4.1. Introduction to Reinforcement Learning in Planning

Ángel Aso Mollar
Eva Onaindía de la Rivaherrera

Planificación Inteligente

Curso 2023-2024

Máster Universitario en Inteligencia Artificial,
Reconocimiento de Formas e Imagen Digital

DSIC DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Contents

1. Introduction

2. Understanding RL

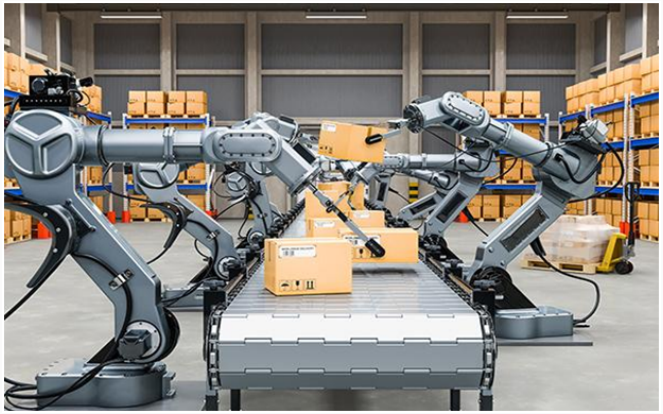
3. Markov Decision Processes (MDPs)

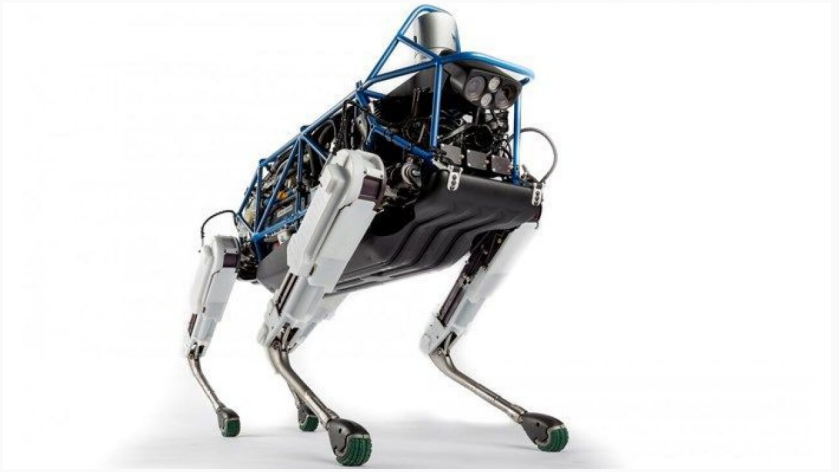
4. Optimal policies and Dynamic Programming

5. Model-free methods

Introduction

MOTIVATION: CONTROL





PLANNING

- A **classical planning problem** is a tuple $P = \langle V, A, I, G \rangle$ in which:
  - V is a set of state variables
  - A is a set of actions
  - I is the initial state
  - G is a goal condition
- A **sequential plan** $\pi = \langle a_1, ..., a_n \rangle$ solves P iff its execution starting from I reaches a state where G is satisfied.

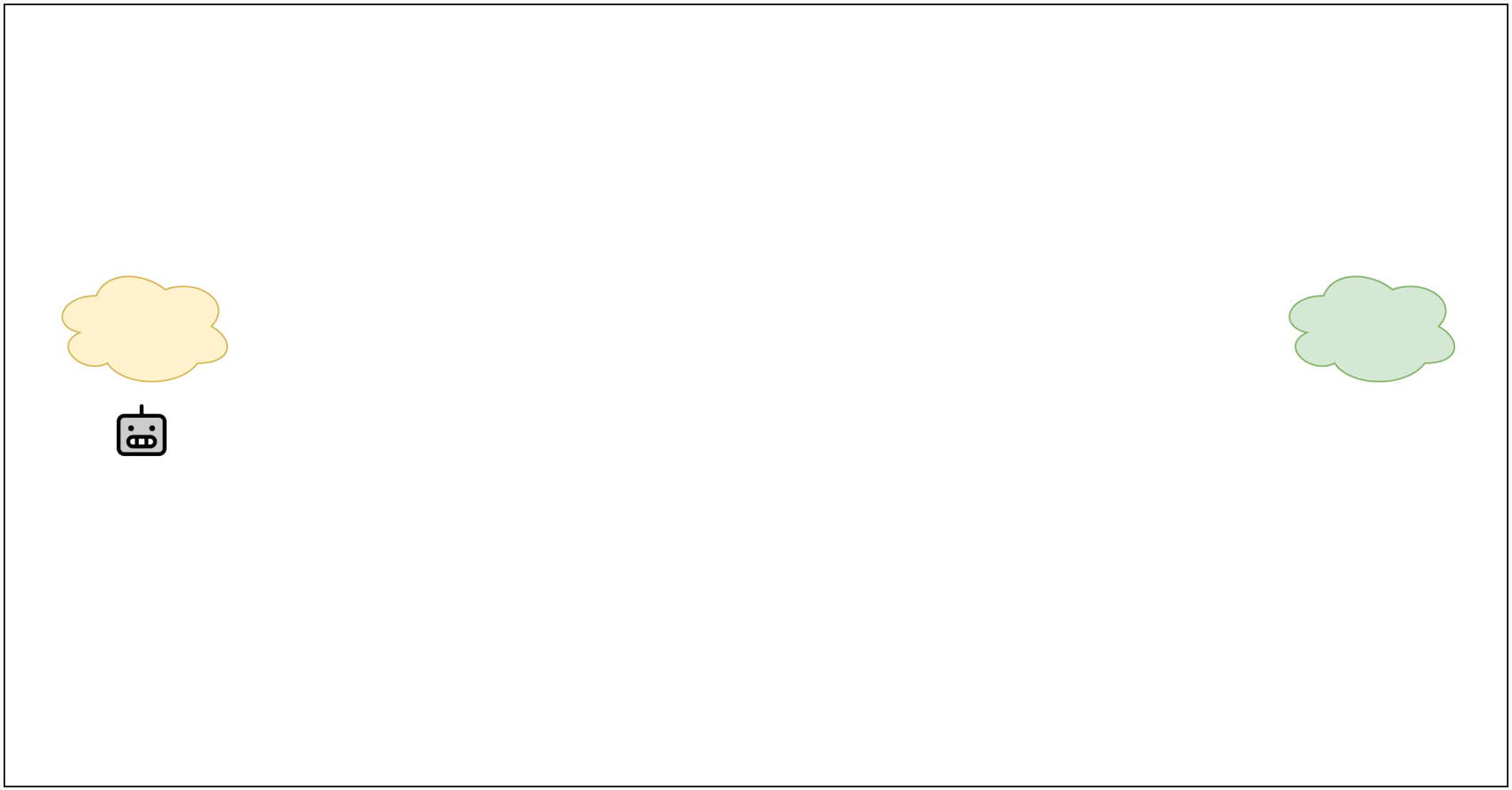
LEARNING

- Automated planning can benefit from:

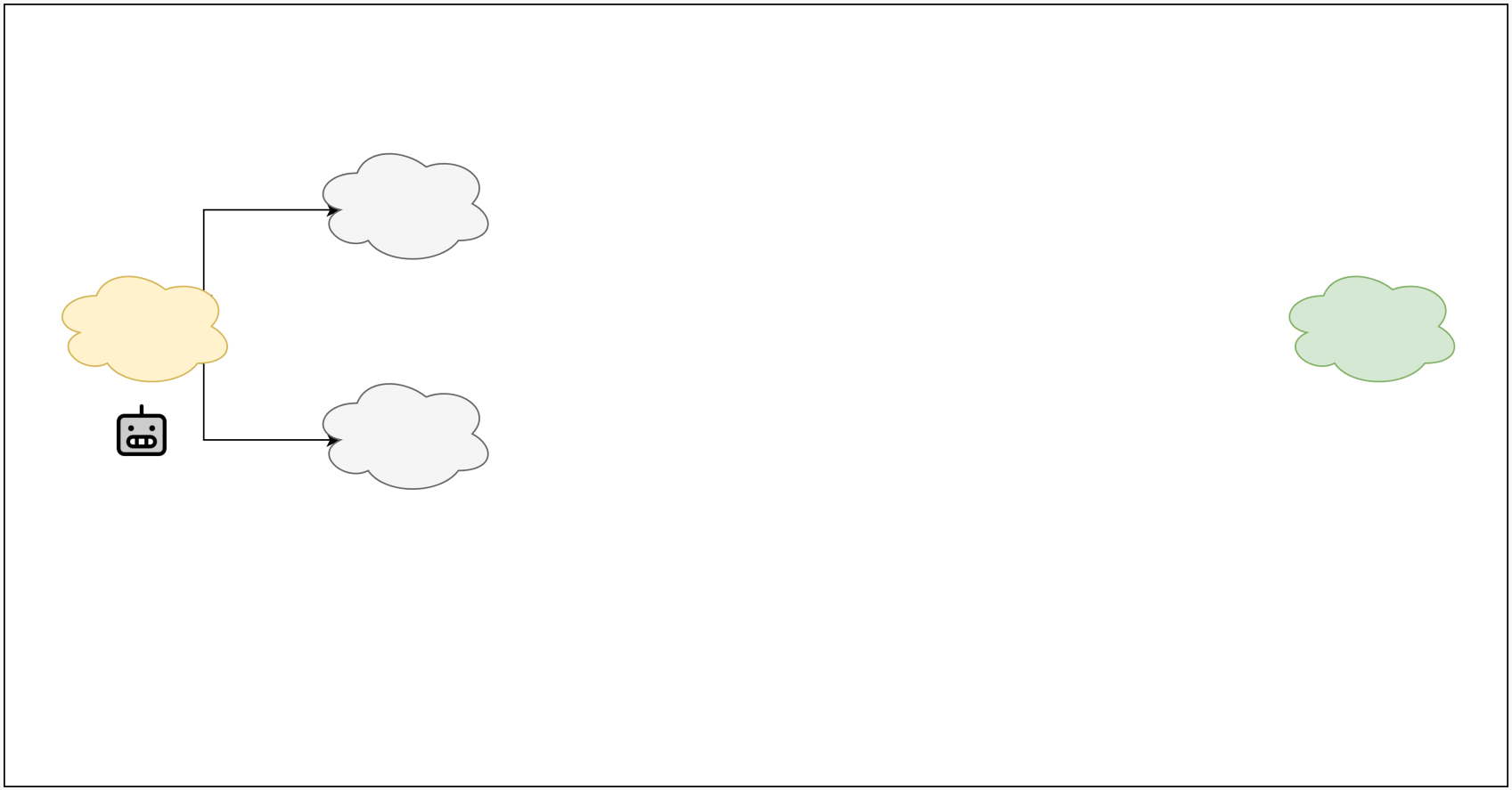
Unsupervised ML	Learning action models
Supervised ML	Learning solution strategies
	Domain-specific heuristics

- What about **Reinforcement Learning** (RL)?

