Probabilistic Model Checking

Marta Kwiatkowska
Dave Parker

Oxford University Computing Laboratory

ESSLLI'10 Summer School, Copenhagen, August 2010

Part 2

Markov decision processes

Overview (Part 2)

- Markov decision processes (MDPs)
- Adversaries & probability spaces
- PCTL for MDPs
- PCTL model checking
- Further model checking (LTL, costs & rewards)
- Case study: Firewire root contention

Nondeterminism

- Some aspects of a system may not be probabilistic and should not be modelled probabilistically; for example:

- Concurrency – scheduling of parallel components
  - e.g. randomised distributed algorithms – multiple probabilistic processes operating asynchronously

- Underspecification – unknown model parameters
  - e.g. a probabilistic communication protocol designed for message propagation delays of between d_{min} and d_{max}

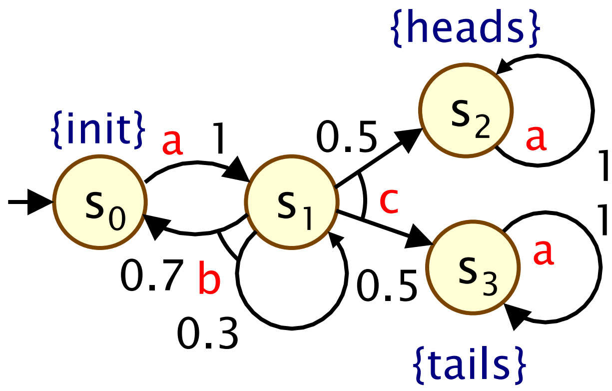
- Unknown environments
  - e.g. probabilistic security protocols – unknown adversary

Markov decision processes

- Markov decision processes (MDPs)
  - extension of DTMCs which allow nondeterministic choice

- Like DTMCs:
  - discrete set of states representing possible configurations of the system being modelled
  - transitions between states occur in discrete time-steps

- Probabilities and nondeterminism
  - in each state, a nondeterministic choice between several discrete probability distributions over successor states



Markov decision processes

- Formally, an MDP M is a tuple (S,s_{init},**Steps**,L) where:
  - S is a finite set of states ("state space")
  - $s_{init} \in S$ is the initial state
  - **Steps** : $S \rightarrow 2^{Act \times Dist(S)}$ is the transition probability function

    where Act is a set of actions and Dist(S) is the set of discrete probability distributions over the set S
  - $L : S \rightarrow 2^{AP}$ is a labelling with atomic propositions

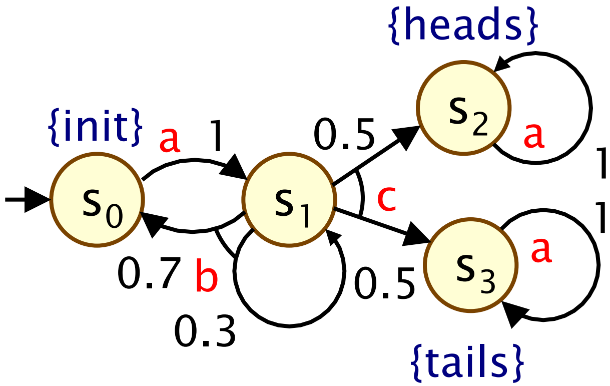
- For example:

  Steps(s0) = { (a,u) } where u is a "distribution" describing the probability of getting to a state if we take an a-transition on s1:

  u(s0)=0, u(s1)=1, u(s2)=0, u(s3)=0

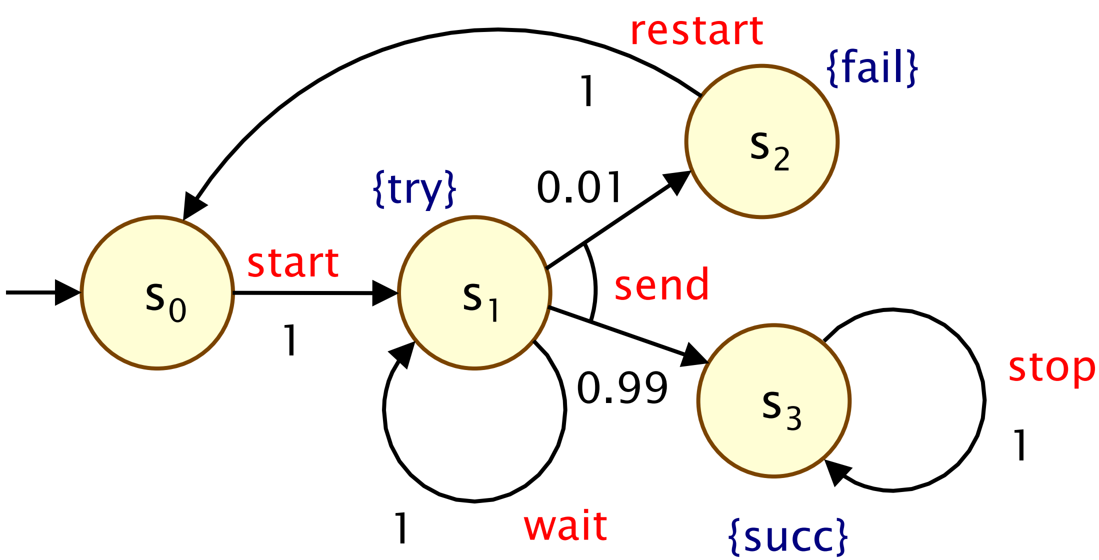
  For state s1, we have

  Steps(s1) = { (b,v), (c,w) }
  where v and w are distributions :
  v(s0) = 0.7 , v(s1) = 0.3, v(s2) = 0. v(s3)=0
  w(s0) = 0, w(s1) = 0, w(s2) = 0.5, w(s3) = 0.5

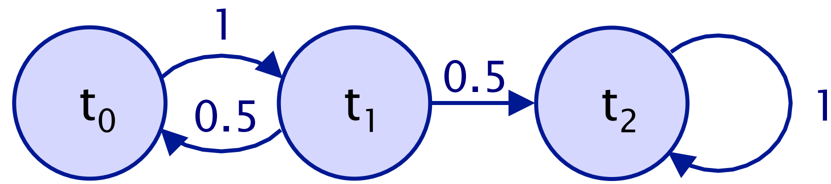


Simple MDP example

- Modification of the simple DTMC communication protocol
  - after one step, process starts trying to send a message
  - then, a nondeterministic choice between: (a) waiting a step because the channel is unready; (b) sending the message
  - if the latter, with probability 0.99 send successfully and stop
  - and with probability 0.01, message sending fails, restart

