# The pi-calculus 

Jesper Bengtson

September 13, 2023


#### Abstract

We formalise the pi-calculus using the nominal datatype package, based on ideas from the nominal logic by Pitts et al., and demonstrate an implementation in Isabelle/HOL. The purpose is to derive powerful induction rules for the semantics in order to conduct machine checkable proofs, closely following the intuitive arguments found in manual proofs. In this way we have covered many of the standard theorems of bisimulation equivalence and congruence, both late and early, and both strong and weak in a uniform manner. We thus provide one of the most extensive formalisations of a the pi-calculus ever done inside a theorem prover.

A significant gain in our formulation is that agents are identified up to alpha-equivalence, thereby greatly reducing the arguments about bound names. This is a normal strategy for manual proofs about the picalculus, but that kind of hand waving has previously been difficult to incorporate smoothly in an interactive theorem prover. We show how the nominal logic formalism and its support in Isabelle accomplishes this and thus significantly reduces the tedium of conducting completely formal proofs. This improves on previous work using weak higher order abstract syntax since we do not need extra assumptions to filter out exotic terms and can keep all arguments within a familiar first-order logic.


## Contents

1 Overview 1
2 Formalisation 2

## 1 Overview

The following results of the pi-calculus meta-theory are formalised, where the notation (e) means that the results cover the early operational semantics and (l) the late one.

- strong bisimilarity is preserved by all operators except the input-prefix (e/l)
- strong equivalence is a congruence (e/l)
- weak bisimilarity is preserved by all operators except the input-prefix and sum (e/l)
- weak congruence is a congruence (e/l)
- strong equivalence respect the laws of structural congruence (l)
- all strongly equivalent agents are also weakly congruent which in turn are weakly bisimilar. Moreover, strongly equivalent agents are also strongly bisimilar (e/l)
- all late equivalences are included in their early counterparts.
- as a corollary of the last three points, all mentioned equivalences respect the laws of structural congruence
- the axiomatisation of the finite fragment of strong late bisimilarity is sound and complete
- The Hennessy lemma (l)

The file naming convention is hopefully self-explanatory, where the prefixes Strong and Weak denote that the file covers theories required to formalise properties of strong and weak bisimilarity respectively; if the file name contians Early or Late the theories work with the early or the late operational semantics of the pi-calculus respectively; if the file name contains Sim the theories cover simulation, file names containing Bisim cover bisimulation, and file names containing Cong cover weak congruence; files with the suffix Pres deal with theories that reason about preservation properties of operators such as a certain simulation or bisimulation being preserved by a certain operator; files with the suffix $S C$ reason about structural congruence.

For a complete exposition of all of theories, please consult Bengtson's Ph. D. thesis [1]. A shorter presentation can be found in our LMCS article 'Formalising the pi-calculus using nominal logic' from 2009 [3]. A recollection of the axiomatisation results can be found in the SOS article 'A completeness proof for bisimulation in the pi-calculus using Isabelle' from 2007 [2].

## 2 Formalisation

```
theory Agent
    imports HOL-Nominal.Nominal
```


## begin

```
lemma \(p t-i d\) :
    fixes \(x::{ }^{\prime} a\)
        and \(a::{ }^{\prime} x\)
    assumes pt: pt TYPE ('a) TYPE ('x)
    and at: at TYPE \(\left.{ }^{\prime} x\right)\)
    shows \([(a, a)] \cdot x=x\)
proof -
    have \(x=\left([]::{ }^{\prime} x\right.\) prm \() \cdot x\)
        by (simp add: pt1[OF pt])
    also have \([(a, a)] \cdot x=([]:: ' x\) prm \() \cdot x\)
        by (simp add: pt3[OF pt] at-ds1 [OF at])
    finally show?thesis by simp
qed
lemma pt-swap:
    fixes \(x::{ }^{\prime} a\)
    and \(a::\) ' \(x\)
    and \(b::{ }^{\prime} x\)
    assumes pt: pt TYPE('a) TYPE ('x)
    and at: at TYPE \(\left({ }^{\prime} x\right)\)
    shows \([(a, b)] \cdot x=[(b, a)] \cdot x\)
proof -
    show ?thesis by (simp add: pt3[OF pt] at-ds5[OF at])
qed
atom-decl name
```

lemmas name-fresh-abs $=$ fresh-abs-fun-iff[OF pt-name-inst, OF at-name-inst, OF fs-name1]
lemmas name-bij $=a t-b i j[$ OF at-name-inst $]$
lemmas name-supp-abs = abs-fun-supp[OF pt-name-inst, OF at-name-inst, OF
fs-name1]
lemmas name-abs-eq = abs-fun-eq[OF pt-name-inst, OF at-name-inst]
lemmas name-supp $=$ at-supp[OF at-name-inst]
lemmas name-calc $=$ at-calc $[$ OF at-name-inst $]$
lemmas name-fresh-fresh $=p t$-fresh-fresh[OF pt-name-inst, OF at-name-inst]
lemmas name-fresh-left $=p t$-fresh-left[OF pt-name-inst, OF at-name-inst
lemmas name-fresh-right $=$ pt-fresh-right[OF pt-name-inst, OF at-name-inst]
lemmas name-id[simp] = pt-id[OF pt-name-inst, OF at-name-inst]
lemmas name-swap-bij[simp] $=$ pt-swap-bij[OF pt-name-inst, OF at-name-inst]
lemmas name-swap $=p t$-swap $[$ OF pt-name-inst, OF at-name-inst $]$
lemmas name-rev-per $=$ pt-rev-pi[OF pt-name-inst, OF at-name-inst $]$
lemmas name-per-rev $=$ pt-pi-rev[OF pt-name-inst, OF at-name-inst]
lemmas name-exists-fresh $=$ at-exists-fresh[OF at-name-inst, OF fs-name1]
lemmas name-perm-compose $=$ pt-perm-compose $[$ OF pt-name-inst, OF at-name-inst $]$

## nominal-datatype $p i=$ PiNil

Output name name pi (-\{-\}.- [120, 120, 110] 110)
Tau pi ( $\tau .-[120]$ 110)
| Input name《name» pi (-<->.- [120, 120, 110] 110)
| Match name name pi ([-○-]- [120, 120, 110] 110)
| Mismatch name name pi ([-申-]-[120, 120, 110] 110)
| Sum pi pi $\quad($ infixr $\oplus 90)$
| Par pi pi (infixr || 85)
|Res«name» pi (<L->-[100, 100] 100)
| Bang pi (!- [110] 110)
lemmas name-fresh $[$ simp $]=$ at-fresh $[$ OF at-name-inst $]$
lemma alphaInput:
fixes $a$ :: name
and $x::$ name
and $\quad P:: p i$
and $c::$ name
assumes $A 1: c \sharp P$
shows $a<x>. P=a<c>.([(x, c)] \cdot P)$
proof (cases $x=c$ )
assume $x=c$
thus ?thesis by (simp)
next
assume $x \neq c$
with $A 1$ show ?thesis
by (simp add: pi.inject alpha name-fresh-left name-calc)
qed
lemma alphaRes:
fixes $a$ :: name
and $P:: p i$
and $c::$ name
assumes $A 1: c \sharp P$
shows $<\nu a>P=\langle\nu c\rangle([(a, c)] \cdot P)$
$\operatorname{proof}($ cases $a=c$ )
assume $a=c$
thus?thesis by simp
next
assume $a \neq c$
with $A 1$ show ?thesis
by (simp add: pi.inject alpha fresh-left name-calc)
qed
definition subst-name $::$ name $\Rightarrow$ name $\Rightarrow$ name $\Rightarrow$ name $\quad(-[-:=-][110,110$, 110] 110)
where

$$
a[b::=c] \equiv \text { if }(a=b) \text { then } c \text { else } a
$$

declare subst-name-def[simp]
lemma subst-name-eqvt[eqvt]:
fixes $p::$ name prm
and $a::$ name
and $b::$ name
and $c::$ name
shows $p \cdot(a[b::=c])=(p \cdot a)[(p \cdot b)::=(p \cdot c)]$
by (auto simp add: at-bij[OF at-name-inst $]$ )
nominal-primrec (freshness-context: (c::name, d::name))
subs $::$ pi $\Rightarrow$ name $\Rightarrow$ name $\Rightarrow$ pi $(-[-::=-][100,100,100] 100)$
where
$\mathbf{0}[c::=d]=\mathbf{0}$
$\mid \tau \cdot(P)[c::=d]=\tau \cdot(P[c::=d])$
$\mid a\{b\} \cdot P[c::=d]=(a[c::=d])\{(b[c::=d])\} .(P[c::=d])$
$\mid \llbracket x \neq a ; x \neq c ; x \neq d \rrbracket \Longrightarrow(a<x>. P)[c::=d]=(a[c::=d])<x>.(P[c::=d])$
$\mid[a \frown b] P[c::=d]=[(a[c::=d]) \frown(b[c::=d])](P[c::=d])$
$\mid[a \neq b] P[c::=d]=[(a[c::=d]) \neq(b[c::=d])](P[c::=d])$
$\mid(P \oplus Q)[c::=d]=(P[c::=d]) \oplus(Q[c::=d])$
$\mid(P \| Q)[c::=d]=(P[c::=d]) \|(Q[c::=d])$
$\mid \llbracket x \neq c ; x \neq d \rrbracket \Longrightarrow(<\nu x>P)[c::=d]=<\nu x>(P[c::=d])$
$\mid!P[c::=d]=!(P[c::=d])$
apply (simp-all add: abs-fresh)
apply(finite-guess)+
by (fresh-guess)+
lemma forget:
fixes $a$ :: name
and $P:: p i$
and $b::$ name
assumes $a \sharp P$
shows $P[a::=b]=P$
using assms
by (nominal-induct $P$ avoiding: a b rule: pi.strong-induct)
(auto simp add: name-fresh-abs)

```
lemma fresh-fact2[rule-format]:
    fixes \(P\) :: \(p i\)
    and \(a::\) name
    and \(b::\) name
    assumes \(a \neq b\)
    shows \(a \sharp P[a::=b]\)
using assms
by (nominal-induct \(P\) avoiding: a b rule: pi.strong-induct)
    (auto simp add: name-fresh-abs)
lemma subst-identity[simp]:
    fixes \(P\) :: \(p i\)
    and \(a::\) name
    shows \(P[a::=a]=P\)
by (nominal-induct \(P\) avoiding: a rule: pi.strong-induct) auto
lemma renaming:
    fixes \(P\) :: \(p i\)
    and \(a::\) name
    and \(b::\) name
    and \(c::\) name
    assumes \(c \sharp P\)
    shows \(P[a::=b]=([(c, a)] \cdot P)[c::=b]\)
using assms
by (nominal-induct \(P\) avoiding: a b c rule: pi.strong-induct)
    (auto simp add: name-calc name-fresh-abs)
lemma fresh-fact1:
    fixes \(P:: p i\)
    and \(a::\) name
    and \(b::\) name
    and \(c::\) name
    assumes \(a \sharp P\)
    and \(\quad a \neq c\)
    shows \(a \sharp P[b::=c]\)
using assms
by (nominal-induct \(P\) avoiding: a b c rule: pi.strong-induct)
    (auto simp add: name-fresh-abs)
lemma eqvt-subs[eqvt]:
```

```
    fixes p :: name prm
    and }P::p
    and a :: name
    and b:: name
    shows p
by(nominal-induct P avoiding: a b rule: pi.strong-induct)
    (auto simp add: name-bij)
lemma substInput[simp]:
    fixes x :: name
    and b :: name
    and c:: name
    and a :: name
    and }P::p
    assumes }x\not=
    and }x\not=
    shows (a<x>.P)[b::=c] = (a[b::=c])<x>.(P[b::=c])
proof -
    obtain y::name where }y\not=a\mathrm{ and }y\sharpP\mathrm{ and }y\not=b\mathrm{ and }y\not=
    by(generate-fresh name) (auto simp add: fresh-prod)
    from \langley\sharpP\rangle}\mathrm{ have }a<x>.P=a<y>.([(x,y)]\cdotP) by(simp add: alphaInput
    moreover have (a[b::=c])<x>. (P[b::=c]) = (a[b::=c])<y>.(([(x,y)] P P)[b::=c])
(is ?LHS = ?RHS)
    proof -
    from \langley\sharpP\rangle\langley\not=c\rangle have y\sharpP[b::=c] by(rule fresh-fact1)
    hence ?LHS = (a[b::=c])<y>.([(x,y)] • (P[b::=c])) by(simp add: alphaInput)
    moreover with }\langlex\not=b\rangle\langlex\not=c\rangle\langley\not=b\rangle\langley\not=c\rangle\mathrm{ have ...=? ?RHS
            by(auto simp add: eqvt-subs name-calc)
    ultimately show ?thesis by simp
    qed
    ultimately show ?thesis using }\langley\not=a\rangle\langley\not=b\rangle\langley\not=c\rangle\mathrm{ by simp
qed
lemma injPermSubst:
    fixes P :: pi
    and a :: name
    and b :: name
    assumes b\sharpP
    shows [(a,b)] P P=P[a::=b]
using assms
by(nominal-induct P avoiding: a b rule: pi.strong-induct)
    (auto simp add: name-calc name-fresh-abs)
```

```
lemma substRes2:
    fixes }P::p
    and a :: name
    and b :: name
    assumes b\sharpP
    shows <\nua>P=<\nub>(P[a::=b])
proof(case-tac a=b)
    assume }a=
    thus ?thesis by auto
next
    assume }a\not=
    moreover with < }b\sharpP\rangle\mathrm{ show ?thesis
        apply(simp add: pi.inject abs-fun-eq[OF pt-name-inst, OF at-name-inst])
        apply auto
        apply(simp add: renaming)
        apply(simp add: pt-swap[OF pt-name-inst, OF at-name-inst])
        apply(simp add: renaming)
        apply(simp add: pt-fresh-left[OF pt-name-inst,OF at-name-inst])
        by(force simp add: at-calc[OF at-name-inst])
qed
lemma freshRes:
    fixes P :: pi
    and a :: name
    shows }a\sharp<\nua>
by(simp add: name-fresh-abs)
lemma substRes3:
    fixes P :: pi
    and a :: name
    and b:: name
    assumes b\sharpP
    shows (<\nua>P)[a::=b]=<\nub>(P[a::=b])
proof -
    have (<\nu a>P)[a::=b]=<\nua>P
        using freshRes by(simp add: forget)
    thus ?thesis using < b #P> by(simp add: substRes2)
qed
lemma suppSubst:
    fixes }P::p
    and a :: name
    and b :: name
```

```
    shows supp (P[a::=b])\subseteq insert b ((supp P) - {a})
apply(nominal-induct P avoiding: a b rule: pi.strong-induct,
    simp-all add: pi.supp name-supp-abs name-supp supp-prod)
by(blast)+
```

primrec seqSubs :: pi $\Rightarrow$ (name $\times$ name $)$ list $\Rightarrow$ pi $(-[<->][100,100] 100)$ where
$P[<[]>]=P$
$\mid P[<(x \# \sigma)>]=(P[($ fst $x)::=($ snd $x)])[<\sigma>]$
primrec seq-subst-name $::$ name $\Rightarrow$ (name $\times$ name) list $\Rightarrow$ name where
seq-subst-name $a[]=a$
$\mid$ seq-subst-name $a(x \# \sigma)=$ seq-subst-name $(a[($ fst $x)::=($ snd $x)]) \sigma$
lemma freshSeqSubstName:
fixes $x$ :: name
and $a::$ name
and $s::($ name $\times$ name $)$ list
assumes $x \neq a$
and $\quad x \sharp s$
shows $x \neq$ seq-subst-name a s
using assms
apply(induct s arbitrary: a)
apply simp
$\operatorname{apply}($ case-tac $a a=f s t(a))$
by (force simp add: fresh-list-cons fresh-prod)+
lemma seqSubstZero[simp]:
fixes $\sigma::($ name $\times$ name $)$ list
shows $\mathbf{0}[<\sigma>]=\mathbf{0}$
by (induct $\sigma$, auto)
lemma seqSubstTau[simp]:
fixes $P$ :: $p i$
and $\quad \sigma::($ name $\times$ name $)$ list
shows $(\tau .(P))[<\sigma\rangle]=\tau .(P[<\sigma\rangle])$
by(induct $\sigma$ arbitrary: $P$, auto)
lemma seqSubstOutput[simp]:
fixes $a$ :: name
and $b::$ name
and $\quad P:: p i$
and $\quad \sigma::($ name $\times$ name $)$ list
shows $(a\{b\} . P)[<\sigma>]=($ seq-subst-name a $\sigma)\{($ seq-subst-name $b \sigma)\} .(P[<\sigma>])$ by(induct $\sigma$ arbitrary: a $b$, auto)

```
lemma seqSubstInput[simp]:
    fixes \(a\) :: name
    and \(x::\) name
    and \(\quad P:: p i\)
    and \(\quad \sigma::(\) name \(\times\) name \()\) list
```

    assumes \(x \sharp \sigma\)
    shows \((a<x>. P)[<\sigma>]=(\) seq-subst-name a \(\sigma)<x>.(P[<\sigma>])\)
    using assms
by (induct $\sigma$ arbitrary: a x P) (auto simp add: fresh-list-cons fresh-prod)
lemma seqSubstMatch[simp]:
fixes $a$ :: name
and $b::$ name
and $\quad P:: p i$
and $\quad \sigma::($ name $\times$ name $)$ list
shows $([a \frown b] P)[<\sigma>]=[($ seq-subst-name a $\sigma) \frown($ seq-subst-name $b \sigma)](P[<\sigma>])$
by (induct $\sigma$ arbitrary: ab $P$, auto)
lemma seqSubstMismatch[simp]:
fixes $a$ :: name
and $b::$ name
and $P:: p i$
and $\quad \sigma::($ name $\times$ name $)$ list
shows $([a \neq b] P)[<\sigma>]=[($ seq-subst-name a $\sigma) \neq($ seq-subst-name $b \sigma)](P[<\sigma>])$
by(induct $\sigma$ arbitrary: a b $P$, auto)
lemma seqSubstSum[simp]:
fixes $P:: p i$
and $\quad Q:: p i$
and $\quad \sigma::($ name $\times$ name $)$ list
shows $(P \oplus Q)[<\sigma\rangle]=(P[<\sigma>]) \oplus(Q[<\sigma>])$
by (induct $\sigma$ arbitrary: $P Q$, auto)
lemma seqSubstPar[simp]:
fixes $P$ :: $p i$
and $\quad Q:: p i$
and $\quad \sigma::($ name $\times$ name $)$ list
shows $(P \| Q)[<\sigma>]=(P[<\sigma>]) \|(Q[<\sigma>])$
by (induct $\sigma$ arbitrary: $P($, auto)

```
lemma seqSubstRes[simp]:
    fixes x :: name
    and }P:: p
    and }\sigma::(name \times name) lis
    assumes }x\sharp
    shows (<\nux>P)[<\sigma>]=<\nux>(P[<\sigma>])
using assms
by(induct \sigma arbitrary: x P) (auto simp add: fresh-list-cons fresh-prod)
lemma seqSubstBang[simp]:
    fixes P :: pi
    and s :: (name }\times\mathrm{ name) list
    shows (!P)[<\sigma>] =!(P[<\sigma>])
by(induct }\sigma\mathrm{ arbitrary: P, auto)
lemma seqSubstEqvt[eqvt, simp]:
    fixes P :: pi
    and }\sigma::(name\times name) lis
    and p :: name prm
    shows p\cdot(P[<\sigma>]) =(p\cdotP)[< (p\cdot\sigma)>]
by(induct \sigma arbitrary: P, auto simp add: eqvt-subs)
lemma seqSubstAppend[simp]:
    fixes }P::p
    and }\sigma::(\mathrm{ name }\times\mathrm{ name) list
    and }\mp@subsup{\sigma}{}{\prime}::(\mathrm{ name }\times\mathrm{ name) list
    shows }P[<(\sigma@\mp@subsup{\sigma}{}{\prime})>]=(P[<\sigma>])[<\mp@subsup{\sigma}{}{\prime}>
by(induct \sigma arbitrary: P, auto)
lemma freshSubstChain[intro]:
    fixes P :: pi
    and }\sigma::(name\timesname) lis
    and a :: name
    assumes }a\sharp
    and }\quada\sharp
    shows }a\sharpP[<\sigma>
using assms
by(induct \sigma arbitrary: a P, auto simp add: fresh-list-cons fresh-prod fresh-fact1)
end
```


## theory Late-Semantics <br> imports Agent <br> begin

nominal-datatype subject $=$ InputS name
| BoundOutputS name
nominal-datatype freeRes $=$ OutputR name name
$(-[-][130,130] 110)$
| TauR
nominal-datatype residual $=$ Bound $R$ subject «name» pi $(-«-» \prec-[80,80,80]$ 80)

$$
\mid \text { FreeR freeRes pi } \quad(-\prec-[80,80] 80)
$$

lemmas residualInject $=$ residual.inject freeRes.inject subject.inject
abbreviation Transitions-Inputjudge :: name $\Rightarrow$ name $\Rightarrow$ pi $\Rightarrow$ residual $(-<->\prec$ - $[80,80,80] 80)$
where $a<x>\prec P^{\prime} \equiv\left((\right.$ InputS $\left.a) « x » \prec P^{\prime}\right)$
abbreviation Transitions-BoundOutputjudge :: name $\Rightarrow$ name $\Rightarrow$ pi $\Rightarrow$ residual (-< $\nu->\prec-[80,80,80] 80)$
where $a<\nu x>\prec P^{\prime} \equiv\left(\right.$ BoundR (BoundOutputS a) x $P^{\prime}$ )
inductive transitions :: pi $\Rightarrow$ residual $\Rightarrow$ bool $\quad(-\longmapsto-[80,80] 80)$
where

$$
\begin{array}{ll}
\text { Tau: } & \tau .(P) \longmapsto \tau \prec P \\
\text { Input: } & x \neq a \Longrightarrow a<x>P \longmapsto a<x>\prec P \\
\text { | Output: } & a\{b\} \cdot P \longmapsto a[b] \prec P \\
\text { Match: } & \llbracket P \longmapsto R s \rrbracket \Longrightarrow[b \frown b] P \longmapsto R s \\
\text { Mismatch: } & \llbracket P \longmapsto R s ; a \neq b \rrbracket \Longrightarrow[a \neq b] P \longmapsto R s
\end{array}
$$

| Open: | $\llbracket P \longmapsto a[b] \prec P^{\prime} ; a \neq b \rrbracket \Longrightarrow<\nu b>P \longmapsto a<\nu b>\prec P^{\prime}$ |
| :---: | :---: |
| Sum1: | $\llbracket P \longmapsto R s \rrbracket \Longrightarrow(P \oplus Q) \longmapsto R s$ |
| Sum2: | $\llbracket Q \longmapsto R s \rrbracket \Longrightarrow(P \oplus Q) \longmapsto R s$ |
| \| Par1B: | $\llbracket P \longmapsto a<x>\prec P^{\prime} ; x \sharp P ; x \sharp Q ; x \sharp a \rrbracket \Longrightarrow P \\| Q \longmapsto a « x \gg$ |
| $\prec\left(P^{\prime} \\| Q\right)$ |  |
| \| Par1F: | $\llbracket P \longmapsto \alpha \prec P^{\prime} \rrbracket \Longrightarrow P \\| Q \longmapsto \alpha \prec\left(P^{\prime} \\| Q\right)$ |
| Par2B: | $\llbracket Q \longmapsto a<x\rangle \prec Q^{\prime} ; x \sharp P ; x \sharp Q ; x \sharp a \rrbracket \Longrightarrow P \\| Q \longmapsto a \ll x »$ |
| $\prec\left(P \\| Q^{\prime}\right)$ |  |
| Par2F: | $\llbracket Q \longmapsto \alpha \prec Q^{\dagger} \Longrightarrow P \\| Q \longmapsto \alpha \prec\left(P \\| Q^{\prime}\right)$ |
| \| Comm1: $\quad\left[P \longmapsto a<x>\prec P^{\prime} ; Q \longmapsto a[b] \prec Q^{\prime} ; x \sharp P ; x \sharp Q ; x \neq a ; x\right.$ |  |
| $\neq b ; x \sharp Q^{\prime} \Longrightarrow P\left\\|Q \longmapsto \tau \prec P^{\prime}[x::=b]\right\\| Q^{\prime}$ |  |
| Comm2: | $\llbracket P \longmapsto a[b] \prec P^{\prime} ; Q \longmapsto a<x>\prec Q^{\prime} ; x \sharp P ; x \sharp Q ; x \neq a ; x$ |
| $\neq b ; x \sharp P \rrbracket \Longrightarrow P\left\\|Q \longmapsto \tau \prec P^{\prime}\right\\| Q^{\prime}[x::=b]$ |  |
| Close1: | $\llbracket P \longmapsto a<x>\prec P^{\prime} ; Q \longmapsto a<\nu y>\prec Q^{\prime} ; x \sharp P ; x \sharp Q ; y \sharp P ;$ |

```
                    y\sharpQ;x\not=a;x\sharp \mp@subsup{Q}{}{\prime};y\not=a;y\sharp\mp@subsup{P}{}{\prime};x\not=y\rrbracket\LongrightarrowP|Q\longmapsto~
<\nuy>(P'[x::=y]| Q 踇
| Close2: }\quad|P\longmapstoa<\nuy>\prec \prec P';Q\longmapstoa<x>\prec\prec晶;x\sharpP;x\sharpQ;y\sharpP
    y\sharpQ;x\not=a;x\sharp\mp@subsup{P}{}{\prime};y\not=a;y\sharp\mp@subsup{Q}{}{\prime};x\not=y\rrbracket\LongrightarrowP||
<\nuy>( }\mp@subsup{P}{}{\prime}|\mp@subsup{Q}{}{\prime}[x::=y]
|esB: _ |P\longmapstoa<x>\prec 敖;y\sharpa;y\not=x;x\sharpP;x\sharpa\rrbracket\Longrightarrow<\nuy>P\longmapsto
a<x><<<\nuy> P'
| ResF: }\quad\llbracketP\longmapsto\alpha\prec\mp@subsup{P}{}{\prime};y\sharp\alpha\rrbracket\Longrightarrow<\nuy>P\longmapsto\alpha\longmapsto<<\nu>>P
| Bang: }\quad\llbracketP|!P\longmapstoRs\rrbracket\Longrightarrow!P\longmapstoR\longmapstoR
equivariance transitions
nominal－inductive transitions
by（auto simp add：abs－fresh fresh－fact2）
lemma alphaBoundResidual：
fixes \(a\) ：：subject
and \(x\) ：：name
and \(P:: p i\)
and \(x^{\prime}::\) name
assumes \(A 1: x^{\prime} \sharp P\)
shows \(a « x>\prec P=a « x^{\prime} » \prec\left(\left[\left(x, x^{\prime}\right)\right] \cdot P\right)\)
proof（cases \(x=x^{\prime}\) ）
assume \(x=x^{\prime}\)
thus？thesis by simp
next
assume \(x \neq x^{\prime}\)
with \(A 1\) show ？thesis
by（simp add：residualInject alpha name－fresh－left name－calc）
qed
lemma freshResidual：
fixes \(P\) ：：\(p i\)
and \(\quad R s::\) residual
and \(x\) ：：name
assumes \(P \longmapsto R s\)
and \(\quad x \sharp P\)
shows \(x \sharp R s\)
using assms
by（nominal－induct rule：transitions．strong－induct）
（auto simp add：abs－fresh fresh－fact2 fresh－fact1）
lemma freshBoundDerivative：
assumes \(P \longmapsto a « x » \prec P^{\prime}\)
```

```
    and }\quady\sharp
    shows }y\sharp
    and }y\not=x\Longrightarrowy\sharp\mp@subsup{P}{}{\prime
apply -
using assms
by(fastforce dest: freshResidual simp add:abs-fresh)+
lemma freshFreeDerivative:
    fixes P :: pi
    and }\alpha\mathrm{ :: freeRes
    and }\mp@subsup{P}{}{\prime}:: p
    and y :: name
    assumes }P\longmapsto\alpha\prec\mp@subsup{P}{}{\prime
    and }\quady\sharp
    shows }y\sharp
    and }y\sharp\mp@subsup{P}{}{\prime
apply -
using assms
by(fastforce dest: freshResidual)+
lemma substTrans[simp]:
    fixes b :: name
    and }P::p
    and a :: name
    and c:: name
    assumes b\sharpP
    shows (P[a::=b])[b::=c]=P[a::=c]
using assms
apply(simp add: injPermSubst[THEN sym])
apply(simp add: renaming)
by(simp add: pt-swap[OF pt-name-inst, OF at-name-inst])
lemma Input:
    fixes a :: name
    and x:: name
    and }P::p
    shows }a<x>.P\longmapstoa<x>\prec
proof -
    obtain y::name where }y\not=a\mathrm{ and }y\sharp
    by(generate-fresh name, auto simp add: fresh-prod)
```



```
([(x,y)] • P)
    by(auto simp add: alphaBoundResidual alphaInput)
```

```
    with }\langley\not=a\rangle\mathrm{ show ?thesis by(force intro: Input)
qed
declare perm-fresh-fresh[simp] name-swap[simp] fresh-prod[simp]
lemma Par1B:
    fixes P :: pi
    and a :: subject
    and }x\mathrm{ :: name
    and }\mp@subsup{P}{}{\prime}::p
    and }Q ::p
    assumes }P\longmapstoa<x>>>\mp@subsup{P}{}{\prime
    and }\quadx\sharp
    shows P|Q\longmapstoa<x»\prec 敖|Q
proof -
    obtain y::name where }y\sharpP\mathrm{ and }y\sharp\mp@subsup{P}{}{\prime}\mathrm{ and }y\sharpQ\mathrm{ and }y\sharp
        by(generate-fresh name, auto)
    from \langleP\longmapstoa<x» \prec P P}\rangle\langley\sharp\mp@subsup{P}{}{\prime}\rangle\mathrm{ have }P\longmapstoa«y»\prec ([(x,y)] \cdot P'
        by(simp add: alphaBoundResidual)
    hence }P||\longmapstoa«y>\prec([(x,y)]\cdot\mp@subsup{P}{}{\prime})|Q\mathrm{ using < y#P><y#Q><y#a>
        by(rule Par1B)
    with \langlex\sharp Q>\langley\sharp 敖〉\langley\sharpQ> show ?thesis
        by(subst alphaBoundResidual[where }\mp@subsup{x}{}{\prime}=y])\mathrm{ auto
qed
lemma Par2B
    fixes Q :: pi
    and a :: subject
    and }x\mathrm{ :: name
    and }\mp@subsup{Q}{}{\prime}:: p
    and }P::p
    assumes QTrans: Q}\longmapstoa<x>\prec Q Q
    and }x\sharp
    shows P| Q\longmapstoa«x»\precP| Q'
proof -
    obtain y::name where }y\sharpQ\mathrm{ and }y\sharp\mp@subsup{Q}{}{\prime}\mathrm{ and }y\sharpP\mathrm{ and }y\sharp
        by(generate-fresh name, auto simp add: fresh-prod)
    from QTrans «y\sharp\mp@subsup{Q}{}{\prime}>\mathrm{ have }Q\longmapstoa«y»\prec([(x,y)]\cdotQ')
        by(simp add:alphaBoundResidual)
    hence P|Q |
        by(rule Par2B)
    moreover have a«y» \precP| ([(x,y)] • Q')=a«x»\precP| | Q'
    proof -
    from <y\sharp Q'>\langlex\sharpP> have }x\sharpP|([(x,y)]\cdot\mp@subsup{Q}{}{\prime})\mathrm{ by(auto simp add: calc-atm
fresh-left)
```

with $\langle x \sharp P\rangle\langle y \sharp P\rangle$ show ?thesis by (simp only: alphaBoundResidual, auto simp add: name-swap name-fresh-fresh)
qed
ultimately show ?thesis by simp
qed
lemma Comm1:
fixes $P:: p i$
and $a$ :: name
and $x$ :: name
and $\quad P^{\prime}:: p i$
and $\quad Q:: p i$
and $b::$ name
and $Q^{\prime}:: p i$
assumes PTrans: $P \longmapsto a<x>\prec P^{\prime}$
and $\quad Q$ Trans: $Q \longmapsto a[b] \prec Q^{\prime}$
shows $P\left\|Q \longmapsto \tau \prec P^{\prime}[x::=b]\right\| Q^{\prime}$
proof -
obtain $y$ ::name where $y \sharp P$ and $y \sharp P^{\prime}$ and $y \sharp Q$ and $y \neq a$ and $y \neq b$ and $y \sharp Q^{\prime}$
by (generate-fresh name, auto simp add: fresh-prod)
from PTrans $\left\langle y \sharp P^{\prime}\right\rangle$ have $P \longmapsto a<y>\prec\left([(x, y)] \cdot P^{\prime}\right)$
by (simp add: alphaBoundResidual)
hence $P\left\|Q \longmapsto \tau \prec\left([(x, y)] \cdot P^{\prime}\right)[y::=b]\right\| Q^{\prime}$
using $Q$ Trans $\langle y \sharp P\rangle\langle y \sharp Q\rangle\langle y \neq a\rangle\langle y \neq b\rangle\left\langle y \sharp Q^{\prime}\right\rangle$
by(rule Comm1)
with $\left\langle y \sharp P^{\prime}\right\rangle$ show ?thesis $\mathbf{b y}($ simp add: renaming name-swap)
qed
lemma Comm2:
fixes $P:: p i$
and $a$ :: name
and $b$ :: name
and $P^{\prime}:: p i$
and $\quad Q:: p i$
and $x$ :: name
and $\quad Q^{\prime}:: p i$
assumes $P$ Trans: $P \longmapsto a[b] \prec P^{\prime}$
and $\quad$ QTrans: $Q \longmapsto a<x>\prec Q^{\prime}$
shows $P\left\|Q \longmapsto \tau \prec P^{\prime}\right\|\left(Q^{\prime}[x::=b]\right)$
proof -
obtain $y$ ::name where $y \sharp P$ and $y \sharp P^{\prime}$ and $y \sharp Q$ and $y \neq a$ and $y \neq b$ and $y \sharp Q^{\prime}$
by (generate-fresh name, auto simp add: fresh-prod)
from $Q$ Trans $\left\langle y \sharp Q^{\prime}\right\rangle$ have $Q \longmapsto a<y>\prec\left([(x, y)] \cdot Q^{\prime}\right)$
by (simp add: alphaBoundResidual)
with PTrans have $P\left\|Q \longmapsto \tau \prec P^{\prime}\right\|\left(\left([(x, y)] \cdot Q^{\prime}\right)[y::=b]\right)$
using $\langle y \sharp P\rangle\langle y \sharp Q\rangle\langle y \neq a\rangle\langle y \neq b\rangle\left\langle y \sharp P^{\prime}\right\rangle$
by(rule Comm2)
with $\left\langle y \sharp Q^{\prime}\right\rangle$ show ?thesis $\mathbf{b y}$ (simp add: renaming name-swap)
qed
lemma Close1:
fixes $P$ :: pi
and $a$ :: name
and $x$ :: name
and $P^{\prime}:: p i$
and $Q$ :: pi
and $y$ :: name
and $\quad Q^{\prime}:: p i$
assumes PTrans: $P \longmapsto a<x>\prec P^{\prime}$
and $\quad Q$ Trans: $Q \longmapsto a<\nu y>\prec Q^{\prime}$
and $\quad y \sharp P$
shows $P \| Q \longmapsto \tau \prec<\nu y>\left(P^{\prime}[x::=y] \| Q^{\prime}\right)$
proof -
obtain $x^{\prime}:: n a m e$ where $x^{\prime} \sharp P$ and $x^{\prime} \sharp P^{\prime}$ and $x^{\prime} \sharp Q$ and $x^{\prime} \sharp Q^{\prime}$ and $x^{\prime} \neq a$
by (generate-fresh name, auto simp add: fresh-prod)
obtain $y^{\prime}:$ :name where $y^{\prime} \sharp P$ and $y^{\prime} \sharp Q^{\prime}$ and $y^{\prime} \sharp Q$ and $y^{\prime} \sharp P^{\prime}$ and $y^{\prime} \neq x^{\prime}$ and $y^{\prime} \neq y$ and $y^{\prime} \neq a$
by (generate-fresh name, auto simp add: fresh-prod)
from PTrans $\left\langle x^{\prime} \sharp P^{\prime}\right\rangle$ have $P \longmapsto a<x^{\prime}>\prec\left(\left[\left(x, x^{\prime}\right)\right] \cdot P^{\prime}\right)$
by (simp add: alphaBoundResidual)
moreover from QTrans $\left\langle y^{\prime} \sharp Q^{\prime}\right\rangle$ have $Q \longmapsto a<\nu y^{\prime}>\prec\left(\left[\left(y, y^{\prime}\right)\right] \cdot Q^{\prime}\right)$
by (simp add: alphaBoundResidual)
ultimately have $P \| Q \longmapsto \tau \prec<\nu y^{\prime}>\left(\left(\left[\left(x, x^{\prime}\right)\right] \cdot P^{\prime}\right)\left[x^{\prime}::=y\right] \|\left(\left[\left(y, y^{\prime}\right)\right] \cdot Q^{\prime}\right)\right)$
using $\left\langle y^{\prime} \sharp P\right\rangle\left\langle y^{\prime} \sharp Q\right\rangle\left\langle x^{\prime} \sharp P\right\rangle\left\langle x^{\prime} \sharp Q\right\rangle\left\langle y^{\prime} \neq x^{\prime}\right\rangle\left\langle y^{\prime} \neq a\right\rangle\left\langle x^{\prime} \neq a\right\rangle$
$\left\langle y^{\prime} \sharp P^{\prime}\right\rangle\left\langle y^{\prime} \sharp Q^{\prime}\right\rangle\left\langle x^{\prime} \sharp P^{\prime}\right\rangle\left\langle x^{\prime} \sharp Q^{\prime}\right\rangle$
apply(rule-tac Close1)
by assumption (auto simp add: fresh-left calc-atm)
moreover have $<\nu y^{\prime}>\left(\left(\left[\left(x, x^{\prime}\right)\right] \cdot P^{\prime}\right)\left[x^{\prime}::=y\right] \|\left(\left[\left(y, y^{\prime}\right)\right] \cdot Q^{\prime}\right)\right)=<\nu y>\left(P^{\prime}[x::=y]\right.$ $\| Q^{\prime}$ )
proof -
from $\left\langle x^{\prime} \sharp P^{\prime}\right\rangle$ have $\left(\left[\left(x, x^{\prime}\right)\right] \cdot P^{\prime}\right)\left[x^{\prime}::=y^{\prime}\right]=P^{\prime}\left[x::=y^{\prime}\right]$ by $($ simp add: renaming name-swap)
moreover have $y \sharp\left(P^{\prime}\left[x::=y^{\prime}\right] \|\left(\left[\left(y, y^{\prime}\right)\right] \cdot Q^{\prime}\right)\right)$
proof (case-tac $y=x$ )
assume $y=x$
with $\left\langle y^{\prime} \sharp Q^{\prime}\right\rangle\left\langle y^{\prime} \neq y\right\rangle$ show ?thesis $\mathbf{b y}($ auto simp add: fresh-fact2 fresh-left calc-atm)
next
assume $y \neq x$
with $\left\langle y \sharp P>P\right.$ Trans have $y \sharp P^{\prime}$ by(force dest: freshBoundDerivative)
with $\left\langle y^{\prime} \sharp Q^{\prime}\right\rangle\left\langle y^{\prime} \neq y\right\rangle$ show ?thesis by (auto simp add: fresh-left calc-atm fresh-fact1)

## qed

ultimately show ?thesis using $\left\langle y^{\prime} \sharp P^{\prime}\right\rangle$ apply (simp only: alphaRes)
by (auto simp add: name-swap eqvt-subs calc-atm renaming)

## qed

ultimately show? ?thesis by simp
qed
lemma Close2:
fixes $P:: p i$
and $a$ :: name
and $y$ :: name
and $P^{\prime}:: p i$
and $Q:: p i$
and $x$ :: name
and $Q^{\prime}:: p i$
assumes PTrans: $P \longmapsto a<\nu y>\prec P^{\prime}$
and $\quad$ TTrans: $Q \longmapsto a<x>\prec Q^{\prime}$
and $\quad y \sharp Q$
shows $P \| Q \longmapsto \tau \prec<\nu y>\left(P^{\prime} \|\left(Q^{\prime}[x::=y]\right)\right)$
proof -
obtain $x^{\prime}:$ :name where $x^{\prime} \sharp P$ and $x^{\prime} \sharp Q^{\prime}$ and $x^{\prime} \sharp Q$ and $x^{\prime} \sharp P^{\prime}$ and $x^{\prime} \neq a$
by (generate-fresh name, auto simp add: fresh-prod)
obtain $y^{\prime}:$ :name where $y^{\prime} \sharp P$ and $y^{\prime} \sharp P^{\prime}$ and $y^{\prime} \sharp Q$
and $y^{\prime} \sharp Q^{\prime}$ and $y^{\prime} \neq x^{\prime}$ and $y^{\prime} \neq y$ and $y^{\prime} \neq a$
by (generate-fresh name, auto simp add: fresh-prod)
from PTrans $\left\langle y^{\prime} \sharp P^{\prime}\right\rangle$ have $P \longmapsto a<\nu y^{\prime}>\prec\left(\left[\left(y, y^{\prime}\right)\right] \cdot P^{\prime}\right)$
by (simp add: alphaBoundResidual)
moreover from $Q$ Trans $\left\langle x^{\prime} \sharp Q^{\prime}\right\rangle$ have $Q \longmapsto a<x^{\prime}>\prec\left(\left[\left(x, x^{\prime}\right)\right] \cdot Q^{\prime}\right)$
by (simp add: alphaBoundResidual)
ultimately have $P \| Q \longmapsto \tau \prec<\nu y^{\prime}>\left(\left(\left[\left(y, y^{\prime}\right)\right] \cdot P^{\prime}\right) \|\left(\left(\left[\left(x, x^{\prime}\right)\right] \cdot Q^{\prime}\right)\left[x^{\prime}::=y^{\prime}\right]\right)\right)$
using $\left\langle y^{\prime} \sharp P\right\rangle\left\langle y^{\prime} \sharp Q\right\rangle\left\langle x^{\prime} \sharp P\right\rangle\left\langle x^{\prime} \sharp Q\right\rangle\left\langle y^{\prime} \neq x^{\prime}\right\rangle\left\langle x^{\prime} \neq a\right\rangle\left\langle y^{\prime} \neq a\right\rangle$ $\left\langle x^{\prime} \sharp P^{\prime}\right\rangle\left\langle x^{\prime} \sharp Q^{\prime}\right\rangle\left\langle y^{\prime} \sharp P^{\prime}\right\rangle\left\langle y^{\prime} \sharp Q^{\prime}\right\rangle$
by (rule-tac Close2) (assumption | auto simp add: fresh-left calc-atm)+
moreover have $<\nu y^{\prime}>\left(\left(\left[\left(y, y^{\prime}\right)\right] \cdot P^{\prime}\right) \|\left(\left(\left[\left(x, x^{\prime}\right)\right] \cdot Q^{\prime}\right)\left[x^{\prime}::=y\right\rceil\right)\right)=<\nu y>\left(P^{\prime}\right.$
$\left.\|\left(Q^{\prime}[x::=y]\right)\right)$
proof -
from $\left\langle x^{\prime} \sharp Q^{\prime}\right\rangle$ have $\left(\left[\left(x, x^{\prime}\right)\right] \cdot Q^{\prime}\right)\left[x^{\prime}::=y^{\prime}\right]=Q^{\prime}\left[x::=y^{\prime}\right]$ by (simp add: renaming name-swap)
moreover have $y \sharp\left(\left(\left[\left(y, y^{\prime}\right)\right] \cdot P^{\prime}\right) \|\left(Q^{\prime}[x::=y\rceil\right)\right)$
proof (case-tac $y=x$ )
assume $y=x$
with $\left\langle y^{\prime} \sharp P^{\prime}\right\rangle\left\langle y^{\prime} \neq y\right\rangle$ show ?thesis $\mathbf{b y}($ auto simp add: fresh-fact2 fresh-left calc-atm)
next
assume $y \neq x$
with $\langle y \sharp Q\rangle Q$ Trans have $y \sharp Q^{\prime}$ by (force dest: freshBoundDerivative)
with $\left\langle y^{\prime} \sharp P^{\prime}\right\rangle\left\langle y^{\prime} \neq y\right\rangle$ show ?thesis by (auto simp add: fresh-left calc-atm fresh-fact1)

## qed

ultimately show ?thesis using $\left\langle y^{\prime} \sharp Q^{\prime}\right\rangle$ apply (simp only: alphaRes)
by (auto simp add: name-swap eqvt-subs calc-atm renaming)
qed
ultimately show?thesis by simp
qed
lemma ResB:
fixes $P:: p i$
and $a$ :: subject
and $x$ :: name
and $\quad P^{\prime}:: p i$
and $y$ :: name
assumes PTrans: $P \longmapsto a « x » \prec P^{\prime}$
and $\quad y \sharp a$
and $\quad y \neq x$
shows $<\nu y>P \longmapsto a « x\rangle \prec<\nu y>P^{\prime}$
proof -
obtain $z$ where $z \sharp P$ and $z \sharp a$ and $z \neq y$ and $z \sharp P^{\prime}$
by (generate-fresh name, auto simp add: fresh-prod)
from PTrans $\left\langle z \sharp P^{\prime}\right\rangle$ have $P \longmapsto a « z » \prec\left([(x, z)] \cdot P^{\prime}\right) \mathbf{b y}(\operatorname{simp}$ add: alphaBoundResidual)
with $\langle z \sharp P\rangle\langle z \sharp a\rangle\langle z \neq y\rangle\langle y \sharp a\rangle$ have $\langle\nu y>P \longmapsto a « z\rangle \prec\langle\nu y>([(x, z)] \cdot$ $\left.P^{\prime}\right) \mathbf{b y}($ rule-tac ResB) auto
moreover have $a « z\rangle \prec\left\langle\nu y>\left([(x, z)] \cdot P^{\prime}\right)=a « x\right.$ » $\prec\left\langle\nu y>P^{\prime}\right.$
proof -
from $\left\langle z \sharp P^{\prime}\right\rangle\langle y \neq x\rangle$ have $\left.x \sharp<\nu y\right\rangle\left([(x, z)] \cdot P^{\prime}\right)$ by (auto simp add: abs-fresh fresh-left calc-atm)
with $\langle y \neq x\rangle\langle z \neq y\rangle$ show ?thesis by (simp add: alphaBoundResidual name-swap calc-atm)
qed
ultimately show ?thesis by simp
qed
lemma outputInduct[consumes 1, case-names Output Match Mismatch Sum1 Sum2 Par1 Par2 Res Bang]:
fixes $P:: p i$
and $a$ :: name
and $b$ :: name
and $P^{\prime}:: p i$
and $\quad F::{ }^{\prime} a::$ fs-name $\Rightarrow p i \Rightarrow$ name $\Rightarrow$ name $\Rightarrow p i \Rightarrow$ bool
and $C$ :: 'a::fs-name
assumes Trans: $P \longmapsto a[b] \prec P^{\prime}$
and $\quad \bigwedge a b P C . F C(a\{b\} . P) a b P$
and $\bigwedge P a b P^{\prime} c C . \llbracket P \longmapsto$ OutputR $a b \prec P^{\prime} ; \bigwedge C . F C P a b P \rrbracket \Longrightarrow F C$ $([c \frown c] P) a b P^{\prime}$
and $\wedge P a b P^{\prime} c d C . \llbracket P \longmapsto$ OutputR a $b \prec P^{\prime} ; \bigwedge C . F C P a b P^{\prime} ; c \neq d \rrbracket$ $\Longrightarrow F C([c \neq d] P) a b P^{\prime}$
and $\quad \bigwedge P a b P^{\prime} Q C . \llbracket P \longmapsto$ OutputR ab$b P^{\prime} ; \bigwedge C . F C P a b P^{\prime} \rrbracket \Longrightarrow F C$ $(P \oplus Q) a b P^{\prime}$
and $\Lambda Q$ ab $Q^{\prime} P C . \llbracket Q \longmapsto$ OutputR $a b \prec Q^{\prime} ; \Lambda C . F C Q a b Q^{\top} \rrbracket \Longrightarrow F$ $C(P \oplus Q) a b Q^{\prime}$
and $\quad \bigwedge P a b P^{\prime} Q C . \llbracket P \longmapsto$ OutputR ab$\prec P^{\prime} ; \bigwedge C . F C P a b P \rrbracket \Longrightarrow F C$ $(P \| Q) a b\left(P^{\prime} \| Q\right)$
and $\Lambda Q a b Q^{\prime} P C . \llbracket Q \longmapsto$ OutputR $a b \prec Q^{\prime} ; \bigwedge C . F C Q a b Q^{\prime} \rrbracket \Longrightarrow F$ $C(P \| Q) a b\left(P \| Q^{\prime}\right)$
and $\bigwedge P a b P^{\prime} x C . \llbracket P \longmapsto$ OutputR $a b \prec P^{\prime} ; x \neq a ; x \neq b ; x \sharp C ; \bigwedge C . F$ $C P a b P^{\prime} \rrbracket \Longrightarrow$

$$
F C(<\nu x>P) \text { ab }\left(<\nu x>P^{\prime}\right)
$$

and $\quad \wedge P a b P^{\prime} C . \llbracket P \|!P \longmapsto$ OutputR $a b \prec P^{\prime} ; \bigwedge C . F C(P \|!P) a b P \rrbracket$ $\Longrightarrow F C(!P) a b P^{\prime}$
shows $F C P a b P^{\prime}$
proof -
from Trans show ?thesis
$\mathbf{b y}$ (nominal-induct $x 2==$ OutputR $a b \prec P^{\prime}$ avoiding: $C$ arbitrary: $P^{\prime}$ rule: transitions.strong-induct, auto simp add: residualInject freeRes.inject intro: assms)
qed
lemma inputInduct[consumes 2, case-names Input Match Mismatch Sum1 Sum2 Par1 Par2 Res Bang]:
fixes $P$ :: $p i$
and $a$ :: name
and $x$ :: name
and $\quad P^{\prime}:: p i$
and $\quad F::\left({ }^{\prime} a:: f s\right.$-name $) \Rightarrow p i \Rightarrow$ name $\Rightarrow$ name $\Rightarrow p i \Rightarrow$ bool
and $C:: ' a:: f s$-name
assumes $a$ : $P \longmapsto a<x>\prec P^{\prime}$
and $\quad x \sharp P$
and cInput: $\bigwedge a x P C . F C(a<x>. P) a x P$
and cMatch: $\bigwedge P a x P^{\prime} b C . \llbracket P \longmapsto a<x>\prec P^{\prime} ; \bigwedge C . F C P a x P \rrbracket \Longrightarrow$ $F C([b \frown b] P) a x P^{\prime}$
and $\quad$ Mismatch: $\bigwedge P a x P^{\prime} b \quad c C . \llbracket P \longmapsto a<x>\prec P^{\prime} ; \bigwedge C . F C P a x P^{\prime} ; b$ $\neq c \rrbracket \Longrightarrow F C([b \neq c] P) a x P^{\prime}$
and cSum1: $\bigwedge P Q a x P^{\prime} C . \llbracket P \longmapsto a<x>\prec P^{\prime} ; \bigwedge C . F C P a x P \rrbracket \Longrightarrow$ $F C(P \oplus Q)$ a $x P^{\prime}$
and cSum2: $\quad \bigwedge P Q a x Q^{\prime} C . \llbracket Q \longmapsto a<x>\prec Q^{\prime} ; \bigwedge C . F C Q a x Q^{\prime} \rrbracket \Longrightarrow$ $F C(P \oplus Q)$ ax $Q^{\prime}$
and $\quad c \operatorname{Par} 1 B: \bigwedge P P^{\prime} Q$ a $x C . \llbracket P \longmapsto a<x>\prec P^{\prime} ; x \sharp P ; x \sharp Q ; x \neq a$; $\bigwedge C . F C P a x P \rrbracket \Longrightarrow$

$$
F C(P \| Q) a x\left(P^{\prime} \| Q\right)
$$

and $\quad c \operatorname{Par2B:} \wedge P Q Q^{\prime}$ ax $C . \llbracket Q \longmapsto a<x>\prec Q^{\prime} ; x \sharp P ; x \sharp Q ; x \neq a ;$ $\bigwedge C . F C Q a x Q \rrbracket \Longrightarrow$

$$
F C(P \| Q) \text { a } x\left(P \| Q^{\prime}\right)
$$

and $\quad c R e s B: ~ \bigwedge P P^{\prime}$ axy $C . \llbracket P \longmapsto a<x>\prec P^{\prime} ; y \neq a ; y \neq x ; y \sharp C$;
$\wedge C . F C P a x P \rrbracket \Longrightarrow F C(<\nu y>P)$ a $x\left(<\nu y>P^{\prime}\right)$
and $\quad c B a n g: \quad \bigwedge P a x P^{\prime} C . \llbracket P \|!P \longmapsto a<x>\prec P^{\prime} ; \bigwedge C . F C(P \|!P) a$
$x P \rrbracket \Longrightarrow$
$F C(!P) a x P^{\prime}$
shows $F C P a x P^{\prime}$
proof -
from $a\langle x \sharp P\rangle$ show ?thesis
proof (nominal-induct $x 2==a<x>\prec P^{\prime}$ avoiding: $C$ a $x P^{\prime}$ rule: transi-
tions.strong-induct)
case(Tau P)
thus ?case by (simp add: residualInject)
next
case(Input $x$ a $\left.P C a^{\prime} x^{\prime} P^{\prime}\right)$
have $x \sharp x^{\prime}$ by fact hence $x \neq x^{\prime}$ by simp
moreover have $a<x>\prec P=a^{\prime}<x^{\prime}>\prec P^{\prime}$ by fact
ultimately have $a e q a^{\prime}: a=a^{\prime}$ and $P e q P^{\prime}: P=\left[\left(x, x^{\prime}\right)\right] \cdot P^{\prime}$
by (simp add: residualInject freeRes.inject subject.inject name-abs-eq)+
have $F C\left(a<x^{\prime}>.\left(\left[\left(x, x^{\prime}\right)\right] \cdot P\right)\right) a x^{\prime}\left(\left[\left(x, x^{\prime}\right)\right] \cdot P\right) \mathbf{b y}($ rule cInput $)$
moreover have $x \sharp P^{\prime}$ by fact
ultimately show ? case using PeqP' aeqa' by (simp add: alphaInput name-swap)
next
case (Output Pab)
thus ?case by (simp add: residualInject)
next
case(Match P b Rs a $x$ )
thus ?case
by (force intro: cMatch simp add: residualInject)
next
case(Mismatch PRs a b Cax)
thus ?case
by(force intro: cMismatch simp add: residualInject)
next
case(Open $P P^{\prime}$ ab $\left.C a^{\prime} x P^{\prime}\right)$
thus ?case by (simp add: residualInject)
next
case(Sum1 P Q Rs C)
thus ?case by(force intro: cSum1)
next
case(Sum2 P Q Rs C)
thus ?case by(force intro: cSum2)
next
case(Par1B P a $\left.x P^{\prime} Q C a^{\prime} x^{\prime} P^{\prime \prime}\right)$
have $x \sharp x^{\prime}$ by fact hence xineqx': $x \neq x^{\prime}$ by simp

```
moreover have Eq: \(a « x\) 》 \(\prec\left(P^{\prime} \| Q\right)=a^{\prime}<x^{\prime}>\prec P^{\prime \prime}\) by fact
hence aeqa': \(a=\) InputS \(a^{\prime}\) by (simp add: residualInject)
have \(x^{\prime} \sharp P \| Q\) by fact
hence \(x^{\prime} \sharp P\) and \(x^{\prime} \sharp Q\) by simp +
have \(P^{\prime \prime}\) eq: \(P^{\prime \prime}=\left(\left[\left(x, x^{\prime}\right)\right] \cdot P^{\prime}\right) \| Q\)
proof -
    from Eq xineqx \({ }^{\prime}\) have \(\left(P^{\prime} \| Q\right)=\left[\left(x, x^{\prime}\right)\right] \cdot P^{\prime \prime}\)
        by (simp add: residualInject name-abs-eq)
    hence \(\left(\left[\left(x, x^{\prime}\right)\right] \cdot\left(P^{\prime} \| Q\right)\right)=P^{\prime \prime}\) by simp
    with \(\left\langle x^{\prime} \sharp Q\right\rangle\langle x \sharp Q\rangle\) show ?thesis by(simp add: name-fresh-fresh)
qed
have \(x \sharp P^{\prime \prime}\) by fact
with \(\left.P^{\prime \prime} e q « x \neq x^{\prime}\right\rangle\) have \(x^{\prime} \sharp P^{\prime}\) by (simp add: name-fresh-left name-calc)
have PTrans: \(P \longmapsto a « x » \prec P^{\prime}\) by fact
with \(\left\langle x^{\prime} \sharp P^{\prime}\right\rangle\) aeqa' have \(P \longmapsto a^{\prime}<x^{\prime}>\prec\left(\left[\left(x, x^{\prime}\right)\right] \cdot P^{\prime}\right)\)
    by (simp add: alphaBoundResidual)
moreover have \(\bigwedge C . F C P a^{\prime} x^{\prime}\left(\left[\left(x, x^{\prime}\right)\right] \cdot P^{\prime}\right)\)
proof -
    fix \(C\)
    have \(\bigwedge C a^{\prime} x^{\prime} P^{\prime \prime} . \llbracket a « x » \prec P^{\prime}=a^{\prime}<x^{\prime}>\prec P^{\prime \prime} ; x^{\prime} \sharp P \rrbracket \Longrightarrow F C P a^{\prime} x^{\prime} P^{\prime \prime}\)
by fact
    moreover with aeqa \({ }^{\prime}\) xineqx \({ }^{\prime}\left\langle x^{\prime} \sharp P^{\prime}\right\rangle\) have \(\left.a « x » \prec P^{\prime}=a^{\prime}<x^{\prime}\right\rangle \prec([(x\),
\(\left.\left.\left.x^{\prime}\right)\right] \cdot P^{\prime}\right)\)
            by (simp add: residualInject name-abs-eq name-fresh-left name-calc)
            ultimately show \(F C P a^{\prime} x^{\prime}\left(\left[\left(x, x^{\prime}\right)\right] \cdot P^{\prime}\right)\) using \(\left\langle x^{\prime} \sharp P\right\rangle\) by blast
    qed
    moreover from PTrans \(\left\langle x^{\prime} \sharp P>\right.\) have \(x^{\prime} \sharp a \operatorname{by}(\) auto dest: freshBoundDeriva-
tive)
    ultimately have \(F C(P \| Q) a^{\prime} x^{\prime}\left(\left(\left[\left(x, x^{\prime}\right)\right] \cdot P^{\prime}\right) \| Q\right)\) using \(\left\langle x^{\prime} \sharp Q\right\rangle a e q a^{\prime}\)
\(\left\langle x^{\prime} \sharp P\right\rangle\)
            by(rule-tac cPar1B) auto
    with \(P^{\prime \prime} e q\) show ? case by simp
next
    case (Par1F P \(\left.P^{\prime} Q \alpha\right)\)
    thus ?case by (simp add: residualInject)
next
    case(Par2B \(Q\) a \(\left.x Q^{\prime} P C a^{\prime} x^{\prime} Q^{\prime \prime}\right)\)
    have \(x \sharp x^{\prime}\) by fact hence xineqx \({ }^{\prime}: x \neq x^{\prime}\) by simp
    moreover have \(E q\) : \(a « x » \prec\left(P \| Q^{\prime}\right)=a^{\prime}<x^{\prime}>\prec Q^{\prime \prime}\) by fact
    hence aeqa': a = InputS \(a^{\prime}\) by (simp add: residualInject)
    have \(x \sharp P\) by fact
    have \(x^{\prime} \sharp P \| Q\) by fact
    hence \(x^{\prime} \sharp P\) and \(x^{\prime} \sharp Q\) by simp +
    have \(Q^{\prime \prime} e q: Q^{\prime \prime}=P \|\left(\left[\left(x, x^{\prime}\right)\right] \cdot Q^{\prime}\right)\)
    proof -
    from \(E q\) xineqx \({ }^{\prime}\) have \(\left(P \| Q^{\prime}\right)=\left[\left(x, x^{\prime}\right)\right] \cdot Q^{\prime \prime}\)
        by (simp add: residualInject name-abs-eq)
```

```
    hence}([(x,\mp@subsup{x}{}{\prime})]\cdot(P|\mp@subsup{Q}{}{\prime}))=\mp@subsup{Q}{}{\prime\prime}\mathrm{ by simp
    with \langlex'\sharpP\rangle\langlex\sharpP\rangle}\mathrm{ show ?thesis by(simp add: name-fresh-fresh)
    qed
    have }x\sharp\mp@subsup{Q}{}{\prime\prime}\mathrm{ by fact
    with }\mp@subsup{Q}{}{\prime\prime}eq«x\not=\mp@subsup{x}{}{\prime}>\mathrm{ have }\mp@subsup{x}{}{\prime}\sharp\mp@subsup{Q}{}{\prime}\mathrm{ by(simp add: name-fresh-left name-calc)
    have QTrans: }Q\longmapstoa<x>\prec Q ' by fac
    with\langle\mp@subsup{x}{}{\prime}\sharp\mp@subsup{Q}{}{\prime}\rangle aeq\mp@subsup{a}{}{\prime}}\mathrm{ have }Q\longmapsto\mp@subsup{a}{}{\prime}<\mp@subsup{x}{}{\prime}>\prec \prec([(x,\mp@subsup{x}{}{\prime})]\cdot\mp@subsup{Q}{}{\prime}
    by(simp add: alphaBoundResidual)
    moreover have \C.F CQ a' x'([(x, x')] • Q')
    proof -
        fix C
```



```
Q" by fact
```



```
x})]\cdot\mp@subsup{Q}{}{\prime}
            by(simp add: residualInject name-abs-eq name-fresh-left name-calc)
            ultimately show FCQ a' x'([(x, x')] \cdot Q') using <x'\sharp # Q>aeqa' by blast
    qed
    moreover from QTrans \langle\mp@subsup{x}{}{\prime}\sharpQ> have \mp@subsup{x}{}{\prime}\sharpa by(force dest: freshBoundDeriva-
tive)
```



```
<x'# Q>
            by(rule-tac cPar2B) auto
    with }\mp@subsup{Q}{}{\prime\prime}eq\mathrm{ show ?case by simp
next
    case(Par2F P P' Q \alpha)
    thus ?case by(simp add: residualInject)
next
    case(Comm1 P P'Q Q' a b x)
    thus ?case by(simp add: residualInject)
next
    case(Comm2 P P' Q Q' a b x)
    thus ?case by(simp add: residualInject)
next
    case(Close1 P P' Q Q' a x y)
    thus ?case by(simp add: residualInject)
next
    case(Close2 P P'Q Q' a x y)
    thus ?case by(simp add: residualInject)
next
    case(ResB P a x P' y C a' (' P'\prime)
    have }x\sharp\mp@subsup{x}{}{\prime}\mathrm{ by fact hence xineqx': x }\not=\mp@subsup{x}{}{\prime}\mathrm{ by simp
    moreover have Eq:a«x> \prec (<\nuy>> ')}=\mp@subsup{a}{}{\prime}<\mp@subsup{x}{}{\prime}>\prec\mp@subsup{P}{}{\prime\prime}\mathrm{ by fact
    hence aeqa': a = InputS a' by(simp add: residualInject)
    have }y\sharp\mp@subsup{x}{}{\prime}\mathrm{ by fact hence yineqx': }y\not=\mp@subsup{x}{}{\prime}\mathrm{ by simp
    moreover have }\mp@subsup{x}{}{\prime}\sharp<\nuy>P\mathrm{ by fact
    ultimately have }\mp@subsup{x}{}{\prime}\sharpP\mathrm{ by(simp add: name-fresh-abs)
```

have $y \neq x$ and yineqa: $y \sharp a$ and $y$ Fresh $C: y \sharp C$ by fact +

```
have \(P^{\prime \prime}\) eq: \(P^{\prime \prime}=<\nu y>\left(\left[\left(x, x^{\prime}\right)\right] \cdot P^{\prime}\right)\)
proof -
    from \(E q\) xineqx \({ }^{\prime}\) have \(<\nu y>P^{\prime}=\left[\left(x, x^{\prime}\right)\right] \cdot P^{\prime \prime}\)
            by (simp add: residualInject name-abs-eq)
    hence \(\left(\left[\left(x, x^{\prime}\right)\right] \cdot\left(<\nu y>P^{\prime}\right)\right)=P^{\prime \prime}\) by simp
    with yineqx \({ }^{\prime}\langle y \neq x\rangle\) show ?thesis by (simp add: name-fresh-fresh)
qed
```

have $x \sharp P^{\prime \prime}$ by fact
with $P^{\prime \prime}$ eq $\langle y \neq x\rangle\left\langle x \neq x^{\prime}\right\rangle$ have $x^{\prime} \sharp P^{\prime} \mathbf{b y}($ simp add: name-fresh-left name-calc name-fresh-abs)
have $P \longmapsto a « x$ » $\prec P^{\prime}$ by fact
with $\left\langle x^{\prime} \sharp P^{\prime}\right\rangle$ aeqa' have $P \longmapsto a^{\prime}<x^{\prime}>\prec\left(\left[\left(x, x^{\prime}\right)\right] \cdot P^{\prime}\right)$
by (simp add: alphaBoundResidual)
moreover have $\bigwedge C . F C P a^{\prime} x^{\prime}\left(\left[\left(x, x^{\prime}\right)\right] \cdot P^{\prime}\right)$
proof -
fix $C$
have $\bigwedge C a^{\prime} x^{\prime} P^{\prime \prime} . \llbracket a « x » \prec P^{\prime}=a^{\prime}<x^{\prime}>\prec P^{\prime \prime} ; x^{\prime} \sharp P \rrbracket \Longrightarrow F C P a^{\prime} x^{\prime} P^{\prime \prime}$
by fact
moreover with aeqa ${ }^{\prime}$ xineqx ${ }^{\prime}\left\langle x^{\prime} \sharp P^{\prime}\right\rangle$ have $\left.a « x » \prec P^{\prime}=a^{\prime}<x^{\prime}\right\rangle \prec([(x$, $\left.\left.\left.x^{\prime}\right)\right] \cdot P^{\prime}\right)$
by (simp add: residualInject name-abs-eq name-fresh-left name-calc)
ultimately show $F C P a^{\prime} x^{\prime}\left(\left[\left(x, x^{\prime}\right)\right] \cdot P^{\prime}\right)$ using $\left\langle x^{\prime} \sharp P\right\rangle$ aeqa' by blast
qed
ultimately have $F C(<\nu y>P) a^{\prime} x^{\prime}\left(<\nu y>\left(\left[\left(x, x^{\prime}\right)\right] \cdot P^{\prime}\right)\right)$ using yineqx ${ }^{\prime}$ yineqa yFresh $C$ aeqa'
by (force intro: cResB)
with $P^{\prime \prime} e q$ show ? case by simp
next
case (ResF P $\left.P^{\prime} \alpha y\right)$
thus ?case by (simp add: residualInject)
next case(Bang P Rs)
thus ?case by(force intro: cBang)
qed
qed
lemma boundOutputInduct[consumes 2, case-names Match Mismatch Open Sum1 Sum2 Par1 Par2 Res Bang]:
fixes $P:: p i$
and $a$ :: name
and $x$ :: name
and $P^{\prime}:: p i$
and $\quad F::\left({ }^{\prime} a:: f s\right.$-name $) \Rightarrow p i \Rightarrow$ name $\Rightarrow$ name $\Rightarrow p i \Rightarrow$ bool
and $C$ :: 'a::fs-name

```
    assumes \(a: P \longmapsto a<\nu x>\prec P^{\prime}\)
    and \(\quad x \sharp P\)
    and cMatch: \(\bigwedge P a x P^{\prime} b C . \llbracket P \longmapsto a<\nu x>\prec P^{\prime} ; \bigwedge C . F C P a x P^{\prime} \rrbracket \Longrightarrow\)
\(F C([b \frown b] P)\) a \(x P^{\prime}\)
    and \(\quad\) Mismatch: \(\bigwedge P a x P^{\prime} b\) c \(C . \llbracket P \longmapsto a<\nu x>\prec P^{\prime} ; \bigwedge C . F C P a x P^{\prime} ;\)
\(b \neq c \rrbracket \Longrightarrow F C([b \neq c] P)\) a \(x P^{\prime}\)
    and cOpen: \(\quad \bigwedge P a x P^{\prime} C . \llbracket P \longmapsto(\) OutputR \(a x) \prec P^{\prime} ; a \neq x \rrbracket \Longrightarrow F C\)
\((<\nu x>P) a x P^{\prime}\)
    and \(\quad c S u m 1: \quad \bigwedge P Q a x P^{\prime} C . \llbracket P \longmapsto a<\nu x>\prec P^{\prime} ; \bigwedge C . F C P a x P \rrbracket\)
\(\Longrightarrow F C(P \oplus Q)\) ax \(P^{\prime}\)
    and cSum2: \(\bigwedge P Q\) ax \(Q^{\prime} C . \llbracket Q \longmapsto a<\nu x>\prec Q^{\prime} ; \bigwedge C . F C Q\) a \(x Q^{\prime} \rrbracket\)
\(\Longrightarrow F C(P \oplus Q)\) ax \(Q^{\prime}\)
    and \(\quad c\) Par1B: \(\quad \bigwedge P P^{\prime} Q a x C . \llbracket P \longmapsto a<\nu x>\prec P^{\prime} ; x \sharp Q ; \bigwedge C . F C P a x\)
\(P^{\dagger} \Longrightarrow\)
                                    \(F C(P \| Q) a x\left(P^{\prime} \| Q\right)\)
    and \(\quad c\) Par2B: \(\quad \bigwedge P Q Q^{\prime} a x C . \llbracket Q \longmapsto a<\nu x>\prec Q^{\prime} ; x \sharp P ; \wedge C . F C Q a\)
\(Q^{\dagger} \rrbracket\)
                                    \(F C(P \| Q) a x\left(P \| Q^{\prime}\right)\)
    and \(\quad c R e s B: \quad \bigwedge P P^{\prime}\) a x y \(C . \llbracket P \longmapsto a<\nu x>\prec P^{\prime} ; y \neq a ; y \neq x ; y \sharp C\);
                    \(\wedge C . F C P\) ax \(P^{\prime} \rrbracket \Longrightarrow F C(<\nu y>P) a x\left(<\nu y>P^{\prime}\right)\)
    and \(\quad c B a n g: \quad \bigwedge P a x P^{\prime} C . \llbracket P \|!P \longmapsto a<\nu x>\prec P^{\prime} ; \bigwedge C . F C(P \|!P) a\)
\(x P^{\rrbracket} \Longrightarrow\)
                                    \(F C(!P) a x P^{\prime}\)
    shows \(F C P a x P^{\prime}\)
proof -
    from \(a\langle x \sharp P\rangle\) show ?thesis
    proof(nominal-induct \(x 2==a<\nu x>\prec P^{\prime}\) avoiding: C \(a\) a \(x P^{\prime}\) rule: transi-
tions.strong-induct)
    case (Tau P)
    thus? case \(\mathbf{b y}\) (simp add: residualInject)
    next
    case(Input Pax)
    thus ?case by(simp add: residualInject)
    next
    case( Output P a b)
    thus? case \(\mathbf{b y}\) (simp add: residualInject)
    next
    case(Match P Rs b C a x)
    thus ?case
        by (force intro: cMatch simp add: residualInject)
    next
    case(Mismatch PRs a b C c x)
    thus ?case
        by (force intro: cMismatch simp add: residualInject)
    next
    case(Sum1 P Q Rs C)
    thus ?case by(force intro: cSum1)
    next
    case(Sum2 P Q Rs C)
```

```
    thus ?case by(force intro:cSum2)
next
    case(Open P a b P' C a' x P '')
    have }b\sharpx\mathrm{ by fact hence bineqx: b}\not=x\mathrm{ by simp
    moreover have }a<\nub>\prec\mp@subsup{P}{}{\prime}=\mp@subsup{a}{}{\prime}<\nux>\prec\prec\mp@subsup{P}{}{\prime\prime}\mathrm{ by fact
    ultimately have aeq\mp@subsup{a}{}{\prime}:a=\mp@subsup{a}{}{\prime}\mathrm{ and }\mp@subsup{P}{}{\prime}eq\mp@subsup{P}{}{\prime\prime}:\mp@subsup{P}{}{\prime\prime}=[(b,x)]\cdot\mp@subsup{P}{}{\prime}
        by(simp add: residualInject name-abs-eq)+
    have }x\sharp<\nub>P\mathrm{ by fact
    with bineqx have }x\sharpP\mathrm{ by(simp add: name-fresh-abs)
    have aineqb: }a\not=b\mathrm{ by fact
    have PTrans: P\longmapstoa[b]\prec P' by fact
    with }\langlex\sharpP\rangle\mathrm{ have xineqa: }x\not=a\mathrm{ by(force dest: freshFreeDerivative)
    from PTrans have ([(b,x)] • P)\longmapsto[(b,x)] • (a[b]\prec \prec P') by(rule transi-
tions.eqvt)
    with }\mp@subsup{P}{}{\prime}eq\mp@subsup{P}{}{\prime\prime}\mathrm{ xineqa aineqb have Trans: ([(b,x)]•P)طa[x]ఒ 年
        by(auto simp add: name-calc)
    hence FC(<\nux>([(b,x)] • P)) a x P '' using xineqa by(blast intro:cOpen)
    with \langlex\sharpP\rangle aeqa' show ?case by(simp add: alphaRes)
next
    case(Par1B P a x P' Q Ca' ( 
    have }x\sharp\mp@subsup{x}{}{\prime}\mathrm{ by fact hence xineqx': x }=\mp@subsup{x}{}{\prime}\mathrm{ by simp
    moreover have Eq:a«x>\prec ( P'| Q)=\mp@subsup{a}{}{\prime}<\nu\mp@subsup{x}{}{\prime}>\prec\mp@subsup{P}{}{\prime\prime}\mathrm{ by fact}
    hence aeqa': a = BoundOutputS a' by(simp add: residualInject)
    have }x\sharpQ\mathrm{ by fact
    have }\mp@subsup{x}{}{\prime}\sharpP|Q\mathrm{ by fact
    hence }\mp@subsup{x}{}{\prime}\sharpP\mathrm{ and }\mp@subsup{x}{}{\prime}\sharpQ\mathrm{ by simp+
    have }\mp@subsup{P}{}{\prime\prime}eq:\mp@subsup{P}{}{\prime\prime}=([(x,\mp@subsup{x}{}{\prime})]\cdot\mp@subsup{P}{}{\prime})|
    proof -
        from Eq xineqx' have ( }\mp@subsup{P}{}{\prime}|Q)=[(x,x)]\cdot\mp@subsup{P}{}{\prime\prime
            by(simp add: residualInject name-abs-eq)
        hence ([(x, x})]\cdot(\mp@subsup{P}{}{\prime}|Q))=\mp@subsup{P}{}{\prime\prime}\mathrm{ by simp
        with \langlex'\sharp Q>\langlex\sharpQ> show ?thesis by(simp add: name-fresh-fresh)
    qed
    have }x\sharp\mp@subsup{P}{}{\prime\prime}\mathrm{ by fact
    with }\mp@subsup{P}{}{\prime\prime}eq\langlex\not=\mp@subsup{x}{}{\prime}\rangle\mathrm{ have }\mp@subsup{x}{}{\prime}\sharp\mp@subsup{P}{}{\prime}\mathrm{ by(simp add: name-fresh-left name-calc)
    have }P\longmapstoa<x>>>\mp@subsup{P}{}{\prime}\mathrm{ by fact
    with \langle\mp@subsup{x}{}{\prime}#\mp@subsup{P}{}{\prime}\rangle\mathrm{ aeqa' have }P\longmapsto\mp@subsup{a}{}{\prime}<\nu\mp@subsup{x}{}{\prime}>\prec \prec([(x, x')] \cdot P')
        by(simp add: alphaBoundResidual)
    moreover have \C.F CP a' x'([(x, x')] • P')
    proof -
    fix C
```



```
P'\prime}\mathrm{ by fact
    moreover with aeqa' xineq\mp@subsup{x}{}{\prime}\langle\mp@subsup{x}{}{\prime}\sharp\mp@subsup{P}{}{\prime}\rangle\mathrm{ have }a<x>><\mp@subsup{P}{}{\prime}=\mp@subsup{a}{}{\prime}<\nu\mp@subsup{x}{}{\prime}>\prec([(x,
x})]\cdot\mp@subsup{P}{}{\prime}
        by(simp add: residualInject name-abs-eq name-fresh-left name-calc)
```

ultimately show $F C P a^{\prime} x^{\prime}\left(\left[\left(x, x^{\prime}\right)\right] \cdot P^{\prime}\right)$ using $\left\langle x^{\prime} \sharp P\right\rangle$ aeqa' by blast qed
ultimately have $F C(P \| Q) a^{\prime} x^{\prime}\left(\left(\left[\left(x, x^{\prime}\right)\right] \cdot P^{\prime}\right) \| Q\right)$ using $\left\langle x^{\prime} \sharp Q>a e q a^{\prime}\right.$ by(blast intro: cPar1B)
with $P^{\prime \prime} e q$ show ? case by simp
next
case (Par1F P $\left.P^{\prime} Q \alpha\right)$
thus ?case by (simp add: residualInject)
next
case (Par2B $Q$ a x $\left.Q^{\prime} P C a^{\prime} x^{\prime} Q^{\prime \prime}\right)$
have $x \sharp x^{\prime}$ by fact hence xineqx': $x \neq x^{\prime}$ by simp
moreover have $E q: a « x » \prec\left(P \| Q^{\prime}\right)=a^{\prime}<\nu x^{\prime}>\prec Q^{\prime \prime}$ by fact
hence aeqa': $a=$ BoundOutputS $a^{\prime}$ by (simp add: residualInject)
have $x \sharp P$ by fact
have $x^{\prime} \sharp P \| Q$ by fact
hence $x^{\prime} \sharp P$ and $x^{\prime} \sharp Q$ by simp +
have $Q^{\prime \prime}$ eq: $Q^{\prime \prime}=P \|\left(\left[\left(x, x^{\prime}\right)\right] \cdot Q^{\prime}\right)$
proof -
from Eq xineqx ${ }^{\prime}$ have $\left(P \| Q^{\prime}\right)=\left[\left(x, x^{\prime}\right)\right] \cdot Q^{\prime \prime}$
by (simp add: residualInject name-abs-eq)
hence $\left(\left[\left(x, x^{\prime}\right)\right] \cdot\left(P \| Q^{\prime}\right)\right)=Q^{\prime \prime}$ by simp
with $\left\langle x^{\prime} \sharp P\right\rangle\langle x \sharp P\rangle$ show ?thesis $\mathbf{b y}$ (simp add: name-fresh-fresh)
qed
have $x \sharp Q^{\prime \prime}$ by fact
with $Q^{\prime \prime}$ eq $\left\langle x \neq x^{\prime}\right\rangle$ have $x^{\prime} \sharp Q^{\prime}$ by (simp add: name-fresh-left name-calc)
have $Q \longmapsto a « x\rangle \prec Q^{\prime}$ by fact
with $\left\langle x^{\prime} \sharp Q^{\prime}\right\rangle$ aeqa ${ }^{\prime}$ have $Q \longmapsto a^{\prime}<\nu x^{\prime}>\prec\left(\left[\left(x, x^{\prime}\right)\right] \cdot Q^{\prime}\right)$
by (simp add: alphaBoundResidual)
moreover have $\bigwedge C . F C Q a^{\prime} x^{\prime}\left(\left[\left(x, x^{\prime}\right)\right] \cdot Q^{\prime}\right)$
proof -
fix $C$
have $\bigwedge C a^{\prime} x^{\prime} Q^{\prime \prime} . \llbracket a « x » \prec Q^{\prime}=a^{\prime}<\nu x^{\prime}>\prec Q^{\prime \prime} ; x^{\prime} \sharp Q \rrbracket \Longrightarrow F C Q a^{\prime} x^{\prime}$
$Q^{\prime \prime}$ by fact
moreover with aeqa' xineqx ${ }^{\prime}\left\langle x^{\prime} \sharp Q^{\prime}\right\rangle$ have $\left.a « x\right\rangle \prec Q^{\prime}=a^{\prime}<\nu x^{\prime}>\prec([(x$, $\left.\left.\left.x^{\prime}\right)\right] \cdot Q^{\prime}\right)$
by (simp add: residualInject name-abs-eq name-fresh-left name-calc)
ultimately show $F C Q a^{\prime} x^{\prime}\left(\left[\left(x, x^{\prime}\right)\right] \cdot Q^{\prime}\right)$ using $\left\langle x^{\prime} \sharp Q\right\rangle a e q a^{\prime}$ by blast
qed
ultimately have $F C(P \| Q) a^{\prime} x^{\prime}\left(P \|\left(\left[\left(x, x^{\prime}\right)\right] \cdot Q^{\prime}\right)\right)$ using $\left\langle x^{\prime} \sharp P\right\rangle$ by(blast intro: cPar2B)
with $Q^{\prime \prime} e q$ show ?case by simp
next
case (Par2F P $P^{\prime} Q \alpha$ )
thus ?case by (simp add: residualInject)
next
case(Comm1 P $P^{\prime} Q Q^{\prime}$ a b $x$ )
thus ?case by(simp add: residualInject)

```
next
    case(Comm2 P P'Q Q' a b x)
    thus ?case by(simp add: residualInject)
next
    case(Close1 P P'Q Q' a x y)
    thus ?case by(simp add: residualInject)
next
    case(Close2 P P' Q Q' a x y)
    thus ?case by(simp add: residualInject)
next
    case(ResB P a x P' y C a' ( }\mp@subsup{x}{}{\prime}\mp@subsup{P}{}{\prime\prime}
    have }x\sharp\mp@subsup{x}{}{\prime}\mathrm{ by fact hence xineqx': x }=\mp@subsup{x}{}{\prime}\mathrm{ by simp
    moreover have Eq: a«x» \prec (<\nuy> P')= a'<\nu\mp@subsup{x}{}{\prime}>}\prec\mp@subsup{P}{}{\prime\prime}\mathrm{ by fact
    hence aeqa': a = BoundOutputS a' by(simp add: residualInject)
    have }y\sharp\mp@subsup{x}{}{\prime}\mathrm{ by fact hence yineqx': }y\not=\mp@subsup{x}{}{\prime}\mathrm{ by simp
    moreover have }\mp@subsup{x}{}{\prime}\sharp<\nuy>P\mathrm{ by fact
    ultimately have }\mp@subsup{x}{}{\prime}\sharpP\mathrm{ by(simp add: name-fresh-abs)
    have }y\not=x\mathrm{ and }y\sharpa\mathrm{ and yFreshC: y }#C\mathrm{ by fact+
    have }\mp@subsup{P}{}{\prime\prime}\mathrm{ eq: }\mp@subsup{P}{}{\prime\prime}=<\nuy>([(x,\mp@subsup{x}{}{\prime})]\cdot\mp@subsup{P}{}{\prime}
    proof -
        from Eq xineqx' have <\nuy> P' = [(x, x')] \cdot P'\prime
            by(simp add: residualInject name-abs-eq)
        hence}([(x,\mp@subsup{x}{}{\prime})]\cdot(<\nuy>\mp@subsup{P}{}{\prime}))=\mp@subsup{P}{}{\prime\prime}\mathrm{ by simp
        with yineqx}\mp@subsup{}{}{\prime}\langley\not=x\rangle\mathrm{ show ?thesis by(simp add: name-fresh-fresh)
    qed
    have }x\sharp\mp@subsup{P}{}{\prime\prime}\mathrm{ by fact
    with }\mp@subsup{P}{}{\prime\prime}eq\langley\not=x\rangle\langlex\not=\mp@subsup{x}{}{\prime}\rangle\mathrm{ have }\mp@subsup{x}{}{\prime}\sharp\mp@subsup{P}{}{\prime}\mathbf{by}(\mathrm{ simp add: name-fresh-left name-calc
name-fresh-abs)
    have }P\longmapstoa<x>> \precP' by fac
    with \langle\mp@subsup{x}{}{\prime}\sharp\mp@subsup{P}{}{\prime}\rangle\mathrm{ aeqa' have }P\longmapsto\mp@subsup{a}{}{\prime}<\nu\mp@subsup{x}{}{\prime}>\prec \prec([(x, x')] \cdot P')
        by(simp add: alphaBoundResidual)
    moreover have \C.FCP a' x'([(x, x')] • P')
    proof -
        fix C
        have \C a' \mp@subsup{x}{}{\prime}\mp@subsup{P}{}{\prime\prime}.\llbracketa<x>\prec \prec P'= a'<\nu\mp@subsup{x}{}{\prime}>\prec\mp@subsup{P}{}{\prime\prime};\mp@subsup{x}{}{\prime}\sharpP\rrbracket\LongrightarrowFCP P a
P'\prime by fact
```



```
x')] • P')
            by(simp add: residualInject name-abs-eq name-fresh-left name-calc)
            ultimately show F CP Pa}\mp@subsup{x}{}{\prime}([(x,\mp@subsup{x}{}{\prime})]\cdot\mp@subsup{P}{}{\prime})\mathrm{ using <x'#P> aeqa' by blast
    qed
    ultimately have FC (<\nuy>P) a' \mp@subsup{x}{}{\prime}(<\nuy>([(x,\mp@subsup{x}{}{\prime})]\cdot\mp@subsup{P}{}{\prime}))\mathrm{ using yineqx}\mp@subsup{}{}{\prime}<y
#a> yFreshC aeqa'
            by(force intro: cResB)
    with P}\mp@subsup{P}{}{\prime\prime}eq\mathrm{ show ?case by simp
next
```

```
        case(ResF P P' \alpha y)
        thus ?case by(simp add: residualInject)
        next
        case(Bang P Rs)
        thus ?case by(force intro: cBang)
    qed
qed
lemma tauInduct[consumes 1, case-names Tau Match Mismatch Sum1 Sum2 Par1
Par2 Comm1 Comm2 Close1 Close2 Res Bang]:
    fixes P :: pi
    and }\mp@subsup{P}{}{\prime}::p
    and F}::\mp@subsup{'}{}{\prime}a::fs-name => pi=>pi=> boo
    and }C:::'a::fs-nam
    assumes Trans: P\longmapsto\tau\prec\mp@subsup{P}{}{\prime}
    and }\PC.FC(\tau.(P))
    and }\{P\mp@subsup{P}{}{\prime}cC.\llbracketP\longmapsto\tau\prec\mp@subsup{P}{}{\prime};\bigwedgeC.FCPP\\\LongrightarrowFC([c\frownc]P)\mp@subsup{P}{}{\prime
    and}\\P\mp@subsup{P}{}{\prime}cdC.\llbracketP\longmapsto\tau\prec\mp@subsup{P}{}{\prime};\bigwedgeC.FCPP\mp@subsup{P}{}{\prime};c\not=d\rrbracket\LongrightarrowFC([c\not=d]P
P'
    and}\quad\P\mp@subsup{P}{}{\prime}QC.\llbracketP\longmapsto\tau\prec\mp@subsup{P}{}{\prime};\bigwedgeC.FCPP\\rrbracket\LongrightarrowFC(P\oplusQ)\mp@subsup{P}{}{\prime
    and }\\Q\mp@subsup{Q}{}{\prime}PC.\llbracketQ\longmapsto\tau\prec\mp@subsup{Q}{}{\prime};\bigwedgeC.FCQQ\rrbracket\LongrightarrowFC(P\oplusQ)\mp@subsup{Q}{}{\prime
    and }\quad\P\mp@subsup{P}{}{\prime}QC.\llbracketP\longmapsto\tau\prec\mp@subsup{P}{}{\prime};\bigwedgeC.FCPP\\\LongrightarrowFC(P|Q)(\mp@subsup{P}{}{\prime}|Q
```



```
    and}\quad\LambdaPax\mp@subsup{P}{}{\prime}Qb\mp@subsup{Q}{}{\prime}C.\llbracketP\longmapsto(BoundR(InputS a) x P');Q\longmapstoOutput
    ab\prec Q'; x\sharpP;x\sharpQ;x\sharpC\rrbracket\LongrightarrowFC(P|Q) (P'[x::=b]|Q')
    and}\quad\Pab\mp@subsup{P}{}{\prime}Qx\mp@subsup{Q}{}{\prime}C.\llbracketP\longmapstoOutputR a b\prec 靘;Q\longmapsto(BoundR (InputS
a) }x\mp@subsup{Q}{}{\prime});x\sharpP;x\sharpQ;x\sharpC\rrbracket\LongrightarrowFC(P|Q)(\mp@subsup{P}{}{\prime}|\mp@subsup{Q}{}{\prime}[x::=b]
    and}\Pax\mp@subsup{P}{}{\prime}Qy \mp@subsup{Q}{}{\prime}C.\llbracketP\longmapsto(BoundR(InputS a) x P P');Q\longmapstoa<\nuy
\prec Q';x\sharpP;x\sharpQ;x\sharpC;y\sharpP;y\sharpQ;y\sharpC;x\not=y\rrbracket\LongrightarrowFC(P|Q)
(<\nuy>(\mp@subsup{P}{}{\prime}[x::=y]| Q ')
    and}\quad\Pay \mp@subsup{P}{}{\prime}Qx\mp@subsup{Q}{}{\prime}C.\llbracketP\longmapstoa<\nuy>\prec\mp@subsup{P}{}{\prime};Q\longmapsto(BoundR (InputS a) x
Q');x\sharpP;x\sharpQ;x\sharpC;y\sharpP;y\sharpQ;y\sharpC;x\not=y\rrbracket\LongrightarrowFC(P|Q)(<\nuy>(\mp@subsup{P}{}{\prime}
| Q ' [x::=y]))
    and}\bigwedge\P\mp@subsup{P}{}{\prime}xC.\llbracketP\longmapsto\tau\prec\mp@subsup{P}{}{\prime};x\sharpC;\bigwedgeC.FCPP午\
                        FC(<\nux>P)(<\nux>>P')
    and}\quad\bigwedgeP\mp@subsup{P}{}{\prime}C.\llbracketP|!P\longmapsto\tau\prec\mp@subsup{P}{}{\prime};\bigwedgeC.FC(P|!P)P\rrbracket\LongrightarrowFC(!P)\mp@subsup{P}{}{\prime
    shows FC P P '
proof -
    from Trans show ?thesis
    by(nominal-induct x2 = = \tau\prec P' avoiding:C arbitrary: }\mp@subsup{P}{}{\prime}\mathrm{ rule: transitions.strong-induct,
        auto simp add: residualInject intro: assms)
qed
inductive bangPred :: pi => pi => bool
where
    aux1: bangPred P (!P)
| aux2: bangPred P (P | !P)
```

inductive-cases nilCases'[simplified pi.distinct residual.distinct]: $\mathbf{0} \longmapsto R s$ inductive-cases tauCases' $[$ simplified pi.distinct residual.distinct] $: \tau .(P) \longmapsto R s$ inductive-cases inputCases'[simplified pi.inject residualInject]: $a<b>. P \longmapsto R s$ inductive-cases outputCases'[simplified pi.inject residualInject]: $a\{b\} . P \longmapsto R s$ inductive-cases matchCases' ${ }^{\prime}$ simplified pi.inject residualInject $]:[a \frown b] P \longmapsto R s$ inductive-cases mismatchCases'[simplified pi.inject residualInject]: $[a \neq b] P \longmapsto$ Rs
inductive-cases sumCases'[simplified pi.inject residualInject]: $P \oplus Q \longmapsto R s$ inductive-cases parCasesB'[simplified pi.distinct residual.distinct]: $P \| Q \longmapsto$ $b « y » \prec P^{\prime}$
inductive-cases parCasesF'[simplified pi.distinct residual.distinct]: $P \| Q \longmapsto \alpha$ $\prec P^{\prime}$
inductive-cases resCases'[simplified pi.distinct residual.distinct]: $\langle\nu x\rangle P \longmapsto R s$ inductive-cases resCases $B^{\prime}\left[\right.$ simplified pi.distinct residual.distinct]: $\left\langle\nu x^{\prime}\right\rangle P \longmapsto$ $a « y^{\prime} » \prec P^{\prime}$
inductive-cases resCases $F^{\prime}[$ simplified pi.distinct residual.distinct]: $\langle\nu x\rangle P \longmapsto \alpha$ $\prec P^{\prime}$
inductive-cases bangCases[simplified pi.distinct residual.distinct]: $!P \longmapsto R s$
lemma tauCases[consumes 1, case-names cTau]:
fixes $P$ :: $p i$
and $\alpha$ :: freeRes
and $\quad P^{\prime}:: p i$
assumes $\tau \cdot(P) \longmapsto \alpha \prec P^{\prime}$
and $\llbracket \alpha=\tau ; P=P \rrbracket \Longrightarrow \operatorname{Prop}(\tau) P$
shows Prop $\alpha P^{\prime}$
using assms
by (erule-tac tauCases', auto simp add: pi.inject residualInject)
lemma outputCases[consumes 1, case-names cOutput]:
fixes $a$ :: name
and $b$ :: name
and $P:: p i$
and $\quad \alpha$ :: freeRes
and $\quad P^{\prime}:: p i$
assumes $a\{b\} . P \longmapsto \alpha \prec P^{\prime}$
and $\llbracket \alpha=a[b] ; P=P \rrbracket \Longrightarrow \operatorname{Prop}(a[b]) P$
shows Prop $\alpha P^{\prime}$
using assms
by(erule-tac outputCases ${ }^{\prime}$, auto simp add: residualInject)
lemma zeroTrans[dest]:
fixes $R s$ :: residual

```
    assumes 0 \longmapsto
    shows False
using assms
by(induct rule: nilCases', auto)
lemma resZeroTrans[dest]:
    fixes x :: name
    and Rs :: residual
    assumes <\nux>0 \longmapsto Rs
    shows False
using assms
by(induct rule: resCases', auto simp add: pi.inject alpha')
lemma matchTrans[dest]:
    fixes a :: name
    and b :: name
    and }P\mathrm{ :: pi
    and Rs :: residual
    assumes [a\frownb]P\longmapstoRs
    and a\not=b
    shows False
using assms
by(induct rule: matchCases', auto)
lemma mismatchTrans[dest]:
    fixes a :: name
    and }P\mathrm{ :: pi
    and Rs :: residual
    assumes [a\not=a]P\longmapstoRs
    shows False
using assms
by(induct rule: mismatchCases', auto)
lemma inputCases[consumes 4, case-names cInput]:
    fixes a :: name
    and }x\mathrm{ :: name
    and }P:: p
    and }\mp@subsup{P}{}{\prime}::p
    assumes Input: }a<x>.P\longmapstob«y»\precyP
    and }y\not=
    and }y\not=
```

and $\quad y \sharp P$
and $\quad A: \quad \llbracket b=\operatorname{InputS} a ; y P^{\prime}=([(x, y)] \cdot P) \rrbracket \Longrightarrow \operatorname{Prop}($ InputS $a) y([(x$, $y)] \cdot P)$
shows Prop b y $y P^{\prime}$
proof -
note assms
moreover from Input $\langle y \neq a\rangle\langle y \neq x\rangle\langle y \sharp P\rangle$ have $y \sharp b$
by (force dest: freshBoundDerivative simp add: abs-fresh)
moreover obtain $z:: n a m e$ where $z \neq y$ and $z \neq x$ and $z \sharp P$ and $z \neq a$ and $z \sharp b$ and $z \sharp y P^{\prime}$
by (generate-fresh name, auto simp add: fresh-prod)
moreover obtain $z^{\prime}::$ name where $z^{\prime} \neq y$ and $z^{\prime} \neq x$ and $z^{\prime} \neq z$ and $z^{\prime} \sharp P$ and $z^{\prime} \neq a$ and $z^{\prime} \sharp b$ and $z^{\prime} \sharp y P^{\prime}$
$\mathbf{b y}$ (generate-fresh name, auto simp add: fresh-prod) ultimately show ?thesis
by (cases rule: transitions.strong-cases[where $x=y$ and $b=z$ and $x a=z$ and $x b=z$ and $x c=z$ and $x d=z$ and $x e=z$
and $x f=z$ and $x g=z$ and $y=z^{\prime}$ and $y a=z^{\prime}$
and $y b=y$ and $\left.\left.y c=z^{\prime}\right]\right)$
(auto simp add: pi.inject residualInject alpha abs-fresh fresh-prod fresh-left calc-atm)+
qed
lemma tauBoundTrans[dest]:
fixes $P:: p i$
and $a$ :: subject
and $x$ :: name
and $P^{\prime}:: p i$
assumes $\tau .(P) \longmapsto a « x » \prec P^{\prime}$
shows False
using assms
by $-\left(\right.$ ind-cases $\left.\tau .(P) \longmapsto a « x » \prec P^{\prime}\right)$
lemma tauOutputTrans[dest]:
fixes $P:: p i$
and $a$ :: name
and $b$ :: name
and $P^{\prime}:: p i$
assumes $\tau .(P) \longmapsto a[b] \prec P^{\prime}$
shows False
using assms
by $-\left(\right.$ ind-cases $\tau .(P) \longmapsto a[b] \prec P^{\prime}$, auto simp add: residualInject $)$
lemma inputFreeTrans[dest]:

```
    fixes a :: name
    and }x\mathrm{ :: name
    and }P::p
    and \alpha :: freeRes
    and }\mp@subsup{P}{}{\prime}:: p
    assumes }a<x>.P\longmapsto\alpha\prec\mp@subsup{P}{}{\prime
    shows False
using assms
by - (ind-cases }a<x>.P\longmapsto\alpha\prec\mp@subsup{P}{}{\prime}
lemma inputBoundOutputTrans[dest]:
    fixes a :: name
    and }x\mathrm{ :: name
    and }P:: p
    and b :: name
    and y :: name
    and }\mp@subsup{P}{}{\prime}::p
    assumes }a<x>.P\longmapstob<\nuy>\prec\mp@subsup{P}{}{\prime
    shows False
using assms
by - (ind-cases }a<x>.P\longmapstob<\nuy>\prec P', auto simp add: residualInject
lemma outputTauTrans[dest]:
    fixes a :: name
    and b :: name
    and }P::p
    and }\mp@subsup{P}{}{\prime}::p
    assumes }a{b}.P\longmapsto\tau\prec\mp@subsup{P}{}{\prime
    shows False
using assms
by - (ind-cases a{b}.P\longmapsto\tau\prec P', auto simp add: residualInject)
lemma outputBoundTrans[dest]:
    fixes a :: name
    and b :: name
    and }P::p
    and c :: subject
    and }x\mathrm{ :: name
    and }\mp@subsup{P}{}{\prime}:: p
    assumes }a{b}.P\longmapstoc«x»\prec\mp@subsup{P}{}{\prime
    shows False
```

```
using assms
by \(-\left(\right.\) ind-cases \(\left.a\{b\} . P \longmapsto c « x » \prec P^{\prime}\right)\)
lemma outputIneqTrans[dest]:
    fixes \(a\) :: name
    and \(b\) :: name
    and \(P\) :: pi
    and \(c\) :: name
    and \(d\) :: name
    and \(\quad P^{\prime}:: p i\)
    assumes \(a\{b\} . P \longmapsto c[d] \prec P^{\prime}\)
    and \(\quad a \neq c \vee b \neq d\)
    shows False
using assms
by \(-\left(\right.\) ind-cases \(a\{b\} . P \longmapsto c[d] \prec P^{\prime}\), auto simp add: residualInject pi.inject al-
pha')
lemma outputFresh Trans[dest]:
    fixes \(a\) :: name
    and \(b\) :: name
    and \(P\) :: pi
    and \(\alpha\) :: freeRes
    and \(P^{\prime}:: p i\)
    assumes \(a\{b\} . P \longmapsto \alpha \prec P^{\prime}\)
    and \(\quad a \sharp \alpha \vee b \sharp \alpha\)
    shows False
using assms
by \(-\left(\right.\) ind-cases \(a\{b\} . P \longmapsto \alpha \prec P^{\prime}\), auto simp add: residualInject pi.inject alpha')
lemma inputIneqTrans[dest]:
    fixes \(a\) :: name
    and \(x\) :: name
    and \(\quad P:: p i\)
    and \(b\) :: subject
    and \(y\) :: name
    and \(P^{\prime}:: p i\)
    assumes \(a<x>. P \longmapsto b « y » \prec P^{\prime}\)
    and \(\quad a \sharp b\)
    shows False
using assms
by \(-\left(\right.\) ind-cases \(a<x>. P \longmapsto b « y » \prec P^{\prime}\), auto simp add: residualInject pi.inject)
lemma resTauBoundTrans[dest]:
```

```
fixes x :: name
and }P\mathrm{ :: pi
and a :: subject
and y :: name
and }\mp@subsup{P}{}{\prime}:: p
assumes <\nux>\tau.(P)\longmapstoa«y»}\prec\mp@subsup{P}{}{\prime
shows False
using assms
by - (ind-cases <\nux>\tau.(P)\longmapstoa<y» \prec P', auto simp add: residualInject pi.inject
alpha')
lemma resTauOutputTrans[dest]:
fixes x :: name
and }P:: p
and a :: name
and b :: name
and }\mp@subsup{P}{}{\prime}::p
assumes <\nux>\tau. (P)\longmapstoa[b]\prec }\mp@subsup{P}{}{\prime
shows False
using assms
by - (ind-cases <\nux>\tau.(P)\longmapstoa[b]\prec 敖, auto simp add: residualInject pi.inject
alpha')
lemma resInputFreeTrans[dest]:
fixes }x\mathrm{ :: name
fixes a :: name
and y :: name
and }P::p
and \alpha :: freeRes
and }\mp@subsup{P}{}{\prime}:: p
assumes <\nux>a<y>.P\longmapsto\alpha\prec事
shows False
using assms
by - (ind-cases <\nux>a<y>.P\longmapsto\alpha\prec P', auto simp add: pi.inject residualInject
alpha')
lemma resInputBoundOutputTrans[dest]:
fixes }x\mathrm{ :: name
and a :: name
and y :: name
and }P::p
and b :: name
and z :: name
```

```
    and }\mp@subsup{P}{}{\prime}::p
    assumes <\nux>a<y>.P\longmapstob<\nuz> \prec P'
    shows False
using assms
by - (ind-cases }<\nux>a<y>.P\longmapstob<\nuz>< < P', auto simp add: pi.inject residu
alInject alpha')
lemma resOutputTauTrans[dest]:
    fixes }x\mathrm{ :: name
    and a :: name
    and b}::\mathrm{ name
    and }P::p
    and }\mp@subsup{P}{}{\prime}::p
    assumes <\nux>a{b}.P\longmapsto\tau\prec 
    shows False
using assms
by - (ind-cases <\nux>a{b}.P\longmapsto\tau\prec P', auto simp add: residualInject pi.inject
alpha')
lemma resOutputInputTrans[dest]:
fixes x :: name
and a :: name
and b :: name
and }P::: p
and c :: name
and y :: name
and }\mp@subsup{P}{}{\prime}::p
assumes <\nux>a{b}.P\longmapstoc<y>\prec \prec P'
    shows False
using assms
by - (ind-cases <\nux>a{b}.P\longmapstoc<y> \prec P', auto simp add: pi.inject residualIn-
ject alpha')
lemma resOutputOutputTrans[dest]:
    fixes x :: name
    and a :: name
    and }P::p
    and b :: name
    and y :: name
    and }\mp@subsup{P}{}{\prime}::p
    assumes <\nux>a{x}.P\longmapstob[y]\prec 㐌
```

```
    shows False
using assms
by - (ind-cases <\nux>a{x}.P\longmapstob[y]\prec P', auto simp add: pi.inject residualInject
alpha' calc-atm)
lemma resTrans[dest]:
    fixes x :: name
    and b :: name
    and Rs :: residual
    and y :: name
    shows <\nux>x{b}.P\longmapstoRs\Longrightarrow False
    and <\nux>x<y>.P\longmapstoRs\Longrightarrow False
apply(ind-cases <\nux>x{b}.P\longmapsto Rs, auto simp add: pi.inject alpha' calc-atm)
by(ind-cases <\nux>x<y>.P\longmapstoRs, auto simp add: pi.inject alpha' calc-atm abs-fresh
fresh-left)
lemma matchCases[consumes 1, case-names cMatch]:
    fixes a :: name
    and b :: name
    and }P::p
    and Rs :: residual
    and F :: name }=>\mathrm{ name }=>\mathrm{ bool
    assumes [a\frownb]P\longmapstoRs
    and }\quad\llbracketP\longmapstoRs;a=b\rrbracket\LongrightarrowFa
    shows F a b
using assms
by(induct rule: matchCases', auto)
lemma mismatchCases[consumes 1, case-names cMismatch]:
    fixes a :: name
    and }b\mathrm{ :: name
    and }P::p
    and Rs :: residual
    and }F\mathrm{ :: name }=>\mathrm{ name }=>\mathrm{ bool
    assumes Trans: [a\not=b]P\longmapstoRs
    and \quadcMatch: }\llbracketP\longmapstoRs;a\not=b\rrbracket\LongrightarrowFa
    shows Fab
using assms
by(induct rule: mismatchCases', auto)
lemma sumCases[consumes 1, case-names cSum1 cSum2]:
    fixes }P\mathrm{ :: pi
    and }Q :: p
    and Rs :: residual
```

```
    assumes Trans: P}\oplusQ\longmapstoR
    and cSum1: P\longmapstoRs\Longrightarrow Prop
    and cSum2: Q}\longmapsto>Rs\Longrightarrow Pro
    shows Prop
using assms
by(induct rule: sumCases', auto)
lemma name-abs-alpha:
    fixes a :: name
    and b:: name
    and }P::p
    assumes b\sharpP
    shows [a].P = [b].([(a,b)] \cdot P)
proof(cases a=b, auto)
    assume a\not=b
    with assms show ?thesis
    by(force intro: abs-fun-eq3[OF pt-name-inst, OF at-name-inst]
                simp add: name-swap name-calc name-fresh-left)
qed
lemma parCasesB[consumes 3, case-names cPar1 cPar2]:
    fixes P :: pi
    and }Q ::p
    and a :: subject
    and }x\mathrm{ :: name
    and }P\mp@subsup{Q}{}{\prime}::p
    and C ::' 'a::fs-name
    assumes }P||\longmapstoa«x»\precP\mp@subsup{Q}{}{\prime
    and }x\sharp
    and }x\sharp
    and}\quad\\mp@subsup{P}{}{\prime}.P\longmapstoa<x>\prec\mp@subsup{P}{}{\prime}\Longrightarrow\operatorname{Prop}(\mp@subsup{P}{}{\prime}|Q
    and}\\\mp@subsup{Q}{}{\prime}.Q\longmapstoa<x»\prec\mp@subsup{Q}{}{\prime}\Longrightarrow\operatorname{Prop}(P|\mp@subsup{Q}{}{\prime}
    shows Prop PQ'
proof -
    note assms
    moreover from <P| |\longmapstoa«x» \precPQ'>\langlex\sharpP\rangle\langlex\sharp Q \ have x\sharpa
        by(force dest: freshBoundDerivative)
    moreover obtain y::name where }y\not=x\mathrm{ and }y\sharpP\mathrm{ and }y\sharpQ\mathrm{ and }y\sharpa\mathrm{ and
y#P\mp@subsup{Q}{}{\prime}
    by(generate-fresh name, auto simp add: fresh-prod)
    moreover obtain z::name where z\not=y and z\not=x and z\sharpP and z\sharpQ and
z\sharpa}\mathrm{ and }z\sharpP\mp@subsup{Q}{}{\prime
    by(generate-fresh name, auto simp add: fresh-prod)
```


## ultimately show ?thesis

by (cases rule: transitions.strong-cases $[$ where $x=y$ and $b=y$ and $x a=x$ and $x b=x$ and $x c=y$ and $x d=y$ and $x e=y$
and $x f=y$ and $x g=y$ and $y=z$ and $y a=z$
and $y b=z$ and $y c=z])$
(auto simp add: pi.inject residualInject alpha abs-fresh fresh-prod)+ qed
lemma parCasesF[consumes 1, case-names cPar1 cPar2 cComm1 cComm2 cClose1 cClose2]:
fixes $P:: p i$
and $Q:: p i$
and $\alpha$ :: freeRes
and $\quad P^{\prime}:: p i$
and $C::{ }^{\prime} a:: f s$-name
and $\quad F::$ freeRes $\Rightarrow p i \Rightarrow$ bool
assumes Trans: $P \| Q \longmapsto \alpha \prec P Q^{\prime}$
and $\quad$ icPar1F: $\wedge P^{\prime} . \llbracket P \longmapsto \alpha \prec P^{\prime} \rrbracket \Longrightarrow F \alpha\left(P^{\prime} \| Q\right)$
and $\quad$ icPar2F: $\bigwedge Q^{\prime} . \llbracket Q \longmapsto \alpha \prec Q^{\prime} \rrbracket \Longrightarrow F \alpha\left(P \| Q^{\prime}\right)$
and $\quad i c \operatorname{Comm1}: \bigwedge P^{\prime} Q^{\prime}$ abs. $\llbracket P \longmapsto a<x>\prec P^{\prime} ; Q \longmapsto a[b] \prec Q^{\prime} ; x \sharp P$; $x \sharp Q ; x \neq a ; x \neq b ; x \sharp Q^{\prime} ; x \sharp C ; \alpha=\tau \rrbracket \Longrightarrow F(\tau)\left(P^{\prime}[x::=b] \| Q^{\prime}\right)$
and $\quad i c C o m m 2: ~ \wedge P^{\prime} Q^{\prime}$ abx. $\llbracket P \longmapsto a[b] \prec P^{\prime} ; Q \longmapsto a<x>\prec Q^{\prime} ; x \sharp P$; $x \sharp Q ; x \neq a ; x \neq b ; x \sharp P^{\prime} ; x \sharp C ; \alpha=\tau \rrbracket \Longrightarrow F(\tau)\left(P^{\prime} \| Q^{\prime}[x::=b]\right)$
and $\quad$ icClose1: $\bigwedge P^{\prime} Q^{\prime}$ a $x y . \llbracket P \longmapsto a<x>\prec P^{\prime} ; Q \longmapsto a<\nu y>\prec Q^{\prime} ; x \sharp$ $P ; x \sharp Q ; x \neq a ; x \neq y ; x \sharp Q^{\prime} ; y \sharp P ; y \sharp Q ; y \neq a ; y \sharp P^{\prime} ; x \sharp C ; y \sharp C ; \alpha=\tau \rrbracket$ $\Longrightarrow$

$$
F(\tau)\left(<\nu y>\left(P^{\prime}[x::=y] \| Q^{\prime}\right)\right)
$$

and $\quad i c$ Close2: $\wedge P^{\prime} Q^{\prime}$ axy. $\llbracket P \longmapsto a<\nu y>\prec P^{\prime} ; Q \longmapsto a<x>\prec Q^{\prime} ; x \sharp$ $P ; x \sharp Q ; x \neq a ; x \neq y ; x \sharp P^{\prime} ; y \sharp P ; y \sharp Q ; y \neq a ; y \sharp Q^{\prime} ; x \sharp C ; y \sharp C ; \alpha=\tau \rrbracket$ $\Longrightarrow$

$$
F(\tau)\left(<\nu y>\left(P^{\prime} \| Q^{\prime}[x::=y]\right)\right)
$$

shows $F \propto P Q^{\prime}$
proof -
note assms
moreover obtain $x$ :: name where $x \sharp P$ and $x \sharp Q$ and $x \sharp \alpha$ and $x \sharp P Q^{\prime}$ and $x \sharp C$
by (generate-fresh name, auto simp add: fresh-prod)
moreover obtain $y$ ::name where $y \sharp P$ and $y \sharp Q$ and $y \sharp \alpha$ and $y \sharp P Q^{\prime}$ and $y \sharp C$ and $x \neq y$
by (generate-fresh name, auto simp add: fresh-prod)
ultimately show ?thesis
by (cases rule: transitions.strong-cases[where $x=x$ and $b=x$ and $x a=x$ and $x b=x$ and $x c=x$ and $x d=x$ and $x e=x$
and $x f=x$ and $x g=x$ and $y=y$ and $y a=y$
and $y b=y$ and $y c=y]$ )
(auto simp add: pi.inject residualInject alpha abs-fresh fresh-prod)+

## qed

lemma resCases $F[$ consumes 1 , case-names cRes]:
fixes $x$ :: name
and $P:: p i$
and $\alpha$ :: freeRes
and $P^{\prime}:: p i$
and $C$ :: 'a::fs-name
assumes $<\nu x>P \longmapsto \alpha \prec x P^{\prime}$
and $\quad \backslash P^{\prime} . \llbracket P \longmapsto \alpha \prec P^{\prime} ; x \sharp \alpha \rrbracket \Longrightarrow F\left(<\nu x>P^{\prime}\right)$
shows $F x P^{\prime}$
proof -
note assms
moreover from $\left\langle<\nu x>P \longmapsto \alpha \prec x P^{\prime}\right.$ 〉 have $x \sharp \alpha$ and $x \sharp x P^{\prime}$
by (force dest: freshFreeDerivative simp add: abs-fresh)+
moreover obtain $y:$ :name where $y \neq x$ and $y \sharp P$ and $y \sharp \alpha$ and $y \sharp x P^{\prime}$
by (generate-fresh name, auto simp add: fresh-prod)
moreover obtain $z:: n a m e$ where $z \neq y$ and $z \neq x$ and $z \sharp P$ and $z \sharp \alpha$ and $z \sharp x P^{\prime}$
by (generate-fresh name, auto simp add: fresh-prod)
ultimately show ?thesis
by (cases rule: transitions.strong-cases $[$ where $x=y$ and $b=y$ and $x a=y$ and $x b=y$ and $x c=y$ and $x d=y$ and $x e=y$
and $x f=y$ and $x g=y$ and $y=z$ and $y a=z$
and $y b=z$ and $y c=x]$ )
(auto simp add: pi.inject residualInject alpha abs-fresh fresh-prod)+ qed
lemma resCasesB[consumes 3, case-names cOpen cRes]:
fixes $x$ :: name
and $P:: p i$
and $a$ :: subject
and $y$ :: name
and $y P^{\prime}:: p i$
and $C$ ::' $a:: f s$-name
assumes Trans: $<\nu y>P \longmapsto a « x » \prec y P^{\prime}$
and xineqy: $x \neq y$
and xineqy: $x \sharp P$
and $\quad r c$ Open: $\bigwedge b P^{\prime} . \llbracket P \longmapsto b[y] \prec P^{\prime} ; b \neq y ; a=$ BoundOutputS $b \rrbracket \Longrightarrow F$
(BoundOutputS b) $\left([(x, y)] \cdot P^{\prime}\right)$
and $\left.\quad r c \operatorname{Res} B: \bigwedge P^{\prime} . \llbracket P \longmapsto a « x\right\rangle \prec P^{\prime} ; y \sharp a \rrbracket \Longrightarrow F a\left(<\nu y>P^{\prime}\right)$
shows $F$ a $y P^{\prime}$
proof -
note assms
moreover from $\left.\langle\langle\nu y\rangle P \longmapsto a « x\rangle \prec y P^{\prime}\right\rangle\langle x \neq y\rangle$ have $y \sharp a$ and $y \sharp y P^{\prime}$
by (force dest: freshBoundDerivative simp add: abs-fresh)+
moreover from 〈< $\left.\langle y\rangle P \longmapsto a « x\rangle \prec y P^{\prime}\right\rangle\langle x \sharp P\rangle$ have $x \sharp a$
by (force dest: freshBoundDerivative simp add: abs-fresh)+
moreover obtain $z:: n a m e$ where $z \neq y$ and $z \neq x$ and $z \sharp P$ and $z \sharp a$ and $z \sharp y P^{\prime}$
by (generate-fresh name, auto simp add: fresh-prod)
moreover obtain $z^{\prime}::$ name where $z^{\prime} \neq y$ and $z^{\prime} \neq x$ and $z^{\prime} \neq z$ and $z^{\prime} \sharp P$ and $z^{\prime} \sharp a$ and $z^{\prime} \sharp y P^{\prime}$
by (generate-fresh name, auto simp add: fresh-prod)
ultimately show ?thesis
by (cases rule: transitions.strong-cases $[$ where $x=z$ and $b=y$ and $x a=z$ and $x b=z$ and $x c=z$ and $x d=z$ and $x e=z$
and $x f=z$ and $x g=x$ and $y=z^{\prime}$ and $y a=z^{\prime}$ and $y b=y$ and $\left.y c=z^{\prime}\right]$ )
(auto simp add: pi.inject residualInject alpha abs-fresh fresh-prod fresh-left calc-atm)+
qed
lemma bangInduct[consumes 1, case-names cPar1B cPar1F cPar2B cPar2F cComm1 cComm2 cClose1 cClose2 cBang]:
fixes $F$ :: ' $a::$ fs-name $\Rightarrow p i \Rightarrow$ residual $\Rightarrow$ bool
and $P:: p i$
and $\quad R s$ :: residual
and $C::$ ' $a:: f s$-name
assumes Trans: ! $P \longmapsto R s$
and $\quad c P a r 1 B: \bigwedge a x P^{\prime} C . \llbracket P \longmapsto a « x » \prec P^{\prime} ; x \sharp P ; x \sharp C \rrbracket \Longrightarrow F C(P \|$ $!P)\left(a « x » \prec P^{\prime} \|!P\right)$
and $\quad c \operatorname{Par} 1 F: \bigwedge \alpha P^{\prime} C . \llbracket P \longmapsto \alpha \prec P \rrbracket \Longrightarrow F C(P \|!P)\left(\alpha \prec P^{\prime} \|!P\right)$
and $\quad c \operatorname{Par2B}: \bigwedge a x P^{\prime} C . \llbracket!P \longmapsto a « x » \prec P^{\prime} ; x \sharp P ; x \sharp C ; \bigwedge C . F C(!P)$ $\left(a « x>\prec P^{\prime}\right) \rrbracket \Longrightarrow$
$\left.F C(P \|!P)(a « x\rangle \prec P \| P^{\prime}\right)$
and cPar2F: $\wedge \alpha P^{\prime} C . \llbracket!P \longmapsto \alpha \prec P^{\prime} ; \wedge C . F C(!P)\left(\alpha \prec P^{\prime}\right) \rrbracket \Longrightarrow F C$ $(P \|!P)\left(\alpha \prec P \| P^{\prime}\right)$
and $\quad c$ Comm1: $\bigwedge a x P^{\prime} b P^{\prime \prime} C . \llbracket P \longmapsto a<x>\prec P^{\prime} ;!P \longmapsto$ (OutputR a b) $\prec P^{\prime \prime} ; x \sharp C$;
^C.FC(!P) $\left(\left(\right.\right.$ OutputR ab) $\left.\prec P^{\prime \prime}\right) \rrbracket \Longrightarrow$
$F C(P \|!P)\left(\tau \prec\left(P^{\prime}[x::=b]\right) \| P^{\prime \prime}\right)$
and $\quad c$ Comm2: $\bigwedge a b P^{\prime} x P^{\prime \prime} C . \llbracket P \longmapsto($ OutputR a $b) \prec P^{\prime} ;!P \longmapsto a<x>$ $\prec P^{\prime \prime} ; x \sharp C$;
$\bigwedge C . F C(!P)\left(a<x>\prec P^{\prime \prime}\right) \rrbracket \Longrightarrow$
$F C(P \|!P)\left(\tau \prec P^{\prime} \|\left(P^{\prime \prime}[x::=b]\right)\right)$
and $\quad c$ Close1: $\bigwedge a x P^{\prime} y P^{\prime \prime} C . \llbracket P \longmapsto a<x>\prec P^{\prime} ;!P \longmapsto a<\nu y>\prec P^{\prime \prime} ; y$ $\sharp P ; x \sharp C ; y \sharp C$;
^C. FC $(!P)\left(a<\nu y>\prec P^{\prime \prime}\right) \rrbracket \Longrightarrow$
$F C(P \|!P)\left(\tau \prec<\nu y>\left(\left(P^{\prime}[x::=y]\right) \| P^{\prime \prime}\right)\right)$
and $\quad c$ Close2: $\bigwedge a y P^{\prime} x P^{\prime \prime} C . \llbracket P \longmapsto a<\nu y>\prec P^{\prime} ;!P \longmapsto a<x>\prec P^{\prime \prime} ; y$ $\sharp P ; x \sharp C ; y \sharp C$;

$$
\bigwedge C . F C(!P)\left(a<x>\prec P^{\prime \prime}\right) \rrbracket \Longrightarrow
$$

```
    FC(P|!P)(\tau\prec<\nuy>(\mp@subsup{P}{}{\prime}|(\mp@subsup{P}{}{\prime\prime}[x::=y]))}
    and cBang: \bigwedgeRs C.\llbracketP|!P\longmapstoRs;\bigwedgeC.FC(P|!P)Rs\rrbracket\LongrightarrowFC(!P)
Rs
    shows F C (!P) Rs
proof -
    have }\bigwedgeXYC.\llbracketX\longmapstoY;bangPred P X\rrbracket\LongrightarrowFCX
    proof -
    fix X Y C
    assume }X\longmapstoY\mathrm{ and bangPred P X
    thus FCXY
    proof(nominal-induct avoiding:C rule: transitions.strong-induct)
            case(Tau Pa)
            thus ?case
                apply -
                by(ind-cases bangPred P (\tau.(Pa)))
    next
            case(Input x a Pa)
            thus ?case
                        apply -
                by(ind-cases bangPred P(a<x>.Pa))
    next
            case(Output a b Pa)
            thus ?case
                        apply -
            by(ind-cases bangPred P (a{b}.Pa))
    next
            case(Match Pa Rs b)
            thus ?case
            apply -
            by(ind-cases bangPred P ([b\frownb]Pa))
    next
            case(Mismatch Pa Rs a b)
            thus ?case
                apply -
            by(ind-cases bangPred P([a\not=b]Pa))
    next
            case(Open Pa a b Pa')
            thus ?case
                    apply -
            by(ind-cases bangPred P(<\nub>Pa))
    next
            case(Sum1 Pa Rs Q)
            thus ?case
                    apply -
                    by(ind-cases bangPred P(Pa\oplusQ))
    next
        case(Sum2 Q Rs Pa)
            thus ?case
```

```
        apply -
    by(ind-cases bangPred P(Pa\oplusQ))
    next
    case(Par1B Pa a x Pa' Q )
    thus ?case
        apply -
        by(ind-cases bangPred P (Pa|Q), auto intro: cPar1B simp add: pi.inject)
        next
    case(Par1F Pa \alpha Pa' Q)
    thus ?case
        apply -
        by(ind-cases bangPred P(Pa|Q), auto intro: cPar1F simp add: pi.inject)
        next
    case(Par2B Qa a x Qa' Pa)
    thus ?case
        apply -
            by(ind-cases bangPred P (Pa| Qa), auto intro: cPar2B aux1 simp add:
pi.inject)
    next
        case(Par2F Qa \alpha Qa' Pa)
        thus ?case
            apply -
            by(ind-cases bangPred P(Pa|Qa), auto intro: cPar2F aux1 simp add:
pi.inject)
    next
        case(Comm1 Pa a x Pa' Q b Q')
        thus ?case
            apply -
            by(ind-cases bangPred P(Pa|Q), auto intro: cComm1 aux1 simp add:
pi.inject)
    next
        case(Comm2 Pa a b Pa' Q x Q')
        thus ?case
            apply -
            by(ind-cases bangPred P(Pa|Q), auto intro: cComm2 aux1 simp add:
pi.inject)
    next
        case(Close1 Pa a x Pa' Q y Q')
        thus ?case
            apply -
            by(ind-cases bangPred P(Pa|Q), auto intro: cClose1 aux1 simp add:
pi.inject)
    next
        case(Close2 Pa a y Pa' Q x Q')
        thus ?case
            apply -
                by(ind-cases bangPred P(Pa|Q), auto intro: cClose2 aux1 simp add:
pi.inject)
    next
```

```
        case(ResB Pa a x P' y)
        thus ?case
            apply -
            by(ind-cases bangPred P (<\nuy>Pa))
        next
            case(ResF Pa \alpha P' y)
            thus ?case
            apply -
            by(ind-cases bangPred P(<\nuy>Pa))
        next
            case(Bang Pa Rs)
            thus ?case
            apply -
            by(ind-cases bangPred P(!Pa), auto intro: cBang aux2 simp add: pi.inject)
        qed
qed
with Trans show ?thesis by(force intro: bangPred.aux1)
qed
end
theory Late-Semantics1
    imports Late-Semantics
begin
free-constructors case-subject for
    InputS
| BoundOutputS
by(auto simp add: subject.inject)
    (metis Rep-subject-inverse subject.constr-rep(1,2) subject-Rep.exhaust)
free-constructors case-freeRes for
    OutputR
| TauR
by(auto simp add: freeRes.inject)
    (metis Abs-freeRes-cases Abs-freeRes-inverse freeRes.constr-rep(1,2) freeRes-Rep.exhaust)
end
theory Rel
    imports Agent
begin
definition eqvt :: (('a::pt-name) > ('a::pt-name)) set }=>\mathrm{ bool
    where eqvt Rel }\equiv(\forallx(\mathrm{ perm::name prm). x }\in\mathrm{ Rel }\longrightarrow\mathrm{ perm }\cdotx\in\operatorname{Rel}
lemma eqvtRelI:
    fixes Rel :: ('a::pt-name > 'a) set
```

```
    and }P\quad::'
    and }Q :: 'a
    and perm :: name prm
    assumes eqvt Rel
    and}\quad(P,Q)\inRe
    shows (perm • P, perm • Q) \in Rel
using assms
by(auto simp add: eqvt-def)
lemma eqvtRelE:
    fixes Rel :: ('a::pt-name > 'a) set
    and }P\mathrm{ :: 'a
    and }Q\quad::'
    and perm :: name prm
    assumes eqvt Rel
    and (perm • P, perm • Q)\inRel
    shows }(P,Q)\in\mathrm{ Rel
proof -
    have rev perm • (perm •P) =P and rev perm • (perm •Q) =Q
    by(simp add: pt-rev-pi[OF pt-name-inst, OF at-name-inst])+
    with assms show ?thesis
    by(force dest: eqvtRelI[of - - rev perm])
qed
lemma eqvtTrans[intro]:
    fixes Rel :: ('a::pt-name > 'a) set
    and Rel':: (' }a\times\mp@subsup{}{}{\prime}a) se
    assumes EqvtRel: eqvt Rel
    and EqvtRel': eqvt Rel'
    shows eqvt (Rel O Rel')
using assms
by(force simp add: eqvt-def)
lemma eqvtUnion[intro]:
    fixes Rel :: ('a::pt-name > 'a) set
    and Rel':: ('}\mp@subsup{}{}{\prime}\times\mp@subsup{}{}{\prime}a) se
    assumes EqvtRel: eqvt Rel
    and EqvtRel': eqvt Rel'
    shows eqvt (Rel \cupRel')
using assms
by(force simp add: eqvt-def)
```

```
definition substClosed :: (pi \times pi) set }=>(pi\timespi) set wher
    substClosed Rel \equiv{(P,Q)|PQ.\forall\sigma.(P[<\sigma>],Q[<\sigma>]) \inRel}
lemma eqvtSubstClosed:
    fixes Rel :: (pi \times pi) set
    assumes eqvtRel: eqvt Rel
    shows eqvt (substClosed Rel)
proof(simp add: eqvt-def substClosed-def, auto)
    fix PQ perm s
    assume }\foralls.(P[<s>],Q[<s>])\in\operatorname{Rel
    hence (P[<(rev (perm::name prm) • s)>],Q[<(rev perm \cdot s)>]) \in Rel by simp
    with eqvtRel have (perm • (P[<(rev perm • s)>]), perm • (Q[< (rev perm • s)>]))
Rel
    by(rule eqvtRelI)
    thus ((perm • P)[<s>], (perm •Q)[<s>]) \in Rel
    by(simp add: name-per-rev)
qed
lemma substClosedSubset:
    fixes Rel :: (pi }\times pi) se
    shows substClosed Rel \subseteqRel
proof(auto simp add: substClosed-def)
    fix P Q
    assume }\foralls.(P[<s>],Q[<s>])\in\operatorname{Rel
    hence (P[<[]>],Q[<[]>])\in Rel by blast
    thus }(P,Q)\in\mathrm{ Rel by simp
qed
lemma partUnfold:
    fixes P :: pi
    and }Q :: p
    and }\sigma::(\mathrm{ name }\times\mathrm{ name) list
    and Rel :: (pi\timespi) set
    assumes }(P,Q)\in\mathrm{ substClosed Rel
    shows (P[<\sigma>],Q[<\sigma>])\in substClosed Rel
using assms
proof(auto simp add: substClosed-def)
    fix }\mp@subsup{\sigma}{}{\prime
    assume }\forall\sigma.(P[<\sigma>],Q[<\sigma>])\inRe
    hence (P[<(\sigma@\mp@subsup{\sigma}{}{\prime})>],Q[<(\sigma@\mp@subsup{\sigma}{}{\prime})>])\inRel by blast
    thus }((P[<\sigma>])[<\mp@subsup{\sigma}{}{\prime}>],(Q[<\sigma>])[<\mp@subsup{\sigma}{}{\prime}>])\in\mathrm{ Rel
    by simp
```

```
qed
inductive-set bangRel :: (pi\timespi) set }=>(pi\timespi) se
for Rel :: (pi < pi) set
where
    BRBang: (P,Q) \in Rel \Longrightarrow(!P,!Q) \in bangRel Rel
| BRPar: (R,T) \inRel \Longrightarrow(P,Q)\in(bangRel Rel) \Longrightarrow(R|P,T|Q)\in(bangRel
Rel)
| BRRes: }(P,Q)\in\mathrm{ bangRel Rel }\Longrightarrow(<\nua>P,<\nua>Q)\in bangRel Re
inductive-cases BRBangCases':(P,!Q)\in bangRel Rel
inductive-cases BRParCases':(P,Q|!Q)\in bangRel Rel
inductive-cases BRResCases':}(P,<\nux>Q)\in\mathrm{ bangRel Rel
lemma eqvtBangRel:
    fixes Rel:: (pi\times pi) set
    assumes eqvtRel: eqvt Rel
    shows eqvt(bangRel Rel)
proof(simp add: eqvt-def, auto)
    fix P Q perm
    assume (P,Q) \in bangRel Rel
    thus ((perm::name prm) • P, perm • Q) \in bangRel Rel
    proof(induct)
        fix PQ
        assume (P,Q) \in Rel
        with eqvtRel have (perm • P, perm • Q) \in Rel
            by(rule eqvtRelI)
        thus (perm •!P, perm •!Q)\in bangRel Rel
            by(force intro: BRBang)
    next
        fix PQ RT
        assume R: (R,T)\inRel
        assume BR:(perm • P, perm •Q)\in bangRel Rel
        from eqvtRel R have (perm • R, perm • T) \in Rel
            by(rule eqvtRelI)
        with BR show (perm • (R|P), perm • (T|Q)) \in bangRel Rel
            by(force intro: BRPar)
    next
        fix PQa
        assume (perm • P, perm •Q) \in bangRel Rel
        thus (perm \cdot <\nua>P, perm \cdot <\nua>Q) \in bangRel Rel
            by(force intro: BRRes)
    qed
qed
```

```
lemma BRBangCases[consumes 1, case-names BRBang]:
    fixes P :: pi
    and }Q :: p
    and Rel :: (pi\timespi) set
    and F :: pi=>bool
    assumes }(P,!Q)\in\mathrm{ bangRel Rel
    and}\quad\bigwedgeP.(P,Q)\in\operatorname{Rel}\LongrightarrowF(!P
    shows F P
using assms
by(induct rule: BRBangCases', auto simp add: pi.inject)
lemma BRParCases[consumes 1, case-names BRPar]:
    fixes P :: pi
    and }Q ::p
    and Rel :: (pi\timespi) set
    and F :: pi=>bool
    assumes }(P,Q|!Q)\in\mathrm{ bangRel Rel
    and}\quad\PR.\llbracket(P,Q)\in\operatorname{Rel};(R,!Q)\in\mathrm{ bangRel Rel】 }\LongrightarrowF(P|R
    shows F P
using assms
by(induct rule: BRParCases', auto simp add: pi.inject)
lemma bangRelSubset:
    fixes Rel :: (pi\timespi) set
    and Rel':: (pi }\times pi) se
    assumes }(P,Q)\in\mathrm{ bangRel Rel
    and}\quad\PQ.(P,Q)\in\operatorname{Rel}\Longrightarrow(P,Q)\inRel'
    shows }(P,Q)\in\mathrm{ bangRel Rel'
using assms
by(induct rule: bangRel.induct) (auto intro: BRBang BRPar BRRes)
lemma bangRelSymetric:
    fixes P :: pi
    and }Q :: p
    and Rel :: (pi\timespi) set
    assumes A: (P,Q)\in bangRel Rel
    and Sym: \PQ.(P,Q)\inRel \Longrightarrow(Q,P)\in\operatorname{Rel}
    shows}(Q,P)\in\mathrm{ bangRel Rel
proof -
    from A show ?thesis
    proof(induct)
```

```
    fix PQ
    assume (P,Q)\inRel
    hence }(Q,P)\in\mathrm{ Rel by(rule Sym)
    thus (!Q,!P) \in bangRel Rel by(rule BRBang)
    next
    fix PQRT
    assume RRelT:}(R,T)\in\mathrm{ Rel
    assume IH: (Q,P)\in bangRel Rel
    from RRelT have (T,R)\inRel by(rule Sym)
    thus (T|Q,R|P)\in bangRel Rel using IH by(rule BRPar)
    next
        fix P Qa
        assume (Q,P)\in bangRel Rel
        thus (<\nua>Q,<\nua>P)\in bangRel Rel by(rule BRRes)
        qed
qed
primrec resChain :: name list }=>\mathrm{ pi # pi where
    base: resChain [] P=P
    | step: resChain (x#xs) P = <\nux>(resChain xs P)
lemma resChainPerm[simp]:
    fixes perm :: name prm
    and lst :: name list
    and }P\mathrm{ :: pi
    shows perm • (resChain lst P) = resChain (perm •lst) (perm • P)
by(induct-tac lst, auto)
lemma resChainFresh:
    fixes a :: name
    and lst :: name list
    and }P\mathrm{ :: pi
    assumes }a\sharp(lst,P
    shows }a\sharp\mathrm{ (resChain lst P)
using assms apply(induct-tac lst)
apply(simp add: fresh-prod)
by(simp add: fresh-prod name-fresh-abs)
end
theory Strong-Late-Sim
    imports Late-Semantics1 Rel
begin
definition derivative :: pi => pi=> subject }=>\mathrm{ name }=>(pi\timespi) set => bool where
    derivative P Q a x Rel \equivcase a of InputS b }=>(\forallu.(P[x::=u],Q[x::=u])\inRel
```

$$
\mid \text { BoundOutputS } b \Rightarrow(P, Q) \in \operatorname{Rel}
$$

definition simulation $:: p i \Rightarrow(p i \times p i)$ set $\Rightarrow$ pi $\Rightarrow$ bool $(-\rightsquigarrow[-]-[80,80,80]$ 80) where

$$
P \rightsquigarrow[\text { Rel }] Q \equiv\left(\forall a x Q^{\prime} . Q \longmapsto a « x » \prec Q^{\prime} \wedge x \sharp P \longrightarrow\left(\exists P^{\prime} . P \longmapsto a « x » \prec P^{\prime}\right.\right.
$$

$\wedge$ derivative $P^{\prime} Q^{\prime}$ a $x$ Rel $\left.)\right) \wedge$

$$
\left(\forall \alpha Q^{\prime} . Q \longmapsto \alpha \prec Q^{\prime} \longrightarrow\left(\exists P^{\prime} . P \longmapsto \alpha \prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in \text { Rel }\right)\right)
$$

lemma monotonic:
fixes $A::(p i \times p i)$ set
and $B::(p i \times p i)$ set
and $P:: p i$
and $P^{\prime}:: p i$
assumes $P \rightsquigarrow[A] P^{\prime}$
and $\quad A \subseteq B$

$$
\begin{aligned}
& \text { shows } P \rightsquigarrow[B] P^{\prime} \\
& \text { using assms } \\
& \text { apply(auto simp add: simulat } \\
& \text { by(case-tac a) fastforce }+ \\
& \text { lemma derivativeMonotonic: } \\
& \text { fixes } A::(p i \times p i) \text { set } \\
& \text { and } B::(p i \times p i) \text { set } \\
& \text { and } P:: p i \\
& \text { and } Q:: p i \\
& \text { and } a:: \text { subject } \\
& \text { and } x:: \text { name }
\end{aligned}
$$

apply (auto simp add: simulation-def derivative-def)
assumes derivative $P Q$ ax $A$
and $\quad A \subseteq B$
shows derivative $P Q$ a $x B$
using assms
by (case-tac a, auto simp add: derivative-def)
lemma derivativeEqvtI:
fixes $P \quad:: p i$
and $Q \quad:: p i$
and $a$ :: subject
and $x$ :: name
and Rel $::(p i \times p i)$ set
and perm :: name prm
assumes Der: derivative $P Q$ a $x$ Rel
and Eqvt: equt Rel
shows derivative $($ perm $\cdot P)($ perm $\cdot Q)($ perm $\cdot a)($ perm $\cdot x)$ Rel

```
using assms
apply(case-tac a, auto simp add: derivative-def)
apply(erule-tac x=rev perm • u in allE)
apply(drule-tac perm=perm in eqvtRelI)
apply(blast)
apply(force simp add: eqvt-subs name-per-rev)
by(force simp add: eqvt-def)
lemma derivativeEqvtI2:
    fixes }P\mathrm{ :: pi
    and }Q :: p
    and a :: subject
    and x :: name
    and Rel :: (pi\times pi) set
    and perm :: name prm
    assumes Der:derivative P Q a x Rel
    and Eqvt: eqvt Rel
    shows derivative (perm • P) (perm •Q) a (perm • x) Rel
using assms
apply(case-tac a, auto simp add: derivative-def)
apply(erule-tac x=rev perm • u in allE)
apply(drule-tac perm=perm in eqvtRelI)
apply(blast)
apply(force simp add: eqvt-subs name-per-rev)
by(force simp add: eqvt-def)
lemma freshUnit[simp]:
    fixes y :: name
    shows }y\sharp(
by(auto simp add: fresh-def supp-unit)
lemma simCasesCont[consumes 1, case-names Bound Free]:
    fixes P :: pi
    and }Q ::p
    and Rel :: (pi\timespi) set
    and C ::'a::fs-name
    assumes Eqvt: eqvt Rel
    and Bound: \a x Q'.\llbracketQ\longmapstoa«x»\prec < '';x\sharpP;x\sharpQ;x\sharpa;x\sharpC\rrbracket\Longrightarrow
\exists}\mp@subsup{P}{}{\prime}.P\longmapstoa«x>>\prec\mp@subsup{P}{}{\prime}\wedge\mathrm{ derivative }\mp@subsup{P}{}{\prime}\mp@subsup{Q}{}{\prime}\mathrm{ a x Rel
    and Free: }\\alpha \mp@subsup{Q}{}{\prime}.Q\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime}\Longrightarrow\exists\mp@subsup{P}{}{\prime}.P\longmapsto\alpha\prec\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mathrm{ Rel
    shows P\rightsquigarrow[Rel] Q
using Free
proof(auto simp add: simulation-def)
    fix ax Q'
```

assume ( $x$ ::name) $\sharp P$
assume Trans: $Q \longmapsto a « x$ » $\prec Q^{\prime}$
obtain $y::$ name where $y \sharp P$ and $y \sharp Q$ and $y \sharp a$ and $y \sharp C$ and $y \sharp Q^{\prime}$ and $y \neq x$
by (generate-fresh name) auto
from Trans $\left\langle y \sharp Q^{\prime}\right\rangle$ have $Q \longmapsto a « y » \prec[(x, y)] \cdot Q^{\prime} \mathbf{b y}($ simp add: alphaBoundResidual)
hence $\exists P^{\prime} . P \longmapsto a « y » \prec P^{\prime} \wedge$ derivative $P^{\prime}\left([(x, y)] \cdot Q^{\prime}\right)$ a y Rel
using $\langle y \sharp P\rangle\langle y \sharp Q\rangle\langle y \sharp a\rangle\langle y \sharp C\rangle$
by(rule Bound)
then obtain $P^{\prime}$ where PTrans: $P \longmapsto a « y » \prec P^{\prime}$ and PDer: derivative $P^{\prime}([(x$, $y)] \cdot Q^{\prime}$ ) a y Rel
by blast
from PTrans $\langle x \sharp P\rangle\langle y \neq x\rangle$ have $x \sharp P^{\prime}$ by(force intro: freshBoundDerivative) with PTrans have $P \longmapsto a « x \gg[(x, y)] \cdot P^{\prime} \mathbf{b y}($ simp add: alphaBoundResidual name-swap)
moreover have derivative $\left([(x, y)] \cdot P^{\prime}\right) Q^{\prime}$ a x Rel
proof -
from PDer Eqvt have derivative $\left([(x, y)] \cdot P^{\prime}\right)\left([(x, y)] \cdot[(x, y)] \cdot Q^{\prime}\right) a([(x$, y)] - y) Rel
by(rule derivativeEqvtI2)
with $\langle y \neq x\rangle$ show ?thesis by (simp add: name-calc)
qed
ultimately show $\exists P^{\prime} . P \longmapsto a « x » \prec P^{\prime} \wedge$ derivative $P^{\prime} Q^{\prime}$ a x Rel by blast qed
lemma simCases[case-names Bound Free]:
fixes $P$ :: $p i$
and $\quad Q \quad:: p i$
and Rel $::(p i \times p i)$ set
assumes Bound: $\bigwedge a y Q^{\prime} . \llbracket Q \longmapsto a « y » \prec Q^{\prime} ; y \sharp P \rrbracket \Longrightarrow \exists P^{\prime} . P \longmapsto a « y » \prec$ $P^{\prime} \wedge$ derivative $P^{\prime} Q^{\prime}$ a y Rel
and Free: $\wedge \alpha Q^{\prime} . Q \longmapsto \alpha \prec Q^{\prime} \Longrightarrow \exists P^{\prime} . P \longmapsto \alpha \prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in$ Rel
shows $P \rightsquigarrow[$ Rel $] Q$
using assms
by (auto simp add: simulation-def)
lemma resSimCases[consumes 1, case-names BoundOutput BoundR FreeR]:
fixes $x$ :: name
and $P$ :: pi
and Rel $::(p i \times p i)$ set
and $Q \quad:: p i$
and $C$ ::' $a:: f s$-name
assumes Eqvt: eqvt Rel
and Bound $O: \bigwedge Q^{\prime} a . \llbracket Q \longmapsto a[x] \prec Q^{\prime} ; a \neq x \rrbracket \Longrightarrow \exists P^{\prime} . P \longmapsto a<\nu x>\prec$ $P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}$
and $B R: \quad \bigwedge Q^{\prime}$ a $y . \llbracket Q \longmapsto a<y>\prec Q^{\prime} ; x \sharp a ; x \neq y ; y \sharp C \rrbracket \Longrightarrow \exists P^{\prime} . P$ $\longmapsto a « y » \prec P^{\prime} \wedge$ derivative $P^{\prime}\left(<\nu x>Q^{\prime}\right)$ a y Rel
and $\quad B F: \quad \bigwedge Q^{\prime} \alpha . \llbracket Q \longmapsto \alpha \prec Q^{\prime} ; x \sharp \alpha \rrbracket \Longrightarrow \exists P^{\prime} . P \longmapsto \alpha \prec P^{\prime} \wedge\left(P^{\prime}\right.$, $\left.<\nu x>Q^{\prime}\right) \in$ Rel
shows $P \rightsquigarrow[$ Rel $]<\nu x>Q$
using Eqvt
$\operatorname{proof}($ induct rule: simCasesCont $[$ where $C=(C, x, Q)])$
case(Bound a y $Q^{\prime}$ )
have $y \sharp(C, x, Q)$ by fact
hence $y$ Fresh $C: y \sharp C$ and yineqx: $y \neq x$ and $y \sharp Q$
by (simp add: fresh-prod) +
have $<\nu x>Q \longmapsto a<y>\prec Q^{\prime}$ by fact
thus ? case using yineqx 〈 $y \sharp Q$ 〉
proof (induct rule: resCasesB)
case (cOpen $a^{\prime} Q^{\prime}$ )
have $Q \longmapsto a a^{\prime}[x] \prec Q^{\prime}$ and $a^{\prime} \neq x$ by fact +
then obtain $P^{\prime}$ where PTrans: $P \longmapsto a^{\prime}<\nu x>\prec P^{\prime}$ and $P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in$ Rel by (force dest: BoundO)
from PTrans $\left\langle y \sharp P>\right.$ yineqx have $y \sharp P^{\prime}$ by (force dest: freshBoundDerivative)
with PTrans have $P \longmapsto a^{\prime}<\nu y>\prec\left([(x, y)] \cdot P^{\prime}\right)$ by $($ simp add: alphaBoundResidual)
moreover from $P^{\prime}$ RelQ $Q^{\prime}$ Eqvt have $\left([(x, y)] \cdot P^{\prime},[(x, y)] \cdot Q^{\prime}\right) \in \operatorname{Rel}$ by (auto simp add: eqvt-def)
ultimately show ?case by (force simp add: derivative-def name-swap)
next
case (cRes $Q^{\prime}$ )
have $Q \longmapsto a « y » \prec Q^{\prime}$ and $x \sharp a$ by fact +
with yineqx yFresh $C$ show ?case by (force dest: BR)
qed
next
case (Free $\alpha Q^{\prime}$ )
have $<\nu x>Q \longmapsto \alpha \prec Q^{\prime}$ by fact
thus ?case
proof (induct rule: resCasesF)
case (cRes $Q^{\prime}$ )
have $Q \longmapsto \alpha \prec Q^{\prime}$ and $x \sharp \alpha$ by fact +
thus ?case by (rule BF)
qed
qed
lemma simE:
fixes $P$ :: $p i$
and Rel $::(p i \times p i)$ set
and $\quad Q \quad:: p i$
and $a$ :: subject
and $x$ :: name
and $Q^{\prime}:: p i$
assumes $P \rightsquigarrow[R e l] Q$
shows $Q \longmapsto a « x » \prec Q^{\prime} \Longrightarrow x \sharp P \Longrightarrow \exists P^{\prime} . P \longmapsto a « x » \prec P^{\prime} \wedge\left(\right.$ derivative $P^{\prime}$ $Q^{\prime}$ a $x$ Rel)
and $\quad Q \longmapsto \alpha \prec Q^{\prime} \Longrightarrow \exists P^{\prime} . P \longmapsto \alpha \prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}$
using assms by (simp add: simulation-def)+
lemma eqvtI:
fixes $P \quad:: p i$
and $Q \quad:: p i$
and Rel :: $(p i \times p i)$ set
and perm :: name prm
assumes Sim: $P \rightsquigarrow[$ Rel $] Q$
and $\quad \operatorname{RelRel}^{\prime}:$ Rel $\subseteq$ Rel $^{\prime}$
and EqvtRel': eqvt Rel'
shows $($ perm $\cdot P) \rightsquigarrow\left[\right.$ Rel $\left.^{\prime}\right]($ perm $\cdot Q)$
proof (induct rule: simCases)
case(Bound a y $Q^{\prime}$ )
have QTrans: $($ perm $\cdot Q) \longmapsto a « y » \prec Q^{\prime}$ and $y$ FreshP: $y \sharp$ perm $\cdot P$ by fact +
from $Q$ Trans have $($ rev perm $\cdot($ perm $\cdot Q)) \longmapsto$ rev perm $\cdot\left(a « y » \prec Q^{\prime}\right)$
by (rule transitions.eqvt)
hence $Q \longmapsto($ rev perm $\cdot a) «($ rev perm $\cdot y)$ » $\prec\left(\right.$ rev perm $\left.\cdot Q^{\prime}\right)$
by (simp add: name-rev-per)
moreover from $y$ Fresh $P$ have (rev perm $\cdot y) \sharp P$ by (simp add: name-fresh-left)
ultimately have $\exists P^{\prime} . P \longmapsto($ rev perm $\cdot a) «$ rev perm $\cdot y » \prec P^{\prime} \wedge$ derivative $P^{\prime}\left(\right.$ rev perm $\left.\cdot Q^{\prime}\right)($ rev perm $\cdot a)($ rev perm $\cdot y)$ Rel using Sim
by (force intro: simE)
then obtain $P^{\prime}$ where PTrans: $P \longmapsto($ rev perm $\cdot a) «$ rev perm $\cdot y » \prec P^{\prime}$ and Pderivative: derivative $P^{\prime}\left(\right.$ rev perm $\left.\cdot Q^{\prime}\right)($ rev perm $\cdot a)($ rev perm $\cdot$ y) Rel by blast
from PTrans have $($ perm $\cdot P) \longmapsto$ perm $\bullet\left((\right.$ rev perm $\cdot a) «$ rev perm $\left.\cdot y » \prec P^{\prime}\right)$ by (rule transitions.eqvt)
hence L1: $($ perm $\cdot P) \longmapsto a « y » \prec\left(\right.$ perm $\left.\cdot P^{\prime}\right) \mathbf{b y}($ simp add: name-per-rev $)$
from Pderivative RelRel' have derivative $P^{\prime}\left(\right.$ rev perm $\left.\cdot Q^{\prime}\right)($ rev perm $\cdot a)$ (rev perm • y) Rel ${ }^{\prime}$
by(rule derivativeMonotonic)
hence derivative $\left(\right.$ perm $\left.\cdot P^{\prime}\right)\left(\right.$ perm $\cdot\left(\right.$ rev perm $\left.\left.\cdot Q^{\prime}\right)\right)($ perm $\cdot($ rev perm $\cdot a))$ (perm • (rev perm • y)) Rel' using EqutRel ${ }^{\prime}$
by (rule derivativeEqvtI)
hence derivative $\left(\right.$ perm $\left.\cdot P^{\prime}\right) Q^{\prime}$ a y Rel ${ }^{\prime}$ by (simp add: name-per-rev)
with $L 1$ show ? case by blast

```
next
    case(Free \alpha Q ')
    have (perm •Q)\longmapsto\alpha\prec Q' by fact
    hence (rev perm • (perm •Q))\longmapsto rev perm • (\alpha\prec < Q)
        by(rule transitions.eqvt)
    hence }Q\longmapsto(\mathrm{ rev perm • 人)}\prec(\mathrm{ rev perm • Q')
    by(simp add: name-rev-per)
    with Sim have }\exists\mp@subsup{P}{}{\prime}.P\longmapsto(\mathrm{ rev perm • 人) ఒ P'^ ( }\mp@subsup{P}{}{\prime},(\mathrm{ rev perm • Q}\mp@subsup{Q}{}{\prime}))\in\mathrm{ Rel
    by(force intro: simE)
    then obtain P' where PTrans: P\longmapsto (rev perm • \alpha)\prec < P' and PRel: ( }\mp@subsup{P}{}{\prime},(\mathrm{ rev
perm}\cdot\mp@subsup{Q}{}{\prime}))\inRe
    by blast
    from PTrans have (perm • P)\longmapsto perm • ((rev perm • \alpha)\prec P') by(rule transi-
tions.eqvt)
    hence L1: (perm • P)\longmapsto\alpha\prec (perm • P') by (simp add: name-per-rev)
    from PRel EqvtRel' RelRel'}\mp@subsup{}{}{\prime}\mathrm{ have ((perm • P'),(perm • (rev perm • Q })))\in\mp@subsup{\mathrm{ Rel'}}{}{\prime
    by(force intro: eqvtRelI)
    hence ((perm • P'), Q') \in Rel' by(simp add: name-per-rev)
```



```
qed
lemma derivativeReflexive:
    fixes P :: pi
    and a :: subject
    and }x\mathrm{ :: name
    and Rel :: (pi\timespi) set
    assumes Id \subseteqRel
    shows derivative P P a x Rel
using assms
apply(cases a)
by(auto simp add: derivative-def)
lemma reflexive:
    fixes P :: pi
    and Rel :: (pi\timespi) set
    assumes Id\subseteqRel
    shows P\rightsquigarrow[Rel] P
using assms
by(auto simp add: simulation-def derivativeReflexive)
lemma transitive:
```

```
fixes P :: pi
and }Q\quad::p
and R :: pi
and Rel :: (pi\timespi) set
and Rel' :: (pi\timespi) set
```



```
assumes PSimQ: P}\rightsquigarrow[Rel] Q
and }\quadQSimR:Q\rightsquigarrow[Rel'] 
and Eqvt': eqvt Rel"
and Trans:Rel O Rel'}\subseteq\mp@subsup{R}{}{\prime
shows P\rightsquigarrow[Rel'}]
using Eqvt'
proof(induct rule: simCasesCont [where C=Q])
case(Bound a x R')
have RTrans: R\longmapstoa«x» \prec R' by fact
from \langlex\sharp Q> QSimR RTrans obtain }\mp@subsup{Q}{}{\prime}\mathrm{ where QTrans: }Q\longmapstoa«x»\prec\mp@subsup{Q}{}{\prime
                                    and QDer:derivative Q' R' a x Rel'
    by(blast dest: simE)
with QTrans «x\sharpP>PSimQ obtain P' where PTrans: P\longmapstoa«x» \prec ' '
                                    and PDer: derivative }\mp@subsup{P}{}{\prime}\mp@subsup{Q}{}{\prime}\mathrm{ a x Rel
    by(blast dest: simE)
    moreover from PDer QDer Trans have derivative P' R' a x Rel'"
    by(cases a) (auto simp add: derivative-def)
    ultimately show ?case by blast
next
    case(Free \alpha R')
    have RTrans: }R\longmapsto\alpha\prec\mp@subsup{R}{}{\prime}\mathrm{ by fact
    with QSimR obtain }\mp@subsup{Q}{}{\prime}\mathrm{ where QTrans: Q}\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime
                            and \mp@subsup{Q}{}{\prime}RelR':}(\mp@subsup{Q}{}{\prime},\mp@subsup{R}{}{\prime})\in\mp@subsup{R}{el}{
    by(blast dest: simE)
    from QTrans PSimQ obtain P' where PTrans: P\longmapsto\alpha\longmapsto 
                            and P'RelQ':}(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\operatorname{Rel
    by(blast dest: simE)
    from P'RelQ' Q'RelR' Trans have ( }\mp@subsup{P}{}{\prime},\mp@subsup{R}{}{\prime})\inRel'/ by blas
    with PTrans show ?case by blast
qed
end
theory Strong-Late-Bisim
    imports Strong-Late-Sim
begin
lemma monoAux: A\subseteqB\LongrightarrowP\rightsquigarrow[A]Q\longrightarrowP\rightsquigarrow[B]Q
by(auto intro:Strong-Late-Sim.monotonic)
```

coinductive-set bisim $::(p i \times p i)$ set where

$$
\text { step }: \llbracket P \rightsquigarrow[\text { bisim }] Q ;(Q, P) \in \operatorname{bisim} \rrbracket \Longrightarrow(P, Q) \in \operatorname{bisim}
$$

monos monoAux

## abbreviation

$$
\text { strongBisimJudge (infixr } \sim 65) \text { where } P \sim Q \equiv(P, Q) \in \text { bisim }
$$

lemma monotonic': mono $(\lambda S .\{(P, Q) \mid P Q . P \rightsquigarrow[S] Q \wedge Q \rightsquigarrow[S] P\})$
apply(rule monoI)
by (auto dest: monoAux)
lemma monotonic: mono $(\lambda p x 1 x 2$.

$$
\exists P Q \cdot x 1=P \wedge
$$

$x 2=Q \wedge P \rightsquigarrow[\{(x a, x) . p x a x\}] Q \wedge Q \rightsquigarrow[\{(x a, x) . p x a x\}] P)$
apply(rule monoI)
by(auto intro: Strong-Late-Sim.monotonic)
lemma bisimCoinduct[case-names cSim cSym, consumes 1]:
assumes $p:(P, Q) \in X$
and $\quad r$ Sim: $\bigwedge R S .(R, S) \in X \Longrightarrow R \rightsquigarrow[(X \cup$ bisim $)] S$
and $\quad r S y m: \wedge R S .(R, S) \in X \Longrightarrow(S, R) \in X$
shows $P \sim Q$
proof -
have aux: $X \cup$ bisim $=\{(P, Q) .(P, Q) \in X \vee P \sim Q\}$ by blast
from $p$ show ?thesis
apply (coinduct, auto)
apply (fastforce dest: rSim simp add: aux)
by(fastforce dest: rSym)
qed
lemma bisime:
fixes $P$ :: $p i$
and $\quad Q:: p i$
assumes $P \sim Q$
shows $P \rightsquigarrow[b i s i m] Q$
using assms
by(auto intro: bisim.cases)
lemma bisimi:
fixes $P:: p i$
and $\quad Q:: p i$
assumes $P \rightsquigarrow[$ bisim $] Q$

```
    and
        Q~P
    shows P~Q
using assms
by(rule bisim.intros)
definition old-bisim :: (pi }\timespi) set => bool where
    old-bisim Rel }\equiv\forall(P,Q)\inRel. P\rightsquigarrow[Rel] Q\wedge(Q,P)\inRe
lemma oldBisimBisimEq:
    shows (\bigcup{Rel.(old-bisim Rel)}) = bisim (is ?LHS = ?RHS)
proof
    show ?LHS \subseteq?RHS
    proof auto
        fix P Q Rel
        assume (P,Q)\inRel and old-bisim Rel
        thus P~Q
        proof(coinduct rule: bisimCoinduct)
            case(cSim P Q)
            with <old-bisim Rel` show ?case
                by(fastforce simp add: old-bisim-def intro: Strong-Late-Sim.monotonic)
        next
            case(cSym P Q)
            with <old-bisim Rel` show ?case
                    by(auto simp add: old-bisim-def)
        qed
    qed
next
    show ?RHS \subseteq?LHS
    proof auto
        fix PQ
        assume P~Q
        moreover hence old-bisim bisim
            by(auto simp add: old-bisim-def dest: bisim.cases)
        ultimately show }\exists\mathrm{ Rel. old-bisim Rel }\wedge(P,Q)\in\mathrm{ Rel
            by blast
    qed
qed
lemma reflexive:
    fixes P :: pi
    shows P~P
proof -
    have }(P,P)\inId by sim
    thus ?thesis
        by(coinduct rule: bisimCoinduct, auto intro: Strong-Late-Sim.reflexive)
qed
```

```
lemma symmetric:
    fixes P :: pi
    and }Q::p
    assumes P~Q
    shows Q~P
using assms
by(auto dest: bisim.cases)
lemma bisimClosed:
    fixes }P\mathrm{ :: pi
    and }Q::p
    and p :: name prm
    assumes P~Q
    shows (p\cdotP)~(p\cdotQ)
proof -
    let ? }X={(p\cdotP,p\cdotQ)|PQ(p::name prm). P~Q
    from }\langleP~Q\rangle\mathrm{ have ( }p\cdotP,p\cdotQ)\in?X\mathrm{ by blast
    thus ?thesis
    proof(coinduct rule: bisimCoinduct)
    case(cSim pP pQ)
    from }\langle(pP,pQ)\in?X> obtain PQ p where P~Q and pP=(p::name prm) 
- P and pQ = p.Q
            by auto
    from }\langleP~Q\rangle\mathrm{ have Pœ[bisim] Q by(rule bisimE)
    moreover have bisim}\subseteq?
    proof
            fix }
            assume x b bisim
            moreover have x=(([]::name prm) • x) by auto
            ultimately show }x\in\mathrm{ ?X
                    apply(case-tac x)
                    by(clarify, simp only: eqvts) metis
    qed
    moreover have eqvt ?X
    proof (auto simp add: eqvt-def)
            fix PQ
            fix perm1::name prm
            fix perm2::name prm
            assume P ~ Q
            moreover have perm1 - perm2 . P = (perm1 @ perm2) . P by(simp add:
pt2[OF pt-name-inst])
            moreover have perm1 • perm2 - Q = (perm1 @ perm2) - Q by(simp add:
pt2[OF pt-name-inst])
```

ultimately show $\exists P^{\prime} Q^{\prime} .(\exists$ (perm::name prm $)$. perm1 $\cdot$ perm2 $\cdot P=$ perm $\cdot P^{\prime} \wedge$
perm1 $\left.\cdot \operatorname{perm2} \cdot Q=\operatorname{perm} \cdot Q^{\prime}\right) \wedge P^{\prime} \sim Q^{\prime}$

```
            by blast
        qed
        ultimately have \((p \cdot P) \rightsquigarrow[? X](p \cdot Q)\)
            by (rule Strong-Late-Sim.eqvtI)
    with \(\langle p P=p \cdot P\rangle\langle p Q=p \cdot Q\rangle\) show ?case
        by (force intro: Strong-Late-Sim.monotonic)
    next
        case \((\) cSym \(P Q)\)
        thus ?case by (auto intro: symmetric)
    qed
qed
lemma bisimEqvt[simp]:
    shows eqvt bisim
by(auto simp add: eqvt-def bisimClosed)
lemma transitive:
    fixes \(P\) :: \(p i\)
    and \(\quad Q:: p i\)
    and \(\quad R:: p i\)
    assumes \(P \sim Q\)
    and \(\quad Q \sim R\)
    shows \(P \sim R\)
proof -
    let ? \(X=\) bisim \(O\) bisim
    from assms have \((P, R) \in ? X\) by blast
    thus ?thesis
    proof (coinduct rule: bisimCoinduct)
        case \((c \operatorname{Sim} P R)\)
        thus? case
        by (fastforce intro: Strong-Late-Sim.transitive dest: bisimE simp add: equtTrans)
    next
        case \((c \operatorname{Sym} P R)\)
        thus ?case
            by (auto dest: symmetric)
    qed
qed
```

lemma bisimTransitiveCoinduct[case-names cSim cSym, case-conclusion bisim step, consumes 2]:
assumes $(P, Q) \in X$
and eqvt $X$
and $r \operatorname{Sim}: \wedge R S .(R, S) \in X \Longrightarrow R \rightsquigarrow[(\operatorname{bisim} O(X \cup$ bisim $) O$ bisim $)] S$
and $r$ Sym: $\bigwedge R S .(R, S) \in X \Longrightarrow(S, R) \in \operatorname{bisim} O(X \cup$ bisim $) O$ bisim

```
    shows \(P \sim Q\)
proof -
    let ? \(X=\operatorname{bisim} O(X \cup\) bisim \() O\) bisim
    from \(\langle(P, Q) \in X\rangle\) have \((P, Q) \in\) ? \(X\) by (auto intro: reflexive)
    thus ?thesis
    proof (coinduct rule: bisimCoinduct)
        case \((c \operatorname{Sim} P Q)\)
        \{
        fix \(P P^{\prime} Q^{\prime} Q\)
        assume \(P \sim P^{\prime}\) and \(\left(P^{\prime}, Q^{\prime}\right) \in X \cup\) bisim and \(Q^{\prime} \sim Q\)
        have \(P \rightsquigarrow[(? X \cup\) bisim \()] Q\)
        proof \(\left(\right.\) cases \(\left.\left(P^{\prime}, Q^{\prime}\right) \in X\right)\)
            case True
            from \(\left\langle P \sim P^{\prime}\right\rangle\) have \(P \rightsquigarrow[\) bisim \(] P^{\prime}\) by (rule bisimE)
            moreover from \(\left\langle\left(P^{\prime}, Q^{\prime}\right) \in X\right\rangle\) have \(P^{\prime} \rightsquigarrow[(? X)] Q^{\prime}\) by (rule rSim)
            moreover from <eqvt \(X\) 〉 bisimEqvt have eqvt(? \(X \cup\) bisim) by blast
            moreover have bisim \(O ? X \subseteq\) ? \(X \cup\) bisim by (auto dest: transitive)
            ultimately have \(P \rightsquigarrow[(? X \cup\) bisim \()] Q^{\prime} \mathbf{b y}\) (rule Strong-Late-Sim.transitive)
            moreover from \(\left\langle Q^{\prime} \sim Q\right\rangle\) have \(Q^{\prime} \rightsquigarrow[\) bisim \(] Q\) by (rule bisimE)
            moreover note <eqvt \((? X \cup\) bisim) 〉
            moreover have (? \(X \cup\) bisim) \(O\) bisim \(\subseteq\) ? \(X \cup\) bisim
                    by auto (blast dest: transitive)+
            ultimately show ?thesis by(rule Strong-Late-Sim.transitive)
        next
            case False
            from \(\left\langle\left(P^{\prime}, Q^{\prime}\right) \notin X\right\rangle\left\langle\left(P^{\prime}, Q^{\prime}\right) \in X \cup\right.\) bisim \(\rangle\) have \(P^{\prime} \sim Q^{\prime}\) by simp
            with \(\left\langle P \sim P^{\prime}\right\rangle\left\langle Q^{\prime} \sim Q\right\rangle\) have \(P \sim Q\) by (blast dest: transitive)
            hence \(P \rightsquigarrow[\) bisim \(] Q\) by(rule bisimE)
            moreover have bisim \(\subseteq\) ? \(X \cup\) bisim by auto
            ultimately show ?thesis by(rule Strong-Late-Sim.monotonic)
            qed
        \}
        with \(\langle(P, Q) \in\) ? \(X\rangle\) show ? case by auto
        case \((c S y m P Q)\)
        \{
            fix \(P P^{\prime} Q^{\prime} Q\)
            assume \(P \sim P^{\prime}\) and \(\left(P^{\prime}, Q^{\prime}\right) \in X \cup\) bisim and \(Q^{\prime} \sim Q\)
            have \((Q, P) \in \operatorname{bisim} O(X \cup\) bisim \() O\) bisim
            proof \(\left(\right.\) cases \(\left.\left(P^{\prime}, Q^{\prime}\right) \in X\right)\)
                case True
                from \(\left\langle\left(P^{\prime}, Q^{\prime}\right) \in X\right\rangle\) have \(\left(Q^{\prime}, P^{\prime}\right) \in ? X\) by (rule rSym)
            then obtain \(Q^{\prime \prime} P^{\prime \prime}\) where \(Q^{\prime} \sim Q^{\prime \prime}\) and \(\left(Q^{\prime \prime}, P^{\prime \prime}\right) \in X \cup\) bisim and \(P^{\prime \prime}\)
\(\sim P^{\prime}\)
                by auto
            from \(\left\langle Q^{\prime} \sim Q\right\rangle\left\langle Q^{\prime} \sim Q^{\prime \prime}\right\rangle\) have \(Q \sim Q^{\prime \prime}\) by (metis transitive symmetric)
                moreover from \(\left\langle P \sim P^{\prime}\right\rangle\left\langle P^{\prime \prime} \sim P^{\prime}\right\rangle\) have \(P^{\prime \prime} \sim P\) by (metis transitive
```

```
symmetric)
            ultimately show ?thesis using }\langle(\mp@subsup{Q}{}{\prime\prime},\mp@subsup{P}{}{\prime\prime})\inX\cup\mathrm{ bisim> by blast
        next
            case False
            from}\langle(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\not\inX\rangle\langle(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\inX\cup\mathrm{ bisim> have }\mp@subsup{P}{}{\prime}~\mp@subsup{Q}{}{\prime}\mathrm{ by simp
            with}\langleP~\mp@subsup{P}{}{\prime}\rangle\langle\mp@subsup{Q}{}{\prime}~Q\rangle\mathrm{ have Q }~P\mathrm{ by(metis transitive symmetric)
            thus?thesis by(blast intro: reflexive)
        qed
    }
    with «(P,Q)\in?X> show ?case by blast
    qed
qed
end
theory Strong-Late-Bisim-Subst
    imports Strong-Late-Bisim
begin
abbreviation
    StrongEqJudge (infixr ~}\mp@subsup{~}{}{s}65)\mathrm{ where }P\mp@subsup{~}{}{s}Q\equiv(P,Q)\in(substClosed bisim)
lemma congBisim:
    fixes P :: pi
    and }Q::p
    assumes P ~}\mp@subsup{~}{}{s}
    shows P~Q
using assms substClosedSubset by blast
lemma eqvt:
    shows eqvt (substClosed bisim)
by(rule eqvtSubstClosed[OF Strong-Late-Bisim.bisimEqvt])
lemma eqClosed:
    fixes P :: pi
    and }Q:: p
    and perm :: name prm
    assumes P ~}\mp@subsup{~}{}{s}
    shows (perm • P) ~s (perm • Q)
using assms
by(rule eqvtRelI[OF eqvt])
lemma reflexive:
    fixes P :: pi
```

```
    shows P ~s}
by(force simp add: substClosed-def intro: Strong-Late-Bisim.reflexive)
lemma symmetric:
    fixes P :: pi
    and }Q::p
    assumes P ~}\mp@subsup{~}{}{s}
    shows Q ~s}
using assms
by(force simp add: substClosed-def intro: Strong-Late-Bisim.symmetric)
lemma transitive:
    fixes P :: pi
    and }Q:: p
    and }R::p
    assumes }P\mp@subsup{~}{}{s}
    and }\quadQ\mp@subsup{~}{}{s}
    shows }P\mp@subsup{~}{}{s}
using assms
by(force simp add: substClosed-def intro: Strong-Late-Bisim.transitive)
end
theory Strong-Late-Sim-Pres
    imports Strong-Late-Sim
begin
lemma tauPres:
    fixes P :: pi
    and }Q\quad::p
    and Rel :: (pi }\times pi) se
    and Rel' :: (pi\timespi) set
    assumes PRelQ: (P,Q)\inRel
    shows }\tau.(P)\rightsquigarrow[Rel] \tau.(Q
proof -
    show }\tau.(P)\rightsquigarrow[Rel] \tau.(Q
    proof(induct rule: simCases)
        case(Bound a x Q')
    have }\tau.(Q)\longmapstoa«x»\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    hence False by auto
    thus ?case by simp
    next
```

```
    case(Free \alpha Q )
    have }\tau.(Q)\longmapsto\alpha\prec ⿰Q' by fac
    thus ?case
    proof(induct rule: tauCases)
        case cTau
        have }\tau.(P)\longmapsto\tau\precP\mathrm{ by(rule Late-Semantics.Tau)
        with PRelQ show ?case by blast
    qed
    qed
qed
lemma inputPres:
    fixes P :: pi
    and }x\mathrm{ :: name
    and }Q :: p
    and a :: name
    and Rel ::(pi\times pi) set
    assumes PRelQ: }\forally.(P[x::=y],Q[x::=y])\in\operatorname{Rel
    and Eqvt: eqvt Rel
    shows }a<x>.P\rightsquigarrow[Rel] a<x>.Q
using Eqvt
proof(induct rule: simCasesCont [where C=(x,a,P,Q)])
    case(Bound b y Q')
    from < }y\sharp(x,a,P,Q)> have y\not=x y\not=ay\sharpPy\sharpQ by simp
    from }\langlea<x>.Q\longmapstob<<y» \prec ⿰Q '`\langley\not=a\rangle\langley\not= x\rangle\langley\sharpQ\rangle show ?cas
    proof(induct rule: inputCases)
    case cInput
    have }a<x>.P\longmapstoa<x>\precP by(rule Input
    hence }a<x>.P\longmapstoa<y>\prec ([(x,y)]\cdotP)\mathrm{ using < y #P>
        by(simp add: alphaBoundResidual)
    moreover have derivative ([(x,y)] P P) ([(x,y)] • Q) (InputS a) y Rel
    proof(auto simp add: derivative-def)
        fix u
        show }(([(x,y)]\cdotP)[y::=u],([(x,y)]\cdotQ)[y::=u])\in\operatorname{Rel
        proof(cases y=u)
            assume y=u
            moreover have ([(y,x)] • P, [(y,x)] •Q) \in Rel
            proof -
                from PRelQ have (P[x::=x], Q[x::=x]) \in Rel by blast
                hence (P,Q)\inRel by simp
                with Eqvt show ?thesis by(rule eqvtRelI)
            qed
            ultimately show ?thesis by simp
    next
                assume yinequ: y}\not=
```

```
        show ?thesis
        proof(cases x=u)
            assume x=u
            moreover have (([(y,x)] • P)[y::=x],([(y,x)] | Q)[y::=x])\in\operatorname{Rel}
            proof -
            from PRelQ have (P[x::=y], Q[x::=y]) \inRel by blast
            with Eqvt have }([(y,x)]\cdot(P[x::=y]),[(y,x)]\cdot(Q[x::=y]))\inRe
                by(rule eqvtRelI)
            with }\langley\not=x\rangle\mathrm{ show ?thesis
                by(simp add: eqvt-subs name-calc)
    qed
    ultimately show ?thesis by simp
next
    assume xinequ: }x\not=
    hence (([(y,x)] • P)[y::=u], ([(y,x)] • Q)[y::=u])\inRel
    proof -
            from PRelQ have (P[x::=u], Q[x::=u]) \in Rel by blast
            with Eqvt have ([(y,x)] • (P[x::=u]), [(y,x)] • (Q[x::=u])) \in Rel
                    by(rule eqvtRelI)
            with }<y\not=x> xinequ yinequ show ?thesi
                by(simp add: eqvt-subs name-calc)
            qed
            thus ?thesis by simp
        qed
        qed
    qed
    ultimately show ?case by blast
    qed
next
    case(Free \alpha Q ')
    have }a<x>.Q\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    hence False by auto
    thus?case by blast
qed
lemma outputPres:
    fixes P :: pi
    and }Q :: p
    and a :: name
    and b :: name
    and Rel :: (pi\times pi) set
    and Rel'::(pi \times pi) set
    assumes PRelQ: (P,Q)\inRel
    shows a{b}.P\rightsquigarrow[Rel] a{b}.Q
proof -
    show ?thesis
```

```
    proof (induct rule: simCases)
        case(Bound c \(x Q^{\prime}\) )
    have \(a\{b\} . Q \longmapsto c « x>\prec Q^{\prime}\) by fact
    hence False by auto
    thus ?case by simp
    next
        case \(\left(\right.\) Free \(\left.\alpha Q^{\prime}\right)\)
    have \(a\{b\} . Q \longmapsto \alpha \prec Q^{\prime}\) by fact
    thus ?case
    proof (induct rule: outputCases)
        case \(c\) Output
        have \(a\{b\} . P \longmapsto a[b] \prec P\) by (rule Late-Semantics.Output)
        with PRelQ show?case by blast
    qed
    qed
qed
lemma matchPres:
    fixes \(P\) :: \(p i\)
    and \(Q \quad:: p i\)
    and \(a\) :: name
    and \(b\) :: name
    and Rel :: \((p i \times p i)\) set
    and Rel' \(::(p i \times p i)\) set
    assumes \(\operatorname{PSimQ}: P \rightsquigarrow[R e l] Q\)
    and \(\quad R e l \subseteq R e l^{\prime}\)
    shows \([a \frown b] P \rightsquigarrow\left[\operatorname{Rel}^{\prime}\right][a \frown b] Q\)
proof -
    show ?thesis
    proof (induct rule: simCases)
    case(Bound cx \(Q^{\prime}\) )
    have \(x \sharp[a \frown b] P\) by fact
    hence \(x\) Fresh \(P: x \sharp P\) by \(\operatorname{simp}\)
    have \([a \frown b] Q \longmapsto c « x>\prec Q^{\prime}\) by fact
    thus ?case
    proof (induct rule: matchCases)
        case \(c\) Match
        have \(Q \longmapsto c « x\) » \(\prec Q^{\prime}\) by fact
        with PSimQ \(x\) Fresh \(P\) obtain \(P^{\prime}\) where PTrans: \(P \longmapsto c « x » \prec P^{\prime}\)
                                    and Pderivative: derivative \(P^{\prime} Q^{\prime}\) cx Rel
                    by (blast dest: simE)
    from PTrans have \([a \frown a] P \longmapsto c « x » \prec P^{\prime}\) by (rule Late-Semantics.Match)
    moreover from Pderivative \(\left\langle\operatorname{Rel} \subseteq \operatorname{Rel}^{\prime}\right\rangle\) have derivative \(P^{\prime} Q^{\prime}\) c \(x\) Rel \(^{\prime}\)
            by (cases c) (auto simp add: derivative-def)
            ultimately show ? case by blast
    qed
```

```
    next
        case(Free \alpha Q')
        have [a\frownb]Q\longmapsto\alpha\prec的' by fact
    thus ?case
    proof(induct rule: matchCases)
        case cMatch
        have }Q\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
        with PSimQ obtain }\mp@subsup{P}{}{\prime}\mathrm{ where PTrans: P}\longmapsto\alpha\prec\mp@subsup{P}{}{\prime
                                    and PRel: ( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mathrm{ Rel
            by(blast dest: simE)
        from PTrans have [a\frowna]P\longmapsto\alpha\longmapsto }\longmapsto\mp@subsup{P}{}{\prime}\mathrm{ by(rule Late-Semantics.Match)
        with PRel «Rel \subseteqRel'〉 show ?case by blast
        qed
    qed
qed
lemma mismatchPres:
fixes }P\mathrm{ :: pi
and }Q\quad::p
and a :: name
and b :: name
and Rel :: (pi\times pi) set
and Rel'::(pi\times pi) set
assumes PSimQ: P\rightsquigarrow[Rel] Q
and Rel\subseteqRel'
shows [a\not=b]P\rightsquigarrow[Rel ] [a\not=b]Q
proof(induct rule: simCases)
case(Bound c x Q')
have }x\sharp[a\not=b]P\mathrm{ by fact
hence xFreshP: x\sharpP by simp
from \[a\not=b]Q\longmapstoc«x» \prec ' '` show ?case
proof(induct rule: mismatchCases)
    case cMismatch
    have }Q\longmapstoc«x»\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    with PSimQ xFreshP obtain P' where PTrans: P\longmapstoc«x»\prec P'
                                    and Pderivative: derivative P' }\mp@subsup{P}{}{\prime}\mathrm{ с x Rel
            by(blast dest: simE)
    from PTrans }\langlea\not=b\rangle\mathrm{ have [a*b]Pط
        moreover from Pderivative \Rel \subseteqRel'> have derivative P' Q' c x Rel'
            by(cases c) (auto simp add: derivative-def)
        ultimately show ?case by blast
    qed
next
    case(Free \alpha Q ')
    have [a\not=b]Q\longmapsto\alpha\prec Q' by fact
    thus ?case
```

```
    proof(induct rule: mismatchCases)
    case cMismatch
    have }Q\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    with PSimQ obtain P' where PTrans: }P\longmapsto\alpha\prec\mp@subsup{P}{}{\prime
                            and PRel: ( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mathrm{ Rel
        by(blast dest: simE)
    from PTrans }\langlea\not=b\rangle\mathrm{ have [ }a\not=b]P\longmapsto\alpha\prec\mp@subsup{P}{}{\prime}\mathbf{by}(rule Late-Semantics.Mismatch
        with PRel <Rel \subseteqRel`` show ?case by blast
    qed
qed
lemma sumPres:
    fixes }P::p
    and }Q::p
    and }R::p
    assumes PSimQ: P}\rightsquigarrow[Rel] 
    and }Id\subseteqRel
    and Rel\subseteqRel'
    shows }P\oplusR\rightsquigarrow[Rel'] Q\oplus
proof -
    show ?thesis
    proof(induct rule: simCases)
    case(Bound a x QR)
    have }x\sharpP\oplusR\mathrm{ by fact
    hence xFreshP: x\sharpP and xFreshR: x }#R\mathrm{ by simp+
    have }Q\oplusR\longmapstoa«x>\precQR by fac
    thus ?case
    proof(induct rule:sumCases)
        case cSum1
        have }Q\longmapstoa<x>> \precQR by fac
        with xFreshP PSimQ obtain }\mp@subsup{P}{}{\prime}\mathrm{ where PTrans: Pطa«x» < P'
                                and Pderivative: derivative P' QR a x Rel
            by(blast dest: simE)
        from PTrans have P}\oplusR\longmapstoa«x»\prec\mp@subsup{P}{}{\prime}\mathbf{by}(rule Late-Semantics.Sum1)
        moreover from Pderivative 〈Rel \subseteqRel'> have derivative P' QR a x Rel'
            by(cases a) (auto simp add: derivative-def)
        ultimately show ?case by blast
    next
        case cSum2
        from 〈R\longmapstoa«x>\prec <QR\rangle have }P\oplusR\longmapstoa<x>\prec \precR by(rule Sum2)
        thus ?case using \Id \subseteqRel`> by(blast intro: derivativeReflexive)
    qed
next
    case(Free \alpha QR)
    have }Q\oplusR\longmapsto\alpha\precQR by fac
    thus ?case
```

```
    proof(induct rule: sumCases)
    case cSum1
    have}Q\longmapsto\alpha\precQR by fac
    with PSimQ obtain P' where PTrans: P\longmapsto\alpha \prec ' ' and PRel: ( }\mp@subsup{P}{}{\prime},QR
ERel
            by(blast dest: simE)
            from PTrans have P}\oplusR\longmapsto\alpha\prec\mp@subsup{P}{}{\prime}\mathrm{ by(rule Late-Semantics.Sum1)
            with PRel«Rel\subseteqRel`` show ?case by blast
        next
            case cSum2
            from }\langleR\longmapsto\alpha\precQR> have P \oplusR\longmapsto\alpha\prec QR by(rule Sum2)
            thus ?case using <Id \subseteqRel`> by(blast intro: derivativeReflexive)
    qed
    qed
qed
lemma parCompose:
fixes P :: pi
and }Q\quad::p
and }R\quad::p
and T :: pi
and Rel ::(pi\timespi) set
and Rel' :: (pi\timespi) set
and Rel" :: (pi\timespi) set
assumes PSimQ: }P\rightsquigarrow[\mathrm{ Rel ] Q
and RSimT: }R\rightsquigarrow[\mathrm{ Rel ] T
and PRelQ: }(P,Q)\in\mathrm{ Rel
and RRel'T: (R,T) \in Rel'
and Par: }\\PQRT.\llbracket(P,Q)\in\operatorname{Rel};(R,T)\in\operatorname{Rel}\rrbracket\Longrightarrow(P|R,Q|T
ERel"
and Res: }\PQa.(P,Q)\in\mp@subsup{Rel}{\prime\prime}{\Longrightarrow}\Longrightarrow(<\nua>P,<\nua>Q)\inRel"
and EqvtRel: eqvt Rel
and EqvtRel': eqvt Rel'
and EqvtRel": eqvt Rel"
shows P|R\rightsquigarrow[Rel'\]Q|T
using EqvtRel"
proof(induct rule: simCasesCont[where C=()])
case(Bound a x Q')
have }x\sharpP|R\mathrm{ and }x\sharpQ|T\mathrm{ by fact+
hence xFreshP: x\sharpP and xFreshR: }x\sharpR\mathrm{ and }x\sharpQ\mathrm{ and }x\sharpT\mathrm{ by simp+
have QTTrans: Q | T\longmapstoa«x»\prec < Q' by fact
thus ?case using <x\sharp Q>\langlex\sharpT\rangle
proof(induct rule: parCasesB)
    case(cPar1 Q')
    have QTrans: Q\longmapstoa<x»\prec Q Q' and xFreshT: x }\sharpT\mathrm{ by fact+
    from xFreshP PSimQ QTrans obtain }\mp@subsup{P}{}{\prime}\mathrm{ where PTrans:P}\longmapstoa<x»\prec\mp@subsup{P}{}{\prime
```

```
        by(blast dest: simE)
    from PTrans xFreshR have P|R\longmapstoa«x» \prec P'| R by(rule Late-Semantics.Par1B)
        moreover from Pderivative xFreshR xFreshT RRel'T have derivative ( }\mp@subsup{P}{}{\prime}
R)( Q'|T) a x Rel'"
        by(cases a, auto intro: Par simp add: derivative-def forget)
    ultimately show ?case by blast
next
    case(cPar2 T')
    have TTrans: T\longmapstoa«x»\prec T' and xFreshQ: x #Q by fact+
    from xFreshR RSimT TTrans obtain R' where RTrans:R\longmapstoa«x» \prec- R'
                    and Rderivative: derivative }\mp@subsup{R}{}{\prime}\mp@subsup{T}{}{\prime}\mathrm{ a x Rel'
        by(blast dest: simE)
    from RTrans xFreshP have ParTrans: P | R\longmapstoa«x»\prec P| | R' by(rule
Late-Semantics.Par2B)
    moreover from Rderivative xFreshP xFreshQ PRelQ have derivative ( }P|\mp@subsup{R}{}{\prime}\mathrm{ )
(Q| T') a x Rel''
    by(cases a, auto intro: Par simp add: derivative-def forget)
    ultimately show ?case by blast
    qed
next
case(Free \alpha QT')
have QTTrans: Q |T\longmapsto\alpha\longmapsto Q QT' by fact
thus ?case using PSimQ RSimT PRelQ RRel'T
proof(induct rule: parCasesF[where C=(P,R)])
    case(cPar1 Q')
    have RRel'T: (R,T) \in Rel' by fact
```



```
    then obtain }\mp@subsup{P}{}{\prime}\mathrm{ where PTrans: P}\longmapsto\alpha\prec\mp@subsup{P}{}{\prime}\mathrm{ and PRel: ( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mathrm{ Rel
        by(blast dest: simE)
    from PTrans have Trans: P|R\longmapsto\alpha\prec '靔|R by(rule Late-Semantics.Par1F)
    moreover from PRel RRel'T have ( }\mp@subsup{P}{}{\prime}|R,\mp@subsup{Q}{}{\prime}|T)\inRel/" by(blast intro
Par)
    ultimately show ?case by blast
    next
    case(cPar2 T')
    have PRelQ: (P,Q)\inRel by fact
    have }R\rightsquigarrow[\mp@subsup{Rel}{}{\prime}]T\mathrm{ and }T\longmapsto\alpha\prec\mp@subsup{T}{}{\prime}\mathrm{ by fact+
    then obtain }\mp@subsup{R}{}{\prime}\mathrm{ where RTrans: }R\longmapsto\alpha\prec\mp@subsup{R}{}{\prime}\mathrm{ and RRel: ( }\mp@subsup{R}{}{\prime},\mp@subsup{T}{}{\prime})\in\mp@subsup{R}{el}{\prime
        by(blast dest: simE)
    from RTrans have Trans: P|R\longmapsto\alpha\precP| R'\mathbf{by}(rule Late-Semantics.Par2F)
    moreover from PRelQ RRel have ( }P|\mp@subsup{R}{}{\prime},Q|\mp@subsup{T}{}{\prime})\in\mp@subsup{R}{\mathrm{ Rel' l by(blast intro:}}{
Par)
    ultimately show ?case by blast
next
    case(cComm1 Q' T' a b x)
    from \langlex\sharp (P,R)\rangle have }x\sharpP\mathrm{ by simp
```


obtain $P^{\prime}$ where PTrans: $P \longmapsto a<x>\prec P^{\prime}$
and Pderivative: derivative $P^{\prime} Q^{\prime}($ InputS $a) x$ Rel
by (blast dest: simE)
from Pderivative have PRel: $\left(P^{\prime}[x::=b], Q^{\prime}[x::=b]\right) \in$ Rel by $($ simp add: deriva-tive-def)
have $R \rightsquigarrow\left[\mathrm{Rel}^{\prime}\right] T$ and $T \longmapsto a[b] \prec T^{\prime}$ by fact +
then obtain $R^{\prime}$ where $R$ Trans: $R \longmapsto a[b] \prec R^{\prime}$ and $R R e l:\left(R^{\prime}, T^{\prime}\right) \in$ Rel $^{\prime}$ by (blast dest: simE)
from PTrans $R$ Trans have $P\left\|R \longmapsto \tau \prec P^{\prime}[x::=b]\right\| R^{\prime}$ by (rule Late-Semantics.Comm1)
moreover from PRel RRel have $\left(P^{\prime}[x::=b]\left\|R^{\prime}, Q^{\prime}[x::=b]\right\| T^{\prime}\right) \in$ Rel $^{\prime \prime}$
by (blast intro: Par)
ultimately show ?case by blast
next
case (cComm2 $Q^{\prime} T^{\prime}$ abl)
have $P \rightsquigarrow[R e l] Q$ and $Q \longmapsto a[b] \prec Q^{\prime}$ by fact +
then obtain $P^{\prime}$ where PTrans: $P \longmapsto a[b] \prec P^{\prime}$ and PRel: $\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}$ by (blast dest: simE)
from $\langle x \sharp(P, R)\rangle$ have $x \sharp R$ by simp
with $\langle R \rightsquigarrow[$ Rel $\left.] T\rangle\langle T \longmapsto a<x\rangle \prec T^{\prime}\right\rangle$
obtain $R^{\prime}$ where RTrans: $R \longmapsto a<x>\prec R^{\prime}$
and Rderivative: derivative $R^{\prime} T^{\prime}\left(\right.$ InputS a) $x$ Rel $^{\prime}$
by (blast dest: simE)
from Rderivative have RRel: $\left(R^{\prime}[x::=b], T^{\prime}[x::=b]\right) \in \operatorname{Rel}^{\prime}$ by (simp add: deriva-tive-def)
from PTrans RTrans have $P\left\|R \longmapsto \tau \prec P^{\prime}\right\| R^{\prime}[x::=b]$ by(rule Late-Semantics.Comm2)
moreover from PRel RRel have $\left(P^{\prime}\left\|R^{\prime}[x::=b], Q^{\prime}\right\| T^{\prime}[x::=b]\right) \in \operatorname{Rel}^{\prime \prime}$
by (blast intro: Par)
ultimately show $\exists P^{\prime} . P \| R \longmapsto \tau \prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime} \| T^{\prime}[x::=b]\right) \in R e l^{\prime \prime}$ by blast
next
case (cClose1 $Q^{\prime} T^{\prime}$ a $x y$ )
from $\langle x \sharp(P, R)\rangle$ have $x \sharp P$ by simp
with $\left.\langle P \rightsquigarrow[R e l] Q\rangle\langle Q \longmapsto a<x\rangle \prec Q^{\prime}\right\rangle$
obtain $P^{\prime}$ where PTrans: $P \longmapsto a<x>\prec P^{\prime}$
and Pderivative: derivative $P^{\prime} Q^{\prime}($ InputS a) x Rel
by (blast dest: simE)
from Pderivative have PRel: $\left(P^{\prime}[x::=y], Q^{\prime}[x::=y]\right) \in \operatorname{Rel}$ by $(\operatorname{simp}$ add: deriva-tive-def)
from $\langle y \sharp(P, R)\rangle$ have $y \sharp R$ and $y \sharp P$ by $\operatorname{simp}+$
from $\langle R \rightsquigarrow[$ Rel $\left.] T\rangle\langle T \longmapsto a<\nu y\rangle \prec T^{\prime}\right\rangle\langle y \sharp R\rangle$
obtain $R^{\prime}$ where $R$ Trans: $R \longmapsto a<\nu y>\prec R^{\prime}$
and Rderivative: derivative $R^{\prime} T^{\prime}\left(\right.$ BoundOutputS a) y Rel ${ }^{\prime}$
by(blast dest: $\operatorname{simE}$ )
from Rderivative have RRel: $\left(R^{\prime}, T^{\prime}\right) \in \operatorname{Rel}^{\prime} \mathbf{b y}(\operatorname{simp}$ add: derivative-def)
from PTrans RTrans $\left\langle y \sharp P>\right.$ have Trans: $P \| R \longmapsto \tau \prec<\nu y>\left(P^{\prime}[x::=y] \|\right.$ $R^{\prime}$ )
by(rule Late-Semantics.Close1)
moreover from PRel RRel have $\left(<\nu y>\left(P^{\prime}[x::=y] \| R^{\prime}\right),<\nu y>\left(Q^{\prime}[x::=y] \|\right.\right.$ $\left.\left.T^{\prime}\right)\right) \in R e l^{\prime \prime}$
by (blast intro: Par Res)
ultimately show ?case by blast
next
case (cClose2 $Q^{\prime} T^{\prime}$ a $x y$ )
from $\langle y \sharp(P, R)\rangle$ have $y \sharp P$ and $y \sharp R$ by simp +
from $\left.\langle P \rightsquigarrow[R e l] Q\rangle\langle Q \longmapsto a<\nu y\rangle \prec Q^{\prime}\right\rangle\langle y \sharp P\rangle$
obtain $P^{\prime}$ where PTrans: $P \longmapsto a<\nu y>\prec P^{\prime}$ and $P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}$ $\mathbf{b y}($ force dest: simE simp add: derivative-def)
from $\langle x \sharp(P, R)\rangle$ have $x \sharp R$ by simp +
with $\left\langle R \rightsquigarrow\left[\right.\right.$ Rel $\left.\left.\left.{ }^{\prime}\right] T\right\rangle\langle T \longmapsto a<x\rangle \prec T^{\prime}\right\rangle$
obtain $R^{\prime}$ where RTrans: $R \longmapsto a<x>\prec R^{\prime}$
and $R^{\prime} \operatorname{Rel}^{\prime} T^{\prime}:\left(R^{\prime}[x::=y], T^{\prime}[x::=y]\right) \in \operatorname{Rel}^{\prime}$
by (force dest: simE simp add: derivative-def)
from PTrans RTrans $\langle y \sharp R\rangle$ have Trans: $P \| R \longmapsto \tau \prec<\nu y>\left(P^{\prime} \| R^{\prime}[x::=y]\right)$ by(rule Close2)
moreover from $P^{\prime} \operatorname{Rel} Q^{\prime} R^{\prime} \operatorname{Rel}^{\prime} T^{\prime}$ have $\left(<\nu y>\left(P^{\prime} \| R^{\prime}[x::=y]\right),<\nu y>\left(Q^{\prime} \|\right.\right.$ $\left.\left.T^{\prime}[x::=y]\right)\right) \in \operatorname{Rel}^{\prime \prime}$
by (blast intro: Par Res)
ultimately show ?case by blast
qed
qed
lemma parPres:
fixes $P$ :: $p i$
and $\quad Q \quad:: p i$
and $\quad R \quad:: p i$
and $a$ :: name
and $b$ :: name
and Rel $::(p i \times p i)$ set
and Rel' $::(p i \times p i)$ set
assumes $\operatorname{PSimQ}: \quad P \rightsquigarrow[$ Rel $] Q$
and PRelQ: $\quad(P, Q) \in$ Rel
and Par: $\quad \bigwedge P Q R .(P, Q) \in \operatorname{Rel} \Longrightarrow(P\|R, Q\| R) \in \operatorname{Rel}^{\prime}$
and Res: $\bigwedge P Q a .(P, Q) \in \operatorname{Rel}^{\prime} \Longrightarrow(<\nu a>P,<\nu a>Q) \in \operatorname{Rel}^{\prime}$
and EqvtRel: eqvt Rel
and EqvtRel': eqvt Rel'
shows $P\left\|R \rightsquigarrow\left[\mathrm{Rel}^{\dagger}\right] Q\right\| R$
proof -
note $\operatorname{PSimQ}$
moreover have $R \operatorname{Sim} R: R \rightsquigarrow[I d] R$ by(auto intro: reflexive)
moreover note $P R e l Q$ moreover have $(R, R) \in I d$ by auto
moreover from Par have $\Lambda P Q R T . \llbracket(P, Q) \in \operatorname{Rel} ;(R, T) \in I d \rrbracket \Longrightarrow(P \|$
$R, Q \| T) \in$ Rel $^{\prime}$
by auto
moreover note Res 〈eqvt Rel〉
moreover have eqvt Id by (auto simp add: eqvt-def)
ultimately show ?thesis using EqvtRel' by(rule parCompose)
qed
lemma resDerivative:
fixes $P \quad:: p i$
and $\quad Q \quad:: p i$
and $a$ :: subject
and $x$ :: name
and $y$ :: name
and Rel $::(p i \times p i)$ set
and Rel $::(p i \times p i)$ set
assumes Der: derivative P Q a x Rel
and Rel: $\bigwedge(P:: p i)(Q:: p i)(x:: n a m e) .(P, Q) \in \operatorname{Rel} \Longrightarrow(<\nu x>P,<\nu x>Q) \in$ $R e l^{\prime}$
and Eqv: equt Rel
shows derivative $(<\nu y>P)(<\nu y>Q)$ a $x$ Rel $^{\prime}$
proof -
from Der Rel show ?thesis
proof(cases a, auto simp add: derivative-def)
fix $u$
assume $A 1: \forall u .(P[x::=u], Q[x::=u]) \in \operatorname{Rel}$
show $((<\nu y>P)[x::=u],(<\nu y>Q)[x::=u]) \in$ Rel $^{\prime}$
proof (cases $x=y$ )
assume xeqy: $x=y$
from A1 have $(P[x::=x], Q[x::=x]) \in$ Rel by blast
hence L1: $(<\nu y>P,<\nu y>Q) \in$ Rel $^{\prime}$ by (force intro: Rel)
have $y \sharp<\nu y>P$ and $y \sharp<\nu y>Q$ by(simp only: freshRes) +
hence $(<\nu y>P)[y::=u]=<\nu y>P$ and $(<\nu y>Q)[y::=u]=<\nu y>Q$ by $(\operatorname{simp}$
add: forget) +
with L1 xeqy show ?thesis by simp
next
assume xineqy: $x \neq y$
show ?thesis
proof (cases $y=u$ )
assume yequ: $y=u$
have $\exists(c:: n a m e) . c \sharp(P, Q, x, y)$ by (blast intro: name-exists-fresh)
then obtain $c$ where cFreshP: $c \sharp P$ and $c$ Fresh $Q: c \sharp Q$ and cineqx: $c \neq$ $x$ and cineqy: $y \neq c$
by (force simp add: fresh-prod name-fresh)
from A1 have $(P[x::=c], Q[x::=c]) \in$ Rel by blast
with Eqv have $([(y, c)] \cdot(P[x::=c]),[(y, c)] \cdot(Q[x::=c])) \in$ Rel by $($ rule eqvtRelI)
with xineqy cineqx cineqy have $(([(y, c)] \cdot P)[x::=y],([(y, c)] \cdot Q)[x::=y])$ $\in$ Rel
by (simp add: eqvt-subs name-calc)
hence $(<\nu c>(([(y, c)] \cdot P)[x::=y]),<\nu c\rangle(([(y, c)] \cdot Q)[x::=y])) \in$ Rel $^{\prime}$ by (rule Rel)
with cineqx cineqy have $((<\nu c>(([(y, c)] \cdot P)))[x::=y],(<\nu c>(([(y, c)] \cdot$ $Q)))[x::=y]) \in$ Rel $^{\prime}$ by $\operatorname{simp}$
moreover from $c$ Fresh $P$ cFresh $Q$ have $<\nu c\rangle([(y, c)] \cdot P)=<\nu y>P$ and $<\nu c>([(y, c)] \cdot Q)=<\nu y>Q$
by (simp add: alphaRes)+
ultimately show ?thesis using yequ by simp
next
assume yinequ: $y \neq u$
from A1 have $(P[x::=u], Q[x::=u]) \in$ Rel by blast
hence $(<\nu y>(P[x::=u]),<\nu y>(Q[x::=u])) \in$ Rel $^{\prime}$ by (rule Rel)
with xineqy yinequ show?thesis by simp
qed
qed
qed
qed
lemma resPres:
fixes $P \quad:: p i$
and $\quad Q \quad:: p i$
and Rel $::(p i \times p i)$ set
and $x$ :: name
and Rel': : $(p i \times p i)$ set
assumes $\operatorname{PSimQ}: P \rightsquigarrow[R e l] Q$
and ResRel: $\bigwedge(P:: p i)(Q:: p i)(x:: n a m e) .(P, Q) \in \operatorname{Rel} \Longrightarrow(<\nu x>P,<\nu x>Q)$
$\in R e l^{\prime}$
and RelRel': Rel $\subseteq$ Rel $^{\prime}$
and EqvtRel: equt Rel
and EqvtRel': eqvt Rel'
shows $<\nu x>P \rightsquigarrow\left[\right.$ Rel $\left.^{\prime}\right]<\nu x>Q$
using EqvtRel'
proof $($ induct rule: resSimCases $[$ of $-\cdots(P, x)])$
case(BoundOutput $Q^{\prime}$ a)
have $Q$ Trans: $Q \longmapsto a[x] \prec Q^{\prime}$ and aineqx: $a \neq x$ by fact +

```
from \(P \operatorname{Sim} Q\) QTrans obtain \(P^{\prime}\) where PTrans: \(P \longmapsto a[x] \prec P^{\prime}\)
    and \(P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}\)
    by (blast dest: \(\operatorname{sim} E)\)
```

```
from PTrans aineqx have \(<\nu x>P \longmapsto a<\nu x>\prec P^{\prime}\) by (rule Late-Semantics.Open)
```

from PTrans aineqx have $<\nu x>P \longmapsto a<\nu x>\prec P^{\prime}$ by (rule Late-Semantics.Open)
moreover from $P^{\prime}$ RelQ' RelRel' have $\left(P^{\prime}, Q^{\prime}\right) \in$ Rel $^{\prime}$ by force
moreover from $P^{\prime}$ RelQ' RelRel' have $\left(P^{\prime}, Q^{\prime}\right) \in$ Rel $^{\prime}$ by force
ultimately show ?case by blast
ultimately show ?case by blast
next
next
case(BoundR $Q^{\prime}$ a $y$ )
case(BoundR $Q^{\prime}$ a $y$ )
have $Q$ Trans: $Q \longmapsto a « y » \prec Q^{\prime}$ and $x$ Fresha: $x \sharp a$ by fact +
have $Q$ Trans: $Q \longmapsto a « y » \prec Q^{\prime}$ and $x$ Fresha: $x \sharp a$ by fact +
have $y \sharp(P, x)$ by fact
have $y \sharp(P, x)$ by fact
hence $y$ FreshP: $y \sharp P$ and yineqx: $y \neq x$ by(simp add: fresh-prod $)+$
hence $y$ FreshP: $y \sharp P$ and yineqx: $y \neq x$ by(simp add: fresh-prod $)+$
from PSimQ yFreshP QTrans obtain $P^{\prime}$ where PTrans: $P \longmapsto a « y » \prec P^{\prime}$
from PSimQ yFreshP QTrans obtain $P^{\prime}$ where PTrans: $P \longmapsto a « y » \prec P^{\prime}$
and Pderivative: derivative $P^{\prime} Q^{\prime}$ a y Rel
and Pderivative: derivative $P^{\prime} Q^{\prime}$ a y Rel
by(blast dest: $\operatorname{sim} E$ )
by(blast dest: $\operatorname{sim} E$ )
from PTrans xFresha yineqx have ResTrans: $\left\langle\nu x>P \longmapsto a « y » \prec<\nu x>P^{\prime}\right.$
from PTrans xFresha yineqx have ResTrans: $\left\langle\nu x>P \longmapsto a « y » \prec<\nu x>P^{\prime}\right.$
by (blast intro: Late-Semantics.ResB)
by (blast intro: Late-Semantics.ResB)
moreover from Pderivative ResRel EqvtRel have derivative $\left.\left(<\nu x>P^{\prime}\right)(<\nu x\rangle Q^{\prime}\right)$
moreover from Pderivative ResRel EqvtRel have derivative $\left.\left(<\nu x>P^{\prime}\right)(<\nu x\rangle Q^{\prime}\right)$
a y Rel'
a y Rel'
by(rule resDerivative)
by(rule resDerivative)
ultimately show ?case by blast
ultimately show ?case by blast
next
next
case(FreeR $Q^{\prime} \alpha$ )
case(FreeR $Q^{\prime} \alpha$ )
have $Q$ Trans: $Q \longmapsto \alpha \prec Q^{\prime}$ and xFreshAlpha: $(x::$ name $) \sharp \alpha$ by fact +
have $Q$ Trans: $Q \longmapsto \alpha \prec Q^{\prime}$ and xFreshAlpha: $(x::$ name $) \sharp \alpha$ by fact +
from $Q$ Trans PSimQ obtain $P^{\prime}$ where PTrans: $P \longmapsto \alpha \prec P^{\prime}$
from $Q$ Trans PSimQ obtain $P^{\prime}$ where PTrans: $P \longmapsto \alpha \prec P^{\prime}$
and $P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}$
and $P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}$
$\mathbf{b y}($ blast dest: $\operatorname{sim} E)$
$\mathbf{b y}($ blast dest: $\operatorname{sim} E)$
from PTrans xFreshAlpha have $<\nu x>P \longmapsto \alpha \prec<\nu x>P^{\prime}$ by (rule Late-Semantics.ResF)
moreover from $P^{\prime} \operatorname{Rel} Q^{\prime}$ have $\left(<\nu x>P^{\prime},<\nu x>Q^{\prime}\right) \in \operatorname{Rel} l^{\prime}$ by (rule ResRel)
ultimately show ?case by blast
qed
lemma resChainI:
fixes $P$ :: $p i$
and $\quad Q \quad:: p i$
and Rel :: $(p i \times p i)$ set
and $x s$ :: name list
assumes PRelQ: $\quad P \rightsquigarrow[$ Rel $] Q$
and eqvtRel: eqvt Rel
and Res: $\quad \bigwedge P Q x .(P, Q) \in \operatorname{Rel} \Longrightarrow(<\nu x>P,<\nu x>Q) \in$ Rel
shows (resChain xs) $P \rightsquigarrow[$ Rel $]($ resChain $x s) ~ Q$
proof (induct $x s$ )
from PRelQ show resChain [] $P \rightsquigarrow[$ Rel ] resChain [] $Q$ by simp

```
```

next
fix x xs
assume IH:(resChain xs P)}\rightsquigarrow[Rel] (resChain xs Q)
moreover note Res
moreover have Rel \subseteqRel by simp
ultimately have <\nux>(resChain xs P)\rightsquigarrow[Rel]<\nux>(resChain xs Q) using
equtRel
by(rule-tac resPres)
thus resChain (x \# xs) P\rightsquigarrow[Rel] resChain (x \# xs) Q
by simp
qed
lemma bangPres:
fixes P :: pi
and }Q :: p
and Rel :: (pi\times pi) set
assumes PRelQ: }\quad(P,Q)\in\mathrm{ Rel
and Sim: }\PQ.(P,Q)\inRel\LongrightarrowP\rightsquigarrow[Rel] Q
and eqvtRel: eqvt Rel
shows !P \rightsquigarrow[bangRel Rel] !Q
proof -
let ?Sim = \lambdaP Rs. (\forallax Q '. Rs=a«x»\prec Q Q \longrightarrow
\prec P'^ derivative P' Q' a x (bangRel Rel))) ^
(\forall\alpha Q'.Rs=\alpha\prec Q'\longrightarrow(\exists\mp@subsup{P}{}{\prime}.P\longmapsto\alpha\prec 在'^( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})
bangRel Rel))
from eqvtRel have EqvtBangRel: eqvt(bangRel Rel) by(rule eqvtBangRel)
{
fix Pa Rs
assume !Q\longmapstoRs and (Pa,!Q) \in bangRel Rel
hence ?Sim Pa Rs using PRelQ
proof(nominal-induct avoiding: Pa P rule: bangInduct)
case(cPar1B a x Q Pa P)
have QTrans: Q\longmapstoa«x» \prec- Q' by fact
have (Pa,Q|!Q)\in bangRel Rel and x\sharpPa by fact+
thus ?Sim Pa (a<x>\prec < (Q'|! !Q))
proof(induct rule: BRParCases)
case(BRPar P R)
have PRelQ: (P,Q)\in Rel by fact
have PBRQ:(R,!Q)\in bangRel Rel by fact
have }x\sharpP||R\mathrm{ by fact
hence xFreshP: }x\sharpP\mathrm{ and xFreshR: }x\sharpR\mathrm{ by simp+
show ?case
proof(auto simp add: residual.inject alpha')
from PRelQ have P\rightsquigarrow[Rel] Q by(rule Sim)

```
with QTrans xFreshP obtain \(P^{\prime}\) where PTrans: \(P \longmapsto a « x » \prec P^{\prime}\) and \(P^{\prime}\) RelQ': derivative \(P^{\prime} Q^{\prime}\) a \(x\) Rel by (blast dest: \(\operatorname{sim} E)\)
from PTrans xFresh \(R\) have \(P \| R \longmapsto a « x » \prec\left(P^{\prime} \| R\right)\)
by (force intro: Late-Semantics.Par1B)
moreover from \(P^{\prime} \operatorname{Rel} Q^{\prime} P B R Q\langle x \sharp Q\rangle\langle x \sharp R\rangle\) have derivative \(\left(P^{\prime} \|\right.\)
\(R)\left(Q^{\prime} \|!Q\right)\) a \(x(\) bangRel Rel \()\)
by (cases a) (auto simp add: derivative-def forget intro: Rel.BRPar)
ultimately show \(\exists P^{\prime} . P \| R \longmapsto a « x » \prec P^{\prime} \wedge\) derivative \(P^{\prime}\left(Q^{\prime} \|!Q\right) a\) \(x\) (bangRel Rel) by blast
next
fix \(y\)
assume ( \(y\) ::name) \(\sharp Q^{\prime}\) and \(y \sharp P\) and \(y \sharp R\) and \(y \sharp Q\)
from QTrans \(\left\langle y \sharp Q^{\prime}\right\rangle\) have \(Q \longmapsto a « y » \prec\left([(x, y)] \cdot Q^{\prime}\right)\)
by (simp add: alphaBoundResidual)
moreover from \(P R e l Q\) have \(P \rightsquigarrow[R e l] Q\) by (rule Sim)
ultimately obtain \(P^{\prime}\) where PTrans: \(P \longmapsto a « y » \prec P^{\prime}\) and \(P^{\prime}\) RelQ \({ }^{\prime}\) : derivative \(P^{\prime}\left([(x, y)] \cdot Q^{\prime}\right)\) a y Rel
using \(\langle y \sharp P\rangle\)
by (blast dest: simE)
from PTrans \(\langle y \sharp R\rangle\) have \(P \| R \longmapsto a « y » \prec\left(P^{\prime} \| R\right)\) by (force intro: Late-Semantics.Par1B)
moreover from \(P^{\prime} \operatorname{Rel} Q^{\prime} P B R Q\langle y \sharp Q\rangle\langle y \sharp R\rangle\) have derivative \(\left(P^{\prime} \| R\right)\) \(\left(\left([(x, y)] \cdot Q^{\prime}\right) \|!Q\right)\) a \(y\) (bangRel Rel)
by (cases a) (auto simp add: derivative-def forget intro: Rel.BRPar)
with \(\langle x \sharp Q\rangle\langle y \sharp Q\rangle\) have derivative \(\left(P^{\prime} \| R\right)\left(\left([(y, x)] \cdot Q^{\prime}\right) \|!([(y, x)]\right.\)
- Q)) a y (bangRel Rel)
by (simp add: name-fresh-fresh name-swap)
ultimately show \(\exists P^{\prime} . P \| R \longmapsto a « y » \prec P^{\prime} \wedge\) derivative \(P^{\prime}(([(y, x)] \cdot\) \(\left.\left.Q^{\prime}\right) \|!([(y, x)] \cdot Q)\right)\) a \(y(\) bangRel Rel \()\) by blast qed
qed
next
case (cPar1F \(\left.\alpha Q^{\prime} P a P\right)\)
have \(Q\) Trans: \(Q \longmapsto \alpha \prec Q^{\prime}\) by fact
have \((P a, Q \|!Q) \in\) bangRel Rel by fact
thus ?case
proof (induct rule: BRParCases)
case (BRPar P R)
have PRelQ: \((P, Q) \in\) Rel and \(B R:(R,!Q) \in\) bangRel Rel by fact +
show ? case
proof (auto simp add: residual.inject)
from PRelQ have \(P \rightsquigarrow[\) Rel \(] Q\) by (rule Sim)
with QTrans obtain \(P^{\prime}\) where PTrans: \(P \longmapsto \alpha \prec P^{\prime}\) and RRel: \(\left(P^{\prime}\right.\), \(\left.Q^{\prime}\right) \in \operatorname{Rel}\)
by (blast dest: simE)
from PTrans have \(P\left\|R \longmapsto \alpha \prec P^{\prime}\right\| R\) by (rule Par1F)
moreover from RRel \(B R\) have \(\left(P^{\prime}\left\|R, Q^{\prime}\right\|!Q\right) \in\) bangRel Rel by (rule Rel.BRPar)
ultimately show \(\exists P^{\prime} . P \| R \longmapsto \alpha \prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime} \|!Q\right) \in\) bangRel Rel by blast
qed
qed
next case(cPar2B ax \(\left.Q^{\prime} P a P\right)\)
hence \(I H: \bigwedge P a .(P a,!Q) \in\) bangRel Rel \(\Longrightarrow\) ?Sim \(P a\left(a « x » \prec Q^{\prime}\right)\) by simp have \((P a, Q \|!Q) \in\) bangRel Rel and \(x \sharp P a\) by fact +
thus ? \(\operatorname{Sim} P a\left(a « x » \prec\left(Q \| Q^{\prime}\right)\right)\)
proof (induct rule: BRParCases)
case (BRPar P R)
have PRelQ: \((P, Q) \in\) Rel and \(R B R Q:(R,!Q) \in\) bangRel Rel by fact + have \(x \sharp P \| R\) by fact
hence \(x\) Fresh \(P: x \sharp P\) and \(x\) Fresh \(R\) : \(x \sharp R\) by simp +
from EqvtBangRel \(\langle x \sharp Q\rangle\) show ? \(\operatorname{Sim}(P \| R)\left(a « x » \prec\left(Q \| Q^{\prime}\right)\right)\)
proof (auto simp add: residual.inject alpha' name-fresh-fresh)
from \(R B R Q\) have ?Sim \(R\left(a « x » \prec Q^{\prime}\right)\) by (rule \(\left.I H\right)\)
with \(x\) Fresh \(R\) obtain \(R^{\prime}\) where RTrans: \(R \longmapsto a « x » \prec R^{\prime}\) and \(R^{\prime} B R Q^{\prime}\) : derivative \(R^{\prime} Q^{\prime}\) a \(x\) (bangRel Rel)
by(auto simp add: residual.inject)
from RTrans xFreshP have \(P \| R \longmapsto a « x » \prec\left(P \| R^{\prime}\right)\) by (auto intro: Par2B)
moreover from PRelQ \(R^{\prime} B R Q^{\prime}\langle x \sharp Q\rangle\langle x \sharp P\rangle\) have derivative \(\left(P \| R^{\prime}\right)\)
\(\left(Q \| Q^{\prime}\right)\) a \(x(\) bangRel Rel)
by (cases a) (auto simp add: derivative-def forget intro: Rel.BRPar)
ultimately show \(\exists P^{\prime} . P \| R \longmapsto a « x » \prec P^{\prime} \wedge\) derivative \(P^{\prime}\left(Q \| Q^{\prime}\right) a\) \(x\) (bangRel Rel) by blast
next
fix \(y\)
assume \((y:: n a m e) \sharp Q\) and \(y \sharp Q^{\prime}\) and \(y \sharp P\) and \(y \sharp R\)
from \(R B R Q\) have ?Sim \(\left.R(a « x\rangle \prec Q^{\prime}\right) \mathbf{b y}(\) rule \(I H)\)
with \(\left\langle y \sharp Q^{\prime}\right\rangle\) have ? Sim \(R\left(a « y » \prec\left([(x, y)] \cdot Q^{\prime}\right)\right)\) by \((\operatorname{simp}\) add: alphaBoundResidual)
with \(\langle y \sharp R\rangle\) obtain \(R^{\prime}\) where RTrans: \(R \longmapsto a « y » \prec R^{\prime}\) and \(R^{\prime} B R Q^{\prime}\) : derivative \(R^{\prime}\left([(x, y)] \cdot Q^{\prime}\right)\) a y (bangRel Rel)
by (auto simp add: residual.inject)
from RTrans \(\langle y \sharp P\rangle\) have \(P \| R \longmapsto a « y » \prec\left(P \| R^{\prime}\right) \mathbf{b y}\) (auto intro: Par2B)
moreover from \(\operatorname{PRelQ} R^{\prime} B R Q^{\prime}\langle y \sharp P\rangle\langle y \sharp Q\rangle\) have derivative \(\left(P \| R^{\prime}\right)\)
\(\left(Q \|\left([(x, y)] \cdot Q^{\prime}\right)\right)\) a \(y\) (bangRel Rel)
by(cases a) (auto simp add: derivative-def forget intro: Rel.BRPar)
hence derivative \(\left(P \| R^{\prime}\right)\left(Q \|\left([(y, x)] \cdot Q^{\prime}\right)\right)\) a \(y\) (bangRel Rel)
by (simp add: name-swap)
ultimately show \(\exists P^{\prime} . P \| R \longmapsto a « y » \prec P^{\prime} \wedge\) derivative \(P^{\prime}(Q \|([(y\), \(\left.\left.x)] \cdot Q^{\prime}\right)\right)\) a \(y(\) bangRel Rel) by blast
qed
qed
next
case \(\left(c P a r 2 F \alpha Q^{\prime} P a P\right)\)
hence \(I H: \wedge P a .(P a,!Q) \in\) bangRel Rel \(\Longrightarrow\) ?Sim Pa \(\left(\alpha \prec Q^{\prime}\right)\) by simp
have \((P a, Q \|!Q) \in\) bangRel Rel by fact
thus ?case
proof (induct rule: BRParCases)
case (BRPar P R)
have PRelQ: \((P, Q) \in\) Rel and \(R B R Q:(R,!Q) \in\) bangRel Rel by fact + show ? case
proof (auto simp add: residual.inject)
from \(R B R Q\) IH have \(\exists R^{\prime} . R \longmapsto \alpha \prec R^{\prime} \wedge\left(R^{\prime}, Q^{\prime}\right) \in\) bangRel Rel by (metis simE)
then obtain \(R^{\prime}\) where RTrans: \(R \longmapsto \alpha \prec R^{\prime}\) and \(R^{\prime} \operatorname{Rel} Q^{\prime}:\left(R^{\prime}, Q^{\prime}\right) \in\) bangRel Rel
by blast
from \(R\) Trans have \(P\|R \longmapsto \alpha \prec P\| R^{\prime}\) by (rule Par2F)
moreover from PRelQ \(R^{\prime}\) Rel \(Q^{\prime}\) have \(\left(P\left\|R^{\prime}, Q\right\| Q^{\prime}\right) \in\) bangRel Rel
by(rule Rel.BRPar)
ultimately show \(\exists P^{\prime} . P \| R \longmapsto \alpha \prec P^{\prime} \wedge\left(P^{\prime}, Q \| Q^{\prime}\right) \in\) bangRel Rel
by blast
qed
qed
next
case \(\left(c\right.\) Comm1 a x \(Q^{\prime} b Q^{\prime \prime}\) Pa \(\left.P\right)\)
hence IH: \(\bigwedge P a .(P a,!Q) \in\) bangRel Rel \(\Longrightarrow\) ?Sim Pa \(\left(a[b] \prec Q^{\prime \prime}\right)\) by simp
have \(Q\) Trans: \(Q \longmapsto a<x>\prec Q^{\prime}\) by fact
have \((P a, Q \|!Q) \in\) bangRel Rel by fact
thus ? case using \(\langle x \sharp P a\rangle\)
proof (induct rule: BRParCases)
case (BRPar P R)
have PRelQ: \((P, Q) \in\) Rel and \(R B R Q:(R,!Q) \in\) bangRel Rel by fact + from \(\langle x \sharp P \| R\rangle\) have \(x \sharp P\) and \(x \sharp R\) by simp +
show ? case
proof (auto simp add: residual.inject)
from \(\operatorname{PRelQ}\) have \(P \rightsquigarrow[\) Rel \(] Q\) by (rule Sim)
with \(Q\) Trans \(\langle x \sharp P\rangle\) obtain \(P^{\prime}\) where PTrans: \(P \longmapsto a<x>\prec P^{\prime}\) and \(P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}[x::=b], Q^{\prime}[x::=b]\right) \in \operatorname{Rel}\)
by (drule-tac simE) (auto simp add: derivative-def)
from \(I H R B R Q\) have RTrans: \(\exists R^{\prime} . R \longmapsto a[b] \prec R^{\prime} \wedge\left(R^{\prime}, Q^{\prime \prime}\right) \in\) bangRel
Rel
by (auto simp add: derivative-def)
then obtain \(R^{\prime}\) where \(R\) Trans: \(R \longmapsto a[b] \prec R^{\prime}\) and \(R^{\prime} \operatorname{Rel} Q^{\prime \prime}:\left(R^{\prime}, Q^{\prime \prime}\right)\) \(\in\) bangRel Rel
by blast
```

            from PTrans RTrans have \(P\left\|R \longmapsto \tau \prec P^{\prime}[x::=b]\right\| R^{\prime}\) by (rule Comm1)
            moreover from \(P^{\prime} R e l Q^{\prime} R^{\prime} R e l Q^{\prime \prime}\) have \(\left(P^{\prime}[x::=b]\left\|R^{\prime}, Q^{\prime}[x::=b]\right\| Q^{\prime \prime}\right)\)
    $\in$ bangRel Rel by(rule Rel.BRPar)
ultimately show $\exists P^{\prime} . P \| R \longmapsto \tau \prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}[x::=b] \| Q^{\prime \prime}\right) \in$
bangRel Rel by blast
qed
qed
next
case (cComm2 a b $Q^{\prime} x Q^{\prime \prime} P a P$ )
hence IH: $\wedge P a .(P a,!Q) \in$ bangRel Rel $\Longrightarrow$ ?Sim Pa $\left(a<x>\prec Q^{\prime \prime}\right)$ by simp
have $Q$ Trans: $Q \longmapsto a[b] \prec Q^{\prime}$ by fact
have $(P a, Q \|!Q) \in$ bangRel Rel by fact
thus ? case using $\langle x \sharp P a\rangle$
proof (induct rule: BRParCases)
case (BRPar P R)
have PRelQ: $(P, Q) \in$ Rel and $R B R Q:(R,!Q) \in$ bangRel Rel by fact +
from $\langle x \sharp P \| R\rangle$ have $x \sharp P$ and $x \sharp R$ by simp +
show? case
proof (auto simp add: residual.inject)
from PRelQ have $P \rightsquigarrow[$ Rel $] Q$ by (rule Sim)
with QTrans obtain $P^{\prime}$ where PTrans: $P \longmapsto a[b] \prec P^{\prime}$ and $P^{\prime}$ RelQ':
$\left(P^{\prime}, Q^{\prime}\right) \in$ Rel
by (blast dest: simE)
from $I H R B R Q\langle x \sharp R\rangle$ have RTrans: $\exists R^{\prime} . R \longmapsto a<x>\prec R^{\prime} \wedge\left(R^{\prime}[x::=b]\right.$,
$\left.Q^{\prime \prime}[x::=b]\right) \in$ bangRel Rel
by(fastforce simp add: derivative-def residual.inject)
then obtain $R^{\prime}$ where RTrans: $R \longmapsto a<x>\prec R^{\prime}$ and $R^{\prime}$ RelQ':
$\left(R^{\prime}[x::=b], Q^{\prime \prime}[x::=b]\right) \in$ bangRel Rel
by blast
from PTrans RTrans have $P\left\|R \longmapsto \tau \prec P^{\prime}\right\| R^{\prime}[x::=b]$ by(rule Comm2)
moreover from $P^{\prime} \operatorname{Rel} Q^{\prime} R^{\prime} \operatorname{Rel} Q^{\prime \prime}$ have $\left(P^{\prime}\left\|R^{\prime}[x::=b], Q^{\prime}\right\| Q^{\prime \prime}[x::=b]\right)$
$\in$ bangRel Rel by(rule Rel.BRPar)
ultimately show $\exists P^{\prime} . P \| R \longmapsto \tau \prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime} \|\left(Q^{\prime \prime}[x::=b]\right)\right) \in$
bangRel Rel by blast
qed
qed
next
case (cClose1 a x $Q^{\prime}$ y $Q^{\prime \prime}$ Pa $P$ )
hence $I H: \wedge P a .(P a,!Q) \in$ bangRel Rel $\longrightarrow$ ?Sim Pa $\left(a<\nu y>\prec Q^{\prime \prime}\right)$ by
simp
have $Q$ Trans: $Q \longmapsto a<x>\prec Q^{\prime}$ by fact
have $(P a, Q \|!Q) \in$ bangRel Rel by fact
moreover have $x F r e s h P a$ : $x \sharp P a$ by fact
ultimately show ? case using $\langle y \sharp P a\rangle$
proof (induct rule: BRParCases)
case (BRPar P R)
have PRelQ: $(P, Q) \in$ Rel and $R B R Q:(R,!Q) \in$ bangRel Rel by fact +

```
have \(x \sharp P \| R\) and \(y \sharp P \| R\) by fact +
hence \(x\) Fresh \(P: x \sharp P\) and \(x\) Fresh \(R\) : \(x \sharp R\) and \(y \sharp R\) and \(y \sharp P\) by simp+ show ?case
proof (auto simp add: residual.inject)
from PRelQ have \(P \rightsquigarrow[\) Rel \(] Q\) by (rule Sim)
with \(Q\) Trans xFreshP obtain \(P^{\prime}\) where PTrans: \(P \longmapsto a<x>\prec P^{\prime}\) and \(P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}[x::=y], Q^{\prime}[x::=y]\right) \in \operatorname{Rel}\)
by (fastforce dest: simE simp add: derivative-def)
from \(R B R Q\langle y \sharp R\rangle I H\) have \(\exists R^{\prime} . R \longmapsto a<\nu y>\prec R^{\prime} \wedge\left(R^{\prime}, Q^{\prime \prime}\right) \in\) bangRel Rel by (auto simp add: residual.inject derivative-def)
then obtain \(R^{\prime}\) where RTrans: \(R \longmapsto a<\nu y>\prec R^{\prime}\) and \(R^{\prime} \operatorname{Rel} Q^{\prime \prime}:\left(R^{\prime}\right.\), \(\left.Q^{\prime \prime}\right) \in\) bangRel Rel by blast
from PTrans RTrans \(\left\langle y \sharp P>\right.\) have \(P \| R \longmapsto \tau \prec<\nu y>\left(P^{\prime}[x::=y] \| R^{\prime}\right)\) by(rule Close1)
moreover from \(P^{\prime} \operatorname{Rel} Q^{\prime} R^{\prime} \operatorname{Rel} Q^{\prime \prime}\) have \(\left(<\nu y>\left(P^{\prime}[x::=y] \| R^{\prime}\right)\right.\), \(\left.<\nu y>\left(Q^{\prime}[x::=y] \| Q^{\prime \prime}\right)\right) \in\) bangRel Rel by(force intro: Rel.BRPar BRRes)
ultimately show \(\exists P^{\prime} . P \| R \longmapsto \tau \prec P^{\prime} \wedge\left(P^{\prime},<\nu y>\left(Q^{\prime}[x::=y] \| Q^{\prime \prime}\right)\right)\) \(\in\) bangRel Rel by blast
qed
qed
next
case(cClose2 a x \(Q^{\prime}\) y \(Q^{\prime \prime}\) Pa P)
hence IH: \(\bigwedge P a .(P a,!Q) \in\) bangRel Rel \(\Longrightarrow\) ? Sim Pa \(\left(a<y>\prec Q^{\prime \prime}\right)\) by simp
have \(Q\) Trans: \(Q \longmapsto a<\nu x>\prec Q^{\prime}\) by fact
have \((P a, Q \|!Q) \in b a n g R e l\) Rel and \(x \sharp P a\) and \(y \sharp P a\) by fact+
thus ?case
proof (induct rule: BRParCases)
case (BRPar P R)
have PRelQ: \((P, Q) \in\) Rel and \(R B R Q:(R,!Q) \in\) bangRel Rel by fact +
have \(x \sharp P \| R\) and \(y \sharp P \| R\) by fact +
hence \(x\) Fresh \(P: x \sharp P\) and \(x\) Fresh \(R: x \sharp R\) and \(y \sharp R\) by simp +
show ? case
proof (auto simp add: residual.inject)
from \(\operatorname{PRelQ}\) have \(P \rightsquigarrow[\) Rel \(] Q\) by (rule Sim)
with QTrans xFreshP obtain \(P^{\prime}\) where PTrans: \(P \longmapsto a<\nu x>\prec P^{\prime}\) and \(P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in\) Rel
by (fastforce dest: simE simp add: derivative-def)
from \(R B R Q\) IH \(\langle y \sharp R\rangle\) have \(\exists R^{\prime} . \quad R \longmapsto a<y>\prec R^{\prime} \wedge\left(R^{\prime}[y::=x]\right.\), \(\left.Q^{\prime \prime}[y::=x]\right) \in\) bangRel Rel
by (fastforce simp add: derivative-def residual.inject)
then obtain \(R^{\prime}\) where RTrans: \(R \longmapsto a<y>\prec R^{\prime}\) and \(R^{\prime} R e l Q^{\prime \prime}:\left(R^{\prime}[y::=x]\right.\), \(\left.Q^{\prime \prime}[y::=x]\right) \in\) bangRel Rel
by blast
```

            from PTrans RTrans xFreshR have P | R\longmapsto\tau\prec<<\nux>(\mp@subsup{P}{}{\prime}|\mp@subsup{R}{}{\prime}[y::=x])
                    by(rule Close2)
            moreover from P'RelQ' R'RelQ'\prime have (<\nux> ( }\mp@subsup{P}{}{\prime}|\mp@subsup{R}{}{\prime}[y::=x]),<\nux>(\mp@subsup{Q}{}{\prime
    | Q''[y::=x])) \in bangRel Rel
by(force intro: Rel.BRPar BRRes)
ultimately show }\exists\mp@subsup{P}{}{\prime}.P|R\longmapsto~\prec\prec\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},<\nux>(\mp@subsup{Q}{}{\prime}|\mp@subsup{Q}{}{\prime\prime}[y::=x])
\epsilonbangRel Rel by blast
qed
qed
next
case(cBang Rs Pa P)
hence IH: \Pa. (Pa,Q|!Q)\inbangRel Rel \Longrightarrow ?Sim Pa Rs by simp
have (Pa,!Q) \in bangRel Rel by fact
thus ?case
proof(induct rule: BRBangCases)
case(BRBang P)
have PRelQ: (P,Q)\in Rel by fact
hence (!P,!Q)\in bangRel Rel by(rule Rel.BRBang)
with PRelQ have (P|!P,Q|!Q) \in bangRel Rel by(rule BRPar)
with IH have ?Sim ( }P|!P)\mathrm{ Rs by simp
thus ?case by(force intro: Bang)
qed
qed
}
moreover from PRelQ have (!P,!Q) \in bangRel Rel by(rule BRBang)
ultimately show ?thesis by(auto simp add: simulation-def)
qed
end
theory Strong-Late-Bisim-Pres
imports Strong-Late-Bisim Strong-Late-Sim-Pres
begin
lemma tauPres:
fixes P :: pi
and }Q:: p
assumes P~Q
shows }\tau.(P)~\tau.(Q
proof -
let ?X = {(\tau.(P),\tau.(Q)),(\tau.(Q),\tau.(P))}
have }(\tau.(P),\tau.(Q))\in\mathrm{ ? X by auto
thus ?thesis using <P ~ Q>
by(coinduct rule: bisimCoinduct)
(auto intro:Strong-Late-Sim-Pres.tauPres dest: symmetric)

```

\section*{qed}
```

lemma inputPres:
fixes $P$ :: $p i$
and $\quad Q:: p i$
and $a::$ name
and $x::$ name
assumes $P \operatorname{Sim} Q: \forall y . P[x::=y] \sim Q[x::=y]$
shows $a<x>. P \sim a<x>. Q$
proof -
let ? $X=\{(a<x>. P, a<x>. Q) \mid a x P Q . \forall y . P[x::=y] \sim Q[x::=y]\}$
\{
fix $a x P a x Q p$
assume $(a x P, a x Q) \in ? X$
then obtain $a x P Q$ where $A: \forall y . P[x::=y] \sim Q[x::=y]$ and $B: a x P=$
$a<x>. P$ and $C: a x Q=a<x>. Q$
by auto
have $\bigwedge y$. $((p::$ name prm $) \cdot P)[(p \cdot x)::=y] \sim(p \cdot Q)[(p \cdot x)::=y]$
proof -
fix $y$
from $A$ have $P[x::=($ rev $p \cdot y)] \sim Q[x::=($ rev $p \cdot y)]$
by blast
hence $(p \cdot(P[x::=($ rev $p \cdot y)])) \sim p \cdot(Q[x::=($ rev $p \cdot y)])$
by (rule bisimClosed)
thus $(p \cdot P)[(p \cdot x)::=y] \sim(p \cdot Q)[(p \cdot x)::=y]$
by (simp add: equts pt-pi-rev[OF pt-name-inst, OF at-name-inst])
qed
hence (( $p:$ :name prm) $\operatorname{axP}, p \cdot a x Q) \in$ ? $X$ using $B C$
by auto
\}
hence eqvt ? $X$ by (simp add: eqvt-def)
from $P \operatorname{Sim} Q$ have $(a<x>. P, a<x>. Q) \in ? X$ by auto
thus ?thesis
proof(coinduct rule: bisimCoinduct)
case $(c \operatorname{Sim} P Q)$
thus ?case using 〈eqvt ? $X$ >
by (force intro: inputPres)
next
case $(c \operatorname{Sym} P Q)$
thus ?case
by(blast dest: symmetric)
qed
qed
lemma outputPres:
fixes $P$ :: $p i$

```
```

    and }Q::p
    and a :: name
    and b:: name
    assumes P~Q
    shows }a{b}.P~a{b}.
    proof -
let ?X = {(a{b}.P,a{b}.Q), (a{b}.Q,a{b}.P)}
have (a{b}.P,a{b}.Q) \in?X by auto
thus ?thesis using <P ~ Q>
by(coinduct rule: bisimCoinduct)
(auto intro:Strong-Late-Sim-Pres.outputPres dest: symmetric)
qed
lemma matchPres:
fixes P :: pi
and }Q::p
and a :: name
and b :: name
assumes P~Q
shows [a\frownb]P~[a\frownb]Q
proof -
let ?X = {([a\frownb]P,[a\frownb]Q),([a\frownb]Q,[a\frownb]P)}
have }([a\frownb]P,[a\frownb]Q)\in?X by aut
thus ?thesis using <P~Q>
by(coinduct rule: bisimCoinduct)
(auto intro: Strong-Late-Sim-Pres.matchPres dest: symmetric bisimE)
qed
lemma mismatchPres:
fixes P :: pi
and }Q::p
and a :: name
and b :: name
assumes P~Q
shows [a\not=b]P~[a\not=b]Q
proof -
let ?X = {([a\not=b]P,[a\not=b]Q),([a\not=b]Q,[a\not=b]P)}
have }([a\not=b]P,[a\not=b]Q)\in?X by aut
thus ?thesis using <P ~ Q>
by(coinduct rule: bisimCoinduct)
(auto intro: Strong-Late-Sim-Pres.mismatchPres dest: symmetric bisimE)
qed

```
```

lemma sumPres:
fixes }P::p
and }Q::p
and }R::p
assumes P~Q
shows }P\oplusR~Q\oplus
proof -
let ?X = {(P\oplusR,Q\oplusR),(Q\oplusR,P\oplusR)}
have (P\oplusR,Q\oplusR)\in?X by auto
thus ?thesis using <P ~ Q>
by(coinduct rule: bisimCoinduct)
(auto intro: Strong-Late-Sim-Pres.sumPres reflexive dest: symmetric bisimE)
qed
lemma resPres:
fixes P :: pi
and }Q::p
and x :: name
assumes P~Q
shows <\nux>P~<\nux>>Q
proof -
let ?X = {x.\existsPQ.P~Q\wedge(\existsa.x=(<\nua>P,<\nua>Q))}

```

