

# CS 277: Control and Reinforcement Learning

Winter 2022

## Lecture 3: Temporal-Difference Methods

Roy Fox

Department of Computer Science

Bren School of Information and Computer Sciences

University of California, Irvine



# Logistics

---

assignments

- Assignments 1 to be published very soon
- Due **next Tuesday**

resources

- Lots of resources on the website
- Will be updated with papers relevant to each lecture

# Today's lecture

---

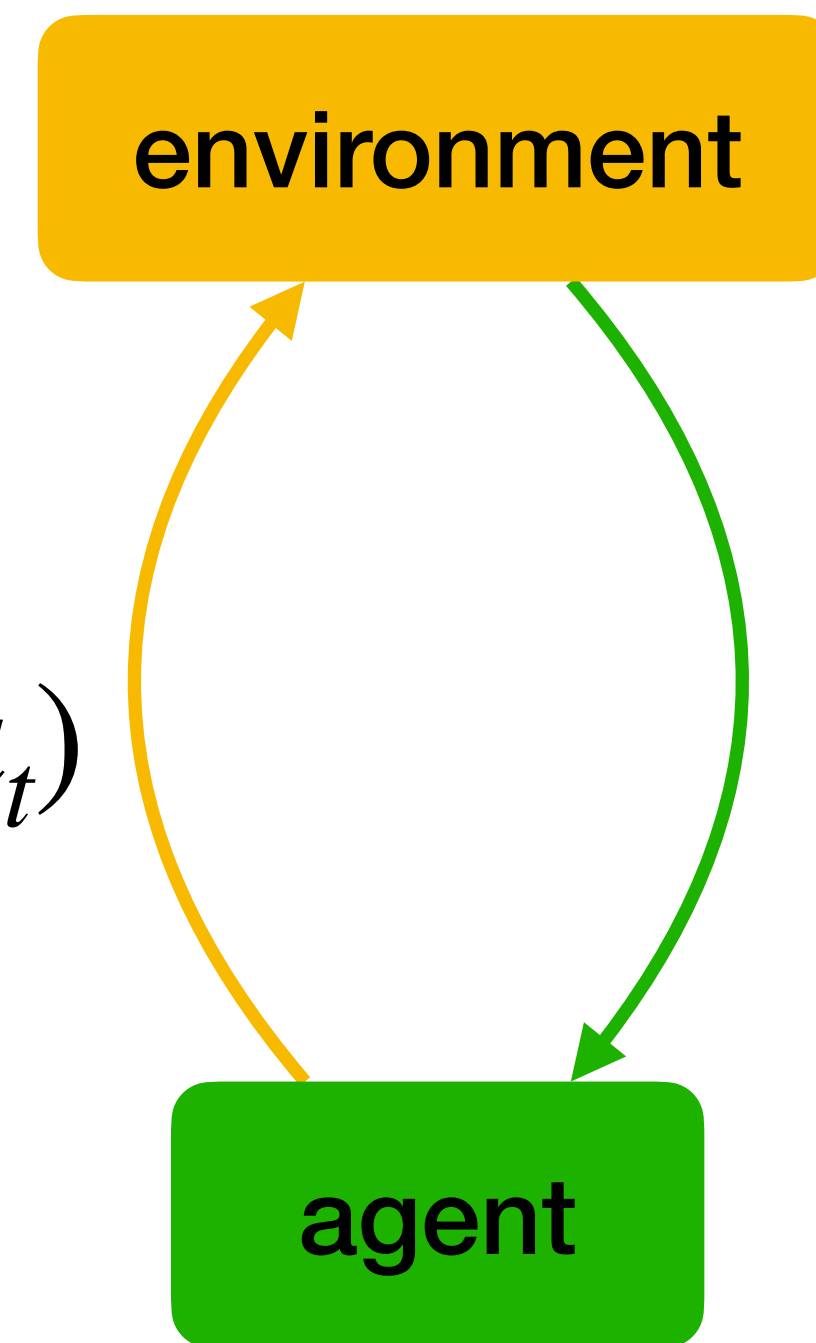
**Policy evaluation**

**Temporal Difference**

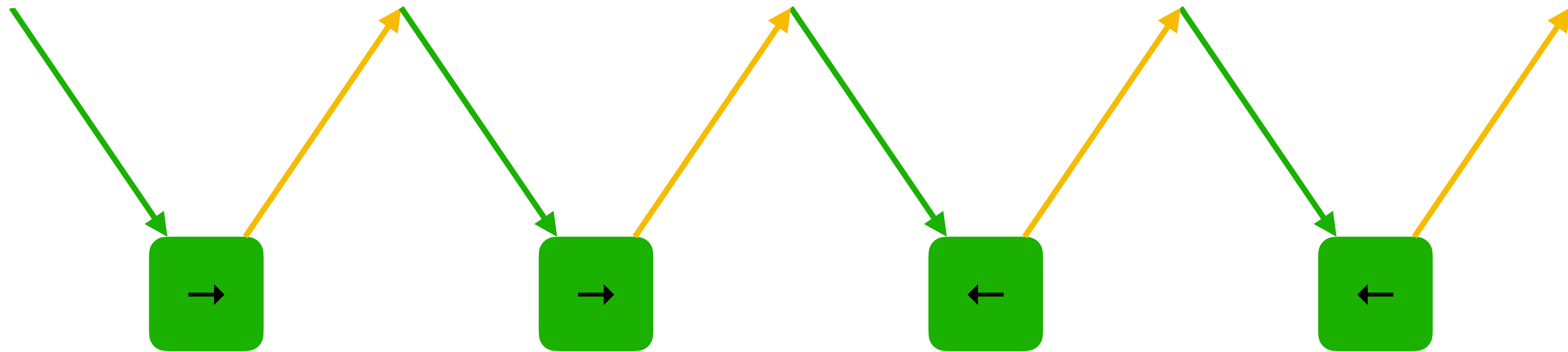
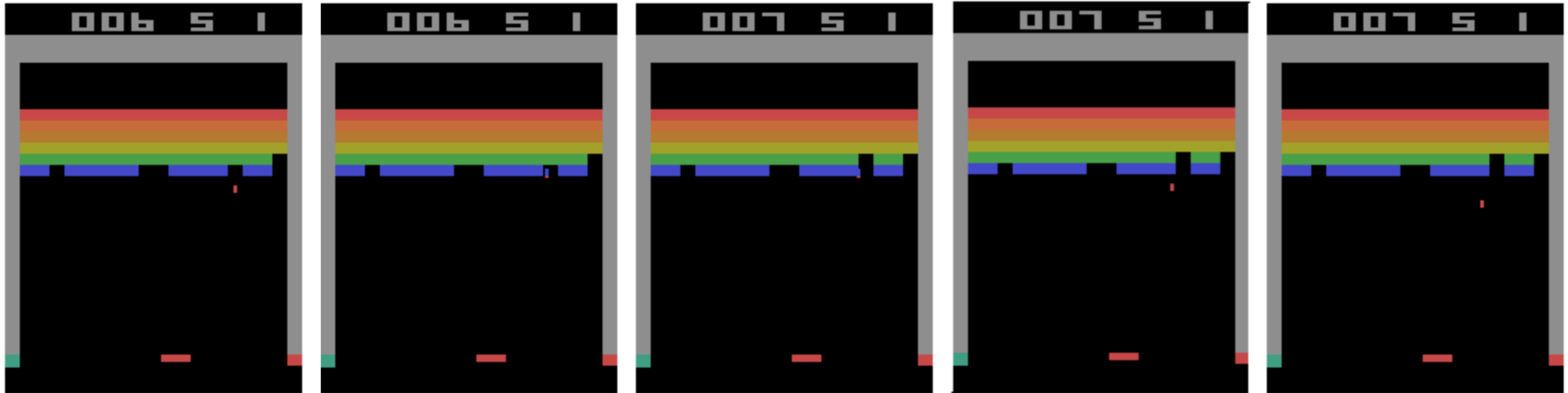
**Policy improvement**

# Basic RL concepts

- **State:**  $s \in \mathcal{S}$ ; **action:**  $a \in \mathcal{A}$ ; **reward:**  $r(s, a) \in \mathbb{R}$
- **Dynamics:**  $p(s_{t+1} | s_t, a_t)$  for stochastic;  $s_{t+1} = f(s_t, a_t)$  for deterministic
- **MDP:**  $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, p \rangle$  or  $\langle \mathcal{S}, \mathcal{A}, p, r \rangle$
- **Policy:**  $\pi(a_t | s_t)$  for stochastic;  $a_t = \pi(s_t)$  for deterministic
- **Trajectory:**  $p_\pi(\xi = s_0, a_0, s_1, a_1, \dots) = p(s_0) \prod_{t \geq 0} \pi(a_t | s_t) p(s_{t+1} | s_t, a_t)$
- **Return:**  $R(\xi) = \sum_{t \geq 0} \gamma^t r(s_t, a_t) \quad 0 \leq \gamma < 1$



