UNIVERSITY OF OXFORD

UNIVERSITÄT DES SAARLANDES

Model Checking for Probabilistic Hybrid Systems

Marta Kwiatkowska, Ernst Moritz Hahn
Oxford University Computing Laboratory

Holger Hermanns, Arnd Hartmanns
Saarland University, Dependable Systems and Software

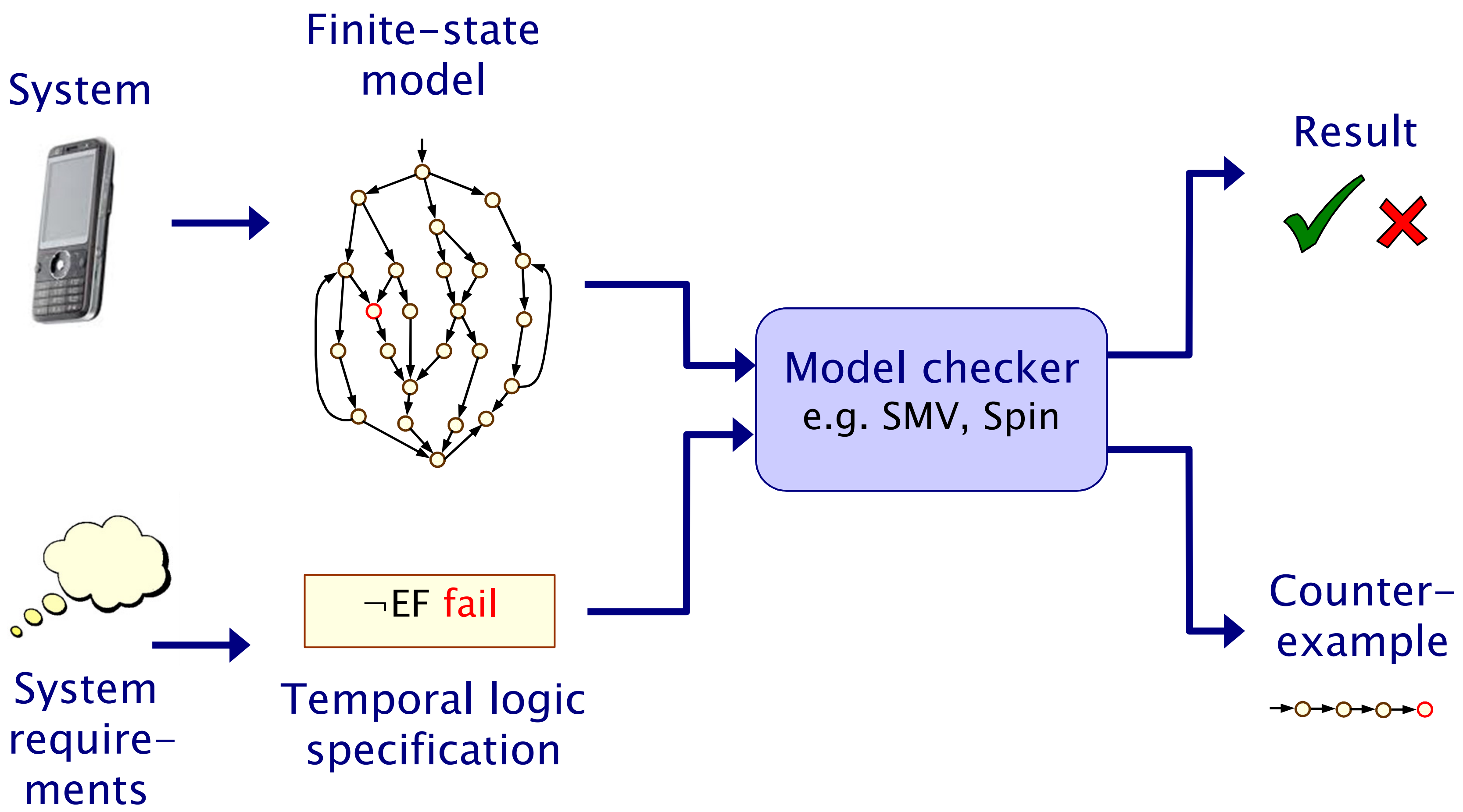
CPSWeek'13, Philadelphia, April 2013

Introduction

Probabilistic models and
probabilistic model checking

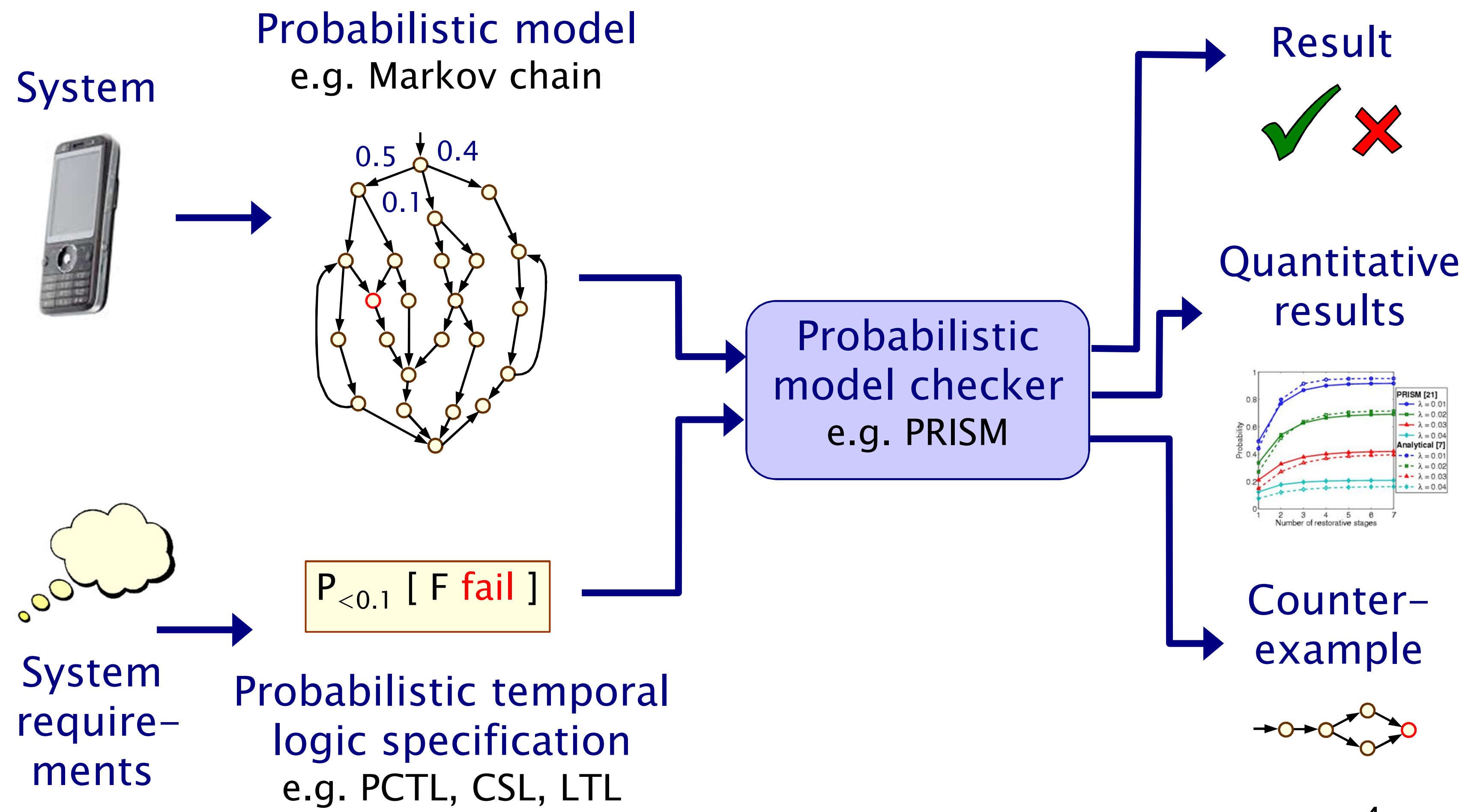
Model checking

Automated formal verification for finite-state models



Probabilistic model checking

Automatic verification of systems with probabilistic behaviour



Why probability?