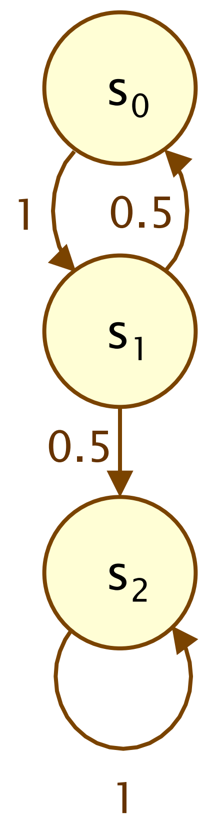


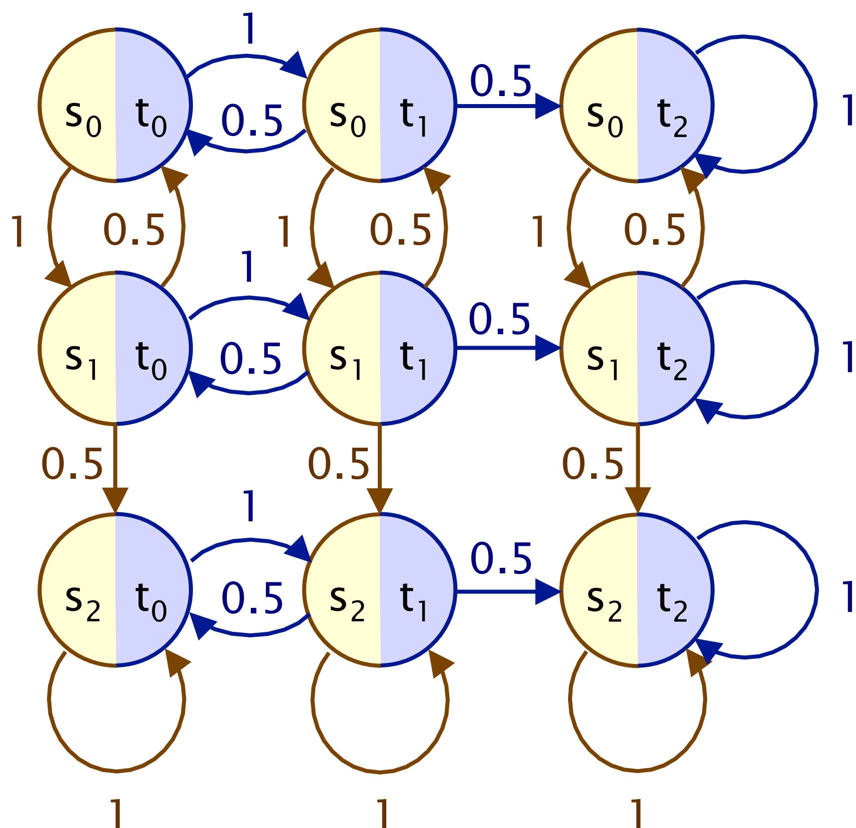
Example – Parallel composition



Asynchronous parallel composition of two 3-state DTMCs

Action labels omitted here





at every state above there is a non-deterministic choice between taking a blue or a brown transition.

Paths and probabilities

- A (finite or infinite) path through an MDP
  - is a sequence of states and action/distribution pairs
  - e.g. $s_0(a_0,\mu_0)s_1(a_1,\mu_1)s_2\ldots$
  - such that $(a_i,\mu_i) \in$ **Steps**(s_i) and $\mu_i(s_{i+1}) > 0$ for all $i \geq 0$
  - represents an execution (i.e. one possible behaviour) of the system which the MDP is modelling
  - note that a path resolves both types of choices: nondeterministic and probabilistic
- To consider the probability of some behaviour of the MDP
  - first need to resolve the nondeterministic choices
  - …which results in a DTMC
  - …for which we can define a probability measure over paths

Adversaries

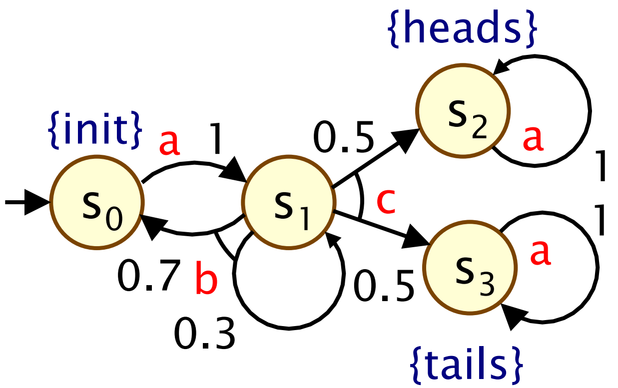
- An adversary resolves nondeterministic choice in an MDP
  - also known as “schedulers”, “strategies” or “policies”

- Formally:
  - an adversary A of an MDP M is a function mapping every finite path $\omega = s_0(a_1,\mu_1)s_1 \ldots s_n$ to an element of **Steps**(s_n)

- For each A can define a probability measure Pr^A_s over paths
  - constructed through an infinite state DTMC ($Path^A_{fin}(s)$,s,$\mathbf{P}^A_s$)
  - states of the DTMC are the finite paths of A starting in state s
  - initial state is s (the path starting in s of length 0)
  - $\mathbf{P}^A_s(\omega,\omega') = \mu(s)$ if $\omega' = \omega(a, \mu)s$ and $A(\omega) = (a,\mu)$
  - $\mathbf{P}^A_s(\omega,\omega') = 0$ otherwise

Adversaries – Examples

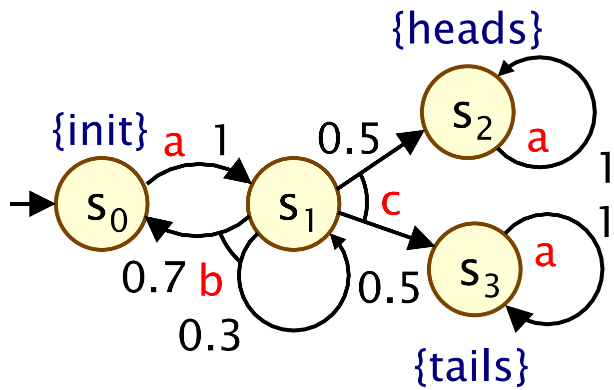
- Consider the simple MDP below
  - note that s_1 is the only state for which $|\mathbf{Steps}(s)| > 1$
  - i.e. s_1 is the only state for which an adversary makes a choice
  - let μ_b and μ_c denote the probability distributions associated with actions b and c in state s_1

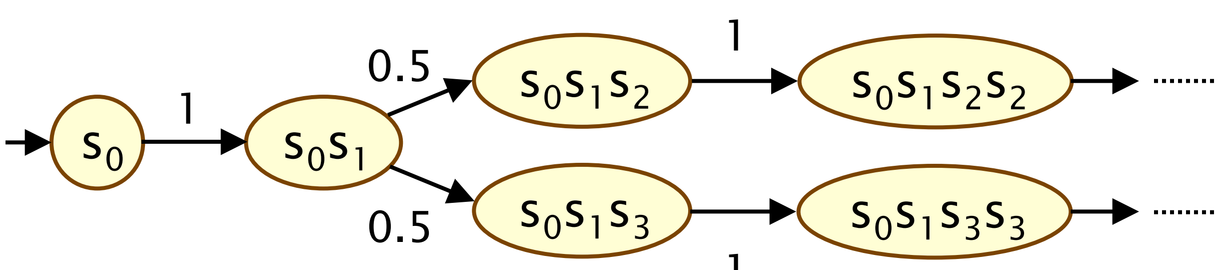


- Adversary A_1
  - picks action c the first time
  - $A_1(s_0s_1)=(c,\mu_c)$
- Adversary A_2
  - picks action b the first time, then c
  - $A_2(s_0s_1)=(b,\mu_b)$, $A_2(s_0s_1s_1)=(c,\mu_c)$, $A_2(s_0s_1s_0s_1)=(c,\mu_c)$

Adversaries – Examples

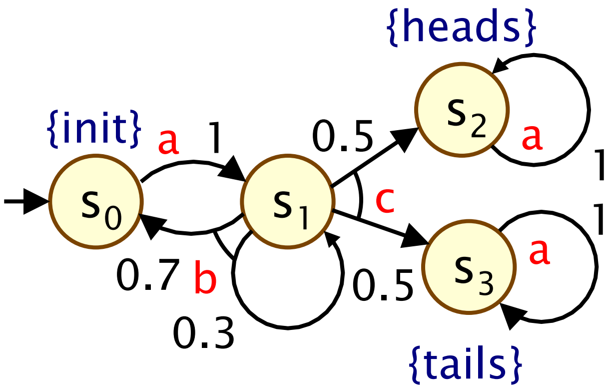
- Fragment of DTMC for adversary A_1
  - A_1 picks action c the first time