```

    thus ?thesis
    proof(coinduct rule: bisimCoinduct)
    case(cSim xP xQ)
    {
        fix PQa
            assume PSimQ: P\rightsquigarrow[bisim] Q
            moreover have }\PQa.P~Q\Longrightarrow(<\nua>P,<\nua>Q)\in?X\cup\mathrm{ bisim by
    blast
moreover have bisim \subseteq?X \cup bisim by blast
moreover have eqvt bisim by simp
moreover have eqvt ?X
by(auto simp add: eqvt-def) (blast intro: bisimClosed)
hence eqvt (?X \cup bisim) by auto
ultimately have <\nua>P\rightsquigarrow[(?X \cup bisim)]<\nua>Q
by(rule Strong-Late-Sim-Pres.resPres)
}
with }\langle(xP,xQ)\in?X\rangle\mathrm{ show ?case
by(auto dest: bisimE)
next
case(cSym xP xQ)
thus ?case by(auto dest: symmetric)
qed

```
```

qed
lemma parPres:
fixes }P\mathrm{ :: pi
and }Q::p
and }R::p
assumes P~Q
shows P|R~Q|R
proof -
let ?X = {(resChain lst (P|R), resChain lst (Q|R))| lst P R Q. P ~ Q}
have EmptyChain: \PQ.P|Q=resChain [] (P|Q) by auto
with }\langleP~Q\rangle\mathrm{ have (P|R,Q|R) E?X by blast
thus ?thesis
proof(coinduct rule: bisimCoinduct)
case(cSim PR QR)
{
fix PQ R lst
assume P~Q
hence P\rightsquigarrow[bisim] Q by(rule bisimE)
moreover note <P ~ Q>
moreover have }\wedgePQR.P~Q\Longrightarrow(P|R,Q|R)\in?
by auto (blast intro: EmptyChain)
moreover
{
fix }xPxQ
assume (xP,xQ)\in?X
then obtain PQ R lst
where P~Q and xP= resChain lst (P|R) and xQeq: xQ = resChain
lst (Q|R)
by auto
moreover hence (resChain (x\#lst) (P|R), resChain (x\#lst) (Q|R))\in
?X
by blast
ultimately have ( <\nux>xP, <\nux>>xQ) \in?X by auto
}
note ResPres = this
moreover have eqvt bisim by simp
moreover have eqvt ?X
by(auto simp add: eqvt-def) (blast intro: bisimClosed)
ultimately have P|R\rightsquigarrow[(?X)] Q| R by(rule parPres)
hence resChain lst (P|R)\rightsquigarrow[?X] (resChain lst (Q|R)) using <eqvt ?X>
ResPres
by(rule resChainI)
hence resChain lst (P|R)\rightsquigarrow[(?X\cup bisim)] (resChain lst (Q|R))
by(force intro:Strong-Late-Sim.monotonic)

```
```

    }
    with}\langle(PR,QR)\in?X> show ?cas
        by auto
    next
        case(cSym PR QR)
        thus ?case by(blast dest: symmetric)
    qed
    qed
lemma bangPres:
fixes P :: pi
and }Q::p
assumes PBiSimQ: P~Q
shows !P~!Q
proof -
let ?X = bangRel bisim
from PBiSimQ have (!P,!Q)\in?X by(rule Rel.BRBang)
thus ?thesis
proof(coinduct rule: bisimCoinduct)
case(cSim bP bQ)
{
fix PQ
assume (P,Q)\in?X
hence P}\rightsquigarrow[?X]
proof(induct)
fix PQ
assume P~Q
thus !P\rightsquigarrow[?X]!Q using bisimE bisimEqvt
by(rule Strong-Late-Sim-Pres.bangPres)
next
fix PQRT
assume RBiSimT: R~T
assume PBangRelQ:}(P,Q)\in?
assume PSimQ: P}\rightsquigarrow[?X]
from RBiSimT have R\rightsquigarrow[bisim] T by(blast dest: bisimE)
thus R|P\rightsquigarrow[?X] T| Q using PSimQ RBiSimT PBangRelQ Rel.BRPar
Rel.BRRes bisimEqvt eqvtBangRel
by(blast intro: Strong-Late-Sim-Pres.parCompose)
next
fix PQa
assume P}\rightsquigarrow[?X]
moreover from eqvtBangRel bisimEqvt have eqvt ?X by blast
ultimately show <\nua>P\rightsquigarrow[?X]<\nua>Q using Rel.BRRes by(blast intro:
Strong-Late-Sim-Pres.resPres)
qed
hence P}\rightsquigarrow[((bangRel bisim)\cup\mathrm{ bisim )] Q by(rule-tac Strong-Late-Sim.monotonic)

```
```

auto
}
with «(bP,bQ)\in?X> show ?case by auto
next
case(cSym bP bQ)
thus ?case by(metis bangRelSymetric symmetric)
qed
qed
end
theory Strong-Late-Bisim-Subst-Pres
imports Strong-Late-Bisim-Subst Strong-Late-Bisim-Pres
begin
lemma tauPres:
fixes P :: pi
and }Q::p
assumes P ~}\mp@subsup{~}{}{s}
shows }\tau.(P)\mp@subsup{~}{}{s}\tau.(Q
using assms
by(force simp add: substClosed-def intro: Strong-Late-Bisim-Pres.tauPres)
lemma inputPres:
fixes P :: pi
and }Q:: p
and a :: name
and x :: name
assumes P ~}\mp@subsup{~}{}{s}
shows }a<x>.P\mp@subsup{~}{}{s}a<x>.
proof(auto simp add: substClosed-def)
fix }\sigma::(\mathrm{ name }\times\mathrm{ name) list
{
fix PQ a x \sigma
assume P ~
then have P[<\sigma>] ~s}Q[<\sigma>] by(rule partUnfold
then have \forally.(P[<\sigma>])[x::=y]~(Q[<\sigma>])[x::=y]
apply(auto simp add: substClosed-def)
by(erule-tac x=[(x,y)] in allE) auto
moreover assume x\sharp\sigma
ultimately have (a<x>.P)[<\sigma>] ~ (a<x>.Q)[<\sigma>] using bisimEqvt
by(force intro:Strong-Late-Bisim-Pres.inputPres)
}
note Goal = this

```
```

    obtain \(y:: n a m e\) where \(y \sharp P\) and \(y \sharp Q\) and \(y \sharp \sigma\)
    by (generate-fresh name) auto
    from \(\left\langle P \sim^{s} Q\right\rangle\) have \(([(x, y)] \cdot P) \sim^{s}([(x, y)] \cdot Q)\) by (rule eqClosed \()\)
    hence \((a<y>.([(x, y)] \cdot P))[<\sigma>] \sim(a<y>.([(x, y)] \cdot Q))[<\sigma>]\) using \(\langle y \sharp \sigma\rangle\)
    by (rule Goal)
moreover from $\langle y \sharp P\rangle\langle y \sharp Q\rangle$ have $a<x>. P=a<y>$. $([(x, y)] \cdot P)$ and
$a<x>. Q=a<y>.([(x, y)] \cdot Q)$
by (simp add: alphaInput) +
ultimately show $(a<x\rangle . P)[<\sigma\rangle] \sim(a<x\rangle . Q)[\langle\sigma\rangle]$ by simp
qed
lemma outputPres:
fixes $P$ :: $p i$
and $\quad Q:: p i$
assumes $P \sim^{s} Q$
shows $a\{b\} . P \sim^{s} a\{b\} . Q$
using assms
by(force simp add: substClosed-def intro: Strong-Late-Bisim-Pres.outputPres)
lemma matchPres:
fixes $P$ :: $p i$
and $\quad Q:: p i$
and $a::$ name
and $b::$ name
assumes $P \sim^{s} Q$
shows $[a \frown b] P \sim^{s}[a \frown b] Q$
using assms
by(force simp add: substClosed-def intro: Strong-Late-Bisim-Pres.matchPres)
lemma mismatchPres:
fixes $P$ :: $p i$
and $\quad Q:: p i$
and $a::$ name
and $b::$ name
assumes $P \sim^{s} Q$
shows $[a \neq b] P \sim^{s}[a \neq b] Q$
using assms
by (force simp add: substClosed-def intro: Strong-Late-Bisim-Pres.mismatchPres)
lemma sumPres:
fixes $P$ :: $p i$
and $\quad Q:: p i$

```
and \(\quad R:: p i\)
assumes \(P \sim^{s} Q\)
shows \(P \oplus R \sim^{s} Q \oplus R\)
using assms
by(force simp add: substClosed-def intro: Strong-Late-Bisim-Pres.sumPres)
lemma parPres:
fixes \(P:: p i\)
and \(\quad Q:: p i\)
and \(\quad R:: p i\)
assumes \(P \sim^{s} Q\)
shows \(P\left\|R \sim^{s} Q\right\| R\)
using assms
by (force simp add: substClosed-def intro: Strong-Late-Bisim-Pres.parPres)
lemma resPres:
fixes \(P:: p i\)
and \(\quad Q:: p i\)
and \(\quad x::\) name
assumes \(P e q Q: P \sim^{s} Q\)
shows \(<\nu x>P \sim^{s}<\nu x>Q\)
proof (auto simp add: substClosed-def)
fix \(s::(\) name \(\times\) name \()\) list
have Res: \(\bigwedge P Q x s . \llbracket P[\langle s\rangle] \sim Q[<s\rangle] ; x \sharp s \rrbracket \Longrightarrow(\langle\nu x>P)[<s\rangle] \sim(\langle\nu x\rangle Q)[<s\rangle]\)
by (force intro: Strong-Late-Bisim-Pres.resPres)
have \(\exists c::\) name. \(c \sharp(P, Q, s)\) by (blast intro: name-exists-fresh)
then obtain \(c::\) name where cFreshP: \(c \sharp P\) and cFresh \(Q: c \sharp Q\) and cFreshs:
\(c \sharp s\)
by(force simp add: fresh-prod)
from PeqQ have \(P[<([(x, c)] \cdot s)>] \sim Q[<([(x, c)] \cdot s)>]\) by \((\) simp add: subst-Closed-def)
hence \(([(x, c)] \cdot P[<([(x, c)] \cdot s)>]) \sim([(x, c)] \cdot Q[<([(x, c)] \cdot s)>])\) by \((r u l e\) bisimClosed)
hence \(([(x, c)] \cdot P)[<s\rangle] \sim([(x, c)] \cdot Q)[<s\rangle]\) by simp
hence \((\langle\nu c\rangle([(x, c)] \cdot P))[\langle s\rangle] \sim(\langle\nu c\rangle([(x, c)] \cdot Q))[\langle s\rangle]\) using cFreshs
by (rule Res)
moreover from cFreshP cFresh \(Q\) have \(\langle\nu x\rangle P=\langle\nu c\rangle([(x, c)] \cdot P)\) and \(<\nu x>Q=<\nu c>([(x, c)] \cdot Q)\)
by (simp add: alphaRes)+
```

    ultimately show \((\langle\nu x\rangle P)[\langle s\rangle] \sim(\langle\nu x\rangle Q)[\langle s\rangle]\) by simp
    qed
lemma bangPres:
fixes $P:: p i$
and $\quad Q:: p i$
assumes $P \sim^{s} Q$
shows $!P \sim^{s}!Q$
using assms
by(force simp add: substClosed-def intro: Strong-Late-Bisim-Pres.bangPres)
end
theory Late-Tau-Chain
imports Late-Semantics1
begin
abbreviation tauChain-judge :: pi $\Rightarrow$ pi $\Rightarrow$ bool $\left(-\Longrightarrow_{\tau}-[80,80] 80\right)$
where $P \Longrightarrow_{\tau} P^{\prime} \equiv\left(P, P^{\prime}\right) \in\left\{\left(P, P^{\prime}\right) \mid P P^{\prime} . P \longmapsto \tau \prec P^{\prime}\right\}{ }^{*} *$
lemma singleTauChain:
fixes $P$ :: $p i$
and $\quad P^{\prime}:: p i$
assumes $P \longmapsto \tau \prec P^{\prime}$
shows $P \Longrightarrow_{\tau} P^{\prime}$
using assms by (simp add: r-into-rtrancl)
lemma tauChainAddTau[dest]:
fixes $P$ :: $p i$
and $P^{\prime}:: p i$
and $P^{\prime \prime}:: p i$
shows $P \Longrightarrow_{\tau} P^{\prime} \Longrightarrow P^{\prime} \longmapsto \tau \prec P^{\prime \prime} \Longrightarrow P \Longrightarrow_{\tau} P^{\prime \prime}$
and $P \longmapsto \tau \prec P^{\prime} \Longrightarrow P^{\prime} \Longrightarrow_{\tau} P^{\prime \prime} \Longrightarrow P \Longrightarrow{ }_{\tau} P^{\prime \prime}$
by(auto dest: singleTauChain)
lemma tauChainInduct[consumes 1, case-names id ih]:
fixes $P$ :: $p i$
and $\quad P^{\prime}:: p i$
assumes $P \Longrightarrow_{\tau} P^{\prime}$
and $\quad F P$
and $\quad \wedge P^{\prime} P^{\prime \prime} . \llbracket P \Longrightarrow_{\tau} P^{\prime} ; P^{\prime} \longmapsto \tau \prec P^{\prime \prime} ; F P^{\prime} \rrbracket \Longrightarrow F P^{\prime \prime}$

```
```

    shows F P '
    using assms
by(drule-tac rtrancl-induct) auto
lemma eqvtChainI[eqvt]:
fixes P :: pi
and }\mp@subsup{P}{}{\prime}::p
and perm :: name prm
assumes }P>\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime
shows (perm • P) \Longrightarrow}\mp@subsup{}{\tau}{}(\mathrm{ perm }\cdot\mp@subsup{P}{}{\prime}
using assms
proof(induct rule: tauChainInduct)
case id
thus ?case by simp
next
case(ih P'\prime P '/\prime)
have }P\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime\prime}\mathrm{ and }\mp@subsup{P}{}{\prime\prime}\longmapsto\tau\prec\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ by fact+
hence (perm • P')}\longmapsto\longmapsto\tau\prec(\mathrm{ perm • P }\mp@subsup{P}{}{\prime\prime\prime})\mathrm{ by(force dest: transitions.eqvt)
moreover have (perm • P) \Longrightarrow>
ultimately show ?case by auto
qed
lemma eqvtChainE:
fixes perm :: name prm
and }P\mathrm{ :: pi
and }\mp@subsup{P}{}{\prime}::p
assumes Trans: (perm • P) \Longrightarrow\Longrightarrow}\mp@subsup{\tau}{}{(
shows }P\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime
proof -
have rev perm • (perm •P) =P by(simp add: pt-rev-pi[OF pt-name-inst, OF
at-name-inst])
moreover have rev perm}\cdot(\mathrm{ perm }\cdot\mp@subsup{P}{}{\prime})=\mp@subsup{P}{}{\prime}\mathbf{by}(\mathrm{ simp add: pt-rev-pi[OF pt-name-inst,
OF at-name-inst])
ultimately show ?thesis using assms
by(drule-tac perm=rev perm in eqvtChainI, simp)
qed
lemma eqvtChainEq:
fixes P :: pi
and }\mp@subsup{P}{}{\prime}::p
and perm :: name prm
shows }P=\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime}=(\mathrm{ perm • P) >}\mp@subsup{}{\tau}{}(\mathrm{ perm • P')
by(blast intro: eqvtChainE eqvtChainI)

```
```

lemma freshChain:
fixes $P$ :: $p i$
and $P^{\prime}:: p i$
and $x$ :: name
assumes $P \Longrightarrow_{\tau} P^{\prime}$
and $\quad x \sharp P$
shows $\quad x \sharp P^{\prime}$
using assms
proof(induct rule: tauChainInduct)
case id
thus ?case by simp
next
case(ih $P^{\prime} P^{\prime \prime}$ )
have $x \sharp P$ and $x \sharp P \Longrightarrow x \sharp P^{\prime}$ by fact +
hence $x \sharp P^{\prime}$ by simp
moreover have $P^{\prime} \longmapsto \tau \prec P^{\prime \prime}$ by fact
ultimately show ?case by (force intro: freshFreeDerivative)
qed
lemma matchChain:
fixes $b$ :: name
and $P:: p i$
and $\quad P^{\prime}:: p i$
assumes $P \Longrightarrow_{\tau} P^{\prime}$
and $\quad P \neq P^{\prime}$
shows $[b \frown b] P \Longrightarrow{ }_{\tau} P^{\prime}$
using assms
proof(induct rule: tauChainInduct)
case id
thus ?case by simp
next
case (ih $\left.P^{\prime \prime} P^{\prime \prime \prime}\right)$
have $P^{\prime \prime}$ Trans $P^{\prime \prime \prime}: P^{\prime \prime} \longmapsto \tau \prec P^{\prime \prime \prime}$ by fact
show $[b \frown b] P \Longrightarrow_{\tau} P^{\prime \prime \prime}$
proof (cases $P=P^{\prime \prime}$ )
assume $P=P^{\prime \prime}$
moreover with $P^{\prime \prime}$ Trans $P^{\prime \prime \prime}$ have $[b \frown b] P \longmapsto \tau \prec P^{\prime \prime \prime}$ by (force intro: Match)
thus $[b \frown b] P \Longrightarrow_{\tau} P^{\prime \prime \prime}$ by (rule singleTauChain)
next
assume $P \neq P^{\prime \prime}$
moreover have $P \neq P^{\prime \prime} \Longrightarrow[b \frown b] P \Longrightarrow_{\tau} P^{\prime \prime}$ by fact
ultimately show $[b \frown b] P \Longrightarrow{ }_{\tau} P^{\prime \prime \prime}$ using $P^{\prime \prime}$ Trans $P^{\prime \prime \prime}$ by (blast)
qed
qed

```
```

lemma mismatchChain:
fixes $a$ :: name
and $b::$ name
and $\quad P:: p i$
and $\quad P^{\prime}:: p i$
assumes PChain: $P \Longrightarrow{ }_{\tau} P^{\prime}$
and aineqb: $a \neq b$
and PineqP': $P \neq P^{\prime}$
shows $[a \neq b] P \Longrightarrow_{\tau} P^{\prime}$
using PChain PineqP
proof $($ induct rule: tauChainInduct)
case id
thus ?case by simp
next
case (ih $P^{\prime \prime} P^{\prime \prime \prime}$ )
have $P^{\prime \prime}$ Trans $P^{\prime \prime \prime}: P^{\prime \prime} \longmapsto \tau \prec P^{\prime \prime \prime}$ by fact
show $[a \neq b] P \Longrightarrow_{\tau} P^{\prime \prime \prime}$
proof(cases $P=P^{\prime \prime}$ )
assume $P=P^{\prime \prime}$
moreover with aineqb $P^{\prime \prime}$ Trans $P^{\prime \prime \prime}$ have $[a \neq b] P \longmapsto \tau \prec P^{\prime \prime \prime}$ by (force intro:
Mismatch)
thus $[a \neq b] P \Longrightarrow_{\tau} P^{\prime \prime \prime}$ by(rule singleTauChain)
next
assume $P \neq P^{\prime \prime}$
moreover have $P \neq P^{\prime \prime} \Longrightarrow[a \neq b] P \Longrightarrow_{\tau} P^{\prime \prime}$ by fact +
ultimately show $[a \neq b] P \Longrightarrow_{\tau} P^{\prime \prime \prime}$ using $P^{\prime \prime}$ Trans $P^{\prime \prime \prime}$ by (blast)
qed
qed
lemma sum1Chain[rule-format]:
fixes $P$ :: $p i$
and $\quad P^{\prime}:: p i$
and $Q$ :: pi
assumes $P \Longrightarrow_{\tau} P^{\prime}$
and $\quad P \neq P^{\prime}$
shows $P \oplus Q \Longrightarrow_{\tau} P^{\prime}$
using assms
proof(induct rule: tauChainInduct)
case $i d$
thus?case by simp
next
case $\left(\right.$ ih $\left.P^{\prime \prime} P^{\prime \prime \prime}\right)$
have $P^{\prime \prime}$ TransP $P^{\prime \prime \prime}: P^{\prime \prime} \longmapsto \tau \prec P^{\prime \prime \prime}$ by fact
show $P \oplus Q \Longrightarrow_{\tau} P^{\prime \prime \prime}$

```
```

    proof \(\left(\right.\) cases \(\left.P=P^{\prime \prime}\right)\)
    assume \(P=P^{\prime \prime}\)
    moreover with \(P^{\prime \prime}\) Trans \(P^{\prime \prime \prime}\) have \(P \oplus Q \longmapsto \tau \prec P^{\prime \prime \prime}\) by(force intro: Sum1)
    thus \(P \oplus Q \Longrightarrow_{\tau} P^{\prime \prime \prime}\) by(force intro: singleTauChain)
    next
    assume \(P \neq P^{\prime \prime}\)
    moreover have \(P \neq P^{\prime \prime} \Longrightarrow P \oplus Q \Longrightarrow_{\tau} P^{\prime \prime}\) by fact
    ultimately show \(P \oplus Q \Longrightarrow{ }_{\tau} P^{\prime \prime \prime}\) using \(P^{\prime \prime}\) Trans \(P^{\prime \prime \prime}\) by (force)
    qed
    qed
lemma sum2Chain[rule-format]:
fixes $P:: p i$
and $\quad Q:: p i$
and $\quad Q^{\prime}:: p i$
assumes $Q \Longrightarrow_{\tau} Q^{\prime}$
and $\quad Q \neq Q^{\prime}$
shows $P \oplus Q \Longrightarrow_{\tau} Q^{\prime}$
using assms
proof(induct rule: tauChainInduct)
case id
thus ?case by simp
next
case $\left(i h Q^{\prime \prime} Q^{\prime \prime \prime}\right)$
have $Q^{\prime \prime}$ Trans $Q^{\prime \prime \prime}: \quad Q^{\prime \prime} \longmapsto \tau \prec Q^{\prime \prime \prime}$ by fact
show $P \oplus Q \Longrightarrow_{\tau} Q^{\prime \prime \prime}$
proof (cases $Q=Q^{\prime \prime}$ )
assume $Q=Q^{\prime \prime}$
moreover with $Q^{\prime \prime}$ Trans $Q^{\prime \prime \prime}$ have $P \oplus Q \longmapsto \tau \prec Q^{\prime \prime \prime}$ by (force intro: Sum2)
thus $P \oplus Q \Longrightarrow_{\tau} Q^{\prime \prime \prime}$ by(force intro: singleTauChain)
next
assume $Q \neq Q^{\prime \prime}$
moreover have $Q \neq Q^{\prime \prime} \Longrightarrow P \oplus Q \Longrightarrow_{\tau} Q^{\prime \prime}$ by fact
ultimately show $P \oplus Q \Longrightarrow_{\tau} Q^{\prime \prime \prime}$ using $Q^{\prime \prime}$ Trans $Q^{\prime \prime \prime}$ by blast
qed
qed
lemma Par1Chain:
fixes $P$ :: $p i$
and $P^{\prime}:: p i$
and $\quad Q:: p i$
assumes $P \Longrightarrow_{\tau} P^{\prime}$
shows $P\left\|Q \Longrightarrow_{\tau} P^{\prime}\right\| Q$
using assms

```
```

proof(induct rule: tauChainInduct)
case id
thus ?case by simp
next
case (ih $\left.P^{\prime \prime} P^{\prime}\right)$
have $P^{\prime \prime}$ Trans $P^{\prime}: P^{\prime \prime} \longmapsto \tau \prec P^{\prime}$ by fact
have $I H: P\left\|Q \Longrightarrow{ }_{\tau} P^{\prime \prime}\right\| Q$ by fact
have $P^{\prime \prime}\left\|Q \longmapsto \tau \prec P^{\prime}\right\| Q$ using $P^{\prime \prime} T$ Trans $P^{\prime}$ by(force intro: Par1F)
thus $P\left\|Q \Longrightarrow{ }_{\tau} P^{\prime}\right\| Q$ using $I H$ by(force)
qed
lemma Par2Chain:
fixes $P$ :: $p i$
and $Q$ :: pi
and $\quad Q^{\prime}:: p i$
assumes $Q \Longrightarrow_{\tau} Q^{\prime}$
shows $P\left\|Q \Longrightarrow_{\tau} P\right\| Q^{\prime}$
using assms
proof(induct rule: tauChainInduct)
case id
thus?case by simp
next
case (ih $Q^{\prime \prime} Q^{\prime}$ )
have $Q^{\prime \prime}$ Trans $Q^{\prime}: \quad Q^{\prime \prime} \longmapsto \tau \prec Q^{\prime}$ by fact
have $I H: P\left\|Q \Longrightarrow_{\tau} P\right\| Q^{\prime \prime}$ by fact
have $P\left\|Q^{\prime \prime} \longmapsto \tau \prec P\right\| Q^{\prime}$ using $Q^{\prime \prime}$ Trans $Q^{\prime}$ by (force intro: Par2F)
thus $P\left\|Q \Longrightarrow_{\tau} P\right\| Q^{\prime}$ using $I H$ by(force)
qed
lemma chainPar:
fixes $P:: p i$
and $P^{\prime}:: p i$
and $\quad Q:: p i$
and $\quad Q^{\prime}:: p i$
assumes $P \Longrightarrow_{\tau} P^{\prime}$
and $\quad Q \Longrightarrow_{\tau} Q^{\prime}$
shows $P\left\|Q \Longrightarrow_{\tau} P^{\prime}\right\| Q^{\prime}$
proof -
from $\left\langle P \Longrightarrow_{\tau} P^{\prime}\right\rangle$ have $P\left\|Q \Longrightarrow_{\tau} P^{\prime}\right\| Q$ by(rule Par1Chain)
moreover from $\left\langle Q \Longrightarrow_{\tau} Q^{\prime}\right\rangle$ have $P^{\prime}\left\|Q \Longrightarrow_{\tau} P^{\prime}\right\| Q^{\prime}$ by(rule Par2Chain)
ultimately show ?thesis by auto
qed

```
```

lemma ResChain:
fixes }P\mathrm{ :: pi
and }\mp@subsup{P}{}{\prime}::p
and a :: name
assumes }P\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime
shows }<\nua>P\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}<\nua>\mp@subsup{P}{}{\prime
using assms
proof(induct rule: tauChainInduct)
case id
thus ?case by simp
next
case(ih P'\prime }\mp@subsup{P}{}{\prime\prime\prime}
have }\mp@subsup{P}{}{\prime\prime}\longmapsto\tau\prec\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ by fact
hence <\nua>P'\prime\prime\longmapsto\tau\prec<\nua>P\mp@subsup{P}{}{\prime\prime\prime}}\mathbf{by}(force intro: ResF
moreover have <\nua>P \Longrightarrow}\mp@subsup{|}{\tau}{}<\nua>\mp@subsup{P}{}{\prime\prime}\mathrm{ by fact
ultimately show ?case by force
qed
lemma substChain:
fixes P :: pi
and }x\mathrm{ :: name
and b :: name
and }\mp@subsup{P}{}{\prime}::p
assumes PTrans: P[x::=b] \Longrightarrow>}\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime
shows }P[x::=b]\Longrightarrow\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime}[x::=b
proof(cases x=b)
assume }x=
with PTrans show ?thesis by simp
next
assume }x\not=
hence }x\sharpP[x::=b] by(simp add: fresh-fact2
with PTrans have }x\sharp\mp@subsup{P}{}{\prime}\mathrm{ by(force intro: freshChain)
hence }\mp@subsup{P}{}{\prime}=\mp@subsup{P}{}{\prime}[x::=b] by(simp add: forget
with PTrans show ?thesis by simp
qed
lemma bangChain:
fixes P :: pi
and }\mp@subsup{P}{}{\prime}:: p
assumes PTrans: P|!P\Longrightarrow\Longrightarrow}\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime
and }\quad\mp@subsup{P}{}{\prime}\mathrm{ ineq: }\mp@subsup{P}{}{\prime}\not=P|!
shows !P\Longrightarrow}\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime
using assms

```
```

proof(induct rule: tauChainInduct)
case $i d$
thus ?case by simp
next
case(ih $\left.P^{\prime} P^{\prime \prime}\right)$
show ?case
proof $\left(\right.$ cases $\left.P^{\prime}=P \|!P\right)$
case True
from $\left\langle P^{\prime} \longmapsto \tau \prec P^{\prime \prime}\right\rangle\left\langle P^{\prime}=P \|!P\right\rangle$ have $!P \longmapsto \tau \prec P^{\prime \prime}$ by (blast intro: Bang)
thus ?thesis by auto
next
case False
from $\left\langle P^{\prime} \neq P \|!P\right\rangle$ have $!P \Longrightarrow_{\tau} P^{\prime}$ by (rule ih)
with $\left\langle P^{\prime} \longmapsto \tau \prec P^{\prime \prime}\right\rangle$ show ?thesis by auto
qed
qed
end
theory Weak-Late-Step-Semantics
imports Late-Tau-Chain
begin
definition inputTransition :: pi $\Rightarrow$ name $\Rightarrow p i \Rightarrow$ name $\Rightarrow$ name $\Rightarrow$ pi $\Rightarrow$ bool (-
$\Longrightarrow_{l^{-}}$in $\left.\rightarrow-<->\prec-[80,80,80,80,80] 80\right)$
where $P \Longrightarrow_{l}$ u in $P^{\prime \prime} \rightarrow a<x>\prec P^{\prime} \equiv \exists P^{\prime \prime \prime} . P \Longrightarrow_{\tau} P^{\prime \prime \prime} \wedge P^{\prime \prime \prime} \longmapsto a<x>\prec P^{\prime \prime}$
$\wedge P^{\prime \prime}[x::=u] \Longrightarrow_{\tau} P^{\prime}$
definition transition :: (pi $\times$ Late-Semantics.residual) set where
transition $\equiv\left\{x . \exists P P^{\prime} \alpha P^{\prime \prime} P^{\prime \prime \prime} . P \Longrightarrow_{\tau} P^{\prime} \wedge P^{\prime} \longmapsto \alpha \prec P^{\prime \prime} \wedge P^{\prime \prime} \Longrightarrow_{\tau} P^{\prime \prime \prime}\right.$
$\left.\wedge x=\left(P, \alpha \prec P^{\prime \prime \prime}\right)\right\} \cup$
$\left\{x . \exists P P^{\prime}\right.$ a y $P^{\prime \prime} P^{\prime \prime \prime} . P \Longrightarrow_{\tau} P^{\prime} \wedge\left(P^{\prime} \longmapsto\left(a<\nu y>\prec P^{\prime \prime}\right)\right) \wedge P^{\prime \prime}$
$\left.\Longrightarrow_{\tau} P^{\prime \prime \prime} \wedge x=\left(P,\left(a<\nu y>\prec P^{\prime \prime \prime}\right)\right)\right\}$

```
abbreviation weakTransition-judge :: pi \(\Rightarrow\) Late-Semantics.residual \(\Rightarrow\) bool (-\(\left.\Longrightarrow_{l}-[80,80] 80\right)\)
where \(P \Longrightarrow_{l}\) Rs \(\equiv(P, R s) \in\) transition
lemma weakNonInput[dest]:
fixes \(P\) :: \(p i\)
and \(a\) :: name
and \(x\) :: name
and \(\quad P^{\prime}:: p i\)
assumes \(P \Longrightarrow_{l} a<x>\prec P^{\prime}\)
shows False
using assms
by(auto simp add: transition-def residual.inject)
```

lemma transitionI:
fixes P :: pi
and }\mp@subsup{P}{}{\prime\prime\prime}:: p
and \alpha :: freeRes
and }\mp@subsup{P}{}{\prime\prime}:: p
and }\mp@subsup{P}{}{\prime}::p
and a :: name
and x :: name
and u :: name
shows \llbracketP\Longrightarrow\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime\prime\prime};\mp@subsup{P}{}{\prime\prime\prime}\longmapsto\alpha\prec\mp@subsup{P}{}{\prime\prime};\mp@subsup{P}{}{\prime\prime}\mp@subsup{\Longrightarrow}{\tau}{}P<br>LongrightarrowP\Longrightarrow\mp@subsup{\Longrightarrow}{l}{}\alpha\prec\mp@subsup{P}{}{\prime}

```

```

P'

```

```

P'\prime}->a<x>\prec\mp@subsup{P}{}{\prime
proof -
assume P \Longrightarrow}\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ and }\mp@subsup{P}{}{\prime\prime\prime}\longmapsto\alpha\prec\mp@subsup{P}{}{\prime\prime}\mathrm{ and }\mp@subsup{P}{}{\prime\prime}\Longrightarrow\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime
thus }P\mp@subsup{\Longrightarrow}{l}{}\alpha\prec\mp@subsup{P}{}{\prime
by(force simp add: transition-def)
next

```

```

    thus }P\mp@subsup{\Longrightarrow}{l}{}a<\nux><\mp@subsup{P}{}{\prime
    by(force simp add: transition-def)
    next
assume P \Longrightarrow}\mp@subsup{~}{\tau}{}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ and }\mp@subsup{P}{}{\prime\prime\prime}\longmapstoa<x>< P '\prime and P'\prime[x::=u] \Longrightarrow\Longrightarrow\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime
thus }P\mp@subsup{\Longrightarrow}{l}{}\mathrm{ u in P}\mp@subsup{P}{}{\prime\prime}->a<x>\prec\mp@subsup{P}{}{\prime
by(force simp add: inputTransition-def)
qed
lemma transitionE:
fixes P :: pi
and \alpha :: freeRes
and }\mp@subsup{P}{}{\prime}::p
and }\mp@subsup{P}{}{\prime\prime}::p
and a :: name
and u :: name
and x :: name
shows }P\mp@subsup{\Longrightarrow}{l}{}\alpha\prec\mp@subsup{P}{}{\prime}\Longrightarrow\exists\mp@subsup{P}{}{\prime\prime}\mp@subsup{P}{}{\prime\prime\prime}.P\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime\prime}\wedge\mp@subsup{P}{}{\prime\prime}\longmapsto\alpha\prec\mp@subsup{P}{}{\prime\prime\prime}\wedge\mp@subsup{P}{}{\prime\prime\prime}\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{
P'(is - \Longrightarrow ?thesis1)
and }\llbracketP\Longrightarrow\mp@subsup{\Longrightarrow}{l}{}a<\nux>\prec\mp@subsup{P}{}{\prime};x\sharpP\rrbracket\Longrightarrow\exists\mp@subsup{P}{}{\prime\prime}\mp@subsup{P}{}{\prime\prime\prime}.P\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime\prime\prime}\wedge\mp@subsup{P}{}{\prime\prime\prime}\longmapstoa<\nux

```


```

P'\prime}\wedge\mp@subsup{P}{}{\prime\prime}[x::=u]\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime
proof -
assume P\Longrightarrow}\mp@subsup{l}{l}{}\alpha\prec\mp@subsup{P}{}{\prime
thus ?thesis1 by(auto simp add: transition-def residual.inject)
next

```
```

    assume \(P \Longrightarrow_{l} a<\nu x>\prec P^{\prime}\) and \(x \sharp P\)
    thus \(\exists P^{\prime \prime} P^{\prime \prime \prime} . P \Longrightarrow_{\tau} P^{\prime \prime \prime} \wedge P^{\prime \prime \prime} \longmapsto a<\nu x>\prec P^{\prime \prime} \wedge P^{\prime \prime} \Longrightarrow_{\tau} P^{\prime}\)
    using [[hypsubst-thin \(=\) true \(]\)
    apply (auto simp add: transition-def residualInject name-abs-eq)
    apply \(\left(\right.\) rule-tac \(x=[(x, y)] \cdot P^{\prime \prime}\) in \(\left.e x I\right)\)
    apply (rule-tac \(x=P^{\prime}\) in \(\left.e x I\right)\)
    apply (clarsimp)
    apply (auto)
    apply (subgoal-tac \(x \sharp P^{\prime \prime}\) )
    apply(simp add: alphaBoundResidual name-swap)
    using freshChain
    apply(force dest: freshBoundDerivative)
    using eqvtChainI
    by \(\operatorname{simp}\)
    next
assume PTrans: $P \Longrightarrow_{l}$ u in $P^{\prime \prime} \rightarrow a<x>\prec P^{\prime}$
thus $\exists P^{\prime \prime \prime} . P \Longrightarrow_{\tau} P^{\prime \prime \prime} \wedge P^{\prime \prime \prime} \longmapsto a<x>\prec P^{\prime \prime} \wedge P^{\prime \prime}[x::=u] \Longrightarrow_{\tau} P^{\prime}$
by (auto simp add: inputTransition-def)
qed
lemma alphaInput:
fixes $P$ :: $p i$
and $u$ :: name
and $P^{\prime \prime}:: p i$
and $a$ :: name
and $x$ :: name
and $\quad P^{\prime}:: p i$
and $y$ :: name
assumes PTrans: $P \Longrightarrow_{l} u$ in $P^{\prime \prime} \rightarrow a<x>\prec P^{\prime}$
and $y$ FreshP: $y \sharp P$
shows $P \Longrightarrow_{l} u$ in $\left([(x, y)] \cdot P^{\prime \prime}\right) \rightarrow a<y>\prec P^{\prime}$
proof (cases $x=y$ )
assume $x=y$
with PTrans show ?thesis by simp
next
assume xineqy: $x \neq y$
from PTrans obtain $P^{\prime \prime \prime}$ where PChain: $P \Longrightarrow{ }_{\tau} P^{\prime \prime \prime}$
and $P^{\prime \prime \prime}$ Trans: $P^{\prime \prime \prime} \longmapsto a<x>\prec P^{\prime \prime}$
and $P^{\prime \prime}$ Chain: $P^{\prime \prime}[x::=u] \Longrightarrow_{\tau} P^{\prime}$
by (blast dest: transitionE)
from PChain yFreshP have $y \sharp P^{\prime \prime \prime}$ by (rule freshChain)
with $P^{\prime \prime \prime}$ Trans xineqy have yFresh $P^{\prime \prime}: y \sharp P^{\prime \prime}$ by (blast dest: freshBoundDerivative)
with $P^{\prime \prime \prime}$ Trans have $P^{\prime \prime \prime} \longmapsto a<y>\prec[(x, y)] \cdot P^{\prime \prime}$ by (simp add: alphaBoundResidual)

```
```

    moreover from P}\mp@subsup{P}{}{\prime\prime}\mathrm{ Chain yFreshP}\mp@subsup{P}{}{\prime\prime}\mathrm{ have ([(x,y)] • P'})[y::=u]\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime
        by(simp add: renaming name-swap)
    ultimately show ?thesis using PChain by(blast intro: transitionI)
    qed
lemma tauActionChain:
fixes P :: pi
and }\mp@subsup{P}{}{\prime}::p
shows }P\Longrightarrow\mp@subsup{}{l}{}\tau\prec\mp@subsup{P}{}{\prime}\LongrightarrowP\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime
and }P\not=\mp@subsup{P}{}{\prime}\LongrightarrowP\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime}\LongrightarrowP\mp@subsup{\Longrightarrow}{l}{}\tau\prec\mp@subsup{P}{}{\prime
proof -
assume P\Longrightarrow}\mp@subsup{\Longrightarrow}{l}{}\tau\prec\mp@subsup{P}{}{\prime
then obtain P}\mp@subsup{P}{}{\prime\prime}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ where }P\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime\prime
and }\mp@subsup{P}{}{\prime\prime}\longmapsto\tau\prec\mp@subsup{P}{}{\prime\prime\prime
and }\mp@subsup{P}{}{\prime\prime\prime}\Longrightarrow\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime
by(blast dest: transitionE)
thus }P\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime}\mathrm{ by auto
next
assume P\Longrightarrow}\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime}\mathrm{ and }P\not=\mp@subsup{P}{}{\prime
thus }P\Longrightarrow\mp@subsup{}{l}{}\tau\prec\mp@subsup{P}{}{\prime
proof(induct rule: tauChainInduct)
case id
thus ?case by simp
next
case(ih P'\prime P}\mp@subsup{P}{}{\prime\prime}
have P\Longrightarrow}\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime\prime}\mathrm{ and }\mp@subsup{P}{}{\prime\prime}\longmapsto\tau\prec\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ by fact+
moreover have P}\mp@subsup{P}{}{\prime\prime\prime}\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ by simp
ultimately show ?case by(rule transitionI)
qed
qed
lemma singleActionChain:
fixes P :: pi
and a :: name
and }x\mathrm{ :: name
and \alpha :: freeRes
and u :: name
shows }P\longmapstoa<\nux>\prec\mp@subsup{P}{}{\prime}\LongrightarrowP\mp@subsup{\Longrightarrow}{l}{}a<\nux>\prec\mp@subsup{P}{}{\prime
and }\llbracketP\longmapstoa<x>\prec\mp@subsup{P}{}{\prime}\rrbracket\LongrightarrowP\Longrightarrow\Longrightarrowlu in P P '->a<x> \prec P'[x::=u
and }P\longmapsto\alpha\prec\mp@subsup{P}{}{\prime}\LongrightarrowP\Longrightarrow\mp@subsup{\Longrightarrow}{l}{}\alpha\prec\mp@subsup{P}{}{\prime
proof -
assume P\longmapstoa<\nux> \prec P'
moreover have P\Longrightarrow}\mp@subsup{\Longrightarrow}{\tau}{}P\mathrm{ by simp
moreover have }\mp@subsup{P}{}{\prime}\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime}\mathrm{ by simp
ultimately show P \Longrightarrow\Longrightarrowla<\nux>}\prec\mp@subsup{}{l}{\prime}a\mp@subsup{P}{}{\prime}\mathbf{by}(\mathrm{ blast intro: transitionI)
next
assume }P\longmapstoa<x>\prec\mp@subsup{P}{}{\prime

```
```

    moreover have P \Longrightarrow>
    moreover have P}\mp@subsup{P}{}{\prime}[x::=u]\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime}[x::=u] by sim
    ```

```

next
assume P\longmapsto\alpha\prec
moreover have P\Longrightarrow}\mp@subsup{\Longrightarrow}{\tau}{}P\mathrm{ by simp
moreover have P' \Longrightarrow}\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime}\mathrm{ by simp
ultimately show P \Longrightarrow\Longrightarrowl }\alpha\prec\mp@subsup{P}{}{\prime}\mathrm{ by(blast intro: transitionI)
qed
lemma Tau:
fixes P :: pi
shows }\tau.(P)\Longrightarrow\mp@subsup{\Longrightarrow}{l}{}\tau\prec
proof -
have \tau.(P) \Longrightarrow\Longrightarrow
moreover have \tau.(P)\longmapsto\tau\precP by(rule transitions.Tau)
moreover have P\Longrightarrow}\mp@subsup{\Longrightarrow}{\tau}{}P\mathrm{ by simp
ultimately show ?thesis by(rule transitionI)
qed
lemma Input:
fixes a :: name
and x :: name
and u:: name
and P :: pi
shows }a<x>.P\Longrightarrow\mp@subsup{}{l}{}u\mathrm{ in }P->a<x>\prec\precP[x::=u
proof -
have }a<x>.P\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}a<x>.P\mathrm{ by simp
moreover have }a<x>.P\longmapstoa<x>\precP by(rule Input
moreover have }P[x::=u]\mp@subsup{\Longrightarrow}{\tau}{}P[x::=u] by sim
ultimately show ?thesis by(rule transitionI)
qed
lemma Output:
fixes a :: name
and b:: name
and }P::p
shows }a{b}.P\Longrightarrow\mp@subsup{\Longrightarrow}{l}{}a[b]\prec
proof -
have a{b}.P\Longrightarrow\Longrightarrow}\mp@subsup{\Longrightarrow}{\tau}{}a{b}.P\mathrm{ by simp
moreover have }a{b}.P\longmapstoa[b]\precP\mathrm{ by(rule transitions.Output)
moreover have P\Longrightarrow}\mp@subsup{\Longrightarrow}{\tau}{}P\mathrm{ by simp
ultimately show ?thesis by(rule transitionI)
qed
lemma Match:

```
```

fixes $P$ :: $p i$
and $\quad R s::$ residual
and $a$ :: name
and $u$ :: name
and $b$ :: name
and $x$ :: name
and $P^{\prime}:: p i$
shows $P \Longrightarrow_{l}$ Rs $\Longrightarrow[a \frown a] P \Longrightarrow_{l}$ Rs
and $P \Longrightarrow_{l} u$ in $P^{\prime \prime} \rightarrow b<x>\prec P^{\prime} \Longrightarrow[a \frown a] P \Longrightarrow_{l} u$ in $P^{\prime \prime} \rightarrow b<x>\prec P^{\prime}$
proof -
assume PTrans: $P \Longrightarrow{ }_{l}$ Rs
thus $[a \frown a] P \Longrightarrow{ }_{l} R s$
proof(nominal-induct avoiding: $P$ rule: residual.strong-inducts)
case (BoundR b x $P^{\prime}$ )
have PTrans: $P \Longrightarrow_{l} b « x » \prec P^{\prime}$ and $x F r e s h P: x \sharp P$ by fact +
from PTrans obtain $b^{\prime}$ where beq: $b=$ BoundOutputS $b^{\prime}$ by (cases b) auto
with PTrans xFreshP obtain $P^{\prime \prime} P^{\prime \prime \prime}$ where PTrans: $P \Longrightarrow_{\tau} P^{\prime \prime}$
and $P^{\prime \prime}$ Trans: $P^{\prime \prime} \longmapsto b^{\prime}<\nu x>\prec P^{\prime \prime \prime}$
and $P^{\prime \prime \prime}$ Trans: $P^{\prime \prime \prime} \Longrightarrow_{\tau} P^{\prime}$
by (blast dest: transitionE)
show ?case
proof (cases $P=P^{\prime \prime}$ )
assume $P=P^{\prime \prime}$
moreover have $[a \frown a] P \Longrightarrow_{\tau}[a \frown a] P$ by simp
moreover from $P^{\prime \prime}$ Trans have $[a \frown a] P^{\prime \prime} \longmapsto b^{\prime}<\nu x>\prec P^{\prime \prime \prime}$ by (rule Match)
ultimately show ?thesis using beq $P^{\prime \prime}$ Trans by (blast intro: transitionI)
next
assume $P \neq P^{\prime \prime}$
with PTrans have $[a \frown a] P \Longrightarrow_{\tau} P^{\prime \prime}$ by (rule matchChain)
thus ?thesis using beq $P^{\prime \prime}$ Trans $P^{\prime \prime \prime}$ Trans by(blast intro: transitionI)
qed
next
case (FreeR $\alpha P^{\prime}$ )
have $P \Longrightarrow_{l} \alpha \prec P^{\prime}$ by fact
then obtain $P^{\prime \prime} P^{\prime \prime \prime}$ where PTrans: $P \Longrightarrow_{\tau} P^{\prime \prime}$
and $P^{\prime \prime}$ Trans: $P^{\prime \prime} \longmapsto \alpha \prec P^{\prime \prime \prime}$
and $P^{\prime \prime \prime}$ Trans: $P^{\prime \prime \prime} \Longrightarrow{ }_{\tau} P^{\prime}$
by (blast dest: transitionE)
show? case
proof (cases $P=P^{\prime \prime}$ )
assume $P=P^{\prime \prime}$
moreover have $[a \frown a] P \Longrightarrow_{\tau}[a \frown a] P$ by simp
moreover from $P^{\prime \prime}$ Trans have $[a \frown a] P^{\prime \prime} \longmapsto \alpha \prec P^{\prime \prime \prime}$ by(rule transi-
tions.Match)
ultimately show ?thesis using $P^{\prime \prime \prime}$ Trans by (blast intro: transitionI)
next
assume $P \neq P^{\prime \prime}$

```
with PTrans have \([a \frown a] P \Longrightarrow_{\tau} P^{\prime \prime}\) by (rule matchChain)
thus ?thesis using \(P^{\prime \prime}\) Trans \(P^{\prime \prime \prime}\) Trans by (rule transitionI)
qed
qed
next
assume \(P \Longrightarrow_{l} u\) in \(P^{\prime \prime} \rightarrow b<x>\prec P^{\prime}\)
then obtain \(P^{\prime \prime \prime}\) where PChain: \(P \Longrightarrow_{\tau} P^{\prime \prime \prime}\)
and \(P^{\prime \prime \prime}\) Trans: \(P^{\prime \prime \prime} \longmapsto b<x>\prec P^{\prime \prime}\)
and \(P^{\prime \prime}\) Chain: \(P^{\prime \prime}[x::=u] \Longrightarrow_{\tau} P^{\prime}\)
by(blast dest: transitionE)
show \([a \frown a] P \Longrightarrow{ }_{l} u\) in \(P^{\prime \prime} \rightarrow b<x>\prec P^{\prime}\)
proof(cases \(P=P^{\prime \prime \prime}\) )
assume \(P=P^{\prime \prime \prime}\)
moreover have \([a \frown a] P \Longrightarrow_{\tau}[a \frown a] P\) by simp
moreover from \(P^{\prime \prime \prime}\) Trans have \([a \frown a] P^{\prime \prime \prime} \longmapsto b<x>\prec P^{\prime \prime}\) by (rule Late-Semantics.Match)
ultimately show ?thesis using \(P^{\prime \prime}\) Chain by (blast intro: transitionI)
next
assume \(P \neq P^{\prime \prime \prime}\)
with PChain have \([a \frown a] P \Longrightarrow_{\tau} P^{\prime \prime \prime}\) by (rule matchChain)
thus ?thesis using \(P^{\prime \prime \prime}\) Trans \(P^{\prime \prime}\) Chain by (rule transitionI)
qed
qed
lemma Mismatch:
fixes \(P:: p i\)
and \(R s::\) residual
and \(a\) :: name
and \(c\) :: name
and \(u\) :: name
and \(b\) :: name
and \(x\) :: name
and \(\quad P^{\prime}:: p i\)
```

shows $\llbracket P \Longrightarrow_{l}$ Rs; $a \neq c \rrbracket \Longrightarrow[a \neq c] P \Longrightarrow_{l}$ Rs
and $\llbracket P \Longrightarrow_{l} u$ in $P^{\prime \prime} \rightarrow b<x>\prec P^{\prime} ; a \neq c \rrbracket \Longrightarrow[a \neq c] P \Longrightarrow_{l} u$ in $P^{\prime \prime} \rightarrow b<x>\prec$
$P^{\prime}$
proof -
assume PTrans: $P \Longrightarrow{ }_{l}$ Rs
and aineqc: $a \neq c$
thus $[a \neq c] P \Longrightarrow{ }_{l}$ Rs
proof(nominal-induct avoiding: P rule: residual.strong-inducts)
case (BoundR b x $P^{\prime}$ )
have PTrans: $P \Longrightarrow_{l} b<x$ » $\prec P^{\prime}$ and $x F r e s h P: x \sharp P$ by fact +
from PTrans obtain $b^{\prime}$ where beq: $b=$ BoundOutputS $b^{\prime}$ by (cases b, auto)
with PTrans xFreshP obtain $P^{\prime \prime} P^{\prime \prime \prime}$ where PTrans: $P \Longrightarrow_{\tau} P^{\prime \prime}$
and $P^{\prime \prime}$ Trans: $P^{\prime \prime} \longmapsto b^{\prime}<\nu x>\prec P^{\prime \prime \prime}$
and $P^{\prime \prime \prime}$ Trans: $P^{\prime \prime \prime} \Longrightarrow{ }_{\tau} P^{\prime}$
by(blast dest: transitionE)
show ?case

```
```

    proof(cases P= P')
    assume P= P'
    moreover have [a\not=c]P\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}[a\not=c]P by simp
    moreover from }\mp@subsup{P}{}{\prime\prime}\mathrm{ Trans aineqc have [a#c] P'|}\longmapsto\mp@subsup{b}{}{\prime}<\nux><<\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ by(rule
    transitions.Mismatch)
ultimately show ?thesis using beq P}\mp@subsup{P}{}{\prime\prime}\mathrm{ Trans by(blast intro: transitionI)
next
assume P}\not=\mp@subsup{P}{}{\prime\prime
with PTrans aineqc have [a\not=c]P\Longrightarrow\Longrightarrow}\mp@subsup{\}{\tau}{}\mp@subsup{P}{}{\prime\prime}\mathbf{by}(\mathrm{ (rule mismatchChain)
thus ?thesis using beq P'Trans P '/'Trans by(blast intro: transitionI)
qed
next
case(FreeR \alpha P')
have}P\Longrightarrow\mp@subsup{\Longrightarrow}{l}{}\alpha\prec\mp@subsup{P}{}{\prime}\mathrm{ by fact
then obtain }\mp@subsup{P}{}{\prime\prime}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ where PTrans: }P\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime\prime

```

```

                    and P}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ Trans: }\mp@subsup{P}{}{\prime\prime\prime}\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime
    by(blast dest: transitionE)
    show ?case
    proof(cases P= P')
    assume P= P'
    moreover have [a\not=c]P\Longrightarrow\Longrightarrow
    ```

```

transitions.Mismatch)
ultimately show ?thesis using P '/' Trans by(blast intro: transitionI)
next
assume P}\not=\mp@subsup{P}{}{\prime\prime
with PTrans aineqc have [a\not=c]P\Longrightarrow\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime\prime}\mathbf{by}(\mathrm{ (rule mismatchChain)}
thus ?thesis using P'\primeTrans P }\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ Trans by(rule transitionI)
qed
qed
next
assume aineqc: a\not=c
assume }P\Longrightarrow\mp@subsup{\Longrightarrow}{l}{}u\mathrm{ in }\mp@subsup{P}{}{\prime\prime}->b<x>\prec\prec P'
then obtain }\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ where PChain: }P\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime\prime\prime

```

```

                    and P}\mp@subsup{P}{}{\prime\prime}\mathrm{ Chain: P}\mp@subsup{P}{}{\prime\prime}[x::=u]\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime
    by(blast dest: transitionE)
    show [a\not=c]P\Longrightarrow>lu in }\mp@subsup{P}{}{\prime\prime}->b<x>< < P
proof(cases P=P'\prime\prime)
assume P=P'\prime\prime
moreover have [a\not=c]P \Longrightarrow>
moreover from }\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ Trans aineqc have [a*c] P'/' }\longmapstob<x>\prec\prec P'\prime by(rul
Late-Semantics.Mismatch)
ultimately show ?thesis using P'Chain by(blast intro: transitionI)
next
assume P}=\mp@subsup{P}{}{\prime\prime\prime
with PChain aineqc have [a\not=c]P\Longrightarrow>}\mp@subsup{\}{\tau}{}\mp@subsup{P}{}{\prime\prime\prime}\mathbf{by}(\mathrm{ rule mismatchChain)

```
```

        thus ?thesis using P}\mp@subsup{P}{}{\prime\prime\prime}Trans P '/ Chain by(rule transitionI)
        qed
    qed
lemma Open:
fixes }P\mathrm{ :: pi
and a :: name
and b :: name
and }\mp@subsup{P}{}{\prime}:: p
assumes Trans: P \Longrightarrow\Longrightarrowl}a[b]\prec\mp@subsup{P}{}{\prime
and aInEqb: a\not=b
shows <\nub>P \Longrightarrow}\mp@subsup{}{l}{}a<\nub>\prec\mp@subsup{P}{}{\prime
proof -
from Trans obtain P}\mp@subsup{P}{}{\prime\prime}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ where A: P >}\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime\prime
and B: P'\prime\longmapstoa[b]\prec 㐌\prime\prime
and C: P
by(force dest: transitionE)
from A have <\nub>P\Longrightarrow\Longrightarrow}\mp@subsup{\}{\tau}{<\nub>\mp@subsup{P}{}{\prime\prime}}\mathbf{by}(\mathrm{ rule ResChain)
moreover from B aInEqb have <\nub>\mp@subsup{P}{}{\prime\prime}\longmapstoa<\nub>}\prec\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ by(rule Open)
ultimately show ?thesis using C by(force simp add: transition-def)
qed
lemma Sum1:
fixes P :: pi
and Rs :: residual
and }Q :: p
and u}::\mathrm{ name
and }\mp@subsup{P}{}{\prime\prime}:: p
and a :: name
and }x\mathrm{ :: name
and }\mp@subsup{P}{}{\prime}::p
shows }P\Longrightarrow\mp@subsup{\Longrightarrow}{l}{}Rs\LongrightarrowP\oplusQ\Longrightarrow\mp@subsup{}{l}{}\mathrm{ Rs

```

```

proof -
assume P\Longrightarrowl Rs
thus }P\oplusQ\Longrightarrow\mp@subsup{}{l}{}R
proof(nominal-induct avoiding: P rule: residual.strong-inducts)
case(BoundR a x P'P)
have PTrans: P\Longrightarrow\Longrightarrowla«x»\prec < P'
and xFreshP: x\sharpP by fact+
from PTrans obtain a' where aeq: a = BoundOutputS a' by(cases a, auto)
with PTrans xFreshP obtain P'\prime P
and }\mp@subsup{P}{}{\prime\prime}\mathrm{ Trans: }\mp@subsup{P}{}{\prime\prime}\longmapsto\mp@subsup{a}{}{\prime}<\nux>\prec\prec\mp@subsup{P}{}{\prime\prime\prime
and P}\mp@subsup{P}{}{\prime\prime}\mathrm{ Trans: P'/' }\Longrightarrow\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime
by(blast dest: transitionE)
show ?case

```
\[
\operatorname{proof}\left(\text { cases } P=P^{\prime \prime}\right)
\]
assume \(P=P^{\prime \prime}\)
moreover have \(P \oplus Q \Longrightarrow_{\tau} P \oplus Q\) by simp
moreover from \(P^{\prime \prime}\) Trans have \(P^{\prime \prime} \oplus Q \longmapsto a^{\prime}<\nu x>\prec P^{\prime \prime \prime}\) by (rule transitions.Sum1)
ultimately show ?thesis using \(P^{\prime \prime \prime}\) Trans aeq by(blast intro: transitionI)
next
assume \(P \neq P^{\prime \prime}\)
with PTrans have \(P \oplus Q \Longrightarrow_{\tau} P^{\prime \prime}\) by (rule sum1Chain)
thus ?thesis using \(P^{\prime \prime}\) Trans \(P^{\prime \prime \prime}\) Trans aeq by(blast intro: transitionI)
qed
next
case(FreeR \(\alpha P^{\prime}\) )
have \(P \Longrightarrow_{l} \alpha \prec P^{\prime}\) by fact
then obtain \(P^{\prime \prime} P^{\prime \prime \prime}\) where PTrans: \(P \Longrightarrow_{\tau} P^{\prime \prime}\)
and \(P^{\prime \prime}\) Trans: \(P^{\prime \prime} \longmapsto \alpha \prec P^{\prime \prime \prime}\)
and \(P^{\prime \prime \prime}\) Trans: \(P^{\prime \prime \prime} \Longrightarrow_{\tau} P^{\prime}\)
by (blast dest: transitionE)
show ?case
proof (cases \(P=P^{\prime \prime}\) )
assume \(P=P^{\prime \prime}\)
moreover have \(P \oplus Q \Longrightarrow{ }_{\tau} P \oplus Q\) by simp
moreover from \(P^{\prime \prime}\) Trans have \(P^{\prime \prime} \oplus Q \longmapsto \alpha \prec P^{\prime \prime \prime}\) by(rule transi-
tions.Sum1)
ultimately show ?thesis using \(P^{\prime \prime \prime}\) Trans by (blast intro: transitionI)
next
assume \(P \neq P^{\prime \prime}\)
with PTrans have \(P \oplus Q \Longrightarrow_{\tau} P^{\prime \prime}\) by (rule sum1Chain)
thus ?thesis using \(P^{\prime \prime}\) Trans \(P^{\prime \prime \prime}\) Trans by (rule transitionI)
qed
qed
next
assume \(P \Longrightarrow_{l} u\) in \(P^{\prime \prime} \rightarrow a<x>\prec P^{\prime}\)
then obtain \(P^{\prime \prime \prime}\) where PChain: \(P \Longrightarrow_{\tau} P^{\prime \prime \prime}\)
and \(P^{\prime \prime \prime}\) Trans: \(P^{\prime \prime \prime} \longmapsto a<x>\prec P^{\prime \prime}\)
and \(P^{\prime \prime}\) Chain: \(P^{\prime \prime}[x::=u] \Longrightarrow_{\tau} P^{\prime}\)
by (blast dest: transitionE)
show \(P \oplus Q \Longrightarrow_{l} u\) in \(P^{\prime \prime} \rightarrow a<x>\prec P^{\prime}\)
proof (cases \(P=P^{\prime \prime \prime}\) )
assume \(P=P^{\prime \prime \prime}\)
moreover have \(P \oplus Q \Longrightarrow_{\tau} P \oplus Q\) by simp
moreover from \(P^{\prime \prime \prime}\) Trans have \(P^{\prime \prime \prime} \oplus Q \longmapsto a<x>\prec P^{\prime \prime}\) by(rule transitions.Sum1)
ultimately show ?thesis using \(P^{\prime \prime}\) Chain by(blast intro: transitionI)
next
assume \(P \neq P^{\prime \prime \prime}\)
with PChain have \(P \oplus Q \Longrightarrow_{\tau} P^{\prime \prime \prime}\) by (rule sum1Chain)
thus ?thesis using \(P^{\prime \prime \prime}\) Trans \(P^{\prime \prime}\) Chain by (blast intro: transitionI)
qed
qed
lemma Sum2:
fixes \(Q\) :: pi
and \(R s\) :: residual
and \(P\) :: pi
and \(u\) :: name
and \(Q^{\prime \prime}:: p i\)
and \(a\) :: name
and \(x\) :: name
and \(\quad Q^{\prime}:: p i\)
shows \(Q \Longrightarrow_{l}\) Rs \(\Longrightarrow P \oplus Q \Longrightarrow_{l}\) Rs
and \(Q \Longrightarrow_{l} u\) in \(Q^{\prime \prime} \rightarrow a<x>\prec Q^{\prime} \Longrightarrow P \oplus Q \Longrightarrow_{l} u\) in \(Q^{\prime \prime} \rightarrow a<x>\prec Q^{\prime}\)
proof -
assume \(Q \Longrightarrow_{l}\) Rs
thus \(P \oplus Q \Longrightarrow l\) Rs
proof(nominal-induct avoiding: \(Q\) rule: residual.strong-inducts)
case (BoundR a x \(Q^{\prime} Q\) )
have \(Q\) Trans: \(\left.Q \Longrightarrow_{l} a « x\right\rangle \prec Q^{\prime}\)
and \(x\) Fresh \(Q: x \sharp Q\) by fact +
from QTrans obtain \(a^{\prime}\) where aeq: \(a=\) BoundOutputS \(a^{\prime}\) by (cases a, auto)
with \(Q\) Trans \(x\) Fresh \(Q\) obtain \(Q^{\prime \prime} Q^{\prime \prime \prime}\) where \(Q\) Trans: \(Q \Longrightarrow_{\tau} Q^{\prime \prime}\)
and \(Q^{\prime \prime}\) Trans: \(Q^{\prime \prime} \longmapsto a^{\prime}<\nu x>\prec Q^{\prime \prime \prime}\)
and \(Q^{\prime \prime \prime}\) Trans: \(Q^{\prime \prime \prime} \Longrightarrow{ }_{\tau} Q^{\prime}\)
by (blast dest: transitionE)
show ?case
\(\operatorname{proof}\left(\right.\) cases \(\left.Q=Q^{\prime \prime}\right)\)
assume \(Q=Q^{\prime \prime}\)
moreover have \(P \oplus Q \Longrightarrow_{\tau} P \oplus Q\) by simp
moreover from \(Q^{\prime \prime}\) Trans have \(P \oplus Q^{\prime \prime} \longmapsto a^{\prime}<\nu x>\prec Q^{\prime \prime \prime}\) by (rule transi-
tions.Sum2)
ultimately show ?thesis using \(Q^{\prime \prime \prime}\) Trans aeq by (blast intro: transitionI)
next
assume \(Q \neq Q^{\prime \prime}\)
with \(Q\) Trans have \(P \oplus Q \Longrightarrow_{\tau} Q^{\prime \prime}\) by (rule sum2Chain)
thus ?thesis using \(Q^{\prime \prime}\) Trans \(Q^{\prime \prime \prime}\) Trans aeq by(blast intro: transitionI)
qed
next
case(FreeR \(\alpha Q^{\prime}\) )
have \(Q \Longrightarrow_{l} \alpha \prec Q^{\prime}\) by fact
then obtain \(Q^{\prime \prime} Q^{\prime \prime \prime}\) where \(Q\) Trans: \(Q \Longrightarrow_{\tau} Q^{\prime \prime}\)
and \(Q^{\prime \prime}\) Trans: \(Q^{\prime \prime} \longmapsto \alpha \prec Q^{\prime \prime \prime}\)
and \(Q^{\prime \prime \prime}\) Trans: \(Q^{\prime \prime \prime} \Longrightarrow_{\tau} Q^{\prime}\)
by (blast dest: transitionE)
show ?case
proof (cases \(Q=Q^{\prime \prime}\) )
```

    assume Q = Q '
    moreover have }P\oplusQ\mp@subsup{\Longrightarrow}{\tau}{}P\oplusQ\mathrm{ by simp
        moreover from Q ''Trans have }P\oplus\mp@subsup{Q}{}{\prime\prime}\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime\prime\prime}\mathrm{ by(rule transi-
    tions.Sum2)
ultimately show ?thesis using Q"'Trans by(blast intro: transitionI)
next
assume Q}=\mp@subsup{Q}{}{\prime\prime
with QTrans have }P\oplusQ\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{Q}{}{\prime\prime}\mathrm{ by(rule sum2Chain)
thus ?thesis using Q"Trans Q '/'Trans by(rule transitionI)
qed
qed
next
assume Q \Longrightarrow}\mp@subsup{l}{l}{}u\mathrm{ in }\mp@subsup{Q}{}{\prime\prime}->a<x>\prec\prec\mp@subsup{Q}{}{\prime
then obtain }\mp@subsup{Q}{}{\prime\prime\prime}\mathrm{ where QChain: Q >}\mp@subsup{}{\tau}{}\mp@subsup{Q}{}{\prime\prime\prime
and }\mp@subsup{Q}{}{\prime\prime\prime}Trans: Q '"\prime \longmapstoa<x>\prec Q ''
and Q'Chain: Q '}[x::=u]\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{Q}{}{\prime
by(blast dest: transitionE)
show }P\oplusQ\Longrightarrow\mp@subsup{\Longrightarrow}{l}{}u\mathrm{ in }\mp@subsup{Q}{}{\prime\prime}->a<x>\prec\prec\mp@subsup{Q}{}{\prime
proof(cases Q = Q''\prime)
assume Q = Q '"\prime
moreover have }P\oplusQ\mp@subsup{\Longrightarrow}{\tau}{}P\oplusQ\mathrm{ by simp

```

```

tions.Sum2)
ultimately show ?thesis using Q"Chain by(blast intro: transitionI)
next
assume Q = Q ''\prime
with QChain have P}\oplusQ\Longrightarrow\mp@subsup{}{\tau}{}\mp@subsup{Q}{}{\prime\prime\prime}\mathrm{ by(rule sum2Chain)
thus ?thesis using Q '"'Trans Q'Chain by(blast intro: transitionI)
qed
qed
lemma Par1B:
fixes P :: pi
and a :: name
and }x\mathrm{ :: name
and }\mp@subsup{P}{}{\prime}::p
and }u\mathrm{ :: name
and }\mp@subsup{P}{}{\prime\prime}::p
shows }\llbracketP\Longrightarrow\mp@subsup{\Longrightarrow}{l}{}a<\nux><\mp@subsup{P}{}{\prime};x\sharpQ\rrbracket\LongrightarrowP|Q <br>Longrightarrowlla<\nux> \prec ( P'| | Q
and }\llbracketP\Longrightarrow\mp@subsup{\Longrightarrow}{l}{}u\mathrm{ in }\mp@subsup{P}{}{\prime\prime}->a<x>\prec\prec\mp@subsup{P}{}{\prime};x\sharpQ\rrbracket\LongrightarrowP|Q\Longrightarrow\mp@subsup{\Longrightarrow}{l}{}u\mathrm{ in ( }\mp@subsup{P}{}{\prime\prime}|Q)->a<x
\prec P'| Q
proof -
assume PTrans: P \Longrightarrow\Longrightarrowl }a<\nux>\prec\mp@subsup{P}{}{\prime
assume xFreshQ:x\sharpQ
have Goal: \P a x P' Q. \llbracketP \Longrightarrow>l a<\nux> \prec P'; x\sharpP;x\sharpQ\rrbracket\LongrightarrowP| |
\Longrightarrowl}\mp@subsup{l}{l}{}a<\nux>< \prec(\mp@subsup{P}{}{\prime}|Q
proof -

```
```

    fix \(P a x P^{\prime} Q\)
    assume PTrans: \(P \Longrightarrow_{l} a<\nu x>\prec P^{\prime}\)
    assume \(x\) Fresh \(P: x \sharp P\)
    assume \(x\) Fresh \(Q: x \sharp(Q:: p i)\)
    from PTrans xFreshP obtain \(P^{\prime \prime} P^{\prime \prime \prime}\) where \(P\) Trans: \(P \Longrightarrow{ }_{\tau} P^{\prime \prime}\)
            and \(P^{\prime \prime}\) Trans: \(P^{\prime \prime} \longmapsto a<\nu x>\prec P^{\prime \prime \prime}\)
            and \(P^{\prime \prime \prime}\) Trans: \(P^{\prime \prime \prime} \Longrightarrow{ }_{\tau} P^{\prime}\)
    by (blast dest: transitionE)
    from PTrans have \(P\left\|Q \Longrightarrow_{\tau} P^{\prime \prime}\right\| Q\) by(rule Par1Chain)
    moreover from \(P^{\prime \prime}\) Trans \(x\) Fresh \(Q\) have \(P^{\prime \prime} \| Q \longmapsto a<\nu x>\prec\left(P^{\prime \prime \prime} \| Q\right)\)
    by (rule Par1B)
moreover from $P^{\prime \prime \prime}$ Trans have $P^{\prime \prime \prime}\left\|Q \Longrightarrow_{\tau} P^{\prime}\right\| Q$ by (rule Par1Chain)
ultimately show $P \| Q \Longrightarrow_{l} a<\nu x>\prec\left(P^{\prime} \| Q\right)$ by (rule transitionI)
qed
have $\exists c::$ name. $c \sharp\left(P, P^{\prime}, Q\right)$ by (blast intro: name-exists-fresh)
then obtain $c::$ name where cFresh $P: c \sharp P$ and cFresh $P^{\prime}: c \sharp P^{\prime}$ and $c F r e s h Q$ :
$c \sharp Q$
by(force simp add: fresh-prod)
from $c$ Fresh $P^{\prime}$ have $a<\nu x>\prec P^{\prime}=a<\nu c>\prec\left([(x, c)] \cdot P^{\prime}\right) \mathbf{b y}($ rule alphaBound-
Residual)
moreover have $a<\nu x>\prec\left(P^{\prime} \| Q\right)=a<\nu c>\prec\left(\left([(x, c)] \cdot P^{\prime}\right) \| Q\right)$
proof -
from $c$ Fresh $P^{\prime}$ cFresh $Q$ have $c \sharp P^{\prime} \| Q$ by simp
hence $a<\nu x>\prec\left(P^{\prime} \| Q\right)=a<\nu c>\prec\left([(x, c)] \cdot\left(P^{\prime} \| Q\right)\right)$ by (rule alphaBound-
Residual)
with $c$ Fresh $Q x F r e s h Q$ show ?thesis by (simp add: name-fresh-fresh)
qed
ultimately show $P\left\|Q \Longrightarrow_{l} a<\nu x>\prec P^{\prime}\right\| Q$ using PTrans cFreshP cFresh $Q$
by(force intro: Goal)
next
assume PTrans: $P \Longrightarrow_{l} u$ in $P^{\prime \prime} \rightarrow a<x>\prec P^{\prime}$
and $x$ Fresh $Q$ : $x \sharp Q$
from PTrans obtain $P^{\prime \prime \prime}$ where PChain: $P \Longrightarrow_{\tau} P^{\prime \prime \prime}$
and $P^{\prime \prime \prime}$ Trans: $P^{\prime \prime \prime} \longmapsto a<x>\prec P^{\prime \prime}$
and $P^{\prime \prime}$ Chain: $P^{\prime \prime}[x::=u] \Longrightarrow{ }_{\tau} P^{\prime}$
by (blast dest: transitionE)
from PChain have $P\left\|Q \Longrightarrow_{\tau} P^{\prime \prime \prime}\right\| Q$ by(rule Par1Chain)
moreover from $P^{\prime \prime \prime}$ Trans xFresh $Q$ have $P^{\prime \prime \prime} \| Q \longmapsto a<x>\prec\left(P^{\prime \prime} \| Q\right)$ by $($ rule
Par1B)
moreover have $\left(P^{\prime \prime} \| Q\right)[x::=u] \Longrightarrow_{\tau} P^{\prime} \| Q$
proof -
from $P^{\prime \prime}$ Chain have $P^{\prime \prime}[x::=u]\left\|Q \Longrightarrow{ }_{\tau} P^{\prime}\right\| Q$ by(rule Par1Chain)
with $x$ Fresh $Q$ show?thesis by (simp add: forget)
qed
ultimately show $P \| Q \Longrightarrow_{l} u$ in $\left(P^{\prime \prime} \| Q\right) \rightarrow a<x>\prec\left(P^{\prime} \| Q\right)$ by (rule transi-
tionI)

```

\section*{qed}
```

lemma Par1F
fixes $P:: p i$
and $\alpha$ :: freeRes
and $P^{\prime}:: p i$
assumes PTrans: $P \Longrightarrow_{l} \alpha \prec P^{\prime}$
shows $P \| Q \Longrightarrow_{l} \alpha \prec\left(P^{\prime} \| Q\right)$
proof -
from PTrans obtain $P^{\prime \prime} P^{\prime \prime \prime}$ where PTrans: $P \Longrightarrow_{\tau} P^{\prime \prime}$
and $P^{\prime \prime}$ Trans: $P^{\prime \prime} \longmapsto \alpha \prec P^{\prime \prime \prime}$
and $P^{\prime \prime \prime}$ Trans: $P^{\prime \prime \prime} \Longrightarrow_{\tau} P^{\prime}$
by(blast dest: transitionE)
from PTrans have $P\left\|Q \Longrightarrow_{\tau} P^{\prime \prime}\right\| Q$ by (rule Par1Chain)
moreover from $P^{\prime \prime}$ Trans have $P^{\prime \prime} \| Q \longmapsto \alpha \prec\left(P^{\prime \prime \prime} \| Q\right)$ by (rule transi-
tions.Par1F)
moreover from $P^{\prime \prime \prime}$ Trans have $P^{\prime \prime \prime}\left\|Q \Longrightarrow_{\tau} P^{\prime}\right\| Q$ by(rule Par1Chain)
ultimately show ?thesis by (rule transitionI)
qed
lemma Par2B:
fixes $Q:: p i$
and $a$ :: name
and $x$ :: name
and $\quad Q^{\prime}:: p i$
and $P:: p i$
and $u$ :: name
and $\quad Q^{\prime \prime}:: p i$
shows $Q \Longrightarrow_{l} a<\nu x>\prec Q^{\prime} \Longrightarrow x \sharp P \Longrightarrow P \| Q \Longrightarrow_{l} a<\nu x>\prec\left(P \| Q^{\prime}\right)$
and $Q \Longrightarrow_{l} u$ in $Q^{\prime \prime} \rightarrow a<x>\prec Q^{\prime} \Longrightarrow x \sharp P \Longrightarrow P \| Q \Longrightarrow_{l} u$ in $(P \|$
$\left.Q^{\prime \prime}\right) \rightarrow a<x>\prec P \| Q^{\prime}$
proof -
assume $Q$ Trans: $Q \Longrightarrow_{l} a<\nu x>\prec Q^{\prime}$
assume $x$ Fresh $P: x \sharp P$
have Goal: $\bigwedge Q$ a x $Q^{\prime} P . \llbracket Q \Longrightarrow_{l} a<\nu x>\prec Q^{\prime} ; x \sharp Q ; x \sharp P \rrbracket \Longrightarrow P \| Q$
$\Longrightarrow_{l} a<\nu x>\prec\left(P \| Q^{\prime}\right)$
proof -
fix $Q$ ax $Q^{\prime} P$
assume $Q$ Trans: $Q \Longrightarrow_{l} a<\nu x>\prec Q^{\prime}$
assume $x$ Fresh $Q: x \sharp Q$
assume $x$ Fresh $P: x \sharp(P:: p i)$
from $Q$ Trans xFresh $Q$ obtain $Q^{\prime \prime} Q^{\prime \prime \prime}$ where $Q$ Trans: $Q \Longrightarrow_{\tau} Q^{\prime \prime}$
and $Q^{\prime \prime}$ Trans: $Q^{\prime \prime} \longmapsto a<\nu x>\prec Q^{\prime \prime \prime}$
and $Q^{\prime \prime \prime}$ Trans: $Q^{\prime \prime \prime} \Longrightarrow{ }_{\tau} Q^{\prime}$

```
```

        by(blast dest: transitionE)
    from QTrans have P|Q \Longrightarrow\Longrightarrow}\mp@subsup{}{\tau}{}P|\mp@subsup{Q}{}{\prime\prime}\mathbf{by}(\mathrm{ rule Par2Chain)
    ```

```

by(rule Par2B)
moreover from Q ''\prime Trans have }P|\mp@subsup{Q}{}{\prime\prime\prime}\mp@subsup{\Longrightarrow}{\tau}{}P|\mp@subsup{Q}{}{\prime}\mathbf{by}(\mathrm{ rule Par2Chain)
ultimately show }P|Q\mp@subsup{\Longrightarrow}{l}{}a<\nux>\prec\prec(P|\mp@subsup{Q}{}{\prime})\mathbf{by}(\mathrm{ rule transitionI)
qed
have \existsc::name. c \# (Q, Q',P) by(blast intro: name-exists-fresh)
then obtain c::name where cFreshQ:c\sharpQ and cFreshQ':c\sharp ' 利 cFreshP:
c\sharpP
by(force simp add: fresh-prod)
from cFresh\mp@subsup{Q}{}{\prime}}\mathrm{ have }a<\nux>\prec\mp@subsup{Q}{}{\prime}=a<\nuc> \prec([(x,c)] \cdot Q') by(rule al-
phaBoundResidual)
moreover have }a<\nux>\prec(P|\mp@subsup{Q}{}{\prime})=a<\nuc>\prec\prec(P|([(x,c)]\cdot\mp@subsup{Q}{}{\prime})
proof -
from cFresh Q' cFreshP have c\sharpP| Q' by simp
hence }a<\nux>\prec\prec(P|\mp@subsup{Q}{}{\prime})=a<\nuc>\prec\prec([(x,c)]\cdot(P|\mp@subsup{Q}{}{\prime}))\mathrm{ by(rule alphaBound-
Residual)
with cFreshP xFreshP show ?thesis by(simp add: name-fresh-fresh)
qed
ultimately show P|Q \Longrightarrow\Longrightarrowl a<\nux>>\precP| Q' using QTrans cFreshQ cFreshP
by(force intro:Goal)
next
assume QTrans: Q \Longrightarrow\Longrightarrowlu in }\mp@subsup{Q}{}{\prime\prime}->a<x>\prec\mp@subsup{Q}{}{\prime
and xFreshP: x\sharpP
from QTrans obtain Q '"\prime where QChain: Q \Longrightarrow>
and }\mp@subsup{Q}{}{\prime\prime\prime}Trans: Q '"\prime \longmapstoa<x> \prec Q ''
and \mp@subsup{Q}{}{\prime\prime}Chain: }\mp@subsup{Q}{}{\prime\prime}[x::=u]\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{Q}{}{\prime
by(blast dest: transitionE)
from QChain have P|Q \Longrightarrow>
moreover from }\mp@subsup{Q}{}{\prime\prime\prime}\mathrm{ Trans xFreshP have P| Q'"}\longmapstoa<x>\prec(P| Q'\prime) by(rule
Par2B)
moreover have (P| Q '})[x::=u]\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}P|\mp@subsup{Q}{}{\prime
proof -
from Q ''Chain have P| ( Q'|}[x:=u])\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}P|\mp@subsup{Q}{}{\prime}\mathbf{by}(\mathrm{ rule Par2Chain)
with xFreshP show ?thesis by (simp add: forget)
qed
ultimately show P|Q \Longrightarrow>lu in (P| Q'')->a<x> \prec(P| Q') by(rule transi-
tionI)
qed
lemma Par2F:
fixes }Q:: p
and \alpha :: freeRes
and }\mp@subsup{Q}{}{\prime}:: p
assumes QTrans: Q >> }\alpha\prec\mp@subsup{Q}{}{\prime

```
```

    shows }P|Q\Longrightarrow\mp@subsup{}{l}{}\alpha\prec(P|\mp@subsup{Q}{}{\prime}
    proof -
from QTrans obtain }\mp@subsup{Q}{}{\prime\prime}\mp@subsup{Q}{}{\prime\prime\prime}\mathrm{ where QTrans: }Q\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{Q}{}{\prime\prime
and }\mp@subsup{Q}{}{\prime\prime}\mathrm{ Trans: }\mp@subsup{Q}{}{\prime\prime}\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime\prime\prime
and Q '/'Trans: Q '"\prime \Longrightarrow}\mp@subsup{}{\tau}{}\mp@subsup{Q}{}{\prime
by(blast dest: transitionE)
from QTrans have P|Q \Longrightarrow>}\mp@subsup{T}{}{\prime}P|\mp@subsup{Q}{}{\prime\prime}\mathbf{by}(rule Par2Chain)
moreover from }\mp@subsup{Q}{}{\prime\prime}\mathrm{ Trans have }P|\mp@subsup{Q}{}{\prime\prime}\longmapsto\alpha\prec(P| Q ''\prime) by(rule transi-
tions.Par2F)

```

```

    ultimately show ?thesis by(rule transitionI)
    qed
lemma Comm1:
fixes P :: pi
and b :: name
and }\mp@subsup{P}{}{\prime\prime}::p
and a :: name
and }x\mathrm{ :: name
and }\mp@subsup{P}{}{\prime}::p
and }Q ::p
and }\mp@subsup{Q}{}{\prime}::p
assumes PTrans: }P\mp@subsup{\Longrightarrow}{l}{}b\mathrm{ b in }\mp@subsup{P}{}{\prime\prime}->a<x>\prec < P
and QTrans:Q \Longrightarrow}\mp@subsup{l}{l}{}a[b]\prec\mp@subsup{Q}{}{\prime
shows }P|Q\Longrightarrow\mp@subsup{}{l}{}\tau\prec\mp@subsup{P}{}{\prime}|\mp@subsup{Q}{}{\prime
proof -
from PTrans obtain P}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ where PChain: P }\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime\prime\prime
and }\mp@subsup{P}{}{\prime\prime\prime}Trans: P'\prime\prime \longmapstoa<x> \prec P'\prime
and }\mp@subsup{P}{}{\prime\prime}\mathrm{ Chain: }\mp@subsup{P}{}{\prime\prime}[x::=b]\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime
by(blast dest: transitionE)
from QTrans obtain }\mp@subsup{Q}{}{\prime\prime}\mp@subsup{Q}{}{\prime\prime\prime}\mathrm{ where QChain: }Q\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{Q}{}{\prime\prime\prime
and }\mp@subsup{Q}{}{\prime\prime\prime}T\mathrm{ Trans: }\mp@subsup{Q}{}{\prime\prime\prime}\longmapstoa[b]\prec\mp@subsup{Q}{}{\prime\prime
and Q"Chain: Q }\mp@subsup{Q}{}{\prime\prime}\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{Q}{}{\prime
by(blast dest: transitionE)
from PChain QChain have P|Q \Longrightarrow>}\mp@subsup{\tau}{}{\prime}\mp@subsup{P}{}{\prime\prime\prime}|\mp@subsup{Q}{}{\prime\prime\prime}\mathrm{ by(rule chainPar)

```

```

    by(rule Comm1)
    ```

```

chainPar)
ultimately show ?thesis by(rule transitionI)
qed
lemma Comm2:
fixes P :: pi
and a :: name

```
```

    and b :: name
    and }\mp@subsup{P}{}{\prime}::p
    and }Q :: p
    and }x\mathrm{ :: name
    and }\mp@subsup{Q}{}{\prime\prime}::p
    and }\mp@subsup{Q}{}{\prime}:: p
    assumes PTrans: P \Longrightarrow\Longrightarrowla[b]\prec P'
    and QTrans:Q \Longrightarrow}\mp@subsup{l}{l}{}b\mathrm{ in }\mp@subsup{Q}{}{\prime\prime}->a<x>\prec\prec Q
    shows }P|Q\Longrightarrow\mp@subsup{}{l}{}\tau\prec\mp@subsup{P}{}{\prime}|\mp@subsup{Q}{}{\prime
    proof -
from PTrans obtain P'\prime P
and P}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ Trans: }\mp@subsup{P}{}{\prime\prime\prime}\longmapstoa[b]\prec\mp@subsup{P}{}{\prime\prime
and P'Chain: P'\prime}\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime
by(blast dest: transitionE)
from QTrans obtain }\mp@subsup{Q}{}{\prime\prime\prime}\mathrm{ where QChain: Q >}\mp@subsup{}{\tau}{}\mp@subsup{Q}{}{\prime\prime\prime
and }\mp@subsup{Q}{}{\prime\prime\prime}\mathrm{ Trans: }\mp@subsup{Q}{}{\prime\prime\prime}\longmapstoa<x>\prec < Q'
and }\mp@subsup{Q}{}{\prime\prime}\mathrm{ Chain: }\mp@subsup{Q}{}{\prime\prime}[x::=b]\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{Q}{}{\prime
by(blast dest: transitionE)
from PChain QChain have P|Q \Longrightarrow>}\mp@subsup{\tau}{}{\prime}\mp@subsup{P}{}{\prime\prime\prime}|\mp@subsup{Q}{}{\prime\prime\prime}\mathbf{by}(rule chainPar)

```

```

        by(rule Comm2)
    ```

```

chainPar)
ultimately show ?thesis by(rule transitionI)
qed
lemma Close1:
fixes P :: pi
and y :: name
and }\mp@subsup{P}{}{\prime\prime}::p
and a :: name
and }x\mathrm{ :: name
and }\mp@subsup{P}{}{\prime}::p
and }Q :: p
and }\mp@subsup{Q}{}{\prime}::p
assumes PTrans: P\Longrightarrow}\mp@subsup{\Longrightarrow}{l}{ly}\mathrm{ in }\mp@subsup{P}{}{\prime\prime}->a<x>\prec < P
and QTrans: Q \Longrightarrow>la<\nuy>\prec < Q'
and yFreshP:y\sharpP
and yFreshQ: y\sharpQ
shows }P|Q\Longrightarrow\mp@subsup{}{l}{}\tau\prec<\nuy>(\mp@subsup{P}{}{\prime}|\mp@subsup{Q}{}{\prime}
proof -
from PTrans obtain P}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ where PChain: P >}\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime\prime\prime

```

```

        and }\mp@subsup{P}{}{\prime\prime}\mathrm{ Chain: }\mp@subsup{P}{}{\prime\prime}[x::=y]\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime
    ```
```

    by(blast dest: transitionE)
    from QTrans yFreshQ obtain }\mp@subsup{Q}{}{\prime\prime}\mp@subsup{Q}{}{\prime\prime\prime}\mathrm{ where QChain: }Q\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{Q}{}{\prime\prime\prime
                                    and Q Q"'Trans: }\mp@subsup{Q}{}{\prime\prime\prime}\longmapstoa<\nuy>\prec\mp@subsup{Q}{}{\prime\prime
                                    and Q'Chain: Q' }\mp@subsup{\Longrightarrow}{\tau}{\prime}\mp@subsup{Q}{}{\prime
    by(blast dest: transitionE)
    from PChain yFreshP have yFreshP\mp@subsup{P}{}{\prime\prime\prime}:y\sharp\mp@subsup{P}{}{\prime\prime\prime}}\mathbf{by}(rule freshChain)
    from PChain QChain have P| Q \Longrightarrow>}\mp@subsup{\tau}{\tau}{}\mp@subsup{P}{}{\prime\prime\prime}|\mp@subsup{Q}{}{\prime\prime\prime}\mathbf{by}(rule chainPar)
    moreover from P }\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ Trans Q 年Trans yFreshP}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ have }\mp@subsup{P}{}{\prime\prime\prime}|\mp@subsup{Q}{}{\prime\prime\prime}\longmapsto\tau\prec<\nuy>(\mp@subsup{P}{}{\prime\prime}[x::=y
    | ( Q ')
by(rule Close1)
moreover have <\nuy>(\mp@subsup{P}{}{\prime\prime}[x::=y]| \mp@subsup{Q}{}{\prime\prime})\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}<\nuy>(\mp@subsup{P}{}{\prime}|\mp@subsup{Q}{}{\prime})
proof -
from P ''Chain Q Chain have }\mp@subsup{P}{}{\prime\prime}[x::=y]|\mp@subsup{Q}{}{\prime\prime}\Longrightarrow\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime}|\mp@subsup{Q}{}{\prime}\mathrm{ by(rule chainPar)
thus ?thesis by(rule ResChain)
qed
ultimately show P| Q \Longrightarrow> }\tau<<\nuy>(\mp@subsup{P}{}{\prime}|\mp@subsup{Q}{}{\prime})\mathbf{by}(rule transitionI
qed
lemma Close2:
fixes P :: pi
and y :: name
and a :: name
and }x\mathrm{ :: name
and }\mp@subsup{P}{}{\prime}::p
and }Q :: p
and }\mp@subsup{Q}{}{\prime\prime}:: p
and }\mp@subsup{Q}{}{\prime}:: p
assumes PTrans: P \Longrightarrow>l}\mp@subsup{l}{}{\prime}a<\nuy><\mp@subsup{P}{}{\prime
and QTrans:Q \Longrightarrow}\mp@subsup{}{l}{}y\mathrm{ in }\mp@subsup{Q}{}{\prime\prime}->a<x>\prec\prec\mp@subsup{Q}{}{\prime
and yFreshP:y\sharpP
and yFreshQ:y\sharpQ
shows }P|Q\Longrightarrow\mp@subsup{}{l}{}\tau\prec<\nuy>(\mp@subsup{P}{}{\prime}|\mp@subsup{Q}{}{\prime}
proof -
from PTrans yFreshP obtain P}\mp@subsup{P}{}{\prime\prime}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ where PChain: P >}\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime\prime\prime
and }\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ Trans: }\mp@subsup{P}{}{\prime\prime\prime}\longmapstoa<\nuy> \prec P ''
and P'Chain: P}\mp@subsup{P}{}{\prime\prime}\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime
by(blast dest: transitionE)
from QTrans obtain Q }\mp@subsup{}{}{\prime\prime\prime}\mathrm{ where QChain: Q >}\mp@subsup{}{\tau}{}\mp@subsup{Q}{}{\prime\prime\prime
and }\mp@subsup{Q}{}{\prime\prime\prime}Trans: Q '/\prime \longmapstoa<x>\prec\prec\mp@subsup{Q}{}{\prime\prime
and }\mp@subsup{Q}{}{\prime\prime}\mathrm{ Chain: }\mp@subsup{Q}{}{\prime\prime}[x::=y]\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{Q}{}{\prime
by(blast dest: transitionE)
from QChain yFreshQ have yFreshQ'\prime\prime: y \# Q'/' by(rule freshChain)

```
```

    from PChain QChain have \(P\left\|Q \Longrightarrow_{\tau} P^{\prime \prime \prime}\right\| Q^{\prime \prime \prime}\) by (rule chainPar)
    moreover from \(P^{\prime \prime \prime}\) Trans \(Q^{\prime \prime \prime}\) Trans yFresh \(Q^{\prime \prime \prime}\) have \(P^{\prime \prime \prime} \| Q^{\prime \prime \prime} \longmapsto \tau \prec<\nu y>\left(P^{\prime \prime}\right.\)
    \(\left.\|\left(Q^{\prime \prime}[x::=y]\right)\right)\)
    by(rule Close2)
    moreover have \(<\nu y>\left(P^{\prime \prime} \|\left(Q^{\prime \prime}[x::=y]\right)\right) \Longrightarrow_{\tau}<\nu y>\left(P^{\prime} \| Q^{\prime}\right)\)
    proof -
    from \(P^{\prime \prime}\) Chain \(Q^{\prime \prime}\) Chain have \(P^{\prime \prime}\left\|\left(Q^{\prime \prime}[x::=y]\right) \Longrightarrow_{\tau} P^{\prime}\right\| Q^{\prime}\) by(rule chain-
    Par)
thus ?thesis by(rule ResChain)
qed
ultimately show $P \| Q \Longrightarrow{ }_{l} \tau \prec<\nu y>\left(P^{\prime} \| Q^{\prime}\right)$ by (rule transition $I$ )
qed
lemma ResF:
fixes $P$ :: $p i$
and $\alpha$ :: freeRes
and $P^{\prime}:: p i$
and $x$ :: name
assumes PTrans: $P \Longrightarrow_{l} \alpha \prec P^{\prime}$
and $\quad x F r e s h A l p h a: x \sharp \alpha$
shows $<\nu x>P \Longrightarrow{ }_{l} \alpha \prec<\nu x>P^{\prime}$
proof -
from PTrans obtain $P^{\prime \prime} P^{\prime \prime \prime}$ where PChain: $P \Longrightarrow_{\tau} P^{\prime \prime}$
and $P^{\prime \prime}$ Trans: $P^{\prime \prime} \longmapsto \alpha \prec P^{\prime \prime \prime}$
and $P^{\prime \prime \prime}$ Chain: $P^{\prime \prime \prime} \Longrightarrow{ }_{\tau} P^{\prime}$
by (blast dest: transitionE)
from PChain have $<\nu x>P \Longrightarrow_{\tau}<\nu x>P^{\prime \prime}$ by(rule ResChain)
moreover from $P^{\prime \prime}$ Trans xFreshAlpha have $<\nu x>P^{\prime \prime} \longmapsto \alpha \prec<\nu x>P^{\prime \prime \prime}$
by (rule transitions.ResF)
moreover from $P^{\prime \prime \prime}$ Chain have $<\nu x>P^{\prime \prime \prime} \Longrightarrow_{\tau}\left\langle\nu x>P^{\prime}\right.$ by(rule ResChain)
ultimately show ?thesis by(rule transitionI)
qed
lemma ResB:
fixes $P:: p i$
and $a$ :: name
and $x$ :: name
and $P^{\prime}:: p i$
and $y$ :: name
and $u$ :: name
and $P^{\prime \prime}:: p i$
shows $\llbracket P \Longrightarrow_{l} a<\nu x>\prec P^{\prime} ; y \neq a ; y \neq x ; x \sharp P \rrbracket \Longrightarrow<\nu y>P \Longrightarrow_{l} a<\nu x>\prec$
$\left(<\nu y>P^{\prime}\right)$
and $\llbracket P \Longrightarrow_{l} u$ in $P^{\prime \prime} \rightarrow a<x>\prec P^{\prime} ; y \neq a ; y \neq x ; y \neq u \rrbracket \Longrightarrow<\nu y>P \Longrightarrow_{l} u$ in
$\left(<\nu y>P^{\prime \prime}\right) \rightarrow a<x>\prec\left(<\nu y>P^{\prime}\right)$

```
```

proof -
assume PTrans: P\Longrightarrow}\mp@subsup{\Longrightarrow}{l}{}a<\nux>\prec\mp@subsup{P}{}{\prime
and yineqa: y }\not=
and yineqx: }y\not=
and xFreshP: x\sharpP
from PTrans xFreshP obtain P}\mp@subsup{P}{}{\prime\prime}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ where PChain: P }\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime\prime
and }\mp@subsup{P}{}{\prime\prime}\mathrm{ Trans: }\mp@subsup{P}{}{\prime\prime}\longmapstoa<\nux>\prec\prec P'\prime
and P}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ Chain: }\mp@subsup{P}{}{\prime\prime\prime}\Longrightarrow\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime
by(blast dest: transitionE)
from PChain have <\nuy>P \Longrightarrow>}\mp@subsup{\tau}{<}{<\nuy>}\mp@subsup{P}{}{\prime\prime}\mathrm{ by(rule ResChain)

```

```

        by(force intro: ResB)
    moreover from P}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ Chain have }<\nuy>\mp@subsup{P}{}{\prime\prime\prime}\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}<\nuy>\mp@subsup{P}{}{\prime}\mathrm{ by(rule ResChain)
    ultimately show <\nuy>P\Longrightarrow\Longrightarrowl}\mp@subsup{l}{l}{}a<\nux><<\nuy>\mp@subsup{P}{}{\prime}\mathrm{ by(rule transitionI)
    next
assume PTrans: P \Longrightarrow\Longrightarrowlu in P}\mp@subsup{P}{}{\prime\prime}->a<x>\prec\mp@subsup{P}{}{\prime
and yineqa: y}\not=
and yineqx: }y\not=
and yinequ: }y\not=
from PTrans obtain P}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ where PChain: P >}\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime\prime\prime
and }\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ Trans: }\mp@subsup{P}{}{\prime\prime\prime}\longmapstoa<x>\prec\mp@subsup{P}{}{\prime\prime
and }\mp@subsup{P}{}{\prime\prime}\mathrm{ Chain: }\mp@subsup{P}{}{\prime\prime}[x::=u]\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime
by(blast dest: transitionE)
from PChain have <\nuy>P \Longrightarrow>

```

```

    by(force intro: ResB)
    moreover have (<\nuy>\mp@subsup{P}{}{\prime\prime})[x::=u] \Longrightarrow>}\mp@subsup{\tau}{\tau}{<\nuy>
    proof -
    from P }\mp@subsup{P}{}{\prime\prime}\mathrm{ Chain have }<\nuy>(\mp@subsup{P}{}{\prime\prime}[x::=u])\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}<\nuy>\mp@subsup{P}{}{\prime}\mathrm{ by(rule ResChain)
    with yineqx yinequ show ?thesis by(simp add: eqvt-subs[THEN sym])
    qed
    ultimately show <\nuy>P\Longrightarrow\Longrightarrowlu in (<\nuy>\mp@subsup{P}{}{\prime\prime})->a<x><<<\nuy>\mp@subsup{P}{}{\prime}\mathrm{ by(rule tran-}\\mp@code{l}
    sitionI)
qed
lemma Bang:
fixes }P\mathrm{ :: pi
and Rs :: residual
and u :: name
and }\mp@subsup{P}{}{\prime\prime}:: p
and a :: name
and x :: name
and }\mp@subsup{P}{}{\prime}::p
shows }P|!P\Longrightarrow\mp@subsup{\Longrightarrow}{l}{}Rs\Longrightarrow!P\Longrightarrow\mp@subsup{\Longrightarrow}{l}{}R

```

```

proof -
assume P|!P\Longrightarrowl Rs
thus !P}\mp@subsup{\Longrightarrow}{l}{l}R
proof(nominal-induct avoiding: P rule: residual.strong-inducts)
case(BoundR a x P' P)
assume xFreshP: x\sharpP
assume PTrans: P|!P \Longrightarrow\Longrightarrowl}\mp@subsup{l}{l}{}a<x>\prec \prec P
from PTrans obtain a' where aeq: a = BoundOutputS a' by(cases a, auto)
with PTrans xFreshP obtain P}\mp@subsup{P}{}{\prime\prime}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ where PChain: P|!P <}\mp@subsup{~}{\tau}{}\mp@subsup{P}{}{\prime\prime

```

```

                                    and P}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ Chain: P}\mp@subsup{P}{}{\prime\prime\prime}\Longrightarrow\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime
    by(force dest: transitionE)
    ```
    show \(\left.!P \Longrightarrow_{l} a « x\right\rangle \prec P^{\prime}\)
    \(\operatorname{proof}\left(\right.\) cases \(\left.P^{\prime \prime}=P \|!P\right)\)
    assume \(P^{\prime \prime}=P \|!P\)
    moreover with \(P^{\prime \prime}\) Trans have \(!P \longmapsto a^{\prime}<\nu x>\prec P^{\prime \prime \prime}\) by (blast intro: transi-
tions.Bang)
            ultimately show ?thesis using PChain \(P^{\prime \prime \prime}\) Chain aeq by (simp, rule-tac
transitionI, auto)
    next
            assume \(P^{\prime \prime} \neq P \|!P\)
            with PChain have \(!P \Longrightarrow_{\tau} P^{\prime \prime}\) by(rule bangChain)
            with \(P^{\prime \prime}\) Trans \(P^{\prime \prime \prime}\) Chain aeq show ?thesis by (blast intro: transitionI)
    qed
next
    fix \(\alpha P^{\prime} P\)
    assume \(P \|!P \Longrightarrow_{l} \alpha \prec P^{\prime}\)
    then obtain \(P^{\prime \prime} P^{\prime \prime \prime}\) where PChain: \(P \|!P \Longrightarrow_{\tau} P^{\prime \prime}\)
                    and \(P^{\prime \prime}\) Trans: \(P^{\prime \prime} \longmapsto \alpha \prec P^{\prime \prime \prime}\)
                    and \(P^{\prime \prime \prime}\) Chain: \(P^{\prime \prime \prime} \Longrightarrow_{\tau} P^{\prime}\)
    by (force dest: transitionE)
    show \(!P \Longrightarrow_{l} \alpha \prec P^{\prime}\)
    proof \(\left(\right.\) cases \(\left.P^{\prime \prime}=P \|!P\right)\)
            assume \(P^{\prime \prime}=P \|!P\)
            moreover with \(P^{\prime \prime}\) Trans have \(!P \longmapsto \alpha \prec P^{\prime \prime \prime}\) by (blast intro: transi-
tions.Bang)
            ultimately show ?thesis using PChain \(P^{\prime \prime \prime}\) Chain by(rule-tac transitionI,
auto)
    next
            assume \(P^{\prime \prime} \neq P \|!P\)
            with PChain have \(!P \Longrightarrow_{\tau} P^{\prime \prime}\) by(rule bangChain)
            with \(P^{\prime \prime}\) Trans \(P^{\prime \prime \prime}\) Chain show ?thesis by(blast intro: transitionI)
    qed
qed
next
assume \(P \|!P \Longrightarrow_{l} u\) in \(P^{\prime \prime} \rightarrow a<x>\prec P^{\prime}\)
then obtain \(P^{\prime \prime \prime}\) where PChain: \(P \|!P \Longrightarrow{ }_{\tau} P^{\prime \prime \prime}\)
and \(P^{\prime \prime \prime}\) Trans: \(P^{\prime \prime \prime} \longmapsto a<x>\prec P^{\prime \prime}\)
and \(P^{\prime \prime}\) Chain: \(P^{\prime \prime}[x::=u] \Longrightarrow{ }_{\tau} P^{\prime}\)
by (force dest: transitionE)
show \(!P \Longrightarrow_{l} u\) in \(P^{\prime \prime} \rightarrow a<x>\prec P^{\prime}\)
\(\operatorname{proof}\left(\right.\) cases \(\left.P^{\prime \prime \prime}=P \|!P\right)\)
assume \(P^{\prime \prime \prime}=P \|!P\)
moreover with \(P^{\prime \prime \prime}\) Trans have \(!P \longmapsto a<x>\prec P^{\prime \prime}\) by (blast intro: transi-
tions.Bang)
ultimately show ?thesis using PChain \(P^{\prime \prime}\) Chain by (rule-tac transitionI, auto)
next
assume \(P^{\prime \prime \prime} \neq P \|!P\)
with PChain have \(!P \Longrightarrow_{\tau} P^{\prime \prime \prime}\) by (rule bangChain)
with \(P^{\prime \prime \prime}\) Trans \(P^{\prime \prime}\) Chain show ?thesis by(blast intro: transitionI)
qed
qed
lemma tauTransitionChain:
fixes \(P:: p i\)
and \(P^{\prime}:: p i\)
assumes \(P \Longrightarrow{ }_{l} \tau \prec P^{\prime}\)
```

    shows \(P \Longrightarrow_{\tau} P^{\prime}\)
    using assms
by (auto simp add: transition-def residualInject)
lemma chainTransitionAppend:
fixes $P$ :: $p i$
and $P^{\prime}:: p i$
and $R s$ :: residual
and $a$ :: name
and $x$ :: name
and $P^{\prime \prime}:: p i$
and $u$ :: name
and $P^{\prime \prime \prime}:: p i$
and $\alpha$ :: freeRes
shows $P \Longrightarrow_{\tau} P^{\prime} \Longrightarrow P^{\prime} \Longrightarrow_{l} R s \Longrightarrow P \Longrightarrow_{l} R s$
and $P \Longrightarrow_{\tau} P^{\prime \prime} \Longrightarrow P^{\prime \prime} \Longrightarrow_{l}$ u in $P^{\prime \prime \prime} \rightarrow a<x>\prec P^{\prime} \Longrightarrow P \Longrightarrow_{l}$ u in $P^{\prime \prime \prime} \rightarrow a<x>$
$\prec P^{\prime}$
and $P \Longrightarrow_{l} a<\nu x>\prec P^{\prime \prime} \Longrightarrow P^{\prime \prime} \Longrightarrow_{\tau} P^{\prime} \Longrightarrow x \sharp P \Longrightarrow P \Longrightarrow_{l} a<\nu x>\prec P^{\prime}$
and $P \Longrightarrow_{l}$ u in $P^{\prime \prime \prime} \rightarrow a<x>\prec P^{\prime \prime} \Longrightarrow P^{\prime \prime} \Longrightarrow_{\tau} P^{\prime} \Longrightarrow P \Longrightarrow_{l}$ u in $P^{\prime \prime \prime} \rightarrow a<x>$
$\prec P^{\prime}$

```
```

    and }P\mp@subsup{\Longrightarrow}{l}{}\alpha\prec\mp@subsup{P}{}{\prime\prime}\Longrightarrow\mp@subsup{P}{}{\prime\prime}\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime}\LongrightarrowP\Longrightarrow\mp@subsup{\Longrightarrow}{l}{}\alpha\prec\mp@subsup{P}{}{\prime
    proof -
assume }P\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime}\mathrm{ and }\mp@subsup{P}{}{\prime}\Longrightarrow\mp@subsup{\Longrightarrow}{l}{}\mathrm{ Rs
thus P\Longrightarrowl}\mp@subsup{l}{l}{Rs
by(auto simp add: transition-def residualInject) (blast dest:rtrancl-trans)+
next
assume P \Longrightarrow}\mp@subsup{\overbrace}{\tau}{}\mp@subsup{P}{}{\prime\prime}\mathrm{ and }\mp@subsup{P}{}{\prime\prime}\mp@subsup{\Longrightarrow}{l}{}u\mathrm{ u in }\mp@subsup{P}{}{\prime\prime\prime}->a<x>\prec < P
thus }P\mp@subsup{\Longrightarrow}{l}{}u\mathrm{ in }\mp@subsup{P}{}{\prime\prime\prime}->a<x>\prec \prec P'
apply(auto simp add: inputTransition-def residualInject)
by(blast dest: rtrancl-trans)+
next
assume PTrans: P\Longrightarrow\Longrightarrowl a<\nux>}\prec\mp@subsup{P}{}{\prime\prime
assume P '/Chain: P' }\mp@subsup{}{~}{\prime\prime}\mp@subsup{P}{\tau}{}\mp@subsup{P}{}{\prime
assume xFreshP: x\sharpP
from PTrans xFreshP obtain P}\mp@subsup{P}{}{\prime\prime\prime}\mp@subsup{P}{}{\prime\prime\prime\prime}\mathrm{ where PChain: P <}\mp@subsup{]}{\tau}{}\mp@subsup{P}{}{\prime\prime\prime
and }\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ Trans: }\mp@subsup{P}{}{\prime\prime\prime}\longmapstoa<\nux>\prec\mp@subsup{P}{}{\prime\prime\prime\prime
and P}\mp@subsup{P}{}{\prime\prime\prime\prime}\mathrm{ Chain: P}\mp@subsup{P}{}{\prime\prime\prime\prime}\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime\prime
by(blast dest: transitionE)
from P}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ Chain P ''Chain have P}\mp@subsup{P}{}{\prime\prime\prime\prime}\Longrightarrow\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime}\mathrm{ by auto
with PChain P}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ Trans show }P>\mp@subsup{\Longrightarrow}{l}{}a<\nux>< \mp@subsup{P}{}{\prime}\mathbf{by}(\mathrm{ rule transitionI)
next
assume PTrans: P\Longrightarrow}\mp@subsup{\}{l}{}u\mathrm{ in }\mp@subsup{P}{}{\prime\prime\prime}->a<x>\prec \prec '\prime
assume P ''Chain: }\mp@subsup{P}{}{\prime\prime}\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime
from PTrans obtain }\mp@subsup{P}{}{\prime\prime\prime\prime}\mathrm{ where PChain: P >}\mp@subsup{~}{\tau}{}\mp@subsup{P}{}{\prime\prime\prime\prime

```

```

        and }\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ Chain: }\mp@subsup{P}{}{\prime\prime\prime}[x::=u]\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime\prime
    by(blast dest: transitionE)
    from P}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ Chain P ''Chain have }\mp@subsup{P}{}{\prime\prime\prime}[x::=u]\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime}\mathrm{ by auto
    with PChain P}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ Trans show }P\mp@subsup{\Longrightarrow}{l}{}u\mathrm{ u in }\mp@subsup{P}{}{\prime\prime\prime}->a<x>\prec \prec P' by(blast intro
    transitionI)
next
assume PTrans: P \Longrightarrow}\mp@subsup{}{l}{}\alpha\prec\mp@subsup{P}{}{\prime\prime
assume P ''Chain: }\mp@subsup{P}{}{\prime\prime}\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime
from PTrans obtain P}\mp@subsup{P}{}{\prime\prime\prime}\mp@subsup{P}{}{\prime\prime\prime\prime}\mathrm{ where PChain: P }\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime\prime\prime
and P}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ Trans: }\mp@subsup{P}{}{\prime\prime\prime}\longmapsto\alpha\prec\mp@subsup{P}{}{\prime\prime\prime\prime
and P}\mp@subsup{P}{}{\prime\prime\prime}Chain: P'\prime\prime\prime \Longrightarrow\Longrightarrow\tau P P'\prime
by(blast dest: transitionE)
from P}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ Chain P ''Chain have P}\mp@subsup{P}{}{\prime\prime\prime\prime}\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime}\mathrm{ by auto
with PChain P}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ Trans show }P\mp@subsup{\Longrightarrow}{l}{}\alpha\prec\mp@subsup{P}{}{\prime}\mathrm{ by(rule transitionI)
qed
lemma freshInputTransition:
fixes P :: pi

```
```

    and a :: name
    and }x\mathrm{ :: name
    and u :: name
    and }\mp@subsup{P}{}{\prime\prime}:: p
    and }\mp@subsup{P}{}{\prime}:: p
    and c :: name
    ```

```

    and cFreshP:c\sharpP
    and cinequ: c\not=u
    shows c\sharp P
    proof -
from PTrans obtain P}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ where PChain: P >}\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime\prime\prime

```

```

                and }\mp@subsup{P}{}{\prime\prime}\mathrm{ Chain: }\mp@subsup{P}{}{\prime\prime}[x::=u]\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime
    by(blast dest: transitionE)
    from PChain cFreshP have cFresh\mp@subsup{P}{}{\prime\prime\prime}:c\sharp\mp@subsup{P}{}{\prime\prime\prime}}\mathbf{by}(rule freshChain
    show c\sharp\mp@subsup{P}{}{\prime}
    proof(cases x=c)
        assume xeqc: x=c
        from cinequ have c\sharp\mp@subsup{P}{}{\prime\prime}[c::=u] apply - by(rule fresh-fact2)
        with P''Chain xeqc show ?thesis by(force intro: freshChain)
    next
    assume xineqc: }x\not=
    with P}\mp@subsup{P}{}{\prime\prime\prime}Trans cFresh\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ have c # P'/ by(blast dest: freshBoundDerivative)
    with cinequ have c}\sharp\mp@subsup{P}{}{\prime\prime}[x::=u
            apply -
            apply(rule fresh-fact1)
            by simp
    with P''Chain show ?thesis by(rule freshChain)
    qed
    qed
lemma freshBoundOutputTransition:
fixes P :: pi
and a :: name
and }x\mathrm{ :: name
and }\mp@subsup{P}{}{\prime}:: p
and c :: name
assumes PTrans: P \Longrightarrow>l }a<\nux>\prec\mp@subsup{P}{}{\prime
and cFreshP:c\sharpP
and cineqx: c\not=x
shows c\sharp P'
proof -

```
```

    have Goal: \(\bigwedge P\) a \(x P^{\prime} c . \llbracket P \Longrightarrow_{l} a<\nu x>\prec P^{\prime} ; x \sharp P ; c \sharp P ; c \neq x \rrbracket \Longrightarrow c \sharp P^{\prime}\)
    proof -
    fix \(P a x P^{\prime} c\)
    assume PTrans: \(P \Longrightarrow_{l} a<\nu x>\prec P^{\prime}\)
    assume \(x\) Fresh \(P: x \sharp P\)
    assume \(c\) Fresh \(P\) : \((c::\) name \() \sharp P\)
    assume cineqx: \(c \neq x\)
    from PTrans xFreshP obtain \(P^{\prime \prime} P^{\prime \prime \prime}\) where PTrans: \(P \Longrightarrow{ }_{\tau} P^{\prime \prime}\)
                                    and \(P^{\prime \prime}\) Trans: \(P^{\prime \prime} \longmapsto a<\nu x>\prec P^{\prime \prime \prime}\)
                                    and \(P^{\prime \prime \prime}\) Trans: \(P^{\prime \prime \prime} \Longrightarrow_{\tau} P^{\prime}\)
        by (blast dest: transitionE)
    from PTrans cFreshP have \(c \sharp P^{\prime \prime}\) by (rule freshChain)
    with \(P^{\prime \prime}\) Trans cineqx have \(c \sharp P^{\prime \prime \prime}\) by (blast dest: Late-Semantics.freshBoundDerivative)
    with \(P^{\prime \prime \prime}\) Trans show \(c \sharp P^{\prime}\) by(rule freshChain)
    qed
    have \(\exists d::\) name. \(d \sharp\left(P, P^{\prime}, c\right)\) by (blast intro: name-exists-fresh)
    then obtain \(d::\) name where \(d F r e s h P: d \sharp P\) and \(d F r e s h P^{\prime}: d \sharp P^{\prime}\) and cineqd:
    $c \neq d$
by(force simp add: fresh-prod)
from PTrans dFresh $P^{\prime}$ have $P \Longrightarrow_{l} a<\nu d>\prec\left([(x, d)] \cdot P^{\prime}\right)$ by $(\operatorname{simp}$ add: al-
phaBoundResidual)
hence $c \sharp[(x, d)] \cdot P^{\prime}$ using $d F r e s h P$ cFreshP cineqd by (rule Goal)
with cineqd cineqx show ?thesis by (simp add: name-fresh-left name-calc)
qed
lemma freshTauTransition:
fixes $P$ :: $p i$
and $c::$ name
assumes PTrans: $P \Longrightarrow_{l} \tau \prec P^{\prime}$
and cFreshP: $c \sharp P$
shows $c \sharp P^{\prime}$
proof -
from PTrans have $P \Longrightarrow_{\tau} P^{\prime}$ by(rule tauTransitionChain)
thus ?thesis using cFreshP by(rule freshChain)
qed
lemma freshOutputTransition:
fixes $P$ :: $p i$
and $a$ :: name
and $b$ :: name
and $P^{\prime}:: p i$
and $c$ :: name

```
```

    assumes PTrans: P\Longrightarrow\Longrightarrowla }a[b]\prec\mp@subsup{P}{}{\prime
    and cFreshP:c\sharpP
    shows c\sharp P'
    proof -
from PTrans obtain P'\prime P'\prime\prime where PTrans: P \Longrightarrow>}\mp@subsup{~}{\tau}{}\mp@subsup{P}{}{\prime\prime
and P}\mp@subsup{P}{}{\prime\prime}\mathrm{ Trans: }\mp@subsup{P}{}{\prime\prime}\longmapstoa[b]\prec\mp@subsup{P}{}{\prime\prime\prime
and P}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ Trans: P}\mp@subsup{P}{}{\prime\prime\prime}\Longrightarrow\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime
by(blast dest: transitionE)
from PTrans cFreshP have c\sharp P'\prime by(rule freshChain)
with P}\mp@subsup{P}{}{\prime\prime}\mathrm{ Trans have c\# P'/' by(blast dest: Late-Semantics.freshFreeDerivative)
with P}\mp@subsup{P}{}{\prime\prime}\mathrm{ Trans show ?thesis by(rule freshChain)
qed
lemma eqvtI:
fixes P :: pi
and Rs :: residual
and perm :: name prm
and u :: name
and }\mp@subsup{P}{}{\prime\prime}::p
and a :: name
and }x\mathrm{ :: name
and }\mp@subsup{P}{}{\prime}::p
shows P\Longrightarrowl}\mp@subsup{\Longrightarrow}{l}{}Rs\Longrightarrow(\mathrm{ perm • P) >}\mp@subsup{l}{l}{}(\mathrm{ perm }\cdotRs
and }P\mp@subsup{\Longrightarrow}{l}{}u\mathrm{ in P ''}->a<x>\prec\mp@subsup{P}{}{\prime}\Longrightarrow(\mathrm{ perm • P) >>l (perm • u) in (perm •
P'\prime})->(\mathrm{ perm • a )< (perm • x )> 々 (perm • P')
proof -
assume P\Longrightarrowl Rs
thus (perm • P) \Longrightarrowl}(\mathrm{ perm • Rs)
proof(nominal-induct Rs avoiding: P rule: residual.strong-inducts)
case(BoundR a x P'P)
have PTrans: P\Longrightarrow\Longrightarrowla<x>}\prec\mp@subsup{P}{}{\prime}\mathrm{ by fact
moreover then obtain b}\mathrm{ where aeqb: a = BoundOutputS b by(cases a, auto)
moreover have }x\sharpP\mathrm{ by fact
ultimately obtain P}\mp@subsup{P}{}{\prime\prime}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ where PTrans: P >}\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime\prime
and }\mp@subsup{P}{}{\prime\prime}\mathrm{ Trans: }\mp@subsup{P}{}{\prime\prime}\longmapstob<\nux>\prec\mp@subsup{P}{}{\prime\prime\prime
and P}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ Trans: }\mp@subsup{P}{}{\prime\prime\prime}\Longrightarrow\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime
by(blast dest: transitionE)
from PTrans have (perm • P) \Longrightarrow>
moreover from P P'Trans have (perm • P'\prime)\longmapsto(perm • (b<\nux><< P'\prime\prime))
by(rule eqvts)
moreover from P}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ Trans have (perm • P''') >}\mp@subsup{~}{\tau}{}(\mathrm{ perm • P') by(rule
eqvtChainI)
ultimately show ?case using aeqb by(force intro: transitionI)
next
case(FreeR \alpha P' P)

```
```

    have P\Longrightarrow\Longrightarrowl}\mp@subsup{l}{l}{}\prec\mp@subsup{P}{}{\prime}\mathrm{ by fact
    then obtain }\mp@subsup{P}{}{\prime\prime}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ where PTrans: }P\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime\prime
                                    and }\mp@subsup{P}{}{\prime\prime}\mathrm{ Trans: }\mp@subsup{P}{}{\prime\prime}\longmapsto\alpha\prec\mp@subsup{P}{}{\prime\prime\prime
                                    and P}\mp@subsup{P}{}{\prime\prime}\mathrm{ Trans: }\mp@subsup{P}{}{\prime\prime\prime}\Longrightarrow\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime
    by(blast dest: transitionE)
    from PTrans have (perm •P) \Longrightarrow>
    moreover from }\mp@subsup{P}{}{\prime\prime}\mathrm{ Trans have (perm • P'')}\longmapsto(\mathrm{ perm • ( }\alpha\prec\mp@subsup{P}{}{\prime\prime\prime})
    by(rule eqvts)
    moreover from P }\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ Trans have (perm • P'/') >>
    eqvtChainI)
ultimately show ?case by(force intro: transitionI)
qed
next
assume P\Longrightarrow}\mp@subsup{\Longrightarrow}{l}{}u\mathrm{ in }\mp@subsup{P}{}{\prime\prime}->a<x>\prec\prec\mp@subsup{P}{}{\prime
then obtain P}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ where PChain: P >>}\mp@subsup{\tau}{\tau}{}\mp@subsup{P}{}{\prime\prime\prime
and }\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ Trans: }\mp@subsup{P}{}{\prime\prime\prime}\longmapstoa<x>\prec\mp@subsup{P}{}{\prime\prime
and }\mp@subsup{P}{}{\prime\prime}\mathrm{ Chain: }\mp@subsup{P}{}{\prime\prime}[x::=u]\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime
by(blast dest: transitionE)
from PChain have (perm • P) \Longrightarrow>

```

```

        by(rule eqvts)
    moreover from }\mp@subsup{P}{}{\prime\prime}\mathrm{ Chain have (perm • P'}[x::=u]) \Longrightarrow> (perm \cdot P') by(rule
    eqvtChainI)
ultimately show (perm • P) \Longrightarrow\Longrightarrowl (perm •u) in (perm • P'\)->(perm •a)<(perm

- x)>\prec (perm • P')
by(force intro: transitionI simp add: eqvt-subs[THEN sym] perm-bij)
qed
lemmas freshTransition = freshBoundOutputTransition freshOutputTransition
freshInputTransition freshTauTransition
end
theory Weak-Late-Semantics
imports Weak-Late-Step-Semantics
begin
definition weakTransition :: (pi }\times\mathrm{ residual) set
where weakTransition \equivWeak-Late-Step-Semantics.transition }\cup{x.\existsP.x
(P,\tau\precP)}
abbreviation weakLateTransition-judge $::$ pi $\Rightarrow$ residual $\Rightarrow$ bool $(-\Longrightarrow \hat{l}-[80,80]$ 80)

```
```

    where \(P \Longrightarrow{ }_{l}{ }^{\wedge} R s \equiv(P, R s) \in\) weakTransition
    ```
```

    where \(P \Longrightarrow{ }_{l}{ }^{\wedge} R s \equiv(P, R s) \in\) weakTransition
    ```
```

lemma transitionI:
fixes P :: pi
and Rs :: residual
and }\mp@subsup{P}{}{\prime}:: p
shows }P\Longrightarrow\mp@subsup{}{l}{}Rs\LongrightarrowP\Longrightarrow\mp@subsup{}{l}{\}R
and }P\Longrightarrow\hat{l}\tau\prec
proof -
assume P\Longrightarrow}\mp@subsup{\Longrightarrow}{l}{}R
thus P\Longrightarrow\Longrightarrow``Rs}\mathbf{by(simp add:weakTransition-def)
next
show P\Longrightarrow\^`}\tau\precP\mathrm{ by(simp add: weakTransition-def) qed lemma transitionCases[consumes 1, case-names Step Stay]:     fixes P :: pi     and Rs :: residual     and }\mp@subsup{P}{}{\prime}::p     assumes P\Longrightarrow^^ Rs and }P\Longrightarrowl Rs\LongrightarrowFR and }\quadRs=\tau\precP\LongrightarrowF(\tau\precP     shows F Rs using assms by(auto simp add: weakTransition-def) lemma singleActionChain:     fixes P :: pi     and }\alpha\mathrm{ :: freeRes     and }\mp@subsup{P}{}{\prime}::p     assumes }P\longmapsto\alpha\prec\mp@subsup{P}{}{\prime     shows }P\Longrightarrow\mp@subsup{l}{}{\wedge}(\alpha\prec\mp@subsup{P}{}{\prime} using assms by(auto intro: Weak-Late-Step-Semantics.singleActionChain     simp add: weakTransition-def) lemma Tau:     fixes P :: pi     shows }\tau.(P)\Longrightarrow\mp@subsup{\Longrightarrow}{l}{`}\tau\prec
by(auto intro:Weak-Late-Step-Semantics.Tau
simp add: weakTransition-def)
lemma Output:
fixes a :: name
and b:: name

```
```

and }P::p
shows a{b}.P\Longrightarrow\hat{l}a[b]\precP
by(auto intro:Weak-Late-Step-Semantics.Output
simp add: weakTransition-def)
lemma Match:
fixes a :: name
and }P::p
and b :: name
and }x\mathrm{ :: name
and }\mp@subsup{P}{}{\prime}:::p
and \alpha :: freeRes

```

```

and }P\Longrightarrow\mp@subsup{\Longrightarrow}{l}{\hat{\prime}}\alpha\prec\mp@subsup{P}{}{\prime}\LongrightarrowP\not=\mp@subsup{P}{}{\prime}\Longrightarrow[a\frowna]P\Longrightarrow\mp@subsup{\Longrightarrow}{l}{\hat{*}}\alpha\prec\mp@subsup{P}{}{\prime
by(auto simp add: residual.inject weakTransition-def intro: Weak-Late-Step-Semantics.Match)
lemma Mismatch:
fixes a :: name
and c :: name
and }P::p
and b :: name
and }x\mathrm{ :: name
and }\mp@subsup{P}{}{\prime}::p
and \alpha :: freeRes

```

```

by(auto simp add:residual.inject weakTransition-def intro:Weak-Late-Step-Semantics.Mismatch)
lemma Open:
fixes P :: pi
and a :: name
and b :: name
and }\mp@subsup{P}{}{\prime}::p
assumes Trans: P\Longrightarrow``}\=[b]\prec\mp@subsup{P}{}{\prime
and aInEqb: a\not=b
shows }<\nub>P\Longrightarrow\mp@subsup{|}{}{`}a<\nub>\prec\mp@subsup{P}{}{\prime
using assms
by(auto intro:Weak-Late-Step-Semantics.Open
simp add: weakTransition-def residual.inject)
lemma Par1B:
fixes P :: pi
and a :: name
and x :: name

```
```

    and }\mp@subsup{P}{}{\prime}:: p
    assumes PTrans: P\Longrightarrow齐 a<\nux>}\prec\mp@subsup{P}{}{\prime
    and xFreshQ: }x\sharp
    shows }P|Q\Longrightarrow```\mp@code{a<\nux> \prec( (P'|Q)
    using assms
by(auto intro:Weak-Late-Step-Semantics.Par1B
simp add: weakTransition-def residual.inject)
lemma Par1F:
fixes P :: pi
and }\alpha\mathrm{ :: freeRes
and }\mp@subsup{P}{}{\prime}::p

```

```

    shows }P|Q\Longrightarrow\hat{l}|\mp@code{\imath}(\mp@subsup{P}{}{\prime}|Q
    using assms
by(auto intro:Weak-Late-Step-Semantics.Par1F
simp add: weakTransition-def residual.inject)
lemma Par2B:
fixes }Q::p
and a :: name
and }x\mathrm{ :: name
and }\mp@subsup{Q}{}{\prime}::p
assumes QTrans: Q \Longrightarrow`la}a<\nux>\prec\prec\mp@subsup{Q}{}{\prime     and xFreshP: x\sharpP     shows }P|Q\Longrightarrow^^^`| a<\nux> \prec(P| Q
using assms
by(auto intro:Weak-Late-Step-Semantics.Par2B
simp add: weakTransition-def residual.inject)
lemma Par2F:
fixes }Q:: p
and }\alpha\mathrm{ :: freeRes
and }\mp@subsup{Q}{}{\prime}:: p
assumes QTrans: Q \Longrightarrow ^^ }\alpha\prec\mp@subsup{Q}{}{\prime
shows }P|Q\Longrightarrow\mp@subsup{}{l}{}\hat{`}\alpha\prec(P|\mp@subsup{Q}{}{\prime}
using assms
by(auto intro:Weak-Late-Step-Semantics.Par2F
simp add: weakTransition-def residual.inject)
lemma Comm1:

```
```

    fixes P :: pi
    and a :: name
    and b :: name
    and }\mp@subsup{P}{}{\prime\prime}::p
    and }\mp@subsup{P}{}{\prime}::p
    and }Q ::p
    and }\mp@subsup{Q}{}{\prime}::p
    assumes PTrans: }P\mp@subsup{\Longrightarrow}{l}{}b\mathrm{ in }\mp@subsup{P}{}{\prime\prime}->a<x>\prec \prec P
    and QTrans: Q \Longrightarrowl }\mp@subsup{l}{l}{}a[b]\prec\mp@subsup{Q}{}{\prime
    shows }P||\Longrightarrow\mp@subsup{}{l}{`}\tau\prec\mp@subsup{P}{}{\prime}|\mp@subsup{Q}{}{\prime
    using assms
by(auto intro:Weak-Late-Step-Semantics.Comm1
simp add: weakTransition-def residual.inject)
lemma Comm2:
fixes }P::p
and a :: name
and b :: name
and }\mp@subsup{Q}{}{\prime\prime}:: p
and }\mp@subsup{P}{}{\prime}::p
and }Q ::p
and }\mp@subsup{Q}{}{\prime}:: p

```

```

    and QTrans:Q \Longrightarrow}\mp@subsup{l}{l}{}b\mathrm{ in }\mp@subsup{Q}{}{\prime\prime}->a<x>\prec\prec\mp@subsup{Q}{}{\prime
    shows }P||\Longrightarrow\hat{l}|\prec\mp@subsup{P}{}{\prime}|\mp@subsup{Q}{}{\prime
    using assms
by(auto intro:Weak-Late-Step-Semantics.Comm2
simp add: weakTransition-def residual.inject)
lemma Close1:
fixes P :: pi
and y :: name
and }\mp@subsup{P}{}{\prime\prime}::p
and a :: name
and }x\mathrm{ :: name
and }\mp@subsup{P}{}{\prime}:: p
and }Q :: p
and }\mp@subsup{Q}{}{\prime}::p

```

```

and QTrans:Q \Longrightarrow^^
and xFreshP:y\sharpP
and xFreshQ: y\sharpQ
shows }P||\Longrightarrow\mp@subsup{}{l}{`}\tau\prec<\nuy>(\mp@subsup{P}{}{\prime}|\mp@subsup{Q}{}{\prime}

```
```

using assms
by (auto intro: Weak-Late-Step-Semantics.Close1
simp add: weakTransition-def residual.inject)
lemma Close2:
fixes $P$ :: $p i$
and $a$ :: name
and $x$ :: name
and $P^{\prime}:: p i$
and $\quad Q$ :: pi
and $y$ :: name
and $Q^{\prime \prime}:: p i$
and $\quad Q^{\prime}:: p i$
assumes PTrans: $P \Longrightarrow \hat{l} a<\nu y>\prec P^{\prime}$
and $\quad$ QTrans: $Q \Longrightarrow{ }_{l} y$ in $Q^{\prime \prime} \rightarrow a<x>\prec Q^{\prime}$
and $\quad x$ Fresh $P: y \sharp P$
and $\quad x$ Fresh $Q: y \sharp Q$
shows $P \| Q \Longrightarrow{ }^{\prime} \hat{}^{\prime} \tau \prec<\nu y>\left(P^{\prime} \| Q^{\prime}\right)$
using assms
by (auto intro: Weak-Late-Step-Semantics.Close2
simp add: weakTransition-def residual.inject)
lemma ResF:
fixes $P$ :: $p i$
and $\alpha$ :: freeRes
and $P^{\prime}:: p i$
and $x$ :: name
assumes PTrans: $P \Longrightarrow \hat{l} \alpha \prec P^{\prime}$
and $\quad x F r e s h A l p h a: ~ x \sharp \alpha$
shows $<\nu x>P \Longrightarrow{ }_{l} \alpha \prec<\nu x>P^{\prime}$
using assms
by (auto intro: Weak-Late-Step-Semantics.ResF
simp add: weakTransition-def residual.inject)
lemma ResB:
fixes $P$ :: $p i$
and $a$ :: name
and $x$ :: name
and $\quad P^{\prime}:: p i$
and $y$ :: name
assumes PTrans: $P \Longrightarrow{ }_{l} \dot{a} a<\nu x>\prec P^{\prime}$
and $\quad y i n e q a: ~ y \neq a$
and yineqx: $y \neq x$
and $\quad x$ Fresh $P: x \sharp P$

```
```

    shows <\nuy>P\Longrightarrow\^a}a<\nux>< (<\nuy>\mp@subsup{P}{}{\prime}
    using assms
by(auto intro:Weak-Late-Step-Semantics.ResB
simp add: weakTransition-def residual.inject)
lemma Bang:
fixes }P::p
and Rs :: residual
assumes }P|!P\Longrightarrow\mp@subsup{}{l}{`}R     and }\quadRs\not=\tau\precP|!     shows !P\Longrightarrow` Rs
using assms
by(auto intro:Weak-Late-Step-Semantics.Bang
simp add: weakTransition-def residual.inject)
lemma tauTransitionChain:
fixes P :: pi
and }\mp@subsup{P}{}{\prime}::p
assumes }P\Longrightarrow\mp@subsup{|}{}{`}\tau\prec\mp@subsup{P}{}{\prime
shows }P\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime
using assms
by(auto intro:Weak-Late-Step-Semantics.tauTransitionChain
simp add: weakTransition-def residual.inject transition-def)
lemma chainTransitionAppend:
fixes P :: pi
and }\mp@subsup{P}{}{\prime}::p
and Rs :: residual
and a :: name
and }x\mathrm{ :: name
and }\mp@subsup{P}{}{\prime\prime}::p
and \alpha :: freeRes

```


```

P'
and }P\Longrightarrow\mp@subsup{\Longrightarrow}{l}{\imath}\alpha\prec\mp@subsup{P}{}{\prime\prime}\Longrightarrow\mp@subsup{P}{}{\prime\prime}\Longrightarrow\mp@subsup{|}{\tau}{}\mp@subsup{P}{}{\prime}\LongrightarrowP\Longrightarrow\mp@subsup{\}{\imath}{\hat{l}}\alpha\prec\mp@subsup{P}{}{\prime
proof -
assume P\Longrightarrow}\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime}\mathrm{ and }\mp@subsup{P}{}{\prime}\Longrightarrow\mp@subsup{}{l}{`}R
thus }P\Longrightarrow\^^\ R
by(auto intro:Weak-Late-Step-Semantics.chainTransitionAppend
Weak-Late-Step-Semantics.tauActionChain
simp add: weakTransition-def residual.inject)
next

```
```

assume $P \Longrightarrow \hat{\imath} \dot{a} a<\nu x>\prec P^{\prime \prime}$ and $P^{\prime \prime} \Longrightarrow_{\tau} P^{\prime}$ and $x \sharp P$
thus $P \Longrightarrow \hat{l} a<\nu x>\prec P^{\prime}$
by (auto intro: Weak-Late-Step-Semantics.chainTransitionAppend
simp add: weakTransition-def residual.inject)
next
assume $P \Longrightarrow \hat{\imath} \alpha \prec P^{\prime \prime}$ and $P^{\prime \prime} \Longrightarrow_{\tau} P^{\prime}$
thus $P \Longrightarrow \hat{\imath} \alpha \prec P^{\prime}$
$\operatorname{apply}\left(\right.$ case-tac $\left.P^{\prime \prime}=P^{\prime}\right)$
by(auto dest: Weak-Late-Step-Semantics.chainTransitionAppend
Weak-Late-Step-Semantics.tauActionChain
simp add: weakTransition-def residual.inject)
qed
lemma weakEqWeakTransitionAppend:
fixes $P$ :: $p i$
and $\quad P^{\prime}:: p i$
and $\alpha$ :: freeRes
and $P^{\prime \prime}:: p i$
assumes PTrans: $P \Longrightarrow_{l} \tau \prec P^{\prime}$
and $P^{\prime}$ Trans: $P^{\prime} \Longrightarrow \hat{l} \alpha \prec P^{\prime \prime}$
shows $P \Longrightarrow_{l} \alpha \prec P^{\prime \prime}$
$\operatorname{proof}($ cases $\alpha=\tau$ )
assume alphaEqTau: $\alpha=\tau$
with $P^{\prime}$ Trans have $P^{\prime} \Longrightarrow_{\tau} P^{\prime \prime}$ by(blast intro: tauTransitionChain)
with PTrans alphaEqTau show ?thesis
by (blast intro: Weak-Late-Step-Semantics.chainTransitionAppend)
next
assume alphaIneqTau: $\alpha \neq \tau$
from PTrans have $P \Longrightarrow_{\tau} P^{\prime}$ by (rule Weak-Late-Step-Semantics.tauTransitionChain)
moreover from $P^{\prime}$ Trans alphaIneqTau have $P^{\prime} \Longrightarrow_{l} \alpha \prec P^{\prime \prime}$
by (auto simp add: weakTransition-def residual.inject)
ultimately show ?thesis
by (rule Weak-Late-Step-Semantics.chainTransitionAppend)
qed
lemma freshBoundOutputTransition:
fixes $P$ :: $p i$
and $a$ :: name
and $x$ :: name
and $P^{\prime}:: p i$
and $c$ :: name
assumes PTrans: $P \Longrightarrow \hat{\imath} a<\nu x>\prec P^{\prime}$
and cFreshP: $c \sharp P$
and cineqx: $c \neq x$
shows $c \sharp P^{\prime}$

```
```

using assms
by(auto intro:Weak-Late-Step-Semantics.freshBoundOutputTransition
simp add: weakTransition-def residual.inject)
lemma freshTauTransition:
fixes P :: pi
and c:: name
assumes PTrans: P\Longrightarrow闾\tau\prec\mp@subsup{P}{}{\prime}
and cFreshP:c\sharpP
shows c\sharp P'
using assms
by(auto intro:Weak-Late-Step-Semantics.freshTauTransition
simp add: weakTransition-def residual.inject)
lemma freshOutputTransition:
fixes }P::p
and a :: name
and }b\mathrm{ :: name
and }\mp@subsup{P}{}{\prime}::p
and c :: name
assumes PTrans: P\Longrightarrow紋 a b] \prec-P'
and cFreshP:c\sharpP
shows c\sharp P'
using assms
by(auto intro:Weak-Late-Step-Semantics.freshOutputTransition
simp add: weakTransition-def residual.inject)
lemma eqvtI:
fixes P :: pi
and Rs :: residual
and perm :: name prm
assumes }P\Longrightarrow^^^|
shows (perm \cdot P)\Longrightarrow\^
using assms
by(auto intro:Weak-Late-Step-Semantics.eqvtI
simp add: weakTransition-def residual.inject)
lemma freshInputTransition:
fixes }P::p
and a :: name
and b :: name
and }\mp@subsup{P}{}{\prime}:: p
and c :: name

```
assumes PTrans: \(P \Longrightarrow{ }_{l}{ }^{\wedge} a<b>\prec P^{\prime}\)
and cFreshP: \(c \sharp P\)
and cineqb: \(c \neq b\)
shows \(c \sharp P^{\prime}\)
using assms
by (auto intro: Weak-Late-Step-Semantics.freshInputTransition
simp add: weakTransition-def residual.inject)
lemmas freshTransition \(=\) freshBoundOutputTransition freshOutputTransition freshInputTransition freshTauTransition
end
theory Weak-Late-Sim
imports Weak-Late-Semantics Strong-Late-Sim
begin
definition weakSimAct \(::\) pi \(\Rightarrow\) residual \(\Rightarrow\left({ }^{\prime} a:: f s\right.\)-name \() \Rightarrow(p i \times p i)\) set \(\Rightarrow\) bool where
weakSimAct PRs CRel \(\equiv\left(\forall Q^{\prime}\right.\) a x. Rs \(=a<\nu x>\prec Q^{\prime} \longrightarrow x \sharp C \longrightarrow\left(\exists P^{\prime}\right.\). \(P \Longrightarrow \hat{l}^{\hat{l}} a<\nu x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in\) Rel \(\left.)\right) \wedge\)
\(\left(\forall Q^{\prime} a x . R s=a<x>\prec Q^{\prime} \longrightarrow x \sharp C \longrightarrow\left(\exists P^{\prime \prime} . \forall u . \exists P^{\prime} . P\right.\right.\)
\(\Longrightarrow_{l} u\) in \(\left.\left.P^{\prime \prime} \rightarrow a<x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}[x::=u]\right) \in \operatorname{Rel}\right)\right) \wedge\)
\(\left(\forall Q^{\prime} \alpha . R s=\alpha \prec Q^{\prime} \longrightarrow\left(\exists P^{\prime} . P \Longrightarrow \hat{l} \alpha \prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in\right.\right.\) Rel))
definition weakSimAux :: pi \(\Rightarrow(p i \times p i)\) set \(\Rightarrow p i \Rightarrow\) bool where
weakSimAux PRel \(Q \equiv\left(\forall Q^{\prime}\right.\) a \(x .\left(Q \longmapsto a<\nu x>\prec Q^{\prime} \wedge x \sharp P\right) \longrightarrow\left(\exists P^{\prime} . P\right.\) \(\Longrightarrow \hat{l}{ }^{\wedge} a<\nu x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in\) Rel \(\left.)\right) \wedge\)
\(\left(\forall Q^{\prime}\right.\) a x. \(\left(Q \longmapsto a<x>\prec Q^{\prime} \wedge x \sharp P\right) \longrightarrow\left(\exists P^{\prime \prime} . \forall u . \exists P^{\prime}\right.\).
\(P \Longrightarrow_{l} u\) in \(P^{\prime \prime} \rightarrow a<x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}[x::=u]\right) \in\) Rel \(\left.)\right) \wedge\)
\(\left(\forall Q^{\prime} \alpha . Q \longmapsto \alpha \prec Q^{\prime} \longrightarrow\left(\exists P^{\prime} . P \Longrightarrow{ }_{\imath} \wedge \alpha \prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right)\right.\right.\)
\(\in R e l))\)
definition weakSimulation \(:: ~ p i \Rightarrow(p i \times p i)\) set \(\Rightarrow\) pi \(\Rightarrow\) bool \((-\rightsquigarrow \wedge<->-[80\), 80, 80] 80) where
\(P \rightsquigarrow<\) Rel \(>Q \equiv(\forall\) Rs. \(Q \longmapsto R s \longrightarrow\) weakSimAct P Rs P Rel \()\)
lemmas \(\operatorname{simDef}=\) weakSimAct-def weakSimulation-def
lemma weakSimAux P Rel \(Q=\) weakSimulation \(P\) Rel \(Q\)
by (auto simp add: weakSimAux-def simDef)

\section*{lemma monotonic:}
fixes \(A::(p i \times p i)\) set
and \(B::(p i \times p i)\) set
and \(P:: p i\)
```

    and }\mp@subsup{P}{}{\prime}:: p
    assumes P\rightsquigarrow^}<A>\mp@subsup{P}{}{\prime
    and }A\subseteq
    shows }P\rightsquigarrow^<B>\mp@subsup{P}{}{\prime
    using assms
apply(auto simp add: simDef)
apply blast
apply(erule-tac x=a<x>\prec\prec Q' in allE)
apply(clarsimp)
apply(rotate-tac 4)
apply(erule-tac x=\mp@subsup{Q}{}{\prime}}\mathrm{ in allE)
apply(erule-tac x=a in allE)
apply(erule-tac x=x in allE)
by blast+
lemma simCasesCont[consumes 1, case-names Bound Input Free]:
fixes P :: pi
and }Q :: p
and Rel :: (pi\timespi) set
and C ::' 'a::fs-name
assumes Eqvt: eqvt Rel

```

```

\prec P'^( (P', Q') \inRel

```

```

in }\mp@subsup{P}{}{\prime\prime}->a<x>\prec\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime}[x::=u])\in\mathrm{ Rel

```

```

Rel)
shows P\leadsto^<Rel>Q
using Free
proof(auto simp add: simDef)
fix }\mp@subsup{Q}{}{\prime}a
assume xFreshP:(x::name) \#P
assume Trans: Q\longmapstoa<\nux>}\prec\mp@subsup{Q}{}{\prime
have \existsc::name.c }\sharp(P,\mp@subsup{Q}{}{\prime},x,C) by(blast intro: name-exists-fresh
then obtain c::name where cFreshP:c\sharpP and cFresh }\mp@subsup{Q}{}{\prime}:c\sharp\mp@subsup{Q}{}{\prime}\mathrm{ and cFreshC:
c\sharpC
and cineqx: c\not=x
by(force simp add: fresh-prod)
from Trans cFreshQ' have }Q\longmapstoa<\nuc> \prec([(x,c)] \cdot Q') by(simp add: al-
phaBoundResidual)
with cFreshC have }\exists\mp@subsup{P}{}{\prime}.P\Longrightarrow\^` a<\nuc>< < P'^( (P',[(x,c)] \cdot Q')\in Re
by(rule Bound)
then obtain P' where PTrans: P\Longrightarrow\ \

- }\mp@subsup{Q}{}{\prime})\inRe

```
by blast
from PTrans xFreshP cineqx have \(x\) Fresh \(P^{\prime}: x \sharp P^{\prime} \mathbf{b y}\) (force dest: freshTransition)
with PTrans have \(P \Longrightarrow \Longrightarrow_{l}^{\wedge} a<\nu x>\prec\left([(x, c)] \cdot P^{\prime}\right)\) by (simp add: alphaBoundResidual name-swap)
moreover have \(\left([(x, c)] \cdot P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}\) (is ?goal)
proof -
from Eqvt \(P^{\prime} \operatorname{Rel} Q^{\prime}\) have \(\left([(x, c)] \cdot P^{\prime},[(x, c)] \cdot[(x, c)] \cdot Q^{\prime}\right) \in \operatorname{Rel}\)
by(rule eqvtRelI)
with cineqx show ?goal by (simp add: name-calc)
qed
ultimately show \(\exists P^{\prime} . P \Longrightarrow{ }_{l}{ }^{\wedge} a<\nu x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in\) Rel by blast

\section*{next}
fix \(Q^{\prime}\) a \(x u\)
assume \(Q\) Trans: \(Q \longmapsto a<x>\prec\left(Q^{\prime}:: p i\right)\) and \(x\) Fresh \(P: x \sharp P\)
have \(\exists c::\) name. \(c \sharp\left(P, Q^{\prime}, C, x\right) \mathbf{b y}(\) blast intro: name-exists-fresh)
then obtain \(c:: n a m e\) where cFresh \(P: c \sharp P\) and cFresh \(Q^{\prime}: c \sharp Q^{\prime}\) and \(c F r e s h C\) : \(c \sharp C\)
and cineqx: \(c \neq x\)
by (force simp add: fresh-prod)
from \(Q\) Trans cFresh \(Q^{\prime}\) have \(Q \longmapsto a<c>\prec\left([(x, c)] \cdot Q^{\prime}\right) \mathbf{b y}(\) simp add: alphaBoundResidual)
with \(c\) Fresh \(C\) have \(\exists P^{\prime \prime} . \forall u . \exists P^{\prime} . P \Longrightarrow_{l} u\) in \(P^{\prime \prime} \rightarrow a<c>\prec P^{\prime} \wedge\left(P^{\prime},([(x, c)]\right.\) - \(\left.\left.Q^{\prime}\right)[c::=u]\right) \in \operatorname{Rel}\)
by (rule Input)
then obtain \(P^{\prime \prime}\) where \(L 1: \forall u . \exists P^{\prime} . P \Longrightarrow_{l} u\) in \(P^{\prime \prime} \rightarrow a<c>\prec P^{\prime} \wedge\left(P^{\prime},([(x\right.\), \(\left.\left.c)] \cdot Q^{\prime}\right)[c::=u]\right) \in \operatorname{Rel}\) by blast
\[
\text { have } \forall u . \exists P^{\prime} . P \Longrightarrow_{l} u \text { in }\left([(c, x)] \cdot P^{\prime \prime}\right) \rightarrow a<x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}[x::=u]\right) \in \operatorname{Rel}
\] proof (auto)
fix \(u\)
from L1 obtain \(P^{\prime}\) where PTrans: \(P \Longrightarrow_{l} u\) in \(P^{\prime \prime} \rightarrow a<c>\prec P^{\prime}\) and \(P^{\prime} \operatorname{RelQ}\) : \(\left(P^{\prime},\left([(x, c)] \cdot Q^{\prime}\right)[c::=u]\right) \in\) Rel
by blast
from PTrans \(x\) Fresh \(P\) have \(P \Longrightarrow_{l} u\) in \(\left([(c, x)] \cdot P^{\prime \prime}\right) \rightarrow a<x>\prec P^{\prime}\) by (rule alphaInput)
moreover from \(P^{\prime}\) Rel \(Q^{\prime}\) cFresh \(Q^{\prime}\) have \(\left(P^{\prime}, Q^{\prime}[x::=u]\right) \in\) Rel by \((\) simp add: renaming[THEN sym] name-swap)
ultimately show \(\exists P^{\prime} . P \Longrightarrow_{l}\) u in \(\left([(c, x)] \cdot P^{\prime \prime}\right) \rightarrow a<x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}[x::=u]\right)\) \(\in\) Rel by blast qed
thus \(\exists P^{\prime \prime} . \forall u . \exists P^{\prime} . P \Longrightarrow_{l} u\) in \(P^{\prime \prime} \rightarrow a<x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}[x::=u]\right) \in\) Rel by
```

blast
qed
lemma simCases[case-names Bound Input Free]:
fixes P :: pi
and }Q ::p
and Rel :: (pi\timespi) set
and C :: 'a::fs-name

```

```

\prec P ^ { \prime } \wedge ( P ^ { \prime } , Q ^ { \prime } ) \in R e l
and Input: }\bigwedge\mp@subsup{Q}{}{\prime}\mathrm{ a x. }\llbracketQ\longmapstoa<x>\prec Q Q';x\sharpP\rrbracket\Longrightarrow\exists\mp@subsup{P}{}{\prime\prime}.\forallu.\exists\mp@subsup{P}{}{\prime}.P\Longrightarrow\mp@subsup{}{l}{l}
in }\mp@subsup{P}{}{\prime\prime}->a<x>\prec\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime}[x::=u])\inRe
and Free: }\bigwedge\mp@subsup{Q}{}{\prime}\alpha.Q\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime}\Longrightarrow(\exists\mp@subsup{P}{}{\prime}.P\Longrightarrow\mp@subsup{\Longrightarrow}{l}{`}\alpha\prec\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})
Rel)
shows P}\rightsquigarrow~<Rel>
using assms
by(auto simp add: simDef)
lemma simActBoundCases[consumes 1, case-names Input BoundOutput]:
fixes P :: pi
and a :: subject
and }x\mathrm{ :: name
and }\mp@subsup{Q}{}{\prime}:: p
and C ::'a::fs-name
and Rel :: (pi\timespi) set
assumes EqvtRel: eqvt Rel
and DerInput: \b.a = InputS b\Longrightarrow(\exists\mp@subsup{P}{}{\prime\prime}.\forallu.\exists\mp@subsup{P}{}{\prime}.(P\Longrightarrowu in P '
\prec P')^( (P', Q' [x::=u]) \in Rel)
and DerBoundOutput: \b. a = BoundOutputS b \Longrightarrow (\exists\mp@subsup{P}{}{\prime}.(P\Longrightarrow\Longrightarrowl b<\nux>
\prec P})\wedge(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\inRel
shows weakSimAct P (a«x» \prec- Q') P Rel
proof(simp add:weakSimAct-def fresh-prod, auto)
fix }\mp@subsup{Q}{}{\prime\prime}b
assume Eq:a«x» \prec Q' = b<\nuy>\prec \prec '"
assume yFreshP: y\#P
from Eq have }a=\mathrm{ BoundOutputS b by(simp add:residual.inject)

```

```

(P', Q')})\in\mathrm{ Rel
proof(cases x=y, auto simp add: residual.inject name-abs-eq)
fix P}\mp@subsup{P}{}{\prime
assume PTrans: P\Longrightarrow\hat{l}
assume P'RelQ':( }\mp@subsup{P}{}{\prime},([(x,y)]\cdot\mp@subsup{Q}{}{\prime\prime}))\in\operatorname{Rel
assume xineqy: }x\not=

```
```

    with PTrans yFreshP have yFreshP': \(y \sharp P^{\prime}\)
    ```
        by (force intro: freshTransition)
```

hence $b<\nu x>\prec P^{\prime}=b<\nu y>\prec[(x, y)] \cdot P^{\prime} \mathbf{b y}($ rule alphaBoundResidual)
moreover have $\left([(x, y)] \cdot P^{\prime}, Q^{\prime \prime}\right) \in \operatorname{Rel}$
proof -
from EqvtRel $P^{\prime} \operatorname{Rel} Q^{\prime}$ have $\left([(x, y)] \cdot P^{\prime},[(x, y)] \cdot\left([(x, y)] \cdot Q^{\prime \prime}\right)\right) \in$ Rel
by(rule eqvtRelI)
thus ?thesis by (simp add: name-calc)
qed

```
    ultimately show \(\exists P^{\prime} . P \Longrightarrow{\underset{l}{ }}^{\wedge} b<\nu y>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime \prime}\right) \in\) Rel using PTrans
by auto
    qed
next
    fix \(Q^{\prime \prime} b y u\)
    assume \(E q: a « x\rangle \prec Q^{\prime}=b<y>\prec Q^{\prime \prime}\)
    assume \(y\) Fresh \(P: y \sharp P\)
    from \(E q\) have \(a=\) InputS \(b\) by(simp add: residual.inject)
    from DerInput[OF this] obtain \(P^{\prime \prime}\) where \(L 1: \forall u . \exists P^{\prime} . P \Longrightarrow_{l} u\) in \(P^{\prime \prime} \rightarrow b<x>\)
\(\prec P^{\prime} \wedge\)
\[
\left(P^{\prime}, Q^{\prime}[x::=u]\right) \in \operatorname{Rel}
\]
by blast
have \(\forall u . \exists P^{\prime} . P \Longrightarrow_{l} u\) in \(\left([(x, y)] \cdot P^{\prime \prime}\right) \rightarrow b<y>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime \prime}[y::=u]\right) \in \operatorname{Rel}\) proof(rule allI)
fix \(u\)
from \(L 1 E q\) show \(\exists P^{\prime} . P \Longrightarrow_{l} u\) in \(\left([(x, y)] \cdot P^{\prime \prime}\right) \rightarrow b<y>\prec P^{\prime} \wedge\left(P^{\prime}\right.\), \(\left.Q^{\prime \prime}[y::=u]\right) \in\) Rel
proof (cases \(x=y\), auto simp add: residual.inject name-abs-eq)
assume Der: \(\forall u . \exists P^{\prime} . P \Longrightarrow_{l} u\) in \(P^{\prime \prime} \rightarrow b<x>\prec P^{\prime} \wedge\left(P^{\prime},([(x, y)] \cdot\right.\) \(\left.\left.Q^{\prime \prime}\right)[x::=u]\right) \in \operatorname{Rel}\)
assume \(x\) Fresh \(Q^{\prime \prime}: x \sharp Q^{\prime \prime}\)
from Der obtain \(P^{\prime}\) where PTrans: \(P \Longrightarrow_{l} u\) in \(P^{\prime \prime} \rightarrow b<x>\prec P^{\prime}\) and \(P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime},\left([(x, y)] \cdot Q^{\prime \prime}\right)[x::=u]\right) \in \operatorname{Rel}\)
by force
from PTrans \(y\) Fresh \(P\) have \(P \Longrightarrow_{l} u\) in \(\left([(x, y)] \cdot P^{\prime \prime}\right) \rightarrow b<y>\prec P^{\prime}\) by \((\) rule alphaInput)
moreover from \(x\) Fresh \(Q^{\prime \prime} P^{\prime} \operatorname{Rel} Q^{\prime}\) have \(\left(P^{\prime}, Q^{\prime \prime}[y::=u]\right) \in\) Rel
by (simp add: renaming)
ultimately show ?thesis by force
qed
qed
thus \(\exists P^{\prime \prime} . \forall u . \exists P^{\prime} . P \Longrightarrow_{l} u\) in \(P^{\prime \prime} \rightarrow b<y>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime \prime}[y::=u]\right) \in \operatorname{Rel}\) by blast
qed
```

lemma simActFreeCases[consumes 0, case-names Der]:
fixes $P$ :: $p i$
and $\alpha$ :: freeRes
and $\quad Q^{\prime}:: p i$
and Rel $::(p i \times p i)$ set
assumes $\exists P^{\prime} .\left(P \Longrightarrow{ }_{l} \hat{} \alpha \prec P^{\prime}\right) \wedge\left(P^{\prime}, Q^{\prime}\right) \in$ Rel
shows weakSimAct $P\left(\alpha \prec Q^{\prime}\right) P$ Rel
using assms
by (simp add: residual.inject weakSimAct-def fresh-prod)
lemma simE:
fixes $P$ :: $p i$
and Rel $::(p i \times p i)$ set
and $\quad Q \quad:: p i$
and $a$ :: name
and $x$ :: name
and $u$ :: name
and $\quad Q^{\prime}:: p i$
assumes $P \rightsquigarrow \wedge<$ Rel $>Q$
shows $Q \longmapsto a<\nu x>\prec Q^{\prime} \Longrightarrow x \sharp P \Longrightarrow \exists P^{\prime} . P \Longrightarrow{ }_{l}{ }^{\wedge} a<\nu x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right)$
$\in$ Rel
and $\quad Q \longmapsto a<x>\prec Q^{\prime} \Longrightarrow x \sharp P \Longrightarrow \exists P^{\prime \prime} . \forall u . \exists P^{\prime} . P \Longrightarrow{ }_{l} u$ in $P^{\prime \prime} \rightarrow a<x>$
$\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}[x::=u]\right) \in \operatorname{Rel}$
and $\quad Q \longmapsto \alpha \prec Q^{\prime} \Longrightarrow\left(\exists P^{\prime} . P \Longrightarrow \hat{i} \alpha \prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in\right.$ Rel $)$
using assms by (simp add: simDef)+
lemma weakSimTauChain:
fixes $P$ :: $p i$
and Rel $::(p i \times p i)$ set
and $Q \quad:: p i$
and $\quad Q^{\prime}:: p i$
assumes QChain: $Q \Longrightarrow_{\tau} Q^{\prime}$
and $\quad P R e l Q:(P, Q) \in$ Rel
and $\quad \operatorname{Sim}: \wedge P Q .(P, Q) \in \operatorname{Rel} \Longrightarrow P \rightsquigarrow \wedge<\operatorname{Rel}>Q$
shows $\exists P^{\prime} . P \Longrightarrow_{\tau} P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}$
proof -
from QChain show ?thesis
proof (induct rule: tauChainInduct)
case $i d$
have $P \Longrightarrow_{\tau} P$ by simp
with PRelQ show ?case by blast
next

```
```

    case(ih Q' (')
    have IH:\exists\mp@subsup{P}{}{\prime}.P\Longrightarrow\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mathrm{ Rel by fact}
    then obtain P' where PChain: P\Longrightarrow}\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime}\mathrm{ and }\mp@subsup{P}{}{\prime}\operatorname{RelQ}\mp@subsup{Q}{}{\prime}:(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\operatorname{Rel}\mathrm{ by
    blast
from }\mp@subsup{P}{}{\prime}\mathrm{ RelQ' have }\mp@subsup{P}{}{\prime}\rightsquigarrow`<R\mathrm{ Rel }>\mp@subsup{Q}{}{\prime}\mathrm{ by(rule Sim)
moreover have \mp@subsup{Q}{}{\prime}Trans: Q'\longmapsto\tau}\prec\mp@subsup{Q}{}{\prime\prime}\mathrm{ by fact
ultimately have }\exists\mp@subsup{P}{}{\prime\prime}.\mp@subsup{P}{}{\prime}\Longrightarrow\mp@subsup{\Longrightarrow}{\imath}{\prime}\tau<\mp@subsup{P}{}{\prime\prime}\wedge(\mp@subsup{P}{}{\prime\prime},\mp@subsup{Q}{}{\prime\prime})\in\mathrm{ Rel by(rule simE)
then obtain }\mp@subsup{P}{}{\prime\prime}\mathrm{ where 靔Trans: }\mp@subsup{P}{}{\prime}\Longrightarrow\mp@subsup{\Longrightarrow}{l}{}\tau\prec\mp@subsup{P}{}{\prime\prime}\mathrm{ and }\mp@subsup{P}{}{\prime\prime}Rel\mp@subsup{Q}{}{\prime\prime}:(\mp@subsup{P}{}{\prime\prime},\mp@subsup{Q}{}{\prime\prime})
Rel by blast
from P'Trans have P' \Longrightarrow>}\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime\prime}\mathrm{ by(rule tauTransitionChain)
with PChain have P \Longrightarrow}\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime\prime}\mathrm{ by auto
with P'RelQ" show ?case by blast
qed
qed
lemma simE2:
fixes P :: pi
and Rel :: (pi\timespi) set
and }Q :: p
and a :: name
and }x\mathrm{ :: name
and }\mp@subsup{Q}{}{\prime}:: p
assumes PSimQ: P\rightsquigarrow^ < Rel> Q
and Sim: \PQ.(P,Q)\inRel\LongrightarrowP\rightsquigarrow^<Rel> Q
and Eqvt: eqvt Rel
and PRelQ: (P,Q)\inRel

```

```

Q')}\in\operatorname{Rel

```

```

proof -
assume QTrans: Q\Longrightarrow`` }\mp@subsup{\Longrightarrow}{l}{`
assume xFreshP: x\sharpP

```

```

(P,Q) \inRel\rrbracket\Longrightarrow
\exists 和. P\Longrightarrow\hat{l}
proof -
fix PQax Q'
assume PSimQ: P\rightsquigarrow^ <Rel>}
assume QTrans: Q\Longrightarrowl`}\mp@subsup{\}{|}{`}a<\nux>\prec\mp@subsup{Q}{}{\prime
assume xFreshP: x\sharpP
assume xFreshQ: }x\sharp
assume PRelQ: (P,Q) \in Rel
from QTrans xFreshQ obtain }\mp@subsup{Q}{}{\prime\prime}\mp@subsup{Q}{}{\prime\prime\prime}\mathrm{ where QChain: }Q\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{Q}{}{\prime\prime
and \mp@subsup{Q}{}{\prime\prime}Trans: }\mp@subsup{Q}{}{\prime\prime}\longmapstoa<\nux>\prec\prec\mp@subsup{Q}{}{\prime\prime
and Q ''\primeChain: Q '/\prime \Longrightarrow>
by(force dest:Weak-Late-Step-Semantics.transitionE simp add: weakTransi-

```
```

tion-def)

```
from QChain PRelQ Sim have \(\exists P^{\prime \prime} . P \Longrightarrow_{\tau} P^{\prime \prime} \wedge\left(P^{\prime \prime}, Q^{\prime \prime}\right) \in\) Rel by(rule weakSim TauChain)
then obtain \(P^{\prime \prime}\) where PChain: \(P \Longrightarrow_{\tau} P^{\prime \prime}\) and \(P^{\prime \prime} \operatorname{Rel} Q^{\prime \prime}:\left(P^{\prime \prime}, Q^{\prime \prime}\right) \in \operatorname{Rel}\) by blast
from PChain \(x\) Fresh \(P\) have \(x\) Fresh \(P^{\prime \prime}: x \sharp P^{\prime \prime}\) by(rule freshChain)
from \(P^{\prime \prime} \operatorname{Rel} Q^{\prime \prime}\) have \(P^{\prime \prime} \rightsquigarrow \wedge<\) Rel \(>Q^{\prime \prime}\) by (rule Sim)
hence \(\exists P^{\prime \prime \prime} . P^{\prime \prime} \Longrightarrow{ }_{l} a<\nu x>\prec P^{\prime \prime \prime} \wedge\left(P^{\prime \prime \prime}, Q^{\prime \prime \prime}\right) \in\) Rel using \(Q^{\prime \prime}\) Trans \(x\) Fresh \(P^{\prime \prime}\)
by (rule \(\operatorname{sim} E)\)
then obtain \(P^{\prime \prime \prime}\) where \(P^{\prime \prime}\) Trans: \(P^{\prime \prime} \Longrightarrow \hat{l^{\prime}} a<\nu x>\prec P^{\prime \prime \prime}\) and \(P^{\prime \prime \prime} \operatorname{Rel} Q^{\prime \prime \prime}\) : \(\left(P^{\prime \prime \prime}, Q^{\prime \prime \prime}\right) \in\) Rel by blast
from \(P^{\prime \prime \prime}\) Rel \(Q^{\prime \prime \prime}\) have \(P^{\prime \prime \prime} \rightsquigarrow \wedge<\) Rel \(>Q^{\prime \prime \prime}\) by (rule Sim)
have \(\exists P^{\prime} . P^{\prime \prime \prime} \Longrightarrow{ }_{\tau} P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in\) Rel using \(Q^{\prime \prime \prime}\) Chain \(P^{\prime \prime \prime} R e l Q^{\prime \prime \prime}\) Sim by (rule weakSim TauChain)
then obtain \(P^{\prime}\) where \(P^{\prime \prime \prime}\) Chain: \(P^{\prime \prime \prime} \Longrightarrow{ }_{\tau} P^{\prime}\) and \(P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}\) by blast
from PChain \(P^{\prime \prime}\) Trans \(P^{\prime \prime \prime}\) Chain \(x\) Fresh \(P^{\prime \prime}\) have \(P \Longrightarrow{ }_{\imath} a<\nu x>\prec P^{\prime}\)
\(\mathbf{b y}\) (blast dest: chainTransitionAppend)
with \(P^{\prime}\) Rel \(Q^{\prime}\) show \(\exists P^{\prime} . P \Longrightarrow{\underset{l}{n}}_{\wedge} a<\nu x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in\) Rel by blast qed
have \(\exists c::\) name. \(c \sharp\left(Q, Q^{\prime}, P, x\right)\) by (blast intro: name-exists-fresh)
then obtain \(c:: n a m e\) where cFresh \(Q: c \sharp Q\) and \(c F r e s h Q^{\prime}: c \sharp Q^{\prime}\) and \(c F r e s h P\) : \(c \sharp P\)
and xineqc: \(x \neq c\)
by (force simp add: fresh-prod)
from \(Q\) Trans cFresh \(Q^{\prime}\) have \(Q \Longrightarrow \hat{l}{ }^{\wedge} a<\nu c>\prec\left([(x, c)] \cdot Q^{\prime}\right)\) by (simp add: alphaBoundResidual)
with \(P \operatorname{Sim} Q\) have \(\exists P^{\prime} . P \Longrightarrow \hat{\imath} \dot{a} a<\nu c>\prec P^{\prime} \wedge\left(P^{\prime},[(x, c)] \cdot Q^{\prime}\right) \in\) Rel using
cFreshP cFresh \(Q\langle(P, Q) \in\) Rel \(\rangle\)
by (rule Goal)
then obtain \(P^{\prime}\) where PTrans: \(P \Longrightarrow \hat{l} a<\nu c>\prec P^{\prime}\) and \(P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime},[(x, c)]\right.\)
- \(\left.Q^{\prime}\right) \in\) Rel
by force
have \(P \Longrightarrow{ }^{\wedge} a<\nu x>\prec\left([(x, c)] \cdot P^{\prime}\right)\)
proof -
from PTrans \(x\) FreshP xineqc have \(x \sharp P^{\prime}\) by(rule freshTransition)
with PTrans show ?thesis by (simp add: alphaBoundResidual name-swap)
qed
moreover have \(\left([(x, c)] \cdot P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}\)
proof -
from Eqvt \(P^{\prime} \operatorname{Rel} Q^{\prime}\) have \(\left([(x, c)] \cdot P^{\prime},[(x, c)] \cdot[(x, c)] \cdot Q^{\prime}\right) \in\) Rel
```

        by(rule eqvtRelI)
        thus ?thesis by simp
    qed
    ```

```

next
assume QTrans: Q \Longrightarrowi }\alpha\prec\mp@subsup{Q}{}{\prime
thus \exists\mp@subsup{P}{}{\prime}.P\Longrightarrow\hat{l}\alpha}\alpha<\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\operatorname{Rel
proof(induct rule: transitionCases)
case Step
have }Q\mp@subsup{\Longrightarrow}{l}{}\alpha\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
then obtain }\mp@subsup{Q}{}{\prime\prime}\mp@subsup{Q}{}{\prime\prime\prime}\mathrm{ where QChain: Q >}\mp@subsup{~}{\tau}{}\mp@subsup{Q}{}{\prime\prime
and }\mp@subsup{Q}{}{\prime\prime}\mathrm{ Trans: }\mp@subsup{Q}{}{\prime\prime}\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime\prime\prime
and Q '/'Chain: Q }\mp@subsup{Q}{}{\prime\prime\prime}\Longrightarrow\mp@subsup{}{\tau}{}\mp@subsup{Q}{}{\prime
by(blast dest:Weak-Late-Step-Semantics.transitionE)
from QChain PRelQ Sim have }\exists\mp@subsup{P}{}{\prime\prime}.P\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime\prime}\wedge(\mp@subsup{P}{}{\prime\prime},\mp@subsup{Q}{}{\prime\prime})\in\mathrm{ Rel
by(rule weakSimTauChain)
then obtain P'\prime where PChain: P \Longrightarrow\Longrightarrow}\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime\prime}\mathrm{ and P}\mp@subsup{P}{}{\prime\prime}RelQ\mp@subsup{Q}{}{\prime\prime}:(\mp@subsup{P}{}{\prime\prime},\mp@subsup{Q}{}{\prime\prime})\in\operatorname{Rel
by blast
from }\mp@subsup{P}{}{\prime\prime}RelQ'/ have P'\prime \rightsquigarrow < Rel> Q' by(rule Sim)
hence }\exists\mp@subsup{P}{}{\prime\prime\prime}.\mp@subsup{P}{}{\prime\prime}\Longrightarrow\hat{l}\alpha<<\mp@subsup{P}{}{\prime\prime\prime}\wedge(\mp@subsup{P}{}{\prime\prime\prime},\mp@subsup{Q}{}{\prime\prime\prime})\in\mathrm{ Rel using }\mp@subsup{Q}{}{\prime\prime}\mathrm{ Trans
by(rule simE)

```

