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 Introduzione

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.

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

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

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.

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 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. The restriction operator $(\nu z)P$ and the is also a binder for z. The definition of an identifier $A(x_1, \cdots, x_n) = P$ is also a binder, specifically the names $x_1, \cdots, x_n$ are bound in the process P.

Definition $bn(P)$ 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 as follows:

$$B(0, I) = \emptyset$$

$$B(\overline{x}y.Q, I) = B(Q, I)$$

$$B(x(y).Q, I) = \{y\} \cup B(Q, I)$$

$$B(\tau.Q, I) = B(Q, I)$$

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

$$B(Q + R, I) = B(Q, I) \cup B(R, I)$$

$$B(Q|R, I) = B(Q, I) \cup B(R, I)$$

$$B((\nu a)Q, I) = \{a, \overline{a}\} \cup B(Q, I)$$

Definition 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 as follows:

$$F(0, I) = \emptyset$$

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

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

$$F(\tau.Q, I) = F(Q, I)$$

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

$$F(Q + R, I) = F(Q, I) \cup F(R, I)$$

$$F(Q|R, I) = F(Q, I) \cup F(R, I)$$

$$F((\nu a)Q, I) = F(Q, I) - \{a, \overline{a}\}$$

Definition $n(P)$ 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 $P\{b/a\}$ is the syntactic substitution of name b for a different name a inside a π calculus process, and it consist 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 as follows:

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

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

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

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

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

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

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

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

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

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

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

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

2.2 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 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 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 We define a *structural congruence* $\equiv$ as the smallest congruence on processes that satisfies the following axioms

SC-ALP	$\dfrac{P \overset{\alpha}{\equiv} Q}{P \equiv Q}$	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 $A(\tilde{x}) \overset{def}{=} P$

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 $x(y).\bar{z}y$ and $x(w).\bar{z}w$. In the semantic of pi calculus we can use the structural congruence with the rule SC-ALP or the SOS rule

$$\frac{P \xrightarrow{\alpha} P' \quad P \overset{\alpha}{\equiv} Q}{Q \xrightarrow{\alpha} P'}$$

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:

$$\textbf{Sum-L}\ \frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'} \qquad \textbf{Sum-R}\ \frac{Q \xrightarrow{\alpha} Q'}{P + Q \xrightarrow{\alpha} Q'}$$

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.

2.3 early 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 occurres at an early point in the inference of this translation, namely during the inference of the input action.

Definition The *early transition relation* $\rightarrow \subseteq \mathbb{P} \times \mathbb{A} \times \mathbb{P}$ is the smallest relation induced by the following rules:

Out $\dfrac{}{\overline{x}y.P \xrightarrow{\overline{x}y} P}$

EInp $\dfrac{}{x(y).P \xrightarrow{xz} P\{z/y\}}$

Par-L $\dfrac{P \xrightarrow{\alpha} P' \quad bn(\alpha) \cap fn(Q) = \emptyset}{P|Q \xrightarrow{\alpha} P'|Q}$

Par-R $\dfrac{Q \xrightarrow{\alpha} Q' \quad bn(\alpha) \cap fn(Q) = \emptyset}{P|Q \xrightarrow{\alpha} P|Q'}$

Sum-L $\dfrac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'}$

Sum-R $\dfrac{Q \xrightarrow{\alpha} Q'}{P + Q \xrightarrow{\alpha} Q'}$

Tau $\dfrac{}{\tau.P \xrightarrow{\tau} P}$

Res $\dfrac{P \xrightarrow{\alpha} P' \quad z \notin n(\alpha)}{(\nu z)P \xrightarrow{\alpha} (\nu z)P'}$

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

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

ClsLOut $\dfrac{P \xrightarrow{\overline{x}(z)} P' \quad Q \xrightarrow{xz} Q' \quad z \notin fn(Q)}{P|Q \xrightarrow{\tau} (\nu z)(P'|Q')}$

ClsROut $\dfrac{P \xrightarrow{xz} P' \quad Q \xrightarrow{\overline{x}(z)} Q' \quad z \notin fn(P)}{P|Q \xrightarrow{\tau} (\nu z)(P'|Q')}$

