Formalisation en Coq du calcul de processus HOcore

Simon Boulier [1] Alan Schmitt [2]

JFLA 2012

1. ENS Cachan - Antenne de Bretagne
2. INRIA

Nous montrons que $\mathcal{R}$ est une bisimulation tardive d'ordre supérieure. La relation est symétrique, il suffit donc de montrer qu'il s'agit d'une simulation. Nous procédons par analyse de cas sur les transitions de $P\{R/X\}$.

Si la transition est engendrée uniquement par P, nous avons $P\{R/X\} \xrightarrow{\alpha} A\{R/X\}$ avec $P \xrightarrow{\alpha} A$. Nous avons donc $P\{\overline{m}.\overline{n}.\mathbf{0}/X\} \xrightarrow{\alpha} A\{\overline{m}.\overline{n}.\mathbf{0}/X\}$. La transition $\xrightarrow{\alpha}$ provient de P, qui ne contient pas d'occurrences de m, n, donc m, n n'apparaissent pas dans α. Nous distinguons les trois cas possibles pour A.

- Supposons que A est un processus P'. Comme $P\{\overline{m}.\overline{n}.\mathbf{0}/X\} \dot{\sim}_l Q\{\overline{m}.\overline{n}.\mathbf{0}/X\}$, il existe Q' tel que $Q\{\overline{m}.\overline{n}.\mathbf{0}/X\} \xrightarrow{\alpha} Q'$ et $P'\{\overline{m}.\overline{n}.\mathbf{0}/X\} \dot{\sim}_l Q'$. Comme $m, n \notin \alpha$, la transition $Q\{\overline{m}.\overline{n}.\mathbf{0}/X\} \xrightarrow{\alpha} Q'$ provient de Q : il existe Q" tel que $Q \xrightarrow{\alpha} Q$" et $Q' = Q"\{\overline{m}.\overline{n}.\mathbf{0}/X\}$. Nous avons donc $Q\{R/X\} \xrightarrow{\alpha} Q"\{R/X\}$ avec $P'\{R/X\}\ \mathcal{R}\ Q"\{R/X\}$, comme voulu.
- Supposons que A est une fonction F. Il existe F' telle que $Q\{\overline{m}.\overline{n}.\mathbf{0}/X\} \xrightarrow{\alpha} F'$ et $(F\{\overline{m}.\overline{n}.\mathbf{0}/X\}) \circ T \dot{\sim}_l F' \circ T$ pour tout processus T. Comme $m, n \notin \alpha$, la transition $Q\{\overline{m}.\overline{n}.\mathbf{0}/X\} \xrightarrow{\alpha} F'$ provient de Q : il existe $F"$ tel que $Q \xrightarrow{\alpha} F"$ et $F' = F"\{\overline{m}.\overline{n}.\mathbf{0}/X\}$. Nous avons donc $Q\{R/X\} \xrightarrow{\alpha} F"\{R/X\}$, et pour tout T clos, nous avons $(F\{R/X\}) \circ T = (F \circ T)\{R/X\}$ et $(F"\{R/X\}) \circ T = (F" \circ T)\{R/X\}$. Nous avons également $(F \circ T)\{\overline{m}.\overline{n}.\mathbf{0}/X\} = (F\{\overline{m}.\overline{n}.\mathbf{0}/X\}) \circ T \dot{\sim}_l (F"\{\overline{m}.\overline{n}.\mathbf{0}/X\}) \circ T = (F" \circ T)\{\overline{m}.\overline{n}.\mathbf{0}/X\}$. Finalement nous avons donc $(F \circ T)\{R/X\}\ \mathcal{R}\ (F" \circ T)\{R/X\}$ comme voulu.
- Supposons que $A = \langle S\rangle T$. Il existe S' et T' tels que $Q\{\overline{m}.\overline{n}.\mathbf{0}/X\} \xrightarrow{\alpha} \langle S'\rangle T'$, $S\{\overline{m}.\overline{n}.\mathbf{0}/X\} \dot{\sim}_l S'$ et $T\{\overline{m}.\overline{n}.\mathbf{0}/X\} \dot{\sim}_l T'$. Comme nous avons $m, n \notin \alpha$, la transition $Q\{\overline{m}.\overline{n}.\mathbf{0}/X\} \xrightarrow{\alpha} \langle S'\rangle T'$ provient de Q : il existe $S", T"$ tels que $Q \xrightarrow{\alpha} \langle S"\rangle T"$, $S' = S"\{\overline{m}.\overline{n}.\mathbf{0}/X\}$ et $T' = T"\{\overline{m}.\overline{n}.\mathbf{0}/X\}$. Par conséquent, nous avons $Q\{R/X\} \xrightarrow{\alpha} \langle S"\{R/X\}\rangle T"\{R/X\}$ avec $S\{R/X\}\ \mathcal{R}\ S"\{R/X\}$ et $T\{R/X\}\ \mathcal{R}\ T"\{R/X\}$, comme voulu.

