# Reinforcement Learning

## A gentle introduction

**Đorđe Božić**
PhD Student
University of Bath

✉️ **djordjebbozic@gmail.com**

# Who am I?

- PhD student at Bath Reinforcement Learning Laboratory under professor Özgür Şimşek
- **Research interests**: transfer learning in reinforcement learning, continual learning, hierarchical reinforcement learning, intrinsically motivation

**Đorđe Božić**
PhD Student
University of Bath

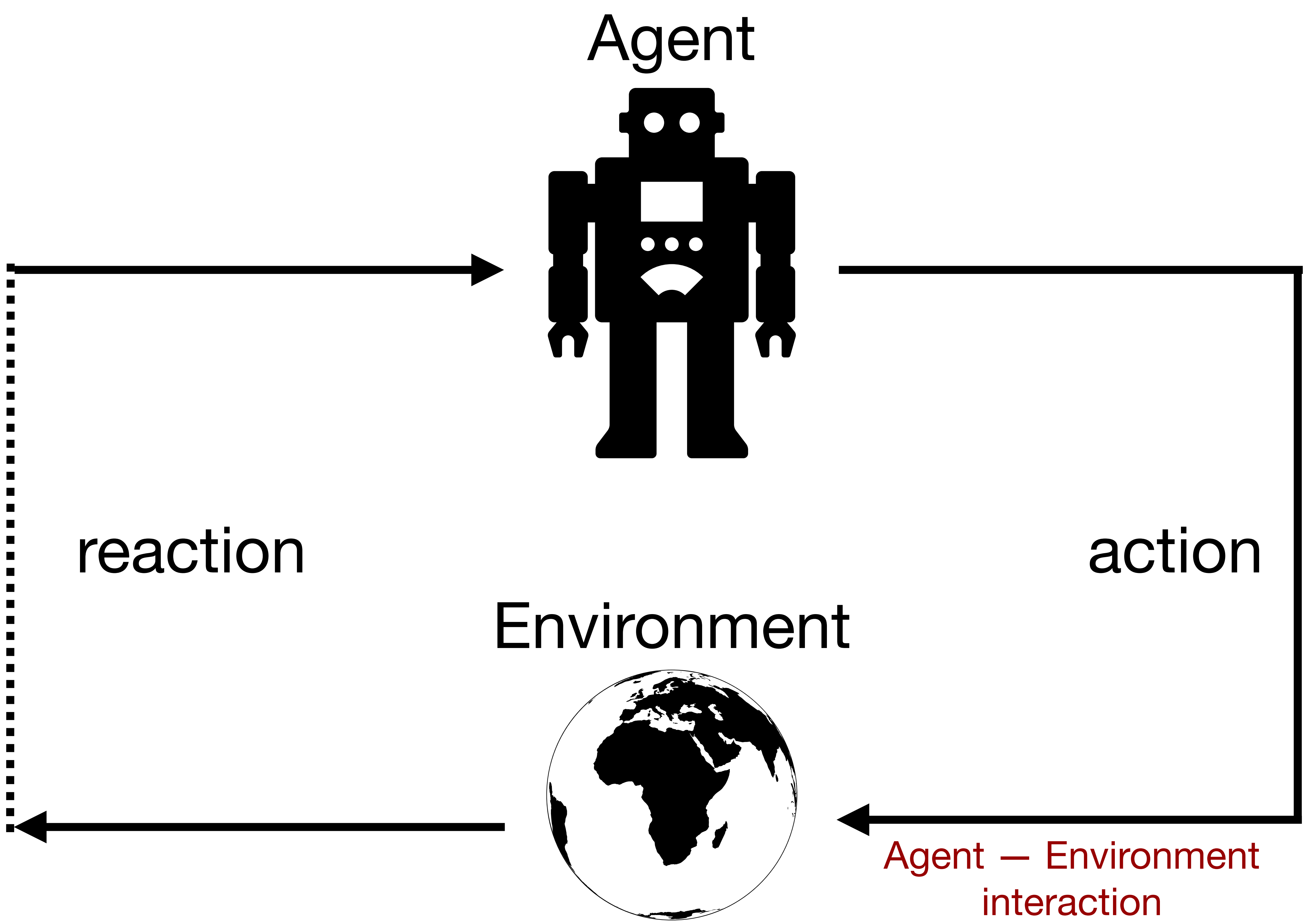
djordjebbozic@gmail.com

UNIVERSITY OF BATH

# Resources

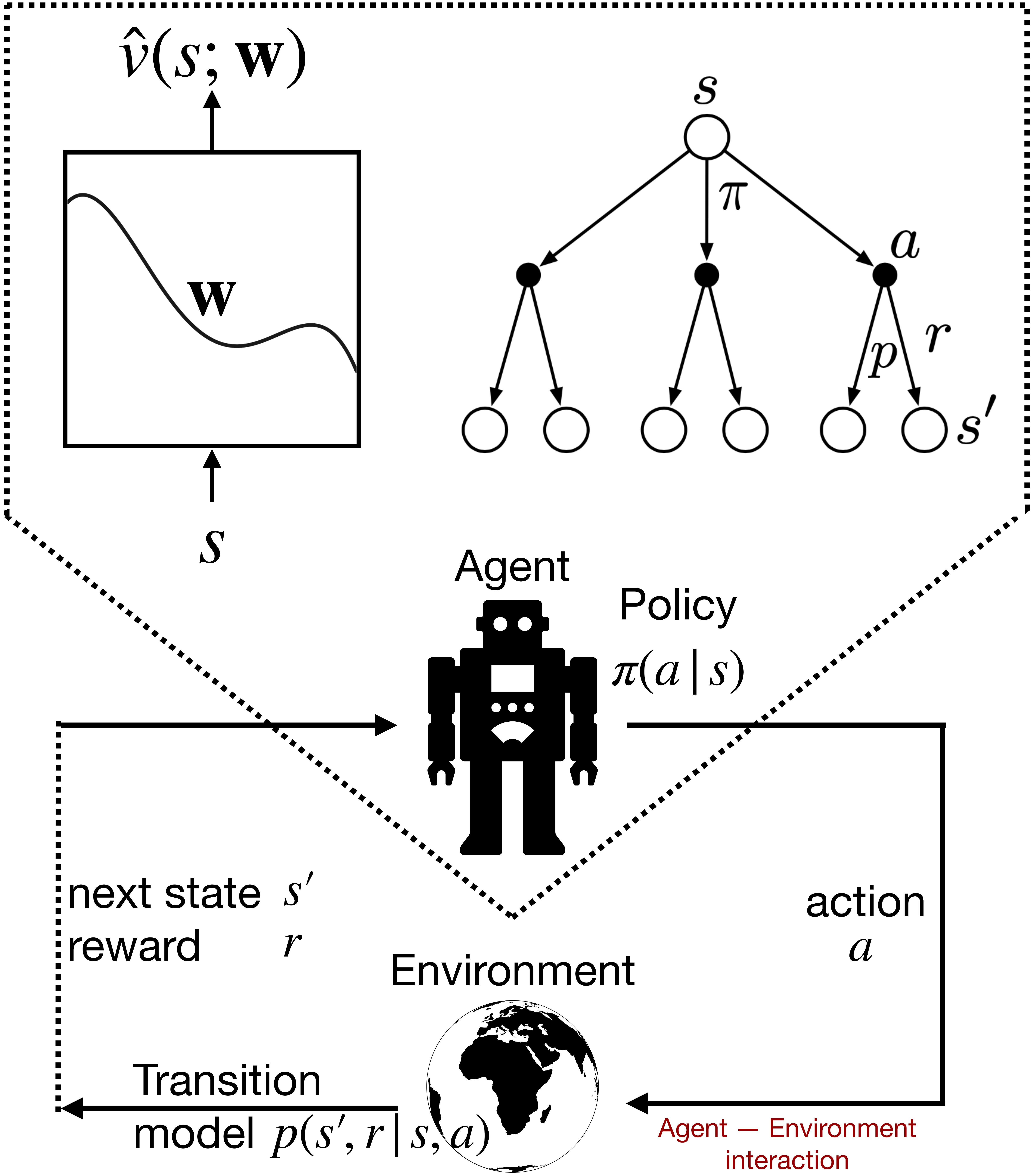
- [Sutton & Barto 2018]: Sutton, Richard S. and Barto, Andrew G.. Reinforcement Learning: An Introduction. Second : The MIT Press, 2018 .
- [Russel & Norvig 2010]: Russell, S. & Norvig, P. (2010), Artificial Intelligence: A Modern Approach , Prentice Hall.

# Introduction

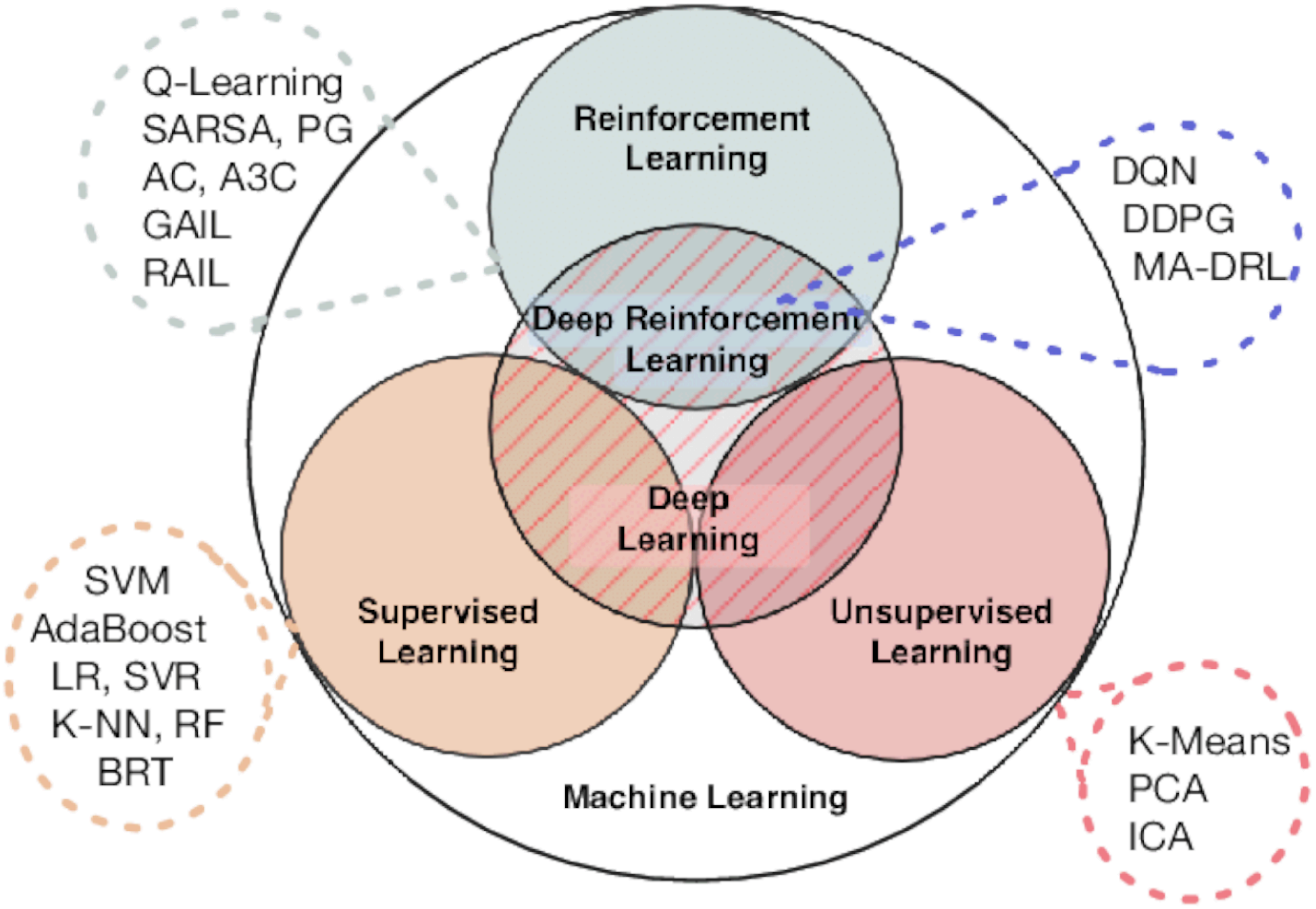



Agent — Environment interaction


Tunnelling strategy discovered by DQN on Breakout Atari environment




Agent — Environment interaction



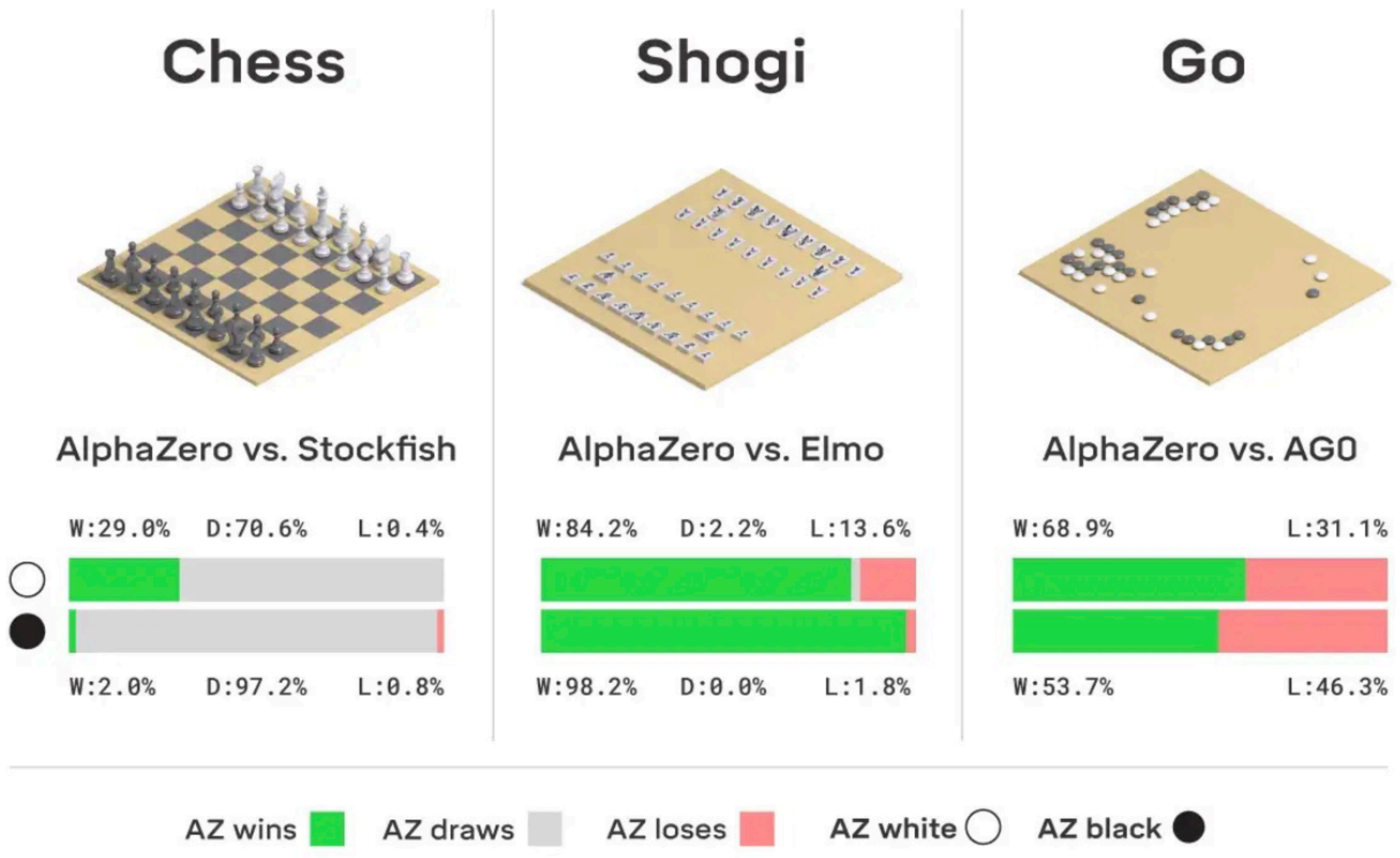

Taxonomy of contemporary ML [Yan et al. 2019]

# Introduction


Alpha Go, Silver et al. 2016

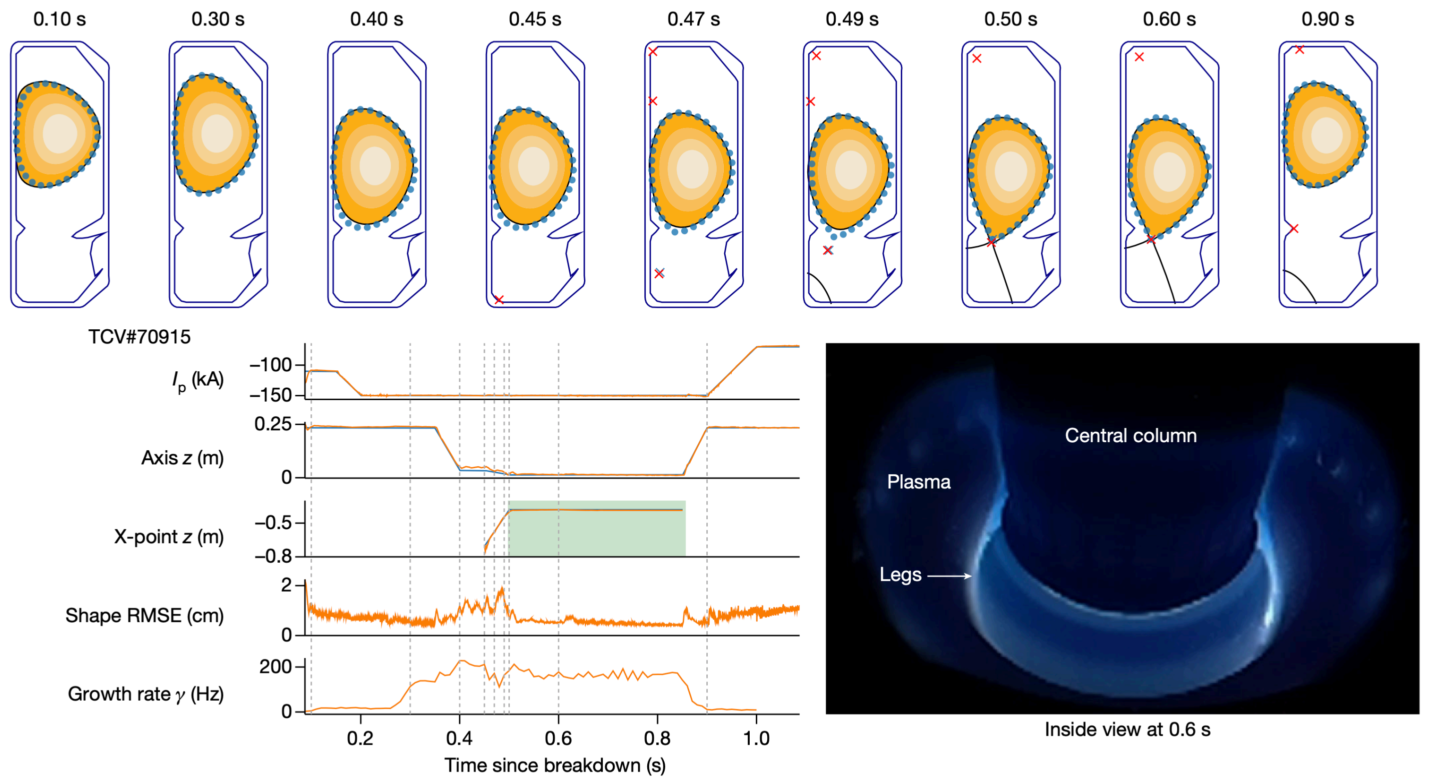
Alpha Star, Vinyals et al. 2019

Stratospheric Balloon Navigation, Bellemare et al, 2020




Alpha Zero, Silver et al. 2017


OpenAI Five, Barner et al. 2019




Magnetic Control of Tokamak Plasmas, Degrave et al, 2022

# Outline

- Introduction
- Reinforcement Learning Formalisation
- Model-Free Reinforcement Learning
- Value Function Approximation
- Policy Gradient Methods

"Deep RL"

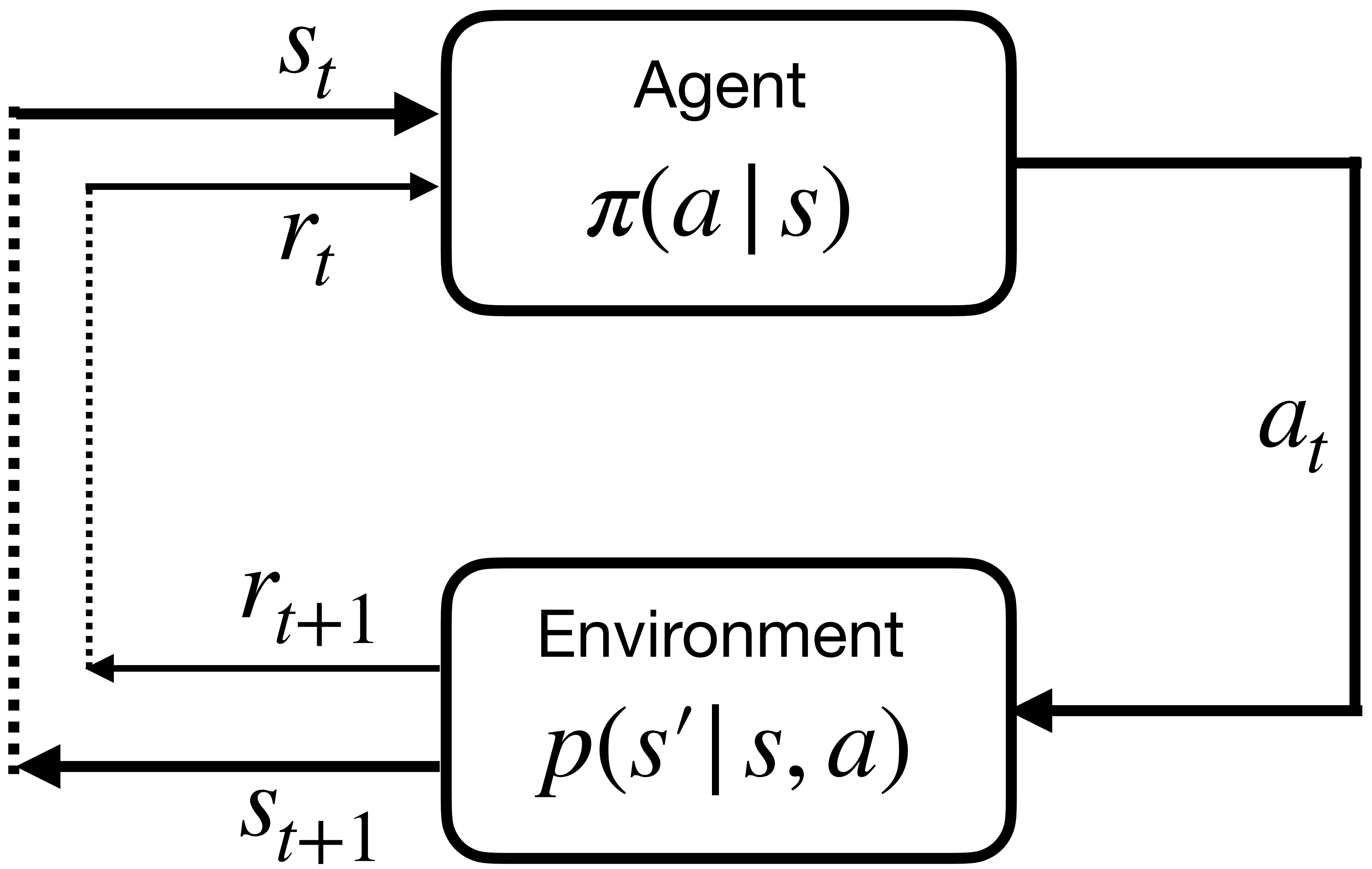
# Outline

- Introduction
- **Reinforcement Learning Formalisation**
  - **Agents and Environments**
  - **Markov Decision Process**
  - **Reward and Return**
  - **Policy**
  - **State-Value Function**
  - **Action-Value Function**
  - **Optimal Policy**
  - **Optimal Value Functions**
  - **Bellman Equations and Planning**
  - **Generalised Policy Iteration (GPI)**
  - **Exploration-Exploitation Trade-Off**
- …

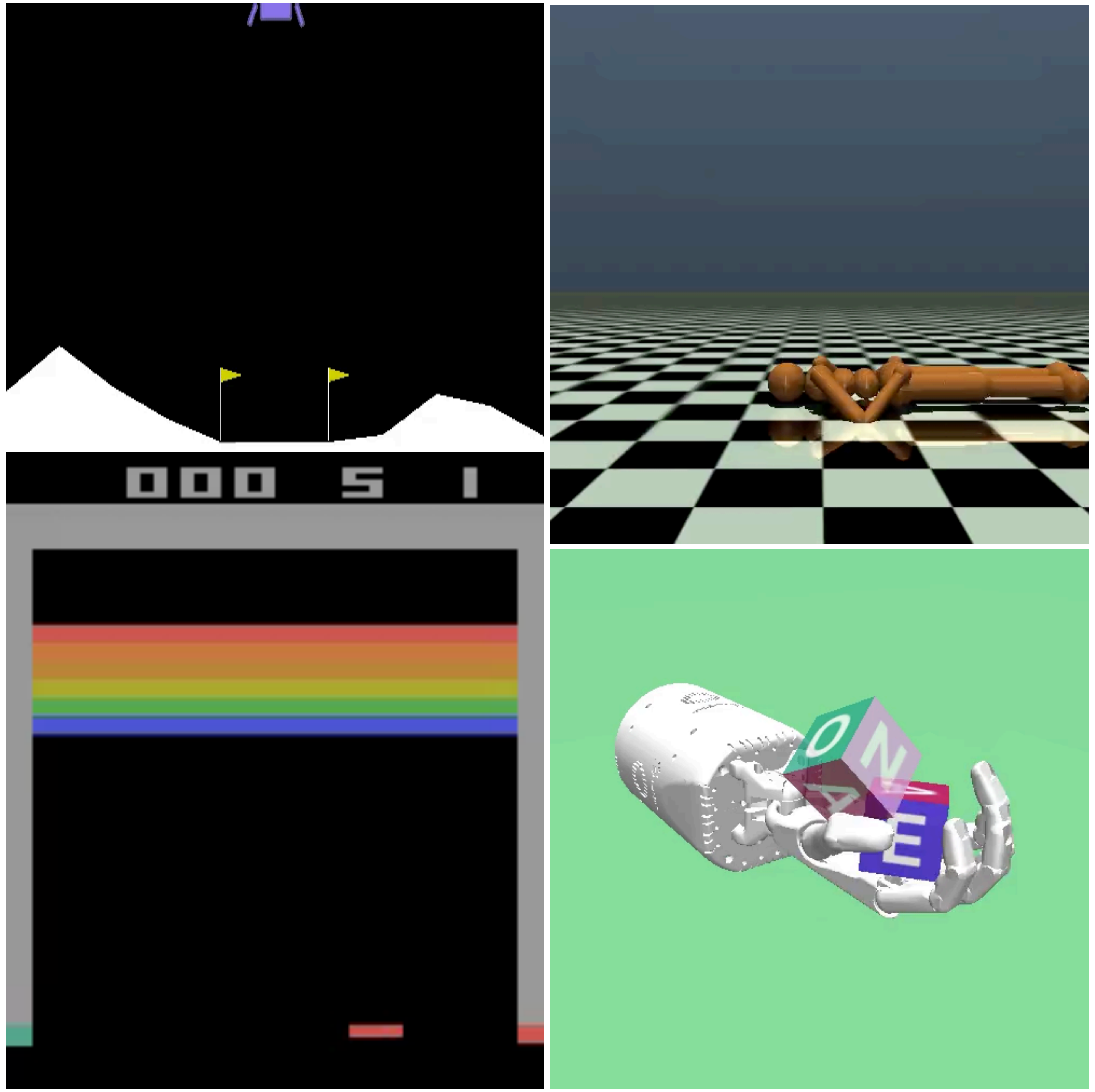
# RL Formalisation

## Agents and Environments

- Agent interacts with the environment
- Rewards given as feedback
- Different kind of supervision:
  - Samples not I.I.D
  - Learning signal may be delayed




Agent-Environment Interaction [Sutton & Barto 2018]

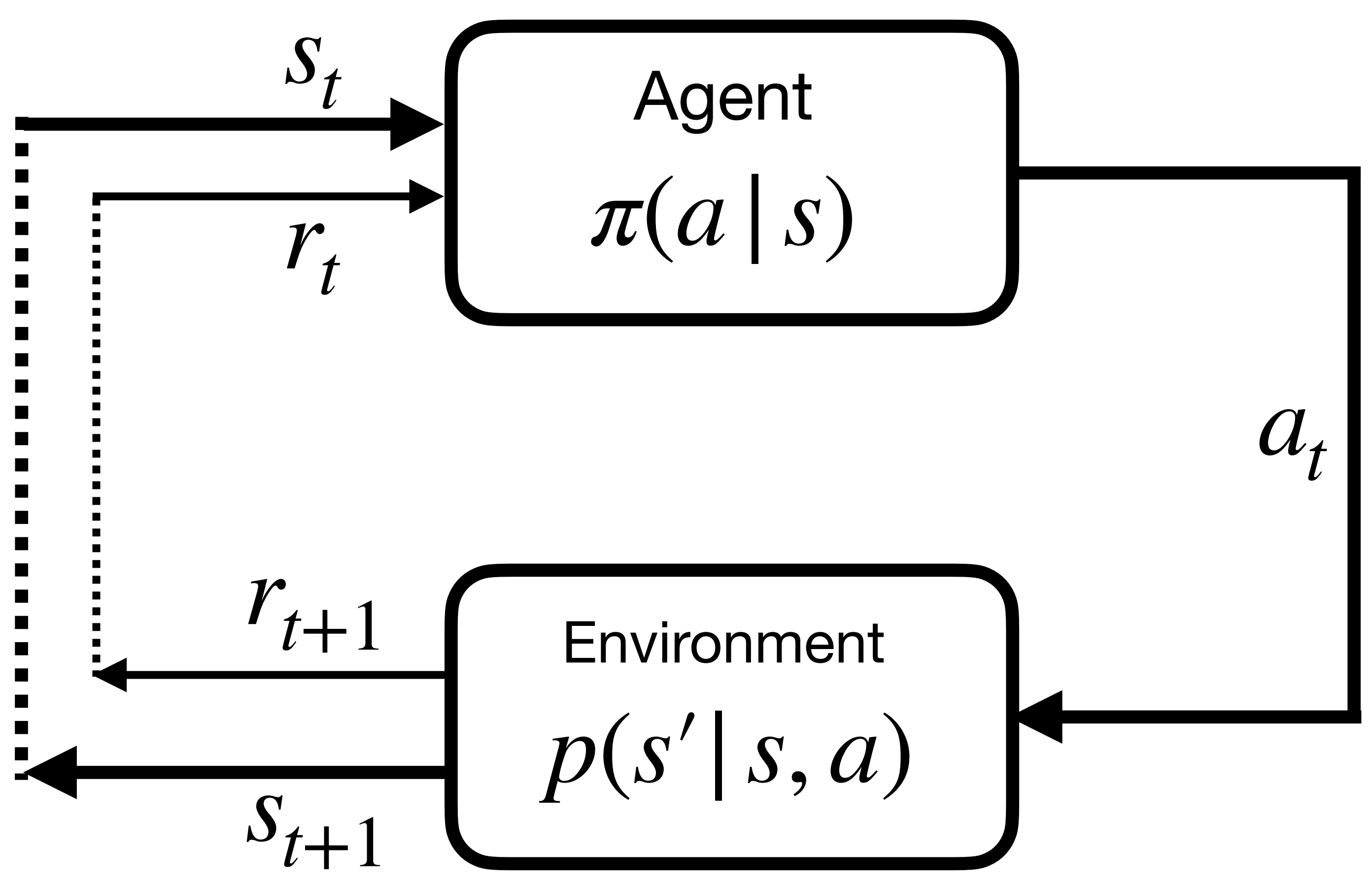


Lunar Lander, Breakout, MuJoCo Humanoid, Hand [Open AI Gym Environment suite]

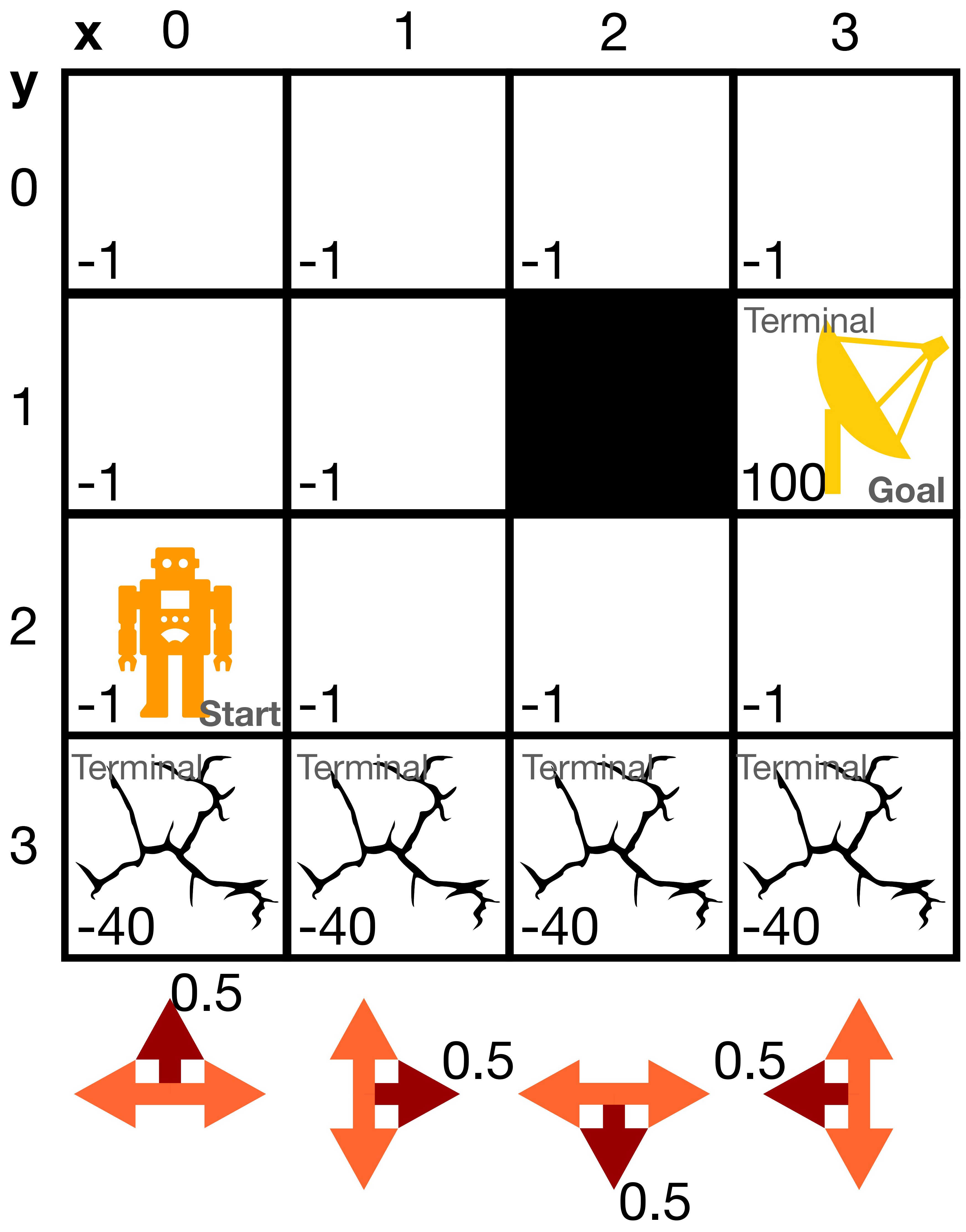
# RL Formalisation

## Agents and Environments

- Environment characteristics?
  - Episodic (finite-horizon) vs. Continuing (indefinite-horizon)
  - Deterministic vs. Stochastic
  - Fully vs. Partially observable
  - Discrete vs. Continuous



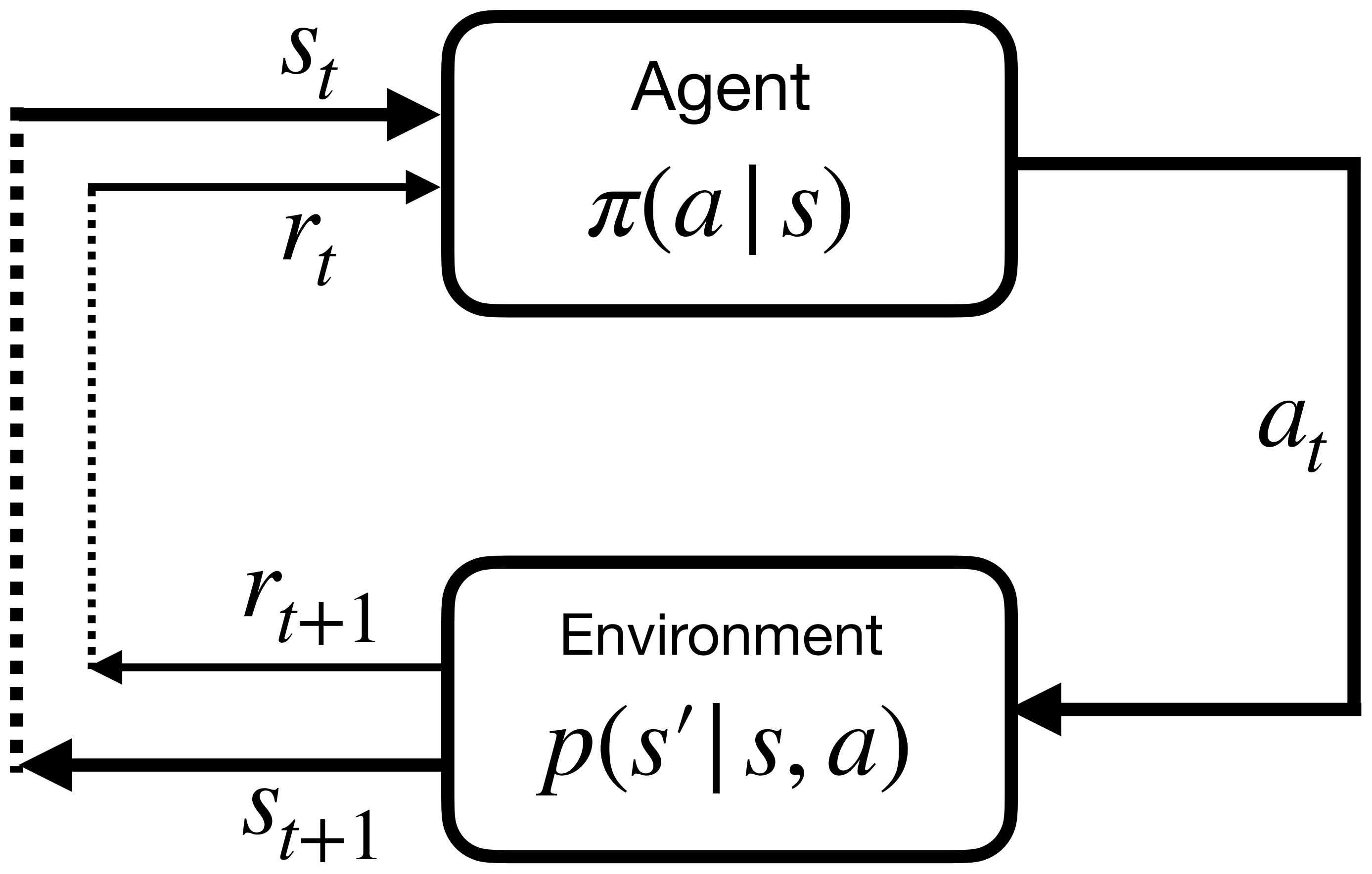
Agent-Environment Interaction [Sutton & Barto 2018]



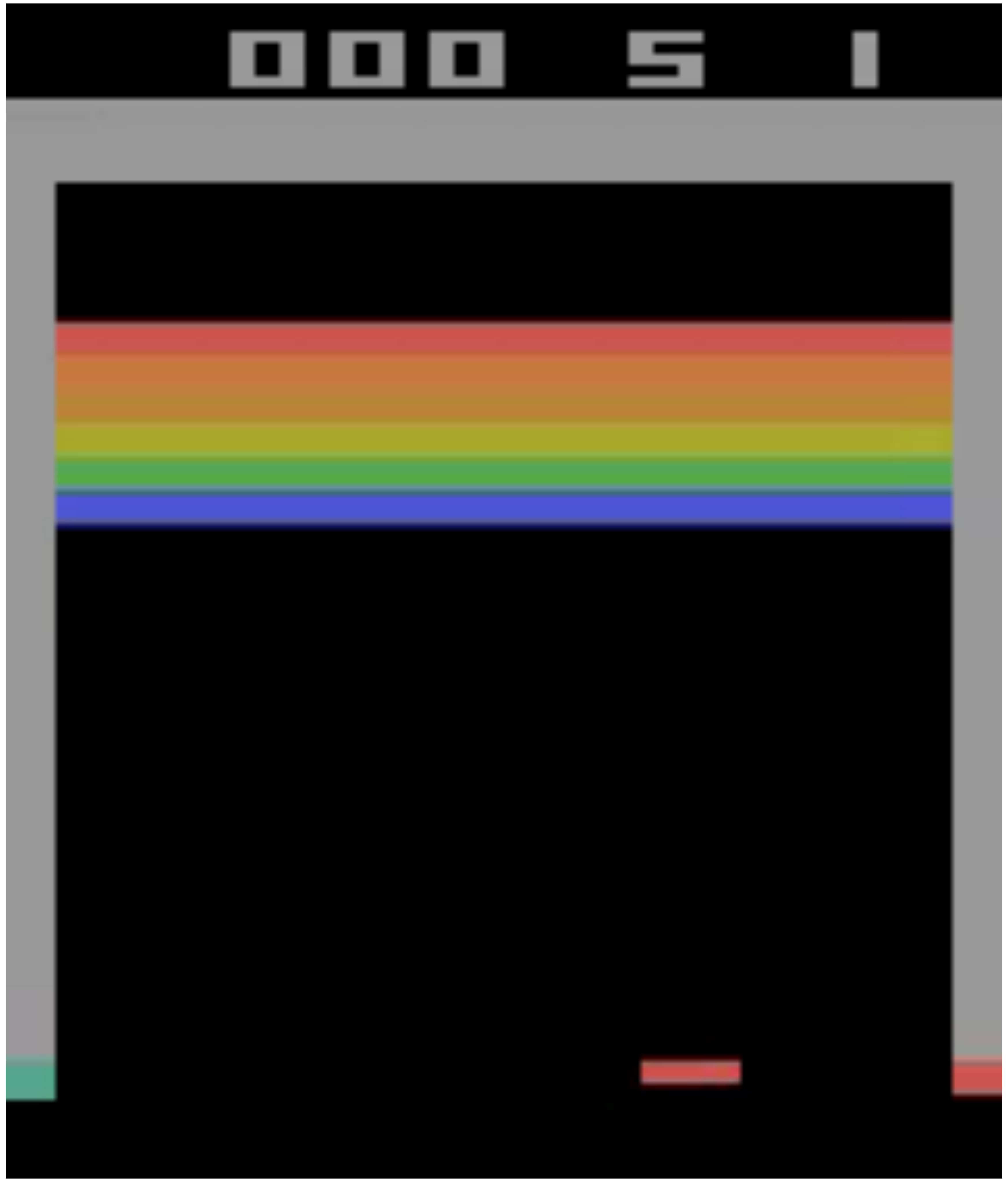
# RL Formalisation

## Agents and Environments

- Environment characteristics?
  - Episodic (finite-horizon) vs. Continuing (indefinite-horizon)
  - Deterministic vs. Stochastic
  - Fully vs. Partially observable
  - Discrete vs. Continuous



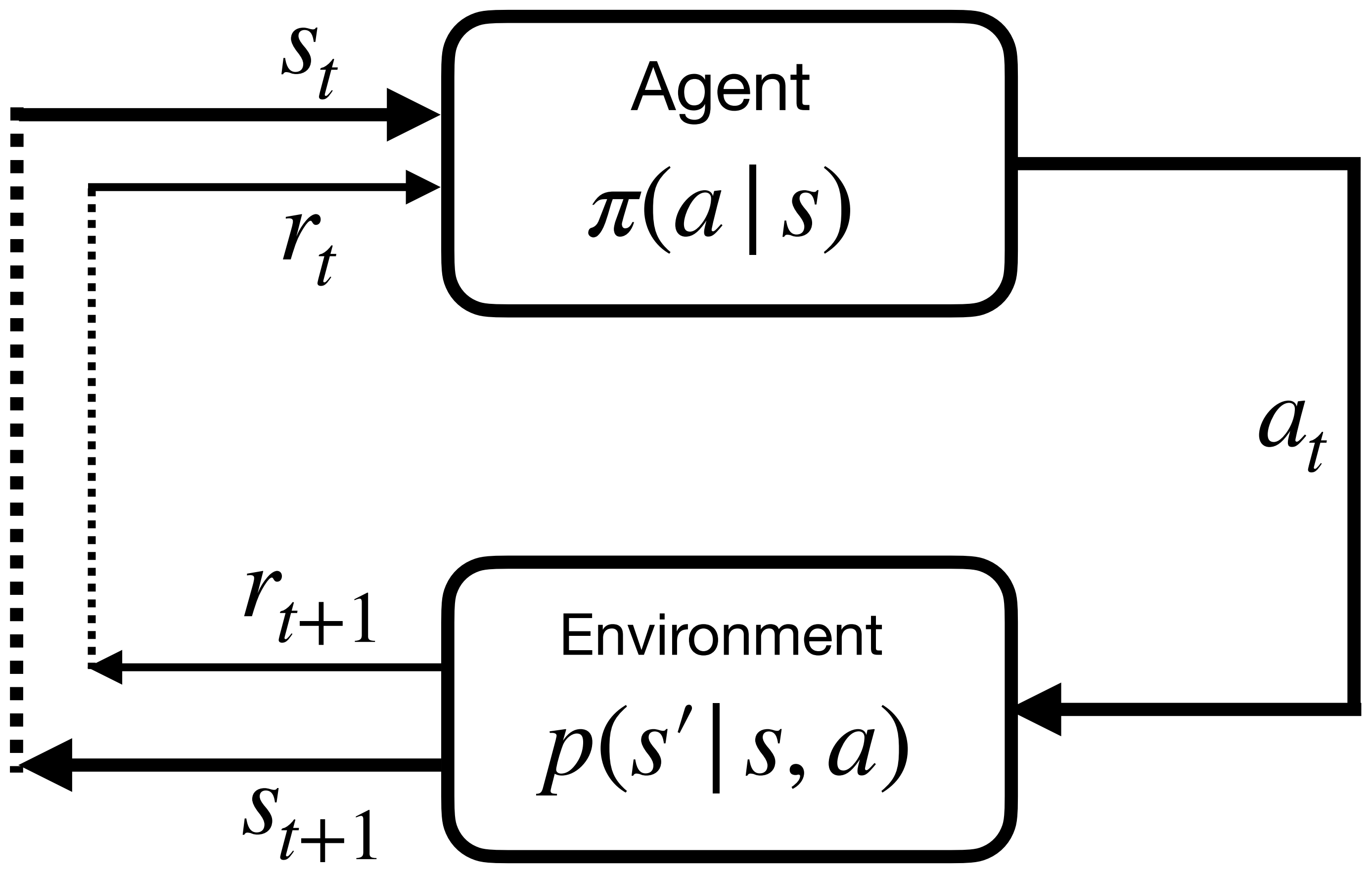

Agent-Environment Interaction [Sutton & Barto 2018]



