MPRI Concurrency (course number 2-3) 2004-2005: π-calculus

16 November 2004

http://pauillac.inria.fr/~leifer/teaching/mpri-concurrency-2004/

James J. Leifer
INRIA Rocquencourt

James.Leifer@inria.fr

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

- Robin Milner. *Communicating and mobile systems: the π-calculus*. (Cambridge University Press, 1999).
- Robin Milner. *Communication and concurrency*. (Prentice Hall, 1989).
- Davide Sangiorgi and David Walker. *The π-calculus: a theory of mobile processes*. (Cambridge University Press, 2001).

Tutorials available online

- Robin Milner. "The polyadic pi-calculus: a tutorial". Technical Report ECS-LFCS-91-180, University of Edinburgh. http://www.lfcs.inf.ed.ac.uk/reports/91/ECS-LFCS-91-180/ECS-LFCS-91-180.ps
- Joachim Parrow. "An introduction to the pi-calculus". http://user.it.uu.se/~joachim/intro.ps
- Peter Sewell. "Applied pi — a brief tutorial". Technical Report 498, University of Cambridge. http://www.cl.cam.ac.uk/users/pes20/apppi.ps

Today's plan

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

Syntax

$$
\begin{array}{lll}
P ::= & \overline{x}y.P & \text{output} \\
& x(y).P & \text{input } (y \text{ binds in } P) \\
& \boldsymbol{\nu}x.P & \text{restriction (new) } (x \text{ binds in } P) \\
& P \mid P & \text{parallel (par)} \\
& \mathbf{0} & \text{empty} \\
& !P & \text{replication (bang)} \\
& \ldots &
\end{array}
$$

Significant difference from CCS: channels carry names.

Free names

The free names of P are written $\mathsf{fn}(P)$.
Example: $\mathsf{fn}(\mathbf{0}) = \varnothing$; $\mathsf{fn}(\overline{x}y.z(y).\mathbf{0}) = \{x, y, z\}$.
Exercise: Calculate $\mathsf{fn}(z(y).\overline{x}y.\mathbf{0})$; $\mathsf{fn}(\boldsymbol{\nu}z.(z(y).\overline{x}y) \mid \overline{y}z)$.
Formally:

$$
\begin{array}{ll}
\mathsf{fn}(\overline{x}y.P) & = \{x, y\} \cup \mathsf{fn}(P) \\
\mathsf{fn}(x(y).P) & = \{x\} \cup (\mathsf{fn}(P) \setminus \{y\}) \\
\mathsf{fn}(\boldsymbol{\nu}x.P) & = \mathsf{fn}(P) \setminus \{x\} \\
\mathsf{fn}(P \mid P') & = \mathsf{fn}(P) \cup \mathsf{fn}(P') \\
\mathsf{fn}(\mathbf{0}) & = \varnothing \\
\mathsf{fn}(!P) & = \mathsf{fn}(P)
\end{array}
$$

Alpha-conversion

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

$$x(y).P = x(y').\{y'/y\}P$$
$$\boldsymbol{\nu}y.P = \boldsymbol{\nu}y'.\{y'/y\}P$$

Exercise: Freshen all bound names: $\boldsymbol{\nu}x.(x(x).\overline{x}x) \mid x(x)$

Reduction ($\longrightarrow$)

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

$$\overline{x}y.P \mid x(u).Q \longrightarrow P \mid \{y/u\}Q \qquad \text{(red-comm)}$$

$$\frac{P \longrightarrow P'}{P \mid Q \longrightarrow P' \mid Q} \qquad \text{(red-par)}$$

$$\frac{P \longrightarrow P'}{\boldsymbol{\nu}x.P \longrightarrow \boldsymbol{\nu}x.P'} \qquad \text{(red-new)}$$

Example: $\boldsymbol{\nu}x.(\overline{x}y \mid x(u).\overline{u}z) \longrightarrow \boldsymbol{\nu}x.(\mathbf{0} \mid \overline{y}z)$

As currently defined, reduction is too limited:

$$(\overline{x}y \mid \mathbf{0}) \mid x(u) \not\longrightarrow$$
$$\boldsymbol{\nu}w.\overline{x}y \mid x(u) \not\longrightarrow$$

