Course on
Probabilistic Methods in Concurrency
(Concurrent Languages for Probabilistic Asynchronous Communication)

Lecture 1
The pi-calculus and the asynchronous pi-calculus.

Catuscia Palamidessi
INRIA Futurs & LIX
France
catuscia@lix.polytechnique.fr
Administrativia

• **Homepage of the course:**
  
  www.lix.polytechnique.fr/~catuscia/teaching/Pisa/

  - Slides
  - Some copies of the papers/books used as references

• **Exam**

• **Schedule**
Plan of the lectures

1. The pi-calculus and the asynchronous pi-calculus
2. The pi-calculus hierarchy: encodings
   - Encoding of output prefix in the asynchronous pi-calculus
   - Encoding of input guarded choice in the asynchronous pi-calculus
3. The pi-calculus hierarchy: separation results
   - Separation between the pi-calculus and the asynchronous pi-calculus
   - Separation between the pi-calculus and CCS
4. Problems in distributed algorithms for which only randomized solutions exist
5. Basics of Measure Theory and Probability Theory
6. Probabilistic Automata
7. The probabilistic pi-calculus
8. Encoding of the pi-calculus into the asynchronous pi-calculus
9. Other uses of randomization: randomized protocols for anonymity and contract signing.
10. A proof search specification of the pi-calculus (speaker: Dale Miller)
The π-calculus

- Milner, Parrow, Walker 1989
- A concurrent calculus where the communication structure among existing processes can change over time.
  - Link mobility.
The $\pi$ calculus: scope extrusion

- A private channel name can be communicated and its scope can be extended to include the recipient
  - **Channel**: the name can be used to communicate
  - **Privacy**: no one else can interfere

- An example of link mobility:
The $\pi$ calculus: scope extrusion

• A private channel name can be communicated and its scope can be extended to include the recipient
  - Channel: the name can be used to communicate
  - Privacy: no one else can interfere

• An example of link mobility:
The $\pi$ calculus: scope extrusion

- A private channel name can be communicated and its scope can be extended to include the recipient
  - **Channel**: the name can be used to communicate
  - **Privacy**: no one else can interfere

- An example of link mobility:

![Diagram showing link mobility in the $\pi$ calculus]
The \(\pi\) calculus: some suggested bibliography

- Robin Milner. *Communicating and mobile systems: the \(\pi\)-calculus*. Cambridge University Press, 1999

- Benjamin Pierce. *Foundational Calculi for Programming Languages*. Chapter in the *CRC Handbook of Computer Science and Engineering*, 1996


The $\pi$-calculus: syntax

- Similar to CCS with value passing, but values are channel names, and recursion is replaced by replication ($!$)

$$\pi ::= x(y) \mid \overline{x}y \mid \tau$$

- action prefixes (input, output, silent)
- $x, y$ are channel names

$$P ::= O \mid \pi.P \mid P \mid P \mid P + P \mid (\nu x)P \mid ! P$$

- inaction
- prefix
- parallel
- sum
- restriction, new name
- replication
The \( \pi \)-calculus: syntax

- **Names:** \( n(P) \)
  - **Free** \( fn(P) \)
  - **Bound** \( bn(P) \)
    - Input and restriction are binders
- **Exercise:** give the formal definition of \( fn(P) \) and \( bn(P) \)

**Example:**

\[ P = ((\nu x) \bar{y} x.x(z).\bar{z}x.0) \mid (y(w).\bar{w}u.0) \]

we have: \( fn(P) = \{y, u\} \), \( bn(P) = \{x, z, w\} \)

- **Alpha conversion**

**Example:**

\[ Q = ((\nu v) \bar{y} v.v(z).\bar{z}v.0) \mid (y(x).\bar{x}u.0) \]

we have: \( P \equiv_\alpha Q \)

Pisa, 28 June 2004

Prob methods in concurrency
The $\pi$-calculus: structural equivalence

- Introduced to simplify the description of the operational semantics
  
  - If $P \equiv_\alpha Q$ then $P \equiv Q$
  - $P \mid Q \equiv Q \mid P$
  - $P + Q \equiv Q + P$
  - $!P \equiv P \mid !P$

- Some presentations include other equivalences, for instance:
  
  - $P \mid 0 \equiv P$, $(P \mid Q) \mid R \equiv P \mid (Q \mid R)$
  - $P + 0 \equiv P$, $(P + Q) + R \equiv P + (Q + R)$, $P + P \equiv P$
  - $(\nu x)(\nu y)P \equiv (\nu y)(\nu x)P$, $(\nu x)P \equiv P$ if $x \notin fn(P)$
  - $P \mid (\nu x)Q \equiv (\nu x)(P \mid Q)$ if $x \notin fn(P)$ (scope extrusion)
The $\pi$–calculus: operational semantics

- The operational semantics of the $\pi$-calculus is defined as a labeled transition system. Transitions have the form

$$P \xrightarrow{\mu} Q$$

Here $P$ and $Q$ are processes and $\mu$ is an action.