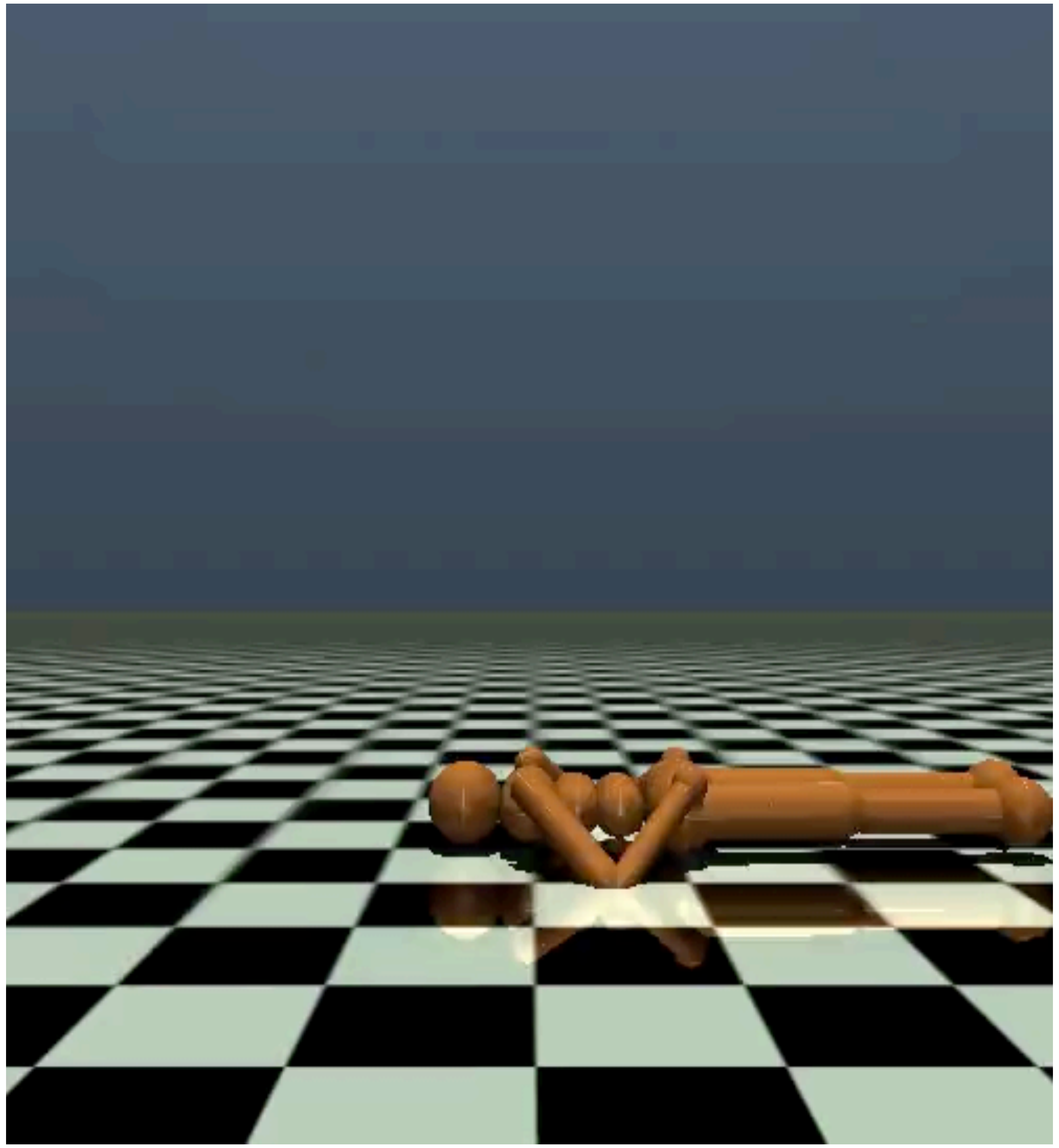
# RL Formalisation

## Agents and Environments

- Environment characteristics?
  - Episodic (finite-horizon) vs. Continuing (indefinite-horizon)
  - Deterministic vs. Stochastic
  - Fully vs. Partially observable
  - Discrete vs. Continuous



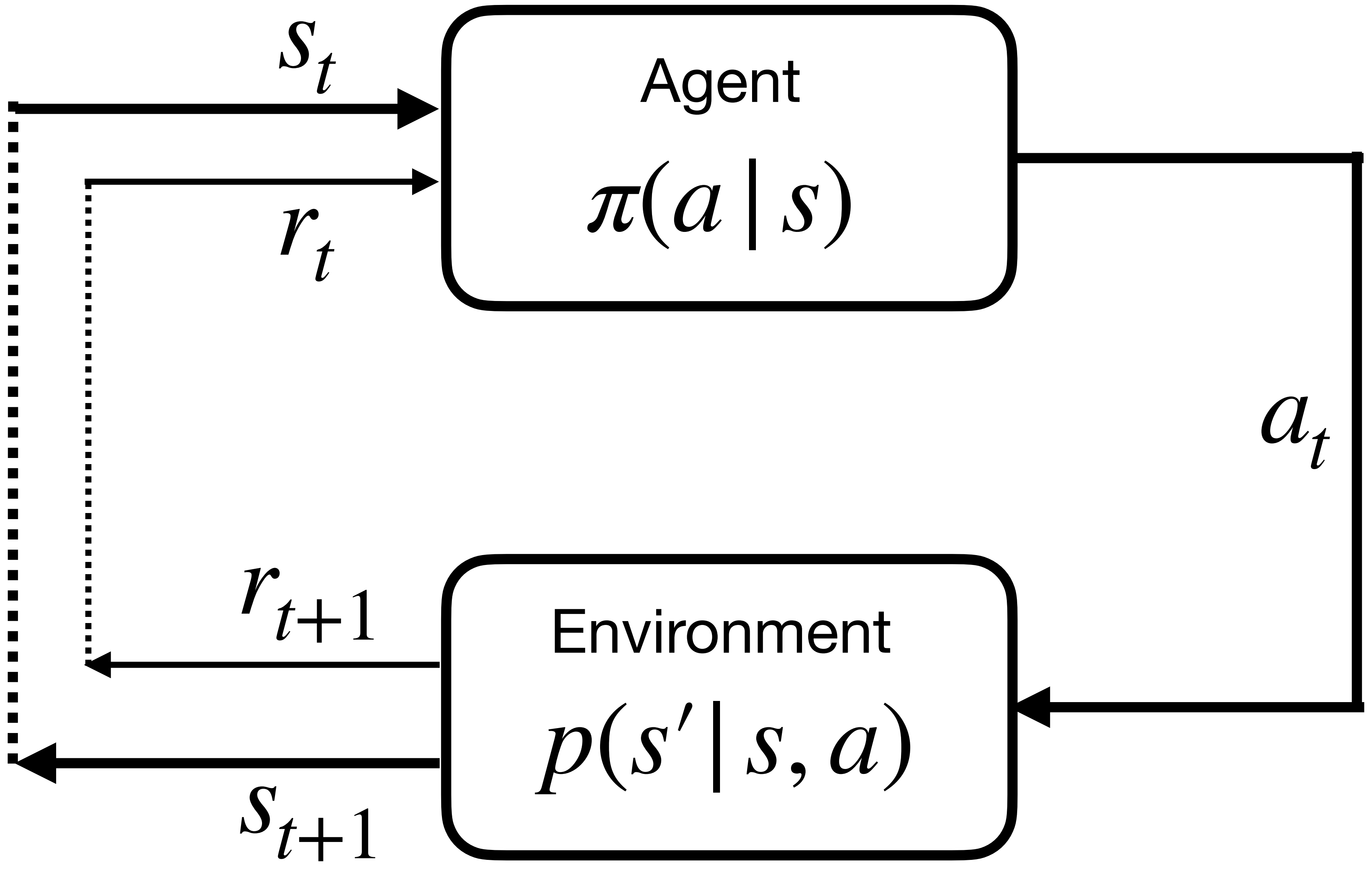

Agent-Environment Interaction [Sutton & Barto 2018]



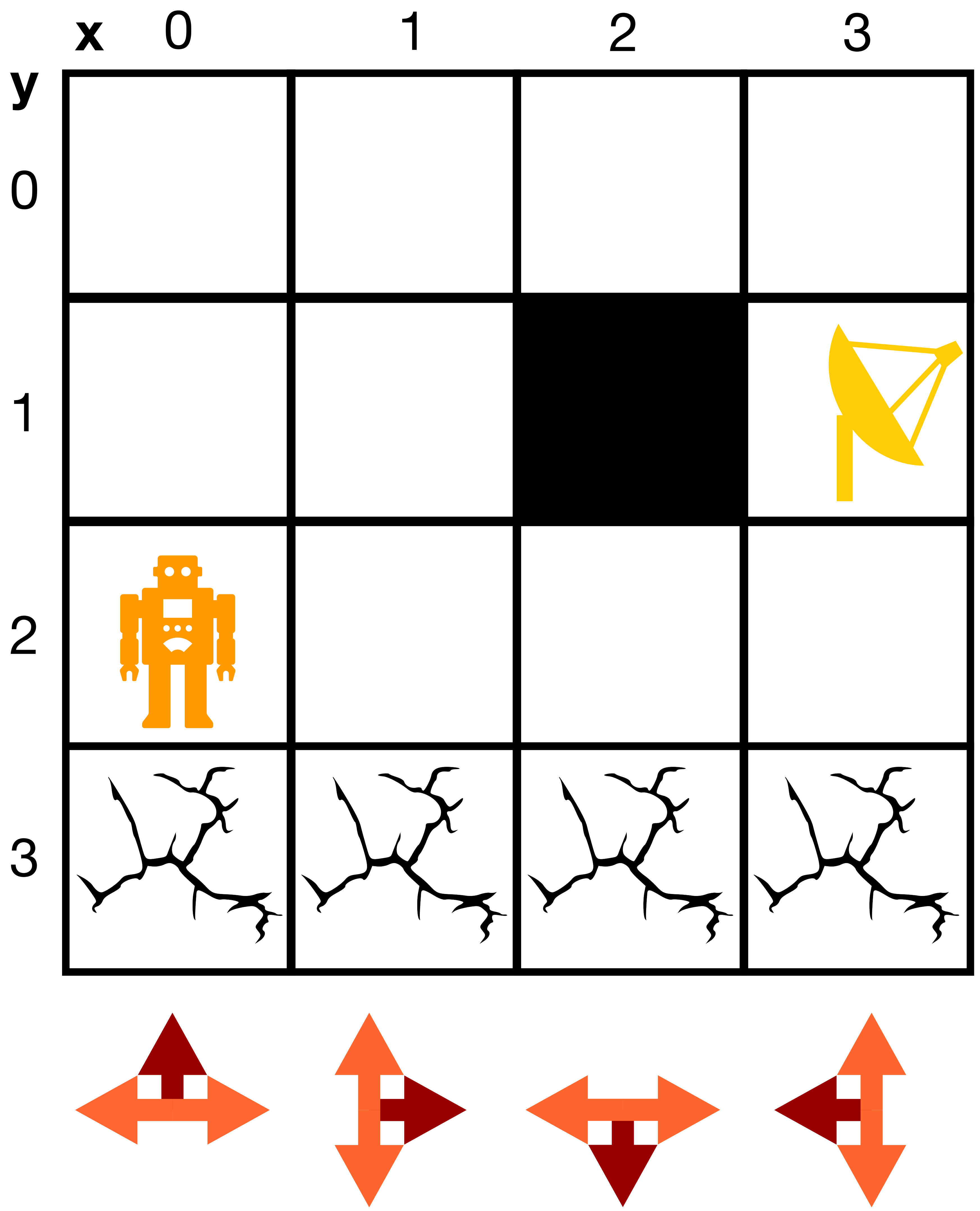
# RL Formalisation

## Agents and Environments

- Environment characteristics?
  - Stochastic, multiple terminal states
- How to find the optimal strategy?
  - Search algorithms (depth-first, A*, …) not applicable




Agent-Environment Interaction [Sutton & Barto 2018]

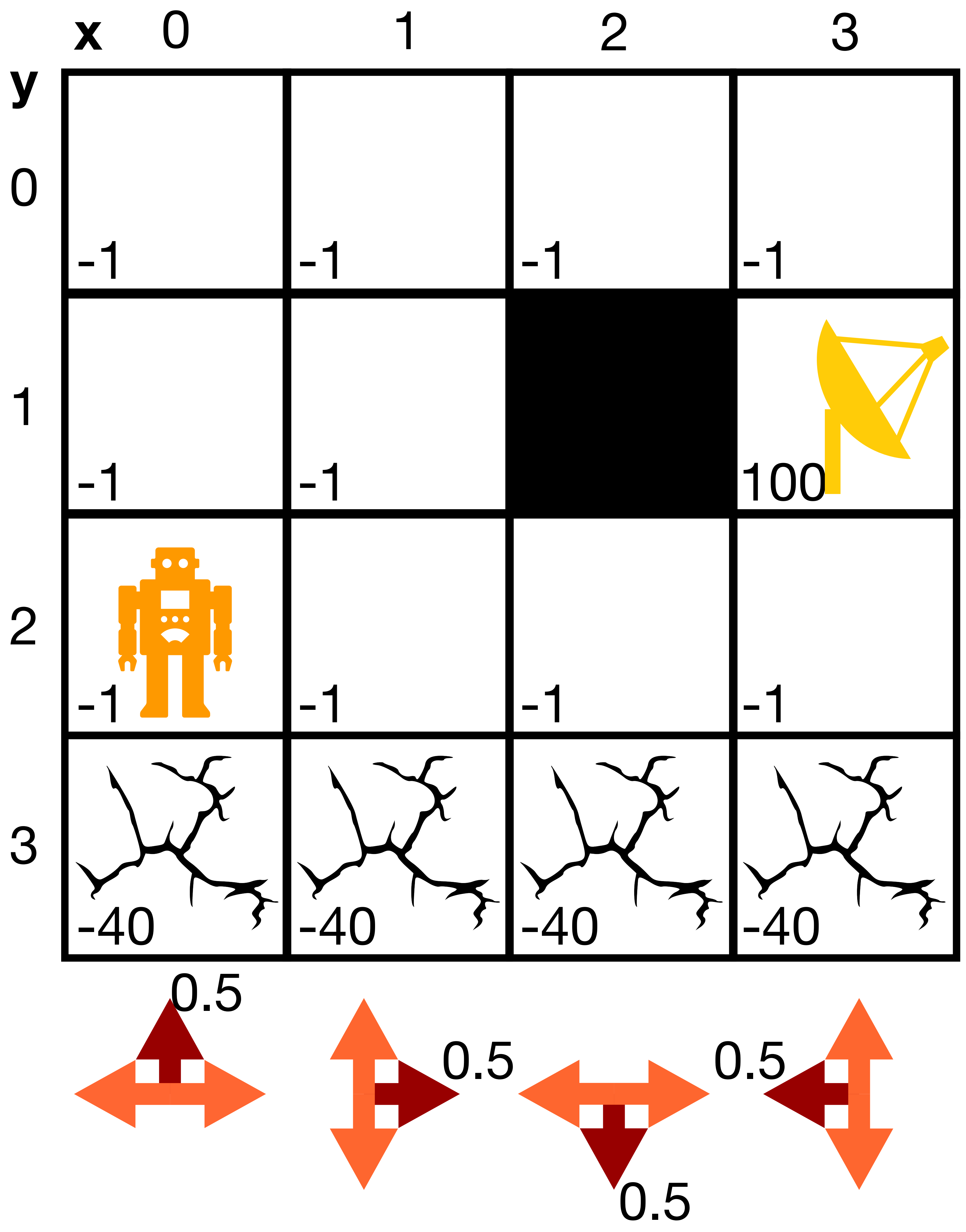


# RL Formalisation

## Markov Decision Process

- Markov Decision Process (MDP) is a tuple $(S, A, R)$:
  - $S$ — finite set of states
  - $A$ — set of actions
  - $R$ — set of rewards
- $p(s' \mid s, a)$ — MDP transition model
- $r_s^a \doteq \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a\right]$

  $= \sum_{s'} p(s' \mid s, a) r_{ss'}^a$ one step expected reward



# RL Formalisation

## Markov Decision Process

- Markov Decision Process (MDP) is a tuple $(S, A, R)$:
  - $S$ — finite set of states
  - $A$ — set of actions
  - $R$ — set of rewards
- $p(s' \mid s, a)$ — MDP transition model
- $r_s^a \doteq \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a\right]$

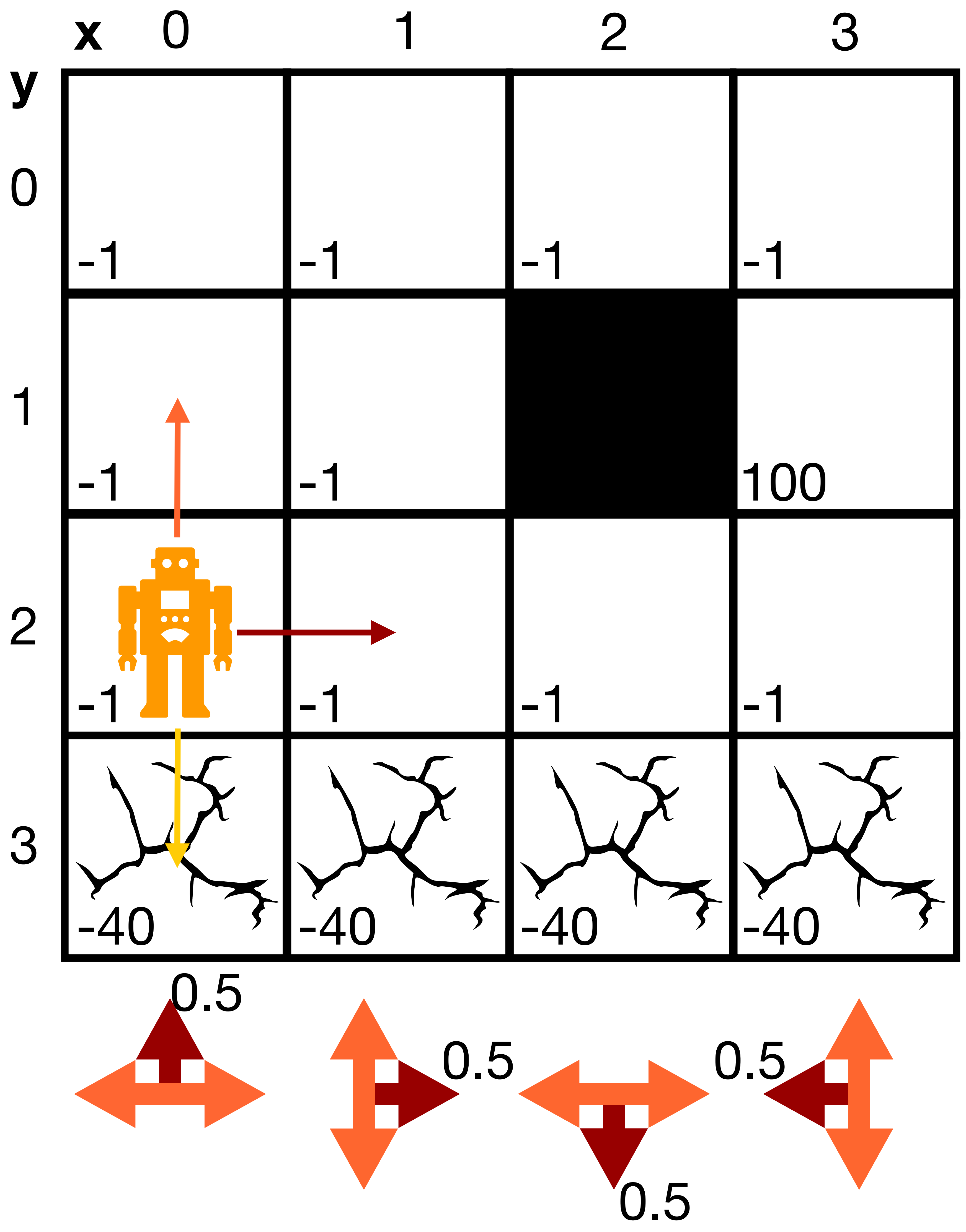
  $= \sum_{s'} p(s' \mid s, a) r$ one step expected reward

$(S_0 = (0,2), A_0 = \text{RIGHT}, \text{R}_1 = -1), (\text{S}_1 = (1,2), \ldots), \ldots$

$(S_0 = (0,2), A_0 = \text{RIGHT}, \text{R}_1 = -1), (\text{S}_1 = (0,1), \ldots), \ldots$

$(S_0 = (0,2), A_0 = \text{RIGHT}, \text{R}_1 = -40)$, ☠

$r_{(0,2)}^{\text{RIGHT}} = 0.5 \cdot (-1) + 0.25 \cdot (-1) + 0.25 \cdot (-40) = -10.75$

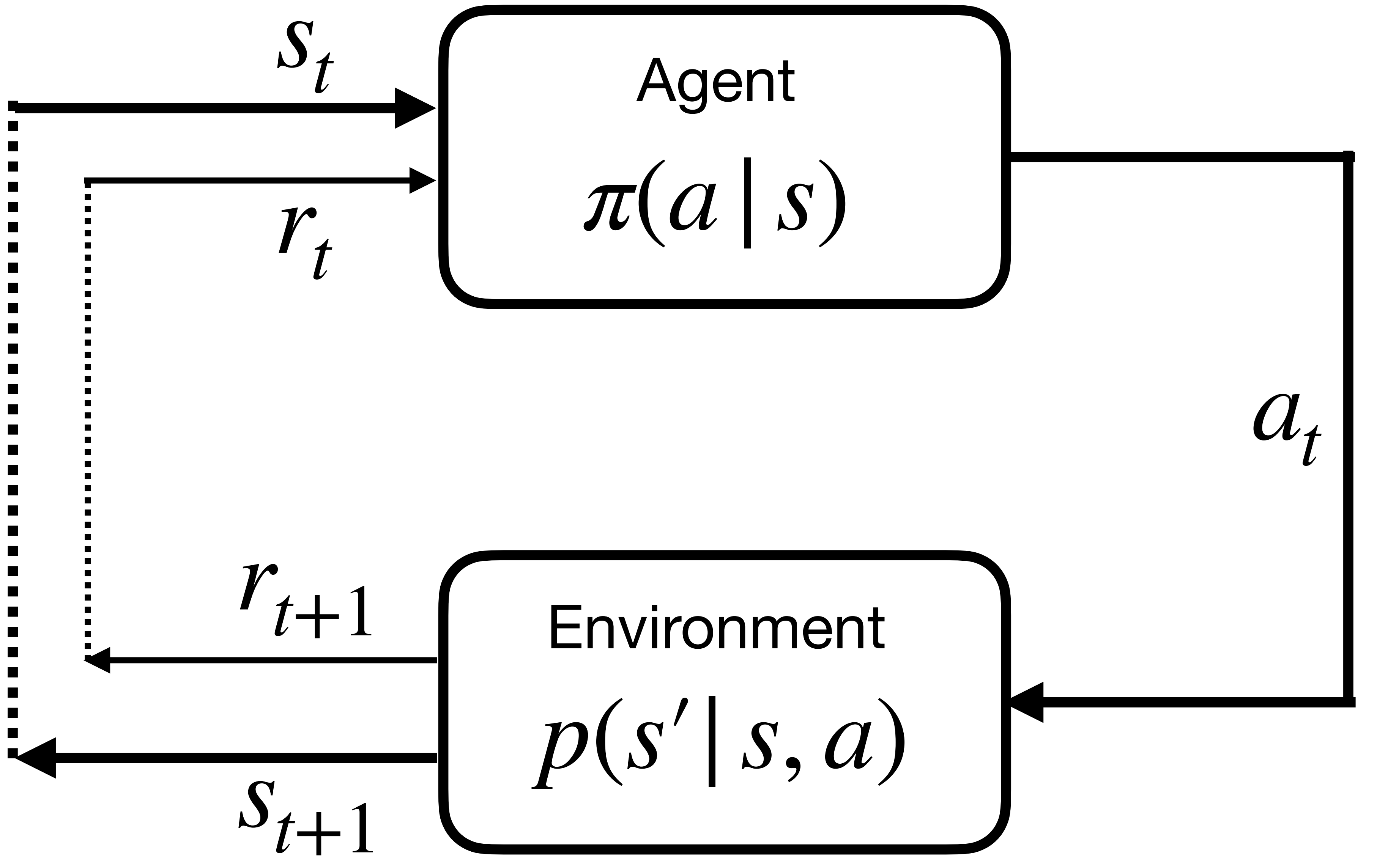


# RL Formalisation

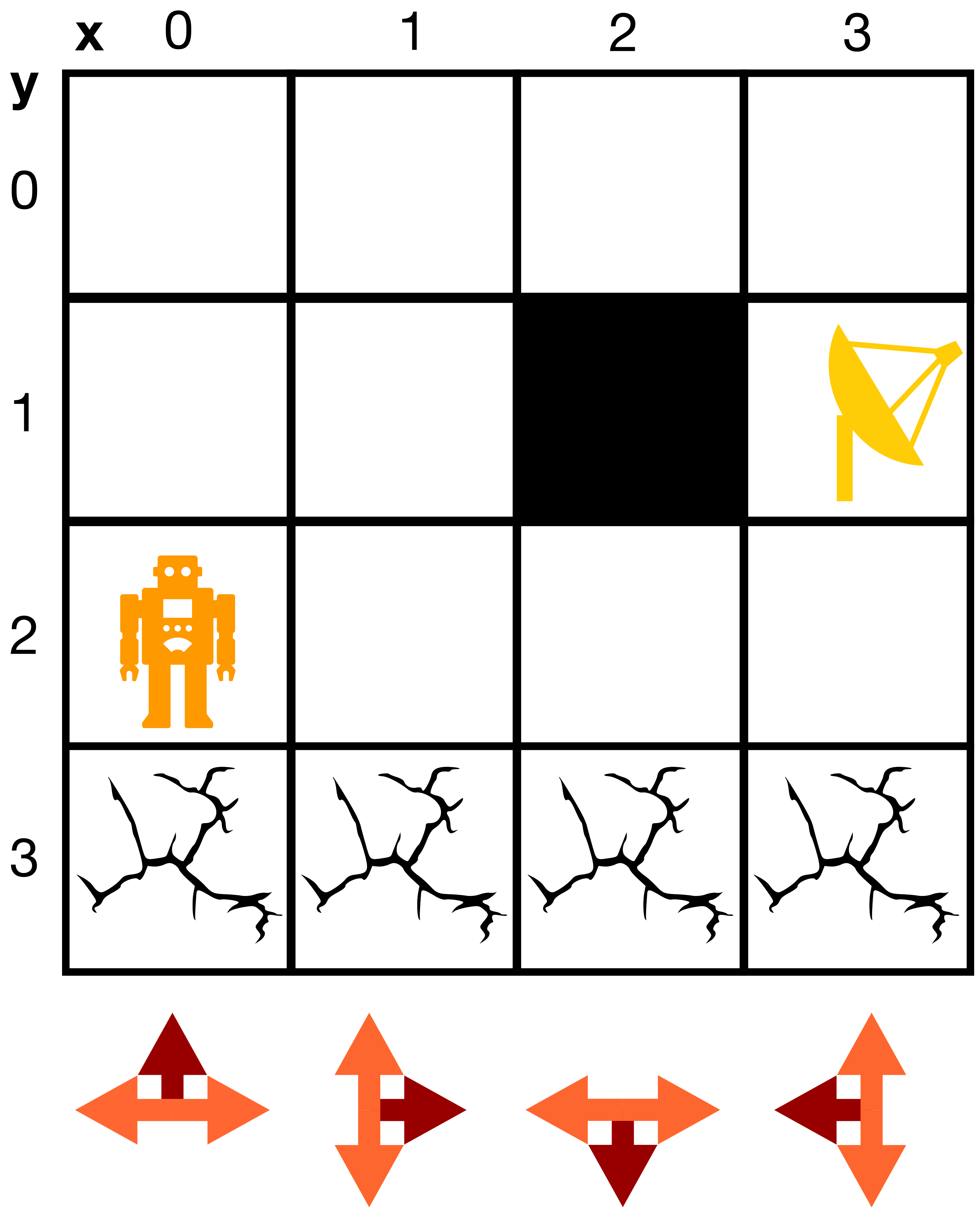
## Agents and Environments

- Agent-environment interaction forms a *trajectory*:

$$\tau = (S_0, A_0, R_1), (S_1, A_1, R_2), \ldots, (S_t, A_t, R_{t+1}), \ldots$$



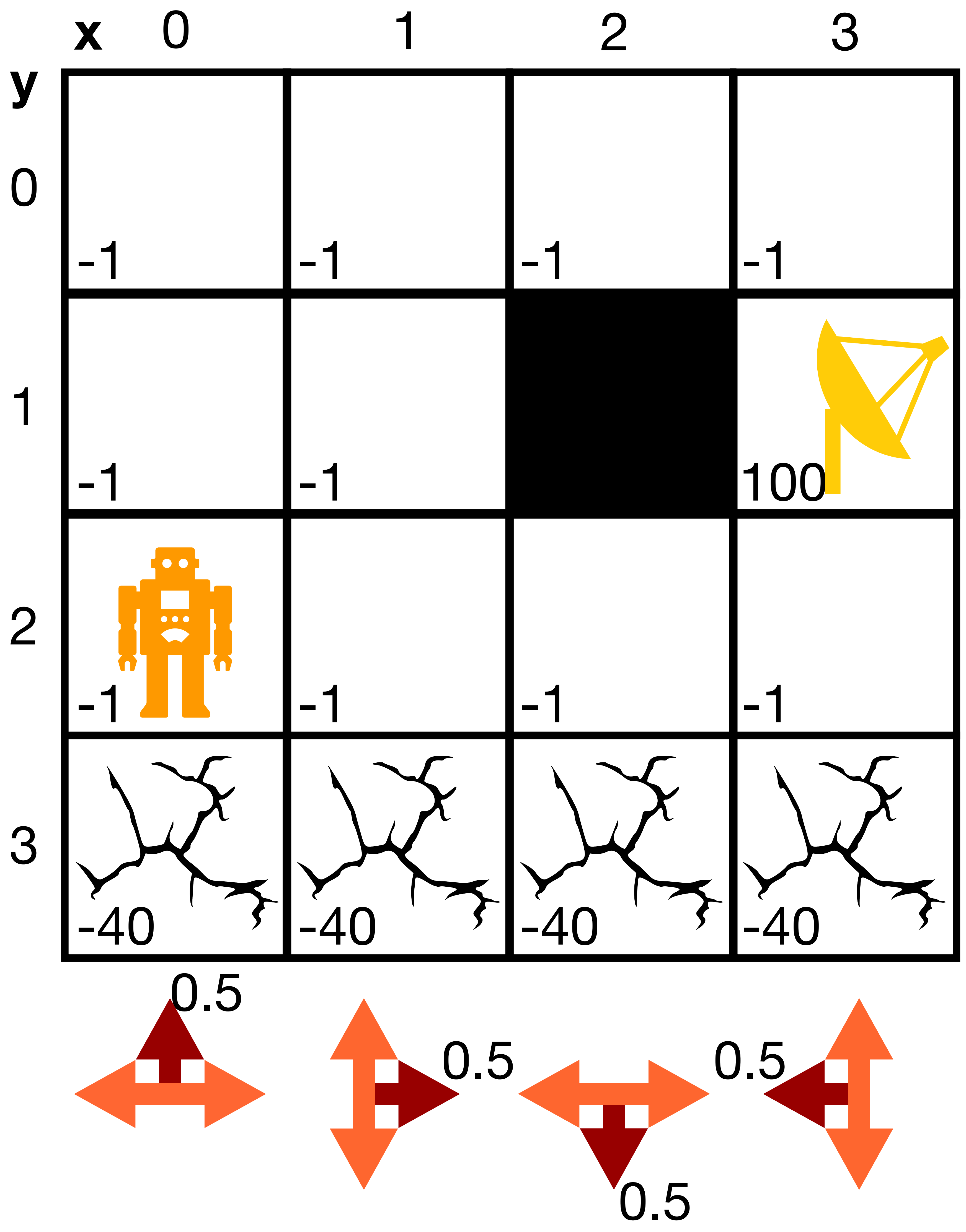

Agent-Environment Interaction [Sutton & Barto 2018]



# RL Formalisation

## Markov Decision Process

- **Markov property:**
  $P[S_{t+1} \mid S_t] = P[S_{t+1} \mid S_t, S_{t-1}, \ldots, S_0]$
  - The future is independent of the past given the present
  - The state is sufficient statistic of the future



# RL Formalisation

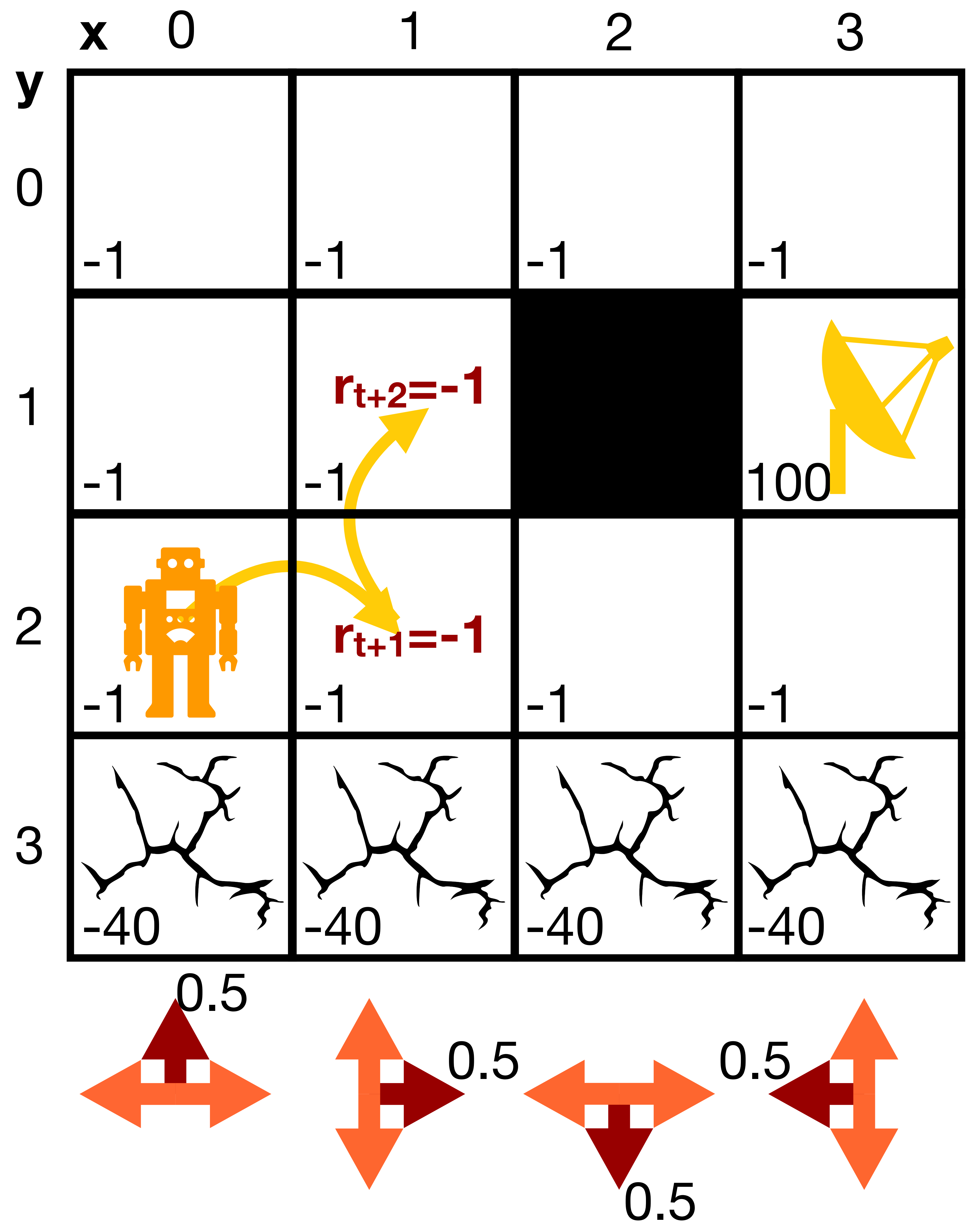
## Reward and Return

- Agent's goal is to maximise the *return* $G$:

$$G_t \doteq r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots$$
$$= r_{t+1} + \gamma(r_{t+2} + \gamma r_{t+3} + \ldots)$$
$$= r_{t+1} + \gamma G_{t+1}$$

- $\gamma \in [0,1]$ — future reward *discount factor*
  - Varying $\gamma$ varies the "far-sightedness"
  - Mathematically convenient in continuing problems and cyclic Markov processes:

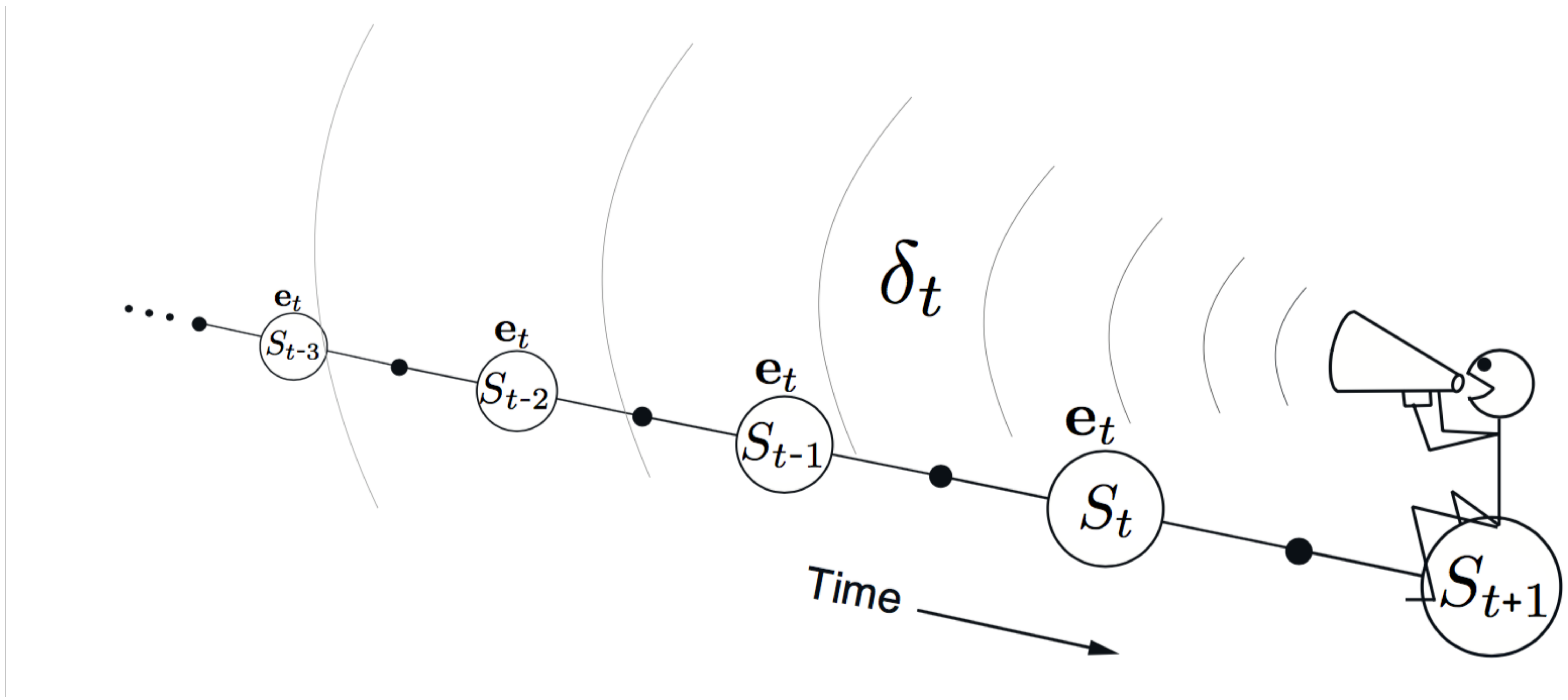
$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} = \frac{r}{1-\gamma}$$

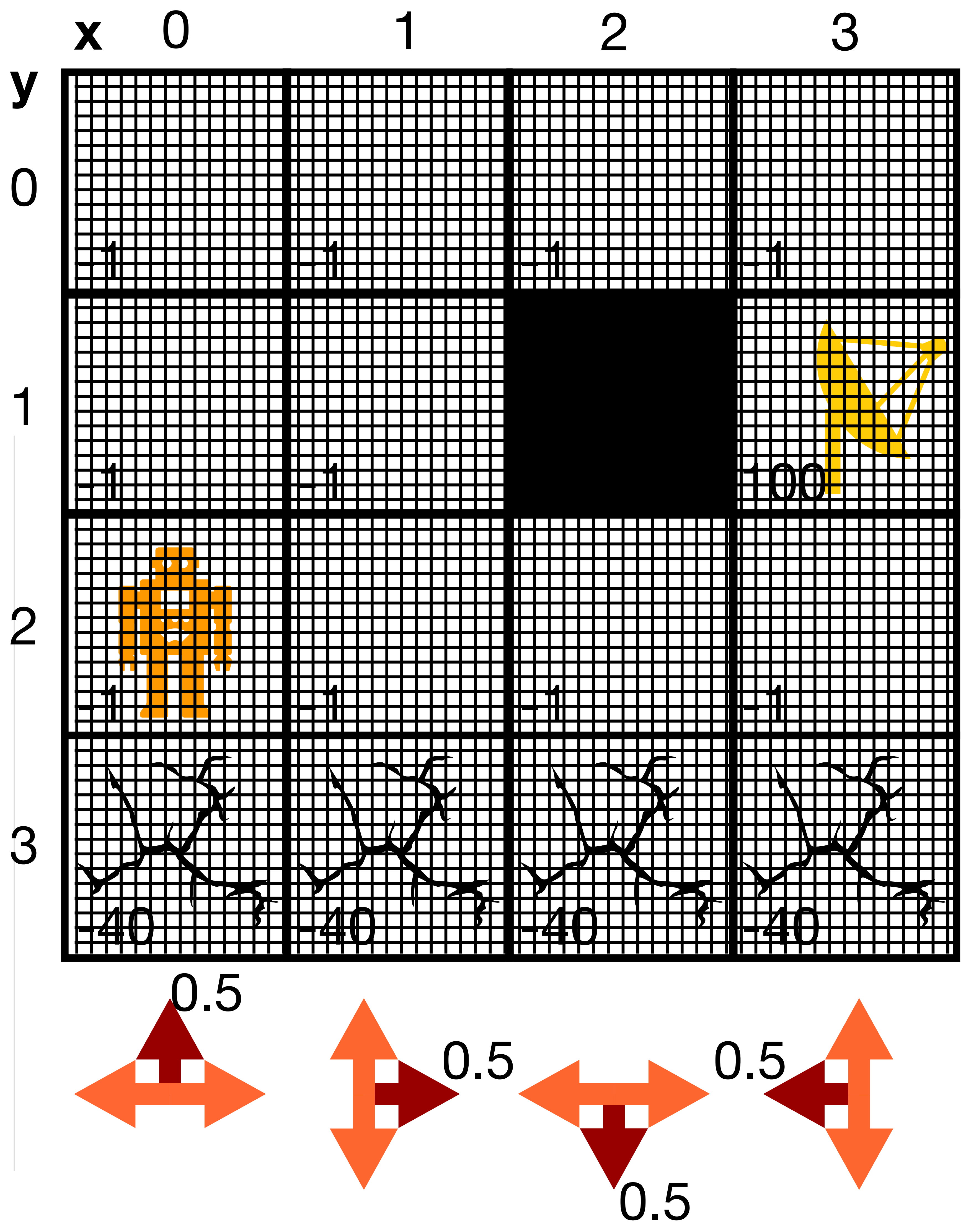


# RL Formalisation

## Reward and Return

- *Credit assignment problem*:
  - How do you distribute credit for success (or blame for failure) of a decision (action) among the many throughout the episode?

