Modelling and Validation of Concurrent System: the π-calculus

António Ravara

May 8, 2024

Motivation

Challenging scenarios for CCS

Mobile/Cell phone

1. Move and connect/disconnect to/from different antenas

Challenging scenarios for CCS

Mobile/Cell phone

1. Move and connect/disconnect to/from different antenas

Workload balancers

Dynamically create new threads

Passing channels on channels

Channels as payload

- A process can receive a channel on a channel and use it

Passing channels on channels

Channels as payload

- A process can receive a channel on a channel and use it

$$\overline{a}\langle b\rangle.0 \mid a(x).\overline{x}\langle 42\rangle.0 \xrightarrow{\tau} 0 \mid \overline{b}\langle 42\rangle.0$$

Passing channels on channels

Channels as payload

- A process can receive a channel on a channel and use it

$$\overline{a}\langle b\rangle.0 \mid a(x).\overline{x}\langle 42\rangle.0 \xrightarrow{\tau} 0 \mid \overline{b}\langle 42\rangle.0$$

- Channels can be revealed

$$(\text{new}\, b)\overline{a}\langle b\rangle.b(v).P \mid a(x).\overline{x}\langle 42\rangle.0 \xrightarrow{\tau} (\text{new}\, b)(b(v).P \mid \overline{b}\langle 42\rangle.0)$$

The (synchronous monadic) π-Calculus

Syntax – "lifting" from CCS

Since we are not interested (for now) in axiomatizations,
we define a minimal Turing-complete calculus.

Actions, Act**, ranged over by** α

Consider a countable set $\mathcal{N}$ of names, ranged over by a, b, x, possibly indexed or primed.

$$\alpha ::= \overline{a}\langle b\rangle \mid a(x) \mid \tau$$

Syntax – "lifting" from CCS

Since we are not interested (for now) in axiomatizations,
we define a minimal Turing-complete calculus.

Actions, Act**, ranged over by** α

Consider a countable set $\mathcal{N}$ of names, ranged over by a, b, x, possibly indexed or primed.

$$\alpha ::= \bar{a}\langle b\rangle \mid a(x) \mid \tau$$

Processes, Proc**, ranged over by** $P, Q, \ldots$

$$P ::= 0 \mid \bar{a}\langle b\rangle.P \mid a(x).P \mid *a(x).P \mid (\mathsf{new}\, a)P \mid [a = b]P \mid P|Q$$

The rigorous explanation of each is its transition rule.

Free and bound names

$$
\begin{aligned}
\mathrm{fn}(0) &= \emptyset \\
\mathrm{fn}(\overline{a}\langle b\rangle.P) &= \{a, b\} \cup \mathrm{fn}(P) \\
\mathrm{fn}(a(x).P) &= \{a\} \cup \mathrm{fn}(P) \setminus \{x\} \\
\mathrm{fn}(*a(x).P) &= \{a\} \cup \mathrm{fn}(P) \setminus \{x\} \\
\mathrm{fn}((\mathrm{new}\, a)P) &= \mathrm{fn}(P) \setminus \{a\} \\
\mathrm{fn}([a = b]P) &= \{a, b\} \cup \mathrm{fn}(P) \\
\mathrm{fn}(P \mid Q) &= \mathrm{fn}(P) \cup \mathrm{fn}(Q)
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{bn}(0) &= \emptyset \\
\mathrm{bn}(\overline{a}\langle b\rangle.P) &= \mathrm{bn}(P) \\
\mathrm{bn}(a(x).P) &= \{x\} \cup \mathrm{bn}(P) \\
\mathrm{bn}(*a(x).P) &= \{x\} \cup \mathrm{bn}(P) \\
\mathrm{bn}((\mathrm{new}\, a)P) &= \{a\} \cup \mathrm{bn}(P) \\
\mathrm{bn}([a = b]P) &= \mathrm{bn}(P) \\
\mathrm{bn}(P \mid Q) &= \mathrm{bn}(P) \cup \mathrm{bn}(Q)
\end{aligned}
$$

A labelled transition system for the π-calculus

Let's "just" adapt the one of CCS, for starters...

