Safe Reinforcement Learning

Florent Delgrange

UMONS Université de Mons

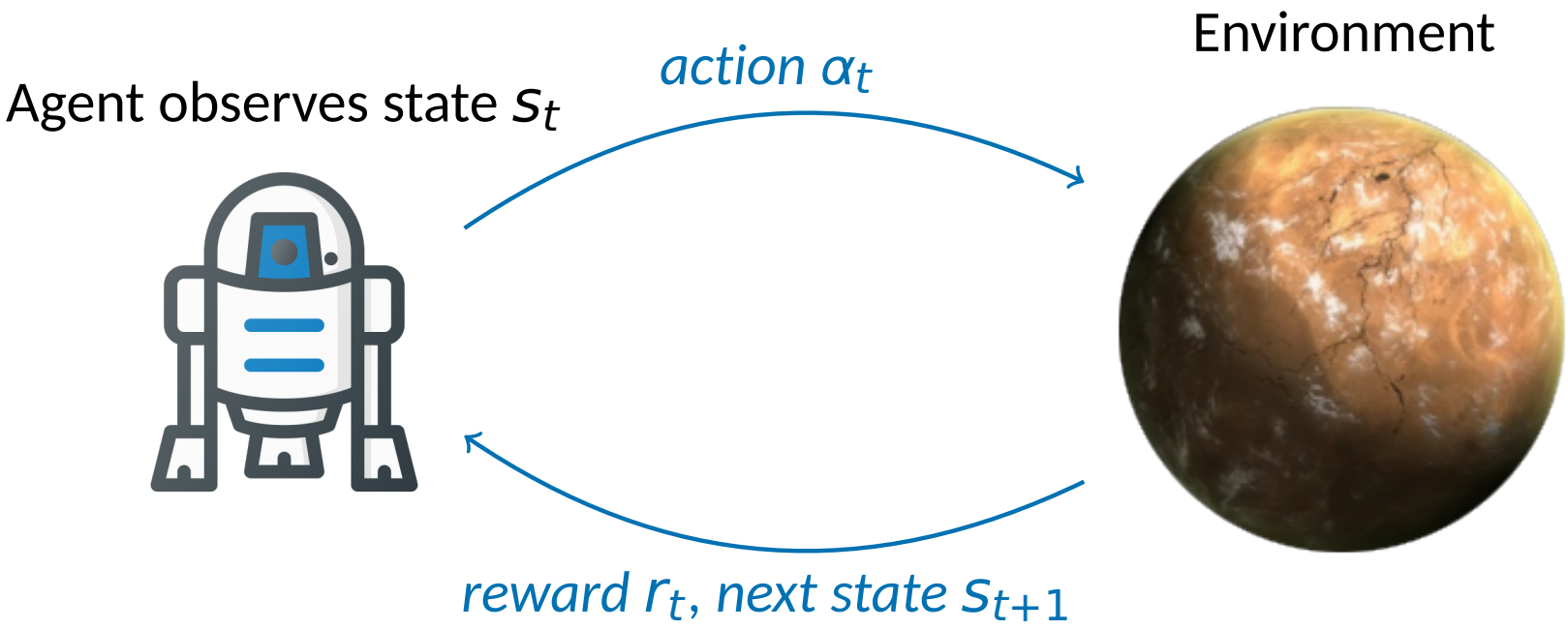
RWTH AACHEN UNIVERSITY

October 19, 2018

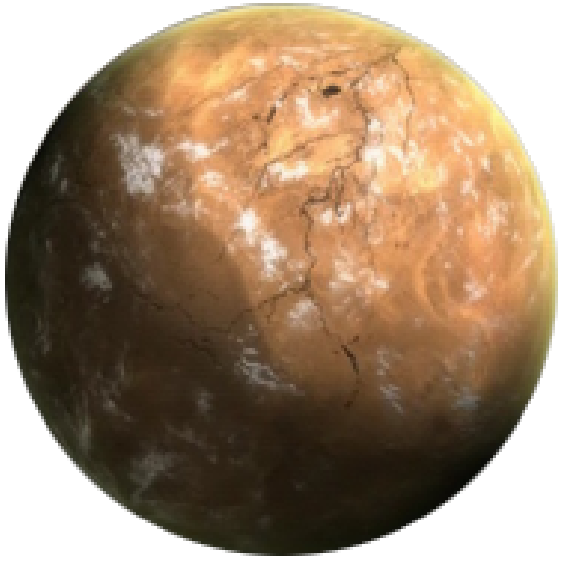
L^2 – Logic meets Learning

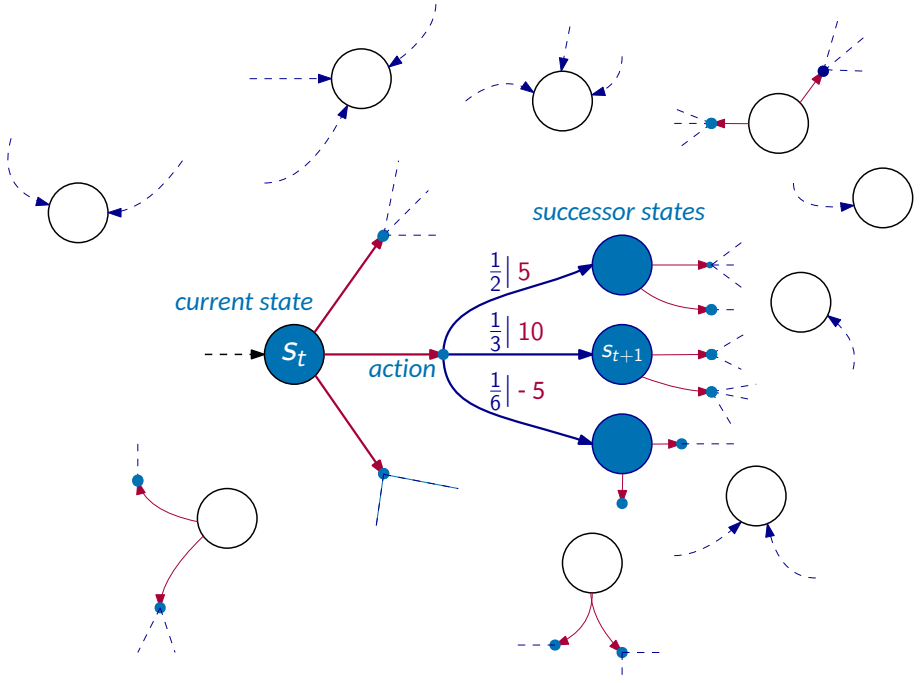
Reinforcement learning

- **Configuration**: agent interacting with its environment
- On each step t of interaction



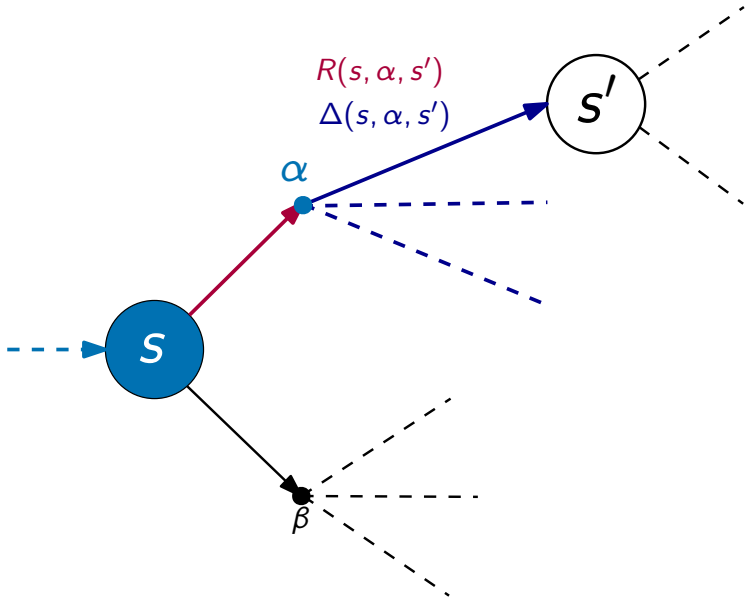
Modelling the Environment





- We assume the environment to be **stochastic**
- Can be modelled as a ***Markov Decision Process (MDP)***
  - ⇝ **system** modelling both **probabilistic** and **nondeterministic** situations

Markov Decision Process (MDP)



