

# Reinforcement Learning

Shusen Wang

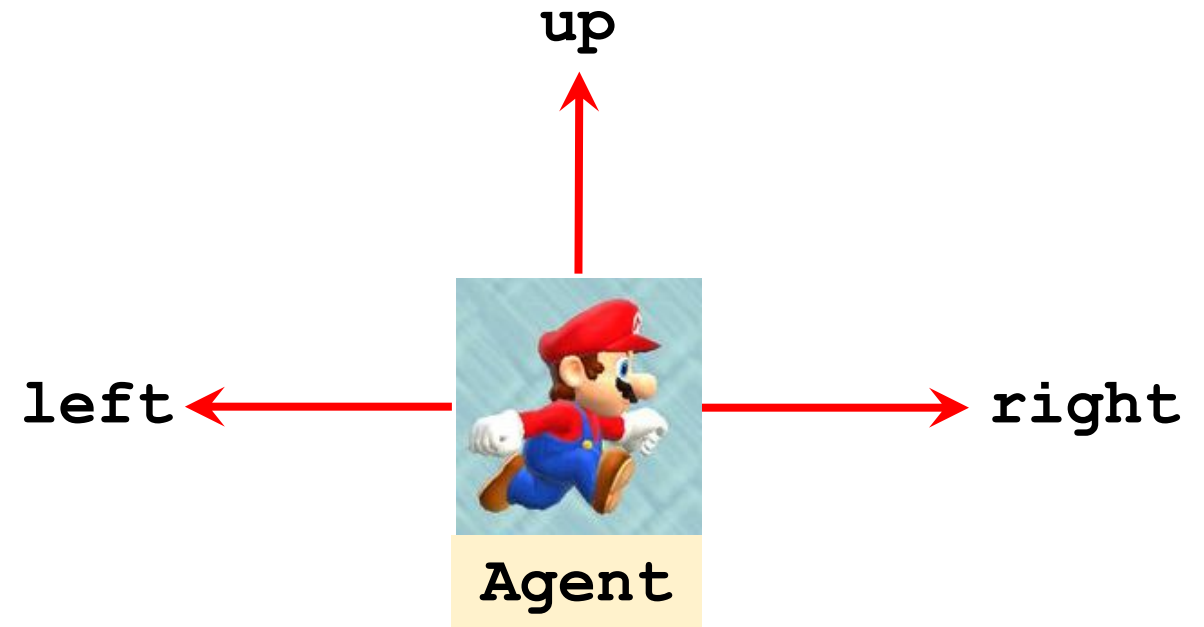
# Terminologies

# Terminology: state and action

state  $s$  (this frame)