```

Q'\prime\prime)}\in\mathrm{ Rel
by blast
from }\mp@subsup{P}{}{\prime\prime\prime}RelQ\mp@subsup{Q}{}{\prime\prime\prime}\mathrm{ have }\mp@subsup{P}{}{\prime\prime\prime}\rightsquigarrow^<Rel> Q Q'\prime by(rule Sim
have }\exists\mp@subsup{P}{}{\prime}.\mp@subsup{P}{}{\prime\prime\prime}\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mathrm{ Rel using Q Q'IChain P}\mp@subsup{P}{}{\prime\prime\prime}RelQ\mp@subsup{Q}{}{\prime\prime\prime}\mathrm{ Sim
by(rule weakSimTauChain)
then obtain }\mp@subsup{P}{}{\prime}\mathrm{ where }\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ Chain: }\mp@subsup{P}{}{\prime\prime\prime}\Longrightarrow\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime}\mathrm{ and P'RelQ':( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\operatorname{Rel
by blast
from PChain P'ITrans P P'\prime}\mathrm{ Chain have P Ci 人}\alpha\prec\mp@subsup{P}{}{\prime
by(blast dest: chainTransitionAppend)
with P'RelQ' show ?case by blast
next
case Stay
have }\alpha\prec\mp@subsup{Q}{}{\prime}=\tau\precQ by fac
hence Q = Q' and \alpha=\tau by(simp add:residual.inject)+
moreover have P\Longrightarrow\^}<br>precP\mathrm{ by(simp add:weakTransition-def)
ultimately show ?case using PRelQ by blast
qed
qed
lemma eqvtI:
fixes }P\mathrm{ :: pi
and Q :: pi
and Rel :: (pi\times pi) set

```
and perm :: name prm
assumes Sim: \(P \rightsquigarrow \wedge<\) Rel \(>Q\)
and RelRel': Rel \(\subseteq\) Rel \(^{\prime}\)
and EqvtRel': eqvt Rel \({ }^{\prime}\)
```

    shows \((\) perm \(\cdot P) \rightsquigarrow{ }^{\wedge}<\) Rel \(^{\prime}>(\) perm \(\cdot Q)\)
    proof -
from EqvtRel' show ?thesis
proof $($ induct rule: simCasesCont $[$ of $-($ perm $\cdot P)])$
case(Bound $Q^{\prime}$ a $x$ )
have Trans: $($ perm $\cdot Q) \longmapsto a<\nu x>\prec Q^{\prime}$ and $x$ Fresh $P: x \sharp$ perm $\cdot P$ by fact +

```
    from Trans have \((\) rev perm \(\cdot(\) perm \(\cdot Q)) \longmapsto\) rev perm \(\cdot\left(a<\nu x>\prec Q^{\prime}\right)\)
        by (rule eqvts)
    hence \(Q \longmapsto(\) rev perm \(\cdot a)<\nu(\) rev perm \(\cdot x)>\prec\left(\right.\) rev perm \(\left.\cdot Q^{\prime}\right)\)
        by (simp add: name-rev-per)
    moreover from \(x\) Fresh \(P\) have (rev perm \(\cdot x) \sharp P\) by (simp add: name-fresh-left)
    ultimately have \(\exists P^{\prime} . P \Longrightarrow{ }_{i}^{\prime}(\) rev perm \(\cdot a)<\nu(\) rev perm \(\cdot x)>\prec P^{\prime} \wedge\left(P^{\prime}\right.\),
rev perm \(\left.\cdot Q^{\prime}\right) \in\) Rel using Sim
        by(force intro: simE)
    then obtain \(P^{\prime}\) where PTrans: \(P \Longrightarrow{ }_{l}\) ^(rev perm \(\left.\cdot a\right)<\nu(\) rev perm \(\cdot x)>\prec\)
\(P^{\prime}\) and \(P^{\prime}\) RelQ \(Q^{\prime}:\left(P^{\prime}\right.\), rev perm \(\left.\cdot Q^{\prime}\right) \in\) Rel by blast
from PTrans have \((\) perm \(\cdot P) \Longrightarrow \hat{l}\) perm \(\cdot((\) rev perm \(\cdot a)<\nu(\) rev perm \(\cdot x)>\) \(\prec P^{\prime}\) )
by(rule Weak-Late-Semantics.eqvtI)
hence L1: \((\) perm \(\cdot P) \Longrightarrow \hat{l^{\prime}} a<\nu x>\prec\left(\right.\) perm \(\left.\cdot P^{\prime}\right) \mathbf{b y}(\) simp add: name-per-rev \()\)
from \(P^{\prime}\) RelQ \(Q^{\prime}\) RelRel \({ }^{\prime}\) have ( \(P^{\prime}\), rev perm \(\cdot Q^{\prime}\) ) Rel \(^{\prime}\) by blast
with EqvtRel' have \(\left(\right.\) perm \(\cdot P^{\prime}\), perm \(\cdot\left(\right.\) rev perm \(\left.\left.\cdot Q^{\prime}\right)\right) \in\) Rel \(^{\prime}\)
by (rule eqvtRelI)
hence (perm • \(\left.P^{\prime}, Q^{\prime}\right) \in\) Rel \(^{\prime} \mathbf{b y}(\) simp add: name-per-rev)
with L1 show ?case by blast
next
case(Input \(\left.Q^{\prime} a x\right)\)
have Trans: \((\) perm \(\cdot Q) \longmapsto a<x>\prec Q^{\prime}\) and \(x\) FreshP: \(x \sharp\) perm \(\cdot P\) by fact +
from Trans have (rev perm • \((\) perm \(\cdot Q)) \longmapsto\) rev perm \(\cdot\left(a<x>\prec Q^{\prime}\right)\)
by (rule eqvts)
hence \(Q \longmapsto(\) rev perm \(\cdot a)<(\) rev perm \(\cdot x)>\prec\left(\right.\) rev perm \(\left.\cdot Q^{\prime}\right)\)
by (simp add: name-rev-per)
moreover from \(x\) Fresh \(P\) have \(x\) FreshP: (rev perm \(\cdot x) \sharp P\) by (simp add:
name-fresh-left)
ultimately have \(\exists P^{\prime \prime} . \forall u . \exists P^{\prime} . P \Longrightarrow_{l}\) u in \(P^{\prime \prime} \rightarrow(\) rev perm \(\cdot a)<(\) rev perm \(\cdot\) \(x)>\prec P^{\prime} \wedge\left(P^{\prime},\left(\right.\right.\) rev perm \(\left.\cdot Q^{\prime}\right)[(\) rev perm \(\left.\cdot x)::=u]\right) \in\) Rel using Sim
by (force intro: simE)
then obtain \(P^{\prime \prime}\) where L1: \(\forall u . \exists P^{\prime} . P \Longrightarrow_{l} u\) in \(P^{\prime \prime} \rightarrow(\) rev perm \(\cdot a)<(\) rev perm \(\cdot x)>\prec P^{\prime} \wedge\left(P^{\prime},\left(\right.\right.\) rev perm \(\left.\cdot Q^{\prime}\right)[(\) rev perm \(\left.\cdot x)::=u]\right) \in\) Rel
by blast
have \(\forall u . \exists P^{\prime} .(\) perm \(\cdot P) \Longrightarrow_{l} u\) in \(\left(\right.\) perm \(\left.\cdot P^{\prime \prime}\right) \rightarrow a<x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}[x::=u]\right)\) \(\in R e l^{\prime}\)
proof (rule allI)
fix \(u\)
from L1 obtain \(P^{\prime}\) where PTrans: \(P \Longrightarrow_{l}(\) rev perm \(\cdot u)\) in \(P^{\prime \prime} \rightarrow(\) rev perm - a) \(<(\) rev perm \(\cdot x)>\prec P^{\prime}\)
and \(P^{\prime}\) RelQ \({ }^{\prime}:\left(P^{\prime},\left(\right.\right.\) rev perm \(\left.\cdot Q^{\prime}\right)[(\) rev perm \(\cdot x)::=(\) rev perm - u)]) \(\in\) Rel by blast
from PTrans have \(\left(\right.\) perm • P) \(\Longrightarrow_{l}(\) perm • (rev perm • u) \()\) in (perm • \(\left.P^{\prime \prime}\right) \rightarrow(\) perm \(\cdot\) rev perm \(\cdot a)<(\) perm \(\cdot\) rev perm \(\cdot x)>\prec\left(\right.\) perm \(\left.\cdot P^{\prime}\right)\)
by (rule-tac Weak-Late-Step-Semantics.eqvtI, auto)
hence L2: \((\) perm \(\cdot P) \Longrightarrow_{l} u\) in \(\left(\right.\) perm \(\left.\cdot P^{\prime \prime}\right) \rightarrow a<x>\prec\left(\right.\) perm \(\left.\cdot P^{\prime}\right) \mathbf{b y}(\) simp add: name-per-rev)
from \(P^{\prime}\) RelQ \({ }^{\prime}\) RelRel \({ }^{\prime}\) have \(\left(P^{\prime},\left(\right.\right.\) rev perm \(\left.\cdot Q^{\prime}\right)[(\) rev perm \(\cdot x)::=(\) rev perm - u)]) \(\in \operatorname{Rel}^{\prime}\) by blast
with EqvtRel \({ }^{\prime}\) have \(\left(\right.\) perm \(\cdot P^{\prime}\), perm • \(\left(\left(\right.\right.\) rev perm \(\left.\cdot Q^{\prime}\right)[(\) rev perm \(\cdot x)::=(\) rev perm \(\cdot u)])) \in \operatorname{Rel}^{\prime}\)
by(rule eqvtRelI)
hence \(\left(\right.\) perm \(\left.\cdot P^{\prime}, Q^{\prime}[x::=u]\right) \in\) Rel \(^{\prime} \mathbf{b y}(\) simp add: name-per-rev eqvt-subs \([\) THEN sym] name-calc)
with \(L 2\) show \(\exists P^{\prime} .(\) perm \(\cdot P) \Longrightarrow_{l} u\) in \(\left(\right.\) perm \(\left.\cdot P^{\prime \prime}\right) \rightarrow a<x>\prec P^{\prime} \wedge\left(P^{\prime}\right.\), \(\left.Q^{\prime}[x::=u]\right) \in\) Rel \(^{\prime}\) by blast
qed
thus ?case by blast
next
case (Free \(Q^{\prime} \alpha\) )
have Trans: \((\) perm \(\cdot Q) \longmapsto \alpha \prec Q^{\prime}\) by fact
from Trans have (rev perm • \((\) perm \(\cdot Q)) \longmapsto\) rev perm \(\cdot\left(\alpha \prec Q^{\prime}\right)\)
by (rule eqvts)
hence \(Q \longmapsto(\) rev perm \(\cdot \alpha) \prec\left(\right.\) rev perm \(\left.\cdot Q^{\prime}\right)\)
by (simp add: name-rev-per)
with Sim have \(\left(\exists P^{\prime} . P \Longrightarrow{\underset{\imath}{i}}^{\wedge}(\right.\) rev perm \(\cdot \alpha) \prec P^{\prime} \wedge\left(P^{\prime},\left(\right.\right.\) rev perm \(\left.\left.\cdot Q^{\prime}\right)\right) \in\) Rel)
by (rule \(\operatorname{sim} E)\)
then obtain \(P^{\prime}\) where PTrans: \(P \Longrightarrow \Longrightarrow_{l}(\) rev perm \(\cdot \alpha) \prec P^{\prime}\) and PRel: \(\left(P^{\prime}\right.\), \(\left(\right.\) rev perm \(\left.\left.\cdot Q^{\prime}\right)\right) \in\) Rel by blast
from PTrans have \((\) perm \(\cdot P) \Longrightarrow \hat{l}\) perm \(\cdot\left((\right.\) rev perm \(\left.\cdot \alpha) \prec P^{\prime}\right)\)
by (rule Weak-Late-Semantics.eqvtI)
hence L1: \((\) perm \(\cdot P) \Longrightarrow{ }^{\wedge} \alpha \prec\left(\right.\) perm \(\left.\cdot P^{\prime}\right)\) by (simp add: name-per-rev)
from PRel EqvtRel \({ }^{\prime}\) RelRel \({ }^{\prime}\) have \(\left(\left(\right.\right.\) perm • \(\left.P^{\prime}\right),\left(\right.\) perm • (rev perm • \(\left.\left.\left.Q^{\prime}\right)\right)\right) \in\) \(R e l^{\prime}\)
by (force intro: equtRelI)
hence \(\left(\left(\right.\right.\) perm \(\left.\left.\cdot P^{\prime}\right), Q^{\prime}\right) \in\) Rel \(^{\prime} \mathbf{b y}(\) simp add: name-per-rev \()\)
with \(L 1\) show? case by blast
qed
qed
```

lemma reflexive:
fixes $P$ :: $p i$
and Rel $::(p i \times p i)$ set
assumes $I d \subseteq$ Rel
shows $P \leadsto \wedge<$ Rel $>P$
using assms
by(auto intro: Weak-Late-Step-Semantics.singleActionChain
simp add: simDef weakTransition-def)
lemma transitive
fixes $P \quad:: p i$
and $\quad Q \quad:: p i$
and $R \quad:: p i$
and Rel :: $(p i \times p i)$ set
and Rel' :: $(p i \times p i)$ set
and Rel ${ }^{\prime \prime}::(p i \times p i)$ set
assumes QSimR: $Q \rightsquigarrow \wedge^{\prime}<$ Rel $^{\prime}>R$
and Eqvt: equt Rel
and Eqvt': equt Rel"
and Trans: Rel $O$ Rel ${ }^{\prime} \subseteq$ Rel $^{\prime \prime}$
and Sim: $\bigwedge P Q .(P, Q) \in \operatorname{Rel} \Longrightarrow P \rightsquigarrow \wedge<\operatorname{Rel}>Q$
and $\quad P R e l Q:(P, Q) \in$ Rel
shows $P \rightsquigarrow{ }^{\wedge}<$ Rel $^{\prime \prime}>R$
proof -
from PRelQ have PSimQ: $P \leadsto \wedge<$ Rel $>Q$ by (rule Sim)
from Eqvt' show ?thesis
proof $($ induct rule: simCasesCont $[$ of $-(P, Q)])$
case(Bound $R^{\prime}$ a $x$ )
have $R$ Trans: $R \longmapsto a<\nu x>\prec R^{\prime}$ by fact
have $x \sharp(P, Q)$ by fact
hence $x$ Fresh $: x \sharp P$ and $x$ Fresh $Q: x \sharp Q$ by(simp add: fresh-prod) +
from QSimR RTrans xFresh $Q$ have $\exists Q^{\prime} . Q \Longrightarrow \Longrightarrow_{l} a<\nu x>\prec Q^{\prime} \wedge\left(Q^{\prime}, R^{\prime}\right) \in$
Rel'
by (rule $\operatorname{sim} E)$
then obtain $Q^{\prime}$ where $Q$ Trans: $Q \Longrightarrow_{l}^{\wedge} a<\nu x>\prec Q^{\prime}$ and $Q^{\prime} \operatorname{RelR}^{\prime}:\left(Q^{\prime}, R^{\prime}\right)$
$\in R e l^{\prime}$ by blast
from PSimQ Sim Eqvt PRelQ QTrans xFreshP have $\exists P^{\prime} . P \Longrightarrow{ }_{l} a<\nu x>\prec$
$P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in$ Rel
by (rule simE2)
then obtain $P^{\prime}$ where PTrans: $P \Longrightarrow \Longrightarrow_{l}^{\hat{a}} a<\nu x>\prec P^{\prime}$ and $P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right)$
$\in$ Rel by blast
moreover from $P^{\prime}$ RelQ' $Q^{\prime}$ Rel $R^{\prime}$ Trans have $\left(P^{\prime}, R^{\prime}\right) \in$ Rel $^{\prime \prime}$ by blast

```
ultimately show ?case by blast

\section*{next}
case (Input \(R^{\prime}\) a \(x\) )
have \(R\) Trans: \(R \longmapsto a<x>\prec R^{\prime}\) by fact
have \(x \sharp(P, Q)\) by fact
hence \(x\) Fresh \(P: x \sharp P\) and \(x\) Fresh \(Q: x \sharp Q\) by(simp add: fresh-prod)+
from \(Q \operatorname{Sim} R\) RTrans \(x\) Fresh \(Q\) obtain \(Q^{\prime \prime}\) where \(\forall u . \exists Q^{\prime} . Q \Longrightarrow_{l} u\) in \(Q^{\prime \prime} \rightarrow a<x>\prec Q^{\prime} \wedge\left(Q^{\prime}, R^{\prime}[x::=u]\right) \in\) Rel \(^{\prime}\)
by (blast dest: simE)
hence \(\exists Q^{\prime \prime \prime} . Q \Longrightarrow_{\tau} Q^{\prime \prime \prime} \wedge Q^{\prime \prime \prime} \longrightarrow a<x>\prec Q^{\prime \prime} \wedge\left(\forall u . \exists Q^{\prime} . Q^{\prime \prime}[x::=u] \Longrightarrow_{\tau}\right.\) \(\left.Q^{\prime} \wedge\left(Q^{\prime}, R^{\prime}[x::=u]\right) \in R e l^{\prime}\right)\)
by (simp add: inputTransition-def, blast)
then obtain \(Q^{\prime \prime \prime}\) where \(Q\) Chain: \(Q \Longrightarrow_{\tau} Q^{\prime \prime \prime}\)
\[
\begin{aligned}
& \text { and } Q^{\prime \prime \prime} \text { Trans: } Q^{\prime \prime \prime} \longmapsto a<x>\prec Q^{\prime \prime} \\
& \text { and } L 1: \forall u . \exists Q^{\prime} . Q^{\prime \prime}[x::=u] \Longrightarrow_{\tau} Q^{\prime} \wedge\left(Q^{\prime}, R^{\prime}[x::=u]\right) \in R e l^{\prime}
\end{aligned}
\]
by blast
from QChain PRelQ Sim have \(\exists P^{\prime \prime \prime} . P \Longrightarrow{ }_{\tau} P^{\prime \prime \prime} \wedge\left(P^{\prime \prime \prime}, Q^{\prime \prime \prime}\right) \in\) Rel by(rule weakSimTauChain)
then obtain \(P^{\prime \prime \prime}\) where PChain: \(P \Longrightarrow{ }_{\tau} P^{\prime \prime \prime}\) and \(P^{\prime \prime \prime} R e l Q^{\prime \prime \prime}:\left(P^{\prime \prime \prime}, Q^{\prime \prime \prime}\right) \in\) Rel by blast
from PChain \(x\) Fresh \(P\) have \(x\) Fresh \(P^{\prime \prime \prime}: x \sharp P^{\prime \prime \prime}\) by (rule freshChain)
from \(P^{\prime \prime \prime}\) Rel \(Q^{\prime \prime \prime}\) have \(P^{\prime \prime \prime} \leadsto \wedge<\) Rel \(>Q^{\prime \prime \prime}\) by (rule Sim)
hence \(\exists P^{\prime \prime \prime \prime} . \forall u . \exists P^{\prime \prime} . P^{\prime \prime \prime} \Longrightarrow_{l}\) u in \(P^{\prime \prime \prime \prime} \rightarrow a<x>\prec P^{\prime \prime} \wedge\left(P^{\prime \prime}, Q^{\prime \prime}[x::=u]\right) \in\) Rel using \(Q^{\prime \prime \prime}\) Trans xFresh \(P^{\prime \prime \prime}\)
by (rule simE)
then obtain \(P^{\prime \prime \prime \prime}\) where L2: \(\forall u . \exists P^{\prime \prime} . P^{\prime \prime \prime} \Longrightarrow_{l} u\) in \(P^{\prime \prime \prime \prime} \rightarrow a<x>\prec P^{\prime \prime} \wedge\) \(\left(P^{\prime \prime}, Q^{\prime \prime}[x::=u]\right) \in \operatorname{Rel}\)
by blast
have \(\forall u . \exists P^{\prime} Q^{\prime} . P \Longrightarrow_{l} u\) in \(P^{\prime \prime \prime \prime} \rightarrow a<x>\prec P^{\prime} \wedge\left(P^{\prime}, R^{\prime}[x::=u]\right) \in R e l^{\prime \prime}\)
proof (rule allI)
fix \(u\)
from L1 obtain \(Q^{\prime}\) where \(Q^{\prime \prime}\) Chain: \(Q^{\prime \prime}[x::=u] \Longrightarrow_{\tau} Q^{\prime}\) and \(Q^{\prime} \operatorname{Rel}^{\prime}:\left(Q^{\prime}\right.\), \(\left.R^{\prime}[x::=u]\right) \in \operatorname{Rel}^{\prime}\)
by blast
from L2 obtain \(P^{\prime \prime}\) where \(P^{\prime \prime \prime}\) Trans: \(P^{\prime \prime \prime} \Longrightarrow_{l}\) u in \(P^{\prime \prime \prime \prime} \rightarrow a<x>\prec P^{\prime \prime}\) and \(P^{\prime \prime} \operatorname{Rel} Q^{\prime \prime}:\left(P^{\prime \prime}, Q^{\prime \prime}[x::=u]\right) \in \operatorname{Rel}\)
by blast
from \(P^{\prime \prime}\) RelQ \(Q^{\prime \prime}\) have \(P^{\prime \prime} \rightsquigarrow \wedge<\) Rel \(>Q^{\prime \prime}[x::=u] \mathbf{b y}(\) rule Sim \()\)
have \(\exists P^{\prime} . P^{\prime \prime} \Longrightarrow_{\tau} P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in\) Rel using \(Q^{\prime \prime}\) Chain \(P^{\prime \prime} \operatorname{Rel} Q^{\prime \prime}\) Sim by(rule weakSimTauChain)
then obtain \(P^{\prime}\) where \(P^{\prime \prime}\) Chain: \(P^{\prime \prime} \Longrightarrow_{\tau} P^{\prime}\) and \(P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}\) by blast
from PChain \(P^{\prime \prime \prime}\) Trans \(P^{\prime \prime}\) Chain have \(P \Longrightarrow_{l}\) u in \(P^{\prime \prime \prime \prime} \rightarrow a<x>\prec P^{\prime}\)
by(blast dest: Weak-Late-Step-Semantics.chainTransitionAppend)
moreover from \(P^{\prime} \operatorname{Rel} Q^{\prime} Q^{\prime} \operatorname{Rel} R^{\prime}\) have \(\left(P^{\prime}, R^{\prime}[x::=u]\right) \in\) Rel \(^{\prime \prime}\) by (insert Trans, auto)
ultimately show \(\exists P^{\prime} Q^{\prime} . P \Longrightarrow_{l}\) u in \(P^{\prime \prime \prime \prime} \rightarrow a<x>\prec P^{\prime} \wedge\left(P^{\prime}, R^{\prime}[x::=u]\right) \in\) Rel" by blast
```

    qed
    thus ?case by force
    next
    case(Free R' \alpha)
    have RTrans: R\longmapsto\alpha\prec R' by fact
    with QSimR have }\exists\mp@subsup{Q}{}{\prime}.Q\Longrightarrow\mp@subsup{\}{l}{\prime}\alpha<<\mp@subsup{Q}{}{\prime}\wedge(\mp@subsup{Q}{}{\prime},\mp@subsup{R}{}{\prime})\in\mp@subsup{R}{}{\prime}\mp@subsup{R}{}{\prime}\mathbf{by}(\mathrm{ bule simE)
    ```

```

by blast

```

```

Rel by(rule simE2)
then obtain P' where PTrans: P\Longrightarrow\hat{l}}
by blast
from P'RelQ' Q'RelR' Trans have ( }\mp@subsup{P}{}{\prime},\mp@subsup{R}{}{\prime})\in\mathrm{ Rel '' by blast
with PTrans show }\exists\mp@subsup{P}{}{\prime}.P\Longrightarrow\mp@subsup{\Longrightarrow}{l}{\hat{*}}\alpha\prec\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},\mp@subsup{R}{}{\prime})\inRel'\prime\prime by blas
qed
qed
lemma strongSim WeakSim:
fixes P :: pi
and }Q ::p
and Rel :: (pi\timespi) set
assumes PSimQ: P}\rightsquigarrow[Rel]
shows P\rightsquigarrow^<Rel>}
proof(induct rule: simCases)
case(Bound Q' a x)
have }Q\longmapstoa<\nux><\prec\mp@subsup{Q}{}{\prime}\mathrm{ and }x\sharpP\mathrm{ by fact+
with PSimQ obtain }\mp@subsup{P}{}{\prime}\mathrm{ where PTrans: P}\longmapstoa<\nux>< < '' and P'RelQ':( ( ''
Q')\inRel
by(force dest: Strong-Late-Sim.simE simp add: derivative-def)
from PTrans have P\Longrightarrow看 }a<\nux>\prec\prec\mp@subsup{P}{}{\prime
by(force intro:Weak-Late-Step-Semantics.singleActionChain simp add: weak-
Transition-def)
with P'RelQ' show ?case by blast
next
case(Input Q' a x)
assume }Q\longmapstoa<x>\prec\mp@subsup{Q}{}{\prime}\mathrm{ and }x\sharp
with PSimQ obtain P' where PTrans: P}\longmapstoa<x>\prec\mp@subsup{P}{}{\prime}\mathrm{ and PDer:derivative
P' Q' (InputS a) x Rel
by(blast dest: Strong-Late-Sim.simE)
have }\forallu.\exists\mp@subsup{P}{}{\prime\prime}.P\mp@subsup{\Longrightarrow}{l}{}u\mathrm{ in }\mp@subsup{P}{}{\prime}->a<x>\prec\prec\mp@subsup{P}{}{\prime\prime}\wedge(\mp@subsup{P}{}{\prime\prime},\mp@subsup{Q}{}{\prime}[x::=u])\in\operatorname{Rel
proof(rule allI)
fix }
from PTrans have P \Longrightarrow\Longrightarrowlu in P}\mp@subsup{P}{}{\prime}->a<x>< \mp@subsup{P}{}{\prime}[x::=u] by(blast intro:Weak-Late-Step-Semantics.singleActio
moreover from PDer have ( }\mp@subsup{P}{}{\prime}[x::=u],\mp@subsup{Q}{}{\prime}[x::=u])\in\mathrm{ Rel by(force simp add:
derivative-def)

```
ultimately show \(\exists P^{\prime \prime} . P \Longrightarrow_{l} u\) in \(P^{\prime} \rightarrow a<x>\prec P^{\prime \prime} \wedge\left(P^{\prime \prime}, Q^{\prime}[x::=u]\right) \in \operatorname{Rel}\) by auto
qed
thus ?case by blast
next
case (Free \(Q^{\prime} \alpha\) )
have \(Q \longmapsto \alpha \prec Q^{\prime}\) by fact
with PSimQ obtain \(P^{\prime}\) where PTrans: \(P \longmapsto \alpha \prec P^{\prime}\) and \(P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in\) Rel
by (blast dest: Strong-Late-Sim.simE)
from PTrans have \(P \Longrightarrow \hat{i} \alpha \prec P^{\prime}\) by (rule Weak-Late-Semantics.singleActionChain)
with \(P^{\prime}\) RelQ' show ?case by blast
qed
lemma strongAppend:
fixes \(P \quad:: p i\)
and \(Q \quad:: p i\)
and \(R \quad:: p i\)
and Rel :: \((p i \times p i)\) set
and Rel' \(::(p i \times p i)\) set
and \(R e l^{\prime \prime}::(p i \times p i)\) set
assumes \(P \operatorname{Sim} Q: P \rightsquigarrow \wedge<\operatorname{Rel}>Q\)
and \(\quad Q \operatorname{SimR}: Q \rightsquigarrow\left[\mathrm{Rel}^{\prime}\right] R\)
and Eqvt": eqvt Rel"
and Trans: Rel \(O\) Rel \(^{\prime} \subseteq\) Rel \(^{\prime \prime}\)
shows \(P \rightsquigarrow \wedge<\) Rel \(^{\prime \prime}>R\)
proof -
from Eqvt" show ?thesis
proof \((\) induct rule: \(\operatorname{sim} C a s e s C o n t[o f-(P, Q)])\)
case(Bound R' a x)
have \(x \sharp(P, Q)\) by fact
hence \(x\) Fresh \(P: x \sharp P\) and \(x\) Fresh \(Q\) : \(x \sharp Q\) by(simp add: fresh-prod)+
have \(R\) Trans: \(R \longmapsto a<\nu x>\prec R^{\prime}\) by fact
from \(x\) Fresh \(Q\) QSimR RTrans obtain \(Q^{\prime}\) where \(Q\) Trans: \(Q \longmapsto a<\nu x>\prec Q^{\prime}\) and \(Q^{\prime}\) Rel \(^{\prime} R^{\prime}:\left(Q^{\prime}, R^{\prime}\right) \in \operatorname{Rel}^{\prime}\)
by(force dest: Strong-Late-Sim.simE simp add: derivative-def)
with PSimQ QTrans xFreshP have \(\exists P^{\prime} . P \Longrightarrow{ }_{l} \quad a<\nu x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in\) Rel
by(blast intro: simE)
then obtain \(P^{\prime}\) where PTrans: \(P \Longrightarrow \Longrightarrow_{l}^{\wedge} a<\nu x>\prec P^{\prime}\) and \(P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right)\)
\(\in\) Rel by blast
moreover from \(P^{\prime} R e l Q^{\prime} Q^{\prime}\) Rel \(^{\prime} R^{\prime}\) Trans have \(\left(P^{\prime}, R^{\prime}\right) \in\) Rel \(^{\prime \prime}\) by blast
ultimately show ?case by blast
next
case(Input \(R^{\prime}\) a \(x\) )
have \(R\) Trans: \(R \longmapsto a<x>\prec R^{\prime}\) by fact
have \(x \sharp(P, Q)\) by fact
hence \(x\) Fresh \(P: x \sharp P\) and \(x\) Fresh \(Q\) : \(x \sharp Q\) by(simp add: fresh-prod)+
from \(Q\) SimR RTrans \(x\) Fresh \(Q\) obtain \(Q^{\prime}\) where \(Q\) Trans: \(Q \longmapsto a<x>\prec Q^{\prime}\) and \(Q^{\prime}\) Der: derivative \(Q^{\prime} R^{\prime}\left(\right.\) InputS a) x Rel \({ }^{\prime}\)
by (blast dest: Strong-Late-Sim.simE)
from QTrans PSimQ xFreshP obtain \(P^{\prime \prime}\) where L2: \(\forall u . \exists P^{\prime} . P \Longrightarrow_{l} u\) in \(P^{\prime \prime} \rightarrow a<x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}[x::=u]\right) \in \operatorname{Rel}\)
by (blast dest: simE)
have \(\forall u . \exists P^{\prime} . P \Longrightarrow{ }_{l} u\) in \(P^{\prime \prime} \rightarrow a<x>\prec P^{\prime} \wedge\left(P^{\prime}, R^{\prime}[x::=u]\right) \in \operatorname{Rel}^{\prime \prime}\)
proof(rule allI)
fix \(u\)
from L2 obtain \(P^{\prime}\) where PTrans: \(P \Longrightarrow_{l} u\) in \(P^{\prime \prime} \rightarrow a<x>\prec P^{\prime}\) and \(P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}[x::=u]\right) \in \operatorname{Rel}\)
by blast
moreover from \(Q^{\prime}\) Der have \(\left(Q^{\prime}[x::=u], R^{\prime}[x::=u]\right) \in\) Rel \(^{\prime}\) by \((\operatorname{simp}\) add:
derivative-def)
ultimately show \(\exists P^{\prime} . P \Longrightarrow_{l} u\) in \(P^{\prime \prime} \rightarrow a<x>\prec P^{\prime} \wedge\left(P^{\prime}, R^{\prime}[x::=u]\right) \in \operatorname{Rel}^{\prime \prime}\) using Trans by blast
qed
thus?case by force
next
case \(\left(\right.\) Free \(R^{\prime} \alpha\) )
have RTrans: \(R \longmapsto \alpha \prec R^{\prime}\) by fact
with \(Q\) SimR obtain \(Q^{\prime}\) where \(Q\) Trans: \(Q \longmapsto \alpha \prec Q^{\prime}\) and \(Q^{\prime} \operatorname{RelR}^{\prime}:\left(Q^{\prime}, R^{\prime}\right)\) \(\in\) Rel \(^{\prime}\)
by (blast dest: Strong-Late-Sim.simE)
from \(P \operatorname{Sim} Q\) QTrans have \(\exists P^{\prime} . P \Longrightarrow \Longrightarrow_{l} \alpha \prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in\) Rel by (blast intro: simE)
then obtain \(P^{\prime}\) where PTrans: \(P \Longrightarrow{ }_{l}{ }^{\wedge} \alpha \prec P^{\prime}\) and \(P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}\) by blast
from \(P^{\prime}\) RelQ \(Q^{\prime} Q^{\prime}\) RelR \({ }^{\prime}\) Trans have \(\left(P^{\prime}, R^{\prime}\right) \in\) Rel \({ }^{\prime \prime}\) by blast
with PTrans show ?case by blast
qed
qed
end
theory Weak-Late-Bisim
imports Weak-Late-Sim Strong-Late-Bisim
begin
lemma monoAux: \(A \subseteq B \Longrightarrow P \rightsquigarrow \wedge<A>Q \longrightarrow P \rightsquigarrow \wedge<B>Q\)
by (auto intro: Weak-Late-Sim.monotonic)
coinductive-set weakBisim :: \((p i \times p i)\) set
where
step: \(\llbracket P \rightsquigarrow<\) weakBisim \(>Q ;(Q, P) \in\) weakBisim \(\rrbracket \Longrightarrow(P, Q) \in\) weakBisim monos monoAux
```

abbreviation
weakBisimJudge (infixr }\approx65)\mathrm{ where }P\approxQ\equiv(P,Q)\in\mathrm{ weakBisim
lemma weakBisimCoinductAux[case-names weakBisim, case-conclusion weakBisim
step, consumes 1]:
assumes p:(P,Q)\inX
and step: }\bigwedgePQ.(P,Q)\inX\LongrightarrowP\rightsquigarrow^<(X\cup\mathrm{ weakBisim )> Q^((Q,P) \&X
\vee Q \approxP)
shows P}\approx
proof -
have aux: X \cup weakBisim}={(P,Q).(P,Q)\inX\veeP\approxQ} by blas
from p show ?thesis
by(coinduct, force dest: step simp add: aux)
qed
lemma weakBisimCoinduct[consumes 1, case-names cSim cSym]:
fixes P :: pi
and }Q:: p
assumes }(P,Q)\in
and }\quad\PQ.(P,Q)\inX\LongrightarrowP\rightsquigarrow`<(X\cup\mathrm{ weakBisim )}>
and}\quad\bigwedgePQ.(P,Q)\inX\Longrightarrow(Q,P)\in
shows P}\approx
using assms
by(coinduct rule: weakBisimCoinductAux) auto
lemma weak-coinduct[case-names weakBisim, case-conclusion weakBisim step, con-
sumes 1]:
assumes p: (P,Q) \inX
and step: }\PQ.(P,Q)\inX\LongrightarrowP\rightsquigarrow^<X>Q\wedge(Q,P)\in
shows P}\approx
using p
proof(coinduct rule: weakBisimCoinductAux)
case (weakBisim P Q)
from step[OF this] show ?case using Weak-Late-Sim.monotonic by blast
qed
lemma weakBisim WeakCoinduct[consumes 1, case-names cSim cSym]:
fixes P :: pi
and }Q::p
assumes }(P,Q)\in
and}\quad\PQ.(P,Q)\inX\LongrightarrowP\rightsquigarrow^<X>Q
and}\quad\PQ.(P,Q)\inX\Longrightarrow(Q,P)\in

```
```

    shows P\approxQ
    using assms
by(coinduct rule: weak-coinduct) auto
lemma monotonic:mono(\lambdap x1 x2. \existsP Q. x1 = P ^ x2 = Q ^P\rightsquigarrow < <{(xa,x).
pxax}>Q \ Q\rightsquigarrow<{(xa,x).p xa x}>P)
by(auto intro: monoI Weak-Late-Sim.monotonic)
lemma unfoldE:
fixes P :: pi
and }Q::p
assumes P}\approx
shows P\rightsquigarrow^<weakBisim> Q
and }Q\approx
using assms
by(auto intro: weakBisim.cases)
lemma unfoldI:
fixes P :: pi
and }Q:: p
assumes P\rightsquigarrow^ <weakBisim>Q
and }Q\approx
shows P\approxQ
using assms
by(auto intro: weakBisim.cases)
lemma eqvt:
shows eqvt weakBisim
proof(auto simp add: eqvt-def)
let ?X = {x.\existsPQ (perm::name prm). P\approxQ\wedgex=(perm • P, perm •Q)}
fix PQ
fix perm::name prm
assume PBiSimQ: P}\approx
hence (perm • P, perm • Q) \in ?X by blast
moreover have }\bigwedgePQ perm::name prm. \llbracketP\rightsquigarrow^<weakBisim> Q\rrbracket\Longrightarrow(perm •
P) \rightsquigarrow`<<?X> (perm •Q)
proof -
fix PQ
fix perm::name prm
assume P\rightsquigarrow<<weakBisim> Q
moreover have weakBisim \subseteq?X
proof(auto)
fix PQ

```
```

    assume P\approxQ
    moreover have P=([]::name prm) P P and Q = ([]::name prm) · Q by
    auto
ultimately show }\exists\mp@subsup{P}{}{\prime}\mp@subsup{Q}{}{\prime}.\mp@subsup{P}{}{\prime}\approx\mp@subsup{Q}{}{\prime}\wedge(\exists(\mathrm{ perm::name prm ). P = perm }\cdot\mp@subsup{P}{}{\prime
\wedge Q = perm • Q )
by blast
qed
moreover have eqvt ?X
proof (auto simp add: eqvt-def)
fix PQ
fix perm1::name prm
fix perm2::name prm
assume P}\approx
moreover have perm1 • perm2 . P = (perm1 @ perm2) . P by(simp add:
pt2[OF pt-name-inst])
moreover have perm1 • perm2 • Q = (perm1 @ perm2) • Q by(simp add:
pt2[OF pt-name-inst])
ultimately show $\exists P^{\prime} Q^{\prime} . P^{\prime} \approx Q^{\prime} \wedge(\exists($ perm::name prm $)$. perm1 - perm2 - $P=\operatorname{perm} \cdot P^{\prime} \wedge$

$$
\text { perm1 } \left.\cdot \text { perm2 } \cdot Q=\operatorname{perm} \cdot Q^{\prime}\right)
$$

by blast
qed
ultimately show (perm • P) \rightsquigarrow <?X> (perm •Q)
by(rule Weak-Late-Sim.eqvtI)
qed
ultimately show (perm • P) \approx (perm • Q) by (coinduct rule: weak-coinduct,
blast dest: unfoldE)
qed
lemma eqvtI:
fixes P :: pi
and }Q:: p
and perm :: name prm
assumes P}\approx
shows (perm \cdot P) \approx (perm •Q)
using assms
by(rule eqvtRelI[OF eqvt])
lemma weakBisimEqvt[simp]:
shows eqvt weakBisim
by(auto simp add: eqvt-def eqvtI)

```
```

lemma strongBisimWeakBisim:
fixes }P::p
and }Q::p
assumes PSimQ: P ~ Q
shows P}\approx
proof -
have }\PQ.P\rightsquigarrow[bisim] Q\LongrightarrowP\rightsquigarrow^<(bisim \cup weakBisim )>Q
proof -
fix PQ
assume P\rightsquigarrow[bisim] Q
hence P\rightsquigarrow`<bisim> Q by(rule strongSimWeakSim)         thus P\rightsquigarrow`<(bisim \cup weakBisim )}>
by(blast intro:Weak-Late-Sim.monotonic)
qed
with PSimQ show ?thesis
by(coinduct rule: weakBisimCoinductAux, force dest: Strong-Late-Bisim.bisimE
symmetric)
qed
lemma reflexive:
fixes P :: pi
shows P\approxP
proof -
have (P,P)\inId by simp
then show ?thesis
proof (coinduct rule: weak-coinduct)
case (weakBisim P Q)
have }(P,Q)\inId by fac
thus ?case by(auto intro: Weak-Late-Sim.reflexive)
qed
qed
lemma symmetric:
fixes P :: pi
and }Q::p
assumes P}\approx
shows Q
using assms
by(auto dest: unfoldE intro: unfoldI)
lemma transitive:
fixes P :: pi

```
\[
\begin{array}{cc}
\text { and } & Q:: p i \\
\text { and } & R:: p i
\end{array}
\]
    assumes \(\operatorname{PBiSimQ} Q \approx Q\)
    and \(\quad\) QBiSimR: \(Q \approx R\)
    shows \(P \approx R\)
proof -
    let ? \(X=\) weakBisim \(O\) weakBisim
    from assms have \((P, R) \in ? X\) by blast
    moreover have \(\bigwedge P Q R . \llbracket Q \rightsquigarrow<\) weakBisim> \(; P ; P \approx \Longrightarrow\)
                        \(P \rightsquigarrow \ll(? X \cup\) weakBisim \()>R\)
    proof -
        fix \(P Q R\)
        assume \(P\) BiSim \(Q: P \approx Q\)
    assume \(Q \rightsquigarrow \wedge<\) weakBisim> \(R\)
    moreover have eqvt weakBisim by (rule eqvt)
        moreover from eqvt have eqvt (?X \(\cup\) weakBisim) by (auto simp add: eqvt-
Trans)
    moreover have weakBisim \(O\) weakBisim \(\subseteq\) ? \(X \cup\) weakBisim by auto
    moreover have \(\bigwedge P Q . P \approx Q \Longrightarrow P \leadsto \wedge\) weakBisim \(>Q\) by (rule unfoldE)
    ultimately show \(P \rightsquigarrow^{\wedge}<(? X \cup\) weakBisim \()>R\) using \(P\) BiSim \(Q\)
        by(rule Weak-Late-Sim.transitive)
    qed
    ultimately show ?thesis
    apply(coinduct rule: weakBisimCoinduct, auto)
    by(blast dest: unfoldE symmetric)+
qed
lemma transitive-coinduct-weak[case-names WeakBisimEarly, case-conclusion Weak-
BisimEarly step, consumes 2]:
    assumes \(p:(P, Q) \in X\)
    and Eqvt: eqvt \(X\)
    and step: \(\wedge P Q .(P, Q) \in X \Longrightarrow P \rightsquigarrow \wedge<(\operatorname{bisim} O X O\) bisim \()>Q \wedge(Q, P) \in\)
X
```

    shows \(P \approx Q\)
    proof -
let $? \mathrm{X}=\operatorname{bisim} O X O$ bisim

```
    have Sim: \(\bigwedge P P^{\prime} Q^{\prime} Q . \llbracket P \sim P^{\prime} ; P^{\prime} \rightsquigarrow \wedge<? X>Q^{\prime} ; Q^{\prime} \rightsquigarrow[b i s i m] Q \rrbracket \Longrightarrow\)
                                    \(P \rightsquigarrow \wedge<? X>Q\)
    proof -
        fix \(P P^{\prime} Q^{\prime} Q\)
        assume \(P\) BisimP \(P^{\prime}: P \sim P^{\prime}\)
    assume \(P^{\prime} \operatorname{Sim} Q^{\prime}: P^{\prime} \rightsquigarrow \wedge<? X>Q^{\prime}\)
```

    assume Q'SimQ: Q'\rightsquigarrow[bisim] Q
    show P\rightsquigarrow^<? X>>Q
    proof -
        have }\mp@subsup{P}{}{\prime}\rightsquigarrow`<? \X>Q
        proof -
            have ?X O bisim \subseteq?X by(blast intro: Strong-Late-Bisim.transitive)
            moreover from Strong-Late-Bisim.bisimEqvt Eqvt have eqvt?X by blast
            ultimately show ?thesis using P'SimQ' Q'SimQ by(blast intro: strongAp-
    pend)
qed
moreover have eqvt bisim by(rule Strong-Late-Bisim.bisimEqvt)
moreover from Strong-Late-Bisim.bisimEqvt Eqvt have eqvt?X by blast
moreover have bisim O?X\subseteq?X by(blast intro: Strong-Late-Bisim.transitive)
moreover have }\PQ.P~Q\LongrightarrowP\rightsquigarrow^<bisim>Q by(blast dest: Strong-Late-Bisim.bisimE
strongSimWeakSim)
ultimately show ?thesis using PBisimP' by(rule Weak-Late-Sim.transitive)
qed
qed
from p have (P,Q)\in?X by(blast intro: Strong-Late-Bisim.reflexive)
moreover from step Sim have }\PQ.(P,Q)\in?X\LongrightarrowP\rightsquigarrow<?X>Q\wedge(Q
P) \in?X
by(blast dest: Strong-Late-Bisim.bisimE Strong-Late-Bisim.symmetric)
ultimately show ?thesis by(rule weak-coinduct)
qed
lemma weakBisimTransitiveCoinduct[case-names cSim cSym, consumes 2]:
assumes p:(P,Q)\inX
and Eqvt: eqvt X
and rSim: \bigwedgePQ.(P,Q) \inX\LongrightarrowP\rightsquigarrow < (bisim O X O bisim )> Q
and rSym: \PQ.(P,Q)\inX\Longrightarrow(Q,P)\inX
shows P}\approx
using assms
by(coinduct rule: transitive-coinduct-weak) auto
end
theory Weak-Late-Step-Sim
imports Weak-Late-Step-Semantics Weak-Late-Sim Strong-Late-Sim
begin

```

```

bool where
weakStepSimAct P Rs C Rel \equiv(\forall Q' a x. Rs =a<\nux>}\prec\mp@subsup{Q}{}{\prime}\longrightarrowx\sharpC
(\exists\mp@subsup{P}{}{\prime}.P\Longrightarrow\mp@subsup{}{l}{}a<\nux><< \mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\inRel))\wedge

```

\[
\begin{aligned}
& \left.\left.P \Longrightarrow_{l} u \text { in } P^{\prime \prime} \rightarrow a<x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}[x::=u]\right) \in \operatorname{Rel}\right)\right) \wedge \\
& \\
& \text { Rel }))
\end{aligned}
\]
definition weakStepSimAux :: pi \(\Rightarrow(p i \times p i)\) set \(\Rightarrow p i \Rightarrow\) bool where
\[
\text { weakStepSimAux } P \text { Rel } Q \equiv\left(\forall Q^{\prime} a x .\left(Q \longmapsto a<\nu x>\prec Q^{\prime} \wedge x \sharp P\right) \longrightarrow\left(\exists P^{\prime} .\right.\right.
\]
\[
\left.\left.P \Longrightarrow_{l} a<\nu x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}\right)\right) \wedge
\]
\[
\left(\forall Q^{\prime} a x .\left(Q \longmapsto a<x>\prec Q^{\prime} \wedge x \sharp P\right) \longrightarrow\left(\exists P^{\prime \prime} . \forall u . \exists P^{\prime} .\right.\right.
\]
\[
\left.\left.P \Longrightarrow_{l} u \text { in } P^{\prime \prime} \rightarrow a<x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}[x::=u]\right) \in \operatorname{Re} l\right)\right) \wedge
\]
\[
\left(\forall Q^{\prime} \alpha . Q \longmapsto \alpha \prec Q^{\prime} \longrightarrow\left(\exists P^{\prime} . P \Longrightarrow_{l} \alpha \prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right)\right.\right.
\]
\[
\in R e l))
\]
definition weakStepSim :: pi \(\Rightarrow(\) pi \(\times\) pi) set \(\Rightarrow\) pi \(\Rightarrow\) bool \((-\rightsquigarrow<->-[80,80\), 80] 80) where
\(P \rightsquigarrow<\) Rel \(>Q \equiv(\forall R s . Q \longmapsto R s \longrightarrow\) weakStepSimAct P Rs P Rel \()\)
lemmas weakStepSimDef \(=\) weakStepSimAct-def weakStepSim-def
lemma weakStepSimAux P Rel \(Q=\) weakStepSim P Rel \(Q\)
by (auto simp add: weakStepSimDef weakStepSimAux-def)
lemma monotonic:
fixes \(A::(p i \times p i)\) set
and \(B::(p i \times p i)\) set
and \(P:: p i\)
and \(P^{\prime}:: p i\)
assumes \(P \rightsquigarrow<A>P^{\prime}\)
and \(\quad A \subseteq B\)
shows \(P \rightsquigarrow<B>P^{\prime}\)
using assms
apply (auto simp add: weakStepSimDef)
apply blast
apply (erule-tac \(x=a<x>\prec Q^{\prime}\) in allE \()\)
apply (clarsimp)
\(\operatorname{apply}(\) rotate-tac 4)
\(\operatorname{apply}\left(\right.\) erule-tac \(x=Q^{\prime}\) in allE)
apply (erule-tac \(x=a\) in allE)
apply (erule-tac \(x=x\) in allE)
by blast+
lemma simCasesCont[consumes 1, case-names Bound Input Free]:
fixes \(P\) :: \(p i\)
and \(Q:: p i\)
and Rel :: \((p i \times p i)\) set
and \(C\) :: ' \(a:: f s\)-name
assumes Eqvt: equt Rel
and Bound: \(\bigwedge Q^{\prime}\) a \(x . \llbracket x \sharp C ; Q \longmapsto a<\nu x>\prec Q^{\top} \Longrightarrow \exists P^{\prime} . P \Longrightarrow_{l} a<\nu x>\)
```

$\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}$
and Input: $\bigwedge Q^{\prime}$ a $x . \llbracket x \sharp C ; Q \longmapsto a<x>\prec Q \rrbracket \Longrightarrow \exists P^{\prime \prime} . \forall u . \exists P^{\prime} . P \Longrightarrow{ }_{l} u$
in $P^{\prime \prime} \rightarrow a<x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}[x::=u]\right) \in$ Rel
and Free: $\bigwedge Q^{\prime} \alpha . Q \longmapsto \alpha \prec Q^{\prime} \Longrightarrow\left(\exists P^{\prime} . P \Longrightarrow_{l} \alpha \prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in\right.$ Rel $)$

```
    shows \(P \rightsquigarrow<\) Rel \(>Q\)
using Free
proof (auto simp add: weakStepSimDef)
    fix \(Q^{\prime} a x\)
    assume \(x\) Fresh \(P:(x::\) name \() \sharp P\)
    assume Trans: \(Q \longmapsto a<\nu x>\prec Q^{\prime}\)
    have \(\exists c:: n a m e . c \sharp\left(P, Q^{\prime}, x, C\right)\) by (blast intro: name-exists-fresh)
    then obtain \(c::\) name where \(c\) Fresh \(P: c \sharp P\) and cFresh \(Q^{\prime}: c \sharp Q^{\prime}\) and \(c F r e s h C\) :
\(c \sharp C\)
                                    and cineqx: \(c \neq x\)
    by(force simp add: fresh-prod)
    from Trans cFresh \(Q^{\prime}\) have \(Q \longmapsto a<\nu c>\prec\left([(x, c)] \cdot Q^{\prime}\right) \mathbf{b y}(\) simp add: al-
phaBoundResidual)
    with \(c\) Fresh \(C\) have \(\exists P^{\prime} . P \Longrightarrow_{l} a<\nu c>\prec P^{\prime} \wedge\left(P^{\prime},[(x, c)] \cdot Q^{\prime}\right) \in\) Rel
        by(rule Bound)
    then obtain \(P^{\prime}\) where PTrans: \(P \Longrightarrow_{l} a<\nu c>\prec P^{\prime}\) and \(P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime},[(x, c)]\right.\)
- \(\left.Q^{\prime}\right) \in\) Rel
        by blast
    from PTrans \(x\) FreshP cineqx have \(x F r e s h P^{\prime}: x \sharp P^{\prime}\) by (force dest: Weak-Late-Step-Semantics.freshTransition
    with PTrans have \(P \Longrightarrow_{l} a<\nu x>\prec\left([(x, c)] \cdot P^{\prime}\right)\) by \((\operatorname{simp}\) add: alphaBound-
Residual name-swap)
    moreover have \(\left([(x, c)] \cdot P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}\) (is ?goal)
    proof -
        from Eqvt \(P^{\prime} \operatorname{Rel} Q^{\prime}\) have \(\left([(x, c)] \cdot P^{\prime},[(x, c)] \cdot[(x, c)] \cdot Q^{\prime}\right) \in \operatorname{Rel}\)
            by(rule eqvtRelI)
    with cineqx show ? goal by (simp add: name-calc)
    qed
    ultimately show \(\exists P^{\prime} . P \Longrightarrow_{l} a<\nu x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in\) Rel by blast
next
    fix \(Q^{\prime}\) a \(x u\)
    assume \(Q\) Trans: \(Q \longmapsto a<x>\prec\left(Q^{\prime}:: p i\right)\)
        and xFreshP: \(x \sharp P\)
    have \(\exists c:: n a m e . c \sharp\left(P, Q^{\prime}, C, x\right) \mathbf{b y}(\) blast intro: name-exists-fresh)
    then obtain \(c:: n a m e\) where \(c\) Fresh \(P: c \sharp P\) and \(c F r e s h ~ Q^{\prime}: c \sharp Q^{\prime}\) and \(c F r e s h C\) :
\(c \sharp C\)
                        and cineqx: \(c \neq x\)
    by (force simp add: fresh-prod)
from \(Q\) Trans cFresh \(Q^{\prime}\) have \(Q \longmapsto a<c>\prec\left([(x, c)] \cdot Q^{\prime}\right)\) by (simp add: alphaBoundResidual)
with \(c\) Fresh \(C\) have \(\exists P^{\prime \prime} . \forall u . \exists P^{\prime} . P \Longrightarrow{ }_{l} u\) in \(P^{\prime \prime} \rightarrow a<c>\prec P^{\prime} \wedge\left(P^{\prime},([(x, c)]\right.\)
- \(\left.\left.Q^{\prime}\right)[c::=u]\right) \in \operatorname{Rel}\)
by (rule Input)
then obtain \(P^{\prime \prime}\) where \(L 1: \forall u . \exists P^{\prime} . P \Longrightarrow_{l} u\) in \(P^{\prime \prime} \rightarrow a<c>\prec P^{\prime} \wedge\left(P^{\prime},([(x\right.\), \(\left.\left.c)] \cdot Q^{\prime}\right)[c::=u]\right) \in\) Rel by blast
have \(\forall u . \exists P^{\prime} . P \Longrightarrow_{l} u\) in \(\left([(c, x)] \cdot P^{\prime \prime}\right) \rightarrow a<x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}[x::=u]\right) \in\) Rel proof (auto)
fix \(u\)
from L1 obtain \(P^{\prime}\) where PTrans: \(P \Longrightarrow_{l} u\) in \(P^{\prime \prime} \rightarrow a<c>\prec P^{\prime}\) and \(P^{\prime}\) RelQ': \(\left(P^{\prime},\left([(x, c)] \cdot Q^{\prime}\right)[c::=u]\right) \in \operatorname{Rel}\)
by blast
from PTrans xFresh \(P\) have \(P \Longrightarrow_{l} u\) in \(\left([(c, x)] \cdot P^{\prime \prime}\right) \rightarrow a<x>\prec P^{\prime}\) by (rule alphaInput)
moreover from \(P^{\prime}\) Rel \(Q^{\prime}\) cFresh \(Q^{\prime}\) have \(\left(P^{\prime}, Q^{\prime}[x::=u]\right) \in\) Rel by \((\operatorname{simp}\) add: renaming[THEN sym] name-swap)
```

    ultimately show \(\exists P^{\prime} . P \Longrightarrow_{l} u\) in \(\left([(c, x)] \cdot P^{\prime \prime}\right) \rightarrow a<x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}[x::=u]\right)\)
    $\in$ Rel by blast
qed

```
thus \(\exists P^{\prime \prime} . \forall u . \exists P^{\prime} . P \Longrightarrow_{l} u\) in \(P^{\prime \prime} \rightarrow a<x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}[x::=u]\right) \in\) Rel by blast
qed
lemma simCases[consumes 0, case-names Bound Input Free]:
fixes \(P\) :: \(p i\)
and \(Q:: p i\)
and Rel :: \((p i \times p i)\) set
and \(C\) :: 'a::fs-name
assumes Bound: \(\bigwedge Q^{\prime}\) a \(x . \llbracket Q \longmapsto a<\nu x>\prec Q^{\prime} ; x \sharp P \rrbracket \Longrightarrow \exists P^{\prime} . P \Longrightarrow_{l} a<\nu x>\) \(\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}\)
and Input: \(\bigwedge Q^{\prime}\) a \(x . \llbracket Q \longmapsto a<x>\prec Q^{\prime} ; x \sharp P \rrbracket \Longrightarrow \exists P^{\prime \prime} . \forall u . \exists P^{\prime} . P \Longrightarrow{ }_{l} u\) in \(P^{\prime \prime} \rightarrow a<x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}[x::=u]\right) \in \operatorname{Rel}\)
and Free: \(\bigwedge Q^{\prime} \alpha . Q \longmapsto \alpha \prec Q^{\prime} \Longrightarrow\left(\exists P^{\prime} . P \Longrightarrow_{l} \alpha \prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in\right.\) Rel \()\)
shows \(P \rightsquigarrow<\) Rel \(>Q\)
using assms
by (auto simp add: weakStepSimDef)
lemma simActBoundCases[consumes 1, case-names Input BoundOutput]:
fixes \(P\) :: \(p i\)
and \(a\) :: subject
and \(x\) :: name
and \(Q^{\prime}:: p i\)
and \(C\) :: 'a::fs-name
and Rel \(::(p i \times p i)\) set
assumes EqvtRel: eqvt Rel
and DerInput: \(\bigwedge b . a=\) InputS \(b \Longrightarrow\left(\exists P^{\prime \prime} . \forall u . \exists P^{\prime} .\left(P \Longrightarrow_{l} u\right.\right.\) in \(P^{\prime \prime} \rightarrow b<x>\) \(\left.\prec P^{\prime}\right) \wedge\left(P^{\prime}, Q^{\prime}[x::=u]\right) \in\) Rel \()\)
and DerBoundOutput: \(\wedge b . a=\) BoundOutputS \(b \Longrightarrow\left(\exists P^{\prime} .\left(P \Longrightarrow_{l} b<\nu x>\right.\right.\)
\(\left.\left.\prec P^{\prime}\right) \wedge\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}\right)\)
shows weakStepSimAct \(P\left(a « x » \prec Q^{\prime}\right) P\) Rel
proof (simp add: weakStepSimAct-def fresh-prod, auto)
fix \(Q^{\prime \prime} b y\)
assume \(E q: a « x » \prec Q^{\prime}=b<\nu y>\prec Q^{\prime \prime}\)
assume \(y\) Fresh \(P: y \sharp P\)
from \(E q\) have \(a=\) BoundOutputS \(b\) by (simp add: residual.inject)
from \(y\) FreshP DerBoundOutput \(\left[\right.\) OF this] Eq show \(\exists P^{\prime} . P \Longrightarrow_{l} b<\nu y>\prec P^{\prime} \wedge\) \(\left(P^{\prime}, Q^{\prime \prime}\right) \in \operatorname{Rel}\)
proof (cases \(x=y\), auto simp add: residual.inject name-abs-eq)
fix \(P^{\prime}\)
assume PTrans: \(P \Longrightarrow_{l} b<\nu x>\prec P^{\prime}\)
assume \(P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime},\left([(x, y)] \cdot Q^{\prime \prime}\right)\right) \in \operatorname{Rel}\)
assume xineqy: \(x \neq y\)
with PTrans yFreshP have \(y\) Fresh \(P^{\prime}: y \sharp P^{\prime}\)
by (force intro: Weak-Late-Step-Semantics.freshTransition)
hence \(b<\nu x>\prec P^{\prime}=b<\nu y>\prec[(x, y)] \cdot P^{\prime} \mathbf{b y}(\) rule alphaBoundResidual)
moreover have \(\left([(x, y)] \cdot P^{\prime}, Q^{\prime \prime}\right) \in\) Rel
proof -
from EqvtRel \(P^{\prime} \operatorname{Rel} Q^{\prime}\) have \(\left([(x, y)] \cdot P^{\prime},[(x, y)] \cdot\left([(x, y)] \cdot Q^{\prime \prime}\right)\right) \in \operatorname{Rel}\) by (rule eqvtRelI)
thus ?thesis by (simp add: name-calc)
qed
ultimately show \(\exists P^{\prime} . P \Longrightarrow_{l} b<\nu y>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime \prime}\right) \in\) Rel using PTrans by auto
qed
next
fix \(Q^{\prime \prime} b y u\)
assume \(E q: a « x » \prec Q^{\prime}=b<y>\prec Q^{\prime \prime}\)
assume \(y\) Fresh \(P: y \sharp P\)
from \(E q\) have \(a=\) InputS \(b\) by (simp add: residual.inject)
from DerInput[OF this] obtain \(P^{\prime \prime}\) where L1: \(\forall u . \exists P^{\prime} . P \Longrightarrow_{l}\) u in \(P^{\prime \prime} \rightarrow b<x>\) \(\prec P^{\prime} \wedge\)
\[
\left(P^{\prime}, Q^{\prime}[x::=u]\right) \in \operatorname{Rel}
\]
by blast
have \(\forall u . \exists P^{\prime} . P \Longrightarrow_{l} u\) in \(\left([(x, y)] \cdot P^{\prime \prime}\right) \rightarrow b<y>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime \prime}[y::=u]\right) \in \operatorname{Rel}\) proof(rule allI)

\section*{fix \(u\)}
from \(L 1 E q\) show \(\exists P^{\prime} . P \Longrightarrow_{l} u\) in \(\left([(x, y)] \cdot P^{\prime \prime}\right) \rightarrow b<y>\prec P^{\prime} \wedge\left(P^{\prime}\right.\), \(\left.Q^{\prime \prime}[y::=u]\right) \in\) Rel
proof (cases \(x=y\), auto simp add: residual.inject name-abs-eq)
assume Der: \(\forall u . \exists P^{\prime} . P \Longrightarrow_{l} u\) in \(P^{\prime \prime} \rightarrow b<x>\prec P^{\prime} \wedge\left(P^{\prime},([(x, y)] \cdot\right.\) \(\left.\left.Q^{\prime \prime}\right)[x::=u]\right) \in \operatorname{Rel}\)
assume \(x \operatorname{Fresh} Q^{\prime \prime}: x \sharp Q^{\prime \prime}\)
from Der obtain \(P^{\prime}\) where PTrans: \(P \Longrightarrow{ }_{l} u\) in \(P^{\prime \prime} \rightarrow b<x>\prec P^{\prime}\) and \(P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime},\left([(x, y)] \cdot Q^{\prime \prime}\right)[x::=u]\right) \in \operatorname{Rel}\)
by force
from PTrans yFreshP have \(P \Longrightarrow_{l}\) u in \(\left([(x, y)] \cdot P^{\prime \prime}\right) \rightarrow b<y>\prec P^{\prime}\) by \((\) rule alphaInput)
moreover from \(x F r e s h Q^{\prime \prime} P^{\prime} \operatorname{Rel} Q^{\prime}\) have \(\left(P^{\prime}, Q^{\prime \prime}[y::=u]\right) \in \operatorname{Rel}\)
by (simp add: renaming)
ultimately show?thesis by force
qed
qed
thus \(\exists P^{\prime \prime} . \forall u . \exists P^{\prime} . P \Longrightarrow_{l} u\) in \(P^{\prime \prime} \rightarrow b<y>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime \prime}[y::=u]\right) \in\) Rel by blast
qed
lemma simActFreeCases[consumes 0, case-names Free]:
fixes \(P\) :: \(p i\)
and \(\alpha\) :: freeRes
and \(C\) ::' \(a:: f s\)-name
and Rel :: \((p i \times p i)\) set
assumes Der: \(\exists P^{\prime} .\left(P \Longrightarrow_{l} \alpha \prec P^{\prime}\right) \wedge\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}\)
shows weakStepSimAct \(P\left(\alpha \prec Q^{\prime}\right) P\) Rel
using assms
by (simp add: weakStepSimAct-def residual.inject)
lemma simE:
fixes \(P\) :: \(p i\)
and Rel :: \((p i \times p i)\) set
and \(Q:: p i\)
and \(a\) :: name
and \(x\) :: name
and \(u\) :: name
and \(Q^{\prime}:: p i\)
assumes \(P \rightsquigarrow<\operatorname{Rel}>Q\)
shows \(Q \longmapsto a<\nu x>\prec Q^{\prime} \Longrightarrow x \sharp P \Longrightarrow \exists P^{\prime} . P \Longrightarrow_{l} a<\nu x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right)\) \(\in\) Rel
```

    and }Q\longmapstoa<x>\prec\mp@subsup{Q}{}{\prime}\Longrightarrowx\sharpP\Longrightarrow\exists\mp@subsup{P}{}{\prime\prime}.\forallu.\exists\mp@subsup{P}{}{\prime}.P\Longrightarrow\mp@subsup{}{l}{}u\mathrm{ in }\mp@subsup{P}{}{\prime\prime}->a<x
    \prec P'^( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime}[x::=u])\inRe
and }Q\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime}\Longrightarrow(\exists\mp@subsup{P}{}{\prime}.P\Longrightarrow\mp@subsup{}{l}{}\alpha\prec\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\inRel
using assms by(simp add: weakStepSimDef)+
lemma weakSimTauChain:
fixes P :: pi
and Rel :: (pi\timespi) set
and }Q :: p
and }\mp@subsup{Q}{}{\prime}:: p
assumes QChain: Q \Longrightarrow}\mp@subsup{}{\tau}{}\mp@subsup{Q}{}{\prime
and PRelQ:}(P,Q)\in\operatorname{Rel
and Sim: }\bigwedgePQ.(P,Q)\in\operatorname{Rel}\LongrightarrowP\rightsquigarrow<\operatorname{Rel}>
shows }\exists\mp@subsup{P}{}{\prime}.P\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\operatorname{Rel
proof -
from QChain show ?thesis
proof(induct rule: tauChainInduct)
case id
have P\Longrightarrow}\mp@subsup{\Longrightarrow}{\tau}{}P\mathrm{ by simp
with PRelQ show ?case by blast
next
case(ih Q' (')
have }IH:\exists\mp@subsup{P}{}{\prime}.P\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mathrm{ Rel by fact
then obtain P' where PChain: P\Longrightarrow政 P' and P'RelQ':( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\operatorname{Rel}\mathrm{ by
blast
from P'RelQ' have }\mp@subsup{P}{}{\prime}\rightsquigarrow<\mathrm{ Rel }>\mp@subsup{Q}{}{\prime}\mathbf{by}(rule Sim)
moreover have Q'Trans: }\mp@subsup{Q}{}{\prime}\longmapsto\tau\prec\mp@subsup{Q}{}{\prime\prime}\mathrm{ by fact
ultimately have }\exists\mp@subsup{P}{}{\prime\prime}.\mp@subsup{P}{}{\prime}\Longrightarrow\mp@subsup{\Longrightarrow}{l}{}\tau\prec\mp@subsup{P}{}{\prime\prime}\wedge(\mp@subsup{P}{}{\prime\prime},\mp@subsup{Q}{}{\prime\prime})\in\operatorname{Rel}\mathrm{ by(rule simE)
then obtain }\mp@subsup{P}{}{\prime\prime}\mathrm{ where P'Trans: }\mp@subsup{P}{}{\prime}\Longrightarrow\mp@subsup{\Longrightarrow}{l}{}\tau\prec\mp@subsup{P}{}{\prime\prime}\mathrm{ and }\mp@subsup{P}{}{\prime\prime}Rel\mp@subsup{Q}{}{\prime\prime}:(\mp@subsup{P}{}{\prime\prime},\mp@subsup{Q}{}{\prime\prime})
Rel by blast
from P'Trans have P' }\mp@subsup{P}{\tau}{}\mp@subsup{P}{}{\prime\prime}\mathrm{ by(rule Weak-Late-Step-Semantics.tauTransitionChain)
with PChain have P \Longrightarrow>}\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime\prime}\mathrm{ by auto
with P'\primeRelQ'\prime show ?case by blast
qed
qed
lemma strongSimWeakEqSim:
fixes P :: pi
and }Q ::p
and Rel :: (pi\timespi) set
assumes PSimQ: P}\rightsquigarrow[Rel]
shows P}\rightsquigarrow<\mathrm{ Rel > Q
proof(auto simp add: weakStepSimDef)
fix }\mp@subsup{Q}{}{\prime}a
assume }Q\longmapstoa<\nux>\prec\mp@subsup{Q}{}{\prime}\mathrm{ and }x\sharp

```
with \(P \operatorname{Sim} Q\) have \(\exists P^{\prime} . P \longmapsto a<\nu x>\prec P^{\prime} \wedge\) derivative \(P^{\prime} Q^{\prime}\) (BoundOutputS
a) \(x \mathrm{Rel}\)
by (rule Strong-Late-Sim.simE)
then obtain \(P^{\prime}\) where PTrans: \(P \longmapsto a<\nu x>\prec P^{\prime}\) and \(P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in\) Rel
by(force simp add: derivative-def)
from PTrans have \(P \Longrightarrow_{l} a<\nu x>\prec^{\prime}\) by (rule Weak-Late-Step-Semantics.singleActionChain)
thus \(\exists P^{\prime} . P \Longrightarrow_{l} a<\nu x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in\) Rel using \(P^{\prime}\) RelQ \(Q^{\prime}\) by blast
next
fix \(Q^{\prime}\) a \(x u\)
assume \(Q \longmapsto a<x>\prec Q^{\prime}\) and \(x \sharp P\)
with \(P \operatorname{Sim} Q\) have L1: \(\exists P^{\prime} . P \longmapsto a<x>\prec P^{\prime} \wedge\) derivative \(P^{\prime} Q^{\prime}(\) InputS a) \(x\) Rel
by (blast intro: Strong-Late-Sim.simE)
then obtain \(P^{\prime}\) where PTrans: \(P \longmapsto a<x>\prec P^{\prime}\) and PDer: derivative \(P^{\prime} Q^{\prime}\) (InputS a) \(x\) Rel
by blast
have \(\forall u . \exists P^{\prime \prime} . P \Longrightarrow_{l} u\) in \(P^{\prime} \rightarrow a<x>\prec P^{\prime \prime} \wedge\left(P^{\prime \prime}, Q^{\prime}[x::=u]\right) \in \operatorname{Rel}\)
proof(rule allI)
fix \(u\)
from PTrans have \(P \Longrightarrow_{l} u\) in \(P^{\prime} \rightarrow a<x>\prec P^{\prime}[x::=u]\) by (blast intro: Weak-Late-Step-Semantics.singleActio moreover from \(P\) Der have \(\left(P^{\prime}[x::=u], Q^{\prime}[x::=u]\right) \in\) Rel by (force simp add:
derivative-def)
ultimately show \(\exists P^{\prime \prime} . P \Longrightarrow_{l}\) u in \(P^{\prime} \rightarrow a<x>\prec P^{\prime \prime} \wedge\left(P^{\prime \prime}, Q^{\prime}[x::=u]\right) \in\) Rel
by auto
qed
thus \(\exists P^{\prime \prime} . \forall u . \exists P^{\prime} . P \Longrightarrow_{l} u\) in \(P^{\prime \prime} \rightarrow a<x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}[x::=u]\right) \in\) Rel by
blast
next
fix \(Q^{\prime} \alpha\)
assume \(Q \longmapsto \alpha \prec Q^{\prime}\)
with \(P \operatorname{Sim} Q\) have \(\exists P^{\prime} . P \longmapsto \alpha \prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in\) Rel by (rule Strong-Late-Sim.simE)
then obtain \(P^{\prime}\) where PTrans: \(P \longmapsto \alpha \prec P^{\prime}\) and \(P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}\) by
blast
from PTrans have \(P \Longrightarrow{ }_{l} \alpha \prec P^{\prime}\) by (rule Weak-Late-Step-Semantics.singleActionChain)
thus \(\exists P^{\prime} . P \Longrightarrow{ }_{l} \alpha \prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in\) Rel using \(P^{\prime}\) RelQ \(Q^{\prime}\) by blast
qed
lemma weakSim WeakEqSim:
fixes \(P\) :: \(p i\)
and \(\quad Q \quad:: p i\)
and Rel :: \((p i \times p i)\) set
assumes \(P \rightsquigarrow<\operatorname{Rel}>Q\)
shows \(P \rightsquigarrow \wedge<\) Rel \(>Q\)
using assms
by (force simp add: weakStepSimDef simDef weakTransition-def)
lemma eqvtI:
\begin{tabular}{ll} 
fixes \(P\) & \(:: p i\) \\
and & \(Q \quad:: p i\) \\
and & Rel \(::(p i \times p i)\) set \\
and & perm
\end{tabular}\(::\) name prm
assumes Sim: \(P \rightsquigarrow<\) Rel \(>Q\)
and RelRel': Rel \(\subseteq\) Rel \(^{\prime}\)
and EqvtRel': eqvt Rel'
shows \((\) perm \(\cdot P) \rightsquigarrow<\) Rel \(^{\prime}>(\) perm \(\cdot Q)\)
using EqvtRel'
proof (induct rule: simCasesCont \([\) of - perm \(\cdot P]\) )
case(Bound \(Q^{\prime}\) a \(x\) )
have QTrans: \((\) perm \(\cdot Q) \longmapsto a<\nu x>\prec Q^{\prime}\) by fact
have \(x\) FreshP: \(x \sharp\) perm \(\cdot P\) by fact
from \(Q\) Trans have \((\) rev perm \(\cdot(\) perm \(\cdot Q)) \longmapsto\) rev perm \(\cdot\left(a<\nu x>\prec Q^{\prime}\right)\)
by (rule eqvts)
hence \(Q \longmapsto(\) rev perm \(\cdot a)<\nu(\) rev perm \(\cdot x)>\prec\left(\right.\) rev perm \(\left.\cdot Q^{\prime}\right)\)
by (simp add: name-rev-per)
moreover from \(x\) Fresh \(P\) have (rev perm \(\cdot x) \sharp P\) by (simp add: name-fresh-left)
ultimately obtain \(P^{\prime}\) where PTrans: \(P \Longrightarrow_{l}(\) rev perm \(\cdot a)<\nu(\) rev perm \(\cdot x)>\)
\(\prec P^{\prime}\)
and \(P^{\prime}\) RelQ \(Q^{\prime}:\left(P^{\prime}\right.\), rev perm \(\left.\cdot Q^{\prime}\right) \in\) Rel using Sim
by (blast dest: simE)
from PTrans have \((\) perm \(\cdot P) \Longrightarrow_{l}\) perm \(\cdot((\) rev perm \(\cdot a)<\nu(\) rev perm \(\cdot x)>\prec\) \(P^{\prime}\) )
by (rule Weak-Late-Step-Semantics.eqvtI)
hence \((\) perm \(\cdot P) \Longrightarrow_{l} a<\nu x>\prec\left(\right.\) perm \(\left.\cdot P^{\prime}\right)\) by (simp add: name-per-rev \()\)
moreover have \(\left(\right.\) perm \(\left.\cdot P^{\prime}, Q^{\prime}\right) \in\) Rel \(^{\prime}\)
proof -
from \(P^{\prime}\) Rel \(Q^{\prime}\) RelRel \({ }^{\prime}\) have ( \(P^{\prime}\), rev perm \(\cdot Q^{\prime}\) ) \(\in\) Rel' by blast
with EqvtRel' have \(\left(\right.\) perm \(\cdot P^{\prime}\), perm \(\cdot\left(\right.\) rev perm \(\left.\left.\cdot Q^{\prime}\right)\right) \in\) Rel \(^{\prime}\)
by (rule eqvtRelI)
thus ?thesis by (simp add: name-per-rev)
qed
ultimately show ?case by blast
next
case(Input \(Q^{\prime}\) a \(x\) )
have QTrans: \((\) perm \(\cdot Q) \longmapsto a<x>\prec Q^{\prime}\) by fact
have \(x\) FreshP: \(x \sharp\) perm \(\cdot P\) by fact
from \(Q\) Trans have \((\) rev perm \(\cdot(\) perm \(\cdot Q)) \longmapsto\) rev perm \(\cdot\left(a<x>\prec Q^{\prime}\right)\)
by (rule eqvts)
hence \(Q \longmapsto(\) rev perm \(\cdot a)<(\) rev perm \(\cdot x)>\prec\left(\right.\) rev perm \(\left.\cdot Q^{\prime}\right)\)
by (simp add: name-rev-per)
moreover from \(x\) Fresh \(P\) have \(x\) Fresh \(P:(\) rev perm \(\cdot x) \sharp P\) by (simp add: name-fresh-left)
ultimately obtain \(P^{\prime \prime}\)
where L1: \(\forall u . \exists P^{\prime} . P \Longrightarrow_{l} u\) in \(P^{\prime \prime} \rightarrow(\) rev perm \(\cdot a)<(\) rev perm \(\cdot x)>\prec P^{\prime} \wedge\) \(\left(P^{\prime},\left(\right.\right.\) rev perm \(\left.\cdot Q^{\prime}\right)[(\) rev perm \(\left.\cdot x)::=u]\right) \in\) Rel using Sim
by (blast dest: simE)
have \(\forall u . \exists P^{\prime}(\) perm \(\cdot P) \Longrightarrow l_{\text {u in }}\left(\right.\) perm \(\left.\cdot P^{\prime \prime}\right) \rightarrow a<x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}[x::=u]\right)\) \(\in R e l^{\prime}\)
proof(rule alli)
fix \(u\)
from \(L 1\) obtain \(P^{\prime}\) where PTrans: \(P \Longrightarrow_{l}(\) rev perm \(\cdot u)\) in \(P^{\prime \prime} \rightarrow(\) rev perm \(\cdot\) a) \(<(\) rev perm \(\cdot x)>\prec P^{\prime}\)
and \(P^{\prime}\) Rel \(Q^{\prime}:\left(P^{\prime},\left(\right.\right.\) rev perm \(\left.\cdot Q^{\prime}\right)[(\) rev perm \(\cdot x)::=(\) rev perm \(\cdot\)
\(u)]) \in\) Rel by blast
from PTrans have (perm • P) \(\Longrightarrow_{l}(\) perm \(\cdot(\) rev perm \(\cdot u)\) ) in (perm • \(\left.P^{\prime \prime}\right) \rightarrow(\) perm \(\cdot\) rev perm \(\cdot a)<(\) perm \(\cdot\) rev perm \(\cdot x)>\prec\left(\right.\) perm \(\left.\cdot P^{\prime}\right)\)
by(rule-tac Weak-Late-Step-Semantics.eqvtI, auto)
hence \((\) perm \(\cdot P) \Longrightarrow{ }_{l} u\) in \(\left(\right.\) perm \(\left.\left.\cdot P^{\prime \prime}\right) \rightarrow a<x\right\rangle \prec\left(\right.\) perm \(\left.\cdot P^{\prime}\right)\) by \((\) simp add: name-per-rev)
moreover have (perm • \(\left.P^{\prime}, Q^{\prime}[x::=u]\right) \in\) Rel \(^{\prime}\)
proof -
from \(P^{\prime}\) RelQ \(Q^{\prime}\) RelRel \(l^{\prime}\) have \(\left(P^{\prime},\left(\right.\right.\) rev perm \(\left.\cdot Q^{\prime}\right)[(\) rev perm \(\cdot x)::=(\) rev perm - u)]) \(\in\) Rel' by blast
with EqvtRel' have \(\left(\right.\) perm \(\cdot P^{\prime}\), perm \(\cdot\left(\left(\right.\right.\) rev perm \(\left.\cdot Q^{\prime}\right)[(\) rev perm \(\cdot x)::=(\) rev perm \(\cdot u)])) \in\) Rel \(^{\prime}\)
by(rule eqvtRelI)
thus ?thesis by (simp add: name-per-rev eqvt-subs[THEN sym] name-calc)
qed
ultimately show \(\exists P^{\prime} .(\) perm \(\cdot P) \Longrightarrow{ }_{l}\) in in \(\left(\right.\) perm \(\left.\left.\cdot P^{\prime}\right) \rightarrow a<x\right\rangle \prec P^{\prime} \wedge\left(P^{\prime}\right.\), \(\left.Q^{\prime}[x::=u]\right) \in\) Rel \(l^{\prime}\) by blast
qed
thus? case by blast
next
case (Free \(Q^{\prime} \alpha\) )
have \(Q\) Trans: \((\) perm \(\cdot Q) \longmapsto \alpha \prec Q^{\prime}\) by fact
from \(Q\) Trans have (rev perm \(\cdot(\) perm \(\cdot Q)) \longmapsto\) rev perm \(\cdot\left(\alpha \prec Q^{\prime}\right)\) by(rule eqvts)
hence \(Q \longmapsto(\) rev perm \(\cdot \alpha) \prec\left(\right.\) rev perm \(\left.\cdot Q^{\prime}\right)\)
by (simp add: name-rev-per)
with Sim obtain \(P^{\prime}\) where PTrans: \(P \Longrightarrow_{l}(\) rev perm \(\cdot \alpha) \prec P^{\prime}\) and PRel: \(\left(P^{\prime}\right.\),
\(\left(\right.\) rev perm \(\left.\left.\cdot Q^{\prime}\right)\right) \in \operatorname{Rel}\)
by (blast dest: simE)
from PTrans have \((\) perm \(\cdot P) \Longrightarrow{ }_{\iota}\) perm \(\cdot\left((\right.\) rev perm \(\left.\cdot \alpha) \prec P^{\prime}\right)\)
by(rule Weak-Late-Step-Semantics.eqvtI)
hence \((\) perm \(\cdot P) \Longrightarrow l \alpha \prec\left(\right.\) perm \(\left.\cdot P^{\prime}\right)\) by (simp add: name-per-rev)
moreover have \(\left(\left(\right.\right.\) perm \(\left.\left.\cdot P^{\prime}\right), Q^{\prime}\right) \in\) Rel \(^{\prime}\)
```

    proof -
    from PRel EqvtRel' RelRel' have ((perm • P'), (perm • (rev perm • Q'))) \in
    Rel'
by(force intro: eqvtRelI)
thus ?thesis by(simp add: name-per-rev)
qed
ultimately show ?case by blast
qed
lemma simE2:
fixes P :: pi
and Rel :: (pi\timespi) set
and }Q :: p
and a :: name
and }x\mathrm{ :: name
and}\mp@subsup{Q}{}{\prime}:: p
assumes PSimQ: P}\rightsquigarrow<\mathrm{ Rel }>
and Sim: \PQ.(P,Q) R Rel\LongrightarrowP\rightsquigarrow^ < Rel> Q
and Eqvt: eqvt Rel
and PRelQ: (P,Q) \in Rel
shows }Q\mp@subsup{\Longrightarrow}{l}{}a<\nux>\prec\mp@subsup{Q}{}{\prime}\Longrightarrowx\sharpP\Longrightarrow\exists\mp@subsup{P}{}{\prime}.P\mp@subsup{\Longrightarrow}{l}{l}a<\nux><\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime}
ERel
and }Q=\mp@subsup{}{l}{}\alpha\prec\mp@subsup{Q}{}{\prime}\Longrightarrow\exists\mp@subsup{P}{}{\prime}.P\Longrightarrow\mp@subsup{\Longrightarrow}{l}{}\alpha\prec\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\inRe
proof -
assume QTrans: Q \Longrightarrow>l }a<\nux>\prec\prec\mp@subsup{Q}{}{\prime
assume xFreshP: x\sharpP
have Goal: }\bigwedgePQ a x \mp@subsup{Q}{}{\prime}.\llbracketP\rightsquigarrow<Rel> Q;Q\Longrightarrow\Longrightarrowl a<\nux> \prec \mp@subsup{Q}{}{\prime};x\sharpP;x\sharpQ
(P,Q) \inRel\rrbracket\Longrightarrow
\exists\mp@subsup{P}{}{\prime}.P\Longrightarrow\mp@subsup{}{l}{}a<\nux><\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mathrm{ Rel}
proof -
fix PQ ax Q'
assume PSimQ: P}\rightsquigarrow<\mathrm{ Rel }>
assume QTrans: Q \Longrightarrow>l}\mp@subsup{l}{l}{}a<\nux>\prec\prec\mp@subsup{Q}{}{\prime
assume xFreshP: x\sharpP
assume xFreshQ: x\sharpQ
assume PRelQ: (P,Q)\inRel
from QTrans xFreshQ obtain }\mp@subsup{Q}{}{\prime\prime}\mp@subsup{Q}{}{\prime\prime\prime}\mathrm{ where QChain: }Q\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{Q}{}{\prime\prime
and \mp@subsup{Q}{}{\prime\prime}\mathrm{ Trans: }\mp@subsup{Q}{}{\prime\prime}\longmapstoa<\nux>}\prec\mp@subsup{Q}{}{\prime\prime\prime
and }\mp@subsup{Q}{}{\prime\prime\prime}\mathrm{ Chain: }\mp@subsup{Q}{}{\prime\prime\prime}\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{Q}{}{\prime
by(force dest: transitionE simp add: weakTransition-def)
from QChain PRelQ Sim have }\exists\mp@subsup{P}{}{\prime\prime}.P\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime\prime}\wedge(\mp@subsup{P}{}{\prime\prime},\mp@subsup{Q}{}{\prime\prime})\inRe
by(rule Weak-Late-Sim.weakSimTauChain)
then obtain }\mp@subsup{P}{}{\prime\prime}\mathrm{ where PChain: }P>\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime\prime}\mathrm{ and }\mp@subsup{P}{}{\prime\prime}Rel\mp@subsup{Q}{}{\prime\prime}:(\mp@subsup{P}{}{\prime\prime},\mp@subsup{Q}{}{\prime\prime})\in\operatorname{Rel
by blast
from PChain xFreshP have xFreshP': x \# P'' by(rule freshChain)

```
from \(P^{\prime \prime} \operatorname{Rel} Q^{\prime \prime}\) have \(P^{\prime \prime} \rightsquigarrow^{\wedge}<\operatorname{Rel}>Q^{\prime \prime}\) by (rule Sim)
hence \(\exists P^{\prime \prime \prime} . P^{\prime \prime} \Longrightarrow{ }^{\wedge} a<\nu x>\prec P^{\prime \prime \prime} \wedge\left(P^{\prime \prime \prime}, Q^{\prime \prime \prime}\right) \in\) Rel using \(Q^{\prime \prime}\) Trans \(x\) Fresh \({ }^{\prime \prime}\)
by(rule Weak-Late-Sim.simE)
then obtain \(P^{\prime \prime \prime}\) where \(P^{\prime \prime}\) Trans: \(P^{\prime \prime} \Longrightarrow_{l} a<\nu x>\prec P^{\prime \prime \prime}\) and \(P^{\prime \prime \prime} R e l Q^{\prime \prime \prime}\) : \(\left(P^{\prime \prime \prime}, Q^{\prime \prime \prime}\right) \in \operatorname{Rel}\)
by (force simp add: weakTransition-def)
have \(\exists P^{\prime} . P^{\prime \prime \prime} \Longrightarrow{ }_{\tau} P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in\) Rel using \(Q^{\prime \prime \prime}\) Chain \(P^{\prime \prime \prime}\) RelQ \({ }^{\prime \prime \prime}\) Sim
by(rule Weak-Late-Sim.weakSimTauChain)
then obtain \(P^{\prime}\) where \(P^{\prime \prime \prime}\) Chain: \(P^{\prime \prime \prime} \Longrightarrow{ }_{\tau} P^{\prime}\) and \(P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}\) by blast
from PChain \(P^{\prime \prime}\) Trans \(P^{\prime \prime \prime}\) Chain xFresh \(P^{\prime \prime}\) have \(P \Longrightarrow_{l} a<\nu x>\prec P^{\prime}\)
by (blast dest: Weak-Late-Step-Semantics.chainTransitionAppend)
with \(P^{\prime} R e l Q^{\prime}\) show \(\exists P^{\prime} . P \Longrightarrow_{l} a<\nu x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in\) Rel by blast qed
have \(\exists c\) ::name. \(c \sharp\left(Q, Q^{\prime}, P, x\right)\) by (blast intro: name-exists-fresh)
then obtain \(c::\) :name where \(c\) Fresh \(Q: c \sharp Q\) and cFresh \(Q^{\prime}: c \sharp Q^{\prime}\) and cFreshP: \(c \sharp P\)
and xineqc: \(x \neq c\)
by(force simp add: fresh-prod)
from \(Q\) Trans cFresh \(Q^{\prime}\) have \(Q \Longrightarrow_{l} a<\nu c>\prec\left([(x, c)] \cdot Q^{\prime}\right)\) by \((\) simp add: alphaBoundResidual)
with \(P \operatorname{Sim} Q\) have \(\exists P^{\prime} . P \Longrightarrow_{l} a<\nu c>\prec P^{\prime} \wedge\left(P^{\prime},[(x, c)] \cdot Q^{\prime}\right) \in\) Rel using cFreshP cFreshQ PRelQ
by (rule Goal)
then obtain \(P^{\prime}\) where PTrans: \(P \Longrightarrow_{l} a<\nu c>\prec P^{\prime}\) and \(P^{\prime} \operatorname{RelQ}^{\prime}:\left(P^{\prime},[(x, c)]\right.\)
- \(\left.Q^{\prime}\right) \in\) Rel
by force
have \(P \Longrightarrow_{l} a<\nu x>\prec\left([(x, c)] \cdot P^{\prime}\right)\)
proof -
from PTrans \(x\) FreshP xineqc have \(x \sharp P^{\prime}\) by (rule Weak-Late-Step-Semantics.freshTransition)
with PTrans show ?thesis by (simp add: alphaBoundResidual name-swap)
qed
moreover have \(\left([(x, c)] \cdot P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}\)
proof -
from Eqvt \(P^{\prime} \operatorname{Rel} Q^{\prime}\) have \(\left([(x, c)] \cdot P^{\prime},[(x, c)] \cdot[(x, c)] \cdot Q^{\prime}\right) \in \operatorname{Rel}\) by(rule eqvtRelI)
thus ?thesis by simp
qed
ultimately show \(\exists P^{\prime} . P \Longrightarrow_{l} a<\nu x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in\) Rel by blast next
assume \(Q\) Trans: \(Q \Longrightarrow_{l} \alpha \prec Q^{\prime}\)
```

then obtain Q" Q '/\prime where QChain: Q \Longrightarrow}\mp@subsup{}{\tau}{}\mp@subsup{Q}{}{\prime\prime
and }\mp@subsup{Q}{}{\prime\prime}\mathrm{ Trans: }\mp@subsup{Q}{}{\prime\prime}\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime\prime\prime
and Q '"'Chain: }\mp@subsup{Q}{}{\prime\prime\prime}\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{Q}{}{\prime
by(blast dest: transitionE)

```
```

thus }\exists\mp@subsup{P}{}{\prime}.P\Longrightarrow\mp@subsup{\Longrightarrow}{l}{}\alpha\prec\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mathrm{ Rel
proof(induct arbitrary:\alpha 嗸 (' 'rule: tauChainInduct)
case(id \alpha Q ''\prime)
from PSimQ<Q\longmapsto\alpha \prec Q ''\prime}>\mathrm{ have }\exists\mp@subsup{P}{}{\prime}.P\Longrightarrow\mp@subsup{\}{l}{}\alpha\prec\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime\prime\prime})\in\mathrm{ Rel
by(blast dest: simE)
then obtain P}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ where PTrans: P >}\mp@subsup{l}{\alpha}{}\alpha\prec\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ and P'RelQ ''':( }\mp@subsup{P}{}{\prime\prime\prime},\mp@subsup{Q}{}{\prime\prime\prime}
Rel

```
    by blast
    have \(\exists P^{\prime} . P^{\prime \prime \prime} \Longrightarrow_{\tau} P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}\) using \(\left\langle Q^{\prime \prime \prime} \Longrightarrow_{\tau} Q^{\prime}\right\rangle P^{\prime} \operatorname{Rel} Q^{\prime \prime \prime} \operatorname{Sim}\)
        by(rule Weak-Late-Sim.weakSimTauChain)
    then obtain \(P^{\prime}\) where \(P^{\prime \prime \prime}\) Chain: \(P^{\prime \prime \prime} \Longrightarrow_{\tau} P^{\prime}\) and \(P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}\)
by blast
    from \(P^{\prime \prime \prime}\) Chain PTrans have \(P \Longrightarrow_{l} \alpha \prec P^{\prime}\)
        by (blast dest: Weak-Late-Step-Semantics.chainTransitionAppend)
    with \(P^{\prime}\) RelQ' show ?case by blast
next
    case (ih \(Q^{\prime \prime \prime \prime} Q^{\prime \prime \prime} \alpha Q^{\prime \prime} Q^{\prime}\) )
    have \(Q^{\prime \prime \prime} \Longrightarrow_{\tau} Q^{\prime \prime \prime}\) by simp
    with \(\left\langle Q^{\prime \prime \prime \prime} \longmapsto \tau \prec Q^{\prime \prime \prime}\right\rangle\) obtain \(P^{\prime \prime \prime}\) where PTrans: \(P \Longrightarrow_{l} \tau \prec P^{\prime \prime \prime}\) and
\(P^{\prime \prime \prime} \operatorname{Rel} Q^{\prime \prime \prime}:\left(P^{\prime \prime \prime}, Q^{\prime \prime \prime}\right) \in \operatorname{Rel}\)
            by (drule-tac ih) auto
    from \(P^{\prime \prime \prime} R e l Q^{\prime \prime \prime}\left\langle Q^{\prime \prime \prime} \longmapsto \alpha \prec Q^{\prime \prime}\right\rangle\) obtain \(P^{\prime \prime}\) where
        \(P^{\prime \prime \prime}\) Trans: \(P^{\prime \prime \prime} \Longrightarrow{ }^{\prime} \alpha \prec P^{\prime \prime}\) and \(P^{\prime \prime} \operatorname{Rel} Q^{\prime \prime}:\left(P^{\prime \prime}, Q^{\prime \prime}\right) \in \operatorname{Rel}\)
        by (blast dest: Weak-Late-Sim.simE Sim)
    from \(P^{\prime \prime} \operatorname{Rel} Q^{\prime \prime}\left\langle Q^{\prime \prime} \Longrightarrow_{\tau} Q^{\prime}\right\rangle \operatorname{Sim}\) obtain \(P^{\prime}\) where
        \(P^{\prime \prime}\) Chain: \(P^{\prime \prime} \Longrightarrow_{\tau} P^{\prime}\) and \(P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}\)
        by (drule-tac Weak-Late-Sim.weakSimTauChain) auto
    from PTrans \(P^{\prime \prime \prime}\) Trans \(P^{\prime \prime}\) Chain have \(P \Longrightarrow_{l} \alpha \prec P^{\prime}\)
        apply (auto simp add: weakTransition-def residual.inject)
        apply (drule-tac Weak-Late-Step-Semantics.tauTransitionChain, auto)
        apply (drule-tac Weak-Late-Step-Semantics.chainTransitionAppend, simp)
        apply (rule Weak-Late-Step-Semantics.chainTransitionAppend, auto)
        by(drule-tac Weak-Late-Step-Semantics.chainTransitionAppend, auto)
    with \(\left\langle\left(P^{\prime}, Q^{\prime}\right) \in\right.\) Rel \(\rangle\) show ?case by blast
qed
qed
lemma reflexive:
```

    fixes P :: pi
    and Rel :: (pi\timespi) set
    assumes Id \subseteq Rel
    shows P}\rightsquigarrow<\mathrm{ Rel> P
    using assms
by(auto intro: Weak-Late-Step-Semantics.singleActionChain simp add: weakStep-
SimDef)
lemma transitive:
fixes P :: pi
and }Q\quad::p
and }R\quad::p
and Rel :: (pi\timespi) set
and Rel' :: (pi\timespi) set
and Rel" :: (pi\timespi) set
assumes PSimQ: P}\rightsquigarrow<\mathrm{ Rel }>
and QSimR:Q}\rightsquigarrow<\mp@subsup{\mathrm{ Rel'}}{}{\prime}>
and Eqvt: eqvt Rel
and Eqvt': eqvt Rel"
and Trans:Rel O Rel'}\subseteqRel"
and Sim: }\bigwedgePQ.(P,Q)\in\operatorname{Rel}\LongrightarrowP\rightsquigarrow^<Rel> Q
and PRelQ:(P,Q)\inRel
shows P\rightsquigarrow<Rel '/> R
using Eqvt'
proof(induct rule: simCasesCont[of - (P,Q)])
case(Bound R' a x)
have RTrans: R\longmapstoa<\nux> \prec R' by fact
have }x\sharp(P,Q)\mathrm{ by fact
hence xFreshP: }x\sharpP\mathrm{ and xFreshQ: x \#Q by(simp add: fresh-prod)+
from QSimR RTrans xFreshQ obtain Q' where QTrans: Q \Longrightarrow>l }a<\nux><\prec\mp@subsup{Q}{}{\prime
and \mp@subsup{Q}{}{\prime}RelR':}(\mp@subsup{Q}{}{\prime},\mp@subsup{R}{}{\prime})\in\mp@subsup{R}{}{\prime}\mp@subsup{l}{}{\prime
by(blast dest: simE)
from PSimQ Sim Eqvt PRelQ QTrans xFreshP obtain P' where PTrans: P
\Longrightarrowl}a<\nux>\prec\mp@subsup{P}{}{\prime
and P'RelQ':}(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\operatorname{Rel
by(blast dest: simE2)
moreover from P'RelQ' Q'RelR' Trans have ( }\mp@subsup{P}{}{\prime},R=R)\inRe\mp@subsup{l}{}{\prime\prime}\mathrm{ by blast
ultimately show ?case by blast
next
case(Input R' a x)
have RTrans: }R\longmapstoa<x>\prec\mp@subsup{R}{}{\prime}\mathrm{ by fact
have }x\sharp(P,Q)\mathrm{ by fact
hence xFreshP: }x\sharpP\mathrm{ and xFresh Q: x }\#Q\mathrm{ by(simp add: fresh-prod)+

```
from \(Q \operatorname{SimR}\) RTrans xFresh \(Q\) obtain \(Q^{\prime \prime}\)
where \(\forall u\). \(\exists Q^{\prime} . Q \Longrightarrow{ }_{l} u\) in \(Q^{\prime \prime} \rightarrow a<x>\prec Q^{\prime} \wedge\left(Q^{\prime}, R^{\prime}[x::=u]\right) \in \operatorname{Rel}^{\prime}\) \(\mathbf{b y}(\) blast dest: \(\operatorname{sim} E)\)
hence \(\exists Q^{\prime \prime \prime} . Q \Longrightarrow_{\tau} Q^{\prime \prime \prime} \wedge Q^{\prime \prime \prime} \longrightarrow a<x>\prec Q^{\prime \prime} \wedge\left(\forall u . \exists Q^{\prime} . Q^{\prime \prime}[x::=u] \Longrightarrow_{\tau} Q^{\prime}\right.\) \(\left.\wedge\left(Q^{\prime}, R^{\prime}[x::=u]\right) \in \operatorname{Rel}^{\prime}\right)\)
by (simp add: inputTransition-def, blast)
then obtain \(Q^{\prime \prime \prime}\) where \(Q\) Chain: \(Q \Longrightarrow_{\tau} Q^{\prime \prime \prime}\)
and \(Q^{\prime \prime \prime}\) Trans: \(Q^{\prime \prime \prime} \longmapsto a<x>\prec Q^{\prime \prime}\)
and L1: \(\forall u . \exists Q^{\prime} . Q^{\prime \prime}[x::=u] \Longrightarrow_{\tau} Q^{\prime} \wedge\left(Q^{\prime}, R^{\prime}[x::=u]\right) \in R e l^{\prime}\)
by blast
from QChain PRelQ Sim obtain \(P^{\prime \prime \prime}\) where PChain: \(P \Longrightarrow{ }_{\tau} P^{\prime \prime \prime}\) and \(P^{\prime \prime \prime} R e l Q^{\prime \prime \prime}\) : \(\left(P^{\prime \prime \prime}, Q^{\prime \prime \prime}\right) \in \operatorname{Rel}\)
by (drule-tac Weak-Late-Sim.weakSimTauChain) auto
from PChain \(x F r e s h P\) have \(x F r e s h P^{\prime \prime \prime}: x \sharp P^{\prime \prime \prime}\) by(rule freshChain)
from \(P^{\prime \prime \prime} \operatorname{Rel} Q^{\prime \prime \prime}\) have \(P^{\prime \prime \prime} \rightsquigarrow \wedge<\) Rel \(>Q^{\prime \prime \prime}\) by (rule Sim)
with xFreshP \(P^{\prime \prime \prime} Q^{\prime \prime \prime}\) Trans obtain \(P^{\prime \prime \prime \prime}\) where L2: \(\forall u . \exists P^{\prime \prime} . P^{\prime \prime \prime} \Longrightarrow_{l} u\) in \(P^{\prime \prime \prime \prime} \rightarrow a<x>\prec P^{\prime \prime} \wedge\left(P^{\prime \prime}, Q^{\prime \prime}[x::=u]\right) \in\) Rel
by (blast dest: Weak-Late-Sim.simE)
have \(\forall u\). \(\exists P^{\prime} Q^{\prime} . P \Longrightarrow_{l} u\) in \(P^{\prime \prime \prime \prime} \rightarrow a<x>\prec P^{\prime} \wedge\left(P^{\prime}, R^{\prime}[x::=u]\right) \in \operatorname{Rel}^{\prime \prime}\)
proof(rule allI)
fix \(u\)
from L1 obtain \(Q^{\prime}\) where \(Q^{\prime \prime}\) Chain: \(Q^{\prime \prime}[x::=u] \Longrightarrow_{\tau} Q^{\prime}\) and \(Q^{\prime} R^{R} R^{\prime}:\left(Q^{\prime}\right.\), \(\left.R^{\prime}[x::=u]\right) \in\) Rel \(^{\prime}\)
by blast
from L2 obtain \(P^{\prime \prime}\) where \(P^{\prime \prime \prime}\) Trans: \(P^{\prime \prime \prime} \Longrightarrow_{l} u\) in \(P^{\prime \prime \prime \prime} \rightarrow a<x>\prec P^{\prime \prime}\) and \(P^{\prime \prime} \operatorname{Rel} Q^{\prime \prime}:\left(P^{\prime \prime}, Q^{\prime \prime}[x::=u]\right) \in \operatorname{Rel}\)
by blast
from \(P^{\prime \prime} \operatorname{Rel} Q^{\prime \prime}\) have \(P^{\prime \prime} \rightsquigarrow^{\wedge}<\operatorname{Rel}>Q^{\prime \prime}[x::=u] \operatorname{by}(\) rule Sim \()\)
have \(\exists P^{\prime} . P^{\prime \prime} \Longrightarrow_{\tau} P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}\) using \(Q^{\prime \prime}\) Chain \(P^{\prime \prime}\) RelQ \({ }^{\prime \prime}\) Sim
by(rule Weak-Late-Sim.weakSimTauChain)
then obtain \(P^{\prime}\) where \(P^{\prime \prime}\) Chain: \(P^{\prime \prime} \Longrightarrow{ }_{\tau} P^{\prime}\) and \(P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}\) by blast
from PChain \(P^{\prime \prime \prime}\) Trans \(P^{\prime \prime}\) Chain have \(P \Longrightarrow_{l} u\) in \(P^{\prime \prime \prime \prime} \rightarrow a<x>\prec P^{\prime}\)
by (blast dest: Weak-Late-Step-Semantics.chainTransitionAppend)
moreover from \(P^{\prime} \operatorname{Rel} Q^{\prime} Q^{\prime} \operatorname{Rel} R^{\prime}\) have \(\left(P^{\prime}, R^{\prime}[x::=u]\right) \in\) Rel \(^{\prime \prime}\) by (insert Trans, auto)
ultimately show \(\exists P^{\prime} Q^{\prime} . P \Longrightarrow_{l}\) u in \(P^{\prime \prime \prime \prime} \rightarrow a<x>\prec P^{\prime} \wedge\left(P^{\prime}, R^{\prime}[x::=u]\right) \in\) \(R e l^{\prime \prime}\) by blast
qed
thus ?case by force
next
case (Free \(R^{\prime} \alpha\) )
have \(R\) Trans: \(R \longmapsto \alpha \prec R^{\prime}\) by fact
with \(Q \operatorname{SimR}\) obtain \(Q^{\prime}\) where \(Q\) Trans: \(Q \Longrightarrow_{l} \alpha \prec Q^{\prime}\) and \(Q^{\prime} \operatorname{RelR}^{\prime}:\left(Q^{\prime}, R^{\prime}\right)\) \(\in R e l^{\prime}\)
by (blast dest: simE)
from PSimQ Sim Eqvt PRelQ QTrans obtain \(P^{\prime}\) where PTrans: \(P \Longrightarrow_{l} \alpha \prec P^{\prime}\) and \(P^{\prime}\) RelQ \(Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}\)
by(blast dest: simE2)
```

    from P'RelQ' Q'RelR' Trans have ( }\mp@subsup{P}{}{\prime},\mp@subsup{R}{}{\prime})\inRe\mp@subsup{R}{}{\prime\prime}\mathrm{ by blast
    with PTrans show ?case by blast
    qed
end
theory Weak-Late-Cong
imports Weak-Late-Bisim Weak-Late-Step-Sim Strong-Late-Bisim
begin
definition congruence :: (pi\timespi) set where
congruence }\equiv{(P,Q)|PQ.P\rightsquigarrow<\mathrm{ weakBisim }>QQQQ\rightsquigarrow<\mathrm{ weakBisim>P}
abbreviation congruenceJudge (infixr }\simeq65)\mathrm{ where }P\simeqQ\equiv(P,Q)\in\mathrm{ congru-
ence
lemma unfoldE:
fixes P :: pi
and }Q::p
and s::(name }\times\mathrm{ name) list
assumes }P\simeq
shows P\rightsquigarrow<weakBisim> Q
and Q}\rightsquigarrow<\mathrm{ weakBisim>P
proof -
from assms show P \rightsquigarrow<weakBisim> Q by(force simp add: congruence-def)
next
from assms show Q}\rightsquigarrow<\mathrm{ weakBisim> P by(force simp add: congruence-def)
qed
lemma unfoldI:
fixes P :: pi
and }Q:: p
assumes P}\rightsquigarrow<\mathrm{ weakBisim> Q
and }Q\rightsquigarrow<\mathrm{ weakBisim>P
shows P\simeqQ
using assms by(force simp add: congruence-def)
lemma eqvt:
shows eqvt congruence
proof -
have }\bigwedgePQ(\mathrm{ perm::name prm). P }\rightsquigarrow<\mathrm{ weakBisim }>Q\Longrightarrow(\mathrm{ perm }\cdotP)\rightsquigarrow<\mathrm{ weakBisim }
(perm • Q)
proof -
fix P Q perm
assume P\rightsquigarrow<weakBisim>Q

```
```

    thus ((perm::name prm)}\cdotP)\rightsquigarrow<\mathrm{ weakBisim> (perm •Q)
        apply -
        by(blast intro:Weak-Late-Step-Sim.eqvtI Weak-Late-Bisim.eqvt)
    qed
    thus ?thesis
    by(simp add: congruence-def eqvt-def)
    qed
lemma eqvtI:
fixes P :: pi
and }Q:: p
and perm :: name prm
assumes P\simeqQ
shows (perm • P)\simeq (perm •Q)
using assms
by(rule eqvtRelI[OF eqvt])
lemma strongBisimWeakEq:
fixes P :: pi
and }Q::p
assumes P~Q
shows P\simeqQ
proof -
have }\PQ.P\rightsquigarrow[bisim] Q\LongrightarrowP\rightsquigarrow<\mathrm{ weakBisim > Q
proof -
fix PQ
assume P\rightsquigarrow[bisim] Q
hence P}\rightsquigarrow<\mathrm{ bisim > Q by(rule strongSimWeakEqSim)
moreover have bisim \subseteq weakBisim
by(auto intro: strongBisim WeakBisim)
ultimately show P}\rightsquigarrow<\mathrm{ weakBisim> Q by(rule Weak-Late-Step-Sim.monotonic)
qed
with assms show ?thesis
by(blast intro: unfoldI dest: Strong-Late-Bisim.bisimE Strong-Late-Bisim.symmetric)
qed
lemma congruenceWeakBisim:
fixes P :: pi
and }Q::p
assumes P\simeqQ
shows P\approxQ
proof -
let ?X = {(P,Q)|PQ.P\simeqQ}

```
```

    from assms have \((P, Q) \in ? X\) by auto
    thus ?thesis
    proof (coinduct rule: weakBisimCoinduct)
    case \((c \operatorname{Sim} P Q)\)
    \{
        fix \(P Q\)
        assume \(P \simeq Q\)
        hence \(P \rightsquigarrow<\) weakBisim \(>Q\) by (simp add: congruence-def)
    hence \(P \rightsquigarrow<(? X \cup\) weakBisim \()>Q\) by(rule-tac Weak-Late-Step-Sim.monotonic)
    auto
hence $P \rightsquigarrow<(? X \cup$ weakBisim $)>Q$ by (rule Weak-Late-Step-Sim.weakSimWeakEqSim)
\}
with $\langle(P, Q) \in$ ? $X\rangle$ show ? case by auto
next
case $(c S y m P Q)$
thus ?case by(auto simp add: congruence-def)
qed
qed
lemma congruenceSubsetWeakBisim:
shows congruence $\subseteq$ weakBisim
by(auto intro: congruenceWeakBisim)
lemma reflexive:
fixes $P$ :: $p i$
shows $P \simeq P$
proof -
from Weak-Late-Bisim.reflexive have $\wedge P . P \rightsquigarrow<$ weakBisim $>P$
by (blast intro: Weak-Late-Step-Sim.reflexive)
thus ?thesis
by (force simp add: substClosed-def congruence-def)
qed
lemma symetric:
fixes $P$ :: $p i$
and $\quad Q:: p i$
assumes $P \simeq Q$
shows $Q \simeq P$
using assms
by (force simp add: substClosed-def congruence-def)
lemma transitive:
fixes $P:: p i$
and $\quad Q:: p i$
and $\quad R:: p i$

```
```

    assumes P\simeqQ
    and }Q\simeq
    shows P\simeqR
    proof -
have Goal: \P Q R.\llbracketP\rightsquigarrow<weakBisim> Q;Q\rightsquigarrow<weakBisim> R; P\approxQ\rrbracket\Longrightarrow
P\rightsquigarrow<weakBisim> R
using Weak-Late-Bisim.eqvt Weak-Late-Bisim.unfoldE Weak-Late-Bisim.transitive
by(blast intro:Weak-Late-Step-Sim.transitive)
from assms show ?thesis
apply(simp add: congruence-def) using assms
by(blast intro: Goal dest: congruenceWeakBisim symetric)
qed
end
theory Weak-Late-Bisim-Subst
imports Weak-Late-Bisim Strong-Late-Bisim-Subst
begin
consts weakBisimSubst :: (pi \times pi) set
abbreviation
weakBisimSubstJudge (infixr }\mp@subsup{\approx}{}{s}65)\mathrm{ where }P\mp@subsup{\approx}{}{s}Q\equiv(P,Q)\in(\mathrm{ substClosed
weakBisim)
lemma congBisim:
fixes P :: pi
and }Q::p
assumes P}\mp@subsup{\approx}{}{s}
shows P}\approx
proof -
from assms substClosedSubset show ?thesis
by blast
qed
lemma strongBisimWeakBisim:
fixes P :: pi
and }Q::p
assumes P ~}\mp@subsup{~}{}{s}
shows P}\mp@subsup{\approx}{}{s}
using assms
by(auto simp add: substClosed-def intro: strongBisimWeakBisim)
lemma eqvt:
shows eqvt (substClosed weakBisim)

```
```

by(rule eqvtSubstClosed[OF Weak-Late-Bisim.eqvt])
lemma eqvtI:
fixes P :: pi
and }Q::p
and perm :: name prm
assumes P}\mp@subsup{\approx}{}{s}
shows (perm • P) *s}(\mathrm{ perm •Q)
using assms
by(rule-tac eqvtRelI[OF eqvt])
lemma reflexive:
fixes P :: pi
shows P }\mp@subsup{\approx}{}{s}
by(force simp add: substClosed-def intro:Weak-Late-Bisim.reflexive)
lemma symetric:
fixes P :: pi
and }Q::p
assumes P}\mp@subsup{\approx}{}{s}
shows Q 秥P
using assms
by(force simp add: substClosed-def intro:Weak-Late-Bisim.symmetric)
lemma transitive:
fixes P :: pi
and }Q::p
and }R::p
assumes P}\mp@subsup{\approx}{}{s}
and }Q\mp@subsup{\approx}{}{s}
shows P }\mp@subsup{\approx}{}{s}
using assms
by(force simp add: substClosed-def intro: Weak-Late-Bisim.transitive)
lemma partUnfold:
fixes P :: pi
and }Q::p
and s::(name }\times\mathrm{ name) list
assumes P}\mp@subsup{\approx}{}{s}

```
```

    shows P[<s>] \approxs }Q[<s>
    using assms
proof(auto simp add: substClosed-def)
fix s}\mp@subsup{s}{}{\prime
assume }\foralls.P[<s>]\approxQ[<s>
hence P[<(s@s')>]\approxQ[<(s@s')>] by blast
moreover have P[<(s@ s')>] =(P[<s>])[<\mp@subsup{s}{}{\prime}>]
by(induct s', auto)
moreover have Q[<(s@s')>]=(Q[<s>])[<\mp@subsup{s}{}{\prime}>]
by(induct s', auto)
ultimately show (P[<s>])[<\mp@subsup{s}{}{\prime}>] \approx(Q[<s>])[<\mp@subsup{s}{}{\prime}>]
by simp
qed
end
theory Weak-Late-Cong-Subst
imports Weak-Late-Cong Weak-Late-Bisim-Subst Strong-Late-Bisim-Subst
begin
definition congruenceSubst :: pi => pi=> bool (infixr }\mp@subsup{\simeq}{}{s}65)\mathrm{ where
P \simeqs}Q\equiv(P,Q)\in(\mathrm{ substClosed congruence)
lemmas congruenceSubstDef = congruenceSubst-def congruence-def substClosed-def
lemma unfoldE:
fixes P :: pi
and }Q::p
and s::(name }\times\mathrm{ name) list
assumes P \simeqs}
shows P[<s>] \rightsquigarrow<\mathrm{ weakBisim }>Q[<s>]
and }Q[<s>]\rightsquigarrow<\mathrm{ weakBisim>P[<s>]
proof -
from assms show P[<s>]\rightsquigarrow<weakBisim}>Q[<s>] by(force simp add: congru
enceSubstDef)
next
from assms show Q[<s>] \rightsquigarrow<\mathrm{ weakBisim }>PP[<s>] by(force simp add: congru-
enceSubstDef)
qed
lemma unfoldI:
fixes P :: pi
and }Q::p
assumes \foralls. P[<s>]\rightsquigarrow<\mathrm{ weakBisim }>Q[<s>]\wedgeQ[<s>] \rightsquigarrow<\mathrm{ weakBisim }>PP[<s>]

```
```

    shows P}\mp@subsup{\simeq}{}{s}
    proof -
from assms show ?thesis by(force simp add: congruenceSubstDef)
qed
lemma weakEqSubset:
shows substClosed congruence}\subseteq\mathrm{ weakBisim
proof(auto simp add: substClosed-def)
fix PQ
assume }\foralls.P[<s>]\simeqQ[<s>
hence P[<[]>]\simeqQ[<[]>] by blast
thus P}\approx
by(force dest: congruenceWeakBisim intro:Weak-Late-Bisim.unfoldI)
qed
lemma weakCongWeakEq:
fixes P :: pi
and }Q::p
assumes P}\mp@subsup{\simeq}{}{s}
shows P\simeqQ
using assms
apply(auto simp add: substClosed-def congruenceSubst-def)
apply(erule-tac x=[] in allE)
by auto
lemma eqvt:
shows eqvt (substClosed congruence)
by(rule eqvtSubstClosed[OF Weak-Late-Cong.eqvt])
lemma eqvtI:
fixes P :: pi
and }Q::p
and perm :: name prm
assumes P}\mp@subsup{\simeq}{}{s}
shows (perm • P) 工s}(\mathrm{ perm •Q)
using assms
by(simp add: congruenceSubst-def) (rule eqvtRelI[OF eqvt])
lemma strongEqWeakCong:
fixes P :: pi
and }Q::p
assumes P ~}\mp@subsup{~}{}{s}

```
```

    shows }P\mp@subsup{\simeq}{}{s}
    using assms
by(force intro: strongBisimWeakEq simp add: substClosed-def congruenceSubst-def)
lemma congSubstBisimSubst:
fixes P :: pi
and }Q::p
assumes P \simeq`s}
shows P}\mp@subsup{\approx}{}{s}
using assms
by(force simp add: congruenceSubst-def substClosed-def intro: congruenceWeak-
Bisim)
lemma reflexive:
fixes P :: pi
shows P}\mp@subsup{\simeq}{}{s}
proof -
from Weak-Late-Bisim.reflexive have \P.P\rightsquigarrow<weakBisim> P
by(blast intro:Weak-Late-Step-Sim.reflexive)
thus ?thesis
by(force simp add: congruenceSubstDef)
qed
lemma symetric:
fixes }P:: p
and }Q::p
assumes P}\mp@subsup{\simeq}{}{s}
shows Q \simeqs}
using assms
by(force simp add: congruenceSubstDef)
lemma transitive:
fixes P :: pi
and }Q::p
and }R::p
assumes P}\mp@subsup{\simeq}{}{s}
and }Q\mp@subsup{\simeq}{}{s}
shows P}\mp@subsup{\simeq}{}{s}
using assms
by(force simp add: congruenceSubst-def substClosed-def intro: Weak-Late-Cong.transitive)

```
```

lemma partUnfold:
fixes P :: pi
and }Q::p
and s :: (name }\times\mathrm{ name) list
assumes P}\mp@subsup{\simeq}{}{s}
shows P[<s>] 工s}Q[<s>
using assms
proof(auto simp add: congruenceSubst-def substClosed-def)
fix s}\mp@subsup{s}{}{\prime
assume }\foralls.(P[<s>],Q[<s>])\in congruenc
hence }(P[<(s@\mp@subsup{s}{}{\prime})>],Q[<(s@\mp@subsup{s}{}{\prime})>])\in\mathrm{ congruence by blast
moreover have P[<(s@ s')>] = (P[<s>])[<\mp@subsup{s}{}{\prime}\rangle]
by(induct s', auto)
moreover have Q[<(s@ s')>] =(Q[<s>])[<\mp@subsup{s}{}{\prime}>]
by(induct s', auto)
ultimately show }((P[<s>])[<\mp@subsup{s}{}{\prime}>],(Q[<s>])[<\mp@subsup{s}{}{\prime}>])\in congruenc
by simp
qed
end
theory Strong-Late-Sim-SC
imports Strong-Late-Sim
begin
lemma nilSim[dest]:
fixes a :: name
and b:: name
and }x:: nam
and }P::p
and }Q::p
shows 0}\rightsquigarrow[Rel]\tau.(P)\Longrightarrow\mathrm{ False
and 0}\rightsquigarrow[Rel] a<x>.P\Longrightarrow Fals
and 0}\rightsquigarrow[Rel]a{b}.P\Longrightarrow\mathrm{ False
by(fastforce simp add: simulation-def intro: Tau Input Output)+
lemma nilSimRight:
fixes P :: pi
and Rel ::(pi\times pi) set
shows P}\rightsquigarrow[Rel]
by(auto simp add: simulation-def)

```
```

lemma matchIdLeft:
fixes $a$ :: name
and $\quad P$ :: pi
and Rel $::(p i \times p i)$ set
assumes $I d \subseteq$ Rel
shows $[a \frown a] P \rightsquigarrow[$ Rel $] P$
using assms
by (force simp add: simulation-def dest: Match derivativeReflexive)
lemma matchIdRight:
fixes $P$ :: $p i$
and $a$ :: name
and Rel $::(p i \times p i)$ set
assumes $I d R e l: I d \subseteq \operatorname{Rel}$
shows $P \rightsquigarrow[R e l][a \frown a] P$
using assms
by(fastforce simp add: simulation-def elim: matchCases intro: derivativeReflexive)
lemma matchNilLeft:
fixes $a$ :: name
and $b::$ name
and $\quad P:: p i$
assumes $a \neq b$
shows $\mathbf{0} \rightsquigarrow[R e l][a \frown b] P$
using assms
by (auto simp add: simulation-def)
lemma mismatchIdLeft:
fixes $a$ :: name
and $b$ :: name
and $P$ :: pi
and Rel $::(p i \times p i)$ set
assumes $I d \subset$ Rel
and $\quad a \neq b$
shows $[a \neq b] P \rightsquigarrow[$ Rel $] P$
using assms
by(fastforce simp add: simulation-def intro: Mismatch dest: derivativeReflexive)

```
```

lemma mismatchIdRight:
fixes P :: pi
and a :: name
and b :: name
and Rel ::(pi\timespi) set
assumes IdRel:Id \subseteq Rel
and aineqb: }a\not=
shows P\rightsquigarrow[Rel][a\not=b]P
using assms
by(fastforce simp add: simulation-def elim: mismatchCases intro:derivativeReflex-
ive)
lemma mismatchNilLeft:
fixes a :: name
and }P\mathrm{ :: pi
shows 0}\rightsquigarrow[Rel][a\not=a]
by(auto simp add: simulation-def)
lemma sumSym:
fixes P :: pi
and }Q :: p
and Rel :: (pi\timespi) set
assumes Id:Id\subseteqRel
shows }P\oplusQ\rightsquigarrow[Rel] Q\oplus
using assms
by(fastforce simp add: simulation-def elim: sumCases intro: Sum1 Sum2 deriva-
tiveReflexive)
lemma sumIdempLeft:
fixes P :: pi
and Rel :: (pi \times pi) set
assumes Id \subseteqRel
shows P\rightsquigarrow[Rel] P\oplusP
using assms
by(fastforce simp add: simulation-def elim: sumCases intro: derivativeReflexive)
lemma sumIdempRight:
fixes P :: pi
and Rel :: (pi \times pi) set

```
assumes \(I: I d \subseteq\) Rel
shows \(P \oplus P \rightsquigarrow[\) Rel \(] P\)
using assms
by(fastforce simp add: simulation-def intro: Sum1 derivativeReflexive)
lemma sumAssocLeft:
fixes \(P\) :: \(p i\)
and \(Q \quad:: p i\)
and \(R \quad:: p i\)
and Rel \(::(p i \times p i)\) set
assumes \(I d: I d \subseteq R e l\)
shows \((P \oplus Q) \oplus R \rightsquigarrow[\operatorname{Rel}] P \oplus(Q \oplus R)\)
using assms
by(fastforce simp add: simulation-def elim: sumCases intro: Sum1 Sum2 derivativeReflexive)
lemma sumAssocRight:
fixes \(P\) :: \(p i\)
and \(Q:: p i\)
and \(R \quad:: p i\)
and Rel :: \((p i \times p i)\) set
assumes \(I d: I d \subseteq\) Rel
```

    shows \(P \oplus(Q \oplus R) \rightsquigarrow[\) Rel \(](P \oplus Q) \oplus R\)
    using assms
by(fastforce simp add: simulation-def elim: sumCases intro: Sum1 Sum2 deriva-
tiveReflexive)
lemma sumZeroLeft:
fixes $P$ :: $p i$
and Rel :: $(p i \times p i)$ set
assumes $I d: I d \subseteq R e l$
shows $P \oplus \mathbf{0} \rightsquigarrow[$ Rel $] P$
using assms
by(fastforce simp add: simulation-def intro: Sum1 derivativeReflexive)
lemma sumZeroRight:
fixes $P$ :: $p i$
and Rel $::(p i \times p i)$ set
assumes $I d: I d \subseteq R e l$

```
```

shows P\rightsquigarrow[Rel] P\oplus0
using assms
by(fastforce simp add: simulation-def elim: sumCases intro:derivativeReflexive)
lemma sumResLeft:
fixes x :: name
and }P\mathrm{ :: pi
and }Q ::p
assumes Id:Id \subseteqRel
and Eqvt: eqvt Rel
shows }(<\nux>P)\oplus(<\nux>Q)\rightsquigarrow[Rel]<\nux>(P\oplusQ
using Eqvt
proof(induct rule: simCasesCont[where C=(x,P,Q)])
case(Bound a y PQ)
from < }y\sharp(x,P,Q)> have y\not=x and y\sharpP and y\sharpQ by(simp add: fresh-prod)
hence }y\sharpP\oplusQ\mathrm{ by simp
with <<\nux>
proof(induct rule: resCasesB)
case(cOpen a PQ)
from \langleP\oplus Q\longmapstoa[x]\precPQ>\langley\sharpP>\langley\sharpQ> have y
freshFreeDerivative)
from }\langleP\oplusQ\longmapstoa[x]\precPQ\rangle\mathrm{ show ?case
proof(induct rule: sumCases)
case cSum1
from <P\longmapstoa[x] \precPQ>\langlea\not= x> have <\nux>P \longmapsto
Open)
hence }(<\nux>P)\oplus(<\nux>Q)\longmapstoa<\nux>< \precPQ by(rule Sum1
with \langley\sharpPQ\rangle have }(<\nux>P)\oplus(<\nux>>Q)\longmapstoa<\nuy>\prec \prec ([(y,x)]\cdotPQ
by(simp add: alphaBoundResidual)
moreover from Id have derivative ([(y,x)]\cdotPQ) ([(y,x)]\cdotPQ) (BoundOutputS
a) y Rel
by(force simp add: derivative-def)
ultimately show ?case by blast
next
case cSum2
from <Q\longmapstoa[x] \precPQ\rangle\langlea\not= x> have <\nux> }\langle\mp@code{\longmapsto
Open)
hence }(<\nux>P)\oplus(<\nux>Q)\longmapstoa<\nux> \precPQ by(rule Sum2)
with < y \#PQ> have }(<\nux>P)\oplus(<\nux>Q)\longmapstoa<\nuy>< \prec ([(y,x)]\cdotPQ
by(simp add: alphaBoundResidual)
moreover from Id have derivative ([(y,x)]\cdotPQ) ([(y,x)]\cdotPQ) (BoundOutputS
a) y Rel
by(force simp add: derivative-def)
ultimately show ?case by blast
qed
next
case(cRes PQ)

```
```

    from 〈P\oplusQ\longmapstoa«y> \precPQ\rangle show ?case
    proof(induct rule: sumCases)
    case cSum1
    from \langleP\longmapstoa<y> \precPQ\rangle\langlex\sharpa\rangle\langley\not=x\rangle have <\nux>P\longmapsto \longmapstoa<y» \prec<\nux>PQ
    by(rule-tac ResB) auto
hence (<\nux>P) \oplus(<\nux>Q)\longmapstoa<y» \prec<\nux>PQ by(rule Sum1)
moreover from Id have derivative (<\nux>PPQ)(<\nux>PQ) a y Rel
by(cases a) (auto simp add: derivative-def)
ultimately show ?case by blast
next
case cSum2
from 〈Q\longmapstoa«y» \precPQ\rangle\langlex\sharpa\rangle\langley\not= x\rangle have <\nux>Q \longmapstoa|y» \prec<\nux>>PQ
by(rule-tac ResB) auto
hence (<\nux>P)\oplus(<\nux>>) \longmapstoa«y» \prec<\nux>PQ by(rule Sum2)
moreover from Id have derivative ( <\nux>PQ) (<\nux>PQ) a y Rel
by(cases a) (auto simp add: derivative-def)
ultimately show ?case by blast
qed
qed
next
case(Free \alpha PQ)
from <<\nux>(P\oplusQ)\longmapsto\alpha\precPQ> show ?case
proof(induct rule: resCasesF)
case(cRes PQ)
from 〈P\oplusQ\longmapsto\alpha\precPQ\rangle show ?case
proof(induct rule: sumCases)
case cSum1
from }\langleP\longmapsto\alpha\precPQ\rangle\langlex\sharp\alpha>\mathrm{ have < < x>P ط ط < < vx>PPQ by(rule ResF)
hence }(<\nux>P)\oplus(<\nux>Q)\longmapsto\alpha\prec<\nux>PQ by(rule Sum1
with Id show ?case by blast
next
case cSum2
from }\langleQ\longmapsto\alpha\precPQ\rangle\langlex\sharp\alpha\rangle\mathrm{ have < < x>>Qط ط < < L x>PQ by(rule ResF)
hence (<\nux>P)\oplus(<\nux>Q)\longmapsto\alpha\prec<\nux>PQ by(rule Sum2)
with Id show ?case by blast
qed
qed
qed
lemma sumResRight:
fixes x :: name
and P :: pi
and }Q::p
assumes Id:Id \subseteqRel
and Eqvt: eqvo Rel
shows <\nux> (P\oplusQ)\rightsquigarrow[Rel] (<\nux>P)\oplus(<\nux>Q)
using \eqvt Rel`

```
proof \((\) induct rule: simCasesCont \([\) where \(C=(x, P, Q)])\)
case (Bound a y \(P Q\) )
from \(\langle y \sharp(x, P, Q)\rangle\) have \(y \neq x\) and \(y \sharp P\) and \(y \sharp Q\) by (simp add: fresh-prod \()+\)
from \(\langle(<\nu x>P) \oplus(<\nu x>Q) \longmapsto a « y » \prec P Q\rangle\) show ?case
proof(induct rule: sumCases)
case \(c\) Sum 1
from \(\langle<\nu x\rangle P \longmapsto a « y\) » \(\langle P Q\rangle\) show ? case using \(\langle y \neq x\rangle\langle y \sharp P\rangle\)
proof (induct rule: resCasesB)
case (cOpen a \(P^{\prime}\) )
from \(\left\langle P \longmapsto a[x] \prec P^{\prime}\right\rangle\langle y \sharp P\rangle\) have \(y \sharp P^{\prime} \mathbf{b y}(\) rule freshFreeDerivative)
from \(\left\langle P \longmapsto a[x] \prec P^{\prime}\right\rangle\) have \(P \oplus Q \longmapsto a[x] \prec P^{\prime}\) by(rule Sum1)
hence \(<\nu x>(P \oplus Q) \longmapsto a<\nu x\rangle \prec P^{\prime}\) using \(\langle a \neq x\rangle \mathbf{b y}(\) rule Open)
with \(\left\langle y \sharp P^{\prime}\right\rangle\) have \(<\nu x>(P \oplus Q) \longmapsto a<\nu y>\prec[(y, x)] \cdot P^{\prime}\) by (simp add:
alphaBoundResidual)
moreover from \(I d\) have derivative \(\left([(y, x)] \cdot P^{\prime}\right)\left([(y, x)] \cdot P^{\prime}\right)\) (BoundOutputS
a) y Rel
by(force simp add: derivative-def)
ultimately show ?case by blast
next
case (cRes \(\left.P^{\prime}\right)\)
from \(\left\langle P \longmapsto a « y » \prec P^{\prime}\right\rangle\) have \(P \oplus Q \longmapsto a « y » \prec P^{\prime} \mathbf{b y}(\) rule Sum1)
hence \(\langle\nu x\rangle(P \oplus Q) \longmapsto a « y » \prec\left\langle\nu x>P^{\prime}\right.\) using \(\langle x \sharp a\rangle\langle y \neq x\rangle\) by \((\) rule-tac ResB) auto
moreover from Id have derivative \(\left(<\nu x>P^{\prime}\right)\left(<\nu x>P^{\prime}\right)\) a y Rel
by (cases a) (auto simp add: derivative-def)
ultimately show ?case by blast
qed
next
case cSum2
from 〈< \(\langle x\rangle Q \longmapsto a « y » \prec P Q\rangle\) show ? case using \(\langle y \neq x\rangle\langle y \sharp Q\rangle\)
proof (induct rule: resCasesB)
case(cOpen a \(Q^{\prime}\) )
from \(\left\langle Q \longmapsto a[x] \prec Q^{\prime}\right\rangle\langle y \sharp Q\rangle\) have \(y \sharp Q^{\prime} \mathbf{b y}(\) rule freshFreeDerivative)
from \(\left\langle Q \longmapsto a[x] \prec Q^{\prime}\right\rangle\) have \(P \oplus Q \longmapsto a[x] \prec Q^{\prime}\) by(rule Sum2)
hence \(<\nu x>(P \oplus Q) \longmapsto a<\nu x>\prec Q^{\prime}\) using \(\langle a \neq x\rangle\) by (rule Open)
with \(\left\langle y \sharp Q^{\prime}>\right.\) have \(<\nu x>(P \oplus Q) \longmapsto a<\nu y>\prec[(y, x)] \cdot Q^{\prime}\) by \((\) simp add: alphaBoundResidual)
moreover from Id have derivative \(\left([(y, x)] \cdot Q^{\prime}\right)\left([(y, x)] \cdot Q^{\prime}\right)(\) BoundOutputS a) y Rel
by(force simp add: derivative-def)
ultimately show ?case by blast
next
case (cRes \(Q^{\prime}\) )
from \(\left\langle Q \longmapsto a « y » \prec Q^{\prime}\right\rangle\) have \(P \oplus Q \longmapsto a « y » \prec Q^{\prime}\) by (rule Sum2)
hence \(\left\langle\nu x>(P \oplus Q) \longmapsto a « y » \prec\left\langle\nu x>Q^{\prime}\right.\right.\) using \(\langle x \sharp a\rangle\langle y \neq x\rangle\) by (rule-tac ResB) auto
moreover from Id have derivative \(\left(<\nu x>Q^{\prime}\right)\left(<\nu x>Q^{\prime}\right)\) a y Rel
```

                by(cases a) (auto simp add:derivative-def)
            ultimately show ?case by blast
        qed
    qed
    next
case(Free \alpha PQ)
from < (<\nux>P ) \oplus(<\nux>Q)\longmapsto\alpha \precPQ> show ?case
proof(induct rule: sumCases)
case cSum1
from <<\nux>P \longmapsto
proof(induct rule: resCasesF)
case(cRes P')
from }\langleP\longmapsto\alpha\prec\mp@subsup{P}{}{\prime}>\mathrm{ have }P\oplusQ\longmapsto\alpha\prec\mp@subsup{P}{}{\prime}\mathrm{ by(rule Sum1)
hence <\nux>(P\oplusQ)\longmapsto\alpha\prec<\nux> '名 using<x\sharp\alpha> by(rule ResF)
with Id show ?case by blast
qed
next
case cSum2
from <<\nux>>Q\longmapsto\alpha\precPQ> show ?case
proof(induct rule: resCasesF)
case(cRes Q')

```

```

            hence <\nux> (P\oplusQ)\longmapsto\alpha\prec<\nux> '' using <x\sharp\alpha> by(rule ResF)
            with Id show ?case by blast
        qed
    qed
    qed

```
lemma parZeroLeft:
    fixes \(P\) :: \(p i\)
    and Rel \(::(p i \times p i)\) set
    assumes ParZero: \(\bigwedge Q .(Q \| \mathbf{0}, Q) \in\) Rel
    shows \(P \| \mathbf{0} \rightsquigarrow[R e l] P\)
proof -
    \{
        fix \(P Q a x\)
        from ParZero have derivative \((P \| \mathbf{0}) P\) a \(x\) Rel
            by (case-tac a) (auto simp add: derivative-def)
    \}
    thus ?thesis using assms
        by (fastforce simp add: simulation-def intro: Par1B Par1F)
qed
lemma parZeroRight:
    fixes \(P\) :: \(p i\)
```

    and Rel ::(pi\timespi) set
    assumes ParZero: }\Q.(Q,Q|\mathbf{0})\in\mathrm{ Rel
    shows P\rightsquigarrow[Rel] P|0
    proof -
{
fix PQ ax
from ParZero have derivative P (P|\mathbf{0})ax Rel
by(case-tac a) (auto simp add: derivative-def)
}
thus ?thesis using assms
by(fastforce simp add: simulation-def elim: parCasesF parCasesB)+
qed
lemma parSym:
fixes P :: pi
and Q :: pi
and Rel :: (pi\times pi) set
assumes Sym: \bigwedgeR S. (R|S,S|R)\in Rel
and Res:\RSx. (R,S)\inRel\Longrightarrow(<\nux>R,<\nux>S) \in Rel
shows P|Q\rightsquigarrow[Rel] Q|P
proof(induct rule: simCases)
case(Bound a x QP)
from <x\sharp (P|Q)\rangle have }x\sharpQQ\mathrm{ and }x\sharpP\mathrm{ by simp +
with \Q|P\longmapsto \ |<x» \prec QP> show ?case
proof(induct rule: parCasesB)
case(cPar1 Q')
from〈Q\longmapstoa«x» \prec ' '> have }P||Q\longmapstoa«x» \precP| Q ' using\langlex\sharpP> by(rule
Par2B)
moreover have derivative (P| Q') ( Q'|P) a x Rel
by(cases a, auto simp add:derivative-def intro:Sym)
ultimately show ?case by blast
next
case(cPar2 P')
from <P\longmapstoa«x» \prec P'> have }P||Q\longmapstoa«x»\prec \prec ' | | Q using <x \# Q> by(rule
Par1B)
moreover have derivative ( }\mp@subsup{P}{}{\prime}|Q)(Q|\mp@subsup{P}{}{\prime})\mathrm{ a x Rel
by(cases a, auto simp add: derivative-def intro: Sym)
ultimately show ?case by blast
qed
next
case(Free \alpha QP)
from <Q|P\longmapsto }\langle<⿱\prec\prec\mp@code{\}\mathrm{ show ?case
proof(induct rule: parCasesF[where C=()])
case(cPar1 Q')
from <Q\longmapsto\alpha\prec 蛈 have P|Q\longmapsto\alpha\longmapsto P

```
```

    moreover have (P| Q', Q'|P)\inRel by(rule Sym)
    ultimately show ?case by blast
    next
    case(cPar2 P')
    from }\langleP\longmapsto\alpha\prec\mp@subsup{P}{}{\prime}\rangle\mathrm{ have }P|Q\longmapsto\alpha\prec\mp@subsup{P}{}{\prime}|Q by(rule Par1F
    moreover have ( }\mp@subsup{P}{}{\prime}|Q,Q|\mp@subsup{P}{}{\prime})\in\mathrm{ Rel by(rule Sym)
    ultimately show ?case by blast
    next
case(cComm1 Q' P' a b x)
from }\langleP\longmapstoa[b]\prec\mp@subsup{P}{}{\prime}\rangle\langleQ\longmapstoa<x><\prec\mp@subsup{Q}{}{\prime}
have }P|Q\longmapsto\tau\prec\mp@subsup{P}{}{\prime}|(\mp@subsup{Q}{}{\prime}[x::=b])\mathbf{by}(rule Comm2)
moreover have ( }\mp@subsup{P}{}{\prime}|\mp@subsup{Q}{}{\prime}[x::=b],\mp@subsup{Q}{}{\prime}[x::=b]|\mp@subsup{P}{}{\prime})\in\mathrm{ Rel by(rule Sym)
ultimately show ?case by blast
next
case(cComm2 Q' P' a b x)
from }\langleP\longmapstoa<x><<\mp@subsup{P}{}{\prime}\rangle\langleQ\longmapstoa[b]\prec\mp@subsup{Q}{}{\prime}
have P|Q\longmapsto\tau\prec(\mp@subsup{P}{}{\prime}[x::=b])| \mp@subsup{Q}{}{\prime}\mathbf{by}(rule Comm1)

```

```

    ultimately show ?case by blast
    next
case(cClose1 Q' P' a x y)
from <P\longmapsto \longmapstoa<\nuy> \prec P'\rangle\langleQ\longmapstoa<x>\prec < '>\langley\#Q>
have }P|Q\longmapsto\tau\prec<\nuy>(\mp@subsup{P}{}{\prime}|(\mp@subsup{Q}{}{\prime}[x::=y]))\mathbf{by}(rule Close2)
moreover have (<\nuy>(\mp@subsup{P}{}{\prime}|\mp@subsup{Q}{}{\prime}[x::=y]),<\nuy>(\mp@subsup{Q}{}{\prime}[x::=y]| |
Res Sym)
ultimately show ?case by blast
next
case(cClose2 Q' P' a x y)
from <P\longmapsto
have }P|Q\longmapsto\tau\prec<\nuy>((\mp@subsup{P}{}{\prime}[x::=y])|\mp@subsup{Q}{}{\prime})\mathbf{by}(\mathrm{ rule Close1)
moreover have (<\nuy>(\mp@subsup{P}{}{\prime}[x::=y]| | 的),<\nuy>(\mp@subsup{Q}{}{\prime}|\mp@subsup{P}{}{\prime}[x::=y]))\in\mathrm{ Rel by(metis}
Res Sym)
ultimately show ?case by blast
qed
qed
lemma parAssocLeft:
fixes P :: pi
and }Q\quad::p
and R :: pi
and Rel :: (pi\times pi) set
assumes Ass: }\quad\STU.((S|T)|U,S|(T|U))\inRe
and Res: }\quad\STx.(S,T)\inRel\Longrightarrow(<\nux>S,<\nux>T)\inRe
and FreshExt: \STUx. x\sharpS\Longrightarrow(<\nux>((S|T)|U),S|<\nux>(T|
U)) \in Rel
and FreshExt': \S T U x. x\sharpU\Longrightarrow((<\nux>(S|T))|U,<\nux>(S|(T|
U)))}\in\operatorname{Rel

```
```

    shows (P|Q)|R\rightsquigarrow[Rel] P|(Q|R)
    proof(induct rule: simCases)
case(Bound a x PQR)
from< < }\#\#(P|Q)|R> have x\sharpP and x\sharp Q and x\sharpR by simp
hence }x\sharp(Q|R)\mathrm{ by simp
with <P| | Q | R)\longmapstoa«x>\prec \longmapstoPQR\rangle\langlex\sharpP> show ?case
proof(induct rule: parCasesB)
case(cPar1 P')

```

```

by(rule Par1B)

```

```

    moreover have derivative ((\mp@subsup{P}{}{\prime}|Q)|R)(\mp@subsup{P}{}{\prime}|(Q|R)) a x Rel
        by(cases a, auto intro: Ass simp add: derivative-def)
    ultimately show ?case by blast
    next
    case(cPar2 QR)
    from〈Q|R\longmapstoa«x>\prec \QR\rangle\langlex #Q\rangle\langlex\sharpR\rangle\mathrm{ show ?case}
    proof(induct rule: parCasesB)
        case(cPar1 Q')
        from〈Q\longmapstoa<x>< < Q > have P| | Q\longmapstoa<x» \precP| | Q' using<x\sharpP>
    by(rule Par2B)
hence (P|Q)|R\longmapstoa<x>\prec (P| Q')|R using<x\sharpR>\mathbf{by}(rule Par1B)
moreover have derivative ((P| Q ')|R) (P|( (Q | R)) a x Rel
by(cases a, auto intro: Ass simp add: derivative-def)
ultimately show ?case by blast
next
case(cPar2 R')
from <R\longmapstoa«x»\prec R'> have ( }P||)|R\longmapstoa<x»\prec(P|Q)| | R' using
<x\sharpP\rangle\langlex\sharpQ\rangle
by(rule-tac Par2B) auto
moreover have derivative ((P|Q)| |
by(cases a, auto intro: Ass simp add: derivative-def)
ultimately show ?case by blast
qed
qed
next
case(Free \alpha PQR)
from <P| (Q|R)\longmapsto\alpha\prec PQR> show ?case
proof(induct rule: parCasesF[where C=Q])
case(cPar1 P')

```

```

    hence (P|Q)|R\longmapsto\alpha\prec (P'|Q)|R by(rule Par1F)
    moreover from Ass have ((\mp@subsup{P}{}{\prime}|Q)|R,\mp@subsup{P}{}{\prime}|(Q|R))\inRel by blast
    ultimately show ?case by blast
    next
    case(cPar2 QR)
    from 〈Q| R\longmapsto\alpha\precQR` show ?case
    proof(induct rule: parCasesF[where C=P])
        case(cPar1 Q')
    ```
from \(\left\langle Q \longmapsto \alpha \prec Q^{\prime}\right\rangle\) have \((P \| Q) \longmapsto \alpha \prec P \| Q^{\prime}\) by (rule Par2F) hence \((P \| Q)\left\|R \longmapsto \alpha \prec\left(P \| Q^{\prime}\right)\right\| R\) by \((\) rule Par1F \()\) moreover from Ass have \(\left(\left(P \| Q^{\prime}\right)\|R, P\|\left(Q^{\prime} \| R\right)\right) \in\) Rel by blast ultimately show ?case by blast
next
case (cPar2 \(R^{\prime}\) )
from \(\left\langle R \longmapsto \alpha \prec R^{\prime}\right\rangle\) have \((P \| Q)\|R \longmapsto \alpha \prec(P \| Q)\| R^{\prime}\) by (rule Par2F)
moreover from Ass have \(\left((P \| Q)\left\|R^{\prime}, P\right\|\left(Q \| R^{\prime}\right)\right) \in\) Rel by blast ultimately show? case by blast
next
case \(\left(c \operatorname{Comm1} Q^{\prime} R^{\prime}\right.\) abla)
from \(\left.\langle Q \longmapsto a<x\rangle \prec Q^{\prime}\right\rangle\langle x \sharp P\rangle\) have \(\left.P \| Q \longmapsto a<x\right\rangle \prec P \| Q^{\prime}\) by (rule Par2B)
hence \((P \| Q)\left\|R \longmapsto \tau \prec\left(P \| Q^{\prime}\right)[x::=b]\right\| R^{\prime}\) using \(\left\langle R \longmapsto a[b] \prec R^{\prime}>\right.\) by (rule Comm1)
with \(\left\langle x \sharp P>\right.\) have \((P \| Q)\left\|R \longmapsto \tau \prec\left(P \|\left(Q^{\prime}[x::=b]\right)\right)\right\| R^{\prime}\) by \((\) simp add: forget)
moreover from Ass have \(\left(\left(P \|\left(Q^{\prime}[x::=b]\right)\right)\left\|R^{\prime}, P\right\|\left(Q^{\prime}[x::=b] \| R^{\prime}\right)\right) \in\) Rel by blast
ultimately show ?case by blast
next
case (cComm2 \(Q^{\prime} R^{\prime}\) abla)
from \(\left\langle Q \longmapsto a[b] \prec Q^{\prime}\right.\) 〉 have \(P\|Q \longmapsto a[b] \prec P\| Q^{\prime}\) by (rule Par2F)
with \(\left.\langle x \sharp P\rangle\langle x \sharp Q\rangle\langle R \longmapsto a<x\rangle \prec R^{\prime}\right\rangle\) have \((P \| Q) \| R \longmapsto \tau \prec(P \|\)
\(\left.Q^{\prime}\right) \| R^{\prime}[x::=b]\)
by(force intro: Comm2)
moreover from Ass have \(\left(\left(P \| Q^{\prime}\right)\left\|R^{\prime}[x::=b], P\right\|\left(Q^{\prime} \| R^{\prime}[x::=b]\right)\right) \in \operatorname{Rel}\) by blast
ultimately show ?case by blast
next
case (cClose1 \(Q^{\prime} R^{\prime}\) a \(x y\) )
from \(\left.\langle Q \longmapsto a<x\rangle \prec Q^{\prime}\right\rangle\langle x \sharp P\rangle\) have \(P\|Q \longmapsto a<x>\prec P\| Q^{\prime}\) by (rule Par2B)
with \(\left.\langle y \sharp P\rangle\langle y \sharp Q\rangle\langle x \sharp P\rangle\langle R \longmapsto a<\nu y\rangle \prec R^{\prime}\right\rangle\) have \((P \| Q) \| R \longmapsto \tau\) \(\prec<\nu y>\left(\left(P \| Q^{\prime}\right)[x::=y] \| R^{\prime}\right)\)
by(rule-tac Close1) auto
with \(\langle x \sharp P\rangle\) have \((P \| Q) \| R \longmapsto \tau \prec<\nu y>\left(\left(P \|\left(Q^{\prime}[x::=y]\right)\right) \| R^{\prime}\right)\) by (simp add: forget)
moreover from \(\left\langle y \sharp P>\right.\) have \(\left(<\nu y>\left(\left(P \| Q^{\prime}[x::=y]\right) \| R^{\prime}\right), P \|<\nu y>\left(Q^{\prime}[x::=y]\right.\right.\) \(\left.\left.\| R^{\prime}\right)\right) \in\) Rel
by (rule FreshExt)
ultimately show ?case by blast
next
case (cClose2 \(Q^{\prime} R^{\prime}\) a \(x y\) )
from \(\left\langle Q \longmapsto a<\nu y>\prec Q^{\prime}\right\rangle\left\langle y \sharp P>\right.\) have \(P\|Q \longmapsto a<\nu y>\prec P\| Q^{\prime}\) by (rule Par2B)
hence \(\operatorname{Act}:(P \| Q) \| R \longmapsto \tau \prec<\nu y>\left(\left(P \| Q^{\prime}\right) \| R^{\prime}[x::=y]\right)\) using \(\langle R\) \(\left.\longmapsto a<x>\prec R^{\prime}\right\rangle\langle y \sharp R\rangle\) by (rule Close2)
moreover from \(\langle y \sharp P\rangle\) have \(\left(<\nu y>\left(\left(P \| Q^{\prime}\right) \| R^{\prime}[x::=y]\right), P \|<\nu y>\left(Q^{\prime}\right.\right.\) \(\left.\left.\| R^{\prime}[x::=y]\right)\right) \in \operatorname{Rel}\)
by(rule FreshExt)
ultimately show ?case by blast
qed
next
case \(\left(c C o m m 1 P^{\prime} Q R\right.\) abx)
from \(\langle Q \| R \longmapsto a[b] \prec Q R\rangle\) show ?case
proof \((\) induct rule: parCases \(F[\) where \(C=()])\)
case (cPar1 \(Q^{\prime}\) )
from \(\left.\langle P \longmapsto a<x\rangle \prec P^{\prime}\right\rangle\left\langle Q \longmapsto a[b] \prec Q^{\prime}\right\rangle\) have \(P \| Q \longmapsto \tau \prec P^{\prime}[x::=b]\)
\(\| Q^{\prime} \mathbf{b y}(\) rule Comm1)
hence \((P \| Q)\left\|R \longmapsto \tau \prec\left(P^{\prime}[x::=b] \| Q^{\prime}\right)\right\| R\) by \((\) rule Par1F)
moreover from Ass have \(\left(\left(P^{\prime}[x::=b] \| Q^{\prime}\right)\left\|R, P^{\prime}[x::=b]\right\|\left(Q^{\prime} \| R\right)\right) \in\) Rel
by blast
ultimately show ?case by blast
next
case(cPar2 \(R^{\prime}\) )
from \(\left.\langle P \longmapsto a<x\rangle \prec P^{\prime}\right\rangle\langle x \sharp Q\rangle\) have \(\left.P \| Q \longmapsto a<x\right\rangle \prec P^{\prime} \| Q\) by(rule Par1B)
hence \((P \| Q)\left\|R \longmapsto \tau \prec\left(P^{\prime} \| Q\right)[x::=b]\right\| R^{\prime}\) using \(\left\langle R \longmapsto a[b] \prec R^{\prime}>\right.\) by (rule Comm1)
with \(\langle x \sharp Q\rangle\) have \((P \| Q)\left\|R \longmapsto \tau \prec\left(P^{\prime}[x::=b] \| Q\right)\right\| R^{\prime}\) by \((\) simp add: forget)
moreover from Ass have \(\left(\left(P^{\prime}[x::=b] \| Q\right)\left\|R^{\prime}, P^{\prime}[x::=b]\right\|\left(Q \| R^{\prime}\right)\right) \in\) Rel by blast
ultimately show ?case by blast
next
case \(\left(c C o m m 1 \quad Q^{\prime} R^{\prime}\right)\)
from \(\langle a[b]=\tau\rangle\) have False by simp thus? case by simp
next
case (cComm2 \(Q^{\prime} R^{\prime}\) )
from \(\langle a[b]=\tau\rangle\) have False by simp thus? case by simp
next
case(cClose1 \(\left.Q^{\prime} R^{\prime}\right)\)
from \(\langle a[b]=\tau\rangle\) have False by simp thus ?case by simp
next
case (cClose2 \(Q^{\prime} R^{\prime}\) )
from \(\langle a[b]=\tau\rangle\) have False by simp thus? case by simp
qed
next
case \(\left(c C o m m 2 P^{\prime} Q R\right.\) a b \(x\) )
from \(\langle x \sharp Q \| R\rangle\) have \(x \sharp Q\) and \(x \sharp R\) by simp +
with \(\langle Q \| R \longmapsto a<x>\prec Q R\rangle\) show ?case
proof (induct rule: parCasesB)
case(cPar1 \(\left.Q^{\prime}\right)\)
from \(\left.\left\langle P \longmapsto a[b] \prec P^{\prime}\right\rangle\langle Q \longmapsto a<x\rangle \prec Q^{\prime}\right\rangle\) have \(P\left\|Q \longmapsto \tau \prec P^{\prime}\right\|\) ( \(Q^{\prime}[x::=b]\) ) by (rule Comm2)
hence \((P \| Q)\left\|R \longmapsto \tau \prec\left(P^{\prime} \| Q^{\prime}[x::=b]\right)\right\| R\) by (rule Par1F)
moreover from Ass have \(\left(\left(P^{\prime} \| Q^{\prime}[x::=b]\right)\left\|R, P^{\prime}\right\| Q^{\prime}[x::=b] \| R\right) \in\) Rel by blast
with \(\langle x \sharp R\rangle\) have \(\left(\left(P^{\prime} \| Q^{\prime}[x::=b]\right)\left\|R, P^{\prime}\right\|\left(Q^{\prime} \| R\right)[x::=b]\right) \in\) Rel by \((\) force simp add: forget)
ultimately show ?case by blast
next
case (cPar2 \(R^{\prime}\) )
from \(\left\langle P \longmapsto a[b] \prec P^{\prime}\right\rangle\) have \(P\left\|Q \longmapsto a[b] \prec P^{\prime}\right\| Q\) by (rule Par1F)
hence \((P \| Q)\left\|R \longmapsto \tau \prec\left(P^{\prime} \| Q\right)\right\|\left(R^{\prime}[x::=b]\right)\) using \(\langle R \longmapsto a<x>\prec\) \(R^{\prime}>\) by (rule Comm2)
moreover from Ass have \(\left(\left(P^{\prime} \| Q\right)\left\|R^{\prime}[x::=b], P^{\prime}\right\| Q \|\left(R^{\prime}[x::=b]\right)\right) \in\) Rel by blast
hence \(\left(\left(P^{\prime} \| Q\right)\left\|R^{\prime}[x::=b], P^{\prime}\right\|\left(Q \| R^{\prime}\right)[x::=b]\right) \in\) Rel using \(\langle x \sharp Q\rangle\) by (force simp add: forget)
ultimately show ?case by blast
qed
next
case (cClose1 \(P^{\prime} Q R\) a \(x y\) )
from \(\langle x \sharp Q \| R\rangle\) have \(x \sharp Q\) by simp
from \(\langle y \sharp Q \| R\rangle\) have \(y \sharp Q\) and \(y \sharp R\) by simp +
from \(\langle Q \| R \longmapsto a<\nu y>\prec Q R\rangle\langle y \sharp Q\rangle\langle y \sharp R\rangle\) show ? case
proof (induct rule: parCasesB)
case(cPar1 \(\left.Q^{\prime}\right)\)
from \(\left.\langle P \longmapsto a<x\rangle \prec P^{\prime}\right\rangle\left\langle Q \longmapsto a<\nu y>\prec Q^{\prime}\right\rangle\langle y \sharp P\rangle\) have \(P \| Q \longmapsto \tau\) \(\prec<\nu y>\left(P^{\prime}[x::=y] \| Q^{\prime}\right) \mathbf{b y}(\) rule Close1 \()\)
hence \((P \| Q)\left\|R \longmapsto \tau \prec\left(<\nu y>\left(P^{\prime}[x::=y] \| Q^{\prime}\right)\right)\right\| R\) by (rule Par1F)
moreover from \(\langle y \sharp R\rangle\) have \(\left(\left(<\nu y>\left(P^{\prime}[x::=y] \| Q^{\prime}\right)\right) \| R,<\nu y>\left(P^{\prime}[x::=y]\right.\right.\)
\(\left.\left.\left\|Q^{\prime}\right\| R\right)\right) \in\) Rel
by (rule FreshExt')
ultimately show ?case by blast
next
case(cPar2 \(\left.R^{\prime}\right)\)
from \(\left.\langle P \longmapsto a<x\rangle \prec P^{\prime}\right\rangle\langle x \sharp Q\rangle\) have \(\left.P \| Q \longmapsto a<x\right\rangle \prec P^{\prime} \| Q\) by (rule Par1B)
with \(\left\langle R \longmapsto a<\nu y>\prec R^{\prime}\right\rangle\langle y \sharp P\rangle\langle y \sharp Q\rangle\) have \((P \| Q) \| R \longmapsto \tau \prec\) \(<\nu y>\left(\left(P^{\prime} \| Q\right)[x::=y] \| R^{\prime}\right)\)
by (rule-tac Close1) auto
with \(\langle x \sharp Q\rangle\) have \((P \| Q) \| R \longmapsto \tau \prec<\nu y>\left(\left(P^{\prime}[x::=y] \| Q\right) \| R^{\prime}\right) \mathbf{b y}(\operatorname{simp}\) add: forget)
moreover have \(\left(<\nu y>\left(\left(P^{\prime}[x::=y] \| Q\right) \| R^{\prime}\right),<\nu y>\left(P^{\prime}[x::=y] \|\left(Q \| R^{\prime}\right)\right)\right)\) \(\in\) Rel by (metis Ass Res)
ultimately show ?case by blast
qed
next
case (cClose2 \(P^{\prime} Q R\) a \(x y\) )
from \(\langle y \sharp Q \| R\rangle\) have \(y \sharp Q\) and \(y \sharp R\) by simp +
from \(\langle x \sharp Q \| R\rangle\) have \(x \sharp Q\) and \(x \sharp R\) by simp +
with \(\langle Q \| R \longmapsto a<x>\prec Q R\rangle\) show ?case
proof (induct rule: parCasesB)
case \(\left(c \operatorname{Par} 1 Q^{\prime}\right)\)
from \(\left.\left\langle P \longmapsto a<\nu y>\prec P^{\prime}\right\rangle\langle Q \longmapsto a<x\rangle \prec Q^{\prime}\right\rangle\) have \(P \| Q \longmapsto \tau \prec<\nu y>\left(P^{\prime}\right.\) \(\left.\| Q^{\prime}[x:=y]\right)\) using \(\langle y \sharp Q\rangle\)
by(rule Close2)
hence \((P \| Q)\left\|R \longmapsto \tau \prec\left(<\nu y>\left(P^{\prime} \| Q^{\prime}[x::=y]\right)\right)\right\| R\) by (rule Par1F)
moreover from \(\langle y \sharp R\rangle\) have \(\left((<\nu y\rangle\left(P^{\prime} \| Q^{\prime}[x::=y]\right)\right) \| R,\langle\nu y\rangle\left(P^{\prime} \|\right.\) \(\left.\left.\left(Q^{\prime}[x::=y] \| R\right)\right)\right) \in \operatorname{Rel}\)
by (rule FreshExt')
with \(\langle x \sharp R\rangle\) have \(\left(\left(<\nu y>\left(P^{\prime} \| Q^{\prime}[x::=y]\right)\right) \| R,<\nu y>\left(P^{\prime} \|\left(Q^{\prime} \| R\right)[x::=y]\right)\right)\) \(\in\) Rel
by (simp add: forget)
ultimately show ? case by blast
next
case (cPar2 R \({ }^{\prime}\) )
from \(\left.\langle P \longmapsto a<\nu y\rangle \prec P^{\prime}\right\rangle\langle y \sharp Q\rangle\) have \(P\left\|Q \longmapsto a<\nu y>\prec P^{\prime}\right\| Q\) by \((\) rule Par1B)
hence \((P \| Q) \| R \longmapsto \tau \prec<\nu y>\left(\left(P^{\prime} \| Q\right) \| R^{\prime}[x::=y]\right)\) using \(\langle R \longmapsto a<x>\) \(\left.\prec R^{\prime}\right\rangle\langle y \sharp R\rangle\) by (rule Close2)
moreover have \(\left(\left(P^{\prime} \| Q\right)\left\|R^{\prime}[x::=y], P^{\prime}\right\|\left(Q \| R^{\prime}[x::=y]\right)\right) \in \operatorname{Rel}\) by \((\) rule Ass)
hence \(\left(<\nu y>\left(\left(P^{\prime} \| Q\right) \| R^{\prime}[x::=y]\right),<\nu y>\left(P^{\prime} \|\left(Q \| R^{\prime}[x::=y]\right)\right)\right) \in\) Rel by (rule Res)
hence \(\left(\left\langle\nu y>\left(\left(P^{\prime} \| Q\right) \| R^{\prime}[x::=y]\right),\left\langle\nu y>\left(P^{\prime} \|\left(Q \| R^{\prime}\right)[x::=y]\right)\right) \in\right.\right.\) Rel using \(\langle x \sharp Q\rangle\) by (simp add: forget)
ultimately show? ?ase by blast
qed
qed
qed
lemma substRes3:
fixes \(a::\) name
and \(P:: p i\)
and \(x::\) name
shows \((<\nu a>P)[x:=a]=\langle\nu x\rangle([(x, a)] \cdot P)\)
proof -
have \(a \sharp<\nu a>P\) by (simp add: name-fresh-abs)
hence \((<\nu a>P)[x::=a]=[(x, a)] \cdot\langle\nu a>P\) by (rule injPermSubst[THEN sym] \()\)
thus \((\langle\nu a\rangle P)[x::=a]=\langle\nu x\rangle([(x, a)] \cdot P)\) by \((\) simp add: name-calc \()\)
qed
lemma scopeExtParLeft:
fixes \(P:: p i\)
and \(Q:: p i\)
and \(a\) :: name
and lst :: name list
and Rel \(::(p i \times p i)\) set
```

    assumes }x\sharp
    and Id: }\quadId\subseteqRe
    and EqvtRel: eqvt Rel
    and Res: }\quad\RSy.y\sharpR\Longrightarrow(<\nuy>(R|S),R|<\nuy>S)\inRe
    and ScopeExt: }\RS\mathrm{ y z. y #R C (< vy>< < z>>(R|S),< |z>(R|
    <\nuy>S)) \in Rel
shows <\nux>(P|Q)\rightsquigarrow[Rel] P|<\nux>QQ
using <eqvt Rel`
proof(induct rule: simCasesCont[where C=(x,P,Q)])
case(Bound a y PxQ)
from}\langley\sharp(x,P,Q)\rangle have y\not=x and y\sharpP and y\sharpQ by simp
hence }y\sharpP\mathrm{ and }y\sharp<\nux>Q by(simp add: abs-fresh)
with \langleP|<\nux>Q \longmapstoa«y» \precPxQ\rangle show ?case
proof(induct rule: parCasesB)
case(cPar1 P')
from \langleP\longmapstoa<y» \prec P '>\langlex\sharpP\rangle\langley\not=x\rangle have }x\sharpa\mathrm{ and }x\sharp\mp@subsup{P}{}{\prime
by(force intro: freshBoundDerivative)+
from〈P\longmapstoa<y>< P'〉\langley\sharpQ> have }P||Q\longmapstoa<y»\prec\mp@subsup{P}{}{\prime}|| Q by(rul
Par1B)
with \langlex\sharpa\rangle\langley\not=x\rangle have <\nux>>(P|Q)\longmapsto \longmapstoa«y>}\prec<\nux>(\mp@subsup{P}{}{\prime}|Q) by(rule-ta
ResB) auto
moreover have derivative (<\nux> (\mp@subsup{P}{}{\prime}|Q)) (\mp@subsup{P}{}{\prime}|<\nux>Q) a y Rel
proof(cases a, auto simp add: derivative-def)
fix }
show }((<\nux>(\mp@subsup{P}{}{\prime}|Q))[y::=u],\quad\mp@subsup{P}{}{\prime}[y::=u]|\quad((<\nux>Q)[y::=u]))\in\operatorname{Rel
proof(cases x=u)
case True
have }(<\nux>(\mp@subsup{P}{}{\prime}|Q))[y::=x]=<\nuy>(([(y,x)]\cdot\mp@subsup{P}{}{\prime})|([(y,x)]\cdotQ)
by(simp add: substRes3)
moreover from <x\sharp P'> have P}\mp@subsup{P}{}{\prime}[y::=x]=[(y,x)]\cdot\mp@subsup{P}{}{\prime}\mathbf{by}(rule in
jPermSubst[THEN sym])
moreover have (<\nux>>Q)[y::=x]=<\nuy>([(y,x)] \cdotQ) by(rule substRes3)

```

```

name-fresh-left name-calc)
ultimately show ?thesis using <x =u>>\mathbf{b}(force intro: Res)
next
case False
with \langley \not= x\rangle have (<\nux>(\mp@subsup{P}{}{\prime}|Q))[y::=u]=<\nux>(\mp@subsup{P}{}{\prime}[y::=u]|Q[y::=u])
by(simp add: fresh-prod name-fresh)
moreover from \langlex\not=u\rangle\langley\not= x\rangle have (<\nux>>Q)[y::=u]=<\nux> (Q[y::=u])
by(simp add: fresh-prod name-fresh)
moreover from }\langlex\sharp\mp@subsup{P}{}{\prime}\rangle\langlex\not=u\rangle\mathrm{ have }x\sharp\mp@subsup{P}{}{\prime}[y::=u] \mathbf{ by}(simp add: fresh-fact1
ultimately show ?thesis by(force intro: Res)
qed
next
from <x\sharp 敖〉 show (<\nux>(\mp@subsup{P}{}{\prime}|Q),\mp@subsup{P}{}{\prime}|<\nux>Q)\inRel by(rule Res)

```
qed
ultimately show ?case by blast
next
case \((c\) Par2 \(x Q)\)
from \(\langle\langle\nu x\rangle Q \longmapsto a « y » \prec x Q\rangle\langle y \neq x\rangle\langle y \sharp Q\rangle\) show ?case
proof (induct rule: resCasesB)
case (cOpen a \(Q^{\prime}\) )
from \(\left\langle Q \longmapsto a[x] \prec Q^{\prime}\right\rangle\langle y \sharp Q\rangle\) have \(y\) Fresh \(Q^{\prime}: y \sharp Q^{\prime}\) by (force intro: freshFreeDerivative)
from \(\left\langle Q \longmapsto a[x] \prec Q^{\prime}\right\rangle\) have \(P\|Q \longmapsto a[x] \prec P\| Q^{\prime}\) by (rule Par2F)
hence \(\langle\nu x\rangle(P \| Q) \longmapsto a<\nu x\rangle \prec P \| Q^{\prime}\) using \(\langle a \neq x\rangle \mathbf{b y}(\) rule Open)
with \(\langle y \sharp P\rangle\left\langle y \sharp Q^{\prime}\right\rangle\) have \(<\nu x>(P \| Q) \longmapsto a<\nu y>\prec[(x, y)] \cdot\left(P \| Q^{\prime}\right)\)
by(subst alphaBoundResidual[where \(\left.x^{\prime}=x\right]\) ) (auto simp add: fresh-left calc-atm)
with \(\langle y \sharp P\rangle\langle x \sharp P\rangle\) have \(<\nu x>(P \| Q) \longmapsto a<\nu y>\prec P \|\left([(x, y)] \cdot Q^{\prime}\right)\)
by (simp add: name-fresh-fresh)
moreover have derivative \(\left(P \|\left([(x, y)] \cdot Q^{\prime}\right)\right)\left(P \|\left([(y, x)] \cdot Q^{\prime}\right)\right)\)
(BoundOutputS a) y Rel using Id
by (auto simp add: derivative-def name-swap)
ultimately show ?case by blast
next
case (cRes \(\left.Q^{\prime}\right)\)
from 〈 \(Q \longmapsto a « y\) » \(\left.\prec Q^{\prime}\right\rangle\langle y \sharp P\rangle\) have \(P\|Q \longmapsto a « y » \prec P\| Q^{\prime}\) by (rule Par2B)
hence \(\langle\nu x\rangle(P \| Q) \longmapsto a « y » \prec<\nu x>\left(P \| Q^{\prime}\right)\) using \(\langle x \sharp a\rangle\langle y \neq x\rangle\) by (rule-tac ResB) auto
moreover have derivative \(\left(<\nu x>\left(P \| Q^{\prime}\right)\right)\left(P \|<\nu x>Q^{\prime}\right)\) a y Rel
proof (cases a, auto simp add: derivative-def)
fix \(u\)
show \(\left(\left(<\nu x>\left(P \| Q^{\prime}\right)\right)[y::=u], \quad P[y::=u] \|\left(<\nu x>Q^{\prime}\right)[y::=u]\right) \in\) Rel
proof (cases \(x=u\) )
case True
from \(\langle x \sharp P\rangle\langle y \sharp P\rangle\) have \(\left.(<\nu x\rangle\left(P \| Q^{\prime}\right)\right)[y:=x]=<\nu y>(P \|([(y, x)]\)
- \(\left.\left.Q^{\prime}\right)\right)\)
by (simp add: substRes3 perm-fresh-fresh)
moreover from \(\langle y \sharp P\rangle\) have \(P[y::=x]=P\) by (simp add: forget)
moreover have \(\left.(<\nu x\rangle Q^{\prime}\right)[y::=x]=<\nu y>\left([(y, x)] \cdot Q^{\prime}\right)\) by (rule substRes3) ultimately show ?thesis using \(\langle x=u\rangle\langle y \sharp P\rangle\) by (force intro: Res)
next
case False
with \(\langle y \neq x\rangle\) have \(\left.\left.(<\nu x\rangle\left(P \| Q^{\prime}\right)\right)[y::=u]=<\nu x\right\rangle\left(\left(P \| Q^{\prime}\right)[y::=u]\right)\) by (simp add: fresh-prod name-fresh)
moreover from \(\langle y \neq x\rangle\langle x \neq u\rangle\) have \(\left(\langle\nu x\rangle Q^{\prime}\right)[y::=u]=\langle\nu x\rangle\left(Q^{\prime}[y::=u]\right)\) by (simp add: fresh-prod name-fresh)
moreover from \(\langle x \sharp P\rangle\langle x \neq u\rangle\) have \(x \sharp P[y::=u]\) by (force simp add: fresh-fact1)
ultimately show ?thesis by (force intro: Res)
qed
next
from \(\langle x \sharp P\rangle\) show \(\left(<\nu x>\left(P \| Q^{\prime}\right), P \|<\nu x>Q^{\prime}\right) \in \operatorname{Rel}\) by(rule Res)
qed
ultimately show ?case by blast
qed
qed
next
case (Free \(\alpha\) Px \(Q\) )
from \(\langle P \|<\nu x\rangle Q \longmapsto \alpha \prec P x Q\rangle\) show ?case
proof (induct rule: parCasesF[where \(C=x]\) )
case (cPar1 \(P^{\prime}\) )
from \(\left\langle P \longmapsto \alpha \prec P^{\prime}\right\rangle\left\langle x \sharp P>\right.\) have \(x \sharp \alpha\) and \(x \sharp P^{\prime}\) by (force intro: freshFreeD-
erivative) +
from \(\left\langle P \longmapsto \alpha \prec P^{\prime}\right\rangle\) have \(P\left\|Q \longmapsto \alpha \prec P^{\prime}\right\| Q\) by (rule Par1F)
hence \(<\nu x\rangle(P \| Q) \longmapsto \alpha \prec<\nu x\rangle\left(P^{\prime} \| Q\right)\) using \(\langle x \sharp \alpha\rangle\) by (rule ResF)
moreover from \(\left\langle x \sharp P^{\prime}\right\rangle\) have \(\left.\left(\langle\nu x\rangle\left(P^{\prime} \| Q\right), P^{\prime} \|<\nu x\right\rangle Q\right) \in\) Rel by (rule Res)
ultimately show ?case by blast
next
case (cPar2 \(Q^{\prime}\) )
from \(\left\langle<\nu x>Q \longmapsto \alpha \prec Q^{\prime}\right\rangle\) show ?case
proof (induct rule: resCasesF)
case (cRes \(Q^{\prime}\) )
from \(\left\langle Q \longmapsto \alpha \prec Q^{\prime}\right.\) ’ have \(P\|Q \longmapsto \alpha \prec P\| Q^{\prime}\) by (rule Par2F)
hence \(<\nu x>(P \| Q) \longmapsto \alpha \prec<\nu x>\left(P \| Q^{\prime}\right)\) using \(\langle x \sharp \alpha\rangle\) by (rule ResF)
moreover from \(\langle x \sharp P\rangle\) have \(\left.(<\nu x\rangle\left(P \| Q^{\prime}\right), P \|<\nu x>Q^{\prime}\right) \in \operatorname{Rel}\) by \((\) rule
Res)
ultimately show ?case by blast
qed
next
case (cComm1 \(P^{\prime} x Q\) aby)
from \(\langle y \sharp x\rangle\) have \(y \neq x\) by simp
from \(\left.\langle P \longmapsto a<y\rangle \prec P^{\prime}\right\rangle\langle x \sharp P\rangle\langle y \neq x\rangle\) have \(x \sharp P^{\prime}\) by(force intro: freshBoundDerivative)
from \(\langle<\nu x\rangle Q \longmapsto a[b] \prec x Q\rangle\) show ?case
proof (induct rule: resCasesF)
case (cRes \(Q^{\prime}\) )
from \(\langle x \sharp a[b]\rangle\) have \(x \neq b\) by simp
from \(\left\langle P \longmapsto a<y>\prec P^{\prime}\right\rangle\left\langle Q \longmapsto a[b] \prec Q^{\prime}\right\rangle\) have \(P \| Q \longmapsto \tau \prec P^{\prime}[y::=b]\)
\(\| Q^{\prime} \mathbf{b y}(\) rule Comm1)
hence \(<\nu x>(P \| Q) \longmapsto \tau \prec<\nu x>\left(P^{\prime}[y::=b] \| Q^{\prime}\right)\) by (rule-tac ResF) auto
moreover from \(\left\langle x \sharp P^{\prime}\right\rangle\langle x \neq b\rangle\) have \(x \sharp P^{\prime}[y::=b]\) by (force intro: fresh-fact1)
hence \(\left(<\nu x>\left(P^{\prime}[y::=b] \| Q^{\prime}\right), P^{\prime}[y::=b] \|<\nu x>Q^{\prime}\right) \in\) Rel by(rule Res)
ultimately show ?case by blast
qed

\section*{next}
case \(\left(c C o m m 2 P^{\prime} x Q\right.\) a by）
from \(\langle y \sharp x\rangle\langle y \sharp<\nu x\rangle Q\rangle\) have \(y \neq x\) and \(y \sharp Q\) by（simp add：abs－fresh）+
with \(\langle<\nu x>Q \longmapsto a<y>\prec x Q\rangle\) show ？case
proof（induct rule：resCasesB）
case（cOpen b \(Q^{\prime}\) ）
from 〈InputS \(a=\) BoundOutputS \(b\) 〉have False by simp
thus ？case by simp
next case \(\left(\right.\) cRes \(\left.Q^{\prime}\right)\)
from \(\left\langle P \longmapsto a[b] \prec P^{\prime}\right\rangle\left\langle Q \longmapsto a<y>\prec Q^{\prime}\right\rangle\) have \(P\left\|Q \longmapsto \tau \prec P^{\prime}\right\|\) \(Q^{\prime}[y::=b] \mathbf{b y}(\) rule Comm2）
hence \(<\nu x>(P \| Q) \longmapsto \tau \prec<\nu x>\left(P^{\prime} \| Q^{\prime}[y::=b]\right)\) by（rule－tac ResF）auto
moreover from \(\left\langle P \longmapsto a[b] \prec P^{\prime}\right\rangle\langle x \sharp P\rangle\) have \(x \sharp P^{\prime}\) and \(x \neq b\) by（force
dest：freshFreeDerivative）+
from \(\left\langle x \sharp P^{\prime}\right\rangle\) have \(\left(<\nu x>\left(P^{\prime} \| Q^{\prime}[y::=b]\right), P^{\prime} \|<\nu x>\left(Q^{\prime}[y::=b]\right)\right) \in\) Rel by（rule Res）
with \(\langle y \neq x\rangle\langle x \neq b\rangle\) have \(\left(<\nu x>\left(P^{\prime} \| Q^{\prime}[y::=b]\right), P^{\prime} \|\left(<\nu x>Q^{\prime}\right)[y::=b]\right)\) \(\in\) Rel by \(\operatorname{simp}\)
ultimately show ？case by blast
qed
next
case \(\left(c\right.\) Close \(1 P^{\prime} Q^{\prime}\) a \(\left.y z\right)\)
from \(\langle y \sharp x\rangle\) have \(y \neq x\) by simp
from \(\langle z \sharp x\rangle\langle z \sharp<\nu x\rangle Q\rangle\) have \(z \sharp Q\) and \(z \neq x\) by（simp add：abs－fresh）+
from \(\left\langle P \longmapsto a<y>\prec P^{\prime}\right\rangle\langle z \sharp P\rangle\) have \(z \neq a \mathbf{b y}\)（force dest：freshBoundDeriva－ tive）
from \(\left.\langle<\nu x\rangle Q \longmapsto a<\nu z\rangle \prec Q^{\prime}\right\rangle\langle z \neq x\rangle\langle z \sharp Q\rangle\) show ？case
proof（induct rule：resCasesB）
case \(\left(c\right.\) Open \(b Q^{\prime}\) ）
from 〈BoundOutputS \(a=\) BoundOutputS \(b\rangle\) have \(a=b\) by simp
with \(\left\langle Q \longmapsto b[x] \prec Q^{\prime}\right\rangle\) have \(([(z, x)] \cdot Q) \longmapsto[(z, x)] \cdot\left(a[x] \prec Q^{\prime}\right)\)
by（rule－tac transitions．eqvt）simp
with \(\langle b \neq x\rangle\langle z \neq a\rangle\langle a=b\rangle\langle z \neq x\rangle\) have \(([(z, x)] \cdot Q) \longmapsto a[z] \prec([(z, x)]\) －\(Q^{\prime}\) ）
by（simp add：name－calc eqvts）
with \(\left\langle P \longmapsto a<y>\prec P^{\prime}\right\rangle\) have \(P\left\|([(z, x)] \cdot Q) \longmapsto \tau \prec P^{\prime}[y::=z]\right\|([(z, x)]\) －\(\left.Q^{\prime}\right)\)
by（rule Comm1）
hence \(<\nu z>(P \|([(x, z)] \cdot Q)) \longmapsto \tau \prec<\nu z>\left(P^{\prime}[y::=z] \|\left([(z, x)] \cdot Q^{\prime}\right)\right)\)
by（rule－tac ResF）auto
hence \(<\nu x>(P \| Q) \longmapsto \tau \prec<\nu z>\left(P^{\prime}[y::=z] \|\left([(z, x)] \cdot Q^{\prime}\right)\right)\) using \(\langle z \sharp\) \(P\rangle\langle z \sharp Q\rangle\langle x \sharp P\rangle\)
by（subst alphaRes［where \(c=z])\) auto
with \(I d\) show ？case by force
next
case（cRes \(Q^{\prime}\) ）
from \(\left\langle P \longmapsto a<y>\prec P^{\prime}\right\rangle\left\langle Q \longmapsto a<\nu z>\prec Q^{\prime}\right\rangle\langle z \sharp P\rangle\) have \(P \| Q \longmapsto \tau \prec\) \(<\nu z>\left(P^{\prime}[y::=z] \| Q^{\prime}\right)\)
```

        by(rule Close1)
    hence <\nux>(P|Q)\longmapsto\tau\prec<\nux><\nuz>(\mp@subsup{P}{}{\prime}[y::=z]| Q') by(rule-tac ResF)
    auto

```

```

                by(force dest: freshBoundDerivative)
    with }\langlez\not=x\rangle\mathrm{ have }x\sharp\mp@subsup{P}{}{\prime}[y::=z] by(simp add: fresh-fact1
    hence (<\nux><<\nuz>(\mp@subsup{P}{}{\prime}[y::=z]| | Q}),<\nuz>(\mp@subsup{P}{}{\prime}[y::=z]|<\nux>\mp@subsup{Q}{}{\prime}))\in\operatorname{Rel
            by(rule ScopeExt)
            ultimately show ?case by blast
    qed
    next
case(cClose2 P' xQ a y z)
from \langlez\sharpx\rangle\langlez\sharp<\nux>>Q\rangle have z\not=x and z\sharpQ by(auto simp add: abs-fresh)
from }\langley\sharpx\rangle\langley\sharp<\nux\rangle\rangleQ\rangle\mathrm{ have }y\not=x\mathrm{ and }y\sharpQ\mathrm{ by(auto simp add: abs-fresh)
with }\langle<\nux>Q\longmapstoa<y>\precxQ\rangle\mathrm{ show ?case
proof(induct rule: resCasesB)
case(cOpen b Q')
from \InputS a = BoundOutputS b〉 have False by simp
thus ?case by simp
next
case(cRes Q')
from <P\longmapstoa<\nuz> \prec P'\rangle\langleQ\longmapstoa<y> \prec Q'\rangle\langlez\sharpQ> have P|| Q\longmapsto \longmapsto |
\prec<\nuz>(P'| Q [ [y::=z])
by(rule Close2)
hence <\nux>
by(rule-tac ResF) auto
moreover from }\langleP\longmapstoa<\nuz>\prec\mp@subsup{P}{}{\prime}\rangle\langlex\sharpP\rangle\langlez\not=x\rangle\mathrm{ have }x\sharp\mp@subsup{P}{}{\prime}\mathrm{ by(force
dest: freshBoundDerivative)
hence (<\nux><<\nuz>(\mp@subsup{P}{}{\prime}|(\mp@subsup{Q}{}{\prime}[y::=z])),<\nuz>(\mp@subsup{P}{}{\prime}|(<\nux>(\mp@subsup{Q}{}{\prime}[y::=z]))))\in\mathrm{ Rel}
by(rule ScopeExt)
with «z\not= x\rangle\langley \not= x> have (<\nux><\nuz>(\mp@subsup{P}{}{\prime}|(\mp@subsup{Q}{}{\prime}[y::=z])),<\nuz>(\mp@subsup{P}{}{\prime}|
(<\nux>溃)[y::=z])) \in Rel
by simp
ultimately show ?case by blast
qed
qed
qed
lemma scopeExtParRight:
fixes P :: pi
and }Q :: p
and a :: name
and Rel ::(pi\timespi) set
assumes }x\sharp
and Id: }\quadId\subseteq\mathrm{ Rel
and eqvt Rel
and Res: \}\quad\RSy.y\sharpR\Longrightarrow(R|<\nuy>S,<\nuy>(R|S))\inRe
and ScopeExt: \bigwedgeR S yz.y\sharpR\Longrightarrow(<\nuz>(R|<\nuy>S),<\nuy><\nuz>>(R|

```
\(S)) \in R e l\)
shows \(P \|<\nu x>Q \rightsquigarrow[R e l]<\nu x>(P \| Q)\)
using 〈eqvt Rel〉
proof \((\) induct rule: \(\operatorname{simCasesCont}[\) where \(C=(x, P, Q)])\)
case(Bound a y \(x P Q\) )
from \(\langle y \sharp(x, P, Q)\rangle\) have \(y \neq x\) and \(y \sharp P\) and \(y \sharp Q\) by simp +
hence \(y \neq x\) and \(y \sharp P \| Q\) by(auto simp add: abs-fresh)
with \(\langle<\nu x\rangle(P \| Q) \longmapsto a « y » \prec x P Q\rangle\) show ?case
proof \((\) induct rule: resCasesB)
case (cOpen a \(P Q\) )
from \(\langle P \| Q \longmapsto a[x] \prec P Q\rangle\) show ?case
proof \((\) induct rule: parCasesF[where \(C=()])\)
case ( \(c\) Par1 \(P^{\prime}\) )
from \(\left\langle P \longmapsto a[x] \prec P^{\prime}\right\rangle\langle x \sharp P\rangle\) have \(x \neq x\) by (force dest: freshFreeDerivative)
thus ?case by simp
next
case (cPar2 \(Q^{\prime}\) )
from \(\left\langle Q \longmapsto a[x] \prec Q^{\prime}\right\rangle\langle y \sharp Q\rangle\) have \(y \sharp Q^{\prime} \mathbf{b y}(\) force dest: freshFreeDerivative)
from \(\left\langle Q \longmapsto a[x] \prec Q^{\prime}\right\rangle\langle a \neq x\rangle\) have \(<\nu x>Q \longmapsto a<\nu x>\prec Q^{\prime}\) by (rule
Open)
hence \(P\|<\nu x>Q \longmapsto a<\nu x>\prec P\| Q^{\prime}\) using \(\langle x \sharp P\rangle\) by (rule Par2B)
with \(\langle y \sharp P\rangle\left\langle y \sharp Q^{\prime}\right\rangle\langle x \sharp P\rangle\) have \(\left.P \|<\nu x\right\rangle Q \longmapsto a<\nu y>\prec([(y, x)] \cdot(P\)
\(\left.\left.\| Q^{\prime}\right)\right)\)
by (subst alphaBoundResidual \(\left[\mathbf{w h e r e} x^{\prime}=x\right]\) ) (auto simp add: fresh-left
calc-atm)
moreover with \(I d\) have derivative \(\left([(y, x)] \cdot\left(P \| Q^{\prime}\right)\right)\)
\(\left([(y, x)] \cdot\left(P \| Q^{\prime}\right)\right)\) (BoundOutputS a) y Rel
by (auto simp add: derivative-def)
ultimately show? case by blast
next
case (cComm1 \(\left.P^{\prime} Q^{\prime} b c y\right)\)
from \(\langle a[x]=\tau\rangle\) show ?case by simp
next
case (cComm2 \(\left.P^{\prime} Q^{\prime} b c y\right)\)
from \(\langle a[x]=\tau\rangle\) show ? case by simp
next
case (cClose1 \(P^{\prime} Q^{\prime}\) b y \(z\) )
from \(\langle a[x]=\tau\rangle\) show ? case by simp
next
case (cClose2 \(P^{\prime} Q^{\prime}\) b y \(z\) )
from \(\langle a[x]=\tau\rangle\) show ? case by simp
qed
next
case \((\) cRes \(P Q)\)
from \(\langle P \| Q \longmapsto a<y>\prec P Q\rangle\langle y \sharp P\rangle\langle y \sharp Q\rangle\)
show ?case
proof (induct rule: parCasesB)
case(cPar1 \(P^{\prime}\) )
from \(\langle y \neq x\rangle\langle x \sharp P\rangle\left\langle P \longmapsto a « y » \prec P^{\prime}\right\rangle\) have \(x \sharp P^{\prime}\) by (force dest: freshBoundDerivative)
from \(\left\langle P \longmapsto a « y\right.\) » \(\left.\prec P^{\prime}\right\rangle\langle y \sharp Q\rangle\) have \(P \|<\nu x>Q \longmapsto a<y\) » \(\left\langle P^{\prime} \|<\nu x>Q\right.\) by (rule-tac Par1B) (auto simp add: abs-fresh)
moreover have derivative \(\left.\left(P^{\prime} \|<\nu x>Q\right)(<\nu x\rangle\left(P^{\prime} \| Q\right)\right)\) a y Rel
proof (cases a, auto simp add: derivative-def)
fix \(u\) ::name
obtain \(z:: n a m e\) where \(z \sharp Q\) and \(y \neq z\) and \(z \neq u\) and \(z \sharp P\) and \(z \sharp P^{\prime}\)
by (generate-fresh name) auto
thus \(\left(P^{\prime}[y::=u] \|(<\nu x>Q)[y::=u],\left(<\nu x>\left(P^{\prime} \| Q\right)\right)[y::=u]\right) \in\) Rel using \(\left\langle x \sharp P^{\prime}\right\rangle\)
\(\mathbf{b y}(\) subst alphaRes \([\) where \(c=z\) and \(a=x]\), auto)
(subst alphaRes[where \(c=z\) and \(a=x]\), auto intro: Res simp add: fresh-fact1)
next
from \(\left\langle x \sharp P^{\prime}\right\rangle\) show \(\left(P^{\prime} \|<\nu x>Q,<\nu x>\left(P^{\prime} \| Q\right)\right) \in\) Rel
by (rule Res)
qed
ultimately show ?case by blast
next
case(cPar2 \(\left.Q^{\prime}\right)\)
from \(\left\langle Q \longmapsto a « y » \prec Q^{\prime}\right\rangle\) have \(\left\langle\nu x>Q \longmapsto a « y » \prec\left\langle\nu x>Q^{\prime}\right.\right.\) using \(\langle x \sharp a\rangle\) \(\langle y \neq x\rangle\)
by(rule-tac ResB) auto
hence \(P\|<\nu x>Q \longmapsto a « y » \prec P\|<\nu x>Q^{\prime}\) using \(\langle y \sharp P\rangle\) by (rule Par2B)
moreover have derivative \(\left(P \|<\nu x>Q^{\prime}\right)\left(<\nu x>\left(P \| Q^{\prime}\right)\right)\) a y Rel
proof (cases a, auto simp add: derivative-def)
fix \(u\) ::name
obtain \(z:: n a m e\) where \(z \sharp Q\) and \(z \neq y\) and \(z \neq u\) and \(z \sharp P\) and \(z \sharp Q^{\prime}\) by(generate-fresh name) auto
thus \(\left(P[y::=u] \|\left(<\nu x>Q^{\prime}\right)[y::=u],\left(<\nu x>\left(P \| Q^{\prime}\right)\right)[y::=u]\right) \in\) Rel using \(\langle x \sharp P\rangle\)
by (subst alphaRes \([\) where \(a=x\) and \(c=z]\), auto)
(subst alphaRes[where \(a=x\) and \(c=z]\), auto intro: Res simp add:
fresh-fact1)
next
from \(\langle x \sharp P\rangle\) show \(\left(P \|<\nu x>Q^{\prime},\langle\nu x\rangle\left(P \| Q^{\prime}\right)\right) \in\) Rel by (rule Res)
qed
ultimately show ?case by blast
qed
qed
next
case (Free \(\alpha x P Q\) )
```

from $\langle<\nu x>(P \| Q) \longmapsto \alpha \prec x P Q\rangle$ show ?case
proof $($ induct rule: resCases $F$ )
case (cRes $P Q$ )
from $\langle P \| Q \longmapsto \alpha \prec P Q\rangle$ show ?case
proof (induct rule: parCasesF[where $C=x]$ )
case (cPar1 $P^{\prime}$ )
from $\left\langle P \longmapsto \alpha \prec P^{\prime}\right\rangle$ have $P\left\|<\nu x>Q \longmapsto \alpha \prec P^{\prime}\right\|<\nu x>Q$ by (rule Par1F)
moreover from $\left\langle P \longmapsto \alpha \prec P^{\prime}\right\rangle\langle x \sharp P\rangle$ have $x \sharp P^{\prime}$ by (rule freshFreeD-
erivative)
hence $\left(P^{\prime} \|<\nu x>Q,<\nu x>\left(P^{\prime} \| Q\right)\right) \in$ Rel by(rule Res $)$
ultimately show ? case by blast
next
case (cPar2 $Q^{\prime}$ )
from $\left\langle Q \longmapsto \alpha \prec Q^{\prime}\right\rangle\langle x \sharp \alpha\rangle$ have $<\nu x>Q \longmapsto \alpha \prec<\nu x>Q^{\prime}$ by (rule ResF)
hence $P\|<\nu x>Q \longmapsto \alpha \prec P\|<\nu x>Q^{\prime}$ by (rule Par2F)
moreover from $\langle x \sharp P\rangle$ have $\left.(P \|<\nu x\rangle Q^{\prime},\langle\nu x\rangle\left(P \| Q^{\prime}\right)\right) \in$ Rel by $($ rule
Res)
ultimately show ?case by blast
next
case(cComm1 $P^{\prime} Q^{\prime}$ ably)
from $\left.\langle x \sharp P\rangle\langle y \sharp x\rangle\langle P \longmapsto a<y\rangle \prec P^{\prime}\right\rangle$ have $x \neq a$ and $x \sharp P^{\prime}$ by (force
dest: freshBoundDerivative) +
show ?case
proof $($ cases $b=x)$
case True
from $\left\langle Q \longmapsto a[b] \prec Q^{\prime}\right\rangle\langle x \neq a\rangle\langle b=x\rangle$ have $\left.\langle\nu x\rangle Q \longmapsto a<\nu x\right\rangle \prec Q^{\prime}$
by(rule-tac Open) auto
with $\left\langle P \longmapsto a<y>\prec P^{\prime}\right\rangle$ have $P \|<\nu x>Q \longmapsto \tau \prec<\nu x>\left(P^{\prime}[y::=x] \| Q^{\prime}\right)$
using $\langle x \sharp P\rangle$ by(rule Close1)
moreover from $I d$ have $\left(\langle\nu x\rangle\left(P^{\prime}[y::=b] \| Q^{\prime}\right),\langle\nu x\rangle\left(P^{\prime}[y::=b] \| Q^{\prime}\right)\right) \in$
Rel by blast
ultimately show ?thesis using $\langle b=x\rangle$ by blast
next
case False
from $\left\langle Q \longmapsto a[b] \prec Q^{\prime}\right\rangle\langle x \neq a\rangle\langle b \neq x\rangle$ have $\left\langle\nu x>Q \longmapsto a[b] \prec<\nu x>Q^{\prime}\right.$
by (rule-tac ResF) auto
with $\left\langle P \longmapsto a<y>\prec P^{\prime}\right\rangle$ have $P \|<\nu x>Q \longmapsto \tau \prec\left(P^{\prime}[y::=b] \|<\nu x>Q^{\prime}\right)$
by(rule Comm1)
moreover from $\left\langle x \sharp P^{\prime}\right\rangle\langle b \neq x\rangle$ have $\left(P^{\prime}[y::=b] \|<\nu x>Q^{\prime},\langle\nu x\rangle\left(P^{\prime}[y::=b]\right.\right.$
$\left.\left.\| Q^{\prime}\right)\right) \in \operatorname{Rel}$
by (force intro: Res simp add: fresh-fact1)
ultimately show ?thesis by blast
qed
next
case (cComm2 $P^{\prime} Q^{\prime}$ ably)
from $\left\langle P \longmapsto a[b] \prec P^{\prime}\right\rangle\langle x \sharp P\rangle$ have $x \neq a$ and $x \neq b$ and $x \sharp P^{\prime}$ by (force
dest: freshFreeDerivative)+
from $\left.\langle Q \longmapsto a<y\rangle \prec Q^{\prime}\right\rangle\langle y \sharp x\rangle\langle x \neq a\rangle$ have $\left.\langle\nu x\rangle Q \longmapsto a<y>\prec<\nu x\right\rangle Q^{\prime}$
by (rule-tac ResB) auto

```
```

            with }\langleP\longmapstoa[b]\prec\mp@subsup{P}{}{\prime}>\mathrm{ have }P|<\nux>Q\longmapsto\tau\prec\mp@subsup{P}{}{\prime}|(<\nux>>\mp@subsup{Q}{}{\prime})[y::=b
    by(rule Comm2)
moreover from <x\sharp | '> have ( }\mp@subsup{P}{}{\prime}|<\nux>(\mp@subsup{Q}{}{\prime}[y::=b]),<\nux>(\mp@subsup{P}{}{\prime}|\mp@subsup{Q}{}{\prime}[y::=b])
Rel by(rule Res)
ultimately show ?case using }\langley\sharpx\rangle\langlex\not=b\rangle\mathrm{ by force
next
case(cClose1 P' Q' a y z)
from \langleP\longmapstoa<y> \precP'\rangle\langlex\sharpP\rangle\langley\sharpx\rangle have }x\not=a\mathrm{ and }x\sharp\mp@subsup{P}{}{\prime}\mathrm{ by(force
dest: freshBoundDerivative)+
from <Q\longmapstoa<\nuz> \prec Q'\rangle\langlez\sharpx\rangle\langlex\not=a\rangle have <\nux> Q \longmapsto \longmapstoa<\nuz>\prec\prec
<\nux>}\mp@subsup{Q}{}{\prime}\mathrm{ by(rule-tac ResB) auto
with <P\longmapstoa<y> \prec P'> have P|<\nux>Q\longmapsto \longmapsto | < <\nuz> (P'[y::=z]|
<\nux> Q') using < z\sharpP> by(rule Close1)

```

```

| Q '))) \in Rel
by(rule-tac ScopeExt) (auto simp add: fresh-fact1)
ultimately show ?case by blast
next
case(cClose2 P' Q' a y z)

```

```

                dest: freshBoundDerivative)+
            from <Q\longmapstoa<y>\prec\prec Q'\rangle\langley\sharpx\rangle\langlex\not=a\rangle have <\nux> Q\longmapsto \longmapstoa<y><<<\nux>>\mp@subsup{Q}{}{\prime}
    by(rule-tac ResB) auto
with }\langleP\longmapstoa<\nuz>\prec\mp@subsup{P}{}{\prime}>\mathrm{ have }P|<\nux>Q\longmapsto\tau\prec<\nuz>(\mp@subsup{P}{}{\prime}|(<\nux>\mp@subsup{Q}{}{\prime})[y::=z]
using<z \#Q>
by(rule-tac Close2) (auto simp add: abs-fresh)
moreover from <x\sharp P'> have (<\nuz>( }\mp@subsup{P}{}{\prime}|<\nux>(\mp@subsup{Q}{}{\prime}[y::=z])),<\nux><\nuz>(\mp@subsup{P}{}{\prime
| Q Q}[y::=z])) \in Rel by(rule ScopeExt
ultimately show ?case using \langlez\sharpx\rangle\langley\sharpx\rangle}\mathrm{ by force
qed
qed
qed
lemma resNilRight:
fixes }x\mathrm{ :: name
and Rel :: (pi\timespi) set
shows 0}\rightsquigarrow[Rel]<\nux>
by(fastforce simp add: simulation-def pi.inject alpha' elim: resCasesB' resCasesF)
lemma resComm:
fixes a :: name
and b :: name
and }P\mathrm{ :: pi
and Rel ::(pi\timespi) set
assumes ResComm: \cd Q. (<\nuc>< < d> Q , <\nud><<\nuc>Q )\in Rel
and Id: Id\subseteqRel
and EqvtRel: eqvt Rel

```
```

    shows \(<\nu a><\nu b>P \rightsquigarrow[\) Rel \(]<\nu b><\nu a>P\)
    proof (cases $a=b$ )
assume $a=b$
with Id show ?thesis by(force intro: Strong-Late-Sim.reflexive)
next
assume aineqb: $a \neq b$
from EqutRel show ?thesis
proof $($ induct rule: simCasesCont $[$ where $C=(a, b, P)])$
case (Bound c xbaP)
from $\langle x \sharp(a, b, P)\rangle$ have $x \neq a$ and $x \neq b$ and $x \sharp P$ by simp +
from $\langle x \sharp P\rangle$ have $x \sharp<\nu a>P$ by(simp add: abs-fresh)
with $\langle<\nu b\rangle\langle\nu a\rangle P \longmapsto c « x\rangle \prec b a P\rangle\langle x \neq b\rangle$ show ?case
proof (induct rule: resCasesB)
case( $c$ Open caP)
from $\langle<\nu a\rangle P \longmapsto c[b] \prec a P\rangle$
show ?case
proof (induct rule: resCasesF)
case (cRes $P^{\prime}$ )
from $\langle a \sharp c[b]$ have $a \neq c$ and $a \neq b$ by simp+
from $\langle x \sharp P\rangle\left\langle P \longmapsto c[b] \prec P^{\prime}\right\rangle$ have $x \neq c$ and $x \sharp P^{\prime}$ by (force dest:
freshFreeDerivative)+
from $\left\langle P \longmapsto c[b] \prec P^{\prime}\right\rangle$ have $([(x, b)] \cdot P) \longmapsto[(x, b)] \cdot\left(c[b] \prec P^{\prime}\right)$ by $($ rule
transitions.eqvt)
with $\langle x \neq c\rangle\langle c \neq b\rangle\langle x \neq b\rangle$ have $([(x, b)] \cdot P) \longmapsto c[x] \prec[(x, b)] \cdot P^{\prime}$
by (simp add: eqvts calc-atm)
hence $<\nu x>([(x, b)] \cdot P) \longmapsto c<\nu x>\prec[(x, b)] \cdot P^{\prime}$ using $\langle x \neq c\rangle$
by(rule-tac Open) auto
with $\langle x \sharp P\rangle$ have $<\nu b>P \longmapsto c<\nu x\rangle \prec[(x, b)] \cdot P^{\prime}$ by $($ simp add:
alphaRes)
hence $\langle\nu a\rangle\langle\nu b\rangle P \longmapsto c<\nu x\rangle \prec\langle\nu a\rangle\left([(x, b)] \cdot P^{\prime}\right)$ using $\langle a \neq c\rangle\langle x$
$\neq a$ >
by(rule-tac ResB) auto
moreover from Id have derivative $\left(<\nu a>\left([(x, b)] \cdot P^{\prime}\right)\right)(<\nu a>([(x, b)] \cdot$
$\left.P^{\prime}\right)$ ) (BoundOutputS c) x Rel
by (force simp add: derivative-def)
ultimately show ? case using $\langle a \neq b\rangle\langle x \neq a\rangle\langle a \neq c\rangle$ by (force simp add:
equts calc-atm)
qed
next
case $(c$ Res $a P)$
from $\langle<\nu a\rangle P \longmapsto c « x\rangle \prec a P\rangle\langle x \neq a\rangle\langle x \sharp P\rangle\langle b \sharp c\rangle$ show ?case
proof (induct rule: resCasesB)
case( $c$ Open $c P^{\prime}$ )
from $\left\langle P \longmapsto c[a] \prec P^{\prime}\right\rangle\langle x \sharp P\rangle$ have $x \sharp P^{\prime}$ by(force intro: freshFreeDeriva-
tive)
from $\langle b \sharp$ BoundOutputS $c\rangle$ have $b \neq c$ by simp
with $\left\langle P \longmapsto c[a] \prec P^{\prime}\right\rangle\langle a \neq b\rangle$ have $\langle\nu b\rangle P \longmapsto c[a] \prec<\nu b>P^{\prime}$ by $($ rule-tac
ResF) auto

```
```

    with }\langlec\not=a\rangle\mathrm{ have < < a>< < b>Pط
    auto
hence }<\nua><\nub>P\longmapstoc<\nux>\prec<\nub>([(x,a)]\cdot\mp@subsup{P}{}{\prime})\mathrm{ using <x = b><a\#=
b\rangle\langlex\sharp P
apply(subst alphaBoundResidual[where \mp@subsup{x}{}{\prime}=a]) by(auto simp add: abs-fresh
fresh-left calc-atm)
moreover have derivative (<\nub>([(x,a)] \cdot P'))(<\nub>([(x,a)] \cdot P
(BoundOutputS c) x Rel using Id
by(force simp add: derivative-def)
ultimately show ?case by blast
next
case(cRes P')
from \langleP\longmapstoc«x\rangle\prec < P'\rangle\langleb\sharpc\rangle\langlex\not=b\rangle have <\nub>P\longmapsto
by(rule-tac ResB) auto
hence <\nua><\nub>P\longmapstoc<ux> \prec <\nua><\nub> P' using<a\sharpc\rangle\langlex\not=a\rangle
by(rule-tac ResB) auto
moreover have derivative ( <\nua><\nu\nub>P') (<\nub><\nu\nua>\mp@subsup{P}{}{\prime}) с x Rel
proof(cases c, auto simp add: derivative-def)
fix u::name
show ((<\nua><\nub>\mp@subsup{P}{}{\prime})[x::=u], (<\nub><\nua>\mp@subsup{P}{}{\prime})[x::=u])\in Rel
proof(cases u=a)
case True
from }\langleu=a\rangle\langlea\not=b\rangle\mathrm{ show ?thesis
by(subst injPermSubst[symmetric], auto simp add: abs-fresh)
(subst injPermSubst[symmetric], auto simp add: abs-fresh calc-atm
intro: ResComm)
next
case False
show ?thesis
proof(cases u=b)
case True
from }\langleu=b\rangle\langleu\not=a\rangle\mathrm{ show ?thesis
by(subst injPermSubst[symmetric], auto simp add: abs-fresh)
(subst injPermSubst[symmetric], auto simp add: abs-fresh calc-atm
intro: ResComm)
next
case False
from }\langleu\not=a\rangle\langleu\not=b\rangle\langlex\not=a\rangle\langlex\not=b\rangle\mathrm{ show ?thesis by(auto intro:
ResComm)
qed
qed
next
show (<\nua><\nu b>P P},<\nub><\nua>\mp@subsup{P}{}{\prime})\inRel by(rule ResComm
qed
ultimately show ?case by blast
qed
qed
next
case(Free \alpha baP)

```
```

    from {<\nub><<\nua>P\longmapsto 
    proof(induct rule: resCasesF)
        case(cRes aP)
        from <<\nua>P\longmapsto \longmapsto\alpha\precaP\rangle show ?case
        proof(induct rule: resCasesF)
            case(cRes P')
            from }\langleP\longmapsto\alpha\prec\mp@subsup{P}{}{\prime}\rangle\langleb\sharp\alpha>\mathrm{ have < < b>P Pط < < < b >> P' by(rule ResF)
            hence <\nua><\nub>P\longmapsto\alpha\longmapsto \prec<\nua><\nub>>\mp@subsup{P}{}{\prime}\mathrm{ using <a甘 }|>\mathbf{by}(rule ResF)
            moreover have (<\nua><\nub>\mp@subsup{P}{}{\prime},<\nub><\nua>\mp@subsup{P}{}{\prime})\inRel by(rule ResComm)
            ultimately show ?case by blast
        qed
    qed
    qed
    qed
lemma bangLeftSC:
fixes P :: pi
and Rel :: (pi\timespi) set
assumes Id \subseteqRel
shows !P\rightsquigarrow[Rel] P|!P
using assms
by(force simp add: simulation-def dest: Bang derivativeReflexive)
lemma bangRightSC:
fixes P :: pi
and Rel :: (pi\timespi) set
assumes IdRel:Id \subseteqRel
shows P|!P\rightsquigarrow[Rel]!P
using assms
by(fastforce simp add: pi.inject simulation-def intro: derivativeReflexive elim: bang-
Cases)
lemma resNilLeft:
fixes x :: name
and y :: name
and }P\mathrm{ :: pi
and Rel::(pi\timespi) set
and b :: name
shows 0}\rightsquigarrow[Rel]<\nux>(x<y>.P
and 0}\rightsquigarrow[Rel]<\nux>(x{b}.P
by(auto simp add: simulation-def)

```
```

lemma resInputLeft
fixes $x$ :: name
and $a::$ name
and $y::$ name
and $\quad P:: p i$
and Rel :: $(p i \times p i)$ set
assumes xineqa: $x \neq a$
and xineqy: $x \neq y$
and Eqvt: eqvt Rel
and $\quad I d: I d \subseteq R e l$
shows $<\nu x>a<y>. P \rightsquigarrow[\operatorname{Rel}] a<y>.(<\nu x>P)$
using Eqvt
$\operatorname{proof}($ induct rule: simCasesCont $[$ where $C=(x, y, a, P)])$
case(Bound b $z P^{\prime}$ )
from $\langle z \sharp(x, y, a, P)\rangle$ have $z \neq x$ and $z \neq y$ and $z \sharp P$ and $z \neq a$ by simp +
from $\langle z \sharp P\rangle$ have $z \sharp<\nu x\rangle P$ by (simp add: abs-fresh)
with $\left.\langle a<y\rangle .(<\nu x\rangle P) \longmapsto b « z\rangle \prec P^{\prime}\right\rangle\langle z \neq a\rangle\langle z \neq y\rangle$ show ?case
proof (induct rule: inputCases)
case cInput
have $a<y>. P \longmapsto a<y>\prec P$ by (rule Input)
with $\langle x \neq y\rangle\langle x \neq a\rangle$ have $<\nu x>a<y>. P \longmapsto a<y>\prec<\nu x>P$ by (rule-tac
ResB) auto
hence $\langle\nu x\rangle a<y>. P \longmapsto a<z\rangle \prec[(y, z)] \cdot\langle\nu x>P$ using $\langle z \sharp P\rangle$
by (subst alphaBoundResidual[where $\left.x^{\prime}=y\right]$ ) (auto simp add: abs-fresh fresh-left
calc-atm)
moreover from $I d$ have derivative $([(y, z)] \cdot<\nu x>P)([(y, z)] \cdot<\nu x>P)$
(InputS a) z Rel
by(rule derivativeReflexive)
ultimately show ?case by blast
qed
next
case (Free $\alpha P^{\prime}$ )
from $\left.\langle a<y\rangle .(<\nu x\rangle P) \longmapsto \alpha \prec P^{\prime}\right\rangle$ have False by auto
thus? ?case by simp
qed
lemma resInputRight:
fixes $a$ :: name
and $y$ :: name
and $x::$ name
and $\quad P:: p i$
and Rel $::(p i \times p i)$ set
assumes xineqa: $x \neq a$
and xineqy: $x \neq y$
and Eqvt: eqvt Rel
and $\quad I d: I d \subseteq$ Rel

```
```

    shows }a<y>.(<\nux>P)\rightsquigarrow[Rel]<\nux>a<y>.
    using Eqvt
    proof(induct rule: simCasesCont [where C=(x,y,a,P)])
case(Bound b z xP)
from }\langlez\sharp(x,y,a,P)\rangle\mathrm{ have }z\not=x\mathrm{ and }z\not=y\mathrm{ and }z\sharpP\mathrm{ and }z\not=a\mathrm{ by simp+
from }\langlez\not=a\rangle\langlez\sharpP\rangle\mathrm{ have z\#a<y>.P by(simp add: abs-fresh)
with <<\nux>a<y>.P\longmapstob }\longmapsto~z>\precxP\rangle\langlez\not=x\rangle\mathrm{ show ?case
proof(induct rule: resCasesB)
case(cOpen b P')
from }\langlea<y>.P\longmapstob[x]\prec\mp@subsup{P}{}{\prime}\rangle\mathrm{ have False by auto
thus ?case by simp
next
case(cRes P')

```