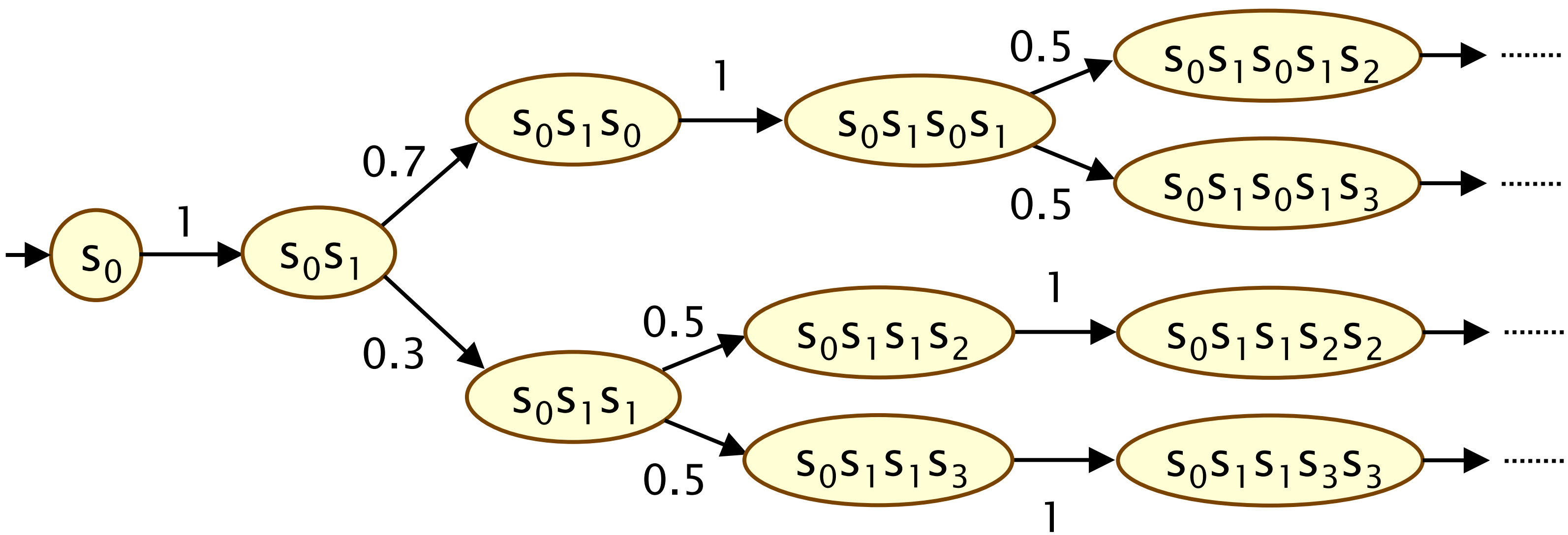




Adversaries – Examples

- Fragment of DTMC for adversary A_2
  - A_2 picks action b, then c

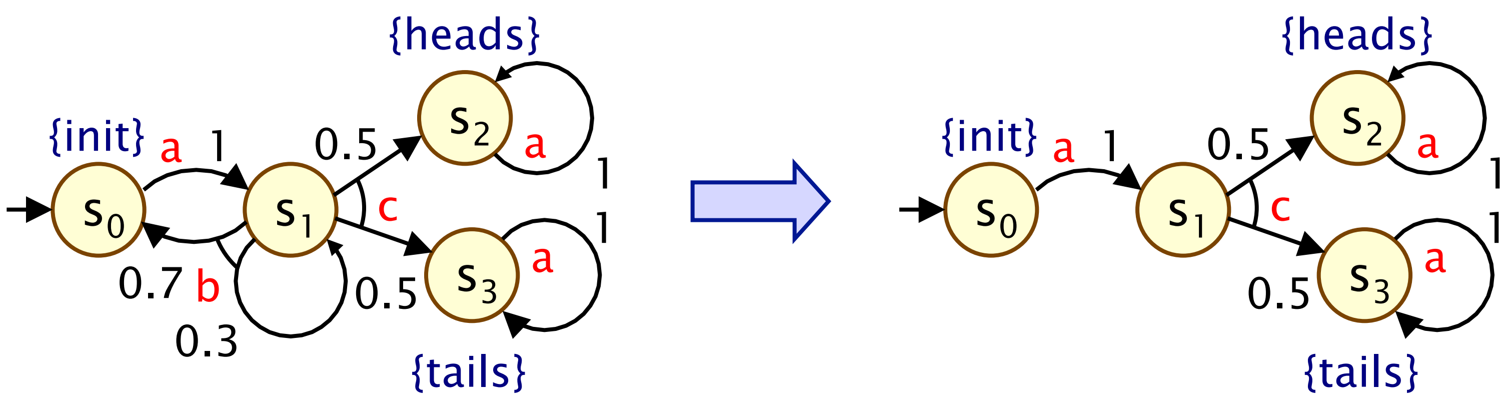




Memoryless adversaries

- Memoryless adversaries always pick same choice in a state
  - also known as: positional, Markov, simple
  - formally, for adversary A:
  - $A(s_0(a_1,\mu_1)s_1 \ldots s_n)$ depends only on s_n
  - resulting DTMC can be mapped to a |S|-state DTMC

- From previous example:
  - adversary A_1 (picks c in s_1) is memoryless, A_2 is not



Overview (Part 2)

- Markov decision processes (MDPs)
- Adversaries & probability spaces
- PCTL for MDPs
- PCTL model checking
- Further model checking (LTL, costs & rewards)
- Case study: Firewire root contention

PCTL for MDPs

- The temporal logic PCTL can also describe MDP properties
- Identical syntax to the DTMC case:

ψ is true with probability $\sim p$

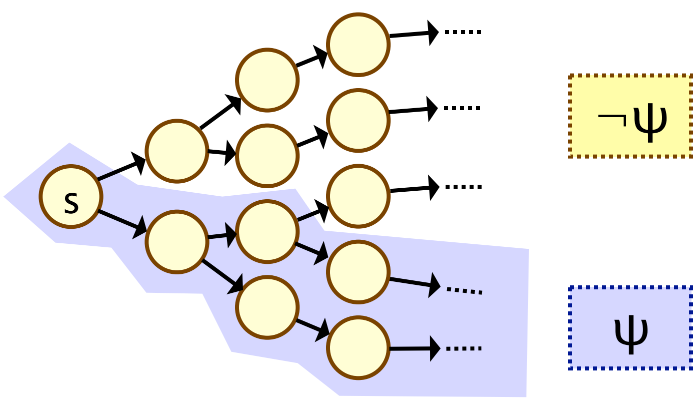
  - ϕ ::= true | a | $\phi \wedge \phi$ | $\neg\phi$ | $P_{\sim p}$ [ψ] (state formulas)
  - ψ ::= $X\ \phi$ | $\phi\ U^{\leq k}\ \phi$ | $\phi\ U\ \phi$ (path formulas)

"next" "bounded until" "until"

- Semantics are also the same as DTMCs for:
  - atomic propositions, logical operators, path formulas