The prefix and synchronisation rules look that this:

$$\frac{}{\alpha.P \xrightarrow{\alpha} P}\ [\mathsf{Pref}] \qquad \frac{Q \xrightarrow{\overline{a}\langle b\rangle} Q' \quad P \xrightarrow{a(x)} P'}{(Q \mid P) \xrightarrow{\tau} (Q' \mid P'\{x \leftarrow b\})}\ [\mathsf{L\text{-}Sync}]$$

A labelled transition system for the π-calculus

Let's "just" adapt the one of CCS, for starters...
The prefix and synchronisation rules look that this:

$$\frac{}{\alpha.P \xrightarrow{\alpha} P}\ [\text{Pref}] \qquad \frac{Q \xrightarrow{\overline{a}\langle b\rangle} Q' \quad P \xrightarrow{a(x)} P'}{(Q \mid P) \xrightarrow{\tau} (Q' \mid P'\{x \leftarrow b\})}\ [\text{L-Sync}]$$

but wait, in the case of input, the rule [Pref] "frees" the bound name x in P... process P has a free variable?!

A labelled transition system for the π-calculus

Let's "just" adapt the one of CCS, for starters...
The prefix and synchronisation rules look that this:

$$\frac{}{\alpha.P \xrightarrow{\alpha} P} \text{[Pref]} \qquad \frac{Q \xrightarrow{\overline{a}\langle b\rangle} Q' \quad P \xrightarrow{a(x)} P'}{(Q \mid P) \xrightarrow{\tau} (Q' \mid P'\{x \leftarrow b\})} \text{[L-Sync]}$$

but wait, in the case of input, the rule [Pref] "frees" the bound name x in P... process P has a free variable?!
We need α-conversion:

$$\frac{Q \xrightarrow{\alpha} P' \quad P =_\alpha Q}{P \xrightarrow{\alpha} P'} \text{[Alpha]}$$

Bisimulations for the π-calculus

(Ground) Bisimulation

is a symmetric binary relation $\mathcal{R}$ on processes such that, whenever $(P, Q) \in \mathcal{R}$ it holds that

$$P \xrightarrow{\alpha} P' \text{ implies } Q \xrightarrow{\alpha} Q' \text{ for some } Q' \text{ such that } (P', Q')$$

(Ground) Bisimulation

is a symmetric binary relation $\mathcal{R}$ on processes such that, whenever $(P, Q) \in \mathcal{R}$ it holds that

$$P \xrightarrow{\alpha} P' \text{ implies } Q \xrightarrow{\alpha} Q' \text{ for some } Q' \text{ such that } (P', Q')$$

Is it a congruence?

(Ground) Bisimulation

is a symmetric binary relation $\mathcal{R}$ on processes such that, whenever $(P, Q) \in \mathcal{R}$ it holds that

$$P \xrightarrow{\alpha} P' \text{ implies } Q \xrightarrow{\alpha} Q' \text{ for some } Q' \text{ such that } (P', Q')$$

Is it a congruence? Consider

$$a(x).[x = b]\overline{b}\langle b\rangle.0 \sim a(x).0$$

since

$$\frac{P \xrightarrow{\alpha} P'}{[a = a]P \xrightarrow{\alpha} P'} \text{ [Match]}$$

Bisimulations for the π-calculus

(Ground) Bisimulation

is a symmetric binary relation $\mathcal{R}$ on processes such that, whenever $(P, Q) \in \mathcal{R}$ it holds that

$$P \xrightarrow{\alpha} P' \text{ implies } Q \xrightarrow{\alpha} Q' \text{ for some } Q' \text{ such that } (P', Q')$$

Is it a congruence? Consider

$$a(x).[x = b]\overline{b}\langle b\rangle.0 \sim a(x).0$$

since

$$\frac{P \xrightarrow{\alpha} P'}{[a = a]P \xrightarrow{\alpha} P'} \text{[Match]}$$

but

$$\overline{a}\langle b\rangle.0 \mid a(x).[x = b]\overline{b}\langle b\rangle.0 \not\sim \overline{a}\langle b\rangle.0 \mid a(x).0$$

The quest for a congruence relation for the π-calculus

Name instantiation must be taken into account

If $\alpha \in \{\bar{a}\langle b\rangle, \tau\}$, the clauses are the same as before

Name instantiation must be taken into account

If $\alpha \in \{\overline{a}\langle b\rangle, \tau\}$, the clauses are the same as before, but in the case of input there are two possibilities.