ClsLIn $\dfrac{P \xrightarrow{\overline{x}z} P' \quad Q \xrightarrow{x(z)} Q' \quad z \notin fn(Q)}{P|Q \xrightarrow{\tau} (\nu z)(P'|Q')}$

ClsRIn $\dfrac{P \xrightarrow{x(\nu z)} P' \quad Q \xrightarrow{\overline{x}z} Q' \quad z \notin fn(P)}{P|Q \xrightarrow{\tau} (\nu z)(P'|Q')}$

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

OpnOut $\dfrac{P \xrightarrow{\overline{x}z} P' \quad z \neq x}{(\nu z)P \xrightarrow{\overline{x}(z)} P'}$

OpnIn $\dfrac{P \xrightarrow{xz} P' \quad z \neq x}{(\nu z)P \xrightarrow{x(z)} P' \quad z \neq x}$

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.

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

If $b \in fn(P)$?

Example We want to prove now that:

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

2.4 early semantic with structural congruence

Definition The *early transition relation with structural congruence* $\rightarrow \subseteq \mathbb{P} \times \mathbb{A} \times \mathbb{P}$ is the smallest relation induced by the following rules:

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

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

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

Example We prove now that

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

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

$$(\nu b)(P\{b/x\} \mid Q) \equiv (P\{b/x\} \mid Q)$$

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:

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

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

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

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

Vanno bene le due regole Out e Einp che elenchi in sezione 2.3. NON HO CAPITO. devo usare le regole Out ed EInp della sezione 2.3 nella sezione 2.5?

2.5 late semantic without structural congruence

Definition The *late transition relation without structural congruence* $\rightarrow \subseteq \mathbb{P}\times\mathbb{A}\times\mathbb{P}$ is the smallest relation induced by the following rules:

$$\textbf{LInp}\;\dfrac{z \notin fn(P)}{x(z).P \xrightarrow{xz} P} \qquad \textbf{Res}\;\dfrac{P \xrightarrow{\alpha} P' \quad z \notin n(\alpha)}{(\nu z)P \xrightarrow{\alpha} (\nu z)P'}$$

$$\textbf{Sum-L}\;\dfrac{P \xrightarrow{\alpha} P'}{P+Q \xrightarrow{\alpha} P'} \qquad \textbf{Sum-R}\;\dfrac{Q \xrightarrow{\alpha} Q'}{P+Q \xrightarrow{\alpha} Q'}$$

$$\textbf{Par-L}\;\dfrac{P \xrightarrow{\alpha} P' \quad bn(\alpha)\cap fn(Q)=\emptyset}{P|Q \xrightarrow{\alpha} P'|Q} \qquad \textbf{Par-R}\;\dfrac{Q \xrightarrow{\alpha} Q' \quad bn(\alpha)\cap fn(Q)=\emptyset}{P|Q \xrightarrow{\alpha} P|Q'}$$

$$\textbf{LCom}\;\dfrac{P \xrightarrow{x(y)} P' \quad Q \xrightarrow{\overline{x}z} Q' \quad z \notin fn(P)}{P|Q \xrightarrow{\tau} P'\{z/y\}|Q'} \qquad \textbf{RCom}\;\dfrac{P \xrightarrow{\overline{x}z} P' \quad Q \xrightarrow{x(y)} Q' \quad z \notin fn(P)}{P|Q \xrightarrow{\tau} P'|Q'\{z/y\}}$$

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

$$\textbf{CloseL}\;\dfrac{P \xrightarrow{\overline{x}(z)} P' \quad Q \xrightarrow{xz} Q' \quad z \notin fn(Q)}{P|Q \xrightarrow{\tau} (\nu z)(P'|Q')} \qquad \textbf{CloseR}\;\dfrac{P \xrightarrow{xz} P' \quad Q \xrightarrow{\overline{x}(z)} Q' \quad z \notin fn(P)}{P|Q \xrightarrow{\tau} (\nu z)(P'|Q')}$$