```

    proof(induct rule: inputCases)
            case cInput
            have }a<y>.(<\nux>P)\longmapstoa<y>\prec(<\nux>P) by(rule Input
            with }\langlez\sharpP\rangle\langlex\not=y\rangle\langlez\not=x\rangle\mathrm{ have }a<y>.(<\nux>P)\longmapstoa<z>\prec<(<\nux>([(y
    z)] • P))
by(subst alphaBoundResidual[where x'=y])(auto simp add: abs-fresh
calc-atm fresh-left)
moreover from Id have derivative (<\nux> ([(y,z)] \cdot P)) (<\nux> ([(y,z)] .
P)) (InputS a) z Rel
by(rule derivativeReflexive)
ultimately show ?case by blast
qed
qed
next
case(Free \alpha P}\mp@subsup{P}{}{\prime}
from <<\nux>a<y>.P\longmapsto\alpha\prec P'> have False by auto
thus ?case by simp
qed
lemma resOutputLeft:
fixes x :: name
and a :: name
and b :: name
and }P\mathrm{ :: pi
and Rel :: (pi\timespi) set
assumes xineqa: x\not=a
and xineqb: }x\not=
and Id:Id\subseteq Rel
shows <\nux>a{b}.P\rightsquigarrow[Rel] a{b}.(<\nux>PP)
using assms
by(fastforce simp add: simulation-def elim: outputCases intro:Output ResF)

```
```

lemma resOutputRight:
fixes x :: name
and a :: name
and b :: name
and }P\mathrm{ :: pi
and Rel :: (pi\timespi) set
assumes xineqa: x\not=a
and xineqb: }x\not=
and Id:Id\subseteq Rel
and Eqvt: eqvt Rel
shows a{b}.(<\nux>P)\rightsquigarrow[Rel]<\nux>a{b}.P
using assms
by(erule-tac simCasesCont[where C=x])
(force simp add: abs-fresh elim: resCasesB resCasesF outputCases intro: ResF
Output)+
lemma resTauLeft:
fixes }x\mathrm{ :: name
and }P\mathrm{ :: pi
and Rel :: (pi\timespi) set
assumes Id:Id\subseteq Rel
shows <\nux>(\tau.(P))\rightsquigarrow[Rel] \tau.(<\nux>P)
using assms
by(force simp add: simulation-def elim: tauCases resCasesF intro: Tau ResF)
lemma resTauRight:
fixes x :: name
and }P\mathrm{ :: pi
and Rel :: (pi\timespi) set
assumes Id: Id\subseteqRel
shows }\tau.(<\nux>P)\rightsquigarrow[Rel]<\nux>(\tau.(P)
using assms
by(force simp add: simulation-def elim: tauCases resCasesF intro: Tau ResF)
end
theory Strong-Late-Bisim-SC
imports Strong-Late-Bisim-Pres Strong-Late-Sim-SC
begin
lemma nilBisim[dest]:
fixes a :: name
and b:: name

```
```

    and x :: name
    and }P\mathrm{ :: pi
    shows }\tau.(P)~\mathbf{0}\Longrightarrow\mathrm{ False
    and a<x>.P~00\Longrightarrow False
    and }a{b}.P~\mathbf{0}\Longrightarrow\mathrm{ False
    and 0}~\tau.(P)\Longrightarrow\mathrm{ False
    and 0 ~ a<x>.P\Longrightarrow False
    and 0}~a{b}.P\Longrightarrow\mathrm{ False
    by(auto dest: bisimE symmetric)
lemma matchId:
fixes a :: name
and }P::p
shows [a\frowna]P~P
proof -
let ?X = {([a\frowna]P,P),(P,[a\frowna]P)}
have }([a\frowna]P,P)\in?X by sim
thus ?thesis
by(coinduct rule: bisimCoinduct) (auto intro: matchIdLeft matchIdRight reflex-
ive)
qed
lemma matchNil:
fixes a :: name
and b :: name
assumes }a\not=
shows [a\frownb]P~0
proof -
let ?X }={([a\frownb]P,\mathbf{0}),(\mathbf{0},[a\frownb]P)
have }([a\frownb]P,0)\in?X by sim
thus ?thesis using < }a\not=b
by(coinduct rule: bisimCoinduct) (auto intro: matchNilLeft nilSimRight reflex-
ive)
qed
lemma mismatchId:
fixes a :: name
and b:: name
and }P::p
assumes }a\not=
shows [a\not=b]P~P

```
```

proof -
let ?X = {([a\not=b]P,P),(P,[a\not=b]P)}
have }([a\not=b]P,P)\in?X by sim
thus ?thesis using < }a\not=b
by(coinduct rule: bisimCoinduct) (auto intro: mismatchIdLeft mismatchIdRight
reflexive)
qed
lemma mismatchNil:
fixes a :: name
and P:: pi
shows [a\not=a]P~0
proof -
let ?X = {([a\not=a]P,0), (0,[a\not=a]P)}
have }([a\not=a]P,0)\in?X by sim
thus ?thesis
by(coinduct rule: bisimCoinduct) (auto intro: mismatchNilLeft nilSimRight re-
flexive)
qed

```
lemma nilRes:
    fixes \(x\) :: name
    shows \(<\nu x>\mathbf{0} \sim \mathbf{0}\)
proof -
    let ? \(X=\{(<\nu x>\mathbf{0}, \mathbf{0}),(\mathbf{0},<\nu x>\mathbf{0})\}\)
    have \((<\nu x>\mathbf{0}, \mathbf{0}) \in\) ? \(X\) by simp
    thus? ?hesis
        by (coinduct rule: bisimCoinduct) (auto intro: nilSimRight resNilRight)
qed
lemma resComm:
    fixes \(x\) :: name
    and \(y::\) name
    and \(\quad P:: p i\)
    shows \(<\nu x><\nu y>P \sim<\nu y><\nu x>P\)
proof -
    let ? \(X=\{(<\nu x><\nu y>P,<\nu y><\nu x>P) \mid x y P\). True \(\}\)
    have \((<\nu x><\nu y>P,<\nu y><\nu x>P) \in ? X\) by auto
    thus ? thesis
    proof(coinduct rule: bisimCoinduct)
        case ( \(c\) Sim \(x y P\) yxP)
        \{
            fix \(x\) y \(P\)
            have \(\bigwedge x\) y \(P .(<\nu x><\nu y>P,<\nu y><\nu x>P) \in ? X \cup\) bisim by auto
            moreover have \(I d \subseteq\) ? \(X \cup\) bisim by(auto intro: reflexive)
```

        moreover have eqvt ?X by(force simp add: eqvt-def)
        hence eqvt(?X U bisim) by auto
            ultimately have <\nux><<\nuy>P}\rightsquigarrow[(?X\cup\mathrm{ bisim )] < < y >< < x > P by(rule
    resComm)
}
with <(xyP, yxP) \in?X> show ?case by auto
next
case(cSym xyP yxP)
thus ?case by auto
qed
qed
lemma sumSym:
fixes }P:: p
and }Q::p
shows }P\oplusQ~Q\oplus
proof -
let ? }X={(P\oplusQ,Q\oplusP),(Q\oplusP,P\oplusQ)
have }(P\oplusQ,Q\oplusP)\in?X by sim
thus ?thesis
by(coinduct rule: bisimCoinduct) (auto intro: reflexive sumSym)
qed
lemma sumIdemp:
fixes P :: pi
shows }P\oplusP~
proof -
let ?X = {(P\oplusP,P),(P,P\oplusP)}
have }(P\oplusP,P)\in?X by sim
thus ?thesis
by(coinduct rule: bisimCoinduct) (auto intro: reflexive sumIdempLeft sumIdem-
pRight)
qed
lemma sumAssoc:
fixes }P:: p
and }Q::p
and }R::p
shows }(P\oplusQ)\oplusR~P\oplus(Q\oplusR
proof -
let ? }X={((P\oplusQ)\oplusR,P\oplus(Q\oplusR)),(P\oplus(Q\oplusR),(P\oplusQ)\oplusR)
have }((P\oplusQ)\oplusR,P\oplus(Q\oplusR))\in?X by sim
thus ?thesis
by(coinduct rule: bisimCoinduct) (auto intro: reflexive sumAssocLeft sumAs-

```
```

socRight)
qed
lemma sumZero:
fixes P :: pi
shows }P\oplus\mathbf{0}~
proof -
let ? }X={(P\oplus\mathbf{0},P),(P,P\oplus\mathbf{0})
have}(P\oplus\mathbf{0},P)\in?X by sim
thus ?thesis
by(coinduct rule: bisimCoinduct) (auto intro: reflexive sumZeroLeft sumZeroR-
ight)
qed

```
lemma parZero:
    fixes \(P:: p i\)
    shows \(P \| \mathbf{0} \sim P\)
proof -
    let ? \(X=\{(P \| \mathbf{0}, P) \mid P\). True \(\} \cup\{(P, P \| \mathbf{0}) \mid P\). True \(\}\)
    have \((P \| \mathbf{0}, P) \in\) ? \(X\) by blast
    thus ? thesis
        by (coinduct rule: bisimCoinduct, auto intro: parZeroRight parZeroLeft)
qed
lemma parSym:
    fixes \(P\) :: pi
    and \(\quad Q:: p i\)
    shows \(P\|Q \sim Q\| P\)
proof -
    let \(? X=\{(\) resChain lst \((P \| Q)\), resChain lst \((Q \| P)) \mid\) lst \(P Q\). True \(\}\)
    have \((P\|Q, Q\| P) \in ? X \operatorname{by}\) (blast intro: resChain.base[THEN sym])
    thus ?thesis
    proof (coinduct rule: bisimCoinduct)
        case \((c \operatorname{Sim} P Q \quad Q P)\)
        \{
            fix lst \(P Q\)
            have \(\bigwedge P Q .(P\|Q, Q\| P) \in ? X \cup\) bisim by (blast intro: resChain.base[THEN
sym])
            moreover have Res: \(\bigwedge x P Q .(P, Q) \in ? X \cup\) bisim \(\Longrightarrow(<\nu x>P,<\nu x>Q)\)
\(\in ? X \cup\) bisim
            by (auto intro: resPres resChain.step \([\) THEN sym])
        ultimately have \(P \| Q \rightsquigarrow[(? X \cup\) bisim \()] Q \| P\) by (rule parSym \()\)
        moreover have eqvt? \(X\) by (force simp add: eqvt-def)
        hence eqvt \((? X \cup\) bisim) by auto
ultimately have resChain lst \((P \| Q) \rightsquigarrow[(? X \cup\) bisim \()]\) resChain lst \((Q \|\) \(P)\) using Res
```

            by(rule resChainI)
    ```
    \}
    with \(\langle(P Q, Q P) \in\) ? \(X\) 〉 show ?case by auto
    next
        case \((c S y m P Q Q P)\)
        thus ?case by auto
    qed
qed
lemma scopeExtPar:
    fixes \(P:: p i\)
    and \(\quad Q:: p i\)
    and \(\quad x::\) name
    assumes \(x \sharp P\)
    shows \(<\nu x>(P \| Q) \sim P \|<\nu x>Q\)
proof -
    let ? \(X=\{(\) resChain lst \((<\nu x>(P \| Q))\), resChain lst \((P \|<\nu x>Q)) \mid\) lst \(x P\)
Q. \(x \sharp P\} \cup\)
                            \(\{(\) resChain lst \((P \|<\nu x>Q)\), resChain lst \((<\nu x>(P \| Q))) \mid\) lst x \(P Q\).
\(x \sharp P\}\)
    let ? \(Y=\operatorname{bisim} O(? X \cup\) bisim \() O\) bisim
have Res: \(\wedge P Q x .(P, Q) \in ? X \Longrightarrow(<\nu x>P,<\nu x>Q) \in ? X\) by (blast intro: resChain.step[THEN sym])
from \(\langle x \sharp P\rangle\) have \((<\nu x>(P \| Q), P \|<\nu x>Q) \in ? X\) by (blast intro: resChain.base[THEN sym])
moreover have EqvtX: eqvt ? \(X\) by (fastforce simp add: eqvt-def name-fresh-left
name-rev-per)
ultimately show ?thesis
proof (coinduct rule: bisimTransitiveCoinduct)
case \((c \operatorname{Sim} P Q)\)
\{
fix \(P Q\) lst \(x\)
assume \((x:: n a m e) \sharp(P:: p i)\)
moreover have \(I d \subseteq\) ? \(Y\) by (blast intro: reflexive)
moreover from <eqvt ? \(X\) > bisimEqvt have eqvt ? \(Y\) by blast
moreover have \(\bigwedge P Q x . x \sharp P \Longrightarrow(<\nu x>(P \| Q), P \|<\nu x>Q) \in\) ? \(Y\)
by(blast intro: resChain.base[THEN sym] reflexive)
moreover \{
fix \(P Q x y\)
have \(<\nu x><\nu y>(P \| Q) \sim<\nu y><\nu x>(P \| Q)\) by(rule resComm)
moreover assume \(x \sharp P\)
hence \((<\nu x>(P \| Q), P \|<\nu x>Q) \in ? X\) by (fastforce intro: resChain.base[THEN
sym])
```

        hence (<\nuy><\nux>(P|Q),<\nuy>(P|<\nux>Q)) \in?X by(rule Res)
    ultimately have (<\nux><\nuy>(P|Q),<\nuy>(P|<\nux>Q)) \in?Y by(blast
    intro: reflexive)
}
ultimately have <\nux> (P|Q)\rightsquigarrow[?Y] (P|<\nux>Q) by(rule scopeExtPar-
Left)
moreover note <eqvt ? Y>
moreover from Res have }\LambdaPQx.(P,Q)\in?Y\Longrightarrow(<\nux>P,<\nux>Q)
?Y
by(blast intro: resChain.step[THEN sym] dest: resPres)
ultimately have resChain lst (<\nux>>(P|Q)) \rightsquigarrow[?Y] resChain lst (P|
<\nux>Q)
by(rule resChainI)
}
moreover {
fix P Q lst x
assume (x::name) \# (P::pi)
moreover have Id \subseteq?Y by(blast intro: reflexive)
moreover from <eqvt ?X> bisimEqvt have eqvt ?Y by blast
moreover have \PQ x. x\sharpP\Longrightarrow(P|<\nux>Q,<\nux>(P|Q))\in?Y
by(blast intro: resChain.base[THEN sym] reflexive)
moreover {
fix PQxy
have < Ly>< < x > (P|Q)~<\nux>< \nu |>(P|Q) by(rule resComm)
moreover assume x\sharpP
hence (P|<\nux>Q,<\nux>(P|Q))\in?X by(fastforce intro: resChain.base[THEN
sym])
hence (<\nuy>(P|<\nux>Q),<\nuy><\nux>(P|Q)) \in?X by(rule Res)
ultimately have (<\nuy> (P|<\nux><Q),<\nux><<\nuy>(P|Q))\in?Y by(blast
intro: reflexive)
}
ultimately have (P|<\nux>>)}\rightsquigarrow[?Y]<\nux>(P|Q
by(rule scopeExtParRight)
moreover note <eqvt?Y>
moreover from Res have }\LambdaPQx.(P,Q)\in?Y\Longrightarrow(<\nux>P,<\nux>Q)
?Y
by(blast intro: resChain.step[THEN sym] dest: resPres)
ultimately have resChain lst ( }P|<\nux>Q)\rightsquigarrow[?Y] resChain lst (<\nux>(
|Q))
by(rule resChainI)
}
ultimately show ?case using «(P,Q)\in? X> by auto
next
case(cSym P Q)
thus ?case
by auto (blast dest: symmetric transitive intro: resChain.base[THEN sym]
reflexive)+
qed
qed

```
```

lemma scopeExtPar':
fixes }P:: p
and }Q:: p
and x :: name
assumes xFreshQ:x\sharpQ
shows <\nux>(P|Q)~(<\nux>P)|Q
proof -
have <\nux>(P|Q)~<\nux>>(Q|P)
proof -
have P|Q ~ Q|P by(rule parSym)
thus ?thesis by(rule resPres)
qed
moreover from xFreshQ have <\nux>(Q|P)~Q|(<\nux>P) by(rule scope-
ExtPar)
moreover have Q|<\nux>P~(<\nux>P)| Q by(rule parSym)
ultimately show ?thesis by(blast intro: transitive)
qed
lemma parAssoc:
fixes P :: pi
and }Q::p
and }R::p
shows (P|Q)|R~P|(Q|R)
proof -
let ?X = {(resChain lst ((P|Q)|R), resChain lst (P| (Q|R)) | lst P Q R.
True}
let ?Y = bisim O(?X \cup bisim) O bisim
have ResX: \PQx. (P,Q) \in?X \Longrightarrow (<\nux>P,<\nux>Q ) \in?X
by(blast intro: resChain.step[symmetric])
hence Res Y: \PQx. (P,Q)\in?Y\Longrightarrow(<\nux>P \,<\nux>Q)\in?Y
by(blast intro: resChain.step[symmetric] dest: resPres)
have ((P|Q)|R,P|(Q|R))\in?X by(blast intro: resChain.base[symmetric])
moreover have eqvt ?X by(fastforce simp add: eqvt-def)
ultimately show ?thesis
proof(coinduct rule: bisimTransitiveCoinduct)
case(cSim P Q)
{
fix P Q R lst
have }\bigwedgePQ R.((P|Q)|R,P|(Q|R))\in?Y by(blast intro: reflexiv
resChain.base[symmetric])
moreover have }\PQx.(P,Q)\in?Y\Longrightarrow(<\nux>P,<\nux>Q)\in?Y by(blas
intro: resChain.step[symmetric] resPres)
moreover {

```
fix \(P Q R x\)
have \((<\nu x>((P \| Q) \| R),<\nu x>(P \|(Q \| R))) \in ? X\) by \((\) rule-tac Res \(X)\) (blast intro: resChain.base[symmetric])
moreover assume \(x \sharp P\)
hence \(<\nu x>(P \|(Q \| R)) \sim P \|<\nu x>(Q \| R)\) by (rule scopeExtPar)
ultimately have \((<\nu x>((P \| Q) \| R), P \|<\nu x>(Q \| R)) \in\) ? \(Y\) by (blast intro: reflexive)
\}
moreover \{
fix \(P Q R x\)
have \((<\nu x>((P \| Q) \| R),<\nu x>(P \|(Q \| R))) \in\) ? \(X\) by (rule-tac ResX)
(blast intro: resChain.base[symmetric \(]\) )
moreover assume \(x \sharp R\)
hence \(<\nu x>(P \| Q) \| R \sim<\nu x>((P \| Q) \| R) \mathbf{b y}(\) metis scopeExtPar' symmetric)
ultimately have \((<\nu x>(P \| Q) \| R,<\nu x>(P \|(Q \| R))) \in\) ? \(Y\) by (blast intro: reflexive)
\}
ultimately have \((P \| Q)\|R \rightsquigarrow[? Y] P\|(Q \| R)\) by (rule parAssocLeft)
moreover from 〈eqvt ? \(X\rangle\) bisimEqvt have eqvt ? Y by blast
ultimately have resChain lst \(((P \| Q) \| R) \rightsquigarrow[\) ? Y] resChain lst \((P \|(Q \|\) \(R)\) ) using Res \(Y\)
by (rule resChainI)
\}
with \(\langle(P, Q) \in\) ? \(X\rangle\) show ? case by auto
next
case \((c S y m P Q)\)
\{
fix \(P Q R l s t\)
have \(P\|(Q \| R) \sim(R \| Q)\| P \mathbf{b y}(\) metis parPres parSym transitive)
moreover have \(((R \| Q)\|P, R\|(Q \| P)) \in\) ? \(X\) by (blast intro: resChain.base[symmetric])
moreover have \(R\|(Q \| P) \sim(P \| Q)\| R\) by \((\) metis parPres parSym transitive)
ultimately have \((P\|(Q \| R),(P \| Q)\| R) \in\) ? \(Y\) by blast
hence (resChain lst \((P \|(Q \| R))\), resChain lst \(((P \| Q) \| R)) \in\) ? \(Y\) using Res \(Y\)
by (induct lst) auto
\}
with \(\langle(P, Q) \in\) ? \(X\rangle\) show ? case by blast
qed
qed
lemma scopeFresh:
fixes \(x\) :: name
and \(P:: p i\)
assumes \(x \sharp P\)
shows \(<\nu x>P \sim P\)
```

proof -
have }<\nux>P~<\nux>P|\mathbf{0}\mathbf{by}(\mathrm{ rule parZero[THEN symmetric])
moreover have <\nux>P| 0 ~ 0 | <\nux>P by(rule parSym)
moreover have 0|<\nux>P~<\nux>(0|P) by(rule scopeExtPar[THEN sym-
metric]) auto
moreover have <\nux>(0|P) ~ <\nux>(P| 0) by(rule resPres[OF parSym])
moreover from }\langlex\sharpP>\mathrm{ have }<\nux>(P|\mathbf{0})~P|<\nux>\mathbf{0}\mathrm{ by(rule scopeExtPar)
moreover have }P|<\nux>0 ~ <\nux>00|P by(rule parSym
moreover have <\nux>0 | P ~ 0 | P by(rule parPres[OF nilRes])
moreover have 0 | P~P|\mathbf{0}\mathrm{ by(rule parSym)}
moreover have P|\mathbf{0}~P
ultimately show ?thesis by(metis transitive)
qed
lemma sumRes:
fixes x :: name
and }P::p
and }Q::p
shows <\nux> (P\oplusQ)~(<\nux>P)\oplus(<\nux>Q)
proof -
let ?X = {(<\nux> (P\oplusQ),<\nux>>P\oplus<\nux>Q)| | P Q. True} }
{(<\nux>>P\oplus<\nux>QQ,<\nux>(P\oplusQ))|xPQ.True}
have (<\nux> (P\oplusQ),<\nux>P\oplus<\nux>Q)\in?X by auto
moreover have eqvt ?X by(fastforce simp add: eqvt-def)
ultimately show ?thesis
by(coinduct rule: bisimCoinduct) (fastforce intro: sumResLeft sumResRight
reflexive)+
qed
lemma scopeExtSum:
fixes }P\mathrm{ :: pi
and }Q::p
and x :: name
assumes }x\sharp
shows <\nux> (P\oplusQ)~P\oplus<\nux>
proof -
have <\nux>(P\oplusQ)~<\nux>P
moreover from <x\sharpP> have <\nux>PP\oplus<\nux>Q ~P\oplus<\nux>>Q
by(rule sumPres[OF scopeFresh])
ultimately show ?thesis by(rule transitive)
qed
lemma bangSC:
fixes P :: pi

```
```

    shows !P~P|!P
    proof -
let ?X = {(!P,P|!P),(P|!P,!P)}
have (!P,P|!P)\in?X by simp
thus ?thesis
by(coinduct rule: bisimCoinduct) (auto intro: bangLeftSC bangRightSC reflexive)
qed
lemma resNil:
fixes x :: name
and y :: name
and }P::p
and b :: name
shows <\nux>x<y>.P~0
and <\nux>x{b}.P~\mathbf{0}
proof -
let ? }X={(<\nux>x<y>.P,\mathbf{0}),(\mathbf{0},<\nux>x<y>.P)
have (<\nux>>x<y>.P,0)\in?X by simp
thus <\nux>x<<y>.P~\mathbf{0}
by(coinduct rule: bisimCoinduct) (auto simp add: simulation-def)
next
let ? }X={(<\nux>x{b}.P,\mathbf{0}),(\mathbf{0},<\nux>x{b}.P)
have (<\nux>x{b}.P,0)\in?X by simp
thus <\nux>x{b}.P ~ 0
by(coinduct rule: bisimCoinduct) (auto simp add: simulation-def)
qed
lemma resInput:
fixes x :: name
and a :: name
and y :: name
and }P:: p
assumes }x\not=
and }x\not=
shows <\nux>a<y>.P~a<y>.(<\nux>>P)
proof -
let ?X = {(<\nux>a<y>.P,a<y>.(<\nux>P))| x а y P. x\not=a\wedgex\not=y}\cup
{(a<y>.(<\nux>P),<\nux>a<y>.P)|x a y P.x\not=a\wedgex\not=y}
from assms have (<\nux>a<y>.P,a<y>.(<\nux>P)) \in?X by auto
moreover have eqvt ?X by(fastforce simp add: eqvt-def pt-bij[OF pt-name-inst,
OF at-name-inst])
ultimately show ?thesis
by(coinduct rule: bisimCoinduct) (fastforce intro: resInputLeft reflexive resIn-
putRight)+
qed

```
```

lemma resOutput:
fixes $x$ :: name
and $a::$ name
and $b::$ name
and $\quad P:: p i$
assumes $x \neq a$
and $\quad x \neq b$
shows $<\nu x>a\{b\} . P \sim a\{b\} .(<\nu x>P)$
proof -
let ? $X=\{(<\nu x>a\{b\} . P, a\{b\} .(<\nu x>P)) \mid x a b P . x \neq a \wedge x \neq b\} \cup$
$\{(a\{b\} .(<\nu x>P),<\nu x>a\{b\} . P) \mid x a b P . x \neq a \wedge x \neq b\}$
from assms have $(<\nu x>a\{b\} . P, a\{b\} .(<\nu x>P)) \in ? X$ by blast
moreover have equt ? $X$ by (fastforce simp add: eqvt-def pt-bij[OF pt-name-inst,
OF at-name-inst])
ultimately show ?thesis
by (coinduct rule: bisimCoinduct) (fastforce intro: resOutputLeft resOutputRight
reflexive)+
qed
lemma resTau:
fixes $x$ :: name
and $P:: p i$
shows $<\nu x>\tau .(P) \sim \tau .(<\nu x>P)$
proof -
let ? $X=\{(<\nu x>\tau .(P), \tau .(<\nu x>P)),(\tau .(<\nu x>P),<\nu x>\tau .(P))\}$
have $(<\nu x>\tau .(P), \tau .(<\nu x>P)) \in ? X$ by auto
thus ?thesis
by (coinduct rule: bisimCoinduct) (fastforce intro: resTauLeft resTauRight re-
flexive)+
qed
inductive structCong :: pi $\Rightarrow$ pi $\Rightarrow$ bool $\left(-\equiv_{s}-[70,70] 70\right)$
where
Refl: $P \equiv{ }_{s} P$
| Sym: $P \equiv_{s} Q \Longrightarrow Q \equiv_{s} P$
| Trans: $\llbracket P \equiv_{s} Q ; Q \equiv_{s} R \rrbracket \Longrightarrow P \equiv_{s} R$
| SumComm: $P \oplus Q \equiv{ }_{s} Q \oplus P$
| SumAssoc: $(P \oplus Q) \oplus R \equiv_{s} P \oplus(Q \oplus R)$
|SumId: $P \oplus \mathbf{0} \equiv_{s} P$
| ParComm: $P\left\|Q \equiv_{s} Q\right\| P$
| ParAssoc: $(P \| Q)\left\|R \equiv_{s} P\right\|(Q \| R)$
| ParId: $P \| \mathbf{0} \equiv_{s} P$

```
```

| MatchId: [a\frowna]P\equiv
|ResNil: <\nux>00 \equiv}\mp@subsup{}{s}{}\mathbf{0
ResComm: <\nux><<\nuy>P \equiv
|esSum: < < x> (P\oplusQ) \equiv
ScopeExtPar:x\sharpP\Longrightarrow<\nux>(P|Q) \equiv
InputRes: }\llbracketx\not=a;x\not=y\rrbracket\Longrightarrow<\nux>a<y>.P \equiv\mp@subsup{\equiv}{s}{}a<y>.(<\nux>P
OutputRes: }\llbracketx\not=a;x\not=b\rrbracket\Longrightarrow<\nux>a{b}.P\mp@subsup{\equiv}{s}{}a{b}.(<\nux>P
| TauRes:<\nux>\tau.(P) \equiv
| BangUnfold: !P \equiv}\mp@subsup{}{s}{}P||
lemma structCongBisim:
fixes }P::p
and }Q:: p
assumes P}\equiv\mp@subsup{}{s}{}
shows P~Q
using assms
by(induct rule: structCong.induct)
(auto intro: reflexive symmetric transitive sumSym sumAssoc sumZero parSym
parAssoc parZero
nilRes resComm resInput resOutput resTau sumRes scopeExtPar bangSC
matchId mismatchId)
end
theory Strong-Late-Bisim-Subst-SC
imports Strong-Late-Bisim-Subst-Pres Strong-Late-Bisim-SC
begin
lemma matchId:
fixes a :: name
and }P::p
shows [a\frowna]P ~
by(auto simp add: substClosed-def intro: Strong-Late-Bisim-SC.matchId)
lemma mismatchNil:
fixes a :: name
and P:: pi
shows [a\not=a]P ~}\mp@subsup{~}{}{s}\mathbf{0
by(auto simp add: substClosed-def intro: Strong-Late-Bisim-SC.mismatchNil)
lemma scopeFresh:
fixes P :: pi
and x :: name

```
```

    assumes xFreshP: }x\sharp
    shows <\nux>P ~
    proof(auto simp add: substClosed-def)
fix s :: (name }\times\mathrm{ name) list
have \existsc::name.c\sharp(P,s) by(blast intro: name-exists-fresh)
then obtain c::name where cFreshP:c\sharpP and cFreshs: c\sharps by(force simp
add: fresh-prod)
have <\nux>P=<\nuc>P
proof -
from cFreshP have <\nux>P = <\nuc> ([(x,c)] \cdot P) by(simp add: alphaRes)
with cFreshP xFreshP show ?thesis by(simp add: name-fresh-fresh)
qed
with cFreshP cFreshs show (<\nux>P)[<s>] ~ P[<s>]
by(force intro:Strong-Late-Bisim-SC.scopeFresh)
qed
lemma resComm:
fixes P :: pi
and x :: name
and y :: name
shows <\nux><\nuyy>P ~
proof(cases x=y)
assume xeqy: }x=
have P ~}\mp@subsup{~}{}{s}P\mathrm{ by(rule Strong-Late-Bisim-Subst.reflexive)
hence }<\nux>P\mp@subsup{~}{}{s}<\nux>P\mathrm{ by(rule resPres)
hence <\nux><<\nux>P ~}\mp@subsup{~}{}{s}<\nux><\nux>P by(rule resPres
with xeqy show ?thesis by simp
next
assume xineqy: }x\not=
show ?thesis
proof(auto simp add: substClosed-def)
fix s::(name > name) list
have \existsc::name. c \sharp (P,s,y) by(blast intro: name-exists-fresh)
then obtain c::name where cFreshP:c\sharpP and cFreshs: c\sharps and cineqy: c
\not= y
by(force simp add: fresh-prod)
have \existsd::name. d \sharp (P,s,c,x,y) by (blast intro: name-exists-fresh)
then obtain d::name where dFreshP:d\sharpP and dFreshs: d\sharps and dineqc:
d\not=c
and dineqx: }d\not=x\mathrm{ and dineqy: }d\not=
by(force simp add: fresh-prod)

```
```

    have < < x>><\nuy>P=<\nuc><\nud>>([(x,c)]\cdot[(y,d)]\cdotP)
    proof -
    from cineqy cFreshP have cFreshyP:c\sharp<\nuy>P by(simp add: name-fresh-abs)
    from dFreshP have <\nuy>P=<\nud>([(y,d)]\cdotP) by(rule alphaRes)
    moreover from cFreshyP have <\nux><<\nuy>P=<\nuc>([(x,c)]\cdot(<\nuy>P))
    by(rule alphaRes)
ultimately show ?thesis using dineqc dineqx by(simp add: name-calc)
qed
moreover have <\nuy><\nux>P = <\nud><\nuc>([(x,c)]\cdot[(y,d)]\cdotP)
proof -
from dineqx dFreshP have dFreshxP: d\sharp<\nux>P by(simp add: name-fresh-abs)
from cFreshP have <\nux>P = <\nuc> ([(x,c)]\cdotP) by(rule alphaRes)
moreover from dFreshxP have <\nuy><\nux>PP=<\nud> ([(y,d)] \cdot(<\nux>P))
by(rule alphaRes)
ultimately have <\nuy><\nux>>P=<\nud><\nuc>([(y,d)]\cdot[(x,c)]\cdotP) using
dineqc cineqy
by(simp add: name-calc)
thus ?thesis using dineqx dineqc cineqy xineqy
by(subst name-perm-compose, simp add: name-calc)
qed
ultimately show (<\nux><<\nuy>P)[<s>] ~ (<\nuy>< <\nux>P)[<s>] using cFreshs
dFreshs
by(force intro:Strong-Late-Bisim-SC.resComm)
qed
qed
lemma sumZero:
fixes P :: pi
shows }P\oplus\mathbf{0}\mp@subsup{~}{}{s}
by(force simp add: substClosed-def intro: Strong-Late-Bisim-SC.sumZero)
lemma sumSym:
fixes P :: pi
and }Q:: p
shows }P\oplusQ\mp@subsup{~}{}{s}Q\oplus
by(force simp add: substClosed-def intro: Strong-Late-Bisim-SC.sumSym)
lemma sumAssoc:
fixes P :: pi
and }Q:: p
and }R::p
shows (P\oplusQ)\oplusR ~
by(force simp add: substClosed-def intro: Strong-Late-Bisim-SC.sumAssoc)

```
```

lemma sumRes:
fixes P :: pi
and }Q::p
and x :: name
shows <\nux> (P\oplusQ) \mp@subsup{~}{}{s}<\nux>P\oplus<\nux>Q
proof(auto simp add: substClosed-def)
fix s :: (name }\times\mathrm{ name) list
have \existsc::name.c }\#(P,Q,s) by(blast intro: name-exists-fresh
then obtain c::name where cFreshP:c\sharpP and cFreshQ:c\sharpQ and cFreshs:
c\sharps
by(force simp add: fresh-prod)
have <\nux>>(P\oplusQ)=<\nuc>(([(x,c)]\cdotP)\oplus([(x,c)]\cdotQ))
proof -
from cFreshP cFreshQ have c\sharpP\oplus Q by simp
hence }<\nux>(P\oplusQ)=\langle\nuc>([(x,c)]\cdot(P\oplusQ))\mathrm{ by(simp add: alphaRes)
thus?thesis by(simp add: name-fresh-fresh)
qed
moreover from cFreshP have <\nux>>P=<\nuc>([(x,c)] - P) by(simp add:
alphaRes)
moreover from cFreshQ have <\nux>>Q = <\nuc>>([(x,c)] - Q) by(simp add:
alphaRes)
ultimately show (<\nux>>(P\oplusQ))[<s>] ~ (<\nux>P)[<s>]\oplus(<\nux>QQ)[<s>]
using cFreshs
by(force intro:Strong-Late-Bisim-SC.sumRes)
qed
lemma scopeExtSum:
fixes }P\mathrm{ :: pi
and }Q:: p
and x :: name
assumes xFreshP: x\sharpP
shows <\nux>}>(P\oplusQ)\mp@subsup{~}{}{s}P\oplus<\nux>
proof(auto simp add: substClosed-def)
fix }s::(\mathrm{ name }\times\mathrm{ name) list
have \existsc::name.c }\#(P,Q,s) by(blast intro: name-exists-fresh
then obtain c::name where cFreshP:c\sharpP and cFreshQ:c\sharpQ and cFreshs:
c\sharps
by(force simp add: fresh-prod)
have < < x > (P\oplusQ) = <\nuc>(P\oplus([(x,c)]\cdotQ))
proof -

```
```

    from cFreshP cFreshQ have c \sharpPP
    hence }<\nux\rangle(P\oplusQ)=\langle\nuc>([(x,c)]\cdot(P\oplusQ))\mathrm{ by(simp add: alphaRes)
    with xFreshP cFreshP show ?thesis by(simp add: name-fresh-fresh)
    qed
    moreover from cFreshQ have <\nux>>Q = <\nuc> ([(x,c)] - Q) by(simp add:
    alphaRes)
ultimately show (<\nux> (P\oplusQ))[<s>] ~ P[<s>]\oplus(<\nux><Q)[<s>] using
cFreshs cFreshP
by(force intro:Strong-Late-Bisim-SC.scopeExtSum)
qed
lemma parZero:
fixes P :: pi
shows P|0 0 ~ s}
by(force simp add: substClosed-def intro: Strong-Late-Bisim-SC.parZero)
lemma parSym:
fixes }P\mathrm{ :: pi
and }Q:: p
shows }P|Q\mp@subsup{~}{}{s}Q|
by(force simp add: substClosed-def intro: Strong-Late-Bisim-SC.parSym)
lemma parAssoc:
fixes P :: pi
and }Q::p
and }R::p
shows (P|Q)|R ~
by(force simp add: substClosed-def intro: Strong-Late-Bisim-SC.parAssoc)
lemma scopeExtPar:
fixes }P:: p
and }Q:: p
and x :: name
assumes xFreshP: }x\sharp
shows <\nux>(P|Q) \mp@subsup{~}{}{s}P|<\nux>Q
proof(auto simp add: substClosed-def)
fix s :: (name }\times\mathrm{ name) list
have \existsc::name. c \# (P,Q,s) by(blast intro: name-exists-fresh)
then obtain c::name where cFreshP:c\sharpP and cFreshQ:c\sharpQ and cFreshs:
c\#s
by(force simp add: fresh-prod)

```
```

    have <\nux>(P|Q)=<\nuc>(P|([(x,c)]\cdotQ))
    proof -
    from cFreshP cFreshQ have c\sharpP|Q by simp
    hence <\nux> (P|Q)=<\nuc>([(x,c)] • (P|Q)) by(simp add: alphaRes)
    with xFreshP cFreshP show ?thesis by(simp add: name-fresh-fresh)
    qed
moreover from cFreshQ have <\nux>>Q = <\nuc> ([(x,c)] - Q) by(simp add:
alphaRes)
ultimately show (<\nux>(P|Q))[<s>] ~ P[<s>]| | (<\nux>>Q)[<s>] using
cFreshs cFreshP
by(force intro:Strong-Late-Bisim-SC.scopeExtPar)
qed
lemma scopeExtPar':
fixes P :: pi
and }Q::p
and x :: name
assumes xFreshP: x\sharpQ
shows <\nux> (P|Q) ~s}(<\nux>P)|
proof -
have <\nux>(P|Q) \mp@subsup{~}{}{s}<\nux>(Q|P) by(blast intro: parSym resPres)
moreover from xFreshP have <\nux>(Q|P) ~s Q |<\nux>P by(rule scope-
ExtPar)
moreover have Q|<\nux>P ~s}(<\nux>P)|Q by(rule parSym
ultimately show ?thesis by (blast intro: transitive)
qed
lemma bangSC:
fixes P :: pi
shows !P ~}\mp@subsup{~}{}{s}P|!
by(auto simp add: substClosed-def intro: Strong-Late-Bisim-SC.bangSC)
lemma nilRes:
fixes x :: name
shows <\nux>0 0 ~
proof(auto simp add: substClosed-def)
fix \sigma::(name > name) list
obtain y::name where y\sharp\sigma
by(generate-fresh name) auto
have <\nuy>0 ~ 0 by (rule Strong-Late-Bisim-SC.nilRes)
with }\langley\sharp\sigma\rangle\mathrm{ have (< < y>00)[< < >] ~ 0 by simp
thus (<\nux>0)[<\sigma>]~0

```
```

    by(subst alphaRes[where c=y]) auto
    qed
lemma resTau:
fixes x :: name
and P :: pi
shows <\nux>(\tau.(P)) ~}\mp@subsup{~}{}{s}\tau.(<\nux>P
proof(auto simp add: substClosed-def)
fix }\sigma::(name \times name) lis
obtain y::name where }y\sharpP\mathrm{ and }y\sharp
by(generate-fresh name, auto)
have <\nuy>(\tau.(([(x,y)]\cdotP)[<\sigma>]))~\tau.(<\nuy>(([(x,y)]\cdotP)[<\sigma>]))
by(rule resTau)
with }\langley\sharp\sigma\rangle\mathrm{ have }(<\nuy>(\tau.([(x,y)]\cdotP)))[<\sigma>]~(\tau.(<\nuy>([(x,y)]\cdotP)))[<\sigma>
by simp
with }\langley\sharpP>\mathrm{ show (< < x> . (P))[<六] }~\tau.((<\nux>P)[<\sigma>]
apply(subst alphaRes[where c=y])
apply simp
apply(subst alphaRes[where }c=y\mathrm{ and }a=x]
by simp+
qed
lemma resOutput:
fixes x :: name
and a :: name
and b:: name
and }P::p
assumes }x\not=
and }x\not=
shows <\nux> (a{b}.(P)) ~}\mp@subsup{~}{}{s}a{b}.(<\nux>P
proof(auto simp add: substClosed-def)
fix }\sigma::(name \times name) lis
obtain y::name where }y\sharpP\mathrm{ and }y\sharp\sigma\mathrm{ and }y\not=a\mathrm{ and }y\not=
by(generate-fresh name, auto)
have <\nuy>((seq-subst-name a \sigma){seq-subst-name b \sigma}.(([(x,y)] •P)[<\sigma>])) ~
seq-subst-name a \sigma{seq-subst-name b \sigma}.(<\nuy> (([(x,y)] \cdot P)[<\sigma>]))
using〈y\not=a\rangle\langley\not=b\rangle\langley\sharp\sigma\rangle freshSeqSubstName
by(rule-tac resOutput) auto

```

```

P)))[<\sigma>]
by simp
with }\langley\sharpP\rangle\langley\not=a\rangle\langley\not=b\rangle\langlex\not=a\rangle\langlex\not=b\rangle\mathrm{ show (<生>a{b}.(P))[< < > ] ~
seq-subst-name a \sigma{seq-subst-name b \sigma}.((<\nux>>P)[<\sigma>])
apply(subst alphaRes[where c=y])
apply simp
apply(subst alphaRes[where c=y and a=x])

```
```

    by simp+
    qed
lemma resInput:
fixes x :: name
and a :: name
and }b:: nam
and }P::p
assumes }x\not=
and }x\not=
shows <\nux>(a<y>.(P)) ~}\mp@subsup{~}{}{s}a<y>.(<\nux>P
proof(auto simp add: substClosed-def)
fix }\sigma::(name \times name) lis
obtain \mp@subsup{x}{}{\prime}::name where }\mp@subsup{x}{}{\prime}\sharpP\mathrm{ and }\mp@subsup{x}{}{\prime}\sharp\sigma\mathrm{ and }\mp@subsup{x}{}{\prime}\not=a\mathrm{ and }\mp@subsup{x}{}{\prime}\not=x\mathrm{ and }\mp@subsup{x}{}{\prime}\not=
by(generate-fresh name, auto)
obtain }\mp@subsup{y}{}{\prime}::name where \mp@subsup{y}{}{\prime}\sharpP\mathrm{ and }\mp@subsup{y}{}{\prime}\sharp\sigma\mathrm{ and }\mp@subsup{y}{}{\prime}\not=a\mathrm{ and }\mp@subsup{y}{}{\prime}\not=x\mathrm{ and }\mp@subsup{y}{}{\prime}\not=
and }\mp@subsup{x}{}{\prime}\not=\mp@subsup{y}{}{\prime
by(generate-fresh name, auto)
have <\nu\mp@subsup{x}{}{\prime}>((seq-subst-name a \sigma)<\mp@subsup{y}{}{\prime}>.(([(y,\mp@subsup{y}{}{\prime})] \cdot [(x, x')] • P)[<\sigma>]))~
seq-subst-name a }\sigma<\mp@subsup{y}{}{\prime}>.(<\nu\mp@subsup{x}{}{\prime}>(([(y,\mp@subsup{y}{}{\prime})]\cdot[(x,\mp@subsup{x}{}{\prime})]\cdotP)[<\sigma>])

```

```

    by(rule-tac resInput) auto
    ```

```

(a<\mp@subsup{y}{}{\prime}>.(<\nu\mp@subsup{x}{}{\prime}>([(y,\mp@subsup{y}{}{\prime})]\cdot[(x,\mp@subsup{x}{}{\prime})]\cdotP)))[<\sigma>]
by simp
with \langle\mp@subsup{x}{}{\prime}\sharpP\rangle\langle\mp@subsup{y}{}{\prime}\not=x\rangle\langle\mp@subsup{x}{}{\prime}\not=y\rangle\langle\mp@subsup{y}{}{\prime}\sharpP\rangle\langle\mp@subsup{x}{}{\prime}\not=\mp@subsup{y}{}{\prime}\rangle\langle\mp@subsup{x}{}{\prime}\not=a\rangle\langle\mp@subsup{y}{}{\prime}\not=a\rangle\langlex\not=a\rangle\langlex
\not=y> show (<\nux>a<y>.(P))[<\sigma>]~a<y>.(<\nux>P)[<\sigma>]
apply(subst alphaInput[where c=y`])
apply simp
apply(subst alphaRes[where c=x ])
apply(simp add: abs-fresh fresh-left calc-atm)
apply(simp add: eqvts calc-atm)
apply(subst alphaRes[where c=\mp@subsup{x}{}{\prime}}\mathrm{ and }a=x]
apply simp
apply(subst alphaInput[where c=\mp@subsup{y}{}{\prime}}\mathrm{ and }x=y]
apply(simp add: abs-fresh fresh-left calc-atm)
apply(simp add: eqvts calc-atm)
apply(subst perm-compose)
by(simp add: eqvts calc-atm)
qed
lemma bisimSubstStructCong:
fixes P :: pi
and }Q::p
assumes P \equiv}\mp@subsup{}{s}{}
shows P ~ ~

```
```

using assms
apply(induct rule: structCong.induct)
by(auto intro: reflexive symmetric transitive sumSym sumAssoc sumZero parSym
parAssoc parZero
nilRes resComm resInput resOutput resTau sumRes scopeExtPar bangSC
matchId mismatchId)
end
theory Weak-Late-Cong-Subst-SC
imports Weak-Late-Cong-Subst Strong-Late-Bisim-Subst-SC
begin
lemma resComm:
fixes P :: pi
shows <\nua><\nub>P \simeqs <\nub><\nua>P
proof -
have <\nua><\nub>P ~
by(rule Strong-Late-Bisim-Subst-SC.resComm)
thus ?thesis by(rule strongEqWeakCong)
qed
lemma matchId:
fixes a :: name
and }P::p
shows [a\frowna]P}\mp@subsup{\simeq}{}{s}
proof -
have [a\frowna]P ~s P by(rule Strong-Late-Bisim-Subst-SC.matchId)
thus ?thesis by(rule strongEqWeakCong)
qed
lemma matchNil:
fixes a :: name
and }P::p
shows [a\not=a]P \simeqs 0
proof -

```
```

    have [a\not=a]P ~}\mp@subsup{~}{}{s}\mathbf{0}\mathrm{ by(rule Strong-Late-Bisim-Subst-SC.mismatchNil)
    thus ?thesis by(rule strongEqWeakCong)
    qed

```
```

lemma sumSym:
fixes $P:: p i$
and $\quad Q:: p i$
shows $P \oplus Q \simeq^{s} Q \oplus P$
proof -
have $P \oplus Q \sim^{s} Q \oplus P \mathbf{b y}($ rule Strong-Late-Bisim-Subst-SC.sumSym)
thus ?thesis by (rule strongEqWeakCong)
qed
lemma sumAssoc:
fixes $P$ :: $p i$
and $\quad Q:: p i$
and $\quad R:: p i$
shows $(P \oplus Q) \oplus R \simeq^{s} P \oplus(Q \oplus R)$
proof -
have $(P \oplus Q) \oplus R \sim^{s} P \oplus(Q \oplus R)$ by(rule Strong-Late-Bisim-Subst-SC.sumAssoc)
thus ?thesis by (rule strongEqWeakCong)
qed
lemma sumZero:
fixes $P$ :: $p i$
shows $P \oplus \mathbf{0} \simeq^{s} P$
proof -
have $P \oplus \mathbf{0} \sim^{s} P$ by(rule Strong-Late-Bisim-Subst-SC.sumZero)
thus ?thesis by (rule strongEqWeakCong)
qed
lemma parZero:
fixes $P$ :: $p i$
shows $P \| \mathbf{0} \simeq^{s} P$
proof -
have $P \| \mathbf{0} \sim^{s} P$ by(rule Strong-Late-Bisim-Subst-SC.parZero)
thus ?thesis by (rule strongEqWeakCong)
qed
lemma parSym:
fixes $P:: p i$

```
```

    and }Q:: p
    shows }P|Q\mp@subsup{\simeq}{}{s}Q|
    proof -
have P|Q ~ s Q |P by(rule Strong-Late-Bisim-Subst-SC.parSym)
thus ?thesis by(rule strongEqWeakCong)
qed
lemma scopeExtPar:
fixes }P\mathrm{ :: pi
and }Q:: p
and x :: name
assumes }x\sharp
shows <\nux>(P|Q) \simeqs}P|<\nux>
proof -
from assms have <\nux>(P|Q) ~
thus ?thesis by(rule strongEqWeakCong)
qed
lemma scopeExtPar':
fixes P :: pi
and }Q::p
and x :: name
assumes xFreshQ: x\sharpQ
shows <\nux>(P|Q) \simeqs}(<\nux>P)|
proof -
from assms have <\nux>>(P|Q) ~s}(<\nux>P)|Q by(rule Strong-Late-Bisim-Subst-SC.scopeExtPar'
thus ?thesis by(rule strongEqWeakCong)
qed
lemma parAssoc:
fixes P :: pi
and }\quadQ::p
and }R:: p
shows (P|Q)|R\mp@subsup{\simeq}{}{s}P|(Q|R)
proof -
have (P|Q)|R ~s}P|(Q|R)\mathbf{by}(rule Strong-Late-Bisim-Subst-SC.parAssoc
thus ?thesis by(rule strongEqWeakCong)
qed
lemma scopeFresh:
fixes P :: pi
and a :: name

```
```

    assumes aFreshP: a\sharpP
    shows <\nua>P 工s}
    proof -
from assms have <\nua>P ~s P by(rule Strong-Late-Bisim-Subst-SC.scopeFresh)
thus ?thesis by(rule strongEqWeakCong)
qed
lemma scopeExtSum:
fixes }P\mathrm{ :: pi
and }Q:: p
and x :: name
assumes }x\sharp
shows <\nux>}(P\oplusQ)\mp@subsup{\simeq}{}{s}P\oplus<\nux>
proof -
from assms have <\nux>(P\oplusQ) ~s}P\oplus<\nux>Q by(rule Strong-Late-Bisim-Subst-SC.scopeExtSum)
thus ?thesis by(rule strongEqWeakCong)
qed
lemma bangSC:
fixes }
shows !P \simeqs }P||
proof -
have !P ~ s}P|!P\mathrm{ by(rule Strong-Late-Bisim-Subst-SC.bangSC)
thus ?thesis by(rule strongEqWeakCong)
qed
end
theory Weak-Late-Step-Sim-Pres
imports Weak-Late-Step-Sim
begin
lemma tauPres:
fixes P :: pi
and Q :: pi
and Rel :: (pi\times pi) set
and Rel':: (pi }\times pi) se
assumes PRelQ: (P,Q)\inRel
shows }\tau.(P)\rightsquigarrow<\mathrm{ Rel> }\tau.(Q
proof(induct rule: simCases)
case(Bound Q' a y)
have }\tau.(Q)\longmapstoa<\nuy>\prec\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
hence False by auto

```
```

    thus ?case by simp
    next
case(Input Q' a x)
have }\tau.(Q)\longmapstoa<x>\prec\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
hence False by auto
thus ?case by simp
next
case(Free Q' \alpha)
have}\tau.(Q)\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
thus ?case using PRelQ
proof(induct rule: tauCases, auto simp add: pi.inject residual.inject)
have }\tau.(P)\Longrightarrow\mp@subsup{\Longrightarrow}{l}{}\tau\precP\mathrm{ by(rule Weak-Late-Step-Semantics.Tau)
moreover assume (P, Q')\in Rel
ultimately show }\exists\mp@subsup{P}{}{\prime}.\tau.(P)\Longrightarrow\mp@subsup{\Longrightarrow}{l}{}\tau\prec\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mathrm{ Rel by blast
qed
qed
lemma inputPres:
fixes P :: pi
and }Q :: p
and a :: name
and x :: name
and Rel :: (pi\times pi) set
assumes PRelQ: }\forally.(P[x::=y],Q[x::=y])\in\operatorname{Rel
and Eqvt: eqvt Rel
shows }a<x>.P\rightsquigarrow<\mathrm{ Rel }>a<x>.
proof -
show ?thesis using Eqvt
proof(induct rule: simCasesCont[of - (P, a, x,Q)])
case(Bound Q'b y)
have }a<x>.Q\longmapstob<\nuy>\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
hence False by auto
thus ?case by simp
next
case(Input Q' b y)
have }y\sharp(P,a,x,Q)\mathrm{ by fact
hence yFreshP:(y::name) \sharpP and yineqx: y\not=x and y\not=a and y\sharpQ
by(simp add: fresh-prod)+
have }a<x>.Q\longmapstob<y>\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
thus ?case using \langley \not=a\rangle\langley\not= x\rangle\langley\sharpQ\rangle
proof(induct rule: inputCases, auto simp add: subject.inject)
have }\forallu.\exists\mp@subsup{P}{}{\prime}.a<x>.P\Longrightarrow\mp@subsup{}{l}{}u\mathrm{ in }([(x,y)]\cdotP)->a<y>\prec\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},([(x,y)

- Q) }[y::=u])\in\operatorname{Rel
proof(rule allI)
fix }
have }a<x>.P\Longrightarrow\mp@subsup{}{l}{}u\mathrm{ in ([(x,y)] P P) }->a<y>< ([(x,y)] \cdot P)[y::=u] (i
?goal)

```
```

    proof -
    from yFreshP have }a<x>.P=a<y>.([(x,y)]\cdotP)\mathbf{by}(rule Agent.alphaInput
    moreover have }a<y>.([(x,y)]\cdotP)\Longrightarrow\Longrightarrowlu in ([(x,y)]\cdotP)->a<y>\prec ([(x
    y)] • P)[y::=u]
by(rule Weak-Late-Step-Semantics.Input)
ultimately show ?goal by(simp add: name-swap)
qed
moreover have }(([(x,y)]\cdotP)[y::=u],([(x,y)]\cdotQ)[y::=u])\in\mathrm{ Rel
proof -
from PRelQ have (P[x::=u], Q[x::=u]) \in Rel by auto
with }\langley\sharpP\rangle\langley\sharpQ\rangle\mathrm{ show ?thesis by(simp add: renaming)
qed
ultimately show }\exists\mp@subsup{P}{}{\prime}.a<x>.P\mp@subsup{\Longrightarrow}{l}{
([(x,y)] • Q)[y::=u]) \inRel
by blast
qed
thus \exists\mp@subsup{P}{}{\prime\prime}.\forallu.\exists\mp@subsup{P}{}{\prime}.a<x>.P\Longrightarrow\mp@subsup{\Longrightarrow}{l}{}u\mathrm{ in }\mp@subsup{P}{}{\prime\prime}->a<y>< < P'^( (P',([(x,y)] \cdot
Q)[y::=u]) \in Rel by blast
qed
next
case(Free Q' \alpha)
have }a<x>.Q\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
hence False by auto
thus ?case by simp
qed
qed
lemma outputPres:
fixes P :: pi
and }Q\quad::p
and a :: name
and b :: name
and Rel :: (pi\times pi) set
and Rel'::(pi\timespi) set
assumes PRelQ: (P,Q)\inRel
shows a{b}.P }\rightsquigarrow<\mathrm{ Rel> a{b}.Q
proof(induct rule: simCases)
case(Bound Q' c x)
have }a{b}.Q\longmapstoc<\nux>\prec\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
hence False by auto
thus ?case by simp
next
case(Input Q' c x)
have }a{b}.Q\longmapstoc<x>\prec\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact

```
```

    hence False by auto
    thus ?case by simp
    next
case(Free \mp@subsup{Q}{}{\prime}\alpha)
have }a{b}.Q\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
thus ?case using PRelQ
proof(induct rule: outputCases, auto simp add: pi.inject residual.inject)
have }a{b}.P\Longrightarrow\mp@subsup{\Longrightarrow}{l}{}a[b]\precP\mathbf{by}(rule Weak-Late-Step-Semantics.Output)
moreover assume (P, Q')\in Rel
ultimately show }\exists\mp@subsup{P}{}{\prime}.a{b}.P\Longrightarrow\mp@subsup{\Longrightarrow}{l}{}a[b]\prec\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mathrm{ Rel by blast
qed
qed
lemma matchPres:
fixes P :: pi
and }Q :: p
and a :: name
and b :: name
and Rel :: (pi\times pi) set
and Rel':: (pi \times pi) set
assumes PSimQ: P\rightsquigarrow<Rel> Q
and RelRel':Rel \subseteqRel'
shows [a\frownb]P\rightsquigarrow<Rel'> [a\frownb]Q
proof(induct rule: simCases)
case(Bound Q' c x)
have }x\sharp[a\frownb]P\mathrm{ by fact
hence xFreshP:(x::name) \#P by simp
have [a\frownb]Q\longmapstoc<\nux>}\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
thus ?case
proof(induct rule: matchCases)
case cMatch
have }Q\longmapstoc<\nux>\prec\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
with PSimQ xFreshP obtain }\mp@subsup{P}{}{\prime}\mathrm{ where PTrans: P }\mp@subsup{\Longrightarrow}{l}{l}c<\nux>\prec\prec\mp@subsup{P}{}{\prime
and P'RelQ':( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mathrm{ Rel
by(blast dest: simE)
from PTrans have [a\frowna]P\Longrightarrow>
moreover from P'RelQ' RelRel' have ( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mp@subsup{\mathrm{ Rel' }}{}{\prime}\mathrm{ by blast
ultimately show ?case by blast
qed
next
case(Input Q' c x)
have }x\sharp[a\frownb]P\mathrm{ by fact
hence xFreshP:(x::name) \sharpP by simp
have [a\frownb]Q\longmapstoc<x>}\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
thus ?case
proof(induct rule: matchCases)
case cMatch

```
```

    have }Q\longmapstoc<x>\prec\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    ```
    with \(P \operatorname{Sim} Q x F r e s h P\) obtain \(P^{\prime \prime}\) where \(L 1: \forall u . \exists P^{\prime} . P \Longrightarrow_{l} u\) in \(P^{\prime \prime} \rightarrow c<x>\)
\(\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}[x::=u]\right) \in\) Rel
            by (blast dest: simE)
    have \(\forall u . \exists P^{\prime} .[a \frown a] P \Longrightarrow_{l} u\) in \(P^{\prime \prime} \rightarrow c<x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}[x::=u]\right) \in \operatorname{Rel}^{\prime}\)
    proof(rule allI)
        fix \(u\)
    from L1 obtain \(P^{\prime}\) where PTrans: \(P \Longrightarrow_{l} u\) in \(P^{\prime \prime} \rightarrow c<x>\prec P^{\prime}\) and \(P^{\prime} R e l Q^{\prime}\) :
\(\left(P^{\prime}, Q^{\prime}[x::=u]\right) \in \operatorname{Rel}\)
            by blast
    from PTrans have \([a \frown a] P \Longrightarrow_{l}\) u in \(P^{\prime \prime} \rightarrow c<x>\prec P^{\prime}\) by (rule Weak-Late-Step-Semantics.Match)
            with \(P^{\prime}\) RelQ \(Q^{\prime}\) RelRel' show \(\exists P^{\prime} .[a \frown a] P \Longrightarrow_{l} u\) in \(P^{\prime \prime} \rightarrow c<x>\prec P^{\prime} \wedge\left(P^{\prime}\right.\),
\(\left.Q^{\prime}[x::=u]\right) \in \operatorname{Rel}^{\prime}\)
            by blast
    qed
    thus ?case by blast
    qed
next
    case \(\left(\right.\) Free \(\left.Q^{\prime} \alpha\right)\)
    have \([a \frown b] Q \longmapsto \alpha \prec Q^{\prime}\) by fact
    thus ? case
    proof (induct rule: matchCases)
        case \(c\) Match
    have \(Q \longmapsto \alpha \prec Q^{\prime}\) by fact
    with PSimQ obtain \(P^{\prime}\) where PTrans: \(P \Longrightarrow{ }_{l} \alpha \prec P^{\prime}\) and PRel: \(\left(P^{\prime}, Q^{\prime}\right) \in\)
Rel
        by (blast dest: \(\operatorname{sim} E)\)
    from PTrans have \([a \frown a] P \Longrightarrow_{l} \alpha \prec P^{\prime}\) by (rule Weak-Late-Step-Semantics.Match)
        with PRel RelRel' show ?case by blast
    qed
qed
lemma mismatchPres:
    fixes \(P\) :: \(p i\)
    and \(Q \quad:: p i\)
    and \(a\) :: name
    and \(b\) :: name
    and Rel :: \((p i \times p i)\) set
    and Rel' \(::(p i \times p i)\) set
    assumes \(P \operatorname{SimQ} Q: P \rightsquigarrow \operatorname{Rel}>Q\)
    and RelRel': Rel \(\subseteq\) Rel \(^{\prime}\)
    shows \([a \neq b] P \rightsquigarrow<\operatorname{Rel}^{\prime}>[a \neq b] Q\)
proof (cases \(a=b\) )
    assume \(a=b\)
    thus ?thesis
        by (auto simp add: weakStepSimDef)
next
```

assume aineqb: }a\not=
show ?thesis
proof(induct rule: simCases)
case(Bound Q' c x)
have }x\sharp[a\not=b]P\mathrm{ by fact
hence xFreshP:(x::name) \sharpP by simp
have [a\not=b]Q\longmapstoc<\nux>}\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
thus ?case
proof(induct rule: mismatchCases)
case cMismatch
have }Q\longmapstoc<\nux>\prec\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
with PSimQ xFreshP obtain P' where PTrans: P \Longrightarrow\Longrightarrowllc<\nux>}\prec\mp@subsup{P}{}{\prime
and P'RelQ':}(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\operatorname{Rel
by(blast dest: simE)
from PTrans aineqb have [a\not=b]P\Longrightarrow\mp@subsup{}{l}{}c<\nux>}\prec\mp@subsup{\}{}{\prime}\mathbf{by}(rule Weak-Late-Step-Semantics.Mismatch)
moreover from P'RelQ' RelRel' have ( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mp@subsup{R}{}{\prime}\mp@subsup{R}{}{\prime}\mp@subsup{l}{}{\prime}\mathrm{ by blast
ultimately show ?case by blast
qed
next
case(Input Q' c x)
have }x\sharp[a\not=b]P\mathrm{ by fact
hence xFreshP:(x::name) \#P by simp
have [a\not=b]Q\longmapstoc<x>\prec\prec Q' by fact
thus ?case
proof(induct rule: mismatchCases)
case cMismatch
have }Q\longmapstoc<x>\prec\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact

```

```

\prec P'^( (P', Q'[x::=u]) \in Rel
by(blast dest: simE)

```

```

    proof(rule allI)
            fix u
                            from L1 obtain P' where PTrans: P \Longrightarrow>lu in P}\mp@subsup{P}{}{\prime\prime}->c<x>\prec\mp@subsup{P}{}{\prime}\mathrm{ and
    P'RelQ':( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime}[x::=u])\in\operatorname{Rel
by blast
from PTrans aineqb have [a\not=b]P\Longrightarrow\mp@subsup{}{l}{}u in P}\mp@subsup{P}{}{\prime\prime}->c<x>\prec\mp@subsup{P}{}{\prime}\mathrm{ by(rule
Weak-Late-Step-Semantics.Mismatch)
with P'RelQ' RelRel' show }\exists\mp@subsup{P}{}{\prime}.[a\not=b]P\mp@subsup{\Longrightarrow}{l}{}u\mathrm{ in }\mp@subsup{P}{}{\prime\prime}->c<x>\prec\prec\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime}
Q}[x::=u])\inRe\mp@subsup{el}{}{\prime
by blast
qed
thus ?case by blast
qed
next
case(Free Q' \alpha)
have [a\not=b]Q\longmapsto\alpha\prec Q' by fact
thus ?case
proof(induct rule: mismatchCases)

```
```

    case cMismatch
    have }Q\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    with PSimQ obtain }\mp@subsup{P}{}{\prime}\mathrm{ where PTrans: }P>\mp@subsup{\}{l}{}\alpha\prec\mp@subsup{P}{}{\prime}\mathrm{ and PRel: ( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime}
    ERel
by(blast dest: simE)
from PTrans }\langlea\not=b\rangle\mathrm{ have [aキb]P>>>l}\alpha\prec\mp@subsup{P}{}{\prime}\mathbf{by}(rule Weak-Late-Step-Semantics.Mismatch)
with PRel RelRel' show ?case by blast
qed
qed
qed
lemma sumCompose:
fixes }P::p
and }Q::p
and }R::p
and T:: pi
assumes PSimQ: P}\rightsquigarrow<\mathrm{ Rel }>
and RSimT:R}\rightsquigarrow<\mathrm{ Rel> T
and RelRel':Rel \subseteqRel'
shows }P\oplusR\rightsquigarrow<\mp@subsup{Rel}{}{\prime}>Q\oplus
proof(induct rule: simCases)
case(Bound Q' a x)
have }x\sharpP\oplusR\mathrm{ by fact
hence xFreshP:(x::name) \sharpP and xFreshR: }x\sharpR\mathrm{ by simp +
have }Q\oplusT\longmapstoa<\nux>\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
thus ?case
proof(induct rule: sumCases)
case cSum1
have }Q\longmapstoa<\nux><\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
with xFreshP PSimQ obtain P' where PTrans: P >>la<\nux>}\prec\mp@subsup{P}{}{\prime}a<\mp@subsup{P}{}{\prime}\mathrm{ and
P'RelQ': (P', Q') \in Rel
by(blast dest: simE)
from PTrans have P }\oplusR\Longrightarrow\mp@subsup{\Longrightarrow}{l}{}a<\nux>\prec\mp@subsup{P}{}{\prime}\mathbf{by}(rule Weak-Late-Step-Semantics.Sum1)
moreover from P'RelQ' RelRel' have ( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\inRel' by blas
ultimately show ?case by blast
next
case cSum2
have}T\longmapstoa<\nux>\prec\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
with xFreshR RSimT obtain R' where RTrans: R \Longrightarrow\Longrightarrowla<\nux>}\prec<\mp@subsup{R}{}{\prime}\mathrm{ and
R'RelQ':( }\mp@subsup{R}{}{\prime},\mp@subsup{Q}{}{\prime})\inRe
by(blast dest: simE)
from RTrans have P\oplusR \Longrightarrow\Longrightarrowla<\nux><\prec R'\mathbf{by}(rule Weak-Late-Step-Semantics.Sum2)
moreover from R'RelQ' RelRel' have ( }\mp@subsup{R}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mathrm{ Rel' by blast
ultimately show ?thesis by blast
qed
next
case(Input Q' a x)

```
```

    have \(x \sharp P \oplus R\) by fact
    hence \(x\) Fresh \(P\) : \((x::\) :name \() \sharp P\) and \(x\) Fresh \(R\) : \(x \sharp R\) by simp +
    have \(Q \oplus T \longmapsto a<x>\prec Q^{\prime}\) by fact
    thus ?case
    proof (induct rule: sumCases)
    case \(c\) Sum1
    have \(Q \longmapsto a<x>\prec Q^{\prime}\) by fact
    with \(x\) Fresh \(P\) PSimQ obtain \(P^{\prime \prime}\) where L1: \(\forall u . \exists P^{\prime} . P \Longrightarrow_{l} u\) in \(P^{\prime \prime} \rightarrow a<x>\)
    $\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}[x::=u]\right) \in \operatorname{Rel}$
by (blast dest: simE)
have $\forall u . \exists P^{\prime} . P \oplus R \Longrightarrow_{l} u$ in $P^{\prime \prime} \rightarrow a<x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}[x::=u]\right) \in \operatorname{Rel}^{\prime}$
proof (rule allI)
fix $u$
from L1 obtain $P^{\prime}$ where PTrans: $P \Longrightarrow_{l} u$ in $P^{\prime \prime} \rightarrow a<x>\prec P^{\prime}$
and $P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}[x::=u]\right) \in$ Rel by blast
from PTrans have $P \oplus R \Longrightarrow{ }_{l}$ u in $P^{\prime \prime} \rightarrow a<x>\prec P^{\prime} \mathbf{b y}$ (rule Weak-Late-Step-Semantics.Sum1)
with $P^{\prime}$ RelQ $Q^{\prime}$ RelRel' show $\exists P^{\prime} . P \oplus R \Longrightarrow_{l} u$ in $P^{\prime \prime} \rightarrow a<x>\prec P^{\prime} \wedge\left(P^{\prime}\right.$,
$\left.Q^{\prime}[x::=u]\right) \in$ Rel $^{\prime}$ by blast
qed
thus? ?ase by blast
next
case cSum2
have $T \longmapsto a<x>\prec Q^{\prime}$ by fact
with $x$ Fresh $R$ RSimT obtain $R^{\prime \prime}$ where L1: $\forall u . \exists R^{\prime} . R \Longrightarrow_{l} u$ in $R^{\prime \prime} \rightarrow a<x>$
$\prec R^{\prime} \wedge\left(R^{\prime}, Q^{\prime}[x::=u]\right) \in$ Rel
by (blast dest: simE)
have $\forall u . \exists P^{\prime} . P \oplus R \Longrightarrow_{l} u$ in $R^{\prime \prime} \rightarrow a<x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}[x::=u]\right) \in R_{e l}{ }^{\prime}$
proof (rule allI)
fix $u$
from L1 obtain $R^{\prime}$ where RTrans: $R \Longrightarrow_{l} u$ in $R^{\prime \prime} \rightarrow a<x>\prec R^{\prime}$
and $R^{\prime}$ Rel $Q^{\prime}:\left(R^{\prime}, Q^{\prime}[x::=u]\right) \in$ Rel by blast
from RTrans have $P \oplus R \Longrightarrow{ }_{l}$ u in $R^{\prime \prime} \rightarrow a<x>\prec R^{\prime}$ by (rule Weak-Late-Step-Semantics.Sum2)
with $R^{\prime}$ RelQ' RelRel' show $\exists P^{\prime} . P \oplus R \Longrightarrow_{l} u$ in $R^{\prime \prime} \rightarrow a<x>\prec P^{\prime} \wedge\left(P^{\prime}\right.$,
$\left.Q^{\prime}[x::=u]\right) \in$ Rel $^{\prime}$ by blast
qed
thus ?case by blast
qed
next
case (Free $\left.Q^{\prime} \alpha\right)$
have $Q \oplus T \longmapsto \alpha \prec Q^{\prime}$ by fact
thus ?case
proof (induct rule: sumCases)
case $c$ Sum1
have $Q \longmapsto \alpha \prec Q^{\prime}$ by fact
with PSimQ obtain $P^{\prime}$ where PTrans: $P \Longrightarrow{ }_{l} \alpha \prec P^{\prime}$ and PRel: $\left(P^{\prime}, Q^{\prime}\right) \in$
Rel
by (blast dest: simE)
from PTrans have $P \oplus R \Longrightarrow{ }_{l} \alpha \prec P^{\prime}$ by(rule Weak-Late-Step-Semantics.Sum1)
with RelRel' PRel show ?case by blast

```
```

    next
    case cSum2
    have }T\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    with RSimT obtain R' where RTrans: R \Longrightarrow\Longrightarrowl}\alpha\prec\mp@subsup{|}{}{\prime}\mathrm{ and RRel: ( }\mp@subsup{R}{}{\prime},\mp@subsup{Q}{}{\prime})
    Rel
by(blast dest: simE)
from RTrans have P}\oplusR\Longrightarrow\mp@subsup{}{l}{}\alpha\prec\mp@subsup{R}{}{\prime}\mathbf{by}\mathrm{ (rule Weak-Late-Step-Semantics.Sum2)
with RelRel' RRel show ?case by blast
qed
qed
lemma sumPres:
fixes }P::p
and }Q::p
and }R:: p
assumes PSimQ: P\rightsquigarrow<Rel>}
and Id:Id\subseteq Rel
and RelRel':Rel }\subseteqRel'
shows }P\oplusR\rightsquigarrow<\mp@subsup{Rel}{}{\prime}>Q\oplus
proof -
from Id have Refl: R}\rightsquigarrow<\mathrm{ Rel> R by(rule reflexive)
from PSimQ Refl RelRel' show ?thesis by(rule sumCompose)
qed
lemma parPres:
fixes P :: pi
and }Q\quad::p
and R :: pi
and Rel :: (pi\timespi) set
and Rel' :: (pi\times pi) set
assumes PSimQ: P}\rightsquigarrow<\mathrm{ Rel }>
and PRelQ: }(P,Q)\in\mathrm{ Rel

```

```

    and Res: \PQa. (P,Q)\inRel'\Longrightarrow(<\nua>P,<\nua>Q)\inRel'
    and EqvtRel: eqvt Rel
    and EqvtRel': eqvt Rel'
    shows P|R\rightsquigarrow<\mp@subsup{Rel}{}{\prime}>Q|R
    using EqvtRel'
proof(induct rule: simCasesCont [where C=(P,Q,R)])
case(Bound Q' a x)
have }x\sharp(P,Q,R)\mathrm{ by fact
hence xFreshP: x\sharpP and xFreshR: x }\sharpR\mathrm{ and }x\sharpQ\mathrm{ by simp+
from <Q| R\longmapstoa<\nux> \prec Q'\rangle\langlex\sharpQ\rangle\langlex\sharpR\rangle\mathrm{ show ?case}
proof(induct rule: parCasesB)
case(cPar1 Q')

```
have \(Q\) Trans: \(Q \longmapsto a<\nu x>\prec Q^{\prime}\) by fact
from \(x\) FreshP PSimQ QTrans obtain \(P^{\prime}\) where PTrans: \(P \Longrightarrow_{l} a<\nu x>\prec P^{\prime}\) and \(P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in\) Rel
by (blast dest: simE)
from PTrans xFreshR have \(P \| R \Longrightarrow_{l} a<\nu x>\prec\left(P^{\prime} \| R\right)\) by (rule Weak-Late-Step-Semantics.Par1B)
moreover from \(P^{\prime} \operatorname{Rel} Q^{\prime}\) have \(\left(P^{\prime}\left\|R, Q^{\prime}\right\| R\right) \in R e l^{\prime}\) by (rule Par)
ultimately show ?case by blast
next
case( \(c\) Par2 \(R^{\prime}\) )
have \(R\) Trans: \(R \longmapsto a<\nu x>\prec R^{\prime}\) by fact
hence \(R \Longrightarrow_{l} a<\nu x>\prec R^{\prime}\)
by (auto simp add: weakTransition-def dest: Weak-Late-Step-Semantics.singleActionChain)
with \(x\) Fresh \(P\) xFresh \(R\) have ParTrans: \(P\left\|R \Longrightarrow_{l} a<\nu x>\prec P\right\| R^{\prime}\) by (blast intro: Weak-Late-Step-Semantics.Par2B)
moreover from \(P R e l Q\) have \(\left(P\left\|R^{\prime}, Q\right\| R^{\prime}\right) \in \operatorname{Rel}^{\prime}\) by (rule Par)
ultimately show? case by blast
qed
next
case (Input \(Q^{\prime}\) a \(x\) )
have \(x \sharp(P, Q, R)\) by fact
hence xFreshP: \(x \sharp P\) and xFreshR: \(x \sharp R\) and \(x \sharp Q\) by simp +
from \(\left.\langle Q \| R \longmapsto a<x\rangle \prec Q^{\prime}\right\rangle\langle x \sharp Q\rangle\langle x \sharp R\rangle\)
show ?case
proof (induct rule: parCasesB)
case( \(c\) Par1 \(Q^{\prime}\) )
have \(Q\) Trans: \(Q \longmapsto a<x>\prec Q^{\prime}\) by fact
from \(x\) FreshP PSimQ QTrans obtain \(P^{\prime \prime}\)
where L1: \(\forall u . \exists P^{\prime} . P \Longrightarrow_{l} u\) in \(P^{\prime \prime} \rightarrow a<x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}[x::=u]\right) \in \operatorname{Rel}\) by (blast dest: simE)
have \(\forall u . \exists P^{\prime} . P \| R \Longrightarrow_{l} u\) in \(\left(P^{\prime \prime} \| R\right) \rightarrow a<x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}[x::=u] \|\right.\)
\(R[x::=u]) \in \operatorname{Rel}^{\prime}\)
proof (rule allI)
fix \(u\)
from L1 obtain \(P^{\prime}\) where PTrans: \(P \Longrightarrow_{l} u\) in \(P^{\prime \prime} \rightarrow a<x>\prec P^{\prime}\)
and \(P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}[x::=u]\right) \in\) Rel by blast
from PTrans xFresh \(R\) have \(P \| R \Longrightarrow_{l} u\) in \(\left(P^{\prime \prime} \| R\right) \rightarrow a<x>\prec\left(P^{\prime} \| R\right)\)
by (rule Weak-Late-Step-Semantics.Par1B)
moreover from \(P^{\prime} \operatorname{Rel} Q^{\prime}\) have \(\left(P^{\prime}\left\|R, Q^{\prime}[x::=u]\right\| R\right) \in \operatorname{Rel}^{\prime}\)
by (rule Par)
ultimately show \(\exists P^{\prime} . P \| R \Longrightarrow_{l} u\) in \(\left(P^{\prime \prime} \| R\right) \rightarrow a<x>\prec P^{\prime} \wedge\left(P^{\prime}\right.\),
\(\left.Q^{\prime}[x::=u] \|(R[x::=u])\right) \in\) Rel \(^{\prime}\)
using \(x\) Fresh \(R\)
by(force simp add: forget)
qed
thus ?case by force
next
case( \(c\) Par2 \(R^{\prime}\) )
have RTrans: \(R \longmapsto a<x>\prec R^{\prime}\) by fact
```

    have \(\forall u . \exists P^{\prime} . P \| R \Longrightarrow_{l} u\) in \(\left(P \| R^{\prime}\right) \rightarrow a<x>\prec P^{\prime} \wedge\left(P^{\prime}, Q \| R^{\prime}[x::=u]\right)\)
    $\in R e l^{\prime}$
proof
fix $u$
from $R$ Trans have $R \Longrightarrow_{l} u$ in $R^{\prime} \rightarrow a<x>\prec R^{\prime}[x::=u]$
by (rule Weak-Late-Step-Semantics.singleActionChain)
hence $P \| R \Longrightarrow_{l} u$ in $\left.P \| R^{\prime} \rightarrow a<x\right\rangle \prec P \| R^{\prime}[x::=u]$ using $\langle x \sharp P\rangle$
by (rule Weak-Late-Step-Semantics.Par2B)
moreover from $P \operatorname{Rel} Q$ have $\left(P\left\|R^{\prime}[x::=u], Q\right\| R^{\prime}[x::=u]\right) \in \operatorname{Rel}^{\prime}$ by $($ rule
Par)
ultimately show $\exists P^{\prime} . P \| R \Longrightarrow_{l} u$ in $\left(P \| R^{\prime}\right) \rightarrow a<x>\prec P^{\prime} \wedge$
$\left(P^{\prime}, Q \| R^{\prime}[x::=u]\right) \in$ Rel $^{\prime}$ by blast
qed
thus ?case using $\langle x \sharp Q\rangle$ by (fastforce simp add: forget)
qed
next
case (Free $Q R^{\prime} \alpha$ )
have $Q \| R \longmapsto \alpha \prec Q R^{\prime}$ by fact
thus? case
proof $($ induct rule: parCasesF[of $\cdots-(P, R)])$
case(cPar1 $Q^{\prime}$ )
have $Q \longmapsto \alpha \prec Q^{\prime}$ by fact
with PSimQ obtain $P^{\prime}$ where PTrans: $P \Longrightarrow_{l} \alpha \prec P^{\prime}$ and PRel: $\left(P^{\prime}, Q^{\prime}\right) \in$
Rel
by (blast dest: simE)
from PTrans have Trans: $P\left\|R \Longrightarrow_{l} \alpha \prec P^{\prime}\right\| R$ by (rule Weak-Late-Step-Semantics.Par1F)
moreover from PRel have $\left(P^{\prime}\left\|R, Q^{\prime}\right\| R\right) \in$ Rel $^{\prime}$ by (blast intro: Par)
ultimately show ?case by blast
next
case ( $c$ Par2 $R^{\prime}$ )
have $R \longmapsto \alpha \prec R^{\prime}$ by fact
hence $R \Longrightarrow{ }_{l} \alpha \prec R^{\prime}$
by(rule Weak-Late-Step-Semantics.singleActionChain)
hence $P \| R \Longrightarrow{ }_{l} \alpha \prec\left(P \| R^{\prime}\right)$ by (rule Weak-Late-Step-Semantics.Par2F)
moreover from PRelQ have $\left(P\left\|R^{\prime}, Q\right\| R^{\prime}\right) \in$ Rel $^{\prime}$ by (blast intro: Par)
ultimately show ?case by blast
next
case $\left(c \operatorname{Comm1} Q^{\prime} R^{\prime}\right.$ a b $x$ )
have $Q$ Trans: $Q \longmapsto a<x>\prec Q^{\prime}$ and RTrans: $R \longmapsto a[b] \prec R^{\prime}$ by fact +
have $x \sharp(P, R)$ by fact
hence $x$ Fresh $P: x \sharp P$ by (simp add: fresh-prod)
from PSimQ QTrans xFreshP obtain $P^{\prime} P^{\prime \prime}$ where PTrans: $P \Longrightarrow_{l} b$ in
$P^{\prime \prime} \rightarrow a<x>\prec P^{\prime}$
and $P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}[x::=b]\right) \in \operatorname{Rel}$
by (blast dest: $\operatorname{sim} E)$
from RTrans have $R \Longrightarrow_{l} a[b] \prec R^{\prime}$
by(rule Weak-Late-Step-Semantics.singleActionChain)

```
```

    with PTrans have P|R\Longrightarrow\Longrightarrow}\mp@subsup{l}{l}{}\tau\prec\mp@subsup{P}{}{\prime}|\mp@subsup{R}{}{\prime}\mathbf{by}(rule Weak-Late-Step-Semantics.Comm1)
    moreover from P'RelQ' have ( }\mp@subsup{P}{}{\prime}|\mp@subsup{R}{}{\prime},\mp@subsup{Q}{}{\prime}[x::=b]| R')\inRel' by(rule Par
    ultimately show ?case by blast
    next
case(cComm2 Q' R' a b x)
have QTrans: }Q\longmapstoa[b]\prec\mp@subsup{Q}{}{\prime}\mathrm{ and RTrans: }R\longmapstoa<x>\prec\mp@subsup{R}{}{\prime}\mathrm{ by fact+
have }x\sharp(P,R)\mathrm{ by fact
hence xFreshR: x\sharpR by(simp add: fresh-prod)
from PSimQ QTrans obtain P' where PTrans: P \Longrightarrow\Longrightarrow}\mp@subsup{l}{l}{}a[b]\prec\mp@subsup{P}{}{\prime
and PRel: (P', Q')\inRel
by(blast dest: simE)
from RTrans have R \Longrightarrow>l b in R'->a<x> \prec R'[x::=b]
by(rule Weak-Late-Step-Semantics.singleActionChain)
with PTrans have P|R\Longrightarrow\mp@subsup{}{l}{}\tau\prec\mp@subsup{P}{}{\prime}| R'[x::=b] by(rule Weak-Late-Step-Semantics.Comm2)
moreover from PRel have ( }\mp@subsup{P}{}{\prime}|\mp@subsup{R}{}{\prime}[x::=b],\mp@subsup{Q}{}{\prime}|\mp@subsup{R}{}{\prime}[x::=b])\inRel' by(rule
Par)
ultimately show ?case by blast
next
case(cClose1 Q' R' a x y)
have QTrans: Q\longmapstoa<x>\prec林 and RTrans: }R\longmapstoa<\nuy>\prec<\mp@subsup{R}{}{\prime}\mathrm{ by fact+
have }x\sharp(P,R)\mathrm{ and }y\sharp(P,R)\mathrm{ by fact+
hence xFreshP: }x\sharpP\mathrm{ and yFreshR: y }\sharpR\mathrm{ and yFreshP: y }\sharpP\mathrm{ by(simp add:
fresh-prod)+

```
from PSimQ QTrans xFreshP obtain \(P^{\prime} P^{\prime \prime}\) where PTrans: \(P \Longrightarrow_{l} y\) in \(P^{\prime \prime} \rightarrow a<x>\prec P^{\prime}\)
and \(P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}[x::=y]\right) \in \operatorname{Rel}\)
by (blast dest: simE)
from RTrans have \(R \Longrightarrow_{l} a<\nu y>\prec R^{\prime}\)
by(auto simp add: weakTransition-def dest: Weak-Late-Step-Semantics.singleActionChain)
with PTrans have Trans: \(P \| R \Longrightarrow_{l} \tau \prec<\nu y>\left(P^{\prime} \| R^{\prime}\right)\) using yFreshP
\(y\) FreshR
by(rule Weak-Late-Step-Semantics.Close1)
moreover from \(P^{\prime} \operatorname{Rel} Q^{\prime}\) have \(\left(<\nu y>\left(P^{\prime} \| R^{\prime}\right),<\nu y>\left(Q^{\prime}[x::=y] \| R^{\prime}\right)\right) \in \operatorname{Rel}^{\prime}\) by (blast intro: Par Res)
ultimately show ?case by blast
next
case \(\left(\right.\) cClose2 \(Q^{\prime} R^{\prime}\) a \(x\) y)
have \(Q\) Trans: \(Q \longmapsto a<\nu y>\prec Q^{\prime}\) and RTrans: \(R \longmapsto a<x>\prec R^{\prime}\) by fact +
have \(x \sharp(P, R)\) and \(y \sharp(P, R)\) by fact+
hence \(x\) FreshR: \(x \sharp R\) and \(y\) Fresh \(P: y \sharp P\) and \(y\) Fresh \(R\) : \(y \sharp R\) by (simp add:
fresh-prod)+
from PSimQ QTrans yFreshP obtain \(P^{\prime}\) where PTrans: \(P \Longrightarrow_{l} a<\nu y>\prec P^{\prime}\) and \(P^{\prime}\) RelQ \({ }^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}\)
by (blast dest: simE)
```

    from RTrans have R\Longrightarrowly in R}\mp@subsup{R}{}{\prime}->a<x><<\mp@subsup{R}{}{\prime}[x::=y
            by(rule Weak-Late-Step-Semantics.singleActionChain)
    ```

```

yFreshR
by(rule Weak-Late-Step-Semantics.Close2)
moreover from P'RelQ' have (<\nuy> (\mp@subsup{P}{}{\prime}| R'[x::=y]),<\nuy>( (\mp@subsup{Q}{}{\prime}|\mp@subsup{R}{}{\prime}[x::=y]))
Rel'
by(blast intro: Par Res)
ultimately show ?case by blast
qed
qed
lemma resPres:
fixes }P\quad::p
and }Q :: p
and Rel :: (pi\times pi) set
and x :: name
and Rel':: (pi\times pi) set
assumes PSimQ: P}\rightsquigarrow<\mathrm{ Rel }>
and ResRel: \bigwedge(P::pi) (Q::pi) (x::name). (P,Q)\inRel \Longrightarrow(<\nux>P,<\nux>Q )
Rel'
and RelRel':Rel }\subseteqRe\mp@subsup{R}{}{\prime
and EqvtRel: eqvt Rel
and EqvtRel': eqvt Rel'
shows <\nux>P}\rightsquigarrow<<\mp@subsup{Rel}{}{\prime}><\nux>>
proof -
from EqvtRel' show ?thesis
proof(induct rule: simCasesCont[of - (P,Q,x)])
case(Bound Q' a y)
have Trans: <\nux>Q \longmapstoa<\nuy>}\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
have }y\sharp(P,Q,x)\mathrm{ by fact
hence yineqx: y\not=x and yFreshP: y\sharpP and y\sharpQ by(simp add: fresh-prod)+
from Trans }\langley\not=x\rangle\langley\sharpQ\rangle\mathrm{ show ?case
proof(induct rule: resCasesB)
case(cOpen a Q')
have QTrans: }Q\longmapstoa[x]\prec\mp@subsup{Q}{}{\prime}\mathrm{ and aineqx: a \# x by fact+
from PSimQ QTrans obtain P' where PTrans: P \Longrightarrow>}\mp@subsup{l}{}{\prime}a[x]\prec\mp@subsup{P}{}{\prime
and P'RelQ':( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\operatorname{Rel
by(blast dest: simE)
have }<\nux>P\Longrightarrow\mp@subsup{\Longrightarrow}{l}{}a<\nuy><([(y,x)]\cdot\mp@subsup{P}{}{\prime}
proof -
from PTrans aineqx have }<\nux>P\Longrightarrow\mp@subsup{\Longrightarrow}{l}{}a<\nux><\prec\mp@subsup{P}{}{\prime}\mathbf{by}(\mathrm{ rule Weak-Late-Step-Semantics.Open)
moreover from PTrans yFreshP have y \# P' by (force intro:Weak-Late-Step-Semantics.freshTransition)
ultimately show ?thesis by(simp add: alphaBoundResidual name-swap)
qed

```
```

    moreover from EqvtRel P'RelQ' RelRel' have ([(y,x)] • P', [(y,x)] \cdot Q')\in
    Rel'
by(blast intro: eqvtRelI)
ultimately show ?case by blast
next
case(cRes Q')
have QTrans: Q\longmapstoa<\nuy> \prec-Q' by fact
from \langlex\sharp BoundOutputS a> have }x\not=a\mathrm{ by simp
from PSimQ yFreshP QTrans obtain P' where PTrans: P >}\mp@subsup{>}{l}{}a<\nuy><
P'
and P'RelQ':(P', Q')\inRel
by(blast dest: simE)
from PTrans }\langlex\not=a\rangle yineqx yFreshP have ResTrans: <\nux>P \Longrightarrow\Longrightarrowll a<\nuy>\prec
(<\nux>>P')
by(blast intro:Weak-Late-Step-Semantics.ResB)
moreover from P'RelQ' have ((<\nux>> ' )},(<\nux>\mp@subsup{Q}{}{\prime}))\inRe\mp@subsup{|}{}{\prime
by(rule ResRel)
ultimately show ?case by blast
qed
next
case(Input Q' a y)
have }y\sharp(P,Q,x)\mathrm{ by fact
hence yineqx: y\not=x and yFreshP: y\sharpP and y\sharpQ by(simp add: fresh-prod)+
have }<\nux>Q\longmapsto\longmapstoa<y>\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
thus ?case using yineqx «y \#Q>
proof(induct rule: resCasesB)
case(cOpen a Q')
thus ?case by simp
next
case(cRes Q')
have QTrans: Q\longmapstoa<y>\prec Q' by fact
from <x \# InputS a〉 have x\not=a by simp
from PSimQ QTrans yFreshP obtain P'
where L1: \forallu. \exists\mp@subsup{P}{}{\prime}.P\Longrightarrow\Longrightarrow}\mp@subsup{l}{l}{}u\mathrm{ in }\mp@subsup{P}{}{\prime\prime}->a<y>< \mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},Q'[y::=u])\inRe
by(blast dest: simE)
have \forallu.\exists\mp@subsup{P}{}{\prime}.<\nux>P\mp@subsup{\Longrightarrow}{l}{}u\mathrm{ in }(<\nux>\mp@subsup{P}{}{\prime\prime})->a<y>\prec\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},(<\nux>\mp@subsup{Q}{}{\prime})[y::=u])
Rel'
proof(rule allI)
fix }

```

```

\inRel'
proof(cases x=u)
assume xequ: x=u
have \existsc::name.c\sharp(P, P',},\mp@subsup{Q}{}{\prime},x,y,a) by(blast intro: name-exists-fresh
then obtain c::name where cFreshP:c\sharpP and cFresh\mp@subsup{P}{}{\prime\prime}:c\sharpP\mp@subsup{P}{}{\prime\prime}\mathrm{ and}

```
cFresh \(Q^{\prime}: c \sharp Q^{\prime}\)
and cineqx: \(c \neq x\) and cineqy: \(c \neq y\) and cineqa: \(c \neq a\)
by (force simp add: fresh-prod)
from L1 obtain \(P^{\prime}\) where PTrans: \(P \Longrightarrow_{l}\) c in \(P^{\prime \prime} \rightarrow a<y>\prec P^{\prime}\)
and \(P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}[y::=c]\right) \in \operatorname{Rel}\)
by blast
have \(<\nu x>P \Longrightarrow{ }_{l} u\) in \(\left(<\nu x>P^{\prime \prime}\right) \rightarrow a<y>\prec<\nu c>\left([(x, c)] \cdot P^{\prime}\right)\)
proof -
from PTrans yineqx \(\langle x \neq a\rangle\) cineqx have \(<\nu x>P \Longrightarrow{ }_{l} c\) in \(\left(<\nu x>P^{\prime \prime}\right) \rightarrow a<y>\) \(\prec<\nu x>P^{\prime}\)
by (blast intro: Weak-Late-Step-Semantics.ResB)
hence \(([(x, c)] \cdot<\nu x>P) \Longrightarrow_{l}([(x, c)] \cdot c)\) in \(\left([(x, c)] \cdot<\nu x>P^{\prime \prime}\right) \rightarrow([(x\), \(c)] \cdot a)<([(x, c)] \cdot y)>\prec[(x, c)] \cdot<\nu x>P^{\prime}\)
by(rule Weak-Late-Step-Semantics.eqvtI)
moreover from \(c\) Fresh \(P\) have \(<\nu c>([(x, c)] \cdot P)=<\nu x>P\) by \((\) simp add: alphaRes)
moreover from \(c\) Fresh \(P^{\prime \prime}\) have \(\left.<\nu c\right\rangle\left([(x, c)] \cdot P^{\prime \prime}\right)=<\nu x>P^{\prime \prime}\) by \((\) simp add: alphaRes)
ultimately show ?thesis using \(\langle x \neq a\rangle\) cineqa yineqx cineqy cineqx xequ by(simp add: name-calc)
qed
moreover have \(\left(<\nu c>\left([(x, c)] \cdot P^{\prime}\right),\left(<\nu x>Q^{\prime}\right)[y::=u]\right) \in\) Rel \(^{\prime}\)
proof -
from \(P^{\prime} \operatorname{Rel} Q^{\prime}\) have \(\left(<\nu x>P^{\prime},<\nu x>\left(Q^{\prime}[y::=c]\right)\right) \in \operatorname{Rel}^{\prime}\) by (rule ResRel)
with EqvtRel' have \(\left([(x, c)] \cdot<\nu x>P^{\prime},[(x, c)] \cdot<\nu x>\left(Q^{\prime}[y::=c]\right)\right) \in\)
Rel' by(rule eqvtRelI)
with cineqy yineqx cineqx have \(\left(<\nu c>\left([(x, c)] \cdot P^{\prime}\right),(<\nu c>([(x, c)] \cdot\right.\) \(\left.\left.\left.Q^{\prime}\right)\right)[y::=x]\right) \in \operatorname{Rel}^{\prime}\)
by (simp add: name-calc eqvt-subs)
with \(c\) Fresh \(Q^{\prime}\) xequ show ?thesis by (simp add: alphaRes)
qed
ultimately show ?thesis by blast
next
assume xinequ: \(x \neq u\)
from L1 obtain \(P^{\prime}\) where PTrans: \(P \Longrightarrow_{l} u\) in \(P^{\prime \prime} \rightarrow a<y>\prec P^{\prime}\)
and \(P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}[y::=u]\right) \in\) Rel by blast
from PTrans \(\langle x \neq a\rangle\) yineqx xinequ have \(<\nu x>P \Longrightarrow_{l} u\) in \(\left(<\nu x>P^{\prime \prime}\right) \rightarrow a<y>\) \(\prec<\nu x>P^{\prime}\)
by (blast intro: Weak-Late-Step-Semantics.ResB)
moreover from \(P^{\prime} R e l Q^{\prime}\) xinequ yineqx have \(\left.\left(<\nu x>P^{\prime},(<\nu x\rangle Q^{\prime}\right)[y::=u]\right)\) \(\in R e l^{\prime}\)
by (force intro: ResRel)
ultimately show ?thesis by blast
qed
qed
thus ?case by blast
qed
```

    next
        case(Free Q' \alpha)
        have <\nux>Q \longmapsto < \prec ', by fact
    thus ?case
    proof(induct rule: resCasesF)
        case(cRes Q')
        have }Q\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
        with PSimQ obtain P' where PTrans: P\Longrightarrow\Longrightarrowl}\alpha\prec\mp@subsup{}{l}{\prime}
                        and P}\mp@subsup{P}{}{\prime}\mathrm{ RelQ': ( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mathrm{ Rel
            by(blast dest: simE)
        have <\nux>P \Longrightarrow\Longrightarrowl}\alpha\prec<\nux>\mp@subsup{P}{}{\prime
        proof -
            have xFreshAlpha: x # 人 by fact
            with PTrans show ?thesis by(rule Weak-Late-Step-Semantics.ResF)
        qed
    ```

```

        ultimately show ?case by blast
        qed
    qed
    qed
lemma bangPres:
fixes P :: pi
and Q :: pi
and Rel :: (pi × pi) set
assumes PSimQ: }P\rightsquigarrow<\mp@subsup{\mathrm{ Rel }}{}{\prime}>
and PRelQ: }(P,Q)\in\mathrm{ Rel
and Sim: }\PQ.(P,Q)\in\operatorname{Rel}\LongrightarrowP\rightsquigarrow<\mp@subsup{\operatorname{Rel}}{}{\prime}>
and RelRel': \PQ. (P,Q) \inRel \Longrightarrow(P,Q) \in Rel'
and eqvtRel': eqvt Rel'
shows !P}\rightsquigarrow<bangRel Rel'>!Q
proof -
from eqvtRel' have EqvtBangRel': eqvt(bangRel Rel') by(rule eqvtBangRel)
from RelRel' have BRelRel': \P Q. (P,Q) \in bangRel Rel \Longrightarrow (P,Q) \in bangRel
Rel'
by(auto intro: bangRelSubset)
have }\RsP.\llbracket!Q\longmapstoRs;(P,!Q)\in\mathrm{ bangRel Rel】 \# weakStepSimAct P Rs P
(bangRel Rel')
proof -
fix Rs P
assume !Q\longmapstoRs and (P,!Q)\in bangRel Rel
thus weakStepSimAct P Rs P (bangRel Rel')
proof(nominal-induct avoiding: P rule: bangInduct)
case(cPar1B aa x Q P P)
have QTrans: }Q\longmapstoaa«x»\prec\mp@subsup{Q}{}{\prime}\mathrm{ and xFreshQ: x \# Q by fact+

```
```

    have (P,Q|!Q)\in bangRel Rel and }x\sharpP\mathrm{ by fact+
    thus ?case
    proof(induct rule: BRParCases)
    case(BRPar P R)
        have PRelQ:(P,Q)\inRel and RBangRelQ:(R,!Q)\in bangRel Rel by
    fact+
have }x\sharpP||\mathrm{ by fact
hence xFreshP: x\sharpP and xFreshR: x }\sharpR\mathrm{ by simp+
from PRelQ have PSimQ: P\rightsquigarrow<\mp@subsup{Rel}{\prime}{\prime}>Q by(rule Sim)
from EqvtBangRel' show ?case
proof(induct rule: simActBoundCases)
case(Input a)
have aa= InputS a by fact
with PSimQ QTrans xFreshP obtain P'\prime
where L1:\forallu.\exists\mp@subsup{P}{}{\prime}.P\Longrightarrow\mp@subsup{\Longrightarrow}{l}{}u\mathrm{ in }\mp@subsup{P}{}{\prime\prime}->a<x>\prec\prec P'^( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime}[x::=u])
Rel'
by(blast dest: simE)
have }\forallu.\exists\mp@subsup{P}{}{\prime}.P|R\Longrightarrow\mp@subsup{\Longrightarrow}{l}{}u\mathrm{ in }(\mp@subsup{P}{}{\prime\prime}|R)->a<x>< < P'^( (P',( (Q'
!Q)[x::=u]) \in bangRel Rel'
proof(rule allI)
fix u
from L1 obtain P' where PTrans: P}\mp@subsup{\Longrightarrow}{l}{
and }\mp@subsup{P}{}{\prime}\operatorname{RelQ}\mp@subsup{Q}{}{\prime}:(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime}[x::=u])\inRe\mp@subsup{l}{}{\prime
by blast
from PTrans xFreshR have P|R \Longrightarrow>lu in ( }\mp@subsup{P}{}{\prime\prime}|R)->a<x>\prec P'|
by(rule Weak-Late-Step-Semantics.Par1B)
moreover have ( }\mp@subsup{P}{}{\prime}|R,(\mp@subsup{Q}{}{\prime}|!Q)[x::=u])\in\mathrm{ bangRel Rel'
proof -
from P'RelQ' RBangRelQ have ( }\mp@subsup{P}{}{\prime}|R,\mp@subsup{Q}{}{\prime}[x::=u]|!Q)\in\mathrm{ bangRel
Rel'
by(blast intro: BRelRel' Rel.BRPar)
with xFreshQ show ?thesis by(force simp add: forget)
qed
ultimately show }\exists\mp@subsup{P}{}{\prime}.P|R\mp@subsup{\Longrightarrow}{l}{}\mathrm{ u in ( }\mp@subsup{P}{}{\prime\prime}|R)->a<x>\prec\prec P'
(P',(\mp@subsup{Q}{}{\prime}|!Q)[x::=u])\in bangRel Rel
by blast
qed
thus ?case by blast
next
case(BoundOutput a)
have aa= BoundOutputS a by fact
with PSimQ QTrans xFreshP obtain P' where PTrans: P \Longrightarrow>l a<\nux>
\prec P ^ { \prime } and P'RelQ':( P ^ { \prime } , Q ^ { \prime } ) \in R _ { Rel }
by(force dest: simE)
from PTrans xFreshR have P|R \Longrightarrow\Longrightarrowl}a<\nux>\prec P'|
by(rule Weak-Late-Step-Semantics.Par1B)
moreover from ( }\mp@subsup{P}{}{\prime}Rel\mp@subsup{Q}{}{\prime}RBangRelQ have ( P'| R, Q'|!Q)\in bangRe
Rel'
by(blast intro: Rel.BRPar BRelRel')

```
```

            ultimately show ?case by blast
        qed
    qed
    next
    case(cPar1F \alpha Q' P)
    have QTrans: Q\longmapsto\alpha\prec Q' by fact
    have (P,Q|!Q)\in bangRel Rel by fact
    thus ?case
    proof(induct rule: BRParCases)
        case(BRPar P R)
            have PRelQ:}(P,Q)\in\mathrm{ Rel and RBangRelQ: (R,!Q) G bangRel Rel by
    fact+
show ?case
proof(induct rule: simActFreeCases)
case Free
from PRelQ have P}\rightsquigarrow<<\mp@subsup{Rel}{l}{\prime}>Q by(rule Sim
with QTrans obtain P' where PTrans: P\Longrightarrow\Longrightarrowl }\alpha\prec\mp@subsup{P}{}{\prime}\mathrm{ and P'RelQ':( (P',
Q')}\in\mp@subsup{R}{Rel}{
by(blast dest: simE)
from PTrans have P|R\Longrightarrow\Longrightarrow}\mp@subsup{}{l}{}\alpha\prec\mp@subsup{P}{}{\prime}|R\mathrm{ by(rule Weak-Late-Step-Semantics.Par1F)
moreover from P'RelQ' RBangRelQ have ( }\mp@subsup{P}{}{\prime}|R,\mp@subsup{Q}{}{\prime}|!Q)\in\mathrm{ bangRel
Rel'
by(blast intro: BRelRel' Rel.BRPar)
ultimately show ?case by blast
qed
qed
next
case(cPar2B aa x Q ' P)
have IH:^P.(P,!Q) \in bangRel Rel \Longrightarrow weakStepSimAct P (aa<x>\prec < Q')
P (bangRel Rel') by fact
have xFreshQ: }x\sharpQ\mathrm{ by fact
have (P,Q|!Q)\in bangRel Rel and }x\sharpP\mathrm{ by fact+
thus ?case
proof(induct rule: BRParCases)
case(BRPar P R)
have PRelQ:(P,Q)\inRel and RBangRelQ:(R,!Q) \in bangRel Rel by
fact+
have }x\sharpP||R\mathrm{ by fact
hence xFreshP: }x\sharpP\mathrm{ and xFreshR: }x\sharpR\mathrm{ by simp+
from RBangRelQ have IH: weakStepSimAct R (aa«x» \prec Q') R (bangRel
Rel') by(rule IH)
from EqvtBangRel' show ?case
proof(induct rule: simActBoundCases)
case(Input a)
have aa=InputS a by fact

```

```

\prec R'^

$$
\left(R^{\prime}, Q^{\prime}[x::=u]\right) \in \text { bangRel Rel }{ }^{\prime}
$$

```
```

            by(simp add: weakStepSimAct-def, blast)
            have }\forallu.\exists\mp@subsup{P}{}{\prime}.P|R\Longrightarrow\mp@subsup{\Longrightarrow}{l}{}u\mathrm{ in }(P|\mp@subsup{R}{}{\prime\prime})->a<x>\prec\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},(Q
    Q})[x::=u])\in\mathrm{ bangRel Rel'
proof(rule allI)
fix u
from L1 obtain R' where RTrans: R \Longrightarrow> }\mp@subsup{|}{l}{}\mathrm{ u in }\mp@subsup{R}{}{\prime\prime}->a<x>\prec R
and R'BangRelT}\mp@subsup{}{}{\prime}:(\mp@subsup{R}{}{\prime},\mp@subsup{Q}{}{\prime}[x::=u])\in\mathrm{ bangRel Rel'
by blast
from RTrans xFreshP have P|R \Longrightarrow\Longrightarrowlu in (P| R'')->a<x>\prec
by(rule Weak-Late-Step-Semantics.Par2B)
moreover have (P| R',(Q| Q')[x::=u]) \in bangRel Rel'
proof -
from PRelQ R'BangRelT' have (P| R',Q| Q ' }|x::=u])\in\mathrm{ bangRel
Rel'
by(blast intro: RelRel' Rel.BRPar)
with xFreshQ show ?thesis by(simp add: forget)
qed
ultimately show }\exists\mp@subsup{P}{}{\prime}.P|R\mp@subsup{\Longrightarrow}{l}{}u\mathrm{ in }(P|\mp@subsup{R}{}{\prime\prime})->a<x>\prec \prec P'^( (P'
(Q| Q ')[x::=u]) \in bangRel Rel'
by blast
qed
thus ?case by blast
next
case(BoundOutput a)
have aa= BoundOutputS a by fact
with IH xFreshR obtain R' where RTrans: R \Longrightarrow\Longrightarrowl}a<\nux><\mp@subsup{R}{}{\prime
and R'BangRelT}\mp@subsup{}{}{\prime}:(\mp@subsup{R}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mathrm{ bangRel Rel'
by(simp add:weakStepSimAct-def, blast)
from RTrans xFreshP have P|R \Longrightarrow\Longrightarrow}\mp@subsup{}{l}{}a<\nux>\precP|\mp@subsup{R}{}{\prime
by(auto intro:Weak-Late-Step-Semantics.Par2B)
moreover from PRelQ R'BangRelT' have ( }P|\mp@subsup{R}{}{\prime},Q|\mp@subsup{Q}{}{\prime})\in\mathrm{ bangRel
Rel'
by(blast intro: RelRel' Rel.BRPar)
ultimately show ?case by blast
qed
qed
next
case(cPar2F \alpha Q')
have IH: \bigwedgeP. (P,!Q) \in bangRel Rel \Longrightarrow weakStepSimAct P ( }\alpha\prec\mp@subsup{Q}{}{\prime})
(bangRel Rel') by fact+
have (P,Q|!Q)\in bangRel Rel by fact
thus ?case
proof(induct rule: BRParCases)
case(BRPar P R)
have PRelQ:(P,Q)\inRel and RBangRelQ:(R,!Q)\in bangRel Rel by
fact+
show ?case

```
proof (induct rule: simActFreeCases)
case Free
from \(R\) BangRelQ have weakStepSimAct \(R\left(\alpha \prec Q^{\prime}\right) R(\) bangRel Rel')
by (rule \(I H\) )
then obtain \(R^{\prime}\) where \(R\) Trans: \(R \Longrightarrow_{l} \alpha \prec R^{\prime}\) and \(R^{\prime}\) BangRelQ': \(\left(R^{\prime}\right.\),
\(\left.Q^{\prime}\right) \in\) bangRel Rel'
by (simp add: weakStepSimAct-def, blast)
from RTrans have \(P\left\|R \Longrightarrow{ }_{l} \alpha \prec P\right\| R^{\prime} \mathbf{b y}\) (rule Weak-Late-Step-Semantics.Par2F) moreover from PRelQ R'BangRelQ' have ( \(P\left\|R^{\prime}, Q\right\| Q^{\prime}\) ) \(\in\) bangRel
\(R e l^{\prime}\)
by(blast intro: RelRel' Rel.BRPar)
ultimately show ?case by blast
qed
qed
next
case (cComm1 a x \(\left.Q^{\prime} b Q^{\prime \prime} P\right)\)
have \(Q\) Trans: \(Q \longmapsto a<x>\prec Q^{\prime}\) by fact
have \(I H: \bigwedge P .(P,!Q) \in\) bangRel Rel \(\Longrightarrow\) weakStepSimAct \(P\left(a[b] \prec Q^{\prime \prime}\right) P\)
(bangRel Rel') by fact+
have \((P, Q \|!Q) \in\) bangRel Rel and \(x \sharp P\) by fact +
thus ?case
proof (induct rule: BRParCases)
case (BRPar P R)
have PRelQ: \((P, Q) \in\) Rel and RBangRel \(Q:(R,!Q) \in\) bangRel Rel by
fact+
have \(x \sharp P \| R\) by fact
hence \(x\) Fresh \(P\) : \(x \sharp P\) by simp
show ? case
proof (induct rule: simActFreeCases)
case Free
from PRelQ have \(P \rightsquigarrow<\) Rel \(^{\prime}>Q\) by(rule Sim)
with QTrans xFreshP obtain \(P^{\prime} P^{\prime \prime}\) where PTrans: \(P \Longrightarrow_{l} b\) in \(P^{\prime \prime} \rightarrow a<x>\)
and \(P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}[x::=b]\right) \in \operatorname{Rel}^{\prime}\)
by (blast dest: simE)
from RBangRelQ have weakStepSimAct \(R\left(a[b] \prec Q^{\prime \prime}\right) R(b a n g R e l ~ R e l ')\)
by (rule \(I H\) )
then obtain \(R^{\prime}\) where \(R\) Trans: \(R \Longrightarrow_{l} a[b] \prec R^{\prime}\)
and \(R^{\prime}\) RelT \(T^{\prime}:\left(R^{\prime}, Q^{\prime \prime}\right) \in\) bangRel Rel \({ }^{\prime}\)
by (simp add: weakStepSimAct-def, blast)
from PTrans RTrans have \(P \| R \Longrightarrow_{l} \tau \prec\left(P^{\prime} \| R^{\prime}\right)\)
by (rule Weak-Late-Step-Semantics.Comm1)
moreover from \(P^{\prime} \operatorname{Rel} Q^{\prime} R^{\prime} \operatorname{Rel} T^{\prime}\) have \(\left(P^{\prime}\left\|R^{\prime}, Q^{\prime}[x::=b]\right\| Q^{\prime \prime}\right) \in\) bangRel Rel'
by(blast intro: RelRel' Rel.BRPar)
ultimately show? case by blast
qed
```

        qed
    next
    case(cComm2 a b Q' x Q '' P)
    have QTrans: Q\longmapstoa[b]\prec Q' by fact
    have IH: \P.(P,!Q) \in bangRel Rel \Longrightarrow weakStepSimAct P (a<x>\prec < Q')
    P(bangRel Rel')
by fact
have (P,Q|!Q)\inbangRel Rel and }x\sharpP\mathrm{ by fact+
thus ?case
proof(induct rule: BRParCases)
case(BRPar P R)
have PRelQ:(P,Q)\inRel and RBangRelQ:(R,!Q)\in bangRel Rel by
fact+
have }x\sharpP|R\mathrm{ by fact
hence xFreshR: x \#R by simp
show ?case
proof(induct rule: simActFreeCases)
case Free
from PRelQ have P }><\mp@subsup{\mathrm{ Rel l}}{}{\prime}>Q by(rule Sim)
with QTrans obtain P' where PTrans: P\Longrightarrow\Longrightarrow}\mp@subsup{|}{l}{}a[b]\prec\mp@subsup{P}{}{\prime
and P'RelQ':}(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mp@subsup{\operatorname{Rel}}{}{\prime
by(blast dest: simE)
from RBangRelQ have weakStepSimAct R (a<x>< < Q') R(bangRel Rel')
by(rule IH)
with xFreshR obtain }\mp@subsup{R}{}{\prime}\mp@subsup{R}{}{\prime\prime}\mathrm{ where RTrans: R >>
and R'BangRelQ'\prime:}(\mp@subsup{R}{}{\prime},\mp@subsup{Q}{}{\prime\prime}[x::=b])\in\mathrm{ bangRel Rel'
by(simp add:weakStepSimAct-def, blast)
from PTrans RTrans have P|R \Longrightarrow>}\mp@subsup{l}{l}{}\tau\prec(\mp@subsup{P}{}{\prime}|\mp@subsup{R}{}{\prime}
by(rule Weak-Late-Step-Semantics.Comm2)
moreover from P'RelQ' R'BangRelQ' have ( }\mp@subsup{P}{}{\prime}|\mp@subsup{R}{}{\prime},\mp@subsup{Q}{}{\prime}||\mp@subsup{Q}{}{\prime\prime}[x::=b]
GangRel Rel'
by(rule Rel.BRPar)
ultimately show ?case by blast
qed
qed
next
case(cClose1 a x Q' y Q '' P)
have QTrans: Q\longmapstoa<x>}\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
have IH: \P. (P,!Q) \in bangRel Rel \LongrightarrowweakStepSimAct P (a<\nuy>\prec Q')
P (bangRel Rel')
by fact
have }(P,Q|!Q)\in\mathrm{ bangRel Rel and }x\sharpP\mathrm{ and }y\sharpP\mathrm{ by fact+
thus ?case
proof(induct rule: BRParCases)
case(BRPar P R)
have PRelQ:(P,Q)\inRel and RBangRelQ: (R,!Q)\in bangRel Rel by

```
```

fact+
have $x \sharp P \| R$ and $y \sharp P \| R$ by fact +
hence $x$ Fresh $P: x \sharp P$ and $y$ FreshR: $y \sharp R$ and $y$ Fresh $P: y \sharp P$ by simp+
show ? case
proof (induct rule: simActFreeCases)
case Free
from PRelQ have $P \rightsquigarrow<$ Rel $^{\prime}>Q$ by(rule Sim)
with QTrans $x$ Fresh $P$ obtain $P^{\prime} P^{\prime \prime}$ where PTrans: $P \Longrightarrow_{l} y$ in $P^{\prime \prime} \rightarrow a<x>$
$\prec P^{\prime}$
and $P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}[x::=y]\right) \in \operatorname{Rel}^{\prime}$
by (blast dest: simE)
from RBangRelQ have weakStepSimAct $R\left(a<\nu y>\prec Q^{\prime \prime}\right) R$ (bangRel
$\left.R e l^{\prime}\right) \mathbf{b y}($ rule $I H)$
with $y$ Fresh $R$ obtain $R^{\prime}$ where $R$ Trans: $R \Longrightarrow_{l} a<\nu y>\prec R^{\prime}$
and $R^{\prime}$ BangRelQ ${ }^{\prime \prime}:\left(R^{\prime}, Q^{\prime \prime}\right) \in$ bangRel Rel ${ }^{\prime}$
by (simp add: weakStepSimAct-def, blast)
from PTrans RTrans yFreshP yFresh $R$ have $P \| R \Longrightarrow_{l} \tau \prec<\nu y>\left(P^{\prime} \|\right.$
$R^{\prime}$ )
by(rule Weak-Late-Step-Semantics.Close1)
moreover from $P^{\prime} \operatorname{Rel} Q^{\prime} R^{\prime}$ BangRelQ ${ }^{\prime \prime}$ have $\left(<\nu y>\left(P^{\prime} \| R^{\prime}\right),<\nu y>\left(Q^{\prime}[x::=y]\right.\right.$
$\left.\left.\| Q^{\prime \prime}\right)\right) \in$ bangRel Rel'
by(force intro: Rel.BRPar Rel.BRRes)
ultimately show ? case by blast
qed
qed
next
case(cClose2 a y $\left.Q^{\prime} x Q^{\prime \prime}\right)$
have $Q$ Trans: $Q \longmapsto a<\nu y>\prec Q^{\prime}$ by fact
have $I H: \wedge P .(P,!Q) \in$ bangRel Rel $\Longrightarrow$ weakStepSimAct $P\left(a<x>\prec Q^{\prime \prime}\right)$
$P$ (bangRel Rel')
by fact
have $(P, Q \|!Q) \in$ bangRel Rel and $x \sharp P$ and $y \sharp P$ by fact +
thus ?case
proof (induct rule: BRParCases)
case (BRPar P R)
have PRelQ: $(P, Q) \in$ Rel and RBangRelQ: $(R,!Q) \in$ bangRel Rel by
fact+
have $x \sharp P \| R$ and $y \sharp P \| R$ by fact +
hence $x$ Fresh $R: x \sharp R$ and $y$ Fresh $R: y \sharp R$ and $y$ Fresh $P: y \sharp P$ by simp +
show ? case
proof (induct rule: simActFreeCases)
case Free
from PRelQ have $P \rightsquigarrow<$ Rel $^{\prime}>Q$ by (rule Sim)
with QTrans yFreshP obtain $P^{\prime}$ where PTrans: $P \Longrightarrow_{l} a<\nu y>\prec P^{\prime}$
and $P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}^{\prime}$
by (blast dest: simE)
from RBangRelQ have weakStepSimAct $R\left(a<x>\prec Q^{\prime \prime}\right) R\left(\right.$ bangRel Rel $\left.{ }^{\prime}\right)$

```
by (rule \(I H\) )
with \(x\) Fresh \(R\) obtain \(R^{\prime} R^{\prime \prime}\) where RTrans: \(R \Longrightarrow{ }_{l} y\) in \(R^{\prime \prime} \rightarrow a<x>\prec R^{\prime}\) and \(R^{\prime}\) BangRelT \({ }^{\prime}:\left(R^{\prime}, Q^{\prime \prime}[x::=y]\right) \in\) bangRel Rel \({ }^{\prime}\) by (simp add: weakStepSimAct-def, blast)
from PTrans RTrans yFreshP yFresh \(R\) have \(P \| R \Longrightarrow_{l} \tau \prec<\nu y>\left(P^{\prime} \|\right.\) \(\left.R^{\prime}\right)\)
by(rule Weak-Late-Step-Semantics.Close2)
moreover from \(P^{\prime}\) RelQ \(Q^{\prime} R^{\prime}\) BangRelT \({ }^{\prime}\) have \(\left(<\nu y>\left(P^{\prime} \| R\right),<\nu y>\left(Q^{\prime}\right.\right.\) \(\left.\left.\| Q^{\prime \prime}[x::=y]\right)\right) \in\) bangRel Rel \({ }^{\prime}\)
by(force intro: Rel.BRPar Rel.BRRes)
ultimately show ?case by blast
qed
qed
next
case (cBang Rs)
have \(I H: \bigwedge P .(P, Q \|!Q) \in\) bangRel Rel \(\Longrightarrow\) weakStepSimAct \(P\) Rs \(P\) (bangRel Rel')
by fact
have \((P,!Q) \in\) bangRel Rel by fact
thus ?case
proof (induct rule: BRBangCases)
case (BRBang P)
have PRelQ: \((P, Q) \in\) Rel by fact
hence \((!P,!Q) \in\) bangRel Rel by (rule Rel.BRBang)
with PRelQ have \((P\|!P, Q\|!Q) \in\) bangRel Rel by (rule Rel.BRPar)
hence weakStepSimAct \((P \|!P)\) Rs \((P \|!P)\) (bangRel Rel') by (rule IH)
thus ?case
proof (simp (no-asm) add: weakStepSimAct-def, auto)
fix \(Q^{\prime}\) a \(x\)
assume weakStepSimAct \((P \|!P)\left(a<\nu x>\prec Q^{\prime}\right)(P \|!P)\left(\right.\) bangRel Rel \(\left.{ }^{\prime}\right)\)
and \(x \sharp P\)
then obtain \(P^{\prime}\) where PTrans: \((P \|!P) \Longrightarrow_{l} a<\nu x>\prec P^{\prime}\)
and \(P^{\prime}\) Rel \(Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in\left(\right.\) bangRel Rel \(\left.{ }^{\prime}\right)\)
by (simp add: weakStepSimAct-def, blast)
from PTrans have \(!P \Longrightarrow_{l} a<\nu x>\prec P^{\prime}\)
by (rule Weak-Late-Step-Semantics.Bang)
with \(P^{\prime}\) RelQ \({ }^{\prime}\) show \(\exists P^{\prime}\). \(!P \Longrightarrow_{l} a<\nu x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in\) bangRel Rel \({ }^{\prime}\)
by blast
next
fix \(Q^{\prime} a x\)
assume weakStepSimAct \((P \|!P)\left(a<x>\prec Q^{\prime}\right)(P \|!P)\left(\right.\) bangRel Rel \(\left.{ }^{\prime}\right)\) and \(x \sharp P\)
then obtain \(P^{\prime \prime}\) where \(L 1: \forall u . \exists P^{\prime} . P \|!P \Longrightarrow_{l} u\) in \(P^{\prime \prime} \rightarrow a<x>\prec P^{\prime}\) \(\wedge\left(P^{\prime}, Q^{\prime}[x::=u]\right) \in\left(\right.\) bangRel Rel \(\left.{ }^{\prime}\right)\)
by (simp add: weakStepSimAct-def, blast)
have \(\forall u . \exists P^{\prime} .!P \Longrightarrow_{l} u\) in \(P^{\prime \prime} \rightarrow a<x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}[x::=u]\right) \in(\) bangRel \(\left.R e l^{\prime}\right)\)
proof(rule allI)
```

            fix }
            from L1 obtain P' where PTrans: P|!P \Longrightarrow\Longrightarrowlu in P'I}->a<x>\prec\prec P
                        and }\mp@subsup{P}{}{\prime}RelQ':(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime}[x::=u])\in(\mathrm{ bangRel Rel')
            by blast
    from PTrans have !P >}\mp@subsup{l}{l}{}u\mathrm{ in }\mp@subsup{P}{}{\prime\prime}->a<x>\prec\mp@subsup{P}{}{\prime}\mathbf{by}(rule Weak-Late-Step-Semantics.Bang)
    with P'RelQ' show }\exists\mp@subsup{P}{}{\prime}.!P>\mp@subsup{\Longrightarrow}{l}{}\mathrm{ u in }\mp@subsup{P}{}{\prime\prime}->a<x>\prec < P'^( ( ', Q Q [x::=u]
    \epsilon(bangRel Rel') by blast
qed
thus \exists}\mp@subsup{P}{}{\prime\prime}.\forallu.\exists\mp@subsup{P}{}{\prime}.!P\mp@subsup{\Longrightarrow}{l}{}u\mathrm{ in }\mp@subsup{P}{}{\prime\prime}->a<x>< \prec P'^( (P', Q [x::=u])
(bangRel Rel') by blast
next
fix }\mp@subsup{Q}{}{\prime}
assume weakStepSimAct (P|!P)(\alpha\prec Q')(P|!P) (bangRel Rel')
then obtain }\mp@subsup{P}{}{\prime}\mathrm{ where PTrans: }(P|!P)\Longrightarrow\mp@subsup{\Longrightarrow}{l}{}\alpha\prec\mp@subsup{P}{}{\prime
and }\mp@subsup{P}{}{\prime}RelQ':(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in(\mathrm{ bangRel Rel')
by(simp add: weakStepSimAct-def, blast)
from PTrans have!P }\mp@subsup{\Longrightarrow}{l}{}\alpha\prec\mp@subsup{P}{}{\prime
by(rule Weak-Late-Step-Semantics.Bang)
with P'RelQ' show }\exists\mp@subsup{P}{}{\prime}.!P\Longrightarrow\mp@subsup{}{l}{}\alpha<\prec\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in(\mathrm{ bangRel Rel') by
blast
qed
qed
qed
qed
moreover from PRelQ have (!P,!Q) \in bangRel Rel by(rule Rel.BRBang)
ultimately show ?thesis by(simp add: weakStepSim-def)
qed
end
theory Weak-Late-Bisim-SC
imports Weak-Late-Bisim Strong-Late-Bisim-SC
begin

```
```

lemma resComm:

```
lemma resComm:
    fixes \(P\) :: \(p i\)
    fixes \(P\) :: \(p i\)
    shows \(<\nu a><\nu b>P \approx<\nu b><\nu a>P\)
    shows \(<\nu a><\nu b>P \approx<\nu b><\nu a>P\)
proof -
proof -
    have \(<\nu a><\nu b>P \sim<\nu b><\nu a>P\) by(rule Strong-Late-Bisim-SC.resComm)
    have \(<\nu a><\nu b>P \sim<\nu b><\nu a>P\) by(rule Strong-Late-Bisim-SC.resComm)
    thus ?thesis by(rule strongBisim WeakBisim)
    thus ?thesis by(rule strongBisim WeakBisim)
qed
```

qed

```
```

lemma matchId:
fixes a :: name
and }P::p
shows [a\frowna]P\approxP
proof -
have [a\frowna]P~P by(rule Strong-Late-Bisim-SC.matchId)
thus?thesis by(rule strongBisim WeakBisim)
qed
lemma mismatchId:
fixes a :: name
and b:: name
and }P::p
assumes }a\not=
shows [a\not=b]P\approxP
proof -
from assms have [a\not=b]P~P by(rule Strong-Late-Bisim-SC.mismatchId)
thus ?thesis by(rule strongBisim WeakBisim)
qed
lemma mismatchZero:
fixes a :: name
and }P::p
shows [a\not=a]P\approx\mathbf{0}
proof -
have [a\not=a]P~0 by(rule Strong-Late-Bisim-SC.mismatchNil)
thus ?thesis by(rule strongBisim WeakBisim)
qed
lemma sumSym:
fixes P :: pi
and }Q:::p
shows }P\oplusQ\approxQ\oplus
proof -
have }P\oplusQ~Q\oplusP\mathrm{ by(rule Strong-Late-Bisim-SC.sumSym)
thus ?thesis by(rule strongBisim WeakBisim)
qed
lemma sumAssoc:
fixes P :: pi

```
```

    and }Q::p
    and }R::p
    shows }(P\oplusQ)\oplusR\approxP\oplus(Q\oplusR
    proof -
have }(P\oplusQ)\oplusR~P\oplus(Q\oplusR)\mathbf{by}(rule Strong-Late-Bisim-SC.sumAssoc
thus?thesis by(rule strongBisim WeakBisim)
qed
lemma sumZero:
fixes P :: pi
shows }P\oplus\mathbf{0}\approx
proof -
have P\oplus\mathbf{0 ~ P by(rule Strong-Late-Bisim-SC.sumZero)}
thus ?thesis by(rule strongBisimWeakBisim)
qed
lemma parZero:
fixes P :: pi
shows }P|\mathbf{0}\approx
proof -
have P|\mathbf{0}~P
thus ?thesis by(rule strongBisim WeakBisim)
qed
lemma parSym:
fixes P :: pi
and }Q::p
shows }P|Q\approxQ|
proof -
have P|Q ~ Q|P吝y(rule Strong-Late-Bisim-SC.parSym)
thus?thesis by(rule strongBisim WeakBisim)
qed
lemma scopeExtPar:
fixes }P:: p
and }Q::p
and x :: name
assumes }x\sharp
shows <\nux>(P|Q)\approxP|<\nux>QQ
proof -
from assms have <\nux> (P|Q)~P|<\nux>Q by(rule Strong-Late-Bisim-SC.scopeExtPar)

```
```

    thus ?thesis by(rule strongBisim WeakBisim)
    qed
lemma scopeExtPar':
fixes P :: pi
and }Q:: p
and x :: name
assumes xFreshQ:x\sharpQ
shows <\nux>(P|Q)\approx(<\nux>P)|Q
proof -
from assms have <\nux>(P|Q)~(<\nux>P)|Q by(rule Strong-Late-Bisim-SC.scopeExtPar')
thus?thesis by(rule strongBisim WeakBisim)
qed
lemma parAssoc:
fixes P :: pi
and }Q::p
and }R::p
shows (P|Q)|R\approxP|(Q|R)
proof -
have (P|Q)|R~P|(Q|R) by(rule Strong-Late-Bisim-SC.parAssoc)
thus?thesis by(rule strongBisim WeakBisim)
qed
lemma freshRes:
fixes }P\mathrm{ :: pi
and a :: name
assumes aFreshP:a\sharpP
shows <\nua>P\approxP
proof -
from assms have <\nua>P ~ P by(rule Strong-Late-Bisim-SC.scopeFresh)
thus?thesis by(rule strongBisim WeakBisim)
qed
lemma scopeExtSum:
fixes }P::p
and }Q::p
and x :: name
assumes }x\sharp
shows <\nux>(P\oplusQ)\approxP\oplus<\nux>Q
proof -
from assms have <\nux> (P\oplusQ) ~P\oplus<\nux>Q by(rule Strong-Late-Bisim-SC.scopeExtSum)

```
```

    thus ?thesis by(rule strongBisimWeakBisim)
    qed
lemma bangSC:
fixes }
shows !P\approxP|!P
proof -
have !P~P|!P by(rule Strong-Late-Bisim-SC.bangSC)
thus?thesis by(rule strongBisim WeakBisim)
qed
end
theory Weak-Late-Sim-Pres
imports Weak-Late-Sim
begin
lemma tauPres:
fixes P :: pi
and Q :: pi
and Rel :: (pi\times pi) set
and Rel'::(pi\times pi) set
assumes PRelQ:(P,Q)\inRel
shows }\tau.(P)\rightsquigarrow^<Rel>\tau.(Q
proof(induct rule: simCases)
case(Bound Q' a x)
have }\tau.(Q)\longmapstoa<\nux> \prec-\mp@subsup{Q}{}{\prime}\mathrm{ by fact
hence False by auto
thus ?case by simp
next
case(Input Q' a x)
have}\tau.(Q)\longmapstoa<x>\prec\prec Q' by fac
hence False by auto
thus ?case by simp
next
case(Free Q' \alpha)
have}\tau.(Q)\longmapsto(\alpha\prec\mp@subsup{Q}{}{\prime})\mathrm{ by fact
thus ?case using PRelQ
proof(induct rule: tauCases, auto simp add: pi.inject residual.inject)
have }\tau.(P)\Longrightarrow\mp@subsup{}{l}{`}\tau\precP\mathrm{ by(rule Tau)
moreover assume (P, Q')\inRel

```

```

    qed
    qed
lemma inputPres:

```
```

    fixes \(P\) :: \(p i\)
    and \(\quad Q \quad:: p i\)
    and \(a\) :: name
    and \(x\) :: name
    and Rel :: \((p i \times p i)\) set
    assumes PRelQ: \(\forall y .(P[x::=y], Q[x::=y]) \in \operatorname{Rel}\)
    and Equt: equt Rel
    shows \(a<x>. P \rightsquigarrow \wedge<\) Rel \(>a<x>. Q\)
    proof -
show ?thesis using Eqvt
proof $($ induct rule: simCasesCont $[$ of $-(P, a, x, Q)])$
case(Bound $Q^{\prime} b y$ )
have $a<x>. Q \longmapsto b<\nu y>\prec Q^{\prime}$ by fact
hence False by auto
thus ?case by simp
next
case(Input $\left.Q^{\prime} b y\right)$
have $y \sharp(P, a, x, Q)$ by fact
hence $y$ FreshP: $(y:: n a m e) \sharp P$ and yineqx: $y \neq x$ and $y \neq a$ and $y \sharp Q$
by (simp add: fresh-prod) +
have $a<x>. Q \longmapsto b<y>\prec Q^{\prime}$ by fact
thus ? case using $\langle y \neq a\rangle\langle y \neq x\rangle\langle y \sharp Q\rangle$
proof $($ induct rule: inputCases, auto simp add: subject.inject)
have $\forall u . \exists P^{\prime} . a<x>. P \Longrightarrow{ }_{l} u$ in $([(x, y)] \cdot P) \rightarrow a<y>\prec P^{\prime} \wedge\left(P^{\prime},([(x, y)]\right.$

- $Q)[y::=u]) \in \operatorname{Rel}$
proof(rule allI)
fix $u$
have $a<x>. P \Longrightarrow{ }_{l} u$ in $([(x, y)] \cdot P) \rightarrow a<y>\prec([(x, y)] \cdot P)[y::=u]$ (is
?goal)
proof -
from $y$ Fresh $P$ have $a<x>. P=a<y>$. $([(x, y)] \cdot P) \mathbf{b y}$ (rule Agent.alphaInput)
moreover have $a<y>.([(x, y)] \cdot P) \Longrightarrow_{l} u$ in $([(x, y)] \cdot P) \rightarrow a<y>\prec([(x$,
$y)] \cdot P)[y::=u]$
by (rule Weak-Late-Step-Semantics.Input)
ultimately show ?goal by (simp add: name-swap)
qed
moreover have $(([(x, y)] \cdot P)[y::=u],([(x, y)] \cdot Q)[y::=u]) \in \operatorname{Rel}$
proof -
from $\operatorname{PRelQ}$ have $(P[x::=u], Q[x::=u]) \in$ Rel by auto
with $\langle y \sharp P\rangle\langle y \sharp Q\rangle$ show ?thesis by (simp add: renaming)
qed
ultimately show $\exists P^{\prime} . a<x>. P \Longrightarrow_{l} u$ in $([(x, y)] \cdot P) \rightarrow a<y>\prec P^{\prime} \wedge\left(P^{\prime}\right.$,
$([(x, y)] \cdot Q)[y::=u]) \in \operatorname{Rel}$
by blast
qed

```
```

            thus \exists\mp@subsup{P}{}{\prime\prime}.\forallu.\exists\mp@subsup{P}{}{\prime}.a<x>.P\Longrightarrow\mp@subsup{\Longrightarrow}{l}{}u\mathrm{ in }\mp@subsup{P}{}{\prime\prime}->a<y>< < P'^( (P',([(x,y)] \cdot
    Q)[y::=u]) \in Rel by blast
qed
next
case(Free Q' \alpha)
have }a<x>.Q\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
hence False by auto
thus ?case by simp
qed
qed
lemma outputPres:
fixes }P\mathrm{ :: pi
and }Q :: p
and a :: name
and b :: name
and Rel :: (pi\times pi) set
and Rel':: (pi }\times pi) se
assumes PRelQ: (P,Q)\in Rel
shows }a{b}.P\rightsquigarrow^<Rel>a{b}.
proof(induct rule: simCases)
case(Bound Q' c x)
have }a{b}.Q\longmapstoc<\nux>\prec\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
hence False by auto
thus ?case by simp
next
case(Input Q' c x)
have }a{b}.Q\longmapstoc<x>\prec\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
hence False by auto
thus ?case by simp
next
case(Free Q' \alpha)
have }a{b}.Q\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
thus \exists\mp@subsup{P}{}{\prime}.a{b}.P\Longrightarrow`` }\alpha\prec\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mathrm{ Rel using PRelQ
proof(induct rule: outputCases, auto simp add: pi.inject residual.inject)
have }a{b}.P\Longrightarrow\mp@subsup{\Longrightarrow}{l}{`}a[b]\precP by(rule Output
moreover assume (P, Q')\in Rel
ultimately show }\exists\mp@subsup{P}{}{\prime}.a{b}.P\Longrightarrow\mp@subsup{\Longrightarrow}{l}{\wedge}a[b]\prec\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mathrm{ Rel by blast
qed
qed
lemma matchPres:
fixes P :: pi
and }Q :: p
and a :: name
and b :: name

```
```

    and Rel :: (pi\times pi) set
    and Rel':: (pi × pi) set
    assumes PSimQ: P\rightsquigarrow^< Rel>Q
    and RelStay: \PQa. }(P,Q)\in\operatorname{Rel}\Longrightarrow([a\frowna]P,Q)\in\operatorname{Rel
    and RelRel': Rel }\subseteqRel'
    shows [a\frownb]P\rightsquigarrow` < Rel'> [a\frownb]Q
    proof(induct rule: simCases)
case(Bound Q' c x)
have }x\sharp[a\frownb]P\mathrm{ by fact
hence xFreshP:(x::name) \sharpP by simp
have [a\frownb]Q\longmapstoc<\nux>}\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
thus ?case
proof(induct rule: matchCases)
case cMatch
have }Q\longmapstoc<\nux>\prec\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
with PSimQ xFreshP obtain P' where PTrans: P\Longrightarrow```}c<\nux><\prec P
and P}\mp@subsup{P}{}{\prime}\mathrm{ RelQ'Q}:(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\operatorname{Rel
by(blast dest: simE)
from PTrans have [a\frowna]P\Longrightarrow\^`|}c<\nux><\mp@subsup{P}{}{\prime}\mathbf{by}(rule Weak-Late-Semantics.Match
with P'RelQ' RelRel' show ?case by blast
qed
next
case(Input Q' c x)
have }x\sharp[a\frownb]P\mathrm{ by fact
hence xFreshP: x\sharpP by simp
have [a\frownb]Q\longmapstoc<x>\prec\prec '' by fact
thus ?case
proof(induct rule: matchCases)
case cMatch
have Q\longmapstoc<x>\prec Q' by fact
with PSimQ xFreshP obtain P'\prime}\mathrm{ where L1: }\forallu.\exists\mp@subsup{P}{}{\prime}.P\Longrightarrow\mp@subsup{\Longrightarrow}{l}{}u in 埥->c<x
\prec P'^( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime}[x::=u])\in\operatorname{Rel
by(force intro: simE)
have }\forallu.\exists\mp@subsup{P}{}{\prime}.[a\frowna]P\mp@subsup{\Longrightarrow}{l}{
proof(rule allI)
fix u
from L1 obtain P' where PTrans: P \Longrightarrow>lu in }\mp@subsup{P}{}{\prime\prime}->c<x>\prec 在' and P'RelQ'
( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime}[x::=u])\in\operatorname{Rel
by blast
from PTrans have [a\frowna]P\Longrightarrow\ < in P'\prime->c<x> \prec P'by(rule Weak-Late-Step-Semantics.Match)
with P'RelQ' RelRel' show }\exists\mp@subsup{P}{}{\prime}.[a\frowna]P\Longrightarrow\mp@subsup{}{l}{
Q}[x::=u])\inRe\mp@subsup{l}{}{\prime
by blast
qed
thus ?case by blast
qed
next

```
```

case(Free Q' \alpha)

```

```

thus ?case
proof(induct rule: matchCases)
case cMatch
have }Q\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
with PSimQ obtain P' where PTrans: P\Longrightarrow吕 \alpha}\prec\mp@subsup{P}{}{\prime}\mathrm{ and PRel: ( ( '', Q') 氏
Rel
by(blast dest: simE)
from PTrans show ?case
proof(induct rule: transitionCases)
case Step
have }P\Longrightarrow\mp@subsup{}{l}{}\alpha\prec\mp@subsup{P}{}{\prime}\mathrm{ by fact
hence [a\frowna]P\Longrightarrowl}\mp@subsup{l}{l}{}\alpha\prec\mp@subsup{P}{}{\prime}\mathbf{by}(\mathrm{ rule Weak-Late-Step-Semantics.Match)
with PRel RelRel' show ?case by(force simp add: weakTransition-def)
next
case Stay
have }\alpha\prec\mp@subsup{P}{}{\prime}=\tau\precP\mathrm{ by fact
hence alphaEqTau: }\alpha=\tau\mathrm{ and PeqP': P= P' by(simp add: residual.inject)+
have [a\frowna]P\Longrightarrow``}\tau<[a\frowna]P by(simp add:weakTransition-def
moreover from PeqP' PRel have ([a\frowna]P, Q')\in Rel by(blast intro: RelStay)
ultimately show ?case using RelRel' alphaEqTau by blast
qed
qed
qed
lemma mismatchPres:
fixes P :: pi
and }Q :: p
and a :: name
and b :: name
and Rel :: (pi\timespi) set
and Rel'::(pi \times pi) set
assumes PSimQ: P\rightsquigarrow` <Rel>Q
and RelStay: \bigwedgePQ a b. \llbracket(P,Q)\inRel; a\not=b\rrbracket\Longrightarrow([a\not=b]P,Q)\inRel
and RelRel':Rel \subseteqRel'
shows [a\not=b]P\rightsquigarrow^<Rel'> [a\not=b]Q
proof(cases a=b)
assume a=b
thus ?thesis by(auto simp add: weakSimulation-def)
next
assume aineqb: }a\not=
show ?thesis
proof(induct rule: simCases)
case(Bound Q' c x)
have }x\sharp[a\not=b]P\mathrm{ by fact
hence xFreshP:(x::name) \#P by simp

```
```

    have \([a \neq b] Q \longmapsto c<\nu x>\prec Q^{\prime}\) by fact
    thus? case
    proof (induct rule: mismatchCases)
    case cMismatch
    have \(Q \longmapsto c<\nu x>\prec Q^{\prime}\) by fact
    with PSimQ xFreshP obtain \(P^{\prime}\) where PTrans: \(P \Longrightarrow \Longrightarrow_{l} c<\nu x>\prec P^{\prime}\)
        and \(P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}\)
        by (blast dest: simE)
    from PTrans aineqb have \([a \neq b] P \Longrightarrow \hat{\imath} c<\nu x>\prec P^{\prime} \mathbf{b y}\) (rule Weak-Late-Semantics.Mismatch)
        with \(P^{\prime}\) RelQ' RelRel' show ? case by blast
    qed
    next
case (Input $Q^{\prime} c x$ )
have $x \sharp[a \neq b] P$ by fact
hence $x$ Fresh $P: x \sharp P$ by simp
have $[a \neq b] Q \longmapsto c<x>\prec Q^{\prime}$ by fact
thus ?case
proof (induct rule: mismatchCases)
case $c$ Mismatch
have $Q \longmapsto c<x>\prec Q^{\prime}$ by fact
with PSim $Q x$ Fresh $P$ obtain $P^{\prime \prime}$ where L1: $\forall u . \exists P^{\prime} . P \Longrightarrow{ }_{l} u$ in $P^{\prime \prime} \rightarrow c<x>$
$\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}[x::=u]\right) \in \operatorname{Rel}$
by (force intro: $\operatorname{simE}$ )
have $\forall u$. $\exists P^{\prime} .[a \neq b] P \Longrightarrow_{l} u$ in $P^{\prime \prime} \rightarrow c<x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}[x::=u]\right) \in \operatorname{Rel}^{\prime}$
proof(rule allI)
fix $u$
from L1 obtain $P^{\prime}$ where PTrans: $P \Longrightarrow_{l} u$ in $P^{\prime \prime} \rightarrow c<x>\prec P^{\prime}$ and
$P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}[x::=u]\right) \in \operatorname{Rel}$
by blast
from PTrans aineqb have $[a \neq b] P \Longrightarrow_{l} u$ in $P^{\prime \prime} \rightarrow c<x>\prec P^{\prime}$ by (rule
Weak-Late-Step-Semantics.Mismatch)
with $P^{\prime}$ RelQ ${ }^{\prime}$ RelRel' show $\exists P^{\prime} .[a \neq b] P \Longrightarrow_{l} u$ in $P^{\prime \prime} \rightarrow c<x>\prec P^{\prime} \wedge\left(P^{\prime}\right.$,
$\left.Q^{\prime}[x::=u]\right) \in \operatorname{Rel}^{\prime}$
by blast
qed
thus ?case by blast
qed
next
case (Free $\left.Q^{\prime} \alpha\right)$
have $[a \neq b] Q \longmapsto \alpha \prec Q^{\prime}$ by fact
thus ?case
proof (induct rule: mismatchCases)
case cMismatch
have $a \neq b$ by fact
have $Q \longmapsto \alpha \prec Q^{\prime}$ by fact
with PSimQ obtain $P^{\prime}$ where PTrans: $P \Longrightarrow \hat{l}^{\hat{a}} \alpha \prec P^{\prime}$ and PRel: $\left(P^{\prime}, Q^{\prime}\right)$
$\in$ Rel
by (blast dest: $\operatorname{simE})$
from PTrans show ?case

```
```

        proof(induct rule: transitionCases)
            case Step
            have }P\Longrightarrow\mp@subsup{\Longrightarrow}{l}{}\alpha\prec\mp@subsup{P}{}{\prime}\mathrm{ by fact
            hence [a\not=b]P\Longrightarrow\Longrightarrowl}\alpha\prec\mp@subsup{P}{}{\prime}\mathrm{ using }\langlea\not=b\rangle\mathrm{ by(rule Weak-Late-Step-Semantics.Mismatch)
            with PRel RelRel' show ?case by(force simp add: weakTransition-def)
                    next
            case Stay
            have }\alpha\prec\mp@subsup{P}{}{\prime}=\tau\precP\mathrm{ by fact
            hence alphaEqTau: \alpha=\tau and PeqP': P= P' by(simp add: residual.inject)+
            have [a\not=b]P\Longrightarrow\mp@subsup{}{l}{`}\tau\prec[a\not=b]P by(simp add:weakTransition-def)
            moreover from PeqP' PRel aineqb have ([a\not=b]P, Q') \in Rel by(blast intro:
    RelStay)
ultimately show ?case using alphaEqTau RelRel' by blast
qed
qed
qed
qed
lemma parCompose:
fixes P :: pi
and }Q\quad::p
and }R\quad::p
and T :: pi
and Rel :: (pi\timespi) set
and Rel' :: (pi\timespi) set
and Rel" :: (pi\timespi) set
assumes PSimQ: P}\rightsquigarrow^<<Rel>
and RSimT: R}<<<\mp@subsup{Rel}{}{\prime}>
and PRelQ: }(P,Q)\in\operatorname{Rel
and RRel'T: (R,T)\inRel'
and Par: }\PQRT.\llbracket(P,Q)\in\operatorname{Rel};(R,T)\in\operatorname{Rel}\rrbracket\Longrightarrow(P|R,Q|T
Rel"

```

```

    and EqvtRel: equt Rel
    and EqvtRel': eqvt Rel'
    and EqvtRel"':eqvt Rel'"
    shows P|R\rightsquigarrow^<Rel'>}>||
    using <equt Rel'>
proof}(\mathrm{ induct rule: simCasesCont [where C=(P,Q,R,T)])
case(Bound Q' a x)
from <x\sharp (P,Q,R,T)\rangle have }x\sharpP\mathrm{ and }x\sharpR\mathrm{ and }x\sharpQ\mathrm{ and }x\sharpT\mathrm{ by simp+
from <Q|T\longmapsto |}a<\nux>\prec\mp@subsup{Q}{}{\prime}\rangle\langlex\sharpQ\rangle\langlex\sharpT
show ?case
proof(induct rule: parCasesB)
case(cPar1 Q')
from PSimQ<Q\longmapstoa<\nux>>\prec Q'〉\langlex\sharpP> obtain P' where PTrans:P\Longrightarrowl}\mp@subsup{}{l}{`
a<\nux>}\prec\mp@subsup{P}{}{\prime

```
```

                                    and }\mp@subsup{P}{}{\prime}\operatorname{RelQ}\mp@subsup{Q}{}{\prime}:(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\operatorname{Rel
    by(blast dest: simE)
    ```

```

    moreover from P'RelQ' RRel'T have ( }\mp@subsup{P}{}{\prime}|R,\mp@subsup{Q}{}{\prime}|T)\inRe\mp@subsup{l}{}{\prime\prime}\mathrm{ by(rule Par)
    ultimately show ?case by blast
    next
case(cPar2 T')
from RSimT <T\longmapsto
a<\nux>}\prec\mp@subsup{R}{}{\prime
and R'Rel'T}\mp@subsup{T}{}{\prime}:(\mp@subsup{R}{}{\prime},\mp@subsup{T}{}{\prime})\inRe\mp@subsup{R}{}{\prime
by(blast dest: simE)

```

```

        by(blast intro:Weak-Late-Semantics.Par2B)
    moreover from PRelQ R'Rel'T}\mp@subsup{T}{}{\prime}\mathrm{ have ( }P|\mp@subsup{R}{}{\prime},Q| |')\inRe\mp@subsup{l}{}{\prime\prime}\mathbf{by}(rule Par
    ultimately show ?case by blast
    qed
    next
case(Input Q' a x)
from \langlex\sharp(P,Q,R,T)\rangle have }x\sharpP\mathrm{ and }x\sharpR\mathrm{ and }x\sharpQ\mathrm{ and }x\sharpT\mathrm{ by simp+
from〈Q|T\longmapsto \
show ?case
proof(induct rule: parCasesB)
case(cPar1 Q')

```

```

            where L1: }\forallu.\exists\mp@subsup{P}{}{\prime}.P\Longrightarrow\mp@subsup{\Longrightarrow}{l}{}u\mathrm{ in }\mp@subsup{P}{}{\prime\prime}->a<x>< \prec P'^( (P', Q'[x::=u])\in Re
            by(blast dest: simE)
    have }\forallu.\exists\mp@subsup{P}{}{\prime}.P|R\Longrightarrow\mp@subsup{}{l}{}u\mathrm{ in }(\mp@subsup{P}{}{\prime\prime}|R)->a<x>\prec\prec P'^( (P', Q'[x::=u]
    T[x::=u]) \inRel'
proof(rule allI)
fix u
from L1 obtain P' where PTrans:P \Longrightarrow>}\mp@subsup{}{l}{}u\mathrm{ in }\mp@subsup{P}{}{\prime\prime}->a<x>\prec \prec P
and }\mp@subsup{P}{}{\prime}Rel\mp@subsup{Q}{}{\prime}:(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime}[x::=u])\in\mathrm{ Rel by blast
from PTrans <x\sharpR` have P|R \Longrightarrow\Longrightarrowlu in ( }\mp@subsup{P}{}{\prime\prime}|R)->a<x>\prec ( (P'|R
by(rule Weak-Late-Step-Semantics.Par1B)
moreover from P'RelQ' RRel'T have ( }\mp@subsup{P}{}{\prime}||,\mp@subsup{Q}{}{\prime}[x::=u]|T)\inRel '|
by(rule Par)
ultimately show }\exists\mp@subsup{P}{}{\prime}.P|R\mp@subsup{\Longrightarrow}{l}{

```

```

            by(force simp add: forget)
    qed
    thus ?case by force
    next
    case(cPar2 T')
    from RSimT <T\longmapstoa<x> \prec T'\rangle\langlex\sharpR\rangle obtain R"
    ```

```

        by(blast dest: simE)
        have }\forallu.\exists\mp@subsup{P}{}{\prime}.P|R\mp@subsup{\Longrightarrow}{l}{}u\mathrm{ in }(P|\mp@subsup{R}{}{\prime\prime})->a<x><< P'^( (P',Q[x::=u]
    T'[x::=u]) \in Rel'
proof(rule allI)

```
fix \(u\)
from L1 obtain \(R^{\prime}\) where RTrans: \(R \Longrightarrow_{l} u\) in \(R^{\prime \prime} \rightarrow a<x>\prec R^{\prime}\) and \(R^{\prime} \operatorname{Rel}^{\prime} T^{\prime}:\left(R^{\prime}, T^{\prime}[x::=u]\right) \in\) Rel \(^{\prime}\) by blast
from RTrans \(\langle x \sharp P\rangle\) have ParTrans: \(P \| R \Longrightarrow_{l} u\) in \(\left.\left(P \| R^{\prime \prime}\right) \rightarrow a<x\right\rangle \prec\) \(\left(P \| R^{\prime}\right)\) by(rule Weak-Late-Step-Semantics.Par2B)
moreover from \(P \operatorname{Rel} Q R^{\prime} \operatorname{Rel}^{\prime} T^{\prime}\) have \(\left(P\left\|R^{\prime}, Q\right\| \quad T^{\prime}[x::=u]\right) \in \operatorname{Rel}^{\prime \prime}\) by (rule Par)
ultimately show \(\exists P^{\prime} . P \| R \Longrightarrow_{l} u\) in \(\left(P \| R^{\prime \prime}\right) \rightarrow a<x>\prec P^{\prime} \wedge\)
\(\left(P^{\prime}, Q[x::=u] \| T^{\prime}[x::=u]\right) \in\) Rel \(^{\prime \prime}\) using \(\langle x \sharp Q\rangle\)
by (force simp add: forget)
qed
thus ?case by force
qed
next
case (Free \(Q T^{\prime} \alpha\) )
have \(Q \| T \longmapsto \alpha \prec Q T^{\prime}\) by fact
thus ?case
\(\operatorname{proof}(\) induct rule: parCasesF\([\) of \(\cdots(P, R)])\)
case( \(c\) Par1 \(Q^{\prime}\) )
have \(Q \longmapsto \alpha \prec Q^{\prime}\) by fact
with \(P \operatorname{Sim} Q\) obtain \(P^{\prime}\) where PTrans: \(P \Longrightarrow{ }_{l}^{\wedge} \alpha \prec P^{\prime}\) and PRel: \(\left(P^{\prime}, Q^{\prime}\right)\)
\(\in\) Rel by (blast dest: simE)
from PTrans have Trans: \(P\left\|R \Longrightarrow \hat{l}{ }_{l} \alpha \prec P^{\prime}\right\| R\) by(rule Weak-Late-Semantics.Par1F)
moreover from PRel RRel'T have \(\left(P^{\prime}\left\|R, Q^{\prime}\right\| T\right) \in \operatorname{Rel}^{\prime \prime}\) by(blast intro:
Par)
ultimately show? case by blast
next
case(cPar2 \(T^{\prime}\) )
have \(T \longmapsto \alpha \prec T^{\prime}\) by fact
with \(R \operatorname{Sim} T\) obtain \(R^{\prime}\) where RTrans: \(R \Longrightarrow{ }_{l}^{\wedge} \alpha \prec R^{\prime}\) and RRel: \(\left(R^{\prime}, T^{\prime}\right)\)
\(\in R e l^{\prime}\)
by (blast dest: \(\operatorname{simE}\) )
from RTrans have Trans: \(P\left\|R \Longrightarrow \Longrightarrow_{l} \alpha \prec P\right\| R^{\prime}\) by (rule Weak-Late-Semantics.Par2F)
moreover from PRelQ RRel have \(\left(P\left\|R^{\prime}, Q\right\| T^{\prime}\right) \in\) Rel \(^{\prime \prime}\) by (blast intro:
Par)
ultimately show ?case by blast
next
case (cComm1 \(Q^{\prime} T^{\prime}\) abx)
have \(Q\) Trans: \(Q \longmapsto a<x>\prec Q^{\prime}\) and TTrans: \(T \longmapsto a[b] \prec T^{\prime}\) by fact +
have \(x \sharp(P, R)\) by fact
hence \(x\) Fresh \(P: x \sharp P\) by (simp add: fresh-prod)
from PSimQ QTrans xFreshP obtain \(P^{\prime} P^{\prime \prime}\) where PTrans: \(P \Longrightarrow_{l} b\) in \(P^{\prime \prime} \rightarrow a<x>\prec P^{\prime}\) and \(P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}[x::=b]\right) \in \operatorname{Rel}\)

> by (blast dest: simE)
from \(R\) Sim T TTrans obtain \(R^{\prime}\) where \(R\) Trans: \(R \Longrightarrow \hat{l^{\prime}} a[b] \prec R^{\prime}\) and RRel: \(\left(R^{\prime}, T^{\prime}\right) \in\) Rel \(^{\prime}\)
by (blast dest: simE)
from PTrans RTrans have \(P\left\|R \Longrightarrow \hat{\imath} \tau \prec P^{\prime}\right\| R^{\prime} \mathbf{b y}\) (rule Weak-Late-Semantics.Comm1)
moreover from \(P^{\prime} \operatorname{Rel} Q^{\prime} R R e l\) have \(\left(P^{\prime}\left\|R^{\prime}, Q^{\prime}[x::=b]\right\| T^{\prime}\right) \in \operatorname{Rel}^{\prime \prime}\) by (rule
Par)
ultimately show ?case by blast
next
case (cComm2 \(\left.Q^{\prime} T^{\prime} a b x\right)\)
have \(Q\) Trans: \(Q \longmapsto a[b] \prec Q^{\prime}\) and TTrans: \(T \longmapsto a<x>\prec T^{\prime}\) by fact +
have \(x \sharp(P, R)\) by fact
hence \(x\) Fresh \(R\) : \(x \sharp R\) by (simp add: fresh-prod)

and PRel: \(\left(P^{\prime}, Q^{\prime}\right) \in\) Rel
by (blast dest: simE)
from \(R\) Sim T TTrans xFresh \(R\) obtain \(R^{\prime} R^{\prime \prime}\) where \(R T r a n s: ~ R \Longrightarrow_{l} b\) in \(R^{\prime \prime} \rightarrow a<x>\prec R^{\prime}\)
and \(R^{\prime} \operatorname{Rel}^{\prime} T^{\prime}:\left(R^{\prime}, T^{\prime}[x::=b]\right) \in \operatorname{Rel}^{\prime}\)
by (blast dest: simE)
from PTrans RTrans have \(P\left\|R \Longrightarrow \Longrightarrow_{i}^{\wedge} \tau \prec P^{\prime}\right\| R^{\prime} \mathbf{b y}\) (rule Weak-Late-Semantics.Comm2)
moreover from PRel \(R^{\prime} \operatorname{Rel}^{\prime} T^{\prime}\) have \(\left(P^{\prime}\left\|R^{\prime}, Q^{\prime}\right\| T^{\prime}[x::=b]\right) \in \operatorname{Rel}^{\prime \prime}\) by (rule Par)
ultimately show ?case by blast
next
case (cClose1 \(Q^{\prime} T^{\prime}\) a \(x\) y)
have \(Q\) Trans: \(Q \longmapsto a<x>\prec Q^{\prime}\) and TTrans: \(T \longmapsto a<\nu y>\prec T^{\prime}\) by fact +
have \(x \sharp(P, R)\) and \(y \sharp(P, R)\) by fact +
hence \(x\) Fresh \(P: x \sharp P\) and \(y\) Fresh \(R\) : \(y \sharp R\) and \(y\) Fresh \(P: y \sharp P\) by (simp add: fresh-prod)+
from PSimQ QTrans xFreshP obtain \(P^{\prime} P^{\prime \prime}\) where PTrans: \(P \Longrightarrow_{l y}\) in \(P^{\prime \prime} \rightarrow a<x>\prec P^{\prime}\)
and \(P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}[x::=y]\right) \in \operatorname{Rel}\)
by (blast dest: simE)
from \(R\) Sim \(T\) TTrans yFresh \(R\) obtain \(R^{\prime}\) where \(R T\) Trans: \(R \Longrightarrow \hat{l}\) a \(a<\nu y>\prec R^{\prime}\)
\[
\text { and } R^{\prime} R e l^{\prime} T^{\prime}:\left(R^{\prime}, T^{\prime}\right) \in R e l^{\prime}
\]
by (blast dest: simE)
from PTrans RTrans yFreshP yFreshR have Trans: \(P \| R \Longrightarrow_{l} \hat{\imath} \tau \prec<\nu y>\left(P^{\prime}\right.\) \(\| R^{\prime}\) )
by(rule Weak-Late-Semantics.Close1)
```

    moreover from \(P^{\prime} \operatorname{Rel} Q^{\prime} R^{\prime} \operatorname{Rel}^{\prime} T^{\prime}\) have \(\left(<\nu y>\left(P^{\prime} \| R^{\prime}\right),<\nu y>\left(Q^{\prime}[x::=y] \|\right.\right.\)
    $\left.\left.T^{\prime}\right)\right) \in R e l^{\prime \prime}$
by (blast intro: Par Res)
ultimately show ?case by blast
next
case (cClose2 $Q^{\prime} T^{\prime}$ a $x$ y)
have $Q$ Trans: $Q \longmapsto a<\nu y>\prec Q^{\prime}$ and TTrans: $T \longmapsto a<x>\prec T^{\prime}$ by fact +
have $x \sharp(P, R)$ and $y \sharp(P, R)$ by fact +
hence $x$ Fresh $R$ : $x \sharp R$ and $y$ Fresh $P: y \sharp P$ and $y$ Fresh $R$ : $y \sharp R$ by (simp add:
fresh-prod)+
from PSimQ QTrans yFreshP obtain $P^{\prime}$ where PTrans: $P \Longrightarrow{ }_{l}$ ^ $a<\nu y>\prec P^{\prime}$
and $P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}$
by(blast dest: simE)
from RSimT TTrans xFreshR obtain $R^{\prime} R^{\prime \prime}$ where $R$ Trans: $R \Longrightarrow_{l} y$ in
$R^{\prime \prime} \rightarrow a<x>\prec R^{\prime}$
and $R^{\prime} \operatorname{Rel}^{\prime} T^{\prime}:\left(R^{\prime}, T^{\prime}[x::=y]\right) \in \operatorname{Rel}^{\prime}$
$\mathbf{b y}($ blast dest: $\operatorname{sim} E)$
from PTrans RTrans yFreshP yFreshR have Trans: $P \| R \Longrightarrow{ }_{l} \hat{\tau} \tau \prec<\nu y>\left(P^{\prime}\right.$
$\| R^{\prime}$ )
by(rule Weak-Late-Semantics.Close2)
moreover from $P^{\prime} \operatorname{Rel} Q^{\prime} R^{\prime} \operatorname{Rel}^{\prime} T^{\prime}$ have $\left(<\nu y>\left(P^{\prime} \| R^{\prime}\right),<\nu y>\left(Q^{\prime} \| T^{\prime}[x::=y]\right)\right)$
$\in R e l^{\prime \prime}$
by(blast intro: Par Res)
ultimately show ?case by blast
qed
qed
lemma parPres:
fixes $P$ :: $p i$
and $\quad Q \quad:: p i$
and $\quad R$ :: pi
and $a$ :: name
and $b$ :: name
and Rel $::(p i \times p i)$ set
and Rel' $::(p i \times p i)$ set
assumes $\operatorname{PSimQ}: \quad P \rightsquigarrow \wedge<\operatorname{Rel}>Q$
and PRelQ: $\quad(P, Q) \in$ Rel
and Par: $\quad \bigwedge P Q R .(P, Q) \in \operatorname{Rel} \Longrightarrow(P\|R, Q\| R) \in \operatorname{Rel}^{\prime}$
and Res: $\bigwedge P Q a .(P, Q) \in \operatorname{Rel}^{\prime} \Longrightarrow(<\nu a>P,<\nu a>Q) \in R^{\prime} l^{\prime}$
and EqvtRel: equt Rel
and EqvtRel': eqvt Rel ${ }^{\prime}$
shows $P \| R \rightsquigarrow \wedge<$ Rel $^{\prime}>Q \| R$
proof -
note PSimQ

```
```

    moreover have RSimR: R\leadsto^ <Id> R by(auto intro: reflexive)
    moreover note PRelQ moreover have (R,R)\inId by auto
    moreover from Par have }\PQRT.\llbracket(P,Q)\in\operatorname{Rel;}(R,T)\inId\rrbracket\Longrightarrow(P
    R,Q|T)\inRel'
by auto
moreover note Res <eqvt Rel>
moreover have eqvt Id by(auto simp add: eqvt-def)
ultimately show ?thesis using EqvtRel' by(rule parCompose)
qed
lemma resPres:
fixes P :: pi
and Q :: pi
and Rel :: (pi }\times pi) se
and x :: name
and Rel'::(pi\times pi) set
assumes PSimQ: P\rightsquigarrow^<Rel> Q
and ResRel: $P::pi) (Q::pi) (x::name). (P,Q)\inRel \Longrightarrow(<\nux>P,<\nux>Q )
Rel'
    and RelRel':Rel }\subseteqRel'
    and EqvtRel: eqvt Rel
    and EqvtRel': eqvt Rel'
    shows <\nux>P \rightsquigarrow`<Rel'}><\nux>
proof -
    from EqvtRel' show ?thesis
    proof(induct rule: simCasesCont[of - (P,Q,x)])
        case(Bound Q' a y)
    have Trans: <\nux>>Q\longmapstoa<\nuy>}\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    have }y\sharp(P,Q,x)\mathrm{ by fact
    hence yineqx: y fx and yFreshP: y }#P\mathrm{ and }y\sharpQ\mathrm{ by(simp add: fresh-prod)+
    from Trans }\langley\not=x\rangle\langley\sharpQ\rangle\mathrm{ show ?case
    proof(induct rule: resCasesB)
        case(cOpen a Q')
        have QTrans: }Q\longmapstoa[x]\prec\mp@subsup{Q}{}{\prime}\mathrm{ and aineqx: }a\not=x\mathrm{ by fact+
        from PSimQ QTrans obtain P' where PTrans: P\Longrightarrow`` }a[x]\prec\mp@subsup{P}{}{\prime
                            and P'RelQ':( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\operatorname{Rel
            by(blast dest: simE)
        have }<\nux>P\Longrightarrow\^`| a<\nuy>< ([(y,x)]\cdot P'
        proof -
    from PTrans aineqx have <\nux>P\Longrightarrow\Longrightarrow\}a<\nux> \prec P'\mathbf{by}(rule Weak-Late-Semantics.Open)
        moreover from PTrans yFreshP have y }#\mp@subsup{P}{}{\prime}\mathbf{by}(force intro: freshTransition)
            ultimately show ?thesis by(simp add: alphaBoundResidual name-swap)
        qed
    moreover from EqvtRel P'RelQ' RelRel' have ([(y,x)] • P', [(y,x)] \cdot Q')\in
Rel'
```
```
                by(blast intro: eqvtRelI)
            ultimately show ?case by blast
    next
            case(cRes Q')
            have QTrans: Q\longmapstoa<\nuy> \prec Q' by fact
            from <x BoundOutputS a> have }x\not=a\mathrm{ by simp
            from PSimQ yFreshP QTrans obtain P' where PTrans: P\Longrightarrow\Longrightarrow^^ a<\nuy>}
P'
                                    and P}\mp@subsup{P}{}{\prime}\operatorname{RelQ}\mp@subsup{Q}{}{\prime}:(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\operatorname{Rel
            by(blast dest: simE)
            from PTrans }\langlex\not=a\rangle yineqx yFreshP have ResTrans: <\nux>P\Longrightarrow\Longrightarrow\hat{l}a<\nuy
\prec(<\nux>P')
            by(blast intro:Weak-Late-Semantics.ResB)
            moreover from P'RelQ' have ((<\nux>> '名),(<\nux>>\mp@subsup{Q}{}{\prime}))\inRe\mp@subsup{l}{}{\prime}
                    by(rule ResRel)
            ultimately show ?case by blast
    qed
next
    case(Input Q' a y)
    have }y\sharp(P,Q,x)\mathrm{ by fact
    hence yineqx: y\not=x and yFreshP: y\sharpP and y\sharpQ by(simp add: fresh-prod)+
    have <\nux>Q \longmapstoa<y> \prec Q' by fact
    thus ?case using yineqx 〈y#Q>
    proof(induct rule: resCasesB)
    case(cOpen a Q )
    thus ?case by simp
    next
        case(cRes Q')
        have QTrans: Q\longmapstoa<y> \prec Q' by fact
        from〈x\sharp InputS a〉 have x\not=a by simp
    from PSimQ QTrans yFreshP obtain P'
            where L1: }\forallu.\exists\mp@subsup{P}{}{\prime}.P\Longrightarrow\mp@subsup{}{l}{
            by(blast dest: simE)
    have }\forallu.\exists\mp@subsup{P}{}{\prime}.<\nux>P\Longrightarrow\Longrightarrow\mp@subsup{}{l}{}u\mathrm{ in }(<\nux>\mp@subsup{P}{}{\prime\prime})->a<y>\prec\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},(<\nux>\mp@subsup{Q}{}{\prime})[y::=u]
Rel'
    proof(rule allI)
            fix }
        show }\exists\mp@subsup{P}{}{\prime}.<\nux>P\Longrightarrow\mp@subsup{\Longrightarrow}{l}{}u\mathrm{ in }<\nux>\mp@subsup{P}{}{\prime\prime}->a<y>\prec\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},(<\nux>\mp@subsup{Q}{}{\prime})[y::=u]
ERel'
            proof(cases x=u)
            assume xequ: x=u
            have \existsc::name. c\sharp(P, P',},\mp@subsup{Q}{}{\prime},x,y,a) by(blast intro: name-exists-fresh
            then obtain c::name where cFreshP:c\sharpP and cFresh\mp@subsup{P}{}{\prime\prime}:c\sharpP\mp@subsup{P}{}{\prime\prime}\mathrm{ and}
cFresh Q': c\sharp Q'
                                    and cineqx:c}c=x\mathrm{ and cineqy: c}\not=y\mathrm{ and cineqa: c }\not=
```
by (force simp add: fresh-prod)
from L1 obtain \(P^{\prime}$ where PTrans: $P \Longrightarrow_{l} c$ in $P^{\prime \prime} \rightarrow a<y>\prec P^{\prime}$

$$
\text { and } P^{\prime} \operatorname{Re} l Q^{\prime}:\left(P^{\prime}, Q^{\prime}[y::=c]\right) \in \operatorname{Rel}
$$

by blast
have $<\nu x>P \Longrightarrow{ }_{l} u$ in $\left(<\nu x>P^{\prime \prime}\right) \rightarrow a<y>\prec<\nu c>\left([(x, c)] \cdot P^{\prime}\right)$
proof -
from PTrans yineqx $\langle x \neq a\rangle$ cineqx have $<\nu x>P \Longrightarrow_{l} c$ in $\left(<\nu x>P^{\prime \prime}\right) \rightarrow a<y>$ $\prec<\nu x>P^{\prime}$
by (blast intro: Weak-Late-Step-Semantics.ResB)
hence $([(x, c)] \cdot<\nu x>P) \Longrightarrow_{l}([(x, c)] \cdot c)$ in $\left([(x, c)] \cdot<\nu x>P^{\prime \prime}\right) \rightarrow([(x$, $c)] \cdot a)<([(x, c)] \cdot y)>\prec[(x, c)] \cdot<\nu x>P^{\prime}$
by(rule Weak-Late-Step-Semantics.eqvtI)
moreover from $c$ Fresh $P$ have $\langle\nu c>([(x, c)] \cdot P)=<\nu x>P$ by $(\operatorname{simp}$ add: alphaRes)
moreover from $c F r e s h P^{\prime \prime}$ have $<\nu c>\left([(x, c)] \cdot P^{\prime \prime}\right)=<\nu x>P^{\prime \prime}$ by $($ simp add: alphaRes)
ultimately show ?thesis using $\langle x \neq a\rangle$ cineqa yineqx cineqy cineqx xequ by (simp add: name-calc)
qed
moreover have $\left.(<\nu c\rangle\left([(x, c)] \cdot P^{\prime}\right),\left(\langle\nu x\rangle Q^{\prime}\right)[y::=u]\right) \in \operatorname{Rel}^{\prime}$
proof -
from $P^{\prime} \operatorname{Rel} Q^{\prime}$ have $\left(<\nu x>P^{\prime},<\nu x>\left(Q^{\prime}[y::=c]\right)\right) \in$ Rel ${ }^{\prime}$ by (rule ResRel)
with EqvtRel' have $\left([(x, c)] \cdot<\nu x>P^{\prime},[(x, c)] \cdot<\nu x>\left(Q^{\prime}[y::=c]\right)\right) \in$ Rel' ${ }^{\prime} \mathbf{b y}$ (rule eqvtRelI)
with cineqy yineqx cineqx have $\left(<\nu c>\left([(x, c)] \cdot P^{\prime}\right),(<\nu c\rangle([(x, c)] \cdot\right.$ $\left.\left.\left.Q^{\prime}\right)\right)[y::=x]\right) \in \operatorname{Rel}^{\prime}$ by (simp add: name-calc eqvt-subs)
with $c$ Fresh $Q^{\prime}$ xequ show ?thesis by (simp add: alphaRes)
qed
ultimately show ?thesis by blast
next
assume xinequ: $x \neq u$
from L1 obtain $P^{\prime}$ where PTrans: $P \Longrightarrow_{l} u$ in $P^{\prime \prime} \rightarrow a<y>\prec P^{\prime}$ and $P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}[y::=u]\right) \in$ Rel by blast
from PTrans $\langle x \neq a\rangle$ yineqx xinequ have $<\nu x>P \Longrightarrow{ }_{l} u$ in $\left(<\nu x>P^{\prime \prime}\right) \rightarrow a<y>$ $\prec<\nu x>P^{\prime}$
by (blast intro: Weak-Late-Step-Semantics.ResB)
moreover from $P^{\prime}$ RelQ $Q^{\prime}$ xinequ yineqx have $\left.\left(<\nu x>P^{\prime},(<\nu x\rangle Q^{\prime}\right)[y:=u]\right)$ $\in R e l^{\prime}$
by (force intro: ResRel)
ultimately show ?thesis by blast
qed
qed
thus?case by blast
qed
next
case (Free $\left.Q^{\prime} \alpha\right)$


```
    thus ?case
    proof(induct rule: resCasesF)
    case(cRes Q')
    have }Q\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    with PSimQ obtain P' where PTrans: P\Longrightarrow^^ }\alpha\prec\mp@subsup{P}{}{\prime
                            and P'RelQ':}(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\operatorname{Rel
            by(blast dest: simE)
    have }<\nux>P\Longrightarrow\{\mp@code{|}\alpha\prec<\nux>\mp@subsup{P}{}{\prime
    proof -
            have xFreshAlpha: x }\sharp\alpha\mathrm{ by fact
            with PTrans show ?thesis by(rule ResF)
        qed
        moreover from P'RelQ' have (<\nux>>P', <\nux>>Q') \in Rel' by(rule ResRel)
        ultimately show ?case by blast
        qed
    qed
qed
lemma resChainI:
    fixes P :: pi
    and }Q ::p
    and Rel :: (pi\timespi) set
    and lst :: name list
    assumes eqvtRel: eqvt Rel
    and Res: }\PQa.(P,Q)\in\operatorname{Rel}\Longrightarrow(<\nua>P,<\nua>Q)\inRe
    and PRelQ: P}\rightsquigarrow<\mathrm{ Rel }>
    shows (resChain lst) P\rightsquigarrow^ <Rel> (resChain lst) Q
proof -
    show ?thesis
    proof(induct lst)
        from PRelQ show resChain [] P `^ <Rel> resChain [] Q by simp
    next
        fix a lst
        assume IH:(resChain lst P) \rightsquigarrow^ <Rel> (resChain lst Q)
        moreover from Res have \PQa.(P,Q)\inRel \Longrightarrow(<\nua>P,<\nua>Q)\in
Rel
            by simp
    moreover have Rel \subseteqRel by simp
        ultimately have <\nua>(resChain lst P)\rightsquigarrow^<Rel> <\nua>(resChain lst Q)
using eqvtRel
            by(rule-tac resPres)
            thus resChain (a# lst) P\rightsquigarrow^ <Rel> resChain (a# lst)Q
            by simp
    qed
qed
```

```
lemma bangPres:
    fixes P :: pi
    and }Q\quad::p
    and Rel :: (pi\times pi) set
    assumes PSimQ: }\quadP\rightsquigarrow^<Rel>
    and PRelQ: }(P,Q)\inRe
    and Sim: }\quad\PQ.(P,Q)\in\operatorname{Rel}\LongrightarrowP\rightsquigarrow^<Rel>Q
    and ParComp: }\PQRT.\llbracket(P,Q)\in\operatorname{Rel;}(R,T)\in\mp@subsup{R}{}{\prime
|T)\inRel
    and Res: }\quad\PQx.(P,Q)\in\mp@subsup{R}{RCl}{\prime}\Longrightarrow(<\nux>P,<\nux>Q)\in\mp@subsup{Rel}{}{\prime
```



```
    and BangRelRel':(bangRel Rel) \subseteqRel'
    and eqvtRel': eqvt Rel'
    shows !P `` < Rel'> !Q
proof -
    have }\RsP.\llbracket!Q\longmapstoRs;(P,!Q)\in\mathrm{ bangRel Rel】 # weakSimAct P Rs P Rel
    proof -
        fix Rs P
        assume !Q\longmapstoRs and (P,!Q)\in bangRel Rel
    thus weakSimAct P Rs P Rel'
    proof(nominal-induct avoiding: P rule: bangInduct)
        case(cPar1B aa x Q')
        have QTrans: Q\longmapstoaa«x» \prec Q' and xFreshQ: x #Q by fact+
        have (P,Q|!Q)\in bangRel Rel and }x\sharpP\mathrm{ by fact+
        thus ?case
        proof(induct rule: BRParCases)
            case(BRPar P R)
                have PRelQ:(P,Q)\inRel and RBangRelT: (R,!Q) \in bangRel Rel by
fact+
            have }x\sharpP||\mathrm{ by fact
            hence xFreshP: }x\sharpP\mathrm{ and xFreshR: }x\sharpR\mathrm{ by simp+
            from PRelQ have PSimQ: P}\mp@subsup{\rightsquigarrow}{}{\wedge}<\mathrm{ Rel }>Q by(rule Sim
            from eqvtRel' show ?case
            proof(induct rule: simActBoundCases)
                    case(Input a)
                    have aa = InputS a by fact
                        with PSimQ QTrans xFreshP obtain P'\prime
                        where L1:\forallu.\exists\mp@subsup{P}{}{\prime}.P\Longrightarrow\mp@subsup{\Longrightarrow}{l}{}u\mathrm{ in }\mp@subsup{P}{}{\prime\prime}->a<x>\prec\prec P'^( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime}[x::=u])
Rel
                by(blast dest: simE)
```



```
!Q)[x::=u]) \in Rel'
            proof(rule allI)
                fix }
```

from L1 obtain $P^{\prime}$ where PTrans: $P \Longrightarrow_{l} u$ in $P^{\prime \prime} \rightarrow a<x>\prec P^{\prime}$ and $P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}[x::=u]\right) \in \operatorname{Rel}$
by blast
from PTrans xFreshR have $P \| R \Longrightarrow_{l} u$ in $\left(P^{\prime \prime} \| R\right) \rightarrow a<x>\prec P^{\prime} \| R$ by(rule Weak-Late-Step-Semantics.Par1B)
moreover have $\left(P^{\prime} \| R,\left(Q^{\prime} \|!Q\right)[x::=u]\right) \in$ Rel $^{\prime}$ proof -
from $P^{\prime}$ Rel $Q^{\prime}$ RBangRelT have $\left(P^{\prime}\left\|R, Q^{\prime}[x::=u]\right\|!Q\right) \in$ bangRel
Rel
by (rule Rel.BRPar)
with $x$ Fresh $Q$ BangRelRel' show ?thesis by(auto simp add: forget)
qed
ultimately show $\exists P^{\prime} . P \| R \Longrightarrow_{l} u$ in $\left(P^{\prime \prime} \| R\right) \rightarrow a<x>\prec P^{\prime} \wedge$

$$
\left(P^{\prime},\left(Q^{\prime} \|!Q\right)[x::=u]\right) \in \text { Rel }^{\prime} \text { by blast }
$$

qed
thus? case by blast
next
case(BoundOutput a)
have $a a=$ BoundOutputS $a$ by fact
with PSimQ QTrans xFreshP obtain $P^{\prime}$ where PTrans: $P \Longrightarrow \hat{l^{\prime}} \dot{\hat{a}} a<\nu x>$
$\prec P^{\prime}$
and $P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}$
by (blast dest: $\operatorname{sim} E)$
from PTrans xFresh $R$ have $P\left\|R \Longrightarrow \Longrightarrow_{i}^{\wedge} a<\nu x>\prec P^{\prime}\right\| R$
by(rule Weak-Late-Semantics.Par1B)
moreover from $P^{\prime}$ RelQ' RBangRelT BangRelRel' have $\left(P^{\prime}\left\|R, Q^{\prime}\right\|\right.$
$!Q) \in \operatorname{Rel}^{\prime}$
by(blast intro: Rel.BRPar)
ultimately show ?case by blast
qed
qed
next
case (cPar1F $\left.\alpha Q^{\prime} P\right)$
have $Q$ Trans: $Q \longmapsto \alpha \prec Q^{\prime}$ by fact
have $(P, Q \|!Q) \in$ bangRel Rel by fact
thus ?case
proof (induct rule: BRParCases)
case (BRPar P R)
have PRelQ: $(P, Q) \in$ Rel and RBangRelQ: $(R,!Q) \in$ bangRel Rel by
fact +
show ? case
proof (induct rule: simActFreeCases)
case Der
from PRelQ have $P \rightsquigarrow{ }^{\wedge}<$ Rel $>Q$ by(rule Sim)
with QTrans obtain $P^{\prime}$ where PTrans: $P \Longrightarrow \hat{i} \alpha \prec P^{\prime}$ and $P^{\prime}$ RelQ':
$\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}$
by (blast dest: $\operatorname{sim} E$ )
from PTrans have $P\left\|R \Longrightarrow{ }_{i} \alpha \alpha \prec P^{\prime}\right\| R \mathbf{b y}$ (rule Weak-Late-Semantics.Par1F) moreover from $P^{\prime}$ Rel $Q^{\prime} R$ BangRel $Q$ have $\left(P^{\prime}\left\|R, Q^{\prime}\right\|!Q\right) \in$ bangRel
Rel
by (rule Rel.BRPar)
ultimately show ?case using BangRelRel' by blast
qed
qed
next
case(cPar2B aa x $\left.Q^{\prime} P\right)$
have $I H: \wedge P .(P,!Q) \in$ bangRel Rel $\Longrightarrow$ weakSimAct $P\left(a a « x » \prec Q^{\prime}\right) P$
Rel' by fact
have $x$ Fresh $Q$ : $x \sharp Q$ by fact
have $(P, Q \|!Q) \in$ bangRel Rel and $x \sharp P$ by fact +
thus ?case
proof (induct rule: BRParCases)
case (BRPar P R)
have PRelQ: $(P, Q) \in$ Rel and RBangRelQ: $(R,!Q) \in$ bangRel Rel by
fact +
have $x \sharp P \| R$ by fact
hence $x$ Fresh $P$ : $x \sharp P$ and $x$ FreshR: $x \sharp R$ by simp +
from equtRel' show ?case
proof (induct rule: simActBoundCases)
case(Input a)
have $a a=$ InputS $a$ by fact
with RBangRelQ IH have weakSimAct $R\left(a<x>\prec Q^{\prime}\right) R$ Rel' by blast
with $x$ Fresh $R$ obtain $R^{\prime \prime}$ where $L 1: \forall u . \exists R^{\prime} . R \Longrightarrow_{l} u$ in $R^{\prime \prime} \rightarrow a<x>\prec$ $R^{\prime} \wedge\left(R^{\prime}, Q^{\prime}[x::=u]\right) \in R e l^{\prime}$
by (force simp add: weakSimAct-def)
have $\forall u . \exists P^{\prime} . P \| R \Longrightarrow_{l} u$ in $\left(P \| R^{\prime \prime}\right) \rightarrow a<x>\prec P^{\prime} \wedge\left(P^{\prime},(Q \|\right.$
$\left.\left.Q^{\prime}\right)[x::=u]\right) \in \operatorname{Rel}^{\prime}$
proof(rule allI)
fix $u$
from L1 obtain $R^{\prime}$ where $R$ Trans: $R \Longrightarrow_{l} u$ in $R^{\prime \prime} \rightarrow a<x>\prec R^{\prime}$ and $R^{\prime} \operatorname{Rel}^{\prime} Q^{\prime}:\left(R^{\prime}, Q^{\prime}[x::=u]\right) \in \operatorname{Rel}^{\prime}$
by blast
from RTrans xFresh $P$ have $P \| R \Longrightarrow_{l} u$ in $\left(P \| R^{\prime \prime}\right) \rightarrow a<x>\prec P \| R^{\prime}$
by(rule Weak-Late-Step-Semantics.Par2B)
moreover have $\left(P \| R^{\prime},\left(Q \| Q^{\prime}\right)[x::=u]\right) \in \operatorname{Rel}^{\prime}$
proof -
from PRelQ $R^{\prime} \operatorname{Rel}^{\prime} Q^{\prime}$ have $\left(P\left\|R^{\prime}, Q\right\| Q^{\prime}[x::=u]\right) \in \operatorname{Rel}^{\prime}$ by (rule ParComp)
with $x$ Fresh $Q$ show ?thesis by (simp add: forget)
qed
ultimately show $\exists P^{\prime} . P \| R \Longrightarrow_{l} u$ in $\left(P \| R^{\prime \prime}\right) \rightarrow a<x>\prec P^{\prime} \wedge\left(P^{\prime}\right.$,
$\left.\left(Q \| Q^{\prime}\right)[x::=u]\right) \in \operatorname{Rel}^{\prime}$
by blast
qed
thus ?case by blast