# Example: Breakout



reward:

+0

+1

+0

+0

# Formulating reward: considerations

- We define  $r(s, a)$ , is that general enough?
- What if the reward depends on the **next state**  $s'$ ?
  - ▶ If we only care about **expected** reward, define  $r(s, a) = \mathbb{E}_{(s'|s,a) \sim p}[r(s, a, s')]$
- What if the reward is a **random** variable  $\tilde{r}$ ?
  - ▶ Define  $r(s, a) = \mathbb{E}[\tilde{r} | s, a]$
  - ▶ In practice we see  $\tilde{r} \Rightarrow$  don't just assume you know  $r(s, a) = \tilde{r}$

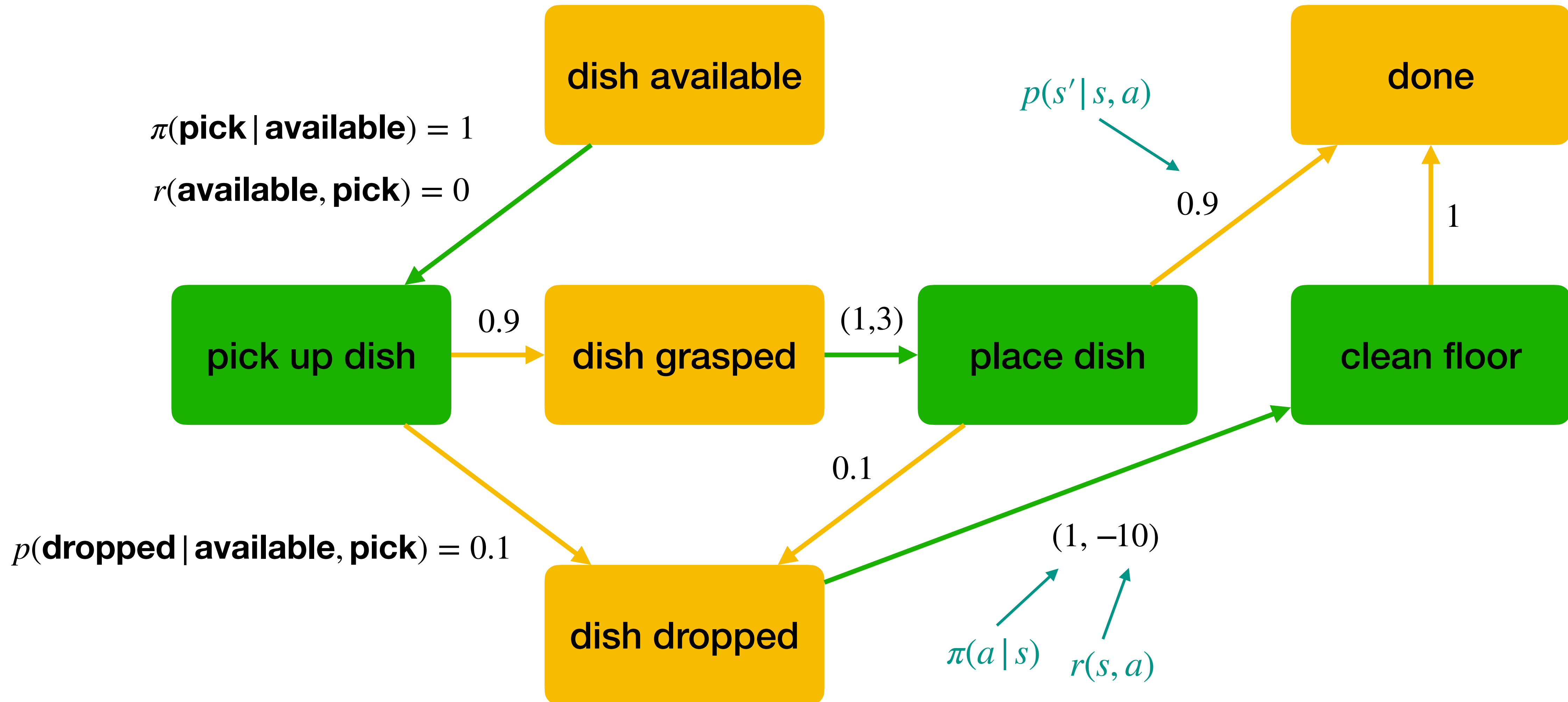


# RL objective: expected return

- We need a **scalar** to optimize
- **Step 1:** we have a whole sequence of rewards  $\{r_t = r(s_t, a_t)\}_{t \geq 0}$ 
  - **Summarize** as return  $R(\xi) = \sum_{t \geq 0} \gamma^t r(s_t, a_t)$
- **Step 2:**  $R(\xi)$  is a random variable, induced by  $p_\pi(\xi)$ 
  - Take **expectation**  $J_\pi = \mathbb{E}_{\xi \sim p_\pi}[R(\xi)]$
- $J_\pi$  can be calculated and optimized



