Register at: essai.si

ESSAI & ACAI 2023
LJUBLJANA, SLOVENIA

MODEL UNCERTAINTY IN SEQUENTIAL DECISION MAKING

DAVID PARKER
University of Oxford

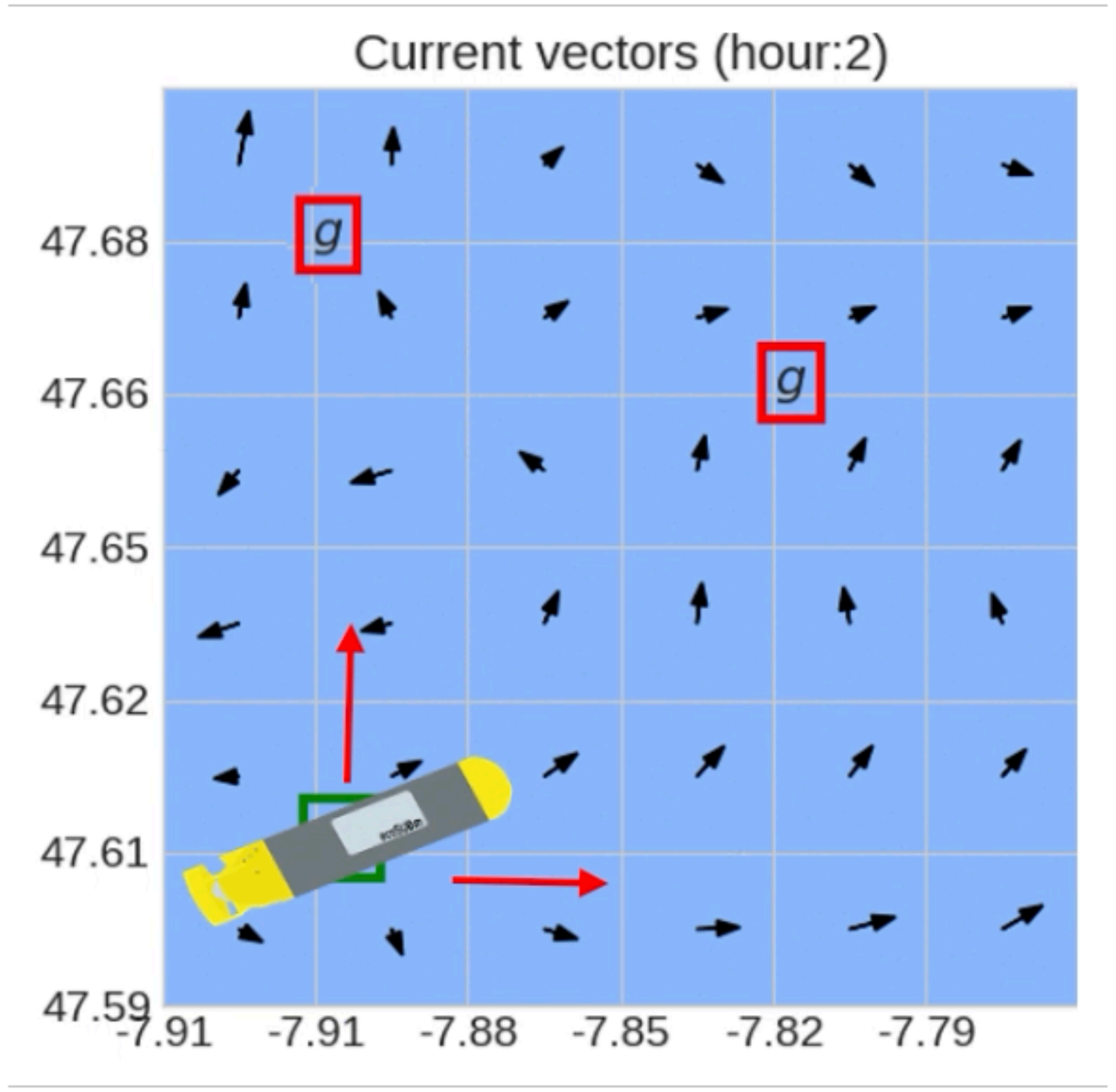
BRUNO LACERDA
University of Oxford

NICK HAWES
University of Oxford

Introduction

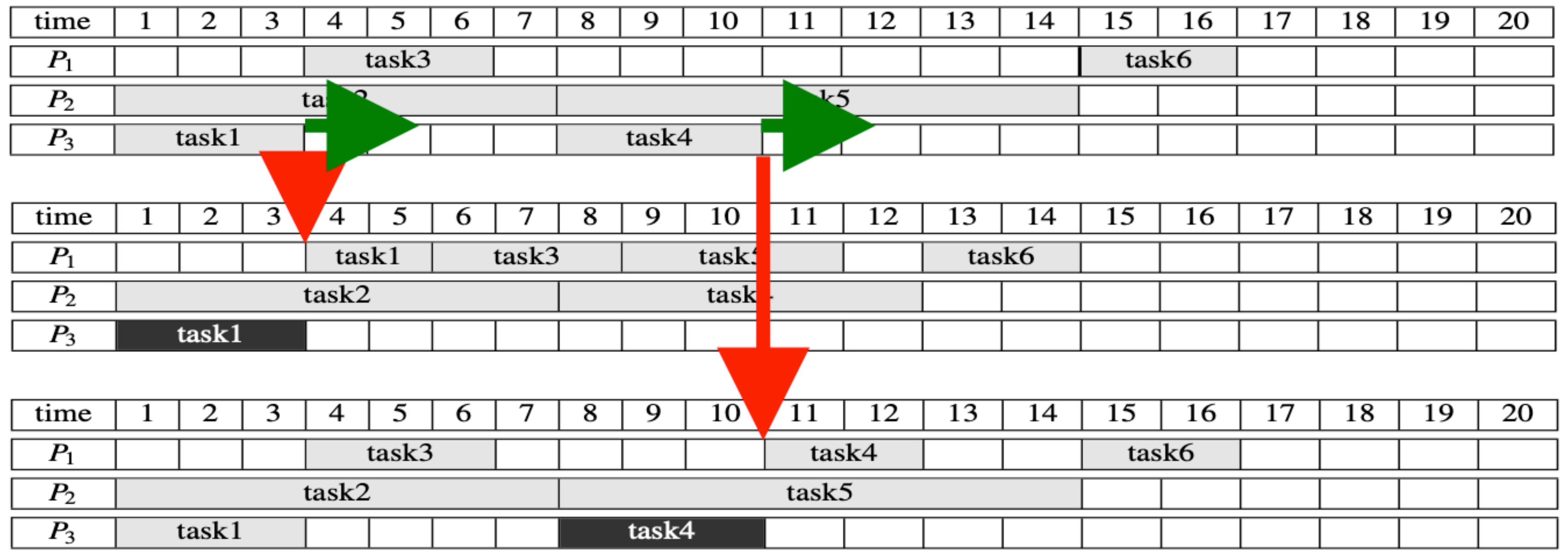
Sequential decision making under uncertainty

- 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, …



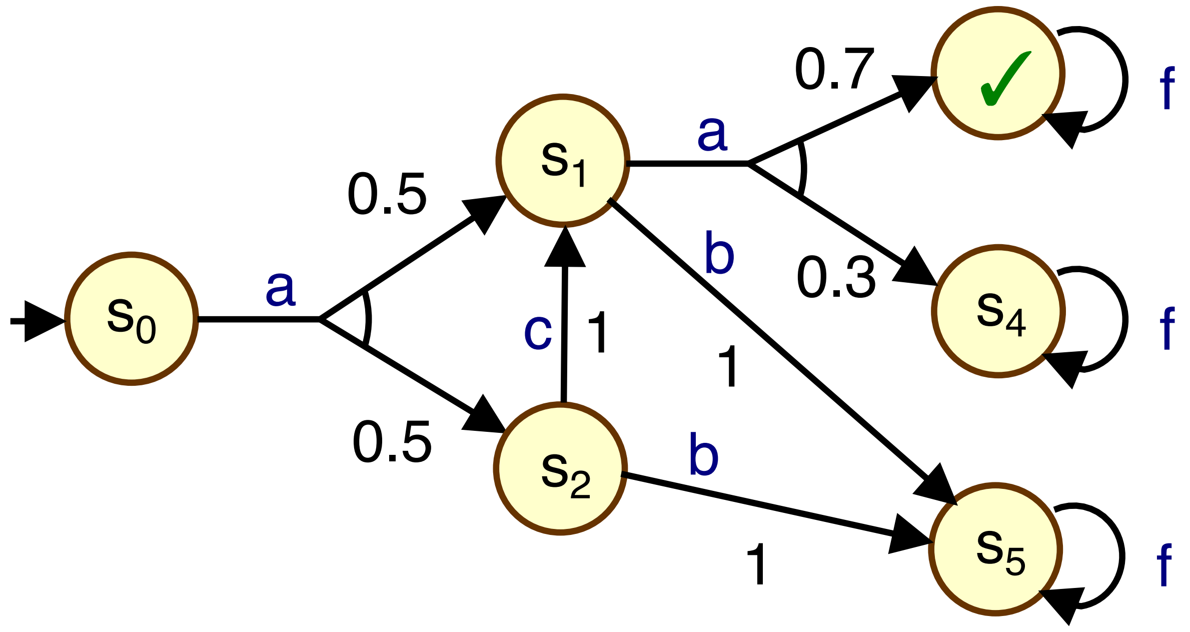
- Sequential decision making under uncertainty
  - noisy sensors, unpredictable conditions, lossy communication, human behaviour, hardware failures, …

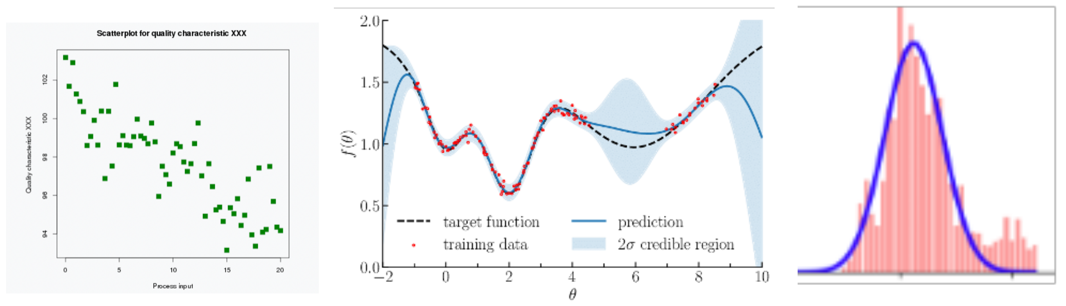
- Trustworthy, safe and robust decision making
  - e.g. for safety-critical applications
  - needs rigorous/systematic quantification of uncertainty



