Mobile Computing

The π-calculus

Pascal Zimmer

pzimmer@daimi.au.dk

BRICS

Resources

- Robin Milner, Joachim Parrow, David Walker: *A Calculus of Mobile Processes (parts I and II)*. Information and Computation **100**(1) (1992).
- Robin Milner: *The Polyadic π-Calculus: a Tutorial*. Technical Report ECS-LFCS-91-180, University of Edinburgh (1991).
- Robin Milner: *Communicating and Mobile Systems: the Pi-Calculus*. Cambridge University Press (2000).
- Davide Sangiorgi and David Walker. *The π-Calculus: a Theory of Mobile Processes*. Cambridge University Press (2001).

Basics

- We assume an infinite enumerable set of *names*

$$a, b, \ldots, x, y, \ldots$$

- We define *terms* or *processes*

$$P, Q, \ldots$$

Example

$$P \;=\; \bar{a}v.b(x).\mathbf{0} \mid a(y).(\bar{c}y.\mathbf{0} \mid \bar{d}y.\mathbf{0})$$

Example

Reduction:

$$P \;=\; \bar{a}v.b(x).\mathbf{0} \mid a(y).(\bar{c}y.\mathbf{0} \mid \bar{d}y.\mathbf{0})$$

$$\downarrow$$

$$b(x).\mathbf{0} \mid \bar{c}v.\mathbf{0} \mid \bar{d}v.\mathbf{0}$$

Example

Reduction:

$$P \;=\; \bar{a}v.b(x).\mathbf{0} \mid a(y).(\bar{c}y.\mathbf{0} \mid \bar{d}y.\mathbf{0})$$

$$\downarrow$$

$$b(x).\mathbf{0} \mid \bar{c}v.\mathbf{0} \mid \bar{d}v.\mathbf{0}$$

Resource conflict:

$$a(x).Q_1 \mid a(x).Q_2 \mid \bar{a}v.\mathbf{0}$$

$$\swarrow \qquad\qquad \searrow$$

$$Q_1\{x \mapsto v\} \mid a(x).Q_2 \mid \mathbf{0} \qquad a(x).Q_1 \mid Q_2\{x \mapsto v\} \mid \mathbf{0}$$

Syntax

Prefixes:

- $a(b)$: reception
- $\bar{a}b$: emission
- a is the subject and b the object
- sometimes useful to consider a silent action τ ($\tau.P \longrightarrow P$)

Syntax

Prefixes:

- $a(b)$: reception
- $\bar{a}b$: emission
- a is the subject and b the object
- sometimes useful to consider a silent action τ ($\tau.P \longrightarrow P$)

Notation:

- $\bar{a}b.0$ often written $\bar{a}b$
- and sometimes $\bar{a}\langle b \rangle$

Reference passing style

$$\bar{a}c.\bar{c}v \mid a(x).x(t).\bar{r}t$$

$$\downarrow$$

$$\bar{c}v \mid c(t).\bar{r}t$$

$$\downarrow$$

$$\mathbf{0} \mid \bar{r}v$$

Reference passing style

$$\bar{a}c.\bar{c}v \mid a(x).x(t).\bar{r}t$$
$$\downarrow$$
$$\bar{c}v \mid c(t).\bar{r}t$$
$$\downarrow$$
$$\mathbf{0} \mid \bar{r}v$$

In many encodings, the first given name is the return channel for the result...

Restriction operator ν

- $(\nu a)P$: process P in which the name a is *private*
- Other interpretation: create a *fresh* name a and execute P
- Example:

$$T = (\nu a)(\bar{a}v \mid a(x).Q_1) \mid a(y).Q_2$$

  no communication is possible with Q_2
- Remark: ν is a binder; T can be α-converted into:

$$(\nu a')(\bar{a}'v \mid a'(x).Q_1\{a \mapsto a'\}) \mid a(y).Q_2$$

  where a' is a fresh name

Name extrusion

- When the object of an emission is a restricted name, we may have to extend its scope.