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

2.6 late semantic with structural congruence

Definition The *late transition relation with structural congruence* $\rightarrow \subseteq \mathbb{P}\times\mathbb{A}\times\mathbb{P}$ is the smallest relation induced by the following rules:

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

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

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

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

Example We prove now that

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

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

$$(\nu b)(P\{b/x\} \mid Q) \equiv (P\{b/x\} \mid Q)$$

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:

$$\textsc{Res}\ \frac{\textsc{LCom}\ \dfrac{\textsc{Inp}\ \dfrac{}{a(x).P \ \xrightarrow{ax}\ P} \qquad \textsc{Out}\ \dfrac{}{\overline{a}b.Q \ \xrightarrow{\overline{a}b}\ Q}}{a(x).P \mid \overline{a}b.Q \ \xrightarrow{\tau}\ P\{b/x\} \mid Q}}{(\nu b)(a(x).P \mid \overline{a}b.Q) \ \xrightarrow{\tau}\ (\nu b)(P\{b/x\} \mid Q)}$$

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

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

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

2.7 behavioural semantic

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

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

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

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

Early bisimulation is preserved by all operators except input prefix.

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

$$P \sim_E Q \text{ is } \forall\sigma \ P\sigma\dot{\sim}_E Q\sigma$$

where σ is a substitution.

The early congruence is the largest congruence in $\dot{\sim}_E$.
In the following definition we consider a subcalculus without restriction.

Definition 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.

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(perche'?). 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:

$$bn(\underline{\overline{x}y}) = \emptyset \quad fn(\underline{\overline{x}y}) = \{x, y\}$$

3.2 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 $\rightarrow \subseteq \mathbb{P}_m \times (\mathbb{A}_m^+ \cup \{\tau\}) \times \mathbb{P}_m$

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 The *early transition relation without structural congruence* is the smallest relation induced by the following rules:

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

$$\textbf{Tau}\ \frac{}{\tau.P \xrightarrow{\tau} P} \qquad \textbf{SOut}\ \frac{P \xrightarrow{\sigma} P^{'} \quad \sigma \neq \tau}{\underline{\overline{x}y}.P \xrightarrow{\overline{x}y\cdot\sigma} P^{'}}$$

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

$$\textbf{PAr}\ \frac{P \xrightarrow{\sigma} P^{'} \quad bn(\sigma) \cap fn(Q) = \emptyset}{P|Q \xrightarrow{\sigma} P^{'}|Q} \qquad \textbf{EComSng}\ \frac{P \xrightarrow{xy} P^{'} \qquad Q \xrightarrow{\overline{x}y} Q^{'}}{P|Q \xrightarrow{\tau} P^{'}|Q^{'}}$$

$$\textbf{Res}\ \frac{P \xrightarrow{\sigma} P^{'} \ z \notin n(\alpha)}{(\nu)zP \xrightarrow{\sigma} (\nu)zP^{'}} \qquad \textbf{EComSeq}\ \frac{P \xrightarrow{xy} P^{'} \qquad Q \xrightarrow{\overline{x}y\cdot\sigma} Q^{'}}{P|Q \xrightarrow{\sigma} P^{'}|Q^{'}}$$

In the following examples we omit sometimes the rule Str.

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

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

$x \notin n(\tau)$

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

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

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

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

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

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

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

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

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

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

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

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

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

SOut $\underline{\overline{a}x}.c(x).0 \xrightarrow{\overline{a}x\cdot cx} 0$

Inp $c(x).0 \xrightarrow{cx} 0$

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:

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

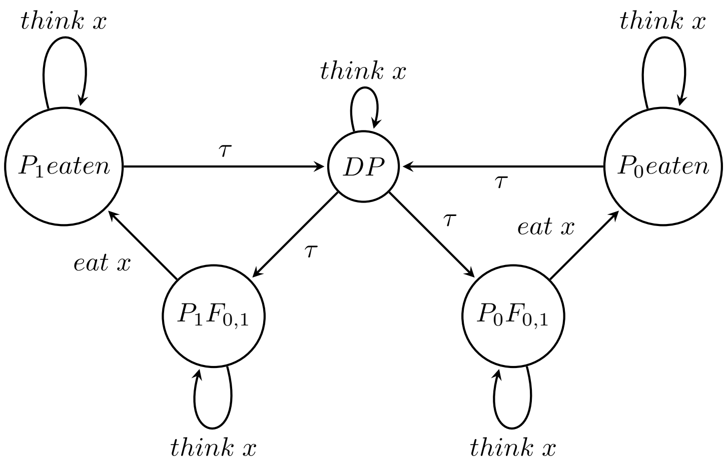
  also in this case the name x is not relevant.

