UNIVERSITY OF OXFORD

Quantitative verification techniques for probabilistic software

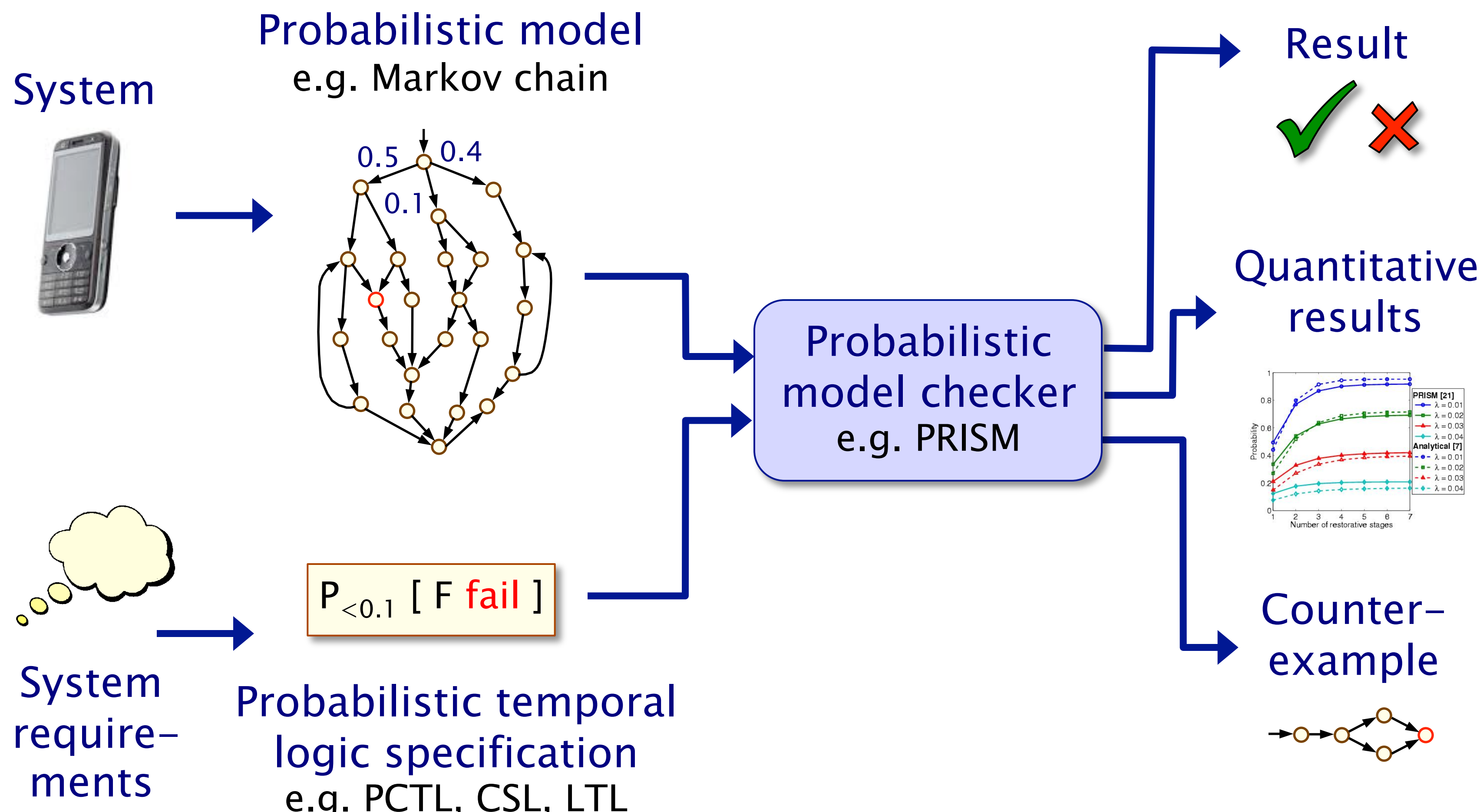
Marta Kwiatkowska

Oxford University Computing Laboratory

Summer School on Model Checking, Beijing, October 2010

Recap: Probabilistic model checking

Automatic verification of systems with probabilistic behaviour



Quantitative verification of software

- What do we need to model?
  - probability (e.g. randomisation, failures)
  - nondeterminism (e.g. concurrency, underspecification)
  - real-time behaviour and constraints (e.g. delays, time-outs)
- What do we want to verify?
  - correctness, safety, reliability, performance, resource usage…

- Goal: efficient, fully-automated probabilistic verification
  - directly from high-level programming languages
- Needs:
  - techniques/tools for verifying finite-state probabilistic models
  - compositional probabilistic verification techniques
  - abstractions for (possibly infinite-state) probabilistic models
  - refinement: automatic methods to construct abstractions

Probabilistic models

	Fully probabilistic	**Nondeterministic**
Discrete time	Discrete-time Markov chains (DTMCs)	Markov decision processes (MDPs) (probabilistic automata)
Continuous time	Continuous-time Markov chains (CTMCs)	Probabilistic timed automata (PTAs) CTMDPs/IMCs

Course overview

- 3 sessions (Mon/Tue/Thur): 6 $\times$ 50 minute lectures
  - 1: Markov decision processes (MDPs)
  - 2: Probabilistic LTL model checking
  - 3: Compositional probabilistic verification
  - 4: Abstraction, refinement and probabilistic software
  - 5: Probabilistic timed automata (PTAs)
  - 6: Software with time and probabilities
- For additional background material
  - and an accompanying list of references
  - see: http://www.prismmodelchecker.org/lectures/

Part 1

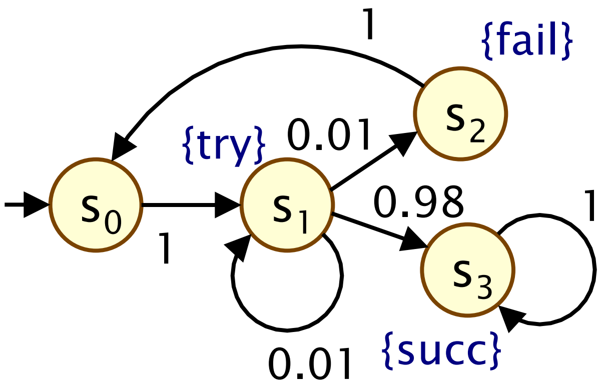
Markov decision processes

Overview (Part 1)

- Markov decision processes (MDPs)
- Adversaries & probability spaces
- Properties of MDPs: The temporal logic PCTL
- PCTL model checking for MDPs
- Case study: Firewire root contention

Recap: Discrete-time Markov chains

- Discrete-time Markov chains (DTMCs)
  - state-transition systems augmented with probabilities
- Formally: DTMC $D = (S, s_{init}, P, L)$ where:
  - S is a set of states and $s_{init} \in S$ is the initial state
  - $P : S \times S \rightarrow [0,1]$ is the transition probability matrix
  - $L : S \rightarrow 2^{AP}$ labels states with atomic propositions
  - define a probability space Pr_s over paths $Path_s$



- Properties of DTMCs
  - can be captured by the logic PCTL
  - e.g. send $\rightarrow$ $P_{\geq 0.95}$ [F deliver]
  - key question: what is the probability of reaching states $T \subseteq S$ from state s?
  - reduces to graph analysis + linear equation system

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}

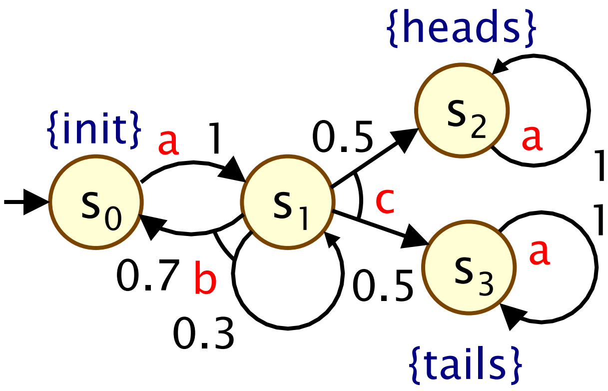
- Unknown environments
  - e.g. probabilistic security protocols – unknown adversary

Markov decision processes

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

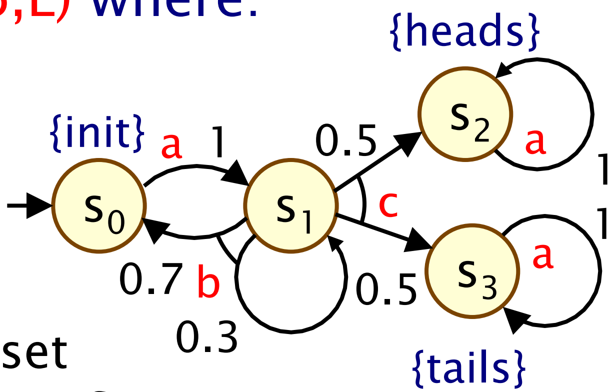
- Like DTMCs:
  - discrete set of states representing possible configurations of the system being modelled
  - transitions between states occur in discrete time-steps

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



Markov decision processes

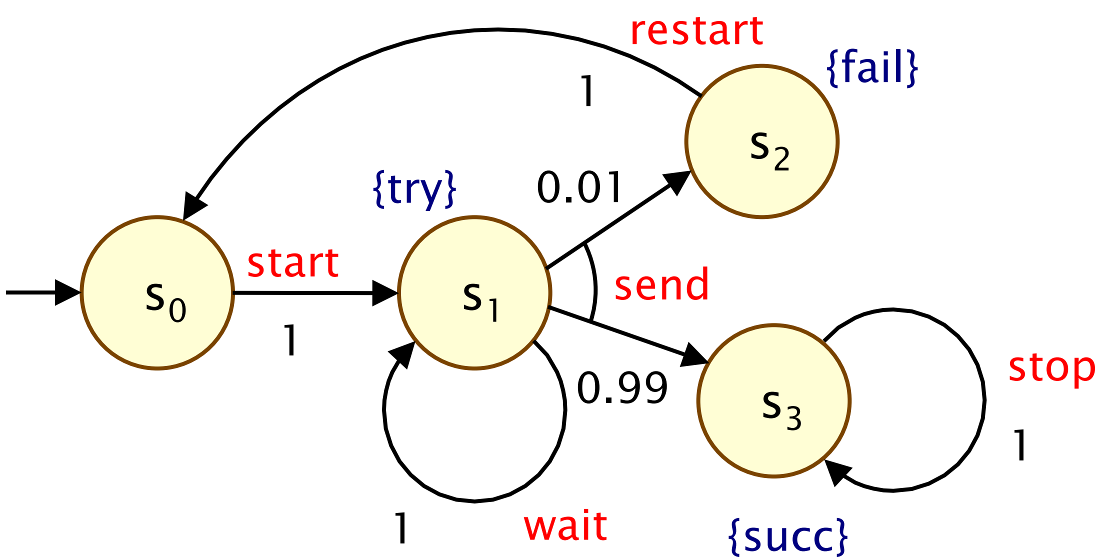
- Formally, an MDP M is a tuple $(S, s_{init}, \alpha, \delta, L)$ where:
  - S is a set of states ("state space")
  - $s_{init} \in S$ is the initial state
  - α is an alphabet of action labels
  - $\delta \subseteq S \times \alpha \times Dist(S)$ is the transition probability relation, where Dist(S) is the set of all discrete probability distributions over S
  - $L : S \rightarrow 2^{AP}$ is a labelling with atomic propositions



