Universita' di Bologna

Facolta' di scienze matematiche fisiche e naturali

Corso di laurea magistrale in Scienze Informatiche

Tesi di laurea

Multi π calcolo

Candidato:
Federico Viscomi

Tutore
Prof. Roberto Gorrieri

. . . .

./pi.jpg

Anno accademico 20011/2012

0.1 Abstract

Il π calcolo e' un formalismo che descrive e analizza le proprieta' del calcolo concorrente. Nasce come proseguio del lavoro gia' svolto sul CCS (Calculus of Communicating Systems). L'aspetto appetibile del π calcolo rispetto ai formalismi precedenti e' l'essere in grado di descrivere la computazione concorrente in sistemi la cui configurazione puo' cambiare nel tempo. Nel CCS e nel π calcolo manca la possibilta' di modellare sequenze atomiche di azioni e di modellare la sincronizzazione multiparte. Il Multi CCS [2] estende il CCS con un'operatore di strong prefixing proprio per colmare tale vuoto. In questa tesi si cerca di trasportare per analogia le soluzioni introdotte dal Multi CCS verso il π calcolo. Il risultato finale e' un linguaggio chiamato Multi π calcolo.

aggiungere una sintesi brevissima dei risultati ottenuti sul Multi π calcolo.

Contents

Chapter 1

Multi ccs

Chapter 2

Π calculus

The π calculus is a mathematical model of processes whose interconnections change as they interact. The basic computational step is the transfer of a communications link between two processes. The idea that the names of the links belong to the same category as the transferred objects is one of the cornerstone of the calculus. The π calculus allows channel names to be communicated along the channels themselves, and in this way it is able to describe concurrent computations whose network configuration may change during the computation.

A coverage of π calculus is on [3], [4] and [5]

2.1 Syntax

We suppose that we have a countable set of names $\mathbb{N}$, ranged over by lower case letters $a, b, \cdots, z$. This names are used for communication channels and values. Furthermore we have a set of identifiers, ranged over by A. We represent the agents or processes by upper case letters $P, Q, \cdots$. A process can perform the following actions:

$$\pi ::= \overline{x}y \mid x(z) \mid \tau$$

The process are defined by the following grammar:

$$P, Q ::= 0 \mid \pi.P \mid P|Q \mid P + Q \mid (\nu x)P \mid A(y_1, \cdots, y_n)$$

and they have the following intuitive meaning:

0 is the empty process, which cannot perform any actions

$\pi.P$ is an action prefixing, this process can perform action π e then behave like P, the action can be:

- $\overline{x}y$ is an output action, this sends the name y along the name x. We can think about x as a channel or a port, and about y as an output datum sent over the channel
- $x(z)$ is an input action, this receives a name along the name x. z is a variable which stores the received data.
- τ is a silent or invisible action, this means that a process can evolve to P without interaction with the environment

$P + Q$ is the sum, this process can enact either P or Q

$P|Q$ is the parallel composition, P and Q can execute concurrently and also syncronize with each other

$(\nu z)P$ is the scope restriction. This process behave as P but the name z is local. This process cannot use the name z to interact with other process but it can for communication within it.

$A(y_1, \cdots, y_n)$ is an identifier whose arity is n. Every identifier has a definition

$$
\begin{array}{ll}
B(0,I) \ = \ \emptyset & B(Q+R,I) \ = \ B(Q,I) \cup B(R,I) \\
B(\overline{x}y.Q,I) \ = \ B(Q,I) & B(Q|R,I) \ = \ B(Q,I) \cup B(R,I) \\
B(x(y).Q,I) \ = \ \{y,\overline{y}\} \cup B(Q,I) & B((\nu x)Q,I) \ = \ \{x,\overline{x}\} \cup B(Q,I) \\
B(\tau.Q,I) \ = \ B(Q,I) &
\end{array}
$$

$$
B(A(x_1,\cdots,x_n),I) = \begin{cases} B(Q, I \cup \{A\}) \ where \ A \overset{def}{=} Q & if \ A \notin I \\ \emptyset & if \ A \in I \end{cases}
$$

Table 2.1: Bound occurrences

$$
\begin{array}{ll}
F(0,I) \ = \ \emptyset & F(Q+R,I) \ = \ F(Q,I) \cup F(R,I) \\
F(\overline{x}y.Q,I) \ = \ \{x,\overline{x},y,\overline{y}\} \cup F(Q,I) & F(Q|R,I) \ = \ F(Q,I) \cup F(R,I) \\
F(x(y).Q,I) \ = \ \{x,\overline{x}\} \cup (F(Q,I) - \{y,\overline{y}\}) & F((\nu x)Q,I) \ = \ F(Q,I) - \{x,\overline{x}\} \\
F(\tau.Q,I) \ = \ F(Q,I) &
\end{array}
$$

$$
F(A(x_1,\cdots,x_n),I) = \begin{cases} F(Q, I \cup \{A\}) \ where \ A \overset{def}{=} Q & if \ A \notin I \\ \emptyset & if \ A \in I \end{cases}
$$

Table 2.2: Free occurrences

$$A(x_1,\cdots,x_n) = P$$

where the x_i must be pairwise disjoint. The intuition is that if the y_i replace the x_i then $A(y_1,\cdots,y_n)$ behave as $P\{y_1/x_1\}\cdots\{y_n/x_n\}$.

To resolve ambiguity we can use parentheses and observe the conventions that prefixing and restriction bind more tightly than composition and prefixing binds more tightly than sum.

Definition 2.1.1. *We say that the input prefix $x(z).P$* binds *z in P or is a* binder *for z in P. We also say that P is the* scope *of the binder and that any occurrence of z in P are* bound *by the binder. Also the restriction operator $(\nu z)P$ is a binder for z in P.*

Definition 2.1.2. *$bn(P)$ is the set of names that have a bound occurrence in P and is defined as $B(P,\emptyset)$, where $B(P,I)$, with I a set of process constants, is defined in table 2.1*

Definition 2.1.3. *We say that a name x is* free *in P if P contains a non bound occurrence of x. We write $fn(P)$ for the set of names with a free occurrence in P. $fn(P)$ is defined as $fn(P,\emptyset)$ where $fn(P,I)$, with I a set of process constants, is defined in table 2.2*

Definition 2.1.4. *$n(P)$ which is the set of all names in P and is defined in the following way:*

$$n(P) \ = \ fn(P) \cup bn(P)$$

In a definition $A(x_1,\cdots,x_n) = P$ we assume that $fn(P) \subseteq \{x_1,\cdots,x_n\}$.

Definition 2.1.5. *$P\{b/a\}$ is the syntactic substitution of name b for a different name a inside a π calculus process, and it consists in replacing every free occurrences of a with b. If b is a bound name in P, in order to avoid name capture we perform an appropriate α conversion. $P\{b/a\}$ is defined in table 2.3*

$$0\{b/a\} = 0$$

$$(\overline{x}y.Q)\{b/a\} = \overline{x}\{b/a\}y\{b/a\}.Q\{b/a\}$$

$$(x(y).Q)\{b/a\} = x\{b/a\}(y).Q\{b/a\} \text{ if } y \neq a \text{ and } y \neq b$$

$$(x(a).Q)\{b/a\} = x\{b/a\}(a).Q$$

$$(x(b).Q)\{b/a\} = x\{b/a\}(c).((Q\{c/b\})\{b/a\}) \text{ where } c \notin n(Q)$$

$$(\tau.Q)\{b/a\} = \tau.Q\{b/a\}$$

$$(A(x_1, \cdots, x_n))\{b/a\} = \begin{cases} A & if\ a \notin fn(A) \\ A_{\{b/a\}}\ where\ A_{\{b/a\}} = Q\{b/a\}\ and\ A \stackrel{def}{=} Q & if\ a \in fn(A) \end{cases}$$

$$(Q + R)\{b/a\} = Q\{b/a\} + R\{b/a\}$$

$$(Q|R)\{b/a\} = Q\{b/a\}|R\{b/a\}$$

$$((\nu y)Q)\{b/a\} = (\nu y)Q\{b/a\} \text{ if } y \neq a \text{ and } y \neq b$$

$$((\nu a)Q)\{b/a\} = (\nu a)Q$$

$$((\nu b)Q)\{b/a\} = (\nu c)((Q\{c/b\})\{b/a\}) \text{ where } c \notin n(Q) \text{ if } a \in fn(Q)$$

$$((\nu b)Q)\{b/a\} = (\nu b)Q \text{ if } a \notin fn(Q)$$

Table 2.3: Syntatic substitution

2.2 Operational Semantic(without structural congruence)

2.2.1 Early operational semantic(without structural congruence)

The semantic of a π calculus process is a labeled transition system such that:

- the nodes are π calculus process. The set of node is $\mathbb{P}$
- the actions can be:
  - unbound input xy
  - unbound output $\overline{x}y$
  - the silent action τ
  - bound output $\overline{x}(y)$

  The set of actions is $\mathbb{A}$, we use α to range over the set of actions.
- the transition relations is $\rightarrow \subseteq \mathbb{P} \times \mathbb{A} \times \mathbb{P}$

In the following section we present the early semantic without structural congruence and without *alpha* conversion. We call this semantic early because in the rule *ECom*

$$\frac{P \xrightarrow{xy} P' \quad Q \xrightarrow{\overline{x}y} Q'}{P|Q \xrightarrow{\tau} P'|Q'}$$

there is no substitution, instead the substitution occurs at an early point in the inference of this translation, namely during the inference of the input action.

Definition 2.2.1. *The* early transition relation $\rightarrow \subseteq \mathbb{P} \times \mathbb{A} \times \mathbb{P}$ *is the smallest relation induced by the rules in table 2.4.*

Example We show now an example of the so called scope extrusion, in particular we prove that

$$a(x).P \mid (\nu b)\overline{a}b.Q \xrightarrow{\tau} (\nu b)(P\{b/x\} \mid Q)$$

where we suppose that $b \notin fn(P)$. In this example the scope of (νb) moves from the right hand component to the left hand.

$$\textsc{CloseR}\ \dfrac{\textsc{EInp}\ \dfrac{}{a(x).P \xrightarrow{ab} P\{b/x\}} \qquad \textsc{Opn}\ \dfrac{\textsc{Out}\ \dfrac{}{\overline{a}b.Q \xrightarrow{\overline{a}b} Q} \qquad a \neq b}{(\nu b)\overline{a}b.Q \xrightarrow{\overline{a}(b)} Q} \qquad b \notin fn((\nu b)\overline{a}b.Q)}{a(x).P \mid (\nu b)\overline{a}b.Q \xrightarrow{\tau} (\nu b)(P\{b/x\} \mid Q)}$$