An MDP $\mathcal{M} = (S, A, \Delta, R)$ is a tuple such that

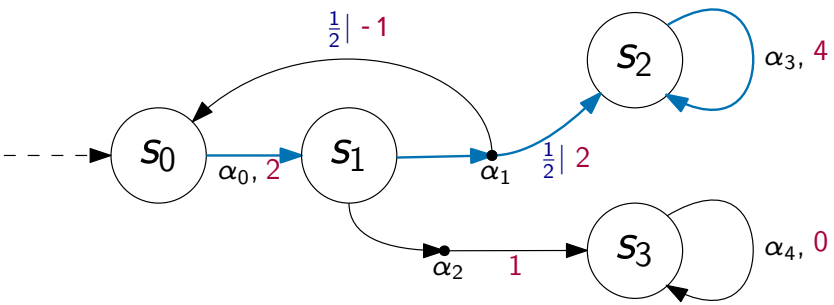
- S is the **set of states** of the system
- A is the **set of actions** of the system
  - ⇝ for all $s \in S$, $A(s) \subseteq A$ is the set of enabled actions of A
- $\Delta : S \times A \times S \to [0, 1]$ is the **probability transition function**
- $R : S \times A \times S \to \mathbb{R}$ is the **reward function**

Discounted Sum

Let $\gamma \in]0,1[$ be a *discount factor* and
$\pi = s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} s_2 \xrightarrow{\alpha_2} \cdots \in Paths(\mathcal{M})$,

$$\begin{aligned} \mathrm{Disc}_\gamma(\pi) &= R(s_0,\alpha_0,s_1) + \gamma \cdot R(s_1,\alpha_1,s_2) + \gamma^2 \cdot R(s_2,\alpha_2,s_3) + \ldots \\ &= \sum_{t\in\mathbb{N}} \gamma^t \cdot R(s_t,\alpha_t,s_{t+1}) \end{aligned}$$

- **Idea of γ:** short terms rewards have more importance that long term ones
- $\gamma \approx 1 \rightsquigarrow$ **Total Payoff**
- $\gamma \approx 0 \rightsquigarrow$ **Fully Greedy**



- $\pi = s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} (s_2 \xrightarrow{\alpha_3})^\omega$
- $\gamma = \frac{1}{2}$
- $\mathrm{Disc}_\gamma(\pi)$
  $= 2 + \frac{1}{2} \cdot 2 + \sum_{t=2}^{\infty} (\frac{1}{2})^t \cdot 4$
  $= 3 + (\frac{4}{1-\frac{1}{2}} - (4 + \frac{1}{2} \cdot 4))$
  $= 3 + 2 = 5$

Stochastic measure

$$\sigma^* := \arg\max_{\sigma} \mathbb{E}^{\sigma}_{\mathcal{M}}(Disc_\gamma)$$

Bellman equation: expected discounted rewards

Maximal expected discounted rewards from any state $s \in S$ of the system

- $\mathbb{E}^{\max}_{s}(Disc_\gamma) = V^*(s) = \max_{\alpha \in A(s)} Q^*(s,\alpha)$
- $Q^*(s,\alpha) = \sum_{s' \in S} \Delta(s,\alpha,s') \cdot \big(R(s,\alpha,s') + \gamma \cdot V^*(s')\big)$
- **Optimal memoryless strategy**:

$$\sigma^*(s) = \arg\max_{\alpha \in A(s)} Q^*(s,\alpha)$$

with $\mathbb{E}^{\sigma^*}_{s}(Disc_\gamma) = \mathbb{E}^{\max}_{s}(Disc_\gamma)$

⇝ V^* can be computed by **dynamic programming** with the **value iteration algorithm**

Value Iteration

Bellman equation: expected discounted rewards

- $\mathbb{E}_s^{\max}(Disc_\gamma) = V^*(s) = \max_{\alpha \in A(s)} Q^*(s,\alpha)$ for all $s \in S$
- $Q^*(s,\alpha) = \sum_{s' \in S} \Delta(s,\alpha,s') \cdot \left(R(s,\alpha,s') + \gamma \cdot V^*(s')\right)$

- At each step $t \in \mathbb{N}_0$, for each $s \in S$ and $\alpha \in A(s)$,

$$V_0(s) = 0$$

$$Q_{t+1}(s,\alpha) = \sum_{s' \in S} \Delta(s,\alpha,s') \cdot \left(R(s,\alpha,s') + \gamma \cdot V_t(s')\right)$$

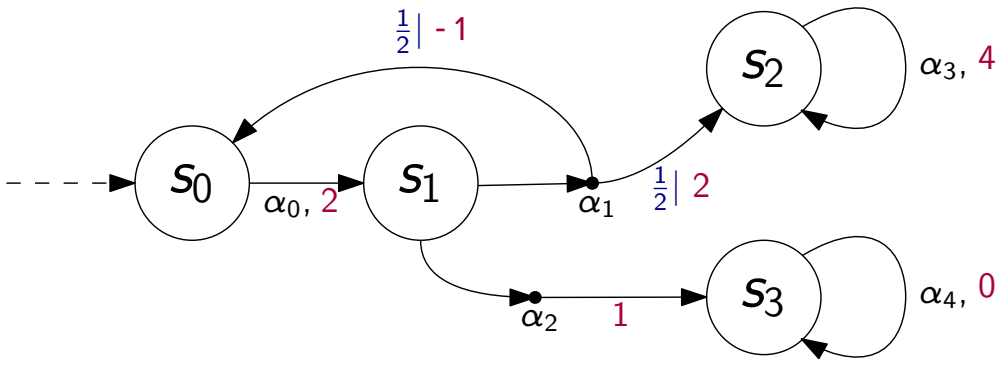
$$V_{t+1}(s) = \max_{\alpha \in A(s)} Q_{t+1}(s,\alpha)$$

- $\mathbb{E}_s^{\max}(Disc_\gamma) = V^*(s) = \lim_{t \to \infty} V_t(s)$
- **In practice:** stop when a **stopping criterion** is reached
  e.g., $|V_{t+1}(s) - V_t(s)| < \epsilon$, for all $s \in S$, with $\epsilon > 0$.

Value Iteration

Expected discounted rewards

$$Q_{t+1}(s,\alpha) = \sum_{s' \in S} \Delta(s,\alpha,s') \cdot \left(R(s,\alpha,s') + \gamma \cdot V_i(s')\right), \; V_{t+1}(s) = \max_{\alpha \in A(s)} Q_{t+1}(s,\alpha)$$



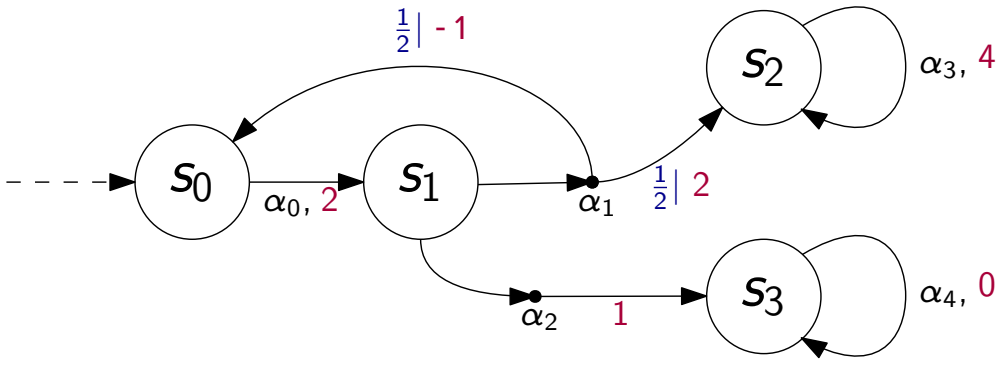
- $\gamma = \frac{1}{2}$

V	s_0	s_1	s_2	s_3
it #0	0	0	0	0
it #1				

Value Iteration

Expected discounted rewards

$$Q_{t+1}(s,\alpha) = \sum_{s' \in S} \Delta(s,\alpha,s') \cdot \left(R(s,\alpha,s') + \gamma \cdot V_i(s')\right), \; V_{t+1}(s) = \max_{\alpha \in A(s)} Q_{t+1}(s,\alpha)$$



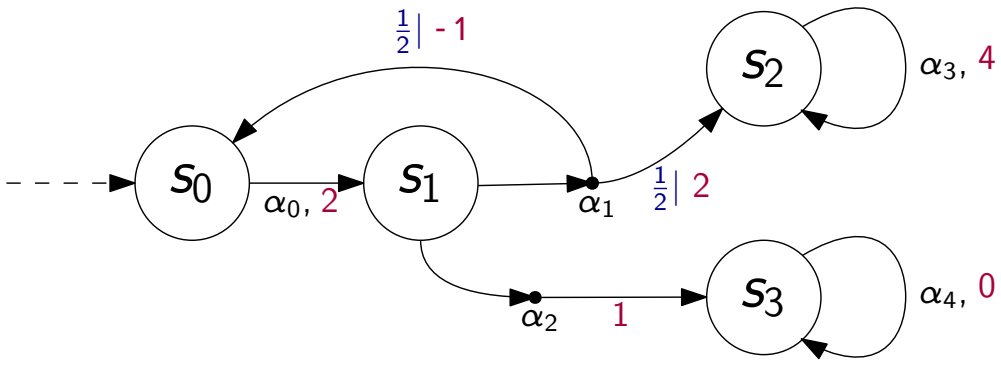
- $\gamma = \frac{1}{2}$

V	s_0	s_1	s_2	s_3
it #0	0	0	0	0
it #1	2			

Value Iteration

Expected discounted rewards

$$Q_{t+1}(s,\alpha) = \sum_{s' \in S} \Delta(s,\alpha,s') \cdot \left(R(s,\alpha,s') + \gamma \cdot V_i(s')\right), \; V_{t+1}(s) = \max_{\alpha \in A(s)} Q_{t+1}(s,\alpha)$$



