Register at: essai.si

ESSAI & ACAI 2023
LJUBLJANA, SLOVENIA

MODEL UNCERTAINTY IN SEQUENTIAL DECISION MAKING

DAVID PARKER
University of Oxford

BRUNO LACERDA
University of Oxford

NICK HAWES
University of Oxford

Lectures 1-3

Dave Parker

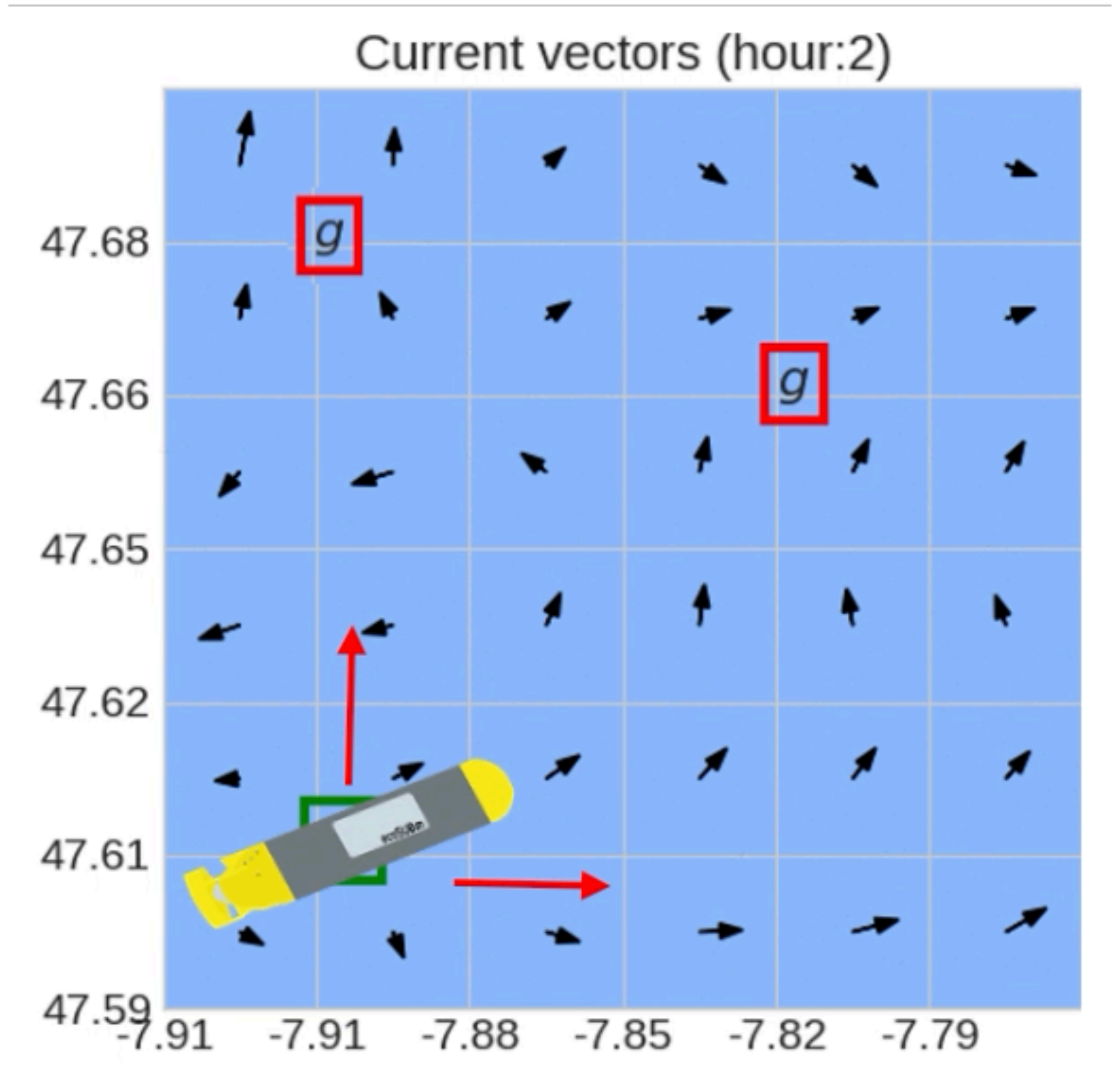
University of Oxford

DEPARTMENT OF COMPUTER SCIENCE

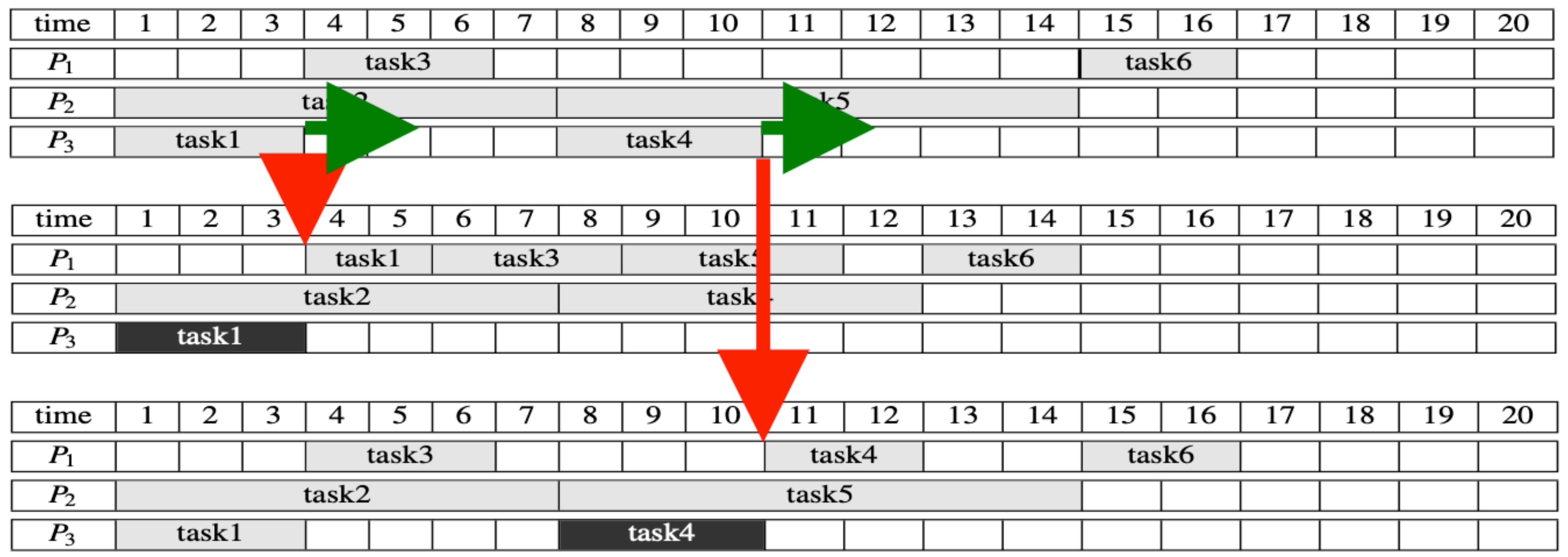
Introduction

Sequential decision making under uncertainty

- Sequential decision making
  - iterative interaction with an environment to achieve a goal
  - sequential process of making observations and executing actions
  - applications in: health, energy, transportation, robotics, …



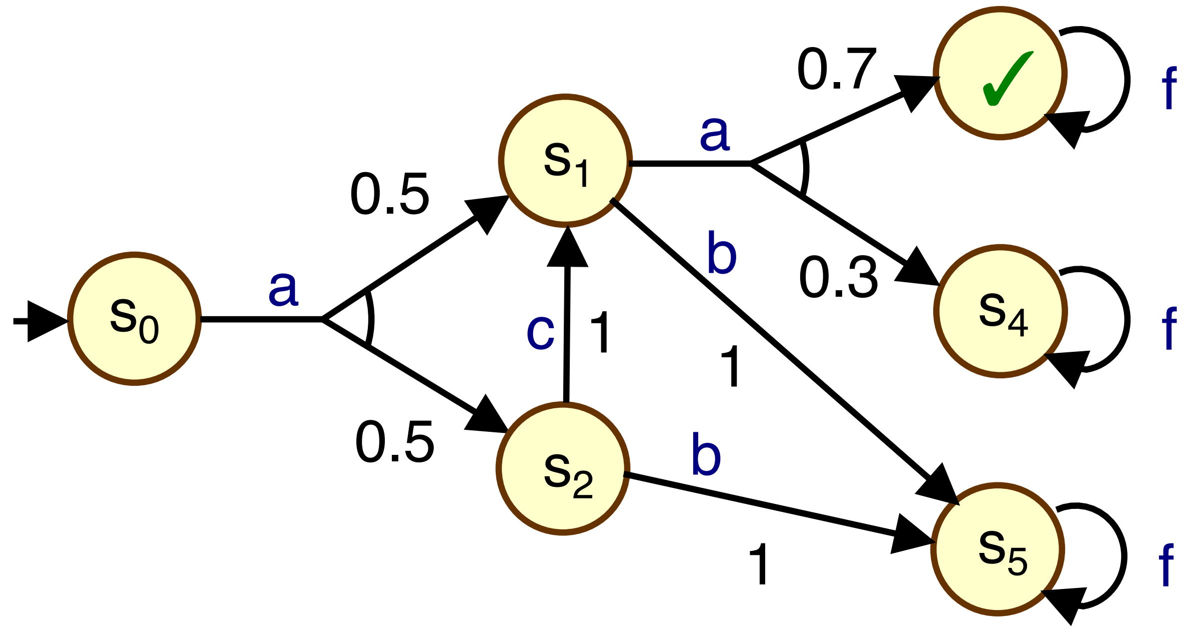
- Sequential decision making under uncertainty
  - noisy sensors, unpredictable conditions, lossy communication, human behaviour, hardware failures, …

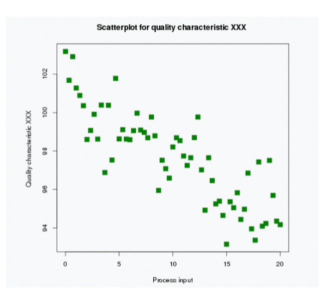


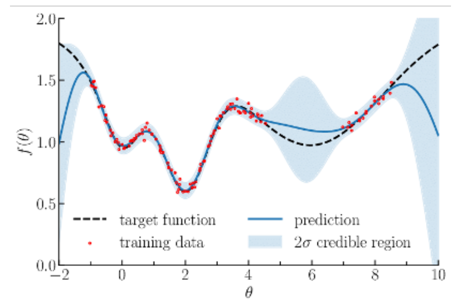
- Trustworthy, safe and robust decision making
  - e.g. for safety-critical applications
  - needs rigorous/systematic quantification of uncertainty

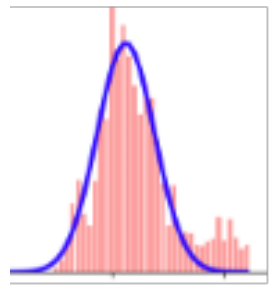
Reasoning about uncertainty

- Markov decision processes (MDPs) and variants
  - standard models for sequential decision making under uncertainty
  - stochastic processes quantify uncertainty
  - but parameters of these often need to be estimated from data









- We will distinguish between:

- Aleatoric uncertainty (randomness intrinsic to environment)
  - e.g., sensor noise, actuator failure, human decisions

(1)

P(H)=P(T)=0.5

- Epistemic uncertainty (quantifies lack of knowledge)
  - reducible: can reduce by collecting more data/observations
  - e.g., poor model quality due to low number of measurements

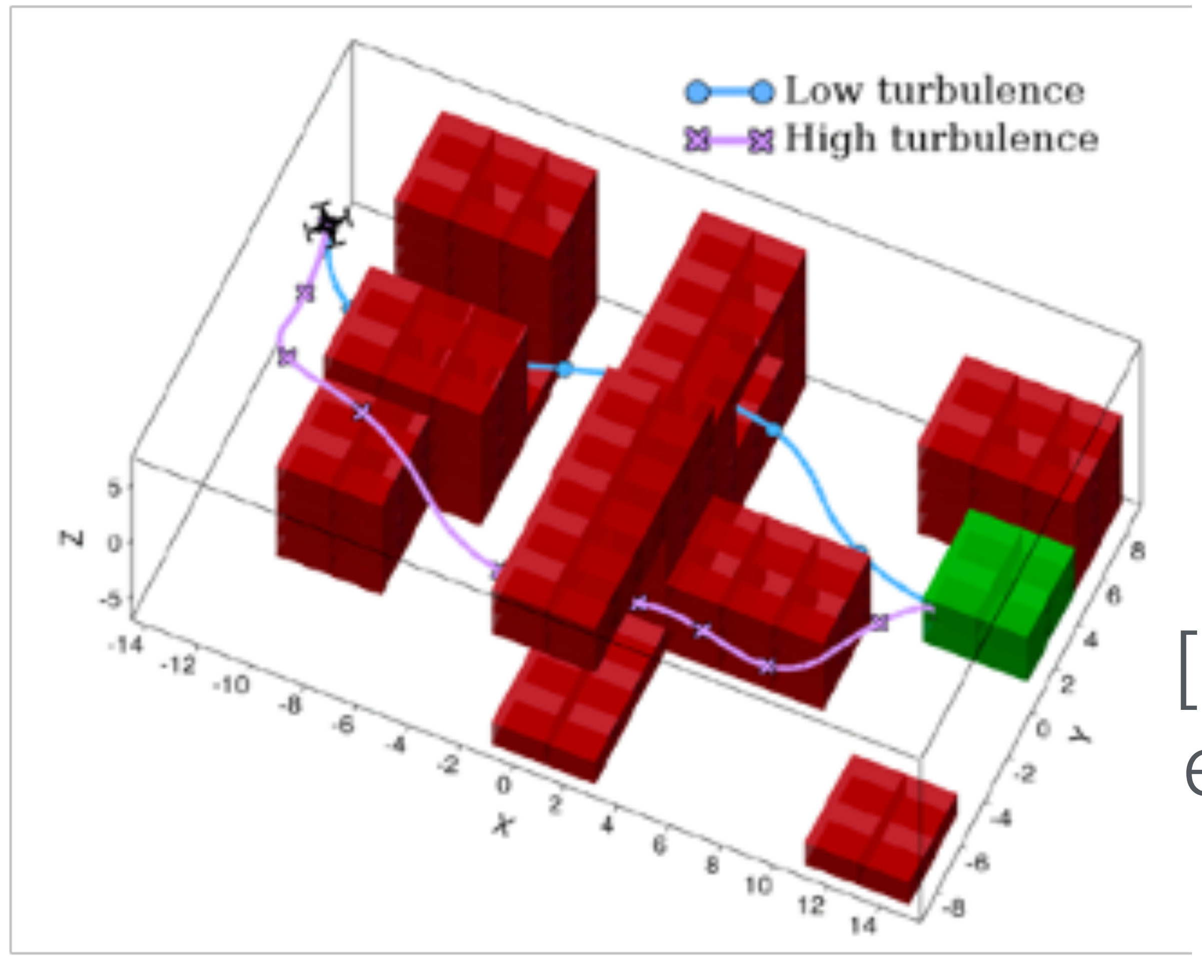
(2)

?

P(H)=1 P(T)=1

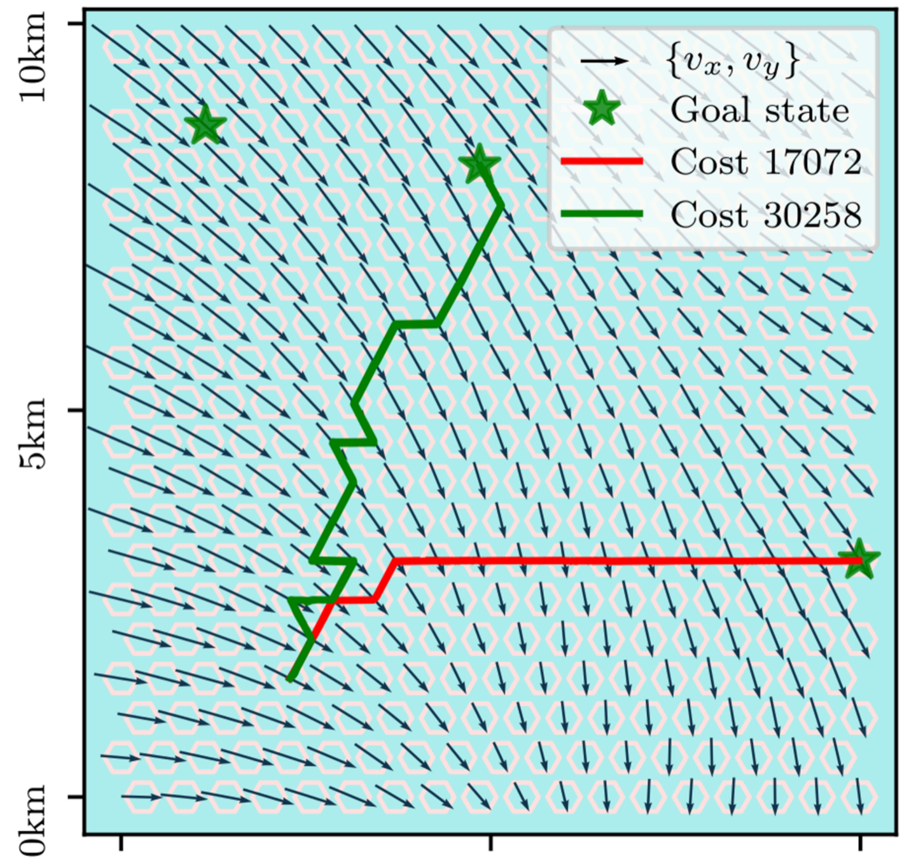
Applications & challenges

- Unmanned aerial vehicle
  - robust control in the presence of turbulence



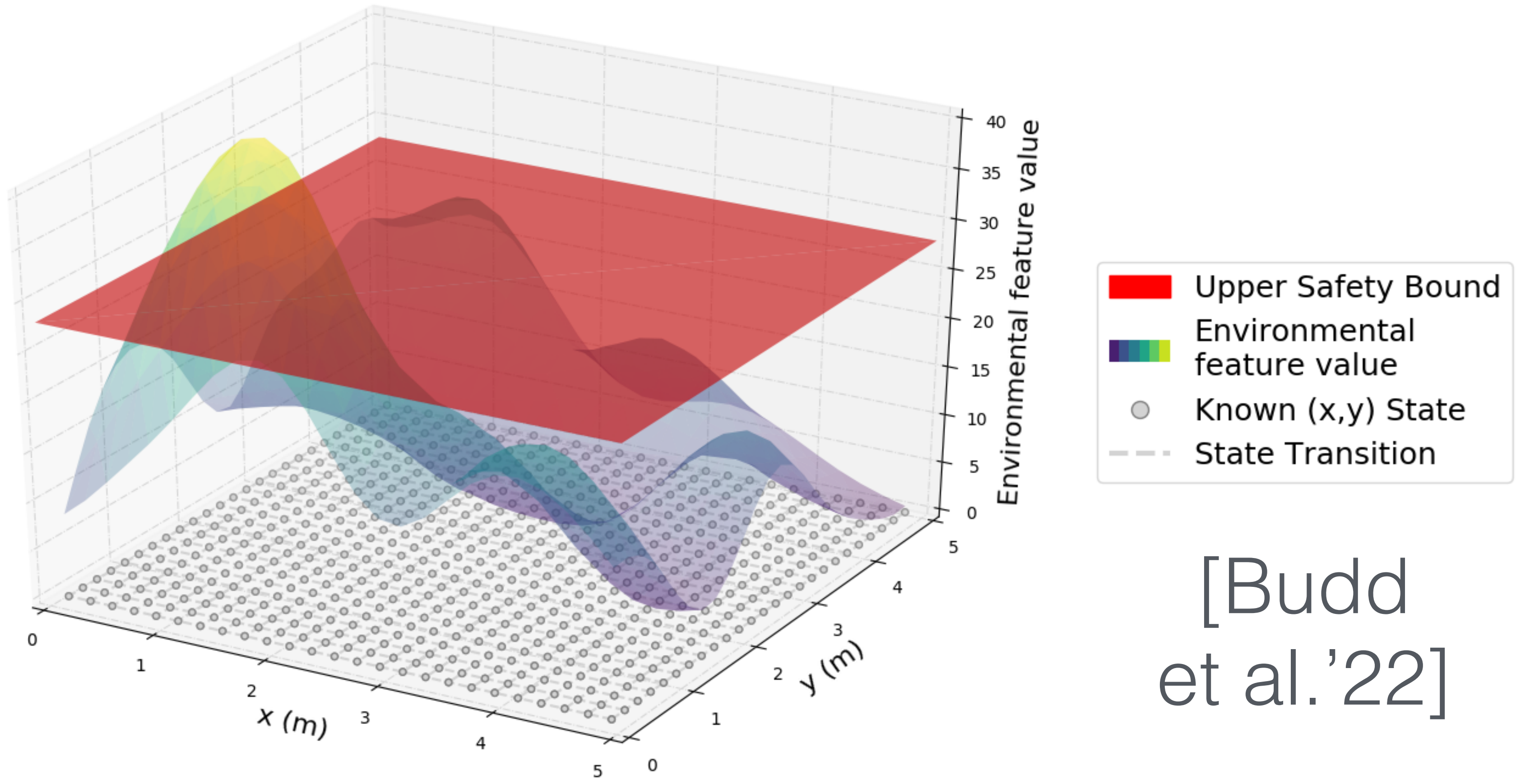
[Badings et al.'23]

- Autonomous underwater vehicle
  - effective navigation against unknown ocean currents



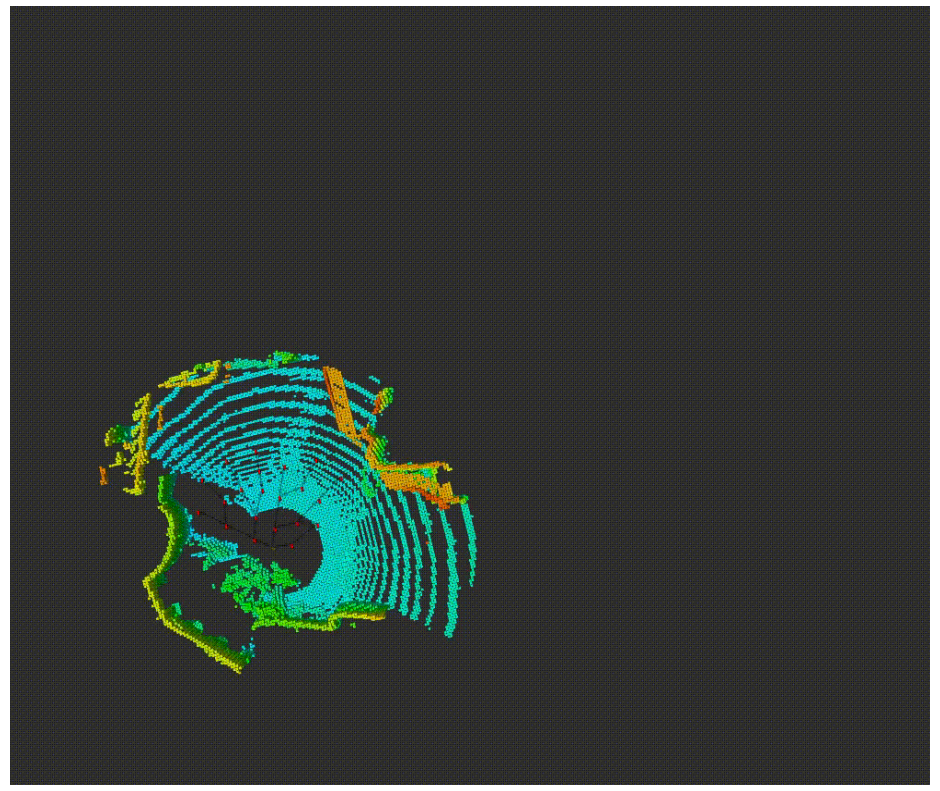
[Budd et al.'22]

- Radiation measuring
  - safe navigation and task completion in unknown environments

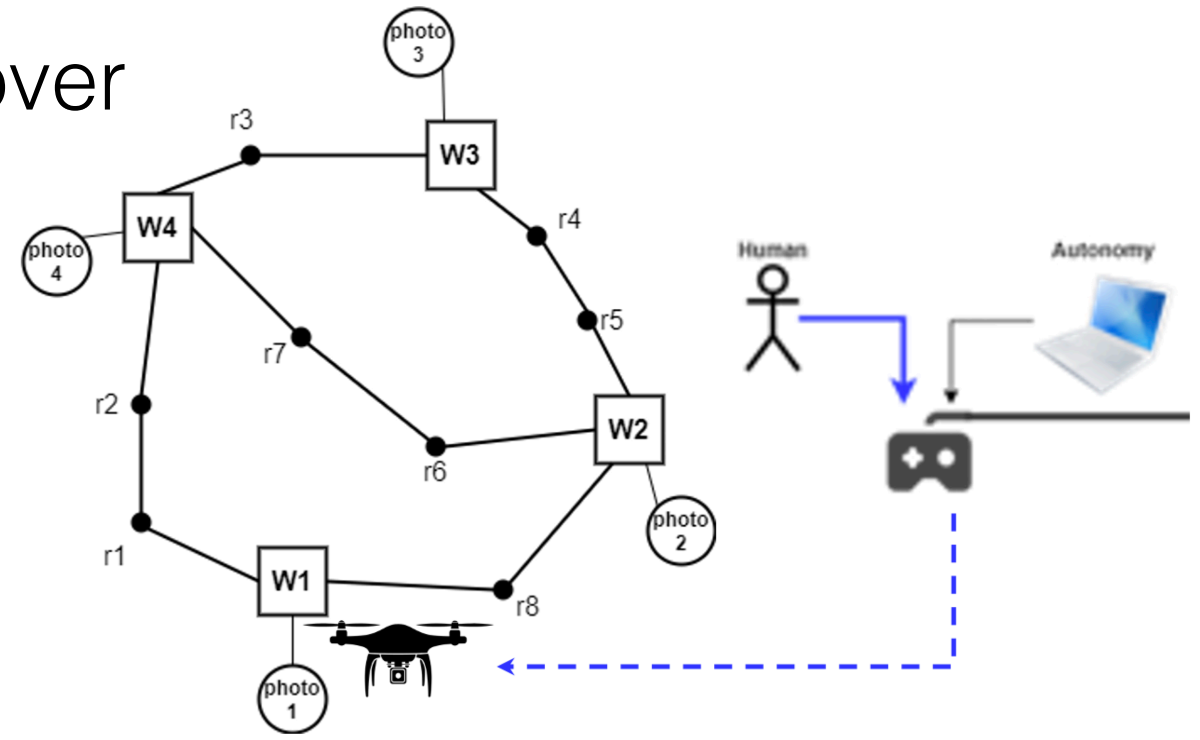


[Budd et al.'22]

- Mine exploration
  - Safe exploration and mapping (avoiding radiation)



- Shared autonomy
  - learning belief over uncertainty on unobservable human state

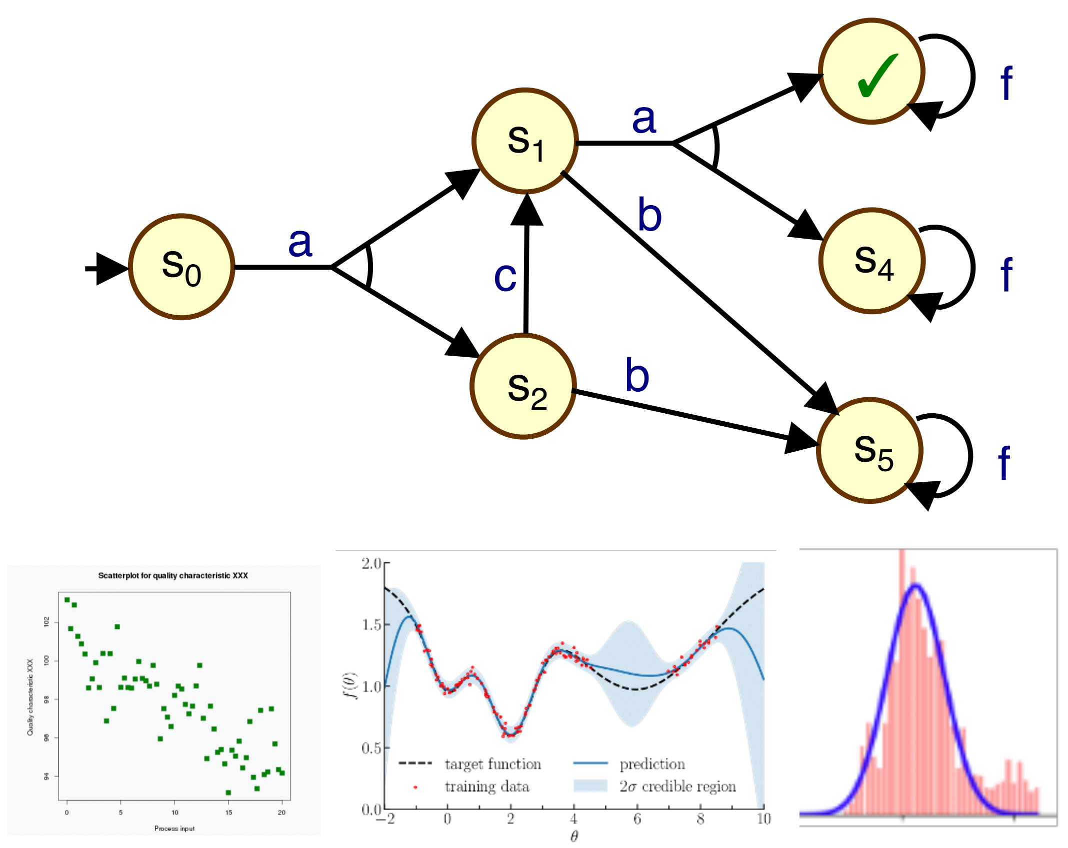


[Costen et al.'22]

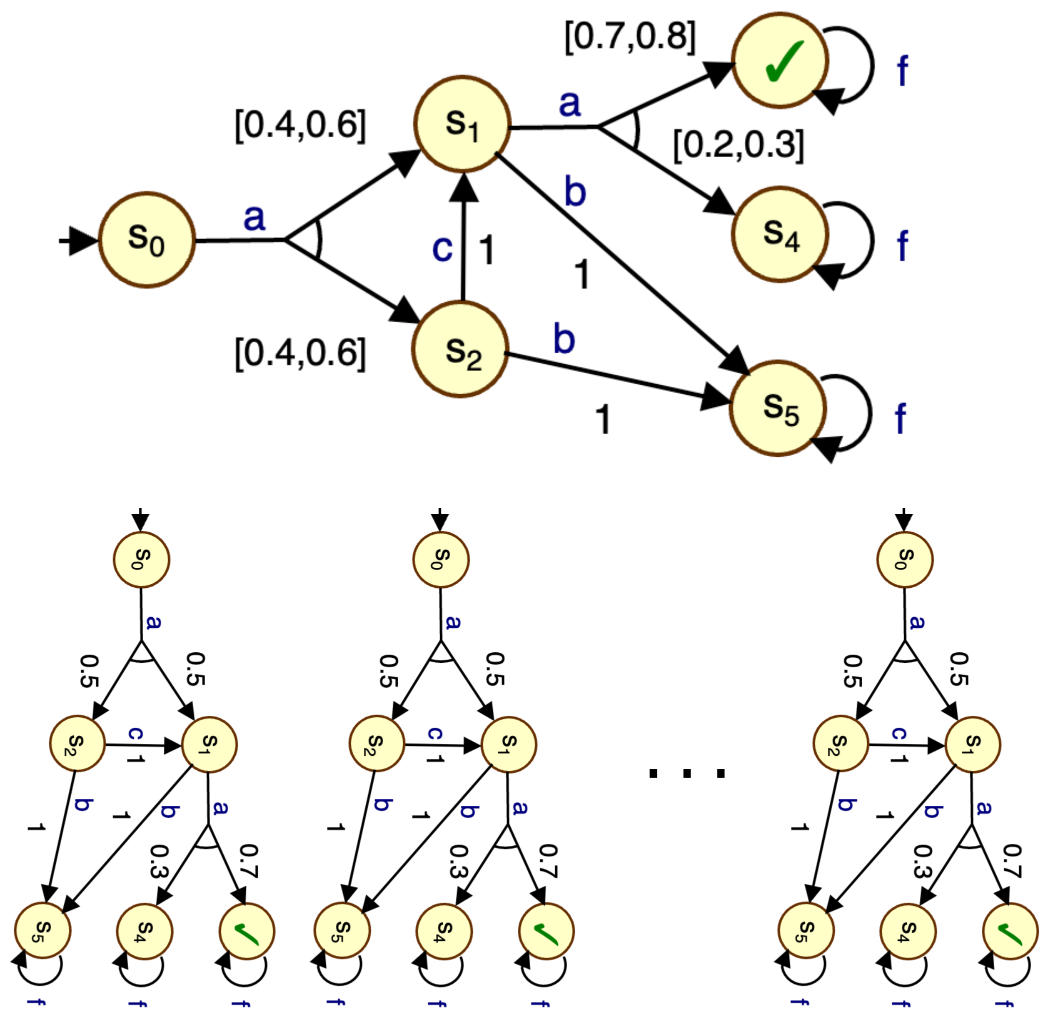
This course

- Model uncertainty in sequential decision making
  - model-based techniques (probabilistic planning, not reinforcement learning)
  - discrete time, discrete space
  - fully observable environments (mostly)
  - rigorous/precise/systematic quantification of uncertainty

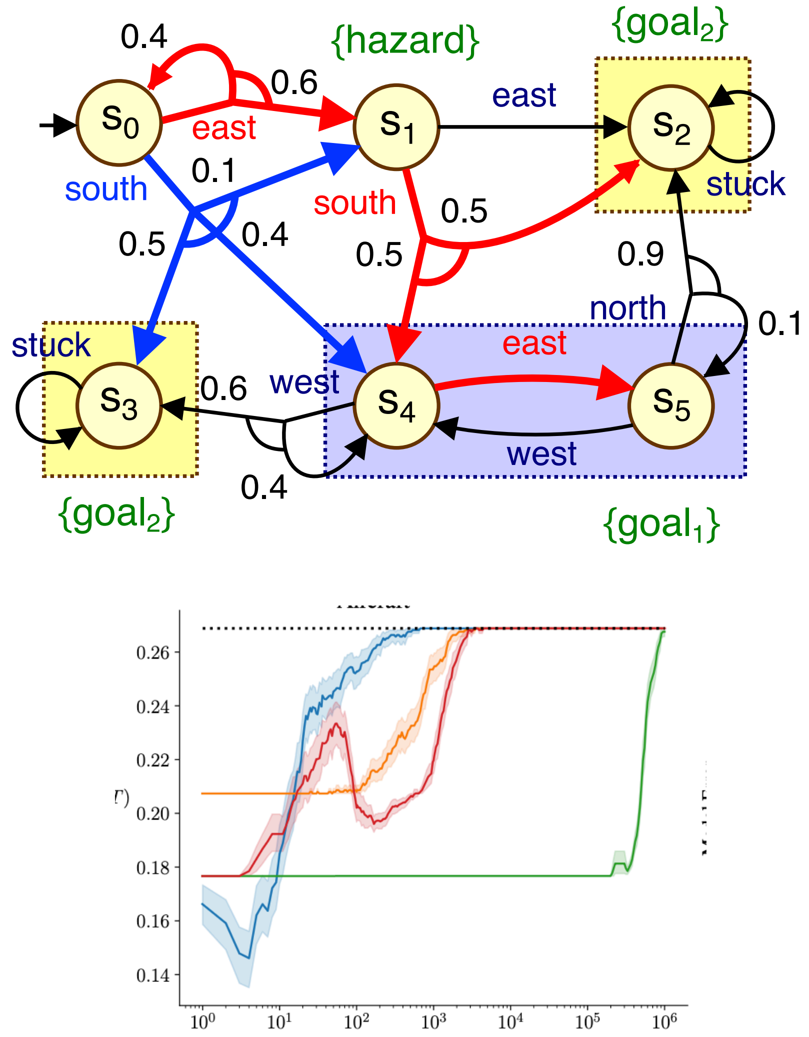
models + data



uncertain MDPs



policies + analysis & guarantees



Course contents

- Markov decision processes (MDPs) and stochastic games
  - MDPs: key concepts and algorithms
  - stochastic games: adding adversarial aspects
- Uncertain MDPs
  - MDPs + epistemic uncertainty, robust control, robust dynamic programming, interval MDPs, uncertainty set representation, challenges, tools
- Sampling-based uncertain MDPs
  - removing the transition independence assumption
- Bayes-adaptive MDPs
  - maintaining a distribution over the possible models

Lecture 1

Lecture 2

Lecture 3

Lecture 4

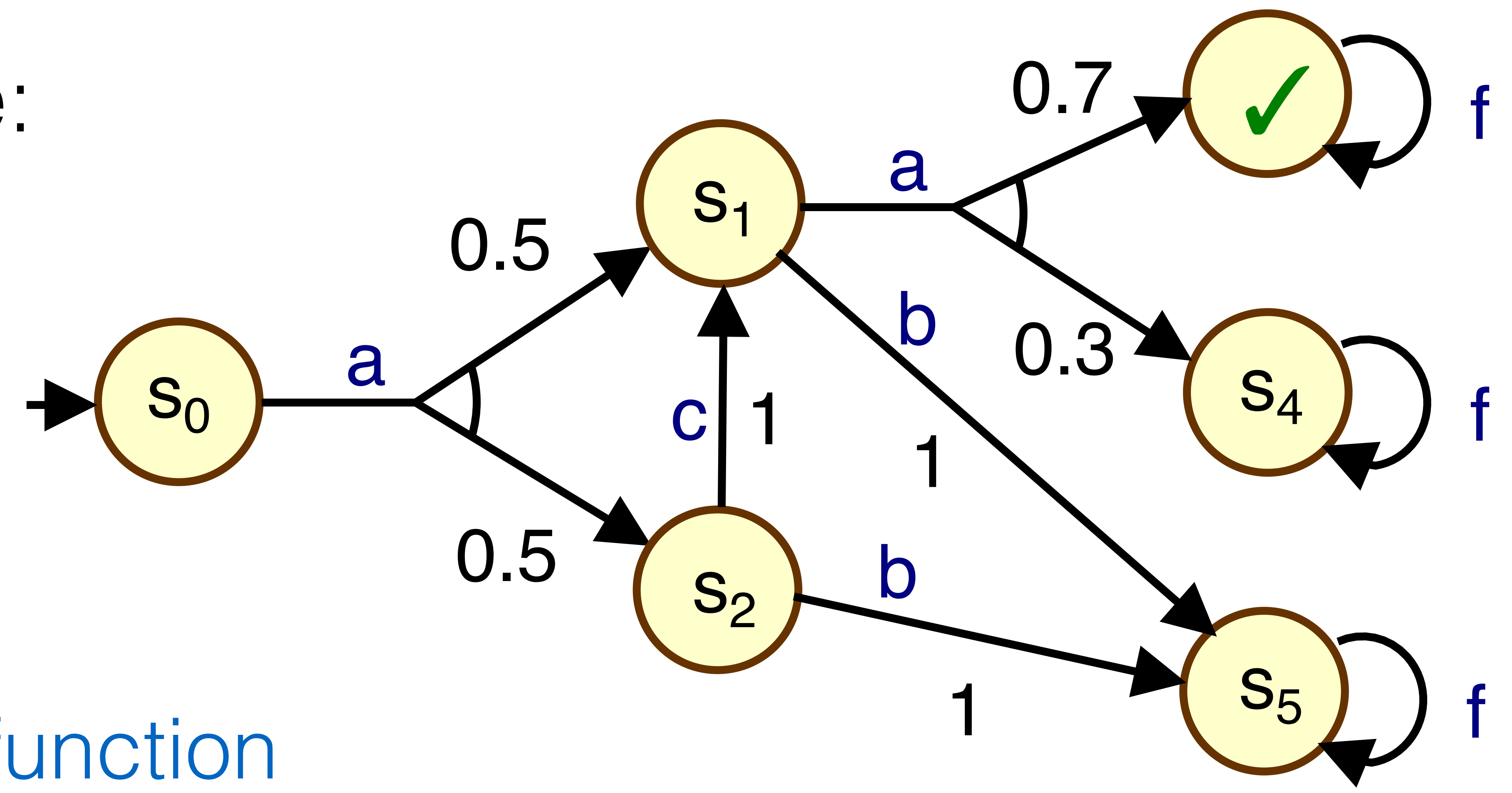
Lecture 5

Markov decision processes

Markov decision processes

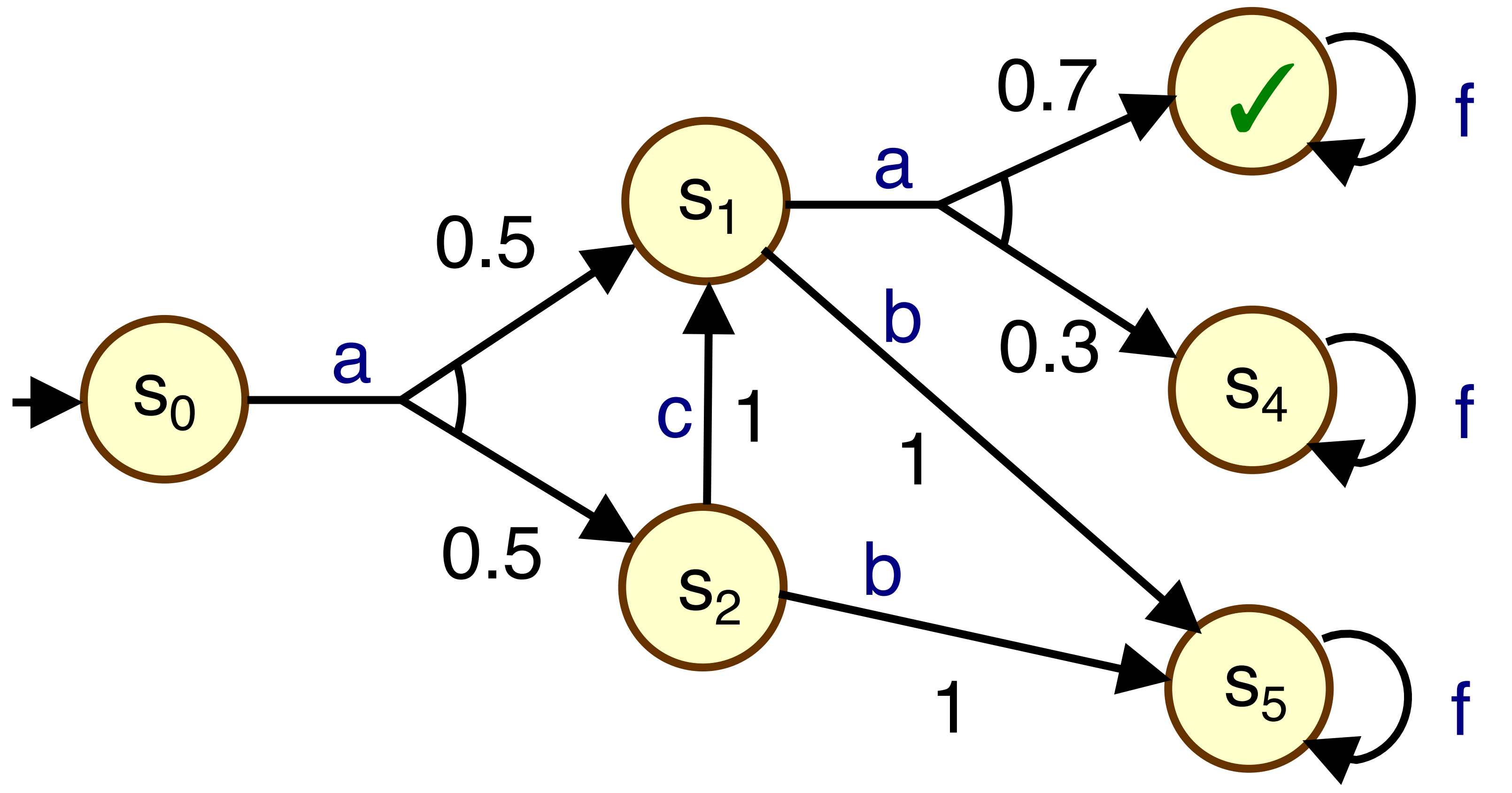
- Markov decision processes (MDPs)
  - standard model for sequential decision making under uncertainty

