Automated Verification and Strategy Synthesis for Probabilistic Systems

Marta Kwiatkowska

Department of Computer Science, University of Oxford

Joint work with: Dave Parker

ATVA 2013, Hanoi, Vietnam, October 2013

Why automated verification?

- Errors in computerised systems can be costly…

Pentium chip (1994)
Bug found in FPU.
Intel (eventually) offers to replace faulty chips.
Estimated loss: $475m

Infusion pumps (2010)
Patients die because of incorrect dosage.
Cause: software malfunction.
79 recalls.

Toyota Prius (2010)
Software “glitch” found in anti-lock braking system.
185,000 cars recalled.

- Why verify?
  - “Testing can only show the presence of errors, not their absence.” [Edsger Dijstra]

Probabilistic verification

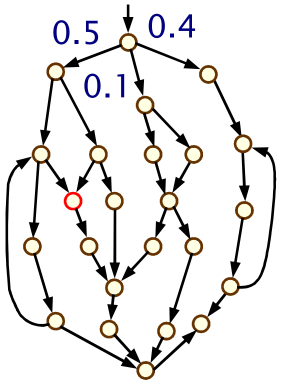
- Probabilistic verification
  - formal verification of systems exhibiting stochastic behaviour
- Why probability?
  - unreliability (e.g. component failures)
  - uncertainty (e.g. message losses/delays over wireless)
  - randomisation (e.g. in protocols such as Bluetooth, ZigBee)
- Quantitative properties
  - reliability, performance, quality of service, …
  - “the probability of an airbag failing to deploy within 0.02s”
  - “the expected time for a network protocol to send a packet”
  - “the expected power usage of a sensor network over 1 hour”

Quantitative (probabilistic) verification

Automatic verification (aka model checking) of quantitative properties of probabilistic system models

System

Probabilistic model
e.g. Markov chain



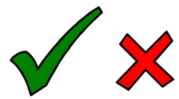
System require-ments

$P_{<0.01}$ [$F^{\leq t}$ fail]

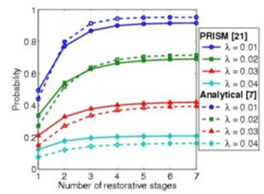
Probabilistic temporal logic specification
e.g. PCTL, CSL, LTL

Probabilistic model checker
e.g. PRISM

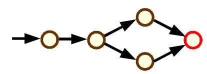
Result



Quantitative results



Counter-example



Historical perspective

- First algorithms proposed in 1980s
  - algorithms [Vardi, Courcoubetis, Yannakakis, …]
  - [Hansson, Jonsson, de Alfaro] & first implementations
- 2000: tools ETMCC (now MRMC) & PRISM released
  - PRISM: efficient extensions of symbolic model checking [Kwiatkowska, Norman, Parker, …]
  - ETMCC: model checking for continuous-time Markov chains [Baier, Hermanns, Haverkort, Katoen, …]
- Now mature area, of industrial relevance
  - successfully used by non-experts for many application domains, but full automation and good tool support essential
    - distributed algorithms, communication protocols, security protocols, biological systems, quantum cryptography, planning, …
  - genuine flaws found and corrected in real-world systems

Quantitative probabilistic verification

- What's involved
  - specifying, extracting and building of quantitative models
  - graph-based analysis: reachability + qualitative verification
  - numerical solution, e.g. linear equations/linear programming
  - typically computationally more expensive than the non-quantitative case
- The state of the art
  - fast/efficient techniques for a range of probabilistic models
  - feasible for models of up to 10^7 states (10^{10} with symbolic)
  - extension to probabilistic real-time systems
  - abstraction refinement (CEGAR) methods
  - probabilistic counterexample generation
  - assume-guarantee compositional verification
  - tool support exists and is widely used, e.g. PRISM, MRMC

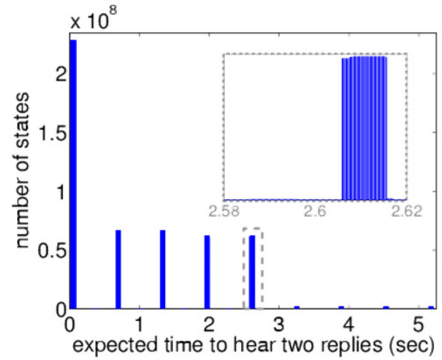
Tool support: PRISM

- PRISM: Probabilistic symbolic model checker [CAV11]
  - developed at Birmingham/Oxford University, since 1999
  - free, open source software (GPL), runs on all major OSs
- Support for:
  - models: DTMCs, CTMCs, MDPs, PTAs, SMGs, …
  - properties: PCTL, CSL, LTL, PCTL*, costs/rewards, rPATL, …
- Features:
  - simple but flexible high-level modelling language
  - user interface: editors, simulator, experiments, graph plotting
  - multiple efficient model checking engines (e.g. symbolic)
  - New! strategy synthesis, stochastic game models (SMGs) , multiobjective verification, parametric models

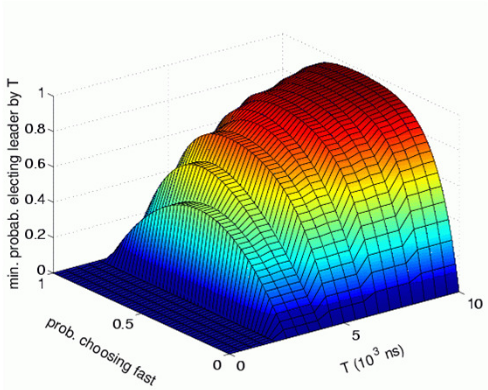
- See: http://www.prismmodelchecker.org/

Quantitative verification in action

- Bluetooth device discovery protocol
  - frequency hopping, randomised delays
  - low-level model in PRISM, based on detailed Bluetooth reference documentation
  - numerical solution of 32 Markov chains, each approximately 3 billion states
  - identified worst-case time to hear one message, 2.5 seconds

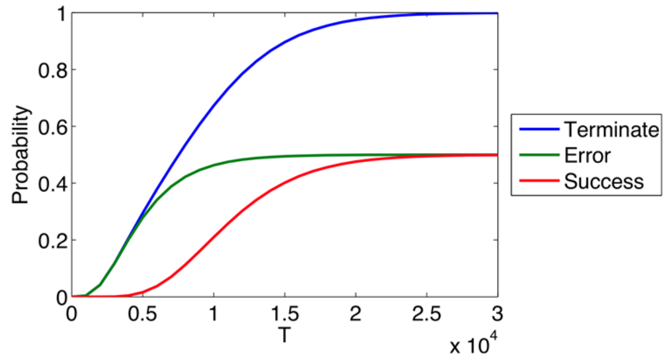


- FireWire root contention
  - wired protocol, uses randomisation
  - model checking using PRISM
  - optimum probability of leader election by time T for various coin biases
  - demonstrated that a biased coin can improve performance

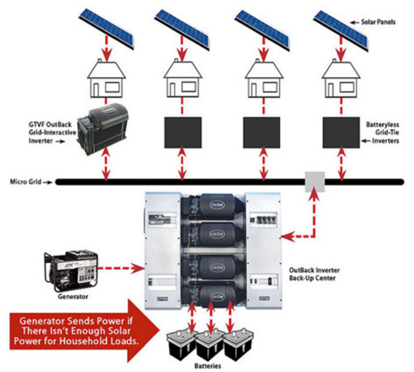


Quantitative verification in action

- DNA transducer gate [Lakin et al, 2012]
  - DNA computing with a restricted class of DNA strand displacement structures
  - transducer design due to Cardelli
  - automatically found and fixed design error, using Microsoft's DSD and PRISM

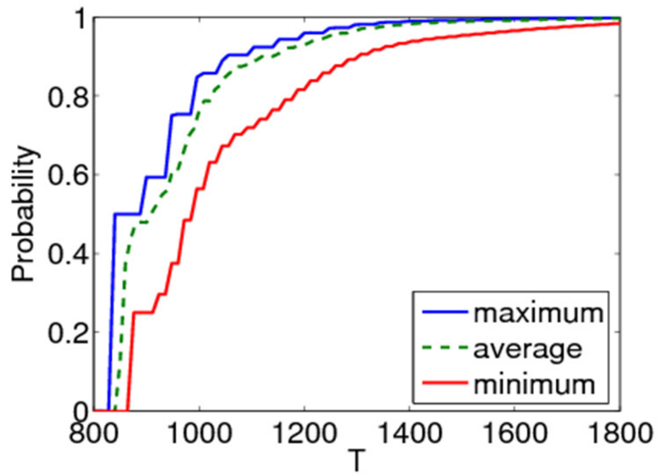


- Microgrid demand management protocol [TACAS12,FMSD13]
  - designed for households to actively manage demand while accessing a variety of energy sources
  - found and fixed a flaw in the protocol, due to lack of punishment for selfish behaviour
  - implemented in PRISM-games



