

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 Abstract

Il π calcolo e' un formalismo che descrive e analizza le proprieta' del calcolo concorrente. Nasce come proseguio del lavoro gia' svolto sul CCS (Calculus of Communicating Systems). L'aspetto appetibile del π calcolo rispetto ai formalismi precedenti e' l'essere in grado di descrivere la computazione concorrente in sistemi la cui configurazione puo' cambiare nel tempo. Nel CCS e nel π calcolo manca la possibilita' di modellare sequenze atomiche di azioni e di modellare la sincronizzazione multiparte. Il Multi CCS [3] estende il CCS con un'operatore di strong prefixing proprio per colmare tale vuoto. In questa tesi si cerca di trasportare per analogia le soluzioni introdotte dal Multi CCS verso il π calcolo. Il risultato finale e' un linguaggio chiamato Multi π calcolo.

In particolare il Multi π calcolo permette la sincronizzazione transazionale e la sincronizzazione multiparte. aggiungere una sintesi brevissima dei risultati ottenuti sul Multi π calcolo.

Contents

0.1	Abstract	3
1	TODO	7
2	Π calculus	9
2.1	Syntax	9
2.2	Operational Semantic(without structural congruence)	11
2.2.1	Early operational semantic(without structural congruence)	11
2.2.2	Late operational semantic(without structural congruence)	13
2.3	Structural congruence	14
2.4	Operational semantic with structural congruence	24
2.4.1	Early semantic with α conversion only	24
2.4.2	Early semantic with structural congruence	24
2.4.3	Late semantic with structural congruence	25
2.5	Equivalence of the semantics	26
2.5.1	Equivalence of the early semantics	26
2.5.2	Equivalence of the late semantics	37
2.6	Bisimilarity and Congruence	37
2.6.1	Bisimilarity	37
2.6.2	Congruence	38
2.6.3	Variants of Bisimilarity	38
3	Multi π calculus with strong output	39
3.1	Syntax	39
3.2	Operational semantic	39
3.2.1	Early operational semantic with structural congruence	39
3.2.2	Late operational semantic with structural congruence	41
4	Multi π calculus with strong input	43
4.1	Syntax	43
4.2	Operational semantic	43
4.2.1	Early operational semantic with structural congruence	43
4.2.2	Late operational semantic with structural congruence	44
4.2.3	low level semantic	45
5	Multi π calculus with strong input and output	47
5.1	Syntax	47
5.2	Operational semantic	47
5.2.1	Early operational semantic with structural congruence	47
5.2.2	Late operational semantic with structural congruence	47

Chapter 1

TODO

- dimostrare(o negare) l'equivalenza del pi calcolo con e senza congruenza strutturale
- nel multi pi calcolo con strong prefixing solo su input o solo su output: definire una semantica di basso livello sulla falsariga di quell'articolo
- fare un quadro generale sulle equivalenze nel pi calcolo
- scegliere una equivalenza(forse la open va bene) per multi pi calcolo(quale?) che sia una congruenza per input(ma non lo sara' per il parallelo)
- trovare equivalenza che sia una congruenza(es: open step) per tutti gli operatori
- trovare la congruenza coarsest contenuta nella bisimulazione scelta in precedenza
- riscrivere tutte le regole usando inferrule
- aggiungere la definizione precisa induttiva di alfa conversione anche per il multipli

Chapter 2

Π calculus

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

A coverage of π calculus is on [4], [5] and [6]

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:

$$P, Q ::= 0 \mid \pi.P \mid P|Q \mid P + Q \mid (\nu x)P \mid A(x_1, \dots, x_n)$$

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

for any action which is not a τ , the first name that appears in the action is called subject of the action and the second name is called object of the action.

$P + Q$ 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

$B(0, I) = \emptyset$	$B(Q + R, I) = B(Q, I) \cup B(R, I)$
$B(\bar{x}y.Q, I) = B(Q, I)$	$B(Q R, I) = B(Q, I) \cup B(R, I)$
$B(x(y).Q, I) = \{y, \bar{y}\} \cup B(Q, I)$	$B((\nu x)Q, I) = \{x, \bar{x}\} \cup B(Q, I)$
$B(\tau.Q, I) = B(Q, I)$	
$B(A(\tilde{x}), I) = \begin{cases} B(Q, I \cup \{A\}) \text{ where } A(\tilde{x}) \stackrel{\text{def}}{=} Q & \text{if } A \notin I \\ \emptyset & \text{if } A \in I \end{cases}$	

Table 2.1: Bound occurrences

$fn(\bar{x}y.Q) = \{x, \bar{x}, y, \bar{y}\} \cup fn(Q)$	$fn(Q + R) = fn(Q) \cup fn(R)$	$fn(0) = \emptyset$
$fn(x(y).Q) = \{x, \bar{x}\} \cup (fn(Q) - \{y, \bar{y}\})$	$fn(Q R) = fn(Q) \cup fn(R)$	
$fn((\nu x)Q) = fn(Q) - \{x, \bar{x}\}$	$fn(\tau.Q) = fn(Q)$	$fn(A(\tilde{x})) = \{\tilde{x}\}$

Table 2.2: Free occurrences

$(\nu z)P$ 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 processes.

$A(x_1, \dots, x_n)$ is an identifier. Every identifier has a definition

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

the x_i s must be pairwise different. The intuition is that we can substitute for some of the x_i s in P to get a π calculus process.

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

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

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

Definition 2.1.3. We say that a name x is free in P if P contains a non bound occurrence of x . We write $fn(P)$ for the set of names with a free occurrence in P . $fn(P)$ is defined in table 2.2

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

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

In a definition

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

the x_1, \dots, x_n are all the free names contained in P , specifically

$$fn(P) \subseteq \{x_1, \dots, x_n\}$$

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

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

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

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

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

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

if $a \in \{x_1, \dots, x_n\}$ then

$$(A(x_1, \dots, x_n \mid y_1, \dots, y_m))\{b/a\} = \begin{cases} A(x_1\{b/a\}, \dots, x_n\{b/a\} \mid y_1, \dots, y_m) & \text{if } b \notin \{y_1, \dots, y_m\} \\ A(x_1\{b/a\}, \dots, x_n\{b/a\} \mid y_1, \dots, y_{i-1}, c, y_{i+1}, \dots, y_m) & \text{if } b = y_i \\ \text{where } c \text{ is fresh} \end{cases}$$

if $a \notin \{x_1, \dots, x_n\}$ then

$$(A(x_1, \dots, x_n \mid y_1, \dots, y_m))\{b/a\} = A(x_1, \dots, x_n \mid y_1, \dots, y_m)$$

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

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

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

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

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

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

Table 2.3: Syntactic substitution

If we look at the definitions of bn and of fn we notice that if P contains another identifier whose definition is:

$$B(z_1, \dots, z_h) = Q$$

then we have

$$fn(Q) \subseteq \{x_1, \dots, x_n\}$$

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

$$\{\tilde{x}/\tilde{y}\} \text{ means } \{x_1/y_1, \dots, x_n/y_n\}$$

2.2 Operational Semantic(without structural congruence)

2.2.1 Early operational semantic(without structural congruence)

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

Out $\frac{}{\bar{x}y.P \xrightarrow{\bar{x}y} P}$	EInp $\frac{}{x(y).P \xrightarrow{xz} P\{z/y\}}$
SumR $\frac{Q \xrightarrow{\alpha} Q'}{P + Q \xrightarrow{\alpha} Q'}$	ParR $\frac{Q \xrightarrow{\alpha} Q' \quad bn(\alpha) \cap fn(Q) = \emptyset}{P Q \xrightarrow{\alpha} P Q'}$
SumL $\frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'}$	ParL $\frac{P \xrightarrow{\alpha} P' \quad bn(\alpha) \cap fn(Q) = \emptyset}{P Q \xrightarrow{\alpha} P' Q}$
Res $\frac{P \xrightarrow{\alpha} P' \quad z \notin n(\alpha)}{(\nu z)P \xrightarrow{\alpha} (\nu z)P'}$	ResAlp $\frac{(\nu w)P\{w/z\} \xrightarrow{xz} P' \quad w \notin n(P)}{(\nu z)P \xrightarrow{xz} P'}$
EComR $\frac{P \xrightarrow{\bar{x}y} P' \quad Q \xrightarrow{xy} Q'}{P Q \xrightarrow{\tau} P' Q'}$	ClsL $\frac{P \xrightarrow{\bar{x}(z)} P' \quad Q \xrightarrow{xz} Q' \quad z \notin fn(Q)}{P Q \xrightarrow{\tau} (\nu z)(P' Q')}$
EComL $\frac{P \xrightarrow{xy} P' \quad Q \xrightarrow{\bar{x}y} Q'}{P Q \xrightarrow{\tau} P' Q'}$	ClsR $\frac{P \xrightarrow{xz} P' \quad Q \xrightarrow{\bar{x}(z)} Q' \quad z \notin fn(P)}{P Q \xrightarrow{\tau} (\nu z)(P' Q')}$
Tau $\frac{}{\tau.P \xrightarrow{\tau} P}$	Cns $\frac{A(\tilde{x}) \stackrel{def}{=} P \quad P\{\tilde{w}/\tilde{x}\} \xrightarrow{\alpha} P'}{A(\tilde{x})\{\tilde{w}/\tilde{x}\} \xrightarrow{\alpha} P'}$
Opn $\frac{P \xrightarrow{\bar{x}z} P' \quad z \neq x}{(\nu z)P \xrightarrow{\bar{x}(z)} P'}$	OpnAlp $\frac{(\nu w)P\{w/z\} \xrightarrow{\bar{x}(w)} P' \quad w \notin n(P) \quad x \neq w \neq z.}{(\nu z)P \xrightarrow{\bar{x}(w)} P'}$

Table 2.4: Early transition relation without structural congruence

- the nodes are π calculus process. The set of node is \mathbb{P}
- the actions can be:
 - unbound input xy
 - unbound output $\bar{x}y$
 - the silent action τ
 - 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 2.2.1. *The early transition relation $\rightarrow \subseteq \mathbb{P} \times \mathbb{A} \times \mathbb{P}$ is the smallest relation induced by the rules in table 2.4. Where with \tilde{x} we mean a sequence of names x_1, \dots, x_n .*

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.

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

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 $b \notin bn(P)$

$$\text{RESALP} \frac{\text{RES} \frac{\text{EINP} \frac{}{(a(x).P)\{c/b\} \xrightarrow{ab} P\{c/b\}\{b/x\}} \quad c \notin n(a(b))}{(\nu c)((a(x).P)\{c/b\}) \xrightarrow{ab} (\nu c)(P\{c/b\}\{b/x\})} \quad b \notin n((a(x).P)\{c/b\})}{(\nu b)a(x).P \xrightarrow{ab} (\nu c)P\{c/b\}\{b/x\}}$$

$$\text{EComL} \frac{(\nu b)a(x).P \xrightarrow{ab} (\nu c)P\{c/b\}\{b/x\} \quad \text{EOUT} \frac{}{\bar{a}b.Q \xrightarrow{\bar{a}b} Q}}{((\nu b)a(x).P) \mid \bar{a}b.Q \xrightarrow{\tau} ((\nu c)(P\{c/b\}\{b/x\})) \mid Q}$$

Example We have to spend some time to deal with the change of bound names in an identifier. Suppose we have

$$A(x) \stackrel{def}{=} \underbrace{x(y).x(a).0}_P$$

From the definition of substitution it follows that

$$A(x)\{y/x\} = A(y)$$

The identifier $A(y)$ is expected to behave consistently with

$$P\{y/x\} = y(z).y(a).0$$

so we have to prove

$$A(y) \xrightarrow{yw} y(a).0$$

We can prove this in the following way:

$$\text{CNS} \frac{A(x) \stackrel{def}{=} P \quad \text{EINP} \frac{}{P\{y/x\} \xrightarrow{yw} y(a).0}}{A(y) \xrightarrow{yw} y(a).0}$$

2.2.2 Late operational semantic(without structural congruence)

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

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

Definition 2.2.2. The late transition relation without structural congruence $\rightarrow_{\subseteq} \mathbb{P} \times \mathbb{A} \times \mathbb{P}$ is the smallest relation induced by the rules in table 2.5. *TUTTE LE SEMANTICHE LATE DEL PI CALCOLO SONO DA AGGIORNARE!!!! !!! !! !*

LInp $\frac{z \notin fn(P)}{x(y).P \xrightarrow{x(z)} P\{z/y\}}$	Res $\frac{P \xrightarrow{\alpha} P' \quad z \notin n(\alpha)}{(\nu z)P \xrightarrow{\alpha} (\nu z)P'}$
SumL $\frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'}$	SumR $\frac{Q \xrightarrow{\alpha} Q'}{P + Q \xrightarrow{\alpha} Q'}$
ParL $\frac{P \xrightarrow{\alpha} P' \quad bn(\alpha) \cap fn(Q) = \emptyset}{P Q \xrightarrow{\alpha} P' Q}$	ParR $\frac{Q \xrightarrow{\alpha} Q' \quad bn(\alpha) \cap fn(Q) = \emptyset}{P Q \xrightarrow{\alpha} P Q'}$
ComL $\frac{P \xrightarrow{x(y)} P' \quad Q \xrightarrow{\bar{x}(z)} Q'}{P Q \xrightarrow{\tau} P'\{z/y\} Q'}$	ComR $\frac{P \xrightarrow{\bar{x}(z)} P' \quad Q \xrightarrow{x(y)} Q'}{P Q \xrightarrow{\tau} P' Q'\{z/y\}}$
Opn $\frac{P \xrightarrow{\bar{x}z} P' \quad z \neq x}{(\nu z)P \xrightarrow{\bar{x}(z)} P'}$	Out $\frac{}{\bar{x}y.P \xrightarrow{\bar{x}y} P}$
ClsL $\frac{P \xrightarrow{\bar{x}(z)} P' \quad Q \xrightarrow{xz} Q' \quad z \notin fn(Q)}{P Q \xrightarrow{\tau} (\nu z)(P' Q')}$	ClsR $\frac{P \xrightarrow{xz} P' \quad Q \xrightarrow{\bar{x}(z)} Q' \quad z \notin fn(P)}{P Q \xrightarrow{\tau} (\nu z)(P' Q')}$
Tau $\frac{}{\tau.P \xrightarrow{\tau} P}$	Cns $\frac{A(\tilde{x}) \stackrel{def}{=} P \quad P\{\tilde{y}/\tilde{x}\} \xrightarrow{\alpha} P'}{A(\tilde{y}) \xrightarrow{\alpha} P'}$