Structural congruence ($\equiv$)

$$P \mid (Q \mid S) \equiv (P \mid Q) \mid S \qquad \text{(str-assoc)}$$
$$P \mid Q \equiv Q \mid P \qquad \text{(str-commut)}$$
$$P \mid \mathbf{0} \equiv P \qquad \text{(str-id)}$$
$$\boldsymbol{\nu}x.\boldsymbol{\nu}y.P \equiv \boldsymbol{\nu}y.\boldsymbol{\nu}x.P \qquad \text{(str-swap)}$$
$$\boldsymbol{\nu}x.\mathbf{0} \equiv \mathbf{0} \qquad \text{(str-zero)}$$
$$\boldsymbol{\nu}x.P \mid Q \equiv \boldsymbol{\nu}x.(P \mid Q) \quad \text{if } x \notin \mathsf{fn}(Q) \qquad \text{(str-ex)}$$
$$!P \equiv P \mid !P \qquad \text{(str-repl)}$$

We close reduction by structural congruence:

$$\frac{P \equiv\longrightarrow\equiv P'}{P \longrightarrow P'} \qquad \text{(red-str)}$$

Exercise: Calculate the reductions of $\boldsymbol{\nu}y.(\overline{x}y \mid y(u).\overline{u}z) \mid x(w).\overline{w}v$ and $\overline{x}y \mid \boldsymbol{\nu}y.(x(u).\overline{u}w \mid y(v))$

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.

$$\begin{array}{rll} P ::= & \overline{x}\langle y_1, ..., y_n\rangle.P & \text{output} \\ & x(y_1, ..., y_n).P & \text{input } (y_1, ..., y_n \text{ bind in } P) \end{array}$$

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

$$[\![\overline{x}\langle y_1, ..., y_n\rangle.P]\!] = \overline{x}y_1....\overline{x}y_n.[\![P]\!]$$
$$[\![x(y_1, ..., y_n).P]\!] = x(y_1)....x(y_n).[\![P]\!]$$

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

$$[\![\overline{x}\langle y_1, ..., y_n\rangle.P]\!] = \boldsymbol{\nu}z.(\overline{x}z.\overline{z}y_1....\overline{z}y_n.[\![P]\!])$$
$$[\![x(y_1, ..., y_n).P]\!] = x(z).z(y_1)....z(y_n).[\![P]\!]$$

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

Application of new binding: from synchronous to asynchronous ouput

In distributed computing, sending and receiving messages may be asymmetric: we clearly know when we have received a message but not necessarily when a message we sent has been delivered. (Think of email.)

$$\begin{array}{rll} P ::= & \overline{x}y & \text{output} \\ & x(y).P & \text{input } (y \text{ binds in } P) \end{array}$$

Nonetheless, one can always achieve synchronous sends by using an *acknowledgement* protocol:

$$[\![\overline{x}y.P]\!] = \boldsymbol{\nu}z.(\overline{x}\langle y, z\rangle \mid z().[\![P]\!])$$
$$[\![x(y).P]\!] = x(y, z).(\overline{z}\langle\rangle \mid [\![P]\!])$$

provided $z \notin \mathsf{fn}(P)$ in both cases.

Labels

The labels α are of the form:

$$\begin{array}{rll} \alpha ::= & \overline{x}y & \text{output} \\ & \overline{x}(y) & \text{bound output} \\ & xy & \text{input} \\ & \tau & \text{silent} \end{array}$$

The names $\mathsf{n}(\alpha)$ and bound names $\mathsf{bn}(\alpha)$ are defined as follows:

α	$\overline{x}y$	$\overline{x}(y)$	xy	τ
$\mathsf{n}(\alpha)$	$\{x, y\}$	$\{x, y\}$	$\{x, y\}$	$\varnothing$
$\mathsf{bn}(\alpha)$	$\varnothing$	y	$\varnothing$	$\varnothing$

Labelled transitions ($P \xrightarrow{\alpha} P'$)

