



Probabilistic model checking with PRISM

Marta Kwiatkowska

Department of Computer Science, University of Oxford

4th SSFT, Menlo College, May 2014



Part 2

Markov decision processes

Overview (Part 2)

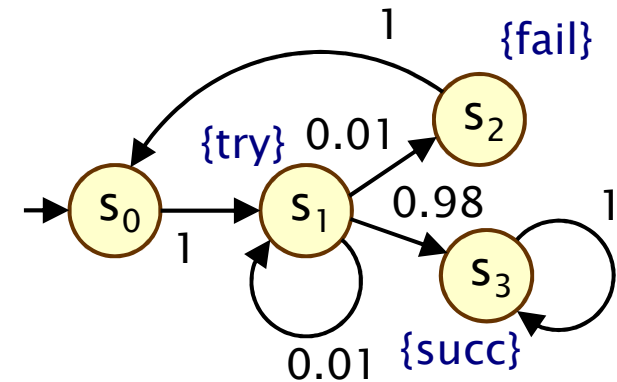
- Introduction
- Model checking for Markov decision processes (MDPs)
 - MDPs: definition
 - Paths, strategies & probability spaces
 - PCTL model checking
 - Costs and rewards
 - Case study: Firewire root contention
- Strategy synthesis for MDPs
 - Properties and objectives
 - Verification vs synthesis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

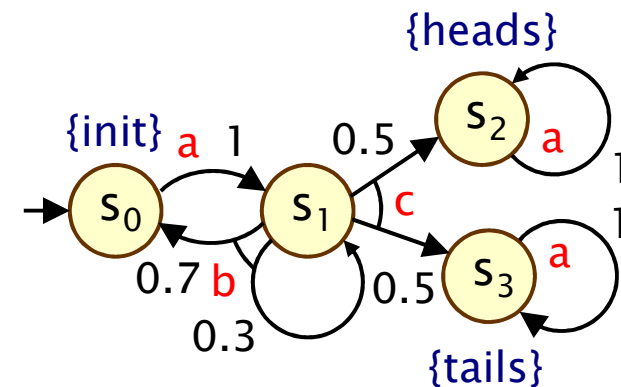


Nondeterminism

- 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

Markov decision processes

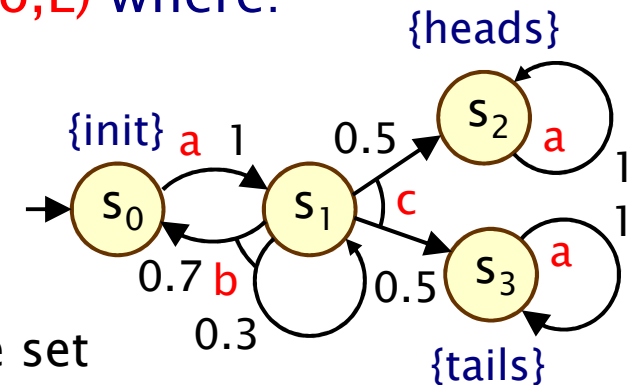
- Markov decision processes (MDPs)
 - extension of DTMCs which allow **nondeterministic choice**
- Like DTMCs:
 - discrete set of states representing possible configurations of the system being modelled
 - transitions between states occur in discrete time-steps
- Probabilities and nondeterminism
 - in each state, a nondeterministic choice between several discrete probability distributions over successor states



Markov decision processes

- Formally, an MDP M is a tuple $(S, s_{\text{init}}, \alpha, \delta, L)$ where:

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

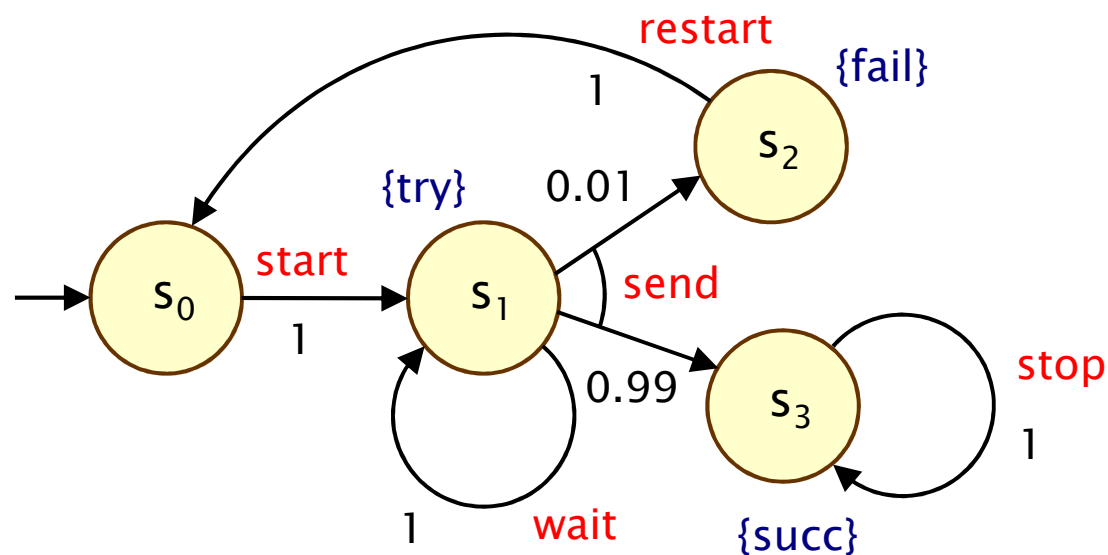


- Notes:

- we also abuse notation and use δ as a function
- i.e. $\delta : S \rightarrow 2^{\alpha \times \text{Dist}(S)}$ 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: $\delta : S \times \alpha \rightarrow \text{Dist}(S)$

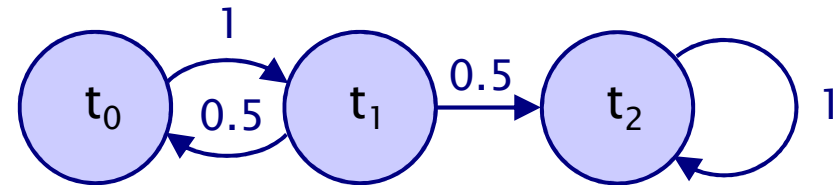
Simple MDP example

- 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**

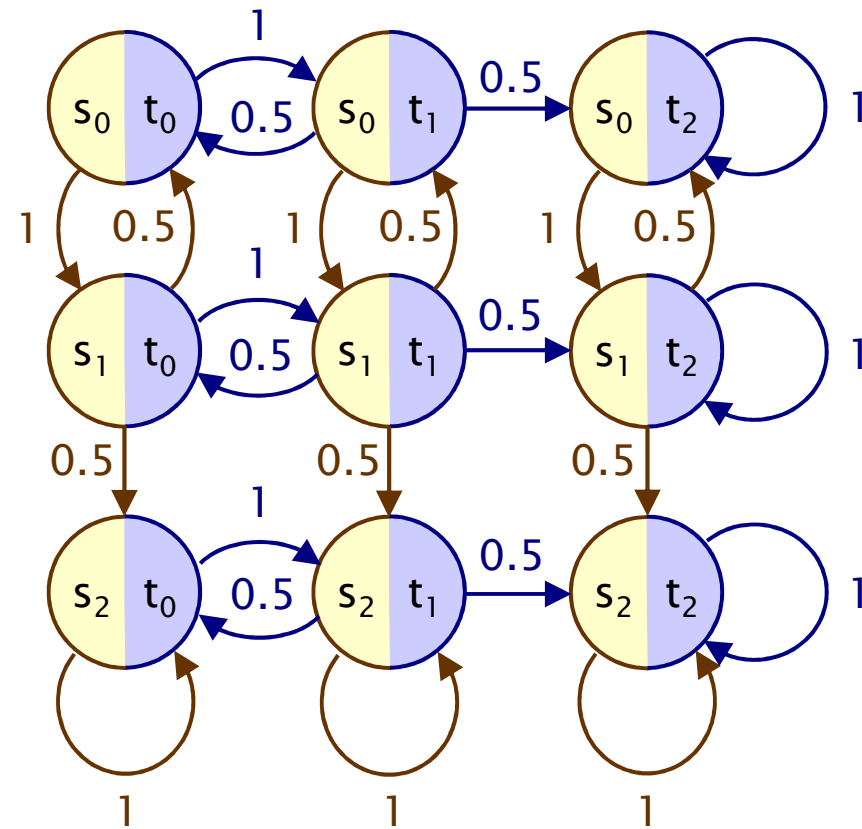
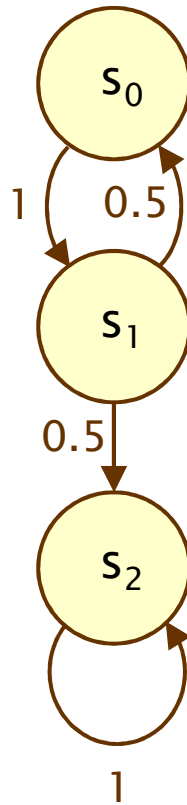


Example – Parallel composition

Asynchronous parallel composition of two 3-state DTMCs



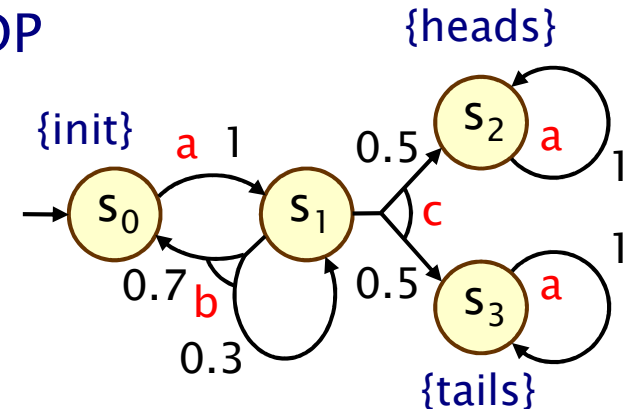
Action labels omitted here



Paths and strategies

- A (finite or infinite) **path** through an MDP

- is a sequence $(s_0 \dots s_n)$ of (connected) states
- represents an execution of the system
- resolves both the probabilistic and nondeterministic choices



- A **strategy** σ (aka. “adversary” or “policy”) of an MDP

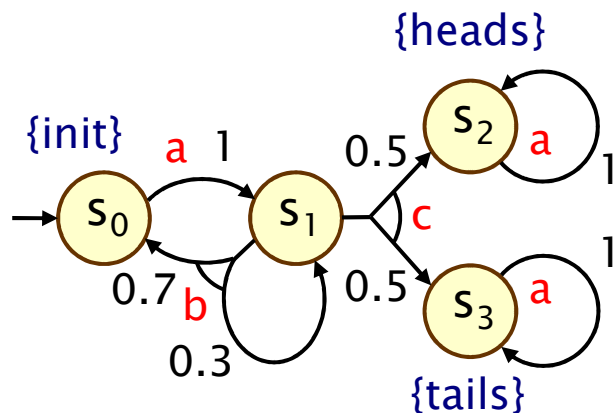
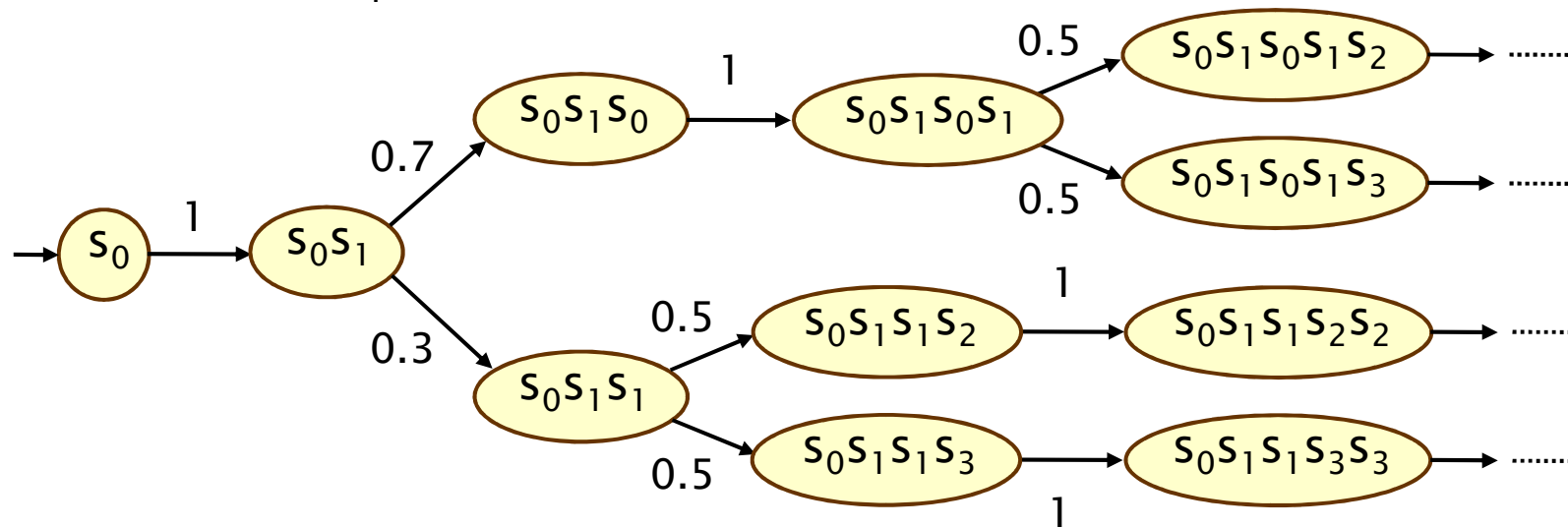
- is a resolution of nondeterminism only
- is (formally) a mapping from finite paths to distributions
- induces a fully probabilistic model
- i.e. an (infinite–state) Markov chain over finite paths
- on which we can define a probability space over infinite paths

