Concurrency theory

proof-techniques for synchronous and asynchronous pi-calculus

Francesco Zappa Nardelli
INRIA Rocquencourt, MOSCOVA research team

francesco.zappa_nardelli@inria.fr

together with

Frank Valencia (INRIA Futurs)  Catuscia Palamidessi (INRIA Futurs)  Roberto Amadio (PPS)

MPRI - Concurrency  October 15, 2007
Summary of last episode

- The syntax and reduction semantics of pi-calculus.
- A general and intuitive contextual equivalence.
- Relationship between lts + bisimulation and contextual equivalence.
  with proofs for CCS
Summary of actions in pi-calculus LTS

<table>
<thead>
<tr>
<th>$\ell$</th>
<th>kind</th>
<th>$\text{fn}(\ell)$</th>
<th>$\text{bn}(\ell)$</th>
<th>$\text{n}(\ell)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x\langle y \rangle$</td>
<td>free output</td>
<td>${x, y}$</td>
<td>$\emptyset$</td>
<td>${x, y}$</td>
</tr>
<tr>
<td>$(\nu y) x\langle y \rangle$</td>
<td>bound output</td>
<td>${x}$</td>
<td>${y}$</td>
<td>${x, y}$</td>
</tr>
<tr>
<td>$x(y)$</td>
<td>input</td>
<td>${x, y}$</td>
<td>$\emptyset$</td>
<td>${x, y}$</td>
</tr>
<tr>
<td>$\tau$</td>
<td>internal</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
</tbody>
</table>
Back on pi-calculus LTS

\[
\begin{align*}
\overline{x}(v).P & \xrightarrow{x(v)} P & x(y).P & \xrightarrow{x(v)} \{v/y\}P \\
\end{align*}
\]

\[
\begin{align*}
P \xrightarrow{\ell} P' & \quad \text{bn}(\ell) \cap \text{fn}(Q) = \emptyset \\
\frac{P \parallel Q \xrightarrow{\ell} P' \parallel Q}{P \parallel Q \xrightarrow{\tau} P' \parallel Q'} & \\
\end{align*}
\]

\[
\begin{align*}
P \xrightarrow{\overline{x}(v)} P' & \quad \text{if } x \neq v \\
(\nu v)P \xrightarrow{(\nu v)\overline{x}(v)} P' & \\
\end{align*}
\]

\[
\begin{align*}
P \xrightarrow{(\nu v)\overline{x}(v)} P' & \quad Q \xrightarrow{x(v)} Q' \quad \text{if } v \notin \text{fn}(Q) \\
\frac{P \parallel Q \xrightarrow{\tau} (\nu v)(P' \parallel Q')}{P \parallel Q \xrightarrow{\tau} (\nu v)(P' \parallel Q')} & \\
\end{align*}
\]
Subtleties of pi-calculus LTS

Exercise: derive a $\tau$ transition corresponding to this reduction:

$$(\nu x)\overline{a}(x).P \parallel a(y).Q \rightarrow (\nu x)(P \parallel Q\{x/y\})$$

Exercise: each side condition in the definition of the LTS is needed to have the theorem

$$P \rightarrow Q \text{ iff } P \xrightarrow{\tau} \equiv Q$$

Remove one side condition at a time and find counter-examples to this theorem.
Weak bisimulation is a sound proof technique for reduction barbed congruence

- Prove that weak bisimulation is *reduction closed*. ...at the blackboard

- Prove that weak bisimulation is *barb preserving*. ...at the blackboard

- Prove that weak-bisimulation is a congruence. ...ahem, think twice...
On soundness of weak bisimilarity

Exercise: Consider the terms (in a pi-calculus extended with +):

\[ P = \overline{x}\langle v \rangle \parallel y(z) \]
\[ Q = \overline{x}\langle v \rangle.y(z) + y(z).\overline{x}\langle v \rangle \]

1. Prove that \( P \approx Q \).

2. Does \( P \simeq Q \)?

---

1Does this hold if we replace \(+\) by \( -1 \oplus -2 = (\nu w)\langle w \rangle \parallel w(). -1 \parallel w(). -2 \) in \( Q \)?

