MPRI C.2.3 - Concurrency

Probabilistic models and applications
Lecture 3

Kostas Chatzikokolakis

Jan 10, 2012

Outline of the lectures

- Dec 13
- Dec 20
- Jan 10
- Jan 17
- Jan 24

Outline of the lectures

- The need for randomization
- Probabilistic automata
- Probabilistic bisimulation
- Probabilistic calculi
- Encoding of the pi-calculus into the asynchronous fragment
- Introduction to probabilistic model checking and PRISM
- Verification of anonymity protocols: Dining Cryptographers, Crowds

Exercises from the last lecture

Exercise 1: Show that probabilistic bisimulation is a generalization of traditional bisimulation

Puzzle from the last lecture

- I select two real numbers in some arbitrary way
- I put them in two envelopes, you select one of them (in any way you want)
- You see the number and you have 2 options: keep it, or exchange it with the other envelope
- Your goal is to select the bigger number
- Is there any strategy that guarantees winning this game with pb higher than 1/2?

Outline

Encoding of π-calculus in the asynchronous fragment

Encoding of π-calculus in the probabilistic asynchronous π

Non-deterministic transition systems and CTL

Probabilistic asynchronous π-calculus

The input guarded choice is probabilistic.

The prefixes

$$\alpha ::= x(y) \mid \tau \qquad \text{input | silent action}$$

The processes

$$
\begin{array}{rcll}
P & ::= & 0 & \text{inaction} \\
& \mid & \Sigma_i p_i \alpha_i . P_i & \text{probabilistic choice} \\
& \mid & \bar{x} y & \text{output} \\
& \mid & P \mid P & \text{parallel} \\
& \mid & (\nu x) P & \text{new name} \\
& \mid & !\, P & \text{replication}
\end{array}
$$

where $\Sigma_i p_i = 1$

Expressive power of π_a wrt π

- Clearly π is at least as expressive as π_a
- The latter is practically a subset of the former:
  $\bar{x}y$ (in π_a) can be seen as $\bar{x}y.0$ (in π)
- What about the opposite direction? We need to encode:
  - the output prefix
  - the choice operator
    Three types of choice: internal, separate, mixed
- In general, in order to compare the expressive power of two languages, we look for the existence/non existence of an encoding with certain properties among these languages
- What is a good notion of encoding to be used as basis to measure the relative expressive power?

A "good" notion of encoding

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

- 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

$$[\![P]\!] \; equiv \; [\![Q]\!] \Rightarrow P \; equiv' \; Q \quad \text{(soundness)}$$

$$[\![P]\!] \; equiv \; [\![Q]\!] \Leftarrow P \; equiv' \; Q \quad \text{(completeness)}$$

$$[\![P]\!] \; equiv \; [\![Q]\!] \Leftrightarrow P \; equiv' \; Q \quad \text{(full abstraction, correctness)}$$

Testing semantics

- A test O is a process with a distinct success action ω
- A process P may pass O iff there is a computation of $[P|O]$ where ω is enabled
  eg. $a.b + a$ may pass $\bar{a}.\bar{b}.\omega$
- A process P must pass O iff all computations of $[P|O]$ reach a state where ω is enabled
  eg. $a.b + a$ must pass $\bar{a}.\omega$

Testing semantics

- $P \sqsubseteq_{\mathbf{may}} Q$ iff $\forall O : P$ may $O \Rightarrow Q$ may O
- $P \sqsubseteq_{\mathbf{must}} Q$ iff $\forall O : P$ must $O \Rightarrow Q$ must O
- Exercise: are $\sqsubseteq_{\mathbf{may}}, \sqsubseteq_{\mathbf{must}}$ pre-congruences for CCS, π?
- We would like the encodings to satisfy:
  - P may pass O iff $[\![P]\!]$ may pass $[\![O]\!]$
  - P must pass O iff $[\![P]\!]$ must pass $[\![O]\!]$

The encoding of Boudol

Encodes the output prefix (but without choice). Idea: we proceed only when it is sure that the communication can take place, by using a sort of rendez-vous protocol.

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

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

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

The encoding satisfies P may pass O iff $[\![P]\!]$ may pass $[\![O]\!]$

Encoding of Honda-Tokoro

A more compact encoding, it takes two steps instead than three.
The idea is to let the receiver take the initiative.