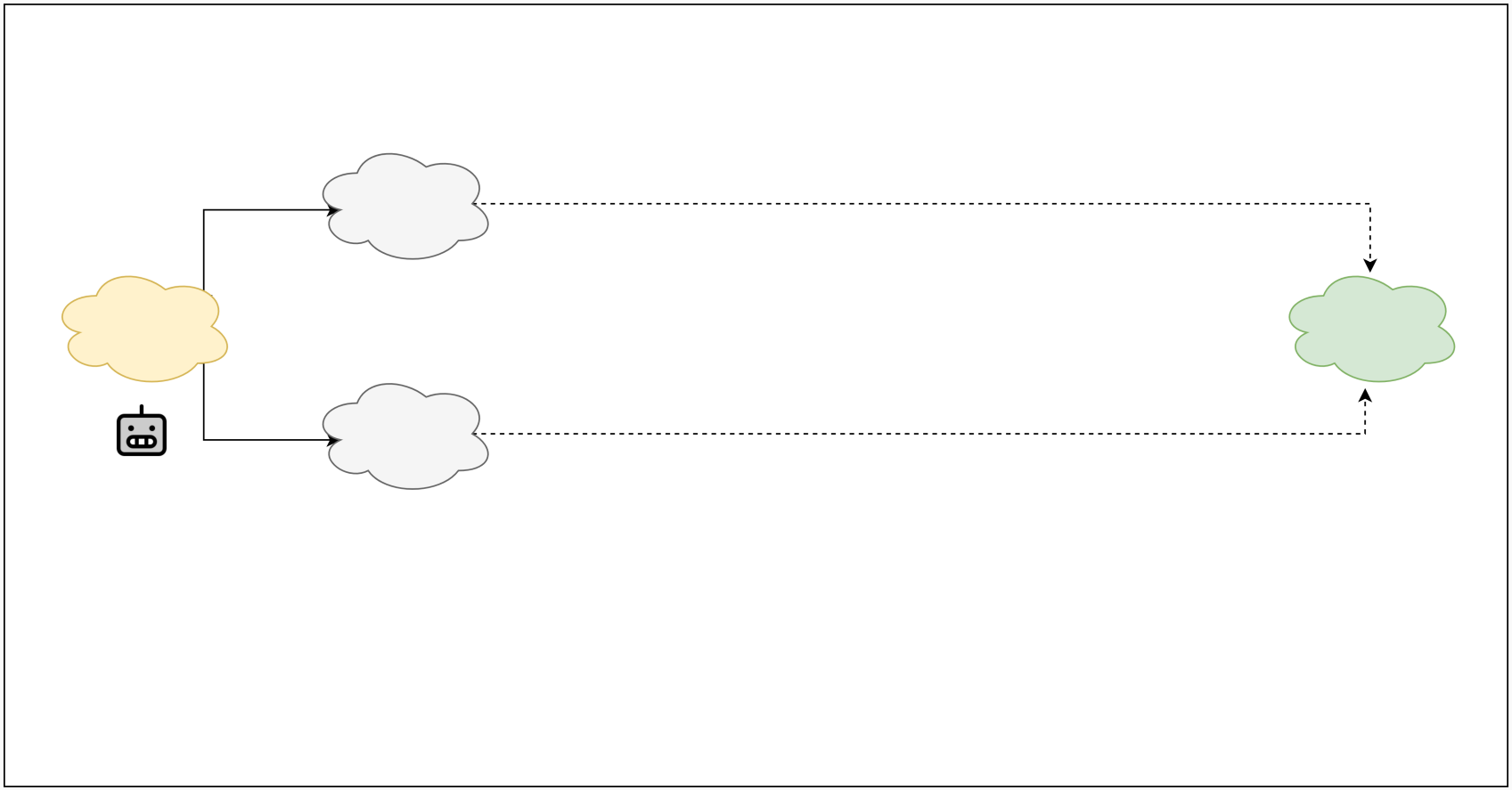
SEQUENTIAL DECISION MAKING: PLANNING



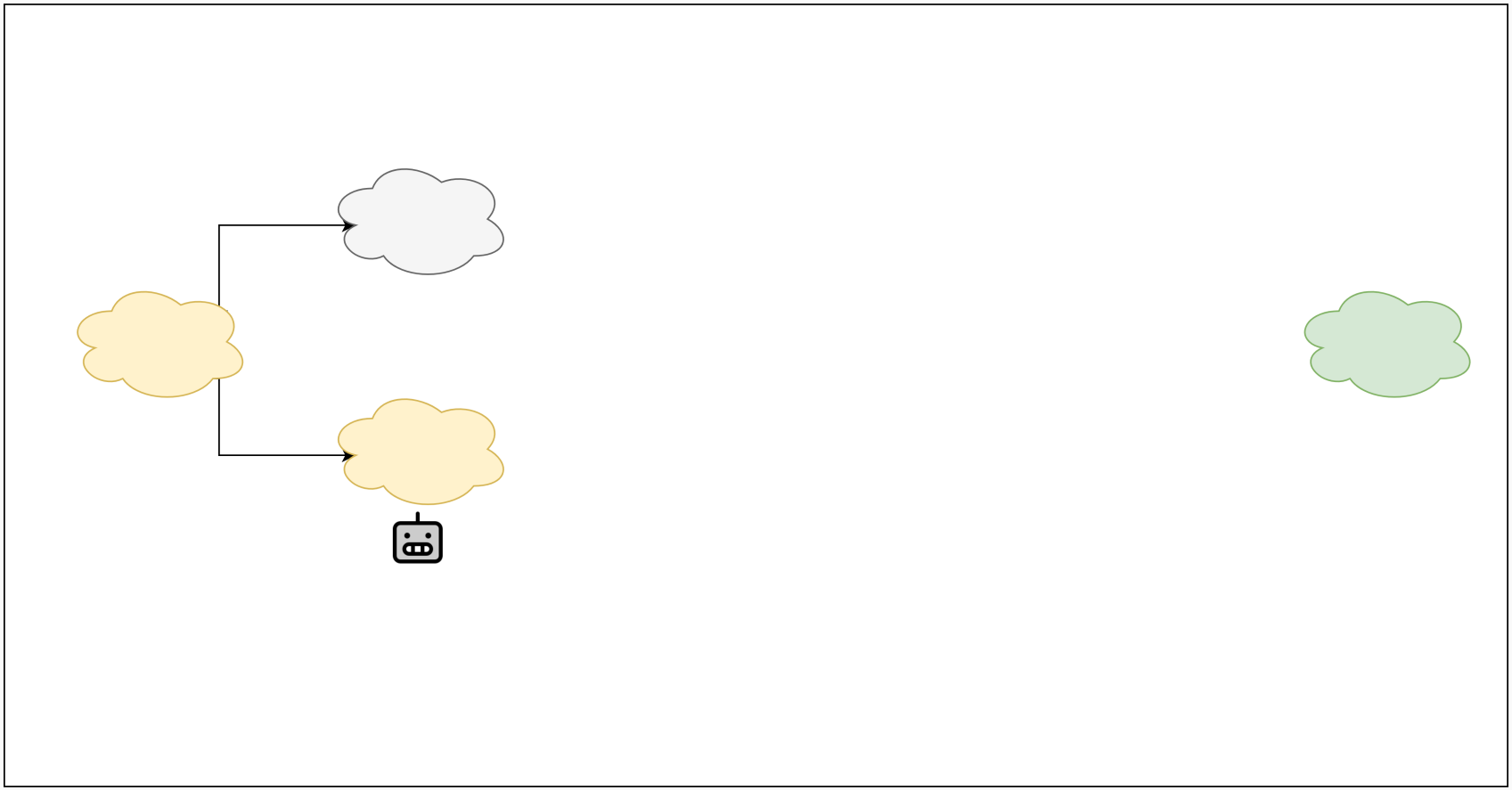
SEQUENTIAL DECISION MAKING: PLANNING



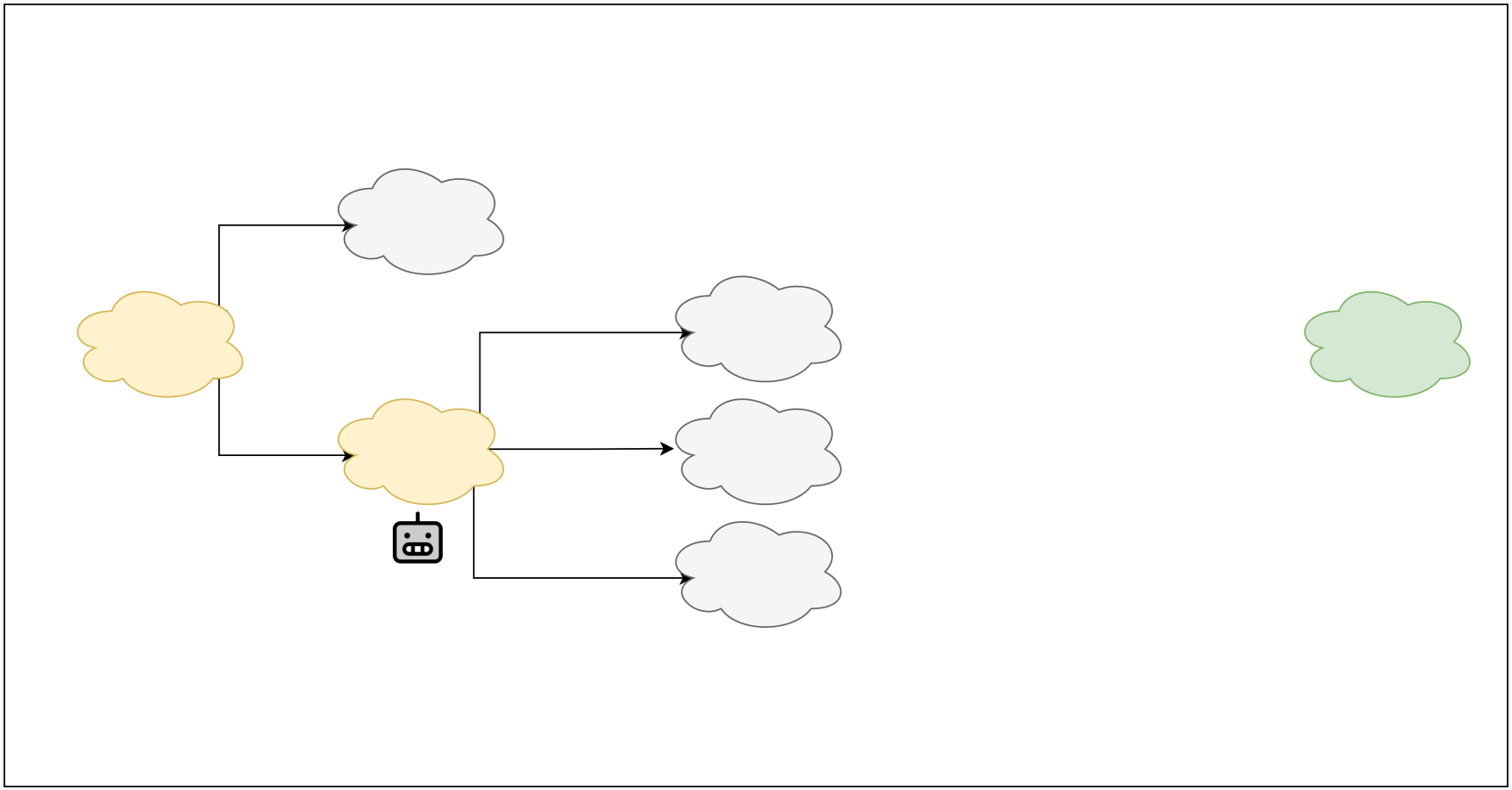
SEQUENTIAL DECISION MAKING: PLANNING



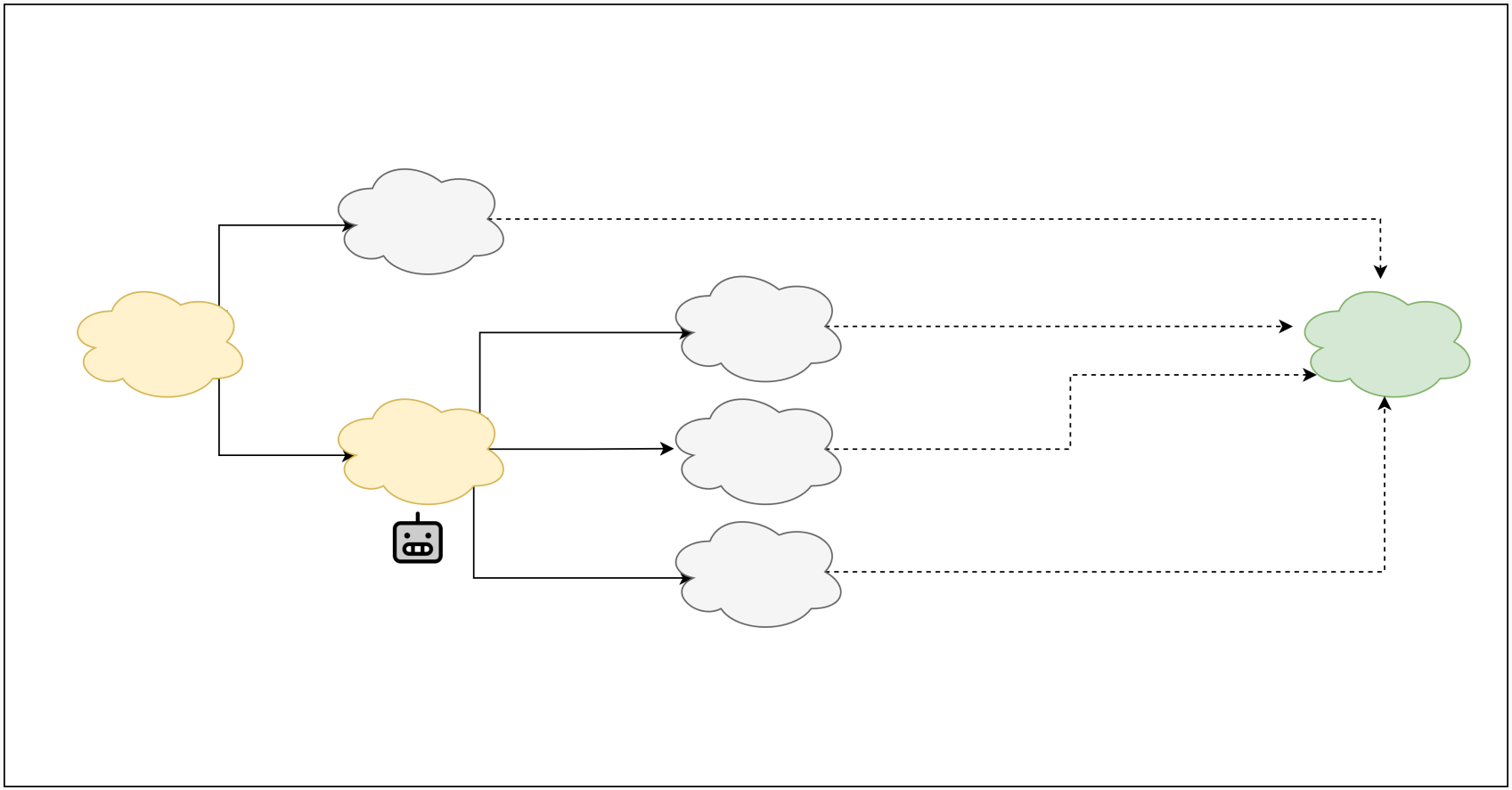
SEQUENTIAL DECISION MAKING: PLANNING



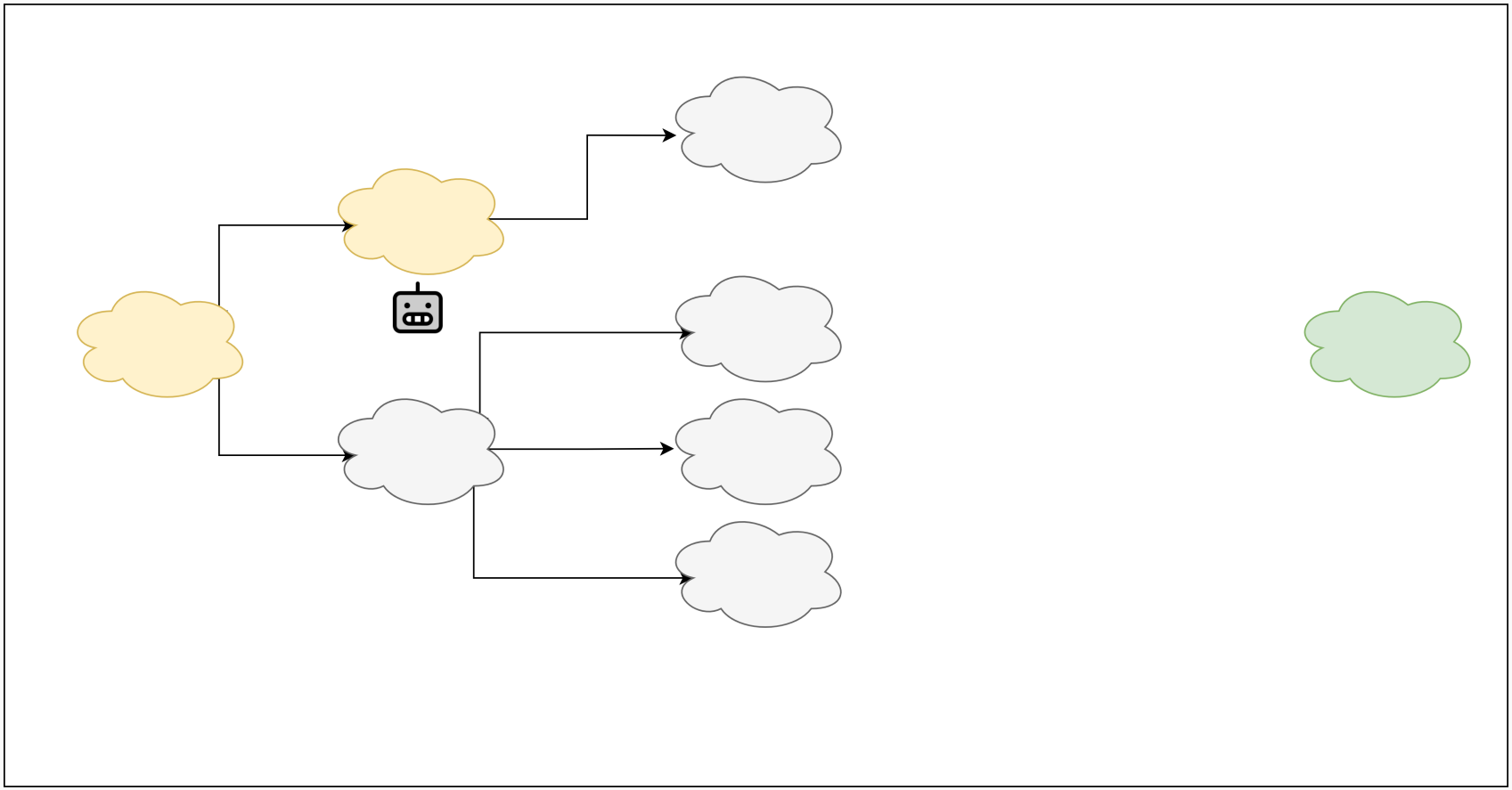
SEQUENTIAL DECISION MAKING: PLANNING



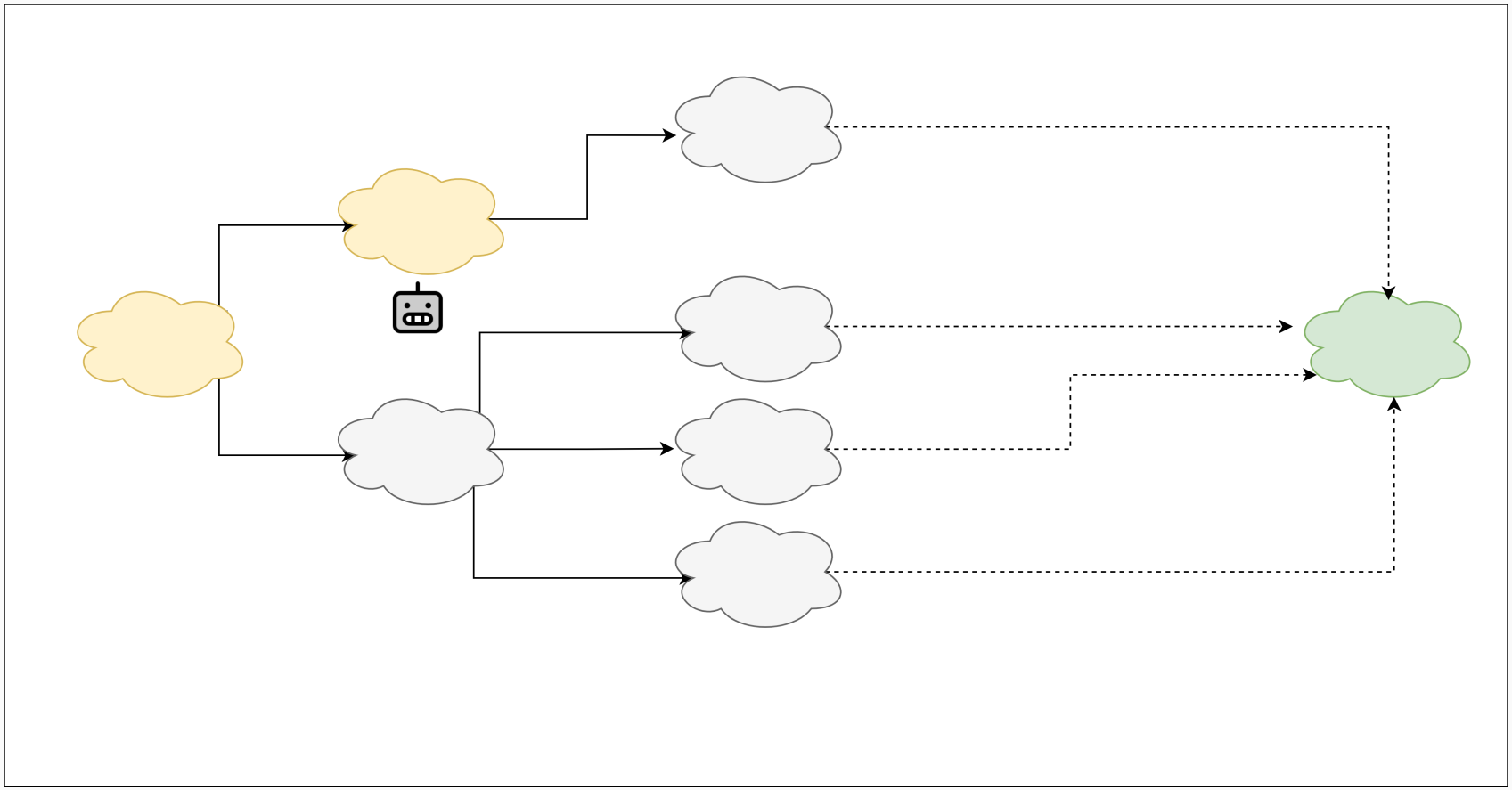
SEQUENTIAL DECISION MAKING: PLANNING



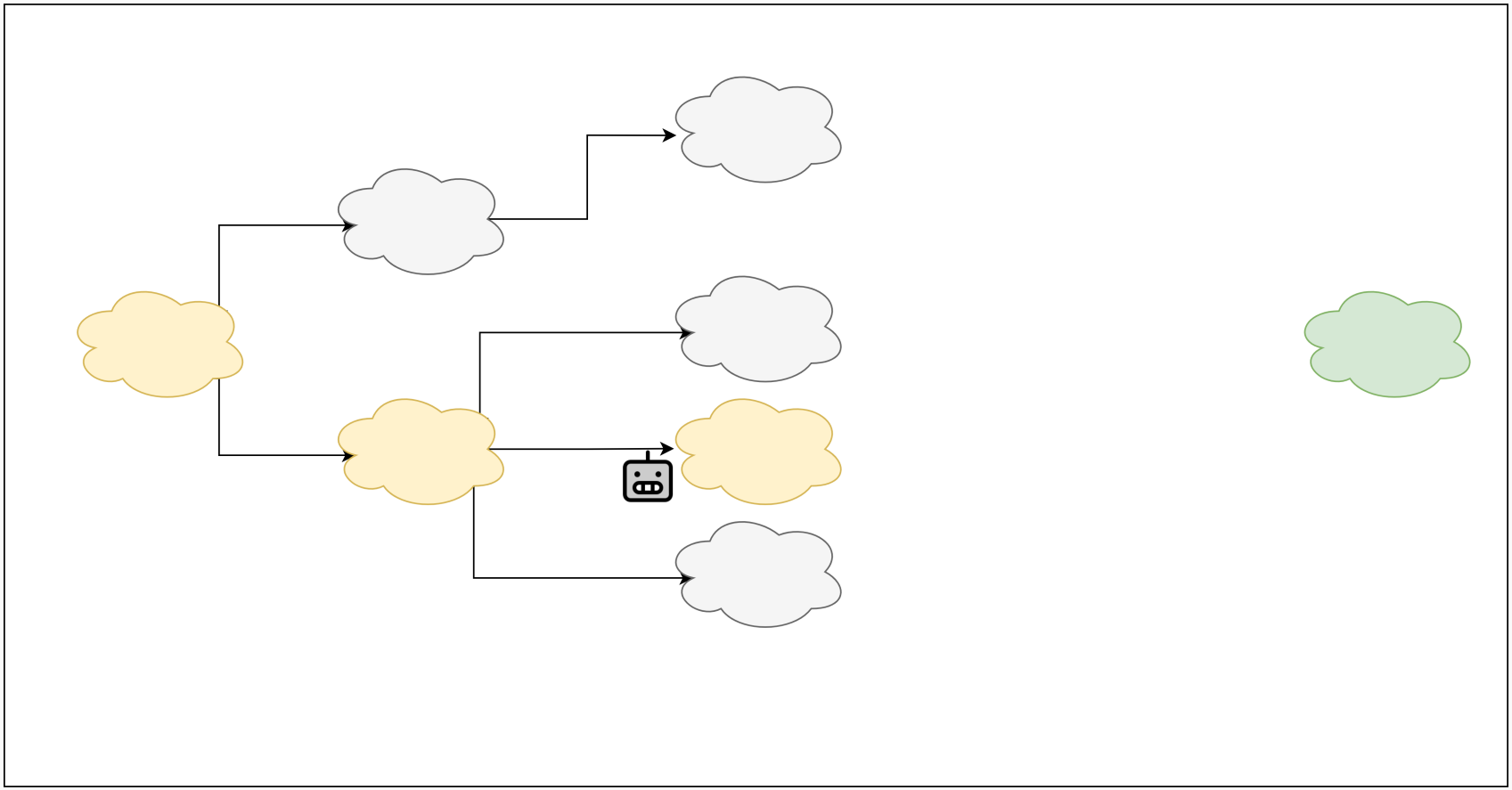
SEQUENTIAL DECISION MAKING: PLANNING



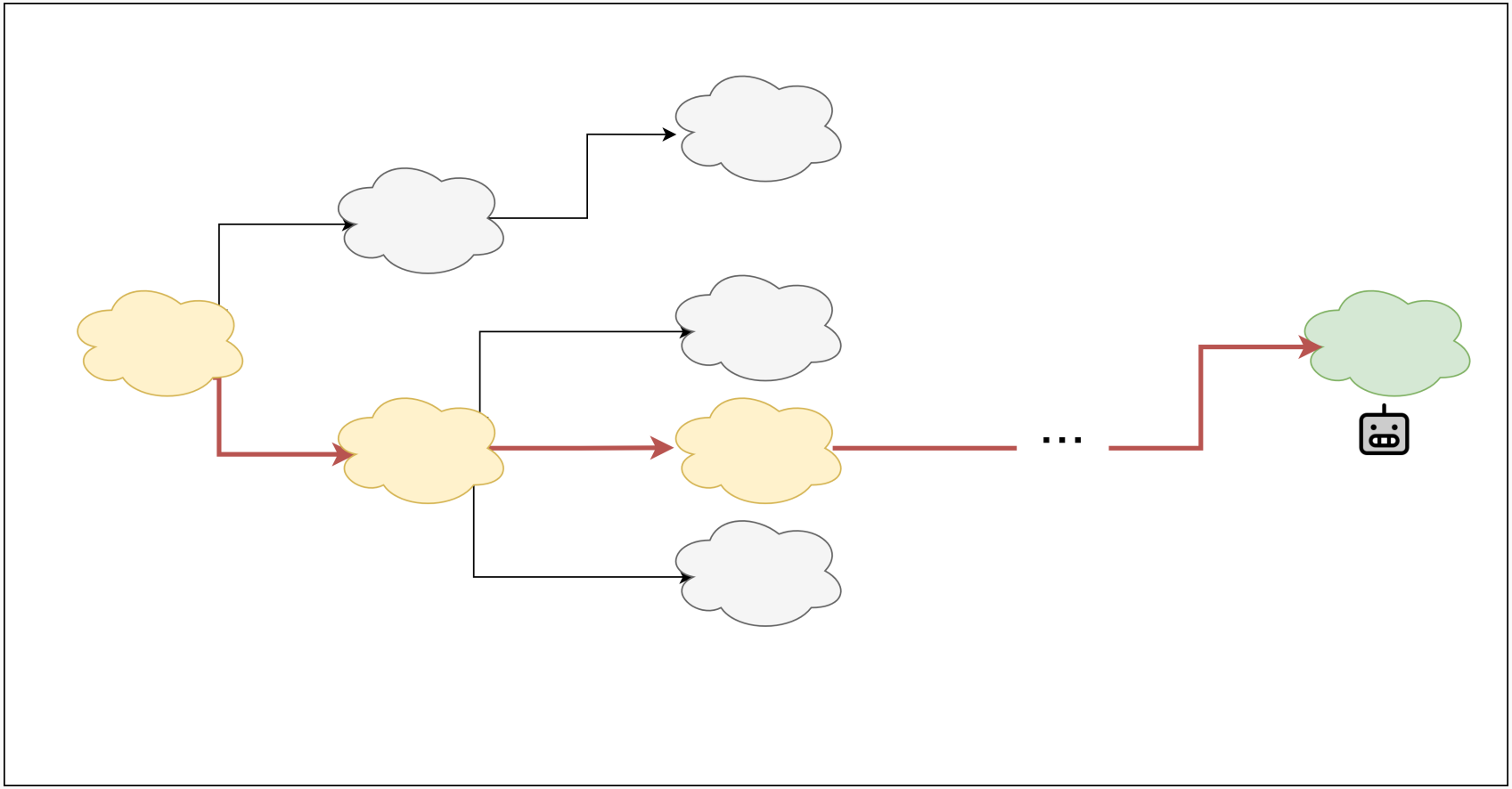
SEQUENTIAL DECISION MAKING: PLANNING



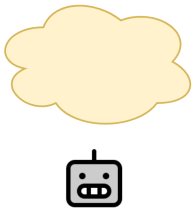
SEQUENTIAL DECISION MAKING: PLANNING



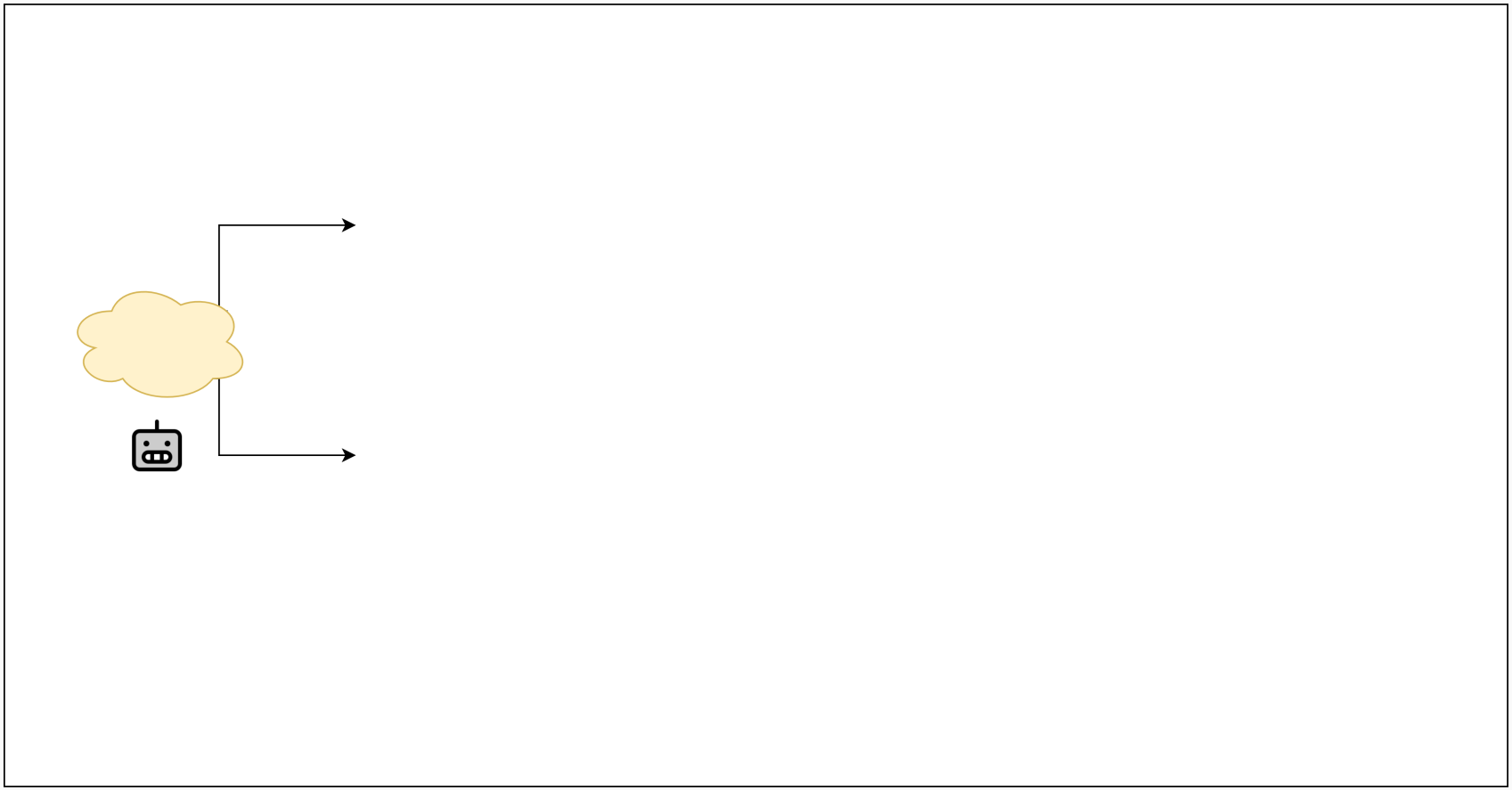
SEQUENTIAL DECISION MAKING: PLANNING