2Hint: define a context that equates the names \( x \) and \( y \).
Bisimilarity is not a congruence

In pi-calculus, bisimilarity (both strong and weak) is not preserved by input prefixes, that is contexts of the form $C[-] = x(y)\ldots$.

Question: how to recover the soundness of the bisimilarity with respect to the reduction barbed congruence? Two solutions:

1. close the reduction barbed congruence under all non input prefix contexts;

2. close the bisimilarity under substitution: let $P \approx^c Q$ ($P$ is fully bisimilar with $Q$) if $P\sigma \approx Q\sigma$ for all substitutions $\sigma$.

Exercise: Show that $P \not\approx^c Q$, where $P$ and $Q$ are defined in the previous slide.
And completeness?

Completeness of bisimulation with respect to barbed congruence\(^3\) (closed under non-input prefixes, denoted \(\simeq^-\)) holds in the strong case. In the weak case, we have that for

\[ P = \bar{a}\langle x \rangle \parallel E_{xy} \quad Q = \bar{a}\langle y \rangle \parallel E_{xy} \]

where

\[ E_{xy} = !x(z).\bar{y}\langle z \rangle \parallel !y(z).\bar{x}\langle z \rangle \]

it holds that \(P \not\approx Q\) but \(P \simeq^- Q\) for each context \(C[-]\).

Completeness (for image-finite processes) holds if a name-matching operator is added to the language.

\(^3\)barbed congruence is a variant of reduction-closed barbed congruence in which closure under context is allowed only at the beginning of the bisimulation game.
To show that two processes are bisimilar, it is enough to find a bisimulation relating them. Easy?

Example: we want to show that (in the pi-calculus) bisimilarity is preserved by parallel composition. We naturally consider

$$\mathcal{R} = \{ (P || R, Q || R) : P \approx Q \}$$

as a candidate bisimulation. But...
The candidate bisimulation

1. may be larger than at first envisaged;

2. may be infinite;
   
   example: to show that $x(z).y(z) \approx (\nu w)(x(z).w(z) \parallel w(v).y(v))$, we must consider:
   
   $\{(x(z).y(z), (\nu w)(x(z).w(z) \parallel w(v).y(v)) ) \} 
   \bigcup \{(y\langle a \rangle, (\nu w)(w\langle a \rangle \parallel w(v).y(v)) ) : a \text{ arbitrary} \}
   
   \bigcup \{(\nu w)(0 \parallel 0) \}
   
   \bigcup \{(0, (\nu w)(0 \parallel 0)) \}

3. hard to guess;
   
   which is the smallest bisimulation relating $!!P$ and $!P$?

4. awkward to describe and to work with...
**Up-to proof techniques**

**Idea:** find classes of relations that:

1. are not themselves bisimulations;
2. can be *automatically* completed into bisimulations.

**Idea, explained:** if we had such a class then to prove that two processes are bisimilar it would be enough to exhibit a relation in this class\(^4\) that contains the two processes.

**Example:** bisimulation up to \(\equiv\) (analogous to what we did with CCS).

\(^4\)Hopefully, it is easier to find such relation than to find the candidate bisimulation directly.
Bisimulation up to non-input context

A symmetric relation $R$ is a *bisimulation up-to non-input context* if whenever $P \ R \ Q$ and $P \xrightarrow{\ell} P'$ then there exists a process $Q'$ such that $Q \xrightarrow{\hat{\ell}} Q'$ and there exist a non-input context $C[\_]$ and processes $P''$ and $Q''$ such that $P' \equiv C[P'']$, $Q' \equiv C[Q'']$, and $P'' \ R \ Q''$.

**Exercise:** Prove that if $R$ is a bisimulation up to non-input context, then

$$\{ (C[P] , C[Q]) : P \ R \ Q \text{ and } C[\_] \text{ is a non-input context}\}$$

is a bisimulation up to structural congruence.

