Introduction to Reinforcement Learning

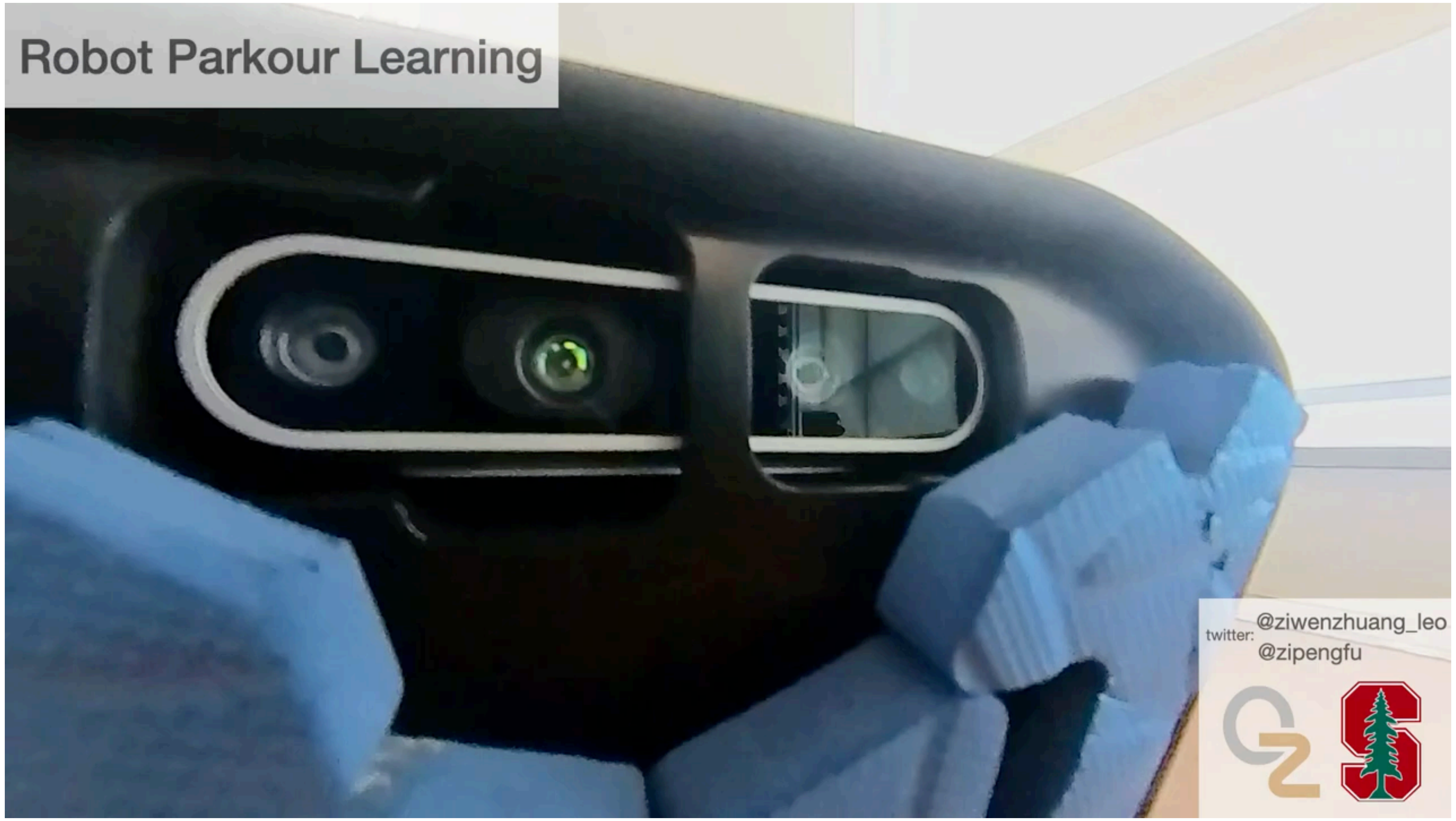
Elena Congeduti, 09-12-2024

TU Delft

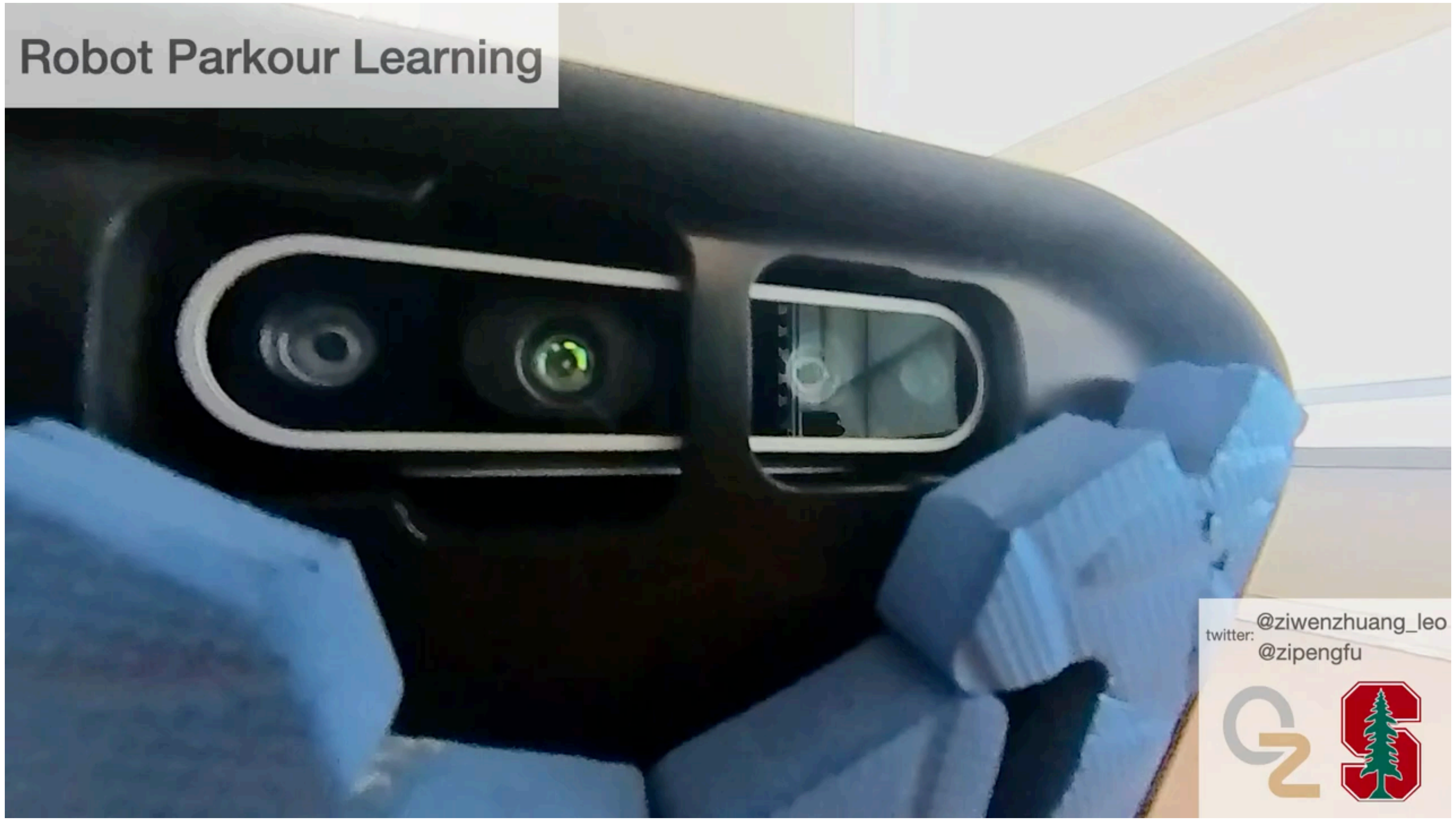
Lecture’s Agenda

- Sequential decision making
- Markov decision processes (MDPs)
- Value iteration
- Tabular Q-learning

What is Reinforcement Learning?

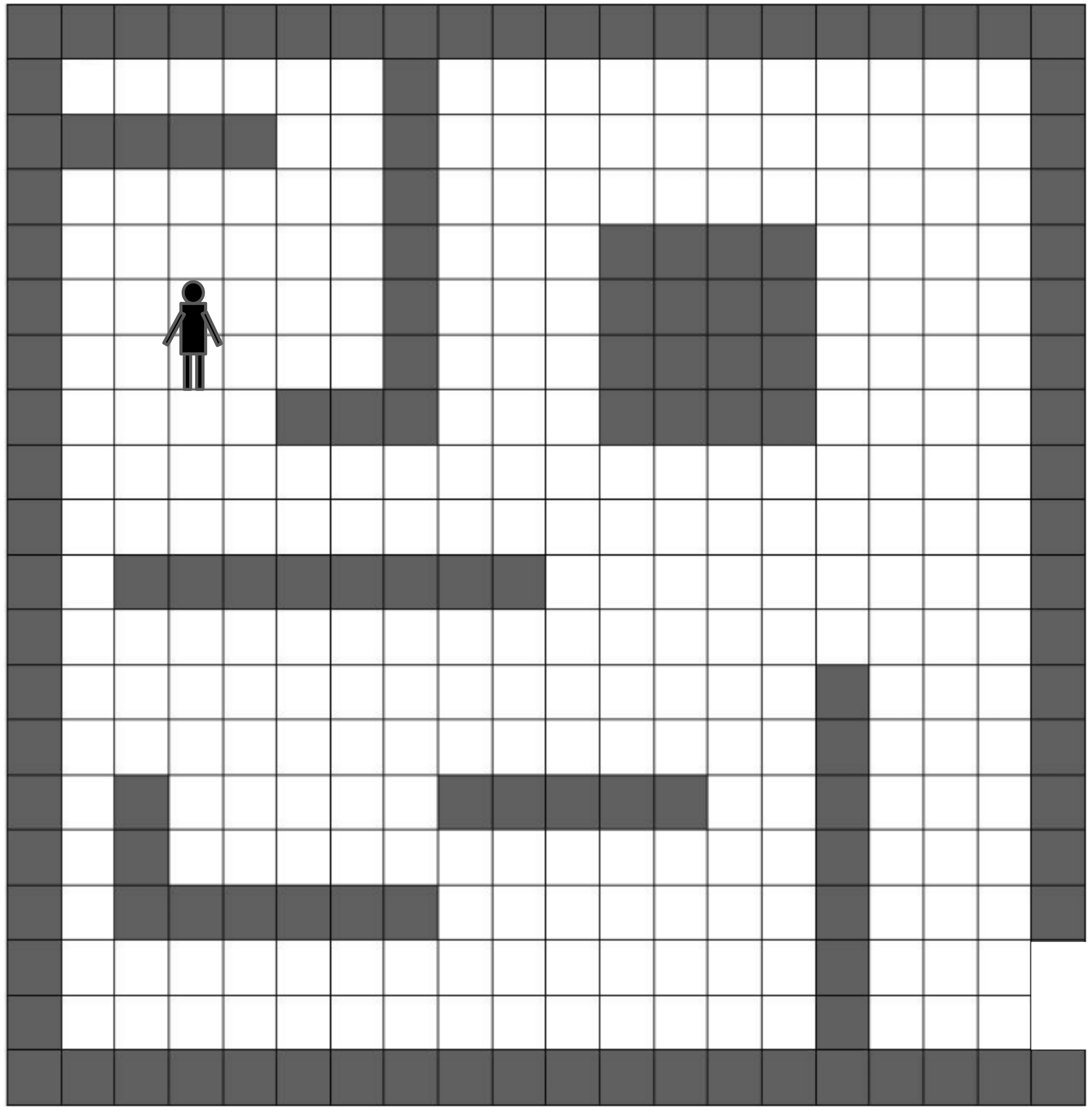


What is Reinforcement Learning?



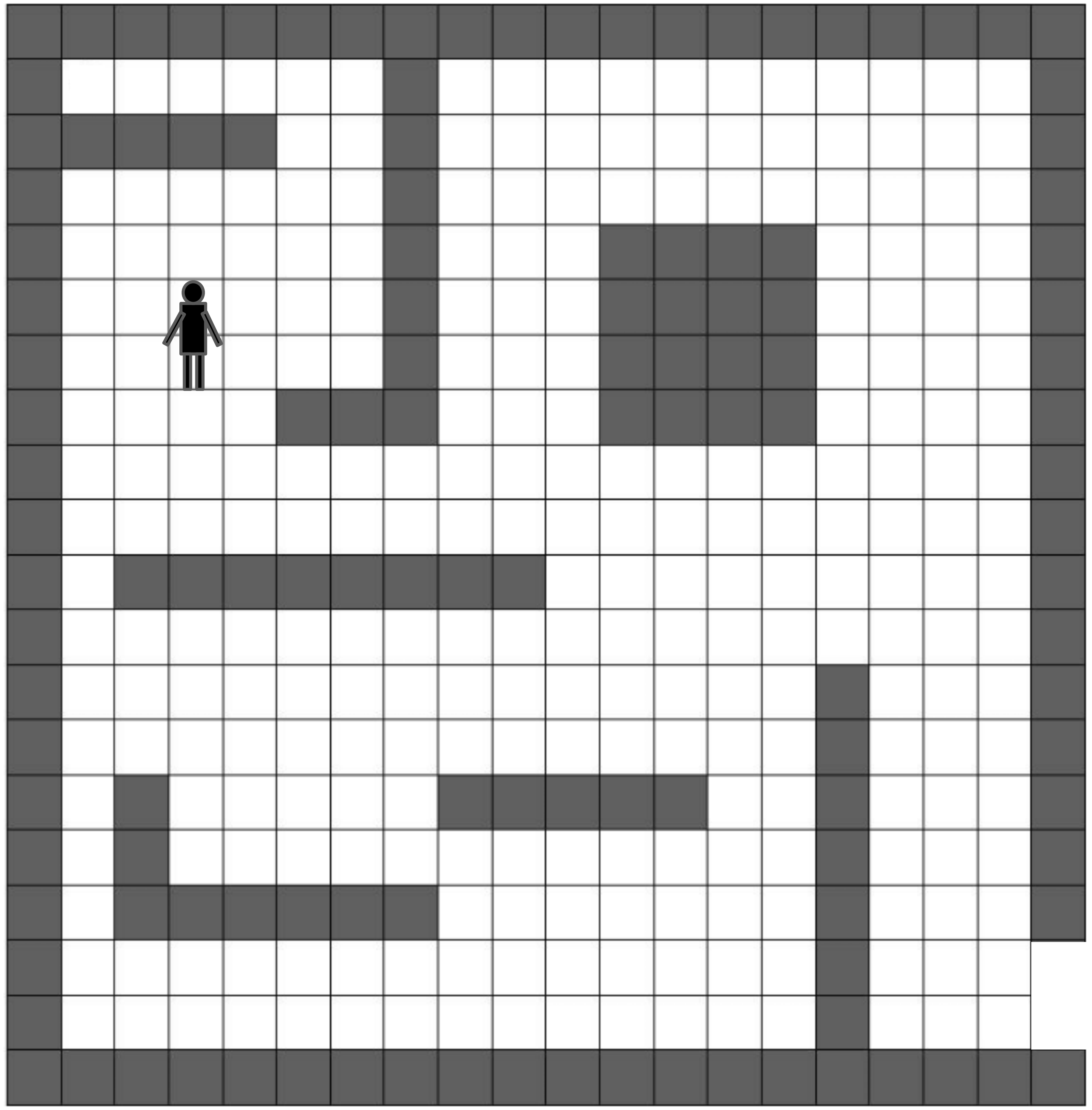
Intelligent agent learns from experience how to make decisions that maximize its return in the face of uncertainty

