AIMS Systems Verification
Quantitative Verification Part 2

Prof. Marta Kwiatkowska

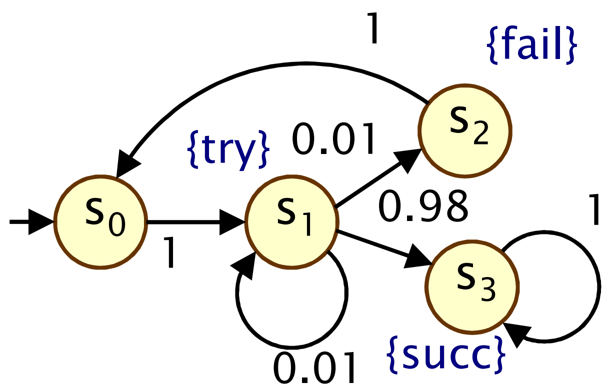
Department of Computer Science
University of Oxford

Overview (Part 2)

- Markov decision processes (MDPs)
  - MDPs: definition
  - Paths, strategies & probability spaces
- PCTL model checking
- Costs and rewards
- Case study: Firewire root contention
- Strategy synthesis for MDPs
  - Properties and objectives
  - Verification vs synthesis
- Case study: Dynamic power management
- Summary

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:

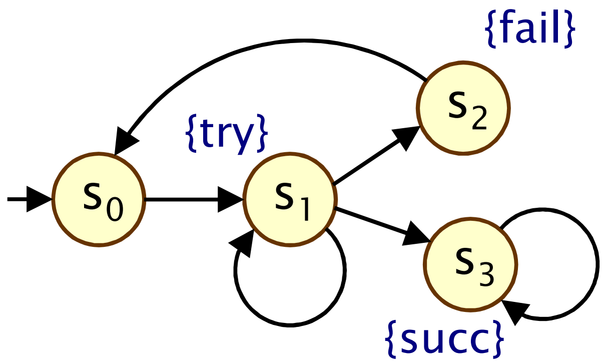
- Concurrency – scheduling of parallel components
  - e.g. randomised distributed algorithms – multiple probabilistic processes operating asynchronously

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

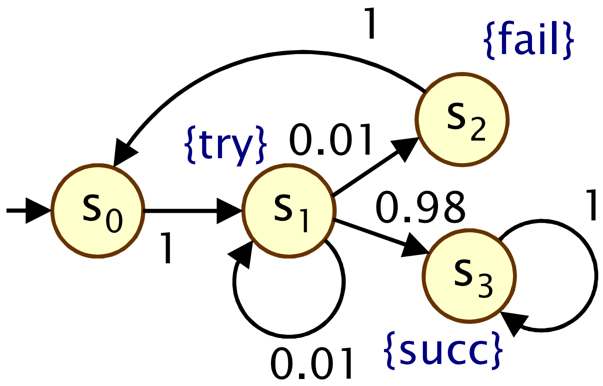
- Unknown environments
  - e.g. probabilistic security protocols – unknown adversary

Probability vs. nondeterminism

- Labelled transition system
  - (S,s_0,R,L) where $R \subseteq S \times S$
  - choice is nondeterministic



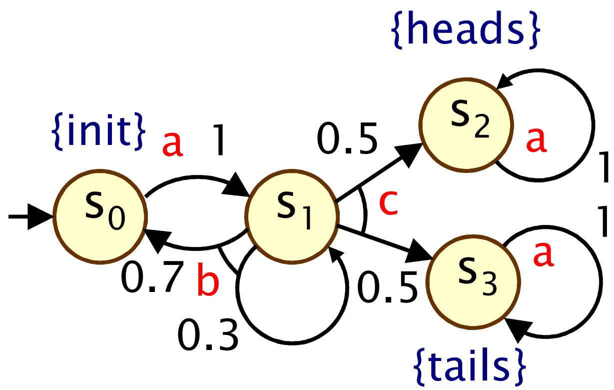
- Discrete-time Markov chain
  - (S,s_0,P,L) where $P : S \times S \rightarrow [0,1]$
  - choice is probabilistic



- How to combine?

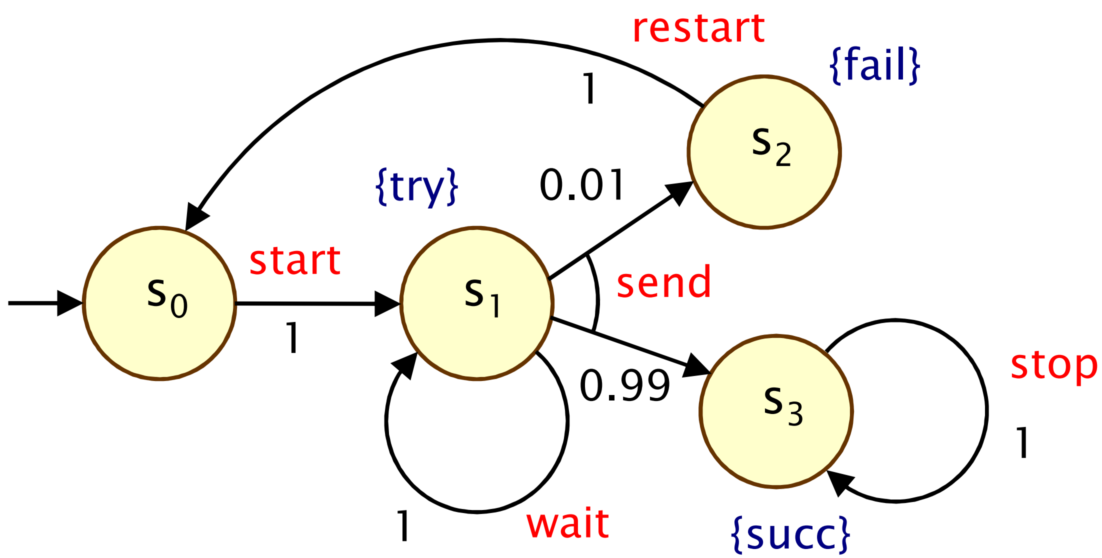
Markov decision processes

- 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



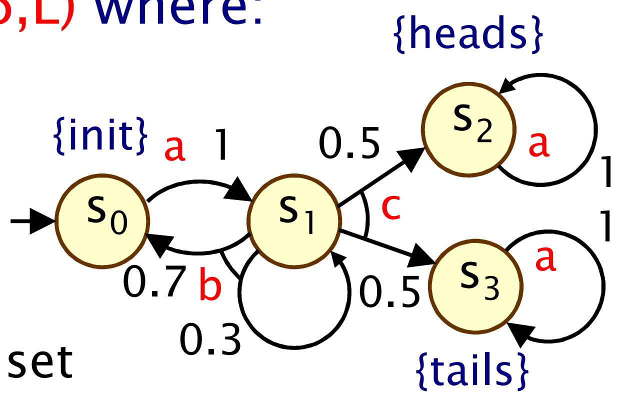
Simple MDP example

- 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



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
  - $\delta \subseteq S \times \alpha \times \mathrm{Dist}(S)$ is the transition probability relation, where Dist(S) is the set of all discrete probability distributions over S
  - $L : S \to 2^{AP}$ is a labelling with atomic propositions



- Notes:
  - we also abuse notation and use δ as a function
  - i.e. $\delta : S \to 2^{\alpha \times \mathrm{Dist}(S)}$ 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]
    - usually, MDPs take the form: $\delta : S \times \alpha \to \mathrm{Dist}(S)$

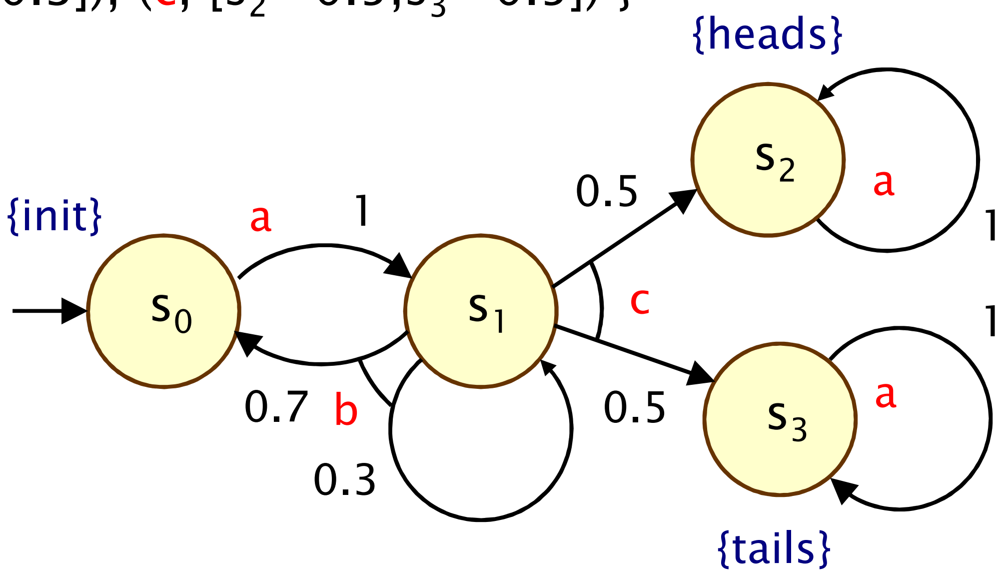
Simple MDP example 2

$M = (S, s_{init}, Steps, L)$

$S = \{s_0, s_1, s_2, s_3\}$
$s_{init} = s_0$

$AP = \{init, heads, tails\}$
$L(s_0) = \{init\}$,
$L(s_1) = \varnothing$,
$L(s_2) = \{heads\}$,
$L(s_3) = \{tails\}$

$Steps(s_0) = \{ (a, [s_1 \mapsto 1]) \}$
$Steps(s_1) = \{ (b, [s_0 \mapsto 0.7, s_1 \mapsto 0.3]), (c, [s_2 \mapsto 0.5, s_3 \mapsto 0.5]) \}$
$Steps(s_2) = \{ (a, [s_2 \mapsto 1]) \}$
$Steps(s_3) = \{ (a, [s_3 \mapsto 1]) \}$



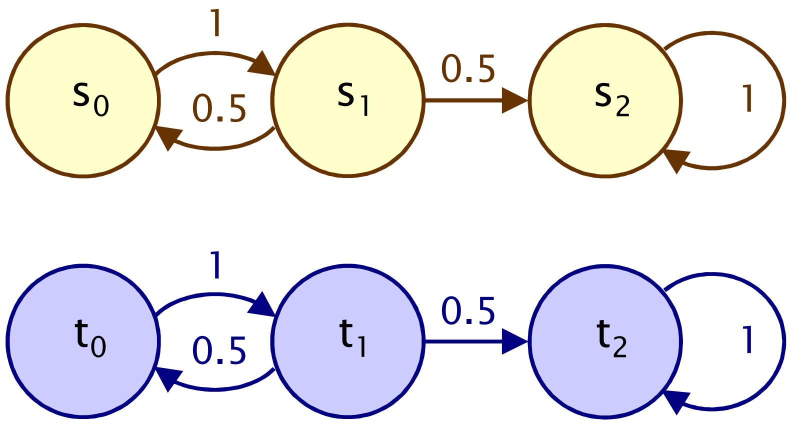
Example - Parallel composition

Asynchronous parallel composition of two 3-state DTMCs

PRISM code:

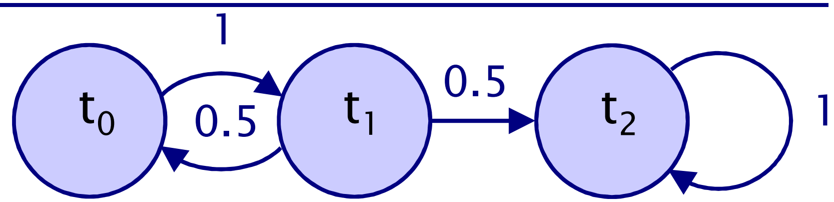
```
module M1
    s : [0..2] init 0;
    [] s=0 -> (s'=1);
    [] s=1 -> 0.5:(s'=0) + 0.5:(s'=2);
    [] s=2 -> (s'=2);
endmodule
```

```
module M2 = M1 [ s=t ] endmodule
```

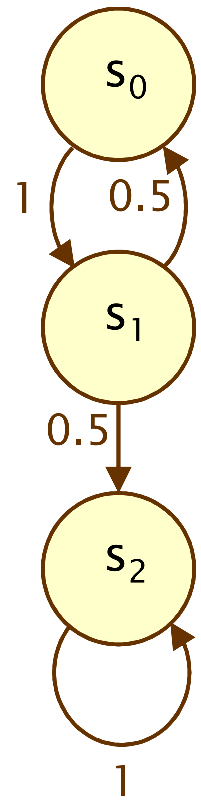


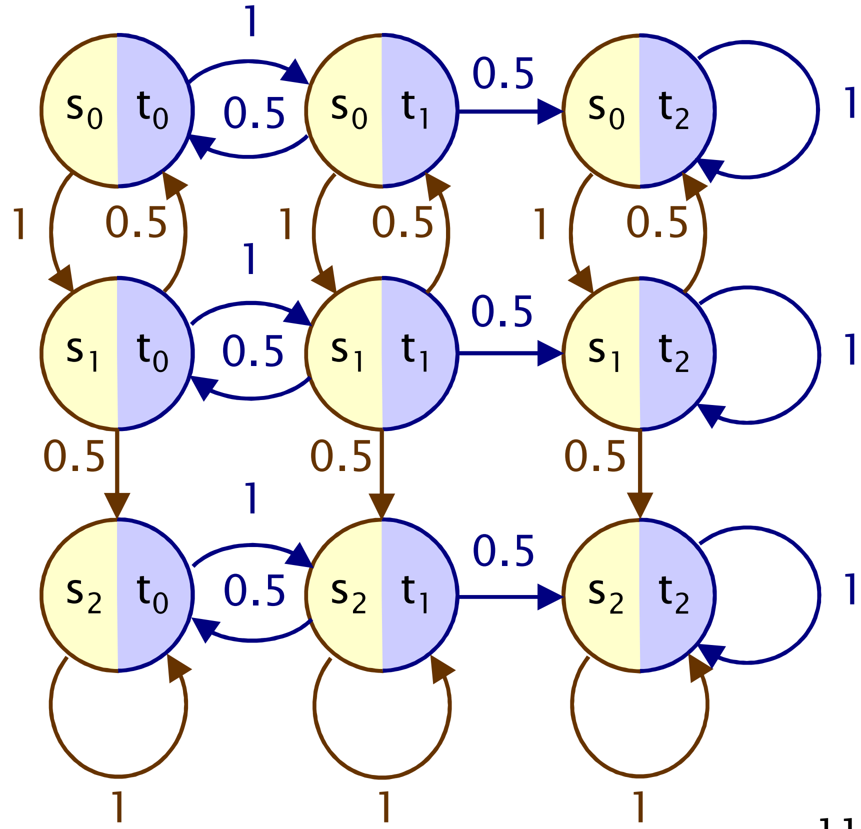
Example – Parallel composition

Asynchronous parallel composition of two 3-state DTMCs



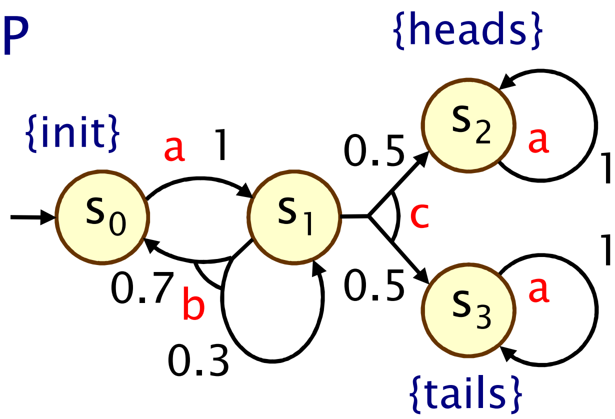
Action labels omitted here





Paths and strategies

- A (finite or infinite) path through an MDP
  - is a sequence of (connected) states
  - e.g. $s_0(a_0,\mu_0)s_1(a_1,\mu_1)s_2\ldots$
  - represents an execution of the system
  - resolves both the probabilistic and nondeterministic choices



- A strategy σ (aka. "adversary" or "policy") of an MDP
  - is a resolution of nondeterminism only
  - is (formally) a mapping from finite paths to distributions on action-distribution pairs
  - induces a fully probabilistic model
  - i.e. an (infinite-state) Markov chain over finite paths
  - on which we can define a probability space over infinite paths

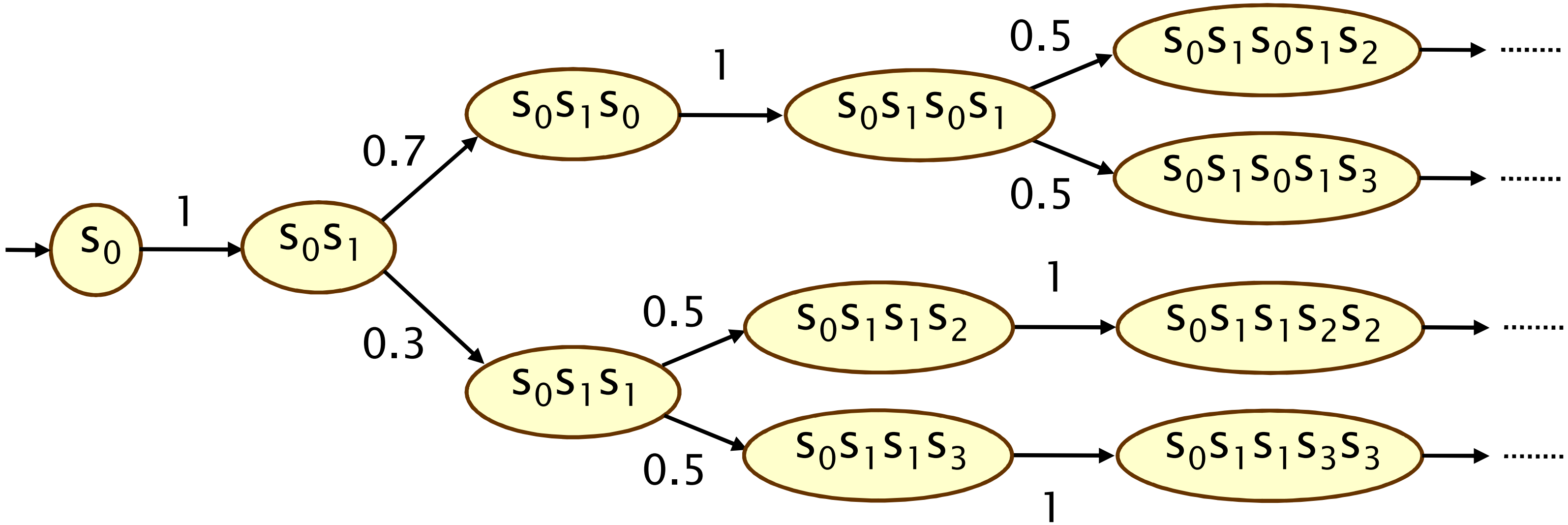
Classification of strategies

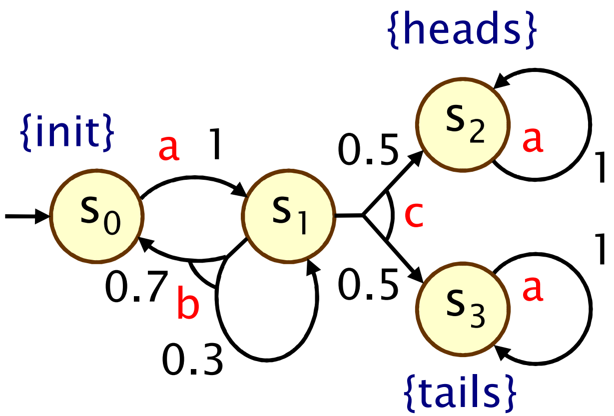
- Strategies are classified according to
- randomisation:
  - σ is deterministic (pure) if $\sigma(s_0 \ldots s_n)$ is a point distribution, and randomised otherwise
- memory:
  - σ is memoryless (simple) if $\sigma(s_0 \ldots s_n) = \sigma(s_n)$ for all $s_0 \ldots s_n$
  - σ is finite memory if there are finitely many modes such that $\sigma(s_0 \ldots s_n)$ depends only on s_n and the current mode, which is updated each time an action is performed
  - otherwise, σ is infinite memory

- A strategy σ induces, for each state s in the MDP:
  - a set of infinite paths $Path^\sigma(s)$
  - a probability space Pr^σ_s over $Path^\sigma(s)$

