Q*› **have** (Ψ, (*R* ‖ *T*) ‖ !*P*, (*R* ‖ *T*) ‖ !*Q*) ∈ *Rel′*
by(*rule C1*)
moreover from ‹(Ψ ⊗ Ψ$_R$, *Q′*, *S* ‖ !*Q*) ∈ *Rel*› ‹(Ψ ⊗ Ψ$_R$, *S*, *T*) ∈ *Rel*› **have** (Ψ ⊗ Ψ$_R$, *Q′*, *T* ‖ !*Q*) ∈ *Rel*
by(*blast intro*: *ParPres Trans*)
with *FrR* ‹*A*$_R$ ♯* Ψ› ‹*A*$_R$ ♯* *P′*› ‹*A*$_R$ ♯* *Q′*› ‹*A*$_R$ ♯* (!*Q*)› *suppT suppS* **have** (Ψ, *R* ‖ *Q′*, *R* ‖ (*T* ‖ !*Q*)) ∈ *Rel*
by(*intro FrameParPres*) (*auto simp add*: *fresh-star-def fresh-def psi.supp*)
then have (Ψ, *R* ‖ *Q′*, (*R* ‖ *T*) ‖ !*Q*) ∈ *Rel* **by**(*blast intro*: *Assoc Trans*)
ultimately have (Ψ, *R* ‖ *P′*, *R* ‖ *Q′*) ∈ *Rel′* **by**(*blast intro*: *cSym Compose*)
}
ultimately show *?case* **by** *blast*
next
case(*cComm1* Ψ$_Q$ *M N R′ A*$_R$ Ψ$_R$ *K xvec Q′ A*$_Q$)
from ‹*extractFrame* (!*Q*) = ⟨*A*$_Q$, Ψ$_Q$⟩› **have** *A*$_Q$ = [] **and** Ψ$_Q$ = *SBottom′* **by** *simp*+
have *RTrans*: Ψ ⊗ Ψ$_Q$ ▷ *R* ⟼*M*⦇*N*⦈ ≺ *R′* **and** *FrR*: *extractFrame R* = ⟨*A*$_R$,

$\Psi_R\rangle$ **by** *fact*+
moreover have *QTrans*: $\Psi \otimes \Psi_R \rhd\, !Q \longmapsto K(\!|\nu{*}xvec|\!)\langle N\rangle \prec Q'$ **by** *fact*
from *FrR* ‹*xvec* ♯* *R*› ‹A_R ♯* *xvec*› **have** *xvec* ♯* Ψ_R **by**(*force dest*: *extractFrameFreshChain*)
with *QTrans* ‹(Ψ, *P*, *Q*) ∈ *Rel*› ‹*xvec* ♯* Ψ›‹*xvec* ♯* *P*› ‹*xvec* ♯* (!*Q*)› ‹*guarded P*› ‹*xvec* ♯* *K*›
obtain P' *S T* **where** *PTrans*: $\Psi \otimes \Psi_R \rhd\, !P \longmapsto K(\!|\nu{*}xvec|\!)\langle N\rangle \prec P'$ **and** ($\Psi \otimes \Psi_R$, P', *T* ‖ !*P*) ∈ *Rel*
and ($\Psi \otimes \Psi_R$, Q', *S* ‖ !*Q*) ∈ *Rel* **and** ($\Psi \otimes \Psi_R$, *S*, *T*) ∈ *Rel*
and *suppT*: ((*supp T*)::*name set*) ⊆ *supp* P' **and** *suppS*: ((*supp S*)::*name set*) ⊆ *supp* Q'
apply(*drule-tac cSym*)
by (*metis Der bn.simps*(3) *cExt freshCompChain*(1) *optionFreshChain*(1) *psiFreshVec*(7) *subject.simps*(3))
note ‹$\Psi \otimes \Psi_R \otimes \Psi_Q \vdash M \leftrightarrow K$›
ultimately have $\Psi \rhd R \parallel !P \longmapsto \tau \prec (\!|\nu{*}xvec|\!)(R' \parallel P')$
using *PTrans* ‹Ψ_Q = *SBottom'*› ‹*xvec* ♯* *R*› ‹A_R ♯* Ψ› ‹A_R ♯* *R*› ‹A_R ♯* *M*› ‹A_R ♯* *P*›
by(*intro Comm1*) (*assumption* | *simp*)+

moreover from ‹A_R ♯* *P*› ‹A_R ♯* (!*Q*)› ‹A_R ♯* *xvec*› *PTrans QTrans* ‹*xvec* ♯* *K*› ‹*distinct xvec*›
have A_R ♯* P' **and** A_R ♯* Q' **by**(*force dest*: *outputFreshChainDerivative*)+
moreover with *RTrans FrR* ‹*distinct* A_R› ‹A_R ♯* *R*› ‹A_R ♯* *N*› ‹A_R ♯* Ψ› ‹A_R ♯* *P*› ‹A_R ♯* (!*Q*)› ‹A_R ♯* *M*›
obtain Ψ' A_R' Ψ_R' **where** *FrR'*: *extractFrame* $R' = \langle A_R', \Psi_R'\rangle$ **and** $\Psi_R \otimes \Psi' \simeq \Psi_R'$ **and** A_R' ♯* Ψ
and A_R' ♯* P' **and** A_R' ♯* Q' **and** A_R' ♯* *P* **and** A_R' ♯* *Q*
by(*auto intro*: *expandFrame*[**where** *C*=(Ψ, *P*, P', *Q*, Q') **and** *C'*=Ψ])

moreover
{
from ‹($\Psi \otimes \Psi_R$, P', *T* ‖ !*P*) ∈ *Rel*› **have** (($\Psi \otimes \Psi_R$) ⊗ Ψ', P', *T* ‖ !*P*) ∈ *Rel* **by**(*rule cExt*)
with ‹$\Psi_R \otimes \Psi' \simeq \Psi_R'$› **have** ($\Psi \otimes \Psi_R'$, P', *T* ‖ !*P*) ∈ *Rel*
by(*metis Associativity StatEq compositionSym*)
with *FrR'* ‹A_R' ♯* Ψ› ‹A_R' ♯* P'› ‹A_R' ♯* *P*› *suppT* **have** (Ψ, R' ‖ P', R' ‖ (*T* ‖ !*P*)) ∈ *Rel*
by(*intro FrameParPres*) (*auto simp add*: *fresh-star-def fresh-def psi.supp*)
then have (Ψ, R' ‖ P', (R' ‖ *T*) ‖ !*P*) ∈ *Rel* **by**(*blast intro*: *Assoc Trans*)
with ‹*xvec* ♯* Ψ› ‹*xvec* ♯* *P*› **have** (Ψ, (|$\nu{*}$*xvec*|)(R' ‖ P'), ((|$\nu{*}$*xvec*|)(R' ‖ *T*)) ‖ !*P*) ∈ *Rel*
by(*metis ResPres psiFreshVec ScopeExt Trans*)
moreover from ‹(*P*, *Q*) ∈ *Rel*› ‹*guarded P*› ‹*guarded Q*› **have** (Ψ, ((|$\nu{*}$*xvec*|)(R' ‖ *T*)) ‖ !*P*, ((|$\nu{*}$*xvec*|)(R' ‖ *T*)) ‖ !*Q*) ∈ *Rel'*
by(*rule C1*)
moreover from ‹($\Psi \otimes \Psi_R$, Q', *S* ‖ !*Q*) ∈ *Rel*› ‹($\Psi \otimes \Psi_R$, *S*, *T*) ∈ *Rel*› **have** ($\Psi \otimes \Psi_R$, Q', *T* ‖ !*Q*) ∈ *Rel*
by(*blast intro*: *ParPres Trans*)

then have $((\Psi \otimes \Psi_R) \otimes \Psi',\ Q',\ T \parallel !Q) \in Rel$ **by**(*rule cExt*)
with $\langle \Psi_R \otimes \Psi' \simeq \Psi_R{}' \rangle$ **have** $(\Psi \otimes \Psi_R{}',\ Q',\ T \parallel !Q) \in Rel$
by(*metis Associativity StatEq compositionSym*)
with *FrR'* $\langle A_R{}' \sharp* \Psi \rangle$ $\langle A_R{}' \sharp* P' \rangle$ $\langle A_R{}' \sharp* Q' \rangle$ $\langle A_R{}' \sharp* Q \rangle$ *suppT suppS* **have** $(\Psi,\ R' \parallel Q',\ R' \parallel (T \parallel !Q)) \in Rel$
by(*intro FrameParPres*) (*auto simp add*: *fresh-star-def fresh-def psi.supp*)
then have $(\Psi,\ R' \parallel Q',\ (R' \parallel T) \parallel !Q) \in Rel$ **by**(*blast intro*: *Assoc Trans*)
with $\langle xvec \sharp* \Psi \rangle$ $\langle xvec \sharp* (!Q) \rangle$ **have** $(\Psi,\ (\!|\nu* xvec|\!)(R' \parallel Q'),\ ((\!|\nu* xvec|\!)(R' \parallel T)) \parallel !Q) \in Rel$
by(*metis ResPres psiFreshVec ScopeExt Trans*)
ultimately have $(\Psi,\ (\!|\nu* xvec|\!)(R' \parallel P'),\ (\!|\nu* xvec|\!)(R' \parallel Q')) \in Rel'$ **by**(*blast intro*: *cSym Compose*)
}
ultimately show *?case* **by** *blast*
next
case(*cComm2* Ψ_Q *M xvec N R'* A_R Ψ_R *K Q'* A_Q)
from $\langle extractFrame\ (!Q) = \langle A_Q,\ \Psi_Q \rangle \rangle$ **have** $A_Q = []$ **and** $\Psi_Q = SBottom'$ **by** *simp+*
have *RTrans*: $\Psi \otimes \Psi_Q \rhd R \longmapsto M(\!|\nu* xvec|\!)\langle N \rangle \prec R'$ **and** *FrR*: *extractFrame* $R = \langle A_R,\ \Psi_R \rangle$ **by** *fact+*
then obtain *p* Ψ' $A_R{}'$ $\Psi_R{}'$ **where** *S*: *set p* $\subseteq$ *set xvec* $\times$ *set*$(p \cdot xvec)$
and *FrR'*: *extractFrame* $R' = \langle A_R{}',\ \Psi_R{}' \rangle$ **and** $(p \cdot \Psi_R) \otimes \Psi' \simeq \Psi_R{}'$ **and** $A_R{}' \sharp* \Psi$
and $A_R{}' \sharp* N$ **and** $A_R{}' \sharp* R'$ **and** $A_R{}' \sharp* P$ **and** $A_R{}' \sharp* Q$ **and** $(p \cdot xvec) \sharp* \Psi$
and $(p \cdot xvec) \sharp* P$ **and** $(p \cdot xvec) \sharp* Q$ **and** $xvec \sharp* A_R{}'$ **and** $(p \cdot xvec) \sharp* A_R{}'$
and *distinctPerm p* **and** $(p \cdot xvec) \sharp* R'$ **and** $(p \cdot xvec) \sharp* N$
using $\langle distinct\ A_R \rangle$ $\langle A_R \sharp* R \rangle$ $\langle A_R \sharp* M \rangle$ $\langle A_R \sharp* xvec \rangle$ $\langle A_R \sharp* N \rangle$ $\langle A_R \sharp* \Psi \rangle$ $\langle A_R \sharp* P \rangle$ $\langle A_R \sharp* (!Q) \rangle$
$\langle xvec \sharp* \Psi \rangle$ $\langle xvec \sharp* P \rangle$ $\langle xvec \sharp* (!Q) \rangle$ $\langle xvec \sharp* R \rangle$ $\langle xvec \sharp* M \rangle$ $\langle distinct\ xvec \rangle$
by(*auto intro*: *expandFrame*[**where** $C=(\Psi,\ P,\ Q)$ **and** $C'=(\Psi,\ P,\ Q)$])

from *RTrans S* $\langle (p \cdot xvec) \sharp* N \rangle$ $\langle (p \cdot xvec) \sharp* R' \rangle$ **have** $\Psi \otimes \Psi_Q \rhd R \longmapsto M(\!|\nu*(p \cdot xvec)|\!)\langle (p \cdot N) \rangle \prec (p \cdot R')$
apply(*simp add*: *residualInject*)
by(*subst boundOutputChainAlpha''*[*symmetric*]) *auto*

moreover have *QTrans*: $\Psi \otimes \Psi_R \rhd !Q \longmapsto K(\!|N|\!) \prec Q'$ **by** *fact*
with *QTrans S* $\langle (p \cdot xvec) \sharp* N \rangle$ **have** $\Psi \otimes \Psi_R \rhd !Q \longmapsto K(\!|(p \cdot N)|\!) \prec (p \cdot Q')$ **using** $\langle distinctPerm\ p \rangle$ $\langle xvec \sharp* (!Q) \rangle$ $\langle (p \cdot xvec) \sharp* Q \rangle$
by(*intro inputAlpha*) *auto*
with $\langle (\Psi,\ P,\ Q) \in Rel \rangle$ $\langle guarded\ P \rangle$
obtain P' *S T* **where** *PTrans*: $\Psi \otimes \Psi_R \rhd !P \longmapsto K(\!|(p \cdot N)|\!) \prec P'$ **and** $(\Psi \otimes \Psi_R,\ P',\ T \parallel !P) \in Rel$
and $(\Psi \otimes \Psi_R,\ (p \cdot Q'),\ S \parallel !Q) \in Rel$ **and** $(\Psi \otimes \Psi_R,\ S,\ T) \in Rel$
and *suppT*: $((supp\ T)::name\ set) \subseteq supp\ P'$ **and** *suppS*: $((supp\ S)::name\ set) \subseteq supp(p \cdot Q')$
by(*drule-tac cSym*) (*auto dest*: *Der cExt*)

note ‹$\Psi \otimes \Psi_R \otimes \Psi_Q \vdash M \leftrightarrow K$›
ultimately have $\Psi \rhd R \parallel !P \longmapsto\tau \prec (\!|\nu*(p \cdot xvec)|\!)((p \cdot R') \parallel P')$
using *PTrans FrR* ‹$\Psi_Q = SBottom'$› ‹$(p \cdot xvec) \sharp* P$› ‹$A_R \sharp* \Psi$› ‹$A_R \sharp* R$› ‹$A_R \sharp* M$› ‹$A_R \sharp* P$›
by(*intro Comm2*) (*assumption* | *simp*)+

moreover from ‹$A_R' \sharp* P$› ‹$A_R' \sharp* Q$› ‹$A_R' \sharp* N$› *S* ‹$xvec \sharp* A_R'$› ‹$(p \cdot xvec) \sharp* A_R'$› *PTrans QTrans* ‹*distinctPerm p*› **have** $A_R' \sharp* P'$ **and** $A_R' \sharp* Q'$
apply(*drule-tac inputFreshChainDerivative*)
apply *simp*
apply(*subst pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*, *symmetric*, *of - - p*], *simp*)
apply *fastforce*
using *QTrans* ‹$A_R' \sharp* N$› ‹$A_R' \sharp* Q$› *inputFreshChainDerivative* **by** *force*
from ‹$xvec \sharp* P$› ‹$(p \cdot xvec) \sharp* N$› *PTrans* ‹*distinctPerm p*› **have** $(p \cdot xvec) \sharp* (p \cdot P')$
apply(*drule-tac inputFreshChainDerivative*)
apply *simp*
by(*subst pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*, *symmetric*, *of - - p*], *simp*)+

{
from ‹$(\Psi \otimes \Psi_R, P', T \parallel !P) \in Rel$› **have** $(p \cdot (\Psi \otimes \Psi_R), (p \cdot P'), p \cdot (T \parallel !P)) \in Rel$
by(*rule Closed*)
with ‹$xvec \sharp* \Psi$› ‹$(p \cdot xvec) \sharp* \Psi$› ‹$xvec \sharp* P$› ‹$(p \cdot xvec) \sharp* P$› *S* **have** $(\Psi \otimes (p \cdot \Psi_R), p \cdot P', (p \cdot T) \parallel !P) \in Rel$
by(*simp add*: *eqvts*)
then have $((\Psi \otimes (p \cdot \Psi_R)) \otimes \Psi', p \cdot P', (p \cdot T) \parallel !P) \in Rel$ **by**(*rule cExt*)
with ‹$(p \cdot \Psi_R) \otimes \Psi' \simeq \Psi_R'$› **have** $(\Psi \otimes \Psi_R', (p \cdot P'), (p \cdot T) \parallel !P) \in Rel$
by(*metis Associativity StatEq compositionSym*)
with *FrR'* ‹$A_R' \sharp* \Psi$› ‹$A_R' \sharp* P'$› ‹$A_R' \sharp* P$› ‹$xvec \sharp* A_R'$› ‹$(p \cdot xvec) \sharp* A_R'$› *S* ‹*distinctPerm p*› *suppT*
have $(\Psi, R' \parallel (p \cdot P'), R' \parallel ((p \cdot T) \parallel !P)) \in Rel$
apply(*intro FrameParPres*)
apply(*assumption* | *simp add*: *freshChainSimps*)+
by(*auto simp add*: *fresh-star-def fresh-def*)
then have $(\Psi, R' \parallel (p \cdot P'), (R' \parallel (p \cdot T)) \parallel !P) \in Rel$ **by**(*blast intro*: *Assoc Trans*)
with ‹$xvec \sharp* \Psi$› ‹$xvec \sharp* P$› **have** $(\Psi, (\!|\nu*xvec|\!)(R' \parallel (p \cdot P')), ((\!|\nu*xvec|\!)(R' \parallel (p \cdot T))) \parallel !P) \in Rel$
by(*metis ResPres psiFreshVec ScopeExt Trans*)
then have $(\Psi, (\!|\nu*(p \cdot xvec)|\!)((p \cdot R') \parallel P'), ((\!|\nu*xvec|\!)(R' \parallel (p \cdot T))) \parallel !P) \in Rel$
using ‹$(p \cdot xvec) \sharp* R'$› ‹$(p \cdot xvec) \sharp* (p \cdot P')$› *S* ‹*distinctPerm p*›
apply(*erule-tac rev-mp*)
by(*subst resChainAlpha*[*of p*]) *auto*
moreover from ‹$(\Psi, P, Q) \in Rel$› ‹*guarded P*› ‹*guarded Q*› **have** $(\Psi, ((\!|\nu*xvec|\!)(R' \parallel (p \cdot T))) \parallel !P, ((\!|\nu*xvec|\!)(R' \parallel (p \cdot T))) \parallel !Q) \in Rel'$

by(*rule C1*)
moreover from ‹$(\Psi \otimes \Psi_R,\ (p \cdot Q'),\ S \parallel !Q) \in Rel$› ‹$(\Psi \otimes \Psi_R,\ S,\ T) \in Rel$› **have** $(\Psi \otimes \Psi_R,\ (p \cdot Q'),\ T \parallel !Q) \in Rel$
by(*blast intro*: *ParPres Trans*)
then have $(p \cdot (\Psi \otimes \Psi_R),\ p \cdot p \cdot Q',\ p \cdot (T \parallel !Q)) \in Rel$ **by**(*rule Closed*)
with *S* ‹*xvec* $\sharp*$ Ψ› ‹$(p \cdot xvec) \sharp* \Psi$› ‹*xvec* $\sharp*$ $(!Q)$› ‹$(p \cdot xvec) \sharp* Q$› ‹*distinctPerm p*›
have $(\Psi \otimes (p \cdot \Psi_R),\ Q',\ (p \cdot T) \parallel !Q) \in Rel$ **by**(*simp add*: *eqvts*)
then have $((\Psi \otimes (p \cdot \Psi_R)) \otimes \Psi',\ Q',\ (p \cdot T) \parallel !Q) \in Rel$ **by**(*rule cExt*)
with ‹$(p \cdot \Psi_R) \otimes \Psi' \simeq \Psi_R{}'$› **have** $(\Psi \otimes \Psi_R{}',\ Q',\ (p \cdot T) \parallel !Q) \in Rel$
by(*metis Associativity StatEq compositionSym*)
with *FrR'* ‹$A_R{}' \sharp* \Psi$› ‹$A_R{}' \sharp* P'$› ‹$A_R{}' \sharp* Q'$› ‹$A_R{}' \sharp* Q$› *suppT suppS* ‹*xvec* $\sharp*$ $A_R{}'$› ‹$(p \cdot xvec) \sharp* A_R{}'$› *S* ‹*distinctPerm p*›
have $(\Psi,\ R' \parallel Q',\ R' \parallel ((p \cdot T) \parallel !Q)) \in Rel$
apply(*intro FrameParPres*)
apply(*assumption* | *simp*)+
apply(*simp add*: *freshChainSimps*)
by(*auto simp add*: *fresh-star-def fresh-def*)
then have $(\Psi,\ R' \parallel Q',\ (R' \parallel (p \cdot T)) \parallel !Q) \in Rel$ **by**(*blast intro*: *Assoc Trans*)
with ‹*xvec* $\sharp*$ Ψ› ‹*xvec* $\sharp*$ $(!Q)$› **have** $(\Psi,\ (\!|\nu*xvec|\!)(R' \parallel Q'),\ ((\!|\nu*xvec|\!)(R' \parallel (p \cdot T))) \parallel !Q) \in Rel$
by(*metis ResPres psiFreshVec ScopeExt Trans*)
ultimately have $(\Psi,\ (\!|\nu*(p \cdot xvec)|\!)((p \cdot R') \parallel P'),\ (\!|\nu*xvec|\!)(R' \parallel Q')) \in Rel'$ **by**(*blast intro*: *cSym Compose*)
}
ultimately show *?case* **by** *blast*
next
case(*cBrMerge* Ψ_Q *M N R'* A_R Ψ_R *Q'* A_Q)
from ‹*extractFrame* $(!Q) = \langle A_Q,\ \Psi_Q\rangle$› **have** $A_Q = []$ **and** $\Psi_Q = SBottom'$ **by** *simp*+
have *RTrans*: $\Psi \otimes \Psi_Q \rhd R \longmapsto ¿M(\!|N|\!) \prec R'$ **and** *FrR*: *extractFrame R* $= \langle A_R,\ \Psi_R\rangle$ **by** *fact*+
have *QTrans*: $\Psi \otimes \Psi_R \rhd !Q \longmapsto ¿M(\!|N|\!) \prec Q'$ **by** *fact*
with ‹$(\Psi,\ P,\ Q) \in Rel$› ‹*guarded P*›
obtain *P' S T* **where** *PTrans*: $\Psi \otimes \Psi_R \rhd !P \longmapsto ¿M(\!|N|\!) \prec P'$ **and** $(\Psi \otimes \Psi_R,\ P',\ T \parallel !P) \in Rel$
and $(\Psi \otimes \Psi_R,\ Q',\ S \parallel !Q) \in Rel$ **and** $(\Psi \otimes \Psi_R,\ S,\ T) \in Rel$
and *suppT*: $((supp\ T)::name\ set) \subseteq supp\ P'$ **and** *suppS*: $((supp\ S)::name\ set) \subseteq supp\ Q'$
by(*drule-tac cSym*) (*auto dest*: *Der intro*: *cExt*)
with *RTrans QTrans FrR* **have** $\Psi \rhd R \parallel !P \longmapsto ¿M(\!|N|\!) \prec (R' \parallel P')$
using *PTrans* ‹$\Psi_Q = SBottom'$› ‹$A_R \sharp* \Psi$› ‹$A_R \sharp* R$› ‹$A_R \sharp* M$› ‹$A_R \sharp* P$›
by(*intro BrMerge*) (*assumption* | *simp*)+

moreover from ‹$A_R \sharp* P$› ‹$A_R \sharp* (!Q)$› ‹$A_R \sharp* N$› *PTrans QTrans*
have $A_R \sharp* P'$ **and** $A_R \sharp* Q'$ **by**(*force dest*: *brinputFreshChainDerivative*)+
moreover with *RTrans FrR* ‹*distinct* A_R› ‹$A_R \sharp* R$› ‹$A_R \sharp* N$› ‹$A_R \sharp* \Psi$› ‹$A_R \sharp* P$› ‹$A_R \sharp* (!Q)$› ‹$A_R \sharp* M$›

obtain Ψ' $A_R{}'$ $\Psi_R{}'$ **where** *FrR'*: *extractFrame* $R' = \langle A_R{}', \Psi_R{}'\rangle$ **and** $\Psi_R \otimes \Psi' \simeq \Psi_R{}'$ **and** $A_R{}' \sharp* \Psi$
and $A_R{}' \sharp* P'$ **and** $A_R{}' \sharp* Q'$ **and** $A_R{}' \sharp* P$ **and** $A_R{}' \sharp* Q$
by(*auto intro*: *expandFrame*[**where** C=(Ψ, P, P', Q, Q') **and** C'=Ψ])

moreover
{
from ‹($\Psi \otimes \Psi_R$, P', $T \parallel !P$) $\in$ *Rel*› **have** (($\Psi \otimes \Psi_R$) $\otimes \Psi'$, P', $T \parallel !P$) $\in$ *Rel* **by**(*rule cExt*)
with ‹$\Psi_R \otimes \Psi' \simeq \Psi_R{}'$› **have** ($\Psi \otimes \Psi_R{}'$, P', $T \parallel !P$) $\in$ *Rel*
by(*metis Associativity StatEq compositionSym*)
with *FrR'* ‹$A_R{}' \sharp* \Psi$› ‹$A_R{}' \sharp* P'$› ‹$A_R{}' \sharp* P$› *suppT* **have** (Ψ, $R' \parallel P'$, $R' \parallel (T \parallel !P)$) $\in$ *Rel*
by(*intro FrameParPres*) (*auto simp add*: *fresh-star-def fresh-def psi.supp*)
then have *one*: (Ψ, $R' \parallel P'$, $(R' \parallel T) \parallel !P$) $\in$ *Rel* **by**(*blast intro*: *Assoc Trans*)
from ‹(Ψ, P, Q) $\in$ *Rel*› ‹*guarded P*› ‹*guarded Q*› **have** *two*: (Ψ, $(R' \parallel T) \parallel !P$, $(R' \parallel T) \parallel !Q$) $\in$ *Rel'*
by(*rule C1*)
from ‹($\Psi \otimes \Psi_R$, Q', $S \parallel !Q$) $\in$ *Rel*› ‹($\Psi \otimes \Psi_R$, S, T) $\in$ *Rel*› **have** ($\Psi \otimes \Psi_R$, Q', $T \parallel !Q$) $\in$ *Rel*
by(*blast intro*: *ParPres Trans*)
then have (($\Psi \otimes \Psi_R$) $\otimes \Psi'$, Q', $T \parallel !Q$) $\in$ *Rel* **by**(*rule cExt*)
with ‹$\Psi_R \otimes \Psi' \simeq \Psi_R{}'$› **have** ($\Psi \otimes \Psi_R{}'$, Q', $T \parallel !Q$) $\in$ *Rel*
by(*metis Associativity StatEq compositionSym*)
with *FrR'* ‹$A_R{}' \sharp* \Psi$› ‹$A_R{}' \sharp* P'$› ‹$A_R{}' \sharp* Q'$› ‹$A_R{}' \sharp* Q$› *suppT suppS* **have** (Ψ, $R' \parallel Q'$, $R' \parallel (T \parallel !Q)$) $\in$ *Rel*
by(*intro FrameParPres*) (*auto simp add*: *fresh-star-def fresh-def psi.supp*)
then have *three*: (Ψ, $R' \parallel Q'$, $(R' \parallel T) \parallel !Q$) $\in$ *Rel* **by**(*blast intro*: *Assoc Trans*)
from *one two three* **have** (Ψ, $(R' \parallel P')$, $(R' \parallel Q')$) $\in$ *Rel'* **by**(*blast intro*: *cSym Compose*)
}
ultimately show *?case* **by** *blast*
next
case(*cBrComm1* Ψ_Q M N R' A_R Ψ_R *xvec* Q' A_Q)
from ‹¡M(⦇ν*$xvec$⦈)⟨N⟩ = α› **have** *xvec*=*bn* α
by(*auto simp add*: *action.inject*)
from ‹*xvec* = *bn* α› ‹*bn* α $\sharp*$ P› ‹*bn* α $\sharp*$ R›
have *xvec* $\sharp*$ P **and** *xvec* $\sharp*$ R **by** *simp*+

from ‹*extractFrame* (!Q) = $\langle A_Q, \Psi_Q\rangle$› **have** A_Q = [] **and** Ψ_Q = *SBottom'* **by** *simp*+
have *RTrans*: $\Psi \otimes \Psi_Q \rhd R \longmapsto$¡$M$(⦇$N$⦈) $\prec$ R' **and** *FrR*: *extractFrame* $R = \langle A_R, \Psi_R\rangle$ **by** *fact*+
moreover have *QTrans*: $\Psi \otimes \Psi_R \rhd !Q \longmapsto$¡$M$(⦇$\nu$*$xvec$⦈)⟨$N$⟩ $\prec$ Q' **by** *fact*
from *FrR* ‹*xvec* $\sharp*$ R› ‹A_R $\sharp*$ *xvec*› **have** *xvec* $\sharp*$ Ψ_R **by**(*force dest*: *extractFrameFreshChain*)
with *QTrans* ‹(Ψ, P, Q) $\in$ *Rel*› ‹*xvec* $\sharp*$ Ψ›‹*xvec* $\sharp*$ P› ‹*xvec* $\sharp*$ (!Q)› ‹*guarded*

$P\rangle$ ‹*xvec* $\sharp*$ $M\rangle$
obtain P' S T **where** *PTrans*: $\Psi \otimes \Psi_R \rhd !P \longmapsto ¡M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P'$ **and** $(\Psi \otimes \Psi_R,\ P',\ T \parallel !P) \in Rel$
and $(\Psi \otimes \Psi_R,\ Q',\ S \parallel !Q) \in Rel$ **and** $(\Psi \otimes \Psi_R,\ S,\ T) \in Rel$
and *suppT*: $((supp\ T)::name\ set) \subseteq supp\ P'$ **and** *suppS*: $((supp\ S)::name\ set) \subseteq supp\ Q'$
apply(*drule-tac cSym*)
by(*metis Der* ‹*bn* $\alpha \sharp*$ $Q\rangle$ ‹*xvec* = *bn* $\alpha\rangle$ *cBrComm1.hyps*(*30*) *cExt cSim.hyps*(*6*) *freshCompChain*(*1*))

ultimately have $\Psi \rhd R \parallel !P \longmapsto ¡M(\!|\nu{*}xvec|\!)\langle N\rangle \prec (R' \parallel P')$
using *PTrans* ‹Ψ_Q = *SBottom*'› ‹*xvec* $\sharp*$ $R\rangle$ ‹$A_R \sharp*$ $\Psi\rangle$ ‹$A_R \sharp*$ $R\rangle$ ‹$A_R \sharp*$ $M\rangle$ ‹$A_R \sharp*$ $P\rangle$
by(*intro BrComm1*) (*assumption* | *simp*)+

moreover from ‹$A_R \sharp*$ $P\rangle$ ‹$A_R \sharp*$ $(!Q)\rangle$ ‹$A_R \sharp*$ *xvec*› *PTrans QTrans* ‹*xvec* $\sharp*$ $M\rangle$ ‹*distinct xvec*›
have $A_R \sharp*$ P' **and** $A_R \sharp*$ Q' **by**(*force dest*: *broutputFreshChainDerivative*)+
moreover with *RTrans FrR* ‹*distinct* $A_R\rangle$ ‹$A_R \sharp*$ $R\rangle$ ‹$A_R \sharp*$ $N\rangle$ ‹$A_R \sharp*$ $\Psi\rangle$ ‹$A_R \sharp*$ $P\rangle$ ‹$A_R \sharp*$ $(!Q)\rangle$ ‹$A_R \sharp*$ $M\rangle$
obtain Ψ' A_R' Ψ_R' **where** *FrR'*: *extractFrame* $R' = \langle A_R', \Psi_R'\rangle$ **and** $\Psi_R \otimes \Psi' \simeq \Psi_R'$ **and** $A_R' \sharp*$ Ψ
and $A_R' \sharp*$ P' **and** $A_R' \sharp*$ Q' **and** $A_R' \sharp*$ P **and** $A_R' \sharp*$ Q
by(*auto intro*: *expandFrame*[**where** C=(Ψ, P, P', Q, Q') **and** C'=Ψ])

moreover
{
from ‹$(\Psi \otimes \Psi_R,\ P',\ T \parallel !P) \in Rel\rangle$ **have** $((\Psi \otimes \Psi_R) \otimes \Psi',\ P',\ T \parallel !P) \in Rel$ **by**(*rule cExt*)
with ‹$\Psi_R \otimes \Psi' \simeq \Psi_R'\rangle$ **have** $(\Psi \otimes \Psi_R',\ P',\ T \parallel !P) \in Rel$
by(*metis Associativity StatEq compositionSym*)
with *FrR'* ‹$A_R' \sharp*$ $\Psi\rangle$ ‹$A_R' \sharp*$ $P'\rangle$ ‹$A_R' \sharp*$ $P\rangle$ *suppT* **have** $(\Psi,\ R' \parallel P',\ R' \parallel (T \parallel !P)) \in Rel$
by(*intro FrameParPres*) (*auto simp add*: *fresh-star-def fresh-def psi.supp*)
then have *one*: $(\Psi,\ R' \parallel P',\ (R' \parallel T) \parallel !P) \in Rel$ **by**(*blast intro*: *Assoc Trans*)
from ‹$(\Psi,\ P,\ Q) \in Rel\rangle$ ‹*guarded* $P\rangle$ ‹*guarded* $Q\rangle$ **have** *two*: $(\Psi,\ (R' \parallel T) \parallel !P,\ (R' \parallel T) \parallel !Q) \in Rel'$
by(*rule C1*)
from ‹$(\Psi \otimes \Psi_R,\ Q',\ S \parallel !Q) \in Rel\rangle$ ‹$(\Psi \otimes \Psi_R,\ S,\ T) \in Rel\rangle$ **have** $(\Psi \otimes \Psi_R,\ Q',\ T \parallel !Q) \in Rel$
by(*blast intro*: *ParPres Trans*)
then have $((\Psi \otimes \Psi_R) \otimes \Psi',\ Q',\ T \parallel !Q) \in Rel$ **by**(*rule cExt*)
with ‹$\Psi_R \otimes \Psi' \simeq \Psi_R'\rangle$ **have** $(\Psi \otimes \Psi_R',\ Q',\ T \parallel !Q) \in Rel$
by(*metis Associativity StatEq compositionSym*)
with *FrR'* ‹$A_R' \sharp*$ $\Psi\rangle$ ‹$A_R' \sharp*$ $P'\rangle$ ‹$A_R' \sharp*$ $Q'\rangle$ ‹$A_R' \sharp*$ $Q\rangle$ *suppT suppS* **have** $(\Psi,\ R' \parallel Q',\ R' \parallel (T \parallel !Q)) \in Rel$
by(*intro FrameParPres*) (*auto simp add*: *fresh-star-def fresh-def psi.supp*)
then have *three*: $(\Psi,\ R' \parallel Q',\ (R' \parallel T) \parallel !Q) \in Rel$ **by**(*blast intro*: *Assoc*

Trans)
from *one two three* **have** $(\Psi, (R' \parallel P'), (R' \parallel Q')) \in Rel'$ **by**(*blast intro*: *cSym Compose*)
}
ultimately show *?case* **by** *blast*
next
case(*cBrComm2* Ψ_Q M *xvec* N R' A_R Ψ_R Q' A_Q)
from ‹¡M(⟪ν*$xvec$⟫)⟨N⟩ = α› **have** *xvec* = *bn* α
by(*auto simp add*: *action.inject*)
from ‹*xvec* = *bn* α› ‹*bn* α ♯* P› ‹*bn* α ♯* R›
have *xvec* ♯* P **and** *xvec* ♯* R **by** *simp*+

from ‹*extractFrame* (!Q) = ⟨A_Q, Ψ_Q⟩› **have** $A_Q = []$ **and** Ψ_Q = *SBottom'* **by** *simp*+
have *RTrans*: $\Psi \otimes \Psi_Q \rhd R \longmapsto$¡$M$(⟪$\nu$*$xvec$⟫)⟨$N$⟩ ≺ R' **and** *FrR*: *extractFrame* R = ⟨A_R, Ψ_R⟩ **by** *fact*+
then obtain p Ψ' A_R' Ψ_R' **where** *S*: *set* $p \subseteq$ *set xvec* × *set*(p · *xvec*)
and *FrR'*: *extractFrame* R' = ⟨A_R', Ψ_R'⟩ **and** (p · Ψ_R) ⊗ $\Psi' \simeq \Psi_R'$ **and** A_R' ♯* Ψ
and A_R' ♯* N **and** A_R' ♯* M **and** A_R' ♯* R **and** A_R' ♯* R' **and** A_R' ♯* P **and** A_R' ♯* Q **and** (*p* · *xvec*) ♯* Ψ
and (*p* · *xvec*) ♯* P **and** (*p* · *xvec*) ♯* Q **and** *xvec* ♯* A_R' **and** (*p* · *xvec*) ♯* A_R'
and *distinctPerm p* **and** (*p* · *xvec*) ♯* R **and** (*p* · *xvec*) ♯* R' **and** (*p* · *xvec*) ♯* N **and** (*p* · *xvec*) ♯* M
using ‹*distinct* A_R› ‹A_R ♯* R› ‹A_R ♯* M› ‹A_R ♯* *xvec*› ‹A_R ♯* N› ‹A_R ♯* Ψ› ‹A_R ♯* P› ‹A_R ♯* (!Q)›
‹*xvec* ♯* Ψ› ‹*xvec* ♯* P› ‹*xvec* ♯* (!Q)› ‹*xvec* ♯* R› ‹*xvec* ♯* M› ‹*distinct xvec*›
by(*auto intro*: *expandFrame*[**where** C=(Ψ, P, R, Q, M) **and** C'=(Ψ, P, R, Q, M)])

from *RTrans* S ‹(*p* · *xvec*) ♯* N› ‹(*p* · *xvec*) ♯* R'› **have** $\Psi \otimes \Psi_Q \rhd R \longmapsto$¡$M$(⟪$\nu$*(*p* · *xvec*)⟫)⟨(*p* · N)⟩ ≺ (*p* · R')
apply(*simp add*: *residualInject*)
by(*subst boundOutputChainAlpha''*[*symmetric*]) *auto*

moreover have *QTrans*: $\Psi \otimes \Psi_R \rhd$!$Q \longmapsto$¿M(⟪N⟫) ≺ Q' **by** *fact*
with *QTrans* S ‹(*p* · *xvec*) ♯* N› **have** $\Psi \otimes \Psi_R \rhd$!$Q \longmapsto$¿M(⟪(*p* · N)⟫) ≺ (*p* · Q') **using** ‹*distinctPerm p*› ‹*xvec* ♯* (!Q)› ‹(*p* · *xvec*) ♯* Q›
by(*intro brinputAlpha*) *auto*
with ‹(Ψ, P, Q) ∈ *Rel*› ‹*guarded* P›
obtain P' S T **where** *PTrans*: $\Psi \otimes \Psi_R \rhd$!$P \longmapsto$¿M(⟪(*p* · N)⟫) ≺ P' **and** (Ψ ⊗ Ψ_R, P', $T \parallel$!P) ∈ *Rel*
and ($\Psi \otimes \Psi_R$, (*p* · Q'), $S \parallel$!Q) ∈ *Rel* **and** ($\Psi \otimes \Psi_R$, S, T) ∈ *Rel*
and *suppT*: ((*supp* T)::*name set*) ⊆ *supp* P' **and** *suppS*: ((*supp* S)::*name set*) ⊆ *supp*(*p* · Q')
by(*drule-tac cSym*) (*auto dest*: *Der cExt*)
ultimately have $\Psi \rhd R \parallel$!$P \longmapsto$¡M(⟪ν*(*p* · *xvec*)⟫)⟨(*p* · N)⟩ ≺ ((*p* · R') ∥ P')
using *PTrans FrR* ‹Ψ_Q = *SBottom'*› ‹(*p* · *xvec*) ♯* P› ‹A_R ♯* Ψ› ‹A_R ♯* R›

$\langle A_R \sharp* M\rangle$ $\langle A_R \sharp* P\rangle$
by(*intro BrComm2*) (*assumption* | *simp*)+
then have $p \cdot (\Psi \rhd R \parallel !P \longmapsto\!\!\!¡ M(\!|\nu*(p \cdot xvec)|\!)\langle(p \cdot N)\rangle \prec ((p \cdot R') \parallel P'))$ **by** *simp*
with $\langle distinctPerm\ p\rangle$ **have** $(p \cdot \Psi) \rhd (p \cdot R) \parallel !(p \cdot P) \longmapsto\!\!\!¡ (p \cdot M)(\!|\nu*xvec|\!)\langle N\rangle \prec (R' \parallel (p \cdot P'))$
by(*simp add*: *eqvts*)
with *S* $\langle xvec \sharp* \Psi\rangle$ $\langle(p \cdot xvec) \sharp* \Psi\rangle$ $\langle xvec \sharp* R\rangle$ $\langle(p \cdot xvec) \sharp* R\rangle$ $\langle xvec \sharp* P\rangle$ $\langle(p \cdot xvec) \sharp* P\rangle$ $\langle xvec \sharp* M\rangle$ $\langle(p \cdot xvec) \sharp* M\rangle$
have $\Psi \rhd R \parallel !P \longmapsto\!\!\!¡ M(\!|\nu*xvec|\!)\langle N\rangle \prec (R' \parallel (p \cdot P'))$
by *simp*

moreover from $\langle A_R{}' \sharp* P\rangle$ $\langle A_R{}' \sharp* Q\rangle$ $\langle A_R{}' \sharp* N\rangle$ *S* $\langle xvec \sharp* A_R{}'\rangle$ $\langle(p \cdot xvec) \sharp* A_R{}'\rangle$ *PTrans QTrans* $\langle distinctPerm\ p\rangle$ **have** $A_R{}' \sharp* P'$ **and** $A_R{}' \sharp* Q'$
apply(*drule-tac brinputFreshChainDerivative*, *simp*)
apply(*subst pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*, *symmetric*, *of - - p*], *simp*)
apply *force*
using *QTrans* $\langle A_R{}' \sharp* N\rangle$ $\langle A_R{}' \sharp* Q\rangle$ *brinputFreshChainDerivative* **by** *force*
from $\langle xvec \sharp* P\rangle$ $\langle(p \cdot xvec) \sharp* N\rangle$ *PTrans* $\langle distinctPerm\ p\rangle$ **have** $(p \cdot xvec) \sharp* (p \cdot P')$
apply(*drule-tac brinputFreshChainDerivative*, *simp*)
apply(*subst pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*, *symmetric*, *of - - p*], *simp*)
by(*subst pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*, *symmetric*, *of - - p*], *simp*)

{
from $\langle(\Psi \otimes \Psi_R, P', T \parallel !P) \in Rel\rangle$ **have** $(p \cdot (\Psi \otimes \Psi_R), (p \cdot P'), p \cdot (T \parallel !P)) \in Rel$
by(*rule Closed*)
with $\langle xvec \sharp* \Psi\rangle$ $\langle(p \cdot xvec) \sharp* \Psi\rangle$ $\langle xvec \sharp* P\rangle$ $\langle(p \cdot xvec) \sharp* P\rangle$ *S* **have** $(\Psi \otimes (p \cdot \Psi_R), p \cdot P', (p \cdot T) \parallel !P) \in Rel$
by(*simp add*: *eqvts*)
then have $((\Psi \otimes (p \cdot \Psi_R)) \otimes \Psi', p \cdot P', (p \cdot T) \parallel !P) \in Rel$
by(*rule cExt*)
with $\langle(p \cdot \Psi_R) \otimes \Psi' \simeq \Psi_R{}'\rangle$ **have** $(\Psi \otimes \Psi_R{}', (p \cdot P'), (p \cdot T) \parallel !P) \in Rel$
by(*metis Associativity StatEq compositionSym*)
with *FrR'* $\langle A_R{}' \sharp* \Psi\rangle$ $\langle A_R{}' \sharp* P'\rangle$ $\langle A_R{}' \sharp* P\rangle$ $\langle xvec \sharp* A_R{}'\rangle$ $\langle(p \cdot xvec) \sharp* A_R{}'\rangle$ *S* $\langle distinctPerm\ p\rangle$ *suppT*
have $(\Psi, R' \parallel (p \cdot P'), R' \parallel ((p \cdot T) \parallel !P)) \in Rel$
apply(*intro FrameParPres*)
apply(*assumption* | *simp add*: *freshChainSimps*)+
by(*auto simp add*: *fresh-star-def fresh-def*)
then have *one*: $(\Psi, R' \parallel (p \cdot P'), (R' \parallel (p \cdot T)) \parallel !P) \in Rel$ **by**(*blast intro*: *Assoc Trans*)
from $\langle(\Psi, P, Q) \in Rel\rangle$ $\langle guarded\ P\rangle$ $\langle guarded\ Q\rangle$ **have** *two*: $(\Psi, (R' \parallel (p \cdot T)) \parallel !P, (R' \parallel (p \cdot T)) \parallel !Q) \in Rel'$
by(*rule C1*)

from ‹$(\Psi \otimes \Psi_R,\ (p \cdot Q'),\ S \parallel !Q) \in Rel$› ‹$(\Psi \otimes \Psi_R,\ S,\ T) \in Rel$› **have** $(\Psi \otimes \Psi_R,\ (p \cdot Q'),\ T \parallel !Q) \in Rel$
by(*blast intro*: *ParPres Trans*)
then have $(p \cdot (\Psi \otimes \Psi_R),\ p \cdot p \cdot Q',\ p \cdot (T \parallel !Q)) \in Rel$ **by**(*rule Closed*)
with *S* ‹*xvec* ♯* Ψ› ‹$(p \cdot xvec)$ ♯* Ψ› ‹*xvec* ♯* $(!Q)$› ‹$(p \cdot xvec)$ ♯* *Q*› ‹*distinctPerm p*›
have $(\Psi \otimes (p \cdot \Psi_R),\ Q',\ (p \cdot T) \parallel !Q) \in Rel$ **by**(*simp add*: *eqvts*)
then have $((\Psi \otimes (p \cdot \Psi_R)) \otimes \Psi',\ Q',\ (p \cdot T) \parallel !Q) \in Rel$ **by**(*rule cExt*)
with ‹$(p \cdot \Psi_R) \otimes \Psi' \simeq {\Psi_R}'$› **have** $(\Psi \otimes {\Psi_R}',\ Q',\ (p \cdot T) \parallel !Q) \in Rel$
by(*metis Associativity StatEq compositionSym*)
with *FrR'* ‹${A_R}'$ ♯* Ψ› ‹${A_R}'$ ♯* *P'*› ‹${A_R}'$ ♯* *Q'*› ‹${A_R}'$ ♯* *Q*› *suppT suppS* ‹*xvec* ♯* ${A_R}'$› ‹$(p \cdot xvec)$ ♯* ${A_R}'$› *S* ‹*distinctPerm p*›
have $(\Psi,\ R' \parallel Q',\ R' \parallel ((p \cdot T) \parallel !Q)) \in Rel$
apply(*intro FrameParPres*)
apply(*assumption* | *simp*)+
apply(*simp add*: *freshChainSimps*)
by(*auto simp add*: *fresh-star-def fresh-def*)
then have *three*: $(\Psi,\ R' \parallel Q',\ (R' \parallel (p \cdot T)) \parallel !Q) \in Rel$ **by**(*blast intro*: *Assoc Trans*)
from *one two three* **have** $(\Psi,\ (R' \parallel (p \cdot P')),\ (R' \parallel Q')) \in Rel'$ **by**(*blast intro*: *cSym Compose*)
}
ultimately show *?case* **by** *blast*
qed
qed

unbundle *relcomp-syntax*

end

end
theory *Sim-Struct-Cong*
imports *Simulation HOL−Library.Multiset*
begin

This file is a (heavily modified) variant of the theory *Psi_Calculi.Sim_Struct_Cong* from [1].

lemma *partitionListLeft*:
assumes *xs@ys=xs'@y#ys'*
and $y \in set\ xs$
and *distinct*(*xs@ys*)

obtains *zs* **where** $xs = xs'@y\#zs$ **and** $ys'=zs@ys$
using *assms*
by(*force simp add*: *append-eq-append-conv2 append-eq-Cons-conv*)

lemma *partitionListRight*:

assumes *xs@ys=xs'@y#ys'*

and $y \in set\ ys$
and $distinct(xs@ys)$

obtains zs **where** $xs' = xs@zs$ **and** $ys{=}zs@y\#ys'$
using *assms*
by(*force simp add*: *append-eq-append-conv2 append-eq-Cons-conv*)

context *env* **begin**

lemma *resOutputCases''''*[*consumes 8*, *case-names cOpen cRes*]:
fixes Ψ :: $'b$
and x :: *name*
and *zvec* :: *name list*
and P :: $('a, 'b, 'c)$ *psi*
and α :: $'a$ *action*
and P' :: $('a, 'b, 'c)$ *psi*
and C :: $'f$::*fs-name*

assumes *Trans*: $\Psi \rhd (\!|\nu x|\!)P \longmapsto \alpha \prec P'$
and *1*: $x \sharp \Psi$
and *2*: $x \sharp \alpha$
and *3*: $x \sharp P'$
and *4*: $bn\ \alpha \sharp* \Psi$
and *5*: $bn\ \alpha \sharp* P$
and *6*: $bn\ \alpha \sharp* subject\ \alpha$
and $\alpha = M(\!|\nu* zvec|\!)\langle N\rangle$
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 $Prop\ \alpha\ P'$
proof −
from *Trans* **have** *distinct* $(bn\ \alpha)$ **by**(*auto dest*: *boundOutputDistinct*)
show *?thesis* **using** *Trans 1 2 3 4 5 6* ‹$\alpha{=}M(\!|\nu*zvec|\!)\langle N\rangle$› *rOpen rScope*
proof(*induct rule*: *resCases'*[**where** $C{=}(zvec, C)$])
case *cBrOpen*
then show *?case*
by(*auto simp add*: *residualInject boundOutputApp*)
next
case *cRes*
then show *?case*
by(*auto simp add*: *residualInject boundOutputApp*)
next

case *cBrClose*
then show *?case*
by(*auto simp add*: *residualInject boundOutputApp*)
next
case(*cOpen M′ xvec yvec y N′ P′*)
show *?case*
using ‹$\Psi \rhd [(x,\ y)] \cdot P \longmapsto M'\langle\!|\nu{*}(xvec\ @\ yvec)|\!\rangle\langle N'\rangle \prec P'$› ‹$y \in supp\ N'$› ‹$x \sharp N'$› ‹$x \sharp P'$›
‹$x \neq y$› ‹$y \sharp xvec$› ‹$y \sharp yvec$› ‹$y \sharp M'$› ‹*distinct xvec*› ‹*distinct yvec*› ‹$xvec \sharp{*} \Psi$›
‹$y \sharp \Psi$› ‹$yvec \sharp{*} \Psi$› ‹$xvec \sharp{*} P$› ‹$y \sharp P$› ‹$yvec \sharp{*} P$› ‹$xvec \sharp{*} M'$› ‹$y \sharp M'$›
‹$yvec \sharp{*} M'$› ‹$xvec \sharp{*} yvec$›
by(*rule cOpen*(*22*))
qed
qed

lemma *resOutputCases′′′′′*[*consumes 7*, *case-names cOpen cRes*]:
fixes Ψ :: *′b*
and x :: *name*
and *zvec* :: *name list*
and P :: (*′a*, *′b*, *′c*) *psi*
and P' :: (*′a*, *′b*, *′c*) *psi*
and C :: *′f*::*fs-name*

assumes *Trans*: $\Psi \rhd (\!|\nu x|\!)P \longmapsto M(\!|\nu{*}zvec|\!)\langle N\rangle \prec P'$
and *1*: $x \sharp \Psi$
and $x \sharp M(\!|\nu{*}zvec|\!)\langle N\rangle$
and *3*: $x \sharp P'$
and $zvec \sharp{*} \Psi$
and $zvec \sharp{*} P$
and $zvec \sharp{*} M$
and *rOpen*: $\bigwedge M'\ xvec\ yvec\ y\ N'\ P'.\ [\![\Psi \rhd ([(x,\ y)] \cdot P) \longmapsto M'\langle\!|\nu{*}(xvec@yvec)|\!\rangle\langle N'\rangle \prec P';\ y \in supp\ N';$
$x \sharp N';\ x \sharp P';\ x \neq y;\ y \sharp xvec;\ y \sharp yvec;\ y \sharp M';$
distinct xvec; *distinct yvec*;
$xvec \sharp{*} \Psi;\ y \sharp \Psi;\ yvec \sharp{*} \Psi;\ xvec \sharp{*} P;\ y \sharp P;$
$yvec \sharp{*} P;\ xvec \sharp{*} M';\ y \sharp M';$
$yvec \sharp{*} M';\ xvec \sharp{*} yvec;\ M'\langle\!|\nu{*}(xvec@y\#yvec)|\!\rangle\langle N'\rangle = M(\!|\nu{*}zvec|\!)\langle N\rangle]\!] \Longrightarrow$
$Prop\ P'$
and *rScope*: $\bigwedge P'.\ [\![\Psi \rhd P \longmapsto M(\!|\nu{*}zvec|\!)\langle N\rangle \prec P']\!] \Longrightarrow Prop\ ((\!|\nu x|\!)P')$

shows $Prop\ P'$
proof −
from *Trans* **have** *distinct zvec* **by**(*auto dest*: *boundOutputDistinct*)
obtain *al* **where** $al{=}M(\!|\nu{*}zvec|\!)\langle N\rangle$ **by** *simp*
from ‹$al = M(\!|\nu{*}zvec|\!)\langle N\rangle$› *Trans* ‹$zvec \sharp{*} \Psi$› ‹$zvec \sharp{*} P$› ‹$zvec \sharp{*} M$›
have $\alpha Trans$: $\Psi \rhd (\!|\nu x|\!)P \longmapsto al \prec P'$ **and** *4*: *bn al* $\sharp{*}$ Ψ **and** *5*: *bn al* $\sharp{*}$ P **and** *6*: *bn al* $\sharp{*}$ *subject al*

```
    by simp+
  from ‹x ♯ M⦇ν∗zvec⦈⟨N⟩› ‹al=M⦇ν∗zvec⦈⟨N⟩› have 2: x ♯ al by simp
  show ?thesis using αTrans 1 2 3 4 5 6 ‹al=M⦇ν∗zvec⦈⟨N⟩› rOpen rScope
  proof(induct rule: resCases'[where C=(zvec, C)])
    case cBrOpen
    then show ?case
      by(auto simp add: residualInject boundOutputApp)
  next
    case cBrClose
    then show ?case
      by(auto simp add: residualInject boundOutputApp)
  next
    case(cOpen M' xvec yvec y N' P')
    show ?case
      using ‹Ψ ▷ [(x, y)] · P ⟼ M'⦇ν∗(xvec @ yvec)⦈⟨N'⟩ ≺ P'› ‹y ∈ supp N'›
‹x ♯ N'› ‹x ♯ P'›
        ‹x ≠ y› ‹y ♯ xvec› ‹y ♯ yvec› ‹y ♯ M'› ‹distinct xvec› ‹distinct yvec› ‹xvec ♯∗
Ψ›
        ‹y ♯ Ψ› ‹yvec ♯∗ Ψ› ‹xvec ♯∗ P› ‹y ♯ P› ‹yvec ♯∗ P› ‹xvec ♯∗ M'› ‹y ♯ M'›
       ‹yvec ♯∗ M'› ‹xvec ♯∗ yvec› ‹M'⦇ν∗(xvec @ y # yvec)⦈⟨N'⟩ = M⦇ν∗zvec⦈⟨N⟩›
      by(rule cOpen(22))
  next
    case (cRes P')
    from ‹Ψ ▷ P ⟼ al ≺ P'› ‹al = M⦇ν∗zvec⦈⟨N⟩›
    show ?case
      by (simp add: cRes(4))
  qed
qed

lemma resBrOutputCases'[consumes 7, case-names cBrOpen cRes]:
  fixes Ψ    :: 'b
    and x    :: name
    and zvec :: name list
    and P    :: ('a, 'b, 'c) psi
    and P'   :: ('a, 'b, 'c) psi
    and C    :: 'f::fs-name

assumes Trans: Ψ ▷ ⦇νx⦈P ⟼¡M⦇ν∗zvec⦈⟨N⟩ ≺ P'
  and    1: x ♯ Ψ
  and    x ♯ ¡M⦇ν∗zvec⦈⟨N⟩
  and    3: x ♯ P'
  and    zvec ♯∗ Ψ
  and    zvec ♯∗ P
  and    zvec ♯∗ M
  and  rBrOpen: ⋀M' xvec yvec y N' P'. ⟦Ψ ▷ ([(x, y)] · P) ⟼¡M'⦇ν∗(xvec@yvec)⦈⟨N'⟩
≺ P'; y ∈ supp N';
                                        x ♯ N'; x ♯ P'; x ≠ y; y ♯ xvec; y ♯ yvec; y ♯ M';
distinct xvec; distinct yvec;
                                          xvec ♯∗ Ψ; y ♯ Ψ; yvec ♯∗ Ψ; xvec ♯∗ P; y ♯ P;
```

$yvec \sharp* P$; $xvec \sharp* M'$; $y \sharp M'$;
$yvec \sharp* M'$; $xvec \sharp* yvec$; $¡M'(\!|\nu*(xvec@y\#yvec)|\!)\langle N'\rangle = ¡M(\!|\nu*zvec|\!)\langle N\rangle]\!] \Longrightarrow Prop\ P'$

and *rScope*: $\bigwedge P'.\ [\![\Psi \rhd P \longmapsto ¡M(\!|\nu*zvec|\!)\langle N\rangle \prec P']\!] \Longrightarrow Prop\ ((\!|\nu x|\!)P')$

shows *Prop* P'
proof −
from *Trans* **have** *distinct zvec* **by**(*auto dest*: *boundOutputDistinct*)
obtain *al* **where** $al{=}¡M(\!|\nu*zvec|\!)\langle N\rangle$ **by** *simp*
from ‹$al = ¡M(\!|\nu*zvec|\!)\langle N\rangle$› *Trans* ‹$zvec \sharp* \Psi$› ‹$zvec \sharp* P$› ‹$zvec \sharp* M$›
have α*Trans*: $\Psi \rhd (\!|\nu x|\!)P \longmapsto al \prec P'$ **and** *4*: $bn\ al \sharp* \Psi$ **and** *5*: $bn\ al \sharp* P$ **and** *6*: $bn\ al \sharp* subject\ al$
by *simp*+
from ‹$x \sharp ¡M(\!|\nu*zvec|\!)\langle N\rangle$› ‹$al{=}¡M(\!|\nu*zvec|\!)\langle N\rangle$› **have** *2*: $x \sharp al$ **by** *simp*
show *?thesis* **using** α*Trans 1 2 3 4 5 6* ‹$al{=}¡M(\!|\nu*zvec|\!)\langle N\rangle$› *rBrOpen rScope*
proof(*induct rule*: *resCases'*[**where** $C{=}(zvec, C)$])
case *cBrOpen*
then show *?case*
by(*auto simp add*: *residualInject boundOutputApp*)
next
case *cBrClose*
then show *?case*
by(*auto simp add*: *residualInject boundOutputApp*)
next
case(*cOpen M' xvec yvec y N' P'*)
then show *?case*
by(*auto simp add*: *residualInject boundOutputApp*)
next
case (*cRes P'*)
from ‹$\Psi \rhd P \longmapsto al \prec P'$› ‹$al = ¡M(\!|\nu*zvec|\!)\langle N\rangle$›
show *?case*
by (*simp add*: *cRes*(*4*))
qed
qed

lemma *brOutputFreshSubject*:
fixes *x*::*name*
assumes $\Psi \rhd P \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec P'$
and $xvec \sharp* M$
and $x \sharp P$
shows $x \sharp M$
using *assms*
proof(*nominal-induct avoiding*: *x rule*: *brOutputInduct'*)
case(*cAlpha* Ψ P M *xvec N P' p*)
then show *?case* **by** *simp*
next
case(*cBrOutput* Ψ M K N P)
then show *?case*

by(*auto simp add*: *fresh-def psi.supp dest*: *chanOutConSupp*)
next
case(*cCase* Ψ *P M xvec N P'* φ *Cs*) **then show** *?case*
by(*induct Cs*) *auto*
next
case(*cPar1* Ψ Ψ_Q *P M xvec N P'* A_Q *Q*) **then show** *?case*
by *simp*
next
case *cPar2* **then show** *?case* **by** *simp*
next
case *cBrComm1* **then show** *?case* **by** *simp*
next
case *cBrComm2* **then show** *?case* **by** *simp*
next
case *cBrOpen* **then show** *?case* **by**(*simp add*: *fresh-abs-fun-iff*[*OF pt-name-inst*, *OF at-name-inst*, *OF fin-supp*])
next
case *cScope* **then show** *?case* **by**(*simp add*: *fresh-abs-fun-iff*[*OF pt-name-inst*, *OF at-name-inst*, *OF fin-supp*])
next
case *cBang* **then show** *?case* **by** *simp*
qed

lemma *brInputFreshSubject*:
fixes *x*::*name*
assumes $\Psi \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$
and $x \sharp P$
shows $x \sharp M$
using *assms*
proof(*nominal-induct avoiding*: *x rule*: *brInputInduct*)
case(*cBrInput* Ψ *K M xvec N Tvec P y*)
then show *?case*
by(*auto simp add*: *fresh-def psi.supp dest*: *chanInConSupp*)
next
case(*cCase* Ψ *P M N P'* φ *Cs y*) **then show** *?case*
by(*induct Cs*) *auto*
next
case(*cPar1* Ψ Ψ_Q *P M N P'* A_Q *Q y*) **then show** *?case*
by *simp*
next
case *cPar2* **then show** *?case* **by** *simp*
next
case *cBrMerge* **then show** *?case* **by** *simp*
next
case *cScope* **then show** *?case* **by**(*simp add*: *fresh-abs-fun-iff*[*OF pt-name-inst*, *OF at-name-inst*, *OF fin-supp*])
next
case *cBang* **then show** *?case* **by** *simp*
qed

lemma *resComm*:
  fixes Ψ :: *'b*
    and *x* :: *name*
    and *y* :: *name*
    and *Rel* :: (*'b* × (*'a*, *'b*, *'c*) *psi* × (*'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*: $\bigwedge \Psi' a b Q.\ [\![a \sharp \Psi'; b \sharp \Psi']\!] \Longrightarrow (\Psi', (\!|\nu a|\!)((\!|\nu b|\!) Q), (\!|\nu b|\!)((\!|\nu a|\!) Q)) \in Rel$
  and *R3*: $\bigwedge \Psi' xvec\ yvec\ Q.\ [\![xvec \sharp* \Psi'; mset\ xvec = mset\ yvec]\!] \Longrightarrow (\Psi', (\!|\nu* xvec|\!) Q, (\!|\nu* yvec|\!) Q) \in Rel$

shows $\Psi \rhd (\!|\nu x|\!)((\!|\nu y|\!) P) \rightsquigarrow [Rel]\ (\!|\nu y|\!)((\!|\nu x|\!) P)$
proof(*cases* $x{=}y$)
  assume $x = y$
  then show *?thesis* **using** *R1*
    by(*force intro*: *reflexive*)
next
  assume $x \neq y$
  note ‹*eqvt Rel*›
  moreover from ‹$x \sharp \Psi$› ‹$y \sharp \Psi$› **have** $[x, y] \sharp* \Psi$ **by**(*simp add*: *fresh-star-def*)
  moreover have $[x, y] \sharp* (\!|\nu x|\!)((\!|\nu y|\!) P)$ **by**(*simp add*: *abs-fresh*)
  moreover have $[x, y] \sharp* (\!|\nu y|\!)((\!|\nu x|\!) P)$ **by**(*simp add*: *abs-fresh*)
  ultimately show *?thesis*
  proof(*induct rule*: *simIChainFresh*[**where** $C{=}(x, y)$])
    case(*cSim* α P')
    from ‹$bn\ \alpha \sharp* (x, y)$› ‹$bn\ \alpha \sharp* ((\!|\nu x|\!)((\!|\nu y|\!) P))$› **have** $x \sharp bn\ \alpha$ **and** $y \sharp bn\ \alpha$ **and** $bn\ \alpha \sharp* P$ **by** *simp+*
    from ‹$[x, y] \sharp* \alpha$› **have** $x \sharp \alpha$ **and** $y \sharp \alpha$ **by** *simp+*
    from ‹$[x, y] \sharp* P'$› **have** $x \sharp P'$ **and** $y \sharp P'$ **by** *simp+*
    from ‹$bn\ \alpha \sharp* P$› ‹$x \sharp \alpha$› **have** $bn\ \alpha \sharp* (\!|\nu x|\!) P$ **by**(*simp add*: *abs-fresh*)
    with ‹$\Psi \rhd (\!|\nu y|\!)((\!|\nu x|\!) P) \longmapsto \alpha \prec P'$› ‹$y \sharp \Psi$› ‹$y \sharp \alpha$› ‹$y \sharp P'$› ‹$bn\ \alpha \sharp* \Psi$›
    show *?case* **using** ‹$bn\ \alpha \sharp* subject\ \alpha$› ‹$x \sharp \alpha$› ‹$x \sharp P'$› ‹$bn\ \alpha \sharp* \Psi$› ‹$bn\ \alpha \sharp* P$› ‹$bn\ \alpha \sharp* subject\ \alpha$› ‹$y \sharp \alpha$›
    proof(*induct rule*: *resCases'*[**where** $C{=}x$])
      case(*cOpen M yvec1 yvec2 y' N P'*)
      from ‹*yvec1* $\sharp*$ *yvec2*› ‹*distinct yvec1*› ‹*distinct yvec2*› **have** *distinct*(*yvec1*@*yvec2*) **by** *auto*
        from ‹$x \sharp M(\!|\nu*(yvec1\ @\ y'\ \#\ yvec2)|\!)\langle N\rangle$› **have** $x \sharp M$ **and** $x \sharp yvec1$ **and** $x \neq y'$ **and** $x \sharp yvec2$ **and** $x \sharp N$
          by *simp+*
        from ‹$y \sharp M(\!|\nu*(yvec1\ @\ y'\ \#\ yvec2)|\!)\langle N\rangle$› **have** $y \sharp M$ **and** $y \sharp yvec1$ **and** $y \sharp yvec2$
          by *simp+*

from ‹$\Psi \rhd ([(y,\ y')] \cdot (\!|\nu x|\!) P) \longmapsto M(\!|\nu*(yvec1@yvec2)|\!)\langle N\rangle \prec P'$› ‹$x \neq y$› ‹$x \neq y'$›
have $\Psi \rhd (\!|\nu x|\!)([(y,\ y')] \cdot P) \longmapsto M(\!|\nu*(yvec1@yvec2)|\!)\langle N\rangle \prec P'$ **by**(*simp add*: *eqvts*)
moreover note ‹$x \sharp \Psi$›
moreover from ‹$x \sharp N$› ‹$x \sharp yvec1$› ‹$x \sharp yvec2$› ‹$x \sharp M$› **have** $x \sharp M(\!|\nu*(yvec1@yvec2)|\!)\langle N\rangle$ **by** *simp*
moreover note ‹$x \sharp P'$›
moreover from ‹$yvec1 \sharp* \Psi$› ‹$yvec2 \sharp* \Psi$› **have** $(yvec1@yvec2) \sharp* \Psi$ **by** *simp*
moreover from ‹$yvec1 \sharp* (\!|\nu x|\!) P$› ‹$yvec2 \sharp* (\!|\nu x|\!) P$› ‹$y \sharp yvec1$› ‹$y' \sharp yvec1$› ‹$y \sharp yvec2$› ‹$y' \sharp yvec2$› ‹$x \sharp yvec1$› ‹$x \sharp yvec2$›
have $(yvec1@yvec2) \sharp* ([(y,\ y')] \cdot P)$ **by** *simp*
moreover from ‹$yvec1 \sharp* M$› ‹$yvec2 \sharp* M$› **have** $(yvec1\ @\ yvec2) \sharp* M$
by *simp*
ultimately show *?case*
proof(*induct rule*: *resOutputCases''''*)
case(*cOpen M' xvec1 xvec2 x' N' P'*)
from ‹$M'(\!|\nu*(xvec1\ @\ x'\ \#\ xvec2)|\!)\langle N'\rangle = M(\!|\nu*(yvec1\ @\ yvec2)|\!)\langle N\rangle$› **have** $yvec1@yvec2 = xvec1@x'\#xvec2$ **and** $M = M'$ **and** $N = N'$ **by** (*simp add*: *action.inject*)+
from ‹$x \sharp yvec1$› ‹$x \sharp yvec2$› ‹$y' \sharp yvec1$› ‹$y' \sharp yvec2$› ‹$y \sharp yvec1$› ‹$y \sharp yvec2$›
have $x \sharp (yvec1@yvec2)$ **and** $y \sharp (yvec1@yvec2)$ **and** $y' \sharp (yvec1@yvec2)$ **by** *simp*+
with ‹$yvec1@yvec2 = xvec1@x'\#xvec2$›
have $x \sharp xvec1$ **and** $x \neq x'$ **and** $x \sharp xvec2$ **and** $y \sharp xvec1$ **and** $y \neq x'$ **and** $y \sharp xvec2$
and $y' \sharp xvec1$ **and** $x' \neq y'$ **and** $y' \sharp xvec2$
by *auto*

show *?case*
proof(*cases* $x' \in set\ yvec1$)
assume $x' \in set\ yvec1$

with ‹$yvec1@yvec2 = xvec1@x'\#xvec2$› ‹*distinct* $(yvec1@yvec2)$›
obtain *xvec2'* **where** *Eq1*: $yvec1 = xvec1@x'\#xvec2'$
and *Eq*: $xvec2 = xvec2'@yvec2$
by(*metis partitionListLeft*)
from ‹$\Psi \rhd ([(x,\ x')] \cdot [(y,\ y')] \cdot P) \longmapsto M'(\!|\nu*(xvec1@xvec2)|\!)\langle N'\rangle \prec P'$› ‹$y' \in supp\ N$› ‹$y' \sharp \Psi$› ‹$y' \sharp M$› ‹$y' \sharp xvec1$› ‹$y' \sharp xvec2$› *Eq* ‹$M = M'$› ‹$N = N'$›
have $\Psi \rhd (\!|\nu y'|\!)([(x,\ x')] \cdot [(y,\ y')] \cdot P) \longmapsto M'(\!|\nu*((xvec1@xvec2')@y'\#yvec2)|\!)\langle N'\rangle \prec P'$
by(*intro Open*) *auto*
then have $\Psi \rhd (\!|\nu x'|\!)((\!|\nu y'|\!)([(x,\ x')] \cdot [(y,\ y')] \cdot P)) \longmapsto M(\!|\nu*(xvec1@x'\#xvec2'@y'\#yvec2)|\!)\langle N\rangle \prec P'$
using ‹$x' \in supp\ N'$› ‹$x' \sharp \Psi$› ‹$x' \sharp M'$› ‹$x' \sharp xvec1$› ‹$x' \sharp xvec2$› ‹$x' \neq y'$› *Eq* ‹$M = M'$› ‹$N = N'$›
by(*intro Open*) *auto*
with ‹$x' \neq y'$› ‹$x \neq y'$› ‹$x' \sharp [(y,\ y')] \cdot P$›

```
      have Ψ ▷ (|νx|)((|νy'|)([(y, y')] • P)) ⟼ M(|ν*(xvec1@x'#xvec2'@y'#yvec2)|)⟨N⟩
≺ P'
        by(subst alphaRes[where y=x']) (simp add: calc-atm eqvts abs-fresh)+
       with Eq1 ‹y' ♯ (|νx|)P› ‹x ≠ y'› R1 show ?case
        by(auto simp add: alphaRes abs-fresh)
      next
       assume ¬x' ∈ set yvec1
       then have x' ♯ yvec1 by(simp add: fresh-def)
       from ‹¬x' ∈ set yvec1› ‹yvec1@yvec2 = xvec1@x'#xvec2›
       have x' ∈ set yvec2
        by(auto simp add: append-eq-append-conv2 append-eq-Cons-conv)
       with ‹yvec1@yvec2 = xvec1@x'#xvec2› ‹distinct (yvec1@yvec2)›
       obtain xvec2' where Eq: xvec1=yvec1@xvec2'
        and Eq1: yvec2=xvec2'@x'#xvec2
        by(metis partitionListRight)
       from ‹Ψ ▷ ([(x, x')] • [(y, y')] • P) ⟼ M'(|ν*(xvec1@xvec2)|)⟨N'⟩ ≺ P'›
‹y' ∈ supp N› ‹y' ♯ Ψ› ‹y' ♯ M› ‹y' ♯ xvec1› ‹y' ♯ xvec2› Eq ‹M=M'› ‹N = N'›
      have Ψ ▷ (|νy'|)([(x, x')] • [(y, y')] • P) ⟼ M'(|ν*(yvec1@y'#xvec2'@xvec2)|)⟨N'⟩
≺ P'
        by(intro Open) (assumption | simp)+
      then have Ψ ▷ (|νx'|)((|νy'|)([(x, x')] • [(y, y')] • P)) ⟼ M(|ν*((yvec1@y'#xvec2')@x'#xvec2)|)⟨N⟩
≺ P'
        using ‹x' ∈ supp N'› ‹x' ♯ Ψ› ‹x' ♯ M'› ‹x' ♯ xvec1› ‹x' ♯ xvec2› ‹x' ≠
y'› Eq ‹M=M'› ‹N=N'›
        by(intro Open) auto
       with ‹x' ≠ y'› ‹x ≠ y'› ‹x' ♯ [(y, y')] • P›
      have Ψ ▷ (|νx|)((|νy'|)([(y, y')] • P)) ⟼ M(|ν*((yvec1@y'#xvec2')@x'#xvec2)|)⟨N⟩
≺ P'
        by(subst alphaRes[where y=x']) (simp add: calc-atm eqvts abs-fresh)+
       with Eq1 ‹y' ♯ (|νx|)P› ‹x ≠ y'› R1 show ?case
        by(auto simp add: alphaRes abs-fresh)
      qed
     next
      case(cRes P')
       from ‹Ψ ▷ ([(y, y')] • P) ⟼ M(|ν*(yvec1@yvec2)|)⟨N⟩ ≺ P'› ‹y' ∈ supp
N› ‹y' ♯ Ψ› ‹y' ♯ M› ‹y' ♯ yvec1› ‹y' ♯ yvec2›
      have Ψ ▷ (|νy'|)([(y, y')] • P) ⟼ M(|ν*(yvec1@y'#yvec2)|)⟨N⟩ ≺ P' by(rule
Open)
     with ‹y' ♯ (|νx|)P› ‹x ≠ y'›have Ψ ▷ (|νy|)P ⟼ M(|ν*(yvec1@y'#yvec2)|)⟨N⟩
≺ P' by(simp add: alphaRes abs-fresh)
       then have Ψ ▷ (|νx|)((|νy|)P) ⟼ M(|ν*(yvec1@y'#yvec2)|)⟨N⟩ ≺ (|νx|)P'
using ‹x ♯ Ψ› ‹x ♯ M› ‹x ♯ yvec1› ‹x ≠ y'› ‹x ♯ yvec2› ‹x ♯ N›
        by(intro Scope) auto
       moreover have (Ψ, (|νx|)P', (|νx|)P') ∈ Rel by(rule R1)
       ultimately show ?case by blast
     qed
    next
     case(cBrOpen M yvec1 yvec2 y' N P')
    from ‹yvec1 ♯* yvec2› ‹distinct yvec1› ‹distinct yvec2› have distinct(yvec1@yvec2)
```

by *auto*
from ‹$x \sharp ¡M(\!|\nu*(yvec1 \mathbin{@} y' \# yvec2)|\!)\langle N\rangle$› **have** $x \sharp M$ **and** $x \sharp yvec1$ **and** $x \neq y'$ **and** $x \sharp yvec2$ **and** $x \sharp N$
by *simp+*
from ‹$y \sharp ¡M(\!|\nu*(yvec1 \mathbin{@} y' \# yvec2)|\!)\langle N\rangle$› **have** $y \sharp M$ **and** $y \sharp yvec1$ **and** $y \sharp yvec2$
by *simp+*
from ‹$\Psi \rhd ([(y, y')] \cdot (\!|\nu x|\!)P) \longmapsto ¡M(\!|\nu*(yvec1@yvec2)|\!)\langle N\rangle \prec P'$› ‹$x \neq y$› ‹$x \neq y'$›
have $\Psi \rhd (\!|\nu x|\!)([(y, y')] \cdot P) \longmapsto ¡M(\!|\nu*(yvec1@yvec2)|\!)\langle N\rangle \prec P'$ **by**(*simp add*: *eqvts*)
moreover note ‹$x \sharp \Psi$›
moreover from ‹$x \sharp N$› ‹$x \sharp yvec1$› ‹$x \sharp yvec2$› ‹$x \sharp M$› **have** $x \sharp ¡M(\!|\nu*(yvec1@yvec2)|\!)\langle N\rangle$ **by** *simp*
moreover note ‹$x \sharp P'$›
moreover from ‹$yvec1 \sharp* \Psi$› ‹$yvec2 \sharp* \Psi$› **have** $(yvec1@yvec2) \sharp* \Psi$ **by** *simp*
moreover from ‹$yvec1 \sharp* (\!|\nu x|\!)P$› ‹$yvec2 \sharp* (\!|\nu x|\!)P$› ‹$y \sharp yvec1$› ‹$y' \sharp yvec1$› ‹$y \sharp yvec2$› ‹$y' \sharp yvec2$› ‹$x \sharp yvec1$› ‹$x \sharp yvec2$›
have $(yvec1@yvec2) \sharp* ([(y, y')] \cdot P)$ **by** *simp*
moreover from ‹$yvec1 \sharp* M$› ‹$yvec2 \sharp* M$› **have** $(yvec1 \mathbin{@} yvec2) \sharp* M$
by *simp*
ultimately show *?case*
proof(*induct rule*: *resBrOutputCases'*)
case(*cBrOpen M' xvec1 xvec2 x' N' P'*)
from ‹$¡M'(\!|\nu*(xvec1 \mathbin{@} x' \# xvec2)|\!)\langle N'\rangle = ¡M(\!|\nu*(yvec1 \mathbin{@} yvec2)|\!)\langle N\rangle$› **have** $yvec1@yvec2 = xvec1@x'\#xvec2$ **and** $M = M'$ **and** $N = N'$ **by** (*simp add*: *action.inject*)+
from ‹$x \sharp yvec1$› ‹$x \sharp yvec2$› ‹$y' \sharp yvec1$› ‹$y' \sharp yvec2$› ‹$y \sharp yvec1$› ‹$y \sharp yvec2$›
have $x \sharp (yvec1@yvec2)$ **and** $y \sharp (yvec1@yvec2)$ **and** $y' \sharp (yvec1@yvec2)$ **by** *simp+*
with ‹$yvec1@yvec2 = xvec1@x'\#xvec2$›
have $x \sharp xvec1$ **and** $x \neq x'$ **and** $x \sharp xvec2$ **and** $y \sharp xvec1$ **and** $y \neq x'$ **and** $y \sharp xvec2$
and $y' \sharp xvec1$ **and** $x' \neq y'$ **and** $y' \sharp xvec2$
by *auto*

show *?case*
proof(*cases* $x' \in set\ yvec1$)
assume $x' \in set\ yvec1$

with ‹$yvec1@yvec2 = xvec1@x'\#xvec2$› ‹*distinct* $(yvec1@yvec2)$›
obtain *xvec2'* **where** *Eq1*: $yvec1 = xvec1@x'\#xvec2'$
and *Eq*: $xvec2 = xvec2'@yvec2$
by(*metis partitionListLeft*)
from ‹$\Psi \rhd ([(x, x')] \cdot [(y, y')] \cdot P) \longmapsto ¡M'(\!|\nu*(xvec1@xvec2)|\!)\langle N'\rangle \prec P'$› ‹$y' \in supp\ N$› ‹$y' \sharp \Psi$› ‹$y' \sharp M$› ‹$y' \sharp xvec1$› ‹$y' \sharp xvec2$› *Eq* ‹$M = M'$› ‹$N = N'$›
have $\Psi \rhd (\!|\nu y'|\!)([(x, x')] \cdot [(y, y')] \cdot P) \longmapsto ¡M'(\!|\nu*((xvec1@xvec2')@y'\#yvec2)|\!)\langle N'\rangle \prec P'$

by(*intro BrOpen*) *auto*
then have $\Psi \rhd (\!|\nu x'|\!)((\!|\nu y'|\!)([(x, x')] \cdot [(y, y')] \cdot P)) \longmapsto\!\!¡ M(\!|\nu*(xvec1@x'\#xvec2'@y'\#yvec2)|\!)\langle N\rangle \prec P'$
using ‹$x' \in supp\ N'$› ‹$x' \sharp \Psi$› ‹$x' \sharp M'$› ‹$x' \sharp xvec1$› ‹$x' \sharp xvec2$› ‹$x' \neq y'$› *Eq* ‹$M{=}M'$› ‹$N{=}N'$›
by(*intro BrOpen*) *auto*
with ‹$x' \neq y'$› ‹$x \neq y'$› ‹$x' \sharp [(y, y')] \cdot P$›
have $\Psi \rhd (\!|\nu x|\!)((\!|\nu y'|\!)([(y, y')] \cdot P)) \longmapsto\!\!¡ M(\!|\nu*(xvec1@x'\#xvec2'@y'\#yvec2)|\!)\langle N\rangle \prec P'$
by(*subst alphaRes*[**where** $y{=}x'$]) (*simp add*: *calc-atm eqvts abs-fresh*)+
with *Eq1* ‹$y' \sharp (\!|\nu x|\!)P$› ‹$x \neq y'$› *R1* **show** *?case*
by(*auto simp add*: *alphaRes abs-fresh*)
next
assume $\neg x' \in set\ yvec1$
then have $x' \sharp yvec1$ **by**(*simp add*: *fresh-def*)
from ‹$\neg x' \in set\ yvec1$› ‹$yvec1@yvec2 = xvec1@x'\#xvec2$›
have $x' \in set\ yvec2$
by(*auto simp add*: *append-eq-append-conv2 append-eq-Cons-conv*)
with ‹$yvec1@yvec2 = xvec1@x'\#xvec2$› ‹*distinct* $(yvec1@yvec2)$›
obtain *xvec2'* **where** *Eq*: $xvec1{=}yvec1@xvec2'$
and *Eq1*: $yvec2{=}xvec2'@x'\#xvec2$
by(*metis partitionListRight*)
from ‹$\Psi \rhd ([(x, x')] \cdot [(y, y')] \cdot P) \longmapsto\!\!¡ M'(\!|\nu*(xvec1@xvec2)|\!)\langle N'\rangle \prec P'$› ‹$y' \in supp\ N$› ‹$y' \sharp \Psi$› ‹$y' \sharp M$› ‹$y' \sharp xvec1$› ‹$y' \sharp xvec2$› *Eq* ‹$M{=}M'$› ‹$N = N'$›
have $\Psi \rhd (\!|\nu y'|\!)([(x, x')] \cdot [(y, y')] \cdot P) \longmapsto\!\!¡ M'(\!|\nu*(yvec1@y'\#xvec2'@xvec2)|\!)\langle N'\rangle \prec P'$
by(*intro BrOpen*) (*assumption* | *simp*)+
then have $\Psi \rhd (\!|\nu x'|\!)((\!|\nu y'|\!)([(x, x')] \cdot [(y, y')] \cdot P)) \longmapsto\!\!¡ M(\!|\nu*((yvec1@y'\#xvec2')@x'\#xvec2)|\!)\langle N\rangle \prec P'$
using ‹$x' \in supp\ N'$› ‹$x' \sharp \Psi$› ‹$x' \sharp M'$› ‹$x' \sharp xvec1$› ‹$x' \sharp xvec2$› ‹$x' \neq y'$› *Eq* ‹$M{=}M'$› ‹$N{=}N'$›
by(*intro BrOpen*) *auto*
with ‹$x' \neq y'$› ‹$x \neq y'$› ‹$x' \sharp [(y, y')] \cdot P$›
have $\Psi \rhd (\!|\nu x|\!)((\!|\nu y'|\!)([(y, y')] \cdot P)) \longmapsto\!\!¡ M(\!|\nu*((yvec1@y'\#xvec2')@x'\#xvec2)|\!)\langle N\rangle \prec P'$
by(*subst alphaRes*[**where** $y{=}x'$]) (*simp add*: *calc-atm eqvts abs-fresh*)+
with *Eq1* ‹$y' \sharp (\!|\nu x|\!)P$› ‹$x \neq y'$› *R1* **show** *?case*
by(*auto simp add*: *alphaRes abs-fresh*)
qed
next
case(*cRes P'*)
from ‹$\Psi \rhd ([(y, y')] \cdot P) \longmapsto\!\!¡ M(\!|\nu*(yvec1@yvec2)|\!)\langle N\rangle \prec P'$› ‹$y' \in supp\ N$› ‹$y' \sharp \Psi$› ‹$y' \sharp M$› ‹$y' \sharp yvec1$› ‹$y' \sharp yvec2$›
have $\Psi \rhd (\!|\nu y'|\!)([(y, y')] \cdot P) \longmapsto\!\!¡ M(\!|\nu*(yvec1@y'\#yvec2)|\!)\langle N\rangle \prec P'$ **by**(*rule BrOpen*)
with ‹$y' \sharp (\!|\nu x|\!)P$› ‹$x \neq y'$›**have** $\Psi \rhd (\!|\nu y|\!)P \longmapsto\!\!¡ M(\!|\nu*(yvec1@y'\#yvec2)|\!)\langle N\rangle \prec P'$ **by**(*simp add*: *alphaRes abs-fresh*)
then have $\Psi \rhd (\!|\nu x|\!)((\!|\nu y|\!)P) \longmapsto\!\!¡ M(\!|\nu*(yvec1@y'\#yvec2)|\!)\langle N\rangle \prec (\!|\nu x|\!)P'$
using ‹$x \sharp \Psi$› ‹$x \sharp M$› ‹$x \sharp yvec1$› ‹$x \neq y'$› ‹$x \sharp yvec2$› ‹$x \sharp N$›

