

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

.....



ANNO ACCADEMICO 20011/2012

0.1 Introduzione

Qual e' lo stato dell'arte? Stato dell'arte: pi-calcolo, mancanza di multi-party synch mobile, per ccs e' stato proposto multi-ccs per multy-party synch statica.

Cosa ho fatto? multi-pi

Perche'?

i lavori di Nobuko Yoshida e Kohei Honda non sono riuscito a trovarli

Contents

0.1	Introduzione	3
1	Multi ccs	7
1.1	syntax	7
1.2	semantic	7
2	Π calculus	9
2.1	syntax	9
2.2	structural congruence	11
2.3	early semantic without structural congruence	13
2.4	early semantic with structural congruence	14
2.5	late semantic without structural congruence	15
2.6	late semantic with structural congruence	15
2.7	altre semantiche?	16
2.8	behavioural semantic	16
3	Multi π calculus solo output	17
3.1	syntax	17
3.2	early operational semantic with structural congruence	17
3.3	late operational semantic with structural congruence	20
3.4	behavioural semantic	20
3.5	linear step semantics	20
4	Multi π calculus solo input	21
4.1	syntax	21
4.2	early operational semantic with structural congruence	21
4.3	late operational semantic with structural congruence	22
5	Multi π calculus input e output	23
5.1	syntax	23
5.2	early operational semantic with structural congruence	23
5.3	late operational semantic with structural congruence	24

Chapter 1

Multi ccs

1.1 syntax

Multi ccs is introduced in [2]. Let \mathbb{L} be a numerable set of channel names, let $\bar{\mathbb{L}}$ be the set of the complementary of channel names $\bar{\mathbb{L}} = \{\bar{l} : l \in \mathbb{L}\}$, let $Act = \mathbb{L} \cup \bar{\mathbb{L}} \cup \{\tau\}$ be the set of all action, where $\tau \notin \mathbb{L} \cup \bar{\mathbb{L}}$. The process terms are generated by the following grammar:

$$p ::= 0 \mid \mu.p \mid \underline{\mu}.p \mid p + p \mid p|p \mid (\nu a)p \mid C$$

and we have also expression in the form

$$C = p$$

in which the constant C is defined as the process p . The meaning of each production is the following:

0 is terminated process

$\mu.p$ is action prefixing, μ is an action

$\underline{\mu}.p$ is strong action prefixing

$p + p$ is non deterministic sum of two processes

$p|p$ is parallel composition of two processes

$(\nu a)p$ is restriction: the name a is private in p . νa is a binder for the name a in the process p

C is a constant

Definition A *guarded process* is

We denote with \mathbb{P} the set of all process that are guarded and closed with respect to constant.

1.2 semantic

Definition A *labeled transition system* is a labeled graph, that is a triple

$$(N, L, LE)$$

such that:

N is a set of nodes

L is a set of labels

LE is a set of labeled edges(also called transition relation): $LE \subseteq N \times L \times N$

Definition The *semantic of a multi-CCS process* is a labeled transition system

$$(\mathbb{P}, \mathbb{A}, \rightarrow)$$

such that

\mathbb{P} is the set of all multi ccs processes

\mathbb{A} is the set $(\mathbb{L} \cup \bar{\mathbb{L}})^+ \cup \{\tau\}$

\rightarrow is the minimal relation contained in $\mathbb{P} \times \mathbb{A} \times \mathbb{P}$ that is generated by the rules of the operational semantic of multi-CCS

Definition The *operational semantic of multi-CCS* is given by the following rules:

Pref :

$$\mu.p \rightarrow^\mu p$$

*S - Pref*₁ :

$$\frac{p \rightarrow^\sigma p'}{\tau.p \rightarrow^\sigma p'}$$

Example We show an example of three processes that synchronize. In particular we prove that

$$\begin{array}{c} \text{COM} \frac{\text{INP} \frac{\overline{a.0 \xrightarrow{a} 0} \quad a \neq \tau}{\text{SPREF3} \frac{\overline{a.a.0 \xrightarrow{aa} 0}}{\text{COM} \frac{\overline{a.a.0 | \bar{a}.0 \xrightarrow{a} 0 | 0} \quad \text{OUT} \frac{\overline{\bar{a}.0 \xrightarrow{\bar{a}} 0} \quad \text{Sync}(aa, \bar{a}, a)}}{\overline{a.a.0 | \bar{a}.0 \xrightarrow{a} 0 | 0}}} \\ \text{RES} \frac{\text{COM} \frac{\overline{a.a.0 | \bar{a}.0 \xrightarrow{a} 0 | 0} \quad \text{OUT} \frac{\overline{\bar{a}.0 \xrightarrow{\bar{a}} 0} \quad \text{Sync}(a, \bar{a}, \tau)}}{((\underline{a}.a.0 | \bar{a}.0) | \bar{a}.0) \xrightarrow{\tau} ((0 | 0) | 0)} \quad a \notin n(\tau)}{(\nu a)((\underline{a}.a.0 | \bar{a}.0) | \bar{a}.0) \xrightarrow{\tau} (\nu a)((0 | 0) | 0)} \end{array}$$

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, \dots, z . These 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, \dots . A process can perform the following actions:

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

The process are defined by the following grammar:

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

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 and then behave like P , the action can be:

$\bar{x}y$ 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 synchronize 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.