Classification of strategies

- Strategies are classified according to
- randomisation:
 - σ is **deterministic** (pure) if $\sigma(s_0 \dots s_n)$ is a point distribution, and **randomised** otherwise
- memory:
 - σ is **memoryless** (simple) if $\sigma(s_0 \dots s_n) = \sigma(s_n)$ for all $s_0 \dots s_n$
 - σ is **finite memory** if there are finitely many modes such as $\sigma(s_0 \dots s_n)$ 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 $^\sigma_s$** over **Path $^\sigma$ (s)**

Example strategy

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



finite-memory,
deterministic

PCTL

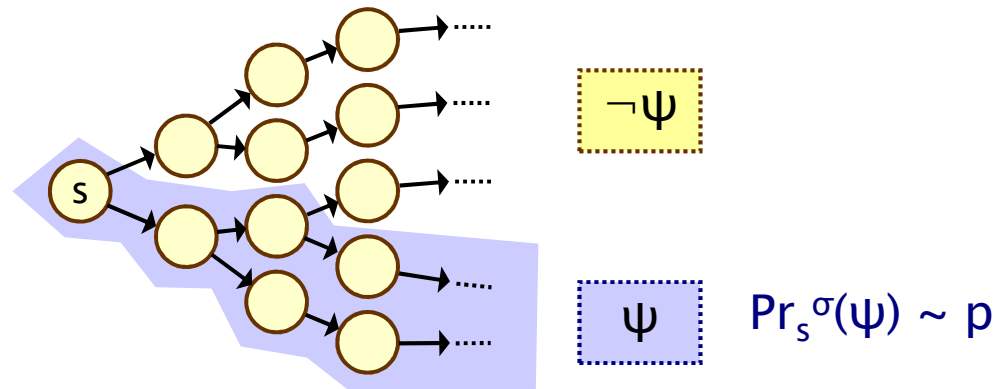
- Temporal logic for properties of MDPs (and DTMCs)
 - extension of (non-probabilistic) temporal logic CTL
 - key addition is **probabilistic operator P**
 - quantitative extension of CTL's A and E operators
- PCTL syntax:
 - $\phi ::= \text{true} \mid a \mid \phi \wedge \phi \mid \neg\phi \mid P_{\sim p} [\psi]$ (state formulas)
 - $\psi ::= X\phi \mid \phi U^{\leq k} \phi \mid \phi U \phi$ (path formulas)
 - 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:** $\text{send} \rightarrow P_{\geq 0.95} [\text{true} U^{\leq 10} \text{deliver}]$

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 $(S, s_{init}, \alpha, \delta, L)$:
 - $s \models a \iff a \in L(s)$
 - $s \models \phi_1 \wedge \phi_2 \iff s \models \phi_1 \text{ and } s \models \phi_2$
 - $s \models \neg\phi \iff s \models \phi \text{ is false}$
- Semantics of path formulas:
 - for a path $\omega = s_0(a_0, \mu_0)s_1(a_1, \mu_1)s_2\dots$ in the MDP:
 - $\omega \models X\phi \iff s_1 \models \phi$
 - $\omega \models \phi_1 U^{\leq k} \phi_2 \iff \exists i \leq k \text{ such that } s_i \models \phi_2 \text{ and } \forall j < i, s_j \models \phi_1$
 - $\omega \models \phi_1 U \phi_2 \iff \exists k \geq 0 \text{ such that } \omega \models \phi_1 U^{\leq k} \phi_2$

PCTL semantics for MDPs

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



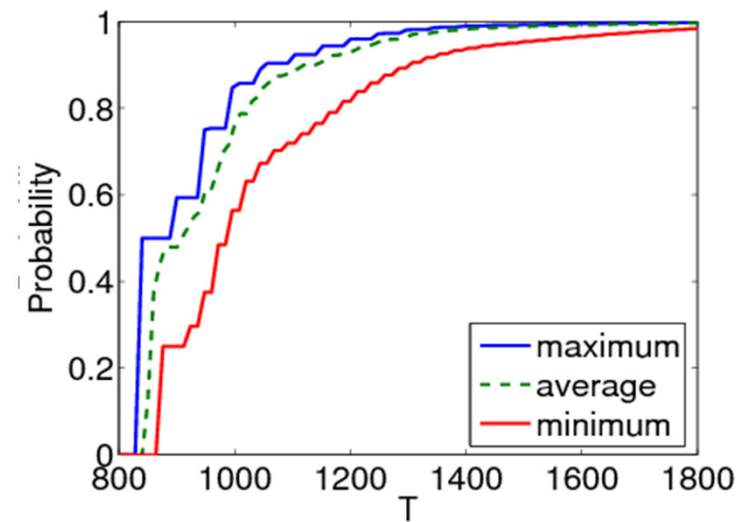
- Some equivalences:
 - $F \phi \equiv \diamond \phi \equiv \text{true} \cup \phi$ (eventually, “future”)
 - $G \phi \equiv \square \phi \equiv \neg(F \neg\phi)$ (always, “globally”)

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 $s \models P_{\sim p}[\psi] \Leftrightarrow \Pr_s^{\min}(\psi) \sim p$
 - if $\sim \in \{<, \leq\}$, then $s \models P_{\sim p}[\psi] \Leftrightarrow \Pr_s^{\max}(\psi) \sim p$
- 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 $\Pr_s^{\sigma_{\max}}(\psi) = \Pr_s^{\max}(\psi)$

Quantitative properties

- For PCTL properties with P as the outermost operator
 - quantitative form (two types): $P_{\min=?} [\psi]$ and $P_{\max=?} [\psi]$
 - 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”



Some real PCTL examples

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

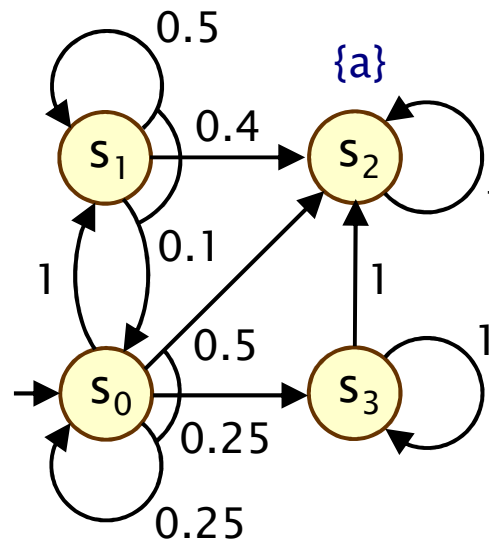
PCTL model checking for MDPs

- Algorithm for PCTL model checking [BdA95]
 - inputs: MDP $M=(S,s_{init},\alpha,\delta,L)$, PCTL formula ϕ
 - output: $\text{Sat}(\phi) = \{ s \in S \mid s \models \phi \}$ = 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 $P_{\sim p} [\psi]$ formulas
 - reduces to computation of $\text{Pr}_s^{\min}(\psi)$ or $\text{Pr}_s^{\max}(\psi)$ for all $s \in S$
 - dependent on whether $\sim \in \{\geq, >\}$ or $\sim \in \{<, \leq\}$
 - these slides cover the case $\text{Pr}_s^{\min}(\phi_1 \text{ U } \phi_2)$, i.e. $\sim \in \{\geq, >\}$
 - case for maximum probabilities is very similar
 - next ($X \phi$) and bounded until ($\phi_1 \text{ U}^{\leq k} \phi_2$) 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 $S^{\text{yes}}, S^{\text{no}}$
- Then compute (min) probabilities for remaining states ($S^?$)
 - either: solve linear programming problem
 - or: approximate with an iterative solution method
 - or: use policy iteration

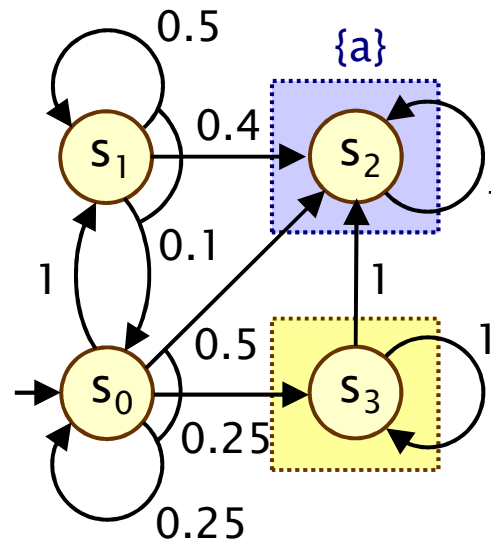
Example:
 $P_{\geq p} [F a]$
 \equiv
 $P_{\geq p} [\text{true U } a]$



PCTL until – Precomputation

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



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

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

Method 1 – Linear programming

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

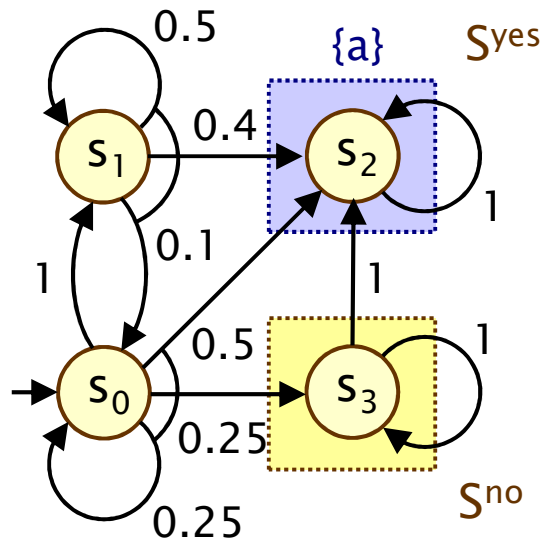
maximize $\sum_{s \in S^?} x_s$ subject to the constraints :

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

for all $s \in S^?$ 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 $x_i = \Pr_{s_i}^{\min}(F a)$

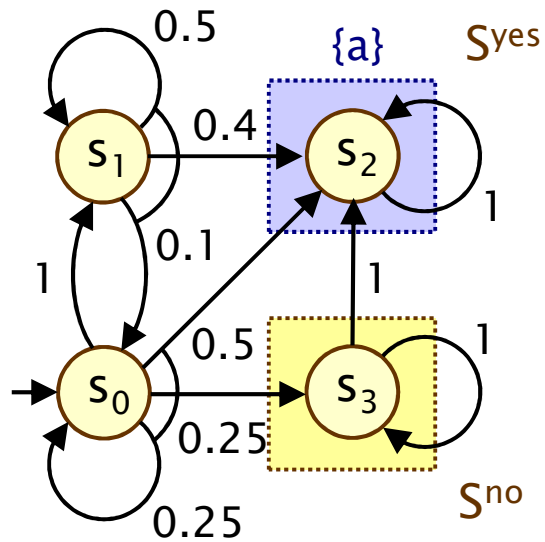
S^{yes} : $x_2=1$, S^{no} : $x_3=0$

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)