Robot Navigation



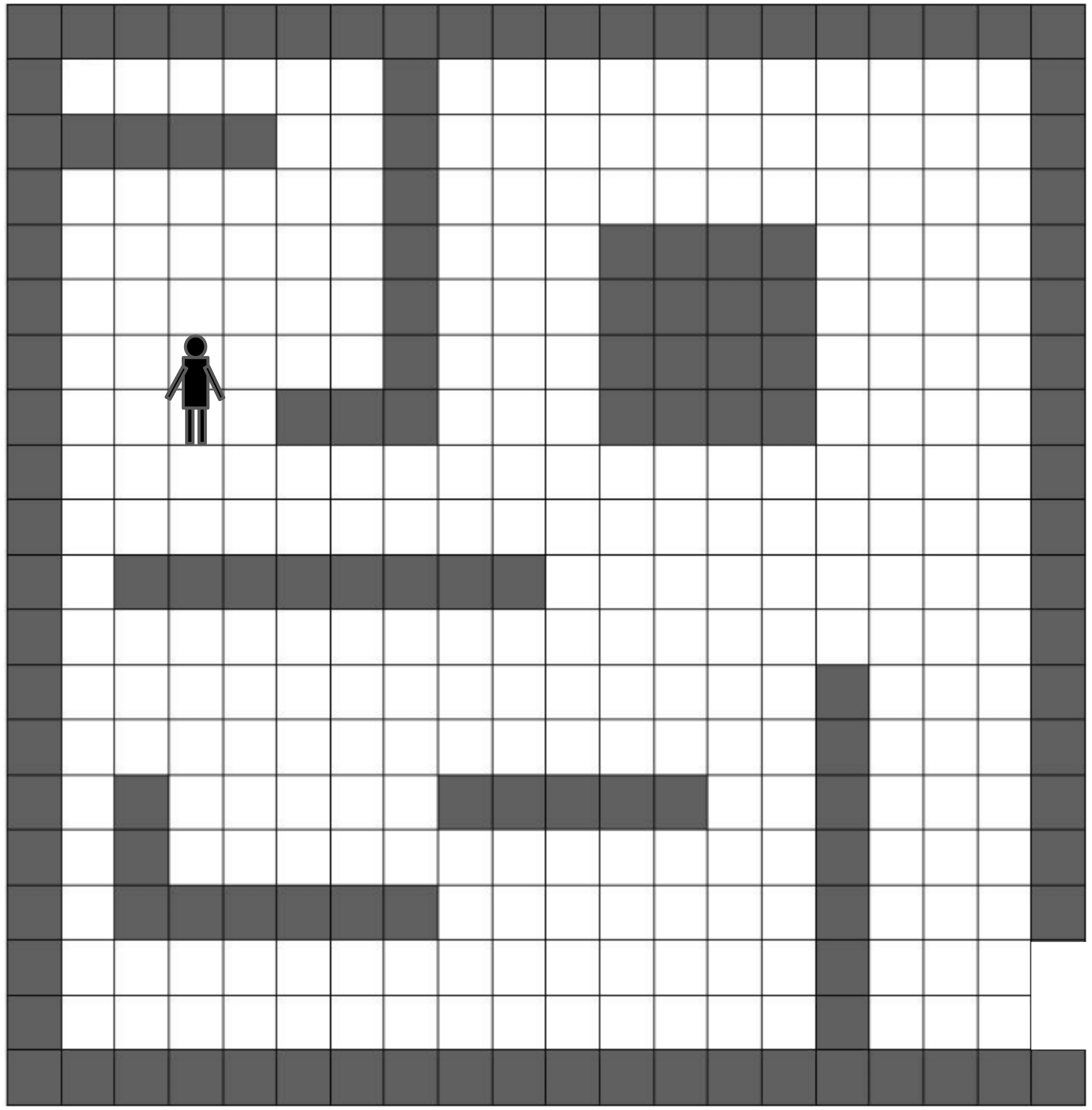
- Robot moving in a maze

Robot Navigation



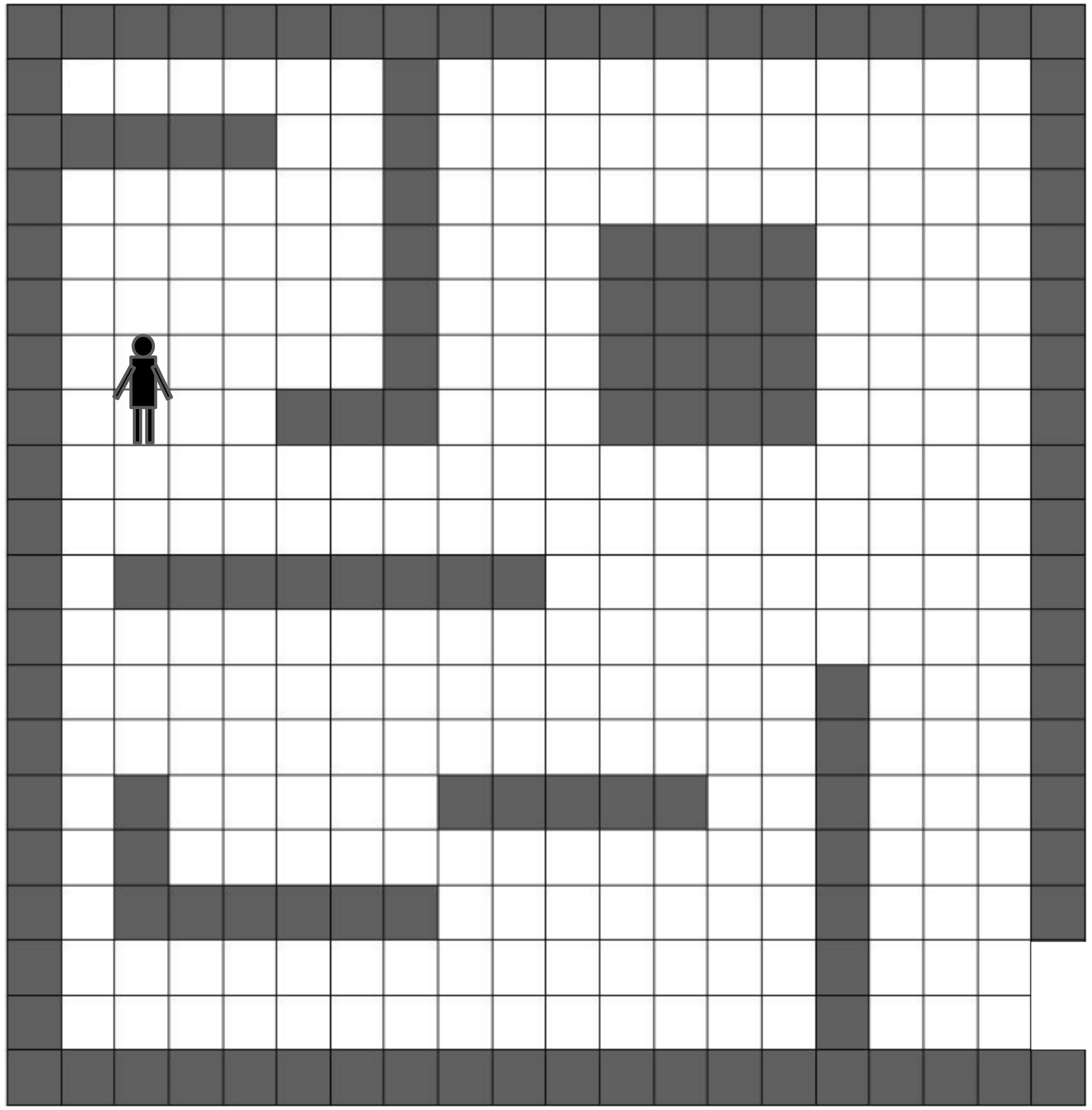
- Robot moving in a maze

Robot Navigation



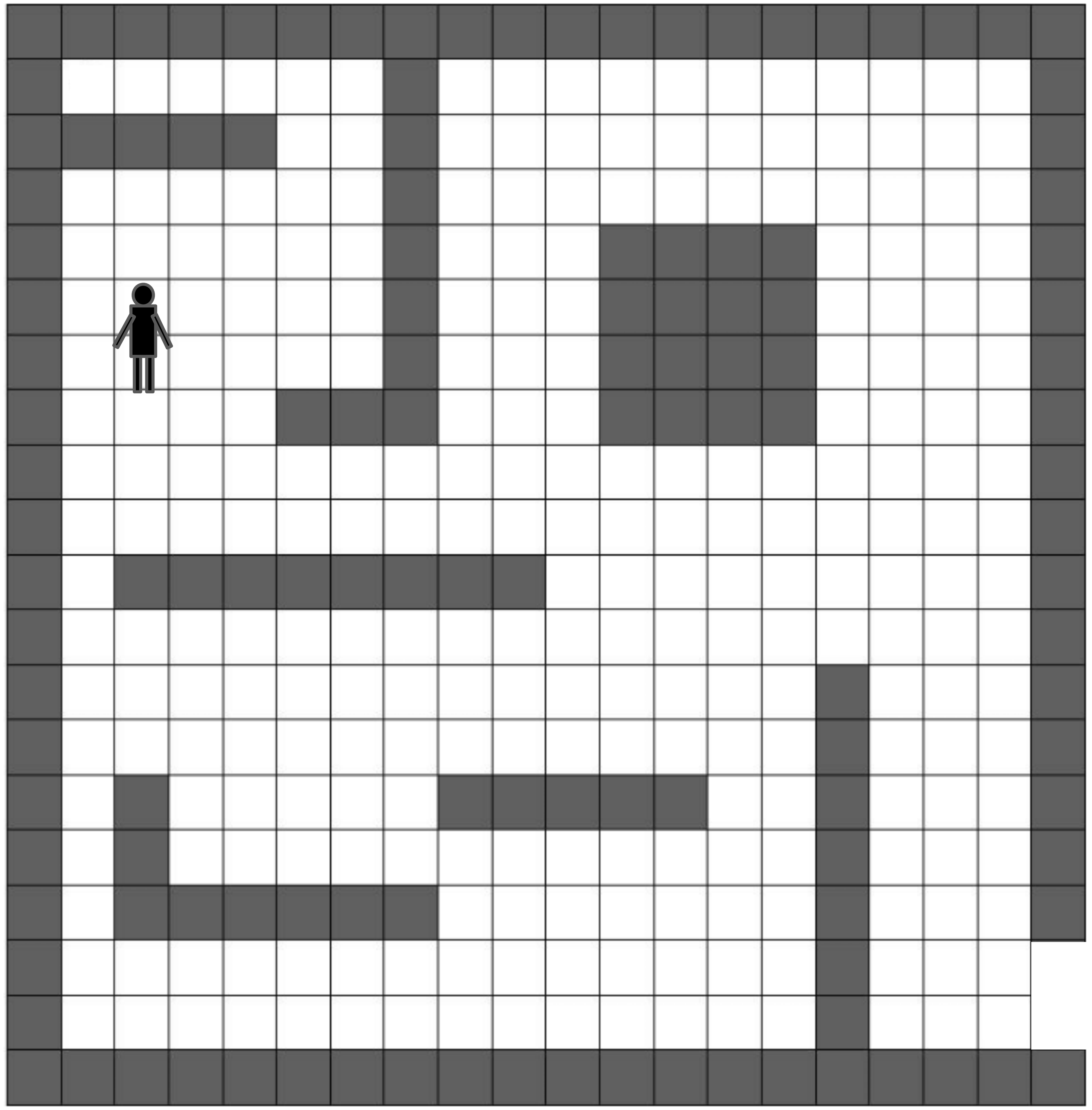
- Robot moving in a maze

Robot Navigation



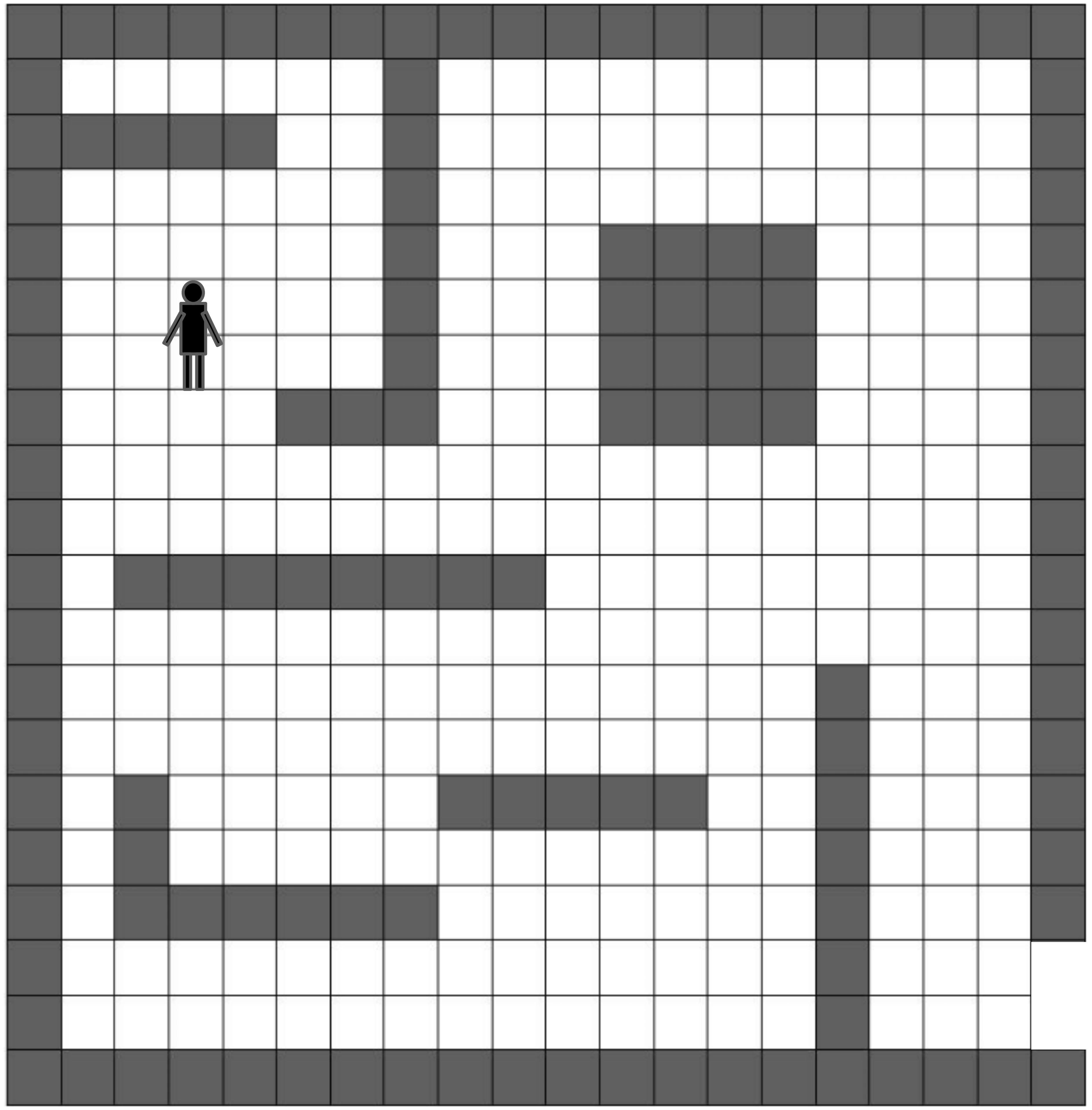
- Robot moving in a maze

Robot Navigation



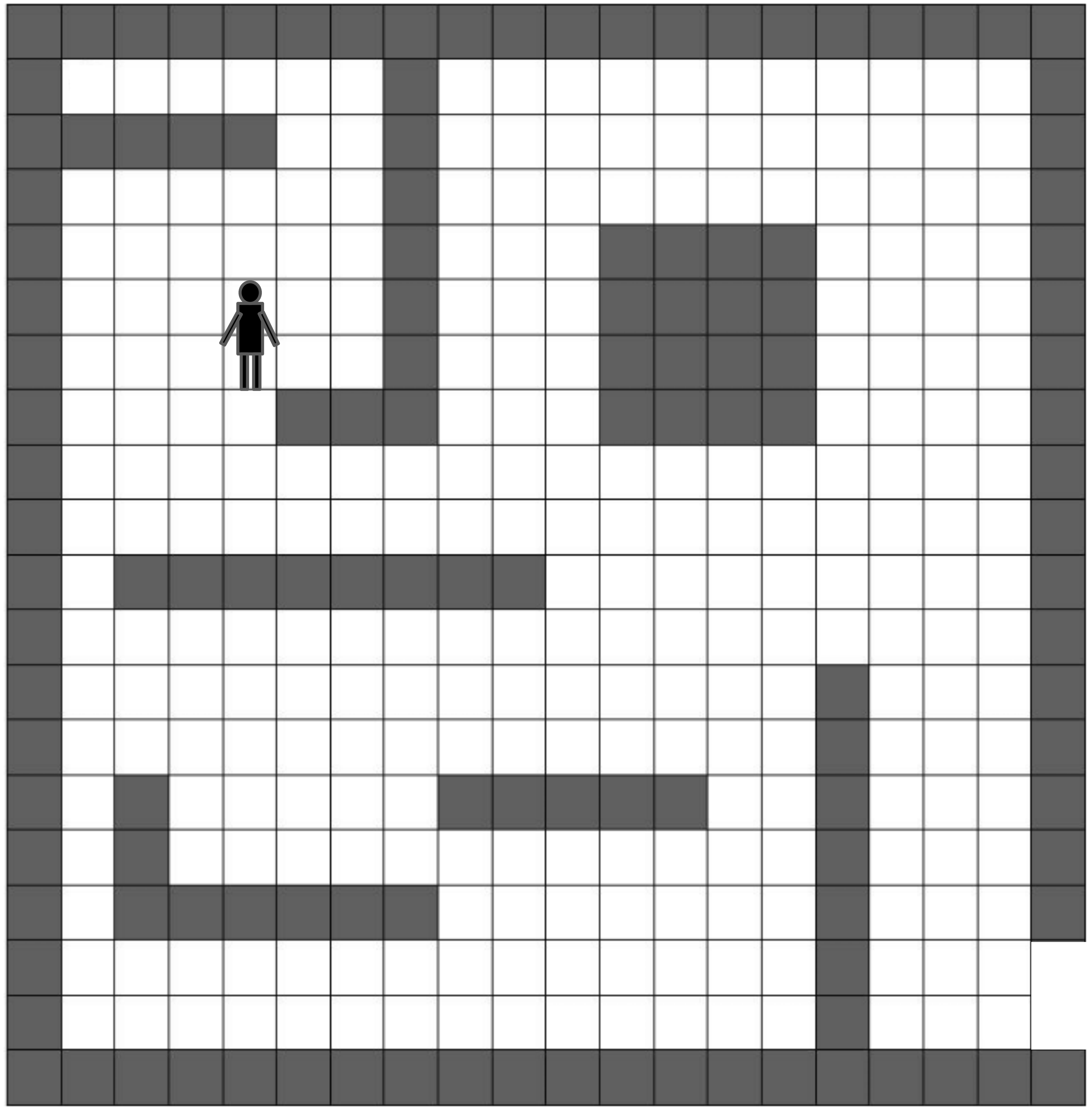
- Robot moving in a maze

Robot Navigation



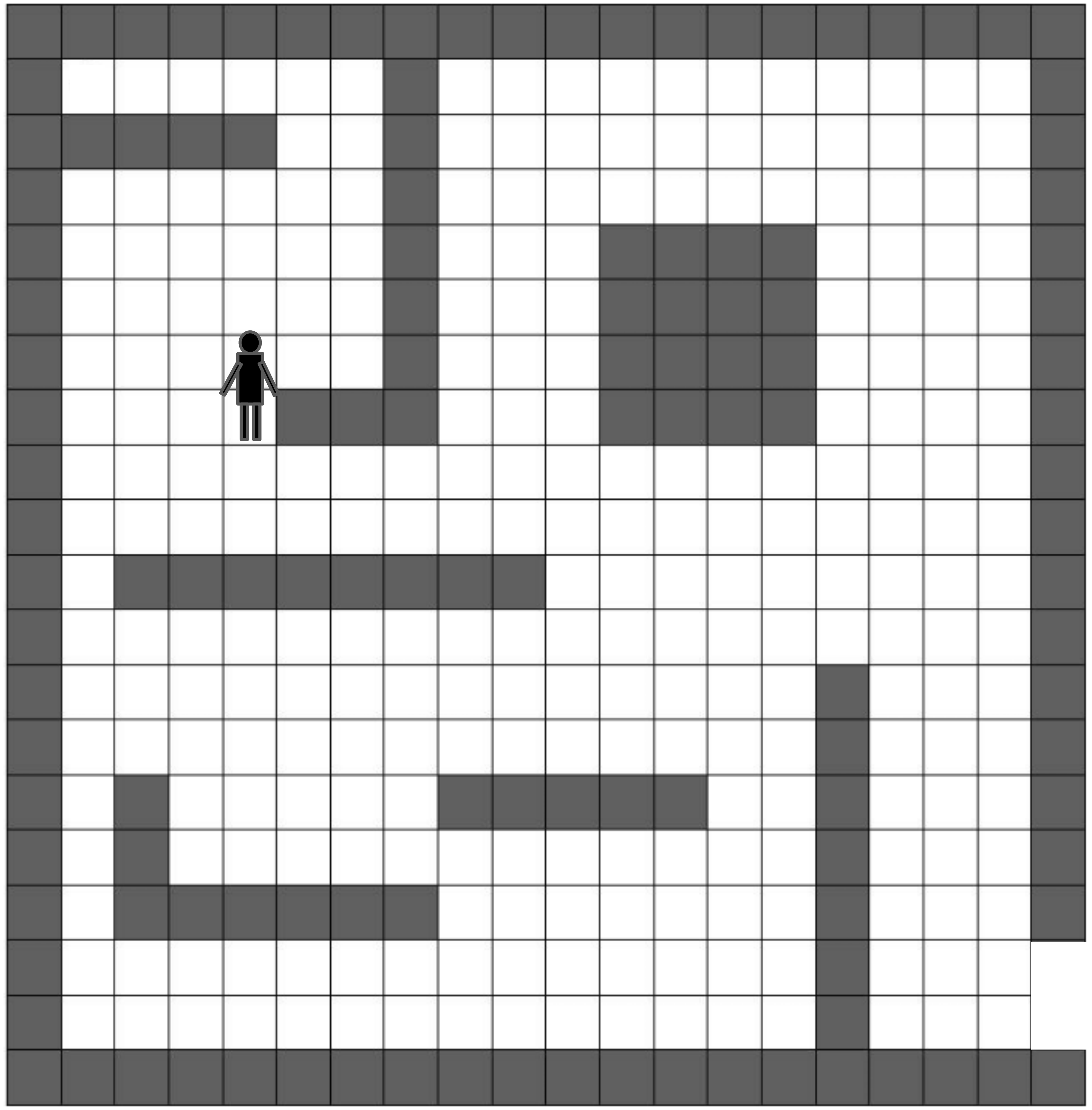
- Robot moving in a maze

Robot Navigation



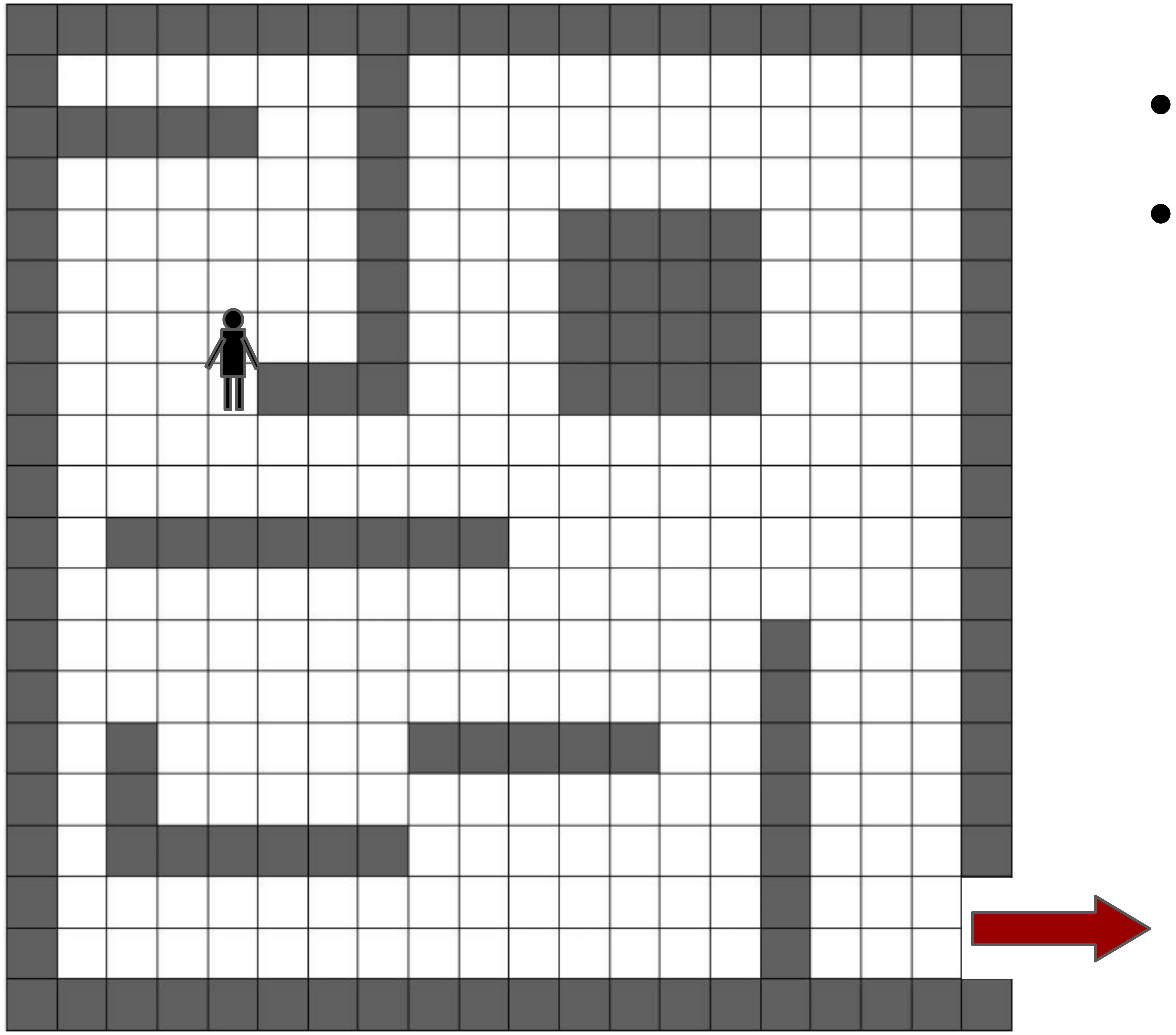
- Robot moving in a maze

Robot Navigation



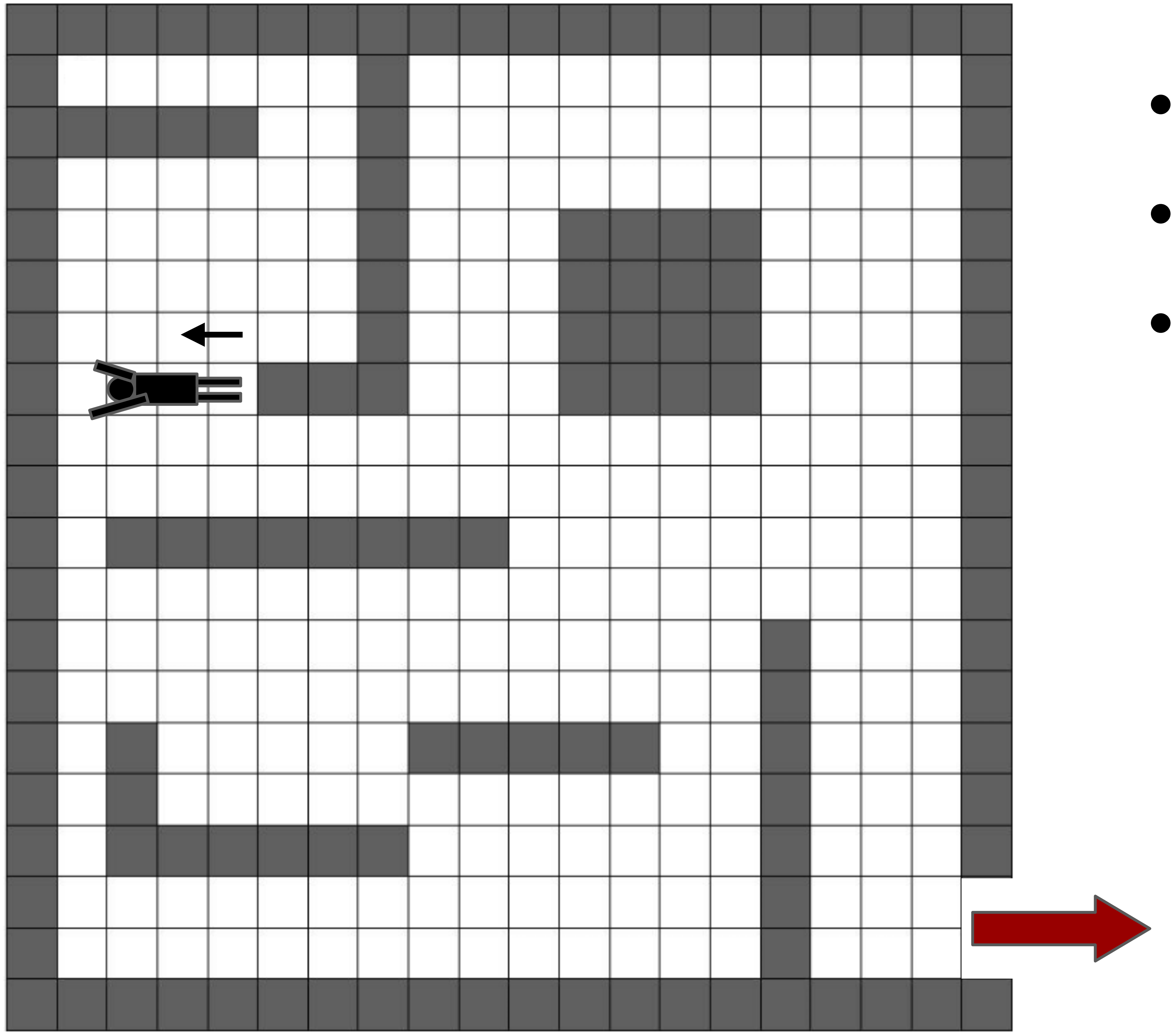
- Robot moving in a maze

Robot Navigation



- Robot moving in a maze
- Goal: find the exit

Robot Navigation

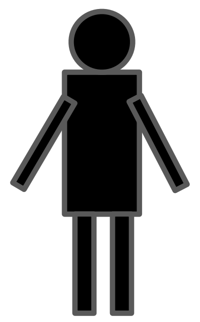


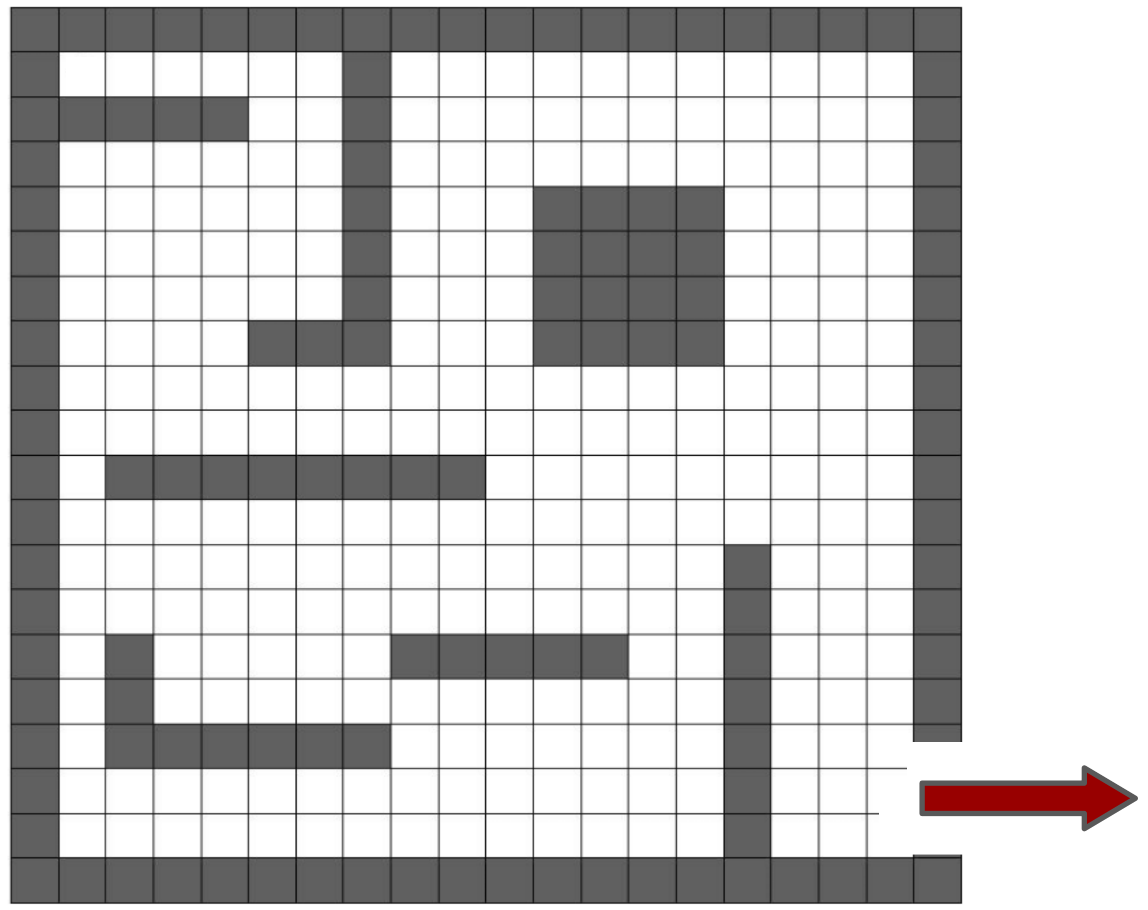
- Robot moving in a maze
- Goal: find the exit
- Chance of failing