- Notes:
  - we also abuse notation and use δ as a function
  - i.e. $\delta : S \rightarrow 2^{\alpha \times Dist(S)}$ where $\delta(s) = \{ (a,\mu) \mid (s,a,\mu) \in \delta \}$
  - we assume $\delta(s)$ is always non-empty, i.e. no deadlocks
  - MDPs, here, are identical to probabilistic automata [Segala]

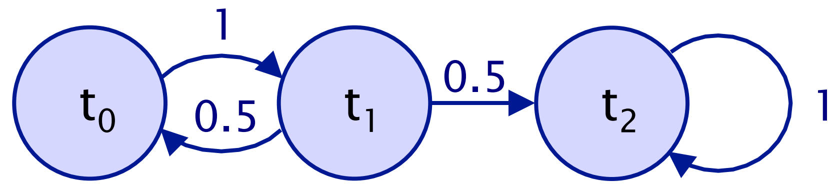
Simple MDP example

- A 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

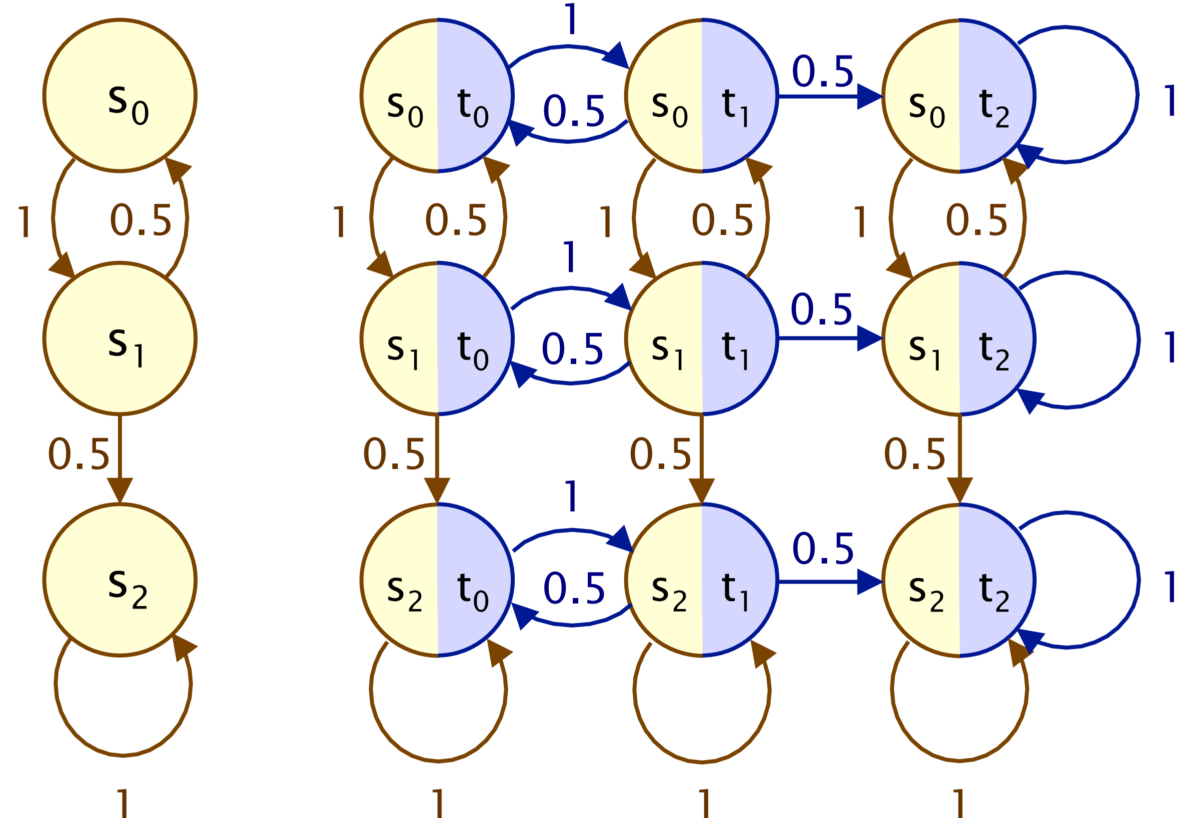


Example – Parallel composition

Asynchronous parallel composition of two 3-state DTMCs



Action labels omitted here



Paths and probabilities

- A (finite or infinite) path through an MDP M
  - 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 $(a_i,\mu_i) \in \delta(s_i)$ 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
  - $Path_{M,s}$ (or just $Path_s$) is the set of all infinite paths starting from state s in MDP M; the set of finite paths is $PathFin_s$

- 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

Overview (Part 1)

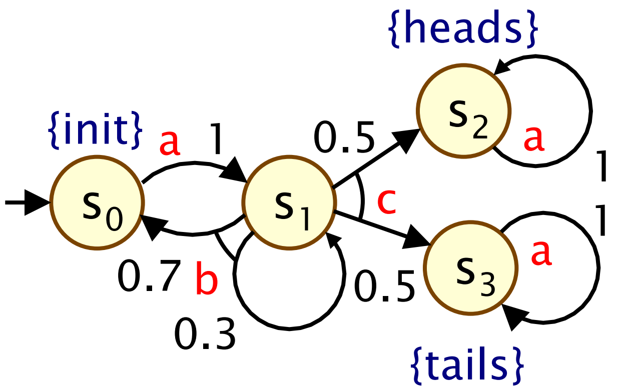
- Markov decision processes (MDPs)
- Adversaries & probability spaces
- Properties of MDPs: The temporal logic PCTL
- PCTL model checking for MDPs
- Case study: Firewire root contention

Adversaries

- An adversary resolves nondeterministic choice in an MDP
  - also known as “schedulers”, “strategies” or “policies”
- Formally:
  - an adversary σ of an MDP is a function mapping every finite path $\omega = s_0(a_0,\mu_0)s_1...s_n$ to an element of $\delta(s_n)$
- Adversary σ restricts the MDP to certain paths
  - $Path_s^\sigma \subseteq Path_s^\sigma$ and $PathFin_s^\sigma \subseteq PathFin_s^\sigma$
- Adversary σ induces a probability measure Pr_s^σ over paths
  - constructed through an infinite state DTMC $(PathFin_s^\sigma, s, \mathbf{P}_s^\sigma)$
  - states of the DTMC are the finite paths of σ starting in state s
  - initial state is s (the path starting in s of length 0)
  - $\mathbf{P}_s^\sigma(\omega,\omega')=\mu(s)$ if $\omega' = \omega(a,\mu)s$ and $\sigma(\omega)=(a,\mu)$
  - $\mathbf{P}_s^\sigma(\omega,\omega')=0$ otherwise

Adversaries – Examples

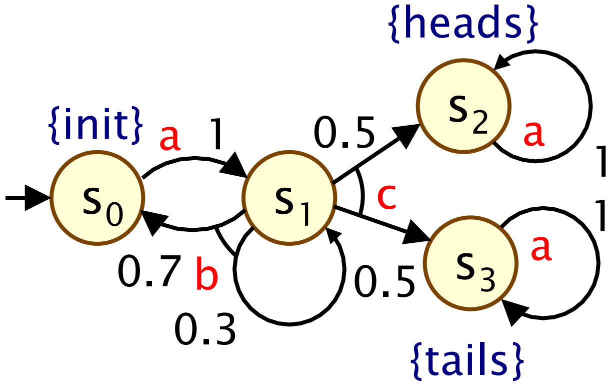
- Consider the simple MDP below
  - note that s_1 is the only state for which $|\delta(s)| > 1$
  - i.e. s_1 is the only state for which an adversary makes a choice
  - let μ_b and μ_c denote the probability distributions associated with actions b and c in state s_1

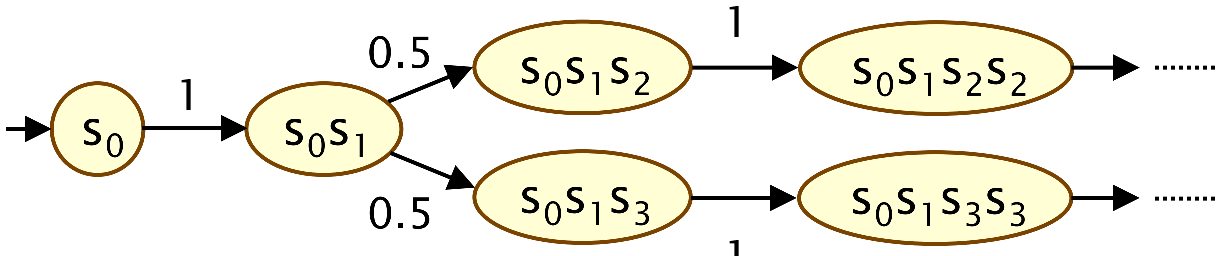


- Adversary σ_1
  - picks action c the first time
  - $\sigma_1(s_0 s_1)=(c,\mu_c)$
- Adversary σ_2
  - picks action b the first time, then c
  - $\sigma_2(s_0 s_1)=(b,\mu_b)$, $\sigma_2(s_0 s_1 s_1)=(c,\mu_c)$, $\sigma_2(s_0 s_1 s_0 s_1)=(c,\mu_c)$

Adversaries – Examples

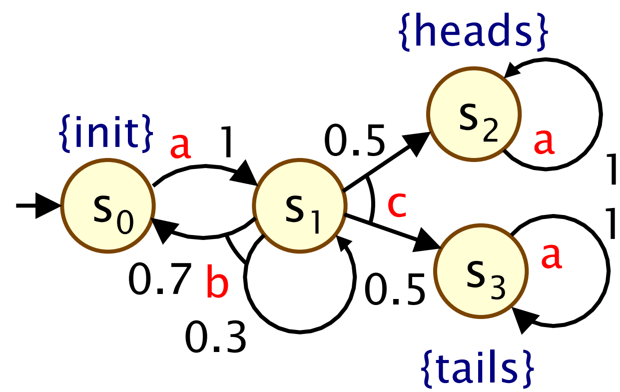
- Fragment of DTMC for adversary σ_1
  - σ_1 picks action c the first time