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

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

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

The encoding satisfies P may pass O iff $[\![P]\!]$ may pass $[\![O]\!]$

Encoding of the output prefix

- The encodings of Boudol and Honda-Tokoro do not satisfy
  P must pass O iff $[\![P]\!]$ must pass $[\![O]\!]$
- This is a problem of fairness
- must testing is preserved if we restrict to fair computations only
- The encodings preserve a version of testing called "fair must testing"

Encoding of internal choice

The blind choice (or internal choice) construct $P \oplus Q$ has the following semantics

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

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

Exercise: Let π be π where the + operator can only occur as a blind choice.

Give an encoding $[\![\cdot]\!] : \pi^{\oplus} \longrightarrow \pi_{\mathsf{a}}$ such that $\forall P \; [\![P]\!] \sim P$

Encoding of input-guarded choice

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

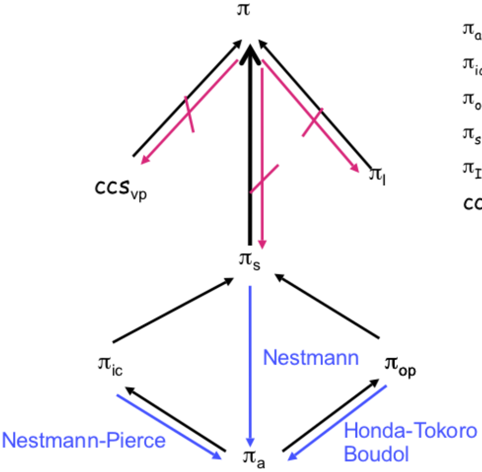
Let π^i be π where + can only occur in an input-guarded choice. The following encoding of π^i into π_a was defined by Nestmann and Pierce [1996]

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

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

Nestmann and Pierce proved that his encoding is fully abstract wrt a notion of equivalence called coupled bisimulation, and it does not introduce divergences.

The π-calculus hierarchy



π_a : asynchronous π

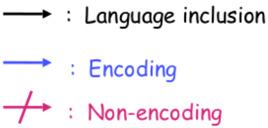
π_{ic} : asynchronous π + input-guarded choic

π_{op} : asynchronous π + output prefix

π_s : asynchronous π + separate choice

π_I : π with internal mobility (Sangiorgi)

ccs_{vp} : value-passing ccs



The separation between π and π_s

This separation result is based on the fact that it is not possible to solve the symmetric leader election problem in π_s, while it is possible in π

Leader Election Problem (LEP): All the nodes of a distributed system must agree on who is the leader. This means that in every possible computation, all the nodes must eventually output the name of the leader on a special channel

- No deadlock
- No livelock
- No conflict (only one leader must be elected, every process outputs its name and only its name)

The separation between π and π_s

Theorem

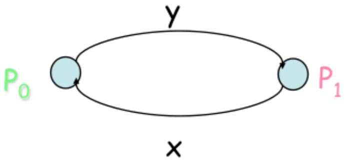
It is impossible to write in π_s a symmetric (having an automorphism with a single orbit) solution to the LEP.

Crucial point: Diamond lemma: when a node P_i performs an action, any other node P_j can perform the same action returning to a symmetric state. (Note: this does not hold in π)

Corollary: in a symmetric π_s network trying to solve the LEP, there is at least one diverging computation.

The separation between π and π_s

Remark: In π (in π with mixed choice) we can easily write a symmetric solution for the LEP in a network of two nodes:



$$P_0 = x.\overline{out}\,0 + \bar{y}.\overline{out}\,1$$

$$P_1 = y.\overline{out}\,1 + \bar{x}.\overline{out}\,0$$

The separation between π and π_s

Corollary: there does not exists an encoding of π in π_s which is homomorphic wrt $|$ and renaming, and preserves the observables on every computation.

Proof (scketch): An encoding homomorphic wrt $|$ and renaming transforms a symmetric solutions to the LEP in the source language into a symmetric solution to the LEP in the target language.

Outline

Encoding of π-calculus in the asynchronous fragment

Encoding of π-calculus in the probabilistic asynchronous π

Non-deterministic transition systems and CTL

Probabilistic testing semantics

- Test O: same as before (but can be probabilistic)
- $sexec([P|O])$ the set of successful executions of $[P|O]$ (those containing ω)
- Note: $sexec([P|O])$ can be obtained as a countable union of disjoint cones
- $\mu_\sigma(sexec([P|O]))$ the probability of success under scheduler σ

