Probabilistic Reasoning in Computer Science

Milan Češka

BRNO UNIVERSITY OF TECHNOLOGY

MSP 2024

Outline

Part 1: Modelling and Analysis of Probabilisitic Systems

- Lecture 1: Markov Chains
- Lecture 2: Markov Decision Processes

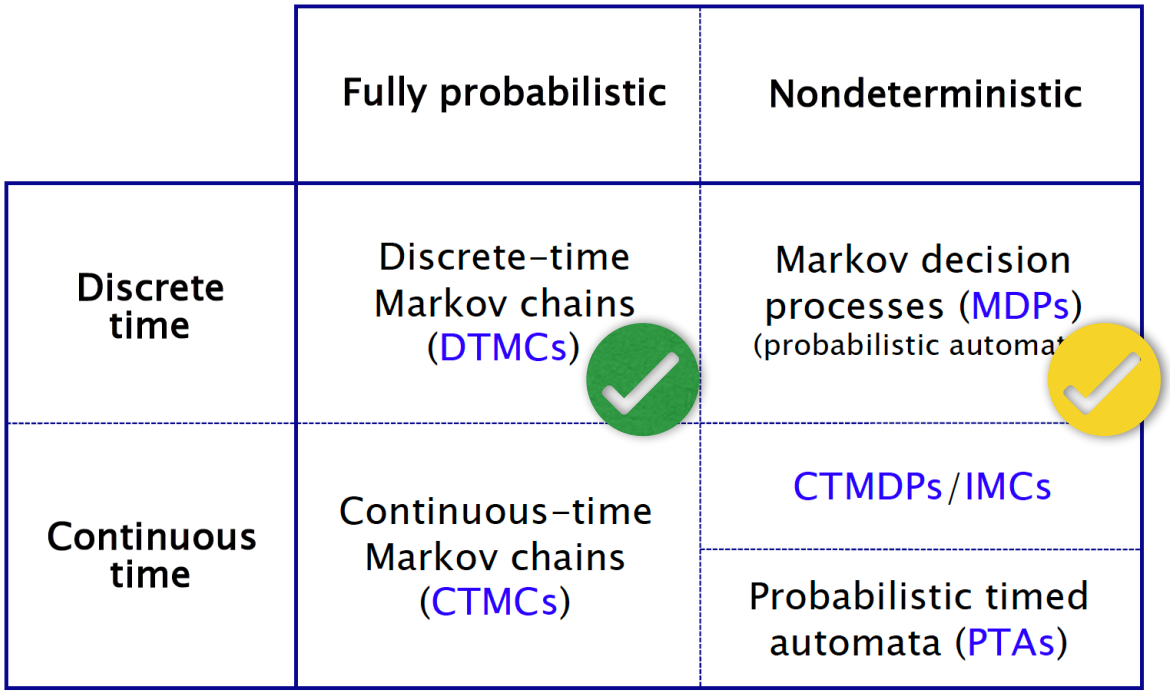
These lectures are partially based on material from Chapter 10 of "Principles of Model Checking" by Christel Baier and Joost-Pieter Katoen (MIT Press) and "Stochastic Model Checking" by Marta Kwiatkowska, Gethin Norman and David Parker. The slides use material from https://www.prismmodelchecker.org/lectures/pmc/

Part 2: Introduction to Randomised Algorithms

- Lecture 3: Probabilisitic analysis of programs and basic randomised algorithms

This lecture is partially based on material from Chapter 5 of "Introduction to Algorithms (3rd edition)" by Thomas Cormen et al. (MIT Press).

Modelling of Probabilistic Systems



Other important characteristics

- finite x infinite x uncountable
- fully observable x parametric x partially observable
- labelled transition systems

BRNO UNIVERSITY OF TECHNOLOGY

Markov Decision Processes — Why Nondeterminism?

Some system aspects/decision should not be modelled probabilistically

- concurrency — scheduling of parallel components
- unknown model parameters
- unknown environment/adversary

Markov Decision Processes — Why Nondeterminism?

Some system aspects/decision should not be modelled probabilistically

- concurrency — scheduling of parallel components
- unknown model parameters
- unknown environment/adversary

To model controllable probabilistic systems

- MDPs are essential in control theory
- performing an action leads to a probability distribution over its outcomes
- many application domains: operational research, motion planning, optimal control

Markov Decision Processes — Why Nondeterminism?

Some system aspects/decision should not be modelled probabilistically

- concurrency — scheduling of parallel components
- unknown model parameters
- unknown environment/adversary

To model controllable probabilistic systems

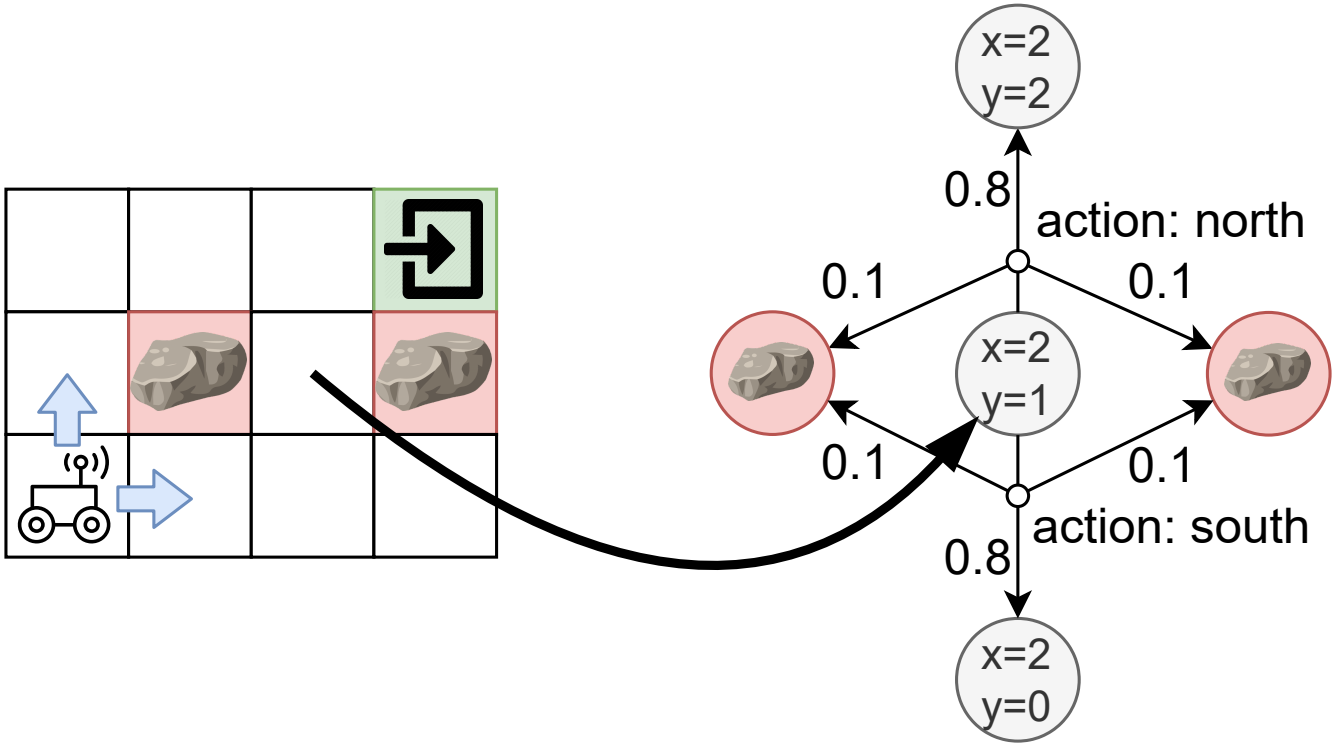
- MDPs are essential in control theory
- performing an action leads to a probability distribution over its outcomes
- many application domains: operational research, motion planning, optimal control

Close connection between MDPs and reinforcement learning problems

- e.g. scheduling in probabilistic environment, motion planning

MDPs — Robot in the Grid World (canonical example)

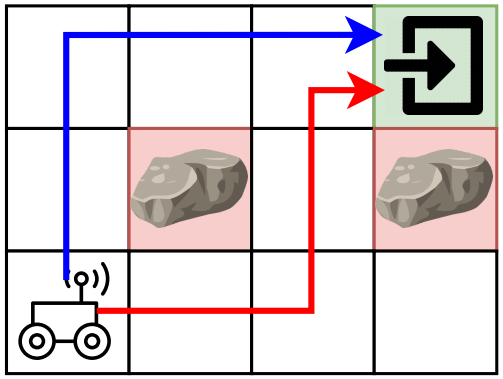
- in each state, multiple actions are available
- actions are not “perfect”: modelled as probability distribution over next states

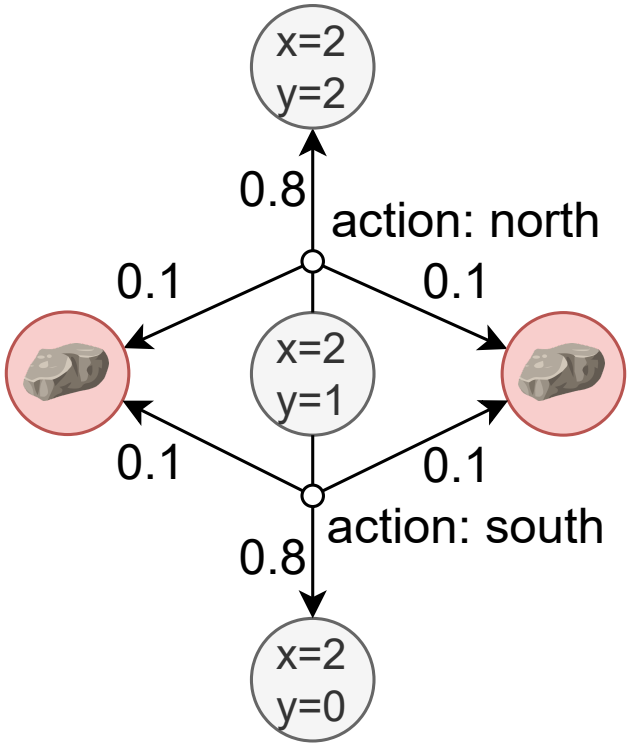


BRNO UNIVERSITY OF TECHNOLOGY

MDPs — Robot in the Grid World (canonical example)