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

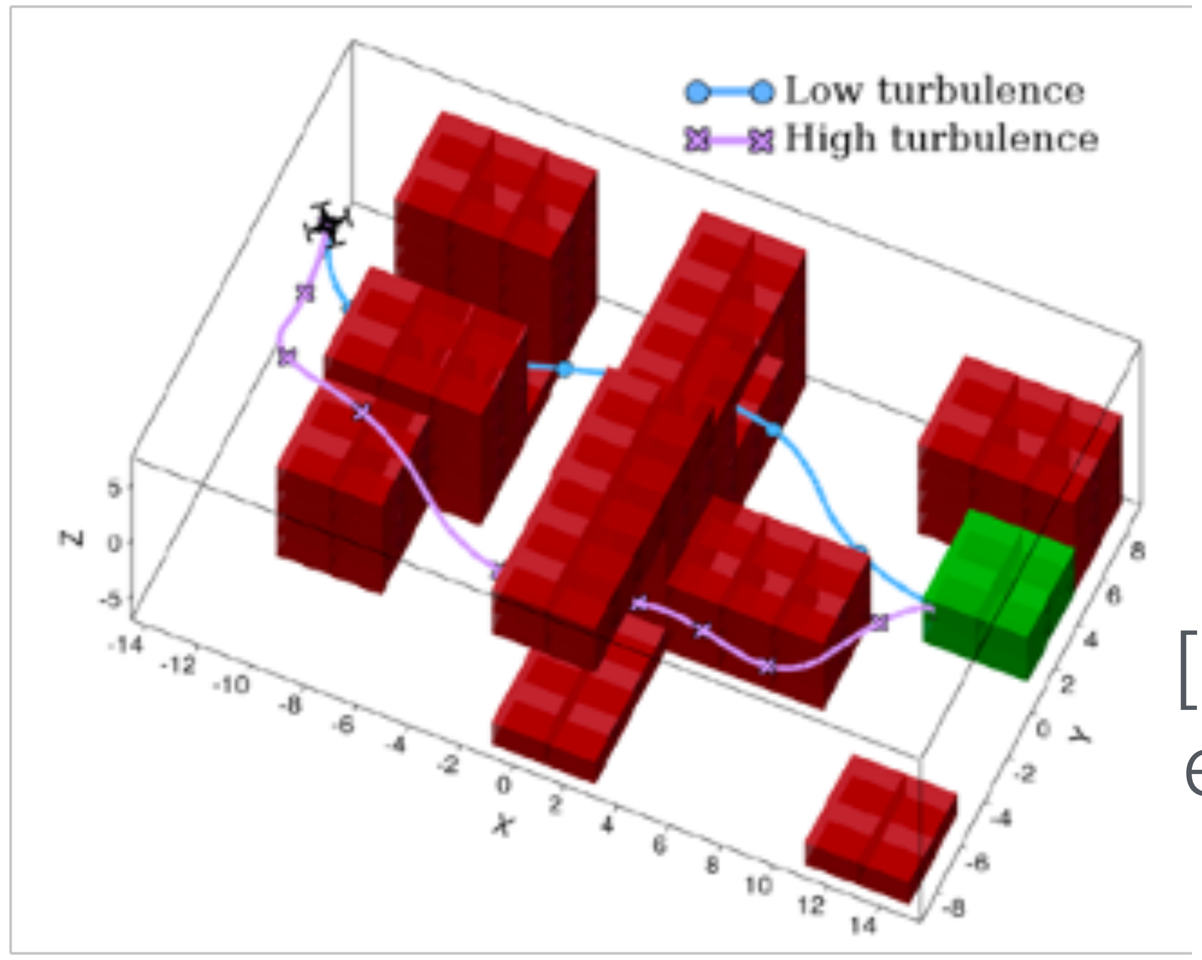
- Epistemic uncertainty (quantifies lack of knowledge)
  - reducible: can reduce by collecting more data/observations
  - e.g., poor model quality due to low number of measurements

(2)

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

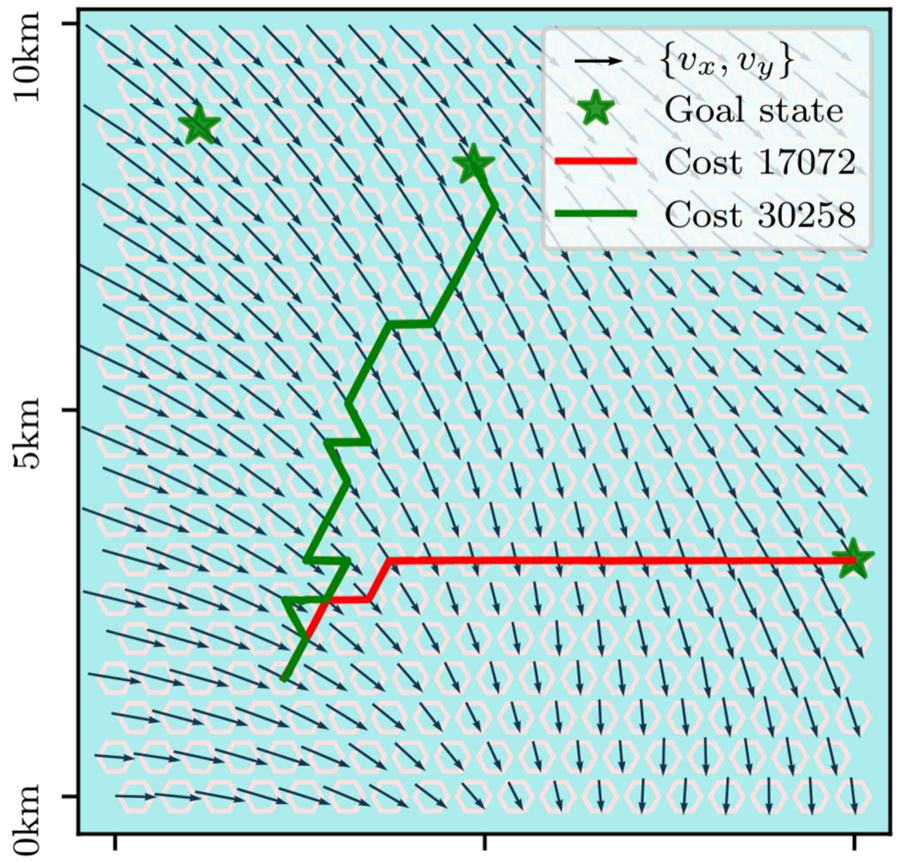
Applications & challenges

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



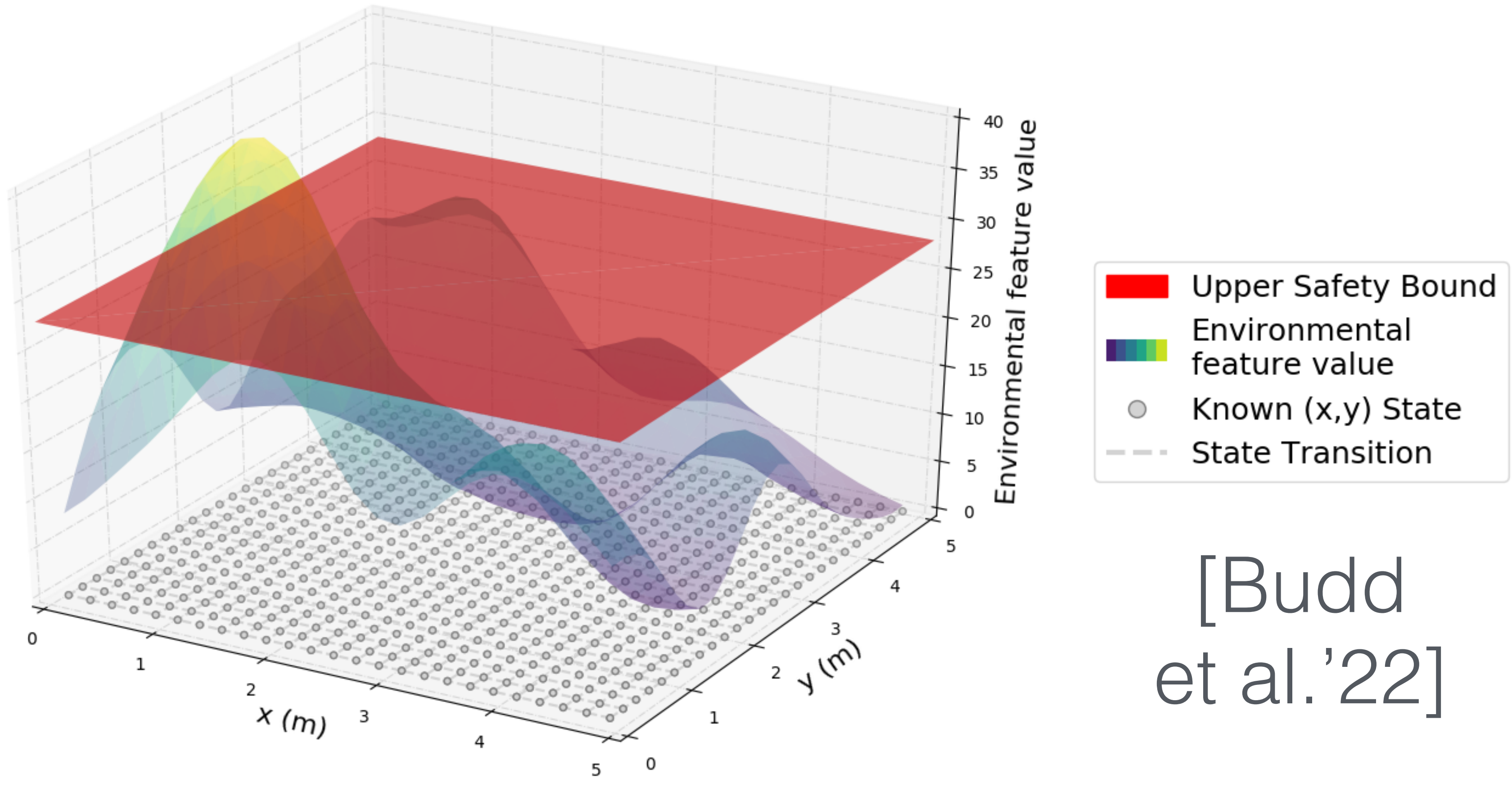
[Badings et al.'23]

- Autonomous underwater vehicle
  - effective navigation against unknown ocean currents



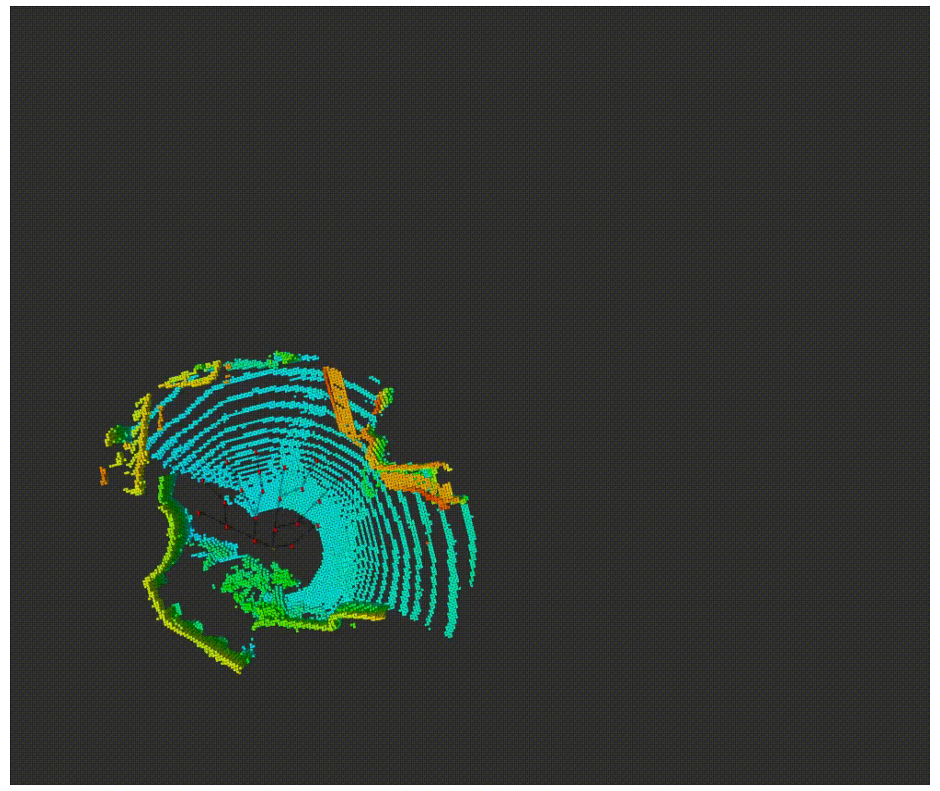
[Budd et al.'22]