**Exercise:** Prove that $!P \parallel !P \approx !P$ (hint: show that the relation $R = \{ (!P \parallel !P , !P) \}$ is a bisimulation up to non-input context).
Alternative LTS rules for replication

It is often convenient to replace the rule:

\[ P \parallel !P \xrightarrow{\ell} P' \]

\[ !P \xrightarrow{\ell} P' \]

with the three rules:

\[ P \xrightarrow{\ell} P' \]

\[ !P \xrightarrow{\ell} P' \parallel !P \]

\[ P \xrightarrow{\bar{x}(y)} P_1 \]

\[ P \xrightarrow{x(y)} P_2 \]

\[ !P \xrightarrow{\tau} (P_1 \parallel P_2) \parallel !P \]

\[ P \xrightarrow{(\nu y)\bar{x}(y)} P_1 \]

\[ P \xrightarrow{x(y)} P_2 \]

\[ !P \xrightarrow{\tau} (\nu y)(P_1 \parallel P_2) \parallel !P \]
The equivalence $!P \parallel !P \approx !P$ shows that duplication of a replicable resource has no behavioural effect. Consider now

$$(\nu x)(P \parallel !x(y).Q)$$

We may call $!x(y).Q$ a *private resource* of $P$. Suppose $P \equiv P_1 \parallel P_2$. It holds that

$$(\nu x)\left(P_1 \parallel P_2 \parallel !x(y).Q\right) \approx (\nu x)\left(P_1 \parallel !x(y).Q\right) \parallel (\nu x)\left(P_2 \parallel !x(y).Q\right)$$

provided that $P_1$ and $P_2$ never read over $x$. 
Intermezzo: two applications of process languages

- Protocol verification using the Mobility Workbench
  http://www.it.uu.se/research/group/mobility/mwb

- Post-hoc specification of TCP
  http://www.cl.cam.ac.uk/~pes20/Netsem

Demos
Asynchronous communication

CCS and pi-calculus (and many others) are based on synchronized interaction, that is, the acts of sending a datum and receiving it coincide:

\[
\overline{a}.P \parallel a.Q \rightarrow P \parallel Q.
\]

In real-world distributed systems, sending a datum and receiving it are distinct acts:

\[
\overline{a}.P \parallel a.Q \rightarrow \cdots \rightarrow \cdots \overline{a} \parallel P \parallel a.Q \rightarrow \cdots \rightarrow \cdots P' \parallel Q.
\]

In an asynchronous world, the prefix . does not express temporal precedence.
Asynchronous interaction made easy

Idea: the only term than can appear underneath an output prefix is $0$.

Intuition: an unguarded occurrence of $\overline{x}\langle y \rangle$ can be thought of as a datum $y$ in an implicit communication medium tagged with $x$.

Formally:

$$\overline{x}\langle y \rangle \parallel x(z).P \rightarrow P\{y/z\}.$$ 

We suppose that the communication medium has unbounded capacity and preserves no ordering among output particles.
Asynchronous pi-calculus

Syntax:

\[ P ::= 0 \mid x(y).P \mid \overline{x}(y) \mid P \parallel P \mid (\nu x)P \mid !P \]

The definitions of free and bound names, of structural congruence \(\equiv\), and of the reduction relation \(\rightarrow\) are inherited from pi-calculus.
Examples

Sequentialization of output actions is still possible:

$$(
u y, z)(\overline{x} \langle y \rangle \parallel \overline{y} \langle z \rangle \parallel \overline{z} \langle a \rangle \parallel R)$$.

Synchronous communication can be implemented by waiting for an acknowledgement:

$$[[\overline{x} \langle y \rangle . P]] = (\nu u)(\overline{x} \langle y, u \rangle \parallel u(). P)$$

$$[[x(v). Q]] = x(v, w). (\overline{w} \langle \rangle \parallel Q) \quad \text{for } w \notin Q$$

Exercise: implement synchronous communication without relying on polyadic primitives.
Contextual equivalence and asynchronous pi-calculus

It is natural to impose two constraints to the basic recipe:

• compare terms using only *asynchronous contexts*;

• restrict the observables to be *co-names*. To observe a process *is* to interact with it by performing a complementary action and reporting it: in asynchronous pi-calculus *input actions cannot be observed*. 
A peculiarity of synchronous equivalences

The terms

\[ P = !x(z) . \overline{x}(z) \]
\[ Q = 0 \]

are not reduction barbed congruent, but they are asynchronous reduction barbed congruent.