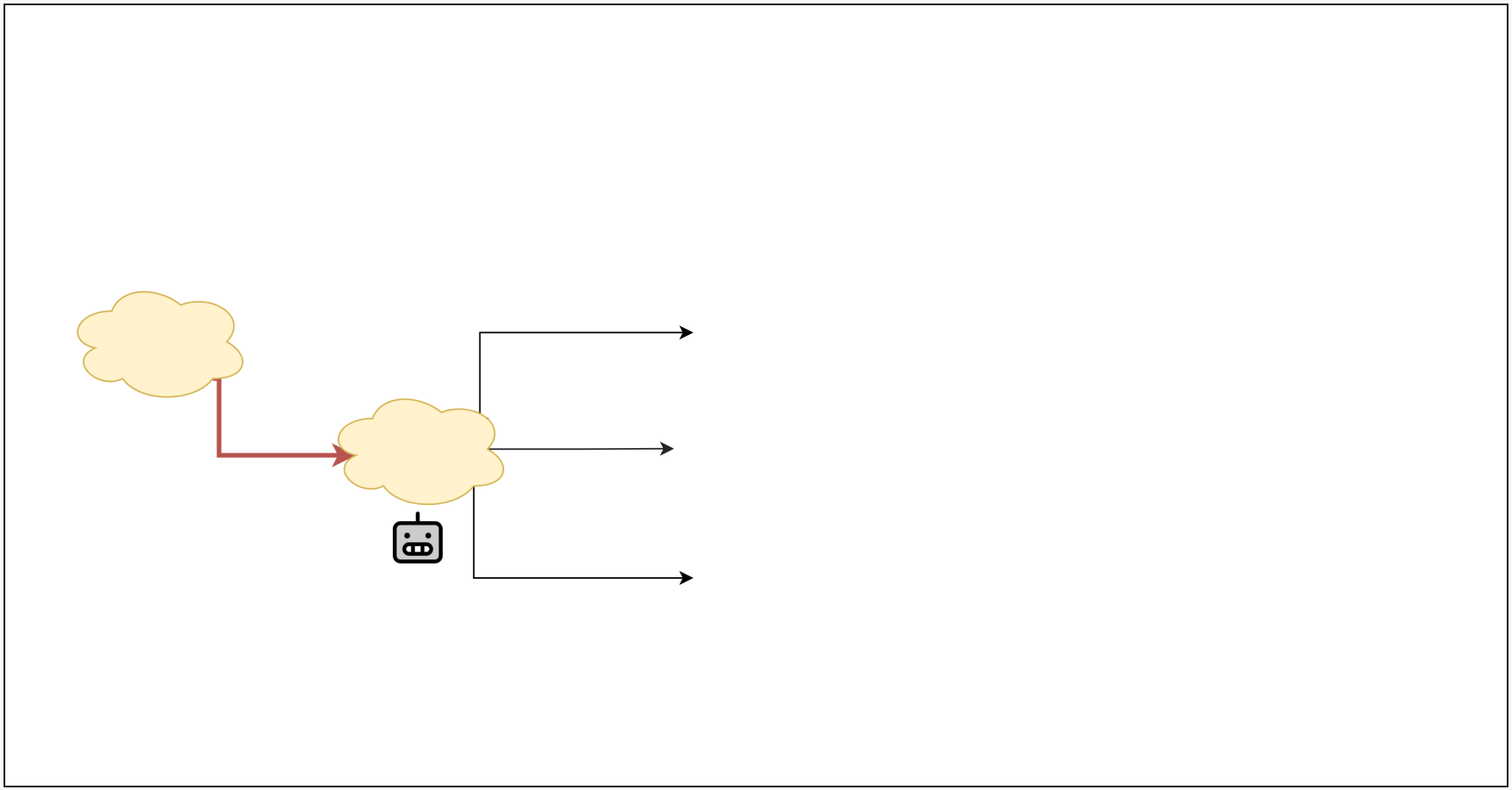
SEQUENTIAL DECISION MAKING: RL



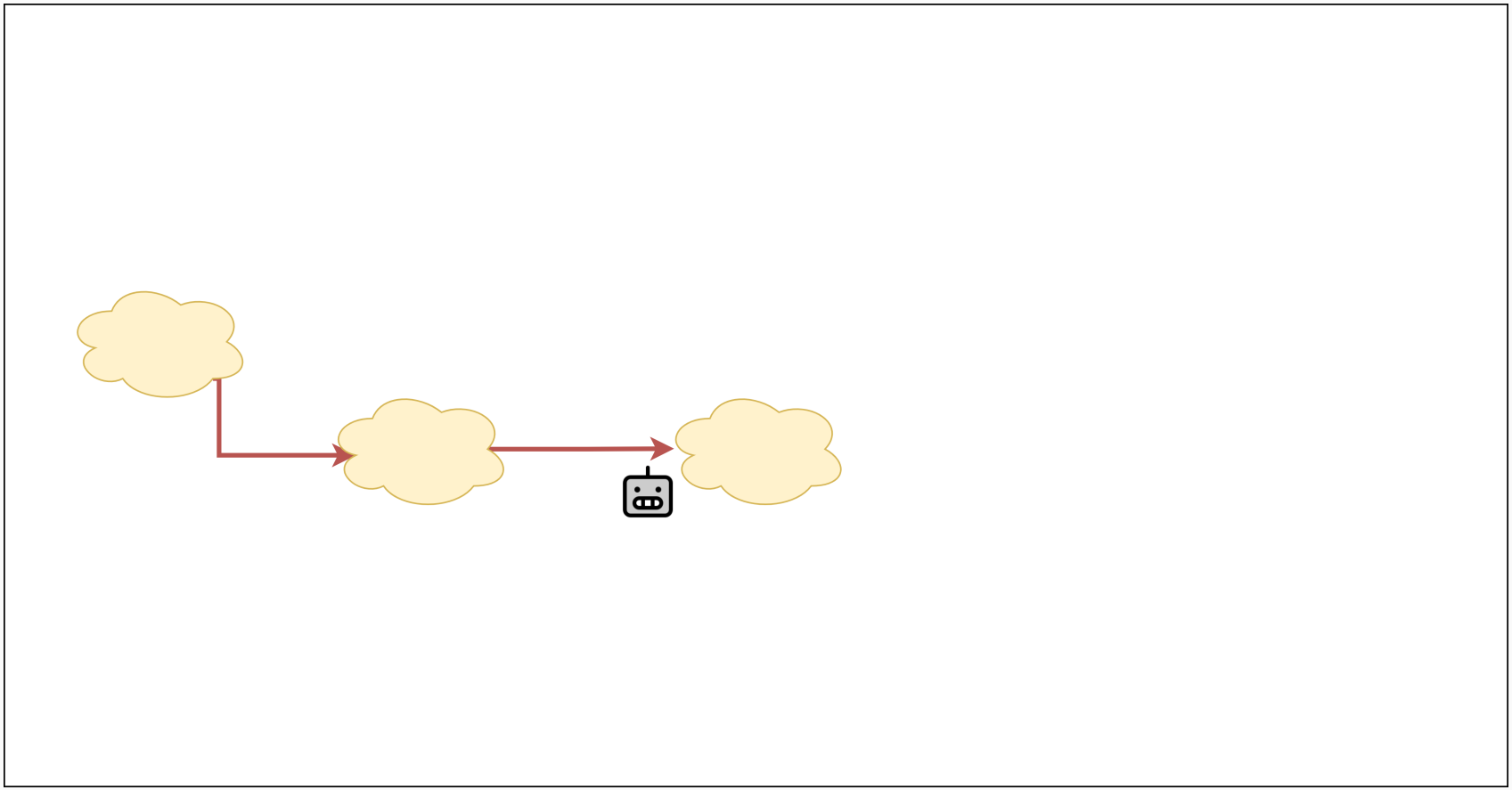
SEQUENTIAL DECISION MAKING: RL



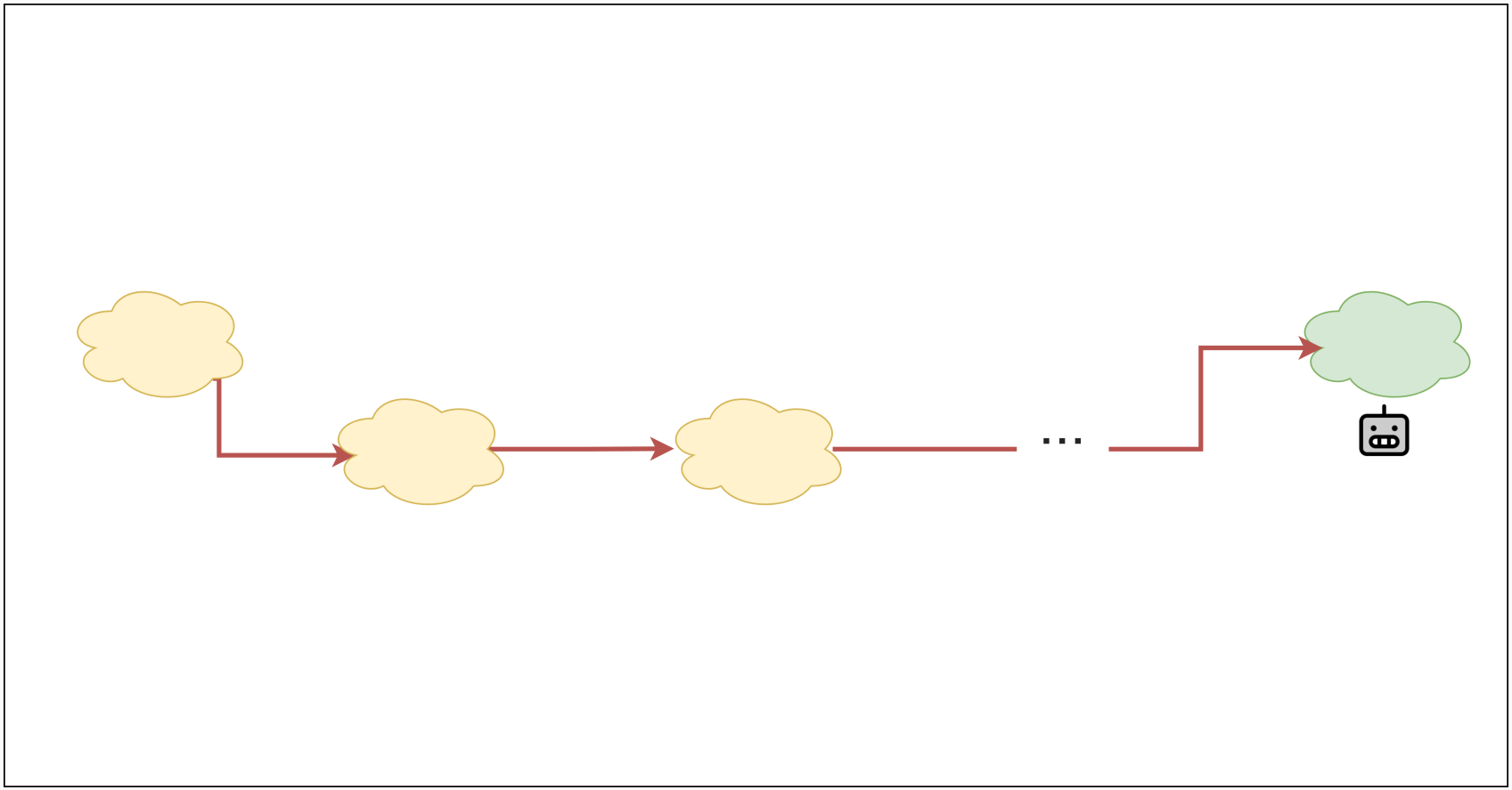
SEQUENTIAL DECISION MAKING: RL



SEQUENTIAL DECISION MAKING: RL

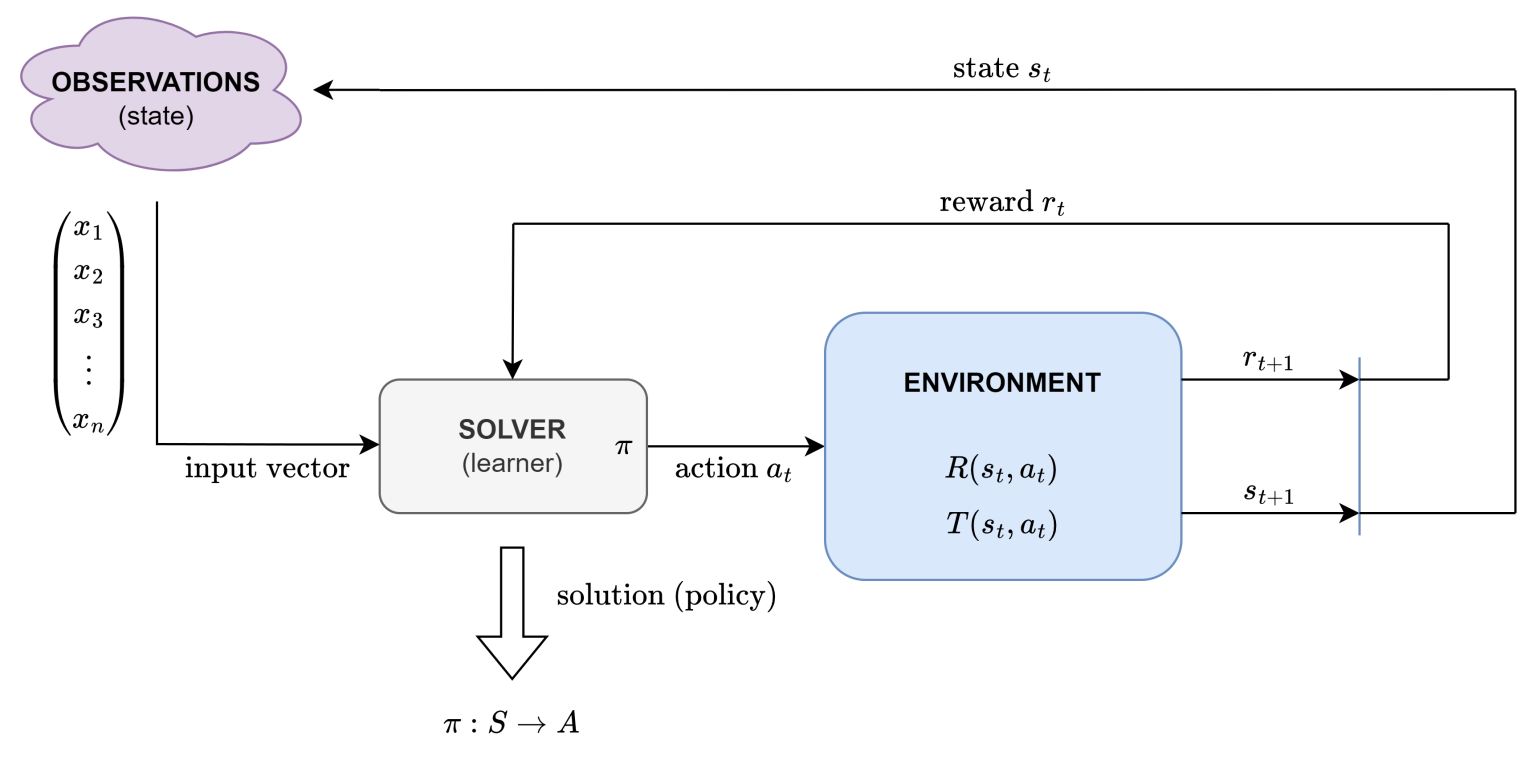


SEQUENTIAL DECISION MAKING: RL



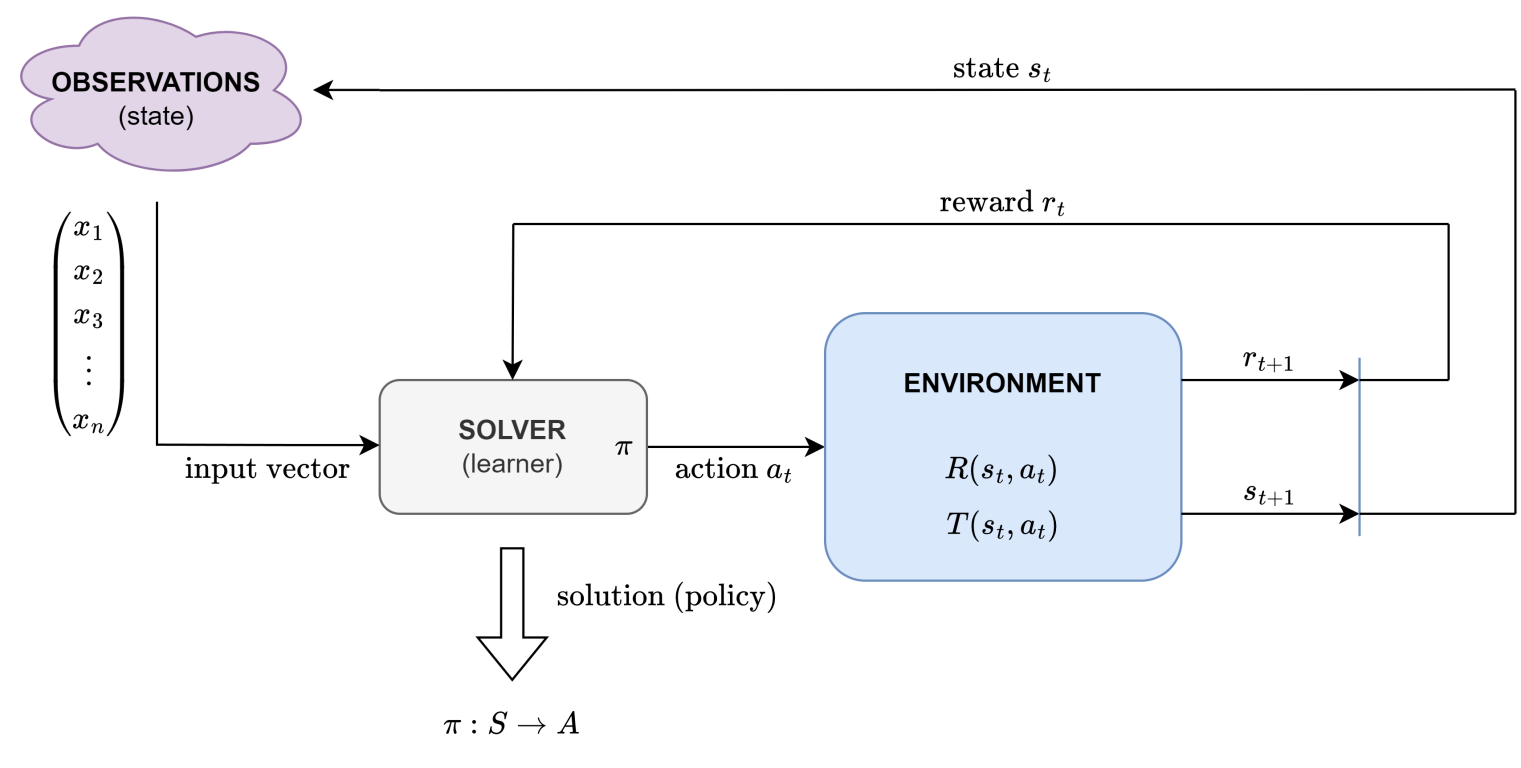
Understanding RL

HUMAN-LIKE APPROACH: LEARNING FROM INTERACTION



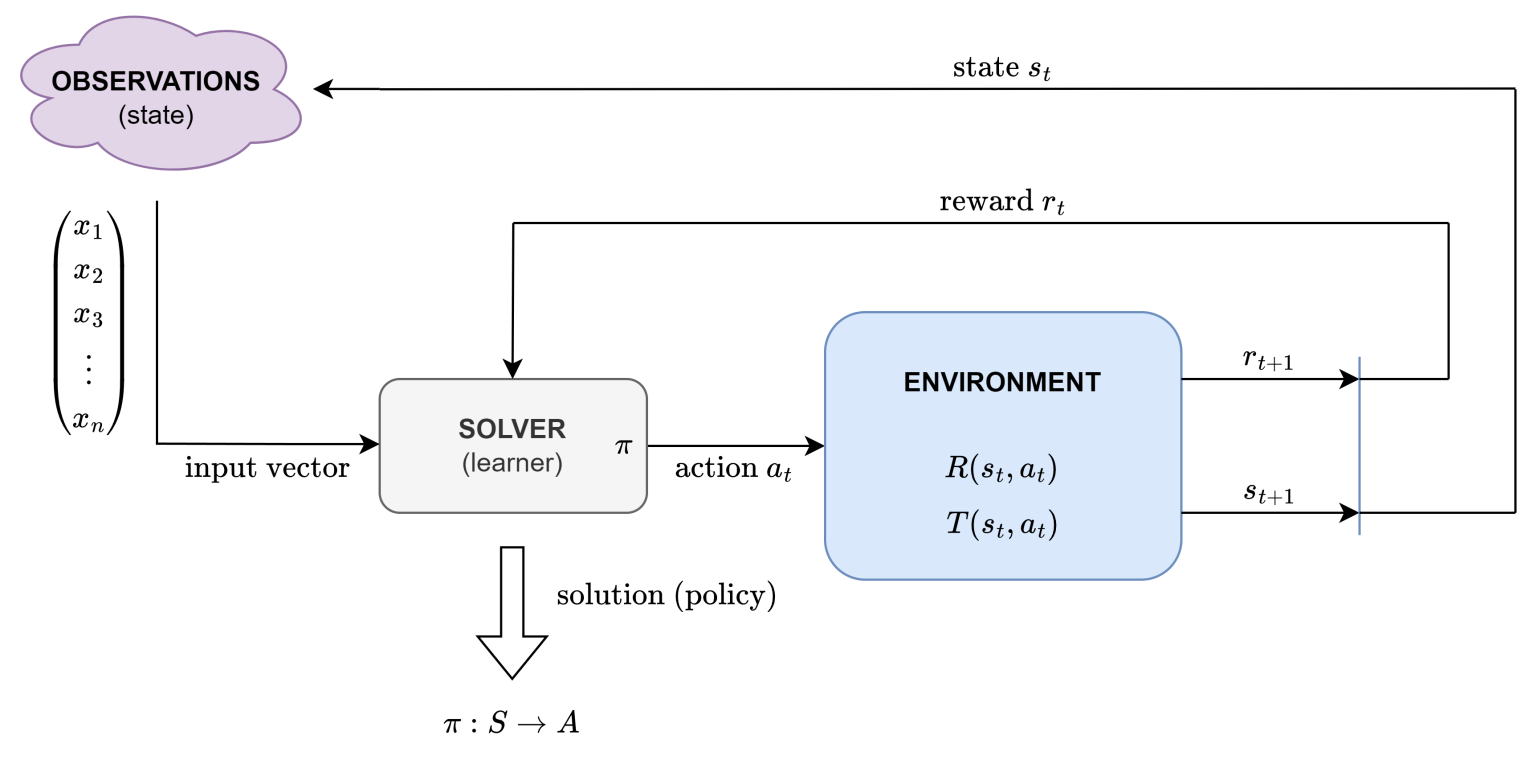
- **Action-oriented** with potential **uncertainty** about environment (different from Planning actions).

HUMAN-LIKE APPROACH: LEARNING FROM INTERACTION



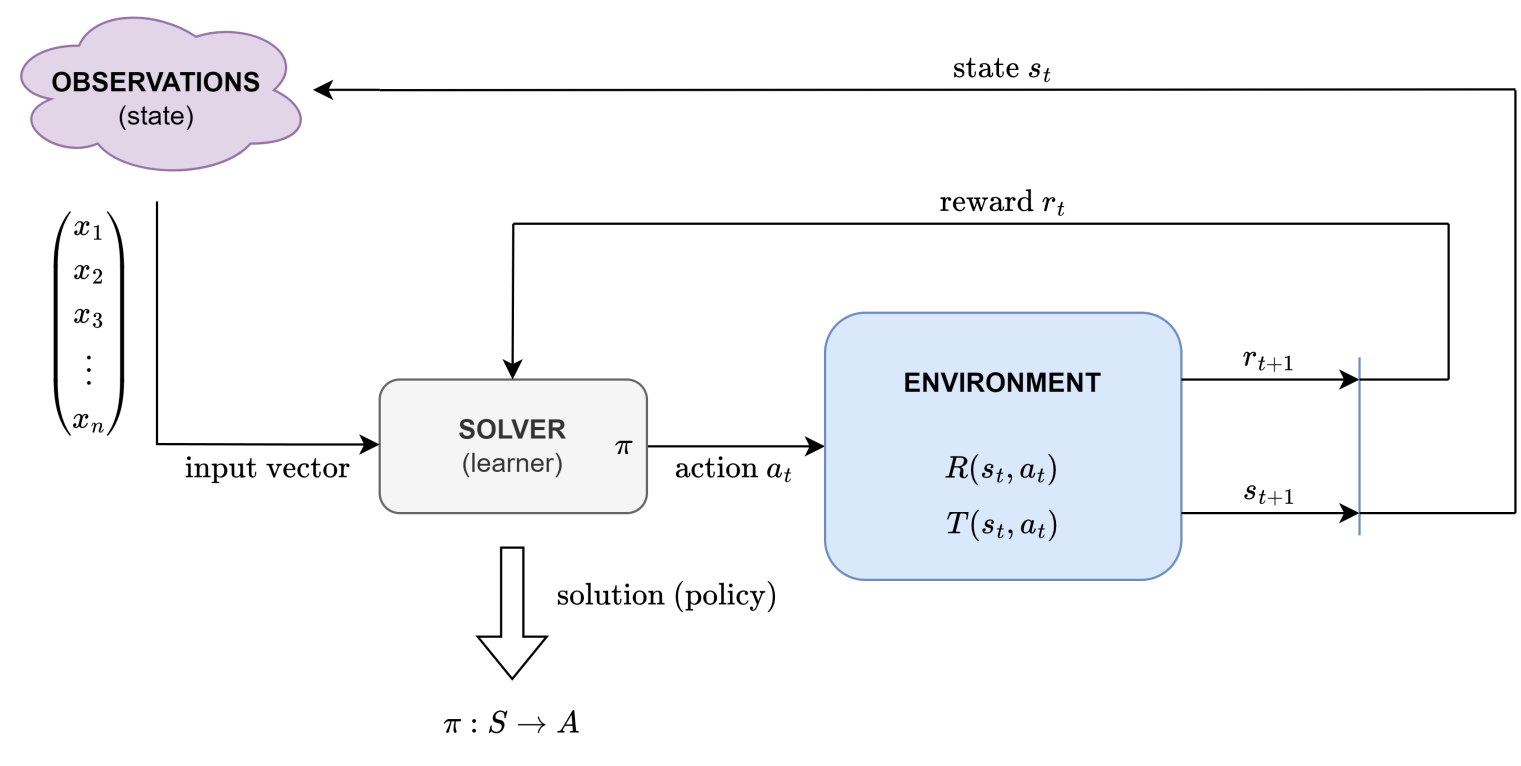
- **Learning from interaction** between an agent and its environment using positive or negative rewards of taking those actions.