## next

case(BoundOutput a)
have $a a=$ BoundOutputS $a$ by fact
with IH RBangRelQ have weakSimAct $R\left(a<\nu x>\prec Q^{\prime}\right) R$ Rel' by blast with $x$ Fresh $R$ obtain $R^{\prime}$ where RTrans: $R \Longrightarrow \Longrightarrow_{l} a<\nu x>\prec R^{\prime}$ and $R^{\prime}$ BangRelQ ${ }^{\prime}:\left(R^{\prime}, Q^{\prime}\right) \in$ Rel $^{\prime}$ by (simp add: weakSimAct-def, blast)
from RTrans xFresh $P$ have $P\left\|R \Longrightarrow \Longrightarrow_{l} \hat{}^{a} a<\nu x>\prec P\right\| R^{\prime}$ by (auto intro: Weak-Late-Semantics.Par2B)
moreover from PRelQ $R^{\prime}$ BangRelQ' have $\left(P\left\|R^{\prime}, Q\right\| Q^{\prime}\right) \in \operatorname{Rel}^{\prime}$
by (rule ParComp)
ultimately show ? case by blast
qed
qed
next
case $\left(c \operatorname{Par2F} \alpha Q^{\prime} P\right)$
have $I H: \bigwedge P .(P,!Q) \in$ bangRel Rel $\Longrightarrow$ weakSimAct $P\left(\alpha \prec Q^{\prime}\right) P$ Rel $^{\prime}$ by fact
have $(P, Q \|!Q) \in$ bangRel Rel by fact
thus ?case
proof (induct rule: BRParCases)
case (BRPar PR)
have PRelQ: $(P, Q) \in$ Rel and RBangRelQ: $(R,!Q) \in$ bangRel Rel by
fact +
show ?case
proof (induct rule: simActFreeCases)
case Der
from $R$ BangRelQ have weakSimAct $R\left(\alpha \prec Q^{\prime}\right) R$ Rel $^{\prime}$ by $($ rule IH $)$
then obtain $R^{\prime}$ where RTrans: $R \Longrightarrow \Longrightarrow_{l}^{\prime} \alpha \prec R^{\prime}$ and $R^{\prime} \operatorname{Rel} Q^{\prime}:\left(R^{\prime}, Q^{\prime}\right) \in$
$R e l^{\prime}$ by (simp add: weakSimAct-def, blast)
from RTrans have $P\|R \Longrightarrow \hat{l} \alpha \prec P\| R^{\prime} \mathbf{b y}$ (rule Weak-Late-Semantics.Par2F) moreover from $P \operatorname{RelQ} R^{\prime} \operatorname{Rel} Q^{\prime}$ have $\left(P\left\|R^{\prime}, Q\right\| Q^{\prime}\right) \in$ Rel $^{\prime}$ by (rule ParComp)
ultimately show ?case by blast
qed
qed
next
case $\left(c C o m m 1\right.$ a x $\left.Q^{\prime} b Q^{\prime \prime} P\right)$
have $Q$ Trans: $Q \longmapsto a<x>\prec Q^{\prime}$ by fact
have $I H: \bigwedge P .(P,!Q) \in$ bangRel Rel $\Longrightarrow$ weakSimAct $P\left(a[b] \prec Q^{\prime \prime}\right) P$ Rel $^{\prime}$
by fact
have $(P, Q \|!Q) \in$ bangRel Rel and $x \sharp P$ by fact +
thus ?case
proof (induct rule: BRParCases)
case (BRPar P R)
have PRelQ: $(P, Q) \in$ Rel and RBangRel $Q:(R,!Q) \in$ bangRel Rel by
fact +
have $x \sharp P \| R$ by fact
hence $x$ Fresh $P: x \sharp P$ by simp
show ?case
proof (induct rule: simActFreeCases)
case Der
from PRelQ have $P \rightsquigarrow \wedge<$ Rel $>Q$ by(rule Sim)
with $Q$ Trans $x$ Fresh $P$ obtain $P^{\prime} P^{\prime \prime}$ where PTrans: $P \Longrightarrow_{l} b$ in $P^{\prime \prime} \rightarrow a<x>$
$\prec P^{\prime}$
and $P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}[x::=b]\right) \in \operatorname{Rel}$
by (blast dest: simE)
from RBangRelQ have weakSimAct $R\left(a[b] \prec Q^{\prime \prime}\right) R$ Rel ${ }^{\prime}$ by (rule IH)
then obtain $R^{\prime}$ where RTrans: $R \Longrightarrow \Longrightarrow_{l}^{\wedge} a[b] \prec R^{\prime}$
and $R^{\prime} \operatorname{Rel} Q^{\prime \prime}:\left(R^{\prime}, Q^{\prime \prime}\right) \in \operatorname{Rel}^{\prime}$
by (simp add: weakSimAct-def, blast)

```
            from PTrans RTrans have \(P \| R \Longrightarrow \hat{l_{i}} \tau \prec\left(P^{\prime} \| R^{\prime}\right)\)
                        by (rule Weak-Late-Semantics.Comm1)
            moreover from \(P^{\prime} \operatorname{Rel} Q^{\prime} R^{\prime} \operatorname{Rel} Q^{\prime \prime}\) have \(\left(P^{\prime}\left\|R^{\prime}, Q^{\prime}[x::=b]\right\| Q^{\prime \prime}\right) \in \operatorname{Rel}^{\prime}\)
                    by (rule ParComp)
            ultimately show ?case by blast
        qed
        qed
    next
        case (cComm2 a b \(Q^{\prime} x Q^{\prime \prime} P\) )
        have \(Q\) Trans: \(Q \longmapsto a[b] \prec Q^{\prime}\) by fact
        have \(I H: \wedge P .(P,!Q) \in\) bangRel Rel \(\Longrightarrow\) weakSimAct \(P\left(a<x>\prec Q^{\prime \prime}\right) P\)
Rel' by fact
    have \((P, Q \|!Q) \in\) bangRel Rel and \(x \sharp P\) by fact +
    thus ?case
    proof (induct rule: BRParCases)
    case (BRPar P R)
        have PRelQ: \((P, Q) \in\) Rel and RBangRelQ: \((R,!Q) \in\) bangRel Rel by
    have \(x \sharp P \| R\) by fact
    hence \(x\) Fresh \(R\) : \(x \sharp R\) by simp
    show ?case
    proof (induct rule: simActFreeCases)
            case \(\operatorname{Der}\)
            from \(P\) RelQ have \(P \rightsquigarrow^{\wedge}<\operatorname{Rel}>Q\) by (rule Sim)
            with QTrans obtain \(P^{\prime}\) where PTrans: \(P \Longrightarrow \Longrightarrow_{l} a[b] \prec P^{\prime}\) and \(P^{\prime}\) RelQ':
```

fact +
$\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}$
by (blast dest: simE)
from RBangRelQ have weakSimAct $R\left(a<x>\prec Q^{\prime \prime}\right) R$ Rel' by (rule IH)
with $x$ Fresh $R$ obtain $R^{\prime} R^{\prime \prime}$ where RTrans: $R \Longrightarrow{ }_{l} b$ in $R^{\prime \prime} \rightarrow a<x>\prec R^{\prime}$
and $R^{\prime}$ BangRel ${ }^{\prime \prime}:\left(R^{\prime}, Q^{\prime \prime}[x::=b]\right) \in$ Rel $^{\prime}$
by (simp add: weakSimAct-def, blast)

```
            from PTrans RTrans have P| R ¢ ^^|}\tau\prec(\mp@subsup{P}{}{\prime}|\mp@subsup{R}{}{\prime}
                by(rule Weak-Late-Semantics.Comm2)
            moreover from P'RelQ' R'BangRelQ' have ( }\mp@subsup{P}{}{\prime}||\mp@subsup{R}{}{\prime},\mp@subsup{Q}{}{\prime}||\mp@subsup{Q}{}{\prime\prime}[x::=b]
Rel'
            by(rule ParComp)
            ultimately show ?case by blast
            qed
        qed
    next
        case(cClose1 a x Q' y Q ' P)
    have QTrans: Q\longmapstoa<x>}\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    have IH:\bigwedgeP.(P,!Q)\in bangRel Rel \Longrightarrow weakSimAct P (a<\nuy>}\prec~\mp@subsup{Q}{}{\prime\prime})
Rel' by fact
    have }(P,Q|!Q)\in\mathrm{ bangRel Rel and }x\sharpP\mathrm{ and }y\sharpP\mathrm{ by fact+
    thus ?case
    proof(induct rule: BRParCases)
        case(BRPar P R)
        have PRelQ: (P,Q)\inRel and RBangRelQ: (R,!Q) \in bangRel Rel by
fact+
            have }x\sharpPP|R\mathrm{ by fact
            hence xFreshP: x\sharpP by simp
            have }y\sharpP|R\mathrm{ by fact
            hence yFreshR: y }\sharpR\mathrm{ and yFreshP: y }\sharpP\mathrm{ by simp +
            show ?case
            proof(induct rule: simActFreeCases)
            case Der
            from PRelQ have P}\rightsquigarrow^<Rel>Q by(rule Sim
    with QTrans xFreshP obtain P' P'\prime where PTrans: P \Longrightarrow>ly in P'}\mp@subsup{P}{}{\prime\prime}->a<x
\prec '
                                    and }\mp@subsup{P}{}{\prime}\operatorname{RelQ}\mp@subsup{Q}{}{\prime}:(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime}[x::=y])\inRe
                    by(blast dest: simE)
            from RBangRelQ have weakSimAct R (a<\nuy>< < Q') R Rel' by(rule IH)
            with yFreshR obtain R' where RTrans: R\Longrightarrow``|
                        and R'RelQ':}:(\mp@subsup{R}{}{\prime},\mp@subsup{Q}{}{\prime\prime})\in\mp@subsup{REl}{}{\prime
                by(simp add: weakSimAct-def, blast)
            from PTrans RTrans yFreshP yFreshR have P| | R \^\hat{l}\tau}\prec<\nuy>(\mp@subsup{P}{}{\prime}
R')
                by(rule Weak-Late-Semantics.Close1)
            moreover from P'RelQ' R'RelQ' have (<\nuy> ( }\mp@subsup{P}{}{\prime}|\mp@subsup{R}{}{\prime}),<\nuy>(\mp@subsup{Q}{}{\prime}[x::=y
| ( ''|}))\inRel'
                by(force intro: ParComp Res)
            ultimately show ?case by blast
        qed
        qed
    next
        case(cClose2 a y Q' x Q'I P)
```

```
    have QTrans: Q\longmapstoa<\nuy> \prec Q' by fact
    have IH:\bigwedgeP.(P,!Q)\in bangRel Rel \LongrightarrowweakSimAct P (a<x>}\prec-\mp@subsup{Q}{}{\prime\prime})
Rel' by fact
    have }(P,Q|!Q)\in\mathrm{ bangRel Rel and }x\sharpP\mathrm{ and }y\sharpP\mathrm{ by fact+
    thus ?case
    proof(induct rule: BRParCases)
        case(BRPar P R)
        have PRelQ: (P,Q)\inRel and RBangRelQ: (R,!Q) \in bangRel Rel by
fact+
    have }x\sharpP||R\mathrm{ by fact
    hence xFreshR: x #R by simp
    have }y\sharpP|R\mathrm{ by fact
    hence yFreshP: y\sharpP and yFreshR: y#R by simp+
    show ?case
    proof(induct rule: simActFreeCases)
        case Der
        from PRelQ have P\rightsquigarrow^<<Rel>}Q\mathrm{ by(rule Sim)
```



```
                        and P'RelQ':}(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\operatorname{Rel
                by(blast dest: simE)
            from RBangRelQ have weakSimAct R ( }a<x>\prec\prec\mp@subsup{Q}{}{\prime\prime})R\mathrm{ Rel' by(rule IH)
            with xFreshR obtain }\mp@subsup{R}{}{\prime}\mp@subsup{R}{}{\prime\prime}\mathrm{ where RTrans: R <}\mp@subsup{}{y}{\prime}y\mathrm{ in }\mp@subsup{R}{}{\prime\prime}->a<x>\prec\prec\mp@subsup{R}{}{\prime
                    and R'RelQ'\prime:}(\mp@subsup{R}{}{\prime},\mp@subsup{Q}{}{\prime\prime}[x::=y])\in\mp@subsup{R}{el}{
            by(simp add: weakSimAct-def, blast)
        from PTrans RTrans yFreshP yFreshR have P| | =\^``}\tau\prec<\nuy>(\mp@subsup{P}{}{\prime}
R')
            by(rule Weak-Late-Semantics.Close2)
            moreover from P'RelQ' R'RelQ'\prime have ( <\nuy> ( }\mp@subsup{P}{}{\prime}|\mp@subsup{R}{}{\prime}),<\nuy>(\mp@subsup{Q}{}{\prime}
Q'\[x::=y])) \in Rel'
                by(force intro: ParComp Res)
            ultimately show ?case by blast
            qed
        qed
    next
        case(cBang Rs)
        have IH:\P.(P,Q|!Q)\in bangRel Rel \Longrightarrow weakSimAct P Rs P Rel' by
fact
    have (P,!Q) \in bangRel Rel by fact
    thus ?case
    proof(induct rule: BRBangCases)
        case(BRBang P)
        have PRelQ: (P,Q)\in Rel by fact
        hence (!P,!Q)\in bangRel Rel by(rule Rel.BRBang)
        with PRelQ have (P|!P,Q|!Q)\in bangRel Rel by(rule Rel.BRPar)
        hence weakSimAct (P|!P) Rs (P|!P) Rel' by(rule IH)
        thus ?case
        proof(simp (no-asm) add: weakSimAct-def, auto)
```

fix $Q^{\prime} a x$
assume weakSimAct $(P \|!P)\left(a<\nu x>\prec Q^{\prime}\right)(P \|!P)$ Rel $^{\prime}$ and $x \sharp P$
then obtain $P^{\prime}$ where PTrans: $(P \|!P) \Longrightarrow{ }_{l}^{\dot{\prime}} a<\nu x>\prec P^{\prime}$
and $P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in$ Rel $^{\prime}$
by (simp add: weakSimAct-def, blast)
from PTrans have $!P \Longrightarrow \Longrightarrow_{i} a<\nu x>\prec P^{\prime}$
by (force intro: Weak-Late-Step-Semantics.Bang simp add: weakTransi-tion-def)
with $P^{\prime}$ RelQ $Q^{\prime}$ show $\exists P^{\prime} .!P \Longrightarrow \hat{l^{\prime}} a<\nu x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in$ Rel $^{\prime}$ by blast next
fix $Q^{\prime} a x$
assume weakSimAct $(P \|!P)\left(a<x>\prec Q^{\prime}\right)(P \|!P) \operatorname{Rel}^{\prime}$ and $x \sharp P$
then obtain $P^{\prime \prime}$ where $L 1: \forall u . \exists P^{\prime} . P \|!P \Longrightarrow_{l} u$ in $P^{\prime \prime} \rightarrow a<x>\prec P^{\prime}$
$\wedge\left(P^{\prime}, Q^{\prime}[x::=u]\right) \in R e l^{\prime}$
by (simp add: weakSimAct-def, blast)
have $\forall u . \exists P^{\prime} .!P \Longrightarrow_{l} u$ in $P^{\prime \prime} \rightarrow a<x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}[x::=u]\right) \in \operatorname{Rel}^{\prime}$
proof(rule allI)
fix $u$
from $L 1$ obtain $P^{\prime}$ where PTrans: $P \|!P \Longrightarrow_{l} u$ in $P^{\prime \prime} \rightarrow a<x>\prec P^{\prime}$ and $P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}[x::=u]\right) \in \operatorname{Rel}^{\prime}$
by blast
from PTrans have $!P \Longrightarrow_{l} u$ in $P^{\prime \prime} \rightarrow a<x>\prec P^{\prime}$ by (rule Weak-Late-Step-Semantics.Bang)
with $P^{\prime} \operatorname{Rel} Q^{\prime}$ show $\exists P^{\prime}$. $!P \Longrightarrow_{l} u$ in $P^{\prime \prime} \rightarrow a<x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}[x::=u]\right)$
$\in R e l^{\prime}$ by blast
qed
thus $\exists P^{\prime \prime} . \forall u . \exists P^{\prime} .!P \Longrightarrow_{l} u$ in $P^{\prime \prime} \rightarrow a<x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}[x::=u]\right) \in$
$R e l^{\prime}$ by blast
next
fix $Q^{\prime} \alpha$
assume weakSimAct $(P \|!P)\left(\alpha \prec Q^{\prime}\right)(P \|!P)$ Rel $^{\prime}$
then obtain $P^{\prime}$ where PTrans: $(P \|!P) \Longrightarrow \Longrightarrow_{l}^{\wedge} \alpha \prec P^{\prime}$
and $P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}^{\prime}$
by (simp add: weakSimAct-def, blast)
from PTrans show $\exists P^{\prime} .!P \Longrightarrow{ }_{l} \alpha \prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in R e l^{\prime}$
proof(induct rule: transitionCases)
case Step
have $P \|!P \Longrightarrow_{l} \alpha \prec P^{\prime}$ by fact
hence $!P \Longrightarrow_{l} \alpha \prec P^{\prime}$ by(rule Weak-Late-Step-Semantics.Bang)
with $P^{\prime}$ RelQ' show ?case by (force simp add: weakTransition-def)
next
case Stay
have $\alpha \prec P^{\prime}=\tau \prec P \|!P$ by fact
hence $\alpha e q \tau$ : $\alpha=\tau$ and $P^{\prime} e q P: P^{\prime}=P \|!P$ by (simp add: residual.inject)+ have $!P \Longrightarrow{ }_{l} \tau \prec!P \operatorname{by}($ simp add: weakTransition-def)
moreover from $P^{\prime}$ eq $P P^{\prime} \operatorname{Rel} Q^{\prime}$ have $\left(!P, Q^{\prime}\right) \in$ Rel $^{\prime}$ by (blast intro:
RelStay)
ultimately show ?case using $\alpha e q \tau$ by blast
qed
qed

```
        qed
        qed
        qed
        moreover from PRelQ have (!P,!Q) \in bangRel Rel by(rule Rel.BRBang)
        ultimately show ?thesis by(simp add: simDef)
qed
end
theory Weak-Late-Bisim-Pres
    imports Weak-Late-Bisim-SC Weak-Late-Sim-Pres Strong-Late-Bisim-SC
begin
lemma tauPres:
    fixes P :: pi
    and }\quadQ::p
    assumes P\approxQ
    shows \tau.(P) \approx\tau.(Q)
proof -
    let ?X = {(\tau.(P),\tau.(Q))|PQ.P\approxQ}
    from assms have (\tau.(P),\tau.(Q)) \in?X by auto
    thus ?thesis
        by(coinduct rule: weakBisimCoinduct)
            (auto simp add: pi.inject intro: Weak-Late-Sim-Pres.tauPres symmetric)
qed
lemma inputPres:
    fixes }P:: p
    and }Q::p
    and a :: name
    and x :: name
    assumes PSimQ:}\forally.P[x::=y]\approxQ[x::=y
    shows }a<x>.P\approxa<x>.
proof -
    let ?X = {(a<x>.P, a<x>.Q)| a xP Q.\forally.P[x::=y]\approxQ[x::=y]}
    {
        fix axP axQ p
        assume (axP, axQ) \in?X
            then obtain ax P Q where A: }\forally.P[x::=y]\approxQ[x::=y] and B:axP
a<x>.P and C: axQ =a<x>.Q
            by auto
            have \y.((p::name prm) • P)[(p | x)::=y] \approx (p\cdotQ)[(p\cdotx)::=y]
            proof -
            fix y
            from A have }P[x::=(\mathrm{ rev p • y)] }~Q[x::=(\mathrm{ rev p • y)]
```

```
            by blast
    hence }(p\cdot(P[x::=(\operatorname{rev}p\cdoty)]))\approxp\cdot(Q[x::=(\operatorname{rev}p\cdoty)]
            by(rule eqvtI)
            thus }(p\cdotP)[(p\cdotx)::=y]\approx(p\cdotQ)[(p\cdotx)::=y
            by(simp add: eqvts pt-pi-rev[OF pt-name-inst, OF at-name-inst])
    qed
    hence ((p::name prm) • axP, p • axQ) \in?X using B C
        by auto
}
hence eqvt ?X by(simp add: eqvt-def)
from PSimQ have ( }a<x>.P,a<x>.Q)\in?X by aut
thus ?thesis
proof(coinduct rule: weakBisimCoinduct)
    case(cSim P Q)
    thus ?case using <eqvt ?X>
        by(force intro: inputPres)
next
    case(cSym P Q)
    thus ?case
        by(blast dest: symmetric)
    qed
qed
lemma outputPres:
    fixes P :: pi
    and }Q:: p
    and a :: name
    and b :: name
    assumes P\approxQ
    shows a{b}.(P)\approxa{b}.(Q)
proof -
    let ? }X={(a{b}.(P),a{b}.(Q))|abPQ.P\approxQ
    from assms have (a{b}.(P),a{b}.(Q)) \in?X by auto
    thus ?thesis
    by (coinduct rule: weakBisimCoinduct)
        (auto simp add: pi.inject intro: Weak-Late-Sim-Pres.outputPres symmetric)
qed
lemma resPres:
    fixes P :: pi
    and }Q::p
    and x :: name
    assumes PBiSimQ:P\approxQ
    shows <\nux>P\approx<\nux>Q
```

```
proof -
    let ?X = {x.\existsPQ.P\approxQ\wedge(\existsa. x=(<\nua>P,<\nua>Q))}
    from PBiSimQ have (<\nux>P,<\nux>>Q)\in?X by blast
    moreover have \}\PQa.P\rightsquigarrow^<weakBisim> Q\Longrightarrow<\nua>P\rightsquigarrow^<<(?X\cup weak
Bisim)}><<\nua>
    proof -
        fix PQa
        assume PSimQ: P\rightsquigarrow^<weakBisim> Q
        moreover have }\PQa.P\approxQ\Longrightarrow(<\nua>P,<\nua>Q)\in?X\cup weakBisi
by blast
    moreover have weakBisim \subseteq?X \cup weakBisim by blast
    moreover have eqvt weakBisim by(rule eqvt)
    moreover have eqvt (?X \cup weakBisim)
            by(auto simp add: eqvt-def dest: eqvtI)+
        ultimately show <\nua>P\leadsto"<(?X\cup weakBisim)}><<\nua>
            by(rule Weak-Late-Sim-Pres.resPres)
    qed
    ultimately show ?thesis using PBiSimQ
        by(coinduct rule: weakBisimCoinductAux, blast dest: unfoldE)
qed
lemma matchPres:
    fixes P :: pi
    and }Q::p
    and a :: name
    and b :: name
    assumes P}\approx
    shows [a\frownb]P\approx[a\frownb]Q
proof -
    let ? }X={([a\frownb]P,[a\frownb]Q)| a b P Q. P \approx Q 
    from assms have ([a\frownb]P,[a\frownb]Q)\in?X by auto
    thus ?thesis
    proof(coinduct rule: weakBisimCoinduct)
        case(cSim P Q)
        {
            fix PQab
            assume P}\approx
            hence P}\mp@subsup{~}{`}{`}<\mathrm{ weakBisim> Q by(rule unfoldE)
            moreover {
                fix PQa
                    assume P}\approx
                        moreover have [a\frowna]P\approxP by(rule matchId)
                    ultimately have [a\frowna]P\approxQ by(blast intro: transitive)
            }
            moreover have weakBisim \subseteq?X \cup weakBisim by blast
            ultimately have [a\frownb]P\leadsto <(?X U weakBisim )}>[a\frownb]
```

```
            by(rule matchPres)
    }
    with}\langle(P,Q)\in?X> show ?case by aut
    next
        case(cSym P Q)
        thus ?case by(auto simp add: pi.inject dest: symmetric)
    qed
qed
lemma mismatchPres:
    fixes P :: pi
    and }Q:: p
    and a :: name
    and b :: name
    assumes P\approxQ
    shows [a\not=b]P\approx[a\not=b]Q
proof -
    let ?X = {([a\not=b]P,[a\not=b]Q)| abPQ.P\approxQ}
    from assms have ([a\not=b]P,[a\not=b]Q)\in?X by auto
    thus ?thesis
    proof(coinduct rule: weakBisimCoinduct)
        case(cSim P Q)
        {
            fix PQab
            assume P}\approx
            hence }P\rightsquigarrow^<weakBisim> Q by(rule unfoldE
            moreover {
            fix PQab
            assume P}\approxQ\mathrm{ and (a::name) }=
            note \langleP}\approxQ
            moreover from <a\not=b\rangle have [a\not=b]P\approxP by(rule mismatchId)
            ultimately have [a\not=b]P\approxQ by(blast intro: transitive)
            }
            moreover have weakBisim \subseteq?X \cup weakBisim by blast
            ultimately have [a\not=b]P\rightsquigarrow}<<(?X\cup\mathrm{ weakBisim )}>[a\not=b]
                by(rule mismatchPres)
        }
            with «(P,Q)\in? \\ show ?case by auto
        next
            case(cSym P Q)
            thus ?case by(auto simp add: pi.inject dest: symmetric)
    qed
qed
lemma parPres:
    fixes P :: pi
    and }Q:: p
```

```
    and \(\quad R:: p i\)
    assumes \(P \approx Q\)
    shows \(P\|R \approx Q\| R\)
proof -
    let ?ParSet \(=\{(\) resChain lst \((P \| R)\), resChain lst \((Q \| R)) \mid\) lst \(P Q R . P \approx\)
\(Q\}\)
    have \(B C: \wedge P Q . P \| Q=\) resChain []\((P \| Q)\) by auto
    from assms have \((P\|R, Q\| R) \in\) ?ParSet by(blast intro: \(B C\) )
    thus ?thesis
    proof (coinduct rule: weakBisimCoinduct)
        case \((c \operatorname{Sim} P R \quad Q R)\)
        \{
            fix \(P Q R\) lst
            assume \(P \approx Q\)
            from eqvtI have eqvt (?ParSet \(\cup\) weakBisim)
            by (auto simp add: eqvt-def, blast)
            moreover have \(\bigwedge P Q a .(P, Q) \in\) ?ParSet \(\cup\) weakBisim \(\Longrightarrow(<\nu a>P\),
\(<\nu a>Q) \in ?\) ParSet \(\cup\) weakBisim
                by(blast intro: resChain.step[THEN sym] resPres)
            moreover \{
                from \(\langle P \approx Q\rangle\) have \(P \rightsquigarrow{ }^{\wedge}<\) weakBisim \(>Q\) by(rule unfold \(E\) )
                    moreover note \(\langle P \approx Q\) 〉
                        moreover \{
                    fix \(P Q R\)
                    assume \(P \approx Q\)
                    moreover have \(P \| R=\operatorname{resChain}[](P \| R)\) by simp
                    moreover have \(Q \| R=\) resChain []\((Q \| R)\) by simp
                    ultimately have \((P\|R, Q\| R) \in\) ?ParSet \(\cup\) weakBisim by blast
            \}
            moreover \{
                    fix \(P Q a\)
                    assume \(A:(P, Q) \in ?\) ParSet \(\cup\) weakBisim
                        hence \((<\nu a>P,<\nu a>Q) \in\) ?ParSet \(\cup\) weakBisim (is ?goal)
                    apply (auto intro: resPres)
                    by (rule-tac \(x=a \#\) lst in exI) auto
            \}
                ultimately have \((P \| R) \rightsquigarrow^{\wedge}<(\) ?ParSet \(\cup\) weakBisim \()>(Q \| R)\) using
eqvt 〈eqvt(?ParSet \(\cup\) weakBisim) >
            by(rule Weak-Late-Sim-Pres.parPres)
            \}
            ultimately have resChain lst \((P \| R) \rightsquigarrow \wedge<(? P a r S e t ~ \cup\) weakBisim \()>\) resChain
lst \((Q \| R)\)
            by (rule resChainI)
        \}
    with \(\langle(P R, Q R) \in\) ?ParSet \(\rangle\) show ?case by blast
```

```
    next
        case(cSym PR QR)
        thus ?case by(auto dest: symmetric)
    qed
qed
lemma bangPres:
    fixes P :: pi
    and }Q::p
    assumes PBisimQ: P}\approx
    shows !P\approx!Q
proof -
    let ?X = (bangRel weakBisim)
    let ?Y = Strong-Late-Bisim.bisim O (bangRel weakBisim) O Strong-Late-Bisim.bisim
    from eqvt Strong-Late-Bisim.bisimEqvt have eqvtY: eqvt?Y by(blast intro: eqvt-
BangRel)
    have XsubY: ?X \subseteq?Y by(auto intro: Strong-Late-Bisim.reflexive)
    have RelStay: }\PQ.(P|!P,Q)\in?Y\Longrightarrow(!P,Q)\in?
    proof(auto)
        fix PQR T
        assume PBisimQ:P|!P~Q
            and QBRR:}(Q,R)\in\mathrm{ bangRel weakBisim
            and RBisimT: R~T
        have !P~Q
        proof -
            have !P~P|!P by(rule Strong-Late-Bisim-SC.bangSC)
            thus ?thesis using PBisimQ by(rule Strong-Late-Bisim.transitive)
        qed
        with QBRR RBisimT show (!P,T) \in?Y by blast
    qed
    have ParCompose: \PQ R T.\llbracketP\approxQ; (R,T)\in?Y\rrbracket\Longrightarrow(P|R,Q|T)\in
?Y
    proof -
        fix PQRT
        assume PBisimQ: P}\approx
            and RYT: }\quad(R,T)\in?
    thus (P|R,Q|T)\in?Y
    proof(auto)
            fix }\mp@subsup{T}{}{\prime}\mp@subsup{R}{}{\prime
            assume T'BisimT: T'~T and RBisimR': R~ R'
                and }\mp@subsup{R}{}{\prime}BR\mp@subsup{T}{}{\prime}:(\mp@subsup{R}{}{\prime},\mp@subsup{T}{}{\prime})\in\mathrm{ bangRel weakBisim
            have }P|R~P|\mp@subsup{R}{}{\prime
            proof -
            from RBisimR' have R|P~ R'|P by(rule Strong-Late-Bisim-Pres.parPres)
```

```
    moreover have P|R~R|P and R'| P~P| R' by(rule Strong-Late-Bisim-SC.parSym)+
            ultimately show ?thesis by(blast intro: Strong-Late-Bisim.transitive)
        qed
    moreover from PBisimQ R'BRT' have (P| R',Q| T') \in bangRel weakBisim
by(rule BRPar)
    moreover have Q| T'~Q|T
    proof -
    from T'BisimT have T'| Q ~ T| Q by(rule Strong-Late-Bisim-Pres.parPres)
                moreover have Q| T'~ T'| Q and T| Q ~ Q |T by(rule
Strong-Late-Bisim-SC.parSym)+
            ultimately show ?thesis by(blast intro: Strong-Late-Bisim.transitive)
        qed
        ultimately show ?thesis by blast
    qed
qed
```

have ResCong: $\bigwedge P Q x .(P, Q) \in ? Y \Longrightarrow(<\nu x>P,<\nu x>Q) \in ? Y$
by (auto intro: BRRes Strong-Late-Bisim-Pres.resPres transitive)
from PBisim $Q$ have $(!P,!Q) \in ? X$ by (rule BRBang)
moreover from eqvt have eqvt (bangRel weakBisim) by(rule eqvtBangRel)
ultimately show ?thesis
proof(coinduct rule: weakBisimTransitiveCoinduct)
case $(c \operatorname{Sim} P Q)$
from $\langle(P, Q) \in$ ? $X$ 〉
show $P \rightsquigarrow<$ ? $Y>Q$
proof (induct)
case (BRBang $P$ Q)
have $P \approx Q$ by fact
moreover hence $P \rightsquigarrow \wedge<$ weakBisim $>Q$ by (blast dest: unfoldE)
moreover have $\wedge P Q . P \approx Q \Longrightarrow P \rightsquigarrow \wedge<$ weakBisim $>Q$ by (blast dest:
unfoldE)
moreover from Strong-Late-Bisim.bisimEqvt eqvt have eqvt ?Y by (blast
intro: eqvtBangRel)
ultimately show $!P \rightsquigarrow \wedge<? Y>!Q$ using ParCompose ResCong RelStay
XsubY
by(rule-tac Weak-Late-Sim-Pres.bangPres, simp-all)
next
case (BRPar P Q R T)
have $P$ BiSim $Q: P \approx Q$ by fact
have RBangRelT: $(R, T) \in$ ? $X$ by fact
have RSimT: $R \rightsquigarrow \subset<? Y>T$ by fact
moreover from PBiSim $Q$ have $P \rightsquigarrow<$ weakBisim $>Q$ by (blast dest: un-
foldE)
moreover from RBangRelT have $(R, T) \in$ ? $Y$ by (blast intro: Strong-Late-Bisim.reflexive)
ultimately show $P\left\|R \rightsquigarrow \wedge^{\wedge}<? Y>Q\right\| T$ using ParCompose ResCong eqvt
eqvt $Y\langle P \approx Q\rangle$
by (rule-tac Weak-Late-Sim-Pres.parCompose)

```
    next
            case(BRRes P Q x)
            have P\rightsquigarrow^<? Y>Q by fact
            thus <\nux>P\rightsquigarrow`<?Y><\nux>Q using ResCong eqvtY XsubY
                by(rule-tac Weak-Late-Sim-Pres.resPres, simp-all)
        qed
    next
        case(cSym P Q)
        thus ?case by(metis symmetric bangRelSymetric)
    qed
qed
end
theory Weak-Late-Cong-Pres
    imports Weak-Late-Cong Weak-Late-Step-Sim-Pres Weak-Late-Bisim-Pres
begin
lemma tauPres:
    fixes P :: pi
    and }Q::p
    assumes P\simeqQ
    shows }\tau.(P)\simeq\tau.(Q
using assms
by(blast intro: unfoldI Weak-Late-Step-Sim-Pres.tauPres dest: congruenceWeakBisim
symetric)
lemma outputPres:
    fixes }P:: p
    and }Q::p
    assumes P\simeqQ
    shows }a{b}.P\simeqa{b}.
using assms
by(blast intro: unfoldI Weak-Late-Step-Sim-Pres.outputPres dest: congruenceWeak-
Bisim symetric)
lemma inputPres:
    fixes P :: pi
    and }Q::p
    and a :: name
    and x :: name
    assumes PSimQ:}\forally.P[x::=y]\simeqQ[x::=y
    shows }a<x>.P\simeqa<x>.
```

```
using assms
apply (rule-tac unfoldI)
apply(rule-tac Weak-Late-Step-Sim-Pres.inputPres, auto intro: congruenceWeak-
Bisim)
by (rule-tac Weak-Late-Step-Sim-Pres.inputPres, auto intro: congruence WeakBisim
Weak-Late-Bisim.symmetric)
lemma matchPres:
    fixes \(P:: p i\)
    and \(\quad Q:: p i\)
    and \(a::\) name
    and \(b::\) name
    assumes \(P \simeq Q\)
    shows \([a \frown b] P \simeq[a \frown b] Q\)
using assms
by (blast intro: unfoldI Weak-Late-Step-Sim-Pres.matchPres dest: unfoldE symet-
ric)
lemma mismatchPres:
    fixes \(P\) :: \(p i\)
    and \(\quad Q:: p i\)
    and \(a::\) name
    and \(b::\) name
    assumes \(P \simeq Q\)
    shows \([a \neq b] P \simeq[a \neq b] Q\)
using assms
by (blast intro: unfoldI Weak-Late-Step-Sim-Pres.mismatchPres dest: unfoldE sy-
metric)
lemma sumPres:
    fixes \(P:: p i\)
    and \(\quad Q:: p i\)
    and \(\quad R:: p i\)
    assumes \(P \simeq Q\)
    shows \(P \oplus R \simeq Q \oplus R\)
using assms
by (blast intro: Weak-Late-Bisim.reflexive unfoldI Weak-Late-Step-Sim-Pres.sumPres
dest: unfoldE symetric)
lemma parPres:
    fixes \(P\) :: \(p i\)
    and \(\quad Q:: p i\)
    and \(\quad R:: p i\)
```

```
    assumes \(P \simeq Q\)
    shows \(P\|R \simeq Q\| R\)
proof -
    have \(\bigwedge P Q R . \llbracket P \rightsquigarrow<\) weakBisim \(>Q ; P \approx Q \rrbracket \Longrightarrow P \| R \rightsquigarrow<\) weakBisim \(>Q \|\)
R
    proof -
        fix \(P Q R\)
        assume \(P \rightsquigarrow<\) weakBisim \(>Q\) and \(P \approx Q\)
        thus \(P \| R \rightsquigarrow<\) weakBisim \(>Q \| R\)
        using Weak-Late-Bisim-Pres.parPres Weak-Late-Bisim-Pres.resPres Weak-Late-Bisim.reflexive
Weak-Late-Bisim.eqvt
            by (blast intro: Weak-Late-Step-Sim-Pres.parPres)
    qed
    with assms show ?thesis
        by (blast intro: unfoldI dest: congruence WeakBisim unfoldE symetric)
qed
lemma resPres:
    fixes \(P:: p i\)
    and \(\quad Q:: p i\)
    and \(x::\) name
    assumes \(P e q Q: P \simeq Q\)
    shows \(<\nu x>P \simeq<\nu x>Q\)
proof -
    have \(\bigwedge P Q x . P \rightsquigarrow<\) weakBisim \(>Q \Longrightarrow<\nu x>P \rightsquigarrow<\) weakBisim \(><\nu x>Q\)
    proof -
        fix \(P Q x\)
        assume \(P \rightsquigarrow<\) weakBisim \(>Q\)
    with Weak-Late-Bisim.eqvt Weak-Late-Bisim-Pres.resPres show \(<\nu x>P \rightsquigarrow<\) weakBisim \(>\)
\(<\nu x>Q\)
            by (blast intro: Weak-Late-Step-Sim-Pres.resPres)
    qed
        with assms show ?thesis
            by (blast intro: unfoldI dest: congruence WeakBisim unfoldE symetric)
qed
lemma congruenceBang:
    fixes \(P:: p i\)
    and \(\quad Q:: p i\)
    assumes \(P \simeq Q\)
    shows \(!P \simeq!Q\)
proof -
    have \(\bigwedge P Q . \llbracket P \rightsquigarrow<\) weakBisim \(>Q ; P \simeq Q \rrbracket \Longrightarrow!P \rightsquigarrow<\) weakBisim \(>!Q\)
```

```
    proof -
    fix PQ
    assume P\rightsquigarrow<weakBisim> Q and P\simeqQ
    hence !P\rightsquigarrow<bangRel weakBisim> !Q using unfoldE(1) congruence WeakBisim
Weak-Late-Bisim.eqvt
            by(rule Weak-Late-Step-Sim-Pres.bangPres)
    moreover have bangRel weakBisim \subseteq weakBisim
        proof auto
            fix ab
            assume (a,b) \in bangRel weakBisim
            thus a\approxb
                    apply(induct rule: bangRel.induct)
                    apply (metis Weak-Late-Bisim-Pres.bangPres)
                    apply (metis Weak-Late-Bisim.reflexive Weak-Late-Bisim.symmetric
Weak-Late-Bisim.transitive Weak-Late-Bisim-Pres.parPres Weak-Late-Bisim-SC.parSym)
                by (metis Weak-Late-Bisim-Pres.resPres)
            qed
        ultimately show!P \rightsquigarrow<\mathrm{ weakBisim }>!Q
            by(rule Weak-Late-Step-Sim.monotonic)
        qed
    with assms show ?thesis
    by(blast intro: unfoldI dest: unfoldE symetric congruenceWeakBisim)
qed
end
theory Early-Semantics
    imports Agent
begin
declare name-fresh[simp del]
nominal-datatype freeRes = InputR name name }\quad(-<-> [110, 110
110)
\begin{tabular}{|lc}
\(\mid\) OutputR name name \\
\(\mid\) TauR & \((-[-][110,110] 110)\) \\
\((\tau 110)\)
\end{tabular}
nominal-datatype residual \(=\) BoundOutputR name «name» pi \((-<\nu->\prec-[110\), 110, 110] 110)
| FreeR freeRes pi
lemma alphaBoundOutput:
fixes \(a\) :: name
and \(x\) :: name
and \(P:: p i\)
and \(x^{\prime}::\) name
assumes \(A 1: x^{\prime} \sharp P\)
```

```
    shows \(a<\nu x>\prec P=a<\nu x^{\prime}>\prec\left(\left[\left(x, x^{\prime}\right)\right] \cdot P\right)\)
proof (cases \(x=x^{\prime}\) )
    assume \(x=x^{\prime}\)
    thus ?thesis by simp
next
    assume \(x \neq x^{\prime}\)
    with \(A 1\) show ?thesis
        by (simp add: residual.inject alpha name-fresh-left name-calc)
qed
```

declare name-fresh [simp]
abbreviation Transitions-Freejudge (- $-[80,80] 80)$ where $\alpha \prec P^{\prime} \equiv($ FreeR $\alpha P^{\prime}$ )
inductive TransitionsEarly $::$ pi $\Rightarrow$ residual $\Rightarrow$ bool $(-\longmapsto-[80,80] 80)$
where

$$
\text { Tau: } \quad \tau .(P) \longmapsto \tau \prec P
$$

| Input: $\quad \llbracket x \neq a ; x \neq u \rrbracket \Longrightarrow a<x>. P \longmapsto a<u>\prec(P[x::=u])$
| Output: $\quad a\{b\} . P \longmapsto a[b] \prec P$

| Match: | $\llbracket P \longmapsto V \rrbracket \Longrightarrow[b \frown b] P \longmapsto V$ |
| :--- | :--- |
| Mismatch: | $\llbracket P \longmapsto V ; a \neq b \rrbracket \Longrightarrow[a \neq b] P \longmapsto V$ |


| Open: | $\llbracket P \longmapsto a[b] \prec P^{\prime} ; a \neq b \rrbracket \Longrightarrow<\nu b>P \longmapsto a<\nu b>\prec P^{\prime}$ |
| :--- | :--- |
| Sum1: | $\llbracket P \longmapsto V \rrbracket \Longrightarrow(P \oplus Q) \longmapsto V$ |
| Sum2: | $\llbracket Q \longmapsto V \rrbracket \Longrightarrow(P \oplus Q) \longmapsto V$ |

|Par1B: $\quad \llbracket P \longmapsto a<\nu x>\prec P^{\prime} ; x \sharp P ; x \sharp Q ; x \neq a \rrbracket \Longrightarrow P \| Q \longmapsto$ $a<\nu x>\prec\left(P^{\prime} \| Q\right)$
| Par1F: $\quad \llbracket P \longmapsto \alpha \prec P \rrbracket \Longrightarrow P \| Q \longmapsto \alpha \prec\left(P^{\prime} \| Q\right)$
| Par2B: $\quad \llbracket Q \longmapsto a<\nu x>\prec Q^{\prime} ; x \sharp P ; x \sharp Q ; x \neq a \rrbracket \Longrightarrow P \| Q \longmapsto$
$a<\nu x>\prec\left(P \| Q^{\prime}\right)$
| Par2F: $\quad \llbracket Q \longmapsto \alpha \prec Q \rrbracket \Longrightarrow P \| Q \longmapsto \alpha \prec\left(P \| Q^{\prime}\right)$
| Comm1: $\quad \llbracket P \longmapsto a<b>\prec P^{\prime} ; Q \longmapsto a[b] \prec Q \rrbracket \Longrightarrow P \| Q \longmapsto \tau \prec P^{\prime}$
$\| Q^{\prime}$
| Comm2: $\quad \llbracket P \longmapsto a[b] \prec P^{\prime} ; Q \longmapsto a<b>\prec Q^{\prime} \rrbracket \Longrightarrow P \| Q \longmapsto \tau \prec P^{\prime}$
$\| Q^{\prime}$
| Close1: $\quad \llbracket P \longmapsto a<x>\prec P^{\prime} ; Q \longmapsto a<\nu x>\prec Q^{\prime} ; x \sharp P ; x \sharp Q ; x \neq a \rrbracket$ $\Longrightarrow P \| Q \longmapsto \tau \prec<\nu x>\left(P^{\prime} \| Q^{\prime}\right)$
| Close2: $\quad \llbracket P \longmapsto a<\nu x>\prec P^{\prime} ; Q \longmapsto a<x>\prec Q^{\prime} ; x \sharp P ; x \sharp Q ; x \neq a \rrbracket$
$\Longrightarrow P \| Q \longmapsto \tau \prec<\nu x>\left(P^{\prime} \| Q^{\prime}\right)$
|ResB: $\quad \llbracket P \longmapsto a<\nu x>\prec P^{\prime} ; y \neq a ; y \neq x ; x \sharp P ; x \neq a \rrbracket \Longrightarrow<\nu y>P$ $\longmapsto a<\nu x>\prec\left(<\nu y>P^{\prime}\right)$
| ResF: $\quad \llbracket P \longmapsto \alpha \prec P^{\prime} ; y \sharp \alpha \rrbracket \Longrightarrow<\nu y>P \longmapsto \alpha \prec<\nu y>P^{\prime}$

$$
\text { | Bang: } \quad \llbracket P \|!P \longmapsto V \rrbracket \Longrightarrow!P \longmapsto V
$$

equivariance TransitionsEarly
nominal-inductive TransitionsEarly
by(auto simp add: abs-fresh fresh-fact2)
lemmas $[$ simp $]=$ freeRes.inject
lemma freshOutputAction:
fixes $P$ :: $p i$
and $a$ :: name
and $b$ :: name
and $P^{\prime}:: p i$
and $c$ :: name
assumes $P \longmapsto a[b] \prec P^{\prime}$
and $\quad c \sharp P$
shows $c \neq a$ and $c \neq b$ and $c \sharp P^{\prime}$
proof -
from assms have $c \neq a \wedge c \neq b \wedge c \sharp P^{\prime}$
$\mathbf{b y}\left(\right.$ nominal-induct $x 2==a[b] \prec P^{\prime}$ arbitrary: $P^{\prime}$ rule: TransitionsEarly.strong-induct)
(fastforce simp add: residual.inject abs-fresh freeRes.inject)+
thus $c \neq a$ and $c \neq b$ and $c \sharp P^{\prime}$
by blast+
qed
lemma freshInputAction:
fixes $P:: p i$
and $a$ :: name
and $b$ :: name
and $\quad P^{\prime}:: p i$
and $c$ :: name
assumes $P \longmapsto a<b>\prec P^{\prime}$
and $\quad c \sharp P$
shows $c \neq a$
using assms
by (nominal-induct $x \mathbb{2}==a<b>\prec P^{\prime}$ arbitrary: $P^{\prime}$ rule: TransitionsEarly.strong-induct) (auto simp add: residual.inject abs-fresh)
lemma freshBoundOutputAction:
fixes $P$ :: $p i$
and $a$ :: name
and $x$ :: name
and $P^{\prime}:: p i$
and $c$ :: name

```
    assumes }P\longmapstoa<\nux>\prec\mp@subsup{P}{}{\prime
    and c}c\sharp
    shows c\not=a
using assms
by(nominal-induct x2 ==a<\nux> \prec P' avoiding: x arbitrary: P' rule:Transition-
sEarly.strong-induct) (auto simp add: residual.inject abs-fresh fresh-left calc-atm
dest: freshOutputAction)
lemmas freshAction = freshOutputAction freshInputAction freshBoundOutputAc-
tion
lemma freshInputTransition:
    fixes }P\mathrm{ :: pi
    and a :: name
    and u :: name
    and }\mp@subsup{P}{}{\prime}:: p
    and c :: name
    assumes }P\longmapstoa<u>\prec\mp@subsup{P}{}{\prime
    and c}c\sharp
    and c}
    shows c\sharp 
using assms
by(nominal-induct x2==a<u> \prec P' arbitrary: P' rule:TransitionsEarly.strong-induct)
    (fastforce simp add: residual.inject name-fresh-abs fresh-fact1 fresh-fact2)+
lemma freshBoundOutputTransition:
    fixes P :: pi
    and a :: name
    and }x\mathrm{ :: name
    and }\mp@subsup{P}{}{\prime}:: p
    and c :: name
    assumes }P\longmapstoa<\nux>\prec\mp@subsup{P}{}{\prime
    and c}c\sharp
    and c}c\not=
    shows c\sharp P'
using assms
apply(nominal-induct x2 ==a<\nux> \prec P' avoiding:x arbitrary: P' rule:Transi-
tionsEarly.strong-induct)
apply(fastforce simp add: residual.inject name-fresh-abs alpha' fresh-left calc-atm
dest: freshOutputAction | simp | auto simp add: abs-fresh residual.inject alpha'
calc-atm)
apply(fastforce simp add: residual.inject name-fresh-abs alpha' fresh-left calc-atm
dest: freshOutputAction | simp | auto simp add: abs-fresh residual.inject alpha'
calc-atm)
```

apply (fastforce simp add: residual.inject name-fresh-abs alpha' fresh-left calc-atm dest: freshOutputAction $|\operatorname{simp}|$ auto simp add: abs-fresh residual.inject alpha' calc-atm)
apply (force simp add: residual.inject name-fresh-abs alpha' fresh-left calc-atm dest: freshOutputAction $\mid$ simp $\mid$ auto simp add: abs-fresh residual.inject alpha' calc-atm) apply (force simp add: residual.inject name-fresh-abs alpha' fresh-left calc-atm dest: freshOutputAction $\mid$ simp $\mid$ auto simp add: abs-fresh residual.inject alpha' calc-atm) apply (force simp add: residual.inject name-fresh-abs alpha' fresh-left calc-atm dest: freshOutputAction $\mid$ simp $\mid$ auto simp add: abs-fresh residual.inject alpha' calc-atm) apply (force simp add: residual.inject name-fresh-abs alpha' fresh-left calc-atm dest: freshOutputAction $\mid$ simp $\mid$ auto simp add: abs-fresh residual.inject alpha' calc-atm) apply (force simp add: residual.inject name-fresh-abs alpha' fresh-left calc-atm dest: freshOutputAction $\mid$ simp $\mid$ auto simp add: abs-fresh residual.inject alpha' calc-atm) apply (force simp add: residual.inject name-fresh-abs alpha' fresh-left calc-atm dest: freshOutputAction $\mid$ simp $\mid$ auto simp add: abs-fresh residual.inject alpha' calc-atm) apply (force simp add: residual.inject name-fresh-abs alpha' fresh-left calc-atm dest: freshOutputAction $\mid$ simp $\mid$ auto simp add: abs-fresh residual.inject alpha' calc-atm) apply (force simp add: residual.inject name-fresh-abs alpha' fresh-left calc-atm dest: freshOutputAction $\mid$ simp $\mid$ auto simp add: abs-fresh residual.inject alpha' calc-atm) apply (force simp add: residual.inject name-fresh-abs alpha' fresh-left calc-atm dest: freshOutputAction $\mid$ simp $\mid$ auto simp add: abs-fresh residual.inject alpha' calc-atm) apply (fastforce simp add: residual.inject name-fresh-abs alpha' fresh-left calc-atm dest: freshOutputAction)
apply (fastforce simp add: residual.inject name-fresh-abs alpha' fresh-left calc-atm dest: freshOutputAction)
apply (fastforce simp add: residual.inject name-fresh-abs alpha' fresh-left calc-atm dest: freshOutputAction)
apply (fastforce simp add: residual.inject name-fresh-abs alpha' fresh-left calc-atm dest: freshOutputAction)
apply (auto simp add: residual.inject name-fresh-abs alpha' fresh-left calc-atm dest: freshOutputAction)
done
lemma freshTauTransition:
fixes $P$ :: $p i$
and $P^{\prime}:: p i$
and $c::$ name
assumes $P \longmapsto \tau \prec P^{\prime}$
and $\quad c \sharp P$
shows $c \sharp P^{\prime}$
using assms
$\operatorname{apply}\left(\right.$ nominal-induct $x 2==\tau \prec P^{\prime}$ arbitrary: $P^{\prime}$ rule: TransitionsEarly.strong-induct)
by(fastforce simp add: residual.inject abs-fresh dest: freshOutputAction freshInputTransition freshBoundOutputTransition)+
lemma freshFreeTransition:
fixes $P$ :: $p i$

```
and \alpha :: freeRes
and }\mp@subsup{P}{}{\prime}:: p
and c :: name
```

assumes $P \longmapsto \alpha \prec P^{\prime}$
and $\quad c \sharp P$
and $c \sharp \alpha$
shows $c \sharp P^{\prime}$
using assms
by (nominal-induct $\alpha$ rule: freeRes.strong-inducts)
(auto dest: freshInputTransition freshOutputAction freshTauTransition)
lemmas freshTransition $=$ freshInputTransition freshOutputAction freshFreeTransition
freshBoundOutputTransition freshTauTransition
lemma substTrans $[$ simp $]: b \sharp P \Longrightarrow((P:: p i)[a::=b])[b::=c]=P[a::=c]$ apply (simp add: injPermSubst[THEN sym])
apply (simp add: renaming)
by (simp add: pt-swap[OF pt-name-inst, OF at-name-inst])
lemma Input:
fixes $a$ :: name
and $x::$ name
and $u::$ name
and $\quad P:: p i$

```
    shows }a<x>.P\longmapstoa<u>\precP[x::=u
proof -
    obtain y::name where }y\not=a\mathrm{ and }y\not=u\mathrm{ and }y\sharp
            by(generate-fresh name) (auto simp add: fresh-prod)
    from }\langley\not=a\rangle\langley\not=u\rangle\mathrm{ have }a<y>.([(x,y)]\cdotP)\longmapstoa<u>\prec ([(x,y)]\cdotP)[y::=u
        by(rule Input)
    with }\langley\sharpP\rangle\mathrm{ show ?thesis by(simp add: alphaInput renaming name-swap)
qed
lemma Par1B:
    fixes P :: pi
    and a :: name
    and }x\mathrm{ :: name
    and }\mp@subsup{P}{}{\prime}:: p
    and }Q :: p
```

    assumes \(P \longmapsto a<\nu x>\prec P^{\prime}\)
    and \(\quad x \sharp Q\)
    shows \(P \| Q \longmapsto a<\nu x>\prec\left(P^{\prime} \| Q\right)\)
    proof -
obtain $y$ :: name where $y \sharp P$ and $y \sharp Q$ and $y \neq a$ and $y \sharp P^{\prime}$ by (generate-fresh name) (auto simp add: fresh-prod)
from $\left.\langle P \longmapsto a<\nu x\rangle \prec P^{\prime}\right\rangle\left\langle y \sharp P^{\prime}\right\rangle$ have $P \longmapsto a<\nu y>\prec\left([(x, y)] \cdot P^{\prime}\right)$ by (simp add: alphaBoundOutput)
hence $P \| Q \longmapsto a<\nu y>\prec\left(\left([(x, y)] \cdot P^{\prime}\right) \| Q\right)$ using $\langle y \sharp P\rangle\langle y \sharp Q\rangle\langle y \neq a\rangle$ by (rule Par1B)
with $\langle x \sharp Q\rangle\langle y \sharp Q\rangle\left\langle y \sharp P^{\prime}\right\rangle$ show ?thesis
by (subst alphaBoundOutput) (auto simp add: name-fresh-fresh)
qed
lemma Par2B:
fixes $Q:: p i$
and $a$ :: name
and $x$ :: name
and $\quad Q^{\prime}:: p i$
and $P:: p i$
assumes $Q \longmapsto a<\nu x>\prec Q^{\prime}$
and $\quad x \sharp P$
shows $P \| Q \longmapsto a<\nu x>\prec\left(P \| Q^{\prime}\right)$
proof -
obtain $y$ ::name where $y \sharp P$ and $y \sharp Q$ and $y \neq a$ and $y \sharp Q^{\prime}$
by(generate-fresh name) (auto simp add: fresh-prod)
from $\left.\langle Q \longmapsto a<\nu x\rangle \prec Q^{\prime}\right\rangle\left\langle y \sharp Q^{\prime}\right\rangle$ have $Q \longmapsto a<\nu y>\prec\left([(x, y)] \cdot Q^{\prime}\right)$ by (simp add: alphaBoundOutput)
hence $P \| Q \longmapsto a<\nu y>\prec\left(P \|\left([(x, y)] \cdot Q^{\prime}\right)\right)$ using $\langle y \sharp P\rangle\langle y \sharp Q\rangle\langle y \neq a\rangle$ by (rule Par2B)
with $\langle x \sharp P\rangle\langle y \sharp P\rangle\left\langle y \sharp Q^{\prime}\right\rangle$ show ? ?thesis
by (subst alphaBoundOutput[of y]) (auto simp add: name-fresh-fresh)
qed
lemma inputInduct[consumes 1, case-names cInput cMatch cMismatch cSum1 cSum2 cPar1 cPar2 cRes cBang]:
fixes $P:: p i$
and $a$ :: name
and $u$ :: name
and $P^{\prime}:: p i$
and $\quad F::{ }^{\prime} a::$ fs-name $\Rightarrow p i \Rightarrow$ name $\Rightarrow$ name $\Rightarrow p i \Rightarrow$ bool
and $C:: ' a:: f s-n a m e$
assumes Trans: $P \longmapsto a<u>\prec P^{\prime}$
and $\quad \bigwedge a x P u C . \llbracket x \sharp C ; x \neq u ; x \neq a \rrbracket \Longrightarrow F C(a<x>. P)$ a $u(P[x::=u])$
and $\quad \bigwedge P a u P^{\prime} b C . \llbracket P \longmapsto a<u>\prec P^{\prime} ; \bigwedge C . F C P$ a $u P^{\prime} \rrbracket \Longrightarrow F C([b \frown b] P)$
a $u P^{\prime}$
and $\quad \bigwedge P a u P^{\prime} b$ c $C . \llbracket P \longmapsto a<u>\prec P^{\prime} ; \bigwedge C . F C P$ a $u P^{\prime} ; b \neq c \rrbracket \Longrightarrow F$ $C([b \neq c] P)$ a u $P^{\prime}$
and $\quad \bigwedge P$ a $u P^{\prime} Q C . \llbracket P \longmapsto a<u>\prec P^{\prime} ; \bigwedge C . F C P$ a $u P \rrbracket \Longrightarrow F C(P \oplus$ Q) $a u P^{\prime}$
and $\quad \bigwedge Q a u Q^{\prime} P C . \llbracket Q \longmapsto a<u>\prec Q^{\prime} ; \bigwedge C . F C Q a u Q^{\prime} \rrbracket \Longrightarrow F C(P$ $\oplus Q) a u Q^{\prime}$
and $\quad \wedge P a u P^{\prime} Q C . \llbracket P \longmapsto a<u>\prec P^{\prime} ; \bigwedge C . F C P$ a $u P \rrbracket \Longrightarrow F C(P \|$ Q) $a u\left(P^{\prime} \| Q\right)$
and $\quad \bigwedge Q$ au $Q^{\prime} P C . \llbracket Q \longmapsto a<u>\prec Q^{\prime} ; \bigwedge C . F C Q$ au $Q \rrbracket \Longrightarrow F C(P \|$ Q) $a u\left(P \| Q^{\prime}\right)$
and $\quad \bigwedge P a u P^{\prime} x C . \llbracket P \longmapsto a<u>\prec P^{\prime} ; x \neq a ; x \neq u ; x \sharp C ; \bigwedge C . F C P a$ $u P \rrbracket \Longrightarrow F C(<\nu x>P)$ a $u\left(<\nu x>P^{\prime}\right)$
and $\quad \bigwedge P$ a u $P^{\prime} C . \llbracket P \|!P \longmapsto a<u>\prec P^{\prime} ; \bigwedge C . F C(P \|!P)$ a u $P^{\prime} \rrbracket \Longrightarrow$ $F C(!P)$ a u $P^{\prime}$

```
    shows F C P a u P'
using assms
by(nominal-induct x2 ==a<u> \prec P' avoiding:C arbitrary: P' rule:Transition-
sEarly.strong-induct)
    (auto simp add: residual.inject)
```

lemma inputAlpha:
assumes $P \longmapsto a<u>\prec P^{\prime}$
and $\quad u \sharp P$
and $\quad r \sharp P^{\prime}$
shows $P \longmapsto a<r>\prec\left([(u, r)] \cdot P^{\prime}\right)$
using assms
proof (nominal-induct avoiding: r rule: inputInduct)
case (cInput a x Pur)
from $\langle x \neq u\rangle\langle u \sharp a<x\rangle . P\rangle$ have $u \neq a$ and $u \sharp P$ by (simp add: abs-fresh) +
have $a<x>. P \longmapsto a<r>\prec P[x::=r]$
by (rule Input)
thus ? case using $\langle r \sharp P[x::=u]\rangle\langle u \sharp P\rangle$
by (simp add: injPermSubst substTrans)
next
case (cMatch P a u $P^{\prime}$ br)
thus ?case by (force intro: Match)
next
case(cMismatch Pau $P^{\prime} b c r$ )
thus ?case by (force intro: Mismatch)
next
case (cSum1 P a u $P^{\prime} Q r$ )
thus ?case by (force intro: Sum1)
next
case (cSum2 $Q$ a u $Q^{\prime} P r$ )
thus ?case by(force intro: Sum2)
next
case (cPar1 P a u $P^{\prime} Q r$ )
thus ?case by (force intro: Par1F simp add: eqvts name-fresh-fresh)
next
case (cPar2 Q a u $Q^{\prime} P r$ )
thus ?case by (force intro: Par2F simp add: eqvts name-fresh-fresh)

```
next
    case(cRes P a u P' x r)
    thus ?case by(force intro: ResF simp add: eqvts calc-atm abs-fresh)
next
    case(cBang P a u P' R)
    thus ?case by(force intro: Bang)
qed
lemma Close1:
    fixes P :: pi
    and a :: name
    and }x\mathrm{ :: name
    and }\mp@subsup{P}{}{\prime}::p
    and }Q :: p
    and }\mp@subsup{Q}{}{\prime}:: p
    assumes }P\longmapstoa<x>\prec\mp@subsup{P}{}{\prime
    and }\quadQ\longmapstoa<\nux>\prec\mp@subsup{Q}{}{\prime
    and }\quadx\sharp
    shows P|Q\longmapsto\tau\prec<\nux>( }
proof -
    obtain y::name where }y\sharpP\mathrm{ and }y\sharpQ\mathrm{ and }y\not=a\mathrm{ and }y\sharp\mp@subsup{Q}{}{\prime}\mathrm{ and }y\sharp\mp@subsup{P}{}{\prime
    by(generate-fresh name) (auto simp add: fresh-prod)
    from }\langleP\longmapstoa<x>< < P'\rangle\langlex\sharpP\rangle\langley\sharp\mp@subsup{P}{}{\prime}\rangle\mathrm{ have }P\longmapstoa<y>\prec ([(x,y)]\cdot\mp@subsup{P}{}{\prime}
    by(rule inputAlpha)
    moreover from <Q \longmapsto a<\nux> \prec Q'〉\langley\sharp Q'> have }Q\longmapstoa<\nuy>< < [(x,y)] 
Q')
    by(simp add: alphaBoundOutput)
    ultimately have P|Q\longmapsto~}\prec<<\nuy>(([(x,y)]\cdot\mp@subsup{P}{}{\prime})|([(x,y)]\cdot\mp@subsup{Q}{}{\prime}))\mathrm{ using « y
#P\rangle\langley\sharpQ>\langley\not=a\rangle
    by(rule Close1)
    with }\langley\sharp\mp@subsup{P}{}{\prime}\rangle\langley\sharp\mp@subsup{Q}{}{\prime}\rangle\mathrm{ show ?thesis by(subst alphaRes) (auto simp add: name-fresh-fresh)
qed
lemma Close2:
    fixes P :: pi
    and a :: name
    and }x\mathrm{ :: name
    and }\mp@subsup{P}{}{\prime}::p
    and }Q :: p
    and }\mp@subsup{Q}{}{\prime}::p
assumes }P\longmapstoa<\nux>\prec\mp@subsup{P}{}{\prime
and }\quadQ\longmapstoa<x>\prec\prec\mp@subsup{Q}{}{\prime
and }x\sharp
shows P|Q\longmapsto\tau\prec<\nux>(\mp@subsup{P}{}{\prime}|\mp@subsup{Q}{}{\prime})
```

```
proof -
    obtain y::name where }y\sharpP\mathrm{ and }y\sharpQ\mathrm{ and }y\not=a\mathrm{ and }y\sharp\mp@subsup{Q}{}{\prime}\mathrm{ and }y\sharp\mp@subsup{P}{}{\prime
        by(generate-fresh name) (auto simp add: fresh-prod)
    from }\langleP\longmapstoa<\nux>< \prec P'\rangle\langley\sharp\mp@subsup{P}{}{\prime}>\mathrm{ have }P\longmapstoa<\nuy> \prec ([(x,y)]\cdot P'
        by(simp add: alphaBoundOutput)
```



```
y)] \cdot Q')
        by(rule inputAlpha)
    ultimately have }P|Q\longmapsto\tau\prec<\nuy>(([(x,y)]\cdot\mp@subsup{P}{}{\prime})|([(x,y)]\cdot\mp@subsup{Q}{}{\prime}))\mathrm{ using < y
#P\rangle\langley#Q\rangle\langley\not=a\rangle
    by(rule Close2)
    with }\langley\sharp\mp@subsup{P}{}{\prime}\rangle\langley\sharp\mp@subsup{Q}{}{\prime}\rangle\mathrm{ show ?thesis by(subst alphaRes) (auto simp add: name-fresh-fresh)
qed
lemma ResB:
    fixes }P::p
    and a :: name
    and }x\mathrm{ :: name
    and }\mp@subsup{P}{}{\prime}::p
    and y :: name
    assumes }P\longmapstoa<\nux>\prec\mp@subsup{P}{}{\prime
    and }y\not=
    and }y\not=
    shows <\nuy>P\longmapstoa<\nux>}\prec(<\nuy>\mp@subsup{P}{}{\prime}
proof -
    obtain z :: name where }z\sharpP\mathrm{ and }z\sharp\mp@subsup{P}{}{\prime}\mathrm{ and }z\not=a\mathrm{ and }z\not=
        by(generate-fresh name) (auto simp add: fresh-prod)
    from <P\longmapstoa<\nux> \prec P'\rangle\langlez\sharp P'> have P\longmapstoa<\nuz> \prec ([(x,z)] \cdot P')
        by(simp add: alphaBoundOutput)
    hence <\nuy>P\longmapstoa<\nuz> \prec(<\nuy>([(x,z)] \cdot P
P\rangle}\langlez\not=a
    by(rule-tac ResB) auto
    thus?thesis using }\langlez\not=y\rangle\langley\not=x\rangle\langlez\sharp\mp@subsup{P}{}{\prime}
    by(subst alphaBoundOutput[where x'=z]) (auto simp add: eqvts calc-atm abs-fresh)
qed
lemma outputInduct[consumes 1, case-names Output Match Mismatch Sum1 Sum2
Par1 Par2 Res Bang]:
    fixes P :: pi
    and a :: name
    and b :: name
    and }\mp@subsup{P}{}{\prime}::p
    and F :: 'a::fs-name }=>\mathrm{ pi m name }=>\mathrm{ name }=>\mathrm{ pi }=>\mathrm{ bool
    and }C::' 'a::fs-nam
    assumes Trans: P\longmapstoa[b]\prec 㐌
```

and $\quad \bigwedge a b P C . F C(a\{b\} . P) a b P$
and $\bigwedge P a b P^{\prime} c C . \llbracket P \longmapsto$ OutputR ab$\prec P^{\prime} ; \bigwedge C . F C P a b P^{\Uparrow} \rrbracket F C$ $([c \frown c] P) a b P^{\prime}$
and $\bigwedge P a b P^{\prime} c d C . \llbracket P \longmapsto$ OutputR $a b \prec P^{\prime} ; \bigwedge C . F C P a b P^{\prime} ; c \neq d \rrbracket$ $\Longrightarrow F C([c \neq d] P) a b P^{\prime}$
and $\quad \bigwedge P a b P^{\prime} Q C . \llbracket P \longmapsto$ OutputR $a b \prec P^{\prime} ; \bigwedge C . F C P a b P^{\prime} \rrbracket \Longrightarrow F C$ $(P \oplus Q) a b P^{\prime}$
and $\Lambda Q$ ab $Q^{\prime} P C . \llbracket Q \longmapsto$ OutputR $a b \prec Q^{\prime} ; \Lambda C . F C Q a b Q^{\top} \rrbracket \Longrightarrow F$ $C(P \oplus Q) a b Q^{\prime}$
and $\quad \bigwedge P a b P^{\prime} Q C . \llbracket P \longmapsto$ OutputR $a b \prec P^{\prime} ; \wedge C . F C P a b P^{\prime} \rrbracket \Longrightarrow F C$ $(P \| Q) a b\left(P^{\prime} \| Q\right)$
and $\Lambda Q$ ab $Q^{\prime} P C . \llbracket Q \longmapsto O u t p u t R a b \prec Q^{\prime} ; \Lambda C . F C Q a b Q^{\top} \Longrightarrow F$ $C(P \| Q) a b\left(P \| Q^{\prime}\right)$
and $\bigwedge P a b P^{\prime} x C . \llbracket P \longmapsto$ OutputR $a b \prec P^{\prime} ; x \neq a ; x \neq b ; x \sharp C ; \bigwedge C . F$
$C P a b P^{\dagger} \rrbracket \Longrightarrow$

$$
F C(<\nu x>P) \text { ab }\left(<\nu x>P^{\prime}\right)
$$

and $\quad \wedge P a b P^{\prime} C . \llbracket P \|!P \longmapsto$ OutputR $a b \prec P^{\prime} ; \bigwedge C . F C(P \|!P) a b P \rrbracket$ $\Longrightarrow F C(!P) a b P^{\prime}$
shows $F C P a b P^{\prime}$
using assms
by (nominal-induct $x 2==a[b] \prec P^{\prime}$ avoiding: $C$ arbitrary: $P^{\prime}$ rule: TransitionsEarly.strong-induct) (auto simp add: residual.inject)
lemma boundOutputInduct[consumes 2, case-names Match Mismatch Open Sum1 Sum2 Par1 Par2 Res Bang]:

```
    fixes \(P\) :: \(p i\)
    and \(a\) :: name
    and \(x\) :: name
    and \(P^{\prime}:: p i\)
    and \(\quad F::\left({ }^{\prime} a:: f s\right.\)-name \() \Rightarrow p i \Rightarrow\) name \(\Rightarrow\) name \(\Rightarrow p i \Rightarrow\) bool
    and \(C:: ' a:: f s\)-name
```

    assumes \(a\) : \(P \longmapsto a<\nu x>\prec P^{\prime}\)
    and \(x\) Fresh \(P\) : \(x \sharp P\)
    and \(\quad c\) Match: \(\bigwedge P\) a x \(P^{\prime} b C . \llbracket P \longmapsto a<\nu x>\prec P^{\prime} ; \bigwedge C . F C P a x P^{\prime} \rrbracket \Longrightarrow\)
    $F C([b \frown b] P) a x P^{\prime}$
and cMismatch: $\bigwedge P$ ax $P^{\prime} b$ c $C . \llbracket P \longmapsto a<\nu x>\prec P^{\prime} ; \bigwedge C$. F CPax $P^{\prime}$;
$b \neq c \rrbracket \Longrightarrow F C([b \neq c] P)$ ax $P^{\prime}$
and cOpen: $\bigwedge P a x P^{\prime} C . \llbracket P \longmapsto($ OutputR $a x) \prec P^{\prime} ; a \neq x \rrbracket \Longrightarrow F C$
$(<\nu x>P)$ a $x P$
and $\quad c S u m 1: ~ \bigwedge P Q a x P^{\prime} C . \llbracket P \longmapsto a<\nu x>\prec P^{\prime} ; \bigwedge C . F C P a x P \rrbracket \Longrightarrow$
$F C(P \oplus Q) a x P^{\prime}$
and $\quad$ SSum2: $\quad \wedge P Q$ a $x Q^{\prime} C . \llbracket Q \longmapsto a<\nu x>\prec Q^{\prime} ; \Lambda C . F C Q a x \quad Q^{\prime} \rrbracket$
$\Longrightarrow F C(P \oplus Q)$ ax $Q^{\prime}$
and $\quad c \operatorname{Par} 1 B: \bigwedge P P^{\prime} Q$ a $x C . \llbracket P \longmapsto a<\nu x>\prec P^{\prime} ; x \sharp Q ; \bigwedge C . F C P a x$
$P^{\dagger} \rrbracket$
$F C(P \| Q) a x\left(P^{\prime} \| Q\right)$
and $\quad c P a r 2 B: \wedge P Q Q^{\prime}$ axC. $\llbracket Q \longmapsto a<\nu x>\prec Q^{\prime} ; x \sharp P ; \bigwedge C . F C Q a x$

```
Q\rrbracket\Longrightarrow
                    FC(P|Q) ax(P| Q')
```



```
                                    \C.FCPaxP\rrbracket\LongrightarrowFC(<\nuy>P) ax(<\nuy>P')
    and cBang: \ \Pax P C}C|P|!P\longmapstoa<\nux>\prec \prec '; \C.FC(P|!P)
x P\ \Longrightarrow
\[
F C(!P) a x P^{\prime}
\]
    shows F C P a x P'
using assms
proof -
    have Goal: \P Rs a x P'C.\llbracketP\longmapstoRs;Rs=a<\nux>\prec 缶; x\sharpP\rrbracket\LongrightarrowFCP
a x P'
    proof -
    fix PRs a x P' C
    assume P\longmapstoRs and Rs=a<\nux>}\prec\mp@subsup{P}{}{\prime}\mathrm{ and }x\sharp
    thus FC C a a x P'
    proof(nominal-induct avoiding: C a x P' rule: TransitionsEarly.strong-induct)
        case(Tau P)
        thus ?case by(simp add: residual.inject)
    next
        case(Input P a x)
        thus ?case by(simp add: residual.inject)
    next
            case(Output P a b)
            thus ?case by(simp add: residual.inject)
    next
        case(Match P Rs b C a x P')
        thus ?case
            by(force intro: cMatch simp add: residual.inject)
    next
        case(Mismatch P Rs b c C a x P')
        thus ?case
            by(force intro: cMismatch simp add: residual.inject)
    next
        case(Sum1 P Q Rs C)
        thus ?case by(force intro: cSum1)
    next
        case(Sum2 P Q Rs C)
        thus ?case by(force intro: cSum2)
    next
        case(Open P a b P'C C ' }x\mp@subsup{P}{}{\prime\prime}\mathrm{ )
        have b\sharpx by fact hence bineqx: b\not=x by simp
        moreover have }a<\nub>\prec\mp@subsup{P}{}{\prime}=\mp@subsup{a}{}{\prime}<\nux>\prec\prec\mp@subsup{P}{}{\prime\prime}\mathrm{ by fact
        ultimately have aeqa': a=\mp@subsup{a}{}{\prime}\mathrm{ and }\mp@subsup{P}{}{\prime}eq\mp@subsup{P}{}{\prime\prime}:\mp@subsup{P}{}{\prime\prime}=[(b,x)]\cdot\mp@subsup{P}{}{\prime}
            by(simp add: residual.inject name-abs-eq)+
        have }x\sharp<\nub>P\mathrm{ by fact
        with bineqx have xFreshP:x\sharpP by(simp add: name-fresh-abs)
        have aineqb: }a\not=b\mathrm{ by fact
```

```
    have PTrans: P\longmapstoa[b]\prec P' by fact
    with xFreshP have xineqa: x }\not=a\mathrm{ by(force dest: freshAction)
    from PTrans have }([(b,x)]\cdotP)\longmapsto[(b,x)]\cdot(a[b]\prec\mp@subsup{P}{}{\prime})\mathrm{ by (rule Transition-
sEarly.eqvt)
    with P'eqP'\prime xineqa aineqb have Trans: }([(b,x)]\cdotP)\longmapstoa[x]\prec\mp@subsup{P}{}{\prime\prime
    by(auto simp add: name-calc)
    hence FC (<\nux>([(b,x)] • P)) a x P'|
    with xFreshP aeqa' show ?case by(simp add: alphaRes)
    next
    case(Par1B P a x P' Q Ca' (' }\mp@subsup{P}{}{\prime}\mp@subsup{P}{}{\prime\prime}
    have }x\sharp\mp@subsup{x}{}{\prime}\mathrm{ by fact hence xineqx': x = x' by simp
    moreover have Eq: a<\nux> \prec( (P'|Q)=\mp@subsup{a}{}{\prime}<\nu\mp@subsup{x}{}{\prime}>\prec \prec P'' by fact
    hence aeqa': a= a' by (simp add: residual.inject)
    have xFreshQ: x }#Q\mathrm{ by fact
    have }\mp@subsup{x}{}{\prime}\sharpP|Q\mathrm{ by fact
    hence \mp@subsup{x}{}{\prime}FreshP: x'\sharpP and \mp@subsup{x}{}{\prime}FreshQ: x'\sharp Q by simp+
    have }\mp@subsup{P}{}{\prime\prime}\mathrm{ eq: }\mp@subsup{P}{}{\prime\prime}=([(x,\mp@subsup{x}{}{\prime})]\cdot\mp@subsup{P}{}{\prime})|
    proof -
        from Eq xineqx' have ( }\mp@subsup{P}{}{\prime}|Q)=[(x,x)]\cdot\mp@subsup{P}{}{\prime\prime
            by(simp add: residual.inject name-abs-eq)
        hence }([(x,\mp@subsup{x}{}{\prime})]\cdot(\mp@subsup{P}{}{\prime}|Q))=\mp@subsup{P}{}{\prime\prime}\mathrm{ by simp
        with x'FreshQ xFreshQ show ?thesis by(simp add: name-fresh-fresh)
    qed
    have }x\sharp\mp@subsup{P}{}{\prime\prime}\mathrm{ by fact
    with P'eq have \mp@subsup{x}{}{\prime}Fresh\mp@subsup{P}{}{\prime}:\mp@subsup{x}{}{\prime}\sharp\mp@subsup{P}{}{\prime}\mathbf{by}(simp add: name-fresh-left name-calc)
    have P\longmapstoa<\nux> \prec P' by fact
    with \mp@subsup{x}{}{\prime}Fresh\mp@subsup{P}{}{\prime}aeq\mp@subsup{a}{}{\prime}}\mathrm{ have }P\longmapsto\mp@subsup{a}{}{\prime}<\nu\mp@subsup{x}{}{\prime}>\prec \prec([(x, x )] \cdot P'
        by(simp add: alphaBoundOutput)
    moreover have \C.F CPa x ([(x, x')] • P')
    proof -
        fix C
        have }\bigwedgeC\mp@subsup{a}{}{\prime}\mp@subsup{x}{}{\prime}\mp@subsup{P}{}{\prime\prime}.\llbracketa<\nux>\prec\mp@subsup{P}{}{\prime}=\mp@subsup{a}{}{\prime}<\nu\mp@subsup{x}{}{\prime}>\prec\mp@subsup{P}{}{\prime\prime};\mp@subsup{x}{}{\prime}\sharpP\rrbracket\LongrightarrowFCP\mp@subsup{a}{}{\prime
\mp@subsup{x}{}{\prime}}\mp@subsup{P}{}{\prime\prime}\mathrm{ by fact
```



```
([(x, \mp@subsup{x}{}{\prime})] \cdot P')
            by(simp add: residual.inject name-abs-eq name-fresh-left name-calc)
            ultimately show F C P a x ' ([(x, x')] • P') using \mp@subsup{x}{}{\prime}FreshP aeqa' by blast
    qed
    ultimately have FC}(P|Q)\mp@subsup{a}{}{\prime}\mp@subsup{x}{}{\prime}(([(x,\mp@subsup{x}{}{\prime})]\cdot\mp@subsup{P}{}{\prime})|Q)\mathrm{ using x'FreshQ
aeqa'
            by(blast intro: cPar1B)
            with }\mp@subsup{P}{}{\prime\prime}eq\mathrm{ show ?case by simp
    next
            case(Par1F P P' Q \alpha)
            thus ?case by(simp add: residual.inject)
    next
            case(Par2B Q a x Q P P C a' x' Q')
```

```
    have }x\sharp\mp@subsup{x}{}{\prime}\mathrm{ by fact hence xineqx': }x\not=\mp@subsup{x}{}{\prime}\mathrm{ by simp
    moreover have Eq: a<\nux>}\prec(P|\mp@subsup{Q}{}{\prime})=\mp@subsup{a}{}{\prime}<\nu\mp@subsup{x}{}{\prime}>\prec\prec\mp@subsup{Q}{}{\prime\prime}\mathrm{ by fact
    hence aeqa': a= a' by (simp add: residual.inject)
    have xFreshP: x\sharpP by fact
    have }\mp@subsup{x}{}{\prime}\sharpP|Q\mathrm{ by fact
    hence \mp@subsup{x}{}{\prime}FreshP: \mp@subsup{x}{}{\prime}\sharpP\mathrm{ and }\mp@subsup{x}{}{\prime}FreshQ: \mp@subsup{x}{}{\prime}\sharpQ by simp+
    have }\mp@subsup{Q}{}{\prime\prime}eq:\mp@subsup{Q}{}{\prime\prime}=P|([(x,\mp@subsup{x}{}{\prime})]\cdot\mp@subsup{Q}{}{\prime}
    proof -
        from Eq xineqx' have (P| Q ) = [(x, x')] \cdot Q ''
        by(simp add: residual.inject name-abs-eq)
        hence }([(x,\mp@subsup{x}{}{\prime})]\cdot(P|\mp@subsup{Q}{}{\prime}))=\mp@subsup{Q}{}{\prime\prime}\mathrm{ by simp
        with \mp@subsup{x}{}{\prime}FreshP xFreshP show ?thesis by(simp add: name-fresh-fresh)
    qed
    have }x\sharp\mp@subsup{Q}{}{\prime\prime}\mathrm{ by fact
```



```
    have }Q\longmapstoa<\nux> \prec Q' by fac
    with \mp@subsup{x}{}{\prime}Fresh\mp@subsup{Q}{}{\prime}\mathrm{ aeqa' have }Q\longmapsto\mp@subsup{a}{}{\prime}<\nu\mp@subsup{x}{}{\prime}>}\prec([(x,\mp@subsup{x}{}{\prime})]\cdot\mp@subsup{Q}{}{\prime}
        by(simp add: alphaBoundOutput)
    moreover have \C.F C Q a x'([(x, x')] • Q')
    proof -
        fix C
        have }\bigwedgeC\mp@subsup{a}{}{\prime}\mp@subsup{x}{}{\prime}\mp@subsup{Q}{}{\prime\prime}.\llbracketa<\nux>\prec\prec\mp@subsup{Q}{}{\prime}=\mp@subsup{a}{}{\prime}<\nu\mp@subsup{x}{}{\prime}>\prec\mp@subsup{Q}{}{\prime\prime};\mp@subsup{x}{}{\prime}\sharpQ\rrbracket\LongrightarrowFCQ a
\mp@subsup{x}{}{\prime}}\mp@subsup{Q}{}{\prime\prime}\mathrm{ by fact
            moreover with aeqa' xineqx' x'Fresh Q' have }a<\nux>\prec\prec\mp@subsup{Q}{}{\prime}=\mp@subsup{a}{}{\prime}<\nu\mp@subsup{x}{}{\prime}>
([(x, x')] • Q')
            by(simp add: residual.inject name-abs-eq name-fresh-left name-calc)
            ultimately show FCQ a x ( 
    qed
    ultimately have FC(P|Q) a' \mp@subsup{x}{}{\prime}(P|([(x,\mp@subsup{x}{}{\prime})] \cdot Q')) using \mp@subsup{x}{}{\prime}FreshP
aeqa'
    by(blast intro: cPar2B)
    with }\mp@subsup{Q}{}{\prime\prime}eq\mathrm{ show ?case by simp
    next
        case(Par2F P P' Q \alpha)
        thus ?case by(simp add: residual.inject)
    next
        case(Comm1 P P' Q Q' a b x)
        thus ?case by(simp add: residual.inject)
    next
        case(Comm2 P P' Q Q' a b x)
        thus ?case by(simp add: residual.inject)
    next
        case(Close1 P P'Q Q' a x y)
        thus ?case by(simp add: residual.inject)
    next
        case(Close2 P P' Q Q' a x y)
        thus ?case by(simp add: residual.inject)
```

```
    next
    case(ResB P a x P' y C a' x' P')
    have }x\sharp\mp@subsup{x}{}{\prime}\mathrm{ by fact hence xineqx': }x\not=\mp@subsup{x}{}{\prime}\mathrm{ by simp
    moreover have Eq: a<\nux> \prec (<\nuy>\mp@subsup{P}{}{\prime})=\mp@subsup{a}{}{\prime}<\nu\mp@subsup{x}{}{\prime}>\prec \prec P'\prime by fact
    hence aeqa': a= a' by (simp add: residual.inject)
    have }y\sharp\mp@subsup{x}{}{\prime}\mathrm{ by fact hence yineqx': y # x' by simp
    moreover have }\mp@subsup{x}{}{\prime}\sharp<\nuy>P\mathrm{ by fact
    ultimately have \mp@subsup{x}{}{\prime}FreshP: \mp@subsup{x}{}{\prime}\sharpP}\mathrm{ by(simp add: name-fresh-abs)
    have yineqx:}y\not=x\mathrm{ and yineqa: }y\not=a\mathrm{ and yFreshC: y }\sharpC\mathrm{ by fact+
    have }\mp@subsup{P}{}{\prime\prime}eq:\mp@subsup{P}{}{\prime\prime}=<\nuy>([(x,\mp@subsup{x}{}{\prime})]\cdot\mp@subsup{P}{}{\prime}
    proof -
        from Eq xineqx' have <\nuy> P' = [(x, x')] \cdot P'\prime
            by(simp add: residual.inject name-abs-eq)
        hence}([(x,\mp@subsup{x}{}{\prime})]\cdot(<\nuy>\mp@subsup{P}{}{\prime}))=\mp@subsup{P}{}{\prime\prime}\mathrm{ by simp
        with yineqx' yineqx show ?thesis by(simp add: name-fresh-fresh)
    qed
```

    have \(x \sharp P^{\prime \prime}\) by fact
    with \(P^{\prime \prime}\) eq yineqx have \(x^{\prime}\) Fresh \(P^{\prime}: x^{\prime} \sharp P^{\prime}\) by (simp add: name-fresh-left
    name-calc name-fresh-abs)
have $P \longmapsto a<\nu x>\prec P^{\prime}$ by fact
with $x^{\prime}$ Fresh $P^{\prime}$ aeqa' have $P \longmapsto a^{\prime}<\nu x^{\prime}>\prec\left(\left[\left(x, x^{\prime}\right)\right] \cdot P^{\prime}\right)$
by (simp add: alphaBoundOutput)
moreover have $\bigwedge C$. F CPax $x^{\prime}\left(\left[\left(x, x^{\prime}\right)\right] \cdot P^{\prime}\right)$
proof -
fix $C$
have $\bigwedge C a^{\prime} x^{\prime} P^{\prime \prime} . \llbracket a<\nu x>\prec P^{\prime}=a^{\prime}<\nu x^{\prime}>\prec P^{\prime \prime} ; x^{\prime} \sharp P \rrbracket \Longrightarrow F C P a^{\prime}$
$x^{\prime} P^{\prime \prime}$ by fact
moreover with aeqa' xineqx' $x^{\prime}$ Fresh $P^{\prime}$ have $a<\nu x>\prec P^{\prime}=a^{\prime}<\nu x^{\prime}>\prec$
$\left(\left[\left(x, x^{\prime}\right)\right] \cdot P^{\prime}\right)$
by (simp add: residual.inject name-abs-eq name-fresh-left name-calc)
ultimately show $F C P a x^{\prime}\left(\left[\left(x, x^{\prime}\right)\right] \cdot P^{\prime}\right)$ using $x^{\prime}$ FreshP aeqa' by blast
qed
ultimately have $F C(<\nu y>P) a^{\prime} x^{\prime}\left(<\nu y>\left(\left[\left(x, x^{\prime}\right)\right] \cdot P^{\prime}\right)\right)$ using yineqx ${ }^{\prime}$
yineqa yFreshC aeqa'
by (force intro: cResB)
with $P^{\prime \prime} e q$ show ?case by simp
next
case(ResFP P ${ }^{\prime} \alpha y$ )
thus ?case by (simp add: residual.inject)
next
case (Bang P Rs)
thus ?case by (force intro: cBang)
qed
qed
with a xFreshP show ?thesis by simp

## qed

lemma tauInduct[consumes 1, case-names Tau Match Mismatch Sum1 Sum2 Par1 Par2 Comm1 Comm2 Close1 Close2 Res Bang]:
fixes $P$ :: $p i$
and $P^{\prime}:: p i$
and $F::{ }^{\prime} a:: f s-n a m e \Rightarrow p i \Rightarrow p i \Rightarrow$ bool
and $C:: ' a:: f s-n a m e$
assumes Trans: $P \longmapsto \tau \prec P^{\prime}$
and $\quad \wedge P C . F C(\tau .(P)) P$
and $\quad \bigwedge P P^{\prime}$ a $C . \llbracket P \longmapsto \tau \prec P^{\prime} ; \bigwedge C . F C P P \rrbracket \Longrightarrow F C([a \frown a] P) P^{\prime}$
and $\quad \bigwedge P P^{\prime} a b C . \llbracket P \longmapsto \tau \prec P^{\prime} ; \bigwedge C . F C P P^{\prime} ; a \neq b \rrbracket \Longrightarrow F C([a \neq b] P)$ $P^{\prime}$
and $\quad \wedge P P^{\prime} Q C . \llbracket P \longmapsto \tau \prec P^{\prime} ; \wedge C . F C P P^{\prime} \rrbracket \Longrightarrow F C(P \oplus Q) P^{\prime}$
and $\quad \bigwedge Q Q^{\prime} P C . \llbracket Q \longmapsto \tau \prec Q^{\prime} ; \bigwedge C . F C Q Q^{\dagger} \rrbracket \Longrightarrow F C(P \oplus Q) Q^{\prime}$
and $\quad \Lambda P P^{\prime} Q C . \llbracket P \longmapsto \tau \prec P^{\prime} ; \Lambda C . F C P P^{\natural} \rrbracket F C(P \| Q)\left(P^{\prime} \| Q\right)$
and $\Lambda Q Q^{\prime} P C . \llbracket Q \longmapsto \tau \prec Q^{\prime} ; \bigwedge C . F C Q Q^{\prime} \rrbracket \Longrightarrow F C(P \| Q)\left(P \| Q^{\prime}\right)$
and $\bigwedge P a b P^{\prime} Q Q^{\prime} C . \llbracket P \longmapsto a<b>\prec P^{\prime} ; Q \longmapsto$ OutputR ab $\quad$ 々 $Q^{\prime} \rrbracket \Longrightarrow F$
$C(P \| Q)\left(P^{\prime} \| Q^{\prime}\right)$
and $\quad \wedge P a b P^{\prime} Q Q^{\prime} C . \llbracket P \longmapsto$ OutputR ab$\prec P^{\prime} ; Q \longmapsto a<b>\prec Q \rrbracket \Longrightarrow F$ $C(P \| Q)\left(P^{\prime} \| Q^{\prime}\right)$
and $\quad \wedge P a x P^{\prime} Q Q^{\prime} C . \llbracket P \longmapsto a<x>\prec P^{\prime} ; Q \longmapsto a<\nu x>\prec Q^{\prime} ; x \sharp P ; x \sharp$
$Q ; x \neq a ; x \sharp C \rrbracket \Longrightarrow F C(P \| Q)\left(<\nu x>\left(P^{\prime} \| Q^{\prime}\right)\right)$
and $\quad \bigwedge P a x P^{\prime} Q Q^{\prime} C . \llbracket P \longmapsto a<\nu x>\prec P^{\prime} ; Q \longmapsto a<x>\prec Q^{\prime} ; x \sharp P ; x \sharp$
$Q ; x \neq a ; x \sharp C \rrbracket \Longrightarrow F C(P \| Q)\left(<\nu x>\left(P^{\prime} \| Q^{\prime}\right)\right)$
and $\quad \triangle P^{\prime} x C . \llbracket P \longmapsto \tau \prec P^{\prime} ; x \sharp C ; \bigwedge C . F C P P^{\rrbracket} \Longrightarrow$
$F C(<\nu x>P)\left(<\nu x>P^{\prime}\right)$
and $\quad \Lambda P P^{\prime} C . \llbracket P \|!P \longmapsto \tau \prec P^{\prime} ; \bigwedge C . F C(P \|!P) P \rrbracket \Longrightarrow F C(!P) P^{\prime}$
shows $F C P P^{\prime}$
using $\left\langle P \longmapsto \tau \prec P^{\prime}\right\rangle$
by(nominal-induct $x 2==\tau \prec P^{\prime}$ avoiding: $C$ arbitrary: $P^{\prime}$ rule: TransitionsEarly.strong-induct)
(auto simp add: residual.inject intro: assms)+
inductive bangPred $::$ pi $\Rightarrow p i \Rightarrow$ bool
where
aux1: bangPred $P(!P)$
| aux2: bangPred $P(P|\mid!P)$
inductive-cases tauCases' ${ }^{\prime}$ simplified pi.distinct residual.distinct] $]: \tau .(P) \longmapsto R s$
inductive-cases inputCases' [simplified pi.inject residual.inject]: $a<b>. P \longmapsto R s$
inductive-cases outputCases' ${ }^{[\text {simplified pi.inject residual.inject] }]: a\{b\} . P \longmapsto R s}$
inductive-cases matchCases' $[$ simplified pi.inject residual.inject $]:[a \frown b] P \longmapsto R s$
inductive-cases mismatchCases'[simplified pi.inject residual.inject]: $[a \neq b] P \longmapsto$ Rs
inductive-cases sumCases'[simplified pi.inject residual.inject]: $P \oplus Q \longmapsto R s$ inductive-cases parCasesB'[simplified pi.distinct residual.distinct]: $A \| B \longmapsto$ $b<\nu y>\prec A^{\prime}$
inductive-cases parCases $F^{\prime}$ [simplified pi.distinct residual.distinct]: $P \| Q \longmapsto \alpha$ $\prec P^{\prime}$
inductive-cases resCases $B^{\prime}[$ simplified pi.distinct residual.distinct $]:\left\langle\nu x^{\prime}\right\rangle A \longmapsto$ $a<\nu y^{\prime}>\prec A^{\prime}$
inductive-cases resCasesF'[simplified pi.distinct residual.distinct]: $\langle\nu x\rangle A \longmapsto \alpha$ $\prec A^{\prime}$
lemma tauCases:
fixes $P$ :: $p i$
and $\alpha$ :: freeRes
and $\quad P^{\prime}:: p i$
assumes $\tau .(P) \longmapsto \alpha \prec P^{\prime}$
and $\operatorname{Prop}(\tau) P$
shows Prop $\alpha P^{\prime}$
using assms
by (cases rule: tauCases') (auto simp add: pi.inject residual.inject)
lemma inputCases[consumes 1, case-names cInput]:
fixes $a$ :: name
and $x$ :: name
and $P:: p i$
and $\quad P^{\prime}:: p i$
assumes Input: $a<x>. P \longmapsto \alpha \prec P^{\prime}$
and $\quad A: \bigwedge u$. Prop $(a<u>)(P[x::=u])$
shows Prop $\alpha P^{\prime}$
proof -
\{
fix $x P$
assume $a<x>. P \longmapsto \alpha \prec P^{\prime}$
moreover assume ( $x$ ::name) $\sharp \alpha$ and $x \sharp P^{\prime}$ and $x \neq a$
moreover assume $\wedge u$. Prop $(a<u>)(P[x::=u])$
moreover obtain $z:: n a m e$ where $z \neq x$ and $z \sharp P$ and $z \sharp \alpha$ and $z \sharp P^{\prime}$
and $z \neq a$
by (generate-fresh name, auto simp add: fresh-prod)
moreover obtain $z^{\prime}:: n a m e$ where $z^{\prime} \neq x$ and $z^{\prime} \neq z$ and $z^{\prime} \sharp P$ and $z^{\prime} \sharp \alpha$
and $z^{\prime} \sharp P^{\prime}$ and $z^{\prime} \neq a$
by (generate-fresh name, auto simp add: fresh-prod)
ultimately have Prop $\alpha P^{\prime}$
by (cases rule: TransitionsEarly.strong-cases $[$ where $x=x$ and $b=z$ and $x a=z$
and $x b=z$ and $x c=z$ and $x d=z$ and $x e=z$

$$
\text { and } \left.\left.y=z^{\prime} \text { and } y a=z\right\rceil\right)
$$

(auto simp add: pi.inject residual.inject abs-fresh alpha)
\}
note Goal $=$ this
obtain $y$ ::name where $y \sharp P$ and $y \sharp \alpha$ and $y \sharp P^{\prime}$ and $y \neq a$

```
            by(generate-fresh name) (auto simp add: fresh-prod)
    from Input }\langley\sharpP>\mathrm{ have }a<y>.([(x,y)]\cdotP)\longmapsto\alpha\prec\mp@subsup{P}{}{\prime}\mathbf{by}(simp add: alphaIn
put)
    moreover note < y# \alpha>\langley\sharp P
    moreover from A<y\sharpP> have \u. Prop (a<u>) (([(x,y)] P P)[y::=u])
    by(simp add: renaming name-swap)
    ultimately show ?thesis by(rule Goal)
qed
lemma outputCases:
    fixes P :: pi
    and }\alpha\mathrm{ :: freeRes
    and }\mp@subsup{P}{}{\prime}::p
    assumes }a{b}.P\longmapsto\alpha\prec\mp@subsup{P}{}{\prime
    and Prop(OutputR a b)P
    shows Prop \alpha P'
using assms
by(cases rule: outputCases') (auto simp add: pi.inject residual.inject)
lemma zeroTrans[dest]:
    fixes Rs :: residual
    assumes 0}\longmapstoeR
    shows False
using assms
by - (ind-cases 0}\longmapstoeRs
lemma mismatchTrans[dest]:
    fixes a :: name
    and }P\mathrm{ :: pi
    and Rs :: residual
    assumes [a\not=a]P\longmapstoRs
    shows False
using assms
by(erule-tac mismatchCases') auto
lemma matchCases[consumes 1, case-names Match]:
    fixes }a\mathrm{ :: name
    and b :: name
    and }P::p
    and Rs :: residual
    and F}::\mathrm{ name }=>\mathrm{ name }=>\mathrm{ bool
    assumes Trans: [a\frownb]P\longmapstoRs
```

and $\quad c$ Match: $P \longmapsto R s \Longrightarrow F a a$
shows $F a b$
using assms
by (erule-tac matchCases' ${ }^{\prime}$, auto)
lemma mismatchCases[consumes 1, case-names Mismatch]:
fixes $a$ :: name
and $b$ :: name
and $P:: p i$
and $\quad R s::$ residual
and $\quad F::$ name $\Rightarrow$ name $\Rightarrow$ bool
assumes Trans: $[a \neq b] P \longmapsto R s$
and $\quad c$ Match: $\llbracket P \longmapsto R s ; a \neq b \rrbracket \Longrightarrow F a b$
shows $F a b$
using assms
by (erule-tac mismatchCases') auto
lemma sumCases[consumes 1, case-names Sum1 Sum2]:
fixes $P$ :: $p i$
and $\quad Q:: p i$
and $\quad R s::$ residual
assumes Trans: $P \oplus Q \longmapsto R s$
and $\quad c S u m 1: P \longmapsto R s \Longrightarrow F$
and $\quad c S u m 2: ~ Q \longmapsto R s \Longrightarrow F$
shows $F$
using assms
by (erule-tac sumCases') auto
lemma parCasesB[consumes 1, case-names cPar1 cPar2]:
fixes $P$ :: $p i$
and $\quad Q$ :: pi
and $a$ :: name
and $x$ :: name
and $P Q^{\prime}:: p i$
assumes Trans: $P \| Q \longmapsto a<\nu x>\prec P Q^{\prime}$
and $\quad$ icPar1B: $\bigwedge P^{\prime} . \llbracket P \longmapsto a<\nu x>\prec P^{\prime} ; x \sharp Q \rrbracket \Longrightarrow F\left(P^{\prime} \| Q\right)$
and icPar2B: $\bigwedge Q^{\prime} . \llbracket Q \longmapsto a<\nu x>\prec Q^{\prime} ; x \sharp P \rrbracket \Longrightarrow F\left(P \| Q^{\prime}\right)$
shows $F P Q^{\prime}$
proof -
from Trans show ?thesis
proof(induct rule: parCasesB', auto simp add: pi.inject residual.inject)
fix $P^{\prime} y$

```
assume PTrans: \(P \longmapsto a<\nu y>\prec P^{\prime}\)
assume \(y\) Fresh \(Q: y \sharp(Q:: p i)\)
assume absEq: \([x] \cdot P Q^{\prime}=[y] \cdot\left(P^{\prime} \| Q\right)\)
have \(\exists c::\) name. \(c \sharp\left(P^{\prime}, x, y, Q\right)\) by (blast intro: name-exists-fresh)
then obtain \(c\) where \(c\) Fresh \(P^{\prime}: c \sharp P^{\prime}\) and cineqx: \(x \neq c\) and cineqy: \(c \neq y\)
and cFresh \(Q: c \sharp Q\)
    by (force simp add: fresh-prod name-fresh)
```

    from \(c\) Fresh \(P^{\prime}\) PTrans have Trans: \(P \longmapsto a<\nu c>\prec\left([(y, c)] \cdot P^{\prime}\right)\) by \((\operatorname{simp}\)
    add: alphaBoundOutput)
from cFresh $P^{\prime}$ cFresh $Q$ have $c \sharp P^{\prime} \| Q$ by simp
hence $[y] \cdot\left(P^{\prime} \| Q\right)=[c] \cdot\left([(y, c)] \cdot\left(P^{\prime} \| Q\right)\right)$
by (auto simp add: alpha fresh-left calc-atm)
with $y$ Fresh $Q$ cFresh $Q$ have $[y] .\left(P^{\prime} \| Q\right)=[c] \cdot\left(\left([(y, c)] \cdot P^{\prime}\right) \| Q\right)$
by(simp add: name-fresh-fresh)
with cineqx absEq have L1: $P Q^{\prime}=[(x, c)] \cdot\left(\left([(y, c)] \cdot P^{\prime}\right) \| Q\right)$ and L2: $x \sharp$
$\left([(y, c)] \cdot P^{\prime}\right) \| Q$
by (simp add: name-abs-eq)+
from L2 have $x$ Fresh $Q: x \sharp Q$ and $x$ Fresh $P^{\prime}: x \sharp[(y, c)] \cdot P^{\prime}$ by simp+
with $c$ Fresh $Q$ L1 have $L 3: P Q^{\prime}=\left([(x, c)] \cdot[(y, c)] \cdot P^{\prime}\right) \| Q$ by (simp add:
name-fresh-fresh)
from Trans xFresh $P^{\prime}$ have $P \longmapsto a<\nu x>\prec\left([(x, c)] \cdot[(y, c)] \cdot P^{\prime}\right) \mathbf{b y}(\operatorname{simp}$
add: alphaBoundOutput name-swap)
thus ?thesis using $x$ Fresh $Q$ L3
by (blast intro: icPar1B)
next
fix $Q^{\prime} y$
assume $Q$ Trans: $Q \longmapsto a<\nu y>\prec Q^{\prime}$
assume $y$ Fresh $P: y \sharp(P:: p i)$
assume absEq: $[x] \cdot P Q^{\prime}=[y] \cdot\left(P \| Q^{\prime}\right)$
have $\exists c:: n a m e, c \sharp\left(Q^{\prime}, x, y, P\right)$ by (blast intro: name-exists-fresh)
then obtain $c$ where $c$ Fresh $Q^{\prime}: c \sharp Q^{\prime}$ and cineqx: $x \neq c$ and cineqy: $c \neq y$
and cFreshP: $c \sharp P$
by (force simp add: fresh-prod name-fresh)
from $c$ Fresh $Q^{\prime}$ QTrans have Trans: $Q \longmapsto a<\nu c>\prec\left([(y, c)] \cdot Q^{\prime}\right) \mathbf{b y}(\operatorname{simp}$ add: alphaBoundOutput)
from cFresh $Q^{\prime}$ cFreshP have $c \sharp P \| Q^{\prime}$ by simp
hence $[y] \cdot\left(P \| Q^{\prime}\right)=[c] \cdot\left([(y, c)] \cdot\left(P \| Q^{\prime}\right)\right)$
by (auto simp add: alpha fresh-left calc-atm)
with $y$ FreshP cFreshP have $[y] .\left(P \| Q^{\prime}\right)=[c] .\left(P \|\left([(y, c)] \cdot Q^{\prime}\right)\right)$

```
        by(simp add: name-fresh-fresh)
    with cineqx absEq have L1: PQ' = [(x,c)] • (P|([(y,c)] • Q')) and L2: x#
P | ([(y,c)] • Q')
    by(simp add: name-abs-eq)+
```

    from L2 have \(x\) Fresh \(P: x \sharp P\) and \(x F r e s h Q^{\prime}: x \sharp[(y, c)] \cdot Q^{\prime}\) by simp+
    with cFreshP L1 have L3: \(P Q^{\prime}=P \|\left([(x, c)] \cdot[(y, c)] \cdot Q^{\prime}\right) \mathbf{b y}(\) simp add:
    name-fresh-fresh)
from Trans xFresh $Q^{\prime}$ have $Q \longmapsto a<\nu x>\prec\left([(x, c)] \cdot[(y, c)] \cdot Q^{\prime}\right) \mathbf{b y}(\operatorname{simp}$
add: alphaBoundOutput name-swap)
thus ?thesis using $x$ FreshP L3
by (blast intro: icPar2B)
qed
qed
lemma parCasesOutput[consumes 1, case-names Par1 Par2]:
fixes $P$ :: $p i$
and $\quad Q:: p i$
and $a$ :: name
and $b$ :: name
and $P^{\prime}:: p i$
assumes $P \| Q \longmapsto a[b] \prec P Q^{\prime}$
and $\quad \wedge^{\prime} . \llbracket P \longmapsto a[b] \prec P^{\rrbracket} \rrbracket \Longrightarrow F\left(P^{\prime} \| Q\right)$
and $\quad \wedge Q^{\prime} . \llbracket Q \longmapsto a[b] \prec Q^{\rrbracket} \Longrightarrow F\left(P \| Q^{\prime}\right)$
shows $F P Q^{\prime}$
using assms
by(erule-tac parCases $F^{\prime}$, auto simp add: pi.inject residual.inject)
lemma parCasesInput[consumes 1, case-names Par1 Par2]:
fixes $P:: p i$
and $Q$ :: pi
and $a$ :: name
and $b$ :: name
and $P^{\prime}:: p i$
assumes Trans: $P \| Q \longmapsto a<b>\prec P Q^{\prime}$
and $\quad i c P a r 1 F: \wedge P^{\prime} . \llbracket P \longmapsto a<b>\prec P^{\prime} \rrbracket \Longrightarrow F\left(P^{\prime} \| Q\right)$
and $\quad i c P a r 2 F: \bigwedge Q^{\prime} . \llbracket Q \longmapsto a<b>\prec Q \rrbracket \Longrightarrow F\left(P \| Q^{\prime}\right)$
shows $F P Q^{\prime}$
using assms
by (erule-tac parCasesF') (auto simp add: pi.inject residual.inject)
lemma parCasesF[consumes 1, case-names cPar1 cPar2 cComm1 cComm2 cClose1

```
cClose2]:
    fixes P :: pi
    and }Q :: p
    and }\alpha\mathrm{ :: freeRes
    and }\mp@subsup{P}{}{\prime}::p
    and C :: 'a::fs-name
    assumes Trans: P|Q\longmapsto\alpha\precPQ'
    and icPar1F: \bigwedge\mp@subsup{P}{}{\prime}.\llbracketP\longmapsto\alpha\precP\\LongrightarrowF\alpha( 
```




```
(P'| Q')
```



```
(P'| Q )
```



```
x\sharpC\rrbracket\LongrightarrowF(\tau)(<\nux>>(\mp@subsup{P}{}{\prime}|\mp@subsup{Q}{}{\prime}))
    and icClose2: \P ' Q' a x.\llbracketP\longmapstoa<\nux> \prec P'; Q\longmapsto \longmapsto a<x>\prec\prec Q'; x\sharp Q;
x\sharpC\rrbracket\LongrightarrowF(\tau)(<\nux>(\mp@subsup{P}{}{\prime}|\mp@subsup{Q}{}{\prime}))
    shows F \alpha PQ'
proof -
    from Trans show ?thesis
    proof(rule parCasesF', auto)
    fix Pa Pa' Qa \alpha'
    assume Trans': Pa\longmapsto \longmapsto '\precPa'
    assume Eq:P|Q=Pa|Qa
    assume Eq': \alpha\precP\mp@subsup{Q}{}{\prime}=\mp@subsup{\alpha}{}{\prime}\precP\mp@subsup{a}{}{\prime}|Qa
    from Eq have P=Pa and Q=Qa
            by(simp add: pi.inject)+
    moreover with Eq' have \alpha=\mp@subsup{\alpha}{}{\prime}}\mathrm{ and }P\mp@subsup{Q}{}{\prime}=P\mp@subsup{a}{}{\prime}|
            by(simp add: residual.inject)+
    ultimately show ?thesis using icPar1F Trans'
        by simp
    next
    fix Pa Qa Qa' \alpha'
    assume Trans': Qa\longmapsto \longmapsto '`}\precQ\mp@subsup{a}{}{\prime
    assume Eq:P|Q=Pa|Qa
    assume Eq': \alpha\precP\mp@subsup{Q}{}{\prime}=\mp@subsup{\alpha}{}{\prime}\precPa|Q\mp@subsup{a}{}{\prime}
    from Eq have P=Pa and Q = Qa
            by(simp add: pi.inject)+
    moreover with E\mp@subsup{q}{}{\prime}}\mathrm{ have }\alpha=\mp@subsup{\alpha}{}{\prime}\mathrm{ and }P\mp@subsup{Q}{}{\prime}=P|Q\mp@subsup{a}{}{\prime
            by(simp add: residual.inject)+
    ultimately show ?thesis using icPar2F Trans'
```

```
    by \(\operatorname{simp}\)
next
    fix \(P a P a^{\prime} Q a Q a^{\prime} a b\)
    assume TransP: Pa \(\longmapsto a<b>\prec P a^{\prime}\)
    assume Trans \(Q: Q a \longmapsto a[b] \prec Q a^{\prime}\)
    assume \(E q: P\|Q=P a\| Q a\)
    assume \(E q^{\prime}: \alpha \prec P Q^{\prime}=\tau \prec P a^{\prime} \| Q a^{\prime}\)
    from TransP TransQ EqEq' icComm1 show ?thesis
        by (simp add: pi.inject residual.inject)
next
    fix \(P a P a^{\prime} Q a Q a^{\prime}\) a \(b x\)
    assume TransP: Pa \(\longmapsto(a::\) name \()[b] \prec P a^{\prime}\)
    assume Trans \(Q: Q a \longmapsto a<b>\prec Q a^{\prime}\)
    assume \(E q: P\|Q=P a\| Q a\)
    assume \(E q^{\prime}: \alpha \prec P Q^{\prime}=\tau \prec P a^{\prime} \| Q a^{\prime}\)
    from TransP TransQ EqEq' icComm2 show ?thesis
        by (simp add: pi.inject residual.inject)
next
    fix \(P a P a^{\prime} Q a Q a^{\prime}\) a \(x\)
    assume TransP: \(P a \longmapsto a<x>\prec P a^{\prime}\)
    assume Trans \(Q: Q a \longmapsto a<\nu x>\prec Q a^{\prime}\)
    assume \(x\) FreshPa: \(x \sharp P a\)
    assume \(E q: P\|Q=P a\| Q a\)
    assume \(E q^{\prime}: \alpha \prec P Q^{\prime}=\tau \prec<\nu x>\left(P a^{\prime} \| Q a^{\prime}\right)\)
    have \(\exists(c:: n a m e) . c \sharp\left(P a, P a^{\prime}, x, Q a^{\prime}, a, C\right)\)
        by(blast intro: name-exists-fresh)
    then obtain \(c:: n a m e\) where cFreshPa: \(c \sharp P a\) and cFreshPa':c\#Pa' and
cineqy: \(c \neq x\) and cFresh \(Q a^{\prime}: c \sharp Q a^{\prime}\) and \(c F r e s h C: c \sharp C\) and cineqa: \(c \neq a\)
    by (force simp add: fresh-prod name-fresh)
from \(c\) Fresh \(Q a^{\prime}\) have \(L 1: a<\nu x>\prec Q a^{\prime}=a<\nu c>\prec\left([(x, c)] \cdot Q a^{\prime}\right)\)
    by (simp add: alphaBoundOutput)
with cFreshQa' cFreshPa' have \(c \sharp\left(P a^{\prime} \| Q a^{\prime}\right)\)
    by \(\operatorname{simp}\)
then have \(L 4:<\nu x>\left(P a^{\prime} \| Q a^{\prime}\right)=<\nu c>\left(\left([(x, c)] \cdot P a^{\prime}\right) \|\left([(x, c)] \cdot Q a^{\prime}\right)\right)\)
    by (simp add: alphaRes)
have TransP: Pa \(\longmapsto a<c>\prec[(x, c)] \cdot P a^{\prime}\)
proof -
    from \(x\) FreshPa TransP have xineqa: \(x \neq a\) by (force dest: freshAction)
    from TransP have \(([(x, c)] \cdot P a) \longmapsto[(x, c)] \cdot\left(a<x>\prec P a^{\prime}\right)\)
        by(rule TransitionsEarly.eqvt)
    with xineqa xFreshPa cFreshPa cineqa show ?thesis
        by (simp add: name-fresh-fresh name-calc)
qed
```

```
    with TransQ L1 L4 icClose1 Eq Eq' cFreshPa cFreshC show ?thesis
        by(simp add: residual.inject, simp add: pi.inject)
    next
    fix Pa Pa' Qa Qa' a x
    assume TransP: Pa \longmapstoa<\nux>}\precP\mp@subsup{Pa}{}{\prime
    assume TransQ:Qa\longmapstoa<x>}\precQ\mp@subsup{a}{}{\prime
    assume xFreshQa: x\sharpQa
    assume Eq:P|Q=Pa|Qa
    assume Eq': \alpha\precP\mp@subsup{Q}{}{\prime}=\tau\prec<\nux>(P\mp@subsup{a}{}{\prime}|Qa}
    have \exists(c::name).c#(Qa,Pa',x,Qa',a,C)
        by(blast intro: name-exists-fresh)
    then obtain c::name where cFreshQa:c\sharpQa and cFreshPa':c\sharpPa'}\mathrm{ and
cineqy:}c\not=x\mathrm{ and cFreshQa':c#Qa' and cFreshC:c#C and cineqa:c}=
    by(force simp add: fresh-prod name-fresh)
    from cFreshPa' have L1: a<\nux>\prec\precPa'=a<\nuc> \prec([(x,c)] \cdotPa')
    by(simp add: alphaBoundOutput)
    with cFreshQa' cFreshPa' have c\sharp(Pa'|Qa')
        by simp
    then have L4: <\nux> (P\mp@subsup{a}{}{\prime}|Q\mp@subsup{a}{}{\prime})=<\nuc>>(([(x,c)] \cdotPa')|([(x,c)]\cdotQ\mp@subsup{a}{}{\prime}))
    by(simp add: alphaRes)
    have TransQ:Qa\longmapstoa<c> \prec [(x,c)]\cdotQa'
    proof -
        from xFreshQa TransQ have xineqa: }x\not=a\mathrm{ by(force dest: freshAction)
        from TransQ have ([(x,c)]•Qa)\longmapsto \longmapsto [(x,c)]•(a<x>\prec Qa')
            by(rule TransitionsEarly.eqvt)
        with xineqa xFreshQa cFreshQa cineqa show ?thesis
            by(simp add: name-fresh-fresh name-calc)
    qed
    with TransP L1 L4 icClose2 Eq Eq' cFreshQa cFreshC show ?thesis
        by(simp add: residual.inject, simp add: pi.inject)
    qed
qed
lemma resCasesF[consumes 2, case-names Res]:
    fixes x :: name
    and }P::p
    and \alpha :: freeRes
    and }\mp@subsup{P}{}{\prime}:: p
    assumes Trans: <\nux>P\longmapsto \longmapsto\alpha\precRP'
    and xFreshAlpha: x\sharp\alpha
    and rcResF: }\bigwedge\mp@subsup{P}{}{\prime}.P\longmapsto\alpha\prec\mp@subsup{P}{}{\prime}\LongrightarrowF(<\nux>\mp@subsup{P}{}{\prime}
    shows F RP'
proof -
```


## from Trans show ?thesis

proof (induct rule: resCasesF', auto)
fix $P a P a^{\prime} \beta y$
assume PTrans: $P a \longmapsto \beta \prec P a^{\prime}$
assume yFreshBeta: (y::name) $\sharp \beta$
assume TermEq: $\langle\nu x\rangle P=\langle\nu y>P a$
assume ResEq: $\alpha \prec R P^{\prime}=\beta \prec<\nu y>P a^{\prime}$
hence alphaeqbeta: $\alpha=\beta$ and L2: $R P^{\prime}=\left\langle\nu y>P a^{\prime}\right.$ by (simp add: residual.inject)+
have $\exists(c:: n a m e) . c \sharp\left(P a, \alpha, P a^{\prime}, x, y\right) \mathbf{b y}$ (blast intro: name-exists-fresh)
then obtain c::name where cFreshPa: $c \sharp P a$ and cFreshAlpha: $c \sharp \alpha$ and cFreshPa $a^{\prime}: c \sharp P a^{\prime}$ and cineqx: $x \neq c$ and cineqy: $c \neq y$
by (force simp add: fresh-prod name-fresh)
from cFreshPa have $<\nu y>P a=<\nu c>([(y, c)] \cdot P a)$ by (rule alphaRes)
with TermEq cineqx have Peq: $P=[(x, c)] \cdot[(y, c)] \cdot P a$ and $x e q: x \sharp[(y, c)]$ - $P a$
by (simp add: pi.inject name-abs-eq)+
from PTrans have $([(y, c)] \cdot P a) \longmapsto[(y, c)] \cdot\left(\beta \prec P a^{\prime}\right)$ by (rule TransitionsEarly.eqvt)
with yFreshBeta cFreshAlpha alphaeqbeta have PTrans': $([(y, c)] \cdot P a) \longmapsto \alpha$ $\prec\left([(y, c)] \cdot P a^{\prime}\right)$
by(simp add: name-fresh-fresh)
from $P \operatorname{Trans}^{\prime}$ have $([(x, c)] \cdot[(y, c)] \cdot P a) \longmapsto[(x, c)] \cdot\left(\alpha \prec[(y, c)] \cdot P a^{\prime}\right)$
by(rule TransitionsEarly.eqvt)
with $x$ FreshAlpha cFreshAlpha Peq have PTrans ${ }^{\prime \prime}: P \longmapsto \alpha \prec[(x, c)] \cdot[(y, c)]$ - $P a^{\prime}$
by (simp add: name-fresh-fresh)
from PTrans ${ }^{\prime}$ xeq xFreshAlpha have $x e q^{\prime}: x \sharp[(y, c)] \cdot P a^{\prime}$
by (nominal-induct $\alpha$ rule: freeRes.strong-induct)
(auto simp add: fresh-left calc-atm eqvts dest: freshTransition)
from $c$ Fresh $P a^{\prime}$ have $\left\langle\nu y>P a^{\prime}=\langle\nu c\rangle\left([(y, c)] \cdot P a^{\prime}\right)\right.$ by (rule alphaRes $)$
moreover from $x e q^{\prime}$ have $\left.\left.<\nu c\right\rangle\left([(y, c)] \cdot P a^{\prime}\right)=<\nu x\right\rangle([(c, x)] \cdot[(y, c)] \cdot$ $P a^{\prime}$ )
by (rule alphaRes)
ultimately have $R P^{\prime}=\langle\nu x\rangle\left([(x, c)] \cdot[(y, c)] \cdot P a^{\prime}\right)$ using ResEq
by (simp add: residual.inject name-swap)
with PTrans" xFreshAlpha show ?thesis
by (blast intro: rcResF)
qed
qed

```
lemma resCasesB[consumes 2, case-names Open Res]:
```

    fixes \(x\) :: name
    and \(P:: p i\)
    and \(a\) :: name
    and \(y\) :: name
    and \(R P^{\prime}:: p i\)
    assumes Trans: \(<\nu y>P \longmapsto a<\nu x>\prec R P^{\prime}\)
    and xineqy: \(x \neq y\)
    and \(\quad r c\) Open: \(\bigwedge P^{\prime} . \llbracket P \longmapsto(\) OutputR a \(y) \prec P^{\prime} ; a \neq y \rrbracket \Longrightarrow F\left([(x, y)] \cdot P^{\prime}\right)\)
    and \(\quad r c \operatorname{Res} B: \bigwedge P^{\prime} . \llbracket P \longmapsto a<\nu x>\prec P^{\prime} ; y \neq a \rrbracket \Longrightarrow F\left(<\nu y>P^{\prime}\right)\)
    shows \(F R P^{\prime}\)
    proof -
from Trans show ?thesis
proof(induct rule: resCasesB', auto)
fix $P a P a^{\prime}$ aa $b$
assume PTrans: $P a \longmapsto(a a:: n a m e)[b] \prec P a^{\prime}$
assume aaineqb: $a a \neq b$
assume TermEq: $\langle\nu y>P=<\nu b>P a$
assume ResEq: $a<\nu x>\prec R P^{\prime}=a a<\nu b>\prec P a^{\prime}$
have $\exists(c:: n a m e) . c \sharp\left(x, a, a a, y, P a, P a^{\prime}, b\right) \mathbf{b y}$ (blast intro: name-exists-fresh)
then obtain $c$ where cineqx: $c \neq x$ and cFresha: $c \sharp a$ and cineqy: $c \neq y$ and
cineqaa: $c \neq a a$ and cFreshPa: $c \sharp P a$ and $c F r e s h P a^{\prime}: c \sharp P a^{\prime}$ and cineq $b: c \neq b$
by (force simp add: fresh-prod name-fresh)
from $c$ FreshPa have $\langle\nu b\rangle P a=\langle\nu c\rangle([(b, c)] \cdot P a)$ by (rule alphaRes $)$
with cineqy TermEq have PEq: $P=[(y, c)] \cdot[(b, c)] \cdot P a$ and $y F r e s h P a: y \sharp$ $[(b, c)] \cdot P a$
by (simp add: pi.inject name-abs-eq)+
from PTrans have $([(b, c)] \cdot P a) \longmapsto\left([(b, c)] \cdot\left(a a[b] \prec P a^{\prime}\right)\right)$ by $($ rule TransitionsEarly.eqvt)
with aaineqb cineqaa have $L 1:([(b, c)] \cdot P a) \longmapsto a a[c] \prec[(b, c)] \cdot P a^{\prime} \operatorname{by}(\operatorname{simp}$ add: name-calc)
with $y$ FreshPa have yineqaa: $y \neq a a$ by (force dest: freshAction)
from L1 yFreshPa cineqy have yFreshPa': y $\sharp[(b, c)] \cdot P a^{\prime}$ by (force intro: freshTransition)
from $L 1$ have $([(y, c)] \cdot[(b, c)] \cdot P a) \longmapsto[(y, c)] \cdot\left(a a[c] \prec[(b, c)] \cdot P a^{\prime}\right)$
by(rule TransitionsEarly.eqvt)
with cineqaa yineqaa cineqy PEq have PTrans: $P \longmapsto a a[y] \prec[(y, c)] \cdot[(b$, c)] $\cdot P a^{\prime}$
by (simp add: name-calc)
moreover from $c F r e s h P a^{\prime}$ have $a a<\nu b>\prec P a^{\prime}=a a<\nu c>\prec\left([(b, c)] \cdot P a^{\prime}\right)$ by (rule alphaBoundOutput)
with ResEq cineqx have ResEq': $R P^{\prime}=[(x, c)] \cdot[(b, c)] \cdot P a^{\prime}$ and $x \sharp[(b, c)]$ - $P a^{\prime}$

```
    by(simp add: residual.inject name-abs-eq)+
    with xineqy cineqy cineqx yFreshPa' have RP' = [(x,y)] • [(y,c)] \cdot [(b,c)] .
Pa'
            by(subst pt-perm-compose[OF pt-name-inst, OF at-name-inst], simp add:
name-calc name-fresh-fresh)
    moreover from ResEq have a=aa by(simp add: residual.inject)
    ultimately show ?thesis using yineqaa rcOpen
        by blast
    next
    fix Pa Pa' aa xa ya
    assume PTrans: Pa\longmapstoaa<\nuxa>}\precP\mp@subsup{a}{}{\prime
    assume yaFreshaa: (ya::name) \not=aa
    assume yaineqxa: ya }\not=x
    assume EqTrans: <\nuy>P=<\nuya>Pa
    assume EqRes:a<\nux>}\prec~\mp@subsup{P}{}{\prime}=aa<\nuxa>\prec(<\nuya>Pa'
```

    hence aeqaa: \(a=a a \operatorname{by}(s i m p ~ a d d:\) residual.inject)
    with yaFreshaa have yaFresha: ya \(\sharp a\) by simp
    have \(\exists(c:: n a m e) . c \sharp\left(P a^{\prime}, y, x a, y a, x, P a, a a\right)\) by (blast intro: name-exists-fresh)
    then obtain \(c\) where cFreshPa \(a^{\prime}: c \sharp P a^{\prime}\) and cineqy: \(c \neq y\) and cineqxa: \(c \neq\)
    $x a$ and cineqya: $c \neq y a$ and cineqx: $c \neq x$ and cFreshP: $c \sharp P a$ and cFreshaa: $c$
$\sharp a a$
by (force simp add: fresh-prod name-fresh)
have $\exists(d:: n a m e) . d \sharp\left(P a, a, x, P a^{\prime}, c, x a, y a, y\right)$ by(blast intro: name-exists-fresh)
then obtain $d$ where dFreshPa: $d \sharp P a$ and dFresha: $d \sharp a$ and dineqx: $d \neq$
$x$ and $d F r e s h P a^{\prime}: d \sharp P a^{\prime}$ and dineqc: $d \neq c$ and dineqxa: $d \neq x a$ and dineqya: $d$
$\neq y a$ and dineqy: $d \neq y$
by(force simp add: fresh-prod name-fresh)
from dFreshPa have $<\nu y a>P a=<\nu d>([(y a, d)] \cdot P a)$ by (rule alphaRes $)$
with EqTrans dineqy have PEq: $P=[(y, d)] \cdot[(y a, d)] \cdot P a$
and $y$ FreshPa: $y \sharp[(y a, d)] \cdot P a$
by (simp add: pi.inject name-abs-eq)+
from $d$ FreshPa ${ }^{\prime}$ have $L 1:<\nu y a>P a^{\prime}=<\nu d>\left([(y a, d)] \cdot P a^{\prime}\right)$ by (rule alphaRes $)$
from $c F r e s h P a^{\prime}$ dineqc cineqya have $c \sharp<\nu d>\left([(y a, d)] \cdot P a^{\prime}\right)$
by (simp add: name-fresh-abs name-calc name-fresh-left)
hence $a a<\nu x a>\prec\left(<\nu d>\left([(y a, d)] \cdot P a^{\prime}\right)\right)=a a<\nu c>\prec([(x a, c)] \cdot<\nu d>([(y a$,
$\left.d)] \cdot P a^{\prime}\right)$ ) (is ? $L H S=-$ )
by (rule alphaBoundOutput)
with dineqxa dineqc have ?LHS $=a a<\nu c>\prec(<\nu d>([(x a, c)] \cdot[(y a, d)] \cdot$
$\left.P a^{\prime}\right)$ )
by(simp add: name-calc)
with L1 EqRes cineqx dineqc dineqx have
$R P^{\prime} E q: R P^{\prime}=<\nu d>\left([(x, c)] \cdot[(x a, c)] \cdot[(y a, d)] \cdot P a^{\prime}\right)$
and $x F r e s h P a^{\prime}: x \sharp[(x a, c)] \cdot[(y a, d)] \cdot P a^{\prime}$
by (simp add: residual.inject name-abs-eq name-fresh-abs name-calc)+

```
    from PTrans aeqaa have ([(ya,d)] •Pa)\longmapsto \longmapsto[(ya,d)] • (a<\nuxa>\prec \precPa')
    by(blast intro:TransitionsEarly.eqvt)
    with yaineqxa yaFresha dineqxa dFresha have L1:
        ([(ya,d)] • Pa)\longmapstoa<\nuxa> \prec ([(ya,d)] Pa') by(simp add: name-calc
name-fresh-fresh)
    with yFreshPa have yineqa: y\not=a by(force dest: freshAction)
    from dineqc cineqya cFreshPa' have c\sharp[(ya,d)]\cdotP\mp@subsup{a}{}{\prime}
        by(simp add: name-fresh-left name-calc)
    hence a<\nuxa> \prec([(ya,d)] P Pa')=a<\nuc>}\prec([(xa,c)]\cdot[(ya,d)]\cdotPa')(i
?LHS = -)
            by(rule alphaBoundOutput)
    with xFreshPa' have L2: ?LHS =a<\nux>}\prec([(c,x)] \cdot [(xa,c)] \cdot[(ya,d)] .
Pa)
    by(simp add: alphaBoundOutput)
    with L1 PEq have P\longmapsto [(y,d)] • (a<\nux>\prec < [(c,x)] • [(xa,c)] •[(ya,d)] •
Pa
    by(force intro:TransitionsEarly.eqvt simp del: residual.perm)
    with yineqa dFresha xineqy dineqx have Trans: }P\longmapstoa<\nux>\prec ([(y,d)]\cdot[(c
x)] \cdot [(xa,c)] \cdot[(ya,d)] P Pa')
            by(simp add: name-calc name-fresh-fresh)
```

    from L1 L2 yFreshPa xineqy have \(y \sharp[(c, x)] \cdot[(x a, c)] \cdot[(y a, d)] \cdot P a^{\prime}\)
            by (force intro: freshTransition)
    with \(R P^{\prime} E q\) have \(R P^{\prime}=\left\langle\nu y>\left([(y, d)] \cdot[(c, x)] \cdot[(x a, c)] \cdot[(y a, d)] \cdot P a^{\prime}\right)\right.\)
            by (simp add: alphaRes name-swap)
    with Trans yineqa show ?thesis
        by (blast intro: rcResB)
    qed
    qed
lemma bangInduct[consumes 1, case-names Par1B Par1F Par2B Par2F Comm1
Comm2 Close1 Close2 Bang]:
fixes $F::$ ' $a::$ fs-name $\Rightarrow p i \Rightarrow$ residual $\Rightarrow$ bool
and $P:: p i$
and $R s::$ residual
and $C$ :: 'a::fs-name
assumes Trans: $!P \longmapsto R s$
and $\quad c P a r 1 B: \bigwedge a x P^{\prime} C . \llbracket P \longmapsto a<\nu x>\prec P^{\prime} ; x \sharp P ; x \sharp C \rrbracket \Longrightarrow F C(P \|$
$!P)\left(a<\nu x>\prec\left(P^{\prime} \|!P\right)\right)$
and $\quad c$ Par1F: $\bigwedge(\alpha:: f r e e R e s)\left(P^{\prime}:: p i\right) C . \llbracket P \longmapsto \alpha \prec P^{\rrbracket} \Longrightarrow F C(P \|!P)$
$\left(\alpha \prec P^{\prime} \|!P\right)$
and $\quad c$ Par2B: $\bigwedge a x P^{\prime} C . \llbracket!P \longmapsto a<\nu x>\prec P^{\prime} ; x \sharp P ; x \sharp C ; \bigwedge C . F C(!P)$
$\left(a<\nu x>\prec P^{\prime}\right) \rrbracket \Longrightarrow F C(P \|!P)\left(a<\nu x>\prec\left(P \| P^{\prime}\right)\right)$
and $\quad c P a r 2 F: \wedge \alpha P^{\prime} C . \llbracket!P \longmapsto \alpha \prec P^{\prime} ; \wedge C . F C(!P)\left(\alpha \prec P^{\prime}\right) \rrbracket \Longrightarrow F C$
$(P \|!P)\left(\alpha \prec P \| P^{\prime}\right)$
and $\quad c$ Comm1: $\bigwedge a P^{\prime} b P^{\prime \prime} C . \llbracket P \longmapsto a<b>\prec P^{\prime} ;!P \longmapsto($ OutputR a $b) \prec$

```
P'\prime; ^C.FC (!P)((OutputR a b)\prec `'')]\Longrightarrow
                        FC(P|!P)(\tau\prec\mp@subsup{P}{}{\prime}|\mp@subsup{P}{}{\prime\prime})
```



```
P'\prime; \C.FC (!P) (a<b>\prec < ''\prime})\rrbracket
                                    FC(P|!P)(\tau\prec\mp@subsup{P}{}{\prime}|\mp@subsup{P}{}{\prime\prime})
```



```
P;x\sharpC;\bigwedgeC.FC(!P) (a<\nux>\prec \prec ' '\prime)\rrbracket\Longrightarrow
                                    FC(P|!P)(\tau\prec<\nux>(\mp@subsup{P}{}{\prime}|\mp@subsup{P}{}{\prime\prime})}
```



```
P;x\sharpC;^C.FC(!P) (a<x>\prec \prec P'\prime)\rrbracket\Longrightarrow
                                    FC(P|!P)(\tau\prec<\nux>(\mp@subsup{P}{}{\prime}|\mp@subsup{P}{}{\prime\prime})}
    and cBang: \bigwedgeRs C.\llbracketP|!P\longmapstoRs;\bigwedgeC.FC(P|!P)Rs\rrbracket\LongrightarrowFC(!P)
Rs
    shows FC (!P) Rs
proof -
    have }\XYC.\llbracketX\longmapstoY;bangPred P X\rrbracket\LongrightarrowFCX
    proof -
        fix X Y C
        assume X\longmapstoY and bangPred P X
        thus FCX Y
        proof(nominal-induct avoiding:C rule:TransitionsEarly.strong-induct)
            case(Tau Pa)
            thus ?case
                apply -
                by(ind-cases bangPred P (\tau.(Pa)))
    next
        case(Input x a u Pa C)
        thus ?case
            by - (ind-cases bangPred P(a<x>.Pa))
    next
        case(Output a b Pa C)
        thus ?case
            by - (ind-cases bangPred P (a{b}.Pa))
    next
        case(Match Pa Rs b C)
        thus ?case
            by - (ind-cases bangPred P ([b\frownb]Pa))
    next
        case(Mismatch Pa Rs a b C)
        thus ?case
            by - (ind-cases bangPred P ([a\not=b]Pa))
    next
        case(Open Pa a b Pa')
        thus ?case
            by - (ind-cases bangPred P (<\nub>Pa))
        next
        case(Sum1 Pa Rs Q)
        thus ?case
```

```
        by - (ind-cases bangPred P(Pa \oplus Q))
    next
        case(Sum2 Q Rs Pa)
        thus ?case
            by - (ind-cases bangPred P (Pa \oplusQ))
    next
        case(Par1B Pa a x P' Q C)
        thus ?case
            by - (ind-cases bangPred P(Pa|Q), auto simp add: pi.inject cPar1B)
    next
        case(Par1F Pa \alpha P' Q C)
        thus ?case
        by - (ind-cases bangPred P (Pa|Q), auto simp add: pi.inject cPar1F)
    next
        case(Par2B Q a x Q' Pa)
        thus ?case
        by - (ind-cases bangPred P (Pa|Q), auto simp add: pi.inject aux1 cPar2B)
    next
    case(Par2F Q \alpha Q' Pa)
    thus ?case
    by - (ind-cases bangPred P (Pa|Q), auto simp add: pi.inject intro: cPar2F
aux1)
    next
        case(Comm1 Pa a b Pa' Q Q C C)
        thus ?case
        by - (ind-cases bangPred P (Pa|Q), auto simp add: pi.inject intro: cComm1
aux1)
    next
        case(Comm2 Pa a b Pa'Q P'I}C
        thus ?case
        by - (ind-cases bangPred P (Pa|Q), auto simp add: pi.inject intro: cComm2
aux1)
    next
        case(Close1 Pa a x Pa' Q Q" C)
        thus ?case
        by - (ind-cases bangPred P (Pa|Q), auto simp add: pi.inject aux1 cClose1)
    next
        case(Close2 Pa a x Pa' Q Q' C)
        thus ?case
        by - (ind-cases bangPred P (Pa|Q), auto simp add: pi.inject aux1 cClose\mathcal{L})
    next
        case(ResB Pa a x Pa' y)
        thus ?case
            by - (ind-cases bangPred P(<\nuy>Pa))
    next
        case(ResF Pa \alpha Pa' y)
        thus ?case
            by - (ind-cases bangPred P (<\nuy>Pa))
    next
```

```
    case(Bang Pa Rs)
    thus ?case
            by - (ind-cases bangPred P (!Pa), auto simp add: pi.inject intro: aux2
cBang)
    qed
    qed
    with Trans show ?thesis by(force intro: bangPred.aux1)
qed
end
theory Strong-Early-Sim
    imports Early-Semantics Rel
begin
definition strongSimEarly :: pi m (pi\times pi) set => pi=> bool (- w[-] - [80, 80,
80] 80) where
    P\rightsquigarrow[Rel] Q\equiv(\forallay Q'.Q\longmapstoa<\nuy>\prec \prec ''\longrightarrowy\sharpP\longrightarrow(\exists\mp@subsup{P}{}{\prime}.P\longmapstoa<\nuy>
\prec P'^( (P', Q')\inRel))^
                                    (\forall\alpha Q'.Q\longmapsto\alpha\prec Q'\longrightarrow(\exists\mp@subsup{P}{}{\prime}.P\longmapsto\alpha\prec P
lemma monotonic:
    fixes }A::(pi\timespi) se
    and }B::(pi\timespi) se
    and }P::p
    and }\mp@subsup{P}{}{\prime}:: p
    assumes P\rightsquigarrow[A] P'
    and }A\subseteq
    shows P\rightsquigarrow[B] P'
using assms
by(fastforce simp add: strongSimEarly-def)
lemma freshUnit[simp]:
    fixes y :: name
    shows }y\sharp(
by(auto simp add: fresh-def supp-unit)
lemma simCasesCont[consumes 1, case-names Bound Free]:
    fixes P :: pi
    and }Q :: p
    and Rel ::(pi\timespi) set
    and C :: 'a::fs-name
    assumes Eqvt: eqvt Rel
    and Bound: \bigwedgea y Q'.\llbracketQ\longmapstoa<\nuy>\prec Q';y\sharpP;y\sharpQ;y\sharpC\rrbracket\Longrightarrow\exists P'.
P\longmapstoa<\nuy> \prec P'^( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\operatorname{Rel
```

and Free: $\bigwedge \alpha Q^{\prime} . Q \longmapsto \alpha \prec Q^{\prime} \Longrightarrow \exists P^{\prime} . P \longmapsto \alpha \prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in$ Rel
shows $P \rightsquigarrow[$ Rel $] Q$
proof -
from Free show ?thesis
proof (auto simp add: strongSimEarly-def)
fix $Q^{\prime}$ a $y$
assume $y$ Fresh $P:(y::$ name $) \sharp P$
assume Trans: $Q \longmapsto a<\nu y>\prec Q^{\prime}$
have $\exists$ c::name. $c \sharp\left(P, Q^{\prime}, y, Q, C\right)$ by (blast intro: name-exists-fresh)
then obtain $c::$ name where $c F r e s h P: c \sharp P$ and $c F r e s h Q^{\prime}: c \sharp Q^{\prime}$ and $c F r e s h C$ : $c \sharp C$
and cineqy: $c \neq y$ and $c \sharp Q$
by (force simp add: fresh-prod name-fresh)
from Trans cFresh $Q^{\prime}$ have $Q \longmapsto a<\nu c>\prec\left([(y, c)] \cdot Q^{\prime}\right)$ by (simp add: alphaBoundOutput)
hence $\left.\exists P^{\prime} . P \longmapsto a<\nu c\right\rangle \prec P^{\prime} \wedge\left(P^{\prime},[(y, c)] \cdot Q^{\prime}\right) \in$ Rel using $\langle c \sharp P\rangle\langle c \sharp$ $Q\rangle\langle c \sharp C\rangle$ by(rule Bound)
then obtain $P^{\prime}$ where PTrans: $P \longmapsto a<\nu c>\prec P^{\prime}$ and $P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime},[(y\right.$, c)] $\left.\cdot Q^{\prime}\right) \in \operatorname{Rel}$
by blast
from PTrans yFreshP cineqy have $y$ Fresh $P^{\prime}: y \sharp P^{\prime} \mathbf{b y}($ force intro: freshTransition)
with PTrans have $P \longmapsto a<\nu y>\prec\left([(y, c)] \cdot P^{\prime}\right)$ by $(\operatorname{simp}$ add: alphaBound-
Output name-swap)
moreover have $\left([(y, c)] \cdot P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}$ (is ?goal)
proof -
from Eqvt $P^{\prime} \operatorname{Rel} Q^{\prime}$ have $\left([(y, c)] \cdot P^{\prime},[(y, c)] \cdot[(y, c)] \cdot Q^{\prime}\right) \in \operatorname{Rel}$
by (rule eqvtRelI)
with cineqy show ?goal by (simp add: name-calc)
qed
ultimately show $\exists P^{\prime} . P \longmapsto a<\nu y>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in$ Rel by blast qed
qed
lemma simCases[consumes 0, case-names Bound Free]:
fixes $P$ :: $p i$
and $Q$ :: pi
and Rel :: $(p i \times p i)$ set
and $C::{ }^{\prime} a:: f s$-name
assumes Bound: $\bigwedge a y Q^{\prime} . \llbracket Q \longmapsto a<\nu y>\prec Q^{\prime} ; y \sharp P \rrbracket \Longrightarrow \exists P^{\prime} . P \longmapsto a<\nu y>$ $\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}$
and Free: $\bigwedge \alpha Q^{\prime} . Q \longmapsto \alpha \prec Q^{\prime} \Longrightarrow \exists P^{\prime} . P \longmapsto \alpha \prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in$ Rel
shows $P \rightsquigarrow[$ Rel $] Q$
using assms
by (auto simp add: strongSimEarly-def)
lemma elim:
fixes $P$ :: $p i$
and Rel $::(p i \times p i)$ set
and $\quad Q:: p i$
and $a$ :: name
and $x$ :: name
and $\quad Q^{\prime}:: p i$
assumes $P \rightsquigarrow[R e l] Q$
shows $Q \longmapsto a<\nu x>\prec Q^{\prime} \Longrightarrow x \sharp P \Longrightarrow \exists P^{\prime} . P \longmapsto a<\nu x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right)$
$\in$ Rel
and $\quad Q \longmapsto \alpha \prec Q^{\prime} \Longrightarrow \exists P^{\prime} . P \longmapsto \alpha \prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in$ Rel
using assms by (simp add: strongSimEarly-def) +
lemma eqvtI:
fixes $P \quad:: p i$
and $\quad Q \quad:: p i$
and Rel $::(p i \times p i)$ set
and perm :: name prm
assumes Sim: $P \rightsquigarrow[$ Rel $] Q$
and RelRel': Rel $\subseteq$ Rel $^{\prime}$
and EqvtRel': eqvt Rel'
shows $($ perm • P) $\rightsquigarrow[$ Rel $]($ perm $\cdot Q)$
proof (induct rule: simCases)
case(Bound a y $Q^{\prime}$ )
have Trans: $($ perm $\cdot Q) \longmapsto a<\nu y>\prec Q^{\prime}$ by fact
have yFreshP: $y \sharp$ perm $\cdot P$ by fact
from Trans have (rev perm $\cdot($ perm $\cdot Q)) \longmapsto$ rev perm $\bullet\left(a<\nu y>\prec Q^{\prime}\right)$
by (rule TransitionsEarly.eqvt)
hence $Q \longmapsto($ rev perm $\cdot a)<\nu($ rev perm $\cdot y)>\prec\left(\right.$ rev perm $\left.\cdot Q^{\prime}\right)$
by (simp add: name-rev-per)
moreover from $y$ Fresh $P$ have (rev perm • y) $\sharp P$ by (simp add: name-fresh-left) ultimately have $\exists P^{\prime} . P \longmapsto($ rev perm $\cdot a)<\nu($ rev perm $\cdot y)>\prec P^{\prime} \wedge\left(P^{\prime}\right.$, rev perm $\left.\cdot Q^{\prime}\right) \in$ Rel using Sim
by (force intro: elim)
then obtain $P^{\prime}$ where PTrans: $P \longmapsto($ rev perm $\cdot a)<\nu($ rev perm $\cdot y)>\prec P^{\prime}$ and $P^{\prime}$ RelQ $Q^{\prime}:\left(P^{\prime}\right.$, rev perm $\left.\cdot Q^{\prime}\right) \in \operatorname{Rel}$
by blast
from PTrans have $($ perm $\cdot P) \longmapsto$ perm • $(($ rev perm $\cdot a)<\nu($ rev perm $\cdot y)>\prec$ $P^{\prime}$ ) $\mathbf{b y}($ rule TransitionsEarly.eqvt)
hence L1: $($ perm $\cdot P) \longmapsto a<\nu y>\prec\left(\right.$ perm $\left.\cdot P^{\prime}\right) \mathbf{b y}($ simp add: name-per-rev $)$

```
    from P'RelQ' RelRel' have ( }\mp@subsup{P}{}{\prime}\mathrm{ , rev perm • Q') }\mp@subsup{Q}{}{\prime}\mp@subsup{\mathrm{ Rel' by blast}}{}{\prime
    with EqvtRel' have (perm • P', perm • (rev perm • Q )})=\mp@subsup{\mathrm{ Rel }}{}{\prime
    by(rule eqvtRelI)
    hence (perm • P', Q') \in Rel' by(simp add: name-per-rev)
    with L1 show ?case by blast
next
    case(Free \alpha Q')
    have Trans: (perm •Q)\longmapsto\alpha\prec Q' by fact
    from Trans have (rev perm • (perm •Q))\longmapsto rev perm • (\alpha\prec Q')
    by(rule TransitionsEarly.eqvt)
    hence }Q\longmapsto(\mathrm{ rev perm • 人)}\prec(\mathrm{ rev perm • Q')
    by(simp add: name-rev-per)
    with Sim have }\exists\mp@subsup{P}{}{\prime}.P\longmapsto(\mathrm{ rev perm • 人) 々 P'^( (P',(rev perm • Q')) & Rel
    by(force intro: elim)
    then obtain P' where PTrans: P\longmapsto (rev perm • \alpha)\prec < P' and PRel: ( }\mp@subsup{P}{}{\prime},(\mathrm{ rev
perm}\cdot\mp@subsup{Q}{}{\prime}))\inRel by blas
    from PTrans have (perm • P)\longmapsto perm • ((rev perm • \alpha)\prec P') by(rule Tran-
sitionsEarly.eqvt)
    hence L1: (perm •P)\longmapsto\alpha\prec (perm • P') by (simp add: name-per-rev)
    from PRel EqvtRel' RelRel' have ((perm • P'), (perm • (rev perm • Q'))) \inRel'
        by(force intro: eqvtRelI)
    hence ((perm • P'), Q') \inRel' by(simp add: name-per-rev)
    with L1 show ?case by blast
qed
```

lemma reflexive:
fixes $P$ :: $p i$
and Rel $::(p i \times p i)$ set
assumes $I d \subseteq$ Rel
shows $P \rightsquigarrow[R e l] P$
using assms
by (auto simp add: strongSimEarly-def)
lemmas fresh-prod[simp]
lemma transitive:
fixes $P \quad:: p i$
and $\quad Q \quad:: p i$
and $R \quad:: p i$
and Rel :: $(p i \times p i)$ set
and Rel' :: $(p i \times p i)$ set
and Rel" $::(p i \times p i)$ set
assumes $P \operatorname{Sim} Q: P \rightsquigarrow[$ Rel $] Q$
and $\quad$ SSimR: $Q \rightsquigarrow[$ Rel $] R$
and Eqvt': equt Rel"
and Trans: Rel O Rel' $\subseteq$ Rel $^{\prime \prime}$

```
    shows \(P \rightsquigarrow\left[\right.\) Rel \(\left.^{\prime}\right] R\)
proof -
    from Eqvt' show ? thesis
    proof \((\) induct rule: simCasesCont \([\) where \(C=Q])\)
        case(Bound a y \(R^{\prime}\) )
        have \(R\) Trans: \(R \longmapsto a<\nu y>\prec R^{\prime}\) by fact
```

    from \(Q\) SimR RTrans \(« y \sharp Q\) have \(\exists Q^{\prime} . Q \longmapsto a<\nu y>\prec Q^{\prime} \wedge\left(Q^{\prime}, R^{\prime}\right) \in\) Rel \(^{\prime}\)
        by (rule elim)
    then obtain \(Q^{\prime}\) where \(Q\) Trans: \(Q \longmapsto a<\nu y>\prec Q^{\prime}\) and \(Q^{\prime} \operatorname{Rel}^{\prime} R^{\prime}:\left(Q^{\prime}, R^{\prime}\right)\)
    $\in$ Rel' by blast
from PSim $Q$ QTrans $\left\langle y \sharp P\right.$ ) have $\exists P^{\prime} . P \longmapsto a<\nu y>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in$ Rel
by(rule elim)
then obtain $P^{\prime}$ where PTrans: $P \longmapsto a<\nu y>\prec P^{\prime}$ and $P^{\prime}$ Rel $Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in$
Rel by blast
moreover from $P^{\prime}$ RelQ $Q^{\prime} Q^{\prime} \mathrm{Rel}^{\prime} R^{\prime}$ Trans have $\left(P^{\prime}, R^{\prime}\right) \in$ Rel ${ }^{\prime \prime}$ by blast
ultimately show ?case by blast
next
case (Free $\alpha R^{\prime}$ )
have RTrans: $R \longmapsto \alpha \prec R^{\prime}$ by fact
with $Q S i m R$ have $\exists Q^{\prime} . Q \longmapsto \alpha \prec Q^{\prime} \wedge\left(Q^{\prime}, R^{\prime}\right) \in$ Rel' $\mathbf{b y}$ (rule elim)
then obtain $Q^{\prime}$ where $Q$ Trans: $Q \longmapsto \alpha \prec Q^{\prime}$ and $Q^{\prime}$ Rell $^{\prime}:\left(Q^{\prime}, R^{\prime}\right) \in$ Rel $^{\prime}$
by blast
from PSimQ QTrans have $\exists P^{\prime} . P \longmapsto \alpha \prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in$ Rel by (rule elim)
then obtain $P^{\prime}$ where PTrans: $P \longmapsto \alpha \prec P^{\prime}$ and $P^{\prime}$ RelQ $Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in$ Rel
by blast
from $P^{\prime}$ RelQ $Q^{\prime} Q^{\prime}$ RelR' Trans have $\left(P^{\prime}, R^{\prime}\right) \in$ Rel' by blast
with PTrans show $\exists P^{\prime} . P \longmapsto \alpha \prec P^{\prime} \wedge\left(P^{\prime}, R\right) \in$ Rel ${ }^{\prime \prime}$ by blast
qed
qed
end
theory Strong-Early-Bisim
imports Strong-Early-Sim
begin
lemma monoAux: $A \subseteq B \Longrightarrow P \rightsquigarrow[A] Q \longrightarrow P \rightsquigarrow[B] Q$
by (auto intro: Strong-Early-Sim.monotonic)
coinductive-set bisim $::(p i \times p i)$ set
where

$$
\text { step } \llbracket P \rightsquigarrow[\text { bisim }] Q ;(Q, P) \in \operatorname{bisim} \rrbracket \Longrightarrow(P, Q) \in \operatorname{bisim}
$$

monos monoAux
abbreviation strongBisimJudge (infixr $\sim 65$ ) where $P \sim Q \equiv(P, Q) \in$ bisim
lemma bisimCoinductAux[case-names bisim, case-conclusion StrongBisim step, consumes 1]:

```
    assumes \(p:(P, Q) \in X\)
    and step: \(\wedge P Q .(P, Q) \in X \Longrightarrow P \rightsquigarrow[(X \cup \operatorname{bisim})] Q \wedge(Q, P) \in \operatorname{bisim} \cup X\)
```

    shows \(P \sim Q\)
    proof -
have aux: $X \cup$ bisim $=\{(P, Q) .(P, Q) \in X \vee P \sim Q\}$ by blast
from $p$ show ?thesis
by (coinduct, force dest: step simp add: aux)
qed
lemma bisimCoinduct[consumes 1, case-names cSim cSym]:
fixes $P$ :: $p i$
and $\quad Q:: p i$
assumes $(P, Q) \in X$
and $\quad \wedge R S .(R, S) \in X \Longrightarrow R \rightsquigarrow[(X \cup$ bisim $)] S$
and $\quad \bigwedge R S \cdot(R, S) \in X \Longrightarrow(S, R) \in X$
shows $P \sim Q$
using assms
by (coinduct rule: bisimCoinductAux) auto
lemma weak-coinduct[case-names bisim, case-conclusion StrongBisim step, consumes 1]:
assumes $p:(P, Q) \in X$
and step: $\wedge P Q .(P, Q) \in X \Longrightarrow P \rightsquigarrow[X] Q \wedge(Q, P) \in X$
shows $P \sim Q$
using $p$
proof (coinduct rule: bisimCoinductAux)
case (bisim P)
from step $[$ OF this] show ?case using Strong-Early-Sim.monotonic by blast
qed
lemma bisimWeakCoinduct[consumes 1, case-names cSim cSym]:
fixes $P:: p i$
and $\quad Q:: p i$
assumes $(P, Q) \in X$

```
    and }\quad\PQ.(P,Q)\inX\LongrightarrowP\rightsquigarrow[X]
    and}\quad\bigwedgePQ.(P,Q)\inX\Longrightarrow(Q,P)\in
    shows P~Q
using assms
by(coinduct rule: weak-coinduct) auto
lemma monotonic: mono( }\lambdapx1 x2
    \existsPQ. x1 = P^
        x2 = Q ^ P\rightsquigarrow[{(xa,x).p xa x}] Q ^Q\rightsquigarrow[{(xa,x). p xa x}] P)
apply(rule monoI)
by(auto intro:Strong-Early-Sim.monotonic)
lemma bisimE:
    fixes P :: pi
    and }Q:: p
    assumes P ~ Q
    shows P\rightsquigarrow[bisim] Q
    and }Q~
using assms
by(auto intro: bisim.cases)
lemma bisimClosed[eqvt]:
    fixes P :: pi
    and }Q::p
    and p :: name prm
    assumes P~Q
    shows (p\cdotP)~(p\cdotQ)
proof -
    let ?X = {(p\cdotP,p\cdotQ)|(p::name prm) P Q.P~Q}
    from assms have ( }p\cdotP,p\cdotQ)\in?X by aut
    thus ?thesis
    proof(coinduct rule: bisimWeakCoinduct)
        case(cSim P Q)
        moreover {
        fix PQ
        fix p::name prm
        assume P}\rightsquigarrow[bisim]
        moreover have bisim \subseteq?X
            by(auto, rule-tac x=[] in exI) auto
            moreover have eqvt?X
                by(auto simp add: eqvt-def pt2[OF pt-name-inst, THEN sym]) blast
                ultimately have ( }p\cdotP)\rightsquigarrow[?X](p\cdotQ
                    by(rule Strong-Early-Sim.eqvtI)
```

```
    }
    ultimately show ?case by(blast dest: bisimE)
    next
    case(cSym P Q)
    thus ?case by(blast dest: bisimE)
    qed
qed
lemma eqvt[simp]:
    shows eqvt bisim
by(auto simp add: eqvt-def eqvts)
lemma reflexive:
    fixes P :: pi
    shows P ~ P
proof -
    have }(P,P)\inId\mathrm{ by simp
    then show ?thesis
        by(coinduct rule: bisimWeakCoinduct) (auto intro: Strong-Early-Sim.reflexive)
qed
lemma transitive:
    fixes P :: pi
    and }Q::p
    and }R:: p
    assumes PBiSimQ: P~Q
    and QBiSimR:Q~R
    shows P~R
proof -
    let ?X = bisim O bisim
    from assms have ( }P,R)\in?X by blas
    thus ?thesis
    proof(coinduct rule: bisimWeakCoinduct)
        case(cSim P Q)
        moreover {
            fix PQR
            assume P~Q and Q~R
            hence P}\rightsquigarrow[bisim] Q and Q\rightsquigarrow[bisim] 
            by(metis bisimE)+
            moreover from eqvt have eqvt ?X by(auto simp add: eqvtTrans)
            moreover have bisim O bisim}\subseteq?X by aut
            ultimately have P}\rightsquigarrow[?X]
            by(rule Strong-Early-Sim.transitive)
        }
        ultimately show ?case by auto
```

```
    next
        case(cSym P Q)
        thus ?case by(auto dest: bisimE)
    qed
qed
end
theory Strong-Early-Bisim-Subst
    imports Strong-Early-Bisim
begin
abbreviation StrongCongEarlyJudge (infixr ~}\mp@subsup{~}{}{s}65)\mathrm{ where }P\mp@subsup{~}{}{s}Q\equiv(P,Q
(substClosed bisim)
lemma congBisim:
    fixes }P\mathrm{ :: pi
    and }Q::p
    assumes P ~}\mp@subsup{~}{}{s}
    shows P~Q
using assms substClosedSubset by blast
lemma eqvt:
    shows eqvt (substClosed bisim)
by(rule eqvtSubstClosed[OF Strong-Early-Bisim.eqvt])
lemma eqvtI:
    fixes }P::p
    and }Q::p
    and perm :: name prm
    assumes P 舡Q
    shows (perm • P) ~}\mp@subsup{~}{}{s}(\mathrm{ perm • Q)
using assms
by(rule eqvtRelI[OF eqvt])
lemma reflexive:
    fixes P :: pi
    shows P ~s}
by(force simp add: substClosed-def intro: Strong-Early-Bisim.reflexive)
lemma symetric:
    fixes P :: pi
    and }Q:: p
```

```
    assumes }P\mp@subsup{~}{}{s}
    shows }Q\mp@subsup{~}{}{s}
using assms
by(force simp add: substClosed-def intro: Strong-Early-Bisim.bisimE)
lemma transitive:
    fixes }P::p
    and }Q::p
    and }R::p
    assumes P ~}\mp@subsup{~}{}{s}
    and }Q\mp@subsup{~}{}{s}
    shows }P\mp@subsup{~}{}{s}
using assms
by(force simp add: substClosed-def intro: Strong-Early-Bisim.transitive)
lemma partUnfold:
    fixes P :: pi
    and }Q:: p
    and s:: (name }\times\mathrm{ name) list
    assumes P ~}\mp@subsup{~}{}{s}
    shows }P[<s>] \mp@subsup{~}{}{s}Q[<s>
using assms
proof(auto simp add: substClosed-def)
    fix s}\mp@subsup{s}{}{\prime
    assume }\foralls.P[<s>]~Q[<s>
    hence P[<(s@s')>] ~ Q[<(s@ s')>] by blast
    moreover have P[<(s@ s')>] =(P[<s>])[<\mp@subsup{s}{}{\prime}>]
    by(induct s', auto)
    moreover have Q[<(s@s')>]=(Q[<s>])[<\mp@subsup{s}{}{\prime}>]
    by(induct s', auto)
    ultimately show (P[<s>])[<\mp@subsup{s}{}{\prime}\rangle] ~ (Q[<s>])[<\mp@subsup{s}{}{\prime}\rangle]
    by simp
qed
end
theory Strong-Early-Sim-Pres
    imports Strong-Early-Sim
begin
lemma tauPres:
    fixes P :: pi
    and }Q :: p
```

```
    and Rel :: (pi\times pi) set
    assumes PRelQ: (P,Q)\inRel
    shows }\tau.(P)\rightsquigarrow[Rel] \tau.(Q
proof(induct rule: simCases)
    case(Bound a y Q')
    have}\tau.(Q)\longmapstoa<\nuy>\prec Q ' by fac
    hence False by(induct rule: tauCases', auto)
    thus ?case by simp
next
    case(Free \alpha Q ')
    have }\tau.(Q)\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    thus }\exists\mp@subsup{P}{}{\prime}.\tau.(P)\longmapsto\alpha\prec\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mathrm{ Rel
    proof(induct rule: tauCases', auto simp add: pi.inject residual.inject)
    have }\tau.(P)\longmapsto\tau\precP\mathrm{ by(rule TransitionsEarly.Tau)
    with PRelQ show }\exists\mp@subsup{P}{}{\prime}.\tau.(P)\longmapsto\tau\prec\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},Q)\inRel by blas
    qed
qed
lemma inputPres:
    fixes P :: pi
    and }x\mathrm{ :: name
    and Q :: pi
    and a :: name
    and Rel :: (pi\times pi) set
    assumes PRelQ: }\forally.(P[x::=y],Q[x::=y])\in\operatorname{Rel
    and Eqvt: eqvt Rel
    shows }a<x>.P\rightsquigarrow[Rel] a<x>.Q
using Eqvt
proof(induct rule: simCasesCont[where C=(x,a,P,Q)])
    case(Bound b y Q')
    from < }y\sharp(x,a,P,Q)> have y\not=x y\not=ay\sharpPy\sharpQ by simp
    from }\langlea<x>.Q\longmapstob<\nuy>\prec\langle\mp@subsup{Q}{}{\prime}\rangle\langley\not=a\rangle\langley\not=x\rangle\langley\sharpQ\rangle\mathrm{ show ?case
    by(erule-tac inputCases') auto
next
    case(Free \alpha Q ')
    from }\langlea<x>.Q\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime}
    show ?case
    proof(induct rule: inputCases)
        case(cInput u)
        have }a<x>.P\longmapstoa<u>\precP[x::=u] by(rule Input
        moreover from PRelQ have (P[x::=u],Q[x::=u])\in Rel by auto
        ultimately show ?case by blast
    qed
qed
```

```
lemma outputPres:
    fixes }P\mathrm{ :: pi
    and }Q\quad:: p
    and a :: name
    and b :: name
    and Rel :: (pi\times pi) set
    and Rel':: (pi }\timespi) se
    assumes PRelQ: (P,Q)\inRel
    shows a{b}.P\rightsquigarrow[Rel] a{b}.Q
proof(induct rule: simCases)
    case(Bound c y Q')
    have }a{b}.Q\longmapstoc<\nuy>\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    hence False by(induct rule: outputCases', auto)
    thus \exists\mp@subsup{P}{}{\prime}.a{b}.P\longmapstoc<\nuy>\prec \prec ' 
next
    case(Free \alpha Q ')
    have }a{b}.Q\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    thus \exists\mp@subsup{P}{}{\prime}.a{b}.P\longmapsto\alpha\prec敖\wedge( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mathrm{ Rel
    proof(induct rule:outputCases', auto simp add: pi.inject residual.inject)
        have }a{b}.P\longmapstoa[b]\precP\mathbf{by}(rule TransitionsEarly.Output
```



```
    qed
qed
lemma matchPres:
    fixes P :: pi
    and }Q\quad::p
    and a :: name
    and b :: name
    and Rel ::(pi\timespi) set
    and Rel'::(pi\times pi) set
    assumes PSimQ: P\rightsquigarrow[Rel] Q
    and RelRel':Rel \subseteqRel'
    shows [a\frownb]P\rightsquigarrow[Rel'] [a\frownb]Q
proof(induct rule: simCases)
    case(Bound c y Q')
    have (y::name) # [a\frownb]P by fact
    hence yFreshP: y\sharpP by simp
    have [a\frownb]Q\longmapstoc<\nuy>\prec\prec Q' by fact
    thus ?case
    proof(induct rule: matchCases)
    case Match
    have }Q\longmapstoc<\nuy>< \prec Q' by fac
        with PSimQ yFreshP obtain P' where PTrans: P\longmapstoc<\nuy>}\prec\mp@subsup{P}{}{\prime}\mathrm{ and
P'RelQ':}(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\operatorname{Rel
```

```
        by(blast dest: elim)
    from PTrans have [a\frowna]P\longmapstoc<\nuy>}\prec\mp@subsup{P}{}{\prime}\mathbf{by}(rule Early-Semantics.Match
    moreover from P'RelQ' RelRel' have ( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mp@subsup{R}{}{\prime}\mp@subsup{R}{}{\prime}\mathrm{ by blast
    ultimately show ?case by blast
    qed
next
    case(Free \alpha Q ')
    assume [a\frownb] Q\longmapsto\alpha\prec㣌
    thus ?case
    proof(induct rule: matchCases)
        case Match
        have }Q\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
        with PSimQ obtain P' where PTrans: P\longmapsto\alpha \prec P' and P'RelQ':( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime}
ERel
        by(blast dest: elim)
    from PTrans have [ }a\frowna]P\longmapsto\alpha\prec\mp@subsup{P}{}{\prime}\mathrm{ by(rule TransitionsEarly.Match)
    moreover from P'RelQ' RelRel' have ( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mathrm{ Rel' by blast
    ultimately show ?case by blast
    qed
qed
lemma mismatchPres:
    fixes P :: pi
    and }Q :: p
    and a :: name
    and b :: name
    and Rel :: (pi\times pi) set
    and Rel'::(pi\timespi) set
    assumes PSimQ: P}\rightsquigarrow[Rel] Q
    and RelRel':Rel \subseteqRel'
    shows [a\not=b]P\rightsquigarrow[Rel] [a\not=b]Q
proof(cases a = b)
    assume }a=
    thus ?thesis
        by(auto simp add: strongSimEarly-def)
next
    assume aineqb: }a\not=
    show ?thesis
    proof(induct rule: simCases)
    case(Bound c x Q')
    have }x\sharp[a\not=b]P\mathrm{ by fact
    hence xFreshP: x\sharpP by simp
    have [a\not=b]Q\longmapstoc<\nux>}\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    thus ?case
    proof(induct rule: mismatchCases)
```

```
        case Mismatch
        have }Q\longmapstoc<\nux>\prec\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
        with PSimQ xFreshP obtain }\mp@subsup{P}{}{\prime}\mathrm{ where PTrans: P}\longmapstoc<\nux>\prec\mp@subsup{P}{}{\prime
                                    and P'RelQ':}(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\operatorname{Rel
            by(blast dest: elim)
    from PTrans aineqb have [a\not=b]P\longmapstoc<\nux>}\prec\mp@subsup{P}{}{\prime}\mathrm{ by(rule Early-Semantics.Mismatch)
            moreover from P'RelQ' RelRel' have ( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mathrm{ Rel' by blast
            ultimately show ?case by blast
    qed
next
    case(Free \alpha Q ')
    have [a\not=b]Q\longmapsto\alpha\prec Q' by fact
    thus ?case
    proof(induct rule: mismatchCases)
        case Mismatch
        have }Q\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
        with PSimQ obtain }\mp@subsup{P}{}{\prime}\mathrm{ where PTrans: P}\longmapsto\alpha\prec\mp@subsup{P}{}{\prime
                    and PRel: ( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\operatorname{Rel
            by(blast dest: elim)
```



```
            with RelRel' PRel show ?case by blast
    qed
    qed
qed
lemma sumPres:
    fixes P :: pi
    and }Q\quad::p
    and R :: pi
    and Rel :: (pi\timespi) set
    and Rel' :: (pi\timespi) set
    assumes P\rightsquigarrow[Rel] Q
    and C1:Id \subseteqRel'
    and Rel\subseteqRel'
    shows }P\oplusR\rightsquigarrow[Rel']Q\oplus
proof(induct rule: simCases)
    case(Bound a y Q')
    have }y\sharpP\oplusR\mathrm{ by fact
    hence (y::name) #P and }y\sharpR\mathrm{ by simp +
    from <Q \oplusR\longmapstoa<\nuy>\prec < '\ show ?case
    proof(induct rule: sumCases)
    case Sum1
    from \langleP\rightsquigarrow[Rel] Q\rangle\langleQ\longmapstoa<\nuy>\prec \prec Q >\langley\sharpP\rangle obtain P' where PTrans:
P\longmapstoa<\nuy> \prec P' and P'RelQ':( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\inRe
    by(blast dest: elim)
```

```
    from PTrans have }P\oplusR\longmapstoa<\nuy><\mp@subsup{P}{}{\prime}\mathrm{ by(rule Early-Semantics.Sum1)
    moreover from P'RelQ' <Rel \subseteqRel'}>\mathrm{ have ( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\inRel' by blas
    ultimately show ?case by blast
    next
    case Sum2
    from }\langleR\longmapstoa<\nuy>\prec\mp@subsup{Q}{}{\prime}>\mathrm{ have }P\oplusR\longmapstoa<\nuy>\prec\mp@subsup{Q}{}{\prime}\mathrm{ by(rule Early-Semantics.Sum2)
    moreover from C1 have ( }\mp@subsup{Q}{}{\prime},\mp@subsup{Q}{}{\prime})\inRe\mp@subsup{l}{}{\prime}\mathrm{ by auto
    ultimately show ?case by blast
    qed
next
    case(Free \alpha Q ')
```



```
    proof(induct rule: sumCases)
    case Sum1
    have }Q\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    with }\langleP\rightsquigarrow[Rel] Q\rangle\mathrm{ obtain }\mp@subsup{P}{}{\prime}\mathrm{ where PTrans: P}\longmapsto\alpha\prec\mp@subsup{P}{}{\prime}\mathrm{ and }\mp@subsup{P}{}{\prime}RelQ'
(P', Q')\inRel
        by(blast dest: elim)
    from PTrans have P}\oplusR\longmapsto\alpha\prec\mp@subsup{P}{}{\prime}\mathbf{by}(rule TransitionsEarly.Sum1)
    moreover from P'RelQ}\mp@subsup{}{}{\prime}\langleRel\subseteqRel'\rangle have ( (P', Q')\inRel' by blas
    ultimately show ?case by blast
    next
        case Sum2
    from <R\longmapsto\alpha\prec \mp@subsup{Q}{}{\prime}> have P}\oplusR\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime}\mathbf{by}(rule TransitionsEarly.Sum2)
    moreover from C1 have ( }\mp@subsup{Q}{}{\prime},\mp@subsup{Q}{}{\prime})\inRe\mp@subsup{R}{}{\prime}\mathrm{ by blast
    ultimately show ?case by blast
    qed
qed
lemma parCompose:
    fixes P :: pi
    and }Q\quad::p
    and }R\quad::p
    and T :: pi
    and Rel :: (pi\timespi) set
    and Rel' :: (pi\timespi) set
    and Rel" :: (pi\timespi) set
    assumes PSimQ: P
    and }R\operatorname{RSimT: }R\rightsquigarrow[\mathrm{ Rel }]
    and PRelQ: }(P,Q)\in\mathrm{ Rel
    and RRel'T: (R,S)\inRel'
    and Par: }\\mp@subsup{P}{}{\prime}\mp@subsup{Q}{}{\prime}\mp@subsup{R}{}{\prime}\mp@subsup{S}{}{\prime}.\llbracket(\mp@subsup{P}{}{\prime},Q')\in\operatorname{Rel};(\mp@subsup{R}{}{\prime},\mp@subsup{S}{}{\prime})\in\operatorname{Rel}\rrbracket\Longrightarrow(\mp@subsup{P}{}{\prime}|\mp@subsup{R}{}{\prime}
Q'| S') \inRel"
    and Res: \STx. (S,T)\inRel'\prime}\Longrightarrow(<\nux>S,<\nux>T)\inRel'"
    shows P| |}\rightsquigarrow[\mp@subsup{Rel}{}{\prime\prime}]Q|
proof(induct rule: simCases)
```

```
case(Bound a x Q')
have }x\sharpP|R\mathrm{ by fact
hence xFreshP: }x\sharpP\mathrm{ and xFreshR: }x\sharpR\mathrm{ by simp+
have }Q|S\longmapstoa<\nux>\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
thus ?case
proof(induct rule: parCasesB)
    case(cPar1 Q')
    have }Q\longmapstoa<\nux>\prec\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
        with PSimQ xFreshP obtain }\mp@subsup{P}{}{\prime}\mathrm{ where PTrans:P}\longmapstoa<\nux>\prec\prec\mp@subsup{P}{}{\prime}\mathrm{ and
P'RelQ':( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\operatorname{Rel
        by(blast dest: elim)
    from PTrans xFreshR have P|R\longmapstoa<\nux>\prec(\mp@subsup{P}{}{\prime}|R) by(rule Early-Semantics.Par1B)
    moreover from P'RelQ' RRel'T have ( }\mp@subsup{P}{}{\prime}|R,\mp@subsup{Q}{}{\prime}|S)\in\mp@subsup{R}{}{\prime}\mp@subsup{|}{}{\prime\prime}\mathrm{ by(rule Par)
    ultimately show ?case by blast
next
    case(cPar2 S')
    have S\longmapstoa<\nux>}\prec\mp@subsup{S}{}{\prime}\mathrm{ by fact
        with RSimT xFreshR obtain R' where RTrans:R\longmapstoa<\nux>}\prec\mp@subsup{R}{}{\prime}\mathrm{ and
R'Rel'T}\mp@subsup{T}{}{\prime}:(\mp@subsup{R}{}{\prime},\mp@subsup{S}{}{\prime})\inRel
        by(blast dest: elim)
    from RTrans xFreshP have ParTrans: P|R\longmapstoa<\nux>}\prec(P| R') by(rule
Early-Semantics.Par2B)
    moreover from PRelQ R'Rel'T' have (P| R',Q| S') \in Rel'l by(rule Par)
    ultimately show ?case by blast
    qed
next
    case(Free \alpha QT')
    have }Q|S\longmapsto\alpha\precQT' by fac
    thus ?case
    proof(induct rule: parCasesF[of - - - (P,R)])
    case(cPar1 Q')
    have }Q\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    with PSimQ obtain }\mp@subsup{P}{}{\prime}\mathrm{ where PTrans: P}\longmapsto\alpha\prec\mp@subsup{P}{}{\prime}\mathrm{ and PRel: ( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})
Rel
        by(blast dest: elim)
    from PTrans have P|R\longmapsto\alpha\prec P'|R by(rule Early-Semantics.Par1F)
    moreover from PRel RRel'T have ( }\mp@subsup{P}{}{\prime}|R,\mp@subsup{Q}{}{\prime}|S)\inRel'l' by(rule Par
    ultimately show ?case by blast
    next
    case(cPar2 S')
    have S}\longmapsto\alpha\prec\mp@subsup{S}{}{\prime}\mathrm{ by fact
    with RSimT obtain }\mp@subsup{R}{}{\prime}\mathrm{ where RTrans: R}\longmapsto\alpha\prec\mp@subsup{R}{}{\prime}\mathrm{ and RRel: ( }\mp@subsup{R}{}{\prime},\mp@subsup{S}{}{\prime})
Rel'
        by(blast dest: elim)
```

    from \(R\) Trans have \(P\|R \longmapsto \alpha \prec P\| R^{\prime}\) by (rule Early-Semantics.Par2F)
    ```
    moreover from PRelQ RRel have (P| R',Q| S')\inRel" by(rule Par)
    ultimately show ?case by blast
next
    case(cComm1 Q' S'a b)
    have }Q\longmapstoa<b>\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    with PSimQ obtain P' where PTrans: P\longmapstoa<b>\prec P' and P'RelQ':( }\mp@subsup{P}{}{\prime}\mathrm{ ,
Q')\inRel
        by(blast dest: elim)
    have }S\longmapstoa[b]\prec\mp@subsup{S}{}{\prime}\mathrm{ by fact
    with RSimT obtain R' where RTrans: }R\longmapstoa[b]\prec\mp@subsup{R}{}{\prime}\mathrm{ and RRel: ( }\mp@subsup{R}{}{\prime},\mp@subsup{S}{}{\prime})
Rel'
    by(blast dest: elim)
    from PTrans RTrans have P|R\longmapsto\tau\prec P'| R' by(rule Early-Semantics.Comm1)
    moreover from P'RelQ' RRel have ( }\mp@subsup{P}{}{\prime}|\mp@subsup{R}{}{\prime},\mp@subsup{Q}{}{\prime}|\mp@subsup{S}{}{\prime})\inRel'\prime\prime by(rule Par
    ultimately show ?case by blast
next
    case(cComm2 Q' S' a b)
    have }Q\longmapsto(OutputR a b)\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    with PSimQ obtain P' where PTrans: P\longmapstoa[b]\prec \prec ' and PRel: ( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})
Rel
    by(blast dest: elim)
    have }S\longmapstoa<b>\prec\mp@subsup{S}{}{\prime}\mathrm{ by fact
    with RSimT obtain R' where RTrans: }R\longmapstoa<b>\prec\mp@subsup{R}{}{\prime}\mathrm{ and }\mp@subsup{R}{}{\prime}\mp@subsup{R}{el}{\prime}\mp@subsup{|}{}{\prime}\mp@subsup{T}{}{\prime}:(\mp@subsup{R}{}{\prime}
S')}\inRe\mp@subsup{el}{}{\prime
    by(blast dest: elim)
    from PTrans RTrans have P|R\longmapsto~
    moreover from PRel R'Rel'T' have ( }\mp@subsup{P}{}{\prime}|\mp@subsup{R}{}{\prime},\mp@subsup{Q}{}{\prime}|\mp@subsup{S}{}{\prime})\inRel'/' by(rule Par
    ultimately show ?case by blast
next
    case(cClose1 Q' S' a x)
    have }x\sharp(P,R)\mathrm{ by fact
    hence xFreshP: x\sharpP and xFreshR: x\sharpR by simp+
    have }Q\longmapstoa<x>\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    with PSimQ obtain P' where PTrans: P\longmapstoa<x> \prec P' and P'RelQ':(P',
Q')\inRel
    by(blast dest: elim)
    have S\longmapstoa<\nux>}\prec\mp@subsup{S}{}{\prime}\mathrm{ by fact
    with RSimT xFreshR obtain R' where RTrans: R\longmapstoa<\nux> \prec R' and
R'Rel'T
    by(blast dest: elim)
from PTrans RTrans xFreshP have P | R\longmapsto\tau\prec<<\nux>(\mp@subsup{P}{}{\prime}|\mp@subsup{R}{}{\prime})
    by(rule Early-Semantics.Close1)
```

```
    moreover from \(P^{\prime} \operatorname{Rel} Q^{\prime} R^{\prime} \operatorname{Rel}^{\prime} T^{\prime}\) have \(\left(<\nu x>\left(P^{\prime} \| R^{\prime}\right),<\nu x>\left(Q^{\prime} \| S^{\prime}\right)\right) \in\)
Rel \({ }^{\prime \prime}\)
            by(blast intro: Par Res)
    ultimately show ?case by blast
    next
        case (cClose2 \(Q^{\prime} S^{\prime}\) a \(x\) )
    have \(x \sharp(P, R)\) by fact
    hence \(x\) Fresh \(P: x \sharp P\) and \(x\) Fresh \(R\) : \(x \sharp R\) by simp +
    have \(Q \longmapsto a<\nu x>\prec Q^{\prime}\) by fact
    with PSimQ xFreshP obtain \(P^{\prime}\) where PTrans: \(P \longmapsto a<\nu x>\prec P^{\prime}\) and
\(P^{\prime}\) RelQ \(Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in\) Rel
            by(blast dest: elim)
    have \(S \longmapsto a<x>\prec S^{\prime}\) by fact
    with \(R \operatorname{Sim} T\) obtain \(R^{\prime}\) where RTrans: \(R \longmapsto a<x>\prec R^{\prime}\) and \(R^{\prime} R^{R} l^{\prime} T^{\prime}:\left(R^{\prime}\right.\),
\(\left.S^{\prime}\right) \in R e l^{\prime}\)
            by (blast dest: elim)
    from PTrans RTrans xFreshR have \(P \| R \longmapsto \tau \prec<\nu x>\left(P^{\prime} \| R^{\prime}\right)\)
            by(rule Early-Semantics.Close2)
    moreover from \(P^{\prime} \operatorname{Rel} Q^{\prime} R^{\prime} \operatorname{Rel}^{\prime} T^{\prime}\) have \(\left(<\nu x>\left(P^{\prime} \| R^{\prime}\right),<\nu x>\left(Q^{\prime} \| S^{\prime}\right)\right) \in\)
Rel"
            by (blast intro: Par Res)
            ultimately show ?case by blast
        qed
qed
lemma parPres:
    fixes \(P\) :: \(p i\)
    and \(Q\) :: pi
    and \(\quad R\) :: pi
    and \(a\) :: name
    and \(b\) :: name
    and Rel \(::(p i \times p i)\) set
    and Rel' \(::(p i \times p i)\) set
    assumes \(\operatorname{PSimQ:} \quad P \rightsquigarrow[R e l] Q\)
    and PRelQ: \(\quad(P, Q) \in \operatorname{Rel}\)
    and Par: \(\quad \bigwedge S T U .(S, T) \in \operatorname{Rel} \Longrightarrow(S\|U, T\| U) \in \operatorname{Rel}^{\prime}\)
    and Res: \(\quad \bigwedge S T x .(S, T) \in \operatorname{Rel}^{\prime} \Longrightarrow(<\nu x>S,<\nu x>T) \in \operatorname{Rel}^{\prime}\)
    shows \(P \| R \rightsquigarrow\left[\right.\) Rel \(\left.^{\prime}\right] Q \| R\)
proof -
    note \(\operatorname{PSimQ}\)
    moreover have \(R \operatorname{Sim} R: R \rightsquigarrow[I d] R\) by(auto intro: reflexive)
    moreover note PRelQ moreover have \((R, R) \in I d\) by auto
    moreover from Par have \(\bigwedge P Q R T . \llbracket(P, Q) \in \operatorname{Rel} ;(R, T) \in I d \rrbracket \Longrightarrow(P \|\)
\(R, Q \| T) \in\) Rel \(^{\prime}\)
```

```
    by auto
    ultimately show ?thesis using Res by(rule parCompose)
qed
lemma resPres:
    fixes P :: pi
    and Q :: pi
    and Rel :: (pi\times pi) set
    and x :: name
    and Rel':: (pi \times pi) set
    assumes PSimQ: P}\rightsquigarrow[Rel]
    and ResSet: \bigwedge(R::pi) (S::pi) (y::name). (R,S)\inRel\Longrightarrow(<\nuy>R,<\nuy>S)
Rel'
    and RelRel': Rel }\subseteqRel'
    and EqvtRel: eqvt Rel
    and EqvtRel': eqvt Rel'
    shows <\nux>P}\rightsquigarrow[REl`]<\nux>Q
proof -
    from EqvtRel' show ?thesis
    proof(induct rule: simCasesCont[where C=(P,x)])
        case(Bound a y Q')
        have Trans: <\nux>>Q\longmapstoa<\nuy>}\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
        have }y\sharp(P,x)\mathrm{ by fact
        hence yineqx:}y\not=x\mathrm{ and yFreshP: y }\sharp(P::pi) by simp
        from Trans yineqx show ?case
        proof(induct rule: resCasesB)
            case(Open Q')
            have QTrans: Q\longmapsto(a::name)[x]\prec Q' by fact
```



```
Q')\inRel
                by(blast dest: elim)
            have <\nux>P\longmapsto}\longmapstoa<\nuy>\prec([(y,x)]\cdot\mp@subsup{P}{}{\prime}
            proof -
                have aineqx: a\not=x by fact
            with PTrans have <\nux>P\longmapsto\longmapstoa<\nux>>\prec P' by(rule TransitionsEarly.Open)
                    moreover have }a<\nux>\prec\mp@subsup{P}{}{\prime}=a<\nuy>\prec([(y,x)]\cdot\mp@subsup{P}{}{\prime}
                    proof -
                    from PTrans yFreshP have yFreshP': y# '' by(force intro: freshTransition)
                    thus ?thesis by(simp add: alphaBoundOutput name-swap)
                qed
                ultimately show ?thesis by simp
            qed
            moreover from EqvtRel P'RelQ' RelRel' have ([(y,x)] • P',[(y,x)] \cdot Q')\in
Rel'
            by(blast intro: eqvtRelI)
            ultimately show ?case by blast
```


## next

case(Res $\left.Q^{\prime}\right)$
have $Q$ Trans: $Q \longmapsto a<\nu y>\prec Q^{\prime}$ by fact
with PSimQ yFreshP obtain $P^{\prime}$ where PTrans: $P \longmapsto a<\nu y>\prec P^{\prime}$ and $P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in$ Rel
by(blast dest: elim)
have xineqa: $x \neq a$ by fact
with PTrans yineqx have ResTrans: $<\nu x>P \longmapsto a<\nu y>\prec\left(<\nu x>P^{\prime}\right)$ by (blast intro: ResB)
moreover from $P^{\prime} \operatorname{Rel} Q^{\prime}$ have $\left(\left(<\nu x>P^{\prime}\right),\left(<\nu x>Q^{\prime}\right)\right) \in R e l^{\prime}$ by (rule ResSet)
ultimately show $\exists P^{\prime} .<\nu x>P \longmapsto a<\nu y>\prec P^{\prime} \wedge\left(P^{\prime},\left(<\nu x>Q^{\prime}\right)\right) \in \operatorname{Rel}^{\prime}$ by blast
qed
next
case (Free $\alpha Q^{\prime}$ )
have Trans: $<\nu x>Q \longmapsto \alpha \prec Q^{\prime}$ by fact
have $\exists c:: n a m e . c \sharp\left(P, Q, Q^{\prime}, \alpha\right)$ by (blast intro: name-exists-fresh)
then obtain c::name where cFresh $Q: c \sharp Q$ and cFreshAlpha: $c \sharp \alpha$ and cFresh $Q^{\prime}: c \sharp Q^{\prime}$ and cFreshP: $c \sharp P$
by(force simp add: fresh-prod)
from $c$ Fresh $P$ have $<\nu x>P=<\nu c>([(x, c)] \cdot P)$ by (simp add: alphaRes)
moreover have $\exists P^{\prime} .<\nu c>([(x, c)] \cdot P) \longmapsto \alpha \prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in$ Rel $^{\prime}$
proof -
from Trans cFresh $Q$ have $<\nu c>([(x, c)] \cdot Q) \longmapsto \alpha \prec Q^{\prime}$ by $(\operatorname{simp}$ add: alphaRes)
moreover from EqvtRel PSimQ have $([(x, c)] \cdot P) \rightsquigarrow[\operatorname{Rel}]([(x, c)] \cdot Q)$ by(blast intro: eqvtI)
ultimately show ?thesis using cFreshAlpha
apply -
apply (erule resCasesF)
apply auto
by(blast intro: ResF ResSet dest: elim)
qed
ultimately show $\exists P^{\prime} .\left\langle\nu x>P \longmapsto \alpha \prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in\right.$ Rel' by auto qed
qed
lemma resChainI:
fixes $P$ :: $p i$
and $Q$ :: pi
and Rel :: $(p i \times p i)$ set
and lst :: name list
assumes eqvtRel: eqvt Rel

```
    and Res: }\RSx.(R,S)\inRel\Longrightarrow(<\nux>>R,<\nux>S)\inRe
    and PRelQ: P\rightsquigarrow[Rel] Q
    shows (resChain lst) P\rightsquigarrow[Rel] (resChain lst) Q
proof -
    show ?thesis
    proof(induct lst)
        from PRelQ show resChain [] P\rightsquigarrow[Rel] resChain [] Q by simp
    next
        fix a lst
    assume IH:(resChain lst P) }\rightsquigarrow[\mathrm{ Rel ] (resChain lst Q)
    moreover from Res have }\PQa.(P,Q)\inRel\Longrightarrow(<\nua>P,<\nua>Q)
Rel
            by simp
    moreover have Rel \subseteqRel by simp
    ultimately have <\nua>(resChain lst P)}\rightsquigarrow[Rel]<\nua>(resChain lst Q) usin
eqvtRel
            by(rule-tac resPres)
    thus resChain (a# lst) P\rightsquigarrow[Rel] resChain (a# lst)Q
            by simp
    qed
qed
lemma bangPres:
    fixes P :: pi
    and }Q :: p
    and Rel :: (pi\times pi) set
    assumes PRelQ: }\quad(P,Q)\in\mathrm{ Rel
    and Sim: }\quad\bigwedgeRS.(R,S)\inRel\LongrightarrowR\rightsquigarrow[Rel]
    and eqvtRel: eqvt Rel
    shows !P\rightsquigarrow[bangRel Rel]!Q
proof -
    let ?Sim = \lambdaP Rs. ( }\forall\textrm{a}x\mp@subsup{Q}{}{\prime}.Rs=a<\nux>\prec\prec\mp@subsup{Q}{}{\prime}\longrightarrowx\sharpP\longrightarrow(\exists\mp@subsup{P}{}{\prime}.
\longmapsto a < \nu x > < ~ P ^ { \prime } \wedge ( P ^ { \prime } , Q ^ { \prime } ) \in \text { bangRel Rel))} \wedge
                                    (\forall\alpha Q'.Rs=\alpha\prec林\longrightarrow(\exists\mp@subsup{P}{}{\prime}.P\longmapsto\alpha\prec 在'^( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})
bangRel Rel))
    from eqvtRel have EqvtBangRel: eqvt(bangRel Rel) by(rule eqvtBangRel)
    {
        fix Pa Rs
        assume !Q\longmapstoRs and (Pa,!Q)\in bangRel Rel
        hence ?Sim Pa Rs using PRelQ
        proof(nominal-induct avoiding: Pa P rule: bangInduct)
            case(Par1B a x Q' Pa P)
            have QTrans: Q\longmapstoa<\nux>}\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
            have (Pa,Q|!Q)\in bangRel Rel and x #Pa by fact+
            thus ?Sim Pa (a<\nux>\prec\prec(\mp@subsup{Q}{}{\prime}||!Q))
```

```
proof(induct rule: BRParCases)
    case(BRPar P R)
    have PRelQ: (P,Q)\in Rel by fact
    have PBRQ:(R,!Q) \in bangRel Rel by fact
    have }x\sharpP|R\mathrm{ by fact
    hence xFreshP: x\sharpP and xFreshR: x }\sharpR\mathrm{ by simp+
    show ?case
    proof(auto simp add: residual.inject alpha')
        from PRelQ have P}\rightsquigarrow[Rel]Q by(rule Sim
    with QTrans xFreshP obtain P' where PTrans: P\longmapstoa<\nux> \prec P' and
P
        by(blast dest: elim)
```

        from PTrans xFresh \(R\) have \(P \| R \longmapsto a<\nu x>\prec\left(P^{\prime} \| R\right)\)
            by(force intro: Early-Semantics.Par1B)
            moreover from \(P^{\prime} \operatorname{Rel} Q^{\prime} P B R Q\) have \(\left(P^{\prime}\left\|R, Q^{\prime}\right\|!Q\right) \in\) bangRel Rel
    by(rule Rel.BRPar)
ultimately show $\exists P^{\prime} . P \| R \longmapsto a<\nu x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime} \|!Q\right) \in$ bangRel
Rel by blast
next
fix $y$
assume $(y:: n a m e) \sharp Q^{\prime}$ and $y \sharp P$ and $y \sharp R$ and $y \sharp Q$
from $Q$ Trans $\left\langle y \sharp Q^{\prime}\right\rangle$ have $Q \longmapsto a<\nu y>\prec\left([(x, y)] \cdot Q^{\prime}\right)$
by (simp add: alphaBoundOutput)
moreover from $\operatorname{PRelQ}$ have $P \rightsquigarrow[$ Rel $] Q$ by (rule Sim)
ultimately obtain $P^{\prime}$ where PTrans: $P \longmapsto a<\nu y>\prec P^{\prime}$ and $P^{\prime}$ RelQ':
$\left(P^{\prime},[(x, y)] \cdot Q^{\prime}\right) \in \operatorname{Rel}$
using $\langle y \sharp P\rangle$
by(blast dest: elim)
from PTrans $\langle y \sharp R\rangle$ have $P \| R \longmapsto a<\nu y>\prec\left(P^{\prime} \| R\right)$ by (force intro:
Early-Semantics.Par1B)
moreover from $P^{\prime} \operatorname{Rel} Q^{\prime} P B R Q$ have $\left(P^{\prime}\left\|R,\left([(x, y)] \cdot Q^{\prime}\right)\right\|!Q\right) \in$
bangRel Rel by(rule Rel.BRPar)
with $\langle x \sharp Q\rangle\langle y \sharp Q\rangle$ have $\left(P^{\prime}\left\|R,\left([(y, x)] \cdot Q^{\prime}\right)\right\|!([(y, x)] \cdot Q)\right) \in$
bangRel Rel
by (simp add: name-fresh-fresh name-swap)
ultimately show $\exists P^{\prime} . P \| R \longmapsto a<\nu y>\prec P^{\prime} \wedge\left(P^{\prime},\left([(y, x)] \cdot Q^{\prime}\right) \|\right.$
$!([(y, x)] \cdot Q)) \in$ bangRel Rel
by blast
qed
qed
next
case(Par1F $\left.\alpha Q^{\prime} P a P\right)$
have $Q$ Trans: $Q \longmapsto \alpha \prec Q^{\prime}$ by fact
have $(P a, Q \|!Q) \in$ bangRel Rel by fact
thus ?case
proof (induct rule: BRParCases)
case(BRPar P R)

```
            have PRelQ: (P,Q)\inRel and BR: (R,!Q)\in bangRel Rel by fact+
                    show ?case
                    proof(auto simp add: residual.inject)
                            from PRelQ have P\rightsquigarrow[Rel] Q by(rule Sim)
                            with QTrans obtain P' where PTrans: P\longmapsto\alpha 
Q')}\in\operatorname{Rel
            by(blast dest: elim)
            from PTrans have P|R\longmapsto\alpha\prec |'| | by(rule TransitionsEarly.Par1F)
            moreover from RRel BR have ( }\mp@subsup{P}{}{\prime}|R,\mp@subsup{Q}{}{\prime}|!Q)\in\mathrm{ bangRel Rel by(rule
Rel.BRPar)
            ultimately show }\exists\mp@subsup{P}{}{\prime}.P|R\longmapsto\alpha\prec\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime}|!Q)\in\mathrm{ bangRel Rel
by blast
            qed
            qed
    next
            case(Par2B a x Q' Pa P)
            hence IH: \bigwedgePa. (Pa,!Q) \in bangRel Rel \Longrightarrow?Sim Pa (a<\nux>}\prec\mp@subsup{Q}{}{\prime})\mathrm{ by
simp
            have (Pa,Q|!Q)\in bangRel Rel and x #Pa by fact+
            thus ?Sim Pa (a<\nux> \prec (Q| Q '))
            proof(induct rule: BRParCases)
                    case(BRPar P R)
                    have PRelQ: (P,Q)\inRel and RBRQ: (R,!Q) \in bangRel Rel by fact+
                    have }x\sharpP|R\mathrm{ by fact
                    hence xFreshP: x\sharpP and xFreshR: x #R by simp+
                    from EqvtBangRel show ?Sim (P|R) (a<\nux>}\prec(Q|\mp@subsup{Q}{}{\prime})
                    proof(auto simp add: residual.inject alpha')
                    from RBRQ have ?Sim R (a<\nux>}\prec\mp@subsup{Q}{}{\prime})\mathbf{by}(\mathrm{ rule IH)
                            with xFreshR obtain R' where RTrans: R\longmapstoa<\nux>}\prec\mp@subsup{R}{}{\prime}\mathrm{ and }\mp@subsup{R}{}{\prime}BR\mp@subsup{Q}{}{\prime}
( }\mp@subsup{R}{}{\prime},\mp@subsup{Q}{}{\prime})\in(\mathrm{ bangRel Rel)
                by(metis elim)
                            from RTrans xFreshP have P|R\longmapstoa<\nux>}\prec(P|\mp@subsup{R}{}{\prime})\mathrm{ by(auto intro:
Early-Semantics.Par2B)
            moreover from PRelQ R'BRQ' have (P| |
by(rule Rel.BRPar)
                            ultimately show }\exists\mp@subsup{P}{}{\prime}.P|R\longmapstoa<\nux>\prec\prec\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},Q|\mp@subsup{Q}{}{\prime})\in\mathrm{ bangRel
Rel by blast
            next
                    fix }
                            assume (y::name) \sharpQ and y\sharp Q' and }y\sharpP\mathrm{ and }y\sharp
                            from RBRQ have ?Sim R (a<\nux><\prec Q') by(rule IH)
                with <y\sharp Q'> have ?Sim R (a<\nuy> \prec ([(x,y)] • Q')) by(simp add:
alphaBoundOutput)
    with }\langley\sharpR\rangle\mathrm{ obtain }\mp@subsup{R}{}{\prime}\mathrm{ where RTrans: }R\longmapstoa<\nuy>\prec \prec R' and R'BR\mp@subsup{Q}{}{\prime}
( }\mp@subsup{R}{}{\prime},([(x,y)]\cdot\mp@subsup{Q}{}{\prime}))\in(\mathrm{ bangRel Rel)
            by(metis elim)
                            from RTrans }\langley\sharpP>\mathrm{ have }P|R\longmapstoa<\nuy> \prec (P| R') by(auto intro
```

Early-Semantics.Par2B)
moreover from $P R e l Q R^{\prime} B R Q^{\prime}$ have $\left(P\left\|R^{\prime}, Q\right\|\left([(x, y)] \cdot Q^{\prime}\right)\right) \in$ (bangRel Rel) by(rule Rel.BRPar)
with $\langle y \sharp Q\rangle\langle x \sharp Q\rangle$ have $\left(P\left\|R^{\prime},([(y, x)] \cdot Q)\right\|\left([(y, x)] \cdot Q^{\prime}\right)\right) \in$ (bangRel Rel)
by (simp add: name-swap name-fresh-fresh)
ultimately show $\exists P^{\prime} . P \| R \longmapsto a<\nu y>\prec P^{\prime} \wedge\left(P^{\prime},([(y, x)] \cdot Q) \|\right.$
$\left.\left([(y, x)] \cdot Q^{\prime}\right)\right) \in$ bangRel Rel by blast
qed
qed
next
case(Par2F $\alpha Q^{\prime}$ Pa P)
hence $I H: \bigwedge P a .(P a,!Q) \in$ bangRel Rel $\Longrightarrow$ ? Sim Pa $\left(\alpha \prec Q^{\prime}\right)$ by simp have $(P a, Q \|!Q) \in$ bangRel Rel by fact
thus ?case
proof (induct rule: BRParCases)
case (BRPar P R)
have PRelQ: $(P, Q) \in$ Rel and $R B R Q:(R,!Q) \in$ bangRel Rel by fact + show ?case
proof (auto simp add: residual.inject)
from $R B R Q$ IH have $\exists R^{\prime} . R \longmapsto \alpha \prec R^{\prime} \wedge\left(R^{\prime}, Q^{\prime}\right) \in$ bangRel Rel
by (metis elim)
then obtain $R^{\prime}$ where RTrans: $R \longmapsto \alpha \prec R^{\prime}$ and $R^{\prime} \operatorname{Rel} Q^{\prime}:\left(R^{\prime}, Q^{\prime}\right) \in$ bangRel Rel
by blast
from RTrans have $P\|R \longmapsto \alpha \prec P\| R^{\prime} \mathbf{b y}$ (rule TransitionsEarly.Par2F)
moreover from $P R e l Q R^{\prime} R e l Q^{\prime}$ have $\left(P\left\|R^{\prime}, Q\right\| Q^{\prime}\right) \in$ bangRel Rel by(rule Rel.BRPar)
ultimately show $\exists P^{\prime} . P \| R \longmapsto \alpha \prec P^{\prime} \wedge\left(P^{\prime}, Q \| Q^{\prime}\right) \in$ bangRel Rel by blast
qed
qed
next
case (Comm1 a $\left.Q^{\prime} b Q^{\prime \prime} P a P\right)$
hence $I H: \wedge P a .(P a,!Q) \in$ bangRel Rel $\Longrightarrow$ ?Sim Pa $\left(a[b] \prec Q^{\prime \prime}\right)$ by simp
have $Q$ Trans: $Q \longmapsto a<b>\prec Q^{\prime}$ by fact
have $(P a, Q \|!Q) \in$ bangRel Rel by fact
thus ?case
proof(induct rule: BRParCases)
case (BRPar P R)
have PRelQ: $(P, Q) \in$ Rel and $R B R Q:(R,!Q) \in$ bangRel Rel by fact +
show ? case
proof (auto simp add: residual.inject)
from PRelQ have $P \rightsquigarrow[$ Rel $] Q$ by (rule Sim)
with $Q$ Trans obtain $P^{\prime}$ where PTrans: $P \longmapsto a<b>\prec P^{\prime}$ and $P^{\prime}$ RelQ': $\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}$
by(blast dest: elim)
from $I H R B R Q$ have RTrans: $\exists R^{\prime} . R \longmapsto a[b] \prec R^{\prime} \wedge\left(R^{\prime}, Q^{\prime \prime}\right) \in$ bangRel Rel
by (metis elim)
then obtain $R^{\prime}$ where $R$ Trans: $R \longmapsto a[b] \prec R^{\prime}$ and $R^{\prime} \operatorname{Rel} Q^{\prime \prime}:\left(R^{\prime}, Q^{\prime \prime}\right)$ $\in$ bangRel Rel
by blast
from PTrans RTrans have $P\left\|R \longmapsto \tau \prec P^{\prime}\right\| R^{\prime}$ by (rule TransitionsEarly.Comm1)
moreover from $P^{\prime} \operatorname{Rel} Q^{\prime} R^{\prime} \operatorname{Rel} Q^{\prime \prime}$ have $\left(P^{\prime}\left\|R^{\prime}, Q^{\prime}\right\| Q^{\prime \prime}\right) \in$ bangRel Rel by (rule Rel.BRPar)
ultimately show $\exists P^{\prime} . P \| R \longmapsto \tau \prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime} \| Q^{\prime \prime}\right) \in$ bangRel Rel by blast

## qed

qed
next
case(Comm2 a b $\left.Q^{\prime} Q^{\prime \prime}\right)$
hence $I H: \bigwedge P a .(P a,!Q) \in$ bangRel Rel $\Longrightarrow$ ?Sim Pa $\left(a<b>\prec Q^{\prime \prime}\right)$ by simp
have $Q$ Trans: $Q \longmapsto a[b] \prec Q^{\prime}$ by fact
have $(P a, Q \|!Q) \in$ bangRel Rel by fact
thus ?case
proof (induct rule: BRParCases)
case (BRPar P R)
have PRelQ: $(P, Q) \in$ Rel and $R B R Q:(R,!Q) \in$ bangRel Rel by fact+ show ? case
proof (auto simp add: residual.inject)
from $\operatorname{PRelQ}$ have $P \rightsquigarrow[$ Rel $] Q$ by (rule Sim)
with QTrans obtain $P^{\prime}$ where PTrans: $P \longmapsto a[b] \prec P^{\prime}$ and $P^{\prime}$ RelQ': $\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}$
by (blast dest: elim)
from $I H R B R Q$ have $R$ Trans: $\exists R^{\prime} . R \longmapsto a<b>\prec R^{\prime} \wedge\left(R^{\prime}, Q^{\prime \prime}\right) \in$ bangRel Rel by (metis elim)
then obtain $R^{\prime}$ where RTrans: $R \longmapsto a<b>\prec R^{\prime}$ and $R^{\prime} \operatorname{Rel} Q^{\prime \prime}:\left(R^{\prime}\right.$, $\left.Q^{\prime \prime}\right) \in$ bangRel Rel
by blast
from PTrans RTrans have $P\left\|R \longmapsto \tau \prec P^{\prime}\right\| R^{\prime}$ by(rule TransitionsEarly.Comm2)
moreover from $P^{\prime} \operatorname{Rel} Q^{\prime} R^{\prime} \operatorname{Rel} Q^{\prime \prime}$ have $\left(P^{\prime}\left\|R^{\prime}, Q^{\prime}\right\| Q^{\prime \prime}\right) \in$ bangRel Rel by(rule Rel.BRPar)
ultimately show $\exists P^{\prime} . P \| R \longmapsto \tau \prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime} \| Q^{\prime \prime}\right) \in$ bangRel Rel
by blast
qed
qed
next
case(Close1 a x $Q^{\prime} Q^{\prime \prime}$ Pa P)
hence $I H: \wedge P a .(P a,!Q) \in$ bangRel Rel $\longrightarrow$ ?Sim $P a\left(a<\nu x>\prec Q^{\prime \prime}\right)$ by
simp
have $Q$ Trans: $Q \longmapsto a<x>\prec Q^{\prime}$ by fact
have $x$ Fresh $Q$ : $x \sharp Q$ by fact
have $(P a, Q \|!Q) \in$ bangRel Rel by fact
moreover have $x$ FreshPa: $x \sharp P a$ by fact
ultimately show ?case
proof (induct rule: BRParCases)
case (BRPar P R)
have PRelQ: $(P, Q) \in$ Rel and $R B R Q:(R,!Q) \in$ bangRel Rel by fact +
have $x \sharp P \| R$ by fact
hence $x$ Fresh $P: x \sharp P$ and $x F r e s h R$ : $x \sharp R$ by simp+
show ?case
proof (auto simp add: residual.inject)
from PRelQ have $P \rightsquigarrow[$ Rel $] Q$ by (rule Sim)
with $Q$ Trans xFreshP obtain $P^{\prime}$ where PTrans: $P \longmapsto a<x>\prec P^{\prime}$ and
$P^{\prime}$ RelQ $Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}$
by(blast dest: elim)
from $R B R Q$ xFresh $R$ IH have $\exists R^{\prime} . R \longmapsto a<\nu x>\prec R^{\prime} \wedge\left(R^{\prime}, Q^{\prime \prime}\right) \in$ bangRel Rel by (metis elim)
then obtain $R^{\prime}$ where RTrans: $R \longmapsto a<\nu x>\prec R^{\prime}$ and $R^{\prime} \operatorname{Rel}^{\prime \prime}:\left(R^{\prime}\right.$, $\left.Q^{\prime \prime}\right) \in$ bangRel Rel by blast
from PTrans RTrans xFreshP have $P \| R \longmapsto \tau \prec<\nu x>\left(P^{\prime} \| R^{\prime}\right)$ by(rule Early-Semantics.Close1)
moreover from $P^{\prime} \operatorname{Rel} Q^{\prime} R^{\prime} R e l Q^{\prime \prime}$ have $\left(<\nu x>\left(P^{\prime} \| R^{\prime}\right),<\nu x>\left(Q^{\prime} \|\right.\right.$ $\left.\left.Q^{\prime \prime}\right)\right) \in$ bangRel Rel
by(force intro: Rel.BRPar BRRes)
ultimately show $\exists P^{\prime} . P \| R \longmapsto \tau \prec P^{\prime} \wedge\left(P^{\prime},<\nu x>\left(Q^{\prime} \| Q^{\prime \prime}\right)\right) \in$ bangRel Rel by blast
qed
qed
next
case(Close2 a x $Q^{\prime} Q^{\prime \prime}$ Pa P)
hence $I H: \bigwedge P a .(P a,!Q) \in$ bangRel Rel $\Longrightarrow$ ?Sim Pa $\left(a<x>\prec Q^{\prime \prime}\right)$ by simp
have $Q$ Trans: $Q \longmapsto a<\nu x>\prec Q^{\prime}$ by fact
have $x$ Fresh $Q: x \sharp Q$ by fact
have $(P a, Q \|!Q) \in$ bangRel Rel and $x \sharp P a$ by fact+
thus ?case
proof (induct rule: BRParCases)
case (BRPar P R)
have PRelQ: $(P, Q) \in$ Rel and $R B R Q:(R,!Q) \in$ bangRel Rel by fact + have $x \sharp P \| R$ by fact
hence $x$ Fresh $P$ : $x \sharp P$ and $x$ Fresh $R$ : $x \sharp R$ by simp +
show ? case
proof (auto simp add: residual.inject)
from PRelQ have $P \rightsquigarrow[$ Rel $] Q$ by (rule Sim)
with $Q$ Trans xFresh $P$ obtain $P^{\prime}$ where PTrans: $P \longmapsto a<\nu x>\prec P^{\prime}$ and $P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}$ by (blast dest: elim)
from $R B R Q$ IH have $\exists R^{\prime} . \quad R \longmapsto a<x>\prec R^{\prime} \wedge\left(R^{\prime}, Q^{\prime \prime}\right) \in$ bangRel Rel by auto
then obtain $R^{\prime}$ where RTrans: $R \longmapsto a<x>\prec R^{\prime}$ and $R^{\prime}$ RelQ ${ }^{\prime \prime}:\left(R^{\prime}\right.$, $\left.Q^{\prime \prime}\right) \in$ bangRel Rel
by blast
from PTrans RTrans xFreshR have $P \| R \longmapsto \tau \prec<\nu x>\left(P^{\prime} \| R^{\prime}\right)$
by(rule Early-Semantics.Close2)
moreover from $P^{\prime} \operatorname{Rel} Q^{\prime} R^{\prime} \operatorname{Rel} Q^{\prime \prime}$ have $\left(<\nu x>\left(P^{\prime} \| R^{\prime}\right),<\nu x>\left(Q^{\prime} \|\right.\right.$
$\left.\left.Q^{\prime \prime}\right)\right) \in$ bangRel Rel
by (force intro: Rel.BRPar BRRes)
ultimately show $\exists P^{\prime} . P \| R \longmapsto \tau \prec P^{\prime} \wedge\left(P^{\prime},<\nu x>\left(Q^{\prime} \| Q^{\prime \prime}\right)\right) \in$
bangRel Rel by blast
qed
qed
next
case(Bang Rs Pa P)
hence $I H: \wedge P a .(P a, Q \|!Q) \in$ bangRel Rel $\Longrightarrow$ ?Sim Pa Rs by simp
have $(P a,!Q) \in$ bangRel Rel by fact
thus ?case
proof (induct rule: BRBangCases)
case (BRBang P)
have PRelQ: $(P, Q) \in$ Rel by fact
hence $(!P,!Q) \in$ bangRel Rel by(rule Rel.BRBang)
with PRelQ have $(P\|!P, Q\|!Q) \in$ bangRel Rel by (rule BRPar)
with $I H$ have ? $\operatorname{Sim}(P \|!P)$ Rs by simp
thus ?case by(force intro: TransitionsEarly.Bang)
qed
qed
\}
moreover from PRelQ have $(!P,!Q) \in$ bangRel Rel by (rule BRBang) ultimately show ?thesis by (auto simp add: strongSimEarly-def)
qed
end
theory Strong-Early-Bisim-Pres
imports Strong-Early-Bisim Strong-Early-Sim-Pres
begin
lemma tauPres:
fixes $P:: p i$
and $\quad Q:: p i$
assumes $P \sim Q$
shows $\tau .(P) \sim \tau .(Q)$
proof -
let $? X=\{(\tau .(P), \tau .(Q)) \mid P Q . P \sim Q\}$
from $\langle P \sim Q\rangle$ have $(\tau .(P), \tau .(Q)) \in$ ? $X$ by auto
thus ?thesis
by (coinduct rule: bisimCoinduct) (auto intro: tauPres dest: bisimE)
qed
lemma inputPres:
fixes $P:: p i$
and $\quad Q:: p i$
and $a::$ name
and $x::$ name
assumes $P \operatorname{Sim} Q: \forall y . P[x::=y] \sim Q[x::=y]$
shows $a<x>. P \sim a<x>. Q$
proof -
let ? $X=\{(a<x>. P, a<x>. Q) \mid$ ax $P Q . \forall y . P[x::=y] \sim Q[x::=y]\}$
\{
fix $a x P a x Q p$
assume $(a x P, a x Q) \in$ ? $X$
then obtain $a x P Q$ where $A: \forall y . P[x::=y] \sim Q[x::=y]$ and $B: a x P=$
$a<x>. P$ and $C: a x Q=a<x>. Q$
by auto
have $\bigwedge y \cdot((p::$ name prm $) \cdot P)[(p \cdot x)::=y] \sim(p \cdot Q)[(p \cdot x)::=y]$
proof -
fix $y$
from $A$ have $P[x::=($ rev $p \cdot y)] \sim Q[x::=($ rev $p \cdot y)]$ by blast
hence $(p \cdot(P[x::=(\operatorname{rev} p \cdot y)])) \sim p \cdot(Q[x::=(\operatorname{rev} p \cdot y)])$ by (rule bisimClosed)
thus $(p \cdot P)[(p \cdot x)::=y] \sim(p \cdot Q)[(p \cdot x)::=y]$
by (simp add: eqvts pt-pi-rev[OF pt-name-inst, OF at-name-inst])
qed
hence $((p::$ name prm) $\cdot a x P, p \cdot a x Q) \in$ ? $X$ using $B C$
by auto
\}
hence eqvt ? $X$ by (simp add: eqvt-def)
from PSimQ have $(a<x>. P, a<x>. Q) \in ? X$ by auto
thus ?thesis
proof(coinduct rule: bisimCoinduct)
case $(c \operatorname{Sim} P Q)$
thus ?case using 〈eqvt ? $X$ 〉
by (force intro: inputPres)

```
    next
        case(cSym P Q)
        thus ?case
        by(blast dest: bisimE)
    qed
qed
lemma outputPres:
    fixes P :: pi
    and }Q:: p
    and a :: name
    and b :: name
    assumes P~Q
    shows a{b}.P~a{b}.Q
proof -
    let ?X = {(a{b}.P,a{b}.Q)|abPQ.P~Q}
    from }\langleP~Q\rangle\mathrm{ have (a{b}.P,a{b}.Q) &?X by auto
    thus ?thesis
        by(coinduct rule: bisimCoinduct) (blast intro: outputPres dest: bisimE)+
qed
lemma matchPres:
    fixes P :: pi
    and }Q::p
    and a :: name
    and b:: name
    assumes P~Q
    shows [a\frownb]P~[a\frownb]Q
proof -
    let ?X = {x. \existsPQ a b. P~Q\wedge x = ([a\frownb]P,[a\frownb]Q)}
    from assms have ([a\frownb]P,[a\frownb]Q)\in?X by blast
    thus ?thesis
    by(coinduct rule: bisimCoinduct) (blast intro: matchPres dest: bisimE)+
qed
lemma mismatchPres:
    fixes }P:: p
    and }Q::p
    and a :: name
    and b :: name
    assumes P~Q
    shows [a\not=b]P~[a\not=b]Q
proof -
```

```
    let ?}X={x.\existsPQab.P~Q\wedgex=([a\not=b]P,[a\not=b]Q)
    from assms have ([a\not=b]P,[a\not=b]Q)\in?X by blast
    thus ?thesis
    by(coinduct rule: bisimCoinduct) (blast intro: mismatchPres dest: bisimE)+
qed
lemma sumPres:
    fixes P :: pi
    and }Q::p
    and }R::p
    assumes P~Q
    shows }P\oplusR~Q\oplus
proof -
    let ? }X={(P\oplusR,Q\oplusR)|PQR.P~Q
    from assms have (P\oplusR,Q\oplusR)\in?X by blast
    thus ?thesis
        by(coinduct rule: bisimCoinduct) (auto dest: bisimE intro: reflexive sumPres)
qed
lemma resPres:
    fixes }P::p
    and }Q::p
    and x :: name
    assumes P~Q
    shows <\nux>P~<\nux>>Q
proof -
    let ?X = {x.\existsPQ.P~Q\wedge(\existsa.x=(<\nu a>P,<\nua>Q))}
    from assms have (<\nux>P,<\nux>>Q)\in?X by blast
    thus ?thesis
    proof(coinduct rule: bisimCoinduct)
        case(cSim xP xQ)
        moreover {
            fix PQa
            assume P~Q
            hence P}>>[bisim] Q by(rule bisimE
            moreover have }\PQa.P~Q\Longrightarrow(<\nua>P,<\nua>Q)\in?X\cup\mathrm{ bisim by
blast
            moreover have bisim \subseteq?X \cup bisim by blast
            moreover have eqvt bisim by(rule eqvt)
            moreover have eqvt (?X \cup bisim) using eqvts
                by(auto simp add: eqvt-def) blast
            ultimately have <\nua>P}\rightsquigarrow[(?X\cup\mathrm{ bisim )}]<\nua>
                by(rule Strong-Early-Sim-Pres.resPres)
            }
            ultimately show ?case by auto
```

```
    next
        case(cSym xP xQ)
        thus ?case by(auto dest: bisimE)
    qed
qed
lemma parPres:
    fixes }P::p
    and }Q:: p
    and }R::p
    and }T::p
    assumes P~Q
    shows P|R~Q|R
proof -
    let ?X = {(resChain lst (P|R), resChain lst (Q|R))| lst P Q R. P ~ Q}
    have BC: \bigwedgePQ.P|Q= resChain [] (P|Q) by auto
    from assms have (P|R,Q|R)\in?X by(blast intro: BC)
    thus ?thesis
    proof(coinduct rule: bisimWeakCoinduct)
        case(cSim PR QR)
        moreover {
            fix lst PQ R
            assume P~Q
            have eqvt ?X using eqvts by(auto simp add: eqvt-def) blast
            moreover have Res: \PQx. (P,Q) \in?X\Longrightarrow(<\nux>P,<\nux>Q) \in?X
                by(auto, rule-tac x=x#lst in exI) auto
            moreover {
                from }\langleP~Q\rangle\mathrm{ have Pw[bisim] Q by(rule bisimE)
                moreover note <P ~ Q>
                    moreover have }\PQR.P~Q\Longrightarrow(P|R,Q|R)\in?
                    by(blast intro: BC)
                    ultimately have P|R\rightsquigarrow[?X] Q|R using Res
                    by(rule parPres)
            }
            ultimately have resChain lst (P|R)\rightsquigarrow[?X] resChain lst (Q|R)
            by(rule resChainI)
        }
        ultimately show ?case by auto
    next
        case(cSym P Q)
        thus ?case by(auto dest: bisimE)
    qed
qed
lemma bangRelBisimE:
    fixes P :: pi
```

```
    and }Q :: p
    and Rel :: (pi\timespi) set
    assumes A: (P,Q)\in bangRel Rel
    and Sym: \PQ.(P,Q)\inRel \Longrightarrow(Q,P)\in\operatorname{Rel}
    shows }(Q,P)\in\mathrm{ bangRel Rel
proof -
    from A show ?thesis
    proof(induct)
    fix P Q
    assume (P,Q)\inRel
    hence (Q,P)\inRel by(rule Sym)
    thus (!Q,!P)\inbangRel Rel by(rule BRBang)
    next
    fix PQRT
    assume RRelT: (R,T)\inRel
    assume IH: (Q,P)\in bangRel Rel
    from RRelT have (T,R)\inRel by(rule Sym)
    thus (T|Q,R|P)\in bangRel Rel using IH by(rule BRPar)
    next
    fix PQa
    assume ( }Q,P)\in\mathrm{ bangRel Rel
    thus (<\nua>Q,<\nua>P)\in bangRel Rel by(rule BRRes)
    qed
qed
lemma bangPres:
    fixes P :: pi
    and }Q::p
    assumes PBiSimQ: P~Q
    shows !P~!Q
proof -
    let ?X = bangRel bisim
    from PBiSimQ have (!P,!Q)\in?X by(rule BRBang)
    thus ?thesis
    proof(coinduct rule: bisim WeakCoinduct)
        case(cSim bP bQ)
        {
            fix PQ
            assume (P,Q)\in?X
            hence P}\rightsquigarrow[?X]
            proof(induct)
                fix PQ
                    assume P~Q
                    thus !P\rightsquigarrow[?X]!Q using bisimE(1) eqvt
                        by(rule Strong-Early-Sim-Pres.bangPres)
```

```
next
            fix PQRT
            assume RBiSimT: R~T
            assume PBangRelQ: }(P,Q)\in?
            assume PSimQ: P\rightsquigarrow[?X] Q
            from RBiSimT have R}\rightsquigarrow[bisim] T by(blast dest: bisimE
            thus R|P\rightsquigarrow[?X] T|Q using PSimQ RBiSimT PBangRelQ BRPar
BRRes eqvt eqvtBangRel
            by(blast intro: Strong-Early-Sim-Pres.parCompose)
                    next
                    fix PQa
                    assume P}\rightsquigarrow[?X]
                    moreover from eqvtBangRel eqvt have eqvt ?X by blast
            ultimately show <\nua>P\rightsquigarrow[?X]<\nua>Q using BRRes by(blast intro:
Strong-Early-Sim-Pres.resPres)
            qed
        }
        with }\langle(bP,bQ)\in?X> show ?case by blas
    next
        case(cSym bP bQ)
        thus ?case by(metis bangRelSymetric bisimE)
    qed
qed
end
theory Strong-Early-Bisim-Subst-Pres
    imports Strong-Early-Bisim-Subst Strong-Early-Bisim-Pres
begin
lemma tauPres:
    fixes P :: pi
    and }Q:: p
    assumes P ~}\mp@subsup{~}{}{s}
    shows }\tau.(P)\mp@subsup{~}{}{s}\tau.(Q
using assms
by(force simp add: substClosed-def intro:Strong-Early-Bisim-Pres.tauPres)
lemma inputPres:
    fixes P :: pi
    and }Q:: p
    and a :: name
    and x :: name
    assumes P ~}\mp@subsup{~}{}{s}
    shows }a<x>.P\mp@subsup{~}{}{s}a<x>.
```

```
proof(auto simp add: substClosed-def)
    fix }\sigma::(name\times name) lis
    {
        fix PQax \sigma
    assume P ~}\mp@subsup{~}{}{s}
    then have P[<\sigma>] ~s}Q[<\sigma>] by(rule partUnfold
    then have }\forally.(P[<\sigma>])[x::=y]~(Q[<\sigma>])[x::=y
            apply(auto simp add: substClosed-def)
            by(erule-tac x=[(x,y)] in allE) auto
    moreover assume }x\sharp
    ultimately have ( }a<x>.P)[<\sigma>]~(a<x>.Q)[<\sigma>
            by(force intro:Strong-Early-Bisim-Pres.inputPres)
    }
    note Goal = this
    obtain y::name where }y\sharpP\mathrm{ and }y\sharpQ\mathrm{ and }y\sharp
        by(generate-fresh name) auto
    from }\langleP\mp@subsup{~}{}{s}Q\rangle\mathrm{ have }([(x,y)]\cdotP)\mp@subsup{~}{}{s}([(x,y)]\cdotQ)\boldsymbol{by}(\mathrm{ rule eqvtI)
    hence (a<y>.([(x,y)]\cdotP))[<\sigma>]~(a<y>.([(x,y)]\cdotQ))[<\sigma>] using<y\sharp\sigma>
by(rule Goal)
    moreover from < }y\sharpP>\langley\sharpQ> have a<x>.P=a<y>.([(x,y)] \cdot P) an
a<x>.Q = a<y>.([(x,y)] \cdot Q)
    by(simp add: alphaInput)+
    ultimately show (a<x>.P)[<\sigma>] ~ (a<x>.Q)[<\sigma>] by simp
qed
lemma outputPres:
    fixes P :: pi
    and }Q::p
    assumes P ~}\mp@subsup{~}{}{s}
    shows a{b}.P (s a{b}.Q
using assms
by(force simp add: substClosed-def intro: Strong-Early-Bisim-Pres.outputPres)
lemma matchPres:
    fixes P :: pi
    and }Q::p
    and a :: name
    and b :: name
    assumes P ~}\mp@subsup{~}{}{s}
    shows [a\frownb]P ~
using assms
by(force simp add: substClosed-def intro: Strong-Early-Bisim-Pres.matchPres)
```

```
lemma mismatchPres:
    fixes \(P:: p i\)
    and \(\quad Q:: p i\)
    and \(a::\) name
    and \(b::\) name
    assumes \(P \sim^{s} Q\)
    shows \([a \neq b] P \sim^{s}[a \neq b] Q\)
using assms
by(force simp add: substClosed-def intro: Strong-Early-Bisim-Pres.mismatchPres)
lemma sumPres:
    fixes \(P\) :: \(p i\)
    and \(\quad Q:: p i\)
    and \(\quad R:: p i\)
    assumes \(P \sim^{s} Q\)
    shows \(P \oplus R \sim^{s} Q \oplus R\)
using assms
by(force simp add: substClosed-def intro: Strong-Early-Bisim-Pres.sumPres)
lemma parPres:
    fixes \(P\) :: \(p i\)
    and \(\quad Q:: p i\)
    and \(\quad R:: p i\)
    assumes \(P \sim^{s} Q\)
    shows \(P\left\|R \sim^{s} Q\right\| R\)
using assms
by(force simp add: substClosed-def intro: Strong-Early-Bisim-Pres.parPres)
lemma resPres:
    fixes \(P\) :: \(p i\)
    and \(\quad Q:: p i\)
    and \(x::\) name
    assumes \(\operatorname{PeqQ:} P \sim^{s} Q\)
    shows \(<\nu x>P \sim^{s}<\nu x>Q\)
proof (auto simp add: substClosed-def)
    fix \(s::(\) name \(\times\) name) list
    have Res: \(\bigwedge P Q x s . \llbracket P[\langle s\rangle] \sim Q[\langle s\rangle] ; x \sharp s \rrbracket \Longrightarrow(\langle\nu x\rangle P)[\langle s\rangle] \sim(\langle\nu x\rangle Q)[\langle s\rangle]\)
        by (force intro: Strong-Early-Bisim-Pres.resPres)
    have \(\exists c::\) name. \(c \sharp(P, Q, s)\) by (blast intro: name-exists-fresh)
```

then obtain c::name where cFreshP:c甘P and cFresh $Q: c \sharp Q$ and cFreshs: $c \sharp s$
by (force simp add: fresh-prod)
from PeqQ have $P[<([(x, c)] \cdot s)>] \sim Q[<([(x, c)] \cdot s)>]$ by $($ simp add: subst-Closed-def)
hence $([(x, c)] \cdot P[<([(x, c)] \cdot s)>]) \sim([(x, c)] \cdot Q[<([(x, c)] \cdot s)>])$ by $($ rule Strong-Early-Bisim.bisimClosed)
hence $([(x, c)] \cdot P)[<s\rangle] \sim([(x, c)] \cdot Q)[<s\rangle]$ by simp
hence $(<\nu c>([(x, c)] \cdot P))[<s\rangle] \sim(<\nu c>([(x, c)] \cdot Q))[<s\rangle]$ using cFreshs by(rule Res)
moreover from cFreshP cFresh $Q$ have $\langle\nu x\rangle P=\langle\nu c\rangle([(x, c)] \cdot P)$ and $<\nu x>Q=<\nu c>([(x, c)] \cdot Q)$
by (simp add: alphaRes)+
ultimately show $(\langle\nu x\rangle P)[\langle s\rangle] \sim(\langle\nu x\rangle Q)[\langle s\rangle]$ by simp
qed
lemma bangPres:
fixes $P:: p i$
and $\quad Q:: p i$
assumes $P \sim^{s} Q$
shows $!P \sim^{s}!Q$
using assms
by(force simp add: substClosed-def intro: Strong-Early-Bisim-Pres.bangPres)
end
theory Early-Tau-Chain
imports Early-Semantics
begin
abbreviation tauChain $::$ pi $\Rightarrow$ pi $\Rightarrow$ bool $\left(-\Longrightarrow_{\tau}-[80,80] 80\right)$
where $P \Longrightarrow_{\tau} P^{\prime} \equiv\left(P, P^{\prime}\right) \in\left\{\left(P, P^{\prime}\right) \mid P P^{\prime} . P \longmapsto \tau \prec P^{\prime}\right\}^{*} *$
lemma tauActTauChain:
fixes $P$ :: $p i$
and $P^{\prime}:: p i$
assumes $P \longmapsto \tau \prec P^{\prime}$
shows $P \Longrightarrow_{\tau} P^{\prime}$
using assms
by auto
lemma tauChainAddTau[intro]:

```
    fixes P :: pi
    and }\mp@subsup{P}{}{\prime}::p
    and }\mp@subsup{P}{}{\prime\prime}::p
    shows }P\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime}\Longrightarrow\mp@subsup{P}{}{\prime}\longmapsto\tau\prec\mp@subsup{P}{}{\prime\prime}\LongrightarrowP\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime\prime
    and }P\longmapsto\tau\prec\mp@subsup{P}{}{\prime}\Longrightarrow\mp@subsup{P}{}{\prime}\Longrightarrow\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime\prime}\LongrightarrowP\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime\prime
by(auto dest: tauActTauChain)
lemma tauChainInduct[consumes 1, case-names id ih]:
    fixes P :: pi
    and }\mp@subsup{P}{}{\prime}::p
    assumes P\Longrightarrow}\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime
    and FP
    and }\\mp@subsup{P}{}{\prime\prime}\mp@subsup{P}{}{\prime\prime\prime}.\llbracketP\Longrightarrow\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime\prime};\mp@subsup{P}{}{\prime\prime}\longmapsto\tau\prec\mp@subsup{P}{}{\prime\prime\prime};F\mp@subsup{P}{}{\prime}\rrbracket\LongrightarrowF\mp@subsup{P}{}{\prime\prime\prime
    shows F P '
using assms
by(drule-tac rtrancl-induct) auto
lemma eqvtChainI:
    fixes P :: pi
    and }\mp@subsup{P}{}{\prime}::p
    and perm :: name prm
    assumes P\Longrightarrow}\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime
    shows (perm • P) \Longrightarrow}\mp@subsup{|}{\tau}{}(\mathrm{ perm }\cdot\mp@subsup{P}{}{\prime}
using assms
proof(induct rule: tauChainInduct)
    case id
    thus ?case by simp
next
    case(ih P'\prime P}\mp@subsup{}{\prime\prime\prime}{\prime\prime
    have }P\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime\prime}\mathrm{ and }\mp@subsup{P}{}{\prime\prime}\longmapsto\tau\prec\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ by fact+
    hence (perm • P')}\longmapsto\tau\prec(\mathrm{ perm • P''') by(drule-tac TransitionsEarly.eqvt)
auto
    moreover have (perm • P) \Longrightarrow>
    ultimately show ?case by(force dest: tauActTauChain)
qed
lemma eqvtChainE:
    fixes perm :: name prm
    and }P\mathrm{ :: pi
    and }\mp@subsup{P}{}{\prime}::p
    assumes Trans: (perm • P) \Longrightarrow>
    shows }P\Longrightarrow\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime
```