by(*intro Scope*) *auto*
moreover have $(\Psi, (\!|\nu x|\!)P', (\!|\nu x|\!)P') \in Rel$ **by**(*rule R1*)
ultimately show *?case* **by** *blast*
qed
next
case(*cRes P'*)
from ‹$x \sharp (\!|\nu y|\!)P'$› ‹$x \neq y$› **have** $x \sharp P'$ **by**(*simp add*: *abs-fresh*)
with ‹$\Psi \rhd (\!|\nu x|\!)P \longmapsto \alpha \prec P'$› ‹$x \sharp \Psi$› ‹$x \sharp \alpha$›
show *?case* **using** ‹*bn* $\alpha \sharp* \Psi$› ‹*bn* $\alpha \sharp* P$› ‹*bn* $\alpha \sharp*$ *subject* α› ‹$y \sharp \alpha$›
proof(*induct rule*: *resCases'*[**where** $C=(x, y)$])
case(*cOpen M xvec1 xvec2 x' N P'*)
from ‹$y \sharp M(\!|\nu*(xvec1@x'\#xvec2)|\!)\langle N\rangle$› **have** $y \neq x'$ **and** $y \sharp M(\!|\nu*(xvec1@xvec2)|\!)\langle N\rangle$ **by** *simp+*
from ‹$\Psi \rhd ([(x, x')] \cdot P) \longmapsto M(\!|\nu*(xvec1@xvec2)|\!)\langle N\rangle \prec P'$› ‹$y \sharp \Psi$› ‹$y \sharp M(\!|\nu*(xvec1@xvec2)|\!)\langle N\rangle$›
have $\Psi \rhd (\!|\nu y|\!)([(x, x')] \cdot P) \longmapsto M(\!|\nu*(xvec1@xvec2)|\!)\langle N\rangle \prec (\!|\nu y|\!)P'$
by(*rule Scope*)
then have $\Psi \rhd (\!|\nu x'|\!)((\!|\nu y|\!)([(x, x')] \cdot P)) \longmapsto M(\!|\nu*(xvec1@x'\#xvec2)|\!)\langle N\rangle \prec (\!|\nu y|\!)P'$
using ‹$x' \in supp\ N$› ‹$x' \sharp \Psi$› ‹$x' \sharp M$› ‹$x' \sharp xvec1$› ‹$x' \sharp xvec2$›
by(*rule Open*)
with ‹$y \neq x'$› ‹$x \neq y$› ‹$x' \sharp P$› **have** $\Psi \rhd (\!|\nu x|\!)((\!|\nu y|\!)P) \longmapsto M(\!|\nu*(xvec1@x'\#xvec2)|\!)\langle N\rangle \prec (\!|\nu y|\!)P'$
by(*subst alphaRes*[**where** $y=x'$]) (*simp add*: *abs-fresh eqvts calc-atm*)+
moreover have $(\Psi, (\!|\nu y|\!)P', (\!|\nu y|\!)P') \in Rel$ **by**(*rule R1*)
ultimately show *?case* **by** *blast*
next
case(*cBrOpen M xvec1 xvec2 x' N P'*)
from ‹$y \sharp ¡M(\!|\nu*(xvec1@x'\#xvec2)|\!)\langle N\rangle$› **have** $y \neq x'$ **and** $y \sharp ¡M(\!|\nu*(xvec1@xvec2)|\!)\langle N\rangle$ **by** *simp+*
from ‹$\Psi \rhd ([(x, x')] \cdot P) \longmapsto ¡M(\!|\nu*(xvec1@xvec2)|\!)\langle N\rangle \prec P'$› ‹$y \sharp \Psi$› ‹$y \sharp ¡M(\!|\nu*(xvec1@xvec2)|\!)\langle N\rangle$›
have $\Psi \rhd (\!|\nu y|\!)([(x, x')] \cdot P) \longmapsto ¡M(\!|\nu*(xvec1@xvec2)|\!)\langle N\rangle \prec (\!|\nu y|\!)P'$
by(*rule Scope*)
then have $\Psi \rhd (\!|\nu x'|\!)((\!|\nu y|\!)([(x, x')] \cdot P)) \longmapsto ¡M(\!|\nu*(xvec1@x'\#xvec2)|\!)\langle N\rangle \prec (\!|\nu y|\!)P'$
using ‹$x' \in supp\ N$› ‹$x' \sharp \Psi$› ‹$x' \sharp M$› ‹$x' \sharp xvec1$› ‹$x' \sharp xvec2$›
by(*rule BrOpen*)
with ‹$y \neq x'$› ‹$x \neq y$› ‹$x' \sharp P$› **have** $\Psi \rhd (\!|\nu x|\!)((\!|\nu y|\!)P) \longmapsto ¡M(\!|\nu*(xvec1@x'\#xvec2)|\!)\langle N\rangle \prec (\!|\nu y|\!)P'$
by(*subst alphaRes*[**where** $y=x'$]) (*simp add*: *abs-fresh eqvts calc-atm*)+
moreover have $(\Psi, (\!|\nu y|\!)P', (\!|\nu y|\!)P') \in Rel$ **by**(*rule R1*)
ultimately show *?case* **by** *blast*
next
case(*cRes P'*)
from ‹$\Psi \rhd P \longmapsto \alpha \prec P'$› ‹$y \sharp \Psi$› ‹$y \sharp \alpha$›
have $\Psi \rhd (\!|\nu y|\!)P \longmapsto \alpha \prec (\!|\nu y|\!)P'$ **by**(*rule Scope*)
then have $\Psi \rhd (\!|\nu x|\!)((\!|\nu y|\!)P) \longmapsto \alpha \prec (\!|\nu x|\!)((\!|\nu y|\!)P')$ **using** ‹$x \sharp \Psi$› ‹$x \sharp \alpha$›
by(*rule Scope*)

```
      moreover from ‹x ♯ Ψ› ‹y ♯ Ψ› have (Ψ, (⦇νx⦈)((⦇νy⦈)P′), (⦇νy⦈)((⦇νx⦈)P′)) ∈
Rel
        by(rule R2)
       ultimately show ?case by blast
     next
       case(cBrClose M xvec N P′)
       then show ?case
       proof(cases y ♯ ¡M⦇ν∗xvec⦈⟨N⟩)
         case True
         with ‹Ψ ▷ P ⟼¡M⦇ν∗xvec⦈⟨N⟩ ≺ P′›
         have Ψ ▷ (⦇νy⦈)P ⟼¡M⦇ν∗xvec⦈⟨N⟩ ≺ (⦇νy⦈)P′ using ‹y ♯ Ψ›
           by(intro Scope)
         then have Ψ ▷ (⦇νx⦈)((⦇νy⦈)P) ⟼ τ ≺ ((⦇νx⦈)((⦇ν∗xvec⦈)⦇νy⦈)P′)) using ‹x
∈ supp M› ‹x ♯ Ψ›
           by(rule BrClose)
         moreover have (Ψ, ((⦇νx⦈)((⦇ν∗xvec⦈)⦇νy⦈)P′)), (⦇νy⦈)((⦇νx⦈)((⦇ν∗xvec⦈)P′))) ∈
Rel
         proof −
           have mset (x#xvec@[y]) = mset (y#x#xvec)
             by simp
           moreover have (x#xvec@[y]) ♯* Ψ using ‹x ♯ Ψ› ‹y ♯ Ψ› ‹xvec ♯* Ψ›
             by simp
           ultimately have (Ψ, (⦇ν∗(x#xvec@[y])⦈)P′, (⦇ν∗(y#x#xvec)⦈)P′) ∈ Rel
             by(metis R3)
           then show ?thesis
             by(auto simp add: resChainAppend)
         qed
         ultimately show ?thesis
           by blast
       next
         case False
         then have y ∈ supp(¡M⦇ν∗xvec⦈⟨N⟩) unfolding fresh-def by simp
         show ?thesis
         proof(cases y ∈ supp(M))
           case True
           with ‹Ψ ▷ P ⟼¡M⦇ν∗xvec⦈⟨N⟩ ≺ P′›
           have Ψ ▷ (⦇νy⦈)P ⟼τ ≺ (⦇νy⦈)((⦇ν∗xvec⦈)P′) using ‹y ♯ Ψ›
             by(rule BrClose)
           then have Ψ ▷ (⦇νx⦈)((⦇νy⦈)P) ⟼τ ≺ (⦇νx⦈)((⦇νy⦈)((⦇ν∗xvec⦈)P′)) using ‹x
♯ Ψ›
             by(rule Scope) simp
           moreover have (Ψ, (⦇νx⦈)((⦇νy⦈)((⦇ν∗xvec⦈)P′)), (⦇νy⦈)((⦇νx⦈)((⦇ν∗xvec⦈)P′)))
∈ Rel using ‹x ♯ Ψ› ‹y ♯ Ψ›
             by(metis R2)
           ultimately show ?thesis
             by blast
         next
           case False
           then have y ♯ M by(simp add: fresh-def)
```

from ‹*xvec* ♯* (*x*, *y*)› **have** *y* ♯ *xvec* **by** *simp*
with *False* ‹*y* ∈ *supp*(¡*M*⦇*ν*∗*xvec*⦈⟨*N*⟩)›
have *y* ∈ *supp N*
by(*simp add*: *fresh-def action.supp*)
from ‹Ψ ▷ *P* ⟼¡*M*⦇*ν*∗*xvec*⦈⟨*N*⟩ ≺ *P*′› **have** Ψ ▷ *P* ⟼¡*M*⦇*ν*∗([]@*xvec*)⦈⟨*N*⟩ ≺ *P*′
by *simp*
then have Ψ ▷ ⦇*ν y*⦈*P* ⟼¡*M*⦇*ν*∗([]@*y*#*xvec*)⦈⟨*N*⟩ ≺ *P*′ **using** ‹*y* ∈ *supp N*› ‹*y* ♯ Ψ› ‹*y* ♯ *M*› ‹*y* ♯ *xvec*›
by(*intro BrOpen*) (*assumption*|*simp*)+
then have Ψ ▷ ⦇*ν y*⦈*P* ⟼¡*M*⦇*ν*∗(*y*#*xvec*)⦈⟨*N*⟩ ≺ *P*′
by *simp*
then have Ψ ▷ ⦇*ν x*⦈(⦇*ν y*⦈*P*) ⟼*τ* ≺ ⦇*ν x*⦈(⦇*ν y*⦈(⦇*ν*∗*xvec*⦈*P*′)) **using** ‹*x* ∈ *supp M*› ‹*x* ♯ Ψ›
by(*auto dest*: *BrClose*)
moreover have (Ψ, ⦇*ν x*⦈(⦇*ν y*⦈(⦇*ν*∗*xvec*⦈*P*′)), ⦇*ν y*⦈(⦇*ν x*⦈(⦇*ν*∗*xvec*⦈*P*′))) ∈ *Rel* **using** ‹*x* ♯ Ψ› ‹*y* ♯ Ψ›
by(*rule R2*)
ultimately show *?thesis* **by** *blast*
qed
qed
qed
next
case(*cBrClose M xvec N P*′)
from ‹*xvec* ♯* *x*› **have** *x* ♯ *xvec* **by** *simp*
have *x* ♯ ⦇*ν x*⦈*P* **by**(*simp add*: *fresh-abs-fun-iff*[*OF pt-name-inst*, *OF at-name-inst*, *OF fin-supp*])
have *x* ♯ *P*′
by(*rule broutputFreshDerivativeP*) *fact*+
have *x* ♯ *N*
by(*rule broutputFreshDerivativeN*) *fact*+
moreover from ‹Ψ ▷ ⦇*ν x*⦈*P* ⟼ ¡*M*⦇*ν*∗*xvec*⦈⟨*N*⟩ ≺ *P*′› ‹*xvec* ♯* *M*› ‹*x* ♯ ⦇*ν x*⦈*P*› **have** *x* ♯ *M*
by(*rule brOutputFreshSubject*)
moreover note ‹*x* ♯ *xvec*›
ultimately have *x* ♯ ¡*M*⦇*ν*∗*xvec*⦈⟨*N*⟩
by *simp*
have *bn* (¡*M*⦇*ν*∗*xvec*⦈⟨*N*⟩) ♯* Ψ **using** ‹*xvec* ♯* Ψ› **by** *simp*
have *bn* (¡*M*⦇*ν*∗*xvec*⦈⟨*N*⟩) ♯* *P* **using** ‹*xvec* ♯* ⦇*ν x*⦈*P*› ‹*x* ♯ *xvec*› **by**(*simp add*: *fresh-abs-fun-iff*[*OF pt-name-inst*, *OF at-name-inst*, *OF fin-supp*])
have *bn* (¡*M*⦇*ν*∗*xvec*⦈⟨*N*⟩) ♯* *subject* (¡*M*⦇*ν*∗*xvec*⦈⟨*N*⟩) **using** ‹*xvec* ♯* *M*› **by** *simp*
have *y* ∈ *supp*(*subject* (¡*M*⦇*ν*∗*xvec*⦈⟨*N*⟩)) **using** ‹*y* ∈ *supp M*›
by(*simp add*: *supp-some*)
obtain *M*′ *xvec*′ *N*′ **where** ¡*M*⦇*ν*∗*xvec*⦈⟨*N*⟩ = ¡*M*′⦇*ν*∗*xvec*′⦈⟨*N*′⟩
by *auto*
from ‹Ψ ▷ ⦇*ν x*⦈*P* ⟼ ¡*M*⦇*ν*∗*xvec*⦈⟨*N*⟩ ≺ *P*′› ‹*x* ♯ Ψ› ‹*x* ♯ ¡*M*⦇*ν*∗*xvec*⦈⟨*N*⟩› ‹*x* ♯ *P*′› ‹*bn* (¡*M*⦇*ν*∗*xvec*⦈⟨*N*⟩) ♯* Ψ› ‹*bn* (¡*M*⦇*ν*∗*xvec*⦈⟨*N*⟩) ♯* *P*› ‹*bn* (¡*M*⦇*ν*∗*xvec*⦈⟨*N*⟩) ♯* *subject* (¡*M*⦇*ν*∗*xvec*⦈⟨*N*⟩)› ‹¡*M*⦇*ν*∗*xvec*⦈⟨*N*⟩ = ¡*M*′⦇*ν*∗*xvec*′⦈⟨*N*′⟩› ‹*y* ∈ *supp*(*subject*

$(¡M(\!|\nu{*}xvec|\!)\langle N\rangle))\rangle$
have $\exists\, Q'.\ \Psi \rhd (\!|\nu x|\!)((\!|\nu y|\!)P) \longmapsto \tau \prec Q' \wedge (\Psi,\ Q',\ (\!|\nu y|\!)((\!|\nu{*}(bn\ (¡M(\!|\nu{*}xvec|\!)\langle N\rangle)))\!|P'))$ *Rel*
proof(*induct rule*: *resCases'*[**where** $C{=}y$])
case *cOpen* **then show** *?case* **by**(*simp add*: *residualInject*)
next
case(*cBrOpen M xvec yvec z N P'*)
from $\langle y \in supp\ (subject\ (¡M(\!|\nu{*}(xvec\ @\ z\ \#\ yvec)|\!)\langle N\rangle)))\rangle$ **have** $y \in supp\ M$
by(*simp add*: *supp-some*)
then have $y \neq z$ **using** $\langle z \sharp M\rangle$ **by**(*auto simp add*: *fresh-def*)
from $\langle \Psi \rhd [(x,\ z)] \cdot P \longmapsto ¡M(\!|\nu{*}(xvec\ @\ yvec)|\!)\langle N\rangle \prec P'\rangle$ $\langle y \in supp\ M\rangle$ $\langle y \sharp \Psi\rangle$
have $\Psi \rhd (\!|\nu y|\!)([(x,\ z)] \cdot P) \longmapsto \tau \prec (\!|\nu y|\!)((\!|\nu{*}(xvec@yvec)|\!)P')$
by(*rule BrClose*)
then have $\Psi \rhd (\!|\nu z|\!)((\!|\nu y|\!)([(x,\ z)] \cdot P)) \longmapsto \tau \prec (\!|\nu z|\!)((\!|\nu y|\!)((\!|\nu{*}(xvec@yvec)|\!)P'))$ **using** $\langle z \sharp \Psi\rangle$
by(*rule Scope*) *simp*
then have $\Psi \rhd (\!|\nu x|\!)((\!|\nu y|\!)P) \longmapsto \tau \prec (\!|\nu z|\!)((\!|\nu y|\!)((\!|\nu{*}(xvec@yvec)|\!)P'))$ **using** $\langle z \sharp P\rangle$ $\langle x \neq y\rangle$ $\langle y \neq z\rangle$
apply(*subst alphaRes*[**where** $x{=}x$ **and** $y{=}z$])
apply(*simp add*: *fresh-abs-fun-iff*[*OF pt-name-inst*, *OF at-name-inst*, *OF fin-supp*])
apply(*simp add*: *eqvts swap-simps*)
done
moreover have $(\Psi,\ (\!|\nu z|\!)((\!|\nu y|\!)((\!|\nu{*}(xvec\ @\ yvec)|\!)P')),\ (\!|\nu y|\!)((\!|\nu{*}bn\ (¡M(\!|\nu{*}(xvec\ @\ z\ \#\ yvec)|\!)\langle N\rangle)|\!)P'))$ *Rel*
proof −
have $mset(z\#y\#xvec@yvec) = mset(y\#xvec@z\#yvec)$
by *simp*
moreover have $(z\#y\#xvec@yvec)\ \sharp{*}\ \Psi$ **using** $\langle z \sharp \Psi\rangle$ $\langle y \sharp \Psi\rangle$ $\langle xvec\ \sharp{*}\ \Psi\rangle$ $\langle yvec\ \sharp{*}\ \Psi\rangle$
by *simp*
ultimately have $(\Psi,\ (\!|\nu{*}(z\#y\#xvec@yvec)|\!)P',\ (\!|\nu{*}(y\#xvec@z\#yvec)|\!)P')$ *Rel*
by(*metis R3*)
then show *?thesis*
by(*simp add*: *resChainAppend*)
qed
ultimately show *?case*
by *blast*
next
case(*cRes P'*)
from $\langle \Psi \rhd P \longmapsto ¡M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P'\rangle$ $\langle y \in supp\ M\rangle$ $\langle y \sharp \Psi\rangle$
have $\Psi \rhd (\!|\nu y|\!)P \longmapsto \tau \prec (\!|\nu y|\!)((\!|\nu{*}xvec|\!)P')$
by(*rule BrClose*)
then have $\Psi \rhd (\!|\nu x|\!)((\!|\nu y|\!)P) \longmapsto \tau \prec (\!|\nu x|\!)((\!|\nu y|\!)((\!|\nu{*}xvec|\!)P'))$ **using** $\langle x \sharp \Psi\rangle$
by(*rule Scope*) *simp*
moreover have $(\Psi,\ (\!|\nu x|\!)((\!|\nu y|\!)((\!|\nu{*}xvec|\!)P')),\ (\!|\nu y|\!)((\!|\nu{*}bn\ (¡M(\!|\nu{*}xvec|\!)\langle N\rangle)|\!)(\!|\nu x|\!)P'))$

$\in$ *Rel*
proof −
have *mset*(*x*#*y*#*xvec*) = *mset*(*y*#*xvec*@[*x*])
by *simp*
moreover have (*x*#*y*#*xvec*) ♯* Ψ **using** ‹*x* ♯ Ψ› ‹*y* ♯ Ψ› ‹*xvec* ♯* Ψ›
by *simp*
ultimately have (Ψ, ⦇*ν**(*x*#*y*#*xvec*)⦈*P*′, ⦇*ν**(*y*#*xvec*@[*x*])⦈*P*′) ∈ *Rel*
by(*metis R3*)
then show *?thesis* **by**(*simp add*: *resChainAppend*)
qed
ultimately show *?case*
by *blast*
next
case *cBrClose* **then show** *?case* **by** *simp*
qed
then show *?case* **by** *simp*
qed
qed
qed

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* × (*'a*, *'b*, *'c*) *psi* × (*'a*, *'b*, *'c*) *psi*) *set*

assumes *eqvt Rel*
and *C1*: ⋀Ψ′ *S T U*. (Ψ, (*S* ∥ *T*) ∥ *U*, *S* ∥ (*T* ∥ *U*)) ∈ *Rel*
and *C2*: ⋀*xvec* Ψ′ *S T U*. ⟦*xvec* ♯* Ψ′; *xvec* ♯* *S*⟧ ⟹ (Ψ′, ⦇*ν**xvec⦈((*S* ∥ *T*) ∥ *U*), *S* ∥ (⦇*ν**xvec⦈(*T* ∥ *U*))) ∈ *Rel*
and *C3*: ⋀*xvec* Ψ′ *S T U*. ⟦*xvec* ♯* Ψ′; *xvec* ♯* *U*⟧ ⟹ (Ψ′, (⦇*ν**xvec⦈(*S* ∥ *T*)) ∥ *U*, ⦇*ν**xvec⦈(*S* ∥ (*T* ∥ *U*))) ∈ *Rel*
and *C4*: ⋀Ψ′ *S T xvec*. ⟦(Ψ′, *S*, *T*) ∈ *Rel*; *xvec* ♯* Ψ′⟧ ⟹ (Ψ′, ⦇*ν**xvec⦈*S*, ⦇*ν**xvec⦈*T*) ∈ *Rel*

shows Ψ ▷ (*P* ∥ *Q*) ∥ *R* ⇝[*Rel*] *P* ∥ (*Q* ∥ *R*)
using ‹*eqvt Rel*›
proof(*induct rule*: *simI*[*of* - - - - ()])
case(*cSim* α *PQR*)
from ‹*bn* α ♯* (*P* ∥ *Q* ∥ *R*)› **have** *bn* α ♯* *P* **and** *bn* α ♯* *Q* **and** *bn* α ♯* *R* **by** *simp*+
then have *bn* α ♯* (*Q* ∥ *R*) **by** *simp*
with ‹Ψ ▷ *P* ∥ (*Q* ∥ *R*) ⟼α ≺ *PQR*› ‹*bn* α ♯* Ψ› ‹*bn* α ♯* *P*›
show *?case* **using** ‹*bn* α ♯* *subject* α›
proof(*induct rule*: *parCases*[**where** *C* = (Ψ, *P*, *Q*, *R*, α)])
case(*cPar1 P*′ A_{QR} Ψ_{QR})
from ‹A_{QR} ♯* (Ψ, *P*, *Q*, *R*, α)› **have** A_{QR} ♯* *Q* **and** A_{QR} ♯* *R*
by *simp*+

with ‹*extractFrame*$(Q \parallel R) = \langle A_{QR}, \Psi_{QR}\rangle$› ‹*distinct* A_{QR}›
obtain A_Q Ψ_Q A_R Ψ_R **where** $A_{QR} = A_Q @ A_R$ **and** $\Psi_{QR} = \Psi_Q \otimes \Psi_R$ **and** *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$ **and** *FrR*: *extractFrame* $R = \langle A_R, \Psi_R\rangle$
and $A_Q \sharp* \Psi_R$ **and** $A_R \sharp* \Psi_Q$
by(*auto intro*: *mergeFrameE dest*: *extractFrameFreshChain*)

from ‹$A_{QR} = A_Q @ A_R$› ‹$A_{QR} \sharp* \Psi$› ‹$A_{QR} \sharp* P$› ‹$A_{QR} \sharp* Q$› ‹$A_{QR} \sharp* \alpha$›
have $A_Q \sharp* \Psi$ **and** $A_R \sharp* \Psi$ **and** $A_Q \sharp* P$ **and** $A_R \sharp* P$ **and** $A_Q \sharp* Q$ **and** $A_R \sharp* Q$ **and** $A_Q \sharp* \alpha$ **and** $A_R \sharp* \alpha$
by *simp*+

from ‹$\Psi \otimes \Psi_{QR} \rhd P \longmapsto \alpha \prec P'$› ‹$\Psi_{QR} = \Psi_Q \otimes \Psi_R$› **have** $(\Psi \otimes \Psi_R) \otimes \Psi_Q \rhd P \longmapsto \alpha \prec P'$
by(*metis statEqTransition Associativity Commutativity Composition*)
then have $\Psi \otimes \Psi_R \rhd P \parallel Q \longmapsto \alpha \prec (P' \parallel Q)$ **using** *FrQ* ‹*bn* $\alpha \sharp* Q$› ‹$A_Q \sharp* \Psi$› ‹$A_Q \sharp* \Psi_R$› ‹$A_Q \sharp* P$› ‹$A_Q \sharp* \alpha$›
by(*intro Par1*) *auto*
then have $\Psi \rhd (P \parallel Q) \parallel R \longmapsto \alpha \prec ((P' \parallel Q) \parallel R)$ **using** *FrR* ‹*bn* $\alpha \sharp* R$› ‹$A_R \sharp* \Psi$› ‹$A_R \sharp* P$› ‹$A_R \sharp* Q$› ‹$A_R \sharp* \alpha$›
by(*auto intro*: *Par1*)
moreover have $(\Psi, (P' \parallel Q) \parallel R, P' \parallel (Q \parallel R)) \in Rel$ **by**(*rule C1*)
ultimately show *?case* **by** *blast*
next
case(*cPar2 QR* A_P Ψ_P)
from ‹$A_P \sharp* (\Psi, P, Q, R, \alpha)$› **have** $A_P \sharp* Q$ **and** $A_P \sharp* R$ **and** $A_P \sharp* \alpha$
by *simp*+
have *FrP*: *extractFrame* $P = \langle A_P, \Psi_P\rangle$ **by** *fact*
with ‹*bn* $\alpha \sharp* P$› ‹$A_P \sharp* \alpha$› **have** *bn* $\alpha \sharp* \Psi_P$ **by**(*auto dest*: *extractFrameFreshChain*)
with ‹*bn* $\alpha \sharp* \Psi$› **have** *bn* $\alpha \sharp* (\Psi \otimes \Psi_P)$ **by** *force*
with ‹$\Psi \otimes \Psi_P \rhd Q \parallel R \longmapsto \alpha \prec QR$›
show *?case* **using** ‹*bn* $\alpha \sharp* Q$› ‹*bn* $\alpha \sharp* R$› ‹*bn* $\alpha \sharp*$ *subject* α› ‹$A_P \sharp* Q$› ‹$A_P \sharp* R$›
proof(*induct rule*: *parCasesSubject*[**where** $C = (A_P, \Psi_P, P, Q, R, \Psi)$])
case(*cPar1* Q' A_R Ψ_R)
from ‹$A_R \sharp* (A_P, \Psi_P, P, Q, R, \Psi)$› **have** $A_R \sharp* A_P$ **and** $A_R \sharp* P$ **and** $A_R \sharp* Q$ **and** $A_R \sharp* \Psi_P$ **and** $A_R \sharp* \Psi$
by *simp*+
from ‹$A_P \sharp* R$› ‹*extractFrame* $R = \langle A_R, \Psi_R\rangle$› ‹$A_R \sharp* A_P$› **have** $A_P \sharp* \Psi_R$
by(*auto dest*: *extractFrameFreshChain*)
from ‹$(\Psi \otimes \Psi_P) \otimes \Psi_R \rhd Q \longmapsto \alpha \prec Q'$› **have** $(\Psi \otimes \Psi_R) \otimes \Psi_P \rhd Q \longmapsto \alpha \prec Q'$
by(*metis statEqTransition Associativity Commutativity Composition*)
then have $\Psi \otimes \Psi_R \rhd P \parallel Q \longmapsto \alpha \prec (P \parallel Q')$
using *FrP* ‹*bn* $\alpha \sharp* P$› ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* \Psi_R$› ‹$A_P \sharp* Q$› ‹$A_P \sharp* \alpha$›
by(*intro Par2*) (*assumption* | *force*)+
then have $\Psi \rhd (P \parallel Q) \parallel R \longmapsto \alpha \prec ((P \parallel Q') \parallel R)$
using ‹*extractFrame* $R = \langle A_R, \Psi_R\rangle$› ‹*bn* $\alpha \sharp* R$› ‹$A_R \sharp* \Psi$› ‹$A_R \sharp* P$› ‹$A_R \sharp* Q$› ‹$A_R \sharp* \alpha$›

by(*intro Par1*) (*assumption* | *simp*)+
moreover have $(\Psi, (P \parallel Q') \parallel R, P \parallel (Q' \parallel R)) \in Rel$ **by**(*rule C1*)
ultimately show *?case* **by** *blast*
next
case(*cPar2* R' A_Q Ψ_Q)
from ‹$A_Q \sharp* (A_P, \Psi_P, P, Q, R, \Psi)$› **have** $A_Q \sharp* A_P$ **and** $A_Q \sharp* R$ **and** $A_Q \sharp* \Psi_P$ **and** $A_Q \sharp* \Psi$
by *simp+*
have *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$ **by** *fact*
from ‹$A_P \sharp* Q$› *FrQ* ‹$A_Q \sharp* A_P$› **have** $A_P \sharp* \Psi_Q$
by(*auto dest*: *extractFrameFreshChain*)
from ‹$(\Psi \otimes \Psi_P) \otimes \Psi_Q \rhd R \longmapsto \alpha \prec R'$›
have $\Psi \otimes (\Psi_P \otimes \Psi_Q) \rhd R \longmapsto \alpha \prec R'$
by(*blast intro*: *statEqTransition Associativity*)
moreover from *FrP FrQ* ‹$A_Q \sharp* A_P$› ‹$A_P \sharp* \Psi_Q$› ‹$A_Q \sharp* \Psi_P$›
have *extractFrame*$(P \parallel Q) = \langle (A_P @ A_Q), \Psi_P \otimes \Psi_Q \rangle$ **by** *simp*
moreover from ‹$bn\ \alpha \sharp* P$› ‹$bn\ \alpha \sharp* Q$› **have** $bn\ \alpha \sharp* (P \parallel Q)$ **by** *simp*
moreover from ‹$A_P \sharp* \Psi$› ‹$A_Q \sharp* \Psi$› **have** $(A_P @ A_Q) \sharp* \Psi$ **by** *simp*
moreover from ‹$A_P \sharp* R$› ‹$A_Q \sharp* R$› **have** $(A_P @ A_Q) \sharp* R$ **by** *simp*
moreover from ‹$A_P \sharp* \alpha$› ‹$A_Q \sharp* \alpha$› **have** $(A_P @ A_Q) \sharp* \alpha$ **by** *simp*
ultimately have $\Psi \rhd (P \parallel Q) \parallel R \longmapsto \alpha \prec ((P \parallel Q) \parallel R')$
by(*rule Par2*)
moreover have $(\Psi, (P \parallel Q) \parallel R', P \parallel (Q \parallel R')) \in Rel$ **by**(*rule C1*)
ultimately show *?case* **by** *blast*
next
case(*cComm1* Ψ_R M N Q' A_Q Ψ_Q K *xvec* R' A_R)
from ‹$A_Q \sharp* (A_P, \Psi_P, P, Q, R, \Psi)$›
have $A_Q \sharp* P$ **and** $A_Q \sharp* Q$ **and** $A_Q \sharp* R$ **and** $A_Q \sharp* A_P$ **and** $A_Q \sharp* \Psi_P$ **and** $A_Q \sharp* \Psi$ **by** *simp+*
from ‹$A_R \sharp* (A_P, \Psi_P, P, Q, R, \Psi)$› **have** $A_R \sharp* P$ **and** $A_R \sharp* Q$ **and** $A_R \sharp* R$ **and** $A_R \sharp* A_P$ **and** $A_R \sharp* \Psi$ **by** *simp+*
from ‹*xvec* $\sharp* (A_P, \Psi_P, P, Q, R, \Psi)$› **have** *xvec* $\sharp* A_P$ **and** *xvec* $\sharp* P$ **and** *xvec* $\sharp* Q$ **and** *xvec* $\sharp* \Psi$ **by** *simp+*

have *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$ **by** *fact*
with ‹$A_P \sharp* Q$› ‹$A_Q \sharp* A_P$› **have** $A_P \sharp* \Psi_Q$
by(*auto dest*: *extractFrameFreshChain*)
have *FrR*: *extractFrame* $R = \langle A_R, \Psi_R \rangle$ **by** *fact*
with ‹$A_P \sharp* R$› ‹$A_R \sharp* A_P$› **have** $A_P \sharp* \Psi_R$
by(*auto dest*: *extractFrameFreshChain*)
from ‹$(\Psi \otimes \Psi_P) \otimes \Psi_Q \rhd R \longmapsto K(\!|\nu * xvec|\!)\langle N \rangle \prec R'$› ‹$A_P \sharp* R$› ‹*xvec* $\sharp* A_P$› ‹*xvec* $\sharp* K$› ‹*distinct xvec*› **have** $A_P \sharp* N$
by(*auto intro*: *outputFreshChainDerivative*)

from ‹$(\Psi \otimes \Psi_P) \otimes \Psi_R \rhd Q \longmapsto M(\!|N|\!) \prec Q'$› **have** $(\Psi \otimes \Psi_R) \otimes \Psi_P \rhd Q \longmapsto M(\!|N|\!) \prec Q'$
by(*metis statEqTransition Associativity Commutativity Composition*)
then have $\Psi \otimes \Psi_R \rhd P \parallel Q \longmapsto M(\!|N|\!) \prec (P \parallel Q')$ **using** *FrP* ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* \Psi_R$› ‹$A_P \sharp* Q$› ‹$A_P \sharp* M$› ‹$A_P \sharp* N$›

by(*intro Par2*) *auto*
moreover from *FrP FrQ* ‹$A_P \sharp* \Psi_Q$› ‹$A_Q \sharp* A_P$› ‹$A_Q \sharp* \Psi_P$› **have** *extractFrame*$(P \parallel Q) = \langle (A_P @ A_Q), \Psi_P \otimes \Psi_Q \rangle$
by *simp*
moreover from ‹$(\Psi \otimes \Psi_P) \otimes \Psi_Q \rhd R \longmapsto K(\!|\nu * xvec|\!)\langle N \rangle \prec R'$› **have** $\Psi \otimes \Psi_P \otimes \Psi_Q \rhd R \longmapsto K(\!|\nu * xvec|\!)\langle N \rangle \prec R'$
by(*metis statEqTransition Associativity*)
moreover note ‹*extractFrame* $R = \langle A_R, \Psi_R \rangle$›
moreover from ‹$(\Psi \otimes \Psi_P) \otimes \Psi_Q \otimes \Psi_R \vdash M \leftrightarrow K$› **have** $\Psi \otimes (\Psi_P \otimes \Psi_Q) \otimes \Psi_R \vdash M \leftrightarrow K$
by(*metis statEqEnt Associativity Commutativity Composition*)
ultimately have $\Psi \rhd (P \parallel Q) \parallel R \longmapsto \tau \prec (\!|\nu * xvec|\!)((P \parallel Q') \parallel R')$
using ‹$A_P \sharp* \Psi$› ‹$A_Q \sharp* \Psi$› ‹$A_R \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_Q \sharp* P$› ‹$A_R \sharp* P$› ‹$A_P \sharp* Q$› ‹$A_Q \sharp* Q$› ‹$A_R \sharp* Q$› ‹$A_P \sharp* R$› ‹$A_Q \sharp* R$› ‹$A_R \sharp* R$›
‹$A_P \sharp* M$› ‹$A_Q \sharp* M$› ‹$A_R \sharp* K$› ‹$A_R \sharp* A_P$› ‹$A_Q \sharp* A_R$› ‹*xvec* $\sharp* P$› ‹*xvec* $\sharp* Q$›
by(*intro Comm1*) (*assumption* | *simp*)+
moreover from ‹*xvec* $\sharp* \Psi$› ‹*xvec* $\sharp* P$› **have** $(\Psi, (\!|\nu * xvec|\!)((P \parallel Q') \parallel R'), P \parallel ((\!|\nu * xvec|\!)(Q' \parallel R'))) \in Rel$
by(*rule C2*)
ultimately show *?case* **by** *blast*
next
case(*cComm2* Ψ_R M *xvec* N Q' A_Q Ψ_Q K R' A_R)
from ‹$A_Q \sharp* (A_P, \Psi_P, P, Q, R, \Psi)$›
have $A_Q \sharp* P$ **and** $A_Q \sharp* Q$ **and** $A_Q \sharp* R$ **and** $A_Q \sharp* A_P$ **and** $A_Q \sharp* \Psi$ **and** $A_Q \sharp* \Psi_P$ **by** *simp*+
from ‹$A_R \sharp* (A_P, \Psi_P, P, Q, R, \Psi)$› **have** $A_R \sharp* P$ **and** $A_R \sharp* Q$ **and** $A_R \sharp* R$ **and** $A_R \sharp* A_P$**and** $A_R \sharp* \Psi$ **by** *simp*+
from ‹*xvec* $\sharp* (A_P, \Psi_P, P, Q, R, \Psi)$› **have** *xvec* $\sharp* A_P$ **and** *xvec* $\sharp* P$ **and** *xvec* $\sharp* Q$ **and** *xvec* $\sharp* \Psi$ **by** *simp*+

have *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$ **by** *fact*
with ‹$A_P \sharp* Q$› ‹$A_Q \sharp* A_P$› **have** $A_P \sharp* \Psi_Q$
by(*auto dest*: *extractFrameFreshChain*)
have *FrR*: *extractFrame* $R = \langle A_R, \Psi_R \rangle$ **by** *fact*
with ‹$A_P \sharp* R$› ‹$A_R \sharp* A_P$› **have** $A_P \sharp* \Psi_R$
by(*auto dest*: *extractFrameFreshChain*)

from ‹$(\Psi \otimes \Psi_P) \otimes \Psi_R \rhd Q \longmapsto M(\!|\nu * xvec|\!)\langle N \rangle \prec Q'$› ‹$A_P \sharp* Q$› ‹*xvec* $\sharp* A_P$› ‹*xvec* $\sharp* M$› ‹*distinct xvec*› **have** $A_P \sharp* N$
by(*auto intro*: *outputFreshChainDerivative*)

from ‹$(\Psi \otimes \Psi_P) \otimes \Psi_R \rhd Q \longmapsto M(\!|\nu * xvec|\!)\langle N \rangle \prec Q'$› **have** $(\Psi \otimes \Psi_R) \otimes \Psi_P \rhd Q \longmapsto M(\!|\nu * xvec|\!)\langle N \rangle \prec Q'$
by(*metis statEqTransition Associativity Commutativity Composition*)
then have $\Psi \otimes \Psi_R \rhd P \parallel Q \longmapsto M(\!|\nu * xvec|\!)\langle N \rangle \prec (P \parallel Q')$ **using** *FrP* ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* \Psi_R$› ‹$A_P \sharp* Q$› ‹$A_P \sharp* M$› ‹$A_P \sharp* N$› ‹*xvec* $\sharp* P$› ‹*xvec* $\sharp* A_P$›
by(*intro Par2*) *auto*
moreover from *FrP FrQ* ‹$A_P \sharp* \Psi_Q$› ‹$A_Q \sharp* A_P$› ‹$A_Q \sharp* \Psi_P$› **have**

extractFrame$(P \parallel Q) = \langle (A_P @ A_Q), \Psi_P \otimes \Psi_Q \rangle$
by *simp+*
moreover from $\langle (\Psi \otimes \Psi_P) \otimes \Psi_Q \rhd R \longmapsto K(\!|N|\!) \prec R' \rangle$ **have** $\Psi \otimes \Psi_P \otimes \Psi_Q \rhd R \longmapsto K(\!|N|\!) \prec R'$
by(*metis statEqTransition Associativity*)
moreover note $\langle$*extractFrame* $R = \langle A_R, \Psi_R \rangle \rangle$
moreover from $\langle (\Psi \otimes \Psi_P) \otimes \Psi_Q \otimes \Psi_R \vdash M \leftrightarrow K \rangle$ **have** $\Psi \otimes (\Psi_P \otimes \Psi_Q) \otimes \Psi_R \vdash M \leftrightarrow K$
by(*metis statEqEnt Associativity Commutativity Composition*)
ultimately have $\Psi \rhd (P \parallel Q) \parallel R \longmapsto \tau \prec (\!|\nu * xvec|\!)((P \parallel Q') \parallel R')$
using $\langle A_P \sharp* \Psi \rangle$ $\langle A_Q \sharp* \Psi \rangle$ $\langle A_R \sharp* \Psi \rangle$ $\langle A_P \sharp* P \rangle$ $\langle A_Q \sharp* P \rangle$ $\langle A_R \sharp* P \rangle$ $\langle A_P \sharp* Q \rangle$ $\langle A_Q \sharp* Q \rangle$ $\langle A_R \sharp* Q \rangle$ $\langle A_P \sharp* R \rangle$ $\langle A_Q \sharp* R \rangle$ $\langle A_R \sharp* R \rangle$
$\langle A_P \sharp* M \rangle$ $\langle A_Q \sharp* M \rangle$ $\langle A_R \sharp* K \rangle$ $\langle A_R \sharp* A_P \rangle$ $\langle A_Q \sharp* A_R \rangle$ $\langle xvec \sharp* R \rangle$
by(*intro Comm2*) (*assumption* | *simp*)+
moreover from $\langle xvec \sharp* \Psi \rangle$ $\langle xvec \sharp* P \rangle$ **have** $(\Psi, (\!|\nu * xvec|\!)((P \parallel Q') \parallel R'), P \parallel ((\!|\nu * xvec|\!)(Q' \parallel R'))) \in Rel$
by(*rule C2*)
ultimately show *?case* **by** *blast*
next
case(*cBrMerge* Ψ_R M N Q' A_Q Ψ_Q R' A_R)
from $\langle A_P \sharp* \alpha \rangle$ $\langle \alpha = ¿M(\!|N|\!) \rangle$ **have** $A_P \sharp* M$ **and** $A_P \sharp* N$ **by** *simp+*
from $\langle A_Q \sharp* (A_P, \Psi_P, P, Q, R, \Psi) \rangle$
have $A_Q \sharp* P$ **and** $A_Q \sharp* Q$ **and** $A_Q \sharp* R$ **and** $A_Q \sharp* A_P$ **and** $A_Q \sharp* \Psi_P$ **and** $A_Q \sharp* \Psi$ **by** *simp+*
from $\langle A_R \sharp* (A_P, \Psi_P, P, Q, R, \Psi) \rangle$ **have** $A_R \sharp* P$ **and** $A_R \sharp* Q$ **and** $A_R \sharp* R$ **and** $A_R \sharp* A_P$ **and** $A_R \sharp* \Psi$ **by** *simp+*

have *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$ **by** *fact*
with $\langle A_P \sharp* Q \rangle$ $\langle A_Q \sharp* A_P \rangle$ **have** $A_P \sharp* \Psi_Q$
by(*auto dest*: *extractFrameFreshChain*)
have *FrR*: *extractFrame* $R = \langle A_R, \Psi_R \rangle$ **by** *fact*
with $\langle A_P \sharp* R \rangle$ $\langle A_R \sharp* A_P \rangle$ **have** $A_P \sharp* \Psi_R$
by(*auto dest*: *extractFrameFreshChain*)

from $\langle (\Psi \otimes \Psi_P) \otimes \Psi_R \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q' \rangle$ **have** $(\Psi \otimes \Psi_R) \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$
by(*metis statEqTransition Associativity Commutativity Composition*)
then have $\Psi \otimes \Psi_R \rhd P \parallel Q \longmapsto ¿M(\!|N|\!) \prec (P \parallel Q')$ **using** *FrP* $\langle A_P \sharp* \Psi \rangle$ $\langle A_P \sharp* \Psi_R \rangle$ $\langle A_P \sharp* Q \rangle$ $\langle A_P \sharp* M \rangle$ $\langle A_P \sharp* N \rangle$
by(*intro Par2*) *auto*
moreover from *FrP FrQ* $\langle A_P \sharp* \Psi_Q \rangle$ $\langle A_Q \sharp* A_P \rangle$ $\langle A_Q \sharp* \Psi_P \rangle$ **have** *extractFrame*$(P \parallel Q) = \langle (A_P @ A_Q), \Psi_P \otimes \Psi_Q \rangle$
by *simp*
moreover from $\langle (\Psi \otimes \Psi_P) \otimes \Psi_Q \rhd R \longmapsto ¿M(\!|N|\!) \prec R' \rangle$ **have** $\Psi \otimes \Psi_P \otimes \Psi_Q \rhd R \longmapsto ¿M(\!|N|\!) \prec R'$
by(*metis statEqTransition Associativity*)
moreover note $\langle$*extractFrame* $R = \langle A_R, \Psi_R \rangle \rangle$
ultimately have $\Psi \rhd (P \parallel Q) \parallel R \longmapsto ¿M(\!|N|\!) \prec (P \parallel Q') \parallel R'$
using $\langle A_P \sharp* \Psi \rangle$ $\langle A_Q \sharp* \Psi \rangle$ $\langle A_R \sharp* \Psi \rangle$ $\langle A_P \sharp* P \rangle$ $\langle A_Q \sharp* P \rangle$ $\langle A_R \sharp* P \rangle$

$\langle A_P \sharp* Q\rangle$ $\langle A_Q \sharp* Q\rangle$ $\langle A_R \sharp* Q\rangle$ $\langle A_P \sharp* R\rangle$ $\langle A_Q \sharp* R\rangle$ $\langle A_R \sharp* R\rangle$
$\langle A_P \sharp* M\rangle$ $\langle A_Q \sharp* M\rangle$ $\langle A_R \sharp* M\rangle$ $\langle A_R \sharp* A_P\rangle$ $\langle A_Q \sharp* A_R\rangle$
by(*auto intro*: *BrMerge*)
moreover have $(\Psi, (P \parallel Q') \parallel R', P \parallel (Q' \parallel R')) \in Rel$
by(*rule C1*)
ultimately show *?case* **by** *blast*
next
case(*cBrComm1* Ψ_R M N Q' A_Q Ψ_Q *xvec* R' A_R)
from $\langle ¡M(\!|\nu*xvec|\!)\langle N\rangle = \alpha\rangle$ **have** *xvec* $= bn\ \alpha$
by(*auto simp add*: *action.inject*)
from $\langle ¡M(\!|\nu*xvec|\!)\langle N\rangle = \alpha\rangle$ $\langle A_P \sharp* \alpha\rangle$
have $A_P \sharp* xvec$ **and** $A_P \sharp* M$ **and** $A_P \sharp* N$ **by** *auto*
from $\langle xvec = bn\ \alpha\rangle$ $\langle bn\ \alpha \sharp* P\rangle$ $\langle bn\ \alpha \sharp* \Psi\rangle$ $\langle bn\ \alpha \sharp* \Psi_P\rangle$ **have** $xvec \sharp* P$ **and** $xvec \sharp* \Psi$ **and** $xvec \sharp* \Psi_P$
by *simp+*
from $\langle A_Q \sharp* (A_P, \Psi_P, P, Q, R, \Psi)\rangle$
have $A_Q \sharp* P$ **and** $A_Q \sharp* Q$ **and** $A_Q \sharp* R$ **and** $A_Q \sharp* A_P$ **and** $A_Q \sharp* \Psi_P$ **and** $A_Q \sharp* \Psi$ **by** *simp+*
from $\langle A_R \sharp* (A_P, \Psi_P, P, Q, R, \Psi)\rangle$ **have** $A_R \sharp* P$ **and** $A_R \sharp* Q$ **and** $A_R \sharp* R$ **and** $A_R \sharp* A_P$ **and** $A_R \sharp* \Psi$ **by** *simp+*

have *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$ **by** *fact*
with $\langle A_P \sharp* Q\rangle$ $\langle A_Q \sharp* A_P\rangle$ **have** $A_P \sharp* \Psi_Q$
by(*auto dest*: *extractFrameFreshChain*)
have *FrR*: *extractFrame* $R = \langle A_R, \Psi_R\rangle$ **by** *fact*
with $\langle A_P \sharp* R\rangle$ $\langle A_R \sharp* A_P\rangle$ **have** $A_P \sharp* \Psi_R$
by(*auto dest*: *extractFrameFreshChain*)

from $\langle(\Psi \otimes \Psi_P) \otimes \Psi_R \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'\rangle$ **have** $(\Psi \otimes \Psi_R) \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$
by(*metis statEqTransition Associativity Commutativity Composition*)
then have $\Psi \otimes \Psi_R \rhd P \parallel Q \longmapsto ¿M(\!|N|\!) \prec (P \parallel Q')$ **using** *FrP* $\langle A_P \sharp* \Psi\rangle$ $\langle A_P \sharp* \Psi_R\rangle$ $\langle A_P \sharp* Q\rangle$ $\langle A_P \sharp* M\rangle$ $\langle A_P \sharp* N\rangle$
by(*intro Par2*) *auto*
moreover from *FrP FrQ* $\langle A_P \sharp* \Psi_Q\rangle$ $\langle A_Q \sharp* A_P\rangle$ $\langle A_Q \sharp* \Psi_P\rangle$ **have** *extractFrame*$(P \parallel Q) = \langle (A_P @ A_Q), \Psi_P \otimes \Psi_Q\rangle$
by *simp*
moreover from $\langle(\Psi \otimes \Psi_P) \otimes \Psi_Q \rhd R \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec R'\rangle$ **have** $\Psi \otimes \Psi_P \otimes \Psi_Q \rhd R \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec R'$
by(*metis statEqTransition Associativity*)
moreover note $\langle$*extractFrame* $R = \langle A_R, \Psi_R\rangle\rangle$
ultimately have $\Psi \rhd (P \parallel Q) \parallel R \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec ((P \parallel Q') \parallel R')$
using $\langle A_P \sharp* \Psi\rangle$ $\langle A_Q \sharp* \Psi\rangle$ $\langle A_R \sharp* \Psi\rangle$ $\langle A_P \sharp* P\rangle$ $\langle A_Q \sharp* P\rangle$ $\langle A_R \sharp* P\rangle$ $\langle A_P \sharp* Q\rangle$ $\langle A_Q \sharp* Q\rangle$ $\langle A_R \sharp* Q\rangle$ $\langle A_P \sharp* R\rangle$ $\langle A_Q \sharp* R\rangle$ $\langle A_R \sharp* R\rangle$
$\langle A_P \sharp* M\rangle$ $\langle A_Q \sharp* M\rangle$ $\langle A_R \sharp* M\rangle$ $\langle A_R \sharp* A_P\rangle$ $\langle A_Q \sharp* A_R\rangle$ $\langle xvec \sharp* P\rangle$ $\langle xvec \sharp* Q\rangle$
by(*intro BrComm1*) (*assumption* | *simp*)+
moreover have $(\Psi, ((P \parallel Q') \parallel R'), (P \parallel (Q' \parallel R'))) \in Rel$
by(*rule C1*)

ultimately show *?case* **by** *blast*
next
case(*cBrComm2* Ψ_R M *xvec* N Q' A_Q Ψ_Q R' A_R)
from ‹¡M(⦇ν∗*xvec*⦈)⟨N⟩ = α› **have** *xvec* = *bn* α
by(*auto simp add*: *action.inject*)
from ‹¡M(⦇ν∗*xvec*⦈)⟨N⟩ = α› ‹A_P ♯* α›
have A_P ♯* *xvec* **and** A_P ♯* M **and** A_P ♯* N **by** *auto*
from ‹*xvec* = *bn* α› ‹*bn* α ♯* P› ‹*bn* α ♯* Ψ› ‹*bn* α ♯* Ψ_P› **have** *xvec* ♯* P **and** *xvec* ♯* Ψ **and** *xvec* ♯* Ψ_P
by *simp*+
from ‹A_Q ♯* (A_P, Ψ_P, P, Q, R, Ψ)›
have A_Q ♯* P **and** A_Q ♯* Q **and** A_Q ♯* R **and** A_Q ♯* A_P **and** A_Q ♯* Ψ **and** A_Q ♯* Ψ_P **by** *simp*+
from ‹A_R ♯* (A_P, Ψ_P, P, Q, R, Ψ)› **have** A_R ♯* P **and** A_R ♯* Q **and** A_R ♯* R **and** A_R ♯* A_P**and** A_R ♯* Ψ **by** *simp*+

have *FrQ*: *extractFrame* Q = ⟨A_Q, Ψ_Q⟩ **by** *fact*
with ‹A_P ♯* Q› ‹A_Q ♯* A_P› **have** A_P ♯* Ψ_Q
by(*auto dest*: *extractFrameFreshChain*)
have *FrR*: *extractFrame* R = ⟨A_R, Ψ_R⟩ **by** *fact*
with ‹A_P ♯* R› ‹A_R ♯* A_P› **have** A_P ♯* Ψ_R
by(*auto dest*: *extractFrameFreshChain*)

from ‹(Ψ ⊗ Ψ_P) ⊗ Ψ_R ▷ Q ⟼¡M(⦇ν∗*xvec*⦈)⟨N⟩ ≺ Q'› **have** (Ψ ⊗ Ψ_R) ⊗ Ψ_P ▷ Q ⟼¡M(⦇ν∗*xvec*⦈)⟨N⟩ ≺ Q'
by(*metis statEqTransition Associativity Commutativity Composition*)
then have Ψ ⊗ Ψ_R ▷ P ∥ Q ⟼¡M(⦇ν∗*xvec*⦈)⟨N⟩ ≺ (P ∥ Q') **using** *FrP* ‹A_P ♯* Ψ› ‹A_P ♯* Ψ_R› ‹A_P ♯* Q› ‹A_P ♯* M› ‹A_P ♯* N› ‹*xvec* ♯* P› ‹A_P ♯* *xvec*›
by(*intro Par2*) *auto*
moreover from *FrP FrQ* ‹A_P ♯* Ψ_Q› ‹A_Q ♯* A_P› ‹A_Q ♯* Ψ_P› **have** *extractFrame*(P ∥ Q) = ⟨(A_P@A_Q), Ψ_P ⊗ Ψ_Q⟩
by *simp*+
moreover from ‹(Ψ ⊗ Ψ_P) ⊗ Ψ_Q ▷ R ⟼¿M(⦇N⦈) ≺ R'› **have** Ψ ⊗ Ψ_P ⊗ Ψ_Q ▷ R ⟼¿M(⦇N⦈) ≺ R'
by(*metis statEqTransition Associativity*)
moreover note ‹*extractFrame* R = ⟨A_R, Ψ_R⟩›
ultimately have Ψ ▷ (P ∥ Q) ∥ R ⟼¡M(⦇ν∗*xvec*⦈)⟨N⟩ ≺ ((P ∥ Q') ∥ R')
using ‹A_P ♯* Ψ› ‹A_Q ♯* Ψ› ‹A_R ♯* Ψ› ‹A_P ♯* P› ‹A_Q ♯* P› ‹A_R ♯* P› ‹A_P ♯* Q› ‹A_Q ♯* Q› ‹A_R ♯* Q› ‹A_P ♯* R› ‹A_Q ♯* R› ‹A_R ♯* R› ‹A_P ♯* M› ‹A_Q ♯* M› ‹A_R ♯* M› ‹A_R ♯* A_P› ‹A_Q ♯* A_R› ‹*xvec* ♯* R›
by(*auto intro*: *BrComm2*)
moreover have (Ψ, ((P ∥ Q') ∥ R'), (P ∥ (Q' ∥ R'))) ∈ *Rel*
by(*rule C1*)
ultimately show *?case* **by** *blast*
qed
next
case(*cComm1* Ψ_{QR} M N P' A_P Ψ_P K *xvec* QR' A_{QR})
from ‹*xvec* ♯* (Ψ, P, Q, R, α)› **have** *xvec* ♯* Q **and** *xvec* ♯* R **by** *simp*+
from ‹A_{QR} ♯* (Ψ, P, Q, R, α)› **have** A_{QR} ♯* Q **and** A_{QR} ♯* R **and** A_{QR} ♯*

Ψ **by** *simp+*
from ‹$A_P \sharp* (Q \parallel R)$› **have** $A_P \sharp* Q$ **and** $A_P \sharp* R$ **by** *simp+*
have *PTrans*: $\Psi \otimes \Psi_{QR} \rhd P \longmapsto M(\!|N|\!) \prec P'$ **and** *FrP*: *extractFrame* $P = \langle A_P, \Psi_P\rangle$ **and** *MeqK*: $\Psi \otimes \Psi_P \otimes \Psi_{QR} \vdash M \leftrightarrow K$ **by** *fact+*
note ‹$\Psi \otimes \Psi_P \rhd Q \parallel R \longmapsto K(\!|\nu*xvec|\!)\langle N\rangle \prec QR'$›
moreover from ‹*xvec* $\sharp* \Psi$› ‹*xvec* $\sharp* \Psi_P$› **have** *xvec* $\sharp* (\Psi \otimes \Psi_P)$ **by** *force*
moreover note ‹*xvec* $\sharp* Q$›‹*xvec* $\sharp* R$› ‹*xvec* $\sharp* K$›
‹*extractFrame*$(Q \parallel R) = \langle A_{QR}, \Psi_{QR}\rangle$› ‹*distinct* A_{QR}›
moreover from ‹$A_{QR} \sharp* \Psi$› ‹$A_{QR} \sharp* \Psi_P$› **have** $A_{QR} \sharp* (\Psi \otimes \Psi_P)$ **by** *force*
ultimately show *?case* **using** ‹$A_{QR} \sharp* Q$› ‹$A_{QR} \sharp* R$› ‹$A_{QR} \sharp* K$›
proof(*induct rule*: *parCasesOutputFrame*)
case(*cPar1* Q' A_Q Ψ_Q A_R Ψ_R)
have *Aeq*: $A_{QR} = A_Q@A_R$ **and** Ψeq: $\Psi_{QR} = \Psi_Q \otimes \Psi_R$ **by** *fact+*
from *PTrans* Ψeq **have** $(\Psi \otimes \Psi_R) \otimes \Psi_Q \rhd P \longmapsto M(\!|N|\!) \prec P'$
by(*metis statEqTransition Associativity Commutativity Composition*)
moreover note *FrP*
moreover from ‹$(\Psi \otimes \Psi_P) \otimes \Psi_R \rhd Q \longmapsto K(\!|\nu*xvec|\!)\langle N\rangle \prec Q'$› **have** $(\Psi \otimes \Psi_R) \otimes \Psi_P \rhd Q \longmapsto K(\!|\nu*xvec|\!)\langle N\rangle \prec Q'$
by(*metis statEqTransition Associativity Commutativity Composition*)
moreover note ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$›
moreover from *MeqK* Ψeq **have** $(\Psi \otimes \Psi_R) \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K$
by(*metis statEqEnt Associativity Commutativity Composition*)
moreover from ‹$A_P \sharp* R$› ‹$A_P \sharp* A_{QR}$› *Aeq* ‹*extractFrame* $R = \langle A_R, \Psi_R\rangle$› **have** $A_P \sharp* A_Q$ **and** $A_P \sharp* \Psi_R$
by(*auto dest*: *extractFrameFreshChain*)
moreover from ‹$A_{QR} \sharp* P$› ‹$A_{QR} \sharp* \Psi$› *Aeq* **have** $A_Q \sharp* P$ **and** $A_Q \sharp* \Psi$ **by** *simp+*
ultimately have $\Psi \otimes \Psi_R \rhd P \parallel Q \longmapsto \tau \prec (\!|\nu*xvec|\!)(P' \parallel Q')$
using ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_P \sharp* Q$› ‹$A_P \sharp* M$› ‹$A_P \sharp* A_Q$› ‹$A_Q \sharp* \Psi$› ‹$A_Q \sharp* \Psi_R$› ‹$A_Q \sharp* P$› ‹$A_Q \sharp* Q$› ‹$A_Q \sharp* K$› ‹*xvec* $\sharp* P$›
by(*intro Comm1*) (*assumption* | *force*)*+*
moreover from ‹$A_{QR} \sharp* \Psi$› *Aeq* **have** $A_R \sharp* \Psi$ **by** *simp*
moreover from ‹$A_{QR} \sharp* P$› *Aeq* **have** $A_R \sharp* P$ **by** *simp*
ultimately have $\Psi \rhd (P \parallel Q) \parallel R \longmapsto \tau \prec ((\!|\nu*xvec|\!)(P' \parallel Q')) \parallel R$ **using** ‹*extractFrame* $R = \langle A_R, \Psi_R\rangle$› ‹$A_R \sharp* Q$›
by(*intro Par1*) (*assumption* | *simp*)*+*
moreover from ‹*xvec* $\sharp* \Psi$› ‹*xvec* $\sharp* R$› **have** $(\Psi, ((\!|\nu*xvec|\!)(P' \parallel Q')) \parallel R, (\!|\nu*xvec|\!)(P' \parallel (Q' \parallel R))) \in Rel$
by(*rule C3*)
ultimately show *?case* **by** *blast*
next
case(*cPar2* R' A_Q Ψ_Q A_R Ψ_R)
have *Aeq*: $A_{QR} = A_Q@A_R$ **and** Ψeq: $\Psi_{QR} = \Psi_Q \otimes \Psi_R$ **by** *fact+*
from ‹$A_{QR} \sharp* \Psi$› ‹$A_{QR} \sharp* P$› ‹$A_{QR} \sharp* \Psi_P$› ‹$A_P \sharp* A_{QR}$› *Aeq* **have** $A_R \sharp* \Psi$ **and** $A_R \sharp* \Psi_P$ **and** $A_P \sharp* A_R$ **and** $A_P \sharp* A_Q$ **and** $A_R \sharp* P$ **by** *simp+*
from ‹$A_{QR} \sharp* \Psi$› *Aeq* **have** $A_Q \sharp* \Psi$ **by** *simp*
from ‹$A_{QR} \sharp* P$› ‹$A_P \sharp* A_{QR}$› *Aeq FrP* **have** $A_Q \sharp* \Psi_P$ **by**(*auto dest*: *extractFrameFreshChain*)
from ‹$A_P \sharp* A_{QR}$› ‹*extractFrame* $R = \langle A_R, \Psi_R\rangle$› ‹*extractFrame* $Q = \langle A_Q,$

$\Psi_Q\rangle$ *Aeq* $\langle A_P \sharp* Q\rangle$ $\langle A_P \sharp* R\rangle$ **have** $A_P \sharp* \Psi_Q$ **and** $A_P \sharp* \Psi_R$ **by**(*auto dest*: *extractFrameFreshChain*)

have *RTrans*: $(\Psi \otimes \Psi_P) \otimes \Psi_Q \rhd R \longmapsto K(\!|\nu* xvec|\!)\langle N\rangle \prec R'$ **and** *FrR*: *extractFrame* $R = \langle A_R, \Psi_R\rangle$ **by** *fact+*

then have $(\Psi \otimes \Psi_P) \otimes \Psi_Q \rhd R \longmapsto ROut\ K\ ((\!|\nu* xvec|\!)N \prec' R')$ **by**(*simp add*: *residualInject*)

then obtain K' **where** *KeqK'*: $((\Psi \otimes \Psi_P) \otimes \Psi_Q) \otimes \Psi_R \vdash K \leftrightarrow K'$ **and** $A_P \sharp* K'$ **and** $A_Q \sharp* K'$

using $\langle A_P \sharp* R\rangle$ $\langle A_Q \sharp* R\rangle$ $\langle A_R \sharp* \Psi\rangle$ $\langle A_R \sharp* \Psi_P\rangle$ $\langle A_R \sharp* \Psi_Q\rangle$ $\langle A_Q \sharp* A_R\rangle$ $\langle A_P \sharp* A_R\rangle$ $\langle A_R \sharp* R\rangle$ $\langle A_R \sharp* R\rangle$ $\langle A_R \sharp* K\rangle$ $\langle$*distinct* $A_R\rangle$ $\langle$*xvec* $\sharp* K\rangle$ $\langle$*distinct xvec*$\rangle$ *FrR*

using *outputObtainPrefix*[**where** B=A_P@A_Q]

by (*smt* (*verit*, *ccfv-threshold*) *freshChainAppend freshChainSym freshCompChain*(*1*))

from *PTrans* Ψ*eq* **have** $(\Psi \otimes \Psi_R) \otimes \Psi_Q \rhd P \longmapsto M(\!|N|\!) \prec P'$

by(*metis statEqTransition Associativity Commutativity Composition*)

moreover from *MeqK KeqK'* Ψ*eq* **have** *MeqK'*: $((\Psi \otimes \Psi_R) \otimes \Psi_Q) \otimes \Psi_P \vdash M \leftrightarrow K'$

by(*metis statEqEnt Associativity Commutativity Composition chanEqTrans*)

ultimately have $(\Psi \otimes \Psi_R) \otimes \Psi_Q \rhd P \longmapsto K'(\!|N|\!) \prec P'$ **using** *FrP* $\langle$*distinct* $A_P\rangle$ $\langle A_P \sharp* \Psi\rangle$ $\langle A_P \sharp* P\rangle$ $\langle A_P \sharp* M\rangle$ $\langle A_P \sharp* K'\rangle$ $\langle A_P \sharp* \Psi_Q\rangle$ $\langle A_P \sharp* \Psi_R\rangle$

by(*auto intro*: *inputRenameSubject*)

moreover from $\langle A_{Q\,R} \sharp* P\rangle$ $\langle A_{Q\,R} \sharp* N\rangle$ *Aeq* **have** $A_Q \sharp* P$ **and** $A_Q \sharp* N$ **by** *simp+*

ultimately have $\Psi \otimes \Psi_R \rhd P \parallel Q \longmapsto K'(\!|N|\!) \prec P' \parallel Q$ **using** $\langle$*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle\rangle$ $\langle A_Q \sharp* \Psi_R\rangle$ $\langle A_Q \sharp* K'\rangle$ $\langle A_Q \sharp* \Psi\rangle$

by(*intro Par1*) (*assumption* | *force*)+

moreover from *FrP* $\langle$*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle\rangle$ $\langle A_P \sharp* A_Q\rangle$ $\langle A_P \sharp* \Psi_Q\rangle$ $\langle A_Q \sharp* \Psi_P\rangle$

have *extractFrame*$(P \parallel Q) = \langle (A_P@A_Q), \Psi_P \otimes \Psi_Q\rangle$ **by** *simp+*

moreover from *RTrans* **have** $\Psi \otimes (\Psi_P \otimes \Psi_Q) \rhd R \longmapsto K(\!|\nu* xvec|\!)\langle N\rangle \prec R'$ **by**(*metis Associativity statEqTransition*)

moreover note *FrR*

moreover from *MeqK' KeqK'* **have** $\Psi \otimes (\Psi_P \otimes \Psi_Q) \otimes \Psi_R \vdash K' \leftrightarrow K$

by(*metis statEqEnt Associativity Commutativity Composition chanEqTrans chanEqSym*)

ultimately have $\Psi \rhd (P \parallel Q) \parallel R \longmapsto \tau \prec (\!|\nu* xvec|\!)((P' \parallel Q) \parallel R')$

using $\langle A_P \sharp* \Psi\rangle$ $\langle A_Q \sharp* \Psi\rangle$ $\langle A_P \sharp* P\rangle$ $\langle A_Q \sharp* P\rangle$ $\langle A_P \sharp* Q\rangle$ $\langle A_Q \sharp* Q\rangle$ $\langle A_P \sharp* R\rangle$ $\langle A_Q \sharp* R\rangle$ $\langle A_P \sharp* K'\rangle$ $\langle A_Q \sharp* K'\rangle$ $\langle A_P \sharp* A_R\rangle$ $\langle A_Q \sharp* A_R\rangle$ $\langle A_R \sharp* \Psi\rangle$ $\langle A_R \sharp* P\rangle$ $\langle A_R \sharp* Q\rangle$ $\langle A_R \sharp* R\rangle$ $\langle A_R \sharp* K\rangle$ $\langle$*xvec* $\sharp* P\rangle$ $\langle$*xvec* $\sharp* Q\rangle$

by(*intro Comm1*) (*assumption* | *simp*)+

moreover from $\langle$*xvec* $\sharp* \Psi\rangle$ **have** $(\Psi, (\!|\nu* xvec|\!)((P' \parallel Q) \parallel R'), (\!|\nu* xvec|\!)(P' \parallel (Q \parallel R'))) \in Rel$

by(*metis C1 C4*)

ultimately show *?case* **by** *blast*

qed

next

case(*cComm2* $\Psi_{Q\,R}$ *M xvec N P'* A_P Ψ_P *K QR'* $A_{Q\,R}$)

from ‹A_{QR} ♯* (Ψ, P, Q, R, α)› **have** A_{QR} ♯* Q **and** A_{QR} ♯* R **and** A_{QR} ♯* Ψ **by** *simp+*

from ‹A_P ♯* (Q ∥ R)› ‹*xvec* ♯* (Q ∥ R)› **have** A_P ♯* Q **and** A_P ♯* R **and** *xvec* ♯* Q **and** *xvec* ♯* R **by** *simp+*

have *PTrans*: $\Psi \otimes \Psi_{QR} \rhd P \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$ **and** *FrP*: *extractFrame* $P = \langle A_P, \Psi_P\rangle$ **and** *MeqK*: $\Psi \otimes \Psi_P \otimes \Psi_{QR} \vdash M \leftrightarrow K$ **by** *fact+*

note ‹$\Psi \otimes \Psi_P \rhd Q \parallel R \longmapsto K(\!|N|\!) \prec QR'$› ‹*extractFrame*($Q \parallel R$) = $\langle A_{QR}, \Psi_{QR}\rangle$› ‹*distinct* A_{QR}›

moreover from ‹A_{QR} ♯* Ψ› ‹A_{QR} ♯* Ψ_P› **have** A_{QR} ♯* ($\Psi \otimes \Psi_P$) **by** *force*

ultimately show *?case* **using** ‹A_{QR} ♯* Q› ‹A_{QR} ♯* R› ‹A_{QR} ♯* K›

proof(*induct rule*: *parCasesInputFrame*)

case(*cPar1* Q' A_Q Ψ_Q A_R Ψ_R)

have *Aeq*: $A_{QR} = A_Q @ A_R$ **and** Ψ*eq*: $\Psi_{QR} = \Psi_Q \otimes \Psi_R$ **by** *fact+*

from *PTrans* Ψ*eq* **have** $(\Psi \otimes \Psi_R) \otimes \Psi_Q \rhd P \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$

by(*metis statEqTransition Associativity Commutativity Composition*)

moreover note *FrP*

moreover from ‹$(\Psi \otimes \Psi_P) \otimes \Psi_R \rhd Q \longmapsto K(\!|N|\!) \prec Q'$› **have** $(\Psi \otimes \Psi_R) \otimes \Psi_P \rhd Q \longmapsto K(\!|N|\!) \prec Q'$

by(*metis statEqTransition Associativity Commutativity Composition*)

moreover note ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$›

moreover from *MeqK* Ψ*eq* **have** $(\Psi \otimes \Psi_R) \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K$

by(*metis statEqEnt Associativity Commutativity Composition*)

moreover from ‹A_P ♯* Q› ‹A_P ♯* R› ‹A_P ♯* A_{QR}› *Aeq* ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› ‹*extractFrame* $R = \langle A_R, \Psi_R\rangle$›

have A_P ♯* A_Q **and** A_P ♯* Ψ_R **by**(*auto dest*: *extractFrameFreshChain*)

moreover from ‹A_{QR} ♯* Ψ› ‹A_{QR} ♯* P› *Aeq* **have** A_Q ♯* Ψ **and** A_Q ♯* P **by** *simp+*

ultimately have $\Psi \otimes \Psi_R \rhd P \parallel Q \longmapsto \tau \prec (\!|\nu * xvec|\!)(P' \parallel Q')$

using ‹A_P ♯* Ψ› ‹A_P ♯* P› ‹A_P ♯* Q› ‹A_P ♯* M› ‹A_P ♯* A_Q› ‹A_Q ♯* Ψ› ‹A_Q ♯* Ψ_R› ‹A_Q ♯* P› ‹A_Q ♯* Q› ‹A_Q ♯* K› ‹*xvec* ♯* Q›

by(*intro Comm2*) (*assumption* | *force*)+

moreover from ‹A_{QR} ♯* Ψ› ‹A_{QR} ♯* P› *Aeq* **have** A_R ♯* Ψ **and** A_R ♯* P **by** *simp+*

ultimately have $\Psi \rhd (P \parallel Q) \parallel R \longmapsto \tau \prec ((\!|\nu * xvec|\!)(P' \parallel Q')) \parallel R$ **using** ‹*extractFrame* $R = \langle A_R, \Psi_R\rangle$› ‹$A_R$ ♯* Q›

by(*intro Par1*) (*assumption* | *simp*)+

moreover from ‹*xvec* ♯* Ψ› ‹*xvec* ♯* R› **have** (Ψ, $(\!|\nu * xvec|\!)(P' \parallel Q')) \parallel R$, $(\!|\nu * xvec|\!)(P' \parallel (Q' \parallel R))) \in Rel$

by(*rule C3*)

ultimately show *?case* **by** *blast*

next

case(*cPar2* R' A_Q Ψ_Q A_R Ψ_R)

have *Aeq*: $A_{QR} = A_Q @ A_R$ **and** Ψ*eq*: $\Psi_{QR} = \Psi_Q \otimes \Psi_R$ **by** *fact+*

from ‹A_{QR} ♯* Ψ› ‹A_{QR} ♯* P› ‹A_{QR} ♯* Ψ_P› ‹A_P ♯* A_{QR}› *Aeq*

have A_R ♯* Ψ **and** A_R ♯* Ψ_P **and** A_P ♯* A_R **and** A_P ♯* A_Q **and** A_R ♯* P **and** A_Q ♯* Ψ **and** A_Q ♯* Ψ_P **by** *simp+*

have *RTrans*: $(\Psi \otimes \Psi_P) \otimes \Psi_Q \rhd R \longmapsto K(\!|N|\!) \prec R'$ **and** *FrR*: *extractFrame* $R = \langle A_R, \Psi_R\rangle$ **by** *fact+*

then obtain K' **where** *KeqK'*: $((\Psi \otimes \Psi_P) \otimes \Psi_Q) \otimes \Psi_R \vdash K \leftrightarrow K'$ **and**

$A_P \sharp* K'$ **and** $A_Q \sharp* K'$
using ‹$A_P \sharp* R$› ‹$A_Q \sharp* R$› ‹$A_R \sharp* \Psi$› ‹$A_R \sharp* \Psi_P$› ‹$A_R \sharp* \Psi_Q$› ‹$A_Q \sharp* A_R$› ‹$A_P \sharp* A_R$› ‹$A_R \sharp* R$› ‹$A_R \sharp* R$› ‹$A_R \sharp* K$› ‹*distinct* A_R›
using *inputObtainPrefix*[**where** $B{=}A_P@A_Q$]
by (*smt* (*verit*, *ccfv-threshold*) *freshChainAppend freshChainSym freshCompChain*(*1*))
from *PTrans* Ψeq **have** $(\Psi \otimes \Psi_R) \otimes \Psi_Q \rhd P \longmapsto M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P'$
by(*metis statEqTransition Associativity Commutativity Composition*)
moreover from *MeqK KeqK'* Ψeq **have** *MeqK'*: $((\Psi \otimes \Psi_R) \otimes \Psi_Q) \otimes \Psi_P \vdash M \leftrightarrow K'$
by(*metis statEqEnt Associativity Commutativity Composition chanEqTrans*)
moreover from ‹$A_P \sharp* R$› ‹$A_P \sharp* Q$› ‹$A_P \sharp* A_{QR}$› *FrR* ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› *Aeq* **have** $A_P \sharp* \Psi_Q$ **and** $A_P \sharp* \Psi_R$
by(*auto dest*: *extractFrameFreshChain*)
ultimately have $(\Psi \otimes \Psi_R) \otimes \Psi_Q \rhd P \longmapsto K'(\!|\nu{*}xvec|\!)\langle N\rangle \prec P'$ **using** *FrP* ‹*distinct* A_P› ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_P \sharp* M$› ‹$A_P \sharp* K'$›
by(*auto intro*: *outputRenameSubject*)
moreover from ‹$A_{QR} \sharp* P$› ‹$A_{QR} \sharp* N$› ‹$A_{QR} \sharp* xvec$› *Aeq* **have** $A_Q \sharp* P$ **and** $A_Q \sharp* N$ **and** $A_Q \sharp* xvec$ **by** *simp*+
ultimately have $\Psi \otimes \Psi_R \rhd P \parallel Q \longmapsto K'(\!|\nu{*}xvec|\!)\langle N\rangle \prec (P' \parallel Q)$ **using** ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› ‹$A_Q \sharp* \Psi_R$› ‹$A_Q \sharp* K'$› ‹$xvec \sharp* Q$› ‹$A_Q \sharp* \Psi$›
by(*intro Par1*) (*assumption* | *force*)+
moreover from *FrP* ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› ‹$A_P \sharp* A_Q$› ‹$A_P \sharp* \Psi_Q$› ‹$A_Q \sharp* \Psi_P$›
have *extractFrame*$(P \parallel Q) = \langle (A_P@A_Q), \Psi_P \otimes \Psi_Q\rangle$ **by** *simp*+
moreover from *RTrans* **have** $\Psi \otimes (\Psi_P \otimes \Psi_Q) \rhd R \longmapsto K(\!|N|\!) \prec R'$ **by**(*metis Associativity statEqTransition*)
moreover note *FrR*
moreover from *MeqK' KeqK'* **have** $\Psi \otimes (\Psi_P \otimes \Psi_Q) \otimes \Psi_R \vdash K' \leftrightarrow K$
by(*metis statEqEnt Associativity Commutativity Composition chanEqTrans chanEqSym*)
ultimately have $\Psi \rhd (P \parallel Q) \parallel R \longmapsto \tau \prec (\!|\nu{*}xvec|\!)((P' \parallel Q) \parallel R')$
using ‹$A_P \sharp* \Psi$› ‹$A_Q \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_Q \sharp* P$› ‹$A_P \sharp* Q$› ‹$A_Q \sharp* Q$› ‹$A_P \sharp* R$› ‹$A_Q \sharp* R$› ‹$A_P \sharp* K'$› ‹$A_Q \sharp* K'$› ‹$A_P \sharp* A_R$› ‹$A_Q \sharp* A_R$›
‹$A_R \sharp* \Psi$› ‹$A_R \sharp* P$› ‹$A_R \sharp* Q$› ‹$A_R \sharp* R$› ‹$A_R \sharp* K$› ‹$xvec \sharp* R$›
by(*intro Comm2*) (*assumption* | *simp*)+
moreover from ‹$xvec \sharp* \Psi$› **have** $(\Psi, (\!|\nu{*}xvec|\!)((P' \parallel Q) \parallel R'), (\!|\nu{*}xvec|\!)(P' \parallel (Q \parallel R'))) \in Rel$
by(*metis C1 C4*)
ultimately show *?case* **by** *blast*
qed
next
case(*cBrMerge* Ψ_{QR} *M N P'* A_P Ψ_P QR' A_{QR})
from ‹$A_{QR} \sharp* (\Psi, P, Q, R, \alpha)$› **have** $A_{QR} \sharp* Q$ **and** $A_{QR} \sharp* R$ **and** $A_{QR} \sharp* \Psi$ **by** *simp*+
from ‹$A_P \sharp* (Q \parallel R)$› **have** $A_P \sharp* Q$ **and** $A_P \sharp* R$ **by** *simp*+
have *PTrans*: $\Psi \otimes \Psi_{QR} \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$ **and** *FrP*: *extractFrame* $P = \langle A_P, \Psi_P\rangle$ **by** *fact*+
note ‹$\Psi \otimes \Psi_P \rhd Q \parallel R \longmapsto ¿M(\!|N|\!) \prec QR'$› ‹*extractFrame*$(Q \parallel R) = \langle A_{QR},$

