MPRI C.2.3 - Concurrency

Probabilistic models and applications
Lecture 3

Kostas Chatzikokolakis

Jan 10, 2014
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
- Testing equivalence
- Introduction to probabilistic model checking and PRISM
- Metrics for probabilistic processes
- Verification of anonymity protocols: Dining Cryptographers, Crowds
Question 1: Show that probabilistic bisimulation is a generalization of traditional bisimulation
Questions from the last lecture

Question 2: are $\sqsubseteq_{\text{may}}$, $\sqsubseteq_{\text{must}}$ pre-congruences for CCS?
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?
A test $O$ is a process with a distinct success action $\omega$

A process $P$ may pass $O$ iff there is a computation of $[P|O]$ where $\omega$ 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 $\omega$ is enabled
eg. $a.b + a$ must pass $\bar{a}.\omega$
Testing semantics

- \( P \sqsubseteq_{\text{may}} Q \iff \forall O : P \text{ may } O \Rightarrow Q \text{ may } O \)
- \( P \sqsubseteq_{\text{must}} Q \iff \forall O : P \text{ must } O \Rightarrow Q \text{ must } O \)
Probabilistic testing semantics

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

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

Examples:

- $P + \frac{1}{2} Q \approx_{\text{may/must}} P + \frac{1}{3} Q$

- $(\tau.P) + \frac{1}{2} (\tau.Q) \approx_{\text{may/must}} \tau.(P + \frac{1}{2} Q)$
Probabilistic testing semantics

Examples:

- $P + \frac{1}{2} Q \; \approx_{\text{may/must}} \; P + \frac{1}{3} Q$

- $(\tau.P) + \frac{1}{2} (\tau.Q) \; \approx_{\text{may/must}} \; \tau.(P + \frac{1}{2} Q)$
Probabilistic testing semantics

Questions:

○ Give alternative definitions that consider the exact probabilities of success.

○ Show that for all probabilistic CCS processes $P, Q$:

- $P +_p Q \sqsubseteq_{\text{may}} \tau. P + \tau. Q$
- $\tau. P + \tau. Q \sqsubseteq_{\text{must}} P +_p Q$
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 $\varphi$ to verify (typically a formula in some temporal logic)
- An algorithm decides whether $M$ satisfies the formula $\varphi$, written
  \[ M \models \varphi \]
A tuple \((S, s_0, \rightarrow)\) where

- \(S\) is a **finite** set of states
- \(s_0 \in S\) is the **initial state**
- \(\rightarrow \subseteq S \times S\) is a **transition relation**.

We write \(s_1 \rightarrow s_2\) when there \((s_1, s_2) \in \rightarrow\)
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:

\[ \varphi ::= p \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid \neg \varphi \mid A\psi \mid E\psi \]  state formulas

\[ \psi ::= \circ \varphi \mid \Box \varphi \mid \Diamond \varphi \mid \varphi U \varphi' \]  path formulas

Path quantifiers:

- \( A\psi \) for all paths starting from this state \( \psi \) holds
- \( E\psi \) there exists a path starting from this state s.t. \( \psi \) holds

Temporal operators:

- \( \circ \varphi \) in the next state \( \varphi \) holds (\( X\varphi \))
- \( \Box \varphi \) (always) in all states \( \varphi \) holds (\( G\varphi \))
- \( \Diamond \varphi \) (eventually) in some future state \( \varphi \) holds (\( F\varphi \))
- \( \varphi U \varphi' \) \( \varphi \) holds in all states until a state where \( \varphi' \) holds
Computation Tree Logic (CTL)

finally $P$

globally $P$

next $P$

$P$ until $q$

$AF_P$

$AG_P$

$AX_P$

$A[P U q]$

$EF_P$

$EG_P$

$EX_P$

$E[P U q]$
We fix a transition system $M$

$s \models p$ iff $p \in L(s)$

$s \models \varphi \land \varphi'$ iff $s \models \varphi$ and $s \models \varphi'$

$s \models \varphi \lor \varphi'$ iff $s \models \varphi$ or $s \models \varphi'$

$s \models \neg \varphi$ iff $s \not\models \varphi$

$s \models A\psi$ iff $\forall s \rightarrow s_1 \rightarrow \ldots : s, s_1, s_2, \ldots \models \psi$

$s \models E\psi$ iff $\exists s \rightarrow s_1 \rightarrow \ldots : s, s_1, s_2, \ldots \models \psi$

$s_0, s_1, \ldots \models \circ \varphi$ iff $s_1 \models \varphi$

$s_0, s_1, \ldots \models \Box \varphi$ iff $\forall i \geq 0 : s_i \models \varphi$

$s_0, s_1, \ldots \models \Diamond \varphi$ iff $\exists i \geq 0 : s_i \models \varphi$

$s_0, s_1, \ldots \models \varphi U \varphi'$ iff $\exists i \geq 0 : s_i \models \varphi'$ and $\forall j < i : s_j \models \varphi$
**CTL**

**Model checking problem**

Given a transition system $M$ and a CTL formula $\varphi$, decide whether $M \models \varphi$ (i.e. $s_0 \models \varphi$).