Table 2.5: Late semantic without structural congruence

2.3 Structural congruence

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

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

Definition 2.3.2. 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 2.3.3. A congruence is a binary relation on processes such that:

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

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

Definition 2.3.4. Processes P and Q are α convertible or α equivalent if Q can be obtained from P by a finite number of changes of bound names. If P and Q are α equivalent then we write $P \equiv_{\alpha} Q$. Specifically the α equivalence is the smallest binary relation on processes that satisfies the laws in table 2.6

$\text{ALPSUM} \frac{P_1 \equiv_{\alpha} Q_1 \quad P_2 \equiv_{\alpha} Q_2}{P_1 + P_2 \equiv_{\alpha} Q_1 + Q_2}$	$\text{ALPTAU} \frac{P \equiv_{\alpha} Q}{\tau.P \equiv_{\alpha} \tau.Q}$
$\text{ALPRES1} \frac{P\{y/x\} \equiv_{\alpha} Q \quad x \neq y \quad y \notin \text{fn}(P)}{(\nu x)P \equiv_{\alpha} (\nu y)Q}$	$\text{ALPRES} \frac{P \equiv_{\alpha} Q}{(\nu x)P \equiv_{\alpha} (\nu x)Q}$
$\text{ALPINP1} \frac{P\{y/x\} \equiv_{\alpha} Q \quad x \neq y \quad y \notin \text{fn}(P)}{z(x).P \equiv_{\alpha} z(y).Q}$	$\text{ALPINP} \frac{P \equiv_{\alpha} Q}{x(y).P \equiv_{\alpha} x(y).Q}$
$\text{ALPPAR} \frac{P_1 \equiv_{\alpha} Q_1 \quad P_2 \equiv_{\alpha} Q_2}{P_1 P_2 \equiv_{\alpha} Q_1 Q_2}$	$\text{ALPOUT} \frac{P \equiv_{\alpha} Q}{\bar{x}y.P \equiv_{\alpha} \bar{x}y.Q}$
$\text{ALPIDE} \frac{}{A(\tilde{x} \tilde{y}) \equiv_{\alpha} A(\tilde{x} \tilde{y})}$	$\text{ALPZERO} \frac{}{0 \equiv_{\alpha} 0}$

Table 2.6: α equivalence laws

It remains the problem of proving that α equivalence is well defined, i.e. if we change only some bound names in a process P then we get a process α equivalent to P .

Lemma 2.3.1. *Inversion lemma for α equivalence*

- If $P \equiv_{\alpha} 0$ then P is also the null process 0
- If $P \equiv_{\alpha} \tau.Q_1$ then $P = \tau.P_1$ for some P_1 such that $P_1 \equiv_{\alpha} Q_1$
- If $P \equiv_{\alpha} \bar{x}y.Q_1$ then $P = \bar{x}y.P_1$ for some P_1 such that $P_1 \equiv_{\alpha} Q_1$
- If $P \equiv_{\alpha} z(y).Q_1$ then one and only one of the following cases holds:
 - $P = z(x).P_1$ for some P_1 such that $P_1\{y/x\} \equiv_{\alpha} Q_1$
 - $P = z(y).P_1$ for some P_1 such that $P_1 \equiv_{\alpha} Q_1$
- If $P \equiv_{\alpha} Q_1 + Q_2$ then $P = P_1 + P_2$ for some P_1 and P_2 such that $P_1 \equiv_{\alpha} Q_1$ and $P_2 \equiv_{\alpha} Q_2$.
- If $P \equiv_{\alpha} Q_1 | Q_2$ then $P = P_1 | P_2$ for some P_1 and P_2 such that $P_1 \equiv_{\alpha} Q_1$ and $P_2 \equiv_{\alpha} Q_2$.
- If $P \equiv_{\alpha} (\nu y)Q_1$ then one and only one of the following cases holds:
 - $P = (\nu x)P_1$ such that $P_1\{y/x\} \equiv_{\alpha} Q_1$
 - $P = (\nu y).P_1$ for some P_1 such that $P_1 \equiv_{\alpha} Q_1$
- If $P \equiv_{\alpha} A(\tilde{x})$ then P is Q .

Proof. This lemma works because given Q we know which rules must be at the end of any proof tree of $P \equiv_{\alpha} Q$. \square

Definition 2.3.5. *We define a structural congruence \equiv as the smallest congruence on processes that satisfies the axioms in table 2.7*

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 the rule *Cns* in table 2.4.

SC-ALP	$\frac{P \equiv_{\alpha} Q}{P \equiv Q}$	α conversion
abelian monoid laws for sum:		
SC-SUM-ASC	$M_1 + (M_2 + M_3) \equiv (M_1 + M_2) + M_3$	associativity
SC-SUM-COM	$M_1 + M_2 \equiv M_2 + M_1$	commutativity
SC-SUM-INC	$M + 0 \equiv M$	zero element
abelian monoid laws for parallel:		
SC-COM-ASC	$P_1 (P_2 P_3) \equiv (P_1 P_2) P_3$	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	$A(\tilde{w}) \equiv P\{\tilde{w}/\tilde{x}\}$	if $A(\tilde{x}) \stackrel{def}{=} P$

Table 2.7: Structural congruence axioms

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 π calculus we can use the structural congruence with the rule SC-ALP or we can embed the α conversion in the SOS rules. In the early case, the rule for input and the rules *ResAlp*, *OpnAlp*, *Cns* take care of α conversion, whether in the late case the rule for communication and the rules *ResAlp*, *OpnAlp*, *Cns* are in charge for α conversion.

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:

$$\text{SumL} \frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'} \quad \text{SumR} \frac{Q \xrightarrow{\alpha} Q'}{P + Q \xrightarrow{\alpha} Q'}$$

or by the following rule and axiom:

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

and the rule *Str*

scope extension We can use the scope extension laws in table 2.7 or the rules *Opn* and *Cls* in table 2.4 to deal with the scope extension.

Lemma 2.3.2.

$$a \in fn(Q) \Rightarrow fn(Q\{b/a\}) = (fn(Q) - \{a\}) \cup \{b\}$$

Proof.

□

Lemma 2.3.3. $P \equiv_{\alpha} Q \Rightarrow fn(P) = fn(Q)$

Proof. The proof goes by induction on rules

AlpZero the lemma holds because P and Q are the same process.

AlpTau :

$$\begin{array}{ll}
P \equiv_{\alpha} Q & \text{rule premise} \\
\Rightarrow fn(P) = fn(Q) & \text{inductive hypothesis} \\
\Rightarrow fn(\tau.P) = fn(\tau.Q) & \text{definition of } fn
\end{array}$$

AlpOut :

$$\begin{array}{ll}
P \equiv_{\alpha} Q & \text{rule premise} \\
\Rightarrow fn(P) = fn(Q) & \text{inductive hypothesis} \\
\Rightarrow fn(P) \cup \{x, y\} = fn(Q) \cup \{x, y\} & \text{definition of } fn \\
\Rightarrow fn(\bar{x}y.P) = fn(\bar{x}y.Q) &
\end{array}$$

AlpRes1 : we consider two cases:

$x \notin fn(P)$:

$$\begin{array}{ll}
P\{y/x\} \equiv_{\alpha} Q \text{ and } y \notin fn(P) & \text{rule premises} \\
\Rightarrow fn(P\{y/x\}) = fn(Q) & \text{inductive hypothesis} \\
\Rightarrow fn(P) = fn(Q) & x \notin fn(P) \text{ and def of substitution} \\
\Rightarrow fn(P) = fn(Q) \text{ and } y \notin fn(Q) & y \notin fn(P)
\end{array}$$

Since $x \notin fn(P)$ then $fn(P) = fn(P) - \{x\}$. Since $y \notin fn(Q)$ then $fn(Q) = fn(Q) - \{y\}$. From $fn(P) = fn(P) - \{x\}$, $fn(Q) = fn(Q) - \{y\}$, $fn(P) = fn(Q)$ and the definition of substitution it follows that $fn((\nu x)P) = fn((\nu y)Q)$

$x \in fn(P)$:

$$\begin{array}{ll}
P\{y/x\} \equiv_{\alpha} Q & \text{rule premise} \\
\Rightarrow fn(P\{y/x\}) = fn(Q) & \text{inductive hypothesis} \\
\Rightarrow fn(P\{y/x\}) - \{y\} = fn(Q) - \{y\} & \\
\Rightarrow (fn(P) - \{x\} \cup \{y\}) - \{y\} = fn(Q) - \{y\} & \\
\Rightarrow fn(P) - \{x\} = fn(Q) - \{y\} & \text{definition of } fn \\
\Rightarrow fn((\nu x)P) = fn((\nu y)Q) &
\end{array}$$

AlpInp1 : we consider two cases:

$x \notin fn(P)$:

$$\begin{array}{ll}
P\{y/x\} \equiv_{\alpha} Q \text{ and } y \notin fn(P) & \text{rule premises} \\
\Rightarrow fn(P\{y/x\}) = fn(Q) & \text{inductive hypothesis} \\
\Rightarrow fn(P) = fn(Q) & x \notin fn(P) \text{ and def of substitution} \\
\Rightarrow fn(P) = fn(Q) \text{ and } y \notin fn(Q) & y \notin fn(P)
\end{array}$$

Since $x \notin fn(P)$ then $fn(P) = fn(P) - \{x\}$. Since $y \notin fn(Q)$ then $fn(Q) = fn(Q) - \{y\}$. From $fn(P) = fn(P) - \{x\}$, $fn(Q) = fn(Q) - \{y\}$ and $fn(P) = fn(Q)$ it follows that $fn(P) - \{x\} = fn(Q) - \{y\}$ and so $(fn(P) - \{x\}) \cup \{z\} = (fn(Q) - \{y\}) \cup \{z\}$ which gives $fn(z(x).P) = fn(z(y).Q)$.

$x \in fn(P)$:

$$\begin{array}{ll}
P\{y/x\} \equiv_{\alpha} Q & \text{rule premise} \\
\Rightarrow fn(P\{y/x\}) = fn(Q) & \text{inductive hypothesis} \\
\Rightarrow fn(P) - \{y\} = fn(Q) - \{y\} & \text{lemma 2.3.2} \\
\Rightarrow (fn(P) - \{x\} \cup \{y\}) - \{y\} = fn(Q) - \{y\} \\
\Rightarrow fn(P) - \{x\} = fn(Q) - \{y\} \\
\Rightarrow (fn(P) - \{x\}) \cup \{z\} = (fn(Q) - \{y\}) \cup \{z\} & \text{definition of } fn \\
\Rightarrow fn(z(x).P) = fn(z(y).Q)
\end{array}$$

AlpSum :

$$\begin{array}{ll}
P_1 \equiv_{\alpha} Q_1 \text{ and } P_2 \equiv_{\alpha} Q_2 & \text{rule premises} \\
\Rightarrow fn(P_1) = fn(Q_1) \text{ and } fn(P_2) = fn(Q_2) & \text{inductive hypothesis} \\
\Rightarrow fn(P_1) \cup fn(P_2) = fn(Q_1) \cap fn(Q_2) & \text{definition of } fn \\
\Rightarrow fn(P_1 + P_2) = fn(Q_1 + Q_2)
\end{array}$$

AlpPar :

$$\begin{array}{ll}
P_1 \equiv_{\alpha} Q_1 \text{ and } P_2 \equiv_{\alpha} Q_2 & \text{rule premises} \\
\Rightarrow fn(P_1) = fn(Q_1) \text{ and } fn(P_2) = fn(Q_2) & \text{inductive hypothesis} \\
\Rightarrow fn(P_1) \cup fn(P_2) = fn(Q_1) \cap fn(Q_2) & \text{definition of } fn \\
\Rightarrow fn(P_1 | P_2) = fn(Q_1 | Q_2)
\end{array}$$

AlpRes :

$$\begin{array}{ll}
P \equiv_{\alpha} Q & \text{rule premise} \\
\Rightarrow fn(P) = fn(Q) & \text{inductive hypothesis} \\
\Rightarrow fn(P) - \{x\} = fn(Q) - \{x\} & \text{definition of } fn \\
\Rightarrow fn((\nu x)P) = fn((\nu x)Q)
\end{array}$$

AlpInp :

$$\begin{array}{ll}
P \equiv_{\alpha} Q\{x/y\} & \text{rule premise} \\
\Rightarrow fn(P) = fn(Q) & \text{inductive hypothesis} \\
\Rightarrow (fn(P) - \{y\}) \cup \{x\} = (fn(Q) - \{y\}) \cup \{x\} & \text{definition of } fn \\
\Rightarrow fn(x(y).P) = fn(x(y).Q)
\end{array}$$

AlpIde the lemma holds because P and Q are the same process.

□

Lemma 2.3.4. $x \notin fn(P) \Rightarrow P\{x/y\}\{b/a\} \equiv_{\alpha} P\{b/a\}\{x/y\}$

Lemma 2.3.5. α equivalence is invariant with respect to substitution. In other words

$$\begin{array}{l}
P \equiv_{\alpha} Q \\
b \notin fn(P) \Rightarrow P\{b/a\} \equiv_{\alpha} Q\{b/a\} \\
b \notin fn(Q)
\end{array}$$

Proof. : If a and b are the same name then the substitution has no effect and the lemma holds. Otherwise:

$$\begin{array}{ll}
P \equiv_{\alpha} Q & \text{lemma hypothesis} \\
\Rightarrow fn(P) = fn(Q) & \text{lemma 2.3.3} \\
\Rightarrow a \notin fn(P) \wedge a \notin fn(Q) \text{ or } a \in fn(P) \wedge a \in fn(Q)
\end{array}$$

In the former case a is not a free name in P and Q so the substitutions have no effects and the lemma holds. In the latter case a is a free names in both processes: the proof goes by induction on the length of the proof tree of $P \equiv_\alpha Q$ and then by cases on the last rule of the proof tree. Let x, y, a and b be pairwise different.

base case The length of the proof is one and the rule used can be only: *AlpZero* or *AlpIde*: the lemma holds because P and Q are syntactically the same process.

inductive case :

AlpTau :

$$\begin{array}{ll}
P_1 \equiv_\alpha Q_1 & \text{rule premise} \\
\Rightarrow P_1\{b/a\} \equiv_\alpha Q_1\{b/a\} & \text{inductive hypothesis} \\
\Rightarrow \tau.(P_1\{b/a\}) \equiv_\alpha \tau.(Q_1\{b/a\}) & \text{rule AlpTau} \\
\Rightarrow (\tau.P_1)\{b/a\} \equiv_\alpha (\tau.Q_1)\{b/a\} & \text{definition of substitution}
\end{array}$$

AlpSum :

$$\begin{array}{ll}
P_1 \equiv Q_1 \text{ and } P_2 \equiv Q_2 & \text{rule premises} \\
\Rightarrow P_1\{b/a\} \equiv Q_1\{b/a\} \text{ and } P_2\{b/a\} \equiv Q_2\{b/a\} & \text{inductive hypothesis} \\
\Rightarrow P_1\{b/a\} + P_2\{b/a\} \equiv Q_1\{b/a\} + Q_2\{b/a\} & \text{rule AlpSum} \\
\Rightarrow (P_1 + P_2)\{b/a\} \equiv_\alpha (Q_1 + Q_2)\{b/a\} & \text{definition of substitution}
\end{array}$$

AlpPar : this case is very similar to the previous one.