$\Psi_{QR}\rangle\rangle$ ‹*distinct* A_{QR}›
moreover from ‹A_{QR} ♯* Ψ› ‹A_{QR} ♯* Ψ_P› **have** A_{QR} ♯* ($\Psi \otimes \Psi_P$) **by** *force*
ultimately show *?case* **using** ‹A_{QR} ♯* Q› ‹A_{QR} ♯* R› ‹A_{QR} ♯* M› ‹A_{QR} ♯* N› ‹A_{QR} ♯* P› ‹A_{QR} ♯* M› ‹A_{QR} ♯* Ψ›
proof(*induct rule*: *parCasesBrInputFrame*)
case(*cPar1* Q' A_Q Ψ_Q A_R Ψ_R)
from ‹A_{QR} ♯* N› ‹$A_{QR} = A_Q$ @ A_R› **have** A_Q ♯* N **and** A_R ♯* N
by *simp+*
have *Aeq*: $A_{QR} = A_Q @ A_R$ **and** Ψ*eq*: $\Psi_{QR} = \Psi_Q \otimes \Psi_R$ **by** *fact+*
from *PTrans* Ψ*eq* **have** ($\Psi \otimes \Psi_R$) $\otimes$ $\Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$
by(*metis statEqTransition Associativity Commutativity Composition*)
moreover note *FrP*
moreover from ‹($\Psi \otimes \Psi_P$) $\otimes$ $\Psi_R \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$› **have** ($\Psi \otimes \Psi_R$) $\otimes$ $\Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$
by(*metis statEqTransition Associativity Commutativity Composition*)
moreover note ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$›
moreover from ‹A_P ♯* Q› ‹A_P ♯* R› ‹A_P ♯* A_{QR}› *Aeq* ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› ‹*extractFrame* $R = \langle A_R, \Psi_R\rangle$›
have A_P ♯* A_Q **and** A_P ♯* Ψ_R **by**(*auto dest*: *extractFrameFreshChain*)
moreover from ‹A_{QR} ♯* Ψ› ‹A_{QR} ♯* P› *Aeq* **have** A_Q ♯* Ψ **and** A_Q ♯* P **by** *simp+*
ultimately have $\Psi \otimes \Psi_R \rhd P \parallel Q \longmapsto ¿M(\!|N|\!) \prec (P' \parallel Q')$
using ‹A_P ♯* Ψ› ‹A_P ♯* P› ‹A_P ♯* Q› ‹A_P ♯* M› ‹A_P ♯* A_Q› ‹A_Q ♯* Ψ› ‹A_Q ♯* Ψ_R› ‹A_Q ♯* P› ‹A_Q ♯* Q› ‹A_Q ♯* M›
by(*intro BrMerge*) (*assumption* | *force*)+
moreover from ‹A_{QR} ♯* Ψ› ‹A_{QR} ♯* P› *Aeq* **have** A_R ♯* Ψ **and** A_R ♯* P **by** *simp+*
ultimately have $\Psi \rhd (P \parallel Q) \parallel R \longmapsto ¿M(\!|N|\!) \prec (P' \parallel Q') \parallel R$ **using** ‹*extractFrame* $R = \langle A_R, \Psi_R\rangle$› ‹$A_R$ ♯* Q› ‹A_R ♯* M› ‹A_R ♯* N›
by(*intro Par1*) (*assumption* | *simp*)+
moreover have (Ψ, ($P' \parallel Q'$) $\parallel R$, ($P' \parallel$ ($Q' \parallel R$))) $\in$ *Rel*
by(*rule C1*)
ultimately show *?case* **by** *blast*
next
case(*cPar2* R' A_Q Ψ_Q A_R Ψ_R)
from ‹A_{QR} ♯* N› ‹$A_{QR} = A_Q$ @ A_R› **have** A_Q ♯* N **and** A_R ♯* N
by *simp+*
have *Aeq*: $A_{QR} = A_Q @ A_R$ **and** Ψ*eq*: $\Psi_{QR} = \Psi_Q \otimes \Psi_R$ **by** *fact+*
from ‹A_{QR} ♯* Ψ› ‹A_{QR} ♯* P› ‹A_{QR} ♯* Ψ_P› ‹A_P ♯* A_{QR}› *Aeq*
have A_R ♯* Ψ **and** A_R ♯* Ψ_P **and** A_P ♯* A_R **and** A_P ♯* A_Q **and** A_R ♯* P **and** A_Q ♯* Ψ **and** A_Q ♯* Ψ_P **by** *simp+*
have *RTrans*: ($\Psi \otimes \Psi_P$) $\otimes$ $\Psi_Q \rhd R \longmapsto ¿M(\!|N|\!) \prec R'$ **and** *FrR*: *extractFrame* $R = \langle A_R, \Psi_R\rangle$ **by** *fact+*
from *PTrans* Ψ*eq* **have** ($\Psi \otimes \Psi_R$) $\otimes$ $\Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$
by(*metis statEqTransition Associativity Commutativity Composition*)
moreover from ‹A_P ♯* R› ‹A_P ♯* Q› ‹A_P ♯* A_{QR}› *FrR* ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› *Aeq* **have** A_P ♯* Ψ_Q **and** A_P ♯* Ψ_R
by(*auto dest*: *extractFrameFreshChain*)
moreover from ‹A_{QR} ♯* P› ‹A_{QR} ♯* N› *Aeq* **have** A_Q ♯* P **and** A_Q ♯* N

by *simp+*
ultimately have $\Psi \otimes \Psi_R \rhd P \parallel Q \longmapsto ¿M(\!|N|\!) \prec (P' \parallel Q)$ **using** ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› ‹$A_Q \sharp\!* \Psi_R$› ‹$A_Q \sharp\!* M$› ‹$A_Q \sharp\!* \Psi$›
by(*intro Par1*) (*assumption* | *force*)+
moreover from *FrP* ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› ‹$A_P \sharp\!* A_Q$› ‹$A_P \sharp\!* \Psi_Q$› ‹$A_Q \sharp\!* \Psi_P$›
have *extractFrame*$(P \parallel Q) = \langle (A_P @ A_Q), \Psi_P \otimes \Psi_Q\rangle$ **by** *simp+*
moreover from *RTrans* **have** $\Psi \otimes (\Psi_P \otimes \Psi_Q) \rhd R \longmapsto ¿M(\!|N|\!) \prec R'$ **by**(*metis Associativity statEqTransition*)
moreover note *FrR*
ultimately have $\Psi \rhd (P \parallel Q) \parallel R \longmapsto ¿M(\!|N|\!) \prec ((P' \parallel Q) \parallel R')$
using ‹$A_P \sharp\!* \Psi$› ‹$A_Q \sharp\!* \Psi$› ‹$A_P \sharp\!* P$› ‹$A_Q \sharp\!* P$› ‹$A_P \sharp\!* Q$› ‹$A_Q \sharp\!* Q$› ‹$A_P \sharp\!* R$› ‹$A_Q \sharp\!* R$› ‹$A_P \sharp\!* M$› ‹$A_Q \sharp\!* M$› ‹$A_P \sharp\!* A_R$› ‹$A_Q \sharp\!* A_R$› ‹$A_R \sharp\!* \Psi$› ‹$A_R \sharp\!* P$› ‹$A_R \sharp\!* Q$› ‹$A_R \sharp\!* R$› ‹$A_R \sharp\!* M$›
by(*auto intro*: *BrMerge*)
moreover have $(\Psi, ((P' \parallel Q) \parallel R'), (P' \parallel (Q \parallel R'))) \in Rel$
by(*rule C1*)
ultimately show *?case* **by** *blast*
next
case(*cBrMerge* $\Psi_R\ Q'\ A_Q\ \Psi_Q\ R'\ A_R$)
have *Aeq*: $A_{QR} = A_Q @ A_R$ **and** Ψeq: $\Psi_{QR} = \Psi_Q \otimes \Psi_R$ **by** *fact+*
from ‹$A_{QR} \sharp\!* N$› ‹$A_{QR} \sharp\!* M$› ‹$A_{QR} \sharp\!* P$› ‹$A_{QR} \sharp\!* \Psi$› ‹$A_{QR} \sharp\!* \Psi_P$› ‹$A_P \sharp\!* A_{QR}$› *Aeq* Ψeq
have $A_Q \sharp\!* N$ **and** $A_R \sharp\!* N$ **and** $A_Q \sharp\!* M$ **and** $A_R \sharp\!* M$
and $A_Q \sharp\!* \Psi$ **and** $A_R \sharp\!* \Psi$ **and** $A_P \sharp\!* A_Q$ **and** $A_P \sharp\!* A_R$
and $A_Q \sharp\!* \Psi_P$ **and** $A_R \sharp\!* \Psi_P$
and $A_Q \sharp\!* P$ **and** $A_R \sharp\!* P$
by *simp+*

from ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› ‹$A_P \sharp\!* Q$› ‹$A_P \sharp\!* A_Q$›
have $A_P \sharp\!* \Psi_Q$
by (*metis extractFrameFreshChain freshFrameDest*)
from *Aeq* Ψeq *PTrans* **have** $\Psi \otimes (\Psi_Q \otimes \Psi_R) \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$ **by** *simp*
then have $\Psi \otimes (\Psi_R \otimes \Psi_Q) \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$
by (*metis Commutativity* Ψeq *compositionSym statEqTransition*)
then have $(\Psi \otimes \Psi_R) \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$
by (*metis AssertionStatEqSym Associativity statEqTransition*)
moreover note *FrP*
moreover from ‹$(\Psi \otimes \Psi_P) \otimes \Psi_R \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$› **have** $(\Psi \otimes \Psi_R) \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$
by (*metis associativitySym statEqTransition*)
moreover note ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$›
moreover note ‹$A_P \sharp\!* \Psi$› ‹$A_Q \sharp\!* \Psi$› ‹$A_Q \sharp\!* \Psi_R$›
moreover from ‹*extractFrame* $R = \langle A_R, \Psi_R\rangle$› ‹$A_P \sharp\!* A_R$› ‹$A_P \sharp\!* R$› **have** $A_P \sharp\!* \Psi_R$
by (*metis extractFrameFreshChain freshFrameDest*)
moreover note ‹$A_P \sharp\!* P$› ‹$A_P \sharp\!* Q$› ‹$A_P \sharp\!* M$› ‹$A_P \sharp\!* A_Q$› ‹$A_Q \sharp\!* P$› ‹$A_Q \sharp\!* Q$› ‹$A_Q \sharp\!* M$›
ultimately have $(\Psi \otimes \Psi_R) \rhd (P \parallel Q) \longmapsto ¿M(\!|N|\!) \prec (P' \parallel Q')$

by(*intro BrMerge*) (*assumption* | *force*)+

from ‹$(\Psi \otimes \Psi_P) \otimes \Psi_Q \rhd R \longmapsto ¿M(\!|N|\!) \prec R'$› **have** $\Psi \otimes (\Psi_P \otimes \Psi_Q) \rhd R \longmapsto ¿M(\!|N|\!) \prec R'$
by (*metis Associativity statEqTransition*)

note ‹$(\Psi \otimes \Psi_R) \rhd (P \parallel Q) \longmapsto ¿M(\!|N|\!) \prec (P' \parallel Q')$›
moreover from *FrP* ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› ‹$A_P \sharp* A_Q$› ‹$A_P \sharp* \Psi_Q$› ‹$A_Q \sharp* \Psi_P$›
have *extractFrame*$(P \parallel Q) = \langle (A_P @ A_Q), (\Psi_P \otimes \Psi_Q)\rangle$ **by** *simp*
moreover note ‹$\Psi \otimes (\Psi_P \otimes \Psi_Q) \rhd R \longmapsto ¿M(\!|N|\!) \prec R'$› ‹*extractFrame* $R = \langle A_R, \Psi_R\rangle$›
moreover note ‹$A_P \sharp* \Psi$› ‹$A_Q \sharp* \Psi$› ‹$A_R \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_Q \sharp* P$› ‹$A_P \sharp* Q$› ‹$A_Q \sharp* Q$›
‹$A_P \sharp* R$› ‹$A_Q \sharp* R$› ‹$A_P \sharp* M$› ‹$A_Q \sharp* M$› ‹$A_P \sharp* A_R$› ‹$A_Q \sharp* A_R$› ‹$A_R \sharp* P$› ‹$A_R \sharp* Q$› ‹$A_R \sharp* R$› ‹$A_R \sharp* M$›
ultimately have $\Psi \rhd (P \parallel Q) \parallel R \longmapsto ¿M(\!|N|\!) \prec (P' \parallel Q') \parallel R'$
by(*auto intro*: *BrMerge*)
moreover have $(\Psi, (P' \parallel Q') \parallel R', (P' \parallel (Q' \parallel R'))) \in Rel$
by(*rule C1*)
ultimately show *?case* **by** *blast*
qed
next
case(*cBrComm1* Ψ_{QR} M N P' A_P Ψ_P *xvec* QR' A_{QR})
from ‹$¡M(\!|\nu * xvec|\!)\langle N\rangle = \alpha$› ‹*bn* $\alpha \sharp* Q$› ‹*bn* $\alpha \sharp* R$› **have** *xvec* $\sharp*$ Q **and** *xvec* $\sharp*$ R **by** *auto*
from ‹$A_{QR} \sharp* (\Psi, P, Q, R, \alpha)$› **have** $A_{QR} \sharp* Q$ **and** $A_{QR} \sharp* R$ **and** $A_{QR} \sharp* \Psi$ **by** *simp*+
from ‹$A_P \sharp* (Q \parallel R)$› **have** $A_P \sharp* Q$ **and** $A_P \sharp* R$ **by** *simp*+
have *PTrans*: $\Psi \otimes \Psi_{QR} \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$ **and** *FrP*: *extractFrame* $P = \langle A_P, \Psi_P\rangle$ **by** *fact*+
note ‹$\Psi \otimes \Psi_P \rhd Q \parallel R \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec QR'$›
moreover from ‹*xvec* $\sharp*$ Ψ› ‹*xvec* $\sharp*$ Ψ_P› **have** *xvec* $\sharp*$ $(\Psi \otimes \Psi_P)$ **by** *force*
moreover note ‹*xvec* $\sharp*$ Q› ‹*xvec* $\sharp*$ R› ‹*xvec* $\sharp*$ M›
‹*extractFrame*$(Q \parallel R) = \langle A_{QR}, \Psi_{QR}\rangle$› ‹*distinct* A_{QR}›
moreover from ‹$A_{QR} \sharp* \Psi$› ‹$A_{QR} \sharp* \Psi_P$› **have** $A_{QR} \sharp* (\Psi \otimes \Psi_P)$ **by** *force*
ultimately show *?case* **using** ‹$A_{QR} \sharp* Q$› ‹$A_{QR} \sharp* R$› ‹$A_{QR} \sharp* M$›
proof(*induct rule*: *parCasesBrOutputFrame*)
case(*cPar1* Q' A_Q Ψ_Q A_R Ψ_R)
from ‹$A_{QR} \sharp* N$› ‹$A_{QR} = A_Q$ @ A_R› **have** $A_Q \sharp* N$ **and** $A_R \sharp* N$
by *simp*+
from ‹$A_{QR} \sharp* xvec$› ‹$A_{QR} = (A_Q @ A_R)$› **have** $A_Q \sharp* xvec$ **and** $A_R \sharp* xvec$
by *simp*+
have *Aeq*: $A_{QR} = A_Q @ A_R$ **and** Ψeq: $\Psi_{QR} = \Psi_Q \otimes \Psi_R$ **by** *fact*+
from *PTrans* Ψeq **have** $(\Psi \otimes \Psi_R) \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$
by(*metis statEqTransition Associativity Commutativity Composition*)
moreover note *FrP*
moreover from ‹$(\Psi \otimes \Psi_P) \otimes \Psi_R \rhd Q \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec Q'$› **have** $(\Psi \otimes \Psi_R) \otimes \Psi_P \rhd Q \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec Q'$

by(*metis statEqTransition Associativity Commutativity Composition*)
moreover note ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$›
moreover from ‹$A_P \sharp* Q$› ‹$A_P \sharp* R$› ‹$A_P \sharp* A_{QR}$› *Aeq* ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› ‹*extractFrame* $R = \langle A_R, \Psi_R\rangle$›
have $A_P \sharp* A_Q$ **and** $A_P \sharp* \Psi_R$ **by**(*auto dest*: *extractFrameFreshChain*)
moreover from ‹$A_{QR} \sharp* \Psi$› ‹$A_{QR} \sharp* P$› *Aeq* **have** $A_Q \sharp* \Psi$ **and** $A_Q \sharp* P$ **by** *simp+*
ultimately have $\Psi \otimes \Psi_R \rhd P \parallel Q \longmapsto ¡M(\!|\nu* xvec|\!)\langle N\rangle \prec (P' \parallel Q')$
using ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_P \sharp* Q$› ‹$A_P \sharp* M$› ‹$A_P \sharp* A_Q$› ‹$A_Q \sharp* \Psi$› ‹$A_Q \sharp* \Psi_R$› ‹$A_Q \sharp* P$› ‹$A_Q \sharp* Q$› ‹$A_Q \sharp* M$› ‹*xvec* $\sharp*$ P›
by(*intro BrComm1*) (*assumption* | *force*)+
moreover from ‹$A_{QR} \sharp* \Psi$› ‹$A_{QR} \sharp* P$› *Aeq* **have** $A_R \sharp* \Psi$ **and** $A_R \sharp* P$ **by** *simp+*
ultimately have $\Psi \rhd (P \parallel Q) \parallel R \longmapsto ¡M(\!|\nu* xvec|\!)\langle N\rangle \prec (P' \parallel Q') \parallel R$ **using** ‹*extractFrame* $R = \langle A_R, \Psi_R\rangle$› ‹$A_R \sharp* Q$› ‹$A_R \sharp* M$› ‹$A_R \sharp* N$› ‹*xvec* $\sharp*$ R› ‹$A_R \sharp* xvec$›
by(*intro Par1*) (*assumption* | *simp*)+
moreover have $(\Psi, (P' \parallel Q') \parallel R, (P' \parallel (Q' \parallel R))) \in Rel$
by(*rule C1*)
ultimately show *?case* **by** *blast*
next
case(*cPar2* R' A_Q Ψ_Q A_R Ψ_R)
from ‹$A_{QR} \sharp* N$› ‹$A_{QR} = A_Q \circledcirc A_R$› **have** $A_Q \sharp* N$ **and** $A_R \sharp* N$
by *simp+*
have *Aeq*: $A_{QR} = A_Q @ A_R$ **and** Ψeq: $\Psi_{QR} = \Psi_Q \otimes \Psi_R$ **by** *fact+*
from ‹$A_{QR} \sharp* \Psi$› ‹$A_{QR} \sharp* P$› ‹$A_{QR} \sharp* \Psi_P$› ‹$A_P \sharp* A_{QR}$› *Aeq*
have $A_R \sharp* \Psi$ **and** $A_R \sharp* \Psi_P$ **and** $A_P \sharp* A_R$ **and** $A_P \sharp* A_Q$ **and** $A_R \sharp* P$ **and** $A_Q \sharp* \Psi$ **and** $A_Q \sharp* \Psi_P$ **by** *simp+*
have *RTrans*: $(\Psi \otimes \Psi_P) \otimes \Psi_Q \rhd R \longmapsto ¡M(\!|\nu* xvec|\!)\langle N\rangle \prec R'$ **and** *FrR*: *extractFrame* $R = \langle A_R, \Psi_R\rangle$ **by** *fact+*
from *PTrans* Ψeq **have** $(\Psi \otimes \Psi_R) \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$
by(*metis statEqTransition Associativity Commutativity Composition*)
moreover from ‹$A_P \sharp* R$› ‹$A_P \sharp* Q$› ‹$A_P \sharp* A_{QR}$› *FrR* ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› *Aeq* **have** $A_P \sharp* \Psi_Q$ **and** $A_P \sharp* \Psi_R$
by(*auto dest*: *extractFrameFreshChain*)
moreover from ‹$A_{QR} \sharp* P$› ‹$A_{QR} \sharp* N$› *Aeq* **have** $A_Q \sharp* P$ **and** $A_Q \sharp* N$ **by** *simp+*
ultimately have $\Psi \otimes \Psi_R \rhd P \parallel Q \longmapsto ¿M(\!|N|\!) \prec (P' \parallel Q)$ **using** ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› ‹$A_Q \sharp* \Psi_R$› ‹$A_Q \sharp* M$› ‹$A_Q \sharp* \Psi$›
by(*intro Par1*) (*assumption* | *force*)+
moreover from *FrP* ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› ‹$A_P \sharp* A_Q$› ‹$A_P \sharp* \Psi_Q$› ‹$A_Q \sharp* \Psi_P$›
have *extractFrame*$(P \parallel Q) = \langle (A_P @ A_Q), \Psi_P \otimes \Psi_Q\rangle$ **by** *simp+*
moreover from *RTrans* **have** $\Psi \otimes (\Psi_P \otimes \Psi_Q) \rhd R \longmapsto ¡M(\!|\nu* xvec|\!)\langle N\rangle \prec R'$ **by**(*metis Associativity statEqTransition*)
moreover note *FrR*
ultimately have $\Psi \rhd (P \parallel Q) \parallel R \longmapsto ¡M(\!|\nu* xvec|\!)\langle N\rangle \prec ((P' \parallel Q) \parallel R')$
using ‹$A_P \sharp* \Psi$› ‹$A_Q \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_Q \sharp* P$› ‹$A_P \sharp* Q$› ‹$A_Q \sharp* Q$› ‹$A_P \sharp* R$› ‹$A_Q \sharp* R$› ‹$A_P \sharp* M$› ‹$A_Q \sharp* M$› ‹$A_P \sharp* A_R$› ‹$A_Q \sharp* A_R$›

$\langle A_R \sharp* \Psi\rangle$ $\langle A_R \sharp* P\rangle$ $\langle A_R \sharp* Q\rangle$ $\langle A_R \sharp* R\rangle$ $\langle A_R \sharp* M\rangle$ $\langle xvec \sharp* P\rangle$ $\langle xvec \sharp* Q\rangle$
by(*intro BrComm1*) (*assumption* | *simp*)+
moreover have $(\Psi, ((P' \parallel Q) \parallel R'), (P' \parallel (Q \parallel R'))) \in Rel$
by(*rule C1*)
ultimately show *?case* **by** *blast*
next
case(*cBrComm1* Ψ_R Q' A_Q Ψ_Q R' A_R)
from $\langle A_{QR} \sharp* N\rangle$ $\langle A_{QR} = A_Q \circledcirc A_R\rangle$ **have** $A_Q \sharp* N$ **and** $A_R \sharp* N$
by *simp*+
from $\langle A_{QR} \sharp* xvec\rangle$ $\langle A_{QR} = (A_Q @ A_R)\rangle$ **have** $A_Q \sharp* xvec$ **and** $A_R \sharp* xvec$
by *simp*+
have *Aeq*: $A_{QR} = A_Q @ A_R$ **and** Ψeq: $\Psi_{QR} = \Psi_Q \otimes \Psi_R$ **by** *fact*+
from *Aeq* $\langle A_{QR} \sharp* P\rangle$ **have** $A_Q \sharp* P$ **and** $A_R \sharp* P$ **by** *simp*+
from *Aeq* $\langle A_{QR} \sharp* \Psi\rangle$ **have** $A_Q \sharp* \Psi$ **and** $A_R \sharp* \Psi$ **by** *simp*+
from *Aeq* $\langle A_P \sharp* A_{QR}\rangle$ **have** $A_P \sharp* A_Q$ **and** $A_P \sharp* A_R$ **by** *simp*+
with $\langle A_P \sharp* Q\rangle$ $\langle extractFrame\ Q = \langle A_Q, \Psi_Q\rangle\rangle$ **have** $A_P \sharp* \Psi_Q$
by (*metis extractFrameFreshChain freshChainSym freshFrameDest*)
from *PTrans* Ψeq **have** $(\Psi \otimes \Psi_R) \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$
by(*metis statEqTransition Associativity Commutativity Composition*)
moreover note *FrP*
moreover from $\langle(\Psi \otimes \Psi_P) \otimes \Psi_R \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'\rangle$ **have** $(\Psi \otimes \Psi_R) \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$
by(*metis statEqTransition Associativity Commutativity Composition*)
moreover note $\langle extractFrame\ Q = \langle A_Q, \Psi_Q\rangle\rangle$
moreover from $\langle A_P \sharp* Q\rangle$ $\langle A_P \sharp* R\rangle$ $\langle A_P \sharp* A_{QR}\rangle$ *Aeq* $\langle extractFrame\ Q = \langle A_Q, \Psi_Q\rangle\rangle$ $\langle extractFrame\ R = \langle A_R, \Psi_R\rangle\rangle$
have $A_P \sharp* A_Q$ **and** $A_P \sharp* \Psi_R$ **by**(*auto dest*: *extractFrameFreshChain*)
moreover from $\langle A_{QR} \sharp* \Psi\rangle$ $\langle A_{QR} \sharp* P\rangle$ *Aeq* **have** $A_Q \sharp* \Psi$ **and** $A_Q \sharp* P$ **by** *simp*+
ultimately have *PQTrans*: $\Psi \otimes \Psi_R \rhd P \parallel Q \longmapsto ¿M(\!|N|\!) \prec (P' \parallel Q')$
using $\langle A_P \sharp* \Psi\rangle$ $\langle A_P \sharp* P\rangle$ $\langle A_P \sharp* Q\rangle$ $\langle A_P \sharp* M\rangle$ $\langle A_P \sharp* A_Q\rangle$ $\langle A_Q \sharp* \Psi\rangle$ $\langle A_Q \sharp* \Psi_R\rangle$ $\langle A_Q \sharp* P\rangle$ $\langle A_Q \sharp* Q\rangle$ $\langle A_Q \sharp* M\rangle$
by(*intro BrMerge*) (*assumption* | *force*)+
have *RTrans*: $(\Psi \otimes \Psi_P) \otimes \Psi_Q \rhd R \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec R'$ **and** *FrR*: $extractFrame\ R = \langle A_R, \Psi_R\rangle$ **by** *fact*+
note *PQTrans*
moreover from *FrP* $\langle extractFrame\ Q = \langle A_Q, \Psi_Q\rangle\rangle$ $\langle A_P \sharp* A_Q\rangle$ $\langle A_P \sharp* \Psi_Q\rangle$ $\langle A_{QR} \sharp* \Psi_P\rangle$ *Aeq*
have $extractFrame(P \parallel Q) = \langle(A_P @ A_Q), \Psi_P \otimes \Psi_Q\rangle$ **by** *simp*+
moreover from *RTrans* **have** $\Psi \otimes (\Psi_P \otimes \Psi_Q) \rhd R \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec R'$
by (*metis Associativity statEqTransition*)
moreover note *FrR*
moreover note $\langle A_P \sharp* \Psi\rangle$ $\langle A_Q \sharp* \Psi\rangle$ $\langle A_P \sharp* P\rangle$ $\langle A_P \sharp* Q\rangle$ $\langle A_Q \sharp* P\rangle$ $\langle A_Q \sharp* Q\rangle$ $\langle A_P \sharp* R\rangle$ $\langle A_Q \sharp* R\rangle$
$\langle A_P \sharp* M\rangle$ $\langle A_Q \sharp* M\rangle$ $\langle A_P \sharp* A_R\rangle$ $\langle A_Q \sharp* A_R\rangle$ $\langle A_R \sharp* \Psi\rangle$ $\langle A_R \sharp* P\rangle$ $\langle A_R \sharp* Q\rangle$ $\langle A_R \sharp* R\rangle$ $\langle A_R \sharp* M\rangle$ $\langle xvec \sharp* P\rangle$ $\langle xvec \sharp* Q\rangle$
ultimately have $\Psi \rhd (P \parallel Q) \parallel R \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec (P' \parallel Q') \parallel R'$

by(*intro BrComm1*) (*assumption* | *force*)+
moreover have $(\Psi, (P' \parallel Q') \parallel R', (P' \parallel (Q' \parallel R'))) \in Rel$
by(*rule C1*)
ultimately show *?case* **by** *blast*
next
case(*cBrComm2* Ψ_R Q' A_Q Ψ_Q R' A_R)
from ‹$A_{QR} \sharp* N$› ‹$A_{QR} = A_Q \mathbin{@} A_R$› **have** $A_Q \sharp* N$ **and** $A_R \sharp* N$
by *simp+*
from ‹$A_{QR} \sharp* xvec$› ‹$A_{QR} = (A_Q @ A_R)$› **have** $A_Q \sharp* xvec$ **and** $A_R \sharp* xvec$
by *simp+*
have *Aeq*: $A_{QR} = A_Q @ A_R$ **and** Ψeq: $\Psi_{QR} = \Psi_Q \otimes \Psi_R$ **by** *fact+*
from *Aeq* ‹$A_{QR} \sharp* P$› **have** $A_Q \sharp* P$ **and** $A_R \sharp* P$ **by** *simp+*
from *Aeq* ‹$A_{QR} \sharp* \Psi$› **have** $A_Q \sharp* \Psi$ **and** $A_R \sharp* \Psi$ **by** *simp+*
from *Aeq* ‹$A_P \sharp* A_{QR}$› **have** $A_P \sharp* A_Q$ **and** $A_P \sharp* A_R$ **by** *simp+*
with ‹$A_P \sharp* Q$› ‹*extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$› **have** $A_P \sharp* \Psi_Q$
by (*metis extractFrameFreshChain freshChainSym freshFrameDest*)
from *PTrans* Ψeq **have** $(\Psi \otimes \Psi_R) \otimes \Psi_Q \rhd P \longmapsto ¡M(\!|N|\!) \prec P'$
by(*metis statEqTransition Associativity Commutativity Composition*)
moreover note *FrP*
moreover from ‹$(\Psi \otimes \Psi_P) \otimes \Psi_R \rhd Q \longmapsto ¡M(\!|\nu * xvec|\!)\langle N \rangle \prec Q'$› **have** $(\Psi \otimes \Psi_R) \otimes \Psi_P \rhd Q \longmapsto ¡M(\!|\nu * xvec|\!)\langle N \rangle \prec Q'$
by(*metis statEqTransition Associativity Commutativity Composition*)
moreover note ‹*extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$›
moreover from ‹$A_P \sharp* Q$› ‹$A_P \sharp* R$› ‹$A_P \sharp* A_{QR}$› *Aeq* ‹*extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$› ‹*extractFrame* $R = \langle A_R, \Psi_R \rangle$›
have $A_P \sharp* A_Q$ **and** $A_P \sharp* \Psi_R$ **by**(*auto dest*: *extractFrameFreshChain*)
moreover from ‹$A_{QR} \sharp* \Psi$› ‹$A_{QR} \sharp* P$› *Aeq* **have** $A_Q \sharp* \Psi$ **and** $A_Q \sharp* P$ **by** *simp+*
ultimately have *PQTrans*: $\Psi \otimes \Psi_R \rhd P \parallel Q \longmapsto ¡M(\!|\nu * xvec|\!)\langle N \rangle \prec (P' \parallel Q')$
using ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_P \sharp* Q$› ‹$A_P \sharp* M$› ‹$A_P \sharp* A_Q$› ‹$A_Q \sharp* \Psi$› ‹$A_Q \sharp* \Psi_R$› ‹$A_Q \sharp* P$› ‹$A_Q \sharp* Q$› ‹$A_Q \sharp* M$› ‹$xvec \sharp* P$›
by(*intro BrComm1*) (*assumption* | *force*)+
have *RTrans*: $(\Psi \otimes \Psi_P) \otimes \Psi_Q \rhd R \longmapsto ¡M(\!|N|\!) \prec R'$ **and** *FrR*: *extractFrame* $R = \langle A_R, \Psi_R \rangle$ **by** *fact+*
note *PQTrans*
moreover from *FrP* ‹*extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$› ‹$A_P \sharp* A_Q$› ‹$A_P \sharp* \Psi_Q$› ‹$A_{QR} \sharp* \Psi_P$› *Aeq*
have *extractFrame*$(P \parallel Q) = \langle (A_P @ A_Q), \Psi_P \otimes \Psi_Q \rangle$ **by** *simp+*
moreover from *RTrans* **have** $\Psi \otimes (\Psi_P \otimes \Psi_Q) \rhd R \longmapsto ¡M(\!|N|\!) \prec R'$
by (*metis Associativity statEqTransition*)
moreover note *FrR*
moreover note ‹$A_P \sharp* \Psi$› ‹$A_Q \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_P \sharp* Q$› ‹$A_Q \sharp* P$› ‹$A_Q \sharp* Q$› ‹$A_P \sharp* R$› ‹$A_Q \sharp* R$›
‹$A_P \sharp* M$› ‹$A_Q \sharp* M$› ‹$A_P \sharp* A_R$› ‹$A_Q \sharp* A_R$› ‹$A_R \sharp* \Psi$› ‹$A_R \sharp* P$› ‹$A_R \sharp* Q$› ‹$A_R \sharp* R$› ‹$A_R \sharp* M$›
‹$xvec \sharp* P$› ‹$xvec \sharp* Q$› ‹$xvec \sharp* R$›
ultimately have $\Psi \rhd (P \parallel Q) \parallel R \longmapsto ¡M(\!|\nu * xvec|\!)\langle N \rangle \prec (P' \parallel Q') \parallel R'$
by(*auto intro*: *BrComm2*)

moreover have $(\Psi, (P' \parallel Q') \parallel R', (P' \parallel (Q' \parallel R'))) \in Rel$
by(*rule C1*)
ultimately show *?case* **by** *blast*
qed
next
case(*cBrComm2* Ψ_{QR} M *xvec* N P' A_P Ψ_P QR' A_{QR})
from ‹$¡M(\!|\nu * xvec|\!)\langle N\rangle = \alpha$› ‹*bn* α $\sharp*$ Q› ‹*bn* α $\sharp*$ R› **have** *xvec* $\sharp*$ Q **and** *xvec* $\sharp*$ R **by** *auto*
from ‹$A_{QR} \sharp* (\Psi, P, Q, R, \alpha)$› **have** $A_{QR} \sharp* Q$ **and** $A_{QR} \sharp* R$ **and** $A_{QR} \sharp* \Psi$ **by** *simp+*
from ‹$A_P \sharp* (Q \parallel R)$› **have** $A_P \sharp* Q$ **and** $A_P \sharp* R$ **by** *simp+*
have *PTrans*: $\Psi \otimes \Psi_{QR} \rhd P \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$ **and** *FrP*: *extractFrame* $P = \langle A_P, \Psi_P\rangle$ **by** *fact+*
note ‹$\Psi \otimes \Psi_P \rhd Q \parallel R \longmapsto ¿M(\!|N|\!) \prec QR'$› ‹*extractFrame*$(Q \parallel R) = \langle A_{QR}, \Psi_{QR}\rangle$› ‹*distinct* A_{QR}›
moreover from ‹$A_{QR} \sharp* \Psi$› ‹$A_{QR} \sharp* \Psi_P$› **have** $A_{QR} \sharp* (\Psi \otimes \Psi_P)$ **by** *force*
ultimately show *?case* **using** ‹$A_{QR} \sharp* Q$› ‹$A_{QR} \sharp* R$› ‹$A_{QR} \sharp* M$›
proof(*induct rule*: *parCasesBrInputFrame*)
case(*cPar1* Q' A_Q Ψ_Q A_R Ψ_R)
from ‹$A_{QR} \sharp* N$› ‹$A_{QR} = A_Q \mathbin{@} A_R$› **have** $A_Q \sharp* N$ **and** $A_R \sharp* N$
by *simp+*
from ‹$A_{QR} \sharp* xvec$› ‹$A_{QR} = (A_Q @ A_R)$› **have** $A_Q \sharp* xvec$ **and** $A_R \sharp* xvec$
by *simp+*
have *Aeq*: $A_{QR} = A_Q @ A_R$ **and** Ψeq: $\Psi_{QR} = \Psi_Q \otimes \Psi_R$ **by** *fact+*
from *PTrans* Ψeq **have** $(\Psi \otimes \Psi_R) \otimes \Psi_Q \rhd P \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$
by(*metis statEqTransition Associativity Commutativity Composition*)
moreover note *FrP*
moreover from ‹$(\Psi \otimes \Psi_P) \otimes \Psi_R \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$› **have** $(\Psi \otimes \Psi_R) \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$
by(*metis statEqTransition Associativity Commutativity Composition*)
moreover note ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$›
moreover from ‹$A_P \sharp* Q$› ‹$A_P \sharp* R$› ‹$A_P \sharp* A_{QR}$› *Aeq* ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› ‹*extractFrame* $R = \langle A_R, \Psi_R\rangle$›
have $A_P \sharp* A_Q$ **and** $A_P \sharp* \Psi_R$ **by**(*auto dest*: *extractFrameFreshChain*)
moreover from ‹$A_{QR} \sharp* \Psi$› ‹$A_{QR} \sharp* P$› *Aeq* **have** $A_Q \sharp* \Psi$ **and** $A_Q \sharp* P$ **by** *simp+*
ultimately have $\Psi \otimes \Psi_R \rhd P \parallel Q \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec (P' \parallel Q')$
using ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_P \sharp* Q$› ‹$A_P \sharp* M$› ‹$A_P \sharp* A_Q$› ‹$A_Q \sharp* \Psi$› ‹$A_Q \sharp* \Psi_R$› ‹$A_Q \sharp* P$› ‹$A_Q \sharp* Q$› ‹$A_Q \sharp* M$› ‹*xvec* $\sharp*$ Q›
by(*intro BrComm2*) (*assumption* | *force*)+
moreover from ‹$A_{QR} \sharp* \Psi$› ‹$A_{QR} \sharp* P$› *Aeq* **have** $A_R \sharp* \Psi$ **and** $A_R \sharp* P$ **by** *simp+*
ultimately have $\Psi \rhd (P \parallel Q) \parallel R \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec (P' \parallel Q') \parallel R$ **using** ‹*extractFrame* $R = \langle A_R, \Psi_R\rangle$› ‹$A_R \sharp* Q$› ‹$A_R \sharp* M$› ‹$A_R \sharp* N$› ‹*xvec* $\sharp*$ R› ‹$A_R \sharp* xvec$›
by(*intro Par1*) (*assumption* | *simp*)+
moreover have $(\Psi, (P' \parallel Q') \parallel R, (P' \parallel (Q' \parallel R))) \in Rel$
by(*rule C1*)
ultimately show *?case* **by** *blast*

next
case(*cPar2* R' A_Q Ψ_Q A_R Ψ_R)
from ‹A_{QR} ♯* N› ‹A_{QR} ♯* $xvec$› ‹$A_{QR} = A_Q$ @ A_R› **have** A_Q ♯* N **and** A_R ♯* N **and** A_Q ♯* $xvec$ **and** A_R ♯* $xvec$
by *simp+*
have *Aeq*: $A_{QR} = A_Q @ A_R$ **and** Ψeq: $\Psi_{QR} = \Psi_Q \otimes \Psi_R$ **by** *fact+*
from ‹A_{QR} ♯* Ψ› ‹A_{QR} ♯* P› ‹A_{QR} ♯* Ψ_P› ‹A_P ♯* A_{QR}› *Aeq*
have A_R ♯* Ψ **and** A_R ♯* Ψ_P **and** A_P ♯* A_R **and** A_P ♯* A_Q **and** A_R ♯* P **and** A_Q ♯* Ψ **and** A_Q ♯* Ψ_P **by** *simp+*
have *RTrans*: $(\Psi \otimes \Psi_P) \otimes \Psi_Q \rhd R \longmapsto ¿M(\!|N|\!) \prec R'$ **and** *FrR*: *extractFrame* $R = \langle A_R, \Psi_R\rangle$ **by** *fact+*
from *PTrans* Ψeq **have** $(\Psi \otimes \Psi_R) \otimes \Psi_Q \rhd P \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec P'$
by(*metis statEqTransition Associativity Commutativity Composition*)
moreover from ‹A_P ♯* R› ‹A_P ♯* Q› ‹A_P ♯* A_{QR}› *FrR* ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› *Aeq* **have** A_P ♯* Ψ_Q **and** A_P ♯* Ψ_R
by(*auto dest*: *extractFrameFreshChain*)
moreover from ‹A_{QR} ♯* P› ‹A_{QR} ♯* N› *Aeq* **have** A_Q ♯* P **and** A_Q ♯* N **by** *simp+*
ultimately have $\Psi \otimes \Psi_R \rhd P \parallel Q \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec (P' \parallel Q)$ **using** ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› ‹$A_Q$ ♯* Ψ_R›
‹A_Q ♯* M› ‹A_Q ♯* Ψ› ‹*xvec* ♯* Q› ‹A_Q ♯* *xvec*›
by(*intro Par1*) (*assumption* | *force*)+
moreover from *FrP* ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› ‹$A_P$ ♯* A_Q› ‹A_P ♯* Ψ_Q› ‹A_Q ♯* Ψ_P›
have *extractFrame*$(P \parallel Q) = \langle (A_P @ A_Q), \Psi_P \otimes \Psi_Q\rangle$ **by** *simp+*
moreover from *RTrans* **have** $\Psi \otimes (\Psi_P \otimes \Psi_Q) \rhd R \longmapsto ¿M(\!|N|\!) \prec R'$ **by**(*metis Associativity statEqTransition*)
moreover note *FrR*
ultimately have $\Psi \rhd (P \parallel Q) \parallel R \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec ((P' \parallel Q) \parallel R')$
using ‹A_P ♯* Ψ› ‹A_Q ♯* Ψ› ‹A_P ♯* P› ‹A_Q ♯* P› ‹A_P ♯* Q› ‹A_Q ♯* Q› ‹A_P ♯* R› ‹A_Q ♯* R› ‹A_P ♯* M› ‹A_Q ♯* M› ‹A_P ♯* A_R› ‹A_Q ♯* A_R›
‹A_R ♯* Ψ› ‹A_R ♯* P› ‹A_R ♯* Q› ‹A_R ♯* R› ‹A_R ♯* M› ‹*xvec* ♯* P› ‹*xvec* ♯* R›
by(*auto intro*: *BrComm2*)
moreover have $(\Psi, ((P' \parallel Q) \parallel R'), (P' \parallel (Q \parallel R'))) \in Rel$
by(*rule C1*)
ultimately show *?case* **by** *blast*
next
case(*cBrMerge* Ψ_R Q' A_Q Ψ_Q R' A_R)
from ‹A_{QR} ♯* N› ‹$A_{QR} = A_Q$ @ A_R› **have** A_Q ♯* N **and** A_R ♯* N
by *simp+*
from ‹A_{QR} ♯* *xvec*› ‹$A_{QR} = (A_Q @ A_R)$› **have** A_Q ♯* *xvec* **and** A_R ♯* *xvec*
by *simp+*
have *Aeq*: $A_{QR} = A_Q @ A_R$ **and** Ψeq: $\Psi_{QR} = \Psi_Q \otimes \Psi_R$ **by** *fact+*
from *Aeq* ‹A_{QR} ♯* P› **have** A_Q ♯* P **and** A_R ♯* P **by** *simp+*
from *Aeq* ‹A_{QR} ♯* M› **have** A_Q ♯* M **and** A_R ♯* M **by** *simp+*
from *Aeq* ‹A_{QR} ♯* Ψ› **have** A_Q ♯* Ψ **and** A_R ♯* Ψ **by** *simp+*
from *Aeq* ‹A_P ♯* A_{QR}› **have** A_P ♯* A_Q **and** A_P ♯* A_R **by** *simp+*
with ‹A_P ♯* Q› ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› **have** A_P ♯* Ψ_Q

by (*metis extractFrameFreshChain freshChainSym freshFrameDest*)
from *PTrans* Ψ*eq* **have** $(\Psi \otimes \Psi_R) \otimes \Psi_Q \rhd P \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$
by(*metis statEqTransition Associativity Commutativity Composition*)
moreover note *FrP*
moreover from $\langle(\Psi \otimes \Psi_P) \otimes \Psi_R \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'\rangle$ **have** $(\Psi \otimes \Psi_R) \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$
by(*metis statEqTransition Associativity Commutativity Composition*)
moreover note $\langle$*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle\rangle$
moreover from $\langle A_P \sharp* Q\rangle$ $\langle A_P \sharp* R\rangle$ $\langle A_P \sharp* A_{QR}\rangle$ *Aeq* $\langle$*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle\rangle$ $\langle$*extractFrame* $R = \langle A_R, \Psi_R\rangle\rangle$
have $A_P \sharp* A_Q$ **and** $A_P \sharp* \Psi_R$ **by**(*auto dest*: *extractFrameFreshChain*)
moreover from $\langle A_{QR} \sharp* \Psi\rangle$ $\langle A_{QR} \sharp* P\rangle$ *Aeq* **have** $A_Q \sharp* \Psi$ **and** $A_Q \sharp* P$ **by** *simp+*
ultimately have *PQTrans*: $\Psi \otimes \Psi_R \rhd P \parallel Q \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec (P' \parallel Q')$
using $\langle A_P \sharp* \Psi\rangle$ $\langle A_P \sharp* P\rangle$ $\langle A_P \sharp* Q\rangle$ $\langle A_P \sharp* M\rangle$ $\langle A_P \sharp* A_Q\rangle$ $\langle A_Q \sharp* \Psi\rangle$ $\langle A_Q \sharp* \Psi_R\rangle$ $\langle A_Q \sharp* P\rangle$ $\langle A_Q \sharp* Q\rangle$ $\langle xvec \sharp* Q\rangle$ $\langle A_Q \sharp* M\rangle$
by(*intro BrComm2*) (*assumption* | *force*)+
have *RTrans*: $(\Psi \otimes \Psi_P) \otimes \Psi_Q \rhd R \longmapsto ¿M(\!|N|\!) \prec R'$ **and** *FrR*: *extractFrame* $R = \langle A_R, \Psi_R\rangle$ **by** *fact+*
note *PQTrans*
moreover from *FrP* $\langle$*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle\rangle$ $\langle A_P \sharp* A_Q\rangle$ $\langle A_P \sharp* \Psi_Q\rangle$ $\langle A_{QR} \sharp* \Psi_P\rangle$ *Aeq*
have *extractFrame*$(P \parallel Q) = \langle (A_P @ A_Q), \Psi_P \otimes \Psi_Q\rangle$ **by** *simp+*
moreover from *RTrans* **have** $\Psi \otimes (\Psi_P \otimes \Psi_Q) \rhd R \longmapsto ¿M(\!|N|\!) \prec R'$
by (*metis Associativity statEqTransition*)
moreover note *FrR*
moreover note $\langle A_P \sharp* \Psi\rangle$ $\langle A_Q \sharp* \Psi\rangle$ $\langle A_P \sharp* P\rangle$ $\langle A_P \sharp* Q\rangle$ $\langle A_Q \sharp* P\rangle$ $\langle A_Q \sharp* Q\rangle$ $\langle A_P \sharp* R\rangle$ $\langle A_Q \sharp* R\rangle$
$\langle A_P \sharp* M\rangle$ $\langle A_Q \sharp* M\rangle$ $\langle A_P \sharp* A_R\rangle$ $\langle A_Q \sharp* A_R\rangle$ $\langle A_R \sharp* \Psi\rangle$ $\langle A_R \sharp* P\rangle$ $\langle A_R \sharp* Q\rangle$ $\langle A_R \sharp* R\rangle$ $\langle A_R \sharp* M\rangle$ $\langle xvec \sharp* P\rangle$ $\langle xvec \sharp* R\rangle$
ultimately have $\Psi \rhd (P \parallel Q) \parallel R \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec (P' \parallel Q') \parallel R'$
by(*auto intro*: *BrComm2*)
moreover have $(\Psi, (P' \parallel Q') \parallel R', (P' \parallel (Q' \parallel R'))) \in Rel$
by(*rule C1*)
ultimately show *?case* **by** *blast*
qed
qed
qed

lemma *parNilLeft*:
fixes $\Psi :: 'b$
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 $\Psi \rhd (P \parallel \mathbf{0}) \rightsquigarrow[Rel]\ P$

```
  using ‹eqvt Rel›
proof(induct rule: simI[of - - - - ()])
  case(cSim α P')
  from ‹Ψ ▷ P ⟼α ≺ P'› have Ψ ⊗ SBottom' ▷ P ⟼α ≺ P'
    by(metis statEqTransition Identity AssertionStatEqSym)
  then have Ψ ▷ (P ∥ 𝟬) ⟼α ≺ (P' ∥ 𝟬)
    by(rule Par1) auto
  moreover have (Ψ, P' ∥ 𝟬, P') ∈ Rel by(rule C1)
  ultimately show ?case by blast
qed

lemma parNilRight:
  fixes Ψ :: 'b
    and P   :: ('a, 'b, 'c) psi
    and Rel :: ('b × ('a, 'b, 'c) psi × ('a, 'b, 'c) psi) set

assumes eqvt Rel
  and   C1: ⋀Q. (Ψ, Q, (Q ∥ 𝟬)) ∈ Rel

shows Ψ ▷ P ⇝[Rel] (P ∥ 𝟬)
  using ‹eqvt Rel›
proof(induct rule: simI[of - - - - ()])
  case(cSim α P')
  note ‹Ψ ▷ P ∥ 𝟬 ⟼α ≺ P'› ‹bn α ♯* Ψ› ‹bn α ♯* P›
  moreover have bn α ♯* 𝟬 by simp
  ultimately show ?case using ‹bn α ♯* subject α›
  proof(induct rule: parCases[where C=()])
    case(cPar1 P' A_Q Ψ_Q)
    from ‹extractFrame(𝟬) = ⟨A_Q, Ψ_Q⟩› have Ψ_Q = SBottom' by auto
    with ‹Ψ ⊗ Ψ_Q ▷ P ⟼α ≺ P'› have Ψ ▷ P ⟼α ≺ P'
      by(metis statEqTransition Identity)
    moreover have (Ψ, P', P' ∥ 𝟬) ∈ Rel by(rule C1)
    ultimately show ?case by blast
  next
    case(cPar2 Q' A_P Ψ_P)
    from ‹Ψ ⊗ Ψ_P ▷ 𝟬 ⟼α ≺ Q'› have False
      by auto
    then show ?case by simp
  next
    case(cComm1 Ψ_Q M N P' A_P Ψ_P K xvec Q' A_Q)
    from ‹Ψ ⊗ Ψ_P ▷ 𝟬 ⟼K⟨νxvec⟩⟨N⟩ ≺ Q'› have False by auto
    then show ?case by simp
  next
    case(cComm2 Ψ_Q M xvec N P' A_P Ψ_P K Q' A_Q)
    from ‹Ψ ⊗ Ψ_P ▷ 𝟬 ⟼K(⟨N⟩) ≺ Q'› have False
      by auto
    then show ?case by simp
  next
    case(cBrMerge Ψ_Q M N P' A_P Ψ_P Q' A_Q)
```

```
    from ‹Ψ ⊗ Ψ_P ▷ 0 ⟼ ¿M(⦇N⦈) ≺ Q'› have False
      by auto
    then show ?case by simp
  next
    case(cBrComm1 Ψ_Q M N P' A_P Ψ_P xvec Q' A_Q)
    from ‹Ψ ⊗ Ψ_P ▷ 0 ⟼ ¡M(⦇ν*xvec⦈)⟨N⟩ ≺ Q'› have False
      by auto
    then show ?case by simp
  next
    case(cBrComm2 Ψ_Q M xvec N P' A_P Ψ_P Q' A_Q)
    from ‹Ψ ⊗ Ψ_P ▷ 0 ⟼ ¿M(⦇N⦈) ≺ Q'› have False
      by auto
    then show ?case by simp
  qed
qed

lemma resNilLeft:
  fixes x   :: name
    and Ψ   :: 'b
    and Rel :: ('b × ('a, 'b, 'c) psi × ('a, 'b, 'c) psi) set

shows Ψ ▷ (⦇νx⦈)0 ⇝[Rel] 0
  by(auto simp add: simulation-def)

lemma resNilRight:
  fixes x   :: name
    and Ψ   :: 'b
    and Rel :: ('b × ('a, 'b, 'c) psi × ('a, 'b, 'c) psi) set

shows Ψ ▷ 0 ⇝[Rel] (⦇νx⦈)0
  apply(clarsimp simp add: simulation-def)
  by(cases rule: semantics.cases) (auto simp add: psi.inject alpha')

lemma inputPushResLeft:
  fixes Ψ   :: '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 ♯ Ψ
  and   x ♯ M
  and   x ♯ xvec
  and   x ♯ N
  and   C1: ⋀Q. (Ψ, Q, Q) ∈ Rel

shows Ψ ▷ (⦇νx⦈)(M(⦇λ*xvec N⦈).P) ⇝[Rel] M(⦇λ*xvec N⦈).(⦇νx⦈)P
```

```
proof −
  note ‹eqvt Rel› ‹x ♯ Ψ›
  moreover have x ♯ (⦇νx⦈)(M⦇λ*xvec N⦈.P) by(simp add: abs-fresh)
  moreover from ‹x ♯ M› ‹x ♯ N› have x ♯ M⦇λ*xvec N⦈.(⦇νx⦈)P
    by(auto simp add: inputChainFresh abs-fresh)
  ultimately show ?thesis
  proof(induct rule: simIFresh[of - - - - - ()])
    case(cSim α P')
    from ‹Ψ ▷ M⦇λ*xvec N⦈.(⦇νx⦈)P ⟼α ≺ P'› ‹x ♯ α› show ?case
    proof(induct rule: inputCases)
      case(cInput K Tvec)
     from ‹x ♯ K⦇N[xvec::=Tvec]⦈› have x ♯ K and x ♯ N[xvec::=Tvec] by simp+
     from ‹Ψ ⊢ M ↔ K› ‹distinct xvec› ‹set xvec ⊆ supp N› ‹length xvec = length
Tvec›
      have Ψ ▷ M⦇λ*xvec N⦈.P ⟼K⦇(N[xvec::=Tvec])⦈ ≺ P[xvec::=Tvec]
        by(rule Input)
    then have Ψ ▷ (⦇νx⦈)(M⦇λ*xvec N⦈.P) ⟼K⦇(N[xvec::=Tvec])⦈ ≺ (⦇νx⦈)(P[xvec::=Tvec])
using ‹x ♯ Ψ› ‹x ♯ K› ‹x ♯ N[xvec::=Tvec]›
        by(intro Scope) auto
      moreover from ‹length xvec = length Tvec› ‹distinct xvec› ‹set xvec ⊆ supp
N› ‹x ♯ N[xvec::=Tvec]› have x ♯ Tvec
        by(rule substTerm.subst3)
      with ‹x ♯ xvec› have (Ψ, (⦇νx⦈)(P[xvec::=Tvec]), ((⦇νx⦈)P)[xvec::=Tvec]) ∈ Rel
        by(force intro: C1)
      ultimately show ?case by blast
    next
      case(cBrInput K Tvec)
     from ‹x ♯ ¿K⦇N[xvec::=Tvec]⦈› have x ♯ K and x ♯ N[xvec::=Tvec] by simp+
     from ‹Ψ ⊢ K ≻ M› ‹distinct xvec› ‹set xvec ⊆ supp N› ‹length xvec = length
Tvec›
      have Ψ ▷ M⦇λ*xvec N⦈.P ⟼¿K⦇(N[xvec::=Tvec])⦈ ≺ P[xvec::=Tvec]
        by(rule BrInput)
    then have Ψ ▷ (⦇νx⦈)(M⦇λ*xvec N⦈.P) ⟼¿K⦇(N[xvec::=Tvec])⦈ ≺ (⦇νx⦈)(P[xvec::=Tvec])
using ‹x ♯ Ψ› ‹x ♯ K› ‹x ♯ N[xvec::=Tvec]›
        by(intro Scope) auto
      moreover from ‹length xvec = length Tvec› ‹distinct xvec› ‹set xvec ⊆ supp
N› ‹x ♯ N[xvec::=Tvec]› have x ♯ Tvec
        by(rule substTerm.subst3)
      with ‹x ♯ xvec› have (Ψ, (⦇νx⦈)(P[xvec::=Tvec]), ((⦇νx⦈)P)[xvec::=Tvec]) ∈ Rel
        by(force intro: C1)
      ultimately show ?case by blast
    qed
  qed
qed

lemma inputPushResRight:
  fixes Ψ   :: '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 −
note ‹*eqvt Rel*› ‹$x \sharp \Psi$›
moreover from ‹$x \sharp M$› ‹$x \sharp N$› **have** $x \sharp M(\!|\lambda * xvec\ N|\!).(|\nu x|)P$
by(*auto simp add*: *inputChainFresh abs-fresh*)
moreover have $x \sharp (|\nu x|)(M(\!|\lambda * xvec\ N|\!).P)$ **by**(*simp add*: *abs-fresh*)
ultimately show *?thesis*
proof(*induct rule*: *simIFresh*[*of* - - - - - ()])
case(*cSim* α P')
note ‹$\Psi \rhd (|\nu x|)(M(\!|\lambda * xvec\ N|\!).P) \longmapsto \alpha \prec P'$› ‹$x \sharp \Psi$› ‹$x \sharp \alpha$› ‹$x \sharp P'$› ‹*bn* α $\sharp*$ Ψ›
moreover from ‹*bn* α $\sharp*$ $((|\nu x|)(M(\!|\lambda * xvec\ N|\!).P))$› ‹$x \sharp \alpha$› **have** *bn* α $\sharp*$ $(M(\!|\lambda * xvec\ N|\!).P)$
by *simp*
ultimately show *?case* **using** ‹*bn* α $\sharp*$ *subject* α›
proof(*induct rule*: *resCases*[**where** *C*=()])
case(*cRes* P')
from ‹$\Psi \rhd M(\!|\lambda * xvec\ N|\!).P \longmapsto \alpha \prec P'$› ‹$x \sharp \alpha$› **show** *?case*
proof(*induct rule*: *inputCases*)
case(*cInput K Tvec*)
from ‹$x \sharp K(\!|N[xvec::=Tvec]|\!)$› **have** $x \sharp K$ **and** $x \sharp N[xvec::=Tvec]$ **by** *simp*+
from ‹$\Psi \vdash M \leftrightarrow K$› ‹*distinct xvec*› ‹*set xvec* ⊆ *supp N*› ‹*length xvec* = *length Tvec*›
have $\Psi \rhd M(\!|\lambda * xvec\ N|\!).((|\nu x|)P) \longmapsto K(\!|(N[xvec::=Tvec])|\!) \prec ((|\nu x|)P)[xvec::=Tvec]$
by(*rule Input*)
moreover from ‹*length xvec* = *length Tvec*› ‹*distinct xvec*› ‹*set xvec* ⊆ *supp N*› ‹$x \sharp N[xvec::=Tvec]$› **have** $x \sharp Tvec$
by(*rule substTerm.subst3*)
with ‹$x \sharp xvec$› **have** $(\Psi,\ ((|\nu x|)P)[xvec::=Tvec],\ (|\nu x|)(P[xvec::=Tvec])) \in Rel$
by(*force intro*: *C1*)
ultimately show *?case* **by** *blast*
next
case(*cBrInput K Tvec*)
from ‹$x \sharp ¿K(\!|N[xvec::=Tvec]|\!)$› **have** $x \sharp K$ **and** $x \sharp N[xvec::=Tvec]$ **by** *simp*+
from ‹$\Psi \vdash K \succeq M$› ‹*distinct xvec*› ‹*set xvec* ⊆ *supp N*› ‹*length xvec* =

```
length Tvec›
      have Ψ ▷ M⦇λ*xvec N⦈.(⦇νx⦈P) ⟼¿K⦇(N[xvec::=Tvec])⦈ ≺ (⦇νx⦈P)[xvec::=Tvec]
          by(rule BrInput)
        moreover from ‹length xvec = length Tvec› ‹distinct xvec› ‹set xvec ⊆ supp
N› ‹x ♯ N[xvec::=Tvec]› have x ♯ Tvec
          by(rule substTerm.subst3)
         with ‹x ♯ xvec› have (Ψ, (⦇νx⦈P)[xvec::=Tvec], ⦇νx⦈(P[xvec::=Tvec])) ∈
Rel
          by(force intro: C1)
        ultimately show ?case by blast
       qed
     next
       case cOpen
       then have False by auto
       then show ?case by simp
     next
       case cBrOpen
       then have False by auto
       then show ?case by simp
     next
       case (cBrClose M xvec N P′)
       then have False by auto
       then show ?case by simp
     qed
   qed
qed

lemma outputPushResLeft:
  fixes Ψ   :: 'b
    and x   :: name
    and M   :: 'a
    and N   :: 'a
    and P   :: ('a, 'b, 'c) psi

assumes eqvt Rel
  and   x ♯ Ψ
  and   x ♯ M
  and   x ♯ N
  and   C1: ⋀Q. (Ψ, Q, Q) ∈ Rel

shows Ψ ▷ ⦇νx⦈(M⟨N⟩.P) ⇝[Rel] M⟨N⟩.⦇νx⦈P
proof −
  note ‹eqvt Rel› ‹x ♯ Ψ›
  moreover have x ♯ ⦇νx⦈(M⟨N⟩.P) by(simp add: abs-fresh)
  moreover from ‹x ♯ M› ‹x ♯ N› have x ♯ M⟨N⟩.⦇νx⦈P
    by(auto simp add: abs-fresh)
  ultimately show ?thesis
  proof(induct rule: simIFresh[of - - - - - ()])
    case(cSim α P′)
```

```
    from ‹Ψ ▷ M⟨N⟩.(⦇νx⦈)P ⟼α ≺ P'› ‹x ♯ α›
    show ?case
    proof(induct rule: outputCases)
      case(cOutput K)
      from ‹Ψ ⊢ M ↔ K› have Ψ ▷ M⟨N⟩.P ⟼K⟨N⟩ ≺ P
        by(rule Output)
      then have Ψ ▷ (⦇νx⦈)(M⟨N⟩.P) ⟼K⟨N⟩ ≺ (⦇νx⦈)P using ‹x ♯ Ψ› ‹x ♯ K⟨N⟩›
        by(rule Scope)
      moreover have (Ψ, (⦇νx⦈)P, (⦇νx⦈)P) ∈ Rel by(rule C1)
      ultimately show ?case by blast
    next
      case(cBrOutput K)
      from ‹Ψ ⊢ M ≍ K› have Ψ ▷ M⟨N⟩.P ⟼¡K⟨N⟩ ≺ P
        by(rule BrOutput)
        then have Ψ ▷ (⦇νx⦈)(M⟨N⟩.P) ⟼¡K⟨N⟩ ≺ (⦇νx⦈)P using ‹x ♯ Ψ› ‹x ♯
¡K⟨N⟩›
        by(rule Scope)
      moreover have (Ψ, (⦇νx⦈)P, (⦇νx⦈)P) ∈ Rel by(rule C1)
      ultimately show ?case by blast
    qed
  qed
qed

lemma broutputNoBind:
  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 Ψ ▷ M⟨N⟩.P ⟼(¡K(⦇ν*xvec⦈)⟨N'⟩) ≺ P'
shows xvec = []
proof −
  from assms have bn(¡K(⦇ν*xvec⦈)⟨N'⟩) = []
    by(induct rule: outputCases) auto
  then show ?thesis by simp
qed

lemma broutputObjectEq:
  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 Ψ ▷ M⟨N⟩.P ⟼(¡K(⦇ν*xvec⦈)⟨N'⟩) ≺ P'
shows N = N'
```

proof −
  from *assms* **have** *object*(¡K(⦇ν∗*xvec*⦈)⟨N'⟩) = *Some* N
    by(*induct rule*: *outputCases*) *auto*
  then show *?thesis*
    by *simp*
qed

lemma *brOutputOutputCases*[*consumes 1*, *case-names cBrOutput*]:
  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 *Trans*: Ψ ▷ M⟨N⟩.P ⟼(¡K(⦇ν∗*xvec*⦈)⟨N'⟩) ≺ P'
  and *rBrOutput*: ⋀K. Ψ ⊢ M ≼ K ⟹ *Prop* (¡K⟨N⟩) P

shows *Prop* (¡K(⦇ν∗*xvec*⦈)⟨N'⟩) P'
proof −
  have *xvec* = [] **using** *Trans* **by**(*rule broutputNoBind*)
  then obtain K' N'' **where** *eq*: (¡K(⦇ν∗*xvec*⦈)⟨N'⟩)= ¡K'⟨N''⟩
    by *blast*
  have $N = N'$ **using** *Trans* **by**(*rule broutputObjectEq*)
  from *Trans* ‹(¡K(⦇ν∗*xvec*⦈)⟨N'⟩)= ¡K'⟨N''⟩›
  show *?thesis* **unfolding** ‹$N = N'$›[*symmetric*]
  proof(*induct rule*: *outputCases*)
    case (*cOutput* K) **then show** *?case* **by** *simp*
  next
    case (*cBrOutput* K) **then show** *?case*
      by(*intro rBrOutput*)
  qed
qed

lemma *outputPushResRight*:
  fixes Ψ :: $'b$
    and x :: *name*
    and M :: $'a$
    and N :: $'a$
    and P :: ($'a$, $'b$, $'c$) *psi*

assumes *eqvt Rel*
  and x ♯ Ψ
  and x ♯ M
  and x ♯ N
  and *C1*: ⋀Q. (Ψ, Q, Q) ∈ *Rel*

shows Ψ ▷ M⟨N⟩.(⦇νx⦈)P ⇝[*Rel*] (⦇νx⦈)(M⟨N⟩.P)
proof −

note ‹*eqvt Rel*› ‹$x \sharp \Psi$›
moreover from ‹$x \sharp M$› ‹$x \sharp N$› **have** $x \sharp M\langle N\rangle.(\!|\nu x|\!)P$
by(*auto simp add*: *abs-fresh*)
moreover have $x \sharp (\!|\nu x|\!)(M\langle N\rangle.P)$ **by**(*simp add*: *abs-fresh*)
ultimately show *?thesis*
proof(*induct rule*: *simIFresh*[*of* - - - - - (M, N)])
case(*cSim* α P')
note ‹$\Psi \rhd (\!|\nu x|\!)(M\langle N\rangle.P) \longmapsto \alpha \prec P'$› ‹$x \sharp \Psi$› ‹$x \sharp \alpha$› ‹$x \sharp P'$› ‹*bn* $\alpha \sharp* \Psi$›
moreover from ‹*bn* $\alpha \sharp* ((\!|\nu x|\!)(M\langle N\rangle.P))$› ‹$x \sharp \alpha$› **have** *bn* $\alpha \sharp* (M\langle N\rangle.P)$ **by** *simp*
ultimately show *?case* **using** ‹*bn* $\alpha \sharp*$ *subject* α› ‹*bn* $\alpha \sharp* (M, N)$› ‹$x \sharp \alpha$› ‹$x \sharp M$› ‹$x \sharp N$›
proof(*induct rule*: *resCases*[**where** C=()])
case(*cOpen K xvec1 xvec2 y N' P'*)
from ‹*bn*$(K(\!|\nu*(xvec1@y\#xvec2)|\!)\langle N'\rangle) \sharp* (M, N)$› **have** $y \sharp N$ **by** *simp+*
from ‹$\Psi \rhd M\langle N\rangle.P \longmapsto K(\!|\nu*(xvec1@xvec2)|\!)\langle([(x, y)] \cdot N')\rangle \prec ([(x, y)] \cdot P')$›
have $N = ([(x, y)] \cdot N')$
apply −
by(*ind-cases* $\Psi \rhd M\langle N\rangle.P \longmapsto K(\!|\nu*(xvec1@xvec2)|\!)\langle([(x, y)] \cdot N')\rangle \prec ([(x, y)] \cdot P')$)
(*auto simp add*: *residualInject psi.inject*)
with ‹$x \sharp N$› ‹$y \sharp N$› ‹$x \neq y$› **have** $N = N'$
by(*subst pt-bij*[*OF pt-name-inst*, *OF at-name-inst*, *symmetric*, **where** *pi*=[(x, y)]])
(*simp add*: *fresh-left calc-atm*)
with ‹$y \in$ *supp* N'› ‹$y \sharp N$› **have** *False* **by**(*simp add*: *fresh-def*)
then show *?case* **by** *simp*
next
case(*cBrOpen K xvec1 xvec2 y N' P'*)
from ‹*bn*$(¡K(\!|\nu*(xvec1@y\#xvec2)|\!)\langle N'\rangle) \sharp* (M, N)$› **have** $y \sharp N$ **by** *simp+*
from ‹$\Psi \rhd M\langle N\rangle.P \longmapsto ¡K(\!|\nu*(xvec1@xvec2)|\!)\langle([(x, y)] \cdot N')\rangle \prec ([(x, y)] \cdot P')$›
have $N = ([(x, y)] \cdot N')$
apply −
by(*ind-cases* $\Psi \rhd M\langle N\rangle.P \longmapsto ¡K(\!|\nu*(xvec1@xvec2)|\!)\langle([(x, y)] \cdot N')\rangle \prec ([(x, y)] \cdot P')$)
(*auto simp add*: *residualInject psi.inject*)
with ‹$x \sharp N$› ‹$y \sharp N$› ‹$x \neq y$› **have** $N = N'$
by(*subst pt-bij*[*OF pt-name-inst*, *OF at-name-inst*, *symmetric*, **where** *pi*=[(x, y)]])
(*simp add*: *fresh-left calc-atm*)
with ‹$y \in$ *supp* N'› ‹$y \sharp N$› **have** *False* **by**(*simp add*: *fresh-def*)
then show *?case* **by** *simp*
next
case(*cRes P'*)
from ‹$\Psi \rhd M\langle N\rangle.P \longmapsto \alpha \prec P'$› **show** *?case*
proof(*induct rule*: *outputCases*)
case(*cOutput K*)

```
      from ‹Ψ ⊢ M ↔ K› have Ψ ▷ M⟨N⟩.(⦇νx⦈P) ⟼K⟨N⟩ ≺ ⦇νx⦈P
        by(rule Output)
      moreover have (Ψ, ⦇νx⦈P, ⦇νx⦈P) ∈ Rel by(rule C1)
      ultimately show ?case by force
    next
      case(cBrOutput K)
      from ‹Ψ ⊢ M ≼ K› have Ψ ▷ M⟨N⟩.(⦇νx⦈P) ⟼¡K⟨N⟩ ≺ ⦇νx⦈P
        by(rule BrOutput)
      moreover have (Ψ, ⦇νx⦈P, ⦇νx⦈P) ∈ Rel by(rule C1)
      ultimately show ?case by force
    qed
  next
    case(cBrClose K xvec N' P')
    from ‹Ψ ▷ M⟨N⟩.P ⟼ ¡K⦇ν*xvec⦈⟨N'⟩ ≺ P'›
    have Ψ ⊢ M ≼ the(subject(¡K⦇ν*xvec⦈⟨N'⟩))
      by(rule brOutputOutputCases) simp
    then have Ψ ⊢ M ≼ K by simp
    then have supp K ⊆ (supp M:: name set) by(rule chanOutConSupp)
    then have False using ‹x ∈ supp K› ‹x ♯ M› unfolding fresh-def
      by blast
    then show ?case by(rule FalseE)
  qed
 qed
qed

lemma casePushResLeft:
 fixes Ψ  :: 'b
  and x  :: name
  and Cs :: ('c × ('a, 'b, 'c) psi) list

assumes eqvt Rel
 and   x ♯ Ψ
 and   x ♯ map fst Cs
 and   C1: ⋀Q. (Ψ, Q, Q) ∈ Rel

shows Ψ ▷ ⦇νx⦈(Cases Cs) ⇝[Rel] Cases (map (λ(φ, P). (φ, ⦇νx⦈P)) Cs)
proof −
 note ‹eqvt Rel› ‹x ♯ Ψ›
 moreover have x ♯ ⦇νx⦈(Cases Cs) by(simp add: abs-fresh)
 moreover from ‹x ♯ map fst Cs› have x ♯ Cases (map (λ(φ, P). (φ, ⦇νx⦈P))
Cs)
   by(induct Cs) (auto simp add: abs-fresh)
 ultimately show ?thesis
 proof(induct rule: simIFresh[of - - - - - Cs])
   case(cSim α P'')
   from ‹Ψ ▷ Cases (map (λ(φ, P). (φ, ⦇νx⦈P)) Cs) ⟼α ≺ P''›
   show ?case
   proof(induct rule: caseCases)
     case(cCase φ P')
```

from ‹$(\varphi,\ P') \in set\ (map\ (\lambda(\varphi,\ P).\ (\varphi,\ (\!|\nu x|\!)P))\ Cs)$›
obtain P **where** $(\varphi,\ P) \in set\ Cs$ **and** $P' = (\!|\nu x|\!)P$
by(*induct Cs*) *auto*
from ‹*guarded* P'› ‹$P' = (\!|\nu x|\!)P$› **have** *guarded* P **by** *simp*
from ‹$\Psi \rhd P' \longmapsto \alpha \prec P''$› ‹$P' = (\!|\nu x|\!)P$› **have** $\Psi \rhd (\!|\nu x|\!)P \longmapsto \alpha \prec P''$
by *simp*
moreover note ‹$x \sharp \Psi$› ‹$x \sharp \alpha$› ‹$x \sharp P''$› ‹$bn\ \alpha \sharp* \Psi$›
moreover from ‹$bn\ \alpha \sharp* Cs$› ‹$(\varphi,\ P) \in set\ Cs$›
have $bn\ \alpha \sharp* P$ **by**(*auto dest*: *memFreshChain*)
ultimately show *?case* **using** ‹$bn\ \alpha \sharp* subject\ \alpha$› ‹$x \sharp \alpha$› ‹$bn\ \alpha \sharp* Cs$›
proof(*induct rule*: *resCases*[**where** C=()])
case(*cOpen M xvec1 xvec2 y N P'*)
from ‹$x \sharp M(\!|\nu*(xvec1@y\#xvec2)|\!)\langle N\rangle$› **have** $x \sharp xvec1$ **and** $x \sharp xvec2$ **and** $x \sharp M$ **by** *simp+*
from ‹$bn(M(\!|\nu*(xvec1@y\#xvec2)|\!)\langle N\rangle) \sharp* Cs$› **have** $y \sharp Cs$ **by** *simp*
from ‹$\Psi \rhd P \longmapsto M(\!|\nu*(xvec1@xvec2)|\!)\langle([(x,\ y)] \cdot N)\rangle \prec ([(x,\ y)] \cdot P')$› ‹$(\varphi,\ P) \in set\ Cs$› ‹$\Psi \vdash \varphi$› ‹*guarded* P›
have $\Psi \rhd Cases\ Cs \longmapsto M(\!|\nu*(xvec1@xvec2)|\!)\langle([(x,\ y)] \cdot N)\rangle \prec ([(x,\ y)] \cdot P')$ **by**(*rule Case*)
then have $([(x,\ y)] \cdot \Psi) \rhd ([(x,\ y)] \cdot (Cases\ Cs)) \longmapsto ([(x,\ y)] \cdot (M(\!|\nu*(xvec1@xvec2)|\!)\langle([(x,\ y)] \cdot N)\rangle \prec ([(x,\ y)] \cdot P')))$
by(*rule semantics.eqvt*)
with ‹$x \sharp \Psi$› ‹$x \sharp M$› ‹$y \sharp xvec1$› ‹$y \sharp xvec2$› ‹$y \sharp \Psi$› ‹$y \sharp M$› ‹$x \sharp xvec1$› ‹$x \sharp xvec2$›
have $\Psi \rhd ([(x,\ y)] \cdot (Cases\ Cs)) \longmapsto M(\!|\nu*(xvec1@xvec2)|\!)\langle N\rangle \prec P'$ **by**(*simp add*: *eqvts*)
then have $\Psi \rhd (\!|\nu y|\!)([(x,\ y)] \cdot (Cases\ Cs)) \longmapsto M(\!|\nu*(xvec1@y\#xvec2)|\!)\langle N\rangle \prec P'$ **using** ‹$y \in supp\ N$› ‹$y \sharp \Psi$› ‹$y \sharp M$› ‹$y \sharp xvec1$› ‹$y \sharp xvec2$›
by(*rule Open*)
then have $\Psi \rhd (\!|\nu x|\!)(Cases\ Cs) \longmapsto M(\!|\nu*(xvec1@y\#xvec2)|\!)\langle N\rangle \prec P'$ **using** ‹$y \sharp Cs$›
by(*simp add*: *alphaRes*)
moreover have $(\Psi,\ P',\ P') \in Rel$ **by**(*rule C1*)
ultimately show *?case* **by** *blast*
next
case(*cBrOpen M xvec1 xvec2 y N P'*)
from ‹$x \sharp ¡M(\!|\nu*(xvec1@y\#xvec2)|\!)\langle N\rangle$› **have** $x \sharp xvec1$ **and** $x \sharp xvec2$ **and** $x \sharp M$ **by** *simp+*
from ‹$bn(¡M(\!|\nu*(xvec1@y\#xvec2)|\!)\langle N\rangle) \sharp* Cs$› **have** $y \sharp Cs$ **by** *simp*
from ‹$\Psi \rhd P \longmapsto ¡M(\!|\nu*(xvec1@xvec2)|\!)\langle([(x,\ y)] \cdot N)\rangle \prec ([(x,\ y)] \cdot P')$› ‹$(\varphi,\ P) \in set\ Cs$› ‹$\Psi \vdash \varphi$› ‹*guarded* P›
have $\Psi \rhd Cases\ Cs \longmapsto ¡M(\!|\nu*(xvec1@xvec2)|\!)\langle([(x,\ y)] \cdot N)\rangle \prec ([(x,\ y)] \cdot P')$ **by**(*rule Case*)
then have $([(x,\ y)] \cdot \Psi) \rhd ([(x,\ y)] \cdot (Cases\ Cs)) \longmapsto ([(x,\ y)] \cdot (¡M(\!|\nu*(xvec1@xvec2)|\!)\langle([(x,\ y)] \cdot N)\rangle \prec ([(x,\ y)] \cdot P')))$
by(*rule semantics.eqvt*)
with ‹$x \sharp \Psi$› ‹$x \sharp M$› ‹$y \sharp xvec1$› ‹$y \sharp xvec2$› ‹$y \sharp \Psi$› ‹$y \sharp M$› ‹$x \sharp xvec1$› ‹$x \sharp xvec2$›
have $\Psi \rhd ([(x,\ y)] \cdot (Cases\ Cs)) \longmapsto ¡M(\!|\nu*(xvec1@xvec2)|\!)\langle N\rangle \prec P'$

by(*simp add*: *eqvts*)
then have $\Psi \rhd (\!|\nu y|\!)([(x, y)] \cdot (Cases\ Cs)) \longmapsto\!\!¡ M(\!|\nu*(xvec1@y\#xvec2)|\!)\langle N\rangle \prec P'$ **using** ‹$y \in supp\ N$› ‹$y \sharp \Psi$› ‹$y \sharp M$› ‹$y \sharp xvec1$› ‹$y \sharp xvec2$›
by(*rule BrOpen*)
then have $\Psi \rhd (\!|\nu x|\!)(Cases\ Cs) \longmapsto\!\!¡ M(\!|\nu*(xvec1@y\#xvec2)|\!)\langle N\rangle \prec P'$ **using** ‹$y \sharp Cs$›
by(*simp add*: *alphaRes*)
moreover have $(\Psi, P', P') \in Rel$ **by**(*rule C1*)
ultimately show *?case* **by** *blast*
next
case(*cRes P'*)
from ‹$\Psi \rhd P \longmapsto \alpha \prec P'$› ‹$(\varphi, P) \in set\ Cs$› ‹$\Psi \vdash \varphi$› ‹*guarded P*›
have $\Psi \rhd Cases\ Cs \longmapsto \alpha \prec P'$ **by**(*rule Case*)
then have $\Psi \rhd (\!|\nu x|\!)(Cases\ Cs) \longmapsto \alpha \prec (\!|\nu x|\!)P'$ **using** ‹$x \sharp \Psi$› ‹$x \sharp \alpha$›
by(*rule Scope*)
moreover have $(\Psi, (\!|\nu x|\!)P', (\!|\nu x|\!)P') \in Rel$ **by**(*rule C1*)
ultimately show *?case* **by** *blast*
next
case(*cBrClose M xvec N P'*)
from ‹$\Psi \rhd P \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec P'$› ‹$x \in supp\ M$› ‹$x \sharp \Psi$›
have $\Psi \rhd (\!|\nu x|\!)P \longmapsto \tau \prec (\!|\nu x|\!)((\!|\nu*xvec|\!)P')$
by(*rule BrClose*)
from ‹$\Psi \rhd P \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec P'$› ‹$(\varphi, P) \in set\ Cs$› ‹$\Psi \vdash \varphi$› ‹*guarded P*›
have $\Psi \rhd Cases\ Cs \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec P'$
by(*rule Case*)
then have $\Psi \rhd (\!|\nu x|\!)(Cases\ Cs) \longmapsto \tau \prec (\!|\nu x|\!)((\!|\nu*xvec|\!)P')$ **using** ‹$x \in supp\ M$› ‹$x \sharp \Psi$›
by(*rule BrClose*)
moreover have $(\Psi,(\!|\nu x|\!)((\!|\nu*xvec|\!)P'),(\!|\nu x|\!)((\!|\nu*xvec|\!)P')) \in Rel$ **by** *fact*
ultimately show *?case* **by** *blast*
qed
qed
qed
qed

lemma *casePushResRight*:
fixes Ψ :: *'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*: $\bigwedge Q.\ (\Psi, Q, Q) \in Rel$

shows $\Psi \rhd Cases\ (map\ (\lambda(\varphi, P).\ (\varphi, (\!|\nu x|\!)P))\ Cs) \rightsquigarrow [Rel]\ (\!|\nu x|\!)(Cases\ Cs)$
proof −
note ‹*eqvt Rel*› ‹$x \sharp \Psi$›

moreover from ‹$x \sharp$ *map fst Cs*› **have** $x \sharp$ *Cases* (*map* ($\lambda(\varphi, P).\ (\varphi, (\!|\nu x|\!)P)$) *Cs*)
by(*induct Cs*) (*auto simp add*: *abs-fresh*)
moreover have $x \sharp (\!|\nu x|\!)$(*Cases Cs*) **by**(*simp add*: *abs-fresh*)
ultimately show *?thesis*
proof(*induct rule*: *simIFresh*[*of* - - - - - *Cs*])
case(*cSim* α P'')
note ‹$\Psi \rhd (\!|\nu x|\!)$(*Cases Cs*) $\longmapsto \alpha \prec P''$› ‹$x \sharp \Psi$› ‹$x \sharp \alpha$› ‹$x \sharp P''$› ‹*bn* $\alpha \sharp* \Psi$›
moreover from ‹*bn* $\alpha \sharp* (\!|\nu x|\!)$(*Cases Cs*)› ‹$x \sharp \alpha$› **have** *bn* $\alpha \sharp* (Cases\ Cs)$ **by** *simp*
ultimately show *?case* **using** ‹*bn* $\alpha \sharp*$ *subject* α› ‹$x \sharp \alpha$› ‹*bn* $\alpha \sharp*$ *Cs*›
proof(*induct rule*: *resCases*[**where** C=()])
case(*cOpen M xvec1 xvec2 y N P'*)
from ‹$x \sharp M(\!|\nu*(xvec1@y\#xvec2)|\!)\langle N\rangle$› **have** $x \sharp$ *xvec1* **and** $x \sharp$ *xvec2* **and** $x \sharp M$ **by** *simp+*
from ‹*bn*($M(\!|\nu*(xvec1@y\#xvec2)|\!)\langle N\rangle$) $\sharp*$ *Cs*› **have** $y \sharp$ *Cs* **by** *simp*
from ‹$\Psi \rhd$ *Cases Cs* $\longmapsto M(\!|\nu*(xvec1@xvec2)|\!)\langle([(x,\ y)] \cdot N)\rangle \prec ([(x,\ y)] \cdot P')$›
show *?case*
proof(*induct rule*: *caseCases*)
case(*cCase* φ P)
from ‹$\Psi \rhd P \longmapsto M(\!|\nu*(xvec1@xvec2)|\!)\langle([(x,\ y)] \cdot N)\rangle \prec ([(x,\ y)] \cdot P')$›
have $([(x,\ y)] \cdot \Psi) \rhd ([(x,\ y)] \cdot P) \longmapsto ([(x,\ y)] \cdot (M(\!|\nu*(xvec1@xvec2)|\!)\langle([(x,\ y)] \cdot N)\rangle \prec ([(x,\ y)] \cdot P')))$
by(*rule semantics.eqvt*)
with ‹$x \sharp \Psi$› ‹$x \sharp M$› ‹$y \sharp$ *xvec1*› ‹$y \sharp$ *xvec2*› ‹$y \sharp \Psi$› ‹$y \sharp M$› ‹$x \sharp$ *xvec1*› ‹$x \sharp$ *xvec2*›
have $\Psi \rhd ([(x,\ y)] \cdot P) \longmapsto M(\!|\nu*(xvec1@xvec2)|\!)\langle N\rangle \prec P'$ **by**(*simp add*: *eqvts*)
then have $\Psi \rhd (\!|\nu y|\!)([(x,\ y)] \cdot P) \longmapsto M(\!|\nu*(xvec1@y\#xvec2)|\!)\langle N\rangle \prec P'$ **using** ‹$y \in$ *supp N*› ‹$y \sharp \Psi$› ‹$y \sharp M$› ‹$y \sharp$ *xvec1*› ‹$y \sharp$ *xvec2*›
by(*rule Open*)
then have $\Psi \rhd (\!|\nu x|\!)P \longmapsto M(\!|\nu*(xvec1@y\#xvec2)|\!)\langle N\rangle \prec P'$ **using** ‹$y \sharp$ *Cs*› ‹$(\varphi, P) \in$ *set Cs*›
by(*subst alphaRes*, *auto dest*: *memFresh*)
moreover from ‹$(\varphi, P) \in$ *set Cs*› **have** $(\varphi, (\!|\nu x|\!)P) \in$ *set* (*map* ($\lambda(\varphi, P).\ (\varphi, (\!|\nu x|\!)P)$) *Cs*)
by(*induct Cs*) *auto*
moreover note ‹$\Psi \vdash \varphi$›
moreover from ‹*guarded P*› **have** *guarded*($(\!|\nu x|\!)P$) **by** *simp*
ultimately have $\Psi \rhd$ (*Cases* (*map* ($\lambda(\varphi, P).\ (\varphi, (\!|\nu x|\!)P)$) *Cs*)) $\longmapsto M(\!|\nu*(xvec1@y\#xvec2)|\!)\langle N\rangle \prec P'$
by(*rule Case*)
moreover have $(\Psi, P', P') \in$ *Rel* **by**(*rule C1*)
ultimately show *?case* **by** *blast*
qed
next
case(*cBrOpen M xvec1 xvec2 y N P'*)
from ‹$x \sharp$ ¡$M(\!|\nu*(xvec1@y\#xvec2)|\!)\langle N\rangle$› **have** $x \sharp$ *xvec1* **and** $x \sharp$ *xvec2* **and**

$x \sharp M$ **by** *simp+*
from ‹$bn(¡M(\!|\nu*(xvec1 @ y\#xvec2)|\!)\langle N\rangle) \sharp* Cs$› **have** $y \sharp Cs$ **by** *simp*
from ‹$\Psi \rhd Cases\ Cs \longmapsto ¡M(\!|\nu*(xvec1 @ xvec2)|\!)\langle([(x,\ y)] \cdot N)\rangle \prec ([(x,\ y)] \cdot P')$›
show *?case*
proof(*induct rule*: *caseCases*)
case(*cCase* $\varphi\ P$)
from ‹$\Psi \rhd P \longmapsto ¡M(\!|\nu*(xvec1 @ xvec2)|\!)\langle([(x,\ y)] \cdot N)\rangle \prec ([(x,\ y)] \cdot P')$›
have $([(x,\ y)] \cdot \Psi) \rhd ([(x,\ y)] \cdot P) \longmapsto([(x,\ y)] \cdot (¡M(\!|\nu*(xvec1 @ xvec2)|\!)\langle([(x, y)] \cdot N)\rangle \prec ([(x,\ y)] \cdot P')))$
by(*rule semantics.eqvt*)
with ‹$x \sharp \Psi$› ‹$x \sharp M$› ‹$y \sharp xvec1$› ‹$y \sharp xvec2$› ‹$y \sharp \Psi$› ‹$y \sharp M$› ‹$x \sharp xvec1$› ‹$x \sharp xvec2$›
have $\Psi \rhd ([(x,\ y)] \cdot P) \longmapsto ¡M(\!|\nu*(xvec1 @ xvec2)|\!)\langle N\rangle \prec P'$ **by**(*simp add*: *eqvts*)
then have $\Psi \rhd (\!|\nu y|\!)([(x,\ y)] \cdot P) \longmapsto ¡M(\!|\nu*(xvec1 @ y\#xvec2)|\!)\langle N\rangle \prec P'$ **using** ‹$y \in supp\ N$› ‹$y \sharp \Psi$› ‹$y \sharp M$› ‹$y \sharp xvec1$› ‹$y \sharp xvec2$›
by(*rule BrOpen*)
then have $\Psi \rhd (\!|\nu x|\!)P \longmapsto ¡M(\!|\nu*(xvec1 @ y\#xvec2)|\!)\langle N\rangle \prec P'$ **using** ‹$y \sharp Cs$› ‹$(\varphi,\ P) \in set\ Cs$›
by(*subst alphaRes*, *auto dest*: *memFresh*)
moreover from ‹$(\varphi,\ P) \in set\ Cs$› **have** $(\varphi,\ (\!|\nu x|\!)P) \in set\ (map\ (\lambda(\varphi,\ P).\ (\varphi,\ (\!|\nu x|\!)P))\ Cs)$
by(*induct Cs*) *auto*
moreover note ‹$\Psi \vdash \varphi$›
moreover from ‹*guarded P*› **have** *guarded*$((\!|\nu x|\!)P)$ **by** *simp*
ultimately have $\Psi \rhd (Cases\ (map\ (\lambda(\varphi,\ P).\ (\varphi,\ (\!|\nu x|\!)P))\ Cs)) \longmapsto ¡M(\!|\nu*(xvec1 @ y\#xvec2)|\!)\langle N\rangle \prec P'$
by(*rule Case*)
moreover have $(\Psi,\ P',\ P') \in Rel$ **by**(*rule C1*)
ultimately show *?case* **by** *blast*
qed
next
case(*cRes P'*)
from ‹$\Psi \rhd Cases\ Cs \longmapsto \alpha \prec P'$›
show *?case*
proof(*induct rule*: *caseCases*)
case(*cCase* $\varphi\ P$)
from ‹$\Psi \rhd P \longmapsto \alpha \prec P'$› ‹$x \sharp \Psi$› ‹$x \sharp \alpha$›
have $\Psi \rhd (\!|\nu x|\!)P \longmapsto \alpha \prec (\!|\nu x|\!)P'$ **by**(*rule Scope*)
moreover from ‹$(\varphi,\ P) \in set\ Cs$› **have** $(\varphi,\ (\!|\nu x|\!)P) \in set\ (map\ (\lambda(\varphi,\ P).\ (\varphi,\ (\!|\nu x|\!)P))\ Cs)$
by(*induct Cs*) *auto*
moreover note ‹$\Psi \vdash \varphi$›
moreover from ‹*guarded P*› **have** *guarded*$((\!|\nu x|\!)P)$ **by** *simp*
ultimately have $\Psi \rhd (Cases\ (map\ (\lambda(\varphi,\ P).\ (\varphi,\ (\!|\nu x|\!)P))\ Cs)) \longmapsto \alpha \prec (\!|\nu x|\!)P'$
by(*rule Case*)
moreover have $(\Psi,\ (\!|\nu x|\!)P',\ (\!|\nu x|\!)P') \in Rel$ **by**(*rule C1*)

```
      ultimately show ?case by blast
    qed
  next
    case(cBrClose M xvec N P')
    then show ?case
    proof(induct rule: caseCases)
      case(cCase φ P)
      from ‹Ψ ▷ P ⟼¡M⦇ν*xvec⦈⟨N⟩ ≺ P'› ‹x ♯ Ψ› ‹x ∈ supp M›
      have Ψ ▷ ⦇νx⦈P ⟼ τ ≺ ⦇νx⦈(⦇ν*xvec⦈P')
        by(intro BrClose)
      moreover from ‹(φ, P) ∈ set Cs› have (φ, ⦇νx⦈P) ∈ set (map (λ(φ, P).
(φ, ⦇νx⦈P)) Cs)
        by(induct Cs) auto
      moreover from ‹guarded P› have guarded(⦇νx⦈P) by simp
      moreover note ‹Ψ ⊢ φ›
       ultimately have Ψ ▷ Cases map (λ(φ, P). (φ, ⦇νx⦈P)) Cs ⟼ τ ≺
⦇νx⦈(⦇ν*xvec⦈P')
        by(intro Case)
      moreover have (Ψ,⦇νx⦈(⦇ν*xvec⦈P'),⦇νx⦈(⦇ν*xvec⦈P')) ∈ Rel
        by fact
      ultimately show ?case
        by blast
    qed
  qed
 qed
qed