Name extrusion

- When the object of an emission is a restricted name, we may have to extend its scope.
- Example:

$$(\nu c)(P \mid \bar{a}c.Q) \mid a(x).R \;\longrightarrow\; (\nu c)(P \mid Q \mid R\{x \mapsto c\})$$

Name extrusion

- When the object of an emission is a restricted name, we may have to extend its scope.
- Example:

$$
\begin{aligned}
(\nu c)(P \mid \bar{a}c.Q) \mid a(x).R \;&\longrightarrow\; (\nu c)(P \mid Q \mid R\{x \mapsto c\}) \\
&\equiv\; (\nu c)(P \mid R\{x \mapsto c\}) \mid Q
\end{aligned}
$$

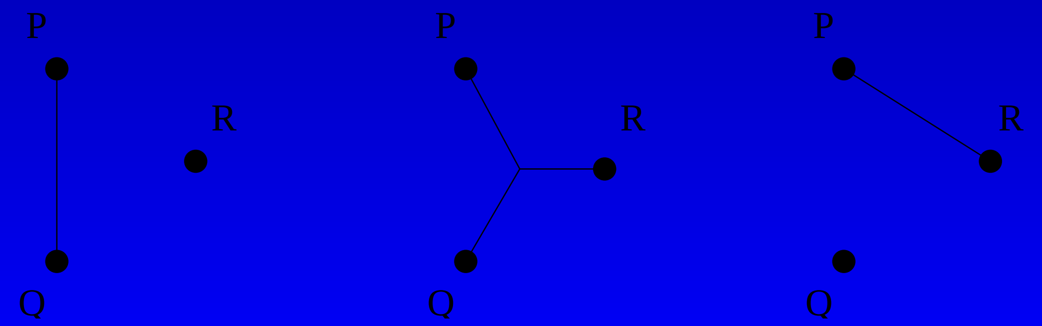
provided $c \notin fn(Q)$

Name extrusion

- When the object of an emission is a restricted name, we may have to extend its scope.
- Example:

$$\begin{aligned}(\nu c)(P \mid \bar{a}c.Q) \mid a(x).R &\rightarrow (\nu c)(P \mid Q \mid R\{x \mapsto c\}) \\ &\equiv (\nu c)(P \mid R\{x \mapsto c\}) \mid Q\end{aligned}$$

  provided $c \notin fn(Q)$
- This induces changes in the system topology:



Replication

In order to get a powerful and expressive formalism, we need a form of recursion:

$$!P$$

which is interpreted as “un unbounded number of copies of P in parellel” (i.e. $!P$ $''=''$ $P|P|P|\ldots$)

Recursive definitions

Useful for specification:

$$A(x) \stackrel{def}{=} P$$

We invoke A with the application $A\langle a\rangle$.

Replication vs recursive defs

Are replication and recursive definitions equivalent ?

Replication vs recursive defs

Are replication and recursive definitions equivalent ?

Idea: in a term Q that makes use of A:

- choose a name $t \notin fn(P) \cup fn(Q)$
- replace $A\langle a\rangle$ with $\bar{t}a$ everywhere in Q, giving $\tilde{Q}$
- execute $(\nu t)(\tilde{Q} \mid !t(x).P)$

Question: how to state and prove the correction of this encoding ?

Choice operator

$P + Q$ behaves as P *or* Q:

$$\overline{heads} + \overline{tails} \mid heads.P \mid tails.Q$$

$$\swarrow \qquad\qquad \searrow$$

$$P \mid tails.Q \qquad\qquad heads.P \mid Q$$

Choice operator

$P + Q$ behaves as P *or* Q:

$$\overline{heads} + \overline{tails} \mid heads.P \mid tails.Q$$

$$\swarrow \qquad\qquad \searrow$$

$$P \mid tails.Q \qquad\qquad\qquad heads.P \mid Q$$

Beware !

$$\overline{heads} + \overline{tails} \;\not\rightarrow\; \overline{heads}$$

i.e. the choice is delayed until a communication occurs.