Sequential Decision Making

Robot Navigation

agent

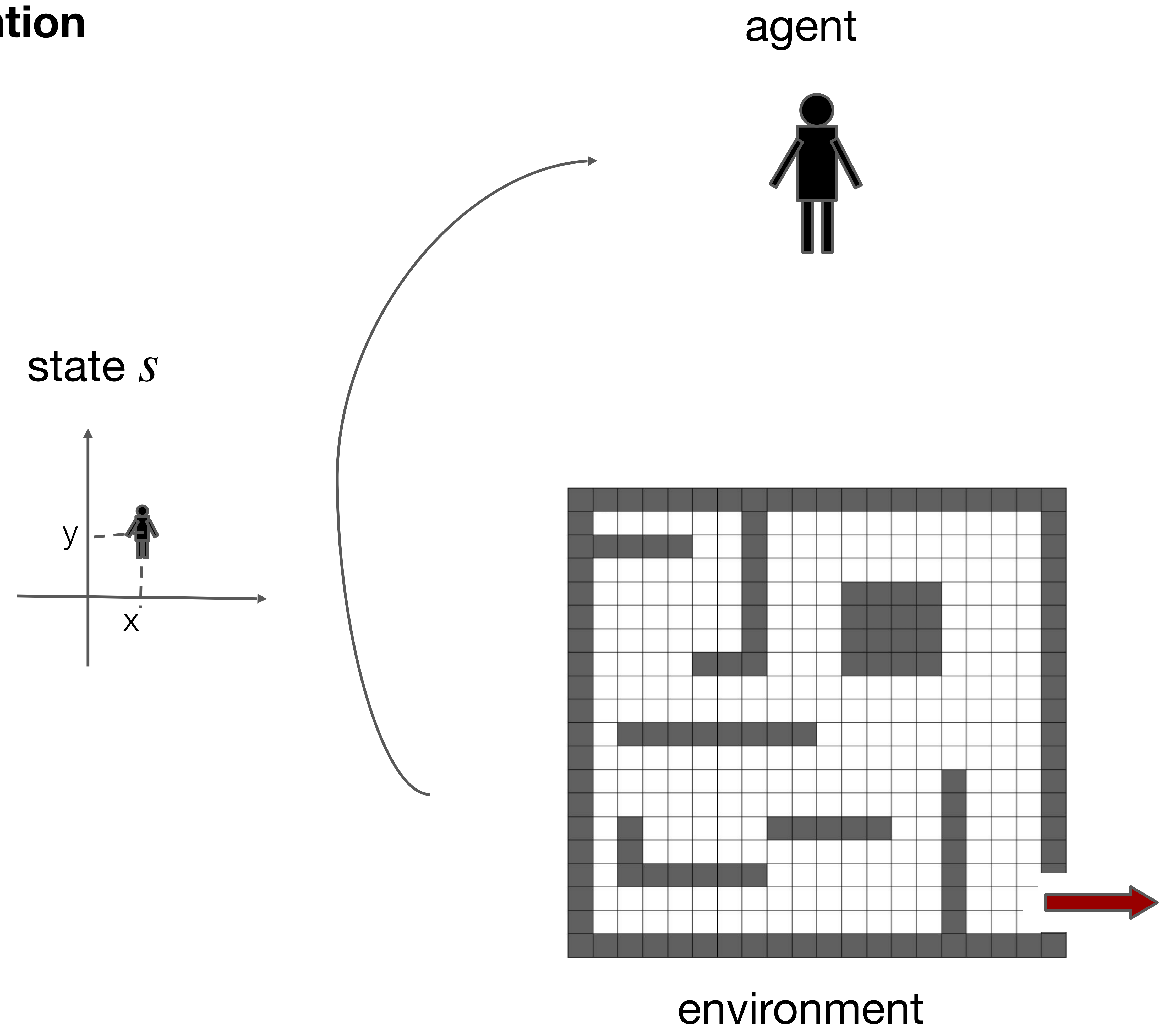




environment

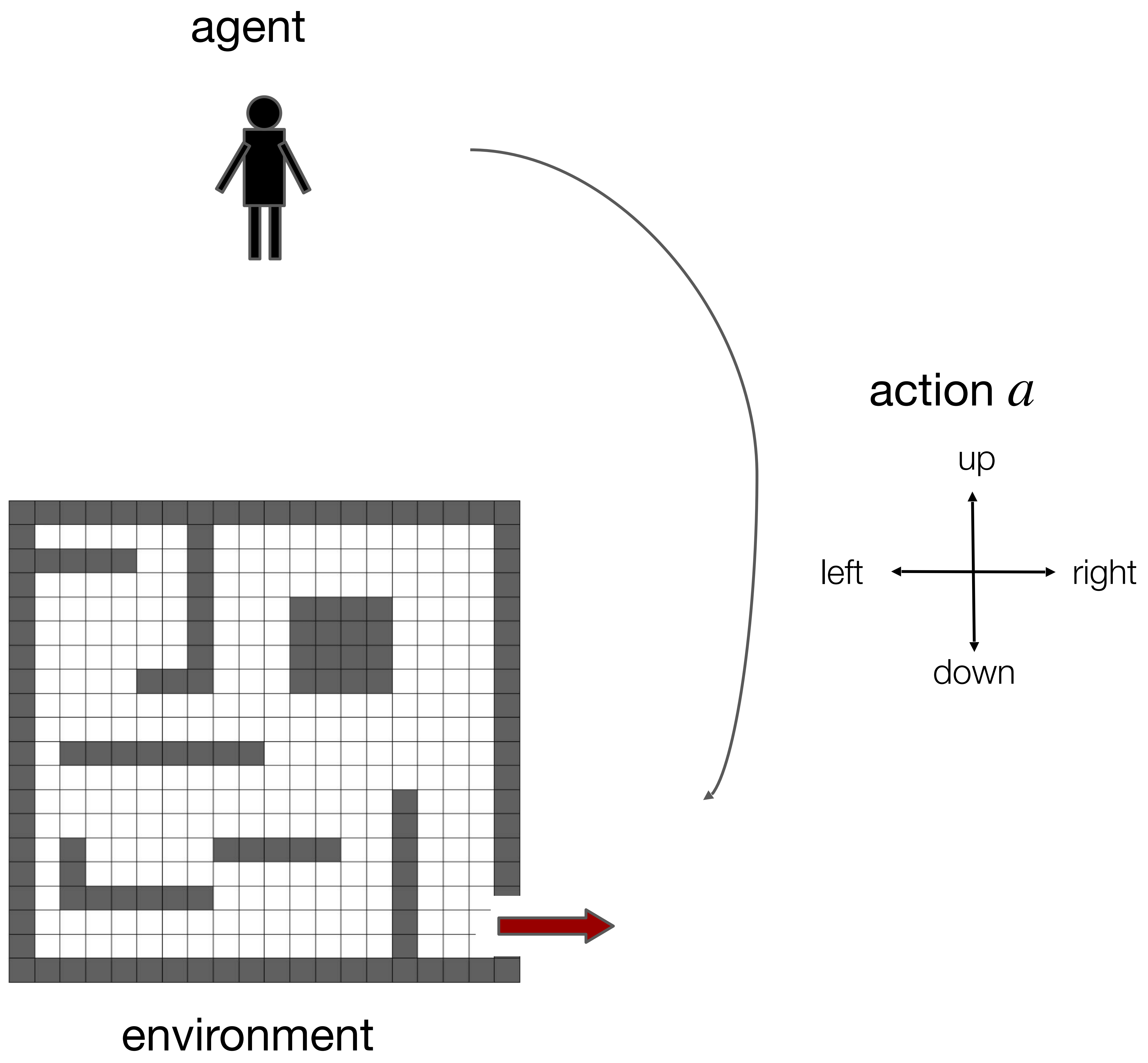
Sequential Decision Making

Robot Navigation



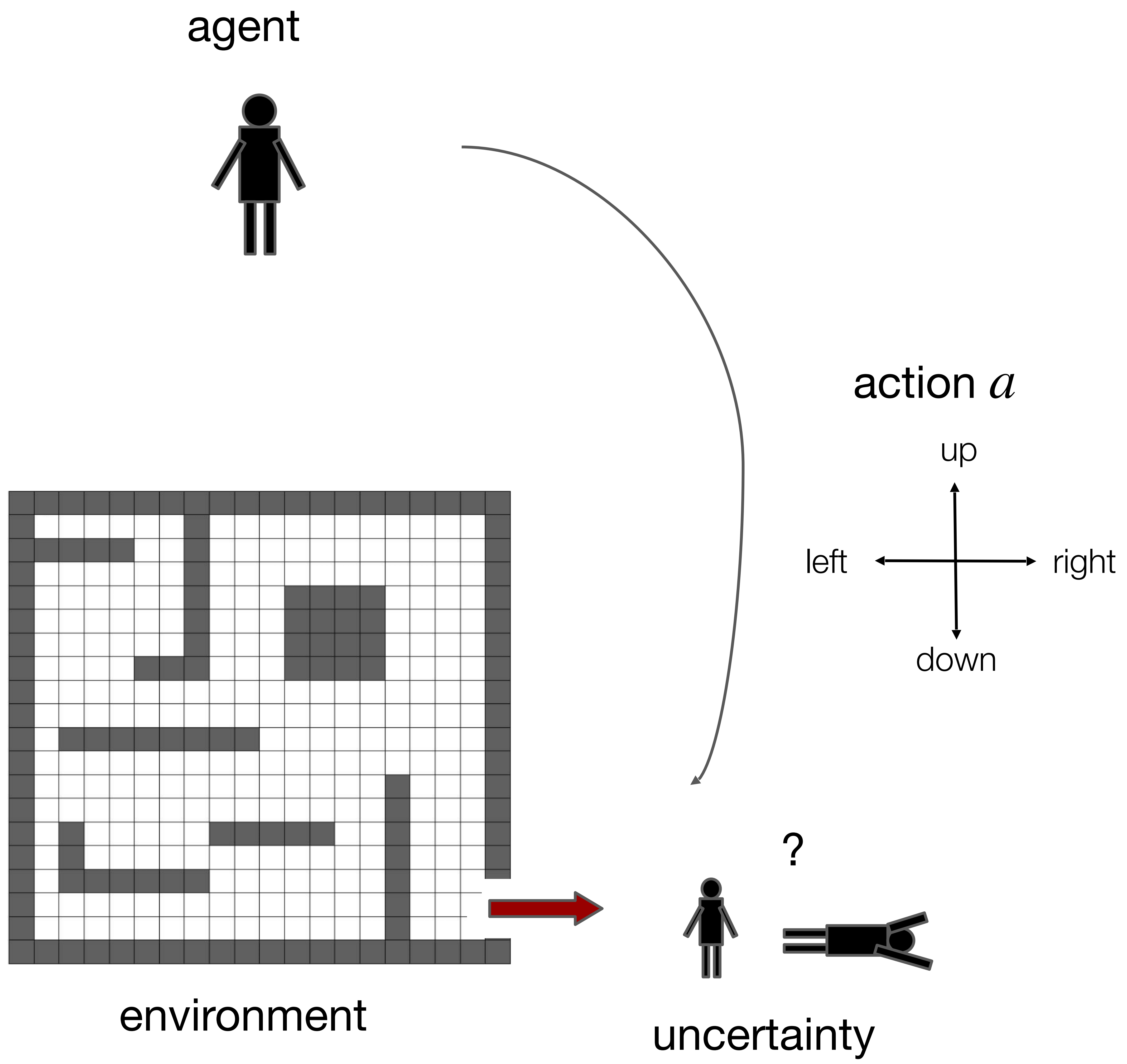
Sequential Decision Making

Robot Navigation



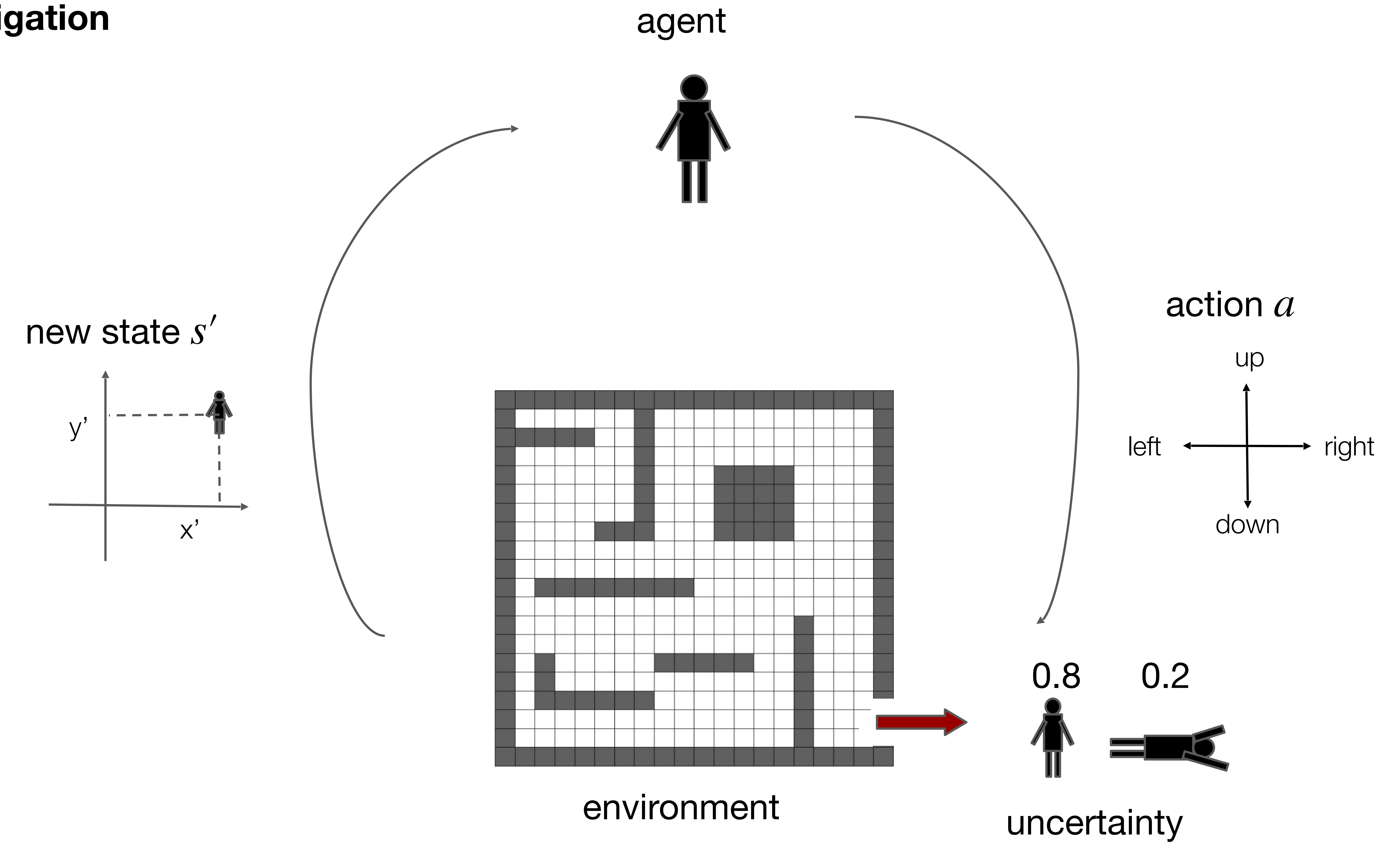
Sequential Decision Making

Robot Navigation



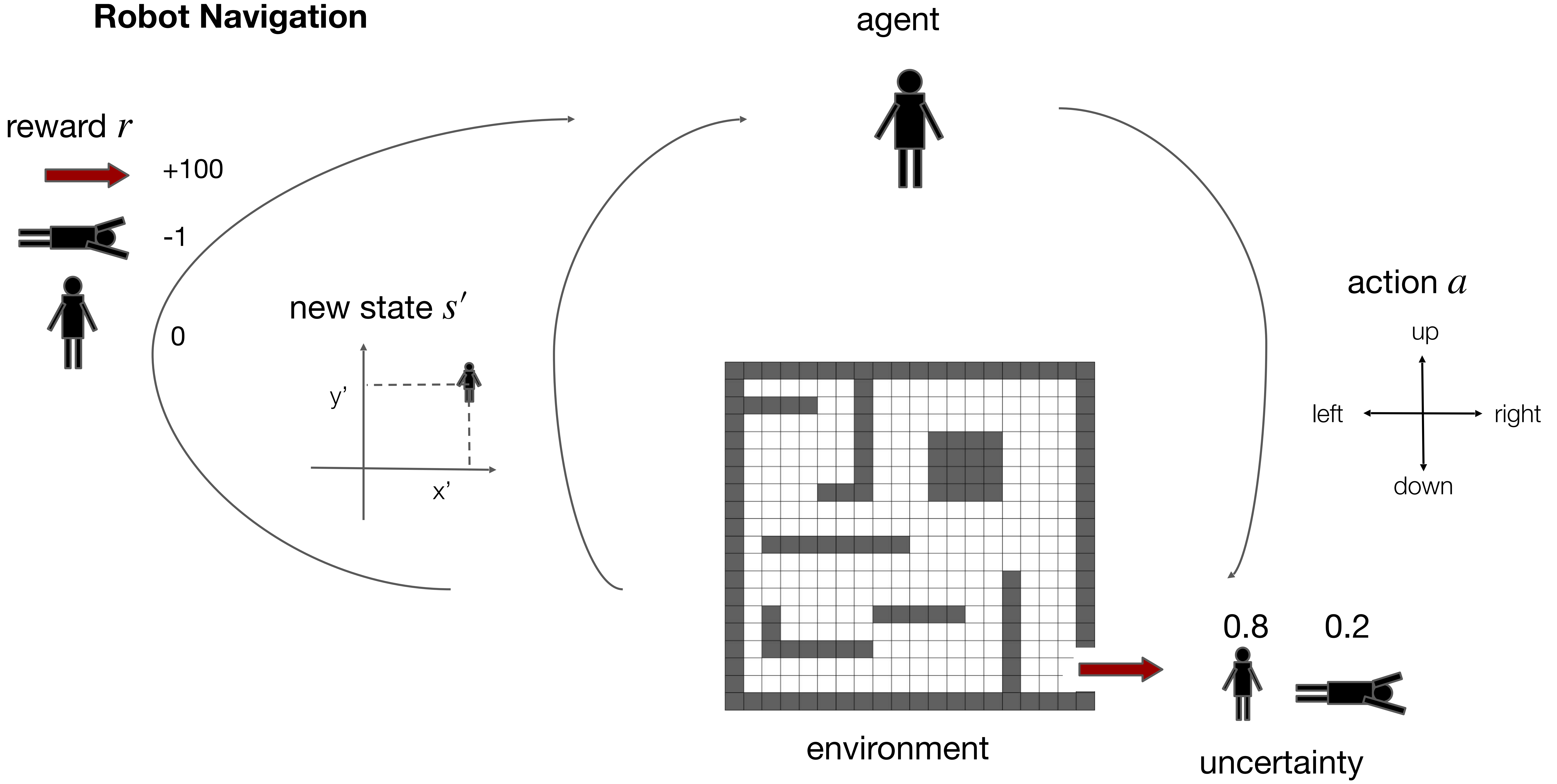
Sequential Decision Making

Robot Navigation

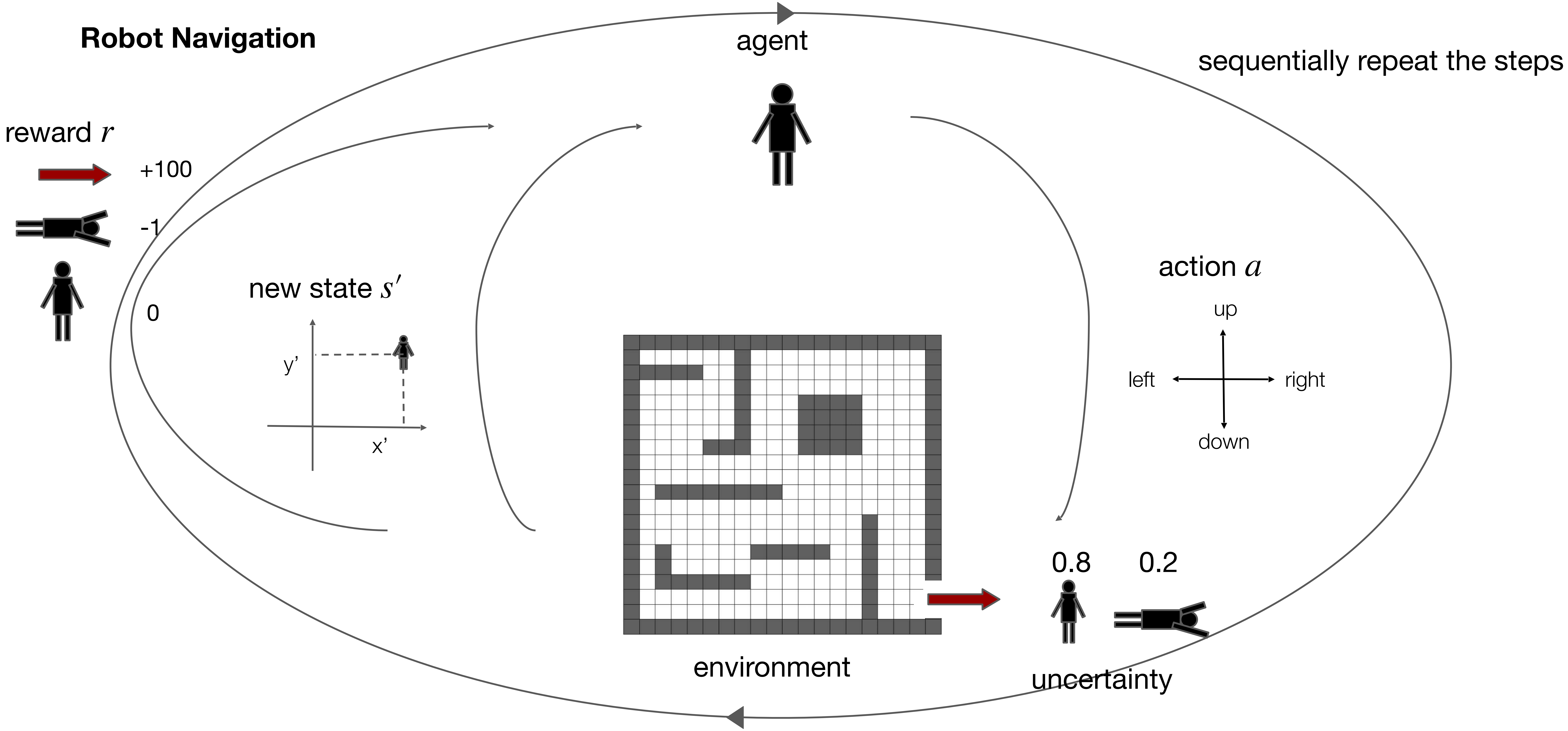


Sequential Decision Making

Robot Navigation



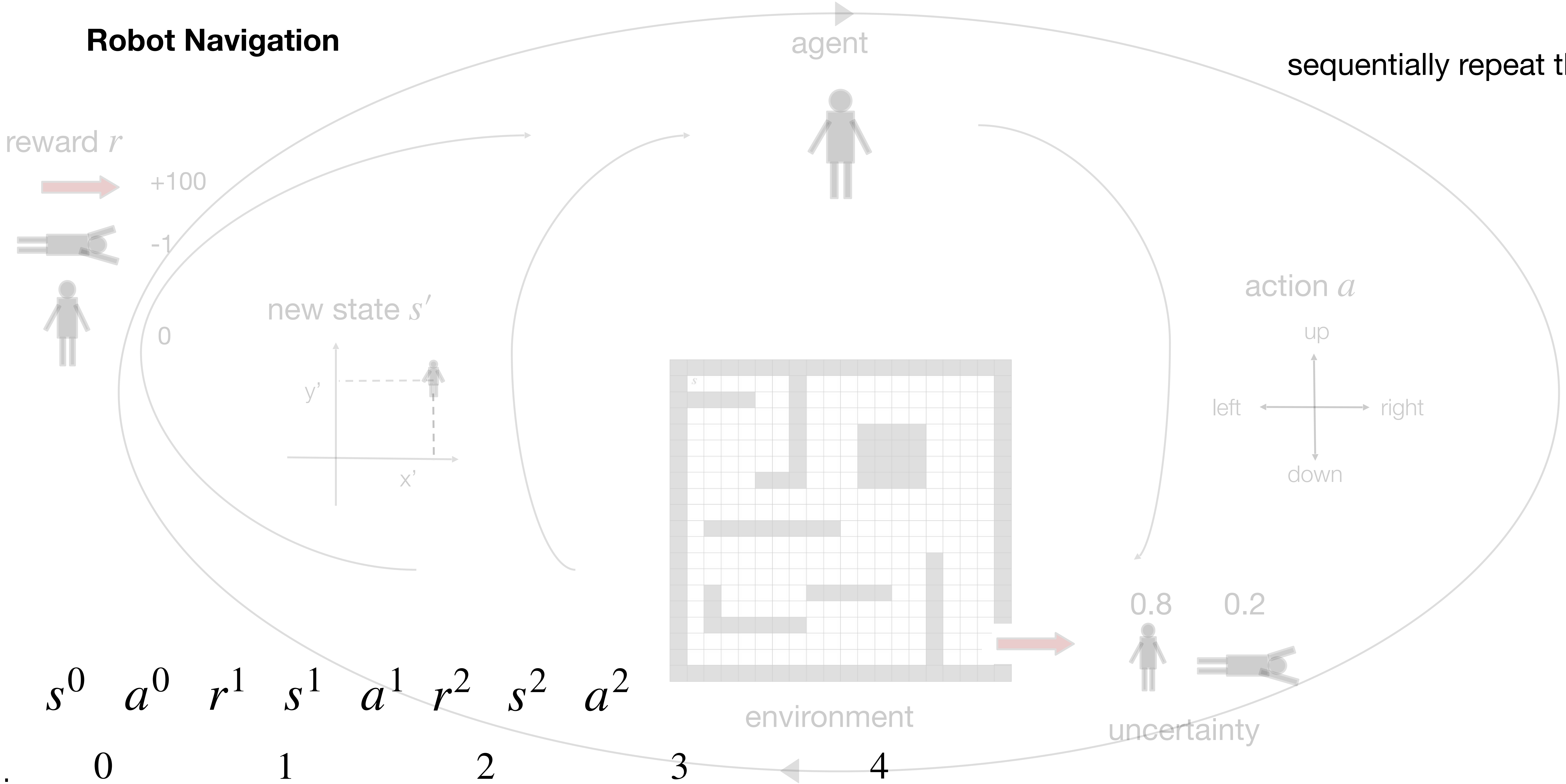
Sequential Decision Making



Sequential Decision Making

Robot Navigation

sequentially repeat the steps

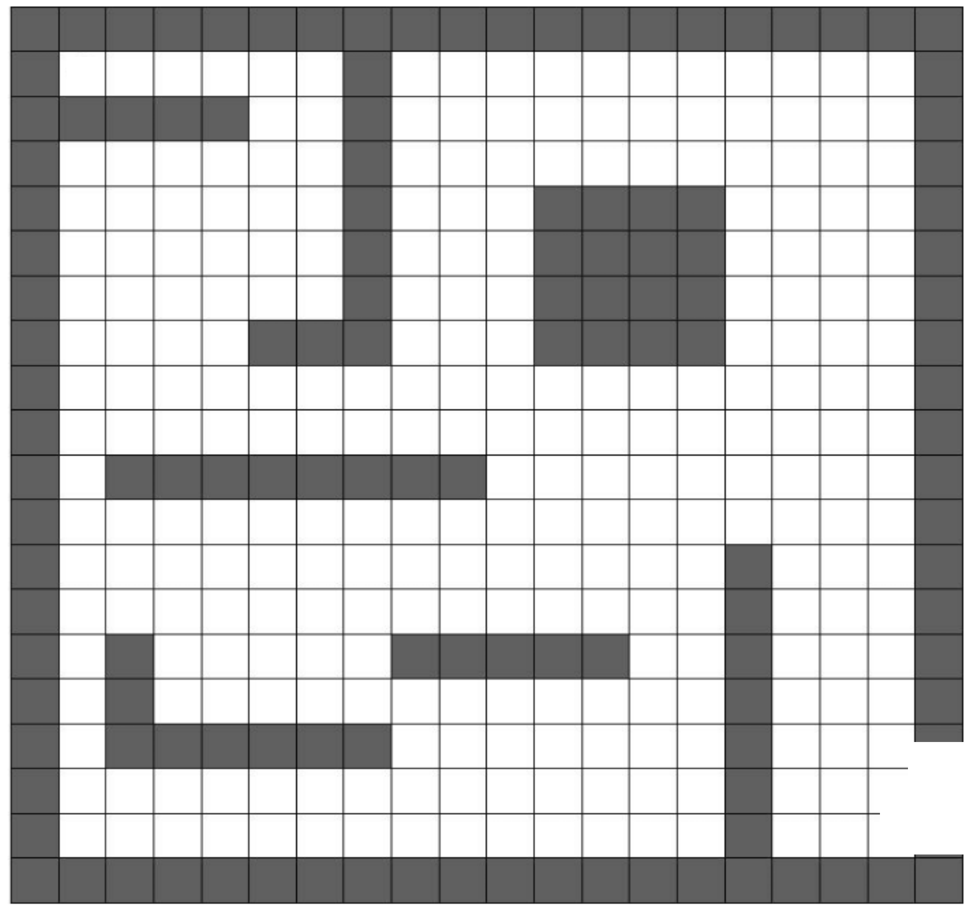


$s^0 \quad a^0 \quad r^1 \quad s^1 \quad a^1 \quad r^2 \quad s^2 \quad a^2$

time 0 1 2 3 4

Markov Decision Processes

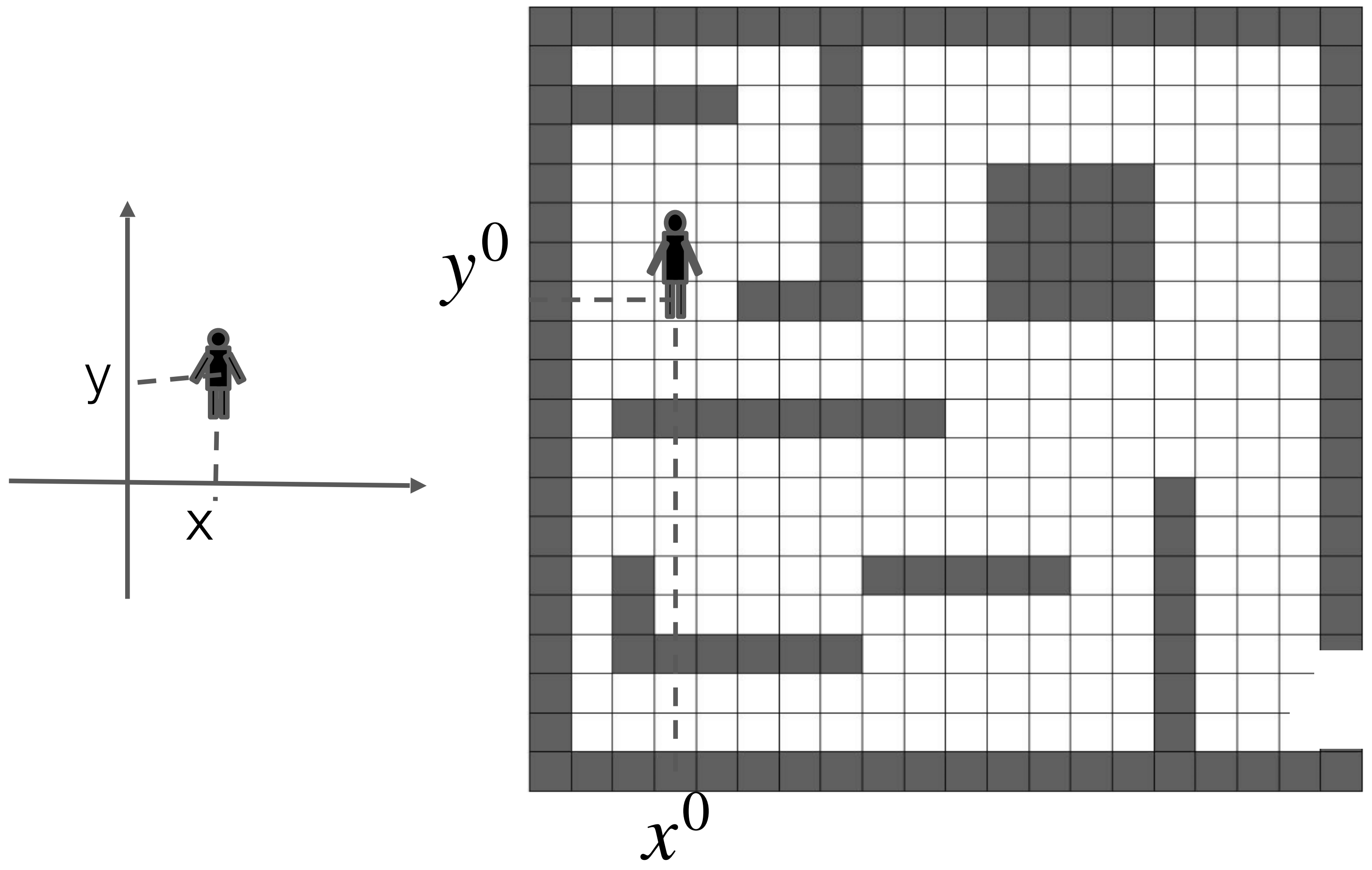
Markov decision process (MDP) $M = (S, A, s^0, P, R)$



Markov Decision Processes

Markov decision process (MDP) $M = (S, A, s^0, P, R)$

- State space S of the environment
- Initial state $s^0 \in S$



$S = \{\text{feasible } (x, y)\}$

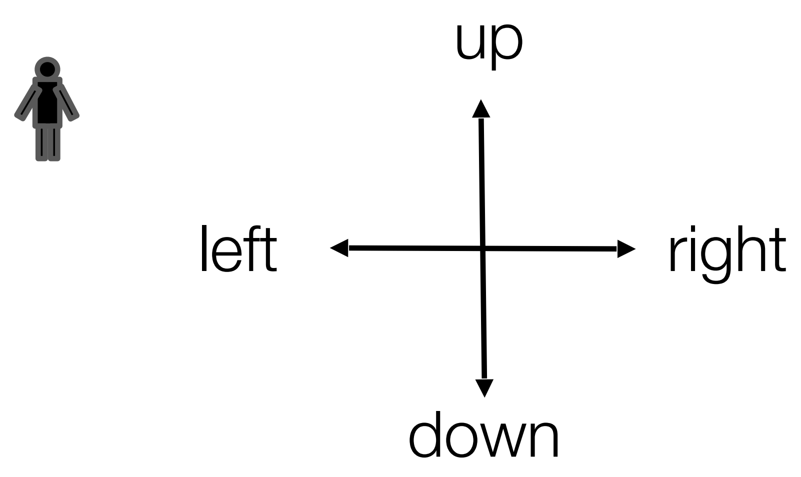
$s^0 = (x^0, y^0)$ initial robot coordinates

Markov Decision Processes

Markov decision process (MDP) $M = (S, A, s^0, P, R)$

- State space S of the environment
- Initial state $s^0 \in S$
- Action space A of possible agent actions

action



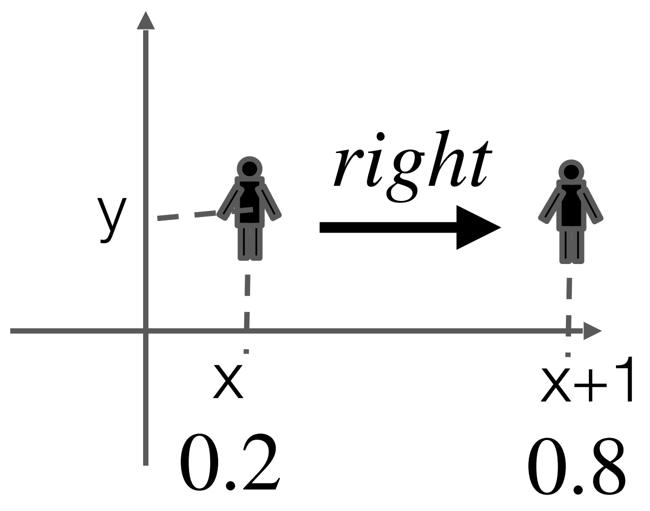
$A = \{right, down, left, up\}$

Markov Decision Processes

Markov decision process (MDP) $M = (S, A, s^0, P, R)$

- State space S of the environment