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

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

where the x_i must be pairwise disjoint. The intuition is that if the y_i replace the x_i then $A(y_1, \dots, y_n)$ 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 P is *guarded* in Q if P is a proper subterm of a prefix from in Q .

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 $(\nu z)P$ is also a binder for z .

Definition $bn(P)$ is the set of names that have a bound occurrence in P and is defined in the following way:

$$\begin{aligned} bn(P) = \text{match } P \text{ with} \\ &| \pi.Q \rightarrow fn(Q) \cup \\ &\quad \text{match } \pi \text{ with} \\ &\quad | \bar{x}y \rightarrow \emptyset \\ &\quad | x(y) \rightarrow \{y\} \\ &\quad | \tau \rightarrow \emptyset \\ &| 0 \rightarrow \emptyset \\ &| P|Q \rightarrow bn(P) \cup bn(Q) \\ &| P + Q \rightarrow bn(P) \cup bn(Q) \\ &| (\nu x)P \rightarrow bn(P) \cup \{x\} \\ &| A(y_1, \dots, y_n) \rightarrow bn(def(A)) \end{aligned}$$

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 .

$$\begin{aligned} fn(P) = \text{match } P \text{ with} \\ &| \pi.Q \rightarrow fn(Q) \cup \\ &\quad \text{match } \pi \text{ with} \\ &\quad | \bar{x}y \rightarrow \{x, y\} \\ &\quad | x(y) \rightarrow \{x\} \\ &\quad | \tau \rightarrow \emptyset \\ &| 0 \rightarrow \emptyset \\ &| P|Q \rightarrow fn(P) \cup fn(Q) \\ &| P + Q \rightarrow fn(P) \cup fn(Q) \\ &| (\nu x)P \rightarrow fn(P) \\ &| A(y_1, \dots, y_n) \rightarrow fn(def(A)) \end{aligned}$$

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 $A(x_1, \dots, x_n) = P$ we assume that $fn(P) \subseteq \{x_1, \dots, x_n\}$.

Definition A *substitution* is a function from names to names

$$\sigma : \mathbb{N} \rightarrow \mathbb{N}$$

We write $\{x_1/y_1, \dots, x_n/y_n\}$ for the substitution that maps y_i to x_i and that identifies all other names.

Definition 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 Processes P and Q are α convertible if Q can be obtained from P by a finite number of changes of bound names

In the rest of the thesis, we assume that the bound names of the processes are different from their free names and from the names of the substitutions.

Definition We can *apply* a substitution σ to a process P , and we write $P\sigma$. In order to apply σ to P , we might first need to perform appropriate α conversions in order to avoid name captures and then we can actually replace every free names of P with the mapping from σ . We define the *application of a substitution* to a process in the following way:

$$\begin{aligned}
P\sigma &= \text{match } P \text{ with} \\
&| 0 \rightarrow 0 \\
&| \pi.P \rightarrow \pi\sigma.P\sigma \\
&| P|Q \rightarrow P\sigma|Q\sigma \\
&| P + Q \rightarrow P\sigma + Q\sigma \\
&| (\nu x)P \rightarrow (\nu x)P\sigma \\
&| A(y_1, \dots, y_n) \rightarrow P\sigma \quad \text{if } A(y_1, \dots, y_n) = P
\end{aligned}$$

We define the application of a substitution to an action in the following way:

$$\begin{aligned}
\pi\sigma &= \text{match } \pi \text{ with} \\
&| \bar{x}y \rightarrow \bar{x}\sigma y\sigma \\
&| x(z) \rightarrow x\sigma(z\sigma) \\
&| \tau \rightarrow \tau
\end{aligned}$$

We regard a substitution as binding more tightly than a process operator.

Definition We say that an occurrence of a process Q in a process P is *guarded* if it is inside a prefix in P . Otherwise we call it *unguarded*.

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 , then 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	$\frac{P \stackrel{\alpha}{\equiv} Q}{P \equiv Q}$	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	$P_1 (P_2 P_3) \equiv (P_1 P_2) P_3$	associativity
SC-COM-COM	$P_1 P_2 \equiv P_2 P_1$	commutativity
SC-COM-INC	$P 0 \equiv P$	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	$(\nu z)(P_1 P_2) \equiv P_1 (\nu z)P_2$ 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 $A(\tilde{x}) \stackrel{def}{=} P$

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:

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

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

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

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:

$$\mathbf{Sum-L} \frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'} \quad \mathbf{Sum-R} \frac{Q \xrightarrow{\alpha} Q'}{P + Q \xrightarrow{\alpha} Q'}$$

or by the following rule and axiom:

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

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 are π calculus actions plus the bound output $\bar{x}(y)$. 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*

