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

Questions from the last lecture

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

Questions from the last lecture

Question 2: are $\sqsubseteq_{\mathbf{may}}, \sqsubseteq_{\mathbf{must}}$ 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?

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

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

Examples:

- $P +_{1/2} Q \approx_{\mathbf{may}/\mathbf{must}} P +_{1/3} Q$
- $(\tau.P) +_{1/2} (\tau.Q) \approx_{\mathbf{may}/\mathbf{must}} \tau.(P +_{1/2} Q)$

Probabilistic testing semantics

Examples:

- $P +_{1/2} Q \approx_{\mathbf{may/must}} P +_{1/3} Q$
- $(\tau.P) +_{1/2} (\tau.Q) \approx_{\mathbf{may/must}} \tau.(P +_{1/2} Q)$

Probabilistic testing semantics

Questions:

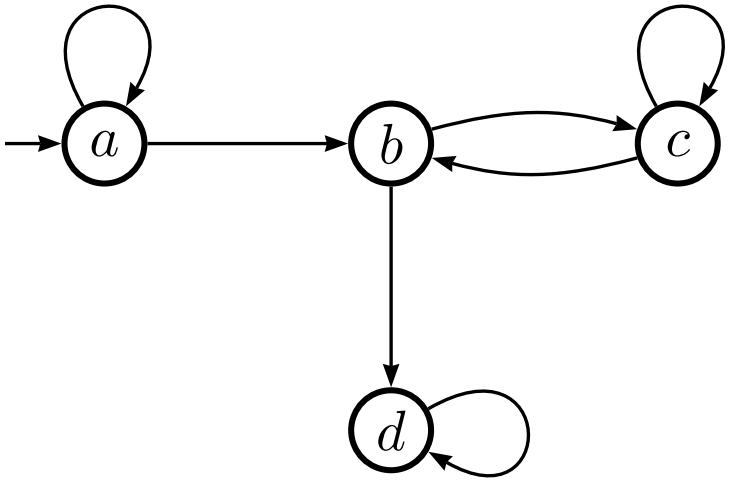
- Give alternative definitions that consider the exact probabilities of success.
- Show that for all probabilistic CCS processes P, Q:
  - $P +_p Q \sqsubseteq_{\mathbf{may}} \tau.P + \tau.Q$
  - $\tau.P + \tau.Q \sqsubseteq_{\mathbf{must}} P +_p Q$

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 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

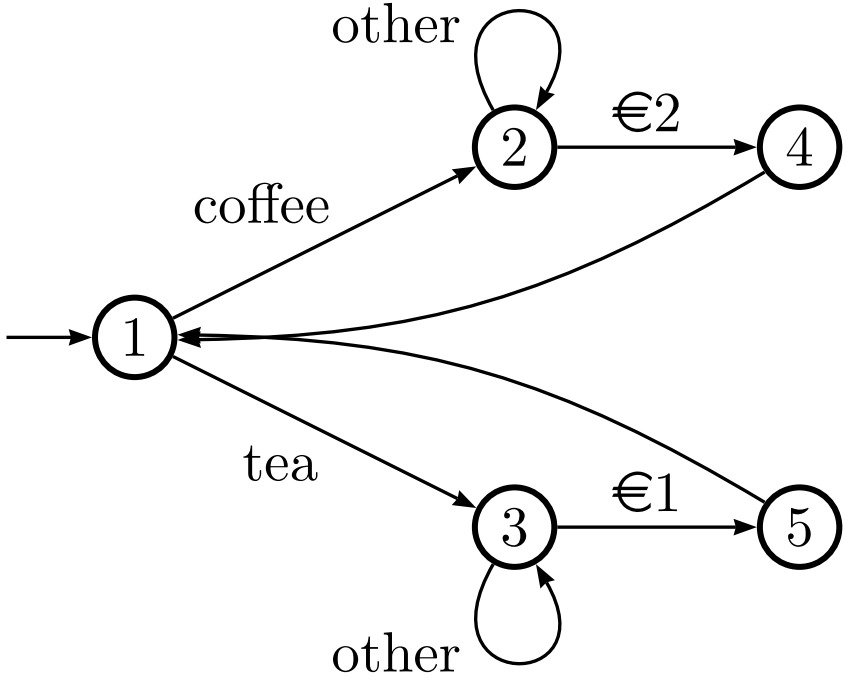


A tuple $(S, s_0, \rightarrow)$ where

- 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:

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

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:

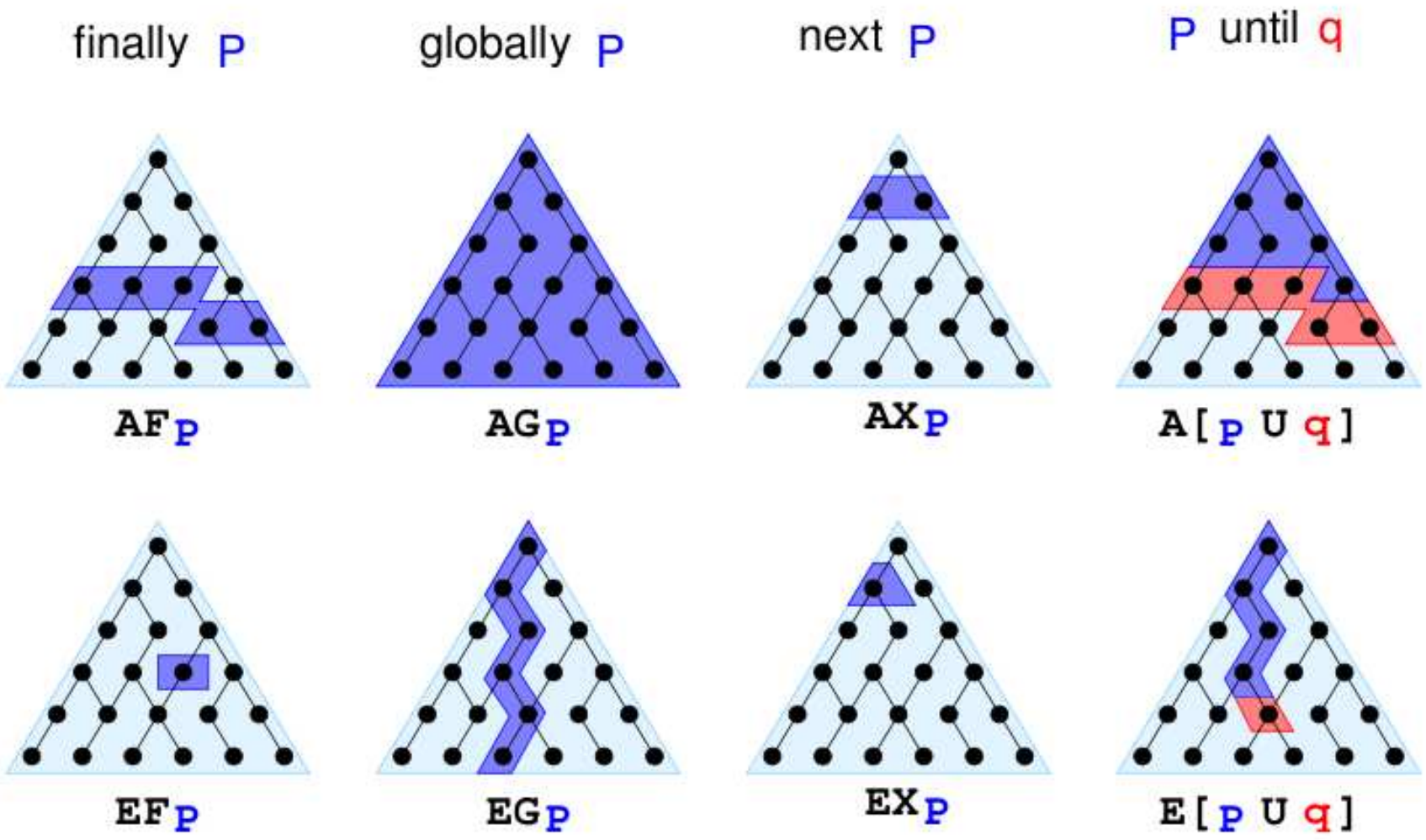
$\circ\varphi$ in the next state φ holds ($X\varphi$)

$\Box\varphi$ (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)