- Initial state $s^0 \in S$
- Action space A of possible agent actions
- Transition probabilities $P(s' \mid s, a)$ from state s to s' after choosing a

Transitions

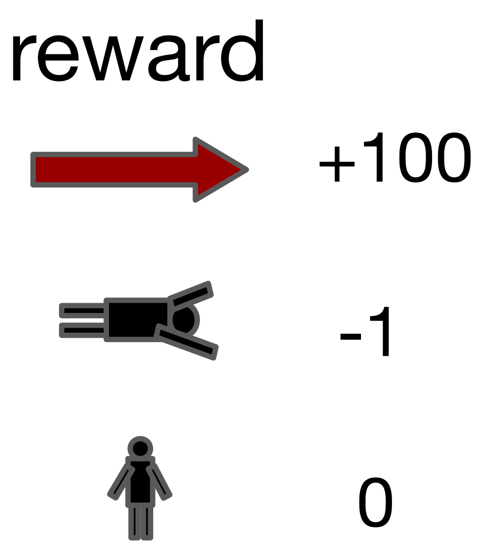


$$P(s' \mid a = right, s = (x, y)) = \begin{cases} 0.8 & \textit{if } s' = (x+1, y) \\ 0.2 & \textit{if } s' = (x, y) \\ 0 & \textit{otherwise} \end{cases}$$

Markov Decision Processes

Markov decision process (MDP) $M = (S, A, s^0, P, R)$

- State space S of the environment
- Initial state $s^0 \in S$
- Action space A of possible agent actions
- Transition probabilities $P(s' \mid s, a)$ from state s to s' after choosing a
- Reward $R(s', a, s)$ or $R(s, a)$ for resulting in state s' from state s after choosing action a



$$R(s', a, s) = \begin{cases} 100 & \textit{if } s' = s_{goal} \\ -1 & \textit{if } s = s' \\ 0 & \textit{otherwise} \end{cases}$$

Policy

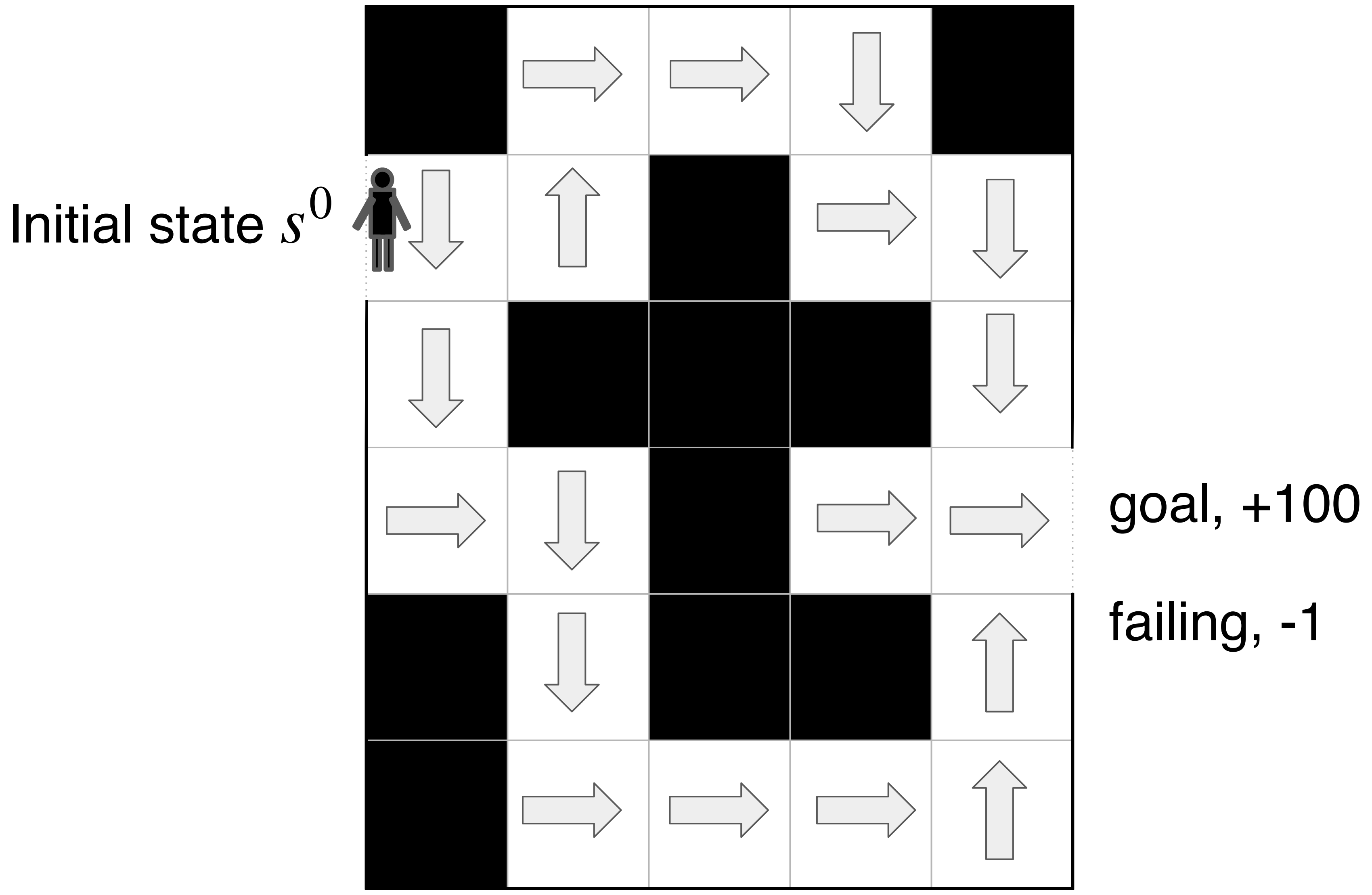
- A policy π encodes the agent's behaviour

- A map from states to actions, $\pi(s) = a$ is the agent's action when the environment state is s

Policy

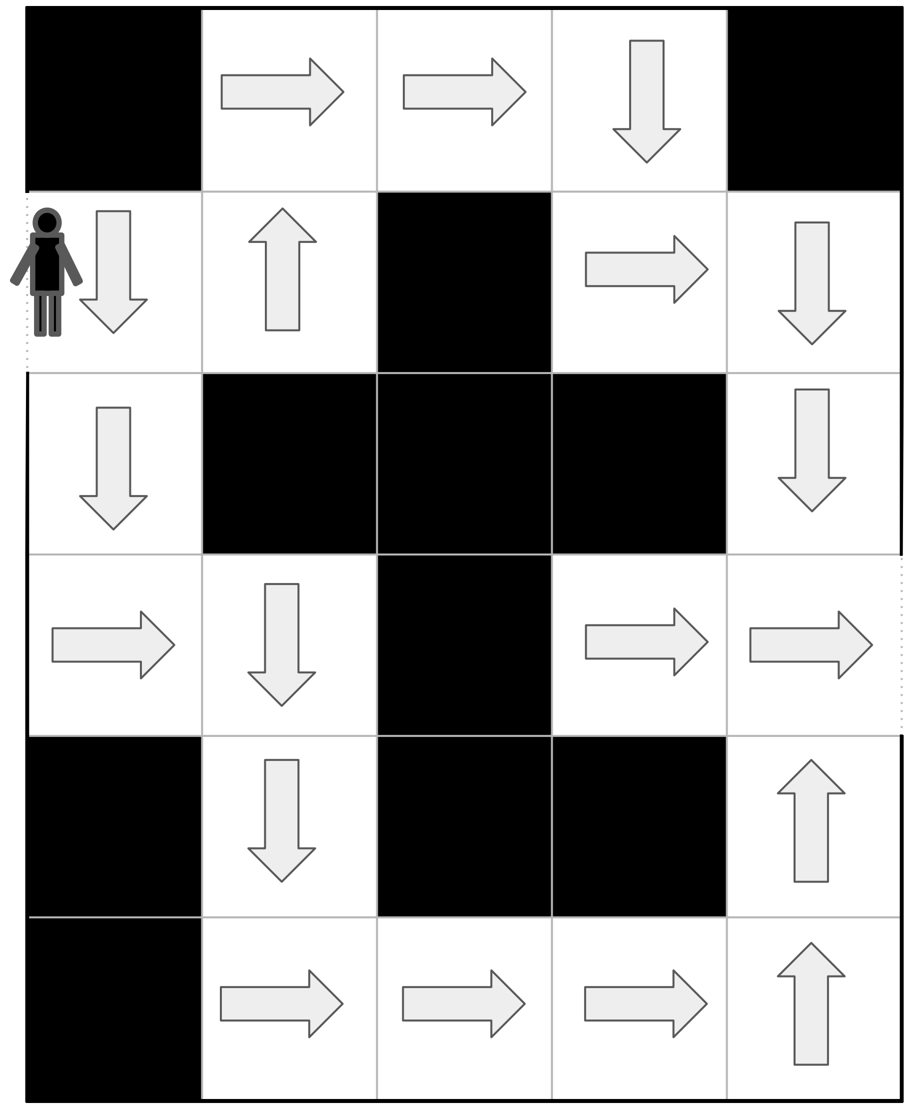
- A policy π encodes the agent's behaviour

- A map from states to actions, $\pi(s) = a$ is the agent's action when the environment state is s



Policy

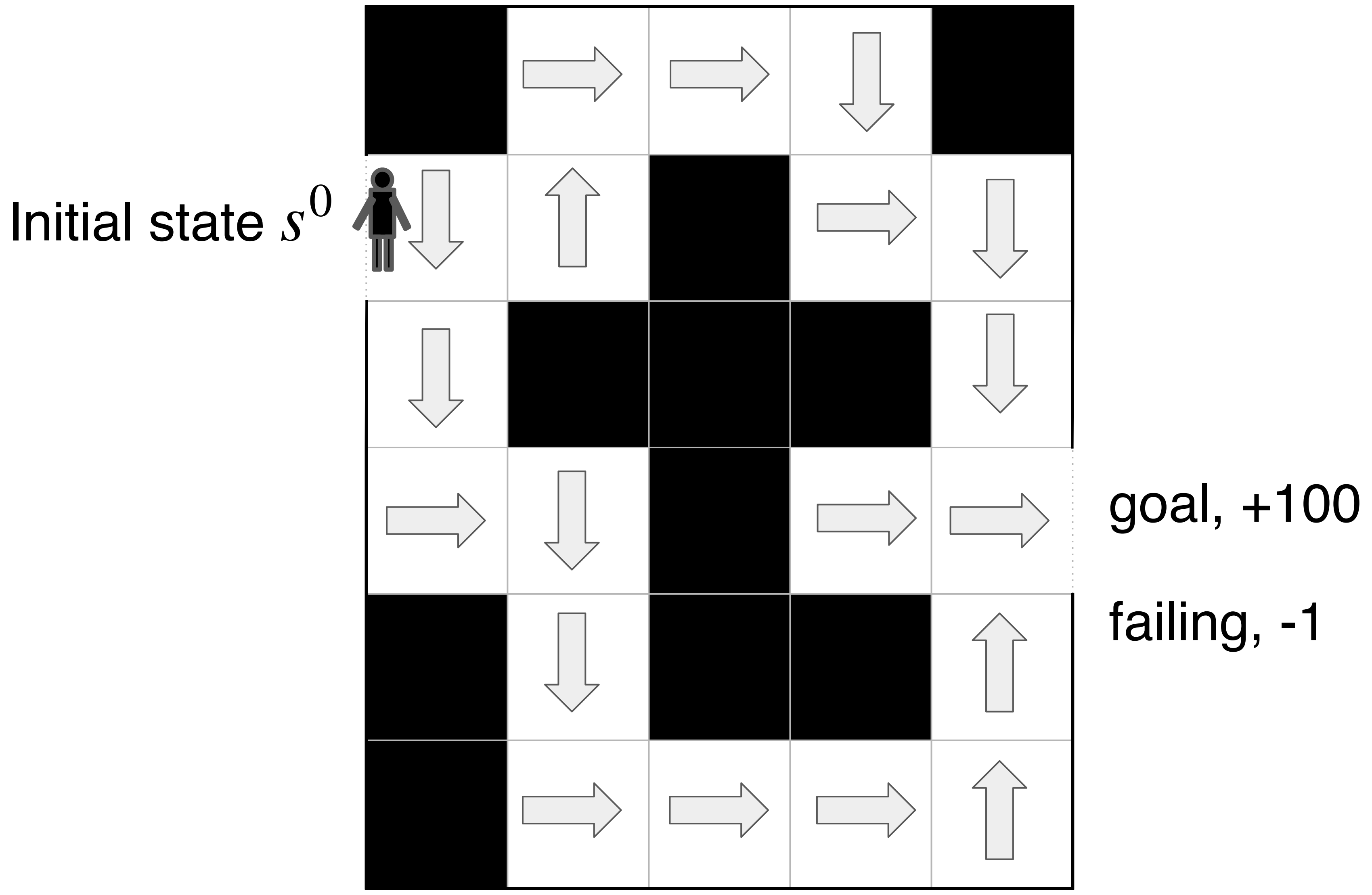
- A policy π encodes the agent's behaviour
- A map from states to actions, $\pi(s) = a$ is the agent's action when the environment state is s



How good is this policy?

Value

$V^{\pi}(s)$ is the expected sum of discounted future rewards for employing a policy π starting from an initial state s

