

# SMU: Lecture 3

Monday, March 6, 2023

*(Heavily inspired by the Stanford RL Course of Prof. Emma Brunskill, but all potential errors are mine.)*

# Plan for Today

- Recap of important concepts from lectures 1 and 2.
- Model-free control:
  - **Monte-Carlo Online Control**
  - **SARSA**
  - **Q-Learning**

# **Part 1: Where are we?**

**(Recap from the previous two lectures)**

# State Value Function of MDP

**Definition:**

$$G_t^\pi = R(X_t, A_t) + \gamma \cdot R(X_{t+1}, A_{t+1}) + \gamma^2 \cdot R(X_{t+2}, A_{t+2}) + \dots = \sum_{i=0}^{\infty} R(X_{t+i}, A_{t+i}) \cdot \gamma^i$$

$$V^\pi(s) = \mathbb{E}[G_t^\pi | X_t = s].$$

**Computing it as a solution of a system of linear equations:**

$$V^\pi(s) = \sum_{a \in A} \pi(a, s) \cdot \left[ R(s, a) + \gamma \cdot \sum_{s' \in S} P(s' | s, a) \cdot V^\pi(s') \right]$$

# MDP Control Problem

How to find  $\pi^*(s) = \arg \max_{\pi} V^{\pi}(s) ???$

# MDP Control Problem

To be fully rigorous, we should write it like this, because there may be multiple optimal policies but only one optimal state-value function.

How to find  $\pi^*(s) \in \arg \max_{\pi} V^{\pi}(s) ???$

# State-Action Value Q

- **Definition:**

$$Q^\pi(s, a) = R(s, a) + \gamma \cdot \sum_{s' \in \mathcal{S}} P(s' | s, a) \cdot V^\pi(s').$$

- **Intuition:**

- The value of the return that we obtain if we first take the action  $a$  in the state  $s$  and then follow the policy  $\pi$  (including when we visit  $s$  again).
- *Think of it as perturbing the policy  $\pi$  — we deviate from following the policy  $\pi$  only in the first step in  $s$ .*

# Policy Improvement Step

- **Given:** An MDP and a **policy**  $\pi_i$  that we want to improve (if possible).
- **DO:**

- For all  $s \in S$ , compute  $Q^{\pi_i}(s, a)$  as defined on the previous slide, i.e.

$$Q^{\pi_i}(s, a) = R(s, a) + \gamma \cdot \sum_{s' \in S} P(s' | s, a) \cdot V^{\pi_i}(s').$$

- **Compute new policy for all  $s \in S$ :**

$$\pi_{i+1}(s) = \arg \max_{a \in S} Q^{\pi_i}(s, a)$$

*Here, we use the fact that our policy is deterministic for simpler notation (treating policy as a function).*

*Using our previous notation we could write:*

$$\pi(a | s) = \begin{cases} 1 & \text{if } a = \arg \max_{a \in A} Q^{\pi_i}(s, a) \\ 0 & \text{otherwise} \end{cases}$$

# Policy Iteration

$i = 0$

**Initialize**  $\pi_0$  randomly.

**DO**

$V^{\pi_i} =$  Compute the state-value function, evaluating  $\pi_i$ .

$\pi_{i+1} =$  Policy improvement of  $\pi_i$ .

$i = i + 1$

**WHILE**  $\|\pi_i - \pi_{i-1}\|_1 > 0$  /\* if policy changed \*/

**Policy iteration finds the globally optimal policy!**

# “Greedy Policy w.r.t. $Q^\pi(s, a)$ ”

## Terminological note:

The policy satisfying

$$\pi'(s) = \arg \max_{a \in \mathcal{S}} Q^\pi(s, a)$$

is called **greedy policy w.r.t. the Q-function  $Q^\pi(s, a)$** .

*(again, formally, we should be writing  $\pi'(s) \in \arg \max_{a \in \mathcal{S}} Q^\pi(s, a)$  but we will*

*just assume for simplicity that  $\arg \max$  breaks ties in some consistent way and returns always only one state).*

# Value Iteration

Set  $k = 1$

Initialize  $V_0(s) = 0$  for all  $s \in \mathcal{S}$

**DO:**

$$V_k(s) = \max_{a \in A} \left[ R(s, a) + \gamma \cdot \sum_{s' \in \mathcal{S}} P(s' | s, a) \cdot V_{k-1}(s') \right]$$

Bellman backup  $B[V]$



**WHILE**  $\|V_k - V_{k-1}\|_\infty \geq \epsilon$

- To extract an optimal policy, we can extract a deterministic (not necessarily unique) policy:

$$\pi(s) = \arg \max_{a \in A} \left[ R(s, a) + \sum_{s' \in \mathcal{S}} P(s' | s, a) \cdot V(s') \right].$$

# Problem: Model-Free Policy Evaluation

- Given a policy and an MDP with unknown parameters (or generally an environment with which we can interact), **estimate the value function.**

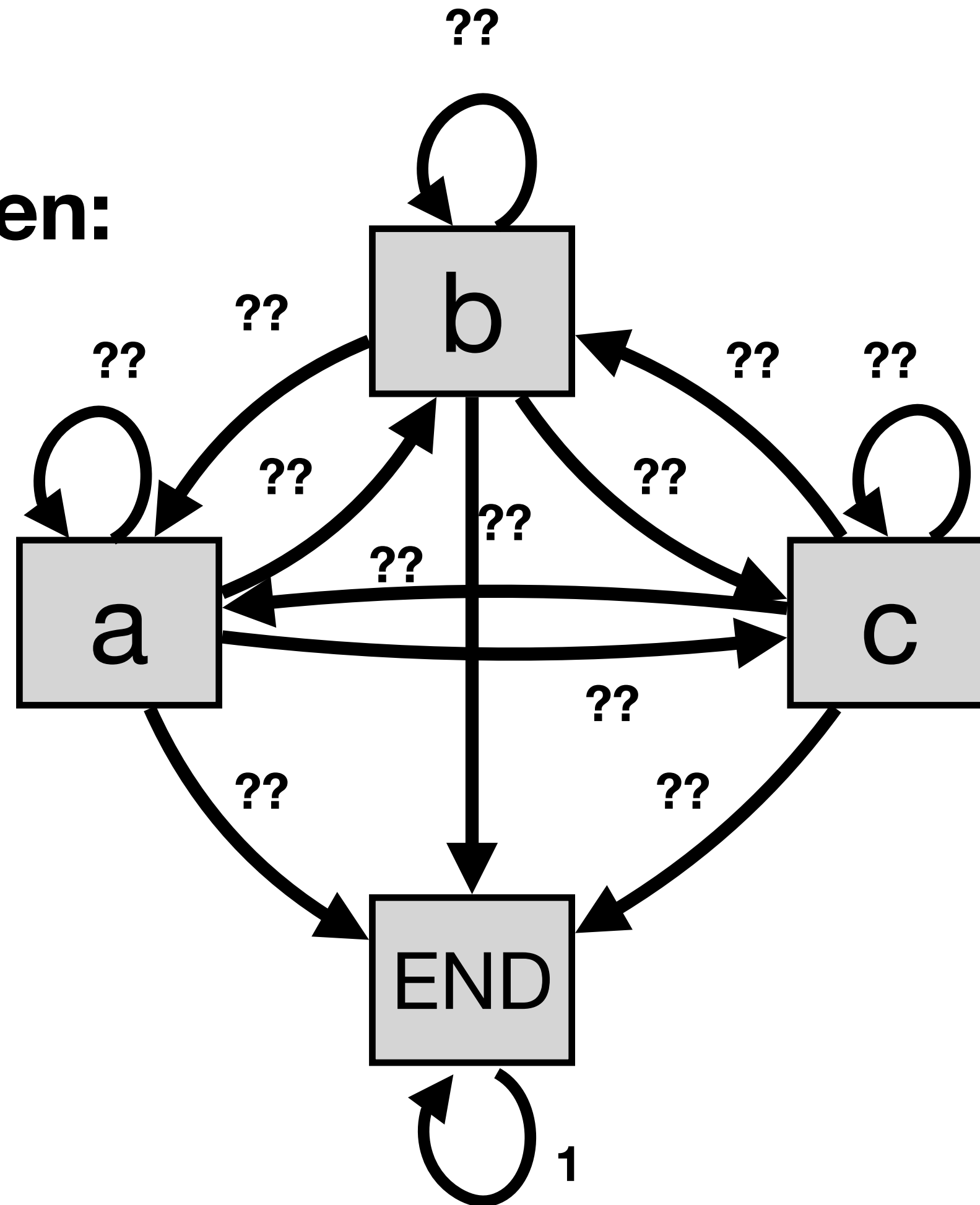
# Example

Agent:



Rewards??

States are given:



Actions are given:

$$A = \{l, r\}$$



Policy is given, e.g.:

$$\pi(l | a) = 0.2, \pi(r | a) = 0.8,$$
$$\pi(l | b) = 0.3, \pi(r | b) = 0.7,$$

...

# First/Every-Visit Monte-Carlo Evaluation

**Initialize:**  $G(s) = 0$ ,  $N(s) = 0$ ,  $V^\pi(s) = \text{undefined}$  for all  $s \in S$ .

**For**  $i = 1, \dots, N$ :

Sample episode  $e_i := s_{i,1}, a_{i,1}, r_{i,1}, s_{i,2}, a_{i,2}, r_{i,2}, \dots, s_{i,T_i}$ .