Example strategy

- Fragment of induced Markov chain for strategy which picks b then c in s_1





finite-memory,
deterministic

Induced DTMCs

- Strategy σ for MDP induces an infinite-state DTMC D^σ
- $D^\sigma = (Path^\sigma_{fin}(s), s, P^\sigma_s)$ where:
  - 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)
  - $P^\sigma_s(\omega,\omega')=\mu(s')$ if $\omega' = \omega(a, \mu)s'$ and $\sigma(\omega)=(a,\mu)$
  - $P^\sigma_s(\omega,\omega')=0$ otherwise
- 1-to-1 correspondence between $Path^\sigma(s)$ and paths of D^σ
- This gives us a probability measure Pr^σ_s over $Path^\sigma(s)$
  - from probability measure over paths of D^σ

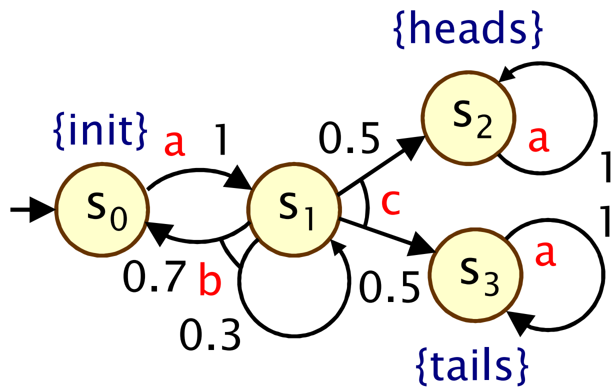
MDPs and probabilities

- $\text{Prob}^\sigma(s, \psi) = \text{Pr}^\sigma_s \{ \omega \in \text{Path}^\sigma(s) \mid \omega \vDash \psi \}$
  - for some path formula ψ
  - e.g. $\text{Prob}^\sigma(s, \text{F tails})$

- MDP provides best-/worst-case analysis
  - based on lower/upper bounds on probabilities
  - over all possible adversaries

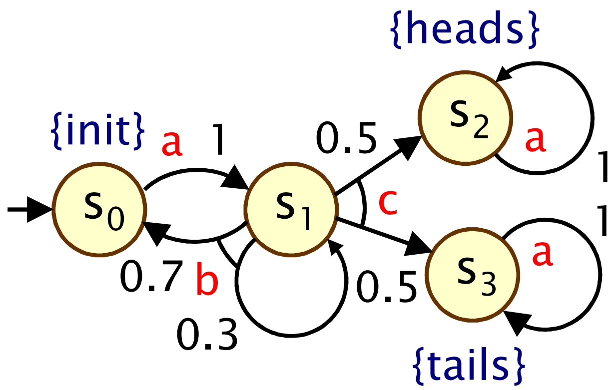
$$p_{min}(s,\psi) = \inf_{\sigma\in Adv} \text{Prob}^\sigma(s,\psi)$$

$$p_{max}(s,\psi) = \sup_{\sigma\in Adv} \text{Prob}^\sigma(s,\psi)$$

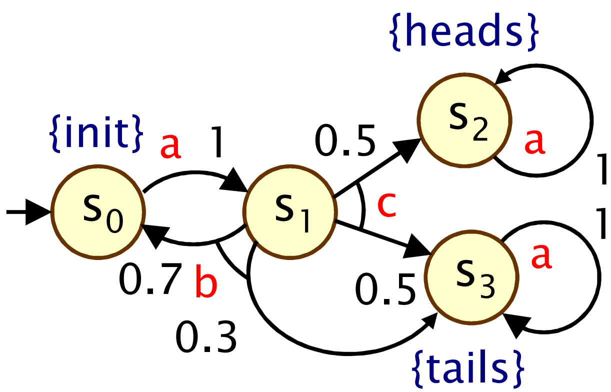


Examples

- $Prob^{\sigma 1}(s_0, F\ tails) = 0.5$
- $Prob^{\sigma 2}(s_0, F\ tails) = 0.5$
  - (where σ_i picks b i−1 times then c)
- …
- $p_{max}(s_0, F\ tails) = 0.5$
- $p_{min}(s_0, F\ tails) = 0$

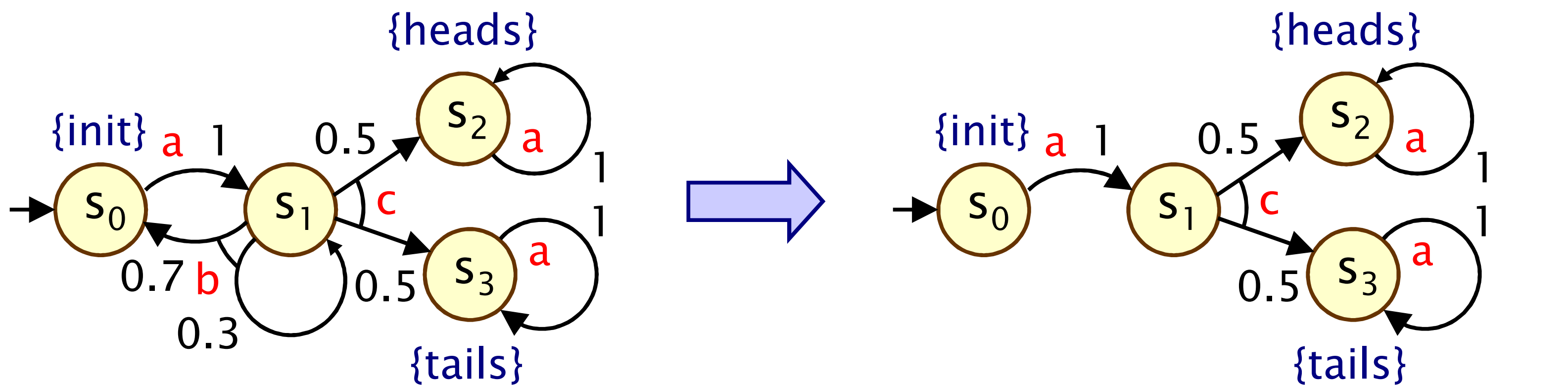


- $Prob^{\sigma 1}(s_0, F\ tails) = 0.5$
- $Prob^{\sigma 2}(s_0, F\ tails)$
  $= 0.3+0.7 \cdot 0.5 = 0.65$
- $Prob^{\sigma 3}(s_0, F\ tails)$
  $= 0.3+0.7 \cdot 0.3+0.7 \cdot 0.7 \cdot 0.5 = 0.755$
- …
- $p_{max}(s_0, F\ tails) = 1$
- $p_{min}(s_0, F\ tails) = 0.5$



Memoryless strategies

- Memoryless strategies always pick same choice in a state
  - also known as: positional, Markov, simple
  - formally, $\sigma(s_0(a_0,\mu_0)s_1 \ldots s_n)$ depends only on s_n
  - can write as a mapping from states, i.e. $\sigma(s)$ for each $s \in S$
  - induced 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



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:
  - $\phi \ ::= \ \mathrm{true} \mid a \mid \phi \wedge \phi \mid \neg\phi \mid P_{\sim p} [\ \psi\ ]$ (state formulas)
  - $\psi \ ::= \ X\ \phi \mid \phi\ U^{\leq k}\ \phi \mid \phi\ U\ \phi$ (path formulas)
  - 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: $\mathrm{send} \rightarrow P_{\geq 0.95}\ [\ \mathrm{true}\ U^{\leq 10}\ \mathrm{deliver}\ ]$

PCTL semantics for MDPs

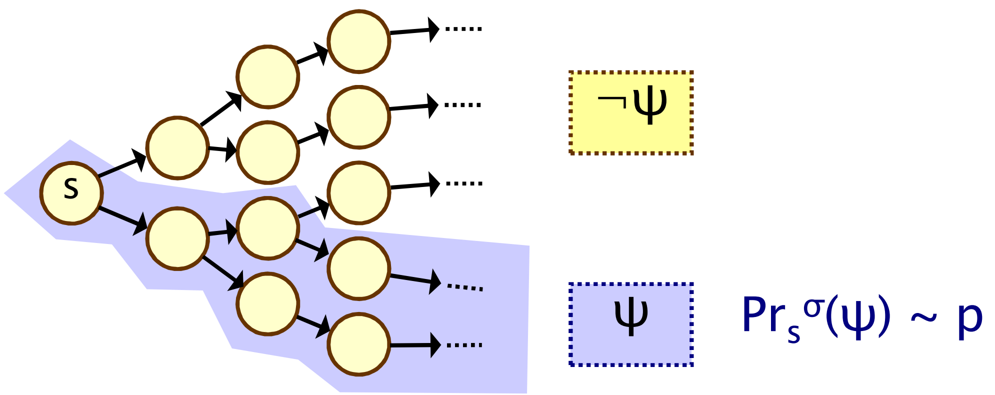
- PCTL formulas interpreted over states of an MDP
  - $s \vDash \phi$ denotes ϕ is "true in state s" or "satisfied in state s"

- Semantics of (non-probabilistic) state formulas:
  - for a state s of the MDP $(S,s_{init},\alpha,\delta,L)$:
  - $s \vDash a \quad \Leftrightarrow \quad a \in L(s)$
  - $s \vDash \phi_1 \wedge \phi_2 \quad \Leftrightarrow \quad s \vDash \phi_1$ and $s \vDash \phi_2$
  - $s \vDash \neg\phi \quad \Leftrightarrow \quad s \vDash \phi$ 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:
  - $\omega \vDash X\,\phi \quad \Leftrightarrow \quad s_1 \vDash \phi$
  - $\omega \vDash \phi_1\ U^{\leq k}\ \phi_2 \quad \Leftrightarrow \quad \exists i \leq k$ 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 $\omega \vDash \phi_1\ U^{\leq k}\ \phi_2$

PCTL semantics for MDPs

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



- 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")

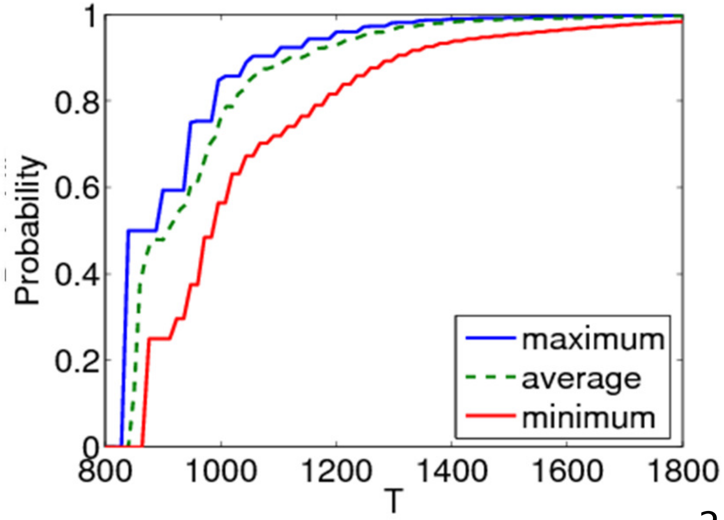
Minimum and maximum probabilities

- 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 strategies of either:
  - the minimum probability of ψ holding
  - the maximum probability of ψ holding
- Crucial result for model checking PCTL on MDPs
  - memoryless strategies suffice, i.e. there are always memoryless strategies 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 $Pr_s^{min}(ψ)$ or $Pr_s^{max}(ψ)$ 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
  - $P_{min=?}$ [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?"

PCTL model checking for MDPs

- Algorithm for PCTL model checking [BdA95]
  - inputs: MDP $M=(S,s_{init},\alpha,\delta,L)$, PCTL formula ϕ
  - output: $Sat(\phi) = \{ s \in S \mid s \vDash \phi \}$ = 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 $Pr_s^{min}(\psi)$ 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

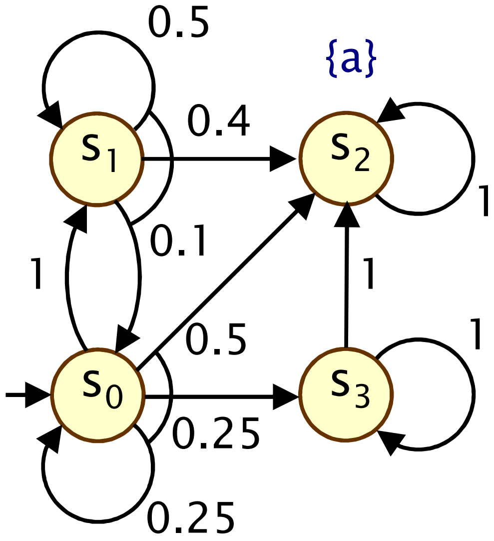
- Computation of probabilities $Pr_s^{min}(\phi_1 \text{ U } \phi_2)$ for all $s \in S$
- First identify all states where the probability is 1 or 0
  - "precomputation" algorithms, yielding sets S^{yes}, S^{no}
- Then compute (min) probabilities for remaining states ($S^?$)
  - either: solve linear programming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

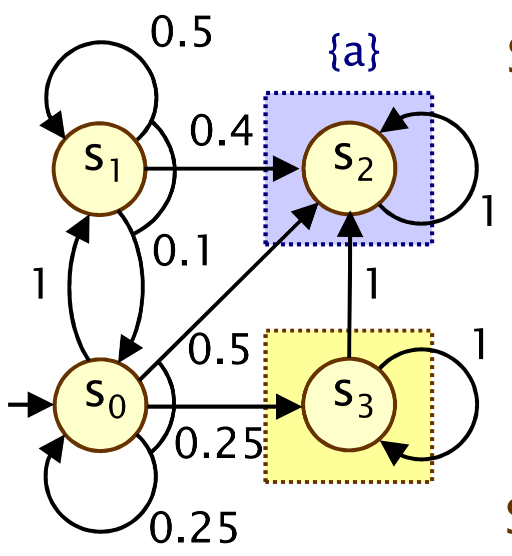


PCTL until – Precomputation

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

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

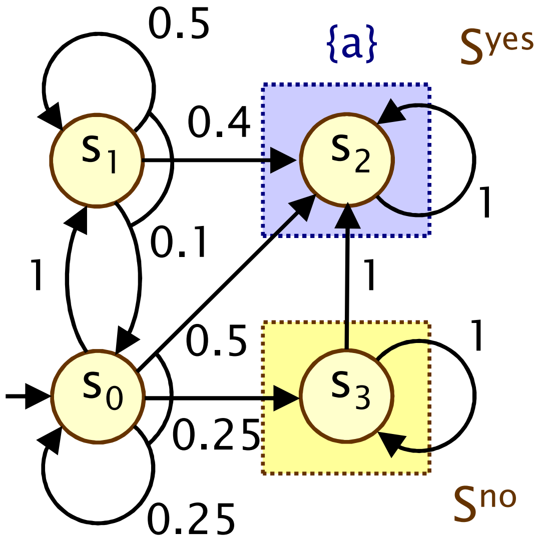
$$x_s \leq \sum_{s' \in S^?} \mu(s') \cdot x_{s'} + \sum_{s' \in S^{yes}} \mu(s')$$

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

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

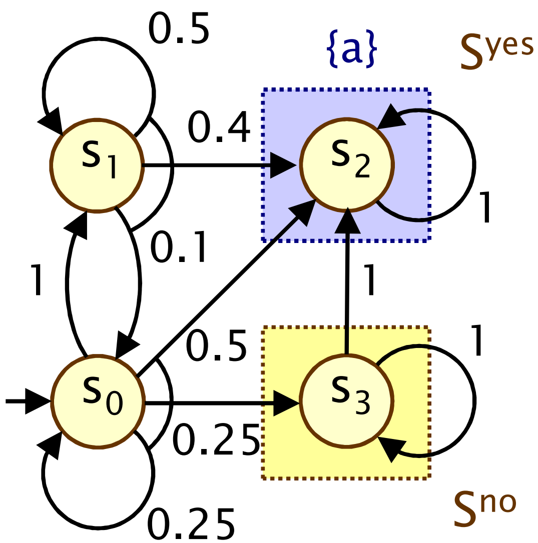
S^{yes}: $x_2=1$, S^{no}: $x_3=0$

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)



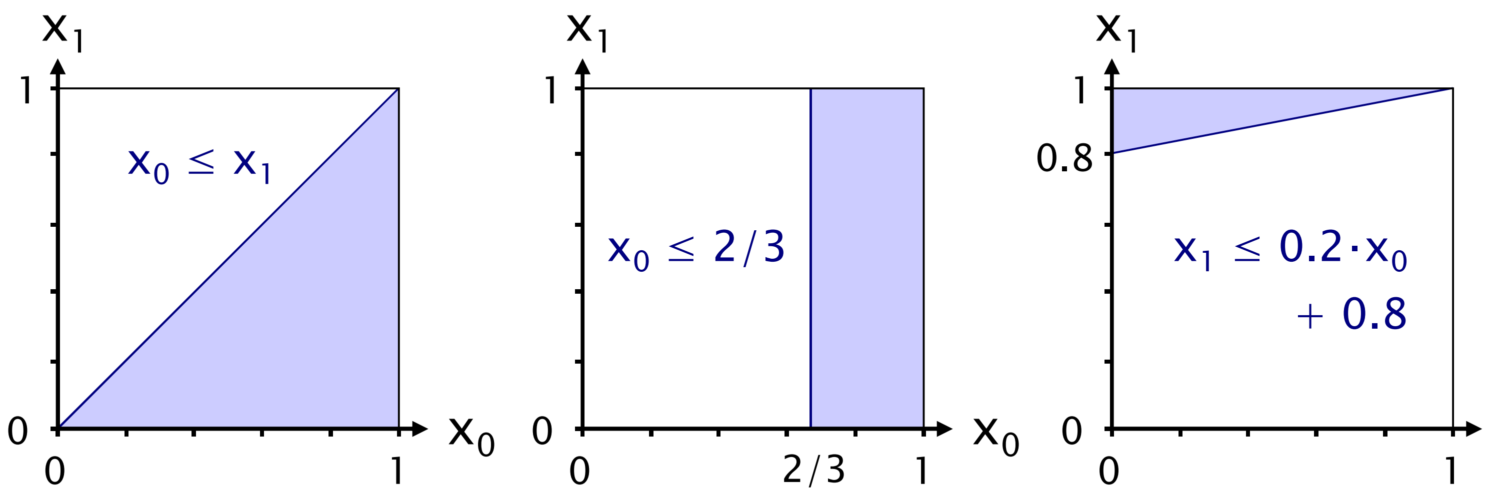
Let $x_i = Pr_{s_i}^{min}(F\ a)$

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

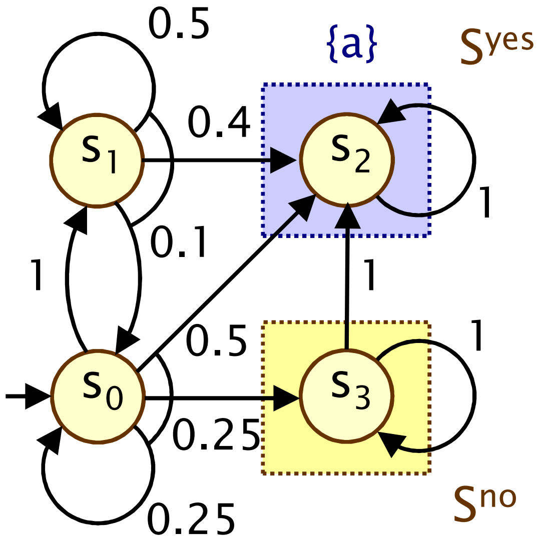
For $S^? = \{x_0, x_1\}$:

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)



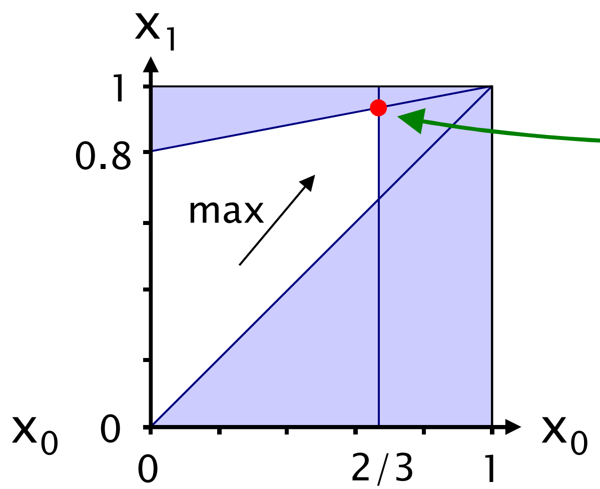
Let $x_i = Pr_{s_i}^{min}(F\ a)$

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

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

Maximise x_0+x_1 subject to constraints:

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



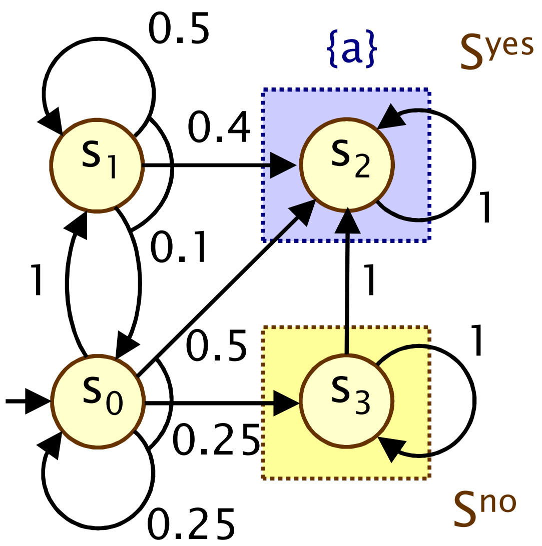
Solution:

(x_0, x_1)

$=$

$(2/3, 14/15)$

Example – PCTL until (LP)



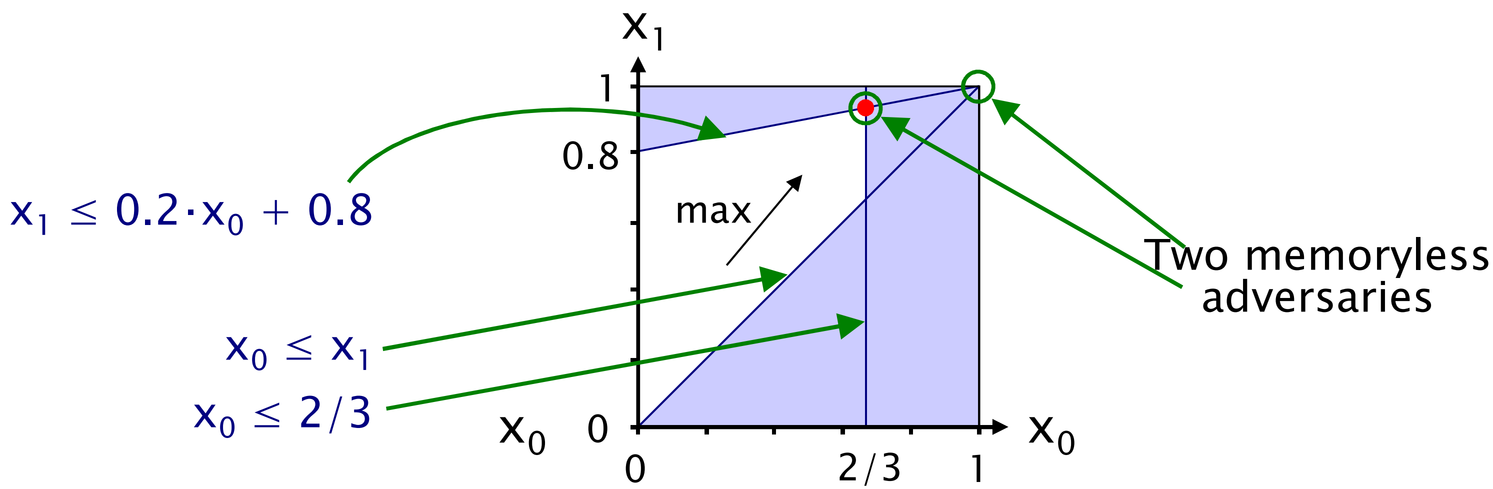
Let $x_i = Pr_{s_i}^{min}(F\ a)$

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

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

Maximise x_0+x_1 subject to constraints:

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



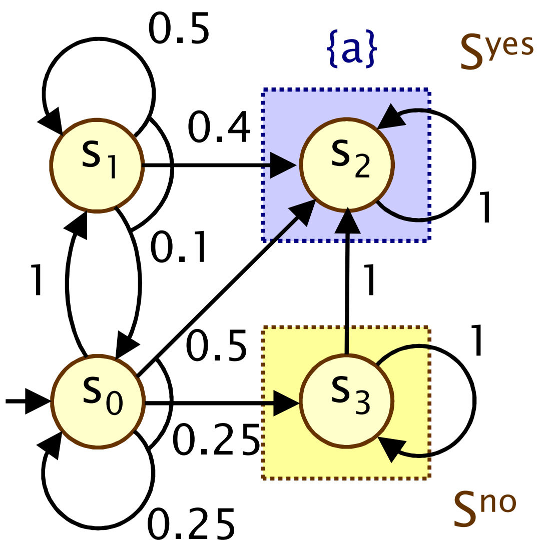
Method 2 – Value iteration

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

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

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

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

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

$0.1 \cdot 0 + 0.5 \cdot 0 + 0.4,\ 1,\ 0\ ]$

$= [\ 0, 0.4, 1, 0\ ]$

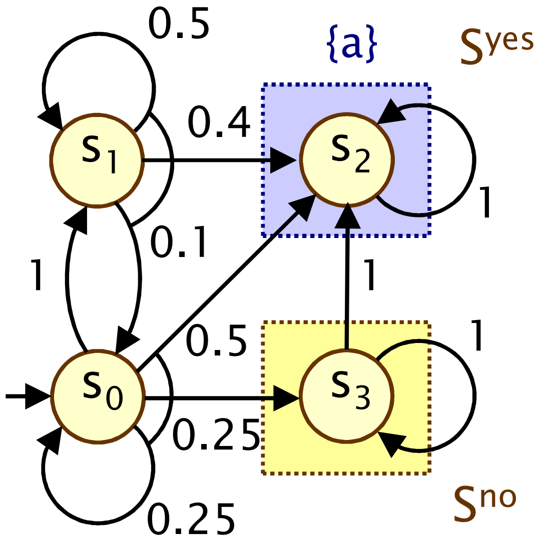
n=2: $[\ \min(0.4, 0.25 \cdot 0 + 0.5),$

$0.1 \cdot 0 + 0.5 \cdot 0.4 + 0.4,\ 1,\ 0\ ]$

$= [\ 0.4, 0.6, 1, 0\ ]$

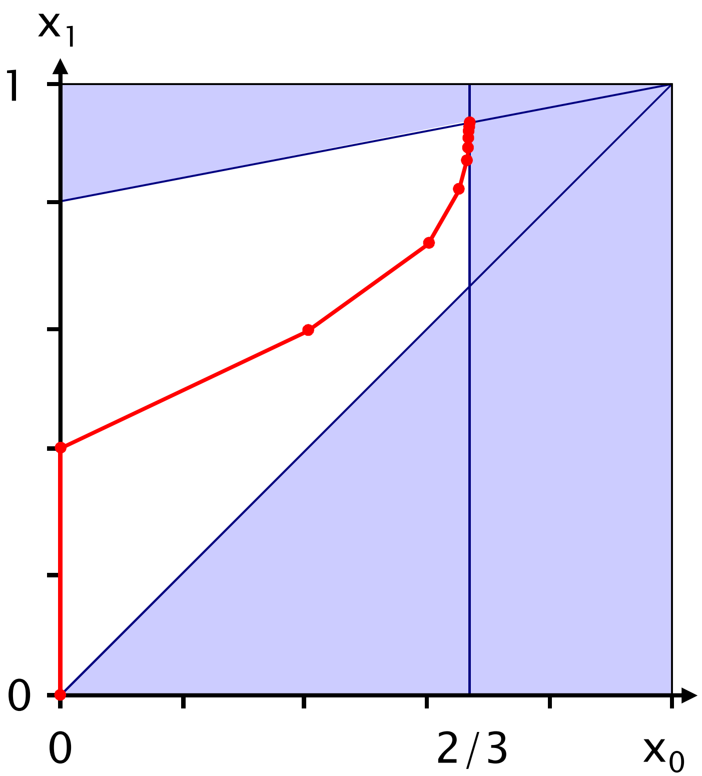
n=3: …

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]

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

Method 3 – Policy iteration

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

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

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

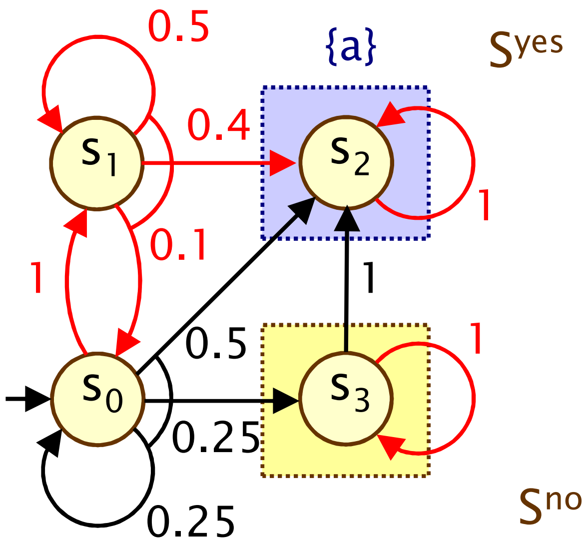
Method 3 – Policy iteration

- 1. Start with an arbitrary (memoryless) strategy 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 strategy in each state

$$\sigma'(s) = \arg\min \left\{ \sum_{s' \in S} \mu(s') \cdot Pr^{\sigma}_{s'}(F\ a) \mid (a, \mu) \in \delta(s) \right\}$$

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

Example - Policy iteration



Arbitrary strategy 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:

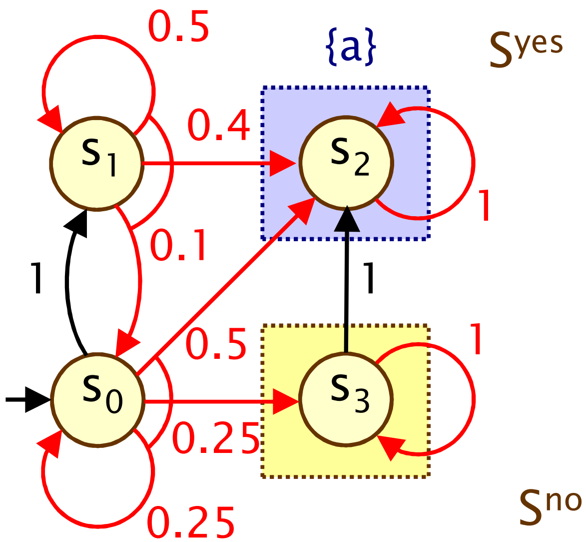
$\underline{Pr}^{\sigma}(F\ a) = [\ 1,\ 1,\ 1,\ 0\ ]$

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 strategy σ':

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

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

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

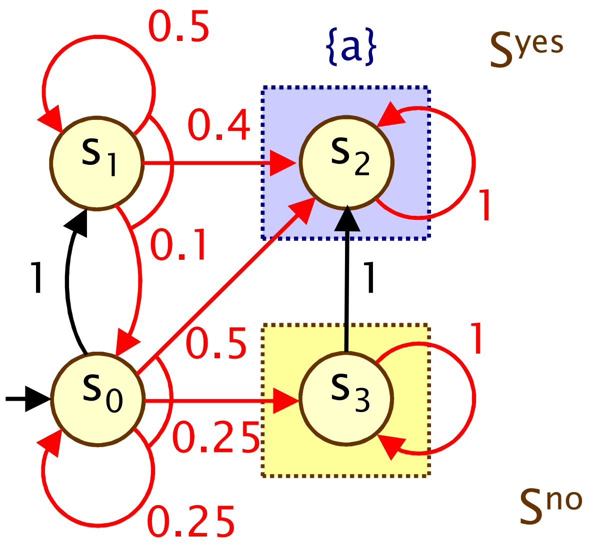
- $x_0 = 0.25 \cdot x_0 + 0.5$
- $x_1 = 0.1 \cdot x_0 + 0.5 \cdot x_1 + 0.4$

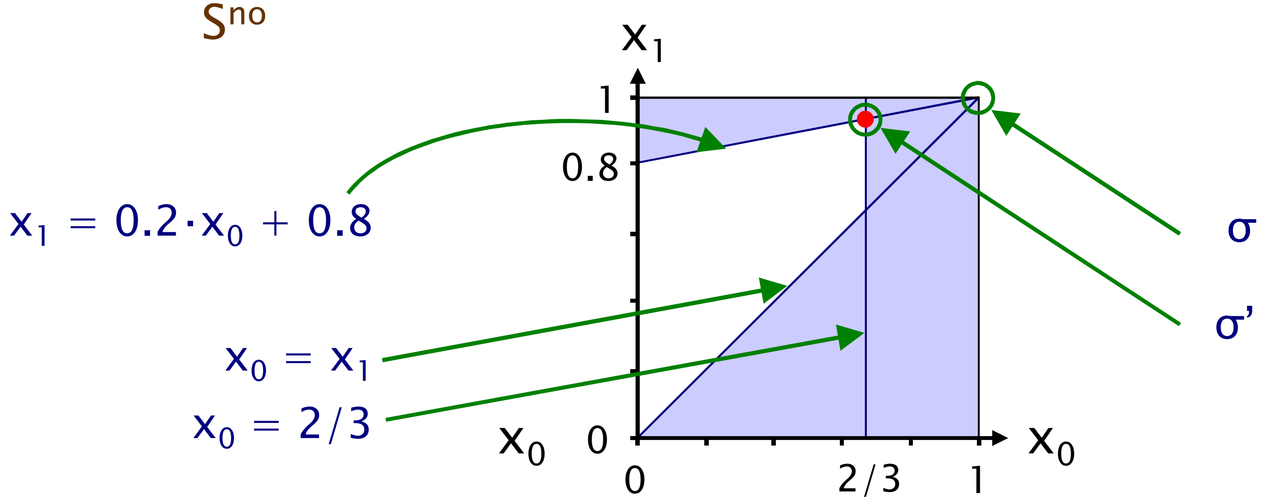
Solution:

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

This is optimal

Example - Policy iteration





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

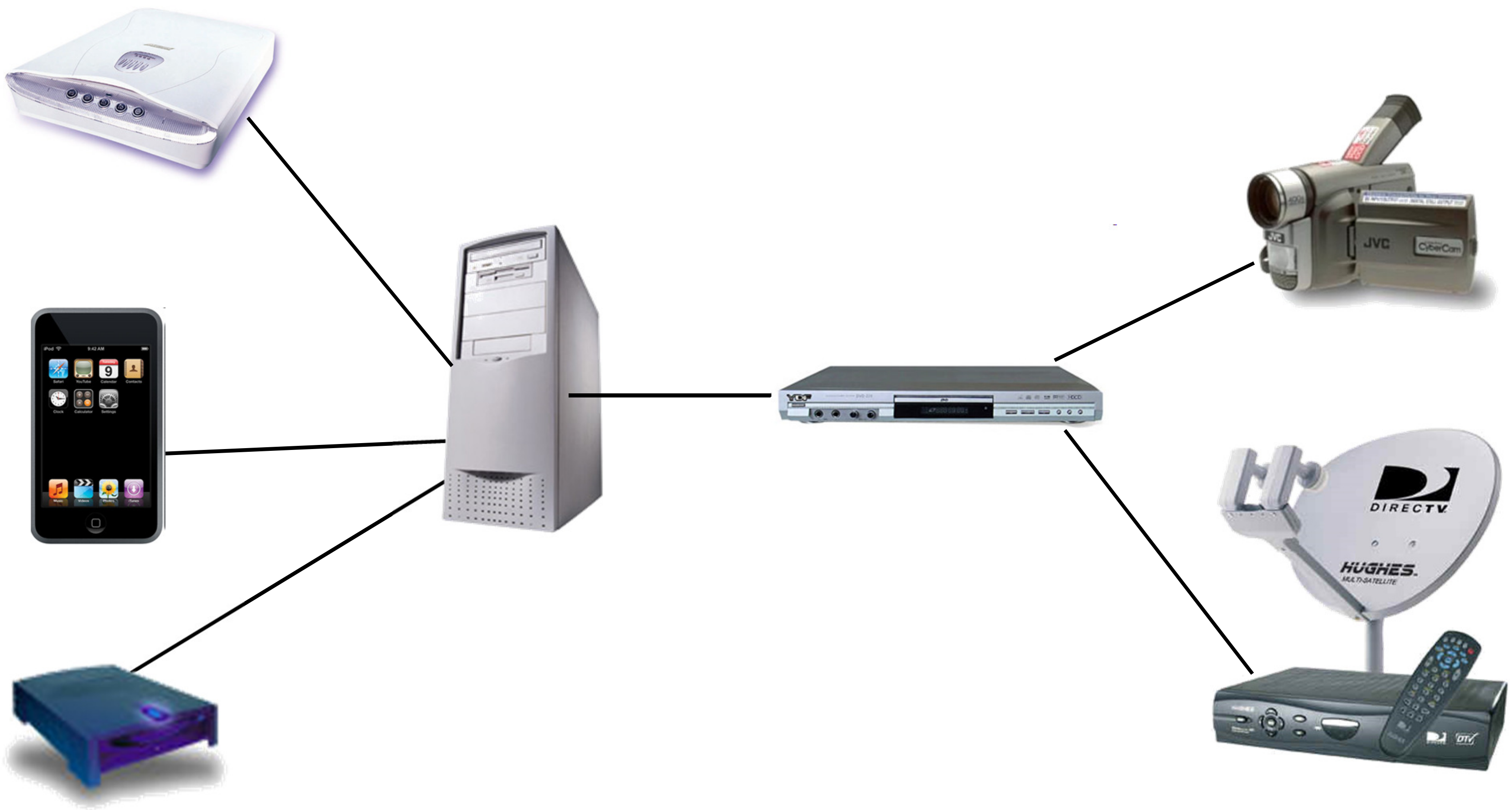
- Extend logic PCTL with R operator, for "expected reward"
  - as for PCTL, either $R_{\sim r}$ […], $R_{min=?}$ […] or $R_{max=?}$ […]
- Some examples:
  - $R_{min=?}$ [$I^{=90}$], $R_{max=?}$ [$C^{\leq 60}$], $R_{max=?}$ [F "end"]
  - "the minimum expected queue size after exactly 90 seconds"
  - "the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

Case study: FireWire root contention

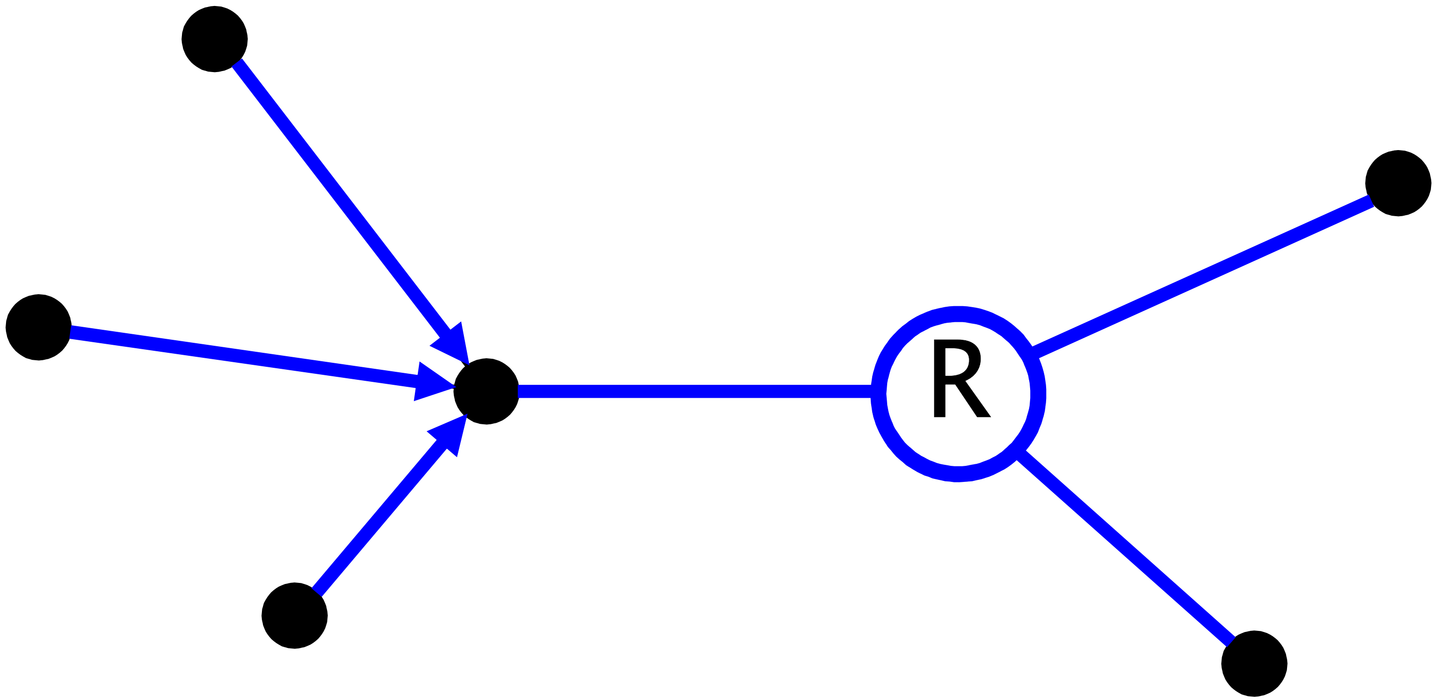
- FireWire (IEEE 1394)
  - high-performance serial bus for networking multimedia devices; originally by Apple
  - "hot-pluggable" - add/remove devices at any time
  - no requirement for a single PC (but need acyclic topology)

