MPRI Concurrency (course number 2-3) 2005-2006: π-calculus

2005-11-09

http://pauillac.inria.fr/∼leifer/teaching/mpri-concurrency-2005/

James J. Leifer
INRIA Rocquencourt

James.Leifer@inria.fr

Process abstractions

We don't need CCS-style "definitions" for infinite behaviour since we have replication, $!P$, as shown later. Nonetheless, they are convenient. In π-calculus, we call them process abstractions:

$$F = (u_1, ..., u_k).P$$

Instantiation takes an abstraction and a vector of names and gives back a process:

$$F\langle x_1, ..., x_k\rangle = \{x_1/u_1, ..., x_k/u_k\}P$$

Booleans

In Ocaml,

```
type bool = True | False;;
let cases b t f = match b with True -> t | False -> f;;
let not b = cases b False True;;
```

In π-calculus,

$$
\begin{aligned}
\mathit{True} &= (l).l(t,f).\overline{t} \\
\mathit{False} &= (l).l(t,f).\overline{f} \\
\mathit{cases}(P,Q) &= (l).\boldsymbol{\nu} t.\boldsymbol{\nu} f.\overline{l}\langle t,f\rangle.(t.P + f.Q) \\
\mathit{not} &= (l,k).\mathit{cases}(\mathit{False}\langle k\rangle, \mathit{True}\langle k\rangle)\langle l\rangle
\end{aligned}
$$

Example: show that

$$\boldsymbol{\nu} l.(\mathit{True}\langle l\rangle \mid \mathit{not}\langle l,k\rangle) \longrightarrow^{*} \mathit{False}\langle k\rangle$$

From linear to replicated data

Can we reuse a boolean? No...

Example: show that we don't have

$$\boldsymbol{\nu} l.(\mathit{True}\langle l \rangle \mid \mathit{not}\langle l, k_0 \rangle \mid \mathit{not}\langle l, k_1 \rangle) \longrightarrow^* \mathit{False}\langle k_0 \rangle \mid \mathit{False}\langle k_1 \rangle$$

Why? After we use $\mathit{True}\langle l \rangle$ once, we "exhaust" it. The solution is to use replication:

$$\mathit{True}' = (l).!l(t, f).\overline{t}$$
$$\mathit{False}' = (l).!l(t, f).\overline{f}$$

Interlude: encoding recursive definitions in terms of replication

Consider the recursive abstraction ("definition" in CCS):

$$F = (\vec{x}).P$$

where P may well contain recursive calls to F of the form $F\langle\vec{z}\rangle$.

We can replace the RHS with the following process abstraction containing no mention of F:

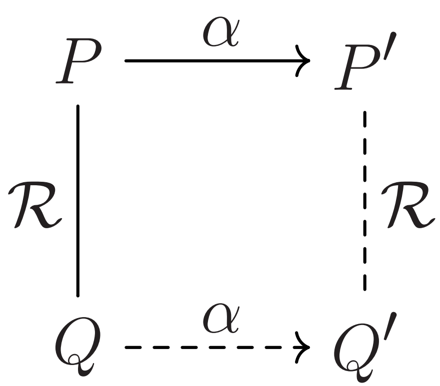
$$(\vec{x}).\boldsymbol{\nu} f.(\overline{f}\langle\vec{x}\rangle \mid !f(\vec{x}).\{\overline{f}/F\}P)$$

provided that f is fresh.

Example: compare the transitions of $F\langle u, v\rangle$, where $F = (x, y).\overline{x}y.F\langle y, x\rangle$ to those of its encoding. Notice the extra τ steps.

Strong bisimulation

A relation $\mathcal{R}$ is a strong bisimulation if for all $(P, Q) \in \mathcal{R}$ and $P \xrightarrow{\alpha} P'$, where $\mathsf{bn}(\alpha) \cap \mathsf{fn}(Q) = \varnothing$, there exists Q' such that $Q \xrightarrow{\alpha} Q'$ and $(P', Q') \in \mathcal{R}$, and symmetrically.