```
proof -
    have rev perm • (perm • P) = P by(simp add: pt-rev-pi[OF pt-name-inst, OF
at-name-inst])
    moreover have rev perm • (perm \cdot P') = P' by (simp add: pt-rev-pi[OF pt-name-inst,
OF at-name-inst])
    ultimately show ?thesis using assms
        by(drule-tac perm=rev perm in eqvtChainI, simp)
qed
lemma eqvtChainEq:
    fixes P :: pi
    and }\mp@subsup{P}{}{\prime}::p
    and perm :: name prm
    shows }P\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime}=(\mathrm{ perm }\cdotP)\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}(\mathrm{ perm }\cdot\mp@subsup{P}{}{\prime}
by(blast intro: eqvtChainE eqvtChainI)
lemma freshChain:
    fixes P :: pi
    and }\mp@subsup{P}{}{\prime}::p
    and }x\mathrm{ :: name
    assumes }P\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime
    and }\quadx\sharp
    shows }x\sharp\mp@subsup{P}{}{\prime
using assms
proof(induct rule: tauChainInduct)
    case id
    thus ?case by simp
next
    case(ih P' P'\prime
    have }x\sharpP\mathrm{ and }x\sharpP\Longrightarrowx\sharp\mp@subsup{P}{}{\prime}\mathrm{ by fact+
    hence }x\sharp\mp@subsup{P}{}{\prime}\mathrm{ by simp
    moreover have }\mp@subsup{P}{}{\prime}\longmapsto\tau\prec\mp@subsup{P}{}{\prime\prime}\mathrm{ by fact
    ultimately show ?case by(force intro: freshTransition)
qed
lemma matchChain:
    fixes b :: name
    and }P::p
    and }\mp@subsup{P}{}{\prime}::p
    assumes P\Longrightarrow}\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime
    and }P\not=\mp@subsup{P}{}{\prime
    shows [b\frownb]P\Longrightarrow}\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime
using assms
proof(induct rule: tauChainInduct)
```

```
    case id
    thus ?case by simp
next
    case(ih P'\prime P '\prime\prime)
    have }\mp@subsup{P}{}{\prime\prime}\mathrm{ TransP}\mp@subsup{P}{}{\prime\prime\prime}:\mp@subsup{P}{}{\prime\prime}\longmapsto\tau\prec\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ by fact
    show [b\frownb]P\Longrightarrow}\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime\prime\prime
    proof(cases P= P')
        assume P=\mp@subsup{P}{}{\prime\prime}
    moreover with }\mp@subsup{P}{}{\prime\prime}\mathrm{ Trans }\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ have [b}b]P\longmapsto\mp@code{\prec \prec P'\prime\prime}\mathbf{by}(\mathrm{ force intro: Match)
    thus [b\frownb]P\Longrightarrow}\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ by(rule tauActTauChain)
    next
        assume P}\not=\mp@subsup{P}{}{\prime\prime
        moreover have }P\not=\mp@subsup{P}{}{\prime\prime}\Longrightarrow[b\frownb]P\Longrightarrow\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime\prime}\mathrm{ by fact
        ultimately show [b\frownb]P\Longrightarrow}\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ using }\mp@subsup{P}{}{\prime\prime}\mathrm{ Trans }\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ by(blast)
    qed
qed
lemma mismatchChain:
    fixes a :: name
    and b:: name
    and }P::p
    and }\mp@subsup{P}{}{\prime}:: p
    assumes PChain: P\Longrightarrow}\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime
    and aineqb: a}\not=
    and PineqP': P}\not=\mp@subsup{P}{}{\prime
    shows [a\not=b]P\Longrightarrow}\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime
proof -
    from PChain PineqP' show ?thesis
    proof(induct rule: tauChainInduct)
        case id
        thus ?case by simp
    next
        case(ih P'\prime}\mp@subsup{P}{}{\prime\prime\prime}
        have }\mp@subsup{P}{}{\prime\prime}\mathrm{ TransP}\mp@subsup{P}{}{\prime\prime\prime}:\mp@subsup{P}{}{\prime\prime}\longmapsto\tau\prec\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ by fact
        show [a\not=b]P\Longrightarrow}\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime\prime\prime
        proof(cases P= P')
            assume P=\mp@subsup{P}{}{\prime\prime}
            moreover with aineqb }\mp@subsup{P}{}{\prime\prime}\mathrm{ Trans }\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ have [ }a\not=b]P\longmapsto\tau\prec\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ by(force intro:
Mismatch)
            thus [a\not=b]P\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime\prime\prime}}\mathbf{by}(\mathrm{ rule tauActTauChain)
        next
            assume P}\not=\mp@subsup{P}{}{\prime\prime
            moreover have }P\not=\mp@subsup{P}{}{\prime\prime}\Longrightarrow[a\not=b]P\Longrightarrow\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime\prime}\mathrm{ by fact
            ultimately show [a\not=b]P\Longrightarrow}\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ using }\mp@subsup{P}{}{\prime\prime}\mathrm{ Trans }\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ by(blast)
        qed
    qed
qed
```

```
lemma sum1Chain:
    fixes \(P:: p i\)
    and \(P^{\prime}:: p i\)
    and \(\quad Q\) :: pi
    assumes \(P \Longrightarrow{ }_{\tau} P^{\prime}\)
    and \(\quad P \neq P^{\prime}\)
    shows \(P \oplus Q \Longrightarrow_{\tau} P^{\prime}\)
using assms
proof(induct rule: tauChainInduct)
    case id
    thus ?case by simp
next
    case(ih \(P^{\prime \prime} P^{\prime \prime \prime}\) )
    have \(P^{\prime \prime} \operatorname{Trans} P^{\prime \prime \prime}: P^{\prime \prime} \longmapsto \tau \prec P^{\prime \prime \prime}\) by fact
    show \(P \oplus Q \Longrightarrow_{\tau} P^{\prime \prime \prime}\)
    proof (cases \(P=P^{\prime \prime}\) )
        assume \(P=P^{\prime \prime}\)
        moreover with \(P^{\prime \prime}\) Trans \(P^{\prime \prime \prime}\) have \(P \oplus Q \longmapsto \tau \prec P^{\prime \prime \prime}\) by(force intro: Sum1)
        thus \(P \oplus Q \Longrightarrow_{\tau} P^{\prime \prime \prime}\) by(force intro: tauActTauChain)
    next
    assume \(P \neq P^{\prime \prime}\)
    moreover have \(P \neq P^{\prime \prime} \Longrightarrow P \oplus Q \Longrightarrow_{\tau} P^{\prime \prime}\) by fact
    ultimately show \(P \oplus Q \Longrightarrow{ }_{\tau} P^{\prime \prime \prime}\) using \(P^{\prime \prime}\) Trans \(P^{\prime \prime \prime}\) by(force dest: tauAct-
TauChain)
    qed
qed
lemma sum2Chain:
    fixes \(P\) :: \(p i\)
    and \(\quad Q:: p i\)
    and \(Q^{\prime}:: p i\)
    assumes \(Q \Longrightarrow_{\tau} Q^{\prime}\)
    and \(\quad Q \neq Q^{\prime}\)
    shows \(P \oplus Q \Longrightarrow_{\tau} Q^{\prime}\)
using assms
proof (induct rule: tauChainInduct)
    case id
    thus ?case by simp
next
case (ih \(\left.Q^{\prime \prime} Q^{\prime \prime \prime}\right)\)
have \(Q^{\prime \prime}\) Trans \(Q^{\prime \prime \prime}: \quad Q^{\prime \prime} \longmapsto \tau \prec Q^{\prime \prime \prime}\) by fact
show \(P \oplus Q \Longrightarrow_{\tau} Q^{\prime \prime \prime}\)
\(\operatorname{proof}\left(\right.\) cases \(\left.Q=Q^{\prime \prime}\right)\)
    assume \(Q=Q^{\prime \prime}\)
```

```
        moreover with \mp@subsup{Q}{}{\prime\prime}Trans\mp@subsup{Q}{}{\prime\prime\prime}}\mathrm{ have }P\oplusQ\longmapsto\tau\prec\mp@subsup{Q}{}{\prime\prime\prime}\mathrm{ by(force intro: Sum2)
        thus }P\oplusQ\Longrightarrow\mp@subsup{\overbrace}{\tau}{}\mp@subsup{Q}{}{\prime\prime\prime}\mathrm{ by(force intro: tauActTauChain)
    next
        assume Q\not=\mp@subsup{Q}{}{\prime\prime}
        moreover have }Q\not=\mp@subsup{Q}{}{\prime\prime}\LongrightarrowP\oplusQ\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{Q}{}{\prime\prime}\mathrm{ by fact
        ultimately show }P\oplusQ\Longrightarrow\mp@subsup{}{\tau}{}\mp@subsup{Q}{}{\prime\prime\prime}\mathrm{ using }\mp@subsup{Q}{}{\prime\prime}\mathrm{ Trans Q'/' by blast
    qed
qed
lemma Par1Chain:
    fixes P :: pi
    and }\mp@subsup{P}{}{\prime}::p
    and Q :: pi
    assumes P\Longrightarrow}\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime
    shows }P|Q\Longrightarrow\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime}|
using assms
proof(induct rule: tauChainInduct)
    case id
    thus?case by simp
next
    case(ih P'\prime P')
    have }\mp@subsup{P}{}{\prime\prime}Trans\mp@subsup{P}{}{\prime}: \mp@subsup{P}{}{\prime\prime}\longmapsto\tau\prec\mp@subsup{P}{}{\prime}\mathrm{ by fact
    have IH:P|Q\Longrightarrow}\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime\prime}|Q\mathrm{ by fact
    have }\mp@subsup{P}{}{\prime\prime}|Q\longmapsto\tau\prec\mp@subsup{P}{}{\prime}|Q\mathrm{ using P''TransP' by(force intro: Par1F)
    thus P|Q \Longrightarrow}\mp@subsup{\tau}{\tau}{}\mp@subsup{P}{}{\prime}|Q\mathrm{ using IH by(force dest: tauActTauChain)
qed
lemma Par2Chain:
    fixes P :: pi
    and Q :: pi
    and Q Q':: pi
    assumes }Q\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{Q}{}{\prime
    shows }P|Q\mp@subsup{\Longrightarrow}{\tau}{}P|\mp@subsup{Q}{}{\prime
using assms
proof(induct rule: tauChainInduct)
    case id
    thus?case by simp
next
    case(ih Q" Q')
```



```
    have IH:P|Q \Longrightarrow}\mp@subsup{\tau}{\tau}{}P|\mp@subsup{Q}{}{\prime\prime}\mathrm{ by fact
    have }P||\mp@subsup{Q}{}{\prime\prime}\longmapsto\tau\precP|\mp@subsup{Q}{}{\prime}\mathrm{ using }\mp@subsup{Q}{}{\prime\prime}T\mathrm{ Trans Q' by(force intro: Par2F)
    thus P|Q \Longrightarrow}\mp@subsup{}{\tau}{}P|\mp@subsup{Q}{}{\prime}\mathrm{ using IH by(force dest: tauActTauChain)
```

```
qed
lemma chainPar:
    fixes \(P:: p i\)
    and \(P^{\prime}:: p i\)
    and \(Q\) :: pi
    and \(Q^{\prime}:: p i\)
    assumes \(P \Longrightarrow_{\tau} P^{\prime}\)
    and \(\quad Q \Longrightarrow_{\tau} Q^{\prime}\)
    shows \(P\left\|Q \Longrightarrow_{\tau} P^{\prime}\right\| Q^{\prime}\)
proof -
    from \(\left\langle P \Longrightarrow_{\tau} P^{\prime}\right\rangle\) have \(P\left\|Q \Longrightarrow_{\tau} P^{\prime}\right\| Q\) by(rule Par1Chain)
    moreover from \(\left\langle Q \Longrightarrow_{\tau} Q^{\prime}\right\rangle\) have \(P^{\prime}\left\|Q \Longrightarrow_{\tau} P^{\prime}\right\| Q^{\prime}\) by (rule Par2Chain)
    ultimately show ?thesis by auto
qed
lemma ResChain:
    fixes \(P\) :: \(p i\)
    and \(\quad P^{\prime}:: p i\)
    and \(a\) :: name
    assumes \(P \Longrightarrow{ }_{\tau} P^{\prime}\)
    shows \(<\nu a>P \Longrightarrow_{\tau}<\nu a>P^{\prime}\)
using assms
proof(induct rule: tauChainInduct)
    case id
    thus? case by simp
next
    case (ih \(P^{\prime \prime} P^{\prime \prime \prime}\) )
    have \(P^{\prime \prime} \longmapsto \tau \prec P^{\prime \prime \prime}\) by fact
    hence \(<\nu a>P^{\prime \prime} \longmapsto \tau \prec<\nu a>P^{\prime \prime \prime}\) by (force intro: ResF)
    moreover have \(\left.<\nu a>P \Longrightarrow_{\tau}<\nu a\right\rangle P^{\prime \prime}\) by fact
    ultimately show ?case by (force dest: tauActTauChain)
qed
lemma substChain:
    fixes \(P:: p i\)
    and \(x\) :: name
    and \(b\) :: name
    and \(\quad P^{\prime}:: p i\)
    assumes PTrans: \(P[x::=b] \Longrightarrow_{\tau} P^{\prime}\)
    shows \(P[x::=b] \Longrightarrow_{\tau} P^{\prime}[x::=b]\)
proof (cases \(x=b\) )
    assume \(x=b\)
```

```
    with PTrans show ?thesis by simp
next
    assume }x\not=
    hence }x\sharpP[x::=b] by(simp add: fresh-fact2
    with PTrans have }x\sharp\mp@subsup{P}{}{\prime}\mathbf{by}(force intro: freshChain
    hence }\mp@subsup{P}{}{\prime}=\mp@subsup{P}{}{\prime}[x::=b] by(simp add: forget
    with PTrans show ?thesis by simp
qed
lemma bangChain:
    fixes P :: pi
    and }\mp@subsup{P}{}{\prime}::p
    assumes PTrans: P|!P\Longrightarrow}\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime
    and }\quad\mp@subsup{P}{}{\prime}\mathrm{ ineq: }\mp@subsup{P}{}{\prime}\not=P|!
    shows !P \Longrightarrow}\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime
using assms
proof(induct rule: tauChainInduct)
    case id
    thus ?case by simp
next
    case(ih P' P'\prime)
    show ?case
    proof(cases }\mp@subsup{P}{}{\prime}=P|!P
        case True
    from }\langle\mp@subsup{P}{}{\prime}\longmapsto\tau\prec\mp@subsup{P}{}{\prime\prime}\rangle\langle\mp@subsup{P}{}{\prime}=P|!P\rangle\mathrm{ have ! P}\longmapsto\tau\prec\mp@subsup{P}{}{\prime\prime}\mathrm{ by(blast intro: Bang)
    thus ?thesis by auto
    next
    case False
    from }\langle\mp@subsup{P}{}{\prime}\not=P|!P\rangle\mathrm{ have ! P <}\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime}\mathrm{ by(rule ih)
    with \langleP''\longmapsto\tau \prec 'P'\rangle}\mathrm{ show ?thesis by(auto dest: tauActTauChain)
    qed
qed
end
theory Weak-Early-Step-Semantics
    imports Early-Tau-Chain
begin
lemma inputSupportDerivative:
    assumes }P\longmapstoa<x>\prec\mp@subsup{P}{}{\prime
    shows (supp P') - {x}\subseteq supp P
using assms
apply(nominal-induct rule: inputInduct)
apply(auto simp add: pi.supp abs-supp supp-atm)
apply(rule ccontr)
```

```
apply(simp add: fresh-def[symmetric])
apply(drule-tac fresh-fact1)
apply(rotate-tac 4)
apply assumption
apply(simp add: fresh-def)
apply force
apply(case-tac x }\sharp>P
apply(drule-tac fresh-fact1)
apply(rotate-tac 2)
apply assumption
apply(simp add: fresh-def)
apply force
apply(rotate-tac 2)
apply(drule-tac fresh-fact2)
apply(simp add: fresh-def)
by force
lemma outputSupportDerivative:
    fixes }P::p
    and a :: name
    and }b\mathrm{ :: name
    and }\mp@subsup{P}{}{\prime}:: p
    assumes }P\longmapstoa[b]\prec\mp@subsup{P}{}{\prime
    shows (supp P})\subseteq((supp P)::name set
using assms
by(nominal-induct rule: outputInduct) (auto simp add: pi.supp abs-supp)
lemma boundOutputSupportDerivative:
    assumes P\longmapstoa<\nux> \prec P'
    and }\quadx\sharp
    shows (supp P
using assms
by(nominal-induct rule: boundOutputInduct) (auto simp add: pi.supp abs-supp supp-atm
dest:outputSupportDerivative)
lemma tauSupportDerivative:
    assumes }P\longmapsto\tau\prec\mp@subsup{P}{}{\prime
    shows ((supp P}\mp@subsup{P}{}{\prime})::name set)\subseteq\operatorname{supp}
using assms
proof(nominal-induct rule: tauInduct)
    case(Tau P)
    thus ?case by(force simp add: pi.supp)
next
    case(Match P)
```

```
    thus ?case by(force simp add: pi.supp)
next
    case(Mismatch P)
    thus ?case by(force simp add: pi.supp)
next
    case(Sum1 P)
    thus ?case by(force simp add: pi.supp)
next
    case(Sum2 P)
    thus ?case by(force simp add: pi.supp)
next
    case(Par1 P)
    thus ?case by(force simp add: pi.supp)
next
    case(Par2 P)
    thus ?case by(force simp add: pi.supp)
next
    case(Comm1 P a b P'Q Q')
    from }\langleP\longmapstoa<b> \prec\mp@subsup{P}{}{\prime}\rangle\mathrm{ have (supp P}\mp@subsup{P}{}{\prime})-{b}\subseteq\mathrm{ supp P by(rule inputSupport-
Derivative)
    moreover from <Q\longmapstoa[b]\prec Q'> have ((supp Q ')::name set)\subseteq supp Q by(rule
outputSupportDerivative)
    moreover from <Q\longmapstoa[b]\prec < Q'> have b \in supp Q
    by(nominal-induct rule: outputInduct) (auto simp add: pi.supp abs-supp supp-atm)
    ultimately show ?case by(auto simp add: pi.supp)
next
    case(Comm2 P a b P'Q Q')
    from <P\longmapstoa[b]\prec \prec'> have ((supp P}\mp@subsup{P}{}{\prime})::name set) \subseteq supp P by(rule output-
SupportDerivative)
    moreover from <Q\longmapstoa<b> \prec Q'> have (supp Q') - {b}\subseteq supp Q by(rule
inputSupportDerivative)
    moreover from <P\longmapstoa[b]\prec P'> have b\in supp P
    by(nominal-induct rule: outputInduct) (auto simp add: pi.supp abs-supp supp-atm)
    ultimately show ?case by(auto simp add: pi.supp)
next
    case(Close1 P a x P' Q Q')
    thus ?case by(auto dest: inputSupportDerivative boundOutputSupportDerivative
simp add: abs-supp pi.supp)
next
    case(Close2 P a x P' Q Q')
    thus ?case by(auto dest: inputSupportDerivative boundOutputSupportDerivative
simp add: abs-supp pi.supp)
next
    case(Res P P' x)
    thus ?case by(force simp add: pi.supp abs-supp)
next
    case(Bang P P')
    thus ?case by(force simp add: pi.supp)
qed
```

```
lemma tauChainSupportDerivative:
    fixes \(P\) :: \(p i\)
    and \(P^{\prime}:: p i\)
    assumes \(P \Longrightarrow_{\tau} P^{\prime}\)
    shows \(\left(\left(\operatorname{supp} P^{\prime}\right)::\right.\) name set \() \subseteq(\operatorname{supp} P)\)
using assms
by(induct rule: tauChainInduct) (auto dest: tauSupportDerivative)
definition outputTransition :: pi \(\Rightarrow\) name \(\Rightarrow\) name \(\Rightarrow\) pi \(\Rightarrow\) bool \((-\Longrightarrow-<\nu->\prec\)
- \([80,80,80,80] 80)\)
    where \(P \Longrightarrow a<\nu x>\prec P^{\prime} \equiv \exists P^{\prime \prime \prime} P^{\prime \prime} . P \Longrightarrow{ }_{\tau} P^{\prime \prime \prime} \wedge P^{\prime \prime \prime} \longmapsto a<\nu x>\prec P^{\prime \prime} \wedge\)
\(P^{\prime \prime} \Longrightarrow_{\tau} P^{\prime}\)
definition freeTransition :: pi \(\Rightarrow\) freeRes \(\Rightarrow\) pi \(\Rightarrow\) bool \((-\Longrightarrow-\prec-[80,80,80] 80)\)
    where \(P \Longrightarrow \alpha \prec P^{\prime} \equiv \exists P^{\prime \prime \prime} P^{\prime \prime} . P \Longrightarrow_{\tau} P^{\prime \prime \prime} \wedge P^{\prime \prime \prime} \longmapsto \alpha \prec P^{\prime \prime} \wedge P^{\prime \prime} \Longrightarrow_{\tau} P^{\prime}\)
lemma transitionI:
    fixes \(P \quad:: p i\)
    and \(\quad P^{\prime \prime \prime}:: p i\)
    and \(\alpha\) :: freeRes
    and \(P^{\prime \prime}:: p i\)
    and \(P^{\prime}:: p i\)
    and \(a\) :: name
    and \(x\) :: name
    shows \(\llbracket P \Longrightarrow_{\tau} P^{\prime \prime \prime} ; P^{\prime \prime \prime} \longmapsto \alpha \prec P^{\prime \prime} ; P^{\prime \prime} \Longrightarrow_{\tau} P \rrbracket \Longrightarrow P \Longrightarrow \alpha \prec P^{\prime}\)
    and \(\llbracket P \Longrightarrow_{\tau} P^{\prime \prime \prime} ; P^{\prime \prime \prime} \longmapsto a<\nu x>\prec P^{\prime \prime} ; P^{\prime \prime} \Longrightarrow_{\tau} P \rrbracket \Longrightarrow P \Longrightarrow a<\nu x>\prec P^{\prime}\)
by (auto simp add: outputTransition-def freeTransition-def)
lemma transitionE:
    fixes \(P:: p i\)
    and \(\alpha\) :: freeRes
    and \(P^{\prime}:: p i\)
    and \(a\) :: name
    and \(x\) :: name
    shows \(P \Longrightarrow \alpha \prec P^{\prime} \Longrightarrow\left(\exists P^{\prime \prime} P^{\prime \prime \prime} . P \Longrightarrow_{\tau} P^{\prime \prime} \wedge P^{\prime \prime} \longmapsto \alpha \prec P^{\prime \prime \prime} \wedge P^{\prime \prime \prime} \Longrightarrow_{\tau}\right.\)
\(P^{\prime}\) )
    and \(P \Longrightarrow a<\nu x>\prec P^{\prime} \Longrightarrow \exists P^{\prime \prime} P^{\prime \prime \prime} . P \Longrightarrow_{\tau} P^{\prime \prime \prime} \wedge P^{\prime \prime \prime} \longmapsto a<\nu x>\prec P^{\prime \prime} \wedge\)
\(P^{\prime \prime} \Longrightarrow_{\tau} P^{\prime}\)
by (auto simp add: outputTransition-def freeTransition-def)
lemma weakTransitionAlpha:
fixes \(P:: p i\)
and \(a\) :: name
and \(x\) :: name
```

```
    and }\mp@subsup{P}{}{\prime}:: p
    and y :: name
    assumes PTrans: P\Longrightarrowa<\nux>}\prec\mp@subsup{P}{}{\prime
    and }\quady\sharp
    shows }P\Longrightarrowa<\nuy>\prec([(x,y)]\cdot\mp@subsup{P}{}{\prime}
proof(cases }y=x\mathrm{ )
    case True
    with PTrans show ?thesis by simp
next
    case False
    from PTrans obtain P}\mp@subsup{P}{}{\prime\prime}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ where PChain: P }\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime\prime\prime
                    and }\mp@subsup{P}{}{\prime\prime\prime}Trans: P ''\prime \longmapstoa<\nux> \prec P'\prime
                        and P'Chain: P'\prime}\Longrightarrow\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime
    by(force dest: transitionE)
    note PChain
    moreover from PChain 〈y\sharpP> have y }#\mp@subsup{P}{}{\prime\prime\prime}\mathbf{by}(rule freshChain)
    with \mp@subsup{P}{}{\prime\prime\prime}Trans have }y\sharp\mp@subsup{P}{}{\prime\prime}\mathrm{ using }\langley\not=x\rangle\mathbf{by}(rule freshTransition)
    with }\mp@subsup{P}{}{\prime\prime\prime}Trans have P ''\prime\prime\longmapstoa<\nuy> \prec ([(x,y)] \cdot P') by(simp add: alphaBound-
Output name-swap)
    moreover from P P'Chain have ([(x,y)] \cdot P'\prime) \Longrightarrow>
    by(rule eqvtChainI)
    ultimately show ?thesis by(rule transitionI)
qed
lemma singleActionChain:
    fixes P :: pi
    and Rs :: residual
    shows }P\longmapstoa<\nux>\prec\mp@subsup{P}{}{\prime}\LongrightarrowP\Longrightarrowa<\nux>\prec\mp@subsup{P}{}{\prime
    and }P\longmapsto\alpha\prec\mp@subsup{P}{}{\prime}\LongrightarrowP\Longrightarrow\alpha\prec\mp@subsup{P}{}{\prime
proof -
    have }P\mp@subsup{\Longrightarrow}{\tau}{}P\mathrm{ by simp
    moreover assume P}\longmapstoa<\nux>\prec\mp@subsup{P}{}{\prime
    moreover have P' \Longrightarrow}\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime}\mathrm{ by simp
    ultimately show }P\Longrightarrowa<\nux>\prec\prec\mp@subsup{P}{}{\prime
    by(rule transitionI)
next
    have P\Longrightarrow}\mp@subsup{\Longrightarrow}{\tau}{}P\mathrm{ by simp
    moreover assume P\longmapsto\alpha\prec 琽
    moreover have }\mp@subsup{P}{}{\prime}\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime}\mathrm{ by simp
    ultimately show }P\Longrightarrow\alpha\prec\mp@subsup{P}{}{\prime
        by(rule transitionI)
qed
lemma Tau:
    fixes P :: pi
```

```
    shows}\tau.(P)\Longrightarrow\tau\prec
proof -
    have }\tau.(P)\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\tau.(P)\mathrm{ by simp
    moreover have }\tau.(P)\longmapsto\tau\precP\mathrm{ by(rule Early-Semantics.Tau)
    moreover have }P\mp@subsup{\Longrightarrow}{\tau}{}P\mathrm{ by simp
    ultimately show ?thesis by(rule transitionI)
qed
lemma Input:
    fixes a :: name
    and x :: name
    and u :: name
    and }P\mathrm{ :: pi
    shows }a<x>.P\Longrightarrowa<u>\precP[x::=u
proof -
    have }a<x>.P\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}a<x>.P by sim
    moreover have }a<x>.P\longmapstoa<u>\precP[x::=u] by(rule Early-Semantics.Input)
    moreover have }P[x::=u]\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}P[x::=u] by sim
    ultimately show ?thesis by(rule transitionI)
qed
lemma Output:
    fixes a :: name
    and b:: name
    and }P::p
    shows }a{b}.P\Longrightarrowa[b]\prec
proof -
    have }a{b}.P\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}a{b}.P\mathrm{ by simp
    moreover have }a{b}.P\longmapstoa[b]\precP\mathrm{ by(rule Early-Semantics.Output)
    moreover have }P\Longrightarrow\mp@subsup{}{\tau}{}P\mathrm{ by simp
    ultimately show ?thesis by(rule transitionI)
qed
lemma Match:
    fixes P :: pi
    and b :: name
    and }x\mathrm{ :: name
    and a :: name
    and }\mp@subsup{P}{}{\prime}::p
    and }\alpha:: freeRe
    shows }P\Longrightarrowb<\nux>\prec\mp@subsup{P}{}{\prime}\Longrightarrow[a\frowna]P\Longrightarrowb<\nux>\prec\mp@subsup{P}{}{\prime
    and }P\Longrightarrow\alpha\prec\mp@subsup{P}{}{\prime}\Longrightarrow[a\frowna]P\Longrightarrow\alpha\prec\mp@subsup{P}{}{\prime
proof -
    assume P\Longrightarrowb<\nux>}\prec\mp@subsup{P}{}{\prime
    then obtain P}\mp@subsup{P}{}{\prime\prime}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ where PChain: P >}\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime\prime\prime
                                    and P}\mp@subsup{P}{}{\prime\prime\prime}Trans: P '/\prime \longmapsto b <\nux> \prec P'\prime
```

$$
\text { and } P^{\prime \prime} \text { Chain: } P^{\prime \prime} \Longrightarrow_{\tau} P^{\prime}
$$

by (force dest: transitionE)
show $[a \frown a] P \Longrightarrow b<\nu x>\prec P^{\prime}$
proof (cases $P=P^{\prime \prime \prime}$ )
case True
have $[a \frown a] P \Longrightarrow_{\tau}[a \frown a] P$ by simp
moreover from $\left\langle P=P^{\prime \prime \prime}\right\rangle P^{\prime \prime \prime}$ Trans have $[a \frown a] P \longmapsto b<\nu x>\prec P^{\prime \prime}$
by (rule-tac Early-Semantics.Match) auto
ultimately show ?thesis using $P^{\prime \prime}$ Chain by (rule transitionI)
next
case False
from PChain $\left\langle P \neq P^{\prime \prime \prime}\right\rangle$ have $[a \frown a] P \Longrightarrow{ }_{\tau} P^{\prime \prime \prime}$ by (rule matchChain)
thus ?thesis using $P^{\prime \prime \prime}$ Trans $P^{\prime \prime}$ Chain by (rule transitionI)
qed
next
assume $P \Longrightarrow \alpha \prec P^{\prime}$
then obtain $P^{\prime \prime} P^{\prime \prime \prime}$ where PChain: $P \Longrightarrow_{\tau} P^{\prime \prime \prime}$
and $P^{\prime \prime \prime}$ Trans: $P^{\prime \prime \prime} \longmapsto \alpha \prec P^{\prime \prime}$
and $P^{\prime \prime}$ Chain: $P^{\prime \prime} \Longrightarrow_{\tau} P^{\prime}$
by (force dest: transitionE)
show $[a \frown a] P \Longrightarrow \alpha \prec P^{\prime}$
proof (cases $P=P^{\prime \prime \prime}$ )
case True
have $[a \frown a] P \Longrightarrow_{\tau}[a \frown a] P$ by simp
moreover from $\left\langle P=P^{\prime \prime \prime}\right\rangle P^{\prime \prime \prime}$ Trans have $[a \frown a] P \longmapsto \alpha \prec P^{\prime \prime}$
by (rule-tac Early-Semantics.Match) auto
ultimately show ?thesis using $P^{\prime \prime}$ Chain by (rule transitionI)
next
case False
from PChain $\left\langle P \neq P^{\prime \prime \prime}\right\rangle$ have $[a \frown a] P \Longrightarrow{ }_{\tau} P^{\prime \prime \prime}$ by (rule matchChain)
thus ?thesis using $P^{\prime \prime \prime}$ Trans $P^{\prime \prime}$ Chain by(rule transitionI)
qed
qed
lemma Mismatch:
fixes $P:: p i$
and $c::$ name
and $x$ :: name
and $a$ :: name
and $b$ :: name
and $P^{\prime}:: p i$
and $\quad \alpha::$ freeRes

$$
\begin{aligned}
& \text { shows } \llbracket P \Longrightarrow c<\nu x>\prec P^{\prime} ; a \neq b \rrbracket \Longrightarrow[a \neq b] P \Longrightarrow c<\nu x>\prec P^{\prime} \\
& \text { and } \llbracket P \Longrightarrow \alpha \prec P^{\prime} ; a \neq b \rrbracket \Longrightarrow[a \neq b] P \Longrightarrow \alpha \prec P^{\prime} \\
& \text { proof }- \\
& \text { assume } P \Longrightarrow c<\nu x>\prec P^{\prime} \\
& \text { then obtain } P^{\prime \prime} P^{\prime \prime \prime} \text { where PChain: } P \Longrightarrow P^{\prime \prime \prime} \\
& \text { and } P^{\prime \prime \prime} \text { Trans: } P^{\prime \prime \prime} \longmapsto c<\nu x>\prec P^{\prime \prime}
\end{aligned}
$$

$$
\text { and } P^{\prime \prime} \text { Chain: } P^{\prime \prime} \Longrightarrow_{\tau} P^{\prime}
$$

by (force dest: transitionE)
assume $a \neq b$
show $[a \neq b] P \Longrightarrow c<\nu x>\prec P^{\prime}$
proof (cases $P=P^{\prime \prime \prime}$ )
case True
have $[a \neq b] P \Longrightarrow_{\tau}[a \neq b] P$ by simp
moreover from $\left\langle P=P^{\prime \prime \prime}\right\rangle\langle a \neq b\rangle P^{\prime \prime \prime}$ Trans have $\left.[a \neq b] P \longmapsto c<\nu x\right\rangle \prec P^{\prime \prime}$
by(rule-tac Early-Semantics.Mismatch) auto
ultimately show ?thesis using $P^{\prime \prime}$ Chain by (rule transitionI)
next
case False
from PChain $\langle a \neq b\rangle\left\langle P \neq P^{\prime \prime \prime}\right\rangle$ have $[a \neq b] P \Longrightarrow{ }_{\tau} P^{\prime \prime \prime}$ by (rule mismatchChain)
thus ?thesis using $P^{\prime \prime \prime}$ Trans $P^{\prime \prime}$ Chain by (rule transitionI)
qed
next
assume $P \Longrightarrow \alpha \prec P^{\prime}$
then obtain $P^{\prime \prime} P^{\prime \prime \prime}$ where PChain: $P \Longrightarrow{ }_{\tau} P^{\prime \prime \prime}$
and $P^{\prime \prime \prime}$ Trans: $P^{\prime \prime \prime} \longmapsto \alpha \prec P^{\prime \prime}$
and $P^{\prime \prime}$ Chain: $P^{\prime \prime} \Longrightarrow_{\tau} P^{\prime}$
by (force dest: transitionE)
assume $a \neq b$
show $[a \neq b] P \Longrightarrow \alpha \prec P^{\prime}$
proof (cases $P=P^{\prime \prime \prime}$ )
case True
have $[a \neq b] P \Longrightarrow_{\tau}[a \neq b] P$ by simp
moreover from $\left\langle P=P^{\prime \prime \prime}\right\rangle\langle a \neq b\rangle P^{\prime \prime \prime}$ Trans have $[a \neq b] P \longmapsto \alpha \prec P^{\prime \prime}$
by(rule-tac Early-Semantics.Mismatch) auto
ultimately show ?thesis using $P^{\prime \prime}$ Chain by (rule transitionI)
next
case False
from PChain $\langle a \neq b\rangle\left\langle P \neq P^{\prime \prime \prime}\right\rangle$ have $[a \neq b] P \Longrightarrow{ }_{\tau} P^{\prime \prime \prime}$ by (rule mismatchChain)
thus ?thesis using $P^{\prime \prime \prime}$ Trans $P^{\prime \prime}$ Chain by (rule transitionI)
qed
qed
lemma Open:
fixes $P:: p i$
and $a$ :: name
and $b$ :: name
and $P^{\prime}:: p i$
assumes PTrans: $P \Longrightarrow a[b] \prec P^{\prime}$
and $\quad a \neq b$
shows $<\nu b>P \Longrightarrow a<\nu b>\prec P^{\prime}$
proof -
from PTrans obtain $P^{\prime \prime} P^{\prime \prime \prime}$ where PChain: $P \Longrightarrow_{\tau} P^{\prime \prime \prime}$ and $P^{\prime \prime \prime}$ Trans: $P^{\prime \prime \prime} \longmapsto a[b] \prec P^{\prime \prime}$
and $P^{\prime \prime}$ Chain: $P^{\prime \prime} \Longrightarrow_{\tau} P^{\prime}$
by (force dest: transitionE)
from PChain have $<\nu b>P \Longrightarrow_{\tau}<\nu b>P^{\prime \prime \prime}$ by(rule ResChain)
moreover from $P^{\prime \prime \prime}$ Trans $\langle a \neq b\rangle$ have $<\nu b>P^{\prime \prime \prime} \longmapsto a<\nu b>\prec P^{\prime \prime}$ by $($ rule Open)
ultimately show ?thesis using $P^{\prime \prime}$ Chain by (rule transitionI)
qed
lemma Sum1:
fixes $P:: p i$
and $a$ :: name
and $x$ :: name
and $P^{\prime}:: p i$
and $Q:: p i$
and $\alpha$ :: freeRes
shows $P \Longrightarrow a<\nu x>\prec P^{\prime} \Longrightarrow P \oplus Q \Longrightarrow a<\nu x>\prec P^{\prime}$
and $P \Longrightarrow \alpha \prec P^{\prime} \Longrightarrow P \oplus Q \Longrightarrow \alpha \prec P^{\prime}$
proof -
assume $P \Longrightarrow a<\nu x>\prec P^{\prime}$
then obtain $P^{\prime \prime} P^{\prime \prime \prime}$ where PChain: $P \Longrightarrow{ }_{\tau} P^{\prime \prime \prime}$
and $P^{\prime \prime \prime}$ Trans: $P^{\prime \prime \prime} \longmapsto a<\nu x>\prec P^{\prime \prime}$
and $P^{\prime \prime}$ Chain: $P^{\prime \prime} \Longrightarrow_{\tau} P^{\prime}$
by (force dest: transitionE)
show $P \oplus Q \Longrightarrow a<\nu x>\prec P^{\prime}$
proof (cases $P=P^{\prime \prime \prime}$ )
case True
have $P \oplus Q \Longrightarrow_{\tau} P \oplus Q$ by simp
moreover from $P^{\prime \prime \prime}$ Trans $\left\langle P=P^{\prime \prime \prime}\right\rangle$ have $P \oplus Q \longmapsto a<\nu x>\prec P^{\prime \prime}$ by (blast
intro: Sum1)
ultimately show ?thesis using $P^{\prime \prime}$ Chain by (rule transitionI)
next
case False
from PChain $\left\langle P \neq P^{\prime \prime \prime}\right\rangle$ have $P \oplus Q \Longrightarrow_{\tau} P^{\prime \prime \prime}$ by(rule sum1Chain)
thus ?thesis using $P^{\prime \prime \prime}$ Trans $P^{\prime \prime}$ Chain by (rule transitionI)
qed
next
assume $P \Longrightarrow \alpha \prec P^{\prime}$
then obtain $P^{\prime \prime} P^{\prime \prime \prime}$ where PChain: $P \Longrightarrow_{\tau} P^{\prime \prime \prime}$
and $P^{\prime \prime \prime}$ Trans: $P^{\prime \prime \prime} \longmapsto \alpha \prec P^{\prime \prime}$
and $P^{\prime \prime}$ Chain: $P^{\prime \prime} \Longrightarrow_{\tau} P^{\prime}$
by (force dest: transitionE)
show $P \oplus Q \Longrightarrow \alpha \prec P^{\prime}$
proof (cases $P=P^{\prime \prime \prime}$ )
case True
have $P \oplus Q \Longrightarrow_{\tau} P \oplus Q$ by simp
moreover from $P^{\prime \prime \prime}$ Trans $\left\langle P=P^{\prime \prime \prime}\right\rangle$ have $P \oplus Q \longmapsto \alpha \prec P^{\prime \prime}$ by (blast intro: Sum1)
ultimately show ?thesis using $P^{\prime \prime}$ Chain by(rule transitionI)

```
    next
        case False
        from PChain }\langleP\not=\mp@subsup{P}{}{\prime\prime\prime}>\mathrm{ have }P\oplusQ\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ by(rule sum1Chain)
        thus ?thesis using P}\mp@subsup{P}{}{\prime\prime}\mathrm{ Trans P}\mp@subsup{P}{}{\prime\prime}\mathrm{ Chain by(rule transitionI)
    qed
qed
lemma Sum2:
    fixes Q :: pi
    and a :: name
    and }x\mathrm{ :: name
    and }\mp@subsup{Q}{}{\prime}:: p
    and }P::p
    and \alpha :: freeRes
    shows }Q\Longrightarrowa<\nux>\prec\mp@subsup{Q}{}{\prime}\LongrightarrowP\oplusP\Longrightarrowa<\nux>\prec\mp@subsup{Q}{}{\prime
    and }Q\Longrightarrow\alpha\prec\mp@subsup{Q}{}{\prime}\LongrightarrowP\oplusQ\Longrightarrow\alpha\prec\mp@subsup{Q}{}{\prime
proof -
    assume }Q\Longrightarrowa<\nux>\prec\mp@subsup{Q}{}{\prime
    then obtain }\mp@subsup{Q}{}{\prime\prime}\mp@subsup{Q}{}{\prime\prime\prime}\mathrm{ where QChain: Q >}\mp@subsup{~}{\tau}{}\mp@subsup{Q}{}{\prime\prime\prime
                and }\mp@subsup{Q}{}{\prime\prime\prime}Trans: Q ''\prime \longmapstoa<\nux> \prec Q ''
                and Q'Chain: Q' }\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{Q}{}{\prime
    by(force dest: transitionE)
    show }P\oplusQ\Longrightarrowa<\nux>\prec\prec\mp@subsup{Q}{}{\prime
    proof(cases Q = Q '/')
    case True
    have }P\oplusQ\mp@subsup{\Longrightarrow}{\tau}{}P\oplusQ\mathrm{ by simp
    moreover from }\mp@subsup{Q}{}{\prime\prime\prime}\mathrm{ Trans <Q = Q '"'> have P }\oplusQ\longmapstoa<\nux>\prec\prec Q' by(blas
intro: Sum2)
    ultimately show ?thesis using Q"Chain by(rule transitionI)
    next
    case False
    from QChain }\langleQ\not=\mp@subsup{Q}{}{\prime\prime\prime}\rangle\mathrm{ have }P\oplusQ\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{Q}{}{\prime\prime\prime}\mathbf{by}(rule sum2Chain
    thus ?thesis using Q '/'Trans Q'Chain by(rule transitionI)
    qed
next
    assume Q\Longrightarrow\alpha\prec Q'
    then obtain }\mp@subsup{Q}{}{\prime\prime}\mp@subsup{Q}{}{\prime\prime\prime}\mathrm{ where QChain: }Q\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{Q}{}{\prime\prime\prime
                            and }\mp@subsup{Q}{}{\prime\prime\prime}\mathrm{ Trans: }\mp@subsup{Q}{}{\prime\prime\prime}\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime\prime
                            and Q"Chain: Q }\mp@subsup{Q}{}{\prime\prime}\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{Q}{}{\prime
    by(force dest: transitionE)
    show }P\oplusQ\Longrightarrow\alpha\prec\mp@subsup{Q}{}{\prime
    proof(cases Q = Q'/')
    case True
    have }P\oplusQ\mp@subsup{\Longrightarrow}{\tau}{}P\oplusQ\mathrm{ by simp
    moreover from }\mp@subsup{Q}{}{\prime\prime\prime}\mathrm{ Trans 〈Q = Q ''|> have }P\oplusQ\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime\prime}\mathrm{ by(blast intro:
Sum2)
    ultimately show ?thesis using Q'Chain by(rule transitionI)
next
```

```
    case False
    from QChain «Q \not= Q ''\prime>}\mathrm{ have P }\oplusQ\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{Q}{}{\prime\prime\prime}\mathrm{ by(rule sum2Chain)
    thus ?thesis using Q '/' Trans Q ''Chain by(rule transitionI)
    qed
qed
lemma Par1B:
    fixes P :: pi
    and a :: name
    and }x\mathrm{ :: name
    and }\mp@subsup{P}{}{\prime}:: p
    and }Q :: p
    assumes PTrans: }P\Longrightarrowa<\nux>\prec\mp@subsup{P}{}{\prime
    and }x\sharp
    shows }P|Q\Longrightarrowa<\nux>\prec(\mp@subsup{P}{}{\prime}|Q
proof -
    from PTrans obtain P'\prime P
                and }\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ Trans: }\mp@subsup{P}{}{\prime\prime\prime}\longmapstoa<\nux>\prec \prec P'
```



```
        by(blast dest: transitionE)
    from PChain have P|Q \Longrightarrow>}\mp@subsup{~}{}{\prime}\mp@subsup{P}{}{\prime\prime\prime}|Q\mathrm{ by(rule Par1Chain)
    moreover from }\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ Trans «x甘Q> have }\mp@subsup{P}{}{\prime\prime\prime}|Q\longmapstoa<\nux>\prec\prec(\mp@subsup{P}{}{\prime\prime}|Q)\mathrm{ by (rule
Early-Semantics.Par1B)
    moreover from P'Chain have }\mp@subsup{P}{}{\prime\prime}|Q\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime}|Q\mathrm{ by(rule Par1Chain)
    ultimately show P|Q\Longrightarrowa<\nux>}\prec(\mp@subsup{P}{}{\prime}|Q)\mathbf{by}(rule transitionI
qed
lemma Par1F:
    fixes P :: pi
    and \alpha :: freeRes
    and }\mp@subsup{P}{}{\prime}:: p
    and }Q :: p
    assumes PTrans: P\Longrightarrow\alpha\prec P
    shows }P|Q\Longrightarrow\alpha\prec(\mp@subsup{P}{}{\prime}|Q
proof -
    from PTrans obtain P}\mp@subsup{P}{}{\prime\prime}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ where PChain: }P\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime\prime\prime
                    and }\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ Trans: }\mp@subsup{P}{}{\prime\prime\prime}\longmapsto\alpha\prec\mp@subsup{P}{}{\prime\prime
                        and P'Chain: P'\prime}\Longrightarrow\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime
        by(blast dest: transitionE)
    from PChain have P|Q \Longrightarrow>}\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime\prime\prime}|Q\mathrm{ by(rule Par1Chain)
    moreover from P}\mp@subsup{P}{}{\prime\prime\prime}Trans have P P'\prime\prime| Q\longmapsto\alpha\prec(P'\prime|Q) by(rule Early-Semantics.Par1F)
    moreover from P'Chain have }\mp@subsup{P}{}{\prime\prime}|Q\Longrightarrow\Longrightarrow\tau P'| Q by(rule Par1Chain)
    ultimately show ?thesis by(rule transitionI)
qed
```

```
lemma Par2B:
    fixes \(Q\) :: \(p i\)
    and \(a\) :: name
    and \(x\) :: name
    and \(\quad Q^{\prime}:: p i\)
    and \(P:: p i\)
    assumes \(Q\) Trans: \(Q \Longrightarrow a<\nu x>\prec Q^{\prime}\)
    and \(\quad x \sharp P\)
    shows \(P \| Q \Longrightarrow a<\nu x>\prec\left(P \| Q^{\prime}\right)\)
proof -
    from QTrans obtain \(Q^{\prime \prime} Q^{\prime \prime \prime}\) where \(Q\) Chain: \(Q \Longrightarrow_{\tau} Q^{\prime \prime \prime}\)
                    and \(Q^{\prime \prime \prime}\) Trans: \(Q^{\prime \prime \prime} \longmapsto a<\nu x>\prec Q^{\prime \prime}\)
                        and \(Q^{\prime \prime}\) Chain: \(Q^{\prime \prime} \Longrightarrow_{\tau} Q^{\prime}\)
        by (blast dest: transitionE)
    from \(Q\) Chain have \(P\left\|Q \Longrightarrow_{\tau} P\right\| Q^{\prime \prime \prime}\) by(rule Par2Chain)
    moreover from \(Q^{\prime \prime \prime}\) Trans \(\langle x \sharp P\rangle\) have \(P \| Q^{\prime \prime \prime} \longmapsto a<\nu x>\prec\left(P \| Q^{\prime \prime}\right)\) by (rule
Early-Semantics.Par2B)
    moreover from \(Q^{\prime \prime}\) Chain have \(P\left\|Q^{\prime \prime} \Longrightarrow_{\tau} P\right\| Q^{\prime}\) by(rule Par2Chain)
    ultimately show \(P \| Q \Longrightarrow a<\nu x>\prec\left(P \| Q^{\prime}\right) \mathbf{b y}(\) rule transitionI)
qed
lemma Par2F:
    fixes \(Q:: p i\)
    and \(\alpha\) :: freeRes
    and \(\quad Q^{\prime}:: p i\)
    and \(P:: p i\)
    assumes \(Q\) Trans: \(Q \Longrightarrow \alpha \prec Q^{\prime}\)
    shows \(P \| Q \Longrightarrow \alpha \prec\left(P \| Q^{\prime}\right)\)
proof -
    from QTrans obtain \(Q^{\prime \prime} Q^{\prime \prime \prime}\) where \(Q\) Chain: \(Q \Longrightarrow_{\tau} Q^{\prime \prime \prime}\)
                    and \(Q^{\prime \prime \prime}\) Trans: \(Q^{\prime \prime \prime} \longmapsto \alpha \prec Q^{\prime \prime}\)
                            and \(Q^{\prime \prime}\) Chain: \(Q^{\prime \prime} \Longrightarrow_{\tau} Q^{\prime}\)
    by(blast dest: transitionE)
    from \(Q\) Chain have \(P\left\|Q \Longrightarrow_{\tau} P\right\| Q^{\prime \prime \prime}\) by(rule Par2Chain)
    moreover from \(Q^{\prime \prime \prime}\) Trans have \(P \| Q^{\prime \prime \prime} \longmapsto \alpha \prec\left(P \| Q^{\prime \prime}\right)\) by (rule Early-Semantics.Par2F)
    moreover from \(Q^{\prime \prime}\) Chain have \(P\left\|Q^{\prime \prime} \Longrightarrow_{\tau} P\right\| Q^{\prime}\) by(rule Par2Chain)
    ultimately show? ?thesis by (rule transitionI)
qed
lemma Comm1:
    fixes \(P:: p i\)
    and \(a\) :: name
    and \(b\) :: name
    and \(P^{\prime}:: p i\)
    and \(\quad Q\) :: pi
```

```
    and \(\quad Q^{\prime}:: p i\)
    assumes PTrans: \(P \Longrightarrow a<b>\prec P^{\prime}\)
    and \(\quad Q\) Trans: \(Q \Longrightarrow a[b] \prec Q^{\prime}\)
    shows \(P\left\|Q \Longrightarrow \tau \prec P^{\prime}\right\| Q^{\prime}\)
proof -
    from PTrans obtain \(P^{\prime \prime} P^{\prime \prime \prime}\) where PChain: \(P \Longrightarrow{ }_{\tau} P^{\prime \prime \prime}\)
                        and \(P^{\prime \prime \prime}\) Trans: \(P^{\prime \prime \prime} \longmapsto a<b>\prec P^{\prime \prime}\)
                        and \(P^{\prime \prime}\) Chain: \(P^{\prime \prime} \Longrightarrow_{\tau} P^{\prime}\)
        by(blast dest: transitionE)
    from \(Q\) Trans obtain \(Q^{\prime \prime} Q^{\prime \prime \prime}\) where QChain: \(Q \Longrightarrow_{\tau} Q^{\prime \prime \prime}\)
                            and \(Q^{\prime \prime \prime}\) Trans: \(Q^{\prime \prime \prime} \longmapsto a[b] \prec Q^{\prime \prime}\)
                            and \(Q^{\prime \prime}\) Chain: \(Q^{\prime \prime} \Longrightarrow_{\tau} Q^{\prime}\)
    by(blast dest: transitionE)
    from PChain QChain have \(P\left\|Q \Longrightarrow_{\tau} P^{\prime \prime \prime}\right\| Q^{\prime \prime \prime}\) by (rule chainPar)
    moreover from \(P^{\prime \prime \prime}\) Trans \(Q^{\prime \prime \prime}\) Trans have \(P^{\prime \prime \prime}\left\|Q^{\prime \prime \prime} \longmapsto \tau \prec P^{\prime \prime}\right\| Q^{\prime \prime}\)
    by (rule Early-Semantics.Comm1)
    moreover from \(P^{\prime \prime}\) Chain \(Q^{\prime \prime}\) Chain have \(P^{\prime \prime}\left\|Q^{\prime \prime} \Longrightarrow_{\tau} P^{\prime}\right\| Q^{\prime}\) by (rule chain-
Par)
    ultimately show ?thesis by(rule transitionI)
qed
lemma Comm2:
    fixes \(P\) :: \(p i\)
    and \(a\) :: name
    and \(b\) :: name
    and \(P^{\prime}:: p i\)
    and \(Q\) :: \(p i\)
    and \(Q^{\prime}:: p i\)
    assumes PTrans: \(P \Longrightarrow a[b] \prec P^{\prime}\)
    and \(\quad\) TTrans: \(Q \Longrightarrow a<b>\prec Q^{\prime}\)
    shows \(P\left\|Q \Longrightarrow \tau \prec P^{\prime}\right\| Q^{\prime}\)
proof -
    from PTrans obtain \(P^{\prime \prime} P^{\prime \prime \prime}\) where PChain: \(P \Longrightarrow_{\tau} P^{\prime \prime \prime}\)
                and \(P^{\prime \prime \prime}\) Trans: \(P^{\prime \prime \prime} \longmapsto a[b] \prec P^{\prime \prime}\)
                        and \(P^{\prime \prime}\) Chain: \(P^{\prime \prime} \Longrightarrow_{\tau} P^{\prime}\)
    \(\mathbf{b y}(\) blast dest: transitionE \()\)
    from QTrans obtain \(Q^{\prime \prime} Q^{\prime \prime \prime}\) where QChain: \(Q \Longrightarrow_{\tau} Q^{\prime \prime \prime}\)
        and \(Q^{\prime \prime \prime}\) Trans: \(Q^{\prime \prime \prime} \longmapsto a<b>\prec Q^{\prime \prime}\)
        and \(Q^{\prime \prime}\) Chain: \(Q^{\prime \prime} \Longrightarrow_{\tau} Q^{\prime}\)
    by (blast dest: transitionE)
    from PChain \(Q\) Chain have \(P\left\|Q \Longrightarrow_{\tau} P^{\prime \prime \prime}\right\| Q^{\prime \prime \prime}\) by (rule chainPar)
    moreover from \(P^{\prime \prime \prime}\) Trans \(Q^{\prime \prime \prime}\) Trans have \(P^{\prime \prime \prime}\left\|Q^{\prime \prime \prime} \longmapsto \tau \prec P^{\prime \prime}\right\| Q^{\prime \prime}\)
    by(rule Early-Semantics.Comm2)
```

```
    moreover from P ''Chain Q ''Chain have P'\prime| Q ' }\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime}|\mp@subsup{Q}{}{\prime}\mathrm{ by(rule chain-
Par)
    ultimately show ?thesis by(rule transitionI)
qed
lemma Close1:
    fixes P :: pi
    and a :: name
    and }x\mathrm{ :: name
    and }\mp@subsup{P}{}{\prime}:: p
    and }Q::p
    and }\mp@subsup{Q}{}{\prime}::p
    assumes PTrans: P\Longrightarrowa<x>\prec 琽
    and QTrans:Q\Longrightarrowa<\nux>}\prec\mp@subsup{Q}{}{\prime
    and }\quadx\sharp
    shows }P|Q\Longrightarrow\tau\prec<\nux>(\mp@subsup{P}{}{\prime}|\mp@subsup{Q}{}{\prime}
proof -
    from PTrans obtain P}\mp@subsup{P}{}{\prime\prime\prime}\mp@subsup{P}{}{\prime\prime}\mathrm{ where PChain: P ఋ>}\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime\prime\prime
                    and }\mp@subsup{P}{}{\prime\prime\prime}Trans: P'\prime\prime \longmapstoa<x> \prec P'\prime
```



```
    by(blast dest: transitionE)
from QTrans obtain }\mp@subsup{Q}{}{\prime\prime}\mp@subsup{Q}{}{\prime\prime\prime}\mathrm{ where QChain: Q }\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{Q}{}{\prime\prime\prime
                    and \mp@subsup{Q}{}{\prime\prime\prime}Trans: }\mp@subsup{Q}{}{\prime\prime\prime}\longmapstoa<\nux>\prec\prec\mp@subsup{Q}{}{\prime\prime
                            and Q"Chain: Q ' }\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{Q}{}{\prime
    by(blast dest: transitionE)
    from PChain QChain have P| Q \Longrightarrow\Longrightarrow\tau P}\mp@subsup{P}{}{\prime\prime\prime}|\mp@subsup{Q}{}{\prime\prime\prime}\mathrm{ by(rule chainPar)
    moreover from PChain <x\sharpP> have }x\sharp\mp@subsup{P}{}{\prime\prime\prime}\mathbf{by}(\mathrm{ rule freshChain)
```



```
    by(rule Early-Semantics.Close1)
    moreover from P'Chain Q ''Chain have P'|}|\mp@subsup{Q}{}{\prime\prime}\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime}|\mp@subsup{Q}{}{\prime}\mathrm{ by(rule chain-
Par)
    hence }<\nux>(\mp@subsup{P}{}{\prime\prime}|\mp@subsup{Q}{}{\prime\prime})\mp@subsup{\Longrightarrow}{\tau}{}<\nux>(\mp@subsup{P}{}{\prime}|\mp@subsup{Q}{}{\prime})\mathbf{by}(\mathrm{ rule ResChain)
    ultimately show ?thesis by(rule transitionI)
qed
lemma Close2:
fixes }P\mathrm{ :: pi
    and a :: name
    and }x\mathrm{ :: name
    and }\mp@subsup{P}{}{\prime}:: p
    and }Q ::p
    and }\mp@subsup{Q}{}{\prime}:: p
    assumes PTrans: P\Longrightarrowa<\nux>\prec }\mp@subsup{P}{}{\prime
    and QTrans: Q\Longrightarrowa<x>\prec\prec Q'
```

and $\quad x$ Fresh $Q: x \sharp Q$
shows $P \| Q \Longrightarrow \tau \prec<\nu x>\left(P^{\prime} \| Q^{\prime}\right)$ proof -
from PTrans obtain $P^{\prime \prime} P^{\prime \prime \prime}$ where PChain: $P \Longrightarrow_{\tau} P^{\prime \prime \prime}$ and $P^{\prime \prime \prime}$ Trans: $P^{\prime \prime \prime} \longmapsto a<\nu x>\prec P^{\prime \prime}$ and $P^{\prime \prime}$ Chain: $P^{\prime \prime} \Longrightarrow_{\tau} P^{\prime}$
by (blast dest: transitionE)
from QTrans obtain $Q^{\prime \prime} Q^{\prime \prime \prime}$ where $Q$ Chain: $Q \Longrightarrow_{\tau} Q^{\prime \prime \prime}$ and $Q^{\prime \prime \prime}$ Trans: $Q^{\prime \prime \prime} \longmapsto a<x>\prec Q^{\prime \prime}$ and $Q^{\prime \prime}$ Chain: $Q^{\prime \prime} \Longrightarrow_{\tau} Q^{\prime}$
by (blast dest: transitionE)
from PChain QChain have $P\left\|Q \Longrightarrow_{\tau} P^{\prime \prime \prime}\right\| Q^{\prime \prime \prime}$ by (rule chainPar)
moreover from $Q$ Chain $\langle x \sharp Q\rangle$ have $x \sharp Q^{\prime \prime \prime}$ by(rule freshChain)
with $P^{\prime \prime \prime}$ Trans $Q^{\prime \prime \prime}$ Trans have $P^{\prime \prime \prime} \| Q^{\prime \prime \prime} \longmapsto \tau \prec<\nu x>\left(P^{\prime \prime} \| Q^{\prime \prime}\right)$
by(rule Early-Semantics.Close2)
moreover from $P^{\prime \prime}$ Chain $Q^{\prime \prime}$ Chain have $P^{\prime \prime}\left\|Q^{\prime \prime} \Longrightarrow_{\tau} P^{\prime}\right\| Q^{\prime}$ by (rule chainPar)
hence $<\nu x>\left(P^{\prime \prime} \| Q^{\prime \prime}\right) \Longrightarrow_{\tau}<\nu x>\left(P^{\prime} \| Q^{\prime}\right)$ by(rule ResChain)
ultimately show ?thesis by (rule transitionI)
qed
lemma ResF:
fixes $P$ :: $p i$
and $\alpha$ :: freeRes
and $P^{\prime}:: p i$
and $x$ :: name
assumes PTrans: $P \Longrightarrow \alpha \prec P^{\prime}$
and $\quad x \sharp \alpha$
shows $<\nu x>P \Longrightarrow \alpha \prec<\nu x>P^{\prime}$
proof -
from PTrans obtain $P^{\prime \prime} P^{\prime \prime \prime}$ where PChain: $P \Longrightarrow_{\tau} P^{\prime \prime \prime}$ and $P^{\prime \prime \prime}$ Trans: $P^{\prime \prime \prime} \longmapsto \alpha \prec P^{\prime \prime}$ and $P^{\prime \prime}$ Chain: $P^{\prime \prime} \Longrightarrow_{\tau} P^{\prime}$
by (blast dest: transitionE)
from PChain have $<\nu x\rangle P \Longrightarrow_{\tau}\langle\nu x\rangle P^{\prime \prime \prime}$ by(rule ResChain)
moreover from $P^{\prime \prime \prime}$ Trans $\langle x \sharp \alpha\rangle$ have $<\nu x>P^{\prime \prime \prime} \longmapsto \alpha \prec<\nu x>P^{\prime \prime}$
by (rule Early-Semantics.ResF)
moreover from $P^{\prime \prime}$ Chain have $<\nu x>P^{\prime \prime} \Longrightarrow_{\tau}<\nu x>P^{\prime}$ by(rule ResChain)
ultimately show ?thesis by(rule transitionI)
qed
lemma ResB:
fixes $P$ :: $p i$

```
    and a :: name
    and x :: name
    and }\mp@subsup{P}{}{\prime}::p
    and y :: name
    assumes PTrans: P\Longrightarrowa<\nux>}\prec~\mp@subsup{P}{}{\prime
    and }y\not=
    and }y\not=
    shows <\nuy>P\Longrightarrowa<\nux>< < (<\nuy>P')
proof -
    from PTrans obtain P}\mp@subsup{P}{}{\prime\prime}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ where PChain: P }\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime\prime\prime
                        and }\mp@subsup{P}{}{\prime\prime\prime}Trans: P P'\prime\prime\longmapstoa<\nux> \prec P'\prime
                        and P'Chain: P'\prime}\Longrightarrow\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime
    by(blast dest: transitionE)
    from PChain have <\nuy>P \Longrightarrow\Longrightarrow}\mp@subsup{\tau}{\tau}{<\nuy>\mp@subsup{P}{}{\prime\prime\prime}}\mathrm{ by(rule ResChain)
    moreover from P P'\prime\primeTrans }\langley\not=a\rangle\langley\not=x\rangle\mathrm{ have < < y>> P'"' }\longmapstoa<\nux>\prec\prec
(<\nuy>P'\prime)
    by(rule Early-Semantics.ResB)
    moreover from }\mp@subsup{P}{}{\prime\prime}\mathrm{ Chain have < < y> P'' }\mp@subsup{\Longrightarrow}{\tau}{}<\nuy>\mp@subsup{P}{}{\prime}\mathrm{ by(rule ResChain)
    ultimately show ?thesis by(rule transitionI)
qed
lemma Bang:
    fixes P :: pi
    and Rs :: residual
    shows }P||P\Longrightarrowa<\nux>\prec \mp@subsup{P}{}{\prime}\Longrightarrow!P\Longrightarrowa<\nux>\prec \prec P
    and }P|!P\Longrightarrow\alpha\prec\mp@subsup{P}{}{\prime}\Longrightarrow!P\Longrightarrow\alpha\prec\mp@subsup{P}{}{\prime
proof -
    assume PTrans: P|!P\Longrightarrowa<\nux>}\prec\mp@subsup{P}{}{\prime
    from PTrans obtain P'\prime P'\prime\prime where PChain: P|!P \Longrightarrow\Longrightarrow}\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime\prime\prime
            and P}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ Trans: }\mp@subsup{P}{}{\prime\prime\prime}\longmapstoa<\nux>\prec\prec\mp@subsup{P}{}{\prime\prime
            and P}\mp@subsup{P}{}{\prime\prime}\mathrm{ Chain: }\mp@subsup{P}{}{\prime\prime}\Longrightarrow\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime
    by(force dest: transitionE)
    show !P\Longrightarrowa<\nux>}\prec\mp@subsup{P}{}{\prime
    proof(cases }\mp@subsup{P}{}{\prime\prime\prime}=P|!P
    case True
    have !P\Longrightarrow\Longrightarrow}\mp@subsup{\Longrightarrow}{\tau}{}!P\mathrm{ by simp
    moreover from }\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ Trans }\langle\mp@subsup{P}{}{\prime\prime\prime}=P|!P>\mathrm{ have !P}\longmapstoa<\nux>< < P'\prime by(blas
intro: Early-Semantics.Bang)
    ultimately show ?thesis using P'Chain by(rule transitionI)
    next
    case False
    from PChain }\langle\mp@subsup{P}{}{\prime\prime\prime}\not=P|!P>\mathrm{ have !P >>}\mp@subsup{~}{~}{}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ by(rule bangChain)
```

with $P^{\prime \prime \prime}$ Trans $P^{\prime \prime}$ Chain show ?thesis by(blast intro: transitionI)
qed
next
fix $\alpha P^{\prime} P$
assume $P \|!P \Longrightarrow \alpha \prec P^{\prime}$
then obtain $P^{\prime \prime} P^{\prime \prime \prime}$ where PChain: $P \|!P \Longrightarrow_{\tau} P^{\prime \prime}$
and $P^{\prime \prime}$ Trans: $P^{\prime \prime} \longmapsto \alpha \prec P^{\prime \prime \prime}$
and $P^{\prime \prime \prime}$ Chain: $P^{\prime \prime \prime} \Longrightarrow_{\tau} P^{\prime}$
by (force dest: transitionE)

```
    show ! P\Longrightarrow\alpha\prec P'
    proof(cases }\mp@subsup{P}{}{\prime\prime}=P|!P
    assume P }\mp@subsup{P}{}{\prime\prime}=P|!
    moreover with P'\primeTrans have ! P\longmapsto\alpha \prec P'\prime\prime by(blast intro: Bang)
        ultimately show ?thesis using PChain P}\mp@subsup{}{\prime\prime\prime}{\prime\prime}\mathrm{ Chain by(rule-tac transitionI,
auto)
    next
    assume }\mp@subsup{P}{}{\prime\prime}\not=P|!
    with PChain have !P\Longrightarrow\Longrightarrow}\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime\prime}\mathrm{ by(rule bangChain)
    with P}\mp@subsup{P}{}{\prime\prime}\mathrm{ Trans P}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ Chain show ?thesis by(blast intro: transitionI)
    qed
qed
lemma tauTransitionChain:
    fixes P :: pi
    and }\mp@subsup{P}{}{\prime}::p
    assumes }P\Longrightarrow\tau\prec\mp@subsup{P}{}{\prime
    shows }P\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime
using assms
by(force dest: transitionE tauActTauChain)
lemma chainTransitionAppend:
    fixes P :: pi
    and }\mp@subsup{P}{}{\prime}::p
    and Rs :: residual
    and a :: name
    and }x\mathrm{ :: name
    and }\mp@subsup{P}{}{\prime\prime}:: p
    and \alpha :: freeRes
    shows }P\Longrightarrowa<\nux>\prec\mp@subsup{P}{}{\prime\prime}\Longrightarrow\mp@subsup{P}{}{\prime\prime}\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime}\LongrightarrowP\Longrightarrowa<\nux>\prec\prec P
    and }P\Longrightarrow\alpha\prec\mp@subsup{P}{}{\prime\prime}\Longrightarrow\mp@subsup{P}{}{\prime\prime}\Longrightarrow\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime}\LongrightarrowP\Longrightarrow\alpha\prec\mp@subsup{P}{}{\prime
    and }P\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime\prime}\Longrightarrow\mp@subsup{P}{}{\prime\prime}\Longrightarrowa<\nux>\prec\prec\mp@subsup{P}{}{\prime}\LongrightarrowP\Longrightarrowa<\nux>\prec\prec P
    and }P\Longrightarrow\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime\prime}\Longrightarrow\mp@subsup{P}{}{\prime\prime}\Longrightarrow\alpha\prec\mp@subsup{P}{}{\prime}\LongrightarrowP\Longrightarrow\alpha\prec\mp@subsup{P}{}{\prime
proof -
    assume PTrans: P\Longrightarrowa<\nux>\prec 事
```

```
    assume \(P^{\prime \prime}\) Chain: \(P^{\prime \prime} \Longrightarrow_{\tau} P^{\prime}\)
    from PTrans obtain \(P^{\prime \prime \prime} P^{\prime \prime \prime \prime}\) where PChain: \(P \Longrightarrow_{\tau} P^{\prime \prime \prime \prime}\)
                and \(P^{\prime \prime \prime \prime}\) Trans: \(P^{\prime \prime \prime \prime} \longmapsto a<\nu x>\prec P^{\prime \prime \prime}\)
                and \(P^{\prime \prime \prime}\) Chain: \(P^{\prime \prime \prime} \Longrightarrow_{\tau} P^{\prime \prime}\)
    by (blast dest: transitionE)
    from \(P^{\prime \prime \prime}\) Chain \(P^{\prime \prime}\) Chain have \(P^{\prime \prime \prime} \Longrightarrow_{\tau} P^{\prime}\) by auto
    with PChain \(P^{\prime \prime \prime \prime}\) Trans show \(P \Longrightarrow a<\nu x>\prec P^{\prime}\) by (rule transitionI)
next
    assume PTrans: \(P \Longrightarrow \alpha \prec P^{\prime \prime}\)
    assume \(P^{\prime \prime}\) Chain: \(P^{\prime \prime} \Longrightarrow_{\tau} P^{\prime}\)
    from PTrans obtain \(P^{\prime \prime \prime} P^{\prime \prime \prime \prime}\) where PChain: \(P \Longrightarrow_{\tau} P^{\prime \prime \prime \prime}\)
                and \(P^{\prime \prime \prime \prime}\) Trans: \(P^{\prime \prime \prime \prime} \longmapsto \alpha \prec P^{\prime \prime \prime}\)
                and \(P^{\prime \prime \prime}\) Chain: \(P^{\prime \prime \prime} \Longrightarrow_{\tau} P^{\prime \prime}\)
    by (blast dest: transitionE)
    from \(P^{\prime \prime \prime}\) Chain \(P^{\prime \prime}\) Chain have \(P^{\prime \prime \prime} \Longrightarrow_{\tau} P^{\prime}\) by auto
    with PChain \(P^{\prime \prime \prime \prime}\) Trans show \(P \Longrightarrow \alpha \prec P^{\prime}\) by(rule transitionI)
next
    assume PChain: \(P \Longrightarrow_{\tau} P^{\prime \prime}\)
    assume \(P^{\prime \prime}\) Trans: \(P^{\prime \prime} \Longrightarrow a<\nu x>\prec P^{\prime}\)
    from \(P^{\prime \prime}\) Trans obtain \(P^{\prime \prime \prime} P^{\prime \prime \prime \prime}\) where \(P^{\prime \prime}\) Chain: \(P^{\prime \prime} \Longrightarrow_{\tau} P^{\prime \prime \prime \prime}\)
                                    and \(P^{\prime \prime \prime \prime}\) Trans: \(P^{\prime \prime \prime \prime} \longmapsto a<\nu x>\prec P^{\prime \prime \prime}\)
                                    and \(P^{\prime \prime \prime}\) Chain: \(P^{\prime \prime \prime} \Longrightarrow{ }_{\tau} P^{\prime}\)
    by (blast dest: transitionE)
    from PChain \(P^{\prime \prime}\) Chain have \(P \Longrightarrow_{\tau} P^{\prime \prime \prime \prime}\) by auto
    thus \(P \Longrightarrow a<\nu x>\prec P^{\prime}\) using \(P^{\prime \prime \prime \prime}\) Trans \(P^{\prime \prime \prime}\) Chain by (rule transitionI)
next
    assume PChain: \(P \Longrightarrow_{\tau} P^{\prime \prime}\)
    assume \(P^{\prime \prime}\) Trans: \(P^{\prime \prime} \Longrightarrow \alpha \prec P^{\prime}\)
    from \(P^{\prime \prime}\) Trans obtain \(P^{\prime \prime \prime} P^{\prime \prime \prime \prime}\) where \(P^{\prime \prime}\) Chain: \(P^{\prime \prime} \Longrightarrow_{\tau} P^{\prime \prime \prime \prime}\)
                        and \(P^{\prime \prime \prime \prime}\) Trans: \(P^{\prime \prime \prime \prime} \longmapsto \alpha \prec P^{\prime \prime \prime}\)
                        and \(P^{\prime \prime \prime}\) Chain: \(P^{\prime \prime \prime} \Longrightarrow{ }_{\tau} P^{\prime}\)
    by(blast dest: transitionE)
    from PChain \(P^{\prime \prime}\) Chain have \(P \Longrightarrow_{\tau} P^{\prime \prime \prime \prime}\) by auto
    thus \(P \Longrightarrow \alpha \prec P^{\prime}\) using \(P^{\prime \prime \prime \prime}\) Trans \(P^{\prime \prime \prime}\) Chain by(rule transitionI)
qed
lemma freshBoundOutputTransition:
    fixes \(P:: p i\)
    and \(a\) :: name
    and \(x\) :: name
    and \(\quad P^{\prime}:: p i\)
```

```
    and c :: name
    assumes PTrans: P\Longrightarrowa<\nux>}\prec\mp@subsup{P}{}{\prime
    and c}c\sharp
    and c}c\not=
    shows c\sharp P'
proof -
    from PTrans obtain P'\prime P}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ where PChain: P ఋ>}\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime\prime\prime
                and }\mp@subsup{P}{}{\prime\prime\prime}Trans: P ''\prime \longmapstoa<\nux> \prec P'\prime
```



```
    by(blast dest: transitionE)
    from PChain 〈c\sharpP> have c\sharp 敖'\prime by(rule freshChain)
    with P}\mp@subsup{P}{}{\prime\prime\prime}Trans have c\sharp\mp@subsup{P}{}{\prime\prime}\mathrm{ using }<c\not=x\rangle\mathrm{ by(rule Early-Semantics.freshTransition)
    with P}\mp@subsup{P}{}{\prime\prime}\mathrm{ Chain show }c\sharp\mp@subsup{P}{}{\prime}\mathrm{ by(rule freshChain)
qed
lemma freshTauTransition:
    fixes P :: pi
    and c :: name
    assumes PTrans: P\Longrightarrow\tau\prec 缶
    and c}c\sharp
    shows c\sharp 部
proof -
    from PTrans have P \Longrightarrow}\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime}\mathrm{ by(rule tauTransitionChain)
    thus ?thesis using <c\sharpP> by(rule freshChain)
qed
lemma freshOutputTransition:
    fixes P :: pi
    and a :: name
    and b :: name
    and }\mp@subsup{P}{}{\prime}:: p
    and c :: name
    assumes PTrans: P\Longrightarrowa[b]\prec }\mp@subsup{P}{}{\prime
    and c}c\sharp
    shows c\sharp 犃
proof -
    from PTrans obtain }\mp@subsup{P}{}{\prime\prime}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ where PChain: P }\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime\prime\prime
                        and }\mp@subsup{P}{}{\prime\prime\prime}Trans: P'\prime\prime \longmapstoa[b]\prec \prec ''
                        and P''Chain: P'\prime}\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime
        by(blast dest: transitionE)
```

    from PChain \(\langle c \sharp P\rangle\) have \(c \sharp P^{\prime \prime \prime}\) by(rule freshChain)
    with $P^{\prime \prime \prime}$ Trans have $c \sharp P^{\prime \prime}$ by (rule Early-Semantics.freshTransition) with $P^{\prime \prime}$ Chain show ?thesis $\mathbf{b y}$ (rule freshChain)
qed
lemma eqvtI[eqvt]:
fixes $P:: p i$
and $a$ :: name
and $x$ :: name
and $P^{\prime}:: p i$
and $p::$ name prm
and $\quad \alpha::$ freeRes

```
    shows \(P \Longrightarrow a<\nu x>\prec P^{\prime} \Longrightarrow(p \cdot P) \Longrightarrow(p \cdot a)<\nu(p \cdot x)>\prec\left(p \cdot P^{\prime}\right)\)
    and \(P \Longrightarrow \alpha \prec P^{\prime} \Longrightarrow(p \cdot P) \Longrightarrow(p \cdot \alpha) \prec\left(p \cdot P^{\prime}\right)\)
proof -
    assume \(P \Longrightarrow a<\nu x>\prec P^{\prime}\)
    then obtain \(P^{\prime \prime} P^{\prime \prime \prime}\) where PChain: \(P \Longrightarrow{ }_{\tau} P^{\prime \prime \prime}\)
            and \(P^{\prime \prime \prime}\) Trans: \(P^{\prime \prime \prime} \longmapsto a<\nu x>\prec P^{\prime \prime}\)
            and \(P^{\prime \prime}\) Chain: \(P^{\prime \prime} \Longrightarrow_{\tau} P^{\prime}\)
    by (blast dest: transitionE)
    from PChain have \((p \cdot P) \Longrightarrow_{\tau}\left(p \cdot P^{\prime \prime \prime}\right)\) by (rule eqvtChainI)
    moreover from \(P^{\prime \prime \prime}\) Trans have \(\left(p \cdot P^{\prime \prime \prime}\right) \longmapsto\left(p \cdot\left(a<\nu x>\prec P^{\prime \prime}\right)\right)\)
        by (rule TransitionsEarly.eqvt)
    hence \(\left(p \cdot P^{\prime \prime \prime}\right) \longmapsto(p \cdot a)<\nu(p \cdot x)>\prec\left(p \cdot P^{\prime \prime}\right)\)
    by (simp add: equts)
    moreover from \(P^{\prime \prime}\) Chain have \(\left(p \cdot P^{\prime \prime}\right) \Longrightarrow_{\tau}\left(p \cdot P^{\prime}\right)\) by (rule eqvtChainI)
    ultimately show \((p \cdot P) \Longrightarrow(p \cdot a)<\nu(p \cdot x)>\prec\left(p \cdot P^{\prime}\right)\)
        by (rule transitionI)
next
    assume \(P \Longrightarrow \alpha \prec P^{\prime}\)
    then obtain \(P^{\prime \prime} P^{\prime \prime \prime}\) where PChain: \(P \Longrightarrow_{\tau} P^{\prime \prime \prime}\)
                    and \(P^{\prime \prime \prime}\) Trans: \(P^{\prime \prime \prime} \longmapsto \alpha \prec P^{\prime \prime}\)
                    and \(P^{\prime \prime}\) Chain: \(P^{\prime \prime} \Longrightarrow_{\tau} P^{\prime}\)
    by(blast dest: transitionE)
    from PChain have \((p \cdot P) \Longrightarrow_{\tau}\left(p \cdot P^{\prime \prime \prime}\right)\) by (rule eqvtChainI)
    moreover from \(P^{\prime \prime \prime}\) Trans have \(\left(p \cdot P^{\prime \prime \prime}\right) \longmapsto\left(p \cdot\left(\alpha \prec P^{\prime \prime}\right)\right)\)
    by (rule TransitionsEarly.eqvt)
    hence \(\left(p \cdot P^{\prime \prime \prime}\right) \longmapsto(p \cdot \alpha) \prec\left(p \cdot P^{\prime \prime}\right)\)
    by (simp add: eqvts)
    moreover from \(P^{\prime \prime}\) Chain have \(\left(p \cdot P^{\prime \prime}\right) \Longrightarrow_{\tau}\left(p \cdot P^{\prime}\right)\) by (rule eqvtChainI)
    ultimately show \((p \cdot P) \Longrightarrow(p \cdot \alpha) \prec\left(p \cdot P^{\prime}\right)\)
    by (rule transitionI)
qed
lemma freshInputTransition:
    fixes \(P\) :: pi
    and \(a\) :: name
```

```
    and b :: name
    and }\mp@subsup{P}{}{\prime}::p
    and c :: name
    assumes PTrans: }P\Longrightarrowa<b>\prec\mp@subsup{P}{}{\prime
    and c}c\sharp
    and c\not=b
    shows c\sharp P'
proof -
    from PTrans obtain P}\mp@subsup{P}{}{\prime\prime}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ where PChain: P }\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime\prime\prime
                        and P P'\prime\primeTrans: }\mp@subsup{P}{}{\prime\prime\prime}\longmapstoa<b>\prec\mp@subsup{P}{}{\prime\prime
                        and P'Chain: P'\prime}\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime
    by(blast dest: transitionE)
    from PChain \langlec\sharpP> have c\sharp 敖\prime\prime by(rule freshChain)
    with P}\mp@subsup{P}{}{\prime\prime\prime}Trans have c\sharp\mp@subsup{P}{}{\prime\prime}\mathbf{using}\langlec\not=b\rangle\mathbf{by}(rule Early-Semantics.freshInputTransition)
    with P''Chain show ?thesis by(rule freshChain)
qed
lemmas freshTransition = freshBoundOutputTransition freshOutputTransition
    freshInputTransition freshTauTransition
end
theory Weak-Early-Semantics
    imports Weak-Early-Step-Semantics
begin
definition weakFreeTransition :: pi=> freeRes }=>\mathrm{ pi mbool(- # - `-[80, 80,
80] 80)
    where P\Longrightarrow}\alpha\prec\mp@subsup{P}{}{\prime}\equivP\Longrightarrow\alpha\prec\mp@subsup{P}{}{\prime}\vee(\alpha=\tau\wedgeP=\mp@subsup{P}{}{\prime}
lemma weakTransitionI:
    fixes P :: pi
    and \alpha :: freeRes
    and }\mp@subsup{P}{}{\prime}:: p
    shows }P\Longrightarrow\alpha\prec\mp@subsup{P}{}{\prime}\LongrightarrowP\Longrightarrow\alpha\prec\mp@subsup{P}{}{\prime
    and }P\Longrightarrow^\imath\prec
by(auto simp add: weakFreeTransition-def)
lemma transitionCases[consumes 1, case-names Step Stay]:
    fixes P :: pi
    and }\alpha\mathrm{ :: freeRes
    and }\mp@subsup{P}{}{\prime}::p
    assumes P\Longrightarrow\alpha\prec 㐌
    and }P\Longrightarrow\alpha\prec\mp@subsup{P}{}{\prime}\LongrightarrowF\alpha\mp@subsup{P}{}{\prime
```

```
    and }\quadF(\tau)
    shows F \alpha P'
using assms
by(auto simp add: weakFreeTransition-def)
lemma singleActionChain:
    fixes P :: pi
    and \alpha :: freeRes
    and }\mp@subsup{P}{}{\prime}:: p
    assumes }P\longmapsto\alpha\prec\mp@subsup{P}{}{\prime
    shows P\Longrightarrow^ }\alpha\prec\mp@subsup{P}{}{\prime
using assms
by(auto dest: singleActionChain intro: weakTransitionI)
lemma Tau:
    fixes P :: pi
    shows }\tau.(P)\Longrightarrow^\tau\prec 
by(auto intro:Weak-Early-Step-Semantics.Tau
    simp add: weakFreeTransition-def)
lemma Input:
    fixes a :: name
    and x :: name
    and u:: name
    and }P::p
    shows }a<x>.P\Longrightarrowa<u>\prec \prec [x::=u
by(auto intro:Weak-Early-Step-Semantics.Input
    simp add: weakFreeTransition-def)
lemma Output:
    fixes a :: name
    and b:: name
    and }P::p
    shows }a{b}.P\Longrightarrow^a[b]\prec
by(auto intro:Weak-Early-Step-Semantics.Output
    simp add: weakFreeTransition-def)
lemma Par1F:
    fixes P :: pi
    and }\alpha\mathrm{ :: freeRes
    and }\mp@subsup{P}{}{\prime}::p
    and }Q ::p
```

```
    assumes \(P \Longrightarrow \hat{\wedge} \alpha P^{\prime}\)
    shows \(P \| Q \Longrightarrow{ }^{\wedge} \alpha \prec\left(P^{\prime} \| Q\right)\)
using assms
by (auto intro: Weak-Early-Step-Semantics.Par1F
    simp add: weakFreeTransition-def residual.inject)
lemma Par2F:
    fixes \(Q\) :: \(p i\)
    and \(\alpha\) :: freeRes
    and \(\quad Q^{\prime}:: p i\)
    and \(P:: p i\)
    assumes \(Q\) Trans: \(Q \Longrightarrow{ }^{\wedge} \alpha \prec Q^{\prime}\)
    shows \(P \| Q \Longrightarrow \alpha \prec\left(P \| Q^{\prime}\right)\)
using assms
by (auto intro: Weak-Early-Step-Semantics.Par2F
    simp add: weakFreeTransition-def residual.inject)
lemma ResF:
    fixes \(P:: p i\)
    and \(\alpha\) :: freeRes
    and \(P^{\prime}:: p i\)
    and \(x\) :: name
    assumes \(P \Longrightarrow \alpha \prec P^{\prime}\)
    and \(\quad x \sharp \alpha\)
    shows \(<\nu x>P \Longrightarrow \wedge \prec<\nu x>P^{\prime}\)
using assms
by(auto intro: Weak-Early-Step-Semantics.ResF
    simp add: weakFreeTransition-def residual.inject)
lemma Bang:
    fixes \(P:: p i\)
    and \(R s::\) residual
    assumes \(P \|!P \Longrightarrow \wedge \prec P^{\prime}\)
    and \(\quad P^{\prime} \neq P \|!P\)
    shows \(!P \Longrightarrow \alpha \prec P^{\prime}\)
using assms
by (auto intro: Weak-Early-Step-Semantics.Bang
    simp add: weakFreeTransition-def residual.inject)
lemma tauTransitionChain[simp]:
    fixes \(P:: p i\)
```

```
    and }\mp@subsup{P}{}{\prime}::p
    shows }P\Longrightarrow\hat{`}\tau\prec\mp@subsup{P}{}{\prime}=P\Longrightarrow\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime
apply(auto dest:Weak-Early-Step-Semantics.tauTransitionChain
    simp add: weakFreeTransition-def)
by(erule rtrancl.cases) (auto intro: transitionI)
lemma tauStepTransitionChain[simp]:
    fixes P :: pi
    and }\mp@subsup{P}{}{\prime}:: p
    assumes P\not= P'
    shows }P\Longrightarrow\tau\prec\mp@subsup{P}{}{\prime}=P\Longrightarrow\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime
using assms
apply(auto dest: Weak-Early-Step-Semantics.tauTransitionChain
    simp add: weakFreeTransition-def)
by(erule rtrancl.cases) (auto intro: transitionI)
lemma chainTransitionAppend:
    fixes P :: pi
    and }\mp@subsup{P}{}{\prime}::p
    and Rs :: residual
    and a :: name
    and }x\mathrm{ :: name
    and }\mp@subsup{P}{}{\prime\prime}:: p
    and \alpha :: freeRes
```

    shows \(P \Longrightarrow_{\tau} P^{\prime \prime} \Longrightarrow P^{\prime \prime} \Longrightarrow \hat{\wedge} \alpha P^{\prime} \Longrightarrow P \Longrightarrow \hat{\rho^{\prime}} \alpha \prec P^{\prime}\)
    and \(P \Longrightarrow \alpha \prec P^{\prime \prime} \Longrightarrow P^{\prime \prime} \Longrightarrow_{\tau} P^{\prime} \Longrightarrow P \Longrightarrow \alpha \prec P^{\prime}\)
    by (auto intro: chainTransitionAppend simp add: weakFreeTransition-def dest: Weak-Early-Step-Semantics.tau?
lemma freshTauTransition:
fixes $P$ :: $p i$
and $c::$ name
assumes $P \Longrightarrow \wedge{ }^{\wedge} \tau \not P^{\prime}$
and $\quad c \sharp P$
shows $c \sharp P^{\prime}$
using assms
by(auto intro: Weak-Early-Step-Semantics.freshTauTransition
simp add: weakFreeTransition-def residual.inject)
lemma freshOutputTransition:
fixes $P:: p i$
and $a$ :: name
and $b$ :: name
and $\quad P^{\prime}:: p i$
and $c$ :: name
assumes $P \Longrightarrow{ }^{\wedge} a[b] \prec P^{\prime}$
and $\quad c \sharp P$
shows $c \sharp P^{\prime}$
using assms
by (auto intro: Weak-Early-Step-Semantics.freshOutputTransition simp add: weakFreeTransition-def residual.inject)
lemma eqvtI:
fixes $P$ :: $p i$
and $\alpha$ :: freeRes
and $P^{\prime}:: p i$
and $\quad p::$ name prm
assumes $P \Longrightarrow{ }^{\wedge} \alpha \prec P^{\prime}$
shows $(p \cdot P) \Longrightarrow \wedge(p \cdot \alpha) \prec\left(p \cdot P^{\prime}\right)$
using assms
by (auto intro: Weak-Early-Step-Semantics.eqvtI
simp add: weakFreeTransition-def residual.inject)
lemma freshInputTransition:
fixes $P:: p i$
and $a$ :: name
and $b$ :: name
and $P^{\prime}:: p i$
and $c::$ name
assumes $P \Longrightarrow{ }^{\wedge} a<b>\prec P^{\prime}$
and $\quad c \sharp P$
and $c \neq b$
shows $c \sharp P^{\prime}$
using assms
by (auto intro: Weak-Early-Step-Semantics.freshInputTransition
simp add: weakFreeTransition-def residual.inject)
lemmas freshTransition $=$ freshBoundOutputTransition freshOutputTransition freshInputTransition freshTauTransition
end
theory Weak-Early-Sim
imports Weak-Early-Semantics Strong-Early-Sim-Pres
begin
definition weakSimulation $:: p i \Rightarrow(p i \times p i)$ set $\Rightarrow p i \Rightarrow$ bool $(-\rightsquigarrow<->-[80,80$,

80] 80$)$
where $P \rightsquigarrow<\operatorname{Rel}>Q \equiv\left(\forall a x Q^{\prime} . Q \longmapsto a<\nu x>\prec Q^{\prime} \wedge x \sharp P \longrightarrow\left(\exists P^{\prime} . P\right.\right.$ $\left.\left.\Longrightarrow a<\nu x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}\right)\right) \wedge$ $\left(\forall \alpha Q^{\prime} . Q \longmapsto \alpha \prec Q^{\prime} \longrightarrow\left(\exists P^{\prime} . P \Longrightarrow \wedge \prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in\right.\right.$ Rel))
lemma monotonic:
fixes $A::(p i \times p i)$ set
and $B::(p i \times p i)$ set
and $P:: p i$
and $P^{\prime}:: p i$
assumes $P \rightsquigarrow<A>P^{\prime}$
and $A \subseteq B$
shows $P \rightsquigarrow<B>P^{\prime}$
using assms
by (simp add: weakSimulation-def) blast
lemma simCasesCont[consumes 1, case-names Bound Free]:
fixes $P$ :: $p i$
and $\quad Q \quad:: p i$
and Rel $::(p i \times p i)$ set
and $C$ :: 'a::fs-name
assumes Eqvt: eqvt Rel
and Bound: $\bigwedge$ ax $Q^{\prime}$. $\llbracket Q \longmapsto a<\nu x>\prec Q^{\prime} ; x \sharp P ; x \sharp Q ; x \neq a ; x \sharp C \rrbracket \Longrightarrow$ $\exists P^{\prime} . P \Longrightarrow a<\nu x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}$
and Free: $\bigwedge \alpha Q^{\prime} . Q \longmapsto \alpha \prec Q^{\prime} \Longrightarrow \exists P^{\prime} . P \Longrightarrow \alpha \prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}$
shows $P \rightsquigarrow<$ Rel $>Q$
proof (auto simp add: weakSimulation-def)
fix $a x Q^{\prime}$
assume $Q$ Trans: $Q \longmapsto a<\nu x>\prec Q^{\prime}$ and $x \sharp P$
obtain $c:$ :name where $c \sharp P$ and $c \sharp Q$ and $c \neq a$ and $c \sharp Q^{\prime}$ and $c \sharp C$ and $c \neq x$
by(generate-fresh name) auto
from QTrans $\left\langle c \sharp Q^{\prime}\right\rangle$ have $Q \longmapsto a<\nu c>\prec\left([(x, c)] \cdot Q^{\prime}\right)$ by $(\operatorname{simp}$ add: alphaBoundOutput)
then obtain $P^{\prime}$ where PTrans: $P \Longrightarrow a<\nu c>\prec P^{\prime}$ and $P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime},[(x, c)]\right.$

- $\left.Q^{\prime}\right) \in$ Rel
using $\langle c \sharp P\rangle\langle c \sharp Q\rangle\langle c \neq a\rangle\langle c \sharp C\rangle$
by(drule-tac Bound) auto
from PTrans $\langle x \sharp P\rangle\langle c \neq x\rangle$ have $P \Longrightarrow a<\nu x\rangle \prec\left([(x, c)] \cdot P^{\prime}\right)$
by (force intro: weakTransitionAlpha simp add: name-swap)
moreover from Eqvt $P^{\prime} \operatorname{Rel} Q^{\prime}$ have $\left([(x, c)] \cdot P^{\prime},[(x, c)] \cdot[(x, c)] \cdot Q^{\prime}\right) \in \operatorname{Rel}$
by (rule eqvtRelI)

```
    hence \(\left([(x, c)] \cdot P^{\prime}, Q^{\prime}\right) \in\) Rel by simp
    ultimately show \(\exists P^{\prime} . P \Longrightarrow a<\nu x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in\) Rel
        by blast
next
    fix \(\alpha Q^{\prime}\)
    assume \(Q \longmapsto \alpha \prec Q^{\prime}\)
    thus \(\exists P^{\prime} . P \Longrightarrow \wedge \prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in\) Rel
        by (rule Free)
qed
lemma simCases[case-names Bound Free]:
    fixes \(P\) :: \(p i\)
    and \(\quad Q \quad:: p i\)
    and Rel :: \((p i \times p i)\) set
    and \(C\) :: 'a::fs-name
    assumes \(\bigwedge Q^{\prime}\) a \(x . \llbracket Q \longmapsto a<\nu x>\prec Q^{\prime} ; x \sharp P \rrbracket \Longrightarrow \exists P^{\prime} . P \Longrightarrow a<\nu x>\prec P^{\prime} \wedge\)
\(\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}\)
    and \(\quad \wedge Q^{\prime} \alpha . Q \longmapsto \alpha \prec Q^{\prime} \Longrightarrow \exists P^{\prime} . P \Longrightarrow \wedge \prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in\) Rel
    shows \(P \rightsquigarrow<\) Rel \(>Q\)
using assms
by (auto simp add: weakSimulation-def)
lemma \(\operatorname{sim} E\) :
    fixes \(P\) :: \(p i\)
    and Rel \(::(p i \times p i)\) set
    and \(\quad Q \quad:: p i\)
    and \(a\) :: name
    and \(x\) :: name
    and \(Q^{\prime}:: p i\)
    assumes \(P \rightsquigarrow<\) Rel \(>Q\)
    shows \(Q \longmapsto a<\nu x>\prec Q^{\prime} \Longrightarrow x \sharp P \Longrightarrow \exists P^{\prime} . P \Longrightarrow a<\nu x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in\)
Rel
    and \(\quad Q \longmapsto \alpha \prec Q^{\prime} \Longrightarrow \exists P^{\prime} . P \Longrightarrow \alpha \prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in\) Rel
using assms by (simp add: weakSimulation-def) +
lemma weakSimTauChain:
fixes \(P\) :: \(p i\)
and Rel :: \((p i \times p i)\) set
and Rel' \(::(p i \times p i)\) set
and \(\quad Q \quad:: p i\)
and \(Q^{\prime}:: p i\)
assumes QChain: \(Q \Longrightarrow_{\tau} Q^{\prime}\)
and \(\quad P R e l Q:(P, Q) \in\) Rel
and \(\quad P S i m Q: \bigwedge R S .(R, S) \in \operatorname{Rel} \Longrightarrow R \rightsquigarrow<\operatorname{Rel}>S\)
```

```
    shows \exists\mp@subsup{P}{}{\prime}.P\Longrightarrow\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mathrm{ Rel}
proof -
    from QChain show ?thesis
    proof(induct rule: tauChainInduct)
        case id
        moreover have P \Longrightarrow>
        ultimately show ?case using PSimQ PRelQ by blast
    next
        case(ih Q' Q '/)
        have }\exists\mp@subsup{P}{}{\prime}.P\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mathrm{ Rel by fact
    then obtain }\mp@subsup{P}{}{\prime}\mathrm{ where PChain: P ב>}\mp@subsup{\tau}{}{\prime}\mp@subsup{P}{}{\prime}\mathrm{ and }\mp@subsup{P}{}{\prime}\mp@subsup{R}{el}{\prime}\mp@subsup{Q}{}{\prime}:(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mathrm{ Rel by
blast
    from P'Rel'Q}\mp@subsup{Q}{}{\prime}\mathrm{ have }\mp@subsup{P}{}{\prime}\rightsquigarrow<\mathrm{ Rel > Q 㐌 by (rule PSimQ)
    moreover have }\mp@subsup{Q}{}{\prime}\mathrm{ Trans: }\mp@subsup{Q}{}{\prime}\longmapsto\tau\prec\mp@subsup{Q}{}{\prime\prime}\mathrm{ by fact
    ultimately obtain }\mp@subsup{P}{}{\prime\prime}\mathrm{ where }\mp@subsup{P}{}{\prime}\mathrm{ Trans: }\mp@subsup{P}{}{\prime}\Longrightarrow^\tau\prec\mp@subsup{P}{}{\prime\prime}\mathrm{ and }\mp@subsup{P}{}{\prime\prime}RelQ\mp@subsup{Q}{}{\prime\prime}:(\mp@subsup{P}{}{\prime\prime}
Q')}\in\operatorname{Rel
    by(blast dest: simE)
    from P'Trans have P' }\mp@subsup{P}{}{\prime}\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime\prime}\mathrm{ by simp
    with PChain have P \Longrightarrow}\mp@subsup{}{\tau}{}\mp@subsup{P}{}{\prime\prime}\mathrm{ by auto
    with P'RelQ'\prime show ?case by blast
    qed
qed
lemma simE2:
    fixes P :: pi
    and Rel :: (pi\timespi) set
    and }Q ::p
    and a :: name
    and }x\mathrm{ :: name
    and }\mp@subsup{Q}{}{\prime}:: p
```

    assumes Sim: \(\bigwedge R S .(R, S) \in \operatorname{Rel} \Longrightarrow R \rightsquigarrow<\) Rel \(>S\)
    and Eqvt: equt Rel
    and \(\quad P R e l Q:(P, Q) \in \operatorname{Rel}\)
    shows \(Q \Longrightarrow a<\nu x>\prec Q^{\prime} \Longrightarrow x \sharp P \Longrightarrow \exists P^{\prime} . P \Longrightarrow a<\nu x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in\)
    Rel
and $Q \Longrightarrow \alpha \prec Q^{\prime} \Longrightarrow \exists P^{\prime} . P \Longrightarrow \wedge \prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in$ Rel
proof -
assume $Q$ Trans: $Q \Longrightarrow a<\nu x>\prec Q^{\prime}$ and $x \sharp P$
from QTrans obtain $Q^{\prime \prime} Q^{\prime \prime \prime}$ where $Q$ Chain: $Q \Longrightarrow_{\tau} Q^{\prime \prime \prime}$
and $Q^{\prime \prime \prime}$ Trans: $Q^{\prime \prime \prime} \longmapsto a<\nu x>\prec Q^{\prime \prime}$
and $Q^{\prime \prime}$ Chain: $Q^{\prime \prime} \Longrightarrow{ }_{\tau} Q^{\prime}$
by (blast dest: transitionE)
from QChain PRelQ Sim obtain $P^{\prime \prime \prime}$ where PChain: $P \Longrightarrow{ }_{\tau} P^{\prime \prime \prime}$ and $P^{\prime \prime \prime} R e l Q^{\prime \prime \prime}$ :
$\left(P^{\prime \prime \prime}, Q^{\prime \prime \prime}\right) \in$ Rel
by(blast dest: weakSimTauChain)

```
from PChain }\langlex\sharpP>\mathrm{ have }x\sharp\mp@subsup{P}{}{\prime\prime\prime}\mathbf{by}(rule freshChain
```

from $P^{\prime \prime \prime}$ RelQ ${ }^{\prime \prime \prime}$ have $P^{\prime \prime \prime} \rightsquigarrow<$ Rel $>Q^{\prime \prime \prime}$ by (rule Sim)
with $Q^{\prime \prime \prime}$ Trans $\left\langle x \sharp P^{\prime \prime \prime}\right\rangle$ obtain $P^{\prime \prime}$ where $P^{\prime \prime \prime}$ Trans: $P^{\prime \prime \prime} \Longrightarrow a<\nu x>\prec P^{\prime \prime}$
and $P^{\prime \prime} \operatorname{Rel} Q^{\prime \prime}:\left(P^{\prime \prime}, Q^{\prime \prime}\right) \in \operatorname{Rel}$
by (blast dest: simE)
from $Q^{\prime \prime}$ Chain $P^{\prime \prime}$ Rel $Q^{\prime \prime} \operatorname{Sim}$ obtain $P^{\prime}$ where $P^{\prime \prime}$ Chain: $P^{\prime \prime} \Longrightarrow_{\tau} P^{\prime}$ and $P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}$
by(blast dest: weakSimTauChain)
from PChain $P^{\prime \prime \prime}$ Trans $P^{\prime \prime}$ Chain have $P \Longrightarrow a<\nu x>\prec P^{\prime}$
by (blast dest: Weak-Early-Step-Semantics.chainTransitionAppend)
with $P^{\prime}$ RelQ $Q^{\prime}$ show $\exists P^{\prime} . P \Longrightarrow a<\nu x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in$ Rel by blast
next
assume $Q \Longrightarrow \alpha \prec Q^{\prime}$
thus $\exists P^{\prime} . P \Longrightarrow \wedge \prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in$ Rel
proof (induct rule: transitionCases)
case Step
have $Q \Longrightarrow \alpha \prec Q^{\prime}$ by fact
then obtain $Q^{\prime \prime} Q^{\prime \prime \prime}$ where $Q$ Chain: $Q \Longrightarrow_{\tau} Q^{\prime \prime}$
and $Q^{\prime \prime}$ Trans: $Q^{\prime \prime} \longmapsto \alpha \prec Q^{\prime \prime \prime}$
and $Q^{\prime \prime \prime}$ Chain: $Q^{\prime \prime \prime} \Longrightarrow_{\tau} Q^{\prime}$
by (blast dest: transitionE)
from $Q$ Chain PRelQ Sim have $\exists P^{\prime \prime} . P \Longrightarrow{ }_{\tau} P^{\prime \prime} \wedge\left(P^{\prime \prime}, Q^{\prime \prime}\right) \in$ Rel
by(rule weakSimTauChain)
then obtain $P^{\prime \prime}$ where PChain: $P \Longrightarrow{ }_{\tau} P^{\prime \prime}$ and $P^{\prime \prime} \operatorname{Rel} Q^{\prime \prime}:\left(P^{\prime \prime}, Q^{\prime \prime}\right) \in$ Rel by blast
from $P^{\prime \prime}$ RelQ $Q^{\prime \prime}$ have $P^{\prime \prime} \rightsquigarrow<$ Rel $>Q^{\prime \prime}$ by (rule Sim)
with $Q^{\prime \prime}$ Trans obtain $P^{\prime \prime \prime}$ where $P^{\prime \prime}$ Trans: $P^{\prime \prime} \Longrightarrow \alpha \prec P^{\prime \prime \prime}$ and $P^{\prime \prime \prime}$ RelQ $Q^{\prime \prime \prime}:\left(P^{\prime \prime \prime}, Q^{\prime \prime \prime}\right) \in$ Rel
by (blast dest: simE)
have $\exists P^{\prime} . P^{\prime \prime \prime} \Longrightarrow_{\tau} P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in$ Rel using $Q^{\prime \prime \prime}$ Chain $P^{\prime \prime \prime}$ Rel $Q^{\prime \prime \prime}$ Sim by (rule weakSimTauChain)
then obtain $P^{\prime}$ where $P^{\prime \prime \prime}$ Chain: $P^{\prime \prime \prime} \Longrightarrow{ }_{\tau} P^{\prime}$ and $P^{\prime}$ RelQ $Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}$ by blast

```
    from PChain \(P^{\prime \prime}\) Trans \(P^{\prime \prime \prime}\) Chain have \(P \Longrightarrow \alpha \prec P^{\prime}\)
        by (blast dest: chainTransitionAppend)
    with \(P^{\prime}\) RelQ' show ?case by blast
next
    case Stay
    have \(P \Longrightarrow \imath \prec P\) by simp
    thus ?case using PRelQ by blast
    qed
qed
```

```
lemma eqvtI:
    fixes P :: pi
    and Q :: pi
    and Rel :: (pi\timespi) set
    and perm :: name prm
    assumes PSimQ: P }\rightsquigarrow<\mathrm{ Rel }>
    and RelRel':Rel \subseteqRel'
    and EqvtRel': eqvt Rel'
    shows (perm • P) \rightsquigarrow<Rel'> (perm •Q)
proof(induct rule: simCases)
    case(Bound Q' a x)
    have xFreshP: x\sharp perm •P by fact
    have QTrans: (perm •Q)\longmapstoa<\nux>}\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    hence (rev perm • (perm •Q))\longmapsto rev perm • (a<\nux> \prec Q') by(rule eqvts)
    hence }Q\longmapsto(\mathrm{ rev perm • a)< < (rev perm •x)> < (rev perm • Q')
    by(simp add: name-rev-per)
    moreover from xFreshP have (rev perm •x)\sharpP by(simp add: name-fresh-left)
    ultimately obtain }\mp@subsup{P}{}{\prime}\mathrm{ where PTrans: P C(rev perm • a)< < (rev perm • x)>
\prec P'
    and P'RelQ':( }\mp@subsup{P}{}{\prime}\mathrm{ , rev perm • Q ') G Rel using PSimQ
    by(blast dest: simE)
    from PTrans have (perm • P)\Longrightarrow(perm • rev perm • a)<\nu(perm • rev perm •
x)>}\prec\mathrm{ perm • P'
    by(rule eqvts)
    hence (perm •P)\Longrightarrowa<\nux> \prec(perm • P') by(simp add: name-per-rev)
    moreover from P'RelQ' RelRel' have ( }\mp@subsup{P}{}{\prime}\mathrm{ , rev perm • Q') }\mp@subsup{Q}{R}{\prime}\mp@subsup{R}{}{\prime}\mp@subsup{l}{}{\prime}\mathrm{ by blast
    with EqvtRel' have (perm • P', perm • (rev perm • Q')) \in Rel'
    by(rule eqvtRelI)
    hence (perm • P', Q') \in Rel' by(simp add: name-per-rev)
    ultimately show ?case by blast
next
    case(Free Q' \alpha)
    have QTrans: (perm •Q)\longmapsto\alpha\prec 林 by fact
    hence (rev perm • (perm •Q))\longmapsto rev perm • ( }\alpha\prec\mp@subsup{Q}{}{\prime})\mathrm{ by(rule eqvts)
    hence }Q\longmapsto(\mathrm{ rev perm • 人)}\prec(\mathrm{ rev perm • Q') by(simp add: name-rev-per)
    with PSimQ obtain P' where PTrans: P\Longrightarrow^ (rev perm \cdot \alpha)\prec \prec''
                        and PRel: ( }\mp@subsup{P}{}{\prime},(\mathrm{ rev perm • Q }\mp@subsup{)}{}{\prime}))\in\mathrm{ Rel
    by(blast dest: simE)
    from PTrans have (perm •P)\Longrightarrow^(perm \cdot rev perm \cdot 人) \prec perm • P'
    by(rule Weak-Early-Semantics.eqvtI)
hence L1:(perm •P)\Longrightarrow^ }\alpha\prec(\mathrm{ perm • P})\mathbf{by}(\mathrm{ simp add: name-per-rev)
from PRel EqvtRel' RelRel' have ((perm • P'), (perm • (rev perm • Q '))) \inRel'
    by(force intro: eqvtRelI)
```

```
    hence ((perm • P'), Q') \in Rel' by(simp add: name-per-rev)
    with L1 show ?case by blast
qed
```

lemma reflexive:
fixes $P$ :: $p i$
and Rel $::(p i \times p i)$ set
assumes $I d \subseteq$ Rel
shows $P \rightsquigarrow<$ Rel $>P$
using assms
by (auto intro: Weak-Early-Step-Semantics.singleActionChain
simp add: weakSimulation-def weakFreeTransition-def)
lemma transitive:
fixes $P \quad:: p i$
and $Q \quad:: p i$
and $R \quad:: p i$
and Rel $::(p i \times p i)$ set
and Rel' $::(p i \times p i)$ set
and $R e l^{\prime \prime}::(p i \times p i)$ set
assumes $Q \operatorname{SimR}: Q \rightsquigarrow<$ Rel $^{\prime}>R$
and Eqvt: eqvt Rel
and Eqvt": eqvt Rel"
and Trans: Rel $O R e l^{\prime} \subseteq R e l^{\prime \prime}$
and $\quad \operatorname{Sim}: \wedge S T .(S, T) \in \operatorname{Rel} \Longrightarrow S \rightsquigarrow<\operatorname{Rel}>T$
and PRelQ: $(P, Q) \in \operatorname{Rel}$
shows $P \rightsquigarrow<$ Rel $^{\prime \prime}>R$
proof -
from Eqvt" show ?thesis
proof $($ induct rule: simCasesCont $[$ where $C=Q])$
case(Bound a $x R^{\prime}$ )
have RTrans: $R \longmapsto a<\nu x>\prec R^{\prime}$ by fact
from $\langle x \sharp Q\rangle Q S i m R$ RTrans obtain $Q^{\prime}$ where $Q$ Trans: $Q \Longrightarrow a<\nu x>\prec Q^{\prime}$
and $Q^{\prime}$ Rel $^{\prime} R^{\prime}:\left(Q^{\prime}, R^{\prime}\right) \in \operatorname{Rel}^{\prime}$
by (blast dest: $\operatorname{sim} E)$
from Sim Eqvt PRelQ QTrans $\langle x \sharp P\rangle$
obtain $P^{\prime}$ where PTrans: $P \Longrightarrow a<\nu x>\prec P^{\prime}$ and $P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}$
by (drule-tac simE2) auto
moreover from $P^{\prime}$ RelQ $Q^{\prime} Q^{\prime} \operatorname{Rel}^{\prime} R^{\prime}$ Trans have $\left(P^{\prime}, R^{\prime}\right) \in$ Rel ${ }^{\prime \prime}$ by blast
ultimately show ?case by blast
next

```
    case(Free \alpha R')
    have RTrans: R\longmapsto\alpha\prec R' by fact
```



```
R') \in Rel'
    by(blast dest: simE)
    from Sim Eqvt PRelQ QTrans have }\exists\mp@subsup{P}{}{\prime}.P\Longrightarrow^ \alpha\prec P'^( ' (P', Q') \in Re
        by(blast intro: simEQ)
    then obtain }\mp@subsup{P}{}{\prime}\mathrm{ where PTrans: P C^ }\alpha\prec\mp@subsup{P}{}{\prime}\mathrm{ and P'RelQ':( (P', Q') & Rel
by blast
    from P'RelQ' Q'RelR' Trans have ( }\mp@subsup{P}{}{\prime},\mp@subsup{R}{}{\prime})\inR\mathrm{ Rel '' by blast
    with PTrans show ?case by blast
    qed
qed
lemma strongAppend:
    fixes P :: pi
    and }Q\quad::p
    and R :: pi
    and Rel :: (pi\timespi) set
    and Rel' :: (pi\timespi) set
    and Rel" :: (pi\times pi) set
    assumes PSimQ: P}\rightsquigarrow<\mathrm{ Rel }>
    and QSimR:Q}~[Rel`]
    and Eqvt'': eqvt Rel"
    and Trans: Rel O Rel'}\subseteqRel"
    shows P}\rightsquigarrow<\mp@subsup{\mathrm{ Rel }}{}{\prime\prime}>
proof -
    from Eqvt" show ?thesis
    proof(induct rule: simCasesCont[where C=Q])
        case(Bound a x R')
        have RTrans: R\longmapstoa<\nux>}\prec\mp@subsup{R}{}{\prime}\mathrm{ by fact
        from QSimR RTrans <x #Q> obtain Q' where QTrans: Q}\longmapstoa<\nux>\prec\prec Q
                                and \mp@subsup{Q}{}{\prime}\mp@subsup{Rel}{}{\prime}\mp@subsup{R}{}{\prime}:(\mp@subsup{Q}{}{\prime},R}\mp@subsup{R}{}{\prime})\in\mp@subsup{Rel}{}{\prime
            by(blast dest: Strong-Early-Sim.elim)
        with PSimQ QTrans <x #P> obtain P' where PTrans: P\Longrightarrowa<\nux>}\prec\mp@subsup{P}{}{\prime
and P'RelQ':}(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\operatorname{Rel
            by(blast dest: simE)
        moreover from P'RelQ' Q'Rel'R' Trans have ( }\mp@subsup{P}{}{\prime},\mp@subsup{R}{}{\prime})\inRe\mp@subsup{l}{}{\prime\prime}\mathrm{ by blast
        ultimately show ?case by blast
    next
    case(Free \alpha R ')
    have RTrans: }R\longmapsto\alpha\prec\mp@subsup{R}{}{\prime}\mathrm{ by fact
    with QSimR obtain }\mp@subsup{Q}{}{\prime}\mathrm{ where QTrans: Q}\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime}\mathrm{ and }\mp@subsup{Q}{}{\prime}\mp@subsup{R}{}{R}\mp@subsup{l}{R}{\prime}:(\mp@subsup{Q}{}{\prime},\mp@subsup{R}{}{\prime}
Rel'
            by(blast dest: Strong-Early-Sim.elim)
    from PSimQ QTrans obtain P' where PTrans: P\Longrightarrow^ \alpha\prec P' and P'RelQ':
```

```
(P', Q})\in\operatorname{Rel
        by(blast dest: simE)
    from P'RelQ' Q'RelR' Trans have ( }\mp@subsup{P}{}{\prime},\mp@subsup{R}{}{\prime})\inRel\mp@subsup{l}{}{\prime\prime}\mathrm{ by blast
    with PTrans show ?case by blast
    qed
qed
lemma strongSim WeakSim:
    fixes P :: pi
    and }Q ::p
    and Rel ::(pi\timespi) set
    assumes PSimQ: P\rightsquigarrow[Rel] Q
    shows P}\rightsquigarrow<\mathrm{ Rel > Q
proof(induct rule: simCases)
    case(Bound Q' a x)
    have }Q\longmapstoa<\nux><<\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    with PSimQ <x\sharpP> obtain P' where PTrans: P\longmapstoa<\nux> \prec P' and P'RelQ':
( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\operatorname{Rel
    by(blast dest:Strong-Early-Sim.elim)
    from PTrans have P\Longrightarrowa<\nux> \prec P'
            by(force intro:Weak-Early-Step-Semantics.singleActionChain simp add: weak-
FreeTransition-def)
    with P'RelQ' show ?case by blast
next
    case(Free Q' \alpha)
    have }Q\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    with PSimQ obtain P' where PTrans: P\longmapsto\alpha \prec- ' ' and P'RelQ':( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})
Rel
    by(blast dest: Strong-Early-Sim.elim)
    from PTrans have P\Longrightarrow^ \alpha}\prec\mp@subsup{P}{}{\prime}\mathbf{by}(rule Weak-Early-Semantics.singleActionChain)
    with P'RelQ' show ?case by blast
qed
end
theory Weak-Early-Bisim
    imports Weak-Early-Sim Strong-Early-Bisim
begin
lemma monoAux: }A\subseteqB\LongrightarrowP\rightsquigarrow<A>Q\longrightarrowP\rightsquigarrow<B>
by(auto intro:Weak-Early-Sim.monotonic)
coinductive-set weakBisim :: (pi\times pi) set
where
    step: \llbracketP\rightsquigarrow<\mathrm{ weakBisim> Q; (Q,P) ש weakBisim}\rrbracket\Longrightarrow(P,Q)\inweakBisim
monos monoAux
```

abbreviation weakEarlyBisimJudge (infixr $\approx 65$ ) where $P \approx Q \equiv(P, Q) \in$ weakBisim
lemma weakBisimCoinductAux[case-names weakBisim, case-conclusion weakBisim step, consumes 1]:
assumes $p:(P, Q) \in X$
and step: $\wedge P Q .(P, Q) \in X \Longrightarrow P \rightsquigarrow<(X \cup$ weakBisim $)>Q \wedge(Q, P) \in X$ $\cup$ weakBisim
shows $P \approx Q$
proof -
have aux: $X \cup$ weakBisim $=\{(P, Q) .(P, Q) \in X \vee P \approx Q\}$ by blast
from $p$ show ?thesis
by (coinduct, force dest: step simp add: aux)
qed
lemma weakBisimWeakCoinductAux[case-names weakBisim, case-conclusion weakBisim step, consumes 1]:
assumes $p:(P, Q) \in X$
and step: $\wedge P Q \cdot(P, Q) \in X \Longrightarrow P \rightsquigarrow<X>Q \wedge(Q, P) \in X$
shows $P \approx Q$
using $p$
proof (coinduct rule: weakBisimCoinductAux)
case (weakBisim P)
from step $[$ OF this] show ?case using Weak-Early-Sim.monotonic by blast
qed
lemma weakBisimCoinduct[consumes 1, case-names cSim cSym]:
fixes $P$ :: $p i$
and $\quad Q:: p i$
and $\quad X::(p i \times p i)$ set
assumes $(P, Q) \in X$
and $\quad \bigwedge R S .(R, S) \in X \Longrightarrow R \rightsquigarrow<(X \cup$ weakBisim $)>S$
and $\quad \bigwedge R S \cdot(R, S) \in X \Longrightarrow(S, R) \in X$
shows $P \approx Q$
using assms
by (coinduct rule: weakBisimCoinductAux) auto
lemma weakBisim WeakCoinduct[consumes 1, case-names cSim cSym]:
fixes $P$ :: $p i$
and $\quad Q:: p i$
and $\quad X::(p i \times p i)$ set
assumes $(P, Q) \in X$
and $\quad \backslash P Q \cdot(P, Q) \in X \Longrightarrow P \rightsquigarrow<X>Q$

```
    and \(\quad \wedge P Q \cdot(P, Q) \in X \Longrightarrow(Q, P) \in X\)
    shows \(P \approx Q\)
using assms
by (coinduct rule: weakBisim WeakCoinductAux) auto
lemma weakBisimE:
    fixes \(P\) :: \(p i\)
    and \(\quad Q:: p i\)
    assumes \(P \approx Q\)
    shows \(P \rightsquigarrow<\) weakBisim> \(Q\)
    and \(Q \approx P\)
using assms
by(auto dest: weakBisim.cases)
lemma weakBisimI:
    fixes \(P:: p i\)
    and \(\quad Q:: p i\)
    assumes \(P \rightsquigarrow<\) weakBisim \(>Q\)
    and \(\quad Q \approx P\)
    shows \(P \approx Q\)
using assms
by(auto intro: weakBisim.intros)
lemma eqvt[simp]:
    shows eqvt weakBisim
proof (auto simp add: eqvt-def)
    let ? \(X=\{x . \exists P Q(\) perm::name prm \() . P \approx Q \wedge x=(\) perm \(\cdot P\), perm \(\cdot Q)\}\)
    fix \(P Q\)
    fix perm::name prm
    assume \(P \operatorname{BiSim} Q: P \approx Q\)
    hence \((\) perm \(\cdot P\), perm \(\cdot Q) \in ? X\) by blast
    moreover have \(\bigwedge P Q\) perm::name prm. \(\llbracket P \rightsquigarrow<\) weakBisim \(>Q \rrbracket \Longrightarrow(\) perm \(\cdot P)\)
\(\rightsquigarrow<\) ? \(X>(\) perm \(\cdot Q)\)
    proof -
        fix \(P Q\)
        fix perm::name prm
        assume \(P \rightsquigarrow<\) weakBisim \(>Q\)
        moreover have weakBisim \(\subseteq\) ? \(X\)
        proof (auto)
            fix \(P Q\)
            assume \(P \approx Q\)
            moreover have \(P=([]::\) name prm \() \cdot P\) and \(Q=([]::\) name prm \() \cdot Q\) by
```

auto
ultimately show $\exists P^{\prime} Q^{\prime} . P^{\prime} \approx Q^{\prime} \wedge\left(\exists(\right.$ perm::name prm $) . P=$ perm $\cdot P^{\prime}$ $\left.\wedge Q=\operatorname{perm} \cdot Q^{\prime}\right)$
by blast
qed
moreover have equt? $X$
proof (auto simp add: eqvt-def)
fix $P Q$
fix perm1:: name prm
fix perm2::name prm
assume $P \approx Q$
moreover have perm1 • perm2 $\cdot P=($ perm1 @ perm2 $) \cdot P$ by $(\operatorname{simp}$ add: pt2[OF pt-name-inst])
moreover have perm1 • perm2 $\cdot Q=($ perm1 @ perm2 $) \cdot Q \mathbf{b y}(\operatorname{simp}$ add: pt2[OF pt-name-inst])
ultimately show $\exists P^{\prime} Q^{\prime} . P^{\prime} \approx Q^{\prime} \wedge(\exists($ perm::name prm $)$. perm1 $\cdot$ perm2 - $P=\operatorname{perm} \cdot P^{\prime} \wedge$
perm1 $\left.\cdot \operatorname{perm2} \cdot Q=\operatorname{perm} \cdot Q^{\prime}\right)$
by blast
qed
ultimately show $($ perm $\cdot P) \rightsquigarrow<? X>($ perm $\cdot Q)$
by(rule Weak-Early-Sim.eqvtI)
qed
ultimately show $($ perm $\cdot P) \approx($ perm $\cdot Q) \mathbf{b y}($ coinduct rule: weakBisimWeakCoinductAux, blast dest: weakBisimE)
qed
lemma eqvtI[eqvt]:
fixes $P:: p i$
and $\quad Q:: p i$
and perm :: name prm
assumes $P \approx Q$

```
shows (perm • P) \approx (perm •Q)
using assms
by(rule eqvtRelI[OF eqvt])
lemma strongBisimWeakBisim:
fixes }P::p
and }Q::p
assumes P~Q
```

```
    shows P\approxQ
proof -
    from <P~ Q\rangle show ?thesis
    proof(coinduct rule: weakBisimWeakCoinduct)
        case(cSim P Q)
        from }\langleP~Q\rangle\mathrm{ have P}>>[bisim] Q by(rule bisimE
        thus P\rightsquigarrow<bisim> Q by(rule strongSimWeakSim)
    next
        case(cSym P Q)
        thus ?case by(rule bisimE)
    qed
qed
lemma reflexive:
    fixes P :: pi
    shows P\approxP
proof -
    have }(P,P)\inId by sim
    thus ?thesis
        by(coinduct rule: weakBisimCoinduct) (auto intro: Weak-Early-Sim.reflexive)
qed
lemma symetric:
    fixes P :: pi
    and }Q:: p
    assumes P\approxQ
    shows Q }~
using assms
by(auto dest: weakBisimE)
lemma transitive:
    fixes P :: pi
    and }Q::p
    and }R:: p
    assumes P\approxQ
    and }Q\approx
    shows P}\approx
proof -
    let ?X = weakBisim O weakBisim
    from assms have ( }P,R)\in?X\mathrm{ by blast
    thus ?thesis
    proof(coinduct rule: weakBisimCoinduct)
        case(cSim P R)
        from}\langle(P,R)\in?X\rangle\mathrm{ obtain Q where P}\approxQ\mathrm{ and }Q\approxR\mathrm{ by auto
```

```
    from〈Q\approx R〉 have Q}<<<weakBisim> R by(rule weakBisimE
    moreover have eqvt ? X by auto
    moreover have ? X \subseteq? \ by simp
    ultimately show P}\rightsquigarrow<(?X\cup\mathrm{ weakBisim )}>R\mathrm{ using weakBisimE(1)<P}
Q>
    by(rule-tac Weak-Early-Sim.transitive) auto
    next
    case(cSym P R)
    thus ?case by(auto dest: symetric)
    qed
qed
lemma weakBisim WeakUpto[case-names cSim cSym, consumes 1]:
    assumes p:(P,Q)\inX
    and Eqvt: eqvt X
    and rSim: }\PQ.(P,Q)\inX\LongrightarrowP\rightsquigarrow<(\mathrm{ weakBisim OX O bisim ) > Q
    and rSym: \bigwedgePQ.(P,Q)\inX\Longrightarrow(Q,P)\inX
    shows P}\approx
proof -
    let ?X = weakBisim O X O weakBisim
    let ?Y = weakBisim O X O bisim
    from Eqvt eqvt have eqvt ?X by blast
    from Strong-Early-Bisim.eqvt Eqvt eqvt have eqvt ?Y by blast
    from}\langle(P,Q)\inX> have (P,Q)\in?X by(blast intro: Strong-Early-Bisim.reflexive
reflexive)
    thus ?thesis
    proof(coinduct rule: weakBisimWeakCoinduct)
    case(cSim P Q)
    {
        fix P P ' }\mp@subsup{Q}{}{\prime}
        assume P}\approx\mp@subsup{P}{}{\prime}\mathrm{ and ( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\inX\mathrm{ and }\mp@subsup{Q}{}{\prime}\approx
        from }\langle\mp@subsup{Q}{}{\prime}\approxQ\rangle\mathrm{ have }\mp@subsup{Q}{}{\prime}\rightsquigarrow<\mathrm{ weakBisim }>Q by(rule weakBisimE
        moreover note <eqvt ? Y> <eqvt ? X>
        moreover have ?Y O weakBisim \subseteq?X by(blast dest: strongBisim WeakBisim
transitive)
moreover {
            fix PQ
            assume (P,Q)\in?Y
            then obtain P' 蛉 where P}\approx\mp@subsup{P}{}{\prime}\mathrm{ and ( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\inX\mathrm{ and }\mp@subsup{Q}{}{\prime}~Q\mathrm{ by auto
            from <( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\inX\rangle\mathrm{ have }\mp@subsup{P}{}{\prime}\rightsquigarrow<?Y>>\mp@subsup{Q}{}{\prime}\mathbf{by}(\mathrm{ rule rSim)
            moreover from < Q ' ~ Q> have }\mp@subsup{Q}{}{\prime}\rightsquigarrow[bisim] Q by(rule bisimE
            moreover note <eqvt?Y>
            moreover have ?Y O bisim \subseteq?Y by(auto dest: Strong-Early-Bisim.transitive)
            ultimately have }\mp@subsup{P}{}{\prime}\rightsquigarrow<\mathrm{ ?Y>}Q by(rule strongAppend)
            moreover note <P}\approx\mp@subsup{P}{}{\prime}
            moreover have weakBisim O?Y\subseteq?Y by(blast dest: transitive)
            ultimately have P}\rightsquigarrow<\mathrm{ ? Y> Q using weakBisimE(1) eqvt <eqvt?Y>
```

```
            by(rule-tac Weak-Early-Sim.transitive)
    }
    moreover from «( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\inX> have ( (P', Q')\in?Y by(blast intro: reflexive
Strong-Early-Bisim.reflexive)
    ultimately have }\mp@subsup{P}{}{\prime}\rightsquigarrow<?X>Q by(rule Weak-Early-Sim.transitive)
    moreover note < P \approx P'>
    moreover have weakBisim O?X \subseteq?X by(blast dest: transitive)
    ultimately have P}\rightsquigarrow<?\\>Q using weakBisimE(1) eqvt <eqvt?X>
        by(rule-tac Weak-Early-Sim.transitive)
    }
    with «(P,Q)\in?X> show ?case by auto
    next
    case(cSym P Q)
    thus ?case
        apply auto
        by(blast dest: bisimE rSym weakBisimE)
    qed
qed
lemma weakBisimUpto[case-names cSim cSym, consumes 1]:
    assumes p:(P,Q)\inX
    and Eqvt: eqvt X
    and rSim: \RS.(R,S)\inX\LongrightarrowR\rightsquigarrow<(weakBisim O(X\cup weakBisim) O
bisim)>S
    and rSym: \RS. (R,S)\inX\Longrightarrow(S,R)\inX
    shows P\approxQ
proof -
    from p have }(P,Q)\inX\cup\mathrm{ weakBisim by simp
    thus ?thesis using Eqvt
        apply(coinduct rule: weakBisimWeakUpto)
        apply(auto dest: rSim rSym weakBisimE)
        apply(rule Weak-Early-Sim.monotonic)
        apply(blast dest: weakBisimE)
        apply(auto simp add: relcomp-unfold)
        by(metis reflexive Strong-Early-Bisim.reflexive transitive)
qed
lemma transitive-coinduct-weak[case-names cSim cSym, consumes 2]:
    assumes p:(P,Q)\inX
    and Eqvt: eqvt X
    and rSim: }\PQ.(P,Q)\inX\LongrightarrowP\rightsquigarrow<(\mathrm{ bisim O X O bisim ) > Q
    and rSym: \bigwedgePQ.(P,Q)\inX\Longrightarrow(Q,P)\in\operatorname{bisim O X O bisim}
    shows P}\approx
proof -
    let ?X = bisim O X O bisim
    from}\langle(P,Q)\inX> have (P,Q)\in?X by(blast intro: Strong-Early-Bisim.reflexive
```

```
thus ?thesis
proof(coinduct rule: weakBisimWeakCoinduct)
    case(cSim P Q)
    {
        fix P P ' }\mp@subsup{Q}{}{\prime}
        assume PBisimP': P~ P'
        assume P'Sim}\mp@subsup{Q}{}{\prime}:\mp@subsup{P}{}{\prime}\rightsquigarrow<?X> Q Q 
        assume Q'SimQ: Q '\rightsquigarrow[bisim] Q
        have P\rightsquigarrow<?X>Q
        proof -
            have }\mp@subsup{P}{}{\prime}\rightsquigarrow<\mathrm{ ? X>>Q
            proof -
            have ?X O bisim \subseteq?X by(blast intro: Strong-Early-Bisim.transitive)
            moreover from Strong-Early-Bisim.eqvt Eqvt have eqvt ?X by blast
            ultimately show ?thesis using P'SimQ' Q'SimQ
                by(rule-tac strongAppend)
            qed
            moreover have eqvt bisim by(rule Strong-Early-Bisim.eqvt)
            moreover from Strong-Early-Bisim.eqvt Eqvt have eqvt ?X by blast
    moreover have bisim O?X\subseteq?X by(blast intro: Strong-Early-Bisim.transitive)
                moreover have }\bigwedgePQ.P~Q\LongrightarrowP\rightsquigarrow<\mathrm{ bisim> Q by(blast dest:
Strong-Early-Bisim.bisimE strongSimWeakSim)
        ultimately show?thesis using PBisimP' by(rule Weak-Early-Sim.transitive)
        qed
    }
    thus ?case using «(P,Q)\in?X> rSim by (blast dest: Strong-Early-Bisim.bisimE)
    next
        case(cSym P Q)
    {
        fix P P P' Q'Q
        assume P~\mp@subsup{P}{}{\prime}\mathrm{ and ( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\inX\mathrm{ and }\mp@subsup{Q}{}{\prime}~Q
        from }\langle(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\inX>\mathrm{ have ( }\mp@subsup{Q}{}{\prime},\mp@subsup{P}{}{\prime})\in?X by(rule rSym
        with }\langleP~\mp@subsup{P}{}{\prime}\rangle\langle\mp@subsup{Q}{}{\prime}~Q\rangle\mathrm{ have }(Q,P)\in?
            apply auto
            apply(drule-tac Strong-Early-Bisim.bisimE(2))
            apply(drule Strong-Early-Bisim.transitive[where Q=P\])
            apply assumption
            apply(drule-tac Strong-Early-Bisim.bisimE(2))
            apply(drule Strong-Early-Bisim.transitive[where Q=Q ])
            apply assumption
            by auto
    }
    thus ?case using <(P,Q) \in?X> by auto
    qed
qed
end
```

```
theory Weak-Early-Step-Sim
    imports Weak-Early-Sim Strong-Early-Sim
begin
definition weakStepSimulation \(:: ~ p i \Rightarrow(p i \times p i)\) set \(\Rightarrow\) pi \(\Rightarrow\) bool \((-\rightsquigarrow 《-»-[80\),
80, 80] 80) where
    \(P \rightsquigarrow «\) Rel» \(Q \equiv\left(\forall Q^{\prime}\right.\) a \(x . Q \longmapsto a<\nu x>\prec Q^{\prime} \longrightarrow x \sharp P \longrightarrow\left(\exists P^{\prime} . P \Longrightarrow a<\nu x>\right.\)
\(\left.\left.\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}\right)\right) \wedge\)
    \(\left(\forall Q^{\prime} \alpha . Q \longmapsto \alpha \prec Q^{\prime} \longrightarrow\left(\exists P^{\prime} . P \Longrightarrow \alpha \prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in\right.\right.\)
Rel))
lemma monotonic:
    fixes \(A::(p i \times p i)\) set
    and \(B::(p i \times p i)\) set
    and \(P:: p i\)
    and \(\quad P^{\prime}:: p i\)
    assumes \(P \rightsquigarrow « A\) » \(P^{\prime}\)
    and \(A \subseteq B\)
    shows \(P \rightsquigarrow « B » P^{\prime}\)
using assms
by(simp add: weakStepSimulation-def) blast
lemma simCasesCont[consumes 1, case-names Bound Free]:
    fixes \(P\) :: \(p i\)
    and \(\quad Q \quad:: p i\)
    and Rel \(::(p i \times p i)\) set
    and \(C\) :: 'a::fs-name
    assumes Eqvt: eqvt Rel
    and Bound: \(\wedge a x Q^{\prime} . \llbracket x \sharp C ; Q \longmapsto a<\nu x>\prec Q^{\wedge} \Longrightarrow \exists P^{\prime} . P \Longrightarrow a<\nu x>\)
\(\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}\)
    and Free: \(\wedge \alpha Q^{\prime} . Q \longmapsto \alpha \prec Q^{\prime} \Longrightarrow \exists P^{\prime} . P \Longrightarrow \alpha \prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in\) Rel
    shows \(P \rightsquigarrow\) Rel» \(Q\)
proof -
    from Free show ?thesis
    proof (auto simp add: weakStepSimulation-def)
            fix \(Q^{\prime} a x\)
            assume \(x\) Fresh \(P:(x::\) name \() \sharp P\)
            assume Trans: \(Q \longmapsto a<\nu x>\prec Q^{\prime}\)
            have \(\exists c:\) :name. \(c \sharp\left(P, Q^{\prime}, x, C\right)\) by (blast intro: name-exists-fresh)
    then obtain \(c:: n a m e\) where \(c\) FreshP: \(c \sharp P\) and \(c F r e s h Q^{\prime}: c \sharp Q^{\prime}\) and \(c F r e s h C\) :
\(c \sharp C\)
                                and cineqx: \(c \neq x\)
            by(force simp add: fresh-prod)
```

            from Trans cFresh \(Q^{\prime}\) have \(Q \longmapsto a<\nu c>\prec\left([(x, c)] \cdot Q^{\prime}\right)\) by (simp add:
    ```
alphaBoundOutput)
    with cFreshC have }\exists\mp@subsup{P}{}{\prime}.P\Longrightarrowa<\nuc>\prec \prec P'^( (P',[(x,c)]\cdotQ')\inRe
        by(rule Bound)
    then obtain }\mp@subsup{P}{}{\prime}\mathrm{ where PTrans: P >a<vc> < P' and P'RelQ':( P', [(x,
c)] \cdot Q') }\in\operatorname{Rel
        by blast
    from PTrans \langlex\sharpP\rangle\langlec\not= x\rangle have P\Longrightarrowa<\nux> \prec ([(x,c)] \cdot P')
        by(simp add: weakTransitionAlpha name-swap)
    moreover from Eqvt P'RelQ' have ([(x,c)] • P',[(x,c)] \cdot [(x,c)] • Q') \in Rel
        by(rule eqvtRelI)
    with }\langlec\not=x\rangle\mathrm{ have ([(x,c)] • P', Q') G Rel
        by simp
    ultimately show }\exists\mp@subsup{P}{}{\prime}.P\Longrightarrowa<\nux><<\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mathrm{ Rel by blast
    qed
qed
lemma simCases[consumes 0, case-names Bound Free]:
    fixes P :: pi
    and }Q ::p
    and Rel :: (pi\timespi) set
    and C :: 'a::fs-name
    assumes \ax \mp@subsup{Q}{}{\prime}.\llbracketQ\longmapstoa<\nux>}\prec\mp@subsup{Q}{}{\prime};x\sharpP\rrbracket\Longrightarrow\exists\mp@subsup{P}{}{\prime}.P\Longrightarrowa<\nux>\prec\prec P
\wedge( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\inRe
    and }\quad\Lambda\alpha\mp@subsup{Q}{}{\prime}.Q\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime}\Longrightarrow\exists\mp@subsup{P}{}{\prime}.P\Longrightarrow\alpha\prec\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mathrm{ Rel
    shows P}\rightsquigarrow«<R\mathrm{ Rel» Q
using assms
by(auto simp add:weakStepSimulation-def)
lemma simE:
    fixes P :: pi
    and Rel ::(pi\timespi) set
    and }Q :: p
    and a :: name
    and }x\mathrm{ :: name
    and }\mp@subsup{Q}{}{\prime}:: p
    assumes P\rightsquigarrow«Rel» Q
    shows }Q\longmapstoa<\nux>\prec\mp@subsup{Q}{}{\prime}\Longrightarrowx\sharpP\Longrightarrow\exists\mp@subsup{P}{}{\prime}.P\Longrightarrowa<\nux>\prec\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})
Rel
    and }Q\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime}\Longrightarrow\exists\mp@subsup{P}{}{\prime}.P\Longrightarrow\alpha\prec\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mathrm{ Rel
using assms by(simp add: weakStepSimulation-def)+
lemma simE2:
    fixes P :: pi
    and Rel ::(pi\timespi) set
```

```
    and }Q :: p
    and a :: name
    and }x\mathrm{ :: name
    and }\mp@subsup{Q}{}{\prime}::p
    assumes PSimQ: P\rightsquigarrow«Rel»}
    and Sim: \bigwedgeRS.(R,S)\inRel\Longrightarrow \Longrightarrow}\rightsquigarrow<\mathrm{ Rel> S
    and Eqvt: eqvt Rel
    and PRelQ:}(P,Q)\in\mathrm{ Rel
    shows }Q\Longrightarrowa<\nux>\prec\mp@subsup{Q}{}{\prime}\Longrightarrowx\sharpP\Longrightarrow\exists\mp@subsup{P}{}{\prime}.P\Longrightarrowa<\nux>\prec\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})
Rel
    and }Q\Longrightarrow\alpha\prec\mp@subsup{Q}{}{\prime}\Longrightarrow\exists\mp@subsup{P}{}{\prime}.P\Longrightarrow\alpha\prec\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mathrm{ Rel
proof -
    assume QTrans: Q\Longrightarrowa<\nux>\prec\prec Q'
    assume }x\sharp
    from QTrans obtain }\mp@subsup{Q}{}{\prime\prime}\mp@subsup{Q}{}{\prime\prime\prime}\mathrm{ where QChain: Q >}\mp@subsup{~}{\tau}{}\mp@subsup{Q}{}{\prime\prime
                                    and }\mp@subsup{Q}{}{\prime\prime}\mathrm{ Trans: }\mp@subsup{Q}{}{\prime\prime}\longmapstoa<\nux> \prec\mp@subsup{Q}{}{\prime\prime\prime
                                    and Q '"'Chain: }\mp@subsup{Q}{}{\prime\prime\prime}\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{Q}{}{\prime
    by(blast dest: transitionE)
    from QChain PRelQ Sim have }\exists\mp@subsup{P}{}{\prime\prime}.P\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime\prime}\wedge(\mp@subsup{P}{}{\prime\prime},\mp@subsup{Q}{}{\prime\prime})\in\mathrm{ Rel
        by(rule weakSimTauChain)
    then obtain P'\prime where PChain: P\Longrightarrow}\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime\prime}\mathrm{ and }\mp@subsup{P}{}{\prime\prime}RelQ\mp@subsup{Q}{}{\prime\prime}:(\mp@subsup{P}{}{\prime\prime},\mp@subsup{Q}{}{\prime\prime})\in\operatorname{Rel}\mathrm{ by
blast
```



```
    from P}\mp@subsup{P}{}{\prime\prime}RelQ\mp@subsup{Q}{}{\prime\prime}\mathrm{ have }\mp@subsup{P}{}{\prime\prime}\rightsquigarrow<\mathrm{ Rel }>\mp@subsup{Q}{}{\prime\prime}\mathbf{by}(rule Sim
    with }\mp@subsup{Q}{}{\prime\prime}\mathrm{ Trans xFreshP'/ obtain }\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ where P}\mp@subsup{P}{}{\prime\prime}\mathrm{ Trans: }\mp@subsup{P}{}{\prime\prime}\Longrightarrowa<\nux>\prec \prec P'\prime\prime
                        and }\mp@subsup{P}{}{\prime\prime\prime}RelQ\mp@subsup{Q}{}{\prime\prime\prime}:(\mp@subsup{P}{}{\prime\prime\prime},\mp@subsup{Q}{}{\prime\prime\prime})\inRe
    by(blast dest:Weak-Early-Sim.simE)
    have }\exists\mp@subsup{P}{}{\prime}.\mp@subsup{P}{}{\prime\prime\prime}\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mathrm{ Rel using Q '/'Chain P
    by(rule weakSimTauChain)
    then obtain P' where }\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ Chain: }\mp@subsup{P}{}{\prime\prime\prime}\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{P}{}{\prime}\mathrm{ and }\mp@subsup{P}{}{\prime}RelQ':(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\operatorname{Rel}\mathrm{ by
blast
```



```
        by(blast dest: Weak-Early-Step-Semantics.chainTransitionAppend)
    with P'RelQ' show }\exists\mp@subsup{P}{}{\prime}.P\Longrightarrowa<\nux>\prec\prec P'^( ' ', Q')\in Re
    by blast
next
    assume }Q\Longrightarrow\alpha\prec\mp@subsup{Q}{}{\prime
    then obtain }\mp@subsup{Q}{}{\prime\prime}\mp@subsup{Q}{}{\prime\prime\prime}\mathrm{ where QChain: }Q\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{Q}{}{\prime\prime
                        and Q '"Trans: }\mp@subsup{Q}{}{\prime\prime}\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime\prime\prime
                        and Q '"'Chain: }\mp@subsup{Q}{}{\prime\prime\prime}\Longrightarrow\mp@subsup{\Longrightarrow}{\tau}{}\mp@subsup{Q}{}{\prime
    by(blast dest: transitionE)
```

from QChain $Q^{\prime \prime}$ Trans $Q^{\prime \prime \prime}$ Chain show $\exists P^{\prime} . P \Longrightarrow \alpha \prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in$ Rel proof(induct arbitrary: $\alpha Q^{\prime \prime \prime} Q^{\prime}$ rule: tauChainInduct)
case id
from $P \operatorname{Sim} Q\left\langle Q \longmapsto \alpha \prec Q^{\prime \prime \prime}\right\rangle$ have $\exists P^{\prime} . P \Longrightarrow \alpha \prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime \prime \prime}\right) \in$ Rel by (blast dest: simE)
then obtain $P^{\prime \prime \prime}$ where PTrans: $P \Longrightarrow \alpha \prec P^{\prime \prime \prime}$ and $P^{\prime} \operatorname{Rel} Q^{\prime \prime \prime}:\left(P^{\prime \prime \prime}, Q^{\prime \prime \prime}\right) \in$ Rel
by blast
have $\exists P^{\prime} . P^{\prime \prime \prime} \Longrightarrow{ }_{\tau} P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in$ Rel using $\left\langle Q^{\prime \prime \prime} \Longrightarrow_{\tau} Q^{\prime}\right\rangle P^{\prime} \operatorname{Rel} Q^{\prime \prime \prime}$ Sim by (rule Weak-Early-Sim.weakSimTauChain)
then obtain $P^{\prime}$ where $P^{\prime \prime \prime}$ Chain: $P^{\prime \prime \prime} \Longrightarrow{ }_{\tau} P^{\prime}$ and $P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}$ by blast
from $P^{\prime \prime \prime}$ Chain PTrans have $P \Longrightarrow \alpha \prec P^{\prime}$
by (blast dest: Weak-Early-Step-Semantics.chainTransitionAppend)
with $P^{\prime}$ RelQ' show ?case by blast

## next

case (ih $\left.Q^{\prime \prime \prime \prime} Q^{\prime \prime} \alpha Q^{\prime \prime \prime} Q^{\prime}\right)$
have $Q^{\prime \prime} \Longrightarrow_{\tau} Q^{\prime \prime}$ by simp
with $\left\langle Q^{\prime \prime \prime \prime} \longmapsto \tau \prec Q^{\prime \prime}\right\rangle$ obtain $P^{\prime \prime}$ where PChain: $P \Longrightarrow \tau \prec P^{\prime \prime}$ and $P^{\prime \prime}$ RelQ ${ }^{\prime \prime}:\left(P^{\prime \prime}, Q^{\prime \prime}\right) \in$ Rel
by (drule-tac ih) auto
from $P^{\prime \prime} \operatorname{Rel} Q^{\prime \prime}$ have $P^{\prime \prime} \rightsquigarrow<$ Rel $>Q^{\prime \prime}$ by (rule Sim)
hence $\exists P^{\prime \prime \prime} . P^{\prime \prime} \Longrightarrow \wedge \prec P^{\prime \prime \prime} \wedge\left(P^{\prime \prime \prime}, Q^{\prime \prime \prime}\right) \in$ Rel using $\left\langle Q^{\prime \prime} \longmapsto \alpha \prec Q^{\prime \prime \prime}\right\rangle$ by (rule Weak-Early-Sim.simE)
then obtain $P^{\prime \prime \prime}$ where $P^{\prime \prime}$ Trans: $P^{\prime \prime} \Longrightarrow \alpha \prec P^{\prime \prime \prime}$ and $P^{\prime \prime \prime} \operatorname{Rel} Q^{\prime \prime \prime}:\left(P^{\prime \prime \prime}, Q^{\prime \prime \prime}\right) \in \operatorname{Rel}$
by blast
from $\left\langle Q^{\prime \prime \prime} \Longrightarrow_{\tau} Q^{\prime}\right\rangle P^{\prime \prime \prime}$ RelQ $Q^{\prime \prime \prime}$ Sim have $\exists P^{\prime} . P^{\prime \prime \prime} \Longrightarrow_{\tau} P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in$ Rel by (rule Weak-Early-Sim.weakSimTauChain)
then obtain $P^{\prime}$ where $P^{\prime \prime \prime}$ Chain: $P^{\prime \prime \prime} \Longrightarrow{ }_{\tau} P^{\prime}$
and $P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}$
by blast
from PChain $P^{\prime \prime}$ Trans have $P \Longrightarrow \alpha \prec P^{\prime \prime \prime}$
apply (auto simp add: freeTransition-def weakFreeTransition-def)
apply (drule tauActTauChain, auto)
by (rule-tac $x=P^{\prime \prime \prime} a a$ in exI) auto
hence $P \Longrightarrow \alpha \prec P^{\prime}$ using $P^{\prime \prime \prime}$ Chain
by(rule Weak-Early-Step-Semantics.chainTransitionAppend)
with $P^{\prime}$ RelQ' show ?case by blast
qed
qed
lemma eqvtI:
fixes $P \quad:: p i$
and $\quad Q \quad:: p i$

```
    and Rel :: (pi\times pi) set
    and perm :: name prm
    assumes PSimQ: P\rightsquigarrow«Rel» Q
    and RelRel':Rel \subseteqRel'
    and EqvtRel': eqvt Rel'
    shows (perm • P) \rightsquigarrow«Rel'» (perm •Q)
proof(induct rule: simCases)
    case(Bound a x Q')
    have xFreshP: x\sharp perm • P by fact
    have QTrans: (perm •Q)\longmapstoa<\nux>}\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    hence (rev perm • (perm •Q))\longmapsto rev perm • (a<\nux> \prec Q') by(rule eqvt)
    hence }Q\longmapsto(\mathrm{ rev perm • a)< < (rev perm •x)> < (rev perm • Q')
    by(simp add: name-rev-per)
    moreover from xFreshP have (rev perm •x)\sharpP by(simp add: name-fresh-left)
    ultimately obtain }\mp@subsup{P}{}{\prime}\mathrm{ where PTrans: P }\Longrightarrow(\mathrm{ rev perm }\cdota)<\nu(\mathrm{ rev perm }\cdotx)
\imath'
                            and }\mp@subsup{P}{}{\prime}\mathrm{ RelQ':( }\mp@subsup{P}{}{\prime}\mathrm{ , rev perm • Q') ( Rel using PSimQ
    by(blast dest: simE)
    from PTrans have (perm • P)\Longrightarrow(perm • rev perm • a)<\nu(perm • rev perm •
x)>}\prec\mathrm{ perm • P'
    by(rule Weak-Early-Step-Semantics.eqvtI)
    hence L1:(perm • P)\Longrightarrowa<\nux> \prec (perm • P') by(simp add: name-per-rev)
    from P'RelQ' RelRel' have ( }\mp@subsup{P}{}{\prime}\mathrm{ , rev perm • Q') & Rel' by blast
    with EqvtRel' have (perm • P', perm • (rev perm • Q ')) \in Rel'
    by(rule eqvtRelI)
    hence (perm • P', Q') \in Rel' by(simp add: name-per-rev)
    with L1 show ?case by blast
next
    case(Free \alpha Q ')
```



```
    hence (rev perm • (perm • Q))\longmapsto \longmapstoev perm • ( }\alpha\prec\mp@subsup{Q}{}{\prime})\mathrm{ by(rule eqvts)
    hence Q\longmapsto(rev perm • \alpha)\prec(rev perm • Q') by(simp add: name-rev-per)
    with PSimQ obtain P' where PTrans: P\Longrightarrow(rev perm \cdot \alpha)\prec 乍
                            and PRel: (P',(rev perm • Q')) \in Rel
    by(blast dest: simE)
    from PTrans have (perm • P)\Longrightarrow(perm • rev perm • \alpha)\prec perm • P'
    by(rule Weak-Early-Step-Semantics.eqvtI)
    hence L1: (perm •P)\Longrightarrow\alpha\prec(perm • P') by (simp add: name-per-rev)
    from PRel EqvtRel' RelRel' have ((perm • P'), (perm • (rev perm • Q'))) \inRel'
        by(force intro: eqvtRelI)
    hence ((perm • P'), Q') \inRel' by(simp add: name-per-rev)
    with L1 show ?case by blast
qed
```

```
lemma reflexive:
    fixes \(P\) :: \(p i\)
    and Rel :: \((p i \times p i)\) set
    assumes \(I d \subseteq R e l\)
    shows \(P \rightsquigarrow «\) Rel» \(P\)
using assms
by (auto intro: Weak-Early-Step-Semantics.singleActionChain
    simp add: weakStepSimulation-def weakFreeTransition-def)
lemma transitive:
    fixes \(P \quad:: p i\)
    and \(\quad Q \quad:: p i\)
    and \(R \quad:: p i\)
    and Rel :: \((p i \times p i)\) set
    and Rel' :: \((p i \times p i)\) set
    and Rel \({ }^{\prime \prime}::(p i \times p i)\) set
    assumes PSimQ: \(P \rightsquigarrow «\) Rel» \(Q\)
    and \(\quad\) SimR: \(Q \rightsquigarrow «\) Rel'» \(^{\prime}\) R
    and Eqvt: equt Rel
    and Eqvt": eqvt Rel \({ }^{\prime \prime}\)
    and Trans: Rel \(O\) Rel \(^{\prime} \subseteq\) Rel \(^{\prime \prime}\)
    and \(\quad \operatorname{Sim}: \wedge S T .(S, T) \in \operatorname{Rel} \Longrightarrow S \rightsquigarrow<\operatorname{Rel}>T\)
    and \(\quad P R e l Q:(P, Q) \in \operatorname{Rel}\)
    shows \(P \rightsquigarrow «\) Rel \(^{\prime \prime} » R\)
proof -
    from Eqvt" show ?thesis
    proof \((\) induct rule: simCasesCont \([o f-(P, Q)])\)
    case(Bound a \(x R^{\prime}\) )
    have \(x \sharp(P, Q)\) by fact
    hence \(x\) FreshP: \(x \sharp P\) and \(x F r e s h Q: x \sharp Q\) by(simp add: fresh-prod) +
    have \(R\) Trans: \(R \longmapsto a<\nu x>\prec R^{\prime}\) by fact
    from \(x\) Fresh \(Q\) QSimR RTrans obtain \(Q^{\prime}\) where \(Q\) Trans: \(Q \Longrightarrow a<\nu x>\prec Q^{\prime}\)
                                    and \(Q^{\prime}\) Rel \(^{\prime} R^{\prime}:\left(Q^{\prime}, R^{\prime}\right) \in \operatorname{Rel}^{\prime}\)
            by (blast dest: simE)
    with PSimQ Sim Eqvt PRelQ QTrans xFreshP have \(\exists P^{\prime} . P \Longrightarrow a<\nu x>\prec P^{\prime}\)
\(\wedge\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}\)
        by (blast intro: simE2)
    then obtain \(P^{\prime}\) where PTrans: \(P \Longrightarrow a<\nu x>\prec P^{\prime}\) and \(P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in\)
Rel by blast
    moreover from \(P^{\prime}\) RelQ \(Q^{\prime} Q^{\prime}\) Rel \(^{\prime} R^{\prime}\) Trans have \(\left(P^{\prime}, R^{\prime}\right) \in R e l^{\prime \prime}\) by blast
    ultimately show ?case by blast
```

```
    next
    case(Free \alpha R
    have RTrans: R\longmapsto\alpha\prec '' by fact
    with QSimR obtain }\mp@subsup{Q}{}{\prime}\mathrm{ where QTrans: Q }\Longrightarrow\alpha\prec\mp@subsup{Q}{}{\prime}\mathrm{ and }\mp@subsup{Q}{}{\prime}Rel\mp@subsup{R}{}{\prime}:(\mp@subsup{Q}{}{\prime},\mp@subsup{R}{}{\prime}
ERel'
            by(blast dest: simE)
    from PSimQ Sim Eqvt PRelQ QTrans have }\exists\mp@subsup{P}{}{\prime}.P\Longrightarrow\alpha\prec\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})
Rel
            by(blast intro: simE2)
                            then obtain P' where PTrans: P\Longrightarrow\alpha\prec P
by blast
    from P'RelQ' Q'RelR' Trans have ( }\mp@subsup{P}{}{\prime},\mp@subsup{R}{}{\prime})\in\mathrm{ Rel '' by blast
    with PTrans show ?case by blast
    qed
qed
lemma strongSimWeakSim:
    fixes P :: pi
    and }Q ::p
    and Rel :: (pi\timespi) set
    assumes PSimQ: P}\rightsquigarrow[Rel]
    shows P\rightsquigarrow«Rel» Q
proof(induct rule: simCases)
    case(Bound a x Q')
    have }Q\longmapstoa<\nux>\prec\prec\mp@subsup{Q}{}{\prime}\mathrm{ and }x\sharpP\mathrm{ by fact+
    with PSimQ obtain P' where PTrans: P\longmapstoa<\nux>}\prec\mp@subsup{P}{}{\prime}\mathrm{ and }\mp@subsup{P}{}{\prime}RelQ':(\mp@subsup{P}{}{\prime}
Q')}\in\textrm{Rel
    by(blast dest: Strong-Early-Sim.elim)
    from PTrans have P\Longrightarrowa<\nux> \prec P'
    by(force intro:Weak-Early-Step-Semantics.singleActionChain simp add: weak-
FreeTransition-def)
    with P'RelQ' show ?case by blast
next
    case(Free \alpha Q')
    have }Q\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    with PSimQ obtain P' where PTrans: P\longmapsto\alpha \prec P' and P'RelQ':( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})
Rel
    by(blast dest: Strong-Early-Sim.elim)
    from PTrans have P\Longrightarrow\alpha\prec\mp@subsup{P}{}{\prime}\mathbf{by}(rule Weak-Early-Step-Semantics.singleActionChain)
    with P'RelQ' show ?case by blast
qed
lemma weakSimWeakEqSim:
    fixes P :: pi
    and Q :: pi
    and Rel :: (pi\times pi) set
```

```
    assumes \(P \rightsquigarrow «\) Rel» \(Q\)
    shows \(P \rightsquigarrow<\) Rel \(>Q\)
using assms
by (force simp add: weakStepSimulation-def Weak-Early-Sim.weakSimulation-def weak-
FreeTransition-def)
end
theory Weak-Early-Cong
    imports Weak-Early-Bisim Weak-Early-Step-Sim Strong-Early-Bisim
begin
definition weakCongruence \(:: p i \Rightarrow p i \Rightarrow\) bool (infixr \(\simeq 65\) )
where \(P \simeq Q \equiv P \rightsquigarrow\) weakBisim» \(Q \wedge Q \rightsquigarrow «\) weakBisim» \(P\)
lemma weakCongISym[consumes 1, case-names cSym cSim]:
    fixes \(P\) :: \(p i\)
    and \(\quad Q:: p i\)
    assumes Prop \(P Q\)
    and \(\quad \wedge R S\). Prop \(R S \Longrightarrow\) Prop \(S R\)
    and \(\quad \bigwedge R S\). Prop \(R S \Longrightarrow(F R) \rightsquigarrow «\) weakBisim» \((F S)\)
    shows \(F P \simeq F Q\)
using assms
by(auto simp add: weakCongruence-def)
lemma weakCongISym2[consumes 1, case-names cSim]:
    fixes \(P:: p i\)
    and \(\quad Q:: p i\)
    assumes \(P \simeq Q\)
    and \(\quad \bigwedge R S . R \simeq S \Longrightarrow(F R) \rightsquigarrow «\) weakBisim» \((F S)\)
    shows \(F P \simeq F Q\)
using assms
by(auto simp add: weakCongruence-def)
lemma weakCongEE:
    fixes \(P\) :: \(p i\)
    and \(\quad Q:: p i\)
    and \(s::(\) name \(\times\) name \()\) list
    assumes \(P \simeq Q\)
    shows \(P \rightsquigarrow\) weakBisim» \(Q\)
    and \(\quad Q \rightsquigarrow «\) weakBisim» \(P\)
using assms
```

```
by(auto simp add: weakCongruence-def)
lemma weakCongI:
    fixes P :: pi
    and }Q:: p
    assumes P\rightsquigarrow«weakBisim» Q
    and }\quadQ\rightsquigarrow«weakBisim» P
    shows P\simeqQ
using assms
by(auto simp add: weakCongruence-def)
lemma eqvtI[eqvt]:
    fixes P :: pi
    and }Q:: p
    and p :: name prm
    assumes }P\simeq
    shows (p\cdotP)\simeq(p\cdotQ)
using assms
by(auto simp add: weakCongruence-def intro: eqvtI)
lemma strongBisim WeakCong:
    fixes P :: pi
    and }\quadQ::p
    assumes P~Q
    shows P\simeqQ
proof -
    have }\PQ.P\rightsquigarrow[bisim] Q\LongrightarrowP\rightsquigarrow«weakBisim» Q
    proof -
        fix PQ
        assume P\rightsquigarrow[bisim] Q
        hence P\rightsquigarrow«bisim»Q by(rule Weak-Early-Step-Sim.strongSimWeakSim)
        moreover have bisim \subseteq weakBisim
            by(auto intro: strongBisim WeakBisim)
        ultimately show P}\rightsquigarrow<<weakBisim» Q by(rule Weak-Early-Step-Sim.monotonic)
    qed
    with assms show ?thesis
        by(blast intro:weakCongI dest: Strong-Early-Bisim.bisimE)
qed
lemma congruenceWeakBisim:
    fixes P :: pi
    and }Q:: p
```

```
    assumes \(P \simeq Q\)
    shows \(P \approx Q\)
using assms
proof -
    let \(? X=\{(P, Q) \mid P Q . P \simeq Q\}\)
    from assms have \((P, Q) \in ? X\) by simp
    thus ?thesis
    proof (induct rule: weakBisimCoinduct)
        case \((c \operatorname{Sim} P Q)\)
        \{
            fix \(P Q\)
            assume \(P \simeq Q\)
            hence \(P \rightsquigarrow\) weakBisim» \(Q\) by (simp add: weakCongruence-def)
    hence \(P \rightsquigarrow «\left(? X \cup\right.\) weakBisim) \({ }^{2} Q\) by (rule-tac Weak-Early-Step-Sim.monotonic)
auto
            hence \(P \rightsquigarrow<(? X \cup\) weakBisim \()>Q\) by(rule weakSimWeakEqSim)
    \}
            with \(\langle(P, Q) \in\) ? \(X\rangle\) show ?case by auto
    next
        case \((c \operatorname{Sym} P Q)\)
        thus ?case by (auto simp add: weakCongruence-def)
    qed
qed
lemma reflexive:
    fixes \(P\) :: \(p i\)
    shows \(P \simeq P\)
proof -
    from Weak-Early-Bisim.reflexive have \(\bigwedge P . P \rightsquigarrow «\) weakBisim» \(P\)
        by (blast intro: Weak-Early-Step-Sim.reflexive)
    thus ?thesis
        by (force simp add: substClosed-def weakCongruence-def)
qed
lemma symetric:
    fixes \(P\) :: \(p i\)
    and \(\quad Q:: p i\)
    assumes \(P \simeq Q\)
    shows \(Q \simeq P\)
using assms
by(force simp add: substClosed-def weakCongruence-def)
lemma transitive:
    fixes \(P\) :: \(p i\)
    and \(\quad Q:: p i\)
```

```
    and }R::p
    assumes }P\simeq
    and }Q\simeq
    shows P\simeqR
proof -
    have Goal: \bigwedgePQ R.\llbracketP\rightsquigarrow«weakBisim» Q; Q\rightsquigarrow«weakBisim» R; P}\approxQ\rrbracket
P\rightsquigarrow《weakBisim» R
    using Weak-Early-Bisim.eqvt Weak-Early-Bisim.weakBisimE Weak-Early-Bisim.transitive
    by(blast intro:Weak-Early-Step-Sim.transitive)
    from assms show ?thesis
        apply(simp add: weakCongruence-def) using assms
    by(blast intro: Goal dest: congruenceWeakBisim symetric)
qed
end
theory Weak-Early-Bisim-Subst
    imports Weak-Early-Bisim Strong-Early-Bisim-Subst
begin
consts weakBisimSubst :: (pi \times pi) set
abbreviation weakEarlyBisimSubstJudge (infixr *s 65) where P **s}Q\equiv(P
Q)\in(substClosed weakBisim)
lemma congBisim:
    fixes P :: pi
    and }Q::p
    assumes P}\mp@subsup{\approx}{}{s}
    shows P}\approx
using assms substClosedSubset
by blast
lemma strongBisimWeakBisim:
    fixes P :: pi
    and }Q::p
    assumes }P\mp@subsup{~}{}{s}
    shows P}\mp@subsup{\approx}{}{s}
using assms
by(auto simp add: substClosed-def intro: strongBisimWeakBisim)
lemma eqvt:
    shows eqvt (substClosed weakBisim)
by(rule eqvtSubstClosed[OF Weak-Early-Bisim.eqvt])
```

```
lemma eqvtI[eqvt]:
    fixes \(P\) :: \(p i\)
    and \(\quad Q:: p i\)
    and perm :: name prm
    assumes \(P \approx^{s} Q\)
    shows \((\) perm \(\cdot P) \approx^{s}(\) perm \(\cdot Q)\)
using assms
by(rule eqvtRelI[OF eqvt])
lemma reflexive:
    fixes \(P\) :: \(p i\)
    shows \(P \approx^{s} P\)
by(force simp add: substClosed-def intro: Weak-Early-Bisim.reflexive)
lemma symetric:
    fixes \(P:: p i\)
    and \(\quad Q:: p i\)
    assumes \(P \approx^{s} Q\)
    shows \(Q \approx^{s} P\)
using assms
by(force simp add: substClosed-def intro: Weak-Early-Bisim.symetric)
lemma transitive:
    fixes \(P:: p i\)
    and \(\quad Q:: p i\)
    and \(\quad R:: p i\)
    assumes \(P \approx^{s} Q\)
    and \(\quad Q \approx^{s} R\)
    shows \(P \approx^{s} R\)
using assms
by(force simp add: substClosed-def intro: Weak-Early-Bisim.transitive)
lemma partUnfold:
    fixes \(P\) :: \(p i\)
    and \(\quad Q:: p i\)
    and \(s::(\) name \(\times\) name \()\) list
    assumes \(P \approx^{s} Q\)
    shows \(P[\langle s\rangle] \approx^{s} Q[\langle s\rangle]\)
using assms
```

```
proof(auto simp add: substClosed-def)
    fix s}\mp@subsup{s}{}{\prime
    assume }\foralls.P[<s>] \approxQ[<s>
    hence P[<(s@s')>]\approxQ[<(s@ s')>] by blast
    moreover have P[<(s@s')>] =(P[<s>])[<\mp@subsup{s}{}{\prime}>]
        by(induct s', auto)
    moreover have Q[<(s@s')>]=(Q[<s>])[<\mp@subsup{s}{}{\prime}>]
        by(induct s', auto)
    ultimately show (P[<s>])[<\mp@subsup{s}{}{\prime}\rangle]\approx(Q[<s>])[<\mp@subsup{s}{}{\prime}\rangle]
        by simp
qed
end
theory Weak-Early-Cong-Subst
    imports Weak-Early-Cong Weak-Early-Bisim-Subst Strong-Early-Bisim-Subst
begin
consts congruenceSubst :: (pi\timespi) set
definition weakCongruenceSubst (infixr }\mp@subsup{\simeq}{}{s}65\mathrm{ ) where }P\mp@subsup{\simeq}{}{s}Q\equiv\forall\sigma.P[<\sigma>
\simeq Q [ < \sigma > ]
lemma unfoldE:
    fixes P :: pi
    and }Q:: p
    and s::(name }\times\mathrm{ name) list
    assumes P}\mp@subsup{\simeq}{}{s}
    shows P[<s>] \rightsquigarrow«weakBisim» Q[<s>]
    and }Q[<s\rangle]\rightsquigarrow«weakBisim» P[<s>
proof -
    from assms show P[<s>] \rightsquigarrow«weakBisim» Q[<s>] by(simp add: weakCongru-
enceSubst-def weakCongruence-def)
next
    from assms show Q[<s>] \rightsquigarrow«weakBisim» P[<s>] by(simp add: weakCongru-
enceSubst-def weakCongruence-def)
qed
lemma unfoldI:
    fixes P :: pi
    and }Q:: p
    assumes \bigwedges. P[<s>] \rightsquigarrow«weakBisim» Q[<s>]
    and \s.Q[<s>] \rightsquigarrow«weakBisim» P[<s>]
    shows P}\mp@subsup{\simeq}{}{s}
```

```
using assms
by (simp add: weakCongruenceSubst-def weakCongruence-def)
lemma weakCongWeakEq:
    fixes \(P\) :: \(p i\)
    and \(\quad Q:: p i\)
    assumes \(P \simeq^{s} Q\)
    shows \(P \simeq Q\)
using assms
apply (simp add: weakCongruenceSubst-def weakCongruence-def)
apply (erule-tac \(x=[]\) in allE)
by auto
lemma eqvtI:
    fixes \(P\) :: pi
    and \(\quad Q:: p i\)
    and \(\quad p::\) name prm
    assumes \(P \simeq^{s} Q\)
    shows \((p \cdot P) \simeq^{s}(p \cdot Q)\)
proof (simp add: weakCongruenceSubst-def, rule allI)
    fix \(s\)
    from assms have \(P[<(\) rev \(p \cdot s)>] \simeq Q[<(\) rev \(p \cdot s)>]\) by (auto simp add:
weakCongruenceSubst-def)
    thus \((p \cdot P)[<s\rangle] \simeq(p \cdot Q)[<s\rangle] \mathbf{b y}(\) drule-tac \(p=p\) in Weak-Early-Cong.eqvtI)
(simp add: equts name-per-rev)
qed
lemma strongEqWeakCong:
    fixes \(P\) :: \(p i\)
    and \(\quad Q:: p i\)
    assumes \(P \sim^{s} Q\)
    shows \(P \simeq^{s} Q\)
using assms
by (auto intro: strongBisim WeakCong simp add: substClosed-def weakCongruence-
Subst-def)
lemma congSubstBisimSubst:
    fixes \(P\) :: \(p i\)
    and \(\quad Q:: p i\)
    assumes \(P \simeq^{s} Q\)
shows \(P \approx^{s} Q\)
```

using assms
by (auto intro: congruence WeakBisim simp add: substClosed-def weakCongruence-Subst-def)
lemma reflexive:
fixes $P$ :: $p i$
shows $P \simeq^{s} P$
proof -
from Weak-Early-Bisim.reflexive have $\bigwedge P . P \rightsquigarrow «$ weakBisim» $P$ by (blast intro: Weak-Early-Step-Sim.reflexive)
thus ?thesis
by (force simp add: weakCongruenceSubst-def weakCongruence-def)
qed
lemma symetric:
fixes $P$ :: $p i$
and $\quad Q:: p i$
assumes $P \simeq^{s} Q$
shows $Q \simeq^{s} P$
using assms by (auto simp add: weakCongruenceSubst-def weakCongruence-def)
lemma transitive:
fixes $P$ :: $p i$
and $\quad Q:: p i$
and $\quad R:: p i$
assumes $P \simeq^{s} Q$
and $\quad Q \simeq^{s} R$
shows $P \simeq^{s} R$
using assms by (auto simp add: weakCongruenceSubst-def intro: Weak-Early-Cong.transitive)
lemma partUnfold:
fixes $P$ :: $p i$
and $\quad Q:: p i$
and $s::($ name $\times$ name $)$ list
assumes $P \simeq^{s} Q$
shows $P[\langle s\rangle] \simeq^{s} Q[<s>]$
using assms
proof (auto simp add: weakCongruenceSubst-def)
fix $s^{\prime}$
assume $\forall s . P[<s>] \simeq Q[<s>]$
hence $P\left[<\left(s @ s^{\prime}\right)>\right] \simeq Q\left[<\left(s @ s^{\prime}\right)>\right]$ by blast
moreover have $\left.\left.P\left[<\left(s @ s^{\prime}\right)>\right]=(P[<s\rangle]\right)\left[<s^{\prime}\right\rangle\right]$

```
    by(induct s', auto)
    moreover have Q[<(s@s')>]=(Q[<s>])[<\mp@subsup{s}{}{\prime}>]
    by(induct s', auto)
    ultimately show }(P[<s\rangle])[<\mp@subsup{s}{}{\prime}\rangle]\simeq(Q[<s>])[<\mp@subsup{s}{}{\prime}\rangle
    by simp
qed
end
theory Weak-Early-Step-Sim-Pres
    imports Weak-Early-Step-Sim
begin
lemma tauPres:
    fixes P :: pi
    and Q :: pi
    and Rel :: (pi\times pi) set
    and Rel':: (pi }\times pi) se
    assumes PRelQ: (P,Q)\inRel
    shows }\tau.(P)\rightsquigarrow<<Rel»\tau.(Q
proof(induct rule: simCases)
    case(Bound a x Q')
    have }\tau.(Q)\longmapstoa<\nux><<\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    hence False by(induct rule: tauCases', auto)
    thus ?case by simp
next
    case(Free \alpha Q ')
    have}\tau.(Q)\longmapsto(\alpha\prec\mp@subsup{Q}{}{\prime})\mathrm{ by fact
    thus ?case
    proof(induct rule: tauCases', auto simp add: pi.inject residual.inject)
        have }\tau.(P)\Longrightarrow\tau\precP\mathrm{ by(rule Weak-Early-Step-Semantics.Tau)
        with PRelQ show }\exists\mp@subsup{P}{}{\prime}.\tau.(P)\Longrightarrow\tau\prec\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},Q)\in\mathrm{ Rel by blast
    qed
qed
lemma inputPres:
    fixes P :: pi
    and }x\mathrm{ :: name
    and Q :: pi
    and a :: name
    and Rel :: (pi\times pi) set
    assumes PRelQ: }\forally.(P[x::=y],Q[x::=y])\in\operatorname{Rel
    and Eqvt: eqvt Rel
    shows }a<x>.P\rightsquigarrow<Rel> a<x>.
```

```
using Eqvt
proof \((\) induct rule: simCasesCont \([\) where \(C=(x, a, P, Q)])\)
    case(Bound by \(Q^{\prime}\) )
    from \(\langle y \sharp(x, a, P, Q)\) ) have \(y \neq x y \neq a y \sharp P y \sharp Q\) by simp+
    from \(\left.\langle a<x\rangle . Q \longmapsto b<\nu y\rangle \prec Q^{\prime}\right\rangle\langle y \neq a\rangle\langle y \neq x\rangle\langle y \sharp Q\rangle\) show ?case
        by (erule-tac inputCases') auto
next
    case \(\left(\right.\) Free \(\left.\alpha Q^{\prime}\right)\)
    from \(\left.\langle a<x\rangle . Q \longmapsto \alpha \prec Q^{\prime}\right\rangle\)
    show ? case
    proof(induct rule: inputCases)
    case(cInput u)
    have \(a<x>. P \Longrightarrow(a<u>) \prec(P[x::=u])\)
            by(rule Weak-Early-Step-Semantics.Input)
    moreover from \(P \operatorname{Rel} Q\) have \((P[x::=u], Q[x::=u]) \in\) Rel by auto
    ultimately show ? case by blast
    qed
qed
lemma outputPres:
    fixes \(P \quad:: p i\)
    and \(\quad Q \quad:: p i\)
    and \(a\) :: name
    and \(b\) :: name
    and Rel :: \((p i \times p i)\) set
    and Rel' \(::(p i \times p i)\) set
    assumes PRelQ: \((P, Q) \in \operatorname{Rel}\)
    shows \(a\{b\} . P \rightsquigarrow «\) Rel» \(a\{b\} . Q\)
proof (induct rule: simCases)
    case(Bound c \(x Q^{\prime}\) )
    have \(a\{b\} . Q \longmapsto c<\nu x>\prec Q^{\prime}\) by fact
    hence False by (induct rule: outputCases', auto)
    thus? case by simp
next
    case (Free \(\alpha Q^{\prime}\) )
    have \(a\{b\} . Q \longmapsto \alpha \prec Q^{\prime}\) by fact
    thus \(\exists P^{\prime} . a\{b\} . P \Longrightarrow \alpha \prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in\) Rel
    proof \((\) induct rule: outputCases', auto simp add: pi.inject residual.inject)
        have \(a\{b\} . P \Longrightarrow a[b] \prec P\) by (rule Weak-Early-Step-Semantics.Output)
        with PRelQ show \(\exists P^{\prime} . a\{b\} . P \Longrightarrow a[b] \prec P^{\prime} \wedge\left(P^{\prime}, Q\right) \in\) Rel by blast
    qed
qed
lemma matchPres:
    fixes \(P\) :: \(p i\)
    and \(\quad Q \quad:: p i\)
    and \(a\) :: name
```

```
    and b :: name
    and Rel :: (pi\times pi) set
    and Rel':: (pi\times pi) set
    assumes PSimQ: P\rightsquigarrow«Rel»Q
    and RelRel': Rel \subseteqRel'
    shows [a\frownb]P\rightsquigarrow<<Rel'» [a\frownb]Q
proof(induct rule: simCases)
    case(Bound c x Q')
    have }x\sharp[a\frownb]P\mathrm{ by fact
    hence xFreshP:(x::name) #P by simp
    have [a\frownb]Q\longmapstoc<\nux>}\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    thus ?case
    proof(induct rule: matchCases)
    case Match
    have }Q\longmapstoc<\nux>\prec\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    with PSimQ xFreshP obtain P' where PTrans: P\Longrightarrowc<\nux>}\prec\mp@subsup{P}{}{\prime
                            and P}\mp@subsup{P}{}{\prime}\mathrm{ RelQ'Q}:(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\operatorname{Rel
        by(blast dest: simE)
    from PTrans have [a\frowna]P\Longrightarrowc<\nux>}\prec\mp@subsup{P}{}{\prime}\mathrm{ by(rule Weak-Early-Step-Semantics.Match)
    moreover from P'RelQ' RelRel' have ( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mp@subsup{\mathrm{ Rel' }}{}{\prime}\mathrm{ by blast
    ultimately show ?case by blast
    qed
next
    case(Free \alpha Q )
    have [a\frownb]Q\longmapsto\alpha\prec Q' by fact
    thus ?case
    proof(induct rule: matchCases)
        case Match
    have }Q\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    with PSimQ obtain P' where PTrans: P\Longrightarrow\alpha\prec P' and PRel: ( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})
Rel
        by(blast dest: simE)
    from PTrans have [a\frowna]P\Longrightarrow\alpha\prec P'by(rule Weak-Early-Step-Semantics.Match)
        with RelRel' PRel show ?case by blast
    qed
qed
lemma mismatchPres:
fixes P :: pi
and }Q :: p
and a :: name
and b :: name
and Rel ::(pi\times pi) set
and Rel':: (pi\timespi) set
assumes PSimQ: P\rightsquigarrow<<Rel» Q
and RelRel':Rel \subseteqRel'
```

```
    shows [a\not=b]P\rightsquigarrow<<Re\mp@subsup{l}{}{\prime}»[a\not=b]Q
proof(induct rule: simCases)
    case(Bound c x Q')
    have }x\sharp[a\not=b]P\mathrm{ by fact
    hence xFreshP:(x::name) #P by simp
    have [a\not=b]Q\longmapstoc<\nux>}\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    thus ?case
    proof(induct rule: mismatchCases)
        case Mismatch
        have aineqb: }a\not=b\mathrm{ by fact
    have }Q\longmapstoc<\nux>\prec\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    with PSimQ xFreshP obtain }\mp@subsup{P}{}{\prime}\mathrm{ where PTrans: P }\Longrightarrowc<\nux>\prec\mp@subsup{P}{}{\prime
                            and P'RelQ':}(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mathrm{ Rel
            by(blast dest: simE)
    from PTrans aineqb have [a\not=b]P\Longrightarrowc<\nux>}\prec\mp@subsup{P}{}{\prime}\mathrm{ by(rule Weak-Early-Step-Semantics.Mismatch)
        moreover from P'RelQ' RelRel' have ( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mp@subsup{\mathrm{ Rel' }}{}{\prime}\mathrm{ by blast
        ultimately show ?case by blast
    qed
next
    case(Free \alpha Q ')
    have [a\not=b]Q\longmapsto\alpha\prec Q' by fact
    thus ?case
    proof(induct rule: mismatchCases)
        case Mismatch
        have }Q\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
        with PSimQ obtain P' where PTrans: P\Longrightarrow\alpha\prec 盾 and PRel: ( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})
Rel
        by(blast dest: simE)
    from PTrans }\langlea\not=b\rangle\mathrm{ have [ }a\not=b]P\Longrightarrow\alpha\prec\mp@subsup{P}{}{\prime}\mathbf{by}(rule Weak-Early-Step-Semantics.Mismatch)
        with RelRel' PRel show ?case by blast
    qed
qed
lemma sumPres:
    fixes P :: pi
    and }Q:: p
    and }R:: p
    assumes PSimQ:P\rightsquigarrow«Rel»Q
    and RelRel':Rel }\subseteqRel'
    and }C:Id\subseteqRel'
    shows }P\oplusR\rightsquigarrow«Rel'»Q\oplus
proof(induct rule: simCases)
    case(Bound a x Q')
    have }x\sharpP\oplusR\mathrm{ by fact
    hence xFreshP:(x::name) \sharpP and xFreshR: }x\sharpR\mathrm{ by simp +
    have }Q\oplusR\longmapstoa<\nux>\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
```

```
thus ?case
proof(induct rule: sumCases)
    case Sum1
    have }Q\longmapstoa<\nux>\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
        with xFreshP PSimQ obtain P' where PTrans: P \Longrightarrowa<\nux>}\prec\mp@subsup{P}{}{\prime}\mathrm{ and
P'RelQ':( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\operatorname{Rel
        by(blast dest: simE)
    from PTrans have P}\oplusR\Longrightarrowa<\nux>\prec\mp@subsup{P}{}{\prime}\mathbf{by}(rule Weak-Early-Step-Semantics.Sum1)
    moreover from P'RelQ' RelRel' have ( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\inRel' by blas
    ultimately show ?case by blast
next
    case Sum2
    from }\langleR\longmapstoa<\nux>\prec Q '> have P \oplusR\longmapstoa<\nux> \prec Q'by(rule Early-Semantics.Sum2) 
```



```
    moreover from C have ( }\mp@subsup{Q}{}{\prime},\mp@subsup{Q}{}{\prime})\inRe\mp@subsup{R}{}{\prime}\mathrm{ by blast
    ultimately show ?case by blast
    qed
next
    case(Free \alpha Q ')
    have }Q\oplusR\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    thus ?case
    proof(induct rule: sumCases)
        case Sum1
        have }Q\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
        with PSimQ obtain P' where PTrans: P\Longrightarrow\alpha\prec P' and PRel: ( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})
Rel
        by(blast dest: simE)
    from PTrans have P\oplusR\Longrightarrow\alpha\prec P' by(rule Weak-Early-Step-Semantics.Sum1)
        with RelRel' PRel show ?case by blast
    next
        case Sum2
        from <R\longmapsto\alpha\prec Q'> have }P\oplusR\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime}\mathbf{by}(\mathrm{ rule Early-Semantics.Sum2)
    hence }P\oplusR\Longrightarrow\alpha\prec\mp@subsup{Q}{}{\prime}\mathbf{by}(rule Weak-Early-Step-Semantics.singleActionChain)
        moreover from C have ( }\mp@subsup{Q}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mp@subsup{R}{el}{
        ultimately show ?case by blast
    qed
qed
lemma parPres:
    fixes P :: pi
    and }Q\quad::p
    and }R\quad::p
    and T :: pi
    and Rel ::(pi\timespi) set
    and Rel' :: (pi\timespi) set
    and Rel"\prime::(pi\timespi) set
    assumes PSimQ: P}\rightsquigarrow<<\mathrm{ Rel» Q
    and PRelQ: }(P,Q)\in\operatorname{Rel
```

```
    and Par: }\quad\STU.(S,T)\inRel\Longrightarrow(S|U,T|U)\inRel'\,
    and Res: \}\\Tx.(S,T)\inRe\mp@subsup{l}{}{\prime}\Longrightarrow(<\nux>S,<\nux>T)\inRel'
    shows P| R\rightsquigarrow<<Rel`» Q || R
proof -
    show ?thesis
    proof(induct rule: simCases)
    case(Bound a x Q')
    have }x\sharpP|R\mathrm{ by fact
    hence xFreshP: x\sharpP and xFreshR: x #R by simp+
    have }Q|R\longmapstoa<\nux>\prec\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    thus ?case
    proof(induct rule: parCasesB)
        case(cPar1 Q')
        have QTrans: Q\longmapstoa<\nux>}\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
        from xFreshP PSimQ QTrans obtain P' where PTrans:P\Longrightarrowa<\nux>}\prec\mp@subsup{P}{}{\prime
                            and \mp@subsup{P}{}{\prime}RelQ':}(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\operatorname{Rel
            by(blast dest: simE)
    from PTrans xFreshR have P|R\Longrightarrowa<\nux> \prec( (P'|R) by(rule Weak-Early-Step-Semantics.Par1B)
        moreover from P'RelQ' have ( }\mp@subsup{P}{}{\prime}|R,\mp@subsup{Q}{}{\prime}|R)\inRe\mp@subsup{R}{}{\prime}\mathrm{ by(rule Par)
        ultimately show ?case by blast
    next
        case(cPar2 R')
        from <R\longmapstoa<\nux> \prec R'>\langlex\sharpP> have }P|R\longmapstoa<\nux> \prec (P| | R'
            by(rule Early-Semantics.Par2B)
    hence }P|R\Longrightarrowa<\nux>< (P| R') by(rule Weak-Early-Step-Semantics.singleActionChain)
        moreover from PRelQ have (P| R',Q| R')\inRel' by(rule Par)
        ultimately show ?case by blast
    qed
next
    case(Free \alpha QR')
    have }Q|R\longmapsto\alpha\precQ\mp@subsup{R}{}{\prime}\mathrm{ by fact
    thus ?case
    proof(induct rule: parCasesF[of - - - (P,R)])
        case(cPar1 Q')
        have }Q\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
        with PSimQ obtain P' where PTrans: P\Longrightarrow\alpha\prec P' and PRel: ( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime}
ERel
        by(blast dest: simE)
    from PTrans have Trans: P|R\Longrightarrow\alpha\prec P'|R by(rule Weak-Early-Step-Semantics.Par1F)
        moreover from PRel have ( }\mp@subsup{P}{}{\prime}|R,\mp@subsup{Q}{}{\prime}|R)\in\mp@subsup{R}{\mathrm{ Rl' by(blast intro: Par)}}{\mathrm{ ( }
        ultimately show ?case by blast
    next
        case(cPar2 R')
        from <R\longmapsto\alpha\prec R'〉 have P|R\longmapsto\alpha\prec(P| R')
            by(rule Early-Semantics.Par2F)
    hence }P|R\Longrightarrow\alpha\prec(P|\mp@subsup{R}{}{\prime})\mathrm{ by(rule Weak-Early-Step-Semantics.singleActionChain)
        moreover from PRelQ have ( }P|\mp@subsup{R}{}{\prime},Q| R')\in\mp@subsup{R}{}{\prime}\mp@subsup{|}{}{\prime}\mathrm{ by(rule Par)
        ultimately show ?case by blast
```

```
next
    case \(\left(c C o m m 1 \quad Q^{\prime} R^{\prime} a b\right)\)
    have \(Q\) Trans: \(Q \longmapsto a<b>\prec Q^{\prime}\) and RTrans: \(R \longmapsto a[b] \prec R^{\prime}\) by fact +
    from PSimQ QTrans obtain \(P^{\prime}\) where PTrans: \(P \Longrightarrow a<b>\prec P^{\prime}\)
                            and \(P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}\)
        by(blast dest: \(\operatorname{sim} E)\)
from RTrans have \(R \Longrightarrow a[b] \prec R^{\prime}\) by (rule Weak-Early-Step-Semantics.singleActionChain)
with PTrans have \(P\left\|R \Longrightarrow \tau \prec P^{\prime}\right\| R^{\prime}\) by(rule Weak-Early-Step-Semantics.Comm1)
    moreover from \(P^{\prime} \operatorname{Rel} Q^{\prime}\) have \(\left(P^{\prime}\left\|R^{\prime}, Q^{\prime}\right\| R^{\prime}\right) \in\) Rel \(^{\prime}\) by (rule Par)
    ultimately show ?case by blast
next
    case \(\left(c C o m m 2 Q^{\prime} R^{\prime} a b\right)\)
    have \(Q\) Trans: \(Q \longmapsto a[b] \prec Q^{\prime}\) and RTrans: \(R \longmapsto a<b>\prec R^{\prime}\) by fact +
    from PSimQ QTrans obtain \(P^{\prime}\) where PTrans: \(P \Longrightarrow a[b] \prec P^{\prime}\)
                                    and \(P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}\)
    by(blast dest: \(\operatorname{sim} E)\)
from RTrans have \(R \Longrightarrow a<b>\prec R^{\prime}\) by (rule Weak-Early-Step-Semantics.singleActionChain)
with PTrans have \(P\left\|R \Longrightarrow \tau \prec P^{\prime}\right\| R^{\prime}\) by(rule Weak-Early-Step-Semantics.Comm2)
    moreover from \(P^{\prime}\) Rel \(Q^{\prime}\) have \(\left(P^{\prime}\left\|R^{\prime}, Q^{\prime}\right\| R^{\prime}\right) \in\) Rel \(^{\prime}\) by (rule Par)
    ultimately show ?case by blast
next
    case(cClose1 \(Q^{\prime} R^{\prime}\) a \(x\) )
    have \(Q\) Trans: \(Q \longmapsto a<x>\prec Q^{\prime}\) and RTrans: \(R \longmapsto a<\nu x>\prec R^{\prime}\) by fact +
    have \(x \sharp(P, R)\) by fact
    hence \(x\) Fresh \(P: x \sharp P\) and \(x F r e s h R: x \sharp R\) by(simp add: fresh-prod) +
    from PSimQ QTrans obtain \(P^{\prime}\) where PTrans: \(P \Longrightarrow a<x>\prec P^{\prime}\)
                        and \(P^{\prime}\) RelQ \({ }^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in\) Rel
    by (blast dest: simE)
from RTrans have \(R \Longrightarrow a<\nu x>\prec R^{\prime} \mathbf{b y}\) (rule Weak-Early-Step-Semantics.singleActionChain)
    with PTrans have Trans: \(P \| R \Longrightarrow \tau \prec<\nu x>\left(P^{\prime} \| R^{\prime}\right)\) using \(\langle x \sharp P\rangle\)
        by(rule Weak-Early-Step-Semantics.Close1)
    moreover from \(P^{\prime} \operatorname{Rel} Q^{\prime}\) have \(\left.(<\nu x\rangle\left(P^{\prime} \| R^{\prime}\right),<\nu x>\left(Q^{\prime} \| R^{\prime}\right)\right) \in \operatorname{Rel}^{\prime}\)
        by(blast intro: Par Res)
    ultimately show ?case by blast
next
    case(cClose2 \(Q^{\prime} R^{\prime}\) a \(x\) )
    have \(Q\) Trans: \(Q \longmapsto a<\nu x>\prec Q^{\prime}\) and RTrans: \(R \longmapsto a<x>\prec R^{\prime}\) by fact +
    have \(x \sharp(P, R)\) by fact
    hence \(x\) Fresh \(: x \sharp R\) and \(x F r e s h P: x \sharp P\) by(simp add: fresh-prod) +
    from PSimQ QTrans xFreshP obtain \(P^{\prime}\) where PTrans: \(P \Longrightarrow a<\nu x>\prec P^{\prime}\)
                        and \(P^{\prime}\) RelQ \(Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}\)
    by (blast dest: \(\operatorname{sim} E)\)
```

```
    from RTrans have R\Longrightarrowa<x>\prec的'by(rule Weak-Early-Step-Semantics.singleActionChain)
            with PTrans have Trans: P|R\Longrightarrow\tau\prec<\nux>( P'| R') using<x |R>
                by(rule Weak-Early-Step-Semantics.Close2)
            moreover from P'RelQ' have ( <\nux>>(\mp@subsup{P}{}{\prime}|\mp@subsup{R}{}{\prime}),<\nux>(\mp@subsup{Q}{}{\prime}|\mp@subsup{R}{}{\prime}))\inRe\mp@subsup{R}{}{\prime}
                by(blast intro: Par Res)
            ultimately show ?case by blast
        qed
    qed
qed
lemma resPres:
    fixes P :: pi
    and Q :: pi
    and Rel :: (pi }\times pi) se
    and x :: name
    and Rel'::(pi\times pi) set
    assumes PSimQ: P\rightsquigarrow«Rel» Q
    and C1:\bigwedgeRSx. (R,S) \inRel\Longrightarrow(<\nux>R,<\nux>S) \inRel'
    and RelRel':Rel \subseteqRel'
    and EqvtRel: eqvt Rel
    and EqvtRel': eqvt Rel'
    shows <\nux>P}\rightsquigarrow<<Re\mp@subsup{l}{}{\prime}»<\nux>
proof -
    from EqvtRel' show ?thesis
    proof(induct rule: simCasesCont[of - (P, x)])
        case(Bound a y Q')
        have Trans: <\nux>Q\longmapstoa<\nuy>}\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
        have }y\sharp(P,x)\mathrm{ by fact
        hence yineqx: y}\not=x\mathrm{ and yFreshP: y }#P\mathrm{ by(simp add: fresh-prod)+
        from Trans yineqx show ?case
    proof(induct rule: resCasesB)
            case(Open Q')
            have QTrans: }Q\longmapstoa[x]\prec\mp@subsup{Q}{}{\prime}\mathrm{ and aineqx: a}\not=x\mathrm{ by fact +
            from PSimQ QTrans obtain P' where PTrans: P\Longrightarrowa[x]\prec 埥
                                    and P'RelQ':( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mathrm{ Rel
            by(blast dest: simE)
    from PTrans aineqx have <\nux>P\Longrightarrowa<\nux> \prec P' by(rule Weak-Early-Step-Semantics.Open)
            hence <\nux>P\Longrightarrowa<\nuy> \prec ([(y,x)] \cdot P') using < }y\sharpP\rangle\langley\not=x
            by(force simp add: weakTransitionAlpha abs-fresh name-swap)
        moreover from EqvtRel P'RelQ' RelRel' have ([(y,x)] • P', [(y,x)] \cdot Q')\in
Rel'
            by(blast intro: eqvtRelI)
        ultimately show ?case by blast
    next
```

```
    case(Res Q')
    have QTrans: Q\longmapstoa<\nuy> \prec Q' and xineqa: }x\not=a\mathrm{ by fact+
    from PSimQ yFreshP QTrans obtain P' where PTrans: P\Longrightarrowa<\nuy>\prec 琽
                                    and \mp@subsup{P}{}{\prime}RelQ':}(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mathrm{ Rel
            by(blast dest: simE)
            from PTrans xineqa yineqx yFreshP have ResTrans: <\nux>P\Longrightarrowa<\nuy>}
(<\nux>>P')
            by(blast intro:Weak-Early-Step-Semantics.ResB)
            moreover from P'RelQ' have ((<\nux>> '}),(<\nux>\mp@subsup{Q}{}{\prime}))\inRe\mp@subsup{l}{}{\prime
            by(rule C1)
            ultimately show ?case by blast
    qed
next
    case(Free \alpha Q ')
```



```
    have \existsc::name. c\sharp(P,Q,Q', \alpha) by(blast intro: name-exists-fresh)
    then obtain c::name where cFreshQ:c\sharpQ and cFreshAlpha:c\sharp\alpha and
cFreshQ':c\sharp Q' and cFreshP:c\sharpP
            by(force simp add: fresh-prod)
    from cFreshP have <\nux>P=<\nuc>([(x,c)] • P) by(simp add: alphaRes)
    moreover have }\exists\mp@subsup{P}{}{\prime}.<\nuc>([(x,c)]\cdotP)\Longrightarrow\alpha\prec\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mp@subsup{R}{Rel}{
    proof -
        from QTrans cFreshQ have <\nuc>([(x,c)] \cdot Q)\longmapsto\alpha \prec Q' by(simp add:
alphaRes)
    moreover have c\sharp\alpha by(rule cFreshAlpha)
    moreover from PSimQ EqvtRel have ([(x,c)] • P)\rightsquigarrow<<Rel» ([(x,c)] •Q)
            by(blast intro: eqvtI)
        ultimately show ?thesis
            apply(induct rule: resCasesF, auto simp add: residual.inject pi.inject
name-abs-eq)
            by(blast intro:Weak-Early-Step-Semantics.ResF C1 dest: simE)
    qed
    ultimately show ?case by force
    qed
qed
lemma resChainI:
fixes P :: pi
and }Q :: p
and Rel ::(pi\times pi) set
and lst :: name list
assumes eqvtRel: eqvt Rel
and Res: }\RSx.(R,S)\inRel\Longrightarrow(<\nux>R,<\nux>S)\inRe
and PRelQ: P}\rightsquigarrow<<Rel» Q
shows (resChain lst) P\rightsquigarrow«Rel»(resChain lst) Q
```

```
proof -
    show ?thesis
    proof(induct lst)
        from PRelQ show resChain [] Pw«Rel» resChain [] Q by simp
    next
        fix a lst
        assume IH:(resChain lst P)}\rightsquigarrow<<Rel» (resChain lst Q
        moreover from Res have \PQa. (P,Q)\inRel\Longrightarrow(<\nua>P,<\nua>Q)\in
Rel
            by simp
            moreover have Rel \subseteqRel by simp
            ultimately have <\nua>(resChain lst P)}\rightsquigarrow«Rel»<\nua>(resChain lst Q) usin
eqvtRel
            by(rule-tac resPres)
        thus resChain (a # lst) P\rightsquigarrow«Rel» resChain (a # lst) Q
            by simp
    qed
qed
lemma bangPres:
    fixes P :: pi
    and Q :: pi
    and Rel :: (pi × pi) set
    assumes PRelQ: }(P,Q)\in\mathrm{ Rel
    and Sim: }\quad\RS.(R,S)\inRel\LongrightarrowR\rightsquigarrow«\mp@subsup{Rel}{}{\prime}>>
    and C1: Rel\subseteqRel'
    and eqvtRel: eqvt Rel'
    shows !P \rightsquigarrow«bangRel Rel`»!Q
proof -
    let ?Sim = \lambdaP Rs. ( }\forall\textrm{a}x\mp@subsup{Q}{}{\prime}.Rs=a<\nux>< \mp@subsup{Q}{}{\prime}\longrightarrowx\sharpP\longrightarrow(\exists\mp@subsup{P}{}{\prime}.
\Longrightarrow a < \nu x > \prec P ^ { \prime } \wedge ( P ^ { \prime } , Q ^ { \prime } ) \in \text { bangRel Rel'} ) ) \wedge
                                    (}\forall\alpha\mp@subsup{Q}{}{\prime}.Rs=\alpha\prec\mp@subsup{Q}{}{\prime}\longrightarrow(\exists\mp@subsup{P}{}{\prime}.P\Longrightarrow\alpha\prec\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})
bangRel Rel'))
    from eqvtRel have EqvtBangRel: eqvt(bangRel Rel') by(rule eqvtBangRel)
    from C1 have BRelRel': \P Q. (P,Q) \in bangRel Rel \Longrightarrow(P,Q) \in bangRel
Rel'
    by(auto intro: bangRelSubset)
{
    fix Pa Rs
    assume !Q\longmapstoRs and (Pa,!Q)\in bangRel Rel
    hence ?Sim Pa Rs using PRelQ
    proof(nominal-induct avoiding: Pa P rule: bangInduct)
            case(Par1B a x Q' Pa P)
            have QTrans: }Q\longmapstoa<\nux>\prec\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
            have (Pa,Q|!Q)\in bangRel Rel and x\sharpPa by fact+
```

```
thus ?Sim Pa (a<\nux>\prec\prec(\mp@subsup{Q}{}{\prime}|!Q))
```

proof (induct rule: BRParCases)
case (BRPar P R)
have PRelQ: $(P, Q) \in$ Rel by fact
have $P B R Q:(R,!Q) \in$ bangRel Rel by fact
have $x \sharp P \| R$ by fact
hence $x$ Fresh $P$ : $x \sharp P$ and $x$ Fresh $R$ : $x \sharp R$ by simp +
show ?case
proof (auto simp add: residual.inject alpha')
from PRelQ have $P \rightsquigarrow « R e l^{\prime} » Q$ by (rule Sim)
with $Q$ Trans $x$ Fresh $P$ obtain $P^{\prime}$ where PTrans: $P \Longrightarrow a<\nu x>\prec P^{\prime}$ and
$P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}^{\prime}$
by (blast dest: simE)
from PTrans xFresh $R$ have $P \| R \Longrightarrow a<\nu x>\prec\left(P^{\prime} \| R\right)$
by(force intro: Weak-Early-Step-Semantics.Par1B)
moreover from $P^{\prime} \operatorname{Rel} Q^{\prime} P B R Q$ BRelRel ${ }^{\prime}$ have $\left(P^{\prime}\left\|R, Q^{\prime}\right\|!Q\right) \in$
bangRel Rel' ${ }^{\prime}$ by (blast intro: Rel.BRPar)
ultimately show $\exists P^{\prime} . P \| R \Longrightarrow a<\nu x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime} \|!Q\right) \in$ bangRel
Rel' by blast
next
fix $y$
assume ( $y$ ::name) $\sharp Q^{\prime}$ and $y \sharp P$ and $y \sharp R$ and $y \sharp Q$
from QTrans $\left\langle y \sharp Q^{\prime}\right\rangle$ have $Q \longmapsto a<\nu y>\prec\left([(x, y)] \cdot Q^{\prime}\right)$
by (simp add: alphaBoundOutput)
moreover from PRelQ have $P \rightsquigarrow «$ Rel'» $^{\prime} Q$ by (rule Sim)
ultimately obtain $P^{\prime}$ where PTrans: $P \Longrightarrow a<\nu y>\prec P^{\prime}$ and $P^{\prime} R e l Q^{\prime}$ :
$\left(P^{\prime},[(x, y)] \cdot Q^{\prime}\right) \in R e l^{\prime}$
using $\langle y \sharp P\rangle$
by (blast dest: simE)
from PTrans $\langle y \sharp R\rangle$ have $P \| R \Longrightarrow a<\nu y>\prec\left(P^{\prime} \| R\right)$ by (force intro:
Weak-Early-Step-Semantics.Par1B)
moreover from $P^{\prime} \operatorname{Rel} Q^{\prime} P B R Q$ BRelRel' have $\left(P^{\prime}\left\|R,\left([(x, y)] \cdot Q^{\prime}\right)\right\|\right.$
$!Q) \in$ bangRel Rel' $\mathbf{b y}($ metis Rel.BRPar)
with $\langle x \sharp Q\rangle\langle y \sharp Q\rangle$ have $\left(P^{\prime}\left\|R,\left([(y, x)] \cdot Q^{\prime}\right)\right\|!([(y, x)] \cdot Q)\right) \in$
bangRel Rel'
by (simp add: name-fresh-fresh name-swap)
ultimately show $\exists P^{\prime} . P \| R \Longrightarrow a<\nu y>\prec P^{\prime} \wedge\left(P^{\prime},\left([(y, x)] \cdot Q^{\prime}\right) \|\right.$
$!([(y, x)] \cdot Q)) \in$ bangRel Rel ${ }^{\prime}$
by blast
qed
qed
next
case(Par1F $\alpha Q^{\prime} P a P$ )
have $Q$ Trans: $Q \longmapsto \alpha \prec Q^{\prime}$ by fact
have $(P a, Q \|!Q) \in$ bangRel Rel by fact
thus ?case
proof (induct rule: BRParCases)

```
    case(BRPar P R)
    have PRelQ:}(P,Q)\in\mathrm{ Rel and BR: (R,!Q) G bangRel Rel by fact+
    show ?case
    proof(auto simp add: residual.inject)
    from PRelQ have P}\rightsquigarrow«<Rel'»Q by(rule Sim
            with QTrans obtain P' where PTrans: P\Longrightarrow\alpha \prec P' and RRel:( }\mp@subsup{P}{}{\prime}\mathrm{ ,
Q')\inRel'
            by(blast dest: simE)
        from PTrans have P|R\Longrightarrow\alpha\prec\mp@subsup{P}{}{\prime}|R\mathrm{ by(rule Weak-Early-Step-Semantics.Par1F)}
            moreover from RRel BR BRelRel' have ( }\mp@subsup{P}{}{\prime}|R,\mp@subsup{Q}{}{\prime}|!Q)\in\mathrm{ bangRel
Rel' by(metis Rel.BRPar)
            ultimately show }\exists\mp@subsup{P}{}{\prime}.P|R\Longrightarrow\alpha\prec\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime}|!Q)\in\mp@subsup{b}{}{\prime}|\mp@code{RRel Rel}\mp@subsup{}{}{\prime
by blast
            qed
        qed
    next
        case(Par2B a x Q ' Pa P)
            hence IH: \Pa. (Pa,!Q) \inbangRel Rel \Longrightarrow?Sim Pa (a<\nux>}\prec\mp@subsup{Q}{}{\prime})\mathrm{ by
simp
    have (Pa,Q|!Q)\in bangRel Rel and x\sharpPa by fact+
    thus ?Sim Pa (a<\nux> \prec(Q| Q'))
    proof(induct rule: BRParCases)
            case(BRPar P R)
            have PRelQ:}(P,Q)\in\mathrm{ Rel and RBRQ: (R,!Q) E bangRel Rel by fact+
            have }x\sharpP||R\mathrm{ by fact
            hence xFreshP: }x\sharpP\mathrm{ and xFreshR: }x\sharpR\mathrm{ by simp +
            from EqvtBangRel show ?Sim (P|R) (a<\nux>}\prec(Q|\mp@subsup{Q}{}{\prime})
            proof(auto simp add: residual.inject alpha')
                from RBRQ have ?Sim R (a<\nux>}\prec\mp@subsup{Q}{}{\prime})\mathbf{by}(\mathrm{ rule IH)
            with xFreshR obtain }\mp@subsup{R}{}{\prime}\mathrm{ where RTrans: R בa<vx>}\prec\mp@subsup{R}{}{\prime}\mathrm{ and }\mp@subsup{R}{}{\prime}BR\mp@subsup{Q}{}{\prime}
(R', Q') \in(bangRel Rel')
                by(metis simE)
            from RTrans xFreshP have P|R\Longrightarrowa<\nux> \prec (P| R') by(auto intro:
Weak-Early-Step-Semantics.Par2B)
                moreover from PRelQ R'BRQ' C1 have ( }P|\mp@subsup{R}{}{\prime},Q|\mp@subsup{Q}{}{\prime})\in(\mathrm{ bangRel
Rel') by(blast dest: Rel.BRPar)
            ultimately show }\exists\mp@subsup{P}{}{\prime}.P|R\Longrightarrowa<\nux> \prec P'^( (P',Q| Q ) \in bangRe
Rel' by blast
            next
                fix }
                assume (y::name) #Q and y\sharp Q' and y\sharpP and y#R
                from RBRQ have ?Sim R (a<\nux>}\prec\mp@subsup{Q}{}{\prime})\boldsymbol{by}(\mathrm{ rule IH)
                with \langley\sharp Q'\rangle have ?Sim R (a<\nuy> \prec ([(x,y)] • Q')) by(simp add:
alphaBoundOutput)
            with }\langley\sharpR\rangle\mathrm{ obtain }\mp@subsup{R}{}{\prime}\mathrm{ where RTrans: }R\Longrightarrowa<\nuy> \prec R' and R'BRQ'
                (R',}([(x,y)]\cdot\mp@subsup{Q}{}{\prime}))\in(bangRel Rel'
            by(metis simE)
```

from RTrans $\langle y \sharp P\rangle$ have $P \| R \Longrightarrow a<\nu y>\prec\left(P \| R^{\prime}\right)$ by (auto intro: Weak-Early-Step-Semantics.Par2B)
moreover from PRelQ $R^{\prime} B R Q^{\prime} C 1$ have $\left(P\left\|R^{\prime}, Q\right\|\left([(x, y)] \cdot Q^{\prime}\right)\right) \in$ (bangRel Rel') by (blast dest: Rel.BRPar)
with $\langle y \sharp Q\rangle\langle x \sharp Q\rangle$ have $\left(P\left\|R^{\prime},([(y, x)] \cdot Q)\right\|\left([(y, x)] \cdot Q^{\prime}\right)\right) \in$ (bangRel Rel')
by (simp add: name-swap name-fresh-fresh)
ultimately show $\exists P^{\prime} . P \| R \Longrightarrow a<\nu y>\prec P^{\prime} \wedge\left(P^{\prime},([(y, x)] \cdot Q) \|\right.$
$\left.\left([(y, x)] \cdot Q^{\prime}\right)\right) \in$ bangRel Rel ${ }^{\prime}$ by blast
qed
qed
next
case(Par2F $\alpha Q^{\prime}$ Pa P)
hence $I H: \wedge P a .(P a,!Q) \in$ bangRel Rel $\Longrightarrow$ ?Sim Pa $\left(\alpha \prec Q^{\prime}\right)$ by simp
have $(P a, Q \|!Q) \in$ bangRel Rel by fact
thus ? case
proof (induct rule: BRParCases)
case (BRPar P R)
have PRelQ: $(P, Q) \in$ Rel and $R B R Q:(R,!Q) \in$ bangRel Rel by fact + show ?case
proof (auto simp add: residual.inject)
from $R B R Q$ IH have $\exists R^{\prime} . R \Longrightarrow \alpha \prec R^{\prime} \wedge\left(R^{\prime}, Q^{\prime}\right) \in$ bangRel Rel ${ }^{\prime}$ by (metis simE)
then obtain $R^{\prime}$ where $R$ Trans: $R \Longrightarrow \alpha \prec R^{\prime}$ and $R^{\prime} \operatorname{Rel} Q^{\prime}:\left(R^{\prime}, Q^{\prime}\right) \in$ bangRel Rel' by blast
from RTrans have $P\|R \Longrightarrow \alpha \prec P\| R^{\prime}$ by (rule Weak-Early-Step-Semantics.Par2F) moreover from PRelQ $R^{\prime} \operatorname{Rel} Q^{\prime} C 1$ have $\left(P\left\|R^{\prime}, Q\right\| Q^{\prime}\right) \in$ bangRel
Rel' ${ }^{\prime} \mathbf{b y}$ (blast dest: Rel.BRPar)
ultimately show $\exists P^{\prime} . P \| R \Longrightarrow \alpha \prec P^{\prime} \wedge\left(P^{\prime}, Q \| Q^{\prime}\right) \in$ bangRel Rel ${ }^{\prime}$
by blast
qed
qed
next
case (Comm1 a $Q^{\prime}$ b $Q^{\prime \prime}$ Pa $P$ )
hence $I H: \bigwedge P a$. $(P a,!Q) \in$ bangRel Rel $\Longrightarrow$ ?Sim $P a\left(a[b] \prec Q^{\prime \prime}\right)$ by simp
have $Q$ Trans: $Q \longmapsto a<b>\prec Q^{\prime}$ by fact
have $(P a, Q \|!Q) \in$ bangRel Rel by fact
thus ?case
proof (induct rule: BRParCases)
case (BRPar P R)
have PRelQ: $(P, Q) \in$ Rel and $R B R Q:(R,!Q) \in$ bangRel Rel by fact + show ? case
proof (auto simp add: residual.inject)
from PRelQ have $P \rightsquigarrow «$ Rel' $^{\prime} » Q$ by (rule Sim)
with $Q$ Trans obtain $P^{\prime}$ where PTrans: $P \Longrightarrow a<b>\prec P^{\prime}$ and $P^{\prime} R e l Q^{\prime}$ : $\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}^{\prime}$ by (blast dest: simE)
from $I H R B R Q$ have RTrans: $\exists R^{\prime} . R \Longrightarrow a[b] \prec R^{\prime} \wedge\left(R^{\prime}, Q^{\prime \prime}\right) \in$ bangRel $R e l^{\prime}$
by (metis simE)
then obtain $R^{\prime}$ where RTrans: $R \Longrightarrow a[b] \prec R^{\prime}$ and $R^{\prime} \operatorname{Rel} Q^{\prime \prime}:\left(R^{\prime}, Q^{\prime \prime}\right)$ $\in$ bangRel Rel ${ }^{\prime}$
by blast
from PTrans RTrans have $P\left\|R \Longrightarrow \tau \prec P^{\prime}\right\| R^{\prime}$ by (rule Weak-Early-Step-Semantics.Comm1)
moreover from $P^{\prime} \operatorname{Rel} Q^{\prime} R^{\prime} \operatorname{Rel} Q^{\prime \prime}$ have $\left(P^{\prime}\left\|R^{\prime}, Q^{\prime}\right\| Q^{\prime \prime}\right) \in$ bangRel
Rel' ${ }^{\prime}$ by(rule Rel.BRPar)
ultimately show $\exists P^{\prime} . P \| R \Longrightarrow \tau \prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime} \| Q^{\prime \prime}\right) \in$ bangRel Rel ${ }^{\prime}$ by blast
qed qed
next

```
        case(Comm2 a b Q' Q')
```

    hence \(I H: \bigwedge P a .(P a,!Q) \in\) bangRel Rel \(\Longrightarrow\) ?Sim \(P a\left(a<b>\prec Q^{\prime \prime}\right)\) by simp
    have \(Q\) Trans: \(Q \longmapsto a[b] \prec Q^{\prime}\) by fact
    have \((P a, Q \|!Q) \in\) bangRel Rel by fact
    thus ?case
    proof (induct rule: BRParCases)
            case(BRPar PR)
            have PRelQ: \((P, Q) \in\) Rel and \(R B R Q:(R,!Q) \in\) bangRel Rel by fact+
            show ?case
            proof (auto simp add: residual.inject)
            from PRelQ have \(P \rightsquigarrow\left\langle\right.\) Rel \(^{\prime} » Q\) by (rule Sim)
                with QTrans obtain \(P^{\prime}\) where PTrans: \(P \Longrightarrow a[b] \prec P^{\prime}\) and \(P^{\prime} R e l Q^{\prime}\) :
    $\left(P^{\prime}, Q^{\prime}\right) \in$ Rel $^{\prime}$
$\mathbf{b y}($ blast dest: $\operatorname{sim} E)$
from $I H R B R Q$ have $R$ Trans: $\exists R^{\prime} . R \Longrightarrow a<b>\prec R^{\prime} \wedge\left(R^{\prime}, Q^{\prime \prime}\right) \in$
bangRel Rel'
by (metis simE)
then obtain $R^{\prime}$ where RTrans: $R \Longrightarrow a<b>\prec R^{\prime}$ and $R^{\prime} \operatorname{Rel} Q^{\prime \prime}:\left(R^{\prime}\right.$,
$\left.Q^{\prime \prime}\right) \in$ bangRel Rel ${ }^{\prime}$
by blast
from PTrans RTrans have $P\left\|R \Longrightarrow \tau \prec P^{\prime}\right\| R^{\prime}$ by (rule Weak-Early-Step-Semantics.Comm2)
moreover from $P^{\prime} \operatorname{Rel} Q^{\prime} R^{\prime} \operatorname{Rel} Q^{\prime \prime}$ have $\left(P^{\prime}\left\|R^{\prime}, Q^{\prime}\right\| Q^{\prime \prime}\right) \in$ bangRel
Rel' by(rule Rel.BRPar)
ultimately show $\exists P^{\prime} . P \| R \Longrightarrow \tau \prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime} \| Q^{\prime \prime}\right) \in$ bangRel Rel ${ }^{\prime}$
by blast
qed
qed
next
case(Close1 ax $Q^{\prime} Q^{\prime \prime}$ Pa P)
hence $I H: \wedge P a .(P a,!Q) \in$ bangRel Rel $\longrightarrow$ ?Sim $P a\left(a<\nu x>\prec Q^{\prime \prime}\right)$ by
simp

```
    have QTrans: Q\longmapstoa<x>}\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    have xFreshQ: x\sharpQ by fact
    have (Pa,Q|!Q)\inbangRel Rel by fact
    moreover have xFreshPa: x\sharpPa}\mathrm{ by fact
    ultimately show ?case
    proof(induct rule: BRParCases)
        case(BRPar P R)
        have PRelQ:}(P,Q)\in\mathrm{ Rel and RBRQ: (R,!Q) E bangRel Rel by fact+
    have }x\sharpP|R\mathrm{ by fact
    hence xFreshP: x\sharpP and xFreshR: x\sharpR by simp+
    show ?case
    proof(auto simp add: residual.inject)
            from PRelQ have P}\rightsquigarrow«<Rel'»Q by(rule Sim
            with QTrans xFreshP obtain }\mp@subsup{P}{}{\prime}\mathrm{ where PTrans: P }\Longrightarrowa<x>\prec\mp@subsup{P}{}{\prime}\mathrm{ and
P'RelQ':( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mp@subsup{R}{}{\prime
            by(blast dest: simE)
            from RBRQ xFreshR IH have \exists}\mp@subsup{R}{}{\prime}.R\Longrightarrowa<\nux>\prec 质^( R', Q'')
bangRel Rel'
                by(metis simE)
            then obtain }\mp@subsup{R}{}{\prime}\mathrm{ where RTrans: R Ca<vx>}\prec\mp@subsup{R}{}{\prime}\mathrm{ and }\mp@subsup{R}{}{\prime}RelQ'\prime:( R'
Q')}\in\mathrm{ bangRel Rel'
                by blast
            from PTrans RTrans xFreshP have P|R\Longrightarrow\tau\prec<\nux>( (P'| R')
                by(rule Weak-Early-Step-Semantics.Close1)
```



```
Q'}))\in\mathrm{ bangRel Rel'
                    by(force intro: Rel.BRPar BRRes)
                    ultimately show }\exists\mp@subsup{P}{}{\prime}.P|R\Longrightarrow\tau\prec\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},<\nux>(\mp@subsup{Q}{}{\prime}|\mp@subsup{Q}{}{\prime\prime}))
bangRel Rel' by blast
            qed
    qed
    next
        case(Close2 a x Q' Q' Pa P)
    hence IH: \bigwedgePa. (Pa,!Q) \in bangRel Rel \Longrightarrow?SimPa (a<x>\prec \ '') by simp
    have QTrans: }Q\longmapstoa<\nux>\prec\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    have xFreshQ: x\sharpQ by fact
    have (Pa,Q|!Q)\in bangRel Rel and x\sharpPa by fact+
    thus ?case
    proof(induct rule: BRParCases)
        case(BRPar P R)
        have PRelQ: (P,Q)\inRel and RBRQ: (R,!Q) \in bangRel Rel by fact+
        have }x\sharpP||R\mathrm{ by fact
        hence xFreshP: }x\sharpP\mathrm{ and xFreshR: }x\sharpR\mathrm{ by simp +
        show ?case
        proof(auto simp add: residual.inject)
            from PRelQ have P}\rightsquigarrow<<Rel'»Q by(rule Sim
            with QTrans xFreshP obtain P' where PTrans: P\Longrightarrowa<\nux>}\prec\mp@subsup{P}{}{\prime}\mathrm{ and
```

```
P'RelQ':( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mp@subsup{R}{}{\prime}\mp@subsup{e}{}{\prime
            by(blast dest: simE)
            from RBRQ IH have }\exists\mp@subsup{R}{}{\prime}.R\Longrightarrowa<x>\prec R R'^( R', Q'')\in bangRel Rel'
                    by auto
            then obtain }\mp@subsup{R}{}{\prime}\mathrm{ where RTrans: R Ca<x> < R' and R'RelQ': ( }\mp@subsup{R}{}{\prime}\mathrm{ ,
Q')}\in\mathrm{ bangRel Rel'
            by blast
            from PTrans RTrans xFreshR have P|R\Longrightarrow\tau\prec<\nux>( P'| R')
                by(rule Weak-Early-Step-Semantics.Close2)
                    moreover from P'RelQ' R'RelQ"| have (<\nux> ( }\mp@subsup{P}{}{\prime}|\mp@subsup{R}{}{\prime}),<\nux>(\mp@subsup{Q}{}{\prime}
Q'}))\in\mathrm{ bangRel Rel'
                            by(force intro: Rel.BRPar BRRes)
                            ultimately show }\exists\mp@subsup{P}{}{\prime}.P|R\Longrightarrow\tau\prec\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},<\nux>(\mp@subsup{Q}{}{\prime}|\mp@subsup{Q}{}{\prime\prime}))
bangRel Rel' by blast
            qed
            qed
        next
            case(Bang Rs Pa P)
            hence IH: \Pa. (Pa,Q|!Q)\in bangRel Rel \Longrightarrow ?Sim Pa Rs by simp
            have (Pa,!Q) \in bangRel Rel by fact
            thus ?case
            proof(induct rule: BRBangCases)
                case(BRBang P)
                have PRelQ: (P,Q)\in Rel by fact
                    hence (!P,!Q)\in bangRel Rel by(rule Rel.BRBang)
            with PRelQ have (P|!P,Q|!Q) \in bangRel Rel by(rule BRPar)
            with IH have ?Sim (P|!P) Rs by simp
            thus ?case by(force intro:Weak-Early-Step-Semantics.Bang)
            qed
        qed
    }
    moreover from PRelQ have (!P,!Q) \in bangRel Rel by(rule BRBang)
    ultimately show ?thesis by(auto simp add: weakStepSimulation-def)
qed
end
theory Weak-Early-Sim-Pres
    imports Weak-Early-Sim
begin
lemma tauPres:
    fixes P :: pi
    and Q :: pi
    and Rel :: (pi }\times pi) se
    and Rel'::(pi\times pi) set
```

```
    assumes PRel \(Q:(P, Q) \in \operatorname{Rel}\)
    shows \(\tau .(P) \rightsquigarrow<\) Rel \(>\tau .(Q)\)
proof(induct rule: simCases)
    case(Bound \(Q^{\prime}\) a \(x\) )
    have \(\tau .(Q) \longmapsto a<\nu x>\prec Q^{\prime}\) by fact
    hence False by (induct rule: tauCases', auto)
    thus? case by simp
next
    case (Free \(Q^{\prime} \alpha\) )
    have \(\tau .(Q) \longmapsto\left(\alpha \prec Q^{\prime}\right)\) by fact
    thus? case
    proof \((\) induct rule: tauCases', auto simp only: pi.inject residual.inject)
    have \(\tau .(P) \Longrightarrow \approx \prec P\) by (rule Tau)
    with PRelQ show \(\exists P^{\prime} . \tau .(P) \Longrightarrow \tau \prec P^{\prime} \wedge\left(P^{\prime}, Q\right) \in\) Rel by blast
    qed
qed
lemma inputPres:
    fixes \(P \quad:: p i\)
    and \(x\) :: name
    and \(\quad Q \quad:: p i\)
    and \(a\) :: name
    and Rel :: \((p i \times p i)\) set
    assumes PRelQ: \(\forall y .(P[x::=y], Q[x::=y]) \in \operatorname{Rel}\)
    and Eqvt: eqvt Rel
    shows \(a<x>. P \rightsquigarrow<\) Rel \(>a<x>. Q\)
using Eqvt
proof \((\) induct rule: simCasesCont \([\) where \(C=(x, a, P, Q)])\)
    case(Bound by \(Q^{\prime}\) )
    from \(\langle y \sharp(x, a, P, Q)\) have \(y \neq x y \neq a y \sharp P y \sharp Q\) by simp+
    from \(\left.\langle a<x\rangle . Q \longmapsto b<\nu y\rangle \prec Q^{\prime}\right\rangle\langle y \neq a\rangle\langle y \neq x\rangle\langle y \sharp Q\rangle\) show ?case
    by (erule-tac inputCases') auto
next
    case (Free \(\alpha Q^{\prime}\) )
    from \(\left\langle a<x>. Q \longmapsto \alpha \prec Q^{\prime}\right\rangle\)
    show ?case
    proof (induct rule: inputCases)
    case (cInput u)
    have \(a<x>. P \Longrightarrow \wedge(a<u>) \prec P[x::=u]\)
        by (rule Input)
    moreover from \(\operatorname{PRelQ}\) have \((P[x::=u], Q[x::=u]) \in\) Rel by auto
    ultimately show? case by blast
    qed
qed
```

```
lemma outputPres:
    fixes P :: pi
    and Q :: pi
    and a :: name
    and b :: name
    and Rel :: (pi }\times pi) se
    assumes PRelQ: (P,Q)\inRel
    shows a{b}.P\rightsquigarrow<Rel> a{b}.Q
proof(induct rule: simCases)
    case(Bound Q' c x)
    have }a{b}.Q\longmapstoc<\nux>\prec\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    hence False by(induct rule: outputCases', auto)
    thus ?case by simp
next
    case(Free Q' \alpha)
    have }a{b}.Q\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    thus \exists 竹.a{b}.P\Longrightarrow < \prec P'^( 
    proof(induct rule:outputCases', auto simp add: pi.inject residual.inject)
    have }a{b}.P\Longrightarrowa[b]\precP\mathbf{by}(rule Output
    with PRelQ show }\exists\mp@subsup{P}{}{\prime}.a{b}.P\Longrightarrowa[b]\prec\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},Q)\in\mathrm{ Rel by blast
    qed
qed
lemma matchPres:
    fixes P :: pi
    and Q :: pi
    and a :: name
    and b :: name
    and Rel :: (pi\times pi) set
    and Rel'::(pi\times pi) set
    assumes PSimQ: P\rightsquigarrow<Rel>}
    and RelRel':Rel }\subseteq\mp@subsup{REl}{}{\prime
    and RelStay: \RSc. (R,S)\inRel \Longrightarrow([c\frownc]R,S)\inRel
    shows [a\frownb]P\rightsquigarrow<\mp@subsup{Rel}{}{\prime}>[a\frownb]Q
proof(induct rule: simCases)
    case(Bound Q' cx)
    have }x\sharp[a\frownb]P\mathrm{ by fact
    hence xFreshP:(x::name) \sharpP by simp
    have [a\frownb]Q\longmapstoc<\nux>}\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    thus ?case
    proof(induct rule: matchCases)
    case Match
    have }Q\longmapstoc<\nux><<\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    with PSimQ xFreshP obtain P' where PTrans: P\Longrightarrowc<\nux>}\prec\mp@subsup{P}{}{\prime
                                    and P'RelQ':}(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\operatorname{Rel
```

```
        by(blast dest: simE)
    from PTrans have [a\frowna]P\Longrightarrowc<\nux>}\prec\mp@subsup{P}{}{\prime}\mathbf{by}(\mathrm{ rule Weak-Early-Step-Semantics.Match)
        moreover from P'RelQ' RelRel' have ( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\inRel' by blas
        ultimately show ?case by blast
    qed
next
    case(Free Q' \alpha)
    have [a\frownb]Q\longmapsto\alpha\prec Q' by fact
    thus ?case
    proof(induct rule: matchCases)
        case Match
        have }Q\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
        with PSimQ obtain P}\mp@subsup{P}{}{\prime}\mathrm{ where P\人 < < P' and ( }\mp@subsup{P}{}{\prime},Q)\in\mathrm{ Rel
        by(blast dest: simE)
    thus ?case
    proof(induct rule: transitionCases)
        case Step
        have }P\Longrightarrow\alpha\prec\mp@subsup{P}{}{\prime}\mathrm{ by fact
        hence [a\frowna]P\Longrightarrow\alpha\prec琽卉y(rule Weak-Early-Step-Semantics.Match)
        with RelRel' }\mp@subsup{}{(}{\prime}(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mathrm{ Rel> show ?case by(force simp add: weakFreeTran-
sition-def)
    next
        case Stay
        have [a\frowna]P\Longrightarrow^ }\tau\prec[a\frowna]P by(simp add: weakFreeTransition-def)
            moreover from < (P, Q') \in Rel> have ([a\frowna]P, Q') \in Rel by(blast intro:
RelStay)
        ultimately show ?case using RelRel' by blast
        qed
    qed
qed
lemma mismatchPres:
    fixes P :: pi
    and }Q :: p
    and a :: name
    and b :: name
    and Rel ::(pi\times pi) set
    and Rel'::(pi\times pi) set
    assumes PSimQ: P}\rightsquigarrow<\mathrm{ Rel }>
    and RelRel':Rel }\subseteqRel'
    and RelStay: \R S c d. \llbracket(R,S) \inRel; c\not=d\rrbracket\Longrightarrow([c\not=d]R,S)\inRel
    shows [a\not=b]P\rightsquigarrow<\mp@subsup{Rel}{}{\prime}>[a\not=b]Q
proof(induct rule: simCases)
    case(Bound Q' c x)
    have }x\sharp[a\not=b]P\mathrm{ by fact
    hence xFreshP: (x::name) \sharpP by simp
    have [a\not=b]Q\longmapstoc<\nux>\prec\prec Q' by fact
```

```
thus ?case
proof(induct rule: mismatchCases)
    case Mismatch
    have aineqb: }a\not=b\mathrm{ by fact
    have }Q\longmapstoc<\nux>\prec\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    with PSimQ xFreshP obtain P' where PTrans: P\Longrightarrowc<\nux>}\prec\mp@subsup{P}{}{\prime
                                    and P'RelQ':( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mathrm{ Rel
    by(blast dest: simE)
    from PTrans aineqb have [a\not=b]P\Longrightarrowc<\nux>}\prec\mp@subsup{P}{}{\prime}\mathrm{ by(rule Weak-Early-Step-Semantics.Mismatch)
    moreover from P'RelQ' RelRel' have ( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mp@subsup{R}{}{\prime}\mp@subsup{R}{}{\prime}\mp@subsup{l}{}{\prime}\mathrm{ by blast
    ultimately show ?case by blast
qed
next
    case(Free Q' \alpha)
    have [a\not=b]Q\longmapsto\alpha\prec Q' by fact
    thus ?case
    proof(induct rule: mismatchCases)
    case Mismatch
    have aineqb: }a\not=b\mathrm{ by fact
    have}Q\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    with PSimQ obtain P' where P\Longrightarrow^\alpha\prec P' and ( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\operatorname{Rel
        by(blast dest: simE)
    thus ?case
    proof(induct rule: transitionCases)
        case Step
        have P\Longrightarrow\alpha\prec P' by fact
    hence [a\not=b]P\Longrightarrow\alpha\prec P' using aineqb by(rule Weak-Early-Step-Semantics.Mismatch)
        with RelRel' }\langle(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mathrm{ Rel> show ?case by(force simp add: weakFreeTran-
sition-def)
    next
        case Stay
        have [a\not=b]P\Longrightarrow^\tau\prec[a\not=b]P by(simp add: weakFreeTransition-def)
            moreover from < (P, Q') \in Rel> aineqb have ([a\not=b]P, Q') \in Rel by(blast
intro: RelStay)
            ultimately show ?case using RelRel' by blast
    qed
    qed
qed
lemma parCompose:
fixes P :: pi
and }Q\quad::p
and }R\quad::p
and S :: pi
and Rel :: (pi\timespi) set
and Rel' :: (pi\timespi) set
and Rel" :: (pi\timespi) set
assumes PSimQ: }P\rightsquigarrow<\mathrm{ Rel }>
```

```
    and RSimT: R}<<<\mp@subsup{Rel}{}{\prime}>
    and PRelQ: }(P,Q)\inRe
    and RRel'T: (R,S)\inRel'
    and Par: }\\mp@subsup{P}{}{\prime}\mp@subsup{Q}{}{\prime}\mp@subsup{R}{}{\prime}\mp@subsup{S}{}{\prime}.\llbracket(\mp@subsup{P}{}{\prime},Q')\in\operatorname{Rel};(\mp@subsup{R}{}{\prime},\mp@subsup{S}{}{\prime})\in\operatorname{Rel}\rrbracket\Longrightarrow(\mp@subsup{P}{}{\prime}|\mp@subsup{R}{}{\prime}\mathrm{ ,
Q'| S') \inRel'|
    and Res: \}\quad\TUx.(T,U)\in\mp@subsup{R}{el}{\prime\prime}\Longrightarrow(<\nux>T,<\nux>U)\inRe\mp@subsup{l}{}{\prime\prime
    shows P|R}\rightsquigarrow<\mp@subsup{\mathrm{ Rel }}{}{\prime\prime}>Q|
proof -
    show ?thesis
    proof(induct rule: simCases)
        case(Bound Q' a x)
    have }x\sharpP|R\mathrm{ by fact
    hence xFreshP: }x\sharpP\mathrm{ and xFreshR: }x\sharpR\mathrm{ by simp+
    have }Q|S\longmapstoa<\nux><\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    thus ?case
    proof(induct rule: parCasesB)
            case(cPar1 Q')
            have QTrans: Q\longmapstoa<\nux>}\prec\mp@subsup{Q}{}{\prime}\mathrm{ and xFreshT: x甘S by fact+
            from xFreshP PSimQ QTrans obtain P' where PTrans:P\Longrightarrowa<\nux>}\prec<\mp@subsup{P}{}{\prime
                                    and }\mp@subsup{P}{}{\prime}\mathrm{ RelQ':}(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\operatorname{Rel
            by(blast dest: simE)
    from PTrans xFreshR have P|R\Longrightarrowa<\nux>\prec (P'|R) by(rule Weak-Early-Step-Semantics.Par1B)
    moreover from P'RelQ' RRel'T have ( }\mp@subsup{P}{}{\prime}|R,\mp@subsup{Q}{}{\prime}|S)\inRel '/ by(rule Par
        ultimately show ?case by blast
    next
        case(cPar2 S')
        have STrans: S\longmapstoa<\nux>}\prec\mp@subsup{S}{}{\prime}\mathrm{ and xFreshQ:x #Q by fact+
        from xFreshR RSimT STrans obtain R' where RTrans:R\Longrightarrowa<\nux>}\prec\mp@subsup{R}{}{\prime
                        and R'Rel'T':( R', S') \in Rel'
            by(blast dest: simE)
    from RTrans xFreshP xFreshR have ParTrans: P|R\Longrightarrowa<\nux>\prec (P| |
            by(blast intro:Weak-Early-Step-Semantics.Par2B)
        moreover from PRelQ R'Rel'}\mp@subsup{T}{}{\prime}\mathrm{ have ( }P||\mp@subsup{R}{}{\prime},Q| | S')\inRel" by(rul
Par)
            ultimately show ?case by blast
        qed
    next
        case(Free QT' \alpha)
    have }Q|S\longmapsto\alpha\precQT' by fac
    thus ?case
    proof(induct rule: parCasesF[of - - - (P,R)])
        case(cPar1 Q')
        have }Q\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
        with PSimQ obtain P' where PTrans: P\Longrightarrow^ \alpha \prec P' and PRel: ( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime}
Rel
            by(blast dest: simE)
    from PTrans have Trans: P|R\Longrightarrow^ \alpha\prec P'|R by(rule Weak-Early-Semantics.Par1F)
            moreover from PRel RRel'T have ( }\mp@subsup{P}{}{\prime}|R,\mp@subsup{Q}{}{\prime}|S)\inRe\mp@subsup{l}{}{\prime\prime}\mathrm{ by(blast intro:
```