- in each state, multiple actions are available
- actions are not “perfect”: modelled as probability distribution over next states



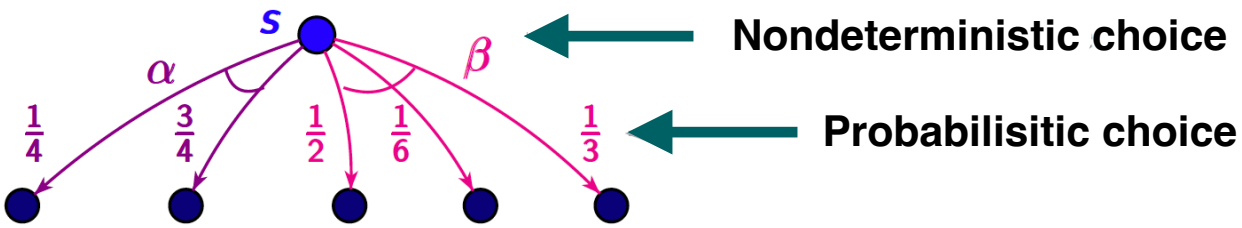


What is the optimal strategy if there is an uncertainty?

BRNO UNIVERSITY OF TECHNOLOGY

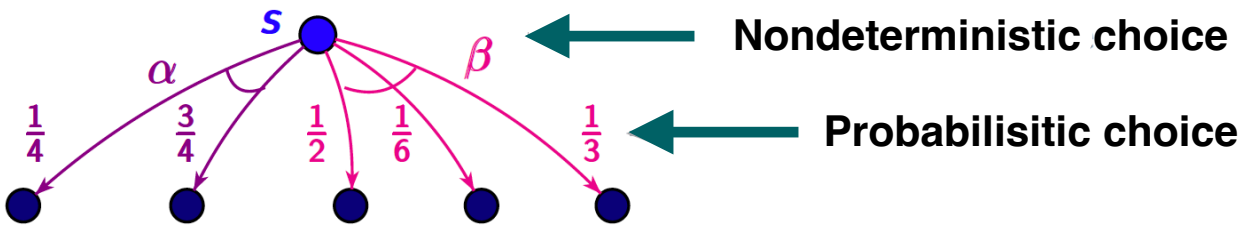
Markov Decision Processes

Extension of DTMCs which allow nondeterministic choice



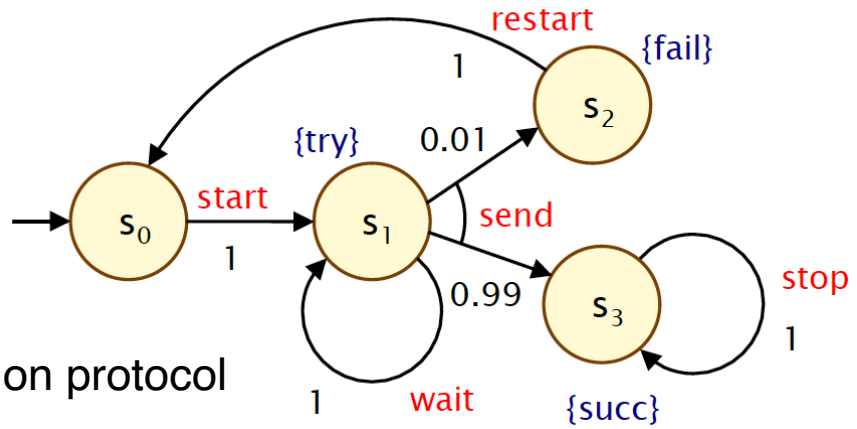
Markov Decision Processes

Extension of DTMCs which allow nondeterministic choice



Combines nondeterminism and probabilities

- in each state, a nondeterministic choice between several probability distributions over successor states



Modified communication protocol

BRNO UNIVERSITY OF TECHNOLOGY

Markov Decision Processes (MDPs)

Formally, an MDP M is a tuple $(S, s_0, Act, \mathbf{P})$, where

- S is a finite set of states (state space)
- $s_0 \in S$ is the initial state
- Act is a finite set of actions
  - $Act(s) \subseteq Act$ denotes the set of actions available at state $s \in S$
  - we assume $\forall s : Act(s) \neq \emptyset$, i.e. no deadlocks
- $\mathbf{P} : S \times Act \times S \to [0,1]$ is the transition probability matrix where for all $s \in S$ and $\alpha \in Act$ holds $\sum_{s' \in S} \mathbf{P}(s, \alpha, s') = 1$ if α is available in s ($\alpha \in Act(s)$), and 0 otherwise
  - $\mathbf{P}(s, \alpha, s')$ represents the probability of transitioning from s to s' when executing action α

Paths in MDPs

- finite path is a sequence of states and actions: $\pi = s_0\alpha_0, s_1\alpha_1, s_2\alpha_2, \ldots, s_n$ where $\alpha_i \in Act(s_i)$ and $\mathbf{P}(s_i, \alpha_i, s_{i+1}) > 0$
- represents a concrete execution of the MDP — resolves both types of "choices"
- let $\text{last}(\pi) = s_n$ denote the last state of path π
- let Paths_M denote the set of finite paths in M

Scheduler

- also known as "controller", "policy", "strategy", "adversary"
- resolves the nondeterministic choices
- scheduler $\sigma\colon \text{Paths}_M \to Distr(Act)$ assigns to every finite path π a probability distribution $\sigma(\pi) \in Distr(Act)$ over actions, $\text{supp}(\sigma(\pi)) \subseteq Act(\text{last}(\pi))$
  - deterministic scheduler assigns to a path a single action: $\sigma\colon \text{Paths}_M \to Act$
  - memoryless scheduler picks action based only on the current state: $\sigma\colon S \to Distr(Act)$
  - deterministic memoryless scheduler: $\sigma\colon S \to Act$

Induced DTMC

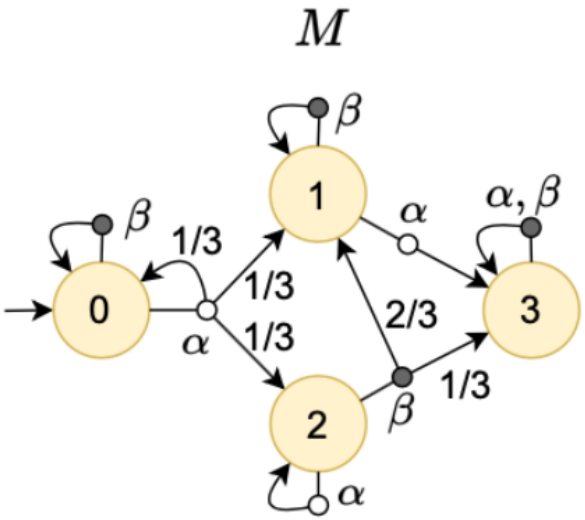
- scheduler σ for MDP $M = (S, s_0, Act, \boldsymbol{P})$ *induces DTMC*
  $M^\sigma = (Paths^M, s_0, \boldsymbol{P}^\sigma)$ where $\boldsymbol{P}^\sigma(\pi, \pi') := \sigma(\pi)(\alpha) \cdot \boldsymbol{P}(\mathrm{last}(\pi), \alpha)(s')$
  if $\pi' = \pi\alpha s'$, and 0 otherwise

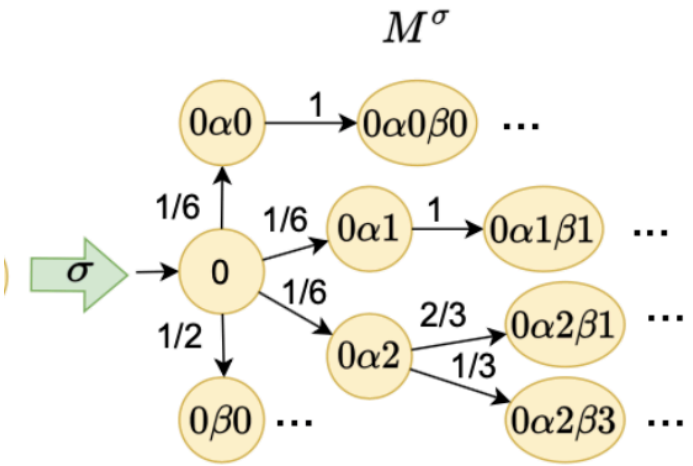
Induced DTMC

- scheduler σ for MDP $M = (S, s_0, Act, \boldsymbol{P})$ *induces DTMC* $M^\sigma = (Paths^M, s_0, \boldsymbol{P}^\sigma)$ where $\boldsymbol{P}^\sigma(\pi, \pi') := \sigma(\pi)(\alpha) \cdot \boldsymbol{P}(\mathrm{last}(\pi), \alpha)(s')$ if $\pi' = \pi\alpha s'$, and 0 otherwise

Example. $\sigma \equiv$ pick 1st action randomly, then alternate

$$\sigma(0) = \frac{1}{2} : \alpha + \frac{1}{2} : \beta$$
$$\sigma(0\ [\cdot\ \cdot]^*\alpha\ \cdot) = \beta$$
$$\sigma(0\ [\cdot\ \cdot]^*\beta\ \cdot) = \alpha$$