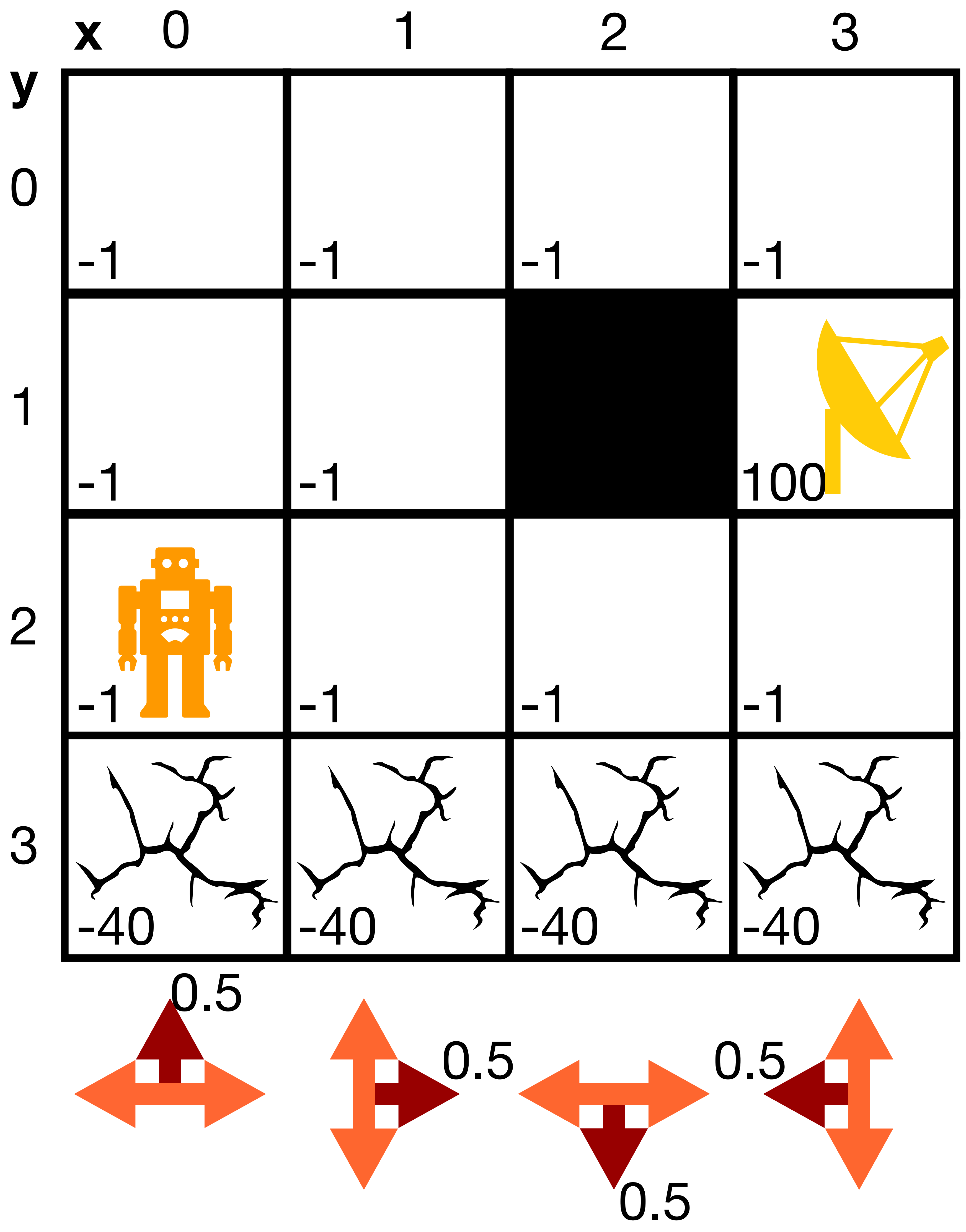


# RL Formalisation

## Policy

- Policy fully captures agent's reasoning process (agent = policy)
- Is the conditional probability distribution over actions $a \in A$ given states $s \in S$:
  - Deterministic policy: $a = \pi(s)$
    - e.g. *greedy policy*
  - Stochastic policy: $\pi(a\,|\,s) = P[A_t = a\,|\,S_t = s]$
    - e.g. *exploratory policy*
- *Learning* in RL refers to learning the policy that maximises the return



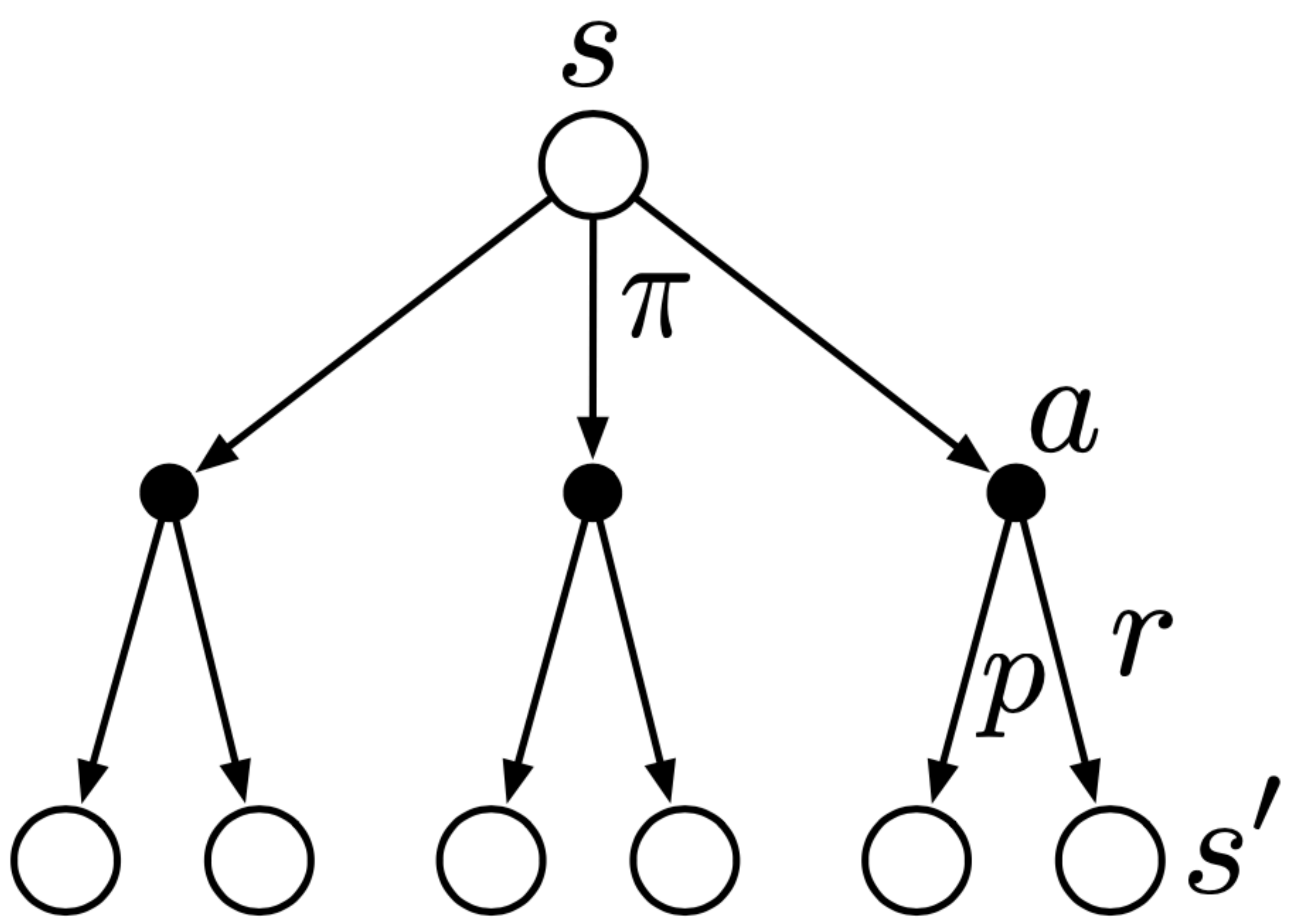
# RL Formalisation

## State-Value Function

- The state-value function $v_\pi(s)$ of an MDP is the expected return starting from state $s$, and then following policy $\pi$:

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s]$$
$$= \mathbb{E}_\pi[r_{t+1} + \gamma G_{t+1} \mid S_t = s]$$
$$= \sum_a \pi(a \mid s) \left[ r_s^a + \gamma \sum_{s'} p(s' \mid s, a) v_\pi(s') \right]$$

Between MDPs and SMDPs [Sutton et al. 1999]




Backup diagram for $v_\pi$ [Sutton & Barto 2018]

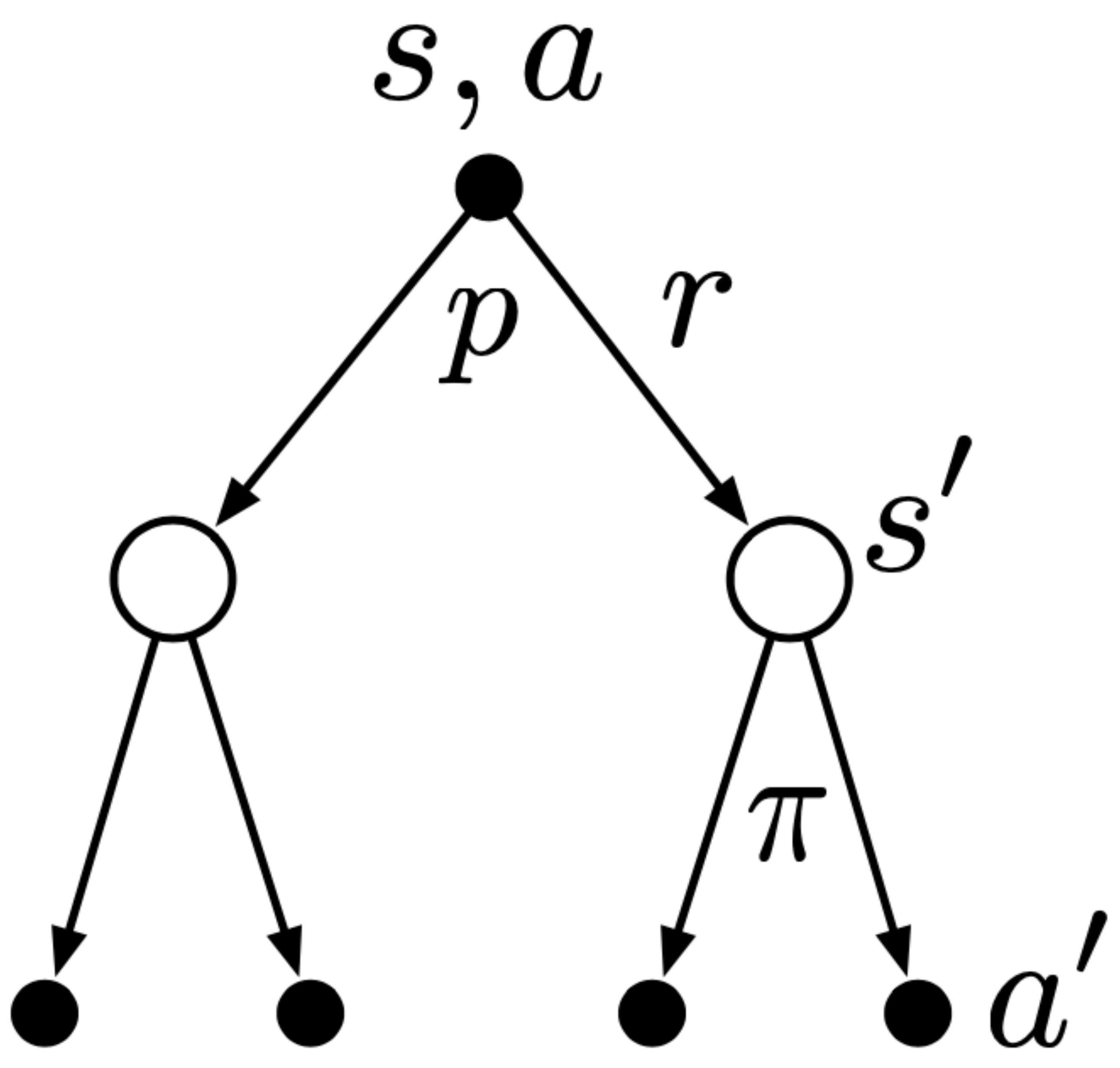
- Bellman equation for state-value

# RL Formalisation
## Action-Value Function

- The action-value function $q_\pi(s, a)$ of an MDP is the expected return starting from state $s$ by taking an action $a$, and then following policy $\pi$:

$$\begin{aligned} q_\pi(s,a) &\doteq \mathbb{E}_\pi[G_t \,|\, S_t = s, A_t = a] \\ &= \mathbb{E}_\pi[r_{t+1} + \gamma G_{t+1} \,|\, S_t = s, A_t = a] \\ &= r_s^a + \gamma \sum_{s'} p(s' \,|\, s, a) v_\pi(s') \\ &= r_s^a + \gamma \sum_{s'} p(s' \,|\, s, a) \sum_{a'} \pi(a' \,|\, s') q_\pi(s', a') \end{aligned}$$

Between MDPs and SMDPs [Sutton et al. 1999]




Backup diagram for $q_\pi$ [Sutton & Barto 2018]

- Bellman equation for action-value

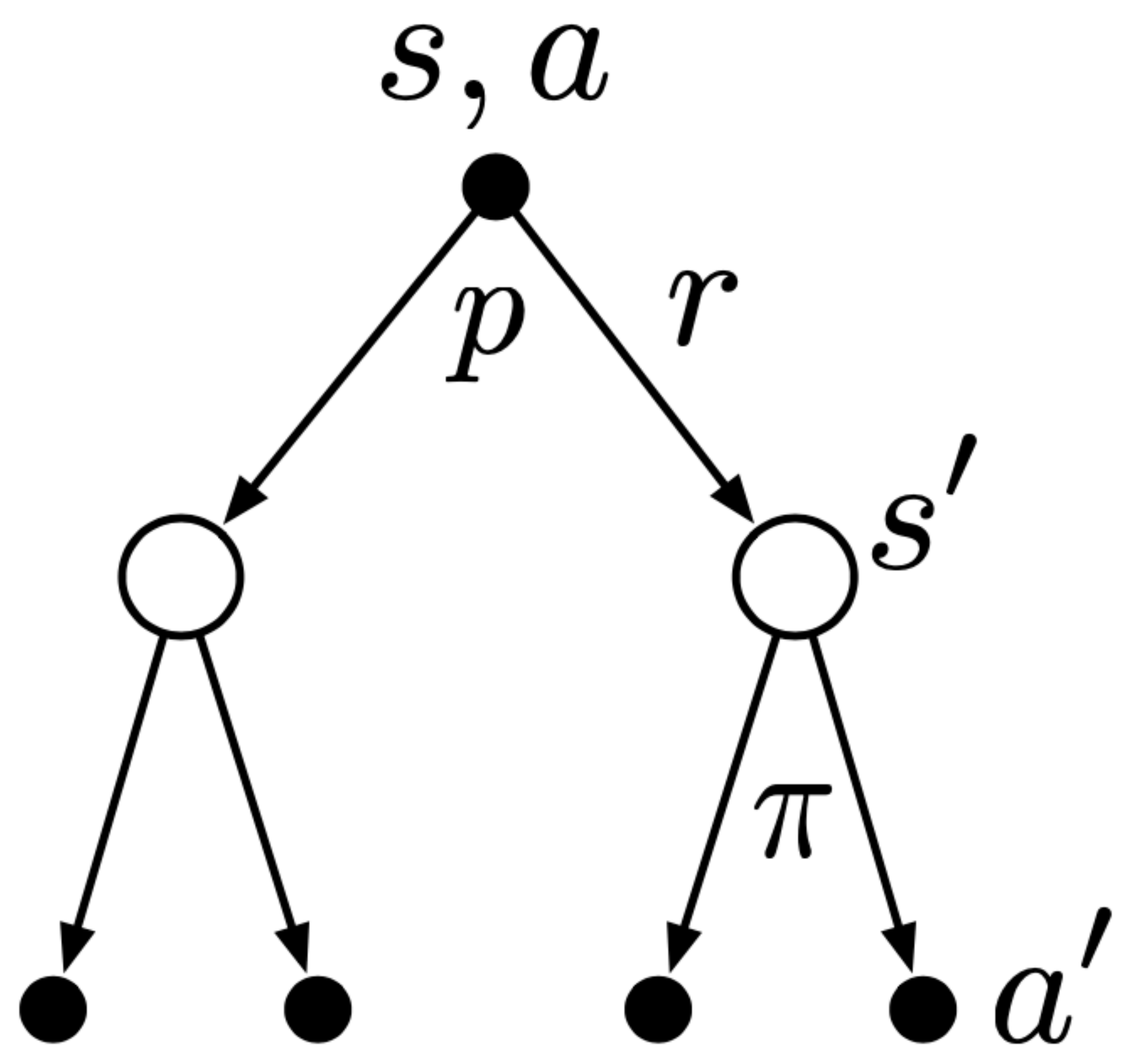
# RL Formalisation

## Action-Value Function

- The action-value function $q_\pi(s, a)$ of an MDP is the expected return starting from state $s$ by taking an action $a$, and then following policy $\pi$:

$$
\begin{aligned}
q_\pi(s,a) &\doteq \mathbb{E}_\pi[G_t \,|\, S_t = s, A_t = a] \\
&= \mathbb{E}_\pi[r_{t+1} + \gamma G_{t+1} \,|\, S_t = s, A_t = a] \\
&= r_s^a + \gamma \sum_{s'} p(s' \,|\, s, a) v_\pi(s') \\
&= r_s^a + \gamma \sum_{s'} p(s' \,|\, s, a) \sum_{a'} \pi(a' \,|\, s') q_\pi(s', a')
\end{aligned}
$$

Between MDPs and SMDPs [Sutton et al. 1999]




Backup diagram for $q_\pi$ [Sutton & Barto 2018]

Relationship between $v_\pi$ and $q_\pi$

# RL Formalisation

## Optimal Policy

- Value functions define a partial ordering over policies:

  $\pi' \geq \pi$ if and olnly if $v_{\pi'}(s) \geq v_\pi(s) \forall s \in S$

- A policy $\pi'$ is defined to be better than or equal to a policy $\pi$ if its expected return is greater than or equal to that of $\pi$ for all states
- There is always at least one policy that is better than or equal to all other policies, called the *optimal policy*, and denoted $\pi^*$

- **Policy Improvement Theorem**: If we have two policies $\pi$ and $\pi'$ so that $\pi(s) = \pi'(s)$ for all $s \in S$ except some $s'$ where $\pi'(s') = a \neq \pi(s')$ and $q_\pi(s, a) > v_\pi(s)$ then $\pi' > \pi$.

Proof: [Sutton & Barto 2018] p. 78

# RL Formalisation

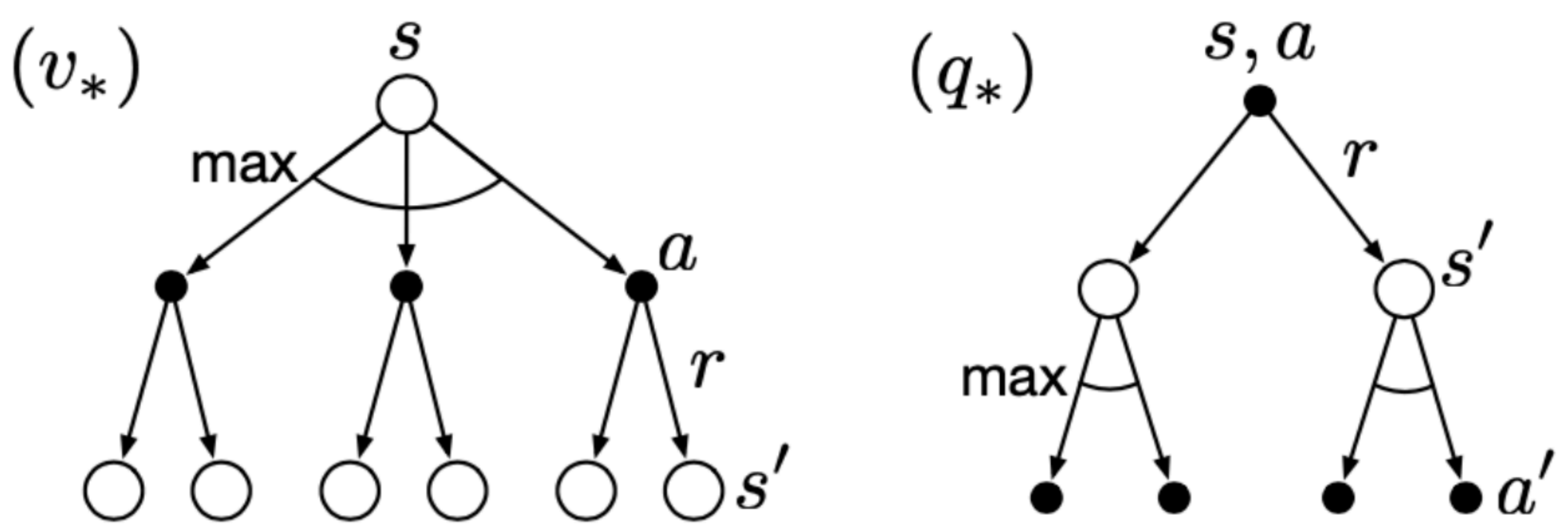
## Optimal Value Functions

- All optimal policies share the same state-value function $v^*(s)$ and action-value function $q^*(s, a)$
- Correspond to optimal policies, and optimal policies are greedy

$$
\begin{aligned}
v^*(s) &\doteq \max_{\pi} v_{\pi}(s) \\
&= \max_{a \in A} \mathbb{E}_{\pi}\left[r_{t+1} + \gamma G_{t+1} \,|\, S_t = s, A_t = a\right] \\
&= \max_{a \in A} \left[ r_s^a + \gamma \sum_{s'} p(s' \,|\, s, a) v^*(s') \right]
\end{aligned}
$$

$$
\begin{aligned}
q^*(s, a) &\doteq \max_{\pi} q_{\pi}(s, a) \\
&= r_s^a + \gamma \sum_{s'} p(s' \,|\, s, a) \max_{a' \in A} q^*(s', a'))
\end{aligned}
$$

Between MDPs and SMDPs [Sutton et al. 1999]




Optimal value backup diagrams [Sutton & Barto 2018]

# RL Formalisation

## Bellman Equations and Planning

- Equations for $v_\pi$ and $q_\pi$ are called *Bellman equations*
  - Set of recursive equations relating states (and actions) to successor states (and actions)
  - In principle, could be solved iteratively, or with a *dynamic programming* methods:
    - *value iteration*, *q-value iteration*, *policy iteration*
- Equations for $v*$ and $q*$ are called *Bellman optimality equations*

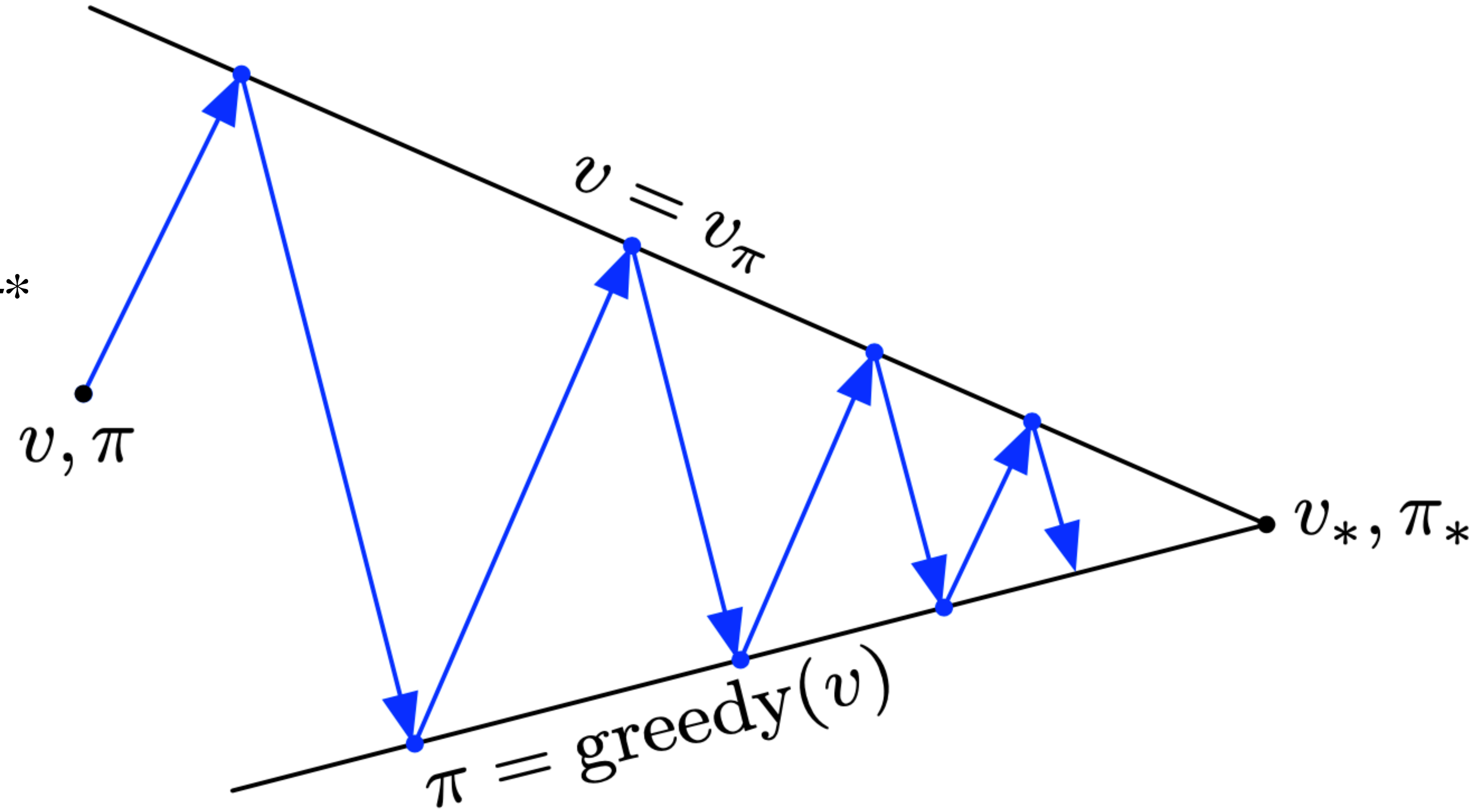
# RL Formalisation

## Generalised Policy Iteration (GPI)

- As a direct consequence of the **policy improvement theorem**:

$$\pi_0 \xrightarrow{Eval} v_{\pi_0} \xrightarrow{Impr} \pi_1 \xrightarrow{Eval} v_{\pi_1} \xrightarrow{Impr} \dots \pi^* \xrightarrow{Eval} v_{\pi^*}$$

- **Policy evaluation**: Estimate the true value $V \approx v_\pi$ iteratively
- **Policy improvement**: Use estimated $V \approx v_\pi$ to select a better policy $\pi' \geq \pi$, $\pi' = \mathrm{greedy(V)}$



GPI convergence [Sutton & Barto 2018]

# RL Formalisation

## Generalised Policy Iteration (GPI)

**Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$**

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

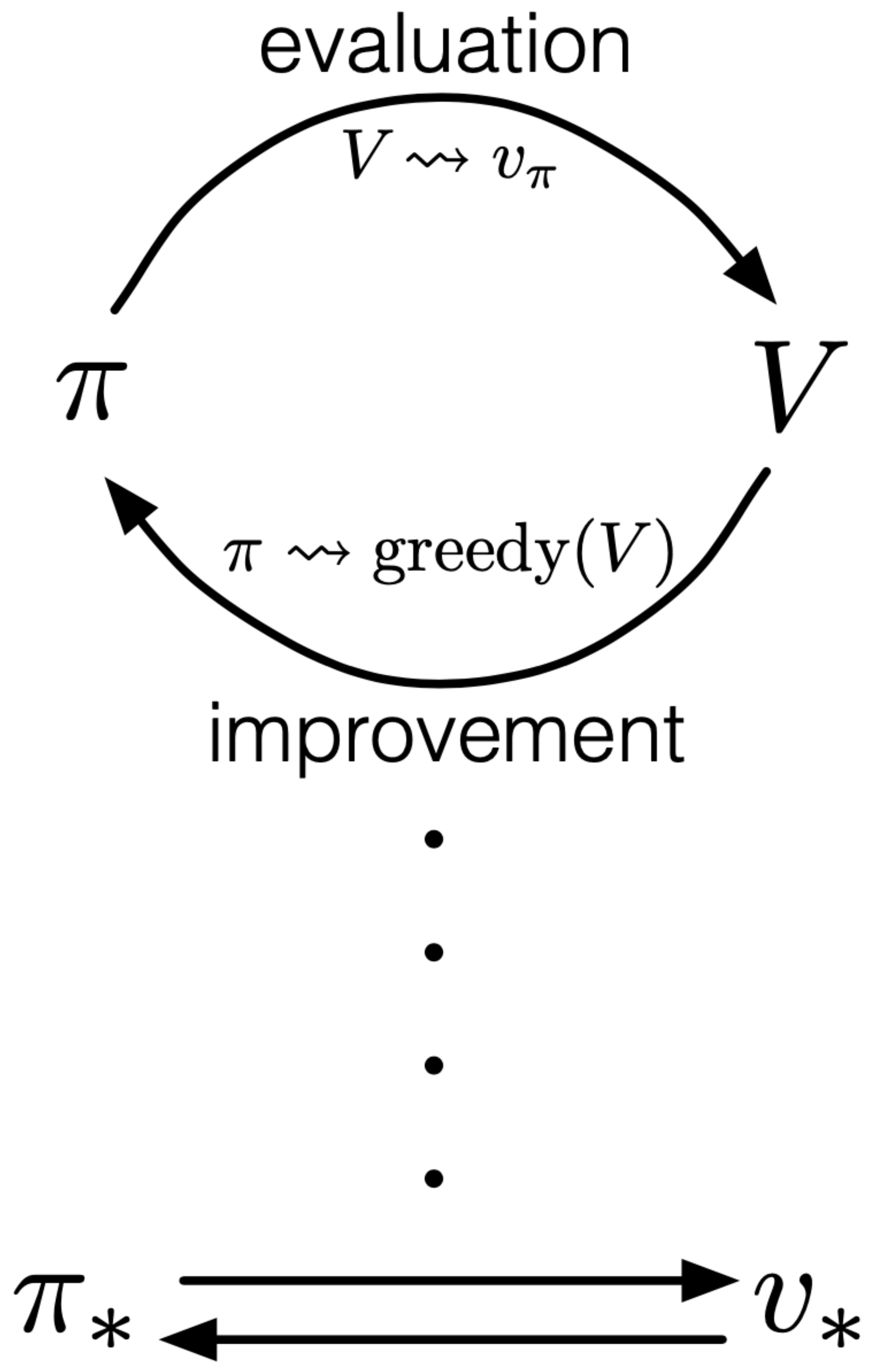
2. Policy Evaluation
   Loop:
   - $\Delta \leftarrow 0$
   - Loop for each $s \in \mathcal{S}$:
     - $v \leftarrow V(s)$
     - $V(s) \leftarrow \sum_{s'} p(s' \mid s, \pi(s)) \left[ r^a_{ss'} + \gamma V(s') \right]$
     - $\Delta \leftarrow \max(\Delta, |v - V(s)|)$

   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   *policy-stable* $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   - *old-action* $\leftarrow \pi(s)$
   - $\pi(s) \leftarrow \arg\max_a \sum_{s'} p(s' \mid s, a) \left[ r^a_{ss'} + \gamma V(s') \right]$
   - If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*

   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Policy Iteration algorithm [Sutton & Barto 2018]




GPI steps [Sutton & Barto 2018]

# RL Formalisation

## Exploration-Exploitation Trade-Off

- *Exploration*: Find more about the environment
- *Exploitation*: Utilise gained knowledge to garner higher returns

- The case of the agent tasked with garnering the highest return in a *continuing environment* throughout its entire lifetime:
  - If it commits to early-found schema for obtaining rewards, it may not find out possibly better schemas
  - If it overly explores, its return will suffer

# Outline

- Introduction
- Reinforcement Learning Formalisation
- **Model-Free Reinforcement Learning**
  - **Motivation**
  - **Monte Carlo and Temporal Difference Methods**
  - **MC and TD: Bias vs Variance Trade-Off**
  - **MC and TD: Future vs Previous Data**
  - **MC and TD: Summary**
  - **On-Policy vs Off-policy Learning**
  - **Example 1: First-visit MC**
  - **Example 2: Q-Learning**
  - **Conclusions**
  - **Beyond Tabular Methods**
- …

# Model free RL

## Motivation

- MDP transition model $p(s' \mid s, a)$ is usually unknown, or using it is impractical
- Without it equations for $v_\pi$, $q_\pi$, $v^*$, $q^*$ incomputable
- Two options:
  - Learn the model $p(s' \mid s, a)$, or use it to some degree if known — *model based RL (MBRL)*
  - Estimate $v_\pi$, $q_\pi$, $v^*$, $q^*$ directly without learning the model — *model free RL (MFRL)*

# Model free RL

## Motivation

- MDP transition model $p(s' \mid s, a)$ is usually unknown, or using it is impractical
- Without it equations for $v_\pi$, $q_\pi$, $v^*$, $q^*$ incomputable
- Two options:
    - Learn the model $p(s' \mid s, a)$, or use it to some degree if known — *model based RL (MBRL)*
    - Estimate $v_\pi$, $q_\pi$, $v^*$, $q^*$ directly without learning the model — *model free RL (MFRL)*


Alpha Go, Silver et al. 2016


OpenAI Five, Barner et al. 2019

# Model free RL

## Motivation

- MDP transition model $p(s' | s, a)$ is usually unknown, or using it is impractical
- Without it equations for $v_\pi$, $q_\pi$, $v^*$, $q^*$ incomputable
- Two options:
  - Learn the model $p(s' | s, a)$, or use it to some degree if known — *model based RL (MBRL)*
  - **Estimate $v_\pi$, $q_\pi$, $v^*$, $q^*$ directly without learning the model — *model free RL (MFRL)***
    - $V_t(s)$, $Q_t(s, a)$ are (imperfect) estimates to $v_\pi$, $q_\pi$ at computation time step $t$

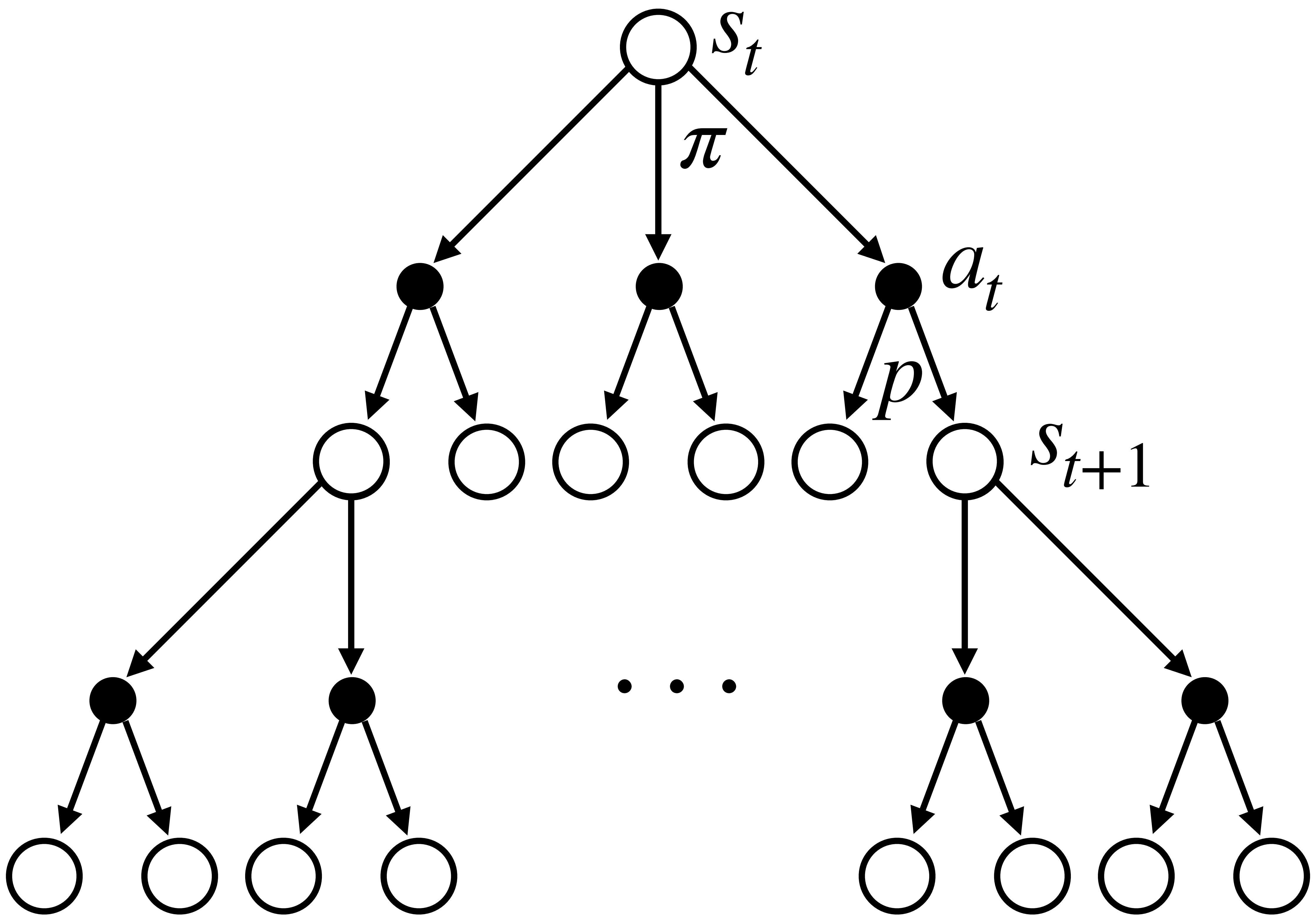

OpenAI Five, Barner et al. 2019

# Model free RL

## Motivation

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s]$$
$$= \mathbb{E}_\pi[r_{t+1} + \gamma G_{t+1} \mid S_t = s]$$

Between MDPs and SMDPs [Sutton et al. 1999]

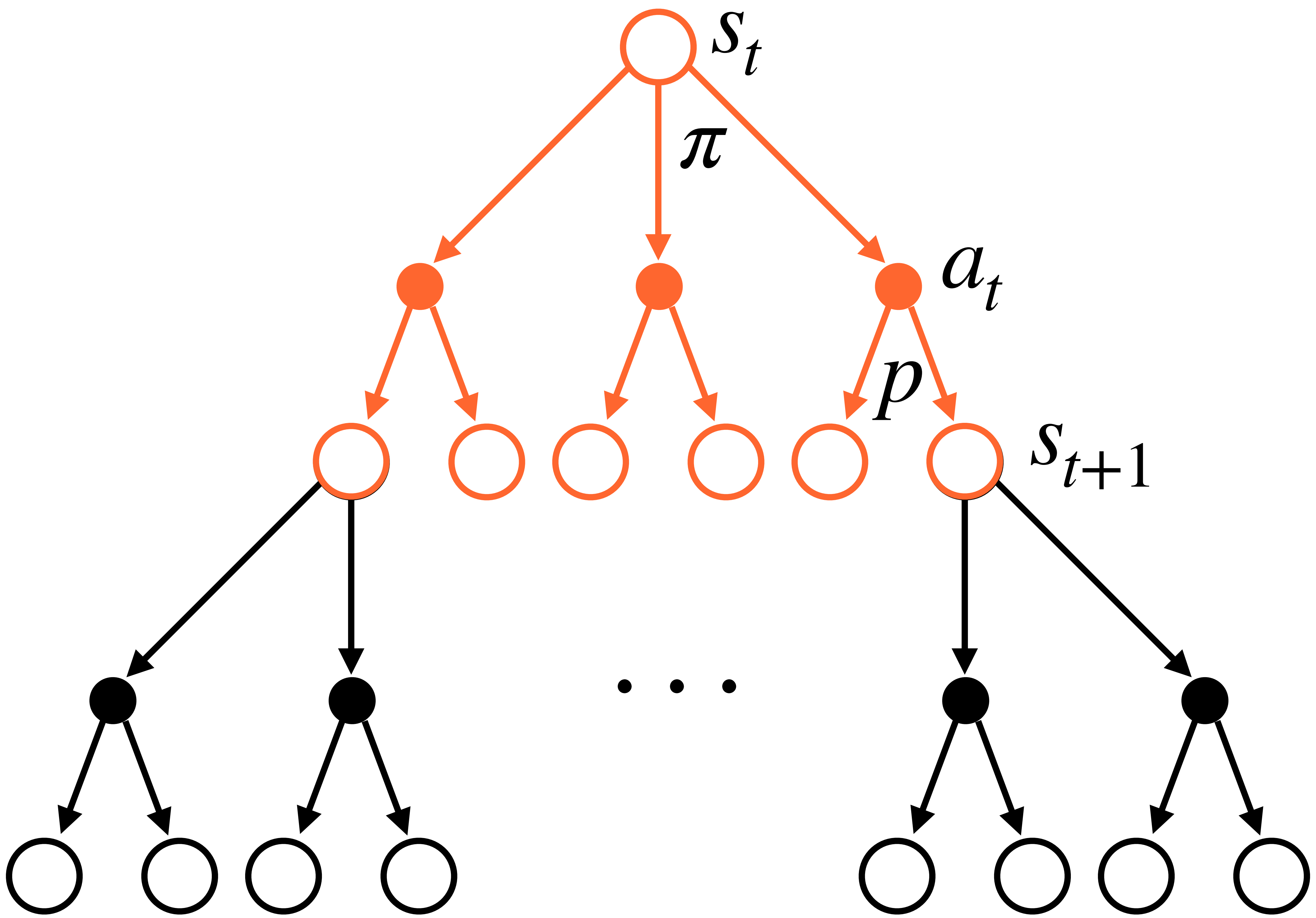


# Model free RL

## Motivation

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s]$$
$$= \mathbb{E}_\pi[r_{t+1} + \gamma G_{t+1} \mid S_t = s]$$
$$= \sum_a \pi(a \mid s) \left[ r_s^a + \gamma \sum_{s'} p(s' \mid s, a) v_\pi(s') \right]$$

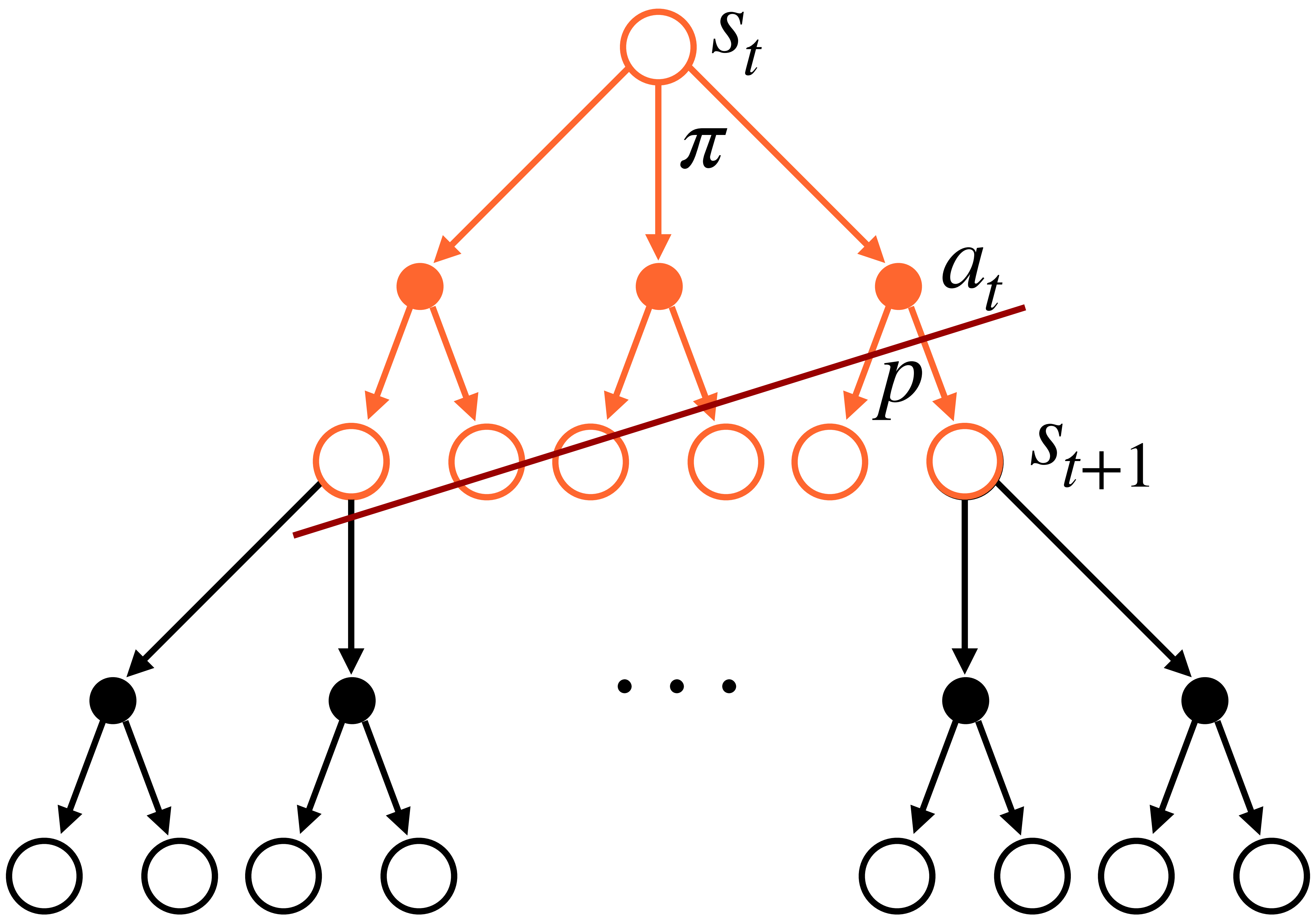
Between MDPs and SMDPs [Sutton et al. 1999]



# Model free RL
## Motivation

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \,|\, S_t = s]$$
$$= \mathbb{E}_\pi[r_{t+1} + \gamma G_{t+1} \,|\, S_t = s]$$
$$= \sum_a \pi(a\,|\,s)\left[r_s^a + \gamma \sum_{s'} \cancel{p(s'|s,a)} v_\pi(s')\right]$$

Between MDPs and SMDPs [Sutton et al. 1999]



# Model free RL

## Monte Carlo (MC) and Temporal Difference (TD)

### Monte Carlo (MC) methods

- **Core idea**:
  - Play entire episodes using a fixed policy $\pi$ and estimate state values $v_\pi$ as empirical means of returns
- Can only be applied to *episodic tasks*

### Temporal Difference (TD) methods

- **Core idea**:
  - Utilise the recursive nature of the Bellman equation to update state values $v_\pi$ based on states agent transitions to
- Learn from incomplete episodes, can be applied to *continuing tasks*
- They *bootstrap* — instead of measuring the true return $G_t$ they use $V(S_t)$ as its estimate, which in turn is also an estimate of the true $v_\pi(S_t)$

# Model free RL

## Monte Carlo (MC) and Temporal Difference (TD)

| Criteria | Monte Carlo Methods | Temporal Difference Methods |
| --- | --- | --- |
| Bias vs Variance | | |
| Online | | ✅ |
| Bootstrapping | | ✅ because $v_t$ is based off of $v_{t+1}$ |
| Estimation | | |
| On-Policy | | |
| Off-Policy | | |
| Past vs Future Future data | | |

# Model free RL

## Monte Carlo (MC) and Temporal Difference (TD)

$$V(s)_{n+1} = \frac{1}{n}\sum_{i=1}^{n} G_{s_i}$$
$$= \frac{1}{n}\left(G_{s_n} + \sum_{i=1}^{(n-1)} G_{s_i}\right)$$
$$= \frac{1}{n}\left(G_{s_n} + (n-1)\cdot\frac{1}{n-1}\sum_{i=1}^{(n-1)} G_{s_i}\right)$$
$$= \frac{1}{n}\left(G_{s_n} - (n-1)\cdot V(s)_n\right)$$
$$= V(s)_n + \frac{1}{n}\left(G_{s_n} - V(s)_n\right)$$

$$G_t = R + \gamma G_{t+1}$$
$$\approx R + \gamma V_{t+1}$$

### Monte Carlo (MC) Update

$$V(s) \leftarrow V(s) + \frac{1}{n}\left[G_t - V(s)\right]$$

### Temporal Difference (TD) Update

$$V(s) \leftarrow V(s) + \alpha\left[R + \gamma V(s') - V(s)\right]$$

# Model free RL

## Monte Carlo (MC) and Temporal Difference (TD)

$$
\begin{aligned}
V(s)_{n+1} &= \frac{1}{n}\sum_{i=1}^{n} G_{s_i} \\
&= \frac{1}{n}\left(G_{s_n} + \sum_{i=1}^{(n-1)} G_{s_i}\right) \\
&= \frac{1}{n}\left(G_{s_n} + (n-1)V(s)_n\right) \\
&= V(s)_n + \frac{1}{n}\left(G_{s_n} - V(s)_n\right)
\end{aligned}
$$

$$
\begin{aligned}
G_t &= R + \gamma G_{t+1} \\
&\approx R + \gamma V_{t+1}
\end{aligned}
$$

**Monte Carlo (MC) Update**

$$V(s) \leftarrow V(s) + \frac{1}{n}\left[G_t - V(s)\right]$$

step size

target

error

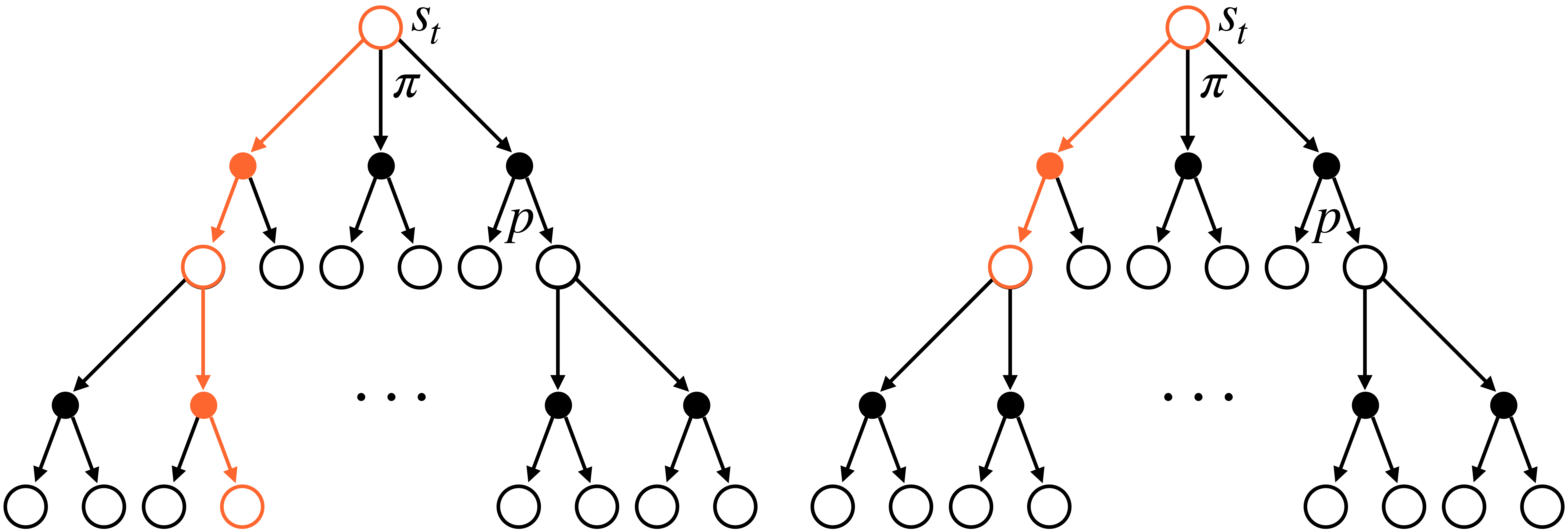
**Temporal Difference (TD) Update**

$\delta_t$

$$V(s) \leftarrow V(s) + \alpha\left[R + \gamma V(s') - V(s)\right]$$

# Model free RL

## Monte Carlo (MC) and Temporal Difference (TD)




**Monte Carlo (MC) Update**

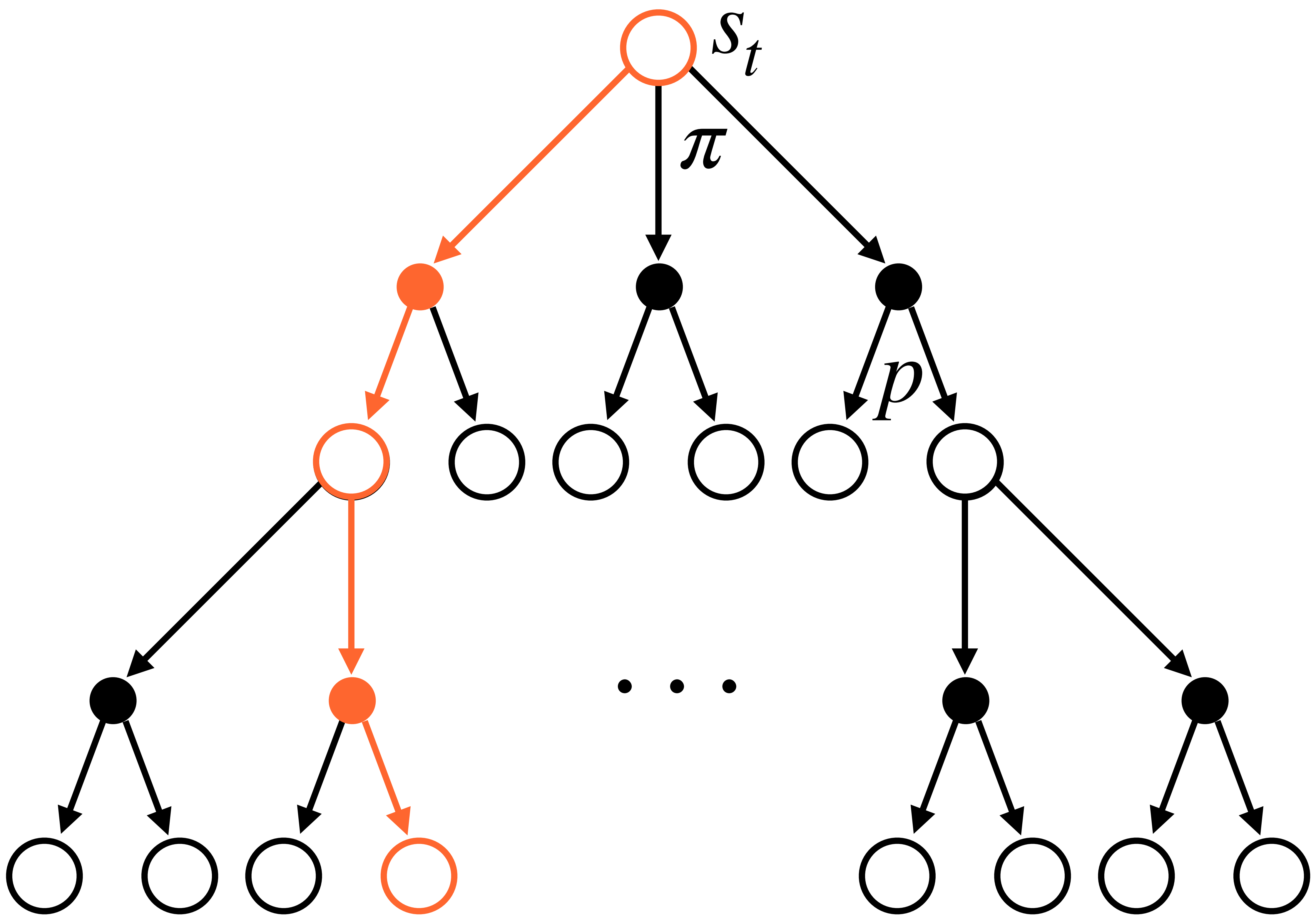
$$V(s) \leftarrow V(s) + \frac{1}{n}\left[G_t - V(s)\right]$$