Example We want to prove now that:

$$((\nu b)a(x).P) \mid \overline{a}b.Q \xrightarrow{\tau} ((\nu c)(P\{c/b\}\{b/x\}))|Q$$

where $b \notin bn(P)$

$$\textsc{ResAlp}\ \dfrac{\textsc{Res}\ \dfrac{\textsc{EInp}\ \dfrac{}{(a(x).P)\{c/b\} \xrightarrow{ab} P\{c/b\}\{b/x\}} \qquad c \notin n(a(b))}{(\nu c)((a(x).P)\{c/b\}) \xrightarrow{ab} (\nu c)(P\{c/b\}\{b/x\})} \qquad b \notin n((a(x).P)\{c/b\})}{(\nu b)a(x).P \xrightarrow{ab} (\nu c)P\{c/b\}\{b/x\}}$$

$$\textsc{EComL}\ \dfrac{(\nu b)a(x).P \xrightarrow{ab} (\nu c)P\{c/b\}\{b/x\} \qquad \textsc{EOut}\ \dfrac{}{\overline{a}b.Q \xrightarrow{\overline{a}b} Q}}{((\nu b)a(x).P) \mid \overline{a}b.Q \xrightarrow{\tau} ((\nu c)(P\{c/b\}\{b/x\}))|Q}$$

$$\textbf{Out}\ \frac{}{\overline{x}y.P \xrightarrow{\overline{x}y} P} \qquad \textbf{EInp}\ \frac{}{x(y).P \xrightarrow{xz} P\{z/y\}}$$

$$\textbf{ParL}\ \frac{P \xrightarrow{\alpha} P' \quad bn(\alpha) \cap fn(Q) = \emptyset}{P|Q \xrightarrow{\alpha} P'|Q} \qquad \textbf{ParR}\ \frac{Q \xrightarrow{\alpha} Q' \quad bn(\alpha) \cap fn(Q) = \emptyset}{P|Q \xrightarrow{\alpha} P|Q'}$$

$$\textbf{SumL}\ \frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'} \qquad \textbf{SumR}\ \frac{Q \xrightarrow{\alpha} Q'}{P + Q \xrightarrow{\alpha} Q'}$$

$$\textbf{Res}\ \frac{P \xrightarrow{\alpha} P' \quad z \notin n(\alpha)}{(\nu z)P \xrightarrow{\alpha} (\nu z)P'} \qquad \textbf{ResAlp}\ \frac{(\nu w)P\{w/z\} \xrightarrow{xz} P' \quad w \notin n(P)}{(\nu z)P \xrightarrow{xz} P'}$$

$$\textbf{EComR}\ \frac{P \xrightarrow{\overline{x}y} P' \quad Q \xrightarrow{xy} Q'}{P|Q \xrightarrow{\tau} P'|Q'} \qquad \textbf{EComL}\ \frac{P \xrightarrow{xy} P' \quad Q \xrightarrow{\overline{x}y} Q'}{P|Q \xrightarrow{\tau} P'|Q'}$$

$$\textbf{ClsL}\ \frac{P \xrightarrow{\overline{x}(z)} P' \quad Q \xrightarrow{xz} Q' \quad z \notin fn(Q)}{P|Q \xrightarrow{\tau} (\nu z)(P'|Q')} \qquad \textbf{ClsR}\ \frac{P \xrightarrow{xz} P' \quad Q \xrightarrow{\overline{x}(z)} Q' \quad z \notin fn(P)}{P|Q \xrightarrow{\tau} (\nu z)(P'|Q')}$$

$$\textbf{Cns}\ \frac{A(\tilde{x}) \overset{def}{=} P \quad P\{\tilde{y}/\tilde{x}\} \xrightarrow{\alpha} P'}{A(\tilde{y}) \xrightarrow{\alpha} P'} \qquad \textbf{Tau}\ \frac{}{\tau.P \xrightarrow{\tau} P}$$

$$\textbf{Opn}\ \frac{P \xrightarrow{\overline{x}z} P' \quad z \neq x}{(\nu z)P \xrightarrow{\overline{x}(z)} P'} \qquad \textsc{OpnAlp}\ \frac{y \notin n(R) \qquad R \xrightarrow{\overline{x}z}_1 R' \qquad z \notin bn(R)}{(\nu y)R\{y/z\} \xrightarrow{\overline{x}(z)}_1 R'}$$

Table 2.4: Early transition relation without structural congruence

$$\textbf{LInp}\ \frac{z \notin fn(P)}{x(y).P \xrightarrow{x(z)} P\{z/y\}} \qquad \textbf{Res}\ \frac{P \xrightarrow{\alpha} P' \quad z \notin n(\alpha)}{(\nu z)P \xrightarrow{\alpha} (\nu z)P'}$$

$$\textbf{SumL}\ \frac{P \xrightarrow{\alpha} P'}{P+Q \xrightarrow{\alpha} P'} \qquad \textbf{SumR}\ \frac{Q \xrightarrow{\alpha} Q'}{P+Q \xrightarrow{\alpha} Q'}$$

$$\textbf{ParL}\ \frac{P \xrightarrow{\alpha} P' \quad bn(\alpha) \cap fn(Q) = \emptyset}{P|Q \xrightarrow{\alpha} P'|Q} \qquad \textbf{ParR}\ \frac{Q \xrightarrow{\alpha} Q' \quad bn(\alpha) \cap fn(Q) = \emptyset}{P|Q \xrightarrow{\alpha} P|Q'}$$

$$\textbf{ComL}\ \frac{P \xrightarrow{x(y)} P' \quad Q \xrightarrow{\overline{x}(z)} Q'}{P|Q \xrightarrow{\tau} P'\{z/y\}|Q'} \qquad \textbf{ComR}\ \frac{P \xrightarrow{\overline{x}(z)} P' \quad Q \xrightarrow{x(y)} Q'}{P|Q \xrightarrow{\tau} P'|Q'\{z/y\}}$$

$$\textbf{Opn}\ \frac{P \xrightarrow{\overline{x}z} P' \quad z \neq x}{(\nu z)P \xrightarrow{\overline{x}(z)} P'} \qquad \textbf{Out}\ \frac{}{\overline{x}y.P \xrightarrow{\overline{x}y} P}$$

$$\textbf{ClsL}\ \frac{P \xrightarrow{\overline{x}(z)} P' \quad Q \xrightarrow{xz} Q' \quad z \notin fn(Q)}{P|Q \xrightarrow{\tau} (\nu z)(P'|Q')} \qquad \textbf{ClsR}\ \frac{P \xrightarrow{xz} P' \quad Q \xrightarrow{\overline{x}(z)} Q' \quad z \notin fn(P)}{P|Q \xrightarrow{\tau} (\nu z)(P'|Q')}$$

$$\textbf{Tau}\ \frac{}{\tau.P \xrightarrow{\tau} P} \qquad \textbf{Cns}\ \frac{A(\tilde{x}) \overset{def}{=} P \quad P\{\tilde{y}/\tilde{x}\} \xrightarrow{\alpha} P'}{A(\tilde{y}) \xrightarrow{\alpha} P'}$$

Table 2.5: Late semantic without structural congruence

2.2.2 Late operational semantic(without structural congruence)

In this case the set of actions $\mathbb{A}$ contains

- bound input $x(y)$
- unbound output $\overline{x}y$
- the silent action τ
- bound output $\overline{x}(y)$

Definition 2.2.2. *The* late transition relation without structural congruence $\rightarrow \subseteq \mathbb{P} \times \mathbb{A} \times \mathbb{P}$ *is the smallest relation induced by the rules in table 2.5.*

2.3 Structural congruence

Structural congruences are a set of equations defining equality and congruence relations on process. They can be used in combination with an SOS semantic for languages. In some cases structural congruences help simplifying the SOS rules: for example they can capture inherent properties of composition operators(e.g. commutativity, associativity and zero element). Also, in process calculi, structural congruences let processes interact even in case they are not adjacent in the syntax. There is a possible trade off between what to include in the structural congruence and what to include in the transition rules: for example in the case of the commutativity of the sum operator. It is worth noticing that in most process calculi every structurally congruent processes should never be distinguished and thus any semantic must assign them the same behaviour.

Definition 2.3.1. *A* change of bound names *in a process P is the replacement of a subterm $x(z).Q$ of P by $x(w).Q\{w/z\}$ or the replacement of a subterm $(\nu z)Q$ of P by $(\nu w)Q\{w/z\}$ where in each case w does not occur in Q.*

Definition 2.3.2. *Processes P and Q are α* convertible *if Q can be obtained from P by a finite number of changes of bound names. If P and Q are α convertible we write $P \equiv_\alpha Q$*

Lemma 2.3.1. inversion lemma *In the proof of equivalence of the semantics in the next section we need this lemma of inversion of the α conversion relation.*

- *If $P \equiv_\alpha Q$ then P and Q have the same operator at the top level.*
- *If $P \equiv_\alpha \tau.Q$ then $P = \tau.P_1$ for some P_1 such that $P_1 \equiv_\alpha Q$*
- *If $P \equiv_\alpha \overline{x}y.Q$ then $P = \overline{x}y.P_1$ for some P_1 such that $P_1 \equiv_\alpha Q$*
- *If $P \equiv_\alpha x(y).Q$ then $P = x(z).P_1$ for some P_1 such that $P_1\{z/y\} \equiv_\alpha Q$*
- *If $P \equiv_\alpha Q_1 + Q_2$ then $P = P_1 + P_2$ for some P_1 and P_2 such that $P_1 \equiv_\alpha Q_1$ and $P_2 \equiv_\alpha Q_2$.*
- *If $P \equiv_\alpha Q_1|Q_2$ then $P = P_1|P_2$ for some P_1 and P_2 such that $P_1 \equiv_\alpha Q_1$ and $P_2 \equiv_\alpha Q_2$.*
- *If $P \equiv_\alpha (\nu y)Q$ then $P = (\nu z)P_1$ such that $P_1\{z/y\} \equiv_\alpha Q$*
- *caso degli identificatori?*

Definition 2.3.3. *A* context *$C[\cdot]$ is a process with a placeholder. If $C[\cdot]$ is a context and we replace the placeholder with P, than we obtain $C[P]$. In doing so, we make no α conversions.*

Definition 2.3.4. *A* congruence *is a binary relation on processes such that:*

- *S is an equivalence relation*
- *S is preserved by substitution in contexts: for each pair of processes (P, Q) and for each context $C[\cdot]$*

$$(P, Q) \in S \;\Rightarrow\; (C[P], C[Q]) \in S$$

Definition 2.3.5. *We define a* structural congruence $\equiv$ *as the smallest congruence on processes that satisfies the axioms in table 2.6*

We can make some clarification on the axioms of the structural congruence:

unfolding this just helps replace an identifier by its definition, with the appropriate parameter instantiation. The alternative is to use an appropriate SOS rule:

$$\textbf{Cns}\ \frac{A(\tilde{x}) \overset{def}{=} P \quad P\{\tilde{y}/\tilde{x}\} \xrightarrow{\alpha} P'}{A(\tilde{y}) \xrightarrow{\alpha} P'}$$

α conversion is the α conversion, i.e., the choice of bound names, it identifies agents like $x(y).\overline{z}y$ and $x(w).\overline{z}w$. In the semantic of pi calculus we can use the structural congruence with the rule SC-ALP or the we can embed the α conversion in the SOS rules. In the early case the rule for input takes care of α conversion, whether in the late case the rule for communication is in charge for α conversion. But this works only for one binder: the input prefix.

abelian monoidal properties of some operators We can deal with associativity and commutativity properties of sum and parallel composition by using SOS rules or by axiom of the structural congruence. For example the commutativity of the sum can be expressed by the following two rules:

$$\textbf{SumL}\ \frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'} \qquad \textbf{SumR}\ \frac{Q \xrightarrow{\alpha} Q'}{P + Q \xrightarrow{\alpha} Q'}$$

or by the following rule and axiom:

$$\textbf{Sum}\ \frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'} \qquad \textbf{SC-SUM} \quad P + Q \equiv Q + P$$

and the rule *Str*

scope extension laws We can use this scope extension laws or the rules *Opn* and *Cls* to deal with the scope extension.

SC-ALP	$\dfrac{P \equiv_\alpha Q}{P \equiv Q}$	α conversion
abelian monoid laws for sum:		
SC-SUM-ASC	$M_1 + (M_2 + M_3) \equiv (M_1 + M_2) + M_3$	associativity
SC-SUM-COM	$M_1 + M_2 \equiv M_2 + M_1$	commutativity
SC-SUM-INC	$M + 0 \equiv M$	zero element
abelian monoid laws for parallel:		
SC-COM-ASC	$P_1 \vert (P_2 \vert P_3) \equiv (P_1 \vert P_2) \vert P_3$	associativity
SC-COM-COM	$P_1 \vert P_2 \equiv P_2 \vert P_1$	commutativity
SC-COM-INC	$P \vert 0 \equiv P$	zero element
scope extension laws:		
SC-RES	$(\nu z)(\nu w)P \equiv (\nu w)(\nu z)P$	
SC-RES-INC	$(\nu z)0 \equiv 0$	
SC-RES-COM	$(\nu z)(P_1 \vert P_2) \equiv P_1 \vert (\nu z)P_2$ if $z \notin fn(P_1)$	
SC-RES-SUM	$(\nu z)(P_1 + P_2) \equiv P_1 + (\nu z)P_2$ if $z \notin fn(P_1)$	
unfolding law:		
SC-IDE	$A(\tilde{y}) \equiv P\{\tilde{y}/\tilde{x}\}$	if $A(\tilde{x}) \stackrel{def}{=} P$

Table 2.6: Structural congruence axioms

2.4 Operational semantic with structural congruence

2.4.1 Early semantic with α conversion only

In this subsection we introduce the early operational semantic for π calculus with the use of a minimal structural congruence, specifically we exploit only the easy of α conversion.