Quantitative verification – Status

- Tools/techniques widely applicable, since real software/systems <u>are</u> quantitative
  - extensions/adaptations of model-based frameworks
  - new application domains
- Analysis "quantitative" & "exhaustive"
  - strength of mathematical proof
  - best/worst-case scenarios, not possible with simulation
  - identifying trends and anomalies



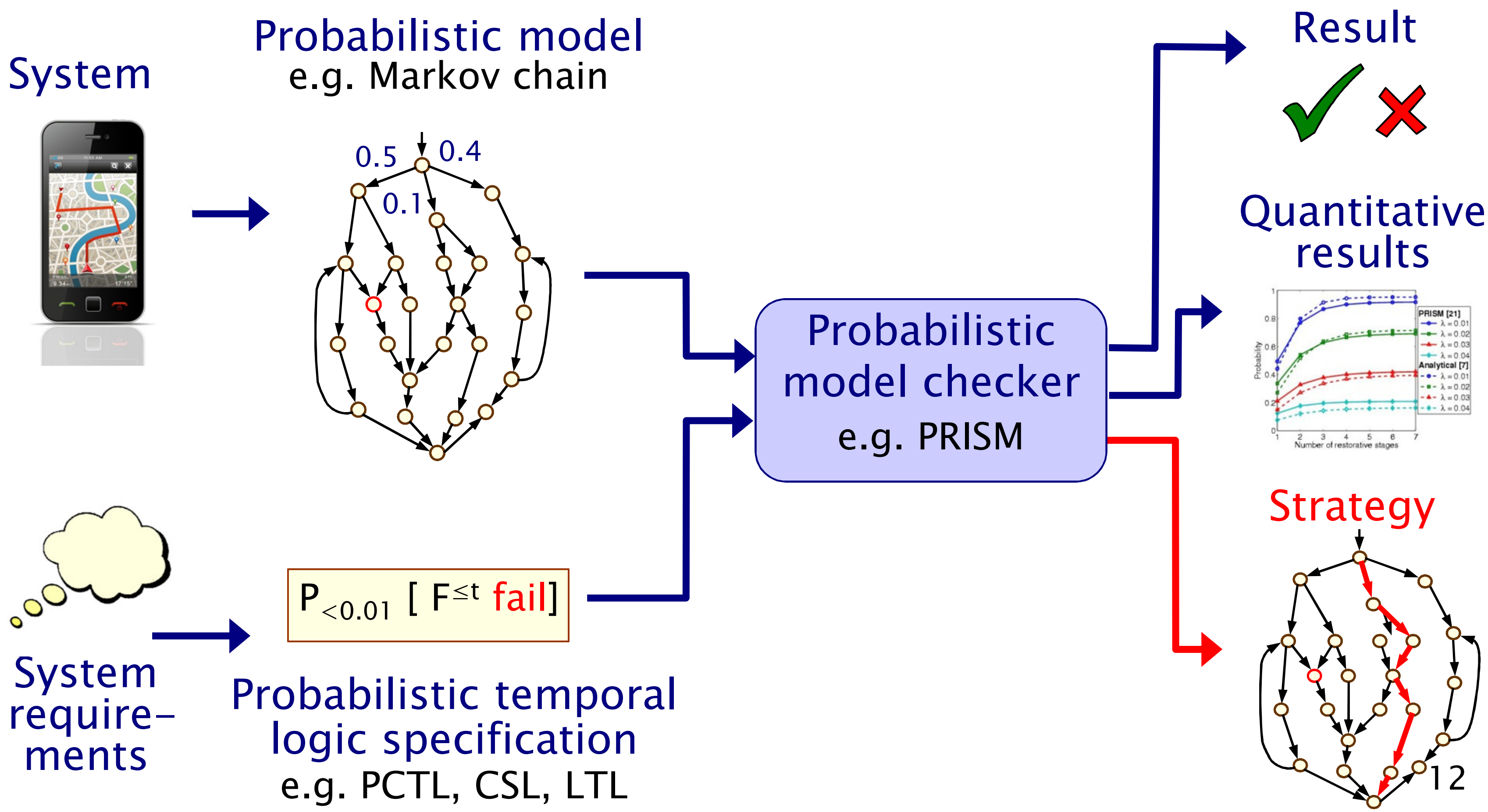
- But
  - the modelling phase time-consuming and error prone
  - potential 'disconnect' between model and the artefact
  - scalability continues to be hard to overcome

This lecture…

- We focus on the problem of strategy 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?"
  - advantage: correct-by-construction
- Not a well known fact…
  - can reuse the verification algorithms for strategy synthesis
- Many application domains
  - robotics (controller synthesis from LTL/PCTL)
  - security (generating attacks)
  - dynamic power management (optimal policy synthesis)
- Move towards quantitative model synthesis
  - simpler problems: strategy synthesis, parameter synthesis, template-based synthesis, etc

Quantitative (probabilistic) verification

Automatic verification and strategy synthesis from quantitative properties for probabilistic models



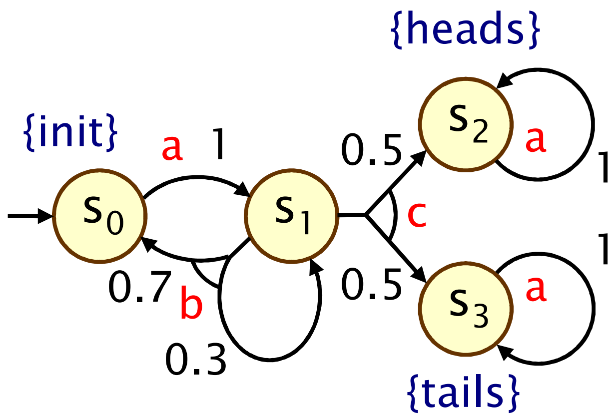
Overview

- Motivation
- Overview of Markov decision processes (MDPs)
  - MDPs: definition, paths & probability spaces
  - Strategies (aka adversaries/policies): definition & classification
- Verification and strategy synthesis
  - Properties and objectives
  - Problem definition
  - Algorithms for MDPs
- Strategy synthesis by example
  - Reachability objectives
  - LTL objectives
  - Multiobjective strategy synthesis
  - Strategy synthesis for stochastic games
- Conclusion

Markov decision processes (MDPs)

- Model nondeterministic as well as probabilistic behaviour
  - e.g. for concurrency, under-specification, abstraction…
  - extension of discrete-time Markov chains
  - nondeterministic choice between probability distributions

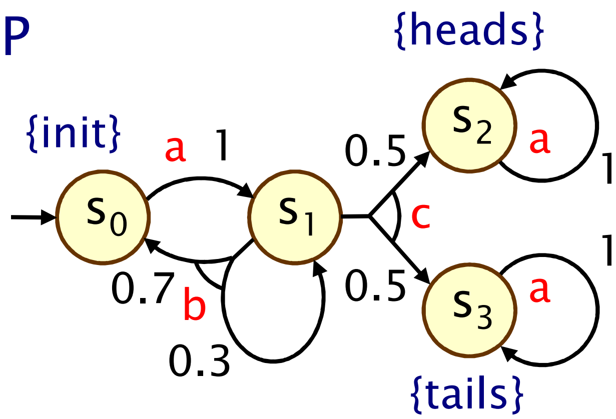
- Formally, an MDP is a tuple
  - $(S, s_{init}, Act, \delta, L)$



- where:
  - S is a set of states
  - $s_{init} \in S$ is the initial state
  - $\delta : S \times Act \rightarrow Dist(S)$ is a (partial) transition probability function
  - $L : S \rightarrow 2^{AP}$ is a labelling function
  - Act is a set of actions, AP is a set of atomic propositions
  - $Dist(S)$ is the set of discrete probability distributions over S

Paths and strategies

- A (finite or infinite) path through an MDP
  - is a sequence ($s_0 ... s_n$) 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