HUMAN-LIKE APPROACH: LEARNING FROM INTERACTION



- The agent performance is being constantly **reshaped**.

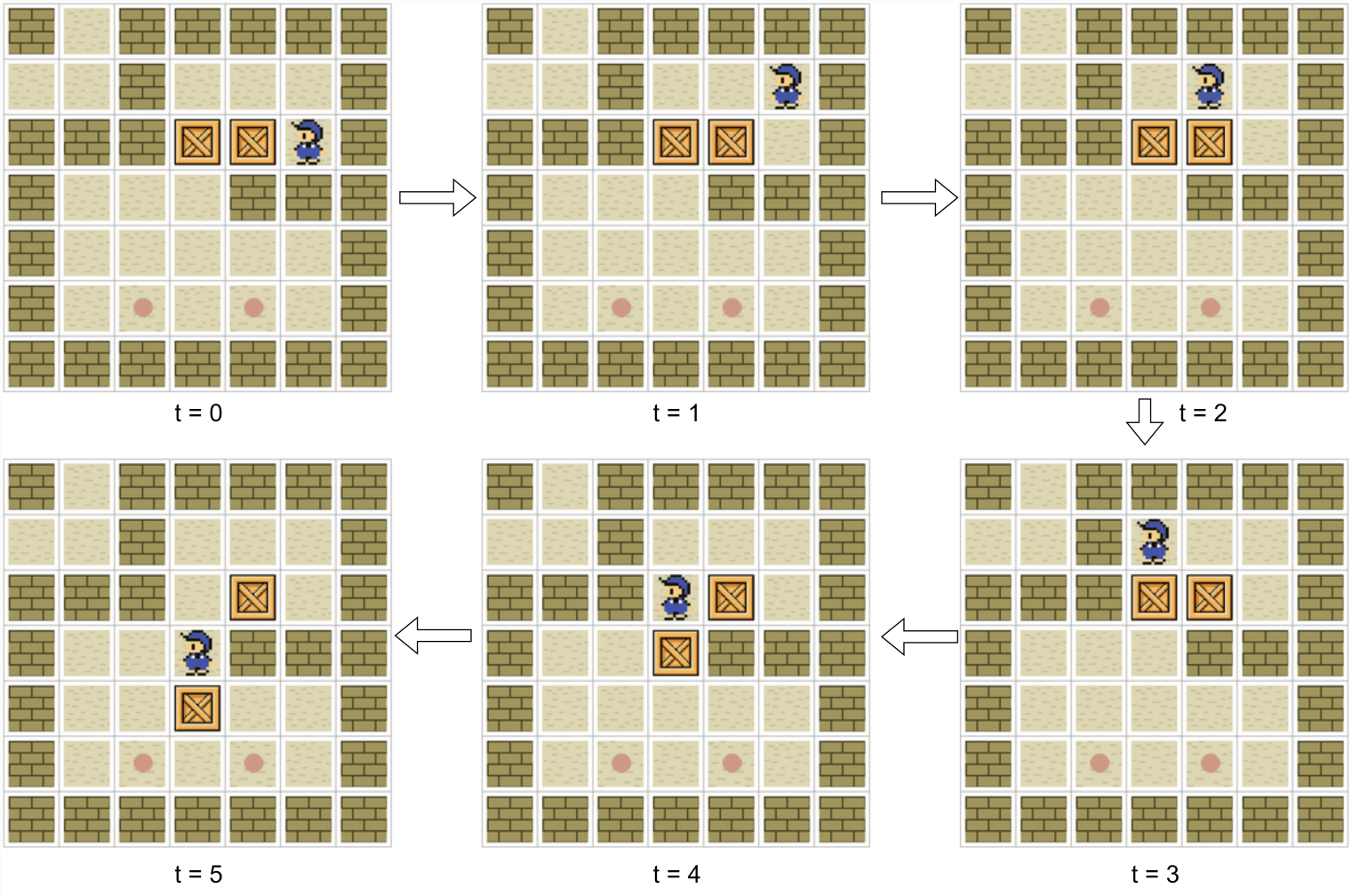
HUMAN-LIKE APPROACH: LEARNING FROM INTERACTION



- The agent stochastically takes action A_t by following a **policy** $\pi : \mathcal{S} \to \mathcal{A}$.

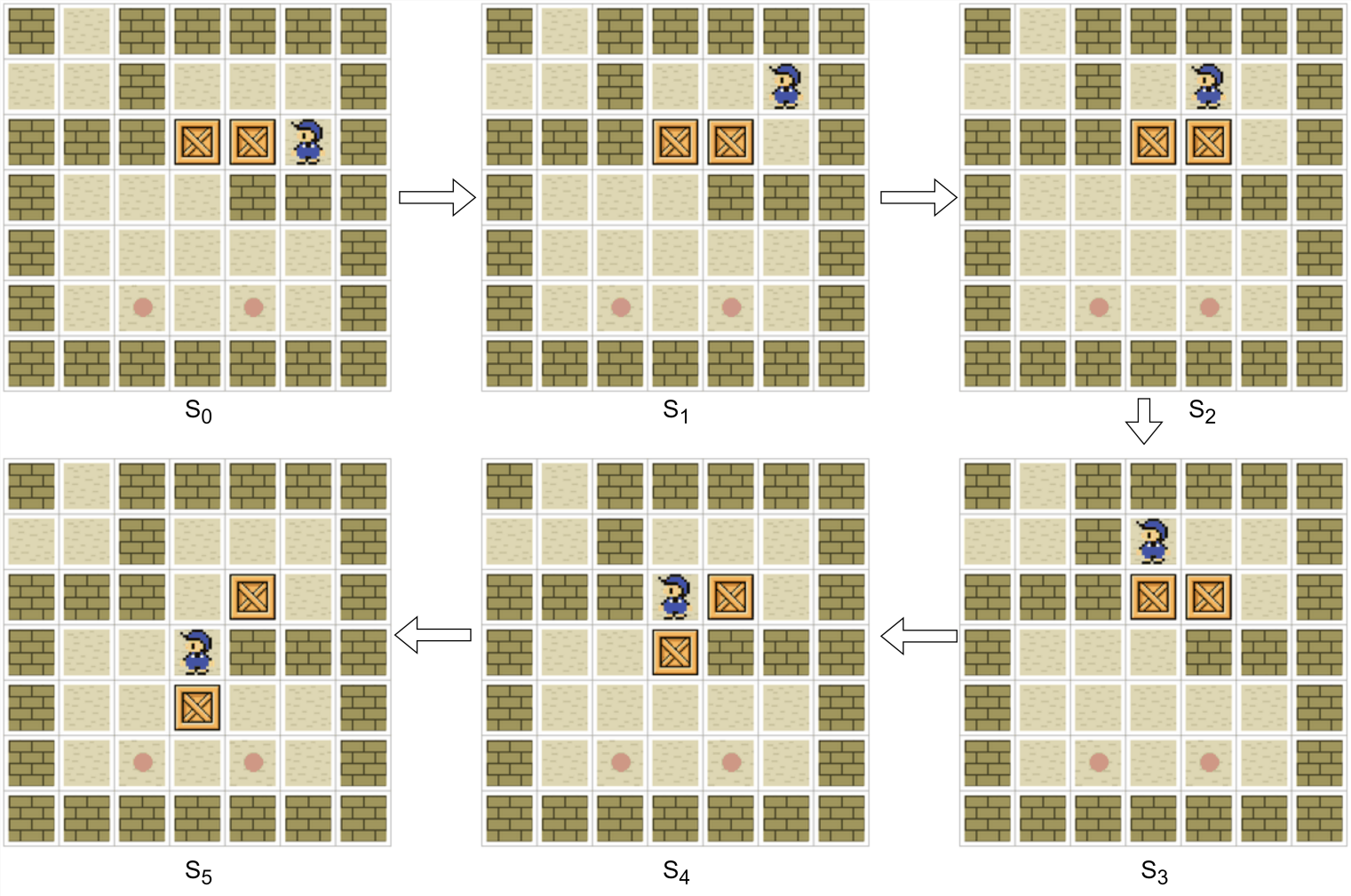
Elements of Reinforcement Learning

- Interaction between agent and environment occur in discrete **timesteps** $t = 0, 1, 2, 3, ...$



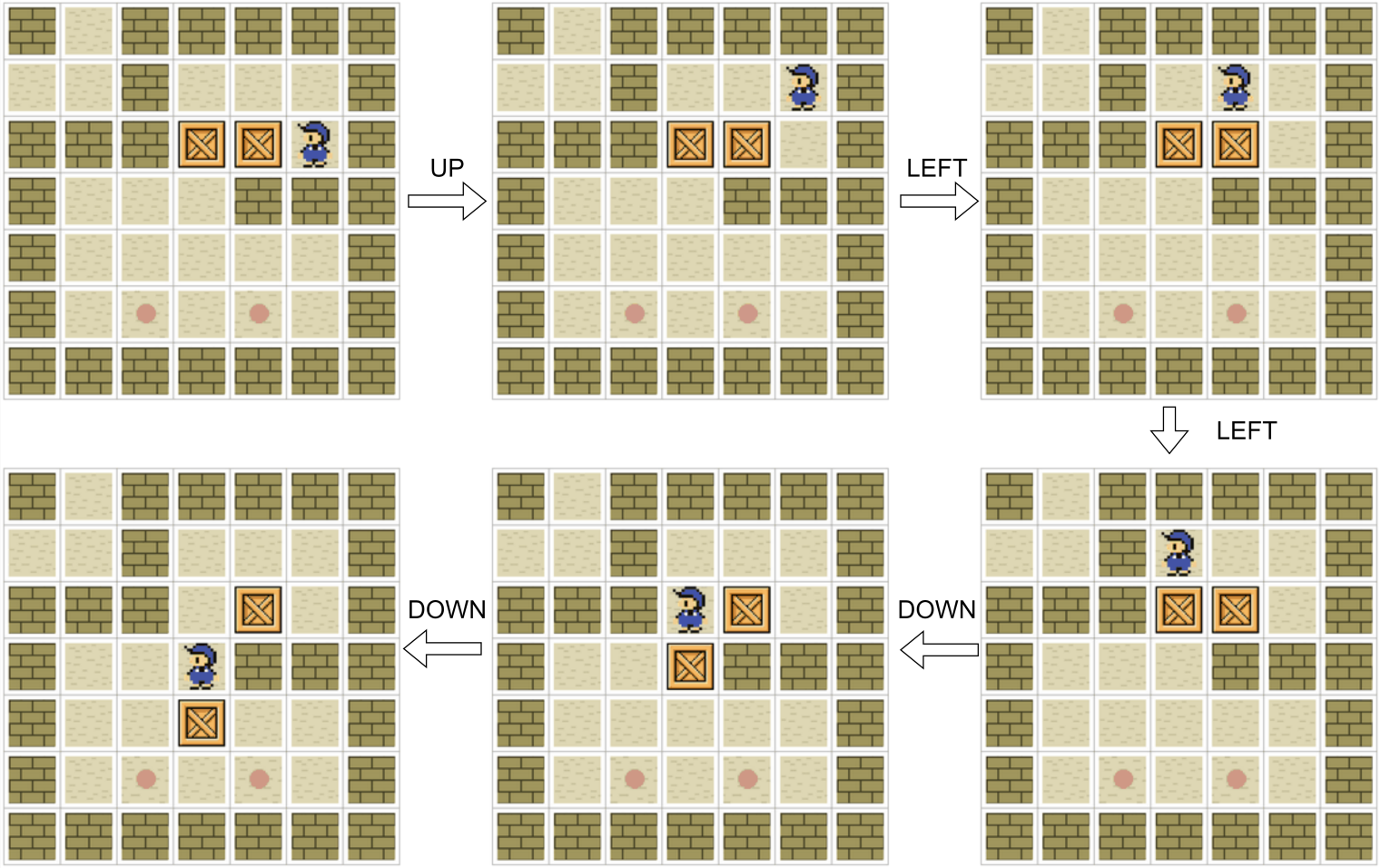
Elements of Reinforcement Learning

- Representation of the environment in the form of **states**.



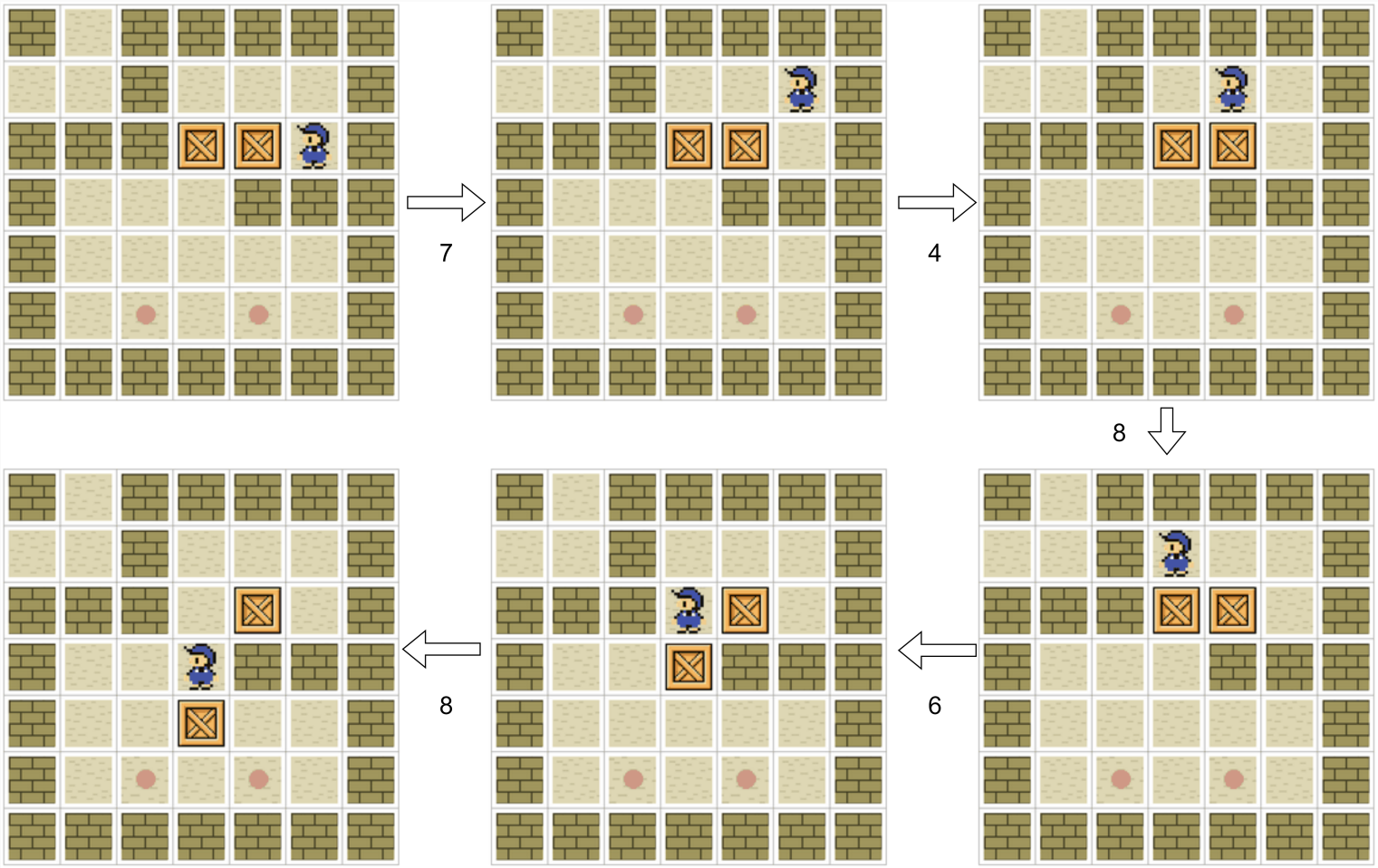
Elements of Reinforcement Learning

- Given the state S_t, the agent selects an **action** which changes the state.