**For** each time step  $1 \leq t \leq T_i$ :

**If**  $t$  is the first occurrence of state  $s$  in the episode  $e_i$

$$g_{i,t} := r_{i,t} + \gamma \cdot r_{i,t+1} + \gamma^2 \cdot r_{i,t+2} + \dots + \gamma^{T_i-t} \cdot r_{i,T_i}$$

$$N(s) := N(s) + 1 \text{ /* Increment total visits counter */}$$

$$G(s) := G(s) + g_{i,t} \text{ /* Increment total return counter */}$$

$$V^\pi(s) := G(s)/N(s) \text{ /* Update current estimate */}$$

# Temporal Difference Learning

- **TD learning** combines Monte-Carlo estimation and dynamic programming ideas.
- **TD learning** can be used both in episodic and infinite-horizon non-episodic settings,
- **TD learning** updates estimates of  $V^\pi$  continually, after every consecutive tuple *state-action-reward-state* (therefore we do not need to wait till the end of an episode).

....

# TD-Learning: Pseudocode

**Initialize:**  $V^\pi(s) = 0$  for all  $s \in \mathcal{S}$

**Loop:**

Sample tuple  $(s_t, a_t, r_t, s_{t+1})$ .

Update  $V^\pi(s_t) := V^\pi(s_t) + \alpha \cdot \underbrace{(r_{i,t} + \gamma \cdot V^\pi(s_{t+1}) - V^\pi(s_t))}_{\text{TD target}}$

# **Part 2: Model-Free Control (Problem Statement)**

# Model-Free Control

- Given an MDP with unknown parameters (or generally an environment with which we can interact), **find an optimal policy  $\pi$** .

# Running Example

- **Example we will use:**
  - Agent (ladybug)
  - State space:  $S = \{b, c, d, e, \text{END}\}$ , END is the terminal state.
  - Action space:  $A = \{\text{left, right, eat}\}$ .
  - **We do not know**  $P(s' | s, a)$ ,  $R(s, a)$  and  $\pi(a | s)$ .
  - We want to learn some optimal policy!



**b**



**c**



**d**



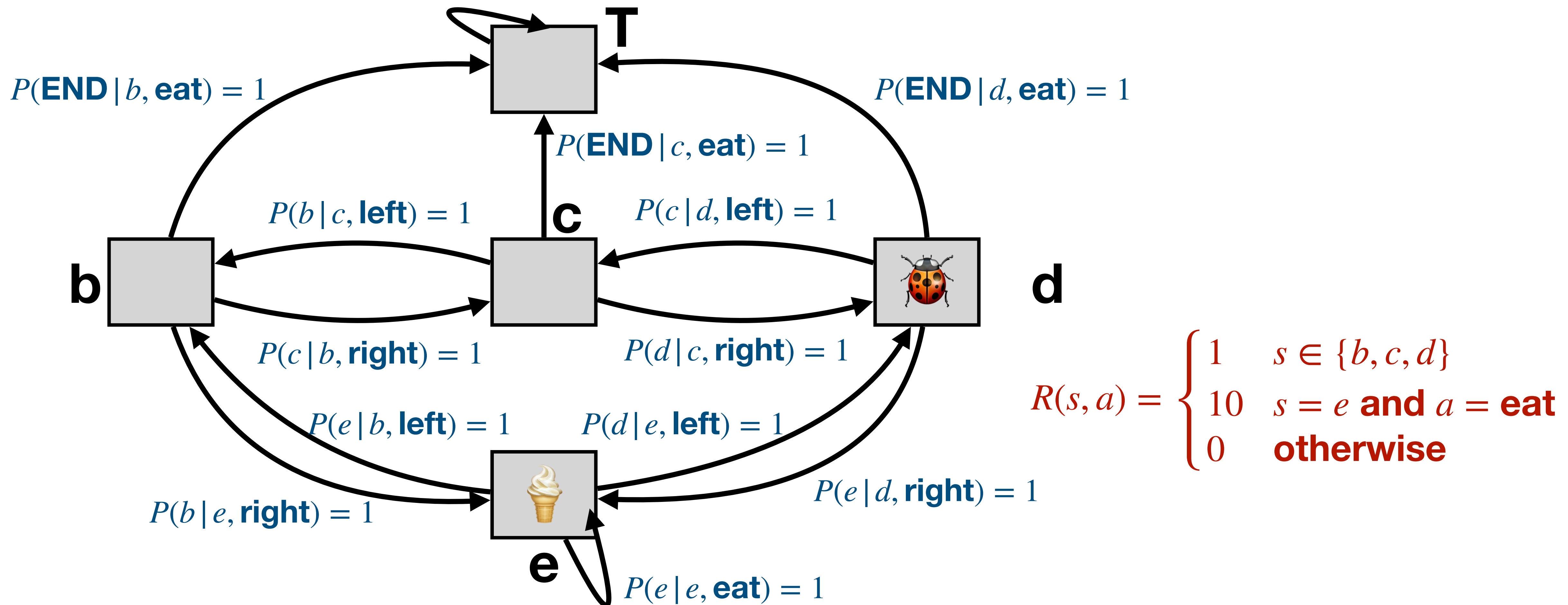
**e**



**END**

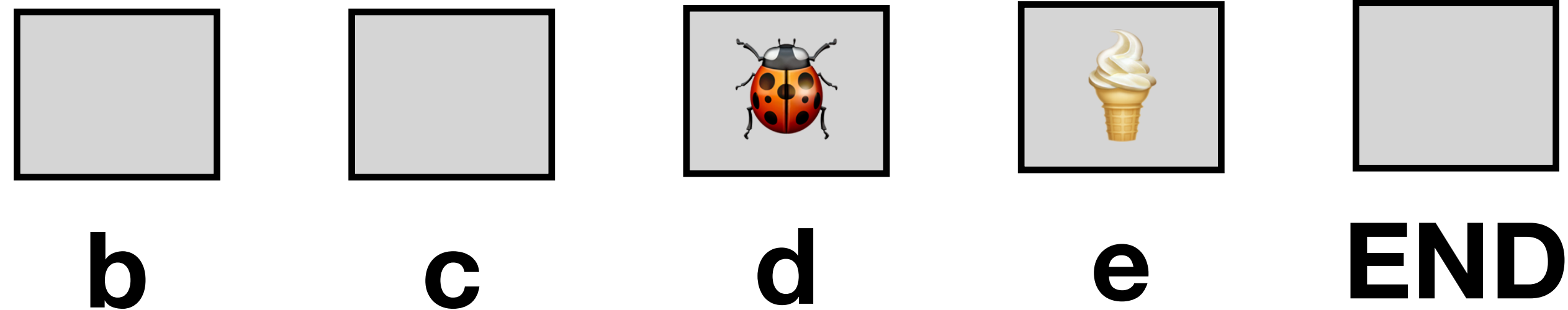
# Running Example

- Here, is what the system will behave like - this is just for you to have some intuition, the RL algorithm will not have access to this information.



# **Part 3: Model-Free Policy Iteration**

# An Idea



- What if we wanted to use policy iteration to find the optimal policy?
- What would we need?
- **Answer:** We would need to be able to compute the state-action value function  $Q^\pi(s, a)$  for any policy  $\pi$ . But that's not possible because we do not know the parameters of the MDP...
- **Idea:** Could we estimate  $Q^\pi(s, a)$  in a similar way as we were estimating  $V^\pi(s)$  last week? And then use policy improvement on that estimated  $Q^\pi(s, a)$ ?

# MC Estimation of $Q^\pi(s, a)$

Last time we talked about MC Estimation of the value function.

We can use the same idea for the estimation of the state-action value function  $Q^\pi(s, a)$ ...

...then use that estimated  $Q^\pi(s, a)$  as in policy iteration...

# MC Estimation of $Q^\pi(s, a)$

Last time we talked about MC Estimation of the value function.

We can use the same idea for the estimation of the state-action value function  $Q^\pi(s, a)$ ...

...then use that estimated  $Q^\pi(s, a)$  as in policy iteration...

...and see how it fails if done naively.

# A Naive Idea

- **THIS WILL NOT WORK (YET):**

**Initialize:**  $G(s, a) = 0$ ,  $N(s, a) = 0$  for all  $s \in S$ ,  $\pi_1 = \pi$  (the given policy).

**For**  $i = 1, \dots, N$ :

Sample episode  $e_i := s_{i,1}, a_{i,1}, r_{i,1}, s_{i,2}, a_{i,2}, r_{i,2}, \dots, s_{i,T_i}$  using  $\pi_i$ .

**For** each time step  $1 \leq t \leq T_i$ :

(If  $t$  is the first occurrence of state  $s$  followed by the action  $a$  in the episode  $e_i$  - Use this if you want first-visit MC)

$s_t$  is the state visited at time  $t$  in the episode  $e_i$

$a_t$  is the action taken at time  $t$  in the episode  $e_i$

$$g_{i,t} := r_{i,t} + \gamma \cdot r_{i,t+1} + \gamma^2 \cdot r_{i,t+2} + \dots + \gamma^{T_i-t} \cdot r_{i,T_i}$$

$$N(s) := N(s) + 1 \text{ /* Increment total visits counter */}$$

$$G(s_t, a_t) := G(s_t, a_t) + g_{i,t} \text{ /* Increment total return counter */}$$

$$Q(s_t, a_t) := G(s_t, a_t) / N(s_t, a_t) \text{ /* Update current estimate */}$$

Set  $\pi_{i+1} = \text{greedy policy w.r.t. } Q$ , i.e.,  $\pi(s) = \arg \max_{a \in A} Q(s, a)$  /\* breaking ties consistently \*/.