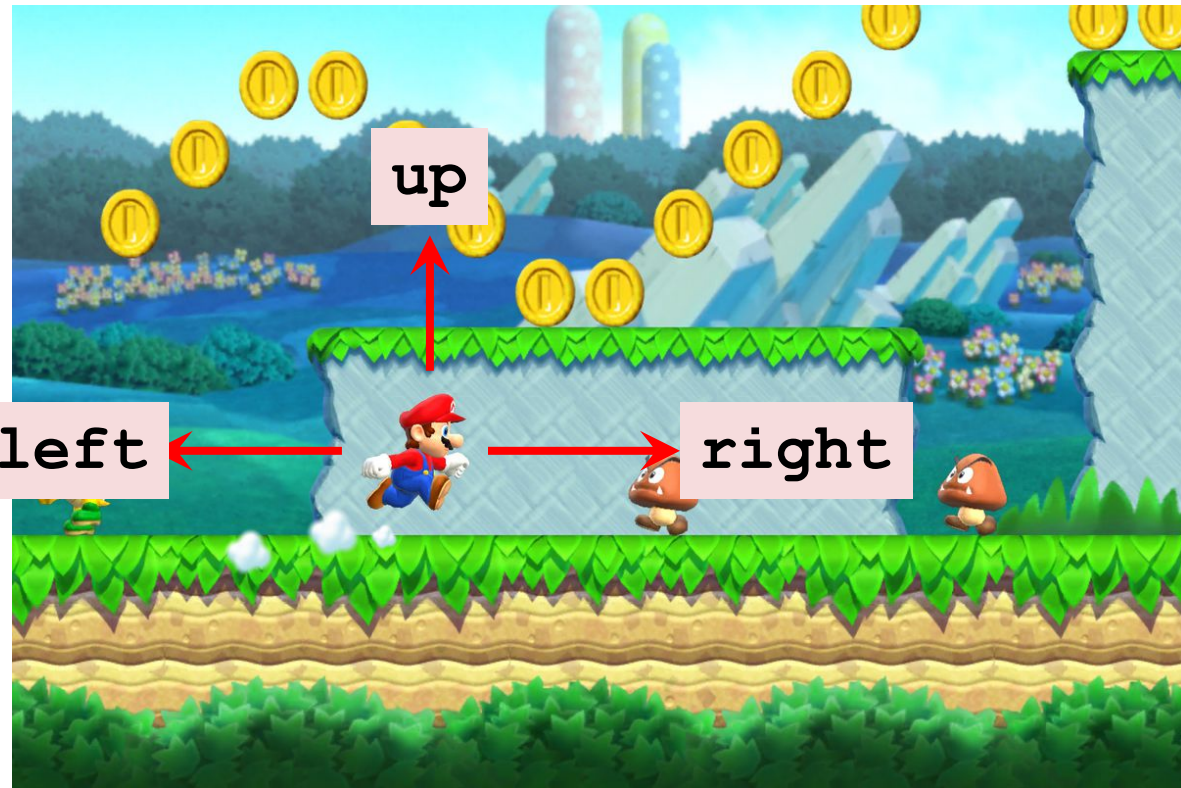


Action  $a \in \{\text{left}, \text{right}, \text{up}\}$



# Terminology: policy

## policy $\pi$



- Policy function  $\pi: (s, a) \mapsto [0,1]$ :  
$$\pi(a | s) = \mathbb{P}(A = a | S = s).$$
- It is the probability of taking action  $a$  given state  $s$  and policy  $\pi$ .
- Upon observing state  $s$ , the agent's action  $a$  can be random, e.g.,
  - $\pi(\text{left} | s) = 0.2$ ,
  - $\pi(\text{right} | s) = 0.1$ ,
  - $\pi(\text{up} | s) = 0.7$ .

# Terminology: reward

reward  $r$

- Collect a coin:  $r = +1$





# Terminology: reward

reward  $r$



- Collect a coin:  $r = +1$
- Win the game:  $r = +10000$

# Terminology: reward

reward  $r$



- Collect a coin:  $r = +1$
- Win the game:  $r = +10000$
- Touch a Goomba:  $r = -10000$  (game over).

# Terminology: reward

## reward $r$

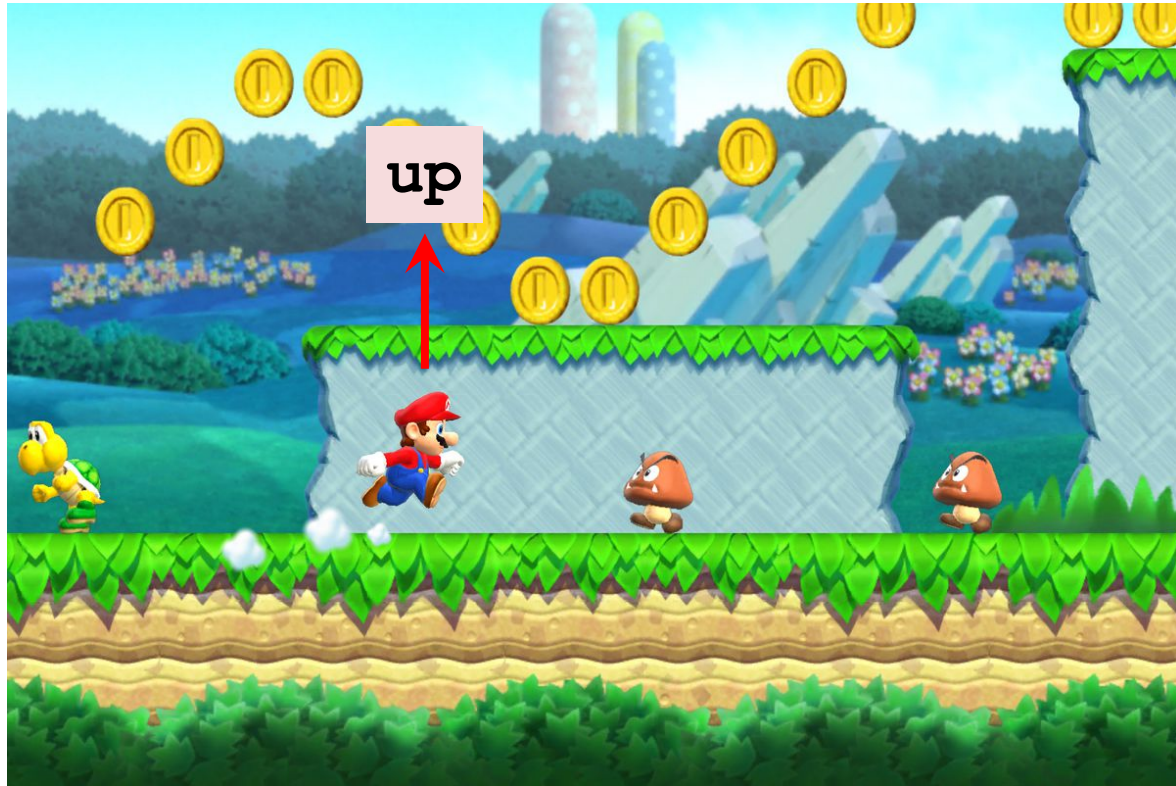


- Collect a coin:  $r = +1$
- Win the game:  $r = +10000$
- Touch a Goomba:  $r = -10000$  (game over).
- Nothing happens:  $r = 0$