- Late bisimulation, $\sim_l$: $P \xrightarrow{a(x)} P'$ implies $Q \xrightarrow{a(x)} Q'$ for some Q' such that, for all y, $(P'\{x \leftarrow y\}, Q'\{x \leftarrow y\})$

The quest for a congruence relation for the π-calculus

Name instantiation must be taken into account

If $\alpha \in \{\overline{a}\langle b\rangle, \tau\}$, the clauses are the same as before,
but in the case of input there are two possibilities.

- Late bisimulation, $\sim_l$: $P \xrightarrow{a(x)} P'$ implies $Q \xrightarrow{a(x)} Q'$ for some Q' such that, for all y, $(P'\{x \leftarrow y\}, Q'\{x \leftarrow y\})$
- Early bisimulation, $\sim_e$, inverts the quantification $P \xrightarrow{a(x)} P'$ implies that, for all y there is some Q' such that $Q \xrightarrow{a(x)} Q'$ and $(P'\{x \leftarrow y\}, Q'\{x \leftarrow y\})$

The quest for a congruence relation for the π-calculus

Name instantiation must be taken into account

If $\alpha \in \{\overline{a}\langle b\rangle, \tau\}$, the clauses are the same as before, but in the case of input there are two possibilities.

- Late bisimulation, $\sim_l$: $P \xrightarrow{a(x)} P'$ implies $Q \xrightarrow{a(x)} Q'$ for some Q' such that, for all y, $(P'\{x \leftarrow y\}, Q'\{x \leftarrow y\})$
- Early bisimulation, $\sim_e$, inverts the quantification $P \xrightarrow{a(x)} P'$ implies that, for all y there is some Q' such that $Q \xrightarrow{a(x)} Q'$ and $(P'\{x \leftarrow y\}, Q'\{x \leftarrow y\})$

They are different: $\sim_l \subset \sim_e$ (distinguishing process on page 8 of [1])

[1] R. Milner, J. Parrow, and D. Walker. A Calculus of Mobile Processes, II. Information and Computation 100, 41-77 (1992)

The quest for a congruence relation for the π-calculus

Name instantiation must be taken into account

If $\alpha \in \{\overline{a}\langle b\rangle, \tau\}$, the clauses are the same as before, but in the case of input there are two possibilities.

- Late bisimulation, $\sim_l$: $P \xrightarrow{a(x)} P'$ implies $Q \xrightarrow{a(x)} Q'$ for some Q' such that, for all y, $(P'\{x \leftarrow y\}, Q'\{x \leftarrow y\})$
- Early bisimulation, $\sim_e$, inverts the quantification $P \xrightarrow{a(x)} P'$ implies that, for all y there is some Q' such that $Q \xrightarrow{a(x)} Q'$ and $(P'\{x \leftarrow y\}, Q'\{x \leftarrow y\})$

Since now $a(x).[x = b]\overline{b}\langle b\rangle.0 \not\sim_e a(x).0$

Name instantiation must be taken into account

If $\alpha \in \{\overline{a}\langle b\rangle, \tau\}$, the clauses are the same as before, but in the case of input there are two possibilities.

- Late bisimulation, $\sim_l$: $P \xrightarrow{a(x)} P'$ implies $Q \xrightarrow{a(x)} Q'$ for some Q' such that, for all y, $(P'\{x \leftarrow y\}, Q'\{x \leftarrow y\})$
- Early bisimulation, $\sim_e$, inverts the quantification $P \xrightarrow{a(x)} P'$ implies that, for all y there is some Q' such that $Q \xrightarrow{a(x)} Q'$ and $(P'\{x \leftarrow y\}, Q'\{x \leftarrow y\})$

Since now $a(x).[x = b]\overline{b}\langle b\rangle.0 \not\sim_e a(x).0$
none is a congruence!!!

“The” good bisimulation for the π-calculus

Open bisimulation

$P\mathcal{R}Q$, if for every name substitution σ and action α, whenever $P\sigma \xrightarrow{\alpha} P'$ then there is a Q' such that $Q\sigma \xrightarrow{\alpha} Q'$ and $P'\mathcal{R}Q'$

“The” good bisimulation for the π-calculus

Open bisimulation

$P\mathcal{R}Q$, if for every name substitution σ and action α, whenever $P\sigma \xrightarrow{\alpha} P'$ then there is a Q' such that $Q\sigma \xrightarrow{\alpha} Q'$ and $P'\mathcal{R}Q'$