AlpOut :

$$\begin{array}{ll}
P_1 \equiv_\alpha Q_1 & \text{rule premise} \\
\Rightarrow P_1\{b/a\} \equiv_\alpha Q_1\{b/a\} & \text{inductive hypothesis} \\
\Rightarrow \bar{x}\{b/a\}y\{b/a\}.P_1\{b/a\} \equiv_\alpha \bar{x}\{b/a\}y\{b/a\}.Q_1\{b/a\} & \text{rule AlpOut} \\
\Rightarrow (\bar{x}y.P_1)\{b/a\} \equiv_\alpha (\bar{x}y.Q_1)\{b/a\} & \text{definition of substitution}
\end{array}$$

AlpInp :

$$\begin{array}{ll}
P_1 \equiv_\alpha Q_1 & \text{rule premise} \\
\Rightarrow P_1\{b/a\} \equiv_\alpha Q_1\{b/a\} & \text{inductive hypothesis} \\
\Rightarrow x\{b/a\}(y).P_1\{b/a\} \equiv_\alpha x\{b/a\}(y).Q_1\{b/a\} & \text{rule AlpInp} \\
\Rightarrow (x(y).P_1)\{b/a\} \equiv_\alpha (x(y).Q_1)\{b/a\} & \text{definition of substitution}
\end{array}$$

$$\begin{array}{ll}
P_1 \equiv_\alpha Q_1 & \text{rule premise} \\
\Rightarrow b(a).P_1 \equiv_\alpha b(a).Q_1 & \text{rule AlpIn} \\
\Rightarrow a\{b/a\}(a).P_1 \equiv_\alpha a\{b/a\}(a).Q_1 & \text{definition of substitution} \\
\Rightarrow (a(a).P_1)\{b/a\} \equiv_\alpha (a(a).Q_1)\{b/a\} & \text{definition of substitution}
\end{array}$$

$$\begin{array}{ll}
P_1 \equiv_\alpha Q_1 & \text{rule premise} \\
\Rightarrow P_1\{b/a\} \equiv_\alpha Q_1\{b/a\} & \text{inductive hypothesis} \\
\Rightarrow b\{b/a\}(x).(P_1\{b/a\}) \equiv_\alpha b\{b/a\}(x).(Q_1\{b/a\}) & \text{rule AlpIn} \\
\Rightarrow (b(x).P_1)\{b/a\} \equiv_\alpha (b(x).Q_1)\{b/a\} & \text{definition of substitution}
\end{array}$$

AlpInp1 : we have various cases:

- the last part of the proof tree of $P \equiv_\alpha Q$ is

$$\text{ALPInp1} \frac{P_1 \equiv_\alpha Q_1\{x/y\} \quad x \neq y \quad x \notin fn(Q_1)}{\underbrace{z(x).P_1}_P \equiv_\alpha \underbrace{z(y).Q_1}_Q}$$

$P_1 \equiv_\alpha Q_1\{x/y\}$ and $x \notin fn(Q_1)$	rule premise
$\Rightarrow P_1\{b/a\} \equiv_\alpha Q_1\{x/y\}\{b/a\}$	inductive hypothesis
$\Rightarrow P_1\{b/a\} \equiv_\alpha Q_1\{b/a\}\{x/y\}$	transitivity and lemma 2.3.4
$\Rightarrow z(x).(P_1\{b/a\}) \equiv_\alpha z(y).(Q_1\{b/a\})$	rule <i>AlpInp1</i>
$\Rightarrow (z(x).P_1)\{b/a\} \equiv_\alpha (z(y).Q_1)\{b/a\}$	definition of substitution

- the last part of the proof tree of $P \equiv_\alpha Q$ is

$$\text{ALP-INP1} \frac{P_1 \equiv_\alpha Q_1\{x/y\} \quad x \neq y \quad x \notin fn(Q_1)}{\underbrace{b(x).P_1}_P \equiv_\alpha \underbrace{b(y).Q_1}_Q}$$

$P_1 \equiv_\alpha Q_1\{x/y\}$ and $x \notin fn(Q_1)$	rule premise
$\Rightarrow P_1\{b/a\} \equiv_\alpha Q_1\{x/y\}\{b/a\}$	inductive hypothesis
$\Rightarrow P_1\{b/a\} \equiv_\alpha Q_1\{b/a\}\{x/y\}$	transitivity and lemma 2.3.4
$\Rightarrow b(x).(P_1\{b/a\}) \equiv_\alpha b(y).(Q_1\{b/a\})$	rule <i>AlpInp1</i>
$\Rightarrow (b(x).P_1)\{b/a\} \equiv_\alpha (b(y).Q_1)\{b/a\}$	definition of substitution

- the last part of the proof tree of $P \equiv_\alpha Q$ is

$$\text{ALP-INP1} \frac{P_1 \equiv_\alpha Q_1\{x/y\} \quad x \neq y \quad x \notin fn(Q_1)}{\underbrace{a(x).P_1}_P \equiv_\alpha \underbrace{a(y).Q_1}_Q}$$

$P_1 \equiv_\alpha Q_1\{x/y\}$ and $x \notin fn(Q_1)$	rule premise
$\Rightarrow P_1\{b/a\} \equiv_\alpha Q_1\{x/y\}\{b/a\}$	inductive hypothesis
$\Rightarrow P_1\{b/a\} \equiv_\alpha Q_1\{b/a\}\{x/y\}$	transitivity and lemma 2.3.4
$\Rightarrow a(x).(P_1\{b/a\}) \equiv_\alpha a(y).(Q_1\{b/a\})$	rule <i>AlpInp1</i>
$\Rightarrow (a(x).P_1)\{b/a\} \equiv_\alpha (a(y).Q_1)\{b/a\}$	definition of substitution

- the last part of the proof tree of $P \equiv_\alpha Q$ is

$$\text{ALP-INP1} \frac{P_1 \equiv_\alpha Q_1\{a/y\} \quad a \neq y \quad a \notin fn(Q_1)}{\underbrace{a(a).P_1}_P \equiv_\alpha \underbrace{a(y).Q_1}_Q}$$

$P_1 \equiv_\alpha Q_1\{a/y\}$ and $x \notin fn(Q_1)$	rule premise
$\Rightarrow P_1\{b/a\} \equiv_\alpha Q_1\{a/y\}\{b/a\}$	inductive hypothesis
$\Rightarrow P_1\{b/a\} \equiv_\alpha Q_1\{b/a\}\{a/y\}$	transitivity and lemma 2.3.4
$\Rightarrow a(a).(P_1\{b/a\}) \equiv_\alpha a(y).(Q_1\{b/a\})$	rule <i>AlpInp1</i>
$\Rightarrow (a(a).P_1)\{b/a\} \equiv_\alpha (a(y).Q_1)\{b/a\}$	definition of substitution

- the last part of the proof tree of $P \equiv_\alpha Q$ is

$$\text{ALP-INP1} \frac{P_1 \equiv_\alpha Q_1\{x/a\} \quad x \neq a \quad x \notin fn(Q_1)}{\underbrace{a(x).P_1}_P \equiv_\alpha \underbrace{a(a).Q_1}_Q}$$

$P_1 \equiv_\alpha Q_1\{x/a\}$ and $x \notin fn(Q_1)$	rule premise
$\Rightarrow P_1\{b/a\} \equiv_\alpha Q_1\{x/a\}\{b/a\}$	inductive hypothesis
$\Rightarrow P_1\{b/a\} \equiv_\alpha Q_1\{b/a\}\{x/a\}$	transitivity and lemma 2.3.4
$\Rightarrow a(x).(P_1\{b/a\}) \equiv_\alpha a(a).(Q_1\{b/a\})$	rule <i>AlpInp1</i>
$\Rightarrow (a(x).P_1)\{b/a\} \equiv_\alpha (a(a).Q_1)\{b/a\}$	definition of substitution

- mancano $x \neq y$ e $x \neq x$

AlpRes :

$$\begin{array}{ll}
P_1 \equiv_\alpha Q_1 & \text{rule premise} \\
\Rightarrow P_1\{b/a\} \equiv_\alpha Q_1\{b/a\} & \text{inductive hypothesis} \\
\Rightarrow (\nu x)(P_1\{b/a\}) \equiv_\alpha (\nu x)(Q_1\{b/a\}) & \text{rule } AlpRes \\
\Rightarrow ((\nu x)P_1)\{b/a\} \equiv_\alpha ((\nu x)Q_1)\{b/a\} & \text{definition of substitution}
\end{array}$$

AlpRes1 :

$$ALPRES1 \frac{P_1 \equiv_\alpha Q_1\{x/y\} \quad x \neq y \quad x \notin fn(Q_1)}{\underbrace{(\nu x)P_1}_P \equiv_\alpha \underbrace{(\nu y)Q_1}_Q}$$

$$\begin{array}{ll}
P_1 \equiv_\alpha Q_1\{x/y\} \text{ and } x \neq y \text{ and } x \notin fn(Q_1) & \text{rule premises} \\
P_1\{b/a\} \equiv_\alpha Q_1\{x/y\}\{b/a\} & \text{inductive hypothesis} \\
P_1\{b/a\} \equiv_\alpha Q_1\{b/a\}\{x/y\} & \text{lemma 2.3.4 and transitivity} \\
(\nu x)(P_1\{b/a\}) \equiv_\alpha (\nu y)(Q_1\{b/a\}) & \text{rule } AlpRes1 \\
((\nu x)P_1)\{b/a\} \equiv_\alpha ((\nu y)Q_1)\{b/a\} & \text{definition of substitution}
\end{array}$$

□

Lemma 2.3.6.

$$P \equiv_\alpha P\{x/y\}\{y/x\}$$

esistono delle precondizioni per le quali il lemma e' vero? esistono delle precondizioni per le quali si puo' addirittura avere l'uguaglianza sintattica?

In the proof of equivalence of the semantics in the next section we need the following lemmas

Lemma 2.3.7. $P\{x/y\} \equiv_\alpha Q$ if and only if $P \equiv_\alpha Q\{y/x\}$.

NON FUNZIONA LA DIMOSTRAZIONE! staro' forse esagerando?

Proof. The proof is an induction on the length of the proof tree of $P\{x/y\} \equiv_\alpha Q$ and then by cases on the last rule:

base case the last rule can be

AlpZero in this case both P and Q are the null process 0 so the thesis holds.

AlpIde for this rule to apply $P\{x/y\}$ and Q must be some identifier A with the same variable.

Suppose that $P = A(\tilde{a}|\tilde{b})$ There can be some different cases:

$y \in \tilde{a}$ we can suppose that $\tilde{a} = y, \tilde{c}$ then

$x \in \tilde{b}$ we can suppose that $\tilde{b} = x, \tilde{d}$, then

$$Q = P\{x/y\} = A(x, \tilde{c}|z, \tilde{d})$$

where z is a fresh name. We need now the identifier equal to $Q\{y/x\} = A(x, \tilde{c}|z, \tilde{d})\{y/x\}$ so we have to distinguish two cases:

$x \in \text{tilded}$

$x \notin \text{tilded}$

$$Q\{y/x\} = A(x, \tilde{c}|z, \tilde{d})\{y/x\} = A(y, \tilde{c}|z, \tilde{d})$$

$y \notin \tilde{y}$ in this case there is no need to change bound names so

$$Q\{y/x\} = A(y, \tilde{z}|\tilde{y})$$

$x \notin \tilde{x}$ then

$$Q\{y/x\} = Q = A(\tilde{x}|\tilde{y})$$

□

Lemma 2.3.8. *The α equivalence is an equivalence relation.*

Proof. :

reflexivity We prove $P \equiv_\alpha P$ by structural induction on P :

0 :

$$\text{ALPZERO} \frac{}{0 \equiv_\alpha 0}$$

$\tau.P_1$: for induction $P_1 \equiv_\alpha P_1$ so

$$\text{ALPTAU} \frac{P_1 \equiv_\alpha P_1}{\tau.P_1 \equiv_\alpha \tau.P_1}$$

$x(y).P_1$: for induction $P_1 \equiv_\alpha P_1$ so

$$\text{ALPINP} \frac{P_1 \equiv_\alpha P_1}{x(y).P_1 \equiv_\alpha x(y).P_1}$$

$\bar{x}y.P_1$: for induction $P_1 \equiv_\alpha P_1$ so

$$\text{ALPOUT} \frac{P_1 \equiv_\alpha P_1}{\bar{x}y.P_1 \equiv_\alpha \bar{x}y.P_1}$$

$P_1 + P_2$: for induction $P_1 \equiv_\alpha P_1$ and $P_2 \equiv_\alpha P_2$ so

$$\text{ALPSUM} \frac{P_1 \equiv_\alpha P_1 \quad P_2 \equiv_\alpha P_2}{P_1 + P_2 \equiv_\alpha P_1 + P_2}$$

$P_1 | P_2$: for induction $P_1 \equiv_\alpha P_1$ and $P_2 \equiv_\alpha P_2$ so

$$\text{ALPPAR} \frac{P_1 \equiv_\alpha P_1 \quad P_2 \equiv_\alpha P_2}{P_1 | P_2 \equiv_\alpha P_1 | P_2}$$

$(\nu x)P_1$: for induction $P_1 \equiv_\alpha P_1$ so

$$\text{ALPRES} \frac{P_1 \equiv_\alpha P_1}{(\nu x)P_1 \equiv_\alpha (\nu x)P_1}$$

$A(\tilde{x}|\tilde{y})$:

$$\text{ALPIDE} \frac{}{A(\tilde{x}|\tilde{y}) \equiv_\alpha A(\tilde{x}|\tilde{y})}$$

symmetry A proof of

$$P \equiv_\alpha Q \Rightarrow Q \equiv_\alpha P$$

can go by induction on the length of the proof tree of $P \equiv_\alpha Q$ and then by cases on the last rule used. Nevertheless we notice that the base case rules *AlpZero* and *AlpIde* are symmetric and the inductive case rules are symmetric except for *AlpRes1* and *AlpInp1*. So we provide with the cases for those last two rules:

AlpRes1 the last part of the proof tree is

$$\text{ALPRES1} \frac{P\{y/x\} \equiv_\alpha Q \quad x \neq y \quad y \notin fn(P)}{(\nu x)P \equiv_\alpha (\nu y)Q}$$

we apply the inductive hypothesis on $P\{y/x\} \equiv_\alpha Q$ and get $Q \equiv_\alpha P\{y/x\}$ which implies $Q\{x/y\} \equiv_\alpha P$

DA DIMOSTRARE $Q \equiv_{\alpha} P\{y/x\}$ and $y \notin fn(P)$ implies $Q\{x/y\} \equiv_{\alpha} P$ and $x \notin fn(Q)$

so an application of the same rule yields:

$$\text{ALPRES1} \frac{Q\{x/y\} \equiv_{\alpha} P \quad x \neq y \quad x \notin fn(Q)}{(\nu y)QP \equiv_{\alpha} (\nu x)}$$

AlpInp1 this is very similar to the previous.

transitivity suppose

$$P \equiv_{\alpha} Q \text{ and } Q \equiv_{\alpha} R$$

we prove the thesis $P \equiv_{\alpha} R$ by induction on the length of the proof tree of $P \equiv_{\alpha} Q$. If the tree has only one node then the rule used must be *AlpZero* or *AlpIde*. In the former case both P and Q are 0 and so $0 \equiv_{\alpha} R$. For symmetry and the inversion lemma then R is also 0. In the latter case a similar argument applies. If the proof tree has more than one node then we proceed by cases on the last rule

AlpInp : In this case $P = x(y).P_1$, $Q = x(y).Q_1$ and $P_1 \equiv_{\alpha} Q_1$ and $x(y).Q_1 \equiv_{\alpha} R$ which implies for symmetry and the inversion lemma that one of the following cases holds:

- $R = x(y).R_1$ and $Q_1 \equiv_{\alpha} R_1$:

$P_1 \equiv_{\alpha} Q_1$ and $Q_1 \equiv_{\alpha} R_1$	inductive hypothesis
$\Rightarrow P_1 \equiv_{\alpha} R_1$	rule <i>AlpInp</i>
$\Rightarrow x(y).P_1 \equiv_{\alpha} x(y).R_1$	
- $R = x(z).R_1$ and $Q_1\{y/z\} \equiv_{\alpha} R_1$:

$P_1 \equiv_{\alpha} Q_1$	lemma 2.3.5
$\Rightarrow P_1\{y/z\} \equiv_{\alpha} Q_1\{y/z\}$	inductive hypothesis
$\Rightarrow P_1\{y/z\} \equiv_{\alpha} R_1$	rule <i>AlpInp1</i>
$\Rightarrow x(y).P_1 \equiv_{\alpha} x(z).R_1$	

AlpRes :

AlpInp1 :

AlpRes1 :

AlpSum :

AlpPar :

AlpSum :

AlpTau :

AlpOut :

□

Lemma 2.3.9. *E' FALSE!!!! !!!! !! !:*

- If $P \equiv \tau.Q$ then $P = \tau.P_1$ for some P_1 such that $P_1 \equiv Q$
- If $P \equiv \bar{x}y.Q$ then $P = \bar{x}y.P_1$ for some P_1 such that $P_1 \equiv Q$
- If $P \equiv x(y).Q$ then one and only one of the following cases holds:
 - $P = x(z).P_1$ for some P_1 such that $P_1\{z/y\} \equiv Q$
 - $P = x(y).P_1$ for some P_1 such that $P_1 \equiv Q$
- If $P \equiv Q_1 + Q_2$ then $P = P_1 + P_2$ for some P_1 and P_2 such that $P_1 \equiv Q_1$ and $P_2 \equiv Q_2$.
- If $P \equiv Q_1|Q_2$ then $P = P_1|P_2$ for some P_1 and P_2 such that $P_1 \equiv Q_1$ and $P_2 \equiv Q_2$.

Out $\frac{}{\bar{x}y.P \xrightarrow{\bar{x}y} P}$	EInp $\frac{}{x(y).P \xrightarrow{xz} P\{z/y\}}$
ParL $\frac{P \xrightarrow{\alpha} P' \quad bn(\alpha) \cap fn(Q) = \emptyset}{P Q \xrightarrow{\alpha} P' Q}$	ParR $\frac{Q \xrightarrow{\alpha} Q' \quad bn(\alpha) \cap fn(Q) = \emptyset}{P Q \xrightarrow{\alpha} P Q'}$
SumL $\frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'}$	SumR $\frac{Q \xrightarrow{\alpha} Q'}{P + Q \xrightarrow{\alpha} Q'}$
Res $\frac{P \xrightarrow{\alpha} P' \quad z \notin n(\alpha)}{(\nu z)P \xrightarrow{\alpha} (\nu z)P'}$	Alp $\frac{P \equiv_{\alpha} Q \quad P \xrightarrow{\alpha} P'}{Q \xrightarrow{\alpha} P'}$
EComL $\frac{P \xrightarrow{xy} P' \quad Q \xrightarrow{\bar{x}y} Q'}{P Q \xrightarrow{\tau} P' Q'}$	EComR $\frac{P \xrightarrow{\bar{x}y} P' \quad Q \xrightarrow{xy} Q'}{P Q \xrightarrow{\tau} P' Q'}$
ClsL $\frac{P \xrightarrow{\bar{x}(z)} P' \quad Q \xrightarrow{xz} Q' \quad z \notin fn(Q)}{P Q \xrightarrow{\tau} (\nu z)(P' Q')}$	ClsR $\frac{P \xrightarrow{xz} P' \quad Q \xrightarrow{\bar{x}(z)} Q' \quad z \notin fn(P)}{P Q \xrightarrow{\tau} (\nu z)(P' Q')}$
Cns $\frac{A(\tilde{x} \tilde{y}) \stackrel{def}{=} P \quad P\{\tilde{w}/\tilde{x}\} \xrightarrow{\alpha} P'}{A(\tilde{x} \tilde{y})\{\tilde{w}/\tilde{x}\} \xrightarrow{\alpha} P'}$	Opn $\frac{P \xrightarrow{\bar{x}z} P' \quad z \neq x}{(\nu z)P \xrightarrow{\bar{x}(z)} P'}$
Tau $\frac{}{\tau.P \xrightarrow{\tau} P}$	

Table 2.8: Early transition relation with α conversion

- If $P \equiv (\nu y)Q$ then one and only one of the following cases holds:
 - $P = (\nu z)P_1$ such that $P_1\{z/y\} \equiv Q$
 - $P = (\nu y).P_1$ for some P_1 such that $P_1 \equiv Q$
- If $P \equiv A(\tilde{x}|\tilde{y})$ then ??? ?? ?

Proof.

□

2.4 Operational semantic with structural congruence

2.4.1 Early semantic with α conversion only

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

Definition 2.4.1. *The early transition relation with α conversion $\rightarrow_{\subseteq} \mathbb{P} \times \mathbb{A} \times \mathbb{P}$ is the smallest relation induced by the rules in table 2.8.*

2.4.2 Early semantic with structural congruence

Definition 2.4.2. *The early transition relation with structural congruence $\rightarrow_{\subseteq} \mathbb{P} \times \mathbb{A} \times \mathbb{P}$ is the smallest relation induced by the rules in table 2.9.*

Example We prove now that

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

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

Table 2.9: Early semantic with structural congruence

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

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

and

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

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

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

Example We want to prove now that:

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

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

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

then we have

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

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

2.4.3 Late semantic with structural congruence

Definition 2.4.3. The late transition relation with structural congruence $\rightarrow \subseteq \mathbb{P} \times \mathbb{A} \times \mathbb{P}$ is the smallest relation induced by the rules in table 2.10.

Example We prove now that

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

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

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

and

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

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

Table 2.10: Late semantic with structural congruence

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

$$\begin{array}{c}
 \text{LINP} \frac{b \notin fn(P)}{a(x).P \xrightarrow{ab} P\{b/x\}} \quad \text{OUT} \frac{}{\bar{a}b.Q \xrightarrow{\bar{a}b} Q} \\
 \text{LCom} \frac{}{a(x).P | \bar{a}b.Q \xrightarrow{\tau} P\{b/x\} | Q} \quad b \notin n(\tau) \\
 \text{RES} \frac{}{(\nu b)(a(x).P | \bar{a}b.Q) \xrightarrow{\tau} (\nu b)(P\{b/x\} | Q)}
 \end{array}$$

Example We want to prove now that:

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

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

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

then we have

$$\begin{array}{c}
 \text{LINP} \frac{b \notin fn(P\{c/b\})}{a(x).P\{c/b\} \xrightarrow{ab} P\{c/b\}\{b/x\}} \quad \text{OUT} \frac{}{\bar{a}b.Q \xrightarrow{\bar{a}b} Q} \\
 \text{LCom} \frac{}{(a(x).(P\{c/b\}) | \bar{a}b.Q) \xrightarrow{\tau} (P\{c/b\}\{b/x\} | Q)} \quad c \notin n(\tau) \\
 \text{RES} \frac{}{(\nu c)(a(x).(P\{c/b\}) | \bar{a}b.Q) \xrightarrow{\tau} (\nu c)(P\{c/b\}\{b/x\} | Q)}
 \end{array}$$

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

2.5 Equivalence of the semantics

2.5.1 Equivalence of the early semantics

In this subsection we write \rightarrow_1 for the early semantic without structural congruence, \rightarrow_2 for the early semantic with just α conversion and \rightarrow_3 for the early semantic with the full structural congruence. We call R_1 the set of rules for \rightarrow_1 , R_2 the set of rules for \rightarrow_2 and R_3 the set of rules for \rightarrow_3 . In the following section we will need:

Lemma 2.5.1.

$$P \equiv Q \Rightarrow fn(Q) = fn(P)$$

Proof. A proof can go by induction on the proof tree of $P \equiv Q$ and then by cases on the last rule used in the proof tree.

base case The last and only rule of the proof tree can be one of the following axioms:

$$\begin{array}{ll}
\text{SC-ALP} & \frac{P \equiv_{\alpha} Q}{P \equiv Q} \\
\text{SC-SUM-ASC} & M_1 + (M_2 + M_3) \equiv (M_1 + M_2) + M_3 \\
\text{SC-SUM-COM} & M_1 + M_2 \equiv M_2 + M_1 \\
\text{SC-SUM-INC} & M + 0 \equiv M \\
\text{SC-COM-ASC} & P_1|(P_2|P_3) \equiv (P_1|P_2)|P_3 \\
\text{SC-COM-COM} & P_1|P_2 \equiv P_2|P_1 \\
\text{SC-COM-INC} & P|0 \equiv P \\
\text{SC-RES} & (\nu z)(\nu w)P \equiv (\nu w)(\nu z)P \\
\text{SC-RES-INC} & (\nu z)0 \equiv 0 \\
\text{SC-RES-COM} & (\nu z)(P_1|P_2) \equiv P_1|(\nu z)P_2 \text{ if } z \notin \text{fn}(P_1) \\
\text{SC-RES-SUM} & (\nu z)(P_1 + P_2) \equiv P_1 + (\nu z)P_2 \text{ if } z \notin \text{fn}(P_1) \\
\text{SC-IDE} & A(\tilde{w}|\tilde{y}) \equiv P\{\tilde{w}/\tilde{x}\}
\end{array}$$

inductive case

$$\text{SC-REFL} \quad P \equiv P$$

$$\text{SC-SIMM} \quad \frac{Q \equiv P}{P \equiv Q}$$

$$\text{SC-TRAN} \quad \frac{P \equiv Q \quad Q \equiv R}{P \equiv R}$$

$$\text{SC-CONG} \quad \frac{P \equiv Q}{C[P] \equiv C[Q]}$$

□

We would like to prove that $P \xrightarrow{\alpha}_2 P' \Rightarrow P \xrightarrow{\alpha}_1 P'$ but this is false because

$$\text{ALP} \quad \frac{\text{OUT} \quad \frac{\overline{xy}.x(y).0 \equiv_{\alpha} \overline{xy}.x(w).0}{\overline{xy}.x(w).0 \xrightarrow{\overline{xy}}_2 x(w).0}}{\overline{xy}.x(y).0 \xrightarrow{\overline{xy}}_2 x(w).0}$$

so we want to prove

$$\overline{xy}.x(y).0 \xrightarrow{\overline{xy}}_1 x(w).0$$

The head of the transition has an output prefixing at the top level so the only rule we could use is *Out*, but the application of *Out* yields

$$\overline{xy}.x(y).0 \xrightarrow{\overline{xy}}_1 x(y).0$$

which is not what we want. So we prove a weaker version

Theorem 2.5.2.

$$P \xrightarrow{\alpha}_2 P' \Rightarrow \exists P'' : P'' \equiv_{\alpha} P' \text{ and } P \xrightarrow{\alpha}_1 P''$$

Proof. The proof goes by induction on the depth of the derivation tree of $P \xrightarrow{\alpha}_2 P'$ and then by cases on the last rule used:

base case If the depth of the derivation tree is one, the rule used has to be a prefix rule

$$\{Out, EInp, Tau\} \subseteq R_1 \cap R_2$$

so a derivation tree of $P \xrightarrow{\alpha}_2 P'$ is also a derivation tree of $P \xrightarrow{\alpha}_1 P'$

inductive case If the depth of the derivation tree is more than one, then we proceed by cases on the last rule R . If the rule R is not a prefix rule and it is in common between the two semantics:

$$R \in \{ParL, ParR, SumL, SumR, Res, EComL, EComR, ClsL, ClsR, Cns, Opn\}$$

then we just apply the inductive hypothesis on the premises of R and then reapply R to get the desired derivation tree. We show just the case for $SumL$ when the end of the derivation tree is

$$\text{SUML} \frac{P_1 \xrightarrow{\alpha}_2 P'_1}{\underbrace{P_1 + P_2}_P \xrightarrow{\alpha}_2 \underbrace{P'_1}_{P'}}$$

rule premise
inductive hypothesis

$$\begin{aligned} &P_1 \xrightarrow{\alpha}_2 P'_1 \\ \Rightarrow P_1 \xrightarrow{\alpha}_1 P''_1 \text{ and } P'_1 \equiv_{\alpha} P''_1 &\quad \text{rule } SumL \\ \Rightarrow P_1 + P_2 \xrightarrow{\alpha}_1 P''_1 & \end{aligned}$$