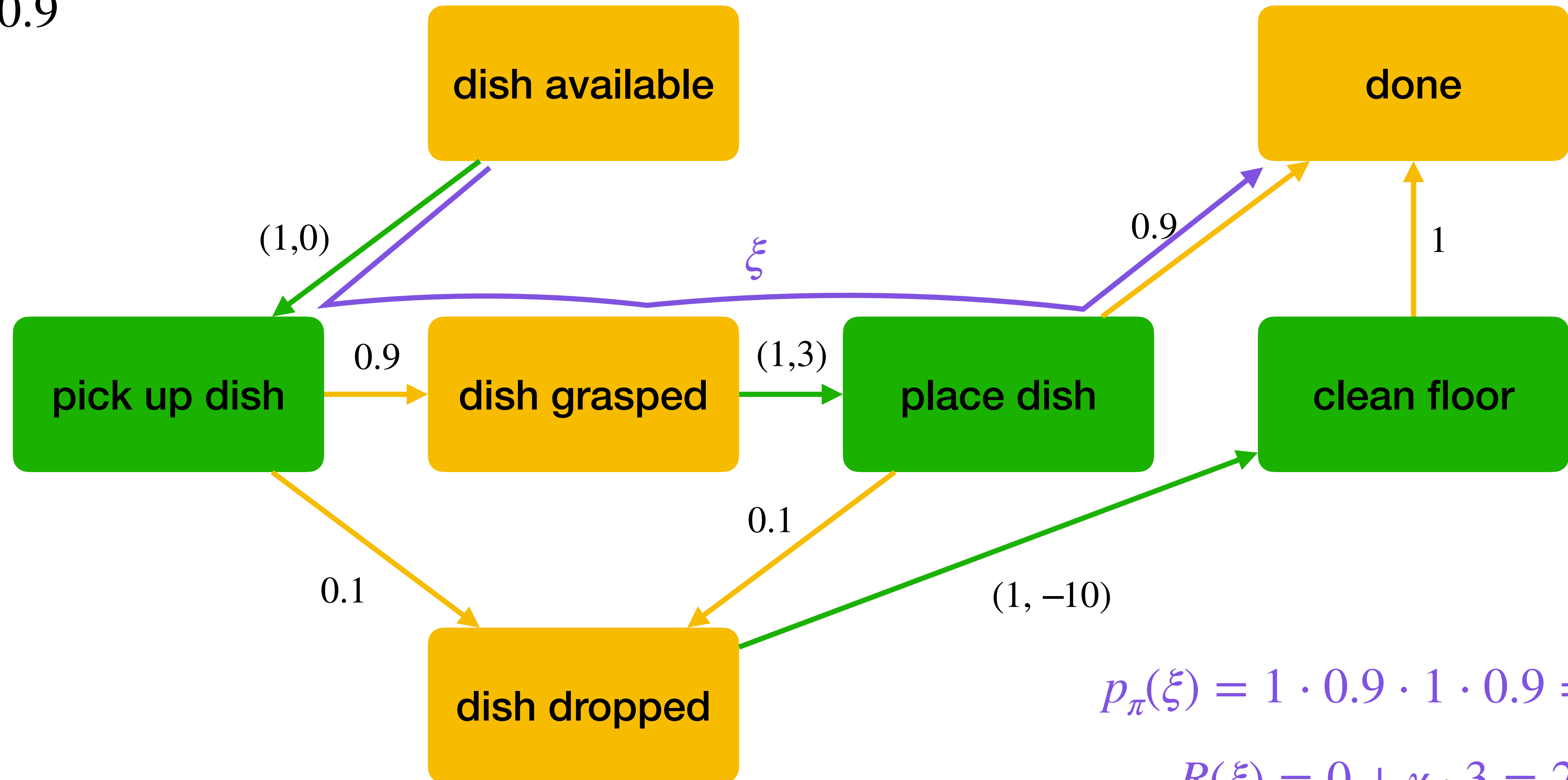
# Policy evaluation: example





# Policy evaluation: example

$\gamma = 0.9$

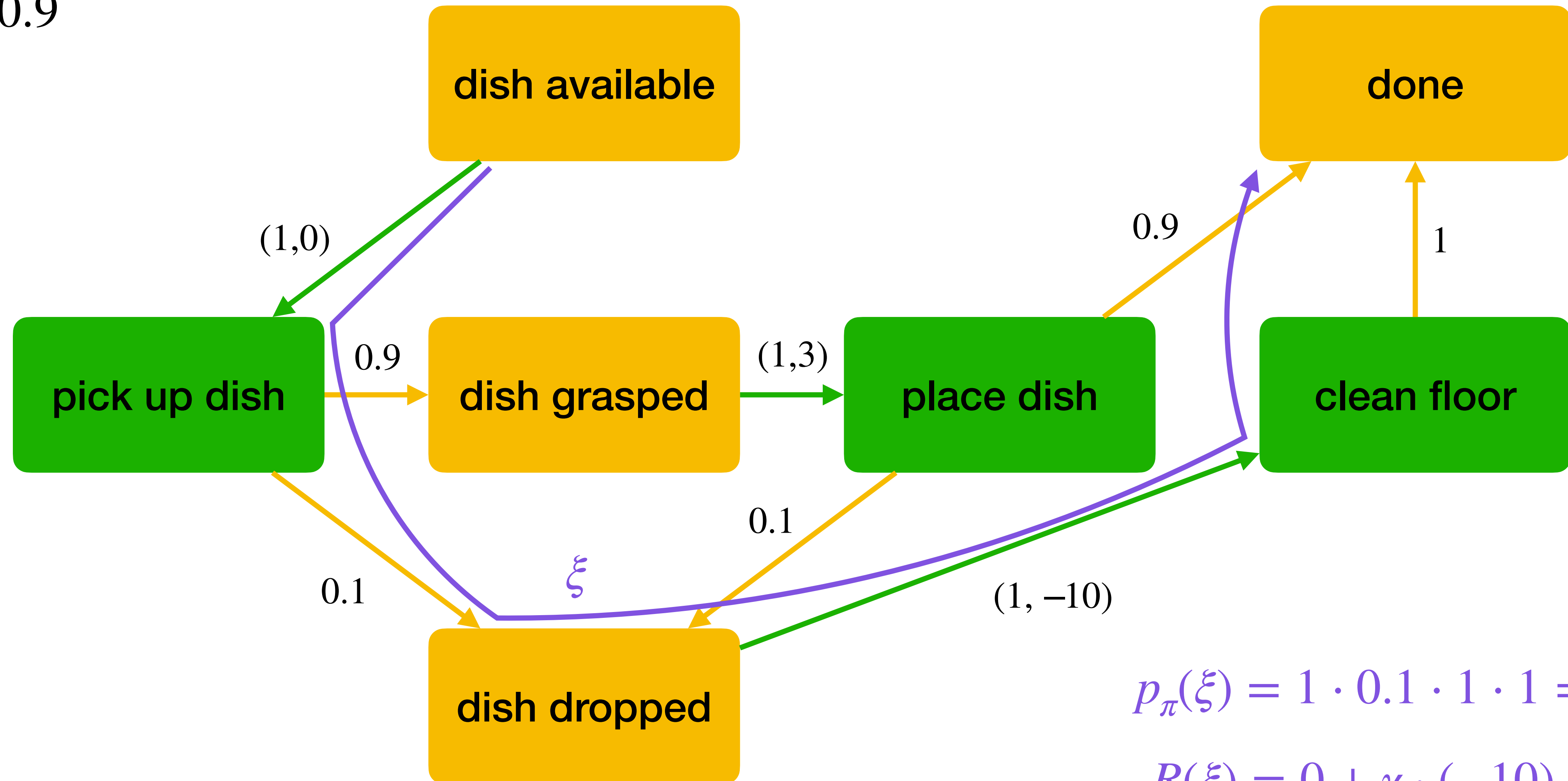


$$p_{\pi}(\xi) = 1 \cdot 0.9 \cdot 1 \cdot 0.9 = 0.81$$

$$R(\xi) = 0 + \gamma \cdot 3 = 2.7$$

# Policy evaluation: example

$\gamma = 0.9$



$$p_{\pi}(\xi) = 1 \cdot 0.1 \cdot 1 \cdot 1 = 0.1$$

$$R(\xi) = 0 + \gamma \cdot (-10) = -9$$

# Monte Carlo (MC) policy evaluation


MF

- Computing  $J_\pi = \mathbb{E}_{\xi \sim p_\pi} [R(\xi)] = \sum_{\xi} p_\pi(\xi) R(\xi)$  can be hard
  - ▶ **Exponentially many** trajectories
  - ▶ **Model-based** = requires  $p(s' | s, a)$ , which may not be known
- **Monte Carlo**: estimate expectation using empirical mean

$$J_\pi \approx \frac{1}{m} \sum_i R(\xi^{(i)}) \quad \xi^{(i)} \sim p_\pi$$

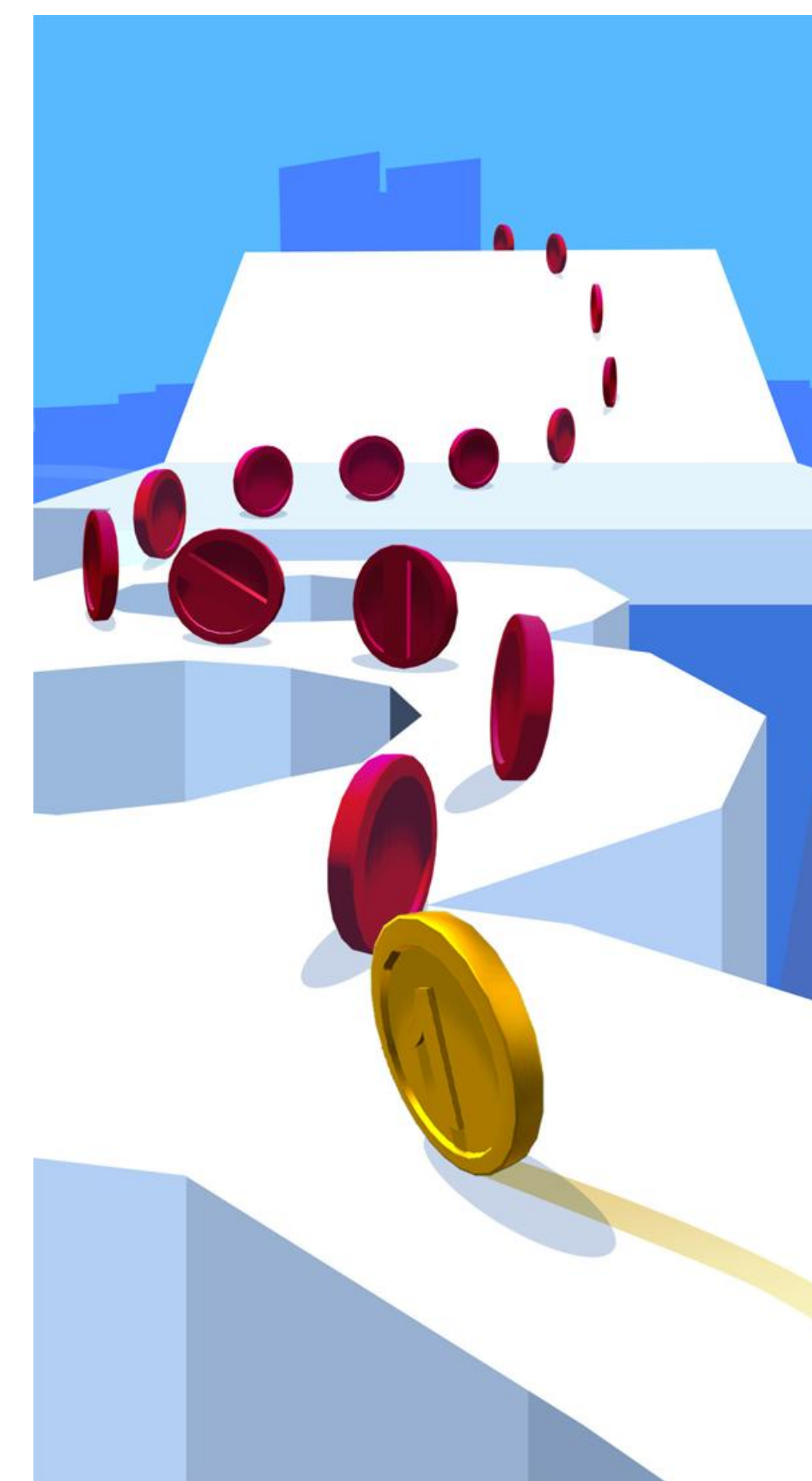
- ▶ **Model-free** = can sample with rollouts, without knowing  $p$