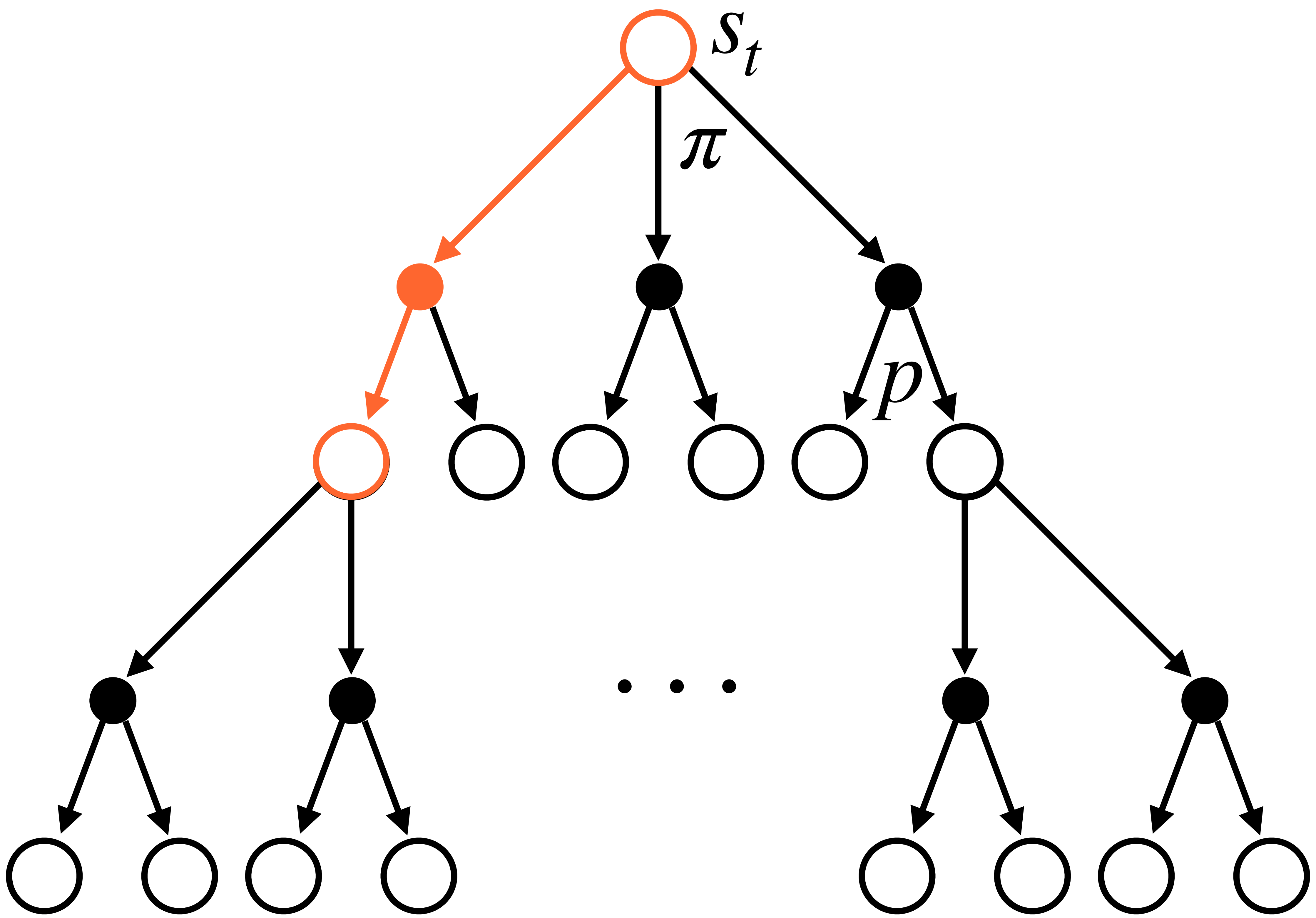
**Temporal Difference (TD) Update**

$$V(s) \leftarrow V(s) + \alpha\left[R + \gamma V(s') - V(s)\right]$$

# Model free RL

## Monte Carlo (MC) and Temporal Difference (TD)




**Monte Carlo (MC) Update**

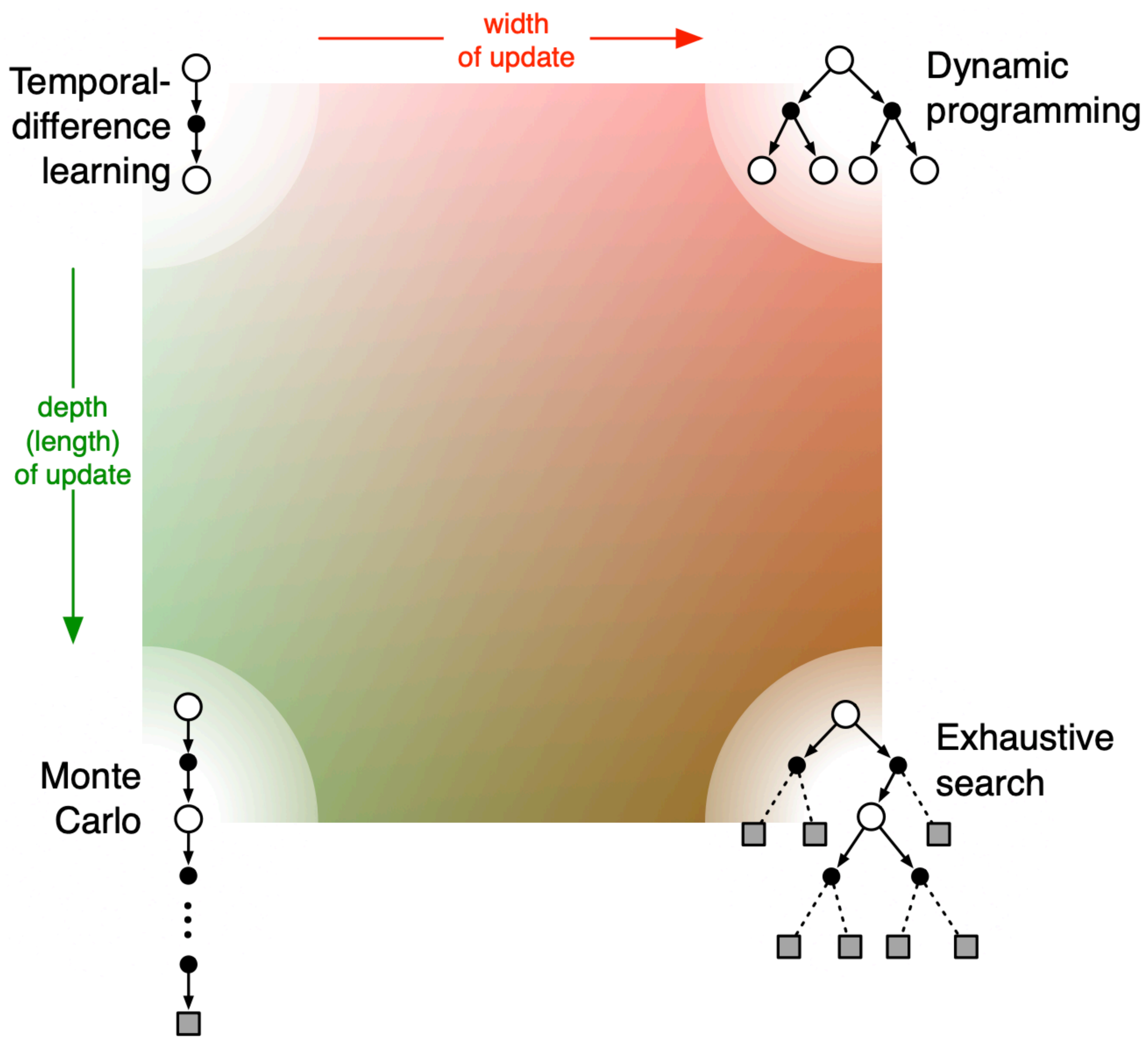
- Is an estimate because expectation over return is not known and is estimated by sample mean

**Temporal Difference (TD) Update**

- Is an estimate both because the expectation is unknown, and true $v_\pi(S_{t+1})$ is approximated by current estimate $V(S_{t+1})$

# Model free RL

## Monte Carlo (MC) and Temporal Difference (TD)




Comparison of RL methods [Sutton & Barto 2018]

# Model free RL

## MC and TD: Bias vs Variance Trade-Off

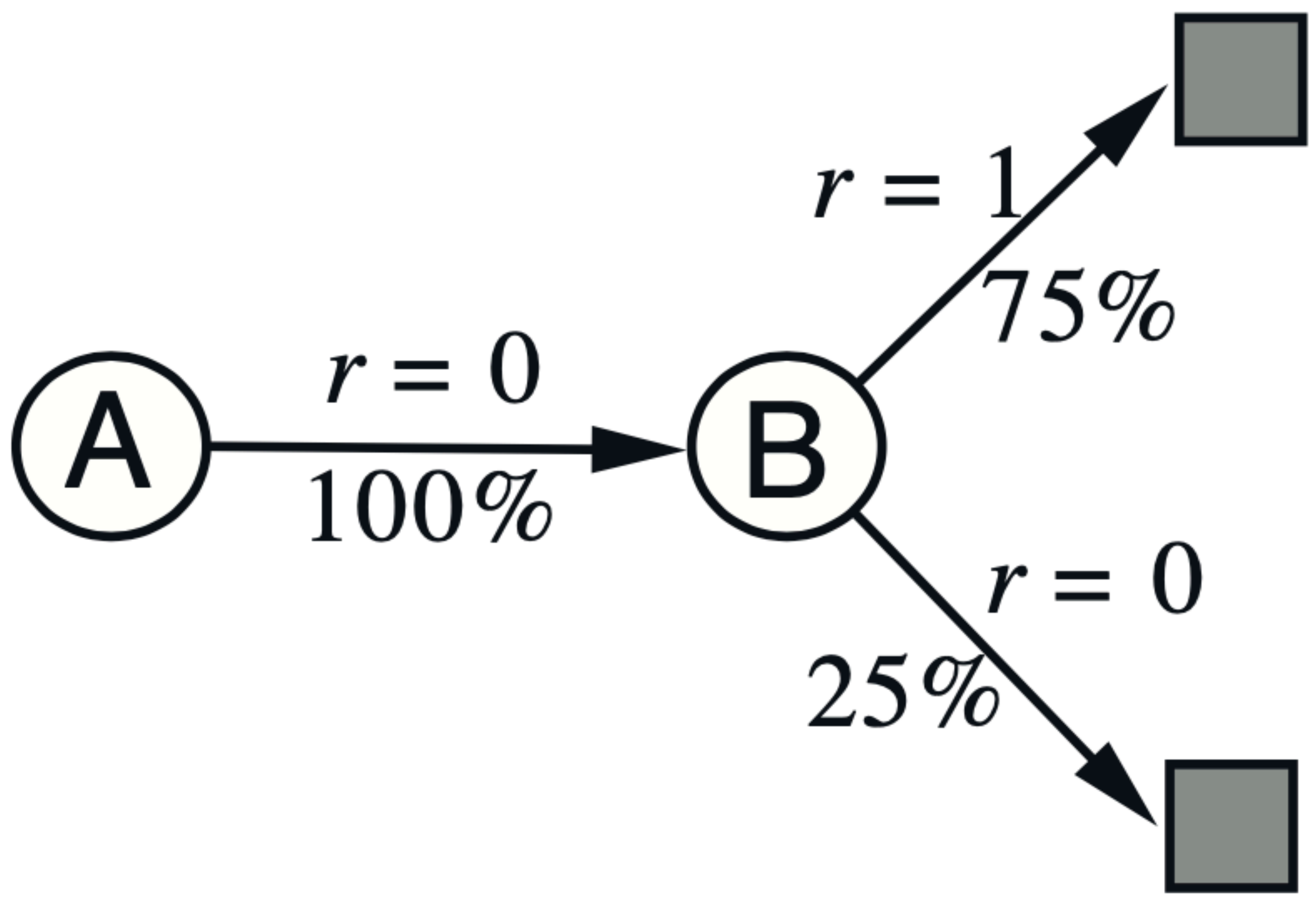
- MC Target [return $G_t$] is an *unbiased estimate* of $v_\pi(S_t)$
- TD Target $[R_{t+1} + \gamma V(S_{t+1})]$ is *biased estimate* of $v_\pi(S_t)$
- On the other hand, TD target has much lower variance compared to MC target:
  - Return depends on many steps during the episode, each potentially affected by the environment stochasticity
  - TD target depends only on a single step
- MC methods do not bootstrap, while TD methods bootstrap because estimating $V(S_t)$ uses another estimate $V(S_{t+1})$

# Model free RL

## MC and TD: Future vs Previous Data

Example

- A, 0, B, 0
- B, 1
- B, 1
- B, 1
- B, 1
- B, 1
- B, 1
- B, 0
- V(A) = ?; V(B) = ?



# Model free RL

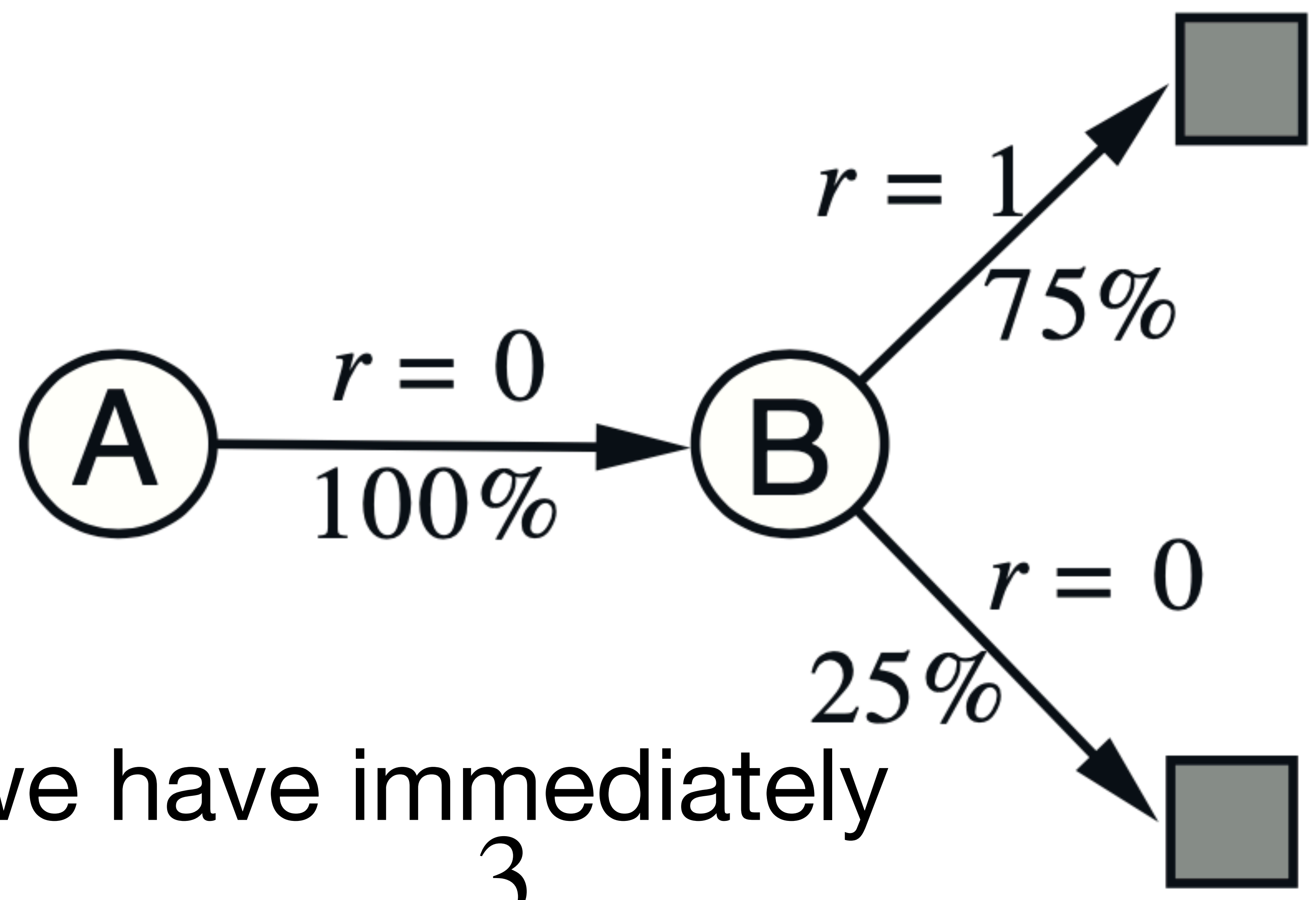
## MC and TD: Bias vs Variance Trade-Off

Example

- A, 0, B, 0
- B, 1
- B, 1
- B, 1
- B, 1
- B, 1
- B, 1
- B, 0
- V(A) = ?; V(B) = ?

**Solution**

**V(B)**: In 6/8 examples $G_t$ for $B$ was 1, in 2/8 examples it was 0. Both TD and MC would agree $V(B) = \frac{3}{4}$.




**V(A)**: Two lines of reasoning:

- TD: Every time $A$ was seen, we have immediately transitioned into $B$, so $V(A) = V(B) = \frac{3}{4}$.
- MC: $A$ was seen only once, and the return was 0, therefore $V(A) = 0$ ¯_(ツ)_/¯

# Model free RL

## MC and TD: Summary

- MC methods minimise the error on the training set
- TD methods try to estimate the underlying MDP and give its maximum likelihood estimate
- MC methods need to wait for episodes to finish in order to update value functions
- TD methods can update value functions at each time-step
- MC methods have low (no) bias but high variance
- TD methods have hight bias and low variance
- MC methods are estimates because the expectation $\mathbb{E}_\pi[G_t \mid S_t = s]$ is unknown
- TD methods are estimates both because the expectation $\mathbb{E}_\pi[R_{t+1} + G_{t+1} \mid S_t = s]$ is unknown, and because $V(S_{t+1})$ is used instead of $v_\pi(S_{t+1})$

# Model free RL

## On-Policy vs Off-Policy Learning

- All learning methods face a dilemma:
  - They seek to learn action values conditional on subsequent optimal behaviour
  - But they need to behave non-optimally in order to explore all actions
- On-policy learning:
  - Learn about policy $\pi$ from experience sampled by $\pi$
  - $\pi$ is neither fully greedy, nor fully exploratory — we need to keep sufficient exploration in order to converge to good behaviour
  - Conceptually very simple
- Off-policy learning:
  - Learn about *target* policy $\pi$ from experience sampled by *behaviour* policy $b$
  - Allows for learning from past policy experiences or people
  - Alleviates exploration-exploitation tradeoff, but is more conceptually challenging

# Model free RL

## Example 1: On-Policy MC Algorithm

**On-policy first-visit MC control (for $\varepsilon$-soft policies), estimates $\pi \approx \pi_*$**

Algorithm parameter: small $\varepsilon > 0$

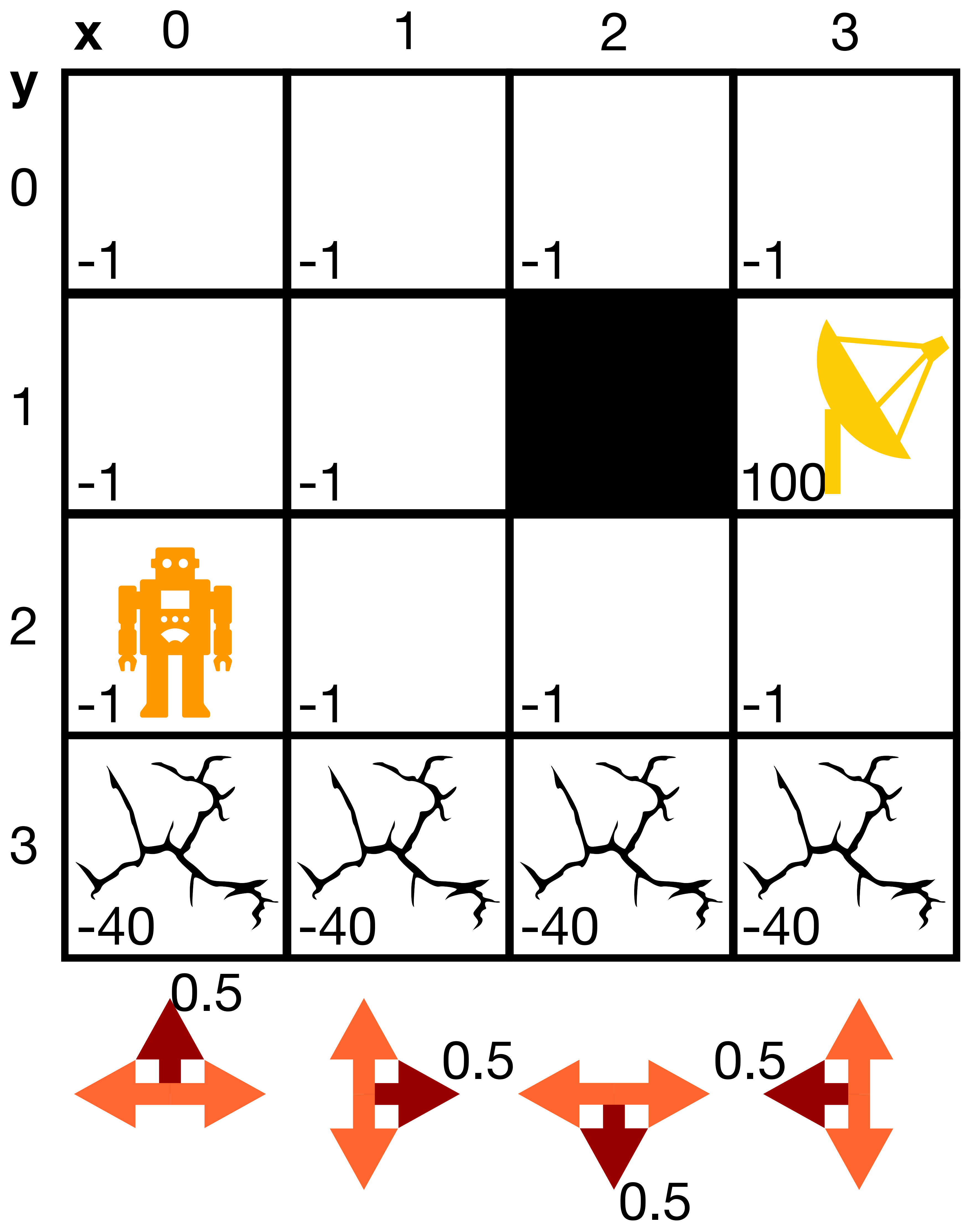
Initialize:

- $\pi \leftarrow$ an arbitrary $\varepsilon$-soft policy
- $Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$
- $Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Repeat forever (for each episode):

- Generate an episode following $\pi$: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
- $G \leftarrow 0$
- Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
  - $G \leftarrow \gamma G + R_{t+1}$
  - Unless the pair $S_t, A_t$ appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:
    - Append $G$ to $Returns(S_t, A_t)$
    - $Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$
    - $A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)
    - For all $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$



# Model free RL

## Example 1: On-Policy MC Algorithm

**On-policy first-visit MC control (for $\varepsilon$-soft policies), estimates $\pi \approx \pi_*$**

Algorithm parameter: small $\varepsilon > 0$

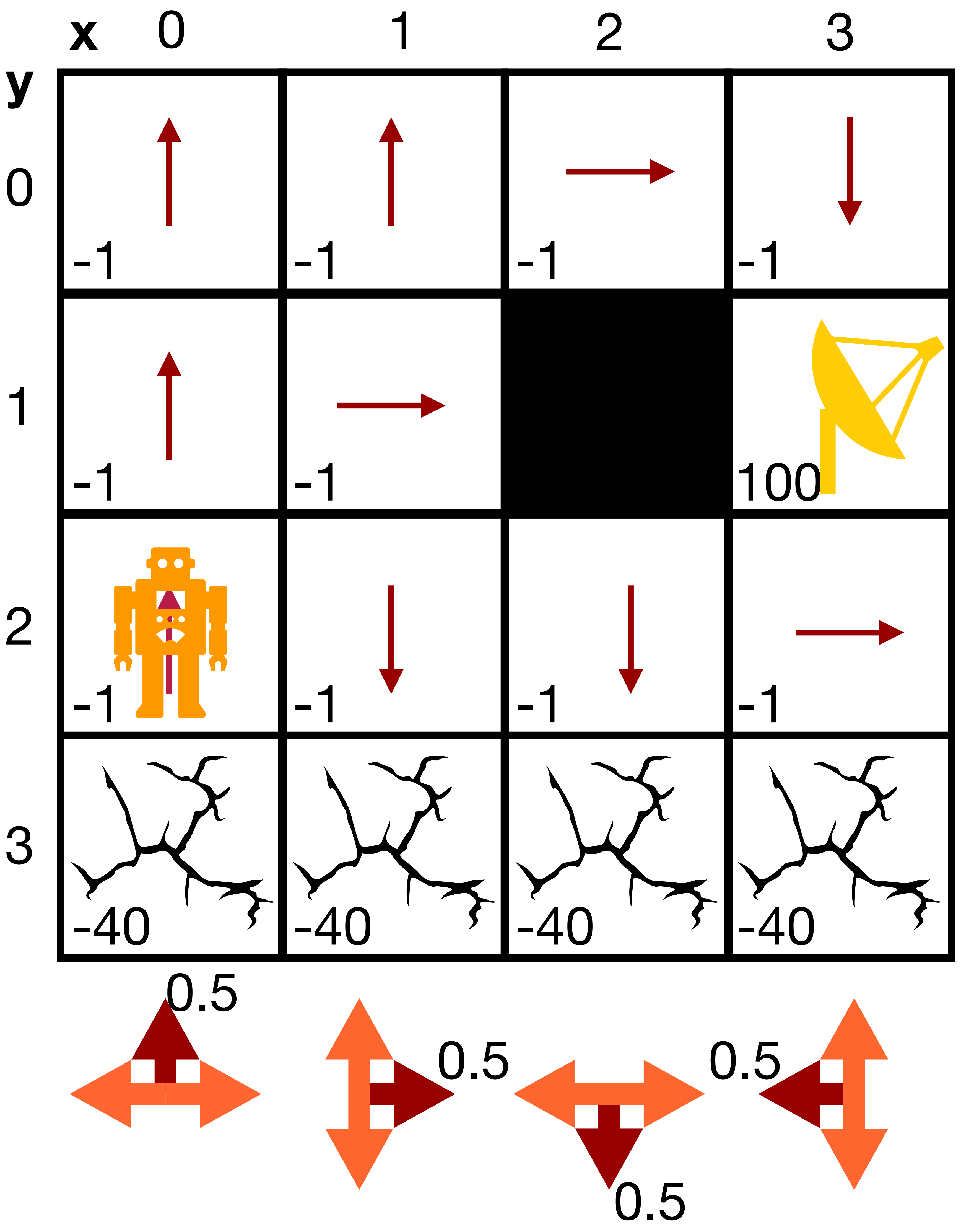
Initialize:

- $\pi \leftarrow$ an arbitrary $\varepsilon$-soft policy
- $Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$
- $Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Repeat forever (for each episode):

- Generate an episode following $\pi$: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
- $G \leftarrow 0$
- Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
  - $G \leftarrow \gamma G + R_{t+1}$
  - Unless the pair $S_t, A_t$ appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:
    - Append $G$ to $Returns(S_t, A_t)$
    - $Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$
    - $A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)
    - For all $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$



# Model free RL

## Example 1: On-Policy MC Algorithm

**On-policy first-visit MC control (for $\varepsilon$-soft policies), estimates $\pi \approx \pi_*$**

Algorithm parameter: small $\varepsilon > 0$

Initialize:

$\quad \pi \leftarrow$ an arbitrary $\varepsilon$-soft policy

$\quad Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

$\quad Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Repeat forever (for each episode):

$\quad$ Generate an episode following $\pi$: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$\quad G \leftarrow 0$

$\quad$ Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$\quad\quad G \leftarrow \gamma G + R_{t+1}$

$\quad\quad$ Unless the pair $S_t, A_t$ appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

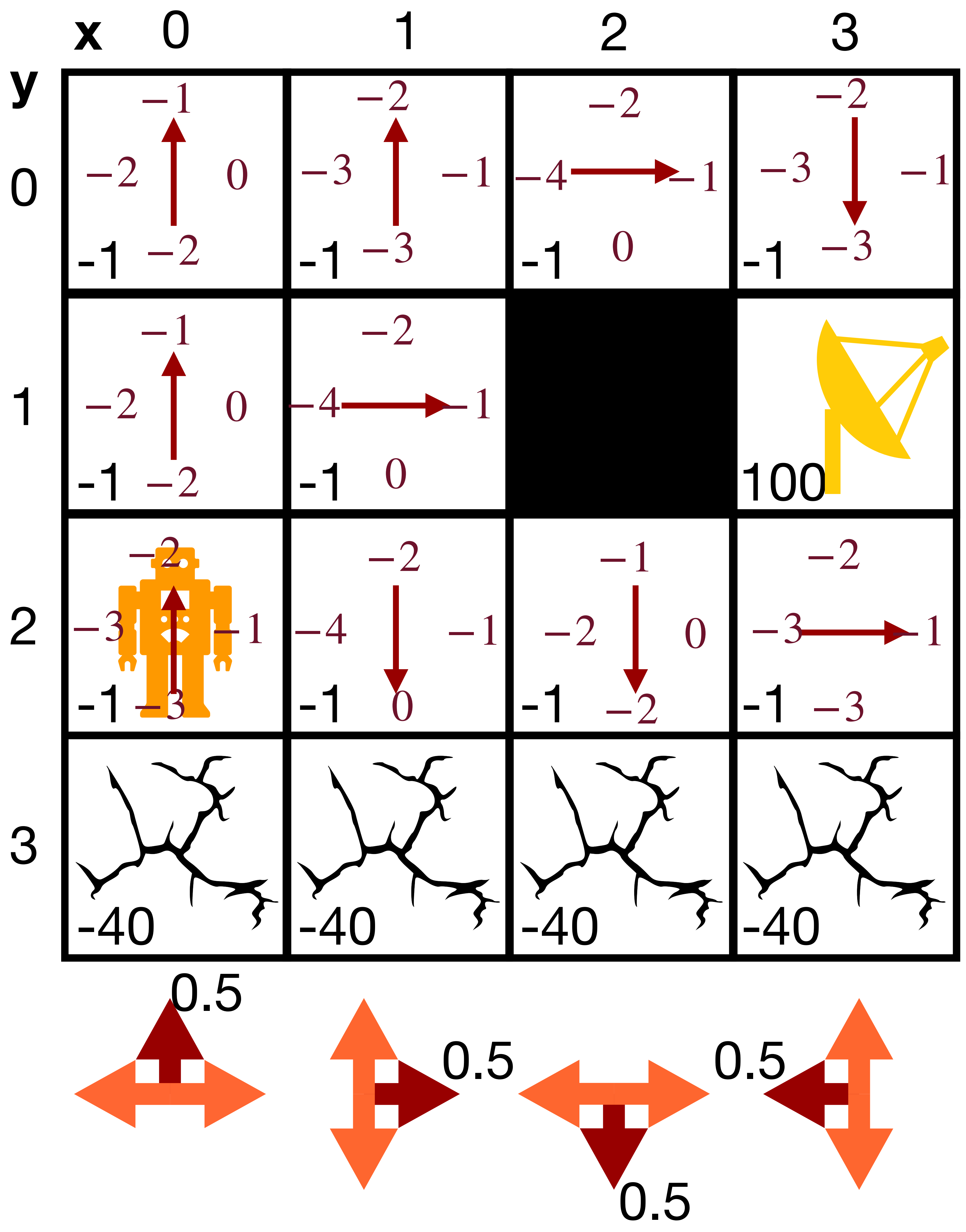
$\quad\quad\quad$ Append $G$ to $Returns(S_t, A_t)$

$\quad\quad\quad Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$\quad\quad\quad A^* \leftarrow \arg\max_a Q(S_t, a)$ $\quad$ (with ties broken arbitrarily)

$\quad\quad\quad$ For all $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$



# Model free RL

## Example 1: On-Policy MC Algorithm

**On-policy first-visit MC control (for $\varepsilon$-soft policies), estimates $\pi \approx \pi_*$**

Algorithm parameter: small $\varepsilon > 0$

Initialize:

  $\pi \leftarrow$ an arbitrary $\varepsilon$-soft policy

  $Q(s, a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

  $Returns(s, a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Repeat forever (for each episode):

  Generate an episode following $\pi$: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

  $G \leftarrow 0$

  Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

    $G \leftarrow \gamma G + R_{t+1}$

    Unless the pair $S_t, A_t$ appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

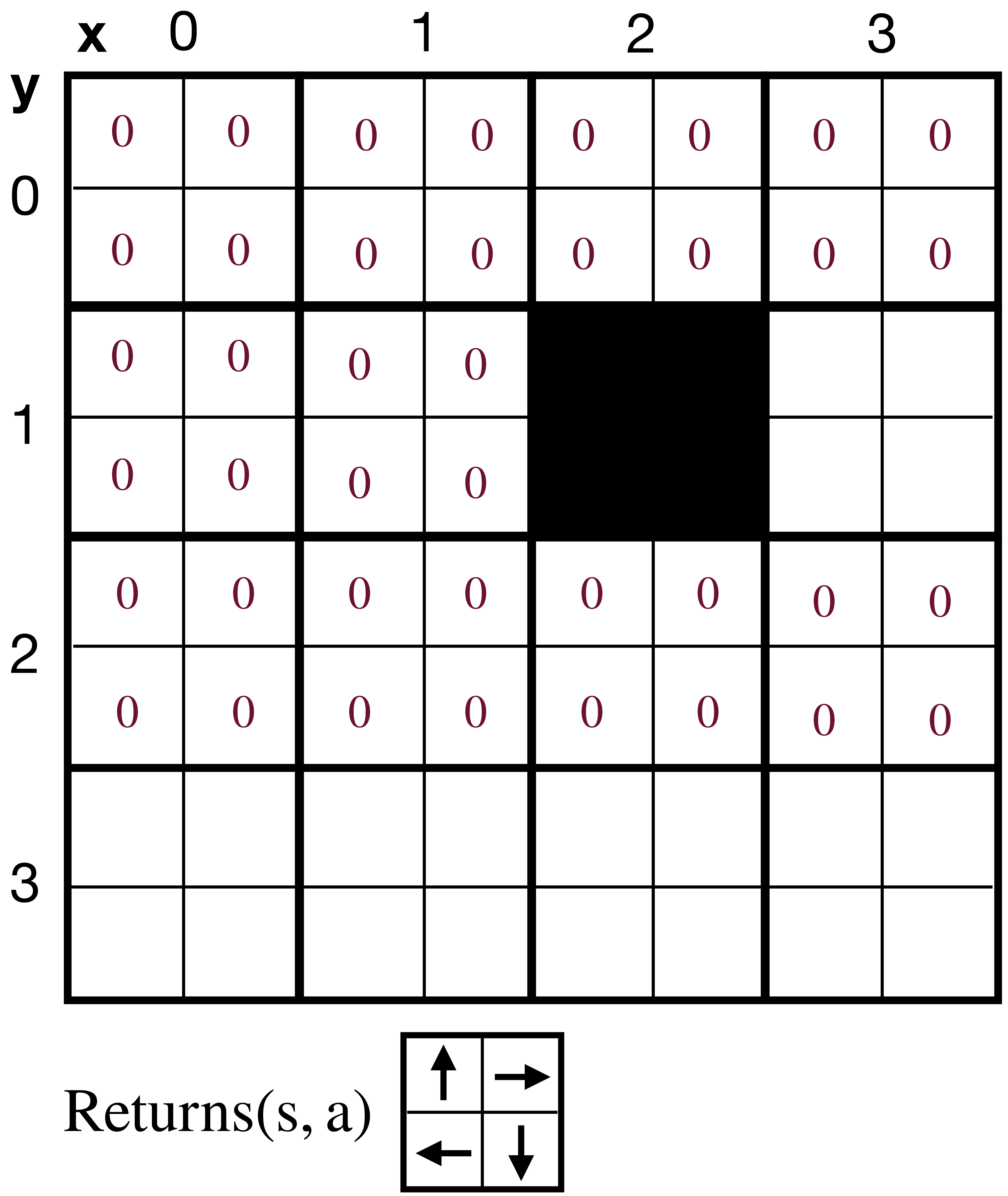
      Append $G$ to $Returns(S_t, A_t)$

      $Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

      $A^* \leftarrow \arg\max_a Q(S_t, a)$     (with ties broken arbitrarily)

      For all $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$



# Model free RL

## Example 1: On-Policy MC Algorithm

**On-policy first-visit MC control (for $\varepsilon$-soft policies), estimates $\pi \approx \pi_*$**

Algorithm parameter: small $\varepsilon > 0$

Initialize:

$\pi \leftarrow$ an arbitrary $\varepsilon$-soft policy

$Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

$Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Repeat forever (for each episode):

Generate an episode following $\pi$: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless the pair $S_t, A_t$ appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

Append $G$ to $Returns(S_t, A_t)$

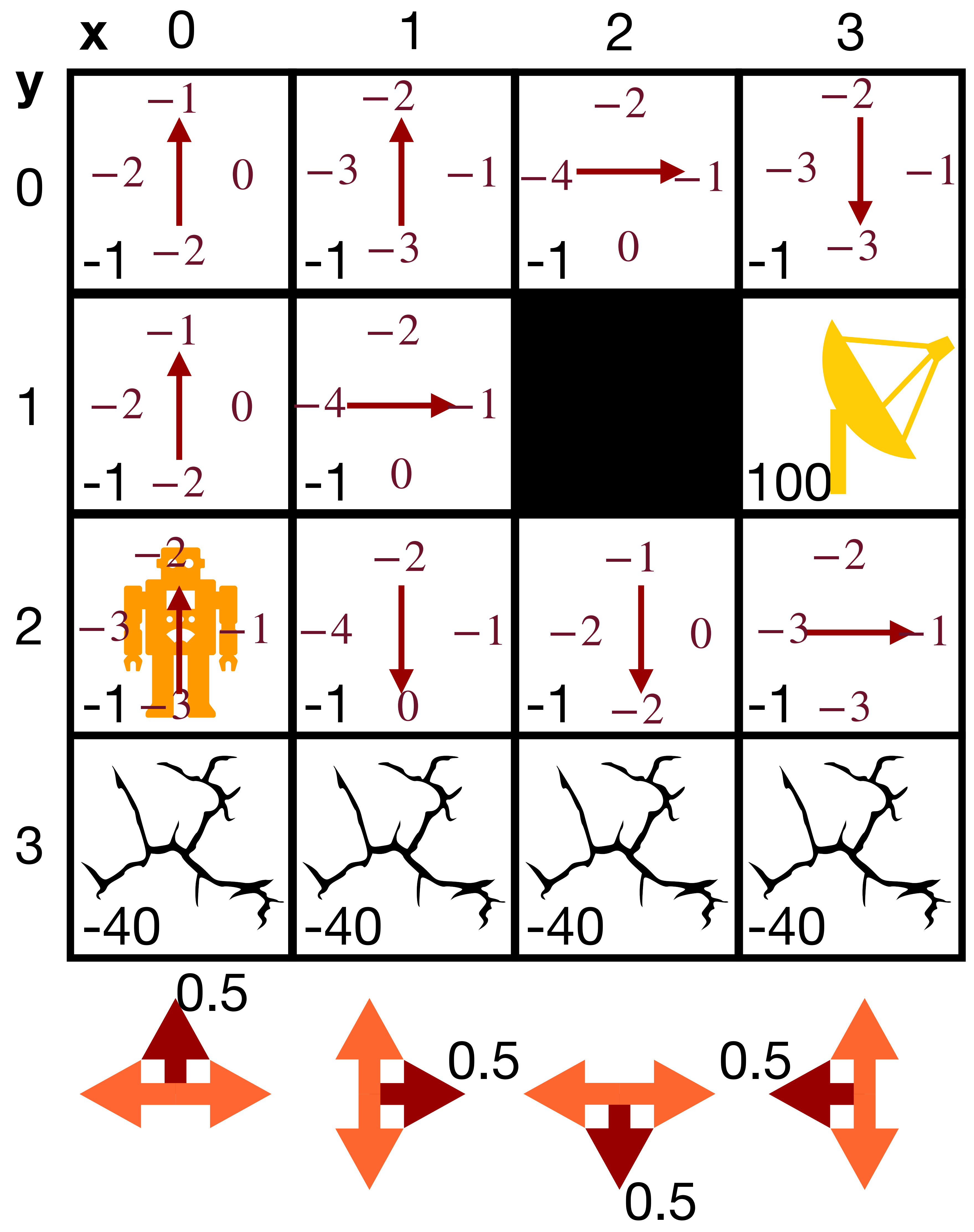
$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)

For all $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$

$\tau_\pi = []$



# Model free RL

## Example 1: On-Policy MC Algorithm

**On-policy first-visit MC control (for $\varepsilon$-soft policies), estimates $\pi \approx \pi_*$**

Algorithm parameter: small $\varepsilon > 0$

Initialize:

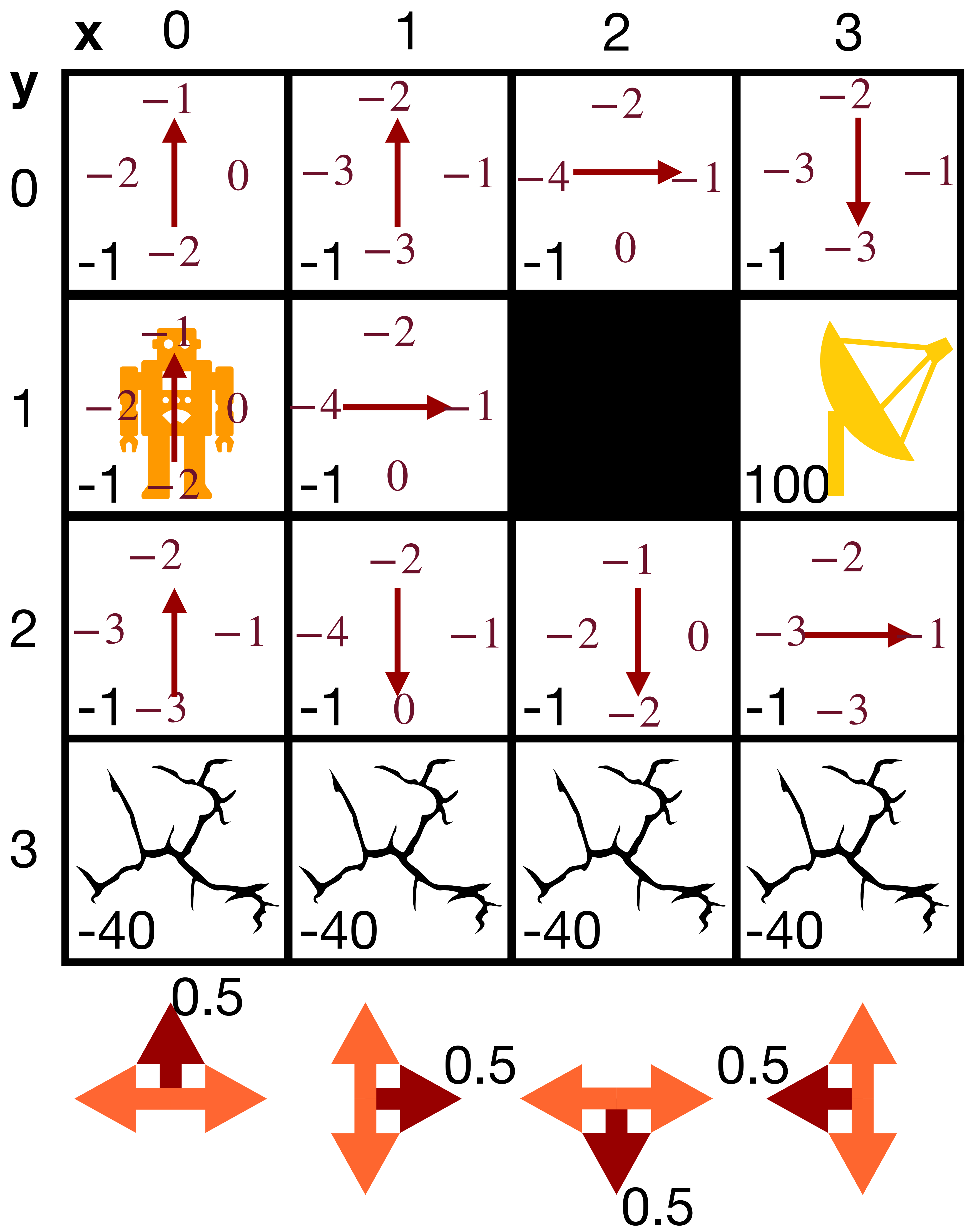
- $\pi \leftarrow$ an arbitrary $\varepsilon$-soft policy
- $Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$
- $Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Repeat forever (for each episode):

- Generate an episode following $\pi$: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
- $G \leftarrow 0$
- Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
  - $G \leftarrow \gamma G + R_{t+1}$
  - Unless the pair $S_t, A_t$ appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:
    - Append $G$ to $Returns(S_t, A_t)$
    - $Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$
    - $A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)
    - For all $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$

$$\tau_\pi = [(0,2), \text{UP}, -1]$$



# Model free RL

## Example 1: On-Policy MC Algorithm

**On-policy first-visit MC control (for $\varepsilon$-soft policies), estimates $\pi \approx \pi_*$**

Algorithm parameter: small $\varepsilon > 0$

Initialize:

$\pi \leftarrow$ an arbitrary $\varepsilon$-soft policy

$Q(s, a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

$Returns(s, a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Repeat forever (for each episode):

Generate an episode following $\pi$: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless the pair $S_t, A_t$ appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