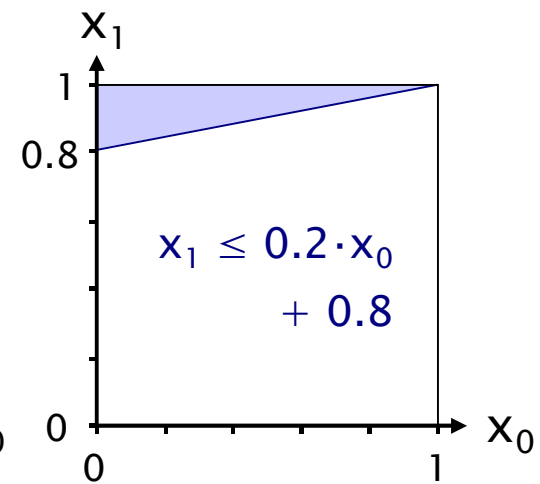
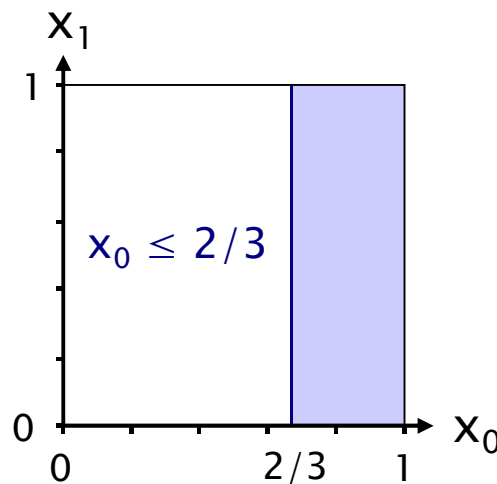
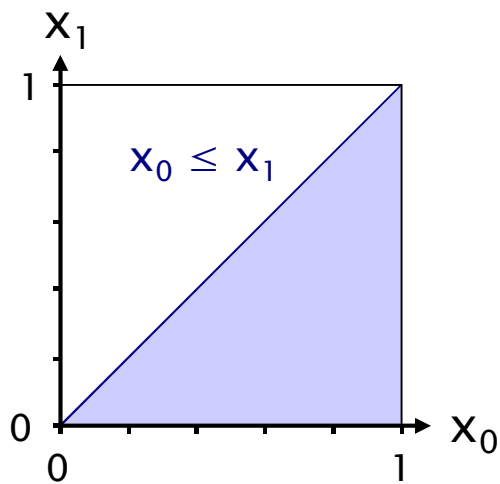
Let $x_i = \Pr_{s_i}^{\min}(F a)$

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

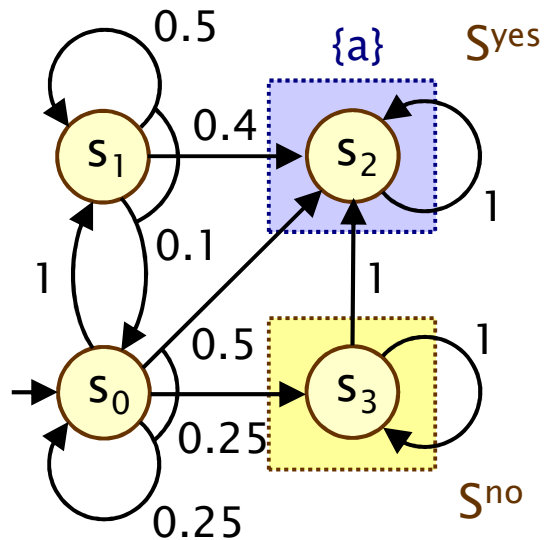
For $S^? = \{x_0, x_1\}$:

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)



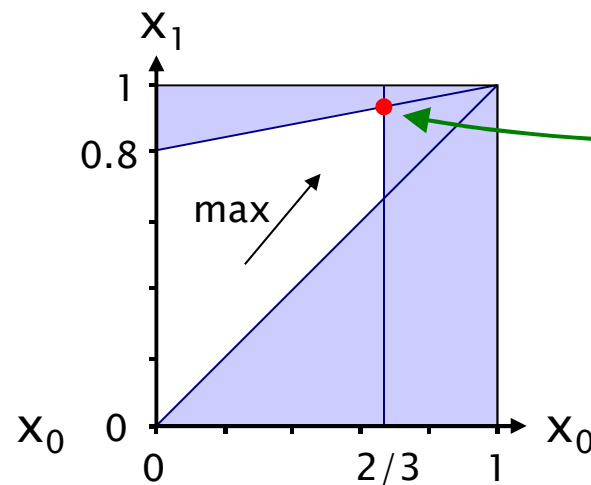
Let $x_i = \Pr_{s_i}^{\min}(F a)$

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

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

Maximise x_0+x_1 subject to constraints:

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



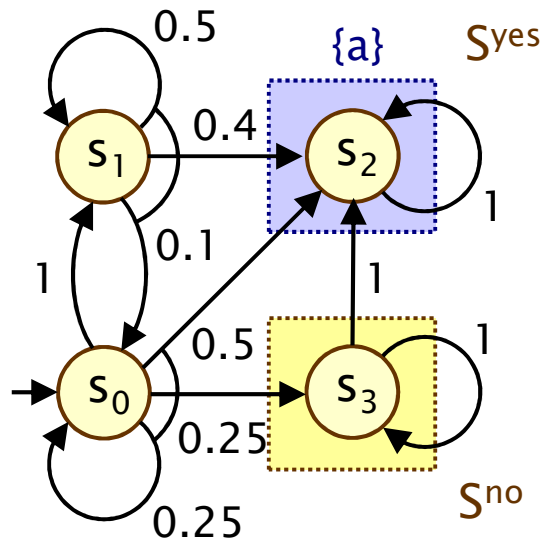
Solution:

(x_0, x_1)

=

$(2/3, 14/15)$

Example – PCTL until (LP)



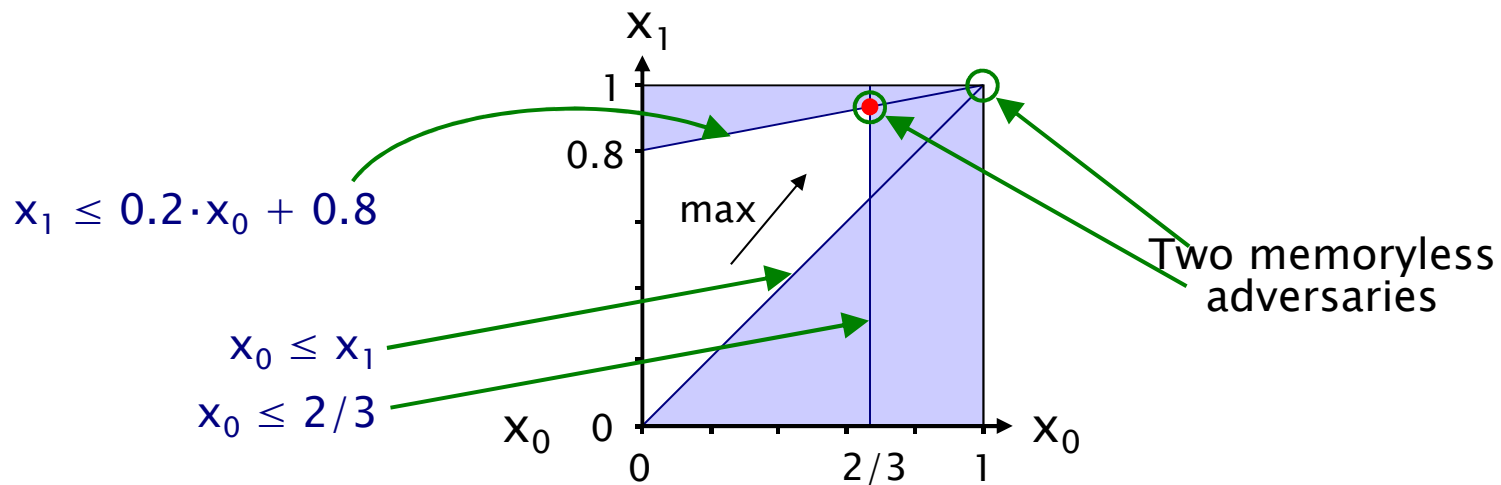
Let $x_i = \Pr_{s_i}^{\min}(F a)$

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

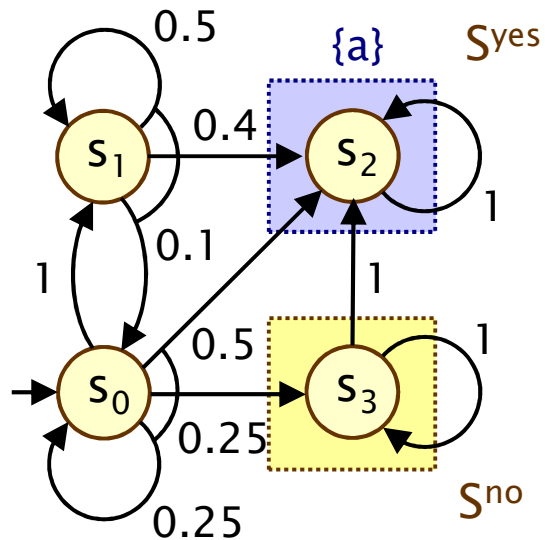
- For probabilities $\Pr_s^{\min}(\phi_1 \cup \phi_2)$ it can be shown that:

– $\Pr_s^{\min}(\phi_1 \cup \phi_2) = \lim_{n \rightarrow \infty} x_s^{(n)}$ where:

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

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

Example – PCTL until (value iteration)



Compute: $\Pr_{s_i}^{\min}(F a)$

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

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

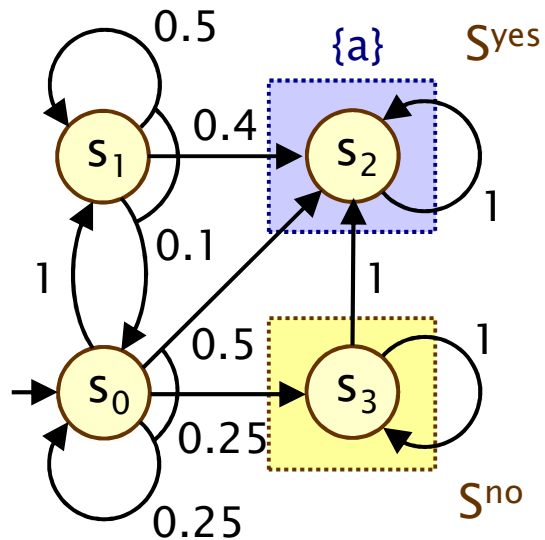
n=0: $[0, 0, 1, 0]$

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

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

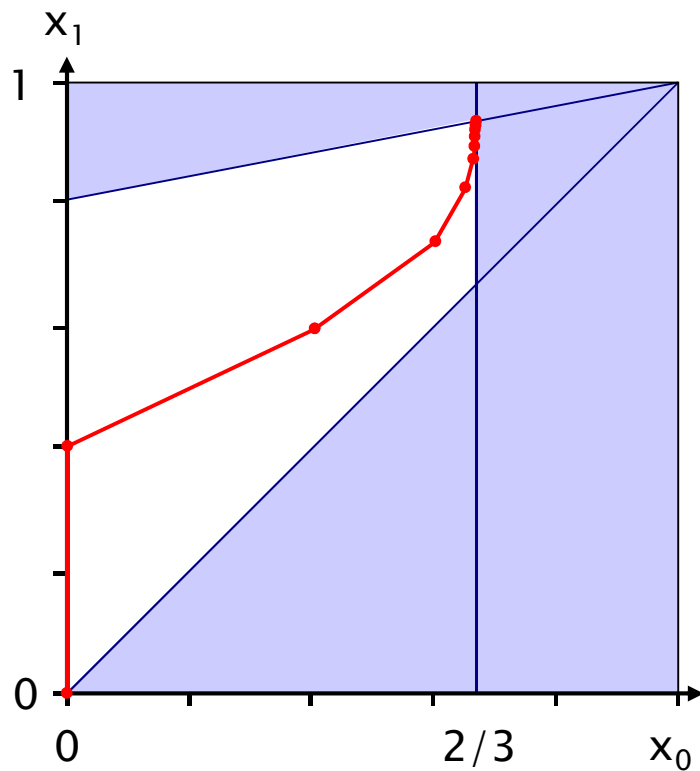
n=3: ...