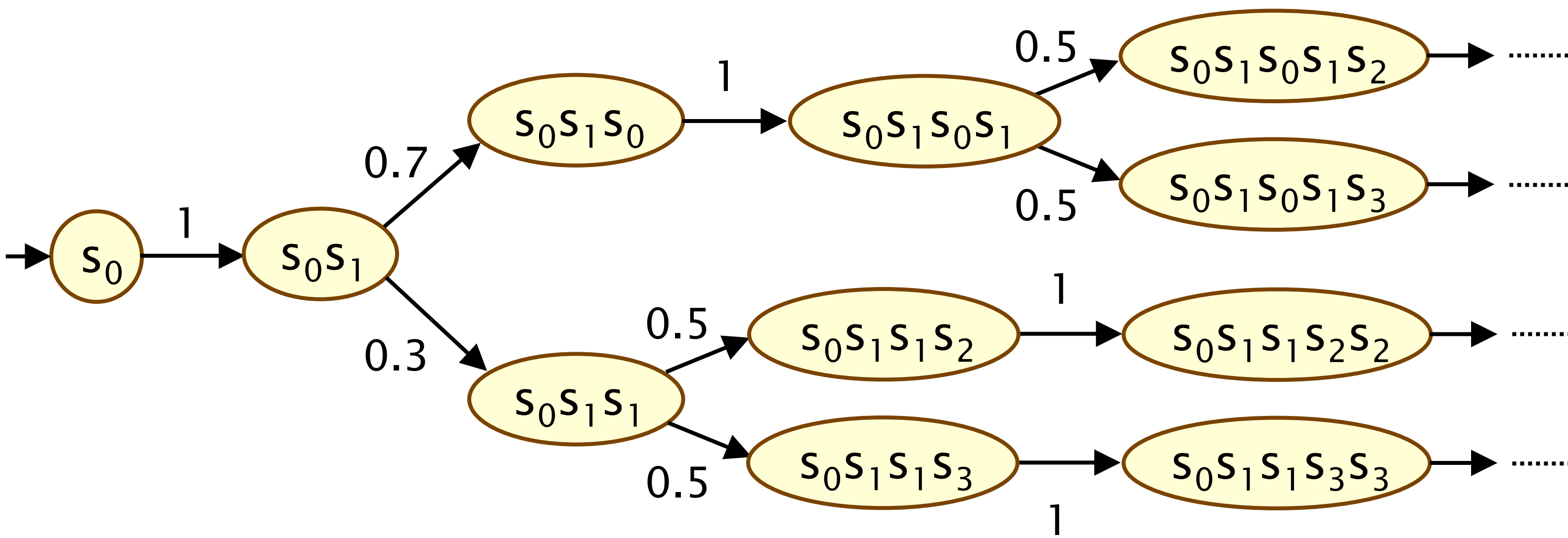




Adversaries – Examples

- Fragment of DTMC for adversary σ_2
  - σ_2 picks action b, then c

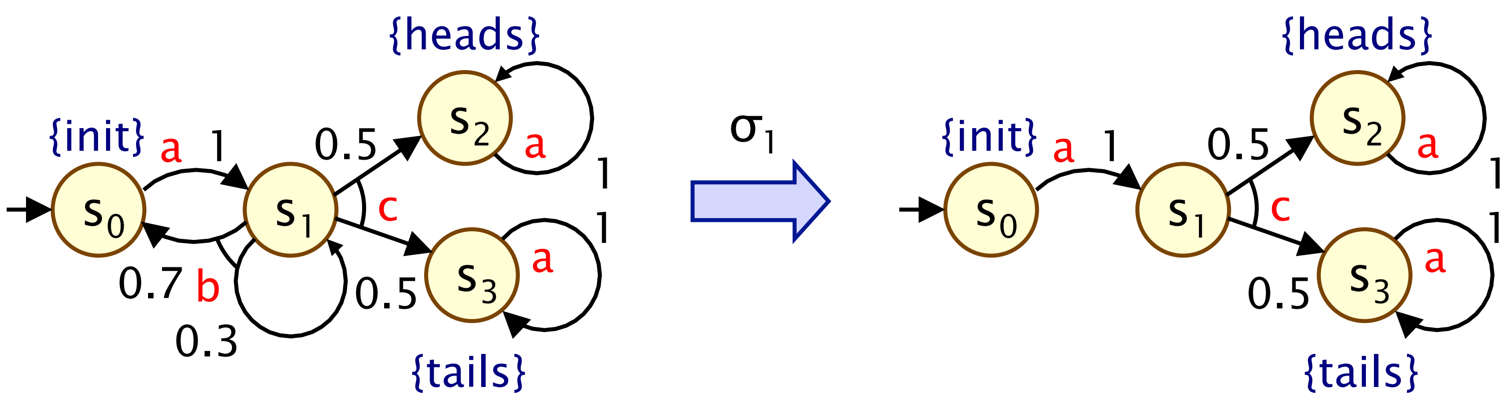




Memoryless adversaries

- Memoryless adversaries always pick same choice in a state
  - also known as: positional, simple, Markov
  - formally, for adversary σ:
  - $\sigma(s_0(a_0,\mu_0)s_1 \ldots s_n)$ depends only on s_n
  - resulting DTMC can be mapped to a |S|-state DTMC

- From previous example:
  - adversary σ_1 (picks c in s_1) is memoryless, σ_2 is not



Overview (Part 1)

- Markov decision processes (MDPs)
- Adversaries & probability spaces
- Properties of MDPs: The temporal logic PCTL
- PCTL model checking for MDPs
- Case study: Firewire root contention

PCTL

- Temporal logic for properties of MDPs (and DTMCs)
  - extension of (non-probabilistic) temporal logic CTL
  - key addition is probabilistic operator P
  - quantitative extension of CTL's A and E operators

- PCTL syntax:
  - ϕ ::= true | a | $\phi \wedge \phi$ | $\neg\phi$ | $P_{\sim p}$ [ψ]   (state formulas)
  - ψ ::= X ϕ | ϕ $U^{\leq k}$ ϕ | ϕ U ϕ (path formulas)
  - where a is an atomic proposition, used to identify states of interest, $p \in [0,1]$ is a probability, $\sim \in \{<,>,\leq,\geq\}$, $k \in \mathbb{N}$

- Example: send $\rightarrow P_{\geq 0.95}$ [true $U^{\leq 10}$ deliver]

PCTL semantics for MDPs

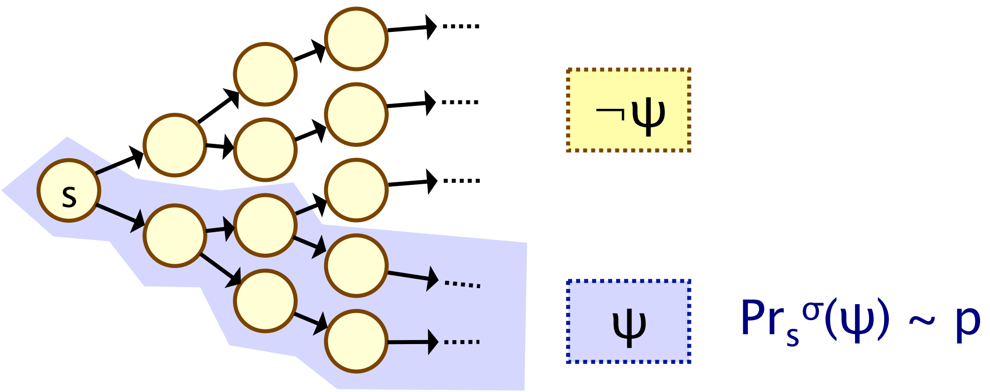
- PCTL formulas interpreted over states of an MDP
  - $s \models \phi$ denotes ϕ is "true in state s" or "satisfied in state s"

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

- Semantics of path formulas:
  - for a path $\omega = s_0(a_0,\mu_0)s_1(a_1,\mu_1)s_2\ldots$ in the MDP:
  - $\omega \models X\,\phi \quad \Leftrightarrow \quad s_1 \models \phi$
  - $\omega \models \phi_1\ U^{\leq k}\ \phi_2 \quad \Leftrightarrow \quad \exists i \leq k$ such that $s_i \models \phi_2$ and $\forall j<i,\ s_j \models \phi_1$
  - $\omega \models \phi_1\ U\ \phi_2 \quad \Leftrightarrow \quad \exists k \geq 0$ such that $\omega \models \phi_1\ U^{\leq k}\ \phi_2$

PCTL semantics for MDPs

- Semantics of the probabilistic operator P
  - can only define probabilities for a specific adversary σ
  - $s \models P_{\sim p}$ [ψ] means "the probability, from state s, that ψ is true for an outgoing path satisfies ~p for all adversaries σ"
  - formally $s \models P_{\sim p} [\, \psi \,] \Leftrightarrow Pr_s^{\sigma}(\psi) \sim p$ for all adversaries σ
  - where we use $Pr_s^{\sigma}(\psi)$ to denote $Pr_s^{\sigma} \{ \omega \in Path_s^{\sigma} \mid \omega \models \psi \}$



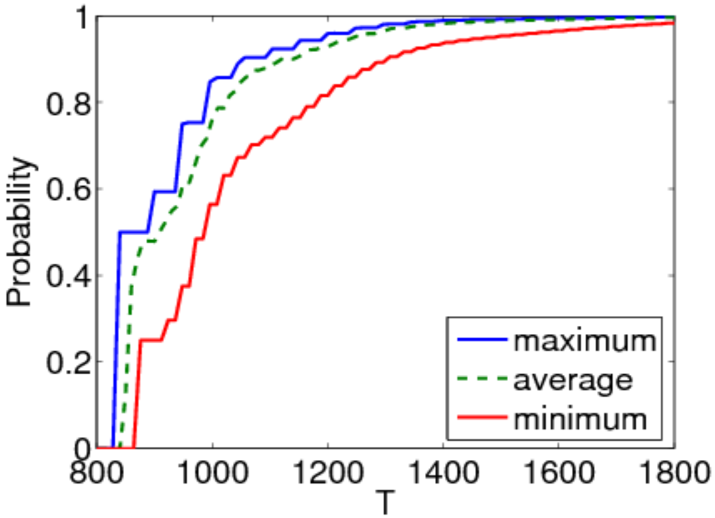
- Some equivalences:
  - $F\ \phi \equiv \Diamond\ \phi \equiv \text{true}\ U\ \phi$ (eventually, "future")
  - $G\ \phi \equiv \Box\ \phi \equiv \neg(F\ \neg\phi)$ (always, "globally")

Minimum and maximum probabilities

- Letting:
  - $Pr_s^{max}(\psi) = \sup_\sigma Pr_s^\sigma(\psi)$
  - $Pr_s^{min}(\psi) = \inf_\sigma Pr_s^\sigma(\psi)$
- We have:
  - if $\sim\, \in \{\geq,>\}$, then $s \vDash P_{\sim p}\,[\,\psi\,] \;\Leftrightarrow\; Pr_s^{min}(\psi) \sim p$
  - if $\sim\, \in \{<,\leq\}$, then $s \vDash P_{\sim p}\,[\,\psi\,] \;\Leftrightarrow\; Pr_s^{max}(\psi) \sim p$
- 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 σ_{min} and σ_{max} for which:
  - $Pr_s^{\sigma_{min}}(\psi) = Pr_s^{min}(\psi)$ and $Pr_s^{\sigma_{max}}(\psi) = Pr_s^{min}(\psi)$

Quantitative properties

- For PCTL properties with P as the outermost operator
  - quantitative form (two types): $P_{min=?}$ [ψ] and $P_{max=?}$ [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 $Pr_s^{min}(\psi)$ or $Pr_s^{max}(\psi)$ anyway
  - useful to spot patterns/trends
- Example: CSMA/CD protocol
  - "min/max probability that a message is sent within the deadline"