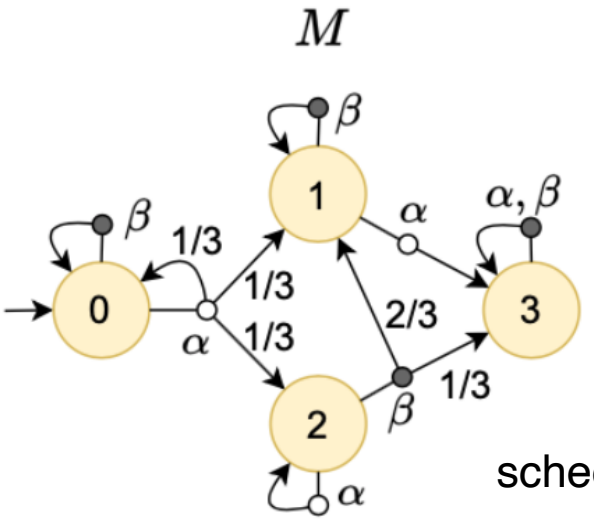
BRNO UNIVERSITY OF TECHNOLOGY

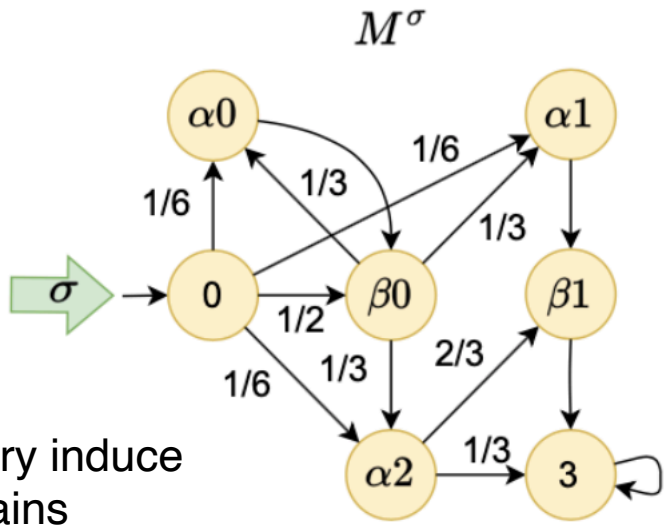
Induced DTMC

- scheduler σ for MDP $M = (S, s_0, Act, \boldsymbol{P})$ *induces DTMC* $M^\sigma = (Paths^M, s_0, \boldsymbol{P}^\sigma)$ where $\boldsymbol{P}^\sigma(\pi, \pi') := \sigma(\pi)(\alpha) \cdot \boldsymbol{P}(\mathrm{last}(\pi), \alpha)(s')$ if $\pi' = \pi\alpha s'$, and 0 otherwise

Example. $\sigma \equiv$ pick 1st action randomly, then alternate

$$\sigma(0) = \frac{1}{2} : \alpha + \frac{1}{2} : \beta$$
$$\sigma(0\ [\cdot\ \cdot]^*\alpha\ \cdot) = \beta$$
$$\sigma(0\ [\cdot\ \cdot]^*\beta\ \cdot) = \alpha$$





schedulers with finite memory induce finite-state Markov chains

BRNO UNIVERSITY OF TECHNOLOGY

Induced DTMC

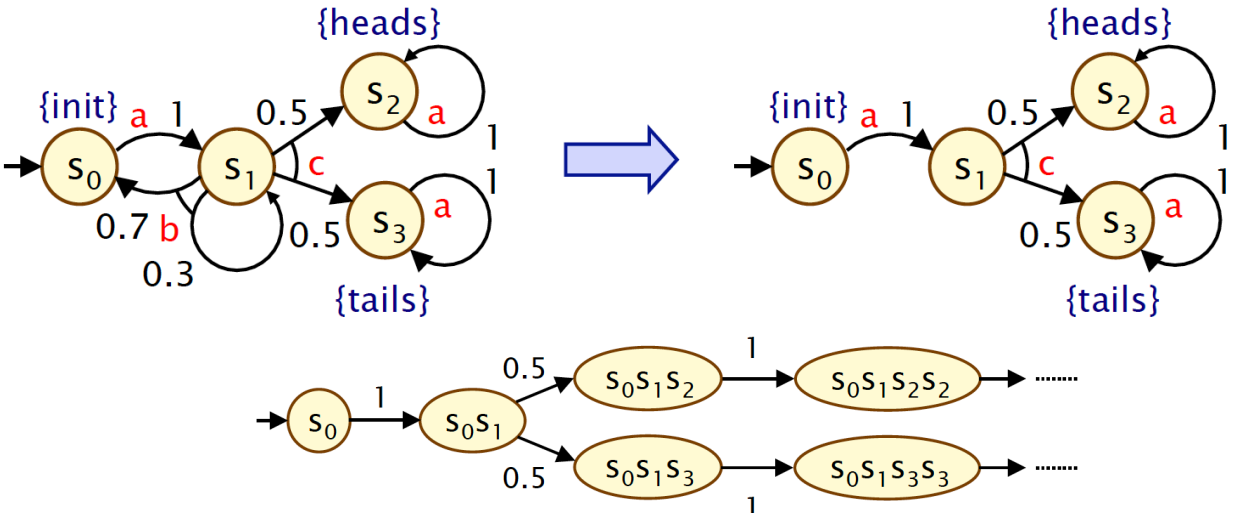
- scheduler σ for MDP $M = (S, s_0, Act, \boldsymbol{P})$ *induces DTMC*
  $M^\sigma = (Paths^M, s_0, \boldsymbol{P}^\sigma)$ where $\boldsymbol{P}^\sigma(\pi, \pi') := \sigma(\pi)(\alpha) \cdot \boldsymbol{P}(\text{last}(\pi), \alpha)(s')$
  if $\pi' = \pi\alpha s'$, and 0 otherwise

- for a deterministic memoryless scheduler σ the induced DTMC is defined as

  $M^\sigma = (S, s_0, P^\sigma)$ where $P^\sigma(s, s') = P(s, \sigma(s), s')$

Scheduler — More examples

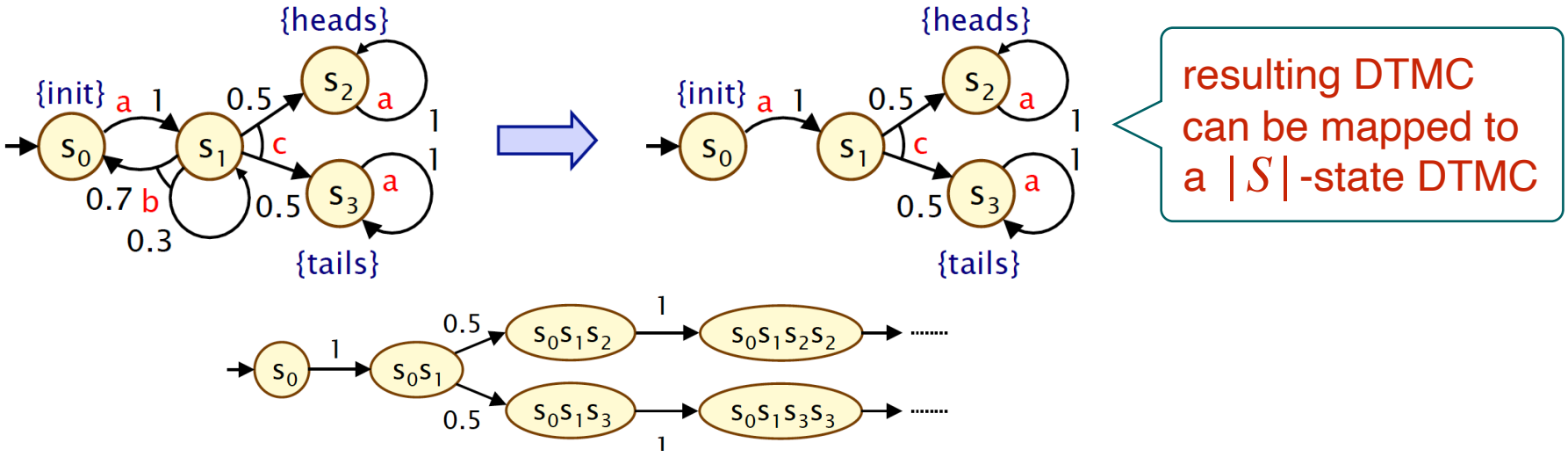
Memoryless scheduler σ_1 always takes action c in state s_1