Strong bisimilarity $\sim_\ell$ is the largest strong bisimulation.

Bisimulation proofs

Theorem: $P \equiv Q$ implies $P \sim_\ell Q$.

Can you think of a counterexample to the converse?

Some easy results:

1. $P \mid \mathbf{0} \sim_\ell P$
2. $\overline{x}y.\boldsymbol{\nu} z.P \sim_\ell \boldsymbol{\nu} z.\overline{x}y.P$, if $z \notin \{x, y\}$
3. $x(y).\boldsymbol{\nu} z.P \sim_\ell \boldsymbol{\nu} z.x(y).P$, if $z \notin \{x, y\}$
4. $!\boldsymbol{\nu} z.P \not\sim_\ell \boldsymbol{\nu} z.!P$ for some P

More difficult:

1. $\boldsymbol{\nu} x.P \mid Q \sim_\ell \boldsymbol{\nu} x.(P \mid Q)$, for $x \notin \mathsf{fn}(Q)$
2. $P \sim_\ell Q$ implies $P \mid S \sim_\ell Q \mid S$
3. $!P \mid !P \sim_\ell !P$
4. $!!P \sim_\ell !P$

Congruence with respect to parallel

Theorem: $P \sim_\ell Q$ implies $P \mid S \sim_\ell Q \mid S$

Proof: Consider $\mathcal{R} = \{(P \mid S, Q \mid S) \ / \ P \sim_\ell Q\}$. If we can show $\mathcal{R} \subseteq \sim_\ell$ then we're done: if $P \sim_\ell Q$, then $(P \mid S, Q \mid S) \in \mathcal{R}$, thus $P \mid S \sim_\ell Q \mid S$.

Claim: $\mathcal{R}$ is a bisimulation. Suppose $P \sim_\ell Q$ and $P \mid S \xrightarrow{\alpha} P_0$, where $\mathsf{bn}(\alpha) \cap \mathsf{fn}(Q \mid S) = \varnothing$.

What are the cases to consider?

Congruence with respect to parallel: case analysis

P is solely responsible:

- $P \xrightarrow{\alpha} P'$ and $P_0 = P' \mid S$ and $\mathsf{bn}(\alpha) \cap \mathsf{fn}(S) = \varnothing$

S is solely responsible:

- $S \xrightarrow{\alpha} S'$ and $P_0 = P \mid S'$ and $\mathsf{bn}(\alpha) \cap \mathsf{fn}(P) = \varnothing$

P and S are jointly responsible:

- $P \xrightarrow{\overline{x}y} P'$ and $S \xrightarrow{xy} S'$ and $P_0 = P' \mid S'$ and $\alpha = \tau$
- $P \xrightarrow{xy} P'$ and $S \xrightarrow{\overline{x}y} S'$ and $P_0 = P' \mid S'$ and $\alpha = \tau$
- $P \xrightarrow{\overline{x}(y)} P'$ and $S \xrightarrow{xy} S'$ and $P_0 = \boldsymbol{\nu} y.(P' \mid S')$ and $\alpha = \tau$ and $y \notin \mathsf{fn}(S)$
- $P \xrightarrow{xy} P'$ and $S \xrightarrow{\overline{x}(y)} S'$ and $P_0 = \boldsymbol{\nu} y.(P' \mid S')$ and $\alpha = \tau$ and $y \notin \mathsf{fn}(P)$: careful!

Congruence with respect to parallel: the tricky case

Case: $P \xrightarrow{xy} P'$ and $S \xrightarrow{\overline{x}(y)} S'$ and $P_0 = \boldsymbol{\nu} y.(P' \mid S')$ and $\alpha = \tau$ and $y \notin \mathsf{fn}(P)$. The following lemmas can help:

1. If $P \xrightarrow{xy} P'$ and $y \notin \mathsf{fn}(P)$ then $P \xrightarrow{xy'} \{y'/y\}P'$.
2. If $S \xrightarrow{\overline{x}(y)} S'$ and $y' \notin \mathsf{fn}(S)$ then $S \xrightarrow{\overline{x}(y')} \{y'/y\}S'$.