# Terminology: state transition

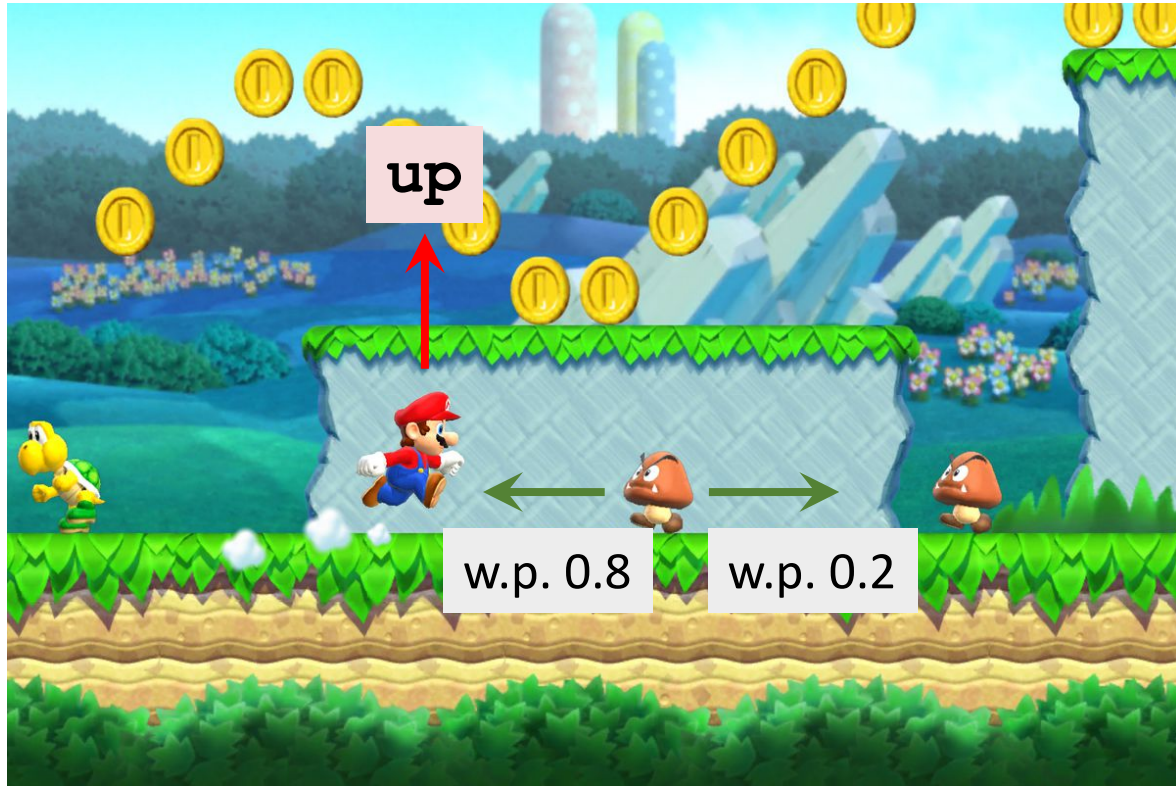
## state transition



- $(s, a) \mapsto s'$ .
- E.g., “up” action leads to a new state.