PCTL semantics for MDPs

- Semantics of the probabilistic operator P
  - can only define probabilities for a specific adversary A
  - $s \vDash P_{\sim p} [\ \psi\ ]$ means "the probability, from state s, that ψ is true for an outgoing path satisfies ~p for all adversaries A"
  - formally $s \vDash P_{\sim p} [\ \psi\ ] \Leftrightarrow Prob^A(s, \psi) \sim p$ for all adversaries A
  - where $Prob^A(s, \psi) = Pr^A_s \{\ \omega \in Path^A(s) \mid \omega \vDash \psi\ \}$



$Prob^A(s, \psi) \sim p$

Minimum and maximum probabilities

- Letting:
  - $p_{max}(s, \psi) = \sup_A \text{Prob}^A(s, \psi)$
  - $p_{min}(s, \psi) = \inf_A \text{Prob}^A(s, \psi)$
- We have:
  - if $\sim \in \{\geq,>\}$, then $s \vDash P_{\sim p} [\ \psi\ ] \quad \Leftrightarrow \quad p_{min}(s, \psi) \sim p$
  - if $\sim \in \{<,\leq\}$, then $s \vDash P_{\sim p} [\ \psi\ ] \quad \Leftrightarrow \quad p_{max}(s, \psi) \sim p$
- Model checking $P_{\sim p}[\ \psi\ ]$ reduces to the computation over all adversaries of either:
  - the minimum probability of ψ holding
  - the maximum probability of ψ holding
- Crucial result for model checking PCTL on MDPs
  - memoryless adversaries suffice, i.e. there are always memoryless adversaries A_{min} and A_{max} for which:
  - $\text{Prob}^{Amin}(s, \psi) = p_{min}(s, \psi)$ and $\text{Prob}^{Amax}(s, \psi) = p_{max}(s, \psi)$

Quantitative properties

- For PCTL properties with P as the outermost operator
  - quantitative form (two types): $\mathrm{Pmin}_{=?}$ [ψ] and $\mathrm{Pmax}_{=?}$ [ψ]
  - i.e. "what is the minimum/maximum probability (over all adversaries) that path formula ψ is true?"
  - corresponds to an analysis of best-case or worst-case behaviour of the system
  - model checking is no harder since compute the values of $p_{min}(s, \psi)$ or $p_{max}(s, \psi)$ anyway
  - useful to spot patterns/trends

- Example: CSMA/CD protocol
  - "min/max probability
    that a message is sent
    within the deadline"

Some real PCTL examples

- Byzantine agreement protocol
  - $\text{Pmin}_{=?}$ [F (agreement $\wedge$ rounds$\leq$2)]
  - "what is the minimum probability that agreement is reached within two rounds?"

- CSMA/CD communication protocol
  - $\text{Pmax}_{=?}$ [F collisions=k]
  - "what is the maximum probability of k collisions?"

- Self-stabilisation protocols
  - $\text{Pmin}_{=?}$ [$F^{\leq t}$ stable]
  - "what is the minimum probability of reaching a stable state within k steps?"

Overview (Part 2)

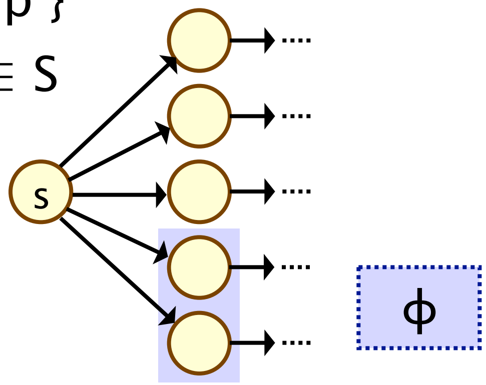
- Markov decision processes (MDPs)
- Adversaries & probability spaces
- PCTL for MDPs
- PCTL model checking
- Further model checking (LTL, costs & rewards)
- Case study: Firewire root contention

PCTL model checking for MDPs

- Algorithm for PCTL model checking [BdA95]
  - inputs: MDP $M=(S,s_{init},\mathbf{Steps},L)$, PCTL formula ϕ
  - output: $Sat(\phi) = \{ s \in S \mid s \vDash \phi \}$ = set of states satisfying ϕ
- Basic algorithm same as PCTL model checking for DTMCs
  - proceeds by induction on parse tree of ϕ
  - non-probabilistic operators (true, a, $\neg$, $\wedge$) straightforward
- Only need to consider $P_{\sim p}$ [ψ] formulas
  - reduces to computation of $p_{min}(s, \psi)$ or $p_{max}(s, \psi)$ for all $s \in S$
  - dependent on whether $\sim \in \{\geq,>\}$ or $\sim \in \{<,\leq\}$
  - these slides cover the case $p_{min}(s, \phi_1 \text{ U } \phi_2)$, i.e. $\sim \in \{\geq,>\}$
  - case for maximum probabilities is very similar
  - next (X ϕ) and bounded until (ϕ_1 $U^{\leq k}$ ϕ_2) are straightforward extensions of the DTMC case

PCTL next for MDPs

- Computation of probabilities for PCTL next operator
- Consider case of minimum probabilities…
  - $\text{Sat}(P_{\sim p}[\ X\ \phi\ ]) = \{\ s \in S \mid p_{min}(s, X\ \phi) \sim p\ \}$
  - need to compute $p_{min}(s, X\ \phi)$ for all $s \in S$
- Recall in the DTMC case
  - sum outgoing probabilities for transitions to ϕ–states
  - $\text{Prob}(s, X\ \phi) = \Sigma_{s' \in \text{Sat}(\phi)}\ \mathbf{P}(s,s')$



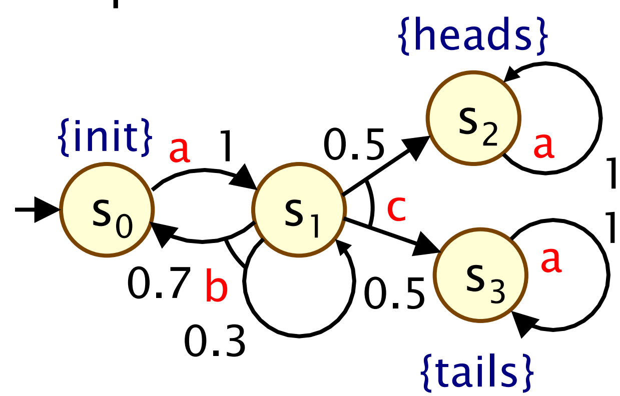
- For MDPs, perform computation for each distribution available in s and then take minimum:
  - $p_{min}(s, X\ \phi) = \min\{\ \Sigma_{s' \in \text{Sat}(\phi)}\ \mu(s') \mid (a,\mu) \in \text{Steps}(s)\ \}$

- Maximum probabilities case is analogous

PCTL next – Example

- Model check: $P_{\geq 0.5}$ [X heads]
  - lower probability bound so minimum probabilities required
  - Sat (heads)= $\{s_2\}$
  - e.g. $p_{min}(s_1, \text{X heads}) = \min(0, 0.5) = 0$
  - can do all at once with matrix-vector multiplication:

$$\text{Steps} \cdot \underline{\text{heads}} = \left[\begin{array}{cccc} 0 & 1 & 0 & 0 \\ \hline 0.7 & 0.3 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 \\ \hline 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \left[\begin{array}{c} 0 \\ \hline 0 \\ 0.5 \\ \hline 1 \\ \hline 0 \end{array}\right]$$



- Extracting the minimum for each state yields
  - $\underline{p}_{min}(\text{X heads}) = [0, 0, 1, 0]$
  - Sat($P_{\geq 0.5}$ [X heads]) = $\{s_2\}$