- Some systems are inherently probabilistic…

- Randomisation, e.g. in distributed coordination algorithms
  - as a symmetry breaker, in gossip routing to reduce flooding

- Examples: real-world protocols featuring randomisation:
  - Randomised back-off schemes
    - CSMA protocol, 802.11 Wireless LAN
  - Random choice of waiting time
    - IEEE1394 Firewire (root contention), Bluetooth (device discovery)
  - Random choice over a set of possible addresses
    - IPv4 Zeroconf dynamic configuration (link-local addressing)
  - Randomised algorithms for anonymity, contract signing, …

Why probability?

- Some systems are inherently probabilistic…
- Randomisation, e.g. in distributed coordination algorithms
  - as a symmetry breaker, in gossip routing to reduce flooding
- To model uncertainty and performance
  - to quantify rate of failures, express Quality of Service
- Examples:
  - computer networks, embedded systems
  - power management policies
  - nano-scale circuitry: reliability through defect-tolerance

Why probability?

- Some systems are inherently probabilistic…
- Randomisation, e.g. in distributed coordination algorithms
  – as a symmetry breaker, in gossip routing to reduce flooding
- To model uncertainty and performance
  – to quantify rate of failures, express Quality of Service
- To model biological processes
  – reactions occurring between large numbers of molecules are naturally modelled in a stochastic fashion

Verifying probabilistic systems

- We are not just interested in correctness
- We want to be able to quantify:
  - security, privacy, trust, anonymity, fairness
  - safety, reliability, performance, dependability
  - resource usage, e.g. battery life
  - and much more…
- Quantitative, as well as qualitative requirements:
  - how reliable is my car's Bluetooth network?
  - how efficient is my phone's power management policy?
  - is my bank's web-service secure?
  - what is the expected long-run percentage of protein X?

Probabilistic models

- Markov Decision Process (MDP)
  - probabilistic and nondeterministic behaviour
  - already allow to express relevant class of models
  - semantic base for extended models below

- Probabilistic Timed Automata (PTA)
  - extend MDPs with clocks to express timed behaviour

- Probabilistic Hybrid Automata (PHA)
  - extend clocks of PTAs to more general continuous variables
  - often described by differential equations

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}

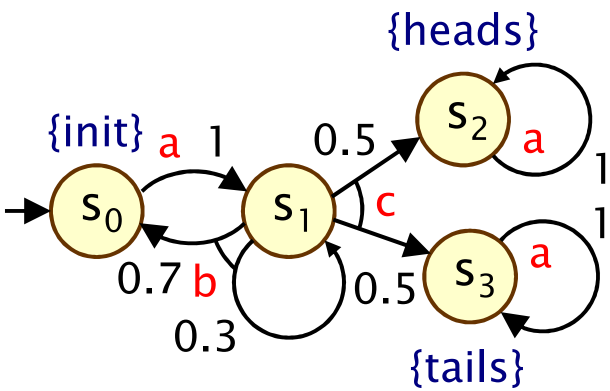
- Unknown environments
  - e.g. probabilistic security protocols - unknown adversary

Markov decision processes

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

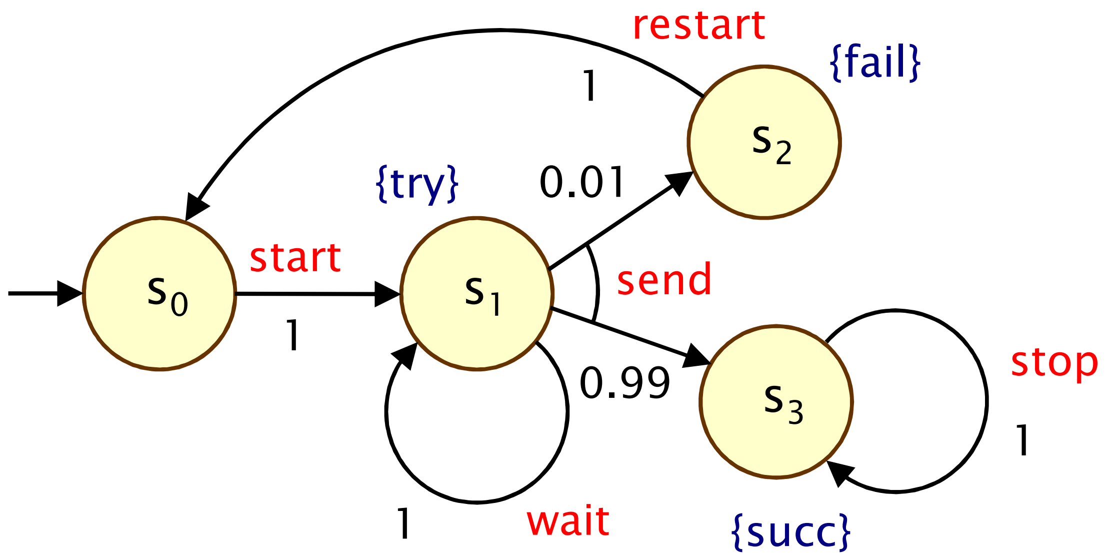
    where Act is a set of actions and Dist(S) is the set of discrete probability distributions over the set S
  - $L : S \rightarrow 2^{AP}$ is a labelling with atomic propositions

- Notes:
  - **Steps**(s) is always non-empty, i.e. no deadlocks
  - the use of actions to label distributions is optional



Simple MDP example

- Simple 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



Modelling MDPs

- Guarded Commands modelling language
  - simple, textual, state-based language
  - based on Reactive Modules [AH99]
  - basic components: modules, variables and commands
- Modules:
  - components of system being modelled
  - a module represents a single MDP

```
module example
    ...
endmodule
```

Modelling MDPs

- Guarded Commands modelling language
  - simple, textual, state-based language
  - based on Reactive Modules [AH99]
  - basic components: modules, variables and commands
- Variables:
  - finite-domain (bounded integer ranges or Booleans)
  - local or global – anyone can read, only owner can modify
  - variable valuation = state of the MDP

```
module example
    s : [0..3] init 0;
    ...
endmodule
```

Modelling MDPs

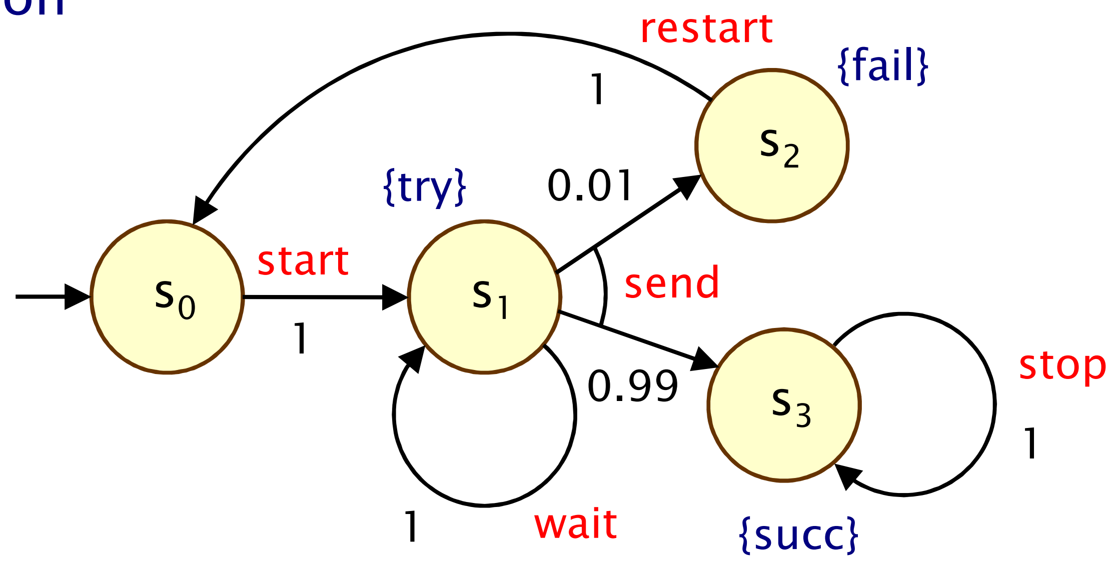
- Guarded Commands modelling language
  - simple, textual, state-based language
  - based on Reactive Modules [AH99]
  - basic components: modules, variables and commands
- Commands:
  - describe the transitions between the states

$$\underbrace{\texttt{[act]}}_{\text{action}}\ \underbrace{\texttt{exp}}_{\text{guard}}\ \texttt{->}\ \underbrace{\texttt{p}_1}_{\text{probability}}\texttt{:}\ \underbrace{\texttt{asgn}_{11}\ \&\ \texttt{asgn}_{12}\ \&\ \texttt{...}}_{\text{update}}\ \texttt{+ ... +}\ \underbrace{\texttt{p}_n}_{\text{probability}}\texttt{:}\ \underbrace{\texttt{asgn}_{n1}\ \&\ \texttt{...}}_{\text{update}}\ \texttt{;}$$

```
module example
    s : [0..3] init 0;
    ...
    [send] (s = 1) -> 0.01: (s' = 2) + 0.99: (s' = 3);
    ...
endmodule
```

