Probabilistic Methods in Concurrency

Lecture 8

Encoding the $\pi$-calculus into the probabilistic asynchronous $\pi$-calculus

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

Page of the course:
www.lix.polytechnique.fr/~catuscia/teaching/Pisa/
Encoding $\pi$ into $\pi_{pa}$

- $[[ ]] : \pi \rightarrow \pi_{pa}$

- **Fully distributed**
  $$[[ P \mid Q ]] = [[ P ]] \mid [[ Q ]]$$

- **Uniform**
  $$[[ P \sigma ]] = [[ P ]]\sigma$$

- **Correct wrt a notion of probabilistic testing semantics**
  $P$ must $O$ iff $[[ P ]]$ must $[[ O ]]$ with prob 1
Encoding $\pi$ into $\pi_{pa}$

- Idea:
  - Every mixed choice is translated into a parallel comp. of processes corresponding to the branches, plus a lock $f$
  - The input processes compete for acquiring both its own lock and the lock of the partner
  - The input process which succeeds first, establishes the communication. The other alternatives are discarded

The problem is reduced to a generalized dining philosophers problem where each fork (lock) can be adjacent to more than two philosophers
Dining Philosophers: classic case

Each fork is shared by exactly two philosophers
The algorithm of Lehmann and Rabin

1. Think
2. choose first_fork in {left,right}  %commit
3. if taken(first_fork) then goto 3
4. take(first_fork)
5. if taken(first_fork) then {release(firstfork); goto 2}
6. take(second_fork)
7. eat
8. release(second_fork)
9. release(first_fork)
10. goto 1
Problems

• Wrt to our encoding goal, the algorithm of Lehmann and Rabin has two problems:

1. It only works for certain kinds of graphs

2. It works only for fair schedulers

• Problem 2 however can be solved by replacing the busy waiting in step 3 with suspension.

[Duflot, Friburg, Picaronny 2002] - see also Herescu’s PhD thesis
The algorithm of Lehmann and Rabin
Modified so to avoid the need for fairness

1. Think
2. choose first_fork in {left,right}  %commit
3. if taken(first_fork) then goto 3
4. take(first_fork)
5. if taken(first_fork) then goto 2
6. take(second_fork)
7. eat
8. release(second_fork)
9. release(first_fork)
10. goto 1
Dining Phils: generalized case

Each fork can be shared by more than two philosophers
Dining Phils: generalized case

• **Theorem:** The algorithm of Lehmann and Rabin is deadlock-free **if and only if** all cycles are pairwise disconnected.

• There are essentially three ways in which two cycles can be connected:
Proof of the theorem

• If part) Each cycle can be considered separately. On each of them the classic algorithm is deadlock-free. Some additional care must be taken for the arcs that are not part of the cycle.

• Only if part) By analysis of the three possible cases. Actually they are all similar. We illustrate the first case.
Proof of the theorem

- The initial situation has probability $p > 0$
- The scheduler forces the processes to loop
- Hence the system has a deadlock (livelock) with probability $p$

- Note that this scheduler is not fair. However we can define even a fair scheduler which induces an infinite loop with probability $> 0$. The idea is to have a scheduler that "gives up" after $n$ attempts when the process keep choosing the "wrong" fork, but that increases (by $f$) its "stubbornness" at every round.

- With a suitable choice of $n$ and $f$ we have that the probability of a loop is $p/4$
Solution for the Generalized DP

• As we have seen, the algorithm of Lehmann and Rabin does not work on general graphs

• However, it is easy to modify the algorithm so that it works in general

• The idea is to reduce the problem to the pairwise disconnected cycles case:

  Each fork is initially associated with one token. Each phil needs to acquire a token in order to participate to the competition. After this initial phase, the algorithm is the same as the Lehmann & Rabin’s Theorem: The competing phils determine a graph in which all cycles are pairwise disconnected

  Proof: By case analysis. To have a situation with two connected cycles we would need a node with two tokens.
Dining Phils: generalized case

Reduction to the classic case: each fork is initially associated with a token. Each phil needs to acquire a token in order to participate to the competition. The competing phils determine a set of subgraphs in which each subgraph contains at most one cycle.