Probabilistic testing semantics

- P may pass O iff $\exists\sigma$: $\mu_\sigma(sexec([P|O])) > 0$
- P must pass O iff $\forall\sigma$: $\mu_\sigma(sexec([P|O])) = 1$
- $\sqsubseteq_{\mathbf{may}}, \sqsubseteq_{\mathbf{must}}$: same as before
  - $P \sqsubseteq_{\mathbf{may}} Q$ iff $\forall O : P$ may $O \Rightarrow Q$ may O
  - $P \sqsubseteq_{\mathbf{must}} Q$ iff $\forall O : P$ must $O \Rightarrow Q$ must O

Probabilistic testing semantics

Exercises:

- Give alternative definitions that consider the exact probabilities of success. Are they equivalent to the ones of the previous slide?
- Show that for all probabilistic CCS processes P, Q:
  - $P +_p Q \sqsubseteq_{\mathbf{may}} \tau.P + \tau.Q$
  - $\tau.P + \tau.Q \sqsubseteq_{\mathbf{must}} P +_p Q$

Encoding of π into π_{ap} (high level idea)

- Similarly to the encoding of Nestmann-Pierce, the branches of the choice are put in parallel together with a lock
- An input process will try to acquire both its own lock and its partner's lock
- If a lock is not available it aborts and tries again
- Crucial point: the locks are tried in random order, similarly to the solution of the Dining Philosophers problem
- This ensures that divergences will have probability 0

Encoding of π into π_{ap} (high level idea)

- This encoding satisfies:
  - P may pass O iff $[\![P]\!]$ may pass $[\![O]\!]$
  - P must pass O iff $[\![P]\!]$ must pass $[\![O]\!]$
- under a weak assumption on the schedulers (weaker than fairness)
- For details: C. Palamidessi, M. O. Herescu. A randomized encoding of the -calculus with mixed choice. Theoretical Computer Science. 335(2-3): 373-404, 2005.

Outline

Encoding of π-calculus in the asynchronous fragment

Encoding of π-calculus in the probabilistic asynchronous π

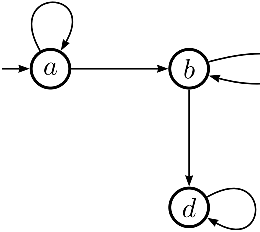
Non-deterministic transition systems and CTL

Model Checking

- Techniques to automatically verify systems (software, hardware, protocols, . . .) using automata and temporal logics
- We give a formal model M of the system (typically using some kind of automaton)
- We give a formal property φ to verify (typically a formula in some temporal logic)
- An algorithm decides whether M satisfies the formula φ, written

$$M \models \varphi$$

Non-deterministic Transition Systems

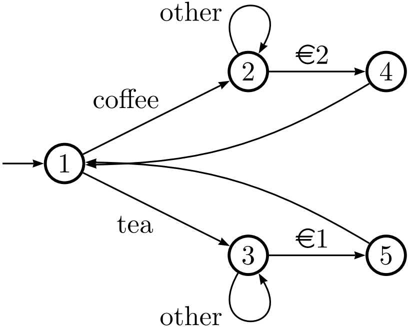


A tuple $(S, s_0, \to)$ where

- S is a finite set of states
- $s_0 \in S$ is the initial state
- $\to \subseteq S \times S$ is a transition relation.
  We write $s_1 \to s_2$ when there $(s_1, s_2) \in \to$

Non-deterministic Transition Systems

Example: a coffee machine



We want to verify properties like "the machine always goes back to its initial state".

Computation Tree Logic (CTL)

- Formulas are evaluated on a transition system M
- On each state s we assign a set $L(s)$ of atomic propositions. These are the propositions that we consider to be true on this state.
- Two types of formulas:
  - state formulas are evaluated on states
  - path formulas are evaluated on infinite sequences of states

Computation Tree Logic (CTL)

Syntax:

$$\varphi ::= p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi \mid A\psi \mid E\psi \quad \text{state formulas}$$
$$\psi ::= \circ\varphi \mid \Box\varphi \mid \Diamond\varphi \mid \varphi U \varphi' \quad \text{path formulas}$$

Path quantifiers:

$A\psi$ for all paths starting from this state ψ holds

$E\psi$ there exists a path starting from this state s.t. ψ holds

Temporal operators:

$\circ\varphi$ in the next state φ holds ($X\varphi$)

$\Box\varphi$ (always) in all states φ holds ($G\varphi$)

$\Diamond\varphi$ (eventually) in some future state φ holds ($F\varphi$)

$\varphi U \varphi'$ φ holds in all states until a state where φ' holds

Computation Tree Logic (CTL)