PCTL until for MDPs

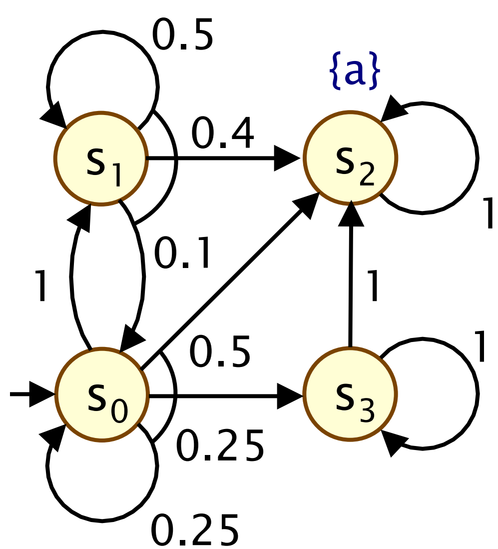
- Computation of probabilities $p_{min}(s, \phi_1 \text{ U } \phi_2)$ for all $s \in S$
- First identify all states where the probability is 1 or 0
  - "precomputation" algorithms, yielding sets S^{yes}, S^{no}
- Then compute (min) probabilities for remaining states ($S^?$)
  - either: solve linear programming problem
  - or: approximate with an iterative solution method
  - or: use policy iteration

Example:

$P_{\geq p}$ [F a]

$\equiv$

$P_{\geq p}$ [true U a]

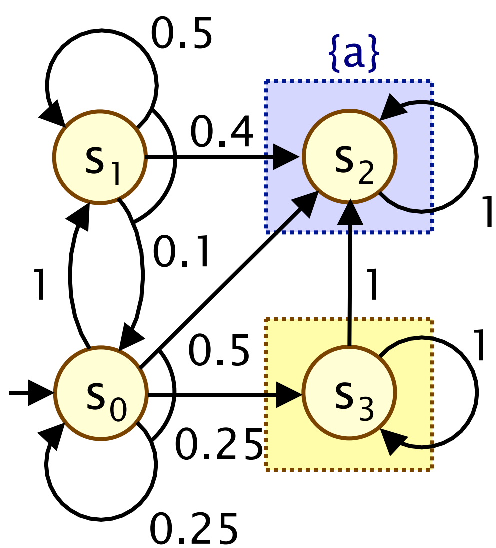


PCTL until – Precomputation

- Identify all states where $p_{min}(s, \phi_1 \text{ U } \phi_2)$ is 1 or 0
  - S^{yes} = Sat($P_{\geq 1}$ [ϕ_1 U ϕ_2]), S^{no} = Sat($\neg$ $P_{>0}$ [ϕ_1 U ϕ_2])
- Two graph-based precomputation algorithms:
  - algorithm Prob1A computes S^{yes}
    - for all adversaries the probability of satisfying ϕ_1 U ϕ_2 is 1
  - algorithm Prob0E computes S^{no}
    - there exists an adversary for which the probability is 0

Example:

$P_{\geq p}$ [F a]



S^{yes} = Sat($P_{\geq 1}$ [F a])

S^{no} = Sat($\neg P_{>0}$ [F a])

Calculating pmin(s, φ_1 **U** φ_2)

pmin(s, φ) = the **minimum** probability for having executions starting from 2 satisfying φ, regardless the adversary. To calculate pmin(s, φ_1 **U** φ_2) :

1. Calculate first the sets S^{yes} and S^{no}.
   - S^{yes} = { s | P(s, φ_1 U φ_2) ≥ 1, for **all** adversaries } → with algorithm prob1A.
   - S^{no} = { s | P(s, φ_1 U φ_2) ≤ 0, for **some** adversary } → with algorithm prob0E.
2. For any state s in S^{yes}, we then know that pmin(s, φ_1 **U** φ_2) ≥ 1.
3. For any state s in S^{no} we have pmin(s, φ_1 **U** φ_2) ≤ 0.
4. We then proceed with calculating the pmin for the remaining states (which are not in S^{yes} nor S^{no}).

Algorithm **Prob0E**

- The algorithm below first calculates the set R of all states s satisfying E[s, φ_1 U φ_2], regardless the adversary. So, for any state in R, and for any adversary Prob(s, φ_1 U φ_2) > 0.
- S^{no} is just complement S/R.
- Sat(φ_1) and Sat(φ_2) in the paremeters are the set of states on which φ_1 and φ_2 respectively hold.

```
PROB0E(Sat(φ1), Sat(φ2))
R := Sat(φ2)
done := false
while (done = false)
    R' := R ∪ {s ∈ Sat(φ1) | ∀μ ∈ Steps(s) . ∃s' ∈ R . μ(s') > 0}
    if (R' = R) then done := true
    R := R'
endwhile
return S\R
```

Prob1A

- Calculate first the set F of states from where we have a path passing exclusively through states satisfying $\varphi_1 \wedge \neg\varphi_2$ and ends in S^{no}, under **some** adversary.
  By definition this F also includes S^{no}.
- So any state s in F has Prob(s, φ_1 U φ_2) < 1, for some adversary. In other words pmin(s, φ_1 U φ_2) < 1.
- P^{yes} in the the complement S/F.

- **Note**: for the calculation of pmin(s, φ_1 **U** φ_2), we can also just take S^{yes} = Sat(φ_2). The calculation would still works, though it would take more steps to get its final results.
-

Method 1 – Linear programming

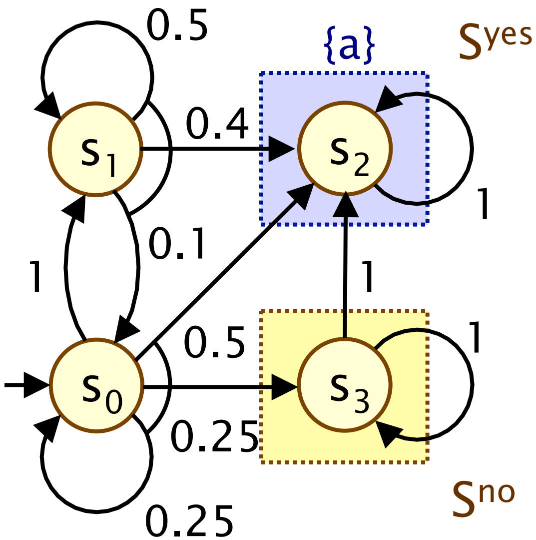
- Probabilities $p_{min}(s, \phi_1 \text{ U } \phi_2)$ for remaining states in the set $S^? = S \setminus (S^{yes} \cup S^{no})$ can be obtained as the unique solution of the following linear programming (LP) problem:

$$\text{maximize } \sum_{s \in S^?} x_s \text{ subject to the constraints:}$$
$$x_s \leq \sum_{s' \in S^?} \mu(s') \cdot x_{s'} + \sum_{s' \in S^{yes}} \mu(s')$$
$$\text{for all } s \in S^? \text{ and for all } (a, \mu) \in Steps(s)$$

- Simple case of a more general problem known as the stochastic shortest path problem [BT91]

- This can be solved with standard techniques
  - e.g. Simplex, ellipsoid method, branch-and-cut

Example – PCTL until (LP)