Definition 2.4.1. *The* early transition relation with α conversion $\rightarrow \subseteq \mathbb{P} \times \mathbb{A} \times \mathbb{P}$ *is the smallest relation induced by the rules in table 2.7.*

2.4.2 Early semantic with structural congruence

Definition 2.4.2. *The* early transition relation with structural congruence $\rightarrow \subseteq \mathbb{P} \times \mathbb{A} \times \mathbb{P}$ *is the smallest relation induced by the rules in table 2.8.*

Example We prove now that

$$a(x).P \mid (\nu b)\overline{a}b.Q \xrightarrow{\tau} (\nu b)(P\{b/x\} \mid Q)$$

where $b \notin fn(P)$. This follows from

$$a(x).P \mid (\nu b)\overline{a}b.Q \equiv (\nu b)(a(x).P \mid \overline{a}b.Q)$$

and

$$(\nu b)(a(x).P \mid \overline{a}b.Q) \xrightarrow{\tau} (\nu b)(P\{b/x\} \mid Q)$$

with the rule *Str*. We can prove the last transition in the following way:

$$\textsc{Res}\ \dfrac{\textsc{Com}\ \dfrac{\textsc{EInp}\ \dfrac{}{a(x).P \xrightarrow{ab} P\{b/x\}} \qquad \textsc{Out}\ \dfrac{}{\overline{a}b.Q \xrightarrow{\overline{a}b} Q}}{a(x).P \mid \overline{a}b.Q \xrightarrow{\tau} P\{b/x\} \mid Q}}{(\nu b)(a(x).P \mid \overline{a}b.Q) \xrightarrow{\tau} (\nu b)(P\{b/x\} \mid Q)}$$

$$\textbf{Out}\ \frac{}{\overline{x}y.P \xrightarrow{\overline{x}y} P} \qquad \textbf{EInp}\ \frac{}{x(y).P \xrightarrow{xz} P\{z/y\}}$$

$$\textbf{ParL}\ \frac{P \xrightarrow{\alpha} P^{'} \quad bn(\alpha) \cap fn(Q) = \emptyset}{P|Q \xrightarrow{\alpha} P^{'}|Q} \qquad \textbf{ParR}\ \frac{Q \xrightarrow{\alpha} Q^{'} \quad bn(\alpha) \cap fn(Q) = \emptyset}{P|Q \xrightarrow{\alpha} P|Q^{'}}$$

$$\textbf{SumL}\ \frac{P \xrightarrow{\alpha} P^{'}}{P + Q \xrightarrow{\alpha} P^{'}} \qquad \textbf{SumR}\ \frac{Q \xrightarrow{\alpha} Q^{'}}{P + Q \xrightarrow{\alpha} Q^{'}}$$

$$\textbf{Res}\ \frac{P \xrightarrow{\alpha} P^{'} \quad z \notin n(\alpha)}{(\nu z)P \xrightarrow{\alpha} (\nu z)P^{'}} \qquad \textbf{Alp}\ \frac{P \equiv_{\alpha} Q \ P \xrightarrow{\alpha} P^{'} \ P^{'} \equiv_{\alpha} Q^{'}}{Q \xrightarrow{\alpha} Q^{'}}$$

$$\textbf{EComL}\ \frac{P \xrightarrow{xy} P^{'} \quad Q \xrightarrow{\overline{x}y} Q^{'}}{P|Q \xrightarrow{\tau} P^{'}|Q^{'}} \qquad \textbf{EComR}\ \frac{P \xrightarrow{\overline{x}y} P^{'} \quad Q \xrightarrow{xy} Q^{'}}{P|Q \xrightarrow{\tau} P^{'}|Q^{'}}$$

$$\textbf{ClsL}\ \frac{P \xrightarrow{\overline{x}(z)} P^{'} \quad Q \xrightarrow{xz} Q^{'} \quad z \notin fn(Q)}{P|Q \xrightarrow{\tau} (\nu z)(P^{'}|Q^{'})} \qquad \textbf{ClsR}\ \frac{P \xrightarrow{xz} P^{'} \quad Q \xrightarrow{\overline{x}(z)} Q^{'} \quad z \notin fn(P)}{P|Q \xrightarrow{\tau} (\nu z)(P^{'}|Q^{'})}$$

$$\textbf{Cns}\ \frac{A(\tilde{x}) \overset{def}{=} P \ P\{\tilde{y}/\tilde{x}\} \xrightarrow{\alpha} P^{'}}{A(\tilde{y}) \xrightarrow{\alpha} P^{'}} \qquad \textbf{Opn}\ \frac{P \xrightarrow{\overline{x}z} P^{'} \quad z \neq x}{(\nu z)P \xrightarrow{\overline{x}(z)} P^{'}}$$

$$\textbf{Tau}\ \frac{}{\tau.P \xrightarrow{\tau} P}$$

Table 2.7: Early transition relation with α conversion

$$\textbf{Out}\ \frac{}{\overline{x}y.P \xrightarrow{\overline{x}y} P} \qquad \textbf{EInp}\ \frac{}{x(z).P \xrightarrow{xy} P\{y/z\}} \qquad \textbf{Par}\ \frac{P \xrightarrow{\alpha} P^{'} \quad bn(\alpha) \cap fn(Q) = \emptyset}{P|Q \xrightarrow{\alpha} P^{'}|Q}$$

$$\textbf{Sum}\ \frac{P \xrightarrow{\alpha} P^{'}}{P + Q \xrightarrow{\alpha} P^{'}} \qquad \textbf{ECom}\ \frac{P \xrightarrow{xy} P^{'} \quad Q \xrightarrow{\overline{x}y} Q^{'}}{P|Q \xrightarrow{\tau} P^{'}|Q^{'}} \qquad \textbf{Res}\ \frac{P \xrightarrow{\alpha} P^{'} \quad z \notin n(\alpha)}{(\nu z)P \xrightarrow{\alpha} (\nu z)P^{'}}$$

$$\textbf{Tau}\ \frac{}{\tau.P \xrightarrow{\tau} P} \qquad \textbf{Opn}\ \frac{P \xrightarrow{\overline{x}z} P^{'} \quad z \neq x}{(\nu z)P \xrightarrow{\overline{x}(z)} P^{'}} \qquad \textbf{Str}\ \frac{P \equiv P^{'} \quad P \xrightarrow{\alpha} Q \quad Q \equiv Q^{'}}{P^{'} \xrightarrow{\alpha} Q^{'}}$$

Table 2.8: Early semantic with structural congruence

$$\textbf{Prf}\ \frac{}{\alpha.P \xrightarrow{\alpha} P} \qquad \textbf{Sum}\ \frac{P \xrightarrow{\alpha} P'}{P+Q \xrightarrow{\alpha} P'}$$

$$\textbf{Par}\ \frac{P \xrightarrow{\alpha} P' \quad bn(\alpha) \cap fn(Q) = \emptyset}{P|Q \xrightarrow{\alpha} P'|Q} \qquad \textbf{Res}\ \frac{P \xrightarrow{\alpha} P' \quad z \notin n(\alpha)}{(\nu z)P \xrightarrow{\alpha} (\nu z)P'}$$

$$\textbf{LCom}\ \frac{P \xrightarrow{x(y)} P' \quad Q \xrightarrow{\overline{x}z} Q'}{P|Q \xrightarrow{\tau} P'\{z/y\}|Q'} \qquad \textbf{Str}\ \frac{P \equiv P' \quad P \xrightarrow{\alpha} Q \quad Q \equiv Q'}{P' \xrightarrow{\alpha} Q'}$$

$$\textbf{Opn}\ \frac{P \xrightarrow{\overline{x}z} P' \quad z \neq x}{(\nu z)P \xrightarrow{\overline{x}(z)} P'}$$

Table 2.9: Late semantic with structural congruence

Example We want to prove now that:

$$((\nu b)a(x).P) \mid \overline{a}b.Q \ \xrightarrow{\tau}\ (\nu c)(P\{c/b\}\{b/x\} \mid Q)$$

where the name c is not in the free names of Q. We can exploit the structural congruence and get that

$$((\nu b)a(x).P)|\overline{a}b.Q \ \equiv\ (\nu c)(a(x).(P\{c/b\})|\overline{a}b.Q)$$

then we have

$$\textsc{Res}\ \dfrac{\textsc{Com}\ \dfrac{\textsc{EInp}\ \dfrac{}{a(x).P\{c/b\} \ \xrightarrow{ab}\ P\{c/b\}\{b/x\}} \qquad \textsc{Out}\ \dfrac{}{\overline{a}b.Q \ \xrightarrow{\overline{a}b}\ Q}}{(a(x).(P\{c/b\})|\overline{a}b.Q) \ \xrightarrow{\tau}\ (P\{c/b\}\{b/x\}|Q)}}{(\nu c)(a(x).(P\{c/b\})|\overline{a}b.Q) \ \xrightarrow{\tau}\ (\nu c)(P\{c/b\}\{b/x\}|Q)}$$

Now we just apply the rule *Str* to prove the thesis.

2.4.3 Late semantic with structural congruence

Definition 2.4.3. *The* late transition relation with structural congruence $\rightarrow \subseteq \mathbb{P} \times \mathbb{A} \times \mathbb{P}$ *is the smallest relation induced by the rules in table 2.9.*

Example We prove now that

$$a(x).P \mid (\nu b)\overline{a}b.Q \ \xrightarrow{\tau} P\{b/x\} \mid Q$$

where $b \notin fn(P)$. This follows from

$$a(x).P \mid (\nu b)\overline{a}b.Q \ \equiv\ (\nu b)(a(x).P \mid \overline{a}b.Q)$$

and

$$(\nu b)(a(x).P \mid \overline{a}b.Q) \xrightarrow{\tau} (\nu b)(P\{b/x\} \mid Q)$$

with the rule *Str*. We can prove the last transition in the following way:

$$\textsc{Res}\ \dfrac{\textsc{LCom}\ \dfrac{\textsc{LInp}\ \dfrac{b \notin fn(P)}{a(x).P \ \xrightarrow{ab}\ P\{b/x\}} \qquad \textsc{Out}\ \dfrac{}{\overline{a}b.Q \ \xrightarrow{\overline{a}b}\ Q}}{a(x).P \mid \overline{a}b.Q \ \xrightarrow{\tau}\ P\{b/x\} \mid Q} \qquad b \notin n(\tau)}{(\nu b)(a(x).P \mid \overline{a}b.Q) \ \xrightarrow{\tau}\ (\nu b)(P\{b/x\} \mid Q)}$$

Example We want to prove now that:

$$((\nu b)a(x).P) \mid \overline{a}b.Q \xrightarrow{\tau} (\nu c)(P\{c/b\}\{b/x\} \mid Q)$$

where the name c is not in the free names of Q and is not in the names of P. We can exploit the structural congruence and get that

$$((\nu b)a(x).P)|\overline{a}b.Q \equiv (\nu c)(a(x).(P\{c/b\})|\overline{a}b.Q)$$

then we have

$$\textsc{Res}\ \dfrac{\textsc{LCom}\ \dfrac{\textsc{LInp}\ \dfrac{b \notin fn(P\{c/b\})}{a(x).P\{c/b\} \xrightarrow{ab} P\{c/b\}\{b/x\}} \qquad \textsc{Out}\ \dfrac{}{\overline{a}b.Q \xrightarrow{\overline{a}b} Q}}{(a(x).(P\{c/b\})|\overline{a}b.Q) \xrightarrow{\tau} (P\{c/b\}\{b/x\}|Q)} \qquad c \notin n(\tau)}{(\nu c)(a(x).(P\{c/b\})|\overline{a}b.Q) \xrightarrow{\tau} (\nu c)(P\{c/b\}\{b/x\}|Q)}$$

Now we just apply the rule *Str* to prove the thesis.