- Root contention protocol
  - leader election algorithm, when nodes join/leave
  - symmetric, distributed protocol
  - uses randomisation (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

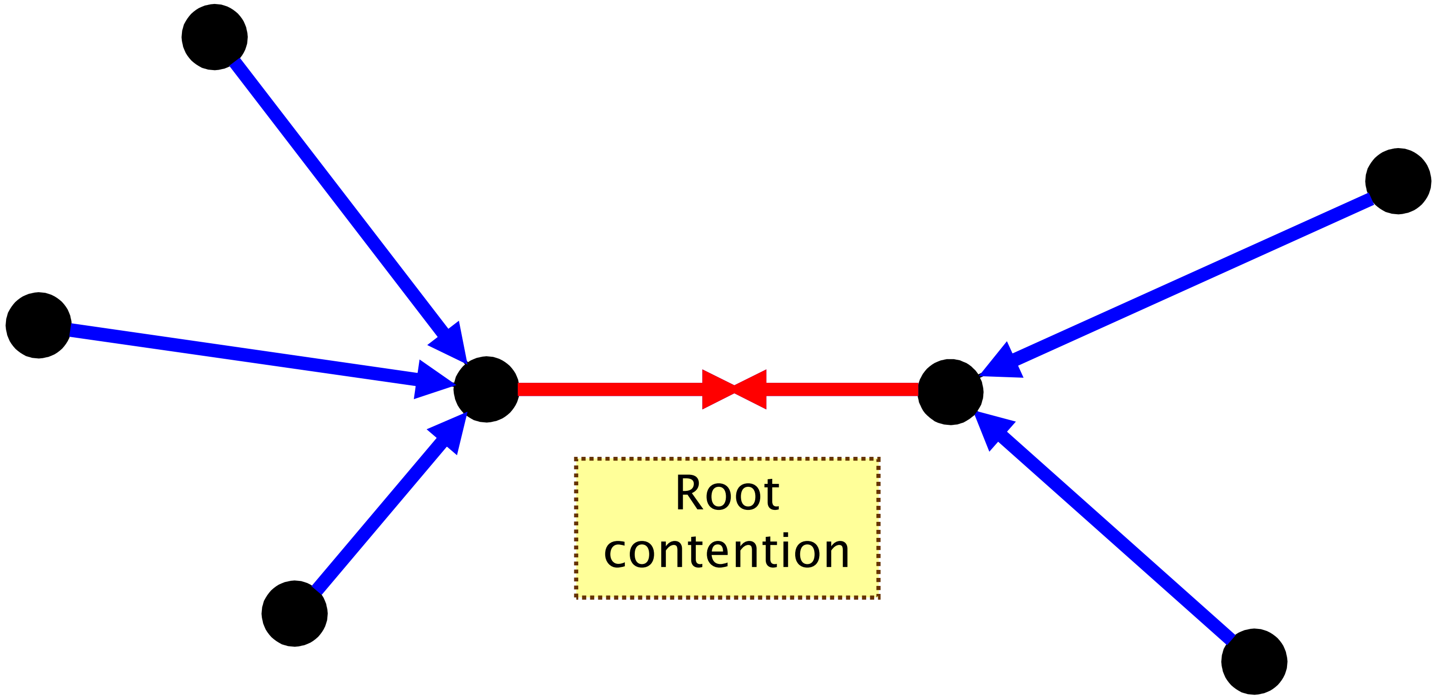
FireWire example



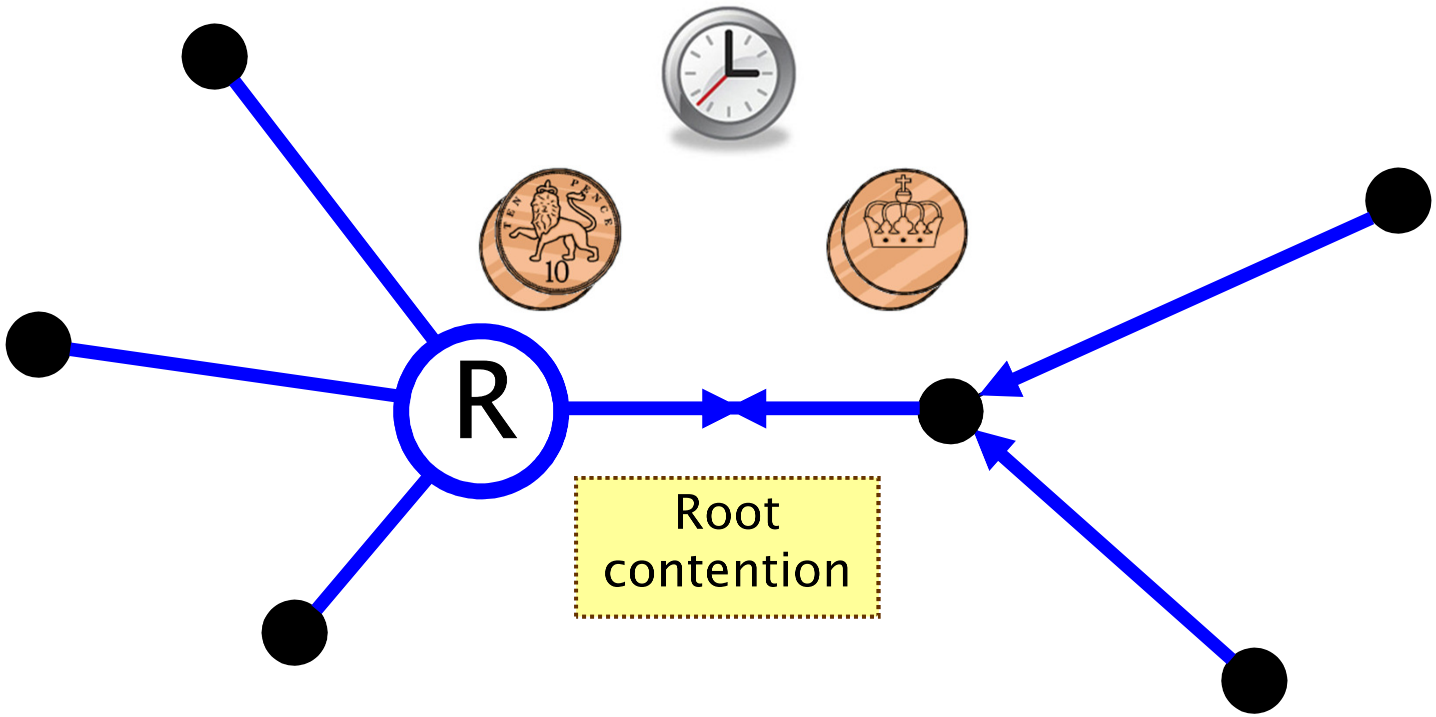
FireWire leader election



FireWire root contention

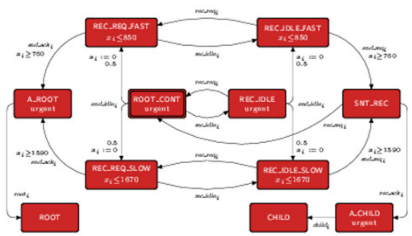


FireWire root contention

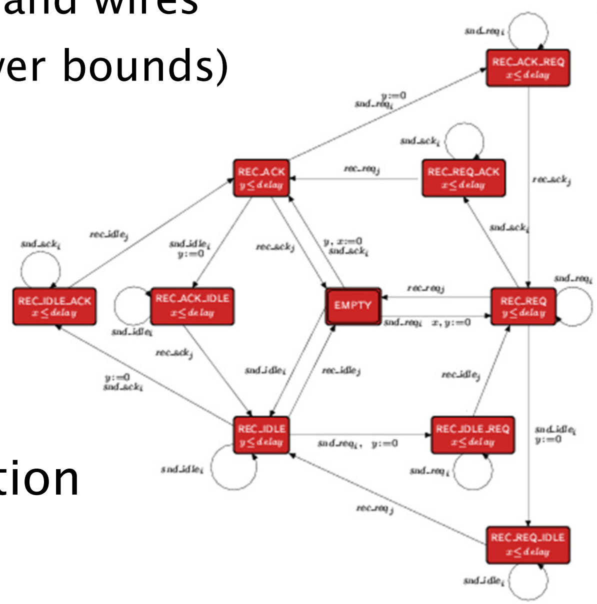


FireWire analysis

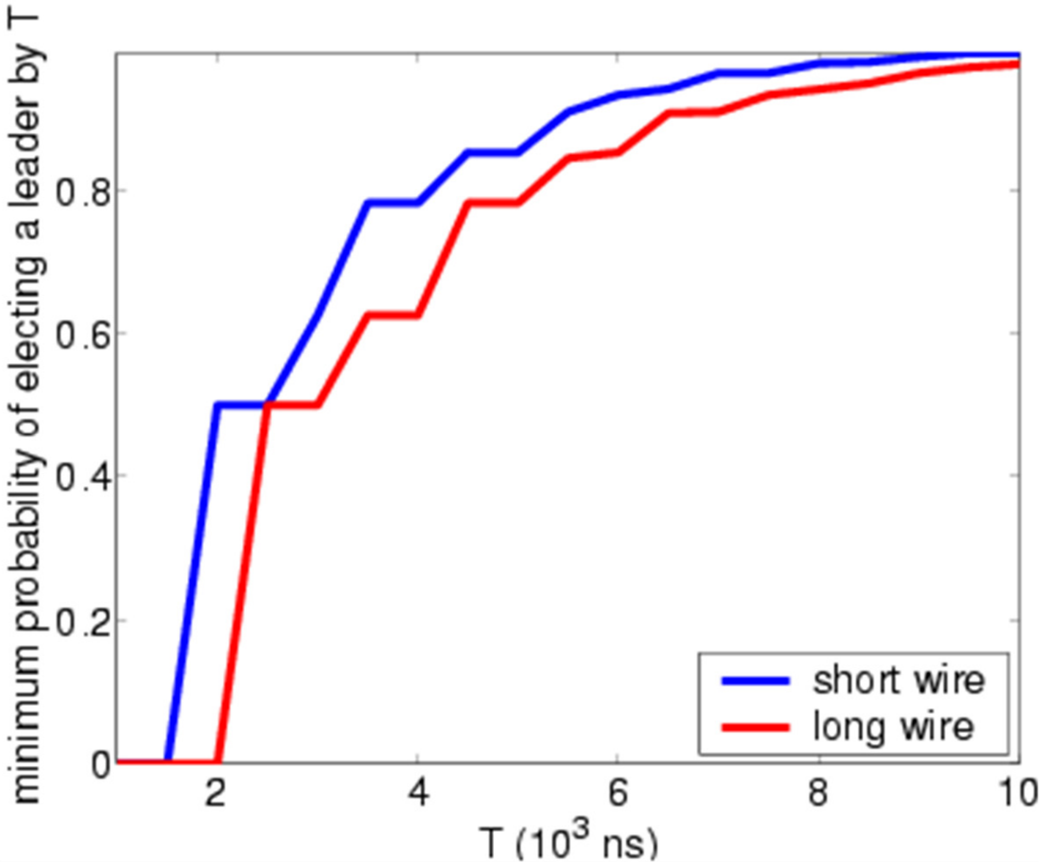
- Probabilistic model checking
  - model constructed and analysed using PRISM
  - timing delays taken from IEEE standard
  - model includes:
    - concurrency: messages between nodes and wires
    - underspecification of delays (upper/lower bounds)
  - max. model size: 170 million states



- 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

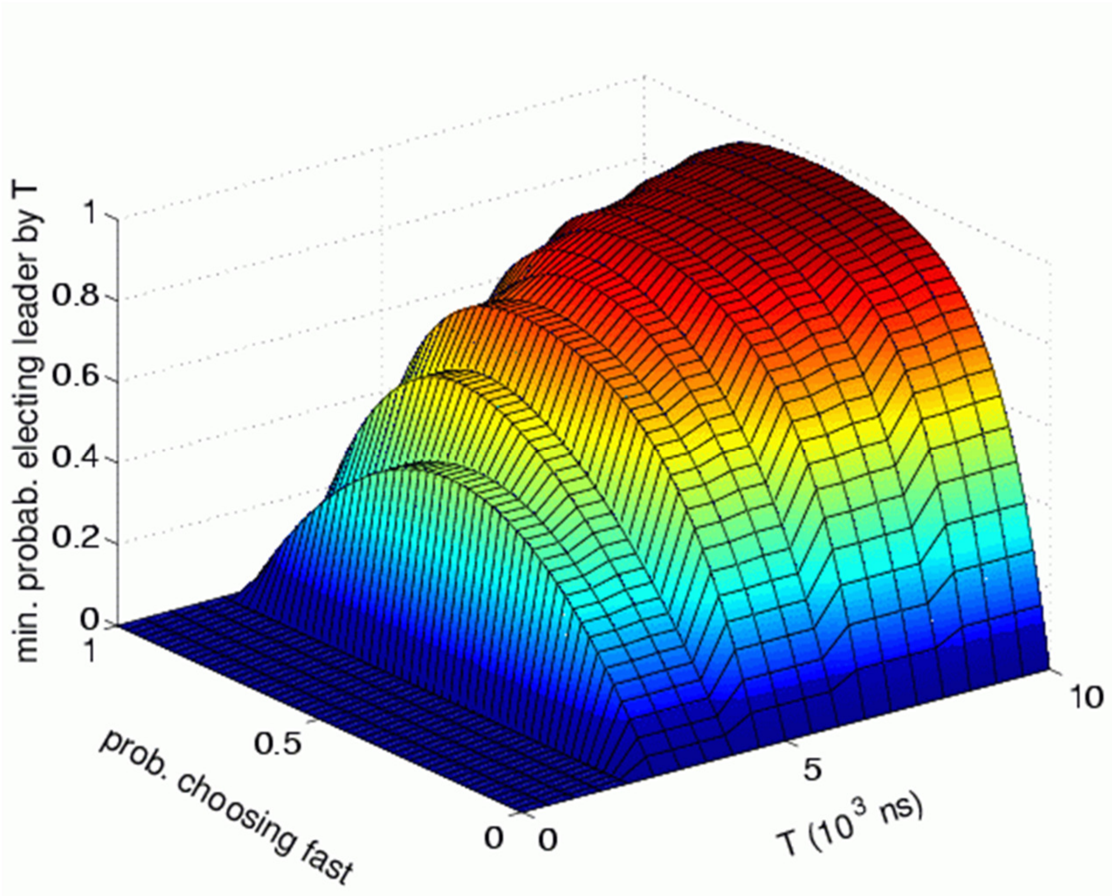


FireWire: Analysis results



"minimum probability of electing leader by time T"

FireWire: Analysis results

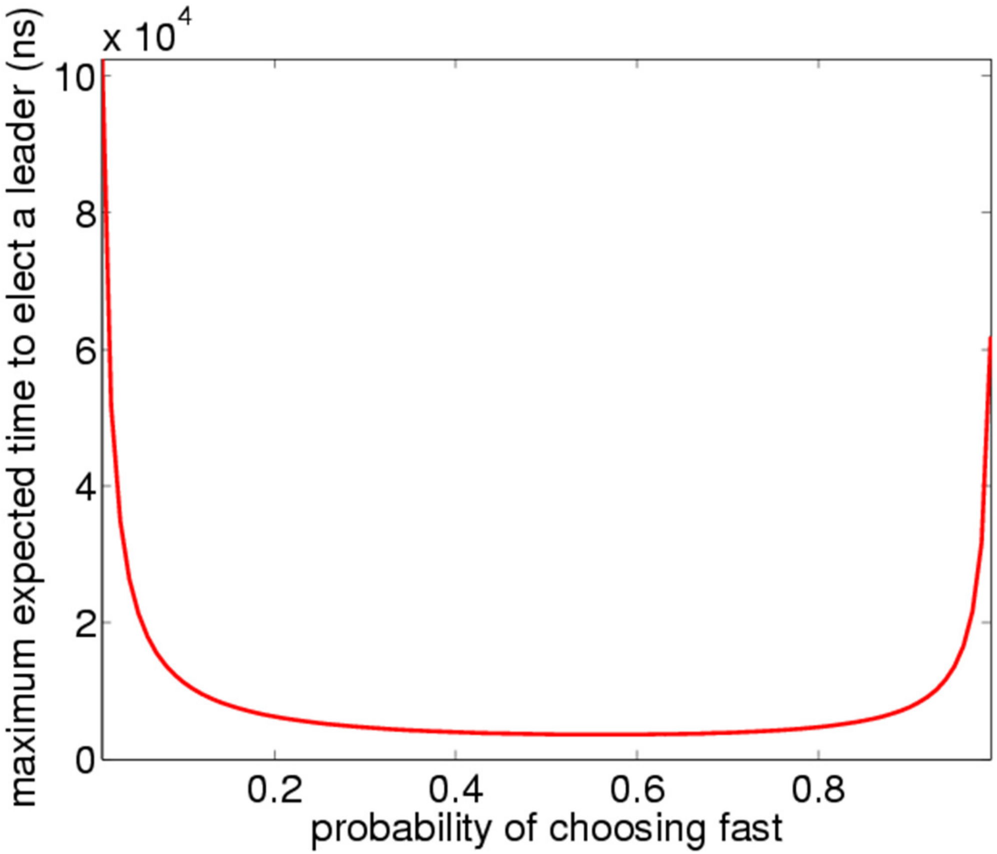


"minimum probability of electing leader by time T"

(short wire length)

Using a biased coin

FireWire: Analysis results

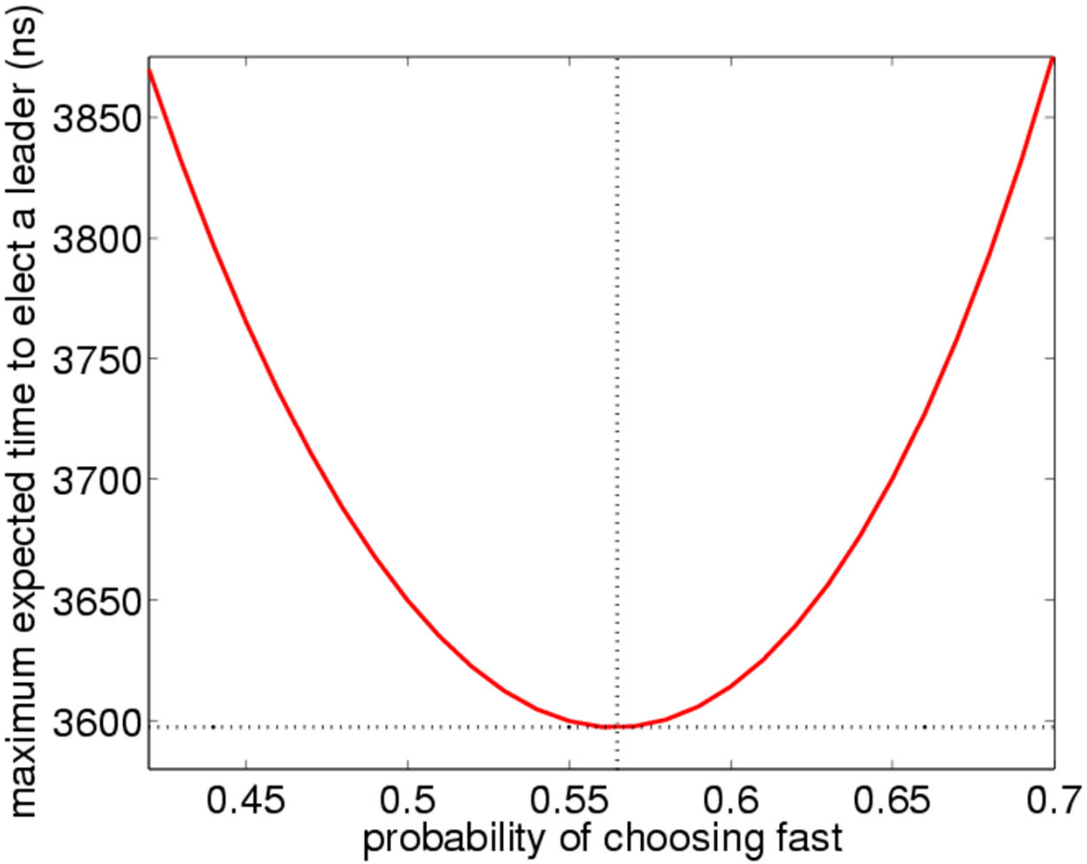


“maximum expected time to elect a leader”

(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!