Simple MDP example

- Simple communication protocol

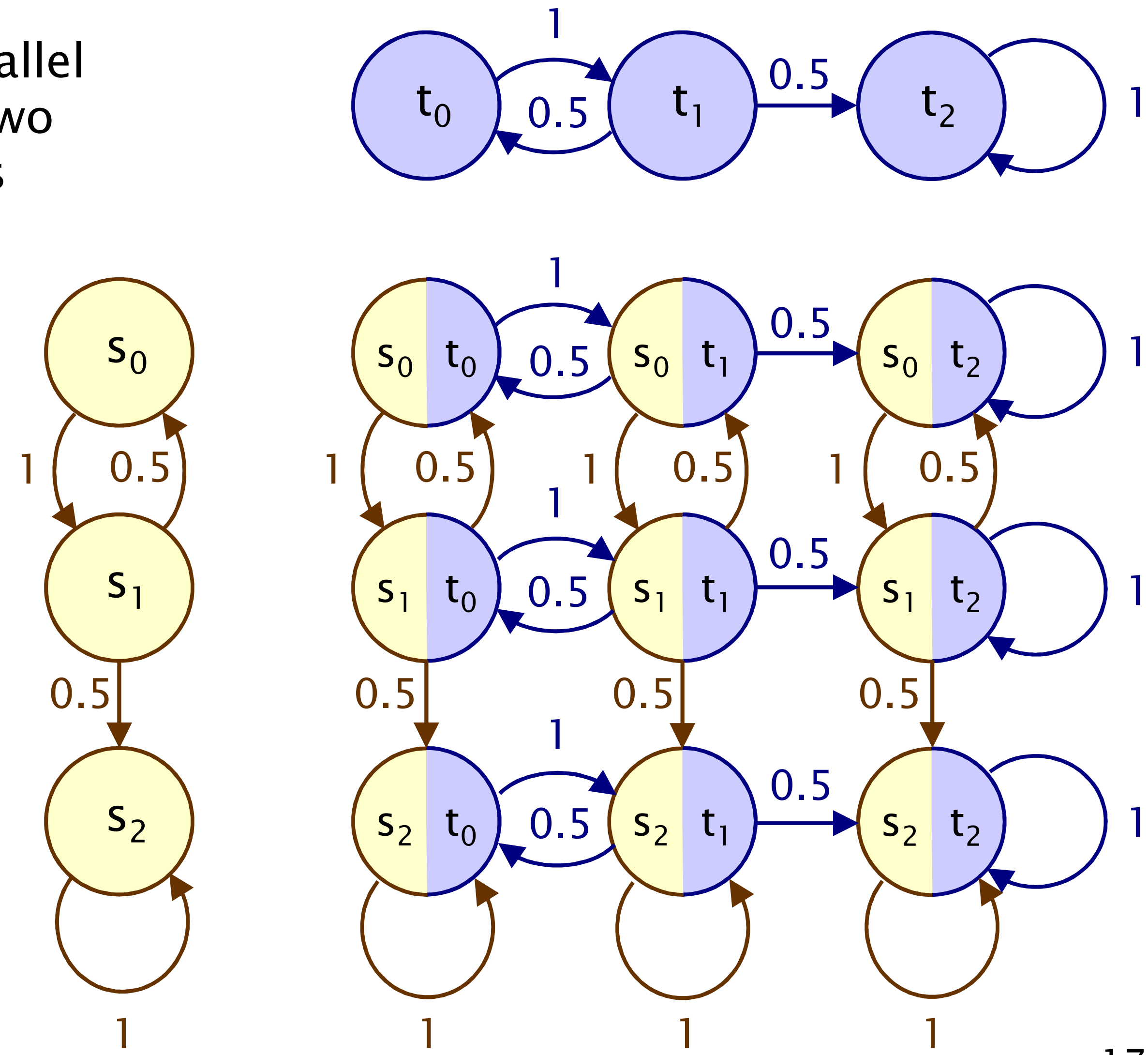



```
module example
   s : [0..3] init 0;
   [start]   (s = 0) -> (s' = 1);
   [wait]    (s = 1) -> true;
   [send]    (s = 1) -> 0.01: (s' = 2) + 0.99: (s' = 3);
   [restart] (s = 2) -> (s' = 0);
   [stop]    (s = 3) -> true;
endmodule
```

Example – Parallel composition

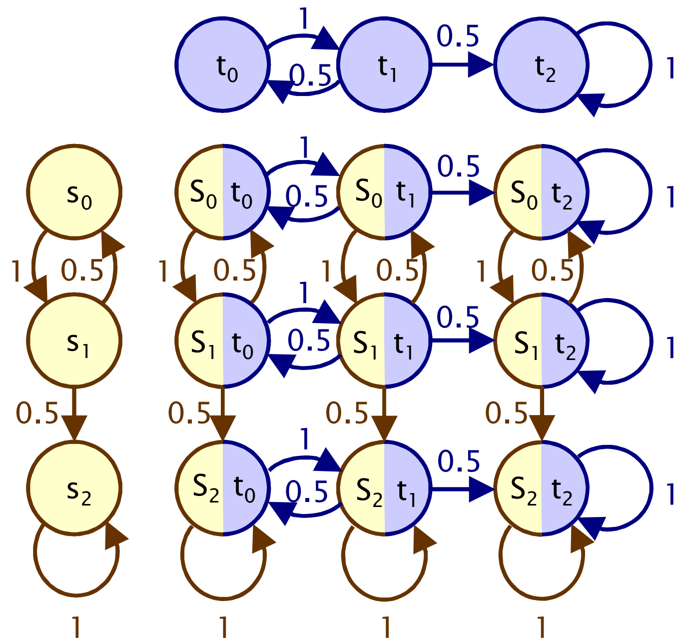
Asynchronous parallel composition of two 3-state DTMCs

Action labels omitted here



Example – Parallel composition

Asynchronous parallel composition of two 3-state DTMCs




```
module threestate
    s : [0..2] init 0;
    [] s=0 -> (s'=1);
    [] s=1 -> 0.5: (s'=s-1)
            + 0.5: (s'=s+1);
    [] s>1 -> true;
endmodule

module copy = threestate[s=t] endmodule

system
    threestate || copy
endsystem
```

Default parallel composition on matching action labels – can be omitted

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 $(a_i,\mu_i) \in \mathbf{Steps}(s_i)$ 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 Markov chain (DTMC)
  - …for which we can define a probability measure over paths

Overview (Part 1)

- Markov decision processes (MDPs)
- Adversaries
- PCTL
- PCTL model checking
- Costs and rewards
- Case study: Firewire root contention

Adversaries

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

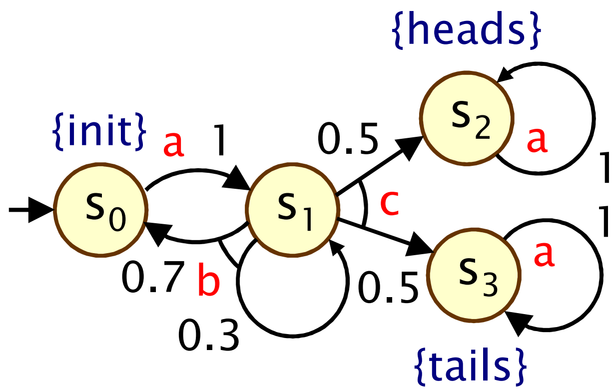
- Formally:
  - an adversary A of an MDP M is a function mapping every finite path $\omega = s_0(a_1,\mu_1)s_1 \ldots s_n$ to an element of **Steps**(s_n)