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

$$\overline{x}y.P \xrightarrow{\overline{x}y} P \quad \text{(lab-out)} \qquad x(y).P \xrightarrow{xz} \{z/y\}P \quad \text{(lab-in)}$$

$$\frac{P \xrightarrow{\alpha} P'}{P \mid Q \xrightarrow{\alpha} P' \mid Q} \text{if } \mathsf{bn}(\alpha) \cap \mathsf{fn}(Q) = \varnothing \quad \text{(lab-par-l)}$$

$$\frac{P \xrightarrow{\alpha} P'}{\boldsymbol{\nu}y.P \xrightarrow{\alpha} \boldsymbol{\nu}y.P'} \text{if } y \notin \mathsf{n}(\alpha) \quad \text{(lab-new)} \qquad \frac{P \xrightarrow{\overline{x}y} P'}{\boldsymbol{\nu}y.P \xrightarrow{\overline{x}(y)} P'} \text{if } y \neq x \quad \text{(lab-open)}$$

$$\frac{P \xrightarrow{\overline{x}y} P' \qquad Q \xrightarrow{xy} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'} \quad \text{(lab-comm-l)} \qquad \frac{P \xrightarrow{\overline{x}(y)} P' \qquad Q \xrightarrow{xy} Q'}{P \mid Q \xrightarrow{\tau} \boldsymbol{\nu}y.(P' \mid Q')} \text{if } y \notin \mathsf{fn}(Q) \quad \text{(lab-close-l)}$$

$$\frac{P \mid !P \xrightarrow{\alpha} P'}{!P \xrightarrow{\alpha} P'} \quad \text{(lab-bang)}$$

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

Labelled transitions and structural congruence

Theorem:

1. $P \longrightarrow P'$ iff $P \xrightarrow{\tau} \equiv P'$.
2. $P \equiv \xrightarrow{\alpha} P'$ 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 $(\longrightarrow)$ and $(\xrightarrow{\tau}\equiv)$:

$$\boldsymbol{\nu} z.\overline{x}z \mid x(u).\overline{y}u \qquad \boldsymbol{\nu} z.\overline{y}z$$

Fun with side conditions

Exercise: Show that the side condition on (lab-par-l) is necessary by considering the process $\boldsymbol{\nu} y.(\overline{x}y.y(u)) \mid \overline{z}v$ and an alpha variant.

Strong bisimulation

A relation $\mathcal{R}$ is a strong bisimulation if for all $(P, Q) \in \mathcal{R}$ and $P \xrightarrow{\alpha} P'$, where $\mathsf{bn}(\alpha) \cap \mathsf{fn}(Q) = \varnothing$, there exists Q' such that $Q \xrightarrow{\alpha} Q'$ and $(P', Q') \in \mathcal{R}$, and symmetrically.

Strong bisimilarity $\sim$ is the largest strong bisimulation.

Bisimulation proofs

Theorem: $P \equiv Q$ implies $P \sim Q$.
Can you think of a counterexample to the converse?

Some easy results:

1. $P \mid \mathbf{0} \sim P$
2. $\overline{x}y.\boldsymbol{\nu} z.P \sim \boldsymbol{\nu} z.\overline{x}y.P$, if $z \notin \{x, y\}$
3. $x(y).\boldsymbol{\nu} z.P \sim \boldsymbol{\nu} z.x(y).P$, if $z \notin \{x, y\}$
4. $!\boldsymbol{\nu} z.P \not\sim \boldsymbol{\nu} z.!P$ for some P

More difficult:

1. $\boldsymbol{\nu} x.P \mid Q \sim \boldsymbol{\nu} x.(P \mid Q)$
2. $!P \mid !P \sim !P$
3. $P \sim Q$ implies $P \mid S \sim Q \mid S$

Adding sum

$P ::= M$	sum
$\quad P \mid P$	parallel (par)
$\quad !P$	replication (bang)
$M ::= \overline{x}y.P$	output
$\quad x(y).P$	input (y binds in P)
$\quad M + M$	sum
$\quad \mathbf{0}$	

Change structural congruence to treat $+$ as associative and commutive with identity $\mathbf{0}$.

Change reduction: $(\overline{x}y.P + M) \mid (x(u).Q + N) \longrightarrow P \mid \{y/u\}Q$.

Change labelled transition: $M + \overline{x}y.P + N \xrightarrow{\overline{x}y} P$
$M + x(y).P + N \xrightarrow{xz} \{z/y\}P$