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

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

$\text{Out} \frac{}{\bar{x}y.P \xrightarrow{\bar{x}y} P}$	$\text{EInp} \frac{z \notin fn(P)}{x(y).P \xrightarrow{xz} P\{z/y\}}$
$\text{Par-L} \frac{P \xrightarrow{\alpha} P' \quad bn(\alpha) \cap fn(Q) = \emptyset}{P Q \xrightarrow{\alpha} P' Q}$	$\text{Par-R} \frac{Q \xrightarrow{\alpha} Q' \quad bn(\alpha) \cap fn(Q) = \emptyset}{P Q \xrightarrow{\alpha} P Q'}$
$\text{Sum-L} \frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'}$	$\text{Sum-R} \frac{Q \xrightarrow{\alpha} Q'}{P + Q \xrightarrow{\alpha} Q'}$
$\text{Tau} \frac{}{\tau.P \xrightarrow{\tau} P}$	$\text{Res} \frac{P \xrightarrow{\alpha} P' \quad z \notin n(\alpha)}{(\nu z)P \xrightarrow{\alpha} (\nu z)P'}$
$\text{EComR} \frac{P \xrightarrow{\bar{x}y} P' \quad Q \xrightarrow{xy} Q'}{P Q \xrightarrow{\tau} P' Q'}$	$\text{EComL} \frac{P \xrightarrow{xy} P' \quad Q \xrightarrow{\bar{x}y} Q'}{P Q \xrightarrow{\tau} P' Q'}$
$\text{ClsLOut} \frac{P \xrightarrow{\bar{x}(\nu z)} P' \quad Q \xrightarrow{xz} Q' \quad z \notin fn(Q)}{P Q \xrightarrow{\tau} (\nu z)(P' Q')}$	$\text{ClsROut} \frac{P \xrightarrow{xz} P' \quad Q \xrightarrow{\bar{x}(z)} Q' \quad z \notin fn(P)}{P Q \xrightarrow{\tau} (\nu z)(P' Q')}$
$\text{ClsLIn} \frac{P \xrightarrow{\bar{x}z} P' \quad Q \xrightarrow{x(z)} Q' \quad z \notin fn(Q)}{P Q \xrightarrow{\tau} (\nu z)(P' Q')}$	$\text{ClsRIn} \frac{P \xrightarrow{x(\nu z)} P' \quad Q \xrightarrow{\bar{x}z} Q' \quad z \notin fn(P)}{P Q \xrightarrow{\tau} (\nu z)(P' Q')}$
$\text{Cns} \frac{A(\tilde{x}) \stackrel{def}{=} P \quad P\{\tilde{y}/\tilde{x}\} \xrightarrow{\alpha} P'}{A(\tilde{y}) \xrightarrow{\alpha} P'}$	
$\text{OpnOut} \frac{P \xrightarrow{\bar{x}z} P' \quad z \neq x}{(\nu z)P \xrightarrow{\bar{x}(z)} P'}$	$\text{OpnIn} \frac{P \xrightarrow{xz} P' \quad z \neq x}{(\nu z)P \xrightarrow{x(z)} P' \quad z \neq x}$

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

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

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.

$$\text{CLOSER} \frac{\text{EINP} \frac{b \notin fn(P)}{a(x).P \xrightarrow{ab} P\{b/x\}} \quad \text{OPN} \frac{\text{OUT} \frac{\overline{ab}.Q \xrightarrow{\bar{a}b} Q} \quad a \neq b}{(\nu b)\bar{a}b.Q \xrightarrow{\bar{a}(b)} Q} \quad b \notin fn((\nu b)\bar{a}b.Q)}{a(x).P \mid (\nu b)\bar{a}b.Q \xrightarrow{\tau} (\nu b)(P\{b/x\} \mid Q)}$$

If $b \in fn(P)$?

Example We want to prove now that:

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

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:

$$\begin{array}{lll} \text{Out} \frac{}{\bar{x}y.P \xrightarrow{\bar{x}y} P} & \text{EInp} \frac{}{x(z).P \xrightarrow{xy} P\{y/z\}} & \text{Par} \frac{P \xrightarrow{\alpha} P' \quad bn(\alpha) \cap fn(Q) = \emptyset}{P \mid Q \xrightarrow{\alpha} P' \mid Q} \\ \text{Sum} \frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'} & \text{ECom} \frac{P \xrightarrow{xy} P' \quad Q \xrightarrow{\bar{x}y} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'} & \text{Res} \frac{P \xrightarrow{\alpha} P' \quad z \notin n(\alpha)}{(\nu z)P \xrightarrow{\alpha} (\nu z)P'} \\ \text{Tau} \frac{}{\tau.P \xrightarrow{\tau} P} & \text{Opn} \frac{P \xrightarrow{\bar{x}z} P' \quad z \neq x}{(\nu z)P \xrightarrow{\bar{x}(z)} P'} & \text{Str} \frac{P \equiv P' \quad P \xrightarrow{\alpha} Q \quad Q \equiv Q'}{P' \xrightarrow{\alpha} Q'} \end{array}$$

Example We prove now that

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

This follows from

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

and

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

and

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

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

$$\text{RES} \frac{\text{COM} \frac{\text{EINP} \frac{}{a(x).P \xrightarrow{ab} P\{b/x\}} \quad \text{OUT} \frac{}{\bar{a}b.Q \xrightarrow{\bar{a}b} Q}}{a(x).P \mid \bar{a}b.Q \xrightarrow{\tau} P\{b/x\} \mid Q}}{(\nu b)(a(x).P \mid \bar{a}b.Q) \xrightarrow{\tau} (\nu b)(P\{b/x\} \mid Q)}$$

Example We want to prove now that:

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

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

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

then we have

$$\text{RES} \frac{\text{COM} \frac{\text{EINP} \frac{b \notin \text{fn}(P\{c/b\})}{a(x).P\{c/b\} \xrightarrow{ab} P\{c/b\}\{b/x\}} \quad \text{OUT} \frac{}{\bar{a}b.Q \xrightarrow{\bar{a}b} Q}}{(a(x).(P\{c/b\})|\bar{a}b.Q) \xrightarrow{\tau} (P\{c/b\}\{b/x\}|Q)} }{(\nu c)(a(x).(P\{c/b\})|\bar{a}b.Q) \xrightarrow{\tau} (\nu c)(P\{c/b\}\{b/x\}|Q)}$$