Let $x_i = p_{min}(s_i, F\ a)$

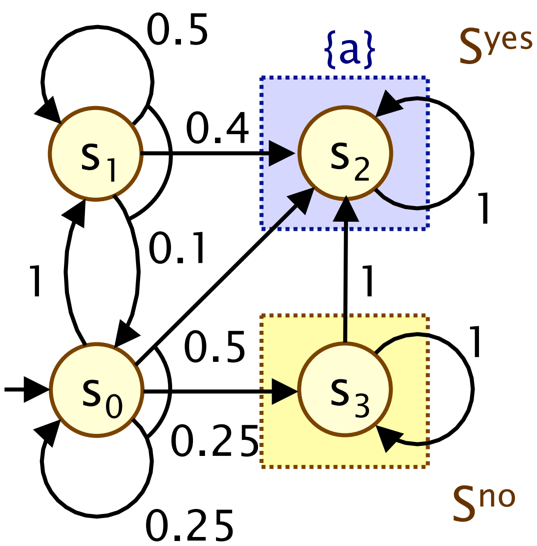
S^{yes}: $x_2=1$, S^{no}: $x_3=0$

For $S^? = \{x_0, x_1\}$:

Maximise x_0+x_1 subject to constraints:

- $x_0 \leq x_1$
- $x_0 \leq 0.25 \cdot x_0 + 0.5$
- $x_1 \leq 0.1 \cdot x_0 + 0.5 \cdot x_1 + 0.4$

Example – PCTL until (LP)



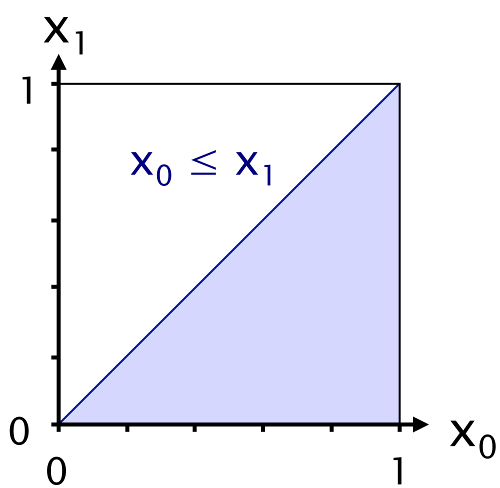
Let $x_i = p_{min}(s_i, F\ a)$

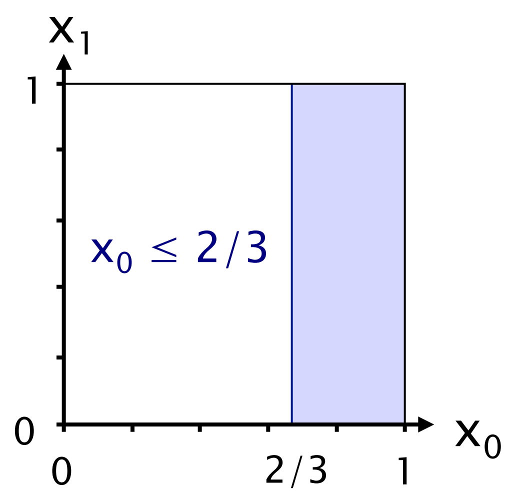
S^{yes}: $x_2=1$, S^{no}: $x_3=0$

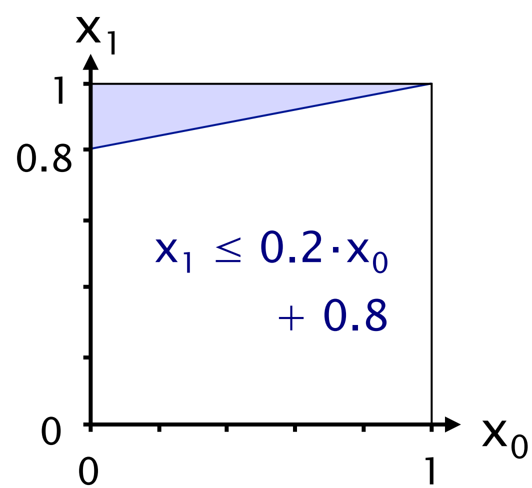
For $S^? = \{x_0, x_1\}$:

Maximise x_0+x_1 subject to constraints:

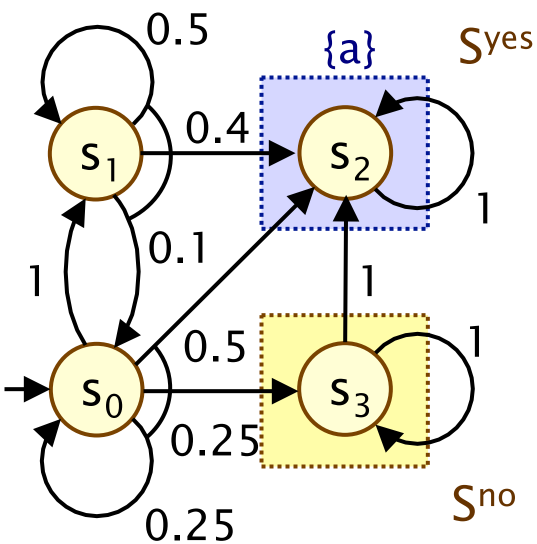
- $x_0 \leq x_1$
- $x_0 \leq 2/3$
- $x_1 \leq 0.2 \cdot x_0 + 0.8$







Example – PCTL until (LP)



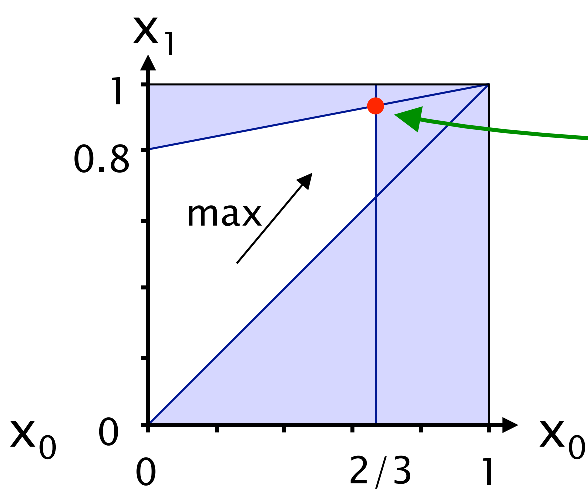
Let $x_i = p_{min}(s_i, F\ a)$

S^{yes}: $x_2=1$, S^{no}: $x_3=0$

For $S^? = \{x_0, x_1\}$:

Maximise x_0+x_1 subject to constraints:

- $x_0 \leq x_1$
- $x_0 \leq 2/3$
- $x_1 \leq 0.2 \cdot x_0 + 0.8$



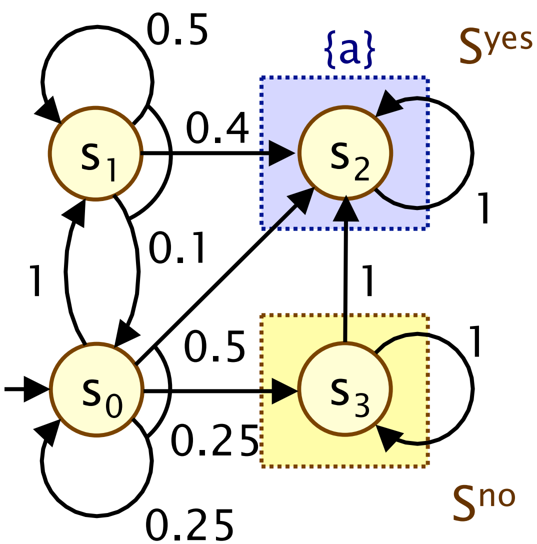
Solution:

(x_0, x_1)

$=$

$(2/3, 14/15)$

Example – PCTL until (LP)