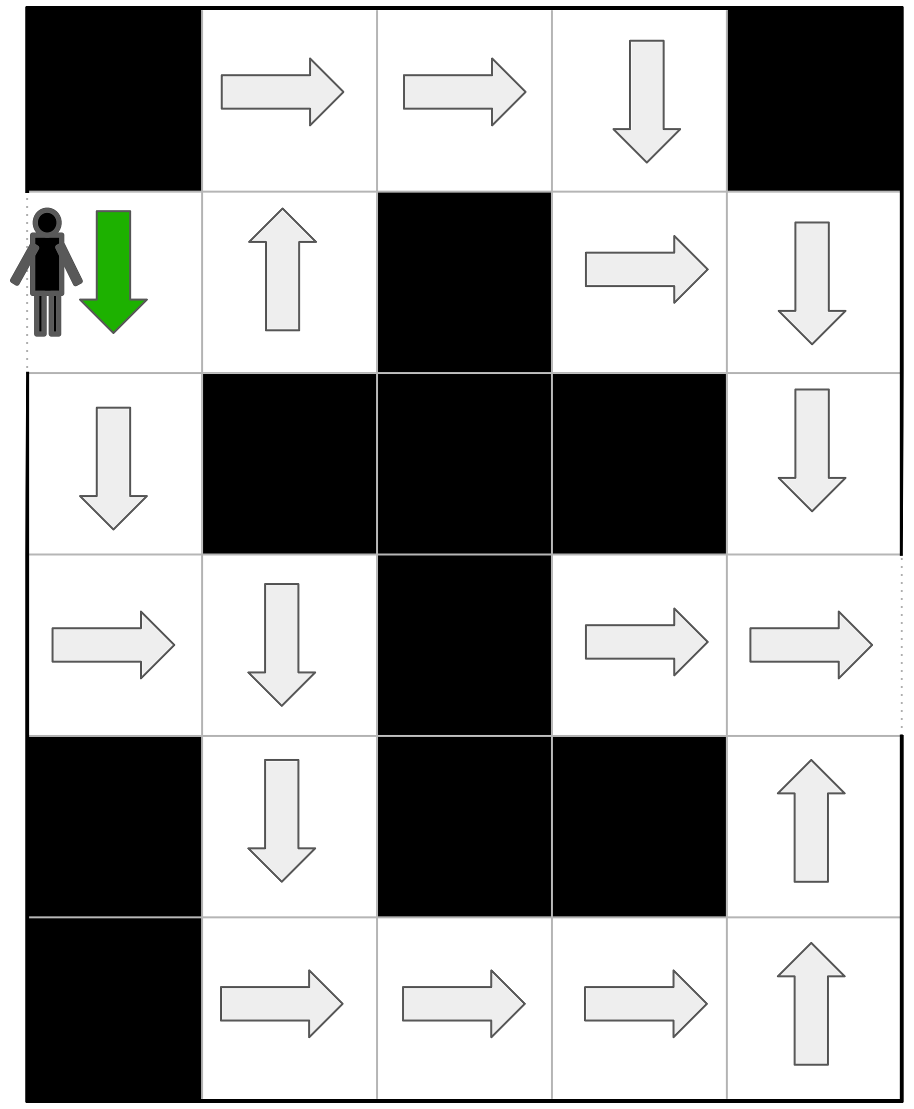


Value

$V^{\pi}(s)$ is the expected sum of discounted **future rewards** for employing a policy π starting from an initial state s

$$r^1, \;\; r^2, \;\; r^3 \ldots$$

Initial state s^0



goal, +100

failing, -1

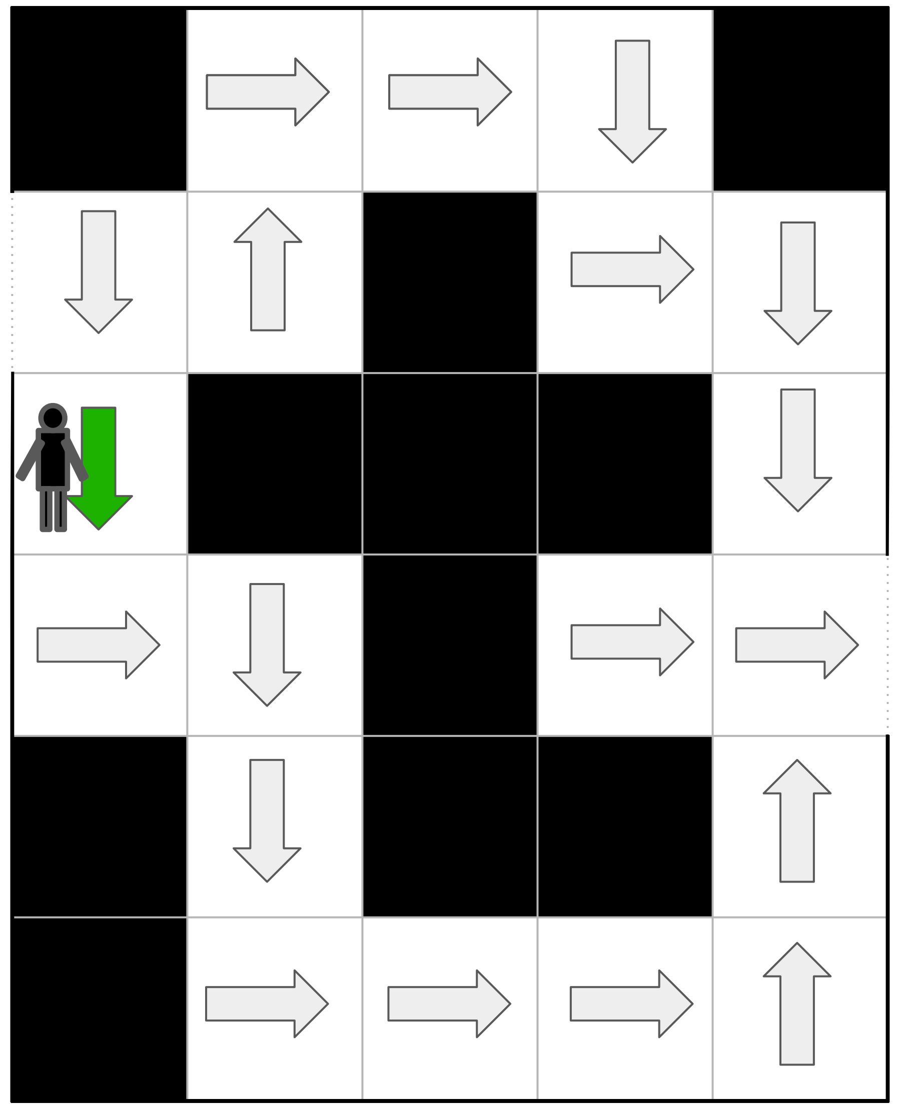
Value

$V^{\pi}(s)$ is the expected sum of discounted **future rewards** for employing a policy π starting from an initial state s

$$0, \quad r^2, \quad r^3 \ldots$$

$r^1 = 0$

Initial state s^0



goal, +100

failing, -1

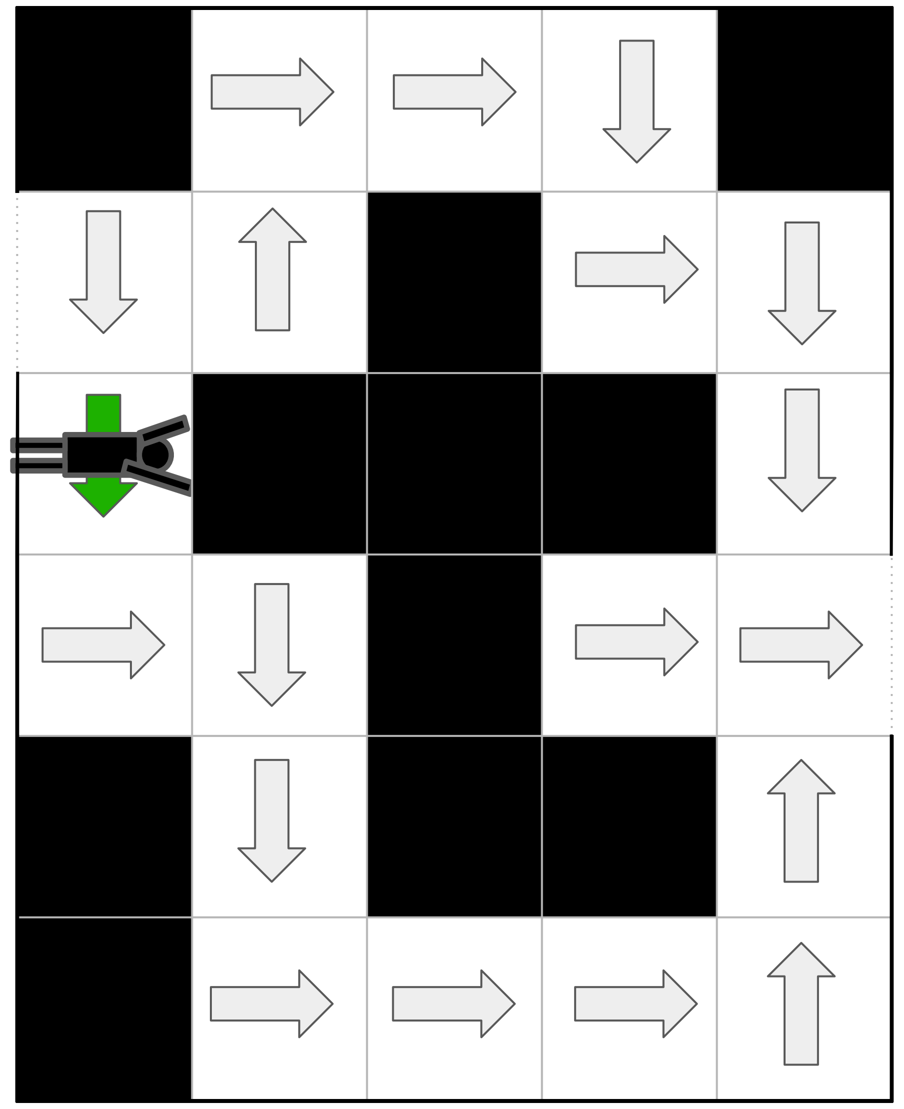
Value

$V^\pi(s)$ is the expected sum of discounted **future rewards** for employing a policy π starting from an initial state s

$$0, \quad -1, \; r^3 \ldots$$

$$r^2 = -1$$

Initial state s^0



goal, +100

failing, -1

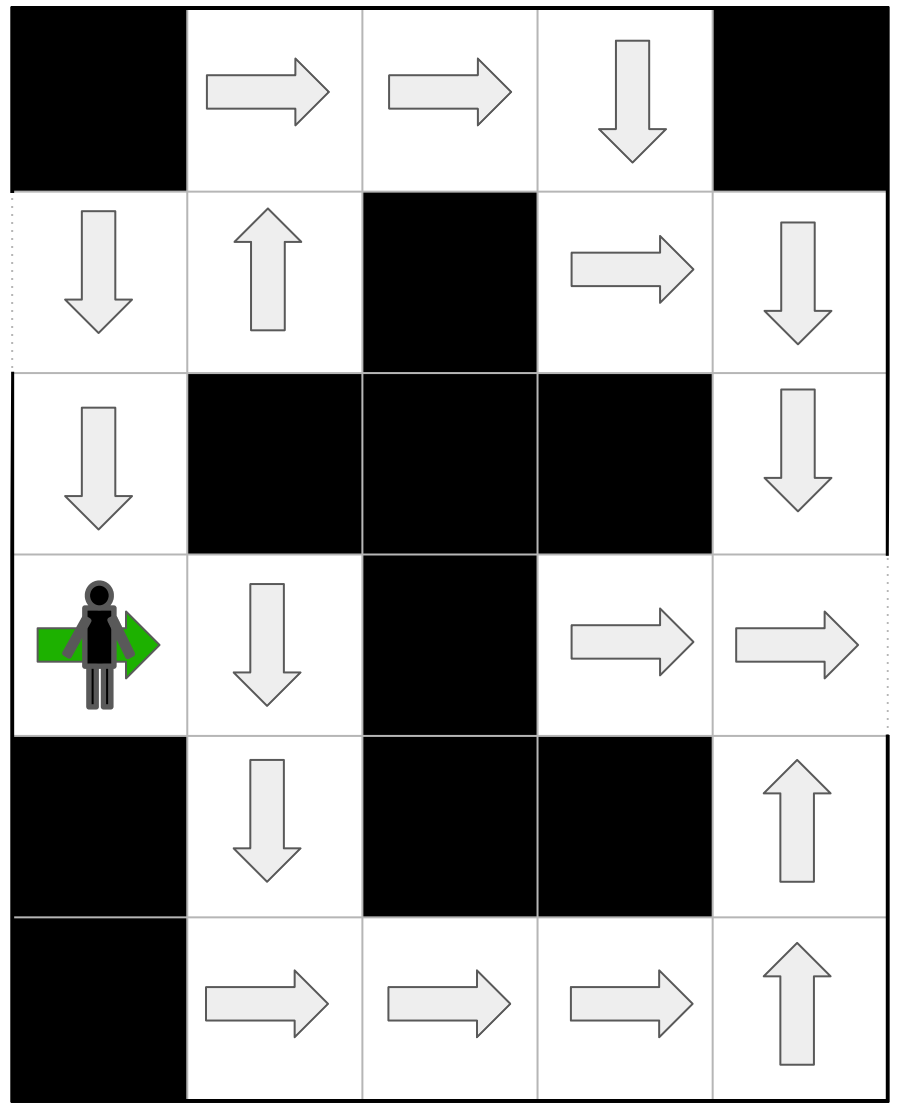
Value

$V^{\pi}(s)$ is the expected sum of discounted **future rewards** for employing a policy π starting from an initial state s

$$0, \quad -1, \; 0 \ldots$$

$r^3 = 0$

Initial state s^0



goal, +100

failing, -1

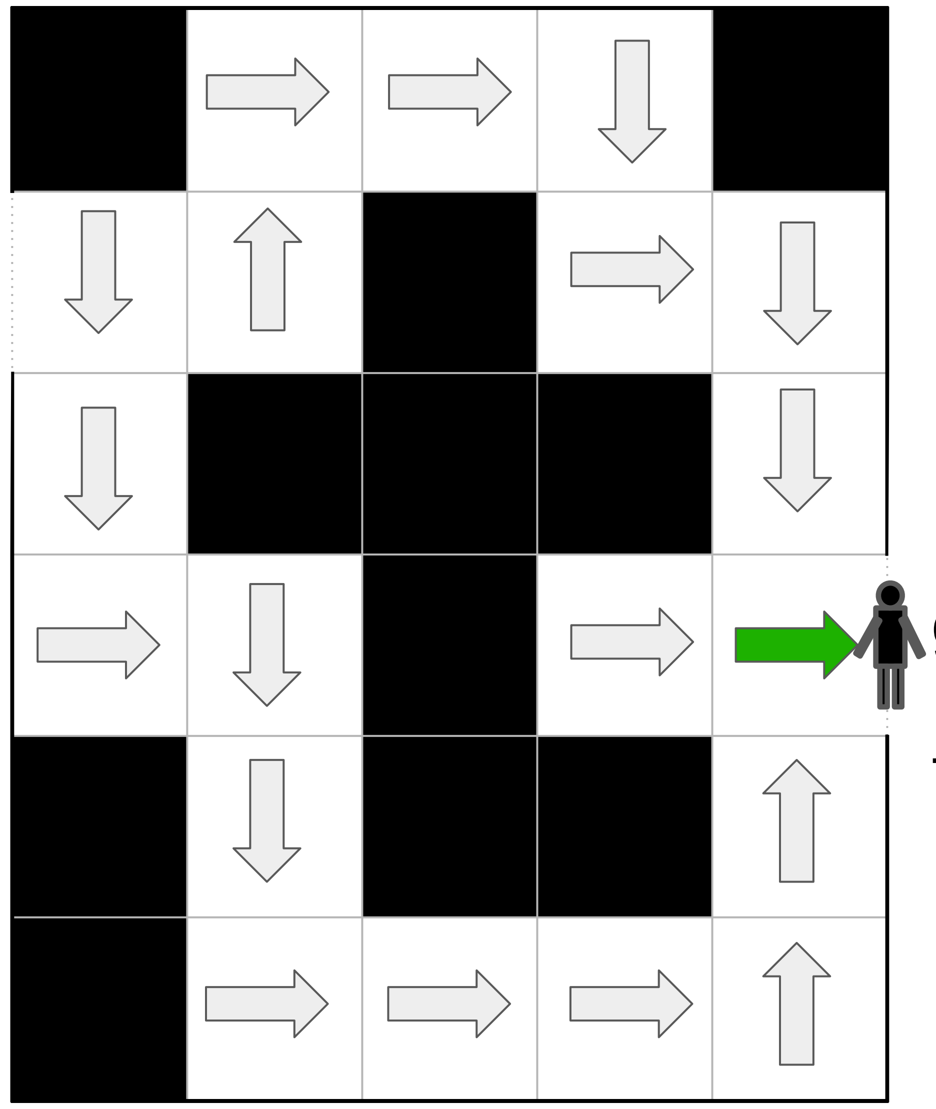
Value

$V^{\pi}(s)$ is the expected sum of discounted **future rewards** for employing a policy π starting from an initial state s

$$0, \quad -1, \;\; 0 \ldots 100$$

$r^T = 100$

Initial state s^0



goal, +100

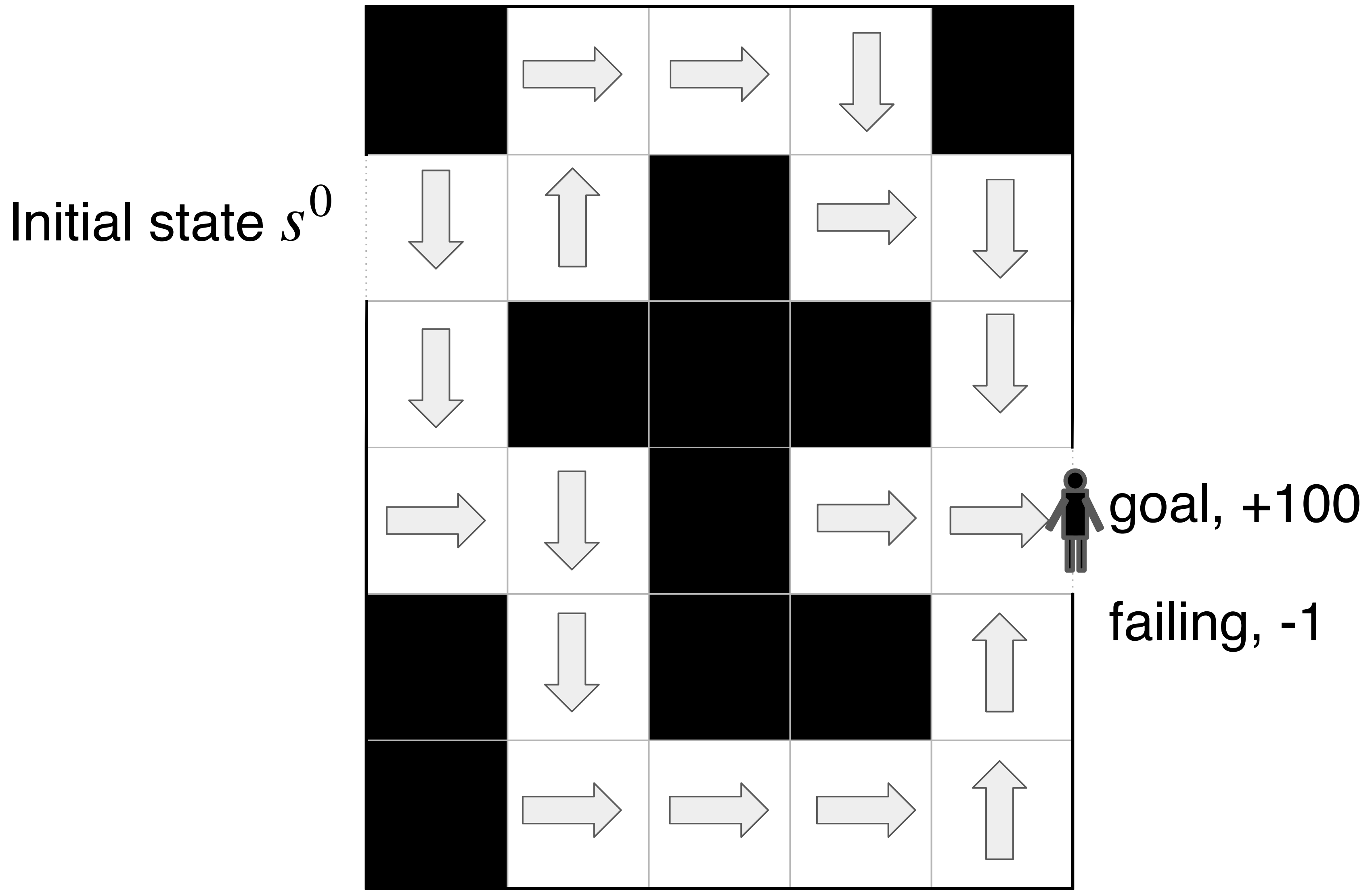
failing, -1

Value

$V^{\pi}(s)$ is the expected **sum of discounted** future rewards for employing a policy π starting from an initial state s

$$\gamma^1 0 + \gamma^2(-1) + \gamma^3 0 + \ldots \gamma^T(100)$$

$0 < \gamma < 1$ discount factor

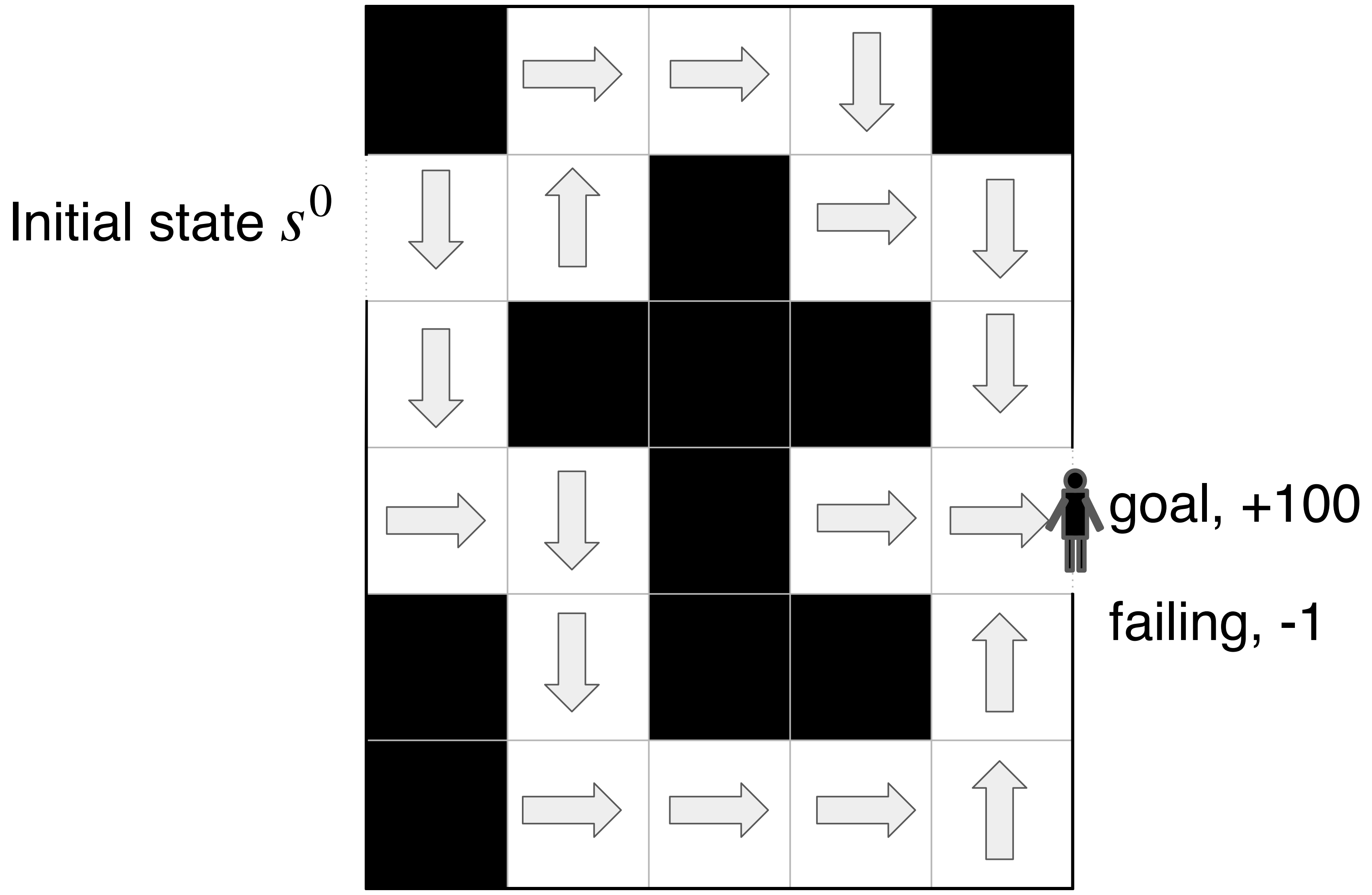


Value

$V^\pi(s)$ is the expected **sum of discounted** future rewards for employing a policy π starting from an initial state s

$$\gamma^1 0 + \gamma^2(-1) + \gamma^3 0 + \ldots \gamma^T(100) = \sum_t \gamma^t r^t$$

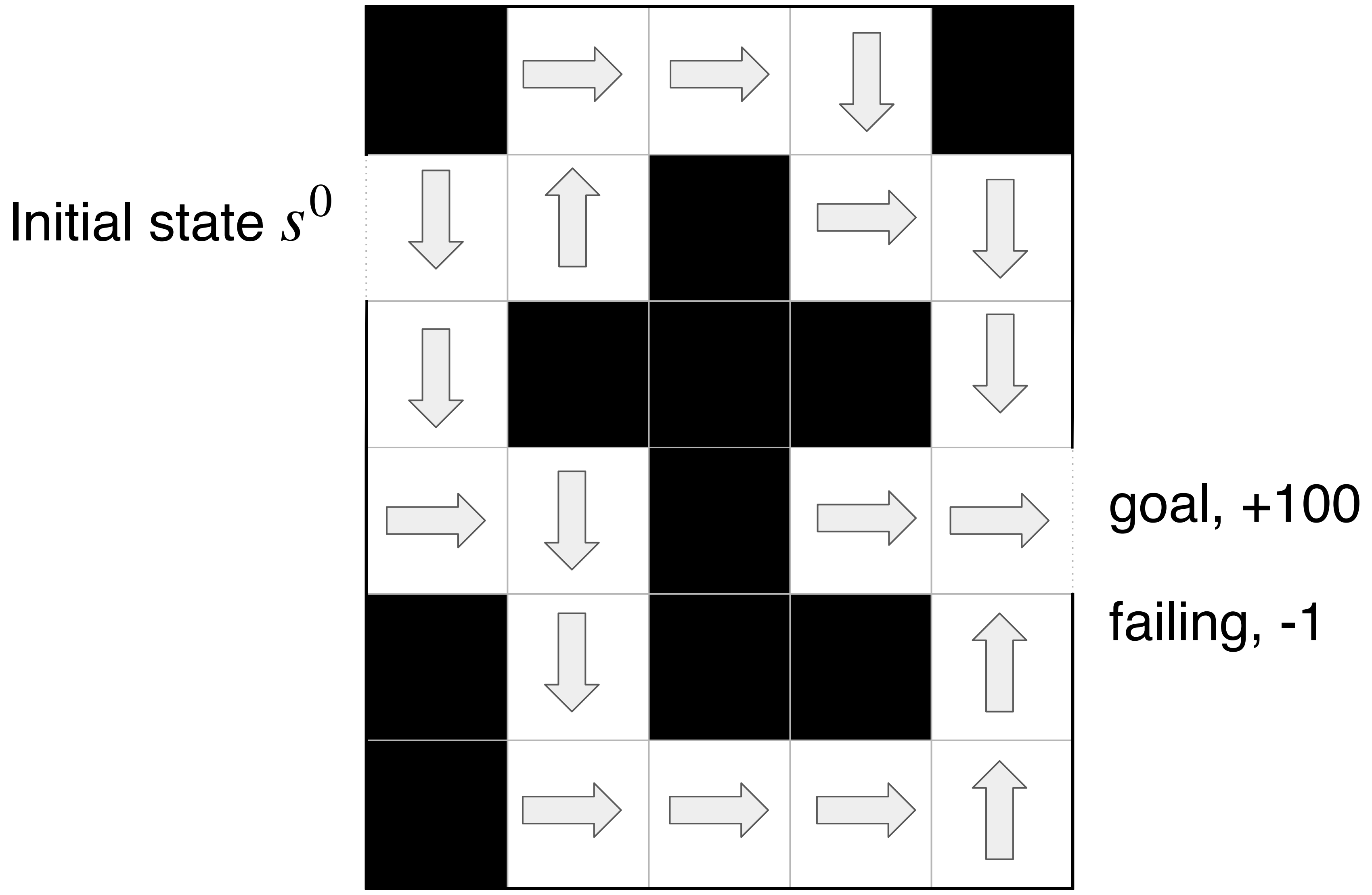
$0 < \gamma < 1$ discount factor



Value

$V^{\pi}(s)$ is the **expected** sum of discounted future rewards for employing a policy π starting from an initial state s

$$V^{\pi}(s) = \underbrace{\mathbb{E}}_{\text{Average over all possible futures}} \left[\sum_t \gamma^t r^t \,|\, s^0 = s, \pi \right]$$

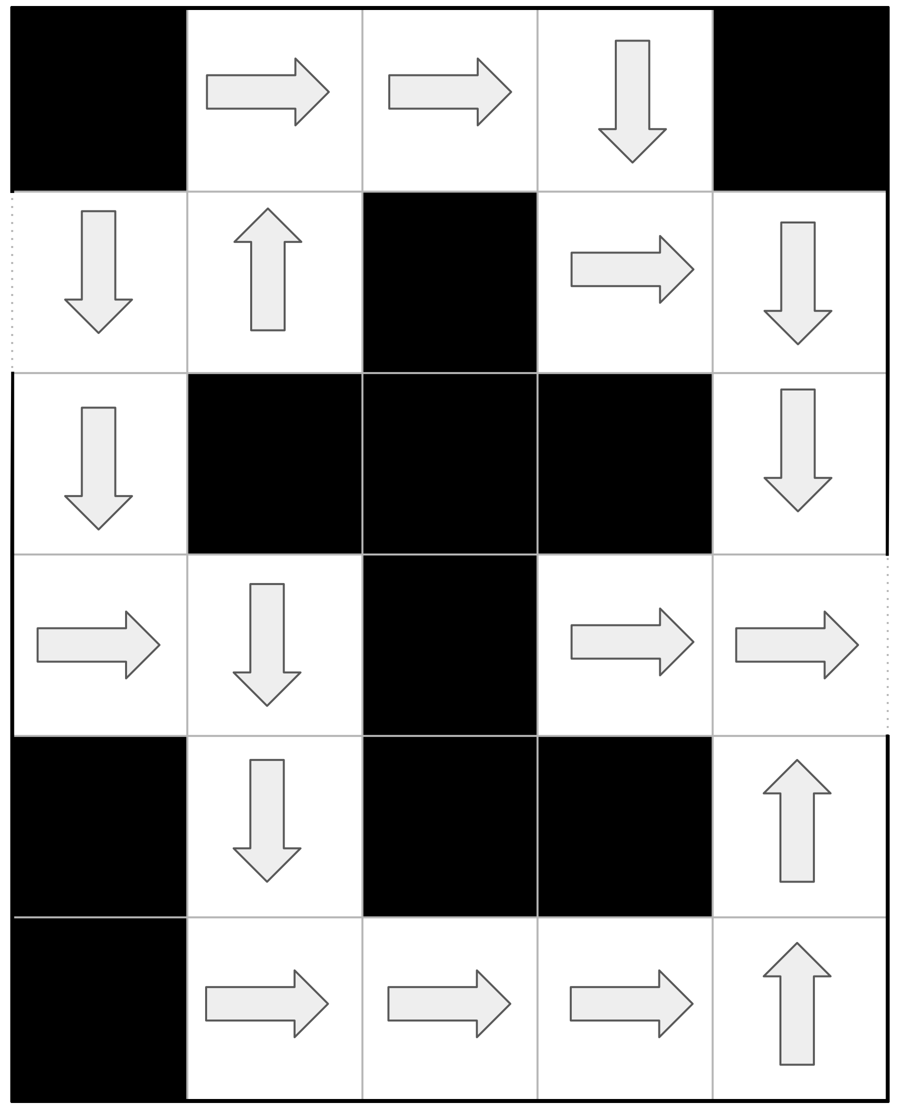


Value

$V^{\pi}(s)$ is the expected sum of discounted future rewards for employing a policy π starting from an initial state s

$$V^{\pi}(s) = \mathbb{E}\left[\sum_t \gamma^t r^t \,|\, s^0 = s, \pi\right]$$

Initial state s^0



goal, +100

~~failing, -1~~

How good is this policy?