# MC: iterative computation

- We can keep a **running average**  $\bar{R}^{(i)}$  of the first  $i$  returns
  - ▶ **Update:**  $\bar{R}^{(i)} = ((i - 1)\bar{R}^{(i-1)} + R(\xi^{(i)}))\frac{1}{i}$
  - ▶ More generally:  $\bar{R}^{(i)} = (1 - \alpha)\bar{R}^{(i-1)} + \alpha R(\xi^{(i)}) = \bar{R}^{(i-1)} + \alpha(R(\xi^{(i)}) - \bar{R}^{(i-1)})$   

  - ▶  $\alpha$  is a **learning rate**, exact average when it vanishes harmonically as  $\frac{1}{i}$
- To simplify expressions, we denote this update:  $J \rightarrow_{\alpha} R(\xi)$ 
  - ▶ Read: **update**  $J$  toward  $R(\xi)$  at rate  $\alpha$

# Value function

- **RL objective:** maximize expected return  $J_\pi = \mathbb{E}_{\xi \sim p_\pi}[R]$
- We don't control  $s_0$ , can **break down:**  $J_\pi = \mathbb{E}_{s_0 \sim p}[V_\pi(s_0) | s_0]$ 
  - ▶ with the **value function**  $V_\pi(s) = \mathbb{E}_{\xi \sim p_\pi}[R | s_0 = s]$
- $V_\pi(s)$  is the expected **reward-to-go** (= future return):
  - ▶ For any  $t_0$ , define  $R_{\geq t_0} = \sum_{t \geq t_0} \gamma^{t-t_0} r(s_t, a_t)$ 
    - future reward after being in state  $s$  in time  $t_0$
  - ▶ Then  $V_\pi(s) = \mathbb{E}_{\xi \sim p_\pi}[R_{\geq t_0} | s_{t_0} = s]$



# MC for value-function estimation

MF

---

**Algorithm** MC for value-function estimation

---

Initialize  $V(s) \leftarrow 0$  for all  $s \in S$

**repeat**

    Sample  $\xi \sim p_\pi$

    Update  $V(s_0) \rightarrow R(\xi)$

---

- Why not use the same samples for **non-initial** states?

---

**Algorithm** MC for value-function estimation (version 2)

---

Initialize  $V(s) \leftarrow 0$  for all  $s \in S$

**repeat**

    Sample  $\xi \sim p_\pi$

    Update  $V(s_t) \rightarrow R_{\geq t}(\xi)$  for all  $t \geq 0$

---

# MC with function approximation

- What if the state space is **large**?
  - Can't represent  $V(s)$  as a **big table**
  - Won't have **enough data** to estimate each  $V(s)$
- **Function approximation**: represent  $V_\theta : S \rightarrow \mathbb{R}$ 
  - $\theta \in \Theta$ , a parametric family of functions; for example, a **neural network**
- **Generalization** over state space  $\Rightarrow$  data efficiency

---

**Algorithm** MC with function approximation

---

Initialize  $V_\theta$

**repeat**

Sample  $\xi \sim p_\pi$

Descend on  $\mathcal{L}_\theta = \sum_{t \geq 0} (R_{\geq t}(\xi) - V_\theta(s_t))^2$

---

MF

$\theta$

**with tabular representation:**

$$V(s_t) += -\alpha \nabla_{V(s_t)} \mathcal{L} = 2\alpha (R_{\geq t}(\xi) - V(s_t))$$