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: Ψ ▷ ⦇νx⦈P ⟼M⦇N⦈ ≺ P'
  and   x ♯ Ψ
  and   x ♯ M
  and   x ♯ N
  and   x ♯ P'
  and   rScope: ⋀P'. ⟦Ψ ▷ P ⟼M⦇N⦈ ≺ P'⟧ ⟹ Prop (⦇νx⦈P')

shows Prop P'
proof −
  note Trans ‹x ♯ Ψ›
  moreover from ‹x ♯ M› ‹x ♯ N› have x ♯ M⦇N⦈ by simp
  moreover note ‹x ♯ P'›
  ultimately show ?thesis using assms
```

by(*induct rule*: *resInputCases'*) *simp*
qed

lemma *resBrInputCases*[*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 −
note *Trans* ‹$x \sharp \Psi$›
moreover from ‹$x \sharp M$› ‹$x \sharp N$› **have** $x \sharp ¿M(\!|N|\!)$ **by** *simp*
moreover note ‹$x \sharp P'$›
ultimately show *?thesis* **using** *assms*
by(*induct rule*: *resBrInputCases'*) *simp*
qed

lemma *swap-supp*:
fixes x :: *name*
and y :: *name*
and N :: *'a*

assumes $y \in supp\ N$

shows $x \in supp\ ([(x, y)] \cdot N)$
using *assms*
by (*metis fresh-bij fresh-def swap-simps*(*2*))

lemma *swap-supp'*:
fixes x :: *name*
and y :: *name*
and N :: *'a*

assumes $x \in supp\ N$

shows $y \in supp\ ([(x, y)] \cdot N)$
using *assms*
by (*metis fresh-bij fresh-def swap-simps*(*1*))

lemma *brOutputFreshSubjectChain*:
  fixes Ψ :: $'b$
    and Q :: $('a, 'b, 'c)$ *psi*
    and M :: $'a$
    and *xvec* :: *name list*
    and N :: $'a$
    and Q' :: $('a, 'b, 'c)$ *psi*
    and *yvec* :: *name list*

assumes $\Psi \rhd Q \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec Q'$
  and *xvec* $\sharp*$ M
  and *yvec* $\sharp*$ Q

shows *yvec* $\sharp*$ M
  using *assms*
proof(*induct yvec*)
  case *Nil*
  then show *?case* **by** *simp*
next
  case(*Cons a yvec*)
  then have *yvec* $\sharp*$ M **by** *simp*
  from ‹$(a \# yvec) \sharp* Q$› **have** $a \sharp Q$ **by** *simp*
  from ‹*xvec* $\sharp*$ M› ‹$a \sharp Q$› ‹$\Psi \rhd Q \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec Q'$›
  have $a \sharp M$
    by(*simp add*: *brOutputFreshSubject*)
  with ‹*yvec* $\sharp*$ M› **show** *?case*
    by *simp*
qed

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*: $\bigwedge \Psi'\ zvec\ R\ y.\ [\![y \sharp \Psi'; zvec \sharp* \Psi']\!] \Longrightarrow (\Psi', (\!|\nu y|\!)((\!|\nu * zvec|\!)R), (\!|\nu * zvec|\!)((\!|\nu y|\!)R)) \in Rel$
  — Addition for Broadcast
  and *C4*: $\bigwedge \Psi' R S\ zvec.\ [\![zvec \sharp* R; zvec \sharp* \Psi']\!] \Longrightarrow (\Psi', ((\!|\nu * zvec|\!)(R \parallel S)), (R \parallel ((\!|\nu * zvec|\!)S))) \in Rel$

shows $\Psi \rhd (\!|\nu x|\!)(P \parallel Q) \rightsquigarrow [Rel]\ P \parallel (\!|\nu x|\!)Q$
proof −
note ‹*eqvt Rel*› ‹$x \sharp \Psi$›
moreover have $x \sharp (\!|\nu x|\!)(P \parallel Q)$ **by**(*simp add*: *abs-fresh*)
moreover from ‹$x \sharp P$› **have** $x \sharp P \parallel (\!|\nu x|\!)Q$ **by**(*simp add*: *abs-fresh*)
ultimately show *?thesis*
proof(*induct rule*: *simIFresh*[*of* - - - - - x])
case(*cSim* α PQ)
from ‹$x \sharp \alpha$› ‹*bn* $\alpha \sharp* (P \parallel (\!|\nu x|\!)Q)$› **have** *bn* $\alpha \sharp* Q$ **by** *simp*
note ‹$\Psi \rhd P \parallel (\!|\nu x|\!)Q \longmapsto \alpha \prec PQ$› ‹*bn* $\alpha \sharp* \Psi$›
moreover from ‹*bn* $\alpha \sharp* (P \parallel (\!|\nu x|\!)Q)$› **have** *bn* $\alpha \sharp* P$ **and** *bn* $\alpha \sharp* (\!|\nu x|\!)Q$ **by** *simp+*
ultimately show *?case* **using** ‹*bn* $\alpha \sharp*$ *subject* α› ‹$x \sharp PQ$›
proof(*induct rule*: *parCases*[**where** $C{=}x$])
case(*cPar1* P' A_Q Ψ_Q)
from ‹$x \sharp P' \parallel (\!|\nu x|\!)Q$› **have** $x \sharp P'$ **by** *simp*
have *PTrans*: $\Psi \otimes \Psi_Q \rhd P \longmapsto \alpha \prec P'$ **by** *fact*
from ‹*extractFrame*$((\!|\nu x|\!)Q) = \langle A_Q, \Psi_Q\rangle$› **have** *FrxQ*: $(\!|\nu x|\!)(\mathit{extractFrame}\ Q) = \langle A_Q, \Psi_Q\rangle$ **by** *simp*
then obtain $y\ A_Q{'}$ **where** *A*: $A_Q = y\#A_Q{'}$ **by**(*cases* A_Q) *auto*
with ‹$A_Q \sharp* \Psi$› ‹$A_Q \sharp* P$› ‹$A_Q \sharp* \alpha$›
have $A_Q{'} \sharp* \Psi$ **and** $A_Q{'} \sharp* P$ **and** $A_Q{'} \sharp* \alpha$
and $y \sharp \Psi$ **and** $y \sharp P$ **and** $y \sharp \alpha$
by *simp+*
from *PTrans* ‹$y \sharp P$› ‹$y \sharp \alpha$› ‹*bn* $\alpha \sharp*$ *subject* α› ‹*distinct*(*bn* α)› **have** $y \sharp P'$
by(*auto intro*: *freeFreshDerivative*)
note *PTrans*
moreover from *A* ‹$A_Q \sharp* x$› *FrxQ* **have** *extractFrame*$([(y, x)] \cdot Q) = \langle A_Q{'}, \Psi_Q\rangle$
by(*simp add*: *frame.inject alpha' fresh-list-cons eqvts*)
moreover from ‹*bn* $\alpha \sharp* Q$› **have** $([(y, x)] \cdot (\mathit{bn}\ \alpha)) \sharp* ([(y, x)] \cdot Q)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$x \sharp \alpha$› ‹$A_Q \sharp* \alpha$› *A* **have** *bn* $\alpha \sharp* ([(y, x)] \cdot Q)$ **by** *simp*
ultimately have $\Psi \rhd P \parallel ([(y, x)] \cdot Q) \longmapsto \alpha \prec (P' \parallel ([(y, x)] \cdot Q))$
using ‹$A_Q{'} \sharp* \Psi$› ‹$A_Q{'} \sharp* P$› ‹$A_Q{'} \sharp* \alpha$›
by(*rule Par1*)
then have $\Psi \rhd (\!|\nu y|\!)(P \parallel ([(y, x)] \cdot Q)) \longmapsto \alpha \prec (\!|\nu y|\!)(P' \parallel ([(y, x)] \cdot Q))$
using ‹$y \sharp \Psi$› ‹$y \sharp \alpha$›
by(*rule Scope*)
then have $([(y, x)] \cdot \Psi) \rhd ([(y, x)] \cdot ((\!|\nu y|\!)(P \parallel ([(y, x)] \cdot Q)))) \longmapsto ([(y, x)] \cdot (\alpha \prec (\!|\nu y|\!)(P' \parallel ([(y, x)] \cdot Q))))$
by(*rule semantics.eqvt*)
with ‹$x \sharp \Psi$› ‹$y \sharp \Psi$› ‹$x \sharp P$› ‹$y \sharp P$› ‹$x \sharp \alpha$› ‹$y \sharp \alpha$› ‹$x \sharp P'$› ‹$y \sharp P'$›
have $\Psi \rhd (\!|\nu x|\!)(P \parallel Q) \longmapsto \alpha \prec (\!|\nu x|\!)(P' \parallel Q)$
by(*simp add*: *eqvts calc-atm*)
moreover from ‹$x \sharp \Psi$› ‹$x \sharp P'$› **have** $(\Psi, (\!|\nu x|\!)((\!|\nu *[]|\!)(P' \parallel Q)), (\!|\nu *[]|\!)(P' \parallel (\!|\nu x|\!)Q)) \in Rel$
by(*rule C2*) *auto*
ultimately show *?case*

by *force*
next
case(*cPar2 xQ′ A*$_P$ Ψ_P)
from ‹$A_P \sharp* (\!|\nu x|\!) Q$› ‹$A_P \sharp* x$› **have** $A_P \sharp* Q$ **by** *simp*
note ‹$\Psi \otimes \Psi_P \rhd (\!|\nu x|\!) Q \longmapsto \alpha \prec xQ'$›
moreover have *FrP*: *extractFrame P* = ⟨*A*$_P$, Ψ_P⟩ **by** *fact*
with ‹$x \sharp P$› ‹$A_P \sharp* x$› **have** $x \sharp \Psi_P$ **and** $x \sharp A_P$
by(*force dest*: *extractFrameFresh*)+
with ‹$x \sharp \Psi$› **have** $x \sharp \Psi \otimes \Psi_P$ **by** *force*
moreover note ‹$x \sharp \alpha$›
moreover from ‹$x \sharp P \parallel xQ'$› **have** $x \sharp xQ'$ **by** *simp*
moreover from *FrP* ‹*bn* $\alpha \sharp* P$› ‹$A_P \sharp* \alpha$› **have** *bn* $\alpha \sharp* \Psi_P$
by(*auto dest*: *extractFrameFreshChain*)
with ‹*bn* $\alpha \sharp* \Psi$› **have** *bn* $\alpha \sharp* (\Psi \otimes \Psi_P)$ **by** *force*
ultimately show *?case* **using** ‹*bn* $\alpha \sharp* Q$› ‹*bn* $\alpha \sharp*$ *subject* α› ‹$x \sharp \alpha$› ‹*bn* $\alpha \sharp* P$› ‹$A_P \sharp* \alpha$› ‹*bn* $\alpha \sharp* \Psi$› ‹$x \sharp P$›
proof(*induct rule*: *resCases′*[**where** *C*=(*P*, *A*$_P$, Ψ)])
case(*cOpen M xvec1 xvec2 y N Q′*)
from ‹$x \sharp M(\!|\nu*(xvec1@y\#xvec2)|\!)\langle N\rangle$› **have** $x \sharp$ *xvec1* **and** $x \neq y$ **and** $x \sharp$ *xvec2* **by** *simp*+
from ‹*bn*($M(\!|\nu*(xvec1@y\#xvec2)|\!)\langle N\rangle$) $\sharp*$ Ψ› **have** $y \sharp \Psi$ **by** *simp*
note ‹$\Psi \otimes \Psi_P \rhd ([(x, y)] \cdot Q) \longmapsto M(\!|\nu*(xvec1@xvec2)|\!)\langle N\rangle \prec Q'$› *FrP*
moreover from ‹*bn*($M(\!|\nu*(xvec1@y\#xvec2)|\!)\langle N\rangle$) $\sharp*$ *P*› **have** (*xvec1*@*xvec2*) $\sharp*$ *P* **and** $y \sharp P$ **by** *simp*+
moreover from ‹$A_P \sharp* (M(\!|\nu*(xvec1@y\#xvec2)|\!)\langle N\rangle)$› **have** $A_P \sharp* (M(\!|\nu*(xvec1@xvec2)|\!)\langle N\rangle)$ **and** $y \sharp A_P$ **by** *simp*+
moreover from ‹$A_P \sharp* Q$› ‹$x \sharp A_P$› ‹$y \sharp A_P$› **have** $A_P \sharp* ([(x, y)] \cdot Q)$ **by** *simp*
ultimately have $\Psi \rhd P \parallel ([(x, y)] \cdot Q) \longmapsto M(\!|\nu*(xvec1@xvec2)|\!)\langle N\rangle \prec (P \parallel Q')$
using ‹$A_P \sharp* \Psi$›
by(*intro Par2*) (*assumption* | *simp*)+
then have $\Psi \rhd (\!|\nu y|\!)(P \parallel ([(x, y)] \cdot Q)) \longmapsto M(\!|\nu*(xvec1@y\#xvec2)|\!)\langle N\rangle \prec (P \parallel Q')$
using ‹$y \in$ *supp N*› ‹$y \sharp \Psi$› ‹$y \sharp M$› ‹$y \sharp$ *xvec1*› ‹$y \sharp$ *xvec2*›
by(*rule Open*)
with ‹$x \sharp P$› ‹$y \sharp P$› ‹$y \sharp Q$› **have** $\Psi \rhd (\!|\nu x|\!)(P \parallel Q) \longmapsto M(\!|\nu*(xvec1@y\#xvec2)|\!)\langle N\rangle \prec (P \parallel Q')$
by(*subst alphaRes*[**where** *y*=*y*]) (*simp add*: *fresh-left calc-atm eqvts*)+
moreover have (Ψ, $P \parallel Q'$, $P \parallel Q'$) ∈ *Rel* **by**(*rule C1*)
ultimately show *?case* **by** *blast*
next
case(*cBrOpen M xvec1 xvec2 y N Q′*)
from ‹$x \sharp ¡M(\!|\nu*(xvec1@y\#xvec2)|\!)\langle N\rangle$› **have** $x \sharp$ *xvec1* **and** $x \neq y$ **and** $x \sharp$ *xvec2* **by** *simp*+
from ‹*bn*($¡M(\!|\nu*(xvec1@y\#xvec2)|\!)\langle N\rangle$) $\sharp*$ Ψ› **have** $y \sharp \Psi$ **by** *simp*
note ‹$\Psi \otimes \Psi_P \rhd ([(x, y)] \cdot Q) \longmapsto ¡M(\!|\nu*(xvec1@xvec2)|\!)\langle N\rangle \prec Q'$› *FrP*
moreover from ‹*bn*($¡M(\!|\nu*(xvec1@y\#xvec2)|\!)\langle N\rangle$) $\sharp*$ *P*› **have** (*xvec1*@*xvec2*) $\sharp*$ *P* **and** $y \sharp P$ **by** *simp*+

moreover from ‹$A_P \sharp* (¡M(\!|\nu*(xvec1@y\#xvec2)|\!)\langle N\rangle)$› **have** $A_P \sharp* (¡M(\!|\nu*(xvec1@xvec2)|\!)\langle N\rangle)$ **and** $y \sharp A_P$ **by** *simp+*

moreover from ‹$A_P \sharp* Q$› ‹$x \sharp A_P$› ‹$y \sharp A_P$› **have** $A_P \sharp* ([(x, y)] \cdot Q)$ **by** *simp*

ultimately have $\Psi \rhd P \parallel ([(x, y)] \cdot Q) \longmapsto ¡M(\!|\nu*(xvec1@xvec2)|\!)\langle N\rangle \prec (P \parallel Q')$

using ‹$A_P \sharp* \Psi$›

by(*intro Par2*) (*assumption* | *simp*)+

then have $\Psi \rhd (\!|\nu y|\!)(P \parallel ([(x, y)] \cdot Q)) \longmapsto ¡M(\!|\nu*(xvec1@y\#xvec2)|\!)\langle N\rangle \prec (P \parallel Q')$

using ‹$y \in supp\ N$› ‹$y \sharp \Psi$› ‹$y \sharp M$› ‹$y \sharp xvec1$› ‹$y \sharp xvec2$›

by(*rule BrOpen*)

with ‹$x \sharp P$› ‹$y \sharp P$› ‹$y \sharp Q$› **have** $\Psi \rhd (\!|\nu x|\!)(P \parallel Q) \longmapsto ¡M(\!|\nu*(xvec1@y\#xvec2)|\!)\langle N\rangle \prec (P \parallel Q')$

by(*subst alphaRes*[**where** *y=y*]) (*simp add*: *fresh-left calc-atm eqvts*)+

moreover have $(\Psi, P \parallel Q', P \parallel Q') \in Rel$ **by**(*rule C1*)

ultimately show *?case* **by** *blast*

next

case(*cRes Q'*)

from ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto \alpha \prec Q'$› *FrP* ‹$bn\ \alpha \sharp* P$›

have $\Psi \rhd P \parallel Q \longmapsto \alpha \prec (P \parallel Q')$ **using** ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* Q$› ‹$A_P \sharp* \alpha$›

by(*rule Par2*)

then have $\Psi \rhd (\!|\nu x|\!)(P \parallel Q) \longmapsto \alpha \prec (\!|\nu x|\!)(P \parallel Q')$ **using** ‹$x \sharp \Psi$› ‹$x \sharp \alpha$›

by(*rule Scope*)

moreover from ‹$x \sharp \Psi$› ‹$x \sharp P$› **have** $(\Psi, (\!|\nu x|\!)((\!|\nu*[]|\!)(P \parallel Q')), (\!|\nu*[]|\!)(P \parallel (\!|\nu x|\!)Q')) \in Rel$

by(*rule C2*) *auto*

ultimately show *?case*

by *force*

next

case(*cBrClose M xvec N Q'*)

from ‹$xvec \sharp* (P, A_P, \Psi)$›

have $xvec \sharp* P$ **and** $A_P \sharp* xvec$ **and** $xvec \sharp* \Psi$

by *simp+*

from ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec Q'$› ‹$A_P \sharp* Q$›

‹$xvec \sharp* M$›

have $A_P \sharp* M$

by(*simp add*: *brOutputFreshSubjectChain*)

from ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec Q'$› ‹$xvec \sharp* M$› ‹*distinct xvec*› ‹$A_P \sharp* Q$› ‹$A_P \sharp* xvec$›

have $A_P \sharp* N$ **and** $A_P \sharp* Q'$ **by**(*simp add*: *broutputFreshChainDerivative*)+

from ‹$A_P \sharp* M$› ‹$A_P \sharp* xvec$› ‹$A_P \sharp* N$› **have** $A_P \sharp* (¡M(\!|\nu*xvec|\!)\langle N\rangle)$

by *simp*

from ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec Q'$› ‹*extractFrame* $P = \langle A_P, \Psi_P\rangle$› ‹$xvec \sharp* P$› ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* Q$› ‹$A_P \sharp* (¡M(\!|\nu*xvec|\!)\langle N\rangle)$›

have $\Psi \rhd P \parallel Q \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec P \parallel Q'$

by(*simp add*: *Par2*)

from ‹$x \sharp \Psi$› ‹$x \sharp P$› ‹$xvec \sharp* P$› ‹$xvec \sharp* \Psi$›
have $(x\#xvec) \sharp* P$ **and** $(x\#xvec) \sharp* \Psi$ **by** *simp+*
then have $(\Psi, (\langle\!|\nu*(x\#xvec)|\!\rangle(P \parallel Q')), P \parallel (\langle\!|\nu*(x\#xvec)|\!\rangle Q')) \in Rel$
by(*rule C4*)
then have $(\Psi, \langle\!|\nu x|\!\rangle(\langle\!|\nu*xvec|\!\rangle(P \parallel Q')), P \parallel \langle\!|\nu x|\!\rangle(\langle\!|\nu*xvec|\!\rangle Q')) \in Rel$
by *simp*

moreover from ‹$\Psi \rhd P \parallel Q \longmapsto ¡M\langle\!|\nu*xvec|\!\rangle\langle N\rangle \prec P \parallel Q'$› ‹$x \in supp\ M$› ‹$x \sharp \Psi$›
have $\Psi \rhd \langle\!|\nu x|\!\rangle(P \parallel Q) \longmapsto \tau \prec \langle\!|\nu x|\!\rangle(\langle\!|\nu*xvec|\!\rangle(P \parallel Q'))$
by(*rule BrClose*)

ultimately show *?case*
by *force*
qed
next
case(*cComm1* $\Psi_Q\ M\ N\ P'\ A_P\ \Psi_P\ K\ xvec\ xQ'\ A_Q$)
have *QTrans*: $\Psi \otimes \Psi_P \rhd \langle\!|\nu x|\!\rangle Q \longmapsto K\langle\!|\nu*xvec|\!\rangle\langle N\rangle \prec xQ'$ **and** *FrQ*: *extractFrame*$(\langle\!|\nu x|\!\rangle Q) = \langle A_Q, \Psi_Q\rangle$ **by** *fact+*
have *PTrans*: $\Psi \otimes \Psi_Q \rhd P \longmapsto M\langle\!|N|\!\rangle \prec P'$ **and** *FrP*: *extractFrame* $P = \langle A_P, \Psi_P\rangle$ **by** *fact+*
have $x \sharp \langle\!|\nu x|\!\rangle Q$ **by**(*simp add*: *abs-fresh*)
with *QTrans* **have** $x \sharp N$ **and** $x \sharp xQ'$ **using** ‹$xvec \sharp* x$› ‹$xvec \sharp* K$› ‹*distinct xvec*›
by(*force intro*: *outputFreshDerivative*)+
from *PTrans* ‹$x \sharp P$› ‹$x \sharp N$› **have** $x \sharp P'$ **by**(*rule inputFreshDerivative*)
from ‹$x \sharp \langle\!|\nu x|\!\rangle Q$› *FrQ* ‹$A_Q \sharp* x$› **have** $x \sharp \Psi_Q$
by(*drule-tac extractFrameFresh*) *auto*
from ‹$x \sharp P$› *FrP* ‹$A_P \sharp* x$› **have** $x \sharp \Psi_P$
by(*drule-tac extractFrameFresh*) *auto*
from ‹$A_P \sharp* \langle\!|\nu x|\!\rangle Q$› ‹$A_P \sharp* x$› **have** $A_P \sharp* Q$ **by** *simp*
from ‹$A_Q \sharp* \langle\!|\nu x|\!\rangle Q$› ‹$A_Q \sharp* x$› **have** $A_Q \sharp* Q$ **by** *simp*
from *PTrans FrP* ‹*distinct* A_P› ‹$x \sharp P$› ‹$A_Q \sharp* P$› ‹$xvec \sharp* P$› ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* \Psi_Q$› ‹$A_P \sharp* x$› ‹$A_P \sharp* A_Q$› ‹$A_P \sharp* P$› ‹$A_P \sharp* M$› ‹$A_P \sharp* xvec$›
obtain M' **where** $(\Psi \otimes \Psi_Q) \otimes \Psi_P \vdash M \leftrightarrow M'$ **and** $x \sharp M'$ **and** $A_Q \sharp* M'$ **and** $xvec \sharp* M'$
by(*elim inputObtainPrefix*[**where** B=$x\#xvec@A_Q$]) (*assumption* | *force simp add*: *fresh-star-list-cons*)+
then have *MeqM'*: $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow M'$ **by**(*metis statEqEnt Associativity Commutativity Composition*)
with ‹$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K$› **have** $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash K \leftrightarrow M'$
by(*blast intro*: *chanEqTrans chanEqSym*)
then have $(\Psi \otimes \Psi_P) \otimes \Psi_Q \vdash K \leftrightarrow M'$ **by**(*metis statEqEnt Associativity Commutativity Composition*)
with *QTrans FrQ* ‹*distinct* A_Q› ‹$A_Q \sharp* \Psi$› ‹$A_Q \sharp* \Psi_P$› ‹$A_Q \sharp* (\langle\!|\nu x|\!\rangle Q)$› ‹$A_Q \sharp* K$› ‹$A_Q \sharp* M'$›
have $\Psi \otimes \Psi_P \rhd \langle\!|\nu x|\!\rangle Q \longmapsto M'\langle\!|\nu*xvec|\!\rangle\langle N\rangle \prec xQ'$
by(*force intro*: *outputRenameSubject*)

moreover from ‹$x \sharp \Psi$› ‹$x \sharp \Psi_P$› **have** $x \sharp \Psi \otimes \Psi_P$ **by** *force*
moreover from ‹$xvec \sharp* x$› **have** $x \sharp xvec$ **by** *simp*
with ‹$x \sharp M'$› ‹$x \sharp N$› **have** $x \sharp M'\langle\!|\nu*xvec|\!\rangle\langle N\rangle$ **by** *simp*
moreover note ‹$x \sharp xQ'$›

moreover from ‹$xvec \sharp* \Psi$› ‹$xvec \sharp* \Psi_P$› **have** $xvec \sharp* (\Psi \otimes \Psi_P)$ **by** *force*
moreover from ‹$xvec \sharp* (\!|\nu x|\!) Q$› ‹$x \sharp xvec$› **have** $xvec \sharp* Q$ **by** *simp*
moreover note ‹$xvec \sharp* M'$›

moreover from ‹$xvec \sharp* \Psi$› ‹$xvec \sharp* \Psi_P$› **have** $bn(M'\langle\!|\nu*xvec|\!\rangle\langle N\rangle) \sharp* (\Psi \otimes \Psi_P)$ **by** *force*
moreover from ‹$xvec \sharp* (\!|\nu x|\!) Q$› ‹$x \sharp xvec$› **have** $bn(M'\langle\!|\nu*xvec|\!\rangle\langle N\rangle) \sharp* Q$ **by** *simp*
moreover from ‹$xvec \sharp* P$› **have** $bn(M'\langle\!|\nu*xvec|\!\rangle\langle N\rangle) \sharp* P$ **by** *simp*
from ‹$xvec \sharp* \Psi$› **have** $bn(M'\langle\!|\nu*xvec|\!\rangle\langle N\rangle) \sharp* \Psi$ **by** *simp*
from ‹$A_Q \sharp* xvec$› ‹$A_Q \sharp* M'$› ‹$A_Q \sharp* N$› **have** $A_Q \sharp* (M'\langle\!|\nu*xvec|\!\rangle\langle N\rangle)$ **by** *simp*
have $object(M'\langle\!|\nu*xvec|\!\rangle\langle N\rangle) = Some\ N$ **by** *simp*
have $bn(M'\langle\!|\nu*xvec|\!\rangle\langle N\rangle) = xvec$ **by** *simp*
have $subject(M'\langle\!|\nu*xvec|\!\rangle\langle N\rangle) = Some\ M'$ **by** *simp*
from ‹$xvec \sharp* M'$› **have** $bn(M'\langle\!|\nu*xvec|\!\rangle\langle N\rangle) \sharp* subject(M'\langle\!|\nu*xvec|\!\rangle\langle N\rangle)$ **by** *simp*
ultimately show *?case*
using ‹$x \sharp M'\langle\!|\nu*xvec|\!\rangle\langle N\rangle$› ‹$bn(M'\langle\!|\nu*xvec|\!\rangle\langle N\rangle) \sharp* P$› ‹$bn(M'\langle\!|\nu*xvec|\!\rangle\langle N\rangle) \sharp* \Psi$› ‹$object(M'\langle\!|\nu*xvec|\!\rangle\langle N\rangle) = Some\ N$›
‹$bn(M'\langle\!|\nu*xvec|\!\rangle\langle N\rangle) = xvec$› ‹$subject(M'\langle\!|\nu*xvec|\!\rangle\langle N\rangle) = Some\ M'$› ‹$A_Q \sharp* (M'\langle\!|\nu*xvec|\!\rangle\langle N\rangle)$›
proof(*induct rule*: *resOutputCases''''*)
case(*cOpen M'' xvec1 xvec2 y N' Q'*)
then have *Eq*: $M'\langle\!|\nu*xvec|\!\rangle\langle N\rangle = M''\langle\!|\nu*(xvec1\ @\ y\ \#\ xvec2)|\!\rangle\langle N'\rangle$ **by** *simp*
from ‹$x \sharp M'\langle\!|\nu*xvec|\!\rangle\langle N\rangle$› *Eq* **have** $x \sharp xvec1$ **and** $x \neq y$ **and** $x \sharp xvec2$ **and** $x \sharp M''$
by *simp+*
from ‹$bn(M'\langle\!|\nu*xvec|\!\rangle\langle N\rangle) \sharp* P$› *Eq* **have** $(xvec1@xvec2) \sharp* P$ **and** $y \sharp P$ **by** *simp+*
from ‹$A_Q \sharp* (M'\langle\!|\nu*xvec|\!\rangle\langle N\rangle)$› *Eq* **have** $(xvec1@xvec2) \sharp* A_Q$ **and** $y \sharp A_Q$ **and** $A_Q \sharp* M''$ **by** *simp+*
from ‹$bn(M'\langle\!|\nu*xvec|\!\rangle\langle N\rangle) \sharp* \Psi$› *Eq* **have** $(xvec1@xvec2) \sharp* \Psi$ **and** $y \sharp \Psi$ **by** *simp+*
from *Eq* **have** $N = N'$ **and** $xvec = xvec1@y\#xvec2$ **and** $M' = M''$ **by**(*simp add*: *action.inject*)+
from ‹$x \sharp P$› ‹$y \sharp P$› ‹$x \neq y$› ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto M\langle\!|N|\!\rangle \prec P'$›
have $\Psi \otimes \Psi_Q \rhd P \longmapsto M\langle\!|([(y,\ x)] \cdot N)|\!\rangle \prec ([(y,\ x)] \cdot P')$
by(*elim inputAlpha*[**where** *xvec*=[*y*]]) (*auto simp add*: *calc-atm*)
then have *PTrans*: $\Psi \otimes \Psi_Q \rhd P \longmapsto M\langle\!|([(x,\ y)] \cdot N)|\!\rangle \prec ([(x,\ y)] \cdot P')$
by(*simp add*: *name-swap*)

from ‹$\Psi \otimes \Psi_P \rhd [(x,\ y)] \cdot Q \longmapsto M''\langle\!|\nu*(xvec1\ @\ xvec2)|\!\rangle\langle N'\rangle \prec Q'$›

have $[(x,\, y)] \cdot (\Psi \otimes \Psi_P \rhd [(x,\, y)] \cdot Q \longmapsto M''(\!|\nu*(xvec1 \mathbin{@} xvec2)|\!)\langle N'\rangle \prec Q')$
by *simp*
then have $[(x,\, y)] \cdot (\Psi \otimes \Psi_P) \rhd ([(x,\, y)] \cdot ([(x,\, y)] \cdot Q)) \longmapsto ([(x,\, y)] \cdot M'')(\!|\nu*([(x,\, y)] \cdot (xvec1 \mathbin{@} xvec2))|\!)\langle([(x,\, y)] \cdot N')\rangle \prec [(x,\, y)] \cdot Q'$
by(*simp add*: *eqvts*)
with $\langle x \sharp (\Psi \otimes \Psi_P)\rangle$ $\langle y \sharp (\Psi \otimes \Psi_P)\rangle$ $\langle x \sharp xvec1\rangle$ $\langle y \sharp xvec1\rangle$ $\langle x \sharp xvec2\rangle$ $\langle y \sharp xvec2\rangle$ $\langle x \sharp M''\rangle$ $\langle y \sharp M''\rangle$
have *QTrans*: $\Psi \otimes \Psi_P \rhd Q \longmapsto M''(\!|\nu*(xvec1 \mathbin{@} xvec2)|\!)\langle([(x,\, y)] \cdot N')\rangle \prec [(x,\, y)] \cdot Q'$
by *simp*
with $\langle A_Q \sharp* x\rangle$ $\langle y \sharp A_Q\rangle$ $\langle$*distinct xvec1*$\rangle$ $\langle$*distinct xvec2*$\rangle$ $\langle xvec1 \sharp* xvec2\rangle$ $\langle xvec1 \sharp* M''\rangle$ $\langle xvec2 \sharp* M''\rangle$
$\langle (xvec1@xvec2) \sharp* A_Q\rangle$
have $A_Q \sharp* ([(x,\, y)] \cdot Q')$ **using** $\langle A_Q \sharp* Q\rangle$
by(*elim outputFreshChainDerivative*(*2*)) (*assumption* | *simp*)+

from $\langle extractFrame((\!|\nu x|\!)\, Q) = \langle A_Q,\, \Psi_Q\rangle\rangle$ **have** *FrxQ*: $(\!|\nu x|\!)(extractFrame\ Q) = \langle A_Q,\, \Psi_Q\rangle$ **by** *simp*
then obtain $z\ A_Q'$ **where** *A*: $A_Q = z \# A_Q'$ **by**(*cases* A_Q) *auto*
with $\langle A_Q \sharp* \Psi\rangle$ $\langle A_Q \sharp* P\rangle$ $\langle A_Q \sharp* P'\rangle$ $\langle A_Q \sharp* \Psi_P\rangle$ $\langle A_Q \sharp* Q\rangle$ $\langle (xvec1@xvec2) \sharp* A_Q\rangle$ $\langle A_Q \sharp* M''\rangle$ $\langle A_Q \sharp* ([(x,\, y)] \cdot Q')\rangle$ $\langle y \sharp A_Q\rangle$ $\langle A_Q \sharp* N\rangle$
have $A_Q' \sharp* \Psi$ **and** $A_Q' \sharp* P$ **and** $A_Q' \sharp* \Psi_P$ **and** $A_Q' \sharp* Q$
and $z \sharp \Psi$ **and** $z \sharp P$ **and** $z \sharp P'$ **and** $z \sharp \Psi_P$ **and** $z \sharp Q$ **and** $z \sharp xvec1$ **and** $z \sharp xvec2$
and $z \sharp M''$ **and** $z \sharp ([(x,\, y)] \cdot Q')$ **and** $A_Q' \sharp* M''$ **and** $z \neq y$ **and** $z \sharp (xvec1@xvec2)$
by *auto*
from A $\langle A_P \sharp* A_Q\rangle$ **have** $A_P \sharp* A_Q'$ **and** $z \sharp A_P$ **by** *simp*+
from A $\langle A_Q \sharp* x\rangle$ **have** $x \neq z$ **and** $x \sharp A_Q'$ **by** *simp*+

from $\langle$*distinct* $A_Q\rangle$ A **have** $z \sharp A_Q'$
by(*induct* A_Q') (*auto simp add*: *fresh-list-nil fresh-list-cons*)
from *PTrans* $\langle x \sharp P\rangle$ $\langle z \sharp P\rangle$ $\langle x \neq z\rangle$ **have** $\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|([(x,\, z)] \cdot [(x,\, y)] \cdot N)|\!) \prec ([(x,\, z)] \cdot [(x,\, y)] \cdot P')$
by(*elim inputAlpha*[**where** *xvec*=[*x*]]) (*auto simp add*: *calc-atm*)
moreover note *FrP*
moreover from *QTrans* **have** $([(x,\, z)] \cdot (\Psi \otimes \Psi_P)) \rhd ([(x,\, z)] \cdot Q) \longmapsto ([(x,\, z)] \cdot (M''(\!|\nu*(xvec1@xvec2)|\!)\langle([(x,\, y)] \cdot N')\rangle \prec ([(x,\, y)] \cdot Q')))$
by(*rule semantics.eqvt*)
with $\langle x \sharp \Psi\rangle$ $\langle z \sharp \Psi\rangle$ $\langle x \sharp \Psi_P\rangle$ $\langle z \sharp \Psi_P\rangle$ $\langle x \sharp M''\rangle$ $\langle z \sharp M''\rangle$ $\langle x \sharp xvec1\rangle$ $\langle x \sharp xvec2\rangle$ $\langle z \sharp xvec1\rangle$ $\langle z \sharp xvec2\rangle$
have $\Psi \otimes \Psi_P \rhd ([(x,\, z)] \cdot Q) \longmapsto M''(\!|\nu*(xvec1@xvec2)|\!)\langle([(x,\, z)] \cdot [(x,\, y)] \cdot N')\rangle \prec ([(x,\, z)] \cdot [(x,\, y)] \cdot Q')$
by(*simp add*: *eqvts*)
moreover from A $\langle A_Q \sharp* x\rangle$ *FrxQ* **have** $extractFrame([(x,\, z)] \cdot Q) = \langle A_Q',\, \Psi_Q\rangle$
by(*clarsimp simp add*: *alpha' eqvts frame.inject fresh-list-cons name-swap*)
moreover from $\langle A_P \sharp* Q\rangle$ **have** $([(x,\, z)] \cdot A_P) \sharp* ([(x,\, z)] \cdot Q)$

by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$A_P \sharp* x$› ‹$z \sharp A_P$› **have** $A_P \sharp* ([(x, z)] \cdot Q)$ **by** *simp*
moreover from ‹$A_Q{}' \sharp* Q$› **have** $([(x, z)] \cdot A_Q{}') \sharp* ([(x, z)] \cdot Q)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$x \sharp A_Q{}'$› ‹$z \sharp A_Q{}'$› **have** $A_Q{}' \sharp* ([(x, z)] \cdot Q)$ **by** *simp*
ultimately have $\Psi \rhd (P \parallel ([(x, z)] \cdot Q)) \longmapsto \tau \prec (\!|\nu*(xvec1@xvec2)|\!)(([(x, z)] \cdot [(x, y)] \cdot P') \parallel ([(x, z)] \cdot [(x, y)] \cdot Q'))$
using *MeqM'* ‹$M'{=}M''$› ‹$N{=}N'$› ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_P \sharp* A_Q{}'$› ‹$A_Q{}' \sharp* \Psi$› ‹$A_Q{}' \sharp* P$› ‹$(xvec1@xvec2) \sharp* P$› ‹$A_P \sharp* M$› ‹$A_Q{}' \sharp* M''$›
by(*intro Comm1*) (*assumption* | *simp*)+
with‹$z \sharp \Psi$› **have** $\Psi \rhd (\!|\nu z|\!)(P \parallel ([(x, z)] \cdot Q)) \longmapsto \tau \prec (\!|\nu z|\!)((\!|\nu*(xvec1@xvec2)|\!)(([(x, z)] \cdot [(x, y)] \cdot P') \parallel ([(x, z)] \cdot [(x, y)] \cdot Q')))$
by(*intro Scope*) *auto*
moreover from ‹$x \sharp P$› ‹$z \sharp P$› ‹$z \sharp Q$› **have** $(\!|\nu z|\!)(P \parallel ([(x, z)] \cdot Q)) = (\!|\nu x|\!)([(x, z)] \cdot (P \parallel ([(x, z)] \cdot Q)))$
by(*subst alphaRes*[*of x*]) (*auto simp add*: *calc-atm fresh-left name-swap*)
with ‹$x \sharp P$› ‹$z \sharp P$› **have** $(\!|\nu z|\!)(P \parallel ([(x, z)] \cdot Q)) = (\!|\nu x|\!)(P \parallel Q)$
by(*simp add*: *eqvts*)
moreover from ‹$z \neq y$› ‹$x \neq z$› ‹$z \sharp P'$› ‹$z \sharp [(x, y)] \cdot Q'$› **have** $(\!|\nu z|\!)((\!|\nu*(xvec1@xvec2)|\!)(([(x, z)] \cdot [(x, y)] \cdot P') \parallel ([(x, z)] \cdot [(x, y)] \cdot Q'))) = (\!|\nu x|\!)([(x, z)] \cdot ((\!|\nu*(xvec1@xvec2)|\!)(([(x, z)] \cdot [(x, y)] \cdot P') \parallel ([(x, z)] \cdot [(x, y)] \cdot Q'))))$
by(*subst alphaRes*[*of x*]) (*auto simp add*: *resChainFresh fresh-left calc-atm name-swap*)
with ‹$x \sharp xvec1$› ‹$x \sharp xvec2$› ‹$z \sharp xvec1$› ‹$z \sharp xvec2$› **have** $(\!|\nu z|\!)((\!|\nu*(xvec1@xvec2)|\!)(([(x, z)] \cdot [(x, y)] \cdot P') \parallel ([(x, z)] \cdot [(x, y)] \cdot Q'))) = (\!|\nu x|\!)((\!|\nu*(xvec1@xvec2)|\!)(([(x, y)] \cdot P') \parallel ([(x, y)] \cdot Q')))$
by(*simp add*: *eqvts*)
moreover from ‹$x \sharp P'$› ‹$x \sharp Q'$› ‹$x \sharp xvec1$› ‹$x \sharp xvec2$› ‹$y \sharp xvec1$› ‹$y \sharp xvec2$›
have $(\!|\nu x|\!)((\!|\nu*(xvec1@xvec2)|\!)(([(x, y)] \cdot P') \parallel ([(x, y)] \cdot Q'))) = (\!|\nu y|\!)((\!|\nu*(xvec1@xvec2)|\!)(P' \parallel Q'))$
by(*subst alphaRes*[*of y*]) (*auto simp add*: *resChainFresh calc-atm eqvts fresh-left name-swap*)
ultimately have $\Psi \rhd (\!|\nu x|\!)(P \parallel Q) \longmapsto \tau \prec (\!|\nu y|\!)((\!|\nu*(xvec1@xvec2)|\!)(P' \parallel Q'))$
by *simp*
moreover from ‹$y \sharp \Psi$› ‹$(xvec1@xvec2) \sharp* \Psi$› ‹$xvec{=}xvec1@y\#xvec2$›
have $(\Psi, (\!|\nu y|\!)((\!|\nu*(xvec1@xvec2)|\!)(P' \parallel Q')), (\!|\nu*xvec|\!)(P' \parallel Q')) \in Rel$
by(*force intro*: *C3 simp add*: *resChainAppend*)
ultimately show *?case* **by** *blast*
next
case(*cRes Q'*)
have *QTrans*: $\Psi \otimes \Psi_P \rhd Q \longmapsto M'(\!|\nu*xvec|\!)\langle N\rangle \prec Q'$ **by** *fact*
with ‹$A_Q \sharp* Q$› ‹$A_Q \sharp* xvec$› ‹$xvec \sharp* M'$› ‹*distinct xvec*› **have** $A_Q \sharp* Q'$
by(*force dest*: *outputFreshChainDerivative*)

with ‹$extractFrame((\!|\nu x|\!)Q) = \langle A_Q, \Psi_Q\rangle$› **have** *FrxQ*: $(\!|\nu x|\!)(extractFrame\ Q) = \langle A_Q, \Psi_Q\rangle$ **by** *simp*

then obtain y $A_Q{}'$ **where** A: $A_Q = y\#A_Q{}'$ **by**(*cases* A_Q) *auto*
with ‹A_Q ♯* Ψ› ‹A_Q ♯* P› ‹A_Q ♯* P'› ‹A_Q ♯* Ψ_P› ‹A_Q ♯* Q› ‹A_Q ♯* *xvec*› ‹A_Q ♯* M'› ‹A_Q ♯* Q'›
have $A_Q{}'$ ♯* Ψ **and** $A_Q{}'$ ♯* P **and** $A_Q{}'$ ♯* Ψ_P **and** $A_Q{}'$ ♯* Q **and** A_Q ♯* *xvec* **and** A_Q ♯* Q'
and y ♯ Ψ **and** y ♯ P **and** y ♯ P' **and** y ♯ Ψ_P **and** y ♯ Q **and** y ♯ *xvec* **and** y ♯ M' **and** y ♯ Q'
and $A_Q{}'$ ♯* M'
by(*simp*)+
from A ‹A_P ♯* A_Q› **have** A_P ♯* $A_Q{}'$ **and** y ♯ A_P **by**(*simp add*: *fresh-star-list-cons*)+
from A ‹A_Q ♯* x› **have** $x \neq y$ **and** x ♯ $A_Q{}'$ **by**(*simp add*: *fresh-list-cons*)+

with A ‹*distinct* A_Q› **have** y ♯ $A_Q{}'$
by(*induct* $A_Q{}'$) (*auto simp add*: *fresh-list-nil fresh-list-cons*)

from ‹x ♯ P› ‹y ♯ P› ‹$x \neq y$› ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|N|\!) \prec P'$›
have $\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|([(y, x)] \cdot N)|\!) \prec [(y, x)] \cdot P'$
by(*intro inputAlpha*[**where** *xvec*=[y]]) (*auto simp add*: *calc-atm*)
then have $\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|([(x, y)] \cdot N)|\!) \prec [(x, y)] \cdot P'$
by(*simp add*: *name-swap*)
moreover note *FrP*
moreover from ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto M'(\!|\nu * xvec|\!)\langle N\rangle \prec Q'$› **have** $([(x, y)] \cdot (\Psi \otimes \Psi_P)) \rhd ([(x, y)] \cdot Q) \longmapsto ([(x, y)] \cdot (M'(\!|\nu * xvec|\!)\langle N\rangle \prec Q'))$
by(*rule semantics.eqvt*)
with ‹x ♯ Ψ› ‹y ♯ Ψ› ‹x ♯ Ψ_P› ‹y ♯ Ψ_P› ‹x ♯ M'› ‹y ♯ M'› ‹x ♯ *xvec*› ‹y ♯ *xvec*›
have $\Psi \otimes \Psi_P \rhd ([(x, y)] \cdot Q) \longmapsto M'(\!|\nu * xvec|\!)\langle([(x, y)] \cdot N)\rangle \prec ([(x, y)] \cdot Q')$
by(*simp add*: *eqvts*)
moreover from A ‹A_Q ♯* x› *FrxQ* **have** *FrQ*: *extractFrame*$([(x, y)] \cdot Q) = \langle A_Q{}', \Psi_Q\rangle$
by(*clarsimp simp add*: *alpha' eqvts frame.inject fresh-list-cons name-swap*)
moreover from ‹A_P ♯* Q› **have** $([(x, y)] \cdot A_P)$ ♯* $([(x, y)] \cdot Q)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹A_P ♯* x› ‹y ♯ A_P› **have** A_P ♯* $([(x, y)] \cdot Q)$ **by** *simp*
moreover from ‹$A_Q{}'$ ♯* Q› **have** $([(x, y)] \cdot A_Q{}')$ ♯* $([(x, y)] \cdot Q)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹x ♯ $A_Q{}'$› ‹y ♯ $A_Q{}'$› **have** $A_Q{}'$ ♯* $([(x, y)] \cdot Q)$ **by** *simp*
ultimately have $\Psi \rhd (P \parallel ([(x, y)] \cdot Q)) \longmapsto \tau \prec (\!|\nu * xvec|\!)([(x, y)] \cdot P') \parallel ([(x, y)] \cdot Q')$
using *MeqM'* ‹A_P ♯* Ψ› ‹A_P ♯* P› ‹A_P ♯* $A_Q{}'$› ‹$A_Q{}'$ ♯* Ψ› ‹$A_Q{}'$ ♯* P› ‹*xvec* ♯* P› ‹A_P ♯* M› ‹$A_Q{}'$ ♯* M'›
by(*intro Comm1*) (*assumption* | *simp*)+
with‹y ♯ Ψ› **have** $\Psi \rhd (\!|\nu y|\!)(P \parallel ([(x, y)] \cdot Q)) \longmapsto \tau \prec (\!|\nu y|\!)((\!|\nu * xvec|\!)(([(x, y)] \cdot P') \parallel ([(x, y)] \cdot Q')))$
by(*intro Scope*) *auto*
moreover from ‹x ♯ P› ‹y ♯ P› ‹y ♯ Q› **have** $(\!|\nu y|\!)(P \parallel ([(x, y)] \cdot Q)) = (\!|\nu x|\!)([(x, y)] \cdot (P \parallel ([(x, y)] \cdot Q)))$
by(*subst alphaRes*[*of* x]) (*auto simp add*: *calc-atm fresh-left name-swap*)

with ‹$x \sharp P$› ‹$y \sharp P$› **have** $(\!|\nu y|\!)(P \parallel ([(x, y)] \cdot Q)) = (\!|\nu x|\!)(P \parallel Q)$
by(*simp add*: *eqvts*)
moreover from ‹$y \sharp P'$› ‹$y \sharp Q'$› ‹$x \sharp xvec$› ‹$y \sharp xvec$› **have** $(\!|\nu y|\!)((\!|\nu{*}xvec|\!)(([(x, y)] \cdot P') \parallel ([(x, y)] \cdot Q'))) = (\!|\nu x|\!)((\!|\nu{*}xvec|\!)(P' \parallel Q'))$
by(*subst alphaRes*[*of y*]) (*auto simp add*: *resChainFresh calc-atm eqvts fresh-left name-swap*)
ultimately have $\Psi \rhd (\!|\nu x|\!)(P \parallel Q) \longmapsto \tau \prec (\!|\nu x|\!)((\!|\nu{*}xvec|\!)(P' \parallel Q'))$
by *simp*
moreover from ‹$x \sharp \Psi$› ‹$x \sharp P'$› ‹$xvec \sharp* \Psi$› **have** $(\Psi, (\!|\nu x|\!)((\!|\nu{*}xvec|\!)(P' \parallel Q')), (\!|\nu{*}xvec|\!)(P' \parallel (\!|\nu x|\!)\, Q')) \in Rel$
by(*rule C2*)
ultimately show *?case* **by** *blast*
qed
next
case(*cComm2* Ψ_Q M *xvec* N P' A_P Ψ_P K xQ' A_Q)
have *QTrans*: $\Psi \otimes \Psi_P \rhd (\!|\nu x|\!)\, Q \longmapsto K (\!|N|\!) \prec xQ'$ **and** *FrQ*: *extractFrame*$((\!|\nu x|\!)\, Q) = \langle A_Q, \Psi_Q \rangle$ **by** *fact+*
have *PTrans*: $\Psi \otimes \Psi_Q \rhd P \longmapsto M (\!|\nu{*}xvec|\!) \langle N \rangle \prec P'$ **and** *FrP*: *extractFrame* $P = \langle A_P, \Psi_P \rangle$ **by** *fact+*
from *PTrans* ‹$x \sharp P$› **have** $x \sharp N$ **and** $x \sharp P'$ **using** ‹$xvec \sharp* x$› ‹$xvec \sharp* M$› ‹*distinct xvec*›
by(*force intro*: *outputFreshDerivative*)+
have $x \sharp (\!|\nu x|\!)\, Q$ **by**(*simp add*: *abs-fresh*)
with *FrQ* ‹$A_Q \sharp* x$› **have** $x \sharp \Psi_Q$
by(*drule-tac extractFrameFresh*) *auto*
from ‹$x \sharp P$› *FrP* ‹$A_P \sharp* x$› **have** $x \sharp \Psi_P$
by(*drule-tac extractFrameFresh*) *auto*
from ‹$A_P \sharp* (\!|\nu x|\!)\, Q$› ‹$A_P \sharp* x$› **have** $A_P \sharp* Q$ **by** *simp*
from ‹$A_Q \sharp* (\!|\nu x|\!)\, Q$› ‹$A_Q \sharp* x$› **have** $A_Q \sharp* Q$ **by** *simp*
from ‹$xvec \sharp* x$› ‹$xvec \sharp* (\!|\nu x|\!)\, Q$› **have** $xvec \sharp* Q$ **by** *simp*

from ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto M (\!|\nu{*}xvec|\!) \langle N \rangle \prec P'$› **have** *PResTrans*: $\Psi \otimes \Psi_Q \rhd P \longmapsto ROut\ M\ ((\!|\nu{*}xvec|\!)N \prec' P')$
by(*simp add*: *residualInject*)

from *PResTrans FrP* ‹*distinct* A_P› ‹$x \sharp P$› ‹$A_Q \sharp* P$› ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* \Psi_Q$› ‹$A_P \sharp* x$› ‹$A_P \sharp* A_Q$› ‹$A_P \sharp* P$› ‹$A_P \sharp* M$› ‹$xvec \sharp* M$› ‹*distinct xvec*›
obtain M' **where** $(\Psi \otimes \Psi_Q) \otimes \Psi_P \vdash M \leftrightarrow M'$ **and** $x \sharp M'$ **and** $A_Q \sharp* M'$
by(*elim outputObtainPrefix*[**where** $B{=}x\#A_Q$]) (*assumption* | *force simp add*: *fresh-star-list-cons*)+
then have *MeqM'*: $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow M'$
by(*metis statEqEnt Associativity Commutativity Composition*)
with ‹$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K$› **have** $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash K \leftrightarrow M'$
by(*blast intro*: *chanEqTrans chanEqSym*)
then have $(\Psi \otimes \Psi_P) \otimes \Psi_Q \vdash K \leftrightarrow M'$
by(*metis statEqEnt Associativity Commutativity Composition*)
with *QTrans FrQ* ‹*distinct* A_Q› ‹$A_Q \sharp* \Psi$› ‹$A_Q \sharp* \Psi_P$› ‹$A_Q \sharp* ((\!|\nu x|\!)\, Q)$› ‹$A_Q \sharp* K$› ‹$A_Q \sharp* M'$›
have $\Psi \otimes \Psi_P \rhd (\!|\nu x|\!)\, Q \longmapsto M' (\!|N|\!) \prec xQ'$ **by**(*force intro*: *inputRenameSubject*)

moreover from ‹$x \sharp \Psi$› ‹$x \sharp \Psi_P$› **have** $x \sharp \Psi \otimes \Psi_P$ **by** *force*
moreover note ‹$x \sharp M'$› ‹$x \sharp N$›
moreover from *QTrans* ‹$x \sharp N$› **have** $x \sharp xQ'$ **by**(*force dest*: *inputFreshDerivative simp add*: *abs-fresh*)
ultimately show *?case*
proof(*induct rule*: *resInputCases*)
case(*cRes* Q')
have *QTrans*: $\Psi \otimes \Psi_P \rhd Q \longmapsto M'(\!|N|\!) \prec Q'$ **by** *fact*
with ‹$A_Q \sharp* Q$› ‹$A_Q \sharp* N$› **have** $A_Q \sharp* Q'$
by(*elim inputFreshChainDerivative*)

with ‹*extractFrame*$((\!|\nu x|\!)\, Q) = \langle A_Q, \Psi_Q\rangle$› **have** *FrxQ*: $(\!|\nu x|\!)(\mathit{extractFrame}\ Q) = \langle A_Q, \Psi_Q\rangle$ **by** *simp*
then obtain $y\ A_Q'$ **where** A: $A_Q = y\#A_Q'$ **by**(*cases* A_Q) *auto*
with ‹$A_Q \sharp* \Psi$› ‹$A_Q \sharp* P$› ‹$A_Q \sharp* P'$› ‹$A_Q \sharp* \Psi_P$› ‹$A_Q \sharp* Q$› ‹$A_Q \sharp* xvec$› ‹$A_Q \sharp* M'$› ‹$A_Q \sharp* Q'$› ‹$A_Q \sharp* N$›
have $A_Q' \sharp* \Psi$ **and** $A_Q' \sharp* P$ **and** $A_Q' \sharp* \Psi_P$ **and** $A_Q' \sharp* Q$ **and** $A_Q \sharp* xvec$ **and** $A_Q \sharp* Q'$
and $y \sharp \Psi$ **and** $y \sharp P$ **and** $y \sharp P'$ **and** $y \sharp \Psi_P$ **and** $y \sharp Q$ **and** $y \sharp xvec$ **and** $y \sharp M'$ **and** $y \sharp Q'$ **and** $y \sharp N$
and $A_Q' \sharp* M'$
by(*simp*)+
from *A* ‹$A_P \sharp* A_Q$› **have** $A_P \sharp* A_Q'$ **and** $y \sharp A_P$ **by**(*simp add*: *fresh-star-list-cons*)+
from *A* ‹$A_Q \sharp* x$› **have** $x \neq y$ **and** $x \sharp A_Q'$ **by**(*simp add*: *fresh-list-cons*)+

with *A* ‹*distinct* A_Q› **have** $y \sharp A_Q'$
by(*induct* A_Q') (*auto simp add*: *fresh-list-nil fresh-list-cons*)

note *PTrans FrP*
moreover from ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto M'(\!|N|\!) \prec Q'$› **have** $([(x, y)] \cdot (\Psi \otimes \Psi_P)) \rhd ([(x, y)] \cdot Q) \longmapsto ([(x, y)] \cdot (M'(\!|N|\!) \prec Q'))$
by(*rule semantics.eqvt*)
with ‹$x \sharp \Psi$› ‹$y \sharp \Psi$› ‹$x \sharp \Psi_P$› ‹$y \sharp \Psi_P$› ‹$x \sharp M'$› ‹$y \sharp M'$› ‹$x \sharp N$› ‹$y \sharp N$›
have $\Psi \otimes \Psi_P \rhd ([(x, y)] \cdot Q) \longmapsto M'(\!|N|\!) \prec ([(x, y)] \cdot Q')$
by(*simp add*: *eqvts*)
moreover from *A* ‹$A_Q \sharp* x$› *FrxQ* **have** *FrQ*: *extractFrame*$([(x, y)] \cdot Q) = \langle A_Q', \Psi_Q\rangle$ **and** $y \sharp$ *extractFrame Q*
by(*clarsimp simp add*: *alpha' eqvts frame.inject fresh-list-cons name-swap*)+
moreover from ‹$A_P \sharp* Q$› **have** $([(x, y)] \cdot A_P) \sharp* ([(x, y)] \cdot Q)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$A_P \sharp* x$› ‹$y \sharp A_P$› **have** $A_P \sharp* ([(x, y)] \cdot Q)$ **by** *simp*
moreover from ‹$A_Q' \sharp* Q$› **have** $([(x, y)] \cdot A_Q') \sharp* ([(x, y)] \cdot Q)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$x \sharp A_Q'$› ‹$y \sharp A_Q'$› **have** $A_Q' \sharp* ([(x, y)] \cdot Q)$ **by** *simp*
moreover from ‹$xvec \sharp* Q$› **have** $([(x, y)] \cdot xvec) \sharp* ([(x, y)] \cdot Q)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$xvec \sharp* x$› ‹$y \sharp xvec$› **have** $xvec \sharp* ([(x, y)] \cdot Q)$ **by** *simp*
ultimately have $\Psi \rhd (P \parallel ([(x, y)] \cdot Q)) \longmapsto \tau \prec (\!|\nu* xvec|\!)(P' \parallel ([(x, y)] \cdot$

Q'))
using *MeqM'* ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_P \sharp* A_Q{}'$› ‹$A_Q{}' \sharp* \Psi$› ‹$A_Q{}' \sharp* P$› ‹$A_P \sharp* M$› ‹$A_Q{}' \sharp* M'$›
by(*intro Comm2*) (*assumption* | *simp*)+
with‹$y \sharp \Psi$› **have** $\Psi \rhd (\!|\nu y|\!)(P \parallel ([(x,\, y)] \cdot Q)) \longmapsto\tau \prec (\!|\nu y|\!)((\!|\nu{*}xvec|\!)(P' \parallel ([(x,\, y)] \cdot Q')))$
by(*intro Scope*) *auto*
moreover from ‹$x \sharp P$› ‹$y \sharp P$› ‹$y \sharp Q$› **have** $(\!|\nu y|\!)(P \parallel ([(x,\, y)] \cdot Q)) = (\!|\nu x|\!)([(x,\, y)] \cdot (P \parallel ([(x,\, y)] \cdot Q)))$
by(*subst alphaRes*[*of x*]) (*auto simp add*: *calc-atm fresh-left name-swap*)
with ‹$x \sharp P$› ‹$y \sharp P$› **have** $(\!|\nu y|\!)(P \parallel ([(x,\, y)] \cdot Q)) = (\!|\nu x|\!)(P \parallel Q)$
by(*simp add*: *eqvts*)
moreover from ‹$x \sharp P'$› ‹$y \sharp P'$› ‹$y \sharp Q'$› ‹$xvec \sharp* x$› ‹$y \sharp xvec$› **have** $(\!|\nu y|\!)((\!|\nu{*}xvec|\!)(P' \parallel ([(x,\, y)] \cdot Q'))) = (\!|\nu x|\!)((\!|\nu{*}xvec|\!)(P' \parallel Q'))$
by(*subst alphaRes*[*of y*]) (*auto simp add*: *resChainFresh calc-atm eqvts fresh-left name-swap*)
ultimately have $\Psi \rhd (\!|\nu x|\!)(P \parallel Q) \longmapsto\tau \prec (\!|\nu x|\!)((\!|\nu{*}xvec|\!)(P' \parallel Q'))$
by *simp*
moreover from ‹$x \sharp \Psi$› ‹$x \sharp P'$› ‹$xvec \sharp* \Psi$› **have** $(\Psi,\, (\!|\nu x|\!)((\!|\nu{*}xvec|\!)(P' \parallel Q')),\, (\!|\nu{*}xvec|\!)(P' \parallel (\!|\nu x|\!)\, Q')) \in Rel$
by(*rule C2*)
ultimately show *?case* **by** *blast*
qed
next
case(*cBrMerge* Ψ_Q M N P' A_P Ψ_P xQ' A_Q)
have *QTrans*: $\Psi \otimes \Psi_P \rhd (\!|\nu x|\!)\, Q \longmapsto ¿M(\!|N|\!) \prec xQ'$ **and** *FrQ*: *extractFrame*$((\!|\nu x|\!)\, Q) = \langle A_Q,\, \Psi_Q\rangle$ **by** *fact+*
have *PTrans*: $\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$ **and** *FrP*: *extractFrame* $P = \langle A_P,\, \Psi_P\rangle$ **by** *fact+*
from ‹$x \sharp \alpha$› ‹$\alpha = ¿M(\!|N|\!)$› **have** $x \sharp M$ **and** $x \sharp N$ **by** *simp+*
from ‹$x \sharp P' \parallel xQ'$› **have** $x \sharp xQ'$ **by** *simp*
from *FrP* ‹$A_P \sharp* x$› ‹$x \sharp P$› **have** $x \sharp \Psi_P$
by(*drule-tac extractFrameFresh*) *auto*
from *PTrans* ‹$x \sharp P$› ‹$x \sharp N$› **have** $x \sharp P'$
by(*rule brinputFreshDerivative*)
have $x \sharp (\!|\nu x|\!)\, Q$ **by**(*simp add*: *abs-fresh*)
with *FrQ* ‹$A_Q \sharp* x$› **have** $x \sharp \Psi_Q$
by(*drule-tac extractFrameFresh*) *auto*
from ‹$A_P \sharp* (\!|\nu x|\!)\, Q$› ‹$A_P \sharp* x$› **have** $A_P \sharp* Q$ **by** *simp*
from ‹$A_Q \sharp* (\!|\nu x|\!)\, Q$› ‹$A_Q \sharp* x$› **have** $A_Q \sharp* Q$ **by** *simp*
from ‹$x \sharp \Psi$› ‹$x \sharp \Psi_P$›
have $x \sharp (\Psi \otimes \Psi_P)$ **by** *force*
from ‹$\Psi \otimes \Psi_P \rhd (\!|\nu x|\!)\, Q \longmapsto ¿M(\!|N|\!) \prec xQ'$› ‹$x \sharp (\Psi \otimes \Psi_P)$› ‹$x \sharp M$› ‹$x \sharp N$› ‹$x \sharp xQ'$›
show *?case*
proof(*induct rule*: *resBrInputCases*)
case(*cRes* Q')
have *QTrans*: $\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$ **by** *fact*
with ‹$A_Q \sharp* Q$› ‹$A_Q \sharp* N$› **have** $A_Q \sharp* Q'$

by(*elim brinputFreshChainDerivative*)

with ‹*extractFrame*(⟪νx⟫ Q) = $\langle A_Q, \Psi_Q\rangle$› **have** *FrxQ*: ⟪$\nu x$⟫(*extractFrame* Q) = $\langle A_Q, \Psi_Q\rangle$ **by** *simp*
then obtain y $A_Q{'}$ **where** *A*: $A_Q = y\#A_Q{'}$ **by**(*cases* A_Q) *auto*
with ‹A_Q ♯* Ψ› ‹A_Q ♯* P› ‹A_Q ♯* P'› ‹A_Q ♯* Ψ_P› ‹A_Q ♯* Q› ‹A_Q ♯* M› ‹A_Q ♯* Q'› ‹A_Q ♯* N›
have $A_Q{'}$ ♯* Ψ **and** $A_Q{'}$ ♯* P **and** $A_Q{'}$ ♯* Ψ_P **and** $A_Q{'}$ ♯* Q **and** A_Q ♯* Q'
and y ♯ Ψ **and** y ♯ P **and** y ♯ P' **and** y ♯ Ψ_P **and** y ♯ Q **and** y ♯ M **and** y ♯ Q' **and** y ♯ N
and $A_Q{'}$ ♯* M
by(*simp*)+
from A ‹A_P ♯* A_Q› **have** A_P ♯* $A_Q{'}$ **and** y ♯ A_P **by**(*simp add*: *fresh-star-list-cons*)+
from A ‹A_Q ♯* x› **have** $x \neq y$ **and** x ♯ $A_Q{'}$ **by**(*simp add*: *fresh-list-cons*)+

with A ‹*distinct* A_Q› **have** y ♯ $A_Q{'}$
by(*induct* $A_Q{'}$) (*auto simp add*: *fresh-list-nil fresh-list-cons*)

note *PTrans FrP*
moreover from ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M⟪N⟫ \prec Q'$› **have** $([(x, y)] \cdot (\Psi \otimes \Psi_P)) \rhd ([(x, y)] \cdot Q) \longmapsto ([(x, y)] \cdot (¿M⟪N⟫ \prec Q'))$
by(*rule semantics.eqvt*)
with ‹x ♯ Ψ› ‹y ♯ Ψ› ‹x ♯ Ψ_P› ‹y ♯ Ψ_P› ‹x ♯ M› ‹y ♯ M› ‹x ♯ N› ‹y ♯ N›
have $\Psi \otimes \Psi_P \rhd ([(x, y)] \cdot Q) \longmapsto ¿M⟪N⟫ \prec ([(x, y)] \cdot Q')$
by(*simp add*: *eqvts*)
moreover from A ‹A_Q ♯* x› *FrxQ* **have** *FrQ*: *extractFrame*$([(x, y)] \cdot Q) = \langle A_Q{'}, \Psi_Q\rangle$ **and** y ♯ *extractFrame* Q
by(*clarsimp simp add*: *alpha'* *eqvts frame.inject fresh-list-cons name-swap*)+
moreover from ‹A_P ♯* Q› **have** $([(x, y)] \cdot A_P)$ ♯* $([(x, y)] \cdot Q)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹A_P ♯* x› ‹y ♯ A_P› **have** A_P ♯* $([(x, y)] \cdot Q)$ **by** *simp*
moreover from ‹$A_Q{'}$ ♯* Q› **have** $([(x, y)] \cdot A_Q{'})$ ♯* $([(x, y)] \cdot Q)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹x ♯ $A_Q{'}$› ‹y ♯ $A_Q{'}$› **have** $A_Q{'}$ ♯* $([(x, y)] \cdot Q)$ **by** *simp*
ultimately have $\Psi \rhd (P \parallel ([(x, y)] \cdot Q)) \longmapsto ¿M⟪N⟫ \prec (P' \parallel ([(x, y)] \cdot Q'))$
using ‹A_P ♯* Ψ› ‹A_P ♯* P› ‹A_P ♯* $A_Q{'}$› ‹$A_Q{'}$ ♯* Ψ› ‹$A_Q{'}$ ♯* P› ‹A_P ♯* M› ‹$A_Q{'}$ ♯* M›
by(*intro BrMerge*)
with‹y ♯ Ψ› ‹y ♯ M› ‹y ♯ N› **have** $\Psi \rhd$ ⟪νy⟫$(P \parallel ([(x, y)] \cdot Q)) \longmapsto ¿M⟪N⟫ \prec$ ⟪νy⟫$(P' \parallel ([(x, y)] \cdot Q'))$
by(*intro Scope*) *auto*
moreover from ‹x ♯ P› ‹y ♯ P› ‹y ♯ Q› **have** ⟪νy⟫$(P \parallel ([(x, y)] \cdot Q)) =$ ⟪νx⟫$([(x, y)] \cdot (P \parallel ([(x, y)] \cdot Q)))$
by(*subst alphaRes*[*of* x]) (*auto simp add*: *calc-atm fresh-left name-swap*)
moreover with ‹x ♯ P› ‹y ♯ P› **have** ⟪νy⟫$(P \parallel ([(x, y)] \cdot Q)) =$ ⟪νx⟫$(P \parallel Q)$
by(*simp add*: *eqvts*)
moreover from ‹x ♯ P'› ‹y ♯ P'› ‹y ♯ Q'› **have** ⟪νy⟫$(P' \parallel ([(x, y)] \cdot Q'))$

$= (\!|\nu x|\!)(P' \parallel Q')$
by(*subst alphaRes*[*of y*]) (*auto simp add*: *resChainFresh calc-atm eqvts fresh-left name-swap*)
ultimately have *finTrans*: $\Psi \rhd (\!|\nu x|\!)(P \parallel Q) \longmapsto ¡M(\!|N|\!) \prec (\!|\nu x|\!)(P' \parallel Q')$
by *simp*
from ‹$x \sharp \Psi$› ‹$x \sharp P'$› **have** $[x] \sharp* P'$ **and** $[x] \sharp* \Psi$ **by** *simp+*
then have $(\Psi, (\!|\nu*[x]|\!)(P' \parallel Q'), (P' \parallel ((\!|\nu*[x]|\!) Q'))) \in Rel$
by(*rule C4*)
then have $(\Psi, (\!|\nu x|\!)(P' \parallel Q'), (P' \parallel (\!|\nu x|\!) Q')) \in Rel$
by *simp*
with *finTrans* **show** *?case* **by** *blast*
qed
next
case(*cBrComm1* Ψ_Q M N P' A_P Ψ_P *xvec* xQ' A_Q)
have *QTrans*: $\Psi \otimes \Psi_P \rhd (\!|\nu x|\!) Q \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec xQ'$ **and** *FrQ*: *extractFrame*$((\!|\nu x|\!) Q) = \langle A_Q, \Psi_Q\rangle$ **by** *fact+*
have *PTrans*: $\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$ **and** *FrP*: *extractFrame* $P = \langle A_P, \Psi_P\rangle$ **by** *fact+*
from ‹*bn* $\alpha \sharp* x$› ‹$¡M(\!|\nu*xvec|\!)\langle N\rangle = \alpha$› **have** $x \sharp$ *xvec* **by** *force*
have $x \sharp (\!|\nu x|\!) Q$ **by**(*simp add*: *abs-fresh*)
with *QTrans* **have** $x \sharp N$ **and** $x \sharp xQ'$ **using** ‹$x \sharp$ *xvec*› ‹*xvec* $\sharp* M$› ‹*distinct xvec*›
by(*force intro*: *broutputFreshDerivative*)+
from *PTrans* ‹$x \sharp P$› ‹$x \sharp N$› **have** $x \sharp P'$ **by**(*rule brinputFreshDerivative*)
from ‹$x \sharp (\!|\nu x|\!) Q$› *FrQ* ‹$A_Q \sharp* x$› **have** $x \sharp \Psi_Q$
by(*drule-tac extractFrameFresh*) *auto*
from ‹$x \sharp P$› *FrP* ‹$A_P \sharp* x$› **have** $x \sharp \Psi_P$
by(*drule-tac extractFrameFresh*) *auto*
from ‹$A_P \sharp* (\!|\nu x|\!) Q$› ‹$A_P \sharp* x$› **have** $A_P \sharp* Q$ **by** *simp*
from ‹$A_Q \sharp* (\!|\nu x|\!) Q$› ‹$A_Q \sharp* x$› **have** $A_Q \sharp* Q$ **by** *simp*
note *QTrans*
moreover from ‹$x \sharp \Psi$› ‹$x \sharp \Psi_P$› **have** $x \sharp \Psi \otimes \Psi_P$ **by** *force*
moreover from ‹$x \sharp$ *xvec*› **have** *xvec* $\sharp* x$ **by** *simp*
from ‹$x \sharp \alpha$› ‹$¡M(\!|\nu*xvec|\!)\langle N\rangle = \alpha$› **have** $x \sharp ¡M(\!|\nu*xvec|\!)\langle N\rangle$ **by** *simp*
moreover note ‹$x \sharp xQ'$›

moreover from ‹*xvec* $\sharp* \Psi$› ‹*xvec* $\sharp* \Psi_P$› **have** *xvec* $\sharp* (\Psi \otimes \Psi_P)$ **by** *force*
moreover from ‹*xvec* $\sharp* (\!|\nu x|\!) Q$› ‹$x \sharp$ *xvec*› **have** *xvec* $\sharp* Q$ **by** *simp*
moreover note ‹*xvec* $\sharp* M$›

moreover from ‹*xvec* $\sharp* \Psi$› ‹*xvec* $\sharp* \Psi_P$› **have** $bn(¡M(\!|\nu*xvec|\!)\langle N\rangle) \sharp* (\Psi \otimes \Psi_P)$ **by** *force*
moreover from ‹*xvec* $\sharp* (\!|\nu x|\!) Q$› ‹$x \sharp$ *xvec*› **have** $bn(¡M(\!|\nu*xvec|\!)\langle N\rangle) \sharp* Q$ **by** *simp*
moreover from ‹*xvec* $\sharp* P$› **have** $bn(¡M(\!|\nu*xvec|\!)\langle N\rangle) \sharp* P$ **by** *simp*
from ‹*xvec* $\sharp* \Psi$› **have** $bn(¡M(\!|\nu*xvec|\!)\langle N\rangle) \sharp* \Psi$ **by** *simp*
from ‹$A_Q \sharp*$ *xvec*› ‹$A_Q \sharp* M$› ‹$A_Q \sharp* N$› **have** $A_Q \sharp* (¡M(\!|\nu*xvec|\!)\langle N\rangle)$ **by** *simp*
have $object(¡M(\!|\nu*xvec|\!)\langle N\rangle) = Some\ N$ **by** *simp*

have $bn(¡M(\!|\nu{*}xvec|\!)\langle N\rangle) = xvec$ **by** *simp*
have $subject(¡M(\!|\nu{*}xvec|\!)\langle N\rangle) = Some\ M$ **by** *simp*
from ‹$xvec \sharp* M$› **have** $bn(¡M(\!|\nu{*}xvec|\!)\langle N\rangle) \sharp* subject(¡M(\!|\nu{*}xvec|\!)\langle N\rangle)$ **by** *simp*
ultimately show *?case*
using ‹$x \sharp ¡M(\!|\nu{*}xvec|\!)\langle N\rangle$› ‹$bn(¡M(\!|\nu{*}xvec|\!)\langle N\rangle) \sharp* P$› ‹$bn(¡M(\!|\nu{*}xvec|\!)\langle N\rangle) \sharp* \Psi$› ‹$object(¡M(\!|\nu{*}xvec|\!)\langle N\rangle) = Some\ N$›
‹$bn(¡M(\!|\nu{*}xvec|\!)\langle N\rangle) = xvec$› ‹$subject(¡M(\!|\nu{*}xvec|\!)\langle N\rangle) = Some\ M$› ‹$A_Q \sharp* (¡M(\!|\nu{*}xvec|\!)\langle N\rangle)$›
proof(*induct rule*: *resBrOutputCases'*)
case(*cBrOpen M'' xvec1 xvec2 y N' Q'*)
then have *Eq*: $¡M(\!|\nu{*}xvec|\!)\langle N\rangle = ¡M''(\!|\nu{*}(xvec1\ @\ y\ \#\ xvec2)|\!)\langle N'\rangle$ **by** *simp*
from ‹$x \sharp ¡M(\!|\nu{*}xvec|\!)\langle N\rangle$› *Eq* **have** $x \sharp xvec1$ **and** $x \neq y$ **and** $x \sharp xvec2$ **and** $x \sharp M''$
by *simp+*
from ‹$bn(¡M(\!|\nu{*}xvec|\!)\langle N\rangle) \sharp* P$› *Eq* **have** $(xvec1@xvec2) \sharp* P$ **and** $y \sharp P$ **by** *simp+*
from ‹$A_Q \sharp* (¡M(\!|\nu{*}xvec|\!)\langle N\rangle)$› *Eq* **have** $(xvec1@xvec2) \sharp* A_Q$ **and** $y \sharp A_Q$ **and** $A_Q \sharp* M''$ **by** *simp+*
from ‹$bn(¡M(\!|\nu{*}xvec|\!)\langle N\rangle) \sharp* \Psi$› *Eq* **have** $(xvec1@xvec2) \sharp* \Psi$ **and** $y \sharp \Psi$ **by** *simp+*
from *Eq* **have** $N = N'$ **and** $xvec = xvec1@y\#xvec2$ **and** $M = M''$ **by**(*simp add*: *action.inject*)+
from ‹$x \sharp P$› ‹$y \sharp P$› ‹$x \neq y$› ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$›
have $\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|([(y,\ x)] \cdot N)|\!) \prec ([(y,\ x)] \cdot P')$
by(*intro brinputAlpha*[**where** *xvec*=[*y*]]) (*auto simp add*: *calc-atm*)
then have *PTrans*: $\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|([(x,\ y)] \cdot N)|\!) \prec ([(x,\ y)] \cdot P')$
by(*simp add*: *name-swap*)

from ‹$\Psi \otimes \Psi_P \rhd [(x,\ y)] \cdot Q \longmapsto ¡M''(\!|\nu{*}(xvec1\ @\ xvec2)|\!)\langle N'\rangle \prec Q'$›
have $[(x,\ y)] \cdot (\Psi \otimes \Psi_P \rhd [(x,\ y)] \cdot Q \longmapsto ¡M''(\!|\nu{*}(xvec1\ @\ xvec2)|\!)\langle N'\rangle \prec Q')$
by *simp*
then have $[(x,\ y)] \cdot (\Psi \otimes \Psi_P) \rhd ([(x,\ y)] \cdot ([(x,\ y)] \cdot Q)) \longmapsto ¡([(x,\ y)] \cdot M'')(\!|\nu{*}([(x,\ y)] \cdot (xvec1\ @\ xvec2))|\!)\langle([(x,\ y)] \cdot N')\rangle \prec [(x,\ y)] \cdot Q'$
by(*simp add*: *eqvts*)
with ‹$x \sharp (\Psi \otimes \Psi_P)$› ‹$y \sharp (\Psi \otimes \Psi_P)$› ‹$x \sharp xvec1$› ‹$y \sharp xvec1$› ‹$x \sharp xvec2$› ‹$y \sharp xvec2$› ‹$x \sharp M''$› ‹$y \sharp M''$›
have *QTrans*: $\Psi \otimes \Psi_P \rhd Q \longmapsto ¡M''(\!|\nu{*}(xvec1\ @\ xvec2)|\!)\langle([(x,\ y)] \cdot N')\rangle \prec [(x,\ y)] \cdot Q'$
by *simp*
with ‹$A_Q \sharp* x$› ‹$y \sharp A_Q$› ‹*distinct xvec1*› ‹*distinct xvec2*› ‹$xvec1 \sharp* xvec2$› ‹$xvec1 \sharp* M''$› ‹$xvec2 \sharp* M''$›
‹$(xvec1@xvec2) \sharp* A_Q$›
have $A_Q \sharp* ([(x,\ y)] \cdot Q')$ **using** ‹$A_Q \sharp* Q$›
by(*elim broutputFreshChainDerivative*(*2*)) (*assumption* | *simp*)+

from ‹$extractFrame(\!|\nu x|\!)\ Q) = \langle A_Q,\ \Psi_Q\rangle$› **have** *FrxQ*: $(\!|\nu x|\!)(extractFrame$

$Q) = \langle A_Q, \Psi_Q\rangle$ **by** *simp*
then obtain $z\ A_Q'$ **where** A: $A_Q = z\#A_Q'$ **by**(*cases* A_Q) *auto*
with $\langle A_Q \sharp* \Psi\rangle$ $\langle A_Q \sharp* P\rangle$ $\langle A_Q \sharp* P'\rangle$ $\langle A_Q \sharp* \Psi_P\rangle$ $\langle A_Q \sharp* Q\rangle$ $\langle(xvec1@xvec2) \sharp* A_Q\rangle$ $\langle A_Q \sharp* M''\rangle$ $\langle A_Q \sharp* ([(x, y)] \cdot Q')\rangle$ $\langle y \sharp A_Q\rangle$ $\langle A_Q \sharp* N\rangle$
have $A_Q' \sharp* \Psi$ **and** $A_Q' \sharp* P$ **and** $A_Q' \sharp* \Psi_P$ **and** $A_Q' \sharp* Q$
and $z \sharp \Psi$ **and** $z \sharp P$ **and** $z \sharp P'$ **and** $z \sharp \Psi_P$ **and** $z \sharp Q$ **and** $z \sharp xvec1$ **and** $z \sharp xvec2$
and $z \sharp M''$ **and** $z \sharp ([(x, y)] \cdot Q')$ **and** $A_Q' \sharp* M''$ **and** $z \neq y$ **and** $z \sharp (xvec1@xvec2)$
by *auto*
from A $\langle A_P \sharp* A_Q\rangle$ **have** $A_P \sharp* A_Q'$ **and** $z \sharp A_P$ **by** *simp+*
from A $\langle A_Q \sharp* x\rangle$ **have** $x \neq z$ **and** $x \sharp A_Q'$ **by** *simp+*

from $\langle A_Q \sharp* (¡M\langle\langle\nu*xvec\rangle\rangle\langle N\rangle)\rangle$ A **have** $z \sharp M$ **and** $z \sharp xvec$ **and** $z \sharp N$ **by** *simp+*

from $\langle y \in supp\ N'\rangle$ $\langle N = N'\rangle$ **have** $y \in supp\ N$ **by** *simp*
then have $x \in supp\ ([(x, y)] \cdot N)$
by (*rule swap-supp*)
then have *zsupp*: $z \in supp\ ([(x, z)] \cdot [(x, y)] \cdot N)$
by (*rule swap-supp'*)

from $\langle distinct\ A_Q\rangle$ A **have** $z \sharp A_Q'$
by(*induct* A_Q') (*auto simp add*: *fresh-list-nil fresh-list-cons*)
from *PTrans* $\langle x \sharp P\rangle$ $\langle z \sharp P\rangle$ $\langle x \neq z\rangle$ **have** $\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M\langle\langle([(x, z)] \cdot [(x, y)] \cdot N)\rangle\rangle \prec ([(x, z)] \cdot [(x, y)] \cdot P')$
by(*elim brinputAlpha*[**where** *xvec*=[*x*]]) (*auto simp add*: *calc-atm*)
moreover note *FrP*
moreover from *QTrans* **have** $([(x, z)] \cdot (\Psi \otimes \Psi_P)) \rhd ([(x, z)] \cdot Q) \longmapsto([(x, z)] \cdot (¡M''\langle\langle\nu*(xvec1@xvec2)\rangle\rangle\langle([(x, y)] \cdot N')\rangle \prec ([(x, y)] \cdot Q')))$
by(*rule semantics.eqvt*)
with $\langle x \sharp \Psi\rangle$ $\langle z \sharp \Psi\rangle$ $\langle x \sharp \Psi_P\rangle$ $\langle z \sharp \Psi_P\rangle$ $\langle x \sharp M''\rangle$ $\langle z \sharp M''\rangle$ $\langle x \sharp xvec1\rangle$ $\langle x \sharp xvec2\rangle$ $\langle z \sharp xvec1\rangle$ $\langle z \sharp xvec2\rangle$
have $\Psi \otimes \Psi_P \rhd ([(x, z)] \cdot Q) \longmapsto ¡M''\langle\langle\nu*(xvec1@xvec2)\rangle\rangle\langle([(x, z)] \cdot [(x, y)] \cdot N')\rangle \prec ([(x, z)] \cdot [(x, y)] \cdot Q')$
by(*simp add*: *eqvts*)
moreover from A $\langle A_Q \sharp* x\rangle$ *FrxQ* **have** *extractFrame*$([(x, z)] \cdot Q) = \langle A_Q', \Psi_Q\rangle$
by(*clarsimp simp add*: *alpha' eqvts frame.inject fresh-list-cons name-swap*)
moreover from $\langle A_P \sharp* Q\rangle$ **have** $([(x, z)] \cdot A_P) \sharp* ([(x, z)] \cdot Q)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with $\langle A_P \sharp* x\rangle$ $\langle z \sharp A_P\rangle$ **have** $A_P \sharp* ([(x, z)] \cdot Q)$ **by** *simp*
moreover from $\langle A_Q' \sharp* Q\rangle$ **have** $([(x, z)] \cdot A_Q') \sharp* ([(x, z)] \cdot Q)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with $\langle x \sharp A_Q'\rangle$ $\langle z \sharp A_Q'\rangle$ **have** $A_Q' \sharp* ([(x, z)] \cdot Q)$ **by** *simp*
ultimately have $\Psi \rhd (P \parallel ([(x, z)] \cdot Q)) \longmapsto ¡M\langle\langle\nu*(xvec1@xvec2)\rangle\rangle\langle([(x, z)] \cdot [(x, y)] \cdot N)\rangle \prec (([(x, z)] \cdot [(x, y)] \cdot P') \parallel ([(x, z)] \cdot [(x, y)] \cdot Q'))$
using $\langle M{=}M''\rangle$ $\langle N{=}N'\rangle$ $\langle A_P \sharp* \Psi\rangle$ $\langle A_P \sharp* P\rangle$ $\langle A_P \sharp* A_Q'\rangle$ $\langle A_Q' \sharp* \Psi\rangle$ $\langle A_Q' \sharp* P\rangle$ $\langle(xvec1@xvec2) \sharp* P\rangle$ $\langle A_P \sharp* M\rangle$ $\langle A_Q' \sharp* M''\rangle$

by(*intro BrComm1*) (*assumption* | *simp*)+
then have *permTrans*: $\Psi \rhd (\!|\nu z|\!)(P \parallel ([(x, z)] \cdot Q)) \longmapsto ¡M (\!|\nu*(xvec1 @ z\#xvec2)|\!) \langle([(x, z)] \cdot [(x, y)] \cdot N)\rangle \prec (([(x, z)] \cdot [(x, y)] \cdot P') \parallel ([(x, z)] \cdot [(x, y)] \cdot Q'))$
using *zsupp* ‹$z \sharp \Psi$› ‹$z \sharp M$› ‹$z \sharp xvec1$› ‹$z \sharp xvec2$›
by(*rule BrOpen*)

from ‹$x \sharp P$› ‹$z \sharp P$› ‹$z \sharp Q$› **have** $(\!|\nu z|\!)(P \parallel ([(x, z)] \cdot Q)) = (\!|\nu x|\!)([(x, z)] \cdot (P \parallel ([(x, z)] \cdot Q)))$
by(*subst alphaRes*[*of x*]) (*auto simp add*: *calc-atm fresh-left name-swap*)
with ‹$x \sharp P$› ‹$z \sharp P$› **have** $(\!|\nu z|\!)(P \parallel ([(x, z)] \cdot Q)) = (\!|\nu x|\!)(P \parallel Q)$
by(*simp add*: *eqvts*)
with *permTrans* **have** *permTrans2*: $\Psi \rhd (\!|\nu x|\!)(P \parallel Q) \longmapsto ¡M (\!|\nu*(xvec1 @ z\#xvec2)|\!) \langle([(x, z)] \cdot [(x, y)] \cdot N)\rangle \prec (([(x, z)] \cdot [(x, y)] \cdot P') \parallel ([(x, z)] \cdot [(x, y)] \cdot Q'))$
by *simp*
then have $\Psi \rhd (\!|\nu x|\!)(P \parallel Q) \longmapsto RBrOut\ M\ ((\!|\nu*(xvec1 @ z\#xvec2)|\!)([(x, z)] \cdot [(x, y)] \cdot N) \prec' (([(x, z)] \cdot [(x, y)] \cdot P') \parallel ([(x, z)] \cdot [(x, y)] \cdot Q')))$
by(*simp add*: *residualInject*)
then have *permTrans3*: $\Psi \rhd (\!|\nu x|\!)(P \parallel Q) \longmapsto RBrOut\ M\ ((\!|\nu*(xvec1 @ z\#xvec2)|\!)(([(x, z)] \cdot [(x, y)] \cdot N) \prec' ([(x, z)] \cdot [(x, y)] \cdot (P' \parallel Q'))))$
by *simp*
from ‹$z \sharp N$› ‹$x \neq z$› ‹$z \neq y$›
have $z \sharp ([(x, y)] \cdot N)$
by (*metis calc-atm*(2) *calc-atm*(3) *fresh-right name-prm-name.simps*(2) *name-prm-name-def singleton-rev-conv swap-name-def*)
from ‹$z \sharp P'$› ‹$x \neq z$› ‹$z \neq y$›
have $z \sharp ([(x, y)] \cdot P')$
by (*metis calc-atm*(2) *calc-atm*(3) *fresh-right name-prm-name.simps*(2) *name-prm-name-def singleton-rev-conv swap-name-def*)
from ‹$z \sharp ([(x, y)] \cdot N)$› **have** $x \sharp ([(x, z)] \cdot [(x, y)] \cdot N)$
by (*metis calc-atm*(2) *calc-atm*(3) *fresh-right name-prm-name.simps*(2) *name-prm-name-def singleton-rev-conv swap-name-def*)
from ‹$z \sharp ([(x, y)] \cdot P')$› **have** $x \sharp ([(x, z)] \cdot [(x, y)] \cdot P')$
by (*metis calc-atm*(2) *calc-atm*(3) *fresh-right name-prm-name.simps*(2) *name-prm-name-def singleton-rev-conv swap-name-def*)
moreover from ‹$z \sharp [(x, y)] \cdot Q'$›
have $x \sharp ([(x, z)] \cdot [(x, y)] \cdot Q')$
by (*metis calc-atm*(2) *calc-atm*(3) *fresh-right name-prm-name.simps*(2) *name-prm-name-def singleton-rev-conv swap-name-def*)
ultimately have $x \sharp ([(x, z)] \cdot [(x, y)] \cdot (P' \parallel Q'))$
by *simp*
have $z \in set\ ((xvec1 @ z\#xvec2))$
by *simp*
from ‹$x \sharp ([(x, z)] \cdot [(x, y)] \cdot N)$› ‹$x \sharp ([(x, z)] \cdot [(x, y)] \cdot (P' \parallel Q'))$› ‹$z \in set\ ((xvec1 @ z\#xvec2))$›
have *eq1*: $((\!|\nu*(xvec1 @ z\#xvec2)|\!)\ (([(x, z)] \cdot [(x, y)] \cdot N) \prec' ([(x, z)] \cdot [(x, y)] \cdot (P' \parallel Q')))) = ((\!|\nu*([(z, x)] \cdot (xvec1 @ z\#xvec2))|\!)\ (([(z, x)] \cdot [(x, z)] \cdot [(x, y)] \cdot N) \prec' ([(z, x)] \cdot [(x, z)] \cdot [(x, y)] \cdot (P' \parallel Q'))))$
by(*rule boundOutputChainSwap*)
have *eq2*: $((\!|\nu*([(z, x)] \cdot (xvec1 @ z\#xvec2))|\!)\ (([(z, x)] \cdot [(x, z)] \cdot [(x, y)] \cdot N)$

$\prec'$ ([(z, x)] · [(x, z)] · [(x, y)] · (P′ ∥ Q′)))) = (⦇νx⦈(P ∥ Q) ⟼ RBrOut M (⦇νx⦈(([(x, y)] · N) $\prec'$ ([(x, y)] · (P′ ∥ Q′)))))
by auto (metis perm-swap(2))+
from ‹x ♯ xvec1› ‹x ♯ xvec2› ‹z ♯ xvec1› ‹z ♯ xvec2›
have ([(z, x)] · (xvec1@z#xvec2)) = (xvec1@x#xvec2)
by(simp add: eqvts swap-simps)
then have eq3: (⦇νx⦈([(z, x)] · (xvec1@z#xvec2))⦇νx⦈ (([(x, y)] · N) $\prec'$ ([(x, y)] · (P′ ∥ Q′)))) = (⦇νx⦈(xvec1@x#xvec2)⦇νx⦈ (([(x, y)] · N) $\prec'$ ([(x, y)] · (P′ ∥ Q′))))
by simp

from permTrans3 eq1 eq2 eq3 have noXZTrans: Ψ ▷ ⦇νx⦈(P ∥ Q) ⟼ RBrOut M (⦇νx⦈(xvec1@x#xvec2)⦇νx⦈ (([(x, y)] · N) $\prec'$ ([(x, y)] · (P′ ∥ Q′))))
by simp

from ‹x ♯ N›
have y ♯ ([(x, y)] · N)
by (metis calc-atm(2) fresh-right name-prm-name.simps(2) name-prm-name-def singleton-rev-conv swap-name-def swap-simps(2))
moreover from ‹x ♯ P′› ‹x ♯ Q′›
have y ♯ ([(x, y)] · (P′ ∥ Q′))
by (metis fresh-left psi.fresh(5) singleton-rev-conv swap-simps(2))
moreover have x ∈ set (xvec1@x#xvec2)
by simp
ultimately have eq1: ⦇νx⦈(xvec1@x#xvec2)⦇νx⦈([(x, y)] · N) $\prec'$ ([(x, y)] · (P′ ∥ Q′)) = ⦇νx⦈([(x, y)] · (xvec1@x#xvec2))⦇νx⦈([(x, y)] · [(x, y)] · N) $\prec'$ ([(x, y)] · [(x, y)] · (P′ ∥ Q′))
by(rule boundOutputChainSwap)
have eq2: ⦇νx⦈([(x, y)] · (xvec1@x#xvec2))⦇νx⦈([(x, y)] · [(x, y)] · N) $\prec'$ ([(x, y)] · [(x, y)] · (P′ ∥ Q′)) = ⦇νx⦈([(x, y)] · (xvec1@x#xvec2))⦇νx⦈N $\prec'$ (P′ ∥ Q′)
by simp
from ‹x ♯ xvec1› ‹x ♯ xvec2› ‹y ♯ xvec1› ‹y ♯ xvec2›
have eq3: ([(x, y)] · (xvec1@x#xvec2)) = (xvec1@y#xvec2)
by(simp add: eqvts swap-simps)

from eq1 eq2 eq3 noXZTrans have Ψ ▷ ⦇νx⦈(P ∥ Q) ⟼ RBrOut M (⦇νx⦈(xvec1@y#xvec2)⦇νx⦈N $\prec'$ (P′ ∥ Q′))
by simp
then have Ψ ▷ ⦇νx⦈(P ∥ Q) ⟼¡M⦇νx⦈(xvec1@y#xvec2)⦇νx⦈⟨N⟩ ≺ (P′ ∥ Q′)
by(simp add: residualInject)

with ‹xvec = xvec1@y#xvec2›
have Ψ ▷ ⦇νx⦈(P ∥ Q) ⟼¡M⦇νx⦈xvec⦇νx⦈⟨N⟩ ≺ (P′ ∥ Q′)
by simp
moreover have (Ψ, P′ ∥ Q′, P′ ∥ Q′) ∈ Rel
by(rule C1)

ultimately show ?case
by force
next

case(*cRes* Q')
from ‹$x \sharp ¡M(\!|\nu*xvec|\!)\langle N\rangle$›
have $x \sharp M$ **and** $x \sharp xvec$ **and** $x \sharp N$
by *simp*+
have *QTrans*: $\Psi \otimes \Psi_P \rhd Q \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec Q'$ **by** *fact*
with ‹$A_Q \sharp* Q$› ‹$A_Q \sharp* xvec$› ‹$xvec \sharp* M$› ‹*distinct xvec*› **have** $A_Q \sharp* Q'$
by(*force dest*: *broutputFreshChainDerivative*)

with ‹*extractFrame*($(\!|\nu x|\!)Q$) = $\langle A_Q, \Psi_Q\rangle$› **have** *FrxQ*: $(\!|\nu x|\!)$(*extractFrame* Q) = $\langle A_Q, \Psi_Q\rangle$ **by** *simp*
then obtain y A_Q' **where** *A*: $A_Q = y\#A_Q'$ **by**(*cases* A_Q) *auto*
with ‹$A_Q \sharp* \Psi$› ‹$A_Q \sharp* P$› ‹$A_Q \sharp* P'$› ‹$A_Q \sharp* \Psi_P$› ‹$A_Q \sharp* Q$› ‹$A_Q \sharp* xvec$› ‹$A_Q \sharp* M$› ‹$A_Q \sharp* Q'$› ‹$A_Q \sharp* N$›
have $A_Q' \sharp* \Psi$ **and** $A_Q' \sharp* P$ **and** $A_Q' \sharp* \Psi_P$ **and** $A_Q' \sharp* Q$ **and** $A_Q \sharp* xvec$ **and** $A_Q \sharp* Q'$
and $y \sharp \Psi$ **and** $y \sharp P$ **and** $y \sharp P'$ **and** $y \sharp \Psi_P$ **and** $y \sharp Q$ **and** $y \sharp xvec$ **and** $y \sharp M$ **and** $y \sharp Q'$
and $A_Q' \sharp* M$ **and** $y \sharp N$
by(*simp*)+
from *A* ‹$A_P \sharp* A_Q$› **have** $A_P \sharp* A_Q'$ **and** $y \sharp A_P$ **by**(*simp add*: *fresh-star-list-cons*)+
from *A* ‹$A_Q \sharp* x$› **have** $x \neq y$ **and** $x \sharp A_Q'$ **by**(*simp add*: *fresh-list-cons*)+

with *A* ‹*distinct* A_Q› **have** $y \sharp A_Q'$
by(*induct* A_Q') (*auto simp add*: *fresh-list-nil fresh-list-cons*)

from ‹$x \sharp N$› **have** $y \sharp [(x, y)] \cdot N$
by (*metis calc-atm*(2) *fresh-right name-prm-name.simps*(2) *name-prm-name-def singleton-rev-conv swap-name-def swap-simps*(2))

from ‹$x \sharp P$› ‹$y \sharp P$› ‹$x \neq y$› ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$›
have $\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|([(y, x)] \cdot N)|\!) \prec [(y, x)] \cdot P'$
by(*elim brinputAlpha*[**where** *xvec*=[*y*]]) (*auto simp add*: *calc-atm*)
then have $\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|([(x, y)] \cdot N)|\!) \prec [(x, y)] \cdot P'$
by(*simp add*: *name-swap*)
moreover note *FrP*
moreover from ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec Q'$› **have** $([(x, y)] \cdot (\Psi \otimes \Psi_P)) \rhd ([(x, y)] \cdot Q) \longmapsto ([(x, y)] \cdot (¡M(\!|\nu*xvec|\!)\langle N\rangle \prec Q'))$
by(*rule semantics.eqvt*)
with ‹$x \sharp \Psi$› ‹$y \sharp \Psi$› ‹$x \sharp \Psi_P$› ‹$y \sharp \Psi_P$› ‹$x \sharp M$› ‹$y \sharp M$› ‹$x \sharp xvec$› ‹$y \sharp xvec$›
have $\Psi \otimes \Psi_P \rhd ([(x, y)] \cdot Q) \longmapsto ¡M(\!|\nu*xvec|\!)\langle([(x, y)] \cdot N)\rangle \prec ([(x, y)] \cdot Q')$
by(*simp add*: *eqvts*)
moreover from *A* ‹$A_Q \sharp* x$› *FrxQ* **have** *FrQ*: *extractFrame*($[(x, y)] \cdot Q$) = $\langle A_Q', \Psi_Q\rangle$
by(*clarsimp simp add*: *alpha'* *eqvts frame.inject fresh-list-cons name-swap*)
moreover from ‹$A_P \sharp* Q$› **have** $([(x, y)] \cdot A_P) \sharp* ([(x, y)] \cdot Q)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$A_P \sharp* x$› ‹$y \sharp A_P$› **have** $A_P \sharp* ([(x, y)] \cdot Q)$ **by** *simp*

moreover from ‹$A_Q{}' \sharp* Q$› **have** $([(x, y)] \cdot A_Q{}') \sharp* ([(x, y)] \cdot Q)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$x \sharp A_Q{}'$› ‹$y \sharp A_Q{}'$› **have** $A_Q{}' \sharp* ([(x, y)] \cdot Q)$ **by** *simp*
ultimately have $\Psi \rhd (P \parallel ([(x, y)] \cdot Q)) \longmapsto ¡M(\!|\nu * xvec|\!)\langle([(x, y)] \cdot N)\rangle \prec ([(x, y)] \cdot P') \parallel ([(x, y)] \cdot Q')$
using ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_P \sharp* A_Q{}'$› ‹$A_Q{}' \sharp* \Psi$› ‹$A_Q{}' \sharp* P$› ‹$xvec \sharp* P$› ‹$A_P \sharp* M$› ‹$A_Q{}' \sharp* M$›
by(*intro BrComm1*) (*assumption* | *simp*)+
with‹$y \sharp \Psi$› ‹$y \sharp M$› ‹$y \sharp xvec$› ‹$y \sharp [(x, y)] \cdot N$› **have** $\Psi \rhd (\!|\nu y|\!)(P \parallel ([(x, y)] \cdot Q)) \longmapsto ¡M(\!|\nu * xvec|\!)\langle([(x, y)] \cdot N)\rangle \prec (\!|\nu y|\!)(([(x, y)] \cdot P') \parallel ([(x, y)] \cdot Q'))$
by(*intro Scope*) *auto*
moreover from ‹$x \sharp P$› ‹$y \sharp P$› ‹$y \sharp Q$› **have** $(\!|\nu y|\!)(P \parallel ([(x, y)] \cdot Q)) = (\!|\nu x|\!)([(x, y)] \cdot (P \parallel ([(x, y)] \cdot Q)))$
by(*subst alphaRes*[*of x*]) (*auto simp add*: *calc-atm fresh-left name-swap*)
with ‹$x \sharp P$› ‹$y \sharp P$› **have** $(\!|\nu y|\!)(P \parallel ([(x, y)] \cdot Q)) = (\!|\nu x|\!)(P \parallel Q)$
by(*simp add*: *eqvts*)
moreover from ‹$y \sharp P'$› ‹$y \sharp Q'$› ‹$x \sharp xvec$› ‹$y \sharp xvec$› **have** $(\!|\nu y|\!)(([(x, y)] \cdot P') \parallel ([(x, y)] \cdot Q')) = (\!|\nu x|\!)(P' \parallel Q')$
by(*subst alphaRes*[*of y*]) (*auto simp add*: *resChainFresh calc-atm eqvts fresh-left name-swap*)
ultimately have $\Psi \rhd (\!|\nu x|\!)(P \parallel Q) \longmapsto ¡M(\!|\nu * xvec|\!)\langle([(x, y)] \cdot N)\rangle \prec (\!|\nu x|\!)(P' \parallel Q')$
by *simp*
with ‹$x \sharp N$› ‹$y \sharp N$›
have *Trans*: $\Psi \rhd (\!|\nu x|\!)(P \parallel Q) \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec (\!|\nu x|\!)(P' \parallel Q')$
by *simp*
from ‹$x \sharp P'$› ‹$x \sharp \Psi$›
have $(\Psi, (\!|\nu * [x]|\!)(P' \parallel Q'), (P' \parallel ((\!|\nu * [x]|\!) Q'))) \in Rel$
by(*intro C4*) *simp*+
then have *Relation*: $(\Psi, (\!|\nu x|\!)(P' \parallel Q'), (P' \parallel ((\!|\nu x|\!) Q'))) \in Rel$
by *simp*
from *Trans Relation* **show** *?case*
by *blast*
qed
next
case(*cBrComm2* Ψ_Q *M xvec N P'* A_P Ψ_P xQ' A_Q)
from ‹$x \sharp \alpha$› ‹$¡M(\!|\nu * xvec|\!)\langle N\rangle = \alpha$›
have $x \sharp M$ $x \sharp xvec$ $x \sharp N$
by *force*+
have *QTrans*: $\Psi \otimes \Psi_P \rhd (\!|\nu x|\!) Q \longmapsto ¿M(\!|N|\!) \prec xQ'$ **and** *FrQ*: *extractFrame*$((\!|\nu x|\!) Q) = \langle A_Q, \Psi_Q\rangle$ **by** *fact*+
have *PTrans*: $\Psi \otimes \Psi_Q \rhd P \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$ **and** *FrP*: *extractFrame* $P = \langle A_P, \Psi_P\rangle$ **by** *fact*+
from *PTrans* ‹$x \sharp P$› **have** $x \sharp N$ **and** $x \sharp P'$ **using** ‹$x \sharp xvec$› ‹$xvec \sharp* M$› ‹*distinct xvec*›
by(*force intro*: *broutputFreshDerivative*)+
have $x \sharp (\!|\nu x|\!) Q$ **by**(*simp add*: *abs-fresh*)
with *FrQ* ‹$A_Q \sharp* x$› **have** $x \sharp \Psi_Q$
by(*drule-tac extractFrameFresh*) *auto*

from ‹$x \sharp P$› *FrP* ‹$A_P \sharp* x$› **have** $x \sharp \Psi_P$
by(*drule-tac extractFrameFresh*) *auto*
from ‹$A_P \sharp* (\!|\nu x|\!) Q$› ‹$A_P \sharp* x$› **have** $A_P \sharp* Q$ **by** *simp*
from ‹$A_Q \sharp* (\!|\nu x|\!) Q$› ‹$A_Q \sharp* x$› **have** $A_Q \sharp* Q$ **by** *simp*
from ‹$x \sharp xvec$› ‹$xvec \sharp* (\!|\nu x|\!) Q$› **have** *xvec* $\sharp* Q$ **by** *simp*

from ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$› **have** *PResTrans*: $\Psi \otimes \Psi_Q \rhd P \longmapsto RBrOut\ M\ ((\!|\nu * xvec|\!) N \prec' P')$
by(*simp add*: *residualInject*)

note *QTrans*
moreover from ‹$x \sharp \Psi$› ‹$x \sharp \Psi_P$› **have** $x \sharp \Psi \otimes \Psi_P$ **by** *force*
moreover note ‹$x \sharp M$› ‹$x \sharp N$›
moreover from *QTrans* ‹$x \sharp N$› **have** $x \sharp xQ'$ **by**(*force dest*: *brinputFreshDerivative simp add*: *abs-fresh*)
ultimately show *?case*
proof(*induct rule*: *resBrInputCases*)
case(*cRes Q'*)
have *QTrans*: $\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$ **by** *fact*
with ‹$A_Q \sharp* Q$› ‹$A_Q \sharp* N$› **have** $A_Q \sharp* Q'$
by(*elim brinputFreshChainDerivative*)

with ‹*extractFrame*$((\!|\nu x|\!) Q) = \langle A_Q, \Psi_Q\rangle$› **have** *FrxQ*: $(\!|\nu x|\!)(extractFrame\ Q) = \langle A_Q, \Psi_Q\rangle$ **by** *simp*
then obtain *y* A_Q' **where** *A*: $A_Q = y \# A_Q'$ **by**(*cases* A_Q) *auto*
with ‹$A_Q \sharp* \Psi$› ‹$A_Q \sharp* P$› ‹$A_Q \sharp* P'$› ‹$A_Q \sharp* \Psi_P$› ‹$A_Q \sharp* Q$› ‹$A_Q \sharp* xvec$› ‹$A_Q \sharp* M$› ‹$A_Q \sharp* Q'$› ‹$A_Q \sharp* N$›
have $A_Q' \sharp* \Psi$ **and** $A_Q' \sharp* P$ **and** $A_Q' \sharp* \Psi_P$ **and** $A_Q' \sharp* Q$ **and** $A_Q \sharp* xvec$ **and** $A_Q \sharp* Q'$
and $y \sharp \Psi$ **and** $y \sharp P$ **and** $y \sharp P'$ **and** $y \sharp \Psi_P$ **and** $y \sharp Q$ **and** $y \sharp xvec$ **and** $y \sharp M$ **and** $y \sharp Q'$ **and** $y \sharp N$
and $A_Q' \sharp* M$
by(*simp*)+
from *A* ‹$A_P \sharp* A_Q$› **have** $A_P \sharp* A_Q'$ **and** $y \sharp A_P$ **by**(*simp add*: *fresh-star-list-cons*)+
from *A* ‹$A_Q \sharp* x$› **have** $x \neq y$ **and** $x \sharp A_Q'$ **by**(*simp add*: *fresh-list-cons*)+

with *A* ‹*distinct* A_Q› **have** $y \sharp A_Q'$
by(*induct* A_Q') (*auto simp add*: *fresh-list-nil fresh-list-cons*)

note *PTrans FrP*
moreover from ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$› **have** $([(x, y)] \cdot (\Psi \otimes \Psi_P)) \rhd ([(x, y)] \cdot Q) \longmapsto ([(x, y)] \cdot (¿M(\!|N|\!) \prec Q'))$
by(*rule semantics.eqvt*)
with ‹$x \sharp \Psi$› ‹$y \sharp \Psi$› ‹$x \sharp \Psi_P$› ‹$y \sharp \Psi_P$› ‹$x \sharp M$› ‹$y \sharp M$› ‹$x \sharp N$› ‹$y \sharp N$›
have $\Psi \otimes \Psi_P \rhd ([(x, y)] \cdot Q) \longmapsto ¿M(\!|N|\!) \prec ([(x, y)] \cdot Q')$
by(*simp add*: *eqvts*)
moreover from *A* ‹$A_Q \sharp* x$› *FrxQ* **have** *FrQ*: *extractFrame*$([(x, y)] \cdot Q) = \langle A_Q', \Psi_Q\rangle$ **and** $y \sharp extractFrame\ Q$
by(*clarsimp simp add*: *alpha' eqvts frame.inject fresh-list-cons name-swap*)+

moreover from ‹$A_P \sharp* Q$› **have** $([(x, y)] \cdot A_P) \sharp* ([(x, y)] \cdot Q)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$A_P \sharp* x$› ‹$y \sharp A_P$› **have** $A_P \sharp* ([(x, y)] \cdot Q)$ **by** *simp*
moreover from ‹$A_Q' \sharp* Q$› **have** $([(x, y)] \cdot A_Q') \sharp* ([(x, y)] \cdot Q)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$x \sharp A_Q'$› ‹$y \sharp A_Q'$› **have** $A_Q' \sharp* ([(x, y)] \cdot Q)$ **by** *simp*
moreover from ‹*xvec* $\sharp* Q$› **have** $([(x, y)] \cdot xvec) \sharp* ([(x, y)] \cdot Q)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$x \sharp xvec$› ‹$y \sharp xvec$› **have** $xvec \sharp* ([(x, y)] \cdot Q)$ **by** *simp*
ultimately have $\Psi \rhd (P \parallel ([(x, y)] \cdot Q)) \longmapsto ¡M (\!| \nu* xvec |\!) \langle N \rangle \prec (P' \parallel ([(x, y)] \cdot Q'))$
using ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_P \sharp* A_Q'$› ‹$A_Q' \sharp* \Psi$› ‹$A_Q' \sharp* P$› ‹$A_P \sharp* M$› ‹$A_Q' \sharp* M$›
by(*intro BrComm2*) (*assumption* | *simp*)+
with ‹$y \sharp \Psi$› ‹$y \sharp M$› ‹$y \sharp xvec$› ‹$y \sharp N$› **have** $\Psi \rhd (\!| \nu y |\!)(P \parallel ([(x, y)] \cdot Q)) \longmapsto ¡M (\!| \nu* xvec |\!) \langle N \rangle \prec (\!| \nu y |\!)(P' \parallel ([(x, y)] \cdot Q'))$
by(*intro Scope*) *auto*
moreover from ‹$x \sharp P$› ‹$y \sharp P$› ‹$y \sharp Q$› **have** $(\!| \nu y |\!)(P \parallel ([(x, y)] \cdot Q)) = (\!| \nu x |\!)([(x, y)] \cdot (P \parallel ([(x, y)] \cdot Q)))$
by(*subst alphaRes*[*of x*]) (*auto simp add*: *calc-atm fresh-left name-swap*)
with ‹$x \sharp P$› ‹$y \sharp P$› **have** $(\!| \nu y |\!)(P \parallel ([(x, y)] \cdot Q)) = (\!| \nu x |\!)(P \parallel Q)$
by(*simp add*: *eqvts*)
moreover from ‹$x \sharp P'$› ‹$y \sharp P'$› ‹$y \sharp Q'$› ‹$x \sharp xvec$› ‹$y \sharp xvec$› **have** $(\!| \nu y |\!)(P' \parallel ([(x, y)] \cdot Q')) = (\!| \nu x |\!)(P' \parallel Q')$
by(*subst alphaRes*[*of y*]) (*auto simp add*: *resChainFresh calc-atm eqvts fresh-left name-swap*)
ultimately have $\Psi \rhd (\!| \nu x |\!)(P \parallel Q) \longmapsto ¡M (\!| \nu* xvec |\!) \langle N \rangle \prec (\!| \nu x |\!)(P' \parallel Q')$
by *simp*
moreover from ‹$x \sharp \Psi$› ‹$x \sharp P'$› **have** $(\Psi, (\!| \nu* [x] |\!)(P' \parallel Q'), (P' \parallel ((\!| \nu* [x] |\!) Q'))) \in Rel$
by(*intro C4*) *simp*+
moreover then have $(\Psi, (\!| \nu x |\!)(P' \parallel Q'), (P' \parallel ((\!| \nu x |\!) Q'))) \in Rel$
by *simp*
ultimately show *?case* **by** *blast*
qed
qed
qed
qed

lemma *scopeExtRight*:
fixes x :: *name*
and P :: ('*a*, '*b*, '*c*) *psi*
and Ψ :: '*b*
and Q :: ('*a*, '*b*, '*c*) *psi*
and *Rel* :: ('*b* × ('*a*, '*b*, '*c*) *psi* × ('*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*: $\bigwedge y\ \Psi'\ R\ S\ zvec.\ [\![y \sharp \Psi';\ y \sharp R;\ zvec \sharp* \Psi]\!] \Longrightarrow (\Psi',\ (\!|\nu * zvec|\!)(R \parallel (\!|\nu y|\!) S),\ (\!|\nu y|\!)((\!|\nu * zvec|\!)(R \parallel S))) \in Rel$
— Addition for Broadcast
and *C3*: $\bigwedge \Psi'\ R\ S\ zvec.\ [\![zvec \sharp* R;\ zvec \sharp* \Psi]\!] \Longrightarrow (\Psi',\ (R \parallel ((\!|\nu * zvec|\!) S)),\ ((\!|\nu * zvec|\!)(R \parallel S))) \in Rel$

shows $\Psi \rhd P \parallel (\!|\nu x|\!) Q \rightsquigarrow [Rel]\ (\!|\nu x|\!)(P \parallel Q)$
proof −
note ‹*eqvt Rel*› ‹$x \sharp \Psi$›
moreover from ‹$x \sharp P$› **have** $x \sharp P \parallel (\!|\nu x|\!) Q$ **by**(*simp add*: *abs-fresh*)
moreover from ‹$x \sharp P$› **have** $x \sharp (\!|\nu x|\!)(P \parallel Q)$ **by**(*simp add*: *abs-fresh*)
ultimately show *?thesis*
proof(*induct rule*: *simIFresh*[*of* - - - - - ()])
case(*cSim* α *xPQ*)
from ‹*bn* $\alpha \sharp* (P \parallel (\!|\nu x|\!) Q)$› ‹$x \sharp \alpha$› **have** *bn* $\alpha \sharp* P$ **and** *bn* $\alpha \sharp* Q$ **by** *simp*+
note ‹$\Psi \rhd (\!|\nu x|\!)(P \parallel Q) \longmapsto \alpha \prec xPQ$› ‹$x \sharp \Psi$› ‹$x \sharp \alpha$› ‹$x \sharp xPQ$› ‹*bn* $\alpha \sharp* \Psi$›
moreover from ‹*bn* $\alpha \sharp* P$› ‹*bn* $\alpha \sharp* Q$› **have** *bn* $\alpha \sharp* (P \parallel Q)$ **by** *simp*
ultimately show *?case* **using** ‹*bn* $\alpha \sharp*$ *subject* α› ‹$x \sharp \alpha$›
proof(*induct rule*: *resCases*[**where** *C*=()])
case(*cOpen M xvec1 xvec2 y N PQ*)
from ‹$x \sharp M(\!|\nu * (xvec1 @ y \# xvec2)|\!)\langle N \rangle$› **have** $x \sharp xvec1$ **and** $x \neq y$ **and** $x \sharp xvec2$ **and** $x \sharp M$ **by** *simp*+
from ‹*xvec1* $\sharp* (P \parallel Q)$› ‹*xvec2* $\sharp* (P \parallel Q)$› ‹$y \sharp (P \parallel Q)$›
have $(xvec1 @ xvec2) \sharp* P$ **and** $(xvec1 @ xvec2) \sharp* Q$ **and** $y \sharp P$ **and** $y \sharp Q$
by *simp*+
from ‹$\Psi \rhd P \parallel Q \longmapsto M(\!|\nu * (xvec1 @ xvec2)|\!)\langle([(x,\ y)] \cdot N)\rangle \prec ([(x,\ y)] \cdot PQ)$›
have $([(x,\ y)] \cdot \Psi) \rhd ([(x,\ y)] \cdot (P \parallel Q)) \longmapsto ([(x,\ y)] \cdot (M(\!|\nu * (xvec1 @ xvec2)|\!)\langle([(x,\ y)] \cdot N)\rangle \prec ([(x,\ y)] \cdot PQ)))$
by(*rule semantics.eqvt*)
with ‹$x \sharp \Psi$› ‹$y \sharp \Psi$› ‹$x \sharp P$› ‹$y \sharp P$› ‹$x \sharp M$› ‹$y \sharp M$› ‹$x \sharp xvec1$› ‹$x \sharp xvec2$› ‹$y \sharp xvec1$› ‹$y \sharp xvec2$›
have $\Psi \rhd P \parallel ([(x,\ y)] \cdot Q) \longmapsto M(\!|\nu * (xvec1 @ xvec2)|\!)\langle N \rangle \prec PQ$
by(*simp add*: *eqvts*)
moreover from ‹*xvec1* $\sharp* \Psi$› ‹*xvec2* $\sharp* \Psi$› **have** $(xvec1 @ xvec2) \sharp* \Psi$ **by** *simp*
moreover note ‹$(xvec1 @ xvec2) \sharp* P$›
moreover from ‹$(xvec1 @ xvec2) \sharp* Q$› **have** $([(x,\ y)] \cdot (xvec1 @ xvec2)) \sharp* ([(x,\ y)] \cdot Q)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$x \sharp xvec1$› ‹$x \sharp xvec2$› ‹$y \sharp xvec1$› ‹$y \sharp xvec2$› **have** $(xvec1 @ xvec2) \sharp* ([(x,\ y)] \cdot Q)$
by(*auto simp add*: *eqvts*)
moreover from ‹*xvec1* $\sharp* M$› ‹*xvec2* $\sharp* M$› **have** $(xvec1 @ xvec2) \sharp* M$ **by** *simp*
ultimately show *?case*
proof(*induct rule*: *parOutputCases*[**where** *C*=*y*])
case(*cPar1 P' A*$_Q$ Ψ_Q)

from ‹$y \sharp$ *xvec1*› ‹$y \sharp$ *xvec2*› **have** $y \sharp (xvec1@xvec2)$ **by**(*auto simp add*: *fresh-list-append*)
with ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|\nu*(xvec1@xvec2)|\!)\langle N\rangle \prec P'$› ‹$(xvec1@xvec2) \sharp* M$› ‹$y \sharp P$›
‹*distinct xvec1*› ‹*distinct xvec2*› ‹*xvec1* $\sharp*$ *xvec2*›
have $y \sharp N$ **by**(*force intro*: *outputFreshDerivative*)
with ‹$y \in$ *supp N*› **have** *False* **by**(*simp add*: *fresh-def*)
then show *?case* **by** *simp*
next
case(*cPar2 Q' A_P* Ψ_P)
have *FrP*: *extractFrame P* = $\langle A_P, \Psi_P\rangle$ **by** *fact*
with ‹$y \sharp P$› ‹$A_P \sharp* y$› **have** $y \sharp \Psi_P$
apply(*drule-tac extractFrameFresh*)
by(*simp add*: *frameResChainFresh*) (*simp add*: *fresh-def name-list-supp*)
from ‹$\Psi \otimes \Psi_P \rhd ([(x, y)] \cdot Q) \longmapsto M(\!|\nu*(xvec1@xvec2)|\!)\langle N\rangle \prec Q'$› ‹$y \in$ *supp N*› ‹$y \sharp \Psi$› ‹$y \sharp \Psi_P$› ‹$y \sharp M$› ‹$y \sharp$ *xvec1*› ‹$y \sharp$ *xvec2*›
have $\Psi \otimes \Psi_P \rhd (\!|\nu y|\!)(([(x, y)] \cdot Q)) \longmapsto M(\!|\nu*(xvec1@y\#xvec2)|\!)\langle N\rangle \prec Q'$ **by**(*force intro*: *Open*)
with ‹$y \sharp Q$› **have** $\Psi \otimes \Psi_P \rhd (\!|\nu x|\!)Q \longmapsto M(\!|\nu*(xvec1@y\#xvec2)|\!)\langle N\rangle \prec Q'$
by(*simp add*: *alphaRes*)
moreover from ‹$A_P \sharp* ([(x, y)] \cdot Q)$› **have** $A_P \sharp* (\!|\nu y|\!)([(x, y)] \cdot Q)$ **by**(*auto simp add*: *fresh-star-def abs-fresh*)
with ‹$y \sharp Q$› **have** $A_P \sharp* (\!|\nu x|\!)Q$ **by**(*simp add*: *alphaRes*)
ultimately have $\Psi \rhd P \parallel ((\!|\nu x|\!)Q) \longmapsto M(\!|\nu*(xvec1@y\#xvec2)|\!)\langle N\rangle \prec (P \parallel Q')$
using *FrP* ‹$(xvec1@xvec2) \sharp* P$› ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* M$› ‹$y \sharp P$› ‹$A_P \sharp* (xvec1@xvec2)$› ‹$A_P \sharp* y$› ‹$A_P \sharp* N$›
by(*intro Par2*) *auto*
moreover have $(\Psi, P \parallel Q', P \parallel Q') \in$ *Rel* **by**(*rule C1*)
ultimately show *?case* **by** *blast*
qed
next
case(*cBrOpen M xvec1 xvec2 y N PQ*)
from ‹$x \sharp ¡M(\!|\nu*(xvec1@y\#xvec2)|\!)\langle N\rangle$› **have** $x \sharp$ *xvec1* **and** $x \neq y$ **and** $x \sharp$ *xvec2* **and** $x \sharp M$ **by** *simp+*
from ‹*xvec1* $\sharp*$ $(P \parallel Q)$› ‹*xvec2* $\sharp*$ $(P \parallel Q)$› ‹$y \sharp (P \parallel Q)$›
have $(xvec1@xvec2) \sharp* P$ **and** $(xvec1@xvec2) \sharp* Q$ **and** $y \sharp P$ **and** $y \sharp Q$
by *simp+*
from ‹$\Psi \rhd P \parallel Q \longmapsto ¡M(\!|\nu*(xvec1@xvec2)|\!)\langle([(x, y)] \cdot N)\rangle \prec ([(x, y)] \cdot PQ)$›
have $([(x, y)] \cdot \Psi) \rhd ([(x, y)] \cdot (P \parallel Q)) \longmapsto ([(x, y)] \cdot (¡M(\!|\nu*(xvec1@xvec2)|\!)\langle([(x, y)] \cdot N)\rangle \prec ([(x, y)] \cdot PQ)))$
by(*rule semantics.eqvt*)
with ‹$x \sharp \Psi$› ‹$y \sharp \Psi$› ‹$x \sharp P$› ‹$y \sharp P$› ‹$x \sharp M$› ‹$y \sharp M$› ‹$x \sharp$ *xvec1*› ‹$x \sharp$ *xvec2*› ‹$y \sharp$ *xvec1*› ‹$y \sharp$ *xvec2*›
have $\Psi \rhd P \parallel ([(x, y)] \cdot Q) \longmapsto ¡M(\!|\nu*(xvec1@xvec2)|\!)\langle N\rangle \prec PQ$
by(*simp add*: *eqvts*)
moreover from ‹*xvec1* $\sharp*$ Ψ› ‹*xvec2* $\sharp*$ Ψ› **have** $(xvec1@xvec2) \sharp* \Psi$ **by**

simp
moreover note ‹(*xvec1*@*xvec2*) ♯* *P*›
moreover from ‹(*xvec1*@*xvec2*) ♯* *Q*› **have** ([(*x*, *y*)] ∙ (*xvec1*@*xvec2*)) ♯* ([(*x*, *y*)] ∙ *Q*)
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹*x* ♯ *xvec1*› ‹*x* ♯ *xvec2*› ‹*y* ♯ *xvec1*› ‹*y* ♯ *xvec2*› **have** (*xvec1*@*xvec2*) ♯* ([(*x*, *y*)] ∙ *Q*)
by(*auto simp add*: *eqvts*)
moreover from ‹*xvec1* ♯* *M*› ‹*xvec2* ♯* *M*› **have** (*xvec1*@*xvec2*) ♯* *M* **by** *simp*
ultimately show *?case*
proof(*induct rule*: *parBrOutputCases*[**where** *C*=*y*])
case(*cPar1 P′ A*$_Q$ Ψ$_Q$)
from ‹*y* ♯ *xvec1*› ‹*y* ♯ *xvec2*› **have** *y* ♯ (*xvec1*@*xvec2*) **by**(*auto simp add*: *fresh-list-append*)
with ‹Ψ ⊗ Ψ$_Q$ ▷ *P* ⟼¡*M*⦇*ν**(*xvec1*@*xvec2*)⦈⟨*N*⟩ ≺ *P′*› ‹(*xvec1*@*xvec2*) ♯* *M*› ‹*y* ♯ *P*›
‹*distinct xvec1*› ‹*distinct xvec2*› ‹*xvec1* ♯* *xvec2*›
have *y* ♯ *N* **by**(*force intro*: *broutputFreshDerivative*)
with ‹*y* ∈ *supp N*› **have** *False* **by**(*simp add*: *fresh-def*)
then show *?case* **by** *simp*
next
case(*cPar2 Q′ A*$_P$ Ψ$_P$)
have *FrP*: *extractFrame P* = ⟨*A*$_P$, Ψ$_P$⟩ **by** *fact*
with ‹*y* ♯ *P*› ‹*A*$_P$ ♯* *y*› **have** *y* ♯ Ψ$_P$
apply(*drule-tac extractFrameFresh*)
by(*simp add*: *frameResChainFresh*) (*simp add*: *fresh-def name-list-supp*)
from ‹Ψ ⊗ Ψ$_P$ ▷ ([(*x*, *y*)] ∙ *Q*) ⟼¡*M*⦇*ν**(*xvec1*@*xvec2*)⦈⟨*N*⟩ ≺ *Q′*› ‹*y* ∈ *supp N*› ‹*y* ♯ Ψ› ‹*y* ♯ Ψ$_P$› ‹*y* ♯ *M*› ‹*y* ♯ *xvec1*› ‹*y* ♯ *xvec2*›
have Ψ ⊗ Ψ$_P$ ▷ ⦇*νy*⦈([(*x*, *y*)] ∙ *Q*)) ⟼¡*M*⦇*ν**(*xvec1*@*y*#*xvec2*)⦈⟨*N*⟩ ≺ *Q′* **by**(*force intro*: *BrOpen*)
with ‹*y* ♯ *Q*› **have** Ψ ⊗ Ψ$_P$ ▷ ⦇*νx*⦈*Q* ⟼¡*M*⦇*ν**(*xvec1*@*y*#*xvec2*)⦈⟨*N*⟩ ≺ *Q′*
by(*simp add*: *alphaRes*)
moreover from ‹*A*$_P$ ♯* ([(*x*, *y*)] ∙ *Q*)› **have** *A*$_P$ ♯* ⦇*νy*⦈([(*x*, *y*)] ∙ *Q*) **by**(*auto simp add*: *fresh-star-def abs-fresh*)
with ‹*y* ♯ *Q*› **have** *A*$_P$ ♯* ⦇*νx*⦈*Q* **by**(*simp add*: *alphaRes*)
ultimately have Ψ ▷ *P* ∥ (⦇*νx*⦈*Q*) ⟼¡*M*⦇*ν**(*xvec1*@*y*#*xvec2*)⦈⟨*N*⟩ ≺ (*P* ∥ *Q′*)
using *FrP* ‹(*xvec1*@*xvec2*) ♯* *P*› ‹*A*$_P$ ♯* Ψ› ‹*A*$_P$ ♯* *M*› ‹*y* ♯ *P*› ‹*A*$_P$ ♯* (*xvec1*@*xvec2*)› ‹*A*$_P$ ♯* *y*› ‹*A*$_P$ ♯* *N*›
by(*intro Par2*) *auto*
moreover have (Ψ, *P* ∥ *Q′*, *P* ∥ *Q′*) ∈ *Rel* **by**(*rule C1*)
ultimately show *?case* **by** *blast*
next
case(*cBrComm1* Ψ$_Q$ *P′ A*$_P$ Ψ$_P$ *Q′ A*$_Q$)
have *FrP*: *extractFrame P* = ⟨*A*$_P$, Ψ$_P$⟩ **by** *fact*
with ‹*y* ♯ *P*› ‹*A*$_P$ ♯* *y*› **have** *y* ♯ Ψ$_P$
apply(*drule-tac extractFrameFresh*)

by(*simp add*: *frameResChainFresh*) (*simp add*: *fresh-def name-list-supp*)

from ‹*extractFrame* ($[(x, y)] \cdot Q$) = $\langle A_Q, \Psi_Q\rangle$›
have *extractFrame* ($(\!|\nu y|\!)([(x, y)] \cdot Q)$) = $\langle(y\#A_Q), \Psi_Q\rangle$
by *simp*
with ‹$y \sharp Q$›
have *FrQ*: *extractFrame* ($(\!|\nu x|\!) Q$) = $\langle(y\#A_Q), \Psi_Q\rangle$
by(*simp add*: *alphaRes*)
from ‹$\Psi \otimes \Psi_P \rhd [(x, y)] \cdot Q \longmapsto ¡M(\!|\nu*(xvec1 \ @ \ xvec2)|\!)\langle N\rangle \prec Q'$› ‹$y \in$ *supp* N› ‹$y \sharp \Psi$› ‹$y \sharp \Psi_P$› ‹$y \sharp M$› ‹$y \sharp$ *xvec1*› ‹$y \sharp$ *xvec2*›
have $\Psi \otimes \Psi_P \rhd (\!|\nu y|\!)([(x, y)] \cdot Q) \longmapsto ¡M(\!|\nu*(xvec1@y\#xvec2)|\!)\langle N\rangle \prec Q'$
by(*force intro*: *BrOpen*)
with ‹$y \sharp Q$› **have** *QTrans*: $\Psi \otimes \Psi_P \rhd (\!|\nu x|\!) Q \longmapsto ¡M(\!|\nu*(xvec1@y\#xvec2)|\!)\langle N\rangle \prec Q'$
by(*simp add*: *alphaRes*)
from ‹$A_P \sharp* ([(x, y)] \cdot Q)$› **have** $A_P \sharp* (\!|\nu y|\!)([(x, y)] \cdot Q)$ **by**(*auto simp add*: *fresh-star-def abs-fresh*)
with ‹$y \sharp Q$› **have** $A_P \sharp* (\!|\nu x|\!) Q$ **by**(*simp add*: *alphaRes*)
from ‹$A_Q \sharp* ([(x, y)] \cdot Q)$› **have** $A_Q \sharp* (\!|\nu y|\!)([(x, y)] \cdot Q)$ **by**(*auto simp add*: *fresh-star-def abs-fresh*)
with ‹$y \sharp Q$› **have** $A_Q \sharp* (\!|\nu x|\!) Q$ **by**(*simp add*: *alphaRes*)
moreover from ‹$y \sharp Q$› **have** $y \sharp (\!|\nu x|\!) Q$
by (*metis resChain.base resChain.step resChainFresh*)
ultimately have $(y \# A_Q) \sharp* ((\!|\nu x|\!) Q)$ **by** *simp*
from ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$› *FrP QTrans FrQ*
‹$A_P \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_P \sharp* (\!|\nu x|\!) Q$› ‹$A_P \sharp* M$› ‹$A_P \sharp* y$› ‹$A_P \sharp* A_Q$›
‹$y \sharp \Psi$› ‹$A_Q \sharp* \Psi$› ‹$y \sharp P$› ‹$A_Q \sharp* P$› ‹$(y\#A_Q) \sharp* (\!|\nu x|\!) Q$›
‹$y \sharp M$› ‹$A_Q \sharp* M$› ‹$y \sharp P$› ‹$(xvec1@xvec2) \sharp* P$›
have $\Psi \rhd P \parallel ((\!|\nu x|\!) Q) \longmapsto ¡M(\!|\nu*(xvec1@y\#xvec2)|\!)\langle N\rangle \prec (P' \parallel Q')$
by(*intro BrComm1*) (*assumption* | *simp*)+
moreover have $(\Psi, P' \parallel Q', P' \parallel Q') \in Rel$ **by**(*rule C1*)
ultimately show *?case* **by** *blast*
next
case(*cBrComm2* Ψ_Q P' A_P Ψ_P Q' A_Q)
from ‹$y \sharp$ *xvec1*› ‹$y \sharp$ *xvec2*› **have** $y \sharp (xvec1@xvec2)$ **by**(*auto simp add*: *fresh-list-append*)
with ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto ¡M(\!|\nu*(xvec1@xvec2)|\!)\langle N\rangle \prec P'$› ‹$(xvec1@xvec2) \sharp* M$› ‹$y \sharp P$›
‹*distinct xvec1*› ‹*distinct xvec2*› ‹*xvec1* $\sharp*$ *xvec2*›
have $y \sharp N$ **by**(*force intro*: *broutputFreshDerivative*)
with ‹$y \in$ *supp* N› **have** *False* **by**(*simp add*: *fresh-def*)
then show *?case* **by** *simp*
qed
next
case(*cRes PQ*)
from ‹$\Psi \rhd P \parallel Q \longmapsto \alpha \prec PQ$› ‹*bn* $\alpha \sharp* \Psi$› ‹*bn* $\alpha \sharp* P$› ‹*bn* $\alpha \sharp* Q$› ‹*bn* $\alpha \sharp*$ *subject* α›
show *?case*
proof(*induct rule*: *parCases*[**where** $C{=}x$])

case(*cPar1* P' A_Q Ψ_Q)
note ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto \alpha \prec P'$›
moreover with ‹$x \sharp P$› ‹$x \sharp \alpha$› ‹$bn\ \alpha \sharp* subject\ \alpha$› ‹*distinct*($bn\ \alpha$)›
have $x \sharp P'$ **by**(*force dest*: *freeFreshDerivative*)
with ‹*extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$› **have** *extractFrame*($(\!|\nu x|\!)\, Q$) = $\langle (x \# A_Q), \Psi_Q \rangle$
by *simp*
moreover from ‹$bn\ \alpha \sharp* Q$› **have** $bn\ \alpha \sharp* (\!|\nu x|\!)\, Q$ **by**(*simp add*: *fresh-star-def abs-fresh*)
moreover from ‹$x \sharp \Psi$› ‹$A_Q \sharp* \Psi$› **have** $(x \# A_Q) \sharp* \Psi$ **by** *simp*
moreover from ‹$x \sharp P$› ‹$A_Q \sharp* P$› **have** $(x \# A_Q) \sharp* P$ **by** *simp*
moreover from ‹$x \sharp \alpha$› ‹$A_Q \sharp* \alpha$› **have** $(x \# A_Q) \sharp* \alpha$ **by** *simp*
ultimately have $\Psi \rhd P \parallel (\!|\nu x|\!)\, Q \longmapsto \alpha \prec (P' \parallel (\!|\nu x|\!)\, Q)$
by(*rule Par1*)
moreover from ‹$x \sharp P'$› ‹$x \sharp \Psi$› **have** $(\Psi, (\!|\nu * []|\!)(P' \parallel ((\!|\nu x|\!)\, Q)), (\!|\nu x|\!)((\!|\nu * []|\!)(P' \parallel Q))) \in Rel$
by(*intro C2*) *auto*
ultimately show *?case*
by *force*
next
case(*cPar2* Q' A_P Ψ_P)
have *FrP*: *extractFrame* $P = \langle A_P, \Psi_P \rangle$ **by** *fact*
with ‹$x \sharp P$› ‹$A_P \sharp* x$› **have** $x \sharp \Psi_P$
apply(*drule-tac extractFrameFresh*)
by(*simp add*: *frameResChainFresh*) (*simp add*: *fresh-def name-list-supp*)
from ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto \alpha \prec Q'$› ‹$x \sharp \Psi$› ‹$x \sharp \Psi_P$› ‹$x \sharp \alpha$›
have $\Psi \otimes \Psi_P \rhd (\!|\nu x|\!)\, Q \longmapsto \alpha \prec (\!|\nu x|\!)\, Q'$
by(*intro Scope*) *auto*
moreover note *FrP* ‹$bn\ \alpha \sharp* P$› ‹$A_P \sharp* \Psi$›
moreover from ‹$A_P \sharp* Q$› **have** $A_P \sharp* (\!|\nu x|\!)\, Q$ **by**(*simp add*: *fresh-star-def abs-fresh*)
ultimately have $\Psi \rhd P \parallel (\!|\nu x|\!)\, Q \longmapsto \alpha \prec (P \parallel (\!|\nu x|\!)\, Q')$ **using** ‹$A_P \sharp* \alpha$›
by(*rule Par2*)
moreover from ‹$x \sharp P$› ‹$x \sharp \Psi$› **have** $(\Psi, (\!|\nu * []|\!)(P \parallel ((\!|\nu x|\!)\, Q')), (\!|\nu x|\!)((\!|\nu * []|\!)(P \parallel Q'))) \in Rel$
by(*intro C2*) *auto*
ultimately show *?case*
by *force*
next
case(*cComm1* Ψ_Q M N P' A_P Ψ_P K *xvec* Q' A_Q)
have *PTrans*: $\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|N|\!) \prec P'$ **and** *FrP*: *extractFrame* $P = \langle A_P, \Psi_P \rangle$ **by** *fact+*
have *QTrans*: $\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|\nu * xvec|\!)\langle N \rangle \prec Q'$ **and** *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$ **by** *fact+*
from *FrP* ‹$x \sharp P$› **have** $x \sharp \langle A_P, \Psi_P \rangle$
by(*auto dest*: *extractFrameFresh*)
with ‹$A_P \sharp* x$› **have** $x \sharp \Psi_P$ **by**(*simp add*: *frameResChainFresh*) (*simp add*: *fresh-def name-list-supp*)
from *PTrans FrP* ‹*distinct* A_P› ‹$x \sharp P$› ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* \Psi_Q$› ‹$A_P \sharp* x$›

$\langle A_P \sharp* P\rangle$ $\langle A_P \sharp* M\rangle$ $\langle A_Q \sharp* P\rangle$ $\langle A_P \sharp* A_Q\rangle$
obtain M' **where** $MeqM'$: $(\Psi \otimes \Psi_Q) \otimes \Psi_P \vdash M \leftrightarrow M'$ **and** $x \sharp M'$ **and** $A_Q \sharp* M'$
by(*elim inputObtainPrefix*[**where** $B{=}x\#A_Q$]) (*assumption* | *force simp add*: *fresh-star-list-cons*)+
from $MeqM'$ **have** $MeqM'$: $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow M'$
by(*metis statEqEnt Associativity Composition Commutativity*)
with $\langle \Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K\rangle$ **have** $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash K \leftrightarrow M'$
by(*blast intro*: *chanEqTrans chanEqSym*)
then have $(\Psi \otimes \Psi_P) \otimes \Psi_Q \vdash K \leftrightarrow M'$
by(*metis statEqEnt Associativity Composition Commutativity*)
with *QTrans FrQ* ‹*distinct* A_Q› **have** *QTrans*: $\Psi \otimes \Psi_P \rhd Q \longmapsto M'(\!|\nu{*}xvec|\!)\langle N\rangle \prec Q'$ **using** $\langle A_Q \sharp* \Psi\rangle$ $\langle A_Q \sharp* \Psi_P\rangle$ $\langle A_Q \sharp* Q\rangle$ $\langle A_Q \sharp* K\rangle$ $\langle A_Q \sharp* M'\rangle$
by(*elim outputRenameSubject*) (*assumption* | *force*)+
show *?case*
proof(*cases* $x \in supp\ N$)
note *PTrans FrP*
moreover assume $x \in supp\ N$
then have $\Psi \otimes \Psi_P \rhd (\!|\nu x|\!)Q \longmapsto M'(\!|\nu{*}([]@(x\#xvec))|\!)\langle N\rangle \prec Q'$ **using** *QTrans* $\langle x \sharp \Psi\rangle$ $\langle x \sharp \Psi_P\rangle$ $\langle x \sharp M'\rangle$ $\langle xvec \sharp* x\rangle$
by(*intro Open*) (*assumption* | *force simp add*: *fresh-list-nil*)+
then have *QTrans*: $\Psi \otimes \Psi_P \rhd (\!|\nu x|\!)Q \longmapsto M'(\!|\nu{*}(x\#xvec)|\!)\langle N\rangle \prec Q'$ **by** *simp*
moreover from ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› **have** $extractFrame((\!|\nu x|\!)Q) = \langle (x\#A_Q), \Psi_Q\rangle$
by *simp*
moreover note $MeqM'$ $\langle A_P \sharp* \Psi\rangle$ $\langle A_P \sharp* P\rangle$
moreover from $\langle A_P \sharp* Q\rangle$ **have** $A_P \sharp* ((\!|\nu x|\!)Q)$ **by**(*simp add*: *fresh-star-def abs-fresh*)
moreover from $\langle A_P \sharp* A_Q\rangle$ $\langle A_P \sharp* x\rangle$ **have** $A_P \sharp* (x\#A_Q)$
by(*simp add*: *fresh-star-def fresh-list-cons*) (*auto simp add*: *fresh-def name-list-supp*)
moreover from $\langle x \sharp \Psi\rangle$ $\langle A_Q \sharp* \Psi\rangle$ **have** $(x\#A_Q) \sharp* \Psi$ **by** *simp*
moreover from $\langle x \sharp P\rangle$ $\langle A_Q \sharp* P\rangle$ **have** $(x\#A_Q) \sharp* P$ **by** *simp*
moreover from $\langle x \sharp M'\rangle$ $\langle A_Q \sharp* M'\rangle$ **have** $(x\#A_Q) \sharp* M'$ **by** *simp*
moreover from $\langle A_Q \sharp* Q\rangle$ **have** $(x\#A_Q) \sharp* ((\!|\nu x|\!)Q)$ **by**(*simp add*: *fresh-star-def abs-fresh*)
moreover from $\langle x \sharp P\rangle$ $\langle xvec \sharp* P\rangle$ **have** $(x\#xvec) \sharp* P$ **by**(*simp*)
ultimately have $\Psi \rhd P \parallel (\!|\nu x|\!)Q \longmapsto \tau \prec (\!|\nu{*}(x\#xvec)|\!)(P' \parallel Q')$ **using** $\langle A_P \sharp* M\rangle$
by(*intro Comm1*) (*assumption* | *simp*)+
moreover have $(\Psi, (\!|\nu x|\!)((\!|\nu{*}xvec|\!)(P' \parallel Q')), (\!|\nu x|\!)((\!|\nu{*}xvec|\!)(P' \parallel Q'))) \in Rel$ **by**(*rule C1*)
ultimately show *?case* **by** *force*
next
note *PTrans FrP*
moreover assume $x \notin supp\ N$
then have $x \sharp N$ **by**(*simp add*: *fresh-def*)
with *QTrans* $\langle x \sharp \Psi\rangle$ $\langle x \sharp \Psi_P\rangle$ $\langle x \sharp M'\rangle$ $\langle xvec \sharp* x\rangle$

have *QTrans*: $\Psi \otimes \Psi_P \rhd (\!|\nu x|\!)\, Q \longmapsto M'(\!|\nu{*}xvec|\!)\langle N\rangle \prec (\!|\nu x|\!)\, Q'$
by(*intro Scope*) (*assumption* | *force*)+
moreover from *PTrans* ‹$x \sharp P$› ‹$x \sharp N$› **have** $x \sharp P'$ **by**(*rule inputFreshDerivative*)
with ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› **have** *extractFrame*$((\!|\nu x|\!)\, Q) = \langle (x\#A_Q), \Psi_Q\rangle$
by *simp*
moreover note *MeqM'* ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* P$›
moreover from ‹$A_P \sharp* Q$› **have** $A_P \sharp* ((\!|\nu x|\!)\, Q)$ **by**(*simp add*: *fresh-star-def abs-fresh*)
moreover from ‹$A_P \sharp* A_Q$› ‹$A_P \sharp* x$› **have** $A_P \sharp* (x\#A_Q)$
by(*simp add*: *fresh-star-def fresh-list-cons*) (*auto simp add*: *fresh-def name-list-supp*)
moreover from ‹$x \sharp \Psi$› ‹$A_Q \sharp* \Psi$› **have** $(x\#A_Q) \sharp* \Psi$ **by** *simp*
moreover from ‹$x \sharp P$› ‹$A_Q \sharp* P$› **have** $(x\#A_Q) \sharp* P$ **by** *simp*
moreover from ‹$x \sharp M'$› ‹$A_Q \sharp* M'$› **have** $(x\#A_Q) \sharp* M'$ **by** *simp*
moreover from ‹$A_Q \sharp* Q$› **have** $(x\#A_Q) \sharp* ((\!|\nu x|\!)\, Q)$ **by**(*simp add*: *fresh-star-def abs-fresh*)
moreover from ‹$x \sharp P$› ‹*xvec* $\sharp* P$› **have** $(x\#xvec) \sharp* P$ **by**(*simp*)
ultimately have $\Psi \rhd P \parallel (\!|\nu x|\!)\, Q \longmapsto \tau \prec (\!|\nu{*}xvec|\!)(P' \parallel (\!|\nu x|\!)\, Q')$ **using** ‹$A_P \sharp* M$›
by(*intro Comm1*) (*assumption* | *simp*)+
moreover from ‹$x \sharp \Psi$› ‹$x \sharp P'$› ‹*xvec* $\sharp* \Psi$› **have** $(\Psi, (\!|\nu{*}xvec|\!)(P' \parallel (\!|\nu x|\!)\, Q'), (\!|\nu x|\!)((\!|\nu{*}xvec|\!)(P' \parallel Q'))) \in Rel$ **by**(*rule C2*)
ultimately show *?case* **by** *blast*
qed
next
case(*cComm2* Ψ_Q M *xvec* N P' A_P Ψ_P K Q' A_Q)
have *PTrans*: $\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P'$ **and** *FrP*: *extractFrame* $P = \langle A_P, \Psi_P\rangle$ **by** *fact*+
have *QTrans*: $\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|N|\!) \prec Q'$ **and** *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$ **by** *fact*+
from *FrP* ‹$x \sharp P$› **have** $x \sharp \langle A_P, \Psi_P\rangle$
by(*auto dest*: *extractFrameFresh*)
with ‹$A_P \sharp* x$› **have** $x \sharp \Psi_P$ **by**(*simp add*: *frameResChainFresh*) (*simp add*: *fresh-def name-list-supp*)
from *PTrans*
have $\Psi \otimes \Psi_Q \rhd P \longmapsto ROut\ M\ ((\!|\nu{*}xvec|\!)N \prec' P')$
by(*simp add*: *residualInject*)
with *FrP* ‹*distinct* A_P› ‹$x \sharp P$› ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* \Psi_Q$› ‹$A_P \sharp* x$› ‹$A_P \sharp* P$› ‹$A_P \sharp* M$› ‹$A_Q \sharp* P$› ‹$A_P \sharp* A_Q$› ‹*xvec* $\sharp* M$› ‹*distinct xvec*›
obtain M' **where** *MeqM'*: $(\Psi \otimes \Psi_Q) \otimes \Psi_P \vdash M \leftrightarrow M'$ **and** $x \sharp M'$ **and** $A_Q \sharp* M'$
by(*elim outputObtainPrefix*[**where** $B{=}x\#A_Q$]) (*assumption* | *force simp add*: *fresh-star-list-cons*)+
from *MeqM'* **have** *MeqM'*: $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow M'$
by(*metis statEqEnt Associativity Commutativity Composition*)
with ‹$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K$› **have** $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash K \leftrightarrow M'$
by(*blast intro*: *chanEqTrans chanEqSym*)

then have $(\Psi \otimes \Psi_P) \otimes \Psi_Q \vdash K \leftrightarrow M'$
by(*metis statEqEnt Associativity Commutativity Composition*)
with *QTrans FrQ* ‹*distinct* A_Q› **have** *QTrans*: $\Psi \otimes \Psi_P \rhd Q \longmapsto M'(\!|N|\!) \prec Q'$ **using** ‹$A_Q \sharp* \Psi$› ‹$A_Q \sharp* \Psi_P$› ‹$A_Q \sharp* Q$› ‹$A_Q \sharp* K$› ‹$A_Q \sharp* M'$›
by(*auto intro*: *inputRenameSubject*)

from *PTrans* ‹$x \sharp P$› ‹*xvec* $\sharp*$ x› ‹*distinct xvec*› ‹*xvec* $\sharp*$ M›
have $x \sharp N$ **and** $x \sharp P'$ **by**(*force intro*: *outputFreshDerivative*)+
from *QTrans* ‹$x \sharp \Psi$› ‹$x \sharp \Psi_P$› ‹$x \sharp M'$› ‹$x \sharp N$› **have** *QTrans*: $\Psi \otimes \Psi_P \rhd (\!|\nu x|\!) Q \longmapsto M'(\!|N|\!) \prec (\!|\nu x|\!) Q'$
by(*intro Scope*) (*assumption* | *force*)+

note *PTrans FrP QTrans*
moreover with ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› **have** *extractFrame*$((\!|\nu x|\!) Q) = \langle(x\#A_Q), \Psi_Q\rangle$
by *simp*
moreover note *MeqM'* ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* P$›
moreover from ‹$A_P \sharp* Q$› **have** $A_P \sharp* ((\!|\nu x|\!) Q)$ **by**(*simp add*: *fresh-star-def abs-fresh*)
moreover from ‹$A_P \sharp* A_Q$› ‹$A_P \sharp* x$› **have** $A_P \sharp* (x\#A_Q)$
by(*simp add*: *fresh-star-def fresh-list-cons*) (*auto simp add*: *fresh-def name-list-supp*)
moreover from ‹$x \sharp \Psi$› ‹$A_Q \sharp* \Psi$› **have** $(x\#A_Q) \sharp* \Psi$ **by** *simp*
moreover from ‹$x \sharp P$› ‹$A_Q \sharp* P$› **have** $(x\#A_Q) \sharp* P$ **by** *simp*
moreover from ‹$x \sharp M'$› ‹$A_Q \sharp* M'$› **have** $(x\#A_Q) \sharp* M'$ **by** *simp*
moreover from ‹$A_Q \sharp* Q$› **have** $(x\#A_Q) \sharp* ((\!|\nu x|\!) Q)$ **by**(*simp add*: *fresh-star-def abs-fresh*)
moreover from ‹*xvec* $\sharp*$ Q› **have** $(x\#xvec) \sharp* ((\!|\nu x|\!) Q)$ **by**(*simp add*: *abs-fresh fresh-star-def*)
ultimately have $\Psi \rhd P \parallel (\!|\nu x|\!) Q \longmapsto \tau \prec (\!|\nu*xvec|\!)(P' \parallel (\!|\nu x|\!) Q')$ **using** ‹$A_P \sharp* M$›
by(*intro Comm2*) (*assumption* | *simp*)+
moreover from ‹$x \sharp \Psi$› ‹$x \sharp P'$› ‹*xvec* $\sharp*$ Ψ› **have** $(\Psi, (\!|\nu*xvec|\!)(P' \parallel (\!|\nu x|\!) Q'), (\!|\nu x|\!)((\!|\nu*xvec|\!)(P' \parallel Q'))) \in Rel$ **by**(*rule C2*)
ultimately show *?case* **by** *blast*
next
case(*cBrMerge* Ψ_Q M N P' A_P Ψ_P Q' A_Q)
have *PTrans*: $\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$ **and** *FrP*: *extractFrame* $P = \langle A_P, \Psi_P\rangle$ **by** *fact*+
have *QTrans*: $\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$ **and** *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$ **by** *fact*+
from *FrP* ‹$x \sharp P$› **have** $x \sharp \langle A_P, \Psi_P\rangle$
by(*auto dest*: *extractFrameFresh*)
with ‹$A_P \sharp* x$› **have** $x \sharp \Psi_P$
by(*simp add*: *frameResChainFresh*) (*simp add*: *fresh-def name-list-supp*)
from ‹$x \sharp \alpha$› ‹$\alpha = ¿M(\!|N|\!)$› **have** $x \sharp M$ **and** $x \sharp N$ **by** *simp*+
from *PTrans* ‹$x \sharp P$› ‹$x \sharp N$›
have $x \sharp P'$
by(*rule brinputFreshDerivative*)

from *QTrans* ‹$x \sharp \Psi$› ‹$x \sharp \Psi_P$› ‹$x \sharp M$› ‹$x \sharp N$› **have** *QTrans*: $\Psi \otimes \Psi_P \rhd (\!|\nu x|\!)\, Q \longmapsto ¿ M (\!|N|\!) \prec (\!|\nu x|\!)\, Q'$
by(*intro Scope*) (*assumption* | *force*)+

note *PTrans FrP QTrans*
moreover with ‹*extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$› **have** *extractFrame*$((\!|\nu x|\!)\, Q) = \langle (x \# A_Q), \Psi_Q \rangle$
by *simp*
moreover note ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* P$›
moreover from ‹$A_P \sharp* Q$› **have** $A_P \sharp* ((\!|\nu x|\!)\, Q)$ **by**(*simp add*: *fresh-star-def abs-fresh*)
moreover from ‹$A_P \sharp* A_Q$› ‹$A_P \sharp* x$› **have** $A_P \sharp* (x \# A_Q)$
by(*simp add*: *fresh-star-def fresh-list-cons*) (*auto simp add*: *fresh-def name-list-supp*)
moreover from ‹$x \sharp \Psi$› ‹$A_Q \sharp* \Psi$› **have** $(x \# A_Q) \sharp* \Psi$ **by** *simp*
moreover from ‹$x \sharp P$› ‹$A_Q \sharp* P$› **have** $(x \# A_Q) \sharp* P$ **by** *simp*
moreover from ‹$x \sharp M$› ‹$A_Q \sharp* M$› **have** $(x \# A_Q) \sharp* M$ **by** *simp*
moreover from ‹$A_Q \sharp* Q$› **have** $(x \# A_Q) \sharp* ((\!|\nu x|\!)\, Q)$ **by**(*simp add*: *fresh-star-def abs-fresh*)
ultimately have *Trans*: $\Psi \rhd P \parallel (\!|\nu x|\!)\, Q \longmapsto ¿ M (\!|N|\!) \prec (P' \parallel (\!|\nu x|\!)\, Q')$ **using** ‹$A_P \sharp* M$›
by(*intro BrMerge*) (*assumption* | *simp*)+
from ‹$x \sharp \Psi$› ‹$x \sharp P'$› **have** $(\Psi, (P' \parallel ((\!|\nu * [x]|\!)\, Q')), (\!|\nu * [x]|\!)(P' \parallel Q')) \in Rel$
by(*intro C3*) *simp*+
then have *Relation*: $(\Psi, (P' \parallel ((\!|\nu x|\!)\, Q')), (\!|\nu x|\!)(P' \parallel Q')) \in Rel$ **by** *simp*
with *Trans Relation* **show** *?case* **by** *blast*
next
case(*cBrComm1* Ψ_Q M N P' A_P Ψ_P *xvec* Q' A_Q)
from ‹$x \sharp \alpha$› ‹$¡ M (\!|\nu * xvec|\!) \langle N \rangle = \alpha$› **have** $x \sharp M$ **and** $x \sharp$ *xvec* **and** $x \sharp N$ **by** *force*+
have *PTrans*: $\Psi \otimes \Psi_Q \rhd P \longmapsto ¿ M (\!|N|\!) \prec P'$ **and** *FrP*: *extractFrame* $P = \langle A_P, \Psi_P \rangle$ **by** *fact*+
have *QTrans*: $\Psi \otimes \Psi_P \rhd Q \longmapsto ¡ M (\!|\nu * xvec|\!) \langle N \rangle \prec Q'$ **and** *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$ **by** *fact*+
from *FrP* ‹$x \sharp P$› **have** $x \sharp \langle A_P, \Psi_P \rangle$
by(*auto dest*: *extractFrameFresh*)
with ‹$A_P \sharp* x$› **have** $x \sharp \Psi_P$
by(*simp add*: *frameResChainFresh*) (*simp add*: *fresh-def name-list-supp*)
show *?case*
proof(*cases* $x \in$ *supp* N)
note *PTrans FrP*
moreover assume $x \in$ *supp* N
with ‹$x \sharp N$› **have** *False*
by(*simp add*: *fresh-def*)
then show *?case*
by *simp*
next
note *PTrans FrP*
moreover assume $x \notin$ *supp* N

from *QTrans* ‹$x \sharp \Psi$› ‹$x \sharp \Psi_P$› ‹$x \sharp M$› ‹$x \sharp N$› ‹$x \sharp xvec$›
have *QTrans*: $\Psi \otimes \Psi_P \rhd (\!|\nu x|\!)\, Q \longmapsto ¡M(\!|\nu{*}xvec|\!)\langle N\rangle \prec (\!|\nu x|\!)\, Q'$
by(*intro Scope*) (*assumption* | *force*)+
moreover from *PTrans* ‹$x \sharp P$› ‹$x \sharp N$› **have** $x \sharp P'$ **by**(*rule brinput-FreshDerivative*)
with ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› **have** *extractFrame*$((\!|\nu x|\!)\, Q) = \langle (x\#A_Q), \Psi_Q\rangle$
by *simp*
moreover note ‹$A_P \sharp{*} \Psi$› ‹$A_P \sharp{*} P$›
moreover from ‹$A_P \sharp{*} Q$› **have** $A_P \sharp{*} ((\!|\nu x|\!)\, Q)$ **by**(*simp add*: *fresh-star-def abs-fresh*)
moreover from ‹$A_P \sharp{*} A_Q$› ‹$A_P \sharp{*} x$› **have** $A_P \sharp{*} (x\#A_Q)$
by(*simp add*: *fresh-star-def fresh-list-cons*) (*auto simp add*: *fresh-def name-list-supp*)
moreover from ‹$x \sharp \Psi$› ‹$A_Q \sharp{*} \Psi$› **have** $(x\#A_Q) \sharp{*} \Psi$ **by** *simp*
moreover from ‹$x \sharp P$› ‹$A_Q \sharp{*} P$› **have** $(x\#A_Q) \sharp{*} P$ **by** *simp*
moreover from ‹$x \sharp M$› ‹$A_Q \sharp{*} M$› **have** $(x\#A_Q) \sharp{*} M$ **by** *simp*
moreover from ‹$A_Q \sharp{*} Q$› **have** $(x\#A_Q) \sharp{*} ((\!|\nu x|\!)\, Q)$ **by**(*simp add*: *fresh-star-def abs-fresh*)
moreover from ‹$x \sharp P$› ‹$xvec \sharp{*} P$› **have** $(x\#xvec) \sharp{*} P$ **by**(*simp*)
ultimately have *Trans*: $\Psi \rhd P \parallel (\!|\nu x|\!)\, Q \longmapsto ¡M(\!|\nu{*}xvec|\!)\langle N\rangle \prec (P' \parallel (\!|\nu x|\!)\, Q')$ **using** ‹$A_P \sharp{*} M$›
by(*intro BrComm1*) (*assumption* | *simp*)+
from ‹$x \sharp \Psi$› ‹$x \sharp P'$› **have** $(\Psi, (P' \parallel ((\!|\nu{*}[x]|\!)\, Q')), (\!|\nu{*}[x]|\!)(P' \parallel Q')) \in Rel$
by(*intro C3*) *simp*+
then have *Relation*: $(\Psi, (P' \parallel ((\!|\nu x|\!)\, Q')), (\!|\nu x|\!)(P' \parallel Q')) \in Rel$ **by** *simp*
from *Trans Relation* **show** *?case* **by** *blast*
qed
next
case(*cBrComm2* Ψ_Q M *xvec* N P' A_P Ψ_P Q' A_Q)
from ‹$x \sharp \alpha$› ‹$¡M(\!|\nu{*}xvec|\!)\langle N\rangle = \alpha$› **have** $x \sharp M$ **and** $x \sharp xvec$ **and** $x \sharp N$ **by** *force*+
have *PTrans*: $\Psi \otimes \Psi_Q \rhd P \longmapsto ¡M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P'$ **and** *FrP*: *extractFrame* $P = \langle A_P, \Psi_P\rangle$ **by** *fact*+
have *QTrans*: $\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$ **and** *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$ **by** *fact*+
from *FrP* ‹$x \sharp P$› **have** $x \sharp \langle A_P, \Psi_P\rangle$
by(*auto dest*: *extractFrameFresh*)
with ‹$A_P \sharp{*} x$› **have** $x \sharp \Psi_P$
by(*simp add*: *frameResChainFresh*) (*simp add*: *fresh-def name-list-supp*)
from *PTrans* ‹$x \sharp P$› ‹$x \sharp xvec$› ‹*distinct xvec*› ‹$xvec \sharp{*} M$›
have $x \sharp N$ **and** $x \sharp P'$ **by**(*force intro*: *broutputFreshDerivative*)+
from *QTrans* ‹$x \sharp \Psi$› ‹$x \sharp \Psi_P$› ‹$x \sharp M$› ‹$x \sharp N$› **have** *QTrans*: $\Psi \otimes \Psi_P \rhd (\!|\nu x|\!)\, Q \longmapsto ¿M(\!|N|\!) \prec (\!|\nu x|\!)\, Q'$
by(*intro Scope*) (*assumption* | *force*)+

note *PTrans FrP QTrans*
moreover with ‹*extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$› **have** *extractFrame*$((\!|\nu x|\!)\, Q)$

$= \langle (x\#A_Q),\ \Psi_Q \rangle$
by *simp*
moreover note ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* P$›
moreover from ‹$A_P \sharp* Q$› **have** $A_P \sharp* (\langle\!|\nu x|\!\rangle Q)$ **by**(*simp add*: *fresh-star-def abs-fresh*)
moreover from ‹$A_P \sharp* A_Q$› ‹$A_P \sharp* x$› **have** $A_P \sharp* (x\#A_Q)$
by(*simp add*: *fresh-star-def fresh-list-cons*) (*auto simp add*: *fresh-def name-list-supp*)
moreover from ‹$x \sharp \Psi$› ‹$A_Q \sharp* \Psi$› **have** $(x\#A_Q) \sharp* \Psi$ **by** *simp*
moreover from ‹$x \sharp P$› ‹$A_Q \sharp* P$› **have** $(x\#A_Q) \sharp* P$ **by** *simp*
moreover from ‹$x \sharp M$› ‹$A_Q \sharp* M$› **have** $(x\#A_Q) \sharp* M$ **by** *simp*
moreover from ‹$A_Q \sharp* Q$› **have** $(x\#A_Q) \sharp* (\langle\!|\nu x|\!\rangle Q)$ **by**(*simp add*: *fresh-star-def abs-fresh*)
moreover from ‹*xvec* $\sharp* Q$› **have** $(x\#xvec) \sharp* (\langle\!|\nu x|\!\rangle Q)$ **by**(*simp add*: *abs-fresh fresh-star-def*)
ultimately have *Trans*: $\Psi \rhd P \parallel \langle\!|\nu x|\!\rangle Q \longmapsto ¡M\langle\!|\nu{*}xvec|\!\rangle\langle N\rangle \prec (P' \parallel \langle\!|\nu x|\!\rangle Q')$ **using** ‹$A_P \sharp* M$›
by(*intro BrComm2*) (*assumption* | *simp*)+
from ‹$x \sharp \Psi$› ‹$x \sharp P'$› **have** $(\Psi, (P' \parallel (\langle\!|\nu{*}[x]|\!\rangle Q')), \langle\!|\nu{*}[x]|\!\rangle(P' \parallel Q')) \in Rel$
by(*intro C3*) *simp*+
then have *Relation*: $(\Psi, (P' \parallel \langle\!|\nu x|\!\rangle Q'), \langle\!|\nu x|\!\rangle(P' \parallel Q')) \in Rel$ **by** *simp*
from *Trans Relation* **show** *?case* **by** *blast*
qed
next
case(*cBrClose M xvec N PQ*)
from ‹*xvec* $\sharp* (P \parallel Q)$› **have** *xvec* $\sharp* P$ **and** *xvec* $\sharp* Q$ **by** *simp*+
note ‹$\Psi \rhd P \parallel Q \longmapsto ¡M\langle\!|\nu{*}xvec|\!\rangle\langle N\rangle \prec PQ$› ‹*xvec* $\sharp* \Psi$› ‹*xvec* $\sharp* P$› ‹*xvec* $\sharp* Q$› ‹*xvec* $\sharp* M$›
then show *?case*
proof(*induct rule*: *parBrOutputCases*[**where** $C{=}x$])
case(*cPar1 P' A_Q Ψ_Q*)
from ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto ¡M\langle\!|\nu{*}xvec|\!\rangle\langle N\rangle \prec P'$› ‹*xvec* $\sharp* M$› ‹$x \sharp P$›
have $x \sharp M$
by(*rule brOutputFreshSubject*)
with ‹$x \in supp\ M$› **have** *False*
by(*simp add*: *fresh-def*)
then show *?case*
by *simp*
next
case(*cPar2 Q' A_P Ψ_P*)
from ‹$A_P \sharp* x$› **have** $x \sharp A_P$ **by** *simp*
with ‹$x \sharp P$› ‹*extractFrame* $P = \langle A_P, \Psi_P \rangle$›
have $x \sharp \Psi_P$
apply(*drule-tac extractFrameFresh*)
by(*simp add*: *frameResChainFresh*) (*simp add*: *fresh-def name-list-supp*)
from ‹$x \sharp \Psi$› ‹$x \sharp \Psi_P$› **have** $x \sharp (\Psi \otimes \Psi_P)$ **by** *force*
from ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto ¡M\langle\!|\nu{*}xvec|\!\rangle\langle N\rangle \prec Q'$› ‹$x \in supp\ M$› ‹$x \sharp (\Psi \otimes \Psi_P)$›
have *QTrans*: $\Psi \otimes \Psi_P \rhd \langle\!|\nu x|\!\rangle Q \longmapsto \tau \prec \langle\!|\nu x|\!\rangle(\langle\!|\nu{*}xvec|\!\rangle Q')$

by(*rule BrClose*)
with ‹*extractFrame* $P = \langle A_P, \Psi_P\rangle$› ‹$A_P \sharp* \Psi$› ‹$x \sharp A_P$› ‹$A_P \sharp* Q$›
have *Trans*: $\Psi \rhd (P \parallel (\!|\nu x|\!) Q) \longmapsto \tau \prec (P \parallel (\!|\nu x|\!)((\!|\nu * xvec|\!) Q'))$
by(*intro Par2*) (*assumption* | *simp*)+
from ‹$x \sharp P$› ‹$xvec \sharp* P$› **have** $(x \# xvec) \sharp* P$ **by** *simp*
moreover from ‹$x \sharp \Psi$› ‹$xvec \sharp* \Psi$› **have** $(x \# xvec) \sharp* \Psi$ **by** *simp*
ultimately have $(\Psi, (P \parallel ((\!|\nu * (x \# xvec)|\!) Q')), ((\!|\nu * (x \# xvec)|\!) P \parallel Q')) \in Rel$
by(*rule C3*)
then have *Relation*: $(\Psi, (P \parallel (\!|\nu x|\!)((\!|\nu * xvec|\!) Q')), (\!|\nu x|\!)((\!|\nu * xvec|\!) P \parallel Q')) \in Rel$
by *simp*
from *Trans Relation*
show *?case*
by *blast*
next
case(*cBrComm1* Ψ_Q P' A_P Ψ_P Q' A_Q)
from ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$› ‹$x \sharp P$›
have $x \sharp M$
by(*rule brInputFreshSubject*)
with ‹$x \in supp\ M$› **have** *False*
by(*simp add*: *fresh-def*)
then show *?case*
by *simp*
next
case(*cBrComm2* Ψ_Q P' A_P Ψ_P Q' A_Q)
from ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$› ‹$xvec \sharp* M$› ‹$x \sharp P$›
have $x \sharp M$
by(*rule brOutputFreshSubject*)
with ‹$x \in supp\ M$› **have** *False*
by(*simp add*: *fresh-def*)
then show *?case*
by *simp*
qed
qed
qed
qed

lemma *simParComm*:
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 *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 $\Psi \rhd P \parallel Q \rightsquigarrow[Rel]\ Q \parallel P$
using ‹*eqvt Rel*›
proof(*induct rule*: *simI*[*of* - - - - ()])
case(*cSim* α PQ)
from ‹*bn* $\alpha \sharp* (P \parallel Q)$› **have** *bn* $\alpha \sharp* Q$ **and** *bn* $\alpha \sharp* P$ **by** *simp*+
with ‹$\Psi \rhd Q \parallel P \longmapsto \alpha \prec PQ$› ‹*bn* $\alpha \sharp* \Psi$› **show** *?case* **using** ‹*bn* $\alpha \sharp*$ *subject* α›
proof(*induct rule*: *parCases*[**where** C=()])
case(*cPar1* Q' A_P Ψ_P)
from ‹$\Psi \otimes \Psi_P \rhd Q \longmapsto \alpha \prec Q'$› ‹*extractFrame* $P = \langle A_P, \Psi_P \rangle$› ‹*bn* $\alpha \sharp* P$› ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* Q$› ‹$A_P \sharp* \alpha$›
have $\Psi \rhd P \parallel Q \longmapsto \alpha \prec (P \parallel Q')$ **by**(*rule Par2*)
moreover have $(\Psi, P \parallel Q', Q' \parallel P) \in Rel$ **by**(*rule C1*)
ultimately show *?case* **by** *blast*
next
case(*cPar2* P' A_Q Ψ_Q)
from ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto \alpha \prec P'$› ‹*extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$› ‹*bn* $\alpha \sharp* Q$› ‹$A_Q \sharp* \Psi$› ‹$A_Q \sharp* P$› ‹$A_Q \sharp* \alpha$›
have $\Psi \rhd P \parallel Q \longmapsto \alpha \prec (P' \parallel Q)$ **by**(*rule Par1*)
moreover have $(\Psi, P' \parallel Q, Q \parallel P') \in Rel$ **by**(*rule C1*)
ultimately show *?case* **by** *blast*
next
case(*cComm1* Ψ_P M N Q' A_Q Ψ_Q K *xvec* P' A_P)
note ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto K(\!|\nu* xvec|\!)\langle N \rangle \prec P'$› ‹*extractFrame* $P = \langle A_P, \Psi_P \rangle$›
‹$\Psi \otimes \Psi_P \rhd Q \longmapsto M(\!|N|\!) \prec Q'$› ‹*extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$›
moreover from ‹$\Psi \otimes \Psi_Q \otimes \Psi_P \vdash M \leftrightarrow K$›
have $\Psi \otimes \Psi_Q \otimes \Psi_P \vdash K \leftrightarrow M$
by(*rule chanEqSym*)
then have $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash K \leftrightarrow M$
by(*blast intro*: *statEqEnt compositionSym Commutativity*)
ultimately have $\Psi \rhd P \parallel Q \longmapsto \tau \prec (\!|\nu* xvec|\!)(P' \parallel Q')$
using ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_P \sharp* Q$› ‹$A_Q \sharp* A_P$› ‹$A_Q \sharp* \Psi$› ‹$A_Q \sharp* P$› ‹$A_Q \sharp* Q$› ‹*xvec* $\sharp* Q$› ‹$A_P \sharp* K$› ‹$A_Q \sharp* M$›
by(*intro Comm2*) (*assumption* | *simp*)+
moreover have $(\Psi, P' \parallel Q', Q' \parallel P') \in Rel$ **by**(*rule C1*)
then have $(\Psi, (\!|\nu* xvec|\!)(P' \parallel Q'), (\!|\nu* xvec|\!)(Q' \parallel P')) \in Rel$ **using** ‹*xvec* $\sharp* \Psi$› **by**(*rule C2*)
ultimately show *?case* **by** *blast*
next
case(*cComm2* Ψ_P M *xvec* N Q' A_Q Ψ_Q K P' A_P)
note ‹$\Psi \otimes \Psi_Q \rhd P \longmapsto K(\!|N|\!) \prec P'$› ‹*extractFrame* $P = \langle A_P, \Psi_P \rangle$›
‹$\Psi \otimes \Psi_P \rhd Q \longmapsto M(\!|\nu* xvec|\!)\langle N \rangle \prec Q'$› ‹*extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$›
moreover from ‹$\Psi \otimes \Psi_Q \otimes \Psi_P \vdash M \leftrightarrow K$›
have $\Psi \otimes \Psi_Q \otimes \Psi_P \vdash K \leftrightarrow M$
by(*rule chanEqSym*)
then have $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash K \leftrightarrow M$
by(*blast intro*: *statEqEnt compositionSym Commutativity*)
ultimately have $\Psi \rhd P \parallel Q \longmapsto \tau \prec (\!|\nu* xvec|\!)(P' \parallel Q')$
using ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* P$› ‹$A_P \sharp* Q$› ‹$A_Q \sharp* A_P$› ‹$A_Q \sharp* \Psi$› ‹$A_Q \sharp* P$›