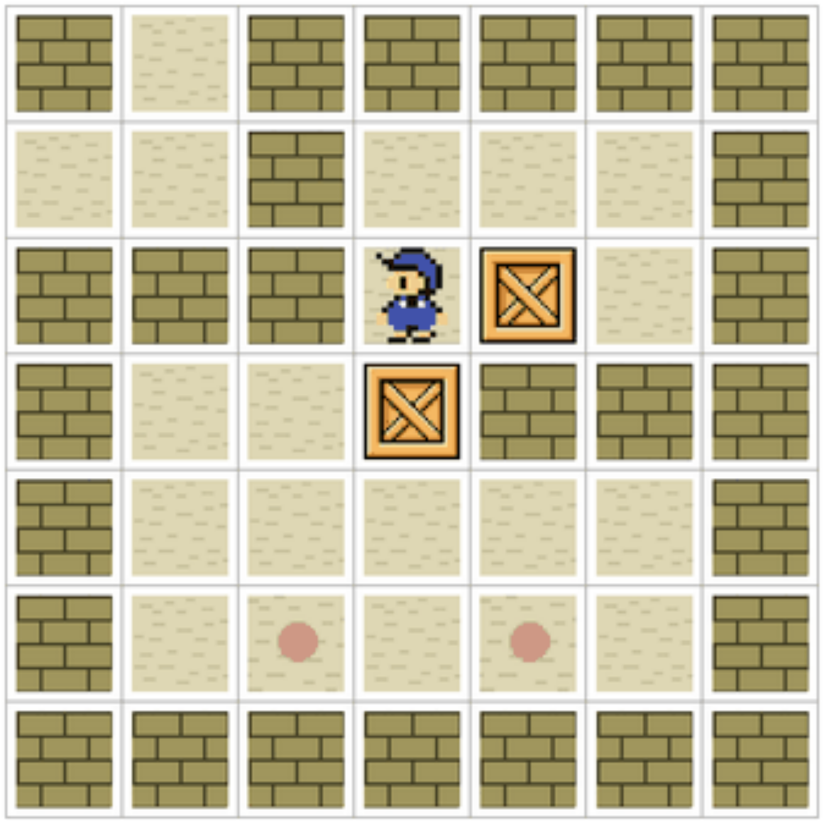
Elements of Reinforcement Learning

- As a consequence, the agent receives a numerical **reward** R_{t+1}.



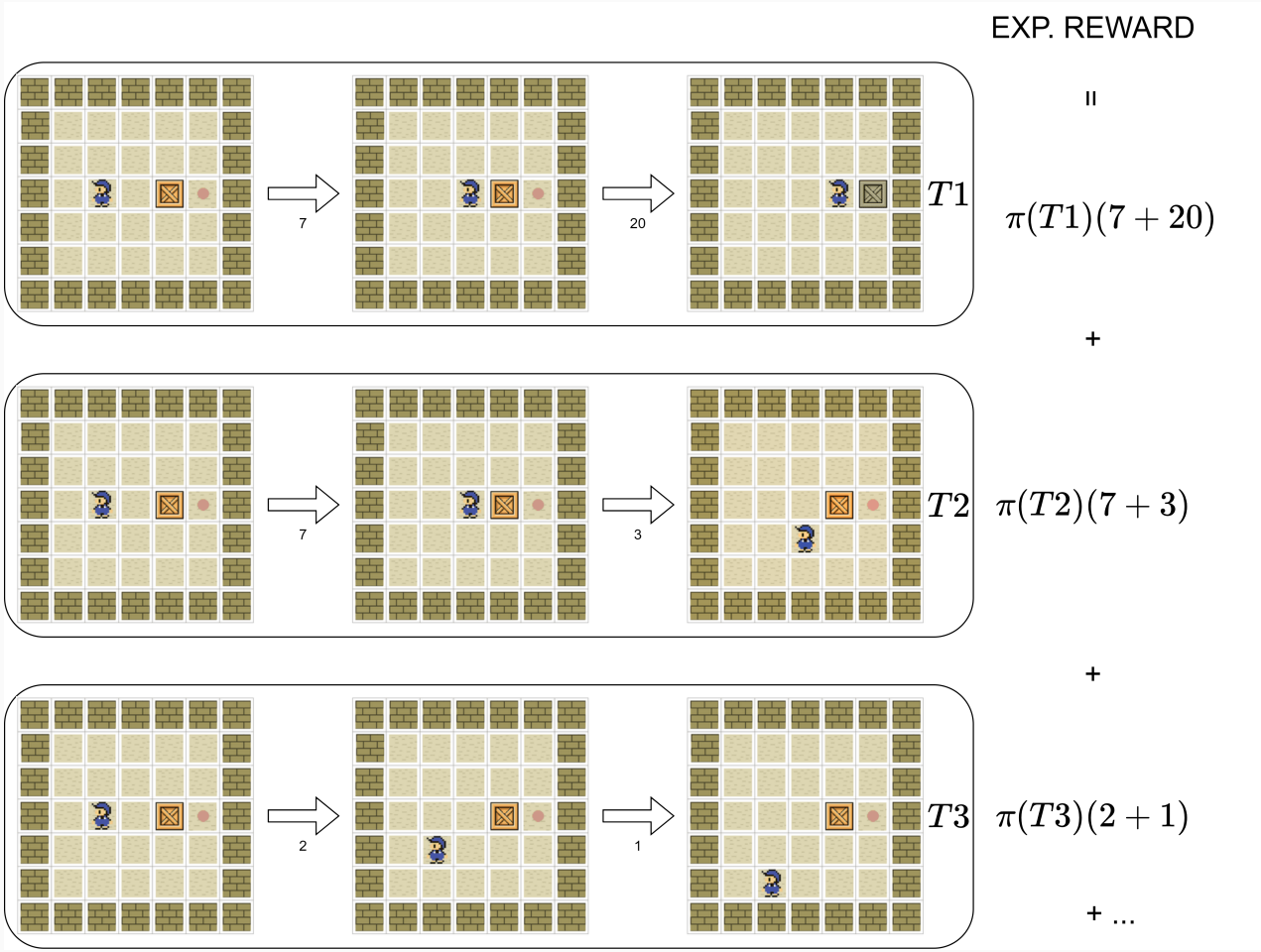
Elements of Reinforcement Learning

- The **policy** π_t defines a mapping between states and applicable actions. It determines the agent's modus operandi: $\pi_t(a|s)$ is the probability of taking action a in the state s.



Elements of Reinforcement Learning

- We want to shape the policy by taking actions in order to maximize the **expected reward**.



Markov Decision Processes (MDPs)

Markov Property

- **Markov property**: an agent's response depends only of the last representation of the environment.
- Any task that satisfies the Markov property is called a **Markov Decision Process** (MDP), which is a theoretical envelope modelled by a function p.
- Thus, RL can be represented by a MDP, because actions depend only on the current state.
- But this p function is usually not retrievable or not known!

Formal definition of a stochastic MDP

A stochastic MDP is defined as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$, such that:

- $\mathcal{S}$ is a set of **states** that represent the environment.
- $\mathcal{A}$ is a set of **actions** that the agent can take at each state in order to change it.
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow Pr(\mathcal{S})$ is a **transition function**. It maps each state and action to a probability distribution among states.
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ is a **reward function**. For each action-state pair, it provides the reward received after executing an action from a state and leading to another state.

Note that, starting from the same action-state pair (a, s), several states can (non-deterministically) be reached. We denote the probability of getting to a state s' from s by executing a as $p(s'|a, s)$.

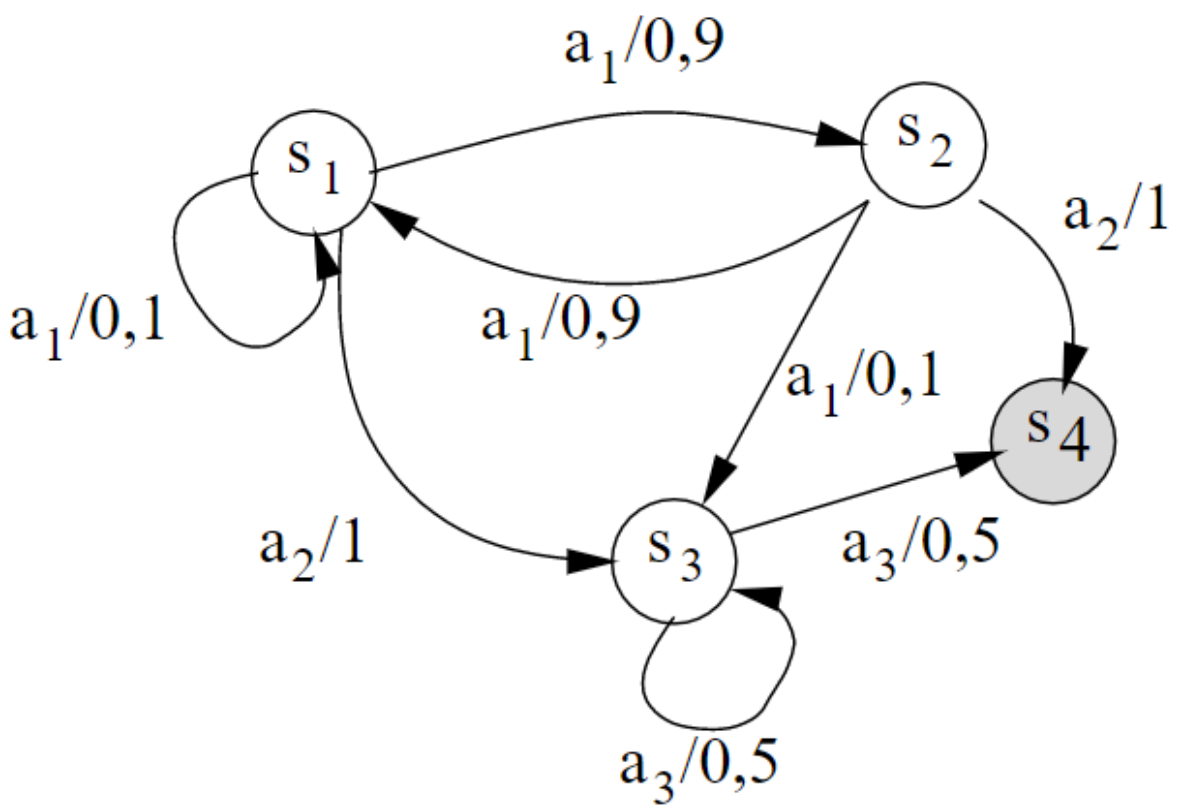
Formal definition of a deterministic MDP

A deterministic MDP is defined as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$, such that:

- $\mathcal{S}$ is a set of **states** that represent the environment.
- $\mathcal{A}$ is a set of **actions** that the agent can take at each state in order to change it.
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is a **transition function**. It maps each state and action to a single state.
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a **reward function**. For each action-state pair, it provides the reward received after executing an action from a state, as the arrival state is static.

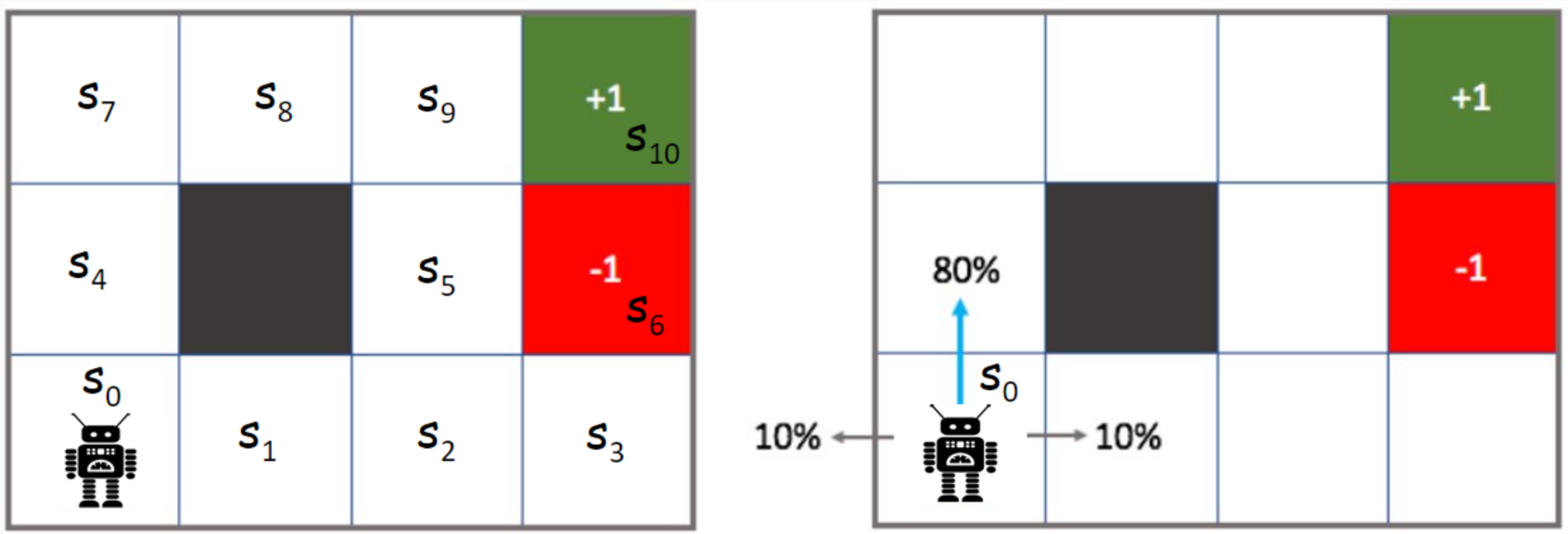
Note that a deterministic MDP can be modelled as a stochastic MDP. In that sense, the function $p(s'|a, s)$ can also be used in deterministic MDPs.

EXAMPLE OF A STOCHASTIC MDP



Here, $p(s_1|s_1, a_1) = 0.1,\ p(s_2|s_1, a_1) = 0.9, p(s_3|s_1, a_2) = 1$, etc.

Example of a stochastic MDP (with deterministic reward)



$\mathcal{S}$: Every square except the black one $(s_1, \ldots, s_{10})$

$\mathcal{A}$: up, down, left, right

$\mathcal{R}(s_9, \text{right}) = 1, \mathcal{R}(s_3, \text{up}) = -1, \ldots$

$$p(s_4|s_0, \text{up}) = 0.8$$
$$p(s_1|s_0, \text{up}) = 0.1$$
$$p(s_0|s_0, \text{up}) = 0.1$$

POLICY

A **stochastic policy** is a probability function over actions. It is defined as

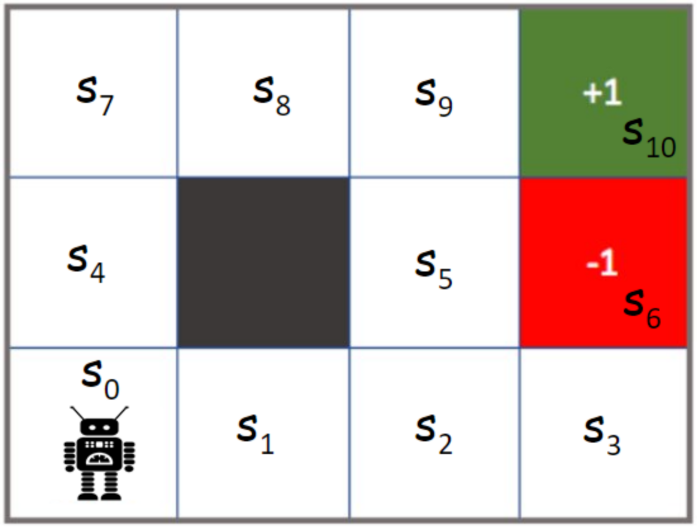
$$\pi : \mathcal{S} \rightarrow Pr(\mathcal{A})$$

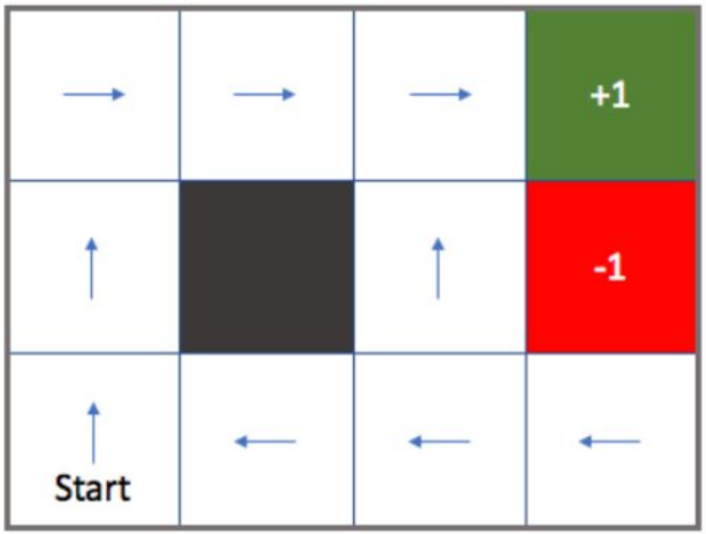
- The probability of taking an action a in state s is defined as $\pi(a|s)$.
- It is a function that determines the behavior of the agent.
- It is designed to navigate through the MDP.

The Reinforcement Learning objective

- An RL agent will navigate through a defined MDP, taking an action at each timestep, sampling from a **policy** π.
- The objective is to get **as much reward over time as possible**, prioritizing rewards closer to the current timestep, by **refining the policy** π.
- Because of the uncertainty of the system, we use random variables S_t, A_t and R_{t+1} to denote the **random events** of current state, action and reward, at each timestep t, respectively.
- The agent can or cannot have information about the transitions of the environment, that is, about the function p (or $\mathcal{T}$).