# Let's see why it will not work!

$S = \{b, c, d, e, \text{END}\}, A = \{\text{left, right, eat}\}$

$\pi(b) = \pi(c) = \pi(e) = \text{left}, \pi(d) = \text{eat}$

$e_1 = c, \text{left}, 1, b, \text{left}, 1, e, \text{left}, 1, d, \text{eat}, 0, \text{END}$

How can we ever estimate, e.g.,  $Q^\pi(b, \text{right})$ ??

*The problem is we may never update the estimate for  $Q^\pi(b, \text{right})$  because the action taken in the state  $b$  is always left.*

- **A simple idea** (that will not work yet... and will illustrate why we need to think about exploration):

- **THIS WILL NOT WORK (YET):**

**Initialize:**  $G(s, a) = 0, N(s, a) = 0$  for all  $s \in S$ .

**For**  $i = 1, \dots, N$ :

Sample episode  $e_i := s_{i,1}, a_{i,1}, r_{i,1}, s_{i,2}, a_{i,2}, r_{i,2}, \dots, s_{i,T_i}$   
using  $\pi$ .

**For** each time step  $1 \leq t \leq T_i$ :

(**If**  $t$  is the first occurrence of state  $s$  followed by the action  $a$  in the episode  $e_i$  - Use this if you want first-visit MC)

$s_t$  is the state visited at time  $t$  in the episode  $e_i$

$a_t$  is the action taken at time  $t$  in the episode  $e_i$

$g_{i,t} := r_{i,t} + \gamma \cdot r_{i,t+1} + \gamma^2 \cdot r_{i,t+2} + \dots + \gamma^{T_i-t} \cdot r_{i,T_i}$

$N(s) := N(s) + 1$  /\* Increment total visits counter \*/

$G(s_t, a_t) := G(s_t, a_t) + g_{i,1}$  /\* Increment total return counter \*/

$Q^\pi(s_t, a_t) := G(s_t, a_t) / N(s_t, a_t)$  /\* Update current estimate \*/

# $\varepsilon$ -Greedy Policy

- Given a Q-function  $Q(s, a)$ , we define the  $\varepsilon$ -greedy policy w.r.t.  $Q$  as

We assume ties are decided consistently

$$\pi(a | s) = \begin{cases} 1 - \varepsilon + \frac{\varepsilon}{|A|} & \text{when } a = \arg \max_{a \in A} Q(s, a) \\ \frac{\varepsilon}{|A|} & \text{when } a \neq \arg \max_{a \in A} Q(s, a) \end{cases}$$

# MC On Policy Iteration

**Initialize:**  $G(s, a) = 0$ ,  $N(s, a) = 0$ ,  $Q(s, a) = 0$  for all  $s \in S$ ,  $a \in A$ .

**Initialize:**  $\varepsilon = 1$ ,  $k = 1$

**For**  $i = 1, \dots, N$ :

Sample episode  $e_i := s_{i,1}, a_{i,1}, r_{i,1}, s_{i,2}, a_{i,2}, r_{i,2}, \dots, s_{i,T_i}$  **given**  $\pi_k$ .

**For** each time step  $1 \leq t \leq T_i$ :

(**If**  $t$  is the first occurrence of state  $s$  followed by the action  $a$  in the episode  $e_i$  - Use this if you want first-visit MC)

$s_t$  is the state visited at time  $t$  in the episode  $e_i$

$a_t$  is the action taken at time  $t$  in the episode  $e_i$

$$g_{i,t} := r_{i,t} + \gamma \cdot r_{i,t+1} + \gamma^2 \cdot r_{i,t+2} + \dots + \gamma^{T_i-t} \cdot r_{i,T_i}$$

$$N(s) := N(s) + 1 \text{ /* Increment total visits counter */}$$

$$G(s_t, a_t) := G(s_t, a_t) + g_{i,t} \text{ /* Increment total return counter */}$$

$$Q(s_t, a_t) := G(s_t, a_t) / N(s_t, a_t) \text{ /* Update current estimate */}$$

**EndFor**

$$k = k + 1, \varepsilon = 1/k$$

$\pi_k = \varepsilon$ -greedy policy w.r.t.  $Q$

# Running Example (Initialization)

Let's run MC On-Policy Iteration on our running example ( $\gamma = 0.5$ ):

$k = 1, \epsilon = 1$



**b**



**c**



**d**



**e**



**END**

<b>G(s,a)</b>	<i>left</i>	<i>right</i>	<i>eat</i>
<b>b</b>	0	0	0
<b>c</b>	0	0	0
<b>d</b>	0	0	0
<b>e</b>	0	0	0

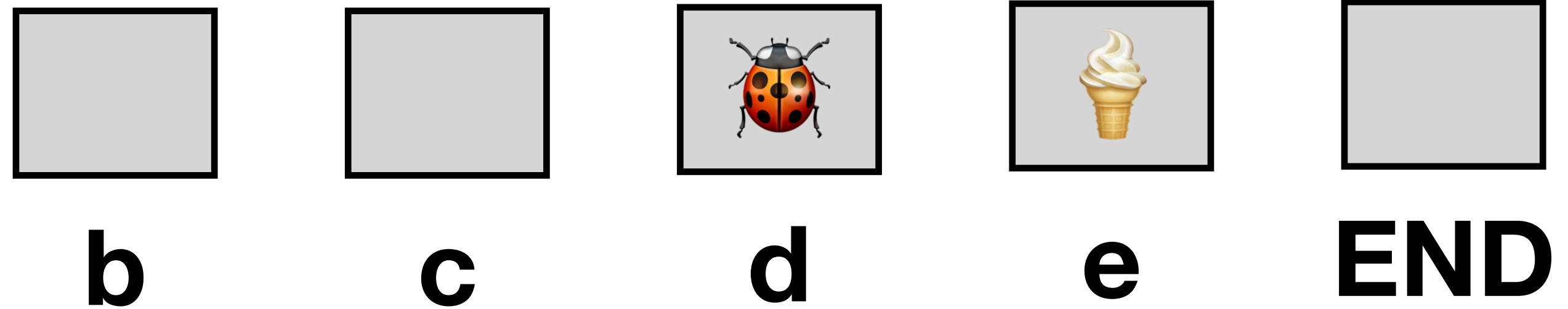
<b>N(s,a)</b>	<i>left</i>	<i>right</i>	<i>eat</i>
<b>b</b>	0	0	0
<b>c</b>	0	0	0
<b>d</b>	0	0	0
<b>e</b>	0	0	0

<b>Q(s,a)</b>	<i>left</i>	<i>right</i>	<i>eat</i>
<b>b</b>	0	0	0
<b>c</b>	0	0	0
<b>d</b>	0	0	0
<b>e</b>	0	0	0

# Running Example (Episode 1)

Let's run MC On-Policy Iteration on our running example ( $\gamma = 0.5$ ):

$$k = 1, \epsilon = 1$$



$$e_1 = d$$

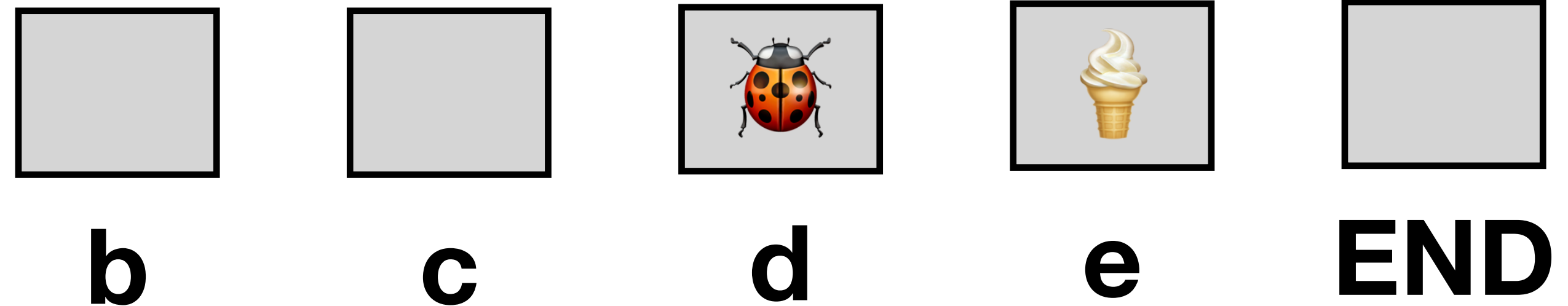
$$\pi_1(a | d) = \begin{cases} 1/3 & a = \text{left} \\ 1/3 & a = \text{right} \\ 1/3 & a = \text{eat} \end{cases}$$

$Q(s,a)$	<i>left</i>	<i>right</i>	<i>eat</i>
<i>b</i>	0	0	0
<i>c</i>	0	0	0
<i>d</i>	0	0	0
<i>e</i>	0	0	0