- There are various operational semantics for the $\pi$-calculus. We describe here the late semantics. Actions are defined as follows:

<table>
<thead>
<tr>
<th>$\mu$</th>
<th>kind</th>
<th>$fn(\mu)$</th>
<th>$bn(\mu)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau$</td>
<td>silent</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$x(y)$</td>
<td>(bound) input</td>
<td>${x}$</td>
<td>${y}$</td>
</tr>
<tr>
<td>$\overline{x}y$</td>
<td>free output</td>
<td>${x, y}$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$\overline{x}(y)$</td>
<td>bound output</td>
<td>${x}$</td>
<td>${y}$</td>
</tr>
</tbody>
</table>
The $\pi$–calculus: late semantics

Cong  \[ P' = P \quad P \xrightarrow{\mu} Q \quad Q = Q' \quad \frac{P' \xrightarrow{\mu} Q'}{\quad} \]

Par  \[ P \xrightarrow{\mu} P' \quad P \mid Q \xrightarrow{\mu} P' \mid Q \quad \text{bn}(\mu) \cap \text{fn}(Q) = \emptyset \]

Res  \[ P \xrightarrow{\mu} P' \quad \nu y P \xrightarrow{\mu} \nu y P' \quad y \not\in n(\mu) \]

L-Com  \[ P \xrightarrow{x(y)} P' \quad Q \xrightarrow{\bar{x} z} Q' \quad \frac{P \mid Q \xrightarrow{\tau} P'[z/y] \mid Q'}{\quad} \]

Prefix  \[ \alpha.P \xrightarrow{\alpha} P \]

Sum  \[ P \xrightarrow{\mu} P' \quad P + Q \xrightarrow{\mu} P' \]

Open  \[ P \xrightarrow{\bar{x} y} P' \quad \nu y P \xrightarrow{\bar{x}(y)} P' \quad x \neq y \]

Close  \[ P \xrightarrow{x(y)} P' \quad Q \xrightarrow{\bar{x}(y)} Q' \quad \frac{P|Q \xrightarrow{\tau} \nu y(P'|Q')}{\quad} \]

Questions: 1) Why the side condition in Par?
2) Could we write $x(z)$ in L-Com and avoid the substitution?
The \( \pi \)-calculus: early semantics

- New kind of action: free input \( xz \)
- Add E-input and replace L-Com by E-Com

\[
\text{Cong} \quad \frac{P' \equiv P \quad \frac{P \xrightarrow{\mu} Q \quad Q \equiv Q'}{P' \xrightarrow{\mu} Q'}}
\]

\[
\begin{align*}
\text{E-Input} \quad & x(y).P \xrightarrow{xz} P[z/y] \\
\text{Par} \quad & \frac{P \xrightarrow{\mu} P'}{P \parallel Q \xrightarrow{\mu} P' \parallel Q} \\
\text{Res} \quad & \frac{P \xrightarrow{\mu} P'}{\nu y P \xrightarrow{\mu} \nu y P'} \\
\text{E-Com} \quad & \frac{P \xrightarrow{xy} P' \quad Q \xrightarrow{xy} Q'}{P \parallel Q \xrightarrow{\tau} P' \parallel Q'}
\end{align*}
\]

\[
\begin{align*}
\text{Prefix} \quad & \alpha.P \xrightarrow{\alpha} P \\
\text{Sum} \quad & \frac{P \xrightarrow{\mu} P'}{P + Q \xrightarrow{\mu} P'} \\
\text{Open} \quad & \frac{P \xrightarrow{\bar{x}(y)} P'}{\nu y P \xrightarrow{\bar{x}(y)} P'} \\
\text{Close} \quad & \frac{P \xrightarrow{x(y)} P' \quad \bar{x}(y) \xrightarrow{Q'} Q'}{P \parallel Q \xrightarrow{\tau} \nu y (P' \parallel Q')}
\end{align*}
\]
The $\pi$–calculus: late bisimulation

- **Definition** We say that a binary relation $S$ is a late simulation if $P S Q$ implies that

1. if $P \xrightarrow{\mu} P'$ and $\mu$ is $\tau$ or output, with $bn(\mu) \cap fn(P,Q) = \emptyset$, then for some $Q'$, $Q \xrightarrow{\mu} Q'$ and $P' S Q'$.