- An MDP is of the form $\mathcal{M} = (S, s_0, A, P)$ where:
  - S is a (finite) set of states
  - $s_0 \in S$ is an initial state
  - A is a (finite) set of actions
  - $P : S \times A \times S \to [0,1]$ is a transition probability function
    - where $\Sigma_{s' \in S} P(s, a, s') \in \{0,1\}$



Markov decision processes

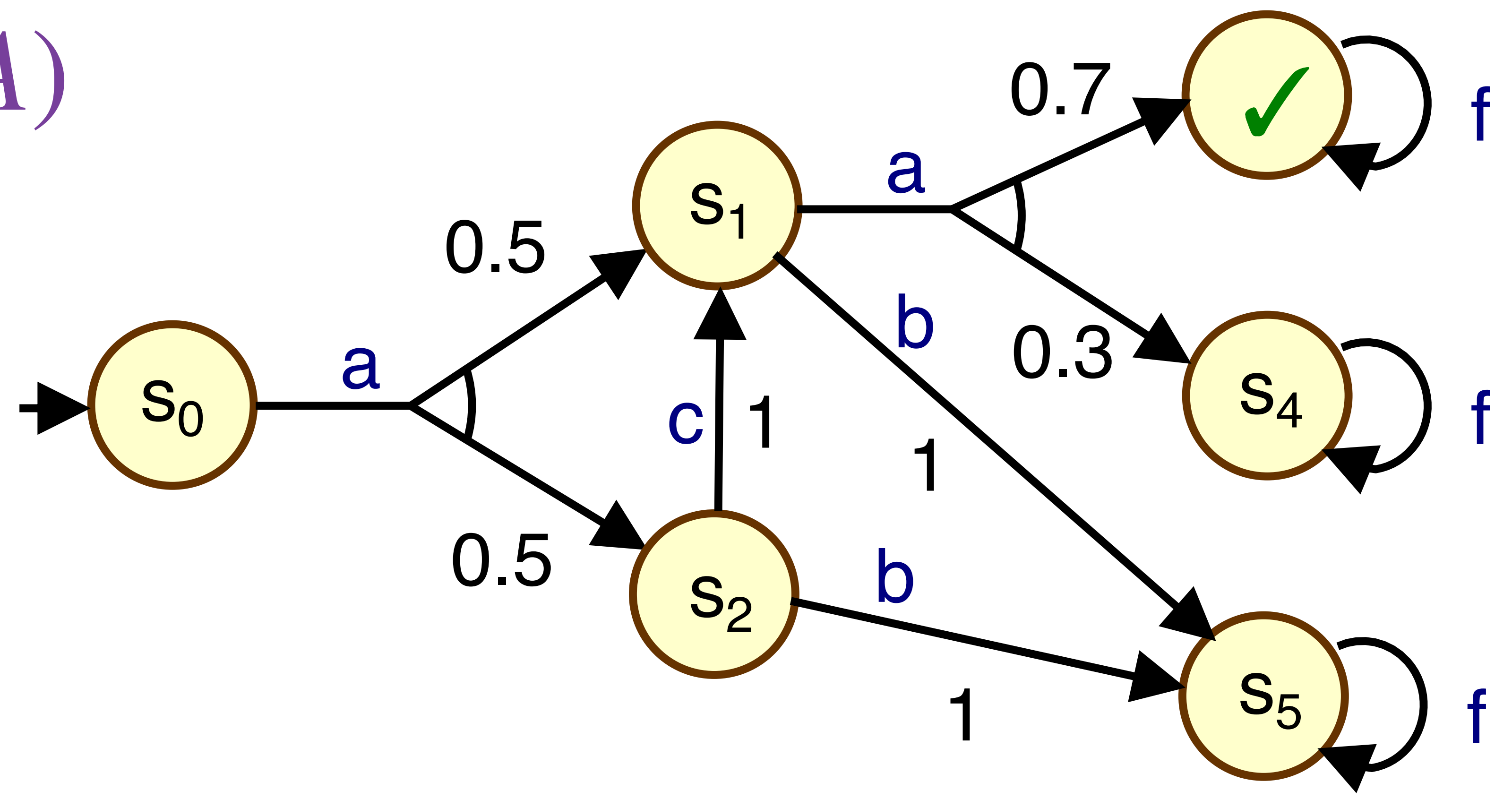
- For an MDP $\mathcal{M} = (S, s_0, A, P)$:
  - the enabled actions $A(s) \subseteq A$ in each state s
    - are $A(s) = \{a \in A : P(s, a, s') > 0$ for some $s'\}$
  - a path is a sequence $\omega = s_0 a_0 s_1 a_1, \ldots$
    - such that $s_i \in S$, $a_i \in A(s_i)$ and $P(s_i, a_i, s_{i+1}) > 0$ for all i



- We also use:
  - $P^a : S \times S \to [0,1]$ is the transition probability matrix for each $a \in A$
  - $P^a_s \in Dist(S)$ is the successor distribution for each state s and action $a \in A(s)$
  - (where $Dist(S)$ is the set of discrete probability distributions over set S)

Policies for MDPs

- Policies (or strategies) π resolves the choice of action in each state
  - based on the execution of the MDP so far
  - formally: a policy is a mapping $\pi : (S \times A)^* \times S \to Dist(A)$
    - such that $\pi(s_0 a_0 \ldots s_n)(a_n) > 0$ implies $a_n \in A(s_n)$
  - $\pi(s_0 a_0 \ldots s_n)(a_n)$ is the probability of picking a_n after observing MDP history $s_0 a_0 \ldots s_n$



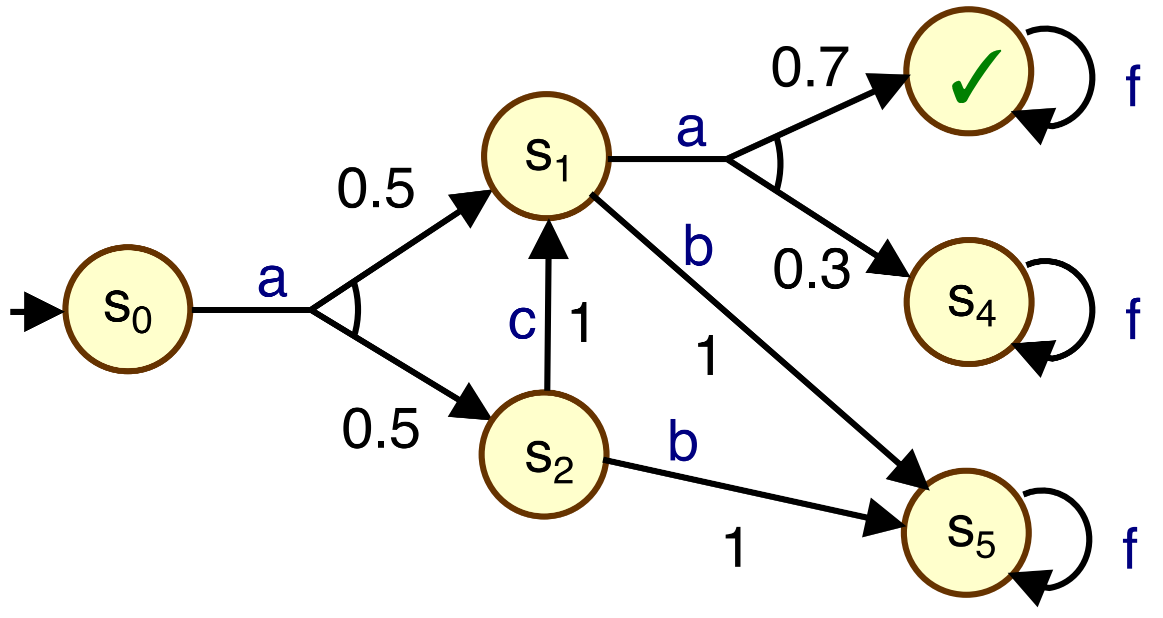
- $\Pi_{\mathscr{M}}$ (or just Π) is the set of all (deterministic) policies for MDP $\mathscr{M}$
- Policies can be classified by (i) use of randomisation; (ii) use of memory
  - which matter for optimality, computation, practicality, …

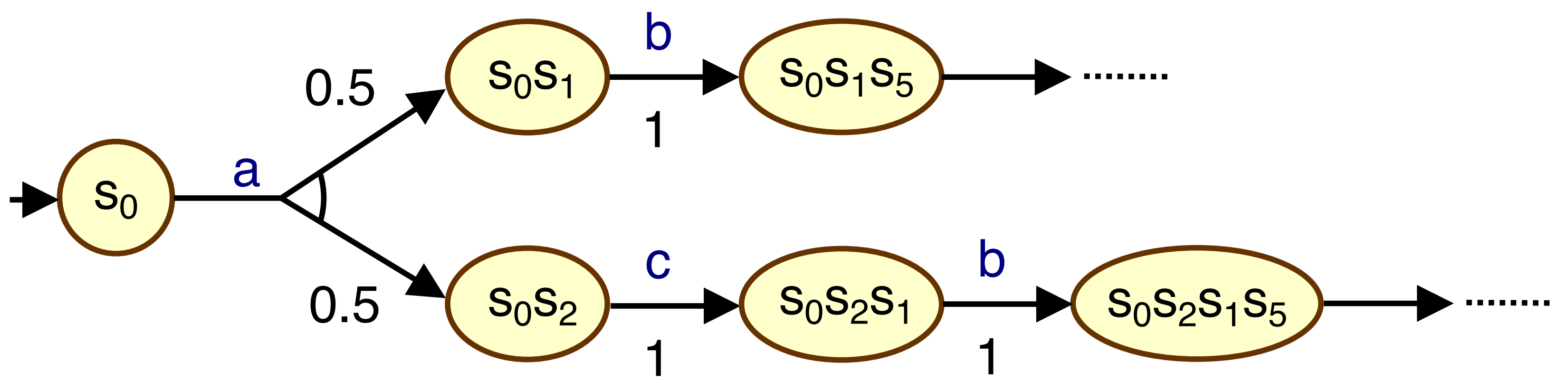
Classes of policies for MDPs

- Randomisation
  - π is deterministic (or pure) if it always picks a single action with probability 1
  - and randomised (or probabilistic) otherwise
  - for now, we'll mostly assume deterministic policies and assume $\pi : (S \times A)^* \times S \to A$
- Memory
  - π is memoryless (or stationary, or Markovian) if $\pi(s_0, \ldots, s_n) = \pi(s'_0, \ldots, s'_n)$ when $s_n = s'_n$
    - in which case we write it in the form $\pi : S \to A$
    - $\Pi_m \subseteq \Pi$ is the set of all memoryless policies
  - otherwise π is history dependent
  - π is finite-memory if it suffices to distinguish a finite number of "modes" based on the history
  - sometimes write a (time-dependent) policy as tuple $\pi = (\pi_0, \pi_1, \ldots)$ where $\pi_i : S \to A$

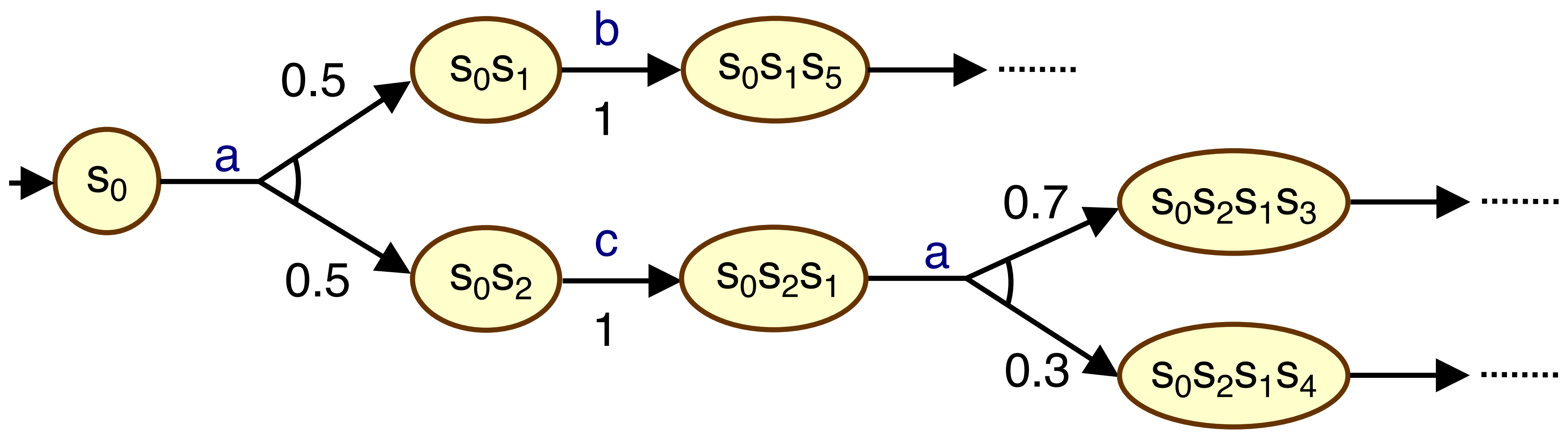
MDPs and policies

- A policy for an MDP yields an induced Markov chain
    - and set of (infinite) paths

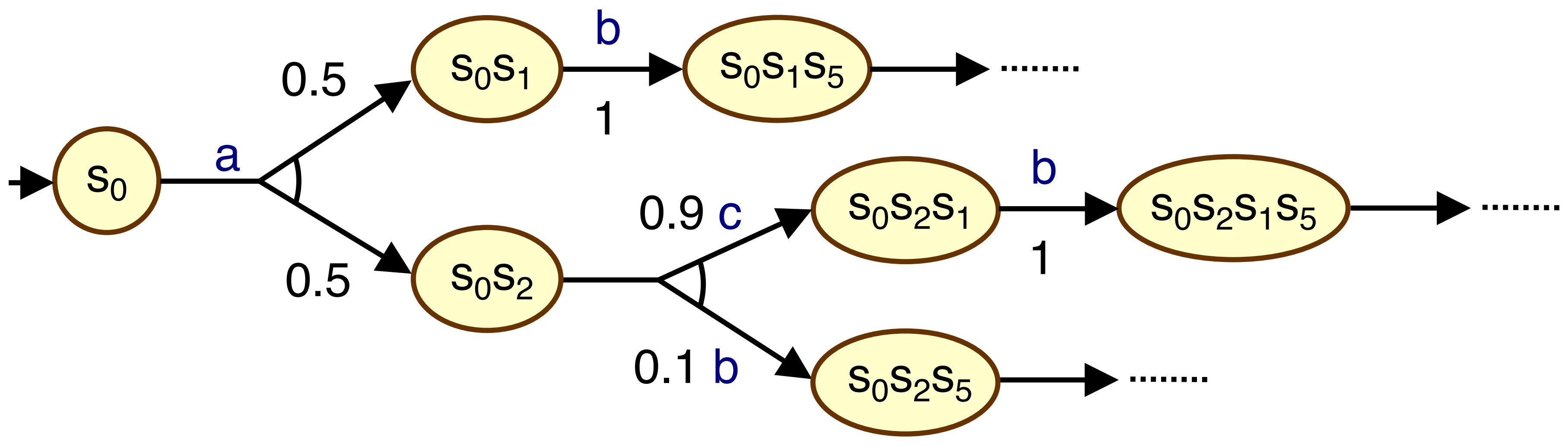




(memoryless, deterministic)



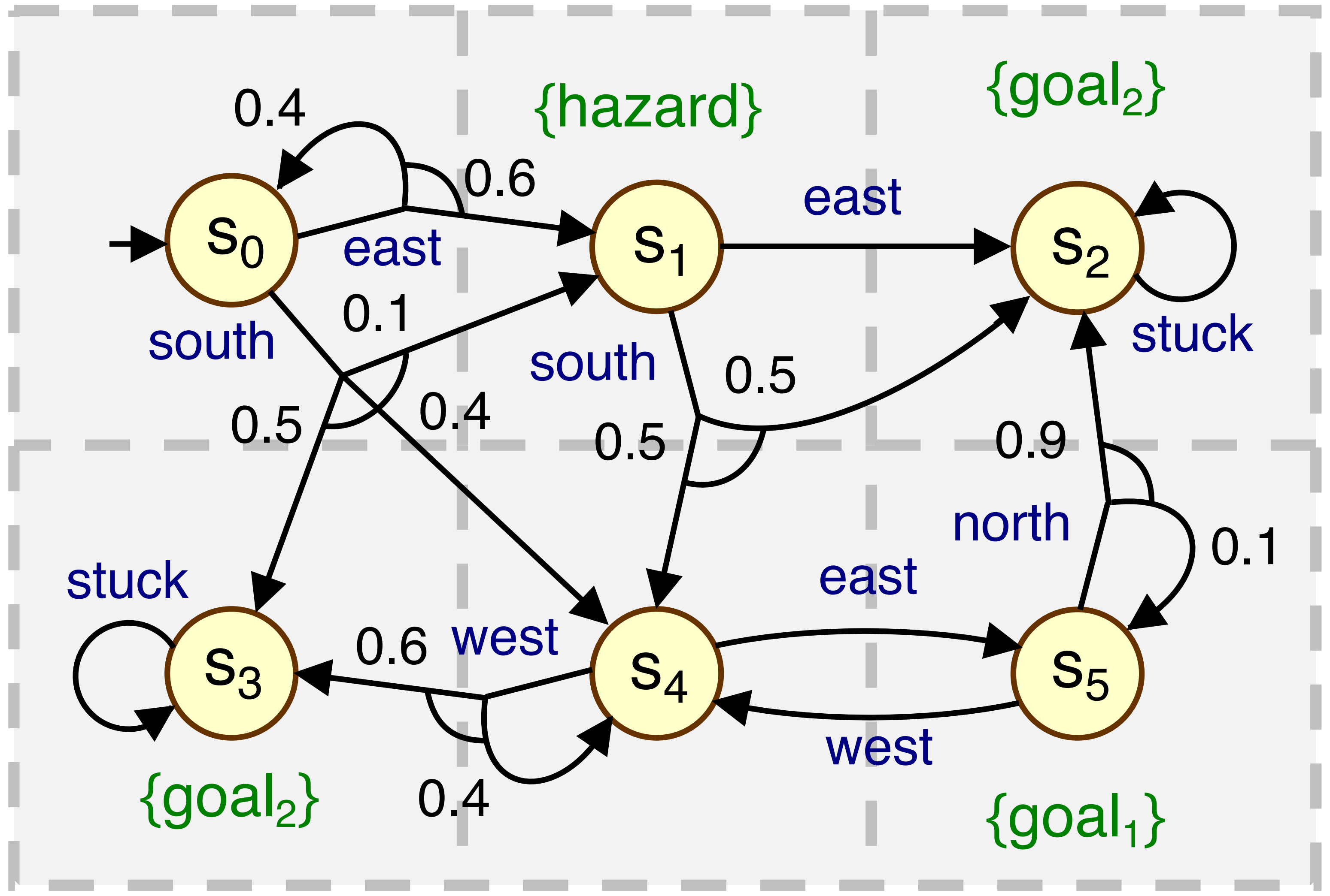
(finite-memory, deterministic)



(memoryless, randomised)

Running example (and objectives)

- Example MDP: robot moving through terrain divided in to 3 x 2 grid



- Objectives (or properties) define an optimisation problem for an MDP
  - MaxProb: maximise the probability of reaching $goal \subseteq S$
  - SSP (stochastic shortest path): minimise the cost of reaching $goal \subseteq S$

} we'll focus mainly on these two

Defining objectives for MDPs

- Execution of an MDP under a policy
  - for a policy $\pi \in \Pi$ on MDP $\mathcal{M}$…
  - Pr_s^π is a probability measure over all (infinite) paths from state s of $\mathcal{M}$
  - $\mathbb{E}_s^\pi(X)$ is the expected value of X (with respect to Pr_s^π)
    - where $X : (S \times A)^\omega \to \mathbb{R}_{\geq 0}$ is a random variable over (infinite) paths

- Value function: $V^\pi : S \to \mathbb{R}$
  - gives the value of an objective under π starting from each state of the MDP
  - define optimal value, e.g.: $V^*(s) = \max_{\pi \in \Pi} V^\pi(s)$
  - and optimal policy, e.g.: $\pi^* = \operatorname{argmax}_{\pi \in \Pi} V^\pi(s_0)$

MaxProb & SSP (stochastic shortest path)

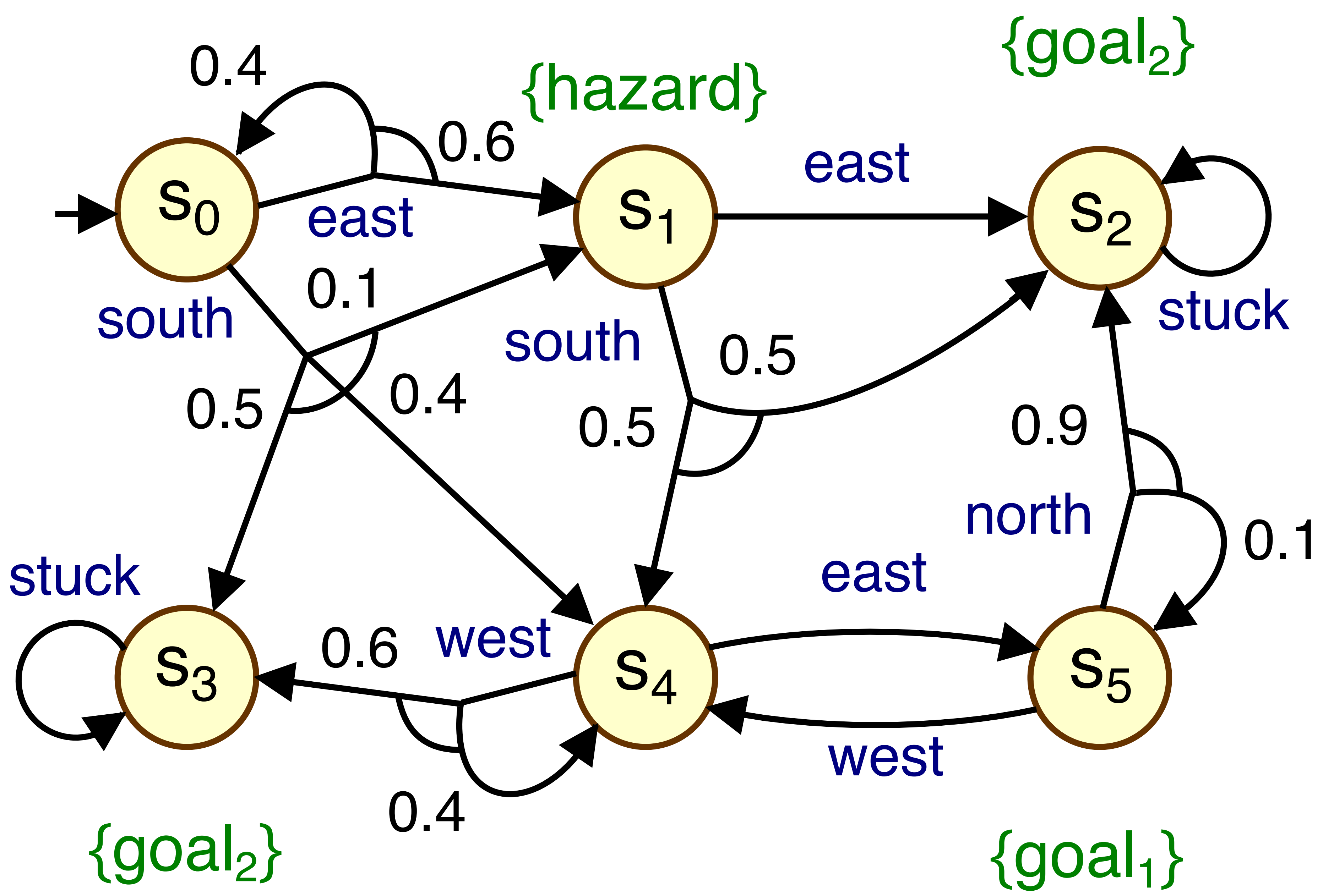
- MaxProb: Maximise the probability of reaching a target state set $goal \subseteq S$
  - maximise $V^\pi(s) = Pr_s^\pi(\{s_0 a_0 s_1 a_1 s_2 \ldots : s_i \in goal$ for some $i\})$

- SSP: Minimise the expected cost of reaching a target state set $goal \subseteq S$
  - for a cost function $C : S \times A \to \mathbb{R}_{\geq 0}$
  - minimise $V^\pi(s) = \mathbb{E}_s^\pi(X^C)$ where $X^C(s_0 a_0 s_1 a_1 \ldots) = \Sigma_{i=0}^{\infty} C(s_i, a_i)$

- Assumptions for SSP
  - $goal$ states are absorbing and zero-cost
  - there is a proper policy (i.e., which reaches $goal$ with probability 1 from all states)
  - every improper policy incurs an infinite cost from every state from which it does not reach $goal$ with probability 1

Running example: MaxProb

- What is the optimal policy for objective MaxProb(goal$_1$)?



Other objectives

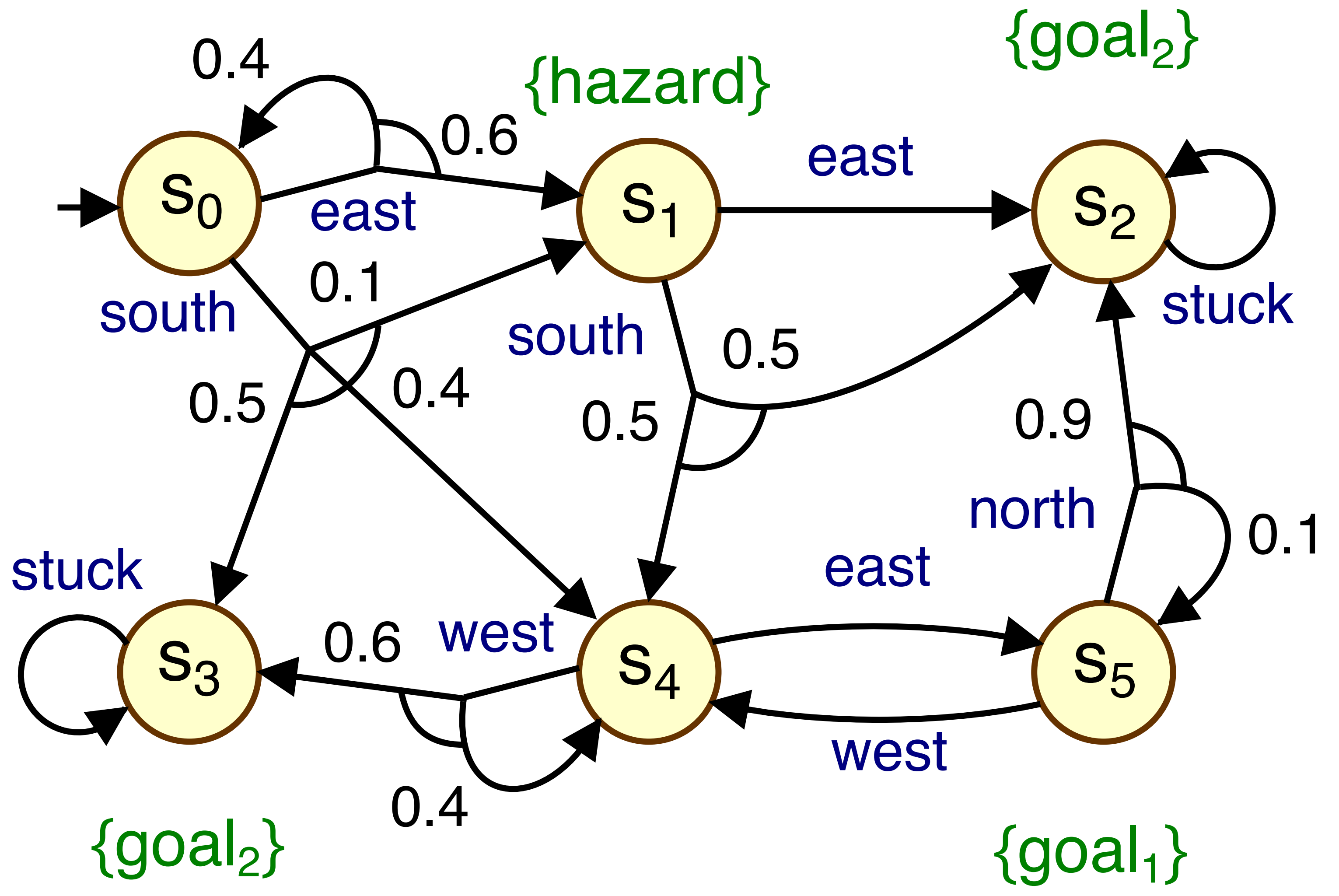
- Some other common objectives for MDPs:

- Finite-horizon variants, e.g., of MaxProb:
  - MaxProb$^{\leq k}$: Maximise the probability of reaching $goal \subseteq S$ within time horizon k
  - maximise $V^{\pi}(s) = Pr_s^{\pi}(\{s_0 a_0 s_1 a_1 s_2 \ldots : s_i \in goal$ for some $i \leq k\})$

- Discounting infinite-horizon objectives
  - DiscSum: Maximise the expected discounted total reward sum
  - for a reward function $R : S \times A \rightarrow \mathbb{R}$ and discount factor $\gamma \in (0,1)$
  - maximise $V^{\pi}(s) = \mathbb{E}_s^{\pi}(X^R)$ where $X^R(s_0 a_0 s_1 a_1 \ldots) = \Sigma_{i=0}^{\infty} \gamma^i R(s_i, a_i)$

Temporal logic objectives

- Specification languages from formal verification
  - probabilistic extensions of temporal logics, e.g., PCTL, PLTL
- Examples
  - $P_{max=?}$ [F $goal_1$] - "probabilistic reachability"
  - $P_{max=?}$ [$F^{\leq 10}$ $goal_1$] - "probabilistic bounded reachability"
  - $P_{max=?}$ [G ¬hazard] - "probabilistic safety/invariance"
  - $P_{max=?}$ [¬hazard U $goal_1$] - "probabilistic reach-avoid"
  - $P_{max=?}$ [(G¬hazard) ∧ (GF $goal_1$)] - "maximise probability of avoiding hazard and also visiting goal 1 infinitely often"
  - $P_{max=?}$ [$¬zone_3$ U ($zone_1$ ∧ (F $zone_4$))] - "maximise probability of patrolling zone 1 (whilst avoiding zone 3) then zone 4"
  - $R_{time,min=?}$ [$¬zone_3$ U ($zone_1$ ∧ (F $zone_4$))] - "minimise the expected time to patrol zone 1 (whilst avoiding zone 3) then zone 4"



Solving MDPs

- We will mainly focus on MaxProb (techniques are very similar for SSP)
- Key result: memoryless (deterministic) policies suffice

$$\max_{\pi \in \Pi} V^{\pi}(s) = \max_{\pi \in \Pi_m} V^{\pi}(s)$$

- The optimal value function satisfies the Bellman equation:

$$V^*(s) = \begin{cases} 1 & \text{if } s \in goal \\ \max_{a \in A(s)} \sum_{s' \in S} P_s^a(s') \cdot V^*(s') & \text{otherwise} \end{cases}$$

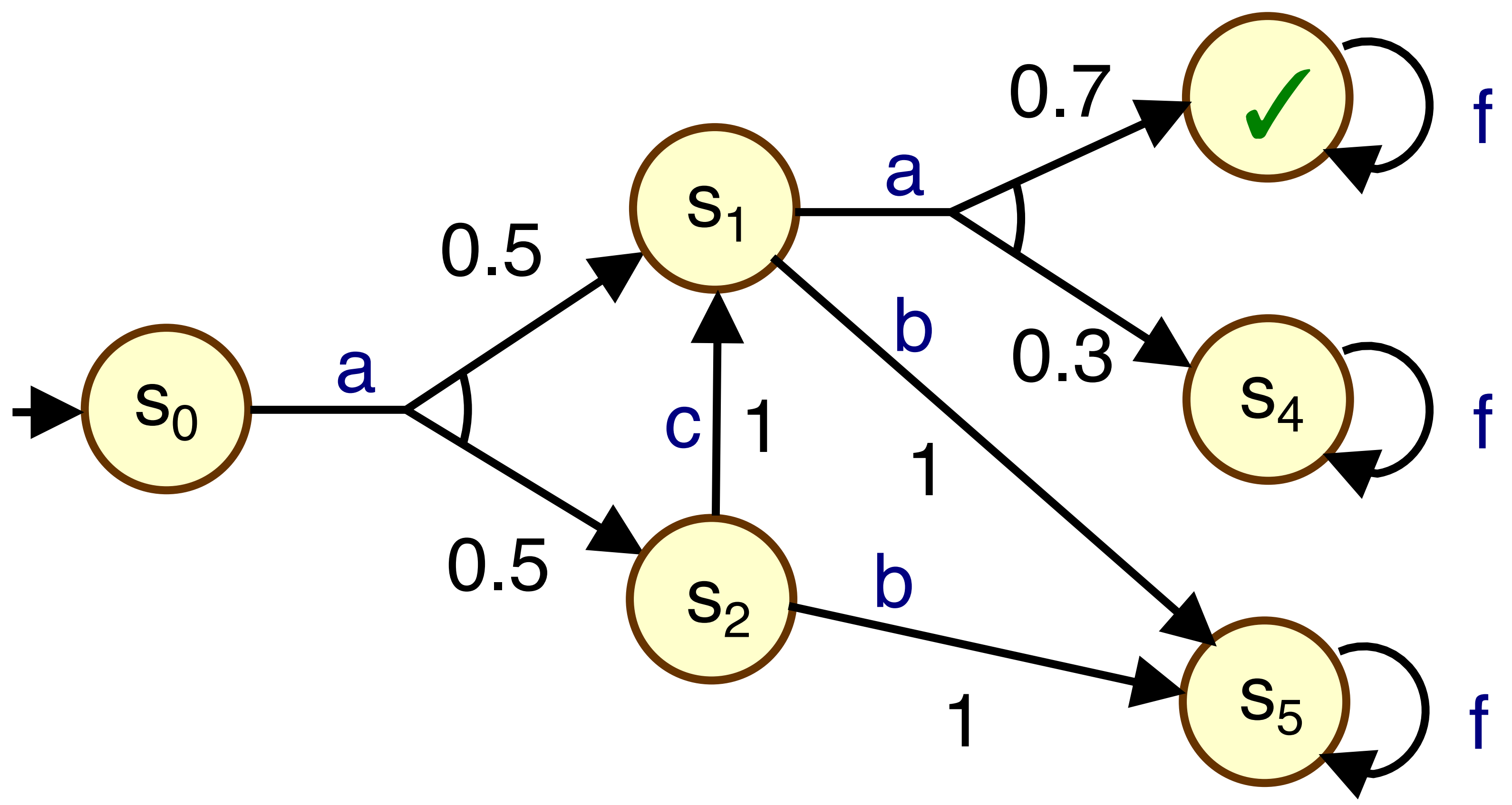
- Solution methods
  - value iteration (dynamic programming)
  - linear programming
  - and many more (e.g., policy iteration, Monte Carlo tree search, BRTDP, …)

MaxProb via value iteration

- Optimal values can be obtained using dynamic programming
  - from the limit of the vector sequence defined below
  - $V^*(s) = \lim_{k\to\infty} x_s^k$ where:

$$x_s^k = \begin{cases} 1 & \text{if } s \in goal \\ 0 & \text{if } s \notin goal \text{ and } k = 0 \\ \max_{a\in A(s)} \sum_{s'\in S} P_s^a(s') \cdot x_{s'}^{k-1} & \text{otherwise} \end{cases}$$

Bellman backup operator



- Known as value iteration (VI)
  - the Bellman operator is (i) monotonic (ii) a contraction in the L_∞ norm
  - optimal values are the least fixed point of the Bellman operator

MaxProb via value iteration

- Optimise via graph-based pre-computation
  - potentially improves accuracy / convergence, resolves uniqueness
  - compute state sets:
    - S^0 = (all) states for which all policies reach $goal$ *with probability 0 (i.e., max = 0)*
    - $S^1 \supseteq goal$ = (some) states for which a policy reaches $goal$ *with probability 1 (i.e., max = 1)*
    - $S^? = S \backslash (S^0 \cup S^1)$

- Then value iteration becomes:

$$x_s^k = \begin{cases} 1 & \text{if } s \in S^1 \\ 0 & \text{if } s \in S^0 \\ 0 & \text{if } s \in S^? \text{ and } k = 0 \\ \max_{a \in A(s)} \sum_{s' \in S} P_s^a(s') \cdot x_{s'}^{k-1} & \text{otherwise} \end{cases}$$

Implementation details:

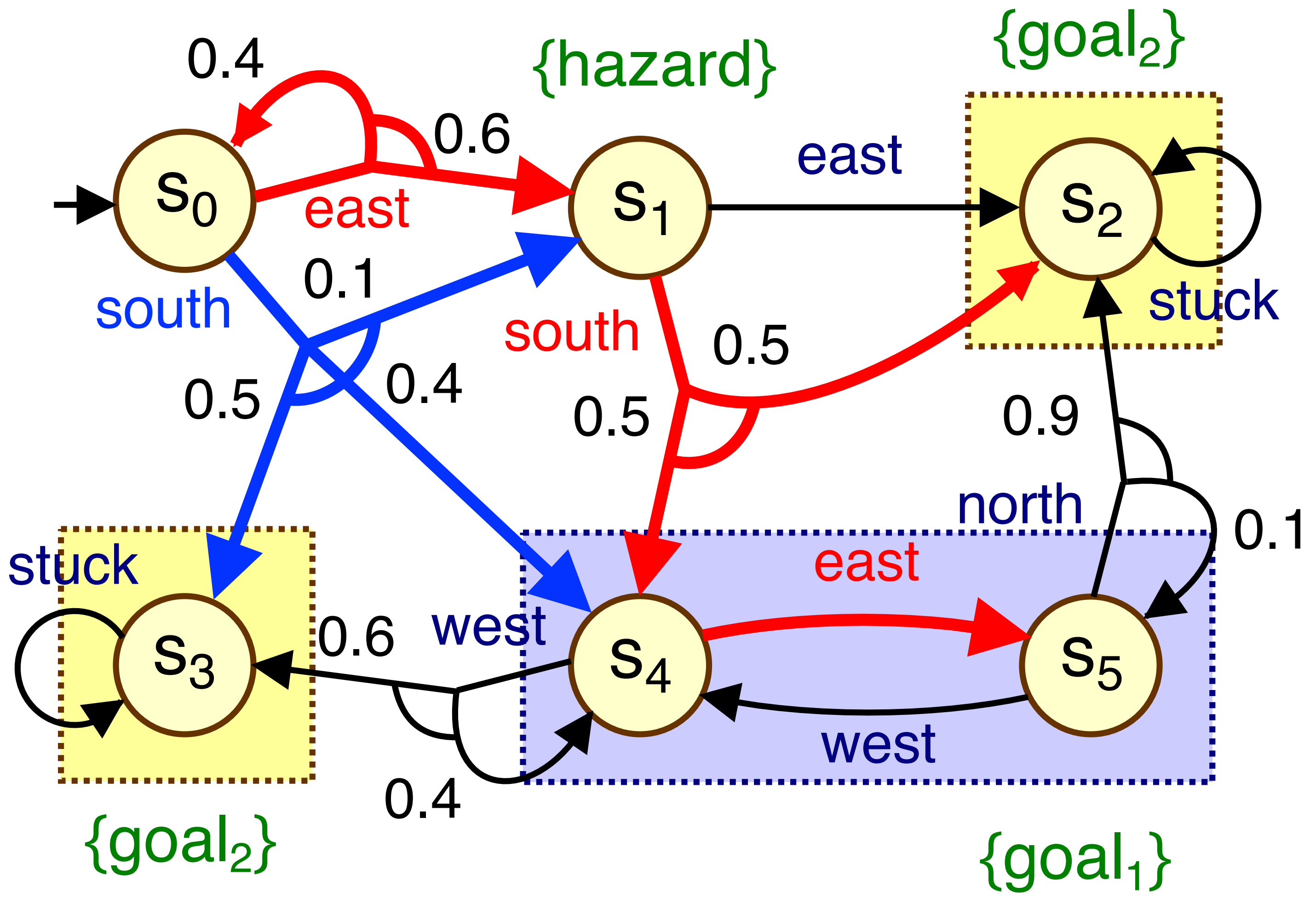
- Extract optimal policy after/during:

$$\pi^*(s) = \operatorname{argmax}_{a \in A(s)} \sum_{s' \in S} P_s^a(s') \cdot x_{s'}^{k-1}$$

- Terminate when $\| x^{k+1} - x^k \| < \varepsilon$
- Choose order to update states s

Running example: Value iteration

- Example: MaxProb(*goal*$_1$)



k	x_0	x_1
0	0	0
1	0.4	0.5
2	0.46	0.5
3	0.484	0.5
4	0.4936	0.5
5	0.49744	0.5
6	0.498976	0.5
7	0.4995904	0.5
8	0.49983616	0.5
9	0.499934464	0.5
10	0.4999737856	0.5

- Fix $x_4=x_5=1$ and $x_2=x_3=0$, just solve for x_0, x_1

- Iteration k=0: $x_0=x_1=0$

- Iteration k=1: $x_0 := \max(0.4\cdot 0 + 0.6\cdot 0,\ 0.1\cdot 0+0.5\cdot 0+0.4\cdot 1)$
$= \max(0, 0.4)$
$= 0.4$

  $x_1 := \max(1\cdot 0,\ 0.5\cdot 0+0.5\cdot 1)$
$= \max(0, 0.5)$
$= 0.5$

- Iteration k=2: $x_0 := \max(0.4\cdot 0.4 + 0.6\cdot 0.5,\ 0.1\cdot 0.5+0.5\cdot 0+0.4\cdot 1)$
$= \max(0.46, 0.45)$
$= 0.46$

  $x_1 := 0.5$ (as before)