If the rule R is in

$$R_2 - R_1 = \{Alp\}$$

then the last part of the derivation tree of $P \xrightarrow{\alpha}_2 P'$ is

$$\text{ALP} \frac{P \equiv_{\alpha} Q \quad \text{S} \frac{\dots}{Q \xrightarrow{\alpha}_2 P'}}{P \xrightarrow{\alpha}_2 P'}$$

and the proof goes by cases on S the last rule in the proof tree of $Q \xrightarrow{\alpha}_2 P'$:

Out : If $S = Out$ then there exists some names x, y and a process Q_1 such that

$$Q = \bar{x}y.Q_1$$

and $\alpha = \bar{x}y$.

$$\begin{aligned} &P \equiv_{\alpha} \bar{x}y.Q_1 && \text{inversion lemma} \\ \Rightarrow P = \bar{x}y.P_1 \text{ and } P_1 \equiv_{\alpha} Q_1 && \text{rule } Out \\ \Rightarrow \bar{x}y.P_1 \xrightarrow{\bar{x}y}_1 P_1 && \end{aligned}$$

EInp If $S = EInp$ then there exists some names x, y, z and a process Q_1 such that $Q = x(y).Q_1$, $\alpha = xz$ and $P' = Q_1\{z/y\}$. Since

$$P \equiv_{\alpha} x(y).Q_1$$

then for the inversion lemma we have two cases:

• :

$$\begin{aligned} &P = x(y).P_1 \text{ and } P_1 \equiv_{\alpha} Q_1 && \text{rule } EInp \\ \Rightarrow x(y).P_1 \xrightarrow{xz}_1 P_1\{z/y\} && \end{aligned}$$

This is what we want because for lemma 2.3.5

$$P_1 \equiv_{\alpha} Q_1 \Rightarrow P_1\{z/y\} \equiv_{\alpha} Q_1\{z/y\}$$

• :

$$\begin{aligned} &P = x(w).P_1 \text{ and } P_1\{y/w\} \equiv_{\alpha} Q_1 && \text{rule } EInp \\ \Rightarrow x(w).P_1 \xrightarrow{xz}_1 P_1\{z/w\} && \end{aligned}$$

This is what we want because

$$\begin{aligned}
P_1\{y/w\} &\equiv_{\alpha} Q_1 && \text{lemma 2.3.5} \\
\Rightarrow P_1\{y/w\}\{z/y\} &\equiv_{\alpha} Q_1\{z/y\} \\
\Rightarrow P_1\{z/w\} &\equiv_{\alpha} Q_1\{z/y\}
\end{aligned}$$

Tau If $S = \text{Tau}$ then there exists a process Q_1 such that $Q = \tau.Q_1$ and $\alpha = \tau$ and $P' = Q_1$.

$$\begin{aligned}
P &\equiv_{\alpha} \tau.Q_1 && \text{inversion lemma} \\
\Rightarrow P &= \tau.P_1 \text{ and } P_1 \equiv_{\alpha} Q_1 && \text{rule Tau} \\
\Rightarrow \tau.P_1 &\xrightarrow{\tau}_1 P_1
\end{aligned}$$

ParL If $S = \text{ParL}$ then there exists some processes Q_1, Q_2 such that

$$Q = Q_1|Q_2$$

Since

$$P \equiv_{\alpha} Q_1|Q_2$$

then for the inversion lemma there exists P_1, P_2 such that

$$P = P_1|P_2 \text{ and } P_1 \equiv_{\alpha} Q_1 \text{ and } P_2 \equiv_{\alpha} Q_2$$

and so the last part of the derivation tree of $P \xrightarrow{\alpha}_2 P'$ looks like this:

$$\text{ALP} \frac{P_1|P_2 \equiv_{\alpha} Q_1|Q_2 \quad \text{PARL} \frac{Q_1 \xrightarrow{\alpha}_2 Q'_1 \quad bn(\alpha) \cap fn(Q_2) = \emptyset}{Q_1|Q_2 \xrightarrow{\alpha}_2 Q'_1|Q_2}}{\underbrace{P_1|P_2}_P \xrightarrow{\alpha}_2 \underbrace{Q'_1|Q_2}_{P'}}$$

from this hypothesis we can create the following proof tree of $P_1 \xrightarrow{\alpha}_2 Q'_1$:

$$\text{ALP} \frac{P_1 \equiv_{\alpha} Q_1 \quad Q_1 \xrightarrow{\alpha}_2 Q'_1}{P_1 \xrightarrow{\alpha}_2 Q'_1}$$

this proof tree is smaller than the proof tree of $P_1|P_2 \xrightarrow{\alpha}_2 Q'_1|Q_2$ so we can apply the inductive hypothesis and get that there exists a process Q''_1 such that

$$Q'_1 \equiv Q''_1 \text{ and } P_1 \xrightarrow{\alpha}_1 Q''_1$$

then we apply again the rule *ParL* and get

$$\text{PARL} \frac{P_1 \xrightarrow{\alpha}_1 Q''_1 \quad bn(\alpha) \cap fn(P_2) = \emptyset}{\underbrace{P_1|P_2}_P \xrightarrow{\alpha}_1 \underbrace{Q''_1|P_2}_{P''}}$$

The second premise of the previous instance holds because:

$$bn(\alpha) \cap fn(Q_2) = \emptyset \text{ and } P_2 \equiv_{\alpha} Q_2 \Rightarrow bn(\alpha) \cap fn(P_2) = \emptyset$$

ParR, SumL, SumR, EComL, EComR, ClsL, ClsR This cases are similar to the previous.

Res If $S = \text{Res}$ then there exists some name z and a process Q_1 such that

$$Q = (\nu z)Q_1$$

and $P' = (\nu z)Q'_1$. Since

$$P \equiv_{\alpha} (\nu z)Q_1$$

then for the inversion lemma we have two cases:

- there exists some P_1 such that

$$P = (\nu z)P_1 \text{ and } P_1 \equiv_\alpha Q_1$$

and so the last part of the derivation tree of $P \xrightarrow{\alpha}_2 P'$ looks like this:

$$\text{ALP} \frac{(\nu z)P_1 \equiv_\alpha (\nu z)Q_1 \quad \text{RES} \frac{Q_1 \xrightarrow{\alpha}_2 Q'_1 \quad z \notin n(\alpha)}{(\nu z)Q_1 \xrightarrow{\alpha}_2 (\nu z)Q'_1}}{(\nu z)P_1 \xrightarrow{\alpha}_2 (\nu z)Q'_1}$$

from this we create the following proof tree of $P_1 \xrightarrow{\alpha}_2 Q'_1$:

$$\text{ALP} \frac{P_1 \equiv_\alpha Q_1 \quad Q_1 \xrightarrow{\alpha}_2 Q'_1}{P_1 \xrightarrow{\alpha}_2 Q'_1}$$

to which we can apply the inductive hypothesis and get that there exists a process Q''_1 such that

$$P_1 \xrightarrow{\alpha}_1 Q''_1 \text{ and } Q''_1 \equiv_\alpha Q'_1$$

then we apply the rule *Res* to get

$$\text{RES} \frac{P_1 \xrightarrow{\alpha}_1 Q''_1 \quad z \notin n(\alpha)}{(\nu z)P_1 \xrightarrow{\alpha}_1 (\nu z)Q''_1}$$

this satisfies the thesis of the theorem because

$$(\nu z)Q''_1 \equiv (\nu z)Q'_1$$

- there exists some P_1 such that

$$P = (\nu y)P_1 \text{ and } P_1\{z/y\} \equiv_\alpha Q_1$$

and so the last part of the derivation tree of $P \xrightarrow{\alpha}_2 P'$ looks like this:

$$\text{ALP} \frac{(\nu y)P_1 \equiv_\alpha (\nu z)Q_1 \quad \text{RES} \frac{Q_1 \xrightarrow{\alpha}_2 Q'_1 \quad z \notin n(\alpha)}{(\nu z)Q_1 \xrightarrow{\alpha}_2 (\nu z)Q'_1}}{(\nu y)P_1 \xrightarrow{\alpha}_2 (\nu z)Q'_1}$$

from this we create the following proof tree of $P_1\{z/y\} \xrightarrow{\alpha}_2 Q'_1$:

$$\text{ALP} \frac{P_1\{z/y\} \equiv_\alpha Q_1 \quad Q_1 \xrightarrow{\alpha}_2 Q'_1}{P_1\{z/y\} \xrightarrow{\alpha}_2 Q'_1}$$

to which we can apply the inductive hypothesis and get that there exists a process Q''_1 such that

$$P_1\{z/y\} \xrightarrow{\alpha}_1 Q''_1 \text{ and } Q''_1 \equiv_\alpha Q'_1$$

then we apply the rule *Res* and *ResALP* to get

$$\text{RESALP} \frac{\text{RES} \frac{P_1\{z/y\} \xrightarrow{\alpha}_1 Q''_1 \quad z \notin n(\alpha)}{(\nu z)P_1\{z/y\} \xrightarrow{\alpha}_1 (\nu z)Q''_1}}{(\nu y)P_1 \xrightarrow{\alpha}_1 (\nu z)Q''_1}$$

this satisfies the thesis of the theorem because

$$(\nu z)Q''_1 \equiv (\nu z)Q'_1$$

Alp we can assume that there are no two consecutive application of the rule *Alp* because we can merge them thanks to the transitivity of the alpha equivalence.

Opn If $S = \text{Opn}$ then there exists some names x, z and a process Q_1 such that

$$Q = (\nu z)Q_1$$

and $P' = Q'_1$ and $\alpha = \bar{x}(z)$. Since

$$P \equiv_\alpha (\nu z)Q_1$$

then for the inversion lemma we have two cases:

- there exists some P_1 such that

$$P = (\nu z)P_1 \text{ and } P_1 \equiv_\alpha Q_1$$

and so the last part of the derivation tree of $P \xrightarrow{\alpha}_2 P'$ looks like this:

$$\text{ALP} \frac{(\nu z)P_1 \equiv_\alpha (\nu z)Q_1 \quad \text{OPN} \frac{Q_1 \xrightarrow{\bar{x}z}_2 Q'_1 \quad z \neq x}{(\nu z)Q_1 \xrightarrow{\bar{x}(z)}_2 Q'_1}}{(\nu z)P_1 \xrightarrow{\bar{x}(z)}_2 Q'_1}$$

from this we create the following proof tree of $P_1 \xrightarrow{\bar{x}z}_2 Q'_1$:

$$\text{ALP} \frac{P_1 \equiv_\alpha Q_1 \quad Q_1 \xrightarrow{\bar{x}z}_2 Q'_1}{P_1 \xrightarrow{\bar{x}z}_2 Q'_1}$$

to which we can apply the inductive hypothesis and get that there exists a process Q''_1 such that

$$P_1 \xrightarrow{\bar{x}z}_1 Q''_1 \text{ and } Q''_1 \equiv_\alpha Q'_1$$

then we apply the rule *Opn* to get

$$\text{OPN} \frac{P_1 \xrightarrow{\bar{x}z}_1 Q''_1 \quad z \neq x}{(\nu z)P_1 \xrightarrow{\alpha}_1 Q''_1}$$

- there exists some P_1 such that

$$P = (\nu y)P_1 \text{ and } P_1\{z/y\} \equiv_\alpha Q_1$$

and so the last part of the derivation tree of $P \xrightarrow{\alpha}_2 P'$ looks like this:

$$\text{ALP} \frac{(\nu y)P_1 \equiv_\alpha (\nu z)Q_1 \quad \text{OPN} \frac{Q_1 \xrightarrow{\bar{x}z}_2 Q'_1 \quad z \neq x}{(\nu z)Q_1 \xrightarrow{\bar{x}z}_2 Q'_1}}{(\nu y)P_1 \xrightarrow{\alpha}_2 Q'_1}$$

from this we create the following proof tree of $P_1\{z/y\} \xrightarrow{\alpha}_2 Q'_1$:

$$\text{ALP} \frac{P_1\{z/y\} \equiv_\alpha Q_1 \quad Q_1 \xrightarrow{\alpha}_2 Q'_1}{P_1\{z/y\} \xrightarrow{\alpha}_2 Q'_1}$$

to which we can apply the inductive hypothesis and get that there exists a process Q''_1 such that

$$P_1\{z/y\} \xrightarrow{\alpha}_1 Q''_1 \text{ and } Q''_1 \equiv_\alpha Q'_1$$

then we apply the rule *Opn* and *OpnAlp* to get

$$\text{OPNALP} \frac{\text{OPN} \frac{P_1\{z/y\} \xrightarrow{\bar{x}z}_1 Q''_1 \quad z \neq x}{(\nu z)P_1\{z/y\} \xrightarrow{\bar{x}(z)}_1 Q''_1} \quad z \notin n(P) \quad x \neq y \neq z}{(\nu y)P_1 \xrightarrow{\bar{x}(z)}_1 Q''_1}$$

Cns Since there is no process α equivalent to an identifier except for the identifier itself, the last part of the derivation tree of $P \xrightarrow{\alpha_2} P'$ looks like this:

$$\text{ALP} \frac{A(\tilde{x}|\tilde{y})\{\tilde{w}/\tilde{x}\} \equiv_{\alpha} A(\tilde{x}|\tilde{y})\{\tilde{w}/\tilde{x}\} \quad \text{CNS} \frac{A(\tilde{x}|\tilde{y}) \stackrel{\text{def}}{=} R \quad R\{\tilde{w}/\tilde{x}\} \xrightarrow{\alpha_2} P'}{A(\tilde{x}|\tilde{y})\{\tilde{w}/\tilde{x}\} \xrightarrow{\alpha_2} P'}}{A(\tilde{x}|\tilde{y})\{\tilde{w}/\tilde{x}\} \xrightarrow{\alpha_2} P'}$$

here we can apply the inductive hypothesis on the conclusion of S and get that there exists a process P'' such that $A(\tilde{x}|\tilde{y})\{\tilde{w}/\tilde{x}\} \xrightarrow{\alpha_1} P''$ and $P' \equiv_{\alpha} P''$

□

Theorem 2.5.3. $P \xrightarrow{\alpha_1} P' \Rightarrow P \xrightarrow{\alpha_2} P'$

Proof. The proof can go by induction on the length of the derivation of a transaction, and then both the base case and the inductive case proceed by cases on the last rule used in the derivation. However it is not necessary to show all the details of the proof because the rules in R_2 are almost the same as the rules in R_1 , the only difference is that in R_2 we have the rule Alp instead of $ResAlp$ and $OpnAlp$. The rule Alp can mimic the rule $ResAlp$ in the following way:

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

And the rule Alp can mimic the rule $OpnAlp$ in the following way:

$$\frac{(\nu z)P \equiv_{\alpha} (\nu w)P\{w/z\} \quad w \notin n(P) \quad (\nu w)P\{w/z\} \xrightarrow{\bar{x}(w)} P' \quad x \neq w \neq z}{(\nu z)P \xrightarrow{\bar{x}(w)} P'}$$

□

Theorem 2.5.4. $P \xrightarrow{\alpha_2} P' \Leftrightarrow \exists P'' : P' \equiv P'' \text{ and } P \xrightarrow{\alpha_3} P''$

Proof. \Rightarrow First we prove $P \xrightarrow{\alpha_2} P' \Rightarrow \exists P'' : P' \equiv P'' \text{ and } P \xrightarrow{\alpha_3} P''$. The proof is by induction on the length of the derivation of $P \xrightarrow{\alpha_2} P'$, and then both the base case and the inductive case proceed by cases on the last rule used.

base case in this case the rule used can be one of the following $Out, EInp, Tau$ which are also in R_3 so a derivation of $P \xrightarrow{\alpha_2} P'$ is also a derivation of $P \xrightarrow{\alpha_3} P'$

inductive case :

- the last rule used can be one in $R_2 \cap R_3 = \{Res, Opn\}$ and so for example we have

$$\text{RES} \frac{P \xrightarrow{\alpha_2} P' \quad z \notin n(\alpha)}{(\nu z)P \xrightarrow{\alpha_2} (\nu z)P'}$$

we apply the inductive hypothesis on $P \xrightarrow{\alpha_2} P'$ and get $\exists P''$ such that $P' \equiv P''$ and $P \xrightarrow{\alpha_3} P''$. The proof we want is:

$$\text{RES} \frac{P \xrightarrow{\alpha_3} P'' \quad z \notin n(\alpha)}{(\nu z)P \xrightarrow{\alpha_3} (\nu z)P''}$$

and $(\nu z)P'' \equiv (\nu z)P'$

- the last rule used can be one in $\{ParL, ParR, SumL, SumR, EComL, EComR\}$, in this case we can proceed as in the previous case and if necessary add an application of Str thus exploiting the commutativity of sum or parallel composition. For example

$$\text{PARR} \frac{Q \xrightarrow{\alpha_2} Q' \quad bn(\alpha) \cap fn(Q) = \emptyset}{P|Q \xrightarrow{\alpha_2} P|Q'}$$

now we apply the inductive hypothesis to $Q \xrightarrow{\alpha}_2 Q'$ and get $Q \xrightarrow{\alpha}_3 Q''$ for a Q'' such that $Q' \equiv Q''$. The proof we want is

$$\text{STR} \frac{P|Q \equiv Q|P \quad \text{PAR} \frac{Q \xrightarrow{\alpha}_3 Q'' \quad bn(\alpha) \cap fn(Q) = \emptyset}{Q|P \xrightarrow{\alpha}_3 Q''|P}}{P|Q \xrightarrow{\alpha}_3 Q''|P}$$

and $Q''|P \equiv P|Q'$

- if the last rule used is *Cns*:

$$\text{CNS} \frac{A(\tilde{x}|\tilde{z}) \stackrel{def}{=} P \quad P\{\tilde{y}/\tilde{x}\} \xrightarrow{\alpha}_2 P'}{A(\tilde{y}|\tilde{z}) \xrightarrow{\alpha}_2 P'}$$

we apply the inductive hypothesis on the premise and get $P\{\tilde{y}/\tilde{x}\} \xrightarrow{\alpha}_3 P''$ such that $P'' \equiv P'$. Now the proof we want is

$$\text{STR} \frac{A(\tilde{y}|\tilde{z}) \equiv P\{\tilde{y}/\tilde{x}\} \quad P\{\tilde{y}/\tilde{x}\} \xrightarrow{\alpha}_3 P''}{A(\tilde{y}|\tilde{z}) \xrightarrow{\alpha}_3 P''}$$

- if the last rule is *Alp*, then we just notice that this rule is a particular case of *Str*
- if the last rule is *ClsL* (the case for *ClsR* is symmetric) then we have

$$\text{CLS L} \frac{P \xrightarrow{\bar{x}(z)}_2 P' \quad Q \xrightarrow{xz}_2 Q' \quad z \notin fn(Q)}{P|Q \xrightarrow{\tau}_2 (\nu z)(P'|Q')}$$

there is no easy way to mimic this rule with the rules in R_3 . But if in the derivation tree we have an introduction of the bound output $\bar{x}(z)$ followed directly by an elimination of the same bound output such as:

$$\text{CLS L} \frac{\text{OPN} \frac{P \xrightarrow{\bar{x}(z)}_2 P' \quad z \neq x}{(\nu z)P \xrightarrow{\bar{x}(z)}_2 P'} \quad Q \xrightarrow{xz}_2 Q' \quad z \notin fn(Q)}{((\nu z)P)|Q \xrightarrow{\tau}_2 (\nu z)(P'|Q')}$$

we can apply the inductive hypothesis and get that

$$P \xrightarrow{\bar{x}z}_3 P'' \text{ and } Q \xrightarrow{xz}_3 Q''$$

where $P' \equiv P''$ and $Q' \equiv Q''$, so we create the needed proof in the following way

$$\text{STR} \frac{(\nu z)(P|Q) \equiv ((\nu z)P)|Q \quad \text{COM} \frac{P \xrightarrow{\bar{x}z}_3 P'' \quad Q \xrightarrow{xz}_3 Q''}{P|Q \xrightarrow{\tau}_3 P''|Q''} \quad \text{RES} \frac{(\nu z)(P|Q) \xrightarrow{\tau}_3 (\nu z)(P''|Q'')}{(\nu z)(P|Q) \xrightarrow{\tau}_3 (\nu z)(P''|Q'')}}{((\nu z)P)|Q \xrightarrow{\tau}_3 (\nu z)(P''|Q'')}$$

We can always take a derivation tree in R_2 and move downward each occurrence of *Opn* until we find the appropriate occurrence of *ClsL*. In this process we might need to use the structural congruence, in particular the scope extension axioms. We can attempt to prove that in the following way:

$$P \xrightarrow{\bar{x}(z)}_2 P' \Rightarrow \exists R : (\nu z)R \equiv P$$

and if $(\nu z)R \xrightarrow{\bar{x}(z)}_2 P'$ then there exists a derivation tree for this transition such that the last rule used is *Opn*

PRIMA DEVO DIMOSTRARE IL LEMMA DI INVERSIONE PER LA CONGRUENZA STRUTTURALE(SE E' VERO)

Secondly we prove $P \xrightarrow{\alpha}_3 P' \Rightarrow \exists P'' : P' \equiv P''$ and $P \xrightarrow{\alpha}_2 P''$. The proof is by induction on the length of the derivation of $P \xrightarrow{\alpha}_3 P'$, and then both the base case and the inductive case proceed by cases on the last rule used.

\Leftarrow **base case** in this case the rule used can be one of the following *Out*, *EInp*, *Tau* which are also in R_2 so a derivation of $P \xrightarrow{\alpha}_3 P'$ is also a derivation of $P \xrightarrow{\alpha}_2 P'$

inductive case :

- the last rule used can be one in $R_2 \cap R_3 = \{Res, Opn\}$, this goes like in the previous proof for the opposite direction with the transition numbers swapped.
- the last rule used can be one of *Par*, *Sum* or *ECom*, in this case we apply the inductive hypothesis to the premises and then apply the appropriate rule: *ParL*, *SumL* or *EComL*. For example

$$\text{PAR} \frac{P \xrightarrow{\alpha}_3 P' \quad bn(\alpha) \cap fn(Q) = \emptyset}{P|Q \xrightarrow{\alpha}_3 P'|Q}$$

now we apply the inductive hypothesis to $P \xrightarrow{\alpha}_3 P'$ and get $P \xrightarrow{\alpha}_2 P''$ for a P'' such that $P' \equiv P''$. The proof we want is

$$\text{PARL} \frac{P \xrightarrow{\alpha}_2 P'' \quad bn(\alpha) \cap fn(Q) = \emptyset}{P|Q \xrightarrow{\alpha}_2 P|Q''}$$

and $Q''|P \equiv P|Q'$

- if the last rule is *Str*, then we have

$$\text{STR} \frac{P \equiv Q \quad Q \xrightarrow{\alpha}_3 P'}{P \xrightarrow{\alpha}_3 P'}$$

we proceed by cases on the premise $Q \xrightarrow{\alpha}_3 P'$. In the cases of prefix we can just use the appropriate prefix rule of R_2 and get rid of the *Str*. In the other cases we can move upward the occurrence of *Str*, after that we have one or two smaller derivation trees that are suitable to application of the inductive hypothesis and finally we apply some appropriate rules in R_2 .

Out Since we are using the rule *Out*, $Q = \bar{x}y.Q_1$ for some Q_1 . $Q \equiv P$ means for the inversion lemma for structural congruence that $P = \bar{x}y.P_1$ for some $P_1 \equiv Q_1$. The last part of the derivation tree is

$$\text{STR} \frac{\bar{x}y.P_1 \equiv \bar{x}y.Q_1 \quad \text{OUT} \frac{\bar{x}y.Q_1 \xrightarrow{\bar{x}y}_3 Q_1}{\bar{x}y.Q_1 \xrightarrow{\bar{x}y}_3 Q_1}}{\bar{x}y.P_1 \xrightarrow{\bar{x}y}_3 Q_1}$$

So we get

$$\text{OUT} \frac{\bar{x}y.P_1 \xrightarrow{\bar{x}y}_3 Q_1}{\bar{x}y.P_1 \xrightarrow{\bar{x}y}_2 P_1}$$

where $P_1 \equiv Q_1$

Tau this is very similar to the previous case

EInp Since we are using the rule *EInp*, $Q = x(y).Q_1$ for some Q_1 . From $Q \equiv P$ using the inversion lemma for structural congruence we can have two cases:

- $P = x(y).P_1$ for some $P_1 \equiv Q_1$. The last part of the derivation tree is

$$\text{STR} \frac{x(y).P_1 \equiv x(y).Q_1 \quad \text{EINP} \frac{x(y).Q_1 \xrightarrow{xw}_3 Q_1\{w/y\}}{x(y).Q_1 \xrightarrow{xw}_3 Q_1\{w/y\}}}{x(y).P_1 \xrightarrow{xw}_3 Q_1\{w/y\}}$$

So we get

$$\text{EINP} \frac{}{x(y).P_1 \xrightarrow{2} P_1\{w/y\}}$$

where $P_1 \equiv Q_1$ implies $P_1\{w/y\} \equiv Q_1\{w/y\}$

- $P = x(z).P_1$ for some $P_1 \equiv Q_1\{z/y\}$. The last part of the derivation tree is

$$\text{STR} \frac{x(z).P_1 \equiv x(y).Q_1 \quad \text{EINP} \frac{}{x(y).Q_1 \xrightarrow{3} Q_1\{w/y\}}}{x(z).P_1 \xrightarrow{3} Q_1\{w/y\}}$$

So we get

$$\text{EINP} \frac{}{x(z).P_1 \xrightarrow{2} P_1\{w/z\}}$$

where $P_1 \equiv Q_1\{z/y\}$ implies $P_1\{w/z\} \equiv Q_1\{z/y\}\{w/z\} \equiv Q_1\{w/y\}$

Par Since we are using the rule *Par*, $Q = Q_1|Q_2$ for some Q_1, Q_2 . $Q \equiv P$ means for the inversion lemma for structural congruence that $P = P_1|P_2$ for some P_1, P_2 such that $P_1 \equiv Q_1$ and $P_2 \equiv Q_2$. The last part of the derivation tree is

$$\text{STR} \frac{P_1|P_2 \equiv Q_1|Q_2 \quad \text{PAR} \frac{Q_1 \xrightarrow{3} Q'_1 \quad bn(\alpha) \cap fn(Q_2) = \emptyset}{Q_1|Q_2 \xrightarrow{3} Q'_1|Q_2}}{P_1|P_2 \xrightarrow{3} Q'_1|Q_2}$$

the first step is the creation of this proof tree:

$$\text{STR} \frac{P_1 \equiv Q_1 \quad Q_1 \xrightarrow{3} Q'_1}{P_1 \xrightarrow{3} Q'_1}$$

which is smaller then the inductive case, so we apply the inductive hypothesis and get $P_1 \xrightarrow{2} Q'_1$ where $Q'_1 \equiv Q''_1$. The last step is

$$\text{PARL} \frac{P_1 \xrightarrow{2} Q''_1 \quad bn(\alpha) \cap fn(P_2) = \emptyset}{P_1|P_2 \xrightarrow{2} Q''_1|P_2}$$

Sum this case is very similar to the previous.

ECom this case is also similar to the *Par* case.

