Probabilistic Methods in Concurrency

Lecture 7
The probabilistic asynchronous π-calculus

Catuscia Palamidessi
catuscia@lix.polytechnique.fr
www.lix.polytechnique.fr/~catuscia

Page of the course:
www.lix.polytechnique.fr/~catuscia/teaching/Pisa/
The probabilistic asynchronous $\pi$-calculus

- Originally developed as an intermediate language for the fully distributed implementation of the $\pi$-calculus (Herescu and Palamidessi)
- The results of Lecture 4 show that a fully distributed implementation of $\pi$ must necessarily be randomized
- A two-steps approach:

$$\begin{align*}
\pi & \downarrow [[ ]] \\
\text{probabilistic asynchronous } \pi & \downarrow \lll \ggg \\
\text{distributed machine}
\end{align*}$$

Advantages:
the correctness proof is easier since $[[ ]]$ (which is the difficult part of the implementation) is between two similar languages.
\( \pi_{pa} \): the Probabilistic Asynchronous \( \pi \)

Syntax: based on the asynchronous \( \pi \) of Amadio, Castellani, Sangiorgi

\[
g ::= x(y) \mid \tau \quad \text{prefixes}
\]

\[
P ::= \Sigma_i p_i g_i . P_i \quad \text{pr. inp. guard. choice } \Sigma_i p_i = 1
\]

| \( x^y \) | \( P \mid P \) | \( (x) P \) | \( \text{rec}_A P \) | \( A \) |
| output action | parallel | new name | recursion | procedure name |
The operational semantics of $\pi_{pa}$

- Based on the Probabilistic Automata of Segala and Lynch
  - Distinction between
    - nondeterministic behavior (choice of the scheduler) and
    - probabilistic behavior (choice of the process)

**Scheduling Policy:**
The scheduler chooses the group of transitions

**Execution:**
The process chooses probabilistically the transition within the group
The operational semantics of \( \pi_{pa} \)

- **Representation of a group of transition**
  \[
  \pi \{ --g_i \rightarrow p_i P_i \}_i
  \]

- **Rules**
  - **Choice**
    \[
    \Sigma_i p_i g_i . P_i \{ --g_i \rightarrow p_i P_i \}_i
    \]
  - **Par**
    \[
    \frac{P \{ --g_i \rightarrow p_i P_i \}_i}{Q \mid P \{ --g_i \rightarrow p_i Q \mid P_i \}_i}
    \]
The operational semantics of $\pi_{pa}$

- Rules (continued)

\[
P \{--x_i(y_i)\rightarrow p_i P_i\}_i \quad Q \{--x^z\rightarrow 1 Q'\}_i
\]

\[
\text{Com} \quad \frac{}{P \mid Q \{--\tau\rightarrow p_i P_i[z/y_i] \mid Q'\}_{x_i=} \cup \{--x_i(y_i)\rightarrow p_i P_i \mid Q\}_{x_i=/=}x}
\]

\[
P \{--x_i(y_i)\rightarrow p_i P_i\}_i
\]

\[
\text{Res} \quad \frac{}{(x) P \{--x_i(y_i)\rightarrow q_i (x) P_i\}_{x_i=} \sim x}
\]

$q_i$ renormalized
The expressive power of $\pi$

- Example of distributed agreement: the leader election problem
Implementation of $\pi_{pa}$

- **Compilation in Java** $\langle\langle \quad \rangle\rangle : \pi_{pa} \rightarrow \text{Java}$

- **Distributed**
  $\langle\langle P \| Q \rangle\rangle = \langle\langle P \rangle\rangle.start(); \langle\langle Q \rangle\rangle.start();$

- **Compositional**
  $\langle\langle P \op Q \rangle\rangle = \langle\langle P \rangle\rangle.jop \langle\langle Q \rangle\rangle$ for all $\op$

- **Channels** are one-position buffers with test-and-set (synchronized) methods for input and output.

- **The probabilistic input guarded construct is implemented as a while loop in which channels to be tried are selected according to their probability. The loop repeats until an input is successful**