Other classes of adversary

- A more general semantics for PCTL over MDPs
  - parameterise by a class of adversaries Adv
- Only change is:
  - $s \models_{Adv} P_{\sim p} [\psi] \;\Leftrightarrow\; Pr_s^{\sigma}(\psi) \sim p$ for all adversaries $\sigma \in Adv$
- Original semantics obtained by taking Adv to be the set of all adversaries for the MDP
- Alternatively, take Adv to be the set of all fair adversaries
  - path fairness: if a state is occurs on a path infinitely often, then each non-deterministic choice occurs infinite often
  - see e.g. [BK98]

Some real PCTL examples

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

Overview (Part 1)

- Markov decision processes (MDPs)
- Adversaries & probability spaces
- Properties of MDPs: The temporal logic PCTL
- PCTL model checking for MDPs
- Case study: Firewire root contention

PCTL model checking for MDPs

- Algorithm for PCTL model checking [BdA95]
  - inputs: MDP $M=(S,s_{init},\alpha,\delta,L)$, PCTL formula ϕ
  - output: $Sat(\phi) = \{ s \in S \mid s \models \phi \}$ = 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 $Pr_s^{min}(\psi)$ or $Pr_s^{max}(\psi)$ for all $s \in S$
  - dependent on whether $\sim \in \{\geq,>\}$ or $\sim \in \{<,\leq\}$
  - these slides cover the case $Pr_s^{min}(\phi_1 \ U \ \phi_2)$, i.e. $\sim \in \{\geq,>\}$
  - case for maximum probabilities is very similar
  - next ($X \ \phi$) and bounded until ($\phi_1 \ U^{\leq k} \ \phi_2$) are straightforward extensions of the DTMC case

PCTL until for MDPs

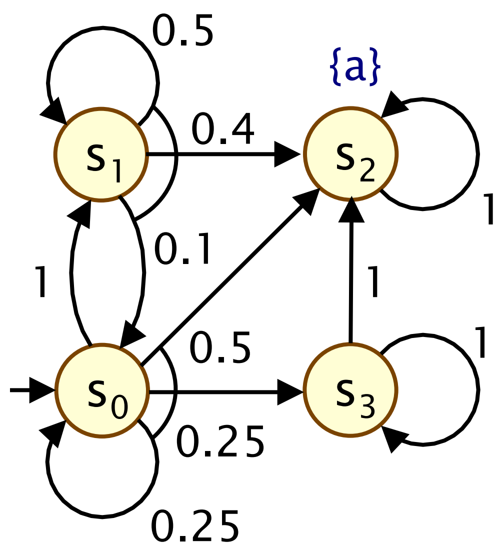
- Computation of probabilities $Pr_s^{min}(\phi_1 \text{ U } \phi_2)$ for all $s \in S$
- First identify all states where the probability is 1 or 0
  - "precomputation" algorithms, yielding sets S^{yes}, S^{no}
- Then compute (min) probabilities for remaining states ($S^?$)
  - either: solve linear programming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

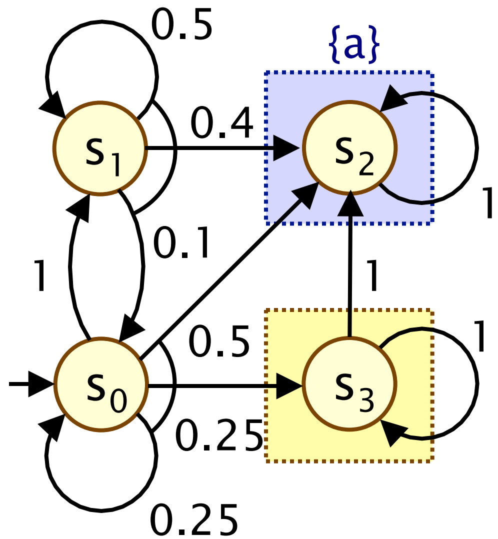


PCTL until – Precomputation

- Identify all states where $Pr_s^{min}(\phi_1 \text{ U } \phi_2)$ 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])

Method 1 – Linear programming

- Probabilities $Pr_s^{min}(\phi_1 \text{ U } \phi_2)$ 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:

$$\text{maximize} \sum_{s \in S^?} x_s \text{ subject to the constraints:}$$

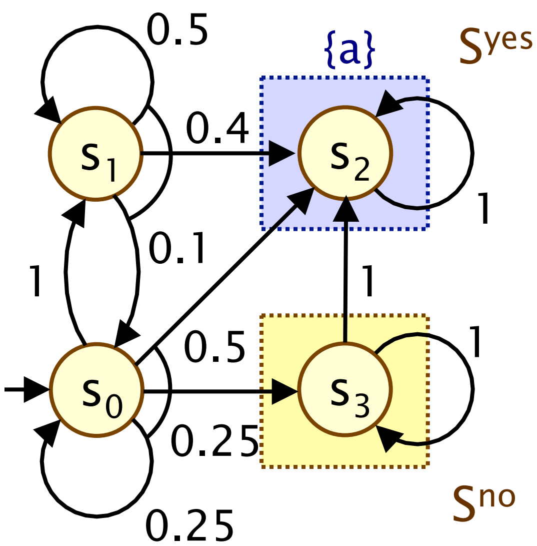
$$x_s \leq \sum_{s' \in S^?} \mu(s') \cdot x_{s'} + \sum_{s' \in S^{yes}} \mu(s')$$

$$\text{for all } s \in S^? \text{ and for all } (a, \mu) \in \delta(s)$$

- 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 $x_i = Pr_{s_i}^{min}(F\ a)$

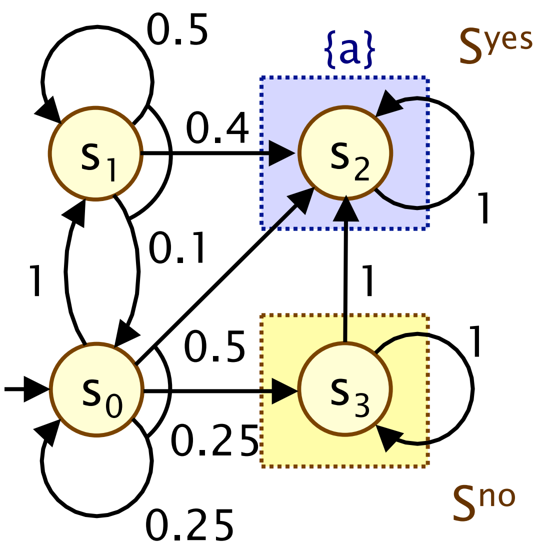
S^{yes}: $x_2=1$, S^{no}: $x_3=0$

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)



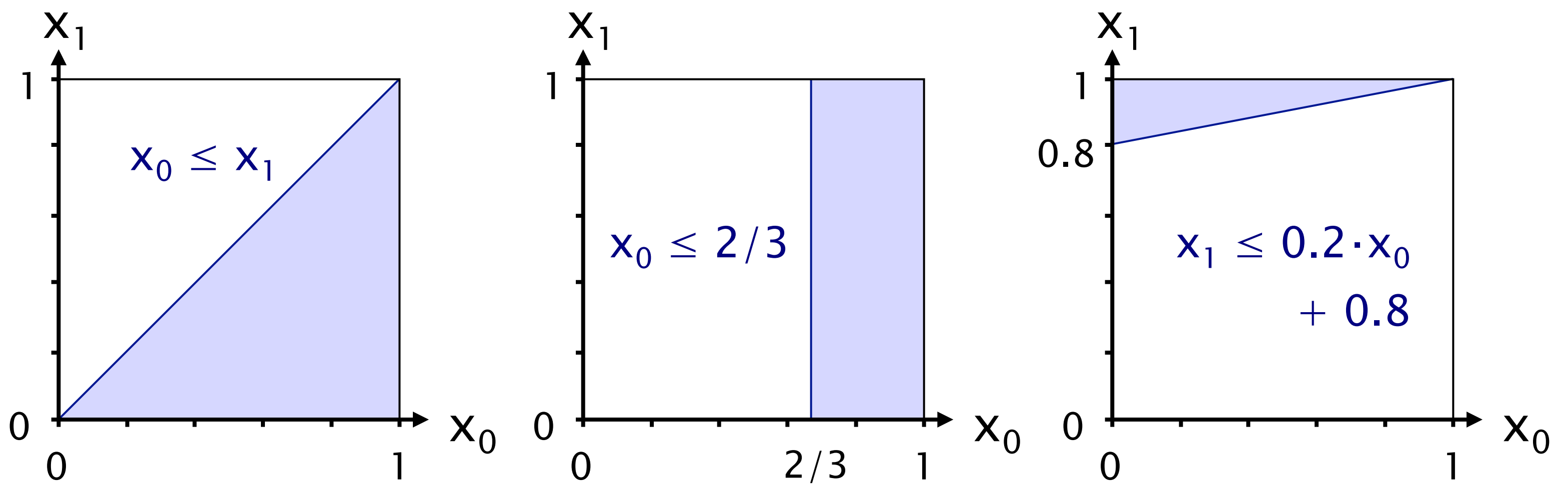
Let $x_i = Pr_{s_i}^{min}(F\ a)$

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

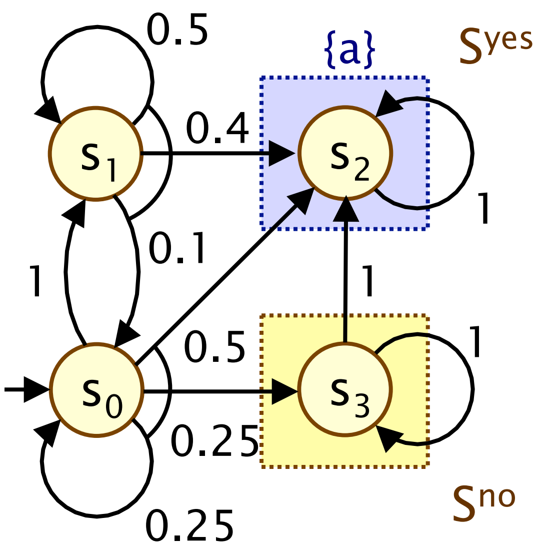
For $S^? = \{x_0, x_1\}$:

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)



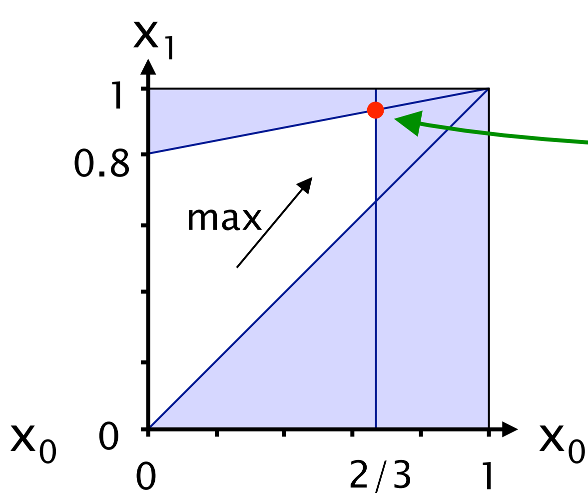
Let $x_i = Pr_{s_i}^{min}(F\ a)$

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

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

Maximise x_0+x_1 subject to constraints:

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



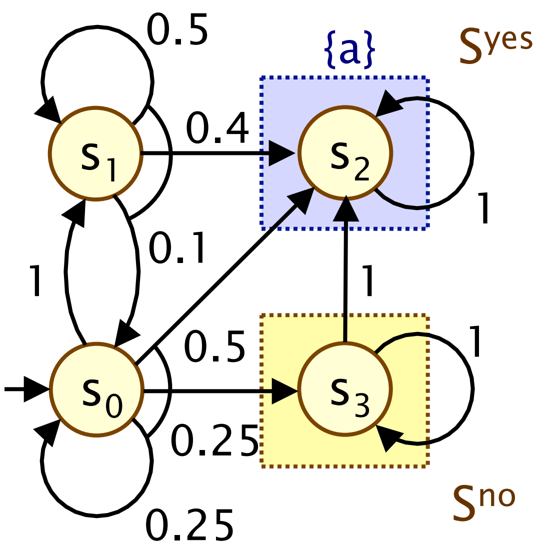
Solution:

(x_0, x_1)

$=$

$(2/3, 14/15)$

Example – PCTL until (LP)



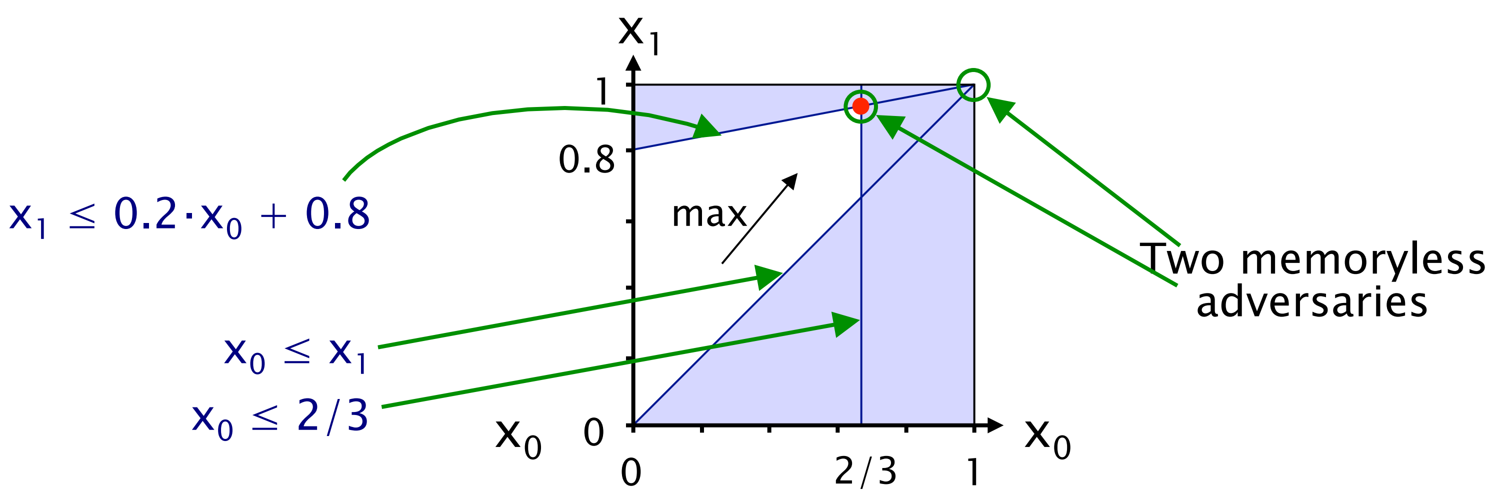
Let $x_i = Pr_{s_i}^{min}(F\ a)$

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

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

Maximise x_0+x_1 subject to constraints:

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



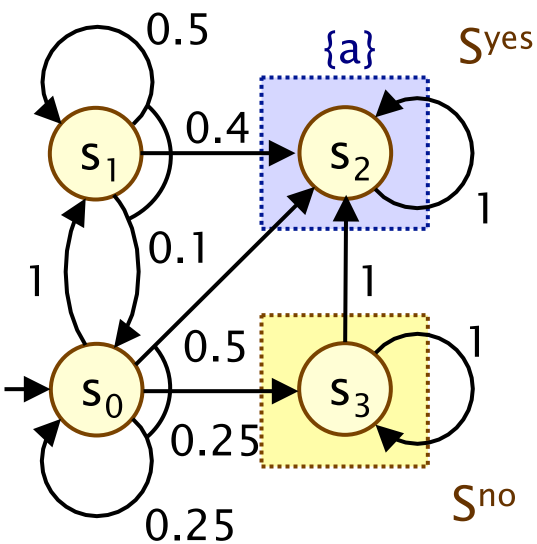
Method 2 – Value iteration

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

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

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

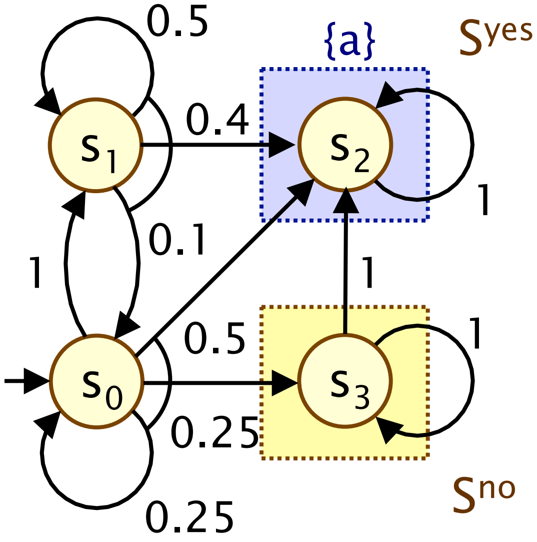
n=0: $[\ 0, 0, 1, 0\ ]$

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

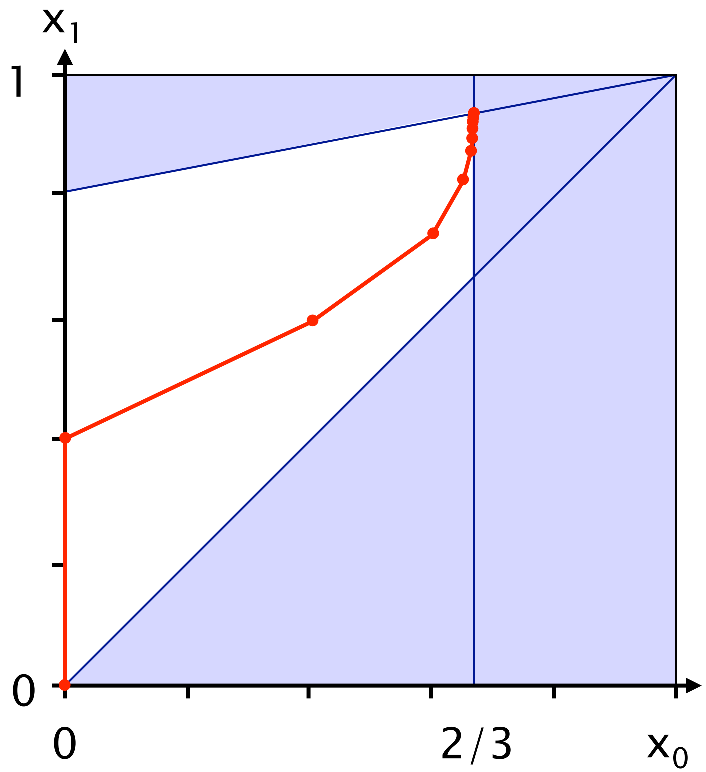
n=3: …

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]

Method 3 – Policy iteration

- Value iteration:
  - iterates over (vectors of) probabilities
- Policy iteration:
  - iterates over adversaries ("policies")

- 1. Start with an arbitrary (memoryless) adversary σ
- 2. Compute the reachability probabilities $\underline{Pr}^{\sigma}(F\ a)$ for σ
- 3. Improve the adversary in each state
- 4. Repeat 2/3 until no change in adversary

- Termination:
  - finite number of memoryless adversaries
  - improvement in (minimum) probabilities each time

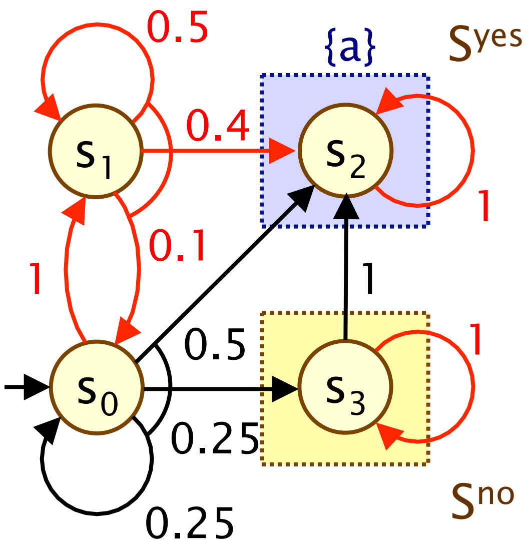
Method 3 – Policy iteration

- 1. Start with an arbitrary (memoryless) adversary σ
  - pick an element of $\delta(s)$ for each state $s \in S$
- 2. Compute the reachability probabilities $\underline{Pr}^{\sigma}(F\ a)$ for σ
  - probabilistic reachability on a DTMC
  - i.e. solve linear equation system
- 3. Improve the adversary in each state

$$\sigma'(s) = \text{argmin}\left\{ \sum_{s' \in S} \mu(s') \cdot Pr_{s'}^{\sigma}(F\ a) \mid (a, \mu) \in \delta(s) \right\}$$

- 4. Repeat 2/3 until no change in adversary

Example – Policy iteration



Arbitrary adversary σ:

Compute: $\underline{Pr}^{\sigma}(F\ a)$

Let $x_i = Pr_{s_i}^{\sigma}(F\ a)$

$x_2=1$, $x_3=0$ and:

- $x_0 = x_1$
- $x_1 = 0.1 \cdot x_0 + 0.5 \cdot x_1 + 0.4$

Solution:

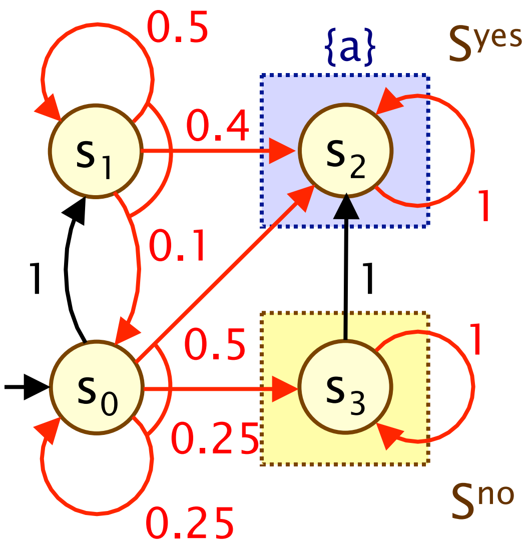
$\underline{Pr}^{\sigma}(F\ a) = [\ 1, 1, 1, 0\ ]$

Refine σ in state s_0:

$\min\{1(1), 0.5(1)+0.25(0)+0.25(1)\}$

$= \min\{1, 0.75\} = 0.75$

Example – Policy iteration



Refined adversary σ':