**same as in previous slide**

# Today's lecture

---

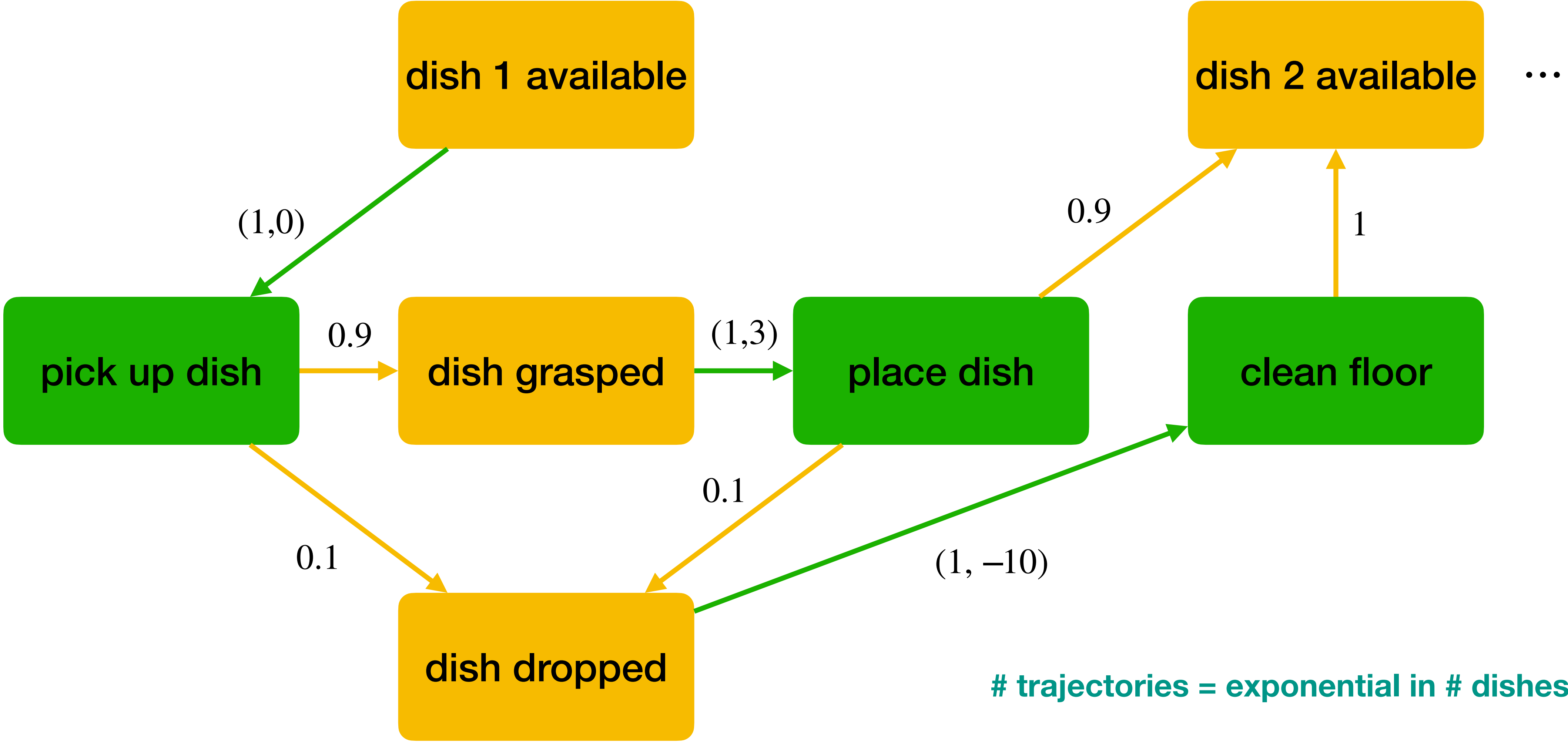
Policy evaluation

**Temporal Difference**

Policy improvement



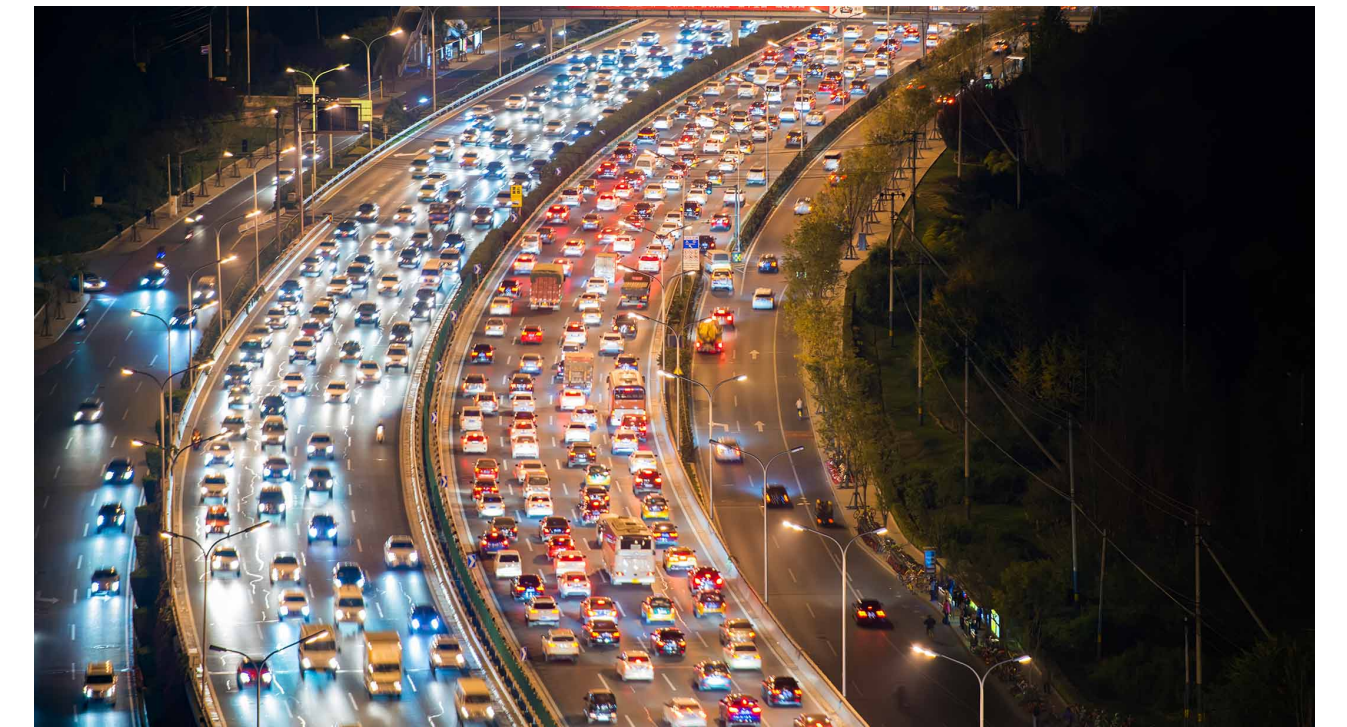
# Policy evaluation: example



# trajectories = exponential in # dishes

# MC inefficiency

- The MC estimator is **unbiased** (correct expectation), but **high variance**
  - Requires many samples to give good estimate
- But MC misses out on the **sequential structure**
- Example:
  - Day 1: I take **route 1** to work — **40 minutes**; I take **route 2** home — **10 minutes**
  - Day 2: I take **route 3** to work — **30 minutes**; I take **route 4** home — **30 minutes**
- Which route should I take to work?
  - Route 1 → 50-minute daily commute, route 3 → 60-minute; is **route 1 better?**



# Dynamic Programming (DP)

- **Dynamic Programming** = remember reusable partial results
- Value recursion:

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

break down sum of rewards

$$= \mathbb{E}_{\xi \sim p_{\pi}}[r(s_0, a_0) + \gamma R_{\geq 1} \mid s_0 = s]$$

first reward only depends on  $a$

$$= \mathbb{E}_{(a|s) \sim \pi}[r(s, a) + \gamma \mathbb{E}_{\xi \sim p_{\pi}}[R_{\geq 1} \mid s_0 = s, a_0 = a]]$$

$s'$  is a state, all that matters for  $R_{\geq 1}$

$$= \mathbb{E}_{(a|s) \sim \pi}[r(s, a) + \gamma \mathbb{E}_{(s'|s,a) \sim p}[\mathbb{E}_{\xi \sim p_{\pi}}[R_{\geq 1} \mid s_1 = s']]]$$

definition of  $V_{\pi}(s')$

$$= \mathbb{E}_{(a|s) \sim \pi}[r(s, a) + \gamma \mathbb{E}_{(s'|s,a) \sim p}[V_{\pi}(s')]]$$



Richard Bellman

MF

$\theta$

DP

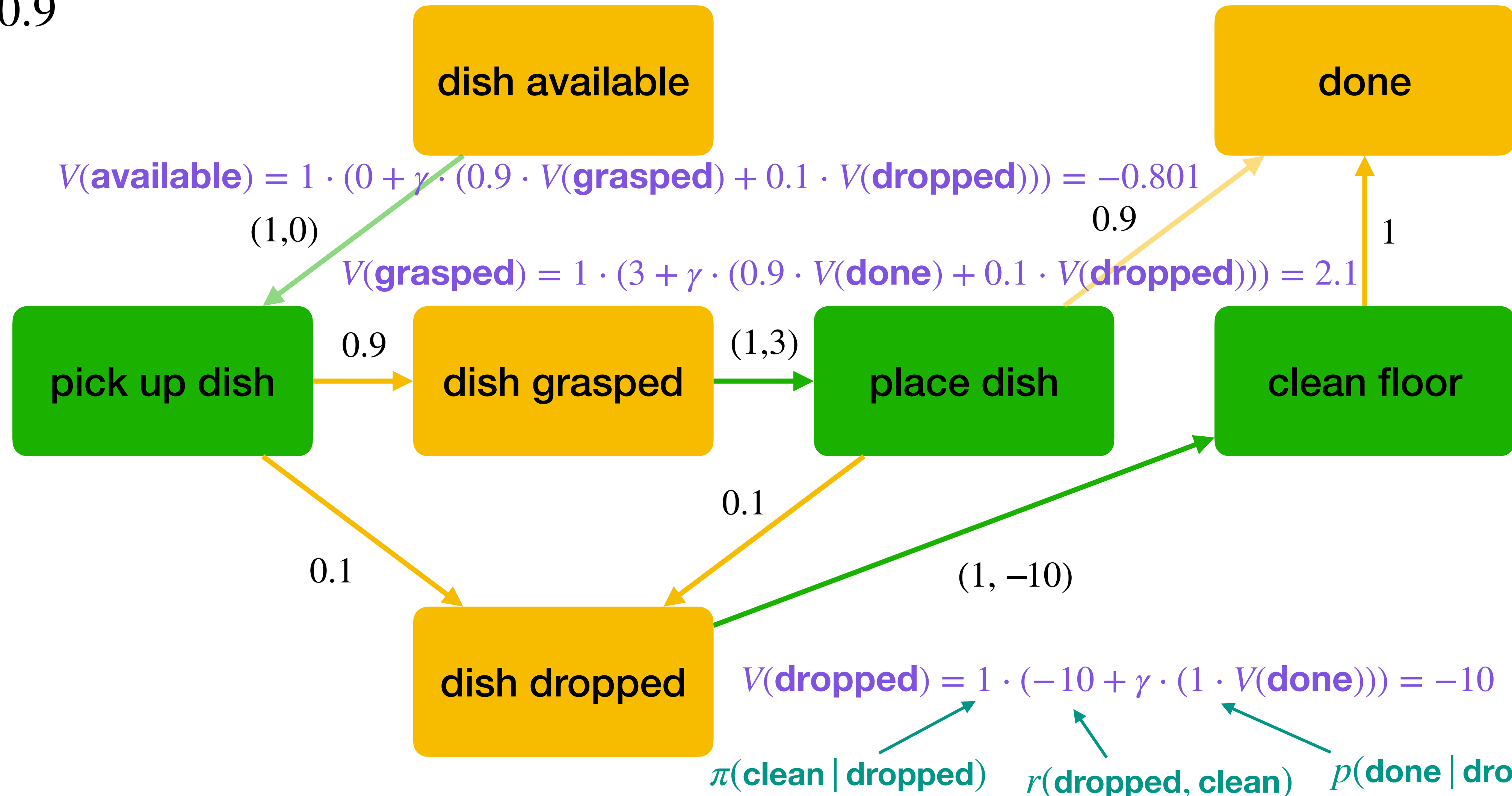
# Policy evaluation: example

$\gamma = 0.9$

$$V_{\pi}(s) = \mathbb{E}_{(a|s) \sim \pi} [r(s, a) + \gamma \mathbb{E}_{(s'|s, a) \sim p} [V_{\pi}(s')]]$$

$V(\text{done}) = 0$

- MF
- $\theta$
- DP



# DP + MC: Temporal Difference (TD)

- Policy evaluation with **DP**:  $V_{\pi}(s) = \mathbb{E}_{(a|s) \sim \pi}[r(s, a) + \gamma \mathbb{E}_{(s'|s, a) \sim p}[V_{\pi}(s')]]$

- ▶ **Drawback**: model-based = need to know  $p$

recursion from  $s'$  to  $s$   
= backward in time!