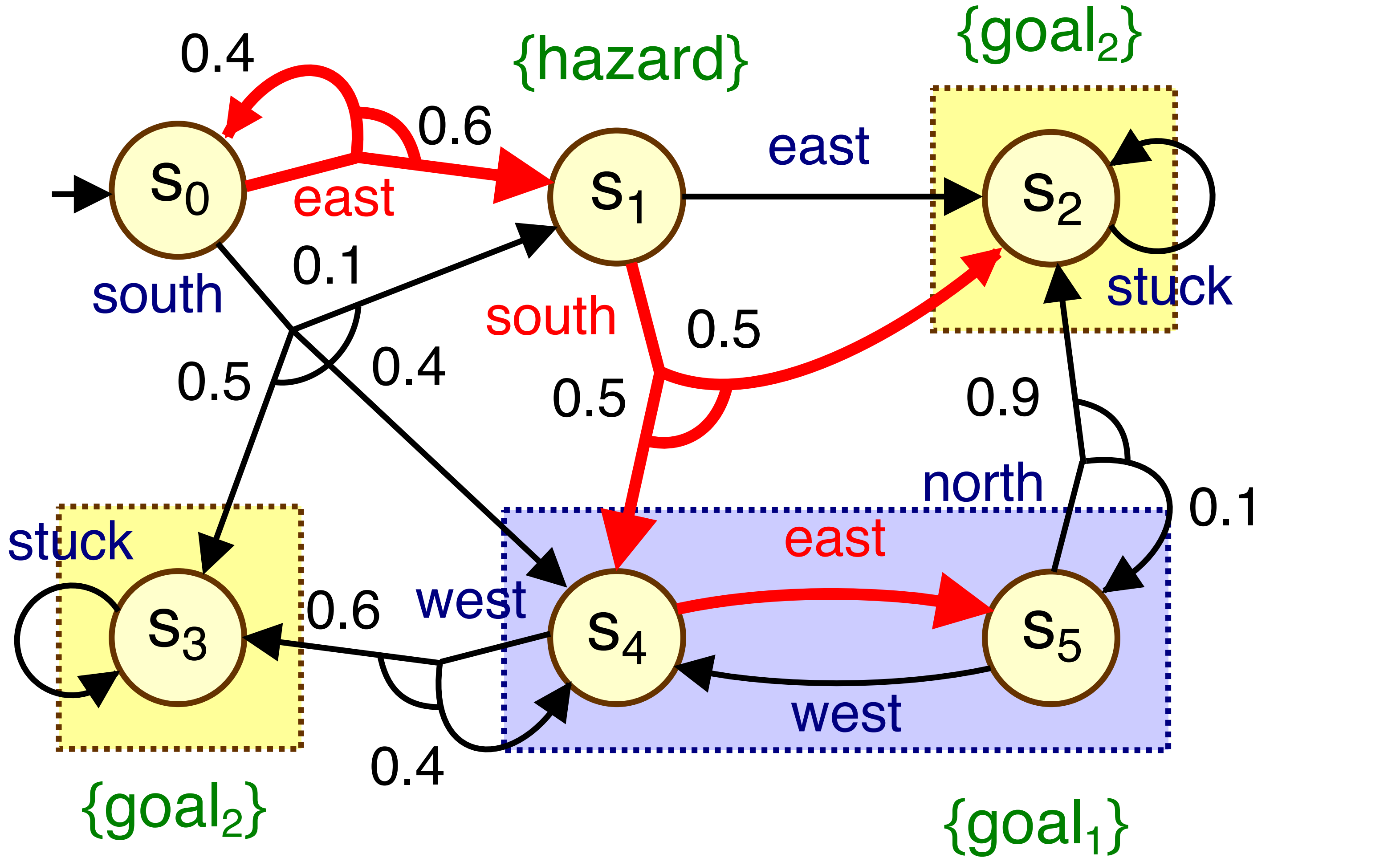
- Finally: $x_0=0.5$, $x_1=0.5$

MaxProb via linear programming

- Optimal values can be computed using linear programming (LP):
  - $V^*(s)$ equals the solution x_s to:

minimise $\Sigma_{s \in S}\, x_s$ subject to the constraints:

$x_s = 1$	for $s \in S^1$
$x_s = 0$	for $s \in S^0$
$x_s \geq \Sigma_{s' \in S}\, P_s^a(s') \cdot x_{s'}$	for $s \in S^?$, $a \in A(s)$

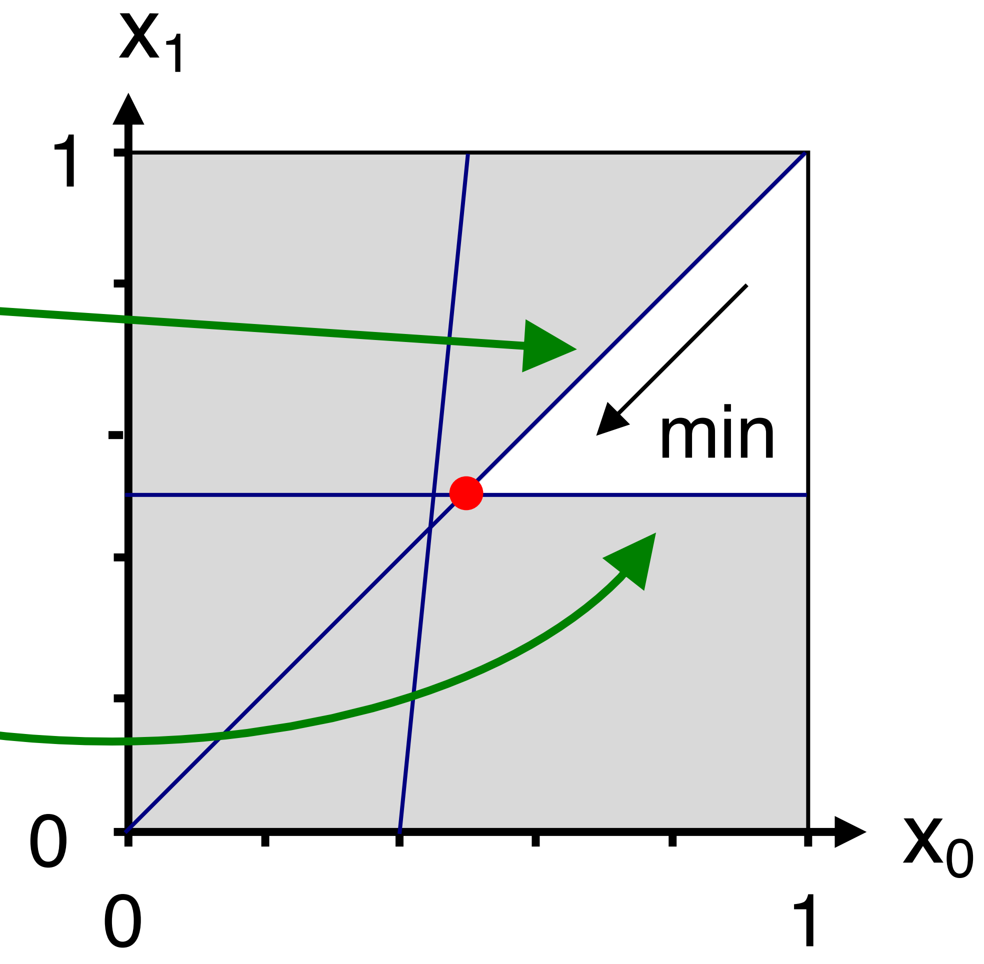


Minimise x_0+x_1 s.t.:
$x_0 \geq 0.4x_0 + 0.6x_1$
$x_0 \geq 0.1x_1 + 0.5x_3 + 0.4x_4$
$x_1 \geq x_2$
$x_1 \geq 0.5x_2 + 0.5x_4$

Minimise x_0+x_1 s.t.:
$x_0 \geq x_1$
$x_0 \geq 0.1x_1 + 0.4$
$x_1 \geq 0.5$

$x_0 \geq x_1$
(s_0:south)

$x_1 \geq 0.5$
(s_1:east)



Solving SSP for MDPs

- Value iteration:

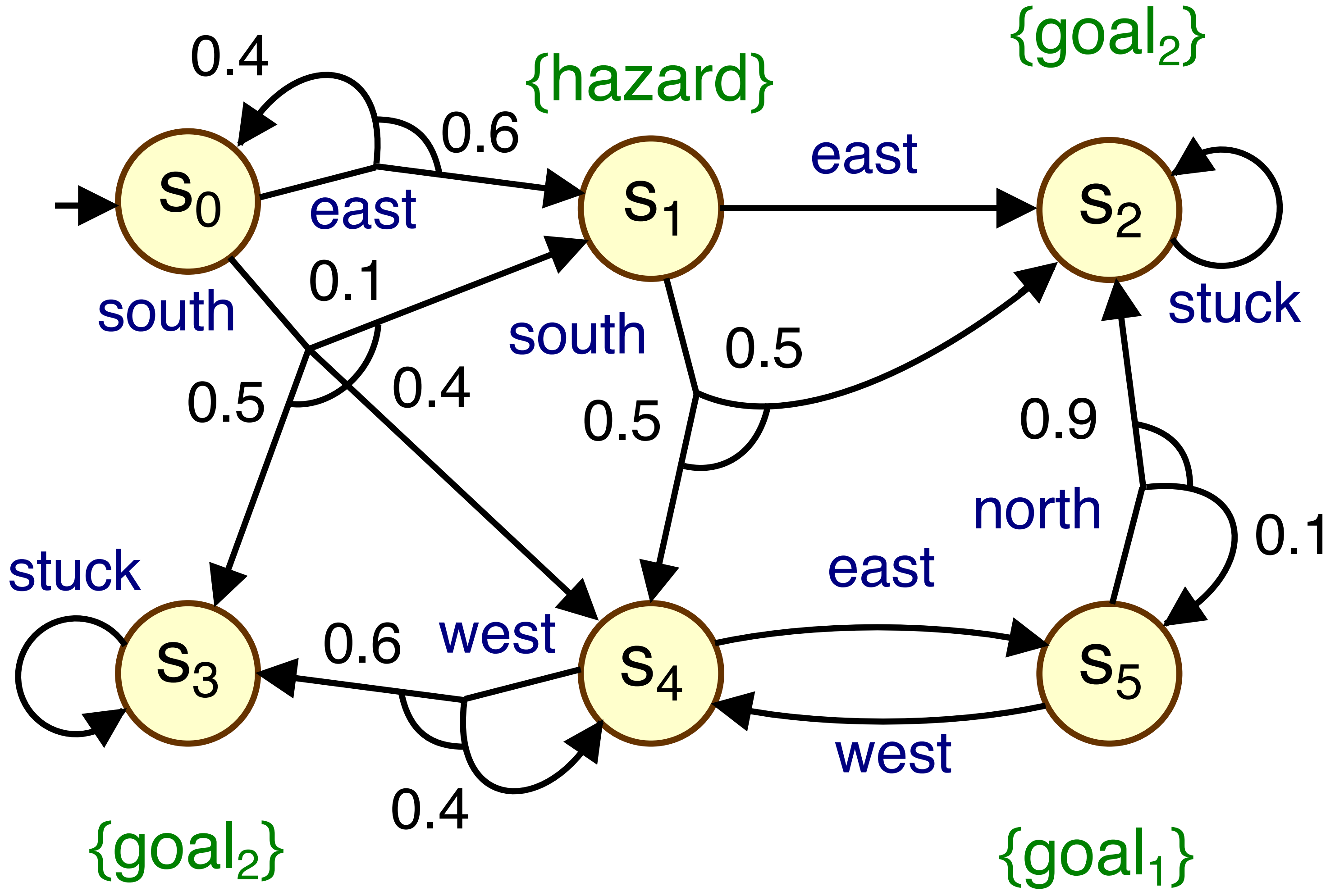
$$x_s^k = \begin{cases} 0 & \text{if } s \in goal \\ \min_{a \in A(s)} \left[C(s,a) + \sum_{s' \in S} P_s^a(s') \cdot x_{s'}^{k-1} \right] & \text{otherwise} \end{cases}$$

- Linear programming

maximise $\Sigma_{s \in S}\, x_s$ subject to the constraints:

$x_s = 0$ for $s \in goal$

$x_s \leq C(s,a) + \Sigma_{s' \in S} P_s^a(s') \cdot x_{s'}$ for $s \in S_?,\ a \in A(s)$



- Pre-computation:
  - we can also use graph-based pre-computation to identify/collapse states and relax SSP assumptions

MDP solution methods

- Solving MaxProb (or SSP) on MDPs (focusing on “exact” algorithms):
- Value iteration (VI)
  - simple, and effective in practice, but care needed with convergence detection
  - complexity unclear (depends on accuracy)
- Linear programming
  - polynomial complexity
  - in principle, can yield exact (arbitrary precision) optimal values; likely scales worse than VI
- Various other algorithms / optimisations
  - Policy iteration, VI + prioritisation, topological partitioning, parallelisation, …
  - Heuristics (e.g., BRTDP), sampling (e.g., Monte Carlo tree search), …

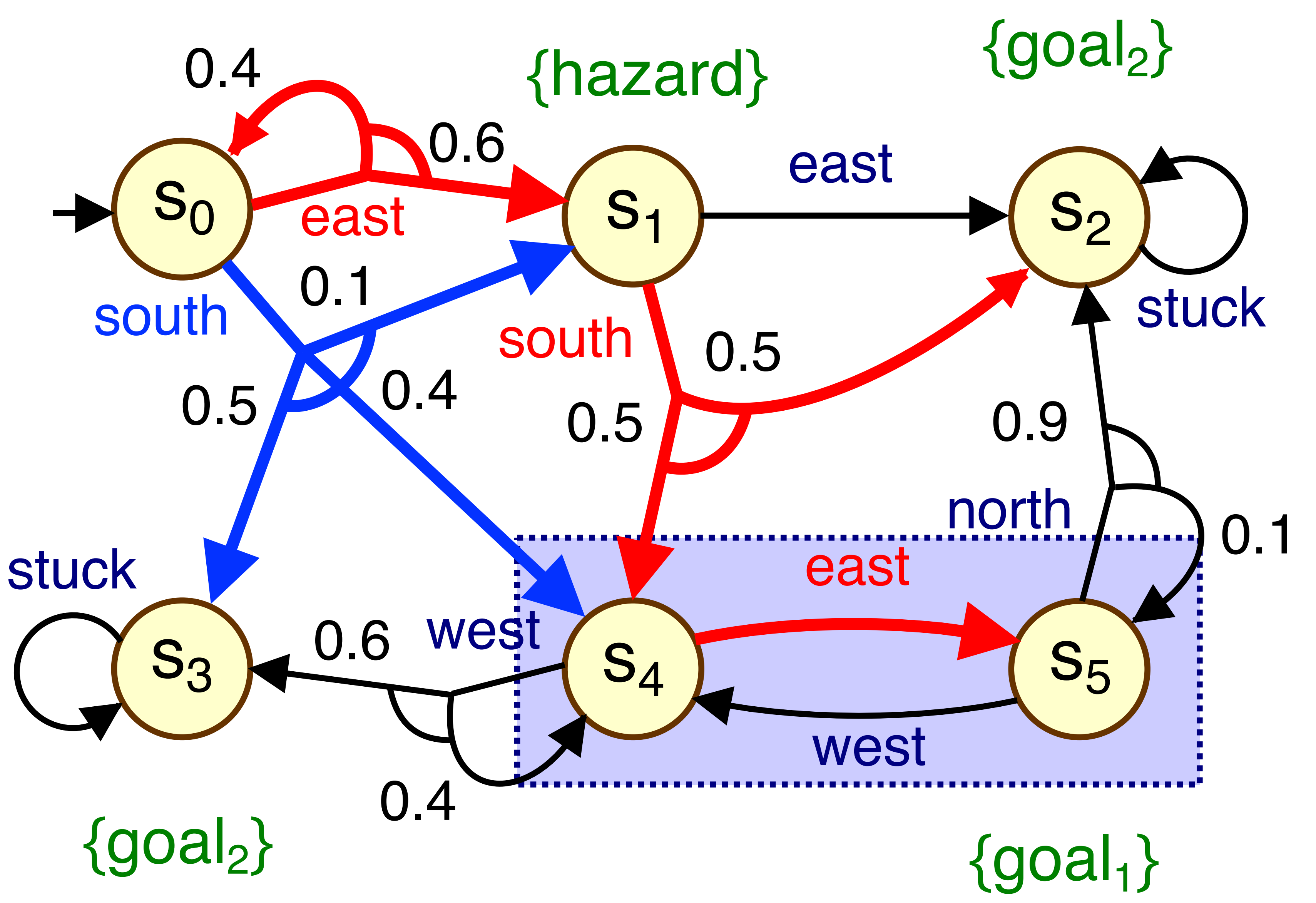
MaxProb over a finite horizon

- Finite-horizon variant solvable with value iteration (without pre-computation)
  - $V^*(s) = x_s^k$ where:

$$x_s^k = \begin{cases} 1 & \text{if } s \in goal \\ 0 & \text{if } s \notin goal \text{ and } n = 0 \\ \max_{a \in A(s)} \sum_{s' \in S} P_s^a(s') \cdot x_{s'}^{k-1} & \text{otherwise} \end{cases}$$

- Running example
  - MaxProb$^{\leq k}$($\{s_4, s_5\}$)
  - optimal policy is not memoryless

k	x_0	x_1
0	0	0
1	0.4	0.5
2	0.46	0.5
3	0.484	0.5

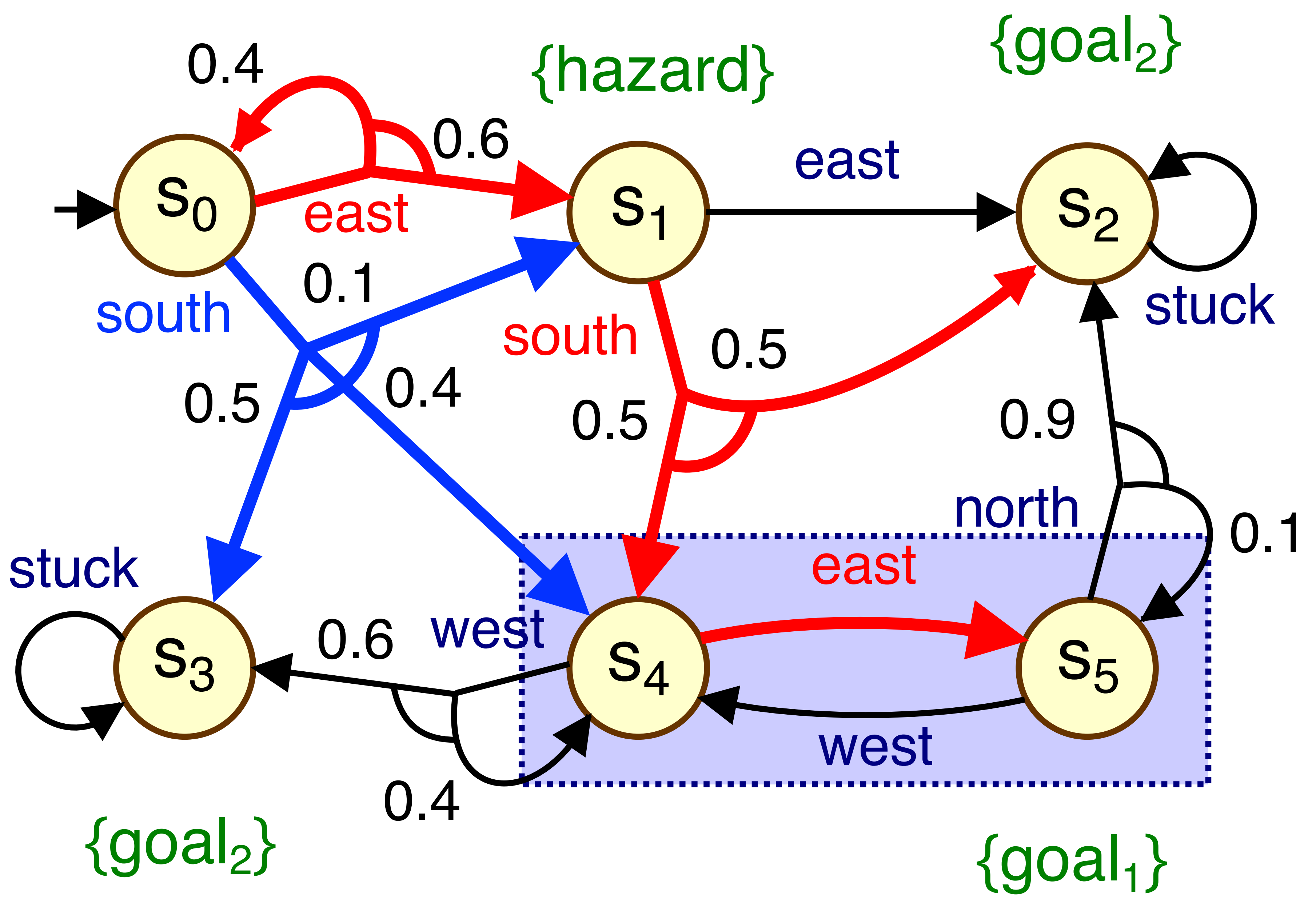


Beyond MDPs

- How do we go beyond the assumptions made so far?
- Full observability (of state, costs, …)
  - partially observable MDPs, beliefs over hidden state
- Finite state spaces, action spaces
  - continuous state/action, dynamic systems
- Full knowledge of the model
  - epistemic uncertainty, also sampling-based models
- Fully controllable model
  - adversarial (or collaborative) scenarios: stochastic game models

Summary (lecture 1)

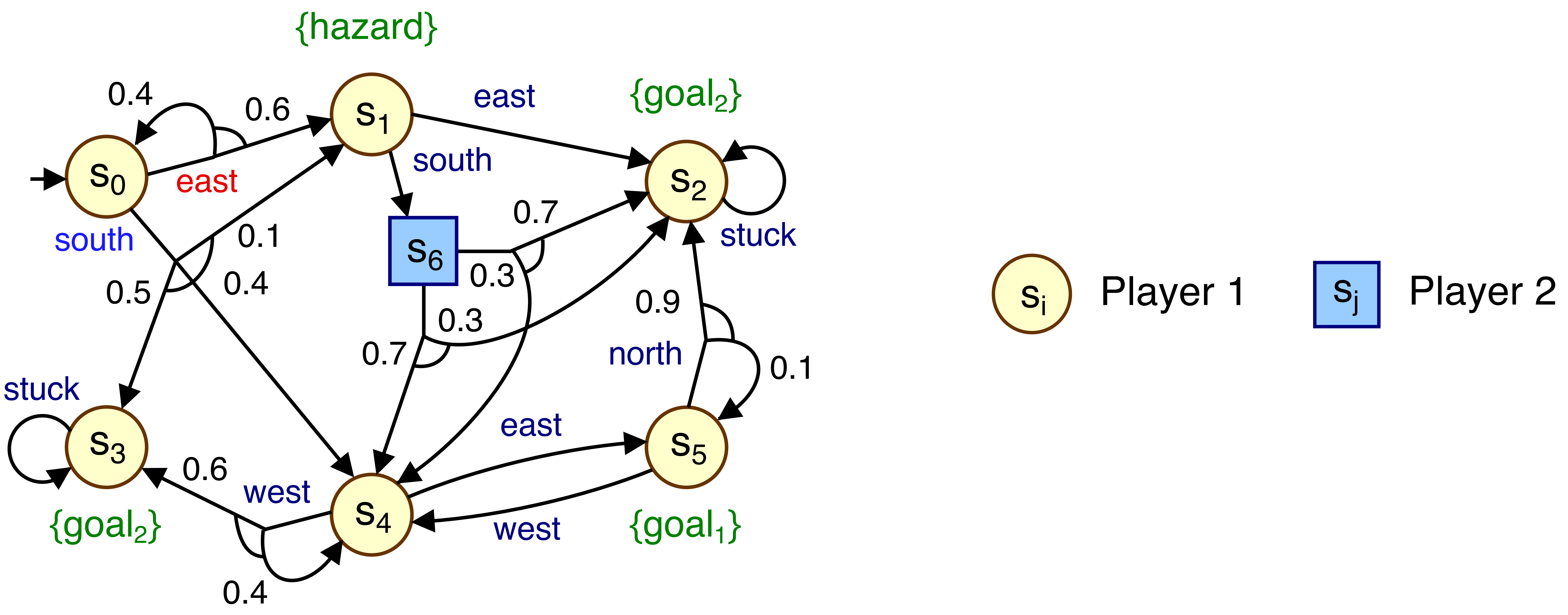
- Introduction
  - aleatoric vs. epistemic uncertainty
- Markov decision processes (MDPs)
  - sequential decision making under uncertainty
  - policies and objectives
    - MaxProb, SSP, finite-horizon, temporal logic
  - solving MDPs (optimal policy generation)
    - linear programming (PTIME)
    - or dynamic programming (value iteration)



Stochastic games

Running example

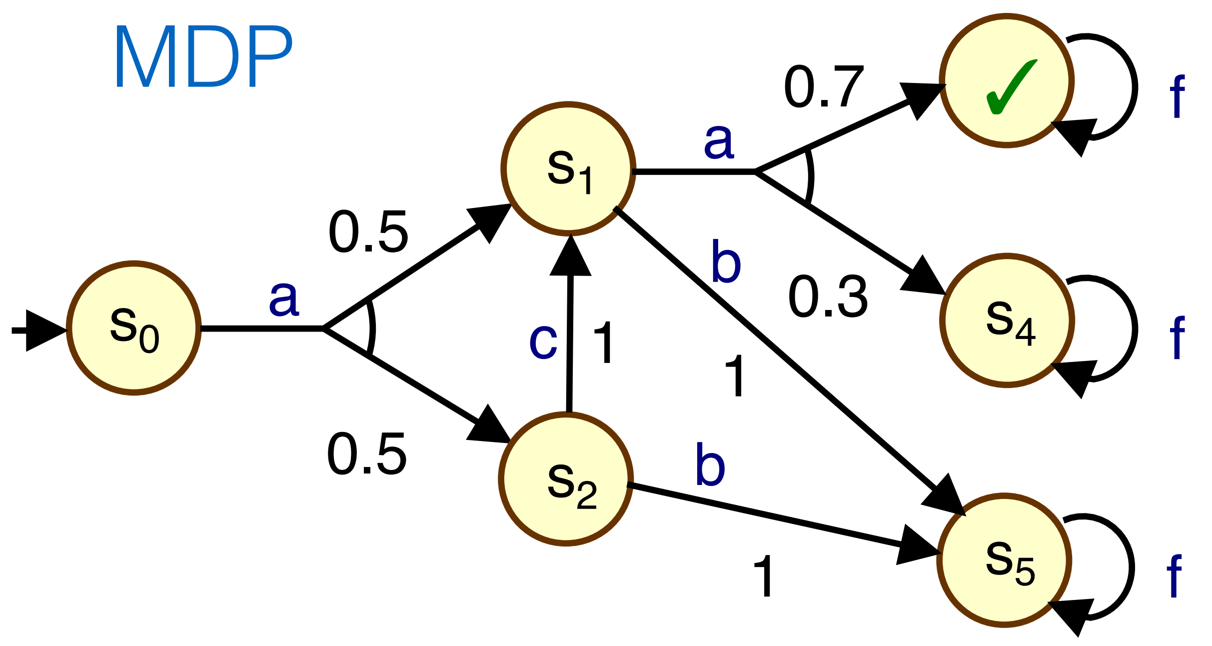
- Interaction with a second robot



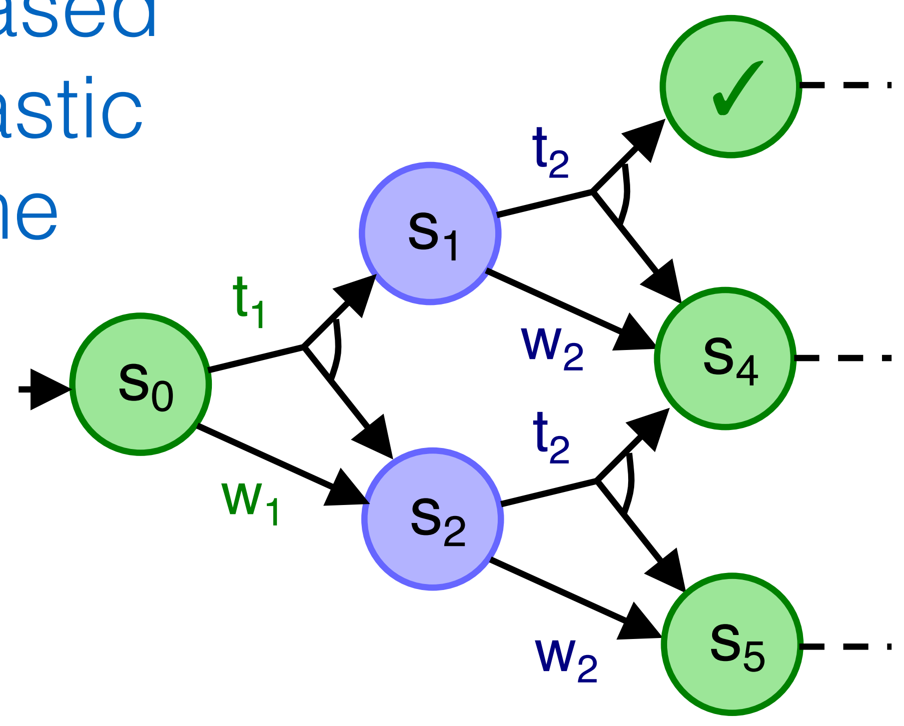
Stochastic games

- MDPs model sequential decision making
  - for a single agent, under stochastic uncertainty
  - we may need adversarial (uncontrollable) decisions
  - or collaborative decision making for multiple agents
- A (turn-based, two-player) stochastic game
  - takes the form $\mathcal{G} = (\{1,2\}, S, \langle S_1, S_2 \rangle, s_0, A, P)$ where:
  - states S, initial state s_0 and actions A are as for MDPs
  - $S_1, S_2 \subseteq S$ are the (disjoint) states controlled by players 1 and 2
  - transition function $P : S \times A \times S \rightarrow [0,1]$ is also as for MDPs
- Another possibility: concurrent stochastic games
  - with $P : S \times (A_1 \times A_2) \times S \rightarrow [0,1]$

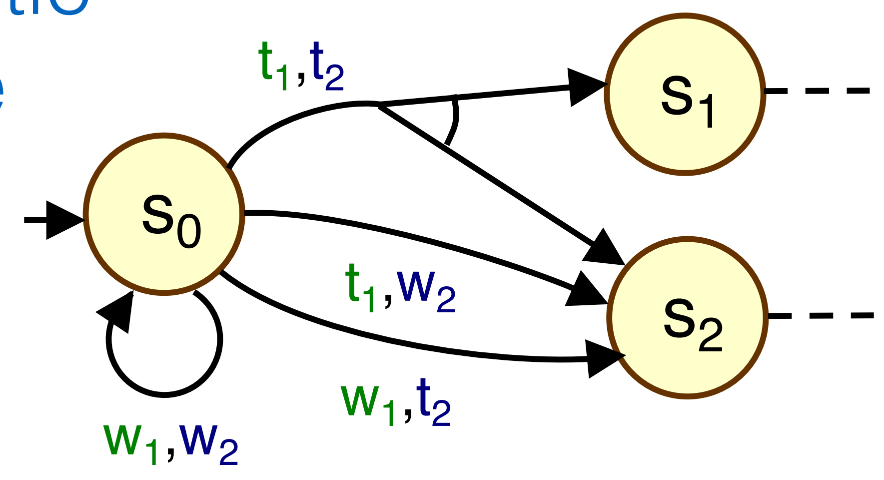
MDP



turn-based stochastic game

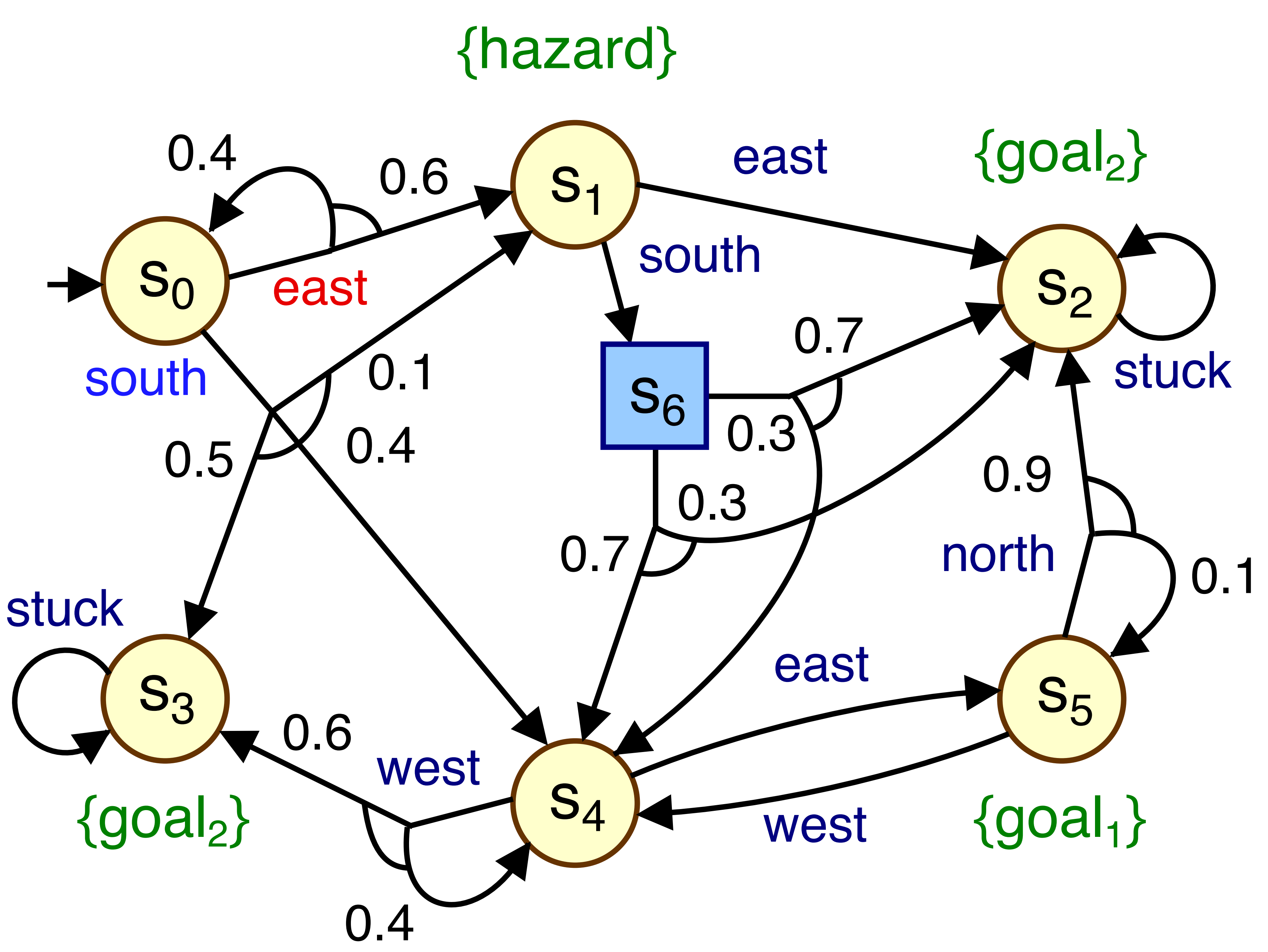


concurrent stochastic game

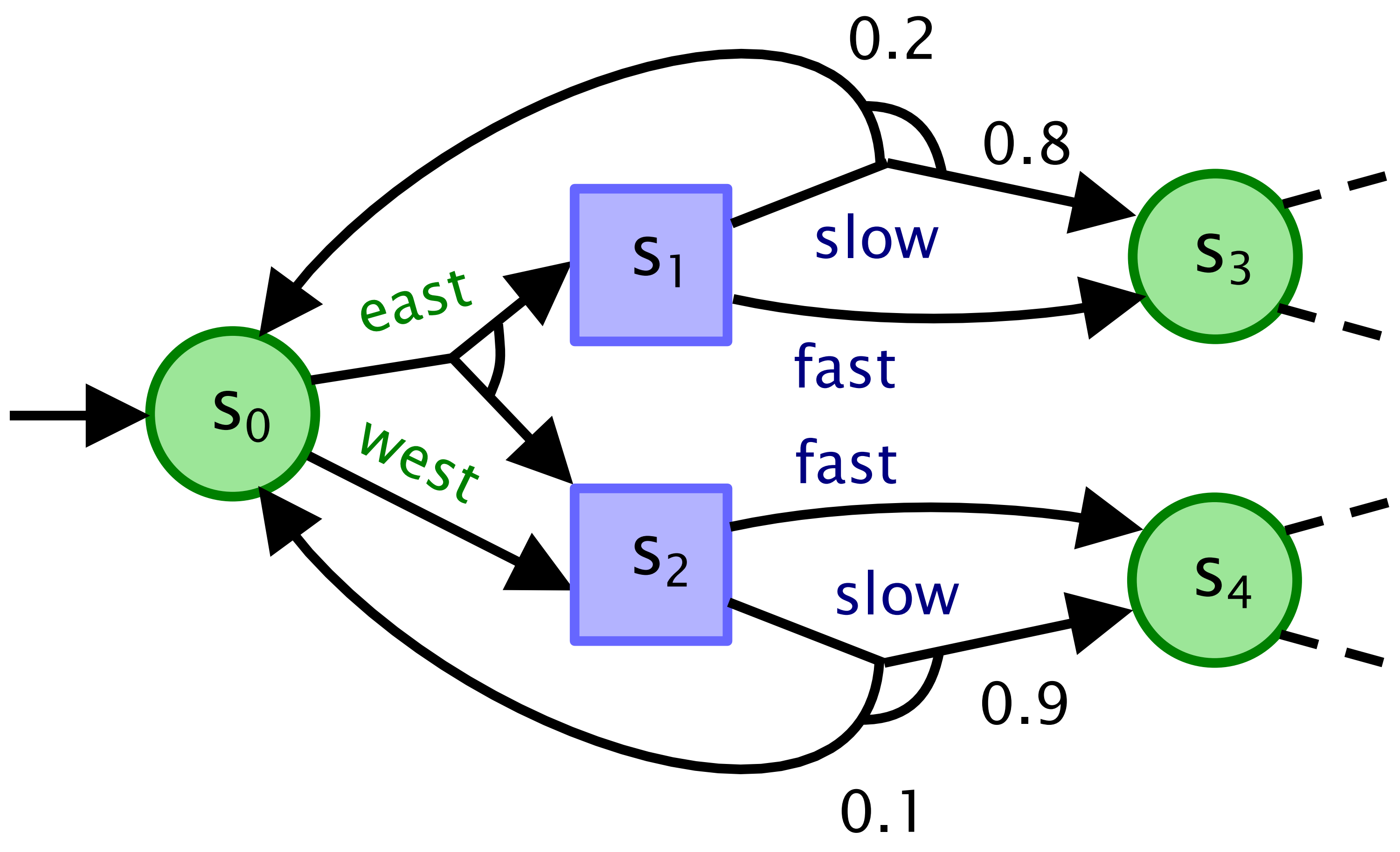


Turn-based stochastic games

uncontrollable/unknown interference

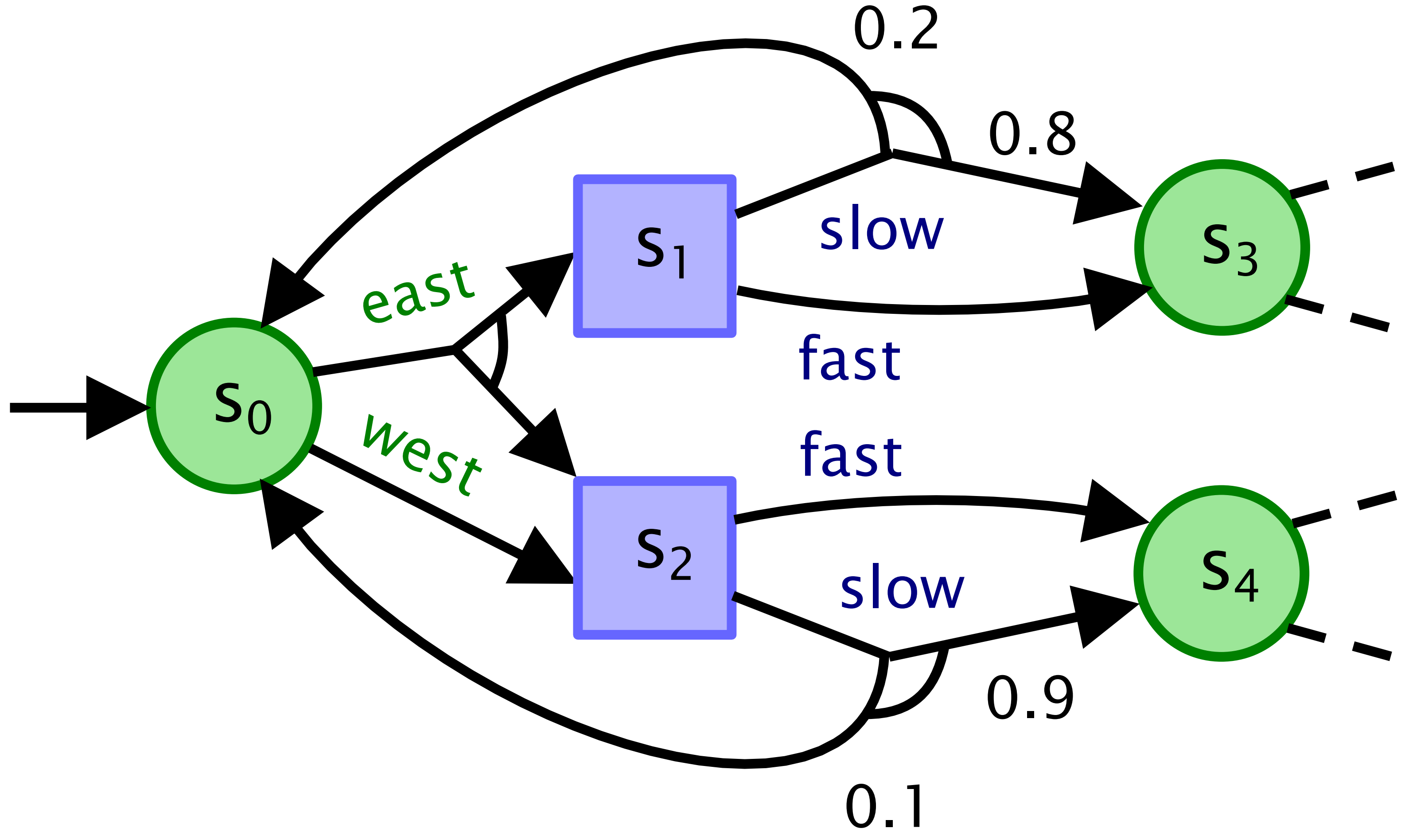


shared autonomy:
human-robot control



Strategies for stochastic games

- Strategies (policies) for turn-based stochastic games
  - a strategy for player i is a mapping $\pi_i : (S \times A)^* \times S_i \rightarrow Dist(A)$
  - a strategy profile (π_1, π_2) defines strategies for both players
- For state s of game $\mathcal{G}$ and strategy profile (π_1, π_2):
  - we can define probability space $Pr_s^{\pi_1,\pi_2}$, random variables $\mathbb{E}_s^{\pi_1,\pi_2}(X)$ and value functions $V^{\pi_1,\pi_2}(s)$



- Strategies
  - can again be deterministic / randomised or memoryless / history-dependent
  - Π_i is the set of all strategies for player $i \in \{1,2\}$

Objectives for stochastic games

- Objectives V_1, V_2 for players 1 and 2 can be distinct
  - simple, useful scenario: zero-sum (directly opposing), i.e., $V_1 = -V_2$
  - so we assume a single objective V which one player maximises and the other minimises
- Consider MaxProb for player 1 (other cases are similar):