It can be solved in polynomial time.
Computation Tree Logic (CTL)

Example:

- \( L(s_1) = \{ \text{start} \} \) and empty for the other states

- “from any state the machine eventually goes back to its initial state”
  
  \[ A\Box A\Diamond start \]

- But \( s_1 \not\models A\Box A\Diamond start \)
  
  - \( s_2 \not\models A\Diamond start \)
  
  - \( s_2, s_2, \ldots \not\models \Diamond start \)
A transition system where we allow probabilistic transitions

- A tuple \((S, s_0, \rightarrow)\) where
  - \(S\) is a set of states
  - \(s_0 \in S\) is the initial state
  - \(\rightarrow \subseteq S \times Disc(S)\) is a transition relation.
    We write \(s_1 \rightarrow \mu\) when \((s_1, \mu) \in \rightarrow\)
Probabilistic CTL (PCTL)

Syntax:

$\varphi ::= p \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid \neg \varphi \mid P_{\leq \lambda} \psi \mid P_{\geq \lambda} \psi$  state formulas

$\psi ::= o \varphi \mid \Box \varphi \mid \Diamond \varphi \mid \varphi U \varphi'$  path formulas

- A probabilistic extension of CTL
- We replaced the path quantifiers $A, E$ by the operator $P$
- $P_{\leq \lambda}$ means: the probability that $\psi$ holds is at most $\lambda$
- We could also use $P_{< \lambda}, P_{> \lambda}$
An MDP contains nondeterminism, thus we cannot generally define the probability of executions.

We use a scheduler $\sigma$ assigning a single transition to each state.

We can now define a probability measure $\mu_{\sigma}$ on executions.
○ An MDP contains nondeterminism, thus we cannot generally define the probability of executions

○ We use a scheduler $\sigma$ assigning a single transition to each state

○ We can now define a probability measure $\mu_\sigma$ on executions
PCTL, semantics

- An MDP contains nondeterminism, thus we cannot generally define the probability of executions.
- We use a scheduler $\sigma$ assigning a single transition to each state.
- We can now define a probability measure $\mu_\sigma$ on executions.
PCTL, semantics

- An MDP contains nondeterminism, thus we cannot generally define the probability of executions.
- We use a scheduler \( \sigma \) assigning a single transition to each state.
- We can now define a probability measure \( \mu_{\sigma} \) on executions.
PCTL, semantics

Semantics of PCTL:

\[ s \models P_{\leq \lambda} \psi \iff \forall \sigma : \mu_\sigma(\{ss_1 \ldots | s, s_1, s_2, \ldots \models \psi\}) \leq \lambda \]
\[ s \models P_{\geq \lambda} \psi \iff \forall \sigma : \mu_\sigma(\{ss_1 \ldots | s, s_1, s_2, \ldots \models \psi\}) \geq \lambda \]

Exercise: is \( A\varphi \) equivalent to \( P_{\geq 1} \varphi \)?
PCTL, example

\[ s_b \models P_{\leq 0.5} \diamond \text{start} \]

\[ s_b \not\models P_{\geq 0.5} \diamond \text{start} \]
Model checking problem
Given an MDP $M$ and a PCTL formula $\varphi$, decide whether $M \models \varphi$ (i.e. $s_0 \models \varphi$).
Model checking PCTL over MDPs

\[ \varphi ::= p \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid \neg \varphi \mid P_{\leq \lambda} \psi \mid P_{\geq \lambda} \psi \]  
state formulas

\[ \psi ::= \circ \varphi \mid \Box \varphi \mid \Diamond \varphi \mid \varphi U \varphi' \]  
path formulas

- We are going to \textit{recursively evaluate} \( \varphi \) starting from the simplest subformulas
- At each stage we assume that \textit{we can already evaluate all subformulas} on all states
- \textbf{Base case} \( \varphi = p: \quad s \models p \iff p \in L(s) \)
- \textbf{Cases} \( \varphi \land \varphi', \varphi \lor \varphi', \neg \varphi \): trivial
- The interesting cases are \( P_{\leq \lambda} \psi, P_{\geq \lambda} \psi \)
Model checking PCTL over MDPs

\[ \varphi ::= P_{\leq \lambda} \psi \mid P_{\geq \lambda} \psi \]

\[ \psi ::= \circ \varphi \mid \square \varphi \mid \Diamond \varphi \mid \varphi U \varphi' \]