Append $G$ to $Returns(S_t, A_t)$

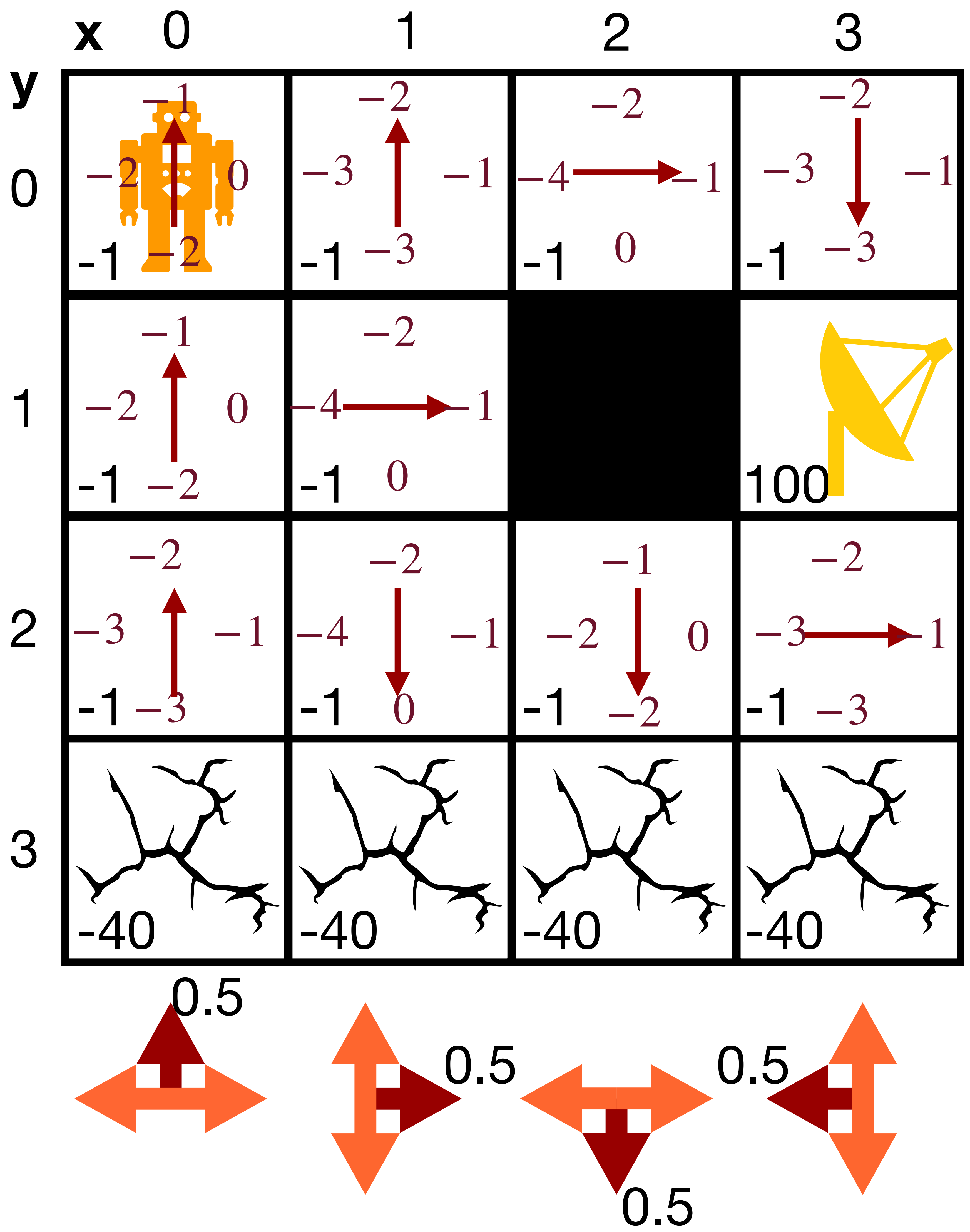
$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)

For all $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$

$$\tau_\pi = [(0,2), \text{UP}, -1], [(0,1), \text{UP}, -1]$$



# Model free RL

## Example 1: On-Policy MC Algorithm

**On-policy first-visit MC control (for $\varepsilon$-soft policies), estimates $\pi \approx \pi_*$**

Algorithm parameter: small $\varepsilon > 0$

Initialize:

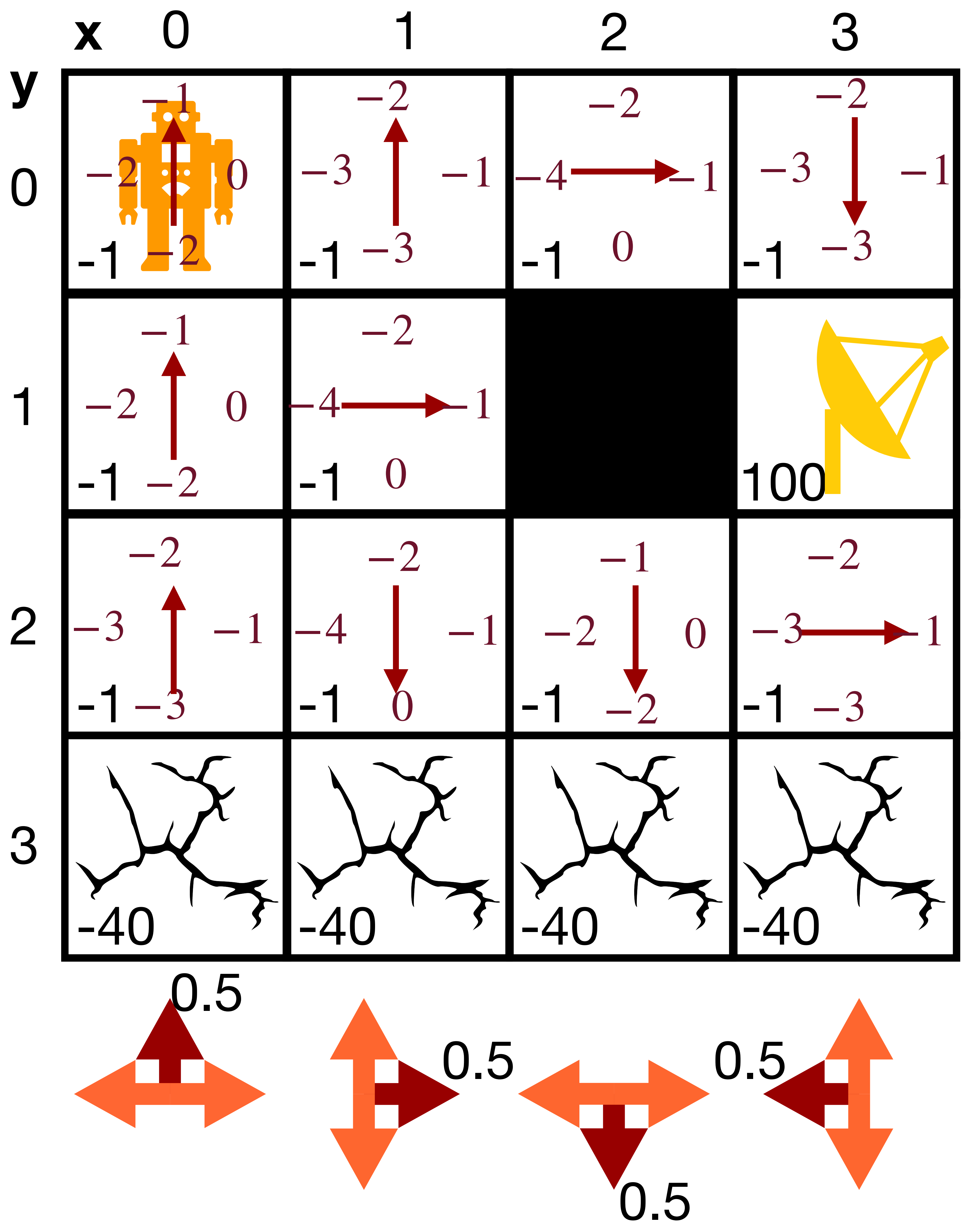
- $\pi \leftarrow$ an arbitrary $\varepsilon$-soft policy
- $Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$
- $Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Repeat forever (for each episode):

- Generate an episode following $\pi$: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
- $G \leftarrow 0$
- Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
  - $G \leftarrow \gamma G + R_{t+1}$
  - Unless the pair $S_t, A_t$ appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:
    - Append $G$ to $Returns(S_t, A_t)$
    - $Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$
    - $A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)
    - For all $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$

$$\tau_\pi = [(0,2), \text{UP}, -1], [(0,1), \text{UP}, -1], [(0,0), \text{UP}, -1]$$



# Model free RL

## Example 1: On-Policy MC Algorithm

**On-policy first-visit MC control (for $\varepsilon$-soft policies), estimates $\pi \approx \pi_*$**

Algorithm parameter: small $\varepsilon > 0$

Initialize:

$\pi \leftarrow$ an arbitrary $\varepsilon$-soft policy

$Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

$Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Repeat forever (for each episode):

Generate an episode following $\pi$: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless the pair $S_t, A_t$ appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

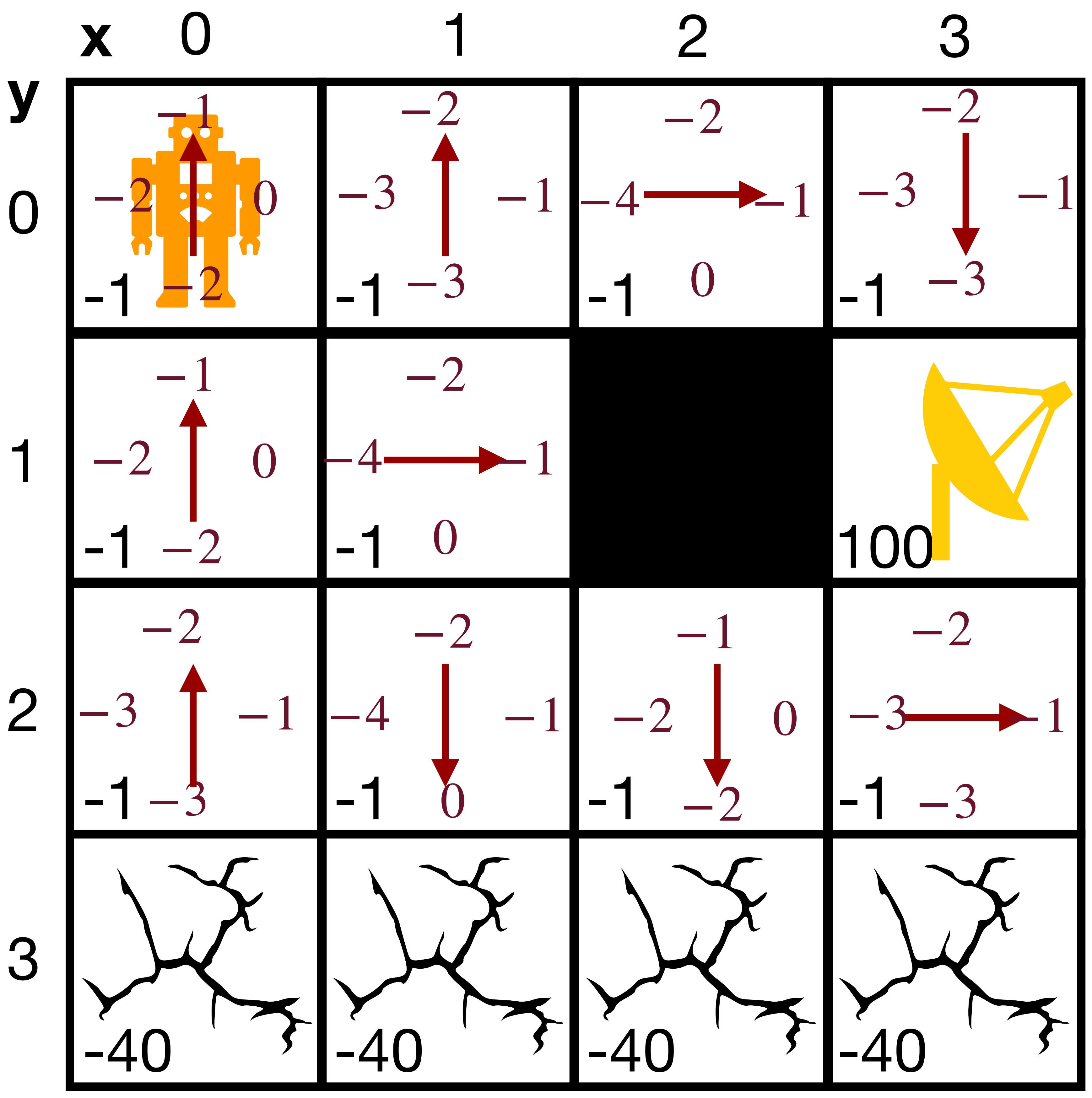
Append $G$ to $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

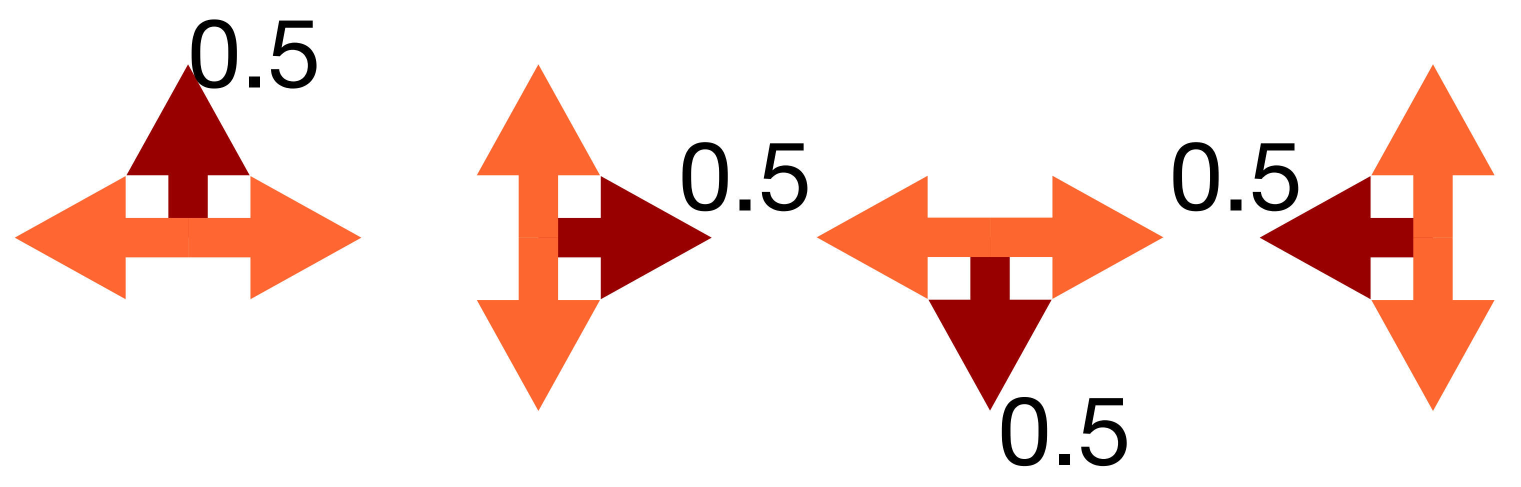
$A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)

For all $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$




$\tau_\pi = [(0,2), \text{UP}, -1], [(0,1), \text{UP}, -1], [(0,0), \text{UP}, -1], [(0,0), \text{UP}, -1]$



# Model free RL

## Example 1: On-Policy MC Algorithm

**On-policy first-visit MC control (for $\varepsilon$-soft policies), estimates $\pi \approx \pi_*$**

Algorithm parameter: small $\varepsilon > 0$

Initialize:

$\pi \leftarrow$ an arbitrary $\varepsilon$-soft policy

$Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

$Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Repeat forever (for each episode):

Generate an episode following $\pi$: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless the pair $S_t, A_t$ appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

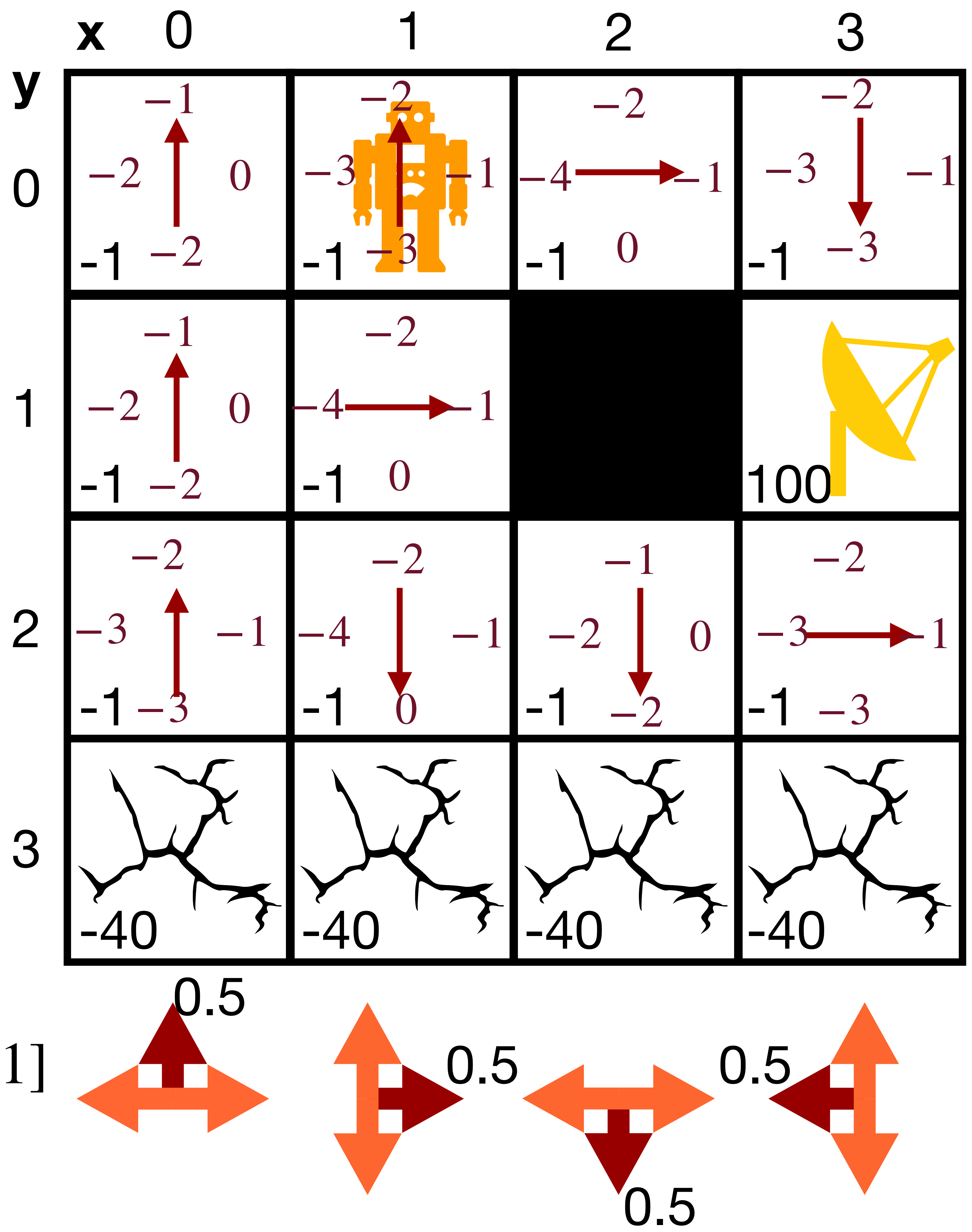
Append $G$ to $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)

For all $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$




$\tau_\pi = [(0,2), \text{UP}, -1], [(0,1), \text{UP}, -1], [(0,0), \text{UP}, -1], [(0,0), \text{UP}, -1]$
$[(0,0), \text{UP}, -1]$

# Model free RL

## Example 1: On-Policy MC Algorithm

**On-policy first-visit MC control (for $\varepsilon$-soft policies), estimates $\pi \approx \pi_*$**

Algorithm parameter: small $\varepsilon > 0$

Initialize:

  $\pi \leftarrow$ an arbitrary $\varepsilon$-soft policy

  $Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

  $Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Repeat forever (for each episode):

  Generate an episode following $\pi$: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

  $G \leftarrow 0$

  Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

    $G \leftarrow \gamma G + R_{t+1}$

    Unless the pair $S_t, A_t$ appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

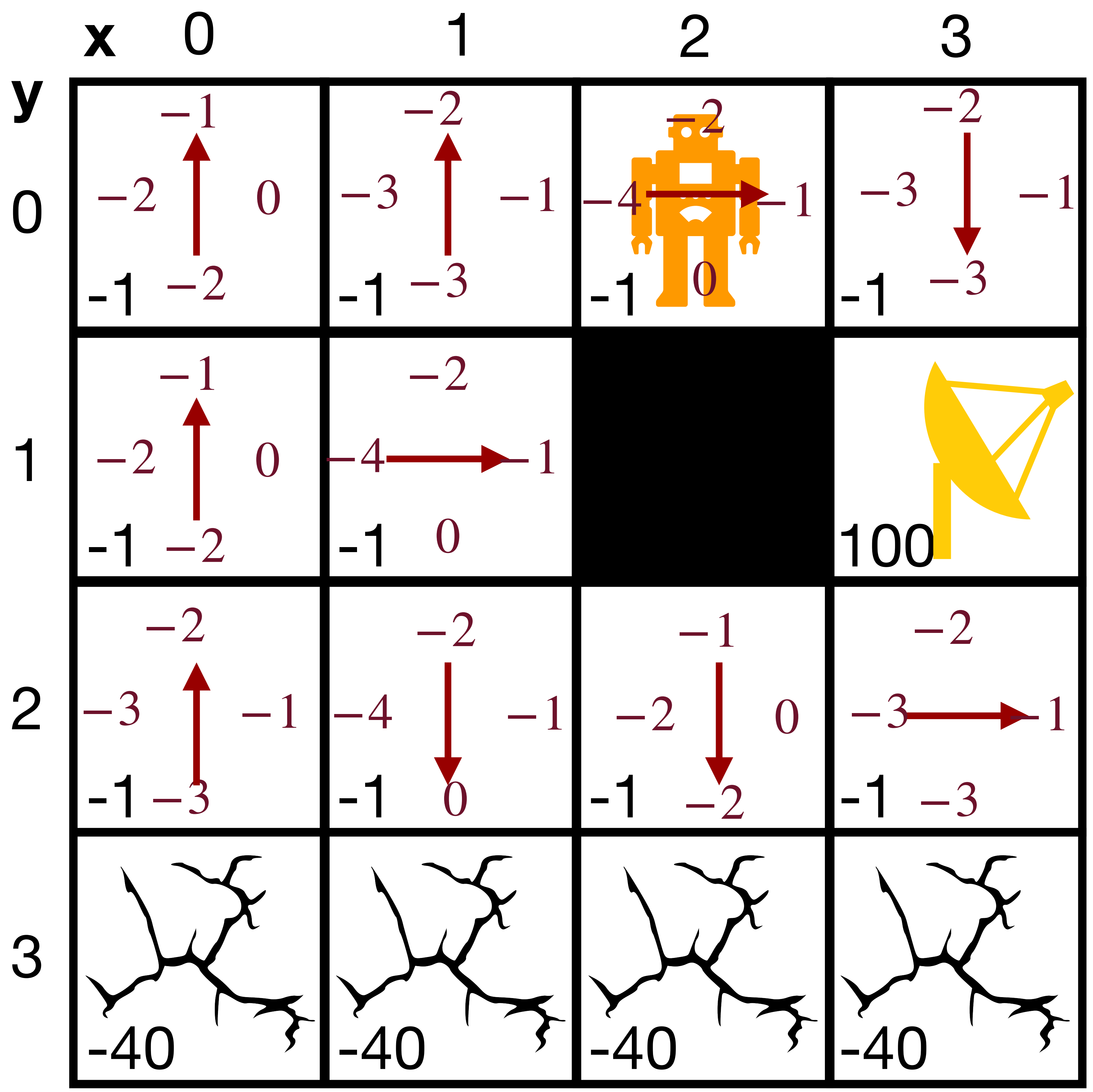
      Append $G$ to $Returns(S_t, A_t)$

      $Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

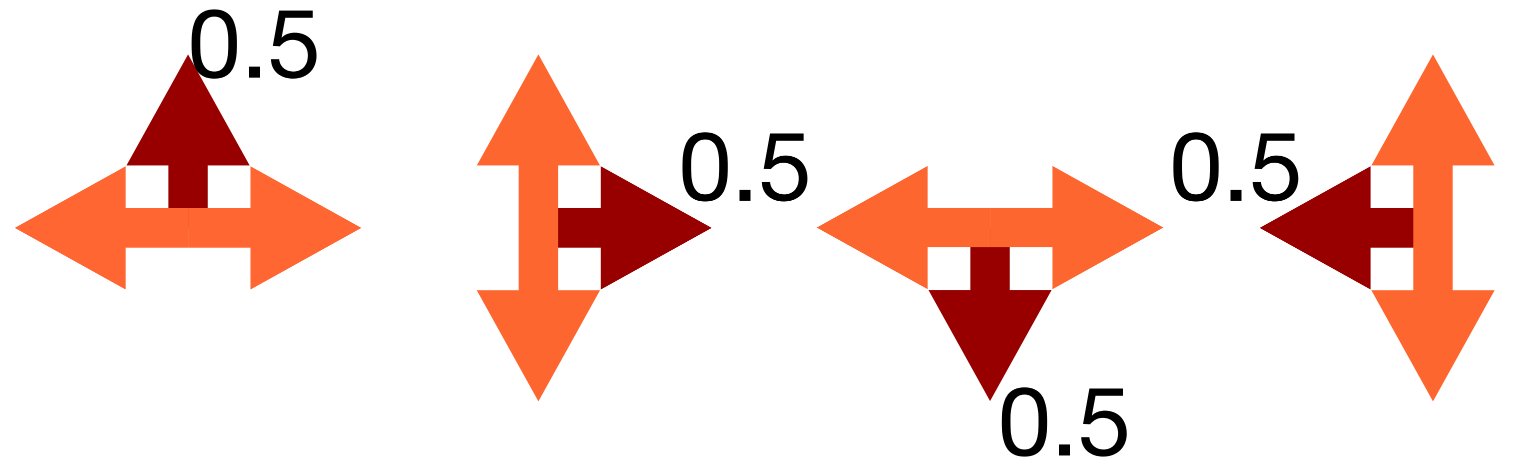
      $A^* \leftarrow \arg\max_a Q(S_t, a)$    (with ties broken arbitrarily)

      For all $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$




$\tau_\pi = [(0{,}2), \text{UP}, -1], [(0{,}1), \text{UP}, -1], [(0{,}0), \text{UP}, -1], [(0{,}0), \text{UP}, -1]$
$[(0{,}0), \text{UP}, -1], [(1{,}0), \text{UP}, -1]$



# Model free RL

## Example 1: On-Policy MC Algorithm

**On-policy first-visit MC control (for $\varepsilon$-soft policies), estimates $\pi \approx \pi_*$**

Algorithm parameter: small $\varepsilon > 0$

Initialize:

$\quad \pi \leftarrow$ an arbitrary $\varepsilon$-soft policy

$\quad Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

$\quad Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Repeat forever (for each episode):

$\quad$ Generate an episode following $\pi$: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$\quad G \leftarrow 0$

$\quad$ Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$\quad\quad G \leftarrow \gamma G + R_{t+1}$

$\quad\quad$ Unless the pair $S_t, A_t$ appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

$\quad\quad\quad$ Append $G$ to $Returns(S_t, A_t)$

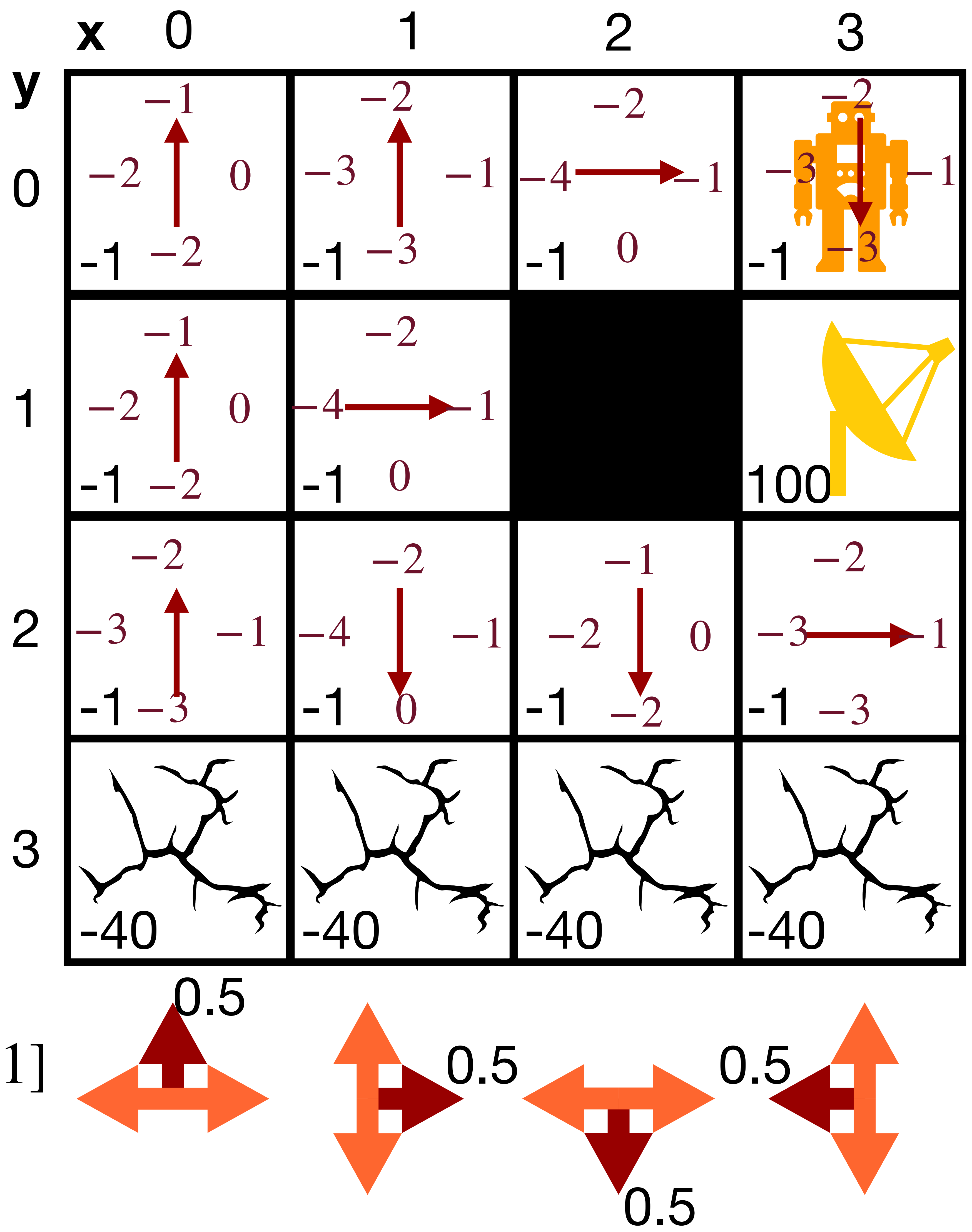
$\quad\quad\quad Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$\quad\quad\quad A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)

$\quad\quad\quad$ For all $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$

$\tau_\pi$ = [(0,2), UP, – 1], [(0,1), UP, – 1], [(0,0), UP, – 1], [(0,0), UP, – 1]
[(0,0), UP, – 1], [(1,0), UP, – 1], [(2,0), RIGHT, – 1]



# Model free RL

## Example 1: On-Policy MC Algorithm

**On-policy first-visit MC control (for $\varepsilon$-soft policies), estimates $\pi \approx \pi_*$**

Algorithm parameter: small $\varepsilon > 0$

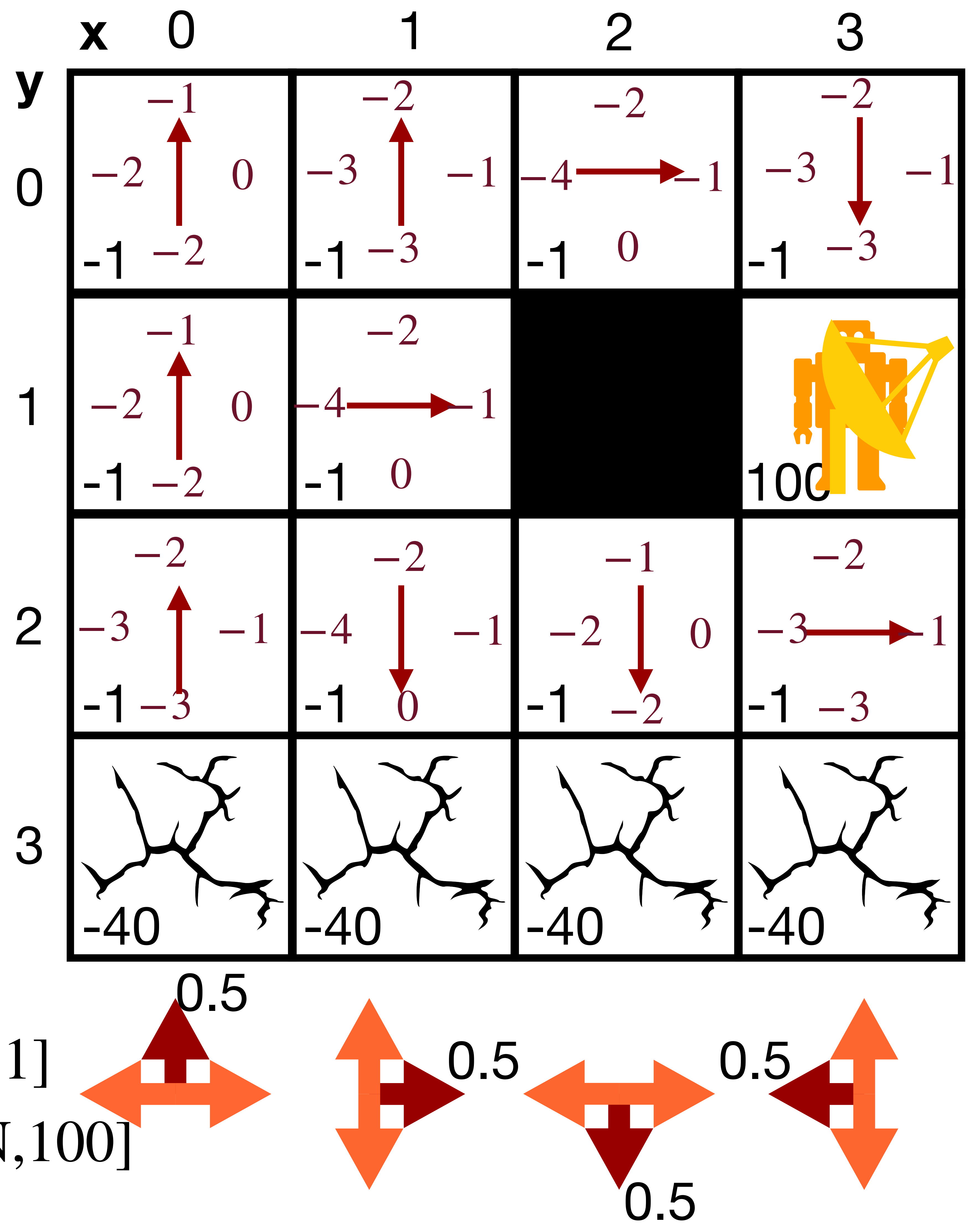
Initialize:

- $\pi \leftarrow$ an arbitrary $\varepsilon$-soft policy
- $Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$
- $Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Repeat forever (for each episode):

- Generate an episode following $\pi$: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
- $G \leftarrow 0$
- Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
  - $G \leftarrow \gamma G + R_{t+1}$
  - Unless the pair $S_t, A_t$ appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:
    - Append $G$ to $Returns(S_t, A_t)$
    - $Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$
    - $A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)
    - For all $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$




$\tau_\pi = [(0,2), \text{UP}, -1], [(0,1), \text{UP}, -1], [(0,0), \text{UP}, -1], [(0,0), \text{UP}, -1]$
$[(0,0), \text{UP}, -1], [(1,0), \text{UP}, -1], [(2,0), \text{RIGHT}, -1], [(3,0), \text{DOWN}, 100]$

# Model free RL

## Example 1: On-Policy MC Algorithm

**On-policy first-visit MC control (for $\varepsilon$-soft policies), estimates $\pi \approx \pi_*$**

Algorithm parameter: small $\varepsilon > 0$

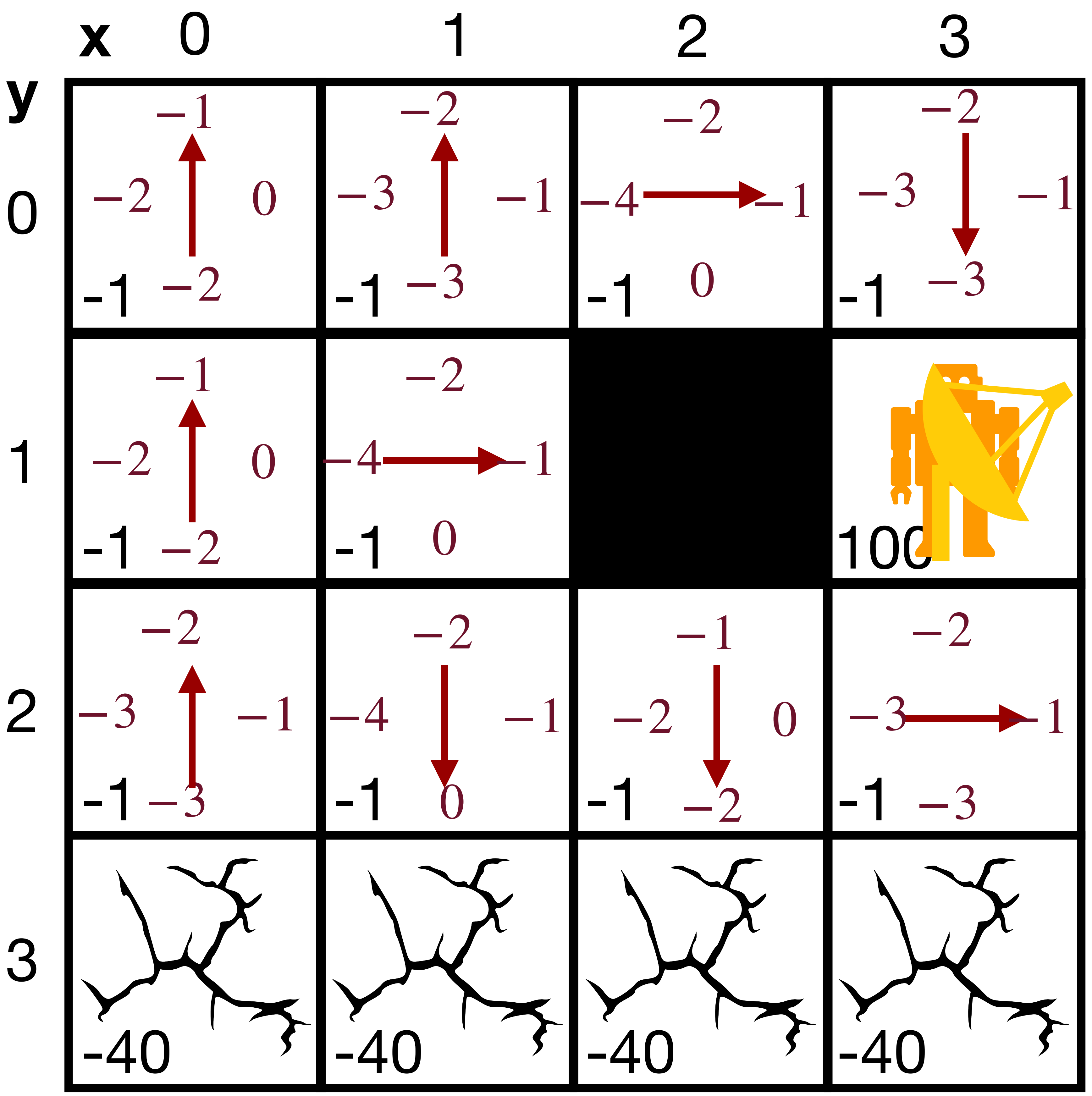
Initialize:

- $\pi \leftarrow$ an arbitrary $\varepsilon$-soft policy
- $Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$
- $Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Repeat forever (for each episode):

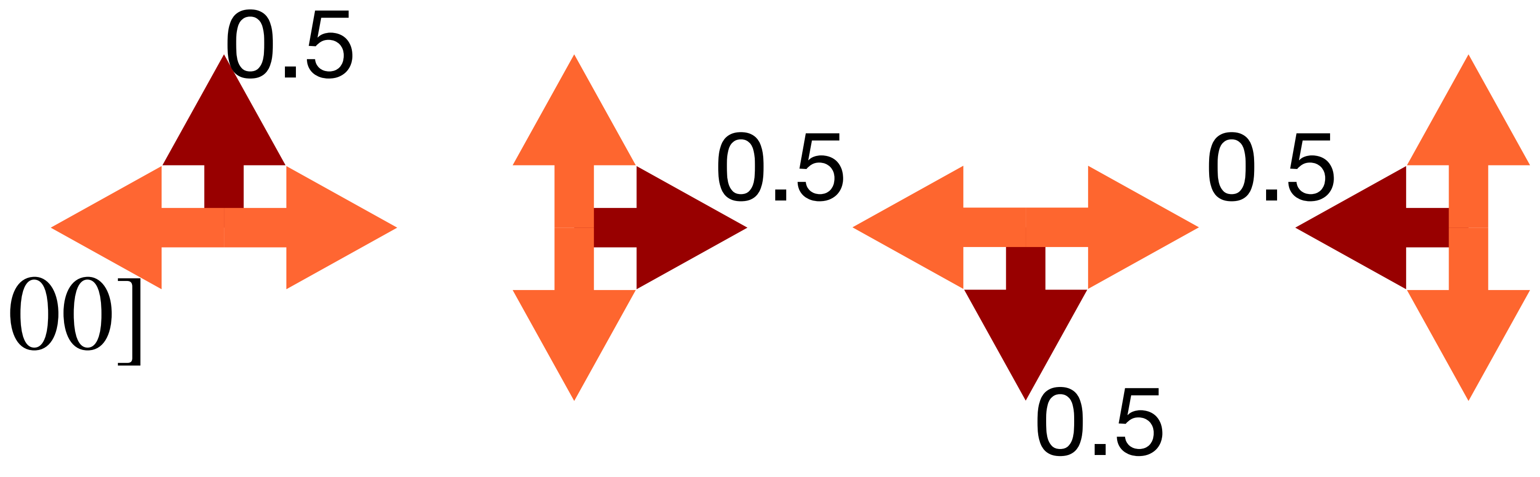
- Generate an episode following $\pi$: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
- $G \leftarrow 0$
- Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
  - $G \leftarrow \gamma G + R_{t+1}$
  - Unless the pair $S_t, A_t$ appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:
    - Append $G$ to $Returns(S_t, A_t)$
    - $Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$
    - $A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)
    - For all $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$




$\tau_\pi = [(0,2), \text{UP}, -1], [(0,1), \text{UP}, -1], [(0,0), \text{UP}, -1], [(0,0), \text{UP}, -1]$
$[(0,0), \text{UP}, -1], [(1,0), \text{UP}, -1], [(2,0), \text{RIGHT}, -1], [(3,0), \text{DOWN}, 100]$

$G = 0$



# Model free RL

## Example 1: On-Policy MC Algorithm

**On-policy first-visit MC control (for $\varepsilon$-soft policies), estimates $\pi \approx \pi_*$**

Algorithm parameter: small $\varepsilon > 0$

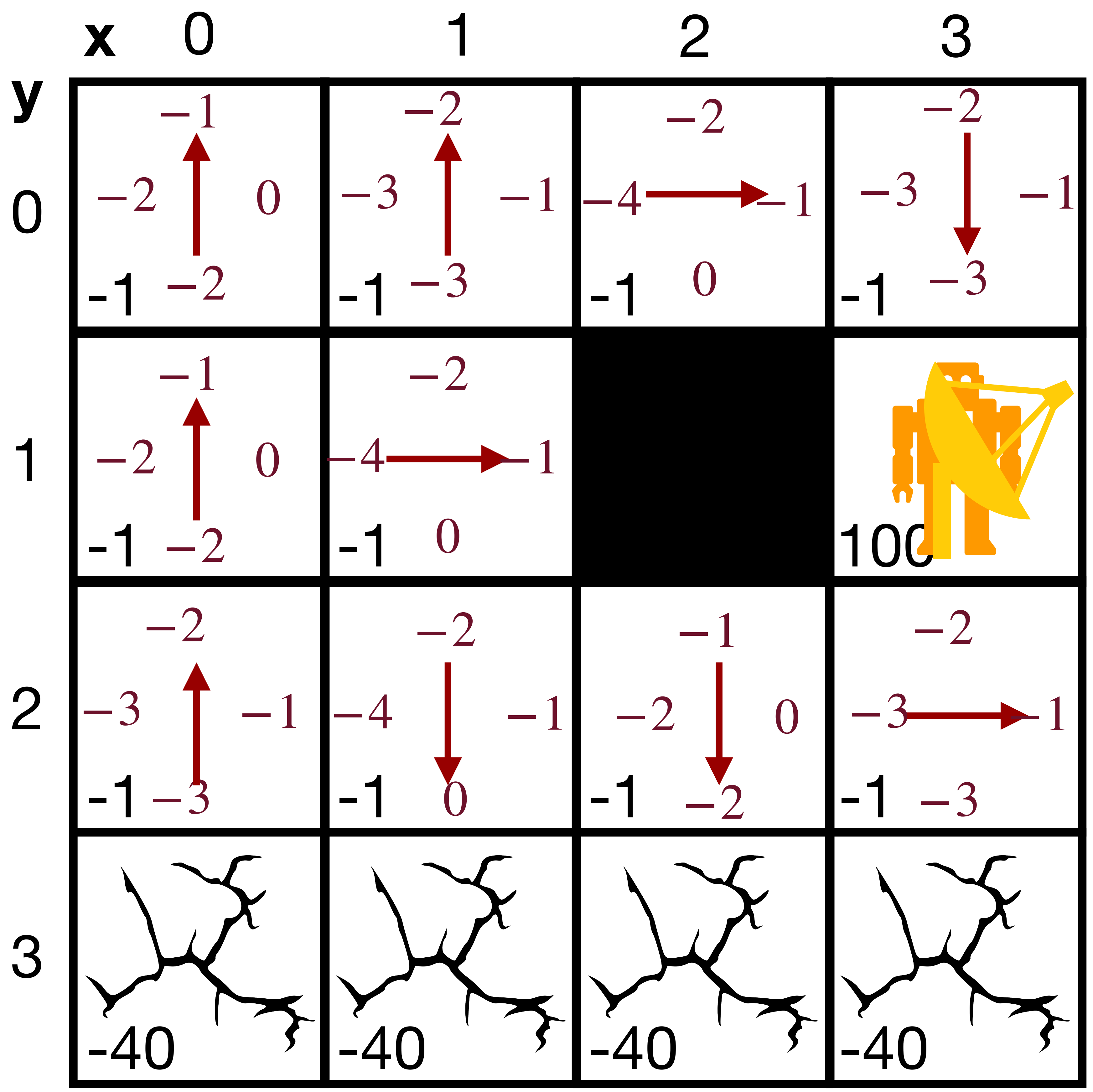
Initialize:

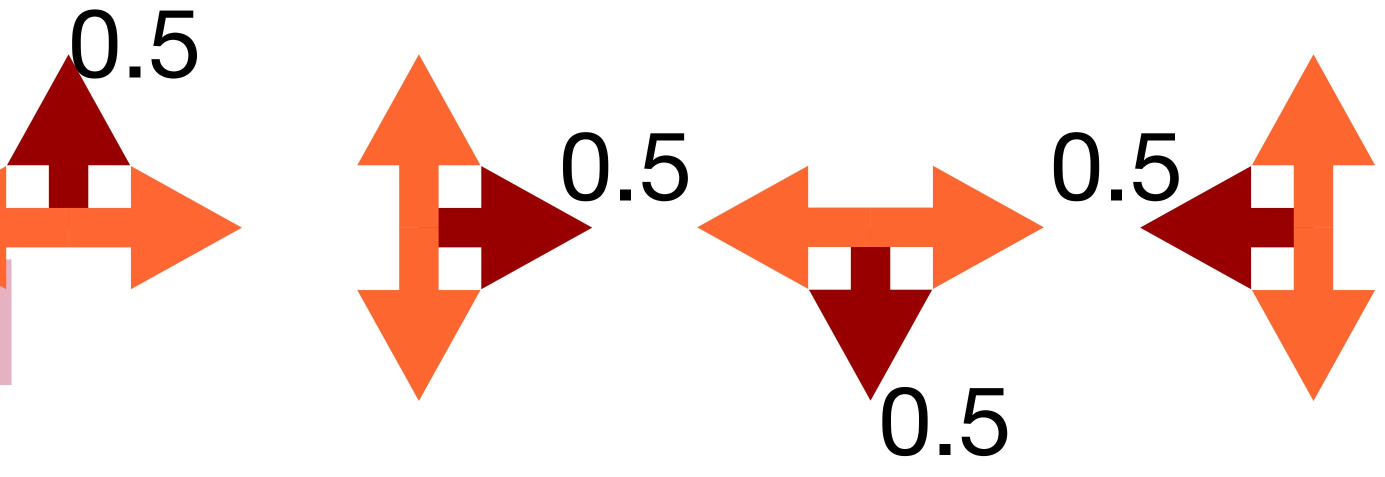
- $\pi \leftarrow$ an arbitrary $\varepsilon$-soft policy
- $Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$
- $Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Repeat forever (for each episode):

- Generate an episode following $\pi$: $S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T$
- $G \leftarrow 0$
- Loop for each step of episode, $t = T-1, T-2, \dots, 0$:
  - $G \leftarrow \gamma G + R_{t+1}$
  - Unless the pair $S_t, A_t$ appears in $S_0, A_0, S_1, A_1 \dots, S_{t-1}, A_{t-1}$:
    - Append $G$ to $Returns(S_t, A_t)$
    - $Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$
    - $A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)
    - For all $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$




$\tau_\pi = [(0{,}2), \text{UP}, -1], [(0{,}1), \text{UP}, -1], [(0{,}0), \text{UP}, -1], [(0{,}0), \text{UP}, -1]$
$[(0{,}0), \text{UP}, -1], [(1{,}0), \text{UP}, -1], [(2{,}0), \text{RIGHT}, -1], [(3{,}0), \text{DOWN}, 100]$