Compute: $\underline{Pr}^{\sigma'}(F\ a)$

Let $x_i = Pr_{s_i}^{\sigma'}(F\ a)$

$x_2=1$, $x_3=0$ and:

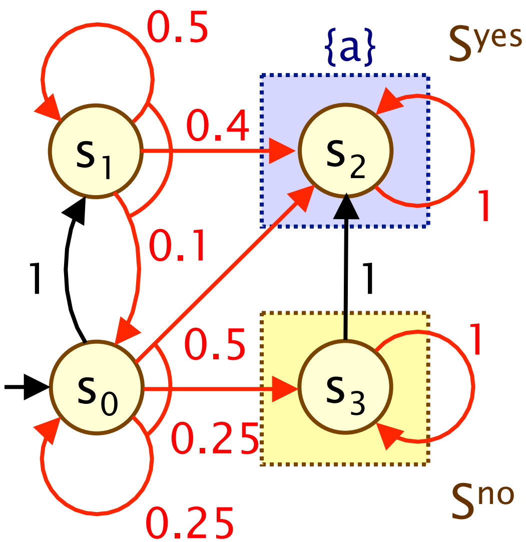
- $x_0 = 0.25 \cdot x_0 + 0.5$
- $x_1 = 0.1 \cdot x_0 + 0.5 \cdot x_1 + 0.4$

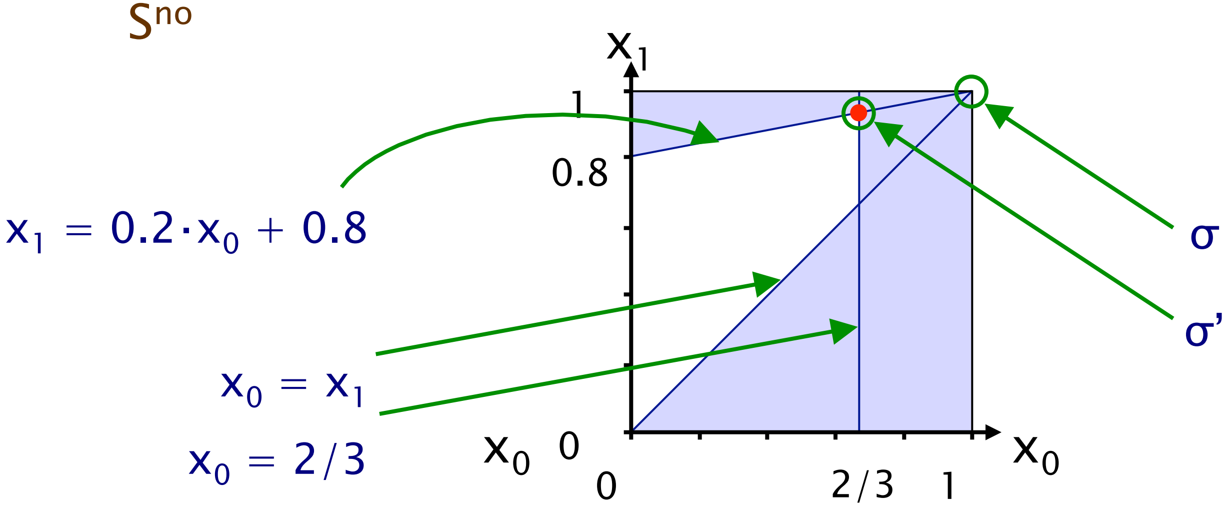
Solution:

$\underline{Pr}^{\sigma'}(F\ a) = [\ 2/3,\ 14/15,\ 1,\ 0\ ]$

This is optimal

Example – Policy iteration





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

Costs and rewards for MDPs

- We can augment MDPs 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

- Extend logic PCTL with R operator, for "expected reward"
  - as for PCTL, either $R_{\sim r}$ […], $R_{min=?}$ […] or $R_{max=?}$ […]
- Some examples:
  - $R_{min=?}$ [$I^{=90}$], $R_{max=?}$ [$C^{\leq 60}$], $R_{max=?}$ [F "end"]
  - "the minimum expected queue size after exactly 90 seconds"
  - "the maximum expected power consumption over one hour"
  - the maximum expected time for the algorithm to terminate

Overview (Part 1)

- Markov decision processes (MDPs)
- Adversaries & probability spaces
- Properties of MDPs: The temporal logic PCTL
- PCTL model checking for MDPs
- Case study: Firewire root contention

The PRISM tool

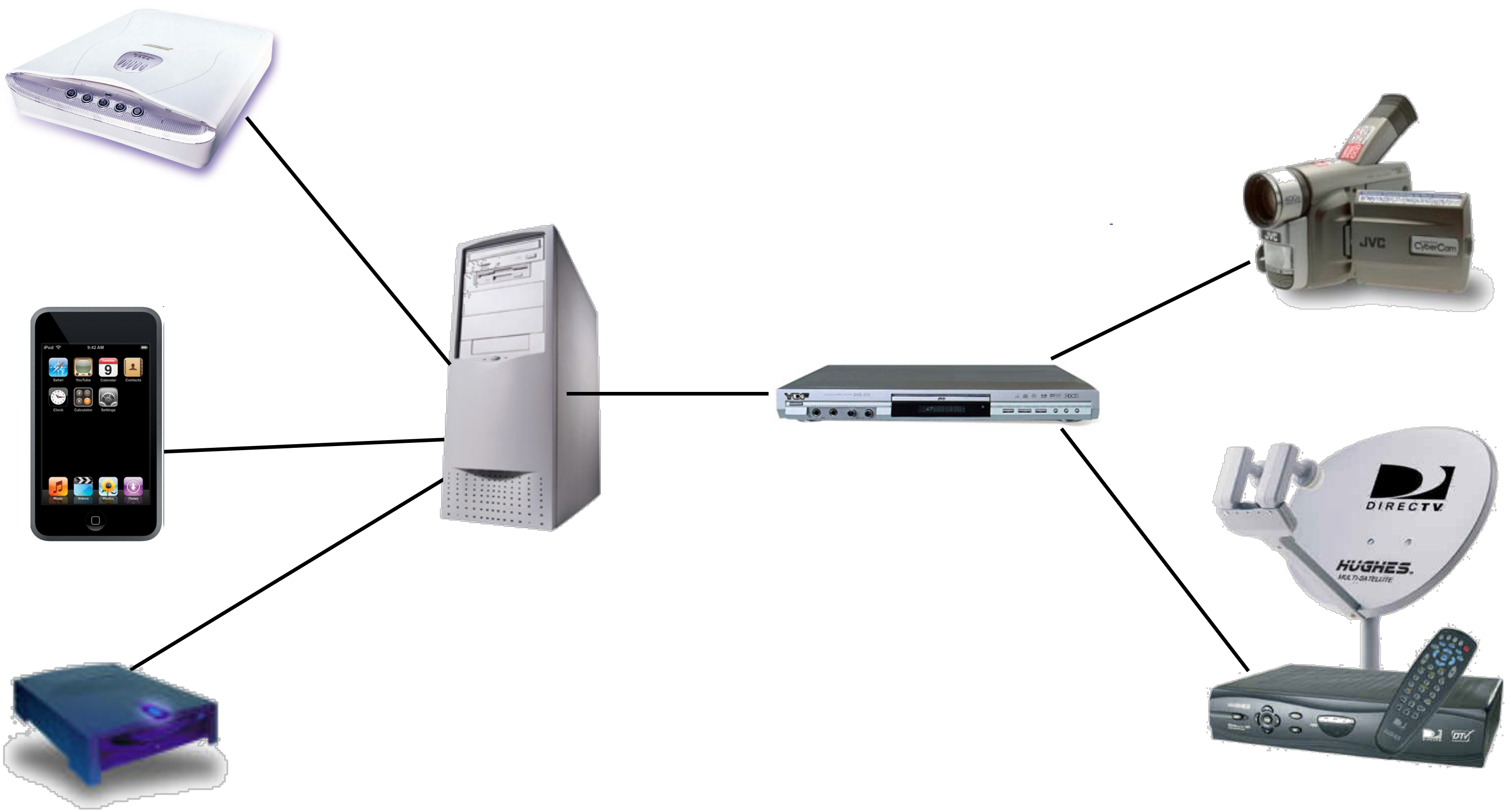
- PRISM: Probabilistic symbolic model checker
  - developed at Birmingham/Oxford University, since 1999
  - free, open source (GPL), runs on all major OSs
- Support for:
  - discrete-/continuous-time Markov chains (D/CTMCs)
  - Markov decision processes (MDPs)
  - probabilistic timed automata (PTAs)
  - PCTL, CSL, LTL, PCTL*, costs/rewards, …
- Multiple efficient model checking engines
  - mostly symbolic (BDDs) (up to 10^{10} states, 10^7–10^8 on avg.)
- Successfully applied to a wide range of case studies
  - communication protocols, security protocols, dynamic power management, cell signalling pathways, …
  - http://www.prismmodelchecker.org/

Case study: FireWire protocol

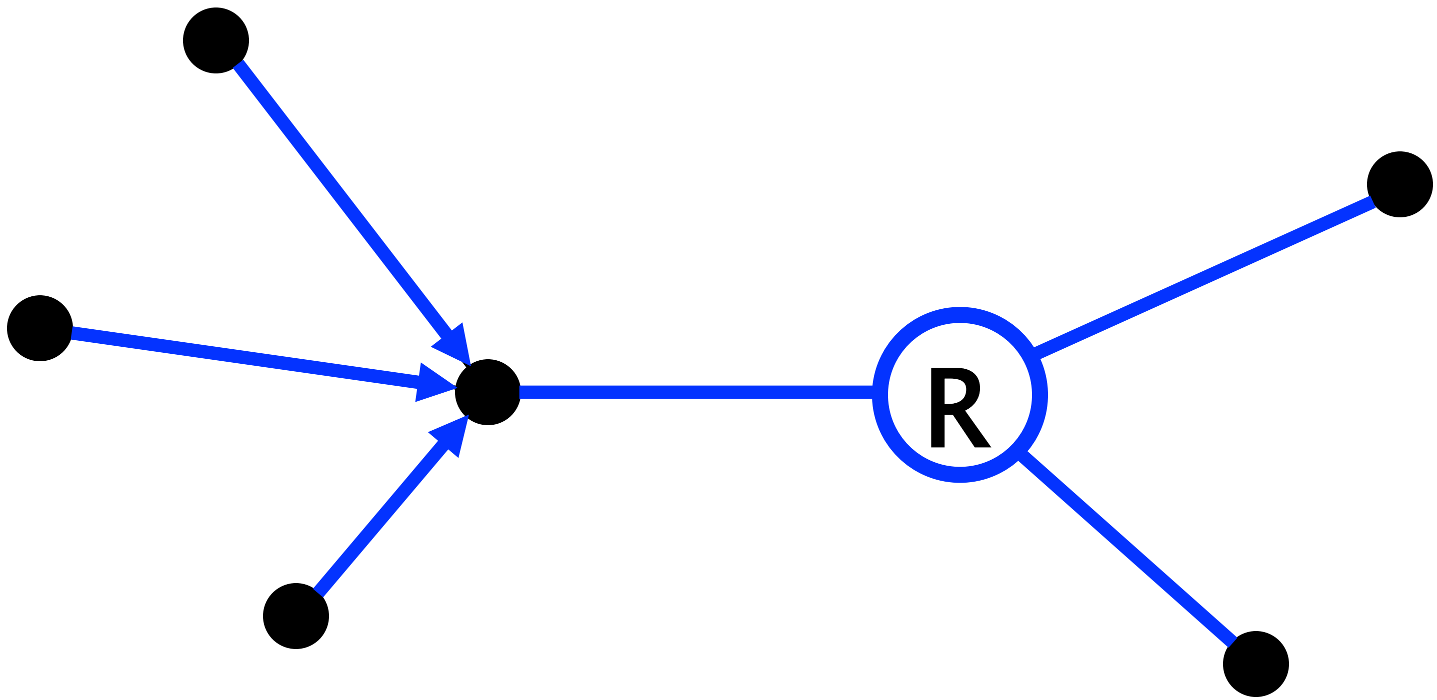
- FireWire (IEEE 1394)
  - high-performance serial bus for networking multimedia devices; originally by Apple
  - "hot-pluggable" - add/remove devices at any time
  - no requirement for a single PC (need acyclic topology)