# Running Example (Episode 1)

Let's run MC On-Policy Iteration on our running example ( $\gamma = 0.5$ ):

$$k = 1, \epsilon = 1$$



$$e_1 = d, \text{right}$$

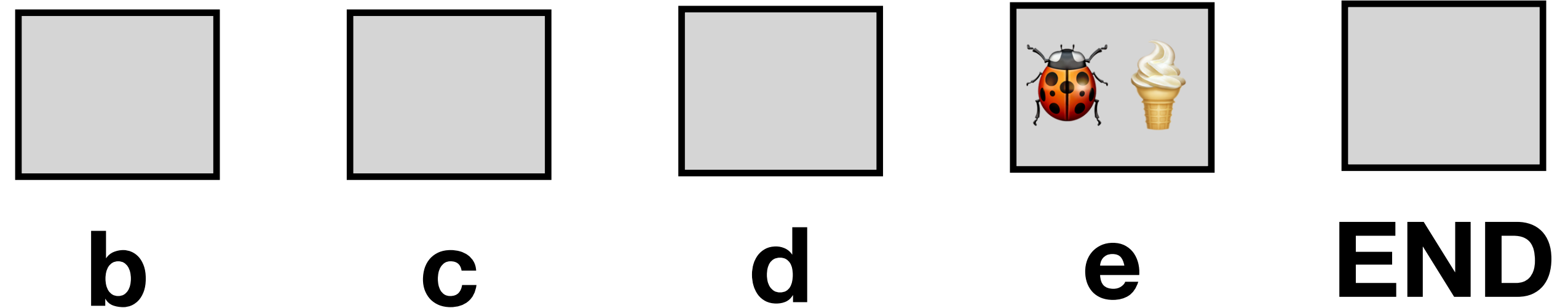
$$\pi_1(a | d) = \begin{cases} 1/3 & a = \text{left} \\ 1/3 & a = \text{right} \\ 1/3 & a = \text{eat} \end{cases}$$

Q(s,a)	left	right	eat
<b>b</b>	0	0	0
<b>c</b>	0	0	0
<b>d</b>	0	0	0
<b>e</b>	0	0	0

# Running Example (Episode 1)

Let's run MC On-Policy Iteration on our running example ( $\gamma = 0.5$ ):

$$k = 1, \epsilon = 1$$



$$e_1 = d, \text{right}, 1, e$$

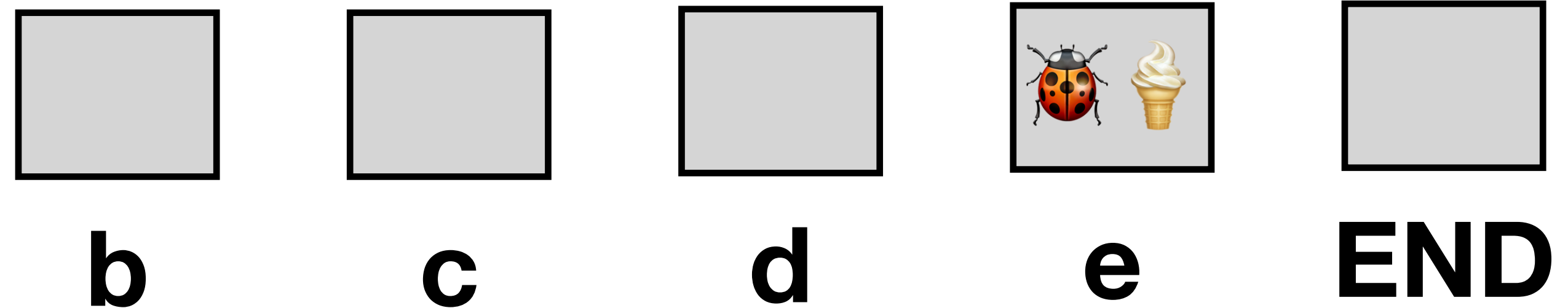
$$\pi_1(a | e) = \begin{cases} 1/3 & a = \text{left} \\ 1/3 & a = \text{right} \\ 1/3 & a = \text{eat} \end{cases}$$

$Q(s,a)$	<i>left</i>	<i>right</i>	<i>eat</i>
<i>b</i>	0	0	0
<i>c</i>	0	0	0
<i>d</i>	0	0	0
<i>e</i>	0	0	0

# Running Example (Episode 1)

Let's run MC On-Policy Iteration on our running example ( $\gamma = 0.5$ ):

$$k = 1, \epsilon = 1$$



$$e_1 = d, \text{right}, 1, e, \text{right}$$

$$\pi_1(a | e) = \begin{cases} 1/3 & a = \text{left} \\ 1/3 & a = \text{right} \\ 1/3 & a = \text{eat} \end{cases}$$

Q(s,a)	<i>left</i>	<i>right</i>	<i>eat</i>
<i>b</i>	0	0	0
<i>c</i>	0	0	0
<i>d</i>	0	0	0
<i>e</i>	0	0	0

# Running Example (Episode 1)

Let's run MC On-Policy Iteration on our running example ( $\gamma = 0.5$ ):

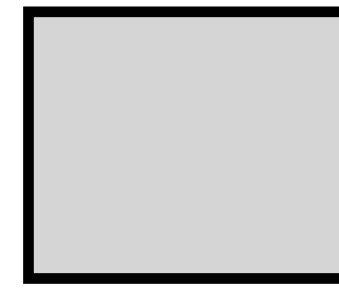
$$k = 1, \epsilon = 1$$



**b**



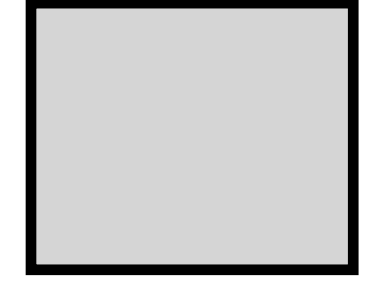
**c**



**d**



**e**



**END**

$$e_1 = d, \text{right}, 1, e, \text{right}, 1, b$$

$$\pi_1(a | b) = \begin{cases} 1/3 & a = \text{left} \\ 1/3 & a = \text{right} \\ 1/3 & a = \text{eat} \end{cases}$$

Q(s,a)	<i>left</i>	<i>right</i>	<i>eat</i>
<i>b</i>	0	0	0
<i>c</i>	0	0	0
<i>d</i>	0	0	0
<i>e</i>	0	0	0

# Running Example (Episode 1)

Let's run MC On-Policy Iteration on our running example ( $\gamma = 0.5$ ):

$$k = 1, \epsilon = 1$$



**b**



**c**



**d**



**e**



**END**

$$e_1 = d, \text{right}, 1, e, \text{right}, 1, b, \text{eat}$$

$$\pi_1(a | b) = \begin{cases} 1/3 & a = \text{left} \\ 1/3 & a = \text{right} \\ 1/3 & a = \text{eat} \end{cases}$$

Q(s,a)	<i>left</i>	<i>right</i>	<i>eat</i>
<i>b</i>	0	0	0
<i>c</i>	0	0	0
<i>d</i>	0	0	0
<i>e</i>	0	0	0

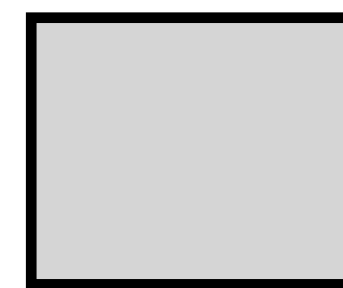
# Running Example (Episode 1)

Let's run MC On-Policy Iteration on our running example ( $\gamma = 0.5$ ):

$k = 1, \epsilon = 1$



**b**



**c**



**d**



**e**



**END**

$e_1 = d, \text{right}, 1, e, \text{right}, 1, b, \text{eat}, 0, \text{END}$

Q(s,a)	<i>left</i>	<i>right</i>	<i>eat</i>
<i>b</i>	0	0	0
<i>c</i>	0	0	0
<i>d</i>	0	0	0
<i>e</i>	0	0	0

# Running Example (Episode 1)

Now we use First-Visit MC to update  $G$ ,  $N$  and  $Q$ .

$$k = 1, \epsilon = 1$$

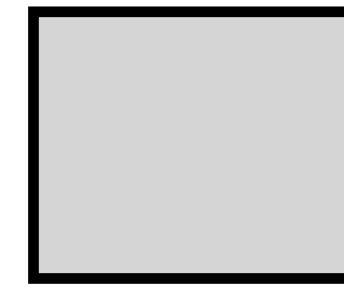
$$e_1 = d, \text{right}, 1, e, \text{right}, 1, b, \text{eat}, T$$



**b**



**c**



**d**



**e**



**END**

$G(s,a)$	<i>left</i>	<i>right</i>	<i>eat</i>
<b>b</b>	0	0	0
<b>c</b>	0	0	0
<b>d</b>	0	1.5	0
<b>e</b>	0	1	0

$N(s,a)$	<i>left</i>	<i>right</i>	<i>eat</i>
<b>b</b>	0	0	1
<b>c</b>	0	0	0
<b>d</b>	0	1	0
<b>e</b>	0	1	0

$Q(s,a)$	<i>left</i>	<i>right</i>	<i>eat</i>
<b>b</b>	0	0	0
<b>c</b>	0	0	0
<b>d</b>	0	1.5	0
<b>e</b>	0	1	0

# Running Example (Episode 1)

Now we use First-Visit MC to update  $G$ ,  $N$  and  $Q$ .

$$k = 1, \epsilon = 1$$

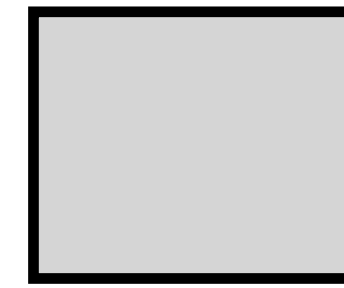
$$e_1 = \boxed{d, \text{right}, 1}, e, \text{right}, 1, b, \text{eat}, T$$



**b**



**c**



**d**



**e**



**END**

$G(s,a)$	<i>left</i>	<i>right</i>	<i>eat</i>
<i>b</i>	0	0	0
<i>c</i>	0	0	0
<i>d</i>	0	1.5	0
<i>e</i>	0	1	0

$N(s,a)$	<i>left</i>	<i>right</i>	<i>eat</i>
<i>b</i>	0	0	1
<i>c</i>	0	0	0
<i>d</i>	0	1	0
<i>e</i>	0	1	0

