About the lectures
The MPRI represents a transition from student to researcher. So...

- Interrupting me with questions is good.
- Working through a problem without already knowing the answer is good.
- I’ll make mistakes. 8-)

About me

- 1995–2001: Ph.D. student of Robin Milner’s in Cambridge, UK
- 2001–2002: Postdoc in INRIA Rocquencourt, France
- 2002–: Research scientist in INRIA Rocquencourt, France
- November 2004: voted against W (who, despite this, was elected for the first time)

Books


Tutorials available online

- Joachim Parrow. “An introduction to the pi-calculus”. http://user.it.uu.se/~joachim/intro.ps

Today’s plan

- syntax
- reduction semantics and structural congruence
- labelled transitions
- bisimulation
Syntax

\[ P ::= \exists y . P \quad \text{output} \]
\[ P ::= x(y).P \quad \text{input} \ (y \text{ binds in } P) \]
\[ \nu x . P \quad \text{restriction (new)} \ (x \text{ binds in } P) \]
\[ P \parallel P' \quad \text{parallel (par)} \]
\[ \emptyset \quad \text{empty} \]
\[ !P \quad \text{replication (bang)} \]

... Significant difference from CCS: channels carry names.

Free names

The free names of \( P \) are written \( \text{fn}(P) \).

\text{Example: } \text{fn}(\emptyset) = \emptyset; \text{fn}(\exists y . z(y) . 0) = \{x, y, z\}.

\text{Exercise: Calculate } \text{fn}(z(y) . x(y) . 0); \text{fn}(\nu z . (z(y) . \exists y) | yz).

Formally:

\[
\begin{align*}
\text{fn}(\exists y . P) & = \{x, y\} \cup \text{fn}(P) \\
\text{fn}(x(y).P) & = \{x\} \cup \left(\text{fn}(P) \setminus \{y\}\right) \\
\text{fn}(\nu x . P) & = \text{fn}(P) \setminus \{x\} \\
\text{fn}(P | P') & = \text{fn}(P) \cup \text{fn}(P') \\
\text{fn}(0) & = \emptyset \\
\text{fn}(!P) & = \text{fn}(P)
\end{align*}
\]

Alpha-conversion

We consider processes up to alpha-conversion: provided \( y' \notin \text{fn}(P) \), we have

\[
\begin{align*}
x(y).P & = x(y').\{y'/y\}P \\
\nu y . P & = \nu y'.\{y'/y\}P
\end{align*}
\]

\text{Exercise: Freshen all bound names: } \nu x . (x(x).\exists x) | x(x)

Structural congruence \((\equiv)\)

The smallest equivalence relation such that:

\[
\begin{align*}
P | (Q | S) & \equiv (P | Q) | S \quad \text{(str-assoc)} \\
P | Q & \equiv Q | P \quad \text{(str-commut)} \\
P | 0 & \equiv P \quad \text{(str-id)} \\
\nu x . \nu y . P & \equiv \nu y . \nu x . P \quad \text{(str-swap)} \\
\nu x . 0 & \equiv 0 \quad \text{(str-zero)} \\
\nu x . P | Q & \equiv \nu x . (P | Q) \quad \text{if } x \notin \text{fn}(Q) \quad \text{(str-ex)} \\
!P & \equiv P | !P \quad \text{(str-repl)}
\end{align*}
\]

And congruence rules:

\[
\begin{align*}
P \equiv P' & \quad \text{(str-par-l)} \\
P | Q \equiv P' | Q \quad \text{(str-par-r)} \\
\nu x . P & \equiv \nu x . P' \quad \text{(str-new)}
\end{align*}
\]

\text{Note: we don’t close up by input or output prefixing.}

Reduction \((\rightarrow)\)

We say that \( P \) reduces to \( P' \), written \( P \rightarrow P' \), if this can be derived from the following rules:

\[
\begin{align*}
\exists y . P | x(u).Q & \rightarrow P | \{y/u\}Q \quad \text{(red-comm)} \\
P | P' & \rightarrow P | P' \quad \text{(red-par)} \\
P | Q & \rightarrow P' | Q \quad \text{(red-new)} \\
\nu x . P & \rightarrow \nu x . P' \quad \text{(red-new)}
\end{align*}
\]

\text{Example: } \nu x . (\exists y | x(u) . \exists z) \rightarrow \nu x . (!0 | \exists z)

As currently defined, reduction is too limited:

\[
\begin{align*}
(\exists y | \emptyset) | x(u) & \not\rightarrow \\
\nu w . \exists y | x(u) & \not\rightarrow
\end{align*}
\]
Fixing reduction

We close reduction by structural congruence:

\[
\frac{P \equiv \equiv P'}{P \rightarrow \rightarrow P'} \quad \text{(red-str)}
\]

Exercise: Calculate the reductions of \(\nu y.(\pi y \mid y(u).\pi z) \mid x(w).\pi v\) and \(\pi y \mid \nu y.(x(u).\pi w \mid y(v))\)

Application of new binding: from polyadic to monadic channels

Let us extend our notion of monadic channels, which carry exactly one name, to polyadic channels, which carry a vector of names, i.e.

\[
P ::= x \langle y_1, \ldots, y_n \rangle.P \quad \text{output}
\]
\[
x(y_1, \ldots, y_n).P \quad \text{input} \quad (y_1, \ldots, y_n \text{ bind in } P)
\]

Is there an encoding from polyadic to monadic channels? We might try:

\[
[\pi \langle y_1, \ldots, y_n \rangle.P] = \pi y_1 \ldots \pi y_n.[P]
\]
\[
[x(y_1, \ldots, y_n).P] = x(y_1) \ldots x(y_n).[P]
\]

but this is broken! Can you see why? The right approach is use new binding:

\[
[\pi \langle y_1, \ldots, y_n \rangle.P] = \nu z.(\pi z \pi y_1 \ldots \pi y_n.[P])
\]
\[
[x(y_1, \ldots, y_n).P] = x(z) z(y_1) \ldots z(y_n).[P]
\]

where \(z \notin \text{fn}(P)\) in both cases. (We also need some well-sorted assumptions.)