Example – PCTL until (value iteration)



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

Example – Value iteration + LP



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

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

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 $\Pr^\sigma(F \text{ a})$ 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

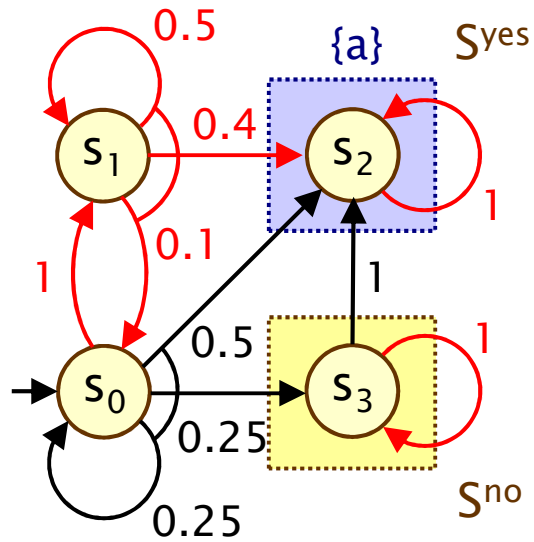
Method 3 – Policy iteration

- 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 $\Pr^\sigma(F a)$ for σ
 - probabilistic reachability on a DTMC
 - i.e. solve linear equation system
- 3. Improve the adversary in each state

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

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

Example – Policy iteration



Arbitrary strategy σ :

Compute: $\Pr^\sigma(F a)$

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:

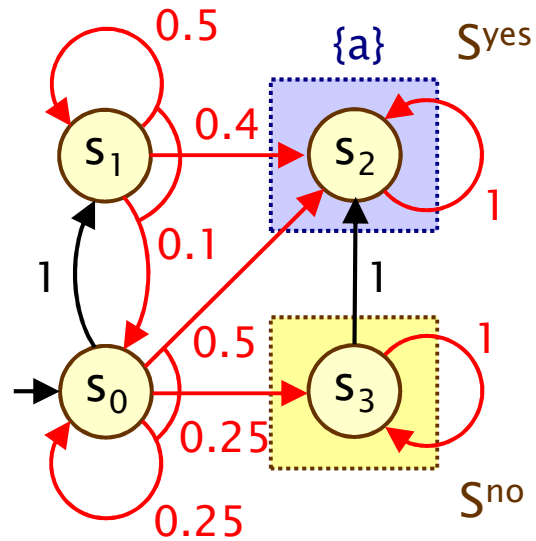
$$\Pr^\sigma(F a) = [1, 1, 1, 0]$$

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

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

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

- $x_0 = 0.25 \cdot x_0 + 0.5$

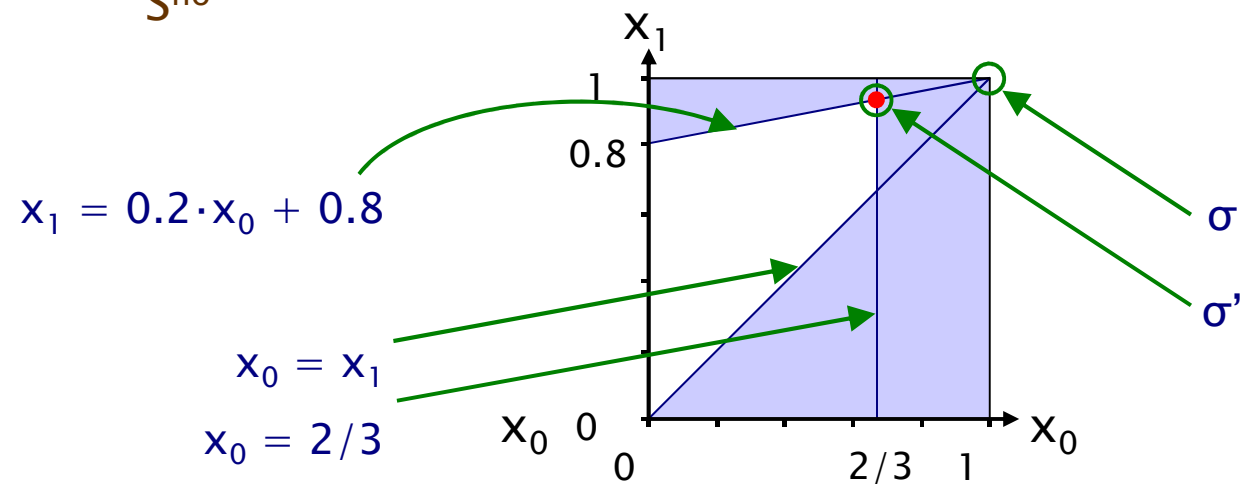
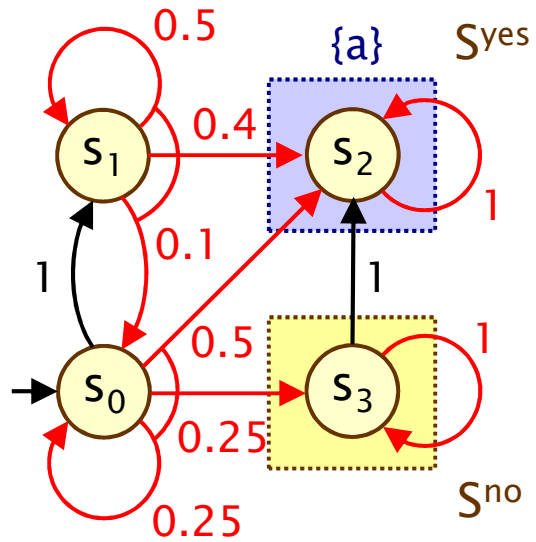
- $x_1 = 0.1 \cdot x_0 + 0.5 \cdot x_1 + 0.4$

Solution:

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

This is optimal

Example – Policy iteration



PCTL model checking – Summary

- Computation of set $\text{Sat}(\Phi)$ for MDP M and PCTL formula Φ
 - recursive descent of parse tree
 - combination of graph algorithms, numerical computation
- Probabilistic operator P :
 - $X \Phi$: one matrix–vector multiplication, $O(|S|^2)$
 - $\Phi_1 U^{\leq k} \Phi_2$: k matrix–vector multiplications, $O(k|S|^2)$
 - $\Phi_1 U \Phi_2$: linear programming problem, **polynomial in $|S|$** (assuming use of linear programming)
- Complexity:
 - **linear in $|\Phi|$** and **polynomial in $|S|$**
 - S is states in MDP, assume $|\delta(s)|$ 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 $R_{\sim r} [\dots]$, $R_{\min=?} [\dots]$ or $R_{\max=?} [\dots]$
- Some examples:
 - $R_{\min=?} [I^{=90}]$, $R_{\max=?} [C^{\leq 60}]$, $R_{\max=?} [F \text{ “end”}]$
 - “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)

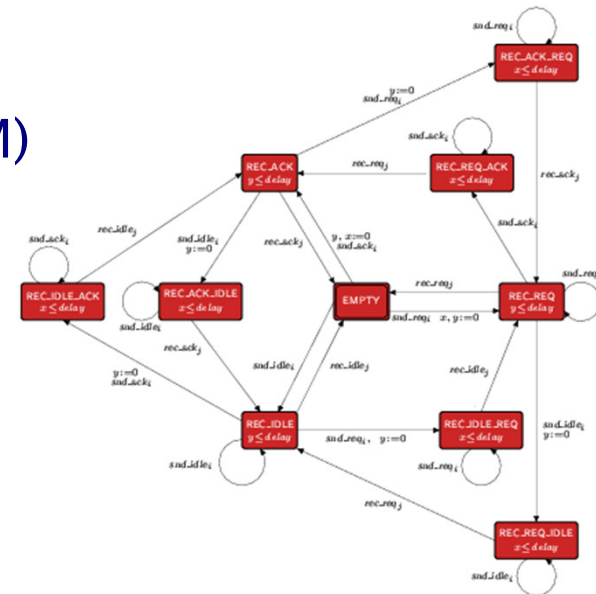
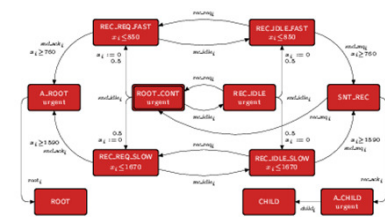


- Root contention protocol

- 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

Case study: FireWire root contention

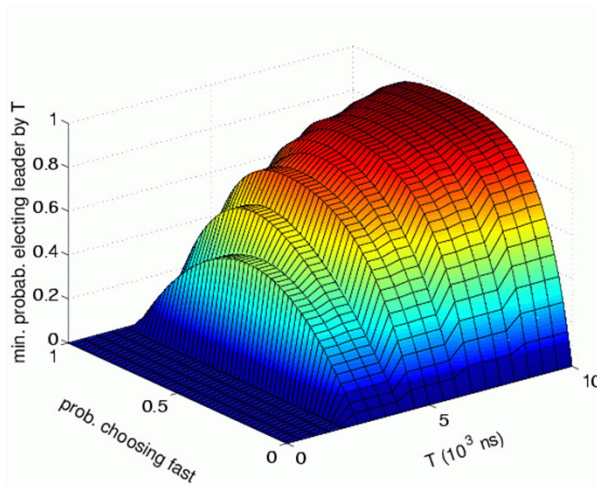
- Detailed probabilistic model:
 - probabilistic timed automaton (PTA), including:
 - concurrency: messages between nodes and wires
 - timing delays taken from official standard
 - underspecification of delays (upper/lower bounds)
 - maximum model size: 170 million states
- Probabilistic model checking (with PRISM)
 - verified that root contention always resolved with probability 1
 - $P_{\geq 1} [F (\text{end} \wedge \text{elected})]$
 - investigated worst-case expected time taken for protocol to complete
 - $R_{\max=?} [F (\text{end} \wedge \text{elected})]$
 - investigated the effect of using biased coin



Case study: FireWire root contention

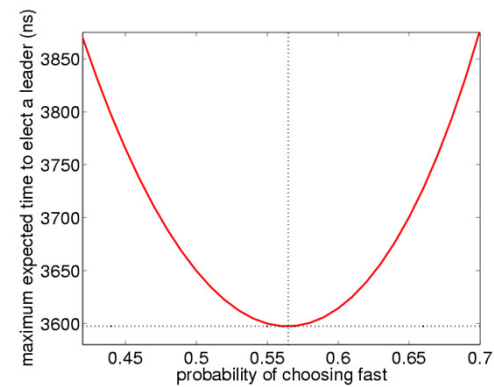
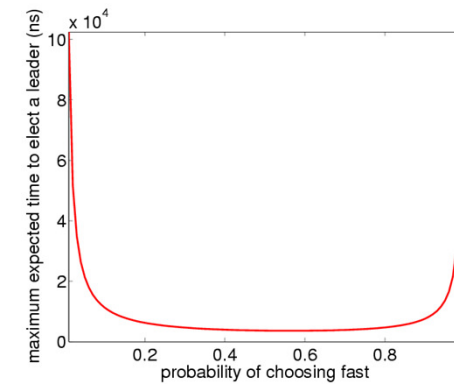
“minimum probability of electing leader by time T”

(using a biased coin)



“maximum expected time to elect a leader”

(using a biased coin)