This relation is finer than the others and IS a congruence

Impact of name instantiation in the π-calculus

Consider the rule

$$\frac{}{a(x).P \xrightarrow{a(x)} P}\ [\mathrm{In}]$$

An alternative

An LTS with the [In] rule above is called *late*

Impact of name instantiation in the π-calculus

Consider the rule

$$\frac{}{a(x).P \xrightarrow{a(x)} P}\ [\text{In}]$$

An alternative

An LTS with the [In] rule above is called *late*

An *early* LTS has instead the rule below

$$a(x).P \xrightarrow{a(b)} P\{x \leftarrow b\}$$

Impact of name instantiation in the π-calculus

Consider the rule

$$\frac{}{a(x).P \xrightarrow{a(x)} P}\ [\mathrm{In}]$$

An alternative

An LTS with the [In] rule above is called *late*
An *early* LTS has instead the rule below

$$a(x).P \xrightarrow{a(b)} P\{x \leftarrow b\}$$

But now the LTS is infinite branching

Impact of name instantiation in the π-calculus

Consider the rule

$$\frac{}{a(x).P \xrightarrow{a(x)} P} \text{[In]}$$

An alternative

An LTS with the [In] rule above is called *late*
An *early* LTS has instead the rule below

$$a(x).P \xrightarrow{a(b)} P\{x \leftarrow b\}$$

But now the LTS is infinite branching,
since there are countable possible names to choose from...

Impact of name instantiation in the π-calculus

Consider the rule

$$\frac{}{a(x).P \xrightarrow{a(x)} P}\ [\mathrm{In}]$$

An alternative

An LTS with the [In] rule above is called *late*
An *early* LTS has instead the rule below

$$a(x).P \xrightarrow{a(b)} P\{x \leftarrow b\}$$

But now the LTS is infinite branching,
since there are countable possible names to choose from...
No “free lunches”

Moreover,

Impact of name instantiation in the π-calculus

Moreover,

Two more bisimulations emerge

- Late LTS with early input transition clause

Impact of name instantiation in the π-calculus

Moreover,

Two more bisimulations emerge

- Late LTS with early input transition clause
- Early LTS with late input transition clause

They are also different and none is a congruence!!!

Impact of name instantiation in the π-calculus

Moreover,

Two more bisimulations emerge

- Late LTS with early input transition clause
- Early LTS with late input transition clause

They are also different and none is a congruence!!!

The price of expressiveness...

Bound output

Scope extrusion of bound names is the key feature of the π-calculus

Channels can be revealed

$$(\mathrm{new}\, b)\overline{a}\langle b\rangle.b(v).P \mid a(x).\overline{x}\langle 42\rangle.0 \xrightarrow{\tau} (\mathrm{new}\, b)(b(v).P \mid \overline{b}\langle 42\rangle.0)$$

Bound output

Scope extrusion of bound names is the key feature of the π-calculus

Channels can be revealed

$$(\mathsf{new}\, b)\overline{a}\langle b\rangle.b(v).P \mid a(x).\overline{x}\langle 42\rangle.0 \xrightarrow{\tau} (\mathsf{new}\, b)(b(v).P \mid \overline{b}\langle 42\rangle.0)$$

How is this transition derivable?

Bound output

Scope extrusion of bound names is the key feature of the π-calculus

Channels can be revealed

$$(\mathsf{new}\, b)\overline{a}\langle b\rangle.b(v).P \mid a(x).\overline{x}\langle 42\rangle.0 \xrightarrow{\tau} (\mathsf{new}\, b)(b(v).P \mid \overline{b}\langle 42\rangle.0)$$

How is this transition derivable?

we need more transition rules

Bound output

Scope extrusion of bound names is the key feature of the π-calculus

Channels can be revealed

$$(\mathsf{new}\, b)\overline{a}\langle b\rangle.b(v).P \mid a(x).\overline{x}\langle 42\rangle.0 \xrightarrow{\tau} (\mathsf{new}\, b)(b(v).P \mid \overline{b}\langle 42\rangle.0)$$

How is this transition derivable?
we need more transition rules, and, to avoid

$$(\mathsf{new}\, b)\overline{a}\langle b\rangle.0 \sim 0$$

Bound output

Scope extrusion of bound names is the key feature of the π-calculus

Channels can be revealed

$$(\mathsf{new}\, b)\overline{a}\langle b\rangle.b(v).P \mid a(x).\overline{x}\langle 42\rangle.0 \xrightarrow{\tau} (\mathsf{new}\, b)(b(v).P \mid \overline{b}\langle 42\rangle.0)$$

How is this transition derivable?
we need more transition rules, and, to avoid

$$(\mathsf{new}\, b)\overline{a}\langle b\rangle.0 \sim 0$$

a new label!