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



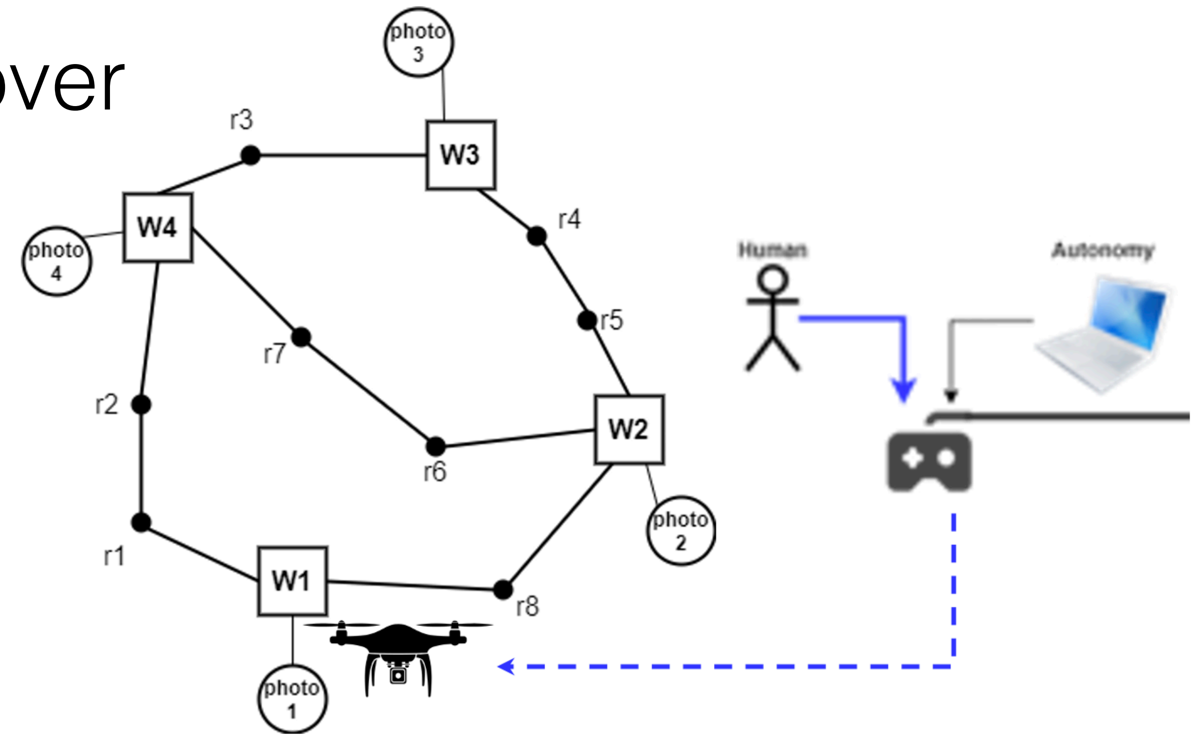
[Budd et al.'22]

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



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

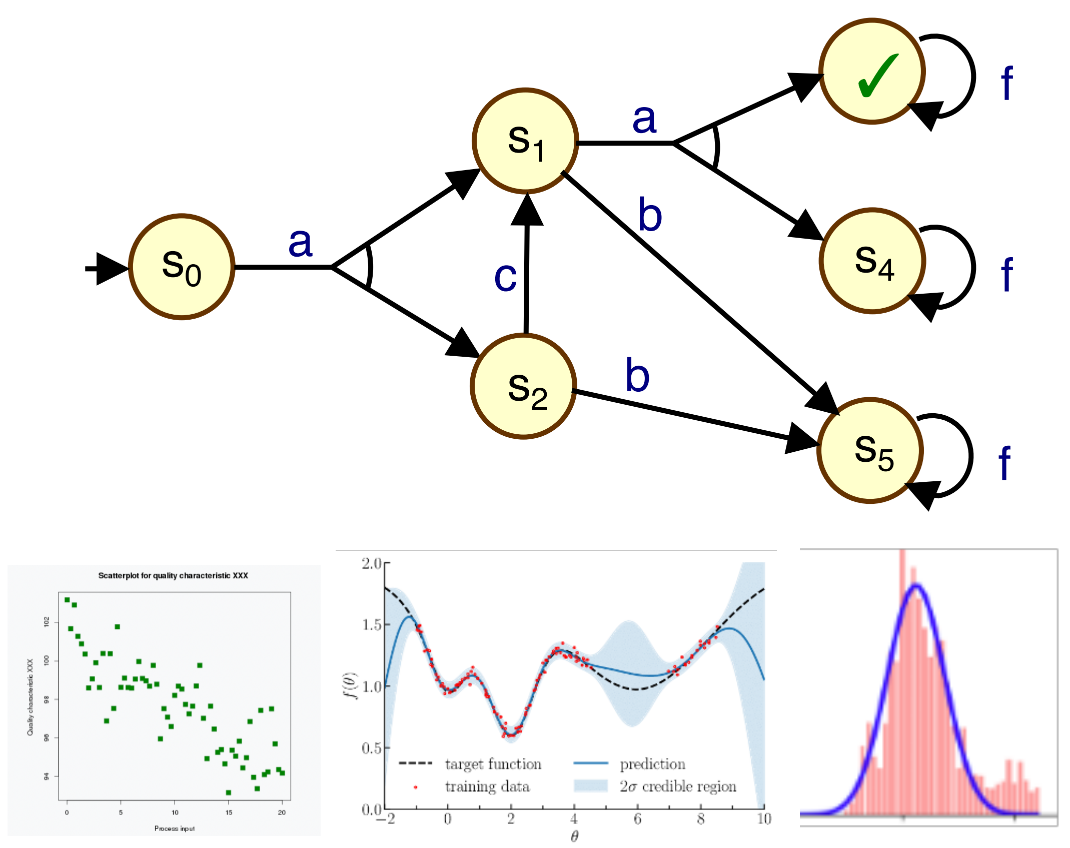
[Costen et al.'22]