- For each A can define a probability measure Pr^A_s over paths
  - constructed through an infinite state Markov chain (DTMC)
  - 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

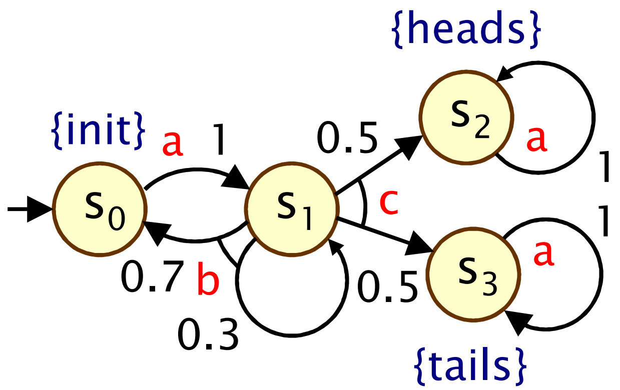
- Adversary A_1
  - picks action c the first time
  - $A_1(s_0 s_1)=(c,\mu_c)$

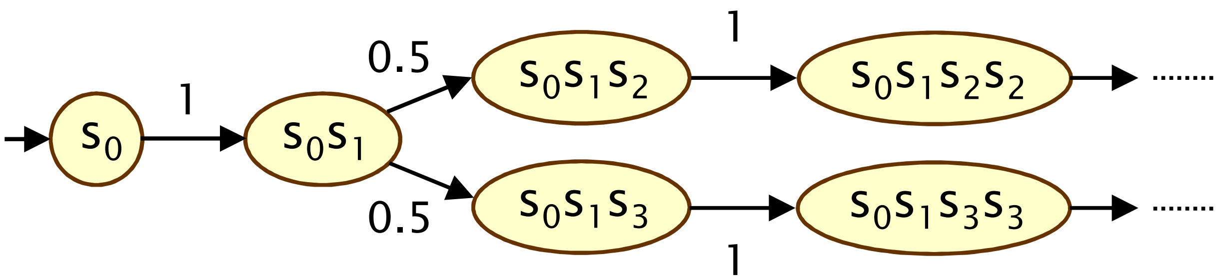


- Adversary A_2
  - picks action b the first time, then c
  - $A_2(s_0 s_1)=(b,\mu_b)$, $A_2(s_0 s_1 s_1)=(c,\mu_c)$, $A_2(s_0 s_1 s_0 s_1)=(c,\mu_c)$

Adversaries – Examples

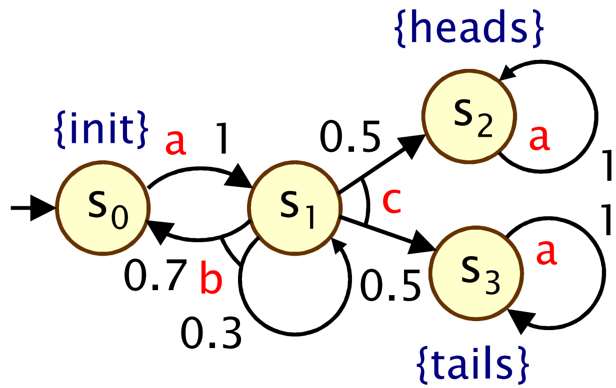
- Fragment of DTMC for adversary A_1
  - A_1 picks action c the first time

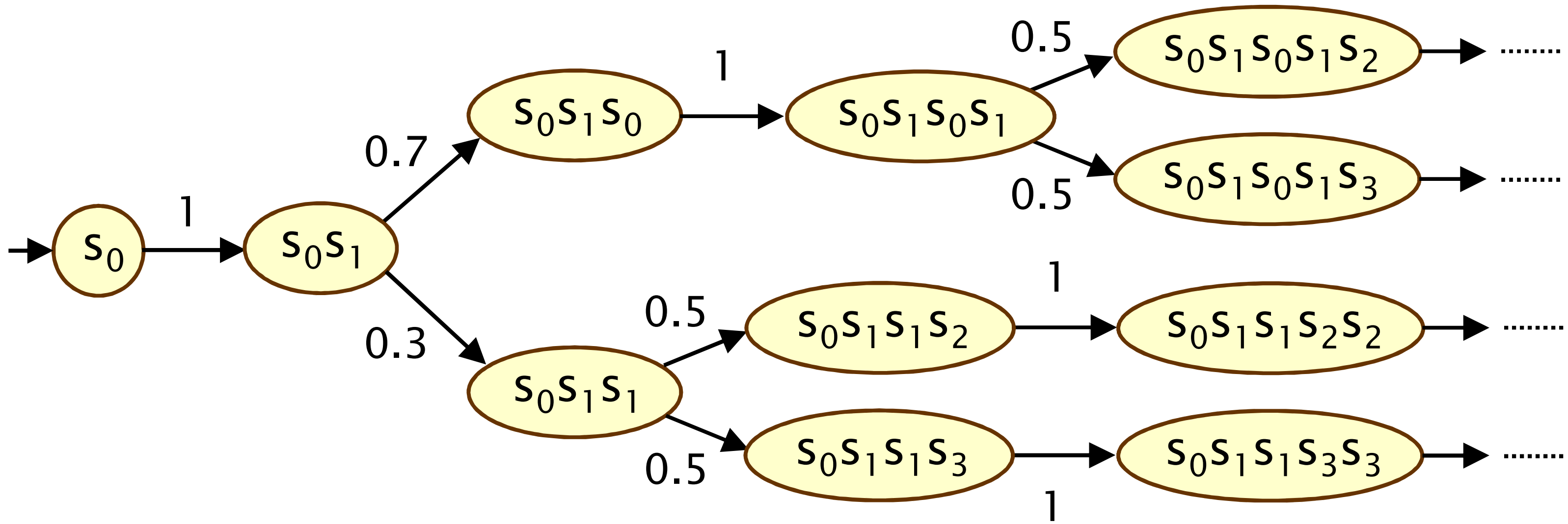




Adversaries – Examples

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

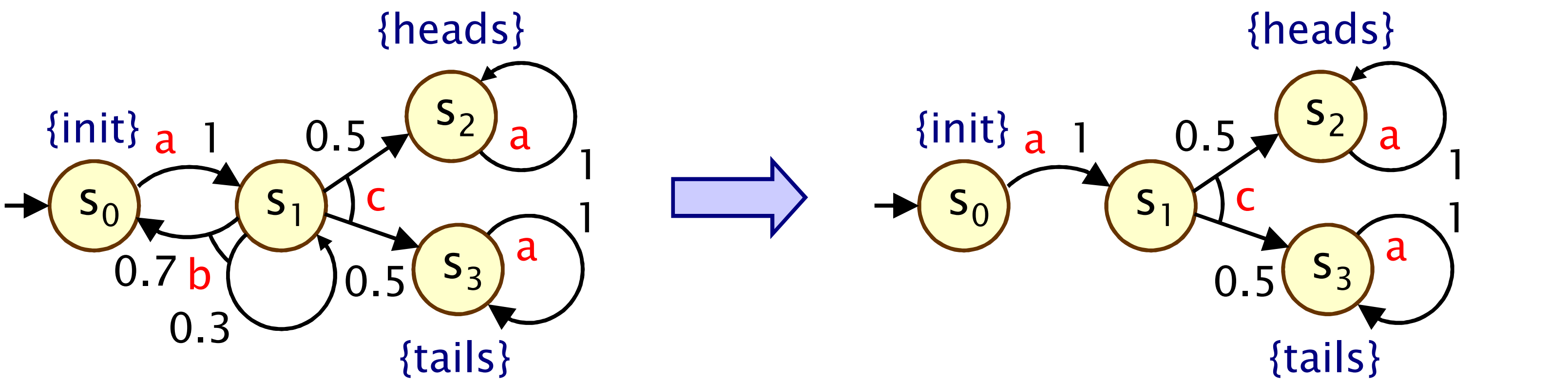




Memoryless adversaries

- 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 1)

- Markov decision processes (MDPs)
- Adversaries
- PCTL
- PCTL model checking
- Costs and rewards
- Case study: Firewire root contention