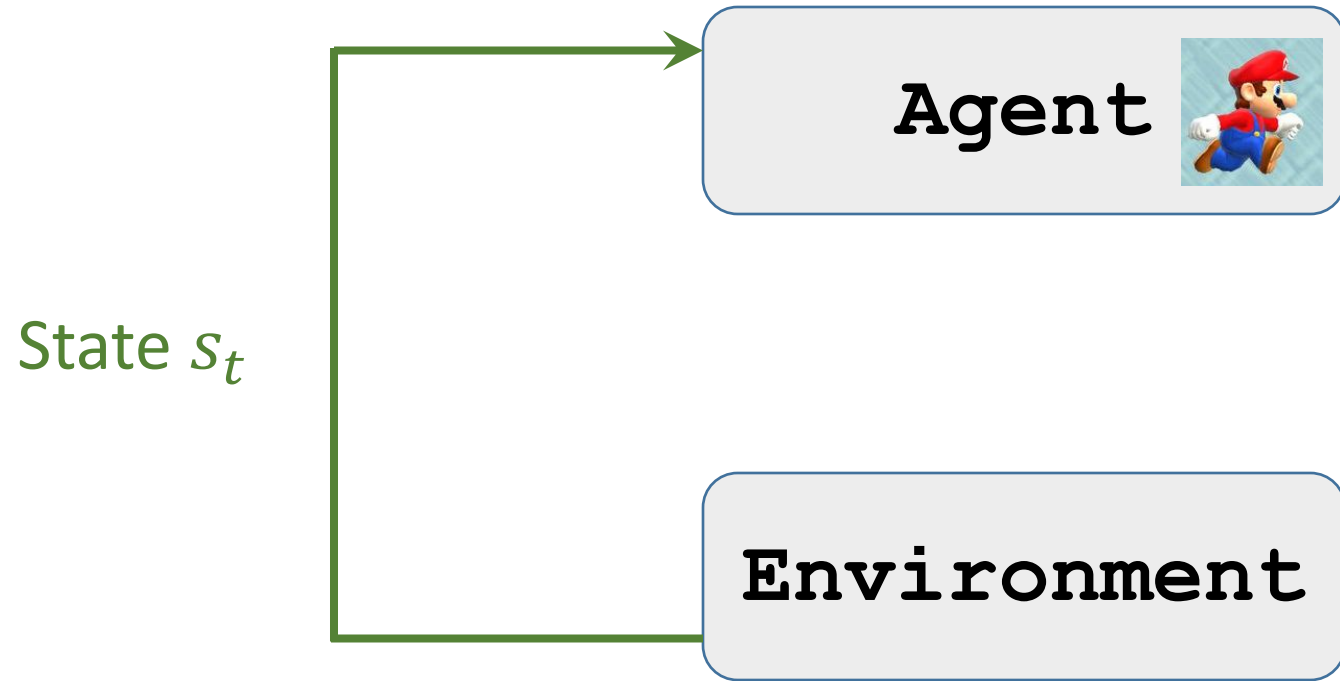
# Terminology: state transition

## state transition

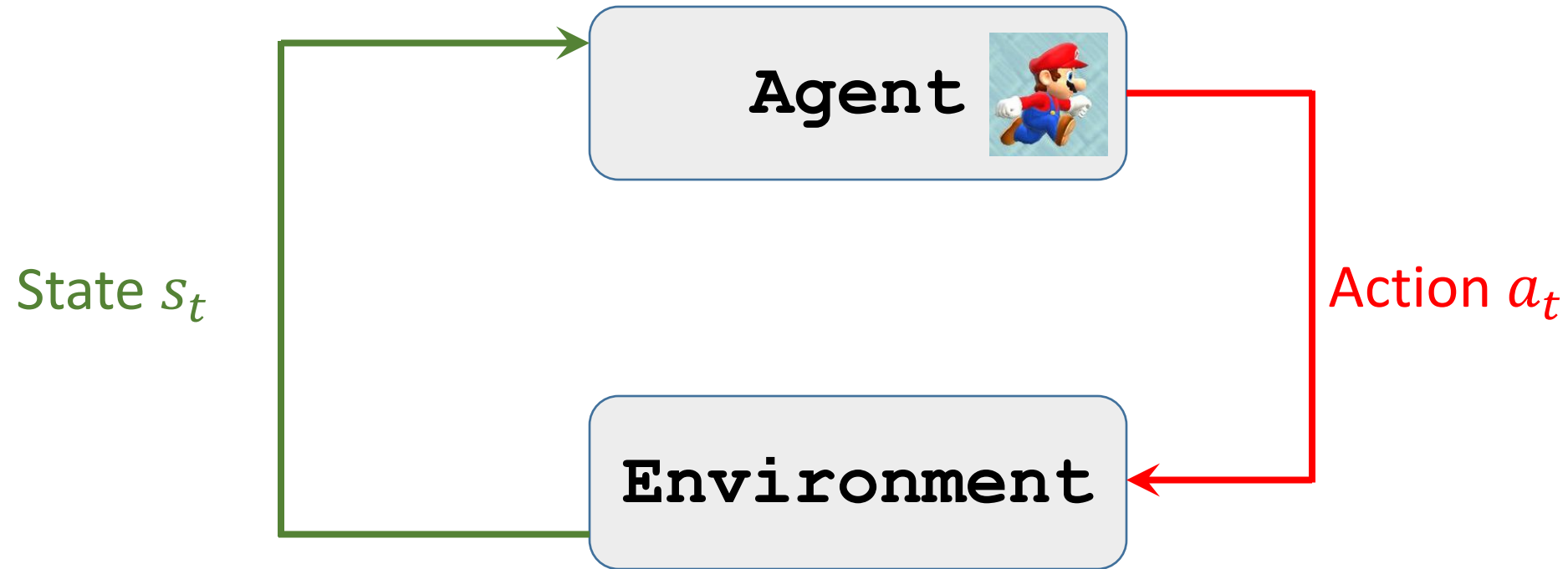


- $(s, a) \mapsto s'$ .
- E.g., “up” **action** leads to a new **state**.
- State transition can be random.
- Randomness is from the environment.
- $p(s' | s, a) = \mathbb{P}(S' = s' | S = s, A = a)$ .