- Define
  
  \[ P_{s^-} \psi = \min_{\sigma} \mu_{\sigma}(\{ss_1 \ldots \mid s, s_1, s_2, \ldots \models \psi\}) \]
  
  \[ P_{s^+} \psi = \max_{\sigma} \mu_{\sigma}(\{ss_1 \ldots \mid s, s_1, s_2, \ldots \models \psi\}) \]

- Then
  
  \[ s \models P_{\leq \lambda} \psi \ \text{iff} \ \ P_{s^+} \psi \leq \lambda \]
  
  \[ s \models P_{\geq \lambda} \psi \ \text{iff} \ \ P_{s^-} \psi \geq \lambda \]

- So the goal is to compute \( P_{s^-} \psi, P_{s^+} \psi \)
Model checking PCTL over MDPs

\[ Pr^+_s \psi, Pr^-_s \psi \]
\[ \psi ::= \circ \varphi \mid \Box \varphi \mid \Diamond \varphi \mid \varphi \mathcal{U} \varphi' \]

○ Simplifications (exercise: prove them formally)
  
  · \( \Diamond \varphi \equiv \text{true} \mathcal{U} \varphi \)
  
  · \( \Box \psi \equiv \neg \Diamond \neg \psi \)
    
    · \( \neg \Diamond \neg \psi \) is not a valid path formula, but
  
  · \( Pr^+_s \neg \psi' = 1 - Pr^-_s \psi' \)
  
  · \( Pr^-_s \neg \psi' = 1 - Pr^+_s \psi' \)

○ The cases \( Pr^+_s \circ \varphi, Pr^-_s \circ \varphi \) are simple (exercise)

○ We will concentrate on \( \varphi \mathcal{U} \varphi' \)
Compute $Pr_S^\varphi U \varphi'$

- We assume that we have evaluated $\varphi, \varphi'$ on all states
- $S_d = \{ s \in S \mid s \models \varphi' \}$ (destination states)
- $S_p = \{ s \in S \mid s \models \varphi \}$ (intermediate states)
- First goal: compute $S_{>0} = \{ s \in S \mid Pr_S^\varphi U \varphi' > 0 \}$
- Define a monotone function $\Lambda : 2^S \to 2^S$:
  \[
  \Lambda(A) = A \cup \{ s \in S_p \mid \forall s \to \mu : \mu(A) > 0 \}
  \]
- $S_{>0} = \Lambda^{|S|}(S_d)$ (a fixpoint of $\Lambda$)
- It can be computed in at most $|S|$ iterations
Compute $Pr_s^{-} \varphi U \varphi'$

- We know:
  - $Pr_s^{-} \varphi U \varphi' = 0, \ s \in S - S_{>0}$
  - $Pr_s^{-} \varphi U \varphi' = 1, \ s \in S_d$

- It remains to determine $Pr_s^{-} \varphi U \varphi'$ for $s \in S'_p = S_{>0} - S_d$

- At each step, the worst scheduler will chose the transition that minimizes the probability of getting to $S_d$:

$$Pr_s^{-} \varphi U \varphi' = \min_{s \rightarrow \mu} \left[ \sum_{t \in S'_p} \mu(t)Pr_t^{-} \varphi U \varphi' + \sum_{t \in S_d} \mu(t) \right]$$
Compute $Pr_s^{-}\varphi U\varphi'$

- We can determine $Pr_s^{-}\varphi U\varphi'$ for all states $s \in S'_p$ by solving a linear optimization program.

- One variable $x_s$ for each state (the solution will be $Pr_s^{-}\varphi U\varphi'$).

- We ask to maximize $\sum_{s \in S'_p} x_s$.

- Subject to the constraints:

  $$x_s \leq \sum_{t \in S'_p} \mu(t)x_t + \sum_{t \in S_d} \mu(t)$$

  for all $s \in S'_p$ and $s \rightarrow \mu$.

- There is always a unique solution, we can compute it using the ellipsoid method in time polynomial in $|M|$. 


Compute $Pr_s^+ \varphi U \varphi'$

- Symmetric to the case $Pr_s^-$

- First compute $S_{>0} = \{ s \in S \mid Pr_s^+ \varphi U \varphi' > 0 \}$

- At each step, the worst scheduler will choose the transition that maximizes the probability of getting to $S_d$:

  $$Pr_s^+ \varphi U \varphi' = \max_{s \rightarrow \mu} \left[ \sum_{t \in S'_p} \mu(t) Pr_t^+ \varphi U \varphi' + \sum_{t \in S_d} \mu(t) \right]$$

- We can determine $Pr_s^+ \varphi U \varphi'$ for all states $s \in S'_p$ by solving a similar linear optimization program.