Deterministic transitions - no chance of failing, $\gamma = 0.9$

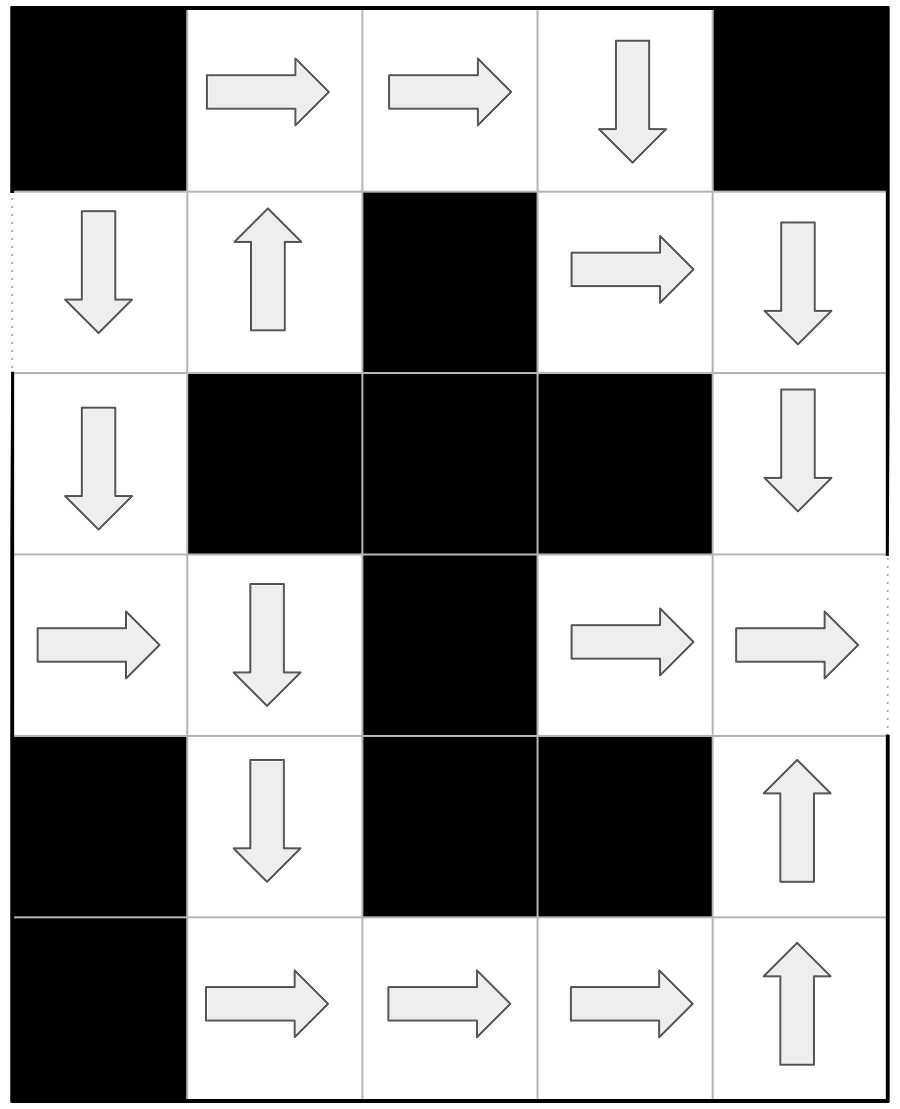
$V^{\pi}(s^0) = ?$

Value

$V^{\pi}(s)$ is the expected sum of discounted future rewards for employing a policy π starting from an initial state s

$$V^{\pi}(s) = \mathbb{E}\left[\sum_t \gamma^t r^t \,|\, s^0 = s, \pi\right]$$

Initial state s^0



goal, +100

~~failing, -1~~

How good is this policy?

Deterministic transitions - no chance of failing, $\gamma = 0.9$

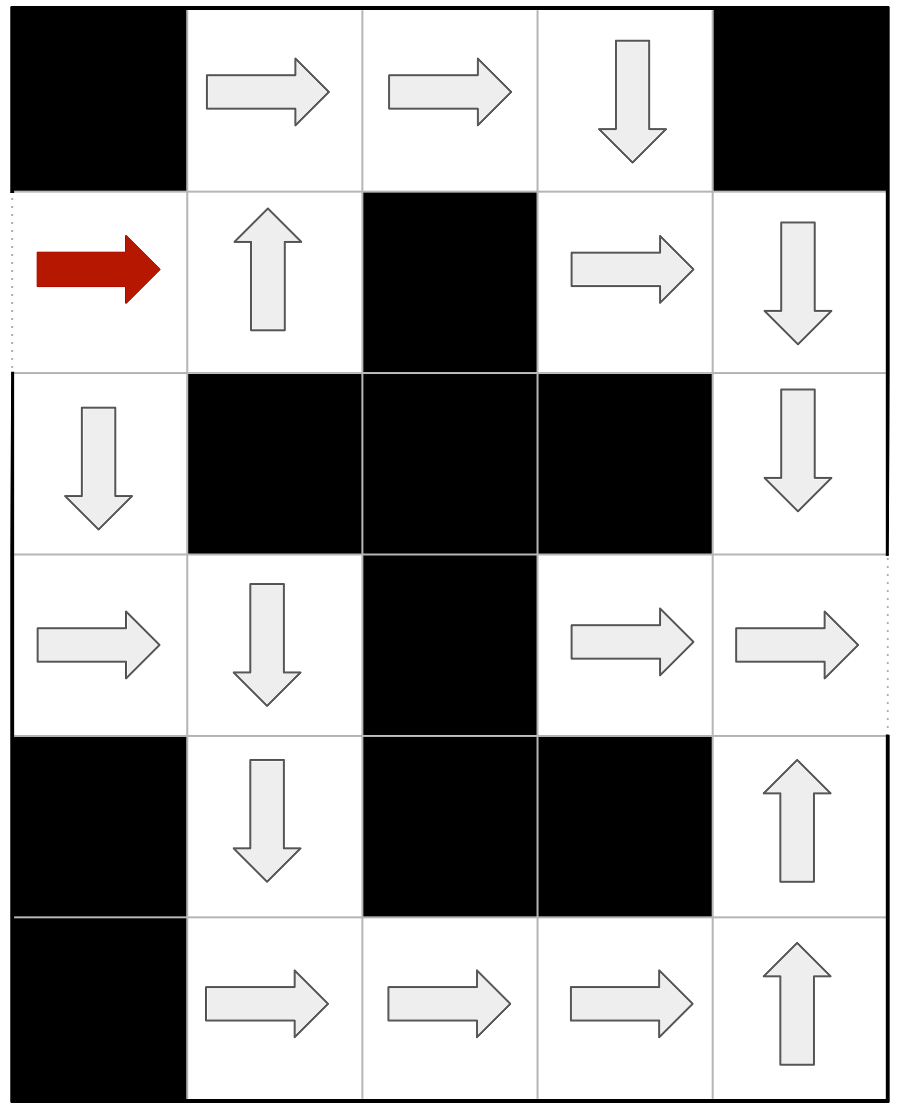
$V^{\pi}(s^0) = 0 + 0 + \ldots + 0 + \gamma^{11} 100 \approx 31$

Value

$V^{\pi}(s)$ is the expected sum of discounted future rewards for employing a policy π starting from an initial state s

$$V^{\pi}(s) = \mathbb{E}\left[\sum_t \gamma^t r^t \,|\, s^0 = s, \pi\right]$$

Initial state s^0



goal, +100

~~failing, -1~~

How good is this policy?

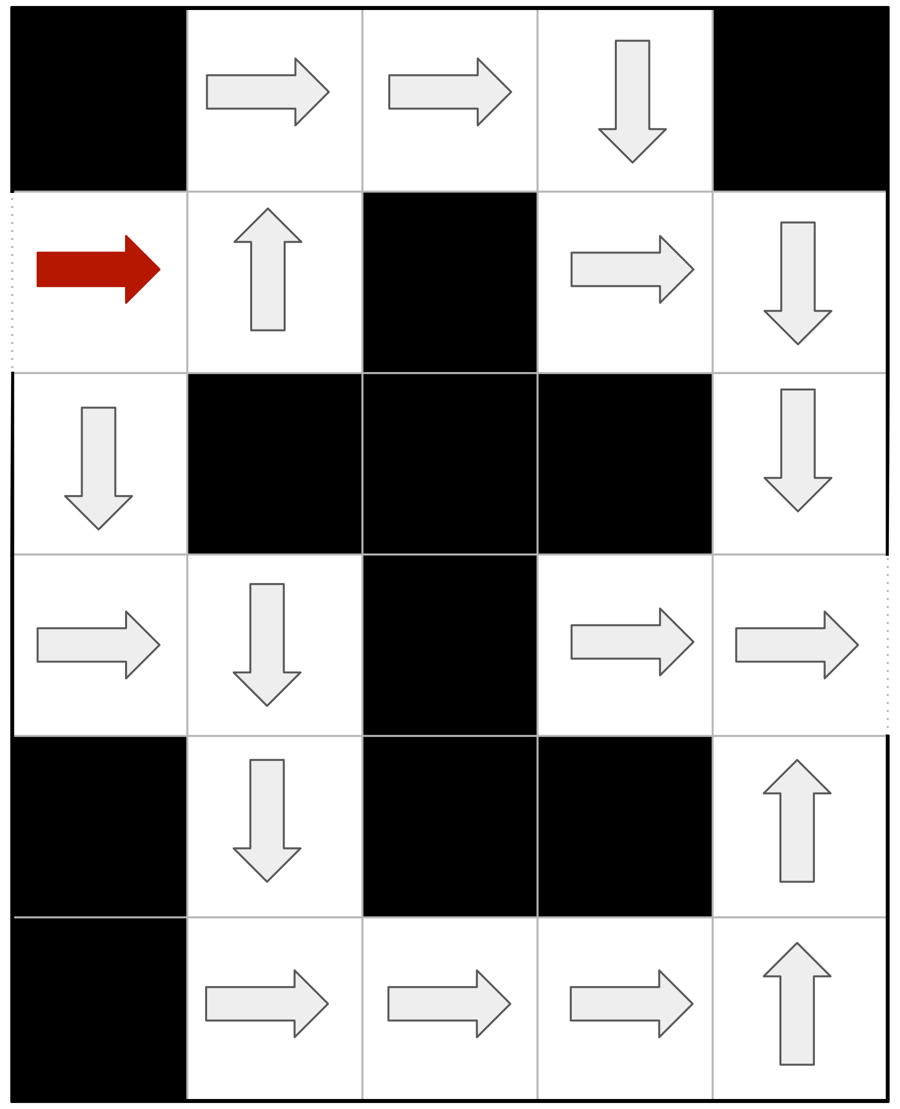
Deterministic transitions - no chance of failing, $\gamma = 0.9$

$V^{\pi}(s^0) = \gamma^{11} 100 \approx 31 \qquad V^{\pi}(s^0) = ?$

Value

$V^{\pi}(s)$ is the expected sum of discounted future rewards for employing a policy π starting from an initial state s

$$V^{\pi}(s) = \mathbb{E}\left[\sum_t \gamma^t r^t \,|\, s^0 = s, \pi\right]$$



How good is this policy?

Deterministic transitions - no chance of failing, $\gamma = 0.9$

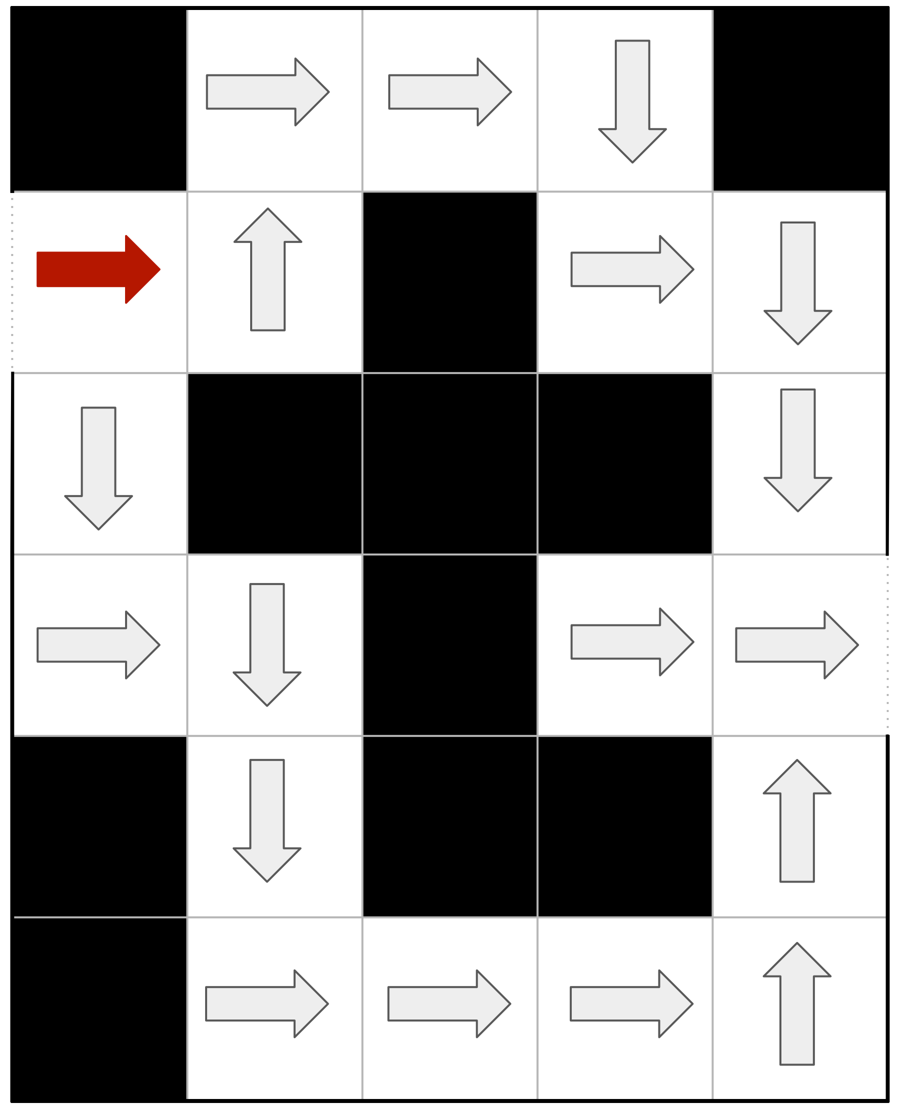
$V^{\pi}(s^0) = \gamma^{11}100 \approx 31$ $\quad V^{\pi}(s^0) = \gamma^{9}100 \approx 39$

Value

$V^{\pi}(s)$ is the expected sum of discounted future rewards for employing a policy π starting from an initial state s

$$V^{\pi}(s) = \mathbb{E}\left[\sum_t \gamma^t r^t \,|\, s^0 = s, \pi\right]$$

Initial state s^0



goal, +100

~~failing, -1~~

How good is this policy?

Deterministic transitions - no chance of failing, $\gamma = 0.9$

$V^{\pi}(s^0) = \gamma^{11}100 \approx 31 < V^{\pi}(s^0) = \gamma^{9}100 \approx 39$

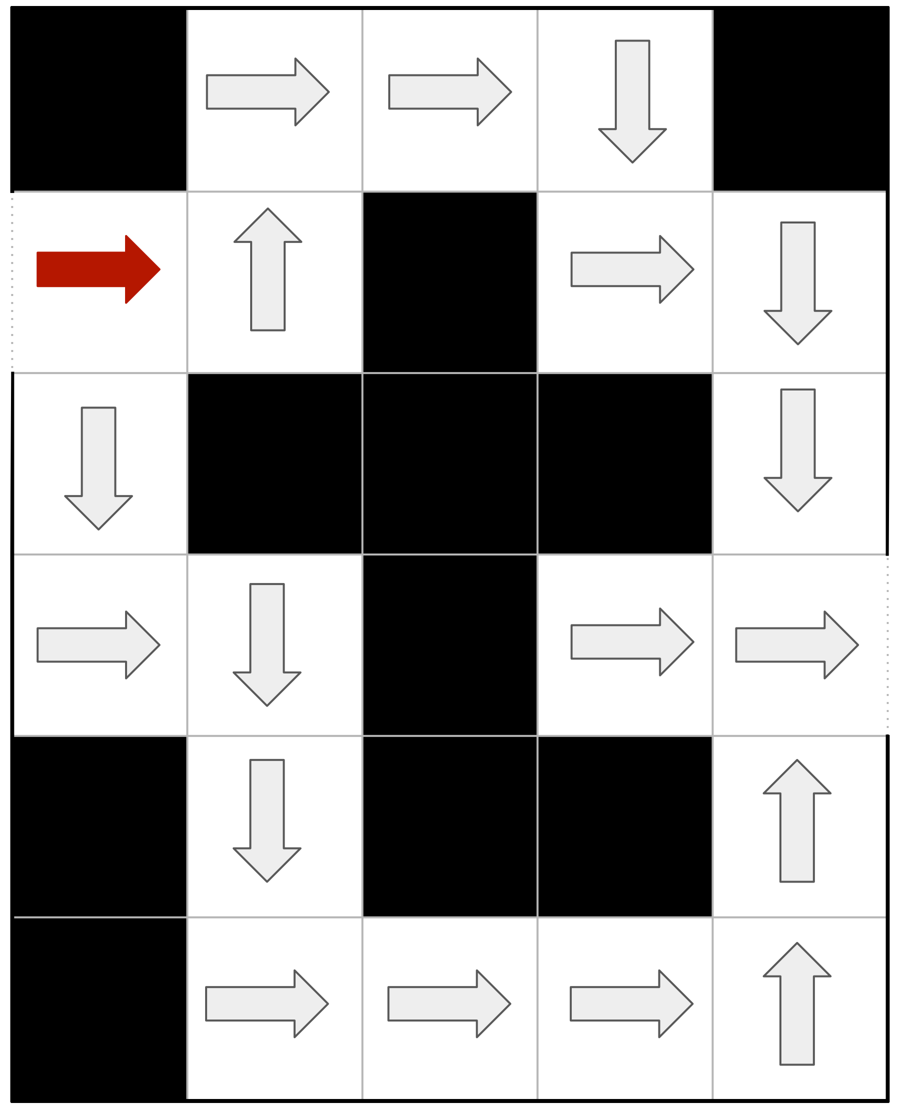
Value measures the quality of a policy: best policy gives the highest value

Value

$V^{\pi}(s)$ is the expected sum of discounted future rewards for employing a policy π starting from an initial state s

$$V^{\pi}(s) = \mathbb{E}\left[\sum_t \gamma^t r^t \,|\, s^0 = s, \pi\right]$$

Initial state s^0



goal, +100

~~failing, -1~~

Goal: find the optimal policy π^* that achieves the highest value

$$V^{\pi^*}(s) = \max_{\pi} V^{\pi}(s) = V^*(s)$$

Value measures the quality of a policy: best policy gives the highest value

Quiz Show

1. Go to brightspace → 9. Reinforcement Learning → Quiz Show
2. Read the description and play the quiz show
3. Do not compute anything just follow your intuition
4. 3 mins then discussion

Quiz Show: discussion

How do you formulate this game as a Markov decision process?

1. Who is the agent?

2. What are the states and initial state?

3. What are the actions?

4. What is the reward?

Quiz Show: discussion

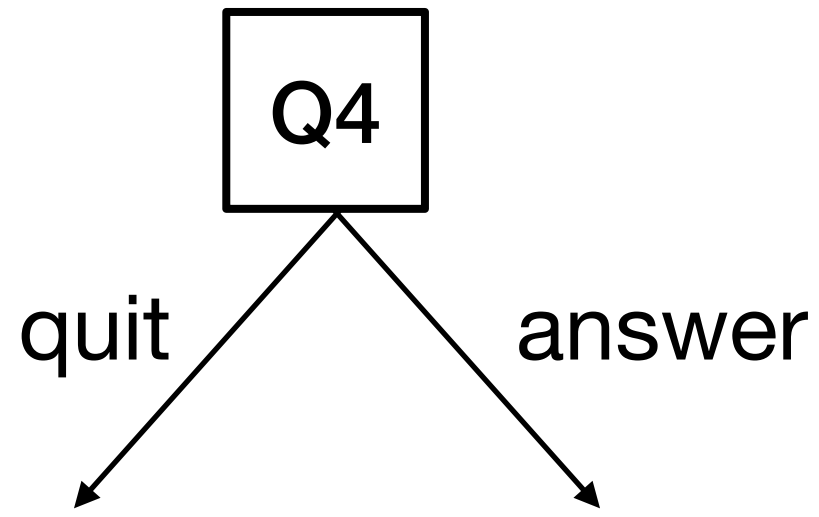
How do you formulate this game as a Markov decision process?

1. Who is the agent?
1. You, the player
2. What are the states and initial state?
2. $S = \{Q_1, Q_2, Q_3, Q_4, home\}$ and $s^0 = Q_1$
3. What are the actions?
3. $A = \{quit, answer\}$
4. What is the reward?
4. Reward $R(home, \ldots) = 0\$, R(Q_2, quit) = 100\$,\ R(Q_3, quit) = 1100\$\ \ldots,$

Optimal policy $\begin{cases} \pi^*(Q_4) & = quit \\ \pi^*(Q) & = answer \quad \text{for } Q \neq Q_4 \end{cases}$ with optimal value $V^*(Q_1) = 3746$

Quiz Show: intuition

Assume we are at question 4, what is the optimal action?