*Intuition:* in an asynchronous world, if the medium is unbound, then buffers do not influence the computation.
Consider now the weak bisimilarity $\approx_s$ built on top of the standard (early) LTS for pi-calculus. As asynchronous pi-calculus is a sub-calculus of pi-calculus, $\approx_s$ is an equivalence for asynchronous pi-calculus terms.

It holds $\approx_s \subseteq \simeq$, that is the standard pi-calculus bisimilarity is a sound proof technique for $\simeq$.

But

$$!x(z).\overline{x}(z) \not\approx_s 0.$$ 

Question: can a labelled bisimilarity recover the natural contextual equivalence?
A problem and two solutions

Transitions in an LTS should represent observable interactions a term can engage with a context:

• if \( P \xrightarrow{x(y)} P' \) then \( P \) can interact with the context \( \parallel x(u).\text{beep} \), where beep is activated if and only if the output action has been observed;

• if \( P \xrightarrow{x(y)} P' \) then in no way beep can be activated if and only if the input action has been observed!

Solutions:

1. relax the matching condition for input actions in the bisimulation game;

2. modify the LTS so that it precisely identifies the interactions that a term can have with its environment.
Idea: relax the matching condition for input actions.

Let \textit{asynchronous bisimulation} \( \approx_a \) be the largest symmetric relation such that whenever \( P \approx_a Q \) it holds:

1. if \( P \xrightarrow{\ell} P' \) and \( \ell \neq x(y) \) then there exists \( Q' \) such that \( Q \xrightarrow{\hat{\ell}} Q' \) and \( P' \approx_a Q' \);

2. if \( P \xrightarrow{x(y)} P' \) then there exists \( Q' \) such that \( Q \parallel \overline{x\langle y \rangle} \xrightarrow{} Q' \) and \( P' \approx_a Q' \).

Remark: \( P' \) is the outcome of the interaction of \( P \) with the context \(- \parallel \overline{x\langle y \rangle}\). Clause 2. allows \( Q \) to interact with the same context, but does not force this interaction.
Honda, Tokoro - 1992

\[
\begin{align*}
\overline{x}(y) \xrightarrow{\overline{y}(y)} 0 & \quad x(u).P \xrightarrow{x(y)} P\{y/u\} & \quad 0 \xrightarrow{x(y)} \overline{x}(y) \\
 P \xrightarrow{\overline{y}(y)} P' & \quad x \neq y & \quad P \xrightarrow{\alpha} P' & \quad y \not\in \alpha \\
 (\nu y)P \xrightarrow{(\nu y)\overline{x}(y)} P' & \quad P \equiv P' & \quad Q \xrightarrow{x(y)} Q' & \quad y \not\in \text{fn}(Q) \\
 P \xrightarrow{\overline{x}(y)} P' & \quad Q \xrightarrow{x(y)} Q' & \quad P \xrightarrow{\overline{x}(y)} P' \quad Q \xrightarrow{x(y)} Q' & \quad P \equiv P' \quad P' \xrightarrow{\alpha} Q' \quad Q' \equiv Q \\
 P \parallel Q \xrightarrow{\tau} P' \parallel Q' & \quad P \equiv P' \quad P' \xrightarrow{\alpha} Q' \quad Q' \equiv Q & \quad P \xrightarrow{\alpha} Q \\
 & & \quad P \parallel Q \xrightarrow{\alpha} P' \parallel Q \\
 P \equiv P' \quad P' \xrightarrow{\alpha} Q' \quad Q' \equiv Q & \\
 & & \quad P \equiv P' \quad P' \xrightarrow{\alpha} Q' \quad Q' \equiv Q
\end{align*}
\]
Honda, Tokoro explained

Idea:

- modify the LTS so that it precisely identifies the interactions that a term can have with its environment;

- rely on a standard weak bisimulation.