2.5 Equivalence of the semantics

2.5.1 Equivalence of the early semantics

Here we write $\rightarrow_1$ for the early semantic without structural congruence, $\rightarrow_2$ for the early semantic with just α conversion and $\rightarrow_3$ for the early semantic with the full structural congruence.

Theorem 2.5.1. $P \xrightarrow{\alpha}_1 P' \Leftrightarrow P \xrightarrow{\alpha}_2 P'$

Proof. TODO: dimostrare che tutte le derivazioni di una qualsiasi transizione sono finite(se e' vero)! The proof is by induction on the length of the derivation of a transaction, and then both the base case and the inductive case proceed by cases on the last rule used in the derivation. We call R_1 the set of rules for $\rightarrow_1$ and R_2 the set of rules for $\rightarrow_2$.

$\Leftarrow$: The cases in which the last rule is in $R_1 \cap R_2$ are easy: for the base cases a derivation of $P \xrightarrow{\alpha}_2 P'$ is also a derivation of $P \xrightarrow{\alpha}_1 P'$ because it uses only the prefix rules; for the inductive cases, let R be the last rule used in the derivation of $P \xrightarrow{\alpha}_2 P'$, we apply the inductive hypothesis to the premises of R and then apply R. Things are more complicated if R is *Alp*. We show that we can move toward the top level of the derivation tree any occurrence of the rule *Alp* until we can replace it by an instance of *ResAlp*, a prefix rule of *OpnAlp*. So suppose we have at the end of the derivation tree of $P \xrightarrow{\alpha}_2 P'$ the following:

$$\textsc{Alp}\ \dfrac{P \equiv_\alpha S \qquad S \xrightarrow{\alpha}_2 S' \qquad P' \equiv S'}{P \xrightarrow{\alpha}_2 P'}$$

now we proceed by cases on the last rule of the derivation of $S \xrightarrow{\alpha}_2 S'$:

Out

$$\textsc{Alp}\ \dfrac{\overline{x}y.P_1 \equiv_\alpha \overline{x}y.S_1 \qquad \textsc{Out}\ \dfrac{}{\overline{x}y.S_1 \xrightarrow{\overline{x}y}_2 S_1} \qquad S_1 \equiv_\alpha P_1}{\overline{x}y.P_1 \xrightarrow{\overline{x}y}_2 P_1}$$

became

$$\textsc{Out}\ \dfrac{}{\overline{x}y.P_1 \xrightarrow{\overline{x}y}_2 P_1}$$

Tau

$$\textsc{Alp}\ \dfrac{\tau.P_1 \equiv_\alpha \tau.S_1 \qquad \textsc{Tau}\ \dfrac{}{\tau.S_1 \xrightarrow{\tau}_2 S_1} \qquad S_1 \equiv_\alpha P_1}{\tau.P_1 \xrightarrow{\tau}_2 P_1}$$

became

$$\textsc{Tau}\ \dfrac{}{\tau.P_1 \xrightarrow{\tau}_2 P_1}$$

EInp from the hypothesis $x(y).P_1 \equiv_\alpha x(z).S_1$ and the inversion lemma we get $P_1\{z/y\} \equiv_\alpha S_1$, it follows that $S_1\{w/z\} \equiv_\alpha (P_1\{z/y\})\{w/z\}$. We also have

$$\textsc{EInp}\ \frac{}{x(z).S_1 \xrightarrow{xw}_2 S_1\{w/z\}}$$

$$\textsc{Alp}\ \frac{\textsc{EInp}\ \dfrac{x(y).P_1 \equiv_\alpha x(z).S_1}{x(z).S_1 \xrightarrow{xw}_2 S_1\{w/z\}} \qquad S_1\{w/z\} \equiv_\alpha (P_1\{z/y\})\{w/z\}}{x(y).P_1 \xrightarrow{xw}_2 (P_1\{z/y\})\{w/z\}}$$

became

$$\textsc{Alp}\ \frac{}{x(y).P_1 \xrightarrow{xw}_2 P_1\{w/y\}}$$

SumL in such case S is $S_1 + S_2$ for some S_1 and S_2. From $P \equiv_\alpha S_1 + S_2$ we get that P is $P_1 + P_2$ for some P_1 and P_2 such that $S_1 \equiv_\alpha P_1$ and $S_2 \equiv_\alpha P_2$. We also now that $S^{'}$ is an $S_1^{'}$ such that $S_1 \xrightarrow{\alpha}_2 S_1^{'}$ and $S_1^{'} \equiv_\alpha S_1$. To sum up the last part of the derivation tree of $P \xrightarrow{\alpha}_2 P^{'}$ is:

$$\textsc{Alp}\ \frac{P_1 + P_2 \equiv_\alpha S_1 + S_2 \qquad \textsc{SumL}\ \dfrac{S_1 \xrightarrow{\alpha}_2 S_1^{'}}{S_1 + S_2 \xrightarrow{\alpha}_2 S_1^{'}} \qquad S_1^{'} \equiv_\alpha P_1^{'}}{P_1 + P_2 \xrightarrow{\alpha}_2 P_1^{'}}$$

we move upward the rule *Alp* like this:

$$\textsc{SumL}\ \frac{\textsc{Alp}\ \dfrac{P_1 \equiv_\alpha S_1 \qquad S_1 \xrightarrow{\alpha}_2 S_1^{'} \qquad S_1^{'} \equiv_\alpha P_1^{'}}{P_1 \xrightarrow{\alpha}_2 P_1^{'}}}{P_1 + P_2 \xrightarrow{\alpha}_2 P_1^{'}}$$

Now we can apply the inductive hypothesis to $P_1 \xrightarrow{\alpha}_2 P_1^{'}$ and get $P_1 \xrightarrow{\alpha}_1 P_1^{'}$. After this we apply the rule *SumL* and get $P_1 + P_2 \xrightarrow{\alpha}_1 P_1^{'}$. The case for *SumR* is simmetric and so omitted.

ParL

$$\textsc{Alp}\ \frac{P_1|P_2 \equiv_\alpha S_1|S_2 \qquad \textsc{ParL}\ \dfrac{S_1 \xrightarrow{\alpha}_2 S_1^{'} \qquad bn(\alpha) \cap fn(S_2) = \emptyset}{S_1|S_2 \xrightarrow{\alpha}_2 S_1^{'}|S_2} \qquad S_1^{'} \equiv_\alpha P_1^{'}}{P_1|P_2 \xrightarrow{\alpha}_2 P_1^{'}|P_2}$$

this became

$$\textsc{ParL}\ \frac{\textsc{Alp}\ \dfrac{P_1 \equiv_\alpha S_1 \qquad S_1 \xrightarrow{\alpha}_2 S_1^{'} \qquad S_1^{'} \equiv_\alpha P_1^{'}}{P_1 \xrightarrow{\alpha}_2 P_1^{'}} \qquad bn(\alpha) \cap fn(P_2) = \emptyset}{P_1|P_2 \xrightarrow{\alpha}_2 P_1^{'}|P_2}$$

Res

$$\textsc{Alp}\ \frac{(\nu z)R \equiv_\alpha (\nu y)S \qquad \textsc{Res}\ \dfrac{S \xrightarrow{\alpha}_2 S^{'} \qquad z \notin n(\alpha)}{(\nu y)S \xrightarrow{\alpha}_2 (\nu y)S^{'}} \qquad (\nu z)R^{'} \equiv_\alpha (\nu y)S^{'}}{(\nu z)R \xrightarrow{\alpha}_2 (\nu z)R^{'}}$$

this became:

$$\textsc{Res}\ \frac{\textsc{Alp}\ \dfrac{R \equiv_\alpha S \qquad S \xrightarrow{\alpha}_2 S^{'} \qquad R^{'} \equiv_\alpha S^{'}}{R \xrightarrow{\alpha}_2 R^{'}} \qquad z \notin n(\alpha)}{(\nu z)R \xrightarrow{\alpha}_2 (\nu z)R^{'}}$$

Alp

$$\textsc{Alp}\ \dfrac{P \equiv_\alpha S \qquad \textsc{Alp}\ \dfrac{S \equiv_\alpha T \qquad T \xrightarrow{\alpha}_2 T' \qquad S' \equiv_\alpha T'}{S \xrightarrow{\alpha}_2 S'} \qquad P' \equiv_\alpha S'}{P \xrightarrow{\alpha}_2 P'}$$

because of the transitivity of α equivalence we can replace the previous part of the tree by the following:

$$\textsc{Alp}\ \dfrac{P \equiv_\alpha T \qquad T \xrightarrow{\alpha}_2 T' \qquad P' \equiv_\alpha T'}{P \xrightarrow{\alpha}_2 P'}$$

so we can assume the tree has no two adjacent application of the rule Alp

EComL

$$\textsc{Alp}\ \dfrac{P_1|P_2 \equiv_\alpha S_1|S_2 \qquad \textsc{EComL}\ \dfrac{S_1 \xrightarrow{\overline{x}y}_2 S_1' \qquad S_2 \xrightarrow{xy}_2 S_2'}{S_1|S_2 \xrightarrow{\tau}_2 S_1'|S_2'} \qquad P_1'|P_2' \equiv_\alpha S_1'|S_2'}{P_1|P_2 \xrightarrow{\tau}_2 P_1'|P_2'}$$

this became

$$\textsc{EComL}\ \dfrac{\textsc{Alp}\ \dfrac{P_1 \equiv_\alpha S_1 \qquad S_1 \xrightarrow{\overline{x}y}_2 S_1' \qquad P_1' \equiv_\alpha S_1'}{P_1 \xrightarrow{\overline{x}y}_2 P_1'} \qquad \textsc{Alp}\ \dfrac{P_2 \equiv_\alpha S_2 \qquad S_2 \xrightarrow{xy}_2 S_2' \qquad P_2' \equiv_\alpha S_2'}{P_2 \xrightarrow{xy}_2 P_2'}}{P_1|P_2 \xrightarrow{\tau}_2 P_1'|P_2'}$$

The case for $EComR$ is simmetric and so omitted.

ClsL

$$\textsc{Alp}\ \dfrac{P_1|P_2 \equiv_\alpha S_1|S_2 \qquad \textsc{ClsL}\ \dfrac{S_1 \xrightarrow{\overline{x}(z)} S_1' \qquad S_2 \xrightarrow{xz} S_2'}{S_1|S_2 \xrightarrow{\tau}_2 (\nu z)(S_1'|S_2')} \qquad (\nu z)(P_1'|P_2') \equiv_\alpha (\nu z)(S_1'|S_2')}{P_1|P_2 \xrightarrow{\tau}_2 (\nu z)(P_1'|P_2')}$$

where $z \notin fn(S_2)$, this became:

$$\textsc{ClsL}\ \dfrac{\textsc{Alp}\ \dfrac{P_1 \equiv_\alpha S_1 \qquad S_1 \xrightarrow{\overline{x}(z)} S_1' \qquad P_1' \equiv_\alpha S_1'}{P_1 \xrightarrow{\overline{x}(z)} P_1'} \qquad \textsc{Alp}\ \dfrac{P_2 \equiv_\alpha S_2 \qquad S_2 \xrightarrow{xz} S_2' \qquad P_2' \equiv_\alpha S_2'}{P_2 \xrightarrow{xz} P_2'}}{P_1|P_2 \xrightarrow{\tau}_2 (\nu z)(P_1'|P_2')}$$

where $z \notin fn(P_2)$ because $S_2 \equiv_\alpha P_2$ and $z \notin fn(S_2)$ and the α conversion does not change any free name.

