An introduction to Reinforcement Learning

05th of July 2022

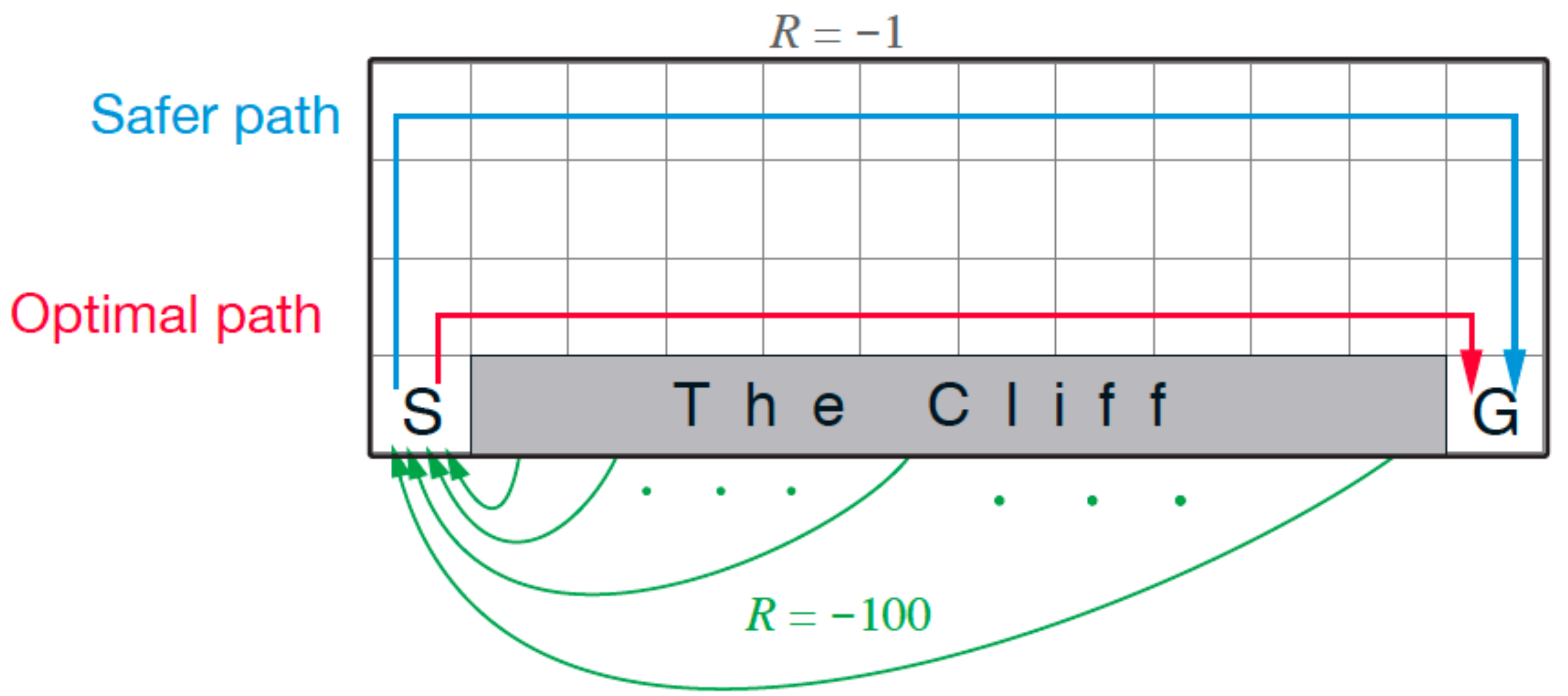
Q-Learning

Limitation of multi-armed bandit problems

Your current action does not influence what happens next!!

How can we solve sequential problems?

The textbook problem:
‘Cliff-World’



The rules:

- Agent has to move from start (S) to goal (G)
- Reaching the goal results in a positive reward of +10
- Falling off the cliff results in a negative reward of -100
- Any other state results in a negative reward of -1

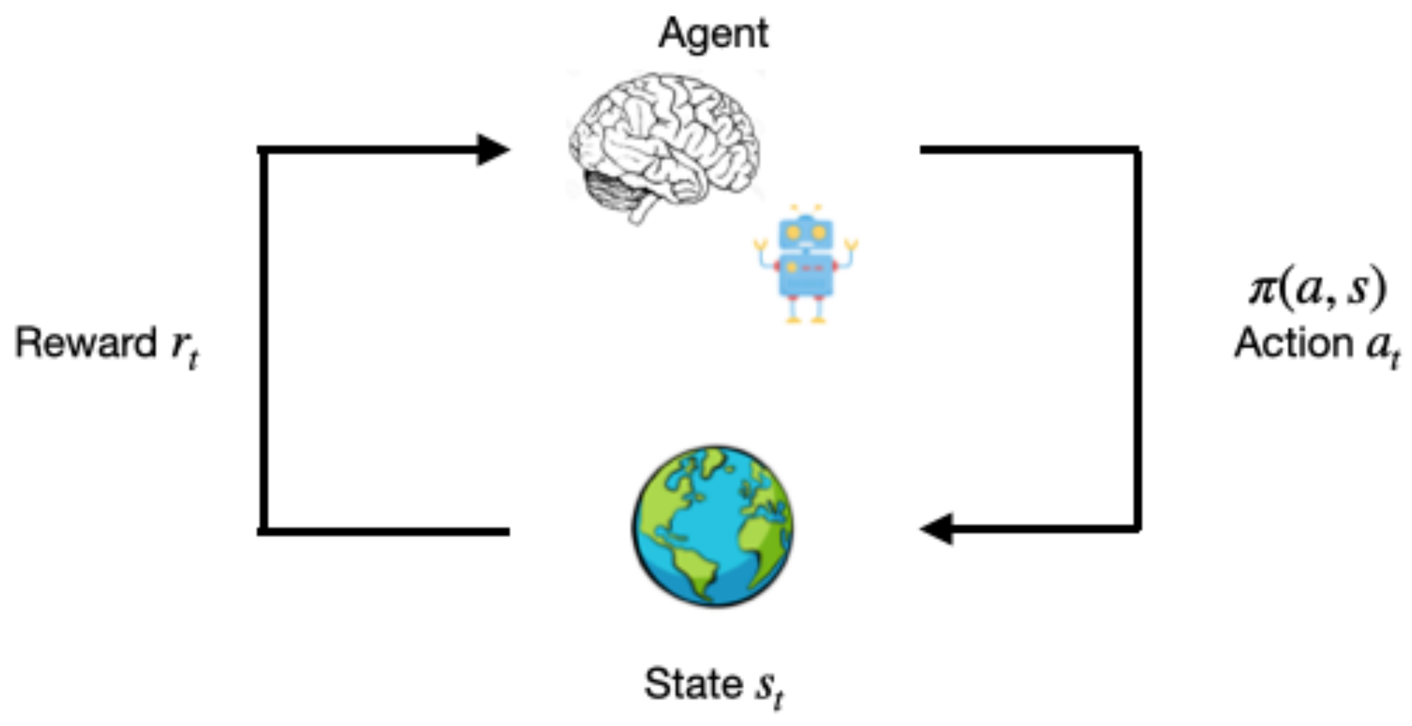
What’s the problem the agent has to solve here??

Note the subtle introduction of the concept of ‘**transition probabilities**’ here
- implicit, later: explicit

From classical to instrumental learning

TD Learning:

$$V(s_t) \leftarrow V(s_t) + \underbrace{\alpha}_{\text{Learning rate}} \cdot \overbrace{\left(r + \underbrace{\gamma}_{\text{Discount rate}} \cdot V(s_{t+1}) - V(s_t)\right)}^{\text{Prediction error}}$$



Q-Learning:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \underbrace{\alpha}_{\text{Learning rate}} \cdot \overbrace{\left(r + \underbrace{\gamma}_{\text{Discount rate}} \cdot max_a Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right)}^{\text{Prediction error}}$$

What's the difference between $V(s_t)$ and $Q(s_t, a_t)$?