```
‹A_Q ♯* Q› ‹xvec ♯* P› ‹A_P ♯* K› ‹A_Q ♯* M›
      by(intro Comm1) (assumption | simp add: freshChainSym)+
    moreover have (Ψ, P' ∥ Q', Q' ∥ P') ∈ Rel by(rule C1)
     then have (Ψ, (⦇ν*xvec⦈)(P' ∥ Q'), (⦇ν*xvec⦈)(Q' ∥ P')) ∈ Rel using ‹xvec ♯*
Ψ› by(rule C2)
    ultimately show ?case by blast
  next
    case(cBrMerge Ψ_P M N Q' A_Q Ψ_Q P' A_P)
    then have Ψ ▷ P ∥ Q ⟼ ¿M⦇N⦈ ≺ P' ∥ Q'
      by(intro BrMerge) (assumption|simp)+
    then show ?case by(blast intro: C1)
  next
    case(cBrComm1 Ψ_P M N Q' A_Q Ψ_Q xvec P' A_P)
    then have Ψ ▷ P ∥ Q ⟼ ¡M(⦇ν*xvec⦈)⟨N⟩ ≺ P' ∥ Q'
      by(intro BrComm2) (assumption|simp)+
    then show ?case by(blast intro: C1)
  next
    case(cBrComm2 Ψ_P M xvec N Q' A_Q Ψ_Q P' A_P)
    then have Ψ ▷ P ∥ Q ⟼ ¡M(⦇ν*xvec⦈)⟨N⟩ ≺ P' ∥ Q'
      by(intro BrComm1) (assumption|simp)+
    then show ?case by(blast intro: C1)
  qed
qed

lemma bangExtLeft:
  fixes Ψ :: 'b
    and P :: ('a, 'b, 'c) psi

assumes guarded P
  and   ⋀Ψ' Q. (Ψ', Q, Q) ∈ Rel

shows Ψ ▷ !P ⇝[Rel] P ∥ !P
  using assms
  by(auto simp add: simulation-def dest: Bang)

lemma bangExtRight:
  fixes Ψ :: 'b
    and P :: ('a, 'b, 'c) psi

assumes C1: ⋀Ψ' Q. (Ψ', Q, Q) ∈ Rel

shows Ψ ▷ P ∥ !P ⇝[Rel] !P
proof −
  {
    fix α P'
    assume Ψ ▷ !P ⟼α ≺ P'
    then have Ψ ▷ P ∥ !P ⟼α ≺ P'
      apply −
      by(ind-cases Ψ ▷ !P ⟼α ≺ P') (auto simp add: psi.inject)
```

moreover have $(\Psi,\ P',\ P') \in Rel$ **by**(*rule C1*)
ultimately have $\exists P''.\ \Psi \rhd P \parallel\ !P \longmapsto \alpha \prec P'' \wedge (\Psi,\ P'',\ P') \in Rel$
by *blast*
}
then show *?thesis*
by(*auto simp add*: *simulation-def*)
qed

end

end
theory *Bisim-Pres*
imports *Bisimulation Sim-Pres*
begin

This file is a (heavily modified) variant of the theory *Psi_Calculi.Bisim_Pres* from [1].

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 −
let $?X = \{(\Psi,\ M(\!|\lambda * xvec\ N|\!).P,\ M(\!|\lambda * xvec\ N|\!).Q) \mid \Psi\ M\ xvec\ N\ P\ Q.\ \forall Tvec.\ length\ xvec = length\ Tvec \longrightarrow \Psi \rhd P[xvec::=Tvec] \sim Q[xvec::=Tvec]\}$

from *assms* **have** $(\Psi,\ M(\!|\lambda * xvec\ N|\!).P,\ M(\!|\lambda * xvec\ N|\!).Q) \in ?X$ **by** *blast*
then show *?thesis*
proof(*coinduct rule*: *bisimCoinduct*)
case(*cStatEq* $\Psi\ P\ Q$)
then show *?case* **by** *auto*
next
case(*cSim* $\Psi\ P\ Q$)
then show *?case* **by**(*blast intro*: *inputPres*)
next
case(*cExt* $\Psi\ P\ Q\ \Psi'$)
then show *?case* **by**(*blast dest*: *bisimE*)
next
case(*cSym* $\Psi\ P\ Q$)
then show *?case* **by**(*blast dest*: *bisimE*)
qed

qed

lemma *bisimOutputPres*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $M :: 'a$
and $N :: 'a$

assumes $\Psi \rhd P \sim Q$

shows $\Psi \rhd M\langle N\rangle.P \sim M\langle N\rangle.Q$
proof −
let $?X = \{(\Psi, M\langle N\rangle.P, M\langle N\rangle.Q) \mid \Psi\ M\ N\ P\ Q.\ \Psi \rhd P \sim Q\}$
from ‹$\Psi \rhd P \sim Q$› **have** $(\Psi, M\langle N\rangle.P, M\langle N\rangle.Q) \in ?X$ **by** *auto*
then show *?thesis*
by(*coinduct rule*: *bisimCoinduct*, *auto*) (*blast intro*: *outputPres dest*: *bisimE*)+
qed

lemma *bisimCasePres*:
fixes $\Psi :: 'b$
and $CsP :: ('c \times ('a, 'b, 'c)\ psi)\ list$
and $CsQ :: ('c \times ('a, 'b, 'c)\ psi)\ list$

assumes $\bigwedge\varphi\ P.\ (\varphi, P) \in set\ CsP \Longrightarrow \exists\, Q.\ (\varphi, Q) \in set\ CsQ \wedge guarded\ Q \wedge \Psi \rhd P \sim Q$
and $\bigwedge\varphi\ Q.\ (\varphi, Q) \in set\ CsQ \Longrightarrow \exists\, P.\ (\varphi, P) \in set\ CsP \wedge guarded\ P \wedge \Psi \rhd P \sim Q$

shows $\Psi \rhd Cases\ CsP \sim Cases\ CsQ$
proof −
let $?X = \{(\Psi, Cases\ CsP, Cases\ CsQ) \mid \Psi\ CsP\ CsQ.\ (\forall\varphi\ P.\ (\varphi, P) \in set\ CsP \longrightarrow (\exists\, Q.\ (\varphi, Q) \in set\ CsQ \wedge guarded\ Q \wedge \Psi \rhd P \sim Q)) \wedge (\forall\varphi\ Q.\ (\varphi, Q) \in set\ CsQ \longrightarrow (\exists\, P.\ (\varphi, P) \in set\ CsP \wedge guarded\ P \wedge \Psi \rhd P \sim Q))\}$
from *assms* **have** $(\Psi, Cases\ CsP, Cases\ CsQ) \in ?X$ **by** *auto*
then show *?thesis*
proof(*coinduct rule*: *bisimCoinduct*)
case(*cStatEq* $\Psi\ P\ Q$)
then show *?case* **by** *auto*
next
case(*cSim* $\Psi\ CasesP\ CasesQ$)
then obtain $CsP\ CsQ$ **where** *C1*: $\bigwedge\varphi\ Q.\ (\varphi, Q) \in set\ CsQ \Longrightarrow \exists\, P.\ (\varphi, P) \in set\ CsP \wedge guarded\ P \wedge \Psi \rhd P \sim Q$
and *A*: $CasesP = Cases\ CsP$ **and** *B*: $CasesQ = Cases\ CsQ$
by *auto*
note *C1*
moreover have $\bigwedge\Psi\ P\ Q.\ \Psi \rhd P \sim Q \Longrightarrow \Psi \rhd P \rightsquigarrow[bisim]\ Q$ **by**(*rule bisimE*)
moreover have $bisim \subseteq ?X \cup bisim$ **by** *blast*