EXAMPLES OF OPTIMAL POLICIES





Policy examples

$\pi(\text{up}|s_0) = 0.95$

$\pi(\text{right}|s_0) = 0.05$

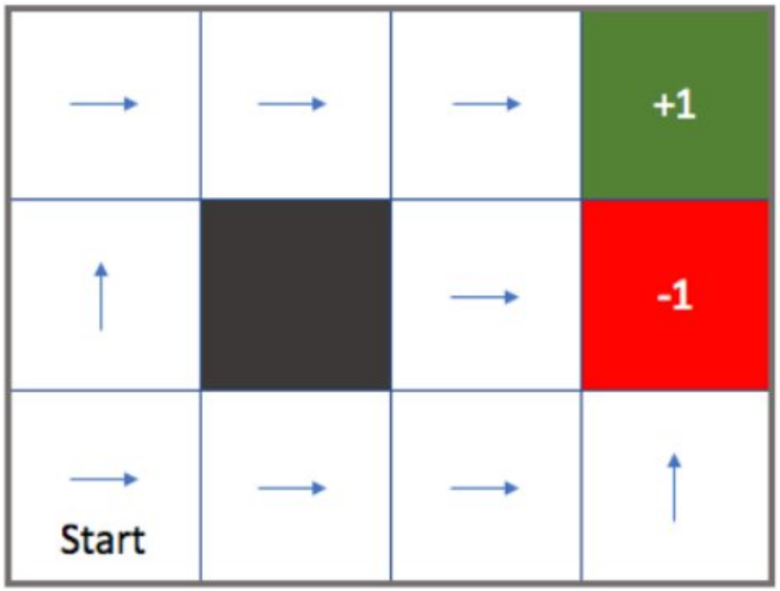
$\pi(\text{up}|s_2) = 0.75$

$\pi(\text{right}|s_2) = 0.1$

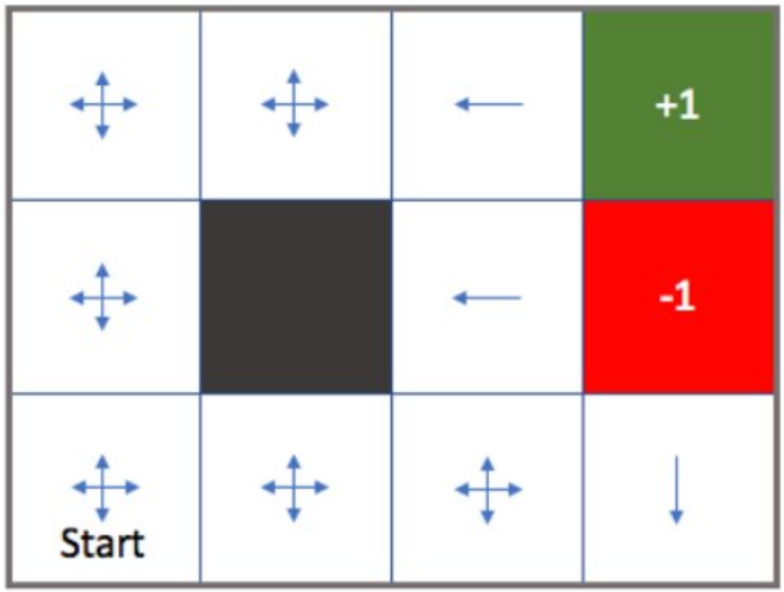
$\pi(\text{left}|s_2) = 0.15$

Optimal policy for $\mathcal{R}(s_i) = -0.04$

EXAMPLES OF OPTIMAL POLICIES



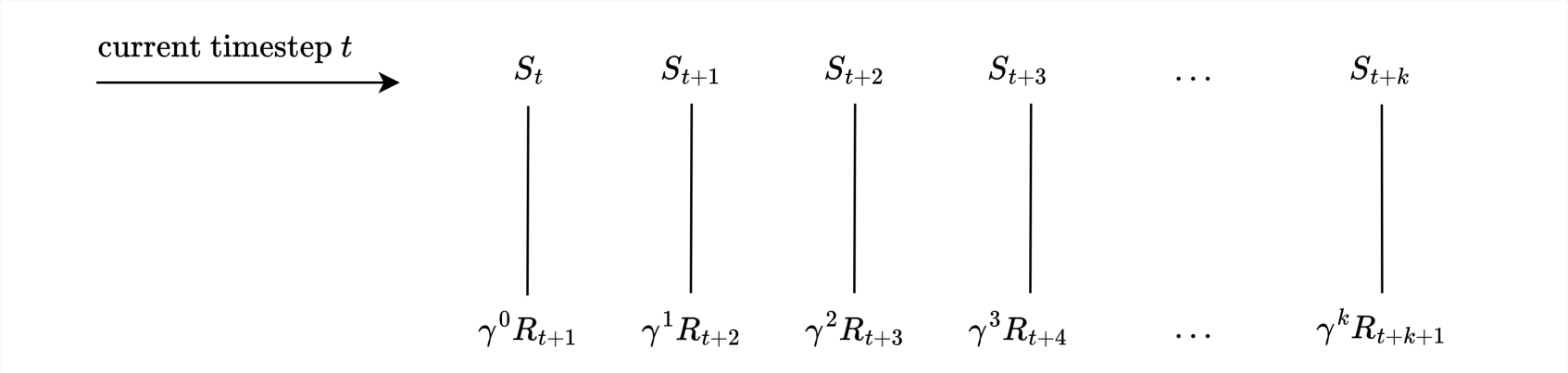
Optimal policy for $\mathcal{R}(s_i) = -2$



Optimal policy for $\mathcal{R}(s_i) = 2$

USEFULNESS

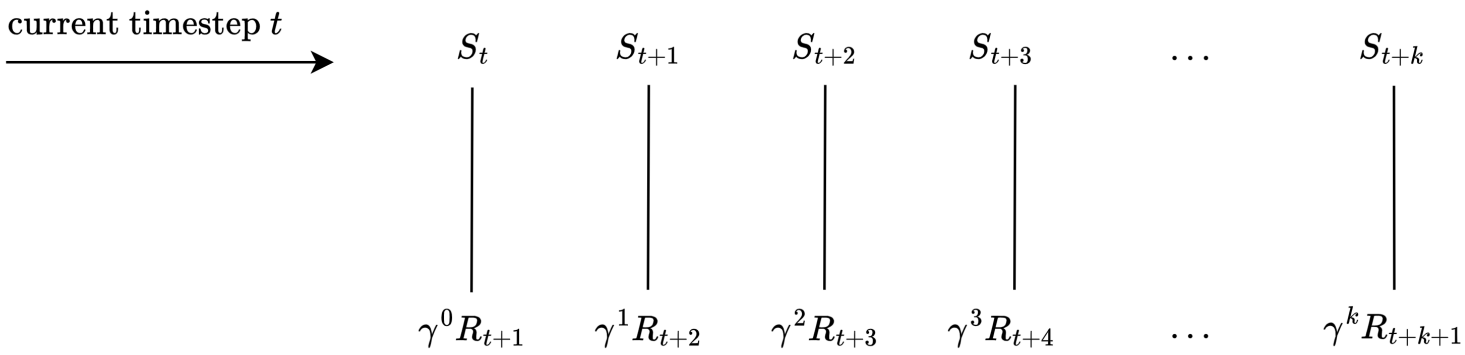
- It would be helpful to calculate the **usefulness** (value) **of a sequence of states** such that our 'view of the future' is not limited to the immediate reward of the next direct action.
- Idea: estimate **what is ahead** so that we can move toward the direction of positive reward and move away from negative reward.



- $0 \leq \gamma \leq 1$ (discount factor): favor **more immediate rewards**, i.e., deemphasize those more distant to the current state.

USEFULNESS

- ◇ It would be helpful to calculate the **usefulness** (value) **of a sequence of states** such that our 'view of the future' is not limited to the immediate reward of the next direct action.
- ◇ Idea: estimate **what is ahead** so that we can move toward the direction of positive reward and move away from negative reward.



- ◇ k: number of steps ahead in the future.

The Reinforcement Learning objective

- We define the **discounted reward** or **usefulness** of timestep t as the random variable:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

- It is a random variable that weights rewards using γ, and denotes the utility after timestep t.

- The objective is, then, to refine a policy π in order to maximize the expectation of the usefulness in the initial timestep:

$$\max_{\pi} \mathbb{E}_{\pi} G_0$$

Optimal policies and Dynamic Programming

VALUE FUNCTIONS

- We define the **state-value** of a state s as the expected return of following the policy π from s:

$$v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right]$$

- We define the **action-value** of a state s and action a pair as the expected return of following the policy π from s after taking action a:

$$q_\pi(s,a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right]$$

- If we know these functions, we can extract greedy policies using them.

Optimality

- Value functions define a partial order over policies:

$$\pi \geq \pi' \Leftrightarrow v_\pi(s) \geq v_{\pi'}(s) \; \forall s$$

- There is a theorem that guarantees the existence of a maximal element in a partially ordered set[1]. Thus, there always exists a policy π^* which is better than or equal to others, an **optimal policy**.

[1] https://en.wikipedia.org/wiki/Zorn%27s_lemma

OPTIMALITY

- Theoretically there can be more than one optimal policy, but all of them share the same optimal value functions:

$$v_*(s) = \max_\pi v_\pi(s)$$

$$q_*(s,a) = \max_\pi q_\pi(s,a)$$

- Optimality is entirely dependent on the reward definition.
- For example, we could force the Sokoban agent to go to the sides if we properly shape the rewards.
- State and action value functions are related with this formula:

$$v_*(s) = \max_a q_*(s,a)$$

BELLMAN EQUATIONS

- The **Bellman equations** are fundamental properties of value functions: recursiveness between any state s and its possible successors. For example, for the v function:

$$
\begin{aligned}
v_\pi(s) &= \mathbb{E}_\pi \left[G_t \mid S_t = s \right] \\
&= \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s \right] \\
&= \mathbb{E}_\pi \left[R_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k R_{t+k+2} \,\middle|\, S_t = s \right] \\
&= \sum_a \pi(a|s) \sum_{s'} p(s'|s,a) \left[r + \gamma \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+2} \,\middle|\, S_{t+1} = s' \right] \right] \\
&= \sum_a \pi(a|s) \sum_{s'} p(s'|s,a) \left[r + \gamma v_\pi(s') \right]
\end{aligned}
$$

Bellman equations

- Concretely, optimal value functions also follow these properties, but as they are optimal they are slightly different:

$$
\begin{aligned}
v_\pi(s) &= \max_a q_{\pi^*}(s,a) \\
&= \max_a \mathbb{E}_{\pi^*}\left[G_t \mid S_t = s, A_t = a\right] \\
&= \max_a \mathbb{E}_{\pi^*}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s, A_t = a\right] \\
&= \max_a \mathbb{E}_{\pi^*}\left[R_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k R_{t+k+2} \,\middle|\, S_t = s, A_t = a\right] \\
&= \max_a \mathbb{E}\left[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a\right] \\
&= \max_a \sum_{s'} p(s'|s,a)\left[r + \gamma v_*(s')\right]
\end{aligned}
$$

- Proceedings for q_* function are similar.

Dynamic programming

If we **know** the transition distribution of the MDP (p or $\mathcal{T}$), we can directly calculate an optimal policy using several methods, such as policy iteration or value iteration.

- ◇ Based on dynamic programming algorithms.
- ◇ Methods that converge at the limit.
- ◇ They yield policies arbitrarily accurate/optimal.
- ◇ Based on recursive relations between value functions using Bellman equations.
- ◇ Directly use the transition system of the MDP, so it has to be known (example: a card game).

VALUE ITERATION: INTUITION

- ◇ Approximate the optimal value function (optimal policy) using the dynamics of the system.
- ◇ Transition probabilities $p(s'|s,a)$ are known.
- ◇ Convergence until the difference between last iteration and current is smaller than a small positive number θ.

VALUE ITERATION

- The v_* function is a fixed point[2] of the Bellman equation (3.17).
- We can use that equation as an update rule: convergence and uniqueness are guaranteed (we need p).
- In our case reward is fixed, so $p(s', r|s, a) = p(s'|s, a)$.

Initialize array V arbitrarily (e.g., $V(s) = 0$ for all $s \in \mathcal{S}^+$)

Repeat
  $\Delta \leftarrow 0$
  For each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, π, such that
  $\pi(s) = \arg\max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$

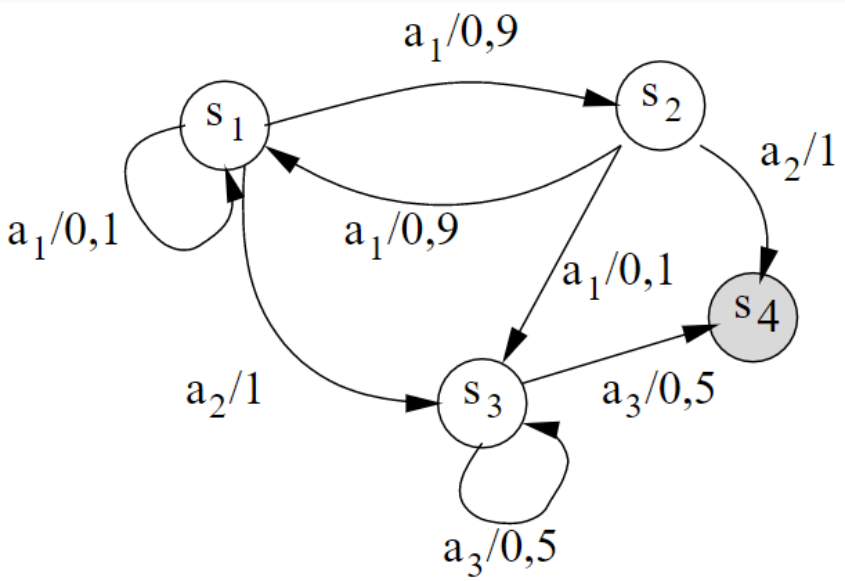
[2] https://en.wikipedia.org/wiki/Fixed-point_iteration (The idea)

VALUE ITERATION: EXAMPLE

FIRST ITERATION

	s_1	s_2	s_3	s_4
V	0	0	0	0

$$V(s_1) = \max_{\substack{a_1,a_2,\\ a_3}} \left\{ \sum_{\substack{s'=s_1,\\ s_2,s_3,s_4}} p(s'|s_1,a)[r+\gamma V(s')] \right\}$$



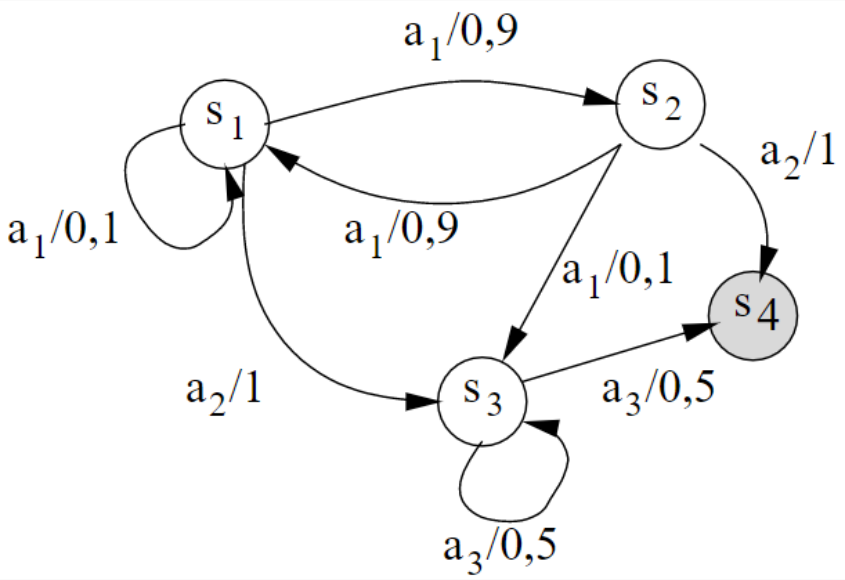
$$\begin{aligned} V(s_1) = \max\{&p(s_1|s_1,a_1)[r+\gamma V(s_1)] + p(s_2|s_1,a_1)[r+\gamma V(s_2)] \\ &+ p(s_3|s_1,a_1)[r+\gamma V(s_3)] + p(s_4|s_1,a_1)[r+\gamma V(s_4)], \\ &p(s_1|s_1,a_2)[r+\gamma V(s_1)] + p(s_2|s_1,a_2)[r+\gamma V(s_2)] \\ &+ p(s_3|s_1,a_2)[r+\gamma V(s_3)] + p(s_4|s_1,a_2)[r+\gamma V(s_4)]\} \end{aligned}$$

VALUE ITERATION: EXAMPLE

FIRST ITERATION

	s_1	s_2	s_3	s_4
V	0	0	0	0

$$V(s_1) = \max_{\substack{a_1,a_2,\\ a_3}} \left\{ \sum_{\substack{s'=s_1,\\ s_2,s_3,s_4}} p(s'|s_1,a)[r+\gamma V(s')] \right\}$$



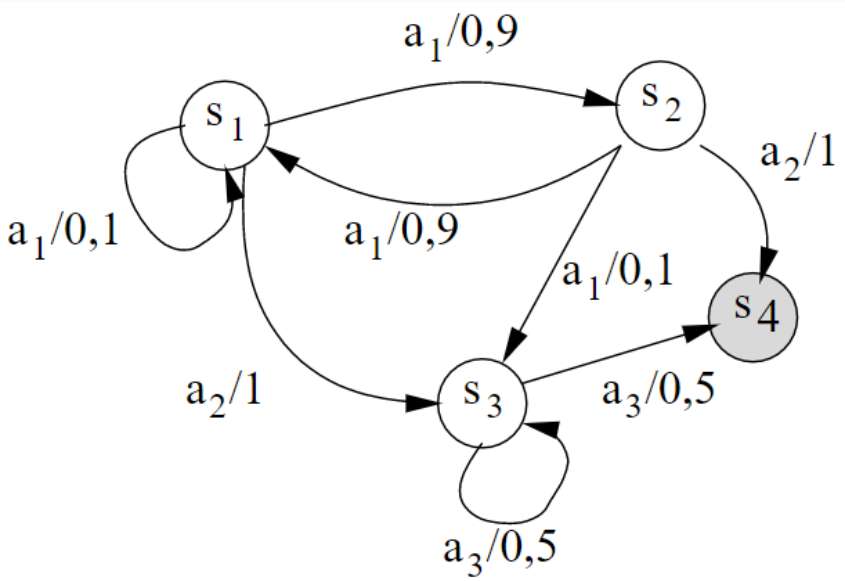
$$\begin{aligned} V(s_1) = \max\{&0.1[0+\gamma\cdot 0]+0.9[0+\gamma\cdot 0] \\ &+0[0+\gamma\cdot 0]+0[0+\gamma\cdot 0], \\ &0[0+\gamma\cdot 0]+0[0+\gamma\cdot 0] \\ &+1[0+\gamma\cdot 0]+0[0+\gamma\cdot 0]\} \\ =\max\{&0,0\}=0 \end{aligned}$$

VALUE ITERATION: EXAMPLE

FIRST ITERATION

	s_1	s_2	s_3	s_4
V	$0 \rightarrow \mathbf{0}$	0	0	0

$$V(s_2) = \max_{\substack{a_1,a_2,\\ a_3}} \left\{ \sum_{\substack{s'=s_1,\\ s_2,s_3,s_4}} p(s'|s_2,a)[r+\gamma V(s')] \right\}$$



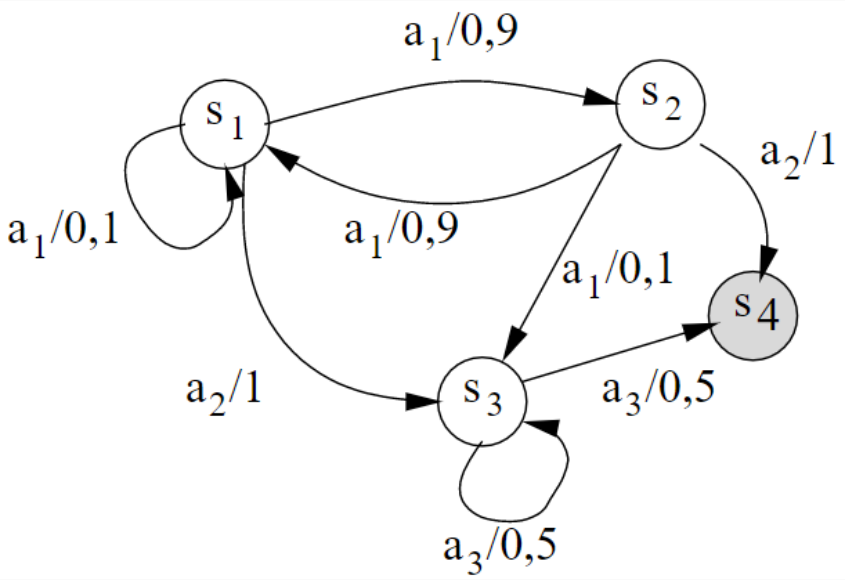
$$\begin{aligned} V(s_2) = \max\{&p(s_1|s_2,a_1)[r+\gamma V(s_1)] + p(s_2|s_2,a_1)[r+\gamma V(s_2)] \\ &+ p(s_3|s_2,a_1)[r+\gamma V(s_3)] + p(s_4|s_2,a_1)[r+\gamma V(s_4)], \\ &p(s_1|s_2,a_2)[r+\gamma V(s_1)] + p(s_2|s_2,a_2)[r+\gamma V(s_2)] \\ &+ p(s_3|s_2,a_2)[r+\gamma V(s_3)] + p(s_4|s_2,a_2)[r+\gamma V(s_4)]\} \end{aligned}$$

VALUE ITERATION: EXAMPLE

FIRST ITERATION

	s_1	s_2	s_3	s_4
V	$0 \rightarrow \mathbf{0}$	0	0	0

$$V(s_2) = \max_{\substack{a_1,a_2,\\ a_3}} \left\{ \sum_{\substack{s'=s_1,\\ s_2,s_3,s_4}} p(s'|s_2,a)[r + \gamma V(s')] \right\}$$



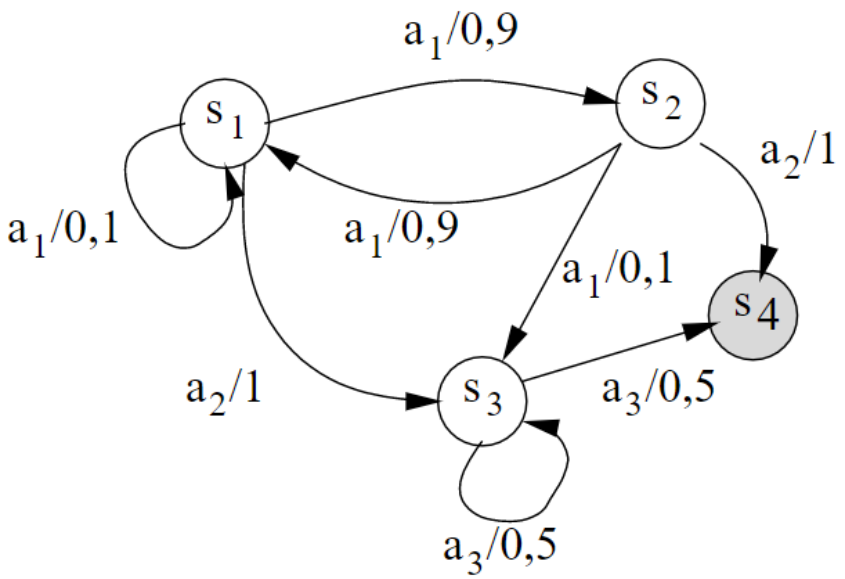
$$\begin{aligned} V(s_2) = \max\{&0.9[0 + \gamma \cdot 0] + 0[0 + \gamma \cdot 0] \\ &+ 0.1[0 + \gamma \cdot 0] + 0[0 + \gamma \cdot 0], \\ &0[0 + \gamma \cdot 0] + 0[0 + \gamma \cdot 0] \\ &+ 0[0 + \gamma \cdot 0] + 1[1 + \gamma \cdot 0]\} \\ = \max\{&0, 1\} = 1 \end{aligned}$$

VALUE ITERATION: EXAMPLE

FIRST ITERATION

	s_1	s_2	s_3	s_4
V	$0 \rightarrow \mathbf{0}$	$0 \rightarrow \mathbf{1}$	0	0

$$V(s_3) = \max_{\substack{a_1,a_2,\\ a_3}} \left\{ \sum_{\substack{s'=s_1,\\ s_2,s_3,s_4}} p(s'|s_3,a)[r+\gamma V(s')] \right\}$$