Cns

$$\textsc{Alp}\ \dfrac{P \equiv_\alpha A(\tilde{y}) \qquad \textsc{Cns}\ \dfrac{A(\tilde{x}) \overset{def}{=} S \qquad S\{\tilde{y}/\tilde{x}\} \xrightarrow{\alpha} S'}{A(\tilde{y}) \xrightarrow{\alpha} S'} \qquad S' \equiv_\alpha P'}{P \xrightarrow{\alpha} P'}$$

Given $P \equiv_\alpha A(\tilde{y})$ and $A(\tilde{x}) = S$ we create another identifier $B(\tilde{y}) = P$ such that $A(\tilde{y}) \equiv_\alpha B(\tilde{y})$ and replace the previous derivation with:

$$\textsc{Cns}\ \dfrac{B(\tilde{y}) \overset{def}{=} P \qquad \textsc{Alp}\ \dfrac{B(\tilde{y}) \equiv_\alpha S\{\tilde{y}/\tilde{x}\} \qquad S\{\tilde{y}/\tilde{x}\} \xrightarrow{\alpha} S' \qquad S' \equiv_\alpha P'}{B(\tilde{y}) \xrightarrow{\alpha} P'}}{P \xrightarrow{\alpha} P'}$$

Opn

$$\textsc{Alp}\ \frac{(\nu z)S \equiv_\alpha (\nu z)R \qquad \textsc{Opn}\ \dfrac{R \xrightarrow{\overline{x}z} R' \qquad z \neq x}{(\nu z)R \xrightarrow{\overline{x}(z)} R'} \qquad R' \equiv_\alpha P'}{(\nu z)S \xrightarrow{\overline{x}(z)} P'}$$

became

$$\textsc{Opn}\ \frac{\textsc{Alp}\ \dfrac{S \equiv_\alpha R \qquad R \xrightarrow{\overline{x}z} R' \qquad R' \equiv_\alpha P'}{S \xrightarrow{\overline{x}z} P'} \qquad z \neq x}{(\nu z)S \xrightarrow{\overline{x}(z)} P'}$$

Now we wonder if we can have the following case:

$$\textsc{Alp}\ \frac{(\nu y)S \equiv_\alpha (\nu z)R \qquad \textsc{Opn}\ \dfrac{R \xrightarrow{\overline{x}z} R' \qquad z \neq x}{(\nu z)R \xrightarrow{\overline{x}(z)} R'} \qquad R' \equiv_\alpha P'}{(\nu y)S \xrightarrow{\overline{x}(z)} P'}$$

The answer is yes, for example

$$\textsc{Alp}\ \frac{(\nu y)\overline{x}y.0 \equiv_\alpha (\nu z)\overline{x}z.0 \qquad \textsc{Opn}\ \dfrac{\textsc{Out}\ \dfrac{}{\overline{x}z.0 \xrightarrow{\overline{x}z}_2 0} \qquad z \neq x}{(\nu z)\overline{x}z.0 \xrightarrow{\overline{x}(z)}_2 0}}{(\nu y)\overline{x}y.0 \xrightarrow{\overline{x}(z)}_2 0}$$

There is no way to prove $(\nu y)\overline{x}y.0 \xrightarrow{\overline{x}(z)}_1 0$ because the head of this derivation has a restriction at the top level so the only rules applicable are:

Res this rule does not work because the restriction in the body of the derivation is missing

ResAlp this rule dose not work because it requires an input action as label for the transition

Opn this rule implies $\overline{x}y.0 \xrightarrow{\overline{x}z}_1 0$ which cannot be proved

The solution is adding the following rule:

$$\textsc{OpnAlp}\ \frac{y \notin n(R) \qquad R \xrightarrow{\overline{x}z}_1 R' \qquad z \notin bn(R)}{(\nu y)R\{y/z\} \xrightarrow{\overline{x}(z)}_1 R'}$$

now the derivation became:

$$\textsc{OpnAlp}\ \frac{y \notin n(S\{z/y\}) \qquad \textsc{Alp}\ \dfrac{S\{z/y\} \equiv_\alpha R \qquad R \xrightarrow{\overline{x}z} R' \qquad P' \equiv_\alpha R'}{S\{z/y\} \xrightarrow{\overline{x}z} P'} \qquad z \notin bn(S)}{(\nu y)S \xrightarrow{\overline{x}(z)} P'}$$

$\Rightarrow$: R_2 contains the same rules as R_1 but *Alp* is in R_2 only, whether *ResAlp* and *OpnAlp* are in R_1 only. We can mimic the rule *ResAlp* using the rule *Alp*, so if at some point in a derivation tree of $P \xrightarrow{\alpha}_1 P'$ we use the rule *ResAlp*, we can replace it with an appropriate instance of the following rule:

$$\frac{(\nu z)P \equiv_\alpha (\nu w)P\{w/z\} \qquad w \notin n(P) \qquad (\nu w)P\{w/z\} \xrightarrow{xz} P' \qquad P' \equiv_\alpha P'}{(\nu z)P \xrightarrow{xz} P'}$$

which is in turn a particular case of the rule *Alp*. We can mimic the rule *OpnAlp* in the following way:

$$\textsc{Alp}\ \dfrac{(\nu y)R\{y/z\} \equiv_\alpha (\nu z)R \qquad \textsc{Opn}\ \dfrac{z \neq x \qquad R \xrightarrow{\overline{x}z} R'}{(\nu z)R \xrightarrow{\overline{x}(z)} R'}}{(\nu y)R\{y/z\} \xrightarrow{\overline{x}(z)} R'}$$

□

Theorem 2.5.2. $P \xrightarrow{\alpha}_2 P' \Leftrightarrow \exists P'' : P' \equiv P''$ *and* $P \xrightarrow{\alpha}_3 P''$

2.5.2 Equivalence of the late semantics

2.6 Bisimilarity and Congruence

We present here some behavioural equivalences and some of their properties.

2.6.1 Bisimilarity

In the following we will use the phrase $bn(\alpha)$ is fresh in a definition to mean that the name in $bn(\alpha)$, if any, is different from any free name occurring in any of the agents in the definition. In this subsection the transition relation is given by the late semantic with structural congruence.

Definition 2.6.1. *A* strong (late) bisimulation *is a symmetric binary relation* $\mathbb{R}$ *on agents satisfying the following:* $P\mathbb{R}Q$ *and* $P \xrightarrow{\alpha}_E P'$ *where* $bn(\alpha)$ *is fresh implies that*

- *if* $\alpha = a(x)$ *then* $\exists Q' : Q \xrightarrow{a(x)} Q' \wedge \forall u :\ P'\{u/x\}\mathbb{R}Q'\{u/x\}$
- *if* α *is not an input the* $\exists Q' : Q \xrightarrow{\alpha} Q' \wedge P'\mathbb{R}Q'$

P and Q are strongly bisimilar, written $P \dot\sim Q$*, if they are related by a bisimulation.*

The union of all bisimulation $\dot\sim$ is a bisimulation. If two process are structurally congruent then because of the rule *Str* they are also strong bisimilar.

Example Two strongly bisimilar processes are the following:

$$a(x).0|\bar{b}x.0 \dot\sim a(x).\bar{b}x.0 + \bar{b}x.a(x).0$$

and the bisimulation(without showing the simmetric part) is the following:

$$\{(a(x).0|\bar{b}x.0, a(x).\bar{b}x.0 + \bar{b}x.a(x).0), (a(x).0|0, a(x).0), (0|0, 0|0)\} \cup \{(0|\bar{b}x.0, \bar{b}x.0)|x \in \mathbb{N}\}$$

If we apply the substitution $\{a/b\}$ to each process then they are not strongly bisimilar anymore because $(a(x).0|\bar{b}x.0)\{a/b\}$ is $a(x).0|\bar{a}x.0$ and this process can perform an invisible action whether $(a(x).\bar{b}x.0 + \bar{b}x.a(x).0)\{a/b\}$ cannot. This shows that strong bisimulation is not closed under substitution.

Proposition 2.6.1. *If* $P \dot\sim Q$ *and* σ *is injective then* $P\sigma \dot\sim Q\sigma$

Proposition 2.6.2. $\dot\sim$ *is an equivalence*

Proposition 2.6.3. $\dot\sim$ *is preserved by all operators except input prefix*

2.6.2 Congruence

Definition 2.6.2. *We say that two agents P and Q are* strongly congruent*, written* $P \sim Q$ *if*

$$P\sigma \dot\sim Q\sigma \text{ for all substitution } \sigma$$

Proposition 2.6.4. *Strong congruence is the largest congruence in bisimilarity.*

2.6.3 Variants of Bisimilarity

We define a bisimulation for the early semantic with structural congruence, for clarity when referring to the early semantic we index the transition with $_E$.

Definition 2.6.3. *A* strong early bisimulation with early semantic *is a symmetric binary relation* $\mathbb{R}$ *on agents satisfying the following:* $P\mathbb{R}Q$ *and* $P \xrightarrow{\alpha}_E P^{'}$ *where* $bn(\alpha)$ *is fresh implies that*

$$\exists Q^{'} : Q \xrightarrow{\alpha} Q^{'} \ \wedge \ P^{'}\mathbb{R}Q^{'}$$

P and Q are strongly early bisimilar, written $P \dot{\sim}_E Q$*, if they are related by an early bisimulation.*

Definition 2.6.4. *A* strong early bisimulation with late semantic *is a symmetric binary relation* $\mathbb{R}$ *on agents satisfying the following:* $P\mathbb{R}Q$ *and* $P \xrightarrow{\alpha} P^{'}$ *where* $bn(\alpha)$ *is fresh implies that*

- *if* $\alpha = a(x)$ *then* $\forall u \ \exists Q^{'} : \ Q \xrightarrow{a(x)} Q^{'} \ \wedge \ P^{'}\{u/x\}\mathbb{R}Q^{'}\{u/x\}$
- *if* α *is not an input then* $\exists Q^{'} : \ Q \xrightarrow{\alpha} Q^{'} \ \wedge \ P^{'}\mathbb{R}Q^{'}$

Proposition 2.6.5. *Early bisimilarity is preserved by all operators except input prefix.*

Definition 2.6.5. *The* early congruence $\sim_E$ *is defined by*

$$P \sim_E Q \ \textit{if} \ \forall \sigma \ P\sigma \dot{\sim}_E Q\sigma$$

where σ *is a substitution.*

Proposition 2.6.6. *The early congruence is the largest congruence in* $\dot{\sim}_E$.

In the following definition we consider a subcalculus without restriction.

Definition 2.6.6. *A* strong open bisimulation *is a symmetric binary relation* $\mathbb{R}$ *on agents satisfying the following for all substitutions* σ*:* $P\mathbb{R}Q$ *and* $P\sigma \xrightarrow{\alpha} P^{'}$ *where* $bn(\alpha)$ *is fresh implies that*

$$\exists Q^{'} : \ Q\sigma \xrightarrow{\alpha} Q^{'} \ \wedge \ P^{'}\mathbb{R}Q^{'}$$

P and Q are strongly open bisimilar, written $P \dot{\sim}_O Q$ *if they are related by an open bisimulation.*

Proposition 2.6.7. *strong open bisimulation is also a late bisimulation, is closed under substitution, is an equivalence and a congruence*

Chapter 3

Multi π calculus solo output

3.1 Syntax

As we did whit π calculus, we suppose that we have a countable set of names $\mathbb{N}$, ranged over by lower case letters $a, b, \cdots, z$. This names are used for communication channels and values. Furthermore we have a set of identifiers, ranged over by A. We represent the agents or processes by upper case letters $P, Q, \cdots$. A multi π process, in addiction to the same actions of a π process, can perform also a strong prefix output:

$$\pi ::= \overline{x}y \mid x(z) \mid \underline{\overline{x}y} \mid \tau$$

The process are defined, just as original π calculus, by the following grammar:

$$P, Q ::= 0 \mid \pi.P \mid P|Q \mid P + Q \mid (\nu x)P \mid A(y_1, \cdots, y_n)$$

and they have the same intuitive meaning as for the π calculus. The strong prefix output allows a process to make an atomic sequence of actions, so that more than one process can synchronize on this sequence. For the moment we allow the strong prefix to be on output names only. Also one can use the strong prefix only as an action prefixing for processes that can make at least a further action. Since the strong prefix can be on output names only, the only synchronization possible is between a process that executes a sequence of n actions(only the last action can be an input) with $n \geq 1$ and n other processes each executing one single action(at least $n-1$ process execute an output and at most one executes an input).

