Pi-calculus

proof-techniques, asynchrony, mobility

Francesco Zappa Nardelli

INRIA Rocquencourt, MOSCOVA research team

francesco.zappa_nardelli@inria.fr

MPRI Concurrency course with:

Pierre-Louis Curien (PPS), Roberto Amadio (PPS), Catuscia Palamidessi (INRIA Futurs)

Premises

- Unless otherwise stated, all the equivalences mentioned are *weak equivalences*;
- reduction barbed congruence = reduction-closed barbed congruence = natural contextual equivalence;
- (at the blackboard) one vertical bar = two vertical bars $||$.

Other doubts?

We can start now...

How to prove...

To show that two processes are bisimilar, it is enough to fo 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

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

as a candidate bisimulation. But...

The candidate bisimulation

1. may be larger than at first envisaged;
2. may be infinite;

   example: to show that $x(z).\overline{y}\langle z\rangle \approx (\boldsymbol{\nu} w)(x(z).\overline{w}\langle z\rangle \parallel w(v).\overline{y}\langle v\rangle)$, we must consider:

$$
\begin{array}{ll}
 & \{(x(z).\overline{y}\langle z\rangle, (\boldsymbol{\nu} w)(x(z).\overline{w}\langle z\rangle \parallel w(v).\overline{y}\langle v\rangle))\} \\
\cup & \{(\overline{y}\langle a\rangle, (\boldsymbol{\nu} w)(\overline{w}\langle a\rangle \parallel w(v).\overline{y}\langle v\rangle)) : \ a \text{ arbitrary}\} \\
\cup & \{(\overline{y}\langle a\rangle, (\boldsymbol{\nu} w)(\mathbf{0} \parallel \overline{y}\langle a\rangle)) : \ a \text{ arbitrary}\} \\
\cup & \{(\mathbf{0}, (\boldsymbol{\nu} w)(\mathbf{0} \parallel \mathbf{0}))\}
\end{array}
$$

3. hard to guess;

   which is the smallest bisimulation relating $!!P$ and $!P$?

4. awkward to describe and to work with...

Completing relations

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[1] that contains the two processes.

[1] Hopefully, it is easier to find such relation than to find the candidate bisimulation directly.

Bisimulation up to structural congruence

A symmetric relation R is a *bisimulation up-to* $\equiv$ if whenever $P \mathcal{R} Q$ and $P \xrightarrow{\ell} P'$ then there exists a process Q' such that $Q \overset{\hat{\ell}}{\Longrightarrow} Q'$ and there exist processes P'' and Q'' such that $P' \equiv P'' \mathcal{R}\, Q'' \equiv Q'$.

Exercise: prove that if $\mathcal{R}$ is a bisimulation up to $\equiv$, then $\equiv \mathcal{R} \equiv$ is a bisimulation.

Exercise: prove that for all P, Q it holds $P \mid\mid Q \approx Q \mid\mid P$.

Bisimulation up to non-input context

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

Exercise: Prove that if $\mathcal{R}$ is a bisimulation up to non-input context, then

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

is a bisimulation up to structural congruence.

Exercise: Prove that $!P \;||\; !P \approx\, !P$ (hint: show that the relation $\mathcal{R} = \{(!P \;||\; !P, !P)\}$ is a bisimulation up to non-input context).

A slippery ground...

It would be nice to be able to abstract from internal reduction steps, thus defining *(weak) bisimulation up to (weak) bisimulation*.

But this proof method is not sound: $\tau.a.\mathbf{0}$ and $\mathbf{0}$ are (weakly) bisimilar up to (weak) bisimulation, but they are not bisimilar!

Several solutions: almost-weak bisimulation, *expansion*, etc...

Some references

D. Sangiorgi, R. Milner, *The problem of weak bisimulation up to*, 1992

D. Sangiorgi, *On the bisimulation proof method*, 1994

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:

$$\overline{a}.P \;\|\; a.Q \;\rightarrow\; P \;\|\; Q \;.$$

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

$$\overline{a}.P \;\|\; a.Q \;\ldots \rightarrow \ldots\; \overline{a} \;\|\; P \;\|\; a.Q \;\ldots \rightarrow \ldots\; P' \;\|\; Q \;.$$

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 $\mathbf{0}$.

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

Formally:

$$\overline{x}\langle y\rangle \;\Big|\Big|\; x(z).P \;\rightharpoonup\; P\{^{y}/_{z}\} \;.$$

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

Asynchronous pi-calculus

Syntax:

$$P \;::=\; \mathbf{0} \;\;\mid\;\; x(y).P \;\;\mid\;\; \overline{x}\langle y\rangle \;\;\mid\;\; P \,\|\, P \;\;\mid\;\; (\boldsymbol{\nu} x)P \;\;\mid\;\; !P$$

The definitions of free and bound names, of structural congruence $\equiv$, and of the reduction relation $\rightarrowtriangle$ are inherited from pi-calculus.