- **MC**:  $V(s) \rightarrow R_{\geq t}(\xi)$ , where  $\xi \sim p_{\pi}$  and  $s_t = s$

- ▶ **Drawback**: high variance

- **Put together**:  $V(s) \rightarrow r + \gamma V(s')$

- ▶ where  $s = s_t$ ,  $r = r(s_t, a_t)$ , and  $s' = s_{t+1}$  in some trajectory

temporal difference  
between  $V(s')$  and  $V(s)$

- ▶ In other words:  $V(s) \leftarrow V(s) + \alpha(r + \gamma V(s') - V(s))$

MF

$\theta$

DP

# Q function

- To approach  $V_\pi$  when we update  $V(s) \rightarrow r + \gamma V(s')$ , we need **on-policy data**
  - Roll out  $\pi$  to see transition  $(s, a) \rightarrow s'$  with reward  $r$
- On-policy data is **expensive**: need more every time  $\pi$  changes
- **Action-value function**:  $Q_\pi(s, a) = \mathbb{E}_{\xi \sim p_\pi}[R \mid s_0 = s, a_0 = a]$ 
  - Compare:  $V_\pi(s) = \mathbb{E}_{\xi \sim p_\pi}[R \mid s_0 = s] = \mathbb{E}_{(a|s) \sim \pi}[Q_\pi(s, a)]$
- Action-value **backward recursion**:  $Q_\pi(s, a) = r(s, a) + \gamma \mathbb{E}_{(s'|s,a) \sim p}[V_\pi(s')]$ 
  - Broke down  $V_\pi(s) = \mathbb{E}_{(a|s) \sim \pi}[r(s, a) + \gamma \mathbb{E}_{(s'|s,a) \sim p}[V_\pi(s')]]$  into two parts



MF

$\theta$

DP

# TD from off-policy data

- Backward recursion in two parts:

$$V_{\pi}(s) = \mathbb{E}_{(a|s) \sim \pi}[Q_{\pi}(s, a)] \quad Q_{\pi}(s, a) = r(s, a) + \gamma \mathbb{E}_{(s'|s, a) \sim p}[V_{\pi}(s')]$$

- This should hold in every state and action

▸  $(s, a)$  can be sampled from **any distribution**  $p_{\pi'}$  for any alternative  $\pi'$

- Put together, we **update**  $Q(s, a) \rightarrow r + \gamma \mathbb{E}_{(a'|s') \sim \pi}[Q(s', a')]$

▸ For any distribution of  $(s, a)$ , giving reward  $r$  and following state  $s' \sim p(\cdot | s, a)$

**temporal difference**

▸ In other words:  $Q(s, a) \leftarrow Q(s, a) + \alpha(r + \gamma \mathbb{E}_{(a'|s') \sim \pi}[Q(s', a')] - Q(s, a))$

MF

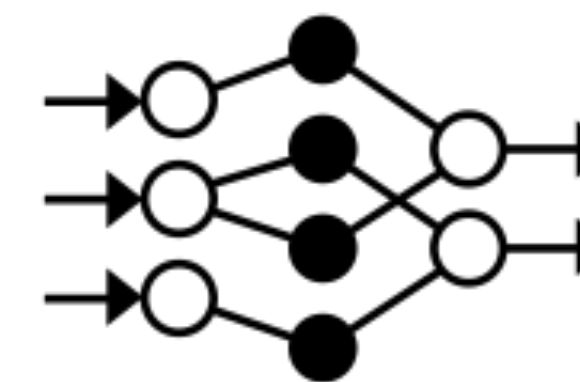
$\theta$

DP

$\pi'$

# TD with function approximation

- With large state space: **represent**  $V_\theta : S \rightarrow \mathbb{R}$  or  $Q_\theta : S \times A \rightarrow \mathbb{R}$



- Instead of the update  $V(s) \rightarrow r + \gamma V(s')$

- ▶ Descend on **square loss**  $\mathcal{L}_\theta = (r + \gamma V_{\bar{\theta}}(s') - V_\theta(s))^2$

- ▶ On **on-policy** experience  $(s, a, r, s')$

**only learn  $V_\theta(s)$**   
 **$V_{\bar{\theta}}(s')$  is the target**  
 **$\Rightarrow$  don't take its gradient!**

- Instead of the update  $Q(s, a) \rightarrow r + \gamma \mathbb{E}_{(a'|s') \sim \pi} [Q(s', a')]$

- ▶ Descend on **square loss**  $\mathcal{L}_\theta = (r + \gamma \mathbb{E}_{(a'|s') \sim \pi} [Q_{\bar{\theta}}(s', a')] - Q_\theta(s, a))^2$

- ▶ On **off-policy** experience  $(s, a, r, s')$

**only learn  $Q_\theta(s, a)$**   
 **$Q_{\bar{\theta}}(s', a')$  is the target**  
 **$\Rightarrow$  don't take its gradient!**