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

2.5 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:

$$\begin{array}{ll} \text{Prf} \frac{}{\alpha.P \xrightarrow{\alpha} P} & \text{Res} \frac{P \xrightarrow{\alpha} P' \quad z \notin \text{fn}(\alpha)}{(\nu z)P \xrightarrow{\alpha} (\nu z)P'} \\ \text{Sum-L} \frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'} & \text{Sum-R} \frac{Q \xrightarrow{\alpha} Q'}{P + Q \xrightarrow{\alpha} Q'} \\ \text{Par-L} \frac{P \xrightarrow{\alpha} P' \quad \text{bn}(\alpha) \cap \text{fn}(Q) = \emptyset}{P|Q \xrightarrow{\alpha} P'|Q} & \text{Par-R} \frac{Q \xrightarrow{\alpha} Q' \quad \text{bn}(\alpha) \cap \text{fn}(Q) = \emptyset}{P|Q \xrightarrow{\alpha} P|Q'} \\ \text{LCom} \frac{P \xrightarrow{x(y)} P' \quad Q \xrightarrow{\bar{x}z} Q' \quad z \notin \text{fn}(P)}{P|Q \xrightarrow{\tau} P'\{z/y\}|Q'} & \text{Cns} \frac{A(\tilde{x}) \stackrel{\text{def}}{=} P \quad P\{\tilde{y}/\tilde{x}\} \xrightarrow{\alpha} P'}{A(\tilde{y}) \xrightarrow{\alpha} P'} \\ \text{Opn} \frac{P \xrightarrow{\bar{x}z} P' \quad z \neq x}{(\nu z)P \xrightarrow{\bar{x}(z)} P'} & \\ \text{CloseL} \frac{P \xrightarrow{\bar{x}(z)} P' \quad Q \xrightarrow{xy} Q' \quad z \notin \text{fn}(Q)}{P|Q \xrightarrow{\tau} (\nu z)(P'\{z/y\}|Q')} & \text{CloseR} \frac{P \xrightarrow{xz} P' \quad Q \xrightarrow{\bar{x}(z)} Q' \quad z \notin \text{fn}(P)}{P|Q \xrightarrow{\tau} (\nu z)(P'|Q')}\end{array}$$

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:

$$\begin{array}{ll} \text{Prf} \frac{}{\alpha.P \xrightarrow{\alpha} P} & \text{Sum} \frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'} \\ \text{Par} \frac{P \xrightarrow{\alpha} P' \quad \text{bn}(\alpha) \cap \text{fn}(Q) = \emptyset}{P|Q \xrightarrow{\alpha} P'|Q} & \text{Res} \frac{P \xrightarrow{\alpha} P' \quad z \notin \text{fn}(\alpha)}{(\nu z)P \xrightarrow{\alpha} (\nu z)P'} \\ \text{LCom} \frac{P \xrightarrow{x(y)} P' \quad Q \xrightarrow{\bar{x}z} Q'}{P|Q \xrightarrow{\tau} P'\{z/y\}|Q'} & \text{Str} \frac{P \equiv P' \quad P \xrightarrow{\alpha} Q \quad Q \equiv Q'}{P' \xrightarrow{\alpha} Q'} \\ \text{Opn} \frac{P \xrightarrow{\bar{x}z} P' \quad z \neq x}{(\nu z)P \xrightarrow{\bar{x}(z)} P'} & \end{array}$$

Example We prove now that

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

This follows from

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

and

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

and

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

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

$$\text{RES} \frac{\text{LCom} \frac{\text{INP} \frac{a(x).P \xrightarrow{ax} P}{a(x).P \mid \bar{a}b.Q \xrightarrow{\tau} P\{b/x\} \mid Q} \quad \text{OUT} \frac{\bar{a}b.Q \xrightarrow{\bar{a}b} Q}{\bar{a}b.Q \xrightarrow{\bar{a}b} Q}}{a(x).P \mid \bar{a}b.Q \xrightarrow{\tau} P\{b/x\} \mid Q}}{(\nu b)(a(x).P \mid \bar{a}b.Q) \xrightarrow{\tau} (\nu b)(P\{b/x\} \mid Q)}$$

Example We want to prove now that:

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

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

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

then we have

$$\text{RES} \frac{\text{LCom} \frac{\text{INP} \frac{b \notin fn(P\{c/b\})}{a(x).P\{c/b\} \xrightarrow{ax} P\{c/b\}} \quad \text{OUT} \frac{\bar{a}b.Q \xrightarrow{\bar{a}b} Q}{\bar{a}b.Q \xrightarrow{\bar{a}b} Q}}{a(x).(P\{c/b\}) \mid \bar{a}b.Q \xrightarrow{\tau} (P\{c/b\}\{b/x\} \mid Q)} \quad \text{OUT} \frac{\bar{a}b.Q \xrightarrow{\bar{a}b} Q}{\bar{a}b.Q \xrightarrow{\bar{a}b} Q}}{(\nu c)(a(x).(P\{c/b\}) \mid \bar{a}b.Q) \xrightarrow{\tau} (\nu c)(P\{c/b\}\{b/x\} \mid Q)}$$

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

2.7 altre semantiche?