$G = 100$

# Model free RL

## Example 1: On-Policy MC Algorithm

**On-policy first-visit MC control (for $\varepsilon$-soft policies), estimates $\pi \approx \pi_*$**

Algorithm parameter: small $\varepsilon > 0$

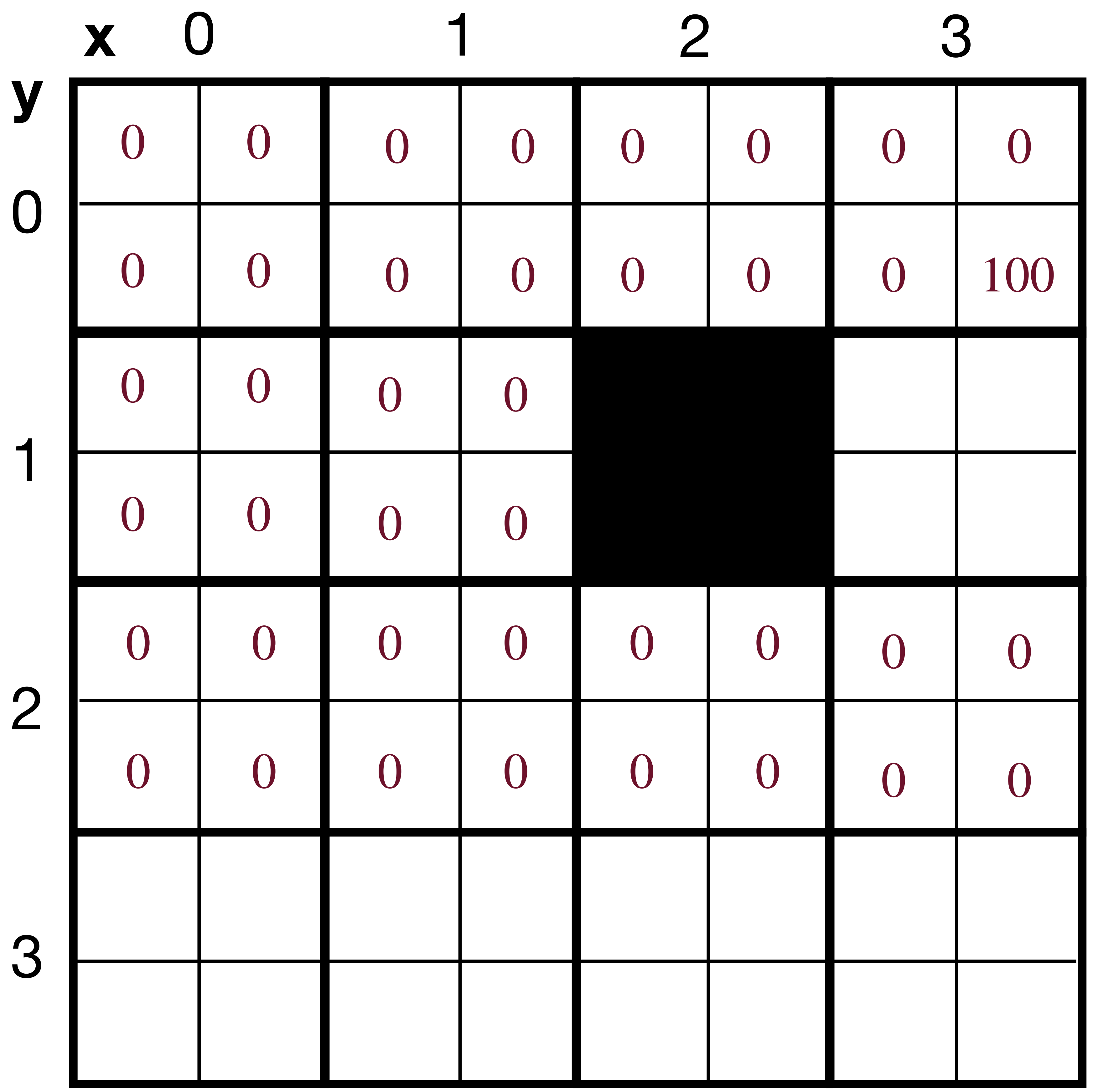
Initialize:

- $\pi \leftarrow$ an arbitrary $\varepsilon$-soft policy
- $Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$
- $Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

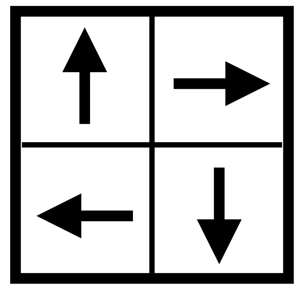
Repeat forever (for each episode):

- Generate an episode following $\pi$: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
- $G \leftarrow 0$
- Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
  - $G \leftarrow \gamma G + R_{t+1}$
  - Unless the pair $S_t, A_t$ appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:
    - Append $G$ to $Returns(S_t, A_t)$
    - $Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$
    - $A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)
    - For all $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$




$\tau_\pi = [(0,2), \text{UP}, -1], [(0,1), \text{UP}, -1], [(0,0), \text{UP}, -1], [(0,0), \text{UP}, -1]$ Returns(s, a)



$[(0,0), \text{UP}, -1], [(1,0), \text{UP}, -1], [(2,0), \text{RIGHT}, -1], [(3,0), \text{DOWN}, 100]$

$G = 100$

# Model free RL

## Example 1: On-Policy MC Algorithm

**On-policy first-visit MC control (for $\varepsilon$-soft policies), estimates $\pi \approx \pi_*$**

Algorithm parameter: small $\varepsilon > 0$

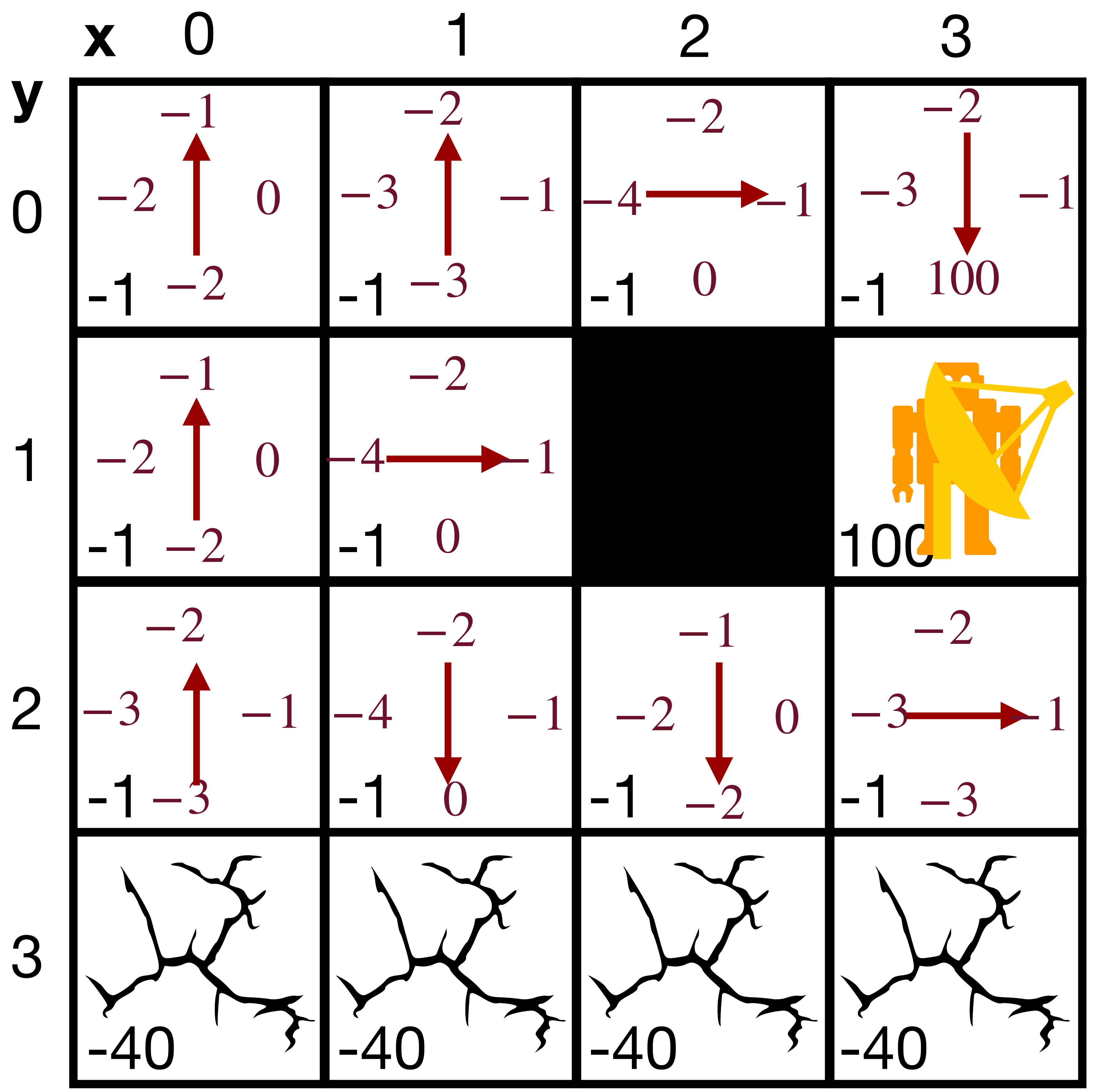
Initialize:

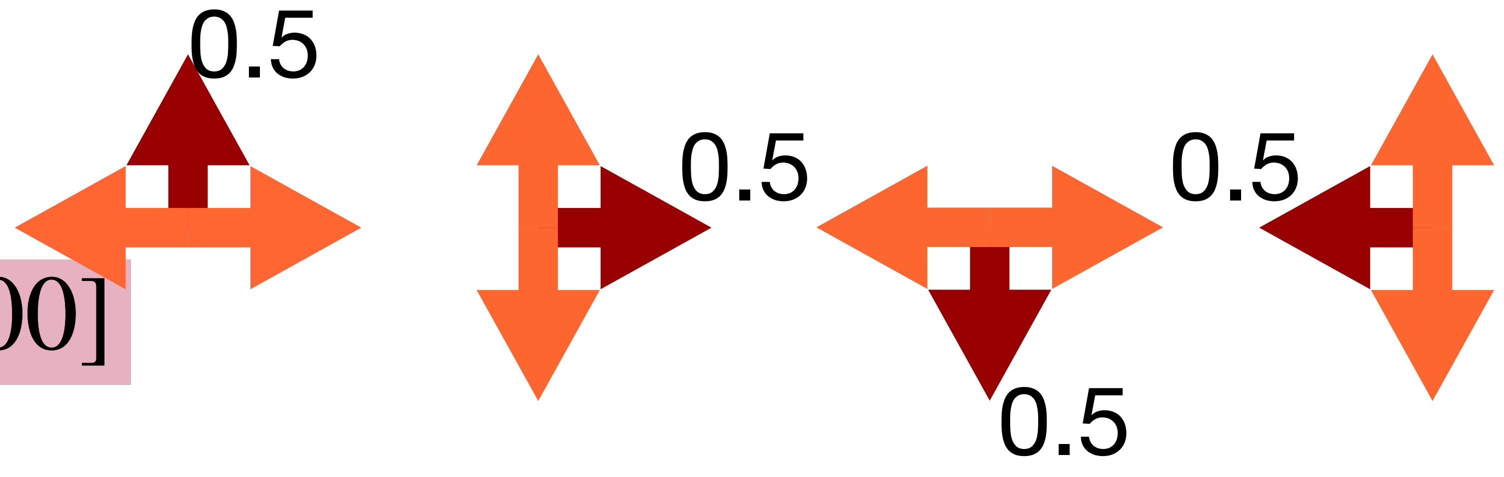
$\pi \leftarrow$ an arbitrary $\varepsilon$-soft policy
$Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$
$Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Repeat forever (for each episode):

Generate an episode following $\pi$: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
$G \leftarrow 0$
Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
$G \leftarrow \gamma G + R_{t+1}$
Unless the pair $S_t, A_t$ appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:
Append $G$ to $Returns(S_t, A_t)$
$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$
$A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)
For all $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$




$\tau_\pi$ = [(0,2), UP, − 1], [(0,1), UP, − 1], [(0,0), UP, − 1], [(0,0), UP, − 1]
[(0,0), UP, − 1], [(1,0), UP, − 1], [(2,0), RIGHT, − 1], [(3,0), DOWN,100]
$G = 100$

# Model free RL

## Example 1: On-Policy MC Algorithm

**On-policy first-visit MC control (for $\varepsilon$-soft policies), estimates $\pi \approx \pi_*$**

Algorithm parameter: small $\varepsilon > 0$

Initialize:

$\pi \leftarrow$ an arbitrary $\varepsilon$-soft policy

$Q(s, a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

$Returns(s, a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Repeat forever (for each episode):

Generate an episode following $\pi$: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless the pair $S_t, A_t$ appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

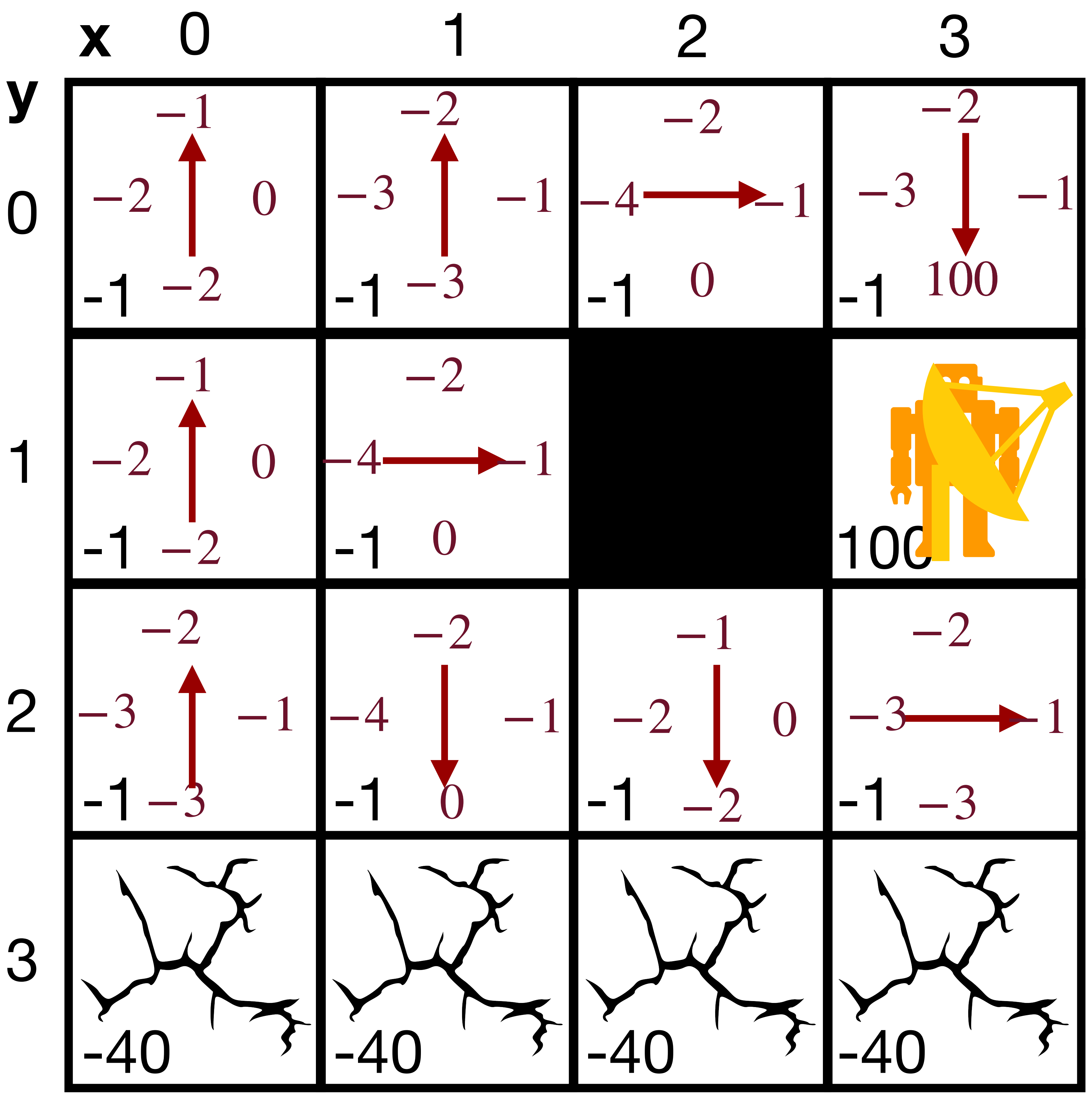
Append $G$ to $Returns(S_t, A_t)$

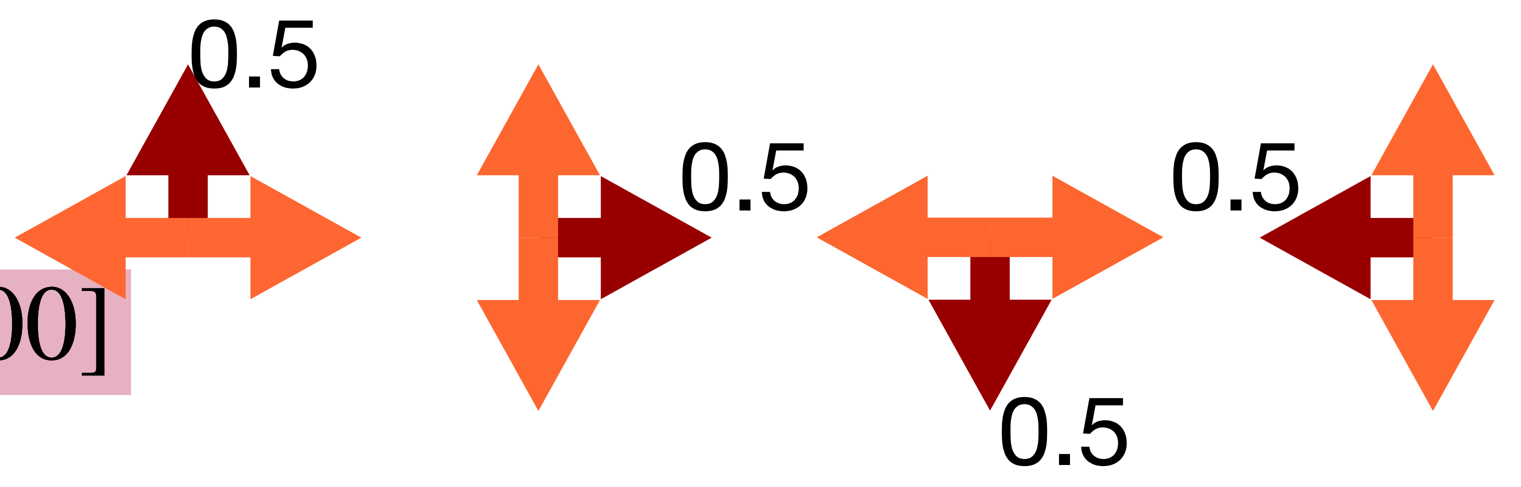
$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)

For all $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$




$\tau_\pi =$ [(0,2), UP, − 1], [(0,1), UP, − 1], [(0,0), UP, − 1], [(0,0), UP, − 1]
[(0,0), UP, − 1], [(1,0), UP, − 1], [(2,0), RIGHT, − 1], [(3,0), DOWN,100]

$G = 100 \;\; A* =$ DOWN

# Model free RL

## Example 1: On-Policy MC Algorithm

**On-policy first-visit MC control (for $\varepsilon$-soft policies), estimates $\pi \approx \pi_*$**

Algorithm parameter: small $\varepsilon > 0$

Initialize:

  $\pi \leftarrow$ an arbitrary $\varepsilon$-soft policy

  $Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

  $Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Repeat forever (for each episode):

  Generate an episode following $\pi$: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

  $G \leftarrow 0$

  Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

    $G \leftarrow \gamma G + R_{t+1}$

    Unless the pair $S_t, A_t$ appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

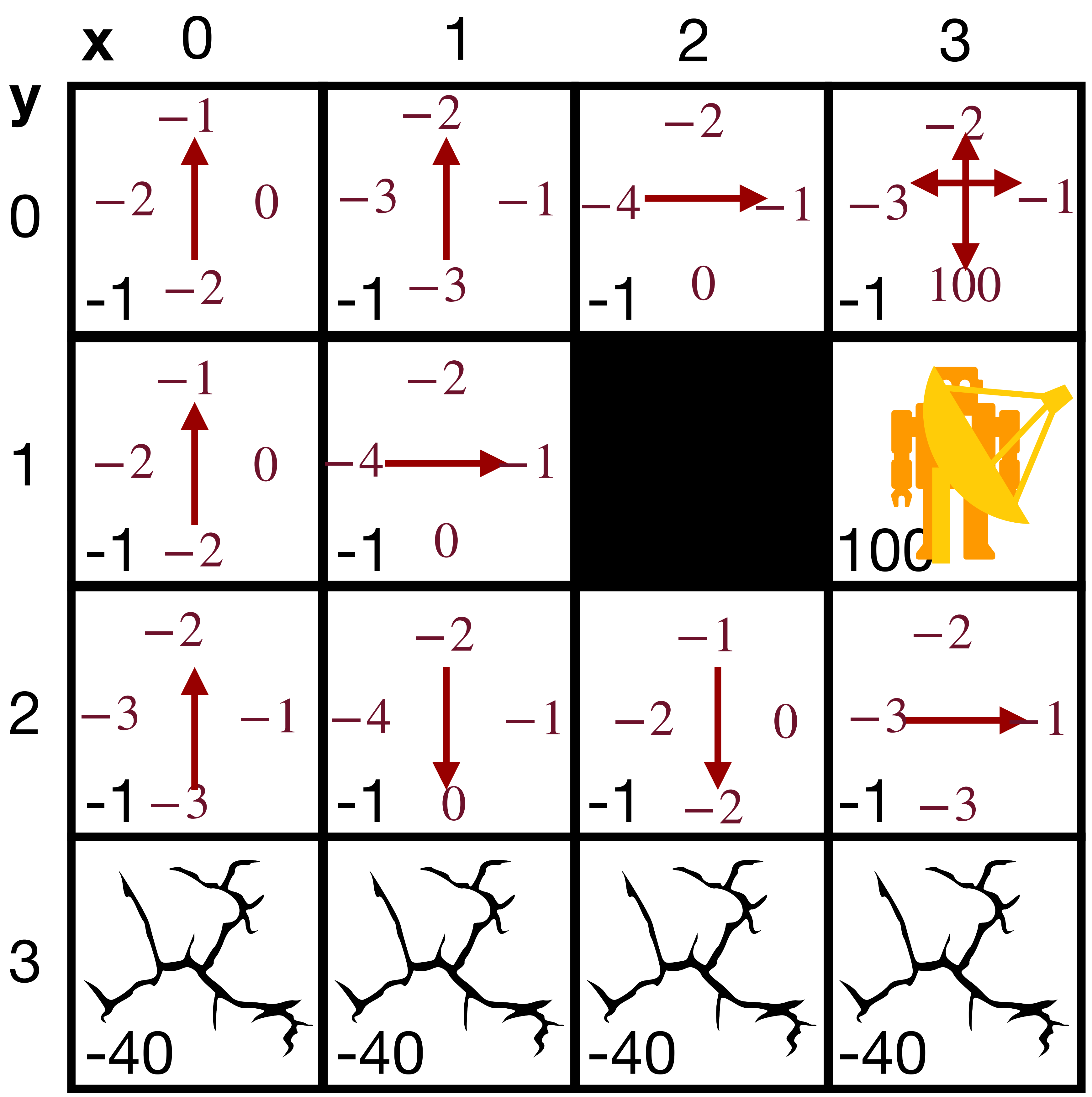
      Append $G$ to $Returns(S_t, A_t)$

      $Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

      $A^* \leftarrow \arg\max_a Q(S_t, a)$   (with ties broken arbitrarily)

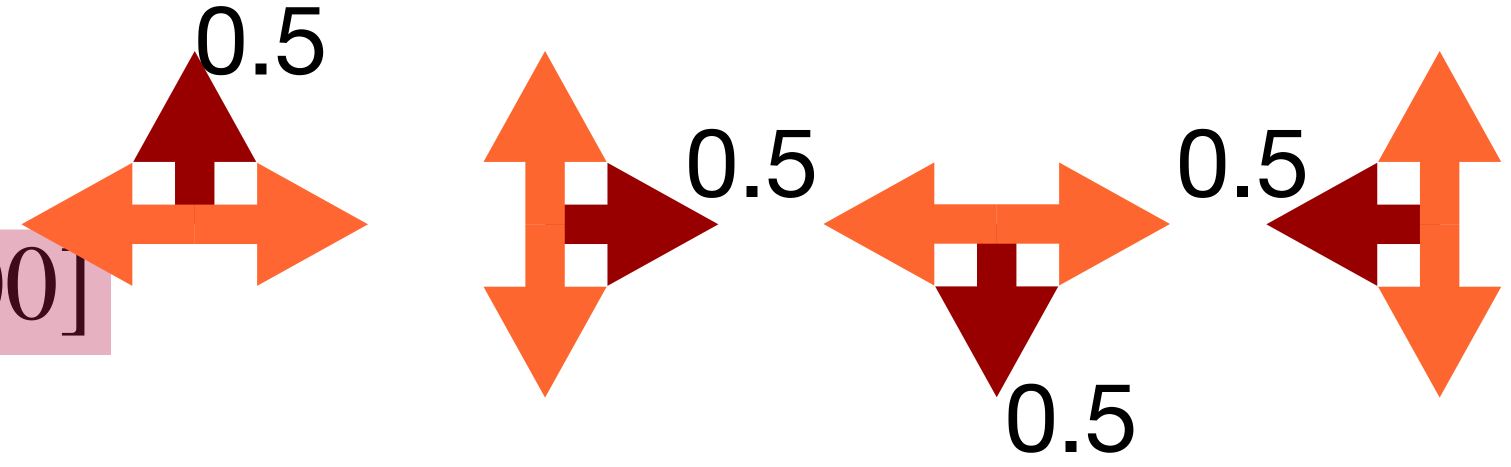
      For all $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$




$\tau_\pi$ = [(0,2), UP, – 1], [(0,1), UP, – 1], [(0,0), UP, – 1], [(0,0), UP, – 1] [(0,0), UP, – 1], [(1,0), UP, – 1], [(2,0), RIGHT, – 1], [(3,0), DOWN,100]

$G = 100$  $A* =$ DOWN



# Model free RL

## Example 1: On-Policy MC Algorithm

**On-policy first-visit MC control (for $\varepsilon$-soft policies), estimates $\pi \approx \pi_*$**

Algorithm parameter: small $\varepsilon > 0$

Initialize:

$\quad \pi \leftarrow$ an arbitrary $\varepsilon$-soft policy

$\quad Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

$\quad Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Repeat forever (for each episode):

$\quad$ Generate an episode following $\pi$: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$\quad G \leftarrow 0$

$\quad$ Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$\quad\quad G \leftarrow \gamma G + R_{t+1}$

$\quad\quad$ Unless the pair $S_t, A_t$ appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

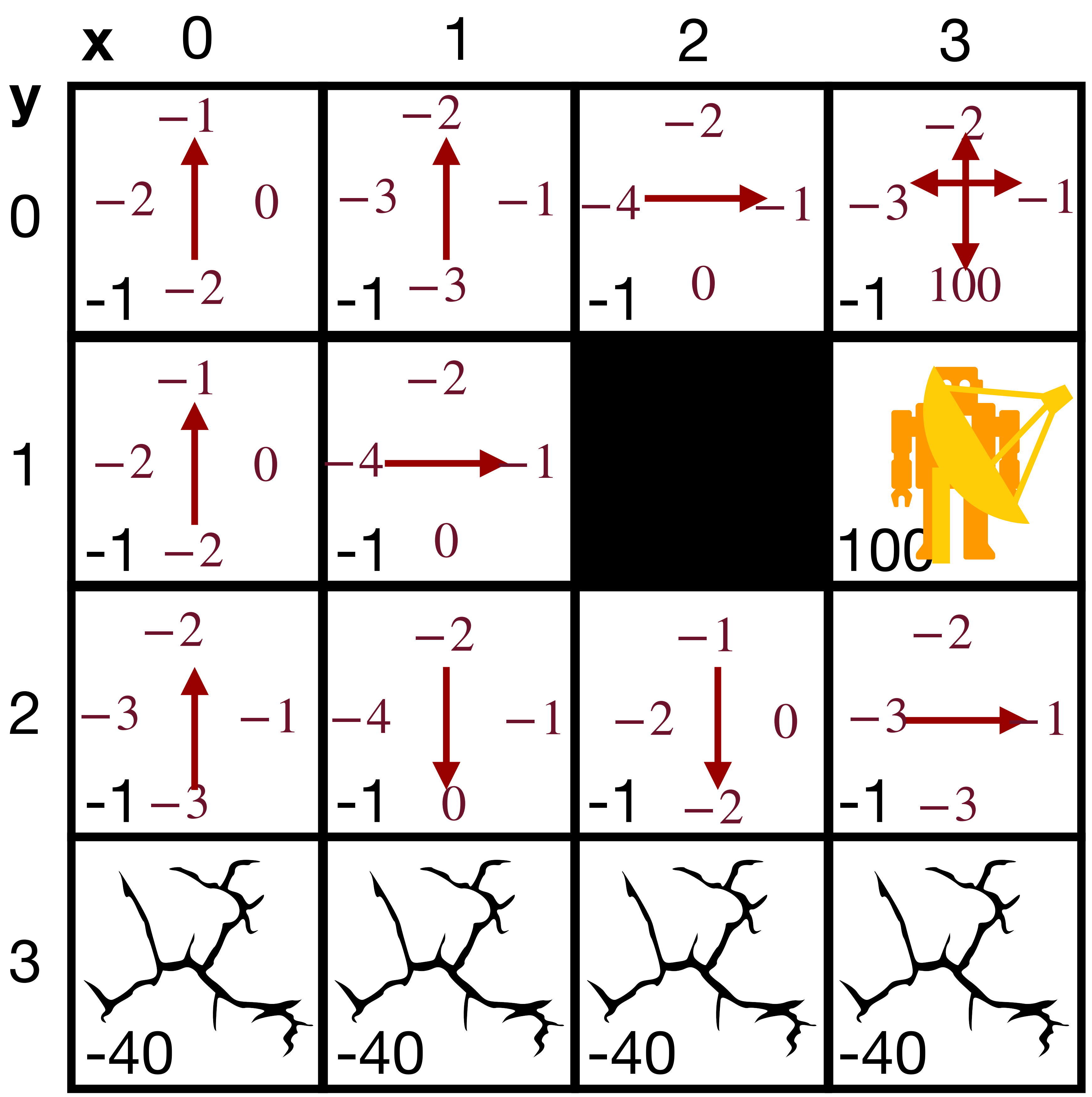
$\quad\quad\quad$ Append $G$ to $Returns(S_t, A_t)$

$\quad\quad\quad Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

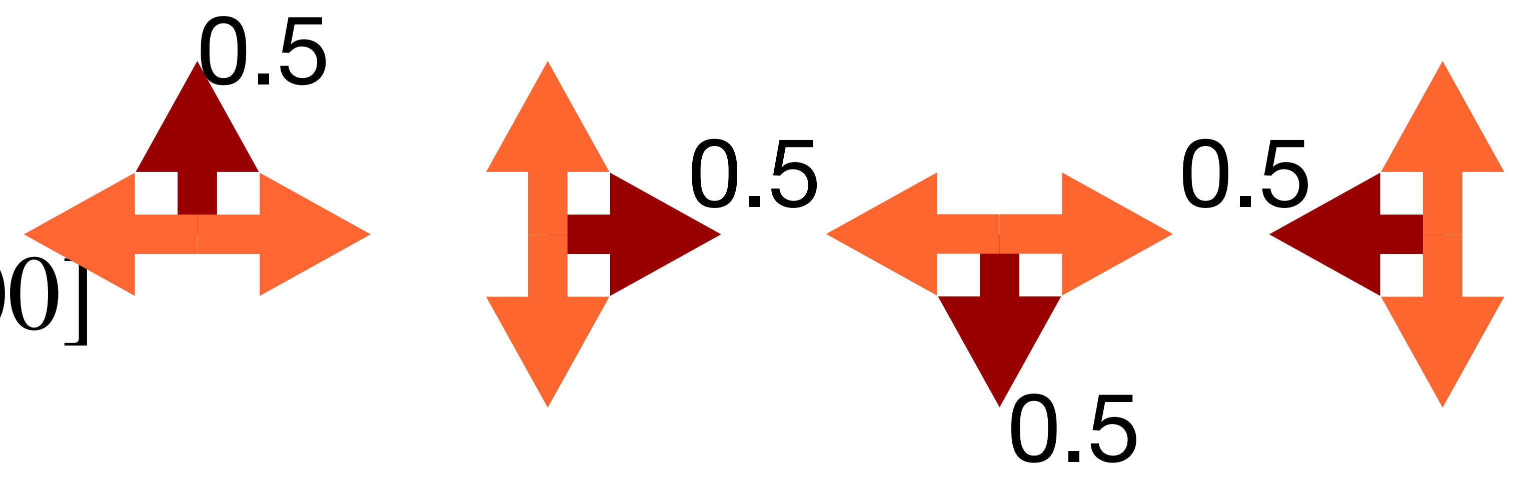
$\quad\quad\quad A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)

$\quad\quad\quad$ For all $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$




$$\tau_\pi = [(0,2), \text{UP}, -1], [(0,1), \text{UP}, -1], [(0,0), \text{UP}, -1], [(0,0), \text{UP}, -1]$$
$$[(0,0), \text{UP}, -1], [(1,0), \text{UP}, -1], [(2,0), \text{RIGHT}, -1], [(3,0), \text{DOWN}, 100]$$



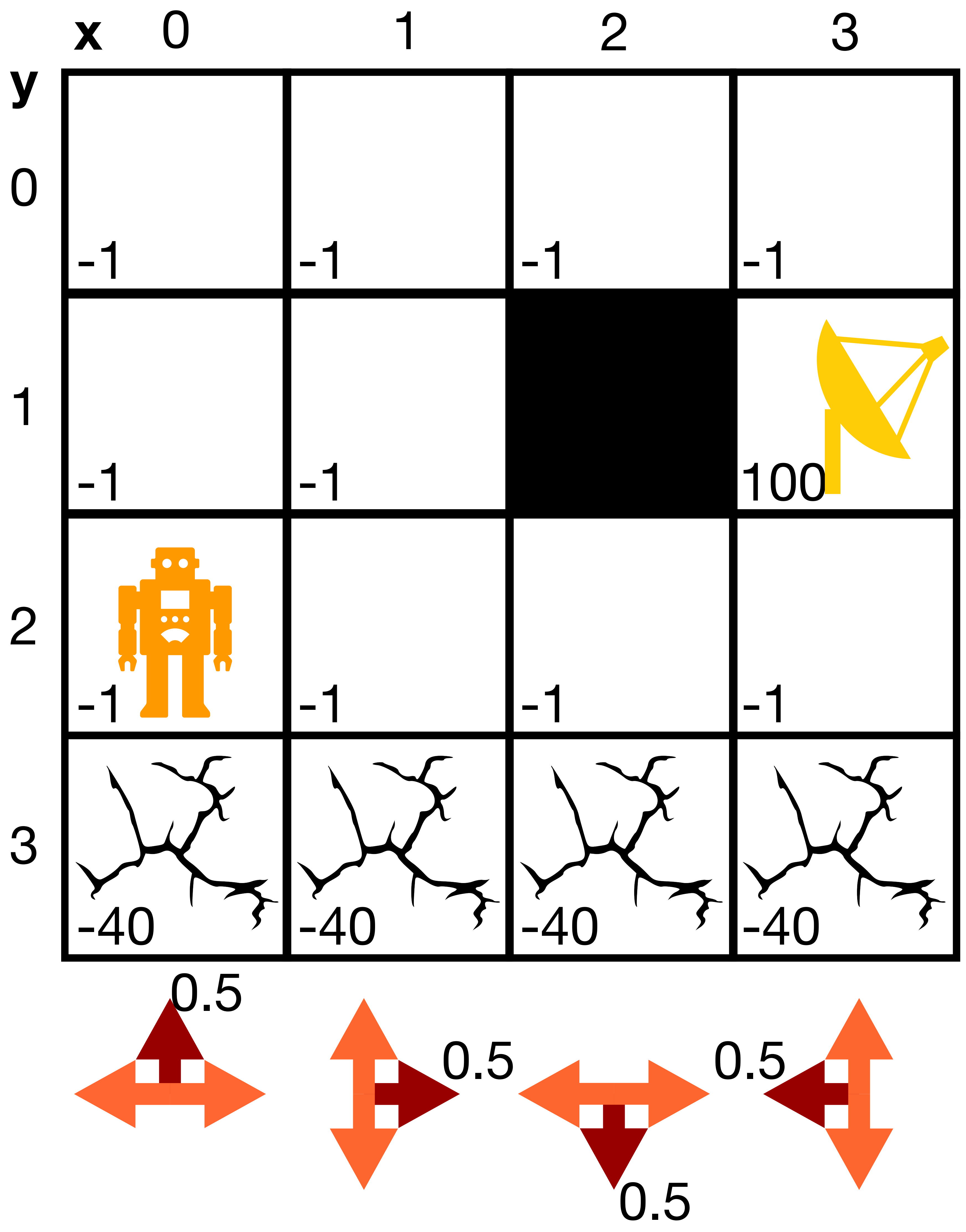
# Model free RL

## Example 2: Off-Policy TD Algorithm

**Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$**

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize $S$
  Loop for each step of episode:
    Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\varepsilon$-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



# Model free RL

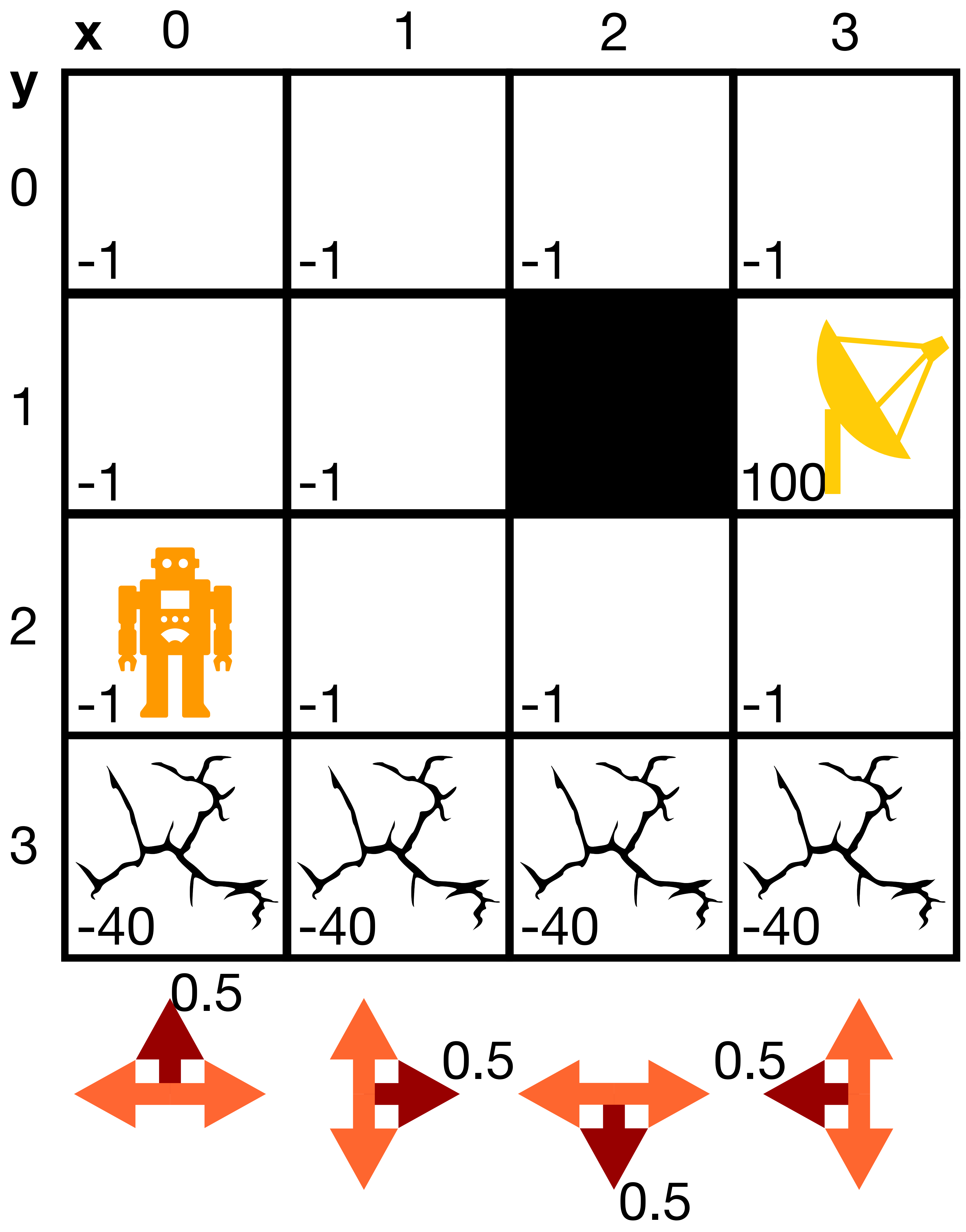
## Example 2: Off-Policy TD Algorithm

**Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$**

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize $S$
  Loop for each step of episode:
    Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\varepsilon$-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

$\alpha = 0.1 \quad \gamma = 0.9 \quad \varepsilon = 0.4$



# Model free RL

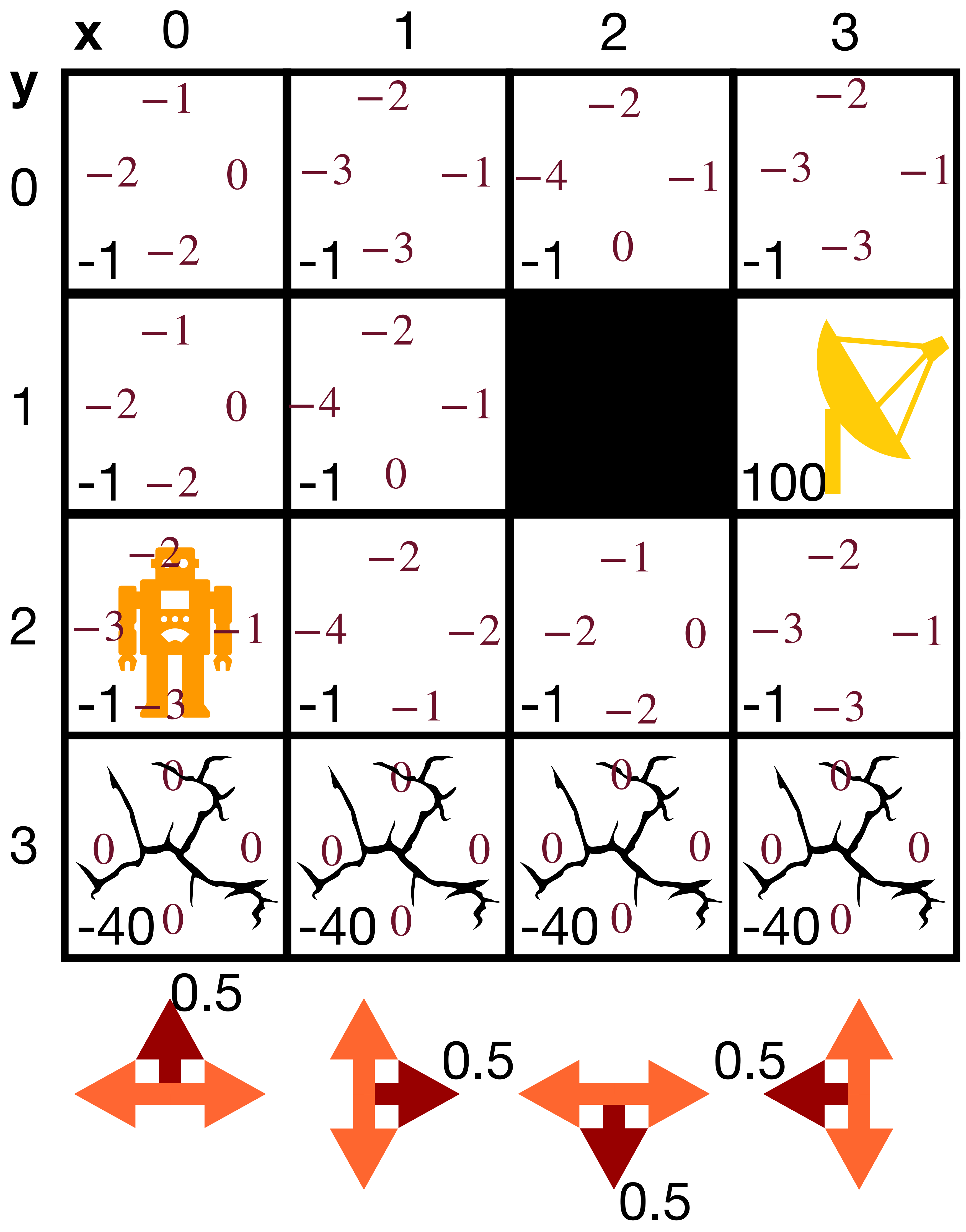
## Example 2: Off-Policy TD Algorithm

**Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$**

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize $S$
  Loop for each step of episode:
    Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\varepsilon$-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

$\alpha = 0.1 \quad \gamma = 0.9 \quad \varepsilon = 0.4$



# Model free RL

## Example 2: Off-Policy TD Algorithm

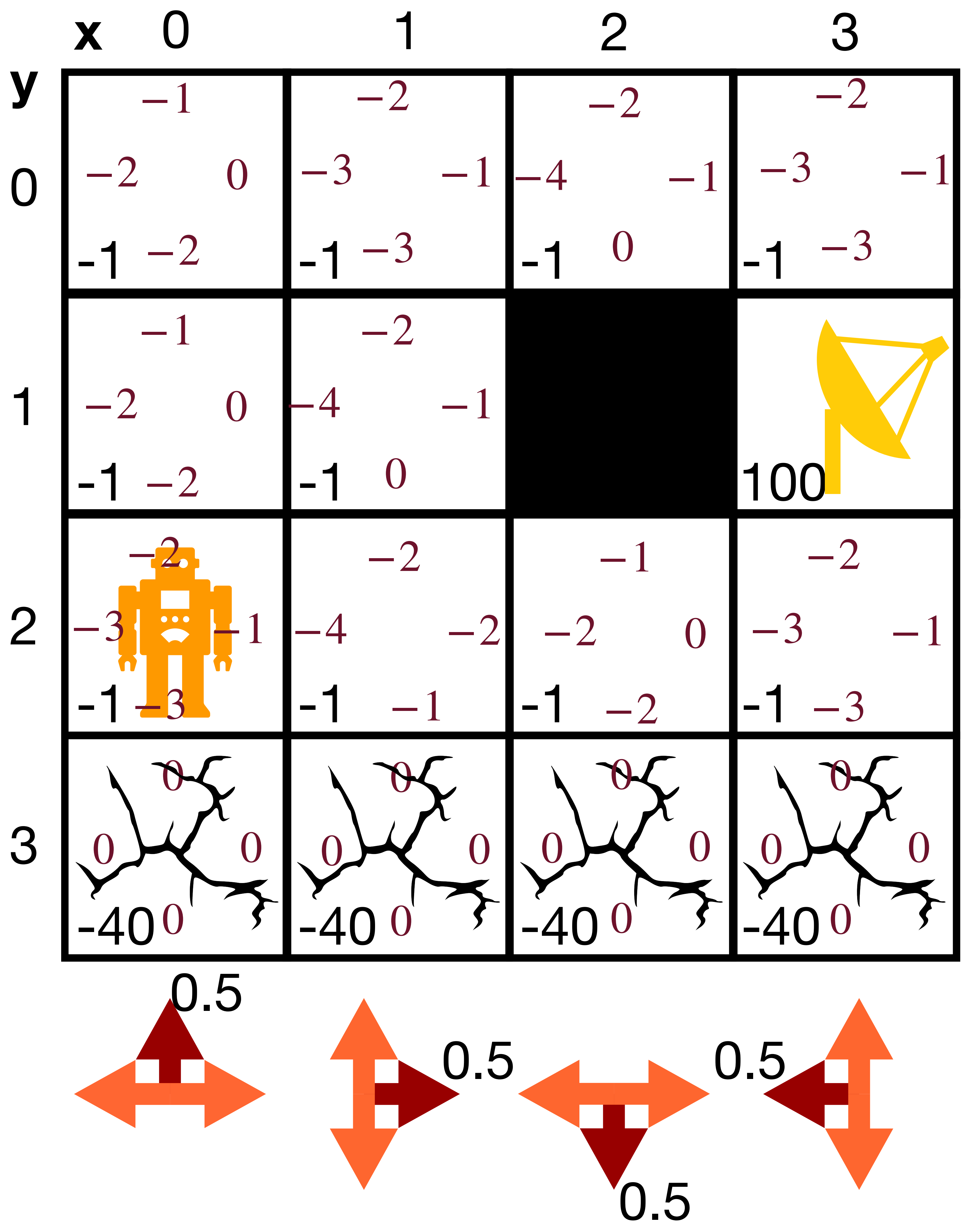
**Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$**