PCTL

- Temporal logic for describing properties of MDPs
  - PCTL = Probabilistic Computation Tree Logic [HJ94]
  - essentially the same as the logic pCTL of [ASB+95]
- Extension of (non-probabilistic) temporal logic CTL
  - key addition is probabilistic operator P
  - quantitative extension of CTL's A and E operators
- Example
  - send → $P_{\geq 0.95}$ [true $U^{\leq 10}$ deliver]
  - "if a message is sent, then the probability of it being delivered within 10 steps is at least 0.95"

PCTL syntax

ψ is true with probability ~p

- PCTL syntax:
  - $\phi ::= \text{true} \mid a \mid \phi \wedge \phi \mid \neg\phi \mid P_{\sim p} [\, \psi \,]$ (state formulas)
  - $\psi ::= X\, \phi \mid \phi\, U^{\leq k}\, \phi \mid \phi\, U\, \phi$ (path formulas)

"next" "bounded until" "until"

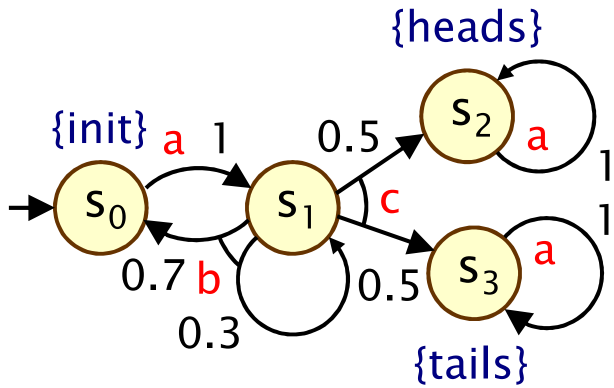
  - where a is an atomic proposition, used to identify states of interest, $p \in [0,1]$ is a probability, $\sim \in \{<,>,\leq,\geq\}$, $k \in \mathbb{N}$
- A PCTL formula is always a state formula
  - path formulas only occur inside the P operator

PCTL semantics for MDPs

- PCTL formulas interpreted over states of an MDP
  - $s \models \phi$ denotes ϕ is "true in state s" or "satisfied in state s"

- Semantics of (non-probabilistic) state formulas:
  - for a state s of the MDP (S,s_{init},P,L):
  - $s \models a \quad \Leftrightarrow \quad a \in L(s)$
  - $s \models \phi_1 \wedge \phi_2 \quad \Leftrightarrow \quad s \models \phi_1$ and $s \models \phi_2$
  - $s \models \neg\phi \quad \Leftrightarrow \quad s \models \phi$ is false

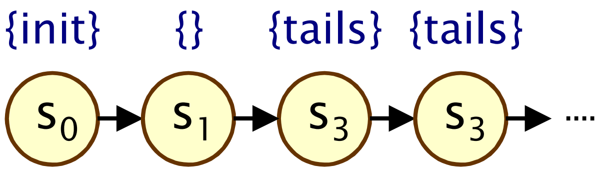
- Examples
  - $s_3 \models$ tails
  - $s_2 \models$ heads $\wedge \neg$init



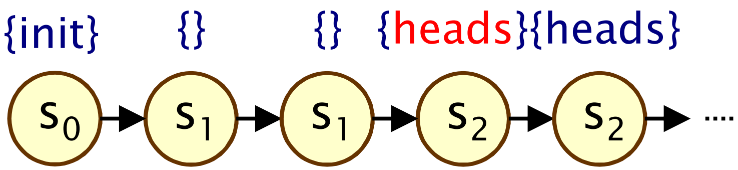
PCTL semantics for MDPs

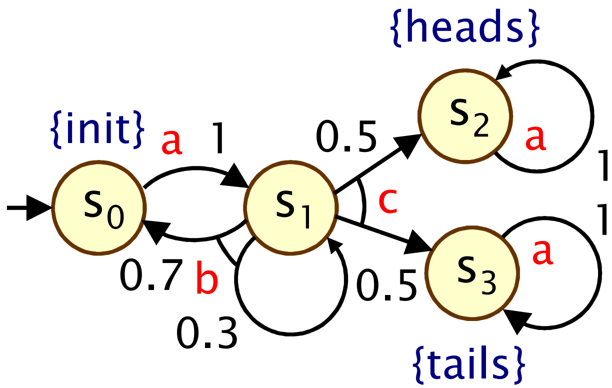
- Semantics of path formulas:
  - for a path $\omega = s_0 s_1 s_2 \ldots$ in the MDP:
  - $\omega \vDash X\ \phi \quad \Leftrightarrow \quad s_1 \vDash \phi$
  - $\omega \vDash \phi_1\ U^{\leq k}\ \phi_2 \quad \Leftrightarrow \quad \exists i \leq k$ such that $s_i \vDash \phi_2$ and $\forall j<i,\ s_j \vDash \phi_1$
  - $\omega \vDash \phi_1\ U\ \phi_2 \quad \Leftrightarrow \quad \exists k \geq 0$ such that $\omega \vDash \phi_1\ U^{\leq k}\ \phi_2$

- Some examples of satisfying paths:
  - $X\ \neg init$



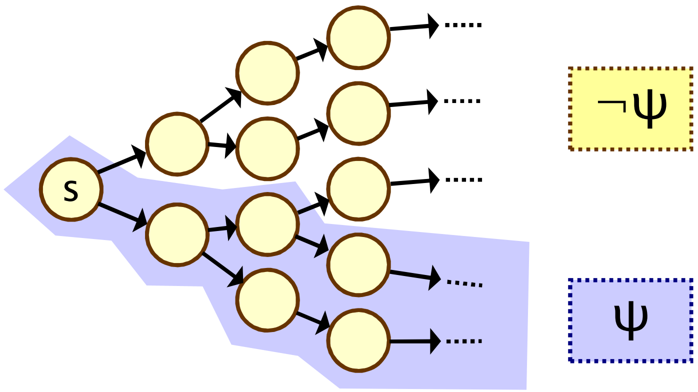
  - $\neg tails\ U\ heads$





PCTL semantics for MDPs

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



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

Minimum and maximum probabilities

- Letting:
  - $p_{max}(s, \psi) = \sup_A Prob^A(s, \psi)$
  - $p_{min}(s, \psi) = \inf_A Prob^A(s, \psi)$
- We have:
  - if $\sim \in \{\geq,>\}$, then $s \models P_{\sim p} [\ \psi\ ] \quad \Leftrightarrow \quad p_{min}(s, \psi) \sim p$
  - if $\sim \in \{<,\leq\}$, then $s \models P_{\sim p} [\ \psi\ ] \quad \Leftrightarrow \quad p_{max}(s, \psi) \sim p$
- 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:
  - $Prob^{Amin}(s, \psi) = p_{min}(s, \psi)$ and $Prob^{Amax}(s, \psi) = p_{max}(s, \psi)$

Overview (Part 1)

- Markov decision processes (MDPs)
- Adversaries
- PCTL
- PCTL model checking
- Costs and rewards
- Case study: Firewire root contention

PCTL model checking

- Algorithm for PCTL model checking [BdA95]
  - inputs: MDP $M=(S,s_{init},Steps,L)$, PCTL formula ϕ
  - output: $Sat(\phi) = \{ s \in S \mid s \models \phi \}$ = set of states satisfying ϕ