$Q(s,a)$	<i>left</i>	<i>right</i>	<i>eat</i>
<i>b</i>	0	0	0
<i>c</i>	0	0	0
<i>d</i>	0	1.5	0
<i>e</i>	0	1	0

# Running Example (Episode 1)

Now we use First-Visit MC to update  $G$ ,  $N$  and  $Q$ .

$$k = 1, \epsilon = 1$$

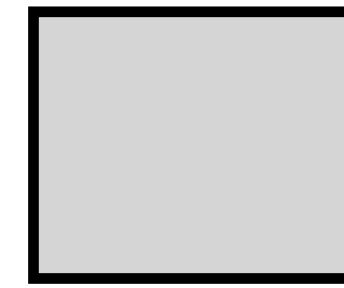
$$e_1 = d, \text{right}, 1, e, \text{right}, 1, b, \text{eat}, T$$



**b**



**c**



**d**



**e**



**END**

$G(s,a)$	<i>left</i>	<i>right</i>	<i>eat</i>
<i>b</i>	0	0	0
<i>c</i>	0	0	0
<i>d</i>	0	1.5	0
<i>e</i>	0	1	0

$N(s,a)$	<i>left</i>	<i>right</i>	<i>eat</i>
<i>b</i>	0	0	1
<i>c</i>	0	0	0
<i>d</i>	0	1	0
<i>e</i>	0	1	0

$Q(s,a)$	<i>left</i>	<i>right</i>	<i>eat</i>
<i>b</i>	0	0	0
<i>c</i>	0	0	0
<i>d</i>	0	1.5	0
<i>e</i>	0	1	0

# Running Example (Episode 1)

Now we use First-Visit MC to update  $G$ ,  $N$  and  $Q$ .

$$k = 1, \epsilon = 1$$

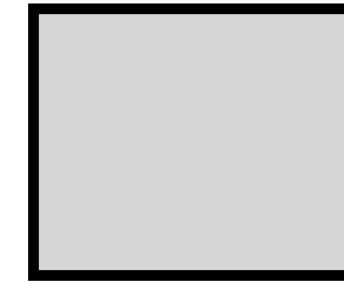
$$e_1 = d, \text{right}, 1, e, \text{right}, 1, \boxed{b, \text{eat}}, T$$



**b**



**c**



**d**



**e**



**END**

$G(s,a)$	<i>left</i>	<i>right</i>	<i>eat</i>
<i>b</i>	0	0	0
<i>c</i>	0	0	0
<i>d</i>	0	1.5	0
<i>e</i>	0	1	0

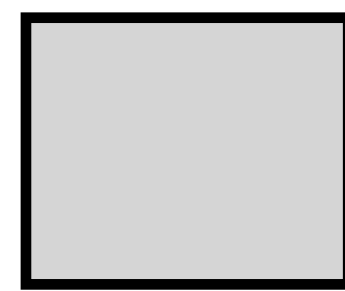
$N(s,a)$	<i>left</i>	<i>right</i>	<i>eat</i>
<i>b</i>	0	0	1
<i>c</i>	0	0	0
<i>d</i>	0	1	0
<i>e</i>	0	1	0

$Q(s,a)$	<i>left</i>	<i>right</i>	<i>eat</i>
<i>b</i>	0	0	0
<i>c</i>	0	0	0
<i>d</i>	0	1.5	0
<i>e</i>	0	1	0

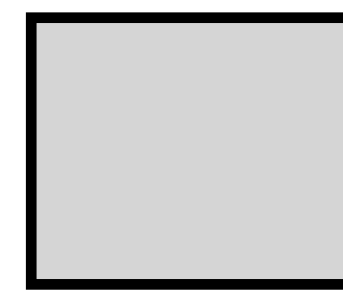
# Running Example (Episode 1)

Now we update the policy  $\pi$ . First, we get the greedy policy w.r.t.  $Q(s, a)$

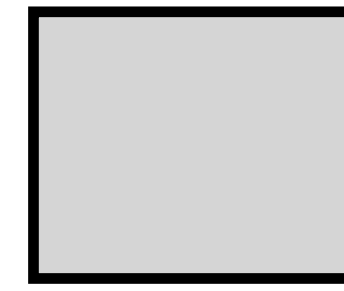
$$k = 1, \epsilon = 1$$



**b**



**c**



**d**



**e**



**END**

Let us suppose that if there is tie in  $\arg \max_{a \in A}$  then the preference is eat  $<$  right  $<$  left (i.e. we prefer left over right and right over eat)

$$\pi_{\text{greedy}}(d) = \pi_{\text{greedy}}(e) = \text{right},$$



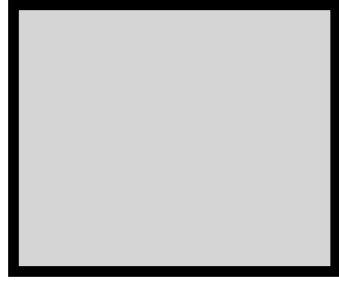


$$\pi_{\text{greedy}}(b) = \pi_{\text{greedy}}(c) = \text{left}.$$

$Q(s,a)$	<i>left</i>	<i>right</i>	<i>eat</i>
<i>b</i>	0	0	0
<i>c</i>	0	0	0
<i>d</i>	0	1.5	0
<i>e</i>	0	1	0

# Running Example (Episode 1)

Now we update the policy  $\pi$ . First, we get the greedy policy w.r.t.  $Q(s, a)$

Now, we update  $k = 2; \epsilon = 0.5$

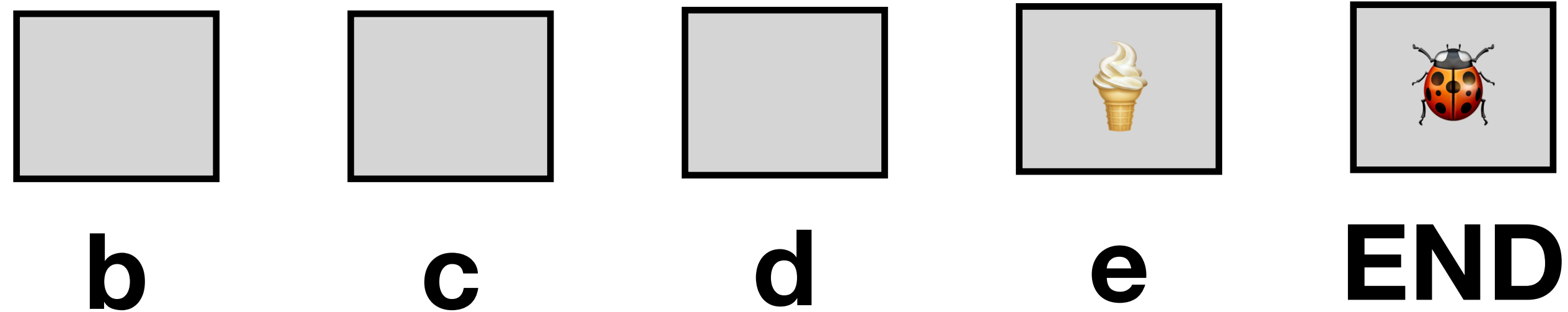
				
<b>b</b>	<b>c</b>	<b>d</b>	<b>e</b>	<b>END</b>

The new policy  $\pi$  will be the  $\epsilon$ -greedy policy:

$$\pi(a | s) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|A|} & \text{when } a = \pi_{\text{greedy}}(s) \\ \frac{\epsilon}{|A|} & \text{when } a \neq \pi_{\text{greedy}}(s) \end{cases}$$

We then run the next iteration with this new policy  $\pi$ .

# Running Example (Episode 1)



As  $k$  increases, the algorithm will converge to the optimal policy:

$$\pi(b) = \mathbf{left}, \pi(c) = \mathbf{left}, \pi(d) = \mathbf{right}, \pi(e) = \mathbf{eat}$$

# GLIE

- We say that an algorithm has the GLIE property (= “greedy in the limit of infinite exploration”), if it satisfies the following two conditions):
- **Definition** (GLIE conditions):
  1. If a state  $s \in S$  is visited infinitely often, then each action in that state is chosen infinitely often (with probability 1)
  2. In the limit (as  $t \rightarrow \infty$ ), the learning policy is greedy with respect to the learned Q-function (with probability 1). By *greedy* we mean (ignoring the possibility of ties in the  $\arg \max$  for simplicity) that

$$\pi_{k+1}(a | s) = \begin{cases} 1 & \text{for } a = \arg \max_{a \in A} Q_k(s, a), \\ 0 & \text{otherwise.} \end{cases}$$

# MC Policy Iteration with $\varepsilon_i = 1/i$ is GLIE

- For a proof, see, e.g. *Singh, S., Jaakkola, T., Littman, M. L., & Szepesvári, C. (2000). Convergence results for single-step on-policy reinforcement-learning algorithms. Machine learning, 38(3), 287-308.*
- The formal proof is a bit tricky...
- **Note:** *There are other sequences of  $\varepsilon_i$  which guarantee GLIE as well.*

# A Theorem (Why GLIE Matters)

- **Theorem:** GLIE Monte-Carlo Control converges to the optimal state-action value function, i.e.  $Q_k(s, a) \rightarrow Q^*(s, a)$  as  $k \rightarrow \infty$ .