The full late LTS for the π-calculus

Actions are now

$\alpha ::= \tau \;\mid\; a(x) \;\mid\; \overline{a}\langle b\rangle \;\mid\; (\mathrm{new}\, b)\overline{a}\langle b\rangle$

where b is *bound* in $(\mathrm{new}\, b)\overline{a}\langle b\rangle$

The full late LTS for the π-calculus

Actions are now

$$\alpha ::= \tau \mid a(x) \mid \overline{a}\langle b\rangle \mid (\mathsf{new}\, b)\overline{a}\langle b\rangle$$

where b is *bound* in $(\mathsf{new}\, b)\overline{a}\langle b\rangle$

Remaining rules for the late LTS

$$\frac{a(x).P \xrightarrow{\alpha} P'}{*a(x).P \xrightarrow{\alpha} P' \mid *a(x).P}\ [\mathrm{Repl}]$$

The full late LTS for the π-calculus

Actions are now

$\alpha ::= \tau \;\mid\; a(x) \;\mid\; \overline{a}\langle b\rangle \;\mid\; (\mathsf{new}\, b)\overline{a}\langle b\rangle$

where b is *bound* in $(\mathsf{new}\, b)\overline{a}\langle b\rangle$

Remaining rules for the late LTS

$$\frac{a(x).P \xrightarrow{\alpha} P'}{*a(x).P \xrightarrow{\alpha} P' \mid *a(x).P}\ [\mathrm{Repl}] \qquad \frac{P \xrightarrow{\alpha} P' \quad \mathrm{bn}(\alpha) \cap \mathrm{fn}(Q) = \emptyset}{P \mid Q \xrightarrow{\alpha} P' \mid Q}\ [\text{L-Par}]$$

The full late LTS for the π-calculus

Actions are now

$\alpha ::= \tau \;\mid\; a(x) \;\mid\; \overline{a}\langle b\rangle \;\mid\; (\mathsf{new}\, b)\overline{a}\langle b\rangle$

where b is *bound* in $(\mathsf{new}\, b)\overline{a}\langle b\rangle$

Remaining rules for the late LTS

$$\frac{a(x).P \xrightarrow{\alpha} P'}{*a(x).P \xrightarrow{\alpha} P' \mid *a(x).P}\ [\text{Repl}] \qquad \frac{P \xrightarrow{\alpha} P' \quad \mathrm{bn}(\alpha) \cap \mathrm{fn}(Q) = \emptyset}{P \mid Q \xrightarrow{\alpha} P' \mid Q}\ [\text{L-Par}]$$

$$\frac{P \xrightarrow{\alpha} Q \quad a \notin \mathbf{nm}(\alpha)}{(\mathsf{new}\, a)P \xrightarrow{\alpha} (\mathsf{new}\, a)Q}\ [\text{Res}]$$

The full late LTS for the π-calculus

Actions are now

$\alpha ::= \tau \mid a(x) \mid \overline{a}\langle b\rangle \mid (\mathsf{new}\, b)\overline{a}\langle b\rangle$

where b is *bound* in $(\mathsf{new}\, b)\overline{a}\langle b\rangle$

Remaining rules for the late LTS

$$\frac{a(x).P \xrightarrow{\alpha} P'}{*a(x).P \xrightarrow{\alpha} P' \mid *a(x).P}\ [\text{Repl}] \qquad \frac{P \xrightarrow{\alpha} P' \quad \mathrm{bn}(\alpha) \cap \mathrm{fn}(Q) = \emptyset}{P \mid Q \xrightarrow{\alpha} P' \mid Q}\ [\text{L-Par}]$$

$$\frac{P \xrightarrow{\alpha} Q \quad a \notin \mathbf{nm}(\alpha)}{(\mathsf{new}\, a)P \xrightarrow{\alpha} (\mathsf{new}\, a)Q}\ [\text{Res}] \qquad \frac{P \xrightarrow{(\mathsf{new}\, a)\overline{c}\langle a\rangle} Q \quad a \neq c}{(\mathsf{new}\, a)P \xrightarrow{(\mathsf{new}\, a)\overline{c}\langle a\rangle} Q}\ [\text{Open}]$$