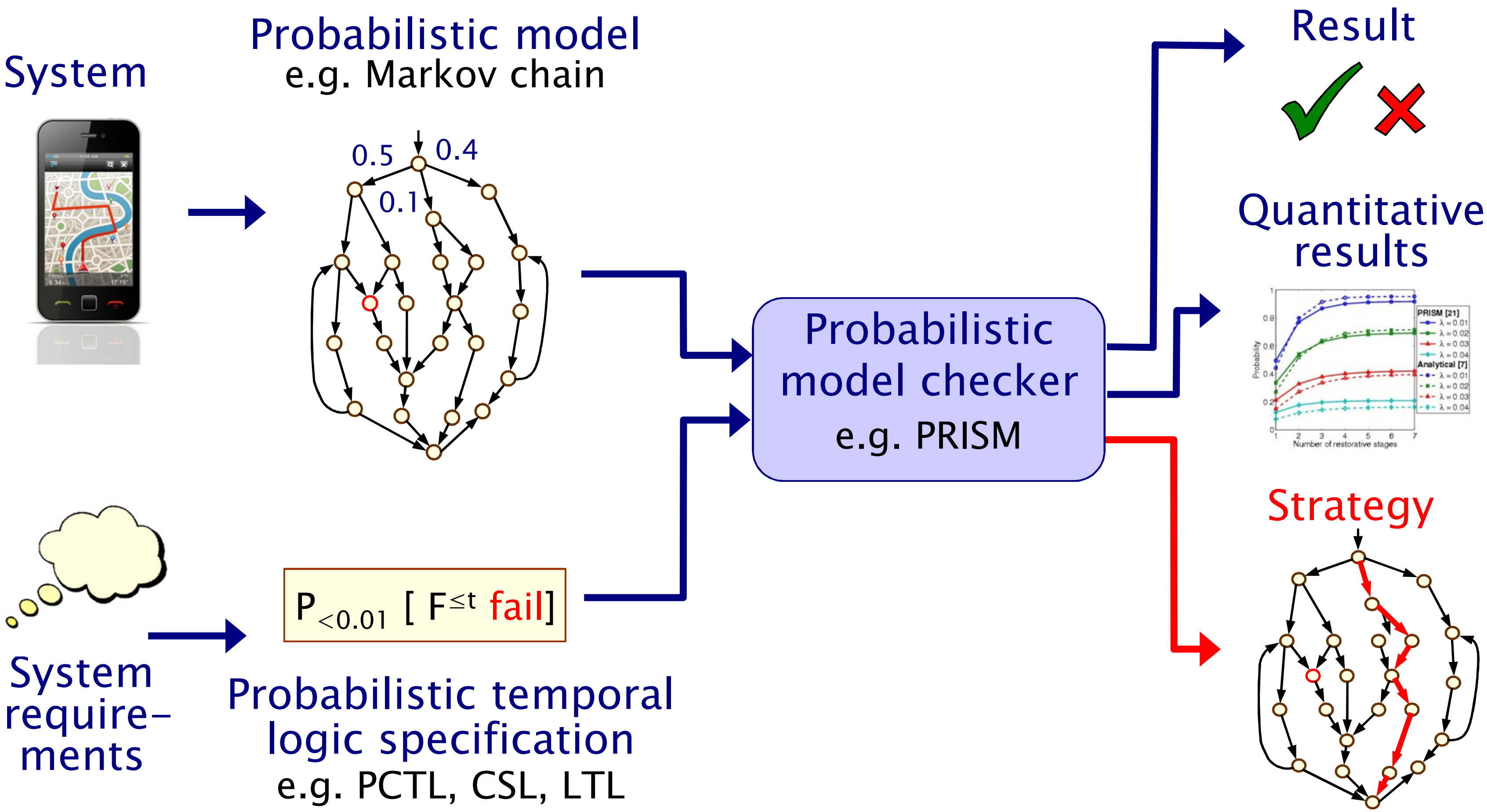
Overview (Part 2)

- Markov decision processes (MDPs)
  - MDPs: definition
  - Paths, strategies & probability spaces
- PCTL model checking
- Costs and rewards
- Case study: Firewire root contention
- Strategy synthesis for MDPs
  - Properties and objectives
  - Verification vs synthesis
- Case study: Dynamic power management
- Summary

From verification to synthesis

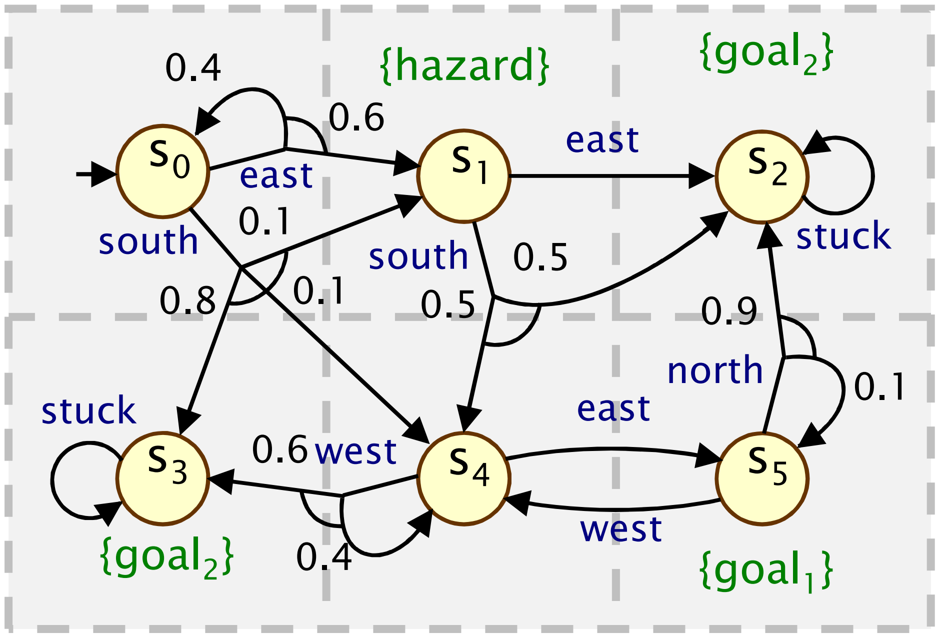
- Shift towards quantitative model synthesis from specification
  - begin with simpler problems: strategy synthesis, template-based synthesis, etc
  - advantage: correct-by-construction
- Here consider the problem of strategy (controller) synthesis
  - i.e. “can we construct a strategy to guarantee that a given quantitative property is satisfied?”
  - instead of “does the model satisfy a given quantitative property?”
  - also parameter synthesis: “find optimal value for parameter to satisfy quantitative objective”
- Many application domains
  - robotics (controller synthesis from LTL/PCTL)
  - dynamic power management (optimal policy synthesis)

Quantitative verification & synthesis



Running example

- Example MDP
  - robot moving through terrain divided into 3 x 2 grid



States:

s_0, s_1, s_2, s_3, s_4, s_5

Actions:

north, east, south, west, stuck

Labels (atomic propositions):

hazard, goal$_1$, goal$_2$

Properties and objectives

- The syntax:

ψ is true with probability ~p

expected reward is ~r

  - $\phi ::= P_{\sim p} [\, \psi \,] \;|\; R_{\sim r} [\, \rho \,]$
  - $\psi ::= \text{true} \mid a \mid \psi \wedge \psi \mid \neg \psi \mid X\, \psi \mid \psi\, U^{\leq k}\, \psi \mid \psi\, U\, \psi$

“next” “bounded until” “until”

  - $\rho ::= F\, b \mid C \mid C^{\leq k}$

“reachability” “cumulative”

  - where b is an atomic proposition, used to identify states of interest, $p \in [0,1]$ is a probability, $\sim \in \{<,>,\leq,\geq\}$, $k \in \mathbb{N}$, and $r \in \mathbb{R}_{\geq 0}$
  - $F\, b \equiv \text{true}\ U\ b$
- We refer to ϕ as property, ψ and ρ as objectives
  - (branching time more challenging for synthesis)

Properties and objectives

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

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

- Some examples:
  - $P_{\geq 0.4}$ [F "goal"] "probability of reaching goal is at least 0.4"
  - $R_{<5}$ [$C^{\leq 60}$] "expected power consumption over one hour is below 5"
  - $R_{\leq 10}$ [F "end"] "expected time to termination is at most 10"

Verification and strategy synthesis

- The verification problem is:
  - Given an MDP M and a property ϕ, does M satisfy ϕ under any possible strategy σ?

- The synthesis problem is dual:
  - Given an MDP M and a property ϕ, find, if it exists, a strategy σ such that M satisfies ϕ under σ

- Verification and strategy synthesis is achieved using <u>the same techniques</u>, namely computing optimal values for probability objectives:
  - $Pr_s^{min}(\psi) = \inf_\sigma Pr_s^\sigma(\psi)$
  - $Pr_s^{max}(\psi) = \sup_\sigma Pr_s^\sigma(\psi)$
  - and similarly for expectations

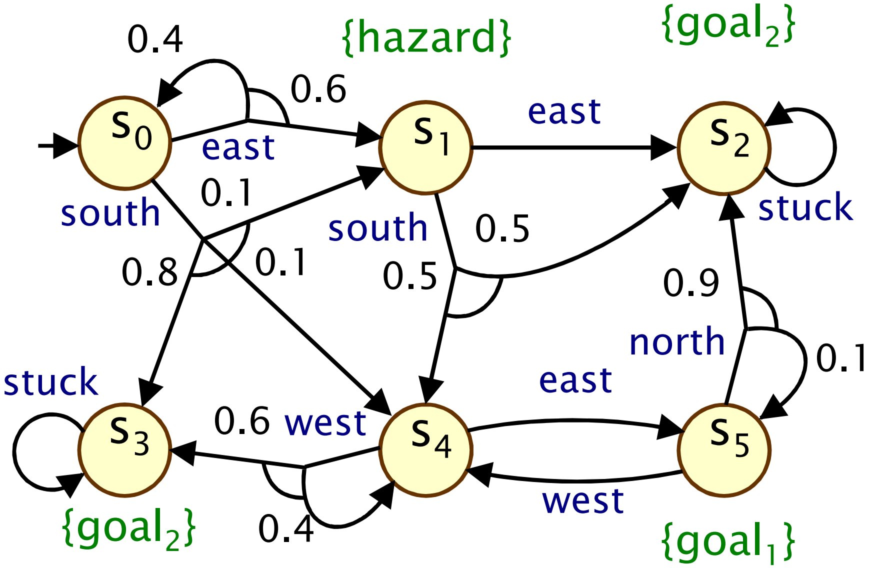
Computing reachability for MDPs

- Computation of probabilities $Pr_s^{max}(F\ b)$ for all $s \in S$

- Step 1: pre-compute all states where probability is 1 or 0
  - graph-based algorithms, yielding sets S^{yes}, S^{no}
- Step 2: compute probabilities for remaining states ($S^?$)
  - (i) solve linear programming problem
  - (i) approximate with value iteration
  - (iii) solve with policy (strategy) iteration

- 1. Precomputation:
  - algorithm Prob1E computes S^{yes}
    - there exists a strategy for which the probability of "F b" is 1
  - algorithm Prob0A computes S^{no}
    - for all strategies, the probability of satisfying "F b" is 0

Example - Reachability



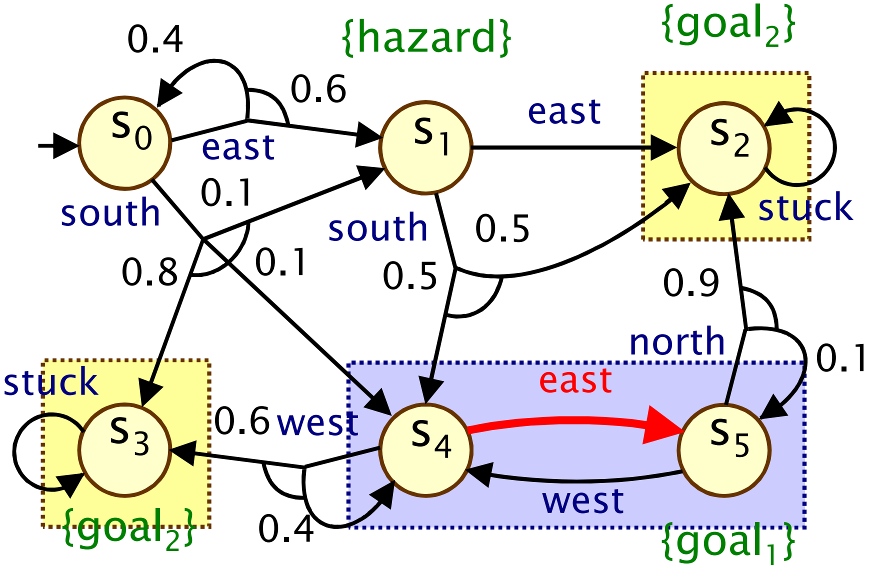
Example:

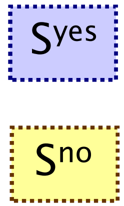
$P_{\geq 0.4}$ [F goal$_1$]

So compute:

Pr_s^{max}(F goal$_1$)

Example – Precomputation





Example:

$P_{\geq 0.4}$ [F goal$_1$]

So compute:

Pr_s^{max}(F goal$_1$)

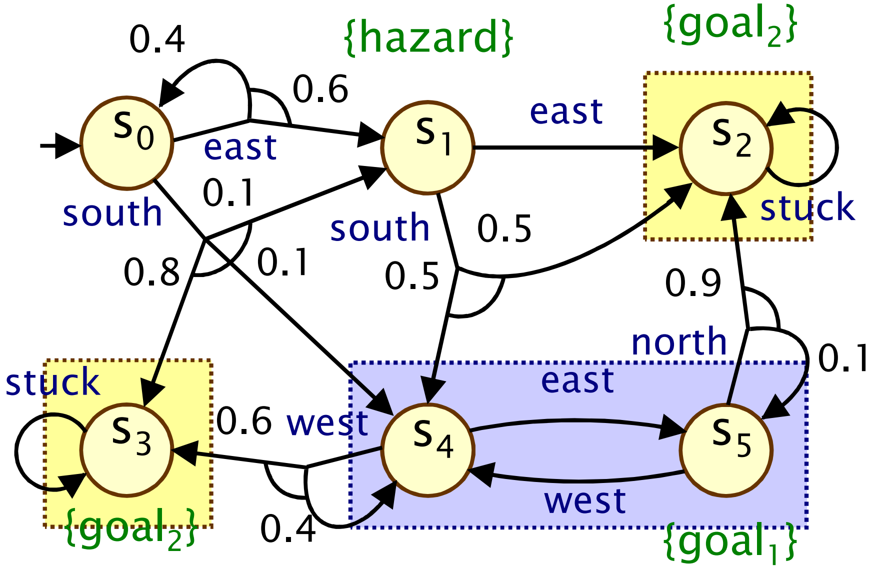
Reachability for MDPs

- 2. Numerical computation
  - compute probabilities $Pr_s^{max}(F\ b)$
  - for remaining states in $S^? = S \setminus (S^{yes} \cup S^{no})$
  - obtained as the unique solution of the linear programming (LP) problem:

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

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

Example – Reachability (LP)



Example:

$P_{\geq 0.4}$ [F $goal_1$]

So compute:

$Pr_s^{max}(F\ goal_1)$

Let $x_i = Pr_{s_i}^{max}(F\ goal_1)$

S^{yes}: $x_4 = x_5 = 1$

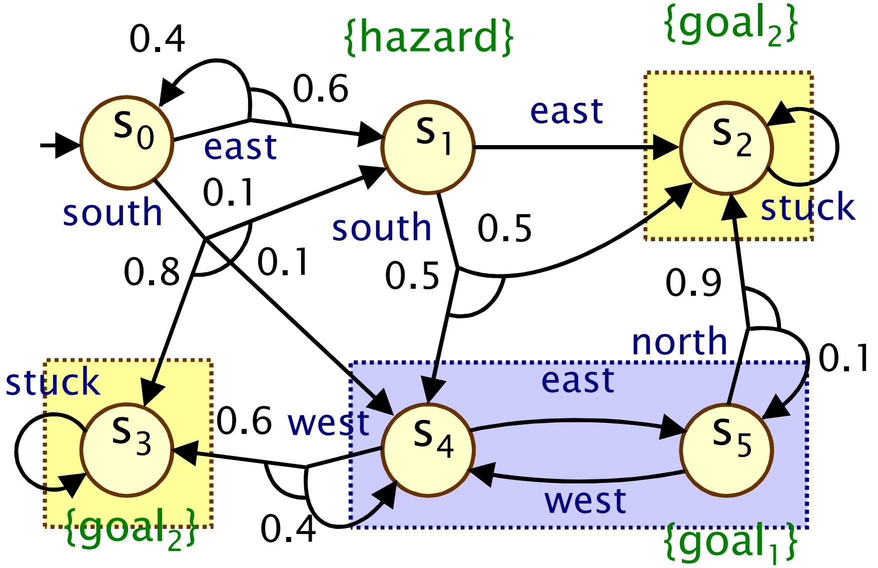
S^{no}: $x_2 = x_3 = 0$

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

Minimise $x_0 + x_1$ subject to:

- $x_0 \geq 0.4 \cdot x_0 + 0.6 \cdot x_1$ (east)
- $x_0 \geq 0.1 \cdot x_1 + 0.1$ (south)
- $x_1 \geq 0.5$ (south)
- $x_1 \geq 0$ (east)

Example – Reachability (LP)



Let $x_i = Pr_{s_i}^{max}(F\ goal_1)$

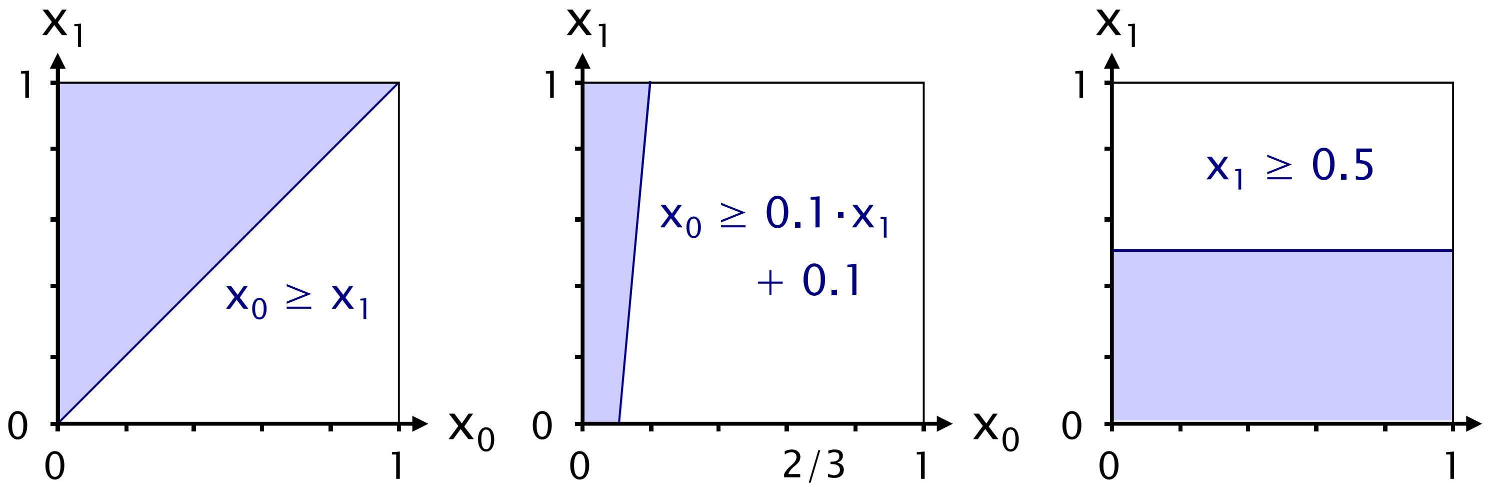
S^{yes}: $x_4 = x_5 = 1$

S^{no}: $x_2 = x_3 = 0$

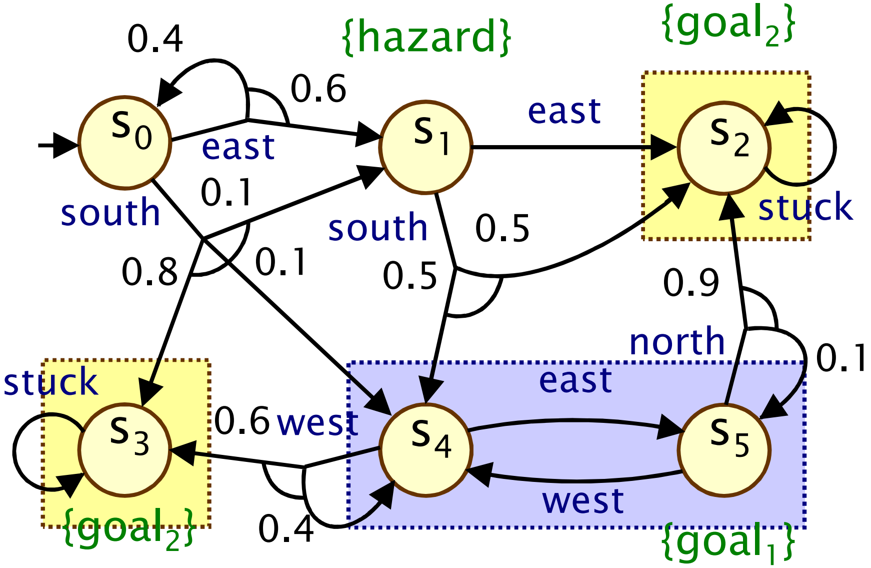
For $S^? = \{x_0, x_1\}$:

Minimise $x_0 + x_1$ subject to:

- $x_0 \geq x_1$ (east)
- $x_0 \geq 0.1 \cdot x_1 + 0.1$ (south)
- $x_1 \geq 0.5$ (south)



Example – Reachability (LP)



Let $x_i = Pr_{s_i}^{max}(F\ goal_1)$

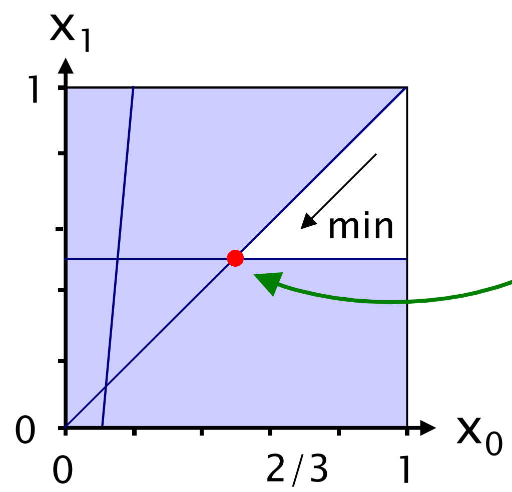
S^{yes}: $x_4 = x_5 = 1$

S^{no}: $x_2 = x_3 = 0$

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

Minimise $x_0 + x_1$ subject to:

- $x_0 \geq x_1$
- $x_0 \geq 0.1 \cdot x_1 + 0.1$
- $x_1 \geq 0.5$



Solution:

$(x_0, x_1) = (0.5, 0.5)$

i.e.