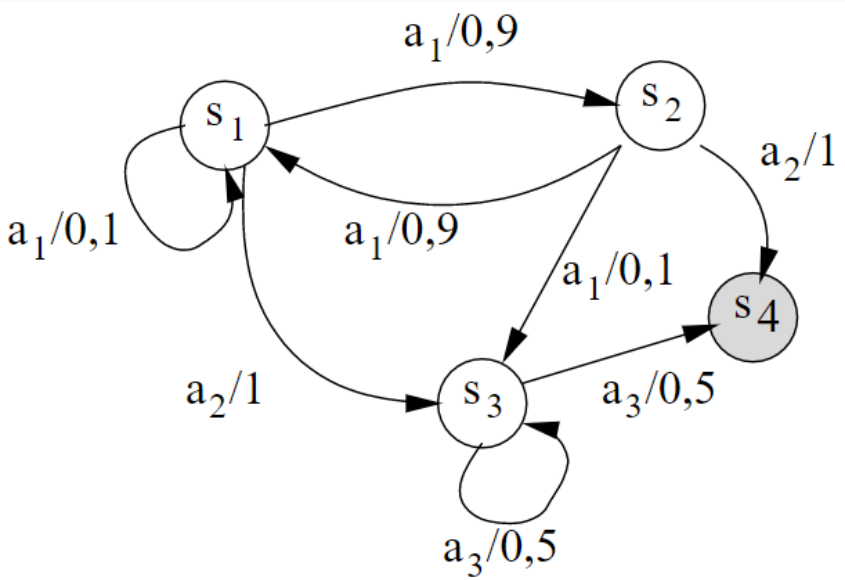
$$V(s_3) = \max\{p(s_1|s_3,a_3)[r+\gamma V(s_1)] + p(s_2|s_3,a_3)[r+\gamma V(s_2)] + p(s_3|s_3,a_3)[r+\gamma V(s_3)] + p(s_4|s_3,a_3)[r+\gamma V(s_4)]\}$$

VALUE ITERATION: EXAMPLE

FIRST ITERATION

	s_1	s_2	s_3	s_4
V	$0 \rightarrow \mathbf{0}$	$0 \rightarrow \mathbf{1}$	0	0

$$V(s_3) = \max_{\substack{a_1,a_2,\\ a_3}} \left\{ \sum_{\substack{s'=s_1,\\ s_2,s_3,s_4}} p(s'|s_3,a)[r + \gamma V(s')] \right\}$$



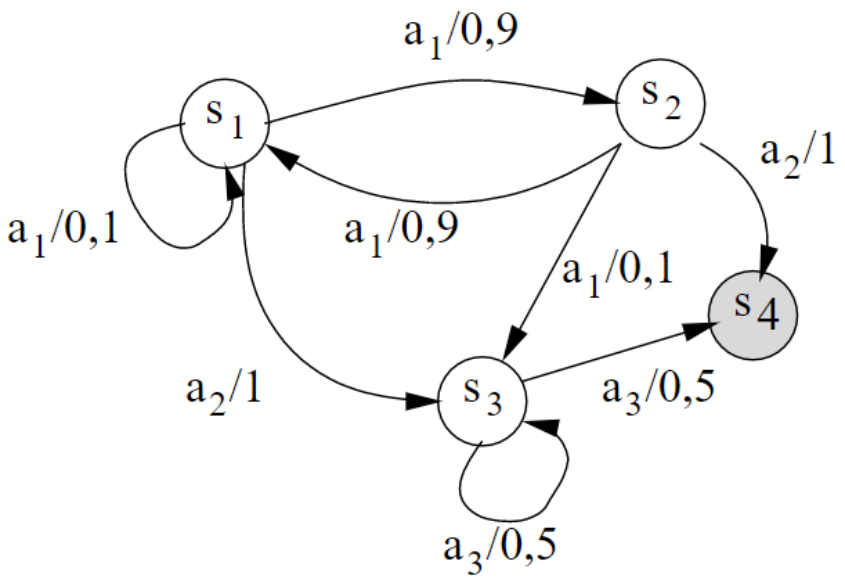
$$\begin{aligned} V(s_3) &= \max\{0[0 + \gamma \cdot 0] + 0[0 + \gamma \cdot 0] \\ &\quad + 0.5[0 + \gamma \cdot 0] + 0.5[1 + \gamma \cdot 0]\} \\ &= \max\{0.5\} = 0.5 \end{aligned}$$

VALUE ITERATION: EXAMPLE

FIRST ITERATION

	s_1	s_2	s_3	s_4
V	$0 \rightarrow \mathbf{0}$	$0 \rightarrow \mathbf{1}$	$0 \rightarrow \mathbf{0.5}$	0

$$V(s_4) = \max_{\substack{a_1,a_2,\\ a_3}} \left\{ \sum_{\substack{s'=s_1,\\ s_2,s_3,s_4}} p(s'|s_4,a)[r + \gamma V(s')] \right\}$$



$$V(s_4) = 0$$

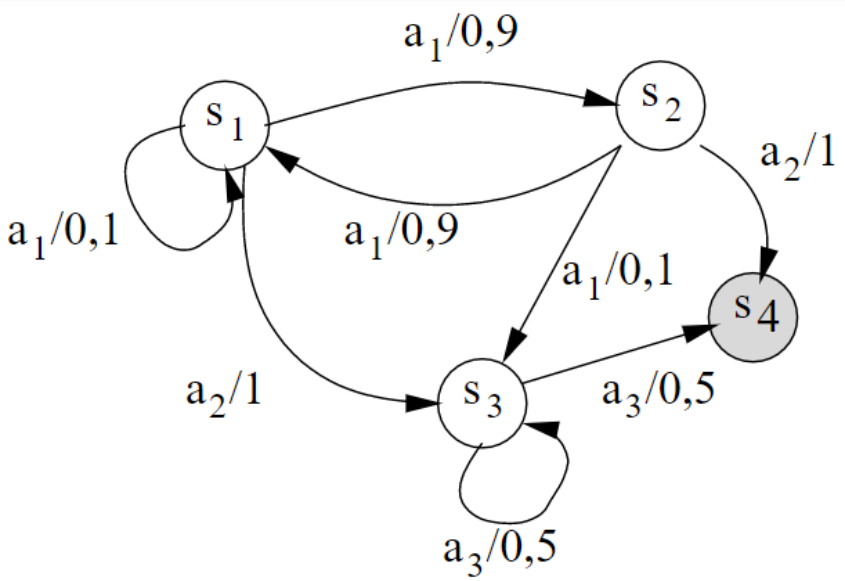
$$\Delta = \max\{0-0, 1-0, 0.5-0, 0-0\} = 1 > \theta = 0.01$$

VALUE ITERATION: EXAMPLE

SECOND ITERATION

	s_1	s_2	s_3	s_4
V	0	1	0.5	0

$$V(s_1) = \max_{\substack{a_1,a_2,\\ a_3}} \left\{ \sum_{\substack{s'=s_1,\\ s_2,s_3,s_4}} p(s'|s_1,a)[r+\gamma V(s')] \right\}$$



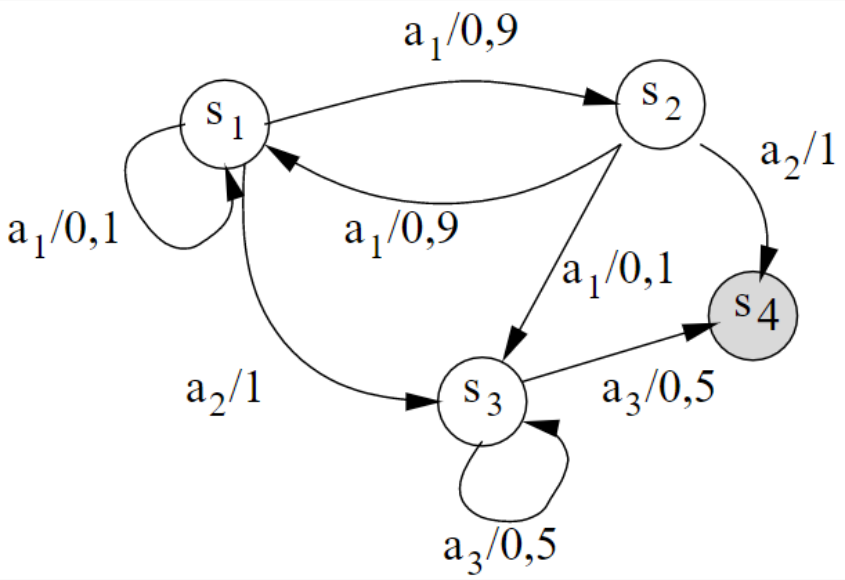
$$\begin{aligned} V(s_1) = \max\{&p(s_1|s_1,a_1)[r+\gamma V(s_1)] + p(s_2|s_1,a_1)[r+\gamma V(s_2)] \\ &+ p(s_3|s_1,a_1)[r+\gamma V(s_3)] + p(s_4|s_1,a_1)[r+\gamma V(s_4)], \\ &p(s_1|s_1,a_2)[r+\gamma V(s_1)] + p(s_2|s_1,a_2)[r+\gamma V(s_2)] \\ &+ p(s_3|s_1,a_2)[r+\gamma V(s_3)] + p(s_4|s_1,a_2)[r+\gamma V(s_4)]\} \end{aligned}$$

VALUE ITERATION: EXAMPLE

SECOND ITERATION

	s_1	s_2	s_3	s_4
V	0	1	0.5	0

$$V(s_1) = \max_{\substack{a_1,a_2,\\ a_3}} \left\{ \sum_{\substack{s'=s_1,\\ s_2,s_3,s_4}} p(s'|s_1,a)[r+\gamma V(s')] \right\}$$



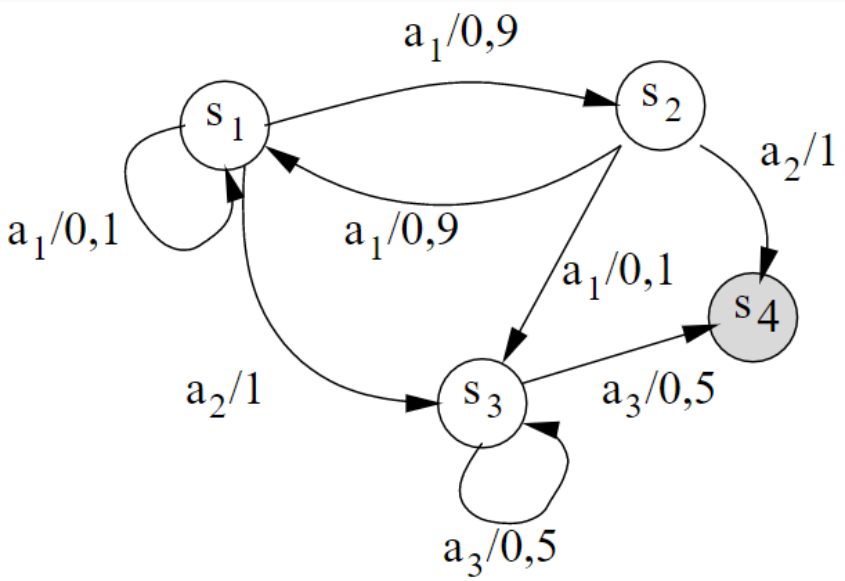
$$\begin{aligned} V(s_1) = \max\{&0.1[0+\gamma\cdot 0]+0.9[0+\gamma\cdot 1] \\ &+0[0+\gamma\cdot 0.5]+0[0+\gamma\cdot 0], \\ &0[0+\gamma\cdot 0]+0[0+\gamma\cdot 1] \\ &+1[0+\gamma\cdot 0.5]+0[0+\gamma\cdot 0]\} \\ = \max\{&\gamma 0.9, 0.5\}_{\gamma=1} = 0.9 \end{aligned}$$

VALUE ITERATION: EXAMPLE

SECOND ITERATION

	s_1	s_2	s_3	s_4
V	$0 \rightarrow \mathbf{0.9}$	1	0.5	0

$$V(s_2) = \max_{\substack{a_1,a_2,\\ a_3}} \left\{ \sum_{\substack{s'=s_1,\\ s_2,s_3,s_4}} p(s'|s_2,a)[r+\gamma V(s')] \right\}$$



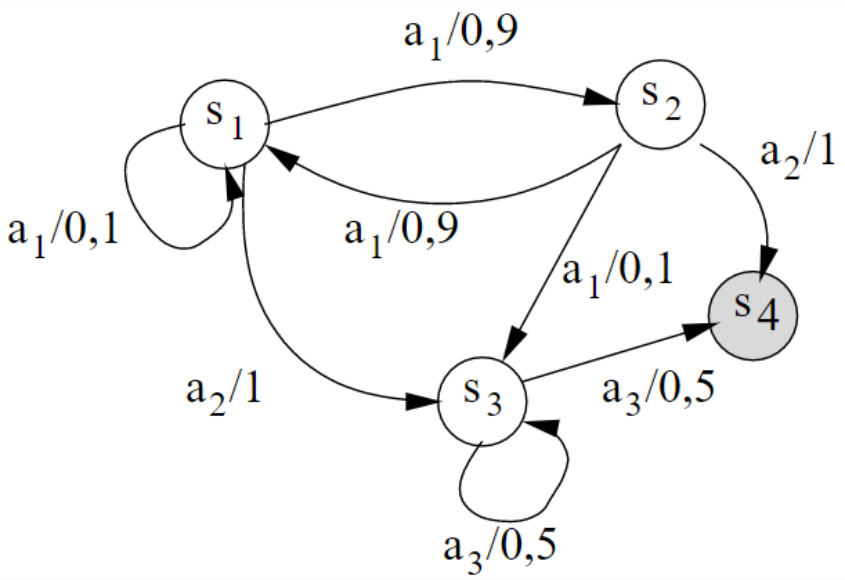
$$\begin{aligned} V(s_2) = \max\{&p(s_1|s_2,a_1)[r+\gamma V(s_1)] + p(s_2|s_2,a_1)[r+\gamma V(s_2)] \\ &+ p(s_3|s_2,a_1)[r+\gamma V(s_3)] + p(s_4|s_2,a_1)[r+\gamma V(s_4)], \\ &p(s_1|s_2,a_2)[r+\gamma V(s_1)] + p(s_2|s_2,a_2)[r+\gamma V(s_2)] \\ &+ p(s_3|s_2,a_2)[r+\gamma V(s_3)] + p(s_4|s_2,a_2)[r+\gamma V(s_4)]\} \end{aligned}$$

VALUE ITERATION: EXAMPLE

SECOND ITERATION

	s_1	s_2	s_3	s_4
V	$0 \rightarrow \mathbf{0.9}$	1	0.5	0

$$V(s_2) = \max_{\substack{a_1,a_2,\\ a_3}} \left\{ \sum_{\substack{s'=s_1,\\ s_2,s_3,s_4}} p(s'|s_2,a)[r + \gamma V(s')] \right\}$$



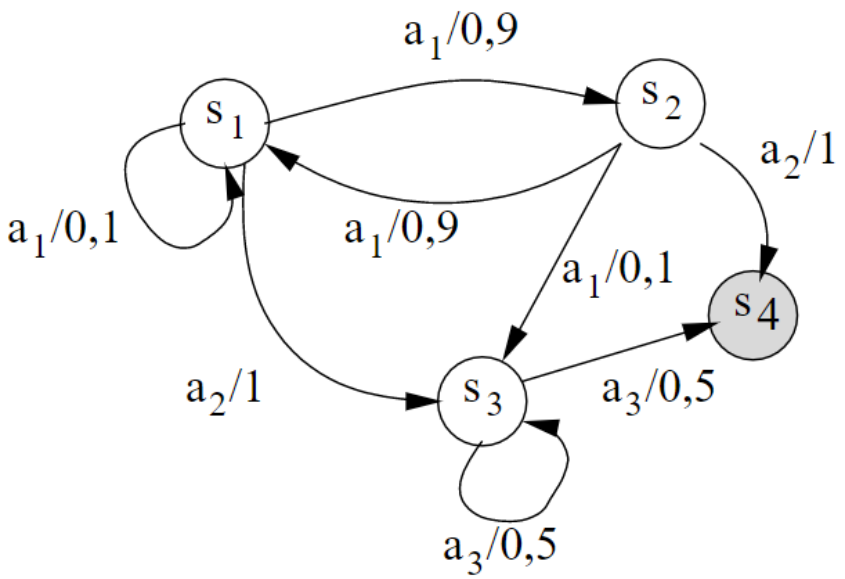
$$\begin{aligned} V(s_2) = \max\{&0.9[0 + \gamma \cdot 0] + 0[0 + \gamma \cdot 1] \\ &+ 0.1[0 + \gamma \cdot 0.5] + 0[0 + \gamma \cdot 0], \\ &0[0 + \gamma \cdot 0] + 0[0 + \gamma \cdot 1] \\ &+ 0[0 + \gamma \cdot 0.5] + 1[1 + \gamma \cdot 0]\} \\ = \max\{&0.05, 1\} = 1 \end{aligned}$$

VALUE ITERATION: EXAMPLE

SECOND ITERATION

	s_1	s_2	s_3	s_4
V	$0 \rightarrow \mathbf{0.9}$	$1 \rightarrow \mathbf{1}$	0.5	0

$$V(s_3) = \max_{\substack{a_1,a_2,\\ a_3}} \left\{ \sum_{\substack{s'=s_1,\\ s_2,s_3,s_4}} p(s'|s_3,a)[r+\gamma V(s')] \right\}$$



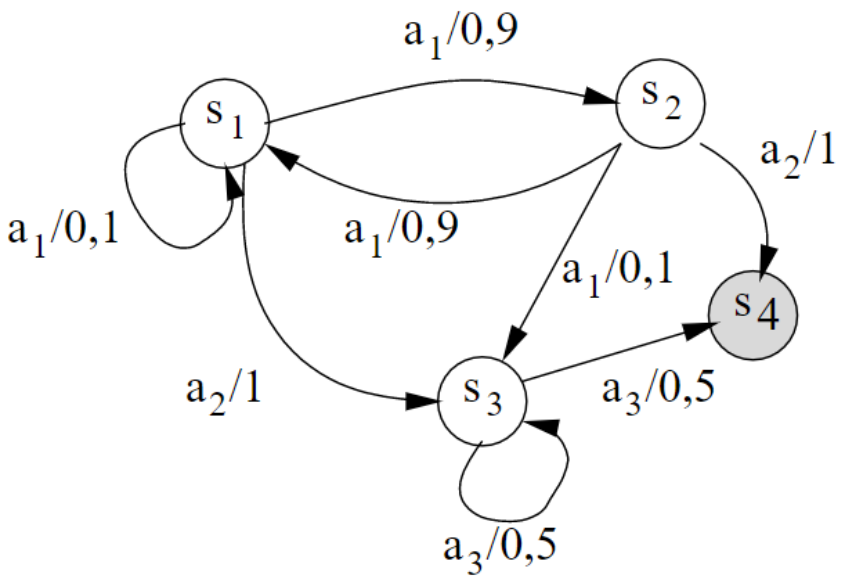
$$V(s_3) = \max\{p(s_1|s_3,a_3)[r+\gamma V(s_1)] + p(s_2|s_3,a_3)[r+\gamma V(s_2)] + p(s_3|s_3,a_3)[r+\gamma V(s_3)] + p(s_4|s_3,a_3)[r+\gamma V(s_4)]\}$$

VALUE ITERATION: EXAMPLE

SECOND ITERATION

	s_1	s_2	s_3	s_4
V	$0 \rightarrow \mathbf{0.9}$	$1 \rightarrow \mathbf{1}$	0.5	0

$$V(s_3) = \max_{\substack{a_1,a_2,\\ a_3}} \left\{ \sum_{\substack{s'=s_1,\\ s_2,s_3,s_4}} p(s'|s_3,a)[r + \gamma V(s')] \right\}$$



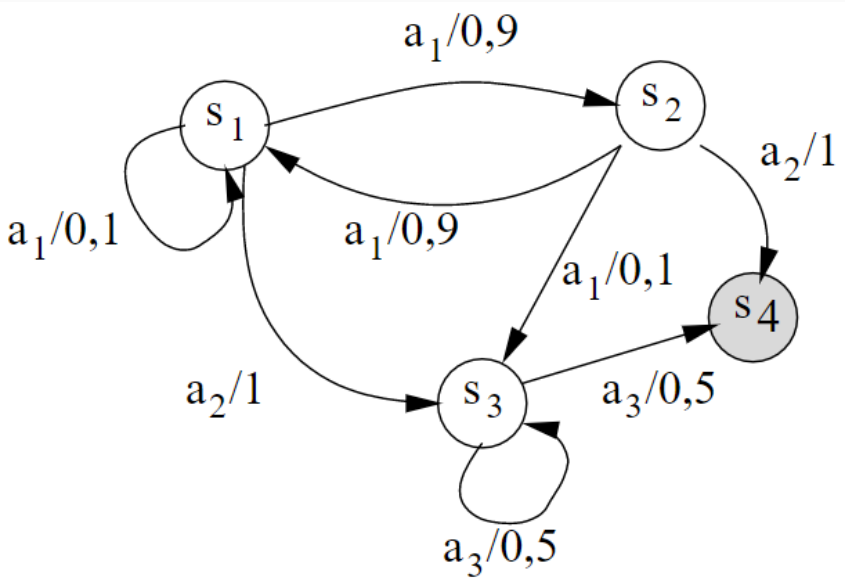
$$\begin{aligned} V(s_3) &= \max\{0[0 + \gamma \cdot 0] + 0[0 + \gamma \cdot 1] \\ &\quad + 0.5[0 + \gamma \cdot 0.5] + 0.5[1 + \gamma \cdot 0]\} \\ &= \max\{0.75\} = 0.75 \end{aligned}$$

VALUE ITERATION: EXAMPLE

SECOND ITERATION

	s_1	s_2	s_3	s_4
V	$0 \rightarrow \mathbf{0.9}$	$1 \rightarrow \mathbf{1}$	$0.5 \rightarrow \mathbf{0.75}$	0

$$V(s_4) = \max_{\substack{a_1,a_2,\\ a_3}} \left\{ \sum_{\substack{s'=s_1,\\ s_2,s_3,s_4}} p(s'|s_4,a)[r + \gamma V(s')] \right\}$$



$$V(s_4) = 0$$

$$\Delta = \max\{0.9 - 0, 1 - 1, 0.75 - 0.5, 0 - 0\} = 0.9 > \theta = 0.01$$

$\circlearrowleft$ And repeat until $\Delta \not> \theta = 0.01$.