Multi π calculus is a conservative extension of the π calculus in the sense that: any π calculus process p is also a multi π calculus process and the semantic of p according to the SOS rules of π calculus is the same as the semantic of p according to the SOS rules of multi π calculus.

We have to extend the following definition to deal with the strong prefix:

$$B(\underline{\overline{x}y}.Q, I) \ = \ B(Q, I) \quad F(\underline{\overline{x}y}.Q, I) \ = \ \{x, \overline{x}, y, \overline{y}\} \cup F(Q, I)$$

3.2 Operational semantic

3.2.1 Early operational semantic with structural congruence

The semantic of a multi π process is labeled transition system such that

- the nodes are multi π calculus process. The set of node is $\mathbb{P}_m$
- the actions are multi π calculus actions. The set of actions is $\mathbb{A}_m$, we use $\alpha, \alpha_1, \alpha_2, \cdots$ to range over the set of actions, we use $\sigma, \sigma_1, \sigma_2, \cdots$ to range over the set $\mathbb{A}_m^+ \cup \{\tau\}$. Note that σ is a non empty sequence of actions.
- the transition relations is $\rightarrow \subseteq \mathbb{P}_m \times (\mathbb{A}_m^+ \cup \{\tau\}) \times \mathbb{P}_m$

Out $\dfrac{}{\overline{x}y.P \xrightarrow{\overline{x}y} P}$	**EInp** $\dfrac{}{x(y).P \xrightarrow{xz} P\{z/y\}}$
Tau $\dfrac{}{\tau.P \xrightarrow{\tau} P}$	**SOut** $\dfrac{P \xrightarrow{\sigma} P' \quad \sigma \neq \tau}{\underline{\overline{x}y}.P \xrightarrow{\overline{x}y\cdot\sigma} P'}$
Sum $\dfrac{P \xrightarrow{\sigma} P'}{P+Q \xrightarrow{\sigma} P'}$	**Str** $\dfrac{P \equiv P' \qquad P' \xrightarrow{\alpha} Q' \qquad Q \equiv Q'}{P \xrightarrow{\alpha} Q}$
Par $\dfrac{P \xrightarrow{\sigma} P' \quad bn(\sigma) \cap fn(Q) = \emptyset}{P\vert Q \xrightarrow{\sigma} P'\vert Q}$	**EComSng** $\dfrac{P \xrightarrow{xy} P' \qquad Q \xrightarrow{\overline{x}y} Q'}{P\vert Q \xrightarrow{\tau} P'\vert Q'}$
Res $\dfrac{P \xrightarrow{\sigma} P' \quad z \notin n(\alpha)}{(\nu)zP \xrightarrow{\sigma} (\nu)zP'}$	**EComSeq** $\dfrac{P \xrightarrow{xy} P' \qquad Q \xrightarrow{\overline{x}y\cdot\sigma} Q'}{P\vert Q \xrightarrow{\sigma} P'\vert Q'}$

Table 3.1: Multi π early semantic with structural congruence

In this case, a label can be a sequence of prefixes, whether in the original π calculus a label can be only a prefix. We use the symbol $\cdot$ to denote the concatenation operator.

Definition 3.2.1. *The* early transition relation without structural congruence *is the smallest relation induced by the rules in table 3.1*

In the following examples we omit sometimes the rule Str.

Example We show an example of a derivation of three processes that synchronize.

Res $(\nu x)((\underline{\overline{x}y}.\overline{x}y.0|x(y).0)|x(y).0) \xrightarrow{\tau} (\nu x)((0|0)|0)$

$\quad x \notin n(\tau)$

$\quad$ **EComSng** $((\underline{\overline{x}y}.\overline{x}y.0|x(y).0)|x(y).0) \xrightarrow{\tau} ((0|0)|0)$

$\qquad$ **EComSeq** $\underline{\overline{x}y}.\overline{x}y.0|x(y).0 \xrightarrow{\overline{x}y} 0|0$

$\qquad\quad$ **EInp** $x(y).0 \xrightarrow{xy} 0$

$\qquad\quad$ **SOut** $\underline{\overline{x}y}.\overline{x}y.0 \xrightarrow{\overline{x}y\cdot\overline{x}y} 0$

$\qquad\qquad \overline{x}y \neq \tau$

$\qquad\qquad$ **Out** $\overline{x}y.0 \xrightarrow{\overline{x}y} 0$

$\qquad$ **Out** $x(y).0 \xrightarrow{xy} 0$

Example We want to prove that

$$(\underline{\overline{a}x}.c(x).0|b(x).0)|(a(x).0|\underline{\overline{b}x}.\overline{c}x.0) \xrightarrow{\tau} (0|0)|(0|0)$$

Str $(\underline{\overline{a}x}.c(x).0|b(x).0)|(a(x).0|\underline{\overline{b}x}.\overline{c}x.0) \xrightarrow{\tau} (0|0)|(0|0)$

$\quad$ **EComSng** $(\underline{\overline{a}x}.c(x).0|a(x).0)|(b(x).0|\underline{\overline{b}x}.\overline{c}x.0) \xrightarrow{\tau} (0|0)|(0|0)$

$\qquad$ **EComSeq** $b(x).0|\underline{\overline{b}x}.\overline{c}x.0 \xrightarrow{\overline{c}x} 0|0$

$\qquad\quad$ **EInp** $b(x).0 \xrightarrow{bx} 0$

$\qquad\quad$ **SOut** $\underline{\overline{b}x}.\overline{c}x.0 \xrightarrow{\overline{b}x\cdot\overline{c}x} 0$

$\qquad\qquad$ **Out** $\overline{c}x.0 \xrightarrow{\overline{c}x} 0$

$\qquad$ **EComSeq** $\underline{\overline{a}x}.c(x).0|a(x).0 \xrightarrow{cx} 0|0$

$$\textbf{SOut}\ \ \underline{\overline{a}x}.c(x).0 \xrightarrow{\overline{a}x\cdot cx} 0$$
$$\textbf{Inp}\ \ c(x).0 \xrightarrow{cx} 0$$
$$\textbf{Inp}\ \ a(x).0 \xrightarrow{ax} 0$$
$$(\underline{\overline{a}x}.c(x).0|b(x).0)|(a(x).0|\underline{\overline{b}x}.\overline{c}x.0) \equiv (\underline{\overline{a}x}.c(x).0|a(x).0)|(b(x).0|\underline{\overline{b}x}.\overline{c}x.0)$$

Example The *dining philosophers* problem, originally proposed by Dijkstra in [1], is defined in the following way: Five silent philosophers sit at a round table. There is one fork between each pair of adjacent philosophers. Each philosopher must alternately think and eat. However, a philosopher can only eat while holding both the fork to the left and the fork to the right. Each philosopher can pick up an adjacent fork, when available, and put it down, when holding it. The problem is to design an algorithm such that no philosopher will starve, i.e. can forever continue to alternate between eating and thinking. We present one solution which uses only two forks and two philosophers:

- we define two constants for the forks:

$$fork_1 \stackrel{def}{=} up_1(x).dn_1(x).fork_1 \quad fork_0 \stackrel{def}{=} up_0(x).dn_0(x).fork_0$$

  the input name x is not important and can be anything else.

- we define two constants for the philosophers:

$$\begin{aligned} phil_1 &\stackrel{def}{=} think(x).phil_1 + \underline{\overline{up_1}x}.\overline{up_0}(x).eat(x).\underline{\overline{dn_1}x}.dn_0(x).phil_1 \\ phil_0 &\stackrel{def}{=} think(x).phil_0 + \underline{\overline{up_0}x}.\overline{up_1}(x).eat(x).\underline{\overline{dn_0}x}.dn_1(x).phil_0 \end{aligned}$$

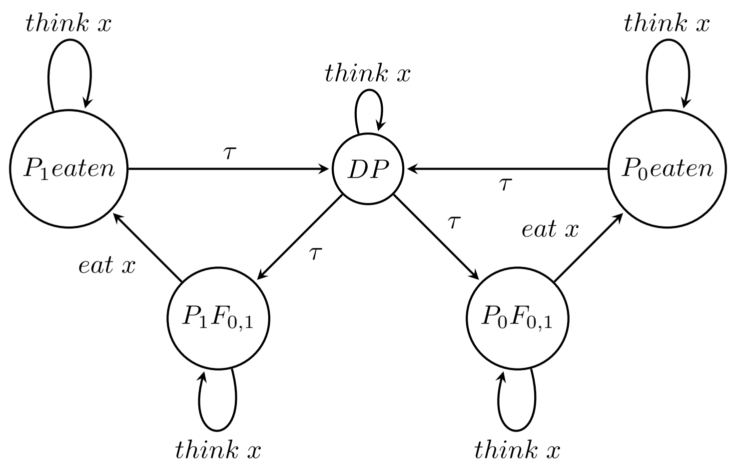
  also in this case the name x is not relevant.

- the following definition describe the whole system with philosophers and forks:

$$DP \stackrel{def}{=} (\nu\{up_0, up_1, down_0, down_1\})(phil_0|phil_1|fork_0|fork_1)$$

  where with $(\nu\{up_0, up_1, down_0, down_1\})$ we mean $(\nu\ up_0)(\nu\ up_1)(\nu\ down_0)(\nu\ down_1)$

- the operational semantic of DP is the following lts:



Now we need to prove every transition in the semantic of DP. Let $L = \{up_0, up_1, down_0, down_1\}$ we start with $DP \xrightarrow{\tau} DP$:

Example We want to show now an example of synchronization between four processes:

Res $(\nu\ a)((((\underline{\overline{a}x}.\underline{\overline{a}x}.\overline{a}x.0|a(x).0)|a(x).0)|a(x).0) \xrightarrow{\tau} (\nu\ a)(((0|0)|0)|0))$

$$\textbf{Pref}\ \frac{\alpha\ not\ a\ strong\ prefix}{\alpha.P \xrightarrow{\alpha} P} \qquad \textbf{Par}\ \frac{P \xrightarrow{\sigma} P^{'} \qquad bn(\sigma)\cap fn(Q)=\emptyset}{P|Q \xrightarrow{\sigma} P^{'}|Q}$$

$$\textbf{SOut}\ \frac{P \xrightarrow{\sigma} P^{'} \qquad \sigma \neq \tau}{\underline{\overline{x}y}.P \xrightarrow{\overline{x}y\cdot\sigma} P^{'}} \qquad \textbf{LComSeq}\ \frac{P \xrightarrow{x(y)} P^{'} \qquad Q \xrightarrow{\overline{x}z\cdot\sigma} Q^{'} \qquad z \notin fn(P)}{P|Q \xrightarrow{\sigma} P^{'}\{z/y\}|Q^{'}}$$

$$\textbf{Sum}\ \frac{P \xrightarrow{\sigma} P^{'}}{P+Q \xrightarrow{\sigma} P^{'}} \qquad \textbf{Str}\ \frac{P \equiv P^{'} \qquad P^{'} \xrightarrow{\alpha} Q^{'} \qquad Q \equiv Q^{'}}{P \xrightarrow{\alpha} Q}$$

$$\textsc{Res}\ \frac{P \xrightarrow{\sigma} P^{'} \quad z \notin n(\alpha)}{(\nu)zP \xrightarrow{\sigma} (\nu)zP^{'}} \qquad \textbf{LComSng}\ \frac{P \xrightarrow{x(y)} P^{'} \qquad Q \xrightarrow{\overline{x}z} Q^{'} \qquad z \notin fn(P)}{P|Q \xrightarrow{\tau} P^{'}\{z/y\}|Q^{'}}$$