2. if $P \xrightarrow{x(y)} P'$ and $y \notin fn(P,Q) = \emptyset$, then for some $Q'$, $Q \xrightarrow{x(y)} Q'$ and for all $z$, $P'\{z/y\} S Q'\{z/y\}$.

- The relation $S$ is a late bisimulation iff both $S$ and $S^{-1}$ are late simulations.

- $P$ and $Q$ are late bisimilar, notation $P \sim_L Q$, iff $P S Q$ for some late bisimulation $S$. 

Pisa, 28 June 2004
The $\pi$–calculus: early bisimulation

Definition

- We say that a binary relation $S$ is an early simulation if $P S Q$ implies that

  \[ \text{if } P \xrightarrow{\mu} P' \text{ and } \mu \text{ is any action with } bn(\mu) \cap fn(P, Q) = \emptyset, \text{ then for some } Q', Q \xrightarrow{\mu} Q' \text{ and } P' S Q'. \]

- The relation $S$ is an early bisimulation iff both $S$ and $S^{-1}$ are early simulations.

- $P$ and $Q$ are early bisimilar, notation $P \sim_E Q$, iff $P S Q$ for some early bisimulation $S$. 
Late vs early bisimulation

Late bisimulation is strictly more discriminating than early bisimulation.

Example

\[ P \equiv x(y).R + x(y).0 \]

\[ Q \equiv x(y).R + x(y).0 + x(y). \text{ if } y = z \text{ then } R \text{ else } 0 \]

We have that \( P \sim_{E} Q \) but \( P \not\sim_{I} Q \)

Exercise: write a similar example without using the match operator (i.e. the if-then-else). Hint: use synchronization
Congruence

**Question:** are $\approx_L$, $\approx_E$ congruences?

**Answer:** No. Example:

\[
x(z).0 | \bar{y}z.0 \sim_E x(z).\bar{y}z.0 + \bar{y}z.x(z).0
\]

but

\[
w(x)(x(z).0 | \bar{y}z.0) \not\sim_E w(x)(x(z).\bar{y}z.0 + \bar{y}z.x(z).0)
\]

There are other equivalences which are defined to be congruences. In particular, Open bisimulation. Cfr. lecture by Dale.
The asynchronous $\pi$-calculus

- If $P \mid Q$ is interpreted as the composition of two remote processes $P$ and $Q$, then the mechanism of synchronous communication seems unrealistic

$$\bar{x}y. P \mid x(y).Q \xrightarrow{\tau} P \mid Q$$

- Synchronization combined with choice seems even less realistic

$$\bar{x}_1y. P_1 + x_2(y).P_2 \mid \bar{x}_2y. Q_1 + x_1(y).Q_2 \xrightarrow{\tau} P_1 \mid Q_2 \quad \xrightarrow{\tau} \quad P_2 \mid Q_1$$

- In a distributed system, communication is asynchronous (exchange of messages). The send takes place independently of the readiness of a receiver, and it is not blocking

- The asynchronous $\pi$-calculus: A calculus for representing asynchronous communication. It was introduced independently by Honda-Tokoro [1991] and by Boudol [1992]
The asynchronous $\pi$-calculus: syntax

- It differs from the $\pi$-calculus for the absence of the output prefix (replaced by output action) and also for the absence of the $+$

$$\pi ::= x(y) | \tau$$

action prefixes (input, silent)

$x, y$ are channel names

$$P ::= O$$
inaction

prefix

$$\pi.P$$

output action

$$\bar{x} y$$

parallel

$$P | P$$

restriction, new name

$$(\nu x)P$$

replication

$$! P$$
The asynchronous $\pi$-calculus: OS

- The operational semantics of the asynchronous $\pi$-calculus ($\pi_a$) are the same as those of the (synchronous) $\pi$-calculus ($\pi$), we only eliminate the rule for $+$ and replace the output rule with the following:

\[
\text{Output} \quad \overline{xy} \quad \overline{xy} \xrightarrow{} 0
\]

- The early and late bisimulations are obtained as usual

- The interpretation is as follows:
  - The send takes place when the output action is at the top-level
    \[(\nu y)(\overline{xy} \mid P) \mid Q\]
  - The receive takes place when the output action matches a corresponding input, i.e. when we apply the rule comm or close
    \[
    \overline{xy} \mid P \mid x(z).Q \rightarrow^* \overline{xy} \mid P' \mid x(z).Q \rightarrow P' \mid Q[z/y]
    \]