$Pr_{s_0}^{max}(F\ goal_1) = 0.5$

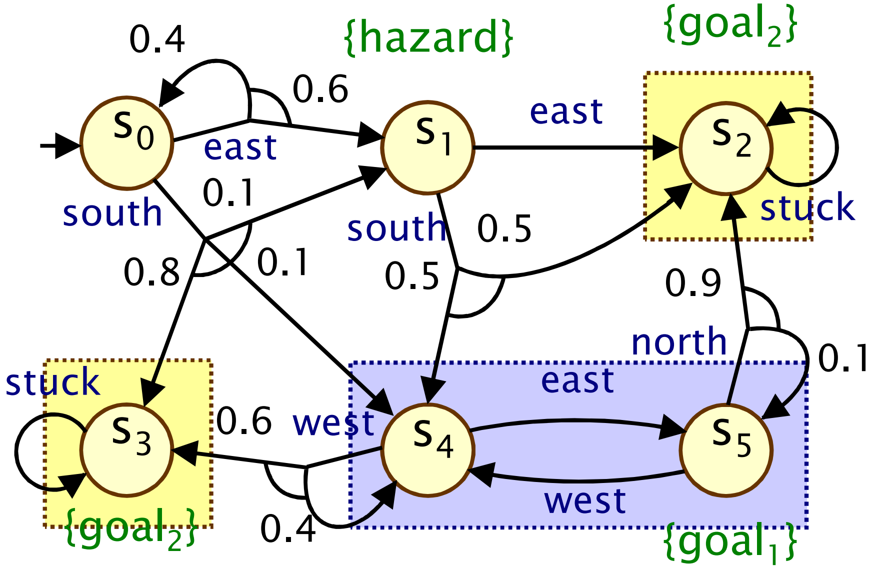
Reachability for MDPs

- 2. Numerical computation (alternative method)
  - value iteration
  - it can be shown that: $Pr_s^{max}(F\ b) = \lim_{n\to\infty} x_s^{(n)}$ where:

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

- Approximate iterative solution technique
  - iterations terminated when solution converges sufficiently

Example - Reachability (val. iter.)



Compute: $Pr_s^{max}(F\ goal_1)$

S^{yes}: $x_4=x_5=1$

S^{no}: $x_2=x_3=0$

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

$$[\ x_0^{(n)}, x_1^{(n)}, x_2^{(n)}, x_3^{(n)}, x_4^{(n)}, x_5^{(n)}\ ]$$

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

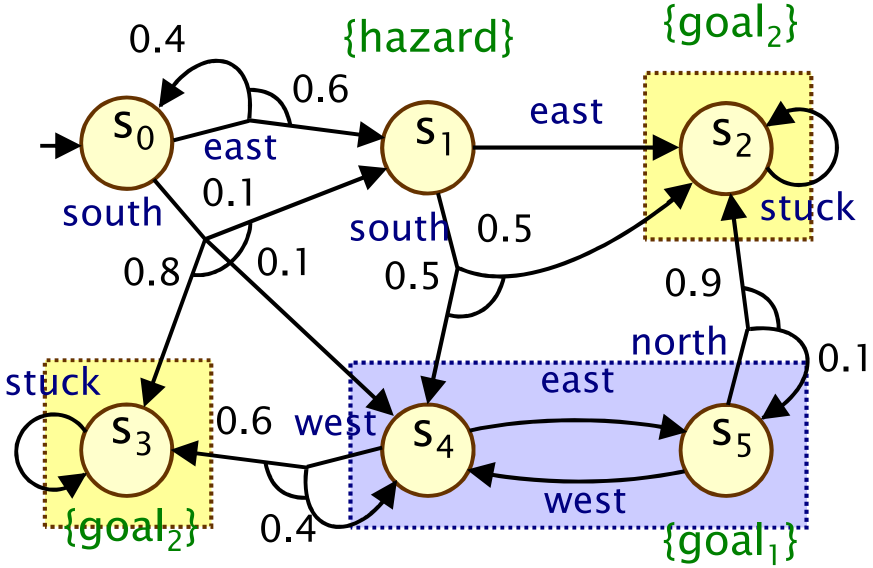
n=1: $[\ \max(0.6\cdot0+0.4\cdot0,\ 0.1\cdot0+0.1\cdot1+0.8\cdot0),\ \max(0, 0.5),\ 0, 0, 1, 1\ ]$

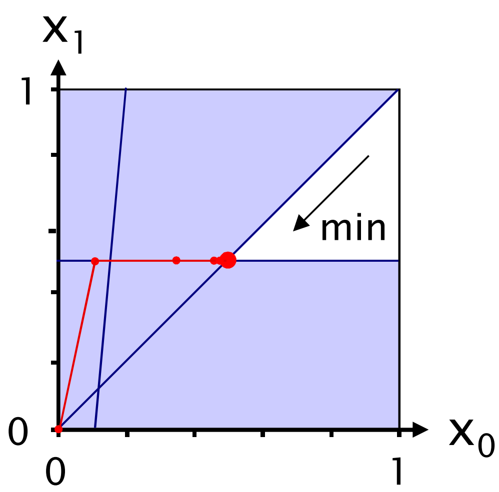
$= [\ 0.1, 0.5, 0, 0, 1, 1\ ]$

n=2: $[\ \max(0.6\cdot0.5+0.4\cdot0.1,\ 0.1\cdot0.5+0.1\cdot1+0.8\cdot0),\ \max(0, 0.5),\ 0, 0, 1, 1\ ]$

$= [\ 0.34, 0.5, 0, 0, 1, 1\ ]$

Example – Reachability (val. iter.)





$[x_0^{(n)}, x_1^{(n)}, x_2^{(n)}, x_3^{(n)}, x_4^{(n)}, x_5^{(n)}]$

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

n=1: [0.1, 0.5, 0, 0, 1, 1]

n=2: [0.34, 0.5, 0, 0, 1, 1]

n=3: [0.436, 0.5, 0, 0, 1, 1]

n=4: [0.4744, 0.5, 0, 0, 1, 1]

n=5: [0.48976, 0.5, 0, 0, 1, 1]

n=6: [0.495904, 0.5, 0, 0, 1, 1]

n=7: [0.4983616, 0.5, 0, 0, 1, 1]

n=8: [0.49934464, 0.5, 0, 0, 1, 1]

…

n=16: [0.49999957, 0.5, 0, 0, 1, 1]

n=17: [0.49999982, 0.5, 0, 0, 1, 1]

… ≈ [0.5 0.5, 0, 0, 1, 1]

Memoryless strategies

- Memoryless strategies suffice for probabilistic reachability
  - i.e. there exist memoryless strategies σ_{min} & σ_{max} such that:
  - $\text{Prob}^{\sigma_{min}}(s, F\ a) = p_{min}(s, F\ a)$ for all states $s \in S$
  - $\text{Prob}^{\sigma_{max}}(s, F\ a) = p_{max}(s, F\ a)$ for all states $s \in S$

- Construct strategies from optimal solution:

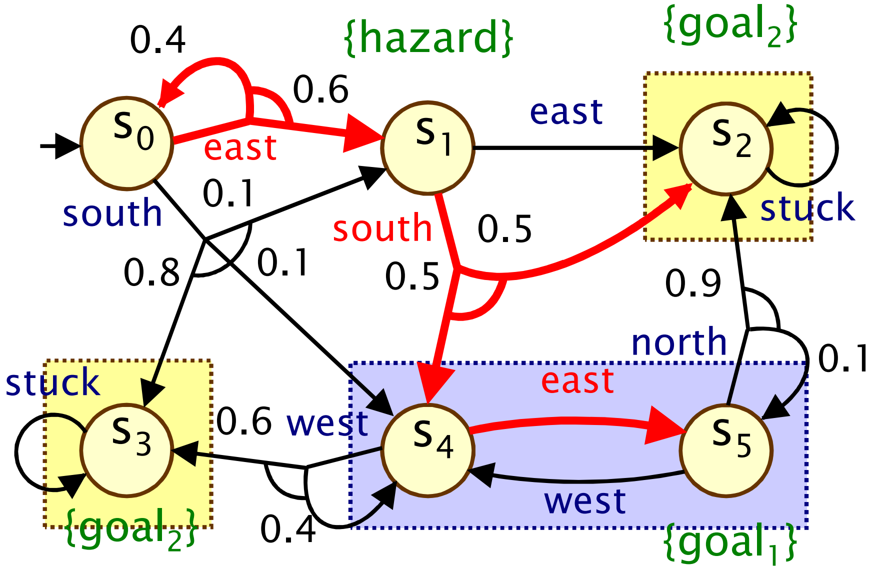
$$\sigma_{min}(s) = \text{argmin}\left\{ \sum_{s' \in S} \mu(s') \cdot p_{min}(s', Fa) \mid (a, \mu) \in \delta(s) \right\}$$

$$\sigma_{max}(s) = \text{argmax}\left\{ \sum_{s' \in S} \mu(s') \cdot p_{max}(s', Fa) \mid (a, \mu) \in \delta(s) \right\}$$

Strategy synthesis

- Compute optimal probabilities $Pr_s^{max}(F\ b)$ for all $s \in S$
- To compute the optimal strategy σ^*, choose the locally optimal action in each state
  - must guarantee progress towards target states
  - in general depends on the method used to compute the optimal probabilities
- For reachability
  - memoryless strategies suffice
- For step-bounded reachability
  - need finite-memory strategies
  - typically requires backward computation for a fixed number of steps

Example – Strategy



Optimal strategy:

s_0 : east

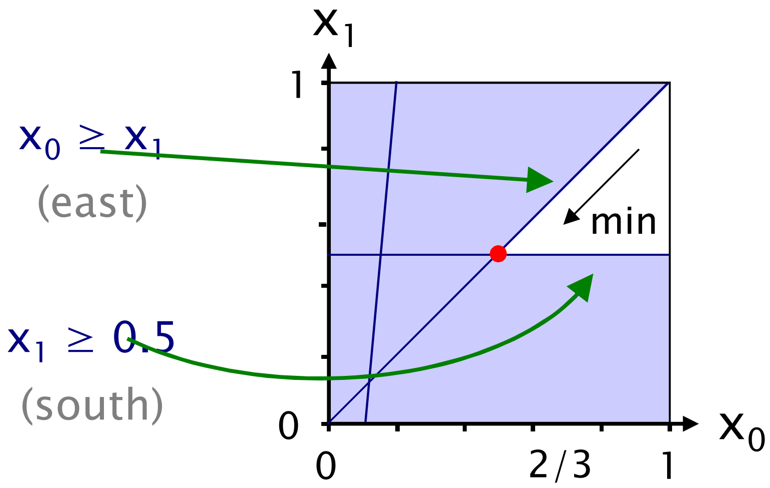
s_1 : south

s_2 : –

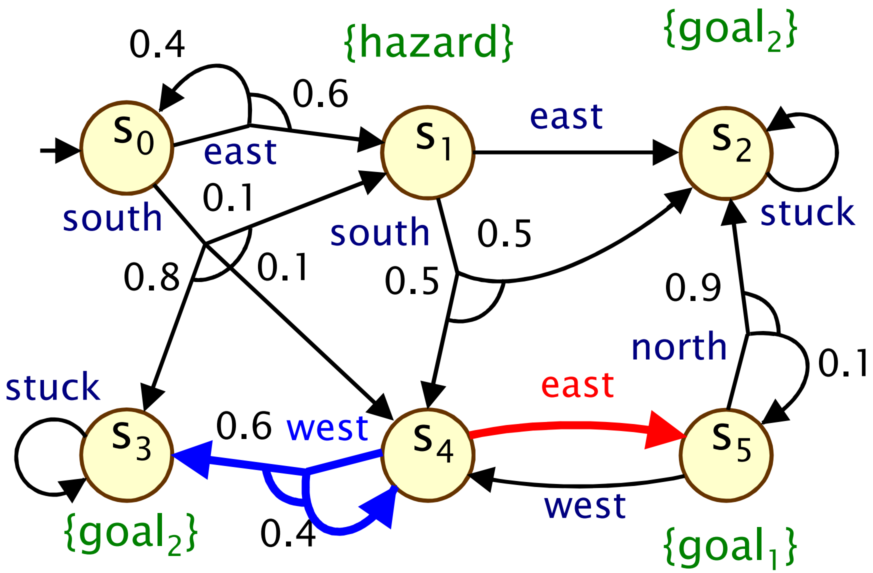
s_3 : –

s_4 : east

s_5 : –



Example – Bounded reachability



Example:

$P_{max=?}$ [$F^{\leq 3}$ goal$_2$]

So compute:

$Pr_s^{max}(F^{\leq 3}\ goal_2) = 0.99$

Optimal strategy

is finite-memory:

s_4 (after 1 step): east

s_4 (after 2 steps): west

Strategy synthesis for LTL objectives

- Reduce to the problem of reachability on the product of MDP M and an omega-automaton representing ψ
  - for example, deterministic Rabin automaton (DRA)

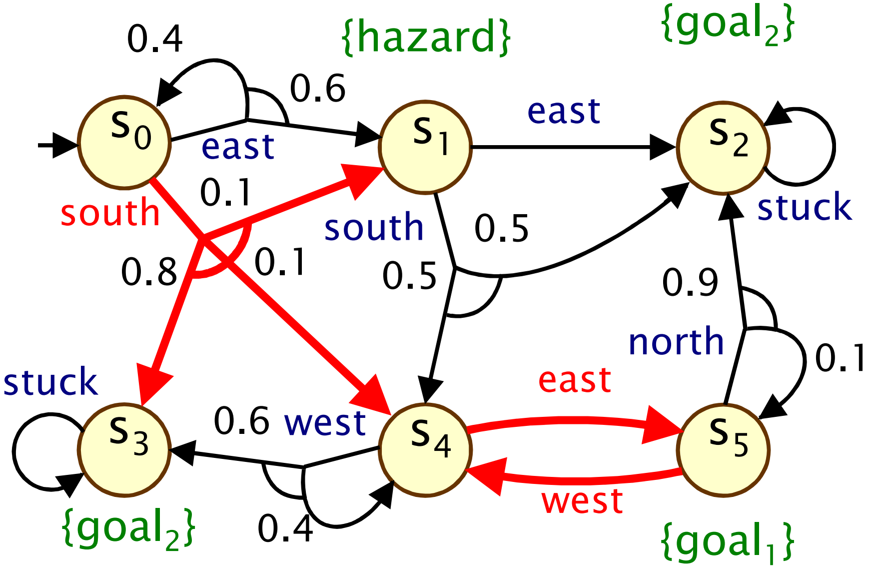
- Need only consider computation of maximum probabilities $Pr_s^{max}(\psi)$
  - since $Pr_s^{min}(\psi) = 1 - Pr_s^{max}(\neg\psi)$

- To compute the optimal strategy σ^*
  - find memoryless deterministic strategy on the product
  - convert to finite-memory strategy with one mode for each state of the DRA for ψ

Example - LTL

- $P_{\geq 0.05}$ [(G ¬hazard) ∧ (GF $goal_1$)]
  - avoid hazard and visit $goal_1$ infinitely often

- Pr_{s0}^{max}((G ¬hazard) ∧ (GF $goal_1$)) = 0.1



Optimal strategy:
(in this instance,
memoryless)

s_0 : south

s_1 : –

s_2 : –

s_3 : –

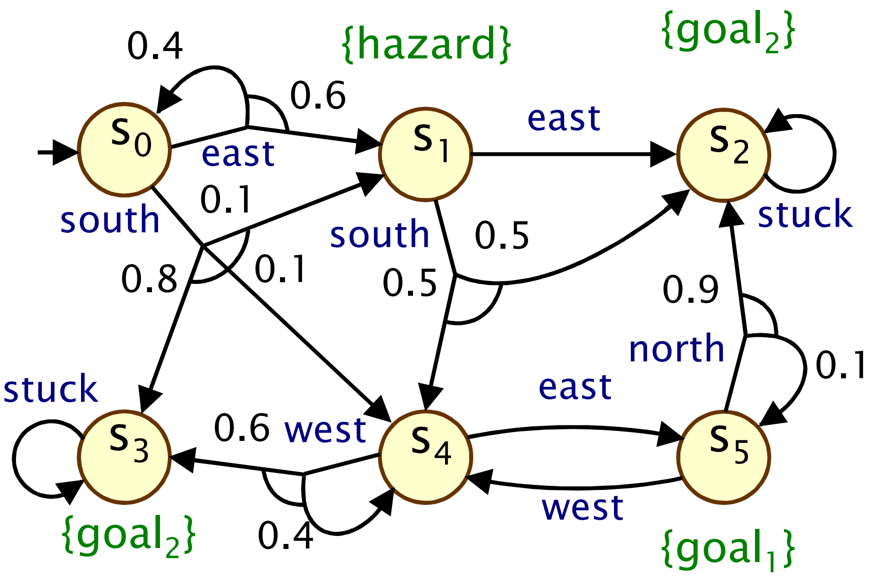
s_4 : east

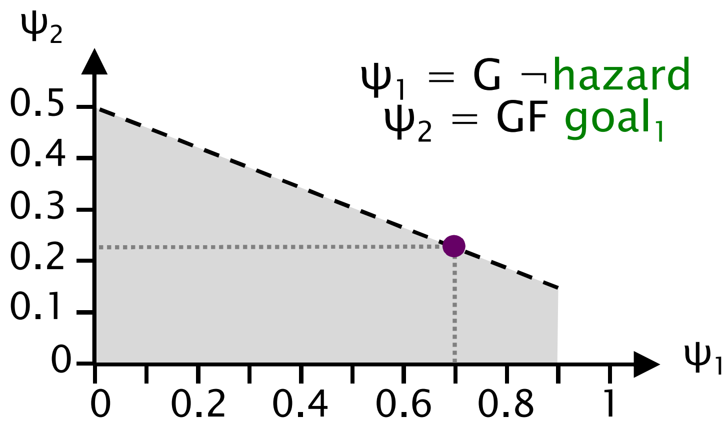
s_5 : west

Multi-objective strategy synthesis

- Consider conjunctions of probabilistic LTL formulas $P_{\sim p}$ [ψ]
  - require all conjuncts to be satisfied
- Reduce to a multi-objective reachability problem on the product of MDP M and the omega-automata representing the conjuncts
  - convert (by negation) to formulas with upper probability bounds ($\geq$, $>$), then to DRA
  - need to consider all combinations of objectives
- The problem can be solved using LP methods [TACAS07] or via approximations to Pareto curve [ATVA12]
  - strategies may be finite memory and randomised
- Continue as for single-objectives to compute the strategy σ*
  - find memoryless deterministic strategy on the product
  - convert to finite-memory strategy

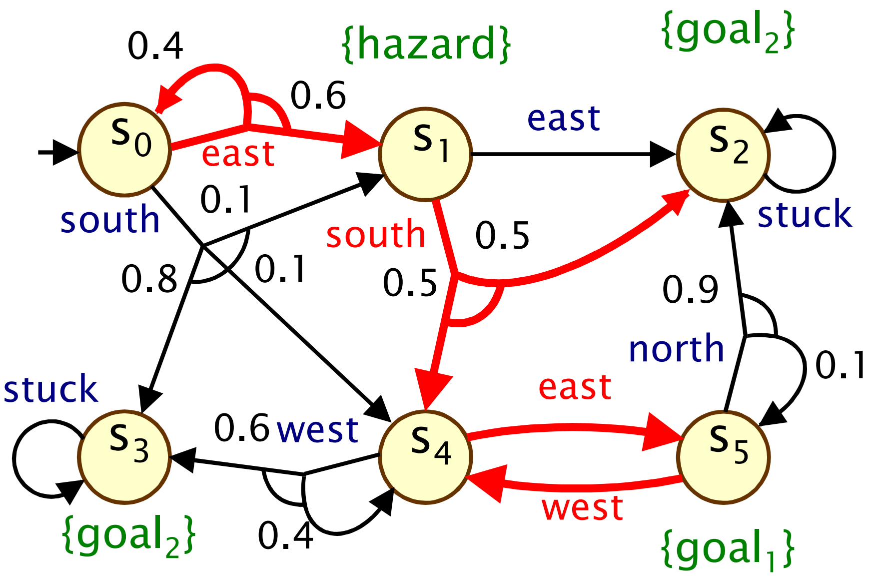
Example - Multi-objective





- Multi-objective formula
  - $P_{\geq 0.7}$ [G ¬hazard] ∧ $P_{\geq 0.2}$ [GF $goal_1$] ? True (achievable)
- Numerical query
  - $P_{max=?}$ [GF $goal_1$] such that $P_{\geq 0.7}$ [G ¬hazard] ? ~0.2278
- Pareto query
  - for $P_{max=?}$ [G ¬hazard] ∧ $P_{max=?}$ [GF $goal_1$] ?