Now, let y' be fresh. We can apply both lemmas. By alpha-conversion, $P_0 = \boldsymbol{\nu} y'.(\{y'/y\}P' \mid \{y'/y\}S')$

Since $P \sim_\ell Q$, there exists Q'' such that $Q \xrightarrow{xy'} Q''$ and $\{y'/y\}P' \sim_\ell Q''$. Since y' is fresh,

$$Q \mid S \xrightarrow{\tau} \boldsymbol{\nu} y'.(Q'' \mid \{y'/y\}S')$$

Our bisimulation isn't big enough! Take instead:

$$\mathcal{R} = \{(\boldsymbol{\nu}\vec{z}.(P \mid S), \boldsymbol{\nu}\vec{z}.(Q \mid S)) \;/\; P \sim_\ell Q\}$$

Exercises for next lecture

1. (a) Show that $!\boldsymbol{\nu} z.P \sim_\ell \boldsymbol{\nu} z.!P$ is not generally true. Make the argument precise by giving a concrete process P and a sequence of labelled transitions showing that bisimulation doesn't hold.

   (b) Let us say that a process Q has a weak barb b, written $Q{\Downarrow}b$ if Q is eventually able to output on b, i.e. there exists Q_0, Q_1, and $\vec{y}$ such that $Q \longrightarrow^* \boldsymbol{\nu}\vec{y}.(\overline{b}u.Q_0 \mid Q_1)$ with $b \notin \vec{y}$.

   Find a context T that can distinguish the two processes above, i.e. such that $(\boldsymbol{\nu} z.!P \mid T){\Downarrow}b$ but not $(!\boldsymbol{\nu} z.P \mid T){\Downarrow}b$.

   (c) Give an example of a general class of processes P for which the bisimulation would hold?

2. Recall the encoding of recursive abstractions in terms of replication.

   (a) Write the process $F\langle x, y\rangle$ in terms of replication, where the abstraction F is defined as follows:

$$F = (u, v).u.F\langle u, v\rangle$$

   (b) Consider the pair of mutually recursive definition

$$G = (u, v).(u.H\langle u, v\rangle \mid k.H\langle u, v\rangle)$$
$$H = (u, v).v.G\langle u, v\rangle$$

   Write the process $G\langle x, y\rangle$ in terms of replication. (Note that we didn't discuss the coding of mutually recursive definitions so you have to invent the technique yourself!)

3. Prove $!P \mid !P \sim_\ell !P$. To make the problem easier, replace the labelled transition rule for replication by the following ones that make the analysis much easier:

$$\frac{P \xrightarrow{\alpha} P'}{!P \xrightarrow{\alpha} P' \mid !P} \text{if } \mathsf{bn}(\alpha) \cap \mathsf{fn}(P) = \varnothing \qquad \text{(lab-bang-simple)}$$

$$\frac{P \xrightarrow{\overline{x}y} P' \qquad P \xrightarrow{xy} P''}{!P \xrightarrow{\tau} (P' \mid P'') \mid !P} \qquad \text{(lab-bang-comm)}$$

$$\frac{P \xrightarrow{\overline{x}(y)} P' \qquad P \xrightarrow{xy} P''}{!P \xrightarrow{\tau} \boldsymbol{\nu} y.(P' \mid P'') \mid !P} \text{if } y \notin \mathsf{fn}(P) \qquad \text{(lab-bang-close)}$$

Furthermore, feel free to use structural congruence (e.g. $!P \equiv P \mid !P$) instead of process equality anywhere you need it in the proof.