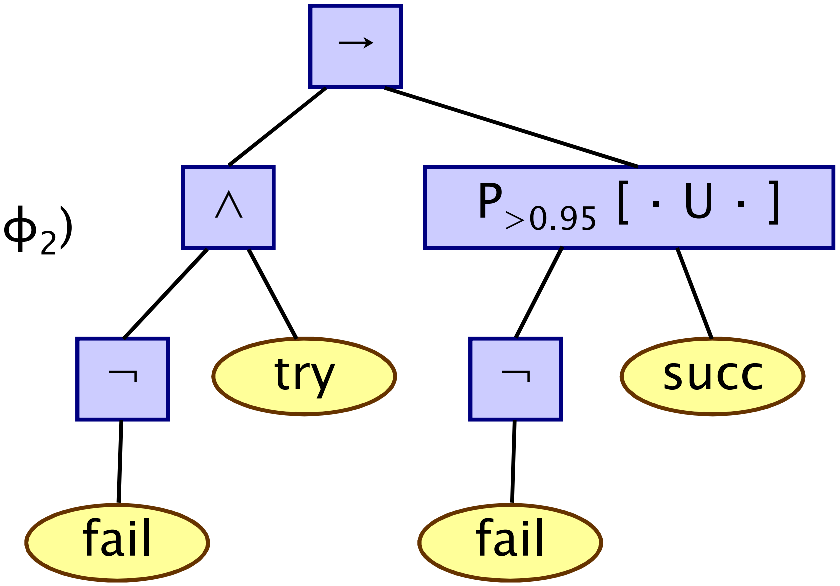
- What does it mean for an MDP D to satisfy a formula ϕ?
  - sometimes, want to check that $s \models \phi \ \forall\ s \in S$, i.e. $Sat(\phi) = S$
  - sometimes, just want to know if $s_{init} \models \phi$, i.e. if $s_{init} \in Sat(\phi)$

- Sometimes, focus on quantitative results
  - e.g. compute result of Pmax=? [F error]
  - e.g. compute result of Pmax=? [$F^{\leq k}$ error] for $0 \leq k \leq 100$

PCTL model checking for MDPs

- Basic algorithm proceeds by induction on parse tree of ϕ
  - example: $\phi = (\neg\text{fail} \wedge \text{try}) \rightarrow P_{>0.95}\ [\ \neg\text{fail}\ U\ \text{succ}\ ]$

- For the non-probabilistic operators:
  - $\text{Sat}(\text{true}) = S$
  - $\text{Sat}(a) = \{\ s \in S \mid a \in L(s)\ \}$
  - $\text{Sat}(\neg\phi) = S \setminus \text{Sat}(\phi)$
  - $\text{Sat}(\phi_1 \wedge \phi_2) = \text{Sat}(\phi_1) \cap \text{Sat}(\phi_2)$

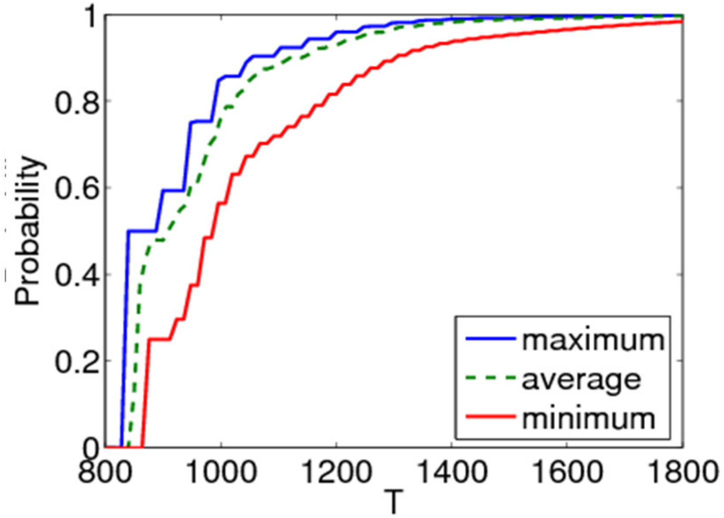


- For the $P_{\sim p}\ [\ \psi\ ]$ operator
  - need to compute the probabilities $\text{Prob}(s, \psi)$ for all states $s \in S$
  - focus here on "until" case: $\psi = \phi_1\ U\ \phi_2$

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
  - $\mathrm{Pmin}_{=?}$ [F (agreement $\wedge$ rounds$\leq$2)]
  - "what is the minimum probability that agreement is reached within two rounds?"
- CSMA/CD communication protocol
  - $\mathrm{Pmax}_{=?}$ [F collisions=k]
  - "what is the maximum probability of k collisions?"
- Self-stabilisation protocols
  - $\mathrm{Pmin}_{=?}$ [$F^{\leq t}$ stable]
  - "what is the minimum probability of reaching a stable state within k steps?"

PCTL until for MDPs

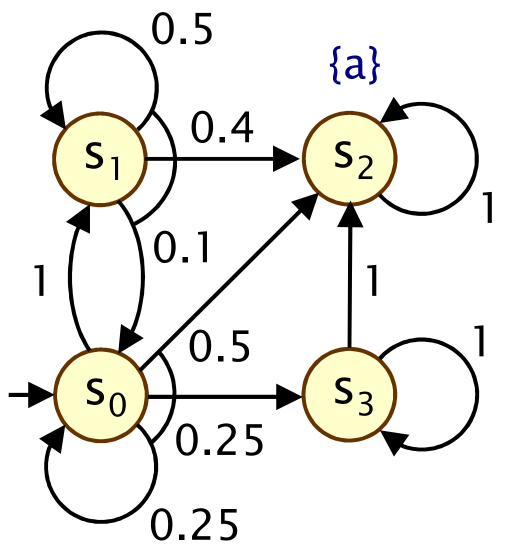
- Computation of probabilities $p_{min}(s, \phi_1 \text{ U } \phi_2)$ for all $s \in S$
- First identify all states where the probability is 1 or 0
  - "precomputation" algorithms, yielding sets S^{yes}, S^{no}
- Then compute (min) probabilities for remaining states ($S^?$)
  - either: solve linear programming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

Example:

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

$\equiv$

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

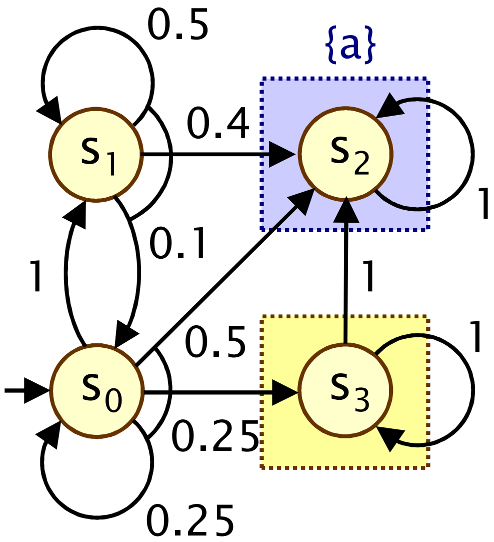


PCTL until – Precomputation

- Identify all states where $p_{min}(s, \phi_1 \text{ U } \phi_2)$ is 1 or 0
  - S^{yes} = $\text{Sat}(P_{\geq 1} [\ \phi_1 \text{ U } \phi_2\ ])$, S^{no} = $\text{Sat}(\neg\ P_{>0} [\ \phi_1 \text{ U } \phi_2\ ])$
- 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]



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

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

Method 1 – Linear programming

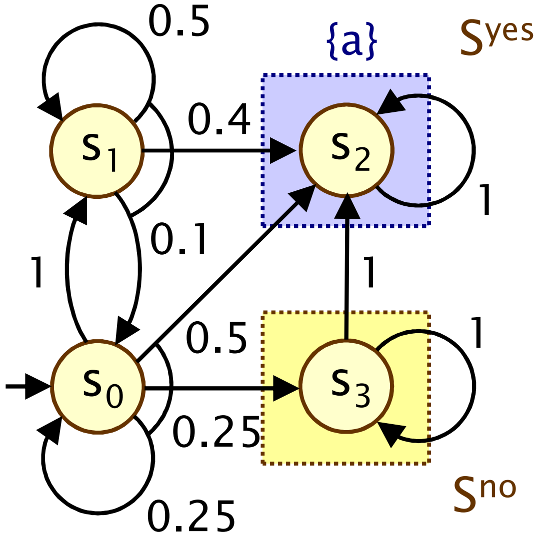
- Probabilities $p_{min}(s, \phi_1 \text{ U } \phi_2)$ for remaining states in the set $S^? = S \setminus (S^{yes} \cup S^{no})$ can be obtained as the unique solution of the following linear programming (LP) problem:

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

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

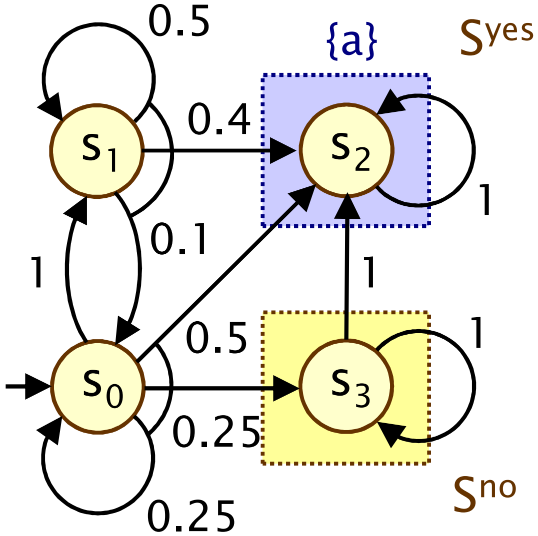
S^{yes}: $x_2=1$, S^{no}: $x_3=0$

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)



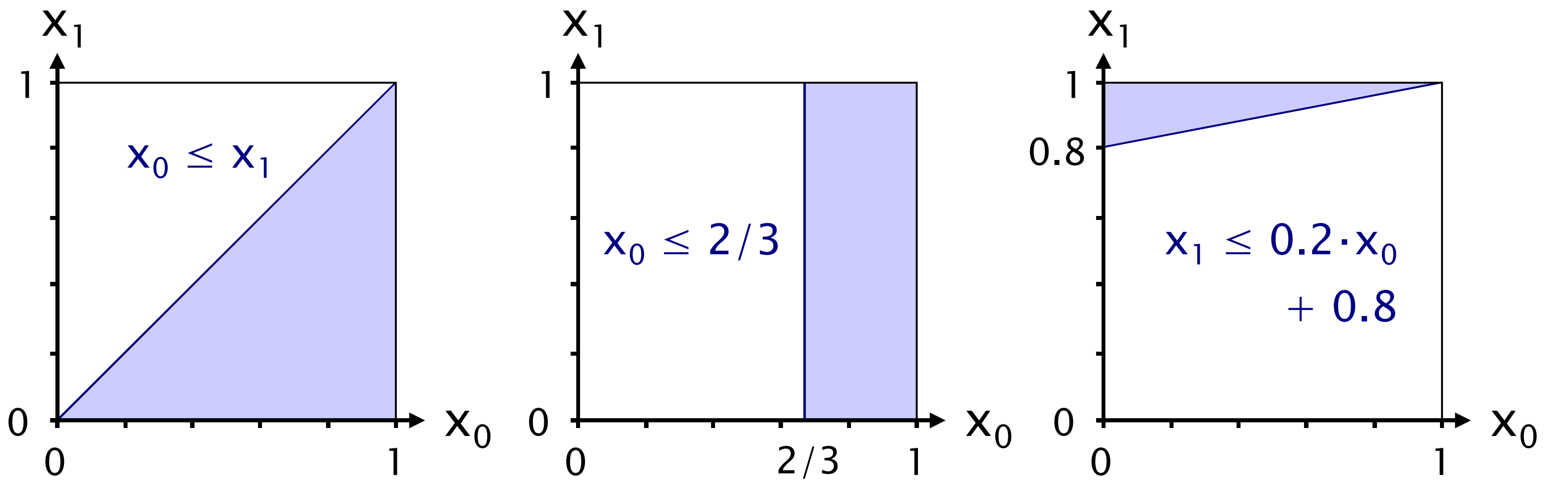
Let $x_i = p_{min}(s_i, \text{F } a)$

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

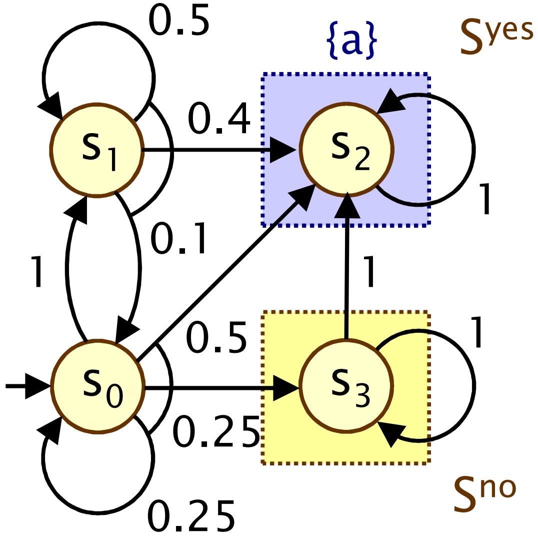
For $S^? = \{x_0, x_1\}$:

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)



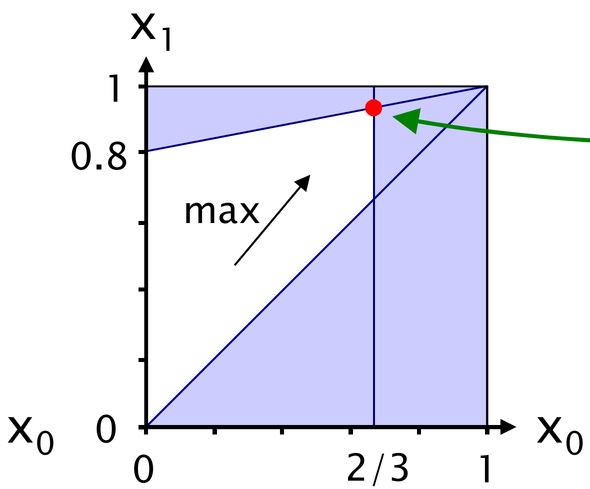
Let $x_i = p_{min}(s_i, \text{F } a)$

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

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

Maximise x_0+x_1 subject to constraints:

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



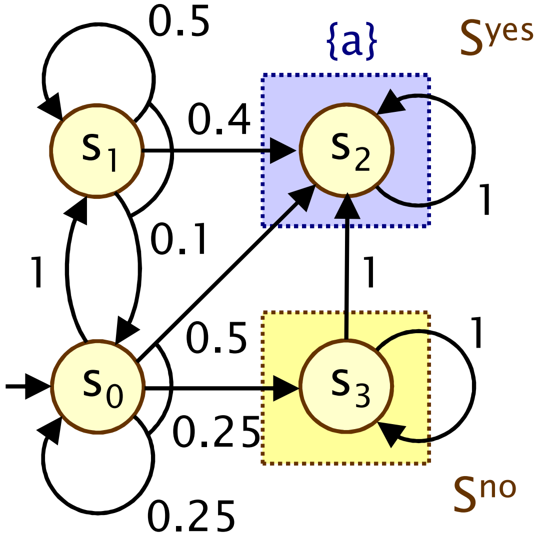
Solution:

(x_0, x_1)

$=$

$(2/3, 14/15)$

Example – PCTL until (LP)



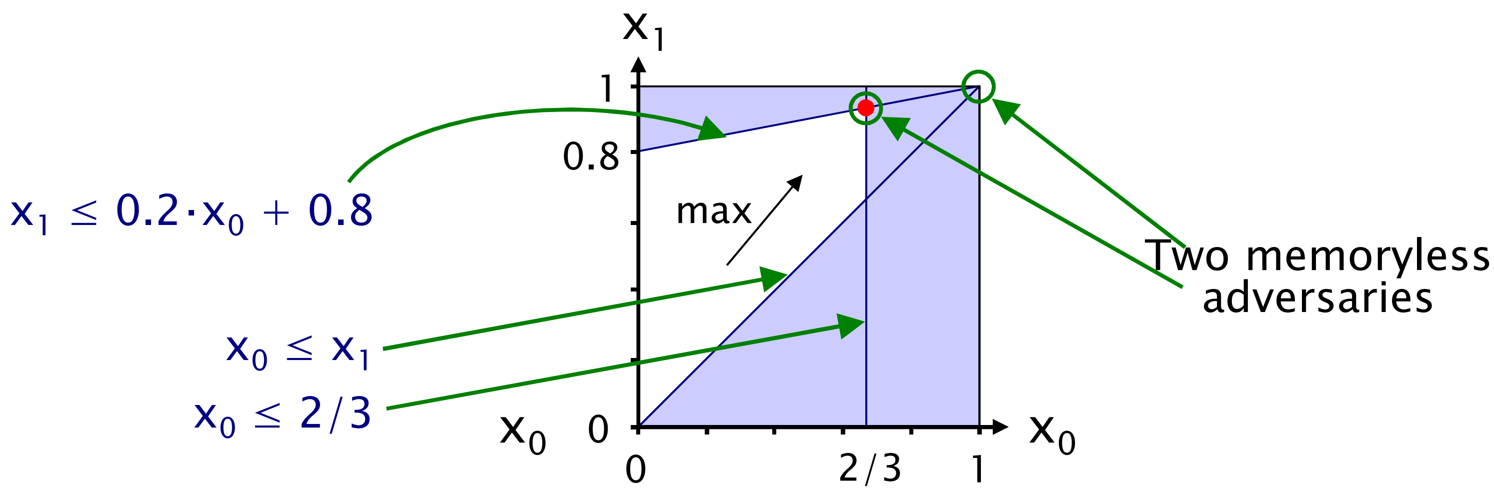
Let $x_i = p_{min}(s_i, F\ a)$

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

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

Maximise x_0+x_1 subject to constraints:

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



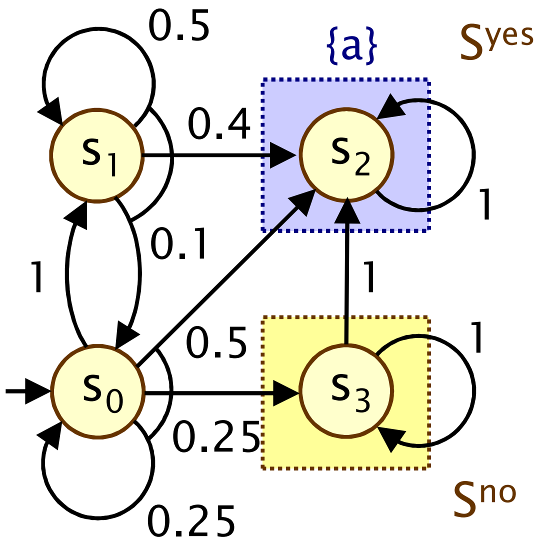
Method 2 – Value iteration

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

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

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

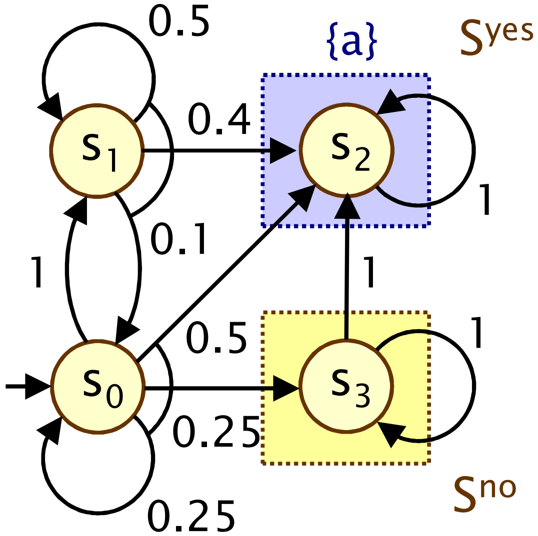
n=0: $[\ 0, 0, 1, 0\ ]$

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

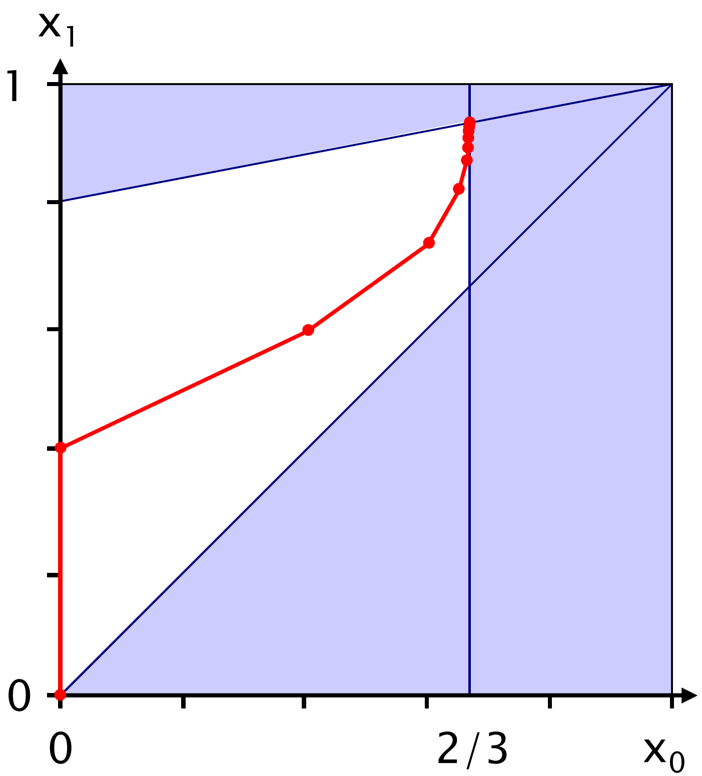
n=3: …

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]
	$\approx$ [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]

$\approx$ [2/3, 14/15, 1, 0]

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 Φ : 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 |Φ| and polynomial in |S|
  - S is states in MDP, assume |Steps(s)| is constant

Overview (Part 1)

- Markov decision processes (MDPs)
- Adversaries
- PCTL
- PCTL model checking
- Costs and rewards
- Case study: Firewire root contention

Costs and rewards

- We augment DTMCs with rewards (or, conversely, costs)
  - real-valued quantities assigned to states and/or transitions
  - these can have a wide range of possible interpretations

- Some examples:
  - elapsed time, power consumption, size of message queue, number of messages successfully delivered, net profit, …

- Costs? or rewards?
  - mathematically, no distinction between rewards and costs
  - when interpreted, we assume that it is desirable to minimise costs and to maximise rewards
  - we will consistently use the terminology “rewards” regardless

Reward-based properties

- Properties of MDPs augmented with rewards
  - allow a wide range of quantitative measures of the system
  - basic notion: expected value of rewards
  - formal property specifications will be in an extension of PCTL

- More precisely, we use two distinct classes of property…

- Instantaneous properties
  - the expected value of the reward at some time point

- Cumulative properties
  - the expected cumulated reward over some period

PCTL and rewards

- Extend PCTL to incorporate reward-based properties
  - add an R operator, which is similar to the existing P operator

expected reward is ~r

  - ϕ ::= … | $P_{\sim p}$ [ψ] | $R_{\sim r}$ [$I^{=k}$] | $R_{\sim r}$ [$C^{\leq k}$] | $R_{\sim r}$ [F ϕ]

"instantaneous" "cumulative" "reachability"

  - where $r \in \mathbb{R}_{\geq 0}$, $\sim \in \{<,>,\leq,\geq\}$, $k \in \mathbb{N}$

- $R_{\sim r}$ [·] means "the expected value of · satisfies ~r"

Types of reward formulas

- Instantaneous: $R_{\sim r}$ [$I^{=k}$]
  - "the expected value of the state reward at time-step k is ~r"
  - e.g. "the expected queue size after exactly 90 seconds"

- Cumulative: $R_{\sim r}$ [$C^{\leq k}$]
  - "the expected reward cumulated up to time-step k is ~r"
  - e.g. "the expected power consumption over one hour"

- Reachability: $R_{\sim r}$ [F ϕ]
  - "the expected reward cumulated before reaching a state satisfying ϕ is ~r"
  - e.g. "the expected time for the algorithm to terminate"

Model checking MDP reward formulas

- Instantaneous: $R_{\sim r}$ [$I^{=k}$]
  - similar to the computation of bounded until probabilities
  - solution of recursive equations
- Cumulative: $R_{\sim r}$ [$C^{\leq k}$]
  - extension of bounded until computation
  - solution of recursive equations
- Reachability: $R_{\sim r}$ [F ϕ]
  - similar to the case for P operator and until
  - graph-based precomputation (identify ∞-reward states)
  - then linear programming problem (or value iteration)

Summary

- Markov decision processes (MDPs)
  - probabilistic as well as nondeterminisitic behaviours
  - to model concurrency, underspecification, …
  - easy to model using guarded commands
- Adversaries resolve nondeterminism in an MDP
  - induce a probability space over paths
  - consider minimum/maximum probabilities over all adversaries
- Property specifications
  - probabilistic extensions of temporal logic, e.g. PCTL
  - also: expected value of costs/rewards
  - quantify over all adversaries
- Model checking algorithms
  - covered two basic techniques for MDPs: linear programming or value iteration