Concurrency theory

proof-techniques for syncronous 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)

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

ℓ	kind	$\mathrm{fn}(\ell)$	$\mathrm{bn}(\ell)$	$\mathsf{n}(\ell)$
$\overline{x}\langle y\rangle$	free output	$\{x, y\}$	$\emptyset$	$\{x, y\}$
$(\boldsymbol{\nu} y)\overline{x}\langle y\rangle$	bound output	$\{x\}$	$\{y\}$	$\{x, y\}$
$x(y)$	input	$\{x, y\}$	$\emptyset$	$\{x, y\}$
τ	internal	$\emptyset$	$\emptyset$	$\emptyset$

Back on pi-calculus LTS

$$\overline{x}\langle v\rangle.P \xrightarrow{\overline{x}\langle v\rangle} P \qquad x(y).P \xrightarrow{x(v)} \{{}^{v}/_{y}\}P \qquad \frac{P \xrightarrow{\overline{x}\langle v\rangle} P' \qquad Q \xrightarrow{x(v)} Q'}{P \mid\mid Q \xrightarrow{\tau} P' \mid\mid Q'}$$

$$\frac{P \xrightarrow{\ell} P' \qquad \mathrm{bn}(\ell) \cap \mathrm{fn}(Q) = \emptyset}{P \mid\mid Q \xrightarrow{\ell} P' \mid\mid Q} \qquad \frac{P \xrightarrow{\ell} P' \qquad v \notin \mathsf{n}(\ell)}{(\boldsymbol{\nu} v)P \xrightarrow{\ell} (\boldsymbol{\nu} v)P'} \qquad \frac{P \mid\mid !P \xrightarrow{\ell} P'}{!P \xrightarrow{\ell} P'}$$

$$\frac{P \xrightarrow{\overline{x}\langle v\rangle} P' \qquad x \neq v}{(\boldsymbol{\nu} v)P \xrightarrow{(\boldsymbol{\nu} v)\overline{x}\langle v\rangle} P'} \qquad \frac{P \xrightarrow{(\boldsymbol{\nu} v)\overline{x}\langle v\rangle} P' \qquad Q \xrightarrow{x(v)} Q' \qquad v \notin \mathrm{fn}(Q)}{P \mid\mid Q \xrightarrow{\tau} (\boldsymbol{\nu} v)(P' \mid\mid Q')}$$

Subtleties of pi-calculus LTS

Exercise: derive a τ transition corresponding to this reduction:

$$(\boldsymbol{\nu} x)\overline{a}\langle x\rangle.P \;\|\; a(y).Q \;\rightarrow\; (\boldsymbol{\nu} x)(P \;\|\; Q\{{}^{x}/_{y}\})$$

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 on 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 $+$):

$$
\begin{aligned}
P &= \overline{x}\langle v\rangle \mathbin{\big|\big|} y(z) \\
Q &= \overline{x}\langle v\rangle.y(z) + y(z).\overline{x}\langle v\rangle
\end{aligned}
$$

1. Prove that $P \approx Q$[1].

2. Does $P \simeq Q$?[2]

[1] Does this hold if we replace $+$ by $-_1 \oplus -_2 = (\boldsymbol{\nu} w)(\overline{w}\langle\rangle \mid\mid w().-_1 \mid\mid w().-_2)$ in Q?
[2] Hint: 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 $C[-] = x(y).-$.

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 σ.

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

$$P = \overline{a}\langle x\rangle \;||\; E_{xy} \qquad\qquad Q = \overline{a}\langle y\rangle \;||\; E_{xy}$$

where

$$E_{xy} = !x(z).\overline{y}\langle z\rangle \;||\; !y(z).\overline{x}\langle z\rangle$$

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.

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

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

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...

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 $P \ \mathcal{R} \ Q$ and $P \xrightarrow{\ell} P'$ then there exists a process Q' such that $Q \overset{\hat{\ell}}{\Longrightarrow} Q'$ and there exist 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).

Alternative LTS rules for replication

It is often convenient to replace the rule:

$$\frac{P \parallel !P \xrightarrow{\ell} P'}{!P \xrightarrow{\ell} P'}$$

with the three rules:

$$\frac{P \xrightarrow{\ell} P'}{!P \xrightarrow{\ell} P' \parallel !P} \qquad \frac{P \xrightarrow{\overline{x}\langle y\rangle} P_1 \quad P \xrightarrow{x(y)} P_2}{!P \xrightarrow{\tau} (P_1 \parallel P_2) \parallel !P} \qquad \frac{P \xrightarrow{(\boldsymbol{\nu} y)\overline{x}\langle y\rangle} P_1 \quad P \xrightarrow{x(y)} P_2}{!P \xrightarrow{\tau} (\boldsymbol{\nu} y)(P_1 \parallel P_2) \parallel !P}$$

Theorems about replication

The equivalence $!P \mid\mid !P \approx !P$ shows that duplication of a replicable resource has no behavioural effect. Consider now

$$(\boldsymbol{\nu} x)(P \;\Big|\Big|\; !x(y).Q)$$

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

$$(\boldsymbol{\nu} x)\left(P_1 \;\Big|\Big|\; P_2 \;\Big|\Big|\; !x(y).Q\right) \approx (\boldsymbol{\nu} x)\left(P_1 \;\Big|\Big|\; !x(y).Q\right) \;\Big|\Big|\; (\boldsymbol{\nu} x)\left(P_2 \;\Big|\Big|\; !x(y).Q\right)$$

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:

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

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

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

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:

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

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:

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

Synchronous communication can be implemented by waiting for an acknoledgement:

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

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

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

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.[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:

$$P \;::=\; \mathbf{0} \;\mid\; a.P \;\mid\; \overline{a} \;\mid\; P \,\|\, P \;\mid\; (\boldsymbol{\nu} a)P\ .$$

Reduction semantics:

$$a.P \,\|\, \overline{a} \;\rightarrowtriangle\; P \qquad\qquad \frac{P \equiv P' \rightarrowtriangle Q' \equiv Q}{P \rightarrowtriangle Q}$$

where $\equiv$ is defined as:

$$P \,\|\, Q \equiv Q \,\|\, P \qquad\qquad (P \,\|\, Q) \,\|\, R \equiv P \,\|\, (Q \,\|\, R)$$

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

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

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

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