Concurrency 9

The asynchronous π-calculus

Catuscia Palamidessi

catuscia@lix.polytechnique.fr

www.lix.polytechnique.fr/~catuscia

Page of the course:

www.lix.polytechnique.fr/~catuscia/DEA

The asynchronous π-calculus

- If P | Q is interpreted as the composition of two remote processes P and Q, then the mechanism of synchronous communication seems unrealistic

$$\bar{x}y.P \mid x(y).Q \xrightarrow{\tau} P \mid Q$$

- Synchronization combined with choice seems even less realistic

$$\begin{tikzcd}
 & P_1 \mid Q_2 \\
\bar{x}_1 y.P_1 + x_2(y).P_2 \mid \bar{x}_2 y.Q_1 + x_1(y).Q_2 \arrow[ur, "\tau"] \arrow[dr, "\tau"] & \\
 & P_2 \mid Q_1
\end{tikzcd}$$

- In a distributed system, communication is asynchronous (exchange of messages). The send takes place independently of the readiness of a receiver, and it is not blocking

- The asynchnous π-calculus: A calculus for representing asynchnous communication. It was introduced independently by Honda-Tokoro [1991] and by Boudol [1992]

The asynchronous π-calculus: syntax

- It differs from the π-calculus for the absence of the output prefix (replaced by output action) and also for the absence of the +

$\pi ::= x(y) \mid \tau$	action prefixes (input, silent) x, y are channel names
$P ::= O$	inaction
$\mid \quad \pi.P$	prefix
$\mid \quad \bar{x}y$	output action
$\mid \quad P \mid P$	parallel
$\mid \quad (\nu x)P$	restriction, new name
$\mid \quad !\, P$	replication

The asynchronous π-calculus: syntax

- The operational semantics of the asynchronous π-calculus (π_a) are the same as those of the (synchronous) π-calculus (π), we only eliminate the rule for + and replace the output rule with the following:

$$\mathit{Output} \quad \frac{}{\bar{x}y \xrightarrow{\bar{x}y} 0}$$

- The early and late bisimulations are obtained as usual

- The interpretation is as follows:
  - The send takes place when the output action is at the top-level $(\nu y)(\bar{x}y \mid P) \mid Q$
  - The receive takes place when the output action matches a corresponding input, i.e. when we apply the rule comm or close

$$\bar{x}y \mid P \mid x(z).Q \longrightarrow^* \bar{x}y \mid P' \mid x(z).Q \longrightarrow P' \mid Q[z/y]$$

π_a versus π

- Clearly the (synchronous) π–calculus is more expressive than the asynchronous π–calculus. In fact, the latter is practically a subset of the former.

  Indeed, the output action can be encoded into the output-prefix process with continuation 0. This encoding is correct up to strong bisimulation:

$$\bar{x}y \sim \bar{x}y.0$$

- What about the opposite direction?

  In general we would be happy with an encoding $[\![\cdot]\!] : \pi \longrightarrow \pi_a$

  - Compositional wrt the operators $[\![P \ op \ Q]\!] = C_{op}[[\![P]\!], [\![Q]\!]]$
  - (Preferably) homomorphic wrt | (distribution-preserving) $[\![P \mid Q]\!] = [\![P]\!] \mid [\![Q]\!]$
  - Preserving some kind of semantics. Here there are several possibilities
    - Preserving observables $Obs(P) = Obs([\![P]\!])$
    - Preserving equivalence $[\![P]\!] \ equiv \ [\![Q]\!] \ \Rightarrow \ P \ equiv' \ Q$ (soundness)

      $[\![P]\!] \ equiv \ [\![Q]\!] \ \Leftrightarrow \ P \ equiv' \ Q$ (full abstraction)
    - (Preferably) the encoding should not introduce divergencies, in the sense that if in the original process all the computations converge, then the same holds for its translation. Note that weak bisimulations are insensitive wrt divergencies

π_a versus π

- Encoding the output prefix

  Honda-Tokoro [1992] provided the following encoding of π into π_a. : The idea is to force both partners to proceed only when it is sure that the communication can take place, by using a sort of rendez-vous protocol

  - $[\![\bar{x}y.P]\!] = x(z).(\bar{z}y \mid [\![P]\!])$
  - $[\![x(y).Q]\!] = (z)(\bar{x}z.(z(y).[\![Q]\!]))$

  $[\![\cdot]\!]$ is homomorphic for all the other operators. Namely:

  - $[\![0]\!] = 0$
  - $[\![P \mid Q]\!] = [\![P]\!] \mid [\![Q]\!]$
  - $[\![(\nu x)P]\!] = (\nu x)[\![P]\!]$
  - $[\![!\, P]\!] = !\, [\![P]\!]$

- This encoding is sound wrt weak bisimulation and fully abstract wrt a weaker relation called coupled bisimulation. Furthermore it does not introduce divergencies

- **Exercise.** Define an encoding with the same properties in which the translation of the sender does the output first. (Such a kind of encoding was defined by Boudol [1992].)

π_a versus π

- **Encoding choice**

- It is very easy to encode the so-called blind choice (or internal choice) $P \oplus Q$

$$\frac{}{P \oplus Q \xrightarrow{\tau} P} \qquad \frac{}{P \oplus Q \xrightarrow{\tau} Q}$$

In π this operator can be represented by the construct $\tau.P + \tau.Q$

Exercise: Let π^- be π without the + operator, and $\pi^{\oplus}$ be π where the + operator can only occur in the context of (a construct representing) a blind choice. Give an encoding $[\![\cdot]\!] : \pi^{\oplus} \longrightarrow \pi^-$ such that $\forall P\ [\![P]\!] \sim P$

π_a versus π

- **Encoding input-guarded choice**

- Input-guarded choice is a construct of the form: $\sum_{i \in I} x_i(y_i).P_i$

- Let π^i be π where + can only occur in an input-guarded choice. The following encoding of π^i into π_a was defined by Nestmann and Pierce [1996]

$$[\![\sum_{i \in I} x_i(y_i)P_i]\!] \quad = \quad (\nu l)(\bar{\ell}\, \mathit{true} \mid \prod_{i \in I} \mathit{Branch}_{\ell i})$$

$$\begin{array}{rcll} \mathit{Branch}_{\ell i} & = & x_i(z_i).\ell(w).(\mathit{if} & w = \mathit{true} \\ & & & \mathit{then}\ (\bar{\ell}\, \mathit{false} \mid [\![P_i]\!]) \\ & & & \mathit{else}\ (\bar{\ell}\, \mathit{false} \mid \bar{x}_i z_i)\ ) \end{array}$$

- This encoding is fully abstract wrt coupled bisimulation, and it does not introduce divergencies.

- **Exercise (difficult):** Prove that the full π cannot be encoded in π_a