What's is $max_a Q(s_t, a_t)$ doing?

Note that this is just an update rule - doesn't tell us how to select an action!

Coding: Q-Learning

https://github.com/schwartenbeckph/RL-Course/tree/main/2022_07_05

MDPs

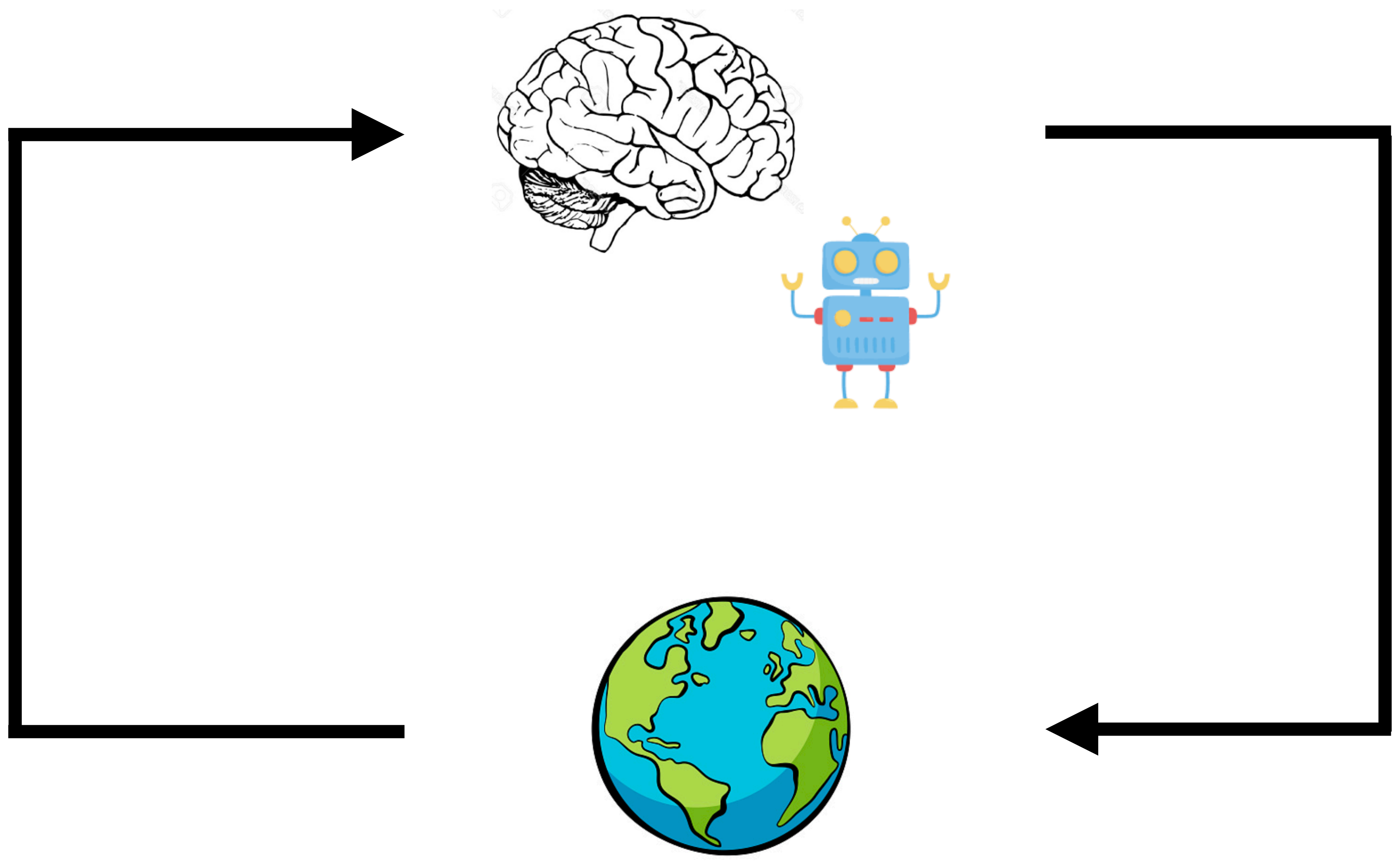
Basic setup: how do agents learn to act?

Based on a reward signal, agents learn **values of actions/states:**

$$V_\pi(s) = \mathbb{E}_\pi[R \mid s_0 = s]$$

Action is governed by a **policy**:

$$\pi(a, s) = P(a_t = a \mid s_t = s)$$



Reward r_t

Action a_t

State s_t

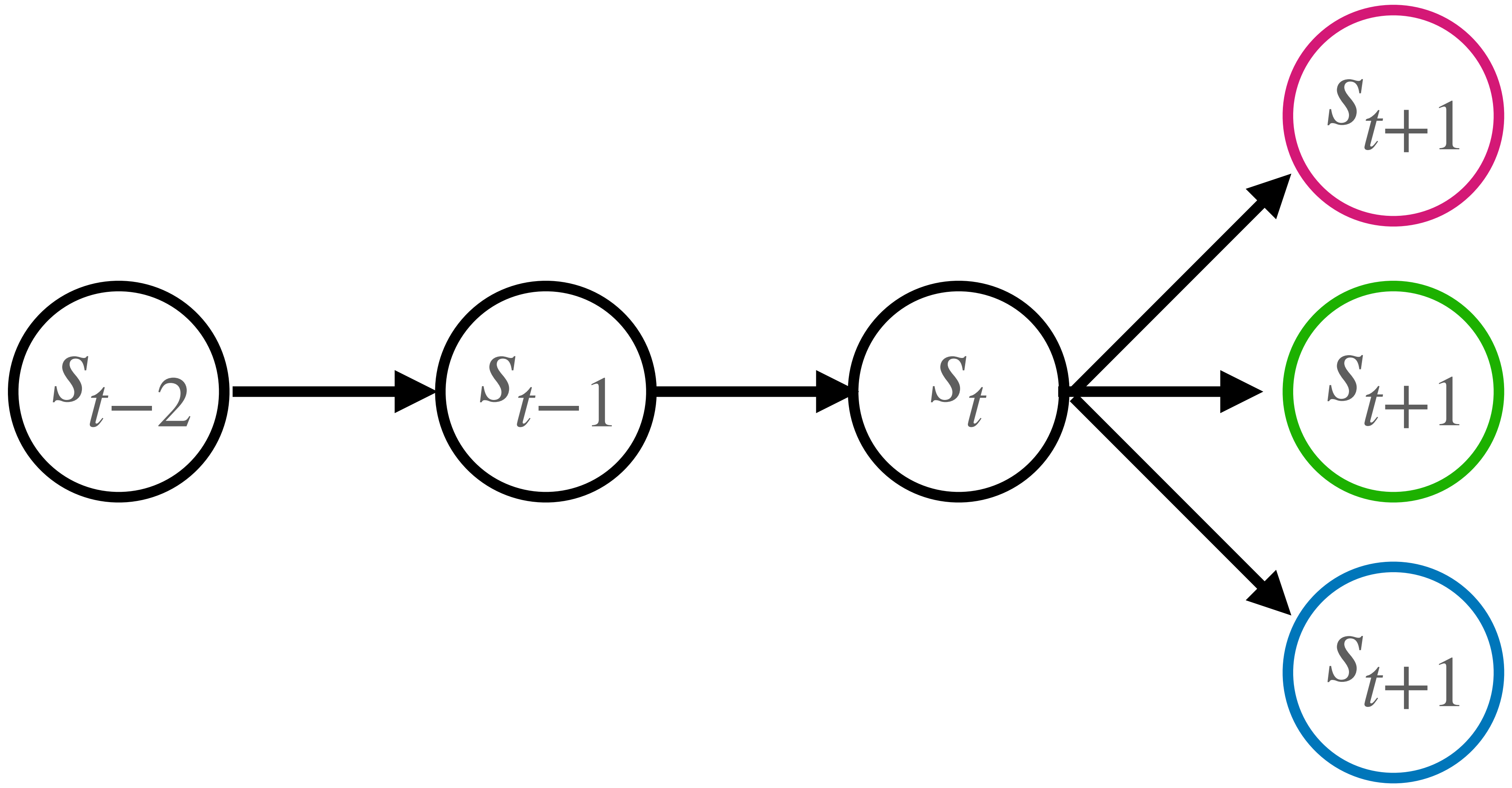
Agents can learn a **model of the environment** to make smarter decisions, e.g.:

$$P(s_{t+1} = s \mid s_t = s, a_t = a)$$

Markov Process → Markov Reward Process → **Markov Decision Process (MDP)**

Markov Process

Most RL problems are problems where agents face sequences of states:

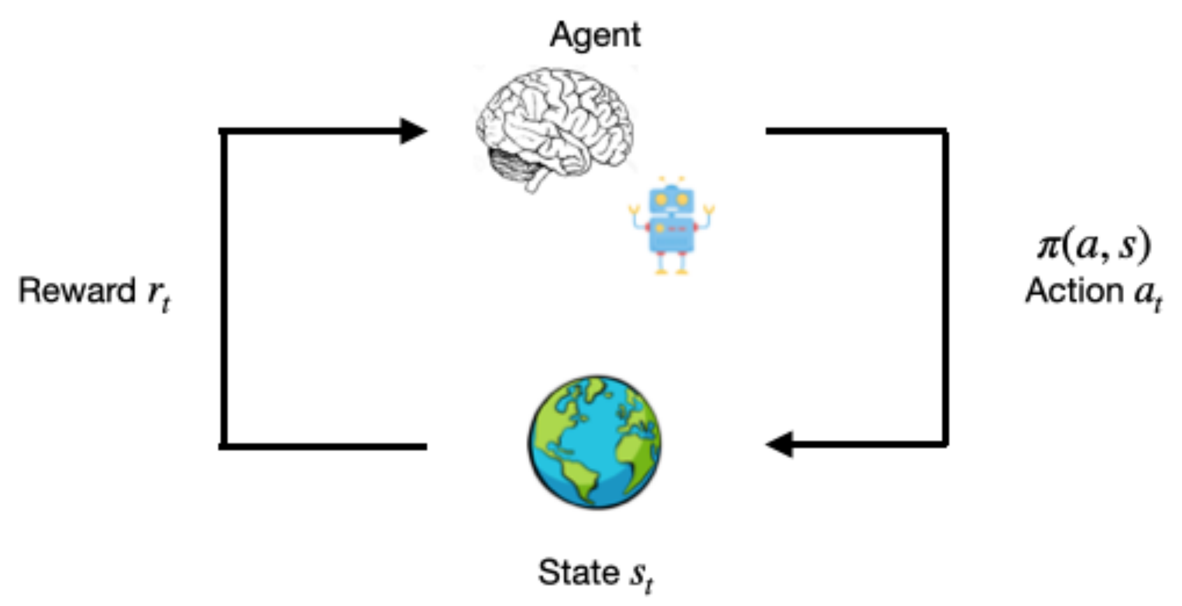


Fundamental property: **Markov property**

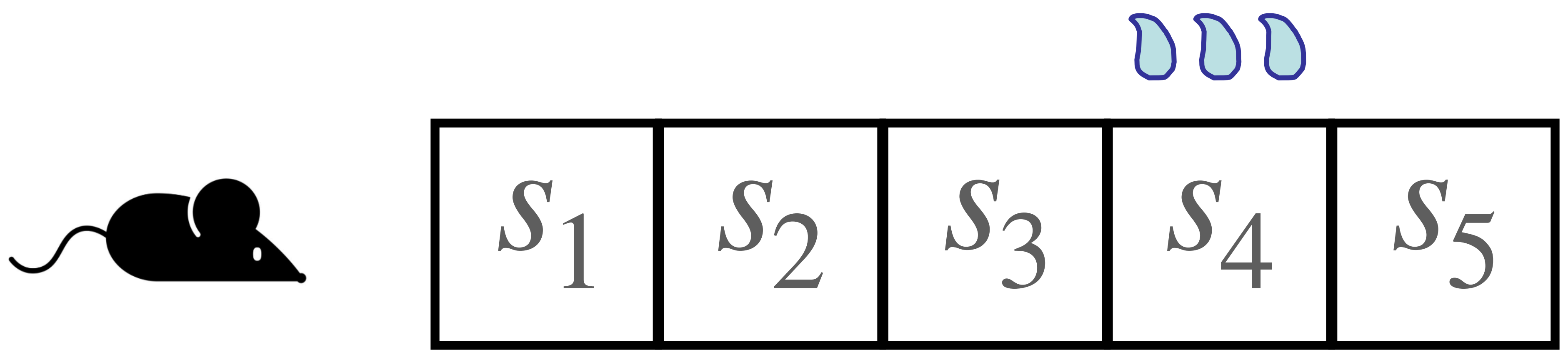
$$P(s_{t+1} = s \,|\, s_t, s_{t-1}, s_{t-2}, \ldots) = P(s_{t+1} = s \,|\, s_t)$$

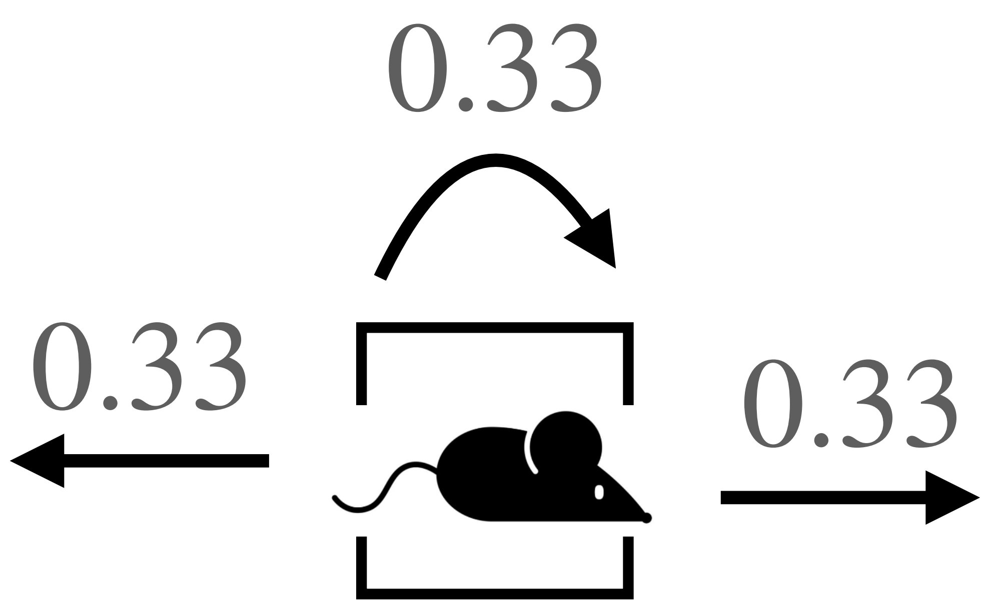
"The future is independent of the past given the present"