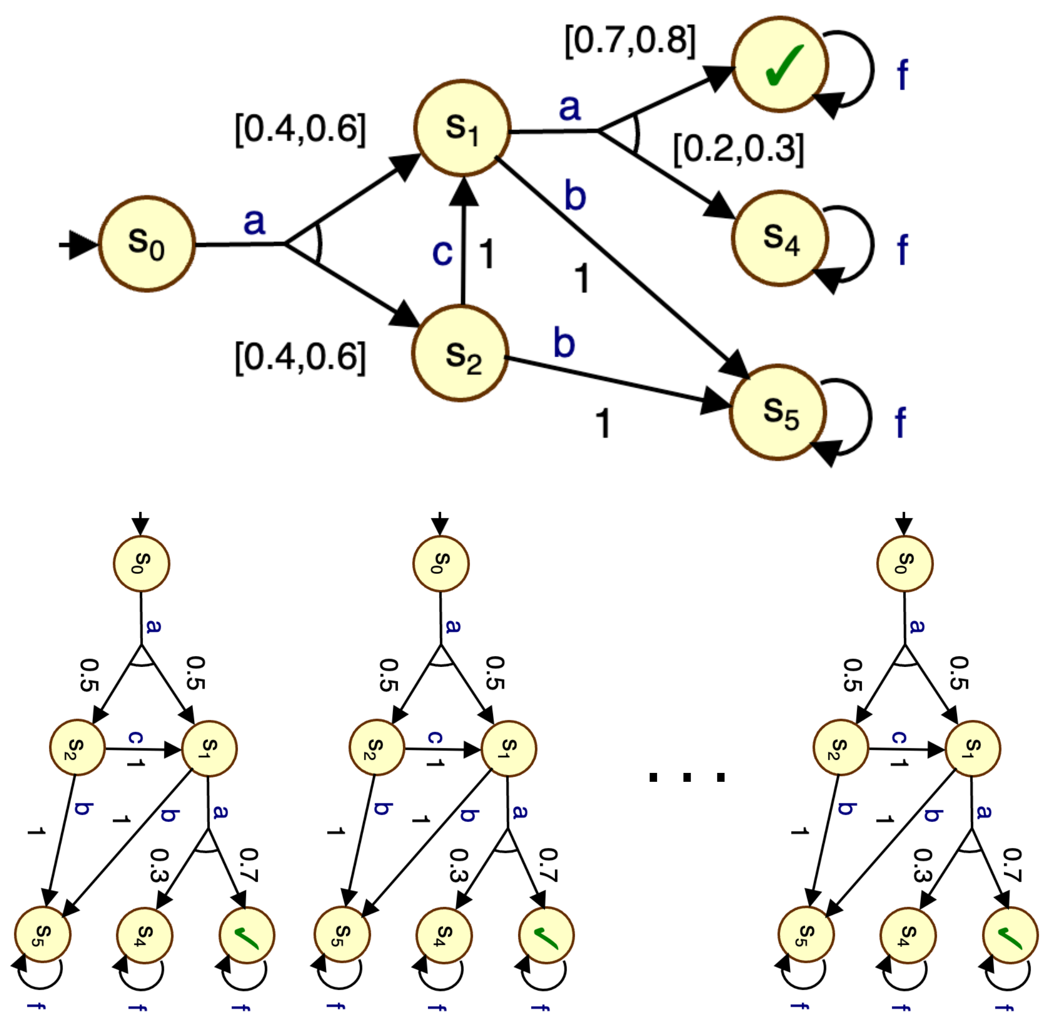
This course

- 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

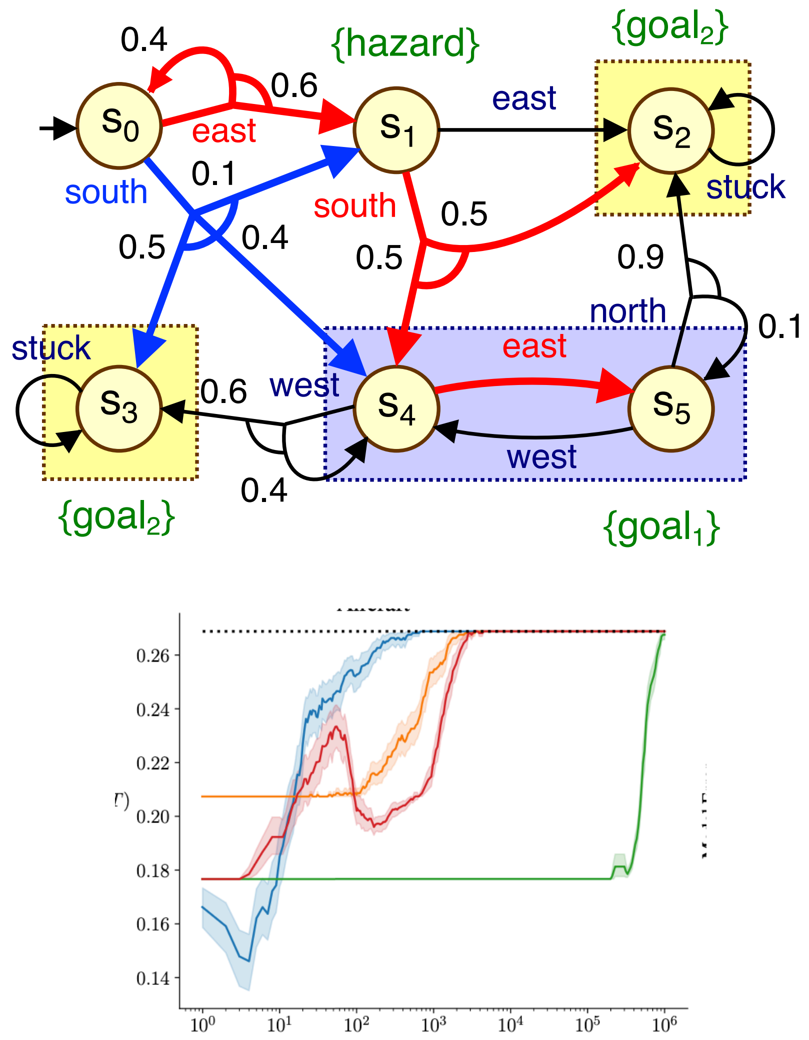
models + data



uncertain MDPs



policies + analysis & guarantees



Course contents

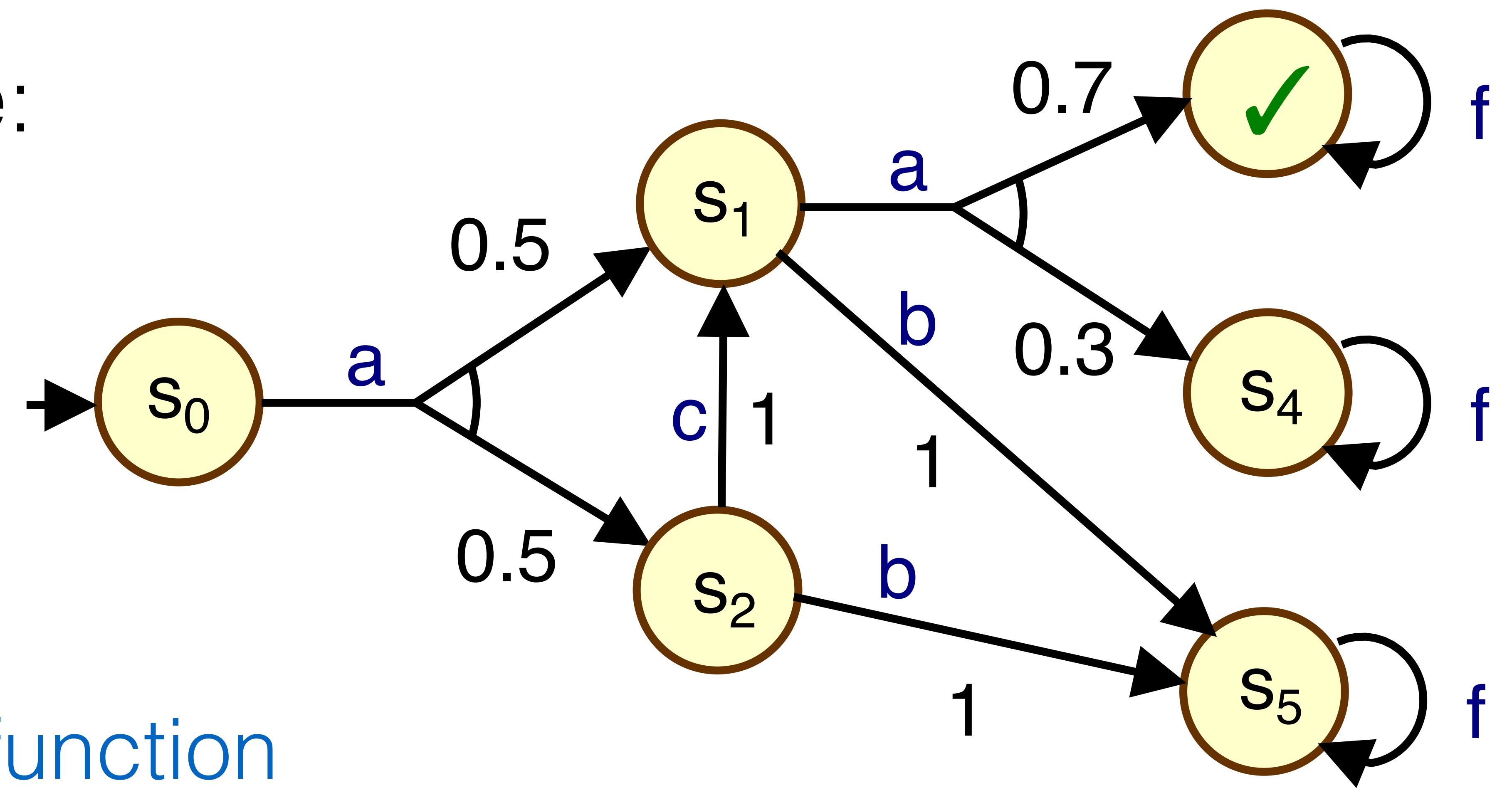
- Markov decision processes (MDPs) and stochastic games
    - MDPs: key concepts and algorithms
    - stochastic games: adding adversarial aspects
- Uncertain MDPs
    - MDPs + epistemic uncertainty, robust control, robust dynamic programming, interval MDPs, uncertainty set representation, challenges,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

Markov decision processes

Markov decision processes

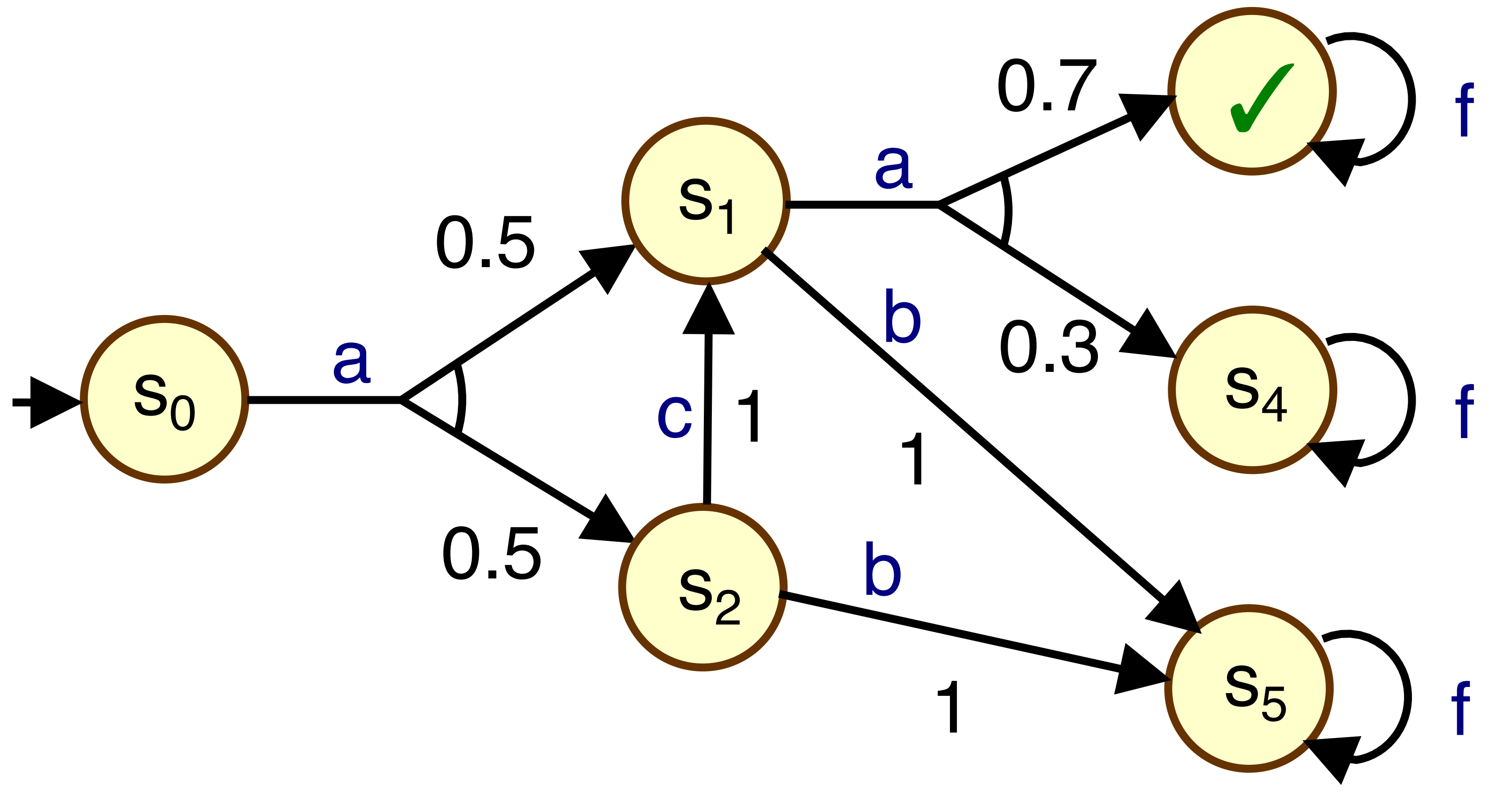
- Markov decision processes (MDPs)
  - standard model for sequential decision making under uncertainty

- 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

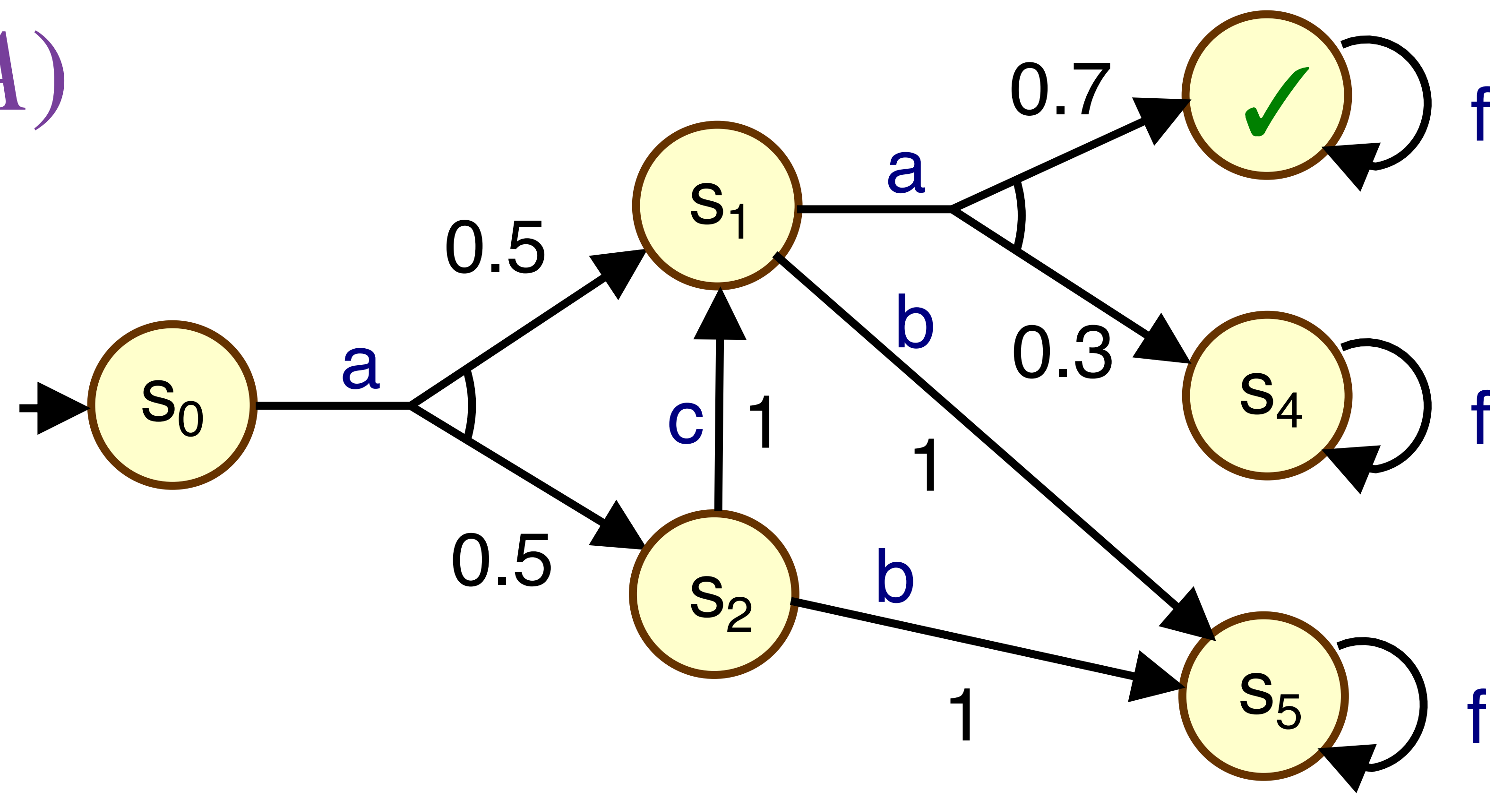
- For an MDP $\mathscr{M} = (S, s_0, A, P)$:
    - the enabled actions $A(s) \subseteq A$ in each state s
        - are $A(s) = \{a \in A : P(s, a, s') > 0$ for some $s'\}$
    - a path is a sequence $\omega = s_0 a_0 s_1 a_1, \ldots$
        - such that $s_i \in S$, $a_i \in A(s_i)$ and $P(s_i, a_i,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