CTL, semantics

We fix a transition system M

$$
\begin{array}{lll}
s \models p & \text{iff} & p \in L(s) \\
s \models \varphi \wedge \varphi' & \text{iff} & s \models \varphi \text{ and } s \models \varphi' \\
s \models \varphi \vee \varphi' & \text{iff} & s \models \varphi \text{ or } s \models \varphi' \\
s \models \neg\varphi & \text{iff} & s \not\models \varphi \\
s \models A\psi & \text{iff} & \forall s \to s_1 \to \ldots : \quad s, s_1, s_2, \ldots \models \psi \\
s \models E\psi & \text{iff} & \exists s \to s_1 \to \ldots : \quad s, s_1, s_2, \ldots \models \psi
\end{array}
$$

$$
\begin{array}{lll}
s_0, s_1, \ldots \models \circ\varphi & \text{iff} & s_1 \models \varphi \\
s_0, s_1, \ldots \models \Box\varphi & \text{iff} & \forall i \geq 0 : s_i \models \varphi \\
s_0, s_1, \ldots \models \Diamond\varphi & \text{iff} & \exists i \geq 0 : s_i \models \varphi \\
s_0, s_1, \ldots \models \varphi U \varphi' & \text{iff} & \exists i \geq 0 : s_i \models \varphi' \text{ and } \forall j < i : s_j \models \varphi
\end{array}
$$

CTL

Model checking problem

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

It can be solved in polynomial time

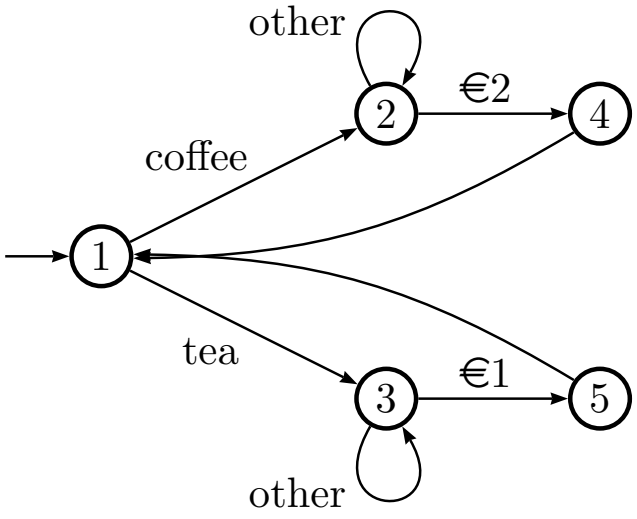
Computation Tree Logic (CTL)

Example:

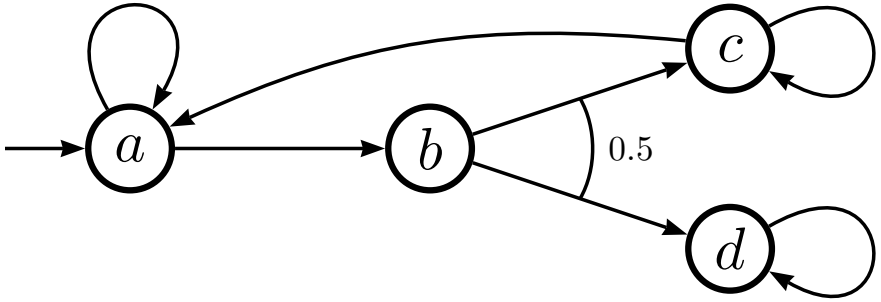
- $L(s_1) = \{start\}$ and empty for the other states
- "from any state the machine eventually goes back to its initial state"

$$A\Box\ A\Diamond\ start$$

- But $s_1 \not\models A\Box\ A\Diamond\ start$
  - $s_2 \not\models A\Diamond start$
  - $s_2, s_2, \ldots \not\models \Diamond start$



Markov Decision Process (MDP)



- 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)