- $\gamma = \frac{1}{2}$

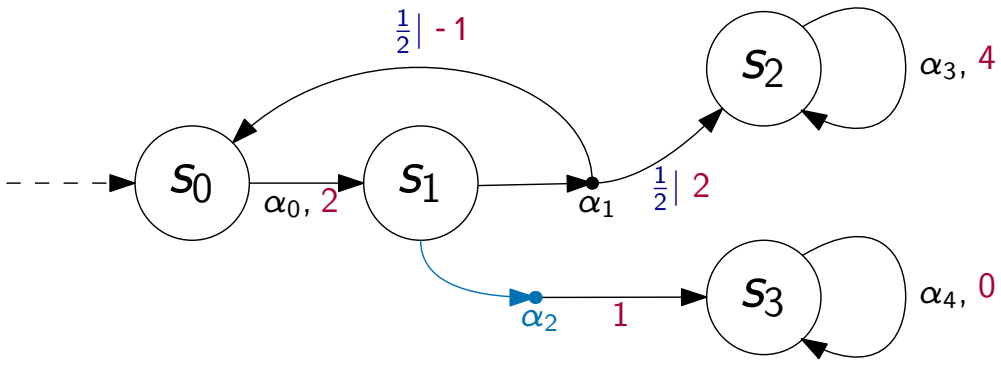
V	s_0	s_1	s_2	s_3
it #0	0	0	0	0
it #1	2			

- $Q_1(s_1,\alpha_1) = \frac{1}{2} \cdot (-1) + \frac{1}{2} \cdot 2 = 0.5$
- $Q_1(s_1,\alpha_2) = 1$

Value Iteration

Expected discounted rewards

$$Q_{t+1}(s,\alpha) = \sum_{s' \in S} \Delta(s,\alpha,s') \cdot \left(R(s,\alpha,s') + \gamma \cdot V_i(s')\right), \; V_{t+1}(s) = \max_{\alpha \in A(s)} Q_{t+1}(s,\alpha)$$



- $\gamma = \frac{1}{2}$

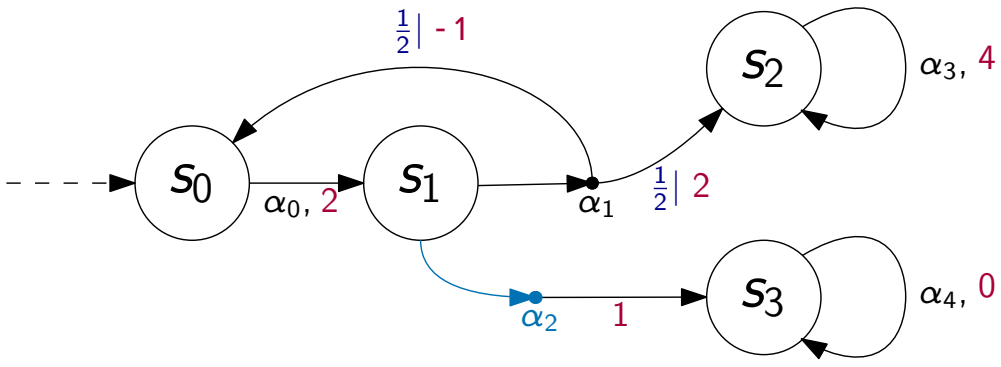
V	s_0	s_1	s_2	s_3
it #0	0	0	0	0
it #1	2	1		

- $Q_1(s_1,\alpha_1) = \frac{1}{2} \cdot (-1) + \frac{1}{2} \cdot 2 = 0.5$
- $Q_1(s_1,\alpha_2) = 1$

Value Iteration

Expected discounted rewards

$$Q_{t+1}(s,\alpha) = \sum_{s' \in S} \Delta(s,\alpha,s') \cdot \left(R(s,\alpha,s') + \gamma \cdot V_i(s')\right), \; V_{t+1}(s) = \max_{\alpha \in A(s)} Q_{t+1}(s,\alpha)$$



- $\gamma = \frac{1}{2}$

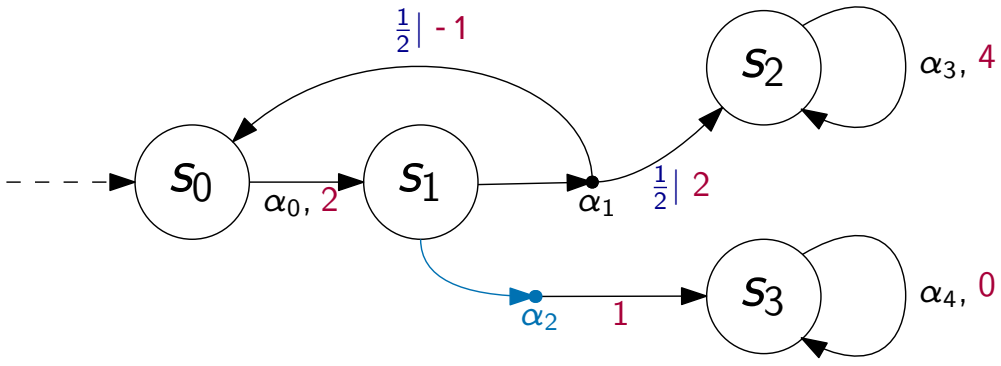
V	s_0	s_1	s_2	s_3
it #0	0	0	0	0
it #1	2	1	4	0

- $Q_1(s_1,\alpha_1) = \frac{1}{2} \cdot (-1) + \frac{1}{2} \cdot 2 = 0.5$
- $Q_1(s_1,\alpha_2) = 1$

Value Iteration

Expected discounted rewards

$$Q_{t+1}(s,\alpha) = \sum_{s' \in S} \Delta(s,\alpha,s') \cdot \left(R(s,\alpha,s') + \gamma \cdot V_i(s')\right), \; V_{t+1}(s) = \max_{\alpha \in A(s)} Q_{t+1}(s,\alpha)$$



- $\gamma = \frac{1}{2}$

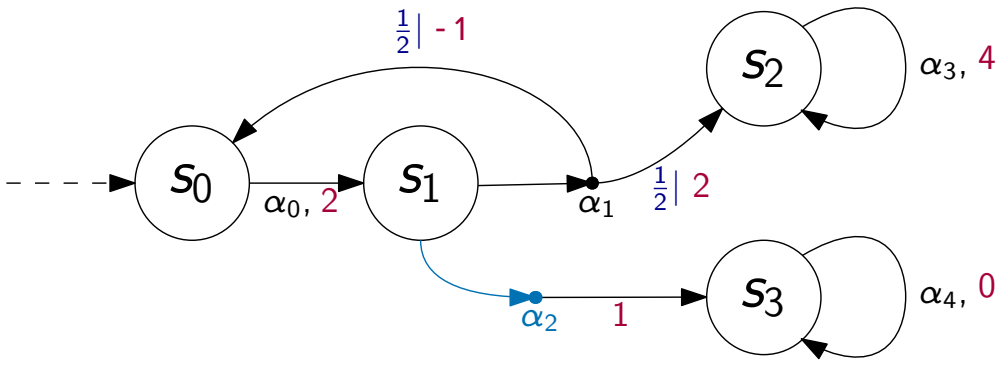
V	s_0	s_1	s_2	s_3
it #0	0	0	0	0
it #1	2	1	4	0
it #2				

- $Q_1(s_1,\alpha_1) = \frac{1}{2} \cdot (-1) + \frac{1}{2} \cdot 2 = 0.5$
- $Q_1(s_1,\alpha_2) = 1$

Value Iteration

Expected discounted rewards

$$Q_{t+1}(s,\alpha) = \sum_{s' \in S} \Delta(s,\alpha,s') \cdot \left(R(s,\alpha,s') + \gamma \cdot V_i(s')\right), \; V_{t+1}(s) = \max_{\alpha \in A(s)} Q_{t+1}(s,\alpha)$$