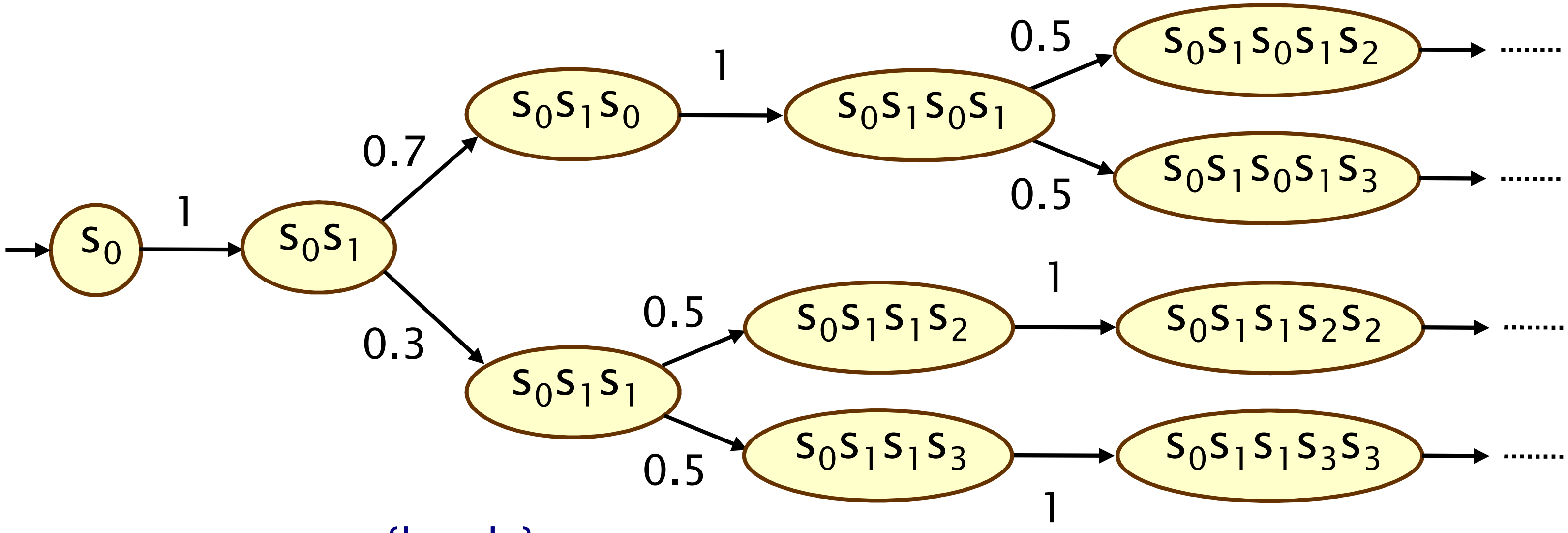
Classification of strategies

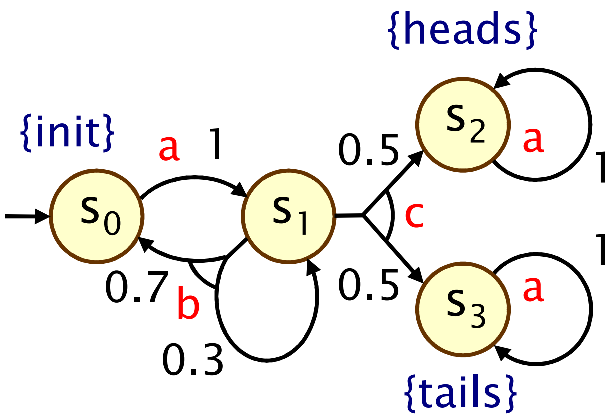
- Strategies are classified according to
- randomisation:
  - σ is deterministic (pure) if $\sigma(s_0 \ldots s_n)$ is a point distribution, and randomised otherwise
- memory:
  - σ is memoryless (simple) if $\sigma(s_0 \ldots s_n) = \sigma(s_n)$ for all $s_0 \ldots s_n$
  - σ is finite memory if there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

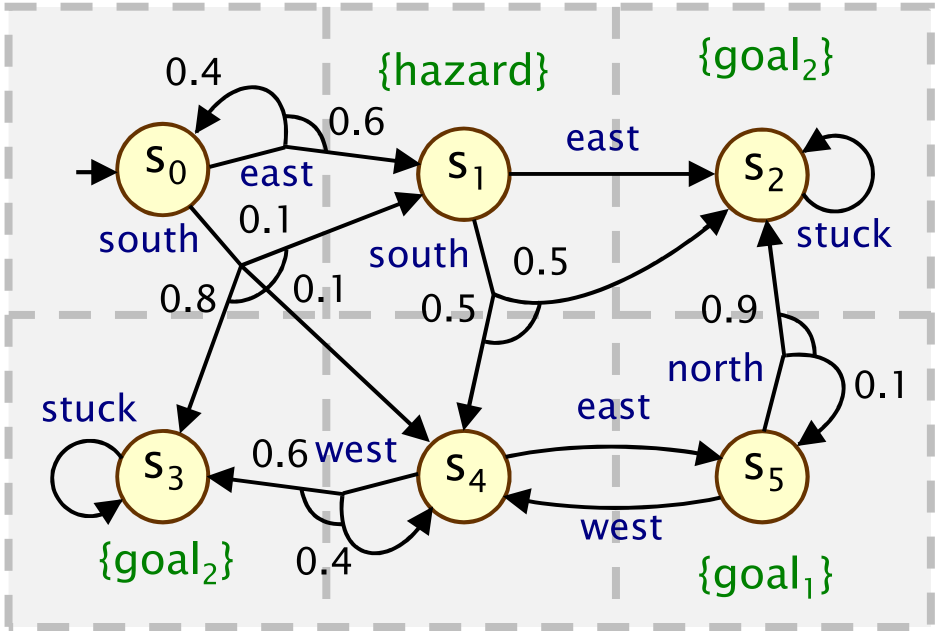




finite-memory,
deterministic

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

  - ϕ ::= $P_{\sim p}$ [ψ] | $R_{\sim r}$ [ρ]
  - ψ ::= true | a | $\psi \wedge \psi$ | $\neg\, \psi$ | X ψ | $\psi\ U^{\leq k}\ \psi$ | $\psi\ U\ \psi$

    "next" "bounded until" "until"

  - ρ ::= F b | C | $C^{\leq k}$

    "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 ≡ 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"

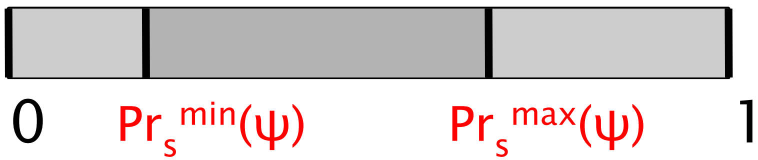
- Some examples:
  - $P_{\geq 0.4}$ [F "goal"] "probability of reaching goal is at least 0.4"
  - $R_{<5}$ [$C^{\leq 60}$] "expected power consumption over one hour is below 5"
  - $R_{\leq 10}$ [F "end"] "expected time to termination is at most 10"

Verification and strategy synthesis

- The verification problem is:
  - Given an MDP M and a property ϕ, does M satisfy ϕ for all possible strategies 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, i.e. for $\phi = P_{\sim p}[\,\psi\,]$:
  - $Pr_s^{min}(\psi) = \inf_\sigma Pr_s^\sigma(\psi)$
  - $Pr_s^{max}(\psi) = \sup_\sigma Pr_s^\sigma(\psi)$



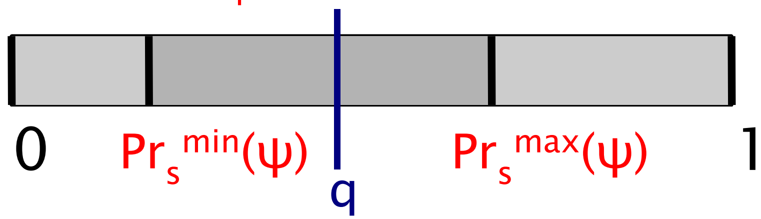
- Expectations (reward objectives $R_{\sim r}[\psi]$) are similar, omitted

Verification and strategy synthesis

- The verification problem is:
  - Given an MDP M and a property ϕ, does M satisfy ϕ for all possible strategies 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

- In particular, we have
  - M satisfies $\phi = P_{\geq q}[\ \psi\ ]$ iff $Pr_s^{min}(\psi) \geq q$
  - There exists a strategy satisfying $\phi = P_{\geq q}[\ \psi\ ]$ iff $Pr_s^{max}(\psi) \geq q$
  - then take optimal strategy



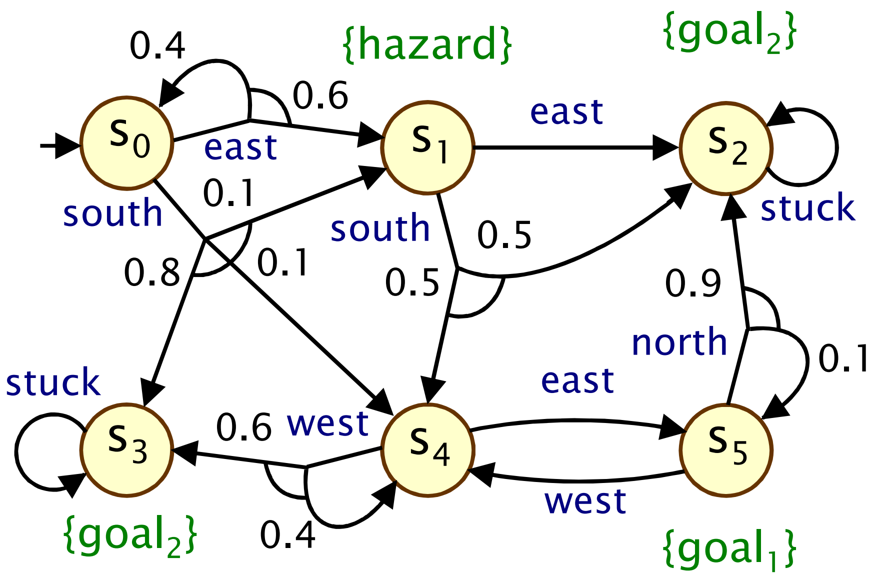
Computing reachability for MDPs

- 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 (for Pr_s^{max}):
  - 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