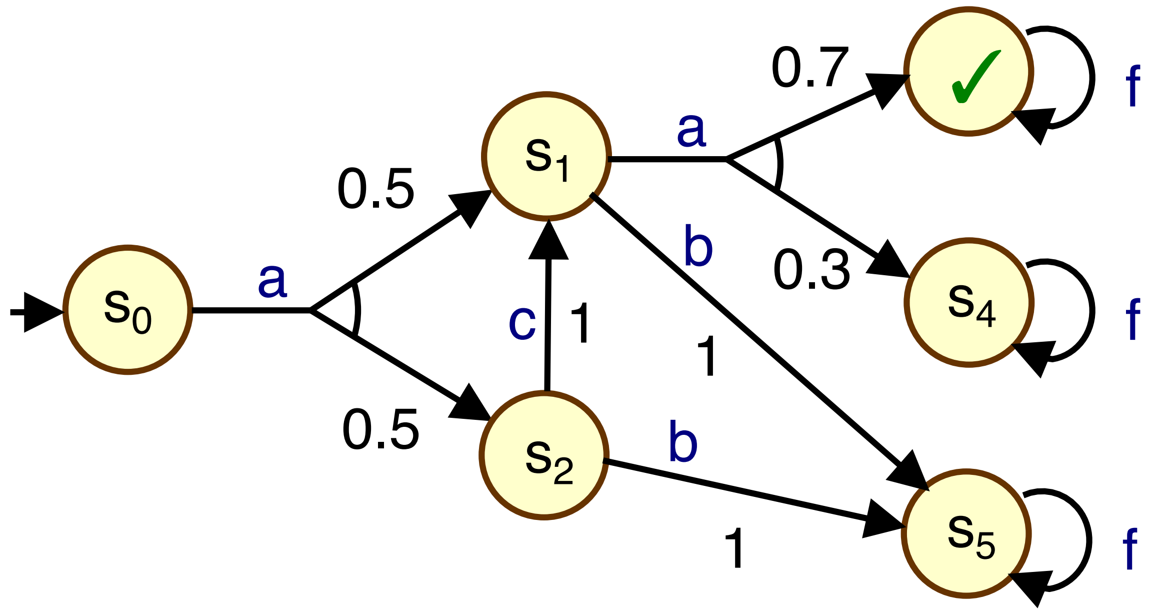
- Policies can be classified by (i) use of randomisation; (ii) use of memory
  - which matter for optimality, computation, practicality, …

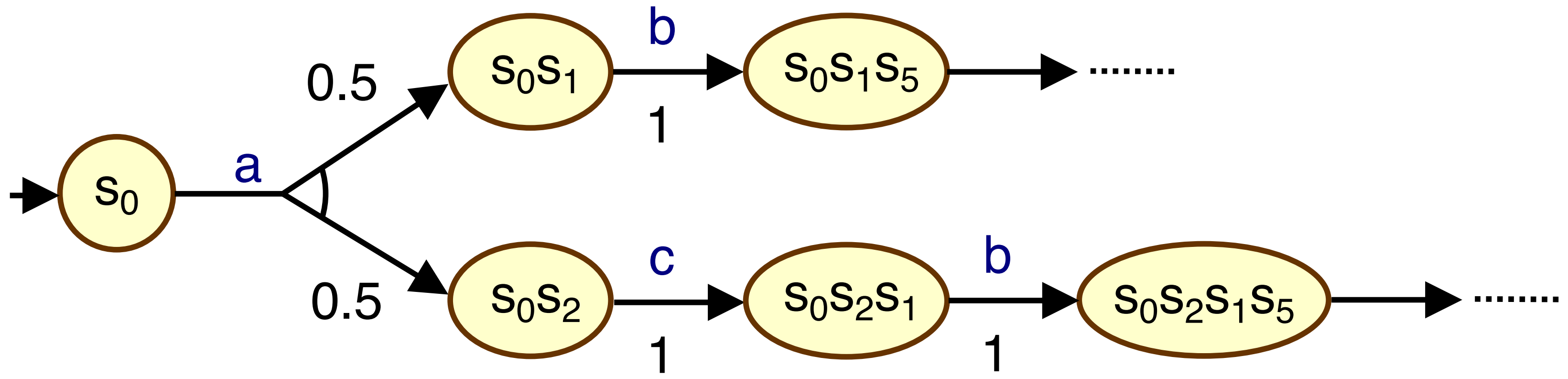
Classes of policies for MDPs

- 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 $\pi_i : S \to A$

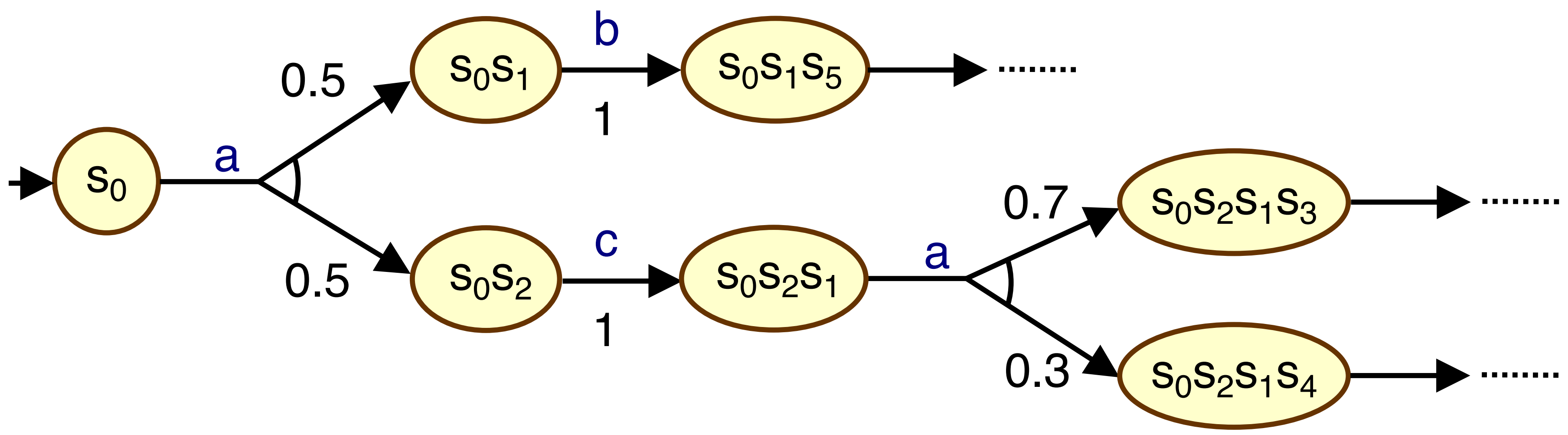
MDPs and policies

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

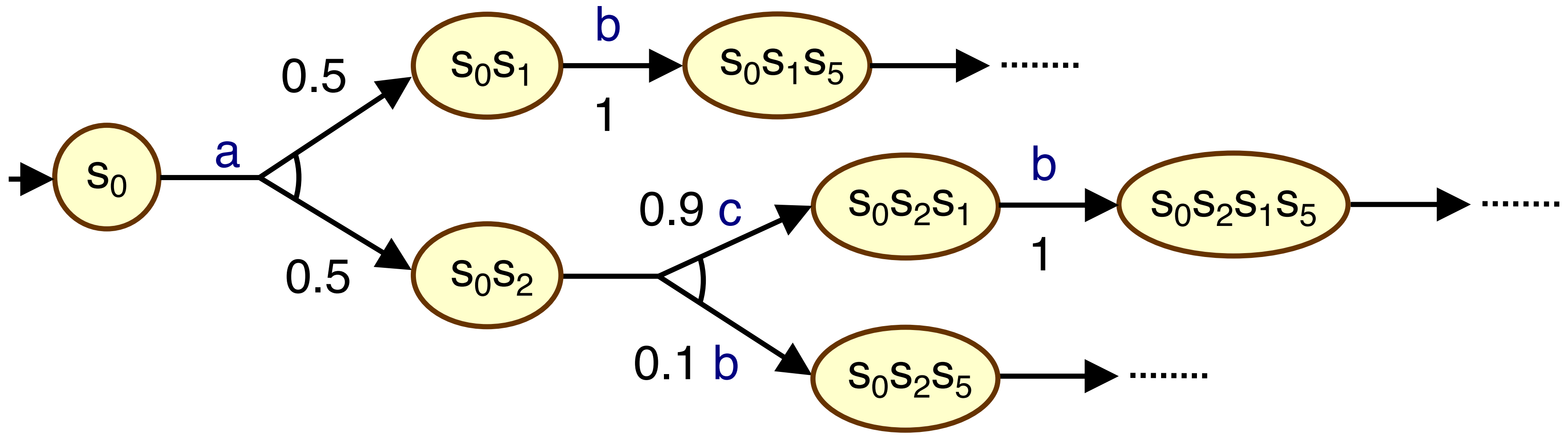




(memoryless, deterministic)