Examples

Sequentialization of output actions is still possible:

$$(\boldsymbol{\nu} y, z)(\overline{x}\langle y\rangle \;\|\; \overline{y}\langle z\rangle \;\|\; \overline{z}\langle a\rangle \;\|\; R) \;.$$

Synchronous communication can be implemented by waiting for an acknoledgement:

$$\begin{aligned} [\![\, \overline{x}\langle y\rangle.P \,]\!] &= (\boldsymbol{\nu} u)(\overline{x}\langle y, u\rangle \;\|\; u().P) \\ [\![\, x(v).Q \,]\!] &= x(v, w).(\overline{w}\langle\rangle \;\|\; Q) \qquad \text{for } w \notin Q \end{aligned}$$

Exercise: implement synchronous communication without relying on polyadic primitives.

Background: a recipe for a *"natural"* contextual equivalence

Say that P and Q are equivalent (in symbols: $P \simeq Q$) if:

Preservation under contexts For all contexts $C[-]$, we have $C[P] \simeq C[Q]$;

Preservation of observations If $P \downarrow x$ then $Q \Downarrow x$, where $P \downarrow x$ is defined as

$$P \equiv (\boldsymbol{\nu}\tilde{n})(\overline{x}\langle y\rangle.P' \mid\mid P'') \text{ or } P \equiv (\boldsymbol{\nu}\tilde{n})(x(u).P' \mid\mid P'') \text{ for } x \notin \tilde{n} \;;$$

Preservation of reductions If $P \simeq Q$ and $P \rightarrow P'$ then there is a Q' such that $Q \rightarrow^* Q'$ and $P' \simeq Q'$.

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

$$
\begin{aligned}
P &= \, !x(z).\overline{x}\langle z\rangle \\
Q &= \mathbf{0}
\end{aligned}
$$

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.

A proof method

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

$$!x(z).\overline{x}\langle z\rangle \;\; \not\approx_s \mathbf{0} \ .$$

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 $P \xrightarrow{\overline{x}\langle y \rangle} P'$ then P can interact with the context $- \parallel x(u).\mathrm{beep}$, 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.

Amadio, Castellani, Sangiorgi - 1996

Idea: relax the matching condition for input actions.

Let *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 $Q \stackrel{\hat{\ell}}{\Longrightarrow} Q'$ and $P' \approx_a Q'$;

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

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

Honda, Tokoro - 1992

$$\overline{x}\langle y\rangle \xrightarrow{\overline{x}\langle y\rangle} \mathbf{0} \qquad x(u).P \xrightarrow{x(y)} P\{{}^{y}\!/_{u}\} \qquad \mathbf{0} \xrightarrow{x(y)} \overline{x}\langle y\rangle$$

$$\frac{P \xrightarrow{\overline{x}\langle y\rangle} P' \quad x \neq y}{(\boldsymbol{\nu} y)P \xrightarrow{(\boldsymbol{\nu} y)\overline{x}\langle y\rangle} P'} \qquad \frac{P \xrightarrow{\alpha} P' \quad y \notin \alpha}{(\boldsymbol{\nu} y)P \xrightarrow{\alpha} (\boldsymbol{\nu} y)P'}$$

$$\frac{P \xrightarrow{\overline{x}\langle y\rangle} P' \quad Q \xrightarrow{x(y)} Q'}{P \parallel Q \xrightarrow{\tau} P' \parallel Q'} \qquad \frac{P \xrightarrow{\overline{x}\langle (y)\rangle} P' \quad Q \xrightarrow{x(y)} Q' \quad y \notin \mathrm{fn}(Q)}{P \parallel Q \xrightarrow{\tau} (\boldsymbol{\nu} y)(P' \parallel Q')}$$

$$\frac{P \xrightarrow{\alpha} P' \quad \mathrm{bn}(\alpha) \cap \mathrm{fn}(Q) = \emptyset}{P \parallel Q \xrightarrow{\alpha} P' \parallel Q} \qquad \frac{P \equiv P' \quad P' \xrightarrow{\alpha} Q' \quad Q' \equiv Q}{P \xrightarrow{\alpha} Q}$$

Honda, Tokoro explained

Ideas:

- 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: asynchrounous bisimilarity in ACS style, bisimilarity on top of HT LTS, and barbed congruence coincide.[2]

[2] 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.[3]

[3] for this the calculus must be equipped with a matching operator.