Amazing results: asynchronous bisimilarity in ACS style, bisimilarity on top of HT LTS, and barbed congruence coincide.\(^5\)

\(^5\) ahem, modulo some technical details.
Properties of asynchronous bisimilarity in ACS style

- Bisimilarity is a congruence;
  *it is preserved also by input prefix, while it is not in the synchronous case*;

- Bisimilarity is an equivalence relation (transitivity is non-trivial);

- Bisimilarity is *sound* with respect to reduction barbed congruence;

- Bisimilarity is *complete* with respect to barbed congruence.\(^6\)

---

\(^6\)for completeness the calculus must be equipped with a matching operator.
Some proofs about ACS bisimilarity... on asynchronous CCS

Syntax:

\[ P ::= 0 \mid a.P \mid \overline{a} \mid P \parallel P \mid (\nu a)P. \]

Reduction semantics:

\[ a.P \parallel \overline{a} \rightarrow P \]

\[ P \equiv P' \rightarrow Q' \equiv Q \]

\[ P \rightarrow Q \]

where \( \equiv \) is defined as:

\[ P \parallel Q \equiv Q \parallel P \]

\[ (P \parallel Q) \parallel R \equiv P \parallel (Q \parallel R) \]

\[ (\nu a)P \parallel Q \equiv (\nu a)(P \parallel Q) \text{ if } a \notin \text{fn}(Q) \]
Background: LTS and weak bisimilarity for asynchronous CCS

\[ a.P \xrightarrow{a} P \]
\[ \overline{a} \xrightarrow{\overline{a}} 0 \]
\[ P \xrightarrow{\ell} P' \]
\[ P \parallel Q \xrightarrow{\ell} P' \parallel Q \]
\[ (\nu a)P \xrightarrow{\ell} (\nu a)P' \]
\[ P \xrightarrow{a} P' \quad Q \xrightarrow{\overline{a}} Q' \]
\[ P \parallel Q \xrightarrow{\tau} P' \parallel Q' \]

**Definition:** Asynchronous weak bisimilarity, denoted \( \approx \), is the largest symmetric relation such that whenever \( P \approx Q \) and

- \( P \xrightarrow{\ell} P' \), \( \ell \in \{\tau, \overline{a}\} \), there exists \( Q' \) such that \( Q \xrightarrow{\hat{\ell}} Q' \) and \( P' \approx Q' \);
- \( P \xrightarrow{a} P' \), there exists \( Q' \) such that \( Q \parallel \overline{a} \xrightarrow{} Q' \) and \( P' \approx Q' \).

Symmetric rules omitted.
Sketch of the proof of transitivity of $\approx$

Let $\mathcal{R} = \{(P, R) : P \approx Q \approx R\}$. We show that $\mathcal{R} \subseteq \approx$.

- Suppose that $P \mathcal{R} R$ because $P \approx Q \approx R$, and that $P \xrightarrow{a} P'$.

The definition of $\approx$ ensures that there exists $Q'$ such that $Q \parallel \bar{a} \Longrightarrow Q'$ and $P' \approx Q'$.

Since $\approx$ is a congruence and $Q \approx R$, it holds that $Q \parallel \bar{a} \approx R \parallel \bar{a}$.

A simple corollary of the definition of the bisimilarity ensures that there exists $R'$ such that $R \parallel \bar{a} \Longrightarrow R'$ and $Q' \approx R'$.

Then $P' \mathcal{R} R'$ by construction of $\mathcal{R}$.

- The other cases are standard.

Remark the unusual use of the congruence of the bisimilarity.
Sketch of the proof of completeness

We show that \( \simeq \subseteq \approx \).

- Suppose that \( P \simeq Q \) and that \( P \xrightarrow{a} P' \).

  We must conclude that there exists \( Q' \) such that \( Q \parallel a \implies Q' \) and \( P' \simeq Q' \).

  Since \( \simeq \) is a congruence, it holds that \( P \parallel a \simeq Q \parallel a \).

  Since \( P \xrightarrow{a} P' \), it holds that \( P \parallel a \xrightarrow{\tau} P' \).

  Since \( P \parallel a \simeq Q \parallel a \), the definition of \( \simeq \) ensures that there exists \( Q' \) such that \( Q \parallel a \implies Q' \) and \( P' \simeq Q' \), as desired.

- The other cases are analogous to the completeness proof in synchronous CCS.

  The difficulty of the completeness proof is to construct contexts that observe the actions of a process. The case \( P \xrightarrow{a} P' \) is straightforward because “there is nothing to observe”.

31
Some references