Example - Multi-objective strategies



Strategy 1

(deterministic)

s_0 : east

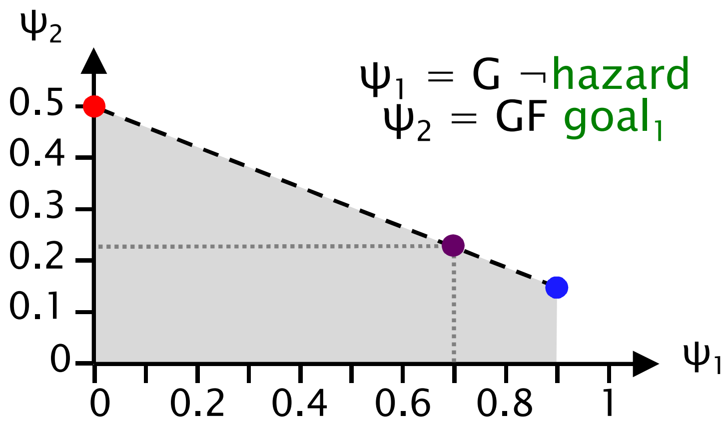
s_1 : south

s_2 : –

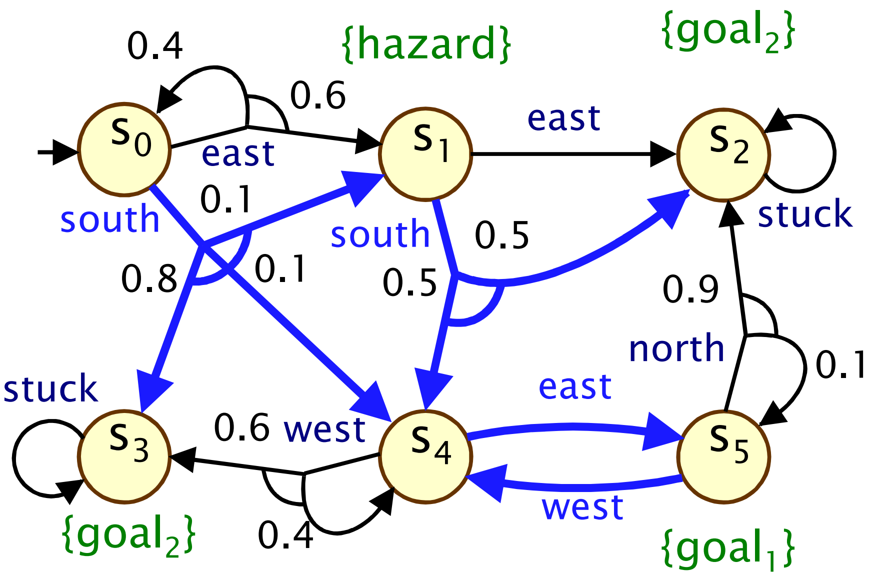
s_3 : –

s_4 : east

s_5 : west



Example – Multi-objective strategies



Strategy 2
(deterministic)

s_0 : south

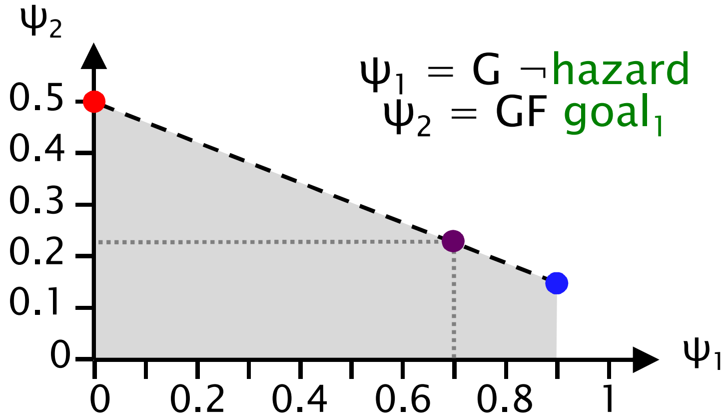
s_1 : south

s_2 : –

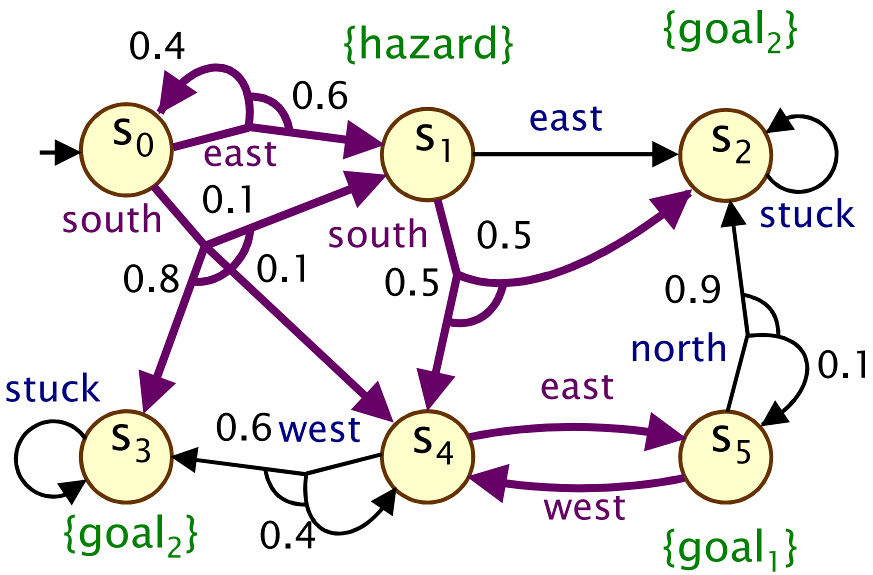
s_3 : –

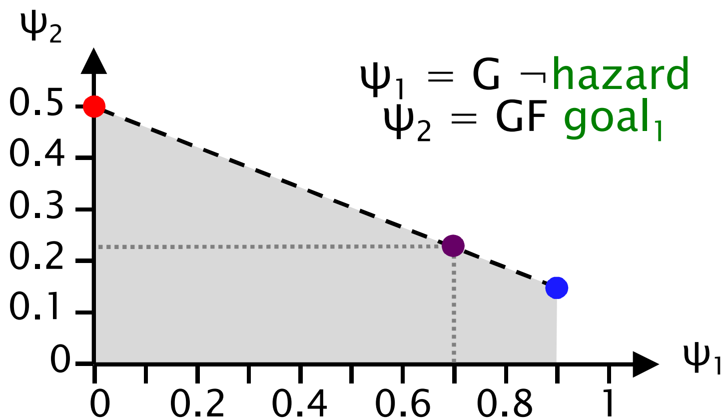
s_4 : east

s_5 : west



Example - Multi-objective strategies





Optimal strategy:
(randomised)

s_0 : 0.3226 : east
0.6774 : south

s_1 : 1.0 : south

s_2 : –

s_3 : –

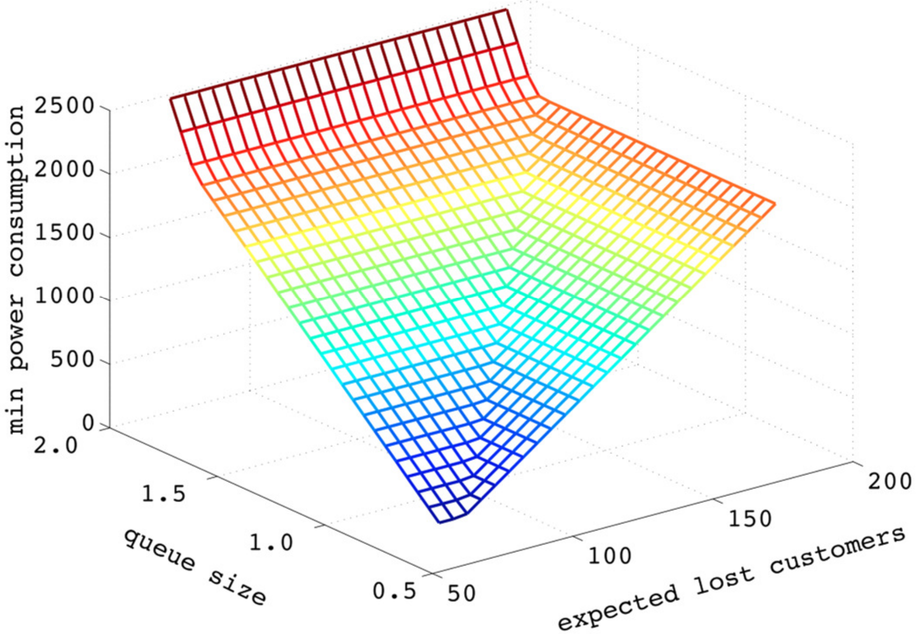
s_4 : 1.0 : east

s_5 : 1.0 : west

Case study: Dynamic power management

- Synthesis of dynamic power management schemes
  - for an IBM TravelStar VP disk drive
  - 5 different power modes: active, idle, idlelp, stby, sleep
  - power manager controller bases decisions on current power mode, disk request queue, etc.

- Build controllers that
  - minimise energy consumption, subject to constraints on e.g.
  - probability that a request waits more than K steps
  - expected number of lost disk requests



- See: lab and http://www.prismmodelchecker.org/files/tacas11/

Summary (Part 2)

- Markov decision processes (MDPs)
  - extend DTMCs with nondeterminism
  - to model concurrency, underspecification, …
- Property specifications
  - PCTL: exactly same syntax as for DTMCs
  - but quantify over all strategies
- Model checking algorithms
  - covered three basic techniques for MDPs: linear programming, value iteration, or policy iteration
- Strategy synthesis
  - can reuse model checking algorithms

PRISM: Recent & new developments

- New features:
  1. parametric model checking
  2. strategy synthesis
  3. real-time: probabilistic timed automata (PTAs)

- Further new additions:
  - enhanced statistical model checking (approximations + confidence intervals, acceptance sampling)
  - efficient CTMC model checking (fast adaptive uniformisation)
  - benchmark suite & testing functionality
  - www.prismmodelchecker.org

– Beyond PRISM…

1. Parametric model checking

- Can specify models in parametric form [TASE13]
  - parameters expressed as unevaluated constants
  - e.g. const double x;
  - transition probabilities specified as expressions over parameters, e.g. 0.5 + x
- Properties are given in PCTL, with parameter constants
  - new construct constfilter (min, x1*x2, prop)
  - filters over parameter values, rather than states
- Determine parameter valuations to guarantee satisfaction of given properties, useful for model repair
- Two methods implemented in PRISM ('explicit' engine)
  - constraints-based approach is a reimplementation of PARAM 2.0 [Hahn et al]
  - sampling-based approaches are new implementation

2. Controller (strategy) synthesis

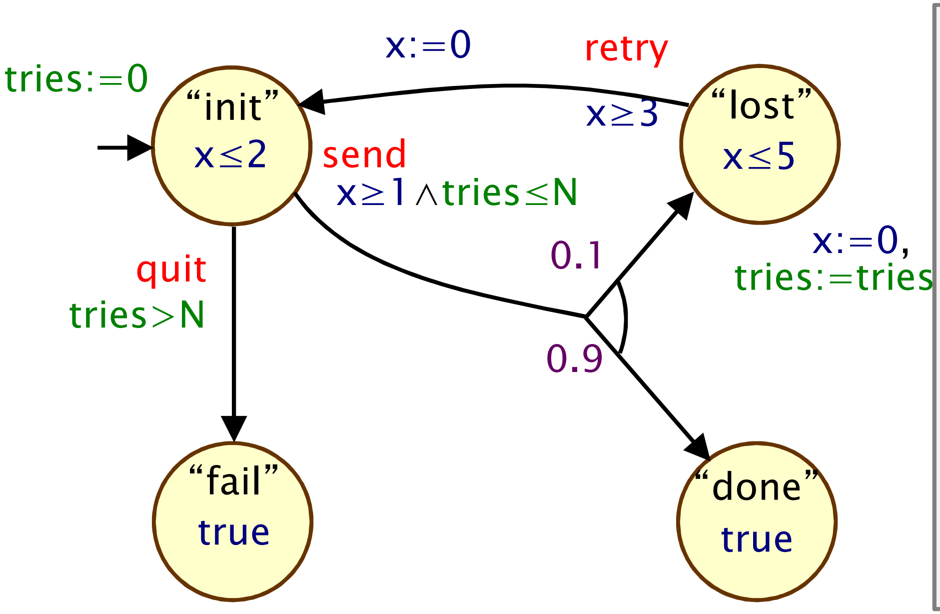
- Can synthesise controllers using machine learning [ATVA14]
  - partial exploration of the state space, with guarantees of accuracy
  - combines real-time dynamic programming (RTDP) with value iteration
  - focus on updating "most important parts" = most often visited by good strategies
  - speeds up value iteration

- Implemented in PRISM
  - for both MDPs and stochastic games
  - not yet integrated into the main release, subject of ongoing research

3. Probabilistic timed automata (PTAs)

- Probability + nondeterminism + real-time
  - timed automata + discrete probabilistic choice, or…
  - probabilistic automata + real-valued clocks

- PTA example: message transmission over faulty channel



States
- locations + data variables

Transitions
- guards and action labels

Real-valued clocks
- state invariants, guards, resets

Probability
- discrete probabilistic choice

Model checking PTAs in PRISM

- Properties for PTAs:
  - min/max probability of reaching X (within time T)
  - min/max expected cost/reward to reach X
    (for "linearly-priced" PTAs, i.e. reward gain linear with time)

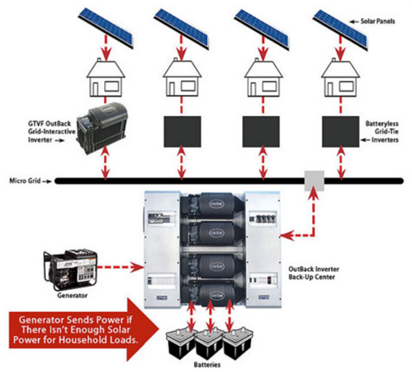
- PRISM has two different PTA model checking techniques…

- "Digital clocks" – conversion to finite-state MDP
  - preserves min/max probability + expected cost/reward/price
  - (for PTAs with closed, diagonal-free constraints)
  - efficient, in combination with PRISM's symbolic engines

- Quantitative abstraction refinement
  - zone-based abstractions of PTAs using stochastic games
  - provide lower/upper bounds on quantitative properties
  - automatic iterative abstraction refinement

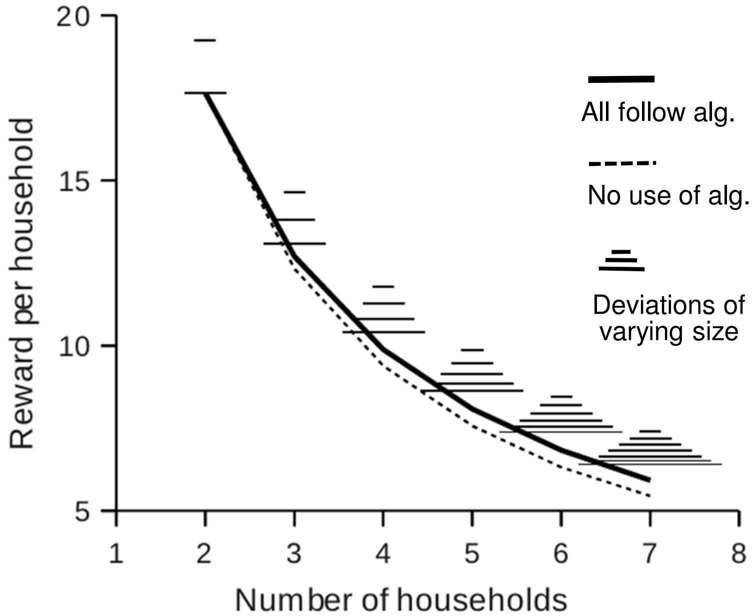
Case study: Energy management

- Energy management protocol for Microgrid
  - Microgrid: local energy management
  - randomised demand management protocol [Hildmann/Saffre'11]
  - probability: randomisation, demand model, …



- Existing analysis
  - simulation-based
  - assumes all clients are unselfish

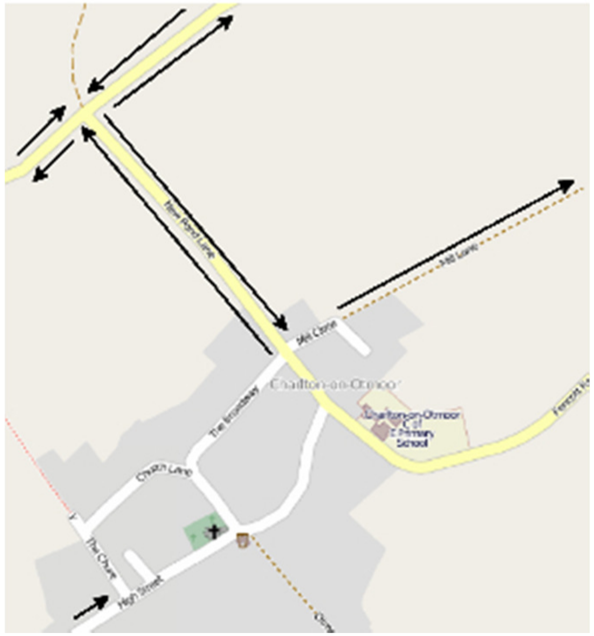
- Our analysis
  - stochastic multi-player game
  - clients can cheat (and cooperate)
  - exposes protocol weakness
  - propose/verify simple fix



Automatic Verification of Competitive Stochastic Systems, Chen et al., In *Proc* TACAS 2012

Case study: Autonomous urban driving

- Inspired by DARPA challenge
  - represent map data as a stochastic game, with environment active, able to select hazards
  - express goals as conjunctions of probabilistic and reward properties
  - e.g. "maximise probability of avoiding hazards and minimise time to reach destination"

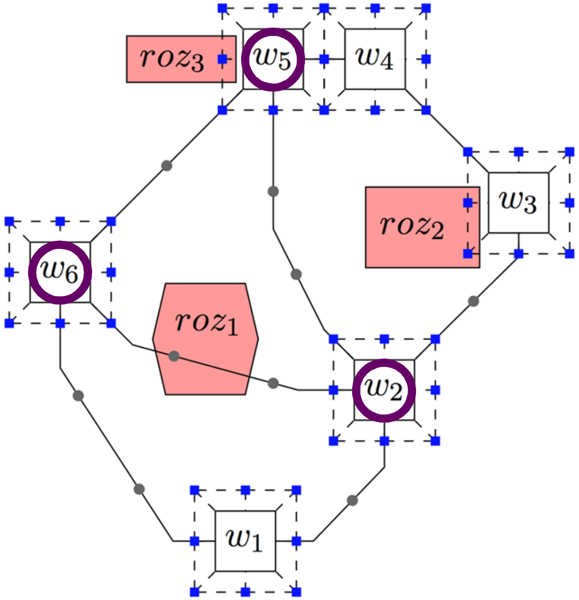


- Solution (PRISM-games 2.0)
  - synthesise a probabilistic strategy to achieve the multi-objective goal
  - enable the exploration of trade-offs between subgoals
  - applied to synthesise driving strategies for English villages