Res Since we are using the rule *Res*, $Q = (\nu z)Q_1$ for some Q_1 and some z . $(\nu z)Q_1 \equiv P$ means thanks to the inversion lemma for structural congruence that one of the following cases holds:

- $P = (\nu z)P_1$ for some P_1 such that $P_1 \equiv Q_1$. The last part of the derivation tree is

$$\text{STR} \frac{(\nu z)P_1 \equiv (\nu z)Q_1 \quad \text{RES} \frac{Q_1 \xrightarrow{3} Q'_1 \quad z \notin n(\alpha)}{(\nu z)Q_1 \xrightarrow{3} (\nu z)Q'_1}}{(\nu z)P_1 \xrightarrow{3} (\nu z)Q'_1}$$

first we create the following proof:

$$\text{STR} \frac{P_1 \equiv Q_1 \quad Q_1 \xrightarrow{3} Q'_1}{P_1 \xrightarrow{3} Q'_1}$$

now we can apply the inductive hypothesis and get $P_1 \xrightarrow{2} Q'_1$ where $Q'_1 \equiv Q''_1$. The last step is

$$\text{RES} \frac{P_1 \xrightarrow{2} Q''_1 \quad z \notin n(\alpha)}{(\nu z)P_1 \xrightarrow{2} (\nu z)Q''_1}$$

- $P = (\nu y)P_1$ for some P_1 such that $P_1\{z/y\} \equiv Q_1$. The last part of the derivation tree is

$$\text{STR} \frac{(\nu y)P_1 \equiv (\nu z)Q_1 \quad \text{RES} \frac{Q_1 \xrightarrow{\alpha}_3 Q'_1 \quad z \notin n(\alpha)}{(\nu z)Q_1 \xrightarrow{\alpha}_3 (\nu z)Q'_1}}{(\nu y)P_1 \xrightarrow{\alpha}_3 (\nu z)Q'_1}$$

we create the following proof of $P_1\{z/y\} \xrightarrow{\alpha}_3 Q'_1$:

$$\text{STR} \frac{P_1\{z/y\} \equiv Q_1 \quad Q_1 \xrightarrow{\alpha}_3 Q'_1}{P_1\{z/y\} \xrightarrow{\alpha}_3 Q'_1}$$

this proof tree is shorter then the one of $(\nu y)P_1 \xrightarrow{\alpha}_3 (\nu z)Q'_1$ so we can apply the inductive hypothesis and get that there exists a process Q''_1 such that

$$P_1\{z/y\} \xrightarrow{\alpha}_2 Q''_1 \text{ and } Q''_1 \equiv Q'_1$$

now we can apply the rules *Res* and *Alp* to get the desired proof tree:

$$\text{ALP} \frac{(\nu z)P_1\{z/y\} \equiv_\alpha (\nu y)P_1 \quad \text{RES} \frac{P_1\{z/y\} \xrightarrow{\alpha}_2 Q''_1 \quad z \notin (\alpha)}{(\nu z)P_1\{z/y\} \xrightarrow{\alpha}_2 (\nu z)Q''_1}}{(\nu y)P_1 \xrightarrow{\alpha}_2 (\nu z)Q''_1}$$

Opn Since we are using the rule *Opn*, $Q = (\nu z)Q_1$ for some Q_1 . $(\nu z)Q_1 \equiv P$ means for the inversion lemma for structural congruence that

- $P = (\nu z)P_1$ for some P_1 such that $P_1 \equiv Q_1$. The last part of the derivation tree is

$$\text{STR} \frac{(\nu z)P_1 \equiv (\nu z)Q_1 \quad \text{OPN} \frac{Q_1 \xrightarrow{\bar{x}z} Q'_1 \quad z \neq x}{(\nu z)Q_1 \xrightarrow{\bar{x}(z)} Q'_1}}{(\nu z)P_1 \xrightarrow{\bar{x}(z)} Q'_1}$$

first:

$$\text{STR} \frac{P_1 \equiv Q_1 \quad Q_1 \xrightarrow{\bar{x}z}_3 Q'_1}{P_1 \xrightarrow{\bar{x}z}_3 Q'_1}$$

then we apply the inductive hypothesis and get $P_1 \xrightarrow{\bar{x}z}_2 Q''_1$ where $Q'_1 \equiv Q''_1$. The last step is

$$\text{RES} \frac{P_1 \xrightarrow{\bar{x}z}_2 Q''_1 \quad z \neq x}{(\nu z)P_1 \xrightarrow{\bar{x}z}_2 Q''_1}$$

- $P = (\nu z)P_1$ for some P_1 such that $P_1 \equiv Q_1$. The last part of the derivation tree is

$$\text{STR} \frac{(\nu z)P_1 \equiv (\nu z)Q_1 \quad \text{OPN} \frac{Q_1 \xrightarrow{\bar{x}z} Q'_1 \quad z \neq x}{(\nu z)Q_1 \xrightarrow{\bar{x}(z)} Q'_1}}{(\nu z)P_1 \xrightarrow{\bar{x}(z)} Q'_1}$$

the first step is:

$$\text{STR} \frac{P_1 \equiv Q_1 \quad Q_1 \xrightarrow{\bar{x}z}_3 Q'_1}{P_1 \xrightarrow{\bar{x}z}_3 Q'_1}$$

then we apply the inductive hypothesis and get $P_1 \xrightarrow{\bar{x}z}_2 Q''_1$ where $Q'_1 \equiv Q''_1$. The last step is

$$\text{RES} \frac{P_1 \xrightarrow{\bar{x}z}_2 Q''_1 \quad z \neq x}{(\nu z)P_1 \xrightarrow{\bar{x}z}_2 Q''_1}$$

- $P = (\nu y)P_1$ for some P_1 such that $P_1\{z/y\} \equiv Q_1$. The last part of the derivation tree is

$$\text{STR} \frac{(\nu y)P_1 \equiv (\nu z)Q_1 \quad \text{OPN} \frac{Q_1 \xrightarrow{\bar{x}z}_3 Q'_1 \quad z \neq x}{(\nu z)Q_1 \xrightarrow{\bar{x}(z)}_3 Q'_1}}{(\nu y)P_1 \xrightarrow{\bar{x}(z)}_3 Q'_1}$$

we can create the following proof of $P_1\{z/y\} \xrightarrow{\bar{x}z}_3 Q'_1$:

$$\text{STR} \frac{P_1\{z/y\} \equiv Q_1 \quad Q_1 \xrightarrow{\bar{x}z}_3 Q'_1}{P_1\{z/y\} \xrightarrow{\bar{x}z}_3 Q'_1}$$

this proof tree is shorter then the one of $(\nu y)P_1 \xrightarrow{\bar{x}(z)}_3 Q'_1$ so we can apply the inductive hypothesis and get that there exists a process Q'_1 such that

$$Q''_1 \equiv Q'_1 \text{ and } P_1\{z/y\} \xrightarrow{\bar{x}z}_2 Q''_1$$

so now we only need to apply the rules *Opn* and *Alp*:

$$\text{ALP} \frac{(\nu y)P_1 \equiv_\alpha (\nu z)P_1\{z/y\} \quad \text{OPN} \frac{P_1\{z/y\} \xrightarrow{\bar{x}z}_2 Q''_1 \quad z \neq x}{(\nu z)P_1\{z/y\} \xrightarrow{\bar{x}(z)}_3 Q''_1}}{(\nu y)P_1 \xrightarrow{\bar{x}(z)}_2 Q''_1}$$

□

2.5.2 Equivalence of the late semantics

2.6 Bisimilarity and Congruence

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

2.6.1 Bisimilarity

In the following we will use the phrase $bn(\alpha)$ is fresh in a definition to mean that the name in $bn(\alpha)$, if any, is different from any free name occurring in any of the agents in the definition. We write

$$\rightarrow_E$$

for the early semantic and

$$\rightarrow_L$$

for the late semantic.

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

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

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

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

Example Two strongly bisimilar processes are the following:

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

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

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

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

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

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

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

2.6.2 Congruence

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

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

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

2.6.3 Variants of Bisimilarity

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

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

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

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

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

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

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

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

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

where σ is a substitution.

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

In the following definition we consider a subcalculus without restriction.

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

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

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

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

Chapter 3

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

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

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

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

3.2 Operational semantic

3.2.1 Early operational semantic with structural congruence

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

- the nodes are multi π calculus process. The set of node is \mathbb{P}_m
- the actions are multi π calculus actions. The set of actions is \mathbb{A}_m , we use $\alpha, \alpha_1, \alpha_2, \dots$ 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 $\rightarrow \subseteq \mathbb{P}_m \times (\mathbb{A}_m^+ \cup \{\tau\}) \times \mathbb{P}_m$

Out $\frac{}{\bar{x}y.P \xrightarrow{\bar{x}y} P}$	EInp $\frac{}{x(y).P \xrightarrow{xz} P\{z/y\}}$
Tau $\frac{}{\tau.P \xrightarrow{\tau} P}$	SOut $\frac{P \xrightarrow{\sigma} P' \quad \sigma \neq \tau}{\bar{x}y.P \xrightarrow{\bar{x}y \cdot \sigma} P'}$
Sum $\frac{P \xrightarrow{\sigma} P'}{P + Q \xrightarrow{\sigma} P'}$	Str $\frac{P \equiv P' \quad P' \xrightarrow{\alpha} Q' \quad Q \equiv Q'}{P \xrightarrow{\alpha} Q}$
Par $\frac{P \xrightarrow{\sigma} P' \quad bn(\sigma) \cap fn(Q) = \emptyset}{P Q \xrightarrow{\sigma} P' Q}$	EComSng $\frac{P \xrightarrow{xy} P' \quad Q \xrightarrow{\bar{x}y} Q'}{P Q \xrightarrow{\tau} P' Q'}$
Res $\frac{P \xrightarrow{\sigma} P' \quad z \notin n(\alpha)}{(\nu)zP \xrightarrow{\sigma} (\nu)zP'}$	EComSeq $\frac{P \xrightarrow{xy} P' \quad Q \xrightarrow{\bar{x}y \cdot \sigma} Q'}{P Q \xrightarrow{\sigma} P' Q'}$
SOutTau $\frac{P \xrightarrow{\tau} P'}{\bar{x}y.P \xrightarrow{\bar{x}y} P'}$	OpnSeq $\frac{P \xrightarrow{\sigma} P' \quad \exists \bar{x}z \in \sigma : x \neq z}{(\nu z)P \xrightarrow{opn(\sigma, z)} P'}$

Table 3.1: Multi π early semantic with structural congruence

$\frac{x \neq z}{opn(\bar{x}z, z) = \bar{x}(z)}$	$\frac{x \neq z}{opn(\bar{x}z \cdot \sigma, z) = \bar{x}(z) \cdot opn(\sigma, z)}$	$\frac{}{opn(xy, z) = xy}$
$\frac{}{opn(\bar{x}y, z) = \bar{x}y}$	$\frac{}{opn(\bar{x}y \cdot \sigma, z) = \bar{x}y \cdot opn(\sigma, z)}$	

Table 3.2: relation opn

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

Definition 3.2.1. *The early transition relation without structural congruence is the smallest relation induced by the rules in table 3.1. The relation opn is defined in table 3.2.*

Multi-party synchronization We show an example of a derivation of three processes that synchronize.

$$\begin{array}{c}
\text{Res} \frac{x \notin n(\tau) \quad \text{EComSeq} \frac{\bar{x}y.\bar{x}y.0|x(y).0 \xrightarrow{\bar{x}y} 0|0 \quad \text{Inp} \frac{}{x(y).0 \xrightarrow{xy} 0}}{((\bar{x}y.\bar{x}y.0|x(y).0)|x(y).0) \xrightarrow{\tau} ((0|0)|0)}}{(\nu x)((\bar{x}y.\bar{x}y.0|x(y).0)|x(y).0) \xrightarrow{\tau} (\nu x)((0|0)|0)} \\
\\
\text{EComSng} \frac{\text{SOut} \frac{\text{Out} \frac{}{\bar{x}y.0 \xrightarrow{\bar{x}y} 0}}{\bar{x}y.\bar{x}y.0 \xrightarrow{\bar{x}y \cdot \bar{x}y} 0} \quad x(y).0 \xrightarrow{xy} 0}{\bar{x}y.\bar{x}y.0|x(y).0 \xrightarrow{\bar{x}y} 0|0}
\end{array}$$

Transactional synchronization In this setting two process cannot synchronize on a sequence of actions with length greater than one. This is because of the rules $EComSng$ and $EComSeq$.

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

Table 3.3: Multi π late semantic with structural congruence

3.2.2 Late operational semantic with structural congruence

Definition 3.2.2. *The late transition relation with structural congruence is the smallest relation induced by the rules in table 3.3.*

Multi-party synchronization We show an example of a derivation of three processes that synchronize in the late semantic.

$$\begin{array}{c}
 \text{Res } \frac{x \notin n(\tau) \quad \text{LComSeq } \frac{\bar{x}y.\bar{x}y.0|x(y).0 \xrightarrow{\bar{x}y} 0|0 \quad \text{Pref } \frac{}{x(y).0 \xrightarrow{x(y)} 0}}{((\bar{x}y.\bar{x}y.0|x(y).0)|x(y).0) \xrightarrow{\tau} ((0|0)|0)}}{(\nu x)((\bar{x}y.\bar{x}y.0|x(y).0)|x(y).0) \xrightarrow{\tau} (\nu x)((0|0)|0)} \\
 \\
 \text{LComSng } \frac{\text{SOut } \frac{\text{Pref } \frac{}{\bar{x}y.0 \xrightarrow{\bar{x}y} 0}}{\bar{x}y.\bar{x}y.0 \xrightarrow{\bar{x}y.\bar{x}y} 0} \quad \text{Pref } \frac{}{x(y).0 \xrightarrow{x(y)} 0}}{\bar{x}y.\bar{x}y.0|x(y).0 \xrightarrow{\bar{x}y} 0|0}
 \end{array}$$

Chapter 4

Multi π calculus with strong input

4.1 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 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:

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

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. Since the strong prefix can be on input names only, the only synchronization possible is between a process that executes a sequence of n actions (only the last action can be an output) with $n \geq 1$ and n other processes each executing one single action (at least $n - 1$ process execute an output and at most one executes an input).

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

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

4.2 Operational semantic

4.2.1 Early operational semantic with structural congruence

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

- the nodes are multi π calculus process. The set of 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.