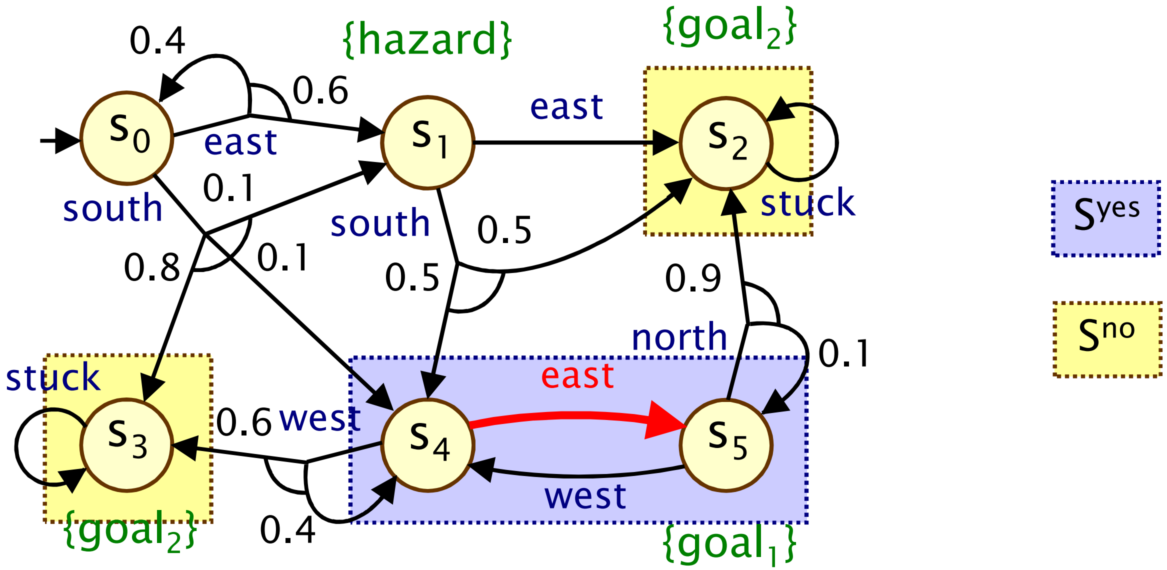
Example goal:

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

So compute:

Pr_s^{max}(F $goal_1$)

Example – Precomputation



Example goal:

$P_{\geq 0.4}$ [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:

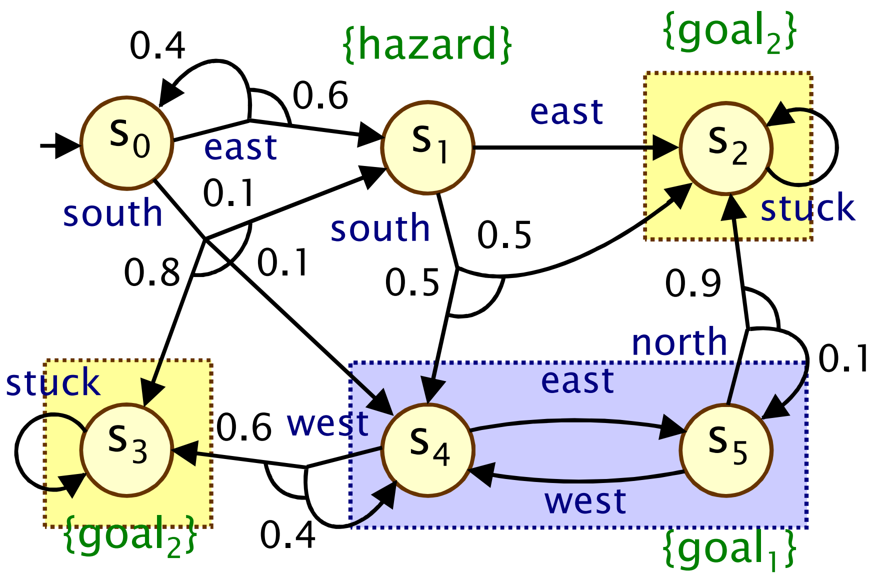
$$\text{minimize } \sum_{s \in S^?} x_s \text{ subject to the constraints:}$$

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

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

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

Example – Reachability (LP)



Example:

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

So compute:

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

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

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

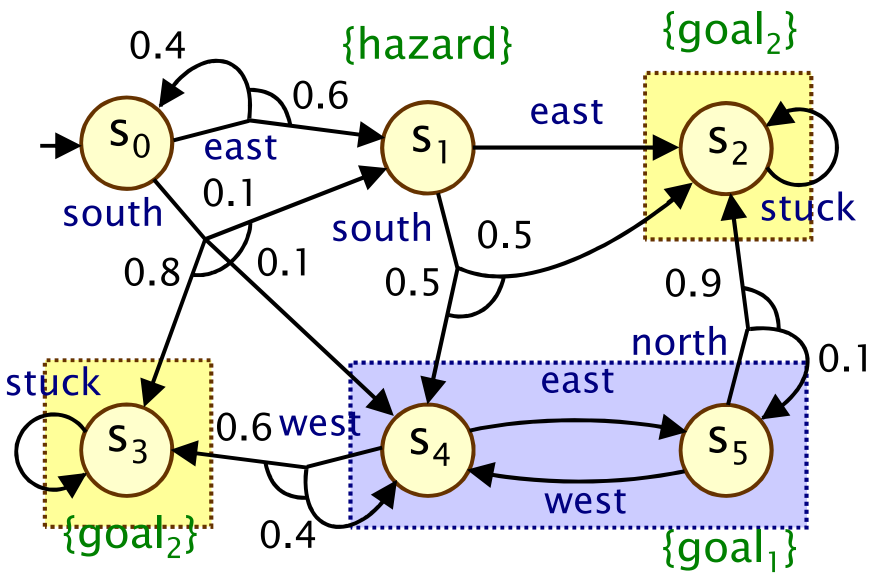
S^{no}: $x_2 = x_3 = 0$

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

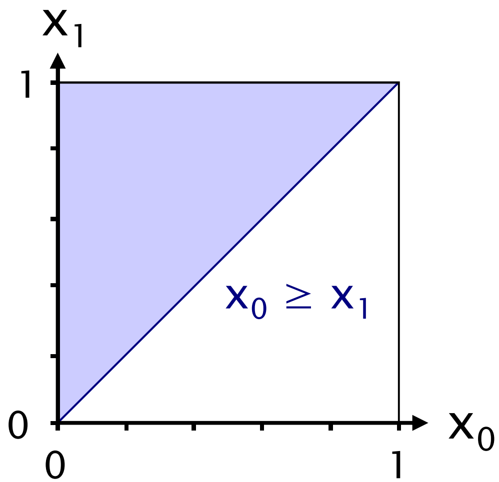
S^{yes}: $x_4 = x_5 = 1$

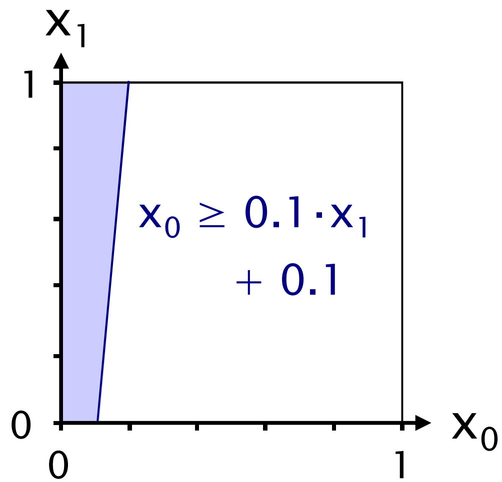
S^{no}: $x_2 = x_3 = 0$

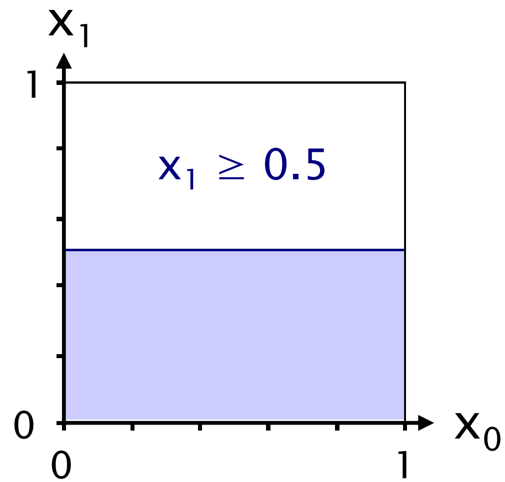
For $S^? = \{x_0, x_1\}$:

Minimise $x_0 + x_1$ subject to:

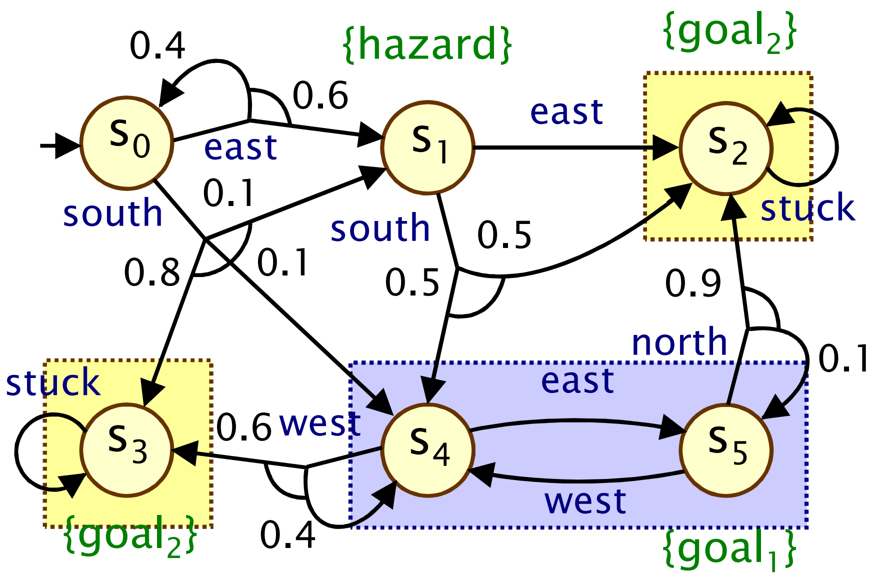
- $x_0 \geq x_1$ (east)
- $x_0 \geq 0.1 \cdot x_1 + 0.1$ (south)
- $x_1 \geq 0.5$ (south)







Example – Reachability (LP)



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

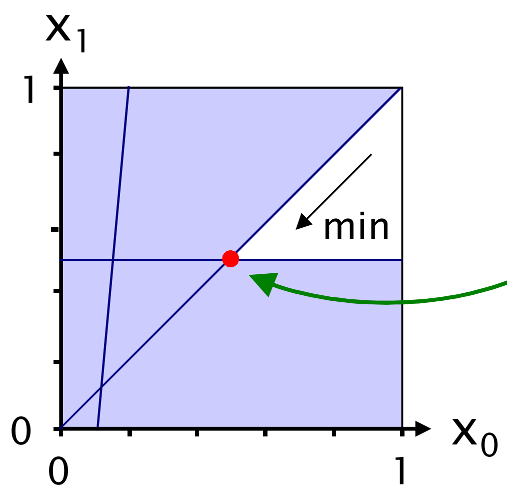
S^{yes}: $x_4 = x_5 = 1$

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.

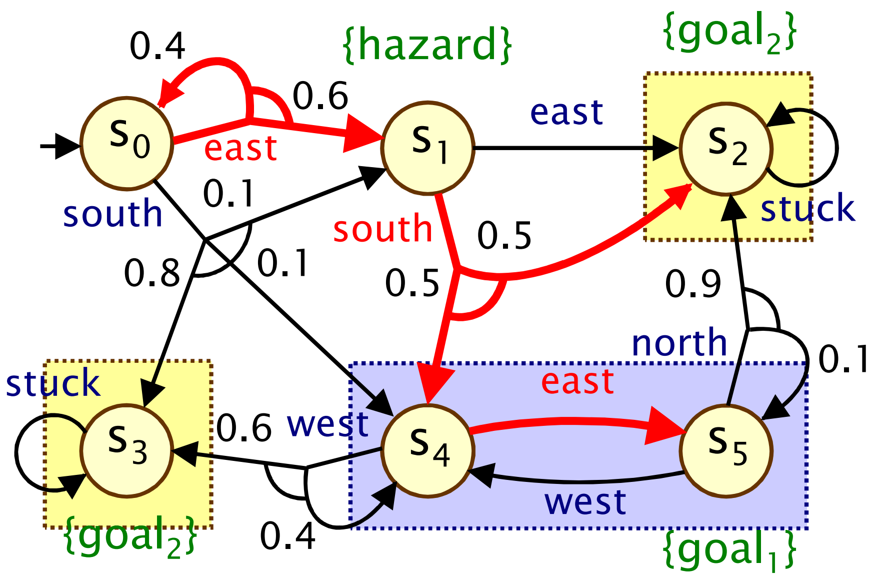
$Pr_{s_0}^{max}(F\ goal_1) = 0.5$

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
  - i.e. policy iteration constructs the optimal strategy
  - for max probabilities, adaptation of precomputation needed
- 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

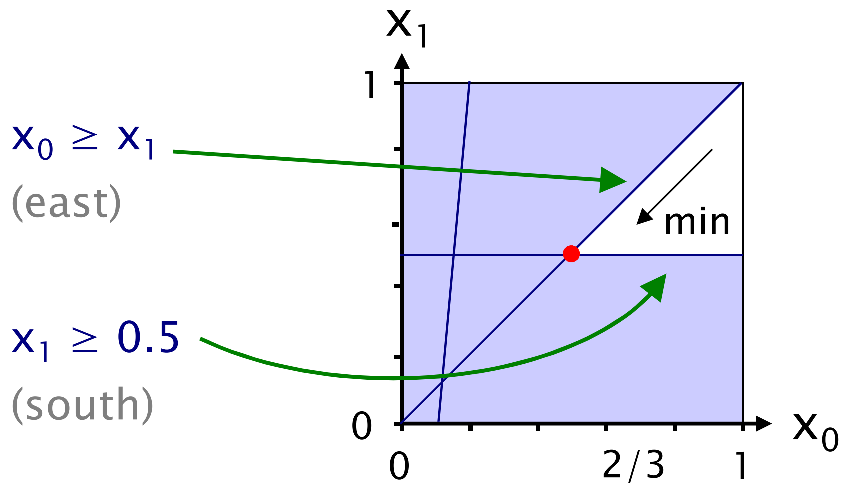
s_1 : south

s_2 : –

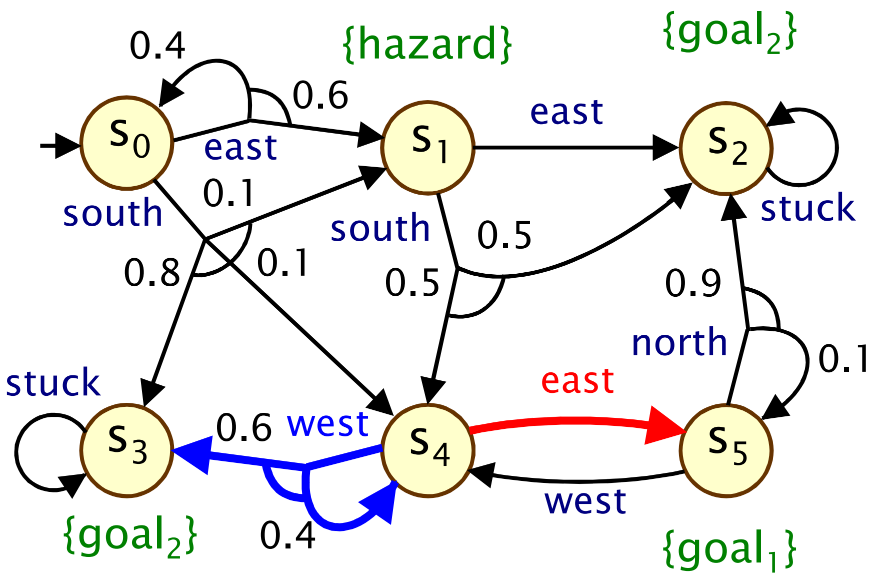
s_3 : –

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

Computation more involved

May need to choose a different action on successive visits

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)

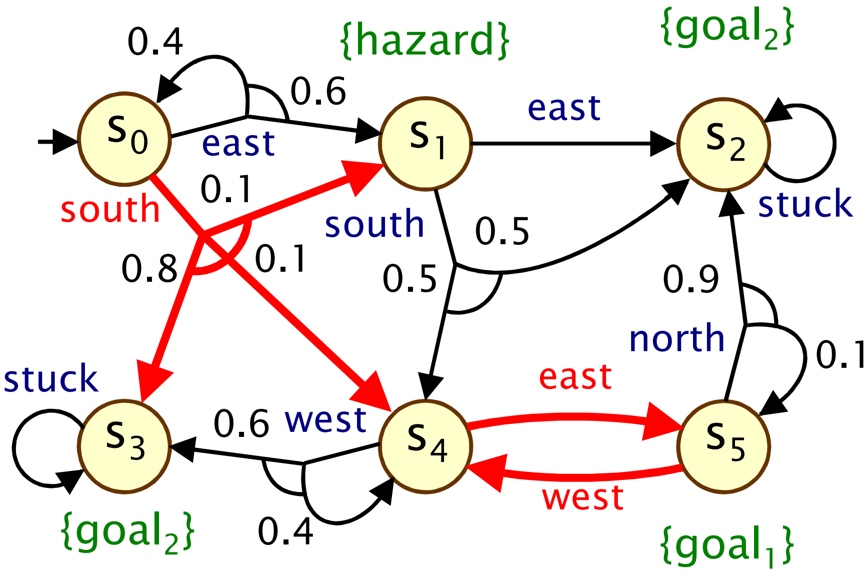
- Need only consider computation of maximum probabilities $Pr_s^{max}(\psi)$
  - since $Pr_s^{min}(\psi) = 1 - Pr_s^{max}(\neg\psi)$