```
    ultimately have Ψ ▷ Cases CsP ⇝[(?X ∪ bisim)] Cases CsQ
      by(rule casePres)
    then show ?case using A B by blast
  next
    case(cExt Ψ P Q)
    then show ?case by(blast dest: bisimE)
  next
    case(cSym Ψ P Q)
    then show ?case by(blast dest: bisimE)
  qed
qed

lemma bisimResPres:
  fixes Ψ :: 'b
    and P :: ('a, 'b, 'c) psi
    and Q :: ('a, 'b, 'c) psi
    and x :: name

assumes Ψ ▷ P ∼ Q
  and   x ♯ Ψ

shows Ψ ▷ (|νx|)P ∼ (|νx|)Q
proof −
  let ?X = {(Ψ, (|ν*xvec|)P, (|ν*xvec|)Q) | Ψ xvec P Q. Ψ ▷ P ∼ Q ∧ xvec ♯* Ψ}
 have eqvt ?X using bisimClosed pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]
    by(auto simp add: eqvts eqvt-def) blast
  have (Ψ, (|νx|)P, (|νx|)Q) ∈ ?X
  proof −
    from ‹x ♯ Ψ› have [x] ♯* Ψ
      by auto
    with ‹Ψ ▷ P ∼ Q› show ?thesis
      by (smt mem-Collect-eq resChain.base resChain.step)
  qed
  then show ?thesis
  proof(coinduct rule: bisimCoinduct)
    case(cStatEq Ψ xP xQ)
    from ‹(Ψ, xP, xQ) ∈ ?X› obtain xvec P Q where Ψ ▷ P ∼ Q and xvec ♯*
Ψ and xP = (|ν*xvec|)P and xQ = (|ν*xvec|)Q
      by auto
    moreover from ‹Ψ ▷ P ∼ Q› have PeqQ: insertAssertion(extractFrame P)
Ψ ≃F insertAssertion(extractFrame Q) Ψ
      by(rule bisimE)
    ultimately show ?case by(auto intro: frameResChainPres)
  next
    case(cSim Ψ xP xQ)
    from ‹(Ψ, xP, xQ) ∈ ?X› obtain xvec P Q where Ψ ▷ P ∼ Q and xvec ♯*
Ψ and xP = (|ν*xvec|)P and xQ = (|ν*xvec|)Q
      by auto
    from ‹Ψ ▷ P ∼ Q› have Ψ ▷ P ⇝[bisim] Q by(rule bisimE)
```

note ‹*eqvt ?X*›
then have *eqvt*(*?X* ∪ *bisim*) **by** *auto*
from ‹*xvec* ♯* Ψ›
have Ψ ▷ (|ν*xvec|)*P* ⇝[(*?X* ∪ *bisim*)] (|ν*xvec|)*Q*
proof(*induct xvec*)
case *Nil*
have Ψ ▷ (|ν*[]|)*P* ⇝[*bisim*] (|ν*[]|)*Q* **using** ‹Ψ ▷ *P* ∼ *Q*›
unfolding *resChain.simps*
by(*rule bisimE*)
then show *?case* **by**(*rule monotonic*) *auto*
next
case(*Cons x xvec*)
then have *x* ♯ Ψ **and** *xvec* ♯* Ψ
by *auto*
from ‹*xvec* ♯* Ψ› **have** Ψ ▷ (|ν*xvec|)*P* ⇝[(*?X* ∪ *bisim*)] (|ν*xvec|)*Q*
by(*rule Cons*)
moreover note ‹*eqvt*(*?X* ∪ *bisim*)›
moreover note ‹*x* ♯ Ψ›
moreover have *?X* ∪ *bisim* ⊆ *?X* ∪ *bisim* **by** *auto*
moreover have ⋀Ψ *P Q xvec*. ⟦(Ψ, *P*, *Q*) ∈ *?X* ∪ *bisim*; *xvec* ♯* Ψ⟧ ⟹ (Ψ, (|ν*xvec|)*P*, (|ν*xvec|)*Q*) ∈ *?X* ∪ *bisim*
by (*smt* (*verit*, *ccfv-threshold*) *Un-iff freshChainAppend mem-Collect-eq prod.inject resChainAppend*)
ultimately have Ψ ▷ (|ν*x|)((|ν*xvec|)*P*) ⇝[(*?X* ∪ *bisim*)] (|ν*x|)((|ν*xvec|)*Q*)
by(*rule resPres*)
then show *?case* **unfolding** *resChain.simps* **by** −
next
qed
with ‹*xP* = (|ν*xvec|)*P*› ‹*xQ* = (|ν*xvec|)*Q*› **show** *?case*
by *simp*
next
case(*cExt* Ψ *xP xQ* Ψ′)
from ‹(Ψ, *xP*, *xQ*) ∈ *?X*› **obtain** *xvec P Q* **where** Ψ ▷ *P* ∼ *Q* **and** *xvec* ♯* Ψ **and** *xpe*: *xP* = (|ν*xvec|)*P* **and** *xqe*: *xQ* = (|ν*xvec|)*Q*
by *auto*
obtain *p*::*name prm* **where** (*p*•*xvec*) ♯* Ψ **and** (*p*•*xvec*) ♯* Ψ′ **and** (*p*•*xvec*) ♯* *P* **and** (*p*•*xvec*) ♯* *Q* **and** (*p*•*xvec*) ♯* *xvec* **and** *distinctPerm p* **and** *set p* ⊆ (*set xvec*) × (*set* (*p*•*xvec*))
by(*rule name-list-avoiding*[**where** *c*=(Ψ,Ψ′,*P*,*Q*,*xvec*)]) *auto*
from ‹Ψ ▷ *P* ∼ *Q*› **have** Ψ ⊗ (*p*•Ψ′) ▷ *P* ∼ *Q*
by(*rule bisimE*)
moreover have *xvec* ♯* (Ψ ⊗ (*p*•Ψ′)) **using** ‹*xvec* ♯* Ψ› ‹(*p*•*xvec*) ♯* Ψ′› ‹*distinctPerm p*›
apply(*intro freshCompChain*)
apply *assumption*
apply(*subst perm-bool*[**where** *pi*=*p*,*symmetric*])
by(*simp add*: *eqvts*)
ultimately have (Ψ ⊗ (*p*•Ψ′),(|ν*xvec|)*P*,(|ν*xvec|)*Q*) ∈ *?X*
by *auto*

```
    then have (p•(Ψ ⊗ (p•Ψ'),(⌈ν*xvec⌋)P,(⌈ν*xvec⌋)Q)) ∈ ?X using ‹eqvt ?X›
      by(intro eqvtI)
     then have (Ψ ⊗ Ψ',(⌈ν*(p•xvec)⌋)(p•P),(⌈ν*(p•xvec)⌋)(p•Q)) ∈ ?X using ‹distinctPerm p› ‹set p ⊆ (set xvec) × (set (p•xvec))› ‹xvec ♯* Ψ› ‹(p•xvec) ♯* Ψ›
      by(simp add: eqvts)
    then have (Ψ ⊗ Ψ',(⌈ν*xvec⌋)P,(⌈ν*xvec⌋)Q) ∈ ?X using ‹(p • xvec) ♯* Q› ‹(p • xvec) ♯* P› ‹set p ⊆ set xvec × set (p • xvec)› ‹distinctPerm p›
      by(subst (1 2) resChainAlpha[where p=p]) auto
    then show ?case unfolding xpe xqe
      by blast
  next
    case(cSym Ψ P Q)
    then show ?case
      by(blast dest: bisimE)
  qed
qed

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

assumes Ψ ▷ P ∼ Q
  and   xvec ♯* Ψ

shows Ψ ▷ (⌈ν*xvec⌋)P ∼ (⌈ν*xvec⌋)Q
  using assms
  by(induct xvec) (auto intro: bisimResPres)

lemma bisimParPresAux:
  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 Ψ ⊗ Ψ_R ▷ P ∼ Q
  and   FrR: extractFrame R = ⟨A_R, Ψ_R⟩
  and   A_R ♯* Ψ
  and   A_R ♯* P
  and   A_R ♯* Q

shows Ψ ▷ P ∥ R ∼ Q ∥ R
proof −
  let ?X = {(Ψ, (⌈ν*xvec⌋)(P ∥ R), (⌈ν*xvec⌋)(Q ∥ R)) | xvec Ψ P Q R. xvec ♯* Ψ ∧
(∀ A_R Ψ_R. (extractFrame R = ⟨A_R, Ψ_R⟩ ∧ A_R ♯* Ψ ∧ A_R ♯* P ∧ A_R ♯* Q) ⟶
                                                                  Ψ ⊗ Ψ_R ▷
```

$P \sim Q)\}$
{
fix *xvec* :: *name list*
and Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and Q :: $('a, 'b, 'c)$ *psi*
and R :: $('a, 'b, 'c)$ *psi*

assume *xvec* $\sharp*$ Ψ
and $\bigwedge A_R\ \Psi_R.\ [\![extractFrame\ R = \langle A_R, \Psi_R\rangle;\ A_R \sharp* \Psi;\ A_R \sharp* P;\ A_R \sharp* Q]\!] \Longrightarrow \Psi \otimes \Psi_R \rhd P \sim Q$

then have $(\Psi, (\!|\nu*xvec|\!)(P \parallel R), (\!|\nu*xvec|\!)(Q \parallel R)) \in ?X$
by *auto blast*
}

note *XI* = *this*
{
fix *xvec* :: *name list*
and Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and Q :: $('a, 'b, 'c)$ *psi*
and R :: $('a, 'b, 'c)$ *psi*
and C :: $'d$::*fs-name*

assume *xvec* $\sharp*$ Ψ
and *A*: $\bigwedge A_R\ \Psi_R.\ [\![extractFrame\ R = \langle A_R, \Psi_R\rangle;\ A_R \sharp* \Psi;\ A_R \sharp* P;\ A_R \sharp* Q;\ A_R \sharp* C]\!] \Longrightarrow \Psi \otimes \Psi_R \rhd P \sim Q$

from ‹*xvec* $\sharp*$ Ψ› **have** $(\Psi, (\!|\nu*xvec|\!)(P \parallel R), (\!|\nu*xvec|\!)(Q \parallel R)) \in ?X$
proof(*rule XI*)
fix $A_R\ \Psi_R$
assume *FrR*: *extractFrame* $R = \langle A_R, \Psi_R\rangle$
obtain *p*::*name prm* **where** $(p \cdot A_R) \sharp* \Psi$ **and** $(p \cdot A_R) \sharp* P$ **and** $(p \cdot A_R) \sharp* Q$ **and** $(p \cdot A_R) \sharp* R$ **and** $(p \cdot A_R) \sharp* C$
and $(p \cdot A_R) \sharp* \Psi_R$ **and** *S*: $(set\ p) \subseteq (set\ A_R) \times (set(p \cdot A_R))$ **and** *distinctPerm p*
by(*rule name-list-avoiding*[**where** *c*=$(\Psi, P, Q, R, \Psi_R, C)$]) *auto*
from *FrR* ‹$(p \cdot A_R) \sharp* \Psi_R$› *S* **have** *extractFrame* $R = \langle (p \cdot A_R), p \cdot \Psi_R\rangle$ **by**(*simp add*: *frameChainAlpha'*)

moreover assume $A_R \sharp* \Psi$
then have $(p \cdot A_R) \sharp* (p \cdot \Psi)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$A_R \sharp* \Psi$› ‹$(p \cdot A_R) \sharp* \Psi$› *S* **have** $(p \cdot A_R) \sharp* \Psi$ **by** *simp*
moreover assume $A_R \sharp* P$
then have $(p \cdot A_R) \sharp* (p \cdot P)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$A_R \sharp* P$› ‹$(p \cdot A_R) \sharp* P$› *S* **have** $(p \cdot A_R) \sharp* P$ **by** *simp*

moreover assume $A_R \sharp* Q$
then have $(p \cdot A_R) \sharp* (p \cdot Q)$ **by**(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$A_R \sharp* Q$› ‹$(p \cdot A_R) \sharp* Q$› S **have** $(p \cdot A_R) \sharp* Q$ **by** *simp*
ultimately have $\Psi \otimes (p \cdot \Psi_R) \rhd P \sim Q$ **using** ‹$(p \cdot A_R) \sharp* C$› A **by** *blast*
then have $(p \cdot (\Psi \otimes (p \cdot \Psi_R))) \rhd (p \cdot P) \sim (p \cdot Q)$ **by**(*rule bisimClosed*)
with ‹$A_R \sharp* \Psi$› ‹$(p \cdot A_R) \sharp* \Psi$› ‹$A_R \sharp* P$› ‹$(p \cdot A_R) \sharp* P$› ‹$A_R \sharp* Q$› ‹$(p \cdot A_R) \sharp* Q$› S ‹*distinctPerm p*›
show $\Psi \otimes \Psi_R \rhd P \sim Q$ **by**(*simp add*: *eqvts*)
qed
}
note $XI' = this$

have *eqvt ?X*
unfolding *eqvt-def*
proof
fix x
assume $x \in ?X$
then obtain *xvec* Ψ P Q R **where** *1*: $x = (\Psi, (\!|\nu*xvec|\!)P \parallel R, (\!|\nu*xvec|\!)Q \parallel R)$ **and** *2*: *xvec* $\sharp* \Psi$ **and**
3: $\forall A_R\ \Psi_R.\ extractFrame\ R = \langle A_R, \Psi_R\rangle \wedge A_R \sharp* \Psi \wedge A_R \sharp* P \wedge A_R \sharp* Q \longrightarrow \Psi \otimes \Psi_R \rhd P \sim Q$
by *blast*
show $\forall p::(name \times name)\ list.\ p \cdot x \in ?X$
proof
fix $p :: (name \times name)\ list$
from *1* **have** $p \cdot x = (p \cdot \Psi, (\!|\nu*p \cdot xvec|\!)(p \cdot P) \parallel (p \cdot R), (\!|\nu*p \cdot xvec|\!)(p \cdot Q) \parallel (p \cdot R))$
by (*simp add*: *eqvts*)
moreover from *2* **have** $(p \cdot xvec) \sharp* (p \cdot \Psi)$
by (*simp add*: *fresh-star-bij*(*2*))
moreover have $\forall A_R\ \Psi_R.\ extractFrame\ (p \cdot R) = \langle A_R, \Psi_R\rangle \wedge A_R \sharp* (p \cdot \Psi) \wedge A_R \sharp* (p \cdot P) \wedge A_R \sharp* (p \cdot Q) \longrightarrow (p \cdot \Psi) \otimes \Psi_R \rhd (p \cdot P) \sim (p \cdot Q)$
proof (*rule allI*, *rule allI*, *rule impI*, (*erule conjE*)+)
fix $A_R\ \Psi_R$
assume *exF*: *extractFrame* $(p \cdot R) = \langle A_R, \Psi_R\rangle$ **and** *freshPsi*: $A_R \sharp* (p \cdot \Psi)$ **and** *freshP*: $A_R \sharp* (p \cdot P)$ **and** *freshQ*: $A_R \sharp* (p \cdot Q)$
from *exF* **have** $extractFrame\ R = \langle rev\ p \cdot A_R, rev\ p \cdot \Psi_R\rangle$
by (*metis Chain.simps*(*5*) *extractFrameEqvt*(*1*) *frame.perm*(*1*) *frameResChainEqvt*)
moreover from *freshPsi* **have** $(rev\ p \cdot A_R) \sharp* \Psi$
by (*metis fresh-star-bij*(*2*) *perm-pi-simp*(*1*))
moreover from *freshP* **have** $(rev\ p \cdot A_R) \sharp* P$
by (*metis fresh-star-bij*(*2*) *perm-pi-simp*(*1*))
moreover from *freshQ* **have** $(rev\ p \cdot A_R) \sharp* Q$
by (*metis fresh-star-bij*(*2*) *perm-pi-simp*(*1*))
ultimately show $(p \cdot \Psi) \otimes \Psi_R \rhd p \cdot P \sim p \cdot Q$
using *3* **by** (*metis* (*no-types*, *opaque-lifting*) *bisimClosed perm-pi-simp*(*2*) *statEqvt'*)
qed

ultimately show $p \cdot x \in ?X$
by *blast*
qed
qed

moreover have *Res*: $\bigwedge \Psi\ P\ Q\ x.\ [\![(\Psi,\ P,\ Q) \in ?X \cup bisim;\ x \sharp \Psi]\!] \Longrightarrow (\Psi,\ (\!|\nu x|\!)P,\ (\!|\nu x|\!)Q) \in ?X \cup bisim$
proof −
fix $\Psi\ P\ Q\ x$
assume $(\Psi,\ P,\ Q) \in ?X \cup bisim$ **and** $(x::name) \sharp \Psi$
show $(\Psi,\ (\!|\nu x|\!)P,\ (\!|\nu x|\!)Q) \in ?X \cup bisim$
proof(*cases* $(\Psi,\ P,\ Q) \in ?X$)
assume $(\Psi,\ P,\ Q) \in ?X$
then obtain *xvec* $P'\ Q'\ R$ **where** $P = (\!|\nu{*}xvec|\!)P' \parallel R$ **and** $Q = (\!|\nu{*}xvec|\!)Q' \parallel R$ **and** $xvec \sharp* \Psi$
and $\forall A_R\ \Psi_R.\ extractFrame\ R = \langle A_R,\ \Psi_R\rangle \wedge A_R \sharp* \Psi \wedge A_R \sharp* P' \wedge A_R \sharp* Q' \longrightarrow \Psi \otimes \Psi_R \rhd P' \sim Q'$
by *blast*
moreover have $(\!|\nu x|\!)((\!|\nu{*}xvec|\!)P' \parallel R) = (\!|\nu{*}(x\ \#\ xvec)|\!)P' \parallel R$
by *simp*
moreover have $(\!|\nu x|\!)((\!|\nu{*}xvec|\!)Q' \parallel R) = (\!|\nu{*}(x\ \#\ xvec)|\!)Q' \parallel R$
by *simp*
moreover from ‹$x \sharp \Psi$› ‹$xvec \sharp* \Psi$› **have** $(x\ \#\ xvec) \sharp* \Psi$
by *simp*
ultimately have $(\Psi,\ (\!|\nu x|\!)P,\ (\!|\nu x|\!)Q) \in ?X$
by *blast*
then show *?thesis* **by** *simp*
next
assume $\neg(\Psi,\ P,\ Q) \in ?X$
with ‹$(\Psi,\ P,\ Q) \in ?X \cup bisim$› **have** $\Psi \rhd P \sim Q$
by *blast*
then have $\Psi \rhd (\!|\nu x|\!)P \sim (\!|\nu x|\!)Q$ **using** ‹$x \sharp \Psi$›
by(*rule bisimResPres*)
then show *?thesis*
by *simp*
qed
qed
have *ResChain*: $\bigwedge \Psi\ P\ Q\ xvec.\ [\![(\Psi,\ P,\ Q) \in ?X \cup bisim;\ xvec \sharp* \Psi]\!] \Longrightarrow (\Psi,\ (\!|\nu{*}xvec|\!)P,\ (\!|\nu{*}xvec|\!)Q) \in ?X \cup bisim$
proof −
fix $\Psi\ P\ Q$
and *xvec*::*name list*
assume $(\Psi,\ P,\ Q) \in ?X \cup bisim$
and $xvec \sharp* \Psi$
then show $(\Psi,\ (\!|\nu{*}xvec|\!)P,\ (\!|\nu{*}xvec|\!)Q) \in ?X \cup bisim$
proof(*induct xvec*)
case *Nil* **then show** *?case* **by** *simp*
next
case(*Cons x xvec*)

```
    then have (Ψ, (⦇ν*xvec⦈)P, (⦇ν*xvec⦈)Q) ∈ ?X ∪ bisim
      by simp
    moreover have x ♯ Ψ using Cons by simp
    ultimately show ?case unfolding resChain.simps
      by(rule Res)
   qed
  qed
  have (Ψ, P ∥ R, Q ∥ R) ∈ ?X
  proof −
   {
    fix A_R' :: name list
      and Ψ_R' :: 'b

    assume FrR': extractFrame R = ⟨A_R', Ψ_R'⟩
      and  A_R' ♯* Ψ
      and  A_R' ♯* P
      and  A_R' ♯* Q

    obtain p where (p • A_R') ♯* A_R and (p • A_R') ♯* Ψ_R' and (p • A_R') ♯* Ψ
and (p • A_R') ♯* P and (p • A_R') ♯* Q
      and Sp: (set p) ⊆ (set A_R') × (set(p • A_R')) and distinctPerm p
      by(rule name-list-avoiding[where c=(A_R, Ψ, Ψ_R', P, Q)]) auto

     from FrR' ‹(p • A_R') ♯*  Ψ_R'› Sp have extractFrame R = ⟨(p • A_R'), p •
Ψ_R'⟩
      by(simp add: frameChainAlpha eqvts)
    with FrR ‹(p • A_R') ♯* A_R› obtain q::name prm
      where Sq: set q ⊆ set(p • A_R') × set A_R and distinctPerm q and Ψ_R = q
• p • Ψ_R'
      by(force elim: frameChainEq)

    from ‹Ψ ⊗ Ψ_R ▷ P ∼ Q› ‹Ψ_R = q • p • Ψ_R'› have Ψ ⊗ (q • p • Ψ_R') ▷ P
∼ Q by simp
   then have (q • (Ψ ⊗ (q • p • Ψ_R'))) ▷ (q • P) ∼ (q • Q) by(rule bisimClosed)
    with Sq ‹A_R ♯* Ψ› ‹(p • A_R') ♯* Ψ› ‹A_R ♯* P› ‹(p • A_R') ♯* P› ‹A_R ♯* Q›
‹(p • A_R') ♯* Q› ‹distinctPerm q›
    have Ψ ⊗ (p • Ψ_R') ▷ P ∼ Q by(simp add: eqvts)
    then have (p • (Ψ ⊗ (p • Ψ_R'))) ▷ (p • P) ∼ (p • Q) by(rule bisimClosed)
   with Sp ‹A_R' ♯* Ψ› ‹(p • A_R') ♯* Ψ› ‹A_R' ♯* P› ‹(p • A_R') ♯* P› ‹A_R' ♯* Q›
‹(p • A_R') ♯* Q› ‹distinctPerm p›
    have Ψ ⊗ Ψ_R' ▷ P ∼ Q by(simp add: eqvts)
   }
   then show ?thesis
    apply clarsimp
    apply(rule exI[where x=[]])
    by auto blast
  qed
  then show ?thesis
  proof(coinduct rule: bisimCoinduct)
```

case(*cStatEq* Ψ *PR QR*)
from ‹(Ψ, *PR*, *QR*) ∈ *?X*›
obtain *xvec P Q R* **where** *PFrR*: $PR = (\!|\nu{*}xvec|\!)(P \parallel R)$ **and** *QFrR*: $QR = (\!|\nu{*}xvec|\!)(Q \parallel R)$
and *xvec* $\sharp*$ Ψ **and** $*$: $\forall\, A_R\ \Psi_R.$ *extractFrame* $R = \langle A_R, \Psi_R\rangle \wedge A_R \sharp* \Psi \wedge A_R \sharp* P \wedge A_R \sharp* Q \longrightarrow \Psi \otimes \Psi_R \rhd P \sim Q$
by *blast*
obtain $A_R\ \Psi_R$ **where** *FrR*: *extractFrame* $R = \langle A_R, \Psi_R\rangle$ **and** *fresh*: $A_R \sharp* (\Psi, P, Q, R)$
using *freshFrame* **by** *metis*
from *fresh* **have** $A_R \sharp* \Psi$ **and** $A_R \sharp* P$ **and** $A_R \sharp* Q$ **and** $A_R \sharp* R$
by *auto*
with *FrR* **have** *PSimQ*: $\Psi \otimes \Psi_R \rhd P \sim Q$
using $*$ **by** *blast*

obtain $A_P\ \Psi_P$ **where** *FrP*: *extractFrame* $P = \langle A_P, \Psi_P\rangle$ **and** $A_P \sharp* \Psi$ **and** $A_P \sharp* A_R$ **and** $A_P \sharp* \Psi_R$
by(*rule freshFrame*[**where** C=(Ψ, A_R, Ψ_R)]) *auto*
obtain $A_Q\ \Psi_Q$ **where** *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$ **and** $A_Q \sharp* \Psi$ **and** $A_Q \sharp* A_R$ **and** $A_Q \sharp* \Psi_R$
by(*rule freshFrame*[**where** C=(Ψ, A_R, Ψ_R)]) *auto*
from ‹$A_R \sharp* P$› ‹$A_R \sharp* Q$› ‹$A_P \sharp* A_R$› ‹$A_Q \sharp* A_R$› *FrP FrQ* **have** $A_R \sharp* \Psi_P$ **and** $A_R \sharp* \Psi_Q$
by(*force dest*: *extractFrameFreshChain*)+

have $\langle (A_P @ A_R), \Psi \otimes \Psi_P \otimes \Psi_R\rangle \simeq_F \langle (A_Q @ A_R), \Psi \otimes \Psi_Q \otimes \Psi_R\rangle$
proof −
have $\langle A_P, \Psi \otimes \Psi_P \otimes \Psi_R\rangle \simeq_F \langle A_P, (\Psi \otimes \Psi_R) \otimes \Psi_P\rangle$
by(*metis frameResChainPres frameNilStatEq Associativity Commutativity AssertionStatEqTrans Composition*)
moreover from *PSimQ* **have** *insertAssertion*(*extractFrame P*) ($\Psi \otimes \Psi_R$) $\simeq_F$ *insertAssertion*(*extractFrame Q*) ($\Psi \otimes \Psi_R$)
by(*rule bisimE*)
with *FrP FrQ freshCompChain* ‹$A_P \sharp* \Psi$› ‹$A_P \sharp* \Psi_R$› ‹$A_Q \sharp* \Psi$› ‹$A_Q \sharp* \Psi_R$› **have** $\langle A_P, (\Psi \otimes \Psi_R) \otimes \Psi_P\rangle \simeq_F \langle A_Q, (\Psi \otimes \Psi_R) \otimes \Psi_Q\rangle$
by *auto*
moreover have $\langle A_Q, (\Psi \otimes \Psi_R) \otimes \Psi_Q\rangle \simeq_F \langle A_Q, \Psi \otimes \Psi_Q \otimes \Psi_R\rangle$
by(*metis frameResChainPres frameNilStatEq Associativity Commutativity AssertionStatEqTrans Composition*)
ultimately have $\langle A_P, \Psi \otimes \Psi_P \otimes \Psi_R\rangle \simeq_F \langle A_Q, \Psi \otimes \Psi_Q \otimes \Psi_R\rangle$
by(*blast intro*: *FrameStatEqTrans*)
then have $\langle (A_R @ A_P), \Psi \otimes \Psi_P \otimes \Psi_R\rangle \simeq_F \langle (A_R @ A_Q), \Psi \otimes \Psi_Q \otimes \Psi_R\rangle$
by(*drule-tac frameResChainPres*) (*simp add*: *frameChainAppend*)
then show *?thesis*
apply(*simp add*: *frameChainAppend*)
by(*metis frameResChainComm FrameStatEqTrans*)
qed
moreover from ‹$A_P \sharp* \Psi$› ‹$A_R \sharp* \Psi$› **have** $(A_P @ A_R) \sharp* \Psi$ **by** *simp*
moreover from ‹$A_Q \sharp* \Psi$› ‹$A_R \sharp* \Psi$› **have** $(A_Q @ A_R) \sharp* \Psi$ **by** *simp*

ultimately have *PFrRQR*: *insertAssertion*(*extractFrame*($P \parallel R$)) $\Psi \simeq_F$ *insertAssertion*(*extractFrame*($Q \parallel R$)) Ψ
using *FrP FrQ FrR* ‹$A_P \sharp* A_R$› ‹$A_P \sharp* \Psi_R$› ‹$A_Q \sharp* A_R$› ‹$A_Q \sharp* \Psi_R$› ‹$A_R \sharp* \Psi_P$› ‹$A_R \sharp* \Psi_Q$›
by *simp*

from ‹*xvec* $\sharp* \Psi$› **have** *insertAssertion* (*extractFrame*($(\!|\nu* xvec|\!)P \parallel R$)) $\Psi \simeq_F$ $(\!|\nu* xvec|\!)$(*insertAssertion* (*extractFrame*($P \parallel R$)) Ψ)
by(*rule insertAssertionExtractFrameFresh*)
moreover from *PFrRQR* **have** $(\!|\nu* xvec|\!)$(*insertAssertion* (*extractFrame*($P \parallel R$)) Ψ) $\simeq_F$ $(\!|\nu* xvec|\!)$(*insertAssertion* (*extractFrame*($Q \parallel R$)) Ψ)
by(*induct xvec*) (*auto intro*: *frameResPres*)
moreover from ‹*xvec* $\sharp* \Psi$› **have** $(\!|\nu* xvec|\!)$(*insertAssertion* (*extractFrame*($Q \parallel R$)) Ψ) $\simeq_F$ *insertAssertion* (*extractFrame*($(\!|\nu* xvec|\!)Q \parallel R$)) Ψ
by(*rule FrameStatEqSym*[*OF insertAssertionExtractFrameFresh*])
ultimately show *?case* **using** *PFrR QFrR*
by(*blast intro*: *FrameStatEqTrans*)
next
case(*cSim* Ψ *PR QR*)
{
fix Ψ *P Q R xvec*
assume $\bigwedge A_R\ \Psi_R$. ⟦*extractFrame* $R = \langle A_R, \Psi_R\rangle$; $A_R \sharp* \Psi$; $A_R \sharp* P$; $A_R \sharp* Q$⟧ $\Longrightarrow \Psi \otimes \Psi_R \rhd P \sim Q$
moreover have *eqvt bisim* **by** *simp*
moreover from ‹*eqvt ?X*› **have** *eqvt*(*?X* ∪ *bisim*) **by** *auto*
moreover from *bisimE*(*1*) **have** $\bigwedge \Psi\ P\ Q.\ \Psi \rhd P \sim Q \Longrightarrow$ *insertAssertion* (*extractFrame Q*) $\Psi \hookrightarrow_F$ *insertAssertion* (*extractFrame P*) Ψ **by**(*simp add*: *FrameStatEq-def*)
moreover note *bisimE*(*2*) *bisimE*(*3*)
moreover
{
fix Ψ *P Q* A_R Ψ_R *R*
assume *PSimQ*: $\Psi \otimes \Psi_R \rhd P \sim Q$
and *FrR*: *extractFrame* $R = \langle A_R, \Psi_R\rangle$
and $A_R \sharp* \Psi$
and $A_R \sharp* P$
and $A_R \sharp* Q$
then have $(\Psi, P \parallel R, Q \parallel R) \in$ *?X*
proof −
have $P \parallel R = (\!|\nu*[]|\!)(P \parallel R)$ **by** *simp*
moreover have $Q \parallel R = (\!|\nu*[]|\!)(Q \parallel R)$ **by** *simp*
moreover have ([]::*name list*) $\sharp* \Psi$ **by** *simp*
moreover
{
fix $A_R'\ \Psi_R'$

assume *FrR'*: *extractFrame* $R = \langle A_R', \Psi_R'\rangle$
and $A_R' \sharp* \Psi$
and $A_R' \sharp* P$

and $A_R' \sharp* Q$
obtain p **where** $(p \cdot A_R') \sharp* A_R$
and $(p \cdot A_R') \sharp* \Psi_R'$
and $(p \cdot A_R') \sharp* \Psi$
and $(p \cdot A_R') \sharp* P$
and $(p \cdot A_R') \sharp* Q$
and S: $(set\ p) \subseteq (set\ A_R') \times (set(p \cdot A_R'))$ **and** $distinctPerm\ p$
by(*rule name-list-avoiding*[**where** c=$(A_R, \Psi, \Psi_R', P, Q)$]) *auto*

from $\langle(p \cdot A_R') \sharp* \Psi_R'\rangle\ S$ **have** $\langle A_R', \Psi_R'\rangle = \langle p \cdot A_R', p \cdot \Psi_R'\rangle$
by(*simp add*: *frameChainAlpha*)

with *FrR'* **have** *FrR''*: *extractFrame* $R = \langle p \cdot A_R', p \cdot \Psi_R'\rangle$ **by** *simp*
with *FrR* $\langle(p \cdot A_R') \sharp* A_R\rangle$
obtain q **where** $p \cdot \Psi_R' = (q::name\ prm) \cdot \Psi_R$ **and** S': $set\ q \subseteq (set\ A_R) \times set(p \cdot A_R')$ **and** $distinctPerm\ q$
apply *clarsimp*
apply(*drule sym*)
apply *simp*
by(*drule frameChainEq*) *auto*
from *PSimQ* **have** $(q \cdot (\Psi \otimes \Psi_R)) \rhd (q \cdot P) \sim (q \cdot Q)$
by(*rule bisimClosed*)
with $\langle A_R \sharp* \Psi\rangle\ \langle A_R \sharp* P\rangle\ \langle A_R \sharp* Q\rangle\ \langle(p \cdot A_R') \sharp* \Psi\rangle\ \langle(p \cdot A_R') \sharp* P\rangle\ \langle(p \cdot A_R') \sharp* Q\rangle\ S'$
have $\Psi \otimes (q \cdot \Psi_R) \rhd P \sim Q$ **by**(*simp add*: *eqvts*)
then have $(p \cdot (\Psi \otimes (q \cdot \Psi_R))) \rhd (p \cdot P) \sim (p \cdot Q)$ **by**(*rule bisimClosed*)
with $\langle A_R' \sharp* \Psi\rangle\ \langle A_R' \sharp* P\rangle\ \langle A_R' \sharp* Q\rangle\ \langle(p \cdot A_R') \sharp* \Psi\rangle\ \langle(p \cdot A_R') \sharp* P\rangle\ \langle(p \cdot A_R') \sharp* Q\rangle\ S\ \langle distinctPerm\ p\rangle\ \langle(p \cdot \Psi_R') = q \cdot \Psi_R\rangle$
have $\Psi \otimes \Psi_R' \rhd P \sim Q$
by(*drule-tac sym*) (*simp add*: *eqvts*)
}
ultimately show *?thesis*
by *blast*
qed
then have $(\Psi, P \parallel R, Q \parallel R) \in ?X \cup bisim$
by *simp*
}
moreover have $\bigwedge \Psi\ P\ Q\ xvec.\ [\![(\Psi, P, Q) \in ?X \cup bisim; (xvec::name\ list) \sharp* \Psi]\!] \Longrightarrow (\Psi, (\!|\nu*xvec|\!)P, (\!|\nu*xvec|\!)Q) \in ?X \cup bisim$
proof −
fix $\Psi\ P\ Q\ xvec$
assume $(\Psi, P, Q) \in ?X \cup bisim$
assume $(xvec::name\ list) \sharp* \Psi$
then show $(\Psi, (\!|\nu*xvec|\!)P, (\!|\nu*xvec|\!)Q) \in ?X \cup bisim$
proof(*induct xvec*)
case *Nil*
then show *?case* **using** $\langle(\Psi, P, Q) \in ?X \cup bisim\rangle$ **by** *simp*
next
case(*Cons x xvec*)

then show *?case* **by**(*simp only*: *resChain.simps*) (*rule Res*, *auto*)
qed
qed
ultimately have $\Psi \rhd P \parallel R \rightsquigarrow[(?X \cup bisim)]\ Q \parallel R$ **using** *statEqBisim*
by(*rule parPres*)
moreover assume (*xvec*::*name list*) $\sharp*\ \Psi$
moreover from ‹*eqvt ?X*› **have** *eqvt*(*?X* ∪ *bisim*) **by** *auto*
ultimately have $\Psi \rhd (\!|\nu*xvec|\!)(P \parallel R) \rightsquigarrow[(?X \cup bisim)]\ (\!|\nu*xvec|\!)(Q \parallel R)$ **using** *ResChain*
by(*intro resChainPres*)
}
with ‹$(\Psi, PR, QR) \in ?X$› **show** *?case* **by** *blast*
next
case(*cExt* Ψ *PR QR* Ψ')

from ‹$(\Psi, PR, QR) \in ?X$›
obtain *xvec P Q R* $A_R\ \Psi_R$ **where** *PFrR*: $PR = (\!|\nu*xvec|\!)(P \parallel R)$ **and** *QFrR*: $QR = (\!|\nu*xvec|\!)(Q \parallel R)$
and *xvec* $\sharp*\ \Psi$ **and** *A*: $\forall A_R\ \Psi_R.\ (extractFrame\ R = \langle A_R, \Psi_R\rangle \wedge A_R \sharp*\ \Psi \wedge A_R \sharp*\ P \wedge A_R \sharp*\ Q) \longrightarrow \Psi \otimes \Psi_R \rhd P \sim Q$
by *auto*

obtain *p* **where** $(p \cdot xvec) \sharp*\ \Psi$
and $(p \cdot xvec) \sharp*\ P$
and $(p \cdot xvec) \sharp*\ Q$
and $(p \cdot xvec) \sharp*\ R$
and $(p \cdot xvec) \sharp*\ \Psi'$
and *S*: $(set\ p) \subseteq (set\ xvec) \times (set(p \cdot xvec))$ **and** *distinctPerm p*
by(*rule name-list-avoiding*[**where** *c*=(Ψ, P, Q, R, Ψ')]) *auto*

from ‹$(p \cdot xvec) \sharp*\ P$› ‹$(p \cdot xvec) \sharp*\ R$› *S* **have** $(\!|\nu*xvec|\!)(P \parallel R) = (\!|\nu*(p \cdot xvec)|\!)(p \cdot (P \parallel R))$
by(*subst resChainAlpha*) *auto*
then have *PRAlpha*: $(\!|\nu*xvec|\!)(P \parallel R) = (\!|\nu*(p \cdot xvec)|\!)((p \cdot P) \parallel (p \cdot R))$
by(*simp add*: *eqvts*)

from ‹$(p \cdot xvec) \sharp*\ Q$› ‹$(p \cdot xvec) \sharp*\ R$› *S* **have** $(\!|\nu*xvec|\!)(Q \parallel R) = (\!|\nu*(p \cdot xvec)|\!)(p \cdot (Q \parallel R))$
by(*subst resChainAlpha*) *auto*
then have *QRAlpha*: $(\!|\nu*xvec|\!)(Q \parallel R) = (\!|\nu*(p \cdot xvec)|\!)((p \cdot Q) \parallel (p \cdot R))$
by(*simp add*: *eqvts*)

have $(\Psi \otimes \Psi', (\!|\nu*(p \cdot xvec)|\!)((p \cdot P) \parallel (p \cdot R)), (\!|\nu*(p \cdot xvec)|\!)((p \cdot Q) \parallel (p \cdot R))) \in ?X$
proof(*rule XI'*[**where** *C2*=$(\Psi, (p \cdot P), (p \cdot Q), R, \Psi', xvec, p \cdot xvec)$])
show $(p \cdot xvec) \sharp*\ (\Psi \otimes \Psi')$
using ‹$(p \cdot xvec) \sharp*\ \Psi$› ‹$(p \cdot xvec) \sharp*\ \Psi'$› **by** *auto*
next
fix $A_R\ \Psi_R$

assume *FrR*: *extractFrame* $(p \cdot R) = \langle A_R, \Psi_R\rangle$ **and** $A_R \sharp* (\Psi \otimes \Psi')$ **and** $A_R \sharp* (p \cdot P)$ **and** $A_R \sharp* (\Psi, p \cdot P, p \cdot Q, R, \Psi', xvec, p \cdot xvec)$
then have $A_R \sharp* \Psi$ **and** $A_R \sharp* (p \cdot Q)$
by *simp-all*
from *FrR* **have** $(p \cdot (extractFrame\ (p \cdot R))) = (p \cdot \langle A_R, \Psi_R\rangle)$
by *simp*
with ‹*distinctPerm p*› **have** $extractFrame\ R = \langle p \cdot A_R, p \cdot \Psi_R\rangle$
by(*simp add*: *eqvts*)
moreover from ‹$A_R \sharp* \Psi$› **have** $(p \cdot A_R) \sharp* (p \cdot \Psi)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$xvec \sharp* \Psi$› ‹$(p \cdot xvec) \sharp* \Psi$› *S* **have** $(p \cdot A_R) \sharp* \Psi$
by *simp*
moreover from ‹$A_R \sharp* (p \cdot P)$› **have** $(p \cdot A_R) \sharp* (p \cdot p \cdot P)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹*distinctPerm p*› **have** $(p \cdot A_R) \sharp* P$
by *simp*
moreover from ‹$A_R \sharp* (p \cdot Q)$› **have** $(p \cdot A_R) \sharp* (p \cdot p \cdot Q)$
by(*simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹*distinctPerm p*› **have** $(p \cdot A_R) \sharp* Q$
by *simp*
ultimately have $\Psi \otimes (p \cdot \Psi_R) \rhd P \sim Q$
using *A* **by** *blast*
then have $(\Psi \otimes (p \cdot \Psi_R)) \otimes (p \cdot \Psi') \rhd P \sim Q$
by(*rule bisimE*)
moreover have $(\Psi \otimes (p \cdot \Psi_R)) \otimes (p \cdot \Psi') \simeq (\Psi \otimes (p \cdot \Psi')) \otimes (p \cdot \Psi_R)$
by(*metis Associativity Commutativity Composition AssertionStatEqTrans AssertionStatEqSym*)
ultimately have $(\Psi \otimes (p \cdot \Psi')) \otimes (p \cdot \Psi_R) \rhd P \sim Q$
by(*rule statEqBisim*)
then have $(p \cdot ((\Psi \otimes (p \cdot \Psi')) \otimes (p \cdot \Psi_R))) \rhd (p \cdot P) \sim (p \cdot Q)$
by(*rule bisimClosed*)
with ‹*distinctPerm p*› ‹$xvec \sharp* \Psi$› ‹$(p \cdot xvec) \sharp* \Psi$› *S* **show** $(\Psi \otimes \Psi') \otimes \Psi_R \rhd (p \cdot P) \sim (p \cdot Q)$
by(*simp add*: *eqvts*)
qed
with *PFrR QFrR PRAlpha QRAlpha* **show** *?case* **by** *simp*
next
case(*cSym* Ψ *PR QR*)
then show *?case* **by**(*blast dest*: *bisimE*)
qed
qed

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 −
obtain A_R Ψ_R **where** *extractFrame* $R = \langle A_R, \Psi_R\rangle$ **and** $A_R \sharp* \Psi$ **and** $A_R \sharp* P$ **and** $A_R \sharp* Q$
by(*rule freshFrame*[**where** $C=(\Psi, P, Q)$]) *auto*
moreover from ‹$\Psi \rhd P \sim Q$› **have** $\Psi \otimes \Psi_R \rhd P \sim Q$ **by**(*rule bisimE*)
ultimately show *?thesis* **by**(*intro bisimParPresAux*)
qed

end

end
theory *Bisim-Struct-Cong*
imports *Bisim-Pres Sim-Struct-Cong Psi-Calculi.Structural-Congruence*
begin

This file is a (heavily modified) variant of the theory *Psi_Calculi.Bisim_Struct_Cong* from [1].

context *env* **begin**

lemma *bisimParComm*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)$ *psi*
and $Q :: ('a, 'b, 'c)$ *psi*

shows $\Psi \rhd P \parallel Q \sim Q \parallel P$
proof −
let *?X* = $\{((\Psi::'b), (\!|\nu*xvec|\!)((P::('a, 'b, 'c)\ psi) \parallel Q), (\!|\nu*xvec|\!)(Q \parallel P)) \mid xvec\ \Psi\ P\ Q.\ xvec \sharp* \Psi\}$

have *eqvt ?X*
by(*force simp add*: *eqvt-def pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*] *eqvts*)

have $(\Psi, P \parallel Q, Q \parallel P) \in$ *?X*
using *resChain.base* **by**(*smt* (*verit*) *freshSets*(*1*) *mem-Collect-eq*)
then show *?thesis*
proof(*coinduct rule*: *bisimWeakCoinduct*)
case(*cStatEq* Ψ *PQ QP*)
from ‹$(\Psi, PQ, QP) \in$ *?X*›
obtain *xvec P Q* **where** *PFrQ*: $PQ = (\!|\nu*xvec|\!)(P \parallel Q)$ **and** *QFrP*: $QP = (\!|\nu*xvec|\!)(Q \parallel P)$ **and** $xvec \sharp* \Psi$
by *auto*

obtain A_P Ψ_P **where** *FrP*: *extractFrame* $P = \langle A_P, \Psi_P\rangle$ **and** $A_P \sharp* \Psi$ **and** $A_P \sharp* Q$
by(*rule freshFrame*[**where** $C=(\Psi, Q)$]) *auto*
obtain A_Q Ψ_Q **where** *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$ **and** $A_Q \sharp* \Psi$ **and**

$A_Q \sharp* A_P$ **and** $A_Q \sharp* \Psi_P$
by(*rule freshFrame*[**where** $C=(\Psi, A_P, \Psi_P)$]) *auto*
from *FrQ* ‹$A_Q \sharp* A_P$› ‹$A_P \sharp* Q$› **have** $A_P \sharp* \Psi_Q$ **by**(*force dest*: *extractFrame-FreshChain*)
have $\langle(xvec@A_P@A_Q), \Psi \otimes \Psi_P \otimes \Psi_Q\rangle \simeq_F \langle(xvec@A_Q@A_P), \Psi \otimes \Psi_Q \otimes \Psi_P\rangle$
by(*simp add*: *frameChainAppend*)
(*metis frameResChainPres frameResChainComm frameNilStatEq compositionSym Associativity Commutativity FrameStatEqTrans*)
with *FrP FrQ PFrQ QFrP* ‹$A_P \sharp* \Psi_Q$› ‹$A_Q \sharp* \Psi_P$› ‹$A_Q \sharp* A_P$› ‹$xvec \sharp* \Psi$› ‹$A_P \sharp* \Psi$› ‹$A_Q \sharp* \Psi$›
show *?case* **by**(*auto simp add*: *frameChainAppend*)
next
case(*cSim* Ψ *PQ QP*)
from ‹$(\Psi, PQ, QP) \in ?X$›
obtain *xvec P Q* **where** *PFrQ*: $PQ = (\!|\nu*xvec|\!)(P \parallel Q)$ **and** *QFrP*: $QP = (\!|\nu*xvec|\!)(Q \parallel P)$
and $xvec \sharp* \Psi$
by *auto*
moreover have $\Psi \rhd (\!|\nu*xvec|\!)(P \parallel Q) \rightsquigarrow[?X] (\!|\nu*xvec|\!)(Q \parallel P)$
proof −
have $\Psi \rhd P \parallel Q \rightsquigarrow[?X] Q \parallel P$
proof −
note ‹*eqvt ?X*›
moreover have $\bigwedge\Psi\ P\ Q.\ (\Psi, P \parallel Q, Q \parallel P) \in ?X$
using *resChain.base* **by**(*smt* (*verit*) *freshSets*(*1*) *mem-Collect-eq*)
moreover have $\bigwedge\Psi\ P\ Q\ xvec.\ [\![(\Psi, P, Q) \in ?X;\ xvec \sharp* \Psi]\!] \Longrightarrow (\Psi, (\!|\nu*xvec|\!)P, (\!|\nu*xvec|\!)Q) \in ?X$
proof (*induct xvec*)
case *Nil*
then show *?case*
by (*smt* (*verit*, *ccfv-threshold*) *Pair-inject freshChainAppend mem-Collect-eq resChainAppend*)
next
case (*Cons a xvec*)
then show *?case*
by *blast*
qed
ultimately show *?thesis* **by**(*rule simParComm*)
qed
moreover note ‹*eqvt ?X*› ‹$xvec \sharp* \Psi$›
moreover have $\bigwedge\Psi\ P\ Q\ xvec.\ [\![(\Psi, P, Q) \in ?X;\ xvec \sharp* \Psi]\!] \Longrightarrow (\Psi, (\!|\nu*xvec|\!)P, (\!|\nu*xvec|\!)Q) \in ?X$
using *resChainAppend*[*symmetric*] *freshChainAppend* **by** *blast*
ultimately show *?thesis*
by(*rule resChainPres*)
qed
ultimately show *?case* **by** *simp*
next
case(*cExt* Ψ *PQ QP* Ψ')

from ‹$(\Psi,\ PQ,\ QP) \in ?X$›
obtain *xvec P Q* **where** *PFrQ*: $PQ = (\!|\nu{*}xvec|\!)(P \parallel Q)$ **and** *QFrP*: $QP = (\!|\nu{*}xvec|\!)(Q \parallel P)$
and $xvec \sharp* \Psi$
by *auto*

obtain p **where** $(p \cdot xvec) \sharp* \Psi$
and $(p \cdot xvec) \sharp* P$
and $(p \cdot xvec) \sharp* Q$
and $(p \cdot xvec) \sharp* \Psi'$
and *S*: $(set\ p) \subseteq (set\ xvec) \times (set(p \cdot xvec))$ **and** *distinctPerm p*
by(*rule name-list-avoiding*[**where** $c=(\Psi,\ P,\ Q,\ \Psi')$]) *auto*

from ‹$(p \cdot xvec) \sharp* P$› ‹$(p \cdot xvec) \sharp* Q$› *S* **have** $(\!|\nu{*}xvec|\!)(P \parallel Q) = (\!|\nu{*}(p \cdot xvec)|\!)(p \cdot (P \parallel Q))$
by(*subst resChainAlpha*) *auto*
then have *PQAlpha*: $(\!|\nu{*}xvec|\!)(P \parallel Q) = (\!|\nu{*}(p \cdot xvec)|\!)((p \cdot P) \parallel (p \cdot Q))$
by(*simp add*: *eqvts*)

from ‹$(p \cdot xvec) \sharp* P$› ‹$(p \cdot xvec) \sharp* Q$› *S* **have** $(\!|\nu{*}xvec|\!)(Q \parallel P) = (\!|\nu{*}(p \cdot xvec)|\!)(p \cdot (Q \parallel P))$
by(*subst resChainAlpha*) *auto*
then have *QPAlpha*: $(\!|\nu{*}xvec|\!)(Q \parallel P) = (\!|\nu{*}(p \cdot xvec)|\!)((p \cdot Q) \parallel (p \cdot P))$
by(*simp add*: *eqvts*)

from ‹$(p \cdot xvec) \sharp* \Psi$› ‹$(p \cdot xvec) \sharp* \Psi'$› **have** $(\Psi \otimes \Psi', (\!|\nu{*}(p \cdot xvec)|\!)((p \cdot P) \parallel (p \cdot Q)), (\!|\nu{*}(p \cdot xvec)|\!)((p \cdot Q) \parallel (p \cdot P))) \in ?X$
by *auto*
with *PFrQ QFrP PQAlpha QPAlpha* **show** *?case* **by** *simp*
next
case(*cSym* Ψ *PR QR*)
then show *?case* **by** *blast*
qed
qed

inductive-set *resCommRel* :: $('b \times ('a,'b,'c)\ psi \times ('a,'b,'c)\ psi)\ set$
where
resCommRel-refl : $(\Psi,P,P) \in resCommRel$
| *resCommRel-swap* : $[\![a \sharp \Psi;\ b \sharp \Psi]\!] \Longrightarrow (\Psi,(\!|\nu a|\!)((\!|\nu b|\!)P),(\!|\nu b|\!)((\!|\nu a|\!)P)) \in resCommRel$
| *resCommRel-res* : $[\![(\Psi,P,Q) \in resCommRel;\ a \sharp \Psi]\!] \Longrightarrow (\Psi,(\!|\nu a|\!)P,(\!|\nu a|\!)Q) \in resCommRel$

lemma *eqvtResCommRel*: *eqvt resCommRel*
proof −
{
fix Ψ *P Q*
and *p*::*name prm*
assume $(\Psi,P,Q) \in resCommRel$

```
    then have (p•Ψ,p•P,p•Q) ∈ resCommRel
    proof(induct rule: resCommRel.inducts)
      case resCommRel-refl then show ?case by(rule resCommRel.intros)
    next
      case(resCommRel-swap a Ψ b P)
      then have (p•a) ♯ p•Ψ and (p•b) ♯ p•Ψ
        apply -
       by(subst (asm) (1 2) perm-bool[where pi=p,symmetric], simp add: eqvts)+
      then show ?case unfolding eqvts
       by(rule resCommRel.intros)
    next
      case(resCommRel-res Ψ P Q a)
      from ‹a ♯ Ψ› have (p•a) ♯ p•Ψ
        apply -
        by(subst (asm) perm-bool[where pi=p,symmetric], simp add: eqvts)
      then show ?case using ‹(p • Ψ, p • P, p • Q) ∈ resCommRel›
        unfolding eqvts
        by(intro resCommRel.intros)
    qed
  }
  then show ?thesis unfolding eqvt-def
    by auto
qed

lemma resCommRelStarRes:
  assumes (Ψ,P,Q) ∈ resCommRel⋆
    and   a ♯ Ψ
  shows   (Ψ,(|νa|)P,(|νa|)Q) ∈ resCommRel⋆
  using assms
proof(induct rule: rel-trancl.induct)
  case r-into-rel-trancl then show ?case by(auto intro: resCommRel-res)
next
  case(rel-trancl-into-rel-trancl Ψ P Q R)
  then show ?case
    by(auto dest: resCommRel-res intro: rel-trancl.intros)
qed

lemma resCommRelStarResChain:
  assumes (Ψ,P,Q) ∈ resCommRel⋆
    and   xvec ♯* Ψ
  shows   (Ψ,(|ν*xvec|)P,(|ν*xvec|)Q) ∈ resCommRel⋆
  using assms
  by(induct xvec) (auto simp add: resCommRelStarRes)

lemma length-induct1[consumes 0, case-names Nil Cons]:
  assumes b: P []
    and   s: ⋀ x xvec. ⟦⋀ yvec. length xvec=length yvec⟹P yvec⟧ ⟹ P(x#xvec)
  shows   P xvec
proof(induct xvec rule: length-induct)
```

```
  case(1 xvec)
  then show ?case
  proof(cases xvec)
    case Nil then show ?thesis by(simp add: b)
  next
    case(Cons y yvec)
     with ‹∀ys. length ys < length xvec ⟶ P ys› have ∀ys. length ys = length
yvec ⟶ P ys by simp
    then show ?thesis unfolding Cons
      using s by auto
  qed
qed

lemma oneStepPerm-in-rel:
  assumes xvec ♯* Ψ
  shows (Ψ,(|ν*xvec|)P,(|ν*(rotate1 xvec)|)P) ∈ resCommRel⋆
  using assms
proof(induct xvec rule: length-induct1)
  case Nil then show ?case by(auto intro: resCommRel-refl)
next
  case(Cons x xvec)
  note Cons1 = this
  show ?case
  proof(cases xvec)
    case Nil then show ?thesis
      by(auto intro: resCommRel-refl)
  next
    case(Cons y yvec)
    then have x ♯ Ψ and y ♯ Ψ and xvec ♯* Ψ using ‹(x # xvec) ♯* Ψ›
      by simp+
     have (Ψ, (|ν*(x#y#yvec)|)P, (|ν*(y#x#yvec)|)P) ∈ resCommRel⋆ using ‹x ♯
Ψ› ‹y ♯ Ψ›
      by(auto intro: resCommRel-swap)
   moreover have (Ψ, (|ν*(y#x#yvec)|)P, (|ν*(y#rotate1(x#yvec))|)P) ∈ resComm-
Rel⋆
    proof −
       have (Ψ, (|ν*(x#yvec)|)P, (|ν*rotate1(x#yvec)|)P) ∈ resCommRel⋆ using
‹xvec ♯* Ψ› Cons ‹x ♯ Ψ›
        by(intro Cons1) auto
      then show ?thesis using ‹y ♯ Ψ›
        unfolding resChain.simps
        by(rule resCommRelStarRes)
    qed
    ultimately show ?thesis unfolding Cons
      by(auto intro: rel-trancl-transitive)
  qed
qed

lemma fresh-same-multiset:
```

```
  fixes xvec::name list
    and yvec::name list
  assumes mset xvec = mset yvec
    and   xvec ♯* X
  shows   yvec ♯* X
proof −
  from ‹mset xvec = mset yvec› have set xvec = set yvec
    using mset-eq-setD by blast
  then show ?thesis using ‹xvec ♯* X›
    unfolding fresh-def fresh-star-def name-list-supp
    by auto
qed

lemma nStepPerm-in-rel:
  assumes xvec ♯* Ψ
  shows (Ψ,(⦇ν*xvec⦈)P,(⦇ν*(rotate n xvec)⦈)P) ∈ resCommRel⋆
  using assms
proof(induct n)
  case 0 then show ?case by(auto intro: resCommRel-refl)
next
  case(Suc n)
  then have (Ψ, (⦇ν*xvec⦈)P, (⦇ν*rotate n xvec⦈)P) ∈ resCommRel⋆
    by simp
  moreover have (Ψ, (⦇ν*rotate n xvec⦈)P,(⦇ν*rotate (Suc n) xvec⦈)P) ∈ resComm-
Rel⋆
  proof −
    {
      fix xvec::name list
      assume xvec ♯* Ψ
      have rotate1 xvec ♯* Ψ using ‹xvec ♯* Ψ›
      proof(induct xvec)
        case Nil then show ?case by simp
      next
        case Cons then show ?case by simp
      qed
    }
    note f1 = this
    have rotate n xvec ♯* Ψ using ‹xvec ♯* Ψ›
      by(induct n) (auto simp add: f1)
    then show ?thesis
      by(auto simp add: oneStepPerm-in-rel)
  qed
  ultimately show ?case
    by(rule rel-trancl-transitive)
qed

lemma any-perm-in-rel:
  assumes xvec ♯* Ψ
    and   mset xvec = mset yvec
```

```
  shows (Ψ,(|ν*xvec|)P,(|ν*yvec|)P) ∈ resCommRel*
  using assms
proof(induct xvec arbitrary: yvec rule: length-induct1)
  case Nil then show ?case by(auto intro: resCommRel.intros)
next
  case(Cons x xvec)
  obtain yvec1 yvec2 where yveceq: yvec=yvec1@x#yvec2
  proof −
    assume 1: ⋀yvec1 yvec2. yvec = yvec1 @ x # yvec2 ⟹ thesis
    {
      note ‹mset (x # xvec) = mset yvec›
      then have ∃ yvec1 yvec2. yvec=yvec1@x#yvec2
      proof(induct xvec arbitrary: yvec rule: length-induct1)
        case Nil
        from ‹mset [x] = mset yvec› have yvec = [x]
          apply(induct yvec)
           apply simp
          apply(simp add: single-is-union)
          done
        then show ?case by blast
      next
        case(Cons x' xvec)
       have less: {#x'#} ⊆# mset yvec unfolding ‹mset (x # x' # xvec) = mset
yvec›[symmetric]
          by simp
        from ‹mset (x # x' # xvec) = mset yvec›
        have mset (x # xvec) = mset (remove1 x' yvec)
          by (metis mset-remove1 remove1.simps(2))
        then have ∃ yvec1 yvec2. (remove1 x' yvec) = yvec1 @ x # yvec2
          by(intro Cons) simp+
        then obtain yvec1 yvec2 where (remove1 x' yvec) = yvec1 @ x # yvec2
          by blast
        then show ?case
        proof(induct yvec arbitrary: yvec1 yvec2)
          case Nil then show ?case by simp
        next
          case(Cons y yvec) then show ?case
          proof(cases x'=y)
            case True
            with ‹remove1 x' (y # yvec) = yvec1 @ x # yvec2›
            have yvec = yvec1 @ x # yvec2
              by simp
            then show ?thesis
              by (metis append-Cons)
          next
            case False
            then have remove1 x' (y#yvec) = y # remove1 x' (yvec)
              by simp
            note Cons2=Cons
```

```
          show ?thesis
          proof(cases yvec1)
            case Nil
            then show ?thesis using Cons2 False
              by auto
          next
            case(Cons y1 yvec1a)
             from ‹remove1 x' (y # yvec) = yvec1 @ x # yvec2› ‹yvec1 = y1 #
yvec1a› False have y1=y
              by simp
            from ‹remove1 x' (y # yvec) = yvec1 @ x # yvec2›
           have remove1 x' yvec = yvec1a @ x # yvec2 unfolding Cons ‹y1=y›
using False
              by simp
            then have ∃yvec1 yvec2. yvec = yvec1 @ x # yvec2
              by(rule Cons2)
            then obtain yvec1 yvec2 where yvec = yvec1 @ x # yvec2
              by blast
            then show ?thesis
              by (metis append-Cons)
          qed
        qed
      qed
    qed
   }
   then show ?thesis using 1
    by blast
  qed
  from ‹(x # xvec) ♯* Ψ› have x ♯ Ψ and xvec ♯* Ψ by auto
  have mset (x # xvec) = mset(yvec1 @ x # yvec2)
   unfolding yveceq[symmetric] by fact
  then have mset (xvec) = mset(yvec1 @ yvec2)
   by(subst add-right-cancel[symmetric,where a={#x#}]) (simp add: add.assoc)
  then have (Ψ, ⦇ν*xvec⦈P, ⦇ν*(yvec1@yvec2)⦈P) ∈ resCommRel* using ‹xvec
♯* Ψ›
   by(intro Cons) auto
  moreover have (Ψ, ⦇ν*(yvec1@yvec2)⦈P, ⦇ν*(yvec2@yvec1)⦈P) ∈ resComm-
Rel*
  proof −
   have e: yvec2 @ yvec1 = rotate (length yvec1) (yvec1@yvec2)
    apply(cases yvec2)
     apply simp
    apply(simp add: rotate-drop-take)
    done
   have (yvec1@yvec2) ♯* Ψ using ‹mset (xvec) = mset(yvec1 @ yvec2)› ‹xvec
♯* Ψ›
    by(rule fresh-same-multiset)
   then show ?thesis unfolding e
    by(rule nStepPerm-in-rel)
```

```
  qed
  ultimately have (Ψ, (|νxvec|)P, (|ν(yvec2 @ yvec1)|)P) ∈ resCommRel⋆
    by(rule rel-trancl-transitive)
  then have (Ψ, (|ν(x#xvec)|)P, (|ν(x # yvec2 @ yvec1)|)P) ∈ resCommRel⋆
    unfolding resChain.simps using ‹x ♯ Ψ›
    by(rule resCommRelStarRes)
  moreover have (Ψ, (|ν(x # yvec2 @ yvec1)|)P, (|ν(yvec1 @ x # yvec2)|)P) ∈
resCommRel⋆
  proof −
    have e: yvec1 @ x # yvec2 = rotate (1+length yvec2) (x#yvec2@yvec1)
      apply(simp add: rotate-drop-take)
      apply(cases yvec1)
      by simp+
    have (yvec1 @ x # yvec2) ♯* Ψ using ‹mset (x # xvec) = mset(yvec1 @ x #
yvec2)› ‹(x#xvec) ♯* Ψ›
      by(rule fresh-same-multiset)
    then have (x # yvec2 @ yvec1) ♯* Ψ by simp
    then show ?thesis unfolding e
      by(rule nStepPerm-in-rel)
  qed
  ultimately show ?case unfolding yveceq
    by(rule rel-trancl-transitive)
qed

lemma bisimResComm:
  fixes x :: name
    and Ψ :: 'b
    and y :: name
    and P :: ('a, 'b, 'c) psi

shows Ψ ▷ (|νx|)((|νy|)P) ∼ (|νy|)((|νx|)P)
proof(cases x=y)
  case True
  then show ?thesis by(blast intro: bisimReflexive)
next
  case False
  {
    fix x::name and y::name and P::('a, 'b, 'c) psi
    assume x ♯ Ψ and y ♯ Ψ
    let ?X = resCommRel
    from ‹x ♯ Ψ› ‹y ♯ Ψ› have (Ψ, (|νx|)((|νy|)P), (|νy|)((|νx|)P)) ∈ ?X
      by(rule resCommRel-swap)
    then have Ψ ▷ (|νx|)((|νy|)P) ∼ (|νy|)((|νx|)P) using eqvtResCommRel
    proof(coinduct rule: bisimStarWeakCoinduct)
      case(cStatEq Ψ R S)
      then show ?case
      proof(induct rule: resCommRel.induct)
        case resCommRel-refl then show ?case by(rule FrameStatEqRefl)
      next
```

```
      case(resCommRel-swap x Ψ y P)
      moreover obtain A_P Ψ_P where extractFrame P = ⟨A_P, Ψ_P⟩ and A_P ♯* Ψ and x ♯ A_P and y ♯ A_P
        by(rule freshFrame[where C=(x, y, Ψ)]) auto
      ultimately show ?case by(force intro: frameResComm FrameStatEqTrans)
     next
      case(resCommRel-res Ψ P Q a)
      then show ?case by(auto intro: frameResPres)
     qed
    next
     case(cSim Ψ R S)
     have eqvt(resCommRel⋆)
      by(rule rel-trancl-eqvt) (rule eqvtResCommRel)
     from cSim show ?case
     proof(induct rule: resCommRel.induct)
      case(resCommRel-refl Ψ P)
      then show ?case
       by(rule simI[OF ‹eqvt(resCommRel⋆)›]) (blast intro: resCommRel.intros)
     next
      case(resCommRel-swap a Ψ b P)
      show ?case
       by(rule resComm) (fact|auto intro: resCommRel.intros any-perm-in-rel)+
     next
      case(resCommRel-res Ψ P Q a)
      show ?case
       by(rule resPres[where Rel=resCommRel⋆]) (fact|simp add: resCommRel-StarResChain)+
     qed
    next
     case(cExt Ψ R S Ψ') then show ?case
     proof(induct arbitrary: Ψ' rule: resCommRel.induct)
      case resCommRel-refl then show ?case by(rule resCommRel-refl)
     next
      case(resCommRel-swap a Ψ b P)
      then show ?case
      proof(cases a=b)
       case True show ?thesis unfolding ‹a=b› by(rule resCommRel-refl)
      next
       case False
       obtain x::name and y::name where x≠y and x ♯ Ψ and x ♯ Ψ' and x ♯ P and x ≠ a and x≠b and y ♯ Ψ and y ♯ Ψ' and y ♯ P and y ≠ a and y≠b
         apply(generate-fresh name)
         apply(generate-fresh name)
         by force
       then show ?thesis using False
         apply(subst (1 2) alphaRes[where x=a and y=x])
          apply assumption
          apply(simp add: fresh-abs-fun-iff[OF pt-name-inst, OF at-name-inst, OF fin-supp])
```

apply(*subst (1 2) alphaRes*[**where** $x{=}b$ **and** $y{=}y$])
apply(*simp add*: *fresh-abs-fun-iff*[*OF pt-name-inst*, *OF at-name-inst*, *OF fin-supp*] *fresh-left*)
apply *assumption*
unfolding *eqvts*
apply(*subst (1) cp1*[*OF cp-pt-inst*, *OF pt-name-inst*, *OF at-name-inst*])
by(*auto simp add*: *eqvts swap-simps intro*: *resCommRel.intros*)
qed
next
case(*resCommRel-res* Ψ P Q a)
obtain b::*name* **where** $b \sharp \Psi$ **and** $b \neq a$ **and** $b \sharp P$ **and** $b \sharp Q$ **and** $b \sharp \Psi'$
by(*generate-fresh name*) *auto*
have $(\Psi \otimes ([(a,b)]\cdot\Psi'), P, Q) \in$ *resCommRel* **by** *fact*
moreover from $\langle b \sharp \Psi' \rangle$ **have** $a \sharp ([(a,\ b)] \cdot \Psi')$ **by**(*simp add*: *fresh-left swap-simps*)
with $\langle a \sharp \Psi \rangle$ **have** $a \sharp \Psi \otimes ([(a,b)]\cdot\Psi')$ **by** *force*
ultimately have $(\Psi \otimes ([(a,b)]\cdot\Psi'), (\!|\nu a|\!)P, (\!|\nu a|\!)Q) \in$ *resCommRel*
by(*rule resCommRel.intros*)
then have $([(a,b)]\cdot(\Psi \otimes ([(a,b)]\cdot\Psi'), (\!|\nu a|\!)P, (\!|\nu a|\!)Q)) \in$ *resCommRel* **using** *eqvtResCommRel*
by(*intro eqvtI*)
then have $(\Psi \otimes \Psi', (\!|\nu b|\!)([(a,b)]\cdot P), (\!|\nu b|\!)([(a,b)]\cdot Q)) \in$ *resCommRel* **using** $\langle a \sharp \Psi \rangle$ $\langle b \sharp \Psi \rangle$
by(*simp add*: *eqvts swap-simps*)
then show *?case* **using** $\langle b \sharp Q \rangle$ $\langle b \sharp P \rangle$
apply(*subst (1 2) alphaRes*[**where** $x{=}a$ **and** $y{=}b$])
by(*assumption*|*simp only*: *eqvts*)+
qed
next
case(*cSym* Ψ R S) **then show** *?case*
by(*induct rule*: *resCommRel.inducts*) (*auto intro*: *resCommRel.intros*)
qed
}
moreover obtain x'::*name* **where** $x' \sharp \Psi$ **and** $x' \sharp P$ **and** $x' \neq x$ **and** $x' \neq y$
by(*generate-fresh name*) *auto*
moreover obtain y'::*name* **where** $y' \sharp \Psi$ **and** $y' \sharp P$ **and** $y' \neq x$ **and** $y' \neq y$ **and** $y' \neq x'$
by(*generate-fresh name*) *auto*
ultimately have $\Psi \rhd (\!|\nu x'|\!)((\!|\nu y'|\!)([(y,\ y'), (x,\ x')] \cdot P)) \sim (\!|\nu y'|\!)((\!|\nu x'|\!)([(y,\ y'), (x,\ x')] \cdot P))$ **by** *auto*
then show *?thesis* **using** $\langle x' \sharp P \rangle$ $\langle x' \neq x \rangle$ $\langle x' \neq y \rangle$ $\langle y' \sharp P \rangle$ $\langle y' \neq x \rangle$ $\langle y' \neq y \rangle$ $\langle y' \neq x' \rangle$ $\langle x \neq y \rangle$
apply(*subst alphaRes*[**where** $x{=}x$ **and** $y{=}x'$ **and** $P{=}P$], *auto*)
apply(*subst alphaRes*[**where** $x{=}y$ **and** $y{=}y'$ **and** $P{=}P$], *auto*)
apply(*subst alphaRes*[**where** $x{=}x$ **and** $y{=}x'$ **and** $P{=}(\!|\nu y'|\!)([(y,\ y')] \cdot P)$], *auto simp add*: *abs-fresh fresh-left*)
apply(*subst alphaRes*[**where** $x{=}y$ **and** $y{=}y'$ **and** $P{=}(\!|\nu x'|\!)([(x,\ x')] \cdot P)$], *auto simp add*: *abs-fresh fresh-left*)
by(*subst perm-compose*) (*simp add*: *eqvts calc-atm*)

qed

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

assumes $x \sharp \Psi$
  and *xvec* $\sharp*$ Ψ

shows $\Psi \rhd (\!|\nu x|\!)((\!|\nu{*}xvec|\!)P) \sim (\!|\nu{*}xvec|\!)((\!|\nu x|\!)P)$
  using *assms*
  by(*induct xvec*) (*auto intro*: *bisimResComm bisimReflexive bisimResPres bisimTransitive*)

lemma *bisimParPresSym*:
  fixes Ψ :: $'b$
    and P :: ($'a$, $'b$, $'c$) *psi*
    and Q :: ($'a$, $'b$, $'c$) *psi*
    and R :: ($'a$, $'b$, $'c$) *psi*

assumes $\Psi \rhd P \sim Q$

shows $\Psi \rhd R \parallel P \sim R \parallel Q$
  using *assms*
  by(*metis bisimParComm bisimParPres bisimTransitive*)

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 −
  {
    fix x::*name* **and** Q :: ($'a$, $'b$, $'c$) *psi*
    assume $x \sharp \Psi$ **and** $x \sharp P$
    let *?X1* = {((Ψ::$'b$), $(\!|\nu{*}xvec|\!)((\!|\nu{*}yvec|\!)((P::('a, 'b, 'c) psi) \parallel Q))$, $(\!|\nu{*}xvec|\!)(P \parallel ((\!|\nu{*}yvec|\!)Q))) \mid \Psi$ *xvec yvec P Q*. *yvec* $\sharp*$ $\Psi \wedge$ *yvec* $\sharp*$ *P* $\wedge$ *xvec* $\sharp*$ Ψ}
    let *?X2* = {((Ψ::$'b$), $(\!|\nu{*}xvec|\!)((P::('a, 'b, 'c) psi) \parallel ((\!|\nu{*}yvec|\!)Q))$, $(\!|\nu{*}xvec|\!)((\!|\nu{*}yvec|\!)(P \parallel Q))) \mid \Psi$ *xvec yvec P Q*. *yvec* $\sharp*$ $\Psi \wedge$ *yvec* $\sharp*$ *P* $\wedge$ *xvec* $\sharp*$ Ψ}
    let *?X* = *?X1* ∪ *?X2*
    from ‹$x \sharp \Psi$› ‹$x \sharp P$› **have** (Ψ, $(\!|\nu x|\!)(P \parallel Q)$, $P \parallel (\!|\nu x|\!)Q$) ∈ *?X*
    proof −
      from ‹$x \sharp \Psi$› ‹$x \sharp P$› **have** (Ψ, $(\!|\nu x|\!)(P \parallel Q)$, $P \parallel (\!|\nu x|\!)Q$) ∈ *?X1*

by (*smt* (*verit*, *del-insts*) *mem-Collect-eq freshSets*(*1*) *freshSets*(*5*) *resChain.base resChain.step*)
then show *?thesis* **by** *auto*
qed
moreover have *eqvt ?X*
by (*rule eqvtUnion*) (*clarsimp simp add*: *eqvt-def eqvts*, *metis fresh-star-bij*(*2*))+
ultimately have $\Psi \rhd (\!|\nu x|\!)(P \parallel Q) \sim P \parallel (\!|\nu x|\!)Q$
proof(*coinduct rule*: *transitiveStarCoinduct*)
case(*cStatEq* Ψ R T)
then have $(\Psi, R, T) \in$ *?X1* $\vee$ $(\Psi, R, T) \in$ *?X2*
by *blast*
then show *?case*
proof(*rule disjE*)
assume $(\Psi, R, T) \in$ *?X1*
then obtain *xvec yvec* P Q **where** $R = (\!|\nu{*}xvec|\!)((\!|\nu{*}yvec|\!)(P \parallel Q))$ **and** $T = (\!|\nu{*}xvec|\!)(P \parallel ((\!|\nu{*}yvec|\!)Q))$ **and** *xvec* $\sharp*$ Ψ **and** *yvec* $\sharp*$ P **and** *yvec* $\sharp*$ Ψ
by *auto*
moreover obtain A_P Ψ_P **where** *FrP*: *extractFrame* $P = \langle A_P, \Psi_P\rangle$ **and** $A_P \sharp* \Psi$ **and** *yvec* $\sharp*$ A_P **and** $A_P \sharp* Q$
by(*rule freshFrame*[**where** C=$(\Psi, yvec, Q)$]) *auto*
moreover obtain A_Q Ψ_Q **where** *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$ **and** $A_Q \sharp* \Psi$ **and** *yvec* $\sharp*$ A_Q **and** $A_Q \sharp* A_P$ **and** $A_Q \sharp* \Psi_P$
by(*rule freshFrame*[**where** C=$(\Psi, yvec, A_P, \Psi_P)$]) *auto*
moreover from *FrQ* ‹$A_P \sharp* Q$› ‹$A_Q \sharp* A_P$› **have** $A_P \sharp* \Psi_Q$
by(*auto dest*: *extractFrameFreshChain*)
moreover from ‹*yvec* $\sharp*$ P› ‹*yvec* $\sharp*$ A_P› *FrP* **have** *yvec* $\sharp*$ Ψ_P
by(*drule-tac extractFrameFreshChain*) *auto*
ultimately show *?case*
by(*auto simp add*: *frameChainAppend*[*symmetric*] *freshChainAppend*) (*auto simp add*: *frameChainAppend intro*: *frameResChainPres frameResChainComm*)
next
assume $(\Psi, R, T) \in$ *?X2*
then obtain *xvec yvec* P Q **where** $T = (\!|\nu{*}xvec|\!)((\!|\nu{*}yvec|\!)(P \parallel Q))$ **and** $R = (\!|\nu{*}xvec|\!)(P \parallel ((\!|\nu{*}yvec|\!)Q))$ **and** *xvec* $\sharp*$ Ψ **and** *yvec* $\sharp*$ P **and** *yvec* $\sharp*$ Ψ
by *auto*
moreover obtain A_P Ψ_P **where** *FrP*: *extractFrame* $P = \langle A_P, \Psi_P\rangle$ **and** $A_P \sharp* \Psi$ **and** *yvec* $\sharp*$ A_P **and** $A_P \sharp* Q$
by(*rule freshFrame*[**where** C=$(\Psi, yvec, Q)$]) *auto*
moreover obtain A_Q Ψ_Q **where** *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$ **and** $A_Q \sharp* \Psi$ **and** *yvec* $\sharp*$ A_Q **and** $A_Q \sharp* A_P$ **and** $A_Q \sharp* \Psi_P$
by(*rule freshFrame*[**where** C=$(\Psi, yvec, A_P, \Psi_P)$]) *auto*
moreover from *FrQ* ‹$A_P \sharp* Q$› ‹$A_Q \sharp* A_P$› **have** $A_P \sharp* \Psi_Q$
by(*auto dest*: *extractFrameFreshChain*)
moreover from ‹*yvec* $\sharp*$ P› ‹*yvec* $\sharp*$ A_P› *FrP* **have** *yvec* $\sharp*$ Ψ_P
by(*drule-tac extractFrameFreshChain*) *auto*
ultimately show *?case*
by(*auto simp add*: *frameChainAppend*[*symmetric*] *freshChainAppend*) (*auto simp add*: *frameChainAppend intro*: *frameResChainPres frameResChainComm*)
qed

next
case(*cSim* Ψ R T)
let *?Y* = $\{(\Psi, P, Q) \mid \Psi\ P\ P'\ Q'\ Q.\ \Psi \rhd P \sim P' \wedge ((\Psi, P', Q') \in ?X \vee \Psi \rhd P' \sim Q') \wedge \Psi \rhd Q' \sim Q\}$
from ‹*eqvt ?X*› **have** *eqvt ?Y* **by** *blast*
have *C1*: $\bigwedge \Psi\ R\ T\ y.\ [\![(\Psi, R, T) \in ?Y;\ (y::name) \sharp \Psi]\!] \Longrightarrow (\Psi, (\!|\nu y|\!)R, (\!|\nu y|\!)T) \in ?Y$
proof −
fix Ψ R T y
assume $(\Psi, R, T) \in ?Y$
then obtain R' T' **where** $\Psi \rhd R \sim R'$ **and** $(\Psi, R', T') \in (?X \cup bisim)$ **and** $\Psi \rhd T' \sim T$ **by** *force*
assume $(y::name) \sharp \Psi$
show $(\Psi, (\!|\nu y|\!)R, (\!|\nu y|\!)T) \in ?Y$
proof(*cases* $(\Psi, R', T') \in ?X$)
assume $(\Psi, R', T') \in ?X$
show *?thesis*
proof(*cases* $(\Psi, R', T') \in ?X1$)
assume $(\Psi, R', T') \in ?X1$
then obtain *xvec yvec P Q* **where** *R'eq*: $R' = (\!|\nu * xvec|\!)((\!|\nu * yvec|\!)(P \parallel Q))$ **and** *T'eq*: $T' = (\!|\nu * xvec|\!)(P \parallel ((\!|\nu * yvec|\!)\, Q))$
and *xvec* $\sharp*$ Ψ **and** *yvec* $\sharp*$ *P* **and** *yvec* $\sharp*$ Ψ
by *auto*
from ‹$\Psi \rhd R \sim R'$› ‹$y \sharp \Psi$› **have** $\Psi \rhd (\!|\nu y|\!)R \sim (\!|\nu y|\!)R'$ **by**(*rule bisimResPres*)
moreover from ‹*xvec* $\sharp*$ Ψ› ‹$y \sharp \Psi$› ‹*yvec* $\sharp*$ *P*› ‹*yvec* $\sharp*$ Ψ› **have** $(\Psi, (\!|\nu * (y\#xvec)|\!)(\!|\nu * yvec|\!)(P \parallel Q), (\!|\nu * (y\#xvec)|\!)(P \parallel ((\!|\nu * yvec|\!)\, Q))) \in ?X1$
by(*force simp del*: *resChain.simps*)
with *R'eq T'eq* **have** $(\Psi, (\!|\nu y|\!)R', (\!|\nu y|\!)\, T') \in ?X \cup bisim$ **by** *simp*
moreover from ‹$\Psi \rhd T' \sim T$› ‹$y \sharp \Psi$› **have** $\Psi \rhd (\!|\nu y|\!)\, T' \sim (\!|\nu y|\!)\, T$ **by**(*rule bisimResPres*)
ultimately show *?thesis* **by** *blast*
next
assume $(\Psi, R', T') \notin ?X1$
with ‹$(\Psi, R', T') \in ?X$› **have** $(\Psi, R', T') \in ?X2$ **by** *blast*
then obtain *xvec yvec P Q* **where** *T'eq*: $T' = (\!|\nu * xvec|\!)((\!|\nu * yvec|\!)(P \parallel Q))$ **and** *R'eq*: $R' = (\!|\nu * xvec|\!)(P \parallel ((\!|\nu * yvec|\!)\, Q))$ **and** *xvec* $\sharp*$ Ψ **and** *yvec* $\sharp*$ *P* **and** *yvec* $\sharp*$ Ψ
by *auto*
from ‹$\Psi \rhd R \sim R'$› ‹$y \sharp \Psi$› **have** $\Psi \rhd (\!|\nu y|\!)R \sim (\!|\nu y|\!)R'$ **by**(*rule bisimResPres*)
moreover from ‹*xvec* $\sharp*$ Ψ› ‹$y \sharp \Psi$› ‹*yvec* $\sharp*$ *P*› ‹*yvec* $\sharp*$ Ψ› **have** $(\Psi, (\!|\nu * (y\#xvec)|\!)(P \parallel ((\!|\nu * yvec|\!)\, Q)), (\!|\nu * (y\#xvec)|\!)((\!|\nu * yvec|\!)(P \parallel Q))) \in ?X2$
by(*force simp del*: *resChain.simps*)
with *R'eq T'eq* **have** $(\Psi, (\!|\nu y|\!)R', (\!|\nu y|\!)\, T') \in ?X \cup bisim$ **by** *simp*
moreover from ‹$\Psi \rhd T' \sim T$› ‹$y \sharp \Psi$› **have** $\Psi \rhd (\!|\nu y|\!)\, T' \sim (\!|\nu y|\!)\, T$ **by**(*rule bisimResPres*)
ultimately show *?thesis* **by** *blast*
qed

next
assume $(\Psi,\ R',\ T') \notin ?X$
with ‹$(\Psi,\ R',\ T') \in ?X \cup bisim$› **have** $\Psi \rhd R' \sim T'$ **by** *blast*
with ‹$\Psi \rhd R \sim R'$› ‹$\Psi \rhd T' \sim T$› ‹$y \sharp \Psi$› **show** *?thesis*
by(*blast dest*: *bisimResPres*)
qed
qed
have *C1'*: $\bigwedge \Psi\ R\ T\ y.\ [\![(\Psi,\ R,\ T) \in ?Y^\star;\ (y::name) \sharp \Psi]\!] \Longrightarrow (\Psi,\ (\!|\nu y|\!)R,\ (\!|\nu y|\!)T) \in ?Y^\star$
proof −
fix $\Psi\ R$
and $T::('a,'b,'c)\ psi$
and $y::name$
assume $(\Psi,\ R,\ T) \in ?Y^\star$
and $y \sharp \Psi$
then show $(\Psi,\ (\!|\nu y|\!)R,\ (\!|\nu y|\!)T) \in ?Y^\star$
proof(*induct rule*: *rel-trancl.induct*)
case(*r-into-rel-trancl* $\Psi\ P\ Q$)
then show *?case*
by (*intro rel-trancl.intros*(*1*)) (*rule C1*)
next
case(*rel-trancl-into-rel-trancl* $\Psi\ P\ Q\ R$)
then show *?case*
using *rel-trancl.intros*(*2*) *C1* **by** *meson*
qed
qed
have *C1''*: $\bigwedge \Psi\ R\ T\ yvec.\ [\![(\Psi,\ R,\ T) \in ?Y^\star;\ (yvec::name\ list) \sharp* \Psi]\!] \Longrightarrow (\Psi,\ (\!|\nu * yvec|\!)R,\ (\!|\nu * yvec|\!)T) \in ?Y^\star$
proof −
fix $\Psi\ R\ T$
and $yvec::name\ list$
assume $(\Psi,\ R,\ T) \in ?Y^\star$
and $yvec \sharp* \Psi$
then show $(\Psi,\ (\!|\nu * yvec|\!)R,\ (\!|\nu * yvec|\!)T) \in ?Y^\star$
apply(*induct yvec*)
apply *simp*
unfolding *resChain.simps*
by(*rule C1'*) *simp+*
qed
have *C2*: $\bigwedge y\ \Psi'\ R\ S\ zvec.\ [\![y \sharp \Psi';\ y \sharp R;\ zvec \sharp* \Psi']\!] \Longrightarrow (\Psi',\ (\!|\nu y|\!)((\!|\nu * zvec|\!)(R \parallel S)),\ (\!|\nu * zvec|\!)(R \parallel (\!|\nu y|\!)S)) \in ?Y^\star$
proof −
fix $y::name$
and $\Psi::'b$
and $R::('a,'b,'c)\ psi$
and $S::('a,'b,'c)\ psi$
and $zvec::name\ list$
assume $y \sharp \Psi$
and $y \sharp R$

and *zvec* $\sharp*$ Ψ
have $\Psi \rhd (\!|\nu y|\!)((\!|\nu{*}zvec|\!)(R \parallel S)) \sim ((\!|\nu{*}zvec|\!)((\!|\nu y|\!)(R \parallel S)))$
by(*rule bisimResComm'*) *fact+*
moreover have $(\Psi, ((\!|\nu{*}zvec|\!)((\!|\nu y|\!)(R \parallel S))), (\!|\nu{*}zvec|\!)(R \parallel (\!|\nu y|\!)S)) \in ?X$
using ‹$y \sharp \Psi$› ‹$y \sharp R$› ‹*zvec* $\sharp*$ Ψ›
apply *clarsimp*
apply(*rule exI*[**where** x=*zvec*])
apply(*rule exI*[**where** x=[y]])
by *auto*
moreover have $\Psi \rhd (\!|\nu{*}zvec|\!)(R \parallel (\!|\nu y|\!)S) \sim (\!|\nu{*}zvec|\!)(R \parallel (\!|\nu y|\!)S)$
by(*rule bisimReflexive*)
ultimately show $(\Psi, (\!|\nu y|\!)((\!|\nu{*}zvec|\!)(R \parallel S)), (\!|\nu{*}zvec|\!)(R \parallel (\!|\nu y|\!)S)) \in ?Y^\star$
by *blast*
qed
have *C2'*: $\bigwedge y\ \Psi'\ R\ S\ zvec.\ [\![y \sharp \Psi'; y \sharp R; zvec \sharp* \Psi']\!] \Longrightarrow (\Psi', (\!|\nu{*}zvec|\!)(R \parallel (\!|\nu y|\!)S), (\!|\nu y|\!)((\!|\nu{*}zvec|\!)(R \parallel S))) \in ?Y^\star$
proof −
fix y::*name*
and Ψ::$'b$
and R::($'a$,$'b$,$'c$) *psi*
and S::($'a$,$'b$,$'c$) *psi*
and *zvec*::*name list*
assume $y \sharp \Psi$
and $y \sharp R$
and *zvec* $\sharp*$ Ψ
have $\Psi \rhd ((\!|\nu{*}zvec|\!)((\!|\nu y|\!)(R \parallel S))) \sim (\!|\nu y|\!)((\!|\nu{*}zvec|\!)(R \parallel S))$
by(*rule bisimSymmetric*[*OF bisimResComm'*]) *fact+*
moreover have $(\Psi, (\!|\nu{*}zvec|\!)(R \parallel (\!|\nu y|\!)S), ((\!|\nu{*}zvec|\!)((\!|\nu y|\!)(R \parallel S)))) \in ?X$
using ‹$y \sharp \Psi$› ‹$y \sharp R$› ‹*zvec* $\sharp*$ Ψ›
apply(*intro UnI2*)
apply *clarsimp*
apply(*rule exI*[**where** x=*zvec*])
apply(*rule exI*[**where** x=[y]])
by *auto*
moreover have $\Psi \rhd (\!|\nu{*}zvec|\!)(R \parallel (\!|\nu y|\!)S) \sim (\!|\nu{*}zvec|\!)(R \parallel (\!|\nu y|\!)S)$
by(*rule bisimReflexive*)
ultimately show $(\Psi, (\!|\nu{*}zvec|\!)(R \parallel (\!|\nu y|\!)S), (\!|\nu y|\!)((\!|\nu{*}zvec|\!)(R \parallel S))) \in ?Y^\star$
by *blast*
qed
have *C3*: $\bigwedge \Psi'\ zvec\ R\ y.\ [\![y \sharp \Psi'; zvec \sharp* \Psi']\!] \Longrightarrow (\Psi', (\!|\nu y|\!)((\!|\nu{*}zvec|\!)R), (\!|\nu{*}zvec|\!)((\!|\nu y|\!)R)) \in ?Y^\star$
proof −
fix y::*name*
and Ψ::$'b$
and R::($'a$,$'b$,$'c$) *psi*
and *zvec*::*name list*
assume $y \sharp \Psi$
and *zvec* $\sharp*$ Ψ
then have $\Psi \rhd (\!|\nu y|\!)((\!|\nu{*}zvec|\!)R) \sim (\!|\nu{*}zvec|\!)((\!|\nu y|\!)R)$

by(*rule bisimResComm'*)
then show $(\Psi, (\!|\nu y|\!)((\!|\nu{*}zvec|\!)R), (\!|\nu{*}zvec|\!)((\!|\nu y|\!)R)) \in ?Y^\star$
by(*blast intro*: *bisimReflexive*)
qed
have *C4*: $\bigwedge \Psi'\ R\ S\ zvec.\ [\![zvec\ \sharp*\ R;\ zvec\ \sharp*\ \Psi']\!] \Longrightarrow (\Psi', ((\!|\nu{*}zvec|\!)(R \parallel S)), (R \parallel ((\!|\nu{*}zvec|\!)S))) \in ?Y^\star$
proof −
fix $\Psi::'b$
and $R::('a,'b,'c)\ psi$
and $S::('a,'b,'c)\ psi$
and $zvec::name\ list$
assume $zvec\ \sharp*\ R$
and $zvec\ \sharp*\ \Psi$
then have $(\Psi, ((\!|\nu{*}zvec|\!)(R \parallel S)), (R \parallel ((\!|\nu{*}zvec|\!)S))) \in ?X$
apply *clarsimp*
apply(*rule exI*[**where** x=[]])
apply(*rule exI*[**where** x=$zvec$])
by *auto*
then show $(\Psi, ((\!|\nu{*}zvec|\!)(R \parallel S)), (R \parallel ((\!|\nu{*}zvec|\!)S))) \in ?Y^\star$
by(*blast intro*: *bisimReflexive*)
qed
have *C4'*: $\bigwedge \Psi'\ R\ S\ zvec.\ [\![zvec\ \sharp*\ R;\ zvec\ \sharp*\ \Psi']\!] \Longrightarrow (\Psi', (R \parallel ((\!|\nu{*}zvec|\!)S)), ((\!|\nu{*}zvec|\!)(R \parallel S))) \in ?Y^\star$
proof −
fix $\Psi::'b$
and $R::('a,'b,'c)\ psi$
and $S::('a,'b,'c)\ psi$
and $zvec::name\ list$
assume $zvec\ \sharp*\ R$
and $zvec\ \sharp*\ \Psi$
then have $(\Psi, (R \parallel ((\!|\nu{*}zvec|\!)S)), ((\!|\nu{*}zvec|\!)(R \parallel S))) \in ?X$
apply(*intro UnI2*)
apply *clarsimp*
apply(*rule exI*[**where** x=[]])
apply(*rule exI*[**where** x=$zvec$])
by *auto*
then show $(\Psi, (R \parallel ((\!|\nu{*}zvec|\!)S)), ((\!|\nu{*}zvec|\!)(R \parallel S))) \in ?Y^\star$
by(*blast intro*: *bisimReflexive*)
qed
{
fix $\Psi\ P\ Q\ R$
assume $\Psi \rhd P \rightsquigarrow[?Y^\star]\ Q$
and $\Psi \rhd Q \rightsquigarrow[?Y^\star]\ R$
moreover note *rel-trancl-eqvt*[*OF* ‹*eqvt ?Y*›]
moreover have $\{(\Psi, P, R) \mid \Psi\ P\ R.\ \exists Q.\ (\Psi, P, Q) \in ?Y^\star \wedge (\Psi, Q, R) \in ?Y^\star\} \subseteq ?Y^\star$
by(*auto intro*: *rel-trancl-transitive*)
ultimately have $\Psi \rhd P \rightsquigarrow[?Y^\star]\ R$
by(*rule transitive*)