# Part 4: SARSA and Q-Learning

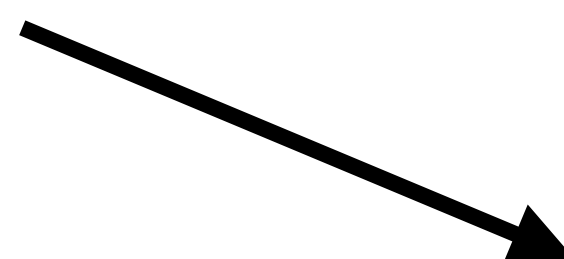
# General Form of TD-Based Methods

- **Basic idea:**
  - Replace Monte Carlo Policy Evaluation by a temporal-difference method.
  - Still use  $\epsilon$ -greedy policies to guarantee that exploration will take place.

# Bellman Equations for Q-Function

*(Something we skipped when we talked about Q-functions for MDPs but something that will be useful now.)*

**We have:**

$$V^\pi(s) = \sum_{a \in A} \pi(a | s) \cdot Q^\pi(s, a)$$

$$Q^\pi(s, a) = R(s, a) + \gamma \cdot \sum_{s' \in S} P(s' | s, a) \cdot V^\pi(s')$$

**Combining the above:**

$$Q^\pi(s, a) = R(s, a) + \gamma \cdot \sum_{s' \in S} P(s' | s, a) \cdot \sum_{a' \in A} \pi(a' | s') \cdot Q^\pi(s', a')$$

# TD-Target

**Bellman for Q-function:**

$$Q^\pi(s_t, a_t) = R(s_t, a_t) + \gamma \cdot \sum_{s_{t+1} \in \mathcal{S}} P(s_{t+1} | s_t, a_t) \cdot \sum_{a_{t+1} \in \mathcal{A}} \pi(a_{t+1} | s_{t+1}) \cdot Q^\pi(s_{t+1}, a_{t+1})$$

$$\mathbb{E}[Q^\pi(X_{t+1}, A_{t+1}) | X_t = s_t, A_t = a_t]$$

**Temporal difference update (SARSA)...**

$$Q(s_t, a_t) := Q(s_t, a_t) + \alpha \left( r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right)$$

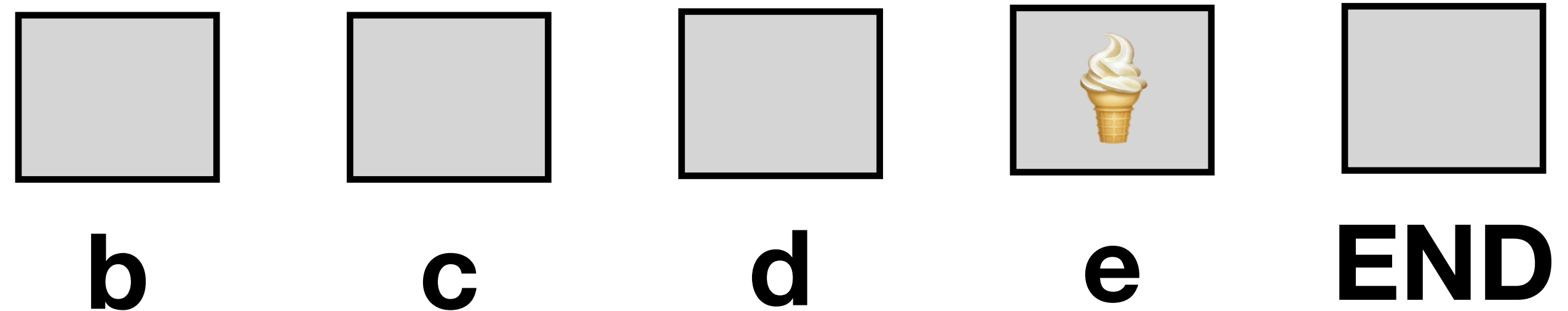
# SARSA

- 1. Initialize:** set  $\pi$  to be some  $\varepsilon$ -greedy policy, set  $t = 1$ , initialize  $Q(s, a)$ .
- 2. Sample**  $a_1$  using the distribution given by  $\pi$  in the state  $s_1$  (*for sampling, we will use the notation  $a_1 \sim \pi(s_1)$* ).
- 3. While**  $s_t$  is not a terminal state:
  - 1. Take** action  $a_t$  and observe  $r_t, s_{t+1}$ .
  - 2. Sample**  $a_{t+1} \sim \pi(s_{t+1})$  and store it for the next iteration.
  - 3.  $Q(s_t, a_t) := Q(s_t, a_t) + \alpha (r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$**
  - 4.  $\pi := \varepsilon$ -greedy( $Q$ )**
  - 5. Set  $t := t + 1$ . Update  $\varepsilon, \alpha$  /\* see next slides \*/**

# Running Example (Initialization)

Let's run SARSA on our running example ( $\gamma = 0.5$ ):

We will use  $\varepsilon_t = 1/t$ .  
 $t = 1, \varepsilon = 1, \alpha = 0.1$



<b>Q(s,a)</b>	<b>left</b>	<b>right</b>	<b>eat</b>
<b>b</b>	0	0	0
<b>c</b>	0	0	0
<b>d</b>	0	0	0
<b>e</b>	0	0	0

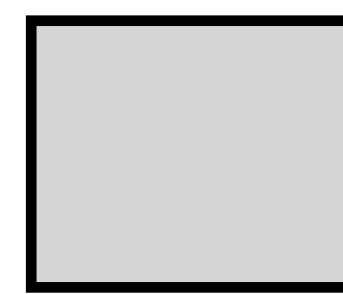
# Running Example (Initialization)

Let's run SARSA on our running example ( $\gamma = 0.5$ ):

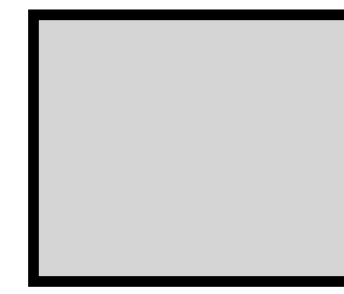
We will use  $\varepsilon_t = 1/t$ .  
 $t = 1, \varepsilon = 1, \alpha = 0.1$



**b**



**c**



**d**



**e**



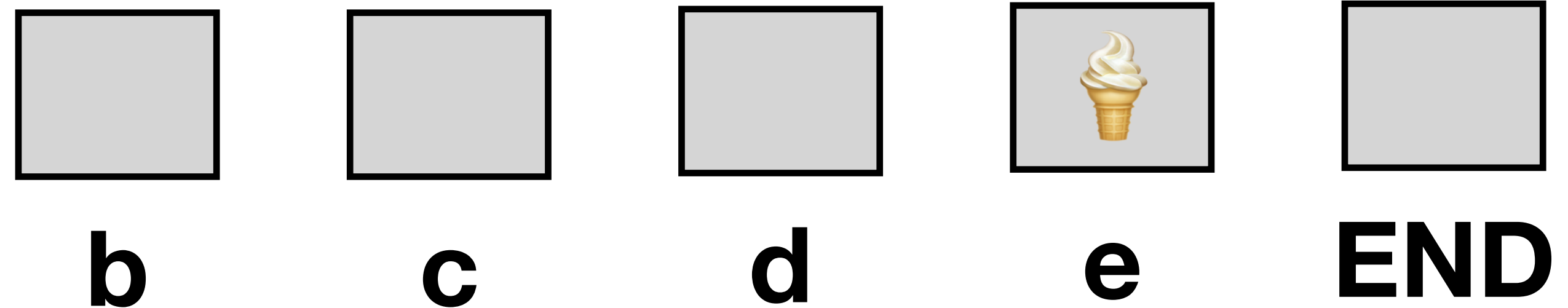
**END**

<b>Q(s,a)</b>	<b>left</b>	<b>right</b>	<b>eat</b>
<b>b</b>	0	0	0
<b>c</b>	0	0	0
<b>d</b>	0	0	0
<b>e</b>	0	0	0

# Running Example (Initialization)

Let's run SARSA on our running example ( $\gamma = 0.5$ ):

We will use  $\varepsilon_t = 1/t$ .  
 $t = 1, \varepsilon = 1, \alpha = 0.1$

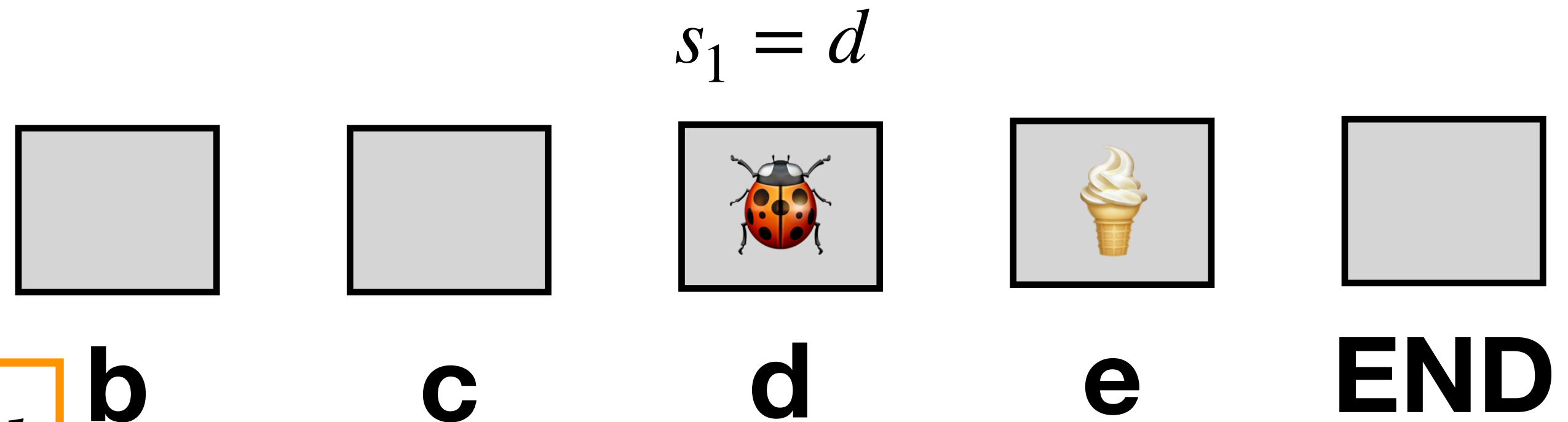


$Q(s,a)$	<i>left</i>	<i>right</i>	<i>eat</i>
<i>b</i>	0	0	0
<i>c</i>	0	0	0
<i>d</i>	0	0	0
<i>e</i>	0	0	0

# Running Example (Initialization)

Let's run SARSA on our running example ( $\gamma = 0.5$ ):

We will use  $\epsilon_t = 1/t$ .  
 $t = 1, \epsilon = 1, \alpha = 0.1$



**World samples the state  $s_1 = d$**

<b>Q(s,a)</b>	<b>left</b>	<b>right</b>	<b>eat</b>
<b>b</b>	0	0	0
<b>c</b>	0	0	0
<b>d</b>	0	0	0
<b>e</b>	0	0	0

# Running Example (Initialization)

Let's run SARSA on our running example ( $\gamma = 0.5$ ):

We will use  $\varepsilon_t = 1/t$ .