Synthesis for Multi-Objective Stochastic Games: An Application to Autonomous Urban Driving, Chen et al., In *Proc* QEST 2013

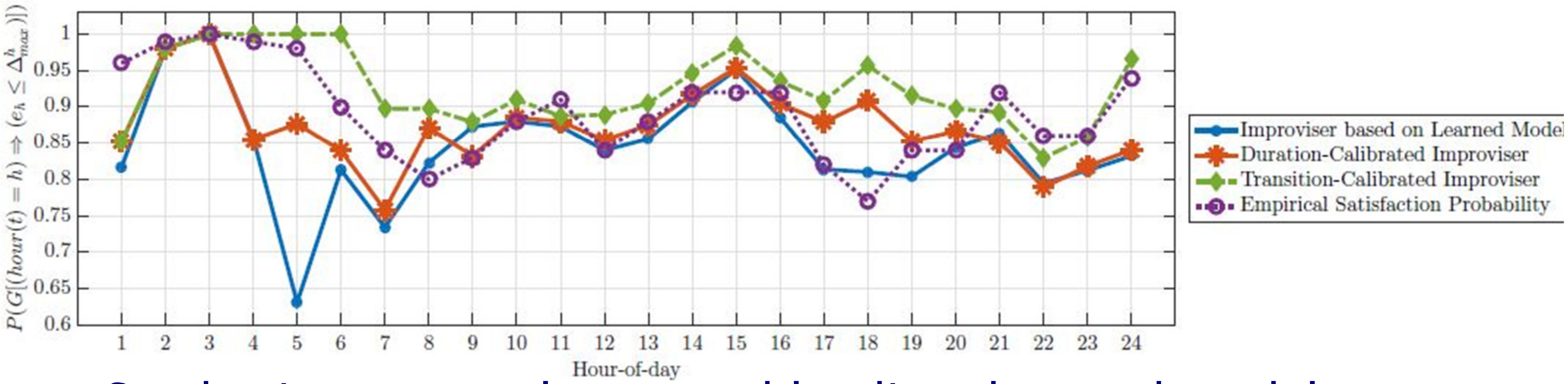
Case study: UAV path planning

- Human operator
  - sensor tasks
  - high-level commands for piloting
- UAV autonomy
  - low-level piloting function
- Quantitative mission objectives
  - road network surveillance with the minimal time, fuel, or restricted operating zone visits
- Analysis of trade-offs
  - consider operator fatigue and workload
  - multi-objective, MDP and SMG models



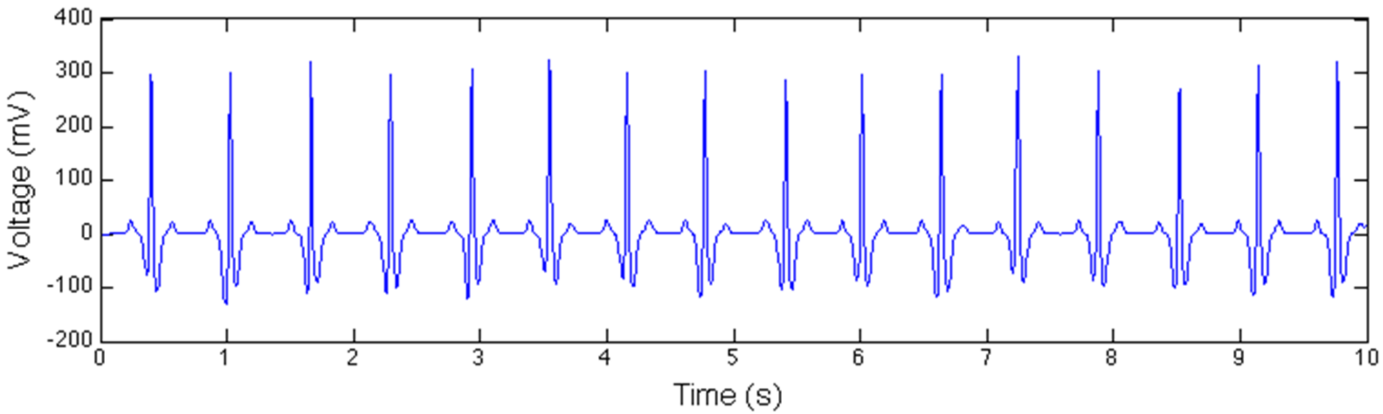
Controller Synthesis for Autonomous Systems Interacting with Human Operators. L. Feng et al, In *Proc*. ICCPS 2015, ACM

Case study: Control improvisation



- Synthesise a control strategy blending data and models
  - hard constraints (that must always be satisfied)
  - soft constraints (that must be “mostly satisfied”)
  - and randomness requirements on system behavior
- Applied PRISM to synthesise strategies for home appliances
  - use PCTL for soft constraints
  - http://arxiv.org/pdf/1511.02279.pdf

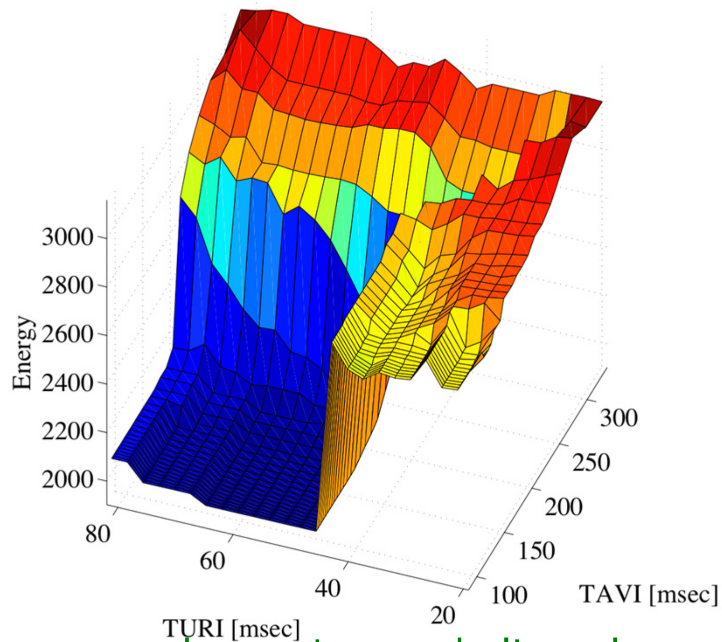
Case study: Personalisation



- Personalisation of wearable devices
  - estimate parameters for a heart model based on ECG data
  - generate synthetic ECG
  - useful for model-based development of personalised devices
- Devoloped HeartVerify based on Simulink/Stateflow
  - variety of tools and techniques
  - http://www.veriware.org/pacemaker.php

Case study: Cardiac pacemaker

- Hybrid model-based framework
  - timed automata model for pacemaker software
  - hybrid heart models in Simulink, adopt synthetic ECG model (non-linear ODE)
- Properties
  - (basic safety) maintain 60-100 beats per minute
  - (advanced) detailed analysis energy usage, plotted against timing parameters of the pacemaker
  - parameter synthesis: find values for timing delays that optimise energy usage

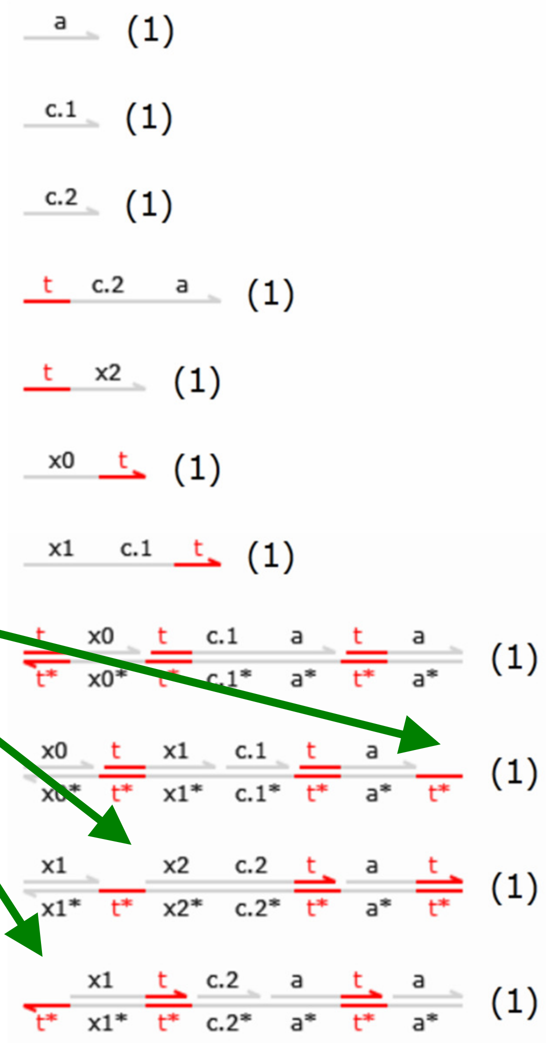


Synthesising robust and optimal parameters for cardiac pacemakers using symbolic and evolutionary computation techniques. Kwiatkowska, Mereacre, Paoletti and Patane, HSB'16

DNA computation

- Cardelli's DNA transducer gate
  - inputs/outputs single strands
  - two transducers connected
- PRISM identifies a bug: 5-step trace to a "bad" deadlock state
  - previously found manually [Cardelli'10]
  - detection now fully automated
- Bug is easily fixed
  - (and verified)

reactive gates

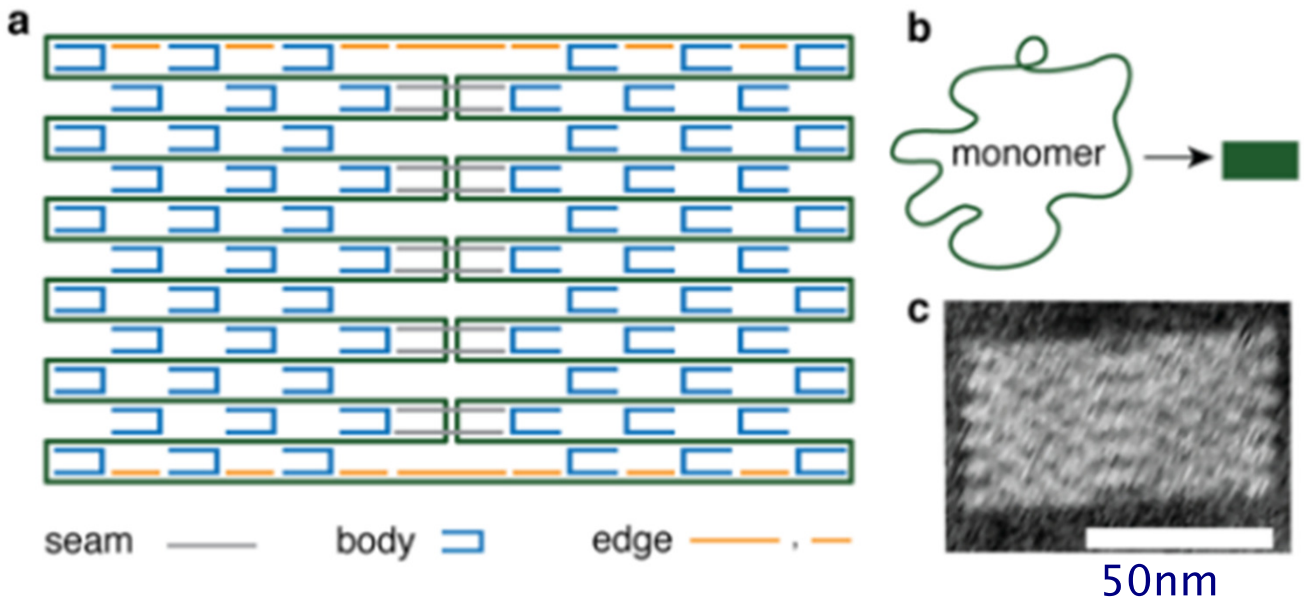


Counterexample:
(1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
(0,1,1,0,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
(0,0,1,0,1,1,1,1,1,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
(0,0,1,0,1,1,1,1,0,0,1,1,1,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
(0,0,1,0,1,1,0,1,0,0,1,1,1,0,0,0,1,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0)
(0,0,1,0,1,1,0,1,0,0,1,0,1,0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0)

Design and Analysis of DNA Strand Displacement Devices using Probabilistic Model Checking, Lakin *et al*, Journal of the Royal Society Interface, 9(72), 1470-1485, 2012

DNA origami tiles

- DNA origami tiles: molecular breadboard [Turberfield lab]

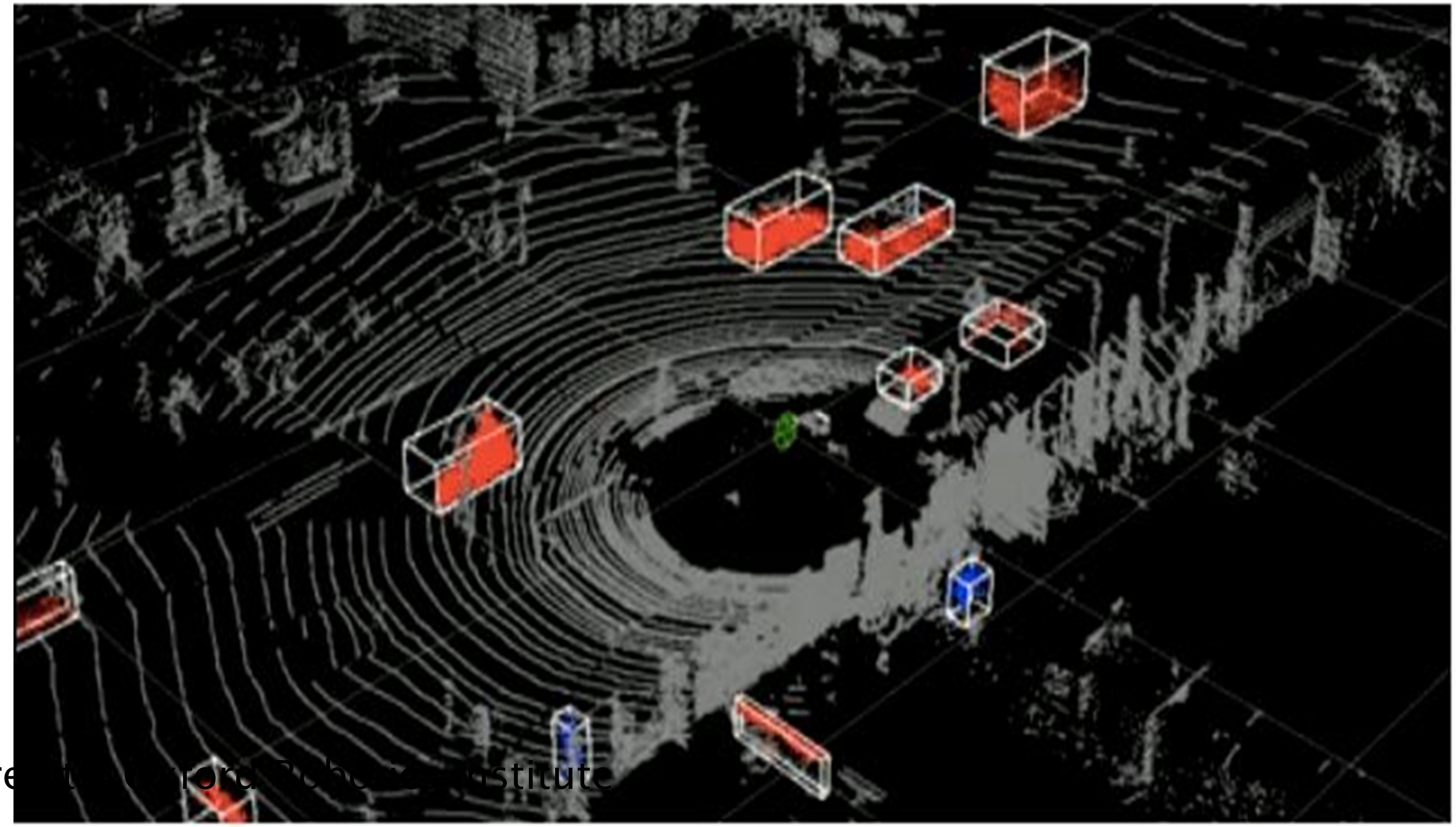


Aim to understand how to control the folding pathways

- formulate an abstract Markov chain model
- obtain model predictions using Gillespie simulation
- perform a range of experiments, consistent with preditions

Guiding the folding pathway of DNA origami. Dunne, Dannenberg, Ouldridge, Kwiatkowska, Turberfield & Bath, Nature 525, pages 82-86, 2015.

Perception software



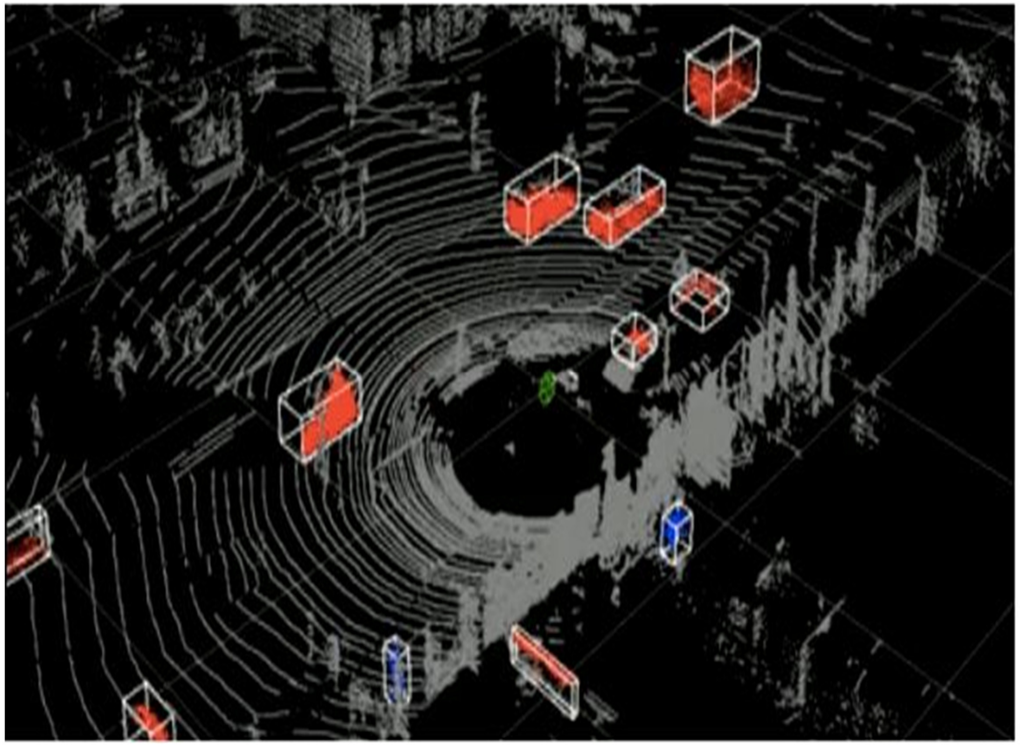
Credit: Oxford Robotics Institute

Things that can go wrong…

- …in perception software
  - sensor failure
  - object detection failure

- Machine learning software
  - not clear how it works
  - does not offer guarantees

– Yet end-to-end solutions are being considered…



Motivating example

- Deep neural network
  - employed as a perception module of an autonomous car
  - must be resilient to image imperfections, change of camera angle, weather, lighting conditions, …

Motivating example

- Deep neural network
  - employed as a perception module of an autonomous car
  - must be resilient to image imperfections, change of camera angle, weather, lighting conditions, …

Deep neural networks can be fooled!

- They are unstable wrt adversarial perturbations
  - often imperceptible changes to the image [Szegedy et al 2014]
  - sometimes artificial white noise
  - potential security risk

- Substantial growth in techniques to evaluate robustness
  - variety of robustness measures, different from risk [Vapnik'91]
  - tools DeepFool [CVPR'16] and constraint-based [NIPS'16]

- This talk: focus on safety and automated verification framework
  - visible and human-recognisable perturbations
  - should not result in class changes
  - tool DLV based on Satisfiability Modulo Theory
  - https://128.84.21.199/abs/1610.06940

Projects

- Several possible topics, happy to discuss
- Modelling, analysis and synthesis
  - driver modelling using PRISM-games
  - autonomous driving using PRISM-games
  - energy -aware protocols using PRISM-games
  - DNA circuits using DSD and PRISM
- Software tool development
  - strategy synthesis using machine learning
- Theory
  - algorithms for model synthesis

- http://www.cs.ox.ac.uk/people/marta.kwiatkowska/research.html