```
    }
    note trans = this

    show ?case
    proof(cases (Ψ, R, T) ∈ ?X1)
      assume (Ψ, R, T) ∈ ?X1
       then obtain xvec yvec P Q where Req: R = (⦇ν*xvec⦈)((⦇ν*yvec⦈)(P ∥ Q))
and Teq: T = (⦇ν*xvec⦈)(P ∥ ((⦇ν*yvec⦈)Q)) and xvec ♯* Ψ and yvec ♯* P and yvec
♯* Ψ
        by auto
      have Ψ ▷ (⦇ν*xvec⦈)((⦇ν*yvec⦈)(P ∥ Q)) ⇝[?Y*] (⦇ν*xvec⦈)(P ∥ ((⦇ν*yvec⦈)Q))
      proof −
        have Ψ ▷ (⦇ν*yvec⦈)(P ∥ Q) ⇝[?Y*] P ∥ ((⦇ν*yvec⦈)Q) using ‹yvec ♯* Ψ›
‹yvec ♯* P›
        proof(induct yvec arbitrary: Q)
          case Nil show ?case
            unfolding resChain.simps
         by(rule monotonic[where A=?Y]) (blast intro: reflexive bisimReflexive)+
        next
          case(Cons y yvec)
          then have yvec ♯* P and yvec ♯* Ψ and y ♯ Ψ and y ♯ P
            by simp+
          have Ψ ▷ (⦇ν*(y#yvec)⦈)P ∥ Q ⇝[?Y*] (⦇ν*yvec⦈)((⦇νy⦈)(P ∥ Q))
          proof −
            have Ψ ▷ (⦇ν*(y#yvec)⦈)P ∥ Q ⇝[bisim] (⦇ν*yvec⦈)((⦇νy⦈)(P ∥ Q))
              unfolding resChain.simps
              apply(rule bisimE)
              apply(rule bisimResComm′)
              by fact+
            then have Ψ ▷ (⦇ν*(y#yvec)⦈)P ∥ Q ⇝[?Y] (⦇ν*yvec⦈)((⦇νy⦈)(P ∥ Q))
              apply −
              apply(drule monotonic[where B=?Y])
              by auto (blast intro: bisimTransitive bisimReflexive)
            then show ?thesis
              apply −
              apply(drule monotonic[where B=?Y*])
              by blast
          qed
           moreover have Ψ ▷ (⦇ν*yvec⦈)((⦇νy⦈)(P ∥ Q)) ⇝[?Y*] (⦇ν*yvec⦈)(P ∥
(⦇νy⦈)Q)
          proof −
            have Ψ ▷ (⦇νy⦈)(P ∥ Q) ⇝[?Y*] P ∥ (⦇νy⦈)Q
              apply(rule scopeExtLeft)
                    apply fact
                   apply fact
                  apply(rule rel-trancl-eqvt)
                  apply fact
                 apply(blast intro: bisimReflexive)
              by fact+
```

```
    then show ?thesis using ‹yvec ♯* Ψ›
    proof(induct yvec)
      case Nil then show ?case by simp
    next
      case(Cons y' yvec)
      then show ?case
        unfolding resChain.simps
        apply −
        apply(rule resPres[where Rel=?Y⋆ and Rel'=?Y⋆])
           apply simp
          apply(rule rel-trancl-eqvt[OF ‹eqvt ?Y›])
         apply simp
        apply simp
       by(rule C1'')
    qed
  qed
moreover have Ψ ▷ (|ν∗yvec|)(P ∥ (|νy|)Q) ⇝[?Y⋆] P ∥ ((|ν∗yvec|)((|νy|)Q))
    by(rule Cons) fact+
moreover have Ψ ▷ P ∥ ((|ν∗yvec|)((|νy|)Q)) ⇝[?Y⋆] P ∥ ((|ν∗(y#yvec)|)Q)
  proof −
    have Ψ ▷ P ∥ ((|ν∗yvec|)((|νy|)Q)) ⇝[bisim] P ∥ ((|ν∗(y#yvec)|)Q)
      unfolding resChain.simps
      apply(rule bisimE)
      apply(rule bisimParPresSym)
      apply(rule bisimSymmetric[OF bisimResComm'])
      by fact+
    then have Ψ ▷ P ∥ ((|ν∗yvec|)((|νy|)Q)) ⇝[?Y] P ∥ ((|ν∗(y#yvec)|)Q)
      apply −
      apply(drule monotonic[where B=?Y])
      by auto (blast intro: bisimTransitive bisimReflexive)
    then show ?thesis
      apply −
      apply(drule monotonic[where B=?Y⋆])
      by blast
  qed
moreover have Ψ ▷ (|ν∗yvec|)P ∥ (|νy|)Q ⇝[?Y⋆] P ∥ ((|ν∗yvec|)((|νy|)Q))
    by(rule Cons) fact+
moreover have Ψ ▷ P ∥ ((|ν∗yvec|)((|νy|)Q)) ⇝[?Y⋆] P ∥ ((|ν∗(y#yvec)|)Q)
  proof −
    have Ψ ▷ P ∥ ((|ν∗yvec|)((|νy|)Q)) ⇝[bisim] P ∥ ((|νy|)((|ν∗yvec|)Q))
      apply(rule bisimE)
      apply(rule bisimParPresSym)
      apply(rule bisimSymmetric[OF bisimResComm'])
      by fact+
   then have Ψ ▷ P ∥ ((|ν∗yvec|)((|νy|)Q)) ⇝[?Y] P ∥ ((|νy|)((|ν∗yvec|)Q))
      apply −
      apply(drule monotonic[where B=?Y])
      by auto (blast intro: bisimTransitive bisimReflexive)
    then show ?thesis
```

```
            unfolding resChain.simps
            apply −
            apply(drule monotonic[where B=?Y⋆])
            by blast
          qed
          ultimately show ?case
            by(blast dest: trans)
        qed
        then show ?thesis using ‹xvec ♯* Ψ›
          apply(induct xvec)
           apply simp
          unfolding resChain.simps
          apply(rule resPres)
             apply simp
            apply(rule rel-trancl-eqvt[OF ‹eqvt ?Y›])
           apply simp
          apply simp
         apply(rule C1'')
          apply simp
         by assumption
      qed
      then show ?case unfolding Req Teq
        by −
    next
      assume (Ψ, R, T) ∉ ?X1
      then have (Ψ, R, T) ∈ ?X2 using ‹(Ψ, R, T) ∈ ?X› by blast
       then obtain xvec yvec P Q where Teq: T = (⦇ν*xvec⦈)((⦇ν*yvec⦈)(P ∥ Q))
and Req: R = (⦇ν*xvec⦈)(P ∥ ((⦇ν*yvec⦈)Q)) and xvec ♯* Ψ and yvec ♯* P and yvec
♯* Ψ
        by auto
      have Ψ ▷ (⦇ν*xvec⦈)(P ∥ ((⦇ν*yvec⦈)Q)) ⇝[?Y⋆] (⦇ν*xvec⦈)((⦇ν*yvec⦈)(P ∥ Q))
      proof −
        have Ψ ▷ P ∥ ((⦇ν*yvec⦈)Q) ⇝[?Y⋆] (⦇ν*yvec⦈)(P ∥ Q) using ‹yvec ♯* P›
‹yvec ♯* Ψ›
        proof(induct yvec arbitrary: Q)
          case Nil show ?case
            unfolding resChain.simps
         by(rule monotonic[where A=?Y]) (blast intro: reflexive bisimReflexive)+
        next
          case(Cons y yvec)
          then have yvec ♯* P and yvec ♯* Ψ and y ♯ Ψ and y ♯ P
            by simp+
          have Ψ ▷ (⦇ν*yvec⦈)((⦇νy⦈)(P ∥ Q)) ⇝[?Y⋆] (⦇ν*(y#yvec)⦈)P ∥ Q
          proof −
            have Ψ ▷ (⦇ν*yvec⦈)((⦇νy⦈)(P ∥ Q)) ⇝[bisim] (⦇ν*(y#yvec)⦈)P ∥ Q
              unfolding resChain.simps
              apply(rule bisimE)
              apply(rule bisimSymmetric[OF bisimResComm'])
              by fact+
```

```
      then have Ψ ▷ (|ν∗yvec|)((|νy|)(P ∥ Q)) ⇝[?Y] (|ν∗(y#yvec)|)P ∥ Q
        apply −
        apply(drule monotonic[where B=?Y])
        by auto (blast intro: bisimTransitive bisimReflexive)
      then show ?thesis
        apply −
        apply(drule monotonic[where B=?Y⋆])
        by blast
    qed
    moreover have Ψ ▷ (|ν∗yvec|)(P ∥ (|νy|)Q) ⇝[?Y⋆] (|ν∗yvec|)((|νy|)(P ∥
Q))
    proof −
      have Ψ ▷ P ∥ (|νy|)Q ⇝[?Y⋆] (|νy|)(P ∥ Q)
        apply(rule scopeExtRight)
            apply fact
           apply fact
          apply(rule rel-trancl-eqvt)
          apply fact
         apply(blast intro: bisimReflexive)
        by fact+
      then show ?thesis using ‹yvec ♯∗ Ψ›
      proof(induct yvec)
        case Nil then show ?case by simp
      next
        case(Cons y′ yvec)
        then show ?case
          unfolding resChain.simps
          apply −
          apply(rule resPres[where Rel=?Y⋆ and Rel′=?Y⋆])
             apply simp
            apply(rule rel-trancl-eqvt[OF ‹eqvt ?Y›])
           apply simp
          apply simp
         by(rule C1′′)
      qed
       moreover have Ψ ▷ P ∥ ((|ν∗yvec|)((|νy|)Q)) ⇝[?Y⋆] (|ν∗yvec|)(P ∥
(|νy|)Q)
        by(rule Cons) fact+
    moreover have Ψ ▷ P ∥ ((|ν∗(y#yvec)|)Q) ⇝[?Y⋆] P ∥ ((|ν∗yvec|)((|νy|)Q))
      proof −
        have Ψ ▷ P ∥ ((|ν∗(y#yvec)|)Q) ⇝[bisim] P ∥ ((|ν∗yvec|)((|νy|)Q))
          unfolding resChain.simps
          apply(rule bisimE)
          apply(rule bisimParPresSym)
          apply(rule bisimResComm′)
          by fact+
       then have Ψ ▷ P ∥ ((|ν∗(y#yvec)|)Q) ⇝[?Y] P ∥ ((|ν∗yvec|)((|νy|)Q))
          apply −
          apply(drule monotonic[where B=?Y])
```

```
          by auto (blast intro: bisimTransitive bisimReflexive)
        then show ?thesis
          apply −
          apply(drule monotonic[where B=?Y⋆])
          by blast
      qed
    qed
   moreover have Ψ ▷ P ∥ (⦇ν∗yvec⦈(⦇νy⦈ Q)) ⇝[?Y⋆] ⦇ν∗yvec⦈P ∥ ⦇νy⦈Q
    by(rule Cons) fact+
  moreover have Ψ ▷ P ∥ (⦇ν∗(y#yvec)⦈ Q) ⇝[?Y⋆] P ∥ (⦇ν∗yvec⦈(⦇νy⦈ Q))
    proof −
      have Ψ ▷ P ∥ (⦇νy⦈(⦇ν∗yvec⦈ Q)) ⇝[bisim] P ∥ (⦇ν∗yvec⦈(⦇νy⦈ Q))
        apply(rule bisimE)
        apply(rule bisimParPresSym)
        apply(rule bisimResComm′)
        by fact+
     then have Ψ ▷ P ∥ (⦇νy⦈(⦇ν∗yvec⦈ Q)) ⇝[?Y] P ∥ (⦇ν∗yvec⦈(⦇νy⦈ Q))
        apply −
        apply(drule monotonic[where B=?Y])
        by auto (blast intro: bisimTransitive bisimReflexive)
      then show ?thesis
        unfolding resChain.simps
        apply −
        apply(drule monotonic[where B=?Y⋆])
        by blast
    qed
    ultimately show ?case
      by(blast dest: trans)
  qed
  then show ?thesis using ‹xvec ♯∗ Ψ›
    apply(induct xvec)
     apply simp
    unfolding resChain.simps
    apply(rule resPres)
       apply simp
      apply(rule rel-trancl-eqvt[OF ‹eqvt ?Y›])
     apply simp
    apply simp
   apply(rule C1′′)
    apply simp
   by assumption
 qed
 then show ?case unfolding Req Teq
  by −
qed
next
 case(cExt Ψ R T Ψ′)
 show ?case
 proof(cases (Ψ, R, T) ∈ ?X1)
```

assume $(\Psi,\ R,\ T) \in ?X1$
then obtain *xvec yvec P Q* **where** *Req*: $R = (\!|\nu{*}xvec|\!)((\!|\nu{*}yvec|\!)(P \parallel Q))$ **and** *Teq*: $T = (\!|\nu{*}xvec|\!)(P \parallel ((\!|\nu{*}yvec|\!)Q))$ **and** $xvec \sharp* \Psi$ **and** $yvec \sharp* P$ **and** $yvec \sharp* \Psi$
by *auto*
obtain p **where** $(p \cdot yvec) \sharp* \Psi$ **and** $(p \cdot yvec) \sharp* P$ **and** $(p \cdot yvec) \sharp* Q$ **and** $(p \cdot yvec) \sharp* \Psi'$ **and** $(p \cdot yvec) \sharp* xvec$
and *S*: $(set\ p) \subseteq (set\ yvec) \times (set(p \cdot yvec))$ **and** *distinctPerm p*
by(*rule name-list-avoiding*[**where** c=$(\Psi,\ P,\ Q,\ xvec,\ \Psi')$]) *auto*
obtain q **where** $(q \cdot xvec) \sharp* \Psi$ **and** $(q \cdot xvec) \sharp* \Psi'$ **and** $(q \cdot xvec) \sharp* P$ **and** $(q \cdot xvec) \sharp* yvec$ **and** $(q \cdot xvec) \sharp* Q$ **and** $(q \cdot xvec) \sharp* (p{\cdot}P)$ **and** $(q \cdot xvec) \sharp* (p{\cdot}Q)$ **and** *distinctPerm q* **and** *T*: $(set\ q) \subseteq (set\ xvec) \times (set(q{\cdot}xvec))$ **and** $(q \cdot xvec) \sharp* (p{\cdot}yvec)$
by(*rule name-list-avoiding*[**where** c=$(\Psi,\ P,\ Q,\ yvec,\ p{\cdot}yvec,\ \Psi',p{\cdot}P,p{\cdot}Q)$]) *auto*
note $\langle(p \cdot yvec) \sharp* Q\rangle\ \langle set\ p \subseteq set\ yvec \times set\ (p \cdot yvec)\rangle$
moreover have $(p \cdot yvec) \sharp* (P \parallel Q)$ **using** $\langle(p \cdot yvec) \sharp* Q\rangle\ \langle(p \cdot yvec) \sharp* P\rangle$
by *simp*
moreover have $(\Psi \otimes \Psi',\ (\!|\nu{*}xvec|\!)(\!|\nu{*}p \cdot yvec|\!)p \cdot P \parallel Q,\ (\!|\nu{*}xvec|\!)P \parallel ((\!|\nu{*}p \cdot yvec|\!)p \cdot Q)) \in ?X$
proof −
have *Pdef*: $(p \cdot P) = P$ **using** $\langle set\ p \subseteq set\ yvec \times set(p{\cdot}yvec)\rangle\ \langle yvec \sharp* P\rangle\ \langle(p{\cdot}yvec) \sharp* P\rangle$
by *simp*
have *yvecdef*: $(q \cdot p \cdot yvec) = (p{\cdot}\ yvec)$ **using** $\langle set\ q \subseteq set\ xvec \times set(q{\cdot}xvec)\rangle\ \langle(p{\cdot}yvec) \sharp* xvec\rangle\ \langle(q{\cdot}xvec) \sharp* (p{\cdot}yvec)\rangle$
by *simp*
have $(q \cdot xvec) \sharp* (P \parallel ((\!|\nu{*}p \cdot yvec|\!)p \cdot Q))$ **using** $\langle(q \cdot xvec) \sharp* P\rangle\ \langle(q \cdot xvec) \sharp* (p{\cdot}Q)\rangle\ \langle(q \cdot xvec) \sharp* (p{\cdot}yvec)\rangle$
by *simp*
moreover have $(q \cdot xvec) \sharp* ((\!|\nu{*}p \cdot yvec|\!)p \cdot P \parallel Q)$
unfolding *eqvts Pdef*
using $\langle(q \cdot xvec) \sharp* P\rangle\ \langle(q \cdot xvec) \sharp* (p{\cdot}Q)\rangle\ \langle(q \cdot xvec) \sharp* (p{\cdot}yvec)\rangle$
by *simp*
moreover note $\langle set\ q \subseteq set\ xvec \times set\ (q \cdot xvec)\rangle$
moreover have $(\Psi \otimes \Psi',\ (\!|\nu{*}q \cdot xvec|\!)q \cdot (\!|\nu{*}p \cdot yvec|\!)p \cdot P \parallel Q,$
$(\!|\nu{*}q \cdot xvec|\!)q \cdot P \parallel ((\!|\nu{*}p \cdot yvec|\!)p \cdot Q)) \in ?X$
proof −
have $(p{\cdot}yvec) \sharp* (\Psi{\otimes}\Psi')$ **using** $\langle(p{\cdot}yvec) \sharp* \Psi\rangle\ \langle(p{\cdot}yvec) \sharp* \Psi'\rangle$
by *auto*
moreover have $(p{\cdot}yvec) \sharp* (q{\cdot}P)$ **using** $\langle(p{\cdot}yvec) \sharp* P\rangle$
apply(*subst yvecdef*[*symmetric*])
by(*subst fresh-star-bij*)
moreover have $(q{\cdot}xvec) \sharp* (\Psi{\otimes}\Psi')$ **using** $\langle(q{\cdot}xvec) \sharp* \Psi\rangle\ \langle(q{\cdot}xvec) \sharp* \Psi'\rangle$
by *auto*
ultimately show *?thesis*
unfolding *Pdef eqvts yvecdef*

by *blast*
qed
ultimately show *?thesis*
by(*subst* (*1 2*) *resChainAlpha*[**where** *p*=*q* **and** *xvec*=*xvec*])
qed
ultimately show *?case* **unfolding** *Req Teq*
apply(*intro disjI1*)
by(*subst* (*1 2*) *resChainAlpha*[**where** *p*=*p* **and** *xvec*=*yvec*])
next
assume (Ψ, *R*, *T*) ∉ *?X1*
then have (Ψ, *R*, *T*) ∈ *?X2* **using** ‹(Ψ,*R*,*T*) ∈ *?X*›
by *blast*
then obtain *xvec yvec P Q* **where** *Teq*: *T* = (⟮ν∗*xvec*⟯)((⟮ν∗*yvec*⟯)(*P* ∥ *Q*)) **and** *Req*: *R* = (⟮ν∗*xvec*⟯)(*P* ∥ ((⟮ν∗*yvec*⟯)*Q*)) **and** *xvec* ♯∗ Ψ **and** *yvec* ♯∗ *P* **and** *yvec* ♯∗ Ψ
by *auto*
obtain *p* **where** (*p* ∙ *yvec*) ♯∗ Ψ **and** (*p* ∙ *yvec*) ♯∗ *P* **and** (*p* ∙ *yvec*) ♯∗ *Q* **and** (*p* ∙ *yvec*) ♯∗ Ψ′ **and** (*p* ∙ *yvec*) ♯∗ *xvec*
and *S*: (*set p*) ⊆ (*set yvec*) × (*set*(*p* ∙ *yvec*)) **and** *distinctPerm p*
by(*rule name-list-avoiding*[**where** *c*=(Ψ, *P*, *Q*, *xvec*, Ψ′)]) *auto*
obtain *q* **where** (*q* ∙ *xvec*) ♯∗ Ψ **and** (*q* ∙ *xvec*) ♯∗ Ψ′ **and** (*q* ∙ *xvec*) ♯∗ *P* **and** (*q* ∙ *xvec*) ♯∗ *yvec* **and** (*q* ∙ *xvec*) ♯∗ *Q* **and** (*q* ∙ *xvec*) ♯∗ (*p*∙*P*) **and** (*q* ∙ *xvec*) ♯∗ (*p*∙*Q*) **and** *distinctPerm q* **and** *T*: (*set q*) ⊆ (*set xvec*) × (*set*(*q*∙*xvec*)) **and** (*q* ∙ *xvec*) ♯∗ (*p*∙*yvec*)
by(*rule name-list-avoiding*[**where** *c*=(Ψ, *P*, *Q*, *yvec*, *p*∙*yvec*, Ψ′,*p*∙*P*,*p*∙*Q*)]) *auto*
note ‹(*p* ∙ *yvec*) ♯∗ *Q*› ‹*set p* ⊆ *set yvec* × *set* (*p* ∙ *yvec*)›
moreover have (*p* ∙ *yvec*) ♯∗ (*P* ∥ *Q*) **using** ‹(*p* ∙ *yvec*) ♯∗ *Q*› ‹(*p* ∙ *yvec*) ♯∗ *P*›
by *simp*
moreover have (Ψ ⊗ Ψ′, ⟮ν∗*xvec*⟯*P* ∥ ((⟮ν∗*p* ∙ *yvec*⟯*p* ∙ *Q*), ⟮ν∗*xvec*⟯⟮ν∗*p* ∙ *yvec*⟯*p* ∙ *P* ∥ *Q*) ∈ *?X*
proof −
have *Pdef*: (*p* ∙ *P*) = *P* **using** ‹*set p* ⊆ *set yvec* × *set*(*p*∙*yvec*)› ‹*yvec* ♯∗ *P*› ‹(*p*∙*yvec*) ♯∗ *P*›
by *simp*
have *yvecdef*: (*q* ∙ *p* ∙ *yvec*) = (*p*∙ *yvec*) **using** ‹*set q* ⊆ *set xvec* × *set*(*q*∙*xvec*)› ‹(*p*∙*yvec*) ♯∗ *xvec*› ‹(*q*∙*xvec*) ♯∗ (*p*∙*yvec*)›
by *simp*
have (*q* ∙ *xvec*) ♯∗ (*P* ∥ ((⟮ν∗*p* ∙ *yvec*⟯*p* ∙ *Q*)) **using** ‹(*q* ∙ *xvec*) ♯∗ *P*› ‹(*q* ∙ *xvec*) ♯∗ (*p*∙*Q*)› ‹(*q* ∙ *xvec*) ♯∗ (*p*∙*yvec*)›
by *simp*
moreover have (*q* ∙ *xvec*) ♯∗ (⟮ν∗*p* ∙ *yvec*⟯*p* ∙ *P* ∥ *Q*)
unfolding *eqvts Pdef*
using ‹(*q* ∙ *xvec*) ♯∗ *P*› ‹(*q* ∙ *xvec*) ♯∗ (*p*∙*Q*)› ‹(*q* ∙ *xvec*) ♯∗ (*p*∙*yvec*)›
by *simp*
moreover note ‹*set q* ⊆ *set xvec* × *set* (*q* ∙ *xvec*)›
moreover have (Ψ ⊗ Ψ′, ⟮ν∗*q* ∙ *xvec*⟯*q* ∙ *P* ∥ ((⟮ν∗*p* ∙ *yvec*⟯*p* ∙ *Q*), ⟮ν∗*q* ∙ *xvec*⟯*q* ∙ ⟮ν∗*p* ∙ *yvec*⟯*p* ∙ *P* ∥ *Q*) ∈ *?X*

```
        proof −
          have (p•yvec) ♯* (Ψ⊗Ψ′) using ‹(p•yvec) ♯* Ψ› ‹(p•yvec) ♯* Ψ′›
            by auto
          moreover have (p•yvec) ♯* (q•P) using ‹(p•yvec) ♯* P›
            apply(subst yvecdef[symmetric])
            by(subst fresh-star-bij)
          moreover have (q•xvec) ♯* (Ψ⊗Ψ′) using ‹(q•xvec) ♯* Ψ› ‹(q•xvec) ♯*
Ψ′›
            by auto
          ultimately show ?thesis
            unfolding Pdef eqvts yvecdef
            by blast
        qed
        ultimately show ?thesis
          by(subst (1 2) resChainAlpha[where p=q and xvec=xvec])
      qed
      ultimately show ?case unfolding Req Teq
        apply(intro disjI1)
        by(subst (1 2) resChainAlpha[where p=p and xvec=yvec])
    qed
  next
    case(cSym Ψ P Q)
    then show ?case
      by(blast dest: bisimE)
  qed
 }
 moreover obtain y::name where y ♯ Ψ and y ♯ P y ♯ Q
   by(generate-fresh name) auto
 ultimately have Ψ ▷ (⦇νy⦈)(P ∥ ([(x, y)] • Q)) ∼ P ∥ (⦇νy⦈)([(x, y)] • Q) by auto
 then show ?thesis using assms ‹y ♯ P› ‹y ♯ Q›
   apply(subst alphaRes[where x=x and y=y and P=Q], auto)
   by(subst alphaRes[where x=x and y=y and P=P ∥ Q]) auto
qed

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

assumes xvec ♯* Ψ
  and   xvec ♯* P

shows Ψ ▷ (⦇ν*xvec⦈)(P ∥ Q) ∼ P ∥ (⦇ν*xvec⦈ Q)
  using assms
  by(induct xvec) (auto intro: bisimScopeExt bisimReflexive bisimTransitive bisim-
ResPres)

lemma bisimParAssoc:
```

fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $R :: ('a, 'b, 'c)\ psi$

shows $\Psi \rhd (P \parallel Q) \parallel R \sim P \parallel (Q \parallel R)$
proof −
let *?X* = $\{(\Psi, (\!|\nu{*}xvec|\!)((P \parallel Q) \parallel R), (\!|\nu{*}xvec|\!)(P \parallel (Q \parallel R))) \mid \Psi\ xvec\ P\ Q\ R.\ xvec \sharp* \Psi\}$
let *?Y* = $\{(\Psi, P, Q) \mid \Psi\ P\ P'\ Q'\ Q.\ \Psi \rhd P \sim P' \wedge (\Psi, P', Q') \in ?X \wedge \Psi \rhd Q' \sim Q\}$

have $(\Psi, (P \parallel Q) \parallel R, P \parallel (Q \parallel R)) \in ?X$
by(*clarsimp*, *rule exI*[**where** x=[]]) *auto*
moreover have *eqvt ?X* **by**(*force simp add*: *eqvt-def simp add*: *pt-fresh-star-bij*[*OF pt-name-inst*, *OF at-name-inst*] *eqvts*)
ultimately show *?thesis*
proof(*coinduct rule*: *weakTransitiveCoinduct'*)
case(*cStatEq* Ψ *PQR PQR'*)
from ‹$(\Psi, PQR, PQR') \in ?X$› **obtain** *xvec P Q R* **where** $xvec \sharp* \Psi$ **and** $PQR = (\!|\nu{*}xvec|\!)((P \parallel Q) \parallel R)$ **and** $PQR' = (\!|\nu{*}xvec|\!)(P \parallel (Q \parallel R))$
by *auto*
moreover obtain $A_P\ \Psi_P$ **where** *FrP*: *extractFrame* $P = \langle A_P, \Psi_P\rangle$ **and** $A_P \sharp* \Psi$ **and** $A_P \sharp* Q$ **and** $A_P \sharp* R$
by(*rule freshFrame*[**where** C=(Ψ, Q, R)]) *auto*
moreover obtain $A_Q\ \Psi_Q$ **where** *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$ **and** $A_Q \sharp* \Psi$ **and** $A_Q \sharp* A_P$ **and** $A_Q \sharp* \Psi_P$ **and** $A_Q \sharp* R$
by(*rule freshFrame*[**where** C=(Ψ, A_P, Ψ_P, R)]) *auto*
moreover obtain $A_R\ \Psi_R$ **where** *FrR*: *extractFrame* $R = \langle A_R, \Psi_R\rangle$ **and** $A_R \sharp* \Psi$ **and** $A_R \sharp* A_P$ **and** $A_R \sharp* \Psi_P$ **and** $A_R \sharp* A_Q$ **and** $A_R \sharp* \Psi_Q$
by(*rule freshFrame*[**where** C=$(\Psi, A_P, \Psi_P, A_Q, \Psi_Q)$]) *auto*
moreover from *FrQ* ‹$A_P \sharp* Q$› ‹$A_Q \sharp* A_P$› **have** $A_P \sharp* \Psi_Q$
by(*auto dest*: *extractFrameFreshChain*)
moreover from *FrR* ‹$A_P \sharp* R$› ‹$A_R \sharp* A_P$› **have** $A_P \sharp* \Psi_R$
by(*auto dest*: *extractFrameFreshChain*)
moreover from *FrR* ‹$A_Q \sharp* R$› ‹$A_R \sharp* A_Q$› **have** $A_Q \sharp* \Psi_R$
by(*auto dest*: *extractFrameFreshChain*)
ultimately show *?case* **using** *freshCompChain*
by *auto* (*metis frameChainAppend compositionSym Associativity frameNilStatEq frameResChainPres*)
next
case(*cSim* Ψ *T S*)
from ‹$(\Psi, T, S) \in ?X$› **obtain** *xvec P Q R* **where** $xvec \sharp* \Psi$ **and** *TEq*: $T = (\!|\nu{*}xvec|\!)((P \parallel Q) \parallel R)$
and *SEq*: $S = (\!|\nu{*}xvec|\!)(P \parallel (Q \parallel R))$
by *auto*
from ‹*eqvt ?X*›**have** *eqvt ?Y* **by** *blast*
have *C1*: $\bigwedge \Psi\ T\ S\ yvec.\ [\![(\Psi, T, S) \in ?Y;\ yvec \sharp* \Psi]\!] \Longrightarrow (\Psi, (\!|\nu{*}yvec|\!)T, (\!|\nu{*}yvec|\!)S) \in ?Y$

proof −
fix Ψ T S *yvec*
assume $(\Psi,\ T,\ S) \in ?Y$
then obtain T' S' **where** $\Psi \rhd T \sim T'$ **and** $(\Psi,\ T',\ S') \in ?X$ **and** $\Psi \rhd S' \sim S$ **by** *force*
assume (*yvec*::*name list*) $\sharp*$ Ψ
from ‹$(\Psi,\ T',\ S') \in ?X$› **obtain** *xvec* P Q R **where** $T'eq$: $T' = (\!|\nu*xvec|\!)((P \parallel Q) \parallel R)$ **and** $S'eq$: $S' = (\!|\nu*xvec|\!)(P \parallel (Q \parallel R))$
and *xvec* $\sharp*$ Ψ
by *auto*
from ‹$\Psi \rhd T \sim T'$› ‹*yvec* $\sharp*$ Ψ› **have** $\Psi \rhd (\!|\nu*yvec|\!)T \sim (\!|\nu*yvec|\!)T'$ **by**(*rule bisimResChainPres*)
moreover from ‹*xvec* $\sharp*$ Ψ› ‹*yvec* $\sharp*$ Ψ› **have** $(\Psi,\ (\!|\nu*(yvec@xvec)|\!)((P \parallel Q) \parallel R),\ (\!|\nu*(yvec@xvec)|\!)(P \parallel (Q \parallel R))) \in ?X$
by *force*
with $T'eq$ $S'eq$ **have** $(\Psi,\ (\!|\nu*yvec|\!)T',\ (\!|\nu*yvec|\!)S') \in ?X$ **by**(*simp add*: *resChainAppend*)
moreover from ‹$\Psi \rhd S' \sim S$› ‹*yvec* $\sharp*$ Ψ› **have** $\Psi \rhd (\!|\nu*yvec|\!)S' \sim (\!|\nu*yvec|\!)S$ **by**(*rule bisimResChainPres*)
ultimately show $(\Psi,\ (\!|\nu*yvec|\!)T,\ (\!|\nu*yvec|\!)S) \in ?Y$ **by** *blast*
qed

{
fix y
have $\bigwedge \Psi\ T\ S.\ [\![(\Psi,\ T,\ S) \in ?Y;\ y \sharp \Psi]\!] \Longrightarrow (\Psi,\ (\!|\nu y|\!)T,\ (\!|\nu y|\!)S) \in ?Y$
by(*drule C1*[**where** *yvec2*=[*y*]]) *auto*
}
note *C2* = *this*

have $\Psi \rhd (\!|\nu*xvec|\!)((P \parallel Q) \parallel R) \rightsquigarrow[?Y]\ (\!|\nu*xvec|\!)(P \parallel (Q \parallel R))$
proof −
have $\Psi \rhd (P \parallel Q) \parallel R \rightsquigarrow[?Y]\ P \parallel (Q \parallel R)$
proof −
note ‹*eqvt* ?*Y*›
moreover have $\bigwedge \Psi\ P\ Q\ R.\ (\Psi,\ (P \parallel Q) \parallel R,\ P \parallel (Q \parallel R)) \in ?Y$
proof −
fix Ψ P Q R
have $(\Psi::'b,\ ((P::('a,\ 'b,\ 'c)\ psi) \parallel Q) \parallel R,\ P \parallel (Q \parallel R)) \in ?X$
by(*clarsimp*, *rule exI*[**where** x=[]]) *auto*
then show $(\Psi,\ (P \parallel Q) \parallel R,\ P \parallel (Q \parallel R)) \in ?Y$
by(*blast intro*: *bisimReflexive*)
qed
moreover have $\bigwedge xvec\ \Psi\ P\ Q\ R.\ [\![xvec \sharp* \Psi;\ xvec \sharp* P]\!] \Longrightarrow (\Psi,\ (\!|\nu*xvec|\!)((P \parallel Q) \parallel R),\ P \parallel ((\!|\nu*xvec|\!)(Q \parallel R))) \in ?Y$
proof −
fix *xvec* Ψ P Q R
assume (*xvec*::*name list*) $\sharp*$ ($\Psi::'b$) **and** *xvec* $\sharp*$ ($P::('a,\ 'b,\ 'c)\ psi$)
from ‹*xvec* $\sharp*$ Ψ› **have** $(\Psi,\ (\!|\nu*xvec|\!)((P \parallel Q) \parallel R),\ (\!|\nu*xvec|\!)(P \parallel (Q \parallel R))) \in ?X$ **by** *blast*

moreover from ‹*xvec* ♯* Ψ› ‹*xvec* ♯* *P*› **have** Ψ ▷ (|ν*xvec|)(*P* ∥ (*Q* ∥ *R*)) ∼ *P* ∥ ((|ν*xvec|)(*Q* ∥ *R*))
by(*rule bisimScopeExtChain*)
ultimately show (Ψ, (|ν*xvec|)((*P* ∥ *Q*) ∥ *R*), *P* ∥ ((|ν*xvec|)(*Q* ∥ *R*))) ∈ *?Y*
by(*blast intro*: *bisimReflexive*)
qed
moreover have ⋀*xvec* Ψ *P Q R*. ⟦*xvec* ♯* Ψ; *xvec* ♯* *R*⟧ ⟹ (Ψ, ((|ν*xvec|)(*P* ∥ *Q*)) ∥ *R*, (|ν*xvec|)(*P* ∥ (*Q* ∥ *R*))) ∈ *?Y*
proof −
fix *xvec* Ψ *P Q R*
assume (*xvec*::*name list*) ♯* (Ψ::'*b*) **and** *xvec* ♯* (*R*::('*a*, '*b*, '*c*) *psi*)
have Ψ ▷ ((|ν*xvec|)(*P* ∥ *Q*)) ∥ *R* ∼ *R* ∥ ((|ν*xvec|)(*P* ∥ *Q*)) **by**(*rule bisimParComm*)
moreover from ‹*xvec* ♯* Ψ› ‹*xvec* ♯* *R*› **have** Ψ ▷ (|ν*xvec|)(*R* ∥ (*P* ∥ *Q*)) ∼ *R* ∥ ((|ν*xvec|)(*P* ∥ *Q*)) **by**(*rule bisimScopeExtChain*)
then have Ψ ▷ *R* ∥ ((|ν*xvec|)(*P* ∥ *Q*)) ∼ (|ν*xvec|)(*R* ∥ (*P* ∥ *Q*)) **by**(*rule bisimE*)
moreover from ‹*xvec* ♯* Ψ› **have** Ψ ▷ (|ν*xvec|)(*R* ∥ (*P* ∥ *Q*)) ∼ (|ν*xvec|)((*P* ∥ *Q*) ∥ *R*)
by(*metis bisimResChainPres bisimParComm*)
moreover from ‹*xvec* ♯* Ψ› **have** (Ψ, (|ν*xvec|)((*P* ∥ *Q*) ∥ *R*), (|ν*xvec|)(*P* ∥ (*Q* ∥ *R*))) ∈ *?X* **by** *blast*
ultimately show (Ψ, ((|ν*xvec|)(*P* ∥ *Q*)) ∥ *R*, (|ν*xvec|)(*P* ∥ (*Q* ∥ *R*))) ∈ *?Y* **by**(*blast dest*: *bisimTransitive intro*: *bisimReflexive*)
qed
ultimately show *?thesis* **using** *C1*
by(*rule parAssocLeft*)
qed
then show *?thesis* **using** ‹*eqvt ?Y*› ‹*xvec* ♯* Ψ› *C1*
by(*rule resChainPres*)
qed
with *TEq SEq* **show** *?case* **by** *simp*
next
case(*cExt* Ψ *T S* Ψ′)
from ‹(Ψ, *T*, *S*) ∈ *?X*› **obtain** *xvec P Q R* **where** *xvec* ♯* Ψ **and** *TEq*: *T* = (|ν*xvec|)((*P* ∥ *Q*) ∥ *R*)
and *SEq*: *S* = (|ν*xvec|)(*P* ∥ (*Q* ∥ *R*))
by *auto*
obtain *p* **where** (*p* ∙ *xvec*) ♯* Ψ **and** (*p* ∙ *xvec*) ♯* *P* **and** (*p* ∙ *xvec*) ♯* *Q* **and** (*p* ∙ *xvec*) ♯* *R* **and** (*p* ∙ *xvec*) ♯* Ψ′
and *S*: (*set p*) ⊆ (*set xvec*) × (*set*(*p* ∙ *xvec*)) **and** *distinctPerm p*
by(*rule name-list-avoiding*[**where** *c*=(Ψ, *P*, *Q*, *R*, Ψ′)]) *auto*

from ‹(*p* ∙ *xvec*) ♯* Ψ› ‹(*p* ∙ *xvec*) ♯* Ψ′› **have** (Ψ ⊗ Ψ′, (|ν*(*p* ∙ *xvec*)|)(((*p* ∙ *P*) ∥ (*p* ∙ *Q*)) ∥ (*p* ∙ *R*)), (|ν*(*p* ∙ *xvec*)|)((*p* ∙ *P*) ∥ ((*p* ∙ *Q*) ∥ (*p* ∙ *R*)))) ∈ *?X*
by *auto*
moreover from *TEq* ‹(*p* ∙ *xvec*) ♯* *P*› ‹(*p* ∙ *xvec*) ♯* *Q*› ‹(*p* ∙ *xvec*) ♯* *R*› *S* **have** *T* = (|ν*(*p* ∙ *xvec*)|)(((*p* ∙ *P*) ∥ (*p* ∙ *Q*)) ∥ (*p* ∙ *R*))

```
    apply clarsimp by(subst resChainAlpha[of p]) auto
   moreover from SEq ‹(p · xvec) ♯* P› ‹(p · xvec) ♯* Q› ‹(p · xvec) ♯* R› S
have S = (⦇ν*(p · xvec)⦈)((p · P) ∥ ((p · Q) ∥ (p · R)))
    apply clarsimp by(subst resChainAlpha[of p]) auto
   ultimately show ?case by simp
 next
   case(cSym Ψ T S)
   from ‹(Ψ, T, S) ∈ ?X› obtain xvec P Q R where xvec ♯* Ψ and TEq: T =
(⦇ν*xvec⦈)((P ∥ Q) ∥ R)
    and SEq: (⦇ν*xvec⦈)(P ∥ (Q ∥ R)) = S
    by auto

   from ‹xvec ♯* Ψ› have Ψ ▷ (⦇ν*xvec⦈)(P ∥ (Q ∥ R)) ∼ (⦇ν*xvec⦈)((R ∥ Q) ∥ P)
    by(metis bisimParComm bisimParPres bisimTransitive bisimResChainPres)
   moreover from ‹xvec ♯* Ψ› have (Ψ, (⦇ν*xvec⦈)((R ∥ Q) ∥ P), (⦇ν*xvec⦈)(R ∥
(Q ∥ P))) ∈ ?X by blast
   moreover from ‹xvec ♯* Ψ› have Ψ ▷ (⦇ν*xvec⦈)(R ∥ (Q ∥ P)) ∼ (⦇ν*xvec⦈)((P
∥ Q) ∥ R)
    by(metis bisimParComm bisimParPres bisimTransitive bisimResChainPres)
   ultimately show ?case using TEq SEq by(blast dest: bisimTransitive)
 qed
qed

lemma bisimParNil:
 fixes P :: ('a, 'b, 'c) psi

shows Ψ ▷ P ∥ 0 ∼ P
proof −
 let ?X1 = {(Ψ, P ∥ 0, P) | Ψ P. True}
 let ?X2 = {(Ψ, P, P ∥ 0) | Ψ P. True}
 let ?X = ?X1 ∪ ?X2
 have eqvt ?X by(auto simp add: eqvt-def)
 have (Ψ, P ∥ 0, P) ∈ ?X by simp
 then show ?thesis
 proof(coinduct rule: bisimWeakCoinduct)
   case(cStatEq Ψ Q R)
   show ?case
   proof(cases (Ψ, Q, R) ∈ ?X1)
     assume (Ψ, Q, R) ∈ ?X1
     then obtain P where Q = P ∥ 0 and R = P by auto
    moreover obtain A_P Ψ_P where extractFrame P = ⟨A_P, Ψ_P⟩ and A_P ♯* Ψ
      by(rule freshFrame)
     ultimately show ?case
      apply clarsimp by(metis frameResChainPres frameNilStatEq Identity Asso-
ciativity AssertionStatEqTrans Commutativity)
   next
     assume (Ψ, Q, R) ∉ ?X1
     with ‹(Ψ, Q, R) ∈ ?X› have (Ψ, Q, R) ∈ ?X2 by blast
     then obtain P where Q = P and R = P ∥ 0 by auto
```

```
    moreover obtain A_P Ψ_P where extractFrame P = ⟨A_P, Ψ_P⟩ and A_P ♯* Ψ
      by(rule freshFrame)
    ultimately show ?case
     apply clarsimp by(metis frameResChainPres frameNilStatEq Identity Associativity AssertionStatEqTrans AssertionStatEqSym Commutativity)
   qed
  next
   case(cSim Ψ Q R)
   then show ?case using ‹eqvt ?X›
    by(auto intro: parNilLeft parNilRight)
  next
   case(cExt Ψ Q R Ψ')
   then show ?case by auto
  next
   case(cSym Ψ Q R)
   then show ?case by auto
  qed
qed

lemma bisimResNil:
  fixes x :: name
    and Ψ :: 'b

shows Ψ ▷ (|νx|)0 ∼ 0
proof −
  {
    fix x::name
    assume x ♯ Ψ
    have Ψ ▷ (|νx|)0 ∼ 0
    proof −
      let ?X1 = {(Ψ, (|νx|)0, 0) | Ψ x. x ♯ Ψ}
      let ?X2 = {(Ψ, 0, (|νx|)0) | Ψ x. x ♯ Ψ}
      let ?X = ?X1 ∪ ?X2

      from ‹x ♯ Ψ› have (Ψ, (|νx|)0, 0) ∈ ?X by auto
      then show ?thesis
      proof(coinduct rule: bisimWeakCoinduct)
        case(cStatEq Ψ P Q)
        then show ?case using freshComp by(force intro: frameResFresh FrameStatEqSym)
      next
        case(cSim Ψ P Q)
        then show ?case
          by(force intro: resNilLeft resNilRight)
      next
        case(cExt Ψ P Q Ψ')
        obtain y where y ♯ Ψ and y ♯ Ψ' and y ≠ x
          by(generate-fresh name) (auto simp add: fresh-prod)
        show ?case
```

```
      proof(cases (Ψ, P, Q) ∈ ?X1)
        assume (Ψ, P, Q) ∈ ?X1
        then obtain x where P = (|νx|)0 and Q = 0 by auto
        moreover have (|νx|)0 = (|νy|) 0 by(subst alphaRes) auto
        ultimately show ?case using ‹y ♯ Ψ› ‹y ♯ Ψ'› by auto
      next
        assume (Ψ, P, Q) ∉ ?X1
        with ‹(Ψ, P, Q) ∈ ?X› have (Ψ, P, Q) ∈ ?X2 by auto
        then obtain x where Q = (|νx|)0 and P = 0 by auto
        moreover have (|νx|)0 = (|νy|) 0 by(subst alphaRes) auto
        ultimately show ?case using ‹y ♯ Ψ› ‹y ♯ Ψ'› by auto
      qed
    next
      case(cSym Ψ P Q)
      then show ?case by auto
    qed
  qed
  }
  moreover obtain y::name where y ♯ Ψ by(generate-fresh name) auto
  ultimately have Ψ ▷ (|νy|)0 ∼ 0 by auto
  then show ?thesis by(subst alphaRes[where x=x and y=y]) auto
qed

lemma bisimOutputPushRes:
  fixes x :: name
    and Ψ :: 'b
    and M :: 'a
    and N :: 'a
    and P :: ('a, 'b, 'c) psi

assumes x ♯ M
  and   x ♯ N

shows Ψ ▷ (|νx|)(M⟨N⟩.P) ∼ M⟨N⟩.(|νx|)P
proof −
  {
    fix x::name and P::('a, 'b, 'c) psi
    assume x ♯ Ψ and x ♯ M and x ♯ N
    let ?X1 = {(Ψ, (|νx|)(M⟨N⟩.P), M⟨N⟩.(|νx|)P) | Ψ x M N P. x ♯ Ψ ∧ x ♯ M ∧
x ♯ N}
    let ?X2 = {(Ψ, M⟨N⟩.(|νx|)P, (|νx|)(M⟨N⟩.P)) | Ψ x M N P. x ♯ Ψ ∧ x ♯ M ∧
x ♯ N}
    let ?X = ?X1 ∪ ?X2

    have eqvt ?X
      by(rule eqvtUnion) (force simp add: eqvt-def pt-fresh-bij[OF pt-name-inst, OF
at-name-inst] eqvts)+
    from ‹x ♯ Ψ› ‹x ♯ M› ‹x ♯ N› have (Ψ, (|νx|)(M⟨N⟩.P), M⟨N⟩.(|νx|)P) ∈ ?X
      by auto
```

```
  then have Ψ ▷ (⦇νx⦈)(M⟨N⟩.P) ∼ M⟨N⟩.(⦇νx⦈)P
  proof(coinduct rule: bisimCoinduct)
    case(cStatEq Ψ Q R)
     then show ?case using freshComp by(force intro: frameResFresh FrameStatEqSym)
  next
    case(cSim Ψ Q R)
    then show ?case using ‹eqvt ?X›
      by(auto intro!: outputPushResLeft outputPushResRight bisimReflexive)
  next
    case(cExt Ψ Q R Ψ')
    show ?case
    proof(cases (Ψ, Q, R) ∈ ?X1)
      assume (Ψ, Q, R) ∈ ?X1
       then obtain x M N P where Qeq: Q = (⦇νx⦈)(M⟨N⟩.P) and Req: R = M⟨N⟩.(⦇νx⦈)P and x ♯ Ψ and x ♯ M and x ♯ N by auto
      obtain y::name where y ♯ Ψ and y ♯ Ψ' and y ♯ M and y ♯ N and y ♯ P
        by(generate-fresh name) (auto simp add: fresh-prod)

      moreover then have (Ψ ⊗ Ψ', (⦇νy⦈)(M⟨N⟩.([(x, y)] ∙ P)), M⟨N⟩.(⦇νy⦈)([(x, y)] ∙ P)) ∈ ?X by auto
       moreover from Qeq ‹x ♯ M› ‹y ♯ M› ‹x ♯ N› ‹y ♯ N› ‹y ♯ P› have Q = (⦇νy⦈)(M⟨N⟩.([(x, y)] ∙ P))
        apply clarsimp by(subst alphaRes[of y]) (auto simp add: eqvts)
       moreover from Req ‹y ♯ P› have R = M⟨N⟩.(⦇νy⦈)([(x, y)] ∙ P)
        apply clarsimp by(subst alphaRes[of y]) (auto simp add: eqvts)
       ultimately show ?case by blast
     next
      assume (Ψ, Q, R) ∉ ?X1
      with ‹(Ψ, Q, R) ∈ ?X› have (Ψ, Q, R) ∈ ?X2 by blast
       then obtain x M N P where Req: R = (⦇νx⦈)(M⟨N⟩.P) and Qeq: Q = M⟨N⟩.(⦇νx⦈)P and x ♯ Ψ and x ♯ M and x ♯ N by auto
      obtain y::name where y ♯ Ψ and y ♯ Ψ' and y ♯ M and y ♯ N and y ♯ P
        by(generate-fresh name) (auto simp add: fresh-prod)

      moreover then have (Ψ ⊗ Ψ', (⦇νy⦈)(M⟨N⟩.([(x, y)] ∙ P)), M⟨N⟩.(⦇νy⦈)([(x, y)] ∙ P)) ∈ ?X by auto
       moreover from Req ‹x ♯ M› ‹y ♯ M› ‹x ♯ N› ‹y ♯ N› ‹y ♯ P› have R = (⦇νy⦈)(M⟨N⟩.([(x, y)] ∙ P))
        apply clarsimp by(subst alphaRes[of y]) (auto simp add: eqvts)
       moreover from Qeq ‹y ♯ P› have Q = M⟨N⟩.(⦇νy⦈)([(x, y)] ∙ P)
        apply clarsimp by(subst alphaRes[of y]) (auto simp add: eqvts)
       ultimately show ?case by blast
     qed
  next
    case(cSym Ψ R Q)
    then show ?case by blast
  qed
 }
```

moreover obtain $y::name$ **where** $y \sharp \Psi$ **and** $y \sharp M$ **and** $y \sharp N$ $y \sharp P$
by(*generate-fresh name*) *auto*
ultimately have $\Psi \rhd (\!|\nu y|\!)(M\langle N\rangle.([(x,\ y)] \cdot P)) \sim M\langle N\rangle.(\!|\nu y|\!)([(x,\ y)] \cdot P)$ **by** *auto*
then show *?thesis* **using** *assms* ‹$y \sharp P$› ‹$y \sharp M$› ‹$y \sharp N$›
apply(*subst alphaRes*[**where** $x{=}x$ **and** $y{=}y$ **and** $P{=}P$], *auto*)
by(*subst alphaRes*[**where** $x{=}x$ **and** $y{=}y$ **and** $P{=}M\langle N\rangle.P$]) *auto*
qed

lemma *bisimInputPushRes*:
fixes x :: *name*
and Ψ :: $'b$
and M :: $'a$
and *xvec* :: *name list*
and N :: $'a$
and P :: ($'a$, $'b$, $'c$) *psi*

assumes $x \sharp M$
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 −
{
fix $x::name$ **and** $P::('a,\ 'b,\ 'c)$ *psi*
assume $x \sharp \Psi$ **and** $x \sharp M$ **and** $x \sharp N$ **and** $x \sharp xvec$
let *?X1* = {(Ψ, $(\!|\nu x|\!)(M(\!|\lambda{*}xvec\ N|\!).P)$, $M(\!|\lambda{*}xvec\ N|\!).(\!|\nu x|\!)P$) | Ψ x M *xvec* N P. $x \sharp \Psi \wedge x \sharp M \wedge x \sharp xvec \wedge x \sharp N$}
let *?X2* = {(Ψ, $M(\!|\lambda{*}xvec\ N|\!).(\!|\nu x|\!)P$, $(\!|\nu x|\!)(M(\!|\lambda{*}xvec\ N|\!).P)$) | Ψ x M *xvec* N P. $x \sharp \Psi \wedge x \sharp M \wedge x \sharp xvec \wedge x \sharp N$}
let *?X* = *?X1* ∪ *?X2*

have *eqvt ?X*
by(*rule eqvtUnion*) (*force simp add*: *eqvt-def pt-fresh-bij*[*OF pt-name-inst*, *OF at-name-inst*] *eqvts*)+

from ‹$x \sharp \Psi$› ‹$x \sharp M$› ‹$x \sharp xvec$› ‹$x \sharp N$› **have** (Ψ, $(\!|\nu x|\!)(M(\!|\lambda{*}xvec\ N|\!).P)$, $M(\!|\lambda{*}xvec\ N|\!).(\!|\nu x|\!)P$) ∈ *?X*
by *blast*
then have $\Psi \rhd (\!|\nu x|\!)(M(\!|\lambda{*}xvec\ N|\!).P) \sim M(\!|\lambda{*}xvec\ N|\!).(\!|\nu x|\!)P$
proof(*coinduct rule*: *bisimCoinduct*)
case(*cStatEq* Ψ Q R)
then show *?case* **using** *freshComp* **by**(*force intro*: *frameResFresh FrameStatEqSym*)
next
case(*cSim* Ψ Q R)
then show *?case* **using** ‹*eqvt ?X*›
by(*auto intro*!: *inputPushResLeft inputPushResRight bisimReflexive*)
next

case(*cExt* Ψ Q R Ψ')
show *?case*
proof(*cases* (Ψ, Q, R) $\in$ *?X1*)
assume (Ψ, Q, R) $\in$ *?X1*
then obtain x M *xvec* N P **where** *Qeq*: $Q = (\!|\nu x|\!)(M(\!|\lambda * xvec\ N|\!).P)$ **and** *Req*: $R = M(\!|\lambda * xvec\ N|\!).(\!|\nu x|\!)P$ **and** $x \sharp \Psi$
and $x \sharp M$ **and** $x \sharp xvec$ **and** $x \sharp N$ **by** *auto*
obtain y::*name* **where** $y \sharp \Psi$ **and** $y \sharp \Psi'$ **and** $y \sharp M$ **and** $y \sharp N$ **and** $y \sharp P$ **and** $y \sharp xvec$
by(*generate-fresh name*) (*auto simp add*: *fresh-prod*)

moreover then have ($\Psi \otimes \Psi'$, $(\!|\nu y|\!)(M(\!|\lambda * xvec\ N|\!).([(x, y)] \cdot P))$, $M(\!|\lambda * xvec\ N|\!).(\!|\nu y|\!)([(x, y)] \cdot P)) \in$ *?X* **by** *force*
moreover from *Qeq* ‹$x \sharp M$› ‹$y \sharp M$› ‹$x \sharp xvec$› ‹$y \sharp xvec$› ‹$x \sharp N$› ‹$y \sharp N$› ‹$y \sharp P$› **have** $Q = (\!|\nu y|\!)(M(\!|\lambda * xvec\ N|\!).([(x, y)] \cdot P))$
apply *clarsimp* **by**(*subst alphaRes*[*of y*]) (*auto simp add*: *eqvts inputChainFresh*)
moreover from *Req* ‹$y \sharp P$› **have** $R = M(\!|\lambda * xvec\ N\ |\!).(\!|\nu y|\!)([(x, y)] \cdot P)$
apply *clarsimp* **by**(*subst alphaRes*[*of y*]) (*auto simp add*: *eqvts*)
ultimately show *?case* **by** *blast*
next
assume (Ψ, Q, R) $\notin$ *?X1*
with ‹(Ψ, Q, R) $\in$ *?X*› **have** (Ψ, Q, R) $\in$ *?X2* **by** *blast*
then obtain x M *xvec* N P **where** *Req*: $R = (\!|\nu x|\!)(M(\!|\lambda * xvec\ N|\!).P)$ **and** *Qeq*: $Q = M(\!|\lambda * xvec\ N|\!).(\!|\nu x|\!)P$ **and** $x \sharp \Psi$
and $x \sharp M$ **and** $x \sharp xvec$ **and** $x \sharp N$ **by** *auto*
obtain y::*name* **where** $y \sharp \Psi$ **and** $y \sharp \Psi'$ **and** $y \sharp M$ **and** $y \sharp N$ **and** $y \sharp P$ **and** $y \sharp xvec$
by(*generate-fresh name*) (*auto simp add*: *fresh-prod*)

moreover then have ($\Psi \otimes \Psi'$, $(\!|\nu y|\!)(M(\!|\lambda * xvec\ N|\!).([(x, y)] \cdot P))$, $M(\!|\lambda * xvec\ N|\!).(\!|\nu y|\!)([(x, y)] \cdot P)) \in$ *?X* **by** *force*
moreover from *Req* ‹$x \sharp M$› ‹$y \sharp M$› ‹$x \sharp xvec$› ‹$y \sharp xvec$› ‹$x \sharp N$› ‹$y \sharp N$› ‹$y \sharp P$› **have** $R = (\!|\nu y|\!)(M(\!|\lambda * xvec\ N|\!).([(x, y)] \cdot P))$
apply *clarsimp* **by**(*subst alphaRes*[*of y*]) (*auto simp add*: *eqvts inputChainFresh*)
moreover from *Qeq* ‹$y \sharp P$› **have** $Q = M(\!|\lambda * xvec\ N\ |\!).(\!|\nu y|\!)([(x, y)] \cdot P)$
apply *clarsimp* **by**(*subst alphaRes*[*of y*]) (*auto simp add*: *eqvts*)
ultimately show *?case* **by** *blast*
qed
next
case(*cSym* Ψ R Q)
then show *?case* **by** *blast*
qed
}
moreover obtain y::*name* **where** $y \sharp \Psi$ **and** $y \sharp M$ **and** $y \sharp N$ **and** $y \sharp P$ **and** $y \sharp xvec$
by(*generate-fresh name*) *auto*
ultimately have $\Psi \rhd (\!|\nu y|\!)(M(\!|\lambda * xvec\ N|\!).([(x, y)] \cdot P)) \sim M(\!|\lambda * xvec\ N|\!).(\!|\nu y|\!)([(x,$

```
y)] · P) by auto
  then show ?thesis using assms ‹y ♯ P› ‹y ♯ M› ‹y ♯ N› ‹y ♯ xvec›
    apply(subst alphaRes[where x=x and y=y and P=P], auto)
    by(subst alphaRes[where x=x and y=y and P=M⟨λ∗xvec N⟩.P]) (auto simp
add: inputChainFresh eqvts)
qed

lemma bisimCasePushRes:
  fixes x  :: name
    and Ψ  :: 'b
    and Cs :: ('c × ('a, 'b, 'c) psi) list

assumes x ♯ (map fst Cs)

shows Ψ ▷ ⟨νx⟩(Cases Cs) ∼ Cases(map (λ(φ, P). (φ, ⟨νx⟩P)) Cs)
proof −
  {
    fix x::name and Cs::('c × ('a, 'b, 'c) psi) list
    assume x ♯ Ψ and x ♯ (map fst Cs)
    let ?X1 = {(Ψ, ⟨νx⟩(Cases Cs), Cases(map (λ(φ, P). (φ, ⟨νx⟩P)) Cs)) | Ψ x
Cs. x ♯ Ψ ∧ x ♯ (map fst Cs)}
    let ?X2 = {(Ψ, Cases(map (λ(φ, P). (φ, ⟨νx⟩P)) Cs), ⟨νx⟩(Cases Cs)) | Ψ x
Cs. x ♯ Ψ ∧ x ♯ (map fst Cs)}
    let ?X = ?X1 ∪ ?X2

    have eqvt ?X
    proof
      show eqvt ?X1
        apply(clarsimp simp add: eqvt-def eqvts)
        apply (rule exI)
        apply (rule exI)
        apply (rule conjI)
         apply (rule refl)
        apply(perm-extend-simp)
          apply(clarsimp simp add: eqvts)
         apply (simp add: fresh-bij)
        apply(drule pt-fresh-bij1[OF pt-name-inst, OF at-name-inst])
        apply(drule pt-fresh-bij1[OF pt-name-inst, OF at-name-inst])
       apply(simp add: eqvts)
        apply(perm-extend-simp)
        apply(simp add: eqvts)
        done
    next
      show eqvt ?X2
        apply(clarsimp simp add: eqvt-def eqvts)
        apply (rule exI)
        apply (rule exI)
        apply (subst conj-commute)
        apply (rule conjI)
```

```
      apply (rule conjI)
       apply (rule refl)
      apply(perm-extend-simp)
       apply (simp add: fresh-bij)
      apply(drule pt-fresh-bij1[OF pt-name-inst, OF at-name-inst])
      apply(drule pt-fresh-bij1[OF pt-name-inst, OF at-name-inst])
      apply(simp add: eqvts)
      apply(perm-extend-simp)
      apply(simp add: eqvts)
     apply(perm-extend-simp)
     apply(clarsimp simp add: eqvts)
     done
  qed

  from ‹x ♯ Ψ› ‹x ♯ map fst Cs› have (Ψ, (|νx|)(Cases Cs), Cases(map (λ(φ, P).
(φ, (|νx|)P)) Cs)) ∈ ?X by auto
  then have Ψ ▷ (|νx|)(Cases Cs) ∼ Cases(map (λ(φ, P). (φ, (|νx|)P)) Cs)
  proof(coinduct rule: bisimCoinduct)
    case(cStatEq Ψ Q R)
     then show ?case using freshComp by(force intro: frameResFresh FrameS-
tatEqSym)
  next
    case(cSim Ψ Q R)
    then show ?case using ‹eqvt ?X›
      by(auto intro!: casePushResLeft casePushResRight bisimReflexive)
  next
    case(cExt Ψ Q R Ψ')
    show ?case
    proof(cases (Ψ, Q, R) ∈ ?X1)
      assume (Ψ, Q, R) ∈ ?X1
    then obtain x Cs where Qeq: Q = (|νx|)(Cases Cs) and Req: R = Cases(map
(λ(φ, P). (φ, (|νx|)P)) Cs)
        and x ♯ Ψ and x ♯ (map fst Cs) by blast
      obtain y::name where y ♯ Ψ and y ♯ Ψ' and y ♯ Cs
        by(generate-fresh name) (auto simp add: fresh-prod)
      from ‹y ♯ Cs› ‹x ♯ (map fst Cs)› have y ♯ map fst ([(x, y)] • Cs) by(induct
Cs) (auto simp add: fresh-list-cons fresh-list-nil)