MF  
 $\theta$   
DP  
 $\pi'$

MF  
 $\theta$   
DP  
 $\pi'$



# Today's lecture

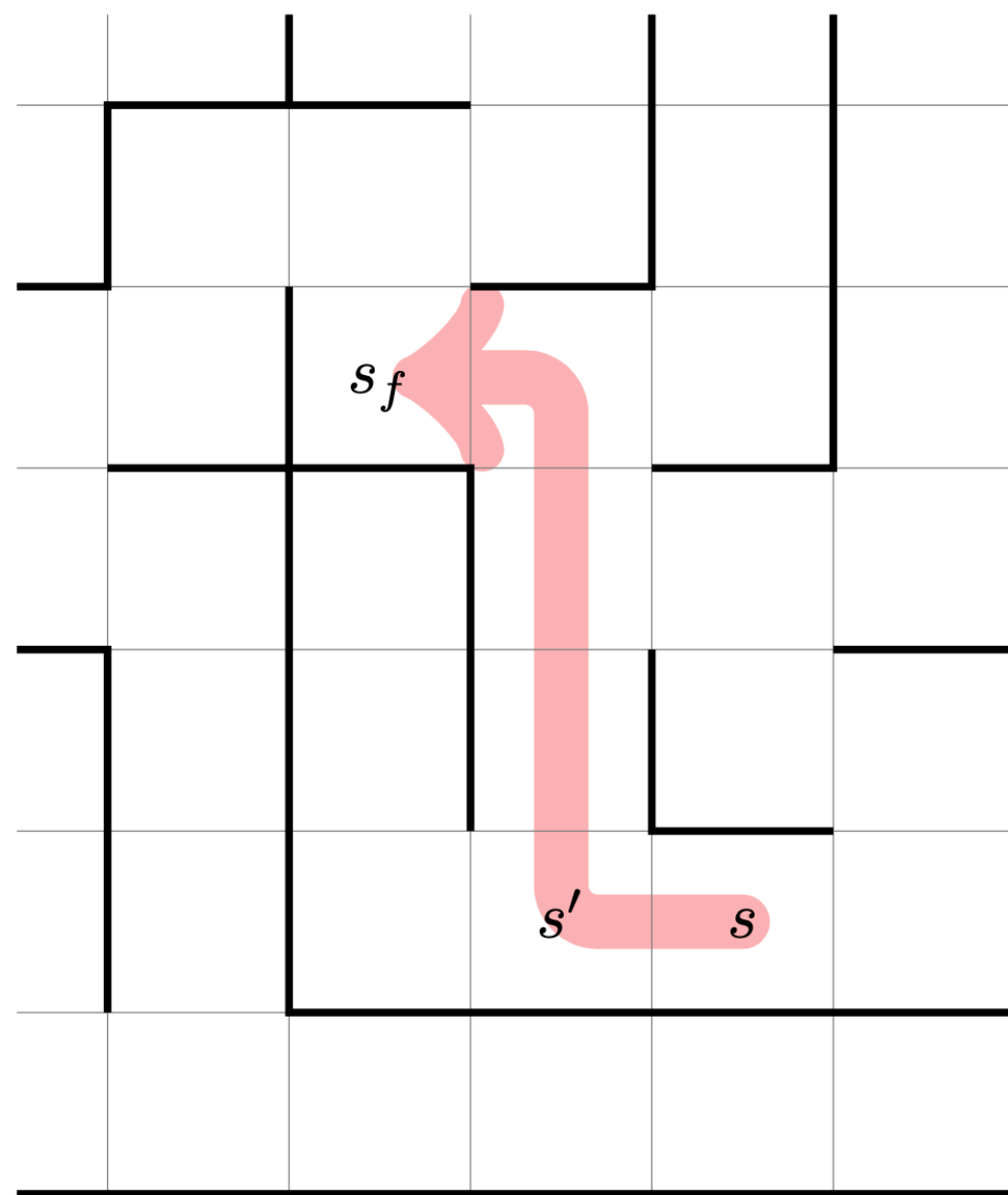
---

Policy evaluation

Temporal Difference

**Policy improvement**

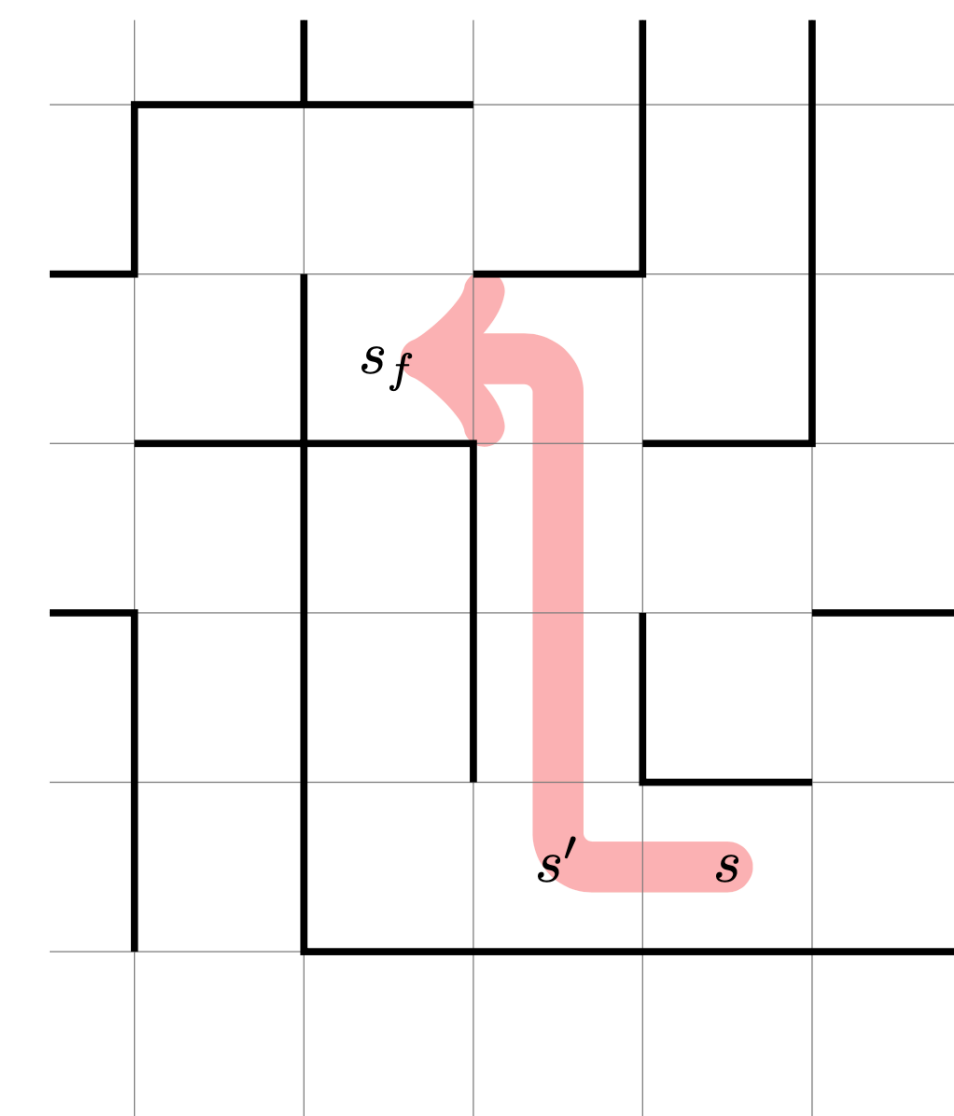
# Special case: shortest path



- **Deterministic dynamics:** in state  $s$ , take action  $a$  to get to state  $s' = f(s, a)$ 
  - Example above:  $s' = f(s, a_{\text{left}})$
- **Reward:**  $(-1)$  in each step (until the goal  $s_f$  is reached)

# Shortest path: optimality principle

- **Proposition:**  $\xi$  is shortest from  $s$  to  $s_f$  through  $s'$   $\Rightarrow$  suffix of  $\xi$  is shortest from  $s'$  to  $s_f$
- **Proof:** otherwise, let  $\xi'$  be a shorter path from  $s'$  to  $s_f$ , then take  $s \xrightarrow{\xi} s' \xrightarrow{\xi'} s_f$
- The proposition is “if” but not “only if”, because we don't know **which  $s'$**  is best
  - **Try them all:** for each  $a$ , try  $s' = f(s, a)$
- Let  $V(s)$  be the shortest path length from  $s$  to  $s_f$ 
  - For each candidate  $s'$ , the **shortest path** through it is  $1 + V(s')$
  - For all  $s \neq s_f$ , we have  $V(s) = \min_a (1 + V(f(s, a)))$