- $\gamma = \frac{1}{2}$

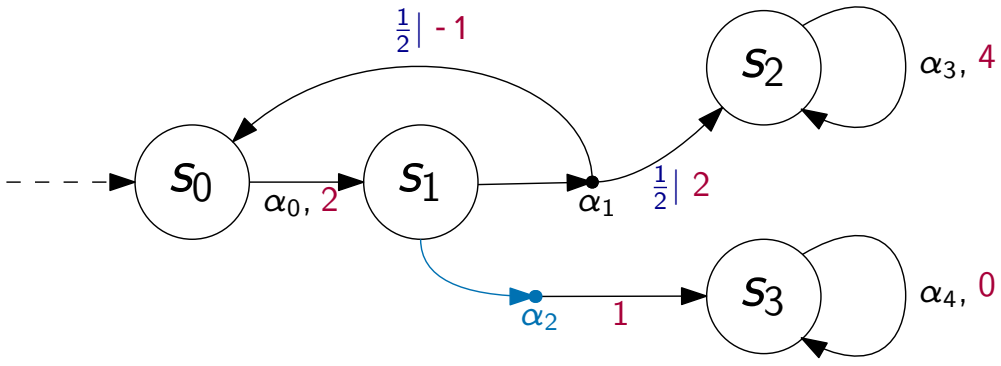
V	s_0	s_1	s_2	s_3
it #0	0	0	0	0
it #1	2	1	4	0
it #2	2.5			

$$V_2(s_0) = 2 + \gamma \cdot 1 = 2.5$$

Value Iteration

Expected discounted rewards

$$Q_{t+1}(s,\alpha) = \sum_{s' \in S} \Delta(s,\alpha,s') \cdot \left(R(s,\alpha,s') + \gamma \cdot V_i(s')\right), \; V_{t+1}(s) = \max_{\alpha \in A(s)} Q_{t+1}(s,\alpha)$$



- $\gamma = \frac{1}{2}$

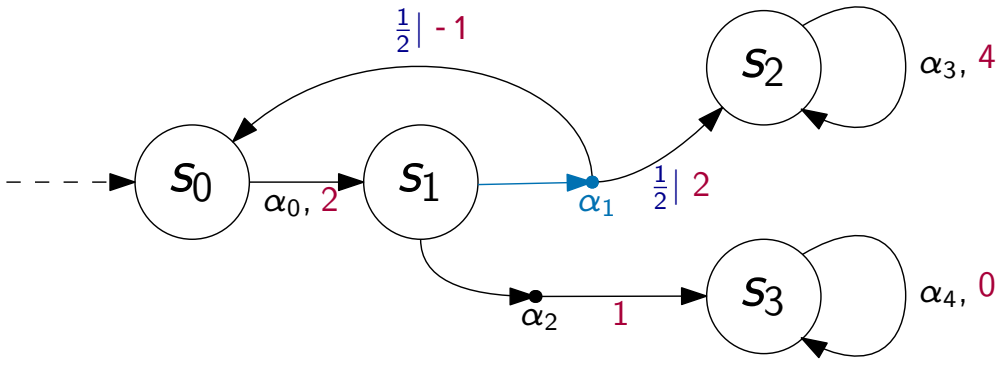
V	s_0	s_1	s_2	s_3
it #0	0	0	0	0
it #1	2	1	4	0
it #2	2.5			

- $Q_2(s_1,\alpha_1) = \frac{1}{2} \cdot (-1 + \gamma \cdot 2) + \frac{1}{2} \cdot (2 + \gamma \cdot 4) = 2$
- $Q_2(s_1,\alpha_2) = 1$

Value Iteration

Expected discounted rewards

$$Q_{t+1}(s,\alpha) = \sum_{s' \in S} \Delta(s,\alpha,s') \cdot \big(R(s,\alpha,s') + \gamma \cdot V_i(s')\big), \; V_{t+1}(s) = \max_{\alpha \in A(s)} Q_{t+1}(s,\alpha)$$



- $\gamma = \frac{1}{2}$

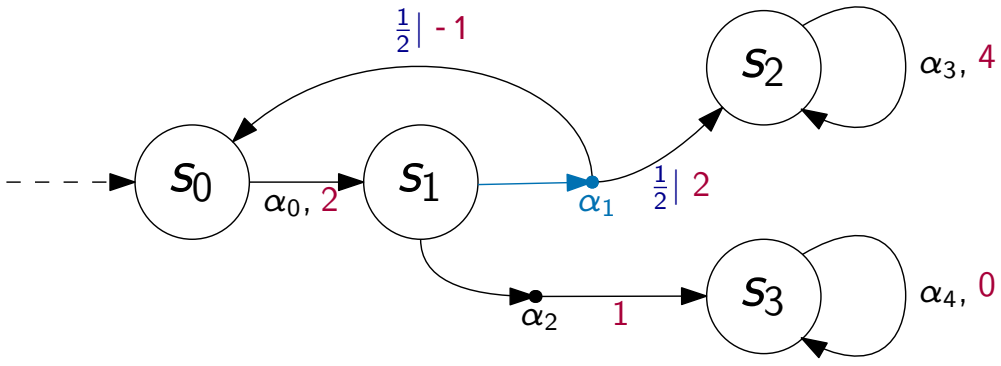
V	s_0	s_1	s_2	s_3
it #0	0	0	0	0
it #1	2	1	4	0
it #2	2.5	2		

- $Q_2(s_1,\alpha_1) = \frac{1}{2} \cdot (-1 + \gamma \cdot 2) + \frac{1}{2} \cdot (2 + \gamma \cdot 4) = 2$
- $Q_2(s_1,\alpha_2) = 1$

Value Iteration

Expected discounted rewards

$$Q_{t+1}(s,\alpha) = \sum_{s' \in S} \Delta(s,\alpha,s') \cdot \left(R(s,\alpha,s') + \gamma \cdot V_i(s')\right), \; V_{t+1}(s) = \max_{\alpha \in A(s)} Q_{t+1}(s,\alpha)$$



- $\gamma = \frac{1}{2}$

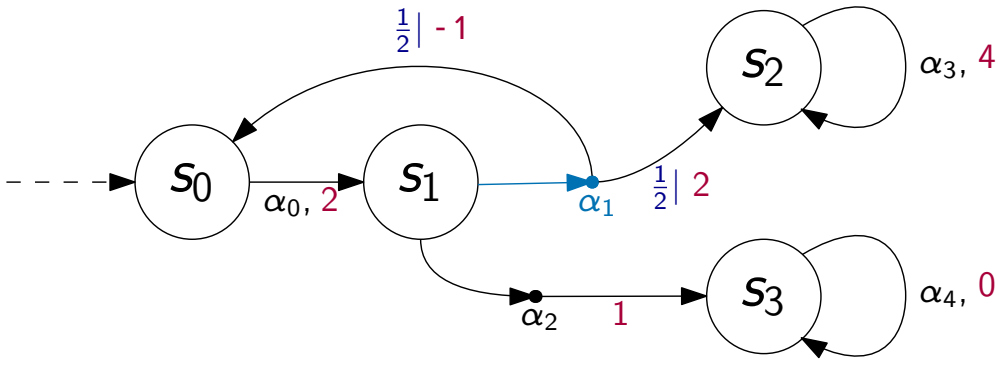
V	s_0	s_1	s_2	s_3
it #0	0	0	0	0
it #1	2	1	4	0
it #2	2.5	2	6	0

$$V_2(s_2) = 4 + \gamma \cdot 4 = 6$$

Value Iteration

Expected discounted rewards

$$Q_{t+1}(s,\alpha) = \sum_{s' \in S} \Delta(s,\alpha,s') \cdot \left(R(s,\alpha,s') + \gamma \cdot V_i(s')\right), \; V_{t+1}(s) = \max_{\alpha \in A(s)} Q_{t+1}(s,\alpha)$$

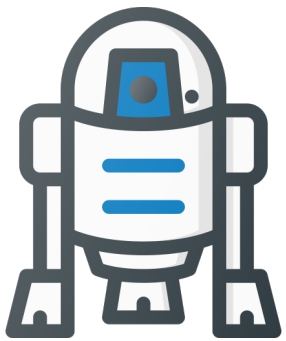


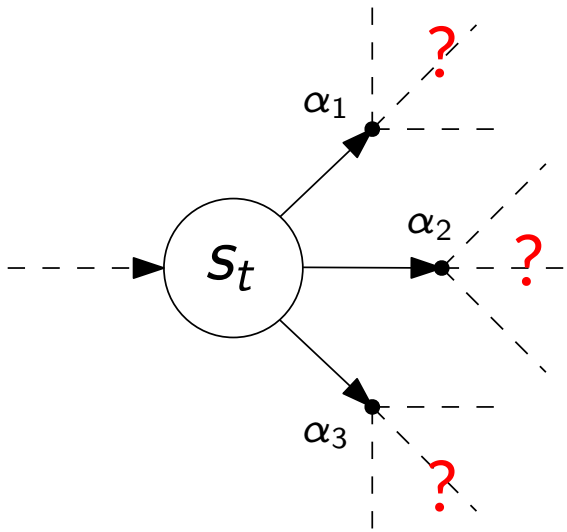
- $\gamma = \frac{1}{2}$

V	s_0	s_1	s_2	s_3
it #0	0	0	0	0
it #1	2	1	4	0
it #2	2.5	2	6	0
...		...	...	