On the Expressiveness and Decidability of Higher-Order Process Calculi* †

Ivan Lanese
Focus Team, Università di Bologna/INRIA, Italy
lanese@cs.unibo.it

Jorge A. Pérez
CITI - Dept. of Computer Science, New University of Lisbon, Portugal
jorge.perez@di.fct.unl.pt

Davide Sangiorgi
Focus Team, Università di Bologna/INRIA, Italy
davide.sangiorgi@cs.unibo.it

Alan Schmitt
INRIA, France
alan.schmitt@inria.fr

June 1, 2010

Abstract

In *higher-order process calculi*, the values exchanged in communications may contain processes. A core calculus of higher-order concurrency is studied; it has

Syntaxe d'un processus

HOcore :

$$
\begin{aligned}
P, P' ::= & \\
& \mid \ \overline{a}\langle P \rangle \\
& \mid \ a(x).P \\
& \mid \ P \parallel P' \\
& \mid \ x \\
& \mid \ \mathbf{0}.
\end{aligned}
$$

Syntaxe d'un processus

HOcore :

$$
\begin{aligned}
P, P' ::= & \\
& \mid \overline{a}\langle P \rangle \\
& \mid a(x).P \\
& \mid P \parallel P' \\
& \mid x \\
& \mid \mathbf{0}.
\end{aligned}
$$

```
Definition chan := nat.

Definition lvar := nat.

Inductive process : Set :=
| Send : chan → process → process
| Receive : chan → lvar → process
    → process
| Par : process → process →
    process
| Lvar : lvar → process
| Nil : process.
```

Communication

Communication :

$$\overline{a}\langle Q\rangle \parallel a(x).P \longrightarrow [Q \,/\, x]P$$

Réduction :

$$\ldots \parallel \overline{a}\langle Q\rangle \parallel \ldots \parallel a(x).P \parallel \ldots \longrightarrow \ldots \parallel [Q \,/\, x]P \parallel \ldots$$

Congruence Structurelle

Congruence Structurelle :

- $P \parallel (Q \parallel R) \equiv (P \parallel Q) \parallel R$
- $P \parallel Q \equiv Q \parallel P$
- $P \parallel \mathbf{0} \equiv P$

Exemple :

$$\overline{a}\langle x \parallel y \rangle \equiv \overline{a}\langle y \parallel x \rangle$$

Réduction :

$$\frac{P \equiv P' \qquad P' \longrightarrow Q' \qquad Q \equiv Q'}{P \longrightarrow Q} \qquad \frac{P \longrightarrow P'}{(P \parallel Q) \longrightarrow (P' \parallel Q)}$$

Résumé

$$
\begin{array}{rl}
P, P' ::= & \\
 & \mid \ \overline{a}\langle P \rangle \\
 & \mid \ a(x).P \\
 & \mid \ P \parallel P' \\
 & \mid \ x \\
 & \mid \ \mathbf{0}.
\end{array}
$$

$$\overline{a}\langle Q \rangle \parallel a(x).P \longrightarrow [Q \,/\, x]P$$

$$\frac{P \longrightarrow P'}{(P \parallel Q) \longrightarrow (P' \parallel Q)}$$

$$\frac{P \equiv P' \qquad P' \longrightarrow Q' \qquad Q \equiv Q'}{P \longrightarrow Q}$$

- $P \parallel (Q \parallel R) \equiv (P \parallel Q) \parallel R$
- $P \parallel Q \equiv Q \parallel P$
- $P \parallel \mathbf{0} \equiv P$

HOcore est Turing complet !