Markov Process

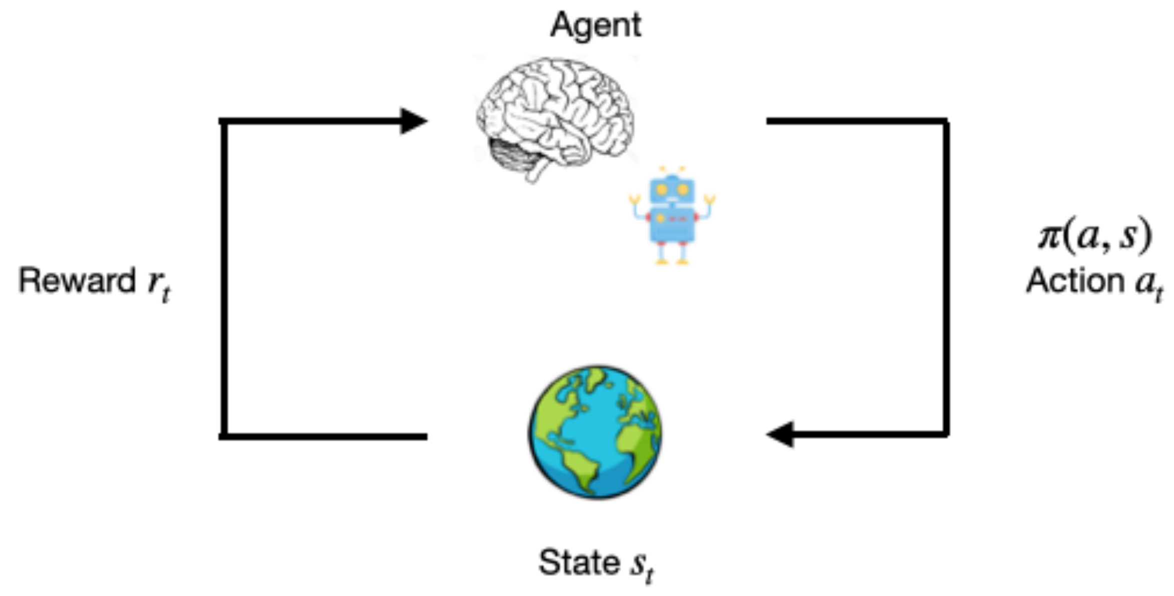


Let’s assume a super simple problem:

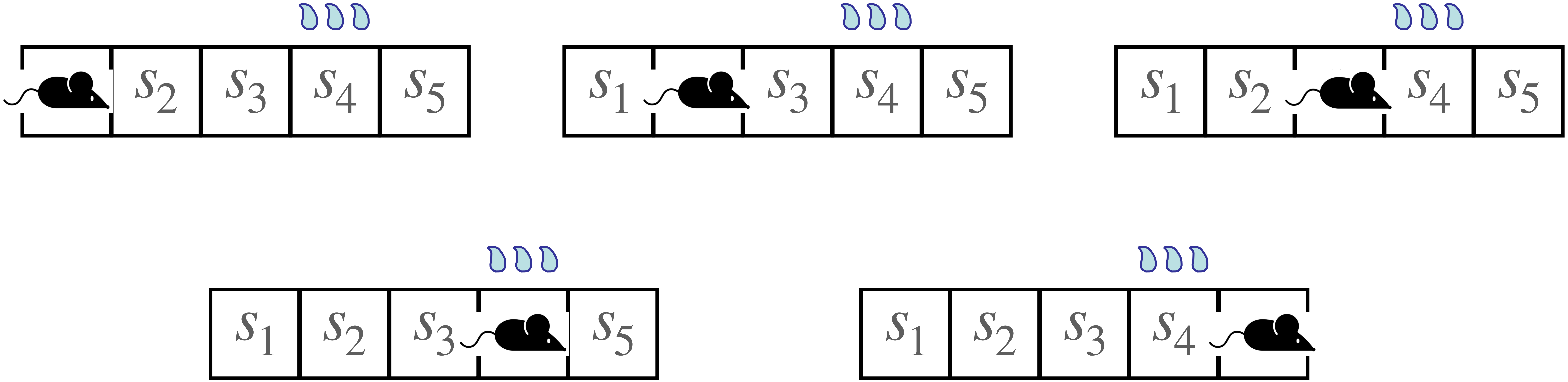




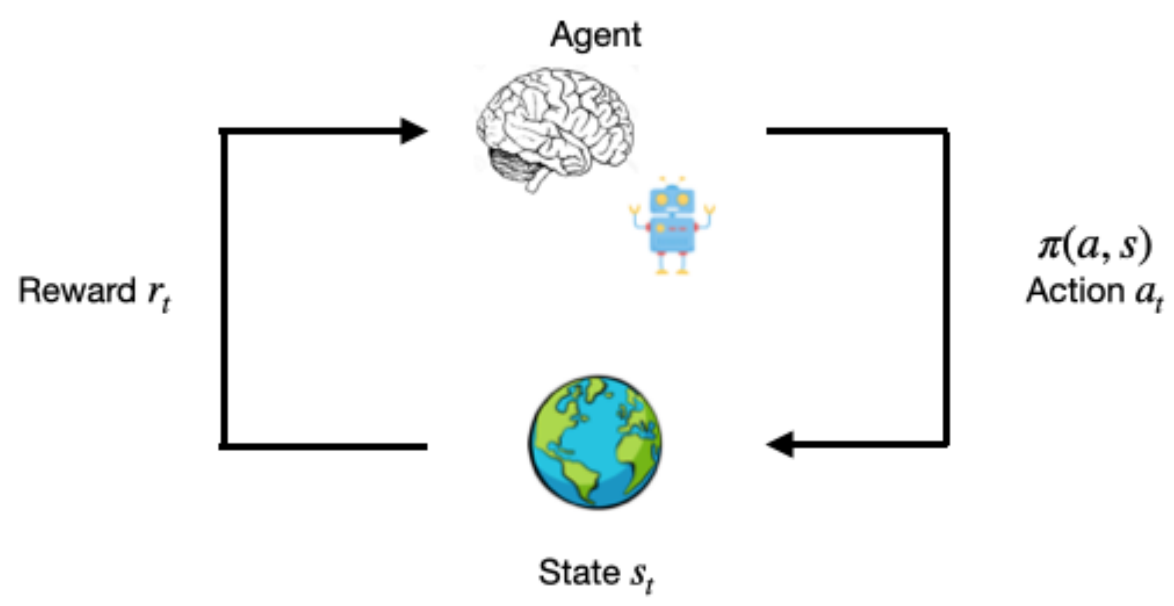
Markov Process



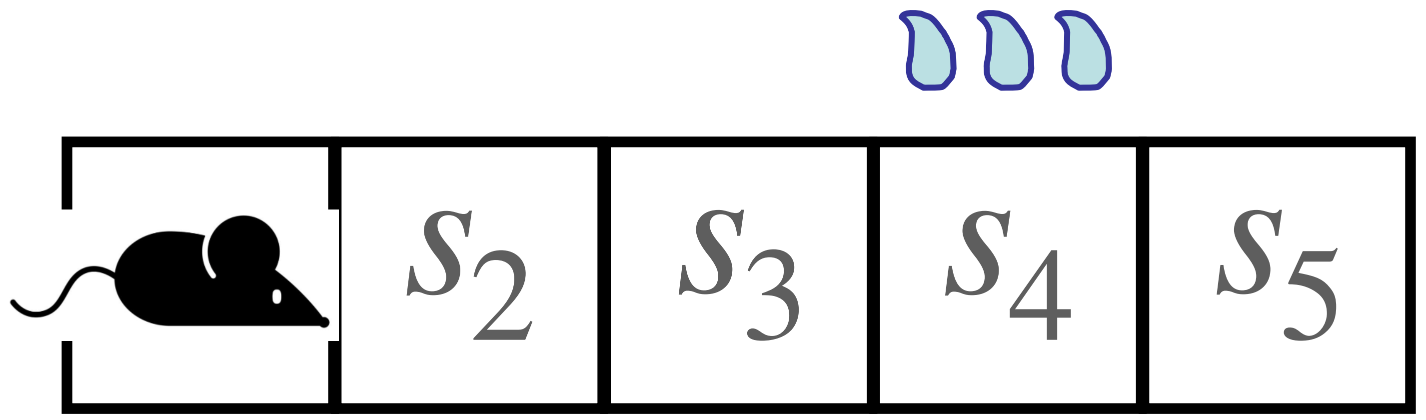
We can now define a **state space** S:



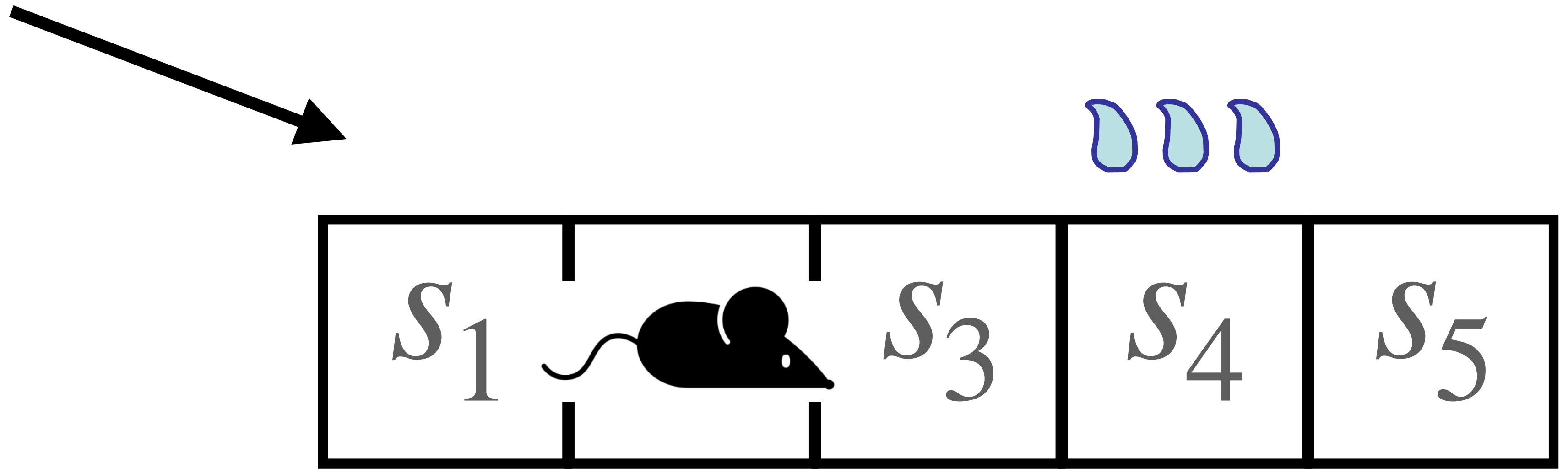
Markov Process

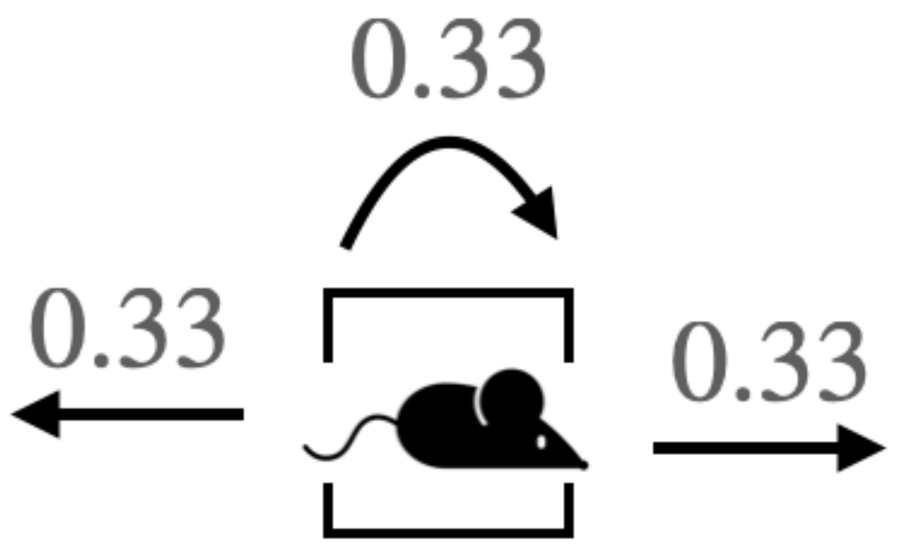


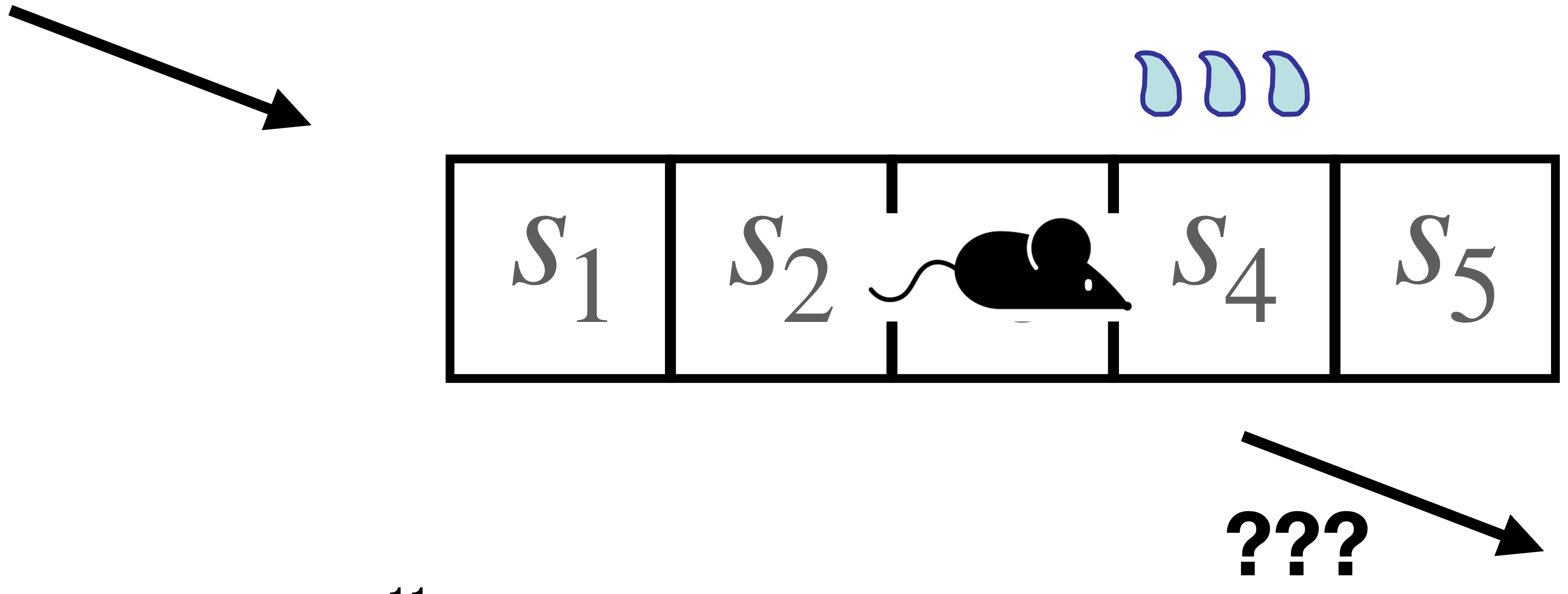
Does this problem have the **Markov Property?**