- Root contention protocol
  - leader election algorithm, when nodes join/leave
  - symmetric, distributed protocol
  - uses electronic coin tossing and timing delays
  - nodes send messages: "be my parent"
  - root contention: when nodes contend leadership
  - random choice: "fast"/"slow" delay before retry

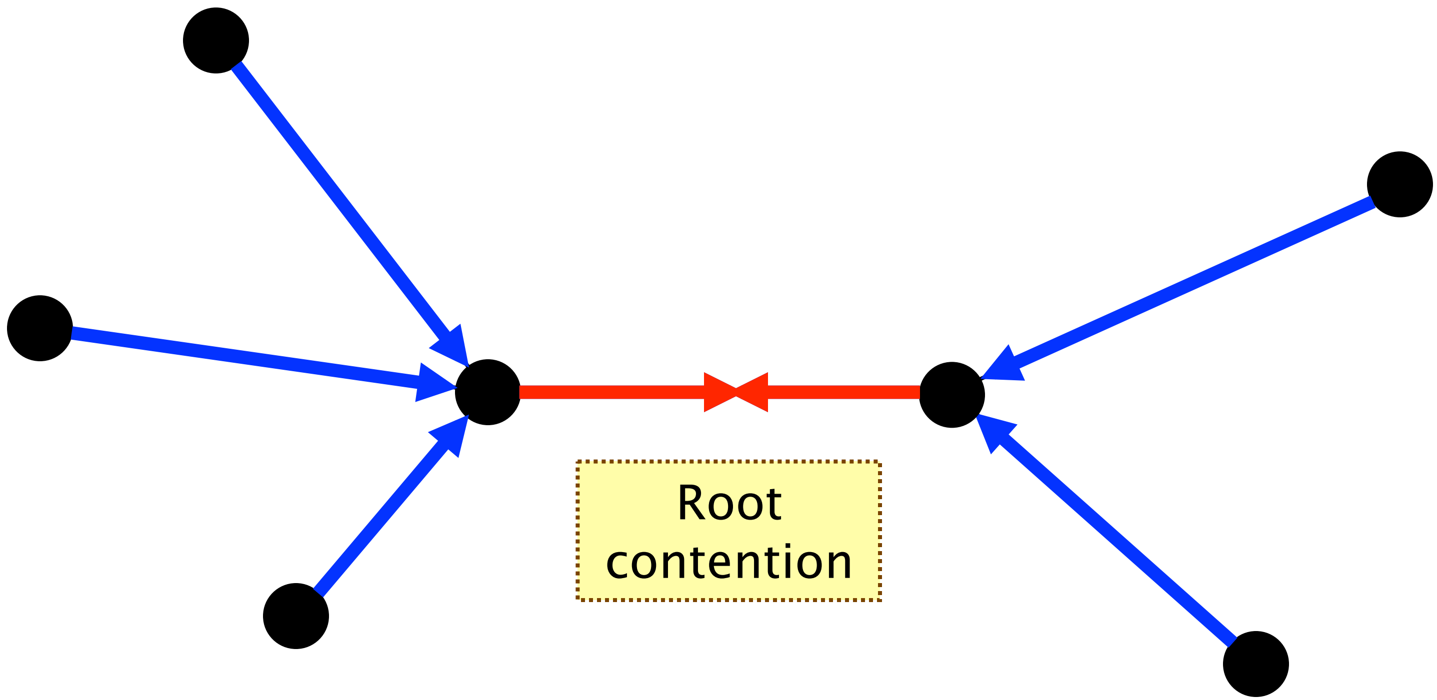
FireWire example



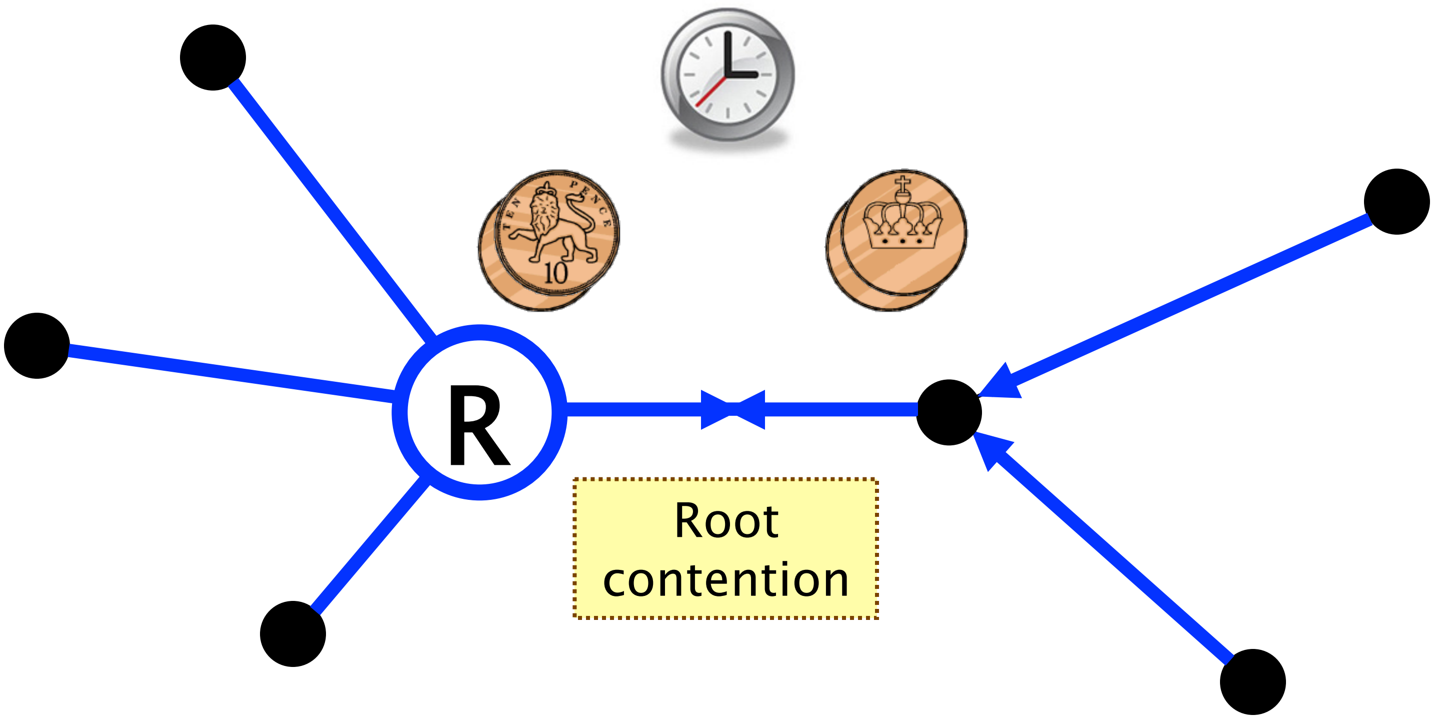
FireWire leader election



FireWire root contention

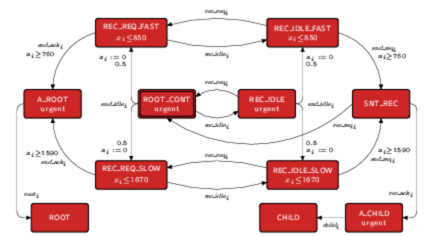


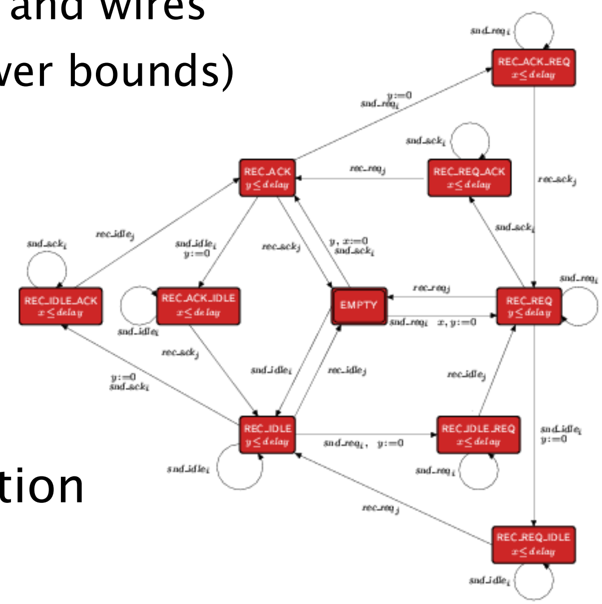
FireWire root contention



FireWire analysis

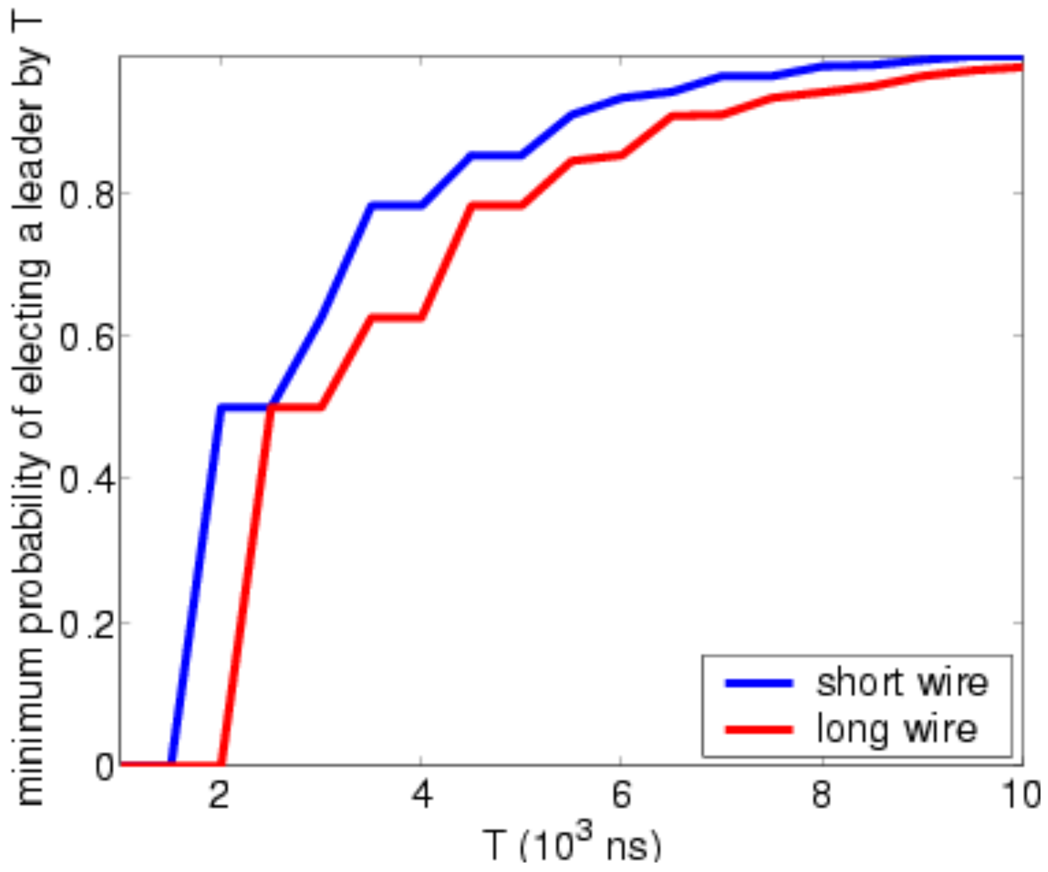
- Probabilistic model checking
  - model constructed and analysed using PRISM
  - timing delays taken from standard
  - model includes:
    - concurrency: messages between nodes and wires
    - underspecification of delays (upper/lower bounds)
  - max. model size: 170 million states