Some proofs about ACS bisimilarity... on asynchronous CCS

Syntax:

$$P \quad ::= \quad \mathbf{0} \quad | \quad a.P \quad | \quad \overline{a} \quad | \quad P \parallel P \quad | \quad (\boldsymbol{\nu} a)P \ .$$

Reduction semantics:

$$a.P \parallel \overline{a} \ \rightarrow \ P \qquad\qquad \frac{P \equiv P' \rightarrow Q' \equiv Q}{P \rightarrow Q}$$

where $\equiv$ is defined as:

$$P \parallel Q \equiv Q \parallel P \qquad\qquad (P \parallel Q) \parallel R \equiv P \parallel (Q \parallel R)$$

$$(\boldsymbol{\nu} a)P \parallel Q \equiv (\boldsymbol{\nu} a)(P \parallel Q) \text{ if } a \notin \mathrm{fn}(Q)$$

Background: LTS and weak bisimilarity for asynchronous CCS

$$a.P \xrightarrow{a} P \qquad \overline{a} \xrightarrow{\overline{a}} \mathbf{0} \qquad \frac{P \xrightarrow{a} P' \quad Q \xrightarrow{\overline{a}} Q'}{P \parallel Q \xrightarrow{\tau} P' \parallel Q'}$$

$$\frac{P \xrightarrow{\ell} P'}{P \parallel Q \xrightarrow{\ell} P' \parallel Q} \qquad \frac{P \xrightarrow{\ell} P' \quad a \notin \mathrm{fn}(\ell)}{(\boldsymbol{\nu} a)P \xrightarrow{\ell} (\boldsymbol{\nu} a)P'}$$

symmetric rules omitted.

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 $Q \stackrel{\hat{\ell}}{\Longrightarrow} Q'$ and $P' \approx Q'$;
- $P \xrightarrow{a} P'$, there exists Q' such that $Q \parallel \overline{a} \Longrightarrow Q'$ and $P' \approx Q'$.

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 $Q \ || \ \overline{a} \Longrightarrow Q'$ and $P' \approx Q'$.

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

A simple corollary of the defintion of the bisimilarity ensures that there exists R' such that $R \ || \ \overline{a} \Longrightarrow R'$ 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 $Q \mid\mid \overline{a} \Longrightarrow Q'$ and $P' \simeq Q'$.

Since $\simeq$ is a congruence, it holds that $P \mid\mid \overline{a} \simeq Q \mid\mid \overline{a}$.

Since $P \xrightarrow{a} P'$, it holds that $P \mid\mid \overline{a} \xrightarrow{\tau} P'$.

Since $P \mid\mid \overline{a} \simeq Q \mid\mid \overline{a}$, the definition of $\simeq$ ensures that there exists Q' such that $Q \mid\mid \overline{a} \Longrightarrow Q'$ 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".

Some references

Kohei Honda, Mario Tokoro: *An Object Calculus for Asynchronous Communication*. ECOOP 1991.

Kohei Honda, Mario Tokoro, *On asynchronous communication semantics*. Object-Based Concurrent Computing 1991.

Gerard Boudol, *Asynchrony and the pi-calculus*. INRIA Research Report, 1992.

Roberto Amadio, Ilaria Castellani, Davide Sangiorgi, *On bisimulations for the asynchronous pi-calculus*. Theor. Comput. Sci. 195(2), 1998.

Distribution, action at distance, and mobility

The parallel composition operator of CCS and pi-calculus does not specify whether the concurrent threads are running on the same machine, or on different machines connected by a network.

Some phenomena typical of distributed systems require a finer model, that explicitly keeps track of the spatial distribution of the processes.

We will briefly sketch two models that have been proposed: *DPI* (Hennessy and Riely, 1998) and *Mobile Ambients* (Cardelli and Gordon, 1998).

The aim of this section is to get a glimpse of more complex process languages, and to rediscover the idea of "transitions in an LTS characterise the interactions a term can have with a context" in this setting.

DPI, design choices

- add explicit locations to pi-calculus processes: $\ell[\![P]\!]$;
- locations are identified by their name: $\ell[\![P]\!] \parallel \ell[\![Q]\!] \equiv \ell[\![P \parallel Q]\!]$;
- communication is local to a location:

$$\ell[\![\overline{x}\langle y \rangle.P]\!] \;\Big|\Big|\; \ell[\![x(u).Q]\!] \rightarrow \ell[\![P]\!] \;\Big|\Big|\; \ell[\![Q\{^{y}/_{u}\}]\!] \;;$$

- add explicit migration: $\ell[\![\text{goto } k.P]\!] \rightarrow k[\![P]\!]$.

We also include the restriction and match operators, subject to the usual pi-calculus semantics.

Behavioural equivalence for DPI

Again, we apply the standard recipe:

- define the suitable contexts:

$$C[-] \ ::= \ - \ \mid \ C[-] \,\|\, \ell[\![P]\!] \ \mid \ (\boldsymbol{\nu} n)C[-] \ .$$

- define the observation:

$$M \downarrow x@\ell \text{ iff } P \equiv (\boldsymbol{\nu}\tilde{n})(\ell[\![x(u).P']\!] \,\|\, P'') \text{ for } x, \ell \notin \tilde{n} \ .$$

Can we characterise this equivalence with a labelled bisimulation?