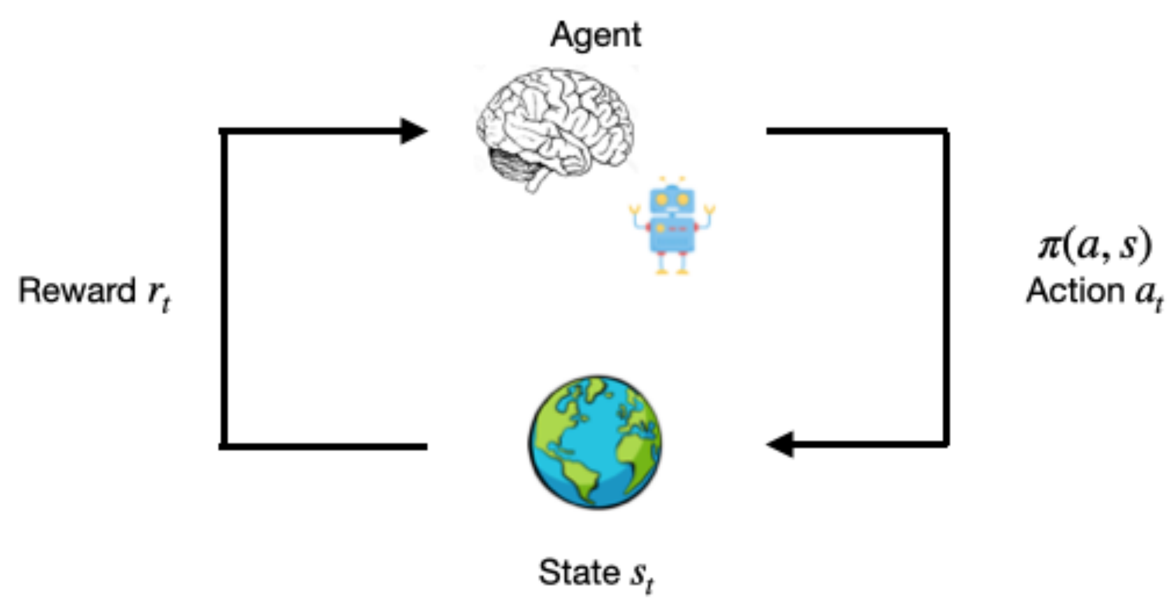
$$P(s_{t+1} = s \,|\, s_t, s_{t-1}, s_{t-2}, \ldots) = P(s_{t+1} = s \,|\, s_t)$$





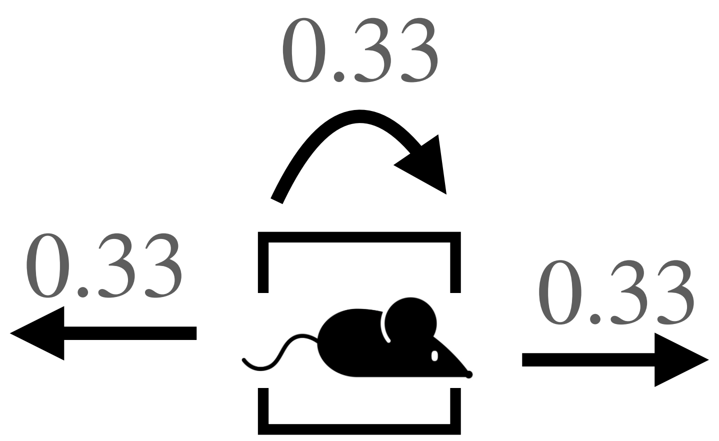


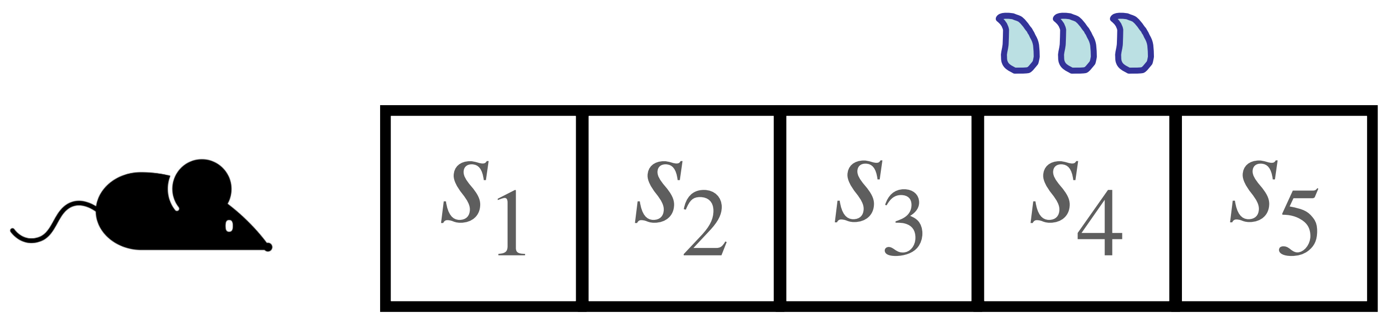
Markov Process



This allows us to define **transition probabilities** P:

$$P(s_{t+1} = s' \mid s_t = s)$$



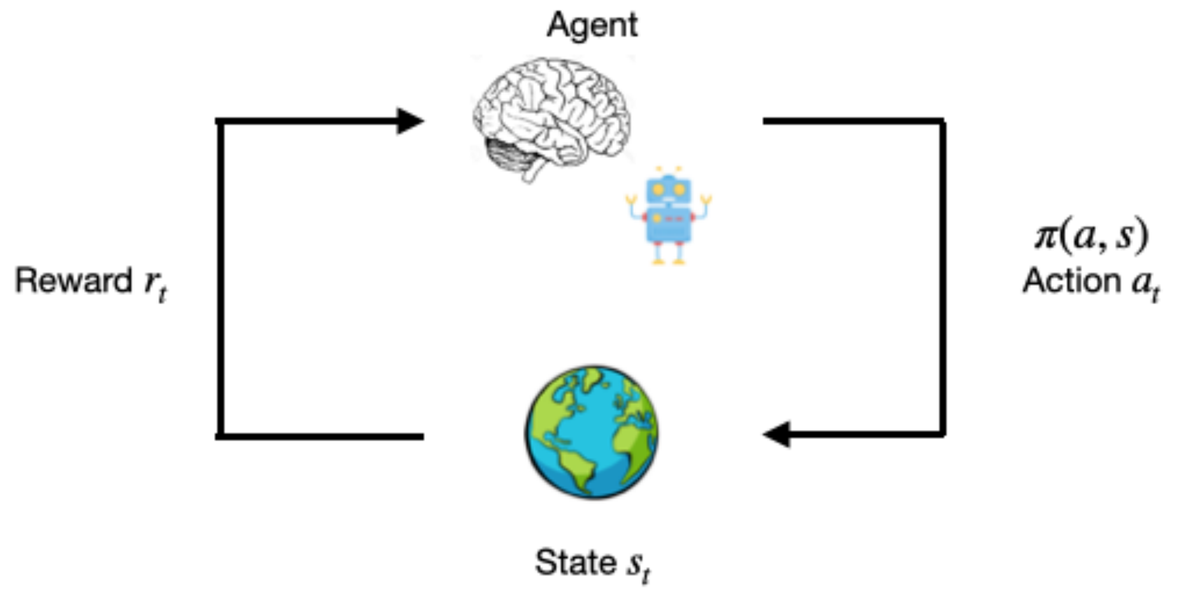


To

From

$$\begin{array}{c c} & \begin{array}{ccccc} \boxed{s_1} & \boxed{s_2} & \boxed{s_3} & \boxed{s_4} & \boxed{s_5} \end{array} \\ \begin{array}{c} \boxed{s_1} \\ \boxed{s_2} \\ \boxed{s_3} \\ \boxed{s_4} \\ \boxed{s_5} \end{array} & \begin{bmatrix} 0.66 & 0.33 & 0 & 0 & 0 \\ 0.33 & 0.33 & 0.33 & 0 & 0 \\ 0 & 0.33 & 0.33 & 0.33 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0.33 & 0.66 \end{bmatrix} \end{array}$$