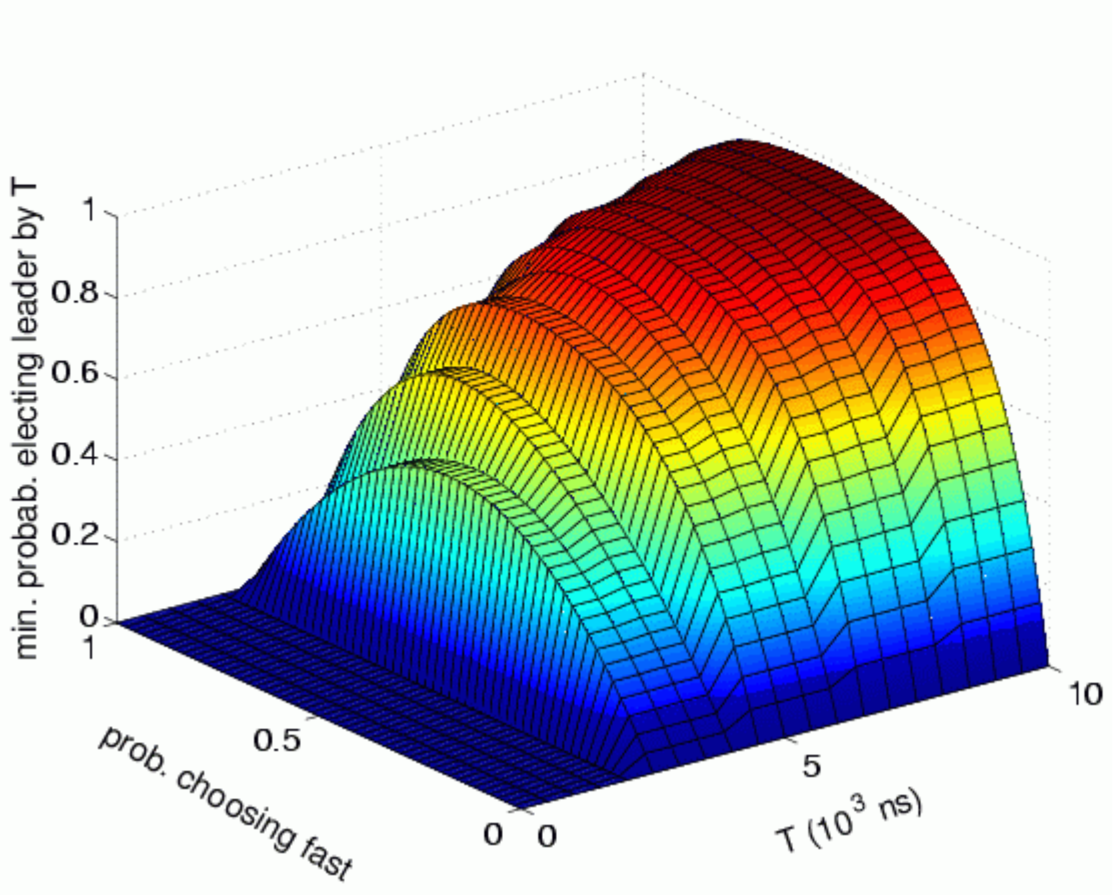
- Analysis:
  - verified that root contention always resolved with probability 1
  - investigated time taken for leader election
  - and the effect of using biased coin
    - based on a conjecture by Stoelinga

FireWire: Analysis results



"minimum probability of electing leader by time T"

FireWire: Analysis results

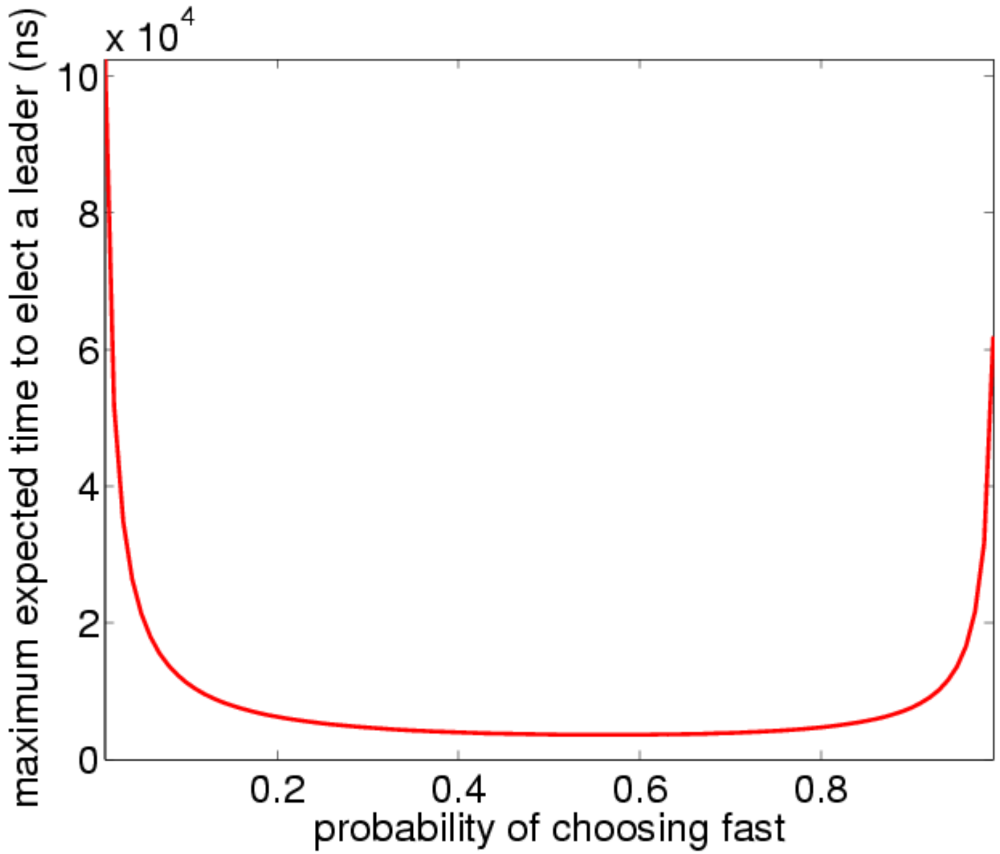


"minimum probability of electing leader by time T"

(short wire length)

Using a biased coin

FireWire: Analysis results

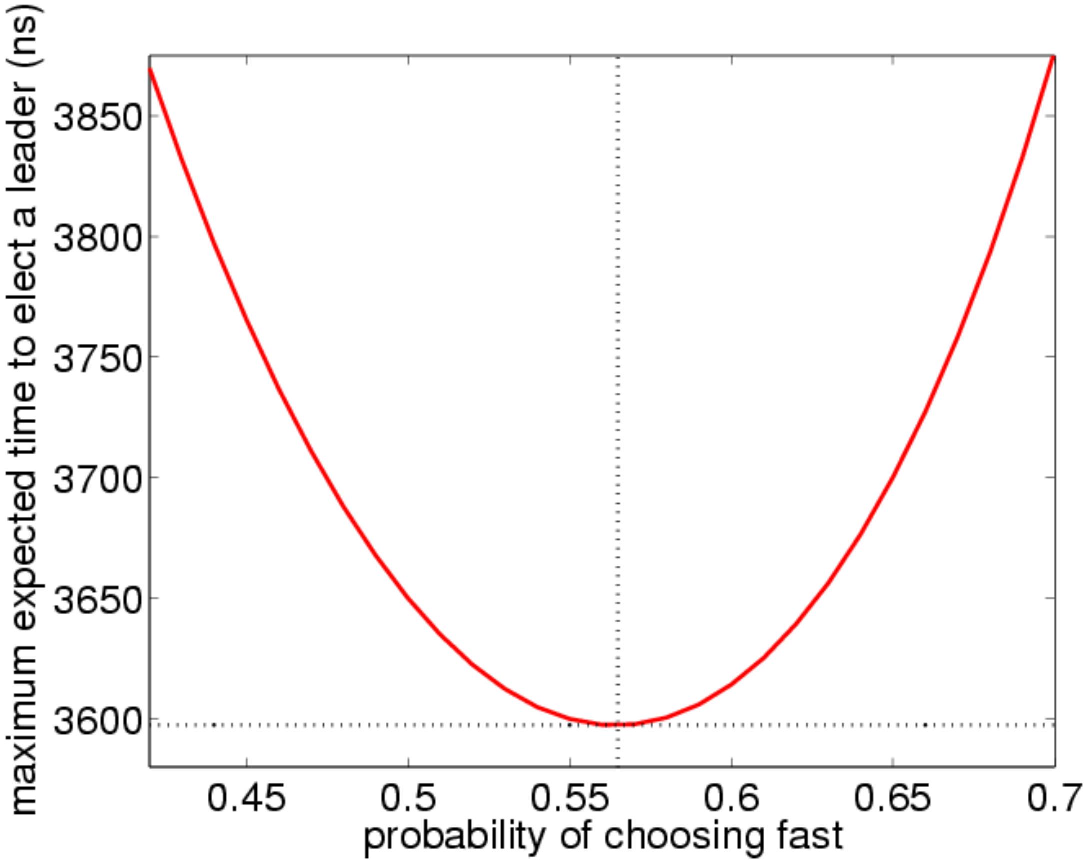


"maximum expected time to elect a leader"

(short wire length)

Using a biased coin

FireWire: Analysis results



"maximum expected time to elect a leader"

(short wire length)

Using a biased coin is beneficial!

Summary (Part 1)

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

- Next: LTL model checking (for DTMCs and MDPs)