- 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

- $Pr_{s_0}^{max}$((G ¬hazard) ∧ (GF goal$_1$)) = 0.1



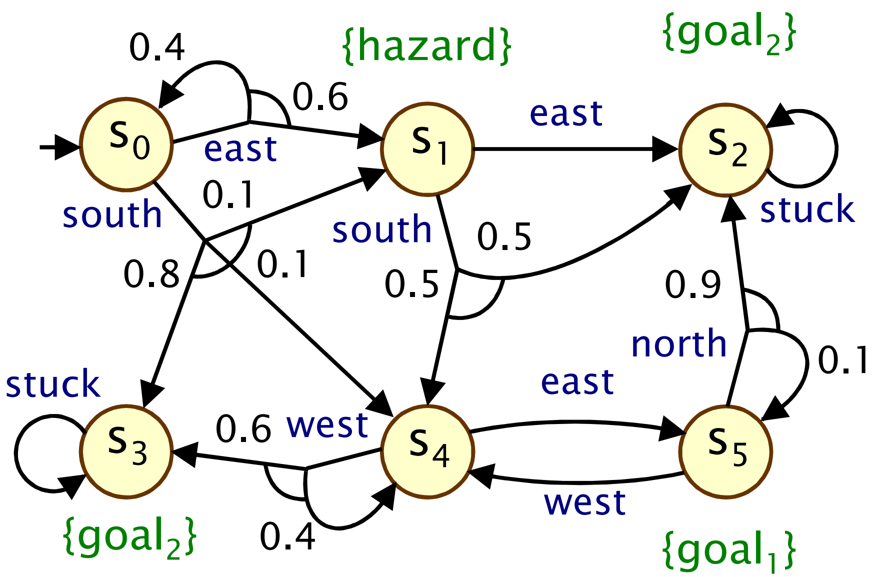
Optimal strategy:
(in this instance, memoryless)

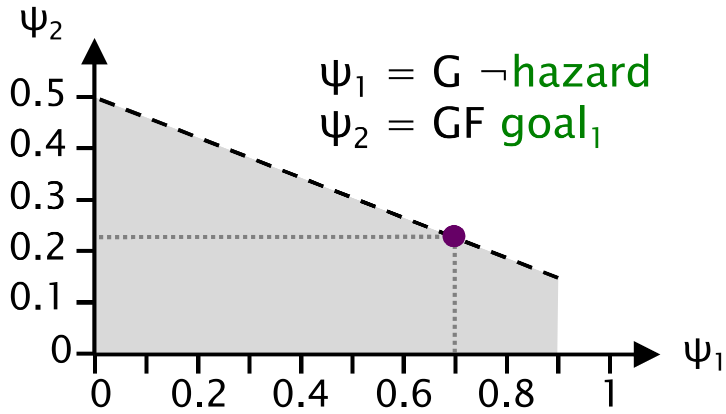
s_0 : south
s_1 : –
s_2 : –
s_3 : –
s_4 : east
s_5 : west

Multi-objective strategy synthesis

- Consider conjunctions of probabilistic LTL formulas $P_{\sim p}\ [\psi]$
  - 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 ($\geq$, $>$), 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

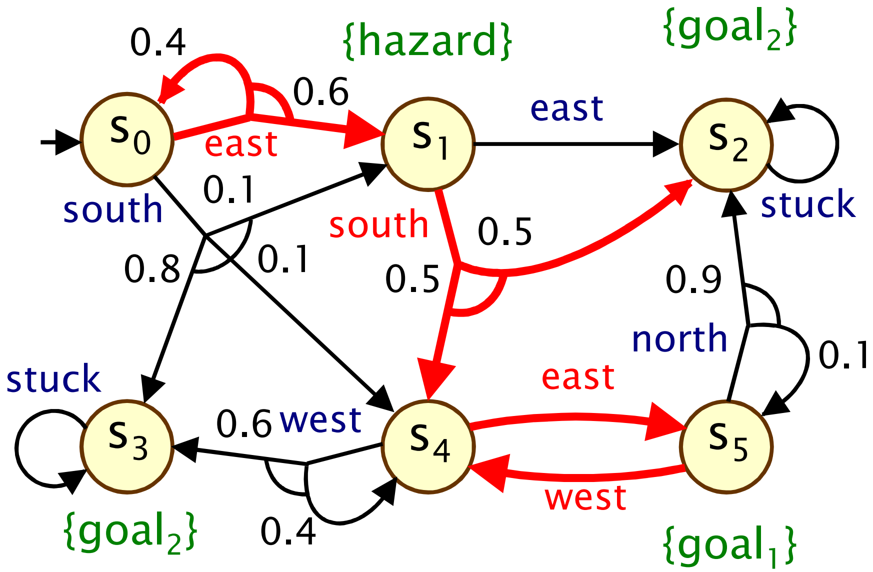
Example – Multi-objective





- 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

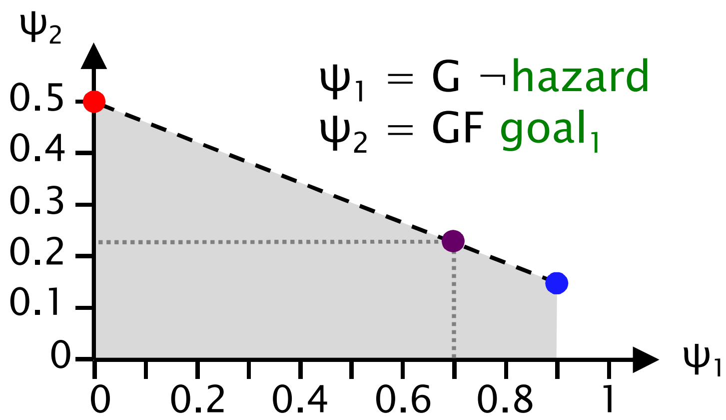
s_1 : south

s_2 : –

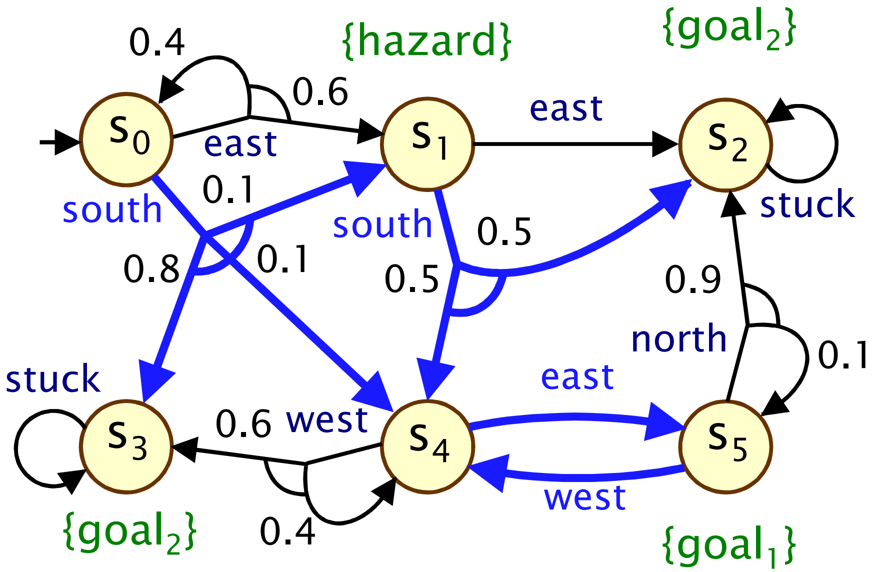
s_3 : –

s_4 : east

s_5 : west



Example – Multi-objective strategies



Strategy 2
(deterministic)