BRNO UNIVERSITY OF TECHNOLOGY

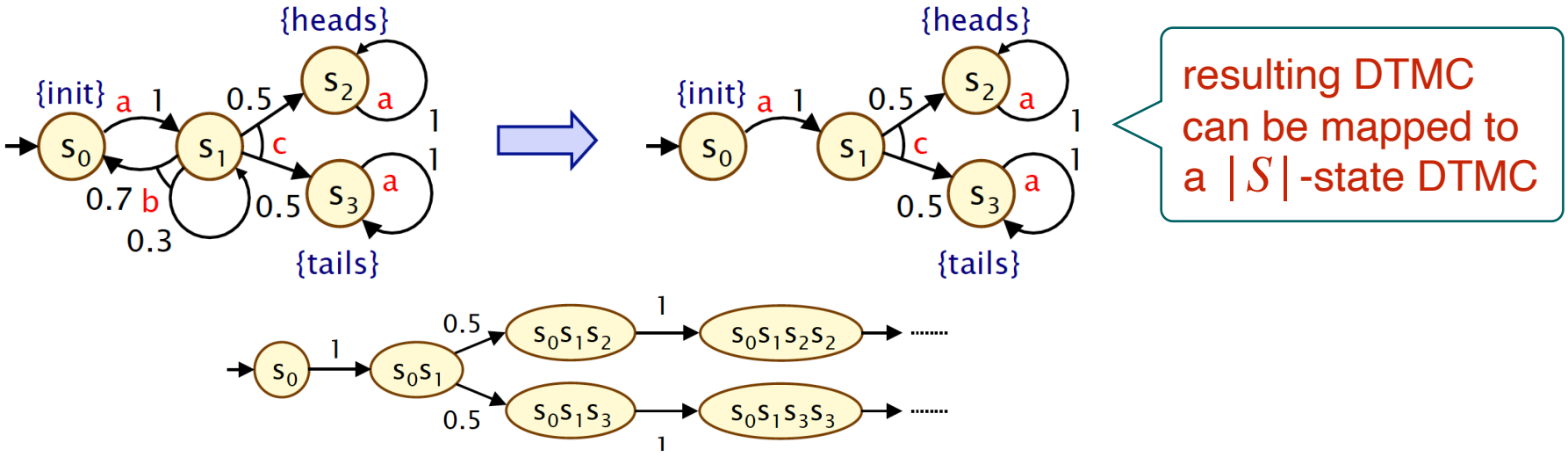
Scheduler — More examples

Memoryless scheduler σ_1 always takes action c in state s_1

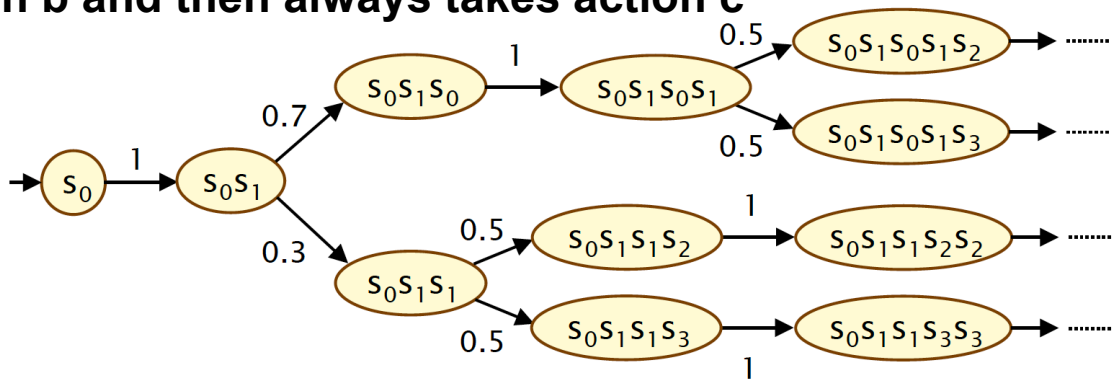


Scheduler — More examples

Memoryless scheduler σ_1 always takes action c in state s_1



Scheduler σ_2 first takes action b and then always takes action c



BRNO UNIVERSITY OF TECHNOLOGY

Model checking MDPs

- we can use the same specification language as for DTMCs
  - in this course we will investigate only reachability properties $P(s_0 \to T)$

Verification vs synthesis in MDPs

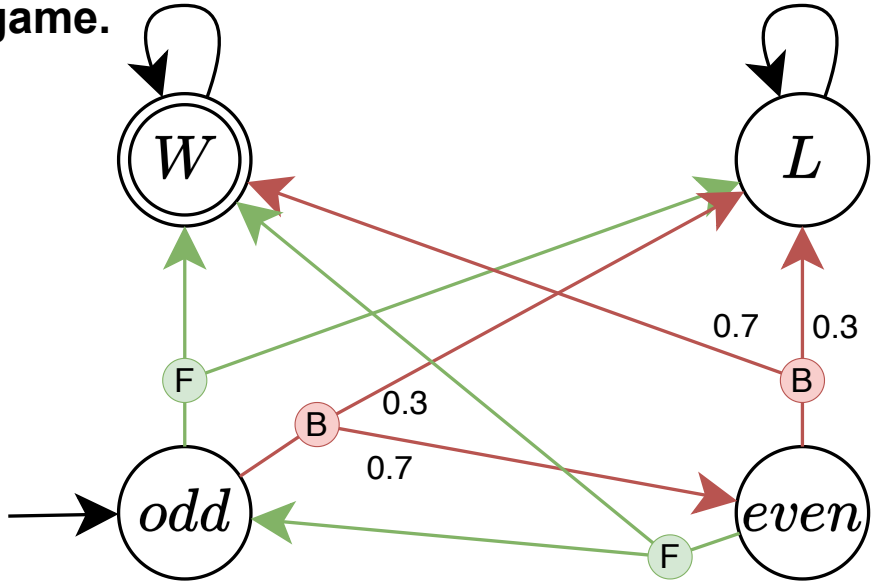
- verification: check whether all schedulers satisfy the specification
- synthesis (a dual problem): find a scheduler that satisfies the specification (or prove it does not exist)

Key idea: Compute minimum $P_{\min}(s_0 \to T)$ or maximum $P_{\max}(s_0 \to T)$ reachability probabilities over all schedulers

- corresponds to the analysis of the best- or worst-case behaviour of the MDP
- to compute min/max probabilities, it suffices to consider only **deterministic memoryless schedulers** (not true for all specifications, but holds for reachability)

Example: solving a coin toss game via MDP model checking

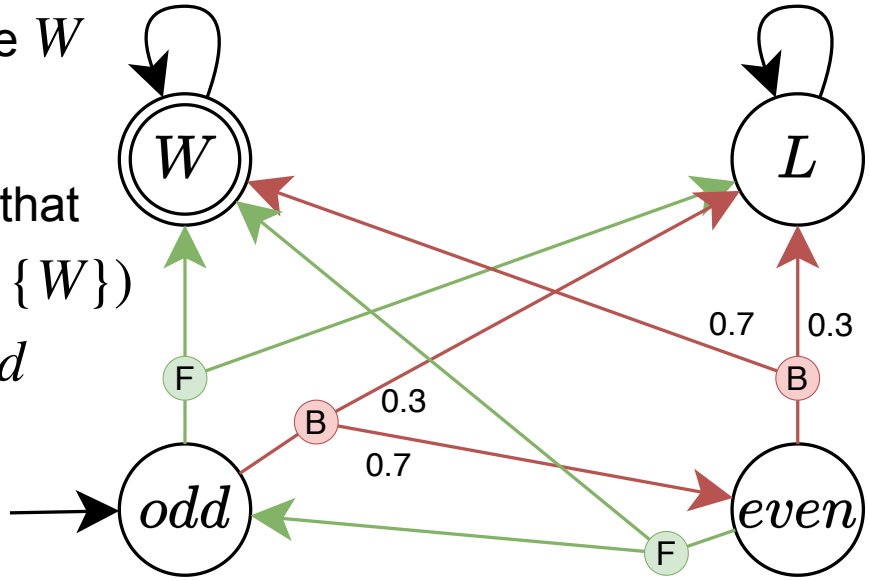
- Assume a coin toss game involving a fair $\left(P(heads_F) = 0.5\right)$ and a biased $\left(P(heads_B) = 0.7\right)$ coin. During each round, you can choose whether to toss a fair (F) or a biased (B) coin:
- - during odd rounds, you can toss a fair coin and either win or lose, or toss a biased coin and either continue the game (on heads) or lose (on tails)
- - during even rounds, you can toss a fair coin and either win or continue the game, or toss a biased coin and either win (on heads) or lose the game (on tails)
- **Find a strategy (coin selection during each round) that maximizes your chances of winning the game.**



BRNO UNIVERSITY OF TECHNOLOGY

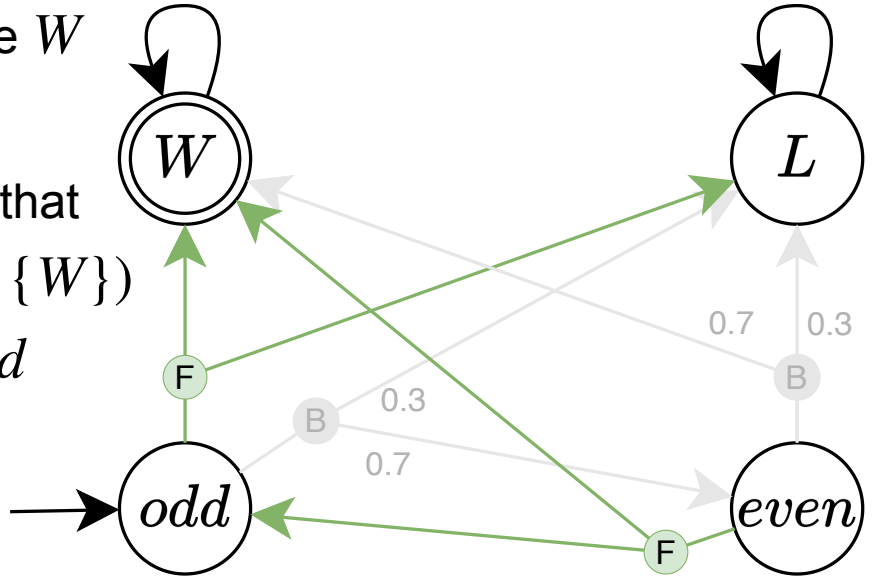
Example: solving a coin toss game via MDP model checking

- let x_s denote the probability of reaching state W from state s; clearly, $x_W = 1$ and $x_L = 0$
- we are interested in identifying scheduler σ that maximizes probability $x_{odd} = P_{\max}(odd \rightarrow \{W\})$ of reaching state W from the initial state odd
- we can enumerate over all **memoryless schedulers**:



Example: solving a coin toss game via MDP model checking

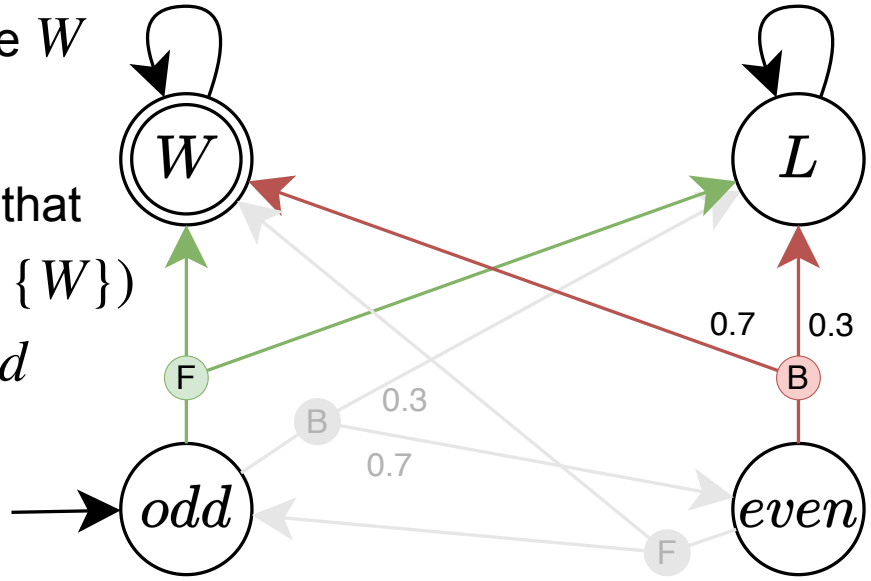
- let x_s denote the probability of reaching state W from state s; clearly, $x_W = 1$ and $x_L = 0$
- we are interested in identifying scheduler σ that maximizes probability $x_{odd} = P_{\max}(odd \rightarrow \{W\})$ of reaching state W from the initial state odd
- we can enumerate over all **memoryless schedulers**:
- $\sigma(odd) = F,\ \sigma(even) = * \implies x_{odd} = 0.5$



BRNO UNIVERSITY OF TECHNOLOGY

Example: solving a coin toss game via MDP model checking

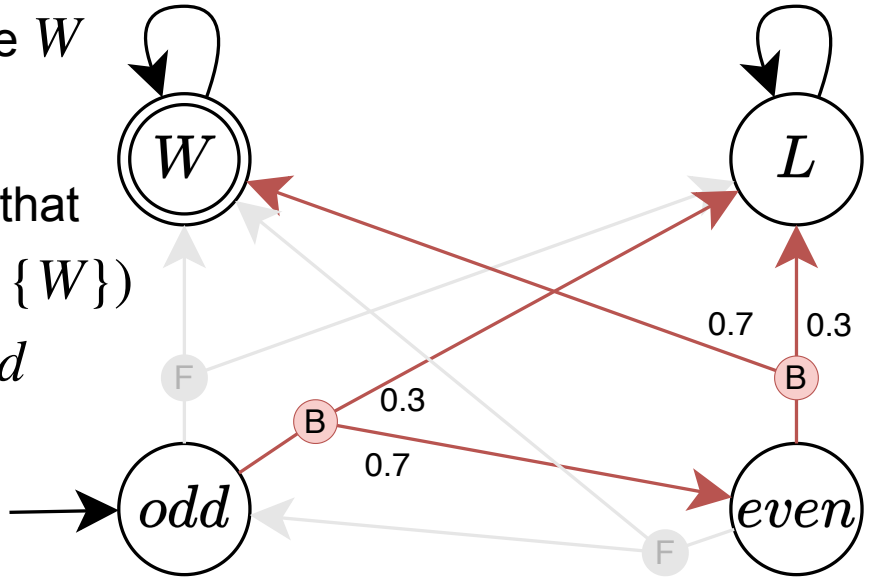
- let x_s denote the probability of reaching state W from state s; clearly, $x_W = 1$ and $x_L = 0$
- we are interested in identifying scheduler σ that maximizes probability $x_{odd} = P_{\max}(odd \rightarrow \{W\})$ of reaching state W from the initial state odd
- we can enumerate over all **memoryless schedulers**:
- $\sigma(odd) = F,\ \sigma(even) = * \implies x_{odd} = 0.5$



BRNO UNIVERSITY OF TECHNOLOGY

Example: solving a coin toss game via MDP model checking

- let x_s denote the probability of reaching state W from state s; clearly, $x_W = 1$ and $x_L = 0$
- we are interested in identifying scheduler σ that maximizes probability $x_{odd} = P_{\max}(odd \rightarrow \{W\})$ of reaching state W from the initial state odd
- we can enumerate over all **memoryless schedulers**:

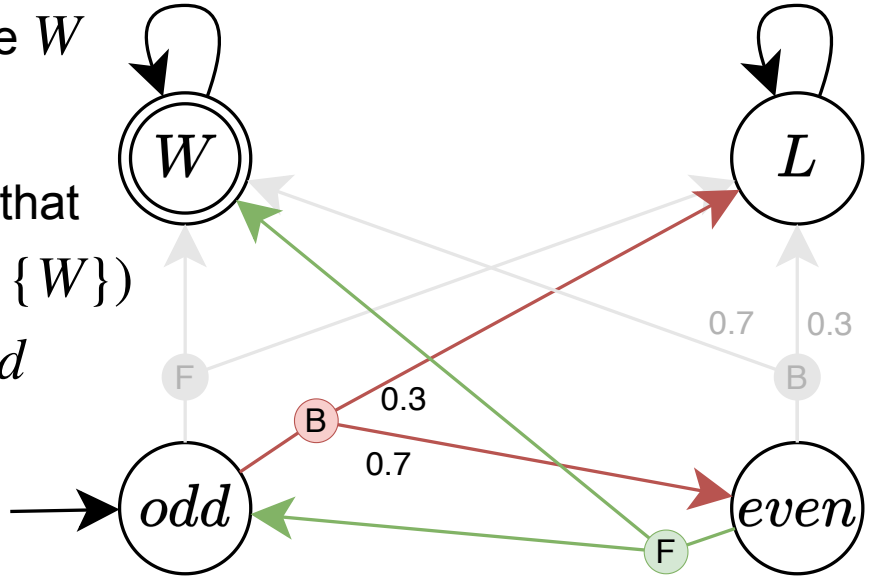


- $\sigma(odd) = F,\ \sigma(even) = * \implies x_{odd} = 0.5$
- $\sigma(odd) = B,\ \sigma(even) = B \implies x_{even} = 0.7,\ x_{odd} = 0.7 \cdot x_{even} = 0.49$

BRNO UNIVERSITY OF TECHNOLOGY

Example: solving a coin toss game via MDP model checking

- let x_s denote the probability of reaching state W from state s; clearly, $x_W = 1$ and $x_L = 0$
- we are interested in identifying scheduler σ that maximizes probability $x_{odd} = P_{\max}(odd \rightarrow \{W\})$ of reaching state W from the initial state odd
- we can enumerate over all **memoryless schedulers**:



- $\sigma(odd) = F,\ \sigma(even) = * \implies x_{odd} = 0.5$
- $\sigma(odd) = B,\ \sigma(even) = B \implies x_{even} = 0.7,\ x_{odd} = 0.7 \cdot x_{even} = 0.49$
- $\sigma(odd) = B,\ \sigma(even) = F \implies x_{odd} = 0.7 \cdot x_{even},\ x_{even} = 0.5 \cdot x_{odd} + 0.5 \implies$
  $\implies x_{even} \approx 0.76,\ x_{odd} \approx 0.53 \leftarrow$ **maximum probability achieved**

Model checking MDPs

Key idea: Compute minimum $P_{\min}(s_0 \to T)$ or maximum $P_{\max}(s_0 \to T)$ reachability probabilities over all schedulers

- corresponds to the analysis of the best- or worst-case behaviour of the MDP
- to compute min/max probabilities, it suffices to consider only **deterministic memoryless schedulers** (not true for all specifications, but holds for reachability)

Enumerating all memoryless schedulers is not tractable

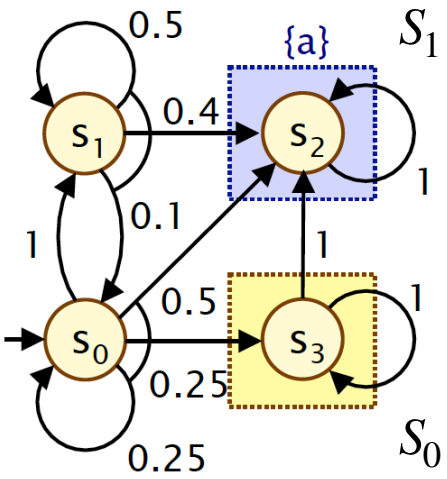
- there are $\mathcal{O}(|Act|^{|S|})$ memoryless schedulers
- more advanced techniques: reduction to linear programming, value iteration, policy iteration…

Reachability in MDPs Using Linear Programming

Compute $P_{\min}(s \to T)$ for all $s \in S$

Step 1: Identify states with probability 1 (S_1) and 0 (S_0) — graph-based problem

- intuition for probability 1: all paths over all schedulers lead to T
- intuition for probability 0: exists scheduler for which no path leads to T



Let $x_i = P_{\min}(s_i \to \{s_2\})$

$S_1 = \{s_2\}, S_0 = \{s_3\}$

$x_2 = 1,\ x_1 = 0$

Reachability in MDPs Using Linear Programming

Compute $P_{\min}(s \to T)$ for all $s \in S$

Step 1: Identify states with probability 1 (S_1) and 0 (S_0) — graph-based problem

- intuition for probability 1: all paths over all schedulers lead to T
- intuition for probability 0: exists scheduler for which no path leads to T

Step 2: For the remaining states $S_? = S \setminus (S_1 \cup S_0)$ solve the linear program:

$$\text{maximise } \sum_{s \in S_?} x_s \text{ subject to the constraints:}$$

$$x_s \leq \sum_{s' \in S_?} \mathbf{P}(s, \alpha, s') \cdot x_{s'} + \sum_{s' \in S_1} \mathbf{P}(s, \alpha, s') \cdot 1$$

$$0 \leq x_s \leq 1$$

$$\text{for all } s \in S_? \text{ and for all } \alpha \in Act(s)$$

- can be solved by standard techniques (e.g. simplex, branch-and-cut)