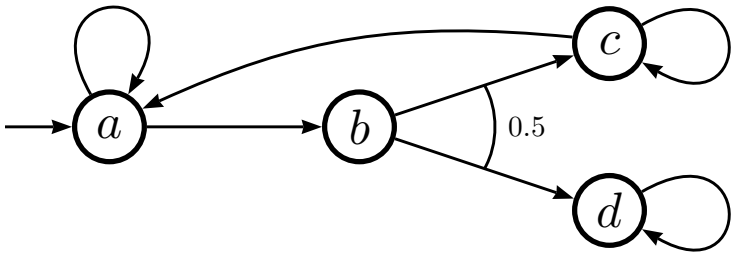
Syntax:

$$\varphi ::= p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi \mid \mathcal{P}_{\leq\lambda}\psi \mid \mathcal{P}_{\geq\lambda}\psi \quad \text{state formulas}$$
$$\psi ::= \circ\varphi \mid \Box\varphi \mid \Diamond\varphi \mid \varphi U \varphi' \quad \text{path formulas}$$

- A probabilistic extension of CTL
- We replaced the path quantifiers A, E by the operator $\mathcal{P}$
- $\mathcal{P}_{\leq\lambda}$ means: the probability that ψ holds is at most λ
- We could also use $\mathcal{P}_{<\lambda}, \mathcal{P}_{>\lambda}$

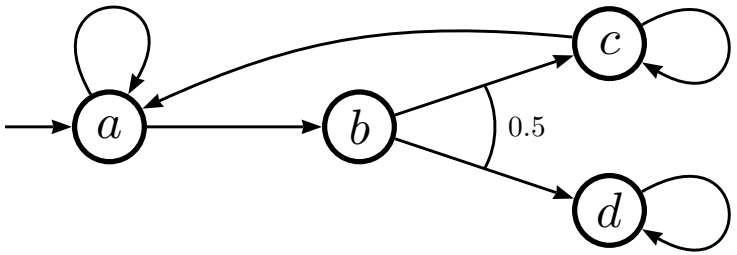
PCTL, semantics

- An MDP contains nondeterminism, thus we cannot generally define the probability of executions
- We use a scheduler σ assigning a single transition to each state
- We can now define a probability measure μ_σ on executions



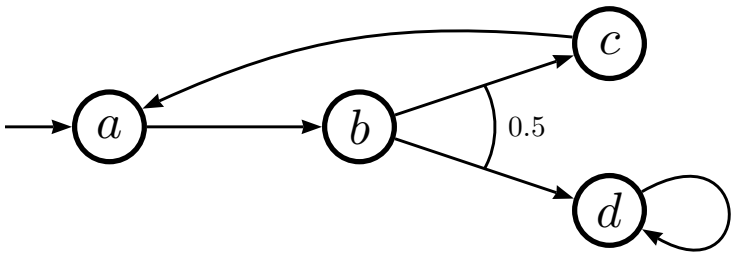
PCTL, semantics

- An MDP contains nondeterminism, thus we cannot generally define the probability of executions
- We use a scheduler σ assigning a single transition to each state
- We can now define a probability measure μ_σ on executions



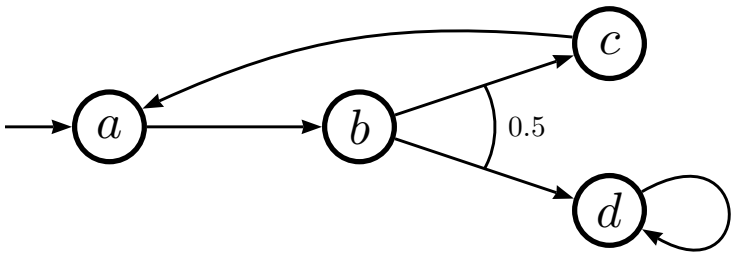
PCTL, semantics

- An MDP contains nondeterminism, thus we cannot generally define the probability of executions
- We use a scheduler σ assigning a single transition to each state
- We can now define a probability measure μ_σ on executions



PCTL, semantics

- An MDP contains nondeterminism, thus we cannot generally define the probability of executions
- We use a scheduler σ assigning a single transition to each state
- We can now define a probability measure μ_σ on executions