# Terminology: agent environment interaction

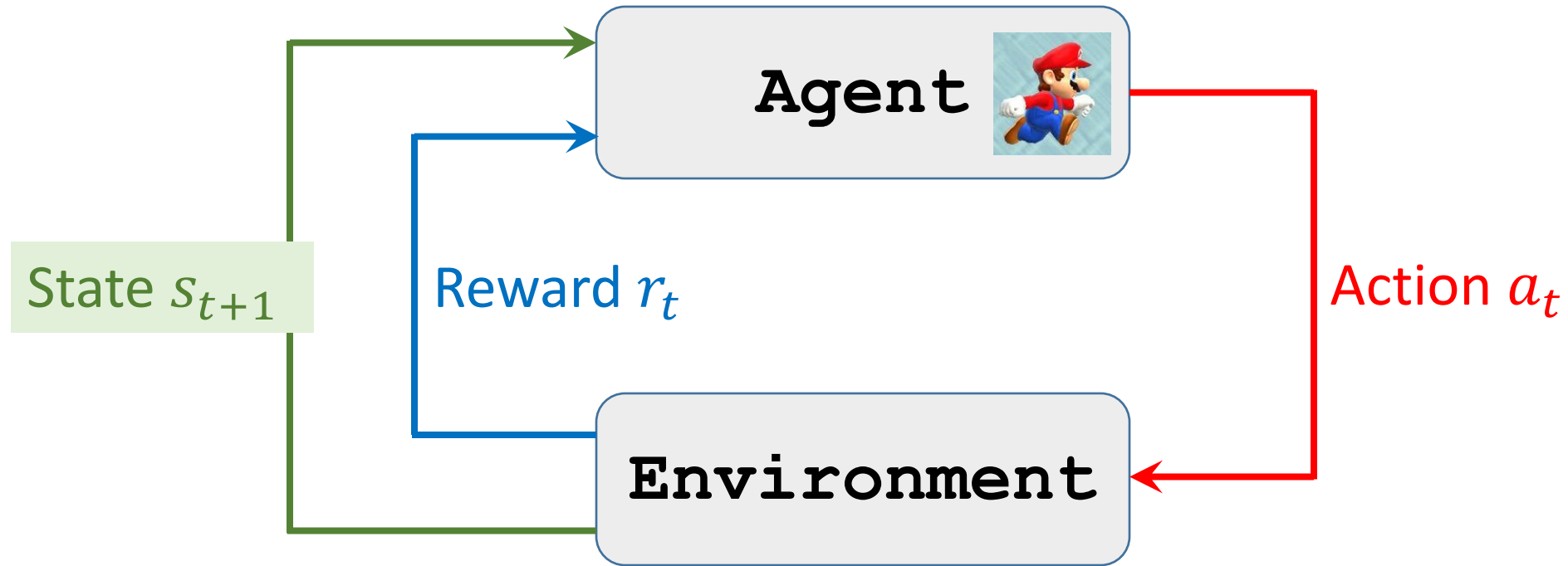


# Terminology: agent environment interaction

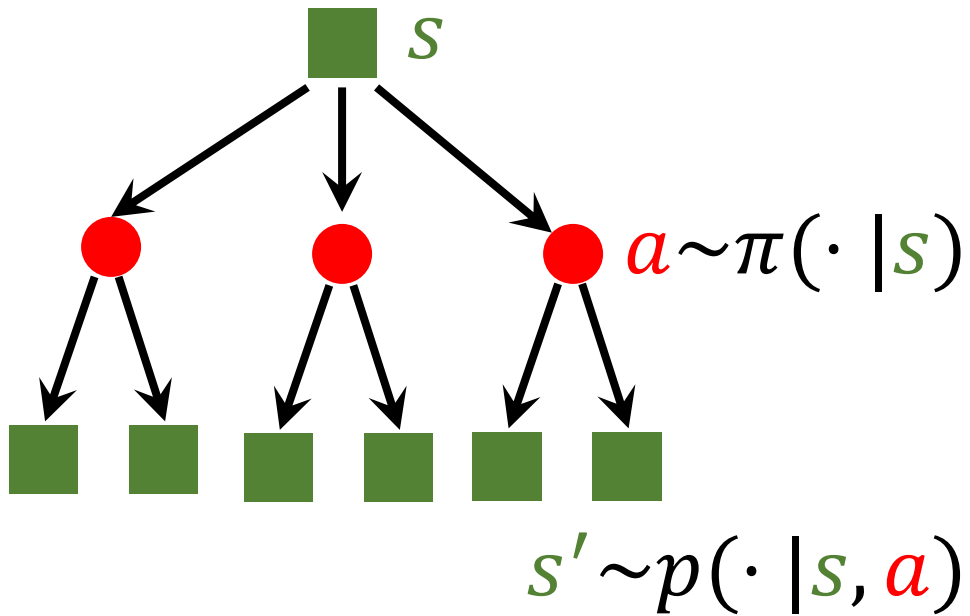




# Terminology: agent environment interaction



# Randomness in Reinforcement Learning



Policy  $\pi(a|s)$  can be random.

- Given state  $s$  and policy  $\pi$ , the action  $a$  can be random, e.g., .
  - $\pi(\text{"left"}|s) = 0.2$ ,
  - $\pi(\text{"right"}|s) = 0.1$ ,
  - $\pi(\text{"up"}|s) = 0.7$ .

State transition  $p(s'|s, a)$  can be random.

- Given state  $s$  and action  $a$ , the environment randomly generates a new state  $s'$ .

# Play the game using AI



- Observe a frame (state  $s_0$ )
- $\rightarrow$  Make action  $a_0$  (left, right, or up)
- $\rightarrow$  Observe a new frame (state  $s_1$ ) and reward  $r_0$
- $\rightarrow$  Make action  $a_1$
- $\rightarrow$  ...
- (state, action, reward) trajectory:  
 $s_0, a_0, r_0, s_1, a_1, r_1, \dots, s_T, a_T, r_T.$

# Rewards and Returns



# Return

**Definition:** Return (aka cumulative future reward).

- $R_t = r_t + r_{t+1} + r_{t+2} + r_{t+3} + \dots$  (to infinity.)

# Return

**Definition:** Return (aka cumulative future reward).

- $R_t = r_t + r_{t+1} + r_{t+2} + r_{t+3} + \dots$  (to infinity.)

**Question:** Are  $r_t$  and  $r_{t+1}$  equally important?

- Which of the followings do you prefer?
  - I give you \$100 right now.
  - I will give you \$100 one year later.

# Return

**Definition:** Return (aka cumulative future reward).

- $R_t = r_t + r_{t+1} + r_{t+2} + r_{t+3} + \dots$  (to infinity.)

**Question:** Are  $r_t$  and  $r_{t+1}$  equally important?

- Which of the followings do you prefer?
  - I give you \$100 right now.
  - I will give you \$100 one year later.
- Future value is less valuable than present value.
- $r_{t+1}$  should be given less weight than  $r_t$ .

# Discounted Return

**Definition:** Discounted return (aka cumulative discounted future reward).

- $\gamma$ : discount rate (tuning hyper-parameter).
- $R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \dots$  (to infinity.)