Out $\frac{}{\bar{x}y.P \xrightarrow{\bar{x}y} P}$	EInp $\frac{}{x(y).P \xrightarrow{xz} P\{z/y\}}$
Tau $\frac{}{\tau.P \xrightarrow{\tau} P}$	SInp $\frac{P\{y/z\} \xrightarrow{\sigma} P' \quad \sigma \neq \tau}{x(z).P \xrightarrow{xy \cdot \sigma} P'}$
Sum $\frac{P \xrightarrow{\sigma} P'}{P + Q \xrightarrow{\sigma} P'}$	Str $\frac{P \equiv P' \quad P' \xrightarrow{\alpha} Q' \quad Q \equiv Q'}{P \xrightarrow{\alpha} Q}$
Com $\frac{P \xrightarrow{\bar{x}y} P' \quad Q \xrightarrow{xy} Q'}{P Q \xrightarrow{\tau} P' Q'}$	ComSeq $\frac{P \xrightarrow{xy \cdot \sigma} P' \quad Q \xrightarrow{\bar{x}y} Q'}{P Q \xrightarrow{\sigma} P' Q'}$
Res $\frac{P \xrightarrow{\sigma} P' \quad z \notin n(\alpha)}{(\nu z)P \xrightarrow{\sigma} (\nu z)P'}$	SInpTau $\frac{P\{y/z\} \xrightarrow{\tau} P'}{x(z).P \xrightarrow{xy} P'}$
Par $\frac{P \xrightarrow{\sigma} P' \quad bn(\sigma) \cap fn(Q) = \emptyset}{P Q \xrightarrow{\sigma} P' Q}$	Opn $\frac{P \xrightarrow{\bar{x}z} P' \quad z \neq x}{(\nu z)P \xrightarrow{\bar{x}(z)} P'}$

Table 4.1: Multi π early semantic with structural congruence

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

Multi-party synchronization We show an example of a derivation of three processes that synchronize.

$$\begin{array}{c}
 \text{EComSng} \frac{\frac{x(a).x(b).P|\bar{x}y.Q \xrightarrow{xz} P\{y/a\}\{z/b\}|Q \quad \text{Out} \frac{}{\bar{x}z.R \xrightarrow{\bar{x}z} R}}{(x(a).x(b).P|\bar{x}y.Q)|\bar{x}z.R \xrightarrow{\tau} (P\{y/a\}\{z/b\}|Q)|R}} \\
 \text{EComSeq} \frac{\frac{\text{SInp} \frac{\text{EInp} \frac{}{(x(b).P)\{y/a\} \xrightarrow{xz} P\{y/a\}\{z/b\}}}{x(a).(x(b).P) \xrightarrow{xy \cdot xz} P\{y/a\}\{z/b\}} \quad \text{Out} \frac{}{\bar{x}y.Q \xrightarrow{\bar{x}y} Q}}{x(a).x(b).P|\bar{x}y.Q \xrightarrow{xz} P\{y/a\}\{z/b\}|Q}}
 \end{array}$$

4.2.2 Late operational semantic with structural congruence

Definition 4.2.2. *The late transition relation with structural congruence is the smallest relation induced by the rules in table 4.2.*

Multi-party synchronization We show an example of a derivation of three processes that synchronize with the late semantic. The three processes are $\underline{x(a)}.x(b).P$, $\bar{x}y.Q$ and $\bar{x}z.R$. We assume that:

$$a \notin fn(x(b)) \cup fn(\underline{x(a)}.x(b).P)$$

and

$$b \notin fn(\underline{x(a)}.x(b).P|\bar{x}y.Q)$$

$$\text{LComSng} \frac{\frac{x(a).x(b).P|\bar{x}y.Q \xrightarrow{x(b)} P\{y/a\}|Q \quad \text{Pref} \frac{}{\bar{x}z.R \xrightarrow{\bar{x}z} R}}{(\underline{x(a)}.x(b).P|\bar{x}y.Q)|\bar{x}z.R \xrightarrow{\tau} (P\{y/a\}|Q)\{z/b\}|R}}$$

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

Table 4.2: Multi π late semantic with structural congruence

$$\begin{array}{c}
\mathbf{Pref} \frac{}{x(b).P \xrightarrow{x(b)} P} \\
\mathbf{SInp} \frac{}{x(a).x(b).P \xrightarrow{x(a).x(b)} P} \quad \mathbf{Out} \frac{}{\bar{x}y.Q \xrightarrow{\bar{x}y} Q} \\
\mathbf{LComSeq} \frac{}{x(a).x(b).P|\bar{x}y.Q \xrightarrow{x(b)} P\{y/a\}|Q}
\end{array}$$

Transactional synchronization In this setting two process cannot synchronize on a sequence of actions with length greater than one.

4.2.3 low level semantic

This section contains the definition of an alternative semantic for multi π . First we define a low level version of the multi π calculus(here with strong prefixing on input only), we call this language low multi π . The low multi π is the multi π enriched with a marked or intermediate process $*P$:

$$P, Q ::= 0 \mid \pi.P \mid P|Q \mid P+Q \mid (\nu x)P \mid A(x_1, \dots, x_n) \mid *P$$

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

Definition 4.2.3. The low level transition relation is the smallest relation induced by the rules in table 4.3 in which P stands for a process without mark, L stands for a process with mark and S can stand for both.

Out $\frac{}{\bar{x}y.P \mapsto^{\bar{x}y} P}$	EInp $\frac{y \notin fn(P) - \{z\}}{x(z).P \mapsto^{xy} P\{y/z\}}$
Tau $\frac{}{\tau.P \mapsto^{\tau} P}$	SInp $\frac{y \notin fn(P) - \{z\}}{x(z).P \mapsto^{xy} *P\{y/z\}}$
Sum $\frac{P \mapsto^{\gamma} S}{P + Q \mapsto^{\gamma} S}$	Str $\frac{P \equiv P' \quad P' \mapsto^{\gamma} Q'}{P \mapsto^{\gamma} Q}$
Com1 $\frac{P \mapsto^{\bar{x}y} P' \quad Q \mapsto^{xy} Q'}{P Q \mapsto^{\tau} P' Q'}$	Com2 $\frac{L_1 \mapsto^{\bar{x}y} L'_1 \quad L_2 \mapsto^{xy} L'_2}{L_1 L_2 \mapsto^{\epsilon} L'_1 L'_2}$
Com3 $\frac{P \mapsto^{\bar{x}y} L \quad Q \mapsto^{xy} S}{P Q \mapsto^{\epsilon} L S}$	Com4 $\frac{L_1 \mapsto^{\bar{x}y} P \quad L_2 \mapsto^{xy} Q}{L_1 L_2 \mapsto^{\tau} P Q}$
Res $\frac{S \mapsto^{\gamma} S' \quad y \notin n(\gamma)}{(\nu y)S \mapsto^{\gamma} (\nu y)S'}$	Opn $\frac{S \mapsto^{\bar{x}y} S' \quad y \neq x \quad w \notin fn(S')}{(\nu y)S \mapsto^{\bar{x}(w)} S'\{w/y\}}$
Par1 $\frac{S \mapsto^{\gamma} S' \quad bn(\gamma) \cap fn(Q) = \emptyset}{S Q \mapsto^{\gamma} S' Q}$	Par1 $\frac{P \mapsto^{\gamma} L \quad bn(\gamma) \cap fn(Q) = \emptyset}{P Q \mapsto^{\gamma} L *Q}$
Par3 $\frac{S \mapsto^{\gamma} S' \quad bn(\gamma) \cap fn(Q) = \emptyset}{S Q \mapsto^{\gamma} S' Q}$	

Table 4.3: Low multi π ?early? semantic ?with structural congruence?

Chapter 5

Multi π calculus with strong input and output

5.1 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:

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

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:

$$\begin{aligned} B(x(y).Q, I) &= \{y, \bar{y}\} \cup B(Q, I) & F(x(y).Q, I) &= \{x, \bar{x}\} \cup (F(Q, I) - \{y, \bar{y}\}) \\ B(\bar{x}y.Q, I) &= B(Q, I) & F(\bar{x}y.Q, I) &= \{x, \bar{x}, y, \bar{y}\} \cup F(Q, I) \end{aligned}$$

5.2 Operational semantic

5.2.1 Early operational semantic with structural congruence

5.2.2 Late operational semantic with structural congruence

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

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

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

S1L $\frac{}{Sync(x(y), \bar{x}z, \tau, \{z/y\}, \{\})}$	S1R $\frac{}{Sync(\bar{x}z, x(y), \tau, \{\}, \{z/y\})}$
S2L $\frac{}{Sync(x(y), \bar{x}z \cdot \sigma, \sigma, \{z/y\}, \{\})}$	S2R $\frac{}{Sync(\bar{x}z \cdot \sigma, x(y), \sigma, \{\}, \{z/y\})}$
S3L $\frac{}{Sync(x(y) \cdot \sigma, \bar{x}z, \sigma\{z/y\}, \{z/y\}, \{\})}$	S3R $\frac{}{Sync(\bar{x}z, x(y) \cdot \sigma, \sigma\{z/y\}, \{\}, \{z/y\})}$
S4L $\frac{Sync(\sigma_1, \sigma_2\{z/y\}, \sigma_3, \delta_1, \delta_2)}{Sync(x(y) \cdot \sigma_1, \bar{x}z \cdot \sigma_2, \sigma_3, \{z/y\}\delta_1, \delta_2)}$	S4R $\frac{Sync(\sigma_1, \sigma_2\{z/y\}, \sigma_3, \delta_1, \delta_2)}{Sync(\bar{x}z \cdot \sigma_1, x(y) \cdot \sigma_2, \sigma_3, \delta_1, \{z/y\}\delta_2)}$
I1L $\frac{Sync(\sigma_1, \sigma_2, \tau, \delta_1, \delta_2)}{Sync(\alpha \cdot \sigma_1, \sigma_2, \alpha, \delta_1, \delta_2)}$	I1R $\frac{Sync(\sigma_1, \sigma_2, \tau, \delta_1, \delta_2)}{Sync(\sigma_1, \alpha \cdot \sigma_2, \alpha, \delta_1, \delta_2)}$
I2L $\frac{Sync(\sigma_1, \sigma_2, \sigma_3, \delta_1, \delta_2)}{Sync(\alpha \cdot \sigma_1, \sigma_2, \alpha \cdot \sigma_3, \delta_1, \delta_2)}$	I2R $\frac{Sync(\sigma_1, \sigma_2, \sigma_3, \delta_1, \delta_2)}{Sync(\sigma_1, \alpha \cdot \sigma_2, \alpha \cdot \sigma_3, \delta_1, \delta_2)}$
I3L $\frac{}{Sync(\alpha, \sigma, \alpha \cdot \sigma, \delta_1, \delta_2)}$	I3R $\frac{}{Sync(\sigma, \alpha, \alpha \cdot \sigma, \delta_1, \delta_2)}$
I4L $\frac{}{Sync(\epsilon, \sigma, \sigma, \delta_1, \delta_2)}$	I4R $\frac{}{Sync(\sigma, \epsilon, \sigma, \delta_1, \delta_2)}$

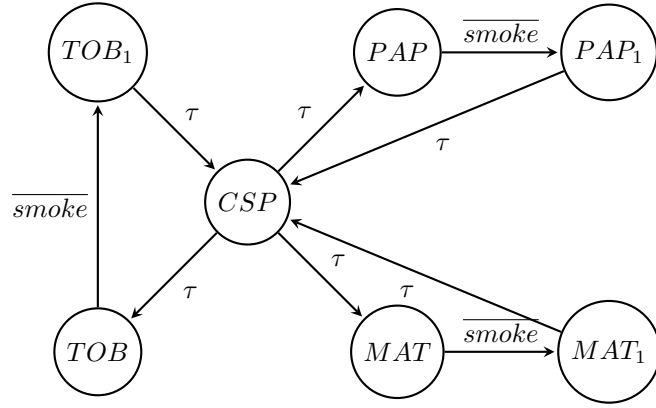
Table 5.2: Synchronization relation

$$\begin{array}{c}
\text{LCom} \frac{\bar{a}f.\bar{b}g.P|a(w).Q \xrightarrow{\bar{b}g} P|Q\{f/w\} \quad \text{Pref} \frac{}{b(y).R \xrightarrow{b(y)} R} \quad \text{S1R} \frac{}{Sync(\bar{b}g, b(y), \tau, \emptyset, \{g/y\})}}{(\bar{a}f.\bar{b}g.P|a(w).Q)|b(y).R \xrightarrow{\tau} (P|Q\{f/w\})|R\{g/y\}} \\
\\
\text{LCom} \frac{\bar{a}f.\bar{b}g.P \xrightarrow{\bar{a}f.\bar{b}g} P \quad \text{Pref} \frac{}{a(w).Q \xrightarrow{a(w)} Q} \quad \text{S2R} \frac{}{Sync(\bar{a}f \cdot \bar{b}g, a(w), \bar{b}g, \emptyset, \{f/w\})}}{\bar{a}f.\bar{b}g.P|a(w).Q \xrightarrow{\bar{b}g} P|Q\{f/w\}} \\
\\
\text{SOut} \frac{\text{Out} \frac{}{\bar{b}g.P \xrightarrow{\bar{b}g} P}}{\bar{a}f.\bar{b}g.P \xrightarrow{\bar{a}f.\bar{b}g} P}
\end{array}$$

Cigarette smokers' problem In this problem there are four processes: an agent and three smokers. Each smoker continuously makes a cigarette and smokes it. To make a cigarette each smoker needs three ingredients: tobacco, paper and matches. One of the smokers has paper, another tobacco and the third matches. The agent has an infinite supply of the ingredients. The agent places two of the ingredients on the table. The smoker who has the remaining ingredient take the others from the table, make a cigarette and smokes. Then the cycle repeats. A solution to the problem is the following:

$$\begin{aligned}
Agent &\stackrel{def}{=} \overline{tob}. \overline{mat}. end(). Agent + \overline{mat}. \overline{pap}. end(). Agent + \overline{pap}. \overline{tob}. end(). Agent \\
S_{pap} &\stackrel{def}{=} \overline{tob}(). \overline{mat}(). \overline{smoke}. end(). S_{pap} \\
S_{tab} &\stackrel{def}{=} \overline{mat}(). \overline{pap}(). \overline{smoke}. end(). S_{tab} \\
S_{mat} &\stackrel{def}{=} \overline{pap}(). \overline{tob}(). \overline{smoke}. end(). S_{mat} \\
CSP &\stackrel{def}{=} (\nu tob, pap, mat, end)(Agent | S_{tob} | S_{mat} | S_{pap})
\end{aligned}$$

The semantic of CSP is the following graph:



where

$$\begin{aligned}
 PAP &\stackrel{def}{=} (\nu tob, pap, mat, end)(end().Agent|S_{tob}|S_{mat}|\overline{smoke}.\overline{end}.S_{pap}) \\
 TOB &\stackrel{def}{=} (\nu tob, pap, mat, end)(end().Agent|\overline{smoke}.\overline{end}.S_{tob}|S_{mat}|S_{pap}) \\
 MAT &\stackrel{def}{=} (\nu tob, pap, mat, end)(end().Agent|S_{tob}|\overline{smoke}.\overline{end}.S_{mat}|S_{pap}) \\
 PAP_1 &\stackrel{def}{=} (\nu tob, pap, mat, end)(end().Agent|S_{tob}|S_{mat}|\overline{end}.S_{pap}) \\
 TOB_1 &\stackrel{def}{=} (\nu tob, pap, mat, end)(end().Agent|\overline{end}.S_{tob}|S_{mat}|S_{pap}) \\
 MAT_1 &\stackrel{def}{=} (\nu tob, pap, mat, end)(end().Agent|S_{tob}|\overline{end}.S_{mat}|S_{pap})
 \end{aligned}$$

Bibliography

- [1] Roberto Gorrieri, Cristian Versari, *Multi π : a calculus for mobile multi-party and transactional communication* .
- [2] E. W. Dijkstra, *Hierarchical ordering of sequential processes*, Acta Informatica 1(2):115-138, 1971 .
- [3] Roberto Gorrieri, *A Fully-Abstract Semantics for Atomicity*, Dipartimento di scienze dell'informazione, Università di Bologna .
- [4] Joachin Parrow, *An Introduction to the π Calculus*, Department Teleinformatics, Rotal Institute of Technology, Stockholm .
- [5] Davide Sangiorgi, David Walker, *The π -calculus*, Cambridge University Press .
- [6] Milner, Robin, *Communicating and Mobile Systems: The π -calculus*, Cambridge University Press .
- [7] MohammedReza Mousavi, Michel A Reniers, *Congruence for Structural Congruences*, Department of Computer Science, Eindhoven University of Technology .