Syntax summary

$$P ::= \mathbf{0} \mid a(x).P \mid \bar{a}x.P \mid (P_1|P_2) \mid (\nu a)P \mid !P \mid P + Q$$

Syntax summary

$$P ::= \mathbf{0} \mid a(x).P \mid \bar{a}x.P \mid (P_1|P_2) \mid (\nu a)P \mid !P \mid P + Q$$

This calculus is:

- monadic
- synchronous
- with replication and choice operators

Syntax summary

$$P ::= \mathbf{0} \mid a(x).P \mid \bar{a}x.P \mid (P_1|P_2) \mid (\nu a)P \mid !P \mid P+Q$$

This calculus is:

- monadic
- synchronous
- with replication and choice operators

There are numerous other possibilities.
For example, the testing of names:

$[a = b]P$	matching
$[a \neq b]P$	mismatching

Operational semantics

- Convenient to write as a *chemical abstract machine* (Berry, Boudol 1992)
- Metaphoric vision: a “soup of terms”

Operational semantics

- Convenient to write as a *chemical abstract machine* (Berry, Boudol 1992)
- Metaphoric vision: a "soup of terms"
- A reduction relation:

$$P \quad \longrightarrow \quad Q$$

to model "real" computation steps

Operational semantics

- Convenient to write as a *chemical abstract machine* (Berry, Boudol 1992)
- Metaphoric vision: a “soup of terms”
- A reduction relation:

$$P \;\rightarrow\; Q$$

to model “real” computation steps

- It makes use of a *structural congruence* relation:

$$P \;\equiv\; Q$$

to model “administrative” reorderings of terms (for example $P \mid Q \;\equiv\; Q \mid P$)

Structural congruence

- $\equiv$ contains α-conversion
- Parallel composition and choice:

$$
\begin{aligned}
&P \mid \mathbf{0} \equiv P \\
&P \mid Q \equiv Q \mid P \\
&P \mid (Q \mid R) \equiv (P \mid Q) \mid R \\
&P + \mathbf{0} \equiv P \\
&P + Q \equiv Q + P \\
&P + (Q + R) \equiv (P + Q) + R
\end{aligned}
$$

- Replication:

$$!P \mid P \equiv\, !P$$

Structural congruence

- Restriction:

$$(\nu x)\mathbf{0} \equiv \mathbf{0}$$
$$(\nu x)(\nu y)P \equiv (\nu y)(\nu x)P$$
$$(\nu x)(P \mid Q) \equiv (\nu x)P \mid Q \quad \text{if } x \notin fn(Q)$$

- Consequence:

$$(\nu x)P \equiv P \quad \text{if } x \notin fn(P)$$

- Normal form:

$$P \equiv (\nu \tilde{x})(M_1 | \ldots | M_k | !R_1 | \ldots | !R_n)$$

Reduction

$$\frac{}{(M + x(y).P) \mid (N + \bar{x}z.Q) \;\rightarrow\; P\{y \mapsto z\} \mid Q}\text{(Com)}$$

$$\frac{P \;\rightarrow\; P'}{P \mid Q \;\rightarrow\; P' \mid Q}\text{(Par)} \qquad \frac{P \;\rightarrow\; P'}{(\nu x)P \;\rightarrow\; (\nu x)P'}\text{(Res)}$$

$$\frac{Q \equiv P \qquad P \;\rightarrow\; P' \qquad P' \equiv Q'}{Q \;\rightarrow\; Q'}\text{(Struct)}$$

Example

Let $P \stackrel{def}{=} (\nu y)\bar{x}y.\mathbf{0}$

$$\begin{array}{rcl} x(z).\bar{w}z \mid !P & \equiv & x(z).\bar{w}z \mid (\nu y)\bar{x}y \mid !P \\ & \equiv & (\nu y)(x(z).\bar{w}z \mid \bar{x}y) \mid !P \\ & \rightarrow & (\nu y)(\bar{w}y \mid \mathbf{0}) \mid !P \\ & \equiv & (\nu y)\bar{w}y \mid !P \end{array}$$

Thus:

$$x(z).\bar{w}z \mid !P \;\rightarrow\; (\nu y)\bar{w}y \mid !P$$

Exercises

- How to encode the polyadic π-calculus into its monadic version ?
- How to encode synchronous communications (with $\bar{a}v.P$) in an asynchronous π-calculus (with only emissions in the form $\bar{a}v$) ?
- How to encode the behaviour of a term like $!\bar{a}v.P$ in a π-calculus where only inputs can be replicated: $!a(x).P$?