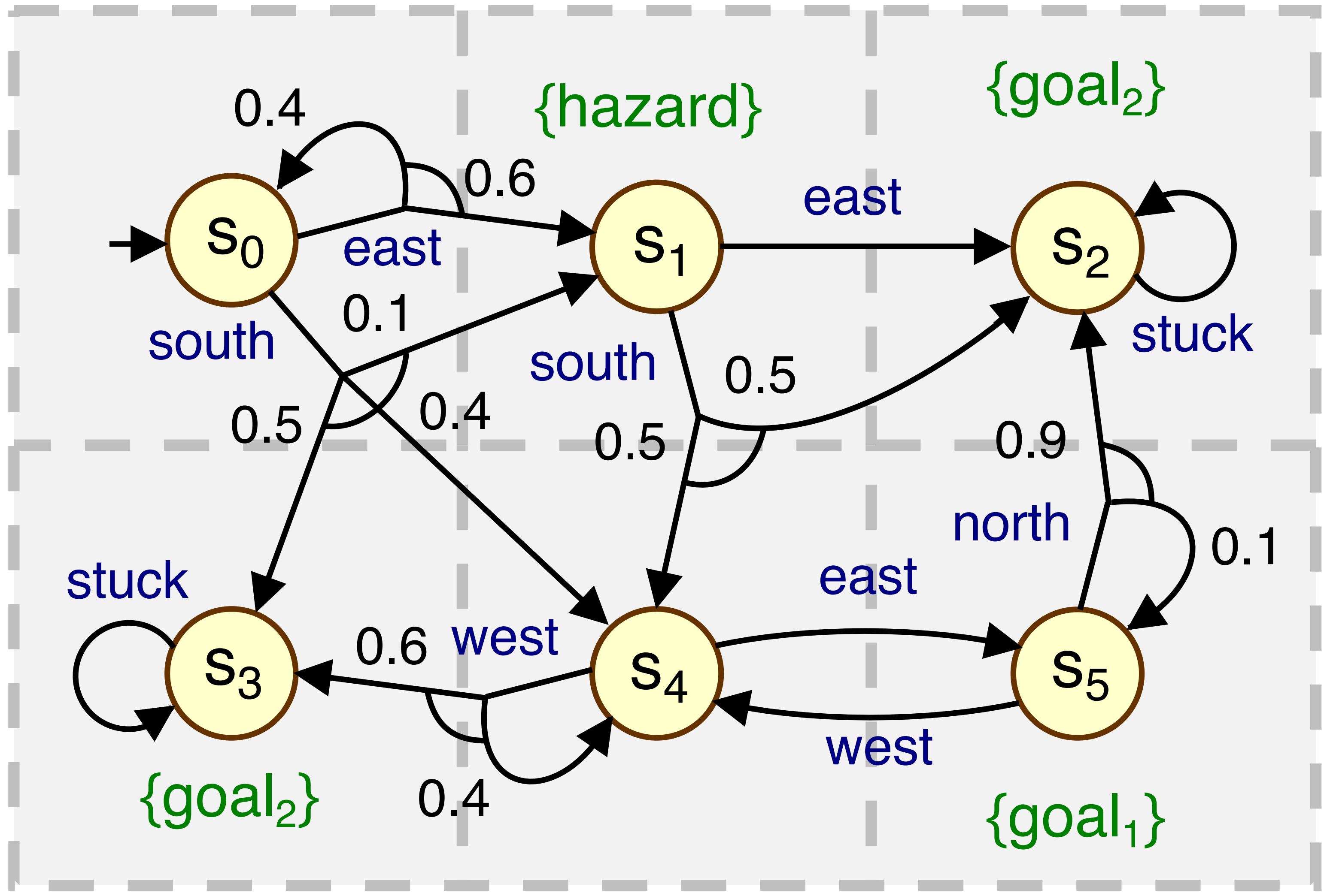
(finite-memory, deterministic)



(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 $X : (S \times A)^\omega \rightarrow \mathbb{R}_{\geq 0}$ is a random variable over (infinite) paths

- Value function: $V^\pi : S \rightarrow \mathbb{R}$
  - 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.: $\pi^* = \text{argmax}_{\pi \in \Pi} V^\pi(s_0)$

MaxProb & SSP (stochastic shortest path)

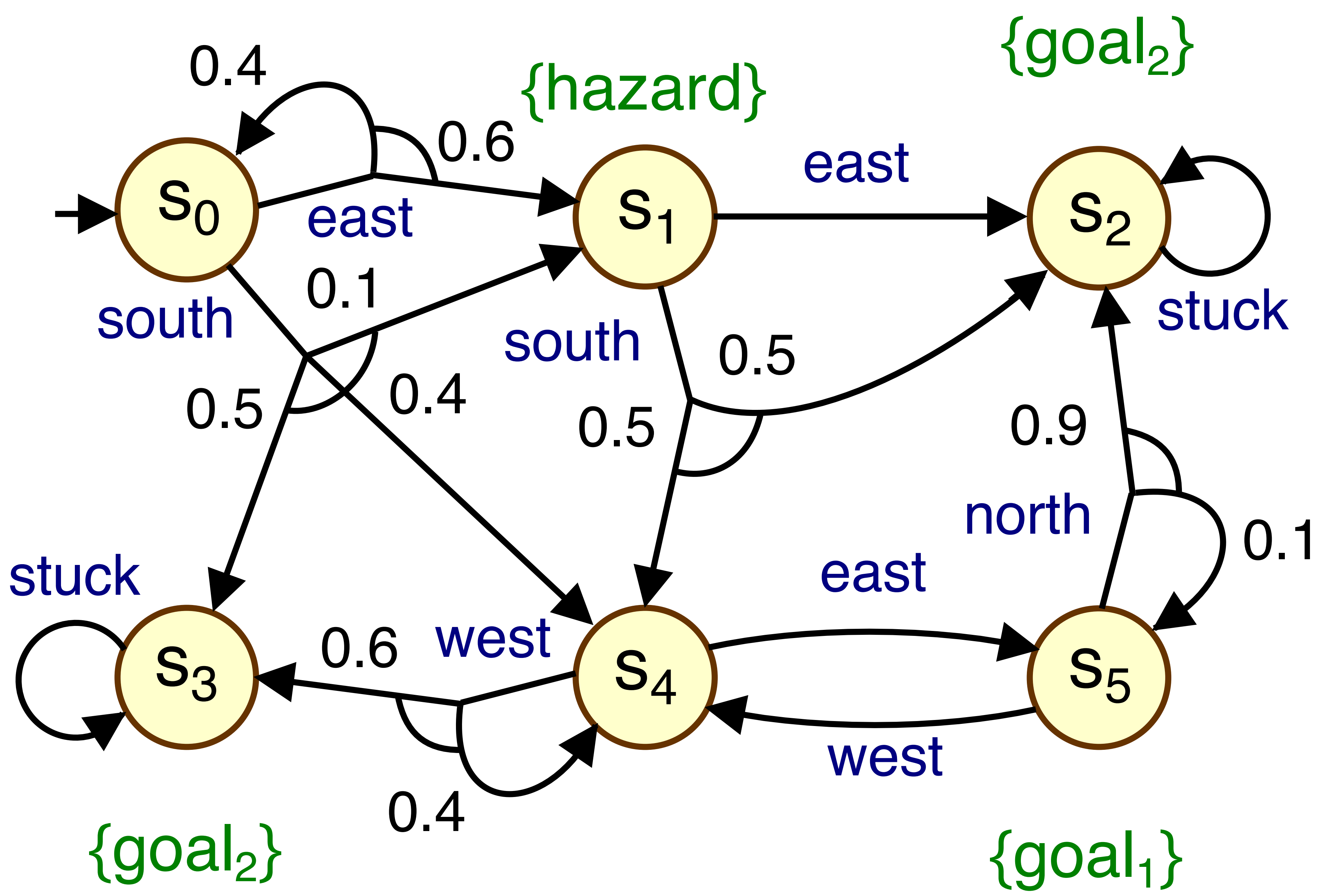
- MaxProb: Maximise the probability of reaching a target state set $goal \subseteq S$
  - maximise $V^\pi(s) = Pr_s^\pi(\{s_0 a_0 s_1 a_1 s_2 \ldots : s_i \in goal$ for some $i\})$

- 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($\color{green}{goal_1}$)?



Other objectives

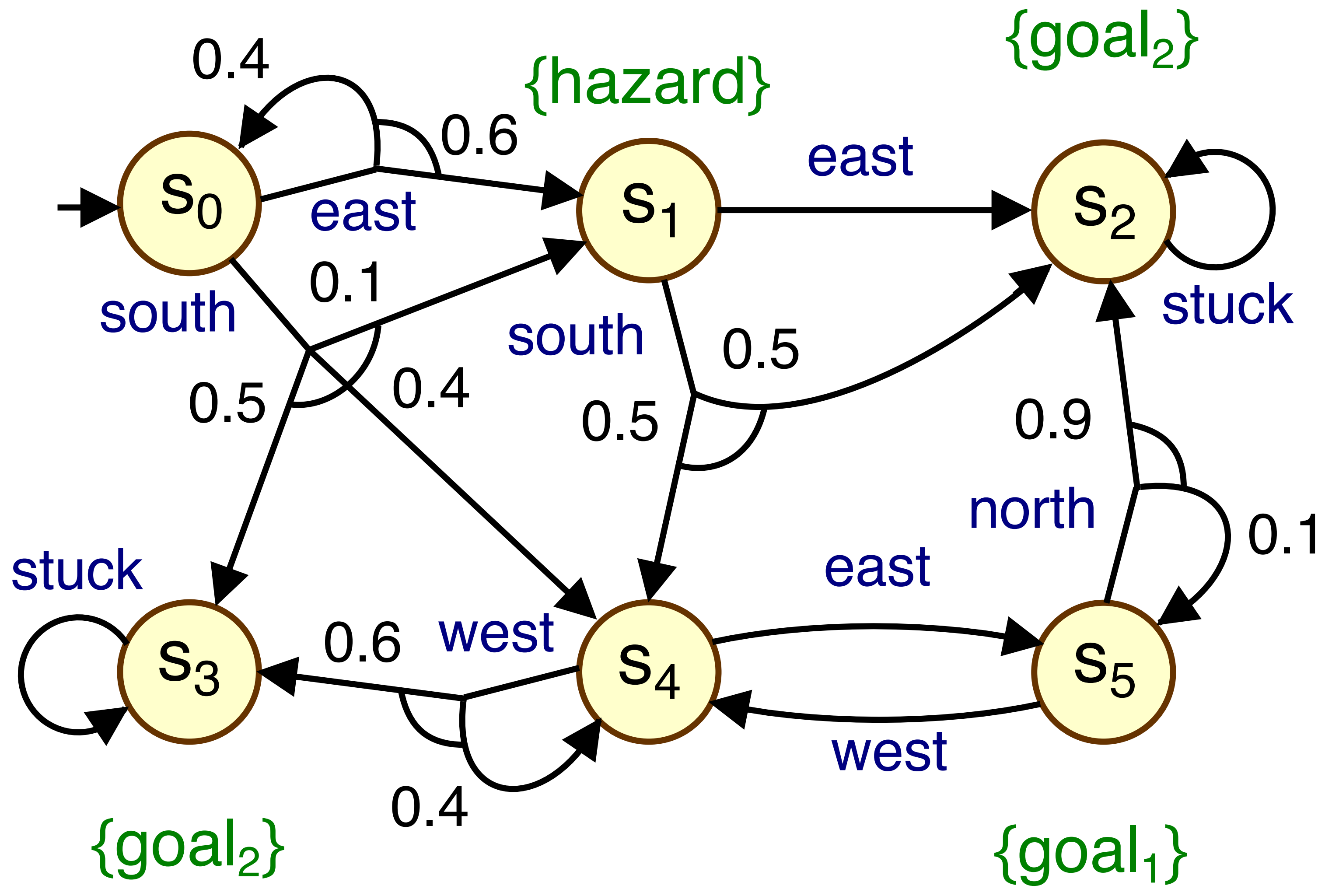
- Some other common objectives for MDPs:

- 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 $R : S \times A \to \mathbb{R}$ 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:

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

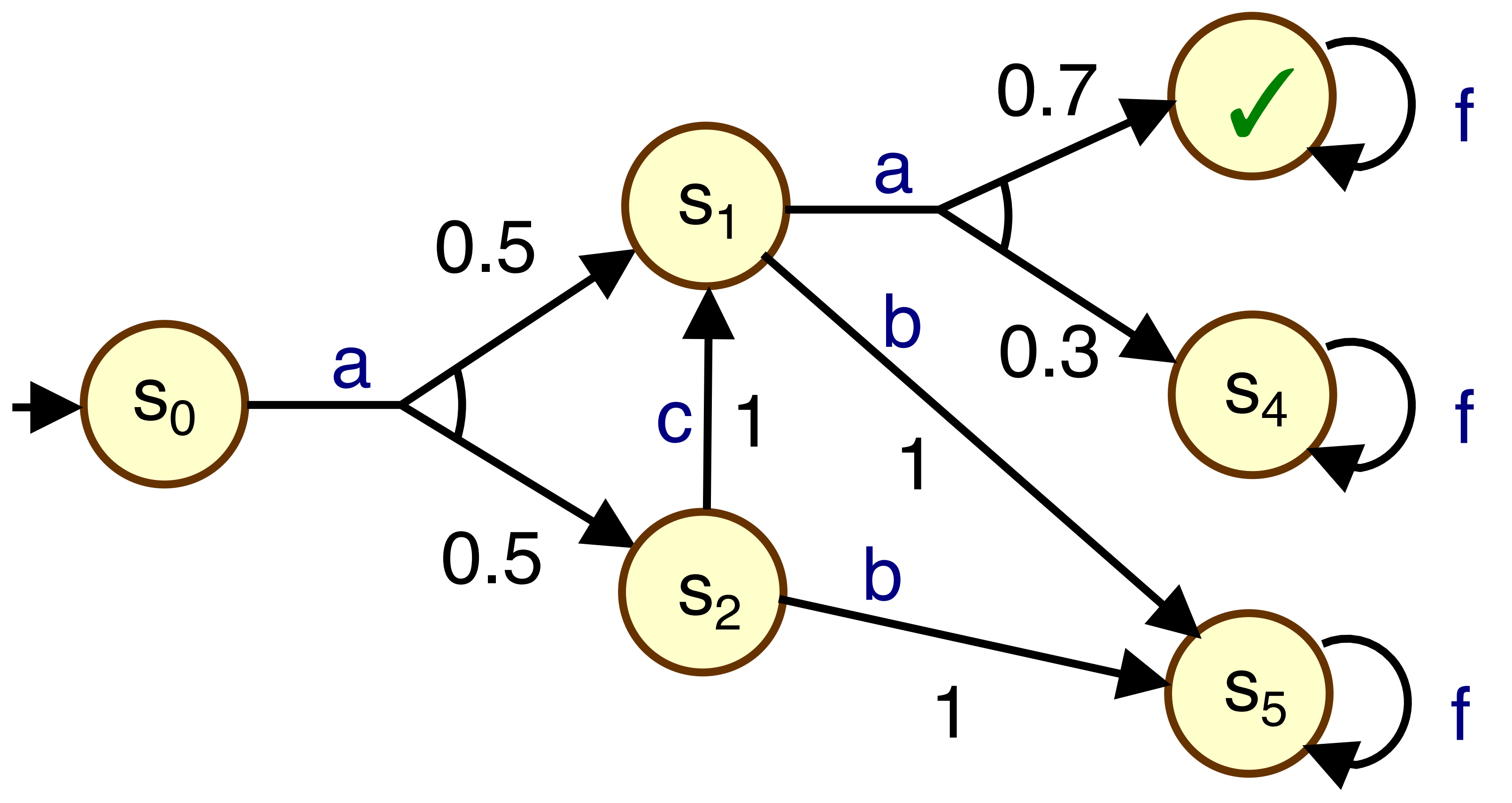
- Solution methods
  - value iteration (dynamic programming)
  - linear programming
  - and many more (e.g., policy iteration, Monte Carlo tree search, BRTDP, …)

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:

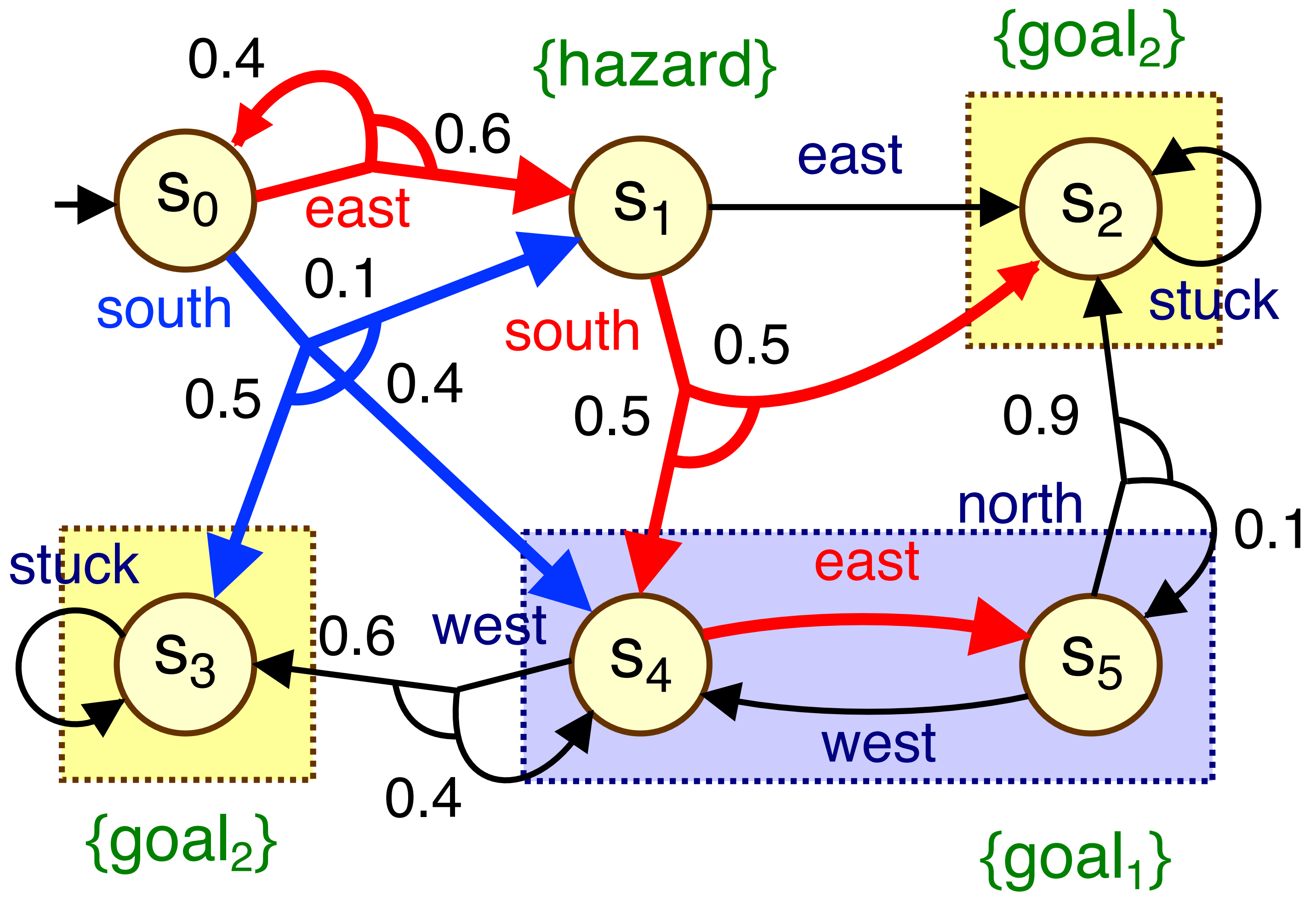
- Extract optimal policy after/during:

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

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

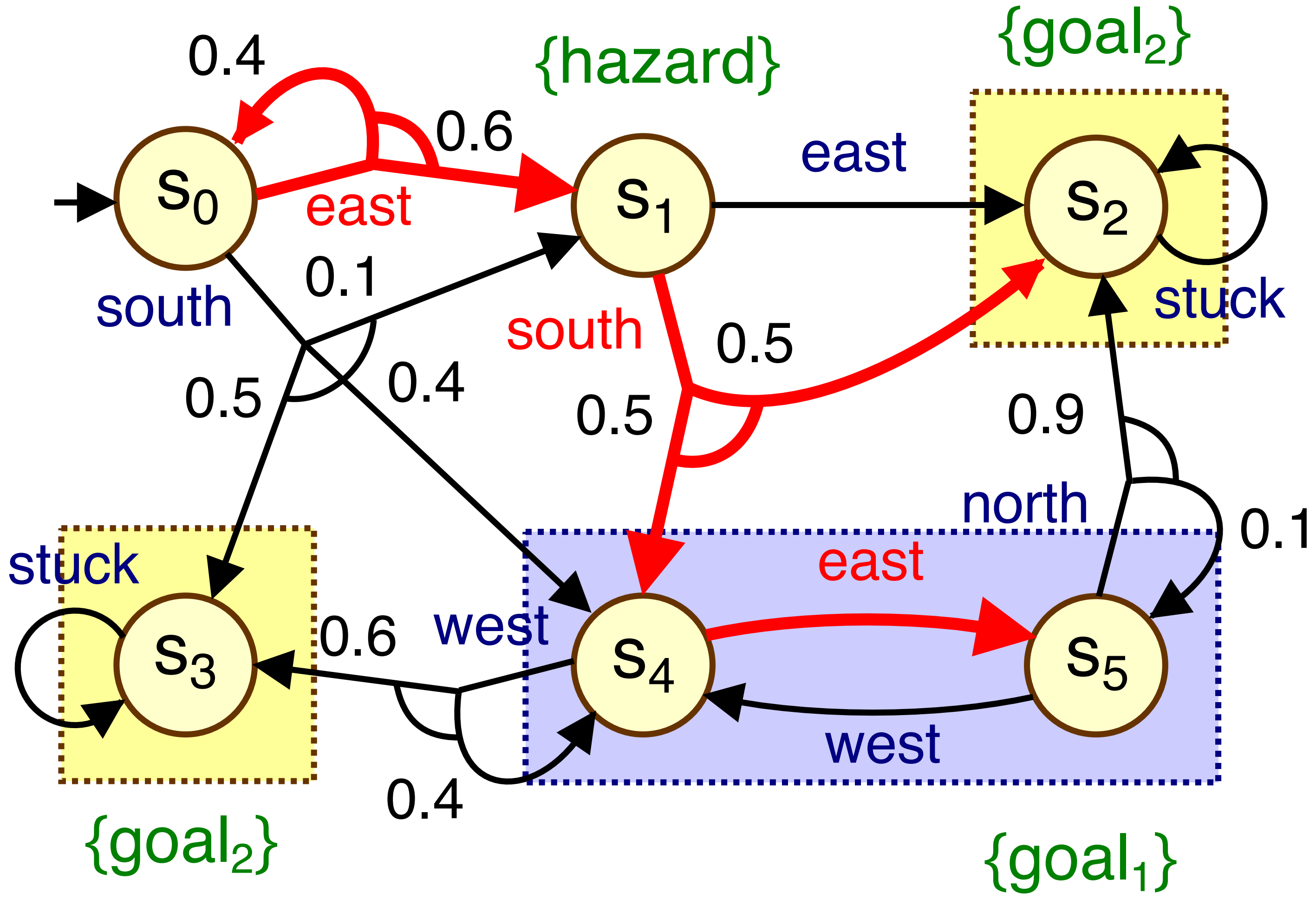
  $x_1 := 0.5$ (as before)
- Finally: $x_0=0.5$, $x_1=0.5$