$$V_2(s_2) = 4 + \gamma \cdot 4 = 6$$

Model-free algorithm



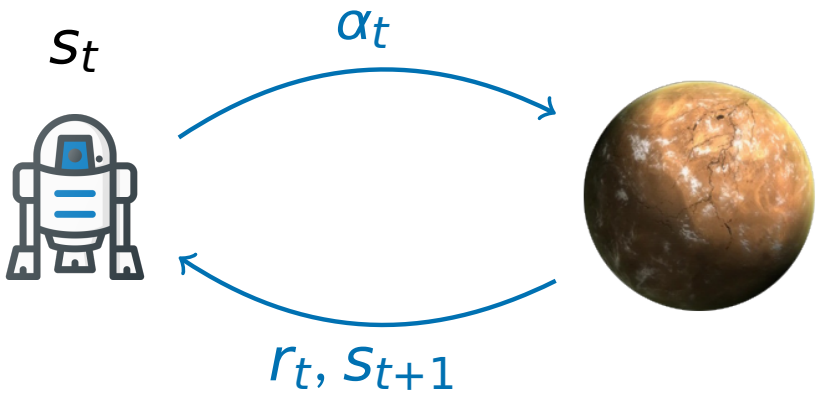


The agent doesn't know the MDP modelling the environment !

- only knows S and A , but Δ and R *(informations about transitions)* are unknown
- observes the **current state** s_t of the system and then knows **enabled actions** $A(s_t)$ of this state
- perceive rewards r_t when an action α_t is chosen

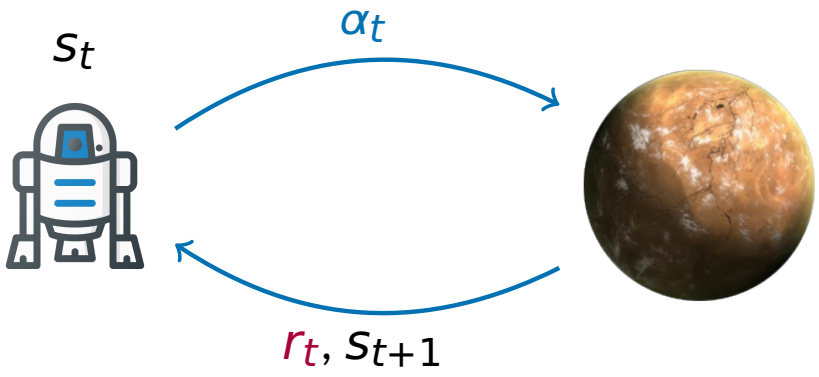
⇝ We need a **model-free** algorithm to synthesise a strategy

Reinforcement Learning



- Strategy synthesis by **learning** from *data*
- ⇝ resulting from the **exploration of the state space** of the environment by the agent

Reinforcement Learning



Algorithm 1 Reinforcement Learning

1: $\sigma \leftarrow$ **initialise** a strategy
2: **while** stop criterion not reached **do**
3: **sample** a finite path $\pi = s_0 \xrightarrow{\alpha_0} r_0\ s_1 \xrightarrow{\alpha_1} r_1\ s_2 \xrightarrow{\alpha_2} r_2\ s_3 \cdots \xrightarrow{\alpha_{n-1}} r_{n-1}\ s_n$
(exploration of the environment *)*
4: **for** $t \in \{0, \ldots, n-1\}$ **do**
5: observation $\leftarrow (s_t, \alpha_t, r_t, s_{t+1})$
6: $\sigma \leftarrow$**update**$(\sigma,$ observation$)$ *(* σ learns with the observations from π *)*
7: **return** σ

Q-learning

$$\textbf{update}(\sigma, [s_t, \alpha_t, r_t, s_{t+1}]) := ?$$

Bellman equation: expected discounted rewards

Let $s \in S$, $\alpha \in A(s)$,

- $\mathbb{E}_s^{\max}(Disc_\gamma) = V^*(s) = \max_{\alpha \in A(s)} Q^*(s, \alpha)$
- $Q^*(s, \alpha) = \sum_{s' \in S} \Delta(s, \alpha, s') \cdot \left(R(s, \alpha, s') + \gamma \cdot \max_{\alpha' \in A(s)} Q^*(s', \alpha')\right)$

$\rightarrow$ **Q-learning** $\rightsquigarrow$ learn Q^* values from observations

- **Bellman**: $Q^*(s,a) = \sum_{s' \in S} \Delta(s,a,s') \cdot \left(R(s,a,s') + \gamma \cdot \max_{a' \in A(s)} Q^*(s',a')\right)$

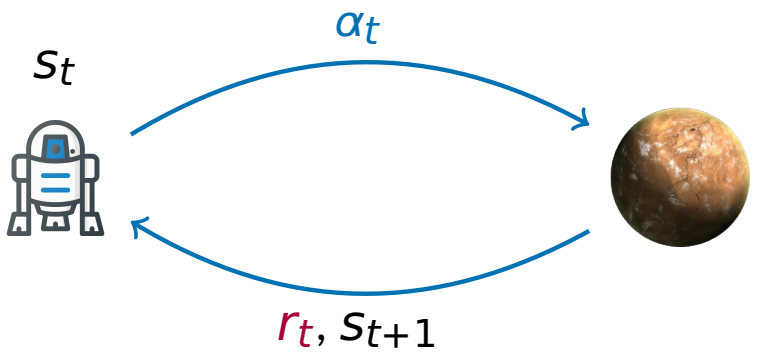
Theorem

- $\mathcal{M} = (S, A, \Delta, R)$: *(unknown) MDP modelling the environment*
- $[s, a, r, s']$: s' ***sampled*** *from the distribution* $\Delta(s,a,\cdot)$, r ***sampled*** *with mean* $R(s,a,\cdot)$,
- Q_t *are Q-values at step* $t \in \mathbb{N}$

$$Q_{t+1}(s,a) \leftarrow Q_t(s,a) + \lambda\left(r + \gamma \cdot [\max_{a' \in A(s')} Q_t(s',a')] - Q_t(s,a)\right)$$

- *Each action is executed in each state an infinite number of times*
- *The learning rate* $\lambda \in]0,1[$ *is decayed appropriately with* t

$\Longrightarrow$ Q ***converges with probability*** 1 ***to*** Q^*



Algorithm 2 Q-learning

$Q \leftarrow$ **initialise** a matrix of size $|S| \times |A|$ with values 0
while stop criterion not reached **do**
  sample a finite path $\pi = s_0 \xrightarrow{\alpha_0} r_0\ s_1 \xrightarrow{\alpha_1} r_1\ s_2 \xrightarrow{\alpha_2} r_2\ s_3 \cdots \xrightarrow{\alpha_{n-1}} r_{n-1}\ s_n$
  for $t \in \{0, \ldots, n-1\}$ **do**
    $Q(s_t, \alpha_t) \leftarrow Q(s_t, \alpha_t) + \lambda\left(r_t + \gamma \cdot [\max_{\alpha' \in A(s_{t+1})} Q(s_{t+1}, \alpha')] - Q(s_t, \alpha_t)\right)$
    $\sigma \leftarrow$ **update**(σ, Q)
  $\lambda \leftarrow \lambda \cdot (1 - x)$ (* $x = 1$ *or* $x \in]0, 1[$ *after a certain number of iterations* *)
return σ

Path sample/exploration algorithm

Sample a finite path $\pi = s_0 \xrightarrow{\alpha_0} r_0\ s_1 \xrightarrow{\alpha_1} r_1\ s_2 \xrightarrow{\alpha_2} r_2\ s_3 \cdots \xrightarrow{\alpha_{n-1}} r_{n-1}\ s_n$

- Q-learning converges to Q^* if the agent explores the MDP thoroughly enough
- Explore the MDP randomly

  ⇝ eventually visit every state and every transition many times

  $\Longrightarrow$ slow

ϵ-greedy exploration

Explore the MDP while gradually favoring promising region of this MDP.
Let $\epsilon \in [0, 1]$

- Explore by choosing best action $\alpha = \arg\max_{\alpha \in A(s)} Q(s, \alpha)$ at each step $t \in [0, \ldots, n-1]$ with probability $(1 - \epsilon)$ and random action with probability ϵ.
- Each time an entire path is sampled, decrease the value of ϵ

Strategy update's rule

Theorem

There exists an optimal memoryless and deterministic strategy $\sigma^* : S \to A$ *such that*

$$\mathbb{E}_s^{\max}(Disc_\gamma) = \mathbb{E}_s^{\sigma^*}(Disc_\gamma)$$

- $\sigma^*(s) = \arg\max_{\alpha \in A(s)} Q^*(s, \alpha) \implies$ requires to know $Q^*(s, \alpha)$
- Let $A^*(s) \subseteq A(s)$ such that for each $\alpha^* \in A^*(s)$, $Q(s, \alpha^*)$ is high
- **Randomised strategy** $\sigma : S \to \mathcal{D}(A)$ that defines a distribution over actions (*softmax*) with the highest $Q(s, \cdot)$-values:

$$\sigma(s)(\alpha) = \frac{e^{Q(s,\alpha)}}{\sum_{\alpha^* \in A^*(s)} e^{Q(s,\alpha^*)}}$$

for each $\alpha \in A^*(s)$, $\sigma(s)(\alpha) = 0$ else.