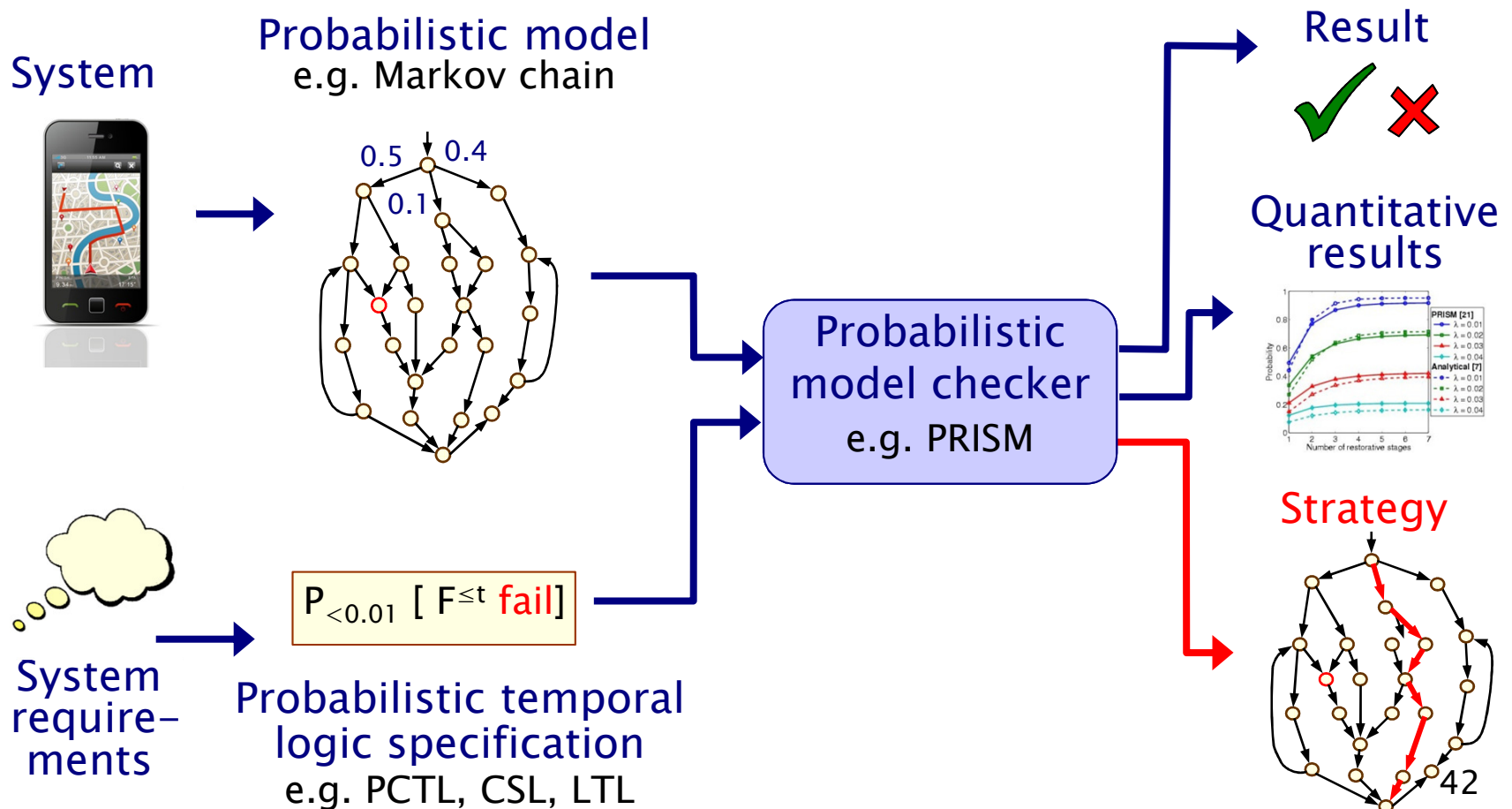


Verification vs synthesis

- Majority of research to date has focused on verification
 - scalability and performance of algorithms
 - extending expressiveness of models and logics
- Some work to date on counterexamples
 - but difficult to represent them compactly
- In this lecture, focus on simpler problem of **strategy synthesis**
 - can we find a strategy to guarantee that a given quantitative property is satisfied?
 - advantage: **correct-by-construction**
- Not a well known fact...
 - incidentally, can **reuse** the verification algorithms...

Quantitative (probabilistic) verification

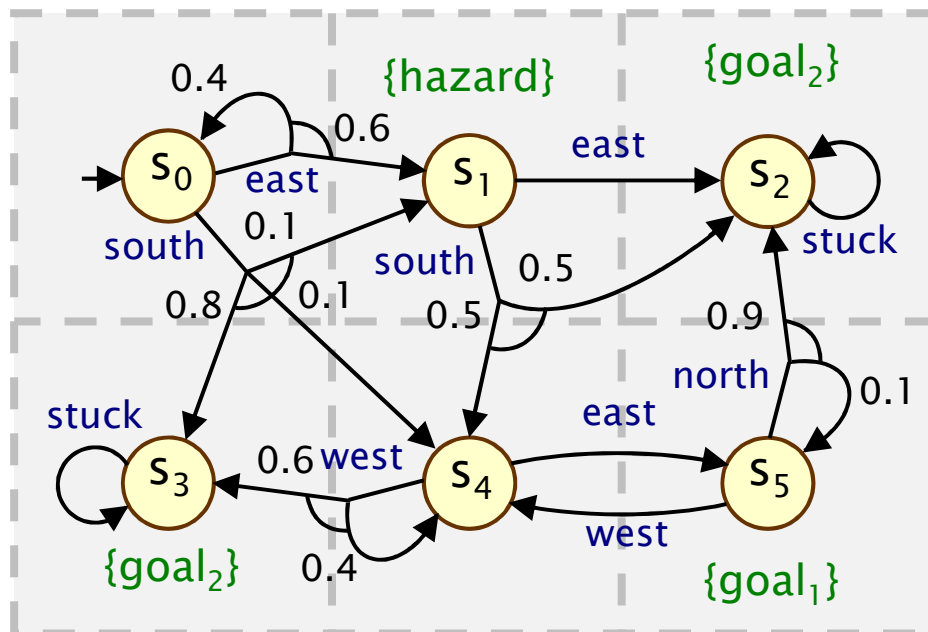
Automatic verification and **strategy synthesis** from quantitative properties for probabilistic models



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

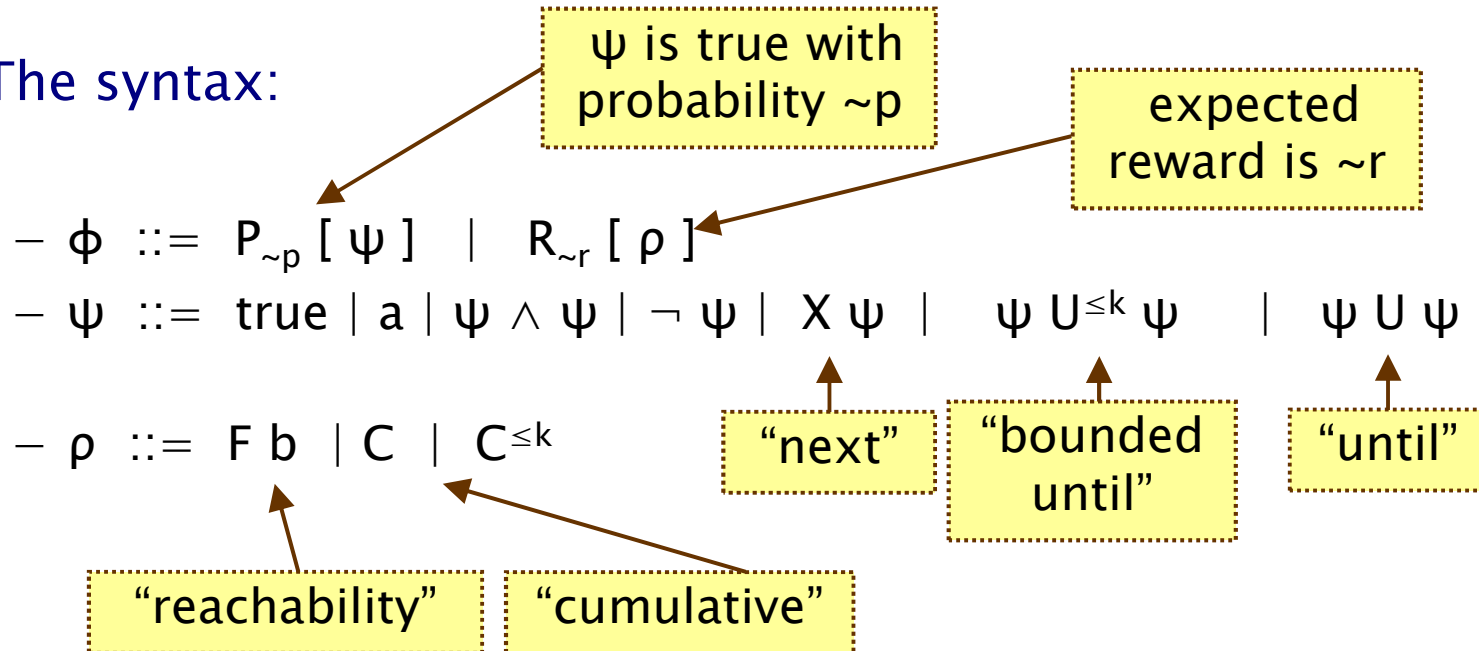
Labels

(atomic propositions):

hazard, goal₁, goal₂

Properties and objectives

- The syntax:



- 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**
 - $s \models P_{\sim p} [\psi]$ means “the probability, from state s , that ψ is true for an outgoing path satisfies $\sim p$ **for all strategies σ** ”
 - formally $s \models P_{\sim p} [\psi] \Leftrightarrow \Pr_s^\sigma(\psi) \sim p$ for all strategies σ
 - where we use $\Pr_s^\sigma(\psi)$ to denote $\Pr_s^\sigma \{ \omega \in \text{Path}_s^\sigma \mid \omega \models \psi \}$
- $R_{\sim r} [\cdot]$ means “the **expected value** of \cdot satisfies $\sim r$ ”
- Some examples:
 - $P_{\geq 0.4} [F \text{ “goal”}]$ “probability of reaching goal is at least 0.4”
 - $R_{<5} [C^{\leq 60}]$ “expected power consumption over one hour is **below 5**”
 - $R_{\leq 10} [F \text{ “end”}]$ “expected time to termination is at most 10”

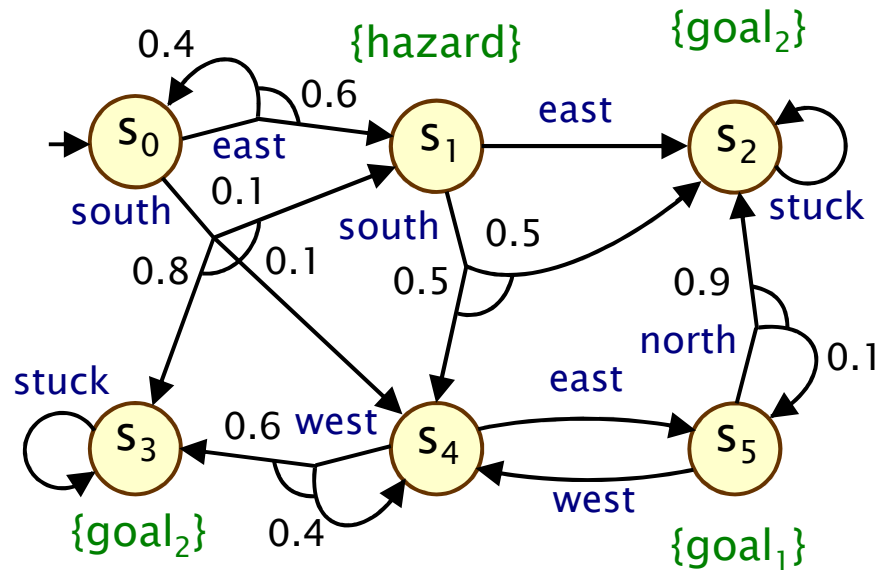
Verification and strategy synthesis

- 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 the same techniques, 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 $S^{\text{yes}}, S^{\text{no}}$
- Step 2: **compute** probabilities for remaining states ($S^?$)
 - (i) solve linear programming problem
 - (ii) 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



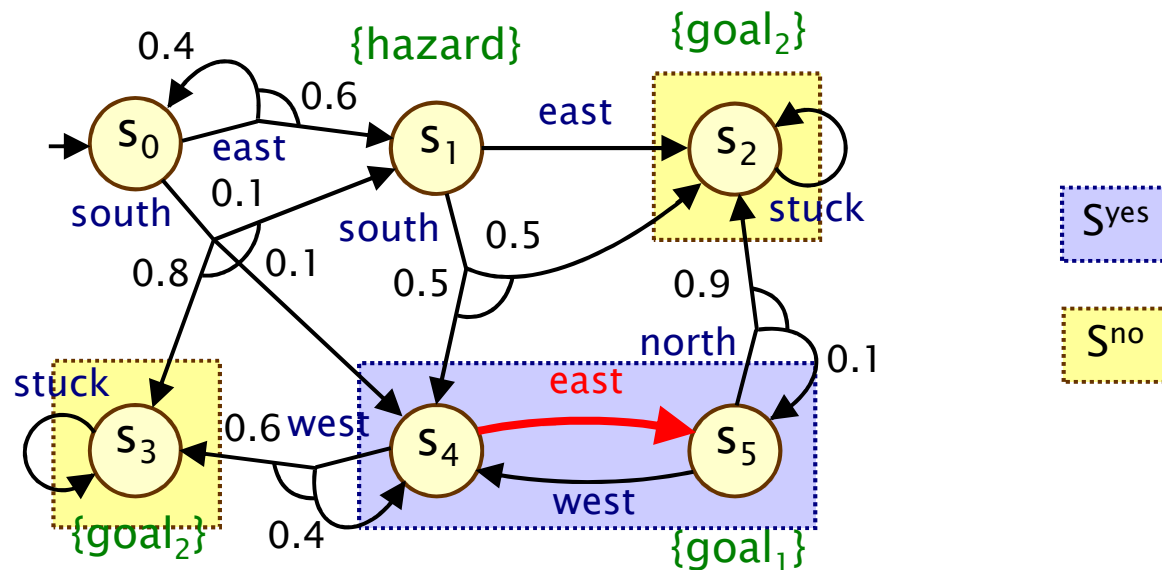
Example:

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

So compute:

$$\Pr_S^{\max}(F \text{ goal}_1)$$

Example – Precomputation



Example:

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

So compute:

$$\Pr_S^{\max}(F \text{ goal}_1)$$

Reachability for MDPs

- 2. Numerical computation

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

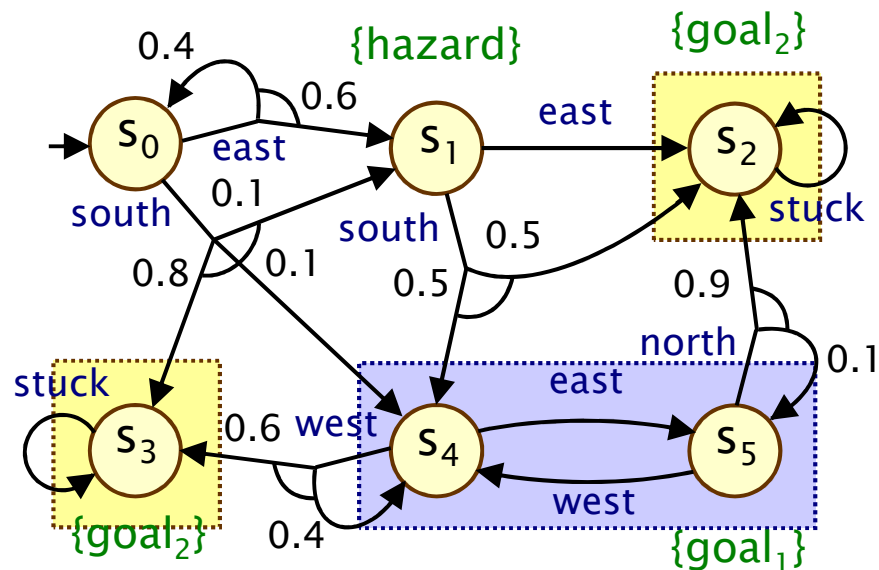
minimize $\sum_{s \in S^?} x_s$ subject to the constraints:

$$x_s \geq \sum_{s' \in S^?} \delta(s, a)(s') \cdot x_{s'} + \sum_{s' \in S^{\text{yes}}} \delta(s, a)(s')$$

for all $s \in S^?$ and for all $a \in A(s)$

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

Example – Reachability (LP)



Example:

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

So compute:

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

Let $x_i = \Pr_{s_i}^{\max}(F \text{ goal}_1)$

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

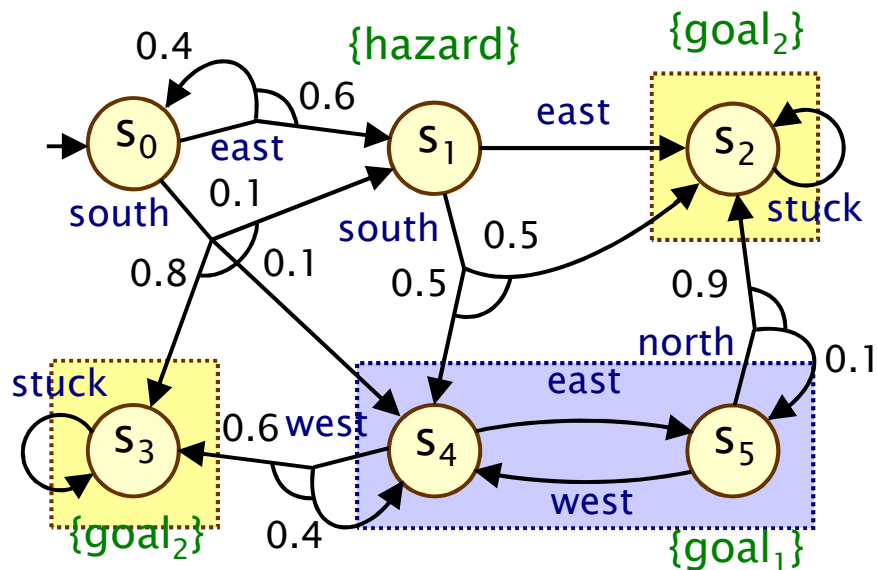
S^{no} : $x_2 = x_3 = 0$

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 $x_i = \Pr_{s_i}^{\max}(F \text{ goal}_1)$

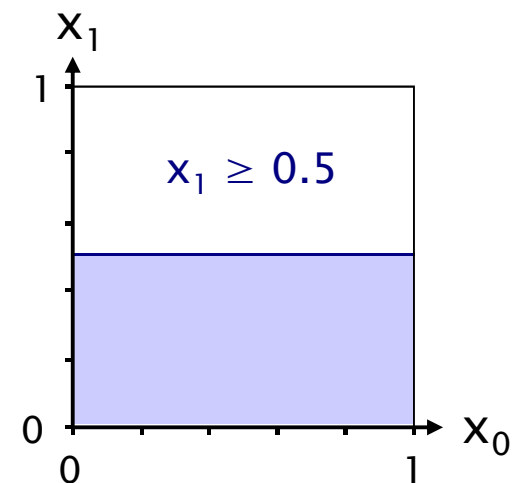
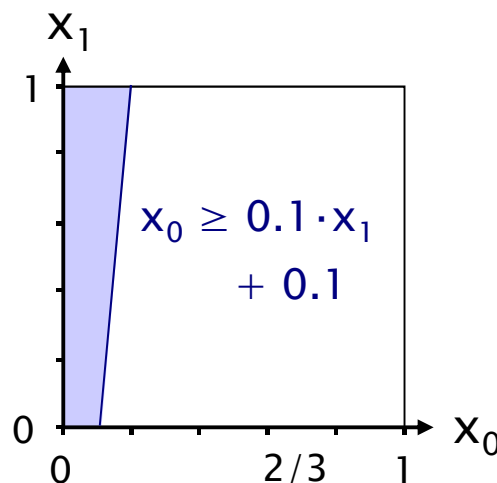
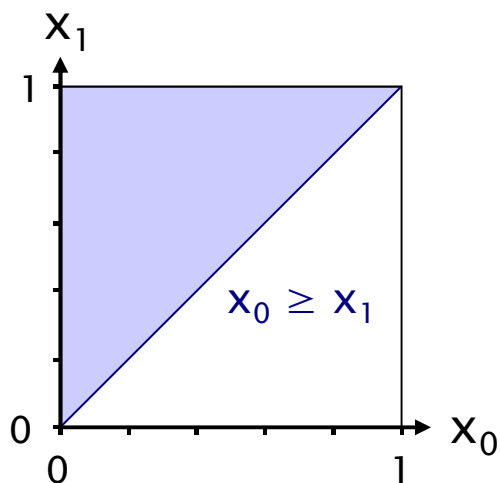
S^{yes} : $x_4 = x_5 = 1$

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

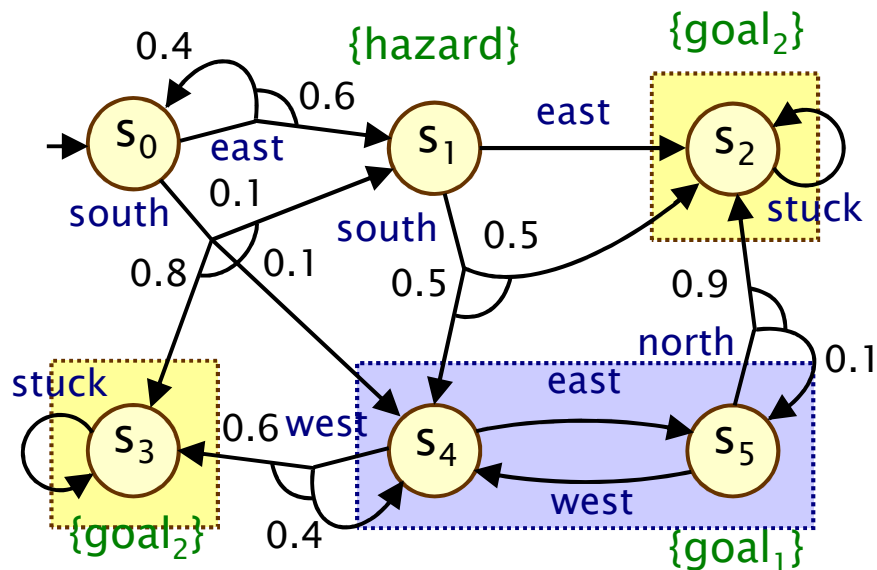
For $S^? = \{x_0, x_1\}$:

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 $x_i = \Pr_{s_i}^{\max}(F \text{ goal}_1)$

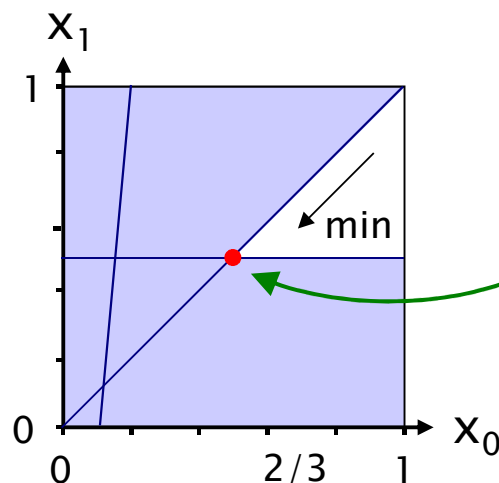
S^{yes} : $x_4 = x_5 = 1$

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.

$$\Pr_{s_0}^{\max}(F \text{ goal}_1) = 0.5$$

Reachability for MDPs

- 2. Numerical computation (alternative method)

- value iteration

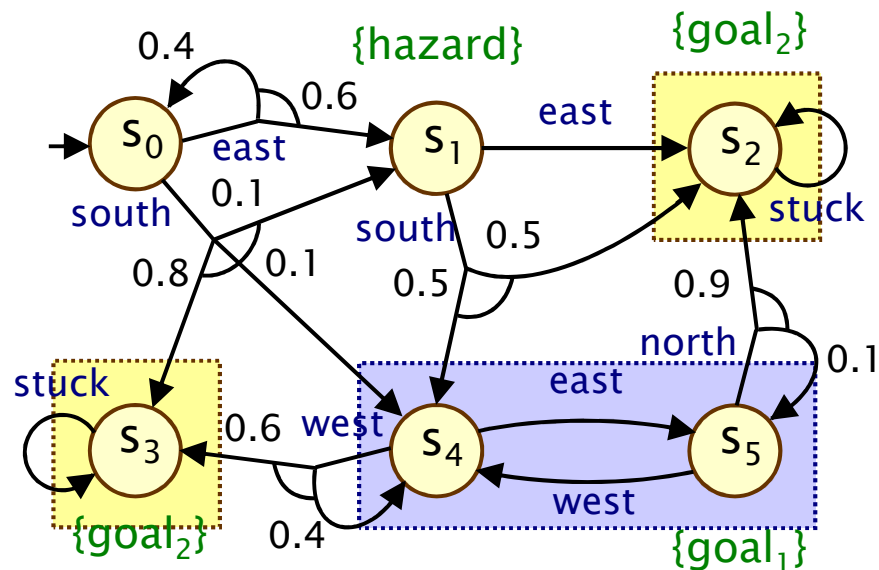
- it can be shown that: $\Pr_s^{\max}(F b) = \lim_{n \rightarrow \infty} x_s^{(n)}$ where:

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

- Approximate iterative solution technique

- iterations terminated when solution converges sufficiently

Example – Reachability (val. iter.)