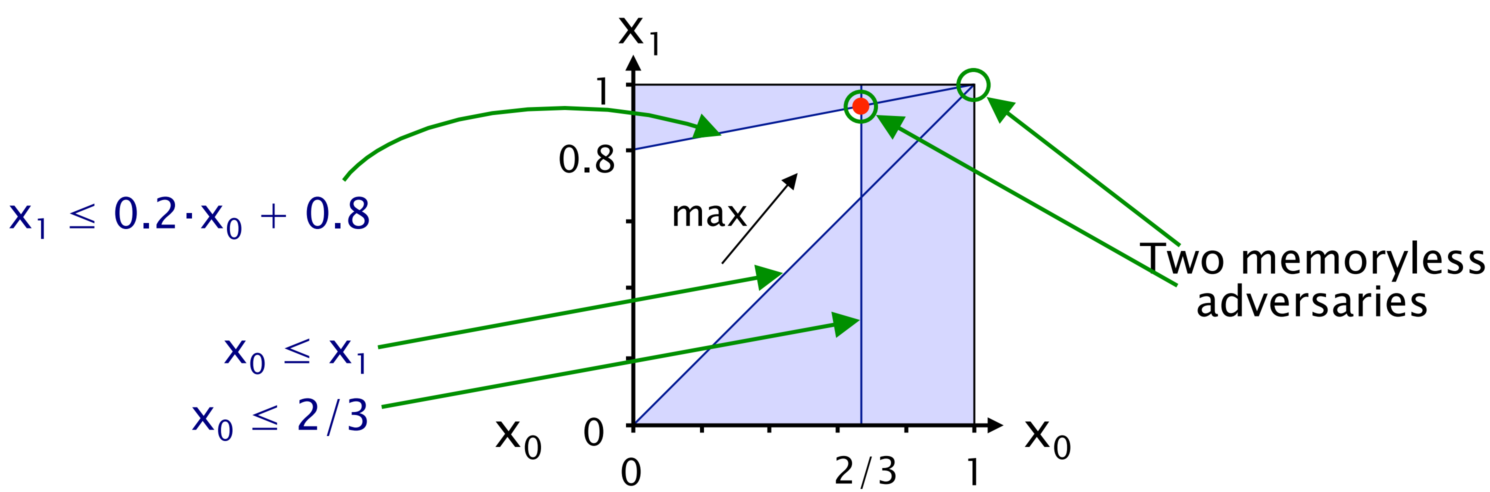
Let $x_i = p_{min}(s_i, F\ a)$

S^{yes}: $x_2=1$, S^{no}: $x_3=0$

For $S^? = \{x_0, x_1\}$:

Maximise x_0+x_1 subject to constraints:

- $x_0 \leq x_1$
- $x_0 \leq 2/3$
- $x_1 \leq 0.2 \cdot x_0 + 0.8$



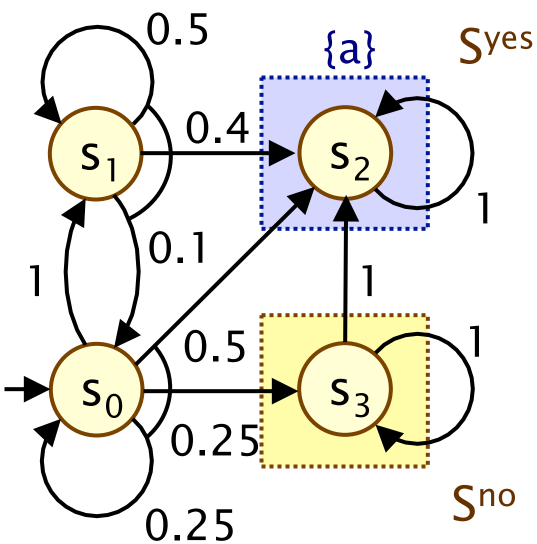
Method 2 – Value iteration

- For probabilities $p_{min}(s, \phi_1 \text{ U } \phi_2)$ it can be shown that:
  - $p_{min}(s, \phi_1 \text{ U } \phi_2) = \lim_{n\to\infty} x_s^{(n)}$ where:

$$x_s^{(n)} = \begin{cases} 1 & \text{if } s \in S^{yes} \\ 0 & \text{if } s \in S^{no} \\ 0 & \text{if } s \in S^{?} \text{ and } n = 0 \\ \min_{(a,\mu)\in Steps(s)} \left(\sum_{s'\in S} \mu(s') \cdot x_{s'}^{(n-1)} \right) & \text{if } s \in S^{?} \text{ and } n > 0 \end{cases}$$

- This forms the basis for an (approximate) iterative solution
  - iterations terminated when solution converges sufficiently

Example – PCTL until (value iteration)



Compute: $p_{min}(s_i, F\ a)$

$S^{yes} = \{x_2\}$, $S^{no} = \{x_3\}$, $S^{?} = \{x_0, x_1\}$

$[\ x_0^{(n)}, x_1^{(n)}, x_2^{(n)}, x_3^{(n)}\ ]$

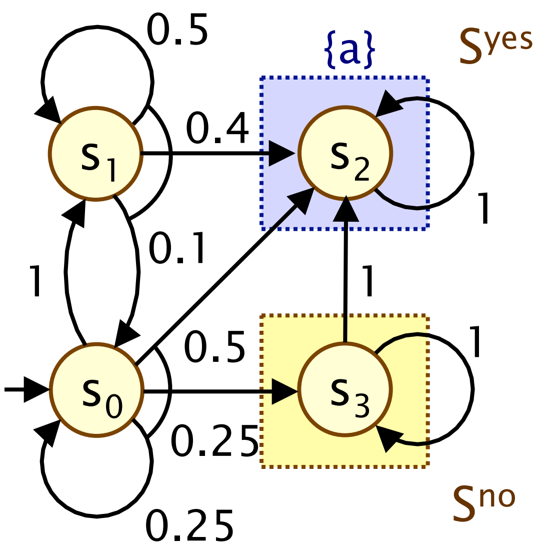
n=0: $[\ 0, 0, 1, 0\ ]$

n=1: $[\ \min(0, 0.25 \cdot 0 + 0.5),$
$0.1 \cdot 0 + 0.5 \cdot 0 + 0.4, 1, 0\ ]$
$= [\ 0, 0.4, 1, 0\ ]$

n=2: $[\ \min(0.4, 0.25 \cdot 0 + 0.5),$
$0.1 \cdot 0 + 0.5 \cdot 0.4 + 0.4, 1, 0\ ]$
$= [\ 0.4, 0.6, 1, 0\ ]$

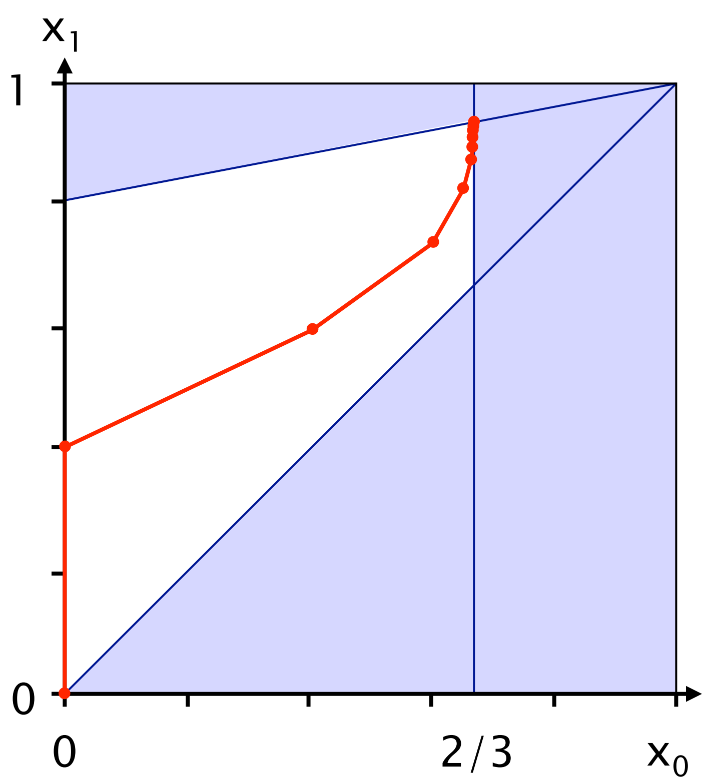
n=3: …

Example – PCTL until (value iteration)