Labelled bisimulation for DPI

$$\frac{P \rightarrowtriangle P'}{P \xrightarrow{\tau} P'} \qquad \frac{P \equiv (\boldsymbol{\nu}\tilde{n})(\ell[\![\, x(u).P' \,]\!] \parallel P'') \qquad x, \ell \notin \tilde{n}}{P \xrightarrow{x(y)@\ell} (\boldsymbol{\nu}\tilde{n})(\ell[\![\, P'\{^{y}/_{u}\} \,]\!] \parallel P'')}$$

$$\frac{P \equiv (\boldsymbol{\nu}\tilde{n})(\ell[\![\, \overline{x}\langle y\rangle.P' \,]\!] \parallel P'') \qquad x, y, \ell \notin \tilde{n}}{P \xrightarrow{\overline{x}\langle y\rangle@\ell} (\boldsymbol{\nu}\tilde{n})(\ell[\![\, P' \,]\!] \parallel P'')}$$

$$\frac{P \equiv (\boldsymbol{\nu}\tilde{n})(\ell[\![\, \overline{x}\langle y\rangle.P' \,]\!] \parallel P'') \qquad x, \ell \notin \tilde{n} \qquad y \in \tilde{n}}{P \xrightarrow{\overline{x}\langle (y)\rangle@\ell} (\boldsymbol{\nu}\tilde{n} \setminus y)(\ell[\![\, P' \,]\!] \parallel P'')}$$

Labelled bisimulation for DPI, ctd.

The standard bisimulation on top of the LTS below coincides with reduction barbed congruence.

Remark: the LTS is written in an *unconventional* style, which precisely characterises the interactions a term can have with a context.

Questions:

1- every label should correspond to a (minimal) interacting context: can you spell out these contexts?

2- why there are no explicit labels for the "goto" action?

Mobile Ambients, design choices

Objective: build a process language on top of the concepts of barriers (administrative domains, firewalls, ...) and of barrier crossing.

A graphical representation of the syntax and of the reduction semantics of Mobile Ambients can be found here:

`http://research.microsoft.com/Users/luca/Slides/`
`2000-11-10%20Wide%20Area%20Computation%20(Valladolid).pdf`

Mobile Ambients syntax (in ISO 10646)

Processes:			*Capabilities:*		
P, Q, R	$::=$	$\mathbf{0}$	C	$::=$	$\mathtt{in\_}n$
	$\mid$	$P_1 \parallel P_2$		$\mid$	$\mathtt{out\_}n$
	$\mid$	$(\boldsymbol{\nu} n)P$		$\mid$	$\mathtt{open\_}n$
	$\mid$	$n[P]$			
	$\mid$	$C.P$			
	$\mid$	$!P$			

Mobile Ambients: interaction

- Locations migrate under the control of the processes located at their inside:

$$n[\mathtt{in\_}m.P \;||\; Q] \;||\; m[R] \rightarrow m[\,n[P \;||\; Q] \;||\; R\,]$$
$$m[\,n[\mathtt{out\_}m.P \;||\; Q] \;||\; R\,] \rightarrow n[P \;||\; Q] \;||\; m[R]$$

- a location may be opened:

$$\mathtt{open\_}n.P \;\Big|\Big|\; n[\,Q\,] \rightarrow P \;\Big|\Big|\; Q$$

Hint about an LTS for Mobile Ambients

Consider the term $M \equiv (\boldsymbol{\nu}\tilde{m})(k[\texttt{in\_}n.P \parallel Q] \parallel R)$ where $k \notin \tilde{m}$. It can interact with the context $n[T] \parallel -$, where T is an arbitrary process, yielding $O \equiv (\boldsymbol{\nu}\tilde{m})(n[T \parallel k[P \parallel Q]] \parallel R)$. This interaction can be captured with a transition $M \xrightarrow{k.\texttt{enter\_}n} O$.

Remark that, contrarily to what happens in CCS and pi-calculus, a bit of the interacting context is still visible in the outcome!

Along these lines (asynchrony is needed too!) it is possible to characterise reduction barbed congruence using a labelled bisimilarity.

References

James Riely, Matthew Hennessy: *Distributed Pprocesses and location failures*. Theoretical Computer Science, 2001. An extended abstract appeard in ICALP 97.

Luca Cardelli, Andrew Gordon: *Mobile Ambients*. Theoretical Computer Science, 2000. An extended abstract appeared in FOSSACS 1998.

Massimo Merro, myself: *A behavioral theory for Mobile Ambients*. Journal of ACM, 2005.