Compute: $\Pr_s^{\max}(F \text{ goal}_1)$

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

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

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

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

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

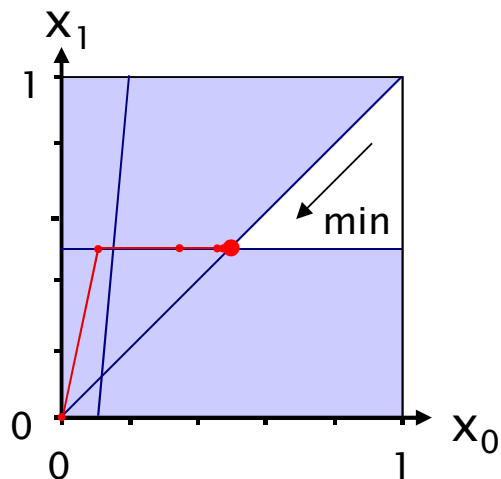
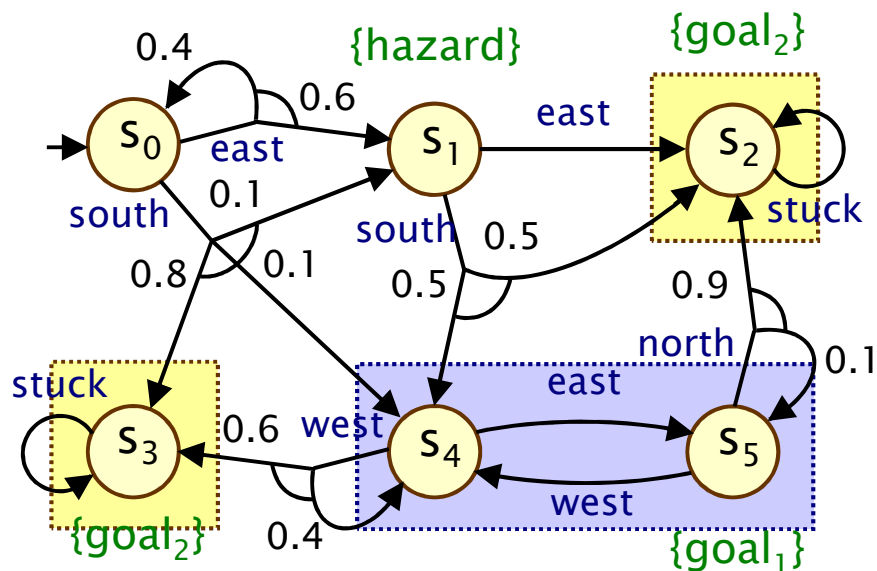
$$n=1: [\max(0.6 \cdot 0 + 0.4 \cdot 0, 0.1 \cdot 0 + 0.1 \cdot 1 + 0.8 \cdot 0), \max(0, 0.5), 0, 0, 1, 1]$$

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

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

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

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]

... $\approx [0.5 \ 0.5, 0, 0, 1, 1]$

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:

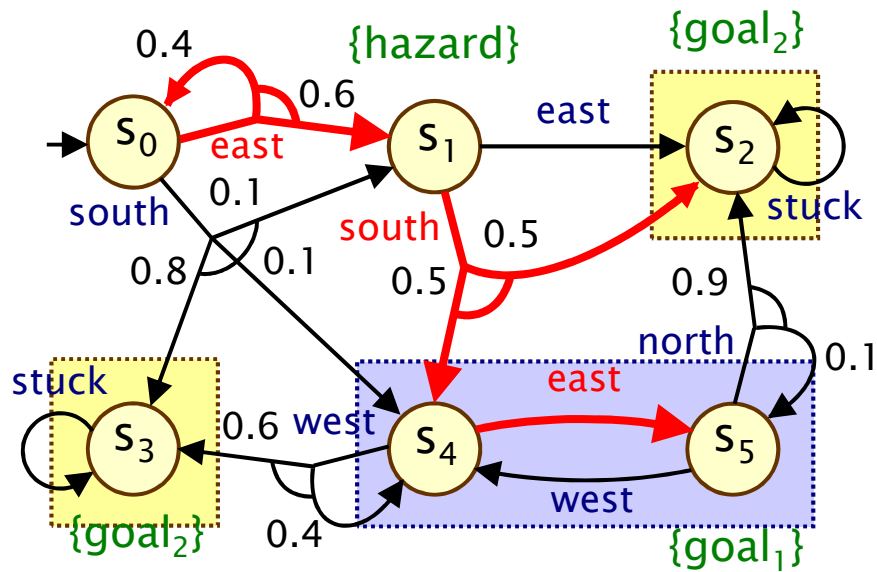
$$\sigma_{\min}(s) = \operatorname{argmin} \left\{ \sum_{s' \in S} \mu(s') \cdot p_{\min}(s', F a) \mid (a, \mu) \in \text{Steps}(s) \right\}$$

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

Strategy synthesis

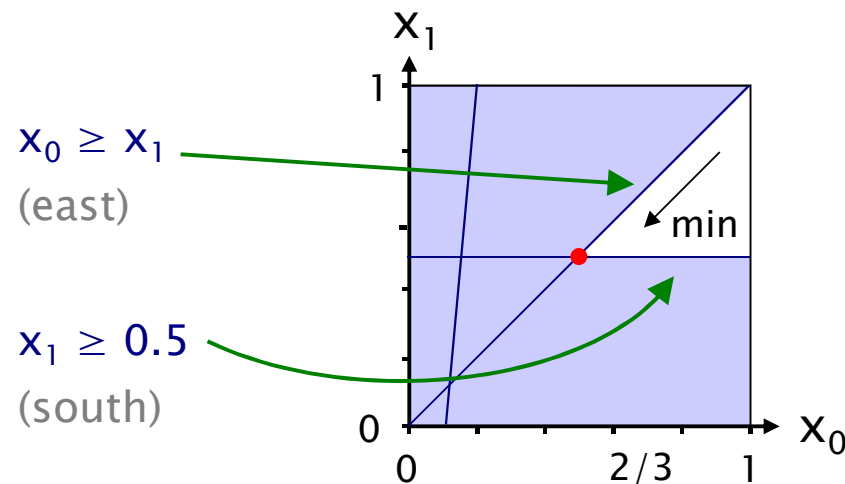
- Compute optimal probabilities $\Pr_s^{\max}(F \ b)$ for all $s \in S$
- To compute the optimal strategy σ^* , choose the **locally optimal** action in each state
 - in general depends on the method used to compute the optimal probabilities
- For reachability
 - **memoryless** strategies suffice
- For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

Example – Strategy

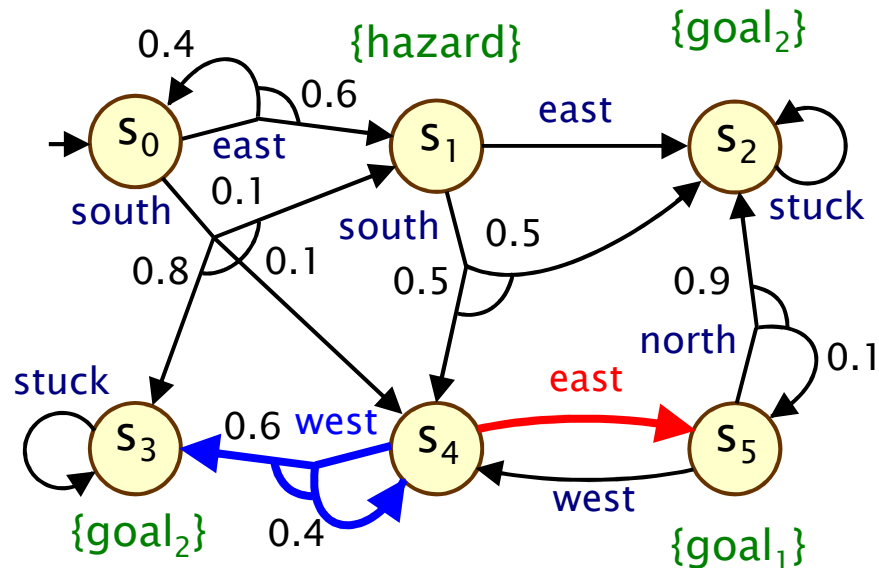


Optimal strategy:

- S_0 : east
- S_1 : south
- S_2 : -
- S_3 : -
- S_4 : east
- S_5 : -



Example – Bounded reachability



Example:

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

So compute:

$$Pr_s^{\max}(F^{\leq 3} \text{goal}_2) = 0.99$$

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

- $P_{\geq 0.05} [(G \neg \text{hazard}) \wedge (GF \text{goal}_1)]$
 - avoid **hazard** and visit **goal₁** infinitely often
- $\Pr_{s_0}^{\max}((G \neg \text{hazard}) \wedge (GF \text{goal}_1)) = 0.1$

Optimal strategy:
(in this instance,
memoryless)

s_0 : **south**

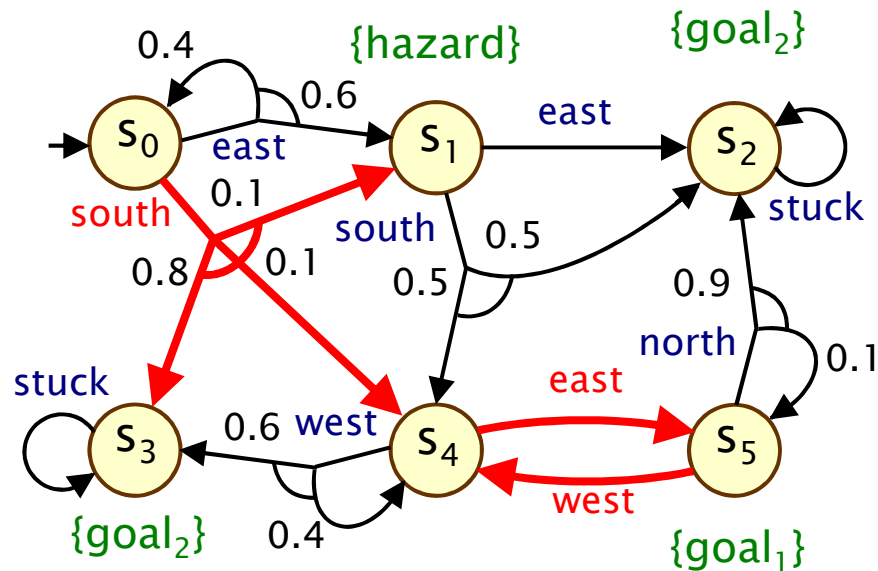
s_1 : –

s_2 : –

s_3 : –

s_4 : **east**

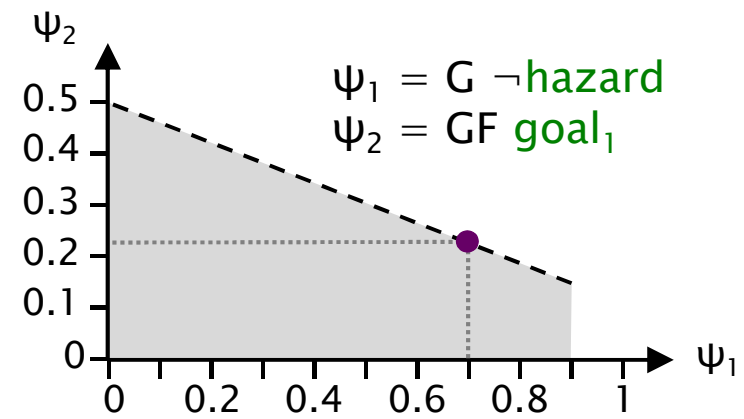
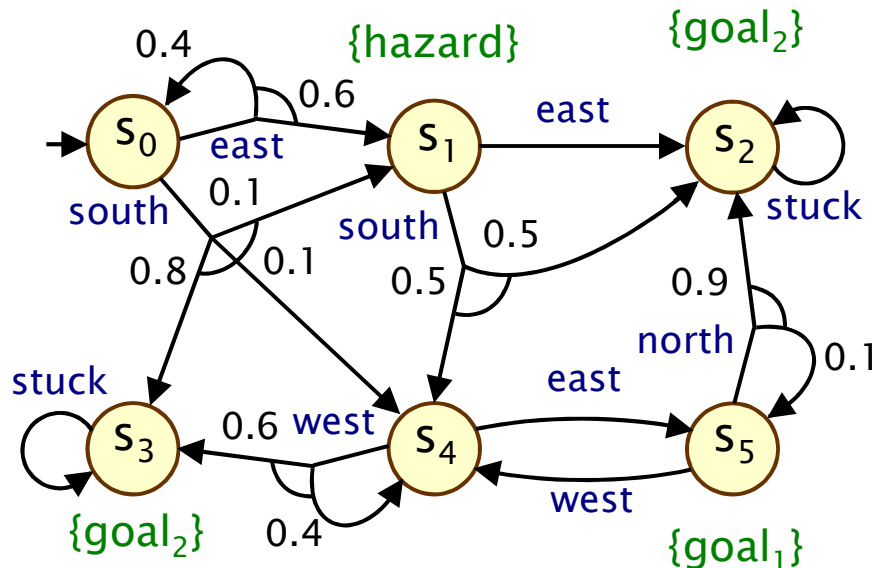
s_5 : **west**