The full late LTS for the π-calculus

Actions are now

$\alpha ::= \tau \ \mid \ a(x) \ \mid \ \overline{a}\langle b\rangle \ \mid \ (\mathrm{new}\, b)\overline{a}\langle b\rangle$

where b is *bound* in $(\mathrm{new}\, b)\overline{a}\langle b\rangle$

Remaining rules for the late LTS

$$\frac{a(x).P \xrightarrow{\alpha} P'}{*a(x).P \xrightarrow{\alpha} P' \mid *a(x).P}\ [\text{Repl}] \qquad \frac{P \xrightarrow{\alpha} P' \quad \mathrm{bn}(\alpha) \cap \mathrm{fn}(Q) = \emptyset}{P \mid Q \xrightarrow{\alpha} P' \mid Q}\ [\text{L-Par}]$$

$$\frac{P \xrightarrow{\alpha} Q \quad a \notin \mathbf{nm}(\alpha)}{(\mathrm{new}\, a)P \xrightarrow{\alpha} (\mathrm{new}\, a)Q}\ [\text{Res}] \qquad \frac{P \xrightarrow{(\mathrm{new}\, a)\overline{c}\langle a\rangle} Q \quad a \neq c}{(\mathrm{new}\, a)P \xrightarrow{(\mathrm{new}\, a)\overline{c}\langle a\rangle} Q}\ [\text{Open}]$$

$$\frac{P \xrightarrow{(\mathrm{new}\, a)\overline{c}\langle a\rangle} P' \quad Q \xrightarrow{c(x)} Q'}{P \mid Q \xrightarrow{\tau} (\mathrm{new}\, a)(P' \mid Q'\{x \leftarrow a\})}\ [\text{L-Close}]$$

Beware of (implicit) α-conversion

We now treat formally α-conversion (following [2])

Beware of (implicit) α-conversion

We now treat formally α-conversion (following [2])
and remove the rule from the LTS,
"internalising" it in the other ones

Beware of (implicit) α-conversion

We now treat formally α-conversion (following [2])
and remove the rule from the LTS,
"internalising" it in the other ones

A name swapping $(a\ b)P$ substitutes all occurrences of a in P with b

Beware of (implicit) α-conversion

We now treat formally α-conversion (following [2])
and remove the rule from the LTS,
"internalising" it in the other ones

A name swapping $(a\ b)P$ substitutes all occurrences of a in P with b

A permutation p is a finite sequence of name swaps; applied from right to left to P is denoted $p \bullet P$

Beware of (implicit) α-conversion

We now treat formally α-conversion (following [2])
and remove the rule from the LTS,
"internalising" it in the other ones

A name swapping $(a\ b)P$ substitutes all occurrences of a in P with b

A permutation p is a finite sequence of name swaps; applied from right to left to P is denoted $p \bullet P$

An equivariant relation $\mathcal{R}$ is such that $\forall\, p\,.\, PQ\,.\,(P, Q) \in \mathcal{R}$ implies $(p \bullet P, p \bullet Q) \in \mathcal{R}$

Beware of (implicit) α-conversion

We now treat formally α-conversion (following [2])
and remove the rule from the LTS,
"internalising" it in the other ones

A name swapping $(a\ b)P$ substitutes all occurrences of a in P with b

A permutation p is a finite sequence of name swaps; applied from right to left to P is denoted $p \bullet P$

An equivariant relation $\mathcal{R}$ is such that $\forall\, p\,.\, PQ\,.\, (P, Q) \in \mathcal{R}$ implies $(p \bullet P, p \bullet Q) \in \mathcal{R}$

Freshness a name x is fresh for a process P,denoted $x \sharp P$, if $x \notin \mathrm{fn}(P)$

[2] J. Bengtson and J. Parrow. Formalising the π-calculus using Nominal Logic. Logical Methods in Computer Science, volume 5 (2:16), pp 1–36, 2009

Formalising α-conversion

α-equivalence is a binary relation satisfying

Restriction $(\mathsf{new}\, x)P = (\mathsf{new}\, y)Q$ implies

$$(x = y \wedge P = Q) \vee (x \neq y \wedge x \,\sharp\, Q \wedge P = (x\ y) \bullet Q)$$