Labels

The labels \(\alpha\) are of the form:

\[
\alpha ::= \pi y \quad \text{output}
\]
\[
x(y) \quad \text{bound output}
\]
\[
xy \quad \text{input}
\]
\[
\tau \quad \text{silent}
\]

The free names \(\text{fn}(\alpha)\) and bound names \(\text{bn}(\alpha)\) are defined as follows:

<table>
<thead>
<tr>
<th>(\alpha)</th>
<th>(\pi y)</th>
<th>(\pi (y))</th>
<th>(xy)</th>
<th>(\tau)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\text{fn}(\alpha))</td>
<td>({x, y})</td>
<td>({x})</td>
<td>({x, y})</td>
<td>(\emptyset)</td>
</tr>
<tr>
<td>(\text{bn}(\alpha))</td>
<td>(\emptyset)</td>
<td>({y})</td>
<td>(\emptyset)</td>
<td>(\emptyset)</td>
</tr>
</tbody>
</table>
Labelled transitions \((P \xrightarrow{\alpha} P')\)

Labelled transitions are of the form \(P \xrightarrow{\alpha} P'\) and are generated by:

\[
\begin{align*}
\pi y. P & \rightarrow P \quad \text{(lab-out)} \\
\nu y. P & \rightarrow \nu y. P' \quad \text{(lab-in)} \\
\frac{P}{P} \rightarrow P' \quad \text{if } \text{bn}(\alpha) \cap \text{fn}(Q) = \emptyset \quad \text{(lab-par-l)} \\
\frac{P \rightarrow P'}{\nu y. P \rightarrow \nu y. P'} \quad \text{if } y \notin \text{fn}(Q) \cup \text{bn}(\alpha) \quad \text{(lab-new)} \\
\frac{P \rightarrow P'}{\nu y. P \rightarrow \nu y. P'} \quad \text{if } y \notin x \quad \text{(lab-open)} \\
\frac{P \rightarrow P'}{\nu y. P \rightarrow \nu y. P'} \quad \text{if } y \notin \text{fn}(Q) \quad \text{(lab-comm-l)} \\
\frac{P \rightarrow P'}{\nu y. P \rightarrow \nu y. P'} \quad \text{if } y \notin \text{fn}(Q) \quad \text{(lab-close-l)} \\
\frac{P \rightarrow P'}{!P \rightarrow P'} \quad \text{(lab-bang)}
\end{align*}
\]

plus symmetric rules (lab-par-r), (lab-comm-r), (lab-close-r).

Adding sum

\[
\begin{align*}
P & ::= M \\
& | P \parallel P \quad \text{parallel (par)} \\
& | \nu x. P \quad \text{restriction (new) \(x\) binds in \(P\)} \\
& | \nu y. P \quad \text{replication (bang)} \\
M & ::= \pi y. P \\
& | x(y). P \quad \text{output \((y\) binds in \(P\)} \\
& | M + M \quad \text{sum} \\
& | 0
\end{align*}
\]

Changes:

- structural congruence: + is associative and commutative with identity 0.
- reduction: \((\pi y. P + M) \mid (x(u).Q + N) \rightarrow P | \{y/u\}Q\).
- labelled transition: \(M + \pi y. P \rightarrow \pi y. P \quad (M + x(y).P + N \rightarrow_{\pi y} P)

\[M + x(y).P + N \rightarrow_{\pi y} \{z/y\}P\]

Labelled transitions and structural congruence

**Theorem:**

1. \(P \rightarrow P'\) iff \(P \xrightarrow{\tau} \equiv P'\).
2. \(P \equiv \alpha \rightarrow P'\) implies \(P \xrightarrow{\alpha} \equiv P'\)

**Exercise:** Why does the converse of the second not hold?

**Exercise:** Show that the following pair of processes are both in \((\rightarrow)\) and \((\xrightarrow{\tau} \equiv)\):

\[
\nu z. xz \mid x(u).y \quad \nu z.z
\]

**Fun with side conditions**

**Exercise:** Show that the side condition on (lab-par-l) is necessary by considering the process \(\nu y. (\pi y. y(u)) | \pi v\) and an alpha variant.

Exercises for next lecture

1. Define an encoding \([\ ]\) from the monadic synchronous \(\pi\)-calculus to the monadic asynchronous \(\pi\)-calculus.

2. Prove that if \(P \xrightarrow{\pi y} P'\) then there exist \(P_0, P_1, \text{ and } z\) such that

\[
P \equiv \nu \bar{z}. (\pi y. P_0 | P_1)
\]

\[
P' \equiv \nu \bar{z}. (P_0 | P_1)
\]

\[
\{x, y\} \cap \bar{z} = \emptyset
\]

NB: the notation \(\nu \bar{z}. P\) is merely a convenient way of expressing a series of new bindings:

\[
\nu \bar{z}. P = \begin{cases} 
P & \text{if } \bar{z} \text{ is empty} \\
\nu w. (\nu \bar{w}. P) & \text{if } \bar{z} = \bar{w}
\end{cases}
\]