## Par)

ultimately show ?case by blast
next
case(cPar2 $\left.S^{\prime}\right)$
have $S \longmapsto \alpha \prec S^{\prime}$ by fact
with $R \operatorname{SimT}$ obtain $R^{\prime}$ where RTrans: $R \Longrightarrow \alpha \prec R^{\prime}$ and RRel: $\left(R^{\prime}, S^{\prime}\right)$
$\in R e l^{\prime}$
by (blast dest: $\operatorname{sim} E)$
from RTrans have Trans: $P\|R \Longrightarrow \wedge \prec P\| R^{\prime}$ by(rule Weak-Early-Semantics.Par2F)
moreover from PRelQ RRel have $\left(P\left\|R^{\prime}, Q\right\| S^{\prime}\right) \in$ Rel $^{\prime \prime}$ by (blast intro:
Par)
ultimately show ?case by blast
next
case (cComm1 $\left.Q^{\prime} S^{\prime} a b\right)$
have $Q$ Trans: $Q \longmapsto a<b>\prec Q^{\prime}$ and $S$ Trans: $S \longmapsto a[b] \prec S^{\prime}$ by fact +
from PSimQ QTrans obtain $P^{\prime}$ where PTrans: $P \Longrightarrow a<b>\prec P^{\prime}$
and $P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}$
by (fastforce dest: simE simp add: weakFreeTransition-def)
from $R$ SimT STrans obtain $R^{\prime}$ where $R$ Trans: $R \Longrightarrow a[b] \prec R^{\prime}$
and RRel: $\left(R^{\prime}, S^{\prime}\right) \in R e l^{\prime}$
by(fastforce dest: simE simp add: weakFreeTransition-def)
from PTrans RTrans have $P\left\|R \Longrightarrow \tau \prec P^{\prime}\right\| R^{\prime}$ by(rule Weak-Early-Step-Semantics.Comm1)
hence $P\left\|R \Longrightarrow \wedge \prec P^{\prime}\right\| R^{\prime}$
by (auto simp add: trancl-into-rtrancl dest: Weak-Early-Step-Semantics.tauTransitionChain)
moreover from $P^{\prime}$ Rel $Q^{\prime}$ RRel have $\left(P^{\prime}\left\|R^{\prime}, Q^{\prime}\right\| S^{\prime}\right) \in$ Rel ${ }^{\prime \prime}$ by (rule Par)
ultimately show ?case by blast
next
case $\left(c\right.$ Comm2 $\left.Q^{\prime} S^{\prime} a b\right)$
have $Q$ Trans: $Q \longmapsto a[b] \prec Q^{\prime}$ and STrans: $S \longmapsto a<b>\prec S^{\prime}$ by fact +
from PSimQ QTrans obtain $P^{\prime}$ where PTrans: $P \Longrightarrow a[b] \prec P^{\prime}$
and PRel: $\left(P^{\prime}, Q^{\prime}\right) \in$ Rel
by(fastforce dest: simE simp add: weakFreeTransition-def)
from RSimT STrans obtain $R^{\prime}$ where $R$ Trans: $R \Longrightarrow a<b>\prec R^{\prime}$
and $R^{\prime}$ Rel $^{\prime} T^{\prime}:\left(R^{\prime}, S^{\prime}\right) \in$ Rel $^{\prime}$
by(fastforce dest: simE simp add: weakFreeTransition-def)
from PTrans RTrans have $P\left\|R \Longrightarrow \tau \prec P^{\prime}\right\| R^{\prime}$ by(rule Weak-Early-Step-Semantics.Comm2)
hence $P\left\|R \Longrightarrow \imath \prec P^{\prime}\right\| R^{\prime}$
by (auto simp add: trancl-into-rtrancl dest: Weak-Early-Step-Semantics.tauTransitionChain)
moreover from PRel $R^{\prime}$ Rel $^{\prime} T^{\prime}$ have $\left(P^{\prime}\left\|R^{\prime}, Q^{\prime}\right\| S^{\prime}\right) \in$ Rel ${ }^{\prime \prime}$ by (rule Par)
ultimately show ? case by blast
next
case $\left(c\right.$ Close1 $\left.Q^{\prime} S^{\prime} a x\right)$

```
    have QTrans: }Q\longmapstoa<x>\prec\mp@subsup{Q}{}{\prime}\mathrm{ and STrans: }S\longmapstoa<\nux>\prec\mp@subsup{S}{}{\prime}\mathrm{ by fact+
    have }x\sharp(P,R)\mathrm{ by fact
    hence xFreshP: x\sharpP and xFreshR: x }\sharpR\mathbf{by}(simp add: fresh-prod)
    from PSimQ QTrans obtain P' where PTrans: P\Longrightarrowa<x>}\prec\mp@subsup{P}{}{\prime
                            and \mp@subsup{P}{}{\prime}RelQ}\mp@subsup{Q}{}{\prime}:(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\operatorname{Rel
    by(fastforce dest: simE simp add: weakFreeTransition-def)
    from RSimT STrans xFreshR obtain R' where RTrans: R\Longrightarrowa<\nux>}\prec\mp@subsup{R}{}{\prime
                and R'Rel'T':}(\mp@subsup{R}{}{\prime},\mp@subsup{S}{}{\prime})\inRe\mp@subsup{R}{}{\prime
            by(blast dest: simE)
    from PTrans RTrans xFreshP have Trans: P|R\Longrightarrow\tau\prec<\nux>( (P'| R')
    by(rule Weak-Early-Step-Semantics.Close1)
    hence }P|R\Longrightarrow^ \tau\prec<\nux>(\mp@subsup{P}{}{\prime}|\mp@subsup{R}{}{\prime}
    by(auto simp add: trancl-into-rtrancl dest: Weak-Early-Step-Semantics.tauTransitionChain)
```



```
Rel'"
            by(blast intro: Par Res)
            ultimately show ?case by blast
        next
            case(cClose2 Q' S' a x)
            have QTrans:Q\longmapstoa<\nux> \prec Q' and STrans: }S\longmapstoa<x>\prec\mp@subsup{S}{}{\prime}\mathrm{ by fact+
            have }x\sharp(P,R)\mathrm{ by fact
            hence xFreshR: }x\sharpR\mathrm{ and xFreshP: x #P by(simp add: fresh-prod)+
            from PSimQ QTrans xFreshP obtain P' where PTrans: P\Longrightarrowa<\nux>\prec 在
                                    and \mp@subsup{P}{}{\prime}RelQ':}(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\operatorname{Rel
            by(blast dest: simE)
            from RSimT STrans obtain R' where RTrans: R\Longrightarrowa<x>\prec R'
                    and R'Rel'T}\mp@subsup{T}{}{\prime}:(\mp@subsup{R}{}{\prime},\mp@subsup{S}{}{\prime})\inRe\mp@subsup{R}{}{\prime
            by(fastforce dest: simE simp add: weakFreeTransition-def)
            from PTrans RTrans xFreshR have Trans: P|R\Longrightarrow\tau\prec<\nux>( }\mp@subsup{P}{}{\prime}|\mp@subsup{R}{}{\prime}
            by(rule Weak-Early-Step-Semantics.Close2)
            hence }P|R\Longrightarrow^`\imath<<\nux>(\mp@subsup{P}{}{\prime}|R'
            by(auto simp add: trancl-into-rtrancl dest: Weak-Early-Step-Semantics.tauTransitionChain)
```



```
Rel"
            by(blast intro: Par Res)
            ultimately show ?case by blast
        qed
    qed
qed
lemma parPres:
    fixes P :: pi
    and }Q ::p
    and }R\mathrm{ :: pi
```

```
    and a :: name
    and Rel :: (pi\timespi) set
    and Rel':: (pi\times pi) set
    assumes PSimQ: }P\rightsquigarrow<\mathrm{ Rel }>
    and PRelQ: }(P,Q)\in\operatorname{Rel
    and Par: }\quad\STU.(S,T)\inRel\Longrightarrow(S|U,T|U)\in\mp@subsup{R}{Rl'}{
    and Res: }\quad\bigwedgeSTx.(S,T)\inRe\mp@subsup{l}{}{\prime}\Longrightarrow(<\nux>S,<\nux>T)\inRel'
    shows P|R\rightsquigarrow<<\mp@subsup{Rel}{}{\prime}>Q|R
proof -
    note PSimQ
    moreover have RSimR: R}\rightsquigarrow<Id>R by(auto intro: reflexive
    moreover note PRelQ moreover have ( }R,R)\inId\mathrm{ by auto
    moreover from Par have \}\PQRT.\llbracket(P,Q)\in\operatorname{Rel; (R,T) \inId\rrbracket\Longrightarrow(P|
R,Q|T)\inRel'
    by auto
    ultimately show ?thesis using Res by(rule parCompose)
qed
lemma resPres:
    fixes P :: pi
    and Q :: pi
    and Rel :: (pi }\times pi) se
    and x :: name
    and Rel'::(pi\times pi) set
    assumes PSimQ: P\rightsquigarrow<Rel>}
```



```
    and RelRel':Rel }\subseteqRel
    and EqvtRel: eqvt Rel
    and EqvtRel': eqvt Rel'
    shows <\nux>P}\rightsquigarrow<<\mp@subsup{Rel}{}{\prime}><\nux>Q
proof -
    from EqvtRel' show ?thesis
    proof(induct rule: simCasesCont[where C=(P,x)])
    case(Bound a y Q')
    have Trans: <\nux>>Q\longmapstoa<\nuy>}\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    have }y\sharp(P,x)\mathrm{ by fact
    hence yineqx: y\not=x and yFreshP:y\sharpP by(simp add: fresh-prod)+
    from Trans yineqx show ?case
    proof(induct rule: resCasesB)
            case(Open Q')
            have QTrans: }Q\longmapstoa[x]\prec\mp@subsup{Q}{}{\prime}\mathrm{ and aineqx: a # x by fact+
            from PSimQ QTrans obtain P' where PTrans: P\Longrightarrow^ a[x]\prec P'
                        and P'RelQ':( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mathrm{ Rel
            by(blast dest: simE)
```

from PTrans aineqx have $<\nu x>P \Longrightarrow a<\nu x>\prec P^{\prime}$
by (force intro: Weak-Early-Step-Semantics.Open simp add: weakFreeTransi-tion-def)
with $\langle y \sharp P\rangle\langle y \neq x\rangle$ have $<\nu x>P \Longrightarrow a<\nu y>\prec\left([(y, x)] \cdot P^{\prime}\right)$
by (force intro: weakTransitionAlpha simp add: abs-fresh name-swap)
moreover from EqvtRel $P^{\prime} \operatorname{RelQ} Q^{\prime} \operatorname{RelRel}^{\prime}$ have $\left([(y, x)] \cdot P^{\prime},[(y, x)] \cdot Q^{\prime}\right) \in$ $R e l^{\prime}$
by (blast intro: eqvtRelI)
ultimately show ?case by blast
next
case(Res $\left.Q^{\prime}\right)$
have QTrans: $Q \longmapsto a<\nu y>\prec Q^{\prime}$ and xineqa: $x \neq a$ by fact +
from PSimQ yFreshP QTrans obtain $P^{\prime}$ where PTrans: $P \Longrightarrow a<\nu y>\prec P^{\prime}$ and $P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}$
by (blast dest: simE)
from PTrans xineqa yineqx yFreshP have ResTrans: $\langle\nu x\rangle P \Longrightarrow a<\nu y>\prec$ $\left(<\nu x>P^{\prime}\right)$
by (blast intro: Weak-Early-Step-Semantics.ResB)
moreover from $P^{\prime} \operatorname{Rel} Q^{\prime}$ have $\left(\left(<\nu x>P^{\prime}\right),\left(<\nu x>Q^{\prime}\right)\right) \in R e l^{\prime}$
by(rule ResRel)
ultimately show ?case by blast
qed
next
case (Free $\alpha Q^{\prime}$ )
have $Q$ Trans: $<\nu x>Q \longmapsto \alpha \prec Q^{\prime}$ by fact
have $\exists c:: n a m e . c \sharp\left(P, Q, Q^{\prime}, \alpha\right)$ by (blast intro: name-exists-fresh)
then obtain c::name where cFresh $Q: c \sharp Q$ and cFreshAlpha: $c \sharp \alpha$ and cFresh $Q^{\prime}: c \sharp Q^{\prime}$ and cFreshP: $c \sharp P$
by (force simp add: fresh-prod)
from $c$ Fresh $P$ have $<\nu x>P=\langle\nu c>([(x, c)] \cdot P)$ by (simp add: alphaRes)
moreover have $\exists P^{\prime} .<\nu c>([(x, c)] \cdot P) \Longrightarrow \alpha \prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in$ Rel $^{\prime}$
proof -
from $Q$ Trans cFresh $Q$ have $<\nu c>([(x, c)] \cdot Q) \longmapsto \alpha \prec Q^{\prime}$ by (simp add: alphaRes)
moreover have $c \sharp \alpha$ by (rule cFreshAlpha)
moreover from PSimQ EqvtRel have $([(x, c)] \cdot P) \rightsquigarrow<\operatorname{Rel}>([(x, c)] \cdot Q)$
by(blast intro: eqvtI)
ultimately show ?thesis
apply (induct rule: resCasesF, auto simp add: residual.inject pi.inject name-abs-eq)
by(blast intro: ResF ResRel dest: simE)
qed
ultimately show ?case by force
qed
qed

```
lemma resChainI:
    fixes }P\mathrm{ :: pi
    and }Q :: p
    and Rel :: (pi\timespi) set
    and lst :: name list
    assumes eqvtRel: eqvt Rel
    and Res: }\{Sy.(R,S)\inRel\Longrightarrow(<\nuy>R,<\nuy>S)\inRe
    and PRelQ: P}\rightsquigarrow<\mathrm{ Rel }>
    shows (resChain lst) P}\rightsquigarrow<\mathrm{ Rel > (resChain lst) Q
proof -
    show ?thesis
    proof(induct lst)
    from PRelQ show resChain [] P\rightsquigarrow<Rel> resChain [] Q by simp
    next
        fix a lst
        assume IH:(resChain lst P)\rightsquigarrow<Rel> (resChain lst Q)
        moreover from Res have \}\PQa.(P,Q)\in\operatorname{Rel \Longrightarrow(<\nua>P,<\nua>Q)\in
Rel
            by simp
    moreover have Rel \subseteqRel by simp
    ultimately have <\nua>(resChain lst P)}\rightsquigarrow<\mathrm{ Rel }><\nu\alpha>(\mathrm{ resChain lst Q) using
eqvtRel
            by(rule-tac resPres)
    thus resChain (a # lst) P }><\mathrm{ Rel> resChain (a # lst) Q
            by simp
    qed
qed
lemma bangPres:
    fixes P :: pi
    and Q :: pi
    and Rel :: (pi\times pi) set
    assumes PRelQ: }\quad(P,Q)\in\mathrm{ Rel
    and Sim: }\quad\RS.(R,S)\inRel\LongrightarrowR\rightsquigarrow<Rel>
    and ParComp: }\RSTU.\llbracket(R,S)\inRel; (T,U)\inRel\rrbracket\Longrightarrow(R|T,
|U)\inRel
    and Res: \}\quad\RSx.(R,S)\inRel'\Longrightarrow(<\nux>>R,<\nux>S)\inRel'
    and RelStay: }\quad\RS.(R|!R,S)\inRel'\Longrightarrow(!R,S)\inRel'
    and BangRelRel':(bangRel Rel)\subseteqRel'
    and eqvtRel': eqvt Rel'
    shows !P}\rightsquigarrow<\mp@subsup{R}{Rel}{
proof -
    let ?Sim = \lambdaP Rs. (\forallax Q . Rs = a<\nux> \prec Q }\longrightarrowx\sharpP\longrightarrow(\exists\mp@subsup{P}{}{\prime}.
```

```
\(\left.\left.\Longrightarrow a<\nu x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in R e l^{\prime}\right)\right) \wedge\)
                                    \(\left(\forall \alpha Q^{\prime} . R s=\alpha \prec Q^{\prime} \longrightarrow\left(\exists P^{\prime} . P \Longrightarrow \alpha \prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in\right.\right.\)
\(\left.R e l^{\prime}\right)\) )
    \(\{\)
        fix \(R s P\)
        assume \(!Q \longmapsto R s\) and \((P,!Q) \in\) bangRel Rel
        hence ?Sim \(P\) Rs using PRelQ
        proof (nominal-induct avoiding: \(P\) rule: bangInduct)
            case(Par1B ax \(Q^{\prime}\) )
            have \(Q\) Trans: \(Q \longmapsto a<\nu x>\prec Q^{\prime}\) and \(x\) Fresh \(Q: x \sharp Q\) by fact +
            have \((P, Q \|!Q) \in\) bangRel Rel and \(x \sharp P\) by fact +
            thus ?case
            proof (induct rule: BRParCases)
                    case (BRPar PR)
                    have PRelQ: \((P, Q) \in\) Rel and RBangRelT: \((R,!Q) \in\) bangRel Rel by
fact+
        have \(x \sharp P \| R\) by fact
            hence \(x\) Fresh \(P: x \sharp P\) and \(x\) Fresh \(R\) : \(x \sharp R\) by simp +
            from PRelQ have PSimQ: \(P \rightsquigarrow<\) Rel \(>Q\) by (rule Sim)
            from \(\langle x \sharp P\rangle\langle x \sharp Q\rangle\) show ?case
            proof(auto simp add: residual.inject alpha' name-fresh-fresh)
                from PSimQ QTrans xFreshP obtain \(P^{\prime}\) where PTrans: \(P \Longrightarrow a<\nu x>\)
\(\prec P^{\prime}\)
                                    and \(P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}\)
                    by (blast dest: simE)
                from PTrans xFreshR have \(P \| R \Longrightarrow a<\nu x>\prec\left(P^{\prime} \| R\right)\)
                    by (rule Weak-Early-Step-Semantics.Par1B)
                moreover from \(P^{\prime}\) RelQ \({ }^{\prime}\) RBangRelT BangRelRel' have \(\left(P^{\prime}\left\|R, Q^{\prime}\right\|\right.\)
\(!Q) \in R e l^{\prime}\)
            by(blast intro: Rel.BRPar)
                ultimately show \(\exists P^{\prime} . P \| R \Longrightarrow a<\nu x>\prec P^{\prime} \wedge\left(P^{\prime}, Q^{\prime} \|!Q\right) \in R e l^{\prime}\)
by blast
            next
                fix \(y\)
            assume ( \(y\) ::name) \(\sharp Q^{\prime}\) and \(y \sharp P\) and \(y \sharp R\)
            from \(Q\) Trans \(\left\langle y \sharp Q^{\prime}\right\rangle\) have \(Q \longmapsto a<\nu y>\prec\left([(x, y)] \cdot Q^{\prime}\right)\) by \((\) simp add:
alphaBoundOutput)
        with \(P \operatorname{Sim} Q\langle y \sharp P\rangle\) obtain \(P^{\prime}\) where \(P\) Trans: \(P \Longrightarrow a<\nu y>\prec P^{\prime}\)
                            and \(P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime},[(x, y)] \cdot Q^{\prime}\right) \in \operatorname{Rel}\)
            by (blast dest: simE)
                    from PTrans \(\langle y \sharp R\rangle\) have \(P \| R \Longrightarrow a<\nu y>\prec\left(P^{\prime} \| R\right)\) by \((\) rule
Weak-Early-Step-Semantics.Par1B)
            moreover from \(P^{\prime}\) RelQ' RBangRelT BangRelRel' have \(\left(P^{\prime} \| R\right.\), ([(y,x)]
- \(\left.\left.Q^{\prime}\right) \|!Q\right) \in\) Rel \(^{\prime}\)
            by (fastforce intro: Rel.BRPar simp add: name-swap)
            ultimately show \(\exists P^{\prime} . P \| R \Longrightarrow a<\nu y>\prec P^{\prime} \wedge\left(P^{\prime},\left([(y, x)] \cdot Q^{\prime}\right) \|\right.\)
\(!Q) \in R e l^{\prime}\) by blast
            qed
        qed
```

```
    next
    case(Par1F \alpha Q' P)
    have QTrans: Q\longmapsto\alpha\prec Q' by fact
    have (P,Q|!Q)\in bangRel Rel by fact
    thus ?case
    proof(induct rule: BRParCases)
            case(BRPar P R)
            have PRelQ:(P,Q)\inRel and RBangRelQ: (R,!Q) \in bangRel Rel by
fact+
            show ?case
            proof(auto simp add: residual.inject)
                from PRelQ have P}\rightsquigarrow<\mathrm{ Rel }>Q\mathrm{ by(rule Sim)
            with QTrans obtain P' where PTrans: P\Longrightarrow\alpha }\Longrightarrow\mp@subsup{P}{}{\prime}\mathrm{ and P'RelQ':( (P',
Q')\inRel
                    by(blast dest: simE)
    from PTrans have P|R\Longrightarrow\alpha \prec P'|R by(rule Weak-Early-Semantics.Par1F)
            moreover from P'RelQ' RBangRelQ have ( }\mp@subsup{P}{}{\prime}|R,\mp@subsup{Q}{}{\prime}|!Q)\in\mathrm{ bangRel
Rel
                by(rule Rel.BRPar)
                ultimately show }\exists\mp@subsup{P}{}{\prime}.P|R\Longrightarrow^ \alpha\prec 质^^(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime}|!Q)\in\mp@subsup{R}{el}{
BangRelRel' by blast
            qed
        qed
    next
    case(Par2B a x Q Q P)
    hence IH: \P.(P,!Q) \in bangRel Rel \Longrightarrow?Sim P (a<\nux>\prec < ') by simp
    have xFreshQ: }x\sharpQ\mathrm{ by fact
    have (P,Q|!Q)\inbangRel Rel and }x\sharpP\mathrm{ by fact+
    thus ?case
    proof(induct rule: BRParCases)
        case(BRPar P R)
                have PRelQ: (P,Q)\inRel and RBangRelQ: (R,!Q) \in bangRel Rel by
fact+
        have }x\sharpP||\mathrm{ by fact
        hence xFreshP: x\sharpP and xFreshR: x }#R\mathrm{ by simp+
        show ?case using <x\sharpQ>
        proof(auto simp add: residual.inject alpha' name-fresh-fresh)
            from IH RBangRelQ have ?Sim R (a<\nux>}\prec<\mp@subsup{Q}{}{\prime})\mathrm{ by blast
                    with xFreshR obtain }\mp@subsup{R}{}{\prime}\mathrm{ where RTrans: R בa<vx>}\prec\mp@subsup{R}{}{\prime}\mathrm{ and
R'BangRelQ':}(\mp@subsup{R}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mp@subsup{R}{Rl}{\prime
                by(blast dest: simE)
            from RTrans xFreshP have }P|R\Longrightarrowa<\nux> \prec(P| | '
                by(auto intro:Weak-Early-Step-Semantics.Par2B)
            moreover from PRelQ R'BangRelQ' have (P| R',Q| Q')\inRel'
                by(rule ParComp)
            ultimately show }\exists\mp@subsup{P}{}{\prime}.P|R\Longrightarrowa<\nux>< \prec P'^( (P',Q| Q')\inRel' by
blast
            next
```

fix $y$
assume ( $y$ :: :name) $\sharp Q^{\prime}$ and $y \sharp R$ and $y \sharp P$
from IH RBangRelQ have? Sim $R\left(a<\nu x>\prec Q^{\prime}\right)$ by blast
with $\left\langle y \sharp Q^{\prime}\right\rangle$ have ? $\operatorname{Sim} R\left(a<\nu y>\prec\left([(x, y)] \cdot Q^{\prime}\right)\right)$ by $(\operatorname{simp}$ add: alphaBoundOutput)
with $\langle y \sharp R\rangle$ obtain $R^{\prime}$ where $R$ Trans: $R \Longrightarrow a<\nu y>\prec R^{\prime}$ and $R^{\prime}$ BangRelQ': $\left(R^{\prime},[(x, y)] \cdot Q^{\prime}\right) \in \operatorname{Rel}^{\prime}$ by (blast dest: simE)
from RTrans $\langle y \sharp P\rangle$ have $P \| R \Longrightarrow a<\nu y>\prec\left(P \| R^{\prime}\right)$ by(auto intro: Weak-Early-Step-Semantics.Par2B)
moreover from PRelQ $R^{\prime}$ BangRelQ' have $\left(P\left\|R^{\prime}, Q\right\|\left([(y, x)] \cdot Q^{\prime}\right)\right)$
$\in R e l^{\prime}$
by (fastforce intro: ParComp simp add: name-swap)
ultimately show $\exists P^{\prime} . P \| R \Longrightarrow a<\nu y>\prec P^{\prime} \wedge\left(P^{\prime}, Q \|\left([(y, x)] \cdot Q^{\prime}\right)\right)$
$\in R e l^{\prime}$ by blast qed
qed
next
case (Par2F $\left.\alpha Q^{\prime} P\right)$
hence $I H: \wedge P$. $(P,!Q) \in$ bangRel Rel $\Longrightarrow$ ?Sim $P\left(\alpha \prec Q^{\prime}\right)$ by simp
have $(P, Q \|!Q) \in$ bangRel Rel by fact
thus ?case
proof (induct rule: BRParCases)
case (BRPar PR)
have PRelQ: $(P, Q) \in$ Rel and RBangRelQ: $(R,!Q) \in$ bangRel Rel by
fact+
show ? case
proof (auto simp add: residual.inject)
from $R$ BangRelQ have ?Sim $R\left(\alpha \prec Q^{\prime}\right)$ by (rule IH)
then obtain $R^{\prime}$ where RTrans: $R \Longrightarrow \alpha \prec R^{\prime}$ and $R^{\prime} \operatorname{Rel}^{\prime}:\left(R^{\prime}, Q^{\prime}\right) \in$
$R e l^{\prime}$
by (blast dest: $\operatorname{sim} E)$
from RTrans have $P\|R \Longrightarrow \wedge \prec P\| R^{\prime}$ by (rule Weak-Early-Semantics.Par2F)
moreover from $P \operatorname{RelQ} R^{\prime} \operatorname{Rel} Q^{\prime}$ have $\left(P\left\|R^{\prime}, Q\right\| Q^{\prime}\right) \in \operatorname{Rel}^{\prime}$ by (rule ParComp)
ultimately show $\exists P^{\prime} . P \| R \Longrightarrow \alpha \prec P^{\prime} \wedge\left(P^{\prime}, Q \| Q^{\prime}\right) \in$ Rel $^{\prime}$ by blast qed
qed
next
case(Comm1 a $\left.Q^{\prime} b Q^{\prime \prime} P\right)$
hence $I H: \wedge P .(P,!Q) \in$ bangRel Rel $\Longrightarrow$ ? Sim $P\left(a[b] \prec Q^{\prime \prime}\right)$ by simp
have $Q$ Trans: $Q \longmapsto a<b>\prec Q^{\prime}$ by fact
have $(P, Q \|!Q) \in$ bangRel Rel by fact
thus ?case
proof (induct rule: BRParCases)
case (BRPar P R)
have PRelQ: $(P, Q) \in$ Rel and RBangRelQ: $(R,!Q) \in$ bangRel Rel by
fact +
show ?case
proof (auto simp add: residual.inject)
from PRelQ have $P \rightsquigarrow<$ Rel $>Q$ by(rule Sim)
with $Q$ Trans obtain $P^{\prime}$ where PTrans: $P \Longrightarrow a<b>\prec P^{\prime}$ and $P^{\prime}$ RelQ':
$\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}$
by(fastforce dest: $\operatorname{simE} \operatorname{simp}$ add: weakFreeTransition-def)
from $R$ BangRelQ have ?Sim $R\left(a[b] \prec Q^{\prime \prime}\right)$ by (rule $\left.I H\right)$
then obtain $R^{\prime}$ where RTrans: $R \Longrightarrow a[b] \prec R^{\prime}$
and $R^{\prime}$ RelQ ${ }^{\prime \prime}:\left(R^{\prime}, Q^{\prime \prime}\right) \in R e l^{\prime}$
by(fastforce dest: simE simp add: weakFreeTransition-def)
from PTrans RTrans have $P \| R \Longrightarrow \tau \prec\left(P^{\prime} \| R^{\prime}\right)$
by(rule Weak-Early-Step-Semantics.Comm1)
hence $P\left\|R \Longrightarrow_{\tau} P^{\prime}\right\| R^{\prime}$
by (auto simp add: trancl-into-rtrancl dest: Weak-Early-Step-Semantics.tauTransitionChain)
moreover from $P^{\prime} \operatorname{Rel} Q^{\prime} R^{\prime} \operatorname{Rel} Q^{\prime \prime}$ have $\left(P^{\prime}\left\|R^{\prime}, Q^{\prime}\right\| Q^{\prime \prime}\right) \in \operatorname{Rel}^{\prime}$
by (rule ParComp)
ultimately show $\exists P^{\prime} .\left(P \| R, P^{\prime}\right) \in\left\{\left(P, P^{\prime}\right) . P \longmapsto \tau \prec P^{\prime}\right\}^{*} \wedge\left(P^{\prime}\right.$,
$\left.Q^{\prime} \| Q^{\prime \prime}\right) \in R e l^{\prime}$
by auto
qed
qed
next
case(Comm2 a b $\left.Q^{\prime} Q^{\prime \prime} P\right)$
hence $I H: \wedge P .(P,!Q) \in$ bangRel Rel $\Longrightarrow$ ?Sim $P\left(a<b>\prec Q^{\prime \prime}\right)$ by simp
have $Q$ Trans: $Q \longmapsto a[b] \prec Q^{\prime}$ by fact
have $(P, Q \|!Q) \in$ bangRel Rel by fact
thus ?case
proof (induct rule: BRParCases)
case (BRPar P R)
have PRelQ: $(P, Q) \in$ Rel and RBangRel $Q:(R,!Q) \in$ bangRel Rel by
fact+
show ?case
proof (auto simp add: residual.inject)
from PRelQ have $P \rightsquigarrow<$ Rel $>Q$ by (rule Sim)
with $Q$ Trans obtain $P^{\prime}$ where PTrans: $P \Longrightarrow a[b] \prec P^{\prime}$ and $P^{\prime}$ RelQ':
$\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}$
by(fastforce dest: simE simp add: weakFreeTransition-def)
from $R$ BangRelQ have ?Sim $R\left(a<b>\prec Q^{\prime \prime}\right)$ by (rule IH $)$
then obtain $R^{\prime}$ where RTrans: $R \Longrightarrow a<b>\prec R^{\prime}$ and $R^{\prime}$ BangRelQ ${ }^{\prime \prime}$ :
$\left(R^{\prime}, Q^{\prime \prime}\right) \in \operatorname{Rel}^{\prime}$
by (fastforce dest: simE simp add: weakFreeTransition-def)
from PTrans RTrans have $P \| R \Longrightarrow \tau \prec\left(P^{\prime} \| R^{\prime}\right)$
by(rule Weak-Early-Step-Semantics.Comm2)
hence $P\left\|R \Longrightarrow_{\tau} P^{\prime}\right\| R^{\prime}$
by (auto simp add: trancl-into-rtrancl dest: Weak-Early-Step-Semantics.tauTransitionChain)
moreover from $P^{\prime}$ RelQ $Q^{\prime} R^{\prime}$ BangRelQ ${ }^{\prime \prime}$ have $\left(P^{\prime}\left\|R^{\prime}, Q^{\prime}\right\| Q^{\prime \prime}\right) \in \operatorname{Rel}^{\prime}$
by (rule ParComp)
ultimately show $\exists P^{\prime} .\left(P \| R, P^{\prime}\right) \in\left\{\left(P, P^{\prime}\right) . P \longmapsto \tau \prec P^{\prime}\right\}^{*} \wedge\left(P^{\prime}\right.$, $\left.Q^{\prime} \| Q^{\prime \prime}\right) \in R e l^{\prime}$ by auto qed qed
next
case (Close1 a x $Q^{\prime} Q^{\prime \prime} P$ )
hence $I H: \wedge P .(P,!Q) \in$ bangRel Rel $\Longrightarrow$ ?Sim $P\left(a<\nu x>\prec Q^{\prime \prime}\right)$ by simp have $Q$ Trans: $Q \longmapsto a<x>\prec Q^{\prime}$ by fact
have $(P, Q \|!Q) \in$ bangRel Rel and $x \sharp P$ by fact+
thus ?case
proof (induct rule: BRParCases)
case (BRPar P R)
have PRelQ: $(P, Q) \in$ Rel and RBangRelQ: $(R,!Q) \in$ bangRel Rel by fact+
have $x \sharp P \| R$ by fact
hence $x$ Fresh $: x \sharp R$ and $x$ Fresh $P: x \sharp P$ by simp +
show ? case
proof (auto simp add: residual.inject)
from PRelQ have $P \rightsquigarrow<$ Rel $>Q$ by (rule Sim)
with QTrans obtain $P^{\prime}$ where PTrans: $P \Longrightarrow a<x>\prec P^{\prime}$ and $P^{\prime}$ RelQ': $\left(P^{\prime}, Q^{\prime}\right) \in \operatorname{Rel}$
by(fastforce dest: simE simp add: weakFreeTransition-def)
from RBangRelQ have ?Sim $R\left(a<\nu x>\prec Q^{\prime \prime}\right) \mathbf{b y}($ rule IH $)$
with $x$ Fresh $R$ obtain $R^{\prime}$ where RTrans: $R \Longrightarrow a<\nu x>\prec R^{\prime}$ and $R^{\prime} R e l Q^{\prime \prime}:\left(R^{\prime}, Q^{\prime \prime}\right) \in R e l^{\prime}$
by (blast dest: $\operatorname{simE}$ )
from PTrans RTrans xFreshP have $P \| R \Longrightarrow \tau \prec<\nu x>\left(P^{\prime} \| R^{\prime}\right)$
by (rule Weak-Early-Step-Semantics.Close1)
moreover from $P^{\prime}$ RelQ $Q^{\prime} R^{\prime} \operatorname{Rel} Q^{\prime \prime}$ have $\left(<\nu x>\left(P^{\prime} \| R^{\prime}\right),<\nu x>\left(Q^{\prime} \|\right.\right.$ $\left.\left.Q^{\prime \prime}\right)\right) \in R e l^{\prime}$
by (force intro: ParComp Res)
ultimately show $\exists P^{\prime} .\left(P \| R, P^{\prime}\right) \in\left\{\left(P, P^{\prime}\right) . P \longmapsto \tau \prec P^{\prime}\right\}^{*} \wedge\left(P^{\prime}\right.$, $\left.<\nu x>\left(Q^{\prime} \| Q^{\prime \prime}\right)\right) \in R e l^{\prime}$ by auto
qed
qed
next
case (Close2 a x $Q^{\prime} Q^{\prime \prime} P$ )
hence $I H: \wedge P .(P,!Q) \in$ bangRel Rel $\Longrightarrow$ ?Sim $P\left(a<x>\prec Q^{\prime \prime}\right)$ by simp have $Q$ Trans: $Q \longmapsto a<\nu x>\prec Q^{\prime}$ by fact
have $(P, Q \|!Q) \in$ bangRel Rel and $x \sharp P$ by fact +
thus ?case
proof (induct rule: BRParCases)
case (BRPar P R)
have PRelQ: $(P, Q) \in$ Rel and RBangRelQ: $(R,!Q) \in$ bangRel Rel by fact +
have $x \sharp P \| R$ by fact
hence $x$ Fresh $: x \sharp P$ and $x F r e s h R$ : $x \sharp R$ by simp +

```
    show ?case
    proof(auto simp add: residual.inject)
    from PRelQ have P}\rightsquigarrow<\mathrm{ Rel }>Q\mathrm{ by(rule Sim)
    with QTrans xFreshP obtain P' where PTrans: P\Longrightarrowa<\nux> \prec P'
                                    and P'RelQ':( }\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mathrm{ Rel
        by(blast dest: simE)
    from RBangRelQ have ?Sim R(a<x> \prec Q') by(rule IH)
    with xFreshR obtain R' where RTrans: R\Longrightarrowa<x> \prec R'
                    and R'RelQ':}:(\mp@subsup{R}{}{\prime},\mp@subsup{Q}{}{\prime\prime})\in\mp@subsup{Rel}{}{\prime
        by(fastforce simp add: weakFreeTransition-def)
    from PTrans RTrans xFreshR have P|R\Longrightarrow\tau\prec<\nux>( P'| R')
        by(rule Weak-Early-Step-Semantics.Close2)
        moreover from P'RelQ' R'RelQ'\prime have ( <\nux> ( }\mp@subsup{P}{}{\prime}||\mp@subsup{R}{}{\prime}),<\nux>(\mp@subsup{Q}{}{\prime}
Q'\)) \inReR'
            by(force intro: ParComp Res)
            ultimately show }\exists\mp@subsup{P}{}{\prime}.(P|R,\mp@subsup{P}{}{\prime})\in{(P,\mp@subsup{P}{}{\prime}).P\longmapsto~\prec\mp@code{P}\mp@subsup{}}{}{*}\wedge(\mp@subsup{P}{}{\prime}
```



```
        qed
        qed
    next
        case(Bang Rs)
    hence IH: \bigwedgeP.(P,Q|!Q) G bangRel Rel \Longrightarrow ?Sim P Rs by simp
    have (P,!Q) \in bangRel Rel by fact
    thus ?case
    proof(induct rule: BRBangCases)
        case(BRBang P)
        have PRelQ: (P,Q)\in Rel by fact
        hence (!P,!Q)\in bangRel Rel by(rule Rel.BRBang)
        with PRelQ have (P|!P,Q|!Q)\in bangRel Rel by(rule Rel.BRPar)
        hence IH:?Sim (P|!P) Rs by(rule IH)
        show ?case
        proof(intro conjI allI impI)
            fix }\mp@subsup{Q}{}{\prime}a
            assume Rs=a<\nux>\prec\prec Q' and x\sharp!P
            then obtain }\mp@subsup{P}{}{\prime}\mathrm{ where PTrans: (P|!P) #a<vx>}\prec\mp@subsup{P}{}{\prime
                    and }\mp@subsup{P}{}{\prime}\mathrm{ RelQ':}(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mp@subsup{\mathrm{ Rel' using IH}}{}{\prime
            by(auto simp add: residual.inject)
            from PTrans have ! P\Longrightarrowa<\nux>}\prec\mp@subsup{P}{}{\prime
                    by(force intro:Weak-Early-Step-Semantics.Bang simp add:weakFree-
Transition-def)
            with }\mp@subsup{P}{}{\prime}Rel\mp@subsup{Q}{}{\prime}\mathrm{ show }\exists\mp@subsup{P}{}{\prime}.!P\Longrightarrowa<\nux>\prec \prec P'^( (P', Q')\inRel' by blas
        next
            fix }\mp@subsup{Q}{}{\prime}
            assume Rs=\alpha\prec\mp@subsup{Q}{}{\prime}
            then obtain P' where PTrans: }(P|!P)\Longrightarrow\alpha < < P
                        and P'RelQ':}(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mathrm{ Rel' using IH
            by auto
```



```
                    proof(induct rule: transitionCases)
                    case Step
                    have P|!P\Longrightarrow\alpha\prec P' by fact
                    hence !P\Longrightarrow\alpha< P' by(rule Weak-Early-Step-Semantics.Bang)
                    with P'RelQ' show ?case by(force simp add: weakFreeTransition-def)
                    next
                    case Stay
                    have !P\Longrightarrow^ }\tau\prec!P\mathrm{ by(simp add: weakFreeTransition-def)
                    moreover assume ( }P|!P,\mp@subsup{Q}{}{\prime})\in\mp@subsup{\operatorname{Rel}}{}{\prime
                    hence (!P, Q') \inRel' by(blast intro: RelStay)
                    ultimately show ?case by blast
                    qed
            qed
            qed
        qed
    }
    moreover from PRelQ have (!P,!Q) \in bangRel Rel by(rule Rel.BRBang)
    ultimately show ?thesis by(auto simp add: weakSimulation-def)
qed
lemma bangRelSim:
    fixes P :: pi
    and Q :: pi
    and Rel :: (pi\times pi) set
    and Rel'l :: (pi\timespi) set
    assumes PBangRelQ:}(P,Q)\in\mathrm{ bangRel Rel
    and Sim: }\RS.(R,S)\inRel\LongrightarrowR\rightsquigarrow<Rel>
    and ParComp: }\quad\bigwedgeRSTU.\llbracket(R,S)\inRel; (T,U)\inRel\rrbracket\Longrightarrow(R|T,
| U)\inRel'
```



```
    and RelStay: }\quad\RS.(R|!R,S)\inRe\mp@subsup{l}{}{\prime}\Longrightarrow(!R,S)\inRel'\
    and BangRelRel':(bangRel Rel)}\subseteqRe\mp@subsup{l}{}{\prime
    and eqvtRel': eqvt Rel'
    and Eqvt: eqvt Rel
    shows P}\rightsquigarrow<\mp@subsup{\mathrm{ Rel'}}{}{\prime}>
proof -
    from PBangRelQ show ?thesis
    proof(induct rule: bangRel.induct)
    case(BRBang P Q)
    have PRelQ: (P,Q)\in Rel by fact
    thus ?case using ParComp Res BangRelRel' eqvtRel' Eqvt RelStay Sim
        by(rule-tac bangPres)
    next
    case(BRPar P Q R T)
    have }(P,Q)\in\mathrm{ Rel by fact
```

```
    moreover hence P}\rightsquigarrow<\mathrm{ Rel }>Q\mathrm{ by(rule Sim)
    moreover have R}\rightsquigarrow<\mp@subsup{Rel}{}{\prime}>T\mathrm{ by fact
    moreover have (R,T)\in bangRel Rel by fact
    ultimately show ?case using ParComp eqvtRel' Res Eqvt BangRelRel'
    by(blast intro: parCompose)
    next
        case(BRRes P Q x)
        have P}\rightsquigarrow<\mp@subsup{\mathrm{ Rel'}}{}{\prime}>Q by fac
        thus ?case using BangRelRel' eqvtRel' Res by(blast intro: resPres)
    qed
qed
end
theory Strong-Early-Late-Comp
    imports Strong-Late-Bisim-Subst-SC Strong-Early-Bisim-Subst
begin
abbreviation TransitionsLate-judge \(\left(-\longmapsto_{l}-[80,80] 80\right)\) where \(P \longmapsto_{l} R s \equiv\) transitions PRs
abbreviation TransitionsEarly-judge \(\left(-\longmapsto_{e}-[80,80] 80\right)\) where \(P \longmapsto_{e} R s \equiv\) TransitionsEarly P Rs
abbreviation Transitions-InputjudgeLate ( \(-<->\prec_{l}-[80,80]\) 80) where \(a<x>\) \(\prec_{l} P^{\prime} \equiv\left(\right.\) Late-Semantics.BoundR \(\left(\right.\) Late-Semantics.InputS a) x \(\left.P^{\prime}\right)\)
abbreviation Transitions-OutputjudgeLate (-[-] \(\left.\prec_{l}-[80,80] 80\right)\) where \(a[b] \prec_{l}\) \(P^{\prime} \equiv\left(\right.\) Late-Semantics.FreeR (Late-Semantics.OutputR a b) \(\left.P^{\prime}\right)\)
abbreviation Transitions-BoundOutputjudgeLate \(\left(-<\nu->\prec_{l}-[80,80] 80\right)\) where \(a<\nu x>\prec_{l} P^{\prime} \equiv\left(\right.\) Late-Semantics.BoundR (Late-Semantics.BoundOutputS a) x \(\left.P^{\prime}\right)\) abbreviation Transitions-TaujudgeLate \(\left(\tau \prec_{l}-80\right)\) where \(\tau \prec_{l} P^{\prime} \equiv\) (Late-Semantics.FreeR Late-Semantics.TauR \(P^{\prime}\) )
abbreviation Transitions-InputjudgeEarly \(\left(-<->\prec_{e}-[80,80] 80\right)\) where \(a<x>\) \(\prec_{e} P^{\prime} \equiv\left(\right.\) Early-Semantics.FreeR (Early-Semantics.InputR a x) \(P^{\prime}\) )
abbreviation Transitions-OutputjudgeEarly \(\left(-[-] \prec_{e}-[80,80]\right.\) 80) where \(a[b] \prec_{e}\) \(P^{\prime} \equiv\left(\right.\) Early-Semantics.FreeR (Early-Semantics.OutputR a b) \(P^{\prime}\) )
abbreviation Transitions-BoundOutputjudgeEarly \(\left(-\langle\nu-\rangle \prec_{e}-[80,80] 80\right)\) where
\(a<\nu x>\prec_{e} P^{\prime} \equiv\left(\right.\) Early-Semantics.BoundOutputR a x \(\left.P^{\prime}\right)\)
abbreviation Transitions-TaujudgeEarly \(\left(\tau \prec_{e}-80\right)\) where \(\tau \prec_{e} P^{\prime} \equiv(\) Early-Semantics.FreeR
Early-Semantics.TauR \(P^{\prime}\) )
lemma earlyLateOutput:
fixes \(P\) :: \(p i\)
and \(a\) :: name
and \(b\) :: name
and \(P^{\prime}:: p i\)
assumes \(P \longmapsto_{e} a[b] \prec_{e} P^{\prime}\)
```

```
    shows }P\longmapsto\mp@subsup{\longmapsto}{l}{}a[b]\mp@subsup{\prec}{l}{}\mp@subsup{P}{}{\prime
using assms
proof(nominal-induct rule: Early-Semantics.outputInduct)
    case(Output a b P)
    show ?case by(rule Late-Semantics.Output)
next
    case(Match Pab P' c)
    have }P\mp@subsup{\longmapsto}{l}{}a[b]\mp@subsup{\prec}{l}{}\mp@subsup{P}{}{\prime}\mathrm{ by fact
    thus ?case by(rule Late-Semantics.Match)
next
    case(Mismatch P a b P' c d)
    from <P \longmapsto\longmapsto }a[b]\mp@subsup{\prec}{l}{}\mp@subsup{P}{}{\prime}\rangle\langlec\not=d
    show ?case by(rule Late-Semantics.Mismatch)
next
    case(Sum1 P a b P' Q)
    have P}\mp@subsup{\longmapsto}{l}{}a[b]\mp@subsup{\prec}{l}{}\mp@subsup{P}{}{\prime}\mathrm{ by fact
    thus ?case by(rule Late-Semantics.Sum1)
next
    case(Sum2 Q a b Q'P)
    have }Q\mp@subsup{\longmapsto}{l}{}a[b]\mp@subsup{\prec}{l}{}\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    thus ?case by(rule Late-Semantics.Sum2)
next
    case(Par1 P a b P' Q)
    have P}\longmapsto\mp@subsup{\longmapsto}{l}{}a[b]\mp@subsup{\prec}{l}{}\mp@subsup{P}{}{\prime}\mathrm{ by fact
    thus ?case by(rule Late-Semantics.Par1F)
next
    case(Par2 Q a b Q' P)
    have }Q\mp@subsup{\longmapsto}{l}{}a[b]\mp@subsup{\prec}{l}{}\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    thus ?case by(rule Late-Semantics.Par2F)
next
    case(Res P a b P' x)
    have }P\mp@subsup{\longmapsto}{l}{}a[b]\mp@subsup{\prec}{l}{}\mp@subsup{P}{}{\prime}\mathrm{ and }x\not=a\mathrm{ and }x\not=b\mathrm{ by fact+
    thus ?case by(force intro: Late-Semantics.ResF)
next
    case(Bang P ab P')
    have }P|!P\mp@subsup{\longmapsto}{l}{}a[b]\mp@subsup{\prec}{l}{}\mp@subsup{P}{}{\prime}\mathrm{ by fact
    thus ?case by(rule Late-Semantics.Bang)
qed
lemma lateEarlyOutput:
    fixes }P::p
    and a :: name
    and b :: name
    and }\mp@subsup{P}{}{\prime}:: p
    assumes }P\mp@subsup{\longmapsto}{l}{}a[b]\mp@subsup{\prec}{l}{}\mp@subsup{P}{}{\prime
    shows }P\mp@subsup{\longmapsto}{e}{}a[b]\mp@subsup{\prec}{e}{}\mp@subsup{P}{}{\prime
using assms
```

```
proof(nominal-induct rule: Late-Semantics.outputInduct)
    case(Output a b P)
    thus ?case by(rule Early-Semantics.Output)
next
    case(Match P a b P' c)
    have }P\mp@subsup{\longmapsto}{e}{}a[b]\mp@subsup{\prec}{e}{}\mp@subsup{P}{}{\prime}\mathrm{ by fact
    thus ?case by(rule Early-Semantics.Match)
next
    case(Mismatch P a b P' c d)
    have }P\mp@subsup{\longmapsto}{e}{}a[b]\mp@subsup{\prec}{e}{}\mp@subsup{P}{}{\prime}\mathrm{ and }c\not=d\mathrm{ by fact+
    thus ?case by(rule Early-Semantics.Mismatch)
next
    case(Sum1 P a b P' Q)
    have }P\mp@subsup{\longmapsto}{e}{}a[b]\mp@subsup{\prec}{e}{}\mp@subsup{P}{}{\prime}\mathrm{ by fact
    thus ?case by(rule Early-Semantics.Sum1)
next
    case(Sum2 Q a b Q'P)
    have }Q\mp@subsup{\longmapsto}{e}{}a[b]\mp@subsup{\prec}{e}{}\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    thus ?case by(rule Early-Semantics.Sum2)
next
    case(Par1 P a b P' Q)
    have P}\mp@subsup{\longmapsto}{e}{}a[b]\mp@subsup{\prec}{e}{}\mp@subsup{P}{}{\prime}\mathrm{ by fact
    thus ?case by(rule Early-Semantics.Par1F)
next
    case(Par2 Q a b Q' P)
    have }Q\mp@subsup{\longmapsto}{e}{}a[b]\mp@subsup{\prec}{e}{}\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    thus ?case by(rule Early-Semantics.Par2F)
next
    case(Res P a b P' x)
    have }P\mp@subsup{\longmapsto}{e}{}a[b]\mp@subsup{\prec}{e}{}\mp@subsup{P}{}{\prime}\mathrm{ and }x\not=a\mathrm{ and }x\not=b\mathrm{ by fact+
    thus ?case by(force intro: Early-Semantics.ResF)
next
    case(Bang P a b P')
    have P|!P\longmapsto\longmapsto}\mp@subsup{\longmapsto}{e}{}a[b]\mp@subsup{\prec}{e}{}\mp@subsup{P}{}{\prime}\mathrm{ by fact
    thus?case by(rule Early-Semantics.Bang)
qed
lemma outputEq:
    fixes P :: pi
    and a :: name
    and b :: name
    and }\mp@subsup{P}{}{\prime}::p
    shows }P\mp@subsup{\longmapsto}{e}{}a[b]\mp@subsup{\prec}{e}{}\mp@subsup{P}{}{\prime}=P\mp@subsup{\longmapsto}{l}{}a[b]\mp@subsup{\prec}{l}{}\mp@subsup{P}{}{\prime
by(auto intro: lateEarlyOutput earlyLateOutput)
lemma lateEarlyBoundOutput:
    fixes P :: pi
    and a :: name
```

```
    and x :: name
    and }\mp@subsup{P}{}{\prime}::p
    assumes }P\longmapsto\mp@subsup{}{l}{}a<\nux>\mp@subsup{\prec}{l}{}\mp@subsup{P}{}{\prime
    shows }P\mp@subsup{\longmapsto}{e}{}a<\nux>\mp@subsup{\prec}{e}{}\mp@subsup{P}{}{\prime
proof -
    have Goal: \P a x P'.\llbracketP\longmapsto\longmapsto \a<\nux> \prec}\mp@subsup{l}{l}{}\mp@subsup{P}{}{\prime};x\sharpP\rrbracket\LongrightarrowP\longmapsto\mp@subsup{\longmapsto}{e}{}a<\nux>\mp@subsup{\prec}{e}{}\mp@subsup{P}{}{\prime
    proof -
    fix Pax P'
    assume P}\mp@subsup{\longmapsto}{l}{}a<\nux>\mp@subsup{\prec}{l}{}\mp@subsup{P}{}{\prime}\mathrm{ and }x\sharp
    thus }P\longmapsto\mp@subsup{\longmapsto}{e}{}a<\nux>>\mp@subsup{\prec}{e}{}\mp@subsup{P}{}{\prime
    proof(nominal-induct rule: Late-Semantics.boundOutputInduct)
        case(Match P a x P'b)
        have }P\mp@subsup{\longmapsto}{e}{}a<\nux>\mp@subsup{\prec}{e}{}\mp@subsup{P}{}{\prime}\mathrm{ by fact
        thus ?case by(rule Early-Semantics.Match)
    next
        case(Mismatch P a x P'bc)
        have }P\mp@subsup{\longmapsto}{e}{}a<\nux>\mp@subsup{\prec}{e}{}\mp@subsup{P}{}{\prime}\mathrm{ and }b\not=c\mathrm{ by fact+
        thus ?case by(rule Early-Semantics.Mismatch)
    next
        case(Open P a x P')
        have }P\mp@subsup{\longmapsto}{l}{}a[x]\mp@subsup{\prec}{l}{}\mp@subsup{P}{}{\prime}\mathrm{ by fact
        hence }P\mp@subsup{\longmapsto}{e}{}a[x]\mp@subsup{\prec}{e}{}\mp@subsup{P}{}{\prime}\mathrm{ by(rule lateEarlyOutput)
        moreover have a\not=x by fact
        ultimately show ?case by(rule Early-Semantics.Open)
    next
        case(Sum1 P Q a x P
        have }P\mp@subsup{\longmapsto}{e}{}a<\nux>\mp@subsup{\prec}{e}{}\mp@subsup{P}{}{\prime}\mathrm{ by fact
        thus ?case by(rule Early-Semantics.Sum1)
    next
        case(Sum2 P Q a x Q')
        have }Q\mp@subsup{\longmapsto}{e}{}a<\nux>\mp@subsup{\prec}{e}{}\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    thus ?case by(rule Early-Semantics.Sum2)
    next
        case(Par1 P P' Q a x)
        have }P\mp@subsup{\longmapsto}{e}{e}a<\nux>\mp@subsup{\prec}{e}{}\mp@subsup{P}{}{\prime}\mathrm{ and }x\sharpQ\mathrm{ by fact+
        thus ?case by(rule Early-Semantics.Par1B)
    next
        case(Par2 P Q Q' a x)
        have }Q\mp@subsup{\longmapsto}{e}{}a<\nux>\mp@subsup{\prec}{e}{}\mp@subsup{Q}{}{\prime}\mathrm{ and }x\sharpP\mathrm{ by fact+
        thus ?case by(rule Early-Semantics.Par2B)
    next
    case(Res P P' a x y)
    have }P\mp@subsup{\longmapsto}{e}{}a<\nux>\mp@subsup{\prec}{e}{}\mp@subsup{P}{}{\prime}\mathrm{ and }y\not=a\mathrm{ and }y\not=x\mathrm{ by fact+
    thus ?case by(force intro: Early-Semantics.ResB)
    next
    case(Bang P a x P')
    have }P|!P\longmapsto\mp@subsup{\longmapsto}{e}{}a<\nux>\mp@subsup{\prec}{e}{}\mp@subsup{P}{}{\prime}\mathrm{ by fact
```

```
        thus ?case by(rule Early-Semantics.Bang)
    qed
qed
have \existsc::name.c\sharp (P, P
then obtain c::name where cFreshP:c\sharpP and cFreshP':c\sharp P' and c\not=x
    by(force simp add: fresh-prod)
from assms cFreshP' have }P\mp@subsup{\longmapsto}{l}{}a<\nuc>\mp@subsup{\prec}{l}{}([(x,c)]\cdot\mp@subsup{P}{}{\prime}
    by(simp add: Late-Semantics.alphaBoundResidual)
hence }P\mp@subsup{\longmapsto}{e}{}a<\nuc>\mp@subsup{\prec}{e}{}([(x,c)]\cdot\mp@subsup{P}{}{\prime})\mathrm{ using cFreshP
    by(rule Goal)
moreover from cFreshP'}\langlec\not=x\rangle\mathrm{ have }x\sharp[(x,c)]\cdot\mp@subsup{P}{}{\prime}\mathbf{by}(simp add: name-fresh-left
name-calc)
ultimately show?thesis by(simp add: Early-Semantics.alphaBoundOutput name-swap)
qed
lemma earlyLateBoundOutput:
    fixes P :: pi
    and a :: name
    and x :: name
    and }\mp@subsup{P}{}{\prime}::p
    assumes }P\longmapsto\mp@subsup{\longmapsto}{e}{}a<\nux>\mp@subsup{\prec}{e}{}\mp@subsup{P}{}{\prime
    shows }P\longmapsto\mp@subsup{}{l}{}a<\nux>\mp@subsup{\prec}{l}{}\mp@subsup{P}{}{\prime
proof -
    have Goal: \P a x P'. \llbracketP\longmapsto\longmapsto }a<\nux>\mp@subsup{\prec}{e}{}\mp@subsup{P}{}{\prime};x\sharpP\rrbracket\LongrightarrowP\longmapsto\mp@subsup{\longmapsto}{l}{}a<\nux>\mp@subsup{\prec}{l}{}\mp@subsup{P}{}{\prime
    proof -
    fix Pax P
    assume P}\longmapsto\mp@subsup{\longmapsto}{e}{}a<\nux>\prec\mp@subsup{P}{}{\prime}\mathrm{ and }x\sharp
    thus }P\longmapsto\mp@subsup{\longmapsto}{l}{}a<\nux>\mp@subsup{\prec}{l}{}\mp@subsup{P}{}{\prime
    proof(nominal-induct rule: Early-Semantics.boundOutputInduct)
        case(Match P a x P' b)
        have P}\mp@subsup{\longmapsto}{l}{}a<\nux>\prec\mp@subsup{P}{}{\prime}\mathrm{ by fact
        thus ?case by(rule Late-Semantics.Match)
    next
        case(Mismatch P a x P'bc)
        have P}\mp@subsup{\longmapsto}{l}{}a<\nux>\prec\mp@subsup{P}{}{\prime}\mathrm{ and }b\not=c\mathrm{ by fact+
        thus ?case by(rule Late-Semantics.Mismatch)
    next
        case(Open P a x P')
        have P}\mp@subsup{\longmapsto}{e}{e}a[x]\mp@subsup{\prec}{e}{}\mp@subsup{P}{}{\prime}\mathrm{ by fact
        hence }P\mp@subsup{\longmapsto}{l}{}a[x]\mp@subsup{\prec}{l}{}\mp@subsup{P}{}{\prime}\mathrm{ by(rule earlyLateOutput)
        moreover have a\not=x by fact
        ultimately show ?case by(rule Late-Semantics.Open)
    next
        case(Sum1 P Q a x P')
        have P\longmapsto\longmapsto}\mp@subsup{l}{l}{}a<\nux>\mp@subsup{\prec}{l}{}\mp@subsup{P}{}{\prime}\mathrm{ by fact
    thus ?case by(rule Late-Semantics.Sum1)
```

```
    next
        case(Sum2 P Q a x Q ')
        have }Q\mp@subsup{\longmapsto}{l}{}a<\nux>\mp@subsup{\prec}{l}{}\mp@subsup{Q}{}{\prime}\mathrm{ by fact
        thus ?case by(rule Late-Semantics.Sum2)
    next
        case(Par1 P P' Q a x)
        have }P\mp@subsup{\longmapsto}{l}{}a<\nux>\mp@subsup{\prec}{l}{}\mp@subsup{P}{}{\prime}\mathrm{ and }x\sharpQ\mathrm{ by fact+
        thus?case by(rule Late-Semantics.Par1B)
    next
        case(Par2 P Q Q' a x)
        have }Q\mp@subsup{\longmapsto}{l}{}a<\nux>\mp@subsup{\prec}{l}{}\mp@subsup{Q}{}{\prime}\mathrm{ and }x\sharpP\mathrm{ by fact+
        thus ?case by(rule Late-Semantics.Par2B)
    next
        case(Res P P' a x y)
        have P\longmapsto}\mp@subsup{\longmapsto}{l}{}a<\nux>\mp@subsup{\prec}{l}{}\mp@subsup{P}{}{\prime}\mathrm{ and }y\not=a\mathrm{ and }y\not=x\mathrm{ by fact+
        thus ?case by(force intro: Late-Semantics.ResB)
    next
        case(Bang P a x P')
        have }P|!P\mp@subsup{\longmapsto}{l}{}a<\nux>\prec\mp@subsup{P}{}{\prime}\mathrm{ by fact
        thus ?case by(rule Late-Semantics.Bang)
    qed
qed
have \existsc::name.c }\sharp(P,\mp@subsup{P}{}{\prime},x)\mathrm{ by(blast intro: name-exists-fresh)
then obtain c::name where cFreshP:c\sharpP and cFreshP':c\sharp P' and c\not=x
    by(force simp add: fresh-prod)
from assms cFreshP' have P\longmapsto\longmapsto }a<\nuc>\mp@subsup{\prec}{e}{}([(x,c)]\cdot\mp@subsup{P}{}{\prime}
    by(simp add: Early-Semantics.alphaBoundOutput)
hence P \longmapsto\longmapstol }a<\nuc>\mp@subsup{\prec}{l}{}([(x,c)]\cdot\mp@subsup{P}{}{\prime})\mathrm{ using cFreshP
    by(rule Goal)
moreover from cFreshP'}\langlec\not=x\rangle\mathrm{ have }x\sharp[(x,c)]\cdot\mp@subsup{P}{}{\prime}\mathbf{by}(\mathrm{ simp add: name-fresh-left
name-calc)
    ultimately show ?thesis by(simp add: Late-Semantics.alphaBoundResidual name-swap)
qed
lemma BoundOutputEq:
    fixes P :: pi
    and a :: name
    and x :: name
    and }\mp@subsup{P}{}{\prime}:: p
    shows }P\mp@subsup{\longmapsto}{e}{}a<\nux>\mp@subsup{\prec}{e}{}\mp@subsup{P}{}{\prime}=P\mp@subsup{\longmapsto}{l}{}a<\nux>\mp@subsup{\prec}{l}{}\mp@subsup{P}{}{\prime
by(auto intro: earlyLateBoundOutput lateEarlyBoundOutput)
lemma lateEarlyInput:
    fixes }P\mathrm{ :: pi
    and a :: name
    and }x\mathrm{ :: name
    and }\mp@subsup{P}{}{\prime}::p
```

$$
\text { and } u \text { :: name }
$$

```
    assumes PTrans: \(P \longmapsto_{l} a<x>\prec_{l} P^{\prime}\)
    shows \(P \longmapsto_{e} a<u>\prec_{e}\left(P^{\prime}[x::=u]\right)\)
proof -
    have Goal: \(\bigwedge P\) a \(x P^{\prime} u . \llbracket P \longmapsto_{l} a<x>\prec_{l} P^{\prime} ; x \sharp P \rrbracket \Longrightarrow P \longmapsto_{e} a<u>\prec_{e}\)
\(\left(P^{\prime}[x::=u]\right)\)
    proof -
    fix \(P\) a \(x P^{\prime} u\)
    assume \(P \longmapsto_{l} a<x>\prec_{l} P^{\prime}\) and \(x \sharp P\)
    thus \(P \longmapsto_{e} a<u>\prec_{e}\left(P^{\prime}[x::=u]\right)\)
    proof (nominal-induct avoiding: u rule: Late-Semantics.inputInduct)
        case(Input a x P)
        thus ?case by(rule Early-Semantics.Input)
    next
            case(Match Pax \(\left.P^{\prime} b u\right)\)
            have \(P \longmapsto_{e} a<u>\prec_{e}\left(P^{\prime}[x::=u]\right)\) by fact
            thus ?case by(rule Early-Semantics.Match)
    next
            case(Mismatch P a x \(P^{\prime}\) b c u)
            have \(P \longmapsto_{e} a<u>\prec_{e}\left(P^{\prime}[x::=u]\right)\) by fact
            moreover have \(b \neq c\) by fact
            ultimately show ?case by(rule Early-Semantics.Mismatch)
    next
            case(Sum1 P \(Q\) a x \(P^{\prime}\) )
            have \(P \longmapsto_{e} a<u>\prec_{e}\left(P^{\prime}[x::=u]\right)\) by fact
            thus ?case by(rule Early-Semantics.Sum1)
    next
            case(Sum2 P \(Q\) a x \(Q^{\prime}\) )
            have \(Q \longmapsto_{e} a<u>\prec_{e}\left(Q^{\prime}[x::=u]\right)\) by fact
            thus ?case by(rule Early-Semantics.Sum2)
    next
            case(Par1 P \(P^{\prime} Q\) a \(x\) )
            have \(P \longmapsto_{e} a<u>\prec_{e}\left(P^{\prime}[x::=u]\right)\) by fact
            hence \(P \| Q \longmapsto_{e} a<u>\prec_{e}\left(P^{\prime}[x::=u] \| Q\right) \mathbf{b y}(\) rule Early-Semantics.Par1F)
            moreover have \(x \sharp Q\) by fact
            ultimately show ? case by (simp add: forget)
    next
            case (Par2 P Q Q \({ }^{\prime}\) a \(x\) )
            have \(Q \longmapsto_{e} a<u>\prec_{e}\left(Q^{\prime}[x::=u]\right)\) by fact
            hence \(P \| Q \longmapsto_{e} a<u>\prec_{e}\left(P \| Q^{\prime}[x::=u]\right)\) by(rule Early-Semantics.Par2F)
            moreover have \(x \sharp P\) by fact
            ultimately show ? case by (simp add: forget)
    next
            case \(\left(\right.\) Res \(P P^{\prime}\) a \(x\) y u)
            have \(P \longmapsto_{e} a<u>\prec_{e}\left(P^{\prime}[x::=u]\right)\) and \(y \neq a\) and yinequ: \(y \sharp u\) by fact +
            hence \(<\nu y>P \longmapsto_{e} a<u>\prec_{e}<\nu y>\left(P^{\prime}[x::=u]\right)\) by(force intro: Early-Semantics.ResF)
            moreover have \(y \neq x\) by fact
```

ultimately show ?case using yinequ by simp

## next

case(Bang Pax $P^{\prime} u$ )
have $P \|!P \longmapsto{ }_{e} a<u>\prec_{e}\left(P^{\prime}[x::=u]\right)$ by fact
thus ?case by(rule Early-Semantics.Bang)
qed
qed
have $\exists c$ ::name. $c \sharp\left(P, P^{\prime}\right) \mathbf{b y}$ (blast intro: name-exists-fresh)
then obtain $c::$ name where cFresh $P: c \sharp P$ and cFresh $P^{\prime}: c \sharp P^{\prime}$ by (force simp add: fresh-prod)
from assms cFresh $P^{\prime}$ have $P \longmapsto_{l} a<c>\prec_{l}\left([(x, c)] \cdot P^{\prime}\right)$ by (simp add: Late-Semantics.alphaBoundResidual)
hence $P \longmapsto_{e} a<u>\prec_{e}\left([(x, c)] \cdot P^{\prime}\right)[c::=u]$ using cFresh $P$ by(rule Goal)
with $c$ Fresh $P^{\prime}$ show ?thesis by (simp add: renaming name-swap)
qed
lemma earlyLateInput:
fixes $P:: p i$
and $a$ :: name
and $x$ :: name
and $\quad P^{\prime}:: p i$
and $u$ :: name
and $C$ :: 'a::fs-name
assumes $P \longmapsto{ }_{e} a<u>\prec_{e} P^{\prime}$
and $\quad x \sharp P$
shows $\exists P^{\prime \prime} . P \longmapsto{ }_{l} a<x>\prec_{l} P^{\prime \prime} \wedge P^{\prime}=P^{\prime \prime}[x::=u]$
proof -
\{
fix $P$ a $u P^{\prime}$
assume $P \longmapsto_{e} a<u>\prec_{e} P^{\prime}$
hence $\exists P^{\prime \prime} x . P \longmapsto_{l} a<x>\prec_{l} P^{\prime \prime} \wedge P^{\prime}=P^{\prime \prime}[x::=u]$
proof (nominal-induct rule: Early-Semantics.inputInduct)
case (cInput a x Pu)
have $a<x>. P \longmapsto_{l} a<x>\prec P$ by(rule Late-Semantics.Input)
thus ?case by blast
next
case(cMatch Pau $\left.P^{\prime} b\right)$
have $\exists P^{\prime \prime} x . P \longmapsto_{l} a<x>\prec P^{\prime \prime} \wedge P^{\prime}=P^{\prime \prime}[x::=u]$ by fact
then obtain $P^{\prime \prime} x$ where PTrans: $P \longmapsto_{l} a<x>\prec P^{\prime \prime}$ and $P^{\prime} e q P^{\prime \prime}: P^{\prime}=$
$P^{\prime \prime}[x::=u]$ by blast
from PTrans have $[b \frown b] P \longmapsto_{l} a<x>\prec P^{\prime \prime} \mathbf{b y}$ (rule Late-Semantics.Match)
with $P^{\prime} e q P^{\prime \prime}$ show ?case by blast
next
case(cMismatch P a u $P^{\prime}$ b c)
have $\exists P^{\prime \prime} x . P \longmapsto_{l} a<x>\prec P^{\prime \prime} \wedge P^{\prime}=P^{\prime \prime}[x::=u]$ by fact
then obtain $P^{\prime \prime} x$ where PTrans: $P \longmapsto_{l} a<x>\prec P^{\prime \prime}$ and $P^{\prime} e q P^{\prime \prime}: P^{\prime}=$ $P^{\prime \prime}[x::=u]$ by blast
have $b \neq c$ by fact
with PTrans have $[b \neq c] P{ }_{l} a<x>\prec P^{\prime \prime}$ by (rule Late-Semantics.Mismatch) with $P^{\prime}$ eq $P^{\prime \prime}$ show ?case by blast
next
case (cSum1 $P$ a u $P^{\prime} Q$ )
have $\exists P^{\prime \prime} x . P \longmapsto_{l} a<x>\prec P^{\prime \prime} \wedge P^{\prime}=P^{\prime \prime}[x::=u]$ by fact
then obtain $P^{\prime \prime} x$ where PTrans: $P \longmapsto_{l} a<x>\prec P^{\prime \prime}$ and $P^{\prime} e q P^{\prime \prime}: P^{\prime}=$ $P^{\prime \prime}[x::=u]$ by blast
from PTrans have $P \oplus Q \longmapsto_{l} a<x>\prec P^{\prime \prime}$ by(rule Late-Semantics.Sum1)
with $P^{\prime} e q P^{\prime \prime}$ show? case by blast
next
case $\left(c S u m 2 ~ Q ~ a ~ u ~ Q^{\prime} P\right)$
have $\exists Q^{\prime \prime} x . Q \longmapsto_{l} a<x>\prec Q^{\prime \prime} \wedge Q^{\prime}=Q^{\prime \prime}[x::=u]$ by fact
then obtain $Q^{\prime \prime} x$ where $Q$ Trans: $Q \longmapsto_{l} a<x>\prec Q^{\prime \prime}$ and $Q^{\prime} e q Q^{\prime \prime}: Q^{\prime}=$ $Q^{\prime \prime}[x::=u]$ by blast
from $Q$ Trans have $P \oplus Q \longmapsto_{l} a<x>\prec Q^{\prime \prime}$ by(rule Late-Semantics.Sum2)
with $Q^{\prime} e q Q^{\prime \prime}$ show ? case by blast
next
case(cPar1 P a и $\left.P^{\prime} Q\right)$
have $\exists P^{\prime \prime} x . P \longmapsto_{l} a<x>\prec P^{\prime \prime} \wedge P^{\prime}=P^{\prime \prime}[x::=u]$ by fact
then obtain $P^{\prime \prime} x$ where PTrans: $P \longmapsto{ }_{l} a<x>\prec P^{\prime \prime}$ and $P^{\prime} e q P^{\prime \prime}: P^{\prime}=$ $P^{\prime \prime}[x::=u]$ by blast
have $\exists c:: n a m e . c \sharp\left(Q, P^{\prime \prime}\right) \mathbf{b y}($ blast intro: name-exists-fresh)
then obtain $c:$ :name where $c$ Fresh $Q: c \sharp Q$ and $c$ Fresh $P^{\prime \prime}: c \sharp P^{\prime \prime}$ by (force simp add: fresh-prod)
from PTrans cFresh $P^{\prime \prime}$ have $P \longmapsto_{l} a<c>\prec[(x, c)] \cdot P^{\prime \prime}$ by (simp add: Late-Semantics.alphaBoundResidual)
hence $\left.P \| Q \longmapsto_{l} a<c\right\rangle \prec\left([(x, c)] \cdot P^{\prime \prime}\right) \| Q$ using $\langle c \sharp Q\rangle$ by $($ rule Late-Semantics.Par1B)
moreover from $c$ Fresh $Q$ cFresh $P^{\prime \prime} P^{\prime}$ eq $P^{\prime \prime}$ have $P^{\prime} \| Q=\left(\left([(x, c)] \cdot P^{\prime \prime}\right)\right.$
$\| Q)[c::=u]$
by (simp add: forget renaming name-swap)
ultimately show ? case by blast
next
case(cPar2 $Q$ a u $Q^{\prime} P$ )
have $\exists Q^{\prime \prime} x . Q \longmapsto_{l} a<x>\prec Q^{\prime \prime} \wedge Q^{\prime}=Q^{\prime \prime}[x::=u]$ by fact
then obtain $Q^{\prime \prime} x$ where $Q$ Trans: $Q \longmapsto{ }_{l} a<x>\prec Q^{\prime \prime}$ and $Q^{\prime} e q Q^{\prime \prime}: Q^{\prime}=$ $Q^{\prime \prime}[x::=u]$ by blast
have $\exists c::$ name. $c \sharp\left(P, Q^{\prime \prime}\right) \mathbf{b y}$ (blast intro: name-exists-fresh)
then obtain $c::$ name where $c$ FreshP: $c \sharp P$ and $c$ Fresh $Q^{\prime \prime}: c \sharp Q^{\prime \prime}$ by (force simp add: fresh-prod)
from $Q$ Trans cFresh $Q^{\prime \prime}$ have $Q \longmapsto_{l} a<c>\prec[(x, c)] \cdot Q^{\prime \prime}$ by (simp add: Late-Semantics.alphaBoundResidual)
hence $P\left\|Q \longmapsto_{l} a<c>\prec P\right\|\left([(x, c)] \cdot Q^{\prime \prime}\right)$ using $\langle c \sharp P\rangle$ by (rule Late-Semantics.Par2B)
moreover from $c$ FreshP $c$ Fresh $Q^{\prime \prime} Q^{\prime} e q Q^{\prime \prime}$ have $P \| Q^{\prime}=(P \|([(x, c)] \cdot$ $\left.Q^{\prime \prime}\right)$ ) $[c::=u]$

```
            by(simp add: forget renaming name-swap)
            ultimately show ?case by blast
    next
    case(cRes P a u P'y)
    have }\exists\mp@subsup{P}{}{\prime\prime}x.P\mp@subsup{\longmapsto}{l}{}a<x>\prec\mp@subsup{P}{}{\prime\prime}\wedge\mp@subsup{P}{}{\prime}=\mp@subsup{P}{}{\prime\prime}[x::=u] by fac
```



```
P
    have yinequ: y}\not=u\mathrm{ by fact
    have \existsc::name.c\sharp(y, P'\prime) \mathbf{by}(blast intro: name-exists-fresh)
    then obtain c::name where cineqy:c\not=y and cFreshP}\mp@subsup{P}{}{\prime\prime}:c\sharp\mp@subsup{P}{}{\prime\prime}\mathrm{ by(force
simp add: fresh-prod)
    from PTrans cFreshP'\prime have }P\mp@subsup{\longmapsto}{l}{}a<c>< < [(x,c)] \cdot P'' by (simp add
Late-Semantics.alphaBoundResidual)
    moreover have }y\not=a\mathrm{ by fact
    ultimately have <\nuy>P \longmapsto\longmapstola<c><<<\nuy>(([(x,c)] \cdot P'\prime}))\mathrm{ using cineqy
            by(force intro: Late-Semantics.ResB)
    moreover from cineqy cFreshP'\prime P'eqP'\prime yinequ have <\nuy> ' ' = (<\nuy> ([(x,
c)] • P'\prime})[c::=u
            by(simp add: renaming name-swap)
        ultimately show ?case by blast
    next
        case(cBang P a u P')
        have }\exists\mp@subsup{P}{}{\prime\prime}x.P|!P\mp@subsup{\longmapsto}{l}{}a<x>\prec\mp@subsup{P}{}{\prime\prime}\wedge\mp@subsup{P}{}{\prime}=\mp@subsup{P}{}{\prime\prime}[x::=u] by fac
        then obtain }\mp@subsup{P}{}{\prime\prime}x\mathrm{ where PTrans: P|!P ط}\mp@subsup{l}{l}{}a<x>\prec\mp@subsup{P}{}{\prime\prime}\mathrm{ and P'eqP'': P'
= P'\prime[x::=u] by blast
    from PTrans have !P \longmapsto\longmapstolla<x> \prec P'' by(rule Late-Semantics.Bang)
    with P'eqP'/ show ?case by blast
    qed
}
with assms obtain }\mp@subsup{P}{}{\prime\prime}y\mathrm{ where PTrans: P}\mp@subsup{\longmapsto}{l}{}a<y>\prec\mp@subsup{P}{}{\prime\prime}\mathrm{ and P'eqP'/: P'
= P'I[y::=u] by blast
    show ?thesis
    proof(cases x=y)
        case True
        from PTrans P'eqP}\mp@subsup{P}{}{\prime\prime}\langlex=y\rangle\mathrm{ show ?thesis by blast
    next
        case False
        from PTrans }\langlex\not=y\rangle\langlex\sharpP\rangle\mathrm{ have }x\sharp\mp@subsup{P}{}{\prime\prime}\mathbf{by}(fastforce dest: freshBoundDeriva-
tive simp add: residual.inject)
    with PTrans have P\longmapsto\longmapstol}a<x>\mp@subsup{\prec}{l}{}([(x,y)]\cdot\mp@subsup{P}{}{\prime\prime}
            by(simp add: Late-Semantics.alphaBoundResidual)
            moreover from <x\sharp P ''> have }\mp@subsup{P}{}{\prime\prime}[y::=u]=([(x,y)] \cdot P ' ) [x::=u] by(sim
add: renaming name-swap)
    ultimately show ?thesis using }\mp@subsup{P}{}{\prime}eq\mp@subsup{P}{}{\prime\prime}\mathrm{ by blast
    qed
qed
lemma lateEarlyTau:
    fixes P :: pi
```

```
    and }\mp@subsup{P}{}{\prime}:: p
    assumes }P\longmapsto\mp@subsup{\longmapsto}{l}{}\tau\mp@subsup{\prec}{l}{}\mp@subsup{P}{}{\prime
    shows }P\longmapsto\mp@subsup{\longmapsto}{e}{}\tau\mp@subsup{\prec}{e}{}\mp@subsup{P}{}{\prime
using assms
proof(nominal-induct rule: Late-Semantics.tauInduct)
    case(Tau P)
    thus ?case by(rule Early-Semantics.Tau)
next
    case(Match P P' a)
    have P\longmapsto}\mp@subsup{\longmapsto}{e}{}\tau\mp@subsup{\prec}{e}{}\mp@subsup{P}{}{\prime}\mathrm{ by fact
    thus [a\frowna]P\longmapsto\longmapsto
next
    case(Mismatch P P' a b)
    have P}\longmapsto\mp@subsup{\longmapsto}{e}{}\tau\mp@subsup{\prec}{e}{}\mp@subsup{P}{}{\prime}\mathrm{ by fact
    moreover have a\not=b by fact
    ultimately show [a\not=b]P\longmapsto\longmapsto}\mp@subsup{\longmapsto}{e}{}\tau\mp@subsup{\prec}{e}{}\mp@subsup{P}{}{\prime}\mathrm{ by(rule Early-Semantics.Mismatch)
next
    case(Sum1 P P' Q)
    have P}\mp@subsup{\longmapsto}{e}{}\tau\mp@subsup{\prec}{e}{}\mp@subsup{P}{}{\prime}\mathrm{ by fact
    thus }P\oplusQ\longmapsto\mp@subsup{ط}{e}{}\tau\mp@subsup{\prec}{e}{}\mp@subsup{P}{}{\prime}\mathbf{by}(rule Early-Semantics.Sum1)
next
    case(Sum2 Q Q' P)
    have}Q\longmapsto\mp@subsup{\longmapsto}{e}{}\tau\mp@subsup{\prec}{e}{}\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    thus P}\oplusQ\longmapsto\mp@subsup{\longmapsto}{e}{}\tau\mp@subsup{\prec}{e}{}\mp@subsup{Q}{}{\prime}\mathbf{by}(rule Early-Semantics.Sum2)
next
    case(Par1 P P' Q)
    have }P\mp@subsup{\longmapsto}{e}{}\tau\mp@subsup{\prec}{e}{}\mp@subsup{P}{}{\prime}\mathrm{ by fact
    thus P|Q \longmapsto\longmapsto e}\tau<\mp@subsup{\prec}{e}{}\mp@subsup{P}{}{\prime}|Q\mathbf{by}(rule Early-Semantics.Par1F)
next
    case(Par2 Q Q' P)
    have }Q\mp@subsup{\longmapsto}{e}{}\tau\mp@subsup{\prec}{e}{}\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    thus }P|Q\longmapsto\mp@subsup{\longmapsto}{e}{}\tau\mp@subsup{\prec}{e}{}P|\mp@subsup{Q}{}{\prime}\mathbf{by}(\mathrm{ rule Early-Semantics.Par2F)
next
    case(Comm1 P a x P' Q b Q')
    have P \longmapsto\longmapsto }a<b>\mp@subsup{\prec}{e}{}\mp@subsup{P}{}{\prime}[x::=b
    proof -
        have }P\mp@subsup{\longmapsto}{l}{}a<x><\mp@subsup{P}{}{\prime}\mathrm{ by fact
        thus ?thesis by(rule lateEarlyInput)
    qed
    moreover have }Q\mp@subsup{\longmapsto}{e}{}a[b]\mp@subsup{\prec}{e}{}\mp@subsup{Q}{}{\prime
    proof -
        have }Q\mp@subsup{\longmapsto}{l}{}a[b]\mp@subsup{\prec}{l}{}\mp@subsup{Q}{}{\prime}\mathrm{ by fact
        thus ?thesis by(rule lateEarlyOutput)
    qed
    ultimately show ?case by(rule Early-Semantics.Comm1)
next
    case(Comm2 P a b P' Q x Q')
```

```
    have \(P \longmapsto_{e} a[b] \prec_{e} P^{\prime}\)
    proof -
        have \(P \longmapsto_{l} a[b] \prec_{l} P^{\prime}\) by fact
        thus ?thesis by(rule lateEarlyOutput)
    qed
    moreover have \(Q \longmapsto_{e} a<b>\prec_{e} Q^{\prime}[x::=b]\)
    proof -
        have \(Q \longmapsto_{l} a<x>\prec_{l} Q^{\prime}\) by fact
        thus ?thesis by (rule lateEarlyInput)
    qed
    ultimately show ?case by(rule Early-Semantics.Comm2)
next
    case(Close1 P a x \(P^{\prime} Q\) y \(Q^{\prime}\) )
    have \(P \longmapsto{ }_{e} a<y>\prec_{e} P^{\prime}[x::=y]\)
    proof -
        have \(P \longmapsto_{l} a<x>\prec P^{\prime}\) by fact
        thus ?thesis by (rule lateEarlyInput)
    qed
    moreover have \(Q \longmapsto_{e} a<\nu y>\prec Q^{\prime}\)
    proof -
        have \(Q \longmapsto_{l} a<\nu y>\prec_{l} Q^{\prime}\) by fact
        thus ?thesis by(rule lateEarlyBoundOutput)
    qed
    moreover have \(y \sharp P\) by fact
    ultimately show ?case by(rule Early-Semantics.Close1)
next
    case(Close2 P a y \(\left.P^{\prime} Q x Q^{\prime}\right)\)
    have \(P \longmapsto_{e} a<\nu y>\prec P^{\prime}\)
    proof -
        have \(P \longmapsto_{l} a<\nu y>\prec_{l} P^{\prime}\) by fact
        thus ?thesis by(rule lateEarlyBoundOutput)
    qed
    moreover have \(Q \longmapsto{ }_{e} a<y>\prec_{e} Q^{\prime}[x::=y]\)
    proof -
        have \(Q \longmapsto_{l} a<x>\prec_{l} Q^{\prime}\) by fact
        thus ?thesis by (rule lateEarlyInput)
    qed
    moreover have \(y \sharp Q\) by fact
    ultimately show ?case by(rule Early-Semantics.Close2)
next
    case(Res \(\left.P P^{\prime} x\right)\)
    have \(P \longmapsto_{e} \tau \prec_{e} P^{\prime}\) by fact
    thus ?case by (force intro: Early-Semantics.ResF)
next
    case (Bang \(P P^{\prime}\) )
    have \(P \|!P \longmapsto{ }_{e} \tau \prec_{e} P^{\prime}\) by fact
    thus ?case by(rule Early-Semantics.Bang)
qed
```

```
lemma earlyLateTau:
    fixes P :: pi
    and }\mp@subsup{P}{}{\prime}:: p
    assumes }P\longmapsto\mp@subsup{\longmapsto}{e}{}\tau\mp@subsup{\prec}{e}{}\mp@subsup{P}{}{\prime
    shows }P\longmapsto\longmapstoll < <l P'\
using assms
proof(nominal-induct rule: Early-Semantics.tauInduct)
    case(Tau P)
    thus ?case by(rule Late-Semantics.Tau)
next
    case(Match P P'a)
    have }P\longmapsto\mp@subsup{\longmapsto}{l}{}\tau\mp@subsup{\prec}{l}{}\mp@subsup{P}{}{\prime}\mathrm{ by fact
    thus ?case by(rule Late-Semantics.Match)
next
    case(Mismatch P P' a b)
    have }P\mp@subsup{\longmapsto}{l}{}\tau\mp@subsup{\prec}{l}{}\mp@subsup{P}{}{\prime}\mathrm{ by fact
    moreover have a\not=b by fact
    ultimately show ?case by(rule Late-Semantics.Mismatch)
next
    case(Sum1 P P' Q)
    have P}\longmapsto\mp@subsup{\longmapsto}{l}{}\tau\mp@subsup{\prec}{l}{}\mp@subsup{P}{}{\prime}\mathrm{ by fact
    thus ?case by(rule Late-Semantics.Sum1)
next
    case(Sum2 Q Q' P)
    have }Q\mp@subsup{\longmapsto}{l}{}\tau\mp@subsup{\prec}{l}{}\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    thus ?case by(rule Late-Semantics.Sum2)
next
    case(Par1 P P'Q)
    have P}\longmapsto\mp@subsup{\longmapsto}{l}{}\tau\mp@subsup{\prec}{l}{}\mp@subsup{P}{}{\prime}\mathrm{ by fact
    thus ?case by(rule Late-Semantics.Par1F)
next
    case(Par2 Q Q' P)
    have}Q\longmapstol \tau \prec. Q Q' by fac
    thus ?case by(rule Late-Semantics.Par2F)
next
    case(Comm1 P a b P' Q Q')
    have }P\mp@subsup{\longmapsto}{e}{}a<b>\mp@subsup{\prec}{e}{}\mp@subsup{P}{}{\prime}\mathrm{ by fact
    moreover obtain x::name where x\sharpP by(generate-fresh name) auto
    ultimately obtain }\mp@subsup{P}{}{\prime\prime}\mathrm{ where PTrans: }P\mp@subsup{\longmapsto}{l}{}a<x>\prec\mp@subsup{P}{}{\prime\prime}\mathrm{ and }\mp@subsup{P}{}{\prime}eq\mp@subsup{P}{}{\prime\prime}:\mp@subsup{P}{}{\prime}
P'\prime}[x::=b
    by(blast dest: earlyLateInput)
    have }Q\mp@subsup{\longmapsto}{e}{}a[b]\mp@subsup{\prec}{e}{}\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    hence }Q\mp@subsup{\longmapsto}{l}{}a[b]\mp@subsup{\prec}{l}{}\mp@subsup{Q}{}{\prime}\mathbf{by}(\mathrm{ rule earlyLateOutput)
    with PTrans P'eqP'\prime show ?case
        by(blast intro: Late-Semantics.Comm1)
next
    case(Comm2 P a b P'Q Q')
```

```
    have \(P \longmapsto_{e} a[b] \prec_{e} P^{\prime}\) by fact
    hence \(Q\) Trans: \(P \longmapsto_{l} a[b] \prec_{l} P^{\prime}\) by (rule earlyLateOutput)
    have \(Q \longmapsto_{e} a<b>\prec_{e} Q^{\prime}\) by fact
    moreover obtain \(x:\) :name where \(x \sharp Q \mathbf{b y}\) (generate-fresh name) auto
    ultimately obtain \(Q^{\prime \prime} x\) where \(Q \longmapsto_{l} a<x>\prec Q^{\prime \prime}\) and \(Q^{\prime}=Q^{\prime \prime}[x::=b]\)
    by (blast dest: earlyLateInput)
    with QTrans show ?case
    by (blast intro: Late-Semantics.Comm2)
next
    case(Close1 P a x \(\left.P^{\prime} Q Q^{\prime}\right)\)
    have \(P \longmapsto_{e} a<x>\prec_{e} P^{\prime}\) and \(x \sharp P\) by fact +
    then obtain \(P^{\prime \prime}\) where \(P \longmapsto_{l} a<x>\prec P^{\prime \prime}\) and \(P^{\prime}=P^{\prime \prime}[x::=x]\)
        by (blast dest: earlyLateInput)
    moreover have \(Q \longmapsto_{e} a<\nu x>\prec_{e} Q^{\prime}\) by fact
    hence \(Q \longmapsto_{l} a<\nu x>\prec_{l} Q^{\prime}\) by (rule earlyLateBoundOutput)
    moreover have \(x \sharp P\) by fact
    ultimately show ?case
    by (blast intro: Late-Semantics.Close1)
next
    case(Close2 P a x \(P^{\prime} Q Q^{\prime}\) )
    have \(P \longmapsto_{e} a<\nu x>\prec_{e} P^{\prime}\) by fact
    hence PTrans: \(P \longmapsto_{l} a<\nu x>\prec_{l} P^{\prime}\) by (rule earlyLateBoundOutput)
    have \(Q \longmapsto_{e} a<x>\prec_{e} Q^{\prime}\) and \(x \sharp Q\) by fact +
    then obtain \(Q^{\prime \prime} y\) where \(Q \longmapsto_{l} a<x>\prec Q^{\prime \prime}\) and \(Q^{\prime}=Q^{\prime \prime}[x::=x]\)
    by (blast dest: earlyLateInput)
    moreover have \(x \sharp Q\) by fact
    ultimately show ? case using PTrans
    by (blast intro: Late-Semantics.Close2)
next
    case \(\left(\right.\) Res \(\left.P P^{\prime} x\right)\)
    have \(P \longmapsto_{l} \tau \prec_{l} P^{\prime}\) by fact
    thus ?case by(force intro: Late-Semantics.ResF)
next
    case(Bang \(P P^{\prime}\) )
    have \(P \|!P \longmapsto_{l} \tau \prec_{l} P^{\prime}\) by fact
    thus ?case by(force intro: Late-Semantics.Bang)
qed
lemma \(t a u E q\) :
    fixes \(P:: p i\)
    and \(\quad P^{\prime}:: p i\)
    shows \(P \longmapsto_{e}\left(\right.\) Early-Semantics.FreeR Early-Semantics.TauR \(\left.P^{\prime}\right)=P \longmapsto \tau \prec_{l}\)
\(P^{\prime}\)
by(auto intro: earlyLateTau lateEarlyTau)
```

abbreviation simLate-judge $\left(-\rightsquigarrow_{l}[-]-[80,80,80] 80\right)$ where $P \rightsquigarrow_{l}[$ Rel $] Q \equiv$ Strong-Late-Sim.simulation P Rel $Q$
abbreviation simEarly-judge $\left(-\rightsquigarrow_{e}[-]-[80,80,80] 80\right)$ where $P \rightsquigarrow_{e}[\operatorname{Rel}] Q \equiv$ Strong-Early-Sim.strongSimEarly P Rel $Q$
lemma lateEarlySim:
fixes $P$ :: $p i$
and $\quad Q \quad:: p i$
and Rel $::(p i \times p i)$ set
assumes $\operatorname{PSimQ} Q \rightsquigarrow_{l}[$ Rel $] Q$
shows $P \rightsquigarrow_{e}[\operatorname{Rel}] Q$
proof (induct rule: Strong-Early-Sim.simCases)
case(Bound a x $Q^{\prime}$ )
have $Q \longmapsto_{e} a<\nu x>\prec_{e} Q^{\prime}$ by fact
hence $Q \longmapsto_{l} a<\nu x>\prec_{l} Q^{\prime}$ by (rule earlyLateBoundOutput)
moreover have $x \sharp P$ by fact
ultimately obtain $P^{\prime}$ where PTrans: $P \longmapsto_{l} a<\nu x>\prec_{l} P^{\prime}$ and $P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}\right.$, $\left.Q^{\prime}\right) \in$ Rel using PSimQ
by (force dest: Strong-Late-Sim.simE simp add: derivative-def)
from PTrans have $P \longmapsto_{e} a<\nu x>\prec_{e} P^{\prime}$ by (rule lateEarlyBoundOutput)
with $P^{\prime}$ RelQ' show ? case by blast
next
case $\left(\right.$ Free $\left.\alpha Q^{\prime}\right)$
have $Q \longmapsto_{e}$ Early-Semantics.residual.FreeR $\alpha Q^{\prime}$ by fact
thus? case
proof (nominal-induct $\alpha$ rule: freeRes.strong-induct)
case(InputR a u)
obtain $x$ ::name where $x \sharp Q$ and $x \sharp P$ by(generate-fresh name) auto
with $\left.\left\langle Q \longmapsto_{e} a<u\right\rangle \prec_{e} Q^{\prime}\right\rangle$ obtain $Q^{\prime \prime}$ where $Q$ Trans: $Q \longmapsto_{l} a<x>\prec_{l} Q^{\prime \prime}$
and $Q^{\prime} e q Q^{\prime \prime}: Q^{\prime}=Q^{\prime \prime}[x::=u]$
by(blast dest: earlyLateInput)
from PSimQ $Q$ Trans $\langle x \sharp P\rangle$ obtain $P^{\prime}$ where PTrans: $P \longmapsto_{l} a<x>\prec P^{\prime}$
and $P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}[x::=u], Q^{\prime \prime}[x::=u]\right) \in$ Rel
by(force dest: Strong-Late-Sim.simE simp add: derivative-def)
from PTrans have $P \longmapsto_{e} a<u>\prec_{e} P^{\prime}[x::=u]$ by (rule lateEarlyInput)
with $P^{\prime}$ Rel $Q^{\prime} Q^{\prime} e q Q^{\prime \prime}$ show $\exists P^{\prime} . P \longmapsto{ }_{e} a<u>\prec_{e} P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in$ Rel by
blast
next
case(OutputR ab)
from $\left\langle Q \longmapsto_{e} a[b] \prec_{e} Q^{\prime}\right\rangle$ have $Q \longmapsto_{l} a[b] \prec_{l} Q^{\prime}$ by (rule earlyLateOutput)
with $P \operatorname{Sim} Q$ obtain $P^{\prime}$ where PTrans: $P \longmapsto_{l} a[b] \prec_{l} P^{\prime}$ and $P^{\prime} \operatorname{Rel} Q^{\prime}:\left(P^{\prime}\right.$,
$\left.Q^{\prime}\right) \in \operatorname{Rel}$
by (blast dest: Strong-Late-Sim.simE)
from PTrans have $P \longmapsto_{e} a[b] \prec_{e} P^{\prime}$ by (rule lateEarlyOutput)
with $P^{\prime}$ RelQ' show $\exists P^{\prime} . P \longmapsto_{e} a[b] \prec_{e} P^{\prime} \wedge\left(P^{\prime}, Q^{\prime}\right) \in$ Rel by blast
next

```
    case TauR
    from \Q \longmapsto\longmapsto }\mp@subsup{e}{e}{}\tau\mp@subsup{\prec}{e}{}\mp@subsup{Q}{}{\prime}\rangle\mathrm{ have }Q\mp@subsup{\longmapsto}{l}{}\tau\mp@subsup{\prec}{l}{}\mp@subsup{Q}{}{\prime}\mathrm{ by(rule earlyLateTau)
    with PSimQ obtain P' where PTrans: }P\mp@subsup{\longmapsto}{l}{}\tau\mp@subsup{\prec}{l}{}\mp@subsup{P}{}{\prime}\mathrm{ and P'RelQ':}(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime}
Rel
    by(blast dest: Strong-Late-Sim.simE)
    from PTrans have P\longmapsto\longmapsto }\mp@subsup{\longmapsto}{e}{}\tau\mp@subsup{\prec}{e}{}\mp@subsup{P}{}{\prime}\mathrm{ by(rule lateEarlyTau)
    with P'RelQ' show }\exists\mp@subsup{P}{}{\prime}.P\longmapsto\mp@subsup{\longmapsto}{e}{}\tau\mp@subsup{\prec}{e}{}\mp@subsup{P}{}{\prime}\wedge(\mp@subsup{P}{}{\prime},\mp@subsup{Q}{}{\prime})\in\mathrm{ Rel by blast
    qed
qed
```

abbreviation bisimLate-judge $\left(-\sim_{l}-[80,80] 80\right)$ where $P \sim_{l} Q \equiv(P, Q) \in$ Strong-Late-Bisim.bisim
abbreviation bisimEarly-judge $\left(-\sim_{e}-[80,80] 80\right)$ where $P \sim_{e} Q \equiv(P, Q) \in$ Strong-Early-Bisim.bisim
lemma lateEarlyBisim:
fixes $P:: p i$
and $\quad Q:: p i$
assumes $P \sim_{l} Q$
shows $P \sim_{e} Q$
using assms
by (coinduct rule: Strong-Early-Bisim.weak-coinduct)
(auto dest: Strong-Late-Bisim.bisimE Strong-Late-Bisim.symmetric intro: lateEarlySim)
abbreviation congLate-judge $\left(-\sim^{s}{ }_{l}-[80,80] 80\right)$ where $P \sim^{s}{ }_{l} Q \equiv(P, Q) \in$ (substClosed Strong-Late-Bisim.bisim)
abbreviation congEarly-judge $\left(-\sim^{s}{ }_{e}-[80,80] 80\right)$ where $P \sim^{s}{ }_{e} Q \equiv(P, Q) \in$ (substClosed Strong-Early-Bisim.bisim)
lemma lateEarlyCong:
fixes $P$ :: $p i$
and $\quad Q:: p i$
assumes $P \sim^{s}{ }_{l} Q$
shows $P \sim^{s}{ }_{e} Q$
using assms
by (auto simp add: substClosed-def intro: lateEarlyBisim)
lemma earlyCongStructCong:
fixes $P:: p i$
and $\quad Q:: p i$
assumes $P \equiv{ }_{s} Q$
shows $P \sim^{s}{ }_{e} Q$
using assms lateEarlyCong bisimSubstStructCong by blast
lemma earlyBisimStructCong:
fixes $P:: p i$
and $\quad Q:: p i$
assumes $P \equiv_{s} Q$
shows $P \sim_{e} Q$
using assms lateEarlyBisim structCongBisim
by blast
end
theory Strong-Early-Bisim-SC
imports Strong-Early-Bisim Strong-Late-Bisim-SC Strong-Early-Late-Comp begin

```
lemma resComm:
    fixes }P::p
    shows <\nu a><\nub>P ~}\mp@subsup{~}{e}{<\nub><\nua>P
proof -
    have <\nua><\nub>P ~}\mp@subsup{~}{l}{<\nub><\nua>P by(rule Strong-Late-Bisim-SC.resComm)
    thus?thesis by(rule lateEarlyBisim)
qed
```

lemma matchId:
fixes $a$ :: name
and $P:: p i$
shows $[a \frown a] P \sim_{e} P$
proof -
have $[a \frown a] P \sim_{l} P$ by (rule Strong-Late-Bisim-SC.matchId)
thus? ?thesis by(rule lateEarlyBisim)
qed

```
lemma mismatchId:
    fixes a :: name
    and b:: name
    and }P:: p
    assumes }a\not=
    shows [a\not=b]P ~ ~e}
proof -
    from assms have [a\not=b]P ~}\mp@subsup{~}{l}{}P\mathrm{ by(rule Strong-Late-Bisim-SC.mismatchId)
    thus?thesis by(rule lateEarlyBisim)
qed
lemma mismatchNil:
    fixes a :: name
    and }P::p
    shows [a\not=a]P ~e}\mathbf{0
proof -
    have [a\not=a]P\mp@subsup{~}{l}{}\mathbf{0}\mathrm{ by(rule Strong-Late-Bisim-SC.mismatchNil)}
    thus ?thesis by(rule lateEarlyBisim)
qed
lemma sumSym:
    fixes }P::p
    and }Q::p
    shows }P\oplusQ\mp@subsup{~}{e}{}Q\oplus
proof -
    have P}\oplusQ\mp@subsup{~}{l}{}Q\oplusP\mathbf{by}(rule Strong-Late-Bisim-SC.sumSym)
    thus?thesis by(rule lateEarlyBisim)
qed
lemma sumAssoc:
    fixes P :: pi
    and }Q::p
    and }R::p
    shows }(P\oplusQ)\oplusR\mp@subsup{~}{e}{}P\oplus(Q\oplusR
proof -
    have }(P\oplusQ)\oplusR\mp@subsup{~}{l}{}P\oplus(Q\oplusR)\mathbf{by(rule Strong-Late-Bisim-SC.sumAssoc)
    thus?thesis by(rule lateEarlyBisim)
qed
lemma sumZero:
```

```
    fixes P :: pi
    shows }P\oplus\mathbf{0}~\mp@subsup{~}{e}{}
proof -
    have }P\oplus\mathbf{0}\mp@subsup{~}{l}{l}P\mathrm{ by(rule Strong-Late-Bisim-SC.sumZero)
    thus?thesis by(rule lateEarlyBisim)
qed
lemma parZero:
    fixes P :: pi
    shows P|0 0 ~
proof -
    have P|00 ~
    thus ?thesis by(rule lateEarlyBisim)
qed
lemma parSym:
    fixes }P:: p
    and }Q:: p
    shows P|Q ~
proof -
    have P|Q ~}|Q|P\mathrm{ by(rule Strong-Late-Bisim-SC.parSym)
    thus ?thesis by(rule lateEarlyBisim)
qed
lemma scopeExtPar:
    fixes P :: pi
    and }Q:: p
    and x :: name
    assumes }x\sharp
    shows <\nux>(P|Q) ~
proof -
    from assms have <\nux>(P|Q) ~
    thus?thesis by(rule lateEarlyBisim)
qed
lemma scopeExtPar':
    fixes P :: pi
    and }Q::p
    and x :: name
    assumes xFreshQ:x\sharpQ
```

```
    shows <\nux> (P|Q) ~
proof -
    from assms have <\nux>(P|Q) ~
    thus?thesis by(rule lateEarlyBisim)
qed
lemma parAssoc:
    fixes P :: pi
    and }Q::p
    and }R::p
    shows }(P|Q)|R\mp@subsup{~}{e}{}P|(Q|R
proof -
    have (P|Q)|R\mp@subsup{~}{l}{}P|(Q|R)\mathbf{by}(rule Strong-Late-Bisim-SC.parAssoc)
    thus ?thesis by(rule lateEarlyBisim)
qed
lemma freshRes:
    fixes P :: pi
    and a :: name
    assumes aFreshP: a\sharpP
    shows <\nua>P ~
proof -
    from aFreshP have <\nua>P \mp@subsup{~}{l}{}P}\mathrm{ Py(rule Strong-Late-Bisim-SC.scopeFresh)
    thus?thesis by(rule lateEarlyBisim)
qed
lemma scopeExtSum:
    fixes }P:: p
    and }Q:: p
    and x :: name
    assumes }x\sharp
    shows <\nux>}(P\oplusQ)\mp@subsup{~}{e}{}P\oplus<\nux>
proof -
    from \langlex\sharpP> have <\nux>(P\oplusQ) ~
    thus?thesis by(rule lateEarlyBisim)
qed
lemma bangSC:
    fixes P
    shows !P ~}\mp@subsup{~}{e}{}P|!
proof -
    have !P ~}\mp@subsup{~}{l}{}P|!P\mathrm{ by(rule Strong-Late-Bisim-SC.bangSC)
    thus?thesis by(rule lateEarlyBisim)
```

```
qed
end
theory Weak-Early-Bisim-SC
    imports Weak-Early-Bisim Strong-Early-Bisim-SC
begin
lemma weakBisimStructCong:
    fixes P :: pi
    and }Q::p
    assumes P}\mp@subsup{\equiv}{s}{}
    shows P}\approx
using assms
by(metis earlyBisimStructCong strongBisimWeakBisim)
lemma matchId:
    fixes a :: name
    and }P:: p
    shows [a\frowna]P\approxP
proof -
    have [a\frowna]P ~
    thus?thesis by(rule strongBisim WeakBisim)
qed
lemma mismatchId:
    fixes a :: name
    and b:: name
    and }P::p
    assumes }a\not=
    shows [a\not=b]P\approxP
proof -
    from }\langlea\not=b\rangle\mathrm{ have [aキb]P 埌P by(rule Strong-Early-Bisim-SC.mismatchId)
    thus?thesis by(rule strongBisim WeakBisim)
qed
lemma mismatchNil:
    fixes a :: name
    and }P::p
    shows [a\not=a]P\approx0
proof -
```

```
    have [a\not=a]P ~}\mp@subsup{~}{e}{}\mathbf{0}\mathrm{ by(rule Strong-Early-Bisim-SC.mismatchNil)
    thus?thesis by(rule strongBisim WeakBisim)
qed
```

```
lemma resComm:
```

lemma resComm:
fixes P :: pi
fixes P :: pi
shows <\nua>< <\nub>P}\approx<\nub><\nu\nua>
shows <\nua>< <\nub>P}\approx<\nub><\nu\nua>
proof -
proof -
have <\nua><\nub>P\mp@subsup{~}{e}{}<\nub><\nua>P by(rule Strong-Early-Bisim-SC.resComm)
have <\nua><\nub>P\mp@subsup{~}{e}{}<\nub><\nua>P by(rule Strong-Early-Bisim-SC.resComm)
thus?thesis by(rule strongBisim WeakBisim)
thus?thesis by(rule strongBisim WeakBisim)
qed

```
qed
```

lemma sumSym:
fixes $P:: p i$
and $\quad Q:: p i$
shows $P \oplus Q \approx Q \oplus P$
proof -
have $P \oplus Q \sim_{e} Q \oplus P \mathbf{b y}($ rule Strong-Early-Bisim-SC.sumSym)
thus ?thesis $\mathbf{b y}$ (rule strongBisim WeakBisim)
qed
lemma sumAssoc:
fixes $P:: p i$
and $\quad Q:: p i$
and $\quad R:: p i$
shows $(P \oplus Q) \oplus R \approx P \oplus(Q \oplus R)$
proof -
have $(P \oplus Q) \oplus R \sim_{e} P \oplus(Q \oplus R)$ by (rule Strong-Early-Bisim-SC.sumAssoc)
thus ?thesis by (rule strongBisim WeakBisim)
qed
lemma sumZero:
fixes $P$ :: pi
shows $P \oplus \mathbf{0} \approx P$
proof -
have $P \oplus \mathbf{0} \sim_{e} P$ by(rule Strong-Early-Bisim-SC.sumZero)
thus?thesis by (rule strongBisim WeakBisim)
qed

```
lemma parZero:
    fixes P :: pi
    shows P|0 0 F P
proof -
    have P|\mathbf{0 ~}
    thus?thesis by(rule strongBisim WeakBisim)
qed
lemma parSym:
    fixes P :: pi
    and }Q::p
    shows }P|Q\approxQ|
proof -
    have }P||\mp@subsup{~}{e}{}Q|P\mathrm{ by(rule Strong-Early-Bisim-SC.parSym)
    thus ?thesis by(rule strongBisim WeakBisim)
qed
lemma scopeExtPar:
    fixes }P\mathrm{ :: pi
    and }Q:: p
    and x :: name
    assumes }x\sharp
    shows <\nux>(P|Q)\approxP|<\nux>Q
proof -
    from \langlex\sharpP> have <\nux> (P|Q) ~
    thus ?thesis by(rule strongBisim WeakBisim)
qed
lemma scopeExtPar':
    fixes }P\mathrm{ :: pi
    and }Q::p
    and x :: name
    assumes }x\sharp
    shows <\nux>(P|Q)\approx(<\nux>P)|Q
proof -
    from <x\sharpQ> have <\nux> (P|Q) ~
    thus ?thesis by(rule strongBisim WeakBisim)
qed
lemma parAssoc:
    fixes P :: pi
    and }\quadQ::p
    and }R:: p
```

```
    shows (P|Q)|R\approxP|(Q|R)
proof -
    have (P|Q)|R ~
    thus?thesis by(rule strongBisim WeakBisim)
qed
lemma freshRes:
    fixes P :: pi
    and a :: name
    assumes }a\sharp
    shows <\nua>P}\approx
proof -
    from \langlea\sharpP> have <\nua>P ~}\mp@subsup{~}{e}{}P\mathrm{ by(rule Strong-Early-Bisim-SC.freshRes)
    thus ?thesis by(rule strongBisim WeakBisim)
qed
lemma scopeExtSum:
    fixes P :: pi
    and }Q:: p
    and x :: name
    assumes }x\sharp
    shows <\nux>(P\oplusQ)\approxP\oplus<\nux>Q
proof -
    from \langlex\sharpP\rangle}\mathrm{ have }<\nux>(P\oplusQ)~\mp@subsup{~}{e}{}P\oplus<\nux>Q by(rule Strong-Early-Bisim-SC.scopeExtSum)
    thus ?thesis by(rule strongBisim WeakBisim)
qed
lemma bangSC:
    fixes P
    shows !P\approxP|!P
proof -
    have !P ~}\mp@subsup{~}{e}{}P|!P\mathrm{ by(rule Strong-Early-Bisim-SC.bangSC)
    thus ?thesis by(rule strongBisim WeakBisim)
qed
end
theory Weak-Early-Bisim-Pres
    imports Strong-Early-Bisim-Pres Weak-Early-Sim-Pres Weak-Early-Bisim-SC
Weak-Early-Bisim
begin
```

```
lemma tauPres:
    fixes }P::p
    and }Q::p
    assumes P\approxQ
    shows }\tau.(P)\approx\tau.(Q
proof -
    let ?X = {(\tau.(P),\tau.(Q))|PQ.P\approxQ}
    from }\langleP\approxQ\rangle\mathrm{ have ( }\tau.(P),\tau.(Q))\in?X by aut
    thus ?thesis
    proof(coinduct rule: weakBisimCoinduct)
        case(cSim P Q)
        thus ?case
            by(force intro:Weak-Early-Sim-Pres.tauPres)
    next
        case(cSym P Q)
        thus ?case by(force dest:Weak-Early-Bisim.symetric simp add: pi.inject)
    qed
qed
lemma outputPres:
    fixes P :: pi
    and }Q::p
    and a :: name
    and b :: name
    assumes P}\approx
    shows }a{b}.P\approxa{b}.
proof -
    let ?X = {(a{b}.(P), a{b}.(Q))|PQ a b. P\approxQ}
```



```
    thus ?thesis
    proof(coinduct rule: weakBisimCoinduct)
        case(cSim P Q)
        thus ?case
            by(force intro:Weak-Early-Sim-Pres.outputPres)
    next
        case(cSym P Q)
        thus ?case by(force dest:Weak-Early-Bisim.symetric simp add: pi.inject)
    qed
qed
lemma inputPres:
    fixes P :: pi
    and }Q::p
    and a :: name
```

and $\quad x::$ name
assumes $P \operatorname{Sim} Q: \forall y . P[x::=y] \approx Q[x::=y]$
shows $a<x>. P \approx a<x>. Q$
proof -
let ? $X=\{(a<x>. P, a<x>. Q) \mid$ a $x P Q . \forall y . P[x::=y] \approx Q[x::=y]\}$
\{
fix $a x P a x Q p$
assume $(a x P, a x Q) \in ? X$
then obtain $a x P Q$ where $A: \forall y . P[x::=y] \approx Q[x::=y]$ and $B: a x P=$
$a<x>. P$ and $C: a x Q=a<x>. Q$
by auto
have $\bigwedge y \cdot((p::$ name prm $) \cdot P)[(p \cdot x)::=y] \approx(p \cdot Q)[(p \cdot x)::=y]$
proof -
fix $y$
from $A$ have $P[x::=($ rev $p \cdot y)] \approx Q[x::=($ rev $p \cdot y)]$
by blast
hence $(p \cdot(P[x::=(\operatorname{rev} p \cdot y)])) \approx p \cdot(Q[x::=(\operatorname{rev} p \cdot y)])$ by (rule eqvts)
thus $(p \cdot P)[(p \cdot x)::=y] \approx(p \cdot Q)[(p \cdot x)::=y]$
by (simp add: eqvts pt-pi-rev[OF pt-name-inst, OF at-name-inst])
qed
hence $((p::$ name prm) $\operatorname{axP}, p \cdot a x Q) \in$ ? $X$ using $B C$
by auto
\}
hence eqvt? $X$ by (simp add: eqvt-def)
from PSim $Q$ have $(a<x>. P, a<x>. Q) \in ? X$ by auto
thus ?thesis
proof (coinduct rule: weakBisimCoinduct)
case $(c \operatorname{Sim} P Q)$
thus ?case using 〈eqvt? $X$ 〉
by(force intro: Weak-Early-Sim-Pres.inputPres)
next
case $(c \operatorname{Sym} P Q)$
thus ?case
by (blast dest: weakBisimE)
qed
qed
lemma resPres:
fixes $P$ :: $p i$
and $\quad Q:: p i$
and $x::$ name
assumes $P \approx Q$
shows $<\nu x>P \approx<\nu x>Q$

```
proof -
    let ?X = {(<\nux>P,<\nux>Q) |xPQ.P\approxQ}
    from }\langleP\approxQ\rangle\mathrm{ have (< < x>>P,< 侪>Q) &?X by blast
    thus ?thesis
    proof(coinduct rule: weakBisimCoinduct)
        case(cSim xP xQ)
        {
            fix PQx
            assume P}\approx
            hence P\rightsquigarrow<weakBisim> Q by(rule weakBisimE)
            moreover have }\PQx.P\approxQ\Longrightarrow(<\nux>P,<\nux>Q)\in?X\cup\mathrm{ weakBisim
by blast
            moreover have weakBisim \subseteq?X U weakBisim by blast
            moreover have eqvt weakBisim by simp
            moreover have eqvt (?X \cup weakBisim)
                by(auto simp add: eqvt-def dest:Weak-Early-Bisim.eqvtI)+
            ultimately have <\nux>P}\rightsquigarrow<(?X\cup\mathrm{ weakBisim )}><\nux>
                    by(rule Weak-Early-Sim-Pres.resPres)
        }
        with}\langle(xP,xQ)\in?X>\mathrm{ show ?case by blast
    next
            case(cSym xP xQ)
            thus ?case by(blast dest:Weak-Early-Bisim.symetric)
    qed
qed
lemma matchPres:
    fixes P :: pi
    and }Q::p
    and a :: name
    and b :: name
    assumes P\approxQ
    shows [a\frownb]P\approx[a\frownb]Q
proof -
    let ?X = {([a\frownb]P,[a\frownb]Q)| a b P Q. P \approx Q }
    from }\langleP\approxQ\rangle\mathrm{ have ([a乞b]P,[a乞b]Q) &?X by blast
    thus ?thesis
    proof(coinduct rule: weakBisimCoinduct)
    case(cSim abPabQ)
    {
        fix PQab
        assume P}\approx
        hence }P\rightsquigarrow<\mathrm{ weakBisim> Q by(rule weakBisimE)
        moreover have weakBisim \subseteq(?X\cup weakBisim) by blast
        moreover have }\PQa.P\approxQ\Longrightarrow[a\frowna]P\approx
        by (metis (full-types) strongBisimWeakBisim Strong-Early-Bisim-SC.matchId
Weak-Early-Bisim.transitive)
```

```
            ultimately have [a\frownb]P\rightsquigarrow<(?X ~ weakBisim )}>[a\frownb]
            by(rule Weak-Early-Sim-Pres.matchPres)
    }
    with }\langle(abP,abQ)\in?\\rangle\mathrm{ show ?case by blast
    next
    case(cSym abP abQ)
    thus ?case by(blast dest:Weak-Early-Bisim.symetric)
    qed
qed
lemma mismatchPres:
    fixes }P:: p
    and }Q::p
    and a :: name
    and b:: name
    assumes P}\approx
    shows [a\not=b]P\approx[a\not=b]Q
proof -
    let ?X = {([a\not=b]P,[a\not=b]Q)| ab P Q. P\approxQ}
    from }\langleP\approxQ\rangle\mathrm{ have ([a*b]P,[aキb]Q) & ?X by blast
    thus ?thesis
    proof(coinduct rule: weakBisimCoinduct)
    case(cSim abP abQ)
    {
        fix PQab
        assume P\approxQ
        hence P}\rightsquigarrow<\mathrm{ weakBisim> Q by(rule weakBisimE)
        moreover have weakBisim \subseteq(?X\cup weakBisim) by blast
        moreover have \}\PQab.\llbracketP\approxQ;a\not=b\rrbracket\Longrightarrow[a\not=b]P\approx
        by (metis (full-types) strongBisimWeakBisim Strong-Early-Bisim-SC.mismatchId
Weak-Early-Bisim.transitive)
            ultimately have }[a\not=b]P\rightsquigarrow<(?X\cup\mathrm{ weakBisim )}>[a\not=b]
            by(rule Weak-Early-Sim-Pres.mismatchPres)
        }
        with }\langle(abP,abQ)\in?X\rangle show ?case by blas
    next
        case(cSym abP abQ)
        thus ?case by(blast dest:Weak-Early-Bisim.symetric)
    qed
qed
lemma parPres:
    fixes P :: pi
    and }Q::p
    and }R::p
    assumes P\approxQ
```

```
    shows }P|R\approxQ|
proof -
    let ?X = {(resChain lst (P|R), resChain lst (Q|R))| lst P R Q. P\approxQ}
    have BC: \bigwedgePQ.P|Q= resChain [] (P|Q) by auto
    from }\langleP\approxQ\rangle\mathrm{ have (P|R,Q|R) &?X by(blast intro: BC)
    thus ?thesis
    proof(coinduct rule: weakBisimCoinduct)
        case(cSym PR QR)
        {
            fix P Q R lst
            assume P\approxQ
            moreover hence P}\rightsquigarrow<\mathrm{ weakBisim> Q by(rule weakBisimE)
            moreover have }\PQR.P\approxQ\Longrightarrow(P|R,Q|R)\in?X using B
                by blast
            moreover {
                fix PR QR x
                    assume (PR,QR)\in?X
            then obtain lst PQ R where P}\approxQ\mathrm{ and A:PR= resChain lst (P|R)
and B:QR= resChain lst (Q|R)
                by auto
            from A have <\nux>PR = resChain (x#lst) (P|R) by auto
            moreover from B have <\nux>QR = resChain (x#lst) (Q|R) by auto
            ultimately have ( <\nux>>PR,<\nux>>QR)\in?X using <P\approxQ>
                by blast
            }
            note Res = this
            ultimately have }P|R\rightsquigarrow<?X>Q|
                    by(rule-tac Weak-Early-Sim-Pres.parPres)
            moreover have eqvt?X
                by(auto simp add: eqvt-def) (blast intro: eqvts)
                            ultimately have resChain lst ( }P|R)\rightsquigarrow<?X> resChain lst ( Q|R) usin
Res
            by(rule-tac Weak-Early-Sim-Pres.resChainI)
            hence resChain lst ( }P|R)\rightsquigarrow<(?X\cup\mathrm{ weakBisim )> resChain lst ( }Q|R
                by(force intro:Weak-Early-Sim.monotonic)
        }
            with}\langle(PR,QR)\in?X>\mathrm{ show }PR\rightsquigarrow<(?X\cup\mathrm{ weakBisim )}>Q
            by blast
    next
        case(cSym PR QR)
        thus ?case by(blast dest:Weak-Early-Bisim.symetric)
    qed
qed
lemma bangPres:
    fixes P :: pi
    and }Q:: p
```

assumes $\operatorname{PBisim} Q: P \approx Q$
shows $!P \approx!Q$
proof -
let ? $X=($ bangRel weakBisim $)$
let?Y = Strong-Early-Bisim.bisim O (bangRel weakBisim) O Strong-Early-Bisim.bisim
from Weak-Early-Bisim.eqvt Strong-Early-Bisim.eqvt have eqvt $Y$ : eqvt? $Y$ by (blast intro: eqvtBangRel)
have $X$ sub $Y: ? X \subseteq$ ?Y by (auto intro: Strong-Early-Bisim.reflexive)

```
    have RelStay: \(\wedge P Q .(P \|!P, Q) \in ? Y \Longrightarrow(!P, Q) \in ? Y\)
    proof (auto)
        fix \(P Q R T\)
        assume \(\operatorname{PBisim} Q: P \|!P \sim_{e} Q\)
            and \(\operatorname{QBRR}:(Q, R) \in\) bangRel weakBisim
            and RBisimT: \(R \sim_{e} T\)
    have \(!P \sim_{e} Q\)
    proof -
                have \(!P \sim_{e} P \|!P\) by (rule Strong-Early-Bisim-SC.bangSC)
                thus ?thesis using PBisimQ by(rule Strong-Early-Bisim.transitive)
    qed
    with \(Q B R R\) RBisim \(T\) show \((!P, T) \in ? Y\) by blast
    qed
    have ParCompose: \(\bigwedge P Q R T . \llbracket P \approx Q ;(R, T) \in ? Y \rrbracket \Longrightarrow(P\|R, Q\| T) \in\)
?Y
    proof -
        fix \(P Q R T\)
        assume \(P\) Bisim \(Q: P \approx Q\)
            and RYT: \(\quad(R, T) \in ? Y\)
        thus \((P\|R, Q\| T) \in ? Y\)
        proof (auto)
            fix \(T^{\prime} R^{\prime}\)
            assume \(T^{\prime} B i s i m T: T^{\prime} \sim_{e} T\) and \(R B i s i m R^{\prime}: R \sim_{e} R^{\prime}\)
                and \(R^{\prime} B R T^{\prime}:\left(R^{\prime}, T^{\prime}\right) \in\) bangRel weakBisim
        have \(P\left\|R \sim_{e} P\right\| R^{\prime}\)
        proof -
        from RBisimR' have \(R\left\|P \sim_{e} R^{\prime}\right\| P\) by(rule Strong-Early-Bisim-Pres.parPres)
                moreover have \(P\left\|R \sim_{e} R\right\| P\) and \(R^{\prime}\left\|P \sim_{e} P\right\| R^{\prime}\) by(rule
Strong-Early-Bisim-SC.parSym)+
            ultimately show ?thesis by (blast intro: Strong-Early-Bisim.transitive)
            qed
            moreover from \(P\) Bisim \(Q R^{\prime} B R T^{\prime}\) have \(\left(P\left\|R^{\prime}, Q\right\| T^{\prime}\right) \in\) bangRel weakBisim
by (rule BRPar)
            moreover have \(Q\left\|T^{\prime} \sim_{e} Q\right\| T\)
            proof -
            from \(T^{\prime}\) BisimT have \(T^{\prime}\left\|Q \sim_{e} T\right\| Q \mathbf{b y}\) (rule Strong-Early-Bisim-Pres.parPres)
                moreover have \(Q\left\|T^{\prime} \sim_{e} T^{\prime}\right\| Q\) and \(T\left\|Q \sim_{e} Q\right\| T\) by(rule
```

```
Strong-Early-Bisim-SC.parSym)+
            ultimately show ?thesis by(blast intro: Strong-Early-Bisim.transitive)
        qed
        ultimately show ?thesis by blast
    qed
qed
have ResCong: }\PQx.(P,Q)\in?Y\Longrightarrow(<\nux>P,<\nux>Q)\in?
    by(auto intro: BRRes Strong-Early-Bisim-Pres.resPres transitive)
have Sim: }\PQ.(P,Q)\in?X\LongrightarrowP\rightsquigarrow<?Y>
proof -
    fix PQ
    assume (P,Q)\in?X
    thus P\rightsquigarrow<?Y>Q
    proof(induct)
        case(BRBang P Q)
        have P}\approxQ\mathrm{ by fact
        moreover hence P}\rightsquigarrow<\mathrm{ weakBisim> Q by(blast dest:weakBisimE)
            moreover have }\PQ.P\approxQ\LongrightarrowP\rightsquigarrow<\mathrm{ weakBisim> Q by(blast dest:
weakBisimE)
            moreover from Strong-Early-Bisim.eqvt Weak-Early-Bisim.eqvt have eqvt
?Y by(blast intro: eqvtBangRel)
    ultimately show !P \rightsquigarrow<?Y>!Q using ParCompose ResCong RelStay XsubY
                by(rule-tac Weak-Early-Sim-Pres.bangPres, simp-all)
    next
        case(BRPar P Q R T)
        have PBiSimQ: P}\approxQ\mathrm{ by fact
        moreover have RBangRelT: (R,T)\in?X by fact
        have RSimT: R\rightsquigarrow<?Y>T by fact
        moreover from PBiSimQ have P\rightsquigarrow<weakBisim> Q by(blast dest:weak-
BisimE)
    moreover from RBangRelT have (R,T) \in?Y by(blast intro: Strong-Early-Bisim.reflexive)
        ultimately show P|R\rightsquigarrow<?Y>Q|T using ParCompose ResCong eqvt
eqvt Y
            by(rule-tac Weak-Early-Sim-Pres.parCompose)
        next
            case(BRRes P Q x)
            have }P\rightsquigarrow<?Y>Q by fac
            thus <\nux>P \rightsquigarrow<?Y><\nux>Q using ResCong eqvtY XsubY
            by(rule-tac Weak-Early-Sim-Pres.resPres, simp-all)
        qed
    qed
    from PBisimQ have (!P,!Q) \in?X by(rule BRBang)
    moreover from Weak-Early-Bisim.eqvt have eqvt (bangRel weakBisim) by(rule
eqvtBangRel)
    ultimately show ?thesis
```

```
        apply(coinduct rule:Weak-Early-Bisim.transitive-coinduct-weak)
        apply(blast intro: Sim)
    by(blast dest: bangRelSymetric Weak-Early-Bisim.symetric intro: Strong-Early-Bisim.reflexive)
qed
lemma bangRelSub WeakBisim:
    shows bangRel weakBisim \subseteq weakBisim
proof(auto)
    fix ab
    assume (a,b) \in bangRel weakBisim
    thus a\approxb
    proof(induct)
        fix PQ
        assume P}\approx
        thus !P\approx!Q by(rule bangPres)
    next
        fix PQRT
        assume R}\approxT\mathrm{ and }P\approx
        thus R|P\approxT|Q by(metis parPres parSym Weak-Early-Bisim.transitive)
    next
        fix PQ
        fix a::name
        assume P\approxQ
        thus <\nua>P}\approx<\nua>Q by(rule resPres
    qed
qed
end
theory Weak-Early-Cong-Pres
    imports Weak-Early-Cong Weak-Early-Step-Sim-Pres Weak-Early-Bisim-Pres
begin
lemma tauPres:
    fixes P :: pi
    and }Q::p
    assumes }P\simeq
    shows }\tau.(P)\simeq\tau.(Q
proof -
    from assms have P\approxQ by(rule congruenceWeakBisim)
    thus ?thesis by(force intro: Weak-Early-Step-Sim-Pres.tauPres simp add: weak-
Congruence-def dest: weakBisimE(2))
qed
lemma outputPres:
    fixes P :: pi
    and }Q:: p
```

```
    assumes \(P \simeq Q\)
    shows \(a\{b\} . P \simeq a\{b\} . Q\)
proof -
    from assms have \(P \approx Q \mathbf{b y}\) (rule congruenceWeakBisim)
    thus ?thesis by(force intro: Weak-Early-Step-Sim-Pres.outputPres simp add:
weakCongruence-def dest: weakBisimE(2))
qed
lemma matchPres:
    fixes \(P\) :: \(p i\)
    and \(\quad Q:: p i\)
    and \(a::\) name
    and \(b::\) name
    assumes \(P \simeq Q\)
    shows \([a \frown b] P \simeq[a \frown b] Q\)
using assms
by(auto simp add: weakCongruence-def intro: Weak-Early-Step-Sim-Pres.matchPres)
lemma mismatchPres:
    fixes \(P:: p i\)
    and \(\quad Q:: p i\)
    and \(a::\) name
    and \(b::\) name
    assumes \(P \simeq Q\)
    shows \([a \neq b] P \simeq[a \neq b] Q\)
using assms
by (auto simp add: weakCongruence-def intro: Weak-Early-Step-Sim-Pres.mismatchPres)
lemma sumPres:
    fixes \(P:: p i\)
    and \(\quad Q:: p i\)
    and \(\quad R:: p i\)
    assumes \(P \simeq Q\)
    shows \(P \oplus R \simeq Q \oplus R\)
using assms
by(auto simp add: weakCongruence-def intro: Weak-Early-Step-Sim-Pres.sumPres
Weak-Early-Bisim.reflexive)
lemma parPres:
    fixes \(P\) :: \(p i\)
    and \(\quad Q:: p i\)
```

```
    and }R::p
    assumes P\simeqQ
    shows P|R\simeqQ|R
proof -
    have }\PQR.\llbracketP\rightsquigarrow«weakBisim» Q;P\approxQ\rrbracket\LongrightarrowP|R\rightsquigarrow<weakBisim» Q| R
    proof -
    fix PQR
    assume P\rightsquigarrow«weakBisim» Q and P}
    thus P|R\rightsquigarrow«weakBisim» Q|R
    using Weak-Early-Bisim-Pres.parPres Weak-Early-Bisim-Pres.resPres Weak-Early-Bisim.reflexive
Weak-Early-Bisim.eqvt
            by(blast intro:Weak-Early-Step-Sim-Pres.parPres)
    qed
    moreover from assms have P}\approxQ\mathbf{by}(rule congruenceWeakBisim)
    ultimately show ?thesis using assms
    by(auto simp add: weakCongruence-def dest: weakBisimE)
qed
lemma resPres:
    fixes P :: pi
    and }Q::p
    and x :: name
    assumes PeqQ:P\simeqQ
    shows <\nux>P}\simeq<\nux>Q
proof -
    have }\bigwedgePQx.P\rightsquigarrow<weakBisim» Q\Longrightarrow<\nux>P\rightsquigarrow«weakBisim»<\nux>>
    proof -
        fix PQ 
        assume P\rightsquigarrow«weakBisim» Q
            with Weak-Early-Bisim.eqvt Weak-Early-Bisim-Pres.resPres show <\nux>P
\rightsquigarrow«weakBisim» <\nux>Q
            by(blast intro:Weak-Early-Step-Sim-Pres.resPres)
    qed
    with assms show ?thesis by(simp add: weakCongruence-def)
qed
lemma bangPres:
    fixes P :: pi
    and }Q::p
    assumes P\simeqQ
    shows !P\simeq!Q
using assms
proof(induct rule: weakCongISym2)
```

```
    case(cSim P Q)
    let ? }X={(P,Q)|PQ.P\simeqQ
    from }\langleP\simeqQ\rangle\mathrm{ have }(P,Q)\in\mathrm{ ?X by auto
    moreover have }\PQ.(P,Q)\in?X\LongrightarrowP\rightsquigarrow«weakBisim» Q by(auto sim
add: weakCongruence-def)
    moreover from congruenceWeakBisim have ?X \subseteq weakBisim by auto
    ultimately have !P \rightsquigarrow«bangRel weakBisim»!Q using Weak-Early-Bisim.eqvt
        by(rule Weak-Early-Step-Sim-Pres.bangPres)
    moreover have bangRel weakBisim \subseteq weakBisim by(rule bangRelSubWeak-
Bisim)
    ultimately show !P \rightsquigarrow«weakBisim»!Q
        by(rule Weak-Early-Step-Sim.monotonic)
qed
end
theory Weak-Early-Cong-Subst-Pres
    imports Weak-Early-Cong-Subst Weak-Early-Cong-Pres
begin
lemma weakCongStructCong:
    fixes P :: pi
    and }Q::p
    assumes P}\mp@subsup{\equiv}{s}{}
    shows P}\mp@subsup{\simeq}{}{s}
using assms
by(metis earlyCongStructCong strongEqWeakCong)
lemma tauPres:
    fixes P :: pi
    and }Q::p
    assumes P}\mp@subsup{\simeq}{}{s}
    shows }\tau.(P)\mp@subsup{\simeq}{}{s}\tau.(Q
using assms
by(auto simp add: weakCongruenceSubst-def intro: Weak-Early-Cong-Pres.tauPres)
lemma inputPres:
    fixes P :: pi
    and }Q:: p
    and a :: name
    and x :: name
    assumes PeqQ: P 工s }
```

```
    shows }a<x>.P\mp@subsup{\simeq}{}{s}a<x>.
proof(auto simp add: weakCongruenceSubst-def)
    fix s::(name }\times\mathrm{ name) list
    from congruenceWeakBisim have Input: \PQ a x s. \llbracketP[<s>] \simeqs}Q[<s>];x
s\rrbracket\Longrightarrow(a<x>.P)[<s>]\simeq(a<x>.Q)[<s>]
    apply(auto simp add: weakCongruenceSubst-def weakCongruence-def)
    apply(rule Weak-Early-Step-Sim-Pres.inputPres, auto)
    apply(erule-tac x=[(x,y)] in allE, auto)
    apply(rule Weak-Early-Step-Sim-Pres.inputPres,auto)
    by(erule-tac x=[(x,y)] in allE, auto)
    then obtain c::name where cFreshP:c\sharpP and cFreshQ:c\sharpQ and cFreshs:
c\sharps
    by(force intro: name-exists-fresh[of (P,Q,s)])
    from PeqQ have P[<([(x,c)] \cdots)>] 工s}Q[<([(x,c)] \cdots)>] by(rule partUnfold)
    hence }([(x,c)]\cdotP[<([(x,c)]\cdots)>])\mp@subsup{\simeq}{}{s}([(x,c)]\cdotQ[<([(x,c)]\cdots)>])\mathrm{ by (rule
Weak-Early-Cong-Subst.eqvtI)
    hence ([(x,c)] • P)[<s>] \simeqs}([(x,c)]\cdotQ)[<s>] by sim
    hence (a<c>. ([(x,c)] \cdot P))[<s>]\simeq (a<c>. ([(x,c)] \cdotQ))[<s>] using cFreshs
by(rule Input)
    moreover from cFreshP cFreshQ have a<x>.P =a<c>.([(x,c)].P) and
a<x>.Q =a<c>.([(x,c)] \cdot Q)
    by(simp add: Agent.alphaInput)+
    ultimately show (a<x>.P)[<s>]}\simeq(a<x>.Q)[<s>] by sim
qed
lemma outputPres:
    fixes P :: pi
    and }Q:: p
    assumes P}\mp@subsup{\simeq}{}{s}
    shows }a{b}.P\mp@subsup{\simeq}{}{s}a{b}.
using assms
by(auto simp add: weakCongruenceSubst-def intro: Weak-Early-Cong-Pres.outputPres)
lemma matchPres:
    fixes P :: pi
    and }Q:: p
    and a :: name
    and b :: name
    assumes P}\mp@subsup{\simeq}{}{s}
    shows }[a\frownb]P\mp@subsup{\simeq}{}{s}[a\frownb]
```

```
using assms
by (auto simp add: weakCongruenceSubst-def intro: Weak-Early-Cong-Pres.matchPres)
lemma mismatchPres:
    fixes \(P\) :: \(p i\)
    and \(\quad Q:: p i\)
    and \(a::\) name
    and \(b::\) name
    assumes \(P \simeq^{s} Q\)
    shows \([a \neq b] P \simeq^{s}[a \neq b] Q\)
using assms
by (auto simp add: weakCongruenceSubst-def intro: Weak-Early-Cong-Pres.mismatchPres)
lemma sumPres:
    fixes \(P\) :: \(p i\)
    and \(\quad Q:: p i\)
    and \(\quad R:: p i\)
    assumes \(P \simeq^{s} Q\)
    shows \(P \oplus R \simeq^{s} Q \oplus R\)
using assms
by (auto simp add: weakCongruenceSubst-def intro: Weak-Early-Cong-Pres.sumPres)
lemma parPres:
    fixes \(P\) :: \(p i\)
    and \(\quad Q:: p i\)
    and \(\quad R:: p i\)
    assumes \(P \simeq^{s} Q\)
    shows \(P\left\|R \simeq^{s} Q\right\| R\)
using assms
by (auto simp add: weakCongruenceSubst-def intro: Weak-Early-Cong-Pres.parPres)
lemma resPres:
    fixes \(P\) :: \(p i\)
    and \(\quad Q:: p i\)
    and \(x::\) name
    assumes \(P e q Q: P \simeq^{s} Q\)
    shows \(<\nu x>P \simeq^{s}<\nu x>Q\)
proof (auto simp add: weakCongruenceSubst-def)
    fix \(s::(\) name \(\times\) name) list
    have Goal: \(\bigwedge P Q x s . \llbracket P[\langle s\rangle] \rightsquigarrow «\) weakBisim» \(Q[\langle s\rangle] ; x \sharp s \rrbracket \Longrightarrow(\langle\nu x\rangle P)[\langle s\rangle]\)
```

$\rightsquigarrow «$ weakBisim» $(<\nu x>Q)[<s>]$
by (force intro: Weak-Early-Step-Sim-Pres.resPres Weak-Early-Bisim-Pres.resPres Weak-Early-Bisim.eqvt)
then obtain c::name where cFreshP:c甘P and cFresh $Q: c \sharp Q$ and cFreshs: $c \sharp s$
by (force intro: name-exists-fresh $[$ of $(P, Q, s)])$
from Peq $Q$ have $P[<([(x, c)] \cdot s)>] \rightsquigarrow «$ weakBisim» $Q[<([(x, c)] \cdot s)>]$ and $Q[<([(x, c)] \cdot s)>] \rightsquigarrow$ weakBisim» $P[<([(x, c)] \cdot s)>]$
by (force simp add: weakCongruenceSubst-def weakCongruence-def)+
hence $([(x, c)] \cdot(P[<([(x, c)] \cdot s)>])) \rightsquigarrow «$ weakBisim» $([(x, c)] \cdot(Q[<([(x, c)] \cdot$ $s)>])$ ) and

$$
([(x, c)] \cdot(Q[<([(x, c)] \cdot s)>])) \rightsquigarrow \text { weakBisim» }([(x, c)] \cdot(P[<([(x, c)] \cdot
$$

s) $>$ ]))
by(blast intro: Weak-Early-Step-Sim.eqvtI Weak-Early-Bisim.eqvt)+
hence $([(x, c)] \cdot P)[\langle s\rangle] \rightsquigarrow$ weakBisim» $([(x, c)] \cdot Q)[\langle s\rangle]$ and

$$
([(x, c)] \cdot Q)[<s>] \rightsquigarrow \text { weakBisim» }([(x, c)] \cdot P)[<s>] \text { by simp }+
$$

with cFreshs have $(<\nu c\rangle([(x, c)] \cdot P))[<s\rangle] \rightsquigarrow «$ weakBisim» $(<\nu c\rangle([(x, c)] \cdot$ $Q)$ ) $[\langle s\rangle]$ and
$(<\nu c\rangle([(x, c)] \cdot Q))[<s\rangle] \rightsquigarrow$ weakBisim» $(\langle\nu c\rangle([(x, c)] \cdot P))[<s\rangle]$
by (blast intro: Goal)+
moreover from cFreshP cFresh $Q$ have $\langle\nu x\rangle P=\langle\nu c\rangle([(x, c)] \cdot P)$ and $<\nu x>Q=<\nu c>([(x, c)] \cdot Q)$
by (simp add: alphaRes)+
ultimately show $(\langle\nu x\rangle P)[<s\rangle] \simeq(\langle\nu x\rangle Q)[\langle s\rangle]$
by (simp add: weakCongruence-def)
qed
lemma bangPres:
fixes $P$ :: $p i$
and $\quad Q:: p i$
assumes $P \simeq^{s} Q$
shows $!P \simeq^{s}!Q$
using assms
by (auto simp add: weakCongruenceSubst-def intro: Weak-Early-Cong-Pres.bangPres)
end
theory Strong-Late-Expansion-Law
imports Strong-Late-Bisim-SC
begin

```
nominal-primrec summands :: pi \(\Rightarrow\) pi set where
    summands \(\mathbf{0}=\{ \}\)
\(\mid\) summands \((\tau .(P))=\{\tau .(P)\}\)
\(\mid x \sharp a \Longrightarrow\) summands \((a<x>. P)=\{a<x>. P\}\)
\(\mid\) summands \((a\{b\} . P)=\{a\{b\} . P\}\)
| summands \(([a \frown b] P)=\{ \}\)
| summands \(([a \neq b] P)=\{ \}\)
| summands \((P \oplus Q)=(\) summands \(P) \cup(\) summands \(Q)\)
| summands \((P \| Q)=\{ \}\)
\(\mid\) summands \((<\nu x>P)=\left(\right.\) if \(\left(\exists a P^{\prime} . a \neq x \wedge P=a\{x\} . P^{\prime}\right)\) then \((\{<\nu x>P\})\) else
\{\})
\(\mid\) summands \((!P)=\{ \}\)
apply (auto simp add: fresh-singleton name-fresh-abs fresh-set-empty fresh-singleton
pi.fresh)
apply(finite-guess)+
by (fresh-guess)+
lemma summandsInput[simp]:
    fixes \(a\) :: name
    and \(x::\) name
    and \(\quad P:: p i\)
    shows summands \((a<x>. P)=\{a<x>. P\}\)
proof -
    obtain \(y\) where yineqa: \(y \neq a\) and \(y\) Fresh \(P: y \sharp P\)
        by(force intro: name-exists-fresh \([\) of \((a, P)]\) simp add: fresh-prod)
    from \(y\) Fresh \(P\) have \(a<x>. P=a<y>.([(x, y)] \cdot P)\) by (simp add: alphaInput)
    with yineqa show ?thesis by simp
qed
lemma finiteSummands:
    fixes \(P\) :: \(p i\)
    shows finite(summands \(P\) )
by (induct \(P\) rule: pi.induct) auto
lemma boundSummandDest[dest]:
    fixes \(x\) :: name
    and \(y\) :: name
    and \(P^{\prime}:: p i\)
    and \(P:: p i\)
    assumes \(<\nu x>x\{y\} . P^{\prime} \in\) summands \(P\)
    shows False
using assms
by (induct P rule: pi.induct, auto simp add: if-split pi.inject name-abs-eq name-calc)
```

```
lemma summandFresh:
    fixes \(P:: p i\)
    and \(\quad Q:: p i\)
    and \(x::\) name
    assumes \(P \in\) summands \(Q\)
    and \(\quad x \sharp Q\)
    shows \(x \sharp P\)
using assms
by(nominal-induct \(Q\) avoiding: \(P\) rule: pi.strong-induct, auto simp add: if-split)
nominal-primrec \(h n f:: p i \Rightarrow\) bool where
    hnf \(\mathbf{0}=\) True
\(\mid h n f(\tau .(P))=\) True
\(\mid x \sharp a \Longrightarrow h n f(a<x>. P)=\) True
| hnf \((a\{b\} . P)=\) True
| hnf \(([a \frown b] P)=\) False
|hnf \(([a \neq b] P)=\) False
\(\mid h n f(P \oplus Q)=((h n f P) \wedge(h n f Q) \wedge P \neq \mathbf{0} \wedge Q \neq \mathbf{0})\)
|hnf \((P \| Q)=\) False
\(\mid h n f(<\nu x>P)=\left(\exists a P^{\prime} . a \neq x \wedge P=a\{x\} . P^{\prime}\right)\)
| hnf \((!P)=\) False
apply (auto simp add: fresh-bool)
apply(finite-guess)+
by (fresh-guess)+
lemma hnfInput[simp]:
    fixes \(a\) :: name
    and \(x::\) name
    and \(\quad P:: p i\)
    shows hnf \((a<x>. P)\)
proof -
    obtain \(y\) where yineqa: \(y \neq a\) and \(y\) Fresh \(P: y \sharp P\)
        by (force intro: name-exists-fresh \([o f(a, P)]\) simp add: fresh-prod)
    from \(y\) Fresh \(P\) have \(a<x>. P=a<y>\). \(([(x, y)] \cdot P)\) by (simp add: alphaInput)
    with yineqa show? ?thesis by simp
qed
lemma summandTransition:
fixes \(P:: p i\)
and \(a\) :: name
and \(x\) :: name
and \(b\) :: name
and \(P^{\prime}:: p i\)
assumes hnf \(P\)
```

```
    shows P\longmapsto\tau\prec P'=(\tau.(P')\in summands P)
    and }P\longmapstoa<x>\prec\mp@subsup{P}{}{\prime}=(a<x>.\mp@subsup{P}{}{\prime}\in\mathrm{ summands }P
    and }P\longmapstoa[b]\prec\mp@subsup{P}{}{\prime}=(a{b}.\mp@subsup{P}{}{\prime}\in\mathrm{ summands }P
    and }a\not=x\LongrightarrowP\longmapstoa<\nux>\prec\mp@subsup{P}{}{\prime}=(<\nux>a{x}.\mp@subsup{P}{}{\prime}\in\mathrm{ summands }P
proof -
    from assms show }P\longmapsto\tau\prec\mp@subsup{P}{}{\prime}=(\tau.(\mp@subsup{P}{}{\prime})\in\mathrm{ summands P)
    proof(induct P rule: pi.induct)
        case PiNil
        show ?case by auto
    next
        case(Output a b P)
    show ?case by auto
    next
    case(Tau P)
    have }\tau.(P)\longmapsto\tau\prec\mp@subsup{P}{}{\prime}\Longrightarrow\tau.(\mp@subsup{P}{}{\prime})\in\mathrm{ summands ( }\tau.(P)
        by(auto elim: tauCases simp add: pi.inject residual.inject)
    moreover have \tau.( }\mp@subsup{P}{}{\prime})\in\mathrm{ summands }(\tau.(P))\Longrightarrow\tau.(P)\longmapsto\tau\prec\mp@subsup{P}{}{\prime
        by(auto simp add: pi.inject intro: transitions.Tau)
    ultimately show ?case by blast
    next
    case(Input a x P)
    show ?case by auto
    next
        case(Match a b P)
        have hnf ([a\frownb]P) by fact
    hence False by simp
    thus ?case by simp
    next
        case(Mismatch a b P)
        have hnf ([a\not=b]P) by fact
        hence False by simp
        thus ?case by simp
    next
    case(Sum P Q)
    have hnf (P\oplusQ) by fact
    hence Phnf:hnf P and Qhnf: hnf Q by simp+
    have IHP:P\longmapsto\tau\prec 榇=(\tau.(P')\in summands P)
    proof -
```



```
    with Phnf show ?thesis by simp
    qed
    have IHQ:Q\longmapsto\tau\prec 靘=(\tau.(P')\in summands Q)
    proof -
```



```
            with Qhnf show ?thesis by simp
    qed
```

```
    from IHP IHQ have P\oplusQ\longmapsto\tau\prec P'\Longrightarrow\tau.(P') summands (P\oplusQ)
    by(erule-tac sumCases, auto)
    moreover from IHP IHQ have \tau.( (P')\in summands }(P\oplusQ)\LongrightarrowP\oplusQ\longmapsto
\prec P'
    by(auto dest: Sum1 Sum2)
    ultimately show ?case by blast
    next
    case(Par P Q)
    have hnf (P|Q) by fact
    hence False by simp
    thus?case by simp
    next
    case(Res x P)
    thus ?case by(auto elim: resCasesF)
    next
    case(Bang P)
    have hnf (!P) by fact
    hence False by simp
    thus?case by simp
    qed
next
    from assms show }P\longmapstoa<x>\prec\mp@subsup{P}{}{\prime}=(a<x>.\mp@subsup{P}{}{\prime}\in\mathrm{ summands P)
    proof(induct P rule: pi.induct)
        case PiNil
        show ?case by auto
    next
        case(Output c b P)
        show ?case by auto
    next
        case(Tau P)
        show ?case by auto
    next
    case(Input b y P)
    have }b<y>.P\longmapstoa<x>\prec\mp@subsup{P}{}{\prime}\Longrightarrowa<x>.\mp@subsup{P}{}{\prime}\in\mathrm{ summands ( }b<y>.P
        by(auto elim: inputCases' simp add: pi.inject residual.inject)
        moreover have }a<x>.\mp@subsup{P}{}{\prime}\in\mathrm{ summands ( }b<y>.P)\Longrightarrowb<y>.P\longmapstoa<x>\prec
P'
        apply(auto simp add: pi.inject name-abs-eq intro: Late-Semantics.Input)
        apply(subgoal-tac b<x>}\prec[(x,y)]\cdotP=(b<y>\prec[(x,y)]\cdot[(x,y)]\cdotP)
        apply(auto intro: Late-Semantics.Input)
        by(simp add: alphaBoundResidual name-swap)
    ultimately show ?case by blast
    next
        case(Match a b P)
        have hnf ([a\frownb]P) by fact
        hence False by simp
        thus ?case by simp
    next
    case(Mismatch a b P)
```

```
    have hnf ([a\not=b]P) by fact
    hence False by simp
    thus ?case by simp
next
    case(Sum P Q)
    have hnf (P\oplusQ) by fact
    hence Phnf:hnf P and Qhnf: hnf Q by simp+
    have IHP:P\longmapstoa<x> \prec P'=( }a<x>.\mp@subsup{P}{}{\prime}\in\mathrm{ summands }P
    proof -
        have hnf P\LongrightarrowP\longmapstoa<x>\prec 质=(a<x>. P' \in summands P) by fact
        with Phnf show ?thesis by simp
    qed
    have IHQ:Q\longmapstoa<x> \prec P'=( }Q<x>.\mp@subsup{P}{}{\prime}\in\mathrm{ summands }Q
    proof -
        have hnf Q Q\longmapstoa<x>\prec P'=(a<x>.P' }\in\mathrm{ summands Q) by fact
        with Qhnf show ?thesis by simp
    qed
    from IHP IHQ have }P\oplusQ\longmapstoa<x>\prec P'\Longrightarrowa<x>.P'P'\in summands (P
Q)
        by(erule-tac sumCases, auto)
```



```
\longmapsto<x>\prec\prec P'
        by(auto dest: Sum1 Sum2)
    ultimately show ?case by blast
    next
    case(Par P Q)
    have hnf (P|Q) by fact
    hence False by simp
    thus ?case by simp
    next
    case(Res y P)
    have hnf(<\nuy>P) by fact
    thus ?case by(auto simp add: if-split)
    next
    case(Bang P)
    have hnf (!P) by fact
    hence False by simp
    thus?case by simp
    qed
next
from assms show }P\longmapstoa[b]\prec\mp@subsup{P}{}{\prime}=(a{b}.\mp@subsup{P}{}{\prime}\in\mathrm{ summands }P
proof(induct P rule: pi.induct)
    case PiNil
    show ?case by auto
next
    case(Output c d P)
```



```
        by(auto elim: outputCases simp add: residual.inject pi.inject)
    moreover have a{b}.\mp@subsup{P}{}{\prime}\in\mathrm{ summands (c{d}.P) ఋc{d}.P}\longmapstoa[b]\prec\mp@subsup{P}{}{\prime}
        by(auto simp add: pi.inject intro: transitions.Output)
    ultimately show ?case by blast
next
    case(Tau P)
    show ?case by auto
next
    case(Input c x P)
    show ?case by auto
next
    case(Match a b P)
    have hnf ([a\frownb]P) by fact
    hence False by simp
    thus ?case by simp
next
    case(Mismatch a b P)
    have hnf ([a\not=b]P) by fact
    hence False by simp
    thus ?case by simp
next
    case(Sum P Q)
    have hnf (P\oplusQ) by fact
    hence Phnf: hnf P and Qhnf: hnf Q by simp+
```



```
    proof -
    have hnf P\LongrightarrowP\longmapstoa[b]\prec ' ' = (a{b}. P' 
    with Phnf show ?thesis by simp
qed
    have IHQ:Q\longmapstoa[b]\prec P'=(a{b}.P'P
    proof -
    have hnf Q\LongrightarrowQ\longmapstoa[b]\prec P'=(a{b}.P'}\mp@subsup{P}{}{\prime}\in\mathrm{ summands }Q)\mathrm{ by fact
    with Qhnf show ?thesis by simp
    qed
    from IHP IHQ have }P\oplusQ\longmapstoa[b]\prec\mp@subsup{P}{}{\prime}\Longrightarrowa{b}.\mp@subsup{P}{}{\prime}\in\mathrm{ summands ( }P\oplusQ
        by(erule-tac sumCases, auto)
    moreover from IHP IHQ have a{b}.P' }\mp@subsup{P}{}{\prime}\in\mathrm{ summands }(P\oplusQ)\LongrightarrowP\oplus
\longmapsto[b]\prec 㐌
    by(auto dest: Sum1 Sum2)
    ultimately show ?case by blast
next
    case(Par P Q)
    have hnf (P|Q) by fact
    hence False by simp
    thus ?case by simp
```

```
    next
        case(Res x P)
    have hnf (<\nux>P) by fact
        thus ?case by(force elim: resCasesF outputCases simp add: if-split resid-
ual.inject)
    next
        case(Bang P)
        have hnf (!P) by fact
        hence False by simp
        thus ?case by simp
    qed
next
    assume }a\not=
    with assms show }P\longmapstoa<\nux>< < P'=(<\nux>a{{x}.\mp@subsup{P}{}{\prime}\in\mathrm{ summands P)
    proof(nominal-induct P avoiding: x P' rule: pi.strong-induct)
    case PiNil
        show ?case by auto
    next
        case(Output a b P)
        show ?case by auto
    next
        case(Tau P)
        show ?case by auto
    next
        case(Input a x P)
        show ?case by auto
    next
        case(Match a b P)
        have hnf ([a\frownb]P) by fact
        hence False by simp
        thus ?case by simp
    next
    case(Mismatch a b P)
    have hnf ([a\not=b]P) by fact
    hence False by simp
    thus ?case by simp
next
    case(Sum P Q)
    have hnf (P\oplusQ) by fact
    hence Phnf: hnf P and Qhnf: hnf Q by simp+
    have aineqx: a\not=x by fact
    have IHP: P\longmapstoa<\nux>}\prec\mp@subsup{P}{}{\prime}=(<\nux>a{x}.\mp@subsup{P}{}{\prime}\in\mathrm{ summands P)
    proof -
        have }\bigwedgex\mp@subsup{P}{}{\prime}.\llbrackethnfP;a\not=x\rrbracket\LongrightarrowP\longmapstoa<\nux>< \prec P'=(<\nux>a{x}.\mp@subsup{P}{}{\prime}
summands P) by fact
            with Phnf aineqx show ?thesis by simp
    qed
```

```
have \(I H Q: Q \longmapsto a<\nu x>\prec P^{\prime}=\left(<\nu x>a\{x\} . P^{\prime} \in\right.\) summands \(\left.Q\right)\)
proof -
    have \(\bigwedge x Q^{\prime} . \llbracket h n f Q ; a \neq x \rrbracket \Longrightarrow Q \longmapsto a<\nu x>\prec P^{\prime}=\left(<\nu x>a\{x\} \cdot P^{\prime} \in\right.\)
summands \(Q\) ) by fact
    with Qhnf aineqx show ?thesis by simp
    qed
```

    from \(I H P I H Q\) have \(P \oplus Q \longmapsto a<\nu x>\prec P^{\prime} \Longrightarrow<\nu x>a\{x\} \cdot P^{\prime} \in\) summands
    $(P \oplus Q)$
by(erule-tac sumCases, auto)
moreover from $I H P I H Q$ have $<\nu x>a\{x\} . P^{\prime} \in$ summands $(P \oplus Q) \Longrightarrow P$
$\oplus Q \longmapsto a<\nu x>\prec P^{\prime}$
by(auto dest: Sum1 Sum2)
ultimately show ?case by blast
next
case (Par P Q)
have $h n f(P \| Q)$ by fact
hence False by simp
thus? ?ase by simp
next
case (Res y P)
have Phnf: hnf $(<\nu y>P)$ by fact
then obtain $b P^{\prime \prime}$ where bineqy: $b \neq y$ and $P e q P^{\prime \prime}: P=b\{y\} . P^{\prime \prime}$
by auto
have $y \sharp x$ by fact hence xineqy: $x \neq y$ by simp
have $y$ Fresh $P^{\prime}: y \sharp P^{\prime}$ by fact
have aineqx: $a \neq x$ by fact
have $<\nu y>P \longmapsto a<\nu x>\prec P^{\prime} \Longrightarrow\left(<\nu x>a\{x\} . P^{\prime} \in\right.$ summands $\left.(<\nu y>P)\right)$
proof -
assume Trans: $<\nu y>P \longmapsto a<\nu x>\prec P^{\prime}$
hence aeqb: $a=b$ using xineqy bineqy PeqP ${ }^{\prime \prime}$
by (induct rule: resCasesB', auto elim: outputCases simp add: residual.inject
alpha' abs-fresh pi.inject)
have Goal: $\bigwedge x P^{\prime} . \llbracket<\nu y>b\{y\} . P^{\prime \prime} \longmapsto b<\nu x>\prec P^{\prime} ; x \neq y ; x \neq b ; x \sharp P^{\prime} \rrbracket$
$\Longrightarrow$
$<\nu x>b\{x\} . P^{\prime} \in \operatorname{summands}\left(<\nu y>b\{y\} \cdot P^{\prime \prime}\right)$
proof -
fix $x P^{\prime}$
assume $x$ Fresh $P^{\prime \prime}:(x:: n a m e) \sharp P^{\prime \prime}$ and xineq $b: x \neq b$
assume $<\nu y>b\{y\} . P^{\prime \prime} \longmapsto b<\nu x>\prec P^{\prime}$ and xineqy: $x \neq y$
moreover from $\langle x \neq b\rangle\left\langle x \sharp P^{\prime \prime}\right\rangle\langle x \neq y\rangle$ have $x \sharp b\{y\} . P^{\prime \prime}$ by simp
ultimately show $<\nu x>b\{x\} . P^{\prime} \in$ summands $\left(<\nu y>b\{y\} . P^{\prime \prime}\right)$
proof(induct rule: resCasesB)
case (cOpen a $P^{\prime \prime \prime}$ )
have BoundOutputS $b=$ BoundOutputS $a$ by fact hence beqa: $b=a$ by
simp
have Trans: $b\{y\} . P^{\prime \prime} \longmapsto a[y] \prec P^{\prime \prime \prime}$ by fact
with $P e q P^{\prime \prime}$ have $P^{\prime \prime} e q P^{\prime \prime \prime}: P^{\prime \prime}=P^{\prime \prime \prime}$

```
                by(force elim: outputCases simp add: residual.inject)
            with bineqy xineqy xFreshP'\prime}\mathrm{ have }y\sharpb{x}.([(x,y)]\cdot\mp@subsup{P}{}{\prime\prime\prime}
                by(simp add: name-fresh-abs name-calc name-fresh-left)
                with bineqy Phnf PeqP'\prime P'\primeeqP'\prime\prime xineqb show ?case
                    by(simp only: alphaRes, simp add: name-calc)
        next
            case(cRes }\mp@subsup{P}{}{\prime\prime\prime}
            have b{y}.\mp@subsup{P}{}{\prime\prime}\longmapstob<\nux>}\prec\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ by fact
            hence False by auto
            thus?case by simp
        qed
    qed
    obtain z where zineqx: z\not=x and zineqy: z\not=y and zFreshP'\prime:z\sharp 的"
                and zineqb:}z\not=b\mathrm{ and zFreshP': }z\sharp\mp@subsup{P}{}{\prime
    by(force intro: name-exists-fresh[of (x,y,b, P'\prime, P')] simp add: fresh-prod)
    from zFreshP' aeqb PeqP'\prime Trans have Trans': <\nuy>b{y}.P'\prime}\longmapstob<\nuz>\prec
[(z,x)]\cdot P
            by(simp add: alphaBoundResidual name-swap)
    hence <\nuz>b{z}.([(z,x)] \cdot P')\in summands (<\nuy>b{y}.\mp@subsup{P}{}{\prime\prime})\mathrm{ using zineqy}
zineqb zFreshP}\mp@subsup{P}{}{\prime\prime
            by(rule Goal)
    moreover from bineqy zineqx zFreshP' aineqx aeqb have x }\forallb{z}.([(z,x)] 
P')
            by(simp add: name-fresh-left name-calc)
            ultimately have <\nux>b{{x}.\mp@subsup{P}{}{\prime}\in\mathrm{ summands (< < y>b{y}.P'利) using zineqb}
            by(simp add: alphaRes name-calc)
            with aeqb PeqP"/ show ?thesis by blast
    qed
    moreover have }<\nux>a{x}.\mp@subsup{P}{}{\prime}\in\operatorname{summands}(<\nuy>P)\Longrightarrow<\nuy>P\longmapstoa<\nux
\prec P'
    proof -
    assume <\nux>a{x}.P'的粍mands(<\nuy>P)
            with PeqP'\prime have Summ: <\nux>>a{x}.\mp@subsup{P}{}{\prime}\in\operatorname{summands(<\nuy>b{y}.\mp@subsup{P}{}{\prime\prime})}\mathrm{ by}
simp
    moreover with bineqy xineqy have aeqb: a=b
            by(auto simp add: if-split pi.inject name-abs-eq name-fresh-fresh)
    from bineqy xineqy yFreshP' have }y\sharpb{x}.\mp@subsup{P}{}{\prime}\mathbf{by}(\mathrm{ simp add: name-calc)
            with Summ aeqb bineqy aineqx have <\nuy>b{y}.([(x,y)] \cdot P') \in sum-
mands(<\nuy>b{y}. P')
            by(simp only: alphaRes, simp add: name-calc)
            with aeqb PeqP'\prime}\mathrm{ have < < y>P ط}a<\nuy> \prec [(x,y)]\cdot P'
            by(auto intro: Open Output simp add: if-split pi.inject name-abs-eq)
    moreover from yFreshP' have }x\sharp[(x,y)]\cdot\mp@subsup{P}{}{\prime}\mathbf{by}(simp add: name-fresh-lef
name-calc)
            ultimately show ?thesis by(simp add: alphaBoundResidual name-swap)
    qed
    ultimately show ?case by blast
next
```

```
    case(Bang P)
    have hnf (!P) by fact
    hence False by simp
    thus?case by simp
    qed
qed
```

definition expandSet :: pi $\Rightarrow p i \Rightarrow$ pi set where
expandSet $P Q \equiv\left\{\tau .\left(P^{\prime} \| Q\right) \mid P^{\prime} . \tau \cdot\left(P^{\prime}\right) \in\right.$ summands $\left.P\right\} \cup$
$\left\{\tau .\left(P \| Q^{\prime}\right) \mid Q^{\prime} \cdot \tau \cdot\left(Q^{\prime}\right) \in\right.$ summands $\left.Q\right\} \cup$
$\left\{a\{b\} .\left(P^{\prime} \| Q\right) \mid a b P^{\prime} . a\{b\} . P^{\prime} \in\right.$ summands $\left.P\right\} \cup$
$\left\{a\{b\} .\left(P \| Q^{\prime}\right) \mid a b Q^{\prime} \cdot a\{b\} \cdot Q^{\prime} \in\right.$ summands $\left.Q\right\} \cup$
$\left\{a<x>.\left(P^{\prime} \| Q\right) \mid a x P^{\prime} . a<x>. P^{\prime} \in\right.$ summands $\left.P \wedge x \sharp Q\right\}$
$\cup$
$\left\{a<x>.\left(P \| Q^{\prime}\right) \mid a x Q^{\prime} . a<x>. Q^{\prime} \in\right.$ summands $Q \wedge x \sharp$
$P\} \cup$
$\left\{<\nu x>a\{x\} .\left(P^{\prime} \| Q\right) \mid a x P^{\prime} .<\nu x>a\{x\} . P^{\prime} \in\right.$ summands $P$
$\wedge x \sharp Q\} \cup$
$Q \wedge x \sharp P\} \cup$
$\left\{\tau .\left(P^{\prime}[x::=b] \| Q^{\prime}\right) \mid x P^{\prime} b Q^{\prime} . \exists a . a<x>. P^{\prime} \in\right.$ summands $P$
$\wedge a\{b\} . Q^{\prime} \in$ summands $\left.Q\right\} \cup$
$\left\{\tau .\left(P^{\prime} \|\left(Q^{\prime}[x::=b]\right)\right) \mid b P^{\prime} x Q^{\prime} . \exists a . a\{b\} . P^{\prime} \in\right.$ summands
$P \wedge a<x>. Q^{\prime} \in$ summands $\left.Q\right\} \cup$
$\left\{\tau .\left(<\nu y>\left(P^{\prime}[x::=y] \| Q^{\prime}\right)\right) \mid x P^{\prime} y Q^{\prime} . \exists a . a<x>. P^{\prime} \in\right.$
summands $\left.P \wedge<\nu y>a\{y\} \cdot Q^{\prime} \in \operatorname{summands} Q \wedge y \sharp P\right\} \cup$
$\left\{\tau .\left(<\nu y>\left(P^{\prime} \|\left(Q^{\prime}[x::=y]\right)\right)\right) \mid y P^{\prime} x Q^{\prime} \cdot \exists a .<\nu y>a\{y\} . P^{\prime}\right.$
$\in$ summands $P \wedge a<x>. Q^{\prime} \in$ summands $\left.Q \wedge y \sharp Q\right\}$
lemma finiteExpand:
fixes $P$ :: $p i$
and $\quad Q:: p i$
shows finite(expandSet $P Q$ )
proof -
have finite $\left\{\tau .\left(P^{\prime} \| Q\right) \mid P^{\prime} . \tau .\left(P^{\prime}\right) \in\right.$ summands $\left.P\right\}$
by (induct $P$ rule: pi.induct, auto simp add: pi.inject Collect-ex-eq conj-disj-distribL
Collect-disj-eq UN-Un-distrib)
moreover have finite $\left\{\tau .\left(P \| Q^{\prime}\right) \mid Q^{\prime} \cdot \tau \cdot\left(Q^{\prime}\right) \in\right.$ summands $\left.Q\right\}$
by (induct $Q$ rule: pi.induct, auto simp add: pi.inject Collect-ex-eq conj-disj-distribL
Collect-disj-eq UN-Un-distrib)
moreover have finite $\left\{a\{b\} .\left(P^{\prime} \| Q\right) \mid a b P^{\prime} . a\{b\} . P^{\prime} \in\right.$ summands $\left.P\right\}$
by (induct P rule: pi.induct, auto simp add: pi.inject Collect-ex-eq conj-disj-distribL
Collect-disj-eq UN-Un-distrib)
moreover have finite $\left\{a\{b\} .\left(P \| Q^{\prime}\right) \mid a b Q^{\prime} . a\{b\} . Q^{\prime} \in\right.$ summands $\left.Q\right\}$
by (induct $Q$ rule: pi.induct, auto simp add: pi.inject Collect-ex-eq conj-disj-distribL
Collect-disj-eq UN-Un-distrib)
moreover have finite $\left\{a<x>.\left(P^{\prime} \| Q\right) \mid a x P^{\prime} . a<x>. P^{\prime} \in\right.$ summands $P \wedge x \sharp$
Q\}

```
    proof -
```



```
=a<x>.P}\wedge \mp@subsup{x}{}{\prime}\sharpQ}={a<x>.(P|Q)
            by(auto simp add: pi.inject name-abs-eq name-fresh-fresh)
    thus ?thesis
            by(nominal-induct P avoiding:Q rule: pi.strong-induct,
                auto simp add: Collect-ex-eq conj-disj-distribL conj-disj-distribR
                    Collect-disj-eq UN-Un-distrib)
    qed
```



```
#P}
    proof -
    have Aux: \bigwedgeaxP Q.(x::name) \sharpP\Longrightarrow\Longrightarrow{\mp@subsup{a}{}{\prime}<x'>.(P| Q')|\mp@subsup{a}{}{\prime}\mp@subsup{x}{}{\prime}\mp@subsup{Q}{}{\prime}.\mp@subsup{a}{}{\prime}<\mp@subsup{x}{}{\prime}>.Q}\mp@subsup{Q}{}{\prime
=a<x>.Q\wedge x'\sharpP}={a<x>.(P|Q)}
by(auto simp add: pi.inject name-abs-eq name-fresh-fresh)
    thus ?thesis
            by(nominal-induct Q avoiding: P rule: pi.strong-induct,
                auto simp add: Collect-ex-eq conj-disj-distribL conj-disj-distribR
                    Collect-disj-eq UN-Un-distrib)
    qed
```



```
P\wedgex\sharpQ}
    proof -
        have Aux: \bigwedgeax P Q. \llbracketx\sharpQ;a\not=x\rrbracket\Longrightarrow \<\nux'>a'{x'}.(P'| |) |a' \mp@subsup{x}{}{\prime}\mp@subsup{P}{}{\prime}.
<\nu\mp@subsup{x}{}{\prime}>\mp@subsup{a}{}{\prime}{\mp@subsup{x}{}{\prime}}.\mp@subsup{P}{}{\prime}=<\nux>a{x}.P\wedge \mp@subsup{x}{}{\prime}\sharpQ}=
                                    {<\nux>a{x}.(P|Q)}
            by(auto simp add: pi.inject name-abs-eq name-fresh-fresh)
    thus ?thesis
            by(nominal-induct P avoiding: Q rule: pi.strong-induct,
                auto simp add: Collect-ex-eq conj-disj-distribL conj-disj-distribR
                        Collect-disj-eq UN-Un-distrib)
    qed
    moreover have finite {<\nux>a{x}.(P| | ')| a x Q '. <\nux>a{{x}.Q'的 summands
Q\wedgex\sharpP}
    proof -
    have Aux: \a x P Q. \llbracketx\sharpP;a\not=x\rrbracket\Longrightarrow \<\nux\mp@subsup{x}{}{\prime}>\mp@subsup{a}{}{\prime}{\mp@subsup{x}{}{\prime}}.(P| | Q ) | a' \mp@subsup{x}{}{\prime}\mp@subsup{Q}{}{\prime}.
<\nu\mp@subsup{x}{}{\prime}>\mp@subsup{a}{}{\prime}{x'}.\mp@subsup{Q}{}{\prime}=<\nux>a{x}.Q\wedge 和\sharpP}=
                                    {<\nux>>a{x}.(P|Q)}
        by(auto simp add: pi.inject name-abs-eq name-fresh-fresh)
    thus ?thesis
            by(nominal-induct Q avoiding: P rule: pi.strong-induct,
                auto simp add: Collect-ex-eq conj-disj-distribL conj-disj-distribR
                    Collect-disj-eq UN-Un-distrib)
    qed
    moreover have finite {\tau.( }\mp@subsup{P}{}{\prime}[x::=b]|\mp@subsup{Q}{}{\prime})|x\mp@subsup{P}{}{\prime}b\mp@subsup{Q}{}{\prime}.\existsa.a<x>.\mp@subsup{P}{}{\prime}\in\mathrm{ summands
P\wedgea{b}.\mp@subsup{Q}{}{\prime}\in\mathrm{ summands }Q}
    proof -
        have Aux: \bigwedgeax P b Q. {\tau.(P'[\mp@subsup{x}{}{\prime}::=b\rceil|| Q')| a' \mp@subsup{x}{}{\prime}\mp@subsup{P}{}{\prime}\mp@subsup{b}{}{\prime}\mp@subsup{Q}{}{\prime}.\mp@subsup{a}{}{\prime}<\mp@subsup{x}{}{\prime}>.\mp@subsup{P}{}{\prime}=
a<x>.P\wedge a'{b'}. Q' =a{b}.Q} ={\tau.(P[x::=b]|Q)}
```

by (auto simp add: name-abs-eq pi.inject renaming)
have $\bigwedge a x P Q b:::^{\prime} a::\{ \}$. finite $\left\{\tau .\left(P^{\prime}\left[x^{\prime}::=b\right] \| Q^{\prime}\right) \mid a^{\prime} x^{\prime} P^{\prime} b Q^{\prime} . a^{\prime}<x^{\prime}>. P^{\prime}\right.$ $=a<x>. P \wedge a^{\prime}\{b\} . Q^{\prime} \in$ summands $\left.Q\right\}$
apply (induct rule: pi.induct, simp-all)
apply (case-tac a=name1)
apply (simp add: Aux)
apply (simp add: pi.inject)
by (simp add: Collect-ex-eq conj-disj-distribL conj-disj-distribR Collect-disj-eq UN-Un-distrib)
hence finite $\left\{\tau .\left(P^{\prime}[x::=b] \| Q^{\prime}\right) \mid a x P^{\prime} b Q^{\prime} . a<x>. P^{\prime} \in\right.$ summands $P \wedge$ $a\{b\} . Q^{\prime} \in$ summands $\left.Q\right\}$
by (nominal-induct $P$ avoiding: $Q$ rule: pi.strong-induct,
auto simp add: Collect-ex-eq conj-disj-distribL conj-disj-distribR
Collect-disj-eq UN-Un-distrib name-abs-eq)
thus ?thesis
apply (rule-tac finite-subset)
defer
by blast+
qed
moreover have finite $\left\{\tau .\left(P^{\prime} \|\left(Q^{\prime}[x::=b]\right)\right) \mid b P^{\prime} x Q^{\prime} . \exists a . a\{b\} . P^{\prime} \in\right.$ summands $P \wedge a<x>. Q^{\prime} \in$ summands $\left.Q\right\}$
proof -
have Aux: $\bigwedge a x P b Q .\left\{\tau \cdot\left(P^{\prime} \|\left(Q^{\prime}\left[x^{\prime}::=b\right\rceil\right)\right) \mid a^{\prime} b^{\prime} P^{\prime} x^{\prime} Q^{\prime} \cdot a^{\prime}\left\{b^{\prime}\right\} . P^{\prime}=\right.$ $\left.a\{b\} . P \wedge a^{\prime}<x^{\prime}>\cdot Q^{\prime}=a<x>. Q\right\}=\{\tau .(P \|(Q[x::=b]))\}$
by (auto simp add: name-abs-eq pi.inject renaming)
have $\wedge a b P Q x::{ }^{\prime} a::\{ \}$. finite $\left\{\tau .\left(P^{\prime} \|\left(Q^{\prime}[x::=b\}\right)\right) \mid a^{\prime} b^{\prime} P^{\prime} x Q^{\prime} . a^{\prime}\left\{b^{\prime}\right\} . P^{\prime}\right.$ $=a\{b\} . P \wedge a^{\prime}<x>. Q^{\prime} \in$ summands $\left.Q\right\}$
apply(induct rule: pi.induct, simp-all)
apply (case-tac a=name1)
apply (simp add: Aux)
apply (simp add: pi.inject)
by (simp add: Collect-ex-eq conj-disj-distribL conj-disj-distribR
Collect-disj-eq UN-Un-distrib)
hence finite $\left\{\tau \cdot\left(P^{\prime} \|\left(Q^{\prime}[x::=b]\right)\right) \mid a b P^{\prime} x Q^{\prime} . a\{b\} . P^{\prime} \in\right.$ summands $P \wedge$ $a<x>. Q^{\prime} \in$ summands $\left.Q\right\}$
by (nominal-induct $P$ avoiding: $Q$ rule: pi.strong-induct, auto simp add: Collect-ex-eq conj-disj-distribL conj-disj-distribR

Collect-disj-eq UN-Un-distrib name-abs-eq)
thus ?thesis
apply (rule-tac finite-subset) defer by blast+
qed
moreover have finite $\left\{\tau .\left(<\nu y>\left(P^{\prime}[x::=y] \| Q^{\prime}\right)\right) \mid x P^{\prime} y Q^{\prime} . \exists a . a<x>. P^{\prime} \in\right.$ summands $P \wedge<\nu y>a\{y\} \cdot Q^{\prime} \in$ summands $\left.Q \wedge y \sharp P\right\}$