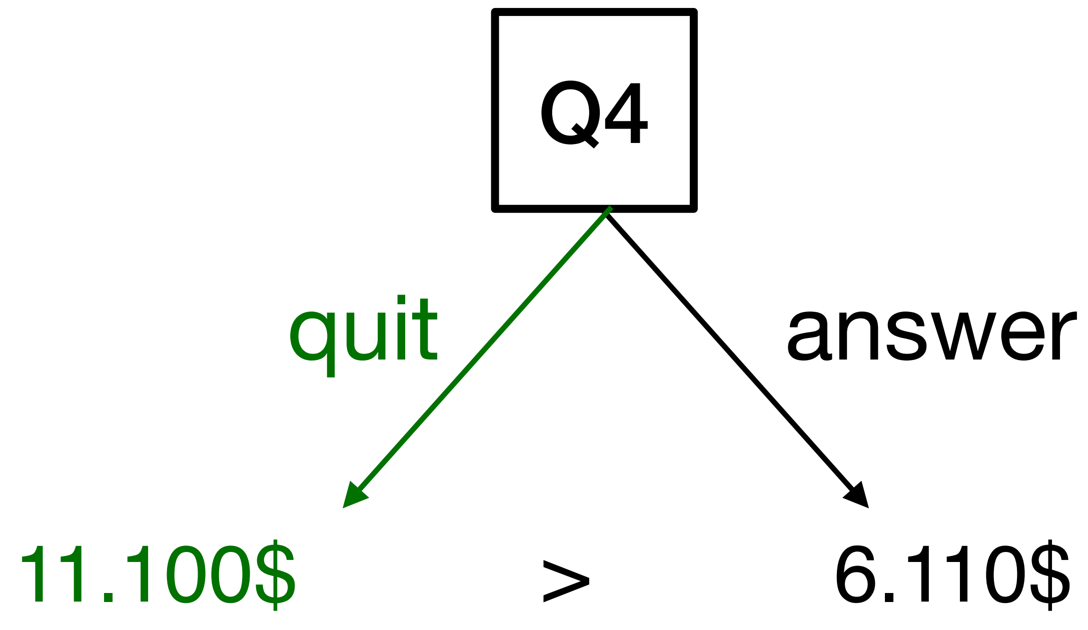
Quiz Show: intuition

Assume we are at question 4, what is the optimal action?

$$
\begin{array}{ccc}
 & \boxed{Q4} & \\
\swarrow_{\text{quit}} & & \searrow^{\text{answer}} \\
11.100\$ & & 0.1 \times 61.100 + 0.9 \times 0 = 6.110\$
\end{array}
$$

Quiz Show: intuition

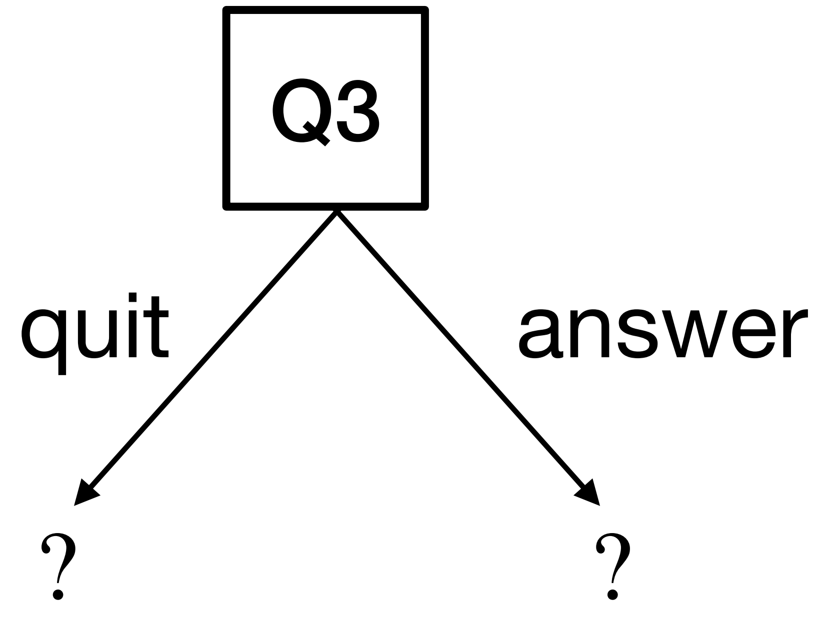
Assume we are at question 4, what is the optimal action?



$$\pi^*(Q_4) = quit \qquad V^*(Q_4) = 11.100\$$$

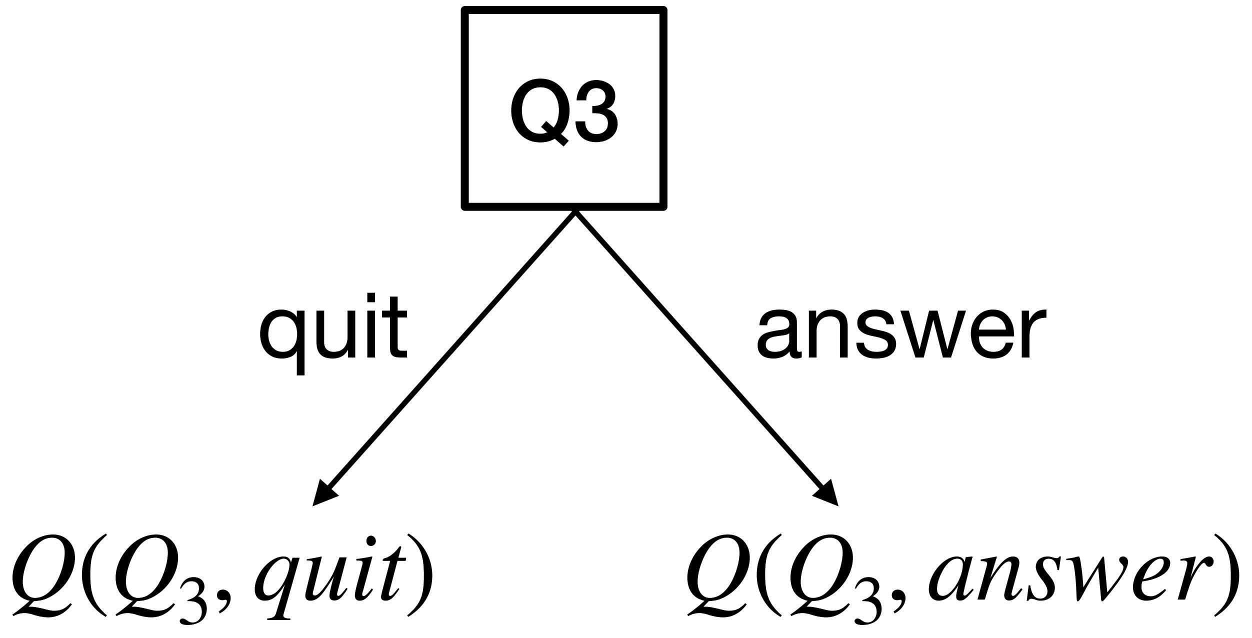
Quiz Show: intuition

Assume we are at question 3, what is the optimal action?



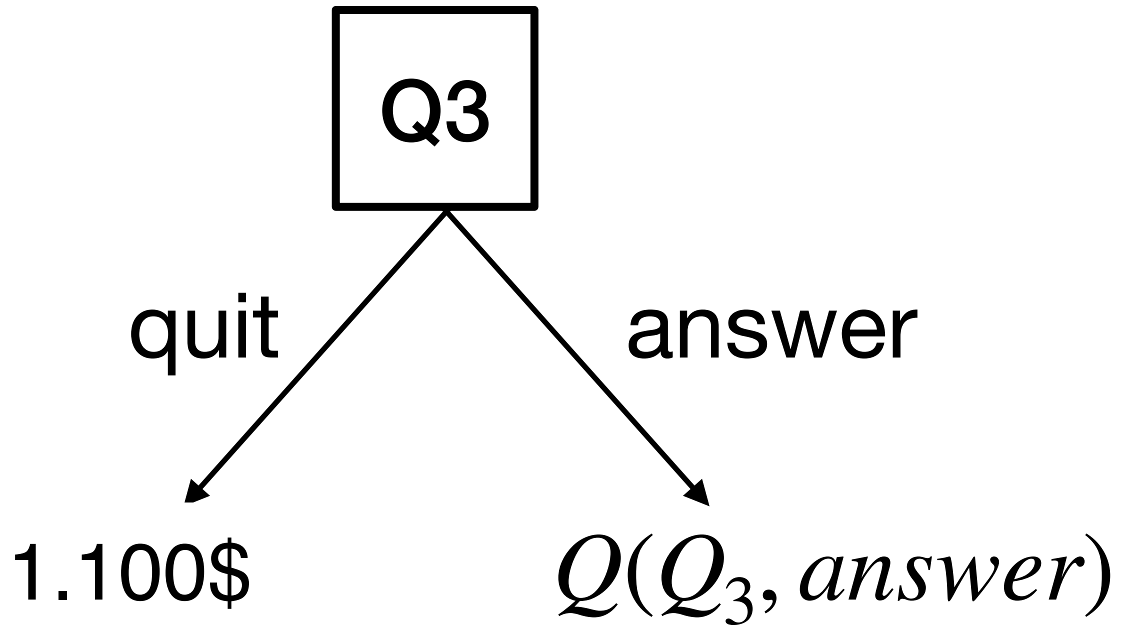
Q-Value

- Expected value for executing an action in a given state assuming that you will act optimally in the future



Q-Value

- Expected value for executing an action in a given state assuming that you will act optimally in the future



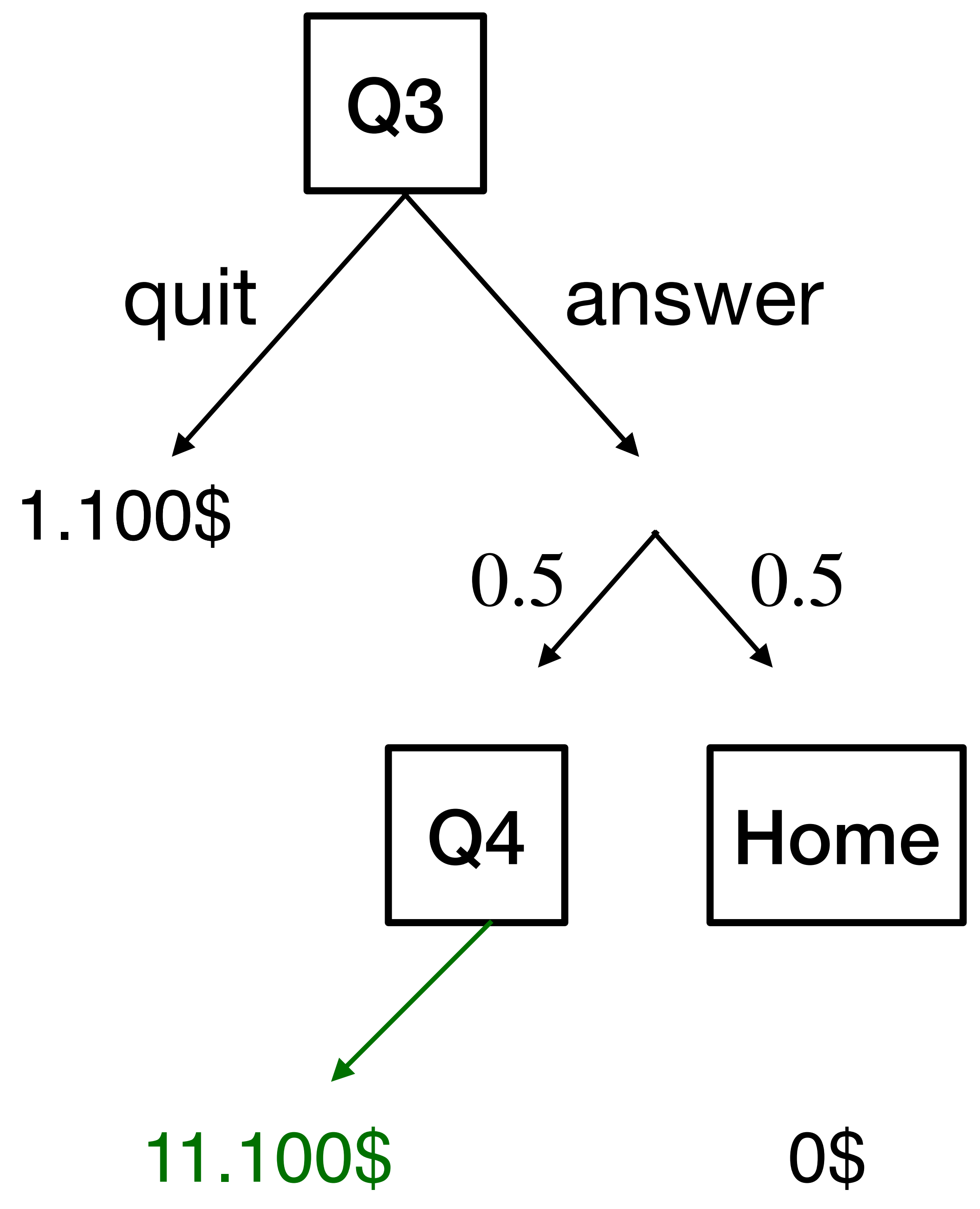
$$Q(Q_3, quit) = R(Q_3, quit) + 0 = 1100$$

Immediate reward for quitting at question 3

There's no future state as your game is terminated

Q-Value

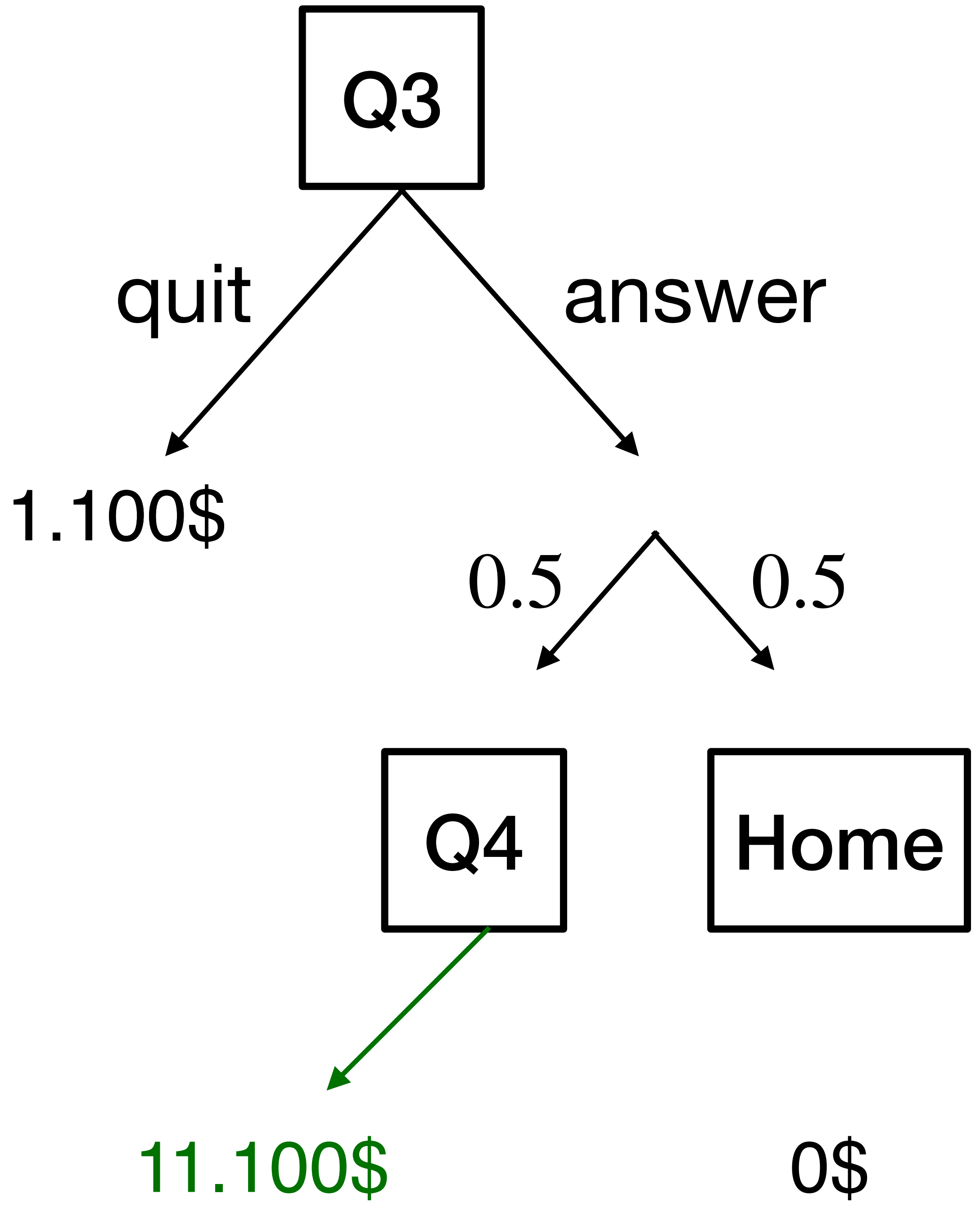
- Expected value for executing an action in a given state assuming that you will act optimally in the future



$Q(Q_3, answer) =$

Q-Value

- Expected value for executing an action in a given state assuming that you will act optimally in the future



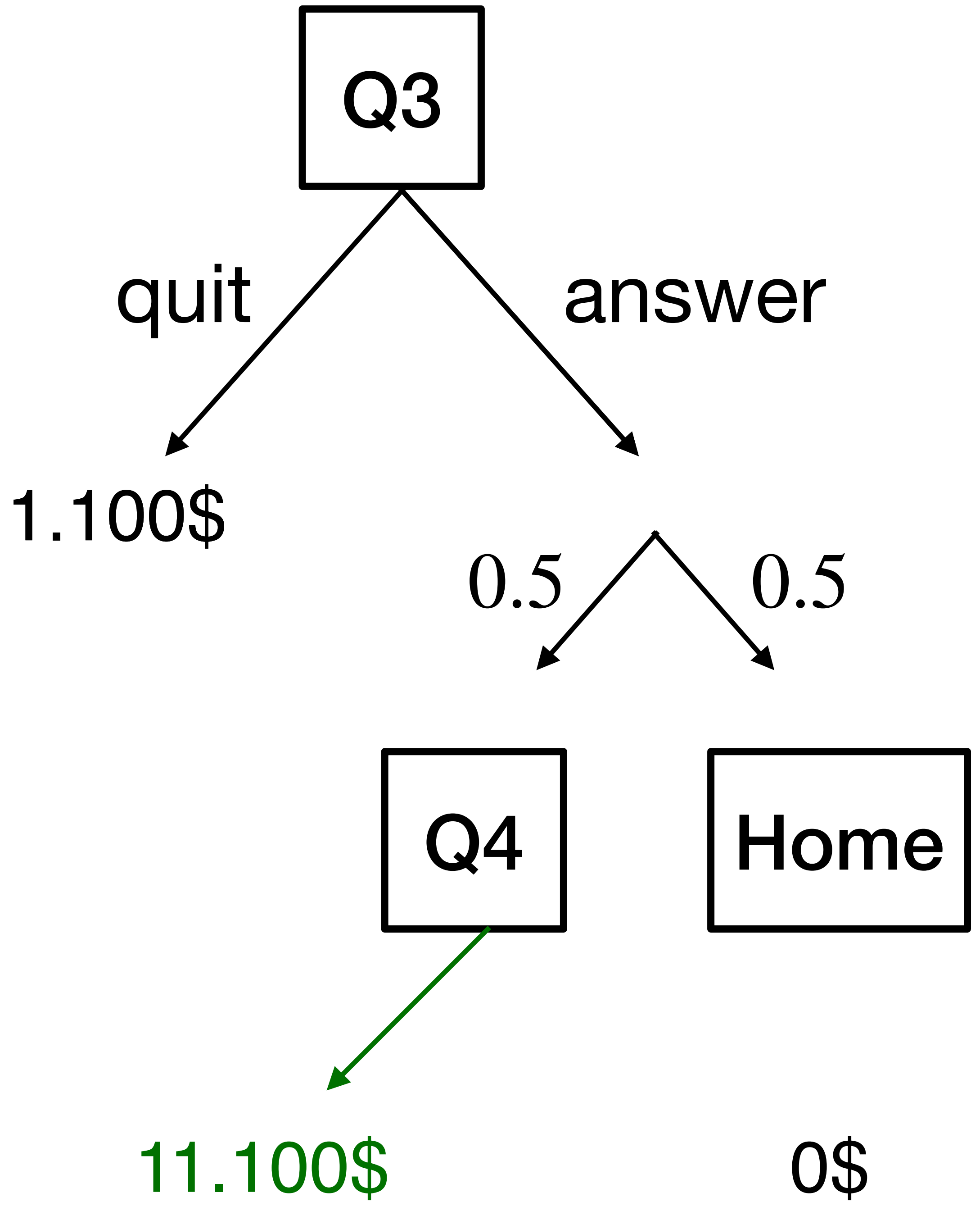
$$Q(Q_3, answer) = \quad 0 \quad + \Big(\quad 0.5 \quad \times 11.100 + \quad 0.5 \quad \times \quad 0 \quad \Big) = 5.550$$

Immediate reward for answering at question 3

Expected value for acting optimally in the future

Q-Value

- Expected value for executing an action in a given state assuming that you will act optimally in the future



$$Q(Q_3, answer) = \quad 0 \quad + (\quad 0.5 \quad \times 11.100 + \quad 0.5 \quad \times \quad 0 \quad) = 5.550$$

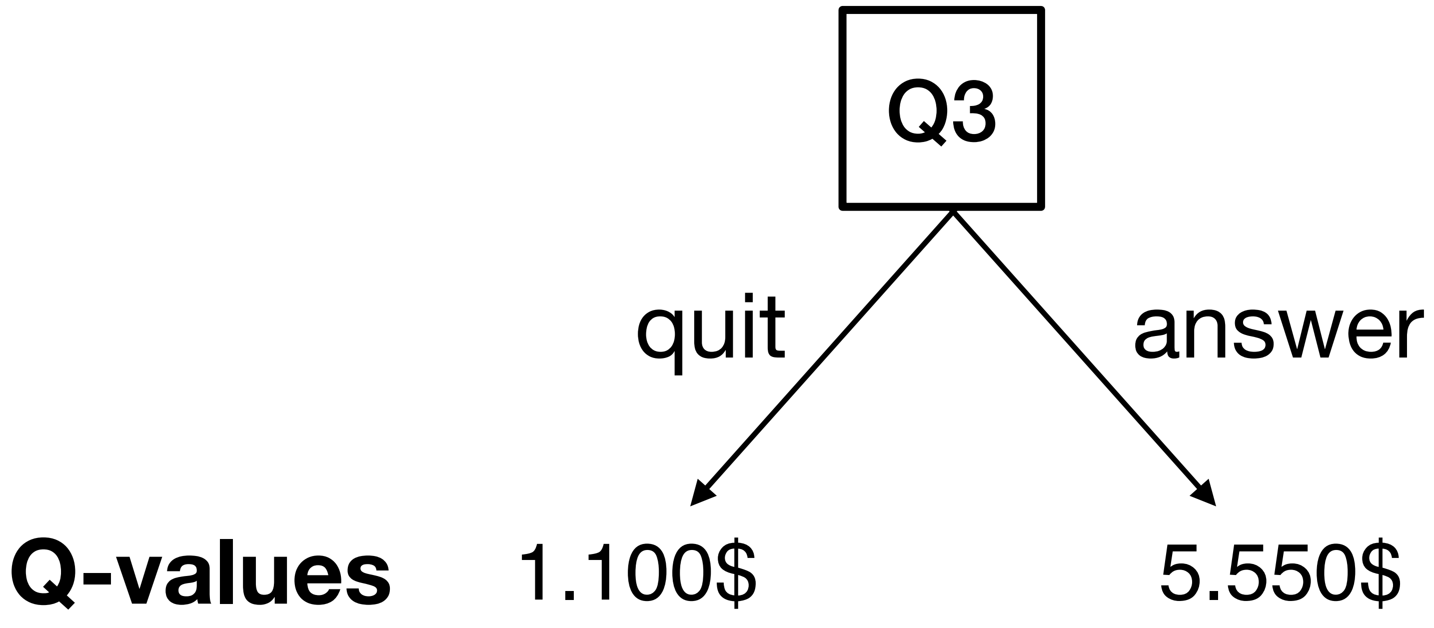
$$R(Q_3, answer) + (P(Q_4 \,|\, answer, Q_3) \times V^*(Q_4) + P(home \,|\, answer, Q_3) \times V^*(home))$$

Immediate reward for answering at question 3

Expected value for acting optimally in the future

Q-Value

- Expected value for executing an action in a given state assuming that you will act optimally in the future



$$Q(Q_3, answer) = 0 + (0.5 \times 11.100 + 0.5 \times 0) = 5.550$$

$$R(Q_3, answer) + (P(Q_4 \,|\, answer, Q_3) \times V^*(Q_4) + P(home \,|\, answer, Q_3) \times V^*(home))$$

Immediate reward for answering at question 3

Expected value for acting optimally in the future

Q-Value

- Expected value for executing an action in a given state assuming that you will act optimally in the future

$$Q(s,a) = \underbrace{R(s,a)}_{\text{Immediate reward}} + \underbrace{\gamma \sum_{s'} P(s' \,|\, s,a) V^*(s')}_{\text{Discounted expected value for acting optimally in the future}}$$

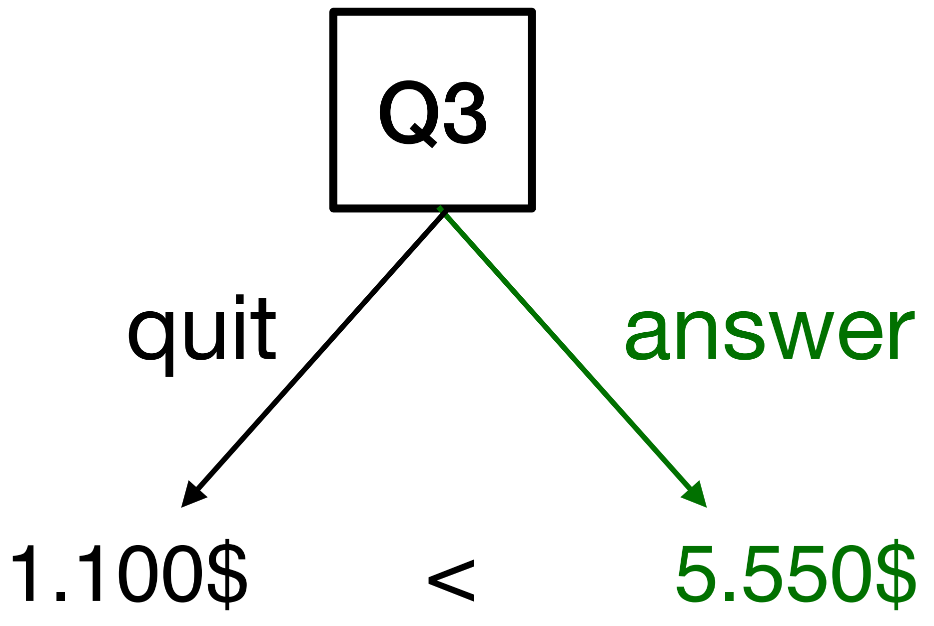
$$Q(Q_3, answer) = R(Q_3, answer) + P(Q_4 \,|\, answer, Q_3) V^*(Q_4) + P(home \,|\, answer, Q_3) V^*(home)$$