- possiamo definire un sistema di transizione non sui processi ma sulle classi di equivalenza di processi modulo una qualche equivalenza, ad esempio la congruenza strutturale?
- possiamo usare delle epsilon transizioni al posto della congruenza strutturale, nel senso di usare lts con epsilon transizioni come semantica intermedia e poi prendere lts delle epsilon chiusure come semantica. Ad esempio

$$\overline{P + (Q + R)} \xrightarrow{\epsilon} (P + Q) + R$$

- denotazionale

2.8 behavioural semantic

Metterci una semantica osservazionale (bisimulazione early e open) ed un po' di proprieta' algebriche (congruenza).

Chapter 3

Multi π calculus solo output

3.1 syntax

As we did with π calculus, we suppose that we have a countable set of names \mathbb{N} , ranged over by lower case letters a, b, \dots, z . These 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, \dots . A multi π process, in addition to the same actions of a π process, can perform also a strong prefix output:

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

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

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

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:

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

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 $\alpha, \alpha_1, \alpha_2, \dots$ to range over the set of actions, we use $\sigma, \sigma_1, \sigma_2, \dots$ 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:

$$\begin{array}{ll}
\mathbf{Out} \frac{}{\bar{x}y.P \xrightarrow{\bar{x}y} P} & \mathbf{EInp} \frac{z \notin fn(P)}{x(y).P \xrightarrow{xz} P\{z/y\}} \\
\\
\mathbf{Tau} \frac{}{\tau.P \xrightarrow{\tau} P} & \mathbf{SOut} \frac{P \xrightarrow{\sigma} P' \quad \sigma \neq \tau}{\bar{x}y.P \xrightarrow{\bar{x}y \cdot \sigma} P'} \\
\\
\mathbf{Sum} \frac{P \xrightarrow{\sigma} P'}{P + Q \xrightarrow{\sigma} P'} & \mathbf{Str} \frac{P \equiv P' \quad P' \xrightarrow{\alpha} Q' \quad Q \equiv Q'}{P \xrightarrow{\alpha} Q} \\
\\
\mathbf{PAr} \frac{P \xrightarrow{\sigma} P' \quad bn(\sigma) \cap fn(Q) = \emptyset}{P|Q \xrightarrow{\sigma} P'|Q} & \mathbf{EComSng} \frac{P \xrightarrow{xy} P' \quad Q \xrightarrow{\bar{x}y} Q'}{P|Q \xrightarrow{\tau} P'|Q'} \\
\\
\mathbf{Res} \frac{P \xrightarrow{\sigma} P' \quad z \notin n(\alpha)}{(\nu)zP \xrightarrow{\sigma} (\nu)zP'} & \mathbf{EComSeq} \frac{P \xrightarrow{xy} P' \quad Q \xrightarrow{\bar{x}y \cdot \sigma} Q'}{P|Q \xrightarrow{\sigma} P'|Q'}
\end{array}$$

In the following examples we omit sometimes the rule *Str*.

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

$$\begin{aligned}
\mathbf{Res} \quad & (\nu x)((\bar{x}y.\bar{x}y.0|x(y).0)|x(y).0) \xrightarrow{\tau} (\nu x)((0|0)|0) \\
& x \notin n(\tau) \\
\mathbf{EComSng} \quad & ((\bar{x}y.\bar{x}y.0|x(y).0)|x(y).0) \xrightarrow{\tau} ((0|0)|0) \\
& \mathbf{EComSeq} \quad \bar{x}y.\bar{x}y.0|x(y).0 \xrightarrow{\bar{x}y} 0|0 \\
& \mathbf{EInp} \quad x(y).0 \xrightarrow{xy} 0 \\
& \mathbf{SOut} \quad \bar{x}y.\bar{x}y.0 \xrightarrow{\bar{x}y \cdot \bar{x}y} 0 \\
& \quad \bar{x}y \neq \tau \\
& \mathbf{Out} \quad \bar{x}y.0 \xrightarrow{\bar{x}y} 0 \\
& \mathbf{Out} \quad x(y).0 \xrightarrow{xy} 0
\end{aligned}$$

Example We want to prove that

$$\begin{aligned}
& (\bar{a}x.c(x).0|b(x).0)|(a(x).0|\bar{b}x.\bar{c}x.0) \xrightarrow{\tau} (0|0)|(0|0) \\
\mathbf{Str} \quad & (\bar{a}x.c(x).0|b(x).0)|(a(x).0|\bar{b}x.\bar{c}x.0) \xrightarrow{\tau} (0|0)|(0|0) \\
& \mathbf{EComSng} \quad (\bar{a}x.c(x).0|a(x).0)|(b(x).0|\bar{b}x.\bar{c}x.0) \xrightarrow{\tau} (0|0)|(0|0) \\
& \mathbf{EComSeq} \quad b(x).0|\bar{b}x.\bar{c}x.0 \xrightarrow{\bar{c}x} 0|0 \\
& \mathbf{EInp} \quad b(x).0 \xrightarrow{bx} 0 \\
& \mathbf{SOut} \quad \bar{b}x.\bar{c}x.0 \xrightarrow{\bar{b}x \cdot \bar{c}x} 0 \\
& \mathbf{Out} \quad \bar{c}x.0 \xrightarrow{\bar{c}x} 0 \\
& \mathbf{EComSeq} \quad \bar{a}x.c(x).0|a(x).0 \xrightarrow{cx} 0|0 \\
& \mathbf{SOut} \quad \bar{a}x.c(x).0 \xrightarrow{\bar{a}x \cdot cx} 0 \\
& \mathbf{Inp} \quad c(x).0 \xrightarrow{cx} 0 \\
& \mathbf{Inp} \quad a(x).0 \xrightarrow{ax} 0 \\
& (\bar{a}x.c(x).0|b(x).0)|(a(x).0|\bar{b}x.\bar{c}x.0) \equiv (\bar{a}x.c(x).0|a(x).0)|(b(x).0|\bar{b}x.\bar{c}x.0)
\end{aligned}$$