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize $S$
  Loop for each step of episode:
    Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\varepsilon$-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

$\alpha = 0.1 \quad \gamma = 0.9 \quad \varepsilon = 0.4$

$S = (0,2)$



# Model free RL

## Example 2: Off-Policy TD Algorithm

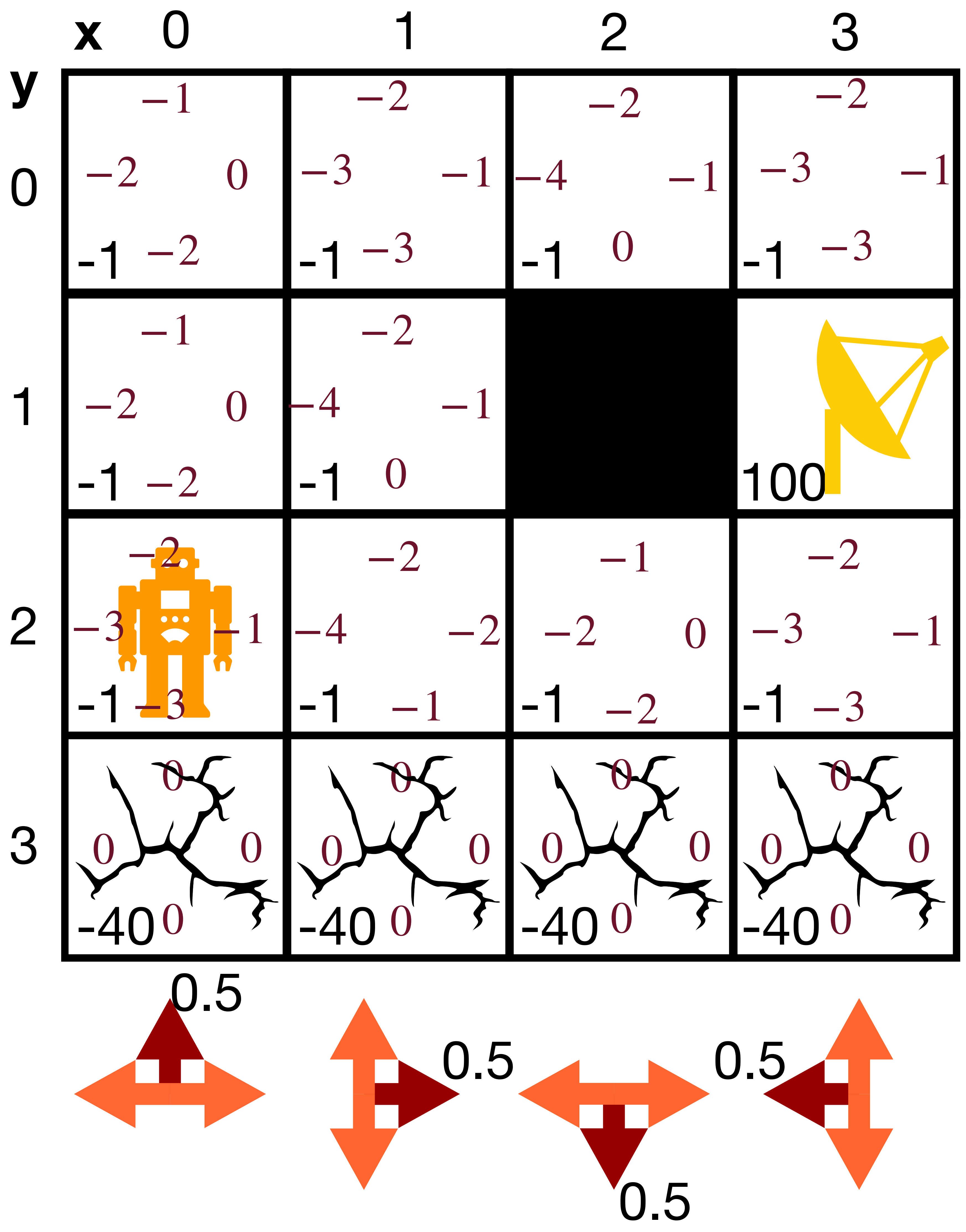
**Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$**

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize $S$
  Loop for each step of episode:
    Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\varepsilon$-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

$\alpha = 0.1 \quad \gamma = 0.9 \quad \varepsilon = 0.4$

$S = (0{,}2) \quad c \sim U_{[0,1]} = 0.42 > \varepsilon \Rightarrow A = \text{RIGHT}$



# Model free RL

## Example 2: Off-Policy TD Algorithm

**Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$**

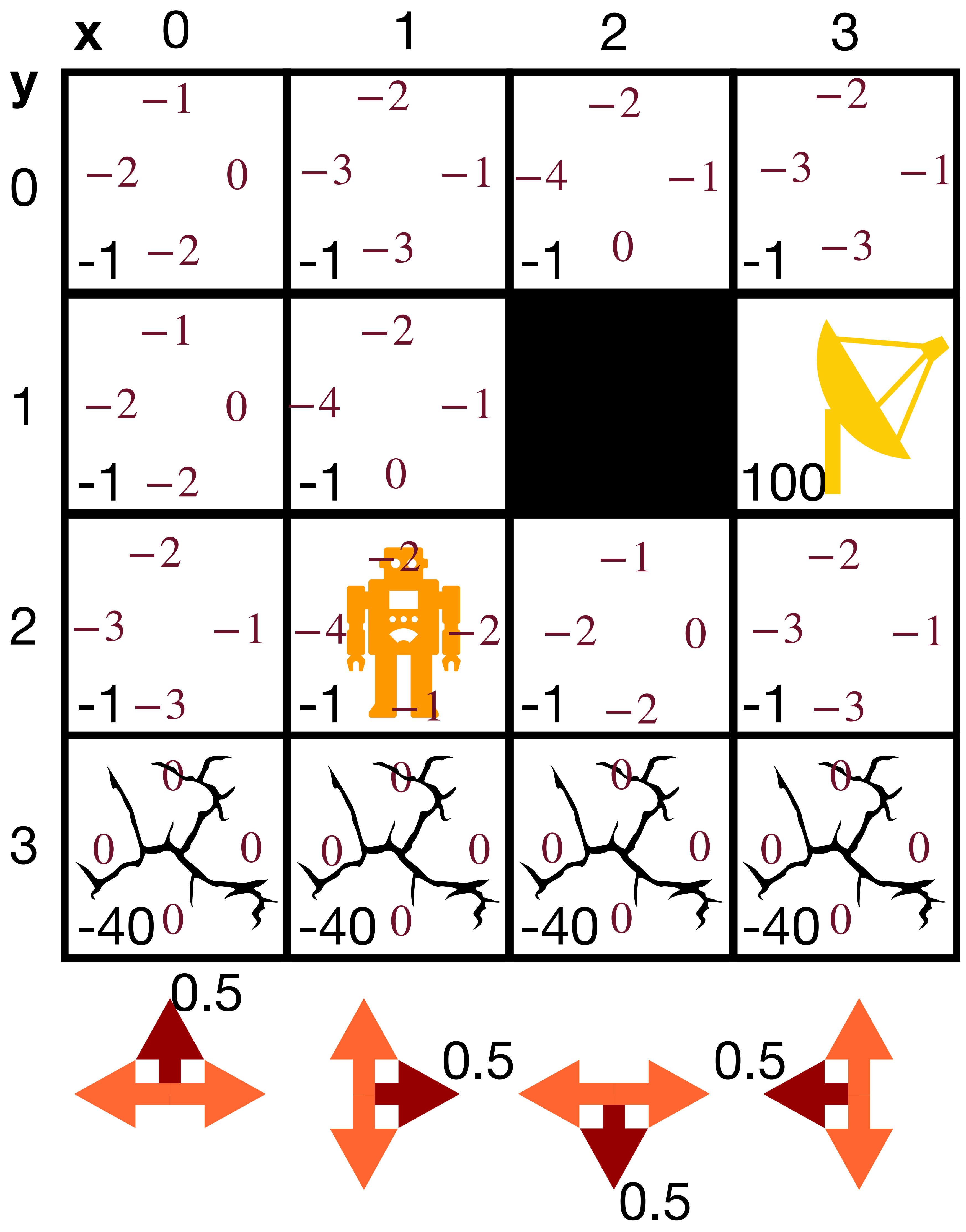
Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize $S$
  Loop for each step of episode:
    Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\varepsilon$-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

$\alpha = 0.1 \quad \gamma = 0.9 \quad \varepsilon = 0.4$

$S = (0,2) \quad c \sim U_{[0,1]} = 0.42 > \varepsilon \Rightarrow A = \text{RIGHT} \quad R = -0.1$

$S' = (1,2)$



# Model free RL

## Example 2: Off-Policy TD Algorithm

**Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$**

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

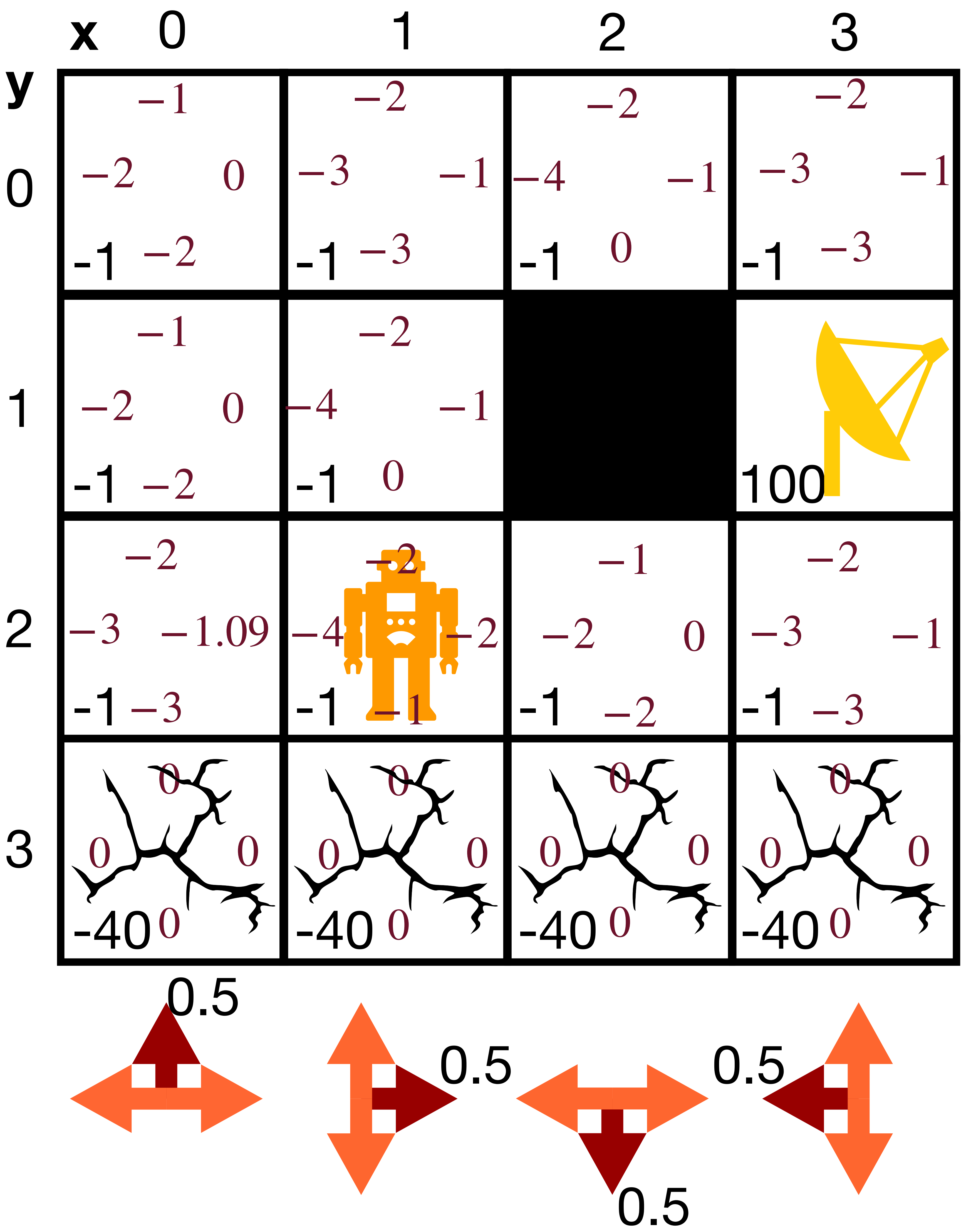
Loop for each episode:
  Initialize $S$
  Loop for each step of episode:
    Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\varepsilon$-greedy)
    Take action $A$, observe $R$, $S'$
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until $S$ is terminal

$\alpha = 0.1 \quad \gamma = 0.9 \quad \varepsilon = 0.4$

$S = (0,2) \quad c \sim U_{[0,1]} = 0.42 > \varepsilon \Rightarrow A = \text{RIGHT} \quad R = -0.1$

$S' = (1,2)$

$Q\big((0,2), \text{RIGHT}\big) \leftarrow -1 + 0.1 \cdot [-1 + 0.9 \cdot 0 - (-1)] = -1.09$



# Model free RL

## Example 2: Off-Policy TD Algorithm

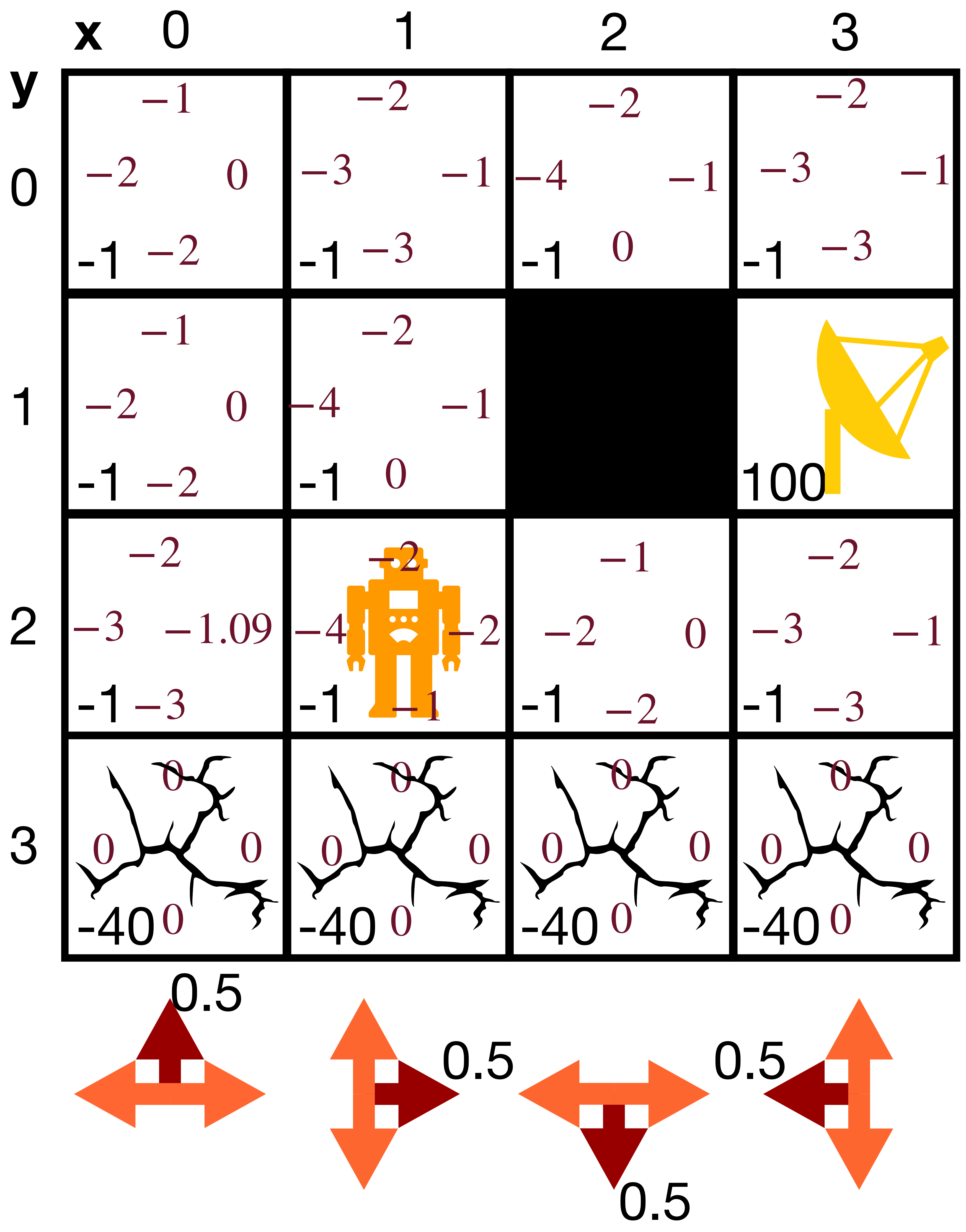
**Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$**

Algorithm parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$
Initialize $Q(s,a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize $S$
  Loop for each step of episode:
    Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\varepsilon$-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

$\alpha = 0.1 \quad \gamma = 0.9 \quad \varepsilon = 0.4$

$S = (1,2)$



# Model free RL

## Example 2: Off-Policy TD Algorithm

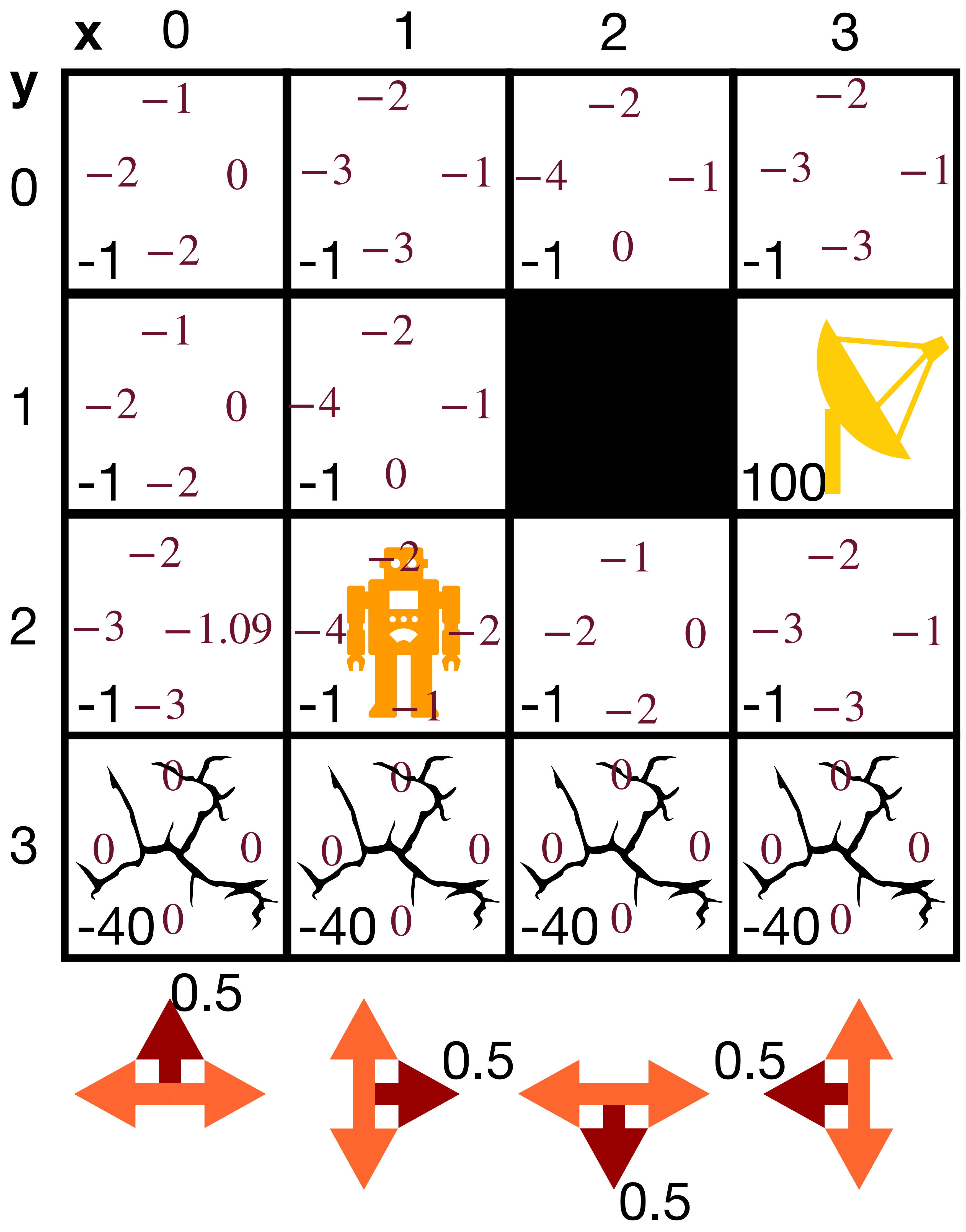
**Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$**

Algorithm parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$
Initialize $Q(s,a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize $S$
  Loop for each step of episode:
    Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\varepsilon$-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

$\alpha = 0.1 \quad \gamma = 0.9 \quad \varepsilon = 0.4$

$S = (1,2) \quad c \sim U_{[0,1]} = 0.82 > \varepsilon \Rightarrow A = \text{DOWN}$



# Model free RL

## Example 2: Off-Policy TD Algorithm

**Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$**

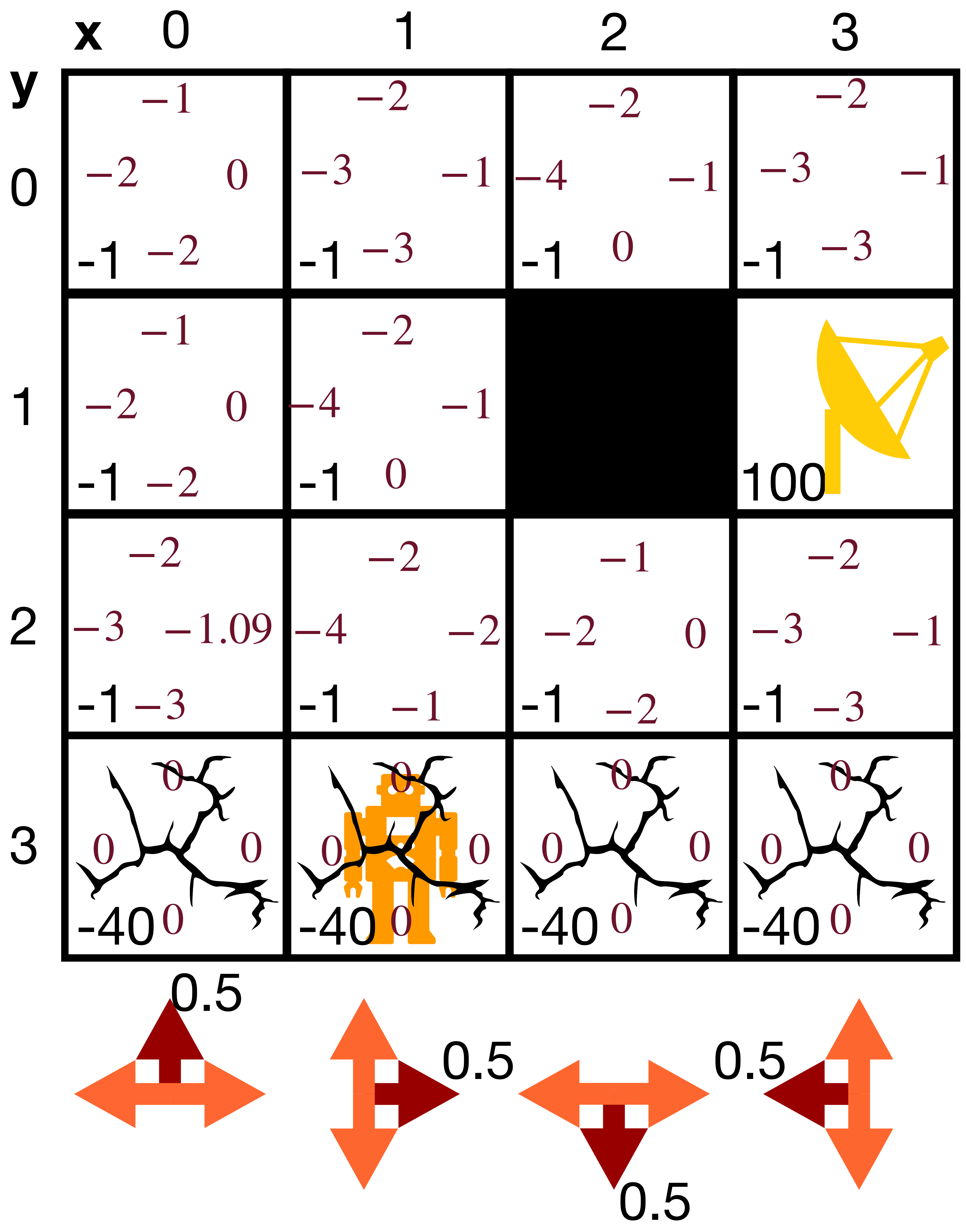
Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize $S$
  Loop for each step of episode:
    Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\varepsilon$-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

$\alpha = 0.1 \quad \gamma = 0.9 \quad \varepsilon = 0.4$

$S = (1{,}2) \quad c \sim U_{[0,1]} = 0.82 < \varepsilon \Rightarrow A = \text{DOWN} \qquad R = -40$

$S' = (1{,}3)$



# Model free RL

## Example 2: Off-Policy TD Algorithm

**Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$**

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

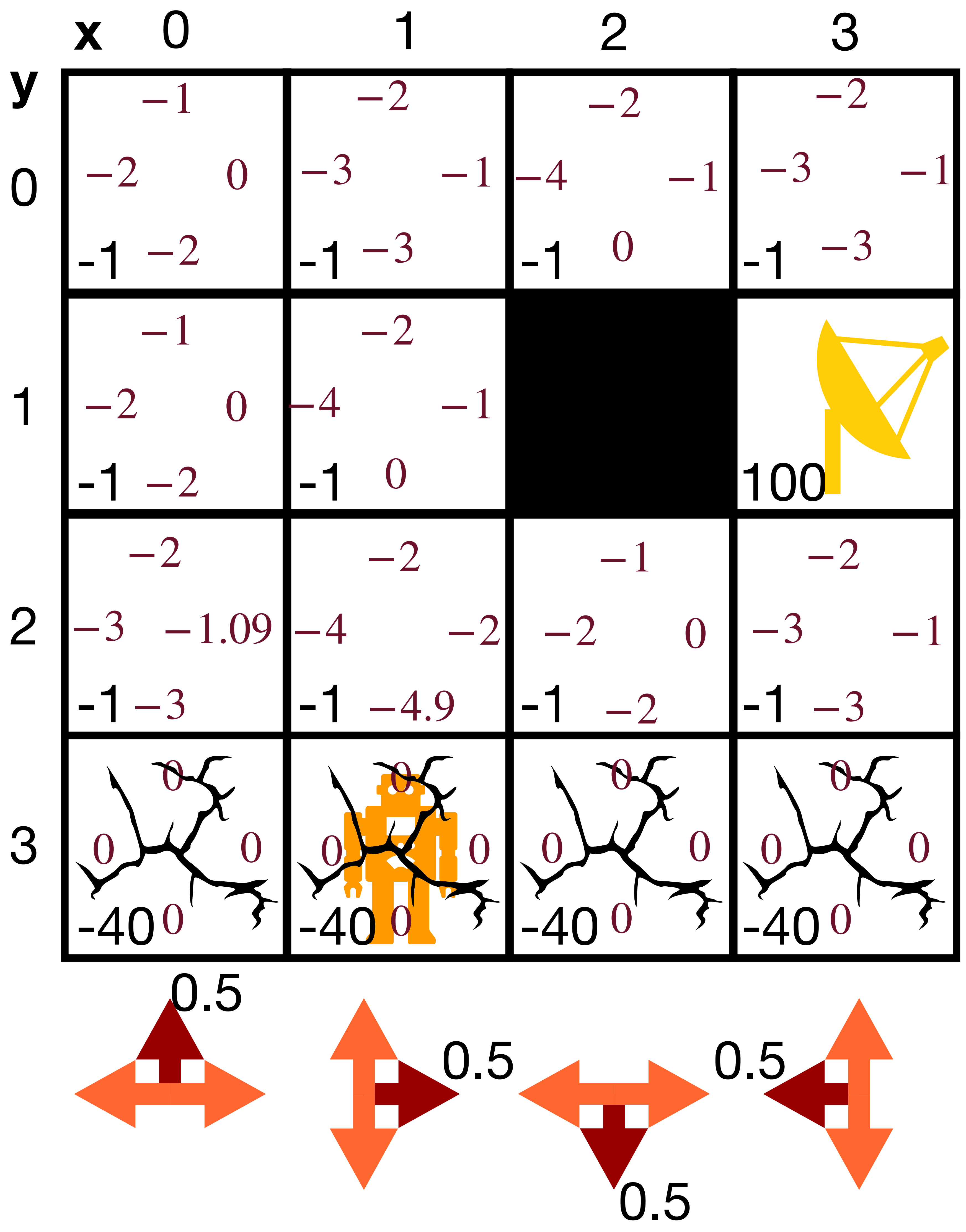
Loop for each episode:
 Initialize $S$
 Loop for each step of episode:
  Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\varepsilon$-greedy)
  Take action $A$, observe $R$, $S'$
  $Q(S, A) \leftarrow Q(S, A) + \alpha\left[R + \gamma \max_a Q(S', a) - Q(S, A)\right]$
  $S \leftarrow S'$
 until $S$ is terminal

$\alpha = 0.1 \quad \gamma = 0.9 \quad \varepsilon = 0.4$

$S = (1,2) \quad c \sim U_{[0,1]} = 0.82 < \varepsilon \Rightarrow A = \text{DOWN} \qquad R = -40$

$S' = (1,3)$

$$Q\left((1,2), \text{DOWN}\right) \leftarrow -1 + 0.1 \cdot [-40 + 0.9 \cdot 0 - (-1)] = -4.9$$



# Model free RL

## Example 2: Off-Policy TD Algorithm

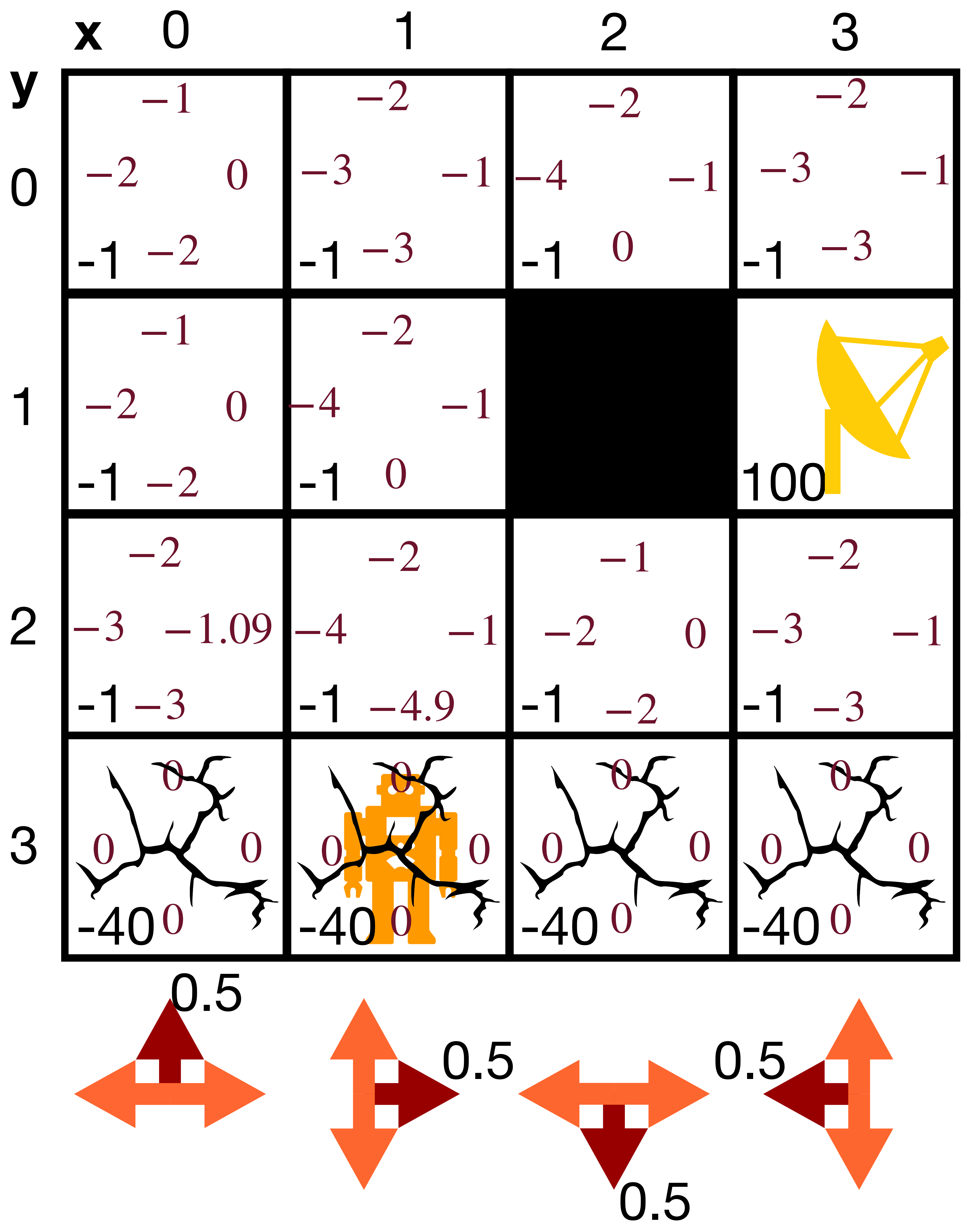
**Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$**

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize $S$
  Loop for each step of episode:
    Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\varepsilon$-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

$\alpha = 0.1 \quad \gamma = 0.9 \quad \varepsilon = 0.4$

$S = (1,3)$



# Model free RL

## Example 2: Off-Policy TD Algorithm

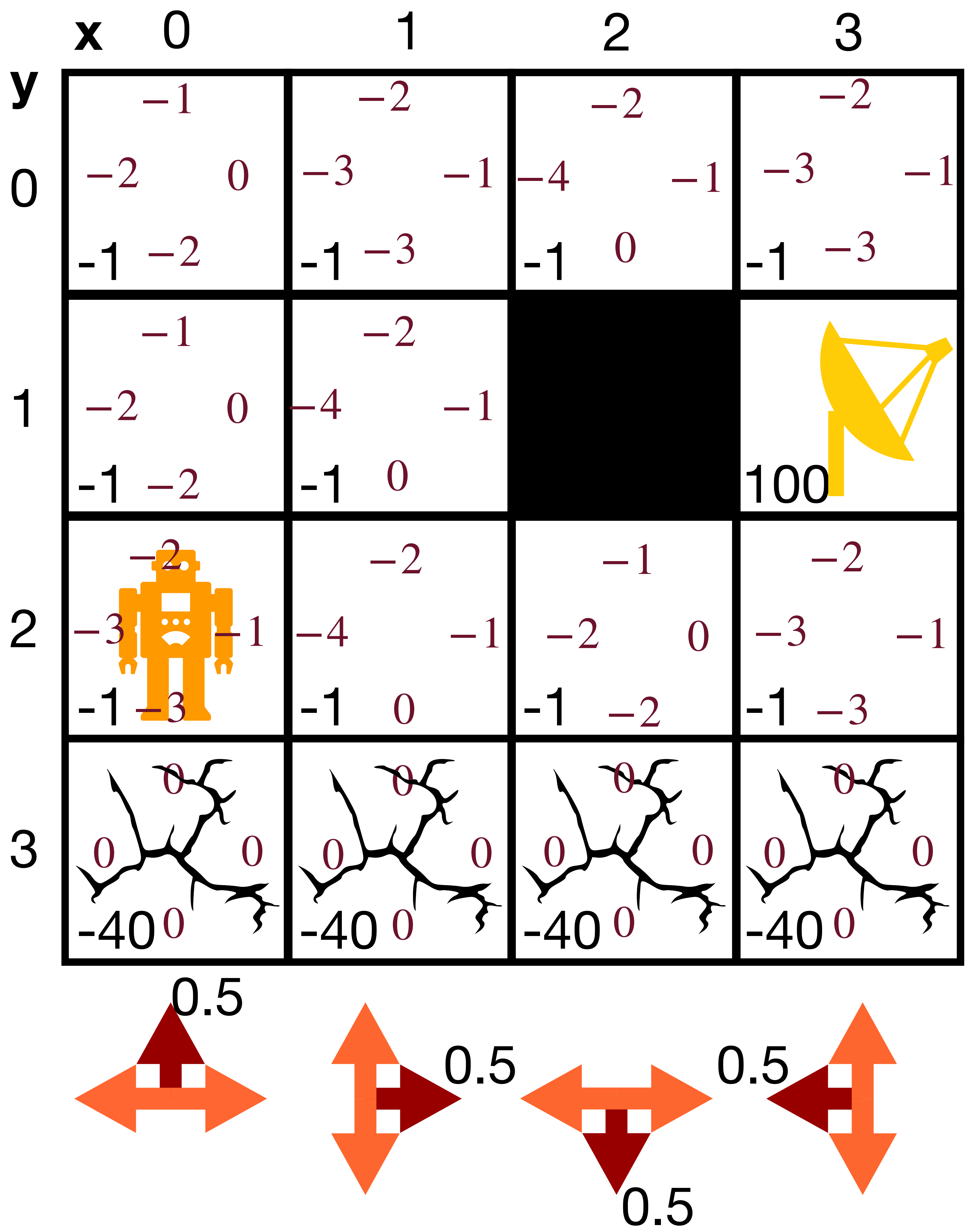
**Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$**

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize $S$
  Loop for each step of episode:
    Choose $A$ from $S$ using policy derived from $Q$ (e.g., $\varepsilon$-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

$\alpha = 0.1 \quad \gamma = 0.9 \quad \varepsilon = 0.4$

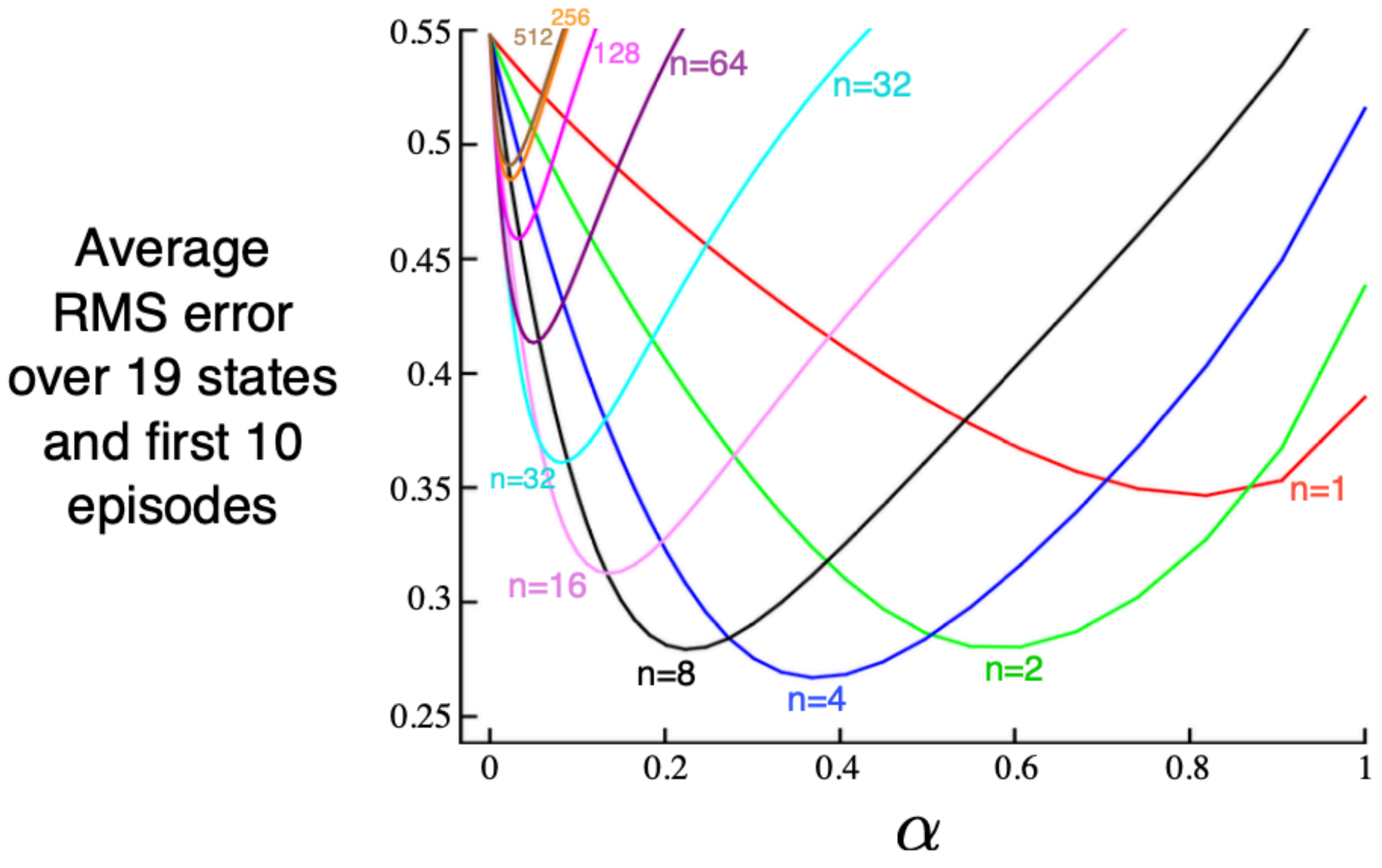
$S = (0,2)$



# Model free RL

## Conclusions

- We have shown on-policy MC methods and off-policy Q-Learning
- This does not mean that all MC methods are on-policy and all TD methods are off-policy
  - Off-policy MC methods: Utilise importance sampling
  - SARSA (Step-Action-Reward-Step-Action): On-policy TD method
- We have seen 1-step TD methods:
  - n-step TD methods bridge the gap between MC and TD paradigms




n-step TD performance with varying *n* — Intermediate solutions may be the best [Sutton & Barto 2018]

# Model free RL

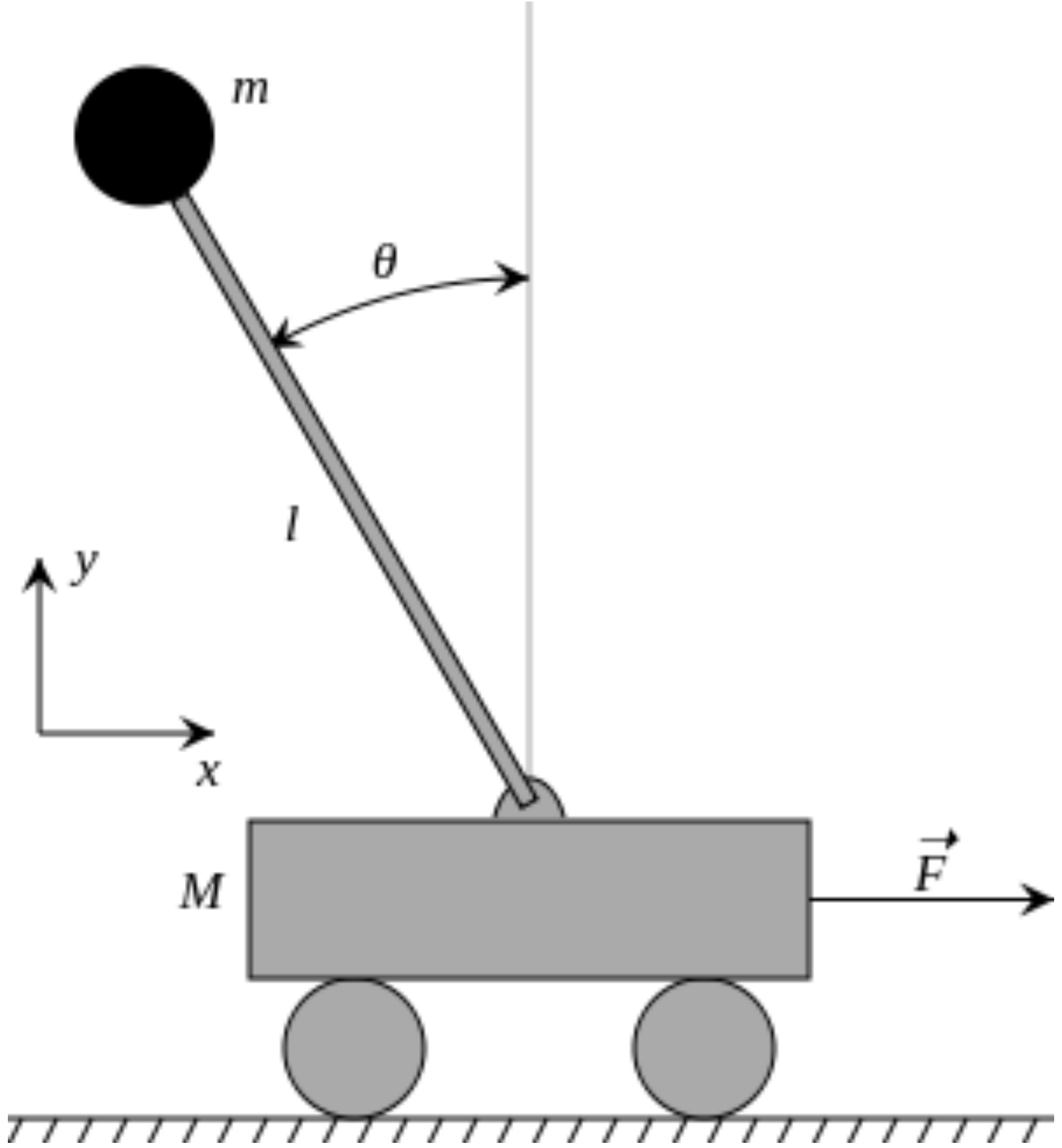
## Conclusions

| Criteria | Monte Carlo Methods | Temporal Difference Methods |
|---|---|---|
| Bias vs Variance | low bias, high variance | high bias, low variance |
| Online | | ✅ |
| Bootstrapping | | ✅ because $v_t$ is based off of $v_{t+1}$ |
| Estimation | ✅ | ✅ |
| On-Policy | On-policy MC | SARSA |
| Off-Policy | Off-policy MC with importance sampling | Q-learning |
| Past vs Future Future data | past experiences | future (models MDP) |

# Model free RL
## Beyond Tabular Methods

- How to handle large multi-dimensional state-spaces?
- Can we expect similar states in a large state-space to be reasonably similar? Is interpolation possible?
- How to handle continuous state-spaces?
  - E.g. representing angles
- How to handle continuous actions?
  - E.g. representing force

- Curse of dimensionality
- **Can we utilise (deep) neural networks somehow?**




The cart pole problem

# Outline

- Introduction
- Reinforcement Learning Formalisation
- Model-Free Reinforcement Learning
- **Value Function Approximation**
  - **Introduction**
  - **Supervised Objective**
  - **Gradient and Semi-Gradient**
  - **Example 1: MC $\hat{v} \approx v_\pi$ Evaluation**
  - **Example 2: TD $\hat{v} \approx v_\pi$ Evaluation**
  - **Conclusions**
- Policy Gradient Methods