Model-free methods

MODEL-FREE METHODS

What do we do when we do not have information about the transition system of the environment, i.e., $\mathcal{T}$ or p?.

- We need to **generate experience** in the environment.
- This experience is collected in the form of **trajectories**:

$$\langle s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_{n-1}, a_{n-1}, r_{n-1}, s_n \rangle$$

- With those trajectories, we can **estimate** (not calculate) an optimal policy.
- The agent will simply execute actions and receive a reward value, and with that it will refine its way of acting.
- Two ways to estimate: Monte-Carlo and Temporal Difference.

MONTE-CARLO METHODS

An agent can estimate π_* using different trajectories produced by π in **complete episodes**. Example: Monte-Carlo with exploring starts.

- ◇ Using the definition of $q(s, a)$ (expected cummul. reward).
- ◇ Converges because you choose any pair of state-action.

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
  $Q(s, a) \leftarrow$ arbitrary
  $\pi(s) \leftarrow$ arbitrary
  $Returns(s, a) \leftarrow$ empty list

Repeat forever:
  Choose $S_0 \in \mathcal{S}$ and $A_0 \in \mathcal{A}(S_0)$ s.t. all pairs have probability > 0
  Generate an episode starting from S_0, A_0, following π
  For each pair s, a appearing in the episode:
    $G \leftarrow$ return following the first occurrence of s, a
    Append G to $Returns(s, a)$
    $Q(s, a) \leftarrow$ average$(Returns(s, a))$
  For each s in the episode:
    $\pi(s) \leftarrow \arg\max_a Q(s, a)$

TEMPORAL DIFFERENCE METHODS

An agent can estimate π_* using **online information** (steps inside an episode). Example: Q-learning.

- This method takes advantage of the recursion of $q(s, a)$.
- Values are updated at each step using a simple difference:

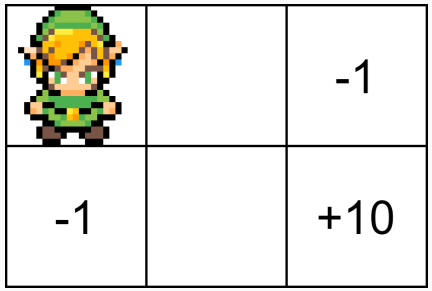
$$NewEstimate \leftarrow OldEstimate + StepSize\,[Target - OldEstimate]$$

- The target here is the optimal Q-value.

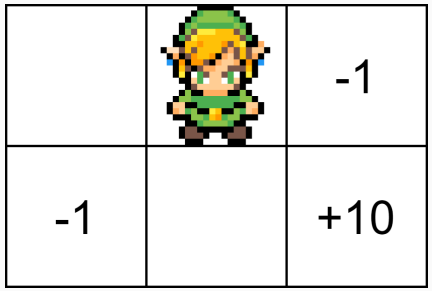
Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(terminal\text{-}state, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    Choose A from S using policy derived from Q (e.g., ϵ-greedy)
    Take action A, observe R, S'
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
    $S \leftarrow S'$;
  until S is terminal

Q-LEARNING EXAMPLE

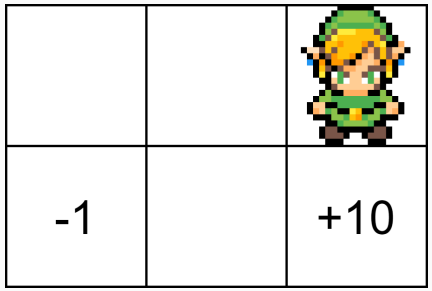
A



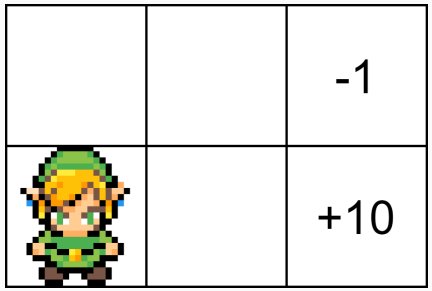
B



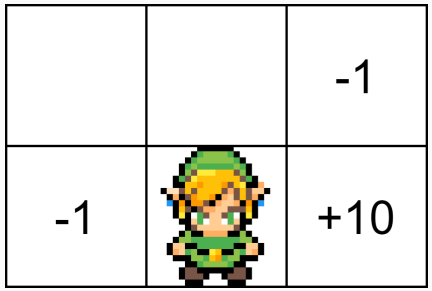
C



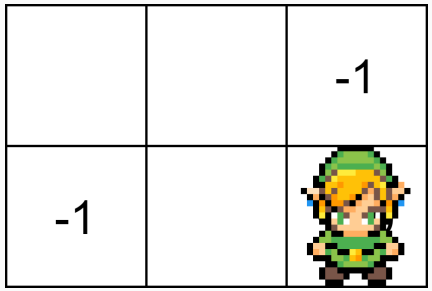
D



E

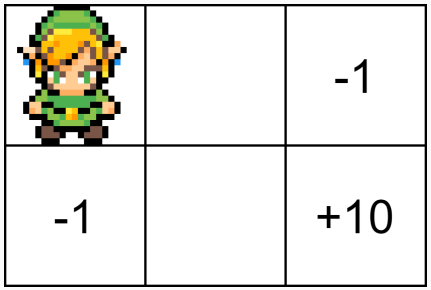


F



Q-LEARNING EXAMPLE

Timestep: $t = 0$



Q(s,a)	LE	RI	UP	DO
A	0	0	0	0
B	0	0	0	0
C	0	0	0	0
D	0	0	0	0
E	0	0	0	0
F	0	0	0	0

(S,A,R,S') = (A,RI,0,B)

$$Q(A, RI) = Q(A, RI) + \alpha(R + \gamma \max Q(B, \cdot) - Q(A, RI))$$

Q-LEARNING EXAMPLE

Timestep: $t = 0$

		-1
-1		+10

Q(s,a)	LE	RI	UP	DO
A	0	**0**	0	0
B	0	0	0	0
C	0	0	0	0
D	0	0	0	0
E	0	0	0	0
F	0	0	0	0

(S,A,R,S') = (A,RI,0,B)

$$\boxed{Q(A, RI) = 0 + 0.2(0 + 1 \cdot 0 - 0) = 0}$$

Q-LEARNING EXAMPLE

Timestep: $t = 1$

		-1
-1		+10

Q(s,a)	LE	RI	UP	DO
A	0	0	0	0
B	0	0	0	0
C	0	0	0	0
D	0	0	0	0
E	0	0	0	0
F	0	0	0	0

(S,A,R,S') = (B,RI,-1,C)

$$Q(B, RI) = Q(B, RI) + \alpha(R + \gamma \max Q(C, \cdot) - Q(B, RI))$$

Q-LEARNING EXAMPLE

Timestep: $t = 1$

		
-1		+10

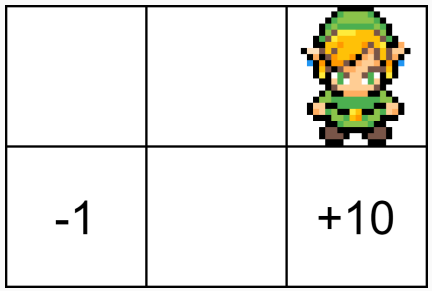
Q(s,a)	LE	RI	UP	DO
A	0	0	0	0
B	0	**-0.2**	0	0
C	0	0	0	0
D	0	0	0	0
E	0	0	0	0
F	0	0	0	0

(S,A,R,S') = (B,RI,-1,C)

$$Q(B, RI) = 0 + 0.2(-1 + 1 \cdot 0 - 0) = -0.2$$

Q-LEARNING EXAMPLE

Timestep: $t = 2$



Q(s,a)	LE	RI	UP	DO
A	0	0	0	0
B	0	-0.2	0	0
C	0	0	0	0
D	0	0	0	0
E	0	0	0	0
F	0	0	0	0

(S,A,R,S') = (C,DO,+10,F)

$$Q(C, DO) = Q(C, DO) + \alpha(R + \gamma \max Q(F, \cdot) - Q(C, DO))$$

Q-LEARNING EXAMPLE

Timestep: $t = 2$

		-1
-1		

Q(s,a)	LE	RI	UP	DO
A	0	0	0	0
B	0	-0.2	0	0
C	0	0	0	**2**
D	0	0	0	0
E	0	0	0	0
F	0	0	0	0

(S,A,R,S') = (C,DO,+10,F)

$$Q(C, DO) = 0 + 0.2(+10 + 1 \cdot 0 - 0) = 2$$

END OF EPISODE

Q-LEARNING EXAMPLE

Timestep: $t = 0$

		-1
-1		+10

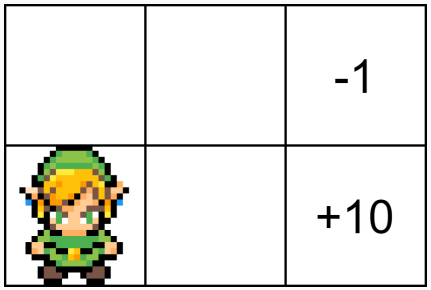
Q(s,a)	LE	RI	UP	DO
A	0	0	0	0
B	0	-0.2	0	0
C	0	0	0	2
D	0	0	0	0
E	0	0	0	0
F	0	0	0	0

(S,A,R,S') = (A,DO,-1,D)

$$Q(A, DO) = Q(A, DO) + \alpha(R + \gamma \max Q(D, \cdot) - Q(A, DO))$$

Q-LEARNING EXAMPLE

Timestep: $t = 0$



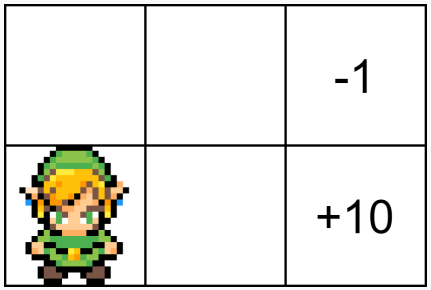
Q(s,a)	LE	RI	UP	DO
A	0	0	0	**-0.2**
B	0	-0.2	0	0
C	0	0	0	2
D	0	0	0	0
E	0	0	0	0
F	0	0	0	0

(S,A,R,S') = (A,DO,-1,D)

$$Q(A, DO) = 0 + 0.2(-1 + 1 \cdot 0 - 0) = -0.2$$

Q-LEARNING EXAMPLE

Timestep: $t = 1$



Q(s,a)	LE	RI	UP	DO
A	0	0	0	-0.2
B	0	-0.2	0	0
C	0	0	0	2
D	0	0	0	0
E	0	0	0	0
F	0	0	0	0

(S,A,R,S') = (D,RI,0,E)

$$Q(D, RI) = Q(D, RI) + \alpha(R + \gamma \max Q(E, \cdot) - Q(D, RI))$$

Q-LEARNING EXAMPLE

Timestep: $t = 1$

		-1
-1		+10

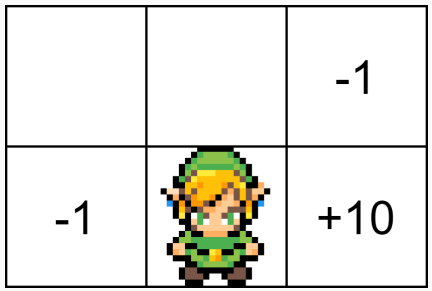
Q(s,a)	LE	RI	UP	DO
A	0	0	0	-0.2
B	0	-0.2	0	0
C	0	0	0	2
D	0	**0**	0	0
E	0	0	0	0
F	0	0	0	0

(S,A,R,S') = (D,RI,0,E)

$$\boxed{Q(D, RI) = 0 + 0.2(0 + 1 \cdot 0 - 0) = 0}$$

Q-LEARNING EXAMPLE

Timestep: $t = 2$



Q(s,a)	LE	RI	UP	DO
A	0	0	0	-0.2
B	0	-0.2	0	0
C	0	0	0	2
D	0	0	0	0
E	0	0	0	0
F	0	0	0	0

(S,A,R,S') = (E,RI,+10,F)

$$Q(E, RI) = Q(E, RI) + \alpha(R + \gamma \max Q(F, \cdot) - Q(E, RI))$$

Q-LEARNING EXAMPLE

Timestep: $t = 2$

		-1
-1		

Q(s,a)	LE	RI	UP	DO
A	0	0	0	-0.2
B	0	-0.2	0	0
C	0	0	0	2
D	0	0	0	0
E	0	**2**	0	0
F	0	0	0	0

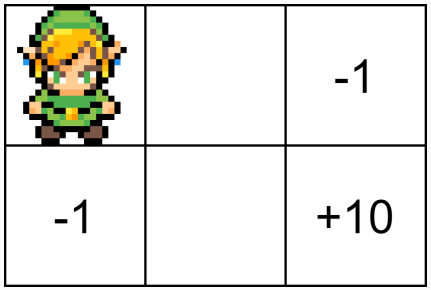
(S,A,R,S') = (E,RI,+10,F)

$$\boxed{Q(E, RI) = 0 + 0.2(10 + 1 \cdot 0 - 0) = 2}$$

END OF EPISODE

Q-LEARNING EXAMPLE

Timestep: $t = 0$



Q(s,a)	LE	RI	UP	DO
A	0	0	0	-0.2
B	0	-0.2	0	0
C	0	0	0	2
D	0	0	0	0
E	0	2	0	0
F	0	0	0	0

(S,A,R,S') = (A,RI,0,B)

$$Q(A, RI) = Q(A, RI) + \alpha(R + \gamma \max Q(B, \cdot) - Q(A, RI))$$

Q-LEARNING EXAMPLE

Timestep: $t = 0$

		-1
-1		+10

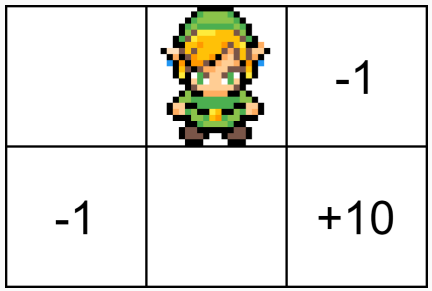
Q(s,a)	LE	RI	UP	DO
A	0	**0**	0	-0.2
B	0	-0.2	0	0
C	0	0	0	2
D	0	0	0	0
E	0	2	0	0
F	0	0	0	0

(S,A,R,S') = (A,RI,0,B)

$$\boxed{Q(A, RI) = 0 + 0.2(0 + 1 \cdot 0 - 0) = 0}$$

Q-LEARNING EXAMPLE

Timestep: $t = 1$



Q(s,a)	LE	RI	UP	DO
A	0	0	0	-0.2
B	0	-0.2	0	0
C	0	0	0	2
D	0	0	0	0
E	0	2	0	0
F	0	0	0	0

(S,A,R,S') = (B,DO,0,E)

$$Q(B, DO) = Q(B, DO) + \alpha(R + \gamma \max Q(E, \cdot) - Q(B, DO))$$

Q-LEARNING EXAMPLE

Timestep: $t = 1$

		-1
-1		+10

Q(s,a)	LE	RI	UP	DO
A	0	0	0	-0.2
B	0	-0.2	0	**0.4**
C	0	0	0	2
D	0	0	0	0
E	0	2	0	0
F	0	0	0	0

(S,A,R,S') = (B,DO,0,E)

$$Q(B, DO) = 0 + 0.2(0 + 1 \cdot 2 - 0) = 0.4$$

Q-LEARNING EXAMPLE

Timestep: $t = 2$

		-1
-1		+10

Q(s,a)	LE	RI	UP	DO
A	0	0	0	-0.2
B	0	-0.2	0	0.4
C	0	0	0	2
D	0	0	0	0
E	0	2	0	0
F	0	0	0	0

(S,A,R,S') = (E,RI,+10,F)

$$Q(E, RI) = Q(E, RI) + \alpha(R + \gamma \max Q(F, \cdot) - Q(E, RI))$$

Q-LEARNING EXAMPLE

Timestep: $t = 2$

		-1
-1		

Q(s,a)	LE	RI	UP	DO
A	0	0	0	-0.2
B	0	-0.2	0	0.4
C	0	0	0	2
D	0	0	0	0
E	0	**3.6**	0	0
F	0	0	0	0

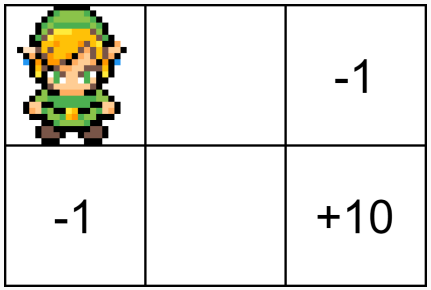
(S,A,R,S') = (E,RI,+10,F)

$$\boxed{Q(E, RI) = 2 + 0.2(10 + 1 \cdot 0 - 2)} = 3.6$$

END OF EPISODE

Q-LEARNING EXAMPLE

Timestep: $t = 0$



Q(s,a)	LE	RI	UP	DO
A	0	0	0	-0.2
B	0	-0.2	0	0.4
C	0	0	0	2
D	0	0	0	0
E	0	3.6	0	0
F	0	0	0	0

(S,A,R,S') = (A,RI,0,B)

$$Q(A, RI) = Q(A, RI) + \alpha(R + \gamma \max Q(B, \cdot) - Q(A, RI))$$

Q-LEARNING EXAMPLE

Timestep: $t = 0$

		-1
-1		+10

Q(s,a)	LE	RI	UP	DO
A	0	**0.08**	0	-0.2
B	0	-0.2	0	0.4
C	0	0	0	2
D	0	0	0	0
E	0	3.6	0	0
F	0	0	0	0

(S,A,R,S') = (A,RI,0,B)

$$\boxed{Q(A, RI) = 0 + 0.2(0 + 1 \cdot 0.4 - 0) = 0.08}$$

Q-LEARNING EXAMPLE

Timestep: $t = 1$

		-1
-1		+10

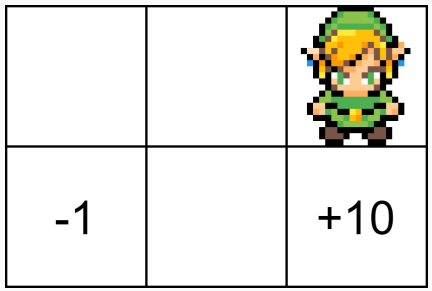
Q(s,a)	LE	RI	UP	DO
A	0	0.08	0	-0.2
B	0	-0.2	0	0.4
C	0	0	0	2
D	0	0	0	0
E	0	3.6	0	0
F	0	0	0	0

(S,A,R,S') = (B,RI,-1,C)

$$Q(B, RI) = Q(B, RI) + \alpha(R + \gamma \max Q(C, \cdot) - Q(B, RI))$$

Q-LEARNING EXAMPLE

Timestep: $t = 1$



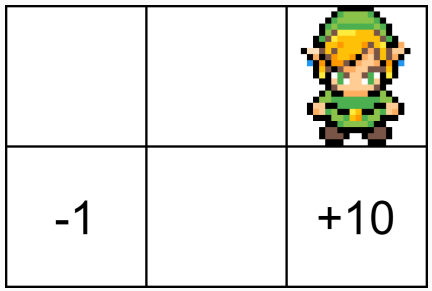
Q(s,a)	LE	RI	UP	DO
A	0	0.08	0	-0.2
B	0	**0.04**	0	0.4
C	0	0	0	2
D	0	0	0	0
E	0	3.6	0	0
F	0	0	0	0

(S,A,R,S') = (B,RI,-1,C)

$$Q(B, RI) = -0.2 + 0.2(-1 + 1 \cdot 2 - -0.2) = 0.04$$

Q-LEARNING EXAMPLE

Timestep: $t = 2$



Q(s,a)	LE	RI	UP	DO
A	0	0.08	0	-0.2
B	0	0.04	0	0.4
C	0	0	0	2
D	0	0	0	0
E	0	3.6	0	0
F	0	0	0	0

(S,A,R,S') = (C,DO,+10,F)

$$Q(C, DO) = Q(C, DO) + \alpha(R + \gamma \max Q(F, \cdot) - Q(C, DO))$$

Q-LEARNING EXAMPLE

Timestep: $t = 2$

		-1
-1		

Q(s,a)	LE	RI	UP	DO
A	0	0.8	0	-0.2
B	0	0.04	0	0.4
C	0	0	0	**3.6**
D	0	0	0	0
E	0	3.6	0	0
F	0	0	0	0

(S,A,R,S') = (C,DO,+10,F)

$$Q(C, DO) = 2 + 0.2(+10 + 1 \cdot 0 - 2) = 3.6$$

END OF EPISODE

Exploration vs Exploitation

When training with RL algorithms, it is necessary to establish a trade-off between **exploration and exploitation**.

- We want to **explore** new and currently worse paths so that we discover better solutions.
- We want to **exploit** current best paths to refine the current discovered solution.

The dilemma of when to exploit and when to explore in RL is a problem that has not yet been solved in general. The purpose is to get rid of local maximums.

Examples of E/E algorithms

The most used algorithms are ε-**greedy** algorithms, $\varepsilon \in [0, 1]$.

- ◇ Choose a random action with probability ε (exploration).
- ◇ Choose the best current action with probability $1 - \varepsilon$ (exploitation).
- ◇ Tune ε as needed: ε close to 1 acts more as a random policy, and close to 0 as a greedy policy.

Other algorithms: Optimistic initialization, ε-greedy with decay, etc.