Example for $P_{\min}$

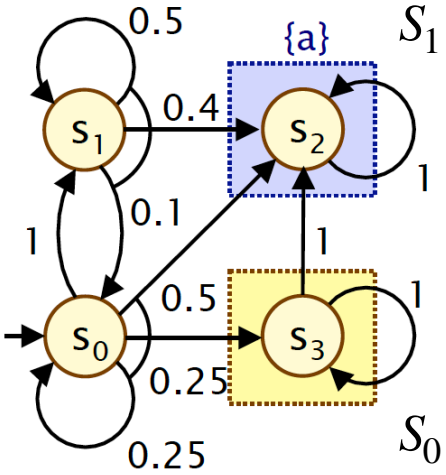
Step 2: For the remaining states $S_? = S \setminus (S_1 \cup S_0)$ solve the linear program:

maximise $\sum_{s \in S_?} x_s$ subject to the constraints:

$$x_s \leq \sum_{s' \in S_?} \mathbf{P}(s, \alpha, s') \cdot x_{s'} + \sum_{s' \in S_1} \mathbf{P}(s, \alpha, s') \cdot 1$$

$$0 \leq x_s \leq 1$$

for all $s \in S_?$ and for all $\alpha \in Act(s)$



maximise $x_0 + x_1$ subject to:

- $x_0 \leq x_1$
- $x_0 \leq 0.25x_0 + 0.5$
- $x_1 \leq 0.1x_0 + 0.5x_1 + 0.4$
- $0 \leq x_0, x_1 \leq 1$

BRNO UNIVERSITY OF TECHNOLOGY

Example for $P_{\min}$

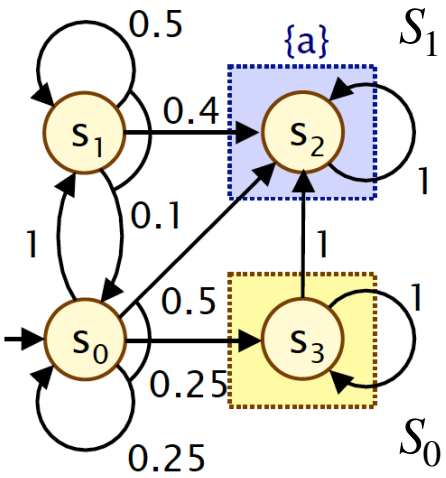
Step 2: For the remaining states $S_? = S \setminus (S_1 \cup S_0)$ solve the linear program:

maximise $\sum_{s \in S_?} x_s$ subject to the constraints:

$$x_s \leq \sum_{s' \in S_?} \mathbf{P}(s, \alpha, s') \cdot x_{s'} + \sum_{s' \in S_1} \mathbf{P}(s, \alpha, s') \cdot 1$$

$$0 \leq x_s \leq 1$$

for all $s \in S_?$ and for all $\alpha \in Act(s)$

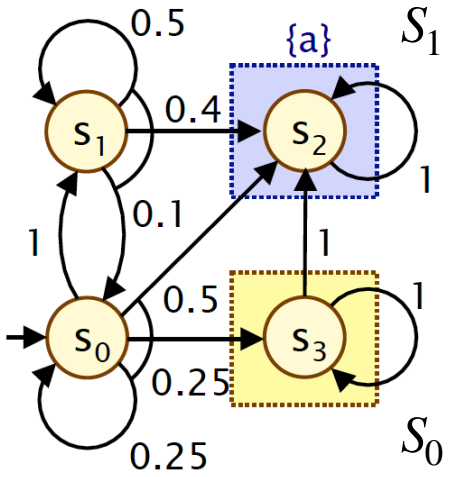


maximise $x_0 + x_1$ subject to:

- $x_0 \leq x_1$
- $x_0 \leq 2/3$
- $x_1 \leq 0.2x_0 + 0.8$
- $0 \leq x_0, x_1 \leq 1$

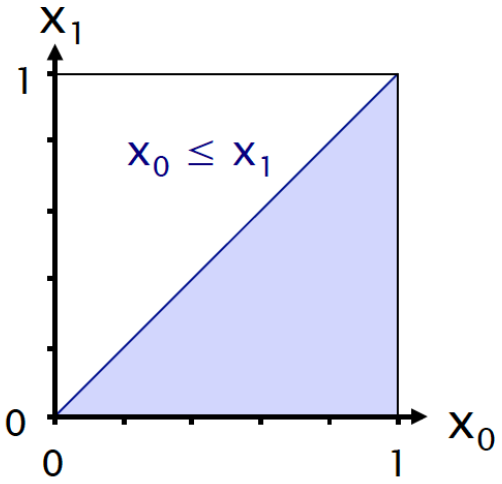
BRNO UNIVERSITY OF TECHNOLOGY

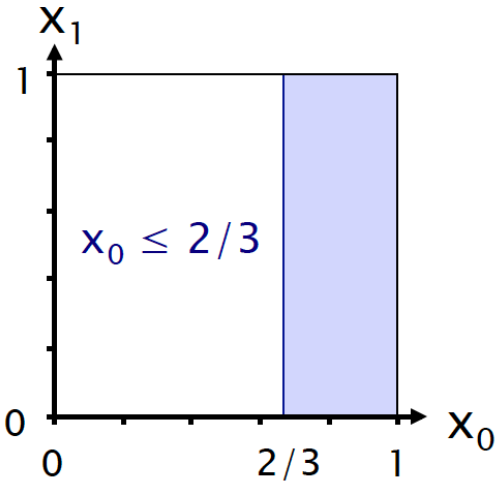
Example for $P_{\min}$

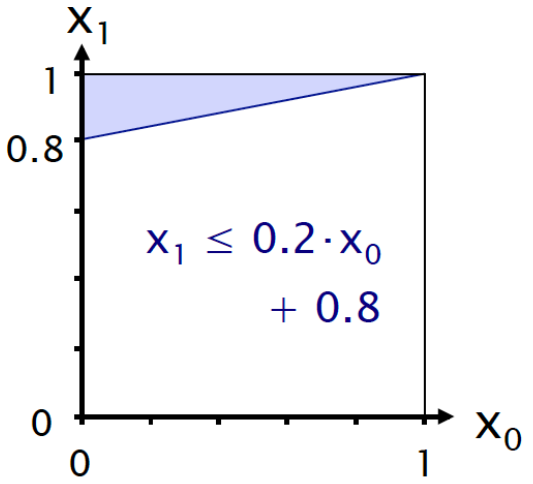


maximise $x_0 + x_1$ subject to:

- $x_0 \leq x_1$
- $x_0 \leq 2/3$
- $x_1 \leq 0.2x_0 + 0.8$
- $0 \leq x_0, x_1 \leq 1$

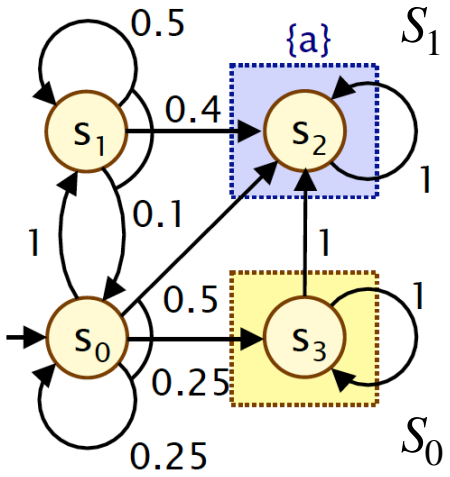






BRNO UNIVERSITY OF TECHNOLOGY

Example for $P_{\min}$

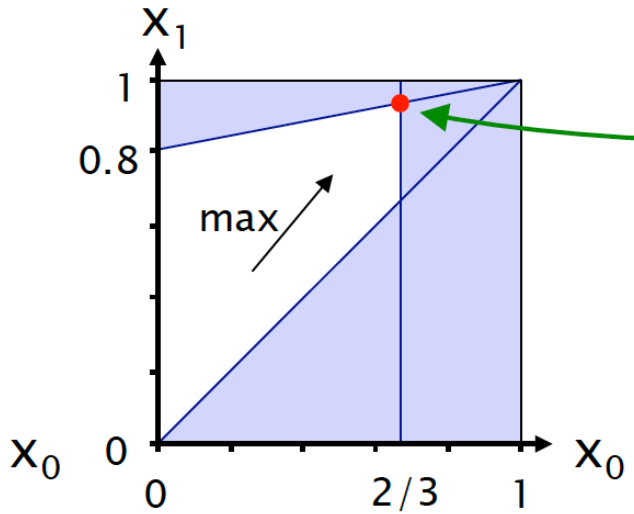


maximise $x_0 + x_1$ subject to:

- $x_0 \leq x_1$
- $x_0 \leq 2/3$
- $x_1 \leq 0.2x_0 + 0.8$
- $0 \leq x_0, x_1 \leq 1$

Interpretation:

- schedulers lie on the intersections of the borders of the constraints (for each action one constraint)
- maximisation of the sum ensures reaching the "optimal" intersection



Solution:

(x_0, x_1)

=

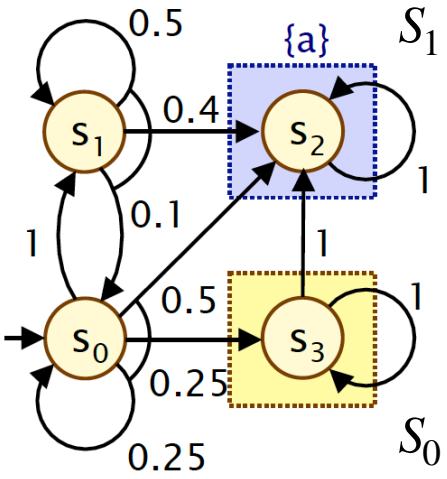
(2/3, 14/15)

Reachability in MDPs Using Linear Programming

Compute $P_{\max}(s \to T)$ for all $s \in S$

Step 1: Identify states with probability 1 (S_1) and 0 (S_0) — graph-based problem