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:

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

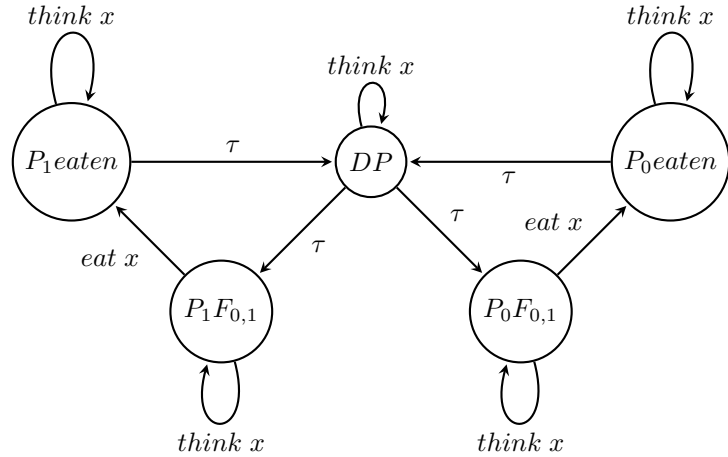
also in this case the name x is not relevant.

- 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:

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

$$a \notin n(\tau)$$

$$\mathbf{EComSng} \ (((\overline{ax}.\overline{ax}.\overline{ax}.0|a(x).0)|a(x).0)|a(x).0) \xrightarrow{\tau} ((0|0)|0)|0)$$

$$\mathbf{EComSeq} \ (\overline{ax}.\overline{ax}.\overline{ax}.0|a(x).0)|a(x).0 \xrightarrow{\overline{ax}} (0|0)|0$$

$$\mathbf{EComSeq} \ \overline{ax}.\overline{ax}.\overline{ax}.0|a(x).0 \xrightarrow{\overline{ax}.\overline{ax}} 0|0$$

$$\begin{array}{l}
\mathbf{SOut} \ \bar{a}x.\bar{a}x.\bar{a}x.0 \xrightarrow{\bar{a}x.\bar{a}x.\bar{a}x} 0 \\
\mathbf{SOut} \ \bar{a}x.\bar{a}x.0 \xrightarrow{\bar{a}x.\bar{a}x} 0 \\
\mathbf{SOut} \ \bar{a}x.\bar{a}x.0 \xrightarrow{\bar{a}x.\bar{a}x} 0 \\
\mathbf{Out} \ \bar{a}x.0 \xrightarrow{\bar{a}x} 0 \\
\mathbf{Inp} \ a(x).0 \xrightarrow{ax} 0 \\
\mathbf{Inp} \ a(x).0 \xrightarrow{ax} 0 \\
\mathbf{Inp} \ a(x).0 \xrightarrow{ax} 0
\end{array}$$

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:

$$\begin{array}{ll}
\mathbf{Pref} \ \frac{\alpha \text{ not a strong prefix}}{\alpha.P \xrightarrow{\alpha} P} & \mathbf{Par} \ \frac{P \xrightarrow{\sigma} P' \quad bn(\sigma) \cap fn(Q) = \emptyset}{P|Q \xrightarrow{\sigma} P'|Q} \\
\\
\mathbf{SOut} \ \frac{P \xrightarrow{\sigma} P' \quad \sigma \neq \tau}{\bar{x}y.P \xrightarrow{\bar{x}y.\sigma} P'} & \mathbf{LComSeq} \ \frac{P \xrightarrow{x(y)} P' \quad Q \xrightarrow{\bar{x}z.\sigma} Q' \quad z \notin fn(P)}{P|Q \xrightarrow{\sigma} P'\{z/y\}|Q'} \\
\\
\mathbf{Sum} \ \frac{P \xrightarrow{\sigma} P'}{P+Q \xrightarrow{\sigma} P'} & \mathbf{Str} \ \frac{P \equiv P' \quad P' \xrightarrow{\alpha} Q' \quad Q \equiv Q'}{P \xrightarrow{\alpha} Q} \\
\\
\mathbf{RES} \ \frac{P \xrightarrow{\sigma} P' \quad z \notin n(\alpha)}{(\nu)zP \xrightarrow{\sigma} (\nu)zP'} & \mathbf{LComSng} \ \frac{P \xrightarrow{x(y)} P' \quad Q \xrightarrow{\bar{x}z} Q' \quad z \notin fn(P)}{P|Q \xrightarrow{\tau} P'\{z/y\}|Q'}
\end{array}$$

3.4 behavioural semantic

3.5 linear step semantics

Chapter 4

Multi π calculus solo input

4.1 syntax

As we did with multi π calculus, we suppose that we have a countable set of names \mathbb{N} , ranged over by lower case letters a, b, \dots, z . These 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, \dots . A multi π process, in addition to the same actions of a π process, can perform also a strong prefix input:

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

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

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

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 nodes is \mathbb{P}_m
- the actions are multi π calculus actions. The set of actions is \mathbb{A}_m , we use $\alpha, \alpha_1, \alpha_2, \dots$ to range over the set of actions, we use $\sigma, \sigma_1, \sigma_2, \dots$ 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:

$$\begin{array}{c}
\text{Out} \frac{}{\bar{x}y.P \xrightarrow{\bar{x}y} P} \qquad \text{EInp} \frac{z \notin fn(P)}{x(y).P \xrightarrow{xz} P\{z/y\}} \\
\\
\text{Tau} \frac{}{\tau.P \xrightarrow{\tau} P} \qquad \text{SInp} \frac{P \xrightarrow{\sigma} P' \quad \sigma \neq \tau}{x(y).P \xrightarrow{xz \cdot \sigma} P'\{z/y\}} \\
\\
\frac{P \xrightarrow{\sigma} P'}{P + Q \xrightarrow{\sigma} P'} \qquad \text{Str} \frac{P \equiv P' \quad P' \xrightarrow{\alpha} Q' \quad Q \equiv Q'}{P \xrightarrow{\alpha} Q} \\
\\
\text{Par} \frac{P \xrightarrow{\sigma} P' \quad bn(\sigma) \cap fn(Q) = \emptyset}{P|Q \xrightarrow{\sigma} P'|Q} \qquad \text{Com} \frac{P \xrightarrow{\sigma_1} P' \quad Q \xrightarrow{\sigma_2} Q' \quad Sync(\sigma_1, \sigma_2, \sigma_3)}{P|Q \xrightarrow{\sigma_3} P'|Q'} \\
\\
\text{Res} \frac{P \xrightarrow{\sigma} P' \quad z \notin n(\alpha)}{(\nu)zP \xrightarrow{\sigma} (\nu)zP'}
\end{array}$$

Definition We define the synchronization relation in the following way:

$$\begin{array}{c}
\text{COM1L} \frac{}{Sync(\bar{x}y, xy, \tau)} \qquad \text{COM2L} \frac{}{Sync(xy \cdot \sigma, \bar{x}y, \sigma)} \\
\\
\text{COM1R} \frac{}{Sync(xy, \bar{x}y, \tau)} \qquad \text{COM2R} \frac{}{Sync(\bar{x}y, xy \cdot \sigma, \sigma)}
\end{array}$$

This does not work because:

$$\begin{array}{c}
\text{OUT} \frac{}{\bar{a}z.P \xrightarrow{\bar{a}z} P} \\
\text{SINP} \frac{}{x(a).\bar{a}z.P \xrightarrow{xb \cdot \bar{a}z} P\{b/a\}}
\end{array}$$

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

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

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:

$$\begin{array}{c}
\text{Pref} \frac{\alpha \text{ not a strong prefix}}{\alpha.P \xrightarrow{\alpha} P} \qquad \text{LComSeq} \frac{P \xrightarrow{x(y) \cdot \sigma} P' \quad Q \xrightarrow{\bar{x}z} Q' \quad z \notin fn(\sigma) \cup fn(P)}{P|Q \xrightarrow{\sigma\{z/y\}} P'\{z/y\}|Q'} \\
\\
\text{SInp} \frac{P \xrightarrow{\sigma} P' \quad \sigma \neq \tau}{x(y).P \xrightarrow{x(y) \cdot \sigma} P'} \qquad \text{LComSng} \frac{P \xrightarrow{x(y)} P' \quad Q \xrightarrow{\bar{x}z} Q' \quad z \notin fn(P)}{P|Q \xrightarrow{\tau} P'\{z/y\}|Q'} \\
\\
\text{Sum} \frac{P \xrightarrow{\sigma} P'}{P + Q \xrightarrow{\sigma} P'} \qquad \text{Str} \frac{P \equiv P' \quad P' \xrightarrow{\alpha} Q' \quad Q \equiv Q'}{P \xrightarrow{\alpha} Q} \\
\\
\text{Res} \frac{P \xrightarrow{\sigma} P' \quad z \notin n(\alpha)}{(\nu)zP \xrightarrow{\sigma} (\nu)zP'} \qquad \text{Par} \frac{P \xrightarrow{\sigma} P' \quad bn(\sigma) \cup fn(Q) = \emptyset}{P|Q \xrightarrow{\sigma} P'|Q}
\end{array}$$

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, \dots, 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, \dots . A multi π process, in addition to the same actions of a π process, can perform also a strong prefix:

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

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

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

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:

$$\begin{aligned} bn(x(y)) &= \{y\} & fn(x(y)) &= \{x\} \\ bn(\bar{x}y) &= \emptyset & fn(\bar{x}y) &= \{x, y\} \end{aligned}$$

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:

$$\begin{array}{cc} \text{strong input} & \text{strong output} \\ \underline{xy} & \bar{xy} \end{array}$$