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 $s \in S^?$, $a \in A(s)$

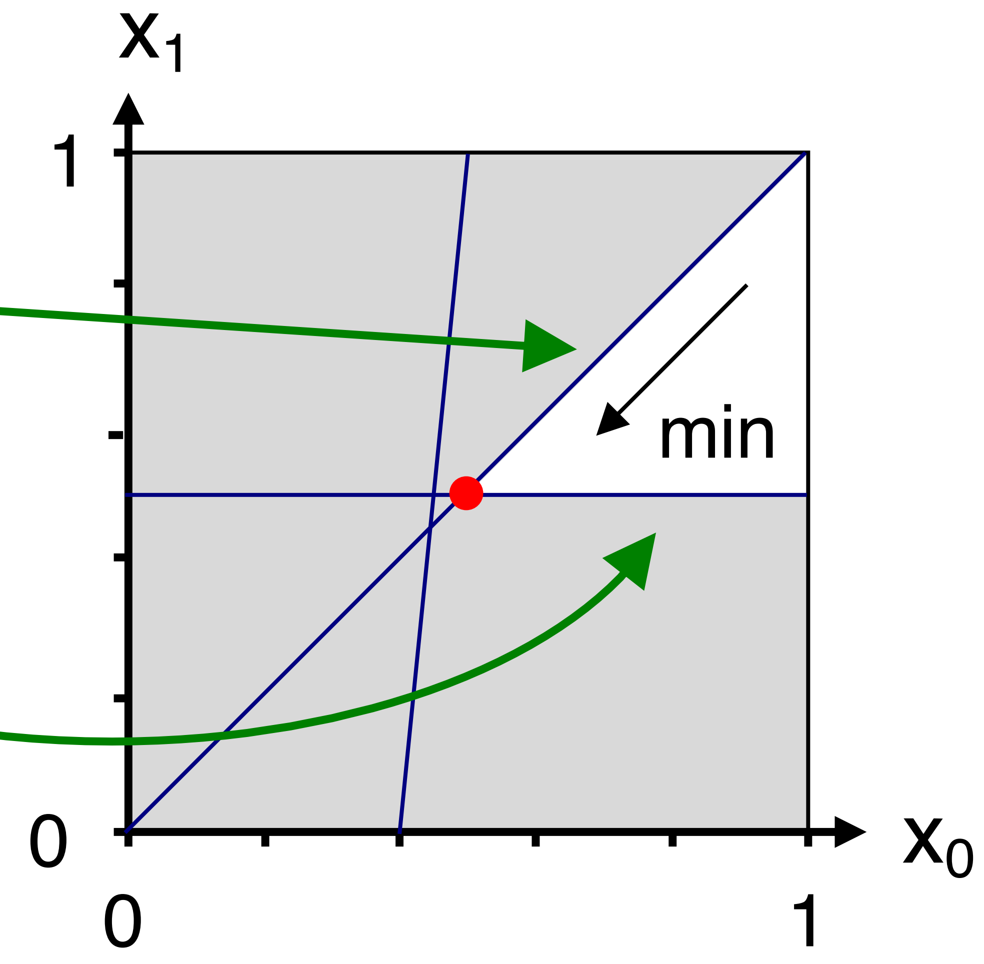


Minimise x_0+x_1 s.t.:
$x_0 \geq 0.4x_0 + 0.6x_1$
$x_0 \geq 0.1x_1 + 0.5x_3 + 0.4x_4$
$x_1 \geq x_2$
$x_1 \geq 0.5x_2 + 0.5x_4$

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:

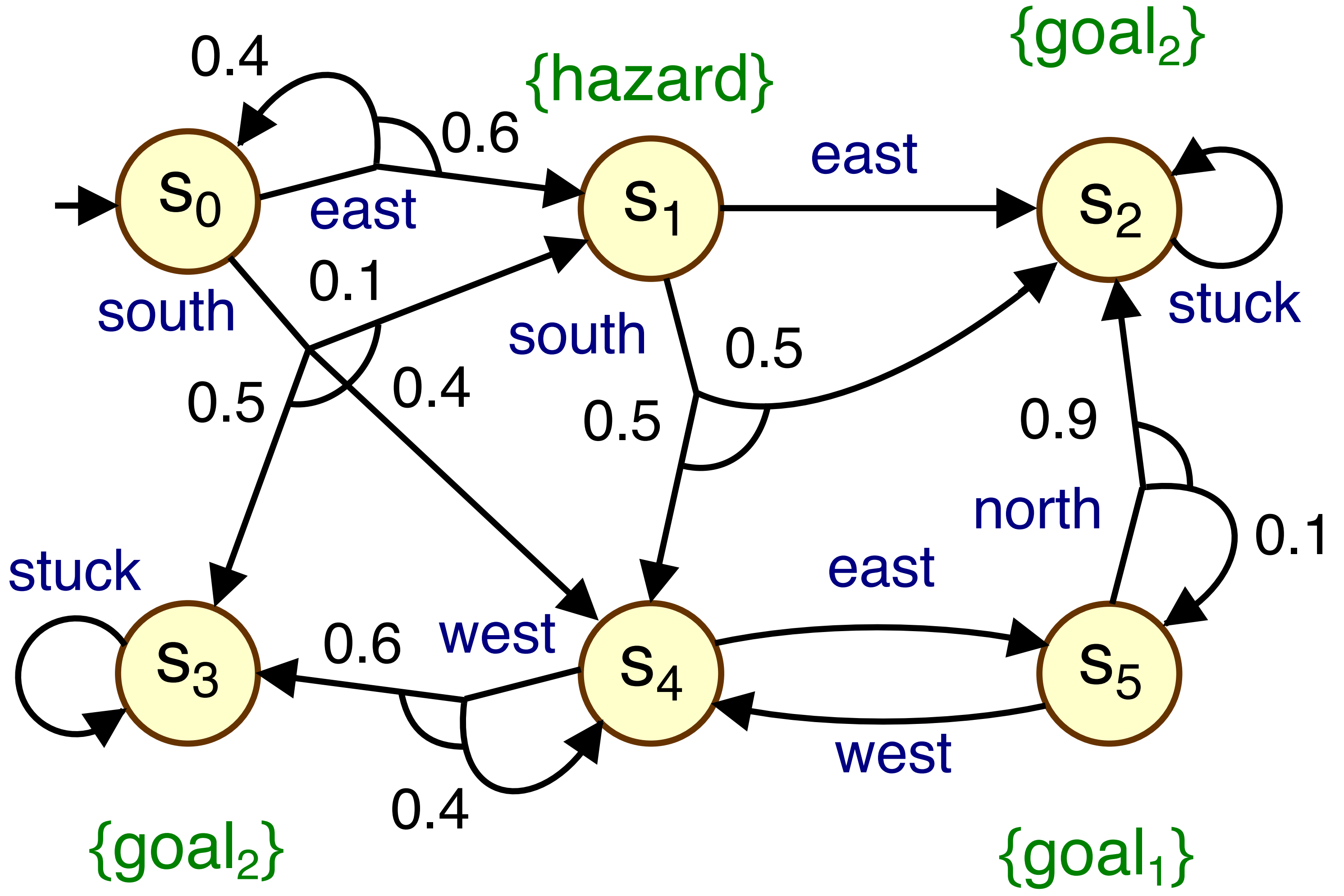
$$x_s^k = \begin{cases} 0 & \text{if } s \in goal \\ \min_{a \in A(s)} \left[C(s,a) + \sum_{s' \in S} P_s^a(s') \cdot x_{s'}^{k-1} \right] & \text{otherwise} \end{cases}$$

- 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), …

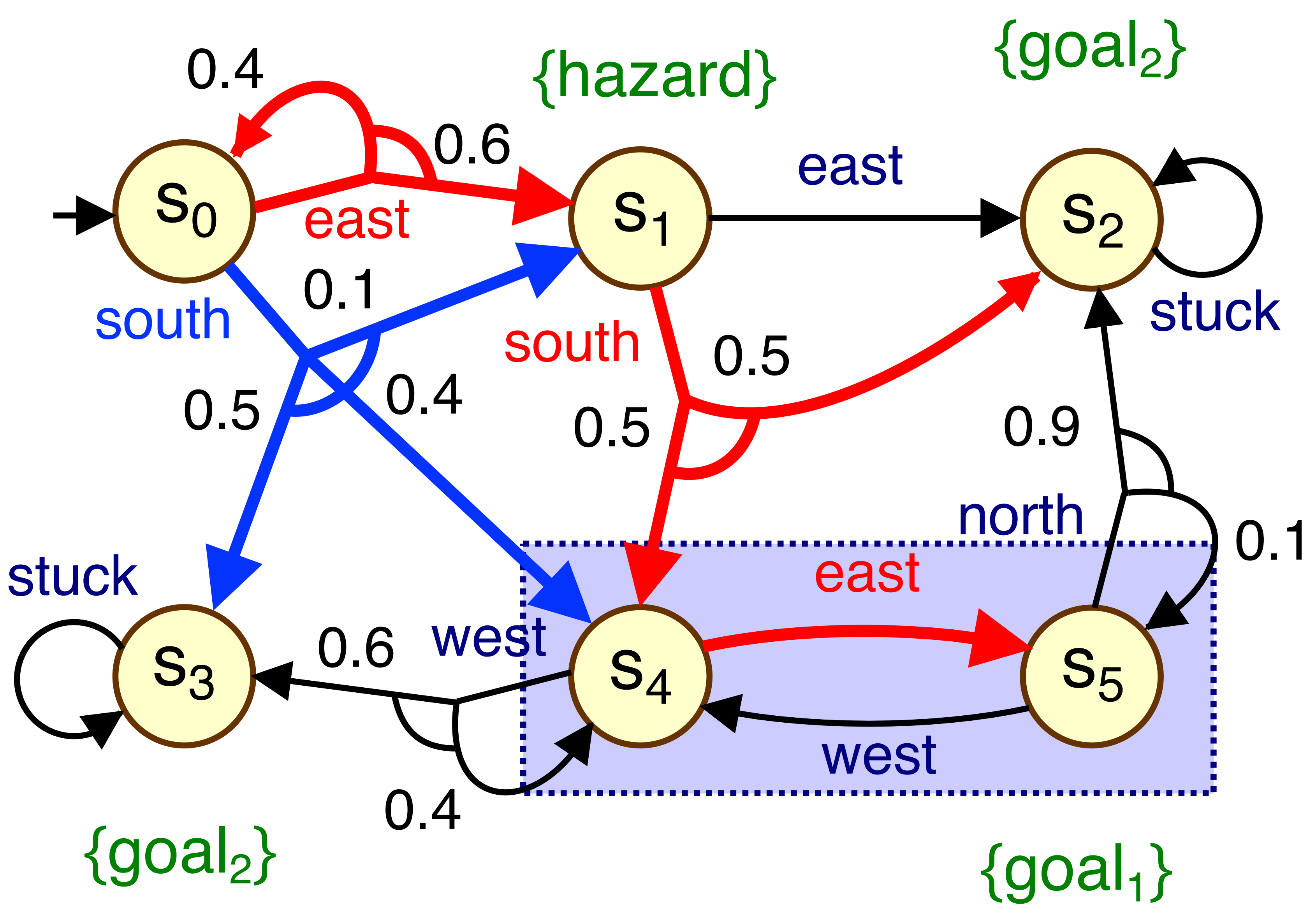
MaxProb over a finite horizon

- 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
  - $\text{MaxProb}^{\leq k}(\{s_4, s_5\})$
  - 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 (part 1)

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

- Next: Stochastic games (adding adversarial aspects)

- Next: Uncertain MDPs (adding epistemic uncertainty)

References (part 1)

- 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