# Value Function Approximation

## Introduction

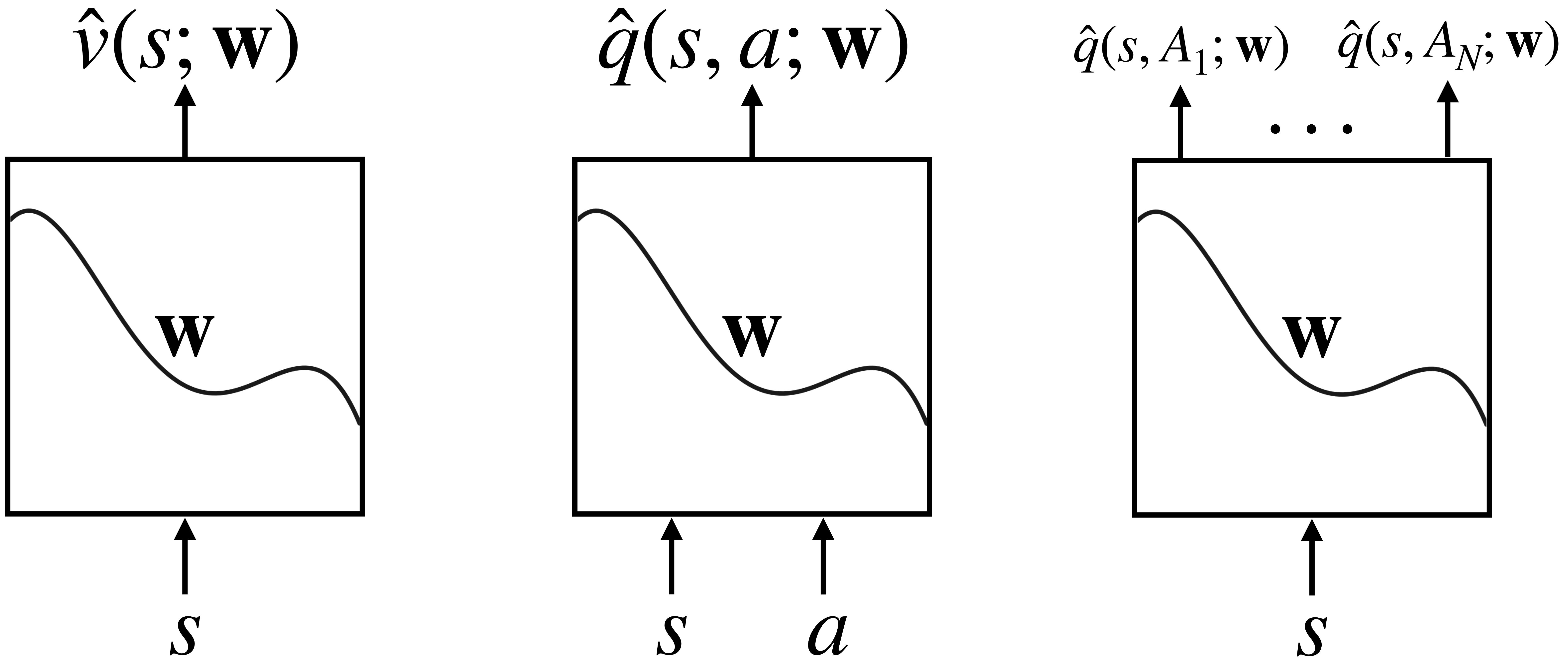
- Solutions presented so far were tabular:
  - Every state $s \in S$ has an entry $V(s)$
  - Every state-action pair $s \in S, a \in A$ has an entry $Q(s, a)$ (see slide 67)

- Three main problems:
  - Large (but potentially discrete) state-spaces:
    - Backgammon $10^{20}$ states
    - Go $10^{70}$
  - Continuous state-spaces
    - Physical properties: distances, velocities, angles, …
    - Robotics applications
  - Continuous actions

# Value Function Approximation

## Introduction

- **Idea**: Use supervised learning to train the function approximator
  - Artificial Neural Networks (ANN) + Stochastic Gradient Descent (SGD)




Left: State-Value function approximation for a given state; Middle: Action-Value function approximation for a given state and action pair; Action-Value function approximation for each action for a given state

# Value Function Approximation

## Supervised Objective

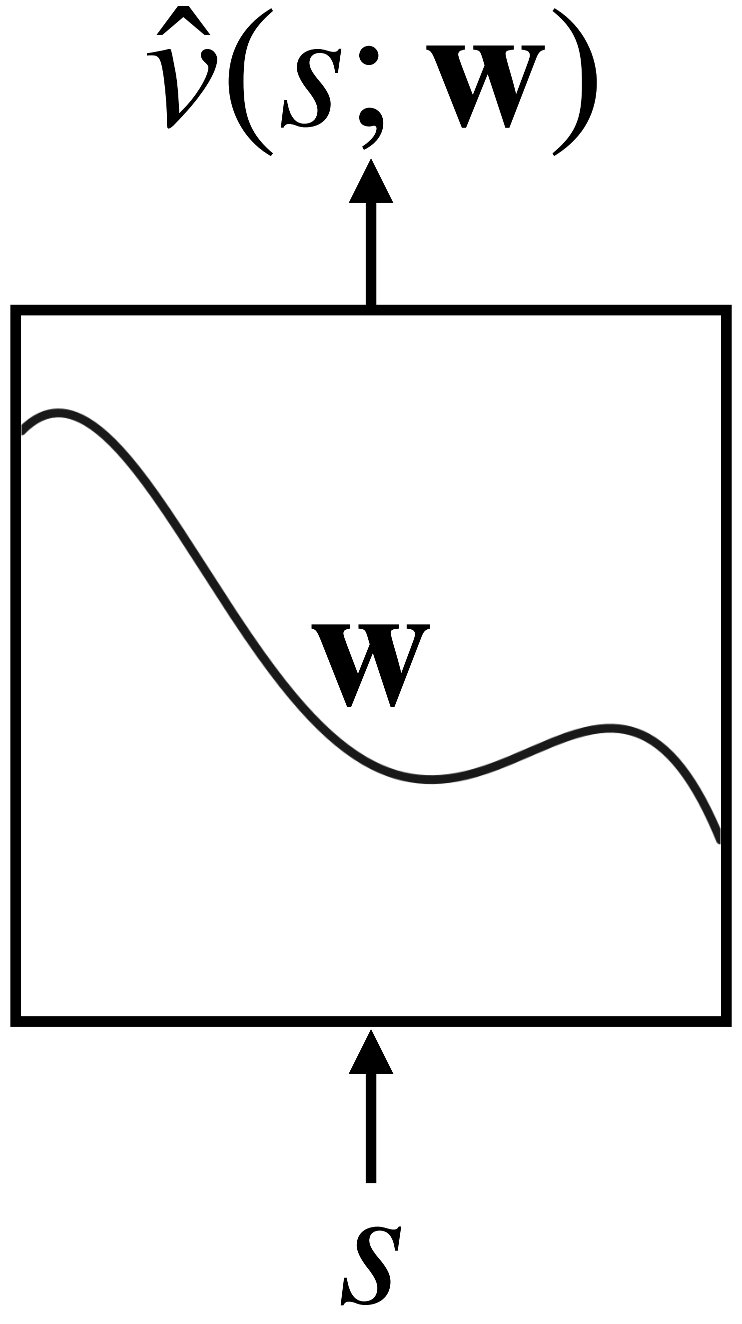
- Use standard Mean Squared Error (MSE) Loss:

$$J(\mathbf{w}) \doteq \sum_{s \in S} \mu(s) \left[ v_\pi(s) - \hat{v}(s; \mathbf{w}) \right]^2$$

- Scale each error by its importance as captured by the state visitation frequency under policy $\pi$:

$$\eta(s) = h(s) + \sum_{\bar{s}} \eta(\bar{s}) \sum_{a} \pi(a \mid \bar{s}) p(s \mid \bar{s}, a)$$

$$\mu(s) = \frac{\eta(s)}{\sum_{s'} \eta(s')}$$ on-policy distribution



# Value Function Approximation

## Gradient and Semi-Gradient Methods

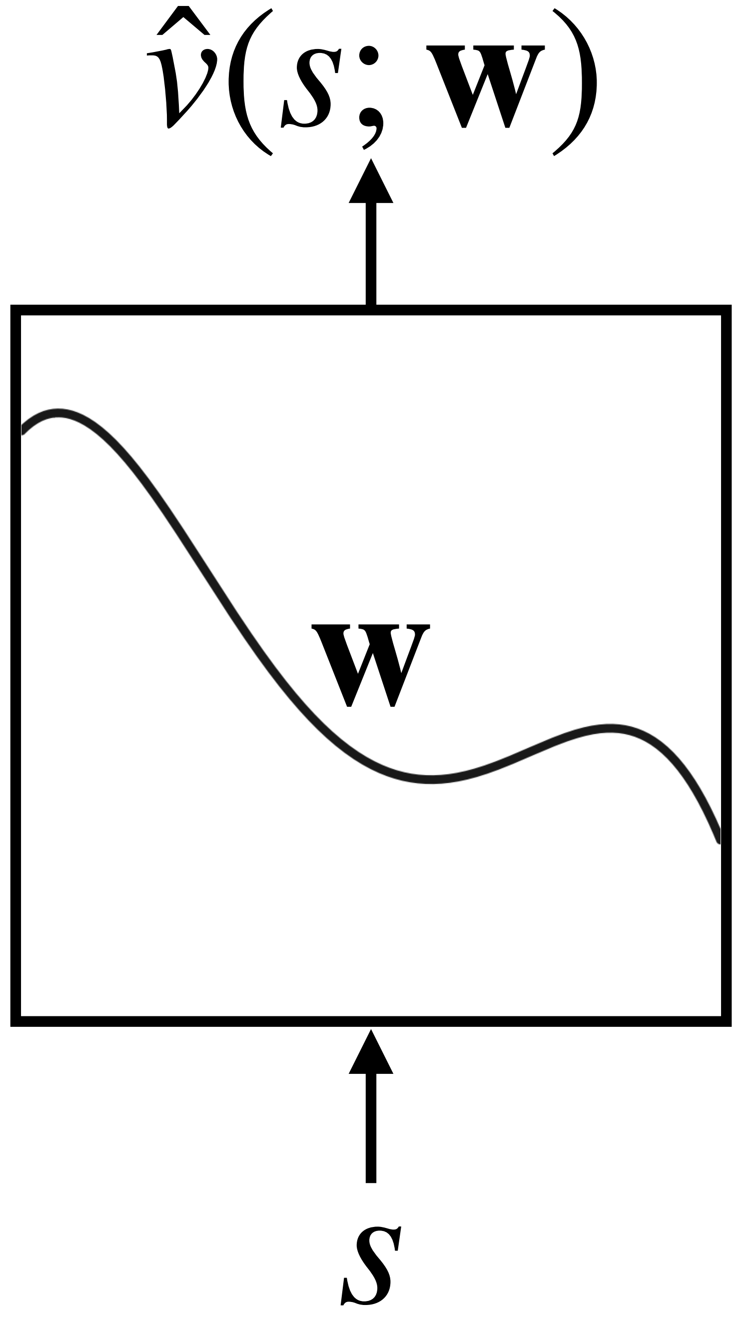
- Use standard Mean Squared Error (MSE) Loss:

$$J(\mathbf{w}) \doteq \sum_{s \in S} \mu(s) \left[ v_\pi(s) - \hat{v}(s; \mathbf{w}) \right]^2$$

- Combined with SGD:

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t - \frac{1}{2} \alpha \nabla \left[ v_\pi(S_t) - \hat{v}(S_t; \mathbf{w}_t) \right]^2$$
$$= \mathbf{w}_t + \alpha \left[ v_\pi(S_t) - \hat{v}(S_t; \mathbf{w}_t) \right] \nabla \hat{v}(S_t; \mathbf{w}_t)$$

$$\nabla f(\mathbf{w}) \doteq \left( \frac{\partial f(\mathbf{w})}{\partial w_1}, \frac{\partial f(\mathbf{w})}{\partial w_2}, \ldots, \frac{\partial f(\mathbf{w})}{\partial w_d} \right)^\top$$



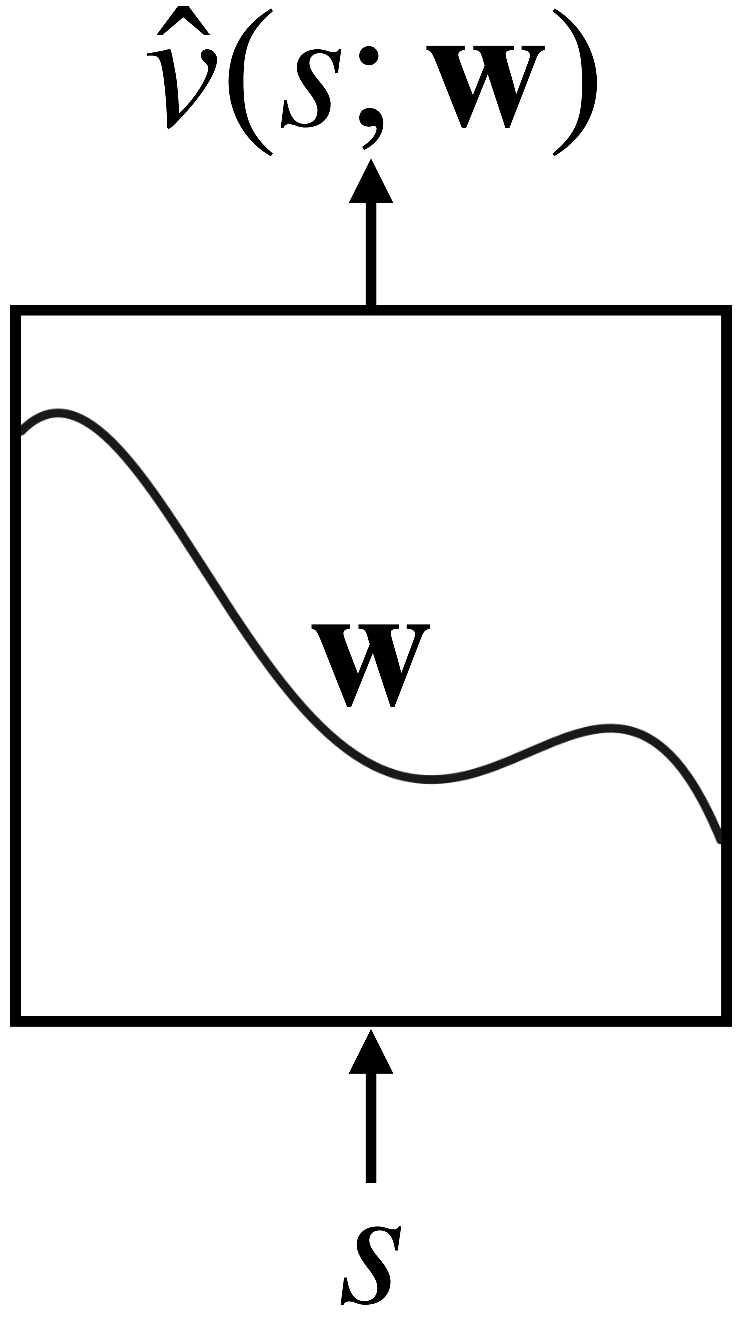
# Value Function Approximation

## Gradient and Semi-Gradient Methods

- Use standard Mean Squared Error (MSE) Loss:

$$J(\mathbf{w}) \doteq \sum_{s \in S} \mu(s) \left[ v_\pi(s) - \hat{v}(s; \mathbf{w}) \right]^2$$

- **Problem**: As this is not an actual supervised learning setting, we do not have access to $v_\pi(s)$!



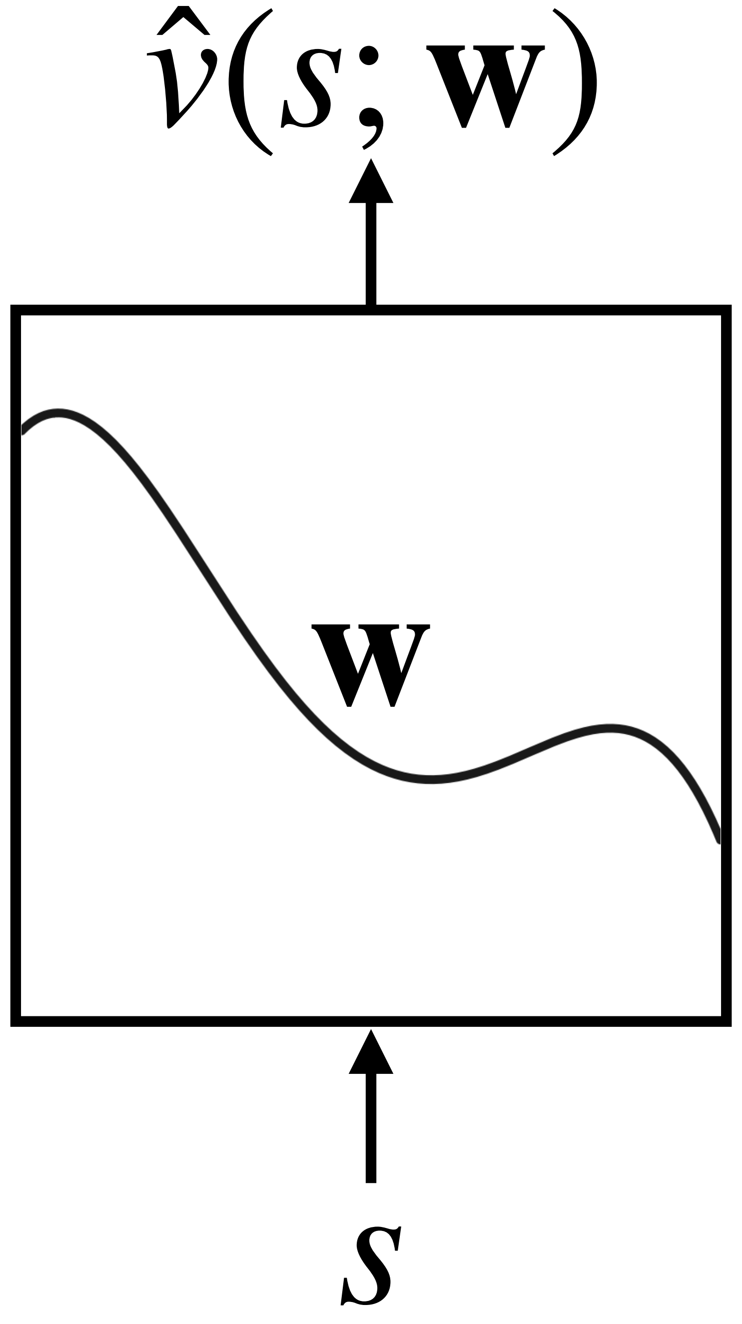
# Value Function Approximation

## Gradient and Semi-Gradient Methods

- Use standard Mean Squared Error (MSE) Loss:

$$J(\mathbf{w}) \doteq \sum_{s \in S} \mu(s) \left[ v_\pi(s) - \hat{v}(s; \mathbf{w}) \right]^2$$

- **Problem**: As this is not an actual supervised learning setting, we do not have access to $v_\pi(s)$!




$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha \left[ U_t - \hat{v}(S_t; \mathbf{w}_t) \right] \nabla \hat{v}(S_t; \mathbf{w}_t)$$

$$U_t = \begin{cases} G_t & \text{MC approach — true gradient} \\ R_{t+1} + \gamma \hat{v}(S_{t+1}; \mathbf{w}_t) & \text{TD approach — semi-gradient} \end{cases}$$

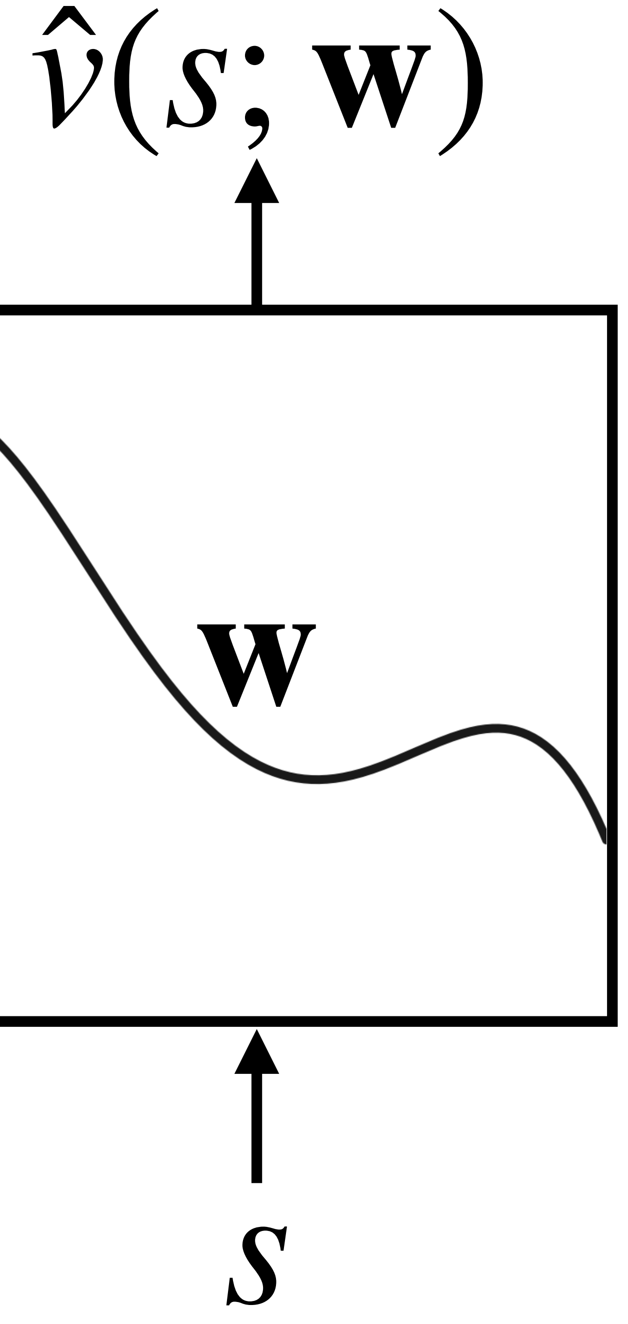
# Value Function Approximation

## Example 1: MC $\hat{v} \approx v_\pi$ Evaluation

**Gradient Monte Carlo Algorithm for Estimating $\hat{v} \approx v_\pi$**

Input: the policy $\pi$ to be evaluated
Input: a differentiable function $\hat{v} : \mathcal{S} \times \mathbb{R}^d \to \mathbb{R}$
Algorithm parameter: step size $\alpha > 0$
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)

Loop forever (for each episode):
  Generate an episode $S_0, A_0, R_1, S_1, A_1, \ldots, R_T, S_T$ using $\pi$
  Loop for each step of episode, $t = 0, 1, \ldots, T-1$:
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha\big[G_t - \hat{v}(S_t,\mathbf{w})\big]\nabla\hat{v}(S_t,\mathbf{w})$



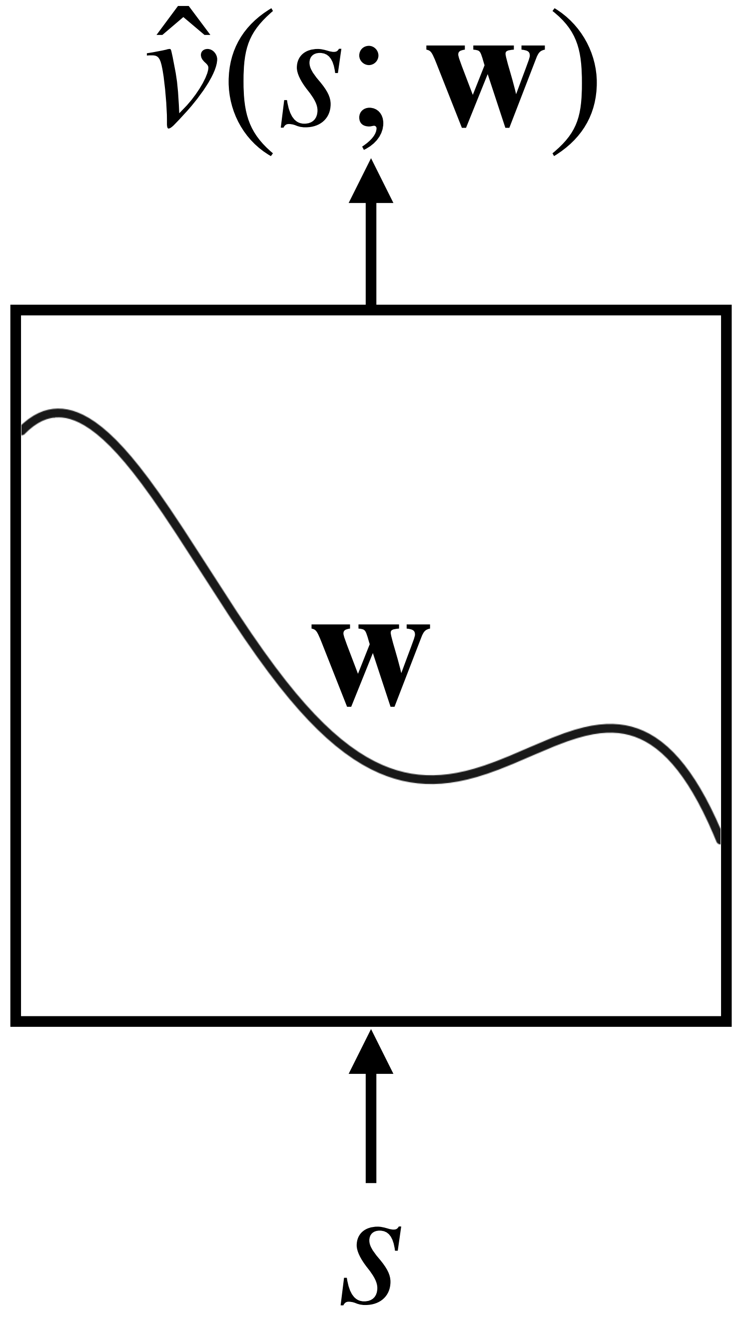
# Value Function Approximation

## Example 2: TD $\hat{v} \approx v_\pi$ Evaluation

**Semi-gradient TD(0) for estimating $\hat{v} \approx v_\pi$**

Input: the policy $\pi$ to be evaluated
Input: a differentiable function $\hat{v} : \mathcal{S}^+ \times \mathbb{R}^d \to \mathbb{R}$ such that $\hat{v}(\text{terminal},\cdot) = 0$
Algorithm parameter: step size $\alpha > 0$
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)

Loop for each episode:
  Initialize $S$
  Loop for each step of episode:
    Choose $A \sim \pi(\cdot|S)$
    Take action $A$, observe $R, S'$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha\big[R + \gamma \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})\big]\nabla \hat{v}(S,\mathbf{w})$
    $S \leftarrow S'$
  until $S$ is terminal



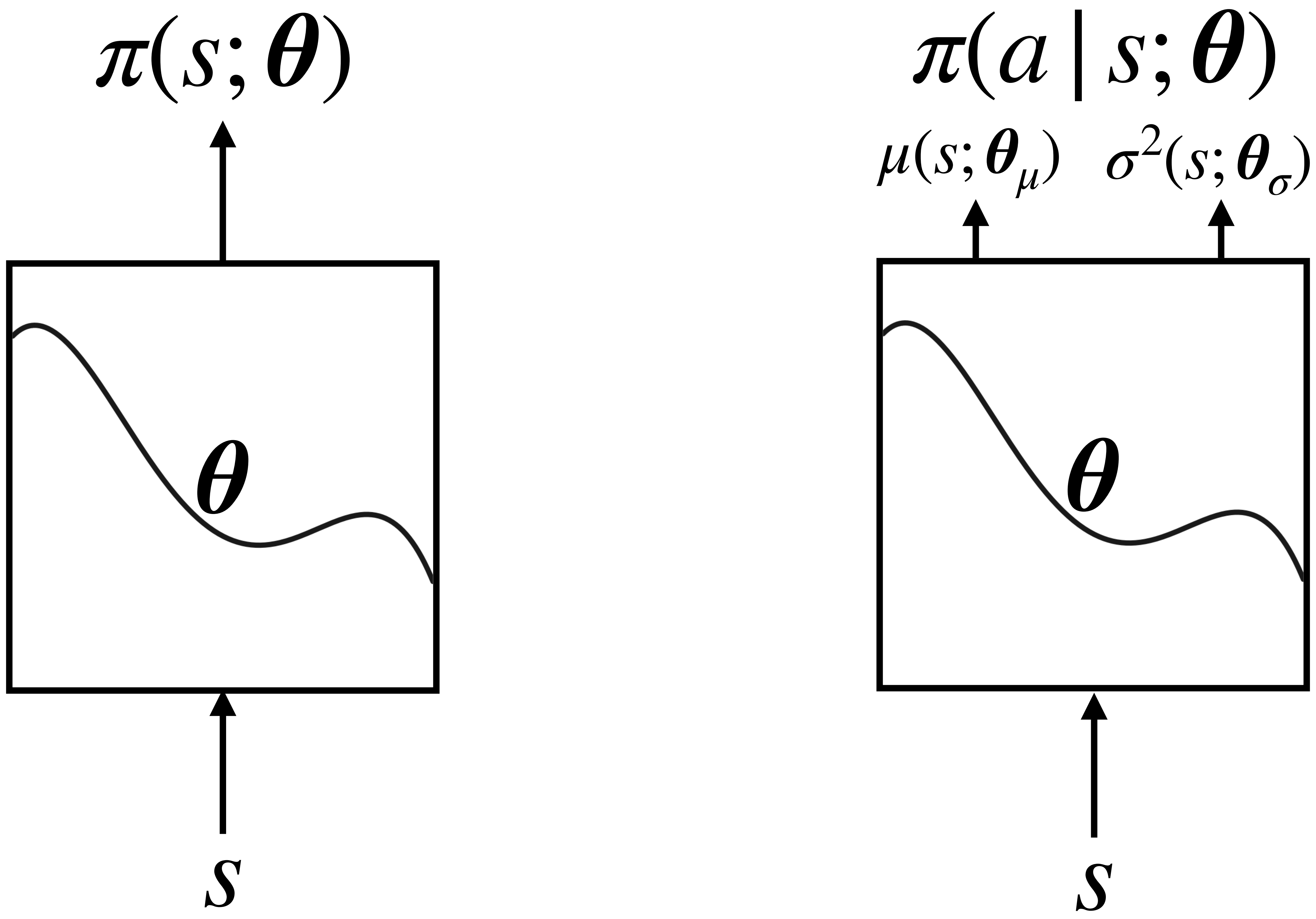
# Outline

- Introduction
- Reinforcement Learning Formalisation
- Model-Free Reinforcement Learning
- Interlude: RL Taxonomy
- Value Function Approximation
- **Policy Gradient Methods**
  - **Introduction**
  - **The Policy Gradient Theorem: Statement**
  - **The Policy Gradient Theorem: Derivation**
  - **REINFORCE Algorithm**
  - **Actor-Critic Methods**
  - **Policy Parametrisation for Continuous Actions**
  - **Conclusions**

# Policy Gradient Methods

## Introduction

- So far policies were **implicit** — we modelled state value functions; policy followed states with high values
- **Idea**: Explicitly model the policy with an ANN




Left: Deterministic policy that produces an action for a given state; Right: Stochastic policy that produces a distribution over actions given the state

# Policy Gradient Methods

## The Policy Gradient Theorem: Statement

- We wish to maximise the performance under policy $\pi$ parameterised by $\boldsymbol{\theta}$ over an entire episode:

$$J(\boldsymbol{\theta}) \doteq v_{\pi_{\boldsymbol{\theta}}}(s_0)$$

$$\nabla J(\boldsymbol{\theta}) = \mathbb{E}_{s\sim\mu_\pi, a\sim\pi,}\left[q_\pi(s,a)\nabla \ln \pi(a\,|\,s;\boldsymbol{\theta})\right]$$

- We can now improve performance using the gradient of the policy represented as an ANN, but we still have $q_\pi(s,a)$:
  - We can explicitly model $q_\pi(s,a) \approx \hat{q}(s,a;\mathbf{w})$
  - We can replace it with the return $G_t$ as $\mathbb{E}_\pi[G_t\,|\,S_t = s, A_t = a] = q_\pi(S_t, A_t)$:

$$\nabla J(\boldsymbol{\theta}) = \mathbb{E}_\pi\left[G_t \nabla \ln \pi(A_t\,|\,S_t;\boldsymbol{\theta})\right] \quad \text{REINFORCE}$$

# Policy Gradient Methods

## The Policy Gradient Theorem: Derivation

$$\nabla v_\pi(s) = \nabla \left[ \sum_a \pi(a \mid s; \boldsymbol{\theta}) q_\pi(s, a) \right] \quad \text{(slide 23)}$$

$$= \sum_a \left[ \nabla \pi(a \mid s; \boldsymbol{\theta}) q_\pi(s, a) + \pi(a \mid s; \boldsymbol{\theta}) \nabla q_\pi(s, a) \right] \quad \text{(derivative product rule)}$$

$$= \sum_a \left[ \nabla \pi(a \mid s; \boldsymbol{\theta}) q_\pi(s, a) + \pi(a \mid s; \boldsymbol{\theta}) \nabla \left[ r(s, a) + \sum_{s'} p(s' \mid s, a) v_\pi(s') \right] \right] \quad \text{(slide 23)}$$

$$= \sum_a \left[ \nabla \pi(a \mid s; \boldsymbol{\theta}) q_\pi(s, a) + \pi(a \mid s; \boldsymbol{\theta}) \sum_{s'} p(s' \mid s, a) \nabla v_\pi(s') \right] \quad \text{(recursion)}$$

# Policy Gradient Methods

## The Policy Gradient Theorem: Derivation

$$\nabla v_\pi(s) = \nabla \left[ \sum_a \pi(a \mid s; \boldsymbol{\theta}) q_\pi(s, a) \right] \quad \text{(slide 22)}$$

$$= \sum_a \left[ \nabla \pi(a \mid s; \boldsymbol{\theta}) q_\pi(s, a) + \pi(a \mid s; \boldsymbol{\theta}) \nabla q_\pi(s, a) \right] \quad \text{(derivative product rule)}$$

$$= \sum_a \left[ \nabla \pi(a \mid s; \boldsymbol{\theta}) q_\pi(s, a) + \pi(a \mid s; \boldsymbol{\theta}) \nabla \left[ r(s, a) + \sum_{s'} p(s' \mid s, a) v_\pi(s') \right] \right] \quad \text{(slide 22)}$$

$$= \sum_a \left[ \nabla \pi(a \mid s; \boldsymbol{\theta}) q_\pi(s, a) + \pi(a \mid s; \boldsymbol{\theta}) \sum_{s'} p(s' \mid s, a) \nabla v_\pi(s') \right] \quad \text{(recursion)}$$

$$= \sum_{x \in S} \sum_{k=0}^{\infty} P(s \to x, k, \pi) \sum_a \nabla \pi(a \mid x; \boldsymbol{\theta}) q_\pi(x, a) \quad \text{(unroll recursion)}$$

# Policy Gradient Methods

## The Policy Gradient Theorem: Derivation

$$\nabla J(\boldsymbol{\theta}) = \nabla v_\pi(s_0)$$

$$= \sum_s \left( \sum_{k=0}^{\infty} P(s_0 \rightarrow s, k, \pi) \right) \sum_a \nabla \pi(a \mid s; \boldsymbol{\theta}) q_\pi(s, a)$$

$$= \sum_s \eta_\pi(s) \sum_a \nabla \pi(a \mid s; \boldsymbol{\theta}) q_\pi(s, a) \qquad \text{(slide 86)}$$

$$= \sum_{s'} \eta_\pi(s') \sum_s \mu_\pi(s) \sum_a \nabla \pi(a \mid s; \boldsymbol{\theta}) q_\pi(s, a) \qquad \text{(slide 86)}$$

$$\propto \sum_s \mu_\pi(s) \sum_a \nabla \pi(a \mid s; \boldsymbol{\theta}) q_\pi(s, a)$$

# Policy Gradient Methods

## The Policy Gradient Theorem: Derivation

*grad log derivative trick*
*or*
*eligibility vector*

$$
\begin{aligned}
\nabla J(\boldsymbol{\theta}) &= \nabla v_\pi(s_0) \\
&\propto \sum_s \mu_\pi(s) \sum_a \nabla \pi(a \mid s; \boldsymbol{\theta}) q_\pi(s, a) \\
&= \sum_s \mu_\pi(s) \sum_a \pi(a \mid s; \boldsymbol{\theta}) q_\pi(s, a) \frac{\nabla \pi(a \mid s; \boldsymbol{\theta})}{\pi(a \mid s; \boldsymbol{\theta})} \\
&= \sum_s \mu_\pi(s) \sum_a \pi(a \mid s) q_\pi(s, a) \nabla \ln \pi(a \mid s; \boldsymbol{\theta}) \\
&= \mathbb{E}_{s \sim \mu_\pi, a \sim \pi} \left[ q_\pi(s, a) \nabla \ln \pi(a \mid s; \boldsymbol{\theta}) \right]
\end{aligned}
$$

and $\nabla \ln \pi(\mathrm{a} \mid \mathrm{s}; \boldsymbol{\theta}) = \frac{1}{\pi(\mathrm{a} \mid \mathrm{s}; \boldsymbol{\theta})} \cdot \nabla \pi(\mathrm{a} \mid \mathrm{s}; \boldsymbol{\theta})$

$$
J(\boldsymbol{\theta}) \doteq v_{\pi_\theta}(s_0)
$$

$$
\nabla J(\boldsymbol{\theta}) = \mathbb{E}_{s \sim \mu_\pi, a \sim \pi,} \left[ q_\pi(s, a) \nabla \ln \pi(a \mid s; \boldsymbol{\theta}) \right]
$$

# Policy Gradient Methods

## REINFORCE Algorithm

$$\nabla J(\boldsymbol{\theta}) = \mathbb{E}_\pi \left[ q_\pi(s,a) \nabla \ln \pi(a \mid s; \boldsymbol{\theta}) \right]$$

$$= \mathbb{E}_\pi \left[ G_t \nabla \ln \pi(A_t \mid S_t; \boldsymbol{\theta}) \right] \text{ as } q_\pi \doteq \mathbb{E}_\pi \left[ G_t \mid A_t = a, S_t = s \right]$$ (Slide 22)

$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha G_t \nabla \ln \pi(A_t \mid S_t; \boldsymbol{\theta}_t)$$ Gradient Ascent Step

**REINFORCE: Monte-Carlo Policy-Gradient Control (episodic) for $\pi_*$**

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Algorithm parameter: step size $\alpha > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):
  Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \boldsymbol{\theta})$
  Loop for each step of the episode $t = 0, 1, \ldots, T-1$:
    $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$ $(G_t)$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t G \nabla \ln \pi(A_t|S_t, \boldsymbol{\theta})$

# Policy Gradient Methods

## Actor-Critic Methods

- Speed-up learning and reduce variance by utilising bootstrapping
- Use $\hat{q}(s, a; \mathbf{w})$ or $\hat{v}(s; \mathbf{w})$ to estimate the TD residual $\delta_t$

$$\begin{aligned}\boldsymbol{\theta}_{t+1} &\doteq \boldsymbol{\theta}_t + \alpha \left(R_{t+1} + \gamma \hat{v}(S_{t+1}; \mathbf{w}) - \hat{v}(S_t; \mathbf{w})\right) \nabla \ln \pi(A_t \mid S_t; \boldsymbol{\theta}_t) \\ &= \boldsymbol{\theta}_t + \alpha \delta_t \ln \pi(A_t \mid S_t; \boldsymbol{\theta}_t)\end{aligned}$$

**One-step Actor–Critic (episodic), for estimating $\pi_{\boldsymbol{\theta}} \approx \pi_*$**

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s,\mathbf{w})$
Parameters: step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^{d}$ (e.g., to $\mathbf{0}$)
Loop forever (for each episode):
    Initialize $S$ (first state of episode)
    $I \leftarrow 1$
    Loop while $S$ is not terminal (for each time step):
        $A \sim \pi(\cdot|S, \boldsymbol{\theta})$
        Take action $A$, observe $S', R$
        $\delta \leftarrow R + \gamma \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$    (if $S'$ is terminal, then $\hat{v}(S',\mathbf{w}) \doteq 0$)
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S,\mathbf{w})$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I \delta \nabla \ln \pi(A|S, \boldsymbol{\theta})$
        $I \leftarrow \gamma I$
        $S \leftarrow S'$

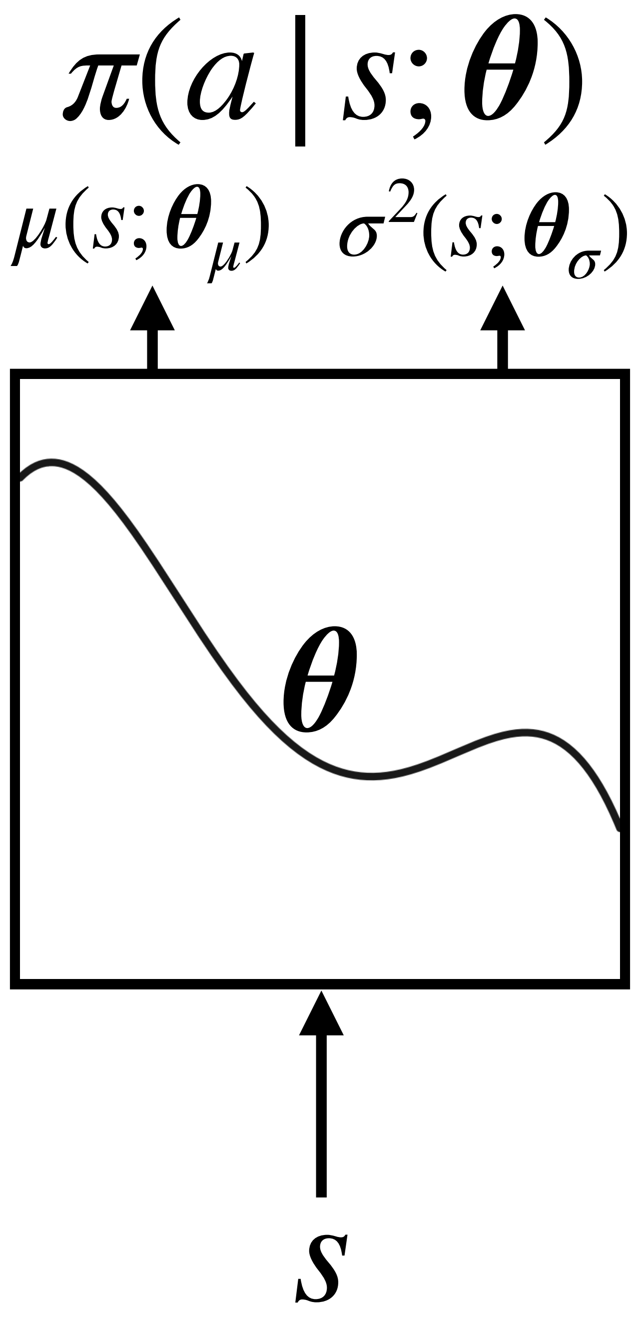
# Policy Gradient Methods

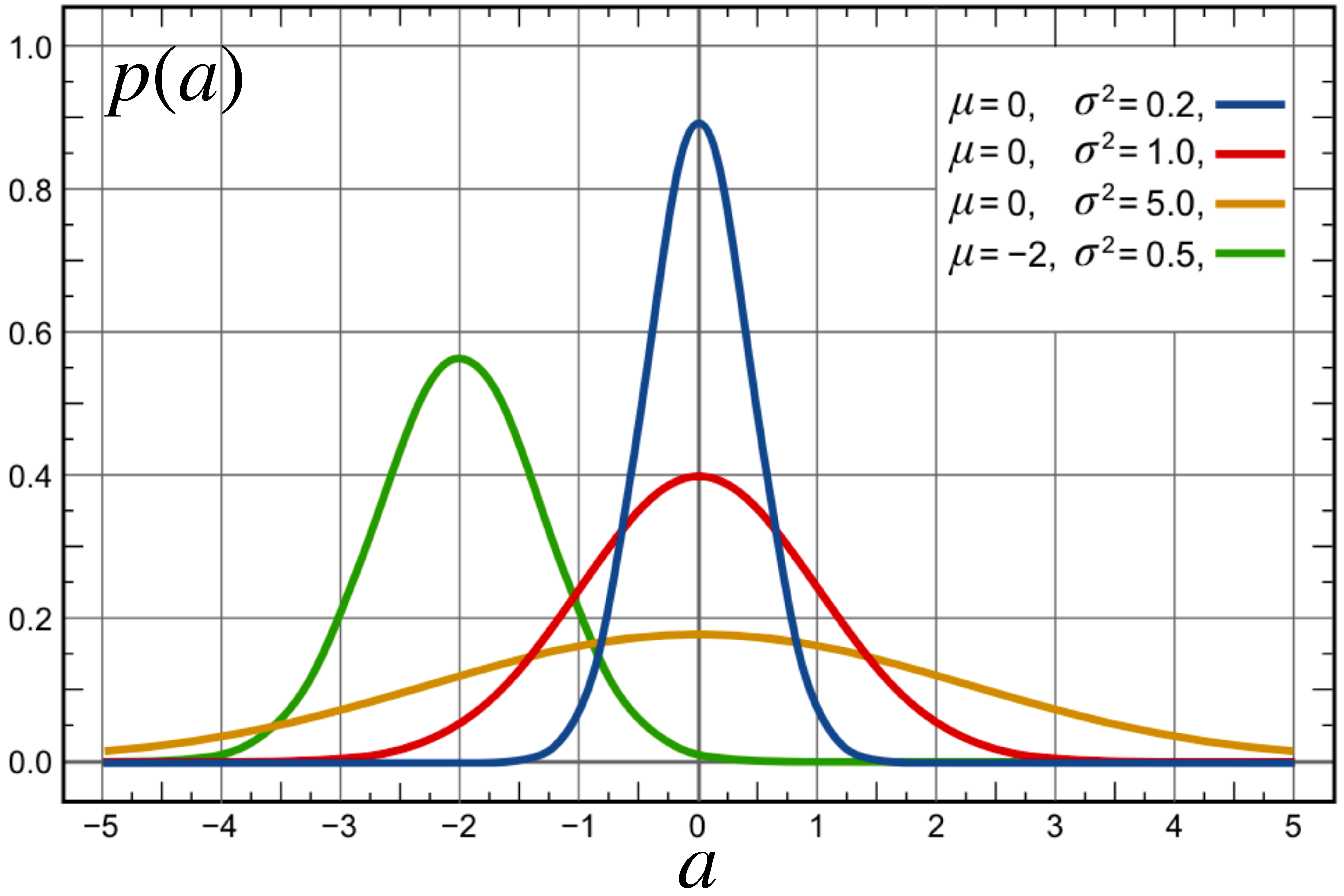
## Policy Parametrisation for Continuous Actions

- Define policy as the Gaussian probability density over the real-valued actions
- Use function approximation for $\mu(s;\boldsymbol{\theta}_\mu)$ and $\sigma^2(s;\boldsymbol{\theta}_\sigma)$, with potentially the same feature extractor base $\mathbf{x}(s)$
- We can either learn the variance, or keep it fixed to ensure sufficient exploration throughout learning

$$\pi(a\,|\,s;\boldsymbol{\theta}) \doteq \frac{1}{\sigma(s;\boldsymbol{\theta}_\sigma)\sqrt{2\pi}} e^{\left(-\frac{(a-\mu(s;\boldsymbol{\theta}_\mu))^2}{2\sigma(s;\boldsymbol{\theta}_\sigma)^2}\right)}, \ \boldsymbol{\theta} = [\boldsymbol{\theta}_\mu, \boldsymbol{\theta}_\sigma]^\top$$

$$\mu(s,\boldsymbol{\theta}_\mu) \doteq \boldsymbol{\theta}_\mu^\top \mathbf{x}(s), \ \sigma(s,\boldsymbol{\theta}_\sigma) \doteq e^{\left(\boldsymbol{\theta}_\sigma^\top \mathbf{x}(s)\right)}$$



# Policy Gradient Methods

## Policy Gradient (Monte Carlo) vs TD Learning

| Criteria | Policy Gradient | Temporal Difference Methods |
| --- | --- | --- |
| Bias vs Variance | low bias, high variance | high bias, low variance |
| Online | | ✅ |
| Bootstrapping | | ✅ because $v_t$ is based off of $v_{t+1}$ |
| Estimation | ✅ | ✅ |
| On-Policy | ✅ | ✅ |
| Off-Policy | ✅ | ✅ |
| Exploration vs Exploitation | may be naturally handled | not naturally handled |
| Past vs Future Future data | past experiences | future (models MDP) |
| **Convergence speed** | ✅ | |
| **Convergence guarantees & stability*** | ✅ | |
| **Sample efficiency** | | ✅ |
| **Stochastic policy representation** | ✅ | |
| **Applicability to continuous action spaces** | ✅ | |

- **The Deadly Triad:** Bootstrapping & Function approximation & Off-policy

# Policy Gradient Methods

## Conclusions

- Stronger convergence guarantees compared to TD function approximation methods due to the Policy Gradient Theorem
- Naturally applicable on continuous action spaces
- Can represent stochastic policies and approach deterministic policies asymptomatically
- Most modern state of the art algorithms belong to either Actor-Critic methods **which combine both Monte Carlo and TD approaches**

# Next Steps

## RL Areas and Papers

- State of the art **model-free** algorithms:
  - Proximal Policy Optimisation (Schulman et al, 2017), Soft Actor Critic (Haarnoja et al, 2018)
- **Model-based** approaches:
  - Dyna-Q (Sutton and Barto 2018), Monte Carlo Tree Search (Sutton and Barto, 2018), World Models (Ha and Schmidhuber, 2018)
- **Hierarchical** reinforcement learning:
  - Between MDPs and Semi-MDPs (The options framework) (Sutton et al. 1999)
- **Intrinsically motivated** reinforcement learning:
  - Intrinsic Motivation and Reinforcement Learning (Barto, 2013), Curiosity-driven Exploration by Self-supervised Prediction (Pathak et al, 2017)

# (~˘▾˘)~ Thanks! ☜( ﾟ∀ ﾟ☜)