  $\max_{\pi_1 \in \Pi_1} \min_{\pi_2 \in \Pi_2} V^{\pi_1,\pi_2}(s)$ where V^{π_1,π_2} is exactly as for MDP MaxProb

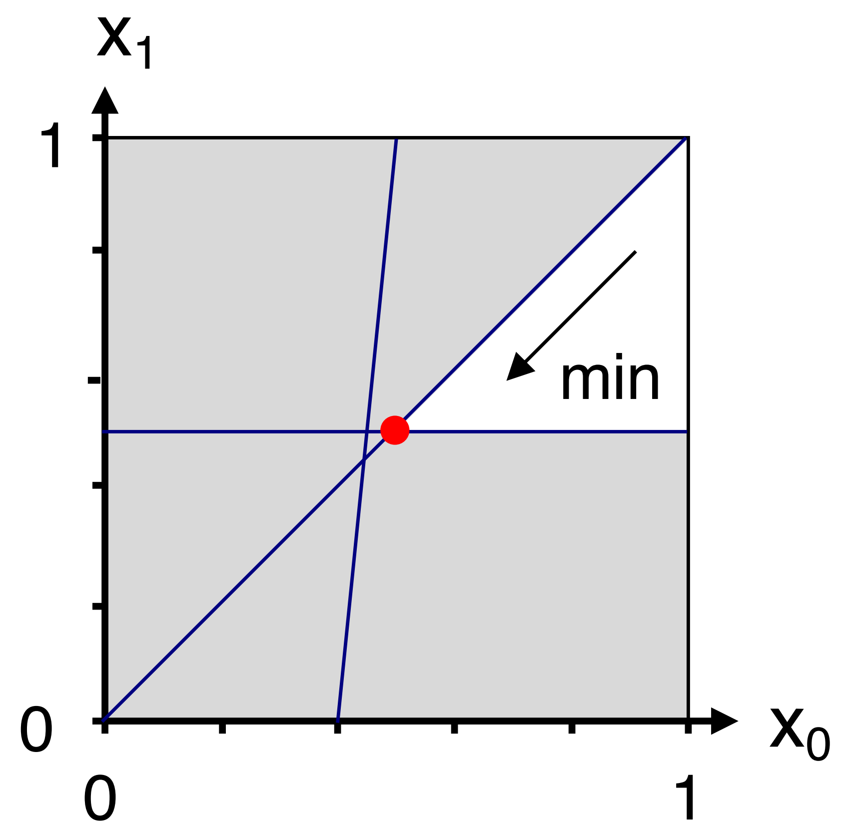
- Games are determined, i.e., for all states s:

$$\max_{\pi_1 \in \Pi_1} \min_{\pi_2 \in \Pi_2} V^{\pi_1,\pi_2}(s) = \min_{\pi_2 \in \Pi_2} \max_{\pi_1 \in \Pi_1} V^{\pi_1,\pi_2}(s)$$

- So we define:
  - optimal value: $V^*(s) = \max_{\pi_1 \in \Pi_1} \min_{\pi_2 \in \Pi_2} V^{\pi_1,\pi_2}(s)$
  - optimal strategy (for player 1): $\pi^* = \text{argmax}_{\pi_1 \in \Pi_1} \min_{\pi_2 \in \Pi_2} V^{\pi_1,\pi_2}(s_0)$

Solving stochastic games

- Memoryless deterministic strategies suffice (for both players)

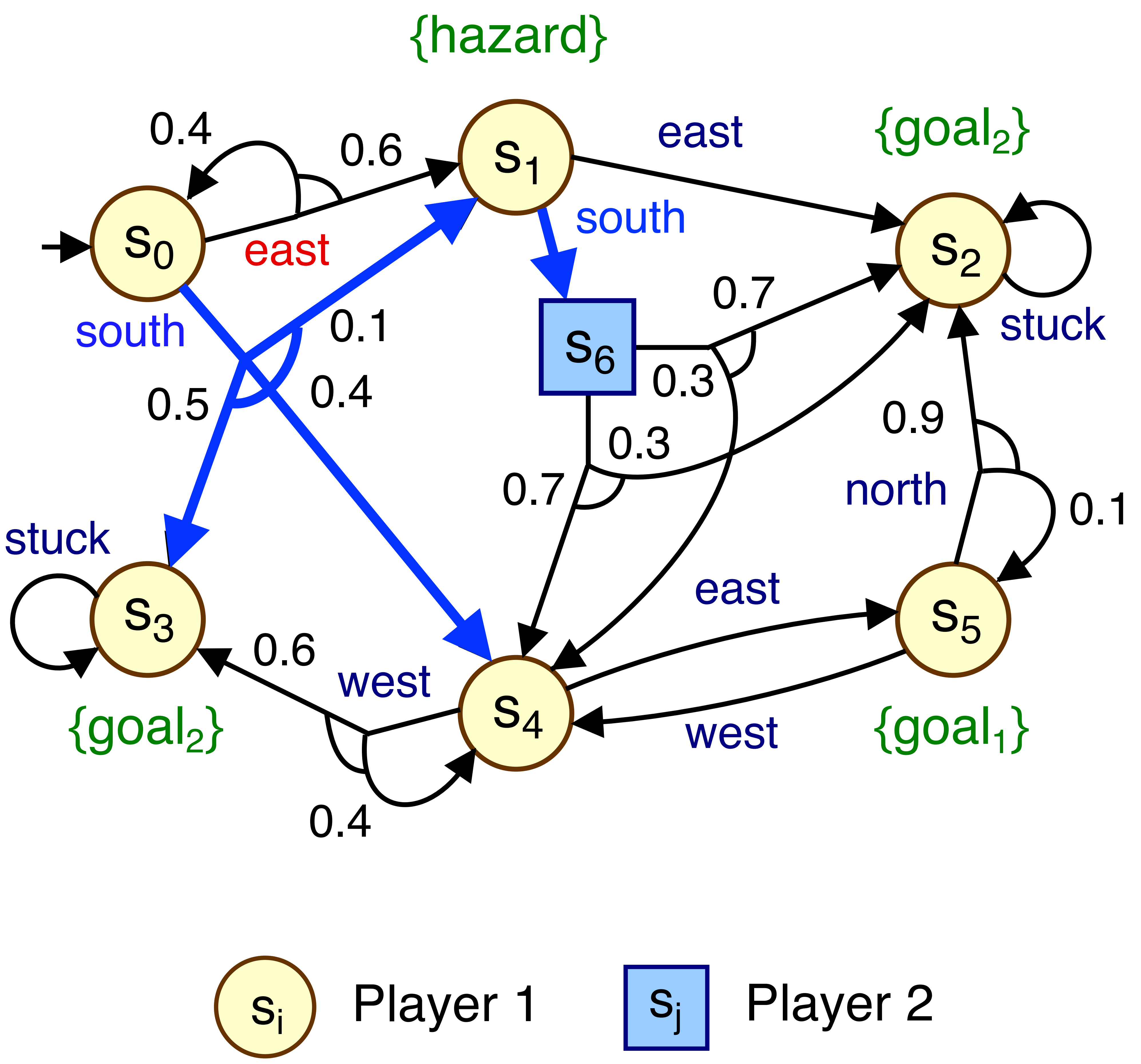


- Complexity worse than for MDPs: NP ∩ co-NP, rather than P
  - LP approach does not adapt (but strategy improvement is possible)
- In practice: dynamic programming (value iteration) works well
  - e.g., for MaxProb:

$$x_s^k = \begin{cases} 1 & \text{if } s \in goal \\ 0 & \text{if } s \notin goal \text{ and } k = 0 \\ \max_{a \in A(s)} \sum_{s' \in S} P_s^a(s') \cdot x_{s'}^{k-1} & \text{if } s \notin goal, s \in S_1 \text{ and } k > 0 \\ \min_{a \in A(s)} \sum_{s' \in S} P_s^a(s') \cdot x_{s'}^{k-1} & \text{if } s \notin goal, s \in S_2 \text{ and } k > 0 \end{cases}$$

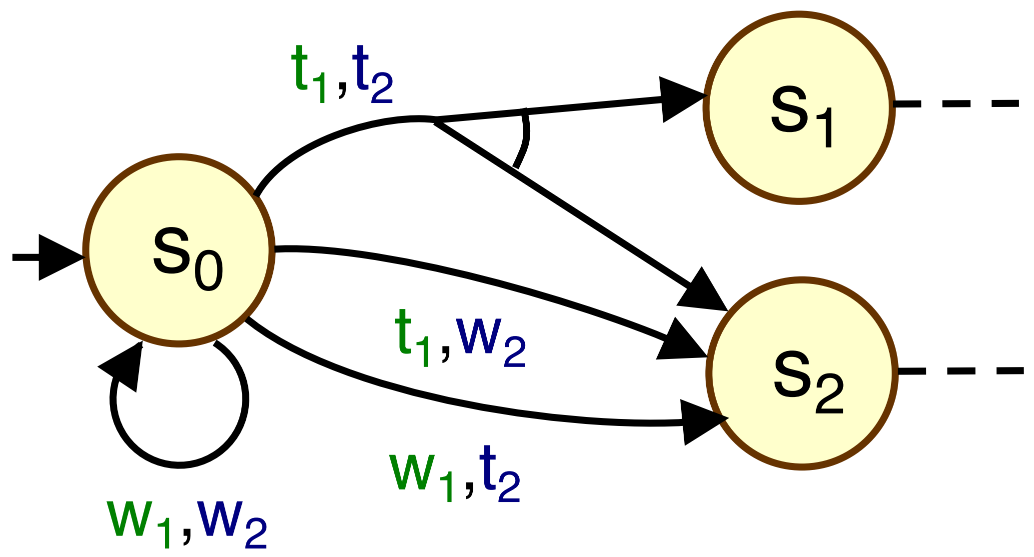
Running example

- Optimal player 1 strategy changes:



Zero-sum concurrent stochastic games

- Concurrent stochastic games: strategies, value functions defined similarly
  - games are still determined: $\max_{\pi_1 \in \Pi_1} \min_{\pi_2 \in \Pi_2} V^{\pi_1,\pi_2}(s) = \min_{\pi_2 \in \Pi_2} \max_{\pi_1 \in \Pi_1} V^{\pi_1,\pi_2}(s)$
  - but optimal strategies still memoryless but now randomised



- Value iteration can be extended: $x_s^k = \begin{cases} 1 & \text{if } s \in goal \\ 0 & \text{if } s \notin goal \text{ and } k = 0 \\ val(Z) & \text{otherwise} \end{cases}$
  - where $val(Z)$ is the value of the matrix game with payoffs: $z_{a,b} = \sum_{s' \in S} P_s^{a,b}(s') \cdot x_{s'}^{k-1}$
  - solved via the linear program →
  - p_a gives the probability of player 1 picking action a in its optimal strategy

Maximise game value v subject to:

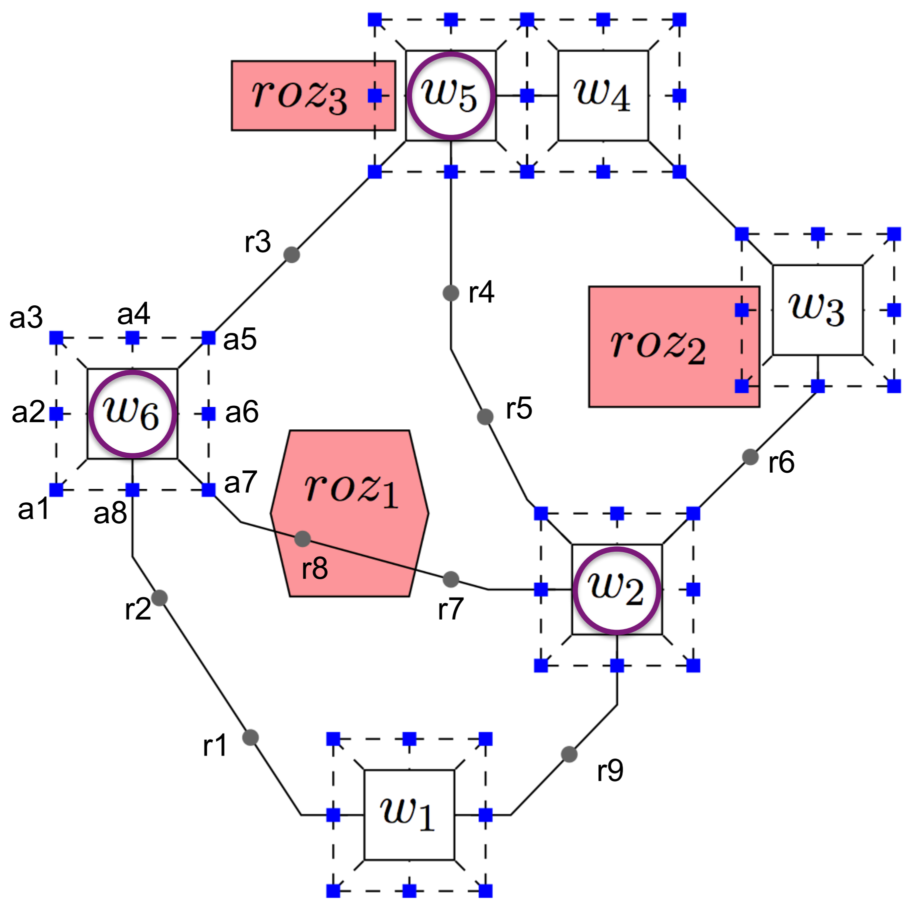
$\Sigma_{a \in A_1} p_a \cdot z_{a,b} \geq v$ for $b \in A_2$

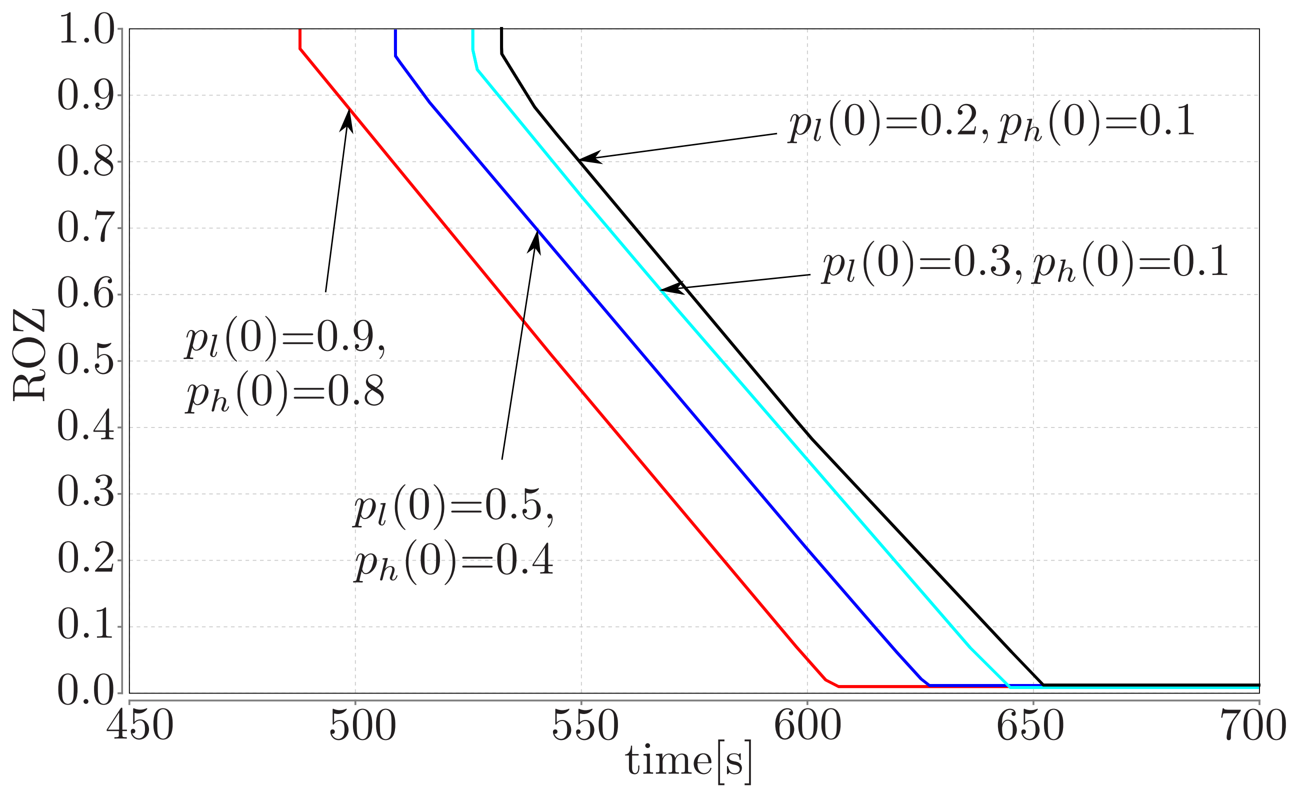
$p_a \geq 0$ for $a \in A_1$

$\Sigma_{a \in A_1} p_a = 1$

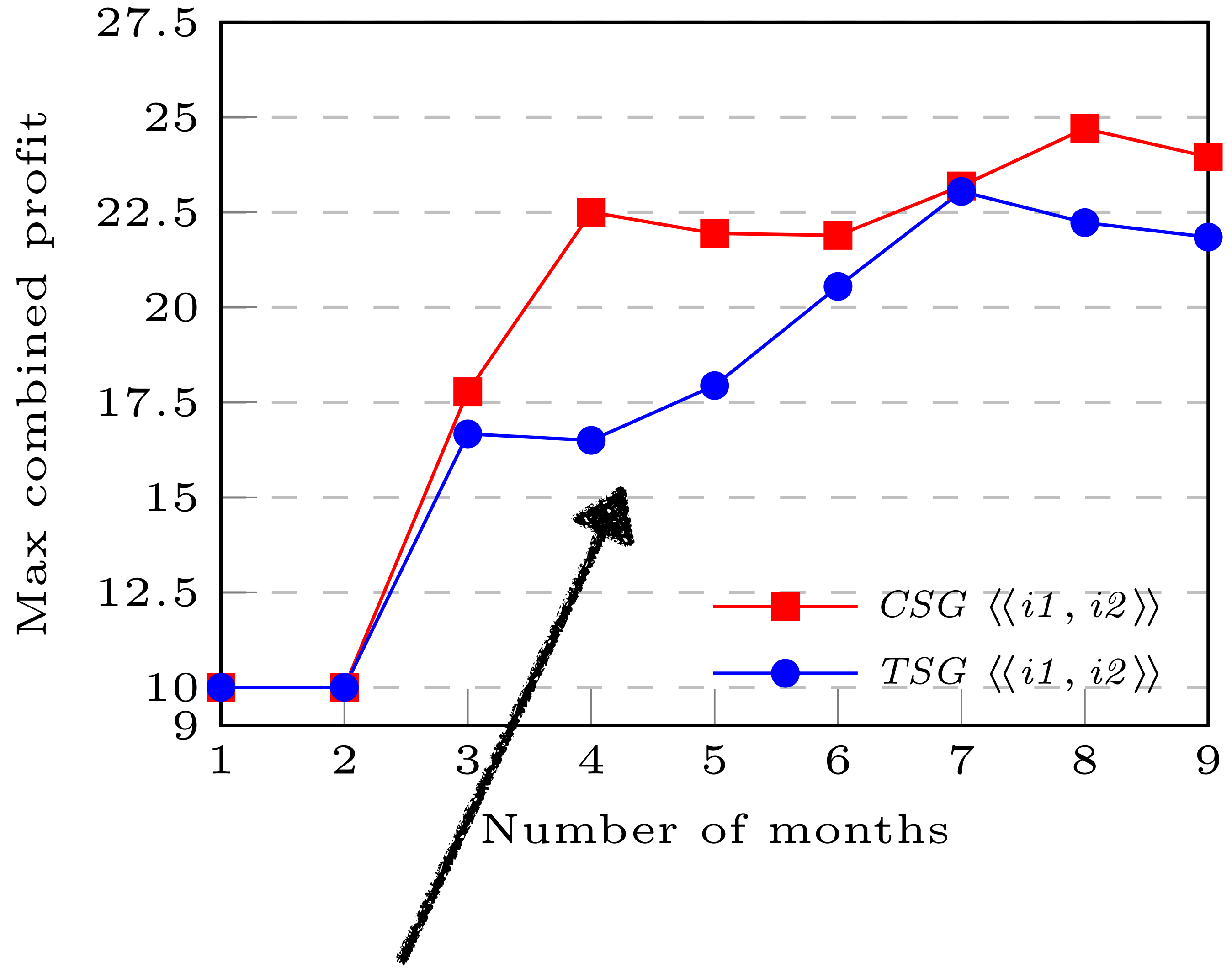
Sequential decision making with stochastic games

- UAV road surveillance
  - with partial human control (varying operator accuracy)





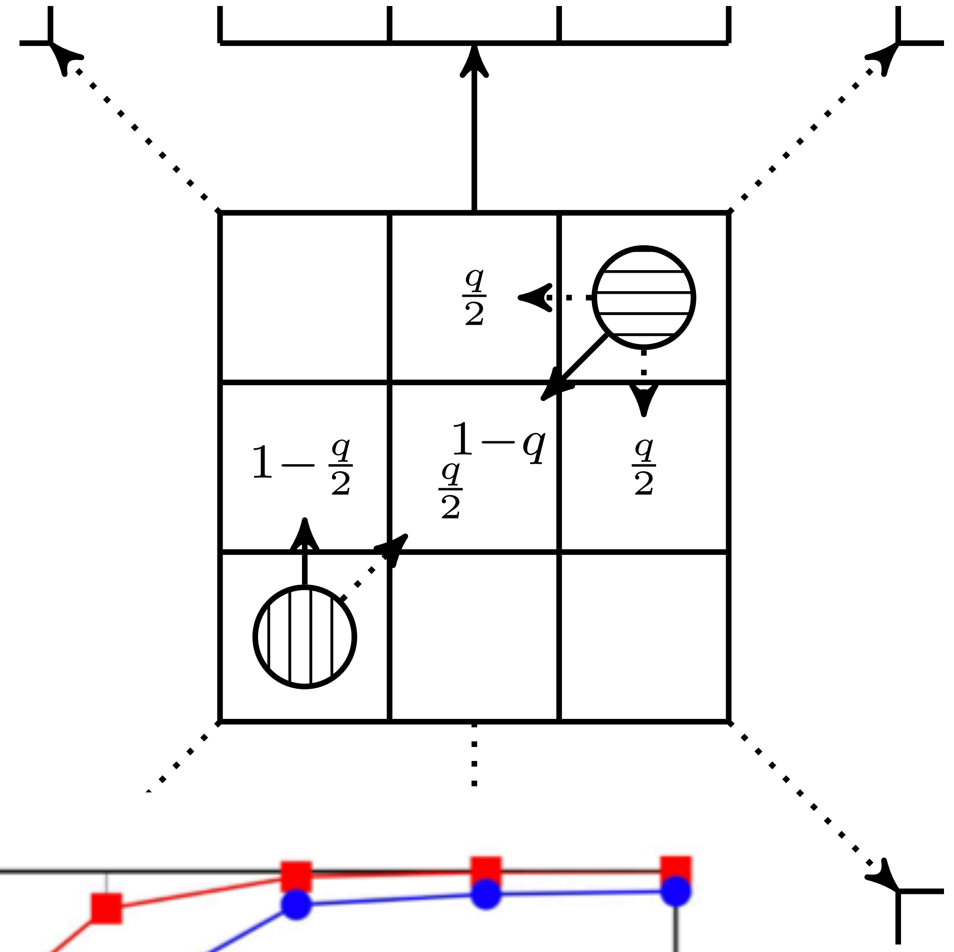
- Futures market investment
  - market is part stochastic, part adversarial

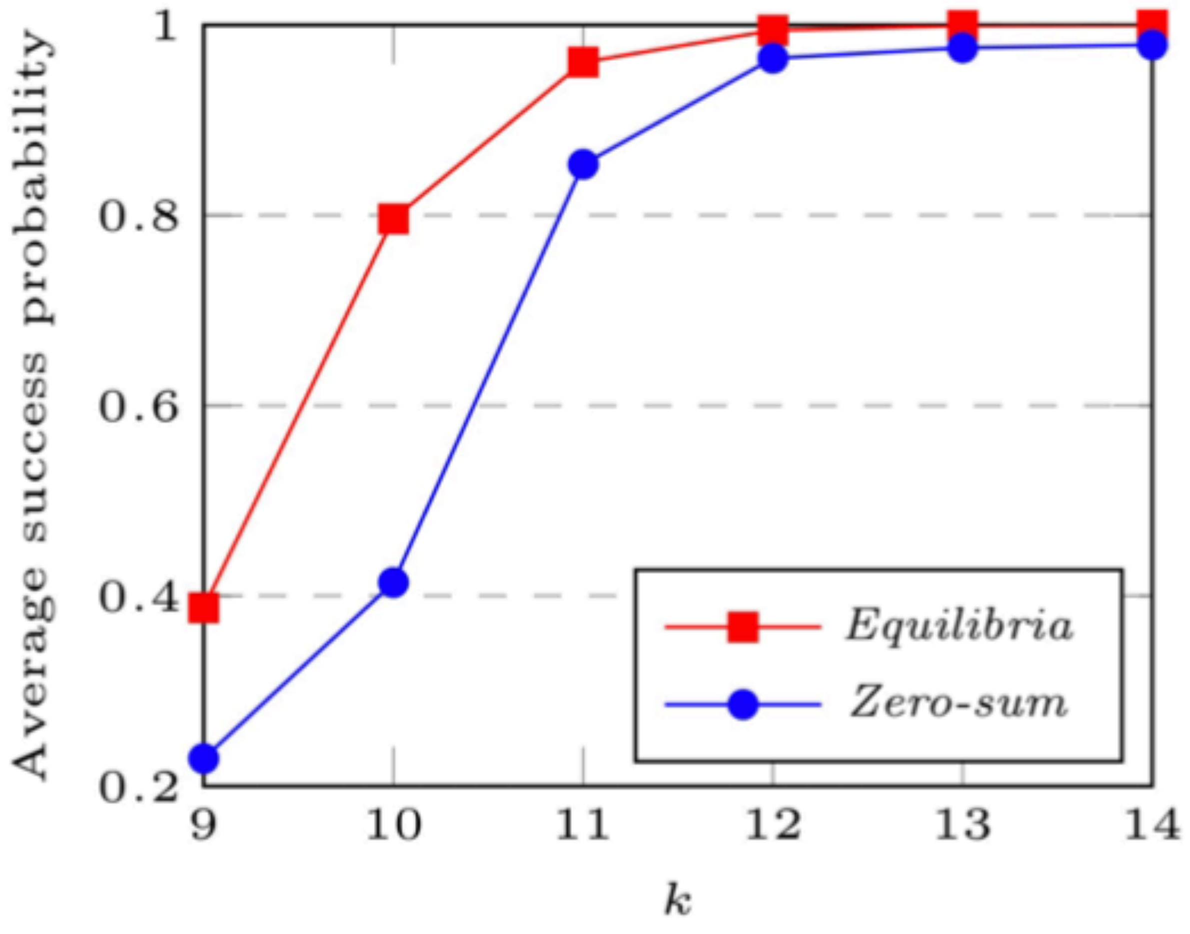


Turn-based game too pessimistic (unrealistic adversary)

PRISM-games

- Multi-robot control
  - adversarial (worst-case) vs. collaborative

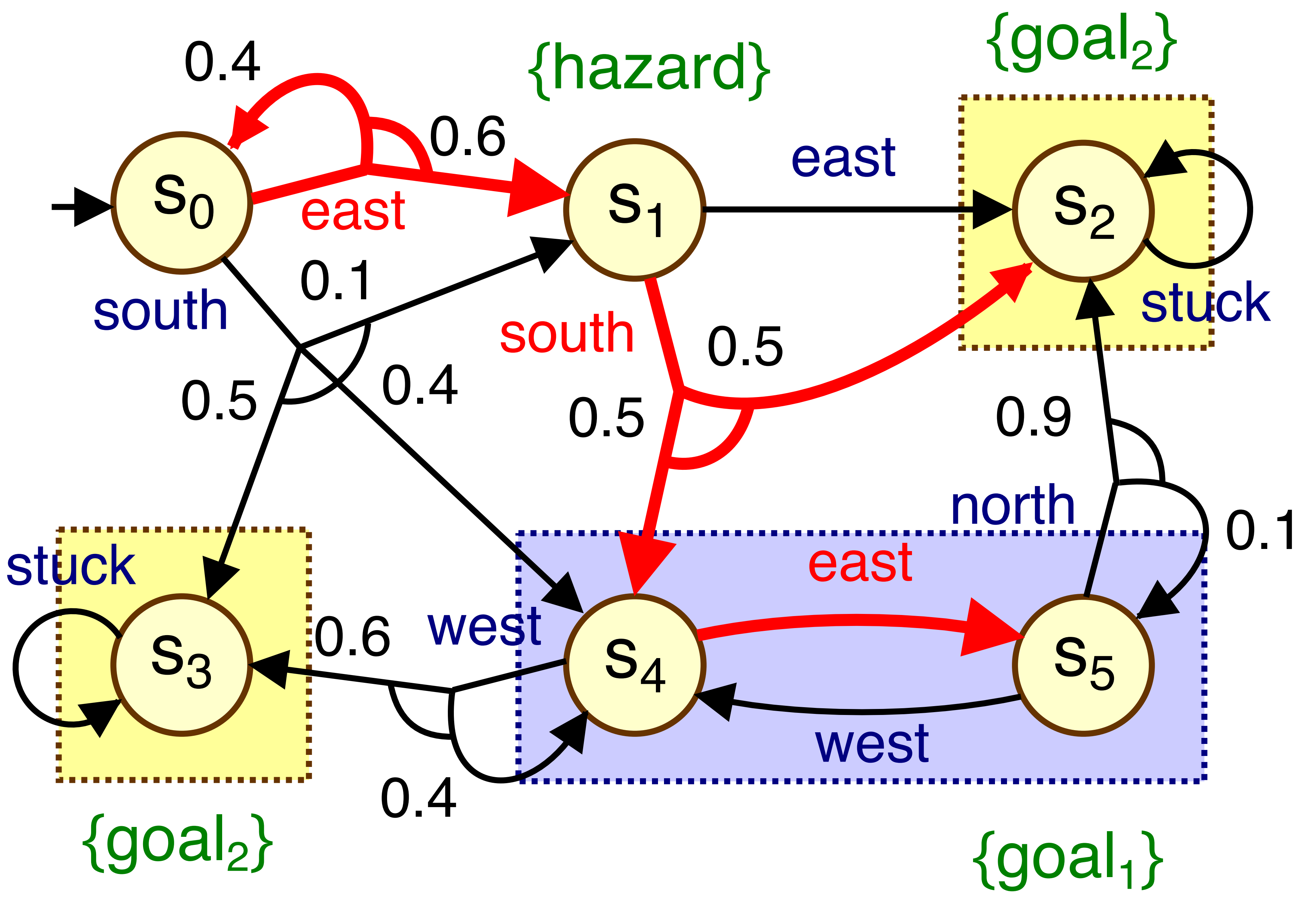




Uncertain MDPs

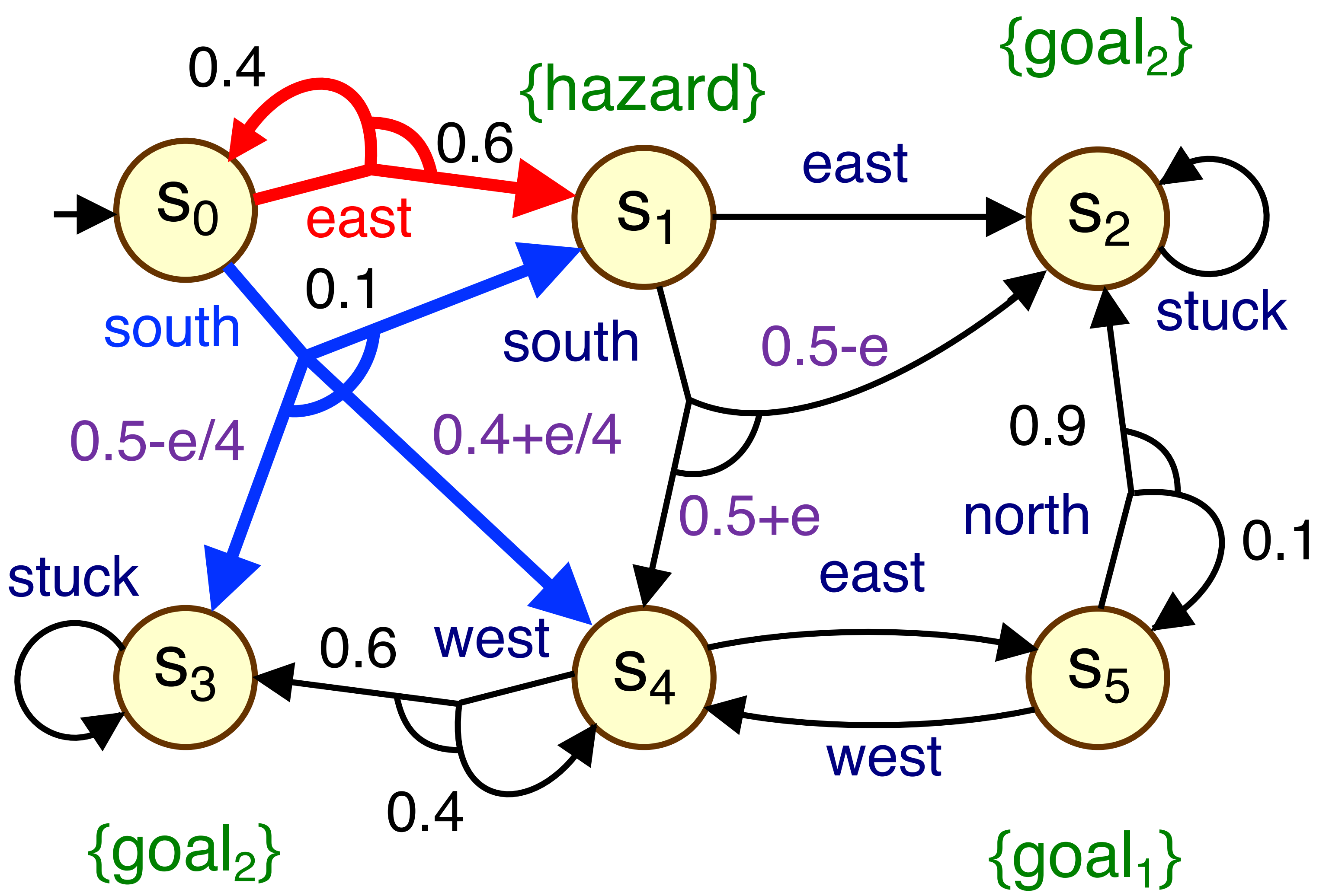
MDPs + epistemic uncertainty

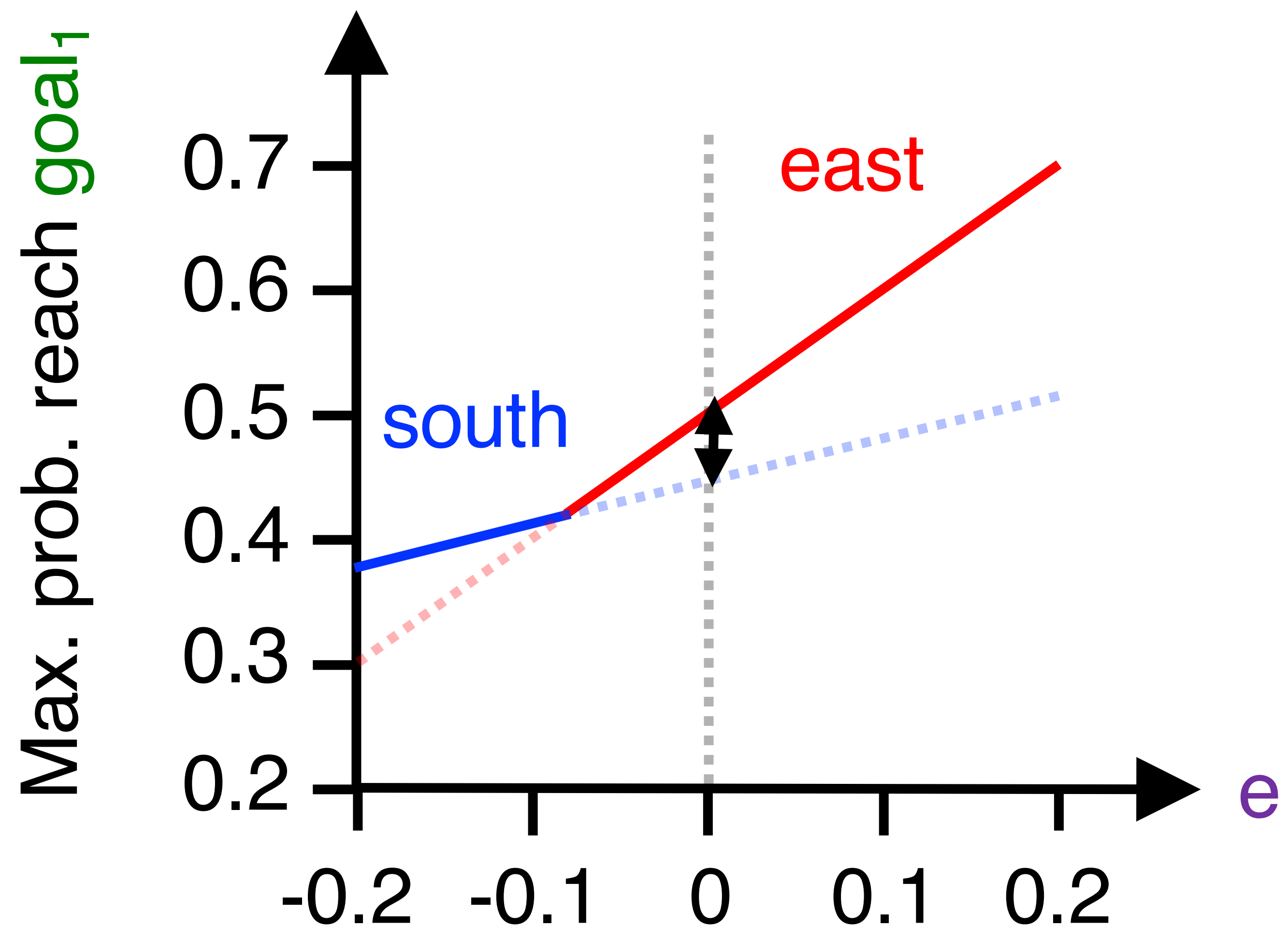
- We can use MDPs for sequential decision making under (aleatoric) uncertainty
  - modelled here using transition probabilities (often learnt from data)



MDPs + epistemic uncertainty

- We can use MDPs for sequential decision making under (aleatoric) uncertainty
  - modelled here using transition probabilities (often learnt from data)
- Policies can be sensitive to small perturbations in transition probabilities
  - so “optimal” policies can in fact be sub-optimal



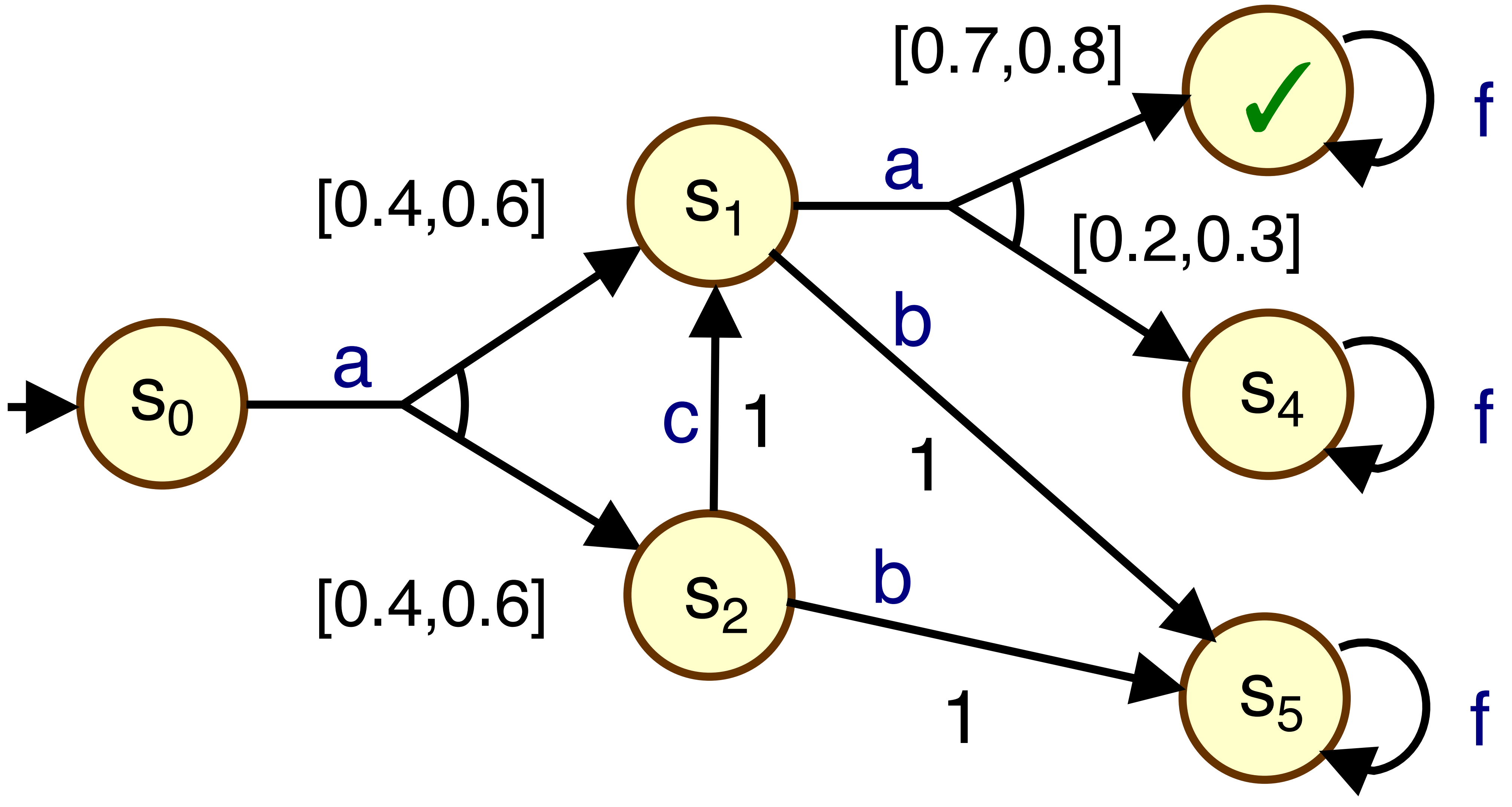


MDPs + epistemic uncertainty

- We can use MDPs for sequential decision making under (aleatoric) uncertainty
  - modelled here using transition probabilities (often learnt from data)
- Policies can be sensitive to small perturbations in transition probabilities
  - so “optimal” policies can in fact be sub-optimal
- Uncertain MDPs: MDPs + epistemic uncertainty (model uncertainty)
  - we focus here on uncertainty in transition probabilities
- Key questions:
  - how to model (and solve for) epistemic uncertainty?
  - what guarantees do we get?
  - is it statistically accurate?
  - how computationally efficient is it?