- intuition for probability 1: exists scheduler for which all paths to T
- intuition for probability 0: for all schedulers there is no path leading to T



Let $x_i = P_{\max}(s_i \to \{s_2\})$

$S_1 = \{s_0, s_1, s_2, s_3\}$

$x_0 = x_1 = x_2 = x_3 = 1$

BRNO UNIVERSITY OF TECHNOLOGY

Reachability in MDPs Using Linear Programming

Compute $P_{\max}(s \to T)$ for all $s \in S$

Step 1: Identify states with probability 1 (S_1) and 0 (S_0) — graph-based problem

- intuition for probability 1: exists scheduler for which all paths to T
- intuition for probability 0: for all schedulers there is no path leading to T

Step 2: For the remaining states $S_? = S \setminus (S_1 \cup S_0)$ solve the linear program:

$$\text{minimise } \sum_{s \in S_?} x_s \text{ subject to the constraints:}$$

$$x_s \geq \sum_{s' \in S_?} \mathbf{P}(s, \alpha, s') \cdot x_{s'} + \sum_{s' \in S_1} \mathbf{P}(s, \alpha, s') \cdot 1$$

$$0 \leq x_s \leq 1$$

$$\text{for all } s \in S_? \text{ and for all } \alpha \in Act(s)$$

- can be solved by standard techniques (e.g. simplex, branch-and-cut)

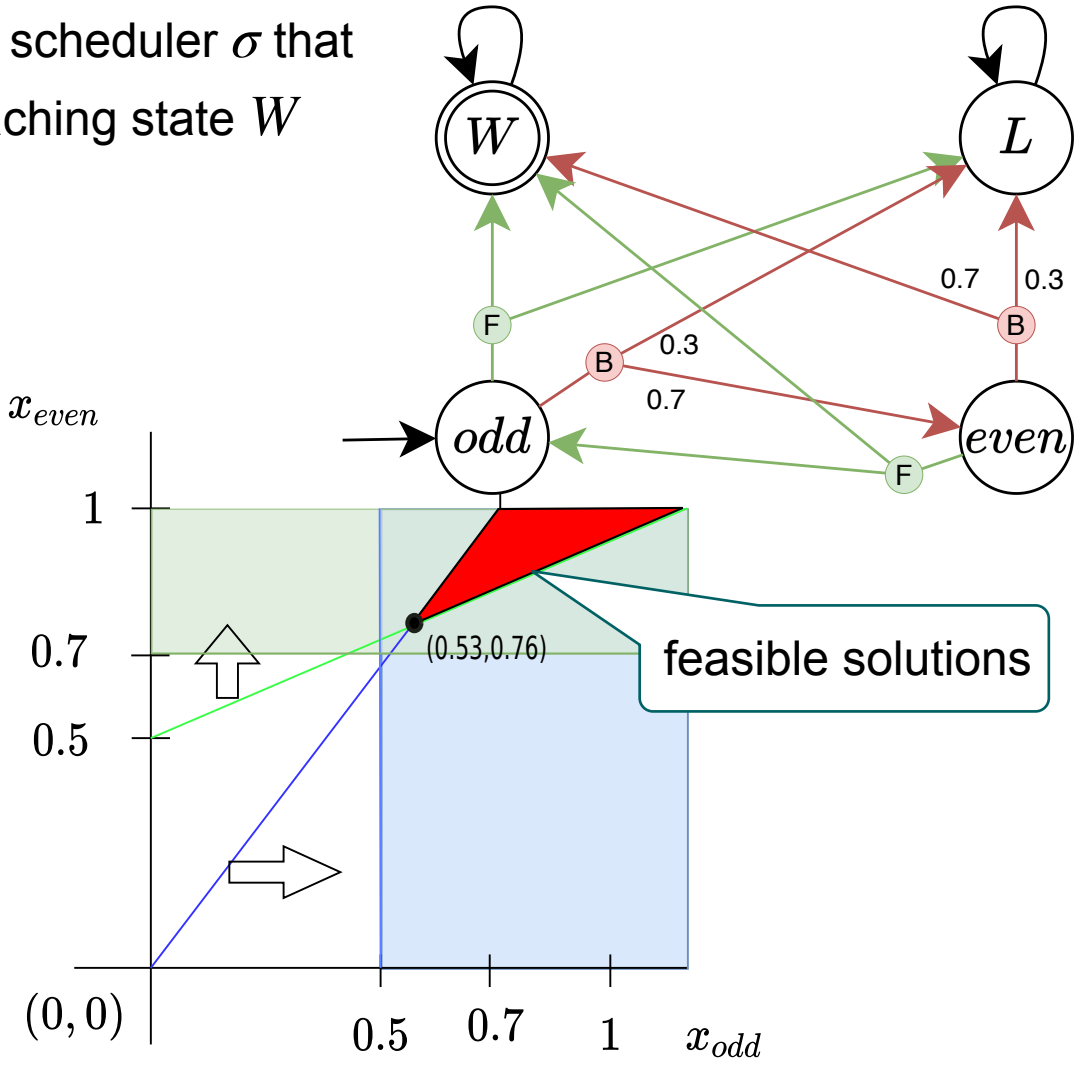
Example for $P_{\max}$: solving a coin toss game

- we are interested in identifying a scheduler σ that maximizes probability x_{odd} of reaching state W from the initial state odd
- **the linear program:**

$S_1 : x_W = 1 \quad S_0 : x_L = 0$

min $x_{odd} + x_{even}$ subject to

$$x_{odd} \geq 0.5$$
$$x_{odd} \geq 0.7x_{even}$$
$$x_{even} \geq 0.5 + 0.5x_{odd}$$
$$x_{even} \geq 0.7$$



BRNO UNIVERSITY OF TECHNOLOGY

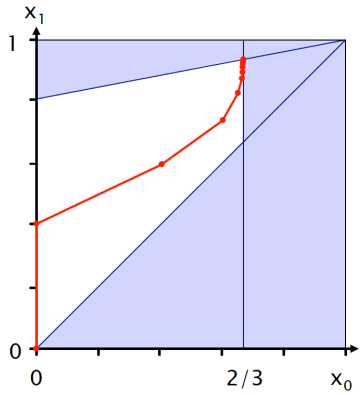
Reachability in MDPs Using Iterative Approach

Value iteration

$$P_{\min}(s \to T) = \lim_{n \to \infty} x_s^{(n)}$$

$$x_s^{(n)} = \begin{cases} 1 & \text{if } s \in T \\ 0 & \text{if } s \in S_{no} \vee n = 0 \\ \min_{\alpha \in Act(s)} \left(\sum_{s' \in S} \mathbf{P}(s, \alpha, s') \cdot x_{s'}^{(n-1)} \right) & \text{otherwise} \end{cases}$$

- similarly for the $P_{\max}(s \to T)$
  - Iterative (approximate) solution
  - iterates over (vectors of) probabilities
  - terminates if converged sufficiently



BRNO UNIVERSITY OF TECHNOLOGY

Value iteration — example

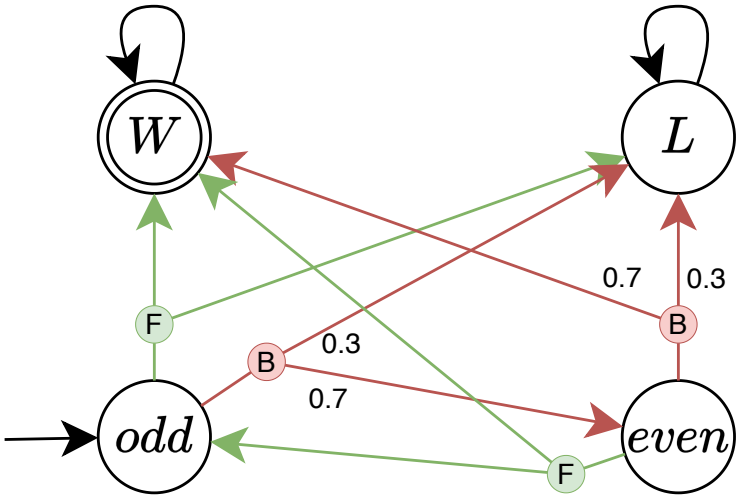
$$x_s^{(n)} = \begin{cases} 1 & \text{if } s \in T \\ 0 & \text{if } s \in S_{no} \vee n = 0 \\ \max_{\alpha \in Act(s)} \left(\sum_{s' \in S} \mathbf{P}(s, \alpha, s') \cdot x_{s'}^{(n-1)} \right) & \text{otherwise} \end{cases}$$

maximising the reachability

- $x^{(0)}(odd, even, W, L) = (0,\ 0,\ 1,\ 0)$

- $x^{(1)}(odd) = \max\{0.5 \cdot 1,\ 0\} = 0.5$
- $x^{(1)}(even) = \max\{0.5 \cdot 1,\ 0.7 \cdot 1\} = 0.7$
- $x^{(1)}(W) = 1$
- $x^{(1)}(L) = 0$

- $x^{(1)} = (0.5,\ 0.7,\ 1,\ 0)$



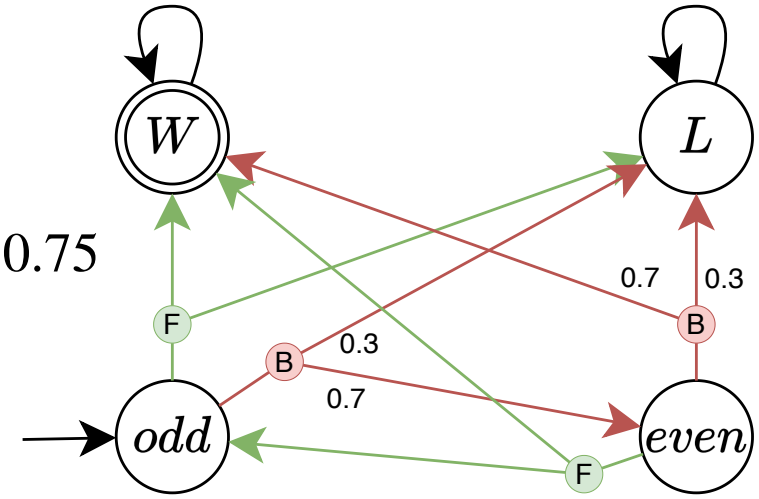
Value iteration — example

$$x_s^{(n)} = \begin{cases} 1 & \text{if } s \in T \\ 0 & \text{if } s \in S_{no} \vee n = 0 \\ \max_{\alpha \in Act(s)} \left(\sum_{s' \in S} \mathbf{P}(s, \alpha, s') \cdot x_{s'}^{(n-1)} \right) & \text{otherwise} \end{cases}$$