Formalising α-conversion

α-equivalence is a binary relation satisfying

Restriction $(\mathsf{new}\, x)P = (\mathsf{new}\, y)Q$ implies
$(x = y \wedge P = Q) \vee (x \neq y \wedge x \mathbin{\sharp} Q \wedge P = (x\ y) \bullet Q)$

Input $a(x).P = b(y).Q$ implies $a = b \wedge$
$((x = y \wedge P = Q) \vee (x \neq y \wedge x \mathbin{\sharp} Q \wedge P = (x\ y) \bullet Q))$

A formal late LTS for the π-calculus

Transitions are elements of a binary relation

- Relate processes with *residuals* – pairs action/process
- If the action binds a name, it is also bound in the process

Rules for parallel and restriction are duplicated

$$\frac{P \xrightarrow{(\mathsf{new}\, a)\overline{c}\langle a\rangle} P' \quad a \sharp Q}{P \mid Q \xrightarrow{(\mathsf{new}\, a)\overline{c}\langle a\rangle} P' \mid Q}\ \text{[L-Par-B]}$$

A formal late LTS for the π-calculus

Transitions are elements of a binary relation

- Relate processes with *residuals* – pairs action/process
- If the action binds a name, it is also bound in the process

Rules for parallel and restriction are duplicated

$$\frac{P \xrightarrow{(\mathsf{new}\, a)\overline{c}\langle a\rangle} P' \quad a \sharp Q}{P \mid Q \xrightarrow{(\mathsf{new}\, a)\overline{c}\langle a\rangle} P' \mid Q} \text{ [L-Par-B]}$$

$$\frac{P \xrightarrow{\alpha} Q \quad a \sharp \alpha}{(\mathsf{new}\, a)P \xrightarrow{\alpha} (\mathsf{new}\, a)Q} \text{ [Res-F]}$$

A formal late LTS for the π-calculus

Transitions are elements of a binary relation

- Relate processes with *residuals* – pairs action/process
- If the action binds a name, it is also bound in the process

Rules for parallel and restriction are duplicated

$$\frac{P \xrightarrow{(\mathsf{new}\, a)\overline{c}\langle a\rangle} P' \quad a \sharp Q}{P \mid Q \xrightarrow{(\mathsf{new}\, a)\overline{c}\langle a\rangle} P' \mid Q}\ \text{[L-Par-B]}$$

$$\frac{P \xrightarrow{\alpha} Q \quad a \sharp \alpha}{(\mathsf{new}\, a)P \xrightarrow{\alpha} (\mathsf{new}\, a)Q}\ \text{[Res-F]} \qquad \frac{P \xrightarrow{(\mathsf{new}\, x)\overline{a}\langle x\rangle} Q \quad y \neq a \wedge y \neq x}{(\mathsf{new}\, y)P \xrightarrow{(\mathsf{new}\, x)\overline{a}\langle x\rangle} (\mathsf{new}\, y)Q}\ \text{[Res-B]}$$

A formal late LTS for the π-calculus

Transitions are elements of a binary relation

- Relate processes with *residuals* – pairs action/process
- If the action binds a name, it is also bound in the process

Rules for parallel and restriction are duplicated

$$\frac{P \xrightarrow{(\mathrm{new}\, a)\overline{c}\langle a\rangle} P' \quad a \sharp Q}{P \mid Q \xrightarrow{(\mathrm{new}\, a)\overline{c}\langle a\rangle} P' \mid Q} \text{ [L-Par-B]}$$

$$\frac{P \xrightarrow{\alpha} Q \quad a \sharp \alpha}{(\mathrm{new}\, a)P \xrightarrow{\alpha} (\mathrm{new}\, a)Q} \text{ [Res-F]} \qquad \frac{P \xrightarrow{(\mathrm{new}\, x)\overline{a}\langle x\rangle} Q \quad y \neq a \wedge y \neq x}{(\mathrm{new}\, y)P \xrightarrow{(\mathrm{new}\, x)\overline{a}\langle x\rangle} (\mathrm{new}\, y)Q} \text{ [Res-B]}$$

$$\frac{P \xrightarrow{(\mathrm{new}\, a)\overline{c}\langle a\rangle} P' \quad Q \xrightarrow{c(x)} Q' \quad a \sharp Q}{P \mid Q \xrightarrow{\tau} (\mathrm{new}\, a)(P' \mid Q'\{x \leftarrow a\})} \text{ [L-Close]}$$