The set of actions is \mathbb{A}_m , we use $\alpha, \alpha_1, \alpha_2, \dots$ to range over the set of actions, we use $\sigma, \sigma_1, \sigma_2, \dots$ 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:

$$\begin{array}{ll}
\mathbf{Out} \frac{}{\overline{xy}.P \xrightarrow{\overline{xy}} P} & \mathbf{EInp} \frac{z \notin fn(P)}{x(y).P \xrightarrow{xz} P\{z/y\}} \\
\mathbf{SOut} \frac{}{\underline{\overline{xy}}.P \xrightarrow{\overline{xy}} P} & \\
\mathbf{Tau} \frac{}{\tau.P \xrightarrow{\tau} P} & \mathbf{SInp} \frac{}{\underline{x(y)}.P \xrightarrow{xz} P\{z/y\}} \\
\mathbf{Sum} \frac{P \xrightarrow{\sigma} P'}{P + Q \xrightarrow{\sigma} P'} & \mathbf{Str} \frac{P \equiv P' \quad P' \xrightarrow{\alpha} Q' \quad Q \equiv Q'}{P \xrightarrow{\alpha} Q} \\
\mathbf{Par} \frac{P \xrightarrow{\sigma} P' \quad bn(\sigma) \cap fn(Q) = \emptyset}{P|Q \xrightarrow{\sigma} P'|Q} & \mathbf{Com} \frac{P \xrightarrow{\sigma_1} P' \quad Q \xrightarrow{\sigma_2} Q' \quad Sync(\sigma_1, \sigma_2, \sigma_3)}{P|Q \xrightarrow{\sigma_3} P'|Q'} \\
\mathbf{Res} \frac{P \xrightarrow{\sigma} P' \quad z \notin n(\alpha)}{(\nu)zP \xrightarrow{\sigma} (\nu)zP'} & \mathbf{Seq} \frac{P \xrightarrow{\alpha} P' \quad P' \xrightarrow{\sigma} P''}{P \xrightarrow{\alpha \cdot \sigma} P''}
\end{array}$$

In this semantic we cannot have

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

nor

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

but we have

$$\mathbf{Seq} \frac{\mathbf{SOut} \frac{}{\underline{\overline{bz}}.P\{b/a\} \xrightarrow{\overline{bz}} P\{b/a\}} \quad \mathbf{SInp} \frac{}{\underline{x(a)}. \underline{\overline{az}}.P \xrightarrow{xb} (\underline{\overline{az}}.P)\{b/a\}}}{\underline{x(a)}. \underline{\overline{az}}.P \xrightarrow{xb \cdot \overline{bz}} P\{b/a\}}$$

Definition We define the synchronization relation in the following way:

$$\begin{array}{lll}
\mathbf{Com1L} \frac{}{Sync(xy, \overline{xy}, \tau)} & \mathbf{Com2L} \frac{}{Sync(xy \cdot \sigma, \overline{xy}, \sigma)} & \mathbf{Com3L} \frac{Sync(\sigma_1, \sigma_2, \sigma_3)}{Sync(xy \cdot \sigma_1, \overline{xy} \cdot \sigma_2, \sigma_3)} \\
\mathbf{Com1R} \frac{}{Sync(\overline{xy}, xy, \tau)} & \mathbf{Com2R} \frac{}{Sync(\overline{xy} \cdot \sigma, xy, \sigma)} & \mathbf{Com3R} \frac{Sync(\sigma_1, \sigma_2, \sigma_3)}{Sync(\overline{xy} \cdot \sigma_1, xy \cdot \sigma_2, \sigma_3)}
\end{array}$$

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:

$$\mathbf{Pref} \frac{\alpha \text{ not a strong prefix}}{\alpha.P \xrightarrow{\alpha} P}$$

$$\mathbf{Par} \frac{P \xrightarrow{\sigma} P' \quad bn(\sigma) \cap fn(Q) = \emptyset}{P|Q \xrightarrow{\sigma} P'|Q}$$

$$\mathbf{SOut} \frac{P \xrightarrow{\sigma} P' \quad \sigma \neq \tau}{\bar{x}y.P \xrightarrow{\bar{x}y.\sigma} P'}$$

$$\mathbf{LComSeq1} \frac{P \xrightarrow{x(y)} P' \quad Q \xrightarrow{\bar{x}z.\sigma} Q' \quad z \notin fn(P)}{P|Q \xrightarrow{\sigma} P'\{z/y\}|Q'}$$

$$\mathbf{Sum} \frac{P \xrightarrow{\sigma} P'}{P + Q \xrightarrow{\sigma} P'}$$

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

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

$$\mathbf{LComSng} \frac{P \xrightarrow{x(y)} P' \quad Q \xrightarrow{\bar{x}z} Q' \quad z \notin fn(P)}{P|Q \xrightarrow{\tau} P'\{z/y\}|Q'}$$

$$\mathbf{SInp} \frac{P \xrightarrow{\sigma} P' \quad \sigma \neq \tau}{\underline{x(y)}.P \xrightarrow{x(y).\sigma} P'}$$

$$\mathbf{LComSeq2} \frac{P \xrightarrow{\bar{x}z} P' \quad Q \xrightarrow{x(y).\sigma} Q' \quad z \notin fn(P)}{P|Q \xrightarrow{\sigma\{z/y\}} P'|Q'\{z/y\}}$$

Index

- α convertible, 11
- bn , 10
- application of a substitution, 11
- bind, 10
- binder, 10
- change of bound names, 10
- congruence, 11
- context, 11
- guarded, 10
- guarded occurrence, 11
- n, 10
- name occurrence
 - bound, 10
 - free, 10
- scope, 10
- structural congruence, 11
- substitution, 10
- sync
 - multipli
 - early, 24
- transition relation
 - early
 - without structural congruence, 13

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, Università 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 .