Uncertain MDPs

- An uncertain MDP (uMDP) takes the form $\mathscr{M} = (S, s_0, A, \mathscr{P})$ where:
  - states S, initial state s_0 and actions A are as for MDPs
  - $\mathscr{P}$ is the transition function uncertainty set
    - i.e., each $P \in \mathscr{P}$ is a transition function $P : S \times A \times S \rightarrow [0,1]$



- The uncertainty set $\mathscr{P}^a_s \subseteq Dist(S)$
  - for each $s \in S$, $a \in A(s)$
  - is $\mathscr{P}^a_s = \{P^a_s \;:\; P \in \mathscr{P}\}$
  - similarly: $\mathscr{P}^a = \{P^a \;:\; P \in \mathscr{P}\}$
  - ($\mathscr{P}^a_s$ sometimes "ambiguity sets")

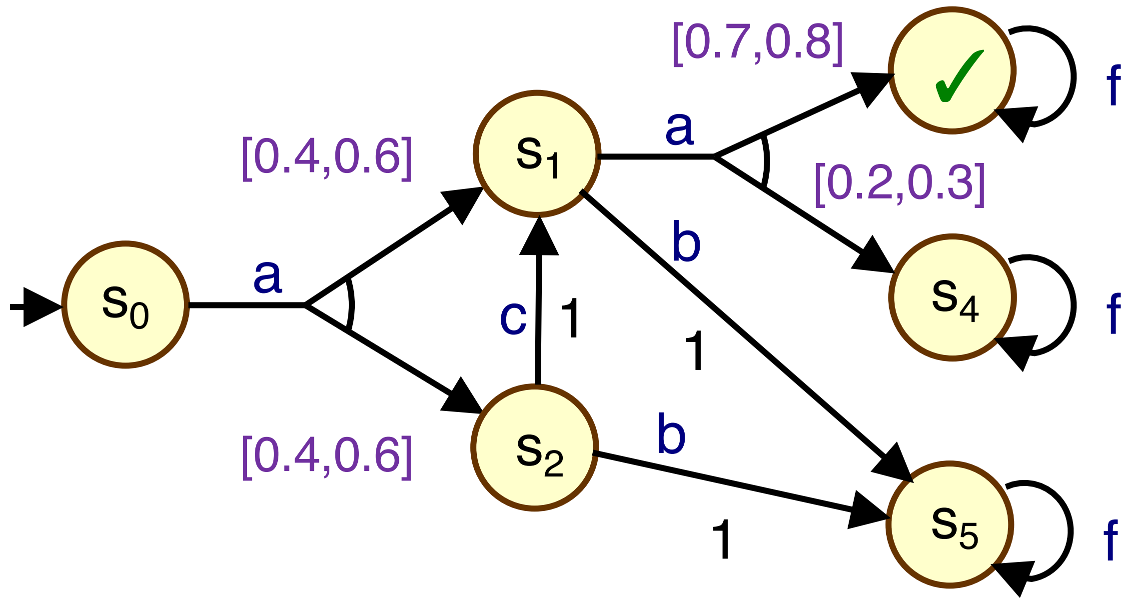
Uncertain MDPs

- Semantics of a uMDP $\mathscr{M} = (S, s_0, A, \mathscr{P})$
  - $\mathscr{M}$ can be seen as a (usually infinite) set of MDPs: $[\![\mathscr{M}]\!] = \{\mathscr{M}[P] : P \in \mathscr{P}\}$
  - where $\mathscr{M}[P] = (S, s_0, A, P)$ is $\mathscr{M}$ instantiated with $P \in \mathscr{P}$

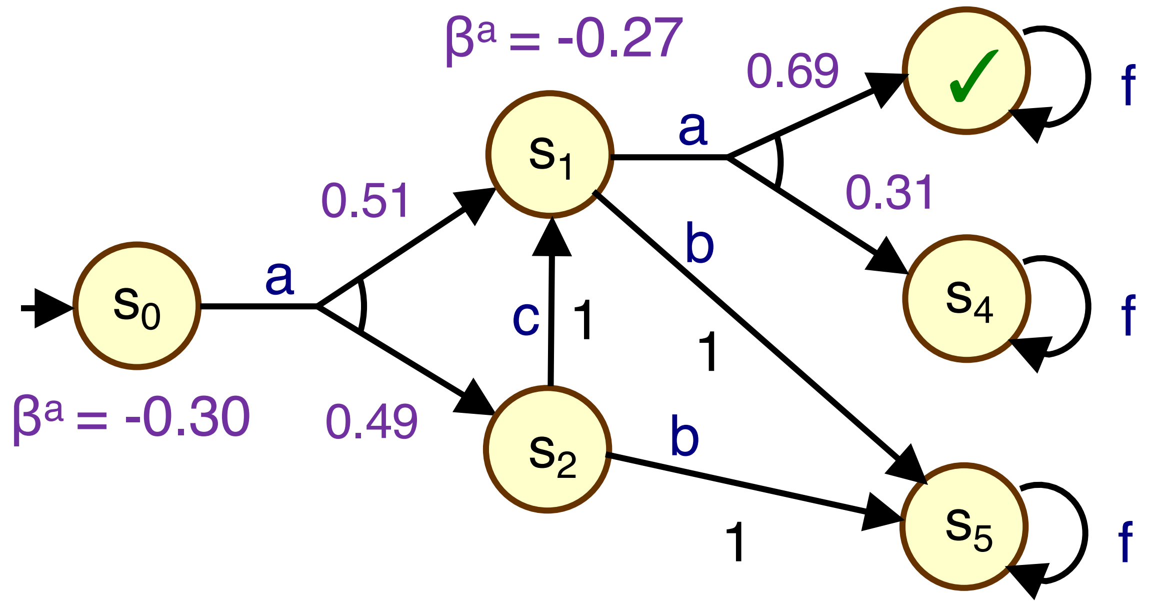
- But other views are possible
  - dynamic, Bayesian, …

- Some examples of uMDPs

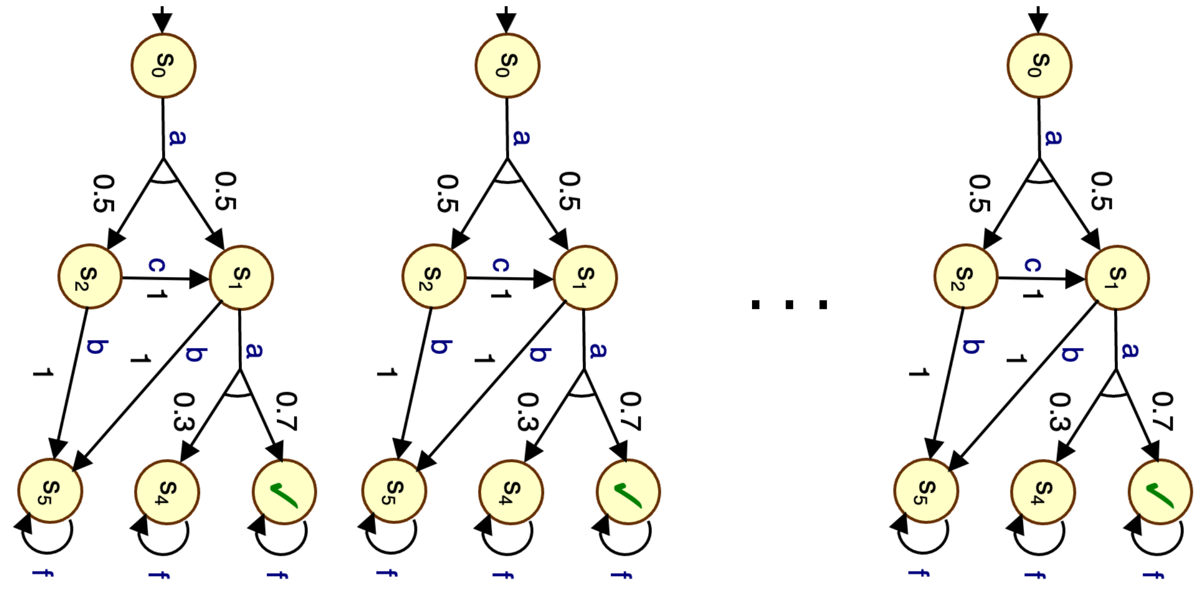
Interval MDPs (IMDPs)



Likelihood MDPs

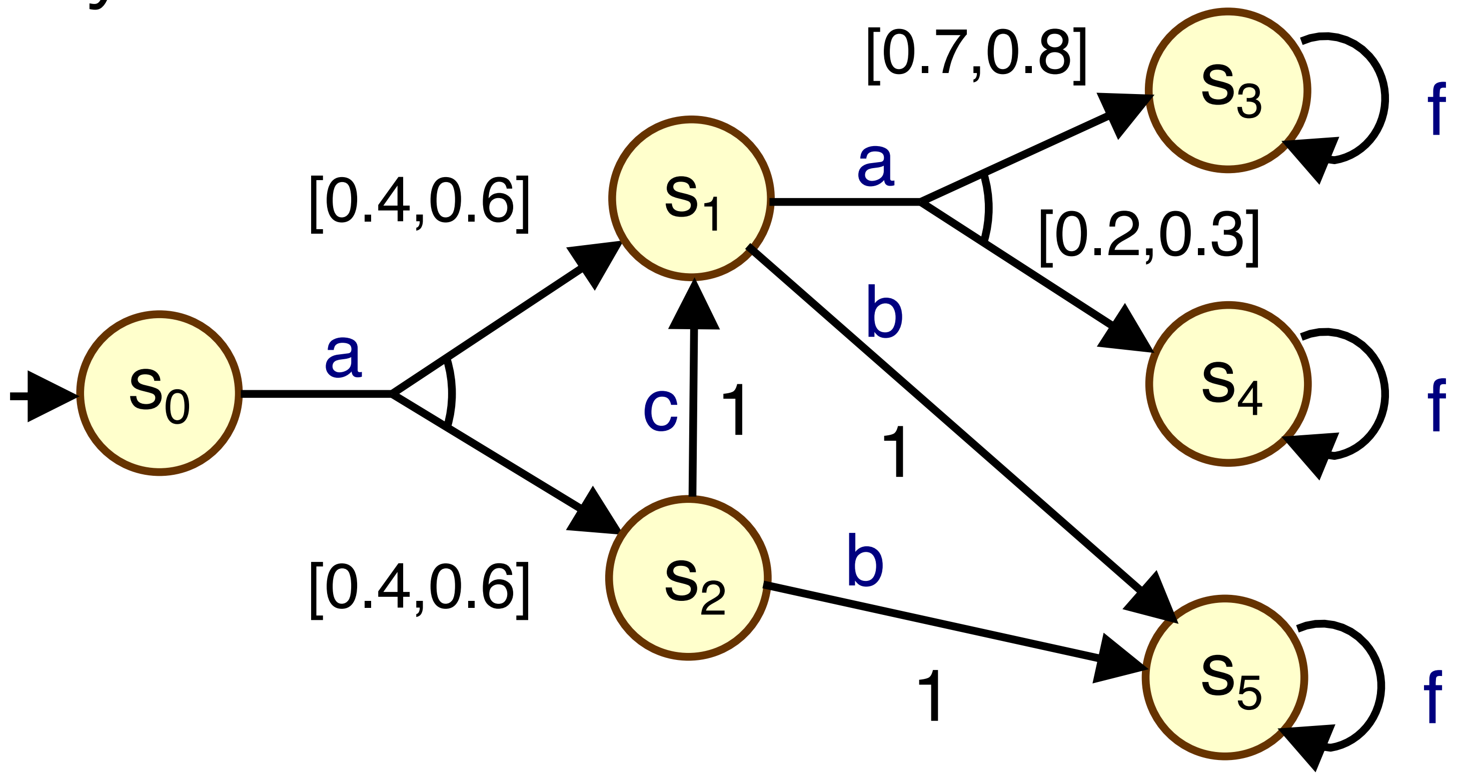


Sampled MDPs



Uncertainty set dependencies

- Can we allow dependencies between uncertainty sets?
  - implications for computational tractability and modelling accuracy



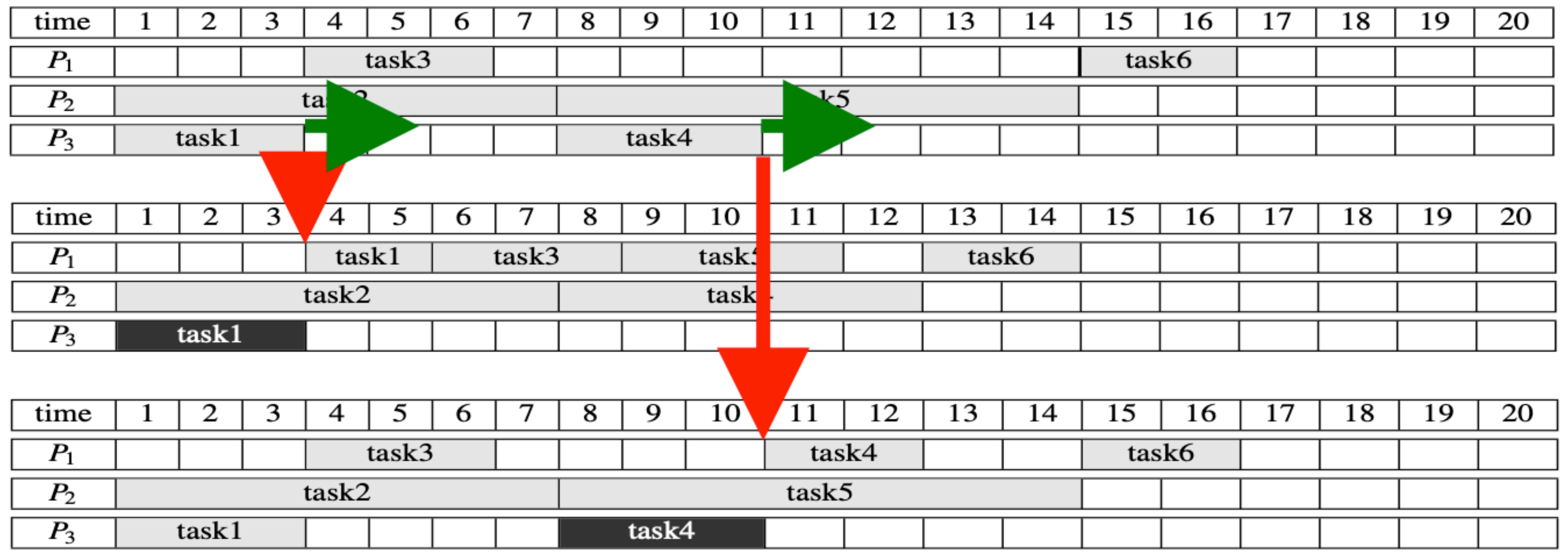
- Rectangularity
  - transition function uncertainty set $\mathscr{P}$ is (s,a)-rectangular
    - if we have $\mathscr{P} = \times_{(s,a)\in S\times A} \mathscr{P}^a_s$
    - i.e., if there are no dependencies between uncertainty sets for each s, a
  - interval MDPs are (s,a)-rectangular ("sampled MDPs" might not be)
  - we will assume (s,a)-rectangularity for now (and later relax it)

- We can also define s-rectangularity [Wiesemann et al.]
  - $\mathscr{P} = \times_{s\in S} \mathscr{P}^s$ where $\mathscr{P}_s = \{(P^a_s)_{a\in A} \;:\; P \in \mathscr{P}\}$

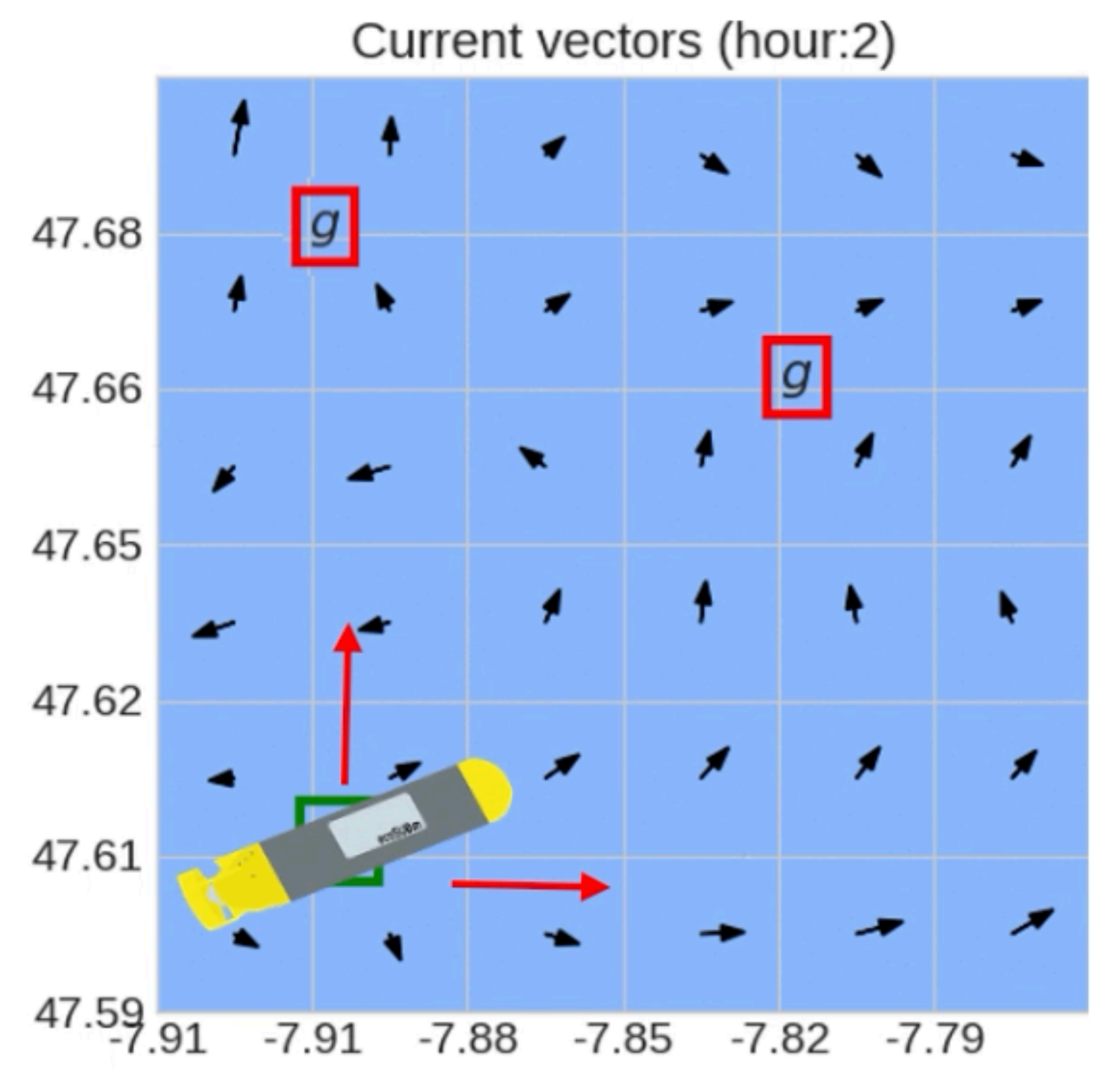
Non-rectangular uMDPs

- When might dependences between uncertainties arise?

Task scheduling in the presence of faulty processors

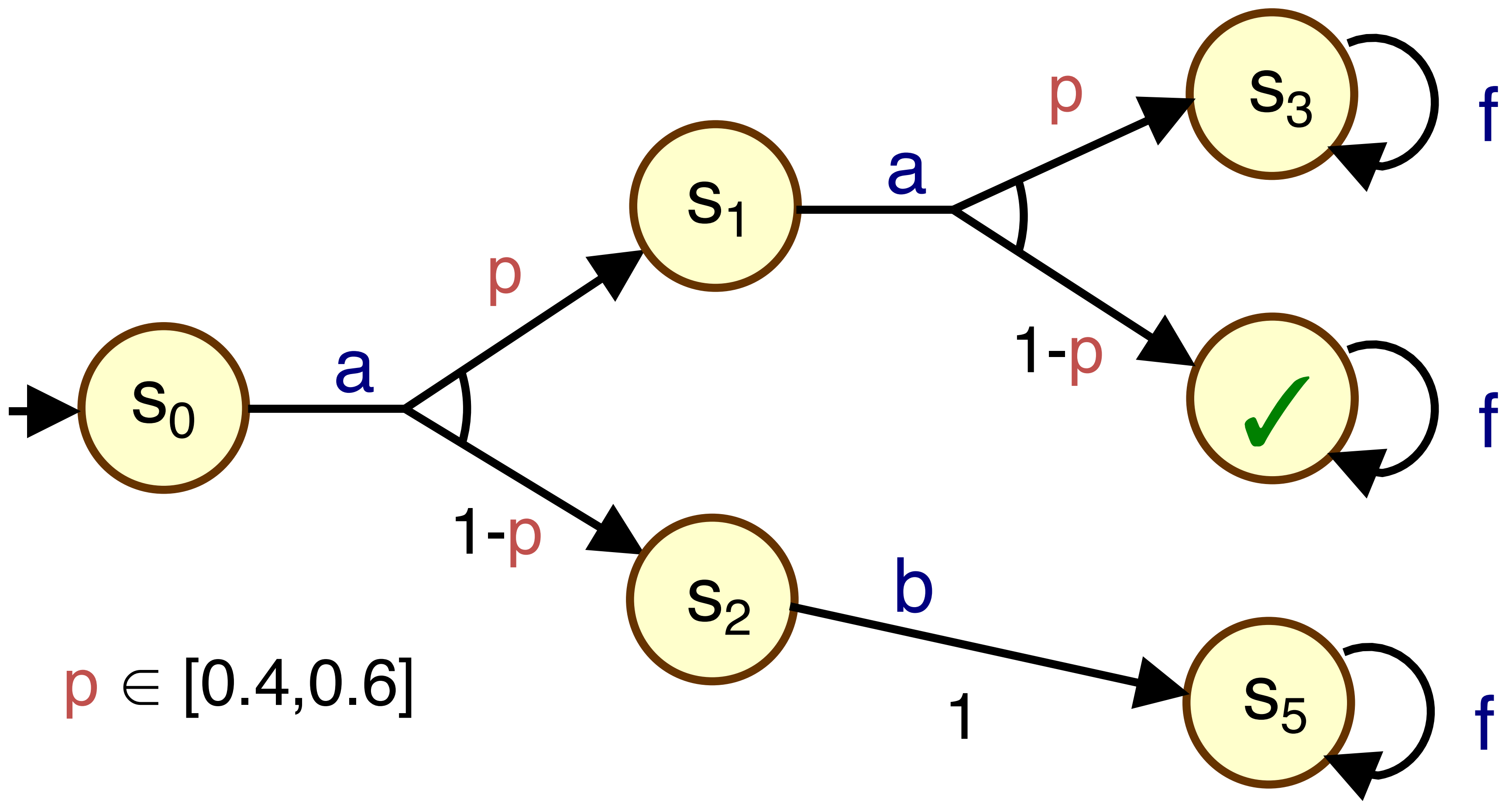


Underwater vehicle control in unknown ocean currents



Non-rectangular uMDPs

- Example MDP (in fact, just a single policy) with parameter p

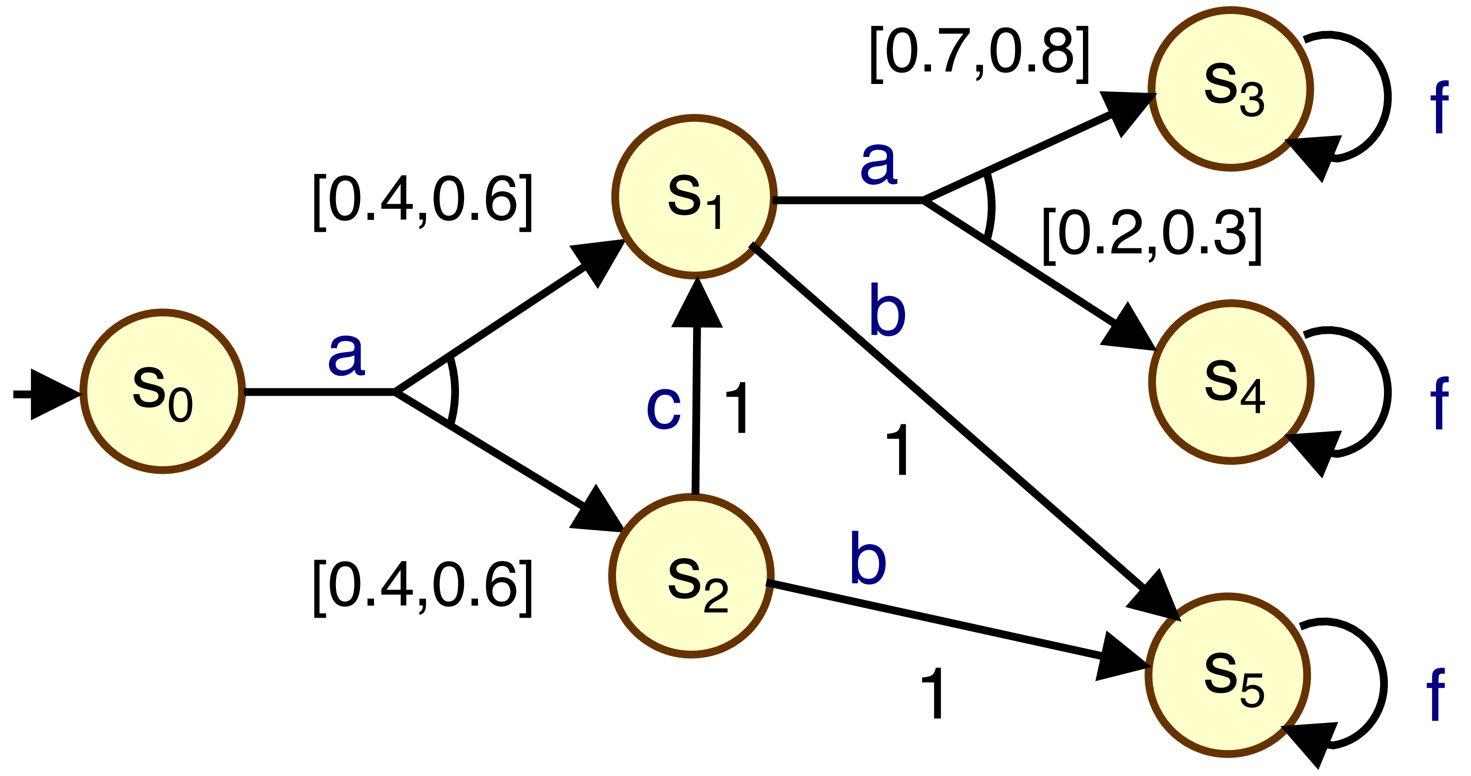


- Worst-case probability to reach ✓?
  - $\min\{p(1-p) \;:\; p \in [0.4,0.6]\} = 0.4 \cdot (1-0.4) = 0.24$

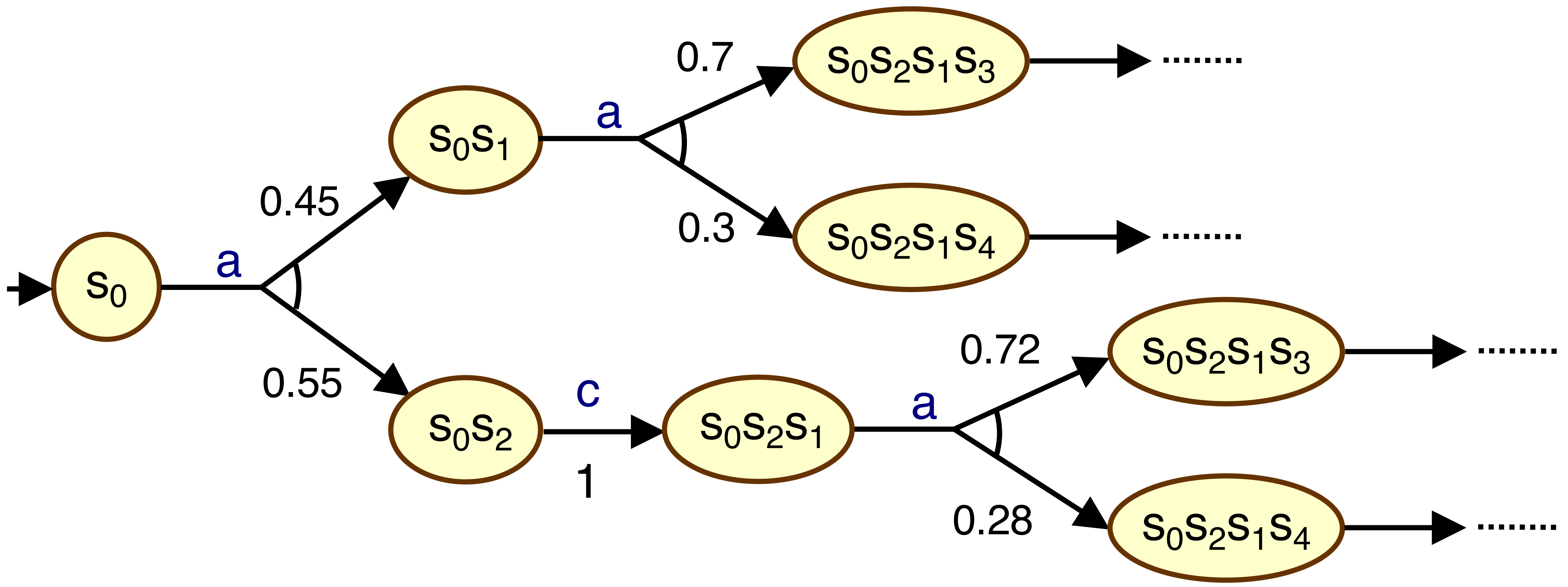
- Worst-case probability to reach ✓ under rectangularity assumptions?
  - $\min\{p_1(1-p_2) \;:\; p_1, p_2 \in [0.4,0.6]\} = 0.4 \cdot (1-0.6) = 0.16$ (too conservative)

Policies in uMDPs

- For uMDPs, as for MDPs, we can define
  - policies $\pi : (S \times A)^* \times S \to A$, or
  - memoryless policies $\pi_m : S \to A$
  - (depending on the set $\mathscr{P}$, some care is needed to make sure policies can be applied)

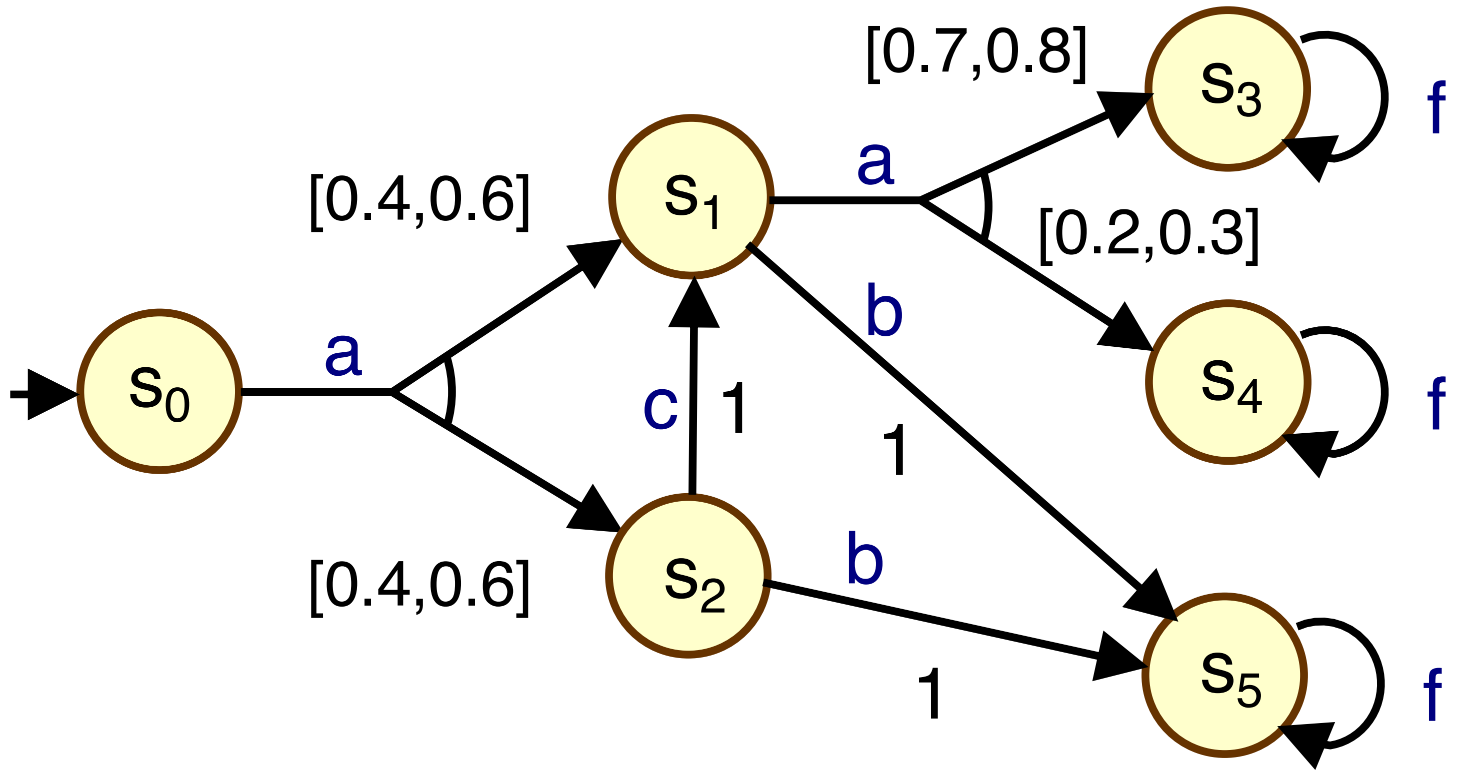


- For policy $\pi \in \Pi$ and transition probabilities $P \in \mathscr{P}$:
  - we can define probability space $Pr_s^{\pi,P}$, random variables $\mathbb{E}_s^{\pi,P}(X)$ and value functions $V^{\pi,P}(s)$
  - which correspond to the MDP $\mathscr{M}[P]$



Robust control

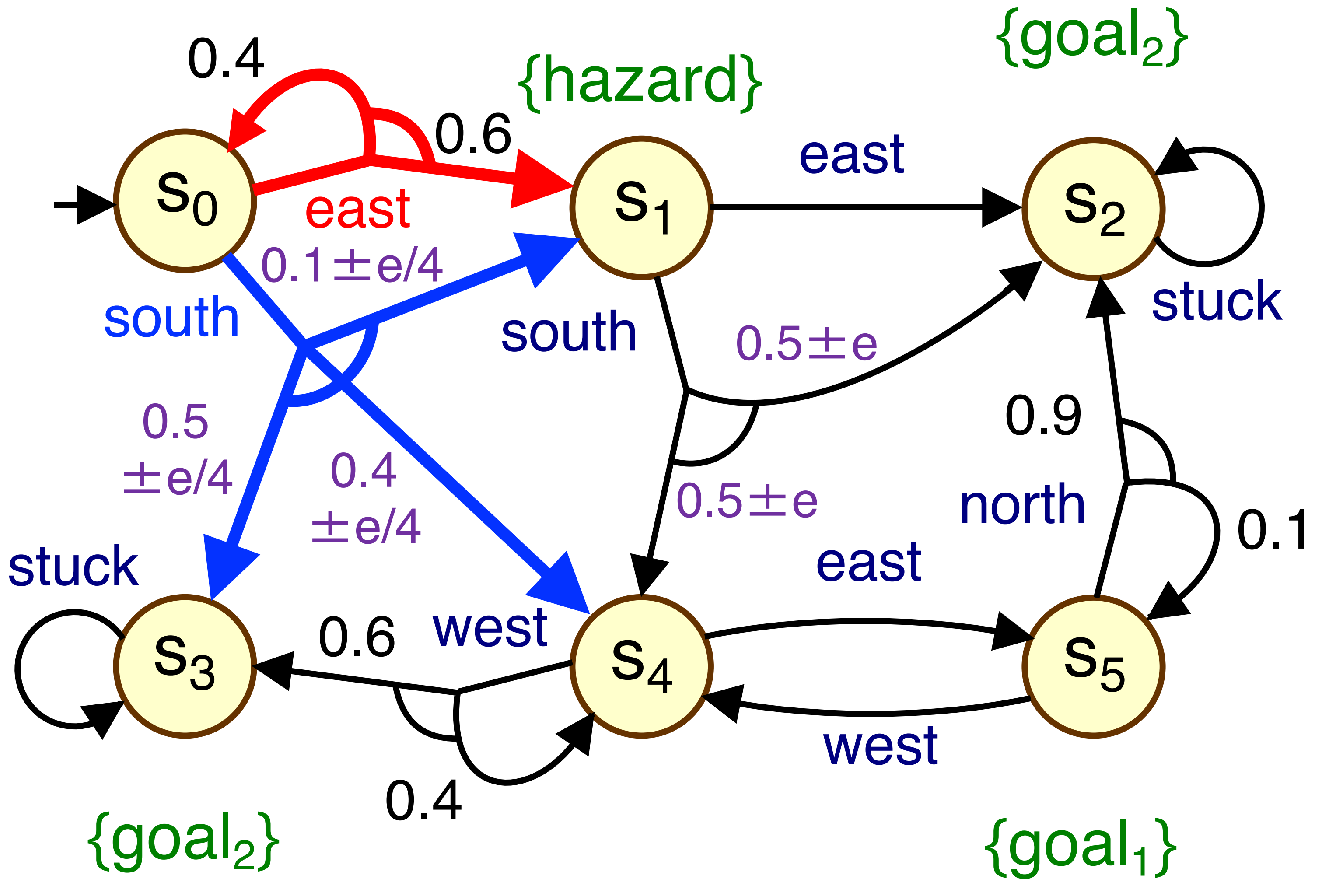
- For now, we consider a robust view of uncertainty
  - i.e., we focus on worst-case (adversarial, pessimistic) scenarios
- Robust policy evaluation:
  - worst-case scenario for (maximising) policy π, i.e.: $\min_{P \in \mathscr{P}} V^{\pi,P}(s)$
- Robust control (policy optimisation):
  - optimal worst-case value $V^*(s) = \max_{\pi \in \Pi} \min_{P \in \mathscr{P}} V^{\pi,P}(s)$
  - optimal worst-case policy $\pi^* = \operatorname{argmax}_{\pi \in \Pi} \min_{P \in \mathscr{P}} V^{\pi,P}(s)$



- Other cases:
  - for a minimising objective (e.g. SPP), we use: $V^*(s) = \min_{\pi \in \Pi} \max_{P \in \mathscr{P}} V^{\pi,P}(s)$
  - we may also consider optimistic scenarios, e.g. $V^*(s) = \max_{\pi \in \Pi} \max_{P \in \mathscr{P}} V^{\pi,P}(s)$

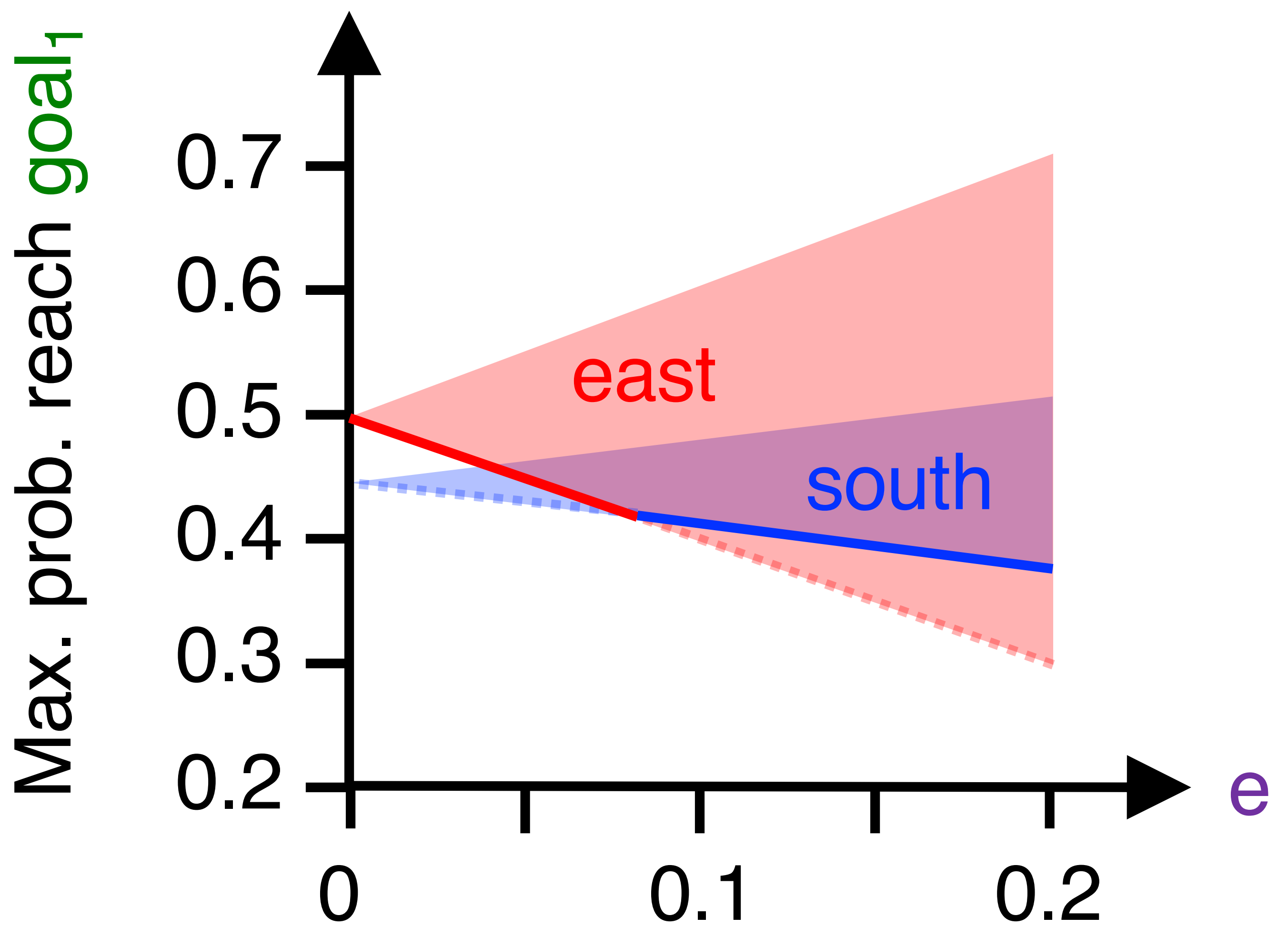
Running example: Robust control

- An IMDP for the robot example
  - uncertainty added to two state-action pairs



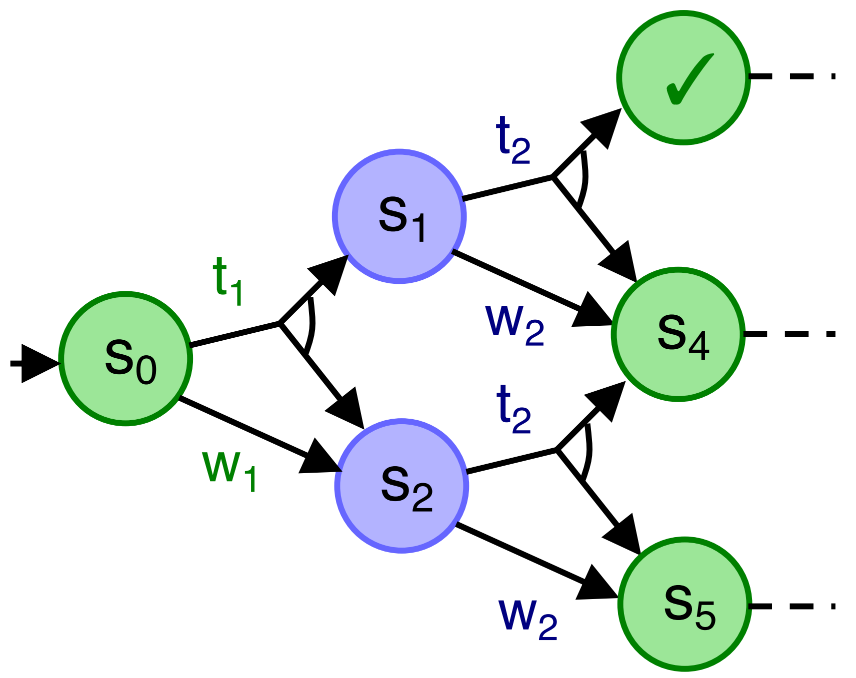
  - Note: the degree of uncertainty (e) in states s_1 and s_2 is correlated here (but the actual transition probabilities are not)