      moreover with ‹y ♯ Ψ› ‹y ♯ Ψ'› have (Ψ ⊗ Ψ', (|νy|)(Cases ([(x, y)] • Cs)),
Cases(map (λ(φ, P). (φ, (|νy|)P)) ([(x, y)] • Cs))) ∈ ?X
        by auto
      moreover from Qeq ‹y ♯ Cs› have Q = (|νy|)(Cases([(x, y)] • Cs))
        apply clarsimp by(subst alphaRes[of y]) (auto simp add: eqvts)
       moreover from Req ‹y ♯ Cs› ‹x ♯ (map fst Cs)› have R = Cases(map
(λ(φ, P). (φ, (|νy|)P)) ([(x, y)] • Cs))
          by(induct Cs arbitrary: R) (auto simp add: fresh-list-cons fresh-prod
alphaRes)
      ultimately show ?case by blast
    next
```

assume $(\Psi, Q, R) \notin ?X1$
with $\langle(\Psi, Q, R) \in ?X\rangle$ **have** $(\Psi, Q, R) \in ?X2$ **by** *blast*
then obtain x Cs **where** Req: $R = (\!|\nu x|\!)(Cases\ Cs)$ **and** Qeq: $Q = Cases(map\ (\lambda(\varphi, P).\ (\varphi, (\!|\nu x|\!)P))\ Cs)$
and $x \sharp \Psi$ **and** $x \sharp (map\ fst\ Cs)$ **by** *blast*
obtain y::*name* **where** $y \sharp \Psi$ **and** $y \sharp \Psi'$ **and** $y \sharp Cs$
by(*generate-fresh name*) (*auto simp add*: *fresh-prod*)
from $\langle y \sharp Cs\rangle$ $\langle x \sharp (map\ fst\ Cs)\rangle$ **have** $y \sharp map\ fst\ ([(x, y)] \cdot Cs)$ **by**(*induct Cs*) (*auto simp add*: *fresh-list-cons fresh-list-nil*)

moreover with $\langle y \sharp \Psi\rangle$ $\langle y \sharp \Psi'\rangle$ **have** $(\Psi \otimes \Psi', (\!|\nu y|\!)(Cases\ ([(x, y)] \cdot Cs)), Cases(map\ (\lambda(\varphi, P).\ (\varphi, (\!|\nu y|\!)P))\ ([(x, y)] \cdot Cs))) \in ?X$
by *auto*
moreover from Req $\langle y \sharp Cs\rangle$ **have** $R = (\!|\nu y|\!)(Cases([(x, y)] \cdot Cs))$
apply *clarsimp* **by**(*subst alphaRes*[*of y*]) (*auto simp add*: *eqvts*)
moreover from Qeq $\langle y \sharp Cs\rangle$ $\langle x \sharp (map\ fst\ Cs)\rangle$ **have** $Q = Cases(map\ (\lambda(\varphi, P).\ (\varphi, (\!|\nu y|\!)P))\ ([(x, y)] \cdot Cs))$
by(*induct Cs arbitrary*: Q) (*auto simp add*: *fresh-list-cons fresh-prod alphaRes*)
ultimately show *?case* **by** *blast*
qed
next
case(*cSym* Ψ R Q)
then show *?case* **by** *blast*
qed
}
moreover obtain y::*name* **where** $y \sharp \Psi$ **and** $y \sharp Cs$ **by**(*generate-fresh name*) *auto*
moreover from $\langle x \sharp map\ fst\ Cs\rangle$ **have** $y \sharp map\ fst([(x, y)] \cdot Cs)$
by(*induct Cs*) (*auto simp add*: *fresh-left calc-atm*)
ultimately have $\Psi \rhd (\!|\nu y|\!)(Cases\ ([(x, y)] \cdot Cs)) \sim Cases(map\ (\lambda(\varphi, P).\ (\varphi, (\!|\nu y|\!)P))\ ([(x, y)] \cdot Cs))$
by *auto*
moreover from $\langle y \sharp Cs\rangle$ **have** $(\!|\nu y|\!)(Cases\ ([(x, y)] \cdot Cs)) = (\!|\nu x|\!)(Cases\ Cs)$
by(*simp add*: *alphaRes eqvts*)
moreover from $\langle x \sharp map\ fst\ Cs\rangle$ $\langle y \sharp Cs\rangle$ **have** $Cases(map\ (\lambda(\varphi, P).\ (\varphi, (\!|\nu y|\!)P))\ ([(x, y)] \cdot Cs)) = Cases(map\ (\lambda(\varphi, P).\ (\varphi, (\!|\nu x|\!)P))\ Cs)$
by(*induct Cs*) (*auto simp add*: *alphaRes*)
ultimately show *?thesis* **by** *auto*
qed

lemma *bangExt*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$

assumes *guarded P*

shows $\Psi \rhd !P \sim P \parallel !P$
proof −

let *?X* = {(Ψ, !*P*, *P* ∥ !*P*) | Ψ *P*. *guarded P*} ∪ {(Ψ, *P* ∥ !*P*, !*P*) | Ψ *P*. *guarded P*}
from ‹*guarded P*› **have** (Ψ, !*P*, *P* ∥ !*P*) ∈ *?X* **by** *auto*
then show *?thesis*
proof(*coinduct rule*: *bisimCoinduct*)
case(*cStatEq* Ψ *Q R*)
from ‹(Ψ, *Q*, *R*) ∈ *?X*› **obtain** *P* **where** *Eq*: (*Q* = !*P* ∧ *R* = *P* ∥ !*P*) ∨ (*Q* = *P* ∥ !*P* ∧ *R* = !*P*) **and** *guarded P*
by *auto*
obtain A_P Ψ_P **where** *FrP*: *extractFrame P* = ⟨A_P, Ψ_P⟩ **and** A_P ♯* Ψ **by**(*rule freshFrame*)
from *FrP* ‹*guarded P*› **have** $\Psi_P \simeq SBottom'$ **by**(*blast dest*: *guardedStatEq*)
from ‹$\Psi_P \simeq SBottom'$› **have** $\Psi \otimes SBottom' \simeq \Psi \otimes \Psi_P \otimes SBottom'$ **by**(*metis Identity Composition AssertionStatEqTrans Commutativity AssertionStatEqSym*)
then have ⟨A_P, $\Psi \otimes SBottom'$⟩ $\simeq_F$ ⟨A_P, $\Psi \otimes \Psi_P \otimes SBottom'$⟩
by(*force intro*: *frameResChainPres*)
moreover from ‹A_P ♯* Ψ› **have** ⟨ε, $\Psi \otimes SBottom'$⟩ $\simeq_F$ ⟨A_P, $\Psi \otimes SBottom'$⟩
apply −
by(*rule FrameStatEqSym*) (*simp add*: *frameResFreshChain freshCompChain*(*1*))
ultimately show *?case* **using** *Eq* ‹A_P ♯* Ψ› *FrP*
by *auto* (*blast dest*: *FrameStatEqTrans FrameStatEqSym*)+
next
case(*cSim* Ψ *Q R*)
then show *?case* **by**(*auto intro*: *bangExtLeft bangExtRight bisimReflexive*)
next
case(*cExt* Ψ *Q R*)
then show *?case* **by** *auto*
next
case(*cSym* Ψ *Q R*)
then show *?case* **by** *auto*
qed
qed

lemma *bisimScopeExtSym*:
fixes *x* :: *name*
and *Q* :: (′*a*, ′*b*, ′*c*) *psi*
and *P* :: (′*a*, ′*b*, ′*c*) *psi*

assumes *x* ♯ Ψ
and *x* ♯ *Q*

shows Ψ ▷ (⦇ν*x*⦈)(*P* ∥ *Q*) ∼ ((⦇ν*x*⦈)*P*) ∥ *Q*
using *assms*
by(*metis bisimScopeExt bisimTransitive bisimParComm bisimSymmetric bisimResPres*)

lemma *bisimScopeExtChainSym*:
fixes *xvec* :: *name list*
and *Q* :: (′*a*, ′*b*, ′*c*) *psi*

and P :: $('a, 'b, 'c)$ *psi*

assumes *xvec* $\sharp*$ Ψ
and *xvec* $\sharp*$ Q

shows $\Psi \rhd (\!|\nu{*}xvec|\!)(P \parallel Q) \sim ((\!|\nu{*}xvec|\!)P) \parallel Q$
using *assms*
by(*induct xvec*) (*auto intro*: *bisimScopeExtSym bisimReflexive bisimTransitive bisimResPres*)

lemma *bisimParPresAuxSym*:
fixes Ψ :: $'b$
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 $\Psi \rhd R \parallel P \sim R \parallel Q$
using *assms*
by(*metis bisimParComm bisimParPresAux bisimTransitive*)

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*) $\subseteq$ *supp* P' **and** ((*supp* T)::*name set*) $\subseteq$ *supp* Q'
proof −
from ‹$\Psi \rhd !P \longmapsto \alpha \prec P'$› **have** *guarded* P
apply −
by(*ind-cases* $\Psi \rhd !P \longmapsto \alpha \prec P'$) (*auto simp add*: *psi.inject*)
assume $\bigwedge Q'\ R\ T.\ [\![\Psi \rhd !Q \longmapsto \alpha \prec Q';\ \Psi \rhd P' \sim R \parallel !P;\ \Psi \rhd Q' \sim T \parallel !Q;\ \Psi \rhd R \sim T;\ ((supp\ R)::name\ set) \subseteq supp\ P'$;

$((supp\ T)::name\ set) \subseteq supp\ Q'\rrbracket \Longrightarrow thesis$
moreover from $\langle \Psi \rhd\ !P \longmapsto \alpha \prec P' \rangle\ \langle bn\ \alpha \sharp* subject\ \alpha \rangle\ \langle bn\ \alpha \sharp* \Psi \rangle\ \langle bn\ \alpha \sharp* P \rangle\ \langle bn\ \alpha \sharp* Q \rangle\ \langle \Psi \rhd P \sim Q \rangle\ \langle guarded\ Q \rangle$
have $\exists Q'\ T\ R\ .\ \Psi \rhd\ !Q \longmapsto \alpha \prec Q' \wedge \Psi \rhd P' \sim R \parallel\ !P \wedge \Psi \rhd Q' \sim T \parallel\ !Q \wedge \Psi \rhd R \sim T \wedge$
$((supp\ R)::name\ set) \subseteq supp\ P' \wedge ((supp\ T)::name\ set) \subseteq supp\ Q'$
proof(*nominal-induct avoiding*: *Q rule*: *bangInduct'*)
case(*cAlpha* $\alpha\ P'\ p\ Q$)
then obtain $Q'\ T\ R$ **where** *QTrans*: $\Psi \rhd\ !Q \longmapsto \alpha \prec Q'$ **and** $\Psi \rhd P' \sim R \parallel (P \parallel\ !P)$ **and** $\Psi \rhd Q' \sim T \parallel\ !Q$ **and** $\Psi \rhd R \sim T$
and *suppR*: $((supp\ R)::name\ set) \subseteq supp\ P'$ **and** *suppT*: $((supp\ T)::name\ set) \subseteq supp\ Q'$
by *blast*
from *QTrans* **have** *distinct*$(bn\ \alpha)$
by(*rule boundOutputDistinct*)
have *S*: $set\ p \subseteq set(bn\ \alpha) \times set(bn(p \cdot \alpha))$
by *fact*
from *QTrans* $\langle bn(p \cdot \alpha) \sharp* Q \rangle\ \langle bn(p \cdot \alpha) \sharp* \alpha \rangle\ \langle bn\ \alpha \sharp* subject\ \alpha \rangle\ \langle distinct(bn\ \alpha) \rangle$ **have** $bn(p \cdot \alpha) \sharp* Q'$
by(*drule-tac freeFreshChainDerivative*) *simp+*
with *QTrans* $\langle bn(p \cdot \alpha) \sharp* \alpha \rangle\ S\ \langle bn\ \alpha \sharp* subject\ \alpha \rangle$ **have** $\Psi \rhd\ !Q \longmapsto (p \cdot \alpha) \prec (p \cdot Q')$
by(*force simp add*: *residualAlpha*)
moreover from $\langle \Psi \rhd P' \sim R \parallel (P \parallel\ !P) \rangle$ **have** $(p \cdot \Psi) \rhd (p \cdot P') \sim (p \cdot (R \parallel (P \parallel\ !P)))$
by(*rule bisimClosed*)
with $\langle bn\ \alpha \sharp* \Psi \rangle\ \langle bn\ \alpha \sharp* P \rangle\ \langle bn(p \cdot \alpha) \sharp* \Psi \rangle\ \langle bn(p \cdot \alpha) \sharp* P \rangle\ S$ **have** $\Psi \rhd (p \cdot P') \sim (p \cdot R) \parallel (P \parallel\ !P)$
by(*simp add*: *eqvts*)
moreover from $\langle \Psi \rhd Q' \sim T \parallel\ !Q \rangle$ **have** $(p \cdot \Psi) \rhd (p \cdot Q') \sim (p \cdot (T \parallel\ !Q))$
by(*rule bisimClosed*)
with $\langle bn\ \alpha \sharp* \Psi \rangle\ \langle bn\ \alpha \sharp* Q \rangle\ \langle bn(p \cdot \alpha) \sharp* \Psi \rangle\ \langle bn(p \cdot \alpha) \sharp* Q \rangle\ S$ **have** $\Psi \rhd (p \cdot Q') \sim (p \cdot T) \parallel\ !Q$
by(*simp add*: *eqvts*)
moreover from $\langle \Psi \rhd R \sim T \rangle$ **have** $(p \cdot \Psi) \rhd (p \cdot R) \sim (p \cdot T)$
by(*rule bisimClosed*)
with $\langle bn\ \alpha \sharp* \Psi \rangle\ \langle bn(p \cdot \alpha) \sharp* \Psi \rangle\ S$ **have** $\Psi \rhd (p \cdot R) \sim (p \cdot T)$
by(*simp add*: *eqvts*)
moreover from *suppR* **have** $((supp(p \cdot R))::name\ set) \subseteq supp(p \cdot P')$
apply(*erule-tac rev-mp*)
by(*subst subsetClosed*[*of p, symmetric*]) (*simp add*: *eqvts*)
moreover from *suppT* **have** $((supp(p \cdot T))::name\ set) \subseteq supp(p \cdot Q')$
apply(*erule-tac rev-mp*)
by(*subst subsetClosed*[*of p, symmetric*]) (*simp add*: *eqvts*)
ultimately show *?case* **by** *blast*
next
case(*cPar1* $\alpha\ P'\ Q$)
from $\langle \Psi \rhd P \sim Q \rangle\ \langle \Psi \rhd P \longmapsto \alpha \prec P' \rangle\ \langle bn\ \alpha \sharp* \Psi \rangle\ \langle bn\ \alpha \sharp* Q \rangle$
obtain Q' **where** *QTrans*: $\Psi \rhd Q \longmapsto \alpha \prec Q'$ **and** $\Psi \rhd P' \sim Q'$

by(*blast dest*: *bisimE simE*)
from *QTrans* **have** $\Psi \otimes SBottom' \rhd Q \longmapsto \alpha \prec Q'$
by(*metis statEqTransition Identity AssertionStatEqSym*)
then have $\Psi \rhd Q \parallel !Q \longmapsto \alpha \prec (Q' \parallel !Q)$
using ‹*bn* $\alpha \sharp* Q$› **by**(*intro Par1*) (*assumption* | *simp*)+
then have $\Psi \rhd !Q \longmapsto \alpha \prec (Q' \parallel !Q)$
using ‹*guarded* Q› **by**(*rule Bang*)
moreover from ‹*guarded* P› **have** $\Psi \rhd P' \parallel !P \sim P' \parallel (P \parallel !P)$
by(*metis bangExt bisimParPresSym*)
moreover have $\Psi \rhd Q' \parallel !Q \sim Q' \parallel !Q$
by(*rule bisimReflexive*)
ultimately show *?case* **using** ‹$\Psi \rhd P' \sim Q'$›
by(*force simp add*: *psi.supp*)
next
case(*cPar2* α P' Q)
then obtain Q' T R **where** *QTrans*: $\Psi \rhd !Q \longmapsto \alpha \prec Q'$ **and** $\Psi \rhd P' \sim R \parallel !P$ **and** $\Psi \rhd Q' \sim T \parallel !Q$
and $\Psi \rhd R \sim T$ **and** *suppR*: ((*supp* R)::*name set*) $\subseteq$ *supp* P' **and** *suppT*: ((*supp* T)::*name set*) $\subseteq$ *supp* Q'
by *blast*
note *QTrans*
from ‹$\Psi \rhd P' \sim R \parallel !P$› **have** $\Psi \rhd P \parallel P' \sim R \parallel (P \parallel !P)$
by(*metis bisimParPresSym bisimParComm bisimTransitive bisimParAssoc*)
with *QTrans* **show** *?case* **using** ‹$\Psi \rhd Q' \sim T \parallel !Q$› ‹$\Psi \rhd R \sim T$› *suppR* *suppT*
by(*force simp add*: *psi.supp*)
next
case(*cComm1 M N P' K xvec P'' Q*)
from ‹$\Psi \rhd P \sim Q$› **have** $\Psi \rhd Q \rightsquigarrow[bisim] P$
by(*metis bisimE*)
with ‹$\Psi \rhd P \longmapsto M(\!|N|\!) \prec P'$› **obtain** Q' **where** *QTrans*: $\Psi \rhd Q \longmapsto M(\!|N|\!) \prec Q'$ **and** $\Psi \rhd Q' \sim P'$
by(*force dest*: *simE*)
from *QTrans* **have** $\Psi \otimes SBottom' \rhd Q \longmapsto M(\!|N|\!) \prec Q'$
by(*metis statEqTransition Identity AssertionStatEqSym*)
moreover obtain A_Q Ψ_Q **where** *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$ **and** $A_Q \sharp* \Psi$ **and** $A_Q \sharp* Q$ **and** $A_Q \sharp* M$
by(*rule freshFrame*[**where** C=(Ψ, Q, M)]) *auto*
note *FrQ*
moreover from *FrQ* ‹*guarded* Q› **have** $\Psi_Q \simeq SBottom'$
by(*blast dest*: *guardedStatEq*)
from ‹$\Psi \rhd !P \longmapsto K(\!|\nu*xvec|\!)\langle N\rangle \prec P''$› ‹*xvec* $\sharp*$ K› ‹$\Psi \rhd P \sim Q$› ‹*xvec* $\sharp*$ Ψ› ‹*xvec* $\sharp*$ P› ‹*xvec* $\sharp*$ Q› ‹*guarded* Q›
obtain Q'' T R **where** *QTrans'*: $\Psi \rhd !Q \longmapsto K(\!|\nu*xvec|\!)\langle N\rangle \prec Q''$ **and** $\Psi \rhd P'' \sim R \parallel !P$ **and** $\Psi \rhd Q'' \sim T \parallel !Q$ **and** $\Psi \rhd R \sim T$
and *suppR*: ((*supp* R)::*name set*) $\subseteq$ *supp* P'' **and** *suppT*: ((*supp* T)::*name set*) $\subseteq$ *supp* Q''
using *cComm1* **by** *force*
from *QTrans'* ‹$\Psi_Q \simeq SBottom'$› **have** $\Psi \otimes \Psi_Q \rhd !Q \longmapsto K(\!|\nu*xvec|\!)\langle N\rangle \prec Q''$

by(*metis statEqTransition Identity compositionSym AssertionStatEqSym*)
moreover from ‹$\Psi \vdash M \leftrightarrow K$› ‹$\Psi_Q \simeq SBottom'$› **have** $\Psi \otimes \Psi_Q \otimes SBottom' \vdash M \leftrightarrow K$
by(*metis statEqEnt Identity compositionSym AssertionStatEqSym*)
ultimately have $\Psi \rhd Q \parallel !Q \longmapsto \tau \prec (\langle\!|\nu * xvec|\!\rangle(Q' \parallel Q''))$ **using** ‹$A_Q \sharp* \Psi$› ‹$A_Q \sharp* Q$› ‹$A_Q \sharp* M$› ‹*xvec* $\sharp*$ Q›
by(*intro Comm1*) (*assumption* | *simp*)+
then have $\Psi \rhd !Q \longmapsto \tau \prec (\langle\!|\nu * xvec|\!\rangle(Q' \parallel Q''))$
using ‹*guarded Q*› **by**(*rule Bang*)
moreover from ‹$\Psi \rhd P'' \sim R \parallel !P$› ‹*guarded P*› ‹*xvec* $\sharp*$ Ψ› **have** $\Psi \rhd \langle\!|\nu * xvec|\!\rangle(P' \parallel P'') \sim \langle\!|\nu * xvec|\!\rangle((P' \parallel R) \parallel (P \parallel !P))$
by(*metis bisimParPresSym bangExt bisimTransitive bisimParAssoc bisimSymmetric bisimResChainPres*)
with ‹*xvec* $\sharp*$ Ψ› ‹*xvec* $\sharp*$ P› **have** $\Psi \rhd (\langle\!|\nu * xvec|\!\rangle(P' \parallel P'') \sim (\langle\!|\nu * xvec|\!\rangle(P' \parallel R)) \parallel (P \parallel !P)$
by(*metis bisimScopeExtChainSym bisimTransitive psiFreshVec*)
moreover from ‹$\Psi \rhd Q'' \sim T \parallel !Q$› ‹*xvec* $\sharp*$ Ψ› ‹*xvec* $\sharp*$ Q› **have** $\Psi \rhd \langle\!|\nu * xvec|\!\rangle(Q' \parallel Q'') \sim (\langle\!|\nu * xvec|\!\rangle(Q' \parallel T)) \parallel !Q$
by(*metis bisimParPresSym bisimTransitive bisimParAssoc bisimSymmetric bisimResChainPres bisimScopeExtChainSym psiFreshVec*)
moreover from ‹$\Psi \rhd R \sim T$› ‹$\Psi \rhd Q' \sim P'$› ‹*xvec* $\sharp*$ Ψ› **have** $\Psi \rhd \langle\!|\nu * xvec|\!\rangle(P' \parallel R) \sim \langle\!|\nu * xvec|\!\rangle(Q' \parallel T)$
by(*metis bisimParPresSym bisimTransitive bisimResChainPres bisimParComm bisimE*(*4*))
moreover from *suppR* **have** $((supp(\langle\!|\nu * xvec|\!\rangle(P' \parallel R)))::name\ set) \subseteq supp((\langle\!|\nu * xvec|\!\rangle(P' \parallel P')))$
by(*auto simp add*: *psi.supp resChainSupp*)
moreover from *suppT* **have** $((supp(\langle\!|\nu * xvec|\!\rangle(Q' \parallel T)))::name\ set) \subseteq supp((\langle\!|\nu * xvec|\!\rangle(Q' \parallel Q')))$
by(*auto simp add*: *psi.supp resChainSupp*)
ultimately show *?case* **by** *blast*
next
case(*cComm2 M xvec N P' K P'' Q*)
from ‹$\Psi \rhd P \sim Q$› ‹$\Psi \rhd P \longmapsto M\langle\!|\nu * xvec|\!\rangle\langle N\rangle \prec P'$› ‹*xvec* $\sharp*$ Ψ› ‹*xvec* $\sharp*$ Q›
obtain Q' **where** *QTrans*: $\Psi \rhd Q \longmapsto M\langle\!|\nu * xvec|\!\rangle\langle N\rangle \prec Q'$ **and** $\Psi \rhd P' \sim Q'$
by(*metis bisimE simE bn.simps*)
from *QTrans* **have** $\Psi \otimes SBottom' \rhd Q \longmapsto M\langle\!|\nu * xvec|\!\rangle\langle N\rangle \prec Q'$
by(*metis statEqTransition Identity AssertionStatEqSym*)
moreover obtain A_Q Ψ_Q **where** *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$ **and** $A_Q \sharp* \Psi$ **and** $A_Q \sharp* Q$ **and** $A_Q \sharp* M$
by(*rule freshFrame*[**where** C=(Ψ, Q, M)]) *auto*
note *FrQ*
moreover from *FrQ* ‹*guarded Q*› **have** $\Psi_Q \simeq SBottom'$
by(*blast dest*: *guardedStatEq*)
from ‹$\Psi \rhd !P \longmapsto K(\!|N|\!) \prec P''$› ‹$\Psi \rhd P \sim Q$› ‹*guarded Q*›
obtain Q'' T R **where** *QTrans'*: $\Psi \rhd !Q \longmapsto K(\!|N|\!) \prec Q''$ **and** $\Psi \rhd P'' \sim R \parallel !P$ **and** $\Psi \rhd Q'' \sim T \parallel !Q$ **and** $\Psi \rhd R \sim T$
and *suppR*: ((*supp R*)::*name set*) $\subseteq$ *supp* P'' **and** *suppT*: ((*supp T*)::*name set*) $\subseteq$ *supp* Q'' **using** *cComm2*

by *force*
from *QTrans'* ‹$\Psi_Q \simeq SBottom'$› **have** $\Psi \otimes \Psi_Q \rhd !Q \longmapsto K(\!|N|\!) \prec Q''$
by(*metis statEqTransition Identity compositionSym AssertionStatEqSym*)
moreover from ‹$\Psi \vdash M \leftrightarrow K$› ‹$\Psi_Q \simeq SBottom'$› **have** $\Psi \otimes \Psi_Q \otimes SBottom' \vdash M \leftrightarrow K$
by(*metis statEqEnt Identity compositionSym AssertionStatEqSym*)
ultimately have $\Psi \rhd Q \parallel !Q \longmapsto \tau \prec ((\!|\nu * xvec|\!)(Q' \parallel Q''))$ **using** ‹$A_Q \sharp* \Psi$› ‹$A_Q \sharp* Q$› ‹$A_Q \sharp* M$› ‹$xvec \sharp* Q$›
by(*intro Comm2*) (*assumption* | *simp*)+
then have $\Psi \rhd !Q \longmapsto \tau \prec ((\!|\nu * xvec|\!)(Q' \parallel Q''))$ **using** ‹*guarded Q*› **by**(*rule Bang*)
moreover from ‹$\Psi \rhd P'' \sim R \parallel !P$› ‹*guarded P*› ‹$xvec \sharp* \Psi$› **have** $\Psi \rhd (\!|\nu * xvec|\!)(P' \parallel P'') \sim (\!|\nu * xvec|\!)((P' \parallel R) \parallel (P \parallel !P))$
by(*metis bisimParPresSym bangExt bisimTransitive bisimParAssoc bisimSymmetric bisimResChainPres*)
with ‹$xvec \sharp* \Psi$› ‹$xvec \sharp* P$› **have** $\Psi \rhd (\!|\nu * xvec|\!)(P' \parallel P'') \sim ((\!|\nu * xvec|\!)(P' \parallel R)) \parallel (P \parallel !P)$
by(*metis bisimScopeExtChainSym bisimTransitive psiFreshVec*)
moreover from ‹$\Psi \rhd Q'' \sim T \parallel !Q$› ‹$xvec \sharp* \Psi$› ‹$xvec \sharp* Q$› **have** $\Psi \rhd (\!|\nu * xvec|\!)(Q' \parallel Q'') \sim ((\!|\nu * xvec|\!)(Q' \parallel T)) \parallel !Q$
by(*metis bisimParPresSym bisimTransitive bisimParAssoc bisimSymmetric bisimResChainPres bisimScopeExtChainSym psiFreshVec*)
moreover from ‹$\Psi \rhd R \sim T$› ‹$\Psi \rhd P' \sim Q'$› ‹$xvec \sharp* \Psi$› **have** $\Psi \rhd (\!|\nu * xvec|\!)(P' \parallel R) \sim (\!|\nu * xvec|\!)(Q' \parallel T)$
by(*metis bisimParPresSym bisimTransitive bisimResChainPres bisimParComm*)
moreover from *suppR* **have** $((supp((\!|\nu * xvec|\!)(P' \parallel R)))::name\ set) \subseteq supp((\!|\nu * xvec|\!)(P' \parallel P''))$
by(*auto simp add*: *psi.supp resChainSupp*)
moreover from *suppT* **have** $((supp((\!|\nu * xvec|\!)(Q' \parallel T)))::name\ set) \subseteq supp((\!|\nu * xvec|\!)(Q' \parallel Q''))$
by(*auto simp add*: *psi.supp resChainSupp*)
ultimately show *?case* **by** *blast*
next
case(*cBang* α P' Q)
then obtain Q' T R **where** *QTrans*: $\Psi \rhd !Q \longmapsto \alpha \prec Q'$ **and** $\Psi \rhd P' \sim R \parallel (P \parallel !P)$ **and** $\Psi \rhd Q' \sim T \parallel !Q$ **and** $\Psi \rhd R \sim T$
and *suppR*: $((supp\ R)::name\ set) \subseteq supp\ P'$ **and** *suppT*: $((supp\ T)::name\ set) \subseteq supp\ Q'$
by *blast*
from ‹$\Psi \rhd P' \sim R \parallel (P \parallel !P)$› ‹*guarded P*› **have** $\Psi \rhd P' \sim R \parallel !P$
by(*metis bangExt bisimParPresSym bisimTransitive bisimSymmetric*)
with *QTrans* **show** *?case* **using** ‹$\Psi \rhd Q' \sim T \parallel !Q$› ‹$\Psi \rhd R \sim T$› *suppR suppT*
by *blast*
next
case(*cBrMerge M N P' P'' Q*)
then obtain Q' T R **where** $\Psi \rhd !Q \longmapsto ¿M(\!|N|\!) \prec Q'$ **and**
$p''eq$: $\Psi \rhd P'' \sim R \parallel !P$ **and**

$\Psi \rhd Q' \sim T \parallel !Q$ **and** $\Psi \rhd R \sim T$ **and**
supp $R \subseteq$ (*supp* P''::*name set*) **and**
supp $T \subseteq$ (*supp* Q'::*name set*)
by *blast*
obtain Q'' **where** $\Psi \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q''$ **and** $\Psi \rhd Q'' \sim P'$
proof −
assume *g*: $(\bigwedge Q''.\ [\![\Psi \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q''; \Psi \rhd Q'' \sim P']\!] \Longrightarrow thesis)$
have $\exists\, Q'.\ \Psi \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q' \wedge (\Psi, Q', P') \in bisim$
apply(*rule simE*)
apply(*rule bisimE*)
apply(*rule bisimSymmetric*)
by *fact*+
then show *thesis*
using *g* **by** *blast*
qed
have $\Psi \otimes \mathbf{1} \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q''$ **using** ‹$\Psi \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q''$›
by(*rule statEqTransition*) (*rule AssertionStatEqSym*[*OF Identity*])
obtain $A_Q\ \Psi_Q$ **where** *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$ **and** *distinct* A_Q **and** $A_Q \sharp* \Psi$ **and** $A_Q \sharp* Q$ **and** $A_Q \sharp* M$
by(*rule freshFrame*[**where** C=(Ψ,Q,M)]) *force*
then have $\Psi_Q \simeq \mathbf{1}$ **using** ‹*guarded Q*›
by(*auto dest*: *guardedStatEq*)
have $\Psi \otimes \Psi_Q \rhd !Q \longmapsto ¿M(\!|N|\!) \prec Q'$ **using** ‹$\Psi \rhd !Q \longmapsto ¿M(\!|N|\!) \prec Q'$›
by(*rule statEqTransition*) (*metis* ‹$\Psi_Q \simeq \mathbf{1}$› *AssertionStatEqTrans AssertionStatEqSym Identity compositionSym*)
have $\Psi \rhd !Q \longmapsto ¿M(\!|N|\!) \prec Q'' \parallel Q'$
by(*rule Bang*) (*rule BrMerge*|*fact*|*simp*)+
moreover have $\Psi \rhd P' \parallel P'' \sim (P' \parallel R) \parallel (P \parallel !P)$
apply(*rule bisimTransitive*[*OF bisimParPresSym*, *OF p''eq*])
apply(*rule bisimTransitive*[*OF bisimSymmetric*, *OF bisimParAssoc*])
apply(*rule bisimParPresSym*)
apply(*rule bangExt*[*OF* ‹*guarded P*›])
done
moreover have $\Psi \rhd Q'' \parallel Q' \sim Q'' \parallel T \parallel !Q$ **using** ‹$\Psi \rhd Q' \sim T \parallel !Q$›
by(*metis bisimTransitive bisimParAssoc bisimParComm bisimParPres bisimSymmetric*)
moreover have $\Psi \rhd (P' \parallel R) \sim Q'' \parallel T$ **using** ‹$\Psi \rhd R \sim T$› **and** ‹$\Psi \rhd Q'' \sim P'$›
apply −
apply(*erule bisimTransitive*[*OF bisimParPres*, *OF bisimSymmetric*])
apply(*erule bisimTransitive*[*OF bisimParPresSym*])
by(*rule bisimReflexive*)
moreover have *supp*($P' \parallel R$) $\subseteq$ (*supp*($P' \parallel P''$)::*name set*)
using ‹*supp* $R \subseteq$ (*supp* P''::*name set*)› **by**(*auto simp add*: *psi.supp*)
moreover have *supp*($Q'' \parallel T$) $\subseteq$ (*supp*($Q'' \parallel Q'$)::*name set*)
using ‹*supp* $T \subseteq$ (*supp* Q'::*name set*)› **by**(*auto simp add*: *psi.supp*)
ultimately show *?case*
by *blast*
next

case(*cBrComm1 M N P′ xvec P″ Q*)
from ‹$\Psi \rhd P \sim Q$› **have** $\Psi \rhd Q \rightsquigarrow[bisim]\ P$ **by**(*metis bisimE*)
with ‹$\Psi \rhd P \longmapsto ¡M(\!|N|\!) \prec P'$› **obtain** Q' **where** *QTrans*: $\Psi \rhd Q \longmapsto ¡M(\!|N|\!) \prec Q'$ **and** $\Psi \rhd Q' \sim P'$
by(*force dest*: *simE*)
from *QTrans* **have** $\Psi \otimes SBottom' \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$
by(*metis statEqTransition Identity AssertionStatEqSym*)
moreover obtain $A_Q\ \Psi_Q$ **where** *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$ **and** $A_Q \sharp* \Psi$ **and** $A_Q \sharp* Q$ **and** $A_Q \sharp* M$
by(*rule freshFrame*[**where** $C=(\Psi, Q, M)$]) *auto*
note *FrQ*
moreover from *FrQ* ‹*guarded Q*› **have** $\Psi_Q \simeq SBottom'$
by(*blast dest*: *guardedStatEq*)
from ‹$\Psi \rhd !P \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec P''$› ‹$\Psi \rhd P \sim Q$› ‹*xvec* $\sharp*$ Ψ› ‹*xvec* $\sharp*$ *P*› ‹*xvec* $\sharp*$ *Q*› ‹*guarded Q*›
obtain $Q''\ T\ R$ **where** *QTrans′*: $\Psi \rhd !Q \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec Q''$ **and** $\Psi \rhd P'' \sim R \parallel !P$ **and** $\Psi \rhd Q'' \sim T \parallel !Q$
and $\Psi \rhd R \sim T$ **and** *suppR*: $((supp\ R)::name\ set) \subseteq supp\ P''$ **and** *suppT*: $((supp\ T)::name\ set) \subseteq supp\ Q''$
using *cBrComm1* **by** *force*
from *QTrans′* ‹$\Psi_Q \simeq SBottom'$› **have** $\Psi \otimes \Psi_Q \rhd !Q \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec Q''$
by(*metis statEqTransition Identity compositionSym AssertionStatEqSym*)
ultimately have $\Psi \rhd Q \parallel !Q \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec Q' \parallel Q''$
using ‹$A_Q \sharp* \Psi$› ‹$A_Q \sharp* Q$› ‹$A_Q \sharp* M$› ‹*xvec* $\sharp*$ *Q*› **by**(*intro BrComm1*) (*assumption* | *simp*)+
then have $\Psi \rhd !Q \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec Q' \parallel Q''$
using ‹*guarded Q*› **by**(*rule Bang*)
moreover from ‹$\Psi \rhd P'' \sim R \parallel !P$› ‹*guarded P*› ‹*xvec* $\sharp*$ Ψ› **have** $\Psi \rhd P' \parallel P'' \sim (P' \parallel R) \parallel (P \parallel !P)$
by(*metis bisimParPresSym bangExt bisimTransitive bisimParAssoc bisimSymmetric*)
moreover from ‹$\Psi \rhd Q'' \sim T \parallel !Q$› ‹*xvec* $\sharp*$ Ψ› ‹*xvec* $\sharp*$ *Q*› **have** $\Psi \rhd Q' \parallel Q'' \sim (Q' \parallel T) \parallel !Q$
by(*metis bisimParPresSym bisimTransitive bisimParAssoc bisimSymmetric*)
moreover from ‹$\Psi \rhd R \sim T$› ‹$\Psi \rhd Q' \sim P'$› ‹*xvec* $\sharp*$ Ψ› **have** $\Psi \rhd P' \parallel R \sim Q' \parallel T$
by(*metis bisimParPresSym bisimTransitive bisimParComm bisimE*(*4*))
moreover from *suppR* **have** $((supp(P' \parallel R))::name\ set) \subseteq supp((P' \parallel P''))$
by(*auto simp add*: *psi.supp resChainSupp*)
moreover from *suppT* **have** $((supp(Q' \parallel T))::name\ set) \subseteq supp((Q' \parallel Q''))$
by(*auto simp add*: *psi.supp resChainSupp*)
ultimately show *?case* **by** *blast*
next
case(*cBrComm2 M N P′ xvec P″ Q*)
from ‹$\Psi \rhd P \sim Q$› **have** $\Psi \rhd Q \rightsquigarrow[bisim]\ P$ **by**(*metis bisimE*)
with ‹$\Psi \rhd P \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec P'$› **obtain** Q' **where** *QTrans*: $\Psi \rhd Q \longmapsto ¡M(\!|\nu*xvec|\!)\langle N\rangle \prec Q'$ **and** $\Psi \rhd Q' \sim P'$
using ‹*xvec* $\sharp*$ Ψ› ‹*xvec* $\sharp*$ *P*› ‹*xvec* $\sharp*$ *Q*› **by**(*auto dest*: *simE*)

from *QTrans* **have** $\Psi \otimes SBottom' \rhd Q \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec Q'$
by(*metis statEqTransition Identity AssertionStatEqSym*)
moreover obtain A_Q Ψ_Q **where** *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$ **and** $A_Q \sharp* \Psi$ **and** $A_Q \sharp* Q$ **and** $A_Q \sharp* M$
by(*rule freshFrame*[**where** $C=(\Psi, Q, M)$]) *auto*
note *FrQ*
moreover from *FrQ* ‹*guarded Q*› **have** $\Psi_Q \simeq SBottom'$ **by**(*blast dest*: *guardedStatEq*)
from ‹$\Psi \rhd !P \longmapsto ¿M(\!|N|\!) \prec P''$› ‹$\Psi \rhd P \sim Q$› ‹*xvec* $\sharp*$ Ψ› ‹*xvec* $\sharp*$ *P*› ‹*xvec* $\sharp*$ *Q*› ‹*guarded Q*›
obtain Q'' *T R* **where** *QTrans'*: $\Psi \rhd !Q \longmapsto ¿M(\!|N|\!) \prec Q''$ **and** $\Psi \rhd P'' \sim R \parallel !P$ **and** $\Psi \rhd Q'' \sim T \parallel !Q$
and $\Psi \rhd R \sim T$ **and** *suppR*: ((*supp R*)::*name set*) $\subseteq$ *supp* P'' **and** *suppT*: ((*supp T*)::*name set*) $\subseteq$ *supp* Q''
using *cBrComm2* **by** *force*
from *QTrans'* ‹$\Psi_Q \simeq SBottom'$› **have** $\Psi \otimes \Psi_Q \rhd !Q \longmapsto ¿M(\!|N|\!) \prec Q''$
by(*metis statEqTransition Identity compositionSym AssertionStatEqSym*)
ultimately have $\Psi \rhd Q \parallel !Q \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec Q' \parallel Q''$
using ‹$A_Q \sharp* \Psi$› ‹$A_Q \sharp* Q$› ‹$A_Q \sharp* M$› ‹*xvec* $\sharp*$ *Q*› **by**(*intro BrComm2*) (*assumption* | *simp*)+
then have $\Psi \rhd !Q \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec Q' \parallel Q''$
using ‹*guarded Q*› **by**(*rule Bang*)
moreover from ‹$\Psi \rhd P'' \sim R \parallel !P$› ‹*guarded P*› ‹*xvec* $\sharp*$ Ψ› **have** $\Psi \rhd P' \parallel P'' \sim (P' \parallel R) \parallel (P \parallel !P)$
by(*metis bisimParPresSym bangExt bisimTransitive bisimParAssoc bisimSymmetric*)
moreover from ‹$\Psi \rhd Q'' \sim T \parallel !Q$› ‹*xvec* $\sharp*$ Ψ› ‹*xvec* $\sharp*$ *Q*› **have** $\Psi \rhd Q' \parallel Q'' \sim (Q' \parallel T) \parallel !Q$
by(*metis bisimParPresSym bisimTransitive bisimParAssoc bisimSymmetric*)
moreover from ‹$\Psi \rhd R \sim T$› ‹$\Psi \rhd Q' \sim P'$› ‹*xvec* $\sharp*$ Ψ› **have** $\Psi \rhd P' \parallel R \sim Q' \parallel T$
by(*metis bisimParPresSym bisimTransitive bisimParComm bisimE*(*4*))
moreover from *suppR* **have** ((*supp*$(P' \parallel R)$)::*name set*) $\subseteq$ *supp*$((P' \parallel P''))$
by(*auto simp add*: *psi.supp resChainSupp*)
moreover from *suppT* **have** ((*supp*$(Q' \parallel T)$)::*name set*) $\subseteq$ *supp*$((Q' \parallel Q''))$
by(*auto simp add*: *psi.supp resChainSupp*)
ultimately show *?case* **by** *blast*
qed
ultimately show *?thesis* **by** *blast*
qed

lemma *structCongBisim*:
fixes P :: ('*a*, '*b*, '*c*) *psi*
and Q :: ('*a*, '*b*, '*c*) *psi*

assumes $P \equiv_s Q$

shows $P \sim Q$
using *assms*

by(*induct rule*: *structCong.induct*)
(*auto intro*: *bisimReflexive bisimSymmetric bisimTransitive bisimParComm bisimParAssoc bisimParNil bisimResNil bisimResComm bisimScopeExt bisimCasePushRes bisimInputPushRes bisimOutputPushRes bangExt*)

lemma *bisimBangPres*:
fixes $\Psi :: 'b$
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 −
let *?X* = $\{(\Psi, R \parallel\ !P, R \parallel\ !Q) \mid \Psi\ P\ Q\ R.\ \Psi \rhd P \sim Q \wedge$ *guarded P* $\wedge$ *guarded Q*$\}$
let *?Y* = $\{(\Psi, P, Q) \mid \Psi\ P\ P'\ Q'\ Q.\ \Psi \rhd P \sim P' \wedge (\Psi, P', Q') \in$ *?X* $\wedge\ \Psi \rhd Q' \sim Q\}$
from *assms* **have** $(\Psi, \mathbf{0} \parallel\ !P, \mathbf{0} \parallel\ !Q) \in$ *?X* **by**(*blast intro*: *bisimReflexive*)

moreover have *eqvt ?X*
apply(*clarsimp simp add*: *eqvt-def*)
apply(*drule bisimClosed*)
by *force*
ultimately have $\Psi \rhd \mathbf{0} \parallel\ !P \sim \mathbf{0} \parallel\ !Q$
proof(*coinduct rule*: *weakTransitiveCoinduct*)
case(*cStatEq* $\Psi\ P\ Q$)
then show *?case* **by** *auto*
next
case(*cSim* $\Psi\ RP\ RQ$)
from ‹$(\Psi, RP, RQ) \in$ *?X*› **obtain** $P\ Q\ R$ **where** $\Psi \rhd P \sim Q$ **and** *guarded P* **and** *guarded Q*
and $RP = R \parallel\ !P$ **and** $RQ = R \parallel\ !Q$
by *auto*
note ‹$\Psi \rhd P \sim Q$›
moreover from ‹*eqvt ?X*› **have** *eqvt ?Y* **by** *blast*
moreover note ‹*guarded P*› ‹*guarded Q*› *bisimE*(*2*) *bisimE*(*3*) *bisimE*(*4*) *statEqBisim bisimClosed bisimParAssoc*[*THEN bisimSymmetric*]
bisimParPres bisimParPresAuxSym bisimResChainPres bisimScopeExtChainSym bisimTransitive
moreover have $\bigwedge\Psi\ P\ Q\ R\ T.\ [\![\Psi \rhd P \sim Q; (\Psi, Q, R) \in$ *?Y*; $\Psi \rhd R \sim T]\!] \Longrightarrow (\Psi, P, T) \in$ *?Y*
by *auto* (*metis bisimTransitive*)
moreover have $\bigwedge\Psi\ P\ Q\ R.\ [\![\Psi \rhd P \sim Q;$ *guarded P*; *guarded Q*$]\!] \Longrightarrow (\Psi, R \parallel\ !P, R \parallel\ !Q) \in$ *?Y* **by**(*blast intro*: *bisimReflexive*)
moreover have $\bigwedge\Psi\ P\ \alpha\ P'\ Q.\ [\![\Psi \rhd\ !P \longmapsto \alpha \prec P'; \Psi \rhd P \sim Q;$ *bn* $\alpha \sharp* \Psi$; *bn* $\alpha \sharp* P$; *bn* $\alpha \sharp* Q$; *guarded Q*; *bn* $\alpha \sharp*$ *subject* $\alpha]\!] \Longrightarrow$

$\exists\, Q'\ R\ T.\ \ \Psi \rhd\, !Q \longmapsto \alpha \prec\ Q' \wedge \Psi \rhd\ P' \sim\ R \parallel\ !P \wedge\ \ \Psi \rhd\ Q' \sim\ T \parallel\ !Q \wedge$
$\Psi \rhd\ R \sim\ T \wedge ((supp\ R)::name\ set) \subseteq\ supp\ P' \wedge$
$((supp\ T)::name\ set) \subseteq\ supp\ Q'$

by(*blast elim*: *bangDerivative*)
ultimately have $\Psi \rhd\ R \parallel\ !P \rightsquigarrow [?Y]\ R \parallel\ !Q$
by(*rule bangPres*)
with ‹$RP = R \parallel\ !P$› ‹$RQ = R \parallel\ !Q$› **show** *?case*
by *blast*
next
case(*cExt* Ψ *RP RQ* Ψ')
then show *?case* **by**(*blast dest*: *bisimE*)
next
case(*cSym* Ψ *RP RQ*)
then show *?case* **by**(*blast dest*: *bisimE*)
qed
then show *?thesis*
by(*metis bisimTransitive bisimParNil bisimSymmetric bisimParComm*)
qed

end

end
theory *Bisim-Subst*
imports *Bisim-Struct-Cong Psi-Calculi.Close-Subst*
begin

This file is a (heavily modified) variant of the theory *Psi_Calculi.Bisim_Subst* from [1].

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 Ψ :: *'b*
and *P* :: (*'a*, *'b*, *'c*) *psi*
and *Q* :: (*'a*, *'b*, *'c*) *psi*
and *M* :: *'a*
and *N* :: *'a*

assumes $\Psi \rhd P \sim_s Q$

shows $\Psi \rhd M\langle N\rangle.P \sim_s M\langle N\rangle.Q$
using *assms closeSubstI closeSubstE bisimOutputPres* **by** *force*

lemma *seqSubstInputChain*[*simp*]:
fixes *xvec* :: *name list*
and N :: $'a$
and P :: $('a, 'b, 'c)$ *psi*
and σ :: (*name list* × $'a$ *list*) *list*

assumes *xvec* $\sharp*$ σ

shows *seqSubs'* (*inputChain xvec N P*) σ = *inputChain xvec* (*substTerm.seqSubst N* σ) (*seqSubs P* σ)
using *assms*
by(*induct xvec*) *auto*

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* $\sharp*$ Ψ
and *distinct xvec*

shows $\Psi \rhd M(\!|\lambda*xvec\ N|\!).P \sim_s M(\!|\lambda*xvec\ N|\!).Q$
proof(*rule closeSubstI*)
fix σ
assume *wellFormedSubst*(σ::(*name list* × $'a$ *list*) *list*)
obtain p **where** $(p \cdot xvec) \sharp* \sigma$
and $(p \cdot xvec) \sharp* P$ **and** $(p \cdot xvec) \sharp* Q$ **and** $(p \cdot xvec) \sharp* \Psi$ **and** $(p \cdot xvec) \sharp* N$
and S: *set* $p \subseteq$ *set xvec* × *set* $(p \cdot xvec)$
by(*rule name-list-avoiding*[**where** c=(σ, P, Q, Ψ, N)]) *auto*

from ‹$\Psi \rhd P \sim_s Q$› **have** $(p \cdot \Psi) \rhd (p \cdot P) \sim_s (p \cdot Q)$
by(*rule bisimSubstClosed*)
with ‹*xvec* $\sharp*$ Ψ› ‹$(p \cdot xvec) \sharp* \Psi$› S **have** $\Psi \rhd (p \cdot P) \sim_s (p \cdot Q)$
by *simp*

{
fix *Tvec* :: $'a$ *list*
from ‹$\Psi \rhd (p \cdot P) \sim_s (p \cdot Q)$› ‹*wellFormedSubst* σ› **have** $\Psi \rhd (p \cdot P)[<\sigma>]$

$\sim_s$ $(p \cdot Q)[<\sigma>]$
by(*rule closeSubstUnfold*)
moreover assume *length xvec* = *length Tvec* **and** *distinct xvec*
ultimately have $\Psi \rhd ((p \cdot P)[<\sigma>])[(p \cdot xvec)::=Tvec] \sim ((p \cdot Q)[<\sigma>])[(p \cdot xvec)::=Tvec]$
apply −
by(*drule closeSubstE*[**where** σ=[((p · xvec), Tvec)]]) *auto*
}

with ‹(p · xvec) ♯* σ› ‹*distinct xvec*›
have $\Psi \rhd (M(\!|\lambda*(p \cdot xvec)\ (p \cdot N)|\!).(p \cdot P))[<\sigma>] \sim (M(\!|\lambda*(p \cdot xvec)\ (p \cdot N)|\!).(p \cdot Q))[<\sigma>]$
by(*force intro*: *bisimInputPres*)
moreover from ‹(p · xvec) ♯* N› ‹(p · xvec) ♯* P› S **have** $M(\!|\lambda*(p \cdot xvec)\ (p \cdot N)|\!).(p \cdot P) = M(\!|\lambda*xvec\ N|\!).P$
apply(*simp add*: *psi.inject*) **by**(*rule inputChainAlpha*[*symmetric*]) *auto*
moreover from ‹(p · xvec) ♯* N› ‹(p · xvec) ♯* Q› S **have** $M(\!|\lambda*(p \cdot xvec)\ (p \cdot N)|\!).(p \cdot Q) = M(\!|\lambda*xvec\ N|\!).Q$
apply(*simp add*: *psi.inject*) **by**(*rule inputChainAlpha*[*symmetric*]) *auto*
ultimately show $\Psi \rhd (M(\!|\lambda*xvec\ N|\!).P)[<\sigma>] \sim (M(\!|\lambda*xvec\ N|\!).Q)[<\sigma>]$
by *force*
qed

lemma *bisimSubstCasePresAux*:
fixes Ψ :: *'b*
and *CsP* :: (*'c* × (*'a*, *'b*, *'c*) *psi*) *list*
and *CsQ* :: (*'c* × (*'a*, *'b*, *'c*) *psi*) *list*

assumes *C1*: $\bigwedge\varphi\ P.\ (\varphi, P) \in set\ CsP \Longrightarrow \exists Q.\ (\varphi, Q) \in set\ CsQ \wedge guarded\ Q \wedge \Psi \rhd P \sim_s Q$
and *C2*: $\bigwedge\varphi\ Q.\ (\varphi, Q) \in set\ CsQ \Longrightarrow \exists P.\ (\varphi, P) \in set\ CsP \wedge guarded\ P \wedge \Psi \rhd P \sim_s Q$

shows $\Psi \rhd Cases\ CsP \sim_s Cases\ CsQ$
proof −
{
fix σ :: (*name list* × *'a list*) *list*

assume *wellFormedSubst* σ

have $\Psi \rhd Cases(caseListSeqSubst\ CsP\ \sigma) \sim Cases(caseListSeqSubst\ CsQ\ \sigma)$
proof(*rule bisimCasePres*)
fix φ P
assume $(\varphi, P) \in set\ (caseListSeqSubst\ CsP\ \sigma)$
then obtain $\varphi'\ P'$ **where** $(\varphi', P') \in set\ CsP$ **and** $\varphi = substCond.seqSubst\ \varphi'\ \sigma$ **and** *PeqP'*: $P = (P'[<\sigma>])$
by(*induct CsP*) *force*+
from ‹$(\varphi', P') \in set\ CsP$› **obtain** Q' **where** $(\varphi', Q') \in set\ CsQ$ **and** *guarded* Q' **and** $\Psi \rhd P' \sim_s Q'$ **by**(*blast dest*: *C1*)

from ‹$(\varphi', Q') \in set\ CsQ$› ‹$\varphi = substCond.seqSubst\ \varphi'\ \sigma$› **obtain** Q **where** $(\varphi, Q) \in set\ (caseListSeqSubst\ CsQ\ \sigma)$ **and** $Q = Q'[<\sigma>]$
by(*induct CsQ*) *auto*
with *PeqP'* ‹*guarded* Q'› ‹$\Psi \rhd P' \sim_s Q'$› ‹*wellFormedSubst* σ› **show** $\exists Q.$ $(\varphi, Q) \in set\ (caseListSeqSubst\ CsQ\ \sigma) \wedge guarded\ Q \wedge \Psi \rhd P \sim Q$
by(*blast dest*: *closeSubstE guardedSeqSubst*)
next
fix $\varphi\ Q$
assume $(\varphi, Q) \in set\ (caseListSeqSubst\ CsQ\ \sigma)$
then obtain $\varphi'\ Q'$ **where** $(\varphi', Q') \in set\ CsQ$ **and** $\varphi = substCond.seqSubst\ \varphi'\ \sigma$ **and** *QeqQ'*: $Q = Q'[<\sigma>]$
by(*induct CsQ*) *force+*
from ‹$(\varphi', Q') \in set\ CsQ$› **obtain** P' **where** $(\varphi', P') \in set\ CsP$ **and** *guarded* P' **and** $\Psi \rhd P' \sim_s Q'$ **by**(*blast dest*: *C2*)
from ‹$(\varphi', P') \in set\ CsP$› ‹$\varphi = substCond.seqSubst\ \varphi'\ \sigma$› **obtain** P **where** $(\varphi, P) \in set\ (caseListSeqSubst\ CsP\ \sigma)$ **and** $P = P'[<\sigma>]$
by(*induct CsP*) *auto*
with *QeqQ'* ‹*guarded* P'› ‹$\Psi \rhd P' \sim_s Q'$› ‹*wellFormedSubst* σ› **show** $\exists P.$ $(\varphi, P) \in set\ (caseListSeqSubst\ CsP\ \sigma) \wedge guarded\ P \wedge \Psi \rhd P \sim Q$
by(*blast dest*: *closeSubstE guardedSeqSubst*)
qed
}
then show *?thesis*
by(*intro closeSubstI*) *auto*
qed

lemma *bisimSubstReflexive*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$

shows $\Psi \rhd P \sim_s P$
by(*auto intro*: *closeSubstI bisimReflexive*)

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 $\Psi \rhd P \sim_s R$
using *assms*
by(*auto intro*: *closeSubstI closeSubstE bisimTransitive*)

lemma *bisimSubstSymmetric*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$

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

assumes Ψ ▷ P ∼s Q

shows Ψ ▷ Q ∼s P
  using assms
  by(auto intro: closeSubstI closeSubstE bisimE)

lemma bisimSubstCasePres:
  fixes Ψ   :: 'b
    and CsP :: ('c × ('a, 'b, 'c) psi) list
    and CsQ :: ('c × ('a, 'b, 'c) psi) list

assumes length CsP = length CsQ
  and   C: ⋀(i::nat) φ P φ' Q. ⟦i <= length CsP; (φ, P) = nth CsP i; (φ', Q) =
nth CsQ i⟧ ⟹ φ = φ' ∧ Ψ ▷ P ∼s Q ∧ guarded P ∧ guarded Q

shows Ψ ▷ Cases CsP ∼s Cases CsQ
proof −
  {
    fix φ
      and P

    assume (φ, P) ∈ set CsP

    with ‹length CsP = length CsQ› have ∃ Q. (φ, Q) ∈ set CsQ ∧ guarded Q ∧
Ψ ▷ P ∼s Q using C
    proof(induct n==length CsP arbitrary: CsP CsQ rule: nat.induct)
      case zero then show ?case by simp
    next
      case(Suc n) then show ?case
      proof(cases CsP)
        case Nil then show ?thesis using ‹Suc n = length CsP› by simp
      next
        case(Cons P'φ CsP')
        note CsPdef = this
        then show ?thesis
        proof(cases CsQ)
          case Nil then show ?thesis using CsPdef ‹length CsP = length CsQ›
            by simp
        next
          case(Cons Q'φ CsQ')
          obtain Q' φ' where Q'phi: Q'φ=(φ',Q')
            by(induct Q'φ) auto
          show ?thesis
          proof(cases P'φ = (φ,P))
            case True
            have 0 <= length CsP unfolding CsPdef
              by simp
```

```
          moreover from True have (φ, P) = nth CsP 0 using ‹(φ, P) ∈ set
CsP› unfolding CsPdef
            by simp
          moreover have (φ', Q') = nth CsQ 0 unfolding Cons Q'phi by simp
          ultimately have φ = φ' ∧ Ψ ▷ P ∼s Q' ∧ guarded P ∧ guarded Q'
            by(rule Suc)
          then show ?thesis unfolding Cons Q'phi
            by(intro exI[where x=Q']) auto
        next
          case False
          have n = length CsP' using ‹Suc n = length CsP› unfolding CsPdef
            by simp
         moreover have length CsP' = length CsQ' using ‹length CsP = length
CsQ› unfolding CsPdef Cons by simp
           moreover from ‹(φ,P) ∈ set CsP› False have (φ, P) ∈ set CsP'
unfolding CsPdef by simp
          moreover
          {
            fix   i::nat
              and φ::'c
              and φ'::'c
              and P::('a,'b,'c) psi
              and Q::('a,'b,'c) psi
            assume i ≤ length CsP'
              and  (φ, P) = CsP' ! i
              and  (φ', Q) = CsQ' ! i
            then have (i+1) ≤ length CsP
              and (φ, P) = CsP ! (i+1)
              and (φ', Q) = CsQ ! (i+1)
              unfolding CsPdef Cons
              by simp+
            then have φ = φ' ∧ Ψ ▷ P ∼s Q ∧ guarded P ∧ guarded Q
              by(rule Suc)
          }
          note this
          ultimately have ∃ Q. (φ, Q) ∈ set CsQ' ∧ guarded Q ∧ Ψ ▷ P ∼s Q
            by(rule Suc)
          then show ?thesis unfolding Cons by auto
        qed
      qed
    qed
  qed
 }
 note this
 moreover
 {
  fix φ
    and Q
```

assume $(\varphi,\ Q) \in$ *set CsQ*

with ‹*length CsP = length CsQ*› **have** $\exists P.\ (\varphi,\ P) \in$ *set CsP* $\wedge$ *guarded P* $\wedge$ $\Psi \rhd P \sim_s Q$ **using** *C*
proof(*induct n==length CsQ arbitrary*: *CsP CsQ rule*: *nat.induct*)
case *zero* **then show** *?case* **by** *simp*
next
case(*Suc n*) **then show** *?case*
proof(*cases CsQ*)
case *Nil* **then show** *?thesis* **using** ‹*Suc n = length CsQ*› **by** *simp*
next
case(*Cons Q'φ CsQ'*)
note *CsPdef = this*
then show *?thesis*
proof(*cases CsP*)
case *Nil* **then show** *?thesis* **using** *CsPdef* ‹*length CsP = length CsQ*›
by *simp*
next
case(*Cons P'φ CsP'*)
obtain *P' φ'* **where** *P'phi*: *P'φ=(φ',P')*
by(*induct P'φ*) *auto*
show *?thesis*
proof(*cases Q'φ = (φ,Q)*)
case *True*
have *0 <= length CsP* **unfolding** *CsPdef*
by *simp*
moreover have $(\varphi',\ P') =$ *nth CsP 0* **unfolding** *Cons P'phi* **by** *simp*
moreover from *True* **have** $(\varphi,\ Q) =$ *nth CsQ 0* **using** ‹$(\varphi,\ Q) \in$ *set CsQ*› **unfolding** *CsPdef*
by *simp*
ultimately have $\varphi' = \varphi \wedge \Psi \rhd P' \sim_s Q \wedge$ *guarded P'* $\wedge$ *guarded Q*
by(*rule Suc*)
then show *?thesis* **unfolding** *Cons P'phi*
by(*intro exI*[**where** *x=P'*]) *auto*
next
case *False*
have *n = length CsQ'* **using** ‹*Suc n = length CsQ*› **unfolding** *CsPdef*
by *simp*
moreover have *length CsP' = length CsQ'* **using** ‹*length CsP = length CsQ*› **unfolding** *CsPdef Cons* **by** *simp*
moreover from ‹$(\varphi,Q) \in$ *set CsQ*› *False* **have** $(\varphi,\ Q) \in$ *set CsQ'* **unfolding** *CsPdef* **by** *simp*
moreover
{
fix *i*::*nat*
and φ::*'c*
and φ'::*'c*
and *P*::(*'a*,*'b*,*'c*) *psi*
and *Q*::(*'a*,*'b*,*'c*) *psi*

```
            assume i ≤ length CsP'
              and  (φ, P) = CsP' ! i
              and  (φ', Q) = CsQ' ! i
            then have (i+1) ≤ length CsP
              and (φ, P) = CsP ! (i+1)
              and (φ', Q) = CsQ ! (i+1)
              unfolding CsPdef Cons
              by simp+
            then have φ = φ' ∧ Ψ ▷ P ∼s Q ∧ guarded P ∧ guarded Q
              by(rule Suc)
          }
          note this
          ultimately have ∃ P. (φ, P) ∈ set CsP' ∧ guarded P ∧ Ψ ▷ P ∼s Q
            by(rule Suc)
          then show ?thesis unfolding Cons by auto
        qed
      qed
    qed
  qed
 }
 ultimately show ?thesis
   by(rule bisimSubstCasePresAux)
qed

lemma bisimSubstParPres:
 fixes Ψ :: 'b
   and P :: ('a, 'b, 'c) psi
   and Q :: ('a, 'b, 'c) psi
   and R :: ('a, 'b, 'c) psi

assumes Ψ ▷ P ∼s Q

shows Ψ ▷ P ∥ R ∼s Q ∥ R
 using assms closeSubstI closeSubstE bisimParPres by force

lemma bisimSubstResPres:
 fixes Ψ :: 'b
   and P :: ('a, 'b, 'c) psi
   and Q :: ('a, 'b, 'c) psi
   and x :: name

assumes Ψ ▷ P ∼s Q
 and   x ♯ Ψ

shows Ψ ▷ (⦇νx⦈)P ∼s (⦇νx⦈)Q
proof(rule closeSubstI)
 fix σ :: (name list × 'a list) list

 assume wellFormedSubst σ
```

obtain y::*name* **where** $y \sharp \Psi$ **and** $y \sharp P$ **and** $y \sharp Q$ **and** $y \sharp \sigma$
by(*generate-fresh name*) (*auto simp add*: *fresh-prod*)

from ‹$\Psi \rhd P \sim_s Q$› **have** $([(x, y)] \cdot \Psi) \rhd ([(x, y)] \cdot P) \sim_s ([(x, y)] \cdot Q)$
by(*rule bisimSubstClosed*)
with ‹$x \sharp \Psi$› ‹$y \sharp \Psi$› **have** $\Psi \rhd ([(x, y)] \cdot P) \sim_s ([(x, y)] \cdot Q)$
by *simp*
then have $\Psi \rhd ([(x, y)] \cdot P)[<\sigma>] \sim ([(x, y)] \cdot Q)[<\sigma>]$ **using** ‹*wellFormedSubst* σ›
by(*rule closeSubstE*)
then have $\Psi \rhd (\!|\nu y|\!)(([(x, y)] \cdot P)[<\sigma>]) \sim (\!|\nu y|\!)(([(x, y)] \cdot Q)[<\sigma>])$ **using** ‹$y \sharp \Psi$›
by(*rule bisimResPres*)
with ‹$y \sharp P$› ‹$y \sharp Q$› ‹$y \sharp \sigma$›
show $\Psi \rhd ((\!|\nu x|\!)P)[<\sigma>] \sim ((\!|\nu x|\!)Q)[<\sigma>]$
by(*simp add*: *alphaRes*)
qed

lemma *bisimSubstBangPres*:
fixes Ψ :: $'b$
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 $\Psi \rhd !P \sim_s !Q$
using *assms closeSubstI closeSubstE bisimBangPres guardedSeqSubst* **by** *force*

lemma *substNil*[*simp*]:
fixes *xvec* :: *name list*
and *Tvec* :: $'a$ *list*

assumes *wellFormedSubst* σ
and *distinct xvec*

shows $(\mathbf{0}[<\sigma>]) = \mathbf{0}$
using *assms*
by *simp*

lemma *bisimSubstParNil*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*

shows $\Psi \rhd P \parallel \mathbf{0} \sim_s P$
by(*auto intro*!: *closeSubstI bisimParNil*)

lemma *bisimSubstParComm*:

fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

shows $\Psi \rhd P \parallel Q \sim_s Q \parallel P$
by(*auto intro*!: *closeSubstI bisimParComm*)

lemma *bisimSubstParAssoc*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $R :: ('a, 'b, 'c)\ psi$

shows $\Psi \rhd (P \parallel Q) \parallel R \sim_s P \parallel (Q \parallel R)$
by(*auto intro*!: *closeSubstI bisimParAssoc*)

lemma *bisimSubstResNil*:
fixes $\Psi :: 'b$
and $x :: name$

shows $\Psi \rhd (\!|\nu x|\!)\mathbf{0} \sim_s \mathbf{0}$
proof(*rule closeSubstI*)
fix σ:: $(name\ list \times 'a\ list)\ list$

assume *wellFormedSubst* σ
obtain y::*name* **where** $y \sharp \Psi$ **and** $y \sharp \sigma$
by(*generate-fresh name*) (*auto simp add*: *fresh-prod*)
have $\Psi \rhd (\!|\nu y|\!)\mathbf{0} \sim \mathbf{0}$ **by**(*rule bisimResNil*)
with ‹$y \sharp \sigma$› ‹*wellFormedSubst* σ› **show** $\Psi \rhd ((\!|\nu x|\!)\mathbf{0})[<\sigma>] \sim \mathbf{0}[<\sigma>]$
by(*subst alphaRes*[*of y*]) *auto*
qed

lemma *seqSubst2*:
fixes $x :: name$
and $P :: ('a, 'b, 'c)\ psi$

assumes *wellFormedSubst* σ
and $x \sharp \sigma$
and $x \sharp P$

shows $x \sharp P[<\sigma>]$
using *assms*
by(*induct* σ *arbitrary*: P, *auto*) (*blast dest*: *subst2*)

notation *substTerm.seqSubst* (‹-[<->]› [*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(*rule closeSubstI*)
fix σ:: $(name\ list \times 'a\ list)\ list$

assume *wellFormedSubst* σ
obtain y::*name* **where** $y \sharp \Psi$ **and** $y \sharp \sigma$ **and** $y \sharp P$ **and** $y \sharp Q$
by(*generate-fresh name*) (*auto simp add*: *fresh-prod*)
moreover from ‹*wellFormedSubst* σ› ‹$y \sharp \sigma$› ‹$y \sharp P$› **have** $y \sharp P[<\sigma>]$
by(*rule seqSubst2*)
then have $\Psi \rhd (\!|\nu y|\!)((P[<\sigma>]) \parallel (([(x, y)] \cdot Q)[<\sigma>])) \sim (P[<\sigma>]) \parallel (\!|\nu y|\!)(([(x, y)] \cdot Q)[<\sigma>])$
by(*rule bisimScopeExt*)
with ‹$x \sharp P$› ‹$y \sharp P$› ‹$y \sharp Q$› ‹$y \sharp \sigma$› **show** $\Psi \rhd ((\!|\nu x|\!)(P \parallel Q))[<\sigma>] \sim (P \parallel (\!|\nu x|\!) Q)[<\sigma>]$
apply(*subst alphaRes*[*of y*], *simp*)
apply(*subst alphaRes*[*of y Q*], *simp*)
by(*simp add*: *eqvts*)
qed

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(*rule closeSubstI*)
fix σ:: $(name\ list \times 'a\ list)\ list$

assume *wellFormedSubst* σ
obtain y::*name* **where** $y \sharp \Psi$ **and** $y \sharp \sigma$ **and** $y \sharp Cs$
by(*generate-fresh name*) (*auto simp add*: *fresh-prod*)

{
fix $x :: name$
and $Cs :: ('c \times ('a, 'b, 'c)\ psi)\ list$
and $\sigma :: (name\ list \times 'a\ list)\ list$

assume $x \sharp \sigma$

then have $(Cases\ map\ (\lambda(\varphi, P).\ (\varphi, (\!|\nu x|\!)P))\ Cs)[<\sigma>] = Cases\ map\ (\lambda(\varphi, P).\ (\varphi, (\!|\nu x|\!)P))\ (caseListSeqSubst\ Cs\ \sigma)$
by(*induct Cs*) *auto*

```
  }
  note C1 = this

  {
    fix x    :: name
      and y    :: name
      and Cs   :: ('c × ('a, 'b, 'c) psi) list

    assume x ♯ map fst Cs
      and  y ♯ map fst Cs
      and  y ♯ Cs

    then have (Cases map (λ(φ, P). (φ, (|νx|)P)) Cs) = Cases map (λ(φ, P). (φ,
(|νy|)P)) ([(x, y)] • Cs)
      by(induct Cs) (auto simp add: fresh-list-cons alphaRes)
  }
  note C2 = this

  from ‹y ♯ Cs› have y ♯ map fst Cs by(induct Cs) (auto simp add: fresh-list-cons
fresh-list-nil)
  from ‹y ♯ Cs› ‹y ♯ σ› ‹x ♯ map fst Cs› ‹wellFormedSubst σ›  have y ♯ map fst
(caseListSeqSubst ([(x, y)] • Cs) σ)
   by(induct Cs) (auto intro: substCond.seqSubst2 simp add: fresh-list-cons fresh-list-nil
fresh-prod)
  then have Ψ ▷ (|νy|)(Cases(caseListSeqSubst ([(x, y)] • Cs) σ)) ∼ Cases map
(λ(φ, P). (φ, (|νy|)P)) (caseListSeqSubst ([(x, y)] • Cs) σ)
    by(rule bisimCasePushRes)

  with ‹y ♯ Cs› ‹x ♯ map fst Cs› ‹y ♯ map fst Cs› ‹y ♯ σ› ‹wellFormedSubst σ›
 show Ψ ▷ ((|νx|)(Cases Cs))[<σ>] ∼ (Cases map (λ(φ, P). (φ, (|νx|)P)) Cs)[<σ>]
    apply(subst C2[of x Cs y])
      apply assumption+
    apply(subst C1)
     apply assumption+
    apply(subst alphaRes[of y], simp)
    by(simp add: eqvts)
qed

lemma bisimSubstOutputPushRes:
  fixes x :: name
    and Ψ :: 'b
    and M :: 'a
    and N :: 'a
    and P :: ('a, 'b, 'c) psi

assumes x ♯ M
  and   x ♯ N

shows Ψ ▷ (|νx|)(M⟨N⟩.P) ∼s M⟨N⟩.(|νx|)P
```

proof(*rule closeSubstI*)
fix σ:: (*name list* × *'a list*) *list*

assume *wellFormedSubst* σ
obtain *y*::*name* **where** $y \sharp \Psi$ **and** $y \sharp \sigma$ **and** $y \sharp P$ **and** $y \sharp M$ **and** $y \sharp N$
by(*generate-fresh name*) (*auto simp add*: *fresh-prod*)
from ‹*wellFormedSubst* σ› ‹$y \sharp M$› ‹$y \sharp \sigma$› **have** $y \sharp M[<\sigma>]$ **by** *auto*
moreover from ‹*wellFormedSubst* σ› ‹$y \sharp N$› ‹$y \sharp \sigma$› **have** $y \sharp N[<\sigma>]$ **by** *auto*
ultimately have $\Psi \rhd (\!|\nu y|\!)((M[<\sigma>])\langle(N[<\sigma>])\rangle.(([(x,\ y)] \cdot P)[<\sigma>])) \sim (M[<\sigma>])\langle(N[<\sigma>])\rangle.((\!|\nu y|\!)(([(x,\ y)] \cdot P)[<\sigma>]))$
by(*rule bisimOutputPushRes*)
with ‹$y \sharp M$› ‹$y \sharp N$› ‹$y \sharp P$› ‹$x \sharp M$› ‹$x \sharp N$› ‹$y \sharp \sigma$› ‹*wellFormedSubst* σ›
show $\Psi \rhd ((\!|\nu x|\!)(M\langle N\rangle.P))[<\sigma>] \sim (M\langle N\rangle.(\!|\nu x|\!)P)[<\sigma>]$
apply(*subst alphaRes*[*of y*], *simp*)
apply(*subst alphaRes*[*of y P*], *simp*)
by(*simp add*: *eqvts*)
qed

lemma *bisimSubstInputPushRes*:
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 $\Psi \rhd (\!|\nu x|\!)(M(\!|\lambda * xvec\ N|\!).P) \sim_s M(\!|\lambda * xvec\ N|\!).(\!|\nu x|\!)P$
proof(*rule closeSubstI*)
fix σ:: (*name list* × *'a list*) *list*

assume *wellFormedSubst* σ
obtain *y*::*name* **where** $y \sharp \Psi$ **and** $y \sharp \sigma$ **and** $y \sharp P$ **and** $y \sharp M$ **and** $y \sharp$ *xvec* **and** $y \sharp N$
by(*generate-fresh name*) (*auto simp add*: *fresh-prod*)
obtain *p*::*name prm* **where** $(p \cdot xvec) \sharp* N$ **and** $(p \cdot xvec) \sharp* P$ **and** $x \sharp (p \cdot xvec)$ **and** $y \sharp (p \cdot xvec)$ **and** $(p \cdot xvec) \sharp* \sigma$
and *S*: *set p* ⊆ *set xvec* × *set*($p \cdot xvec$)
by(*rule name-list-avoiding*[**where** *c*=(N, P, x, y, σ)]) *auto*

from ‹*wellFormedSubst* σ› ‹$y \sharp M$› ‹$y \sharp \sigma$ ›› **have** $y \sharp M[<\sigma>]$ **by** *auto*
moreover note ‹$y \sharp (p \cdot xvec)$›
moreover from ‹$y \sharp N$› **have** $(p \cdot y) \sharp (p \cdot N)$ **by**(*simp add*: *pt-fresh-bij*[*OF pt-name-inst*, *OF at-name-inst*])
with ‹$y \sharp$ *xvec*› ‹$y \sharp (p \cdot xvec)$› *S* **have** $y \sharp p \cdot N$ **by** *simp*
with ‹*wellFormedSubst* σ› **have** $y \sharp (p \cdot N)[<\sigma>]$ **using** ‹$y \sharp \sigma$› **by** *auto*

ultimately have $\Psi \rhd (\!|\nu y|\!)((M[<\sigma>])(\!|\lambda*(p \cdot xvec)\ ((p \cdot N)[<\sigma>])|\!).(([[(x,\ y)] \cdot (p \cdot P))[<\sigma>])) \sim (M[<\sigma>])(\!|\lambda*(p \cdot xvec)\ ((p \cdot N)[<\sigma>])|\!).((\!|\nu y|\!)(([[(x,\ y)] \cdot p \cdot P)[<\sigma>]))$
by(*rule bisimInputPushRes*)
with $\langle y \sharp M\rangle\ \langle y \sharp N\rangle\ \langle y \sharp P\rangle\ \langle x \sharp M\rangle\ \langle x \sharp N\rangle\ \langle y \sharp xvec\rangle\ \langle x \sharp xvec\rangle\ \langle(p \cdot xvec) \sharp* N\rangle\ \langle(p \cdot xvec) \sharp* P\rangle$
$\langle x \sharp (p \cdot xvec)\rangle\ \langle y \sharp (p \cdot xvec)\rangle\ \langle y \sharp \sigma\rangle\ \langle(p \cdot xvec) \sharp* \sigma\rangle\ S\ \langle wellFormedSubst\ \sigma\rangle$
show $\Psi \rhd ((\!|\nu x|\!)(M(\!|\lambda*xvec\ N|\!).P))[<\sigma>] \sim (M(\!|\lambda*xvec\ N|\!).(\!|\nu x|\!)P)[<\sigma>]$
apply(*subst inputChainAlpha'*)
apply *assumption+*
apply(*subst inputChainAlpha'*[*of p xvec*])
apply(*simp add*: *abs-fresh-star*)
apply *assumption+*
apply(*simp add*: *eqvts*)
apply(*subst alphaRes*[*of y*], *simp*)
apply(*simp add*: *inputChainFresh*)
apply(*simp add*: *freshChainSimps*)
apply(*subst alphaRes*[*of y* $(p \cdot P)$])
apply(*simp add*: *freshChainSimps*)
by(*simp add*: *freshChainSimps eqvts*)
qed

lemma *bisimSubstResComm*:
fixes x :: *name*
and y :: *name*

shows $\Psi \rhd (\!|\nu x|\!)((\!|\nu y|\!)P) \sim_s (\!|\nu y|\!)((\!|\nu x|\!)P)$
proof(*cases* $x = y$)
assume $x = y$
then show *?thesis* **by**(*force intro*: *bisimSubstReflexive*)
next
assume $x \neq y$
show *?thesis*
proof(*rule closeSubstI*)
fix σ:: (*name list* × *'a list*) *list*
assume *wellFormedSubst* σ

obtain x'::*name* **where** $x' \sharp \Psi$ **and** $x' \sharp \sigma$ **and** $x' \sharp P$ **and** $x \neq x'$ **and** $y \neq x'$
by(*generate-fresh name*) (*auto simp add*: *fresh-prod*)
obtain y'::*name* **where** $y' \sharp \Psi$ **and** $y' \sharp \sigma$ **and** $y' \sharp P$ **and** $x \neq y'$ **and** $y \neq y'$ **and** $x' \neq y'$
by(*generate-fresh name*) (*auto simp add*: *fresh-prod*)

have $\Psi \rhd (\!|\nu x'|\!)((\!|\nu y'|\!)(([[(x,\ x')] \cdot [(y,\ y')] \cdot P)[<\sigma>])) \sim (\!|\nu y'|\!)((\!|\nu x'|\!)(([[(x,\ x')] \cdot [(y,\ y')] \cdot P)[<\sigma>]))$
by(*rule bisimResComm*)
moreover from $\langle x' \sharp P\rangle\ \langle y' \sharp P\rangle\ \langle x \neq y'\rangle\ \langle x' \neq y'\rangle$ **have** $(\!|\nu x|\!)((\!|\nu y|\!)P) = (\!|\nu x'|\!)((\!|\nu y'|\!)(([[(x,\ x')] \cdot [(y,\ y')] \cdot P)))$

apply(*subst alphaRes*[*of* y' P], *simp*)
by(*subst alphaRes*[*of* x']) (*auto simp add*: *abs-fresh fresh-left calc-atm eqvts*)
moreover from ‹$x' \sharp P$› ‹$y' \sharp P$› ‹$y \neq x'$› ‹$x \neq y'$› ‹$x' \neq y'$› ‹$x \neq x'$› ‹$x \neq y$›
have $(\!|\nu y|\!)((\!|\nu x|\!)P) = (\!|\nu y'|\!)((\!|\nu x'|\!)(([(x, x')] \cdot [(y, y')] \cdot P)))$
apply(*subst alphaRes*[*of* x' P], *simp*)
apply(*subst alphaRes*[*of* y'], *simp add*: *abs-fresh fresh-left calc-atm*)
apply(*simp add*: *eqvts calc-atm*)
by(*subst perm-compose*) (*simp add*: *calc-atm*)

ultimately show $\Psi \rhd ((\!|\nu x|\!)((\!|\nu y|\!)P))[<\sigma>] \sim ((\!|\nu y|\!)((\!|\nu x|\!)P))[<\sigma>]$
using ‹*wellFormedSubst* σ› ‹$x' \sharp \sigma$› ‹$y' \sharp \sigma$›
by *simp*
qed
qed

lemma *bisimSubstExtBang*:
fixes Ψ :: $'b$
and P :: ($'a$, $'b$, $'c$) *psi*

assumes *guarded* P

shows $\Psi \rhd !P \sim_s P \parallel !P$
using *assms closeSubstI bangExt guardedSeqSubst* **by** *force*

lemma *structCongBisimSubst*:
fixes P :: ($'a$, $'b$, $'c$) *psi*
and Q :: ($'a$, $'b$, $'c$) *psi*

assumes $P \equiv_s Q$

shows $P \sim_s Q$
using *assms*
by(*induct rule*: *structCong.induct*)
(*auto intro*: *bisimSubstReflexive bisimSubstSymmetric bisimSubstTransitive bisimSubstParComm bisimSubstParAssoc bisimSubstParNil bisimSubstResNil bisimSubstResComm bisimSubstScopeExt bisimSubstCasePushRes bisimSubstInputPushRes bisimSubstOutputPushRes bisimSubstExtBang*)

end

end
theory *Broadcast-Thms*
imports *Broadcast-Chain Broadcast-Frame Semantics Simulation Bisimulation Sim-Pres*
Sim-Struct-Cong Bisim-Pres Bisim-Struct-Cong Bisim-Subst
begin

context *env*
begin

2.1 Syntax

2.1.1 Psi calculus agents – Definition 2

M, N range over message terms. P, Q range over processes. C ranges over cases.

- Output: $M\langle N\rangle.P$
- Input: $M(\!|\lambda * xvec\ N|\!).P$
- Case: $Case\ C$
- Par: $P \parallel Q$
- Res: $(\!|\nu x|\!)P$
- Assert: $\{\!|\Psi|\!\}$
- Bang: $!P$
- Cases: $\Box\ \varphi \Rightarrow P\ C$

2.1.2 Parameters – Definition 1

- Channel equivalence: $M \leftrightarrow N$
- Composition: $\Psi_P \otimes \Psi_Q$
- Unit: $\mathbf{1}$
- Entailment: $\Psi \vdash \varphi$

2.1.3 Extra predicates for broadcast – Definition 5

- Output connectivity: $M \preceq N$
- Input connectivity: $M \succeq N$

2.1.4 Transitions – Definition 3

- $\Psi \rhd P \longmapsto \alpha\ P'$

2.1.5 Actions (α) – Definition 7

- Input: $M(\!|N|\!)$
- Output: $M(\!|\nu * xvec|\!)\langle N\rangle$
- Broadcast input: ¿$M(\!|N|\!)$
- Broadcast output: ¡$M(\!|\nu * xvec|\!)\langle N\rangle$
- Silent action: τ

2.2 Semantics

2.2.1 Basic Psi semantics – Table 1

- Theorem *Input*:

  $[\![\Psi \vdash M \leftrightarrow K;\ \mathit{distinct}\ xvec;\ \mathit{set}\ xvec \subseteq \mathit{supp}\ N;\ \mathit{length}\ xvec = \mathit{length}\ Tvec]\!]$
  $\Longrightarrow \Psi \rhd M(\!|\lambda * xvec\ N|\!).P \longmapsto K(\!|N[xvec ::= Tvec]|\!) \prec P[xvec ::= Tvec]$

- Theorem *Output*:

  $\Psi \vdash M \leftrightarrow K \Longrightarrow \Psi \rhd M\langle N\rangle.P \longmapsto K\langle N\rangle \prec P$

- Theorem *Case*:

  $[\![\Psi \rhd P \longmapsto Rs;\ (\varphi,\ P)\ \mathit{mem}\ Cs;\ \Psi \vdash \varphi;\ \mathit{guarded}\ P]\!] \Longrightarrow \Psi \rhd \mathit{Cases}\ Cs \longmapsto Rs$

- Theorems *Par1* and *Par2*:

  $[\![\Psi \otimes \Psi_Q \rhd P \longmapsto \alpha \prec P';\ \mathit{extractFrame}\ Q = \langle A_Q,\ \Psi_Q\rangle;\ \mathit{bn}\ \alpha\ \sharp *\ Q;\ A_Q\ \sharp *\ \Psi;$
  $A_Q\ \sharp *\ P;\ A_Q\ \sharp *\ \alpha]\!]$
  $\Longrightarrow \Psi \rhd P \parallel Q \longmapsto \alpha \prec P' \parallel Q$

  $[\![\Psi \otimes \Psi_P \rhd Q \longmapsto \alpha \prec Q';\ \mathit{extractFrame}\ P = \langle A_P,\ \Psi_P\rangle;\ \mathit{bn}\ \alpha\ \sharp *\ P;\ A_P\ \sharp *\ \Psi;$
  $A_P\ \sharp *\ Q;\ A_P\ \sharp *\ \alpha]\!]$
  $\Longrightarrow \Psi \rhd P \parallel Q \longmapsto \alpha \prec P \parallel Q'$

- Theorems *Comm1* and *Comm2*:

  $[\![\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|N|\!) \prec P';\ \mathit{extractFrame}\ P = \langle A_P,\ \Psi_P\rangle;$
  $\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|\nu * xvec|\!)\langle N\rangle \prec Q';\ \mathit{extractFrame}\ Q = \langle A_Q,\ \Psi_Q\rangle;$
  $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K;\ A_P\ \sharp *\ \Psi;\ A_P\ \sharp *\ P;\ A_P\ \sharp *\ Q;\ A_P\ \sharp *\ M;\ A_P\ \sharp *\ A_Q;\ A_Q\ \sharp *\ \Psi;$
  $A_Q\ \sharp *\ P;\ A_Q\ \sharp *\ Q;\ A_Q\ \sharp *\ K;\ xvec\ \sharp *\ P]\!]$
  $\Longrightarrow \Psi \rhd P \parallel Q \longmapsto \tau \prec (\!|\nu * xvec|\!)P' \parallel Q'$

$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P';\ extractFrame\ P = \langle A_P,\ \Psi_P\rangle;$
$\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|N|\!) \prec Q';\ extractFrame\ Q = \langle A_Q,\ \Psi_Q\rangle;\ \Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K;$
$A_P \sharp{*} \Psi;\ A_P \sharp{*} P;\ A_P \sharp{*} Q;\ A_P \sharp{*} M;\ A_P \sharp{*} A_Q;\ A_Q \sharp{*} \Psi;\ A_Q \sharp{*} P;\ A_Q \sharp{*} Q;$
$A_Q \sharp{*} K;\ xvec \sharp{*} Q]\!]$
$\Longrightarrow \Psi \rhd P \parallel Q \longmapsto \tau \prec (\!|\nu{*}xvec|\!)P' \parallel Q'$

- Theorem *Open*:

  $[\![\Psi \rhd P \longmapsto M(\!|\nu{*}(xvec\ @\ yvec)|\!)\langle N\rangle \prec P';\ x \in supp\ N;\ x \sharp \Psi;\ x \sharp M;\ x \sharp xvec;$
  $x \sharp yvec]\!]$
  $\Longrightarrow \Psi \rhd (\!|\nu x|\!)P \longmapsto M(\!|\nu{*}(xvec\ @\ x\ \#\ yvec)|\!)\langle N\rangle \prec P'$

- Theorem *Scope*:

  $[\![\Psi \rhd P \longmapsto \alpha \prec P';\ x \sharp \Psi;\ x \sharp \alpha]\!] \Longrightarrow \Psi \rhd (\!|\nu x|\!)P \longmapsto \alpha \prec (\!|\nu x|\!)P'$

- Theorem *Bang*:

  $[\![\Psi \rhd P \parallel !P \longmapsto Rs;\ guarded\ P]\!] \Longrightarrow \Psi \rhd !P \longmapsto Rs$

2.2.2 Broadcast rules – Table 2

- Theorem *BrInput*:

  $[\![\Psi \vdash K \succeq M;\ distinct\ xvec;\ set\ xvec \subseteq supp\ N;\ length\ xvec = length\ Tvec]\!]$
  $\Longrightarrow \Psi \rhd M(\!|\lambda{*}xvec\ N|\!).P \longmapsto ¿K(\!|N[xvec::=Tvec]|\!) \prec P[xvec::=Tvec]$

- Theorem *BrOutput*:

  $\Psi \vdash M \preceq K \Longrightarrow \Psi \rhd M\langle N\rangle.P \longmapsto ¡K\langle N\rangle \prec P$

- Theorem *BrMerge*:

  $[\![\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P';\ extractFrame\ P = \langle A_P,\ \Psi_P\rangle;$
  $\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q';\ extractFrame\ Q = \langle A_Q,\ \Psi_Q\rangle;\ A_P \sharp{*} \Psi;\ A_P \sharp{*} P;$
  $A_P \sharp{*} Q;\ A_P \sharp{*} M;\ A_P \sharp{*} A_Q;\ A_Q \sharp{*} \Psi;\ A_Q \sharp{*} P;\ A_Q \sharp{*} Q;\ A_Q \sharp{*} M]\!]$
  $\Longrightarrow \Psi \rhd P \parallel Q \longmapsto ¿M(\!|N|\!) \prec P' \parallel Q'$

- Theorems *BrComm1* and *BrComm2*:

$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto ¿M(\!|N|\!) \prec P'$; $extractFrame\ P = \langle A_P, \Psi_P\rangle$;
$\Psi \otimes \Psi_P \rhd Q \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec Q'$; $extractFrame\ Q = \langle A_Q, \Psi_Q\rangle$; $A_P \sharp* \Psi$;
$A_P \sharp* P$; $A_P \sharp* Q$; $A_P \sharp* M$; $A_P \sharp* A_Q$; $A_Q \sharp* \Psi$; $A_Q \sharp* P$; $A_Q \sharp* Q$; $A_Q \sharp* M$;
$xvec \sharp* P]\!]$
$\Longrightarrow \Psi \rhd P \parallel Q \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec P' \parallel Q'$

$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$; $extractFrame\ P = \langle A_P, \Psi_P\rangle$;
$\Psi \otimes \Psi_P \rhd Q \longmapsto ¿M(\!|N|\!) \prec Q'$; $extractFrame\ Q = \langle A_Q, \Psi_Q\rangle$; $A_P \sharp* \Psi$;
$A_P \sharp* P$;
$A_P \sharp* Q$; $A_P \sharp* M$; $A_P \sharp* A_Q$; $A_Q \sharp* \Psi$; $A_Q \sharp* P$; $A_Q \sharp* Q$; $A_Q \sharp* M$;
$xvec \sharp* Q]\!]$
$\Longrightarrow \Psi \rhd P \parallel Q \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec P' \parallel Q'$

- Theorem *BrClose*:

  $[\![\Psi \rhd P \longmapsto ¡M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$; $x \in supp\ M$; $x \sharp \Psi]\!]$
  $\Longrightarrow \Psi \rhd (\!|\nu x|\!)P \longmapsto \tau \prec (\!|\nu x|\!)((\!|\nu * xvec|\!)P')$

- Theorem *BrOpen*:

  $[\![\Psi \rhd P \longmapsto ¡M(\!|\nu * (xvec\ @\ yvec)|\!)\langle N\rangle \prec P'$; $x \in supp\ N$; $x \sharp \Psi$; $x \sharp M$; $x \sharp xvec$;
  $x \sharp yvec]\!]$
  $\Longrightarrow \Psi \rhd (\!|\nu x|\!)P \longmapsto ¡M(\!|\nu * (xvec\ @\ x\ \#\ yvec)|\!)\langle N\rangle \prec P'$

2.2.3 Requirements for broadcast – Definition 6

- Theorem *chanOutConSupp*:

  $\Psi \vdash M \preceq N \Longrightarrow supp\ N \subseteq supp\ M$

- Theorem *chanInConSupp*:

  $\Psi \vdash N \succeq M \Longrightarrow supp\ N \subseteq supp\ M$

2.2.4 Strong bisimulation – Definition 4

- Theorem *bisim.step*:

  $[\![insertAssertion\ (extractFrame\ P)\ \Psi \simeq_F insertAssertion\ (extractFrame\ Q)\ \Psi$;
  $\Psi \rhd P \rightsquigarrow [bisim]\ Q$; $\forall \Psi'.\ \Psi \otimes \Psi' \rhd P \sim Q$; $\Psi \rhd Q \sim P]\!]$
  $\Longrightarrow \Psi \rhd P \sim Q$

2.3 Theorems

2.3.1 Congruence properties of strong bisimulation – Theorem 8

- Theorem *bisimOutputPres*:

  $\Psi \rhd P \sim Q \Longrightarrow \Psi \rhd M\langle N\rangle.P \sim M\langle N\rangle.Q$

- Theorem *bisimInputPres*:

  $(\bigwedge Tvec.\ length\ xvec = length\ Tvec \Longrightarrow \Psi \rhd P[xvec::=Tvec] \sim Q[xvec::=Tvec])$
  $\Longrightarrow$
  $\Psi \rhd M(\!|\lambda * xvec\ N|\!).P \sim M(\!|\lambda * xvec\ N|\!).Q$

- Theorem *bisimCasePres*:

  $[\![\bigwedge \varphi\ P.\ (\varphi,\ P)\ mem\ CsP \Longrightarrow \exists\, Q.\ (\varphi,\ Q)\ mem\ CsQ \wedge guarded\ Q \wedge \Psi \rhd P \sim Q;$
  $\bigwedge \varphi\ Q.\ (\varphi,\ Q)\ mem\ CsQ \Longrightarrow \exists\, P.\ (\varphi,\ P)\ mem\ CsP \wedge guarded\ P \wedge \Psi \rhd P \sim Q]\!]$
  $\Longrightarrow \Psi \rhd Cases\ CsP \sim Cases\ CsQ$

- Theorems *bisimParPres* and *bisimParPresSym*:

  $\Psi \rhd P \sim Q \Longrightarrow \Psi \rhd P \parallel R \sim Q \parallel R$

  $\Psi \rhd P \sim Q \Longrightarrow \Psi \rhd R \parallel P \sim R \parallel Q$

- Theorem *bisimResPres*:

  $[\![\Psi \rhd P \sim Q;\ x \sharp \Psi]\!] \Longrightarrow \Psi \rhd (\!|\nu x|\!)P \sim (\!|\nu x|\!)\,Q$

- Theorem *bisimBangPres*:

  $[\![\Psi \rhd P \sim Q;\ guarded\ P;\ guarded\ Q]\!] \Longrightarrow \Psi \rhd\, !P \sim\, !Q$

2.3.2 Strong congruence, bisimulation closed under substitution – Definition 9

- Theorem *closeSubst_def*:

  $closeSubst\ Rel \equiv$
  $\{(\Psi,\ P,\ Q)\ |\Psi\ P\ Q.\ \forall \sigma.\ wellFormedSubst\ \sigma \longrightarrow (\Psi,\ P[<\sigma>],\ Q[<\sigma>]) \in Rel\}$

- $\Psi \rhd P \sim_s Q$

2.3.3 Strong congruence is a process congruence for all Ψ – Theorem 10

- Theorem *bisimSubstOutputPres*:

  $\Psi \rhd P \sim_s Q \Longrightarrow \Psi \rhd M\langle N\rangle.P \sim_s M\langle N\rangle.Q$

- Theorem *bisimSubstInputPres*:

  $[\![\Psi \rhd P \sim_s Q;\ xvec \sharp* \Psi;\ distinct\ xvec]\!] \Longrightarrow \Psi \rhd M(\!|\lambda* xvec\ N|\!).P \sim_s M(\!|\lambda* xvec\ N|\!).Q$

- Theorem *bisimSubstCasePres*:

  $[\![length\ CsP = length\ CsQ;$
  $\bigwedge i\ \varphi\ P\ \varphi'\ Q.$
  $\quad [\![i \leq length\ CsP;\ (\varphi,\ P) = CsP\ !\ i;\ (\varphi',\ Q) = CsQ\ !\ i]\!]$
  $\quad \Longrightarrow \varphi = \varphi' \wedge \Psi \rhd P \sim_s Q \wedge guarded\ P \wedge guarded\ Q]\!]$
  $\Longrightarrow \Psi \rhd Cases\ CsP \sim_s Cases\ CsQ$

- Theorem *bisimSubstParPres*:

  $\Psi \rhd P \sim_s Q \Longrightarrow \Psi \rhd P \parallel R \sim_s Q \parallel R$

- Theorem *bisimSubstResPres*:

  $[\![\Psi \rhd P \sim_s Q;\ x \sharp \Psi]\!] \Longrightarrow \Psi \rhd (\!|\nu x|\!)P \sim_s (\!|\nu x|\!)Q$

- Theorem *bisimSubstBangPres*:

  $[\![\Psi \rhd P \sim_s Q;\ guarded\ P;\ guarded\ Q]\!] \Longrightarrow \Psi \rhd !P \sim_s !Q$

2.3.4 Structural equivalence – Theorem 11

- Theorem *bisimCasePushRes*:

  $x \sharp map\ fst\ Cs \Longrightarrow \Psi \rhd (\!|\nu x|\!)(Cases\ Cs) \sim Cases\ map\ (\lambda(\varphi,\ P).\ (\varphi,\ (\!|\nu x|\!)P))\ Cs$

- Theorem *bisimInputPushRes*:

  $[\![x \sharp M;\ x \sharp xvec;\ x \sharp N]\!] \Longrightarrow \Psi \rhd (\!|\nu x|\!)(M(\!|\lambda* xvec\ N|\!).P) \sim M(\!|\lambda* xvec\ N|\!).(\!|\nu x|\!)P$

- Theorem *bisimOutputPushRes*:

$[\![x \sharp M; x \sharp N]\!] \Longrightarrow \Psi \rhd (\!|\nu x|\!)(M\langle N\rangle.P) \sim M\langle N\rangle.(\!|\nu x|\!)P$

- Theorem *bisimParAssoc*:

  $\Psi \rhd P \parallel Q \parallel R \sim P \parallel (Q \parallel R)$

- Theorem *bisimParComm*:

  $\Psi \rhd P \parallel Q \sim Q \parallel P$

- Theorem *bisimResNil*:

  $\Psi \rhd (\!|\nu x|\!)\mathbf{0} \sim \mathbf{0}$

- Theorem *bisimScopeExt*:

  $x \sharp P \Longrightarrow \Psi \rhd (\!|\nu x|\!)(P \parallel Q) \sim P \parallel (\!|\nu x|\!)Q$

- Theorem *bisimResComm*:

  $\Psi \rhd (\!|\nu x|\!)((\!|\nu y|\!)P) \sim (\!|\nu y|\!)((\!|\nu x|\!)P)$

- Theorem *bangExt*:

  $guarded\ P \Longrightarrow \Psi \rhd !P \sim P \parallel !P$

- Theorem *bisimParNil*:

  $\Psi \rhd P \parallel \mathbf{0} \sim P$

2.3.5 Support of processes in broadcast transitions – Lemma 14

- Theorem *brInputTermSupp*:

  $\Psi \rhd P \longmapsto ¿K(\!|N|\!) \prec P' \Longrightarrow supp\ K \subseteq supp\ P$

- Theorem *brOutputTermSupp*:

  $\Psi \rhd P \longmapsto RBrOut\ K\ ((\!|\nu * xvec|\!)N \prec' P') \Longrightarrow supp\ K \subseteq supp\ P$

end

end

References

[1] Jesper Bengtson. Psi-calculi in Isabelle. *Arch. Formal Proofs*, 2012. https://www.isa-afp.org/entries/Psi_Calculi.html, Formal proof development.

[2] Jesper Bengtson, Magnus Johansson, Joachim Parrow, and Björn Victor. Psi-calculi: a framework for mobile processes with nominal data and logic. *Log. Methods Comput. Sci.*, 7(1), 2011.

[3] Jesper Bengtson, Joachim Parrow, and Tjark Weber. Psi-calculi in Isabelle. *J. Autom. Reason.*, 56(1):1–47, 2016.

[4] Johannes Borgström, Shuqin Huang, Magnus Johansson, Palle Raabjerg, Björn Victor, Johannes Åman Pohjola, and Joachim Parrow. Broadcast Psi-calculi with an application to wireless protocols. In Gilles Barthe, Alberto Pardo, and Gerardo Schneider, editors, *Software Engineering and Formal Methods - 9th International Conference, SEFM 2011, Montevideo, Uruguay, November 14-18, 2011. Proceedings*, volume 7041 of *Lecture Notes in Computer Science*, pages 74–89. Springer, 2011.

[5] Johannes Borgström, Shuqin Huang, Magnus Johansson, Palle Raabjerg, Björn Victor, Johannes Åman Pohjola, and Joachim Parrow. Broadcast psi-calculi with an application to wireless protocols. *Softw. Syst. Model.*, 14(1):201–216, 2015.