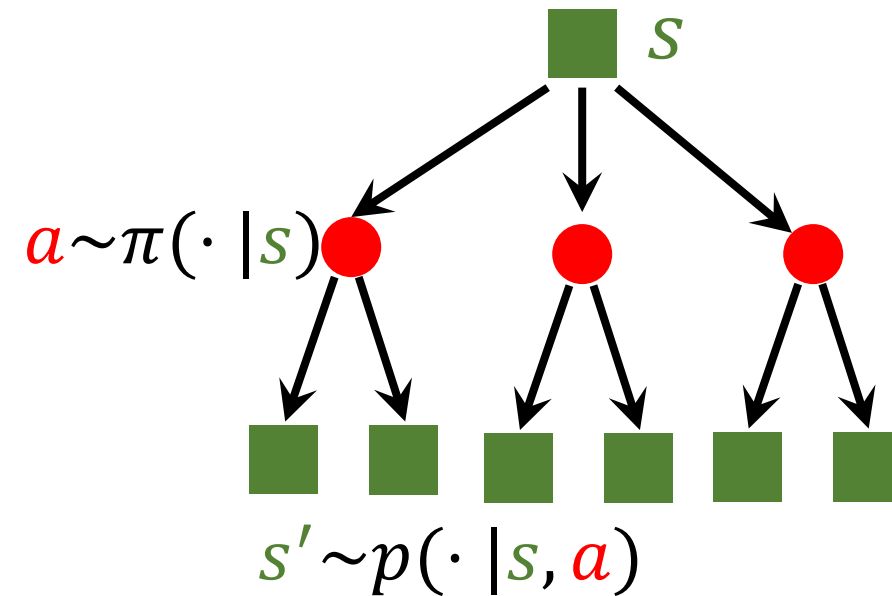
# Discounted Return

**Definition:** Discounted return (aka cumulative discounted future reward).

- $\gamma$ : discount rate (tuning hyper-parameter).
- $R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \dots$  (to infinity.)

$R_t$  can be random. (Two sources of randomness.)

1. Action can be random:  $a \sim \pi(\cdot | s)$ , where  
$$\pi(a | s) = \mathbb{P}(A = a | S = s).$$
2. State transition can be random:  $s' \sim p(\cdot | s, a)$ ,  
where  
$$p(s' | s, a) = \mathbb{P}(S' = s' | S = s, A = a).$$



# Value Functions

# Action-Value Function $Q(s, a)$

**Definition:** Discounted return (aka cumulative discounted future reward).

- $R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \dots$  (to infinity.)

**Definition:** Action-value function for policy  $\pi$ .

- $Q_\pi(s_t, a_t) = \mathbb{E} [R_t | s_t, a_t, \pi].$



- Taken w.r.t. actions  $a_{t+1}, a_{t+2}, a_{t+3}, \dots$  and states  $s_{t+1}, s_{t+2}, s_{t+3}, \dots$
- Actions are randomly sampled:  $a_t \sim \pi(\cdot | s_t)$ . (Policy function.)
- States are randomly sampled:  $s_{t+1} \sim p(\cdot | s_t, a_t)$ . (State transition.)

# Action-Value Function $Q(s, a)$

**Definition:** Discounted return (aka cumulative discounted future reward).

- $R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \dots$  (to infinity.)

**Definition:** Action-value function for policy  $\pi$ .

- $Q_\pi(s_t, a_t) = \mathbb{E} [R_t | s_t, a_t, \pi]$ .

**Definition:** Optimal action-value function.

- $Q^*(s_t, a_t) = \max_{\pi} Q_\pi(s_t, a_t)$ .



# State-Value Function $V(s)$

**Definition:** Discounted return (aka cumulative discounted future reward).

- $R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \dots$  (to infinity.)

**Definition:** Action-value function for policy  $\pi$ .

- $Q_\pi(s_t, a_t) = \mathbb{E} [R_t | s_t, a_t, \pi]$ .

**Definition:** State-value function.

- $V_\pi(s_t) = \mathbb{E}_{a_t} [Q_\pi(s_t, a_t)]$ .



- Taken w.r.t. the action  $a_t \sim \pi(\cdot | s_t)$ .

# State-Value Function $V(s, \pi)$

**Definition:** Discounted return (aka cumulative discounted future reward).

- $R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \dots$  (to infinity.)

**Definition:** Action-value function for policy  $\pi$ .

- $Q_\pi(s_t, a_t) = \mathbb{E} [R_t | s_t, a_t, \pi]$ .

**Definition:** State-value function.

- $V_\pi(s_t) = \mathbb{E}_{a_t} [Q_\pi(s_t, a_t)] = \sum_a \pi(a | s_t) \cdot Q_\pi(s_t, a)$ .

# Explaining Value Functions

- Optimal **action**-value function:  $Q^*(s_t, a_t) = \max_{\pi} Q_{\pi}(s_t, a_t)$ .
- How good it is for an agent to pick action  $a_t$  while being in state  $s_t$ .
- **State**-value function:  $V_{\pi}(s_t) = \mathbb{E} [Q_{\pi}(s_t, a_t)]$
- For policy  $\pi$ , how good the situation is in state  $s_t$ .

# Goals

**Goal of agent:** Make actions  $a_1, a_2, a_3, \dots$  to maximize the return  $R_t$ .

**Goal of reinforcement learning:** Guide the agent to make good actions.

Suppose we have a good policy  $\pi(a|s)$ .

- Upon observe the state  $s_t$ ,
- random sampling:  $a_t \sim \pi(\cdot | s_t)$ .

# Goals

**Goal of agent:** Make actions  $a_1, a_2, a_3, \dots$  to maximize the return  $R_t$ .

**Goal of reinforcement learning:** Guide the agent to make good actions.

Suppose we have a good policy  $\pi(a|s)$ .

Alternatively, suppose we know the optimal value function  $Q^*(s, a)$ .

- Upon observe the state  $s_t$ ,
- choose the action that maximizes the value:  $a_t = \operatorname{argmax}_a Q^*(s_t, a)$ .