- Robust control
  - for any e, we can pick a "robust" (optimal worst-case) policy
  - and give a safe lower bound on its performance



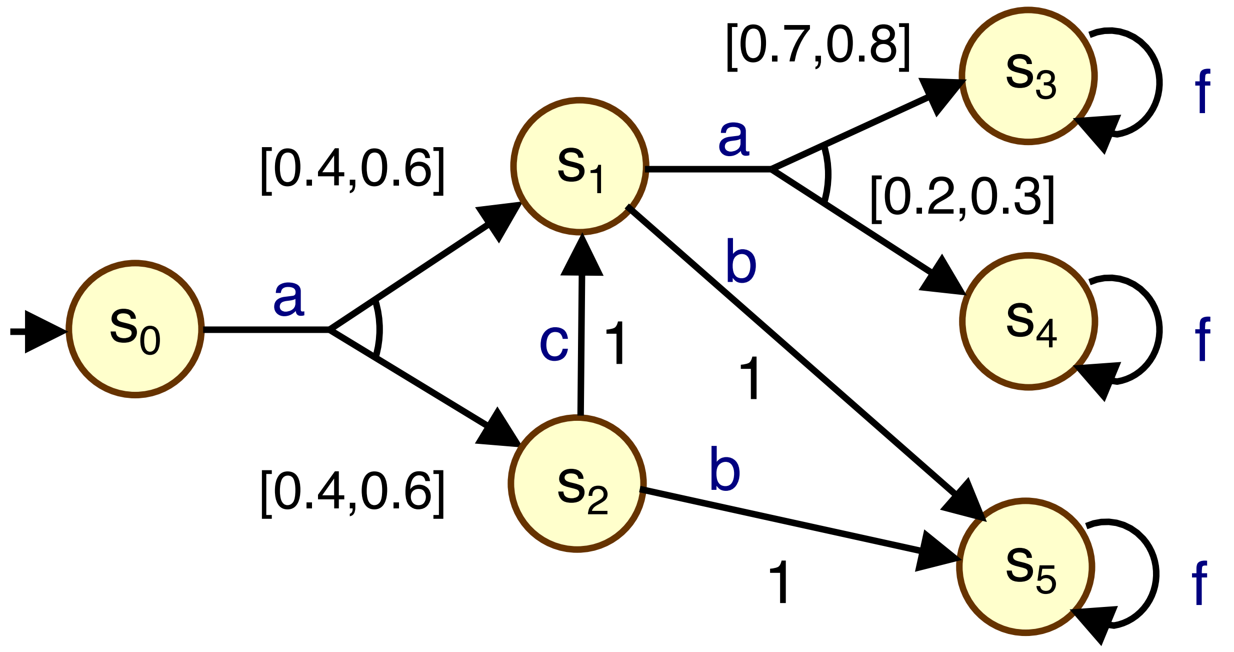
Summary (lecture 2)

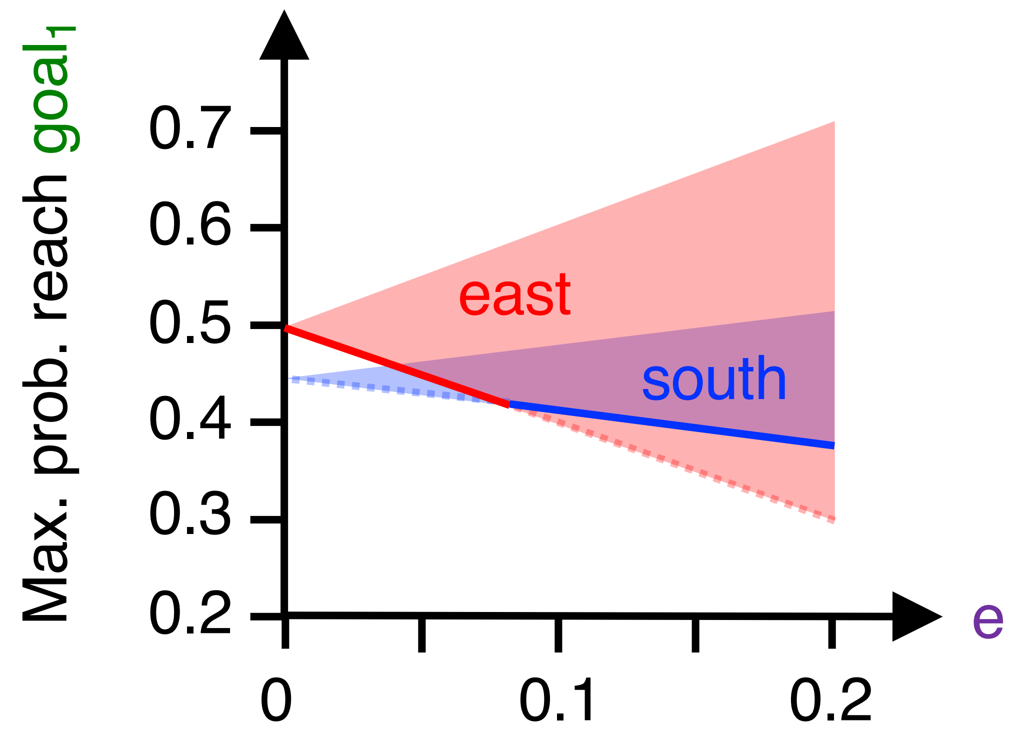
- Stochastic games
  - unknown parts of the system can be modelled adversarially
  - zero-sum turn-based (or concurrent) stochastic games
    - dynamic programming (value iteration) generalises



- Uncertain MDPs
  - MDPs plus epistemic uncertainty: set of transition functions
    - each $P \in \mathscr{P}$ is a transition function $P : S \times A \times S \to [0,1]$
  - rectangularity (dependencies)
  - control policies + robust control

$$V^*(s) = \max_{\pi \in \Pi} \min_{P \in \mathscr{P}} V^{\pi,P}(s)$$

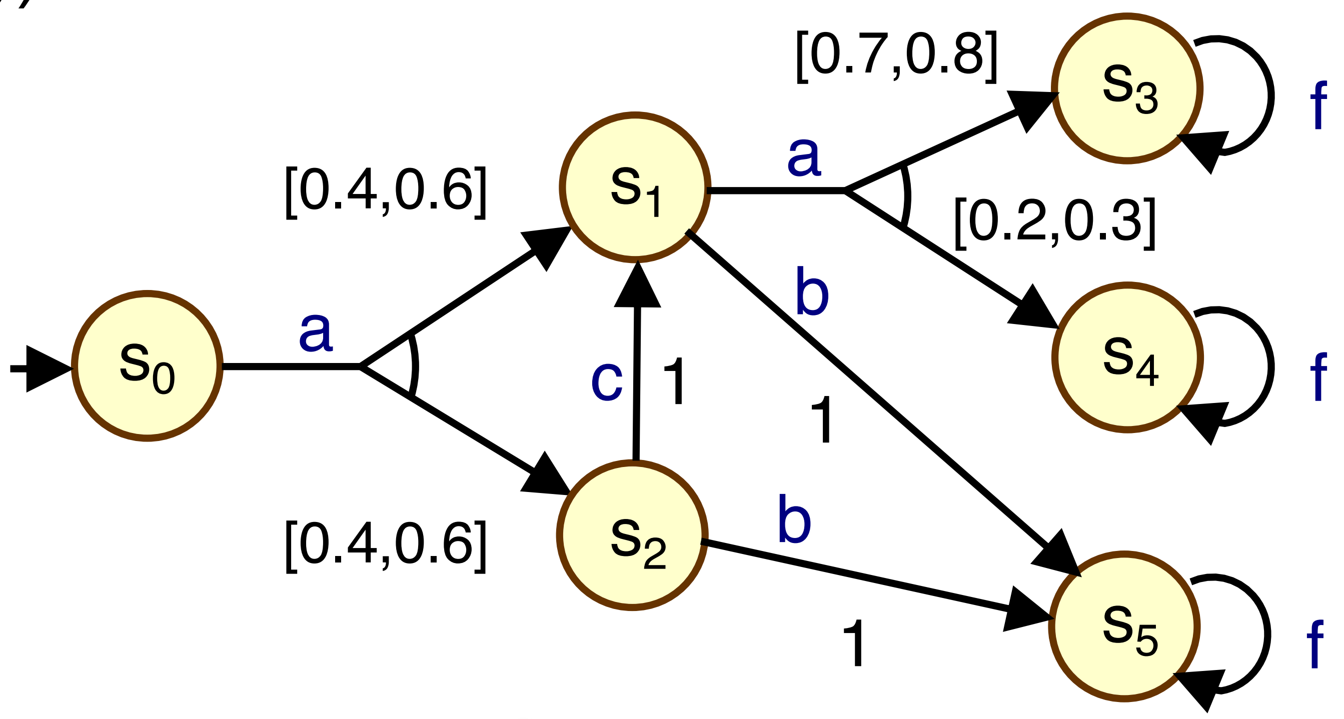




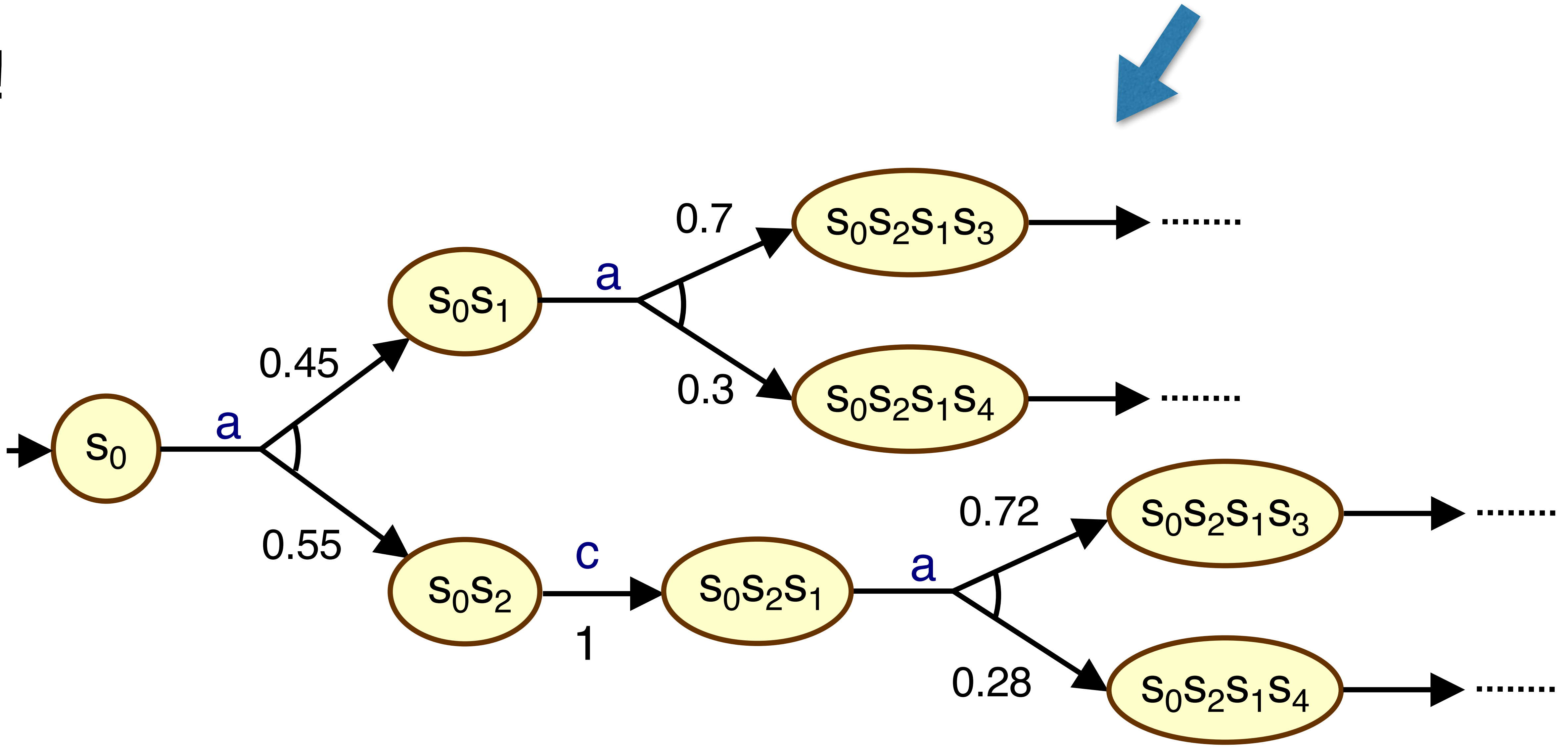
Uncertain MDPs

Resolving uncertainty

- Now we consider a more dynamic approach to resolving uncertainty
    - (which we will need to extend dynamic programming to this setting)
- An environment policy (or nature policy, or adversary) $\tau \in \mathscr{T}$
    - is a mapping $\tau : (S \times A)^* \times (S \times A) \rightarrow Dist(S)$
    - such that $\tau(s_0, a_0, \ldots, s_n, a_n) \in \mathscr{P}_s^a$
    - note: this assumes (s,a)-rectangularity!

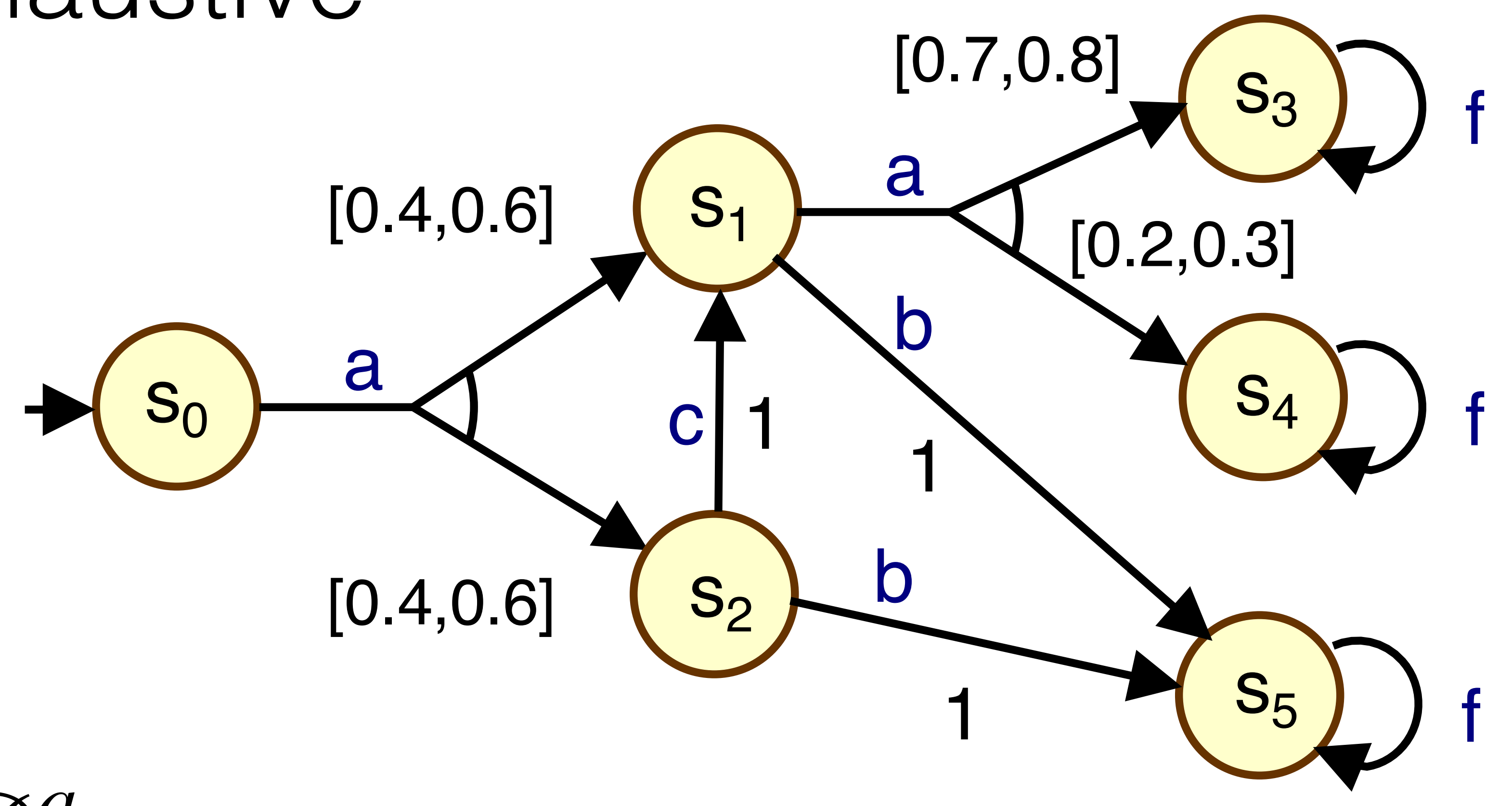


- Policies π, τ yield
    - a probability space $Pr_s^{\pi,\tau}$
    - random variables $\mathbb{E}_s^{\pi,\tau}(X)$
    - and value functions $V^{\pi,\tau}$



Dynamic vs. static uncertainty

- Quantifying over environment policies $\tau \in \mathcal{T}$ is more exhaustive
  - than quantifying over transition probabilities $P \in \mathcal{P}$
  - $\{Pr_s^{\pi,P} : P \in \mathcal{P}\} \subseteq \{Pr_s^{\pi,\tau} : \tau \in \mathcal{T}\}$

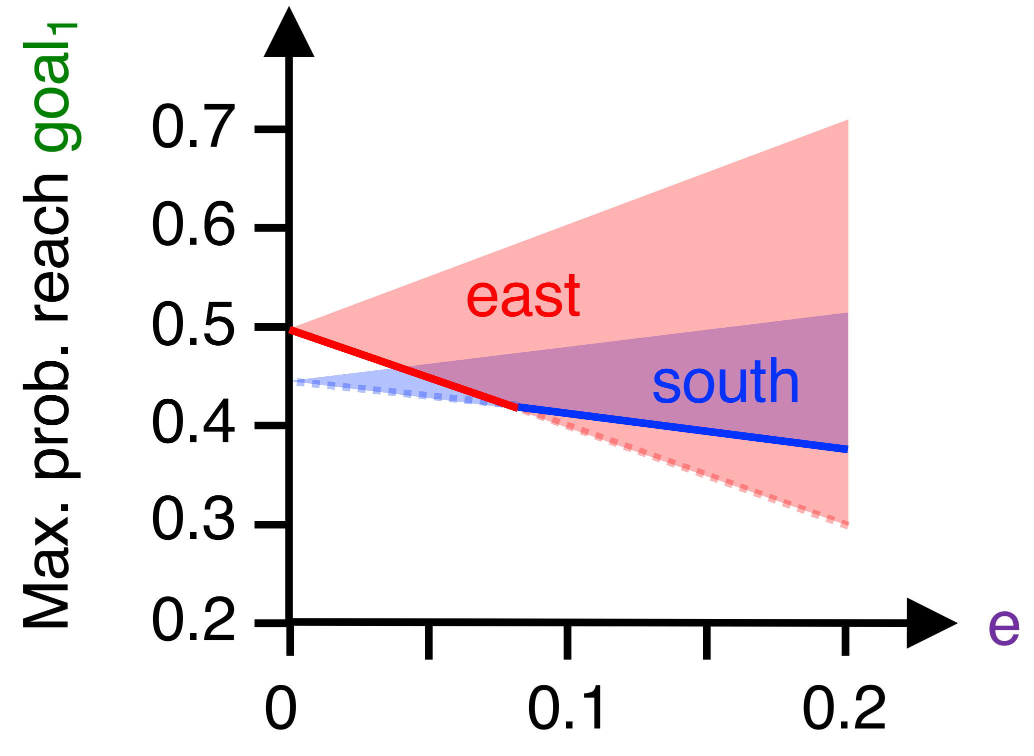


- Memoryless (stationary) environment policies $\tau_m \in \mathcal{T}_m$
  - are mappings $\tau_m : S \times A \to Dist(S)$ such that $\tau_m(s,a) \in \mathcal{P}_s^a$
  - in this case, the semantics now coincide:
  - $\{Pr_s^{\pi,P} : P \in \mathcal{P}\} = \{Pr_s^{\pi,\tau_m} : \tau_m \in \mathcal{T}_m\}$

- We call this dynamic uncertainty ($\tau \in \mathcal{T}$) vs. static uncertainty ($P \in \mathcal{P}$)
  - which to use is a modelling decision (e.g., on the timing of events)
  - but there are also implications for tractability
  - similar situation to rectangularity (uncertainty set independence)

Robust control (revisited)

- Robust control
  - but quantifying over policies (rather than uncertainty sets)



- Now we have
  - optimal worst-case value

$$V^*(s) = V^{\Pi,\mathcal{T}}(s) = \max_{\pi\in\Pi}\min_{\tau\in\mathcal{T}} V^{\pi,\tau}(s)$$

notation for optimal value for sets of control/environment policy sets $\Pi, \mathcal{T}$

  - optimal worst-case policy

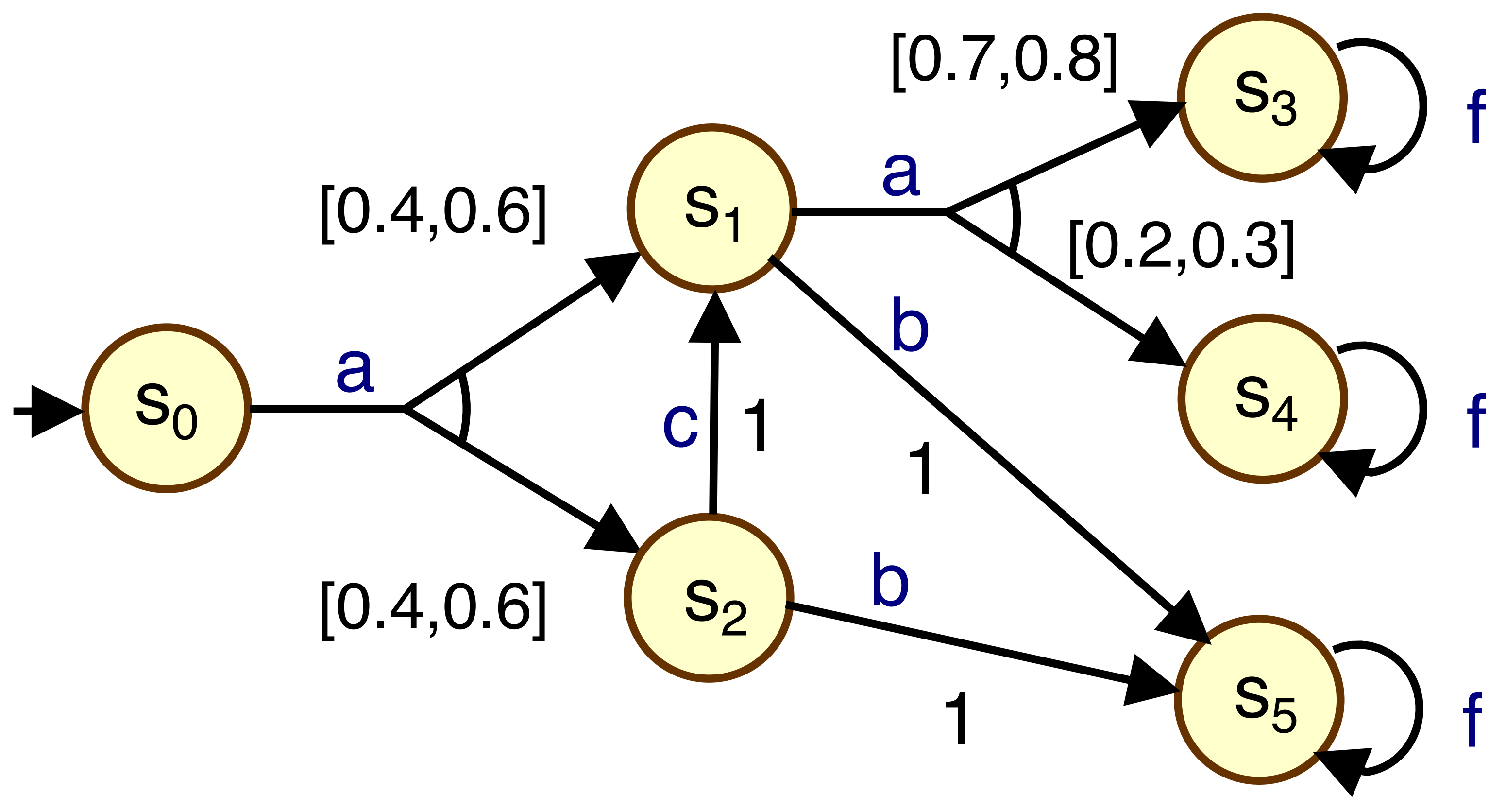
$$\pi^* = \operatorname*{argmax}_{\pi\in\Pi}\min_{\tau\in\mathcal{T}} V^{\pi,\tau}(s)$$

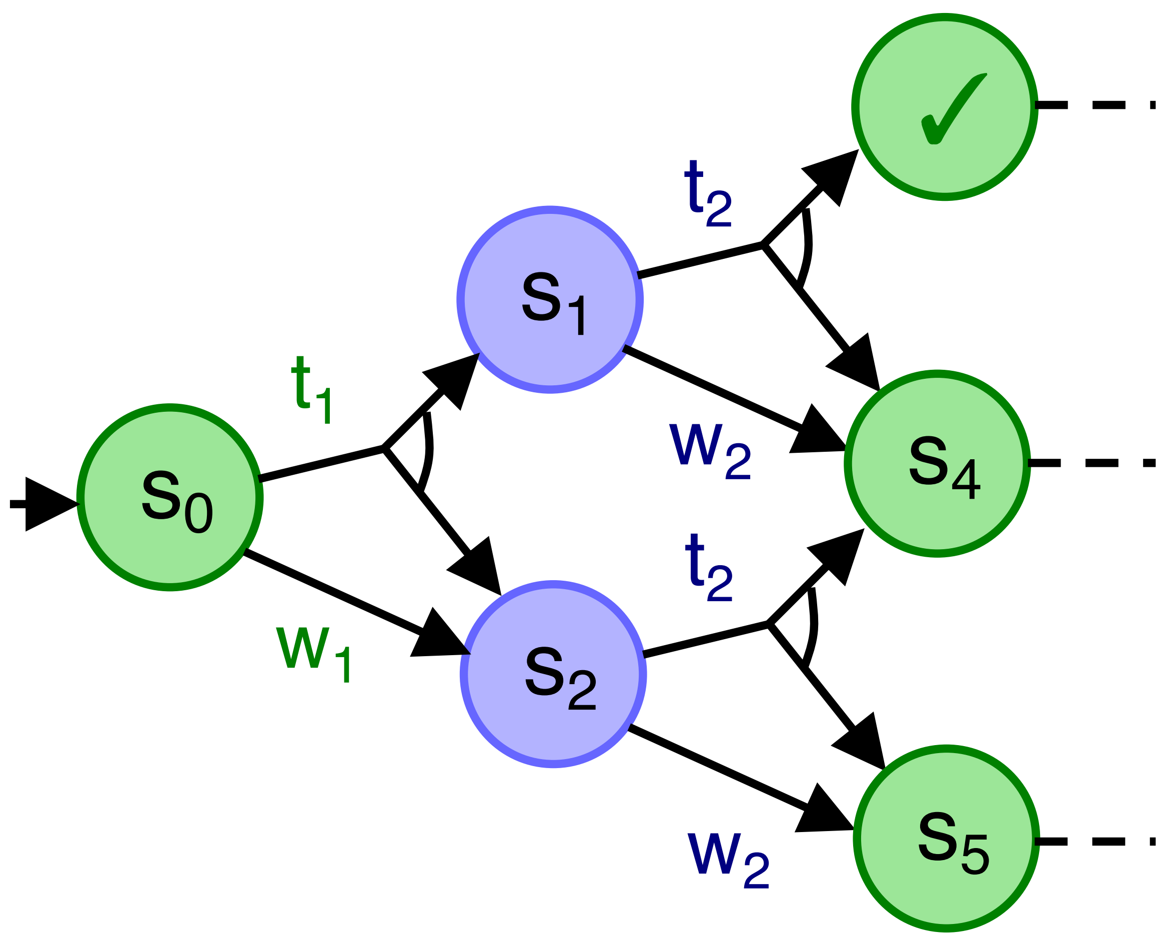
- Note that we may want to quantify over mismatching sets of policies, e.g.:

$$V^{\Pi,\mathcal{T}_m}(s) = \max_{\pi\in\Pi}\min_{\tau_m\in\mathcal{T}_m} V^{\pi,\tau_m}(s) = \max_{\pi\in\Pi}\min_{P\in\mathcal{P}} V^{\pi,P}(s)$$

e.g. for static uncertainty

uMDPs vs stochastic games





Robust dynamic programming

- Let's again focus on optimising MaxProb (the situation is similar for SSP)
  - and recall: we <u>need</u> to assume (s,a)-rectangularity

- Memoryless policies suffice, for <u>both</u> the controller and the environment

$$V^{\Pi,\mathcal{T}}(s_0) = V^{\Pi_m,\mathcal{T}_m}(s_0) = V^{\Pi_m,\mathcal{T}}(s_0) = V^{\Pi,\mathcal{T}_m}(s_0)$$

- Perfect duality:

$$V^{\Pi,\mathcal{T}}(s_0) = \max_{\pi\in\Pi}\min_{\tau\in\mathcal{T}} V^{\pi,\tau}(s_0) = \min_{\tau\in\mathcal{T}}\max_{\pi\in\Pi} V^{\pi,\tau}(s_0)$$

- The optimal value function satisfies the Bellman equation:

$$V^*(s) = V^{\Pi,\mathcal{T}}(s) = \begin{cases} 1 & \text{if } s \in goal \\ \max_{a\in A(s)} \inf_{P_s^a\in\mathcal{P}_s^a} \sum_{s'\in S} P_s^a(s') \cdot V^{\Pi,\mathcal{T}}(s') & \text{otherwise} \end{cases}$$

Robust value iteration

- Optimal values for uMDPs can be obtained using robust value iteration (robust VI)
  - from the limit of the vector sequence defined below
  - $V^*(s) = \lim_{k\to\infty} x_s^k$ where:

We will re-use graph-based pre computation for MDPs

$$x_s^k = \begin{cases} 1 & \text{if } s \in S^1 \\ 0 & \text{if } s \in S^0 \\ 0 & \text{if } s \in S^? \text{ and } k = 0 \\ \max_{a\in A(s)} \inf_{P_s^a \in \mathscr{P}_s^a} \sum_{s'\in S} P_s^a(s') \cdot x_{s'}^{k-1} & \text{otherwise} \end{cases}$$

- Again, this Bellman operator is (i) monotonic (ii) a contraction in the L_∞ norm
  - needs (s-a)-rectangularity, but no assumptions on convexity
  - (it suffices to take convex hull of each $\mathscr{P}_s^a$)

Uncertainty set representations

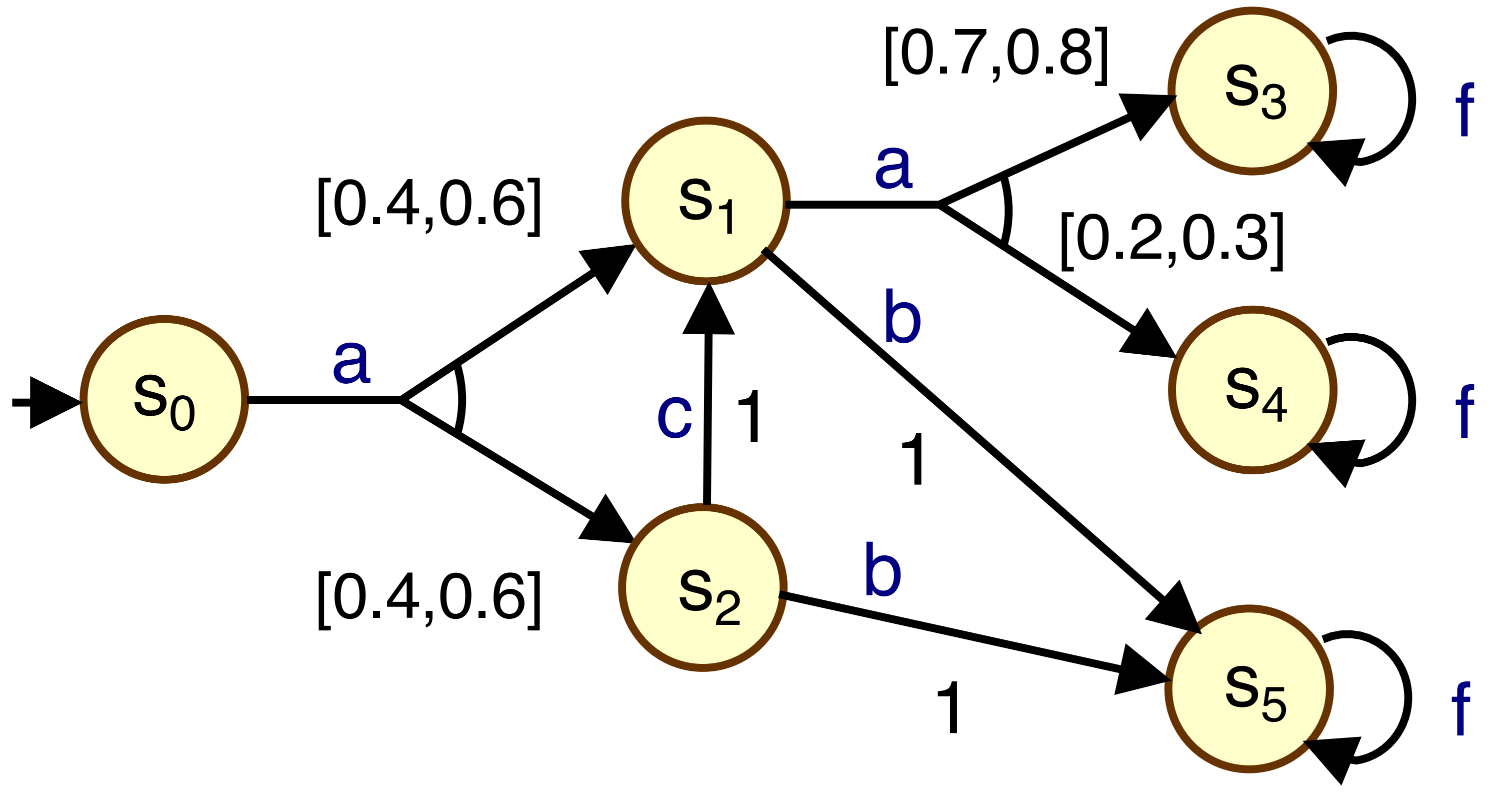
- The core step of robust VI comprises two nested optimisation problems:

$$\max_{a \in A(s)} \inf_{P_s^a \in \mathscr{P}_s^a} \sum_{s' \in S} P_s^a(s') \cdot x_{s'}$$

  - Outer problem (optimal control action)
  - Inner problem (worst-case transition probabilities)

  where x is some vector of values

- Computational cost: robust VI potentially not much more expensive than VI for MDPs
  - if the inner problem can solved efficiently
  - note: uncertainty sets $\mathscr{P}_s^a$ are usually infinite
- Definition/representation of uncertainty sets?
  - trade off statistical accuracy vs. computation efficiency?

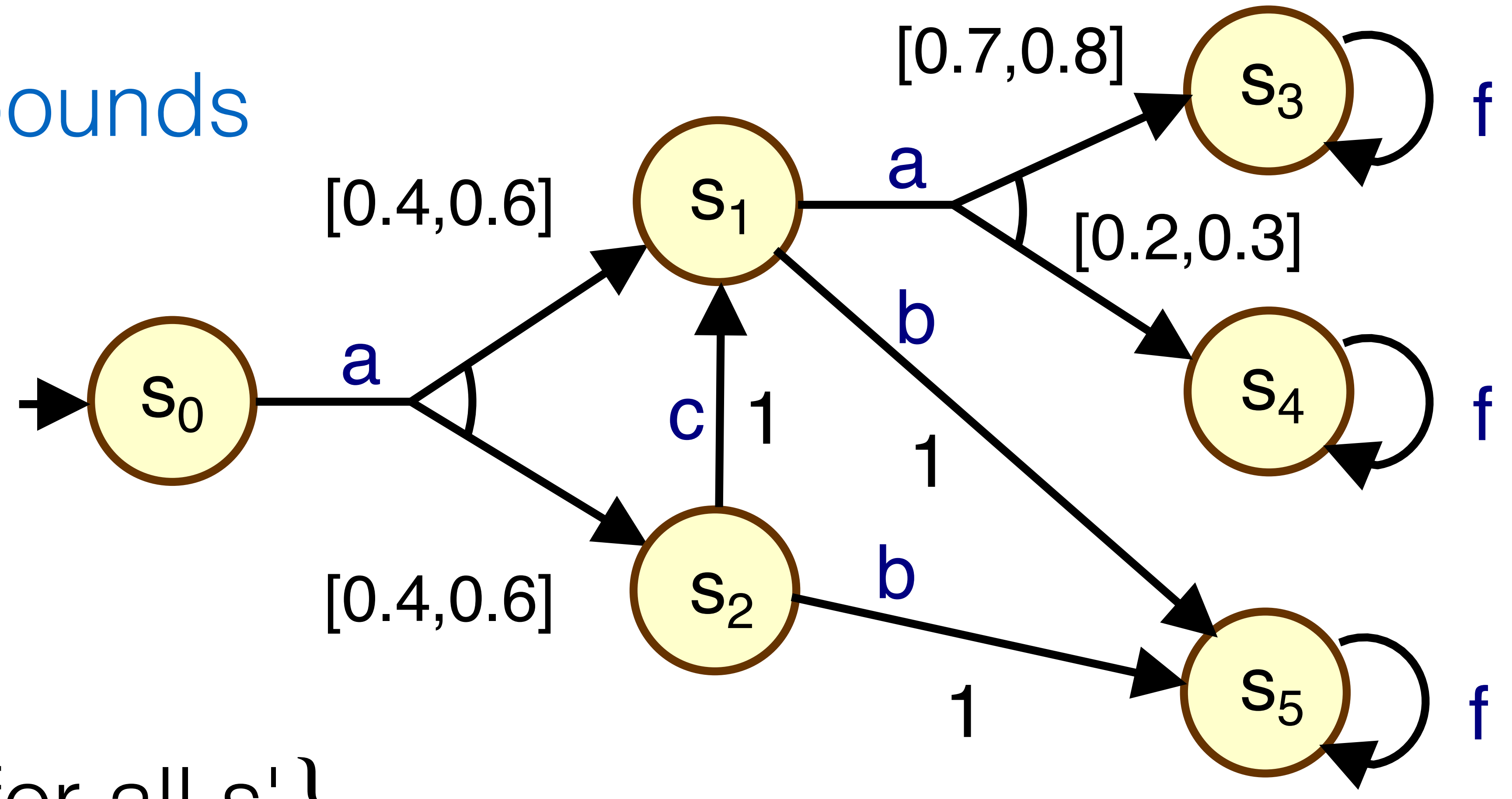


- First example: intervals, a simple uncertainty set representation
  - which suit statistical estimates of confidence intervals for individual transition probabilities

Interval MDPs

Interval MDPs

- An interval MDP (IMDP) is of the form $\mathscr{M} = (S, s_0, A, \underline{P}, \overline{P})$ where:
  - states S, initial state s_0 and actions A are as for MDPs
  - $\underline{P} : S \times A \times S \rightarrow [0,1]$ gives transition probability lower bounds
  - $\overline{P} : S \times A \times S \rightarrow [0,1]$ gives transition probability upper bounds
    - such that $\underline{P}(s, a, s') \leq \overline{P}(s, a, s')$ for all s, a, s'



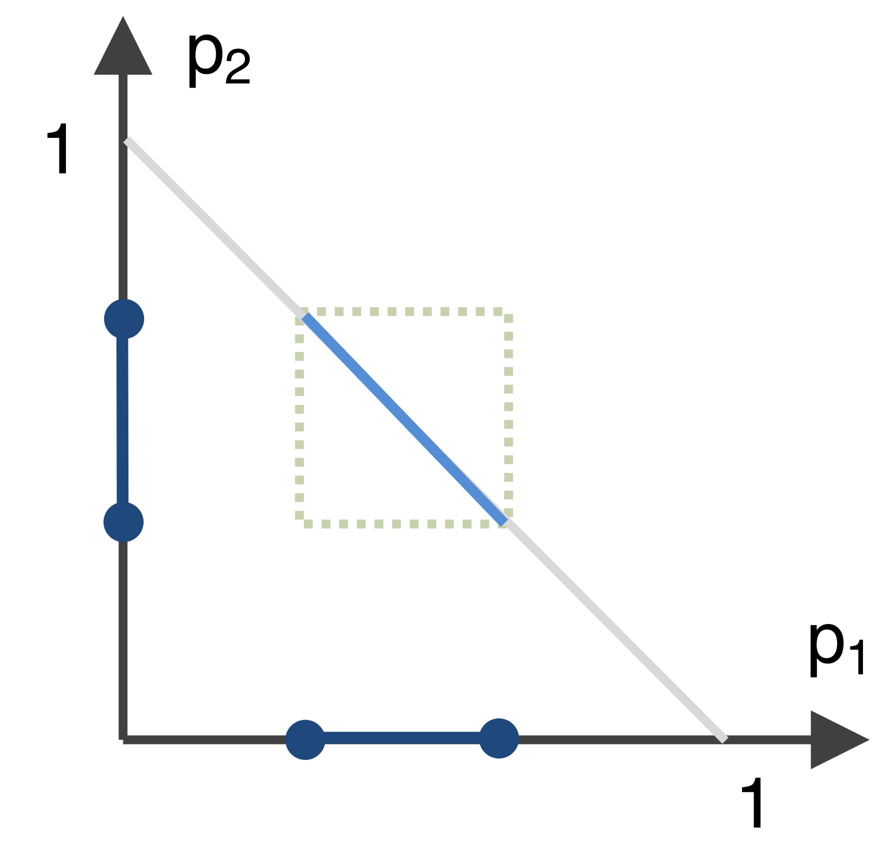
- IMDP uncertainty sets
  - $\mathscr{P}_s^a = \{P_s^a \in Dist(S) \mid \underline{P}(s, a, s') \leq P_s^a(s') \leq \overline{P}(s, a, s')$ for all s'$\}$
    - probabilities are independent (except for the need to sum to 1)
  - $\mathscr{P} = \times_{(s,a) \in S \times A} \mathscr{P}_s^a$
    - i.e., IMDPs are (s-a)-rectangular