	$[\, x_0^{(n)}, x_1^{(n)}, x_2^{(n)}, x_3^{(n)} \,]$
n=0:	[0.000000, 0.000000, 1, 0]
n=1:	[0.000000, 0.400000, 1, 0]
n=2:	[0.400000, 0.600000, 1, 0]
n=3:	[0.600000, 0.740000, 1, 0]
n=4:	[0.650000, 0.830000, 1, 0]
n=5:	[0.662500, 0.880000, 1, 0]
n=6:	[0.665625, 0.906250, 1, 0]
n=7:	[0.666406, 0.919688, 1, 0]
n=8:	[0.666602, 0.926484, 1, 0]
n=9:	[0.666650, 0.929902, 1, 0]
	…
n=20:	[0.666667, 0.933332, 1, 0]
n=21:	[0.666667, 0.933332, 1, 0]
	≈ [2/3, 14/15, 1, 0]

Example – Value iteration + LP



$[\ x_0^{(n)}, x_1^{(n)}, x_2^{(n)}, x_3^{(n)}\ ]$

n=0:	[0.000000, 0.000000, 1, 0]
n=1:	[0.000000, 0.400000, 1, 0]
n=2:	[0.400000, 0.600000, 1, 0]
n=3:	[0.600000, 0.740000, 1, 0]
n=4:	[0.650000, 0.830000, 1, 0]
n=5:	[0.662500, 0.880000, 1, 0]
n=6:	[0.665625, 0.906250, 1, 0]
n=7:	[0.666406, 0.919688, 1, 0]
n=8:	[0.666602, 0.926484, 1, 0]
n=9:	[0.666650, 0.929902, 1, 0]
	…
n=20:	[0.666667, 0.933332, 1, 0]
n=21:	[0.666667, 0.933332, 1, 0]
	≈ [2/3, 14/15, 1, 0]

PCTL until for MDPs – Prob0A

- Maximum probabilities 0
  - $S^{no} = \{ s \in S \mid p_{max}(s, \phi_1 \text{ U } \phi_2)=0 \}$

$\textsc{Prob0A}(Sat(\phi_1), Sat(\phi_2))$
1. $R := Sat(\phi_2)$ 2. $done :=$ **false**   3. **while** $(done =$ **false**$)$ 4. $\quad R' := R \cup \{s \in Sat(\phi_1) \mid \exists \mu \in Steps(s) \,.\, \exists s' \in R \,.\, \mu(s') > 0\}$ 5. $\quad$ **if** $(R' = R)$ **then** $done :=$ **true**   6. $\quad R := R'$ 7. **endwhile**   8. **return** $S \backslash R$

PCTL until for MDPs – Prob1E

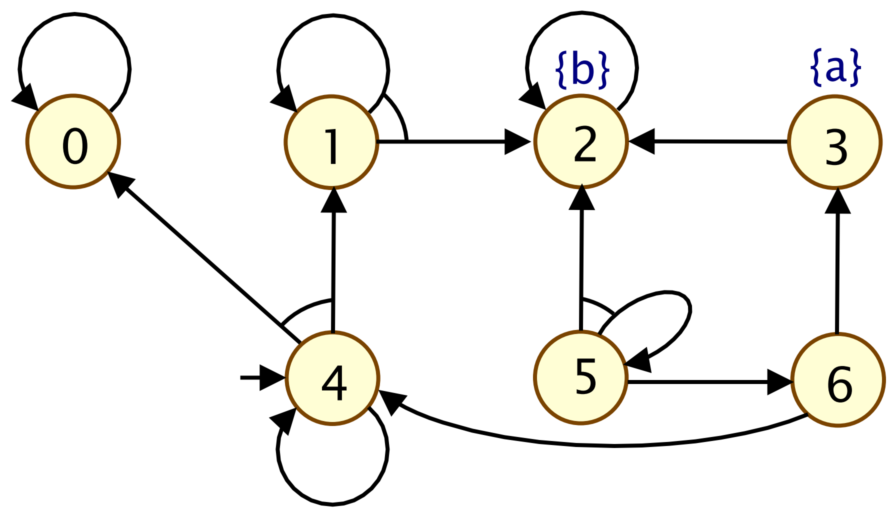
- Maximum probabilities 1
  - $S^{yes} = \{ s \in S \mid p_{max}(s, \phi_1 \ U \ \phi_2)=1 \} = Sat(\neg P_{<1} [\phi_1 \ U \ \phi_2])$
- Prob1E algorithm (see next slide)
  - two nested loops (double fixed point)
  - result, stored in R, will be S^{yes}; initially R is S
  - iteratively remove (some) states u with $p_{max}(u, \phi_1 \ U \ \phi_2)<1$
    - i.e. remove (some) states for which, under no adversary σ, is $Prob^{\sigma}(s, \phi_1 \ U \ \phi_2)=1$
  - done by inner loop which computes subset R' of R
    - R' contains ϕ_1-states with a probability distribution for which all transitions stay within R and at least one eventually reaches ϕ_2
  - note: after first iteration, R contains:
    - $\{ s \mid Prob^A(s, \phi_1 \ U \ \phi_2)>0$ for some A $\}$
    - essentially: execution of Prob0A and removal of S^{no} from R

Prob1E – Example

- S^{yes} = { s∈S | p_{max}(s, ¬a U b)=1 }

- R = { 0, 1, 2, 3, 4 ,5 6 }
  - R' = {2} ; R' = {1, 2, 5} ; R' = {1, 2, 4, 5} ; R' = {1, 2, 4, 5, 6}
- R = { 1, 2, 4, 5, 6 }
  - R' = {2} ; R' = {1, 2, 5}
- R = { 1, 2, 5 }
  - R' = {2} ; R' = {1, 2, 5}
- R = { 1, 2, 5 }

- S^{yes} = { 1, 2, 5 }



PCTL model checking - Summary

- Computation of set Sat(Φ) for MDP M and PCTL formula Φ
  - recursive descent of parse tree
  - combination of graph algorithms, numerical computation

- Probabilistic operator P:
  - $X\ \Phi$: one matrix-vector multiplication, $O(|S|^2)$
  - $\Phi_1\ U^{\leq k}\ \Phi_2$: k matrix-vector multiplications, $O(k|S|^2)$
  - $\Phi_1\ U\ \Phi_2$: linear programming problem, polynomial in $|S|$ (assuming use of linear programming)

- Complexity:
  - linear in $|\Phi|$ and polynomial in $|S|$
  - S is states in MDP, assume $|Steps(s)|$ is constant

Summary

- Markov decision processes (MDPs)
  - extend DTMCs with nondeterminism
  - to model concurrency, underspecification, …
- An adversary resolve nondeterminism in an MDP
  - induce a probability space over paths
  - consider minimum/maximum probabilities over all adversaries
- Property specifications
  - use e.g. PCTL or LTL, as for DTMCs
  - but quantify over all adversaries
- Model checking algorithms
  - covered three basic techniques for MDPs: linear programming, value iteration, or policy iteration

- ~~Tomorrow: continuous-time Markov chains (CTMCs)~~