s_0 : south

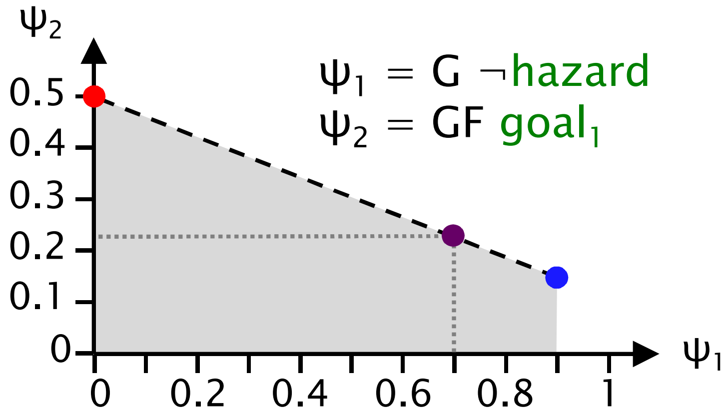
s_1 : south

s_2 : –

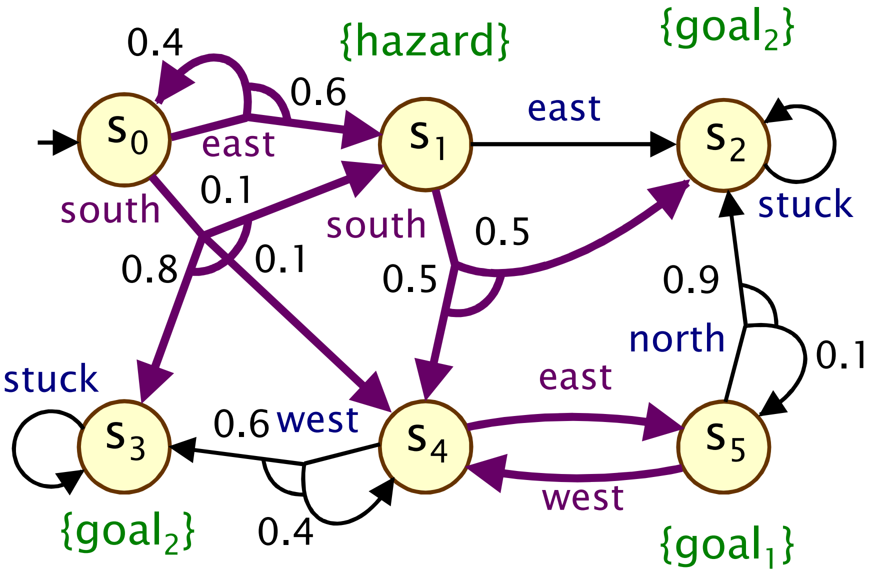
s_3 : –

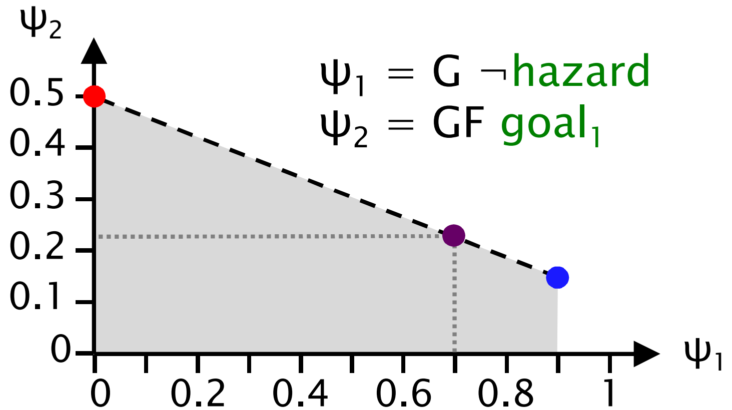
s_4 : east

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 : –

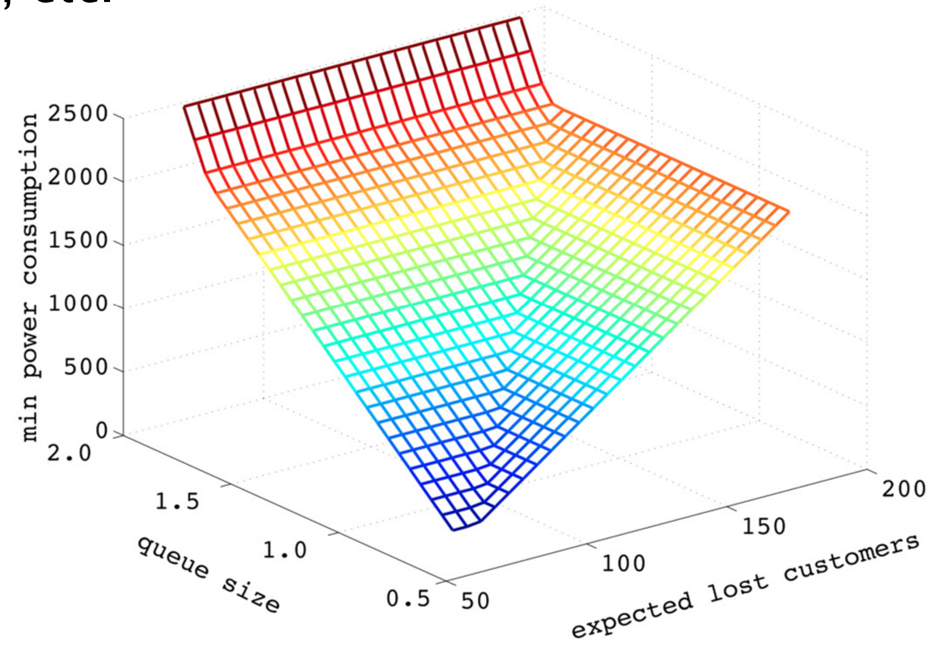
s_4 : 1.0 : east

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.

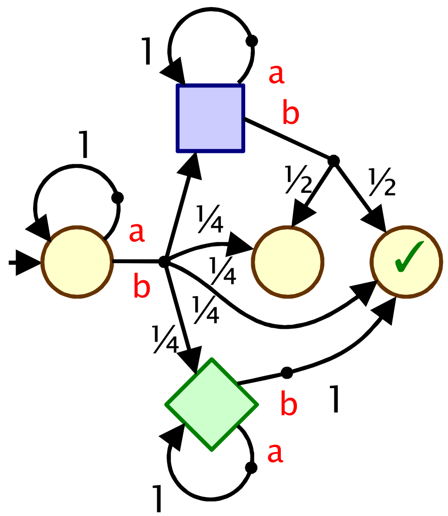
- Build controllers that
  - minimise energy consumption, subject to constraints on e.g.
  - probability that a request waits more than K steps
  - expected number of lost disk requests



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

Stochastic multi-player games (SMGs)

- Stochastic multi-player games
  - players control states; choose actions
  - models competitive/collaborative behaviour
- Property specifications
  - rPATL: extends Alternating Temporal Logic (and PCTL with the R operator)
  - $\langle\langle\{\bigcirc, \square\}\rangle\rangle\, P_{>1/3}\, [\, F\ ✓\,]$
  - “does the coalition have a strategy to ensure that the probability of reaching end state is greater than 1/3, regardless of the strategies of other players?”
- Applications
  - controller synthesis (controller vs. environment), security (system vs. attacker), distributed algorithms, …
- PRISM-games: www.prismmodelchecker.org/games



Model checking rPATL

- Basic algorithm: as for any branching-time temporal logic
  - recursive descent of formula parse tree
  - compute Sat(ϕ) = { s∈S | s⊨ϕ } for each subformula ϕ
- Main task: checking P and R operators
  - reduction to solution of stochastic 2-player game G_C
  - e.g. $\langle\langle C\rangle\rangle P_{\geq q}[\psi] \;\Leftrightarrow\; \sup_{\sigma_1 \in \Sigma_1} \inf_{\sigma_2 \in \Sigma_2} Pr_s^{\sigma_1,\sigma_2}(\psi) \geq q$
  - complexity: NP ∩ coNP (for sublogic)
  - compared to, e.g. P for Markov decision processes
  - complexity for full logic: NEXP ∩ coNEXP
- In practice though:
  - evaluation of numerical fixed points ("value iteration")
  - up to a desired level of convergence
  - usual approach taken in probabilistic model checking tools

Probabilities for P operator

- E.g. $\langle\langle C \rangle\rangle P_{\geq q}[\ F\ \phi\ ]$: max/min reachability probabilities
  - compute $\sup_{\sigma_1 \in \Sigma_1} \inf_{\sigma_2 \in \Sigma_2} Pr_s^{\sigma_1,\sigma_2}(F\ \phi)$ for all states s
  - deterministic memoryless strategies suffice
- Value is:
  - 1 if $s \in Sat(\phi)$, and otherwise least fixed point of:

$$f(s) = \begin{cases} \max_{a \in A(s)} \left(\sum_{s' \in S} \Delta(s,a)(s') \cdot f(s') \right) & \text{if } s \in S_1 \\ \min_{a \in A(s)} \left(\sum_{s' \in S} \Delta(s,a)(s') \cdot f(s') \right) & \text{if } s \in S_2 \end{cases}$$