Q-Value

- Expected value for executing an action in a given state assuming that you will act optimally in the future

$$Q(s,a) = R(s,a) + \gamma \sum_{s'} P(s' \mid s,a) V^*(s')$$

- Select the optimal action as the one that maximizes the Q-value



$Q(Q_3, quit)$ $\qquad$ $Q(Q_3, answer)$

$\pi^*(Q_3) = \text{argmax}_a Q(Q_3, a) = answer$ $\qquad$ $V^*(Q_3) = \max_a Q(s,a) = 5500$

Value Iteration

- Compute approximation of $V^*(s)$ for every state s

- Compute Q-values for every state s and action a

$$Q(s,a) = R(s,a) + \gamma \sum_{s'} P(s' \mid s,a) V^*(s')$$

- Build the optimal policy as $\pi^*(s) = argmax_a\, Q(s,a)$ that maximizes the Q-value

Value Iteration

Algorithm

Initialize random values V^0

$$\Delta \longleftarrow 0$$

Repeat:

For $s \in S$:

For $a \in A$

$$Q^k(s,a) \longleftarrow R(s,a) + \gamma \sum_{s'} P(s' \mid s,a) V^{k-1}(s')$$

$$V^k(s) = \max_a Q^k(s,a)$$

$$\Delta \longleftarrow \max(\Delta, |V^k(s) - V^{k-1}(s)|)$$

Until $\Delta < \varepsilon$

Return $V^k \approx V^* \in \mathbb{R}^{|S|}$

Iterative algorithm that converges to the optimal value V^*

Value Iteration

Algorithm

Initialize random values $\boxed{V^0}$

$$\Delta \longleftarrow 0$$

Repeat:

For $s \in S$:

For $a \in A$

$$Q^k(s,a) \longleftarrow R(s,a) + \gamma \sum_{s'} P(s' \mid s,a) V^{k-1}(s')$$

$$V^k(s) = \max_a Q^k(s,a)$$

$$\Delta \longleftarrow \max(\Delta, |V^k(s) - V^{k-1}(s)|)$$

Until $\Delta < \varepsilon$

Return $V^k \approx V^* \in \mathbb{R}^{|S|}$

Vector of values $V \in \mathbb{R}^{|S|}$, one value for each state

Value Iteration

Algorithm

Initialize random values V^0

$$\Delta \longleftarrow 0$$

Repeat:

For $s \in S$:

For $a \in A$

$$\boxed{Q^k(s,a) \longleftarrow R(s,a) + \gamma \sum_{s'} P(s' \mid s,a) V^{k-1}(s')}$$

$$V^k(s) = \max_a Q^k(s,a)$$

$$\Delta \longleftarrow \max(\Delta, |V^k(s) - V^{k-1}(s)|)$$

Until $\Delta < \varepsilon$

Return $V^k \approx V^* \in \mathbb{R}^{|S|}$

Updates the Q-values for each action a and state s

Value Iteration

Algorithm

Initialize random values V^0

$$\Delta \longleftarrow 0$$

Repeat:

For $s \in S$:

For $a \in A$

$$Q^k(s,a) \longleftarrow R(s,a) + \gamma \sum_{s'} P(s' \mid s,a) V^{k-1}(s')$$

$$\boxed{V^k(s) = \max_a Q^k(s,a)}$$

$$\Delta \longleftarrow \max(\Delta, |V^k(s) - V^{k-1}(s)|)$$

Until $\Delta < \varepsilon$

Return $V^k \approx V^* \in \mathbb{R}^{|S|}$

Retain the maximum Q-value

Value Iteration

Algorithm

Initialize random values V^0

$$\Delta \longleftarrow 0$$

Repeat:

For $s \in S$:

For $a \in A$

$$Q^k(s,a) \longleftarrow R(s,a) + \gamma \sum_{s'} P(s' \mid s,a) V^{k-1}(s')$$

$$V^k(s) = \max_a Q^k(s,a)$$

$$\Delta \longleftarrow \max(\Delta, |V^k(s) - V^{k-1}(s)|)$$

Until $\Delta < \varepsilon$

Return $V^k \approx V^* \in \mathbb{R}^{|S|}$

Repeat until the updates are small enough

Value Iteration

Algorithm

Initialize random values V^0

$$\Delta \longleftarrow 0$$

Repeat:

For $s \in S$:

For $a \in A$

$$Q^k(s, a) \longleftarrow R(s, a) + \gamma \sum_{s'} P(s' | s, a) V^{k-1}(s')$$

$$V^k(s) = \max_a Q^k(s, a)$$

$$\Delta \longleftarrow \max(\Delta, | V^k(s) - V^{k-1}(s) |)$$

Until $\Delta < \varepsilon$

Return $V^k \approx V^* \in \mathbb{R}^{|S|}$

Return the optimal value V^*

Value Iteration

Algorithm

Initialize random values V^0

$$\Delta \longleftarrow 0$$

Repeat:

For $s \in S$:

For $a \in A$

$$Q^k(s,a) \longleftarrow R(s,a) + \gamma \sum_{s'} P(s' \mid s,a) V^{k-1}(s')$$

$$V^k(s) = \max_a Q^k(s,a)$$

$$\Delta \longleftarrow \max(\Delta, |V^k(s) - V^{k-1}(s)|)$$

Until $\Delta < \varepsilon$

Return $V^k \approx V^* \in \mathbb{R}^{|S|}$

For $s \in S$:

For $a \in A$

$$Q(s,a) = R(s,a) + \gamma \sum_{s'} P(s' \mid s,a) V^*(s')$$

$$\pi^*(s) = argmax_a \; Q(s,a)$$

From the optimal value V^* to the optimal policy π^*

Value Iteration

Algorithm

Initialize random values V^0

$$\Delta \longleftarrow 0$$

Repeat:

For $s \in S$:

For $a \in A$

$$Q^k(s,a) \longleftarrow \boxed{R(s,a)} + \gamma \sum_{s'} \boxed{P(s'|s,a)} V^{k-1}(s')$$

$$V^k(s) = \max_a Q^k(s,a)$$

$$\Delta \longleftarrow \max(\Delta, |V^k(s) - V^{k-1}(s)|)$$

Until $\Delta < \varepsilon$

Return $V^k \approx V^* \in \mathbb{R}^{|S|}$

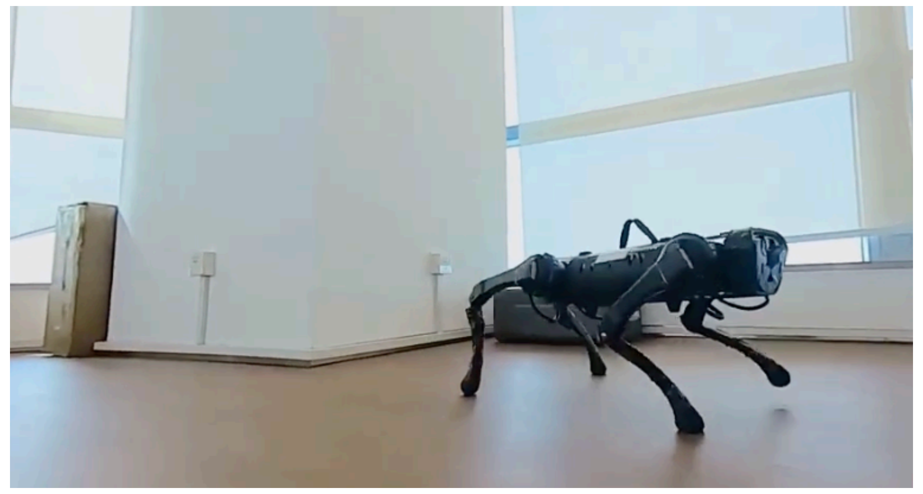
For $s \in S$:

For $a \in A$

$$Q(s,a) = \boxed{R(s,a)} + \gamma \sum_{s'} \boxed{P(s'|s,a)} V^*(s')$$

$$\pi^*(s) = argmax_a \; Q(s,a)$$

Assumption that transitions and rewards are known



Planning vs Reinforcement Learning

Planning

→ The transitions and reward model are known

→ Quiz show, robot navigation

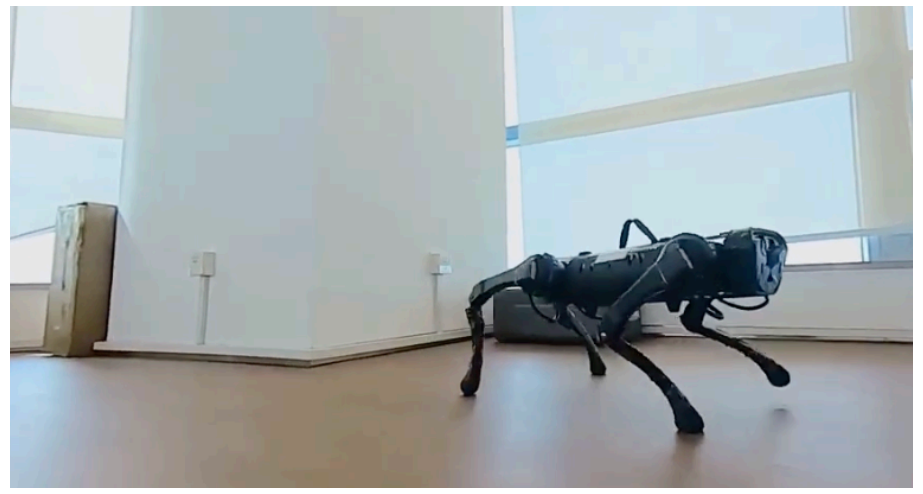
→ Use value iteration to find optimal policy to deploy afterwards

Reinforcement learning

→ No prior knowledge of transition or reward model

→ Robot parkour, Atari Breakout

→ Find an optimal policy **while** interacting with an unfamiliar environment



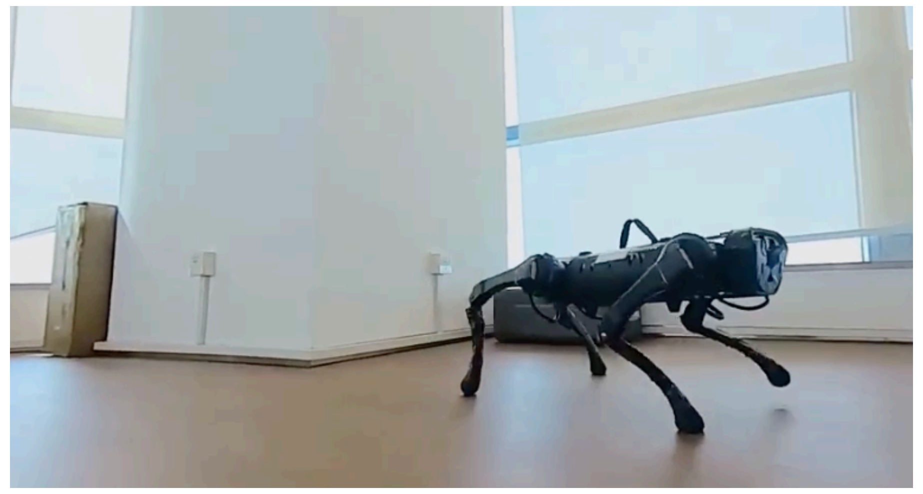
Supervised Learning vs Reinforcement Learning

Supervised Learning

- → Examples of correct or incorrect behaviour
- → Independent sample of experience
- → Learn a model and use it afterwards

Reinforcement learning

- → Only reward for a sequence of actions tried
- → Agent has partial control over the training data
- → Agent learns on-line: it must maximize performance during learning



Tabular Q-Learning

Estimate Q-value while interacting with the environment: MDP is unknown but can be sampled!

Table of q-values Q with shape $|S| \times |A|$ (one $Q(s, a)$ for every s, a)

Tabular Q-Learning

Estimate Q-value while interacting with the environment: MDP is unknown but can be sampled!

Table of q-values Q with shape $|S| \times |A|$ (one $Q(s, a)$ for every s, a)

After observing a transition s, a, s', r update the table Q

$$Q(s, a) \longleftarrow (1 - \alpha)Q(s, a) + \alpha(r + \gamma \max_{a'} Q(s', a'))$$

Tabular Q-Learning

Estimate Q-value while interacting with the environment: MDP is unknown but can be sampled!

Table of q-values Q with shape $|S| \times |A|$ (one $Q(s,a)$ for every s, a)

After observing a transition s, a, s', r update the table Q

$$Q(s,a) \longleftarrow (1-\alpha)\underbrace{Q(s,a)}_{\text{old value}} + \alpha\underbrace{(r + \gamma \max_{a'} Q(s',a'))}_{\text{update}}$$

Tabular Q-Learning

Estimate Q-value while interacting with the environment: MDP is unknown but can be sampled!

Table of q-values Q with shape $|S| \times |A|$ (one $Q(s, a)$ for every s, a)

After observing a transition s, a, s', r update the table Q

$$Q(s,a) \longleftarrow (1-\alpha)Q(s,a) + \alpha\underbrace{(r + \gamma \max_{a'} Q(s', a'))}_{\text{update}}$$

$$Q(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q(s',a')$$

Tabular Q-Learning

Estimate Q-value while interacting with the environment: MDP is unknown but can be sampled!

Table of q-values Q with shape $|S| \times |A|$ (one $Q(s,a)$ for every s, a)

After observing a transition s, a, s', r update the table Q

$$Q(s,a) \longleftarrow (1-\alpha)Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a')) = Q(s,a) + \alpha\underbrace{(r + \gamma \max_{a'} Q(s',a') - Q(s,a))}_{\text{Temporal difference TD error}}$$

Temporal average between the old value and the update weighted by the learning rate α

Tabular Q-Learning

Algorithm

Initialize constant Q-table $Q(s, a) \longleftarrow c$

Repeat

Initialize the state s^0

Repeat:

Learning agent chooses $a^t = \max_a Q(s^t, a)$

Perform one env step $s^{t+1}, r^{t+1} \longleftarrow \text{env}(s^t, a^t)$

Update Q-table

$$Q(s^t, a^t) \longleftarrow Q(s^t, a^t) + \alpha \left(r^{t+1} + \gamma \max_{a'} Q(s^{t+1}, a') - Q(s^t, a^t) \right)$$

Until the episode is done

Until max number of steps

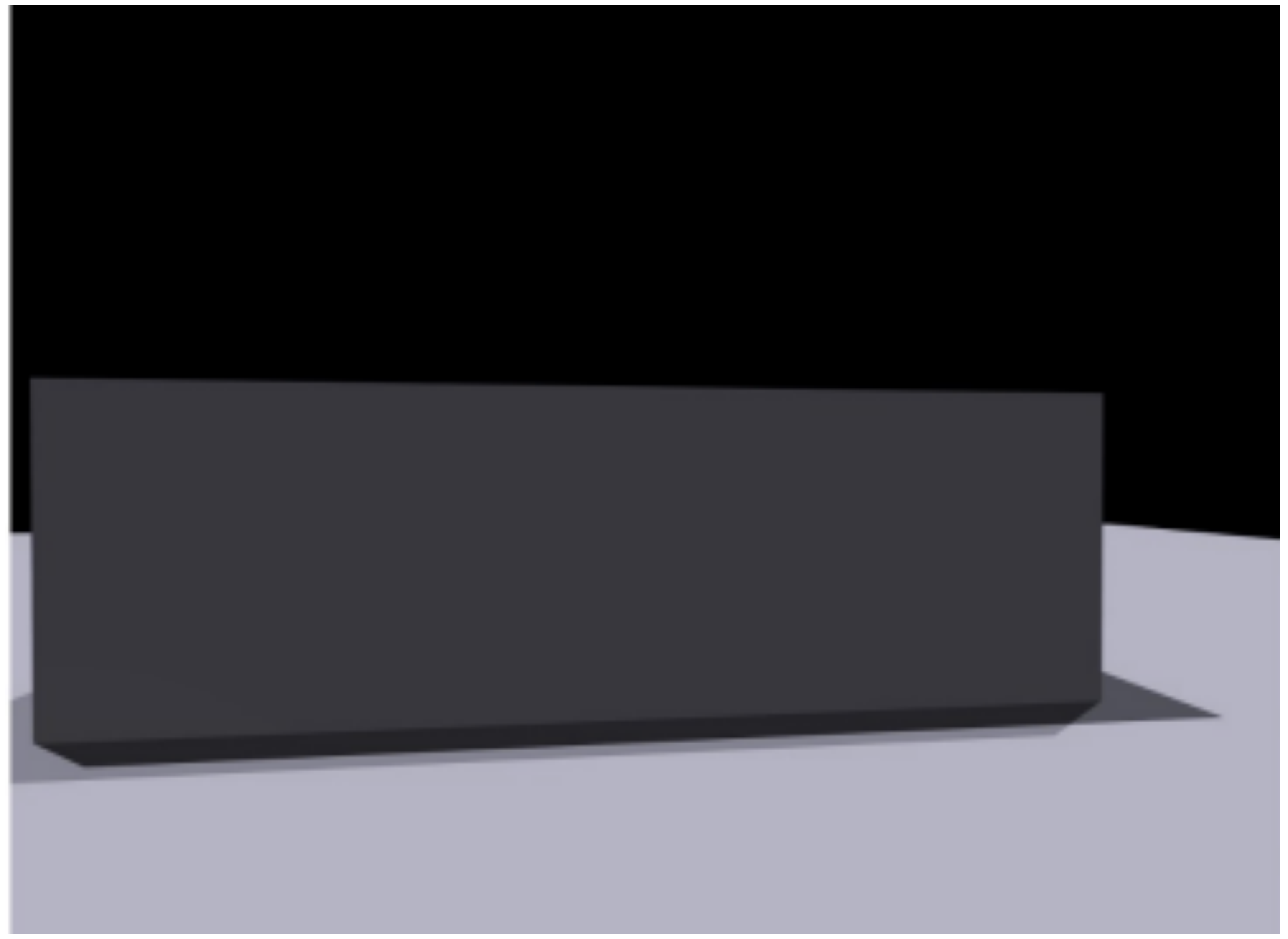
If all state-action pairs are visited often enough, tabular Q-learning converges to Q-value

From Tabular to Deep RL

- The size of the Q table grows exponentially with the state and action dimensions
- Tabular RL can only work for small state-action spaces

From Tabular to Deep RL

- The size of the Q table grows exponentially with the state and action dimensions
- Tabular RL can only work for small state-action spaces
- How can the robot learn parkour?
  - No transitions and reward models available
  - High dimensional state space

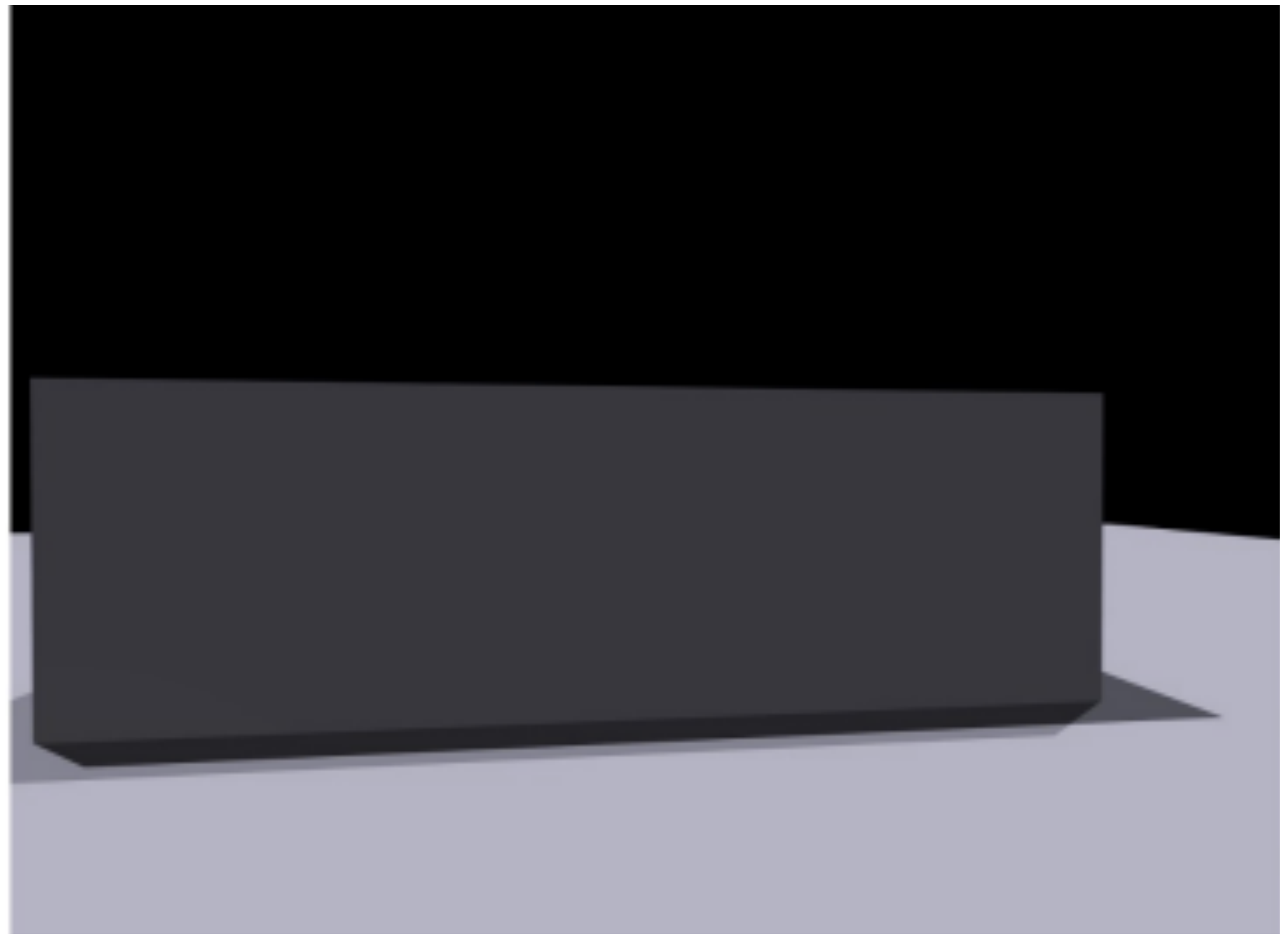


Robot Parkour vision of an obstacle

$$|S| = 48 \times 64 \times 255 \approx 10^6$$

From Tabular to Deep RL

- The size of the Q table grows exponentially with the state and action dimensions
- Tabular RL can only work for small state-action spaces
- How can the robot learn parkour?
  - No transitions and reward models available
  - High dimensional state space
- Idea: use a neural network to approximate Q

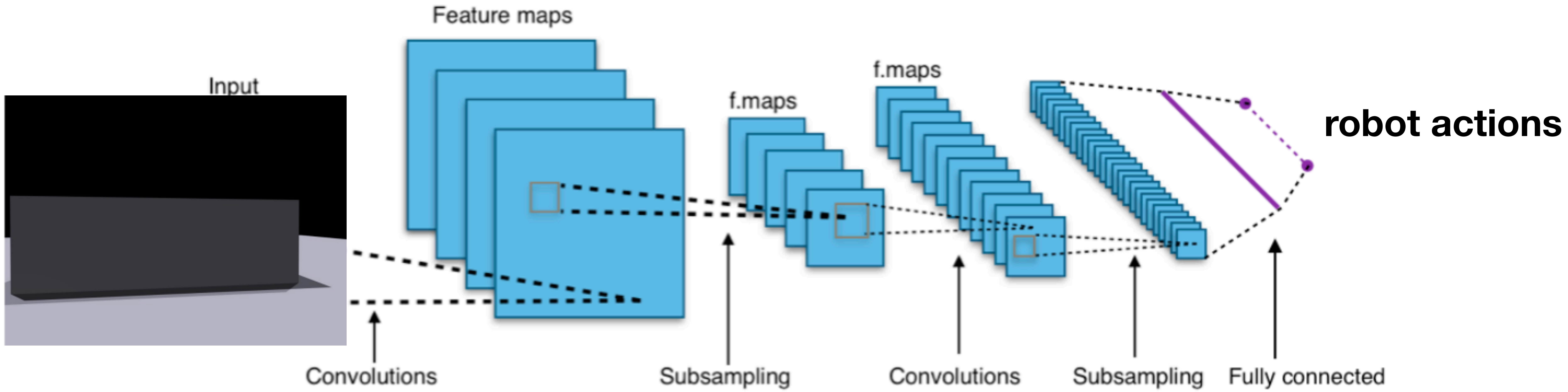


Robot Parkour vision of an obstacle

$$|S| = 48 \times 64 \times 255 \approx 10^6$$

Deep Q-Network

Convolutional neural networks $q(s;\theta) \in \mathbb{R}^{|A|}$: output Q-value approximation per action



Gradient descent on mean-squared TD error

$$\theta \longleftarrow \theta - \alpha \nabla \Bigg(\underbrace{r + \gamma \max_{a'} q(s';\theta))_{a'} - q(s;\theta)_a}_{\text{TD-error}} \Bigg)^2$$

Summary

Topics

- Sequential decision making: agents, environment, reward…
- MDPs
- Planning and Value Iteration
- Reinforcement learning and Tabular Q-learning

Reading material

- Reinforcement Learning: An Introduction - Chapter 3
- Wendelin Böhmer - Learning to interact, tabular Q-learning