$t = 1, \varepsilon = 1, \alpha = 0.1$



**b**



**c**

$s_1 = d$



**d**



**e**



**END**

World samples the state  $s_1 = d$

We sample  $a_1$  (we do not take it yet)

$$a_1 \sim \pi_1(a | d) = \begin{cases} 1/3 & a = \text{left} \\ 1/3 & a = \text{right} \\ 1/3 & a = \text{eat} \end{cases}$$

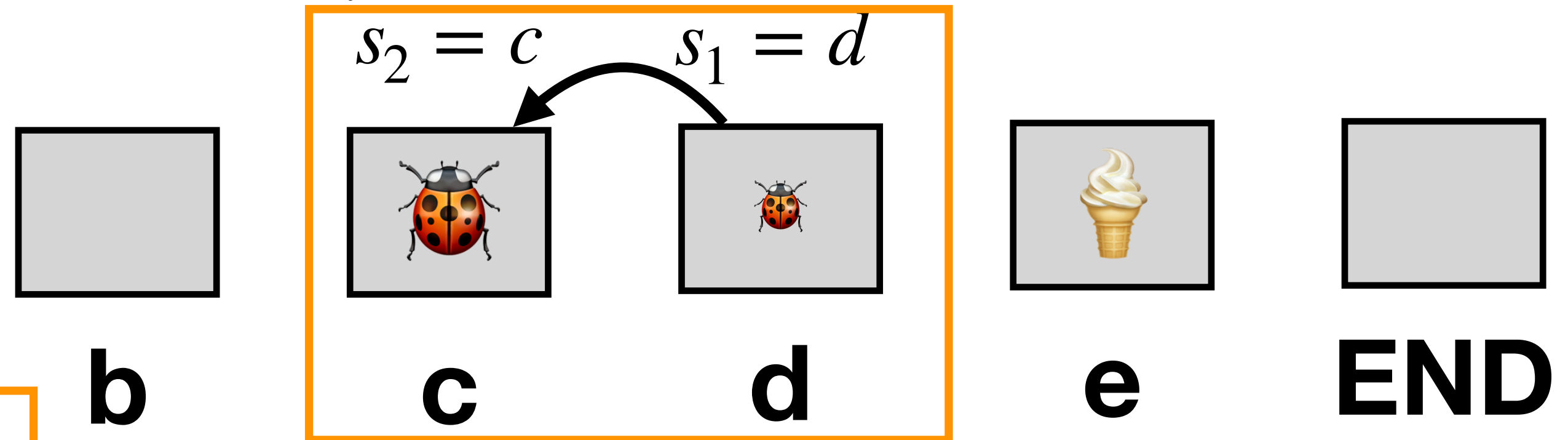
Q(s,a)	left	right	eat
<b>b</b>	0	0	0
<b>c</b>	0	0	0
<b>d</b>	0	0	0
<b>e</b>	0	0	0

# Running Example ( $t = 1$ )

Let's run SARSA on our running example ( $\gamma = 0.5$ ):

We will use  $\varepsilon_t = 1/t$ .

$t = 1, \varepsilon = 1, \alpha = 0.1$



We take the action  $a_1 = \text{left}$

We observe:  $r_1 = 1$  and  $s_2 = c$

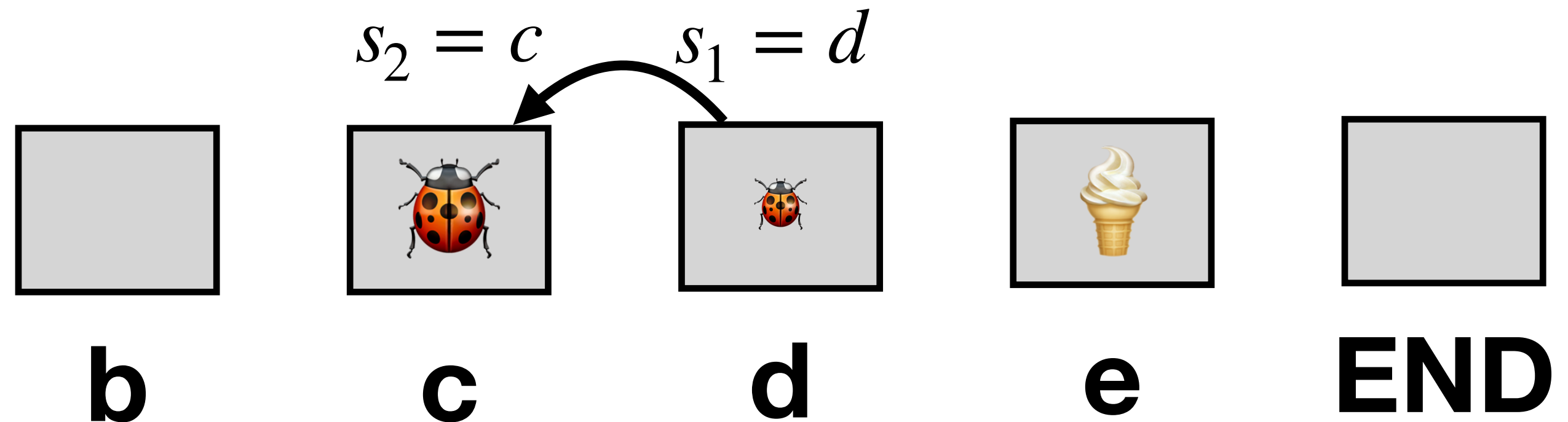
Q(s,a)	<i>left</i>	<i>right</i>	<i>eat</i>
<i>b</i>	0	0	0
<i>c</i>	0	0	0
<i>d</i>	0	0	0
<i>e</i>	0	0	0

# Running Example ( $t = 1$ )

Let's run SARSA on our running example ( $\gamma = 0.5$ ):

We will use  $\varepsilon_t = 1/t$ .

$t = 1, \varepsilon = 1, \alpha = 0.1$



We have:  $r_1 = 1$  and  $s_2 = c$

We sample  $a_2$  (we are not taking it yet)

$$a_2 \sim \pi_1(a | c) = \begin{cases} 1/3 & a = \text{left} \\ 1/3 & a = \text{right} \\ 1/3 & a = \text{eat} \end{cases}$$

Say, it is  $a_2 = \text{left}$ .

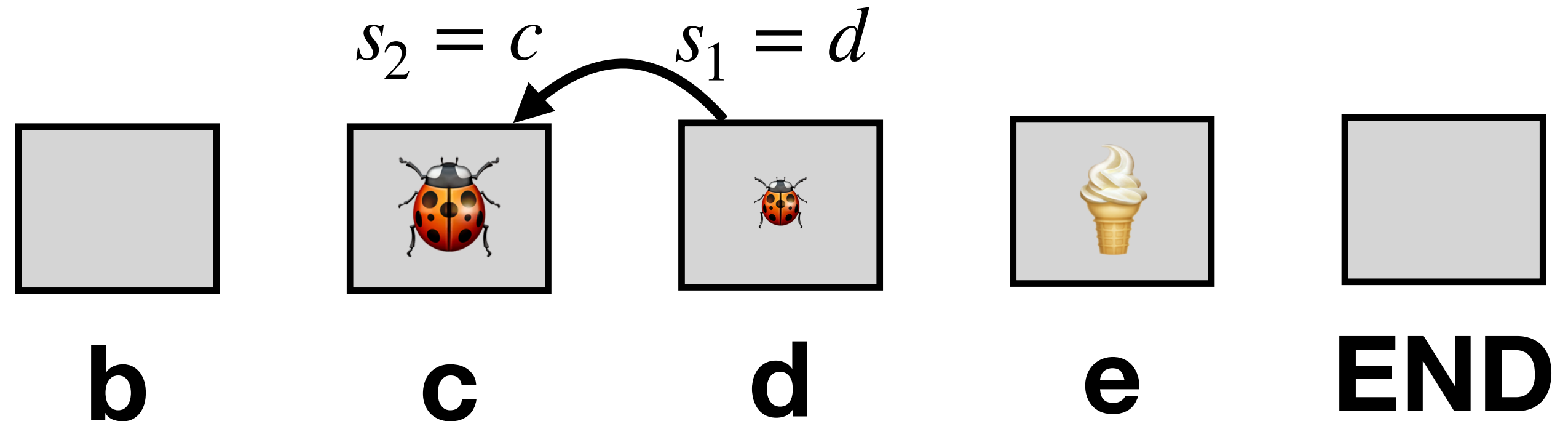
Q(s,a)	left	right	eat
<b>b</b>	0	0	0
<b>c</b>	0	0	0
<b>d</b>	0	0	0
<b>e</b>	0	0	0

# Running Example ( $t = 1$ )

Let's run SARSA on our running example ( $\gamma = 0.5$ ):

We will use  $\varepsilon_t = 1/t$ .