Markov Reward Process

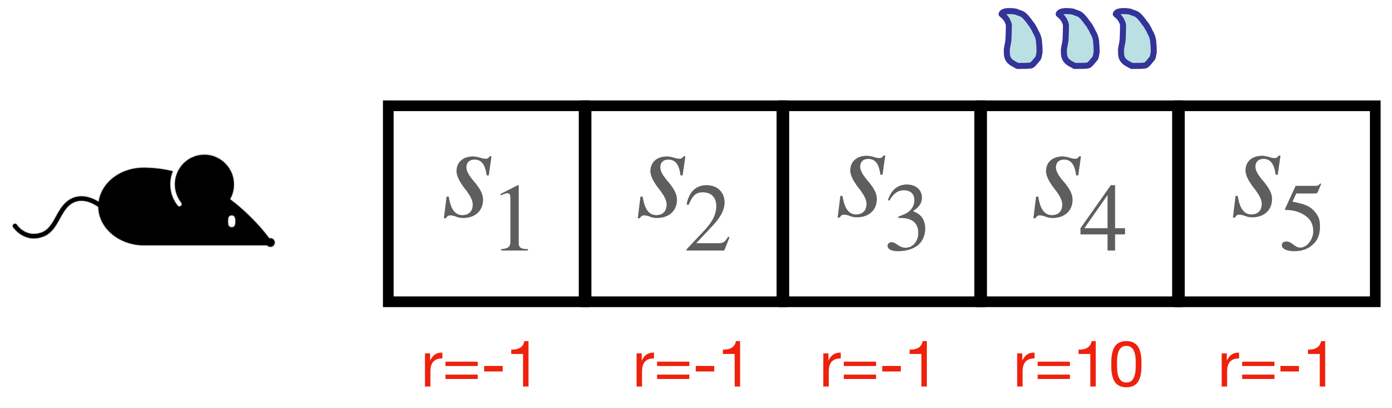


A **Markov Process** is defined based on

- A **State Space** S
- **Transition Probabilities** P $\quad P(s_{t+1} = s' \mid s_t = s)$

A **Markov Reward Process** is defined based on

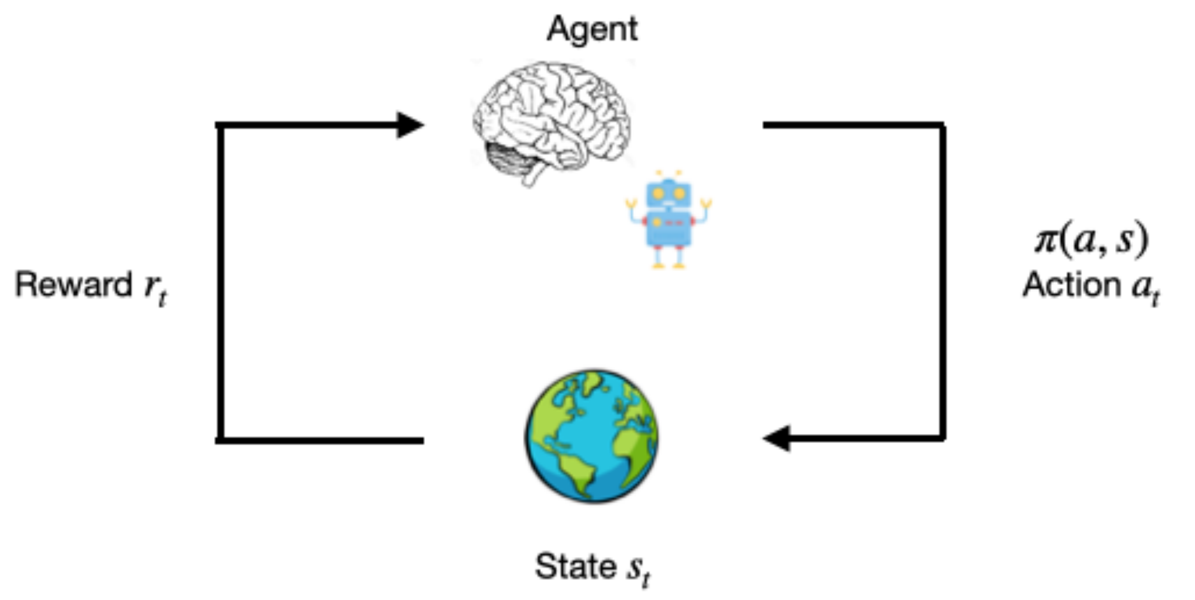
- A State Space S
- Transition Probabilities P
- A **Reward Function** $R_s = \mathbb{E}[r_{t+1} \mid s_t = s]$
- A **Discount Factor** $\gamma \in [0,1]$



Allows to define **Return**

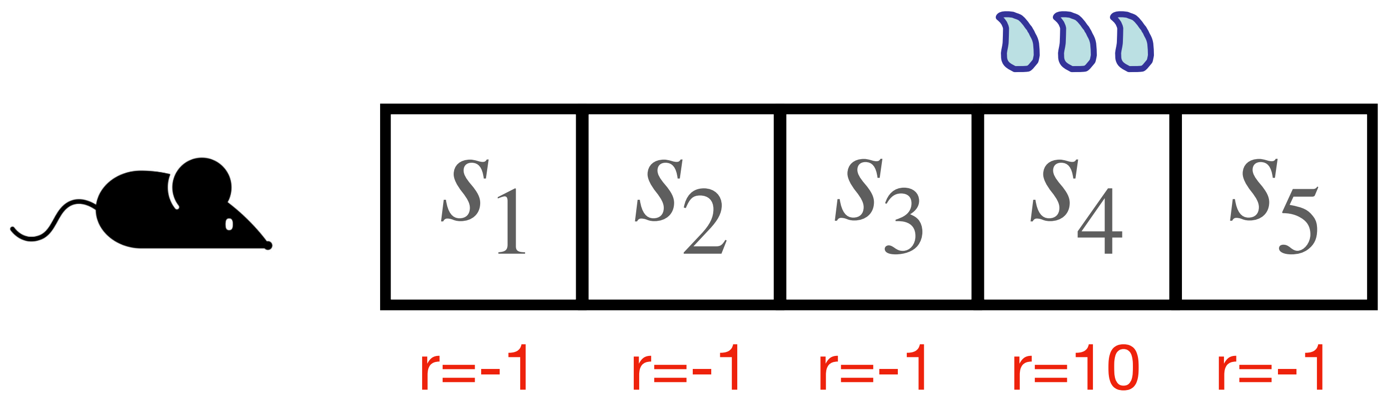
$$G_t = r_{t+1} + \gamma \cdot r_{t+2} + \gamma^2 \cdot r_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k \cdot r_{t+k+1}$$

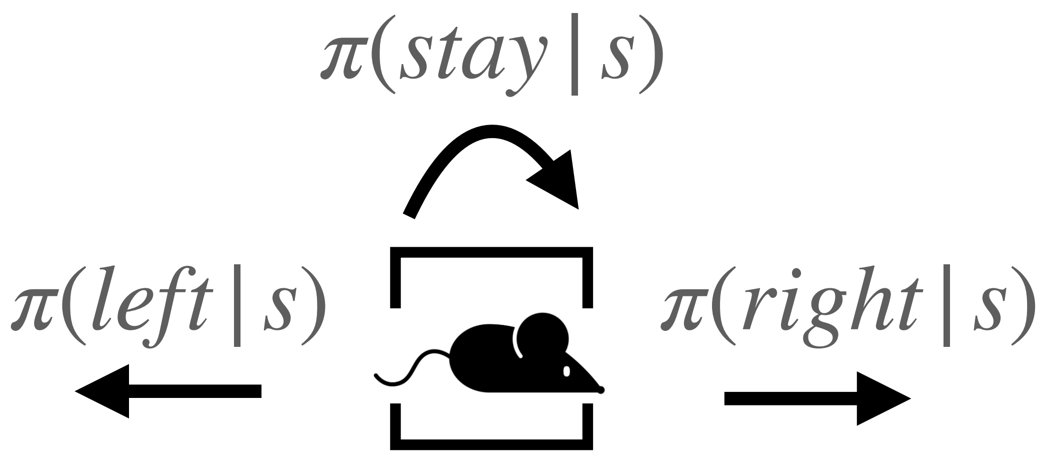
Markov Decision Process



A **Markov Decision Process** is defined based on

- A State Space S
- An **Action Space** A
- Transition Probabilities P
- A Reward Function $R_s = \mathbb{E}[r_{t+1} \mid s_t = s]$
- A Discount Factor $\gamma \in [0,1]$