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

$a \notin n(\tau)$

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

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

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

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

$$.\quad \textbf{Out}\ \overline{a}x.0 \xrightarrow{\overline{a}x} 0$$

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

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

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

3.3 late operational semantic with structural congruence

Definition The *late transition relation with structural congruence* is the smallest relation induced by the following rules:

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

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

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

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

3.4 behavioural semantic

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(perche'?). 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:

$$bn(\underline{x(y)}) = \{y\} \quad fn(\underline{x(y)}) = \{x\}$$

4.2 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.

Definition The *early transition relation with structural congruence* is the smallest relation induced by the following rules:

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

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

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

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

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

Definition 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:

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

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

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

4.3 late operational semantic with structural congruence

Definition The *late transition relation with structural congruence* is the smallest relation induced by the following rules:

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

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

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

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

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:

$$\begin{array}{lcl} bn(\underline{x(y)}) & = & \{y\} \\ bn(\underline{\overline{x}y}) & = & \emptyset \end{array} \quad \begin{array}{lcl} fn(\underline{x(y)}) & = & \{x\} \\ fn(\underline{\overline{x}y}) & = & \{x, y\} \end{array}$$

5.2 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 plus two new kind of action:

strong input	strong output
$\underline{xy}$	$\underline{\overline{x}y}$

  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.

Definition The *early transition relation with structural congruence* is the smallest relation induced by the following rules:

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

$$\textbf{SOut}\ \dfrac{}{\underline{\overline{x}y}.P \xrightarrow{\overline{x}y} P}$$

$$\textbf{Tau}\ \dfrac{}{\tau.P \xrightarrow{\tau} P} \qquad \textbf{SInp}\ \dfrac{}{\underline{x(y)}.P \xrightarrow{xz} P\{z/y\}}$$

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

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

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

In this semantic we cannot have

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

nor

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

but we have

$$\textsc{Seq}\ \dfrac{\textsc{SOut}\ \dfrac{}{\underline{\bar{b}z}.P\{b/a\} \xrightarrow{\bar{b}z} P\{b/a\}} \qquad \textsc{SInp}\ \dfrac{}{\underline{x(a)}.\underline{\bar{a}z}.P \xrightarrow{xb} (\underline{\bar{a}z}.P)\{b/a\}}}{\underline{x(a)}.\underline{\bar{a}z}.P \xrightarrow{xb\cdot\bar{b}z} P\{b/a\}}$$

Definition We define the synchronization relation in the following way:

$$\textsc{Com1L}\ \dfrac{}{Sync(xy, \overline{x}y, \tau)} \qquad \textsc{Com2L}\ \dfrac{}{Sync(xy\cdot\sigma, \overline{x}y, \sigma)} \qquad \textsc{Com3L}\ \dfrac{Sync(\sigma_1, \sigma_2, \sigma_3)}{Sync(xy\cdot\sigma_1, \overline{x}y\cdot\sigma_2, \sigma_3)}$$

$$\textsc{Com1R}\ \dfrac{}{Sync(\overline{x}y, xy, \tau)} \qquad \textsc{Com2R}\ \dfrac{}{Sync(\overline{x}y\cdot\sigma, xy, \sigma)} \qquad \textsc{Com3R}\ \dfrac{Sync(\sigma_1, \sigma_2, \sigma_3)}{Sync(\overline{x}y\cdot\sigma_1, xy\cdot\sigma_2, \sigma_3)}$$

NON FUNZIONA!!

5.3 late operational semantic with structural congruence

Definition The *late transition relation with structural congruence* is the smallest relation induced by the following rules:

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

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

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

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

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

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