# Bellman-Ford shortest path algorithm

- For all  $s \neq s_f$ , we have  $V(s) = \min_a (1 + V(f(s, a)))$

---

## Algorithm Bellman-Ford

---

$$V(s_f) \leftarrow 0$$

$V(s) \leftarrow \infty$  for each non-terminal state  $s$

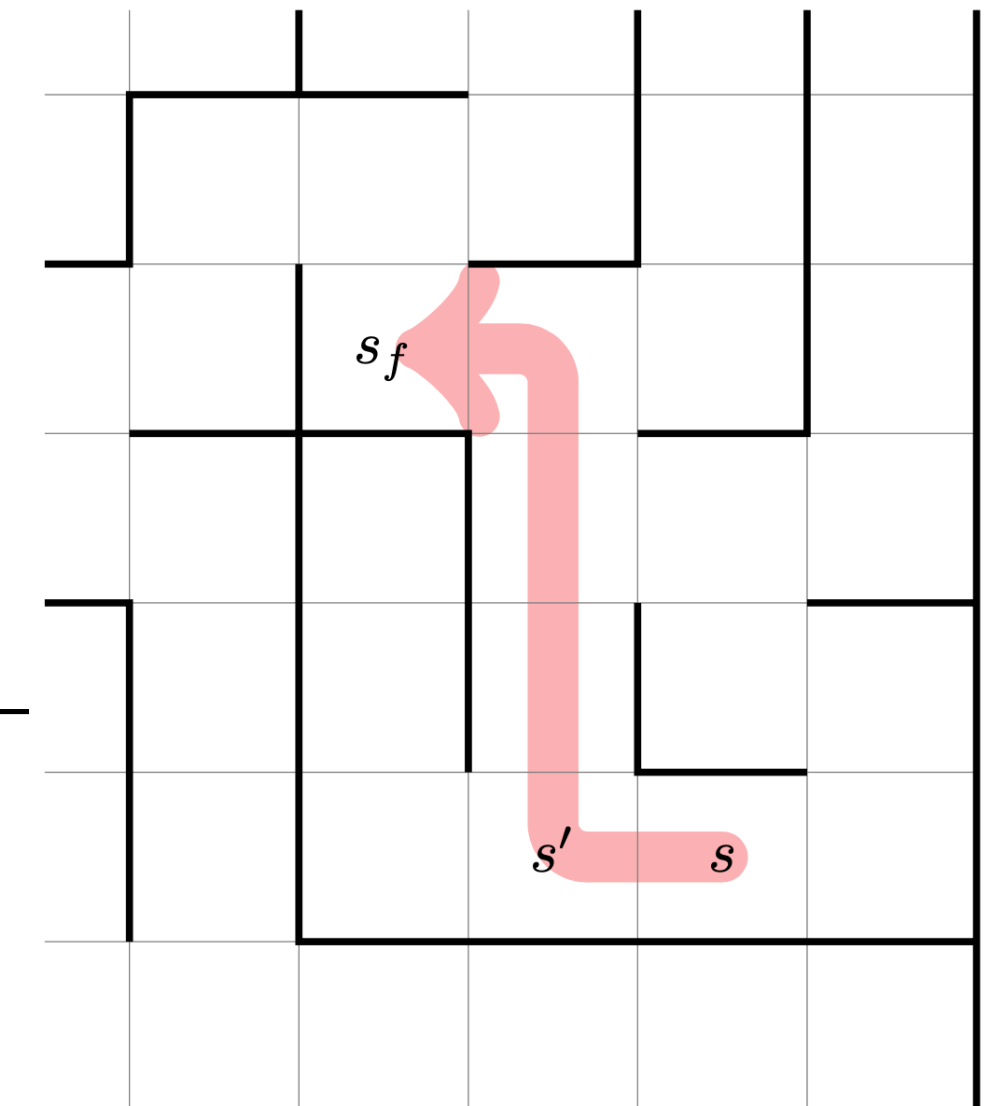
**for**  $|S| - 1$  iterations

**for each** non-terminal state  $s$

$$V(s) \leftarrow \min_{a \in A} (1 + V(f(s, a)))$$

---

- The **optimal policy** is  $\pi(s) = \arg \min_a (1 + V(f(s, a)))$



MF

$\theta$

DP

$\pi'$

max

# Policy improvement

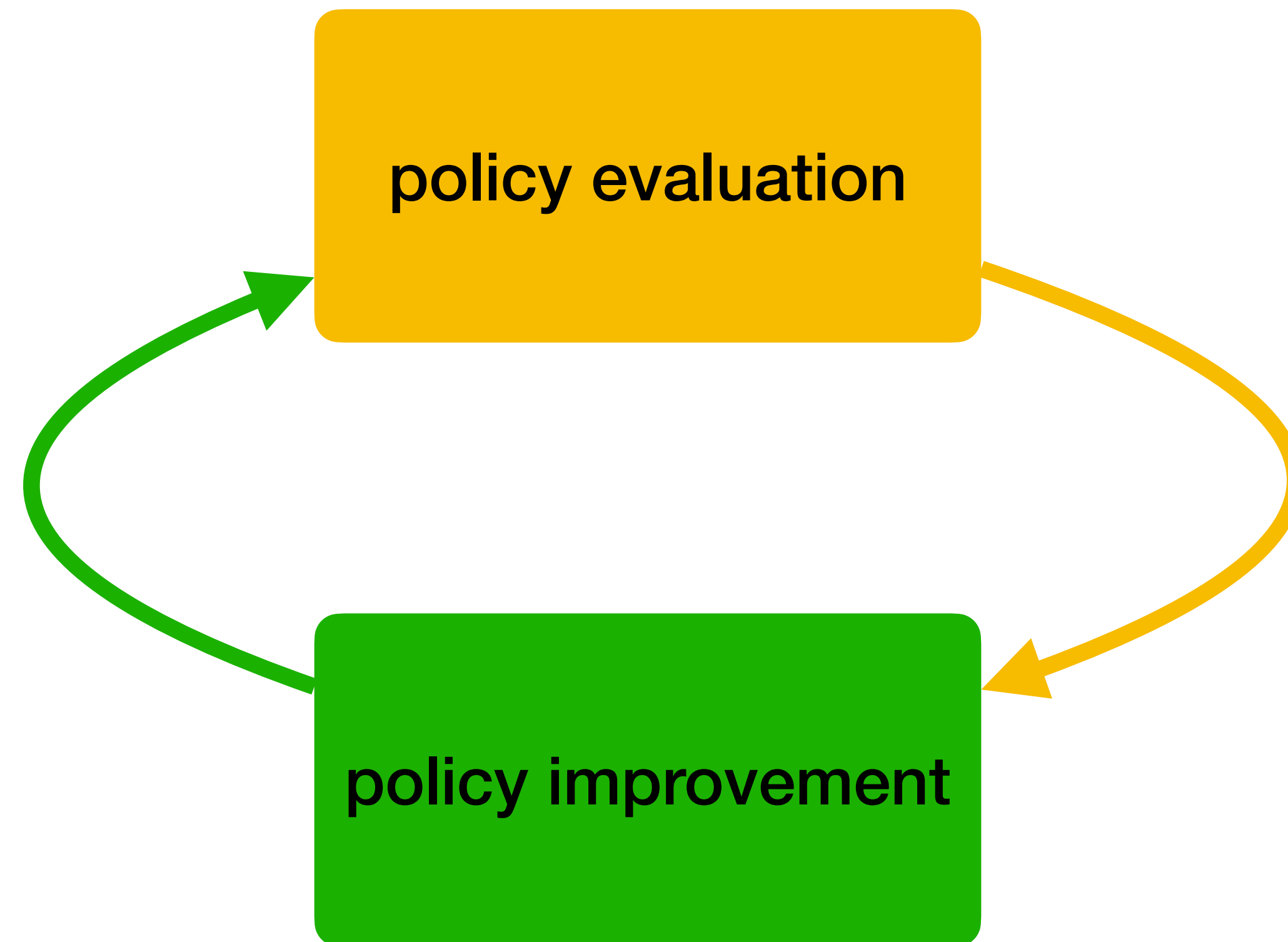
- A value function suggests the **greedy policy**:

$$\pi(s) = \arg \max_a Q(s, a) = \arg \max_a (r(s, a) + \gamma \mathbb{E}_{(s'|s,a) \sim p} [V(s')])$$

- The greedy policy **may not be the optimal policy**  $\pi^* = \arg \max_{\pi} J_{\pi}$ 
  - But is the greedy policy always an **improvement**?
- **Proposition**: the greedy policy for  $Q_{\pi}$  (value of  $\pi$ ) is never worse than  $\pi$
- Corollary (**Bellman optimality**): if  $\pi$  is greedy for its value  $Q_{\pi}$  then it is optimal
  - In a finite MDP, the iteration  $\pi \xrightarrow{\text{evaluate}} Q_{\pi} \xrightarrow{\text{greedy}} \pi$  **converges**, and then  $\pi$  is optimal

# The RL scheme

---



# Policy Iteration

- If we know the MDP (**model-based**), we can just alternate evaluate/greedy:

---

## Algorithm Policy Iteration

---

Initialize some policy  $\pi$

**repeat**

Evaluate the policy  $Q(s, a) \leftarrow \mathbb{E}_{\xi \sim p_{\pi}} [R | s_0 = s, a_0 = a]$

Update to the greedy policy  $\pi(s) \leftarrow \arg \max_a Q(s, a)$

---

- Upon convergence,  $\pi = \pi^*$  and  $Q = Q^*$

MF

$\theta$

DP

$\pi'$

max

# Value Iteration

- We can also alternate evaluate/greedy **inside the loop** over states:

---

## Algorithm Value Iteration

---

Initialize some value function  $V$

**repeat**

**for each** state  $s$

Update  $V(s) \leftarrow \max_a (r(s, a) + \gamma \mathbb{E}_{(s'|s,a) \sim p} [V(s')])$

---

- Must update each state **repeatedly** until convergence
- Upon convergence,  $\pi^*(s) = \arg \max_a (r(s, a) + \gamma \mathbb{E}_{(s'|s,a) \sim p} [V(s')])$

MF

$\theta$

DP

$\pi'$

max



# Generalized Policy Iteration

MF

$\theta$

DP

$\pi'$

max

- We can even alternate in **any order** we wish:

$$V(s) \leftarrow \mathbb{E}_{(a|s) \sim \pi} [r(s, a) + \gamma \mathbb{E}_{(s'|s, a) \sim p} [V(s')]]$$

$$\pi(s) \leftarrow \arg \max_a (r(s, a) + \gamma \mathbb{E}_{(s'|s, a) \sim p} [V(s')])$$

- As long as each state gets each of the two update **without starvation**
  - The process will eventually **converge to  $V^*$  and  $\pi^*$**

# Model-free reinforcement learning

- We can be **model-free** using MC policy evaluation:

---

## Algorithm MC model-free RL

---

Initialize some policy  $\pi$

**repeat**

Initialize some value function  $Q$

**repeat** to convergence

Sample  $\xi \sim p_\pi$

Update  $Q(s_t, a_t) \rightarrow R_{\geq t}(\xi)$  for all  $t \geq 0$

$\pi(s) \leftarrow \arg \max_a Q(s, a)$  for all  $s$

---

- On-policy policy evaluation in the inner loop — **very inefficient**

MF

$\theta$

DP

$\pi'$

max

# Off-policy model-free reinforcement learning

- Value iteration is **model-based**:  $V(s) \leftarrow \max_a (r(s, a) + \gamma \mathbb{E}_{(s'|s,a) \sim p} [V(s')])$
- **Action-value** version:  $Q(s, a) \leftarrow r(s, a) + \gamma \mathbb{E}_{(s'|s,a) \sim p} [\max_{a'} Q(s', a')]$
- A **model-free** (data-driven) version — **Q-Learning**:
  - On **off-policy** data  $(s, a, r, s')$ , update

$$Q(s, a) \rightarrow r + \gamma \max_{a'} Q(s', a')$$

MF

$\theta$

DP

$\pi'$

max

# Recap

---

- RL is a (policy evaluation ↔ policy improvement) loop
- Policy evaluation: model-based, Monte Carlo, or Temporal-Difference
  - Temporal-Difference exploits the sequential structure using dynamic programming
- TD can be off-policy by considering the action-value Q function
  - Off-policy data can be thrown out less often as the policy changes
- Policy improvement can be greedy
  - Arbitrarily alternated with policy evaluation of any kind (MB, MC, or TD)
- Many approaches can be made differentiable for Deep RL