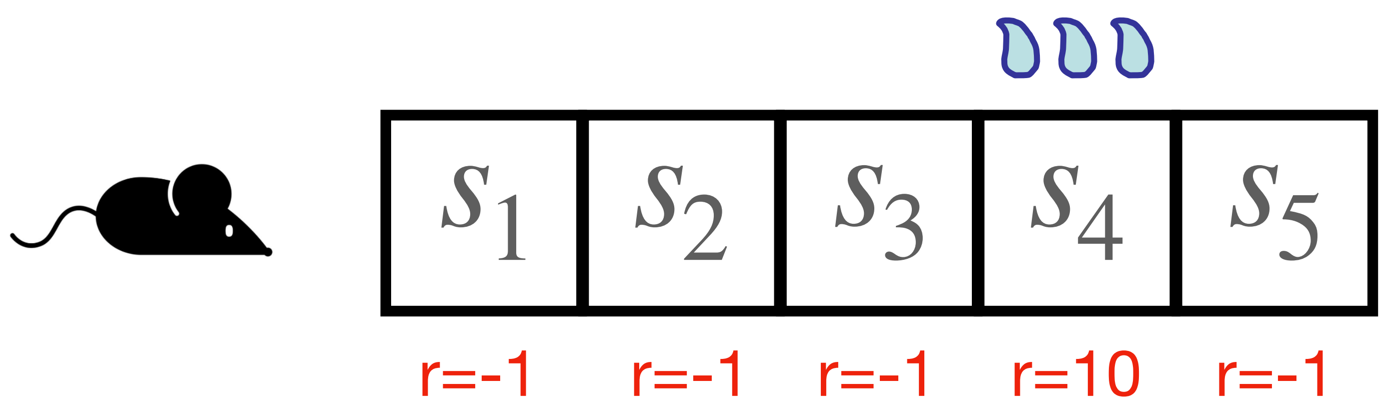


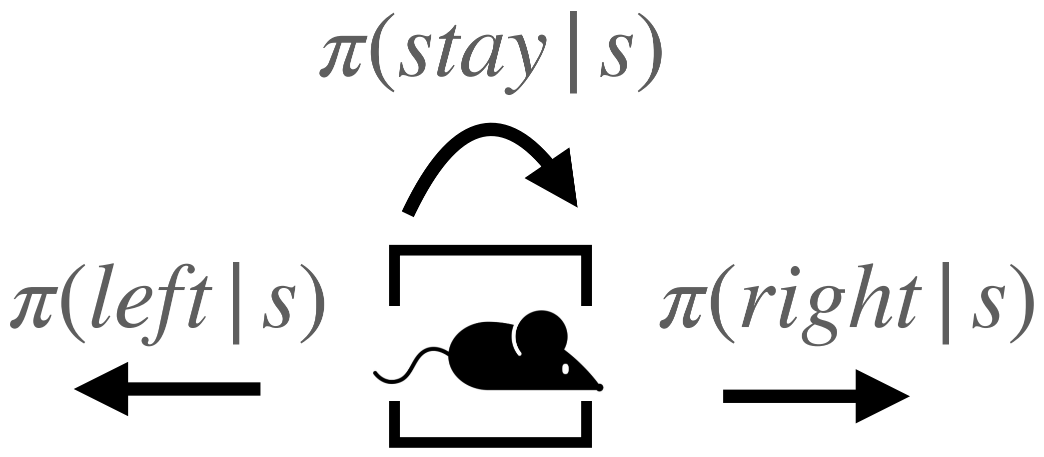
Actions are governed via a **policy**: $\pi(a, s) = P(a_t = a \mid s_t = s)$

MDPs basis for model-based RL

Allows to specify all environment dynamics for RL problem:

$$P(s', r \mid s, a) = P(s_{t+1} = s', r_{t+1} = r \mid s_t = s, a_t = a)$$



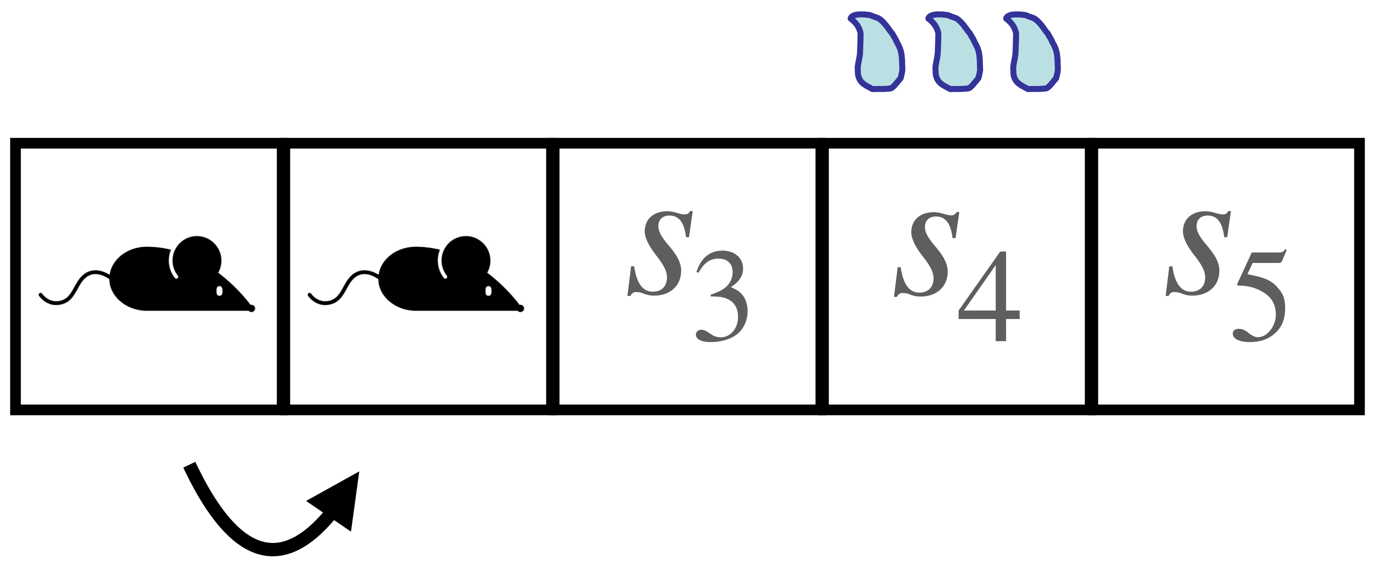


MDPs basis for model-based RL

$$P(s', r \,|\, s, a) = P(s_{t+1} = s', r_{t+1} = r \,|\, s_t = s, a_t = a)$$

Allows to specify useful thinks like **state-transition probabilities** (often T):

$$P(s' \,|\, s, a) = P(s_{t+1} = s' \,|\, s_t = s, a_t = a) = \sum_r P(s', r \,|\, s, a)$$



$$P(s' \,|\, s, right) = \begin{array}{c c} & \begin{array}{ccccc} s_1 & s_2 & s_3 & s_4 & s_5 \end{array} \\ \begin{array}{c} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \end{array} & \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$