## proof -

have Aux: $\wedge a x P y$ Q. $y \sharp P \wedge y \neq a \Longrightarrow\left\{\tau .\left(<\nu y^{\prime}>\left(P^{\prime}\left[x^{\prime}::=y\right\} \| Q^{\prime}\right)\right) \mid\right.$ $a^{\prime} x^{\prime} P^{\prime} y^{\prime} Q^{\prime} . a^{\prime}<x^{\prime}>. P^{\prime}=a<x>. P \wedge<\nu y^{\prime}>a^{\prime}\left\{y^{\prime}\right\} . Q^{\prime}=<\nu y>a\{y\} . Q \wedge y^{\prime} \sharp$ $a<x>. P\}=\{\tau .(<\nu y>(P[x::=y] \| Q))\}$
apply (auto simp add: pi.inject name-abs-eq name-fresh-abs name-calc fresh-fact2 fresh-fact1 equts forget)
apply(subst name-swap, simp add: injPermSubst fresh-fact1 fresh-fact2)+ by (simp add: name-swap injPermSubst)+
have $B C: \bigwedge a x P Q$. finite $\left\{\tau .\left(<\nu y>\left(P^{\prime}\left[x^{\prime}::=y\right] \| Q^{\prime}\right)\right) \mid a^{\prime} x^{\prime} P^{\prime} y Q^{\prime} . a^{\prime}<x^{\prime}>. P^{\prime}\right.$ $=a<x>. P \wedge<\nu y>a^{\prime}\{y\} . Q^{\prime} \in$ summands $\left.Q \wedge y \sharp a<x>. P\right\}$
proof -
fix $a \times P Q$
show finite $\left\{\tau .\left(<\nu y>\left(P^{\prime}\left[x^{\prime}::=y\right] \| Q^{\prime}\right)\right) \mid a^{\prime} x^{\prime} P^{\prime} y Q^{\prime} . a^{\prime}<x^{\prime}>. P^{\prime}=a<x>. P\right.$ $\wedge<\nu y>a^{\prime}\{y\} . Q^{\prime} \in$ summands $\left.Q \wedge y \sharp a<x>. P\right\}$
apply(nominal-induct $Q$ avoiding: a $P$ rule: pi.strong-induct, simp-all)
apply (simp add: Collect-ex-eq conj-disj-distribL conj-disj-distribR Collect-disj-eq UN-Un-distrib)
apply (clarsimp)
apply (case-tac $a=a a$ )
apply (insert Aux, auto)
by (simp add: pi.inject name-abs-eq name-calc)
qed
have $I H: \wedge P P^{\prime} Q .\left\{\tau .\left(<\nu y>\left(P^{\prime \prime}[x::=y] \| Q^{\prime}\right)\right) \mid\right.$ ax $P^{\prime \prime} y Q^{\prime} .\left(a<x>. P^{\prime \prime} \in\right.$ summands $P \vee a<x>. P^{\prime \prime} \in$ summands $\left.P^{\prime}\right) \wedge<\nu y>a\{y\} . Q^{\prime} \in$ summands $Q \wedge y \sharp$ $\left.P \wedge y \sharp P^{\prime}\right\}=\left\{\tau .\left(<\nu y>\left(P^{\prime \prime}[x::=y] \| Q^{\prime}\right)\right) \mid a x P^{\prime \prime} y Q^{\prime} . a<x>. P^{\prime \prime} \in\right.$ summands $P \wedge<\nu y>a\{y\} . Q^{\prime} \in$ summands $\left.Q \wedge y \sharp P \wedge y \sharp P^{\prime}\right\} \cup\left\{\tau .\left(<\nu y>\left(P^{\prime \prime}[x::=y] \|\right.\right.\right.$ $\left.\left.Q^{\prime}\right)\right) \mid a x P^{\prime \prime} y Q^{\prime} . a<x>. P^{\prime \prime} \in$ summands $P^{\prime} \wedge<\nu y>a\{y\} . Q^{\prime} \in$ summands $Q \wedge$ $\left.y \sharp P \wedge y \sharp P^{\prime}\right\}$
by blast
have $I H^{\prime}: \wedge P Q P^{\prime} .\left\{\tau .\left(<\nu y>\left(P^{\prime \prime}[x::=y] \| Q^{\prime}\right)\right) \mid a x P^{\prime \prime} y Q^{\prime} . a<x>. P^{\prime \prime} \in\right.$ summands $P \wedge<\nu y>a\{y\} . Q^{\prime} \in$ summands $\left.Q \wedge y \sharp P \wedge y \sharp P^{\prime}\right\} \subseteq\left\{\tau .\left(<\nu y>\left(P^{\prime \prime}[x::=y]\right.\right.\right.$ $\left.\left.\| Q^{\prime}\right)\right) \mid a x P^{\prime \prime} y Q^{\prime} . a<x>. P^{\prime \prime} \in$ summands $P \wedge<\nu y>a\{y\} . Q^{\prime} \in$ summands $Q$ $\wedge y \sharp P\}$
by blast
have $I H^{\prime \prime}: \wedge P Q P^{\prime} .\left\{\tau .\left(<\nu y>\left(P^{\prime \prime}[x::=y] \| Q^{\prime}\right)\right) \mid\right.$ ax $P^{\prime \prime} y Q^{\prime} . a<x>. P^{\prime \prime} \in$ summands $P^{\prime} \wedge<\nu y>a\{y\} . Q^{\prime} \in$ summands $\left.Q \wedge y \sharp P \wedge y \sharp P^{\prime}\right\} \subseteq\left\{\tau .\left(<\nu y>\left(P^{\prime \prime}[x::=y]\right.\right.\right.$ $\left.\left.\| Q^{\prime}\right)\right) \mid a x P^{\prime \prime} y Q^{\prime} . a<x>. P^{\prime \prime} \in$ summands $P^{\prime} \wedge<\nu y>a\{y\} . Q^{\prime} \in$ summands $Q$ $\left.\wedge y \sharp P^{\prime}\right\}$
by blast
have finite $\left\{\tau .\left(<\nu y>\left(P^{\prime}[x::=y] \| Q^{\prime}\right)\right) \mid\right.$ a $x P^{\prime} y Q^{\prime} . a<x>. P^{\prime} \in$ summands $P$ $\wedge<\nu y>a\{y\} . Q^{\prime} \in$ summands $\left.Q \wedge y \sharp P\right\}$
apply (nominal-induct $P$ avoiding: $Q$ rule: pi.strong-induct, simp-all)
apply (insert BC, force)
apply (insert IH, auto)
apply (blast intro: finite-subset[OF IH $]$ )
by (blast intro: finite-subset $\left[O F I H^{\prime}\right\rceil$ )
thus ?thesis
apply(rule-tac finite-subset) defer by(blast)+
qed
moreover have finite $\left\{\tau .\left(<\nu y>\left(P^{\prime} \|\left(Q^{\prime}[x::=y]\right)\right)\right) \mid y P^{\prime} x Q^{\prime} . \exists a .<\nu y>a\{y\} . P^{\prime}\right.$ $\in$ summands $P \wedge a<x>. Q^{\prime} \in$ summands $\left.Q \wedge y \sharp Q\right\}$
proof -
have Aux: $\bigwedge a y P x Q . \llbracket y \sharp Q ; y \neq a \rrbracket \Longrightarrow\left\{\tau .\left(<\nu y^{\prime}>\left(P^{\prime} \|\left(Q^{\prime}\left[x^{\prime}::=y\right\}\right)\right)\right)\right.$


```
a<x>.Q} ={\tau.(<\nuy>(P|(Q[x::=y])))}
```

apply (auto simp add: pi.inject name-abs-eq name-fresh-abs name-calc fresh-fact2 fresh-fact1 forget eqvts fresh-left renaming[symmetric])
apply(subst name-swap, simp add: injPermSubst fresh-fact1 fresh-fact2)+ by (simp add: name-swap injPermSubst) +
have $I H: \wedge P$ y a $Q Q^{\prime}$. $\left\{\tau .\left(<\nu y^{\prime}>\left(P^{\prime} \|\left(Q^{\prime \prime}[x::=y]\right)\right)\right) \mid a^{\prime} y^{\prime} P^{\prime} x Q^{\prime \prime}\right.$. $<\nu y^{\prime}>a^{\prime}\left\{y^{\prime}\right\} . P^{\prime}=<\nu y>a\{y\} . P \wedge\left(a^{\prime}<x>. Q^{\prime \prime} \in\right.$ summands $Q \vee a^{\prime}<x>. Q^{\prime \prime} \in$ summands $\left.\left.Q^{\prime}\right) \wedge y^{\prime} \sharp Q \wedge y^{\prime} \sharp Q^{\prime}\right\}=\left\{\tau .\left(<\nu y^{\prime}>\left(P^{\prime} \|\left(Q^{\prime \prime}[x::=y]\right)\right)\right) \mid a^{\prime} y^{\prime} P^{\prime}\right.$ $x Q^{\prime \prime} .<\nu y^{\prime}>a^{\prime}\left\{y^{\prime}\right\} . P^{\prime}=<\nu y>a\{y\} . P \wedge a^{\prime}<x>\cdot Q^{\prime \prime} \in$ summands $Q \wedge y^{\prime} \sharp Q$ $\left.\wedge y^{\prime} \sharp Q^{\prime}\right\} \cup\left\{\tau .\left(<\nu y^{\prime}>\left(P^{\prime} \|\left(Q^{\prime \prime}[x::=y\rceil\right)\right)\right) \mid a^{\prime} y^{\prime} P^{\prime} x Q^{\prime \prime} .<\nu y^{\prime}>a^{\prime}\left\{y^{\prime}\right\} . P^{\prime}=\right.$ $<\nu y>a\{y\} . P \wedge a^{\prime}<x>. Q^{\prime \prime} \in$ summands $\left.Q^{\prime} \wedge y^{\prime} \sharp Q \wedge y^{\prime} \sharp Q^{\prime}\right\}$
by blast
have $I H^{\prime}: \bigwedge$ a y $P Q Q^{\prime} .\left\{\tau \cdot\left(<\nu y^{\prime}>\left(P^{\prime} \|\left(Q^{\prime \prime}[x::=y]\right)\right)\right) \mid a^{\prime} y^{\prime} P^{\prime} x Q^{\prime \prime}\right.$. $<\nu y^{\prime}>a^{\prime}\left\{y^{\prime}\right\} . P^{\prime}=<\nu y>a\{y\} . P \wedge a^{\prime}<x>. Q^{\prime \prime} \in$ summands $\left.Q \wedge y^{\prime} \sharp Q \wedge y^{\prime} \sharp Q^{\prime}\right\}$ $\subseteq\left\{\tau .\left(<\nu y^{\prime}>\left(P^{\prime} \|\left(Q^{\prime \prime}[x::=y\rceil\right)\right)\right) \mid a^{\prime} y^{\prime} P^{\prime} x Q^{\prime \prime} .<\nu y^{\prime}>a^{\prime}\left\{y^{\prime}\right\} . P^{\prime}=<\nu y>a\{y\} . P\right.$ $\wedge a^{\prime}<x>. Q^{\prime \prime} \in$ summands $\left.Q \wedge y^{\prime} \sharp Q\right\}$
by blast
have $I H^{\prime \prime}: \bigwedge a$ y $P Q Q^{\prime} .\left\{\tau .\left(<\nu y^{\prime}>\left(P^{\prime} \|\left(Q^{\prime \prime}[x::=y]\right)\right)\right) \mid a^{\prime} y^{\prime} P^{\prime} x Q^{\prime \prime}\right.$. $<\nu y^{\prime}>a^{\prime}\left\{y^{\prime}\right\} . P^{\prime}=<\nu y>a\{y\} . P \wedge a^{\prime}<x>. Q^{\prime \prime} \in$ summands $\left.Q^{\prime} \wedge y^{\prime} \sharp Q \wedge y^{\prime} \sharp Q^{\prime}\right\}$ $\subseteq\left\{\tau .\left(<\nu y^{\prime}>\left(P^{\prime} \|\left(Q^{\prime \prime}[x::=y]\right)\right)\right) \mid a^{\prime} y^{\prime} P^{\prime} x Q^{\prime \prime} .<\nu y^{\prime}>a^{\prime}\left\{y^{\prime}\right\} . P^{\prime}=<\nu y>a\{y\} . P\right.$ $\wedge a^{\prime}<x>. Q^{\prime \prime} \in$ summands $\left.Q^{\prime} \wedge y^{\prime} \sharp Q^{\prime}\right\}$
by blast
have $B C: \bigwedge a y P Q . \llbracket y \sharp Q ; y \neq a \rrbracket \Longrightarrow$ finite $\left\{\tau \cdot\left(<\nu y^{\prime}>\left(P^{\prime} \|\left(Q^{\prime}[x::=y\rceil\right)\right)\right)\right.$ $\mid a^{\prime} y^{\prime} P^{\prime} x Q^{\prime} .<\nu y^{\prime}>a^{\prime}\left\{y^{\prime}\right\} . P^{\prime}=<\nu y>a\{y\} . P \wedge a^{\prime}<x>. Q^{\prime} \in$ summands $Q \wedge y^{\prime}$ $\sharp Q\}$
proof -
fix $a$ y $P Q$
assume ( $y::$ name) $\sharp(Q:: p i)$ and $y \neq a$
thus finite $\left\{\tau .\left(<\nu y^{\prime}>\left(P^{\prime} \|\left(Q^{\prime}[x::=y\}\right)\right)\right) \mid a^{\prime} y^{\prime} P^{\prime} x Q^{\prime} .<\nu y^{\prime}>a^{\prime}\left\{y^{\prime}\right\} . P^{\prime}=\right.$ $<\nu y>a\{y\} . P \wedge a^{\prime}<x>. Q^{\prime} \in$ summands $\left.Q \wedge y^{\prime} \sharp Q\right\}$
apply (nominal-induct $Q$ avoiding: y rule: pi.strong-induct, simp-all)
apply (case-tac a=name1)
apply auto
apply(subgoal-tac ya $\#(p i:: p i))$
apply (insert Aux)
apply auto
apply (simp add: name-fresh-abs)
apply (simp add: pi.inject name-abs-eq name-calc)
apply (insert IH)
apply auto
apply(blast intro: finite-subset[OF IH $]$ )
by (blast intro: finite-subset $\left.\left[O F I H^{\prime \prime}\right]\right)$
qed
have finite $\left\{\tau .\left(<\nu y>\left(P^{\prime} \|\left(Q^{\prime}[x::=y]\right)\right)\right) \mid a y P^{\prime} x Q^{\prime} .<\nu y>a\{y\} . P^{\prime} \in\right.$ summands $P \wedge a<x>. Q^{\prime} \in$ summands $\left.Q \wedge y \sharp Q\right\}$

```
            apply(nominal-induct P avoiding:Q rule: pi.strong-induct, simp-all)
            apply(simp add: Collect-ex-eq conj-disj-distribL conj-disj-distribR name-fresh-abs
                                    Collect-disj-eq UN-Un-distrib)
            by(auto intro: BC)
    thus ?thesis
            apply(rule-tac finite-subset) defer by blast+
    qed
    ultimately show ?thesis
    by(simp add: expandSet-def)
qed
lemma expandHnf:
    fixes P :: pi
    and }Q:: p
    shows }\forallR\in(\mathrm{ expandSet P Q). hnf R
by(force simp add: expandSet-def)
inductive-set sumComposeSet :: (pi\times pi set) set
where
    empty:}(\mathbf{0},{})\in\mathrm{ sumComposeSet
| insert: }\llbracketQ\inS;(P,S-{Q})\in sumComposeSet\rrbracket\Longrightarrow(P\oplusQ,S)\in sumCom
poseSet
lemma expandAction:
    fixes }P:: p
    and }Q::p
    and}S:: pi se
    assumes }(P,S)\in\mathrm{ sumComposeSet
    and }\quadQ\in
    and }\quadQ\longmapstoR
    shows }P\longmapstoR
using assms
proof(induct arbitrary:Q rule: sumComposeSet.induct)
    case empty
    have Q\in{} by fact
    hence False by simp
    thus?case by simp
next
    case(insert Q'S PQ)
    have QTrans: Q\longmapstoRs by fact
    show ?case
    proof(case-tac Q = Q')
    assume Q = Q'
    with QTrans show P}\oplus\mp@subsup{Q}{}{\prime}\longmapstoRs\mathrm{ by(blast intro: Sum2)
    next
```

```
    assume QineqQ':Q}\not=\mp@subsup{Q}{}{\prime
    have IH:\Q.\llbracketQ\inS-{Q'};Q\longmapstoRs\rrbracket\LongrightarrowP\longmapsto \
    have QinS: Q \inS by fact
    with QineqQ' have Q S S-{Q'} by simp
    hence }P\longmapstoRs\mathrm{ using QTrans by(rule IH)
    thus ?case by(rule Sum1)
    qed
qed
lemma expandAction':
    fixes P :: pi
    and }Q::p
    and }R::p
    assumes (R,S)\in sumComposeSet
    and }R\longmapstoR
    shows }\existsP\inS.P\longmapstoR
using assms
proof(induct rule: sumComposeSet.induct)
    case empty
    have 0}\longmapstoRs\mathrm{ by fact
    hence False by blast
    thus?case by simp
next
    case(insert Q S P)
    have QinS:Q\inS by fact
    have }P\oplusQ\longmapstoRs\mathrm{ by fact
    thus ?case
    proof(induct rule: sumCases)
    case cSum1
    have P\longmapstoRs by fact
    moreover have }P\longmapstoRs\Longrightarrow\existsP\in(S-{Q}).P\longmapstoRs by fac
    ultimately obtain P where PinS: P}\in(S-{Q})\mathrm{ and PTrans: P}\longmapsto>R
by blast
    show ?case
    proof(case-tac P = Q)
        assume P=Q
        with PTrans QinS show ?case by blast
    next
        assume PineqQ: P}\not=
        from PinS have P}\inS\mathrm{ by simp
        with PTrans show ?thesis by blast
    qed
next
    case cSum2
    have Q}\longmapstoRs\mathrm{ by fact
    with QinS show ?case by blast
qed
```


## qed

lemma expandTrans:

$$
\text { fixes } P:: p i
$$

and $\quad Q:: p i$
and $\quad R:: p i$
and $a::$ name
and $b::$ name
and $x::$ name
assumes Exp: $(R$, expandSet $P Q) \in$ sumComposeSet
and Phnf:hnf P
and Qhnf: hnf $Q$
shows $\left(P \| Q \longmapsto \tau \prec P^{\prime}\right)=\left(R \longmapsto \tau \prec P^{\prime}\right)$
and $\quad\left(P \| Q \longmapsto a[b] \prec P^{\prime}\right)=\left(R \longmapsto a[b] \prec P^{\prime}\right)$
and $\quad\left(P \| Q \longmapsto a<x>\prec P^{\prime}\right)=\left(R \longmapsto a<x>\prec P^{\prime}\right)$
and $\quad\left(P \| Q \longmapsto a<\nu x>\prec P^{\prime}\right)=\left(R \longmapsto a<\nu x>\prec P^{\prime}\right)$
proof -
show $P \| Q \longmapsto \tau \prec P^{\prime}=R \longmapsto \tau \prec P^{\prime}$
proof(rule iffI)
assume $P \| Q \longmapsto \tau \prec P^{\prime}$
thus $R \longmapsto \tau \prec P^{\prime}$
proof (induct rule: parCasesF[of $-\cdots(P, Q)])$
case( $c$ Par1 $P^{\prime}$ )
have $P \longmapsto \tau \prec P^{\prime}$ by fact
with Phnf have $\tau .\left(P^{\prime}\right) \in$ summands $P \mathbf{b y}($ simp add: summandTransition)
hence $\tau .\left(P^{\prime} \| Q\right) \in$ expandSet $P Q$ by (auto simp add: expandSet-def)
moreover have $\tau .\left(P^{\prime} \| Q\right) \longmapsto \tau \prec\left(P^{\prime} \| Q\right)$ by (rule Tau)
ultimately show ?case using Exp by (blast intro: expandAction)
next
case(cPar2 $Q^{\prime}$ )
have $Q \longmapsto \tau \prec Q^{\prime}$ by fact
with $Q h n f$ have $\tau .\left(Q^{\prime}\right) \in$ summands $Q$ by (simp add: summandTransition)
hence $\tau .\left(P \| Q^{\prime}\right) \in$ expandSet $P Q$ by(auto simp add: expandSet-def)
moreover have $\tau .\left(P \| Q^{\prime}\right) \longmapsto \tau \prec\left(P \| Q^{\prime}\right)$ by (rule Tau)
ultimately show ?case using Exp by (blast intro: expandAction)
next
case $\left(c C o m m 1 P^{\prime} Q^{\prime}\right.$ abl)
have $P \longmapsto a<x>\prec P^{\prime}$ and $Q \longmapsto a[b] \prec Q^{\prime}$ by fact +
with Phnf Qhnf have $a<x>. P^{\prime} \in$ summands $P$ and $a\{b\} . Q^{\prime} \in$ summands
$Q \mathbf{b y}($ simp add: summandTransition $)+$
hence $\tau .\left(P^{\prime}[x::=b] \| Q^{\prime}\right) \in$ expandSet $P Q$ by (simp add: expandSet-def, blast)
moreover have $\tau .\left(P^{\prime}[x::=b] \| Q^{\prime}\right) \longmapsto \tau \prec\left(P^{\prime}[x::=b] \| Q^{\prime}\right)$ by (rule Tau)
ultimately show ?case using Exp by (blast intro: expandAction)
next
case (cComm2 $P^{\prime} Q^{\prime}$ abl)
have $P \longmapsto a[b] \prec P^{\prime}$ and $Q \longmapsto a<x>\prec Q^{\prime}$ by fact +
with Phnf Qhnf have $a\{b\} . P^{\prime} \in$ summands $P$ and $a<x>. Q^{\prime} \in$ summands
$Q \mathbf{b y}($ simp add: summandTransition $)+$
hence $\tau \cdot\left(P^{\prime} \|\left(Q^{\prime}[x::=b]\right)\right) \in$ expandSet $P Q$ by $($ simp add: expandSet-def, blast)
moreover have $\tau .\left(P^{\prime} \|\left(Q^{\prime}[x::=b]\right)\right) \longmapsto \tau \prec\left(P^{\prime} \|\left(Q^{\prime}[x::=b]\right)\right)$ by (rule Tau)
ultimately show ?case using Exp by (blast intro: expandAction)
next
case $\left(\right.$ cClose1 $P^{\prime} Q^{\prime}$ a $x y$ )
have $y \sharp(P, Q)$ by fact
hence $y$ FreshP: $y \sharp P$ by (simp add: fresh-prod)
have PTrans: $P \longmapsto a<x>\prec P^{\prime}$ by fact
with Phnf have PSumm: $a<x>. P^{\prime} \in$ summands $P$ by(simp add: summandTransition)
have $Q \longmapsto a<\nu y>\prec Q^{\prime}$ by fact
moreover from PTrans $y$ Fresh $P$ have $y \neq a$ by (force dest: freshBoundDerivative)
ultimately have $<\nu y>a\{y\} . Q^{\prime} \in$ summands $Q$ using $Q h n f$ by (simp add: summandTransition)
with PSumm yFreshP have $\tau .\left(<\nu y>\left(P^{\prime}[x::=y] \| Q^{\prime}\right)\right) \in$ expandSet $P Q$
by (auto simp add: expandSet-def)
moreover have $\tau .\left(<\nu y>\left(P^{\prime}[x::=y] \| Q^{\prime}\right)\right) \longmapsto \tau \prec<\nu y>\left(P^{\prime}[x::=y] \| Q^{\prime}\right)$ by(rule Tau)
ultimately show ?case using Exp by(blast intro: expandAction)
next
case $\left(c\right.$ Close2 $P^{\prime} Q^{\prime}$ a $x y$ )
have $y \sharp(P, Q)$ by fact
hence $y$ Fresh $Q: y \sharp Q$ by (simp add: fresh-prod)
have $Q$ Trans: $Q \longmapsto a<x>\prec Q^{\prime}$ by fact
with $Q h n f$ have $Q S u m m$ : $a<x>. Q^{\prime} \in$ summands $Q$ by (simp add: summandTransition)
have $P \longmapsto a<\nu y>\prec P^{\prime}$ by fact
moreover from $Q$ Trans yFresh $Q$ have $y \neq a$ by (force dest: freshBoundDerivative)
ultimately have $<\nu y>a\{y\} . P^{\prime} \in$ summands $P$ using Phnf by (simp add: summandTransition)
with $Q$ Summ $y$ Fresh $Q$ have $\tau .\left(<\nu y>\left(P^{\prime} \|\left(Q^{\prime}[x::=y]\right)\right)\right) \in \operatorname{expandSet} P Q$
by (simp add: expandSet-def, blast)
moreover have $\tau .\left(<\nu y>\left(P^{\prime} \|\left(Q^{\prime}[x::=y]\right)\right)\right) \longmapsto \tau \prec<\nu y>\left(P^{\prime} \|\left(Q^{\prime}[x::=y]\right)\right)$
by(rule Tau)
ultimately show ?case using Exp by(blast intro: expandAction)
qed
next
assume $R \longmapsto \tau \prec P^{\prime}$
with Exp obtain $R$ where $R \in \operatorname{expandSet} P Q$ and $R \longmapsto \tau \prec P^{\prime}$ by (blast dest: expandAction')
thus $P \| Q \longmapsto \tau \prec P^{\prime}$
proof (auto simp add: expandSet-def)
fix $P^{\prime \prime}$
assume $\tau .\left(P^{\prime \prime}\right) \in$ summands $P$
with Phnf have $P \longmapsto \tau \prec P^{\prime \prime}$ by (simp add: summandTransition)
hence $P Q$ Trans: $P\left\|Q \longmapsto \tau \prec P^{\prime \prime}\right\| Q$ by (rule Par1F)
assume $\tau .\left(P^{\prime \prime} \| Q\right) \longmapsto \tau \prec P^{\prime}$
hence $P^{\prime}=P^{\prime \prime} \| Q$ by (erule-tac tauCases, auto simp add: pi.inject residual.inject)
with PQTrans show ?thesis by simp
next
fix $Q^{\prime}$
assume $\tau .\left(Q^{\prime}\right) \in$ summands $Q$
with Qhnf have $Q \longmapsto \tau \prec Q^{\prime} \mathbf{b y}$ (simp add: summandTransition)
hence PQTrans: $P\|Q \longmapsto \tau \prec P\| Q^{\prime}$ by (rule Par2F)
assume $\tau .\left(P \| Q^{\prime}\right) \longmapsto \tau \prec P^{\prime}$
hence $P^{\prime}=P \| Q^{\prime}$ by (erule-tac tauCases, auto simp add: pi.inject residual.inject)
with PQTrans show ?thesis by simp
next
fix $a x P^{\prime \prime} b Q^{\prime}$
assume $a<x>. P^{\prime \prime} \in$ summands $P$ and $a\{b\} . Q^{\prime} \in$ summands $Q$
with Phnf Qhnf have $P \longmapsto a<x>\prec P^{\prime \prime}$ and $Q \longmapsto a[b] \prec Q^{\prime}$ by (simp add: summandTransition) +
hence PQTrans: $P\left\|Q \longmapsto \tau \prec P^{\prime \prime}[x::=b]\right\| Q^{\prime} \mathbf{b y}($ rule Comm1)
assume $\tau$. $\left(P^{\prime \prime}[x::=b] \| Q^{\prime}\right) \longmapsto \tau \prec P^{\prime}$
hence $P^{\prime}=P^{\prime \prime}[x::=b] \| Q^{\prime}$ by (erule-tac tauCases, auto simp add: pi.inject residual.inject)
with $P Q$ Trans show ?thesis by simp
next
fix $a b P^{\prime \prime} x Q^{\prime}$
assume $a\{b\} . P^{\prime \prime} \in$ summands $P$ and $a<x>. Q^{\prime} \in$ summands $Q$
with Phnf Qhnf have $P \longmapsto a[b] \prec P^{\prime \prime}$ and $Q \longmapsto a<x>\prec Q^{\prime}$ by (simp add: summandTransition)+
hence $P Q$ Trans: $P\left\|Q \longmapsto \tau \prec P^{\prime \prime}\right\|\left(Q^{\prime}[x::=b]\right) \mathbf{b y}($ rule Comm2)
assume $\tau .\left(P^{\prime \prime} \|\left(Q^{\prime}[x::=b]\right)\right) \longmapsto \tau \prec P^{\prime}$
hence $P^{\prime}=P^{\prime \prime} \|\left(Q^{\prime}[x::=b]\right)$ by (erule-tac tauCases, auto simp add: pi.inject residual.inject)
with PQTrans show ?thesis by simp
next
fix $a x P^{\prime \prime} y Q^{\prime}$
assume $y$ FreshP: ( $y$ ::name) $\sharp P$
assume $a<x>. P^{\prime \prime} \in$ summands $P$
with Phnf have PTrans: $P \longmapsto a<x>\prec P^{\prime \prime}$ by (simp add: summandTransition)
assume $<\nu y>a\{y\} . Q^{\prime} \in$ summands $Q$
moreover from $y$ FreshP PTrans have $y \neq a$ by (force dest: freshBoundDerivative)
ultimately have $Q \longmapsto a<\nu y>\prec Q^{\prime}$ using Qhnf by(simp add: summandTransition)
with PTrans have PQTrans: $P \| Q \longmapsto \tau \prec<\nu y>\left(P^{\prime \prime}[x::=y] \| Q^{\prime}\right)$ using $y$ FreshP by(rule Close1)
assume $\tau .\left(<\nu y>\left(P^{\prime \prime}[x::=y] \| Q^{\prime}\right)\right) \longmapsto \tau \prec P^{\prime}$
hence $P^{\prime}=<\nu y>\left(P^{\prime \prime}[x::=y] \| Q^{\prime}\right)$ by(erule-tac tauCases, auto simp add:

```
pi.inject residual.inject)
            with PQTrans show ?thesis by simp
    next
            fix \(a y P^{\prime \prime} x Q^{\prime}\)
            assume \(y\) Fresh \(Q:(y::\) name \() \sharp Q\)
            assume \(a<x>. Q^{\prime} \in\) summands \(Q\)
            with \(Q h n f\) have \(Q\) Trans: \(Q \longmapsto a<x>\prec Q^{\prime}\) by (simp add: summandTransi-
tion)
    assume \(<\nu y>a\{y\} . P^{\prime \prime} \in\) summands \(P\)
            moreover from \(y\) Fresh \(Q\) QTrans have \(y \neq a\) by (force dest: freshBound-
Derivative)
                            ultimately have \(P \longmapsto a<\nu y>\prec P^{\prime \prime}\) using Phnf by (simp add: summand-
Transition)
    hence \(P Q\) Trans: \(P \| Q \longmapsto \tau \prec<\nu y>\left(P^{\prime \prime} \| Q^{\prime}[x::=y]\right)\) using \(Q\) Trans \(y F r e s h ~ Q\)
by(rule Close2)
    assume \(\tau .\left(<\nu y>\left(P^{\prime \prime} \| Q^{\prime}[x::=y]\right)\right) \longmapsto \tau \prec P^{\prime}\)
    hence \(P^{\prime}=<\nu y>\left(P^{\prime \prime} \| Q^{\prime}[x::=y]\right)\) by (erule-tac tauCases, auto simp add:
pi.inject residual.inject)
            with PQTrans show ?thesis by simp
        qed
    qed
next
    show \(P \| Q \longmapsto a[b] \prec P^{\prime}=R \longmapsto a[b] \prec P^{\prime}\)
    proof (rule iffI)
    assume \(P \| Q \longmapsto a[b] \prec P^{\prime}\)
    thus \(R \longmapsto a[b] \prec P^{\prime}\)
    proof \((\) induct rule: parCasesF[where \(C=()])\)
        case(cPar1 \(P^{\prime}\) )
        have \(P \longmapsto a[b] \prec P^{\prime}\) by fact
        with Phnf have \(a\{b\} . P^{\prime} \in\) summands \(P\) by (simp add: summandTransition)
        hence \(a\{b\} .\left(P^{\prime} \| Q\right) \in\) expandSet \(P Q\) by (auto simp add: expandSet-def)
        moreover have \(a\{b\} .\left(P^{\prime} \| Q\right) \longmapsto a[b] \prec\left(P^{\prime} \| Q\right)\) by (rule Output)
        ultimately show ?case using Exp by(blast intro: expandAction)
    next
        case (cPar2 \(Q^{\prime}\) )
        have \(Q \longmapsto a[b] \prec Q^{\prime}\) by fact
        with Qhnf have \(a\{b\} . Q^{\prime} \in\) summands \(Q\) by (simp add: summandTransition)
        hence \(a\{b\} .\left(P \| Q^{\prime}\right) \in\) expandSet \(P Q\) by (simp add: expandSet-def, blast)
        moreover have \(a\{b\} .\left(P \| Q^{\prime}\right) \longmapsto a[b] \prec\left(P \| Q^{\prime}\right)\) by (rule Output)
        ultimately show ?case using Exp by(blast intro: expandAction)
    next
        case \(c\) Comm1
        thus ?case by auto
    next
        case cComm2
        thus ?case by auto
    next
        case \(c\) Close 1
        thus ?case by auto
```

```
    next
            case cClose2
            thus?case by auto
    qed
    next
    assume R\longmapstoa[b]\prec\mp@subsup{P}{}{\prime}
    with Exp obtain R where R expandSet P Q and R\longmapstoa[b]\prec 盾 by(blast
dest: expandAction')
    thus }P|Q\longmapstoa[b]\prec\mp@subsup{P}{}{\prime
    proof(auto simp add: expandSet-def)
        fix }\mp@subsup{a}{}{\prime}\mp@subsup{b}{}{\prime}\mp@subsup{P}{}{\prime\prime
        assume a'{b}}.\mp@subsup{P}{}{\prime\prime}\in\mathrm{ summands }
        with Phnf have }P\longmapsto\mp@subsup{a}{}{\prime}[b]\prec\mp@subsup{P}{}{\prime\prime}\mathrm{ by(simp add: summandTransition)
        hence PQTrans: P|Q\longmapstoa\longmapsto |b]\prec P'\prime|}Q\mathbf{by}(\mathrm{ rule Par1F)
        assume }\mp@subsup{a}{}{\prime}{\mp@subsup{b}{}{\prime}}.(\mp@subsup{P}{}{\prime\prime}|Q)\longmapstoa[b]\prec\mp@subsup{P}{}{\prime
        hence P' = P'\prime|}|\mathrm{ and }a=\mp@subsup{a}{}{\prime}\mathrm{ and b= b
            by(erule-tac outputCases, auto simp add: pi.inject residual.inject)+
        with PQTrans show ?thesis by simp
    next
        fix }\mp@subsup{a}{}{\prime}\mp@subsup{b}{}{\prime}\mp@subsup{Q}{}{\prime
        assume }\mp@subsup{a}{}{\prime}{\mp@subsup{b}{}{\prime}}.\mp@subsup{Q}{}{\prime}\in\mathrm{ summands }
        with Qhnf have Q\longmapstoa'[b]\prec Q' by(simp add: summandTransition)
        hence PQTrans: P|Q\longmapstoa}|\mp@code{|}[b]\precP|\mp@subsup{Q}{}{\prime}\mathbf{by}(\mathrm{ rule Par2F)
        assume a}\mp@subsup{a}{}{\prime}{b}.(P|\mp@subsup{Q}{}{\prime})\longmapstoa[b]\prec\mp@subsup{P}{}{\prime
        hence }\mp@subsup{P}{}{\prime}=P|\mp@subsup{Q}{}{\prime}\mathrm{ and }a=\mp@subsup{a}{}{\prime}\mathrm{ and }b=\mp@subsup{b}{}{\prime
            by(erule-tac outputCases, auto simp add: pi.inject residual.inject)+
        with PQTrans show ?thesis by simp
        qed
    qed
next
    show }P||\longmapsto\longmapstoa<x>\prec\mp@subsup{P}{}{\prime}=R\longmapstoa<x>\prec 质
    proof(rule iffI)
        {
        fix x P
        assume P|Q\longmapstoa<x>\prec P' and x\sharpP and x\sharpQ
        hence }R\longmapstoa<x>\prec\mp@subsup{P}{}{\prime
        proof(induct rule: parCasesB)
            case(cPar1 P')
            have P\longmapstoa<x>}\prec\mp@subsup{P}{}{\prime}\mathrm{ by fact
        with Phnf have a<x>. P' 屐 summands P by(simp add: summandTransition)
            moreover have }x\sharpQ\mathrm{ by fact
                    ultimately have a<x>.( (P'|Q) \in expandSet P Q by(auto simp add:
expandSet-def)
            moreover have }a<x>.(\mp@subsup{P}{}{\prime}|Q)\longmapstoa<x>\prec(\mp@subsup{P}{}{\prime}|Q)\mathrm{ by(rule Input)
            ultimately show ?case using Exp by(blast intro: expandAction)
        next
            case(cPar2 Q')
            have }Q\longmapstoa<x>\prec\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
        with Qhnf have a<x>.Q' \in summands Q by(simp add: summandTransition)
```

moreover have $x \sharp P$ by fact
ultimately have $a<x>$. $\left(P \| Q^{\prime}\right) \in$ expandSet $P Q$ by (simp add: expand-Set-def, blast)
moreover have $a<x>.\left(P \| Q^{\prime}\right) \longmapsto a<x>\prec\left(P \| Q^{\prime}\right)$ by (rule Input)
ultimately show ?case using Exp by (blast intro: expandAction)
qed
\}
moreover obtain $y:$ :name where $y \sharp P$ and $y \sharp Q$ and $y \sharp P^{\prime}$
by (generate-fresh name) auto
assume $P \| Q \longmapsto a<x>\prec P^{\prime}$
with $\left\langle y \sharp P^{\prime}\right\rangle$ have $P \| Q \longmapsto a<y>\prec\left([(x, y)] \cdot P^{\prime}\right)$
by (simp add: alphaBoundResidual)
ultimately have $R \longmapsto a<y>\prec\left([(x, y)] \cdot P^{\prime}\right)$ using $\langle y \sharp P\rangle\langle y \sharp Q\rangle$
by auto
thus $R \longmapsto a<x>\prec P^{\prime}$ using $\left\langle y \sharp P^{\prime}\right\rangle$ by (simp add: alphaBoundResidual)
next
assume $R \longmapsto a<x>\prec P^{\prime}$
with Exp obtain $R$ where $R \in \operatorname{expandSet} P Q$ and $R \longmapsto a<x>\prec P^{\prime}$ by (blast dest: expandAction')
thus $P \| Q \longmapsto a<x>\prec P^{\prime}$
proof (auto simp add: expandSet-def)
fix $a^{\prime} y P^{\prime \prime}$
assume $a^{\prime}<y>. P^{\prime \prime} \in$ summands $P$
with Phnf have $P \longmapsto a^{\prime}<y>\prec P^{\prime \prime}$ by (simp add: summandTransition)
moreover assume $y \sharp Q$
ultimately have PQTrans: $P\left\|Q \longmapsto a^{\prime}<y>\prec P^{\prime \prime}\right\| Q$ by (rule Par1B)
assume $a^{\prime}<y>.\left(P^{\prime \prime} \| Q\right) \longmapsto a<x>\prec P^{\prime}$
hence $a<x>\prec P^{\prime}=a^{\prime}<y>\prec P^{\prime \prime} \| Q$ and $a=a^{\prime}$
by (erule-tac inputCases' ${ }^{\prime}$, auto simp add: pi.inject residual.inject) +
with PQTrans show ?thesis by simp
next
fix $a^{\prime} y Q^{\prime}$
assume $a^{\prime}<y>. Q^{\prime} \in$ summands $Q$
with $Q h n f$ have $Q \longmapsto\left(a^{\prime}:: n a m e\right)<y>\prec Q^{\prime} \mathbf{b y}($ simp add: summandTransition)
moreover assume $y \sharp P$
ultimately have PQTrans: $P\left\|Q \longmapsto a^{\prime}<y>\prec P\right\| Q^{\prime}$ by(rule Par2B)
assume $a^{\prime}<y>.\left(P \| Q^{\prime}\right) \longmapsto a<x>\prec P^{\prime}$
hence $a<x>\prec P^{\prime}=a^{\prime}<y>\prec P \| Q^{\prime}$ and $a=a^{\prime}$
by(erule-tac inputCases ${ }^{\prime}$, auto simp add: pi.inject residual.inject) + with PQTrans show ?thesis by simp
qed
qed
next
have Goal: $\bigwedge P Q$ ax $P^{\prime} R . \llbracket(R$, expandSet $P Q) \in$ sumComposeSet; hnf $P$; hnf $Q ; a \neq x \rrbracket \Longrightarrow P \| Q \longmapsto a<\nu x>\prec P^{\prime}=R \longmapsto a<\nu x>\prec P^{\prime}$
proof -
fix $P Q$ a $x P^{\prime} R$
assume aineqx: $(a::$ name $) \neq x$

```
    assume Exp:(R, expandSet P Q) \in sumComposeSet
    assume Phnf:hnf P
    assume Qhnf:hnf Q
```



```
    proof(rule iffI)
    {
        fix x P'
        assume }P|Q\longmapstoa<\nux>\prec\mp@subsup{P}{}{\prime}\mathrm{ and }x\sharpP\mathrm{ and }x\sharpQ\mathrm{ and }a\not=
        hence }R\longmapstoa<\nux>\prec\mp@subsup{P}{}{\prime
        proof(induct rule: parCasesB)
            case(cPar1 P')
            have }P\longmapstoa<\nux>\prec\mp@subsup{P}{}{\prime}\mathrm{ by fact
                with Phnf }\langlea\not=x\rangle\mathrm{ have }<\nux>a{x}.\mp@subsup{P}{}{\prime}\in\mathrm{ summands P by(simp add:
summandTransition)
            moreover have }x\sharpQ\mathrm{ by fact
                ultimately have <\nux>a{x}.(\mp@subsup{P}{}{\prime}|Q)\in expandSet P Q by(auto simp
add: expandSet-def)
```



```
x>
                    by(blast intro: Open Output)
            ultimately show ?case using Exp by(blast intro: expandAction)
        next
            case(cPar2 Q')
            have }Q\longmapstoa<\nux>\prec\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
            with Qhnf }\langlea\not=x\rangle\mathrm{ have < < x >a{x}.Q' 的 summands Q by(simp add:
summandTransition)
            moreover have }x\sharpP\mathrm{ by fact
                    ultimately have <\nux>a{x}.(P| Q') \in expandSet P Q by(simp add:
expandSet-def,blast)
            moreover have <\nux>a{x}.(P|\mp@subsup{Q}{}{\prime})\longmapstoa<\nux> \prec(P| | Q') using <a\not=
x>
                    by(blast intro: Open Output)
            ultimately show ?case using Exp by(blast intro: expandAction)
        qed
        }
        moreover obtain y::name where }y\sharpP\mathrm{ and }y\sharpQ\mathrm{ and }y\sharp\mp@subsup{P}{}{\prime}\mathrm{ and }y\not=
            by(generate-fresh name) auto
    assume P| Q\longmapstoa<\nux>< \prec P'
    with \langley\sharp P'> have }P|Q\longmapstoa<\nuy> \prec ([(x,y)]\cdot\mp@subsup{P}{}{\prime}
            by(simp add: alphaBoundResidual)
        ultimately have }R\longmapstoa<\nuy>\prec ([(x,y)]\cdot P') using\langley\sharpP\rangle\langley\sharpQ\rangle\langley\not
a>
            by auto
        thus }R\longmapstoa<\nux>\prec\prec\mp@subsup{P}{}{\prime}\mathrm{ using <y# 鱼〉 by(simp add:alphaBoundResidual)
        next
    {
        fix R x P 
        assume R\longmapstoa<\nux>}\prec\mp@subsup{P}{}{\prime}\mathrm{ and }R\in\operatorname{expandSet P Q and x\sharpR and x\sharpP
and }x\sharp
```

hence $P \| Q \longmapsto a<\nu x>\prec P^{\prime}$
proof (auto simp add: expandSet-def)
fix $a^{\prime} y P^{\prime \prime}$
assume $<\nu y>a^{\prime}\{y\} . P^{\prime \prime} \in$ summands $P$
moreover hence $a^{\prime} \neq y$ by auto
ultimately have $P \longmapsto a^{\prime}<\nu y>\prec P^{\prime \prime}$ using Phnf by (simp add: summandTransition)
moreover assume $y \sharp Q$
ultimately have $P Q$ Trans: $P\left\|Q \longmapsto a^{\prime}<\nu y>\prec P^{\prime \prime}\right\| Q$ by (rule Par1B)
assume ResTrans: $<\nu y>a^{\prime}\{y\} .\left(P^{\prime \prime} \| Q\right) \longmapsto a<\nu x>\prec P^{\prime}$ and $x \sharp$ [y]. $a^{\prime}\{y\} .\left(P^{\prime \prime} \| Q\right)$
with ResTrans $\left\langle a^{\prime} \neq y\right\rangle\langle x \sharp P\rangle\langle x \sharp Q\rangle$ have $a<\nu x>\prec P^{\prime}=a^{\prime}<\nu y>\prec$ $P^{\prime \prime} \| Q$
apply (case-tac $x=y$ )
defer
apply (erule-tac resCasesB)
apply simp
apply (simp add: abs-fresh)
apply (auto simp add: residual.inject alpha' calc-atm fresh-left abs-fresh elim: outputCases)
$\operatorname{apply}\left(\right.$ ind-cases $\left.<\nu y>a^{\prime}\{y\} .\left(P^{\prime \prime} \| Q\right) \longmapsto a<\nu y>\prec P^{\prime}\right)$
apply (simp add: pi.inject alpha' residual.inject abs-fresh eqvts calc-atm)
apply(auto elim: outputCases)
apply (simp add: pi.inject residual.inject alpha' calc-atm)
apply auto
$\operatorname{apply}\left(\right.$ ind-cases $\left.<\nu y>a^{\prime}\{y\} .\left(P^{\prime \prime} \| Q\right) \longmapsto a<\nu y>\prec P^{\prime}\right)$
apply(auto simp add: pi.inject alpha' residual.inject abs-fresh eqvts
calc-atm)
apply(auto elim: outputCases)
apply(erule-tac outputCases)
apply(auto simp add: freeRes.inject)
apply hypsubst-thin
$\operatorname{apply}(d r u l e-t a c ~ p i=[(b, y)]$ in $p t-b i j 3)$
by $\operatorname{simp}$
with PQTrans show? ?thesis by simp
next
fix $a^{\prime} y Q^{\prime}$
assume $<\nu y>a^{\prime}\{y\} . Q^{\prime} \in$ summands $Q$
moreover hence $a^{\prime} \neq y$ by auto
ultimately have $Q \longmapsto a^{\prime}<\nu y>\prec Q^{\prime}$ using Qhnf by(simp add: summandTransition)
moreover assume $y \sharp P$
ultimately have $P Q$ Trans: $P\left\|Q \longmapsto a^{\prime}<\nu y>\prec P\right\| Q^{\prime}$ by (rule Par2B)
assume ResTrans: $<\nu y>a^{\prime}\{y\} .\left(P \| Q^{\prime}\right) \longmapsto a<\nu x>\prec P^{\prime}$ and $x \sharp[y] . a^{\prime}\{y\} .(P$ $\| Q^{\prime}$ )
with ResTrans $\left\langle a^{\prime} \neq y\right\rangle$ have $a<\nu x>\prec P^{\prime}=a^{\prime}<\nu y>\prec P \| Q^{\prime}$
apply (case-tac $x=y$ )
defer
apply (erule-tac resCasesB)

```
    apply simp
    apply(simp add: abs-fresh)
    apply(auto simp add: residual.inject alpha' calc-atm fresh-left abs-fresh
elim: outputCases)
    apply(ind-cases <\nuy>\mp@subsup{a}{}{\prime}{y}.(P| Q')\longmapstoa<\nuy> \prec P')
    apply(simp add: pi.inject alpha' residual.inject abs-fresh eqvts calc-atm)
    apply(auto elim: outputCases)
    apply(simp add: pi.inject residual.inject alpha' calc-atm)
    apply auto
    apply(ind-cases <\nuy>a}\{y}.(P|\mp@subsup{Q}{}{\prime})\longmapstoa<\nuy> \prec P'
    apply(auto simp add: pi.inject alpha' residual.inject abs-fresh eqvts
calc-atm)
            apply(auto elim: outputCases)
            apply(erule-tac outputCases)
            apply(auto simp add: freeRes.inject)
            apply hypsubst-thin
            apply(drule-tac pi=[(b,y)] in pt-bij3)
            by simp
        with PQTrans show ?thesis by simp
        qed
    }
    moreover assume }R\longmapstoa<\nux>\prec\mp@subsup{P}{}{\prime
    with Exp obtain R where R expandSet P Q and R\longmapstoa<\nux>}\prec\mp@subsup{P}{}{\prime
        apply(drule-tac expandAction') by auto
    moreover obtain y::name where }y\sharpP\mathrm{ and }y\sharpQ\mathrm{ and }y\sharpR\mathrm{ and }y\sharp\mp@subsup{P}{}{\prime
        by(generate-fresh name) auto
    moreover with <y # P'>\langleR\longmapstoa<\nux> \prec P'> have R\longmapstoa<\nuy> \prec ([(x,y)]
- P})\mathrm{ by(simp add: alphaBoundResidual)
    ultimately have }P|Q\longmapstoa<\nuy> \prec([(x,y)]\cdot\mp@subsup{P}{}{\prime})\mathrm{ by auto
```



```
    qed
    qed
    obtain }y\mathrm{ where yineqx: }a\not=y\mathrm{ and yFreshP':y# 疎
    by(force intro: name-exists-fresh[of (a, P
    from Exp Phnf Qhnf yineqx have ( }P|Q\longmapstoa<\nuy> \prec[(x,y)] \cdot P')=(
\longmapstoa<\nuy>\prec[(x,y)]\cdot P')
    by(rule Goal)
    moreover with yFreshP' have x # [(x,y)] \cdot P' by(simp add: name-fresh-left
name-calc)
    ultimately show ( }P|Q\longmapstoa<\nux>\prec \prec P')=(R\longmapstoa<\nux> \prec P'
    by(simp add: alphaBoundResidual name-swap)
qed
lemma expandLeft:
    fixes P :: pi
    and }Q ::p
    and }R\mathrm{ :: pi
    and Rel :: (pi\timespi) set
```

```
    assumes Exp:(R, expandSet P Q) \in sumComposeSet
    and Phnf:hnf P
    and Qhnf:hnf Q
    and Id:Id\subseteq Rel
    shows P|Q\rightsquigarrow[Rel] R
proof(induct rule: simCases)
    case(Bound a x R')
    have }R\longmapstoa<x> \prec R' by fac
    with Exp Phnf Qhnf have P|Q Q\longmapstoa<x» \prec R' by(cases a, auto simp add:
expandTrans)
    moreover from Id have derivative R' R' a x Rel by(cases a, auto simp add:
derivative-def)
    ultimately show ?case by blast
next
    case(Free \alpha R')
    have }R\longmapsto\alpha\prec\mp@subsup{R}{}{\prime}\mathrm{ by fact
    with Exp Phnf Qhnf have P|Q\longmapsto\alpha\prec 林 by(cases \alpha, auto simp add: ex-
pandTrans)
    moreover from Id have ( }\mp@subsup{R}{}{\prime},\mp@subsup{R}{}{\prime})\inRel by blas
    ultimately show ?case by blast
qed
lemma expandRight:
    fixes P :: pi
    and }Q :: p
    and }R\mathrm{ :: pi
    and Rel :: (pi\timespi) set
    assumes Exp:(R, expandSet P Q) \in sumComposeSet
    and Phnf:hnf P
    and Qhnf:hnf Q
    and Id:Id\subseteqRel
    shows R\rightsquigarrow[Rel] P|Q
proof(induct rule: simCases)
    case(Bound a x R')
    have P|Q\longmapstoa<x» \prec R' by fact
    with Exp Phnf Qhnf have R\longmapstoa«x» \prec R' by(cases a, auto simp add: expand-
Trans)
    moreover from Id have derivative 的 R' a x Rel by(cases a, auto simp add:
derivative-def)
    ultimately show ?case by blast
next
    case(Free \alpha R ')
    have P|Q\longmapsto\alpha\prec R' by fact
    with Exp Phnf Qhnf have R\longmapsto\alpha\prec R' by (cases \alpha, auto simp add: expandTrans)
    moreover from Id have ( }\mp@subsup{R}{}{\prime},\mp@subsup{R}{}{\prime})\inRel by blas
```

```
    ultimately show ?case by blast
qed
lemma expandSC:
    fixes P :: pi
    and }Q:: p
    and }R::p
    assumes (R, expandSet P Q)\in sumComposeSet
    and hnfP
    and hnf Q
    shows P|Q~R
proof -
    let ?X = {(P|Q,R)|PQ R. (R, expandSet P Q ) \in sumComposeSet ^ hnf P
^hnf Q} \cup{(R,P|Q)|PQR.(R, expandSet P Q ) { sumComposeSet ^ hnf P
\wedge hnf Q}
    from assms have (P|Q,R)\in?X by auto
    thus ?thesis
    proof(coinduct rule: bisimCoinduct)
    case(cSim P Q)
    thus ?case
            by(blast intro: reflexive expandLeft expandRight)
    next
    case(cSym P Q)
            thus ?case by auto
    qed
qed
end
theory Strong-Late-Axiomatisation
    imports Strong-Late-Expansion-Law
begin
lemma inputSuppPres:
    fixes }P:: p
    and }Q:: p
    and x :: name
    and Rel :: (pi\times pi) set
    assumes PRelQ: \bigwedgey. y \in supp (P,Q,x)\Longrightarrow(P[x::=y],Q[x::=y])\in\operatorname{Rel}
    and Eqvt: eqvt Rel
    shows }a<x>.P\rightsquigarrow[Rel] a<x>.Q
proof -
    from Eqvt show ?thesis
    proof(induct rule: simCasesCont[where C=(x,a,Q,P)])
    case(Bound b y Q')
```

```
    have }x\in\operatorname{supp}(P,Q,x)\mathbf{by}(\operatorname{simp}\mathrm{ add: supp-prod supp-atm)
    with PRelQ have (P,Q)\inRel by fastforce
    have QTrans: a<x>.Q \longmapstob«y» \prec Q' by fact
    have }y\sharp(x,a,Q,P)\mathrm{ by fact
    hence }y\not=a\mathrm{ and yineqx: }y\not=x\mathrm{ and }y\sharpQ\mathrm{ and }y\sharpP\mathrm{ by(simp add: fresh-prod)+
    with QTrans show ?case
    proof(induct rule: inputCases)
    have }a<y>.([(x,y)]\cdotP)\longmapstoa<y>\prec ([(x,y)]\cdotP) by(rule Input
        hence }a<x>.P\longmapstoa<y>\prec \prec [[(x,y)] \cdot P) using〈y\sharpP> by(simp add
alphaInput)
    moreover have derivative ([(x,y)] | P) ([(x,y)] •Q)(InputS a) y Rel
    proof(auto simp add: derivative-def)
        fix u
        have }x\in\operatorname{supp}(P,Q,x)\mathbf{by}(\mathrm{ simp add: supp-prod supp-atm)
        have (P[x::=u],Q[x::=u]) \in Rel
        proof(cases u\in\operatorname{supp}(P,Q,x))
            case True
            with PRelQ show ?thesis by auto
        next
            case False
            hence }u\sharpP\mathrm{ and }u\sharpQ\mathbf{by}(auto simp add: fresh-def supp-prod
                moreover from <eqvt Rel\rangle\langle(P,Q) \in Rel\rangle have ([(x,u)] • P, [(x,u)] .
Q) }\in\mathrm{ Rel
                by(rule eqvtRelI)
                ultimately show ?thesis by(simp only: injPermSubst)
        qed
        with \langley\sharpP>\langley\sharpQ\rangle\mathrm{ show }(([(x,y)] \cdot P)[y::=u],([(x,y)]|Q)[y::=u])\in
Rel
            by(simp add: renaming)
        qed
```



```
Q) (InputS a) y Rel
            by blast
        qed
    next
        case(Free \alpha Q')
        have }a<x>.Q\longmapsto\alpha\prec\mp@subsup{Q}{}{\prime}\mathrm{ by fact
        hence False by auto
        thus ?case by blast
    qed
qed
lemma inputSuppPresBisim:
    fixes P :: pi
    and }Q::p
    and x :: name
assumes PSimQ: \bigwedgey. y \in supp (P,Q,x)\LongrightarrowP[x::=y] ~ Q[x::=y]
```

```
    shows }a<x>.P~a<x>.
proof -
    let ?X = {(a<x>.P,a<x>.Q)| a x P Q. \forally \in supp(P,Q,x). P[x::=y] ~
Q[x::=y]}
    have eqvt?X
        apply(auto simp add: eqvt-def)
        apply(rule-tac x=perma • aa in exI)
        apply(rule-tac x=perma \cdotx in exI)
        apply(rule-tac x=perma \cdot P in exI)
        apply auto
        apply(rule-tac x=perma •Q in exI)
        apply auto
    apply(drule-tac pi=rev perma in pt-set-bij2[OF pt-name-inst, OF at-name-inst])
        apply(simp add: eqvts pt-rev-pi[OF pt-name-inst, OF at-name-inst])
        apply(erule-tac x=rev perma \cdot y in ballE)
        apply auto
        apply(drule-tac p=perma in bisimClosed)
    by(simp add: eqvts pt-pi-rev[OF pt-name-inst, OF at-name-inst])
    from assms have ( }a<x>.P,a<x>.Q)\in?X by fastforc
    thus ?thesis
    proof(coinduct rule: bisimCoinduct)
        case(cSim P Q)
        thus ?case using <eqvt?X>
        by(fastforce intro: inputSuppPres)
    next
        case(cSym P Q)
        thus ?case by(fastforce simp add: supp-prod dest: symmetric)
    qed
qed
inductive equiv :: pi => pi => bool (infixr \equiv}\mp@subsup{\equiv}{e}{}80
where
    Refl: P}\equiv\mp@subsup{\equiv}{e}{}
| Sym: }\quadP\equiv\mp@subsup{\equiv}{e}{}Q\LongrightarrowQ\mp@subsup{\equiv}{e}{}
| Trans: }\quad\llbracketP\mp@subsup{\equiv}{e}{}Q;Q\mp@subsup{\equiv}{e}{}R\rrbracket\LongrightarrowP\mp@subsup{\equiv}{e}{}
| Match: }\quad[a\frowna]P\equiv\mp@subsup{\equiv}{e}{}
| Match': }\quada\not=b\Longrightarrow[a\frownb]P\equiv\mp@subsup{\equiv}{e}{}\mathbf{0
| Mismatch: }\quada\not=b\Longrightarrow[a\not=b]P\equiv\mp@subsup{\equiv}{e}{}
| Mismatch': }\quad[a\not=a]P\mp@subsup{\equiv}{e}{}\mathbf{0
| SumSym: }\quadP\oplusQ\equiv\mp@subsup{\equiv}{e}{}Q\oplus
| SumAssoc: }\quad(P\oplusQ)\oplusR=\mp@subsup{\equiv}{e}{}P\oplus(Q\oplusR
| SumZero: }\quadP\oplus\mathbf{0}\equiv\mp@subsup{}{e}{}
| SumIdemp: }\quadP\oplusP\equiv\mp@subsup{\equiv}{e}{}
| SumRes: }\quad<\nux>(P\oplusQ)\mp@subsup{\equiv}{e}{}(<\nux>P)\oplus(<\nux>Q
| ResNil: }\quad<\nux>\mathbf{0}\equiv\mp@subsup{\equiv}{e}{}\mathbf{0
```

```
| ResInput: \(\quad \llbracket x \neq a ; x \neq y \rrbracket \Longrightarrow<\nu x>a<y>. P \equiv_{e} a<y>.(<\nu x>P)\)
| ResInput': \(\quad<\nu x>x<y>. P \equiv{ }_{e} \mathbf{0}\)
| ResOutput: \(\quad \llbracket x \neq a ; x \neq b \rrbracket \Longrightarrow<\nu x>a\{b\} \cdot P \equiv_{e} a\{b\} \cdot(<\nu x>P)\)
| ResOutput': \(\quad<\nu x>x\{b\} . P \equiv_{e} \mathbf{0}\)
ResTau: \(\quad<\nu x>\tau .(P) \equiv_{e} \tau .(<\nu x>P)\)
ResComm: \(\quad<\nu x><\nu y>P \equiv_{e}<\nu y><\nu x>P\)
ResFresh: \(\quad x \sharp P \Longrightarrow<\nu x>P \equiv_{e} P\)
| Expand: \(\quad \llbracket(R\), expandSet \(P Q) \in\) sumComposeSet; hnf \(P ;\) hnf \(Q \rrbracket \Longrightarrow P\)
\(\| Q \equiv{ }_{e} R\)
\(\begin{array}{ll}\mid \text { SumPres: } & P \equiv_{e} Q \Longrightarrow P \oplus R \equiv_{e} Q \oplus R \\ \text { | ParPres: } & \llbracket P \equiv_{e} P^{\prime} Q \equiv_{e} Q \rrbracket \Longrightarrow P\left\|Q \equiv_{e} P^{\prime}\right\| Q^{\prime} \\ \text { | ResPres: } & P \equiv_{e} Q \Longrightarrow<\nu>P \equiv_{e}<\nu x>Q \\ \text { | TauPres: } & P \equiv_{e} Q \Longrightarrow \tau \cdot(P) \equiv_{e} \tau \cdot(Q) \\ \text { OutputPres: } & P \equiv_{e} Q \Longrightarrow a\{b\} \cdot P \equiv_{e} a\{b\} \cdot Q \\ \mid \text { InputPres: } & \llbracket \forall \in \operatorname{supp}(P, Q, x) \cdot P[x::=y] \equiv_{e} Q[x::=y] \rrbracket \Longrightarrow a<x>\cdot P \equiv_{e}\end{array}\)
lemma SumIdemp \({ }^{\prime}\) :
    fixes \(P\) :: \(p i\)
    and \(P^{\prime}:: p i\)
    assumes \(P \equiv{ }_{e} P^{\prime}\)
    shows \(P \oplus P^{\prime} \equiv{ }_{e} P\)
proof -
    have \(P \equiv_{e} P \oplus P\) by(blast intro: Sym SumIdemp)
    moreover from assms have \(P \oplus P \equiv{ }_{e} P^{\prime} \oplus P\) by (rule SumPres)
    moreover have \(P^{\prime} \oplus P \equiv{ }_{e} P \oplus P^{\prime}\) by (rule SumSym)
    ultimately have \(P \equiv_{e} P \oplus P^{\prime}\) by (blast intro: Trans)
    thus ?thesis by (rule Sym)
qed
lemma SumPres':
    fixes \(P:: p i\)
    and \(\quad P^{\prime}:: p i\)
    and \(Q:: p i\)
    and \(\quad Q^{\prime}:: p i\)
    assumes \(P e q P^{\prime}: P \equiv{ }_{e} P^{\prime}\)
    and \(Q e q Q^{\prime}: Q \equiv{ }_{e} Q^{\prime}\)
    shows \(P \oplus Q \equiv{ }_{e} P^{\prime} \oplus Q^{\prime}\)
proof -
    from PeqP' have \(P \oplus Q \equiv{ }_{e} P^{\prime} \oplus Q\) by(rule SumPres)
    moreover have \(P^{\prime} \oplus Q \equiv_{e} Q \oplus P^{\prime}\) by(rule SumSym)
    moreover from \(Q e q Q^{\prime}\) have \(Q \oplus P^{\prime} \equiv_{e} Q^{\prime} \oplus P^{\prime}\) by (rule SumPres)
    moreover have \(Q^{\prime} \oplus P^{\prime} \equiv{ }_{e} P^{\prime} \oplus Q^{\prime}\) by (rule SumSym)
```

```
    ultimately show ?thesis by(blast intro: Trans)
qed
lemma sound:
    fixes P :: pi
    and }Q:: p
    assumes P}\equiv\mp@subsup{}{e}{}
    shows P~Q
using assms
proof(induct)
    case(Refl P)
    show ?case by(rule reflexive)
next
    case(Sym P Q)
    have P~Q by fact
    thus ?case by(rule symmetric)
next
    case(Trans P Q R)
    have }P~Q\mathrm{ and Q }~R\mathrm{ by fact+
    thus ?case by(rule transitive)
next
    case(Match a P)
    show ?case by(rule matchId)
next
    case(Match' a b P)
    have }a\not=b\mathrm{ by fact
    thus ?case by(rule matchNil)
next
    case(Mismatch a b P)
    have }a\not=b\mathrm{ by fact
    thus ?case by(rule mismatchId)
next
    case(Mismatch' a P)
    show ?case by(rule mismatchNil)
next
    case(SumSym P Q)
    show ?case by(rule sumSym)
next
    case(SumAssoc P Q R)
    show ?case by(rule sumAssoc)
next
    case(SumZero P)
    show ?case by(rule sumZero)
next
    case(SumIdemp P)
    show ?case by(rule sumIdemp)
next
```

```
    case(SumRes x P Q)
    show ?case by(rule sumRes)
next
    case(ResNil x)
    show ?case by(rule nilRes)
next
    case(ResInput x a y P)
    have }x\not=a\mathrm{ and }x\not=y\mathrm{ by fact+
    thus ?case by(rule resInput)
next
    case(ResInput' x y P)
    show ?case by(rule resNil)
next
    case(ResOutput x a b P)
    have }x\not=a\mathrm{ and }x\not=b\mathrm{ by fact+
    thus ?case by(rule resOutput)
next
    case(ResOutput' x b P)
    show ?case by(rule resNil)
next
    case(ResTau x P)
    show ?case by(rule resTau)
next
    case(ResComm x P)
    show ?case by(rule resComm)
next
    case(ResFresh x P)
    have }x\sharpP\mathrm{ by fact
    thus ?case by(rule scopeFresh)
next
    case(Expand R P Q)
    have (R, expandSet P Q) \in sumComposeSet and hnf P and hnf Q by fact+
    thus ?case by(rule expandSC)
next
    case(SumPres P Q R)
    from <P~ Q\rangle show ?case by(rule sumPres)
next
    case(ParPres P P' Q Q')
    have }P~\mp@subsup{P}{}{\prime}\mathrm{ and }Q~\mp@subsup{Q}{}{\prime}\mathrm{ by fact+
    thus ?case by(metis transitive symmetric parPres parSym)
next
    case(ResPres P Q x)
    from }\langleP~Q\rangle\mathrm{ show ?case by(rule resPres)
next
    case(TauPres P Q)
    from }\langleP~Q\rangle\mathrm{ show ?case by(rule tauPres)
next
    case(OutputPres P Q a b)
    from }\langleP~Q\rangle\mathrm{ show ?case by(rule outputPres)
```

```
next
    case(InputPres \(P Q x a)\)
    have \(\forall y \in \operatorname{supp}(P, Q, x) . P[x::=y] \equiv_{e} Q[x::=y] \wedge P[x::=y] \sim Q[x::=y]\) by fact
    hence \(\forall y \in \operatorname{supp}(P, Q, x) . P[x::=y] \sim Q[x::=y]\) by blast
    thus ?case by (rule-tac inputSuppPresBisim) auto
qed
lemma zeroDest[dest]:
    fixes \(a\) :: name
    and \(b::\) name
    and \(x::\) name
    and \(\quad P:: p i\)
    shows \((a\{b\} . P) \equiv_{e} \mathbf{0} \Longrightarrow\) False
    and \((a<x>. P) \equiv_{e} \mathbf{0} \Longrightarrow\) False
    and \(\quad(\tau .(P)) \equiv_{e} \mathbf{0} \Longrightarrow\) False
    and \(\quad \mathbf{0} \equiv_{e} a\{b\} . P \Longrightarrow\) False
    and \(\mathbf{0} \equiv_{e} a<x>. P \Longrightarrow\) False
    and \(\quad \mathbf{0} \equiv_{e} \tau .(P) \Longrightarrow\) False
by(auto dest: sound)
lemma eq-equt:
    fixes pi::name prm
    and \(x:: ' a:: p t-n a m e\)
    shows \(p i \cdot(x=y)=((p i \cdot x)=(p i \cdot y))\)
by (simp add: perm-bool perm-bij)
nominal-primrec depth \(:: p i \Rightarrow\) nat where
    depth \(\mathbf{0}=0\)
depth \((\tau .(P))=1+(\) depth \(P)\)
\(a \sharp x \Longrightarrow\) depth \((a<x>. P)=1+(\) depth \(P)\)
depth \((a\{b\} . P)=1+(\) depth \(P)\)
depth \(([a \frown b] P)=(\) depth \(P)\)
depth \(([a \neq b] P)=(\operatorname{depth} P)\)
depth \((P \oplus Q)=\max (\) depth \(P)(\) depth \(Q)\)
\(\mid\) depth \((P \| Q)=((\operatorname{depth} P)+(\operatorname{depth} Q))\)
depth \((<\nu x>P)=(\) depth \(P)\)
\(\mid\) depth \((!P)=(\) depth \(P)\)
apply (auto simp add: fresh-nat)
apply(finite-guess)+
by(fresh-guess)+
lemma depthEqvt[simp]:
    fixes \(P\) :: \(p i\)
    and \(p::\) name prm
    shows \(\operatorname{depth}(p \cdot P)=\operatorname{depth} P\)
by(nominal-induct \(P\) rule: pi.strong-induct, auto simp add: name-bij)
```

```
lemma depthInput[simp]:
    fixes a :: name
    and }x:: nam
    and }P::p
    shows depth (a<x>.P)=1+(depth P)
proof -
    obtain }y\mathrm{ where yineqa: }y\not=a\mathrm{ and yFreshP: y }\sharp
        by(force intro: name-exists-fresh[of (a,P)] simp add: fresh-prod)
    from yFreshP have a<x>.P=a<y>.([(x,y)] P P) by(simp add: alphaInput)
    with yineqa show ?thesis by simp
qed
nominal-primrec valid :: pi => bool where
    valid 0}=\mathrm{ True
|valid (\tau. (P)) = valid P
| }||=\mathrm{ valid ( }a<x>.P)=valid 
valid (a{b}.P) = valid P
valid ([a\frownb]P) = valid P
valid ([a\not=b]P)= valid P
valid }(P\oplusQ)=((valid P)\wedge(valid Q)
valid }(P|Q)=((valid P)\wedge(valid Q)
valid ( <\nux>P) = valid P
valid (!P) = False
apply(auto simp add: fresh-bool)
apply(finite-guess)+
by(fresh-guess)+
lemma validEqvt[simp]:
    fixes P :: pi
    and p :: name prm
    shows valid ( }p\cdotP)=\operatorname{valid}
by(nominal-induct P rule: pi.strong-induct, auto simp add: name-bij)
lemma validInput[simp]:
    fixes a :: name
    and x :: name
    and }P::p
    shows valid ( }a<x>.P)=valid 
proof -
    obtain }y\mathrm{ where yineqa: }y\not=a\mathrm{ and yFreshP: y#P
    by(force intro: name-exists-fresh[of (a,P)] simp add: fresh-prod)
    from yFreshP have a<x>.P=a<y>.([(x,y)]\cdotP) by(simp add: alphaInput)
    with yineqa show ?thesis by simp
qed
```

```
lemma depthMin[intro]
    fixes P
    shows 0 \leq depth P
by(induct P rule: pi.induct, auto)
lemma hnfTransition:
    fixes P :: pi
    assumes hnf P
    and }P\not=\mathbf{0
    shows }\existsRs.P\longmapstoR
using assms
by(induct rule: pi.induct, auto intro: Output Tau Input Sum1 Open)
definition uhnf :: pi => bool where
    uhnf P}\equiv\operatorname{hnf}P\wedge(\forallR\in\mathrm{ summands P.}\forall\mp@subsup{R}{}{\prime}\in\mathrm{ summands }P.R\not=\mp@subsup{R}{}{\prime}\longrightarrow\neg(
\equiv
lemma summandsIdemp:
    fixes P :: pi
    and }Q::p
    assumes Q\in summands P
    and }Q\mp@subsup{\equiv}{e}{}\mp@subsup{Q}{}{\prime
    shows }P\oplus\mp@subsup{Q}{}{\prime}\equiv\mp@subsup{}{e}{}
using assms
proof(nominal-induct P arbitrary: Q rule: pi.strong-inducts)
    case(PiNil Q)
    have Q}\in\mathrm{ summands 0 by fact
    hence False by simp
    thus ?case by simp
next
    case(Output a b P Q)
    have Q \equiv
    hence }a{b}.P\oplus\mp@subsup{Q}{}{\prime}\mp@subsup{\equiv}{e}{}a{b}.P\oplusQ\mathrm{ by(blast intro:SumPres' Refl Sym)
    moreover have Q =a{b}.P
    proof -
        have }Q\in\mathrm{ summands (a{b}.P) by fact
        thus ?thesis by simp
    qed
    ultimately show ?case by(blast intro: SumIdemp Trans)
next
    case(Tau P Q)
    have Q \equiv
    hence }\tau.(P)\oplus\mp@subsup{Q}{}{\prime}\mp@subsup{\equiv}{e}{}\tau.(P)\oplusQ\mathrm{ by(blast intro:SumPres' Refl Sym)
    moreover have Q =\tau.(P)
```

```
    proof -
    have Q &ummands (\tau.(P)) by fact
    thus ?thesis by simp
    qed
    ultimately show ?case by(blast intro: SumIdemp Trans)
next
    case(Input a x P Q)
    have Q}\equiv\mp@subsup{}{e}{}\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    hence }a<x>.P\oplus\mp@subsup{Q}{}{\prime}\mp@subsup{\equiv}{e}{}a<x>.P\oplusQ\mathrm{ by(blast intro:SumPres' Refl Sym)
    moreover have Q =a<x>.P
    proof -
        have Q < summands ( }a<x>.P)\mathrm{ by fact
        thus ?thesis by simp
    qed
    ultimately show ?case by(blast intro: SumIdemp Trans)
next
    case(Match a b P Q)
    have Q summands ([a\frownb]P) by fact
    hence False by simp
    thus ?case by simp
next
    case(Mismatch a b P Q)
    have}Q\in\mathrm{ summands ([a#b]P) by fact
    hence False by simp
    thus ?case by simp
next
    case(Sum P Q R)
    have IHP: \bigwedgeP'. \llbracketP'的 summands P; P' \equiv
    have IHQ: \bigwedge\mp@subsup{Q}{}{\prime\prime}.\llbracket\mp@subsup{Q}{}{\prime\prime}\in\mathrm{ summands }Q;\mp@subsup{Q}{}{\prime\prime}\mp@subsup{\equiv}{e}{}Q\rrbracket\LongrightarrowQ\oplus\mp@subsup{Q}{}{\prime}\mp@subsup{\equiv}{e}{}Q\mathrm{ by fact}
    have Req\mp@subsup{Q}{}{\prime}:R}\mp@subsup{\equiv}{e}{}\mp@subsup{Q}{}{\prime}\mathrm{ by fact
    have}R\in\operatorname{summands(P\oplusQ) by fact
    hence R}\in\mathrm{ summands }P\veeR\in\mathrm{ summands Q by simp
    thus ?case
    proof(rule disjE)
    assume R 堆mands P
    hence }P\mp@subsup{Q}{}{\prime}eqP:P\oplus\mp@subsup{Q}{}{\prime}\equiv\mp@subsup{\equiv}{e}{}P\mathrm{ using ReqQ' by(rule IHP)
    have }(P\oplusQ)\oplus\mp@subsup{Q}{}{\prime}\mp@subsup{\equiv}{e}{}P\oplus(Q\oplus\mp@subsup{Q}{}{\prime})\mathrm{ by(rule SumAssoc)
    moreover have }P\oplus(Q\oplus\mp@subsup{Q}{}{\prime})\equiv\mp@subsup{}{e}{}P\oplus(\mp@subsup{Q}{}{\prime}\oplusQ)\mathrm{ by(blast intro: Refl SumSym
SumPres')
    moreover have P}\oplus(\mp@subsup{Q}{}{\prime}\oplusQ)\mp@subsup{\equiv}{e}{}(P\oplus\mp@subsup{Q}{}{\prime})\oplusQ\mathrm{ by(blast intro: SumAssoc
Sym)
    moreover from PQ'eqP have ( }P\oplus\mp@subsup{Q}{}{\prime})\oplusQ\mp@subsup{\equiv}{e}{}P\oplusQ\mathrm{ by(blast intro:
SumPres' Refl)
    ultimately show ?case by(blast intro: Trans)
    next
    assume R 堆mands Q
    hence QQ'eqQ:Q Q Q Q' \equiv
```

```
    have }(P\oplusQ)\oplus\mp@subsup{Q}{}{\prime}\mp@subsup{\equiv}{e}{}PP\oplus(Q\oplus\mp@subsup{Q}{}{\prime})\mathbf{by}(rule SumAssoc
    moreover from QQ'eqQ have P}\oplus(Q\oplus\mp@subsup{Q}{}{\prime})\mp@subsup{\equiv}{e}{}P\oplusQ\mathrm{ by(blast intro:Refl
SumPres')
    ultimately show ?case by(rule Trans)
    qed
next
    case(Par P Q R)
    have R}\in\mathrm{ summands (P|Q) by fact
    hence False by simp
    thus ?case by simp
next
    case(Res x P Q)
    have Q \equiv
    hence (<\nux>P)\oplus Q' \equiv
    moreover have Q = <\nux>P
    proof -
    have }Q\in\mathrm{ summands (< < x>P) by fact
    thus ?thesis by(auto simp add: if-split)
    qed
    ultimately show ?case by(blast intro: SumIdemp Trans)
next
    case(Bang P Q)
    have Q\in summands(!P) by fact
    hence False by simp
    thus ?case by simp
qed
lemma uhnfSum:
    fixes }P::p
    and }Q::p
    assumes Phnf:uhnf P
    and Qhnf:uhnf Q
    and validP: valid P
    and validQ: valid Q
    shows \existsR. uhnf R ^ valid R}\wedgeP\oplusQ \equiv\mp@subsup{e}{e}{}R\wedge(\operatorname{depth}R)\leq(\operatorname{depth}(P\oplusQ)
using assms
proof(nominal-induct P arbitrary:Q rule: pi.strong-inducts)
    case(PiNil Q)
    have uhnf Q by fact
    moreover have valid Q by fact
    moreover have 0}\oplusQ\mp@subsup{\equiv}{e}{}Q\mathrm{ by(blast intro: SumZero SumSym Trans)
    moreover have depth Q depth(0}\oplusQ) by aut
    ultimately show ?case by blast
next
    case(Output a b P Q)
    show ?case
    proof(case-tac Q = 0)
```

```
    assume Q = 0
    moreover have uhnf (a{b}.P) by(simp add:uhnf-def)
    moreover have valid (a{b}.P) by fact
    moreover have a{b}.P\oplus\mathbf{0}\mp@subsup{\equiv}{e}{}a{b}.P by(rule SumZero)
    moreover have depth(a{b}.P)\leq\operatorname{depth}(a{b}.P\oplus\mathbf{0})\mathrm{ by simp}
    ultimately show ?case by blast
next
    assume QineqNil: Q}\not=\mathbf{0
    have Qhnf:uhnf Q by fact
    have validQ: valid Q by fact
    have validP: valid(a{b}.P) by fact
    show ?case
    proof(case-tac \exists\mp@subsup{Q}{}{\prime}\in\mathrm{ summands Q. Q' 洊a{b}.P)})
        assume }\exists\mp@subsup{Q}{}{\prime}\in\mathrm{ summands }Q.\mp@subsup{Q}{}{\prime}\mp@subsup{\equiv}{e}{}a{b}.
        then obtain Q' where Q'\in summands Q and }\mp@subsup{Q}{}{\prime}\mp@subsup{\equiv}{e}{}a{b}.P\mathrm{ by blast
        hence }Q\oplusa{b}.P\mp@subsup{\equiv}{e}{}Q\mathbf{by}(rule summandsIdemp
        moreover have depth Q}\leq\operatorname{depth}(Q\oplusa{b}.P) by sim
        ultimately show ?case using Qhnf validQ by(force intro: SumSym Trans)
    next
        assume}\neg(\exists\mp@subsup{Q}{}{\prime}\in\mathrm{ summands Q. Q' 洊 a{b}.P)
        hence }\forall\mp@subsup{Q}{}{\prime}\in\mathrm{ summands Q. }\neg(\mp@subsup{Q}{}{\prime}\mp@subsup{\equiv}{e}{}a{b}.P)\mathrm{ by simp
        with Qhnf QineqNil have uhnf (a{b}.P \oplusQ)
            by(force dest: Sym simp add: uhnf-def)
    moreover from validQ validP have valid (a{b}.P }\oplusQ) by sim
    moreover have }a{b}.P\oplusQ\mp@subsup{\equiv}{e}{}a{b}.P\oplusQ\mathrm{ by(rule Refl)
    moreover have depth(a{b}.P\oplusQ)\leq\operatorname{depth}(a{b}.P\oplusQ) by simp
    ultimately show ?case by blast
    qed
    qed
next
    case(Tau P Q)
    show ?case
    proof(case-tac Q = 0)
        assume Q =0
        moreover have uhnf (\tau.(P)) by(simp add: uhnf-def)
    moreover have valid (\tau.(P)) by fact
    moreover have }\tau.(P)\oplus\mathbf{0}\mp@subsup{\equiv}{e}{}\tau.(P)\mathrm{ by(rule SumZero)
    moreover have depth}(\tau.(P))\leq\operatorname{depth}(\tau.(P)\oplus\mathbf{0})\mathrm{ by simp
    ultimately show ?case by blast
next
    assume QineqNil: Q =0
    have Qhnf:uhnf Q by fact
    have validP: valid(\tau.(P)) and validQ: valid Q by fact+
    show ?case
    proof(case-tac \exists}\mp@subsup{Q}{}{\prime}\in\mathrm{ summands Q. Q' }\mp@subsup{\equiv}{e}{}\tau.(P)
        assume }\exists\mp@subsup{Q}{}{\prime}\in\mathrm{ summands Q. Q 䬶 }\tau.(P
        then obtain Q}\mp@subsup{Q}{}{\prime}\mathrm{ where }\mp@subsup{Q}{}{\prime}\in\mathrm{ summands Q and }\mp@subsup{Q}{}{\prime}\mp@subsup{\equiv}{e}{}\tau.(P)\mathrm{ by blast
        hence }Q\oplus\tau.(P) \equiv\mp@subsup{e}{e}{}Q\mathbf{by}(rule summandsIdemp
        moreover have depth Q}\leq\operatorname{depth}(Q\oplus\tau.(P)) by sim
```

```
        ultimately show ?case using Qhnf validQ by(force intro: SumSym Trans)
    next
        assume }\neg(\exists\mp@subsup{Q}{}{\prime}\in\mathrm{ summands }Q.\mp@subsup{Q}{}{\prime}\mp@subsup{\equiv}{e}{}\tau.(P)
        hence }\forall\mp@subsup{Q}{}{\prime}\in\mathrm{ summands }Q.\neg(\mp@subsup{Q}{}{\prime}\mp@subsup{\equiv}{e}{}\tau.(P))\mathrm{ by simp
        with Qhnf QineqNil have uhnf (\tau.(P)\oplusQ)
            by(force dest: Sym simp add: uhnf-def)
            moreover from validP validQ have valid (\tau.(P)\oplusQ) by simp
            moreover have }\tau.(P)\oplusQ\mp@subsup{\equiv}{e}{}\tau.(P)\oplusQ\mathrm{ by(rule Refl)
            moreover have depth(\tau.(P)\oplusQ)\leq\operatorname{depth}(\tau.(P)\oplusQ) by simp
            ultimately show ?case by blast
    qed
    qed
next
    case(Input a x P Q)
    show ?case
    proof(case-tac Q = 0)
        assume Q = 0
        moreover have uhnf (a<x>.P) by(simp add: uhnf-def)
        moreover have valid ( }a<x>.P)\mathrm{ by fact
        moreover have a<x>.P}\oplus\mathbf{0}\mp@subsup{\equiv}{e}{}a<x>.P by(rule SumZero
        moreover have depth (a<x>.P)\leq\operatorname{depth}(a<x>.P\oplus\mathbf{0})\mathrm{ by simp}
        ultimately show ?case by blast
    next
        assume QineqNil: Q\not=\mathbf{0}
        have validP: valid(a<x>.P) and validQ: valid Q by fact+
        have Qhnf:uhnf Q by fact
        show ?case
        proof(case-tac \exists Q ' }\in\mathrm{ summands Q. Q' 泎 a<x>.P)
            assume }\exists\mp@subsup{Q}{}{\prime}\in\mathrm{ summands Q. Q 㐌e}a<x>.
            then obtain }\mp@subsup{Q}{}{\prime}\mathrm{ where }\mp@subsup{Q}{}{\prime}\in\mathrm{ summands Q and }\mp@subsup{Q}{}{\prime}\mp@subsup{\equiv}{e}{}a<x>.P by blas
            hence }Q\oplusa<x>.P \mp@subsup{\equiv}{e}{}Q\mathrm{ by(rule summandsIdemp)
            moreover have depth Q
            ultimately show ?case using Qhnf validQ by(force intro: SumSym Trans)
    next
            assume }\neg(\exists\mp@subsup{Q}{}{\prime}\in\mathrm{ summands Q. Q' }\mp@subsup{\equiv}{e}{}a<x>.P
            hence }\forall\mp@subsup{Q}{}{\prime}\in\mathrm{ summands Q. ᄀ( Q' 洊a<x>.P) by simp
            with Qhnf QineqNil have uhnf ( }a<x>.P\oplusQ
                by(force dest: Sym simp add: uhnf-def)
            moreover from validP validQ have valid ( }a<x>.P\oplusQ)\mathrm{ by simp
            moreover have }a<x>.P\oplusQ\mp@subsup{\equiv}{e}{}a<x>.P\oplusQ\mathrm{ by(rule Refl)
            moreover have depth(a<x>.P\oplusQ)\leqdepth (a<x>.P }\oplusQ)\mathrm{ by simp
            ultimately show ?case by blast
        qed
    qed
next
    case(Match a b P Q)
    have uhnf ([a\frownb]P) by fact
    hence False by(simp add: uhnf-def)
    thus ?case by simp
```

```
next
    case(Mismatch a b P Q)
    have uhnf ([a\not=b]P) by fact
    hence False by(simp add: uhnf-def)
    thus ?case by simp
next
    case(Sum P Q R)
    have Rhnf:uhnf R by fact
    have validR: valid R by fact
    have PQhnf: uhnf ( }P\oplusQ)\mathrm{ by fact
    have validPQ: valid (P\oplusQ) by fact
    have }\exists\mathrm{ T. uhnf T^valid T^Q }\oplusR\equiv\mp@subsup{\equiv}{e}{}T\wedge\mathrm{ depth }T\leq\operatorname{depth}(Q\oplusR
    proof -
    from PQhnf have uhnf Q by(simp add:uhnf-def)
    moreover from validPQ have valid Q by simp+
    moreover have }\wedgeR\mathrm{ . «uhnf Q; uhnf R;valid Q; valid R】 ב 
valid T}\wedgeQ\oplusR\mp@subsup{\equiv}{e}{}T\wedge\mathrm{ depth T}\leq\operatorname{depth}(Q\oplusR)\mathrm{ by fact
    ultimately show ?thesis using Rhnf validR by blast
    qed
    then obtain T where Thnf:uhnf T and QReqT: Q \oplus R \equiv
valid T
                                    and Tdepth: depth T}\leq\operatorname{depth}(Q\oplusR)\mathrm{ by blast
    have \existsS. uhnf S\wedge valid S ^P\oplusT \equiv
    proof -
    from PQhnf have uhnf P by(simp add: uhnf-def)
    moreover from validPQ have valid P by simp
    moreover have }\T.\llbracketuhnf P; uhnf T; valid P; valid T\rrbracket \Longrightarrow\existsS. uhnf S ^
valid S\wedgeP\oplusT 餀S\wedge depth S\leqdepth(P\oplusT) by fact
    ultimately show ?thesis using Thnf validT by blast
    qed
    then obtain S where Shnf:uhnf S and PTeqS: P\oplusT #
S
                    and Sdepth: depth S \leq depth (P\oplusT) by blast
    have }(P\oplusQ)\oplusR\mp@subsup{\equiv}{e}{}
    proof -
    have (P\oplusQ)\oplusR \equiv
    moreover from QReqT have }P\oplus(Q\oplusR)\equiv\mp@subsup{}{e}{}P\oplus
        by(blast intro: Refl SumPres')
            ultimately show ?thesis using PTeqS by(blast intro: Trans)
    qed
    moreover from Tdepth Sdepth have depth S \leq depth ((P\oplusQ)\oplusR) by auto
    ultimately show ?case using Shnf validS by blast
next
    case(Par P Q R)
    have uhnf (P|Q) by fact
    hence False by(simp add:uhnf-def)
```

```
    thus ?case by simp
next
    case(Res x P Q)
    show ?case
    proof(case-tac Q = 0)
        assume Q = 0
        moreover have uhnf (<\nux>P) by fact
        moreover have valid ( <\nux>P) by fact
    moreover have <\nux>P\oplus\mathbf{0}\mp@subsup{\equiv}{e}{}<\nux>P}\mathrm{ by(rule SumZero)
    moreover have depth (<\nux>P)\leq\operatorname{depth}((<\nux>P)\oplus0) by simp
    ultimately show ?case by blast
    next
    assume QineqNil: Q\not=\mathbf{0}
    have Qhnf: uhnf Q by fact
    have validP: valid (<\nux>P) and validQ: valid Q by fact+
    show ?case
    proof(case-tac \exists\mp@subsup{Q}{}{\prime}\in\mathrm{ summands Q. Q' }\mp@subsup{\equiv}{e}{}<\nux>P)
        assume }\exists\mp@subsup{Q}{}{\prime}\in\mathrm{ summands Q. Q Q }\mp@subsup{\equiv}{e}{}<\nux>
        then obtain }\mp@subsup{Q}{}{\prime}\mathrm{ where }\mp@subsup{Q}{}{\prime}\in\mathrm{ summands Q and }\mp@subsup{Q}{}{\prime}\equiv\mp@subsup{\equiv}{e}{}<\nux>P by blas
        hence }Q\oplus<\nux>P\mp@subsup{\equiv}{e}{}Q\mathrm{ by(rule summandsIdemp)
        moreover have depth Q
        ultimately show ?case using Qhnf validQ by(force intro:SumSym Trans)
    next
        assume }\neg(\exists\mp@subsup{Q}{}{\prime}\in\mathrm{ summands Q. Q' }\mp@subsup{Q}{e}{}<<\nux>P
        hence }\forall\mp@subsup{Q}{}{\prime}\in\mathrm{ summands Q.}\neg(\mp@subsup{Q}{}{\prime}\mp@subsup{\equiv}{e}{}<\nux>P)\mathrm{ by simp
        moreover have uhnf (<\nux>P) by fact
        ultimately have uhnf ( <\nux>P \oplusQ) using Qhnf QineqNil
            by(force dest: Sym simp add: uhnf-def)
        moreover from validP validQ have valid ( <\nux>>P\oplusQ) by simp
        moreover have }(<\nux>P)\oplusQ\mp@subsup{\equiv}{e}{}(<\nux>P)\oplusQ\mathrm{ by(rule Refl)
        moreover have depth ((<\nux>P)\oplusQ)\leq\operatorname{depth}((<\nux>P)\oplusQ) by simp
        ultimately show ?case by blast
    qed
    qed
next
    case(Bang P Q)
    have uhnf (!P) by fact
    hence False by(simp add: uhnf-def)
    thus ?case by simp
qed
lemma uhnfRes:
    fixes x :: name
    and }P::p
    assumes Phnf:uhnf P
    and validP: valid P
```



```
using assms
proof (nominal-induct \(P\) avoiding: x rule: pi.strong-inducts)
    case (PiNil x)
    have uhnf \(\mathbf{0}\) by (simp add: uhnf-def)
    moreover have valid 0 by simp
    moreover have \(<\nu x>\mathbf{0} \equiv_{e} \mathbf{0}\) by (rule ResNil)
    moreover have depth \(\mathbf{0} \leq\) depth \((<\nu x>\mathbf{0})\) by simp
    ultimately show ?case by blast
next
    case(Output a b P)
    have \(\operatorname{valid}(a\{b\} . P)\) by fact
    hence validP: valid \(P\) by simp
    show ? case
    proof (case-tac \(x=a\) )
        assume \(x=a\)
        moreover have uhnf \(\mathbf{0}\) by (simp add: uhnf-def)
        moreover have valid 0 by simp
        moreover have \(\mathbf{0} \equiv_{e}<\nu x>x\{b\} . P\) by (blast intro: ResOutput' Sym)
        moreover have depth \(\mathbf{0} \leq\) depth \((<\nu x>x\{b\} . P)\) by simp
        ultimately show ? case by (blast intro: Sym)
    next
    assume xineqa: \(x \neq a\)
    show ?case
    proof (case-tac \(x=b\) )
        assume \(x=b\)
    moreover from xineqa have \(u h n f(<\nu x>a\{x\} . P)\) by (force simp add: uhnf-def)
        moreover from validP have valid \((<\nu x>a\{x\} . P)\) by simp
        moreover have \(<\nu x>a\{x\} . P \equiv_{e}<\nu x>a\{x\} . P\) by(rule Refl)
        moreover have depth \((<\nu x>a\{x\} . P) \leq \operatorname{depth}(<\nu x>a\{x\} . P)\) by simp
        ultimately show ?case by blast
    next
        assume xineqb: \(x \neq b\)
        have \(\operatorname{uhnf}(a\{b\} .(<\nu x>P))\) by (simp add: uhnf-def)
        moreover from validP have \(\operatorname{valid}(a\{b\} .(<\nu x>P))\) by \(\operatorname{simp}\)
        moreover from xineqa xineqb have \(a\{b\} .(<\nu x>P) \equiv_{e}<\nu x>a\{b\} . P\) by (blast
intro: ResOutput Sym)
            moreover have \(\operatorname{depth}(a\{b\} .(<\nu x>P)) \leq \operatorname{depth}(<\nu x>a\{b\} . P)\) by simp
            ultimately show? ?ase by (blast intro: Sym)
    qed
    qed
next
    case (Tau P)
    have \(\operatorname{valid}(\tau .(P))\) by fact
    hence validP: valid \(P\) by simp
    have \(u h n f(\tau .(<\nu x>P))\) by (simp add: uhnf-def)
    moreover from validP have valid \((\tau .(<\nu x>P))\) by simp
    moreover have \(\tau .(<\nu x>P) \equiv_{e}<\nu x>\tau .(P)\) by(blast intro: ResTau Sym)
    moreover have depth \((\tau .(<\nu x>P)) \leq \operatorname{depth}(<\nu x>\tau .(P))\) by simp
```

```
    ultimately show ?case by(blast intro: Sym)
next
    case(Input a y P x)
    have valid( }a<y>.P)\mathrm{ by fact
    hence validP: valid P by simp
    have }y\sharpx\mathrm{ by fact hence yineqx: }y\not=x\mathrm{ by simp
    show ?case
    proof(case-tac x=a)
        assume }x=
        moreover have uhnf 0 by(simp add: uhnf-def)
    moreover have valid 0 by simp
    moreover have 0 \equiv}\mp@subsup{e}{e}{<\nux>x<y>.P by(blast intro: ResInput' Sym)
    moreover have depth 0 \leq depth( }\langle\nux>x<y>.P) by sim
    ultimately show ?case by(blast intro: Sym)
    next
    assume xineqa: x\not=a
    have uhnf(a<y>.(<\nux>P)) by(simp add: uhnf-def)
    moreover from validP have valid( }a<y>.(<\nux>P))\mathrm{ by simp
    moreover from xineqa yineqx have }a<y>.(<\nux>P)\mp@subsup{\equiv}{e}{}<\nux>a<y>.P by (blas
intro: ResInput Sym)
    moreover have depth(a<y>.(<\nux>P))\leqdepth(<\nux>a<y>.P) by simp
    ultimately show ?case by(blast intro: Sym)
    qed
next
    case(Match a b P x)
    have uhnf([a\frownb]P) by fact
    hence False by(simp add: uhnf-def)
    thus ?case by simp
next
    case(Mismatch a b P x)
    have uhnf([a\not=b]P) by fact
    hence False by(simp add: uhnf-def)
    thus ?case by simp
next
    case(Sum P Q x)
    have valid(P\oplusQ) by fact
    hence validP: valid P and validQ: valid Q by simp+
    have uhnf(P\oplusQ) by fact
    hence Phnf:uhnf P and Qhnf: uhnf Q by(auto simp add: uhnf-def)
```



```
    proof -
    have \llbracketuhnf P; valid P\rrbracket\Longrightarrow \exists P'.uhnf P'^ valid P'^ <\nux>P \equiv}\mp@subsup{P}{e}{}\mp@subsup{P}{}{\prime}\wedge(\mathrm{ depth
P')}\leq(\mathrm{ depth }(<\nux>P))\mathrm{ by fact
    with validP Phnf show ?thesis by(blast intro: Sym)
    qed
    then obtain P' where P'hnf:uhnf P' and P'eqP: 的' }\mp@subsup{\equiv}{e}{}<\nux>P\mathrm{ and validP':
valid P'
        and }\mp@subsup{P}{}{\prime}\mathrm{ depth: (depth P')}\leq(\mathrm{ depth (< < x>P)) by blast
```

have $\exists Q^{\prime}$. uhnf $Q^{\prime} \wedge$ valid $Q^{\prime} \wedge Q^{\prime} \equiv_{e}<\nu x>Q \wedge\left(\right.$ depth $\left.Q^{\prime}\right) \leq(\operatorname{depth}(<\nu x>Q))$ proof -
have $\llbracket u h n f Q ;$ valid $Q \rrbracket \Longrightarrow \exists Q^{\prime}$. uhnf $Q^{\prime} \wedge$ valid $Q^{\prime} \wedge<\nu x>Q \equiv_{e} Q^{\prime} \wedge$ (depth $\left.Q^{\prime}\right) \leq($ depth $(<\nu x>Q))$ by fact
with validQ Qhnf show ?thesis by(blast intro: Sym)
qed
then obtain $Q^{\prime}$ where $Q^{\prime} h n f:$ uhnf $Q^{\prime}$ and $Q^{\prime} e q Q: Q^{\prime} \equiv_{e}<\nu x>Q$ and validQ': valid $Q^{\prime}$

$$
\text { and } Q^{\prime} \text { depth: }\left(\text { depth } Q^{\prime}\right) \leq(\operatorname{depth}(<\nu x>Q)) \text { by blast }
$$

from $P^{\prime} h n f Q^{\prime} h n f$ validP $P^{\prime}$ valid $Q^{\prime}$ obtain $R$ where $R h n f$ : uhnf $R$ and validR: valid $R$
and $P^{\prime} Q^{\prime} e q R: P^{\prime} \oplus Q^{\prime} \equiv_{e} R$
and Rdepth: depth $R \leq \operatorname{depth}\left(P^{\prime} \oplus Q^{\prime}\right)$
apply (drule-tac uhnfSum) apply assumption+ by blast
from $P^{\prime} e q P Q^{\prime} e q Q P^{\prime} Q^{\prime} e q R$ have $<\nu x>(P \oplus Q) \equiv_{e} R$ by (blast intro: Sym SumPres' SumRes Trans)
moreover from Rdepth $P^{\prime}$ depth $Q^{\prime}$ depth have depth $R \leq \operatorname{depth}(<\nu x>(P \oplus Q))$ by auto ultimately show ?case using validR Rhnf by (blast intro: Sym)

## next

case $(\operatorname{Par} P Q)$
have $\operatorname{uhnf}(P \| Q)$ by fact
hence False by (simp add: uhnf-def)
thus? case by simp
next
case $($ Res y $P$ )
have valid $(<\nu y>P)$ by fact hence validP: valid $P$ by simp
have uhnf $(<\nu y>P)$ by fact
then obtain $a P^{\prime}$ where aineqy: $a \neq y$ and $P e q P^{\prime}: P=a\{y\} . P^{\prime}$
by (force simp add: uhnf-def)
show ?case
proof (case-tac $x=y$ )
assume $x=y$
moreover from aineqy have $u h n f\left(<\nu y>a\{y\} . P^{\prime}\right)$ by (force simp add: uhnf-def)
moreover from validP PeqP' have valid $\left(<\nu y>a\{y\} . P^{\prime}\right)$ by simp
moreover have $<\nu y><\nu y>a\{y\} \cdot P^{\prime} \equiv_{e}<\nu y>a\{y\} . P^{\prime}$
proof -
have $y \sharp<\nu y>a\{y\} . P^{\prime}$ by (simp add: name-fresh-abs)
hence $<\nu y><\nu y>a\{y\} . P^{\prime} \equiv_{e}<\nu y>a\{y\} . P^{\prime}$ by (rule ResFresh)
thus ?thesis by(blast intro: Trans)
qed
moreover have $\operatorname{depth}\left(<\nu y>a\{y\} . P^{\prime}\right) \leq \operatorname{depth}\left(<\nu y><\nu y>a\{y\} . P^{\prime}\right)$ by simp ultimately show ? case using PeqP ${ }^{\prime}$ by blast
next

```
    assume xineqy: }x\not=
    show ?case
    proof(case-tac x=a)
        assume x=a
        moreover have uhnf 0 by(simp add: uhnf-def)
        moreover have valid 0 by simp
        moreover have <\nua><\nuy>a{y}.P'看0
        proof -
            have <\nua><\nuy>a{y}.\mp@subsup{P}{}{\prime}}\mp@subsup{\equiv}{e}{}<\nuy><\nu\nua>a{y}.\mp@subsup{P}{}{\prime}\mathbf{by}(rule ResComm
            moreover have <\nuy><\nua>a{y}.\mp@subsup{P}{}{\prime}\equiv\mp@subsup{\equiv}{e}{}0
                by(blast intro: ResOutput' ResNil ResPres Trans)
            ultimately show ?thesis by(blast intro: Trans)
        qed
        moreover have depth \mathbf{0}\leq\mathrm{ depth( }<\nua><\nuy>a{y}.\mp@subsup{P}{}{\prime})\mathrm{ by simp}
        ultimately show ?case using PeqP' by blast
    next
        assume xineqa: }x\not=
        from aineqy have uhnf(<\nuy>a{y}.(<\nux>>P')) by(force simp add: uhnf-def)
        moreover from validP PeqP' have valid( <\nuy>a{y}.(<\nux>\mp@subsup{P}{}{\prime})) by simp
        moreover have <\nux><\nuy>a{y}.\mp@subsup{P}{}{\prime}}\mp@subsup{\equiv}{e}{<\nu\nuy>a{y}.(<\nux>\mp@subsup{P}{}{\prime})
        proof -
            have <\nux><<\nuy>a{y}.\mp@subsup{P}{}{\prime}}\mp@subsup{\equiv}{e}{}<\nuy><\nux>a{y}.\mp@subsup{P}{}{\prime}\mathbf{by}(rule ResComm
        moreover from xineqa xineqy have <\nuy><\nux>a{y}.\mp@subsup{P}{}{\prime}}\mp@subsup{\equiv}{e}{<}<\nuy>a{y}.(<\nux>>P'
            by(blast intro: ResOutput ResPres Trans)
            ultimately show ?thesis by(blast intro: Trans)
        qed
        moreover have depth(<\nuy>a{y}.(<\nux>\mp@subsup{P}{}{\prime}))\leq\operatorname{depth}(<\nux><\nuy>a{y}.\mp@subsup{P}{}{\prime})
            by simp
            ultimately show ?case using PeqP' by blast
    qed
    qed
next
    case(Bang P x)
    have valid(!P) by fact
    hence False by simp
    thus?case by simp
qed
lemma expandHnf:
    fixes P :: pi
    and }S:: pi se
    assumes }(P,S)\in\mathrm{ sumComposeSet
    and}\quad\forallP\inS.uhnf P^valid 
```



```
using assms
proof(induct rule: sumComposeSet.induct)
    case empty
```

```
    have uhnf 0 by(simp add: uhnf-def)
    moreover have valid 0 by simp
    moreover have 0 }\mp@subsup{\equiv}{e}{}0\mathrm{ by(rule Refl)
    moreover have depth 0}\leq\mathrm{ depth 0 by simp
    ultimately show ?case by blast
next
    case(insert Q SP)
    have Shnf: }\forallP\inS\mathrm{ . uhnf }P\wedge\mathrm{ valid P by fact
    hence }\forallP\in(S-{(Q)}). uhnf P ^ valid P by simp
    moreover have }\forallP\in(S-{(Q)}). uhnf P\wedge valid P\Longrightarrow\exists\mp@subsup{P}{}{\prime}.\mathrm{ uhnf P}\mp@subsup{P}{}{\prime}\wedge vali
P
    ultimately obtain }\mp@subsup{P}{}{\prime}\mathrm{ where P'hnf: uhnf P' and validP': valid P'
                and PeqP': P \equiv
    by blast
    have }Q\inS\mathrm{ by fact
    with Shnf have uhnf Q and valid Q by simp+
    with P'hnf validP' obtain R where Rhnf: uhnf R and validR: valid R
                            and }\mp@subsup{P}{}{\prime}QeqR: P'\oplus Q \equiv>e R and \mp@subsup{P}{}{\prime}QRdepth: depth R\leq
depth ( }\mp@subsup{P}{}{\prime}\oplusQ
    by(auto dest: uhnfSum)
    from PeqP' P'QeqR have P}\oplusQ\equiv\mp@subsup{\equiv}{e}{}R\mathrm{ by(blast intro: SumPres Trans)
    moreover from P\mp@subsup{P}{}{\prime}}\mathrm{ depth P'QRdepth have depth R}\leq\operatorname{depth}(P\oplusQ)\mathrm{ by simp
    ultimately show ?case using Rhnf validR by blast
qed
lemma hnfSummandsRemove:
    fixes }P:: p
    and }Q::p
    assumes P\in summands Q
    and uhnf Q
```



```
Q) - {P}
using assms
by(auto intro: Refl simp add: uhnf-def)
lemma pullSummand:
    fixes }P\mathrm{ :: pi
    and }Q :: p
    assumes PsummQ: P\in summands Q
    and Qhnf: uhnf Q
    shows }\exists\mp@subsup{Q}{}{\prime}.P\oplus\mp@subsup{Q}{}{\prime}\equiv\mp@subsup{\equiv}{e}{}Q\wedge(\mathrm{ summands }\mp@subsup{Q}{}{\prime})=((\mathrm{ summands Q) - {x. ヨ P'. x
```



```
proof -
```

have SumGoal: $\bigwedge P Q R . \llbracket P \in$ summands $Q ; \operatorname{uhnf}(Q \oplus R)$;

$$
\bigwedge P \llbracket P \in \text { summands } Q \rrbracket \Longrightarrow \exists Q^{\prime} . P \oplus Q^{\prime} \equiv_{e} Q \wedge
$$

(summands $\left.Q^{\prime}\right)=\left((\right.$ summands $Q)-\left\{P^{\prime} \mid P^{\prime} . P^{\prime} \in\right.$ summands $\left.\left.Q \wedge P^{\prime} \equiv{ }_{e} P\right\}\right) \wedge$ uhnf $Q^{\prime} ;$
$\wedge P . \llbracket P \in$ summands $R \rrbracket \Longrightarrow \exists R^{\prime} . P \oplus R^{\prime} \equiv_{e} R \wedge$
(summands $\left.R^{\prime}\right)=\left((\right.$ summands $R)-\left\{P^{\prime} \mid P^{\prime} . P^{\prime} \in\right.$
summands $\left.\left.R \wedge P^{\prime} \equiv_{e} P\right\}\right) \wedge$ uhnf $R \rrbracket$
$\Longrightarrow \exists Q^{\prime} . P \oplus Q^{\prime} \equiv_{e} Q \oplus R \wedge$
summands $Q^{\prime}=$ summands (pi.Sum $Q R$ ) $-\left\{P^{\prime} \mid P^{\prime} . P^{\prime} \in\right.$ summands $\left.(Q \oplus R) \wedge P^{\prime} \equiv_{e} P\right\} \wedge u h n f Q^{\prime}$
proof -
fix $P Q R$
assume IHR: $\bigwedge P . P \in$ summands $R \Longrightarrow \exists R^{\prime} . P \oplus R^{\prime} \equiv_{e} R \wedge$
(summands $\left.R^{\prime}\right)=\left((\right.$ summands $R)-\left\{P^{\prime} \mid P^{\prime} . P^{\prime}\right.$
$\in$ summands $\left.\left.R \wedge P^{\prime} \equiv_{e} P\right\}\right) \wedge$ uhnf $R^{\prime}$
assume PsummQ: $P \in$ summands $Q$
moreover assume $\wedge P . P \in$ summands $Q \Longrightarrow \exists Q^{\prime} . P \oplus Q^{\prime} \equiv_{e} Q \wedge$
(summands $\left.Q^{\prime}\right)=\left((\right.$ summands $Q)-\left\{P^{\prime} \mid P^{\prime} . P^{\prime} \in\right.$ summands $\left.\left.Q \wedge P^{\prime} \equiv_{e} P\right\}\right) \wedge u h n f Q^{\prime}$
ultimately obtain $Q^{\prime}$ where $P Q^{\prime} e q Q: P \oplus Q^{\prime} \equiv_{e} Q$
and $Q^{\prime}$ summ $Q$ : (summands $\left.Q^{\prime}\right)=\left((\right.$ summands $Q)-\left\{P^{\prime}\right.$
$\mid P^{\prime} . P^{\prime} \in$ summands $\left.\left.Q \wedge P^{\prime} \equiv_{e} P\right\}\right)$
and $Q^{\prime} h n f: u h n f Q^{\prime}$
by blast
assume $Q R h n f: u h n f(Q \oplus R)$
show $\exists Q^{\prime} . P \oplus Q^{\prime} \equiv_{e} Q \oplus R \wedge$
summands $Q^{\prime}=$ summands (pi.Sum $\left.Q R\right)-\left\{P^{\prime} \mid P^{\prime} . P^{\prime} \in\right.$ summands
$\left.(Q \oplus R) \wedge P^{\prime} \equiv_{e} P\right\} \wedge$ uhnf $Q^{\prime}$
proof $\left(\right.$ cases $\exists P^{\prime} \in$ summands $R . P^{\prime} \equiv_{e} P$ )
assume $\exists P^{\prime} \in$ summands $R . P^{\prime} \equiv_{e} P$
then obtain $P^{\prime}$ where $P^{\prime}$ summR: $P^{\prime} \in$ summands $R$ and $P^{\prime} e q P: P^{\prime} \equiv{ }_{e} P$ by blast
with $I H R$ obtain $R^{\prime}$ where $P R^{\prime} e q R: P^{\prime} \oplus R^{\prime} \equiv_{e} R$
and $R^{\prime}$ summR: (summands $\left.R^{\prime}\right)=\left((\right.$ summands $R)-\left\{P^{\prime \prime} \mid P^{\prime \prime} . P^{\prime \prime} \in\right.$ summands $\left.R \wedge P^{\prime \prime} \equiv{ }_{e} P^{\prime}\right\}$ ) and $R^{\prime} h n f$ : uhnf $R^{\prime}$
by blast
have L1: $P \oplus\left(Q^{\prime} \oplus R^{\prime}\right) \equiv_{e} Q \oplus R$
proof -
from $P^{\prime} e q P$ have $P \oplus\left(Q^{\prime} \oplus R^{\prime}\right) \equiv_{e}\left(P \oplus P^{\prime}\right) \oplus\left(Q^{\prime} \oplus R^{\prime}\right)$
by (blast intro: SumIdemp ${ }^{\prime}$ SumPres Sym)
moreover have $\left(P \oplus P^{\prime}\right) \oplus\left(Q^{\prime} \oplus R^{\prime}\right) \equiv_{e} P \oplus\left(P^{\prime} \oplus\left(Q^{\prime} \oplus R^{\prime}\right)\right)$ by $($ rule SumAssoc)
moreover have $P \oplus\left(P^{\prime} \oplus\left(Q^{\prime} \oplus R^{\prime}\right)\right) \equiv_{e} P \oplus\left(P^{\prime} \oplus\left(R^{\prime} \oplus Q^{\prime}\right)\right)$
by (blast intro: Refl SumPres' SumSym)
moreover have $P \oplus\left(P^{\prime} \oplus\left(R^{\prime} \oplus Q^{\prime}\right)\right) \equiv_{e} P \oplus\left(P^{\prime} \oplus R^{\prime}\right) \oplus Q^{\prime}$

```
    by(blast intro: Refl SumPres' Sym SumAssoc)
    moreover have }P\oplus(\mp@subsup{P}{}{\prime}\oplus\mp@subsup{R}{}{\prime})\oplus\mp@subsup{Q}{}{\prime}\mp@subsup{\equiv}{e}{}(P\oplus\mp@subsup{Q}{}{\prime})\oplus(\mp@subsup{P}{}{\prime}\oplus\mp@subsup{R}{}{\prime}
    proof -
    have }P\oplus(\mp@subsup{P}{}{\prime}\oplus\mp@subsup{R}{}{\prime})\oplus\mp@subsup{Q}{}{\prime}\mp@subsup{\equiv}{e}{}P\oplus\mp@subsup{Q}{}{\prime}\oplus(\mp@subsup{P}{}{\prime}\oplus\mp@subsup{R}{}{\prime}
        by(blast intro: Refl SumPres' SumSym)
    thus ?thesis by(blast intro: Sym SumAssoc Trans)
    qed
    moreover from PQ 'eqQ PR'eqR have ( }P\oplus\mp@subsup{Q}{}{\prime
by(rule SumPres')
    ultimately show ?thesis by(blast intro!: Trans)
    qed
    show ?thesis
    proof(cases }\mp@subsup{Q}{}{\prime}=\mathbf{0}
        assume Q'eqNil: Q' = 0
        have}P\oplus\mp@subsup{R}{}{\prime}\equiv\mp@subsup{\equiv}{e}{}Q\oplus
    proof -
            have P\oplus R' \equiv
SumPres' Sym)
            moreover have P\oplus( R'\oplus\mathbf{0})\equiv\mp@subsup{\equiv}{e}{}P\oplus(\mathbf{0}\oplus\mp@subsup{R}{}{\prime})
                    by(blast intro:SumSym Trans SumPres' Refl)
            ultimately show ?thesis using L1 Q'eqNil by(blast intro: Trans)
        qed
        moreover from R'summR Q'summQ P'eqP Q'eqNil have summands ( }\mp@subsup{R}{}{\prime}\mathrm{ )
= (summands (Q\oplusR)-{\mp@subsup{P}{}{\prime}|\mp@subsup{P}{}{\prime}.\mp@subsup{P}{}{\prime}\in\operatorname{summands}(Q\oplusR)\wedge P' \equiv
            by(auto intro:Sym Trans)
            ultimately show ?thesis using R'hnf by blast
    next
            assume Q'ineqNil: Q'}=\mathbf{0
            show ?thesis
            proof(case-tac R'=0)
            assume R'eqNil: R'=0
            have}P\oplus\mp@subsup{Q}{}{\prime}\mp@subsup{\equiv}{e}{}Q\oplus
            proof -
                have P}\oplus\mp@subsup{Q}{}{\prime}\equiv\mp@subsup{\equiv}{e}{}P\oplus(\mp@subsup{Q}{}{\prime}\oplus\mathbf{0})\mathbf{by}(blast intro: SumZero Refl Tran
SumPres' Sym)
            with L1 R'eqNil show ?thesis by(blast intro: Trans)
            qed
    moreover from R'summR Q'summQ P'eqP R'eqNil have summands ( }\mp@subsup{Q}{}{\prime}\mathrm{ )
= (summands (Q\oplusR)-{\mp@subsup{P}{}{\prime}|\mp@subsup{P}{}{\prime}.\mp@subsup{P}{}{\prime}\in\operatorname{summands}(Q\oplusR)\wedge P' \equiv
            by(auto intro: Sym Trans)
        ultimately show ?thesis using Q'hnf by blast
    next
        assume R'ineqNil: R'\not=\mathbf{0}
    from R'summR Q'summQ P'eqP have summands ( }\mp@subsup{Q}{}{\prime}\oplus\mp@subsup{R}{}{\prime})=(\mathrm{ summands
(Q\oplusR)-{\mp@subsup{P}{}{\prime}|\mp@subsup{P}{}{\prime}.\mp@subsup{P}{}{\prime}\in\operatorname{summands}(Q\oplusR)\wedge P
            by(auto intro: Sym Trans)
                moreover from QRhnf Q'hnf R'hnf R'summR Q'summQ Q'ineqNil
```

```
R'ineqNil have uhnf( ( }\mp@subsup{Q}{}{\prime}\oplus\mp@subsup{R}{}{\prime}
                    by(auto simp add:uhnf-def)
                    ultimately show ?thesis using L1 by blast
                qed
            qed
    next
        assume }\neg(\exists\mp@subsup{P}{}{\prime}\in\mathrm{ summands R. P}\mp@subsup{P}{}{\prime}\equiv\mp@subsup{\equiv}{e}{}P
        hence Case: }\forall\mp@subsup{P}{}{\prime}\in\mathrm{ summands R.}\neg(\mp@subsup{P}{}{\prime}\mp@subsup{\equiv}{e}{}P)\mathrm{ by simp
        show ?thesis
        proof(case-tac Q' = 0)
            assume Q'eqNil: Q' = 0
            have }P\oplusR\mp@subsup{\equiv}{e}{}Q\oplus
            proof -
            have }P\oplusR\mp@subsup{\equiv}{e}{}(P\oplus\mathbf{0})\oplusR\mathrm{ by(blast intro: SumZero Sym Trans SumPres)
            moreover from PQ'eqQ have P}\oplus(\mp@subsup{Q}{}{\prime}\oplusR)\mp@subsup{\equiv}{e}{}Q\oplus
                    by(blast intro:SumAssoc Trans Sym SumPres)
            ultimately show ?thesis using Q'eqNil by(blast intro: SumAssoc Trans)
            qed
            moreover from Q'summQ Q'eqNil Case have summands (R)=(summands
(Q\oplusR)-{\mp@subsup{P}{}{\prime}|\mp@subsup{P}{}{\prime}.\mp@subsup{P}{}{\prime}\in\operatorname{summands}(Q\oplusR)\wedge P
            by auto
            moreover from QRhnf have uhnf R by(simp add:uhnf-def)
            ultimately show ?thesis by blast
            next
            assume Q'ineqNil: }\mp@subsup{Q}{}{\prime}\not=\mathbf{0
            from P\mp@subsup{Q}{}{\prime}eqQ have P}\oplus(\mp@subsup{Q}{}{\prime}\oplusR)\mp@subsup{\equiv}{e}{}Q\oplus
                    by(blast intro:SumAssoc Trans Sym SumPres)
            moreover from Q'summQ Case have summands ( }\mp@subsup{Q}{}{\prime}\oplusR)=(\mathrm{ summands
```



```
            by auto
            moreover from QRhnf Q'hnf Q'summQ Q'ineqNil have uhnf ( }\mp@subsup{Q}{}{\prime}\oplusR
            by(auto simp add: uhnf-def)
            ultimately show ?thesis by blast
        qed
    qed
qed
    from assms show ?thesis
    proof(nominal-induct Q arbitrary: P rule: pi.strong-inducts)
        case PiNil
        have P}\in\mathrm{ summands 0 by fact
        hence False by auto
        thus ?case by simp
    next
```

```
    case(Output a b Q)
    have }P\in\mathrm{ summands (a{b}.Q) by fact
    hence PeqQ:P=a{b}.Q by simp
    have }P\oplus\mathbf{0}\mp@subsup{\equiv}{e}{}a{b}.
    proof -
        have P\oplus00 \equiv
        with PeqQ show ?thesis by simp
    qed
    moreover have (summands 0) = (summands (a{b}.Q)) - {P'| | P'. P' 
summands (a{b}.Q)\wedge P' }\mp@subsup{\equiv}{e}{}P
    proof -
        have }a{b}.Q\mp@subsup{\equiv}{e}{}a{b}.Q by(rule Refl
        with PeqQ show ?thesis by simp
    qed
    moreover have uhnf 0 by(simp add: uhnf-def)
    ultimately show ?case by blast
    next
    case(Tau Q)
    have }P\in\mathrm{ summands ( }\tau.(Q))\mathrm{ by fact
    hence PeqQ: P=\tau.(Q) by simp
    have }P\oplus\mathbf{0}\mp@subsup{\equiv}{e}{e}\tau.(Q
    proof -
        have }P\oplus\mathbf{0}\equiv\mp@subsup{}{e}{}P\mathrm{ by(rule SumZero)
        with PeqQ show ?thesis by simp
    qed
```



```
summands (\tau.(Q))\wedge P' }\mp@subsup{\equiv}{e}{}P
    proof -
        have}\tau.(Q) \equiv\mp@subsup{e}{e}{}\tau.(Q)\mathbf{by}(rule Refl
        with PeqQ show ?thesis by simp
    qed
    moreover have uhnf 0 by(simp add:uhnf-def)
    ultimately show ?case by blast
    next
    case(Input a x Q)
    have }P\in\mathrm{ summands ( }a<x>.Q)\mathrm{ by fact
    hence PeqQ: P=a<x>.Q by simp
    have P\oplus\mathbf{0}\equiv
    proof -
        have P\oplus\mathbf{0}\equiv\mp@subsup{}{e}{}P\mathrm{ by(rule SumZero)}
        with PeqQ show ?thesis by simp
    qed
```



```
summands (a<x>.Q)\wedge P' }\mp@subsup{\equiv}{e}{}P
    proof -
        have }a<x>.Q\equiv\mp@subsup{\equiv}{e}{}a<x>.Q by(rule Refl
        with PeqQ show ?thesis by simp
    qed
    moreover have uhnf 0 by(simp add: uhnf-def)
```

```
    ultimately show ?case by blast
    next
    case(Match a b Q)
    have }P\in\mathrm{ summands ([a乞b]Q) by fact
    hence False by simp
    thus ?case by simp
next
    case(Mismatch a b Q)
    have }P\in\mathrm{ summands ([a申b]Q) by fact
    hence False by simp
    thus ?case by simp
    next
    case(Sum Q R)
    have QRhnf:uhnf (Q\oplusR) by fact
    hence Qhnf:uhnf Q and Rhnf:uhnf R by(simp add: uhnf-def)+
```



```
                                    (summands }\mp@subsup{Q}{}{\prime})=((\mathrm{ summands }Q)-{\mp@subsup{P}{}{\prime}|\mp@subsup{P}{}{\prime}.\mp@subsup{P}{}{\prime
\in summands Q}\wedge \mp@subsup{P}{}{\prime}\mp@subsup{\equiv}{e}{}P})\wedge\mathrm{ uhnf Q'
            by fact
    with Qhnf have IHQ: \P. P\in summands Q\Longrightarrow\exists\mp@subsup{Q}{}{\prime}.P\oplus Q' 汭 Q^
                                    (summands Q')=((summands Q)-{徎|蚆. P' 
summands Q ^ P' 汭 P}) ^uhnf Q'
            by simp
    have }\P.\llbracketP\in\mathrm{ summands R; uhnf R】 ב ヨ R'. P}\oplus\mp@subsup{R}{}{\prime}\equiv\mp@subsup{\equiv}{e}{}R
```



```
\epsilon summands R}\wedge \mp@subsup{P}{}{\prime}\mp@subsup{\equiv}{e}{}P})\wedgeuhnf R'
            by fact
    with Rhnf have IHR: \P.P\in summands R\Longrightarrow\exists 质.P\oplus R' \equiv
```



```
summands R}\wedge \mp@subsup{P}{}{\prime}\equiv\mp@subsup{\equiv}{e}{}P})\wedgeuhnf R'
            by simp
    have P
    hence P
    thus ?case
    proof(rule disjE)
            assume P}\in\mathrm{ summands Q
            thus ?case using QRhnf IHQ IHR by(rule SumGoal)
            next
            assume P}\in\mathrm{ summands }
            moreover from QRhnf have uhnf (R}\oplusQ) by(auto simp add: uhnf-def
            ultimately have }\exists\mp@subsup{Q}{}{\prime}.(\mathrm{ pi.Sum P Q ) }\mp@subsup{\equiv}{e}{}(\mathrm{ pi.Sum R Q)^
                summands }\mp@subsup{Q}{}{\prime}=\mathrm{ summands (pi.Sum R Q) - {P'|P'. 徎' }\in\mathrm{ summands
(pi.Sum R Q ) ^ P' 汭 P} ^uhnf Q' using IHR IHQ
            by(rule SumGoal)
            thus ?case
            by(force intro: SumSym Trans)
    qed
    next
    case(Par Q R P)
```

```
        have \(P \in\) summands \((Q \| R)\) by fact
        hence False by simp
        thus? case by simp
    next
    case \((\) Res \(x \quad Q \quad P)\)
    have \(P \in\) summands \((<\nu x>Q)\) by fact
    hence \(P e q Q: P=\langle\nu x\rangle Q\) by (simp add: if-split)
    have \(P \oplus \mathbf{0} \equiv_{e}<\nu x>Q\)
    proof -
        have \(P \oplus \mathbf{0} \equiv_{e} P\) by(rule SumZero)
        with \(\operatorname{PeqQ}\) show ?thesis by simp
    qed
    moreover have (summands \(\mathbf{0})=(\) summands \((<\nu x>Q))-\left\{P^{\prime} \mid P^{\prime} . P^{\prime} \in\right.\)
summands \(\left.(<\nu x>Q) \wedge P^{\prime} \equiv_{e} P\right\}\)
    proof -
            have \(<\nu x>Q \equiv_{e}<\nu x>Q\) by (rule Refl)
            with \(\operatorname{PeqQ}\) show ?thesis by simp
        qed
        moreover have uhnf \(\mathbf{0}\) by (simp add: uhnf-def)
        ultimately show ?case by blast
    next
    case (Bang Q P)
    have \(P \in\) summands (! \(Q\) ) by fact
    hence False by simp
    thus? case by simp
    qed
qed
lemma nSym:
    fixes \(P:\) : \(p i\)
    and \(\quad Q:: p i\)
    assumes \(\neg\left(P \equiv_{e} Q\right)\)
    shows \(\neg\left(Q \equiv{ }_{e} P\right)\)
using assms
by (blast dest: Sym)
lemma summandsZero:
    fixes \(P\) :: \(p i\)
    assumes summands \(P=\{ \}\)
    and \(\quad h n f P\)
    shows \(P=\mathbf{0}\)
using assms
by (nominal-induct \(P\) rule: pi.strong-inducts, auto intro: Refl SumIdemp SumPres \({ }^{\prime}\)
Trans)
```

```
lemma summandsZero':
    fixes P :: pi
    assumes summP: summands P}={
    and Puhnf:uhnf P
    shows }P=
proof -
    from Puhnf have hnf P by(simp add: uhnf-def)
    with summP show ?thesis by(rule summandsZero)
qed
lemma summandEquiv:
    fixes }P\mathrm{ :: pi
    and }Q::p
    assumes Phnf:uhnf P
    and Qhnf:uhnf Q
    and PinQ:}\forall\mp@subsup{P}{}{\prime}\in\mathrm{ summands }P.\exists\mp@subsup{Q}{}{\prime}\in\mathrm{ summands Q. P' }\mp@subsup{\equiv}{e}{\prime}\mp@subsup{Q}{}{\prime
```



```
    shows P}\equiv\mp@subsup{\equiv}{e}{}
proof -
    from finiteSummands assms show ?thesis
    proof(induct F==summands P arbitrary: P Q rule: finite-induct)
    case(empty P Q)
    have PEmpty: {} = summands P by fact
    moreover have }\forall\mp@subsup{Q}{}{\prime}\in\mathrm{ summands }Q.\exists\mp@subsup{P}{}{\prime}\in\mathrm{ summands }P.\mp@subsup{Q}{}{\prime}\equiv\mp@subsup{\equiv}{e}{}\mp@subsup{P}{}{\prime}\mathrm{ by fact
    ultimately have QEmpty: summands Q ={} by simp
    have }P=\mathbf{0
    proof -
            have uhnf P by fact
            with PEmpty show ?thesis by(blast intro: summandsZero')
    qed
    moreover have Q=0
    proof -
            have uhnf Q by fact
            with QEmpty show ?thesis by(blast intro: summandsZero')
        qed
        ultimately show ?case by(blast intro: Refl)
    next
        case(insert P' F P Q)
    have Phnf: uhnf P by fact
    have Qhnf: uhnf Q by fact
```

    have \(I H: \bigwedge P Q . \llbracket F=\) summands \(P\); uhnf \(P\); uhnf \(Q ; \forall P^{\prime} \in\) summands \(P\).
    $\exists Q^{\prime} \in$ summands $Q . P^{\prime} \equiv_{e} Q^{\prime} ;$
$\forall Q^{\prime} \in$ summands $Q . \exists P^{\prime} \in$ summands $P . Q^{\prime} \equiv_{e} P \rrbracket \Longrightarrow P \equiv_{e} Q$
by fact
have PeqQ: $\forall P^{\prime} \in$ summands $P . \exists Q^{\prime} \in$ summands $Q . P^{\prime} \equiv{ }_{e} Q^{\prime}$ by fact
have QeqP: $\forall Q^{\prime} \in$ summands $Q . \exists P^{\prime} \in$ summands $P . Q^{\prime} \equiv_{e} P^{\prime}$ by fact
have PSumm: insert $P^{\prime} F=$ summands $P$ by fact
hence $P^{\prime}$ SummP: $P^{\prime} \in$ summands $P$ by auto
with Phnf obtain $P^{\prime \prime}$ where $P^{\prime} P^{\prime \prime} e q P: P^{\prime} \oplus P^{\prime \prime} \equiv_{e} P$
and $P^{\prime \prime}$ Summ: summands $P^{\prime \prime}=$ summands $P-\left\{P^{\prime \prime} \mid P^{\prime \prime}\right.$.
$P^{\prime \prime} \in$ summands $\left.P \wedge P^{\prime \prime} \equiv{ }_{e} P^{\prime}\right\}$
and $P^{\prime \prime} h n f:$ uhnf $P^{\prime \prime}$
by(blast dest: pullSummand)
from PeqQ $P^{\prime}$ SummP obtain $Q^{\prime}$ where $Q^{\prime} S u m m Q: Q^{\prime} \in$ summands $Q$ and $P^{\prime} e q Q^{\prime}: P^{\prime} \equiv{ }_{e} Q^{\prime}$ by blast
from $Q^{\prime} \operatorname{Summ} Q$ Qhnf obtain $Q^{\prime \prime}$ where $Q^{\prime} Q^{\prime \prime} e q Q: Q^{\prime} \oplus Q^{\prime \prime} \equiv_{e} Q$ and $Q^{\prime \prime}$ Summ: summands $Q^{\prime \prime}=$ summands $Q-\left\{Q^{\prime \prime}\right.$ $\mid Q^{\prime \prime} . Q^{\prime \prime} \in$ summands $\left.Q \wedge Q^{\prime \prime} \equiv_{e} Q^{\prime}\right\}$ and $Q^{\prime \prime} h n f: u h n f Q^{\prime \prime}$
by(blast dest: pullSummand)
have $F e q P^{\prime \prime}: F=$ summands $P^{\prime \prime}$
proof -
have $P^{\prime} \notin F$ by fact
with $P^{\prime \prime}$ Summ PSumm hnfSummandsRemove $\left[O F P^{\prime} S u m m P\right.$ Phnf] show ?thesis by blast
qed
moreover have $\forall P^{\prime} \in$ summands $P^{\prime \prime} . \exists Q^{\prime} \in$ summands $Q^{\prime \prime} . P^{\prime} \equiv{ }_{e} Q^{\prime}$ proof (rule ballI)
fix $P^{\prime \prime \prime}$
assume $P^{\prime \prime \prime}$ Summ: $P^{\prime \prime \prime} \in$ summands $P^{\prime \prime}$
with $P^{\prime \prime}$ Summ have $P^{\prime \prime \prime} \in$ summands $P$ by simp
with PeqQ obtain $Q^{\prime \prime \prime}$ where $Q^{\prime \prime \prime}$ Summ: $Q^{\prime \prime \prime} \in$ summands $Q$ and $P^{\prime \prime \prime}$ eq $Q^{\prime \prime \prime}$ :
$P^{\prime \prime \prime} \equiv_{e} Q^{\prime \prime \prime}$ by blast
have $Q^{\prime \prime \prime} \in$ summands $Q^{\prime \prime}$
proof -
from $P^{\prime \prime \prime}$ Summ $P^{\prime \prime}$ Summ have $\neg\left(P^{\prime \prime \prime} \equiv_{e} P^{\prime}\right)$ by simp
with $P^{\prime} e q Q^{\prime} P^{\prime \prime \prime} e q Q^{\prime \prime \prime}$ have $\neg\left(Q^{\prime \prime \prime} \equiv_{e} Q^{\prime}\right)$ by (blast intro: Trans Sym)
with $Q^{\prime \prime}$ Summ $Q^{\prime \prime \prime}$ Summ show ?thesis by simp
qed
with $P^{\prime \prime \prime} e q Q^{\prime \prime \prime}$ show $\exists Q^{\prime} \in$ summands $Q^{\prime \prime} . P^{\prime \prime \prime} \equiv_{e} Q^{\prime}$ by blast qed
moreover have $\forall Q^{\prime} \in$ summands $Q^{\prime \prime} . \exists P^{\prime} \in$ summands $P^{\prime \prime} . Q^{\prime} \equiv{ }_{e} P^{\prime}$ proof (rule ballI)
fix $Q^{\prime \prime \prime}$

```
        assume Q '/'Summ: Q '/\prime }\in\mathrm{ summands Q '/
        with }\mp@subsup{Q}{}{\prime\prime}Summ have \mp@subsup{Q}{}{\prime\prime\prime}\in\mathrm{ summands Q by simp
        with QeqP obtain }\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ where }\mp@subsup{P}{}{\prime\prime\prime}Summ: P '/\prime \in summands 
        and }\mp@subsup{Q}{}{\prime\prime\prime}eq\mp@subsup{P}{}{\prime\prime\prime}:\mp@subsup{Q}{}{\prime\prime\prime}\equiv\mp@subsup{}{e}{}\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ by blast
        have }\mp@subsup{P}{}{\prime\prime\prime}\in\mathrm{ summands P}\mp@subsup{P}{}{\prime\prime
        proof -
            from Q '/\primeSumm Q ''Summ have }\neg(\mp@subsup{Q}{}{\prime\prime\prime}\equiv\mp@subsup{\equiv}{e}{}\mp@subsup{Q}{}{\prime})\mathrm{ by simp
            with }\mp@subsup{P}{}{\prime}eq\mp@subsup{Q}{}{\prime}\mp@subsup{Q}{}{\prime\prime\prime}eq\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ have }\neg(\mp@subsup{P}{}{\prime\prime\prime}\equiv\mp@subsup{\equiv}{e}{}\mp@subsup{P}{}{\prime})\quad\mathbf{by}(\mathrm{ blast intro: Trans)
            with P}\mp@subsup{P}{}{\prime\prime}Summ \mp@subsup{P}{}{\prime\prime}Summ show ?thesis by sim
        qed
        with }\mp@subsup{Q}{}{\prime\prime\prime}eq\mp@subsup{P}{}{\prime\prime\prime}\mathrm{ show }\exists\mp@subsup{P}{}{\prime}\in\mathrm{ summands }\mp@subsup{P}{}{\prime\prime}.\mp@subsup{Q}{}{\prime\prime\prime}\equiv\mp@subsup{\equiv}{e}{}\mp@subsup{P}{}{\prime}\mathrm{ by blast
        qed
        ultimately have }\mp@subsup{P}{}{\prime\prime}eq\mp@subsup{Q}{}{\prime\prime}:\mp@subsup{P}{}{\prime\prime}\equiv\mp@subsup{}{e}{}\mp@subsup{Q}{}{\prime\prime}\mathrm{ using }\mp@subsup{P}{}{\prime\prime}hnf Q'hnf by(rule-tac IH
    from P P'P '\prime eqP have }P\equiv\mp@subsup{}{e}{}\mp@subsup{P}{}{\prime}\oplus\mp@subsup{P}{}{\prime\prime}\mathrm{ by(rule Sym)
    moreover from P' }eq\mp@subsup{Q}{}{\prime}\mp@subsup{P}{}{\prime\prime}eq\mp@subsup{Q}{}{\prime\prime}\mathrm{ have }\mp@subsup{P}{}{\prime}\oplus\mp@subsup{P}{}{\prime\prime}\equiv\mp@subsup{}{e}{}\mp@subsup{Q}{}{\prime}\oplus\mp@subsup{Q}{}{\prime\prime}\mathbf{by}(rule SumPres')
    ultimately show ?case using }\mp@subsup{Q}{}{\prime}\mp@subsup{Q}{}{\prime\prime}eqQ by(blast intro: Trans
    qed
qed
lemma validSubst[simp]:
    fixes P :: pi
    and a :: name
    and }b:: nam
    and p::pi
    shows valid}(P[a::=b])=\operatorname{valid}
by(nominal-induct P avoiding: a b rule: pi.strong-inducts, auto)
lemma validOutputTransition:
    fixes P :: pi
    and a :: name
    and b :: name
    and }\mp@subsup{P}{}{\prime}::p
    assumes }P\longmapstoa[b]\prec\mp@subsup{P}{}{\prime
    and valid P
    shows valid P'
using assms
by(nominal-induct rule: outputInduct, auto)
lemma validInputTransition:
    fixes P :: pi
    and a :: name
    and }x\mathrm{ :: name
    and }\mp@subsup{P}{}{\prime}::p
```

```
    assumes PTrans: P\longmapstoa<x> \prec- P'
    and validP: valid P
    shows valid P'
proof -
    have Goal: \P a x P'.\llbracketP\longmapstoa<x>\prec P'; x\sharpP; valid P\rrbracket\Longrightarrow valid P'
    proof -
        fix Pax P P
```



```
        thus valid P'
            by(nominal-induct rule: inputInduct, auto)
    qed
    obtain y::name where yFreshP: y\sharpP and yFreshP': y\sharp 的
    by(rule-tac name-exists-fresh[of ( }P,\mp@subsup{P}{}{\prime})]\mathrm{ ], auto simp add: fresh-prod)
    from yFresh\mp@subsup{P}{}{\prime}}\mathrm{ PTrans have }P\longmapstoa<y>\prec[(x,y)] \cdot P' by(simp add: al-
phaBoundResidual)
    hence valid ([(x,y)] \cdot P') using yFreshP validP by(rule Goal)
    thus valid P' by simp
qed
lemma validBoundOutputTransition:
    fixes P :: pi
    and a :: name
    and }x\mathrm{ :: name
    and }\mp@subsup{P}{}{\prime}:: p
    assumes PTrans: }P\longmapstoa<\nux>\prec\mp@subsup{P}{}{\prime
    and validP: valid P
    shows valid P'
proof -
```



```
    proof -
        fix Pax P
        assume P}\longmapstoa<\nux>\prec\prec\mp@subsup{P}{}{\prime}\mathrm{ and }x\sharpP\mathrm{ and valid P
        thus valid P'
            apply(nominal-induct rule: boundOutputInduct, auto)
            proof -
                fix Pax P '
                assume P}\longmapsto(a::name)[x]\prec\mp@subsup{P}{}{\prime}\mathrm{ and valid }
                thus valid P'
                    by(nominal-induct rule: outputInduct, auto)
            qed
    qed
    obtain y::name where yFreshP: }y\sharpP\mathrm{ and yFreshP': y }#\mp@subsup{P}{}{\prime
        by(rule-tac name-exists-fresh[of ( }P,\mp@subsup{P}{}{\prime})]\mathrm{ ], auto simp add: fresh-prod)
    from yFreshP' PTrans have }P\longmapstoa<\nuy>\prec \prec[(x,y)] \cdot P' by (simp add: al-
phaBoundResidual)
```

```
    hence valid \(\left([(x, y)] \cdot P^{\prime}\right)\) using \(y\) FreshP validP by (rule Goal)
    thus valid \(P^{\prime}\) by simp
qed
lemma validTauTransition:
    fixes \(P\) :: \(p i\)
    and \(\quad P^{\prime}:: p i\)
    assumes PTrans: \(P \longmapsto \tau \prec P^{\prime}\)
    and validP: valid \(P\)
    shows valid \(P^{\prime}\)
using assms
by (nominal-induct rule: tauInduct, auto dest: validOutputTransition validInput-
Transition validBoundOutputTransition)
lemmas validTransition \(=\) validInputTransition validOutputTransition validTau-
Transition validBoundOutputTransition
lemma validSummand:
    fixes \(P\) :: \(p i\)
    and \(\quad P^{\prime}:: p i\)
    and \(a\) :: name
    and \(b\) :: name
    and \(x\) :: name
    assumes valid \(P\)
    and \(\quad h n f P\)
    shows \(\tau .\left(P^{\prime}\right) \in\) summands \(P \Longrightarrow\) valid \(P^{\prime}\)
    and \(a\{b\} . P^{\prime} \in\) summands \(P \Longrightarrow\) valid \(P^{\prime}\)
    and \(a<x>. P^{\prime} \in\) summands \(P \Longrightarrow\) valid \(P^{\prime}\)
    and \(\llbracket a \neq x ;<\nu x>a\{x\} . P^{\prime} \in\) summands \(P \rrbracket \Longrightarrow\) valid \(P^{\prime}\)
proof -
    assume \(\tau .\left(P^{\prime}\right) \in\) summands \(P\)
    with assms show valid \(P^{\prime}\) by (force intro: validTauTransition simp add: sum-
mandTransition)
next
    assume \(a\{b\} . P^{\prime} \in\) summands \(P\)
    with assms show valid \(P^{\prime}\) by (force intro: validOutputTransition simp add: sum-
mandTransition)
next
    assume \(a<x>. P^{\prime} \in\) summands \(P\)
    with assms show valid \(P^{\prime}\) by (force intro: validInputTransition simp add: sum-
mandTransition)
next
    assume \(<\nu x>a\{x\} . P^{\prime} \in\) summands \(P\) and \(a \neq x\)
    with assms show valid \(P^{\prime}\)
    by (force intro: validBoundOutputTransition simp add: summandTransition[THEN
```

```
sym])
qed
lemma validExpand:
    fixes P :: pi
    and }Q::p
    assumes valid P
    and valid Q
    and uhnf P
    and uhnf Q
    shows }\forallR\in(\mathrm{ expandSet P Q ).uhnf R}\wedge valid 
proof -
    from assms have hnf P and hnf Q by(simp add:uhnf-def)+
    with assms show ?thesis
    apply(auto simp add: expandSet-def)
    apply(force dest: validSummand simp add: uhnf-def)
    apply(force dest: validSummand)
    apply(force dest: validSummand simp add: uhnf-def)
    apply(force dest: validSummand)
    apply(force dest: validSummand simp add: uhnf-def)
    apply(force dest: validSummand)
    apply(force dest: validSummand simp add: uhnf-def)
    apply(force dest: validSummand)
    apply(force dest: validSummand simp add: uhnf-def)
    apply(force dest: validSummand)
    apply(force dest: validSummand simp add: uhnf-def)
    apply(force dest: validSummand)
    apply(subgoal-tac a\not=x)
    apply(force dest: validSummand simp add: uhnf-def)
    apply blast
    apply(subgoal-tac a\not=x)
    apply(drule-tac validSummand(4)) apply assumption+
    apply blast
    apply(subgoal-tac a\not=x)
    apply(drule-tac validSummand(4)[where P=Q]) apply assumption+
    apply(force dest: validSummand simp add: uhnf-def)
    apply blast
    apply(subgoal-tac a\not=x)
    apply(drule-tac validSummand(4)[where P=Q]) apply assumption+
    apply blast
    apply(force dest: validSummand simp add: uhnf-def)
    apply(force dest: validSummand)
    apply(force dest: validSummand simp add: uhnf-def)
    apply(force simp add: uhnf-def)
    apply(force dest: validSummand)
    apply(force dest: validSummand)
    apply(force simp add: uhnf-def)
```

```
    apply(force dest: validSummand)
    apply(subgoal-tac a\not=y)
    apply(drule-tac validSummand(4)[where P=Q]) apply assumption+
    apply blast
    apply(force dest: validSummand simp add: uhnf-def)
    apply(subgoal-tac a\not=y)
    apply(drule-tac validSummand(4)) apply assumption+
    apply blast
    by(force dest: validSummand)
qed
lemma expandComplete:
    fixes F :: pi set
    assumes finite F
    shows }\existsP.(P,F)\in\mathrm{ sumComposeSet
using assms
proof(induct F rule: finite-induct)
    case empty
    have (0, {}) \in sumComposeSet by(rule sumComposeSet.empty)
    thus ?case by blast
next
    case(insert Q F)
    have }\existsP.(P,F)\in sumComposeSet by fac
    then obtain P where (P,F)\in sumComposeSet by blast
    moreover have Q\in insert Q F by simp
    moreover have Q\not\inF by fact
    ultimately have ( }P\oplusQ\mathrm{ , insert Q F) & sumComposeSet
    by(force intro: sumComposeSet.insert)
    thus ?case by blast
qed
lemma expandDepth:
    fixes F :: pi set
    and }P::p
    and }Q::p
    assumes (P,F)\in sumComposeSet
    and F\not={}
    shows }\existsQ\inF.depth P\leqdepth Q\wedge(\forallR\inF.depth R\leqdepth Q
using assms
proof(induct arbitrary:Q rule: sumComposeSet.induct)
    case empty
    have ({}::pi set) }={{}\mathrm{ by fact
    hence False by simp
    thus ?case by simp
next
```

```
case(insert Q SP)
have QinS: Q\inS by fact
show ?case
proof(case-tac (S-{Q})={})
    assume (S-{Q})={}
    with QinS have SeqQ:S={Q} by auto
    have (P,S-{Q})\in sumComposeSet by fact
    with SeqQ have (P,{})\in sumComposeSet by simp
```



```
    with QinS SeqQ show ?case by simp
next
    assume (S-{Q})}\not={
    moreover have (S-{Q})\not={}\Longrightarrow\exists\mp@subsup{Q}{}{\prime}\in(S-{Q}). depth P}\leq\mathrm{ depth Q'
\wedge(\forallR (S - {Q}). depth R \leq depth Q ') by fact
    ultimately obtain }\mp@subsup{Q}{}{\prime}\mathrm{ where }\mp@subsup{Q}{}{\prime}\mathrm{ inS: Q Q}\inS-{Q} and PQ'depth:depth P
sdepth }\mp@subsup{Q}{}{\prime}\mathrm{ and All: }\forallR\in(S-{Q}).depth R\leqdepth Q' by aut
    show ?case
    proof(case-tac Q = Q')
        assume Q = Q '
        with PQ'depth All QinS show ?case by auto
    next
        assume QineqQ':Q}=\mp@subsup{Q}{}{\prime
        show ?case
        proof(case-tac depth Q \leq depth Q )
            assume depth Q\leq depth Q'
            with QineqQ' PQ'depth All Q'inS show ?thesis by force
        next
            assume }\neg\mathrm{ depth Q
            with QineqQ' PQ'depth All Q'inS QinS show ?thesis apply auto
                apply(rule-tac x=Q in bexI)
                apply auto
                apply(case-tac R=Q)
                apply auto
                apply(erule-tac x=R in ballE)
                by auto
        qed
    qed
    qed
qed
lemma depthSubst[simp]:
    fixes P :: pi
    and a :: name
    and b :: name
    shows depth}(P[a::=b])=\operatorname{depth}
by(nominal-induct P avoiding: a b rule: pi.strong-inducts, auto)
lemma depthTransition:
```

```
    fixes P :: pi
    and a :: name
    and }b\mathrm{ :: name
    and }\mp@subsup{P}{}{\prime}::p
    assumes Phnf: hnf P
    shows }P\longmapstoa[b]\prec\mp@subsup{P}{}{\prime}\Longrightarrow\mathrm{ depth }\mp@subsup{P}{}{\prime}<\mathrm{ depth }
    and }P\longmapstoa<x>\prec\mp@subsup{P}{}{\prime}\Longrightarrow\mathrm{ depth }\mp@subsup{P}{}{\prime}<\mathrm{ depth }
    and }P\longmapsto\tau\prec\mp@subsup{P}{}{\prime}\Longrightarrow\mathrm{ depth }\mp@subsup{P}{}{\prime}<\mathrm{ depth }
    and }P\longmapstoa<\nux>\prec\prec\mp@subsup{P}{}{\prime}\Longrightarrow\mathrm{ depth }\mp@subsup{P}{}{\prime}<\mathrm{ depth }
proof -
    assume P}\longmapstoa[b]\prec\mp@subsup{P}{}{\prime
    thus depth }\mp@subsup{P}{}{\prime}<\mathrm{ depth P using assms
    by(nominal-induct rule: outputInduct, auto)
next
    assume Trans: P\longmapstoa<x>\prec }\mp@subsup{P}{}{\prime
    have Goal: \P a x P
P
    proof -
        fix Pax P '
        assume P\longmapstoa<x>\prec \prec P' and }x\sharpP\mathrm{ and hnf P
        thus depth }\mp@subsup{P}{}{\prime}<\mathrm{ depth P
            by(nominal-induct rule: inputInduct, auto)
    qed
    obtain y::name where yFreshP: y\sharpP and yFreshP': y # P'
    by(rule-tac name-exists-fresh[of ( }P,\mp@subsup{P}{}{\prime})]\mathrm{ ], auto simp add: fresh-prod)
from yFreshP' Trans have }P\longmapstoa<y>\prec[(x,y)]\cdot\mp@subsup{P}{}{\prime}\mathrm{ by (simp add: alphaBound-
Residual)
    hence depth }([(x,y)]\cdot\mp@subsup{P}{}{\prime})<\operatorname{depth}P\mathrm{ using yFreshP Phnf by(rule Goal)
    thus depth }\mp@subsup{P}{}{\prime}<\mathrm{ depth P by simp
next
    assume P\longmapsto\tau \prec P'
    thus depth }\mp@subsup{P}{}{\prime}<\mathrm{ depth P using assms
        by(nominal-induct rule: tauInduct, auto simp add: uhnf-def)
next
    assume Trans: P\longmapstoa<\nux>}\prec\mp@subsup{P}{}{\prime
    have Goal: \P a x P'. \llbracketP\longmapstoa<\nux> \prec P'; x\sharpP; hnf P\rrbracket\Longrightarrow depth }\mp@subsup{P}{}{\prime}<\mathrm{ depth
P
    proof -
        fix Pax P '
        assume P}\longmapstoa<\nux>\prec\prec\mp@subsup{P}{}{\prime}\mathrm{ and }x\sharpP\mathrm{ and hnf P
        thus depth }\mp@subsup{P}{}{\prime}<\mathrm{ depth P
            by(nominal-induct rule: boundOutputInduct,
                auto elim:outputCases simp add: residual.inject)
    qed
    obtain y::name where yFreshP: y\sharpP and yFreshP': y\sharp 的
    by(rule-tac name-exists-fresh[of (P, P
    from yFreshP' Trans have P\longmapstoa<\nuy> \prec [(x,y)] \cdot P' by(simp add: al-
```

```
phaBoundResidual)
    hence depth ([(x,y)] \cdot P
    thus depth }\mp@subsup{P}{}{\prime}<\mathrm{ depth P by simp
qed
lemma maxExpandDepth:
    fixes P :: pi
    and }Q::p
    and }R::p
    assumes R\in expandSet P Q
    and hnf P
    and hnf Q
    shows depth R}\leq\operatorname{depth}(P|Q
using assms
apply(auto simp add: expandSet-def summandTransition[THEN sym] dest: depth-
Transition)
apply(subgoal-tac a}=x\mathrm{ )
apply(simp add: summandTransition[THEN sym])
apply(force dest: depthTransition)
apply blast
apply(subgoal-tac a\not=x)
apply(simp add: summandTransition[THEN sym])
apply(force dest: depthTransition)
apply blast
apply(force dest: depthTransition)
apply(force dest: depthTransition)
apply(subgoal-tac a\not= y)
apply(simp add: summandTransition[THEN sym])
apply(force dest: depthTransition)
apply blast
apply(subgoal-tac a\not= y)
apply(simp add: summandTransition[THEN sym])
apply(force dest: depthTransition)
by blast
lemma expandDepth':
    fixes P :: pi
    and }Q::p
    assumes Phnf: hnf P
    and Qhnf:hnf Q
    shows }\existsR.(R, expandSet PQ)\in\operatorname{sumComposeSet }\wedge\operatorname{depth}R\leq\operatorname{depth}(P|Q
proof(case-tac expandSet P Q = {})
    assume expandSet P Q = {}
    with Phnf Qhnf show ?thesis by(auto intro: sumComposeSet.empty)
next
```

assume expandSet $P Q \neq\{ \}$
moreover from Phnf Qhnf finiteExpand obtain $R$ where TSC: ( $R$, expandSet $P(Q) \in$ sumComposeSet by(blast dest: expandComplete)
ultimately obtain $T$ where $T \in \operatorname{expandSet} P Q$ and depth $R \leq$ depth $T$
by(blast dest: expandDepth)
with Phnf Qhnf have depth $R \leq \operatorname{depth}(P \| Q)$
by (force dest: maxExpandDepth)
with TSC show ?thesis by blast
qed
lemma validToHnf:
fixes $P$ :: pi
assumes valid $P$
shows $\exists Q$. uhnf $Q \wedge$ valid $Q \wedge Q \equiv_{e} P \wedge($ depth $Q) \leq($ depth $P)$ proof -
have MatchGoal: $\left\lfloor a b P Q . \llbracket u h n f Q ;\right.$ valid $Q ; Q \equiv{ }_{e} P ;$ depth $Q \leq$ depth $P \rrbracket \Longrightarrow$
$\exists Q$. uhnf $Q \wedge$ valid $Q \wedge Q \equiv_{e}[a \frown b] P \wedge$ depth $Q \leq$ depth
$([a \frown b] P)$
proof -
fix $a b P Q$
assume $Q h n f:$ uhnf $Q$ and validQ: valid $Q$ and $Q e q P: Q \equiv_{e} P$
and QPdepth: depth $Q \leq$ depth $P$
show $\exists Q$. uhnf $Q \wedge$ valid $Q \wedge Q \equiv_{e}[a \frown b] P \wedge$ depth $Q \leq \operatorname{depth}([a \frown b] P)$
proof (case-tac $a=b$ )
assume $a=b$
with $Q e q P$ have $Q \equiv_{e}[a \frown b] P$ by (blast intro: Sym Trans equiv.Match)
with Qhnf validQ QPdepth show ?thesis by force
next
assume $a \neq b$
hence $\mathbf{0} \equiv_{e}[a \frown b] P \mathbf{b y}$ (blast intro: Sym Match')
moreover have uhnf $\mathbf{0}$ by (simp add: uhnf-def)
ultimately show? ?thesis by force
qed
qed
from assms show ?thesis
proof(nominal-induct P rule: pi.strong-inducts)
case PiNil
have uhnf $\mathbf{0} \mathbf{b y}(s i m p$ add: uhnf-def)
moreover have valid $\mathbf{0}$ by simp
moreover have $\mathbf{0} \equiv_{e} \mathbf{0}$ by (rule Refl)
moreover have (depth $\mathbf{0}) \leq($ depth $\mathbf{0})$ by simp
ultimately show ?case by blast
next

```
    case(Output a b P)
    have uhnf (a{b}.P) by (simp add: uhnf-def)
    moreover have valid(a{b}.P) by fact
    moreover have a{b}.P \equiv
    moreover have (depth (a{b}.P))}\leq(\mathrm{ depth (a{b}.P)) by simp
    ultimately show ?case by blast
next
    case(Tau P)
    have uhnf (\tau.(P)) by(simp add:uhnf-def)
    moreover have valid (\tau.(P)) by fact
    moreover have }\tau.(P)\mp@subsup{\equiv}{e}{}\tau.(P)\boldsymbol{by}(rule Refl
    moreover have (depth (\tau.(P)))}\leq(\mathrm{ depth }(\tau.(P)))\mathrm{ by simp
    ultimately show ?case by blast
next
    case(Input a x P)
    have uhnf ( }a<x>.P)\mathrm{ by(simp add:uhnf-def)
    moreover have valid ( }a<x>.P\mathrm{ ) by fact
    moreover have }a<x>.P\mp@subsup{\equiv}{e}{}a<x>.P\mathrm{ by(rule Refl)
    moreover have (depth (a<x>.P)) \leq (depth (a<x>.P)) by simp
    ultimately show ?case by blast
next
    case(Match a b P)
    have valid ([a\frownb]P) by fact
    hence valid P by simp
    moreover have valid P\Longrightarrow\existsQ.uhnf Q}\wedge valid Q\wedgeQ \equiv\mp@subsup{e}{e}{}P\wedge\mathrm{ depth Q}
depth P by fact
    ultimately obtain Q where Qhnf:uhnf Q and validQ: valid Q and QeqP:
Q \equiv
                            and QPdepth: depth Q \leq depth P by blast
    thus ?case by(rule MatchGoal)
next
    case(Mismatch a b P)
    have valid ([a\not=b]P) by fact
    hence valid P by simp
    moreover have valid P\Longrightarrow\existsQ. uhnf Q}\\mathrm{ valid }Q\wedgeQ\equiv\mp@subsup{\equiv}{e}{}P\wedge\mathrm{ depth }Q
depth P by fact
    ultimately obtain Q where Qhnf:uhnf Q and validQ: valid Q and QeqP:
Q =e P
                            and QPdepth: depth Q { depth P by blast
show ?case
proof(case-tac a=b)
    assume a=b
    hence 0 = = [a\not=b]P by(blast intro:Sym Mismatch')
    moreover have uhnf 0 by(simp add: uhnf-def)
    ultimately show ?case by force
next
    assume a\not=b
    with QeqP have Q \equiv
    with Qhnf validQ QPdepth show ?case by force
```


## qed

next
case $($ Sum P $Q$ )
have $\operatorname{valid}(P \oplus Q)$ by fact
hence validP: valid $P$ and valid $Q$ : valid $Q$ by simp+
have $\exists P^{\prime}$. uhnf $P^{\prime} \wedge$ valid $P^{\prime} \wedge P^{\prime} \equiv{ }_{e} P \wedge\left(\right.$ depth $\left.P^{\prime}\right) \leq($ depth $P)$
proof -
have valid $P \Longrightarrow \exists P^{\prime}$. uhnf $P^{\prime} \wedge$ valid $P^{\prime} \wedge P^{\prime} \equiv{ }_{e} P \wedge\left(\right.$ depth $\left.P^{\prime}\right) \leq($ depth
$P)$ by fact
with validP show ?thesis by simp
qed
then obtain $P^{\prime}$ where $P^{\prime} h n f:$ uhnf $P^{\prime}$ and $P^{\prime} e q P: P^{\prime} \equiv{ }_{e} P$ and validP': valid $P^{\prime}$

$$
\text { and } P^{\prime} \text { depth: }\left(\text { depth } P^{\prime}\right) \leq(\text { depth } P) \text { by blast }
$$

have $\exists Q^{\prime}$. uhnf $Q^{\prime} \wedge$ valid $Q^{\prime} \wedge Q^{\prime} \equiv{ }_{e} Q \wedge\left(\right.$ depth $\left.Q^{\prime}\right) \leq($ depth $Q)$ proof -
have valid $Q \Longrightarrow \exists Q^{\prime}$. uhnf $Q^{\prime} \wedge$ valid $Q^{\prime} \wedge Q^{\prime} \equiv_{e} Q \wedge\left(\right.$ depth $\left.Q^{\prime}\right) \leq($ depth $Q)$ by fact
with validQ show ?thesis by simp
qed
then obtain $Q^{\prime}$ where $Q^{\prime} h n f: u h n f Q^{\prime}$ and $Q^{\prime} e q Q: Q^{\prime} \equiv_{e} Q$ and validQ $Q^{\prime}$ : valid $Q^{\prime}$
and $Q^{\prime}$ depth: $\left(\right.$ depth $\left.Q^{\prime}\right) \leq($ depth $Q)$ by blast
from $P^{\prime} h n f Q^{\prime} h n f$ validP' valid $Q^{\prime}$ obtain $R$ where $R h n f$ : uhnf $R$ and validR: valid $R$
and $P^{\prime} Q^{\prime} e q R: P^{\prime} \oplus Q^{\prime} \equiv_{e} R$
and Rdepth: depth $R \leq \operatorname{depth}\left(P^{\prime} \oplus Q^{\prime}\right)$
apply (drule-tac uhnfSum) apply assumption + by blast
from validP $P^{\prime}$ valid $Q^{\prime}$ have $\operatorname{valid}\left(P^{\prime} \oplus Q^{\prime}\right)$ by simp
from $P^{\prime} e q P Q^{\prime} e q Q P^{\prime} Q^{\prime} e q R$ have $P \oplus Q \equiv_{e} R$ by (blast intro: Sym SumPres ${ }^{\prime}$ Trans)
moreover from $R$ depth $P^{\prime}$ depth $Q^{\prime}$ depth have depth $R \leq \operatorname{depth}(P \oplus Q)$ by auto
ultimately show ?case using validR Rhnf by (blast intro: Sym)

## next

case $(\operatorname{Par} P Q)$
have $\operatorname{valid}(P \| Q)$ by fact
hence validP: valid $P$ and valid $Q$ : valid $Q$ by simp+
have $\exists P^{\prime}$. uhnf $P^{\prime} \wedge$ valid $P^{\prime} \wedge P^{\prime} \equiv{ }_{e} P \wedge\left(\right.$ depth $\left.P^{\prime}\right) \leq($ depth $P)$
proof -
have valid $P \Longrightarrow \exists P^{\prime}$. uhnf $P^{\prime} \wedge$ valid $P^{\prime} \wedge P^{\prime} \equiv{ }_{e} P \wedge\left(\right.$ depth $\left.P^{\prime}\right) \leq($ depth $P)$ by fact
with validP show ?thesis by simp
qed
then obtain $P^{\prime}$ where $P^{\prime} h n f:$ uhnf $P^{\prime}$ and $P^{\prime} e q P: P^{\prime} \equiv_{e} P$ and valid $P^{\prime}$ : valid $P^{\prime}$ and $P^{\prime}$ depth: $\left(\right.$ depth $\left.P^{\prime}\right) \leq($ depth $P)$ by blast
have $\exists Q^{\prime}$. uhnf $Q^{\prime} \wedge$ valid $Q^{\prime} \wedge Q^{\prime} \equiv_{e} Q \wedge\left(\right.$ depth $\left.Q^{\prime}\right) \leq($ depth $Q)$ proof -
have valid $Q \Longrightarrow \exists Q^{\prime}$. uhnf $Q^{\prime} \wedge$ valid $Q^{\prime} \wedge Q^{\prime} \equiv_{e} Q \wedge\left(\right.$ depth $\left.Q^{\prime}\right) \leq($ depth $Q)$ by fact
with validQ show ?thesis by simp
qed
then obtain $Q^{\prime}$ where $Q^{\prime} h n f: u h n f Q^{\prime}$ and $Q^{\prime} e q Q: Q^{\prime} \equiv_{e} Q$ and validQ': valid $Q^{\prime}$
and $Q^{\prime}$ depth: $\left(\right.$ depth $\left.Q^{\prime}\right) \leq($ depth $Q)$ by blast
from $P^{\prime} h n f Q^{\prime} h n f$ obtain $R$ where Exp: $\left(R\right.$, expandSet $\left.P^{\prime} Q^{\prime}\right) \in$ sumCompos$e S e t$ and Rdepth: depth $R \leq \operatorname{depth}\left(P^{\prime} \| Q^{\prime}\right)$
by (force dest: expandDepth' simp add: uhnf-def)
from Exp $P^{\prime} h n f Q^{\prime} h n f$ have $P^{\prime} Q^{\prime}$ eqR: $P^{\prime} \| Q^{\prime} \equiv_{e} R$ by (force intro: Expand simp add: uhnf-def)
from $P^{\prime} h n f Q^{\prime} h n f$ validP $P^{\prime}$ valid $Q^{\prime}$ have $\forall P \in\left(\right.$ expandSet $\left.P^{\prime} Q^{\prime}\right)$. uhnf $P \wedge$ valid $P$ by (blast dest: validExpand)
with Exp obtain $R^{\prime}$ where $R^{\prime} h n f$ : uhnf $R^{\prime}$ and validR': valid $R^{\prime}$
and $R e q R^{\prime}: R \equiv{ }_{e} R^{\prime}$
and $R^{\prime}$ depth: depth $R^{\prime} \leq$ depth $R$
by(blast dest: expandHnf)
from $P^{\prime} e q P Q^{\prime} e q Q P^{\prime} Q^{\prime} e q R$ ReqR $R^{\prime}$ have $P \| Q \equiv_{e} R^{\prime}$ by(blast intro: Sym ParPres Trans)
moreover from $R$ depth $P^{\prime}$ depth $Q^{\prime}$ depth $R^{\prime}$ depth have depth $R^{\prime} \leq \operatorname{depth}(P \|$ Q) by auto
ultimately show ?case using validR' $R^{\prime} h n f$ by (blast dest: Sym)
next
case (Res x P)
have valid $(<\nu x>P)$ by fact
hence validP: valid $P$ by simp
moreover have valid $P \Longrightarrow \exists Q$. uhnf $Q \wedge$ valid $Q \wedge Q \equiv_{e} P \wedge$ depth $Q \leq$ depth $P$ by fact
ultimately obtain $Q$ where $Q h n f:$ uhnf $Q$ and validQ: valid $Q$ and $Q e q P$ : $Q \equiv{ }_{e} P$
and QPDepth: depth $Q \leq$ depth $P$ by blast
from validP show ?case
proof(nominal-induct $P$ avoiding: x rule: pi.strong-inducts)
case PiNil
have uhnf $\mathbf{0}$ by (simp add: uhnf-def)
moreover have valid $\mathbf{0}$ by simp
moreover have $\mathbf{0} \equiv_{e}<\nu x>0$

```
    proof -
        have }x\sharp\mathbf{0}\mathrm{ by simp
        thus ?thesis by(blast intro: Sym ResFresh)
    qed
    moreover have depth 0}\leq\mathrm{ depth ( }<\nux>\mathbf{0})\mathrm{ by simp
    ultimately show ?case by blast
    next
    case(Output a b P)
    have valid(a{b}.P) by fact
    hence validP: valid P by simp
    show ?case
    proof(case-tac x=a)
        assume }x=
        moreover have uhnf 0 by(simp add: uhnf-def)
        moreover have valid 0 by simp
        moreover have 0 }\mp@subsup{\equiv}{e}{<}<\nux>x{b}.P by(blast intro: ResOutput' Sym
        moreover have depth 0}\leq\mathrm{ depth ( < vx>x{b}.P) by simp
        ultimately show ?case by blast
    next
    assume xineqa: x\not=a
    show ?case
    proof (case-tac x=b)
        assume }x=
            moreover from xineqa have uhnf(<\nux>a{x}.P) by(force simp add:
uhnf-def)
            moreover from validP have valid (<\nux>a{x}.P) by simp
            moreover have <\nux>a{x}.P \equiv
            moreover have depth (<\nux>a{x}.P) \leq depth(<\nux>a{x}.P) by simp
            ultimately show ?case by blast
            next
            assume xineqb: }x\not=
            have uhnf(a{b}.(<\nux>P)) by(simp add:uhnf-def)
            moreover from validP have valid(a{b}.(<\nux>P)) by simp
                moreover from xineqa xineqb have a{b}.(<\nux>P) \equiv
by(blast intro: ResOutput Sym)
            moreover have depth(a{b}.(<\nux>PP))\leq\operatorname{depth}(<\nux>a{b}.P) by simp
            ultimately show ?case by blast
        qed
    qed
    next
    case(Tau P)
    have valid(\tau.(P)) by fact
    hence validP: valid P by simp
    have uhnf(\tau.(<\nux>P)) by(simp add:uhnf-def)
    moreover from validP have valid(\tau.(<\nux>P)) by simp
    moreover have }\tau.(<\nux>P)\mp@subsup{\equiv}{e}{}<\nux>\tau.(P) by(blast intro:ResTau Sym
    moreover have depth (\tau.(<\nux>>P))\leq\operatorname{depth}(<\nux>\tau.(P)) by simp
    ultimately show ?case by blast
```


## next

case (Input a y $P$ )
have $\operatorname{valid}(a<y>. P)$ by fact
hence validP: valid $P$ by simp
have $y \sharp x$ by fact hence yineqx: $y \neq x$ by simp
show ?case
proof (case-tac $x=a$ )
assume $x=a$
moreover have uhnf $\mathbf{0}$ by (simp add: uhnf-def)
moreover have valid 0 by simp
moreover have $\mathbf{0} \equiv_{e}<\nu x>x<y>$.P by(blast intro: ResInput' Sym)
moreover have depth $\mathbf{0} \leq \operatorname{depth}(<\nu x>x<y>. P)$ by simp
ultimately show ? case by blast
next
assume xineqa: $x \neq a$
have $u h n f(a<y>.(<\nu x>P))$ by (simp add: uhnf-def)
moreover from validP have valid $(a<y>.(<\nu x>P))$ by simp
moreover from xineqa yineqx have $a<y>.(<\nu x>P) \equiv{ }_{e}<\nu x>a<y>. P$
by(blast intro: ResInput Sym)
moreover have $\operatorname{depth}(a<y>.(<\nu x>P)) \leq \operatorname{depth}(<\nu x>a<y>. P)$ by $\operatorname{simp}$
ultimately show ? case by blast
qed
next
case (Match a b $P$ x)
have $\operatorname{valid}([a \frown b] P)$ by fact hence valid $P$ by $\operatorname{simp}$
moreover have $\bigwedge x$. valid $P \Longrightarrow \exists Q$. uhnf $Q \wedge \operatorname{valid} Q \wedge Q \equiv{ }_{e}<\nu x>P \wedge$ depth $Q \leq \operatorname{depth}(<\nu x>P)$
by fact
ultimately obtain $Q$ where $Q h n f: u h n f ~ Q ~ a n d ~ v a l i d Q: ~ v a l i d ~ Q ~$
and $Q e q P: Q \equiv_{e}(<\nu x>P)$
and QPdepth: depth $Q \leq \operatorname{depth}(<\nu x>P)$
by blast
show ?case
$\operatorname{proof}($ case-tac $a=b)$
assume $a=b$
moreover have $Q \equiv_{e}<\nu x>[a \frown a] P$
proof -
have $P \equiv{ }_{e}[a \frown a] P$ by (blast intro: equiv.Match Sym)
hence $<\nu x>P \equiv{ }_{e}<\nu x>[a \frown a] P$ by (rule ResPres)
with QeqP show ?thesis by(blast intro: Trans)
qed
moreover from $Q P d e p t h$ have depth $Q \leq \operatorname{depth}(<\nu x>[a \frown a] P)$ by $\operatorname{simp}$
ultimately show ? case using Qhnf validQ by blast
next
assume aineqb: $a \neq b$
have uhnf $\mathbf{0}$ by (simp add: uhnf-def)
moreover have valid $\mathbf{0}$ by simp
moreover have $\mathbf{0} \equiv_{e}<\nu x>[a \frown b] P$
proof -

```
            from aineqb have \(\mathbf{0} \equiv_{e}[a \frown b] P\) by (blast intro: Match \(^{\prime}\) Sym)
            hence \(<\nu x>\mathbf{0} \equiv_{e}<\nu x>[a \frown b] P\) by (rule ResPres)
            thus ?thesis by (blast intro: ResNil Trans Sym)
qed
moreover have depth \(\mathbf{0} \leq \operatorname{depth}(<\nu x\rangle[a \frown b] P)\) by simp
ultimately show ?case by blast
    qed
next
    case(Mismatch a b P x)
    have valid \(([a \neq b] P)\) by fact hence valid \(P\) by simp
    moreover have \(\wedge x\). valid \(P \Longrightarrow \exists Q\). uhnf \(Q \wedge\) valid \(Q \wedge Q \equiv_{e}<\nu x>P \wedge\)
                        depth \(Q \leq \operatorname{depth}(<\nu x>P)\)
    by fact
    ultimately obtain \(Q\) where \(Q h n f:\) uhnf \(Q\) and valid \(Q\) : valid \(Q\)
                and \(Q e q P: Q \equiv_{e}(<\nu x>P)\)
                and QPdepth: depth \(Q \leq \operatorname{depth}(<\nu x>P)\)
    by blast
show ?case
    \(\operatorname{proof}(\) case-tac \(a=b\) )
    assume \(a=b\)
    moreover have uhnf \(\mathbf{0}\) by (simp add: uhnf-def)
    moreover have valid 0 by simp
    moreover have \(\mathbf{0} \equiv_{e}\langle\nu x\rangle[a \neq a] P\)
    proof -
            have \(\mathbf{0} \equiv_{e}[a \neq a] P \operatorname{by}\left(b l a s t\right.\) intro: Mismatch \({ }^{\prime}\) Sym)
            hence \(<\nu x>\mathbf{0} \equiv_{e}<\nu x>[a \neq a] P\) by(rule ResPres)
            thus ?thesis by(blast intro: ResNil Trans Sym)
    qed
    moreover have depth \(\mathbf{0} \leq \operatorname{depth}(<\nu x\rangle[a \neq a] P)\) by simp
    ultimately show ?case by blast
next
    assume aineqb: \(a \neq b\)
    have \(Q \equiv_{e}<\nu x>[a \neq b] P\)
    proof -
            from aineqb have \(P \equiv_{e}[a \neq b] P\) by(blast intro: equiv.Mismatch Sym)
            hence \(\langle\nu x\rangle P \equiv_{e}\langle\nu x\rangle[a \neq b] P\) by(rule ResPres)
            with \(Q e q P\) show ?thesis by (blast intro: Trans)
    qed
    moreover from \(Q P\) depth have depth \(Q \leq \operatorname{depth}(<\nu x\rangle[a \neq b] P)\) by simp
    ultimately show ?case using Qhnf validQ by blast
    qed
next
    case (Sum P \(Q x\) )
    have \(\operatorname{valid}(P \oplus Q)\) by fact
    hence validP: valid \(P\) and valid \(Q\) : valid \(Q\) by simp+
have \(\exists P^{\prime}\). uhnf \(P^{\prime} \wedge\) valid \(P^{\prime} \wedge P^{\prime} \equiv_{e}<\nu x>P \wedge\left(\right.\) depth \(\left.P^{\prime}\right) \leq(\operatorname{depth}(<\nu x>P))\)
proof -
    have valid \(P \Longrightarrow \exists P^{\prime}\). uhnf \(P^{\prime} \wedge\) valid \(P^{\prime} \wedge P^{\prime} \equiv_{e}<\nu x>P \wedge\left(\right.\) depth \(\left.P^{\prime}\right) \leq\)
```

(depth $(<\nu x>P))$ by fact
with validP show? ?thesis by simp
qed
then obtain $P^{\prime}$ where $P^{\prime} h n f: u h n f P^{\prime}$ and $P^{\prime} e q P: P^{\prime} \equiv_{e}<\nu x>P$ and validP': valid $P^{\prime}$ and $P^{\prime}$ depth: $\left(\right.$ depth $\left.P^{\prime}\right) \leq($ depth $(<\nu x>P))$ by blast
have $\exists Q^{\prime}$. uhnf $Q^{\prime} \wedge$ valid $Q^{\prime} \wedge Q^{\prime} \equiv_{e}<\nu x>Q \wedge\left(\right.$ depth $\left.Q^{\prime}\right) \leq(\operatorname{depth}(<\nu x>Q))$
proof -
have valid $Q \Longrightarrow \exists Q^{\prime}$. uhnf $Q^{\prime} \wedge$ valid $Q^{\prime} \wedge Q^{\prime} \equiv_{e}<\nu x>Q \wedge\left(\right.$ depth $\left.Q^{\prime}\right)$
$\leq($ depth $(<\nu x>Q))$ by fact
with valid $Q$ show ?thesis by simp
qed
then obtain $Q^{\prime}$ where $Q^{\prime} h n f:$ uhnf $Q^{\prime}$ and $Q^{\prime} e q Q: Q^{\prime} \equiv_{e}<\nu x>Q$ and validQ': valid $Q^{\prime}$
and $Q^{\prime}$ depth: $\left(\right.$ depth $\left.Q^{\prime}\right) \leq($ depth $(<\nu x>Q))$ by blast
from $P^{\prime} h n f Q^{\prime} h n f$ valid $P^{\prime}$ valid $Q^{\prime}$ obtain $R$ where Rhnf: uhnf $R$ and validR: valid $R$
and $P^{\prime} Q^{\prime} e q R: P^{\prime} \oplus Q^{\prime} \equiv{ }_{e} R$
and Rdepth: depth $R \leq \operatorname{depth}\left(P^{\prime} \oplus Q^{\prime}\right)$
apply (drule-tac uhnfSum) apply assumption+ by blast
from $P^{\prime} e q P Q^{\prime} e q Q P^{\prime} Q^{\prime} e q R$ have $<\nu x>(P \oplus Q) \equiv_{e} R$ by (blast intro: Sym SumPres' SumRes Trans)
moreover from $R$ depth $P^{\prime}$ depth $Q^{\prime}$ depth have depth $R \leq \operatorname{depth}(<\nu x>(P \oplus$ Q)) by auto
ultimately show ?case using validR Rhnf by (blast intro: Sym)
next
case $(\operatorname{Par} P \quad Q x)$
have $\operatorname{valid}(P \| Q)$ by fact
hence validP: valid $P$ and valid $Q$ : valid $Q$ by simp+
have $\exists P^{\prime}$. uhnf $P^{\prime} \wedge$ valid $P^{\prime} \wedge P^{\prime} \equiv{ }_{e} P \wedge\left(\right.$ depth $\left.P^{\prime}\right) \leq($ depth $P)$
proof -
obtain $x$ ::name where $x$ FreshP: $x \sharp P$ by(rule name-exists-fresh)
moreover have $\wedge x$. valid $P \Longrightarrow \exists P^{\prime}$. uhnf $P^{\prime} \wedge \operatorname{valid} P^{\prime} \wedge P^{\prime} \equiv_{e}(<\nu x>P)$ $\wedge\left(\right.$ depth $\left.P^{\prime}\right) \leq($ depth $(<\nu x>P))$ by fact
with valid $P$ obtain $P^{\prime}$ where uhnf $P^{\prime}$ and valid $P^{\prime}$ and $P^{\prime} e q P: P^{\prime} \equiv_{e}$ $(<\nu x>P)$ and $P^{\prime}$ depth: $\left(\right.$ depth $\left.P^{\prime}\right) \leq($ depth $(<\nu x>P))$ by blast
moreover from $x$ Fresh $P P^{\prime} e q P$ have $P^{\prime} \equiv{ }_{e} P$ by(blast intro: Trans ResFresh)
moreover with $P^{\prime}$ depth have depth $P^{\prime} \leq$ depth $P$ by simp
ultimately show ?thesis by blast
qed
then obtain $P^{\prime}$ where $P^{\prime} h n f:$ uhnf $P^{\prime}$ and $P^{\prime} e q P: P^{\prime} \equiv_{e} P$ and validP': valid $P^{\prime}$
and $P^{\prime}$ depth: $\left(\right.$ depth $\left.P^{\prime}\right) \leq($ depth $P)$ by blast
have $\exists Q^{\prime}$. uhnf $Q^{\prime} \wedge$ valid $Q^{\prime} \wedge Q^{\prime} \equiv_{e} Q \wedge\left(\right.$ depth $\left.Q^{\prime}\right) \leq($ depth $Q)$
proof -
obtain $x$ ::name where $x$ Fresh $Q: x \sharp Q$ by(rule name-exists-fresh)
moreover have $\bigwedge x$. valid $Q \Longrightarrow \exists Q^{\prime}$. uhnf $Q^{\prime} \wedge$ valid $Q^{\prime} \wedge Q^{\prime} \equiv_{e}(<\nu x>Q)$ $\wedge\left(\right.$ depth $\left.Q^{\prime}\right) \leq($ depth $(<\nu x>Q))$ by fact
with valid $Q$ obtain $Q^{\prime}$ where uhnf $Q^{\prime}$ and valid $Q^{\prime}$ and $Q^{\prime} e q Q: Q^{\prime} \equiv_{e}$ $(<\nu x>Q)$ and $Q^{\prime}$ depth: $\left(\right.$ depth $\left.Q^{\prime}\right) \leq($ depth $(<\nu x>Q))$ by blast
moreover from $x$ Fresh $Q Q^{\prime} e q Q$ have $Q^{\prime} \equiv_{e} Q$ by(blast intro: Trans ResFresh)
moreover with $Q^{\prime}$ depth have depth $Q^{\prime} \leq$ depth $Q$ by simp
ultimately show ?thesis by blast
qed
then obtain $Q^{\prime}$ where $Q^{\prime} h n f:$ uhnf $Q^{\prime}$ and $Q^{\prime} e q Q: Q^{\prime} \equiv_{e} Q$ and validQ': valid $Q^{\prime}$

$$
\text { and } Q^{\prime} \text { depth: }\left(\text { depth } Q^{\prime}\right) \leq(\text { depth } Q) \text { by blast }
$$

from $P^{\prime} h n f Q^{\prime} h n f$ obtain $R$ where Exp: $\left(R\right.$, expandSet $\left.P^{\prime} Q^{\prime}\right) \in$ sumComposeSet and Rdepth: depth $R \leq \operatorname{depth}\left(P^{\prime} \| Q^{\prime}\right)$
by (force dest: expandDepth' simp add: uhnf-def)
from Exp $P^{\prime} h n f Q^{\prime} h n f$ have $P^{\prime} Q^{\prime} e q R: P^{\prime} \| Q^{\prime} \equiv_{e} R$ by (force intro: Expand simp add: uhnf-def)
from $P^{\prime} h n f Q^{\prime} h n f$ validP ${ }^{\prime}$ valid $Q^{\prime}$ have $\forall P \in\left(\right.$ expandSet $\left.P^{\prime} Q^{\prime}\right)$. uhnf $P \wedge$ valid $P$ by (blast dest: validExpand)
with Exp obtain $R^{\prime}$ where $R^{\prime} h n f$ : uhnf $R^{\prime}$ and validR': valid $R^{\prime}$
and $R e q R^{\prime}: R \equiv{ }_{e} R^{\prime}$
and $R^{\prime}$ depth: depth $R^{\prime} \leq$ depth $R$
by(blast dest: expandHnf)
from $P^{\prime}$ eqP $Q^{\prime} e q Q P^{\prime} Q^{\prime}$ eqR ReqR $R^{\prime}$ have $P \| Q \equiv_{e} R^{\prime}$ by (blast intro: Sym ParPres Trans)
hence ResTrans: $<\nu x>(P \| Q) \equiv_{e}<\nu x>R^{\prime}$ by (rule ResPres)
from validR' $R^{\prime} h n f$ obtain $R^{\prime \prime}$ where $R^{\prime \prime} h n f$ : uhnf $R^{\prime \prime}$ and valid $R^{\prime \prime}$ : valid $R^{\prime \prime}$ and $R^{\prime} e q R^{\prime \prime}:<\nu x>R^{\prime} \equiv_{e} R^{\prime \prime}$ and $R^{\prime \prime}$ depth: depth $R^{\prime \prime} \leq \operatorname{depth}\left(<\nu x>R^{\prime}\right)$
by (force dest: uhnfRes)
from ResTrans $R^{\prime} e q R^{\prime \prime}$ have $<\nu x>(P \| Q) \equiv{ }_{e} R^{\prime \prime}$ by(rule Trans)
moreover from Rdepth $P^{\prime}$ depth $Q^{\prime}$ depth $R^{\prime}$ depth $R^{\prime \prime}$ depth have depth $R^{\prime \prime} \leq$ $\operatorname{depth}(<\nu x\rangle(P \| Q))$ by auto
ultimately show ?case using valid $R^{\prime \prime} R^{\prime \prime} h n f$ by (blast dest: Sym)
next
case (Res y $P x$ )
have $\operatorname{valid}(<\nu y>P)$ by fact hence valid $P$ by simp
moreover have $\bigwedge x$. valid $P \Longrightarrow \exists Q$. uhnf $Q \wedge$ valid $Q \wedge Q \equiv_{e}<\nu x>P \wedge$
depth $Q \leq$ depth $(<\nu x>P)$
by fact
ultimately obtain $Q$ where $Q h n f: u h n f ~ Q$ and valid $Q$ : valid $Q$ and $Q e q P$ :

$$
Q \equiv_{e}<\nu y>P
$$

and QPdepth: depth $Q \leq \operatorname{depth}(<\nu y>P)$ by blast
from Qhnf validQ obtain $Q^{\prime}$ where $Q^{\prime} h n f$ : uhnf $Q^{\prime}$ and valid $Q^{\prime}$ : valid $Q^{\prime}$ and $Q e q Q^{\prime}:<\nu x>Q \equiv_{e} Q^{\prime}$ and $Q^{\prime}$ Qdepth: depth $Q^{\prime} \leq \operatorname{depth}(<\nu x>Q)$
by (force dest: uhnfRes)
from QeqP have $<\nu x>Q \equiv_{e}<\nu x><\nu y>P$ by(rule ResPres)
with $Q e q Q^{\prime}$ have $Q^{\prime} \equiv_{e}<\nu x><\nu y>P$ by(blast intro: Trans Sym)
moreover from $Q^{\prime} Q$ depth $Q P$ depth have depth $\left.Q^{\prime} \leq \operatorname{depth}(<\nu x\rangle<\nu y>P\right)$ by $\operatorname{simp}$
ultimately show?case using $Q^{\prime} h n f$ valid $Q^{\prime}$ by blast
next
case (Bang $P$ x)
have $\operatorname{valid}(!P)$ by fact
hence False by simp
thus?case by simp
qed
next
case (Bang P)
have valid $(!P)$ by fact
hence False by simp
thus ?case by simp
qed
qed
lemma depthZero:
fixes $P:: p i$
assumes depth $P=0$
and uhnf $P$
shows $P=\mathbf{0}$
using assms
apply(nominal-induct $P$ rule: pi.strong-inducts, auto simp add: uhnf-def max-def if-split)
apply (case-tac depth pi1 $\leq$ depth pi2)
by auto
lemma completeAux:
fixes $n::$ nat
and $P:: p i$
and $\quad Q:: p i$
assumes depth $P+$ depth $Q \leq n$
and valid $P$
and valid $Q$
and uhnf $P$

```
    and uhnf Q
    and }\quadP~
    shows P}\equiv\mp@subsup{}{e}{}
using assms
proof(induct n arbitrary: P Q rule: nat.induct)
    case(zero P Q)
    have depth P}+\mathrm{ depth Q}\leq0\mathrm{ by fact
    hence Pdepth: depth P}=0\mathrm{ and Qdepth: depth Q = 0 by auto
    moreover have uhnf P and uhnf Q by fact+
    ultimately have P=0 and Q = 0 by(blast intro:depthZero)+
    thus ?case by(blast intro: Refl)
next
    case(Suc n P Q)
    have validP: valid P and validQ: valid Q by fact+
    have Phnf: uhnf P and Qhnf: uhnf Q by fact+
    have PBisimQ: P ~ Q by fact
    have IH:\bigwedgePQ.\llbracketdepth P + depth Q \leqn; valid P; valid Q;uhnf P; uhnf Q;P
~ Q\rrbracket\LongrightarrowP \equiv
    by fact
    have PQdepth: depth P + depth Q}\leq\mathrm{ Suc n by fact
    have Goal: }\PQ Q Q'.\llbracketdepth P + depth Q \leq Suc n; valid P; valid Q; uhnf P
uhnf Q;
                                    P\rightsquigarrow[bisim] Q; Q' 
Q' \equiv
    proof -
    fix PQ Q'
    assume PQdepth: depth P + depth Q \leqSuc n
    assume validP: valid P and validQ: valid Q
    assume Phnf: uhnf P and Qhnf: uhnf Q
    assume PSimQ: P\rightsquigarrow[bisim] Q
    assume Q'inQ: Q' }\in\mathrm{ summands }
```



```
    proof(nominal-induct Q' avoiding: P rule: pi.strong-inducts)
        case PiNil
        have 0}\in\mathrm{ summands Q by fact
            hence False by(nominal-induct Q rule: pi.strong-inducts, auto simp add:
if-split)
            thus ?case by simp
    next
        case(Output a b Q' P)
        have validP: valid P and Phnf: uhnf P and PSimQ: P\rightsquigarrow[bisim] Q by fact+
        have PQdepth: depth P + depth Q}\leq\mathrm{ Suc n by fact
        have a{b}.\mp@subsup{Q}{}{\prime}\in\mathrm{ summands Q by fact}
        with Qhnf have QTrans: Q\longmapstoa[b]\prec <' by (simp add: summandTransition
uhnf-def)
        with PSimQ obtain P' where PTrans: P\longmapstoa[b]\prec\mp@subsup{P}{}{\prime}\mathrm{ and P'BisimQ': P'}
```

by (blast dest: $\operatorname{sim} E)$
from Phnf PTrans have $a\{b\} . P^{\prime} \in$ summands $P$ by (simp add: summandTransition uhnf-def)
moreover have $P^{\prime} \equiv_{e} Q^{\prime}$
proof -
from validP PTrans have valid $P^{\prime}:$ valid $P^{\prime}$ by (blast intro: validTransition)
from validQ $Q$ Trans have valid $Q^{\prime}$ : valid $Q^{\prime}$ by(blast intro: validTransition)
from valid $P^{\prime}$ obtain $P^{\prime \prime}$ where $P^{\prime \prime} h n f$ : uhnf $P^{\prime \prime}$ and valid $P^{\prime \prime}$ : valid $P^{\prime \prime}$ and $P^{\prime \prime} e q P^{\prime}: P^{\prime \prime} \equiv_{e} P^{\prime}$ and $P^{\prime \prime}$ depth: depth $P^{\prime \prime} \leq$ depth $P^{\prime}$
by (blast dest: validToHnf)
from valid $Q^{\prime}$ obtain $Q^{\prime \prime}$ where $Q^{\prime \prime} h n f:$ uhnf $Q^{\prime \prime}$ and valid $Q^{\prime \prime}$ : valid $Q^{\prime \prime}$ and $Q^{\prime \prime} e q Q^{\prime}: Q^{\prime \prime} \equiv_{e} Q^{\prime}$ and $Q^{\prime \prime}$ depth: depth $Q^{\prime \prime} \leq$ depth $Q^{\prime}$
by (blast dest: validToHnf)
have depth $P^{\prime \prime}+$ depth $Q^{\prime \prime} \leq n$
proof -
from Phnf PTrans have depth $P^{\prime}<$ depth $P$
by (force intro: depthTransition simp add: uhnf-def) moreover from Qhnf QTrans have depth $Q^{\prime}<$ depth $Q$ by (force intro: depthTransition simp add: uhnf-def) ultimately show ?thesis using $P Q$ depth $P^{\prime \prime}$ depth $Q^{\prime \prime}$ depth by simp
qed
moreover have $P^{\prime \prime} \sim Q^{\prime \prime}$
proof -
from $P^{\prime \prime} e q P^{\prime}$ have $P^{\prime \prime} \sim P^{\prime}$ by (rule sound)
moreover from $Q^{\prime \prime} e q Q^{\prime}$ have $Q^{\prime \prime} \sim Q^{\prime}$ by (rule sound)
ultimately show ?thesis using $P^{\prime}$ Bisim $Q^{\prime}$ by (blast dest: transitive symmetric)
qed
ultimately have $P^{\prime \prime} \equiv_{e} Q^{\prime \prime}$ using validP ${ }^{\prime \prime}$ valid $Q^{\prime \prime} P^{\prime \prime} h n f Q^{\prime \prime} h n f$ by (rule-tac IH)
with $P^{\prime \prime} e q P^{\prime} Q^{\prime \prime} e q Q^{\prime}$ show ?thesis by(blast intro: Sym Trans)
qed
ultimately show ?case by (blast intro: Sym equiv.OutputPres)
next
case (Tau $Q^{\prime} P$ )
have validP: valid $P$ and Phnf: uhnf $P$ and $\operatorname{PSimQ}$ : $P \rightsquigarrow[$ bisim $] Q$ by fact + have $P Q d e p t h$ : depth $P+$ depth $Q \leq S u c n$ by fact
have $\tau .\left(Q^{\prime}\right) \in$ summands $Q$ by fact
with Qhnf have $Q$ Trans: $Q \longmapsto \tau \prec Q^{\prime}$ by (simp add: summandTransition uhnf-def)
with $\operatorname{PSimQ}$ obtain $P^{\prime}$ where PTrans: $P \longmapsto \tau \prec P^{\prime}$ and $P^{\prime}$ Bisim $Q^{\prime}: P^{\prime} \sim$

## $Q^{\prime}$

by (blast dest: $\operatorname{sim} E)$
from Phnf PTrans have $\tau .\left(P^{\prime}\right) \in$ summands $P$ by (simp add: summandTransition uhnf-def)
moreover have $P^{\prime} \equiv{ }_{e} Q^{\prime}$
proof -
from validP PTrans have valid $P^{\prime}:$ valid $P^{\prime} \mathbf{b y}$ (blast intro: validTransition)
from validQ QTrans have valid $Q^{\prime}$ : valid $Q^{\prime}$ by (blast intro: validTransition)
from valid $P^{\prime}$ obtain $P^{\prime \prime}$ where $P^{\prime \prime} h n f$ : uhnf $P^{\prime \prime}$ and valid $P^{\prime \prime}$ : valid $P^{\prime \prime}$ and $P^{\prime \prime} e q P^{\prime}: P^{\prime \prime} \equiv_{e} P^{\prime}$ and $P^{\prime \prime}$ depth: depth $P^{\prime \prime} \leq$ depth $P^{\prime}$
by (blast dest: validToHnf)
from validQ' obtain $Q^{\prime \prime}$ where $Q^{\prime \prime} h n f$ : uhnf $Q^{\prime \prime}$ and valid $Q^{\prime \prime}$ : valid $Q^{\prime \prime}$ and $Q^{\prime \prime} e q Q^{\prime}: Q^{\prime \prime} \equiv_{e} Q^{\prime}$ and $Q^{\prime \prime}$ depth: depth $Q^{\prime \prime} \leq$ depth $Q^{\prime}$
by (blast dest: validToHnf)
have depth $P^{\prime \prime}+$ depth $Q^{\prime \prime} \leq n$
proof -
from Phnf PTrans have depth $P^{\prime}<$ depth $P$
by (force intro: depthTransition simp add: uhnf-def) moreover from Qhnf $Q$ Trans have depth $Q^{\prime}<$ depth $Q$ by (force intro: depthTransition simp add: uhnf-def) ultimately show ?thesis using $P Q$ depth $P^{\prime \prime}$ depth $Q^{\prime \prime}$ depth by simp
qed
moreover have $P^{\prime \prime} \sim Q^{\prime \prime}$
proof -
from $P^{\prime \prime} e q P^{\prime}$ have $P^{\prime \prime} \sim P^{\prime}$ by (rule sound)
moreover from $Q^{\prime \prime} e q Q^{\prime}$ have $Q^{\prime \prime} \sim Q^{\prime}$ by (rule sound)
ultimately show ?thesis using $P^{\prime}$ Bisim $Q^{\prime}$ by (blast dest: transitive symmetric)
qed
ultimately have $P^{\prime \prime} \equiv_{e} Q^{\prime \prime}$ using validP ${ }^{\prime \prime}$ valid $Q^{\prime \prime} P^{\prime \prime} h n f Q^{\prime \prime} h n f$ by (rule-tac IH)
with $P^{\prime \prime} e q P^{\prime} Q^{\prime \prime} e q Q^{\prime}$ show ?thesis by (blast intro: Sym Trans)
qed
ultimately show? ?ase by(blast intro: Sym equiv.TauPres)
next
case(Input a $x Q^{\prime} P$ )
have validP: valid $P$ and Phnf: uhnf $P$ and $\operatorname{PSimQ}: P \rightsquigarrow[b i s i m] Q$ and $x$ Fresh $P: x \sharp P$ by fact +
have $P Q$ depth: depth $P+$ depth $Q \leq S u c n$ by fact
have $a<x>. Q^{\prime} \in$ summands $Q$ by fact
with Qhnf have $Q$ Trans: $Q \longmapsto a<x>\prec Q^{\prime}$ by (simp add: summandTransition uhnf-def)
with PSimQ xFreshP obtain $P^{\prime}$ where PTrans: $P \longmapsto a<x>\prec P^{\prime}$ and $P^{\prime}$ der $Q^{\prime}$ : derivative $P^{\prime} Q^{\prime}($ InputS a) x bisim by (blast dest: $\operatorname{sim} E)$
from Phnf PTrans have $a<x>. P^{\prime} \in$ summands $P$ by (simp add: summandTransition uhnf-def)

```
moreover have \(\forall y \in \operatorname{supp}\left(P^{\prime}, Q^{\prime}, x\right) . P^{\prime}[x::=y] \equiv_{e} Q^{\prime}[x::=y]\)
proof (rule ballI)
    fix \(y\) ::name
    assume ysupp: \(y \in \operatorname{supp}\left(P^{\prime}, Q^{\prime}, x\right)\)
    have \(\operatorname{valid} P^{\prime}: \operatorname{valid}\left(P^{\prime}[x::=y]\right)\)
    proof -
    from validP PTrans have validP \({ }^{\prime}\) : valid \(P^{\prime}\) by(blast intro: validTransition)
        thus?thesis by simp
    qed
    have \(\operatorname{valid} Q^{\prime}: \operatorname{valid}\left(Q^{\prime}[x::=y]\right)\)
    proof -
    from valid \(Q\) QTrans have validQ': valid \(Q^{\prime} \mathbf{b y}\) (blast intro: validTransition)
        thus?thesis by simp
    qed
```

    from valid \(P^{\prime}\) obtain \(P^{\prime \prime}\) where \(P^{\prime \prime} h n f\) : uhnf \(P^{\prime \prime}\) and valid \(P^{\prime \prime}\) : valid \(P^{\prime \prime}\)
                and \(P^{\prime \prime} e q P^{\prime}: P^{\prime \prime} \equiv_{e} P^{\prime}[x::=y]\) and \(P^{\prime \prime}\) depth:depth \(P^{\prime \prime}\)
    $\leq \operatorname{depth}\left(P^{\prime}[x::=y]\right)$
by (blast dest: validToHnf)
from valid $Q^{\prime}$ obtain $Q^{\prime \prime}$ where $Q^{\prime \prime} h n f: u h n f Q^{\prime \prime}$ and valid ${ }^{\prime \prime}$ : valid $Q^{\prime \prime}$
and $Q^{\prime \prime} e q Q^{\prime}: Q^{\prime \prime} \equiv_{e} Q^{\prime}[x::=y]$ and $Q^{\prime \prime}$ depth: depth $Q^{\prime \prime}$
$\leq \operatorname{depth}\left(Q^{\prime}[x::=y]\right)$
by (blast dest: validToHnf)
have depth $P^{\prime \prime}+$ depth $Q^{\prime \prime} \leq n$
proof -
from Phnf PTrans have depth $P^{\prime}<$ depth $P$
by (force intro: depthTransition simp add: uhnf-def)
moreover from Qhnf QTrans have depth $Q^{\prime}<$ depth $Q$
by (force intro: depthTransition simp add: uhnf-def)
ultimately show ?thesis using $P Q$ depth $P^{\prime \prime}$ depth $Q^{\prime \prime}$ depth by simp
qed
moreover have $P^{\prime \prime} \sim Q^{\prime \prime}$
proof -
from $P^{\prime} \operatorname{der} Q^{\prime}$ have $P^{\prime} \operatorname{Bisim} Q^{\prime}: P^{\prime}[x::=y] \sim Q^{\prime}[x::=y]$
by (auto simp add: derivative-def)
from $P^{\prime \prime}$ eq $P^{\prime}$ have $P^{\prime \prime} \sim P^{\prime}[x::=y]$ by (rule sound)
moreover from $Q^{\prime \prime} e q Q^{\prime}$ have $Q^{\prime \prime} \sim Q^{\prime}[x::=y]$ by (rule sound)
ultimately show ?thesis using $P^{\prime}$ Bisim $Q^{\prime}$ by (blast dest: transitive
symmetric)
qed
ultimately have $P^{\prime \prime} \equiv_{e} Q^{\prime \prime}$ using validP ${ }^{\prime \prime}$ valid $Q^{\prime \prime} P^{\prime \prime} h n f Q^{\prime \prime} h n f$ by (rule-tac IH)
with $P^{\prime \prime} e q P^{\prime} Q^{\prime \prime} e q Q^{\prime}$ show $P^{\prime}[x::=y] \equiv_{e} Q^{\prime}[x::=y]$ by (blast intro: Sym Trans)
qed
ultimately show ?case
apply -
apply (rule-tac $x=a<x>. P^{\prime}$ in bexI)
apply (rule equiv.InputPres)
apply (rule ballI)
apply (erule-tac $x=y$ in ballE)
apply (blast dest: Sym)
by (auto simp add: supp-prod)
next
case (Match a b $P^{\prime} P$ )
have $[a \frown b] P^{\prime} \in$ summands $Q$ by fact
hence False by(nominal-induct $Q$ rule: pi.strong-inducts, auto simp add: if-split)
thus ?case by simp
next
case(Mismatch a b $P^{\prime} P$ )
have $[a \neq b] P^{\prime} \in$ summands $Q$ by fact
hence False by(nominal-induct $Q$ rule: pi.strong-inducts, auto simp add: if-split)
thus ?case by simp
next
case (Sum $\left.P^{\prime} Q^{\prime} P\right)$
have $P^{\prime} \oplus Q^{\prime} \in$ summands $Q$ by fact
hence False by(nominal-induct $Q$ rule: pi.strong-inducts, auto simp add: if-split)
thus? ?case by simp
next
case (Par $P^{\prime} Q^{\prime} P$ )
have $P^{\prime} \| Q^{\prime} \in$ summands $Q$ by fact
hence False by(nominal-induct $Q$ rule: pi.strong-inducts, auto simp add: if-split)
thus ?case by simp
next
case(Res $x Q^{\prime \prime} P$ )
have $x$ Fresh $P: x \sharp P$ by fact
have validP: valid $P$ and Phnf: uhnf $P$ and $P S i m Q: P \rightsquigarrow[b i s i m] ~ Q$ by fact + have $P Q$ depth: depth $P+$ depth $Q \leq S u c n$ by fact
have $Q^{\prime \prime}$ summ $Q:<\nu x>Q^{\prime \prime} \in$ summands $Q$ by fact
hence $\exists a Q^{\prime} . a \neq x \wedge Q^{\prime \prime}=a\{x\} . Q^{\prime}$
by (nominal-induct $Q$ rule: pi.strong-inducts, auto simp add: if-split pi.inject name-abs-eq name-calc)
then obtain $a Q^{\prime}$ where aineqx: $a \neq x$ and $Q^{\prime} e q Q^{\prime \prime}: Q^{\prime \prime}=a\{x\} \cdot Q^{\prime}$
by blast
with Qhnf $Q^{\prime \prime}$ summ $Q$ have $Q$ Trans: $Q \longmapsto a<\nu x>\prec Q^{\prime}$ by (simp add: summandTransition uhnf-def)
with PSimQ $x$ Fresh $P$ obtain $P^{\prime}$ where PTrans: $P \longmapsto a<\nu x>\prec P^{\prime}$ and $P^{\prime}$ Bisim $Q^{\prime}: P^{\prime} \sim Q^{\prime}$
by (force dest: simE simp add: derivative-def)
from Phnf PTrans aineqx have $\left(<\nu x>a\{x\} . P^{\prime}\right) \in$ summands $P$ by (simp add: summandTransition uhnf-def)
moreover have $a\{x\} \cdot P^{\prime} \equiv_{e} a\{x\} \cdot Q^{\prime}$
proof -
have $P^{\prime} \equiv{ }_{e} Q^{\prime}$
proof -
from validP PTrans have validP': valid $P^{\prime}$ by(blast intro: validTransition)
from validQ $Q$ Trans have valid $Q^{\prime}:$ valid $Q^{\prime}$ by(blast intro: validTransition)
from valid $P^{\prime}$ obtain $P^{\prime \prime}$ where $P^{\prime \prime} h n f$ : uhnf $P^{\prime \prime}$ and valid $P^{\prime \prime}$ : valid $P^{\prime \prime}$ and $P^{\prime \prime} e q P^{\prime}: P^{\prime \prime} \equiv{ }_{e} P^{\prime}$ and $P^{\prime \prime}$ depth: depth $P^{\prime \prime} \leq$ depth $P^{\prime}$
by (blast dest: validToHnf)
from valid $Q^{\prime}$ obtain $Q^{\prime \prime}$ where $Q^{\prime \prime} h n f:$ uhnf $Q^{\prime \prime}$ and valid $Q^{\prime \prime}$ : valid $Q^{\prime \prime}$ and $Q^{\prime \prime} e q Q^{\prime}: Q^{\prime \prime} \equiv_{e} Q^{\prime}$ and $Q^{\prime \prime \prime}$ depth: depth $Q^{\prime \prime} \leq$ depth $Q^{\prime}$
by(blast dest: validToHnf)
have depth $P^{\prime \prime}+$ depth $Q^{\prime \prime} \leq n$
proof -
from Phnf PTrans have depth $P^{\prime}<$ depth $P$ by (force intro: depthTransition simp add: uhnf-def)
moreover from Qhnf QTrans have depth $Q^{\prime}<$ depth $Q$
by (force intro: depthTransition simp add: uhnf-def)
ultimately show ?thesis using $P Q$ depth $P^{\prime \prime}$ depth $Q^{\prime \prime \prime}$ depth by simp qed
moreover have $P^{\prime \prime} \sim Q^{\prime \prime}$
proof -
from $P^{\prime \prime} e q P^{\prime}$ have $P^{\prime \prime} \sim P^{\prime}$ by (rule sound)
moreover from $Q^{\prime \prime} e q Q^{\prime}$ have $Q^{\prime \prime} \sim Q^{\prime}$ by (rule sound)
ultimately show ?thesis using $P^{\prime}$ Bisim $Q^{\prime}$ by (blast dest: transitive symmetric)
qed
ultimately have $P^{\prime \prime} \equiv_{e} Q^{\prime \prime}$ using validP ${ }^{\prime \prime}$ validQ ${ }^{\prime \prime} P^{\prime \prime} h n f Q^{\prime \prime} h n f$ by (rule-tac $I H$ )
with $P^{\prime \prime} e q P^{\prime} Q^{\prime \prime} e q Q^{\prime}$ show ?thesis by (blast intro: Sym Trans)
qed
thus ?thesis by(rule OutputPres)
qed
ultimately show ?case using $Q^{\prime} e q Q^{\prime \prime}$ by (blast intro: Sym equiv.ResPres) next

```
        case(Bang P' P)
        have ! }\mp@subsup{P}{}{\prime}\in\mathrm{ summands Q by fact
            hence False by(nominal-induct Q rule: pi.strong-inducts, auto simp add:
if-split)
            thus ?case by simp
        qed
    qed
    from Phnf Qhnf PQdepth validP validQ PBisimQ show ?case
        apply(rule-tac summandEquiv, auto)
        apply(rule Goal)
        apply auto
        apply(blast dest: bisimE symmetric)
        by(blast intro: Goal dest: bisimE)
qed
lemma complete:
    fixes P :: pi
    and }Q::p
    assumes validP: valid P
    and validQ: valid Q
    and PBisimQ: P ~ Q
    shows P}\equiv\mp@subsup{}{e}{}
proof -
    from validP obtain }\mp@subsup{P}{}{\prime}\mathrm{ where validP': valid }\mp@subsup{P}{}{\prime}\mathrm{ and }\mp@subsup{P}{}{\prime}hnf:uhnf \mp@subsup{P}{}{\prime}\mathrm{ and }\mp@subsup{P}{}{\prime}eqP
P' }\mp@subsup{\equiv}{e}{}
    by(blast dest: validToHnf)
    from validQ obtain }\mp@subsup{Q}{}{\prime}\mathrm{ where validQ': valid Q' and Q'hnf:uhnf Q' and Q'eqQ:
Q' \equiv
    by(blast dest: validToHnf)
    have }\existsn\mathrm{ . depth }\mp@subsup{P}{}{\prime}+\mathrm{ depth }\mp@subsup{Q}{}{\prime}\leqn\mathrm{ by auto
    then obtain n where depth }\mp@subsup{P}{}{\prime}+\mathrm{ depth }\mp@subsup{Q}{}{\prime}\leqn\mathrm{ by blast
    moreover have }\mp@subsup{P}{}{\prime}~\mp@subsup{Q}{}{\prime
    proof -
        from }\mp@subsup{P}{}{\prime}eqP\mathrm{ have }\mp@subsup{P}{}{\prime}~P\mathrm{ by(rule sound)
        moreover from Q'eqQ have }\mp@subsup{Q}{}{\prime}~Q\mathrm{ by(rule sound)
        ultimately show ?thesis using PBisimQ by(blast intro: symmetric transitive)
    qed
    ultimately have }\mp@subsup{P}{}{\prime}\equiv\mp@subsup{\equiv}{e}{}\mp@subsup{Q}{}{\prime}\mathrm{ using validP' validQ' P'hnf Q'hnf by(rule-tac com-
pleteAux)
    with P'eqP Q'eqQ show ?thesis by(blast intro: Sym Trans)
qed
end
```


## References

[1] J. Bengtson. Formalising process calculi, volume 94. Uppsala Dissertations from the Faculty of Science and Technology, 2010.
[2] J. Bengtson and J. Parrow. A completeness proof for bisimulation in the pi-calculus using isabelle. Electr. Notes Theor. Comput. Sci., 192(1):61-75, 2007.
[3] J. Bengtson and J. Parrow. Formalising the pi-calculus using nominal logic. Logical Methods in Computer Science, 5(2), 2009.