Une nouvelle sémantique

Les règles du LTS :

Une nouvelle sémantique

Les règles du LTS :

$$\textsc{Out}\ \ \overline{a}\langle P\rangle \xrightarrow{\overline{a}\langle P\rangle} \mathbf{0}$$

Une nouvelle sémantique

Les règles du LTS :

$$\textsc{Out}\ \ \overline{a}\langle P\rangle \xrightarrow{\overline{a}\langle P\rangle} \mathbf{0}$$

$$\textsc{Inp}\ \ a(x).P \xrightarrow{a} (x).P$$

Fonction :

$F ::=$

$\quad \mid \ (x).P$

Une nouvelle sémantique

Les règles du LTS :

$$\textsc{Out}\ \ \overline{a}\langle P\rangle \xrightarrow{\overline{a}\langle P\rangle} \mathbf{0}$$

$$\textsc{Inp}\ \ a(x).P \xrightarrow{a} (x).P$$

$$\textsc{Act}\ \ \text{si } P \xrightarrow{\alpha} A \text{ alors } P \parallel Q \xrightarrow{\alpha} A \parallel Q$$

Fonction :

$F ::=$

$\mid\ (x).P$

$\mid\ F \parallel P$

$\mid\ P \parallel F.$

$A ::= P \mid F.$

Une nouvelle sémantique

Les règles du LTS :

$$\textsc{Out}\ \ \overline{a}\langle P\rangle \xrightarrow{\overline{a}\langle P\rangle} \mathbf{0}$$

$$\textsc{Inp}\ \ a(x).P \xrightarrow{a} (x).P$$

$$\textsc{Act}\ \ \text{si } P \xrightarrow{\alpha} A \text{ alors } P \parallel Q \xrightarrow{\alpha} A \parallel Q$$

$$\textsc{Tau}\ \ \text{si } P \xrightarrow{\overline{a}\langle P''\rangle} P' \text{ et } Q \xrightarrow{a} F \text{ alors } P \parallel Q \xrightarrow{\tau} P' \parallel F \bullet P''$$

Fonction :

$$F ::= \mid (x).P \mid F \parallel P \mid P \parallel F.$$

Instanciation :

$$(x).P \bullet P'' = [P'' \,/\, x]P$$
$$(F \parallel P) \bullet P'' = (F \bullet P'') \parallel P$$
$$(P \parallel F) \bullet P'' = P \parallel (F \bullet P'')$$

$$A ::= P \mid F.$$

Le problème de la capture

$$a(y).x \sim_\alpha a(z).x$$

$$[y \,/\, x](a(y).x) = a(y).y \not\sim_\alpha [y \,/\, x](a(z).x) = a(z).y$$

Une Solution : la Représentation Canonique des Lieurs

- pour cela, on distingue variables libres et liées

libres : X, Y, Z (`gvar` : globales)
liées : x, y, z (`lvar` : locales)

$$
\begin{aligned}
P, P' ::= \\
&\mid \overline{a}\langle P \rangle \\
&\mid a(x).P \\
&\mid P \parallel P' \\
&\mid x \\
&\mid X \\
&\mid \mathbf{0}.
\end{aligned}
$$

Une Solution : la Représentation Canonique des Lieurs

- pour cela, on distingue variables libres et liées

libres : X, Y, Z (`gvar` : globales)
liées : x, y, z (`lvar` : locales)

$$
\begin{aligned}
P, P' ::= \\
&\mid \overline{a}\langle P \rangle \\
&\mid a(x).P \\
&\mid P \parallel P' \\
&\mid x \\
&\mid X \\
&\mid \mathbf{0}.
\end{aligned}
$$

- on veut choisir un représentant

Exemple

$$a(x).(x \parallel x) \text{ ou } a(y).(y \parallel y) \,?$$

Exemple

$$a(x).(x \parallel x) \text{ ou } a(y).(y \parallel y) \,?$$

1. on considère $X \parallel X$
2. on calcule $f_x(X \parallel X) \rightarrow x_0$
3. le représentant canonique est :
   $a(x_0).([x_0 \,/\, X](X \parallel X)) = a(x_0).(x_0 \parallel x_0)$

On note $\mathbf{abs}_f\ a\, X\, P = a(x_0).([x_0\ /\ X]P)$ où $x_0 = f_x(P)$.

Conclusion : $a(x).(x \parallel x) \rightsquigarrow \mathbf{abs}_f\ a\, X\, (X \parallel X)$.