Table 3.2: Multiπ late semantic with structural congruence

$a \notin n(\tau)$

EComSng $(((\underline{\bar{a}x}.\underline{\bar{a}x}.\bar{a}x.0|a(x).0)|a(x).0)|a(x).0) \xrightarrow{\tau} ((0|0)|0)|0)$

EComSeq $(\underline{\bar{a}x}.\underline{\bar{a}x}.\bar{a}x.0|a(x).0)|a(x).0 \xrightarrow{\bar{a}x} (0|0)|0$

EComSeq $\underline{\bar{a}x}.\underline{\bar{a}x}.\bar{a}x.0|a(x).0) \xrightarrow{\bar{a}x\cdot\bar{a}x} 0|0$

SOut $\underline{\bar{a}x}.\underline{\bar{a}x}.\bar{a}x.0 \xrightarrow{\bar{a}x\cdot\bar{a}x\cdot\bar{a}x} 0$

SOut $\underline{\bar{a}x}.\bar{a}x.0 \xrightarrow{\bar{a}x\cdot\bar{a}x} 0$

SOut $\underline{\bar{a}x}.\bar{a}x.0 \xrightarrow{\bar{a}x\cdot\bar{a}x} 0$

. **Out** $\bar{a}x.0 \xrightarrow{\bar{a}x} 0$

Inp $a(x).0 \xrightarrow{ax} 0$

Inp $a(x).0 \xrightarrow{ax} 0$

Inp $a(x).0 \xrightarrow{ax} 0$

3.2.2 Late operational semantic with structural congruence

Definition 3.2.2. *The* late transition relation with structural congruence *is the smallest relation induced by the rules in table 3.2.*

Chapter 4

Multi π calculus solo input

4.1 Syntax

As we did whit multi π calculus, we suppose that we have a countable set of names $\mathbb{N}$, ranged over by lower case letters $a, b, \cdots, z$. This names are used for communication channels and values. Furthermore we have a set of identifiers, ranged over by A. We represent the agents or processes by upper case letters $P, Q, \cdots$. A multi π process, in addiction to the same actions of a π process, can perform also a strong prefix input:

$$\pi ::= \overline{x}y \mid x(z) \mid \underline{x(y)} \mid \tau$$

The process are defined, just as original π calculus, by the following grammar:

$$P, Q ::= 0 \mid \pi.P \mid P|Q \mid P + Q \mid (\nu x)P \mid A(y_1, \cdots, y_n)$$

and they have the same intuitive meaning as for the π calculus. The strong prefix input allows a process to make an atomic sequence of actions, so that more than one process can synchronize on this sequence. For the moment we allow the strong prefix to be on input names only. Also one can use the strong prefix only as an action prefixing for processes that can make at least a further action. Since the strong prefix can be on input names only, the only synchronization possible is between a process that executes a sequence of n actions(only the last action can be an output) with $n \geq 1$ and n other processes each executing one single action(at least $n-1$ process execute an output and at most one executes an input).

Multi π calculus is a conservative extension of the π calculus in the sense that: any π calculus process p is also a multi π calculus process and the semantic of p according to the SOS rules of π calculus is the same as the semantic of p according to the SOS rules of multi π calculus. We have to extend the following definition to deal with the strong prefix:

$$B(\underline{x(y)}.Q, I) = \{y, \overline{y}\} \cup B(Q, I) \qquad F(\underline{x(y)}.Q, I) = \{x, \overline{x}\} \cup (F(Q, I) - \{y, \overline{y}\})$$

4.2 Operational semantic

4.2.1 Early operational semantic with structural congruence

The semantic of a multi π process is labeled transition system such that

- the nodes are multi π calculus process. The set of node is $\mathbb{P}_m$
- the actions are multi π calculus actions. The set of actions is $\mathbb{A}_m$, we use $\alpha, \alpha_1, \alpha_2, \cdots$ to range over the set of actions, we use $\sigma, \sigma_1, \sigma_2, \cdots$ to range over the set $\mathbb{A}_m^+ \cup \{\tau\}$.
- the transition relations is $\rightarrow \subseteq \mathbb{P}_m \times (\mathbb{A}_m^+ \cup \{\tau\}) \times \mathbb{P}_m$

In this case, a label can be a sequence of prefixes, whether in the original π calculus a label can be only a prefix. We use the symbol $\cdot$ to denote the concatenation operator.

$$\textbf{Out}\ \frac{}{\overline{x}y.P \xrightarrow{\overline{x}y} P} \qquad \textbf{EInp}\ \frac{}{x(y).P \xrightarrow{xz} P\{z/y\}}$$

$$\textbf{Tau}\ \frac{}{\tau.P \xrightarrow{\tau} P} \qquad \textbf{SInp}\ \frac{P \xrightarrow{\sigma} P^{'} \qquad \sigma \neq \tau}{\underline{x(y)}.P \xrightarrow{xz \cdot \sigma} P^{'}\{z/y\}}$$

$$\textbf{Sum}\ \frac{P \xrightarrow{\sigma} P^{'}}{P + Q \xrightarrow{\sigma} P^{'}} \qquad \textbf{Str}\ \frac{P \equiv P^{'} \qquad P^{'} \xrightarrow{\alpha} Q^{'} \qquad Q \equiv Q^{'}}{P \xrightarrow{\alpha} Q}$$

$$\textbf{Par}\ \frac{P \xrightarrow{\sigma} P^{'} \qquad bn(\sigma) \cap fn(Q) = \emptyset}{P|Q \xrightarrow{\sigma} P^{'}|Q} \qquad \textbf{Com}\ \frac{P \xrightarrow{\sigma_1} P^{'} \qquad Q \xrightarrow{\sigma_2} Q^{'} \qquad Sync(\sigma_1, \sigma_2, \sigma_3)}{P|Q \xrightarrow{\sigma_3} P^{'}|Q^{'}}$$

$$\textbf{Res}\ \frac{P \xrightarrow{\sigma} P^{'} \qquad z \notin n(\alpha)}{(\nu)zP \xrightarrow{\sigma} (\nu)zP^{'}}$$

Table 4.1: Multi π early semantic with structural congruence

Definition 4.2.1. *The* early transition relation with structural congruence *is the smallest relation induced by the rules in table 4.1.*

Definition 4.2.2. *We define the synchronization relation in the following way:*

$$\textsc{Com1L}\ \frac{}{Sync(\overline{x}y, xy, \tau)} \qquad \textsc{Com2L}\ \frac{}{Sync(xy \cdot \sigma, \overline{x}y, \sigma)}$$

$$\textsc{Com1R}\ \frac{}{Sync(xy, \overline{x}y, \tau)} \qquad \textsc{Com2R}\ \frac{}{Sync(\overline{x}y, xy \cdot \sigma, \sigma)}$$

This does not work because:

$$\textsc{SInp}\ \frac{\textsc{Out}\ \frac{}{\overline{a}z.P \xrightarrow{\overline{a}z} P}}{\underline{x(a)}.\overline{a}z.P \xrightarrow{xb \cdot \overline{a}z} P\{b/a\}}$$

whether the semantic for $\underline{x(a)}.\overline{a}z.P$ is supposed to be

$$\underline{x(a)}.\overline{a}z.P \xrightarrow{xb \cdot \overline{b}z} P\{b/a\}$$

4.2.2 Late operational semantic with structural congruence

Definition 4.2.3. *The* late transition relation with structural congruence *is the smallest relation induced by the rules in table 4.2.*

$$\textbf{Pref}\ \frac{\alpha\ not\ a\ strong\ prefix}{\alpha.P\ \xrightarrow{\alpha}\ P} \qquad \textbf{LComSeq}\ \frac{P\ \xrightarrow{x(y)\cdot\sigma} P' \qquad Q\ \xrightarrow{\overline{x}z} Q' \qquad z \notin fn(\sigma) \cup fn(P)}{P|Q\ \xrightarrow{\sigma\{z/y\}} P'\{z/y\}|Q'}$$

$$\textbf{SInp}\ \frac{P\ \xrightarrow{\sigma}\ P' \qquad \sigma \neq \tau}{\underline{x(y)}.P\ \xrightarrow{x(y)\cdot\sigma} P'} \qquad \textbf{LComSng}\ \frac{P\ \xrightarrow{x(y)} P' \qquad Q\ \xrightarrow{\overline{x}z} Q' \qquad z \notin fn(P)}{P|Q\ \xrightarrow{\tau} P'\{z/y\}|Q'}$$

$$\textbf{Sum}\ \frac{P\ \xrightarrow{\sigma}\ P'}{P+Q\ \xrightarrow{\sigma} P'} \qquad \textbf{Str}\ \frac{P \equiv P' \qquad P'\ \xrightarrow{\alpha}\ Q' \qquad Q \equiv Q'}{P\ \xrightarrow{\alpha}\ Q}$$

$$\textbf{Res}\ \frac{P\ \xrightarrow{\sigma}\ P' \qquad z \notin n(\alpha)}{(\nu)zP\ \xrightarrow{\sigma} (\nu)zP'} \qquad \textbf{Par}\ \frac{P\ \xrightarrow{\sigma}\ P' \qquad bn(\sigma) \cup fn(Q) = \emptyset}{P|Q\ \xrightarrow{\sigma} P'|Q}$$

Table 4.2: Multi π late semantic with structural congruence

Chapter 5

Multi π calculus input e output

5.1 Syntax

As we did whit multi π calculus, we suppose that we have a countable set of names $\mathbb{N}$, ranged over by lower case letters $a, b, \cdots, z$. This names are used for communication channels and values. Furthermore we have a set of identifiers, ranged over by A. We represent the agents or processes by upper case letters $P, Q, \cdots$. A multi π process, in addiction to the same actions of a π process, can perform also a strong prefix:

$$\pi ::= \overline{x}y \mid x(z) \mid \underline{x(y)} \mid \underline{\overline{x}y} \mid \tau$$

The process are defined, just as original π calculus, by the following grammar:

$$P, Q ::= 0 \mid \pi.P \mid P|Q \mid P + Q \mid (\nu x)P \mid A(y_1, \cdots, y_n)$$

and they have the same intuitive meaning as for the π calculus. The strong prefix input allows a process to make an atomic sequence of actions, so that more than one process can synchronize on this sequence.

We have to extend the following definition to deal with the strong prefix:

$$\begin{array}{llll} B(\underline{x(y)}.Q, I) = \{y, \overline{y}\} \cup B(Q, I) & & F(\underline{x(y)}.Q, I) = \{x, \overline{x}\} \cup (F(Q, I) - \{y, \overline{y}\}) \\ B(\underline{\overline{x}y}.Q, I) = B(Q, I) & & F(\underline{\overline{x}y}.Q, I) = \{x, \overline{x}, y, \overline{y}\} \cup F(Q, I) \end{array}$$

5.2 Operational semantic

5.2.1 Early operational semantic with structural congruence

5.2.2 Late operational semantic with structural congruence

The semantic of a multi π process is labeled transition system such that

- the nodes are multi π calculus process. The set of node is $\mathbb{P}_m$
- The set of actions is $\mathbb{A}_m$ and can contain
  - bound output $\overline{x}(y)$
  - unbound output $\overline{x}y$
  - bound input $x(z)$

  We use $\alpha, \alpha_1, \alpha_2, \cdots$ to range over the set of actions, we use $\sigma, \sigma_1, \sigma_2, \cdots$ to range over the set $\mathbb{A}_m^+ \cup \{\tau\}$.
- the transition relations is $\rightarrow \subseteq \mathbb{P}_m \times (\mathbb{A}_m^+ \cup \{\tau\}) \times \mathbb{P}_m$

In this case, a label can be a sequence of prefixes, whether in the original π calculus a label can be only a prefix. We use the symbol $\cdot$ to denote the concatenation operator.