# Goals

**Goal of agent:** Make actions  $a_1, a_2, a_3, \dots$  to maximize the return  $R_t$ .

**Goal of reinforcement learning:** Guide the agent to make good actions.

Suppose we have a good policy  $\pi(a|s)$ .

Alternatively, suppose we know the optimal value function  $Q^*(s, a)$ .



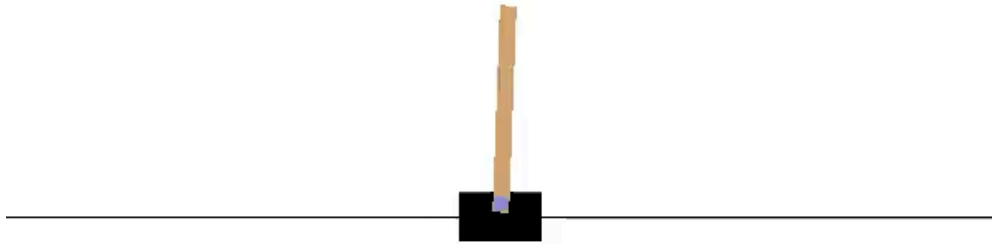
Learn either  $\pi(a|s)$  or  $Q^*(s, a)$  from the rewards:  $r_1, r_2, r_3, \dots$

**Play games using reinforcement learning**

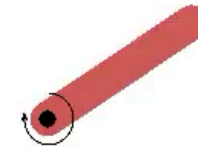
# OpenAI Gym

- Gym is a toolkit for developing and comparing reinforcement learning algorithms.
- <https://gym.openai.com/>

## Classical control problems



Cart Pole

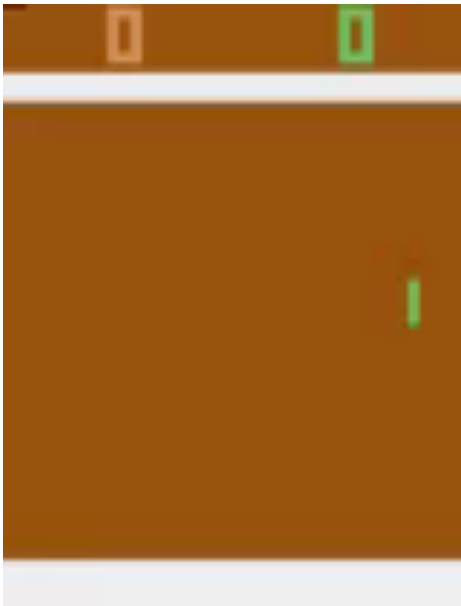


Pendulum

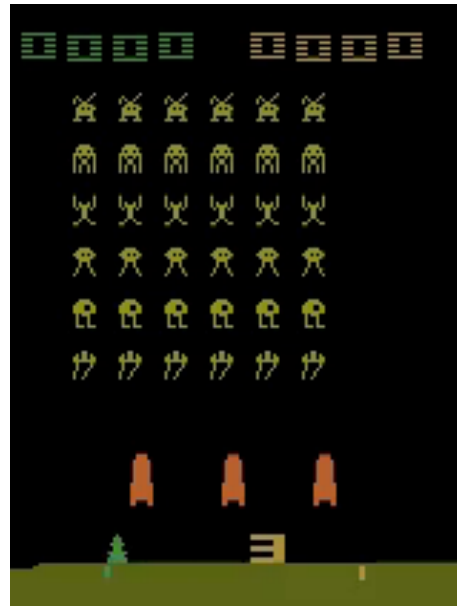
# OpenAI Gym

- Gym is a toolkit for developing and comparing reinforcement learning algorithms.
- <https://gym.openai.com/>

## Atari Games



Pong



Space Invader

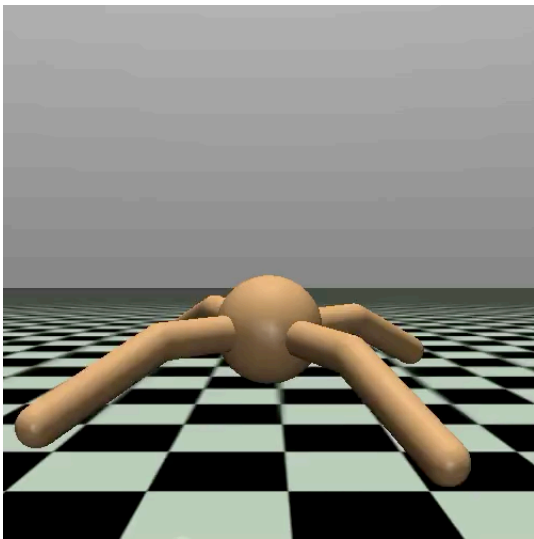


Breakout

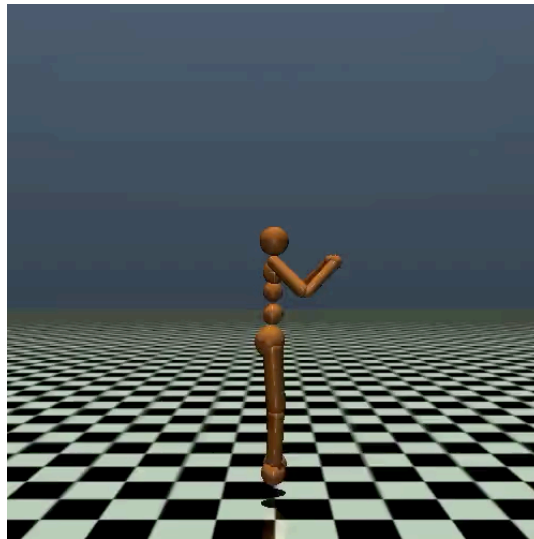
# OpenAI Gym

- Gym is a toolkit for developing and comparing reinforcement learning algorithms.
- <https://gym.openai.com/>