On note $\mathbf{abs}_f\ a\, X\, P = a(x_0).([x_0\ /\ X]P)$ où $x_0 = f_x(P)$.

$f : \texttt{gvar} \to \texttt{process} \to \texttt{lvar}$ est une fonction de hauteur, elle vérifie :

Équivariance $f_X(P) = f_{\pi(X)}(\pi(P))$ pour toute permutation π
$\Rightarrow \mathbf{abs}_f\ a\, X\, (X \parallel X) = \mathbf{abs}_f\ a\, Y\, (Y \parallel Y)$

Fraîcheur ex : $\mathbf{abs}_f\ b\, X\, (a(x).X) \begin{cases} \neq b(x).(a(x).x) \\ = b(y).(a(x).y) \end{cases}$

Substitution si $X \sharp Q, Q'$ alors $f_X(\ldots Q \ldots) = f_X(\ldots Q' \ldots)$

Congruence si $P \equiv Q$ alors $f_X(P) = f_X(Q)$

```
Fixpoint GoodF X P :=
  match P with
    | Gvar Y ⇒if beq_nat X Y then 1 else 0
    | Lvar _ ⇒0
    | Nil ⇒0
    | Par P1 P2 ⇒max (GoodF X P1) (GoodF X P2)
    | Send a P ⇒GoodF X P
    | Receive a x P ⇒let m' := GoodF X P in
          if beq_nat m' 0 then 0
          else if lt_dec x m' then m'
               else S x
  end.


Theorem GoodF_is_good : good GoodF.
```

- $\forall X$, $\mathrm{wf}_f(X)$
- si $\mathrm{wf}_f(P)$ alors $\mathrm{wf}_f(\overline{a}\langle P\rangle)$
- si $\mathrm{wf}_f(P)$ et $\mathrm{wf}_f(Q)$ alors $\mathrm{wf}_f(P \parallel Q)$
- $\mathrm{wf}_f(\mathbf{0})$
- si $\mathrm{wf}_f(P)$, alors $\mathrm{wf}_f(\mathbf{abs}_f\ a\, X\, P)$

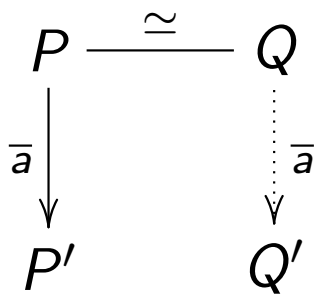
```
Inductive wf (f:height_fun) : process → Prop :=
  | WfGvar : ∀X, wf f (Gvar X)
  | WfSend : ∀a p, (wf f p) → wf f (Send a p)
  | WfPar : ∀p q, (wf f p) → (wf f q) → wf f (Par p q)
  | WfNil : wf f Nil
  | WfReceive : ∀a X p, (wf f p) → wf f (abs f a X p).
```

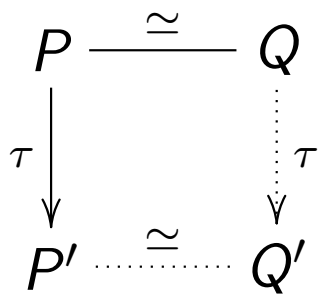
```
Lemma wf_ind : ∀(P : process → Prop),
  (∀ a p, wf f p → P p → P (Send a p)) →

  (∀ a X p, wf f p →
      P p →
          P (abs f a X p)) →

  (∀ X, P (Gvar X)) →
  (∀ p q, wf f p → P p → wf f q → P q → P (Par p q)) →
  (P Nil) →
    ∀p, wf f p → P p.
```

```
Lemma wf_ind_gen : ∀(P : process → Prop),
  (∀ a p, wf f p → P p → P (Send a p)) →

  (∀ a X p, wf f p →
      (∀ Y, Y#p → P ([Y/X]p)) →
          P (abs f a X p)) →

  (∀ X, P (Gvar X)) →
  (∀ p q, wf f p → P p → wf f q → P q → P (Par p q)) →
  (P Nil) →
    ∀p, wf f p → P p.
```