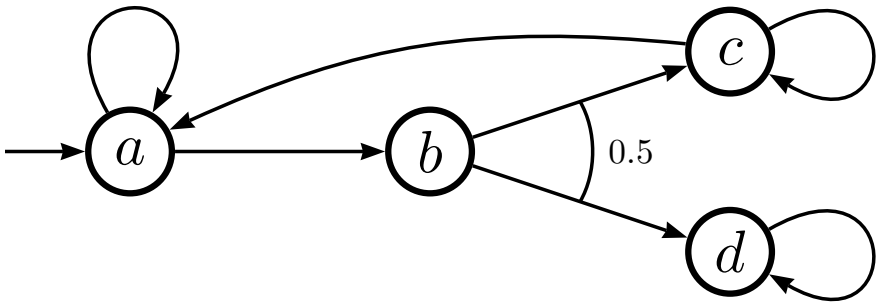
PCTL, semantics

- Semantics of PCTL:

$$s \models \mathcal{P}_{\leq\lambda}\psi \quad \text{iff} \quad \forall\sigma : \mu_\sigma(\{ss_1 \ldots \mid s, s_1, s_2, \ldots \models \psi\}) \leq \lambda$$
$$s \models \mathcal{P}_{\geq\lambda}\psi \quad \text{iff} \quad \forall\sigma : \mu_\sigma(\{ss_1 \ldots \mid s, s_1, s_2, \ldots \models \psi\}) \geq \lambda$$
$$\ldots$$

Exercise: is $A\varphi$ equivalent to $\mathcal{P}_{\geq 1}\varphi$?

PCTL, example



- $s_b \models \mathcal{P}_{\leq 0.5} \Diamond \; start$
- $s_b \not\models \mathcal{P}_{\geq 0.5} \Diamond \; start$

Model checking PCTL over MDPs

Model checking problem

Given an MDP M and a PCTL formula φ, decide whether $M \models \varphi$ (i.e. $s_0 \models \varphi$).

Model checking PCTL over MDPs

$$\varphi ::= p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi \mid \mathcal{P}_{\leq\lambda}\psi \mid \mathcal{P}_{\geq\lambda}\psi \quad \text{state formulas}$$
$$\psi ::= \circ\varphi \mid \Box\varphi \mid \Diamond\varphi \mid \varphi U \varphi' \quad \text{path formulas}$$

- We are going to recursively evaluate φ starting from the simplest subformulas
- At each stage we assume that we can already evaluate all subformulas on all states
- Base case $\varphi = p$: $\quad s \models p$ iff $p \in L(s)$
- Cases $\varphi \wedge \varphi', \varphi \vee \varphi', \neg\varphi$: trivial
- The interesting cases are $\mathcal{P}_{\leq\lambda}\psi, \mathcal{P}_{\geq\lambda}\psi$

Model checking PCTL over MDPs

$$\varphi ::= \mathcal{P}_{\leq\lambda}\psi \mid \mathcal{P}_{\geq\lambda}\psi$$
$$\psi ::= \circ\varphi \mid \Box\varphi \mid \Diamond\varphi \mid \varphi U \varphi'$$

- Define
  - $Pr_s^- \psi = \min_\sigma \mu_\sigma(\{ss_1 \ldots \mid s, s_1, s_2, \ldots \models \psi\})$
  - $Pr_s^+ \psi = \max_\sigma \mu_\sigma(\{ss_1 \ldots \mid s, s_1, s_2, \ldots \models \psi\})$
- Then
  - $s \models \mathcal{P}_{\leq\lambda}\psi$ iff $Pr_s^+ \psi \leq \lambda$
  - $s \models \mathcal{P}_{\geq\lambda}\psi$ iff $Pr_s^- \psi \geq \lambda$
- So the goal is to compute $Pr_s^- \psi, Pr_s^+ \psi$

Model checking PCTL over MDPs

$$Pr_s^+\psi, Pr_s^-\psi$$
$$\psi ::= \circ\varphi \mid \Box\varphi \mid \Diamond\varphi \mid \varphi U \varphi'$$

- Simplifications (exercise: prove them formally)
  - $\Diamond\varphi \equiv \mathbf{true} U \varphi$
  - $\Box\psi \equiv \neg\Diamond\neg\psi$
    - $\neg\Diamond\neg\psi$ 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 $\varphi U \varphi'$

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

- We assume that we have evaluated φ, φ' 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 Λ)
- 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:

$$Pr_s^- \varphi U \varphi' = \min_{s \to \mu} \Big[\sum_{t \in S'_p} \mu(t) Pr_t^- \varphi U \varphi' \; + \; \sum_{t \in S_d} \mu(t) \Big]$$

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 $s \to \mu$
- 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 chose the transition that maximizes the probability of getting to S_d:

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

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