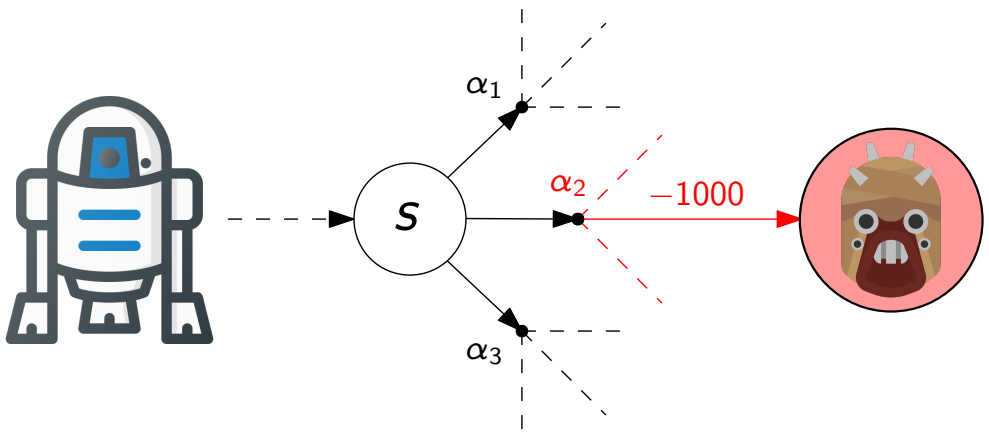
Safety

- We want the strategy σ learnt to be **safe** w.r.t. some properties of the form

$$\mathbb{P}^{\sigma}_{\mathcal{M}}(\Diamond S_{bad}) \leq \lambda$$

  for some $S_{bad} \subseteq S$ and $\lambda \in [0, 1]$.

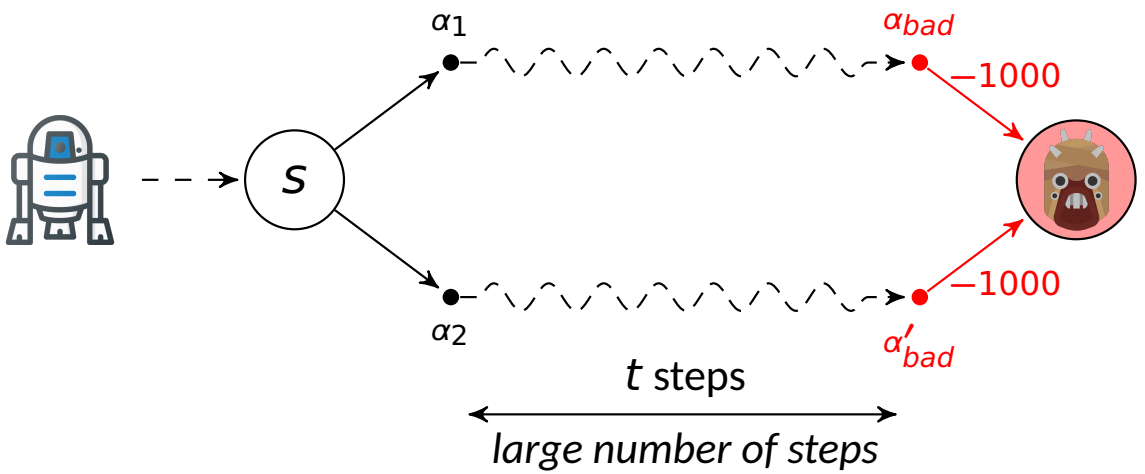
- **First attempt**: assign a **punishment** to actions that may directly lead to a bad state



S_{bad} observed after executing $\alpha_2 \implies$ set $R(s, \alpha_2, \cdot) < 0$

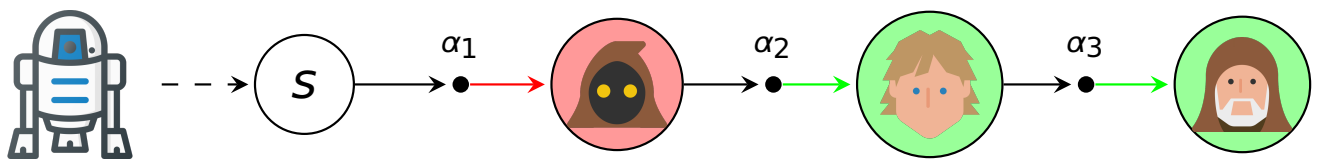
Safety

- **First attempt**: assign a **punishment** to actions that may directly lead to a bad state



Punishent hidden by γ

Future reward
$r = \gamma^t \cdot (-1000)$



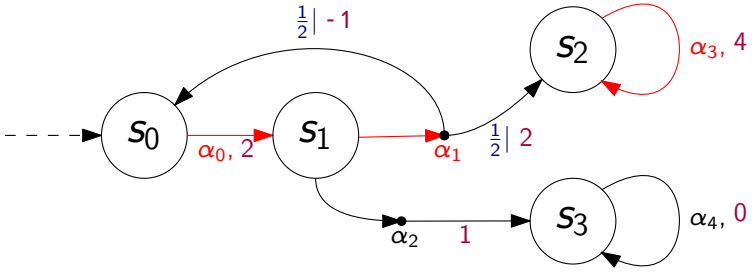
Punishment attenuated by good rewards

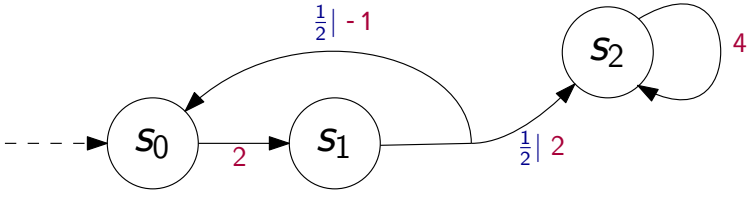
- no guarantees of safety
- no probability upper bound of reaching bad states

Markov chain

- Once the strategy is fixed in the MDP, it has a **purely stochastic behaviour**

⇝ A ***discrete-time Markov chain*** (DTMC) is induced





Theorem (Model Checking: reachability in Markov chains)

Let $\mathcal{D} = (S, \Delta)$ be a DTMC, $B \subseteq S$ and $\lambda \in [0, 1]$, $\mathbb{P}_{\mathcal{D}}(\Diamond B) \leq \lambda$ can be decided via

- *a **system of linear equations**: polynomial time in $\mathcal{D}$*
- *the **value iteration** algorithm: exponential time in $\mathcal{D}$ but very fast in practice*

Moreover, $\mathbb{P}_{\mathcal{D}}(\Diamond B) = 0$ can be decided in $\mathcal{O}(E)$ with purely graph theory algorithms, with $E = \#\text{transitions in } \mathcal{D}$.

Model generation

- The strategy σ learnt by Q-learning is *randomised*
- $\sigma(s)$ is a **probability distributon over actions offering best Q-values learnt**

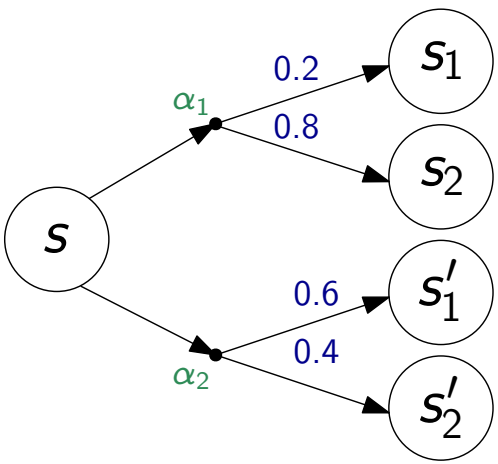
$$\sigma(s)(\alpha) = \begin{cases} \frac{e^{Q(s,\alpha)}}{\sum_{\alpha^* \in A^*(s)} e^{Q(s,\alpha^*)}} & \text{if } \alpha \in A^*(s) \\ 0 & \text{else} \end{cases}$$