Équivalence de processus

La Congruence Barbue

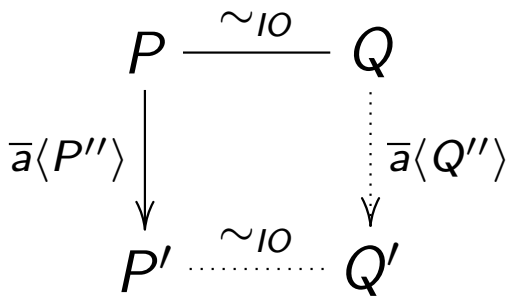


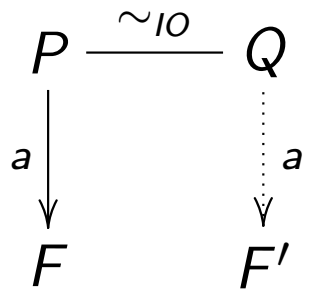


$$P \simeq Q \Rightarrow C[P] \simeq C[Q]$$

pour tout contexte C

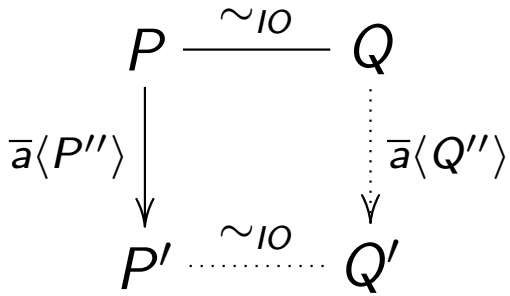
L'IO-bissimulation

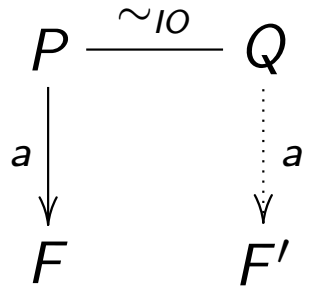




$P'' \sim_{IO} Q''$ $\quad F \bullet M \sim_{IO} F' \bullet M \quad \forall M$

L'IO-bissimulation



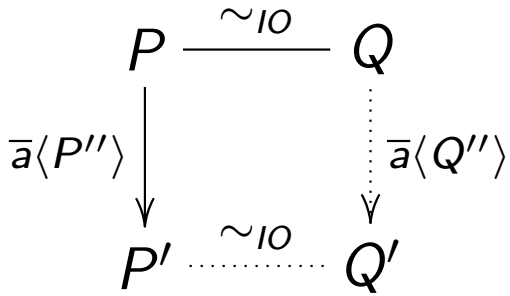


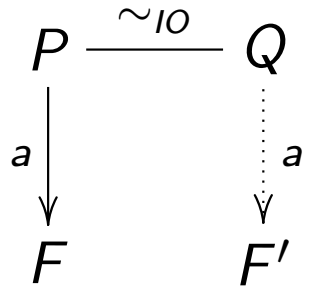
$P'' \sim_{IO} Q''$

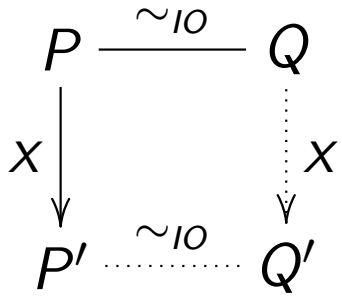
$F \bullet M \sim_{IO} F' \bullet M \quad \forall M$

$F \bullet Y \sim_{IO} F' \bullet Y \quad$ avec $Y \sharp P, Q$

L'IO-bissimulation







$P'' \sim_{IO} Q''$ $\quad F \bullet M \sim_{IO} F' \bullet M \quad \forall M$

$F \bullet Y \sim_{IO} F' \bullet Y$ $\quad$ avec $Y \sharp P, Q$

REM $X \xrightarrow{X} \mathbf{0}$

```
Lemma bio_ok : ∀p q,
  wf f p → wf f q → (p ∼ q ↔ bio p q = true).
```

Théorème : $P \sim_{IO} Q \iff P \simeq Q$

Théorème : $P \sim_{IO} Q \iff P \simeq Q$

Donc $\simeq$ n'est pas si compliquée !

- décidable
- axiomatisable :

$$\Pi_{k+1} a(x).P \simeq a(x).(P \parallel \Pi_k a(x).P)$$

```
Theorem IObis_correct : ∀p q,
    wf f p → wf f q → IObis p q → equiv p q.
```

Conclusion