IMDP uncertainty sets

- Interval uncertainty sets $\mathscr{P}_s^a$ are convex subsets of $[0,1]^{|S|}$

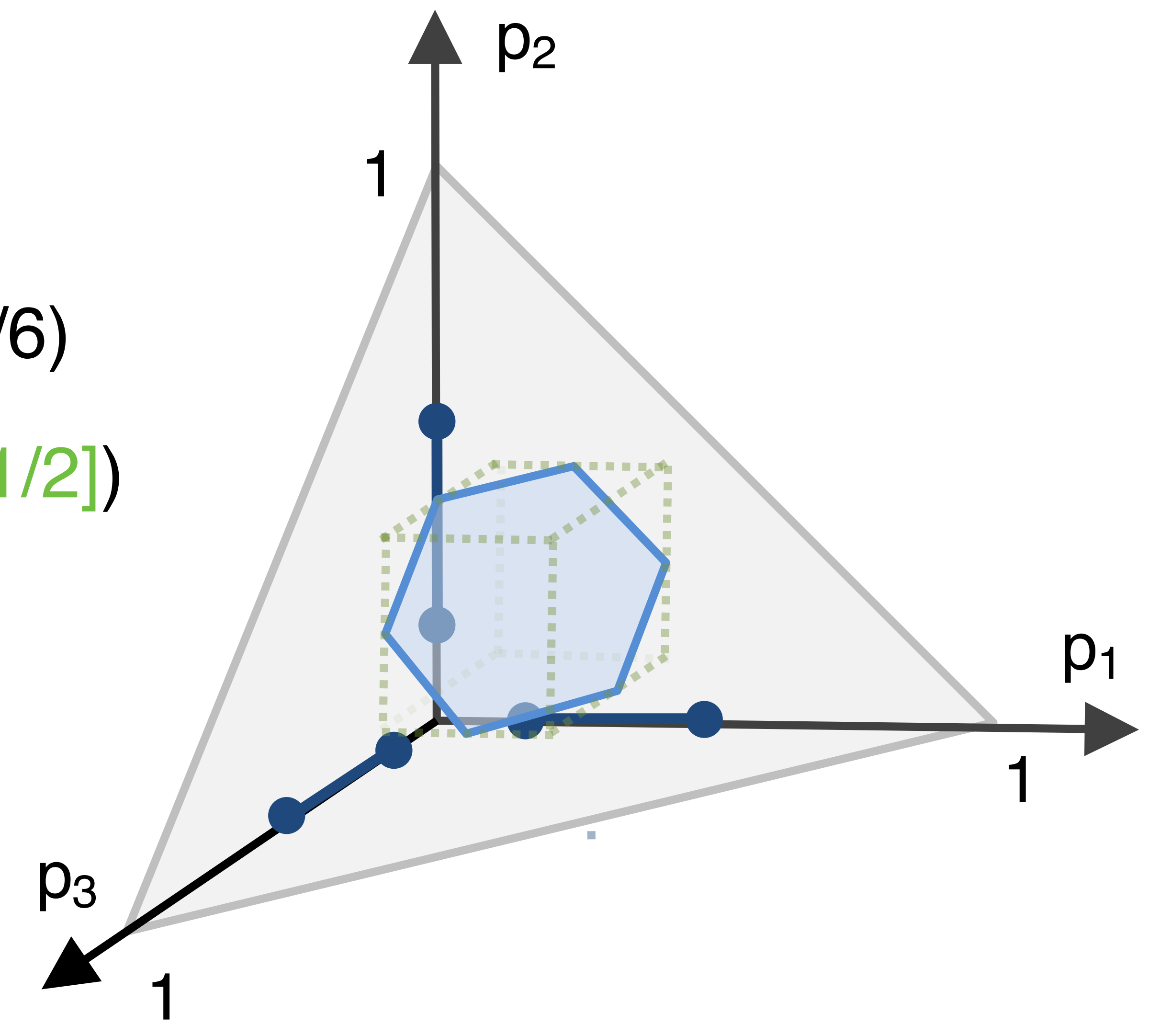
$(p_1,p_2) \in (1/2,1/2) \pm (1/6,1/6)$

$= ([1/3,2/3], [1/3,2/3])$



$(p_1,p_2,p_3) \in (1/3,1/3,1/3) \pm (1/6,1/6,1/6)$

$= ([1/6,1/2], [1/6,1/2], [1/6,1/2])$



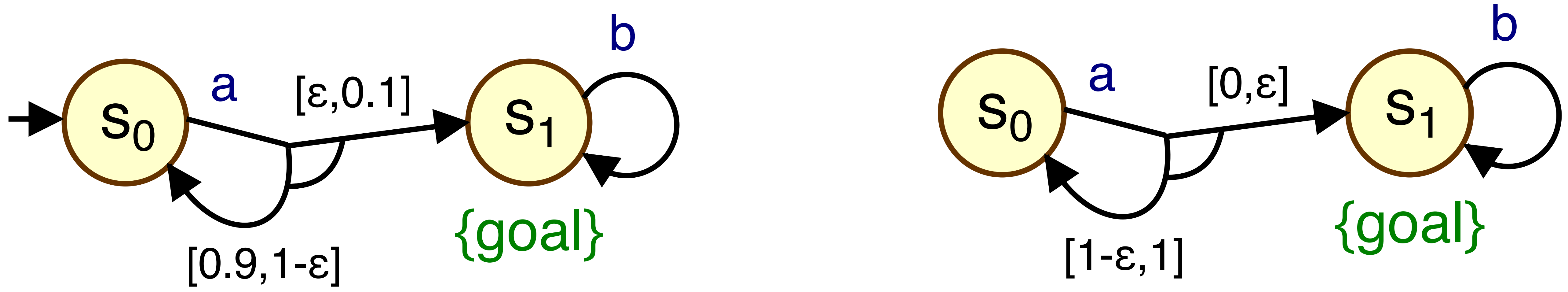
- We can delimit the intervals
  - i.e., trim the interval bounds such that at least one possible distribution takes each extremal value
  - e.g., $\underline{P}(s') := \max[\underline{P}(s'), 1 - \Sigma_{s \neq s'} \overline{P}(s)]$
    - e.g. [0.1,0.4], [0.5,0.8] → [0.2,0.4], [0.6,0.8]

An assumption on IMDPs

- Assumption: IMDPs have a fixed underlying transition graph
  - i.e., for each s,a,s' either: (i) $\underline{P}(s,a,s') > 0$; or

    (ii) $\underline{P}(s,a,s') = \overline{P}(s,a,s') = 0$

- Otherwise behaviour can be qualitatively different for small changes in $P(s,a,s')$

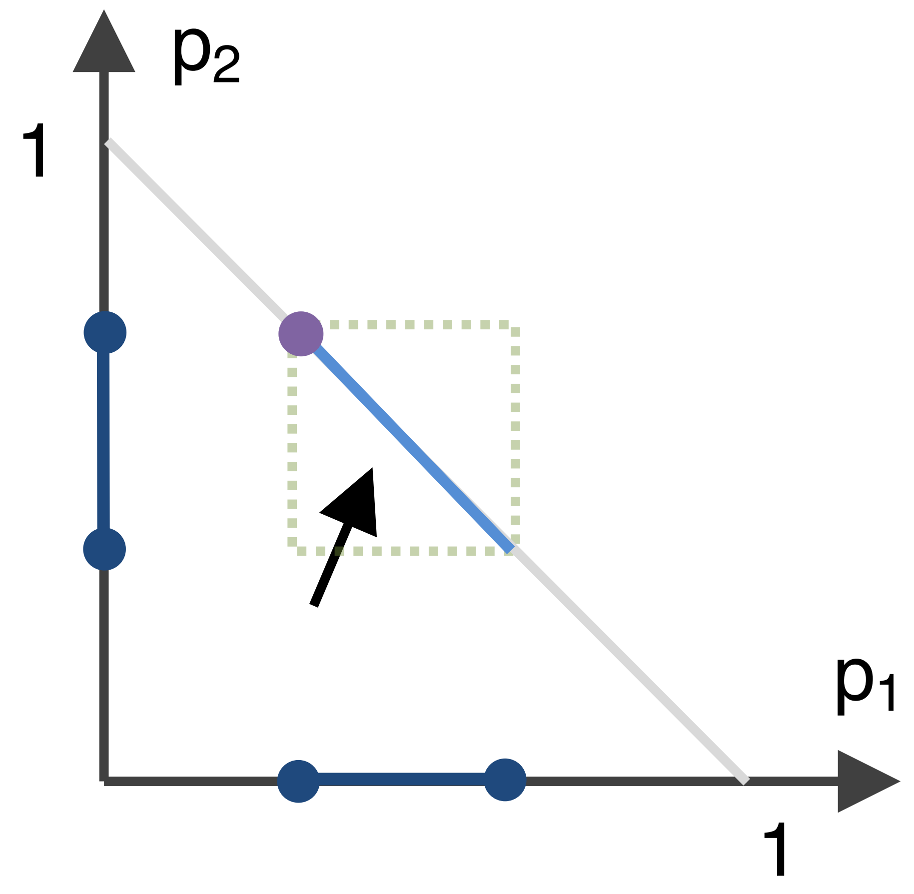


  - For $\varepsilon > 0$, the probability to reach goal is always 1
  - For $\varepsilon = 0$, the probability to reach goal can be 0
  - (contrast to, e.g., a finite-horizon property MaxProb$^{\leq k}$(goal))

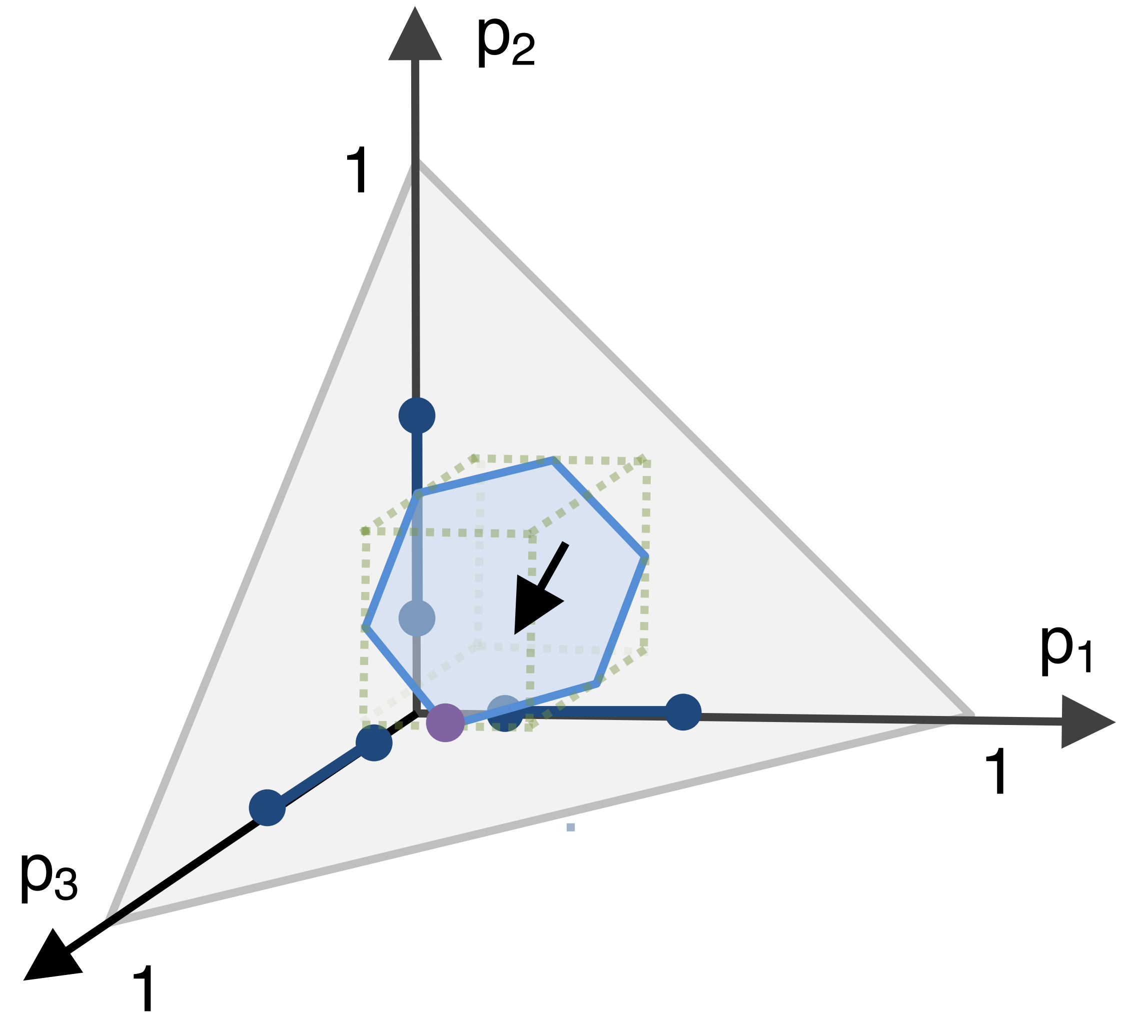
Robust value iteration for IMDPs

- The inner problem for each iteration, and each (s, a) is: $\inf_{P_s^a \in \mathscr{P}_s^a} \sum_{s' \in S} P_s^a(s') \cdot x_{s'}$

- Can be solved via a linear programming problem:
    - let $p_{s'}$ be $|S|$ variables for the chosen probabilities $P_s^a(s')$



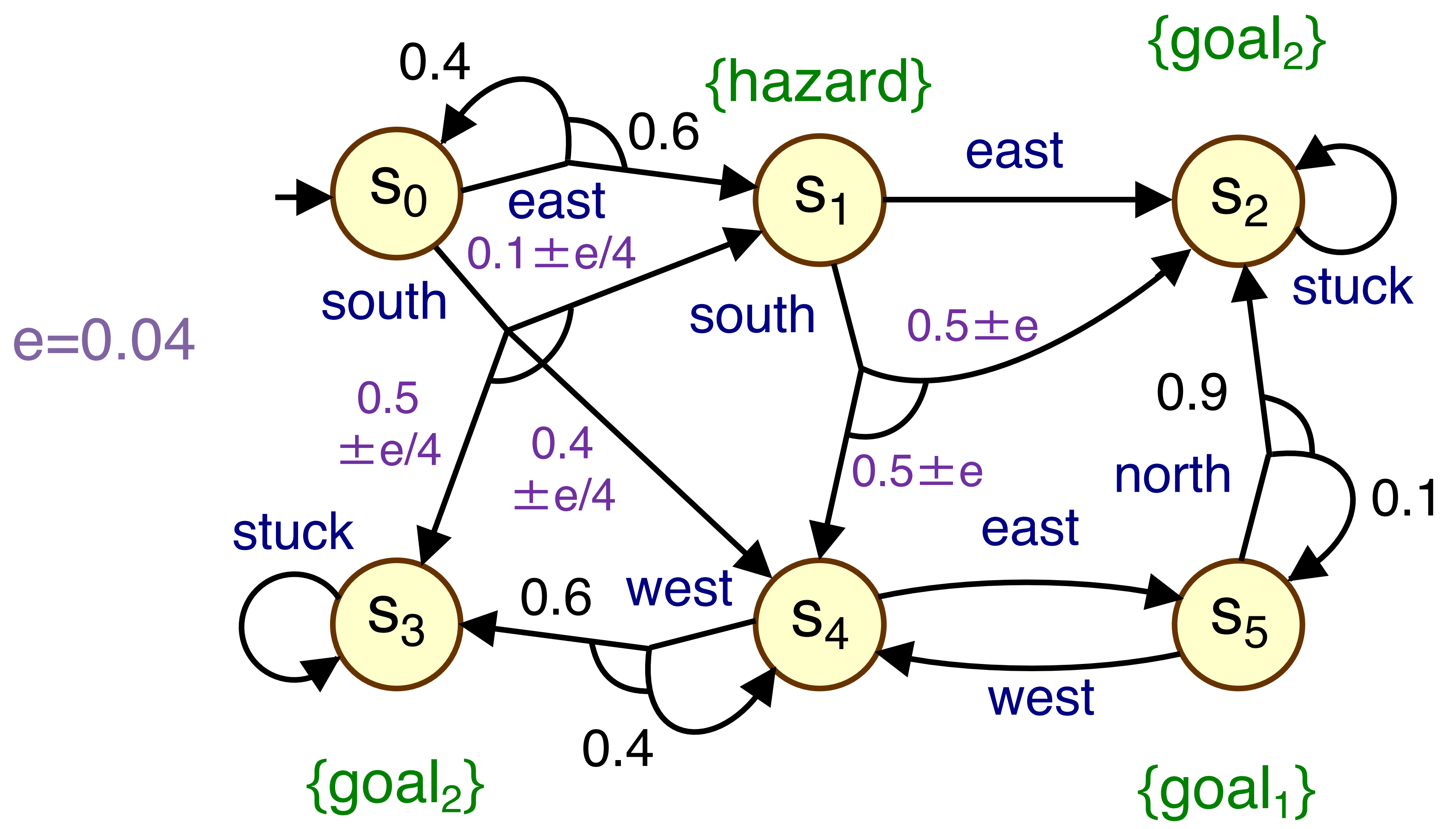
> minimise $\Sigma_{s'} p_{s'} \cdot x_{s'}$ such that:
>
> $\underline{P}_s^a(s') \leq p_{s'} \leq \overline{P}_s^a(s')$ for all s' and $\Sigma_{s'} p_{s'} = 1$



- We can also solve this more directly by sorting
    - sort the values $x_{s'}$ into ascending order
    - for increasing values x_{s_i} assign the maximum possible value to p_{s_i}
    - which is bounded by 1 - (the sum of actual/min values for other p_{s_j})

Running example: IMDPs and robust VI

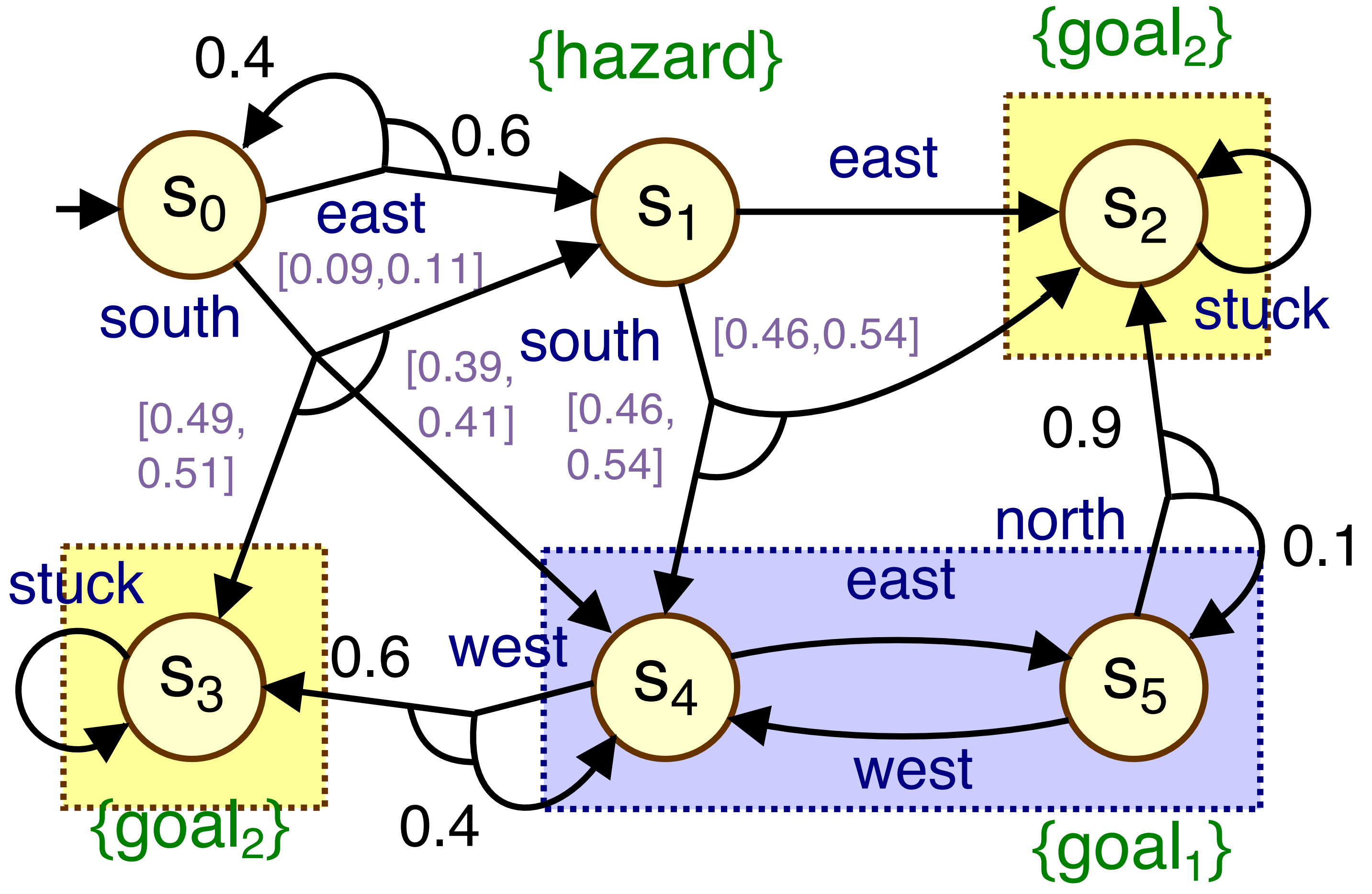
- Example: MaxProb(*goal*$_1$)



Running example: IMDPs and robust VI

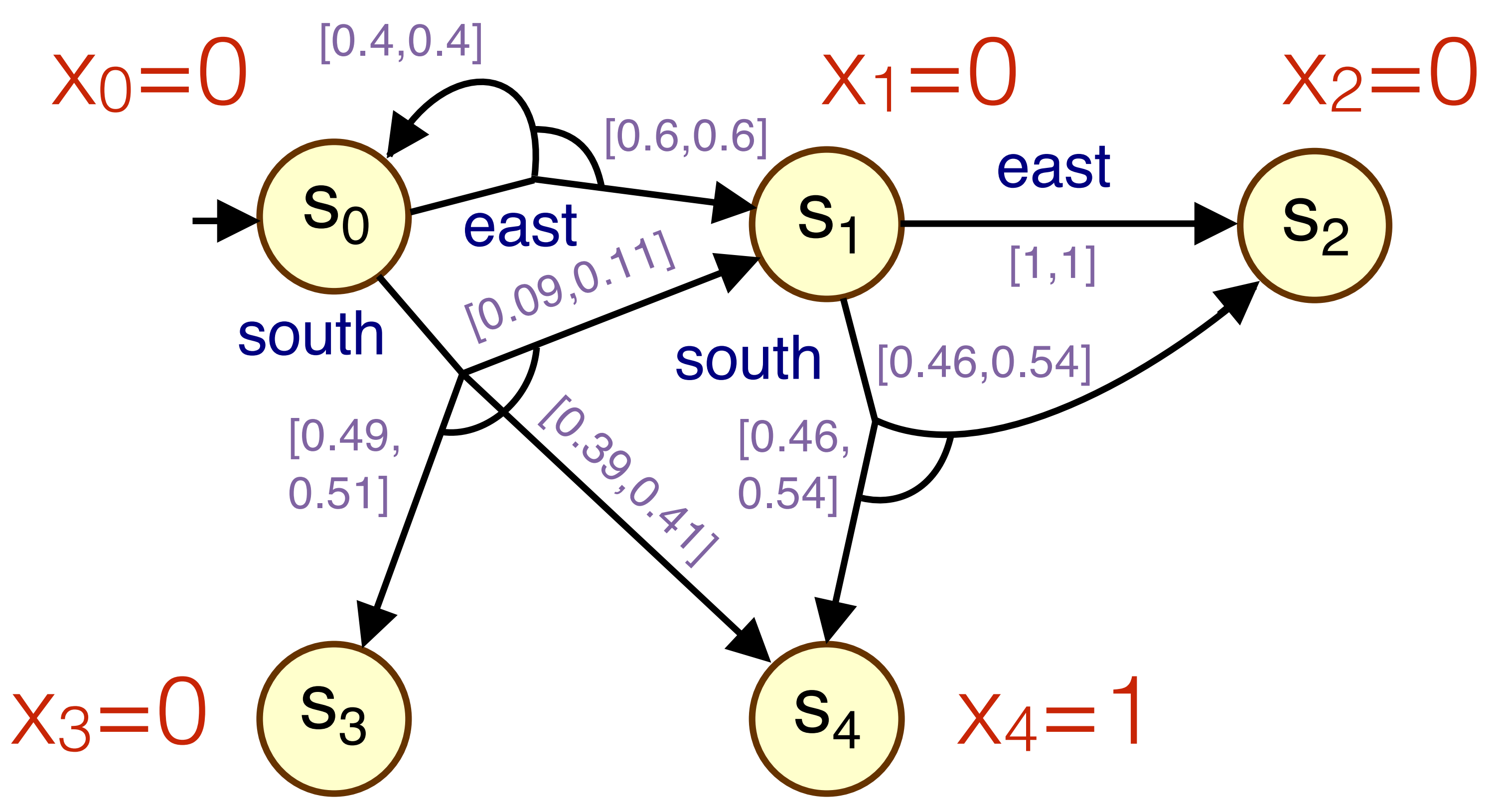
- Example: MaxProb(*goal*$_1$)

e=0.04



Running example: IMDPs and robust VI

- Example: MaxProb(*goal*$_1$)



- Fix $x_4=1$ and $x_2=x_3=0$, just solve for x_0, x_1
- Iteration k=0: $x_0=x_1=0$
- Iteration k=1:

$$x_0 := \max(\min(0\cdot 0.4 + 0\cdot 0.6), \min(0\cdot p_1 + 0\cdot p_3 + 1\cdot p_4))$$
$$= \max(0, 0.39)$$
$$= 0.39$$

$p_4 = 0.39, \ldots$

subject to:
$0.09 \le p_1 \le 0.11$
$0.49 \le p_3 \le 0.51$
$0.39 \le p_4 \le 0.41$
$p_1+p_3+p_4=1$

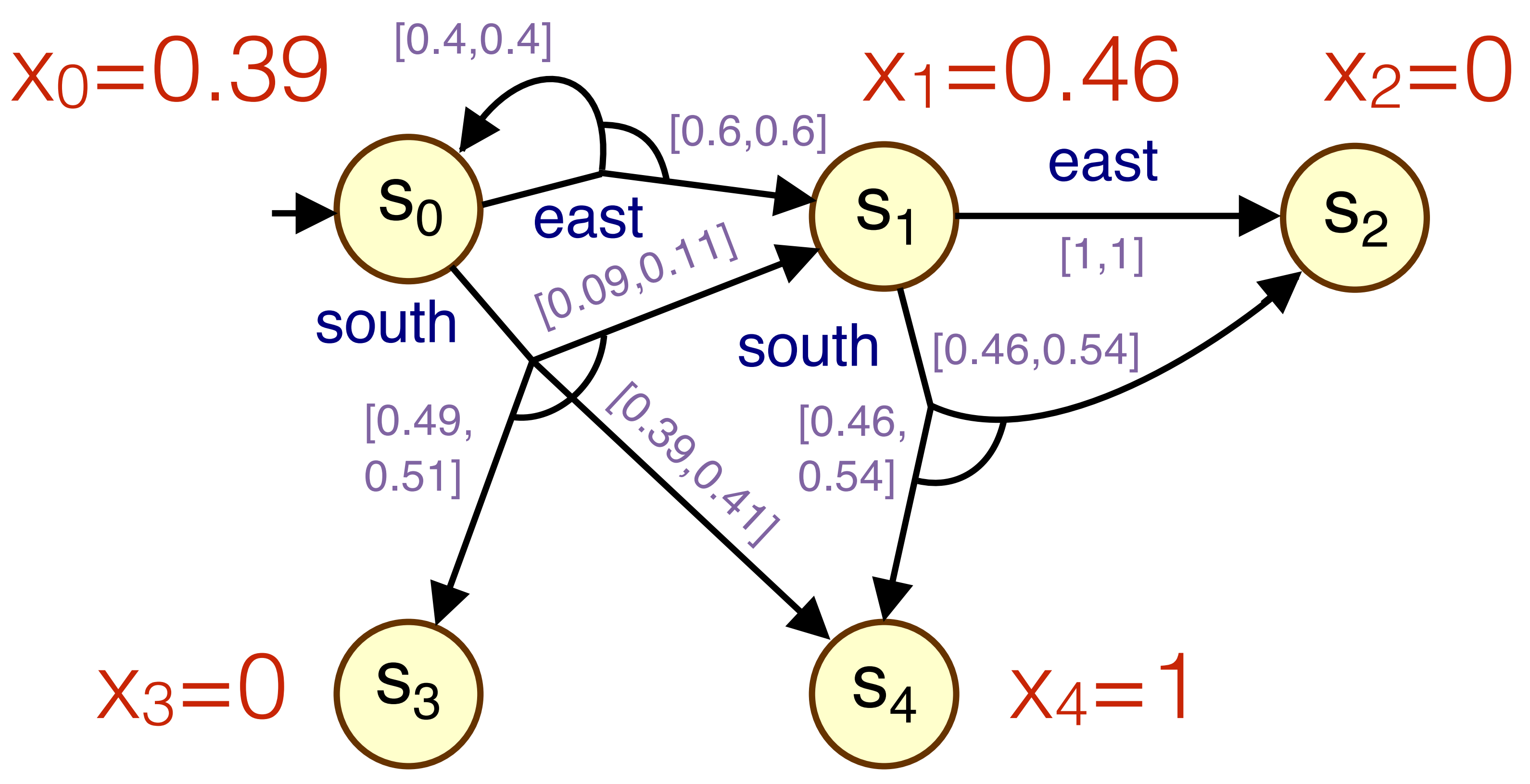
$$x_1 := \max(\min(0\cdot 1), \min(0\cdot p_2 + 1\cdot p_4))$$
$$= \max(0, 0.46)$$
$$= 0.46$$

$p_4 = 0.46, \ldots$

subject to:
$0.46 \le p_2 \le 0.54$
$0.46 \le p_4 \le 0.54$
$p_2+p_4=1$

Running example: IMDPs and robust VI

- Example: MaxProb(*goal*$_1$)



- Iteration k=2:

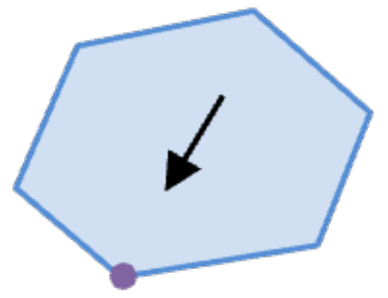
$$x_0 := \max(\min(0.39\cdot 0.4 + 0.46\cdot 0.6),\ \min(0.46\cdot p_1 + 0\cdot p_3 + 1\cdot p_4))$$

subject to:
$0.09 \le p_1 \le 0.11$
$0.49 \le p_3 \le 0.51$
$0.39 \le p_4 \le 0.41$
$p_1 + p_3 + p_4 = 1$

$$= \max(0.432,\ 0.436)$$
$$= 0.436$$

$x_3=0$
$x_1=0.46$
$x_4=1$

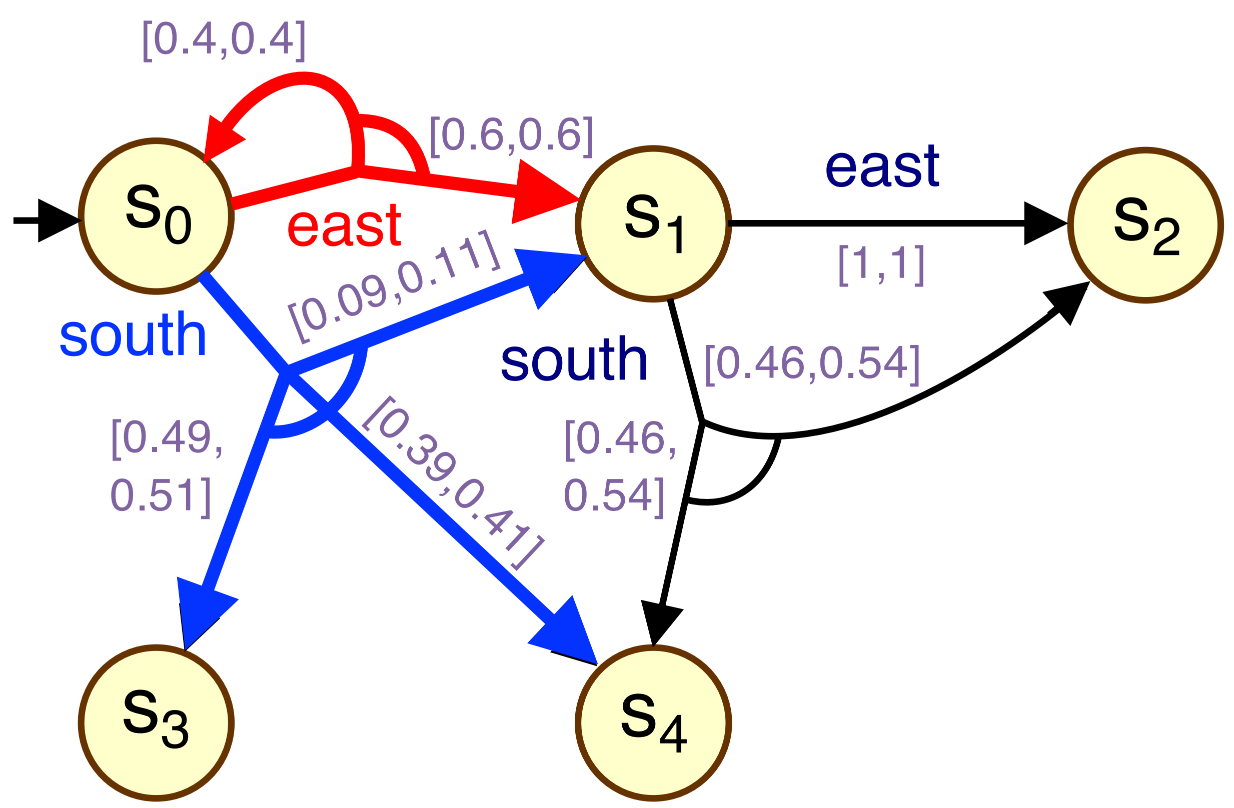
$p_3 = 0.51$
$p_1 = \min(0.11,\ 1-(0.51+0.39)) = 0.1$
$p_4 = 1-(0.51+0.1) = 0.39$

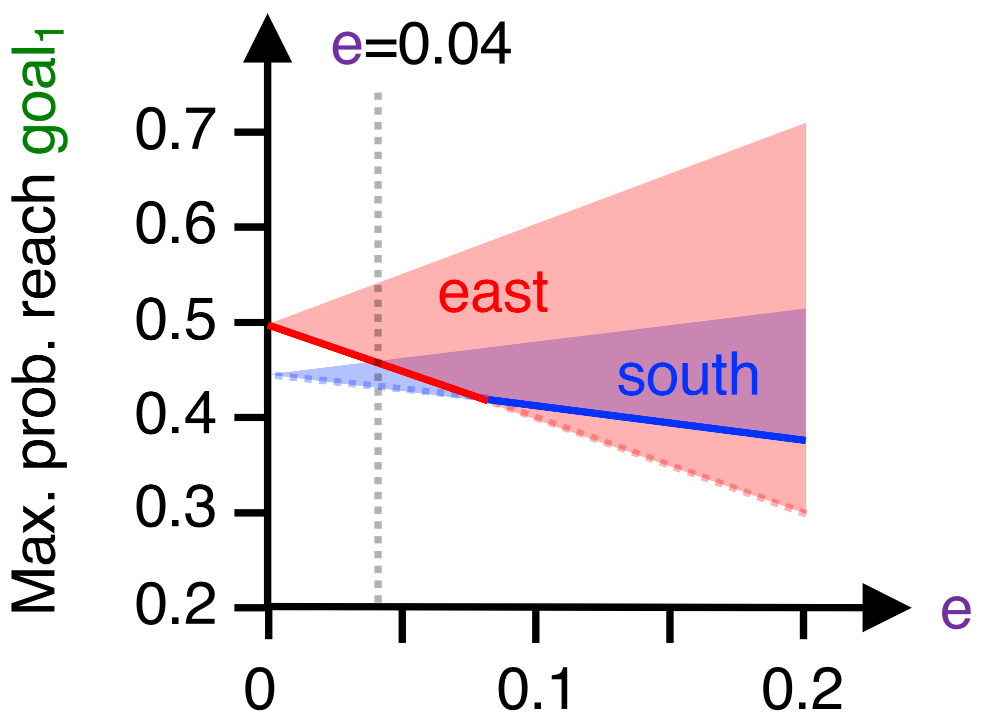


$x_1 := 0.46$ (as before)

Running example: IMDPs and robust VI

- Example: MaxProb(*goal*$_1$)





k	x_0	x_1
0	0	0
1	0.39	0.46
2	0.436	0.46
3	0.4504	0.46
4	0.45616	0.46
5	0.458464	0.46
6	0.4593856	0.46
7	0.45975424	0.46
8	0.459901696	0.46
9	0.4599606784	0.46
10	0.45998427136	0.46

- Iteration k=2:

x_0 := max(min($0.39 \cdot 0.4 + 0.46 \cdot 0.6$),
min($0.46 \cdot p_1 + 0 \cdot p_3 + 1 \cdot p_4$))

subject to:
$0.09 \leq p_1 \leq 0.11$
$0.49 \leq p_3 \leq 0.51$
$0.39 \leq p_4 \leq 0.41$
$p_1 + p_3 + p_4 = 1$

= max(0.432, 0.436)
= 0.436

$x_3=0$
$x_1=0.46$
$x_4=1$

$p_3 = 0.51$
$p_1 = \min(0.11,\ 1-(0.51+0.39)) = 0.1$
$p_4 = 1-(0.51+0.1) = 0.39$

x_1 := 0.46 (as before)

- Finally: $x_0=0.46$, $x_1=0.46$

Interval MDPs - so far…

- Robust control is computationally efficient (robust value iteration)
    - (s,a)-rectangular and inner problem is easy to solve
    - another possibility not discussed here: convex optimisation [Puggelli et al.'13]
- For MaxProb (and SSP), optimal policies are memoryless (and deterministic)
    - so computed policies are optimal worst case with respect to static uncertainty

What about objectives that need memory? (e.g. finite horizon, or temporal logic)

- Intervals are a simple, natural way to model transition probability uncertainty

How do we generate the intervals?

Are there better models of uncertainty sets?