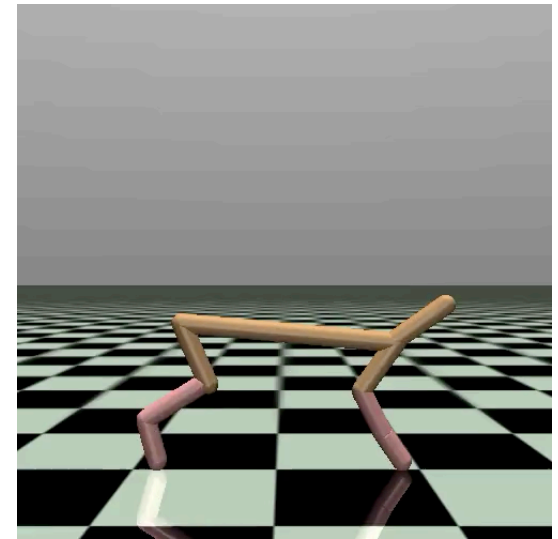
## MuJoCo (Continuous control tasks.)



Ant



Humanoid

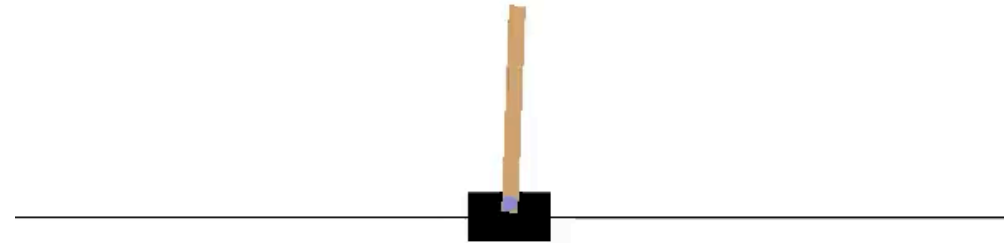


Half Cheetah

# Play CartPole Game

```
import gym  
env = gym.make( 'CartPole-v0' )
```

- Get the environment of CartPole from Gym.
- “env” provides states and reward.



# OpenAI Gym

```
state = env.reset()
```

```
for t in range(100):  
    env.render()  
    print(state)
```

A window pops up rendering CartPole.

A random **action**.

```
action = env.action_space.sample()  
state, reward, done, info = env.step(action)
```

```
if done:    "done=1" means finished (win or lose the game)  
    print('Finished')  
    break
```

```
env.close()
```



# Summary

# Summary

## Terminologies

- Agent



- Environment
- State  $s$ .
- Action  $a$ .
- Reward  $r$ .
- Policy  $\pi(a|s)$
- State transition  $p(s'|s, a)$ .

## Return and Value

- Return:

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots$$

- Action-value function:

$$Q_\pi(s, a) = \mathbb{E} [R_t | s, a, \pi].$$

- Optimal action-value function:

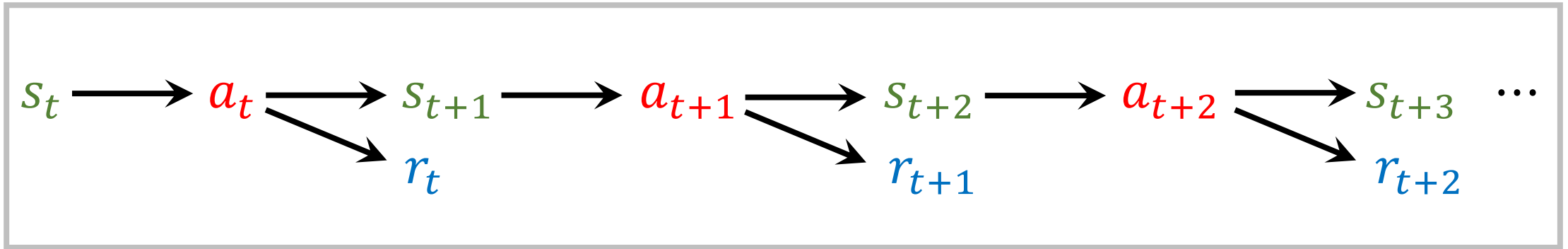
$$Q^*(s, a) = \max_{\pi} Q_\pi(s, a).$$

- State-value function:

$$V_\pi(s) = \mathbb{E}_a [Q_\pi(s, a)].$$

# Play game using reinforcement learning

- Observe state  $s_t$ , make action  $a_t$ , environment gives  $s_{t+1}$  and reward  $r_t$ .



- Suppose we know either policy function  $\pi(a|s)$  or the optimal action-value function  $Q^*(s, a)$ .
- Then action  $a$  can be made according to  $\pi(a|s)$  or  $Q^*(s, a)$ .

# We are going to study...

- Deep Q network (DQN) for approximating  $Q^*(s, a)$ .
- Learn the network parameters using temporal different (TD).
- Policy network for approximating  $\pi(a|s)$ .
- Learn the network parameters using policy gradient.
- Actor-critic method. (Policy network + value network.)