Model generation

- The strategy σ learnt by Q-learning is *randomised*
- $\sigma(s)$ is a **probability distributon over actions offering best Q-values learnt**
- Remember **statistical informations of observations during learning** in $\mathcal{O}$
  - when $[s, \alpha, r, s']$ is observed, $\mathcal{O}(s, \alpha, s') \leftarrow \mathcal{O}(s, \alpha, s') + 1$
  - $\mathcal{O}(s, \alpha, s') =$ # times the transition $s \xrightarrow{\alpha} s'$ has been observed
  - $\mathcal{O}(s, \alpha) = \sum_{s' \in S} \mathcal{O}(s, \alpha, s') =$ # times the agent has chosen α in state s

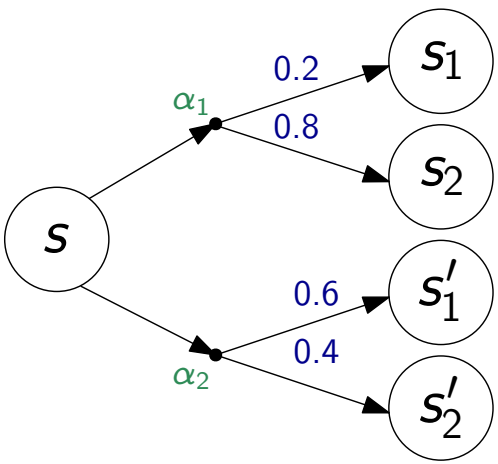
Model generation

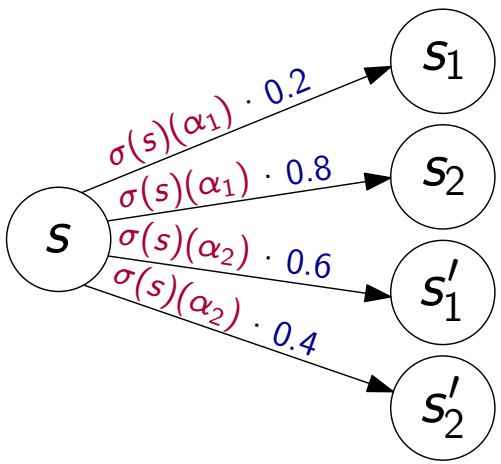
- The strategy σ learnt by Q-learning is *randomised*
- $\sigma(s)$ is a **probability distributon over actions offering best Q-values learnt**
- Remember **statistical informations of observations during learning** in $\mathcal{O}$
  - when $[s, \alpha, r, s']$ is observed, $\mathcal{O}(s, \alpha, s') \leftarrow \mathcal{O}(s, \alpha, s') + 1$
  - $\mathcal{O}(s, \alpha, s')$ = # times the transition $s \xrightarrow{\alpha} s'$ has been observed
  - $\mathcal{O}(s, \alpha) = \sum_{s' \in S} \mathcal{O}(s, \alpha, s')$ = # times the agent has chosen α in state s
- **Probability transition function of the MDP modelling the environment**: $\Delta^{\mathcal{O}}(s, \alpha, s') = \frac{\mathcal{O}(s,\alpha,s')}{\mathcal{O}(s,\alpha)}$



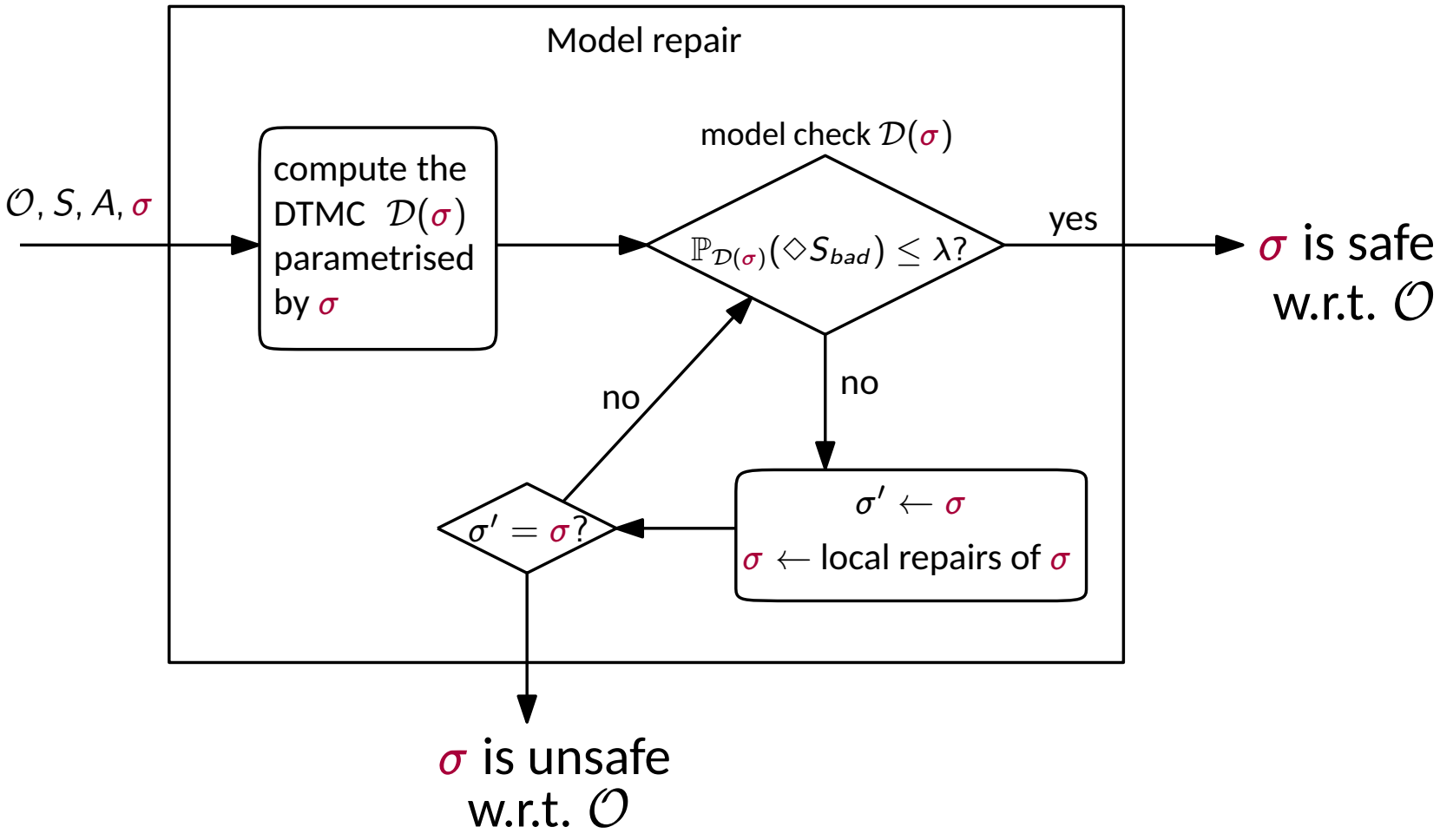
Model generation

- The strategy σ learnt by Q-learning is *randomised*
- $\sigma(s)$ is a **probability distributon over actions offering best Q-values learnt**
- Remember **statistical informations of observations during learning** in $\mathcal{O}$
  - when $[s, \alpha, r, s']$ is observed, $\mathcal{O}(s, \alpha, s') \leftarrow \mathcal{O}(s, \alpha, s') + 1$
  - $\mathcal{O}(s, \alpha, s')$ = # times the transition $s \xrightarrow{\alpha} s'$ has been observed
  - $\mathcal{O}(s, \alpha) = \sum_{s' \in S} \mathcal{O}(s, \alpha, s')$ = # times the agent has chosen α in state s
- **Probability transition function of the MDP modelling the environment**: $\Delta^{\mathcal{O}}(s, \alpha, s') = \frac{\mathcal{O}(s,\alpha,s')}{\mathcal{O}(s,\alpha)}$
- DTMC $\mathcal{D} = (S, \Delta)$ induced by $\sigma \rightsquigarrow \Delta(s, s') = \sum_{\alpha \in A(s)} \Delta^{\mathcal{O}}(s, \alpha, s') \cdot \sigma(s)(\alpha)$





Model Repair



Repairing the strategy

- From the **model checking**, we know $\mathbb{P}_s^{\mathcal{D}}(\Diamond S_{bad})$ from each state $s \in S$ in $\mathcal{D}$