Multi-objective strategy synthesis

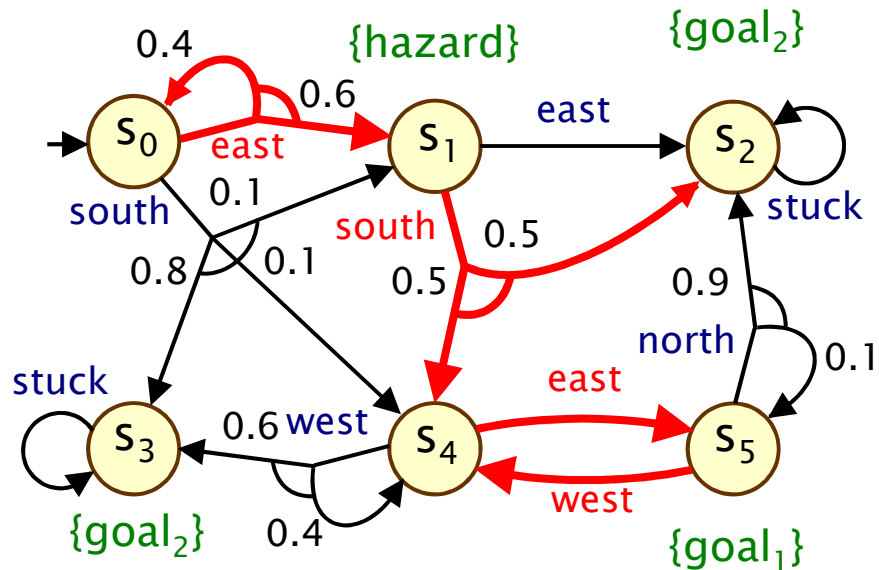
- Consider **conjunctions** of probabilistic LTL formulas $P_{\sim p} [\psi]$
 - require all conjuncts to be satisfied
- Reduce to a **multi-objective reachability** problem on the product of MDP M and the omega-automata representing the conjuncts
 - convert (by negation) to formulas with lower probability bounds (\leq , $<$), 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
 - $P_{\geq 0.7} [G \neg \text{hazard}] \wedge P_{\geq 0.2} [GF \text{goal}_1]$? **True (achievable)**
- Numerical query
 - $P_{\max=?} [GF \text{goal}_1]$ such that $P_{\geq 0.7} [G \neg \text{hazard}]$? **~0.2278**
- Pareto query
 - for $P_{\max=?} [G \neg \text{hazard}] \wedge P_{\max=?} [GF \text{goal}_1]$? ↪

Example – Multi-objective strategies



Strategy 1
(deterministic)

S_0 : east

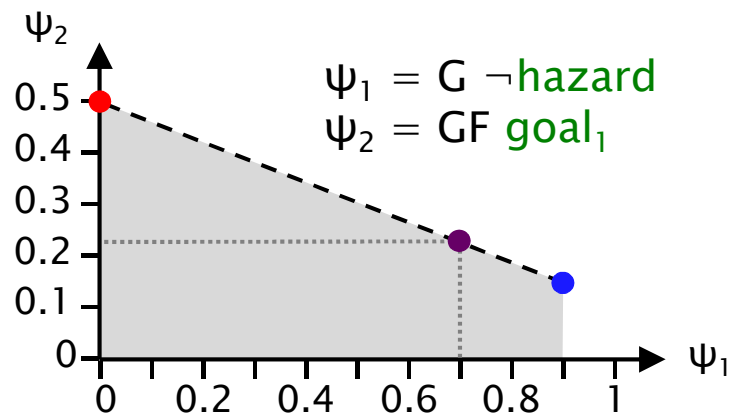
S_1 : south

S_2 : -

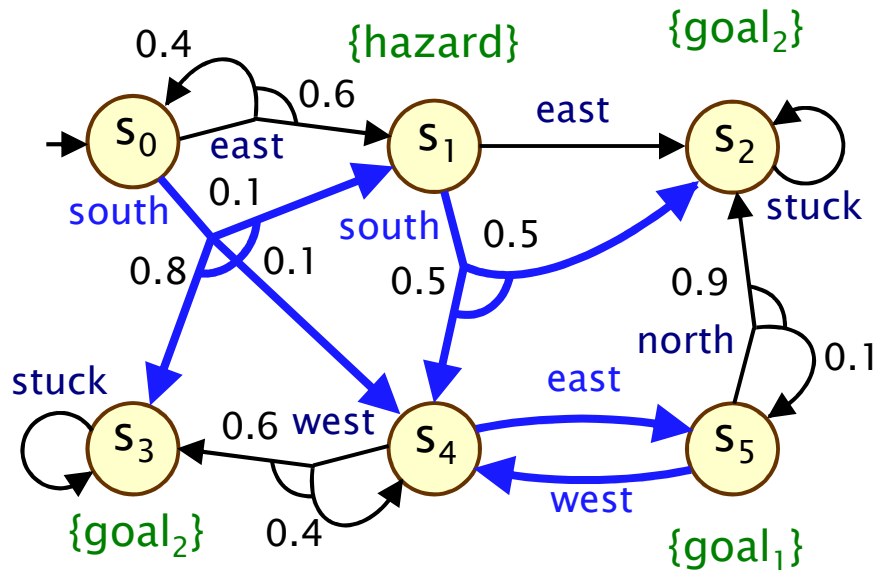
S_3 : -

S_4 : east

S_5 : west



Example – Multi-objective strategies



Strategy 2
(deterministic)

s_0 : south

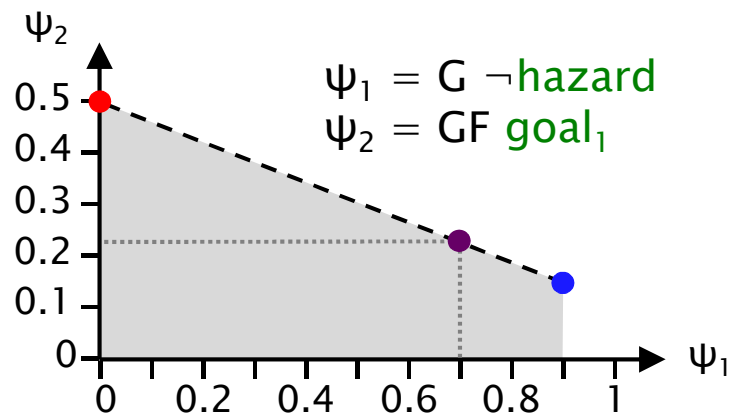
s_1 : south

s_2 : -

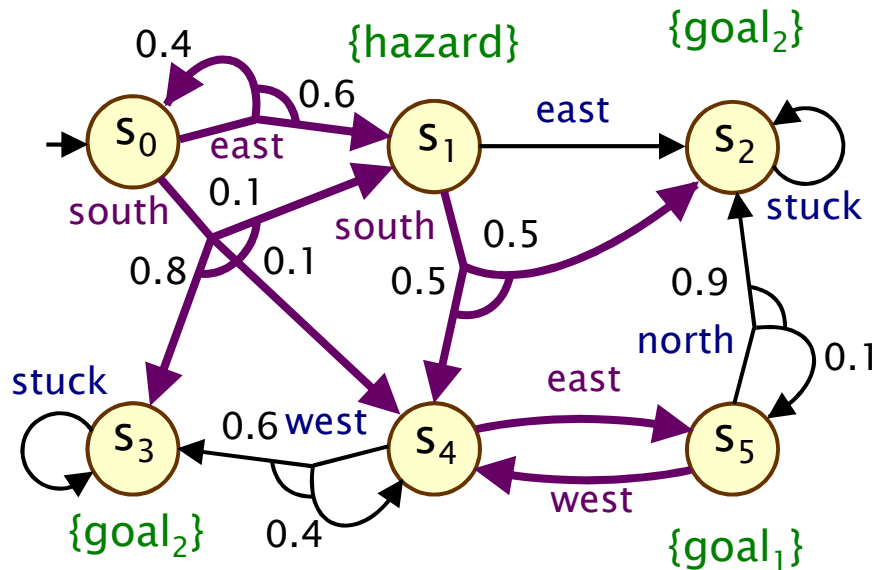
s_3 : -

s_4 : east

s_5 : west



Example – Multi-objective strategies



Optimal strategy:

(randomised)

s_0 : 0.3226 : east

0.6774 : south

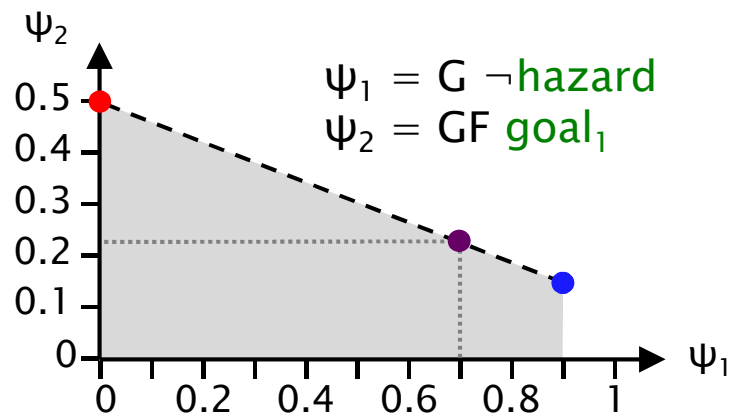
s_1 : 1.0 : south

s_2 : -

s_3 : -

s_4 : 1.0 : east

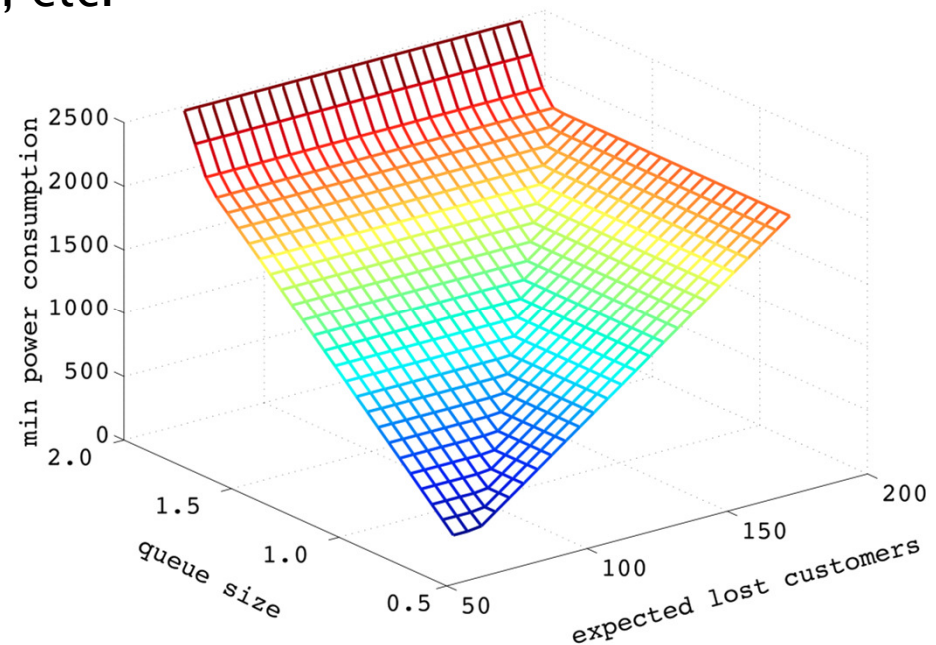
s_5 : 1.0 : west



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: <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

- Major new features:
 1. **multi-objective** model checking
 2. **parametric** model checking
 3. **real-time**: probabilistic timed automata (PTAs)
 4. **games**: stochastic multi-player games (SMGs)
- Further new additions:
 - strategy (adversary) synthesis
 - CTL model checking & counterexample generation
 - enhanced statistical model checking (approximations + confidence intervals, acceptance sampling)
 - efficient CTMC model checking (fast adaptive uniformisation) [Mateescu et al., CMSB'13]
 - benchmark suite & testing functionality [QEST'12]
www.prismmodelchecker.org/benchmarks/

Acknowledgements

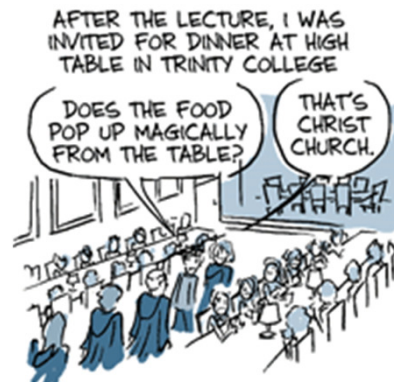
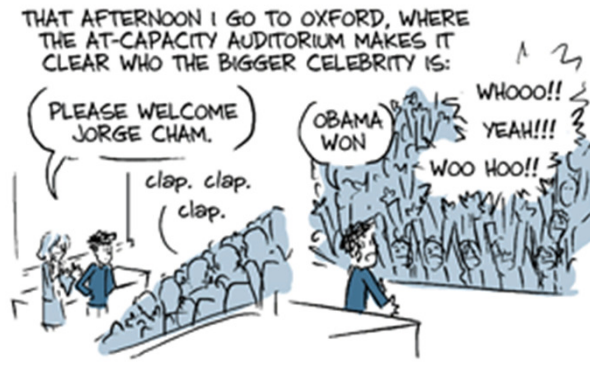
- My group and collaborators in this work
- Project funding
 - ERC, EPSRC, Microsoft Research
 - Oxford Martin School, Institute for the Future of Computing
- See also
 - **VERIWARE** www.veriware.org
 - PRISM www.prismmodelchecker.org

PhD Comics and Oxford...



PHD TALES FROM THE ROAD PRESENTS: "2nd Desserts"

JORGE CHAM © 2008
WWW.PHDCOMICS.COM



- You are welcome to visit Oxford!
- PhD scholarships, postdocs in verification and synthesis, and more