Definition 5.2.1. *The* late transition relation with structural congruence *is the smallest relation induced by the rules in table 5.1*

$$\textbf{Pref}\ \frac{\alpha\ not\ a\ strong\ prefix}{\alpha.P \xrightarrow{\alpha} P} \qquad \textbf{Par}\ \frac{P \xrightarrow{\sigma} P' \quad bn(\sigma)\cap fn(Q)=\emptyset}{P|Q \xrightarrow{\sigma} P'|Q}$$

$$\textbf{SOut}\ \frac{P \xrightarrow{\sigma} P' \quad \sigma\neq\tau}{\underline{\overline{x}y}.P \xrightarrow{\overline{x}y\cdot\sigma} P'} \qquad \textbf{LComSeq1}\ \frac{P \xrightarrow{x(y)} P' \qquad Q \xrightarrow{\overline{x}z\cdot\sigma} Q' \qquad z\notin fn(P)}{P|Q \xrightarrow{\sigma} P'\{z/y\}|Q'}$$

$$\textbf{Sum}\ \frac{P \xrightarrow{\sigma} P'}{P+Q \xrightarrow{\sigma} P'} \qquad \textbf{Str}\ \frac{P\equiv P' \qquad P' \xrightarrow{\alpha} Q' \qquad Q\equiv Q'}{P \xrightarrow{\alpha} Q}$$

$$\textbf{Res}\ \frac{P \xrightarrow{\sigma} P' \quad z\notin n(\alpha)}{(\nu z)P \xrightarrow{\sigma} (\nu z)P'} \qquad \textbf{LComSng}\ \frac{P \xrightarrow{x(y)} P' \qquad Q \xrightarrow{\overline{x}z} Q' \qquad z\notin fn(P)}{P|Q \xrightarrow{\tau} P'\{z/y\}|Q'}$$

$$\textbf{SInp}\ \frac{P \xrightarrow{\sigma} P' \qquad \sigma\neq\tau}{\underline{x(y)}.P \xrightarrow{x(y)\cdot\sigma} P'} \qquad \textbf{LComSeq2}\ \frac{P \xrightarrow{\overline{x}z} P' \qquad Q \xrightarrow{x(y)\cdot\sigma} Q' \qquad z\notin fn(P)}{P|Q \xrightarrow{\sigma\{z/y\}} P'|Q'\{z/y\}}$$

Table 5.1: Multi π late semantic with structural congruence

5.2.3 Another attemp to late operational semantic with structural congruence

Definition 5.2.2. *The* late transition relation with structural congruence *is the smallest relation induced by the rules in table 5.2:*

In what follows, the names $\delta, \delta_1, \delta_2$ represents substitutions, they can also be empty; the names $\sigma, \sigma_1, \sigma_2, \sigma_3$ are non empty sequences of actions. The relation *Sync* is defined by the axioms in table 5.3

Example We want to prove that:

$$\underline{\overline{a}x}.\overline{a}y.P|\underline{a(w)}.a(z).Q \xrightarrow{\tau} P|Q\{x/w\}\{y/z\}$$

We start first noticing that

$$\text{S4R}\ \frac{\text{S1R}\ \overline{Sync(\overline{a}y, a(z)\{x/w\}, \tau, \{\}, \{y/z\})}}{Sync(\overline{a}x\cdot\overline{a}y, a(w)\cdot a(z), \tau, \{\}, \{x/w\}\{y/z\})}$$

and that

$$\text{SOut}\ \frac{\text{Pref}\ \overline{\overline{a}y.P \xrightarrow{\overline{a}y} P}}{\underline{\overline{a}x}.\overline{a}y.P \xrightarrow{\overline{a}x\cdot\overline{a}y} P} \qquad \text{SInp}\ \frac{\text{Pref}\ \overline{a(z).Q \xrightarrow{a(z)} Q}}{\underline{a(w)}.a(z).Q \xrightarrow{a(w)\cdot a(z)} Q}$$

and in the end we just need to apply the rule **LCom**

$$\textbf{Pref}\ \frac{\alpha \ not\ a\ strong\ prefix}{\alpha.P \xrightarrow{\alpha} P} \qquad \textbf{Par}\ \frac{P \xrightarrow{\sigma} P' \quad bn(\sigma) \cap fn(Q) = \emptyset}{P|Q \xrightarrow{\sigma} P'|Q}$$

$$\textbf{SOut}\ \frac{P \xrightarrow{\sigma} P' \quad \sigma \neq \tau}{\underline{\overline{x}y}.P \xrightarrow{\overline{x}y \cdot \sigma} P'} \qquad \textbf{LCom}\ \frac{P \xrightarrow{\sigma_1} P' \qquad Q \xrightarrow{\sigma_2} Q' \qquad Sync(\sigma_1, \sigma_2, \sigma_3, \delta_1, \delta_2)}{P|Q \xrightarrow{\sigma_3} P'\delta_1|Q'\delta_2}$$

$$\textbf{Sum}\ \frac{P \xrightarrow{\sigma} P'}{P + Q \xrightarrow{\sigma} P'} \qquad \textbf{Str}\ \frac{P \equiv P' \qquad P' \xrightarrow{\alpha} Q' \qquad Q \equiv Q'}{P \xrightarrow{\alpha} Q}$$

$$\textbf{Res}\ \frac{P \xrightarrow{\sigma} P' \quad z \notin n(\alpha)}{(\nu z)P \xrightarrow{\sigma} (\nu z)P'} \qquad \textbf{SInp}\ \frac{P \xrightarrow{\sigma} P' \quad \sigma \neq \tau}{\underline{x(y)}.P \xrightarrow{x(y) \cdot \sigma} P'}$$

Table 5.2: Multi π late semantic with structural congruence

$$\text{S1L}\ \frac{}{Sync(x(y), \overline{x}z, \tau, \{z/y\}, \{\})} \qquad \text{S1R}\ \frac{}{Sync(\overline{x}z, x(y), \tau, \{\}, \{z/y\})}$$

$$\text{S2L}\ \frac{}{Sync(x(y), \overline{x}z \cdot \sigma, \sigma, \{z/y\}, \{\})} \qquad \text{S2R}\ \frac{}{Sync(\overline{x}z \cdot \sigma, x(y), \sigma, \{\}, \{z/y\})}$$

$$\text{S3L}\ \frac{}{Sync(x(y) \cdot \sigma, \overline{x}z, \sigma\{z/y\}, \{z/y\}, \{\})} \qquad \text{S3R}\ \frac{}{Sync(\overline{x}z, x(y) \cdot \sigma, \sigma\{z/y\}, \{\}, \{z/y\})}$$

$$\text{S4L}\ \frac{Sync(\sigma_1, \sigma_2\{z/y\}, \sigma_3, \delta_1, \delta_2)}{Sync(x(y) \cdot \sigma_1, \overline{x}z \cdot \sigma_2, \sigma_3, \{z/y\}\delta_1, \delta_2)} \qquad \text{S4R}\ \frac{Sync(\sigma_1, \sigma_2\{z/y\}, \sigma_3, \delta_1, \delta_2)}{Sync(\overline{x}z \cdot \sigma_1, x(y) \cdot \sigma_2, \sigma_3, \delta_1, \{z/y\}\delta_2)}$$

$$\text{I1L}\ \frac{Sync(\sigma_1, \sigma_2, \tau, \delta_1, \delta_2)}{Sync(\alpha \cdot \sigma_1, \sigma_2, \alpha, \delta_1, \delta_2)} \qquad \text{I1R}\ \frac{Sync(\sigma_1, \sigma_2, \tau, \delta_1, \delta_2)}{Sync(\sigma_1, \alpha \cdot \sigma_2, \alpha, \delta_1, \delta_2)}$$

$$\text{I2L}\ \frac{Sync(\sigma_1, \sigma_2, \sigma_3, \delta_1, \delta_2)}{Sync(\alpha \cdot \sigma_1, \sigma_2, \alpha \cdot \sigma_3, \delta_1, \delta_2)} \qquad \text{I2R}\ \frac{Sync(\sigma_1, \sigma_2, \sigma_3, \delta_1, \delta_2)}{Sync(\sigma_1, \alpha \cdot \sigma_2, \alpha \cdot \sigma_3, \delta_1, \delta_2)}$$

$$\text{I3L}\ \frac{}{Sync(\alpha, \sigma, \alpha \cdot \sigma, \delta_1, \delta_2)} \qquad \text{I3R}\ \frac{}{Sync(\sigma, \alpha, \alpha \cdot \sigma, \delta_1, \delta_2)}$$

$$\text{I3L}\ \frac{}{Sync(\epsilon, \sigma, \sigma, \delta_1, \delta_2)} \qquad \text{I3R}\ \frac{}{Sync(\sigma, \epsilon, \sigma, \delta_1, \delta_2)}$$

Table 5.3: Synchronization relation

Example

1	$(\underline{\overline{a}f}.\overline{b}g.P \mid \underline{a(w)}.a(z).Q) \mid \underline{b(y)}.\overline{a}h.R \xrightarrow{\tau} (P \mid Q\{f/w\})\{h/z\} \mid R\{g/y\}$	LCom
2	$\underline{\overline{a}f}.\overline{b}g.P \mid \underline{a(w)}.a(z).Q \xrightarrow{\overline{b}g \cdot a(z)} P \mid Q\{f/w\}$	LCom
3	$\underline{\overline{a}f}.\overline{b}g.P \xrightarrow{\overline{a}f \cdot \overline{b}g} P$	SOut
4	$\overline{b}g.P \xrightarrow{\overline{b}g} P$	Pref
5	$\underline{a(w)}.a(z).Q \xrightarrow{a(w) \cdot a(z)} Q$	SInp
6	$a(z).Q \xrightarrow{a(z)} Q$	Pref
7	$Sync(\overline{a}f \cdot \overline{b}g, a(w) \cdot a(z), \overline{b}g \cdot a(z), \{\}, \{f/w\})$	S4R
8	$Sync(\overline{b}g, a(z)\{f/w\}, \overline{b}g \cdot a(z), \{\}, \{\})$	I3L
9	$Sync(\epsilon, a(z), a(z), \{\}, \{\})$	I4R
10	$\underline{b(y)}.\overline{a}h.R \xrightarrow{b(y) \cdot \overline{a}h} R$	SInp
11	$\overline{a}h.R \xrightarrow{\overline{a}h} R$	Pref
12	$Sync(\overline{b}g \cdot a(z), b(y) \cdot \overline{a}h, \tau, \{h/z\}, \{g/y\})$	S4R
13	$Sync(a(z), \overline{a}h, \tau, \{h/z\}, \{g/y\})$	S1L

Example

1	$\underline{x(a)}.\overline{a}z.P \mid \overline{x}b.Q \xrightarrow{\overline{b}z} P\{b/a\} \mid Q$	LCom
2	$\underline{x(a)}.\overline{a}z.P \xrightarrow{x(a) \cdot \overline{a}z} P$	SInp
3	$\overline{a}z.P \xrightarrow{\overline{a}z} P$	Inp
4	$\overline{x}b.Q \xrightarrow{\overline{x}b} Q$	Pref
5	$Sync(x(a) \cdot \overline{a}z, \overline{x}b, \overline{b}z, \{b/a\}, \{\})$	S3L

Index

Bibliography

[1] E. W. Dijkstra, *Hierarchical ordering of sequential processes*, Acta Informatica 1(2):115-138, 1971 .

[2] Roberto Gorrieri, *A Fully-Abstract Semantics for Atomicity*, Dipartimento di scienze dell'informazione, Universita' di Bologna .

[3] Joachin Parrow, *An Introduction to the π Calculus*, Department Teleinformatics, Rotal Institute of Technology, Stockholm .

[4] Davide Sangiorgi, David Walker, *The π-calculus*, Cambridge University Press .

[5] Milner, Robin, *Communicating and Mobile Systems: The π-calculus*, Cambridge University Press .

[6] MohammedReza Mousavi, Michel A Reniers, *Congruence for Structural Congruences*, Department of Computer Science, Eindhoven University of Technology .