Algorithm 3 Greedy strategy repairing

Input : $\delta \in]0, 1[$

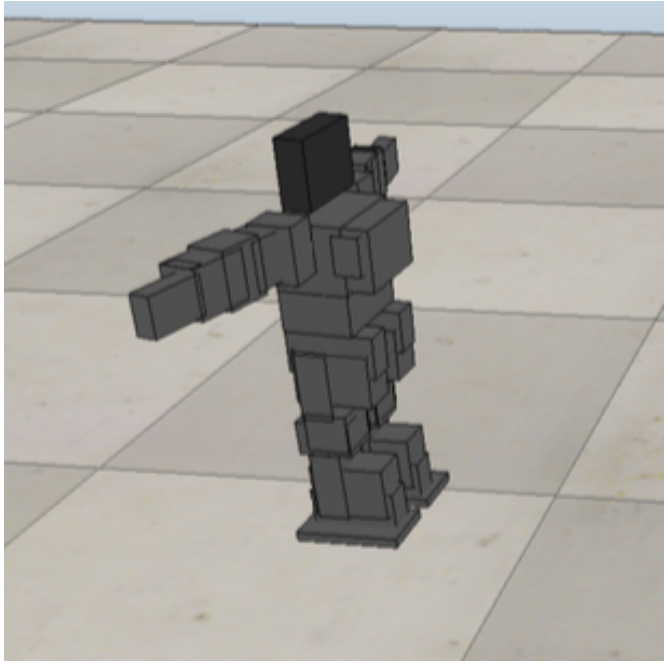
1: **for all** $s \in S$ **do**
2: $\quad \alpha_{safe} \leftarrow \arg\min_{\alpha \in A(s)} \sum_{s' \in S} \Delta^{\mathcal{O}}(s, \alpha, s') \cdot \mathbb{P}_{s'}^{\mathcal{D}}(\Diamond S_{bad})$
3: $\quad$ **for** $\alpha \in A(s) \setminus \{\alpha_{safe}\}$ with $\sigma(s)(\alpha) > \delta$ **do**
4: $\quad\quad \sigma(s)(\alpha_{safe}) \leftarrow \sigma(s)(\alpha_{safe}) + \delta$
5: $\quad\quad \sigma(s)(\alpha_{safe}) \leftarrow \sigma(s)(\alpha) - \delta$

- **Speed up the process**: begin with a large δ and decrement when no further repair is possible (e.g., $\delta \in \{0.2, 0.1, 0.05, 0.01, \dots\}$)

Standing-Up Task for Humanoids Robots

F. Leofante et al., 2016

- Bipedal locomotion
- Replace scripted strategies
  - Lack flexibility
  - Reliability and robustness issues
- **Q-learning approach**



⇝ Synthesise a standing-up strategy that **minimises** the expected **number of falls**, **self-collisions** and **actions**

Action space

- 750 actions: 3 actions for each limb (3 upper limbs, 3 lower limbs)
- $A = \{1, 0, -1\}^6$
- **Increment (+1)** or **decrement (-1)** each of the joint angles by a fixed amount Δ^{act}
- *Examples:*
  - $(1, -1, 0, 0, 0, 0) \implies$ **increment angle of limbs 1** by Δ^{act} and **decrement limbs 2** by Δ^{act}
  - $(0, 0, 0, 0, 0, 0) \implies$ do nothing

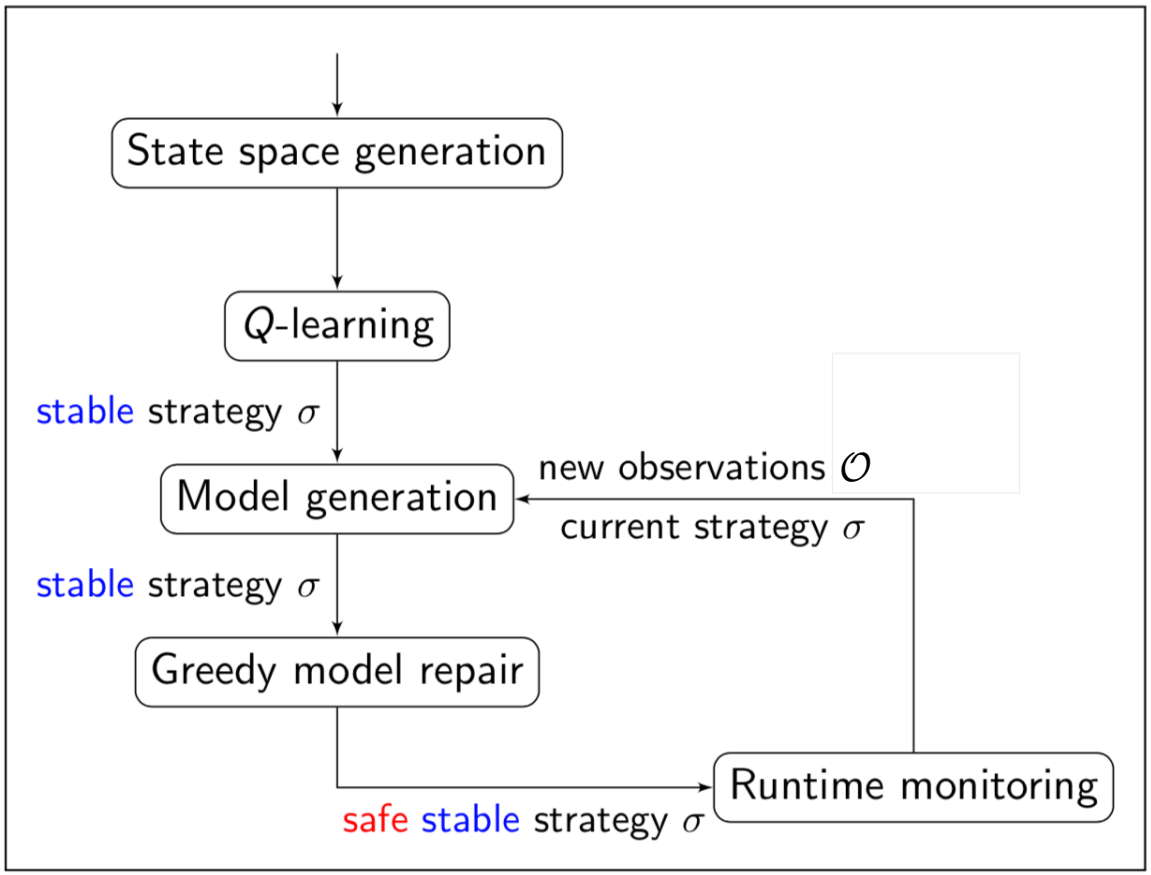
State space

- Robot state $s = (x, y, z, q_0, q_1, q_2, q_3, \rho_1, \ldots, \rho_{18}) \in \mathbb{R}^{25}$
- coordinates, orientation of the torso, joint angles
- S is infinite
- Grid-based discretisation not applicable (x^{25} states)
- To enable learning + formal verification, we need to restrict ourselves to a finite subset S of states
- **Abstraction of the state space** with informations from simulations $\rightsquigarrow |S| = 17641$
  - $s^{init}, s^{goal}, s^{far}, s^{fall}, s^{coll}$
  - s^{goal}: statically stable poses
  - s^{far}: far from poses of interests

Rewards

- **Collisions**: $R(s, \alpha, s^{coll}) = -c^{coll}$
- **Bad move**: $R(s, \alpha, s^{far}) = -c^{far}$
- **Stable pose**: $R(s, \alpha, s^{goal}) = +c^{goal}$
- **Energy consumption**: $R(s, \alpha, s') = -c^{exec}, c^{exec} <<$

- Use a simulated model of the robot (simulator VREP) during Q-learning to learn a stable strategy
- Markov Chain model checked with STORM
- Deploy the safe stable strategy on the real robot during the **runtime monitoring**



Model repair

	s^{fall}	s^{coll}	s^{far}
Reach.prob. in model *before repair*	0.001	0.005	0.048
Reach.prob. in simulation *before repair*	0	0.003	0.046
Reach.prob. in model *after repair*	0.0003	$6.8 \cdot 10^{-6}$	0.02
Reach.prob. in simulation *after repair*	0	0	0

References I

[BK08] Christel Baier and Joost-Pieter Katoen, *Principles of model checking*, The MIT Press, 2008.

[KLM96] Leslie Pack Kaelbling, Michael L. Littman, and Andrew P. Moore, *Reinforcement learning: A survey*, Journal of Artificial Intelligence Research **4** (1996), 237–285.

[LVÁ+16] Francesco Leofante, Simone Vuotto, Erika Ábrahám, Armando Tacchella, and Nils Jansen, *Combining static and runtime methods to achieve safe standing-up for humanoid robots*, ISoLA, 2016.

Further reading I

[ABE+17] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu, *Safe reinforcement learning via shielding*, CoRR **abs/1708.08611** (2017).

[JJD+15] Sebastian Junges, Nils Jansen, Christian Dehnert, Ufuk Topcu, and Joost-Pieter Katoen, *Safety-constrained reinforcement learning for mdps*, CoRR **abs/1510.05880** (2015).

[JJK+18] Sebastian Junges, Nils Jansen, Joost-Pieter Katoen, Ufuk Topcu, Ruohan Zhang, and Mary Hayhoe, *Model checking for safe navigation among humans: 15th international conference, qest 2018, beijing, china, september 4-7, 2018, proceedings*, 01 2018, pp. 207–222.