Policies with memory

- Quantifying over memoryless environment policies
  - gives us worst-case behaviour over static uncertainty

$$V^{\Pi,\mathcal{T}_m}(s) = \max_{\pi\in\Pi} \min_{\tau_m\in\mathcal{T}_m} V^{\pi,\tau_m}(s) = \max_{\pi\in\Pi} \min_{P\in\mathcal{P}} V^{\pi,P}(s)$$

- But for objectives that require non-memoryless control policies
  - computation methods typically also assume non-memoryless environment policies

$$V^{\Pi,\mathcal{T}}(s) = \max_{\pi\in\Pi} \min_{\tau_m\in\mathcal{T}} V^{\pi,\tau_m}(s)$$

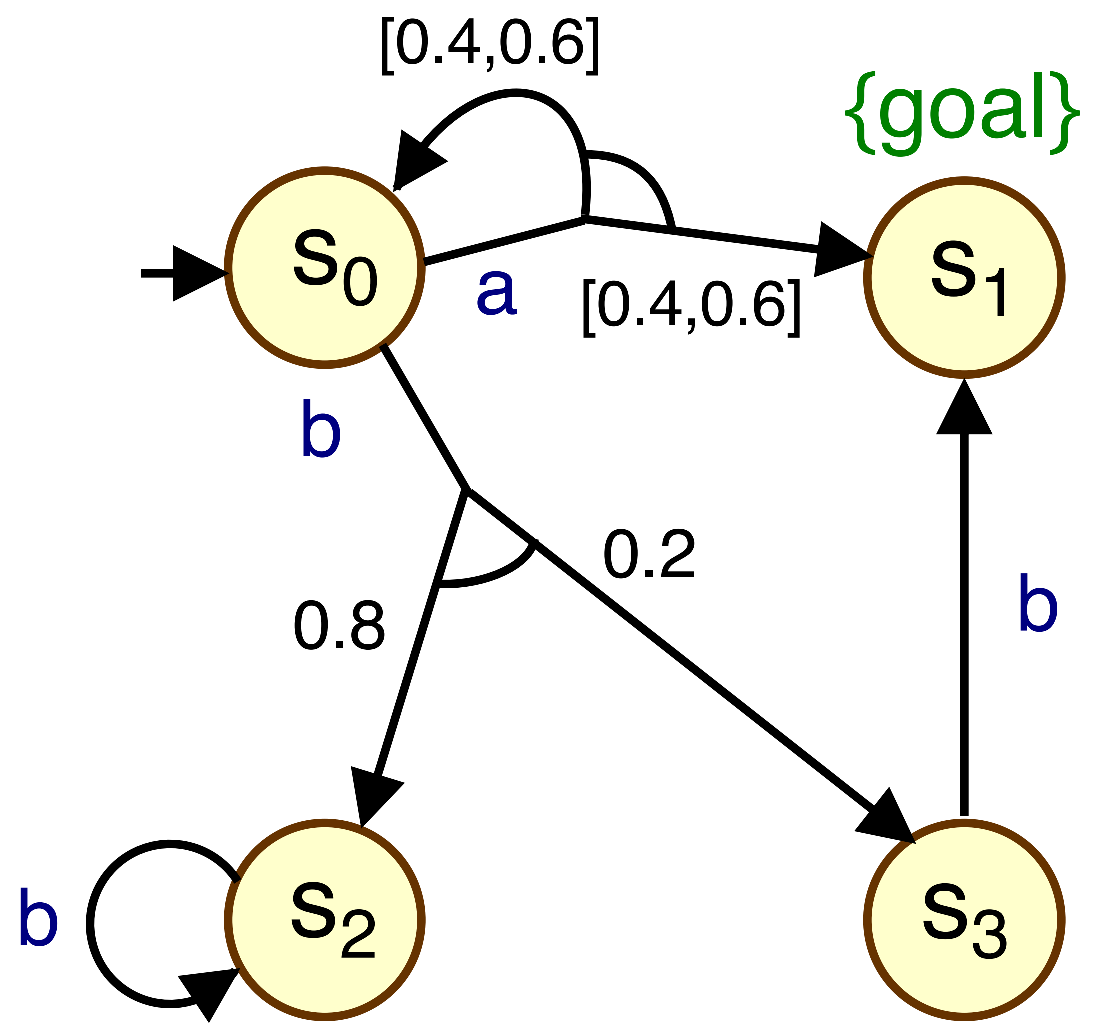
  - i.e., worst-case behaviour over dynamic uncertainty
  - which is often (but not always) unrealistic (depends on time-scales)

- This however gives a conservative bound over static uncertainty

$$V^{\Pi,\mathcal{T}}(s) \leq \max_{\pi\in\Pi} \min_{P\in\mathcal{P}} V^{\pi,P}(s)$$

Memory (time dependencies)

- Objective: MaxProb$^{=2}$(goal), i.e., get to goal in <u>exactly</u> 2 steps
  - so we need time-dependent strategies for the controller
  - computable via k steps of value iteration



- Worst-case probabilities (time-dependent environment strategies)
  - "b,b" 0.2 (optimal)
  - "a,b": 0
  - "a,a": $\min\{p_1(1-p_2)\ :\ p_1,p_2 \in [0.4,0.6]\} = 0.4 \cdot (1-0.6) = 0.16$ (too conservative)

from value iteration; dynamic uncertainty; maybe unrealistic

- Worst-case probabilities (memoryless environment strategies)
  - "b,b": 0.2
  - "a,b": 0
  - "a,a": $\min\{p(1-p)\ :\ p \in [0.4,0.6]\} = 0.4 \cdot (1-0.4) = 0.24$ (better bound) (now optimal)

static uncertainty; may be more realistic; hard to compute

Memory (temporal logic objectives)

- Temporal logic (in particular LTL) allows more complex objectives, e.g.:
  - $P_{max=?}$ [(G¬hazard) ∧ (GF $goal_1$)] - “maximise probability of avoiding hazard and also visiting goal 1 infinitely often”
  - $P_{max=?}$ [¬$zone_3$ U ($zone_1$ ∧ (F $zone_4$))] - “maximise probability of patrolling zone 1 (whilst avoiding zone 3) then zone 4”

- For MDPs, we generate optimal policies by:
  - converting the LTL formula to a deterministic automaton
  - building a product of the MDP and the automaton
  - optimising a simpler objective (e.g. MaxProb) on the product MDP

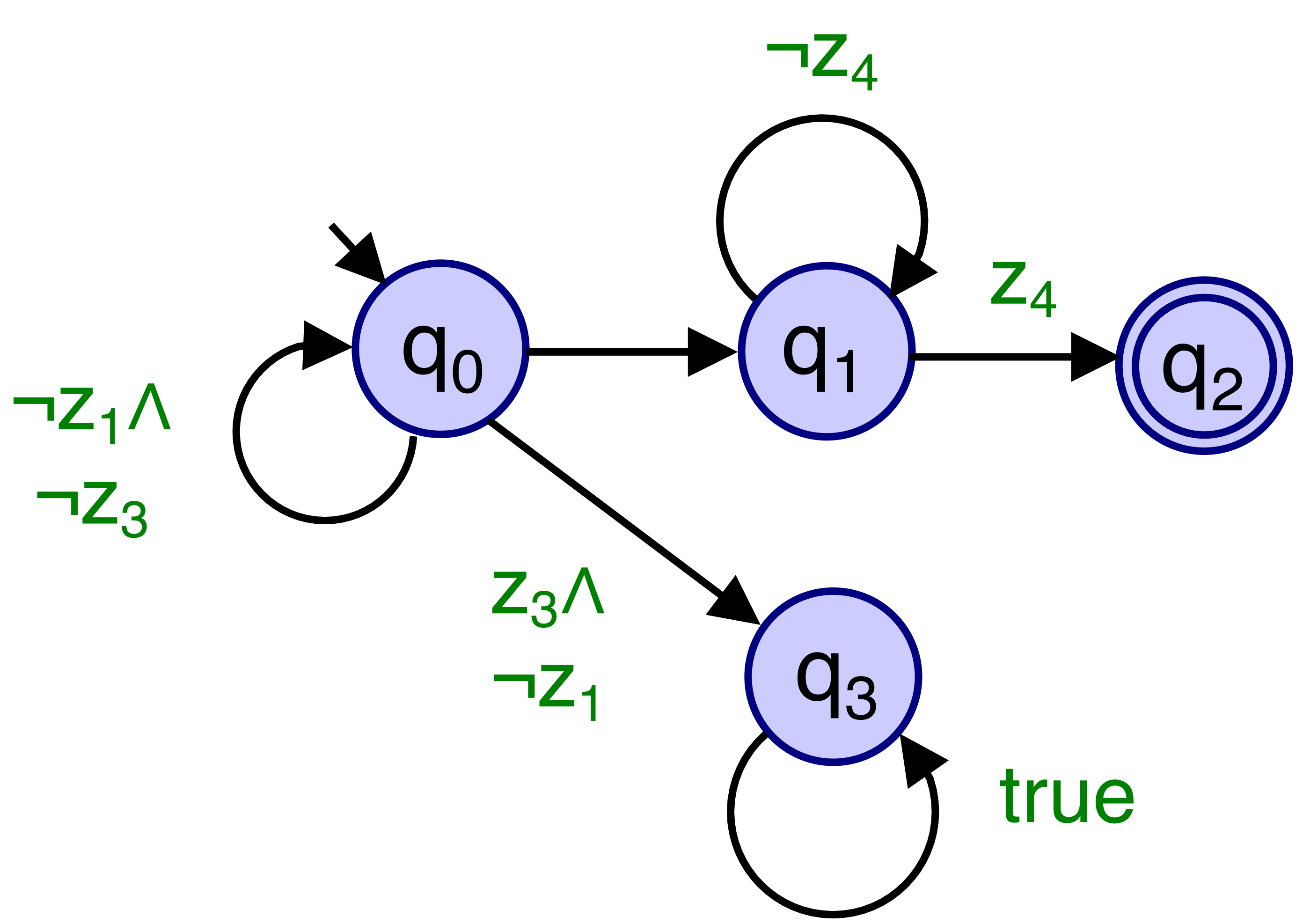
- The techniques extend to uMDPs/IMDPs [Wolff et al.’12]
  - but (like for MDPs), optimal policies need memory

Automata for LTL objectives

- For co-safe LTL (satisfaction occurs in finite time), we use finite automata

¬zone$_3$ U (zone$_1$ ∧ (F zone$_4$))

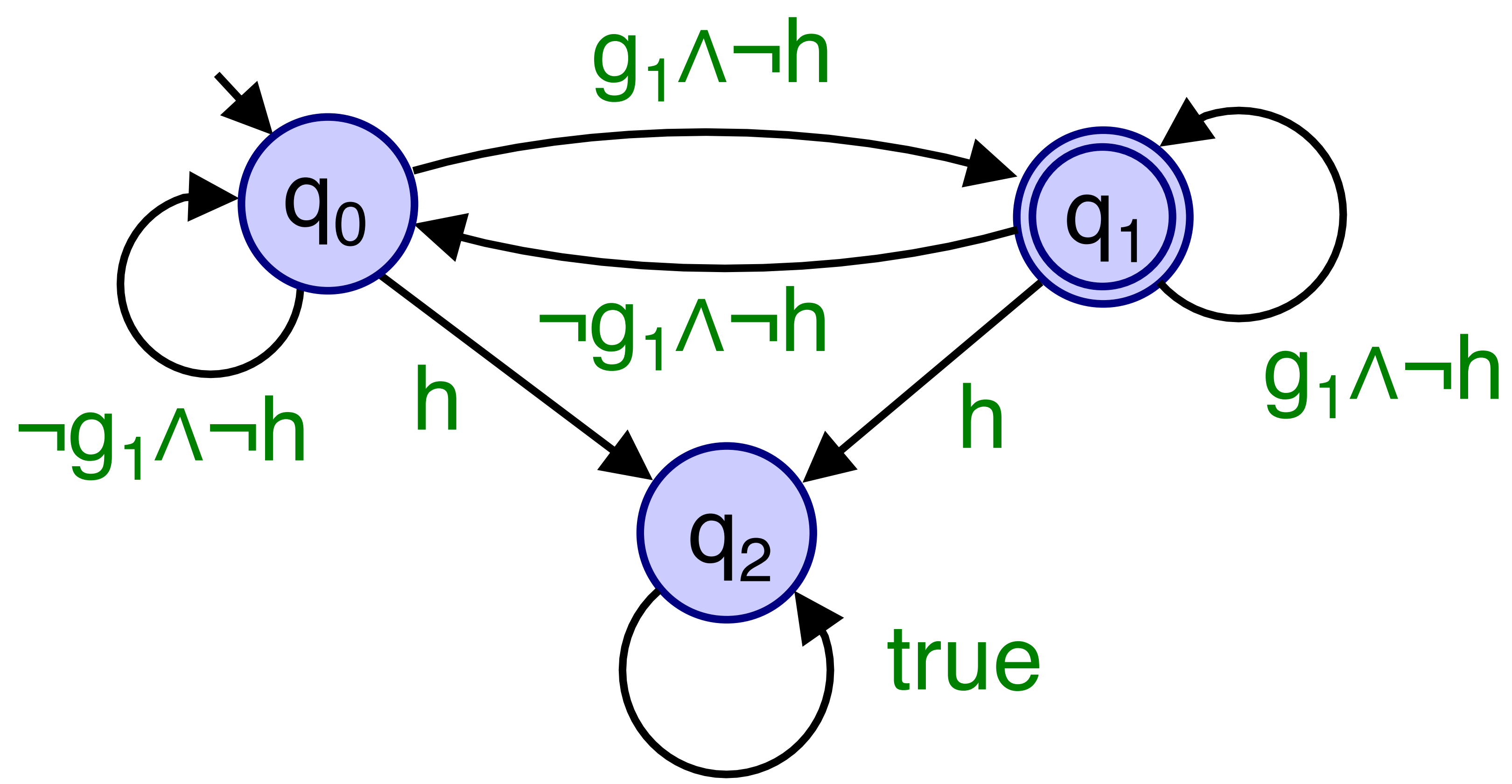
(avoiding hazard and also visiting goal 1 infinitely often)



- For general LTL, we use e.g. Rabin automata

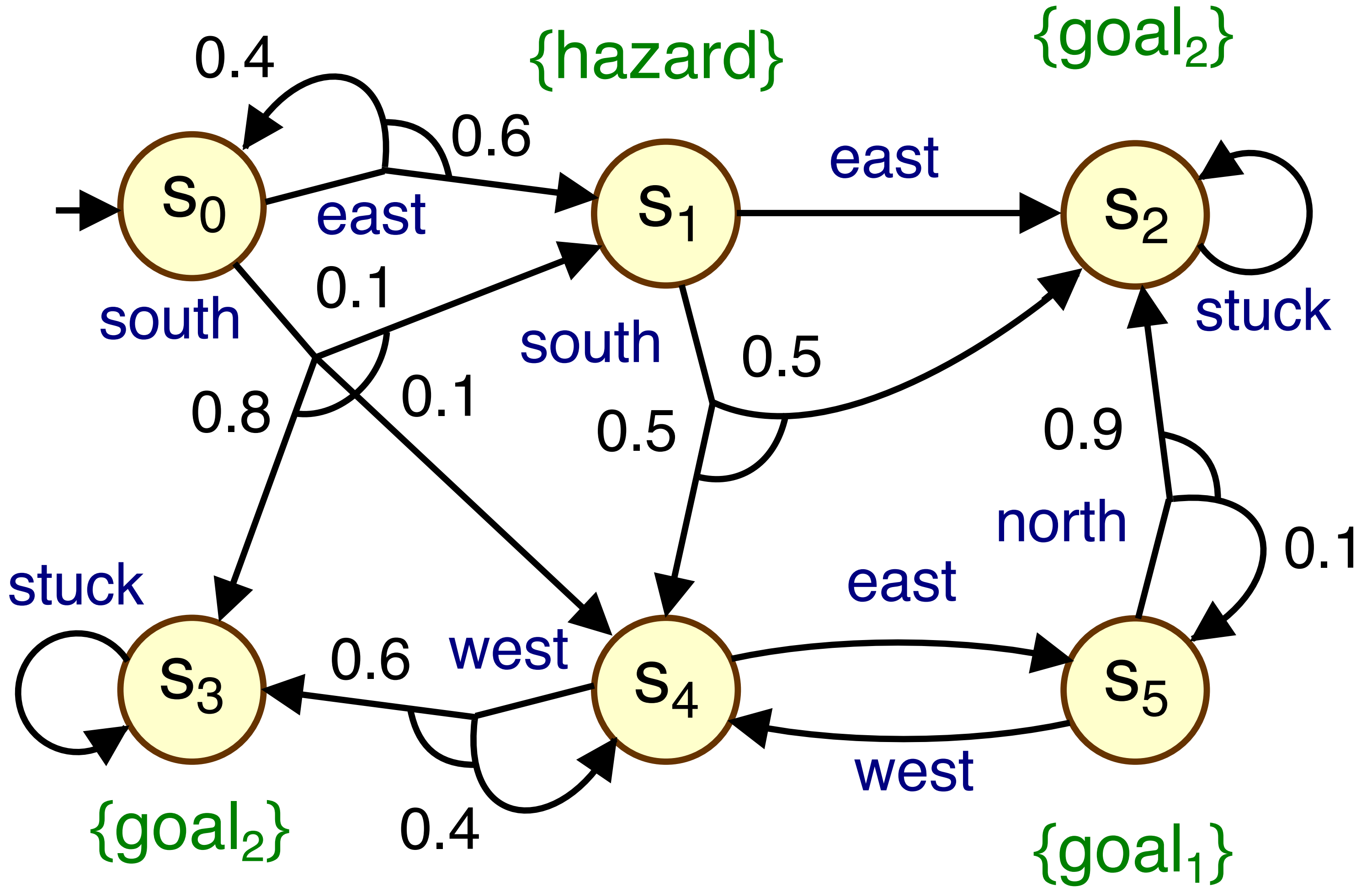
(G¬hazard) ∧ (GF goal$_1$)

(visit zone 1 (whilst avoiding zone 3) then zone 4)

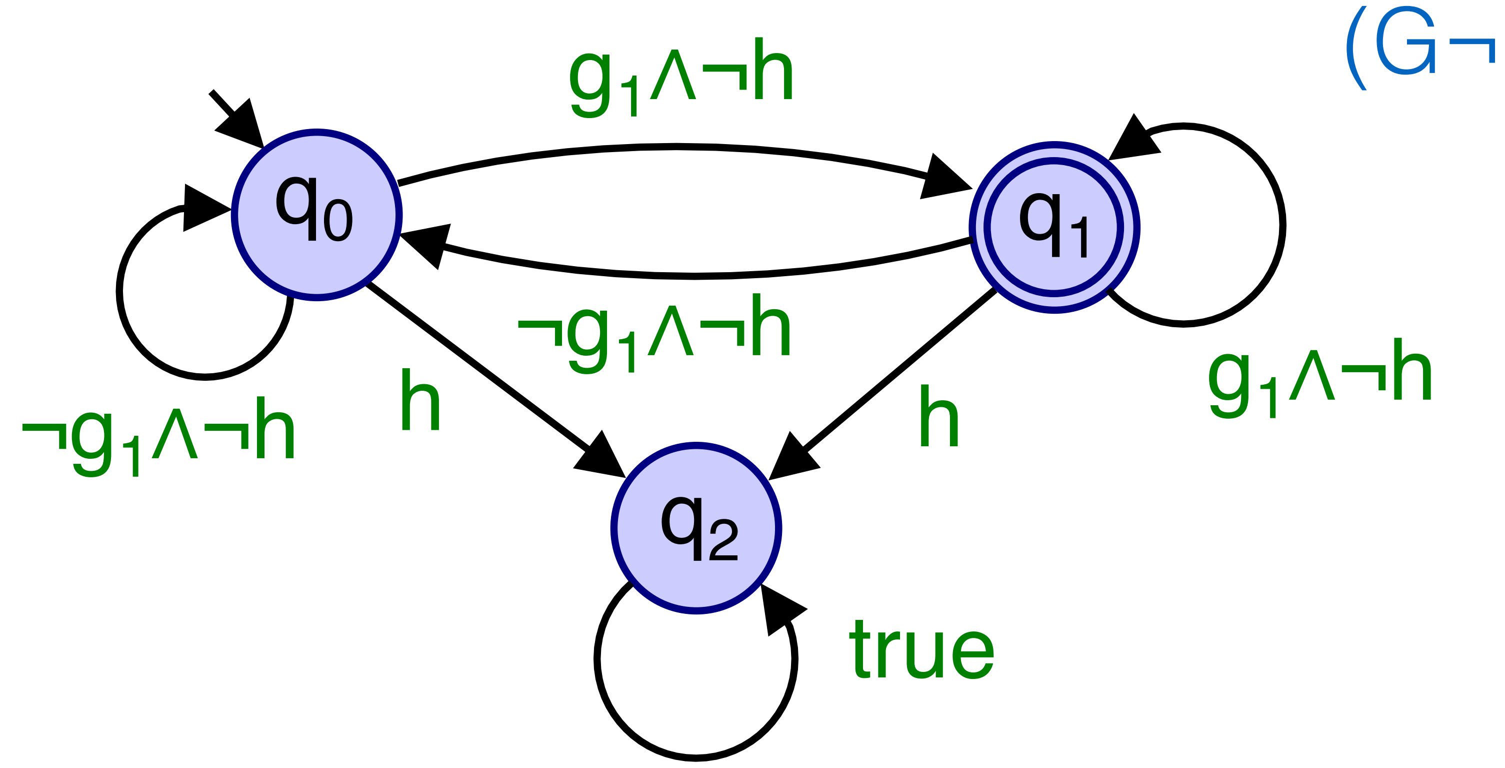


Optimising for LTL on a product MDP

MDP $\mathcal{M}$

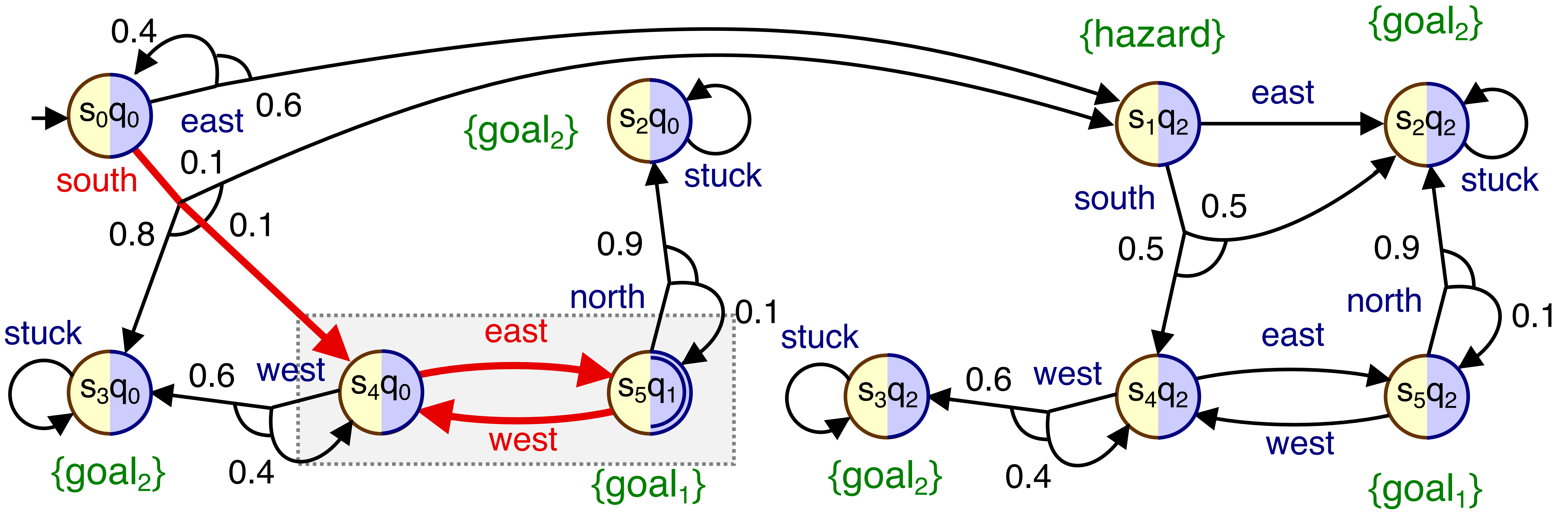


Automaton $\mathcal{A}$ for
(G¬hazard) ∧ (GF goal$_1$)



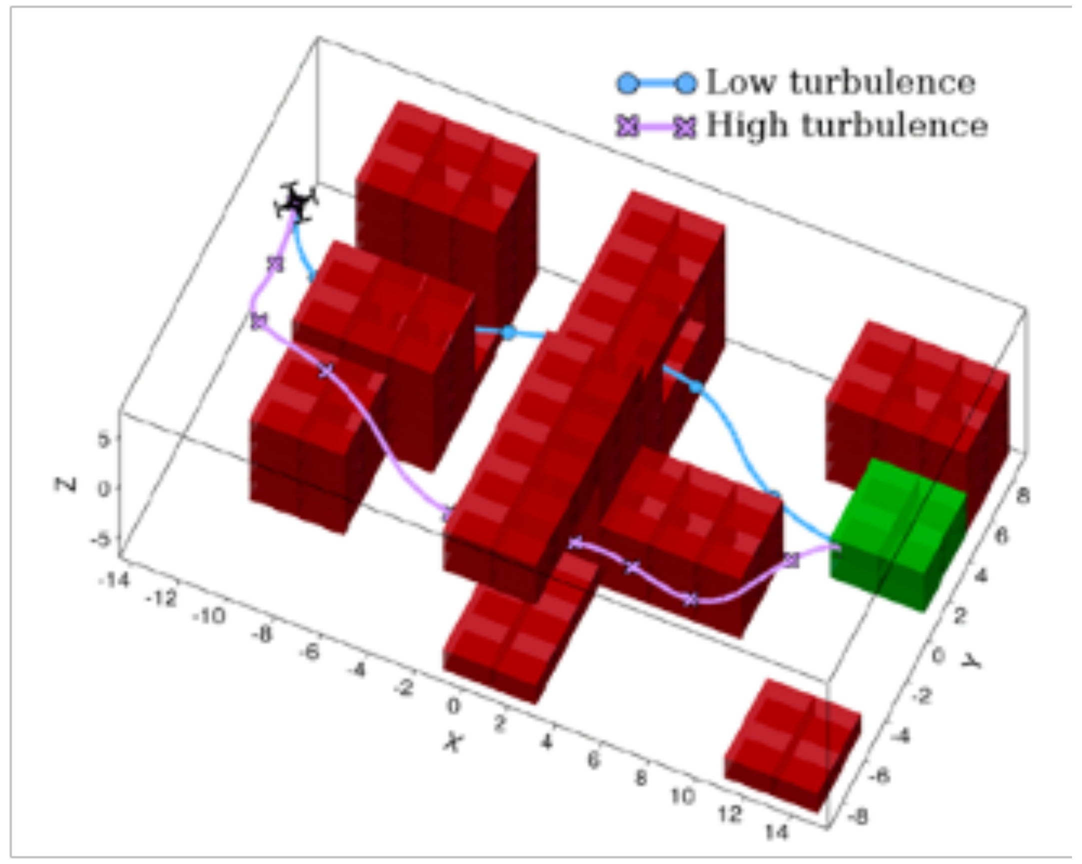
Product MDP $\mathcal{M} \otimes \mathcal{A}$

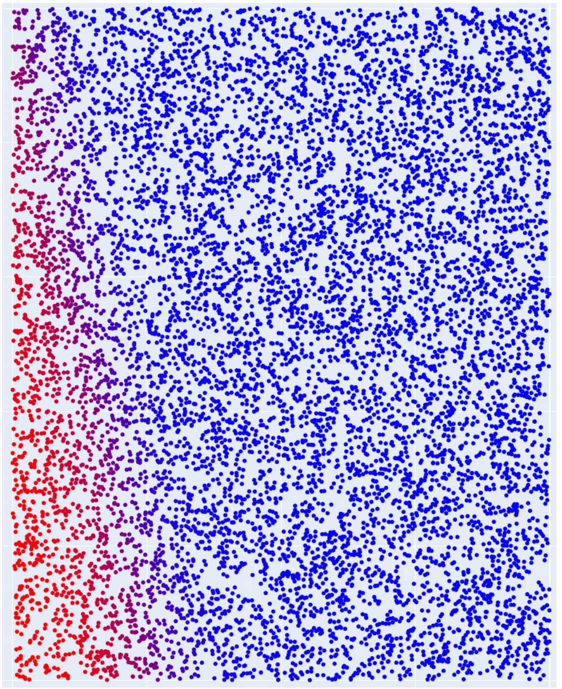
Optimal memoryless policy of $\mathcal{M} \otimes \mathcal{A}$ corresponds to finite-memory optimal policy of MDP $\mathcal{M}$

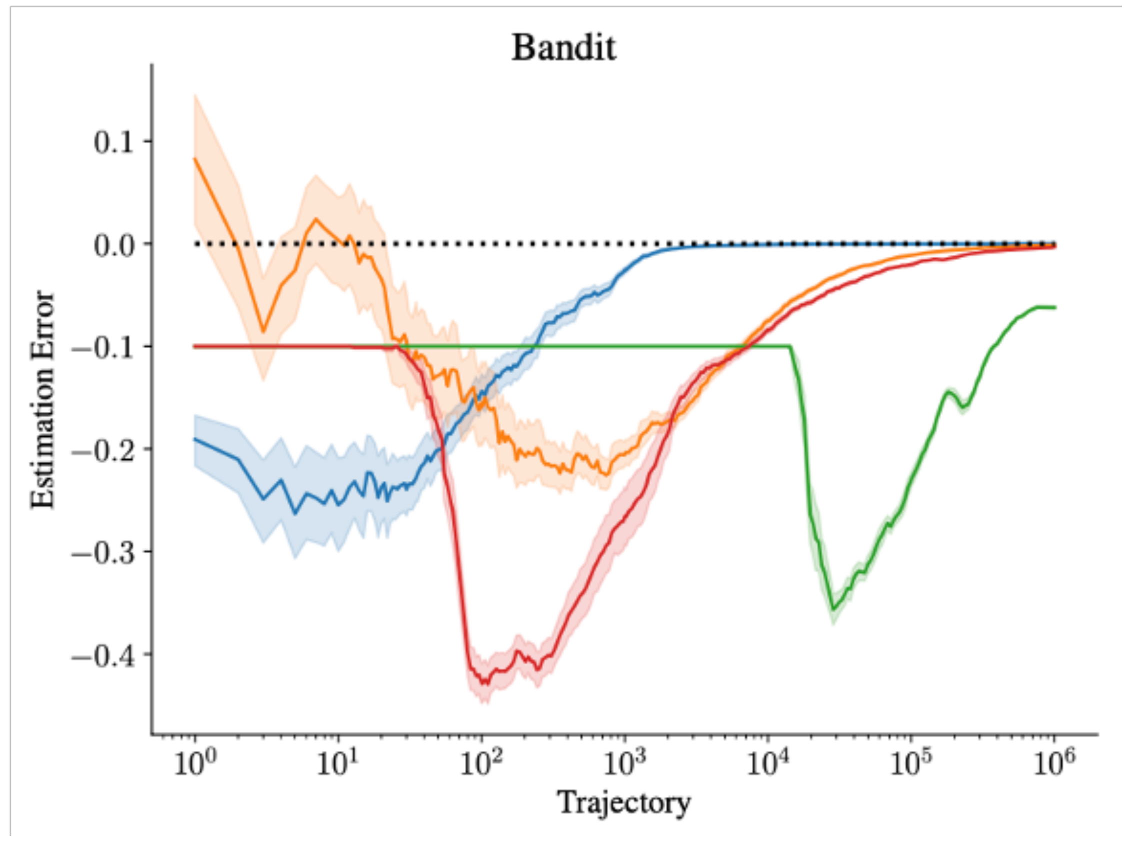


Generating IMDP intervals

- Some examples of IMDP generation







- Unmanned aerial vehicle
  - robust control in turbulence
  - continuous-space dynamical model with unknown noise
  - discrete abstraction + finite "scenarios" of sampled noise yields IMDP abstraction

[Badings et al.'23]

- Deep reinforcement learning
  - worst-case analysis of abstractions of probabilistic policies for neural networks
  - intervals between IMDP abstract states constructed by sampling the policy

[Bacci&Parker'20]

- Robust anytime MDP learning
  - sampled MDP trajectories
  - IMDPs constructed and solved periodically to yield robust predictions on current model
  - PAC or Bayesian interval learning

[Suilen et al.'22]

Learning IMDP intervals

- One approach: sampling from the (fixed, but unknown) "true" MDP
  - generate sample paths and keep separate counts of transition frequencies

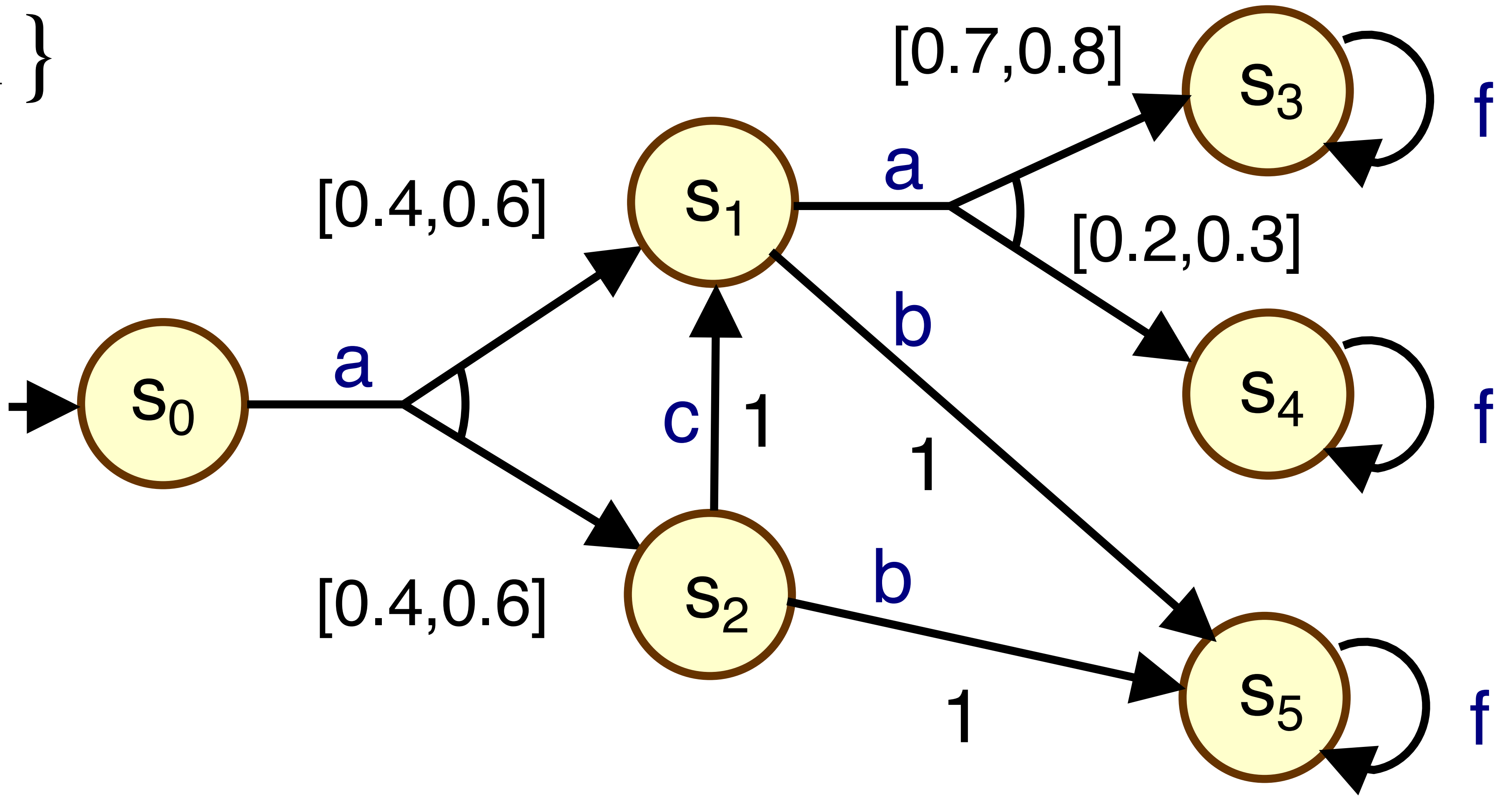
- Gives confidence intervals around point estimates for transition probabilities $P_s^a(s_i)$
  - using probably approximately correct (PAC) guarantees
  - we fix an error rate γ and compute an error δ
  - standard method of maximum a-posteriori probability (MAP) estimation
    to infer point estimates of probabilities

- For each state s, we have sample counts $N = \#(s, a)$ and $k_i = \#(s, a, s_i)$
  - point estimate of the transition probability $P_s^a(s_i)$ is: $\tilde{P}_s^a(s_i) \approx k_i / N$
  - confidence interval for the transition probability: $\tilde{P}_s^a(s_i) \pm \delta$ where $\delta = \sqrt{\log(2/\gamma)/2N}$
  - then we have: $Pr(P_s^a(s_i) \in \tilde{P}_s^a(s_i) \pm \delta) \geq 1 - \gamma$ (via Hoeffding's inequality)

Learning IMDP intervals

- If desired, we can lift the PAC guarantee from individual transitions to the uMDP

- Distribute the chosen error rate γ across all transitions:
  - $\gamma_P = \gamma / (\Sigma(s,a) \in S \times A \,|\, Succ_{>1}(s,a) \,|\,)$
  - where $Succ_{>1}(s,a) = \{s \in S : 0 < P_s^a(s') < 1\}$ is the set of successor states of each (s,a) with more than one successor



- To construct the IMDP, we use:
  - $\underline{P}_s^a(s_i) = \max(\varepsilon, \tilde{P}_s^a(s_i) - \delta_P)$
  - $\overline{P}_s^a(s_i) = \min(\tilde{P}_s^a(s_i) + \delta_P, 1)$

- Then we have: $Pr(P \in \mathscr{P}) \geq 1 - \gamma$

[Suilen et al.'22]

Likelihood uncertainty sets

[Nilim&Ghaoui'05]

- Likelihood models suit experimentally determined transition probabilities
  - and are less conservative than interval representations
- Uncertainty sets are :
  - are derived from empirical frequencies $F_s^a(s')$ of a transition to s' after action a in state s
  - are described by likelihood regions: $\mathscr{P}_s^a = \{P_s^a \in Dist(S) \mid \sum_{s'} F_s^a(s') \log(P_s^a(s')) \geq \beta_s^a)\}$
  - where β_s^a is the uncertainty level (can be estimated for a desired confidence level)
  - $\beta_s^a < \beta_{s,\max}^a$ where $\beta_{s,\max}^a = \sum_{s'} F_s^a(s') \log(F_s^a(s'))$ is the optimal log-likelihood
- Inner optimisation problems

$$\inf_{P_s^a \in \mathscr{P}_s^a} \sum_{s' \in S} P_s^a(s') \cdot x_{s'}$$

  - can be solved (approximately) using a bisection algorithm
  - to within an accuracy δ in time $O(\log(x_{\max}/\delta))$ where $x_{\max}$ is the maximum value in vector x

Uncertainty set models - Summary

- Intervals & likelihood models
  - both quite computationally tractable and statistically meaningful
  - interval models are more conservative (sometimes projected to as an estimate)

- Finite scenarios ("sampled"): $\mathscr{P}_s^a = \{P_{s,1}^a, \ldots, P_{s,k}^a\}$
  - inner optimisation is simple (min over finite set)
  - but worst-case choice can be very conservative

$$\inf_{P_s^a \in \mathscr{P}_s^a} \sum_{s' \in S} P_s^a(s') \cdot x_{s'}$$

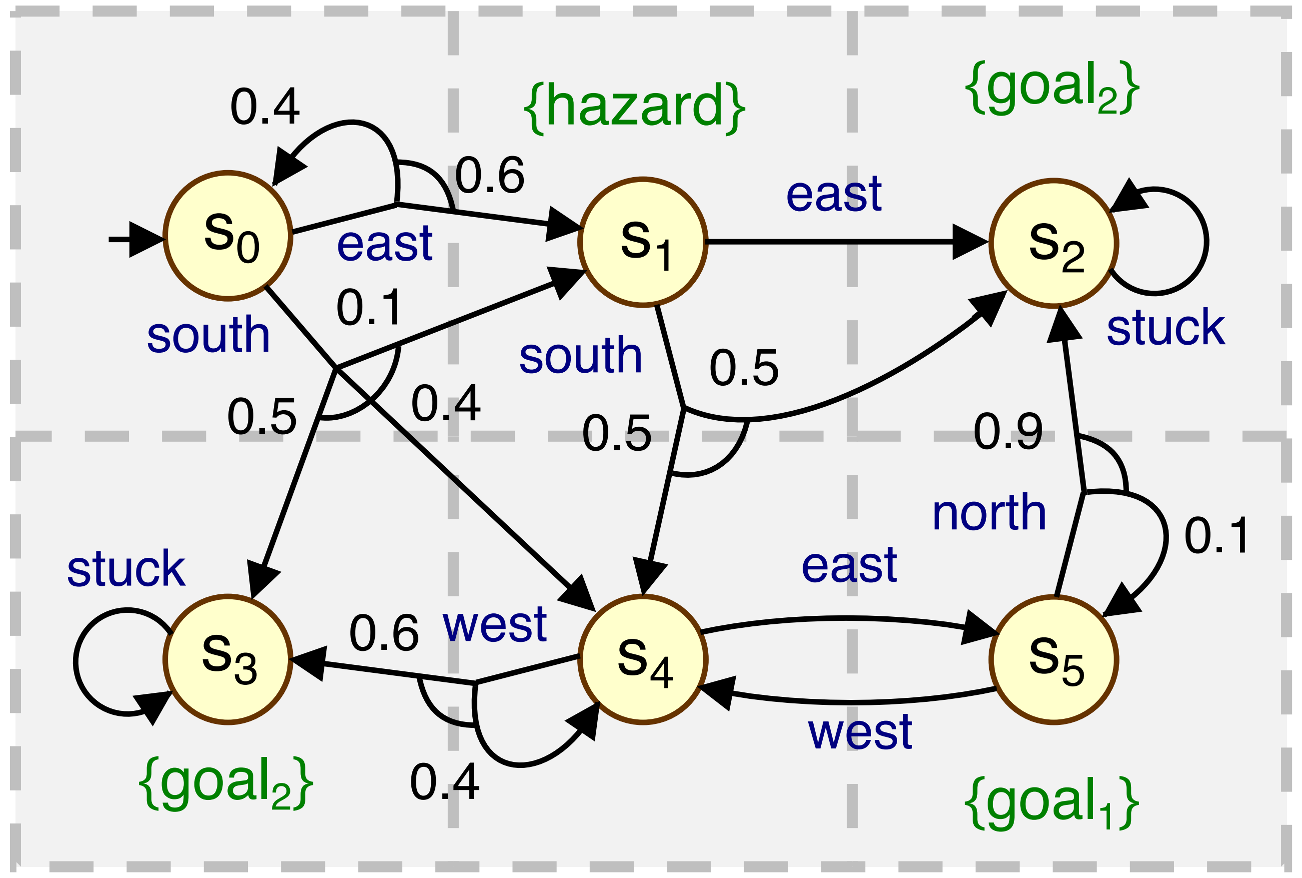
- Many other possibilities, e.g.:
  - maximum a posteriori models, entropy models, ellipsoidal models, …
  - most have similar (approximate) optimisation approaches to likelihood models
  - see: [Nilim&Ghaoui'05] for details

Tool support: PRISM

- PRISM: probabilistic model checking tool
  - formal modelling and analysis (using temporal logic properties) of:
    - Markov chains, Markov decision processes,
    - interval Markov chains, interval Markov decision processes,
    - stochastic games (via PRISM-games), and much more…

PRISM

- See: www.prismmodelchecker.org
  - download, documentation, tutorials, papers, case studies, …



- Supporting files for ESSAI examples here:

  www.prismmodelchecker.org/courses/essai23/

Summary (lecture 3)

- Uncertain MDPs
  - environment policies - static vs dynamic uncertainty
  - robust value iteration (robust dynamic programming)
  - implementation with interval MDPs (IMDPs)
  - non-memoryless policies (static uncertainty)
  - generating / learning intervals
  - uncertainty set representations
  - tool support: PRISM

Advertisement

- ERC-funded project FUN2MODEL, based at Oxford
  - lead by Marta Kwiatkowska
  - model-based reasoning for learning and uncertainty

- Postdoc position available now
  - http://www.fun2model.org/
  - http://www.prismmodelchecker.org/news.php

European Research Council

Established by the European Commission

Email: david.parker@cs.ox.ac.uk

marta.kwiatkowska@cs.ox.ac.uk

References

- Applications & challenges

  - T. Badings, L. Romao, A. Abate, D. Parker, H. A. Poonawala, M. Stoelinga and N. Jansen, Robust Control for Dynamical Systems with Non-Gaussian Noise via Formal Abstractions, *Journal of Artificial Intelligence Research*, 76, pages 341-391, 2023
  - M. Budd, P. Duckworth, N. Hawes and B. Lacerda, Bayesian Reinforcement Learning for Single-Episode Missions in Partially Unknown Environments, In CoRL, 2022
  - C. Costen, M. Rigter, B. Lacerda and N. Hawes, Shared Autonomy Systems with Stochastic Operator Models, IJCAI'22, 4614-4620, 2022

- Markov decision processes

  - Mausam & A. Kolobov, *Planning with Markov Decision Processes: An AI Perspective*, Morgan & Claypool, 2012
  - M. Puterman, *Markov Decision Processes*, Wiley, 1994

References

- Stochastic games

  - J. Filar and K. Vrieze, *Competitive Markov Decision Processes*, Springer, 1997
  - M. Kwiatkowska, G. Norman and D. Parker, *Probabilistic Model Checking and Autonomy,* Annual Review of Control, Robotics, and Autonomous Systems, 5, 2022

- Uncertain MDPs and interval MDPs

  - G. N. Iyengar, Robust dynamic programming, *Mathematics of Operations Research*, 30(2), 2005
  - A. Nilim and L. Ghaoui, Robust control of Markov decision processes with uncertain transition matrices, *Operations Research*, 53(5), 780–798, 2005
  - E. Wolff, U. Topcu, and R. Murray, Robust control of uncertain Markov decision processes with temporal logic specifications, In *Proc. 51th IEEE Conference on Decision and Control (CDC'12)*, 2012
  - W. Wiesemann, D. Kuhn and B. Rustem, Robust Markov Decision Processes, *Math. Oper. Res.,* 38(1), 153-183, 2013
  - A. Puggelli, W. Li, A. Sangiovanni-Vincentelli and S. Seshia, Polynomial-time verification of PCTL properties of MDPs with convex uncertainties, In *Proc. 25th International Conference on Computer Aided Verification (CAV'13)*, LNCS, vol. 8044, Springer, 2013

References

- Learning and using IMDPs

  - T. Badings, L. Romao, A. Abate, D. Parker, H. A. Poonawala, M. Stoelinga and N. Jansen, Robust Control for Dynamical Systems with Non-Gaussian Noise via Formal Abstractions, *Journal of Artificial Intelligence Research*, 76, pages 341-391, 2023
  - E. Bacci and D. Parker, Verified Probabilistic Policies for Deep Reinforcement Learning, In *Proc. 14th International Symposium NASA Formal Methods (NFM'22)*, volume 13260 of LNCS, pages 193-212, Springer, 2022
  - M. Suilen, T. D. Simão, N. Jansen and D. Parker, Robust Anytime Learning of Markov Decision Processes, In *Proc. 36th Annual Conference on Neural Information Processing Systems (NeurIPS'22)*, 2022