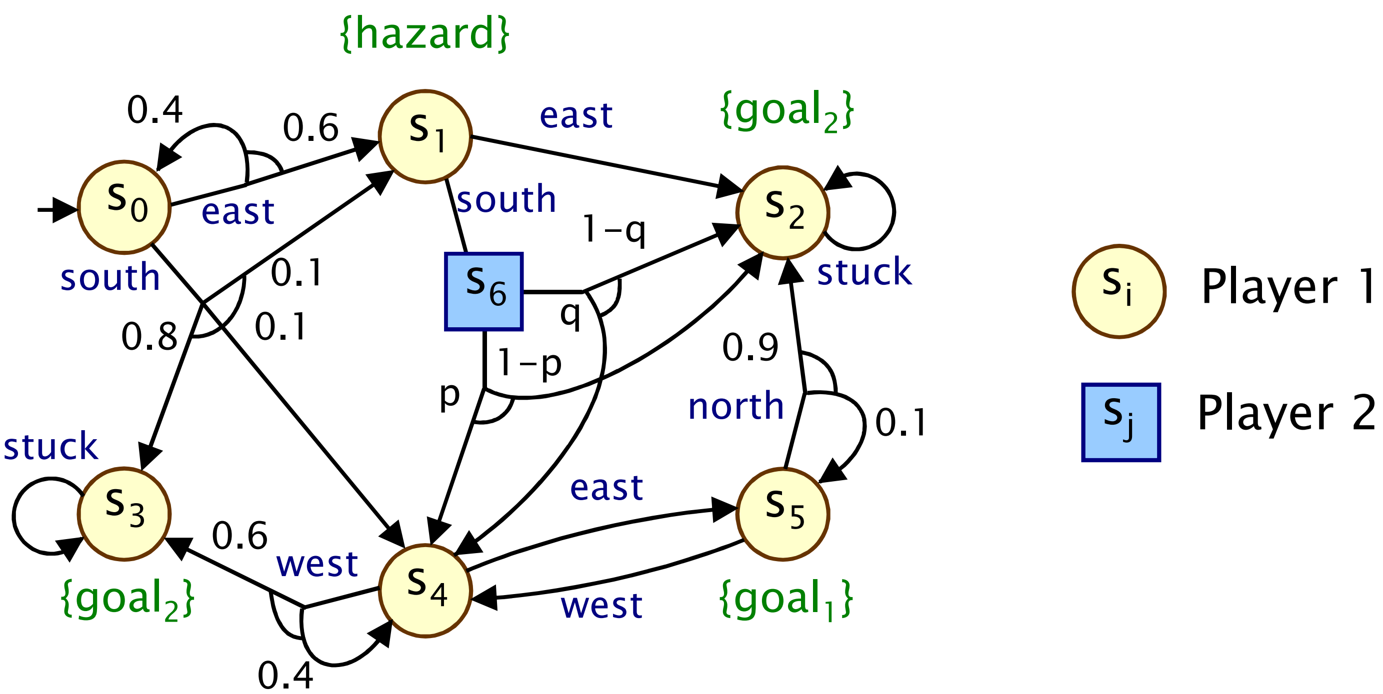
- Computation:
  - start from zero, propagate probabilities backwards
  - guaranteed to converge, similarly to value iteration for MDPs

Strategy synthesis for stochastic games

- Generate strategies for individual players, or for a coalition
- Problem statement:
  – Given a game G and an rPATL property $\langle\langle C\rangle\rangle P_{\sim q}[\ \psi\ ]$, does there exist a strategy σ_1 for players in C such that, for all strategies σ_2 outside C, the probability of satisfying ψ under σ_1 and σ_2 meets the bound $\sim q$
- Compute optimal probabilities
  – for reachability, value or policy iteration, similar to that for MDPs
  – for LTL ψ, again work via product with the Rabin automaton for the formula
- To compute the optimal strategy
  – compute parity objectives for parity automaton from DRA
  – for reachability, memoryless deterministic strategies suffice

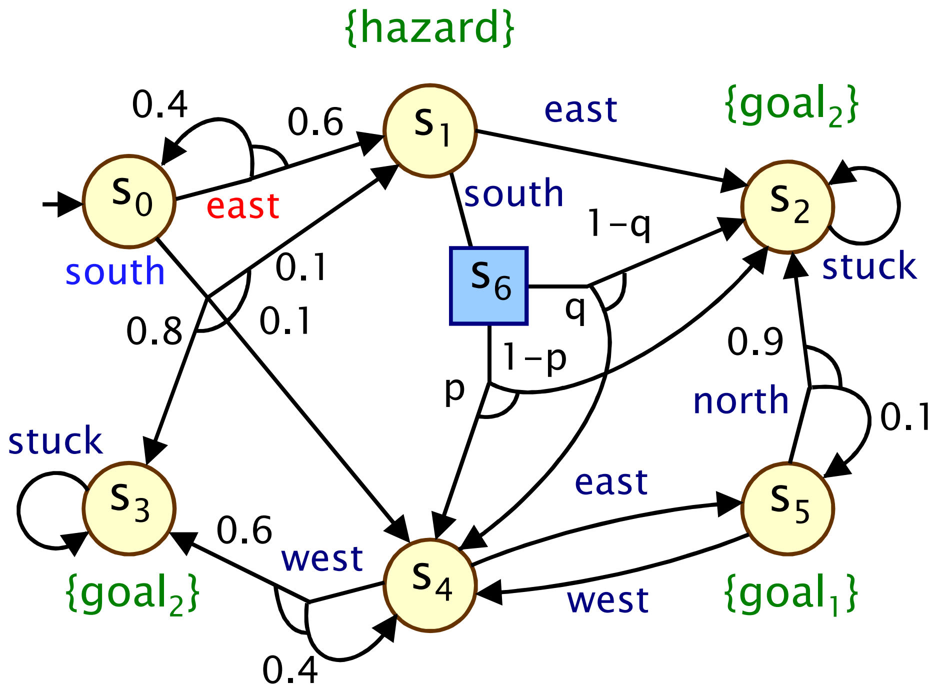
Example – Stochastic games

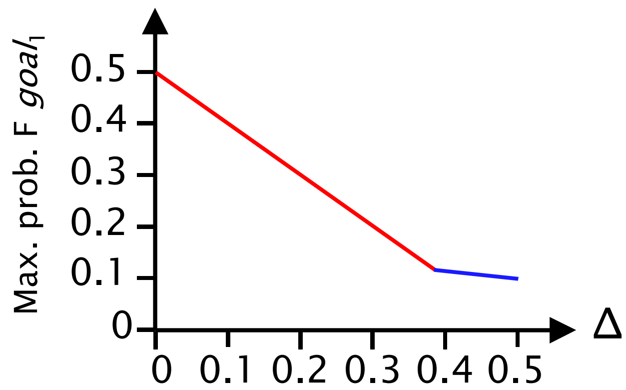
- Two players: 1 (robot controller), 2 (environment)
  - when taking action south in state s_6
  - probability of (correctly) going to s_4 is in interval [p,q]
  - rPATL: $\langle\langle\{1\}\rangle\rangle\, P_{max=?}\,[\, F\, goal_1\,]$



Example – Stochastic games

- rPATL: $\langle\langle\{1\}\rangle\rangle$ Pmax=? [F $goal_1$]
  - let [p,q] = [0.5−Δ, 0.5+Δ]; vary Δ
  - optimal strategy: if Δ ≥ 7/18 (i.e. if p ≤ 1/9), then pick south in s_0, otherwise pick east





Conclusion

- Overview of strategy synthesis
  - for probabilistic LTL and reward objectives
  - multi-objective properties
  - Markov decision process and stochastic games models

- Highlighting new features of PRISM
  - strategy (adversary) synthesis
  - multi-objective verification

– Further/related work
  - task graph scheduling [FMSD'13]
  - probabilistic parameter synthesis [TASE'13]
  - strategy generation for autonomous driving [QEST'13,MFCS'13]
  - template-based synthesis for UAV missions [ESEC/FSE'13]

References

- Tutorial papers
  - V. Forejt, M. Kwiatkowska, G. Norman and D. Parker. *Automated Verification Techniques for Probabilistic Systems*. In SFM'11, pp 53-113, Springer, 2011.
  - V. Forejt, M. Kwiatkowska, G. Norman, D. Parker and H. Qu. *Quantitative Multi-Objective Verification for Probabilistic Systems*. In Proc. TACAS'11, pp 112-127, Springer, 2011.
  - V. Forejt, M. Kwiatkowska and D. Parker. *Pareto Curves for Probabilistic Model Checking*. In Proc. ATVA'12, pp 317-332, Springer, 2012.
  - T. Chen, V. Forejt, M. Kwiatkowska, D. Parker and A. Simaitis. *Automatic Verification of Competitive Stochastic Systems*. FMSD, 43(1), pp 61-92, Springer, 2013.
  - G. Norman, D. Parker and J. Sproston. *Model Checking for Probabilistic Timed Automata*. FMSD, 43(2), pp 164-190, Springer, 2013.
- PRISM tool paper
  - M. Kwiatkowska, G. Norman and D. Parker. *PRISM 4.0: Verification of Probabilistic Real-time Systems*. In Proc. CAV'11, volume 6806 of LNCS, pages 585-591, Springer. July 2011.

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
  - PRISM-games: www.prismmodelchecker.org/games