- $x^{(1)}(odd, even, W, L) = (0.5,\ 0.7,\ 1,\ 0)$

- $x^{(2)}(odd) = \max\{0.5 \cdot 1,\ 0.7 \cdot 0.7\} = 0.5$
- $x^{(2)}(even) = \max\{0.5 \cdot 0.5 + 0.5 \cdot 1,\ 0.7 \cdot 1\} = 0.75$
- $x^{(2)}(W) = 1$
- $x^{(2)}(L) = 0$

- $x^{(2)} = (0.5,\ 0.75,\ 1,\ 0)$

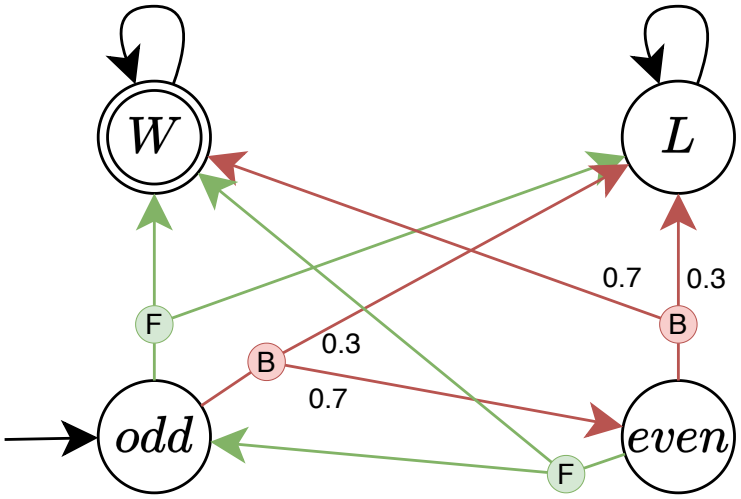


Value iteration — example

$$x_s^{(n)} = \begin{cases} 1 & \text{if } s \in T \\ 0 & \text{if } s \in S_{no} \vee n = 0 \\ \min_{\alpha \in Act(s)} \left(\sum_{s' \in S} \mathbf{P}(s, \alpha, s') \cdot x_{s'}^{(n-1)} \right) & \text{otherwise} \end{cases}$$

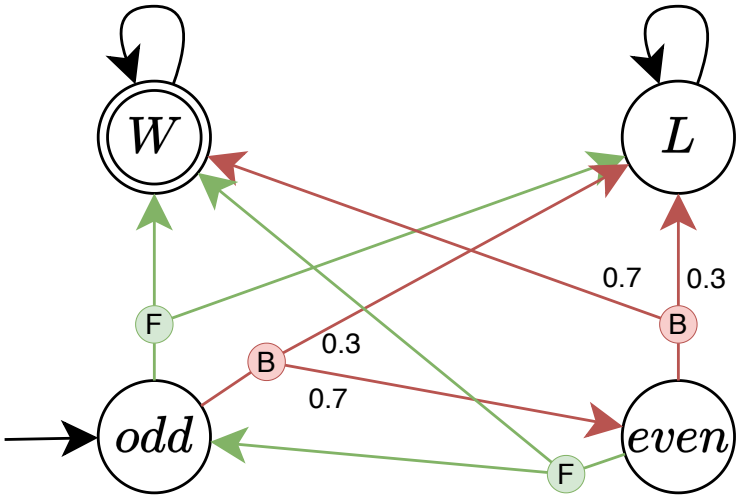
- $x^{(0)}(odd, even, W, L) = (0,\ 0,\ 1,\ 0)$

$$\begin{aligned} x^{(1)} &= (0.500,\ 0.700,\ 1,\ 0) \\ x^{(2)} &= (0.500,\ 0.750,\ 1,\ 0) \\ x^{(3)} &= (0.525,\ 0.750,\ 1,\ 0) \\ x^{(4)} &\approx (0.525,\ 0.762,\ 1,\ 0) \\ x^{(5)} &\approx (0.533,\ 0.762,\ 1,\ 0) \\ &\vdots \\ x^{(15)} = x^{(14)} &\approx (0.538,\ 0.769,\ 1,\ 0) \end{aligned}$$



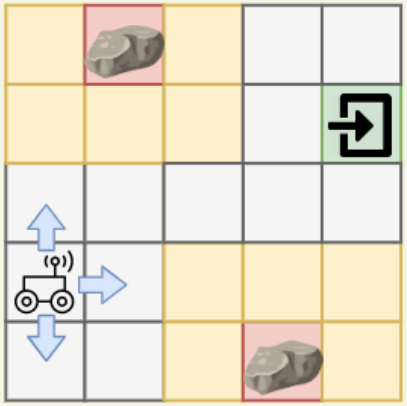
Value iteration — example

- optimal reachability probabilities: $x^{(15)} = x^{(14)} \approx (0.538,\ 0.769,\ 1,\ 0)$
- to reconstruct the optimal action, observe which action is picked during the last iteration:
- $x^{(15)}(odd) = \max\{0.5 \cdot 1,\ 0.7 \cdot 0.769\} = \max\{0.5,\ 0.538\} \approx 0.538$
  $\Rightarrow \sigma(odd) = B$
- $x^{(15)}(even) = \max\{0.5 \cdot 0538 + 0.5 \cdot 1,\ 0.7 \cdot 1\} = \max\{0.769,\ 0.7\} \approx 0.769$
  $\Rightarrow \sigma(even) = F$



Advanced Problems Related to MDPs

- **Partially observable Markov decision processes (POMDPs)** – important model for sequential decision-making under uncertainty and limited state observability

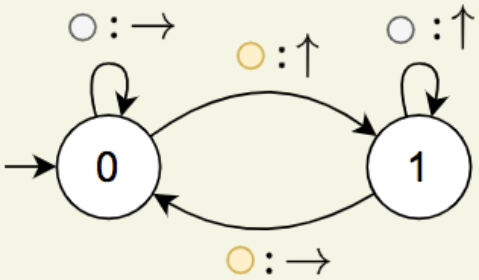


- **Indefinite-horizon specifications** given as conjunction of temporal constraints and a single optimisation objective (no discounting)

$$\text{constraint} : \mathbb{P}_{\leq 0.1}\,[\mathrm{F}\ \mathit{crash}]$$

$$\text{objective} : \text{minimize}\ \mathbb{R}^{\mathit{steps}}\,[\mathrm{F}\ \mathit{exit}]$$

- **Finite-state controllers (FSCs)** – compact automata-based representation of policies (deterministic Mealy machines where output is determined by the taken transition)



State-of-the-Art Tools for Analysis of Probabilistic Systems

PRISM (short demo)

STORM

- supports effective analysis of a wide class of probabilistic models

MODEST

- focuses on more complex modelling formalism (stochastic hybrid automata): hybrid, real-time, distributed and stochastic systems

PASS and STAMINA: focus on infinite-state probabilistic models

DFTCalc: focuses on failure probability of Dynamic Fault Trees

MRMC: supports bisimulation and continuous-time MDPs

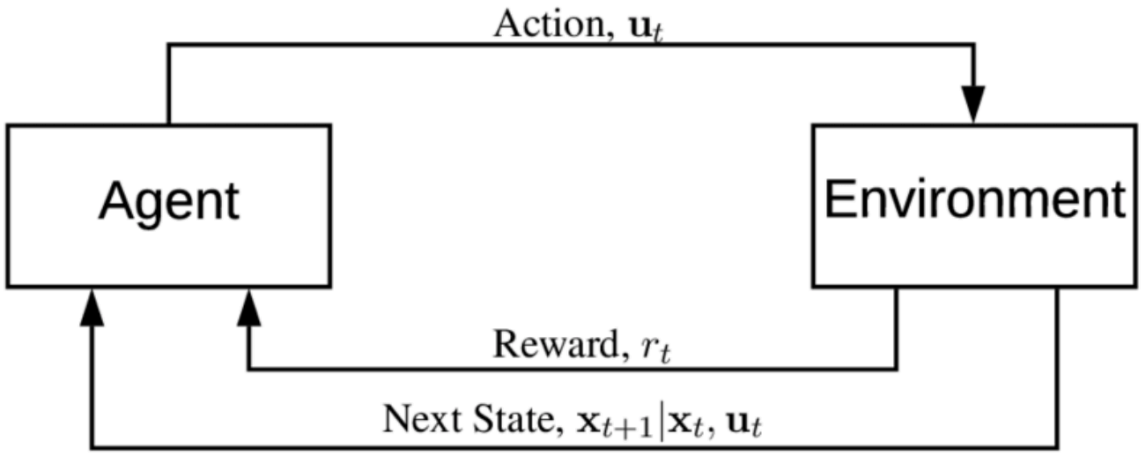
Uppall SMC: tool for statistical model-checking

GreatSPN: tool for Generalized Stochastic Petri Nets

SARSOP, DESPOT, and PAYNT: focus on POMDPs

Connection to reinforcement learning

What to do if the environment (MDP) is not known or it is too complex?



Exploration vs. exploitation problem

A great book by Sutton & Barto: ***Reinforcement Learning: An Introduction***

Useful code base: https://github.com/Svalorzen/AI-Toolbox