$t = 1, \varepsilon = 1, \alpha = 0.1$



We have:  $r_1 = 1$  and  $s_2 = c$

We sample  $a_2$  (we are not taking it yet)

$$a_2 \sim \pi_1(a | c) = \begin{cases} 1/3 & a = \text{left} \\ 1/3 & a = \text{right} \\ 1/3 & a = \text{eat} \end{cases}$$

Say, it is  $a_2 = \text{left}$ .

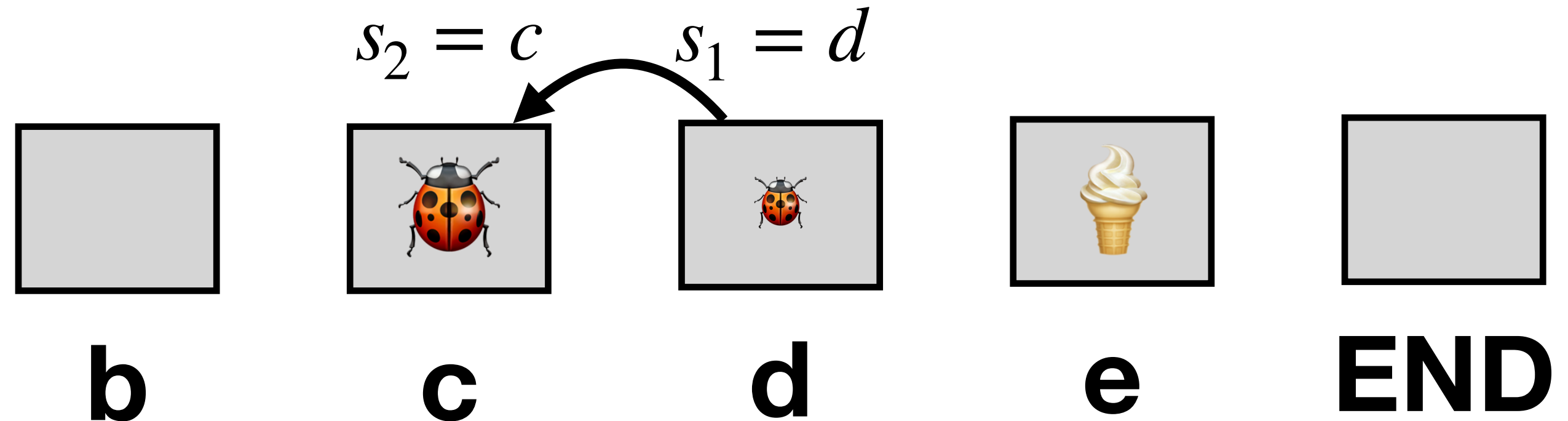
Q(s,a)	left	right	eat
<b>b</b>	0	0	0
<b>c</b>	0	0	0
<b>d</b>	0	0	0
<b>e</b>	0	0	0

# Running Example ( $t = 1$ )

Let's run SARSA on our running example ( $\gamma = 0.5$ ):

We will use  $\varepsilon_t = 1/t$ .

$t = 1, \varepsilon = 1, \alpha = 0.1$



We have:  $r_1 = 1$  and  $s_2 = c$

We sample  $a_2$  (we are not taking it yet)

$$a_2 \sim \pi_1(a | c) = \begin{cases} 1/3 & a = \text{left} \\ 1/3 & a = \text{right} \\ 1/3 & a = \text{eat} \end{cases}$$

Say, it is  $a_2 = \text{left}$ .

Q(s,a)	left	right	eat
<b>b</b>	0	0	0
<b>c</b>	0	0	0
<b>d</b>	0	0	0
<b>e</b>	0	0	0

# Running Example ( $t = 1$ )

Let's run SARSA on our running example ( $\gamma = 0.5$ ):

We will use  $\varepsilon_t = 1/t$ .

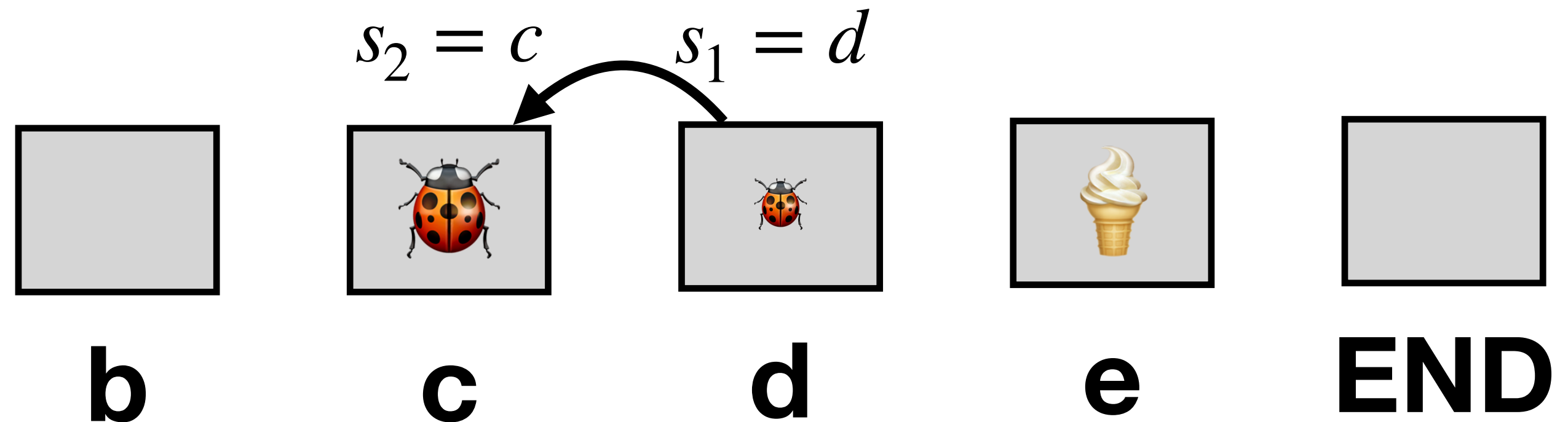
$t = 1, \varepsilon = 1, \alpha = 0.1$

We have:  $r_1 = 1$  and  $s_2 = c$

We now update the Q-function:

$$Q(d, \text{left}) := 0 + 0.1 (1 + 0.5 \cdot 0 - 0) = 0.1$$

$$Q(s_t, a_t) := Q(s_t, a_t) + \alpha (r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$$



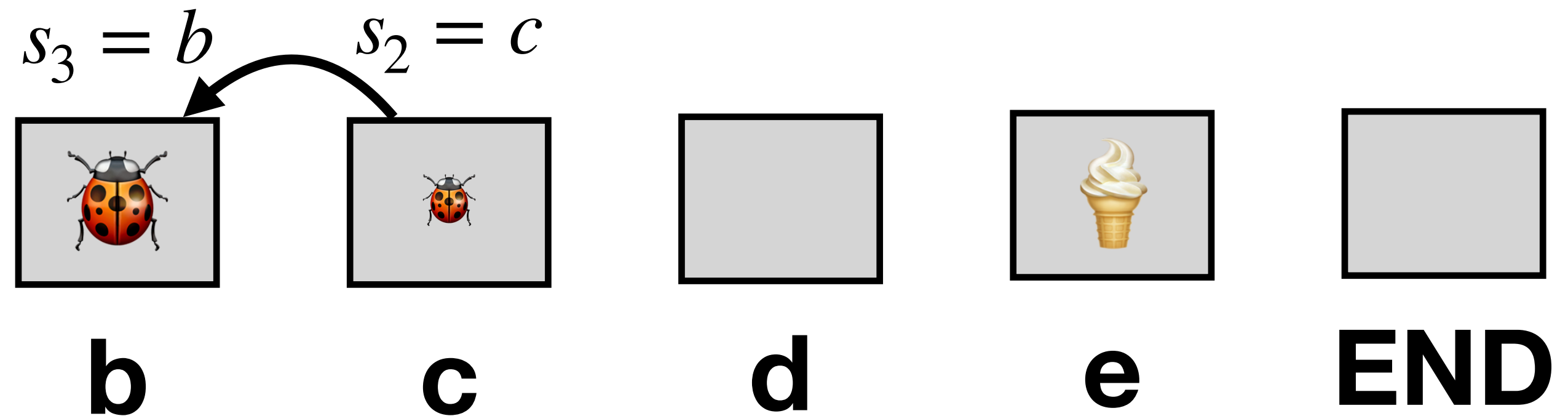
Q(s,a)	<i>left</i>	<i>right</i>	<i>eat</i>
<b>b</b>	0	0	0
<b>c</b>	0	0	0
<b>d</b>	<b>0.1</b>	0	0
<b>e</b>	0	0	0

# Running Example ( $t = 2$ )

Let's run SARSA on our running example ( $\gamma = 0.5$ ):

We will use  $\epsilon_t = 1/t$ .

$t = 2, \epsilon = 0.5, \alpha = 0.1$



We take the action  $a_2 = \text{left}$

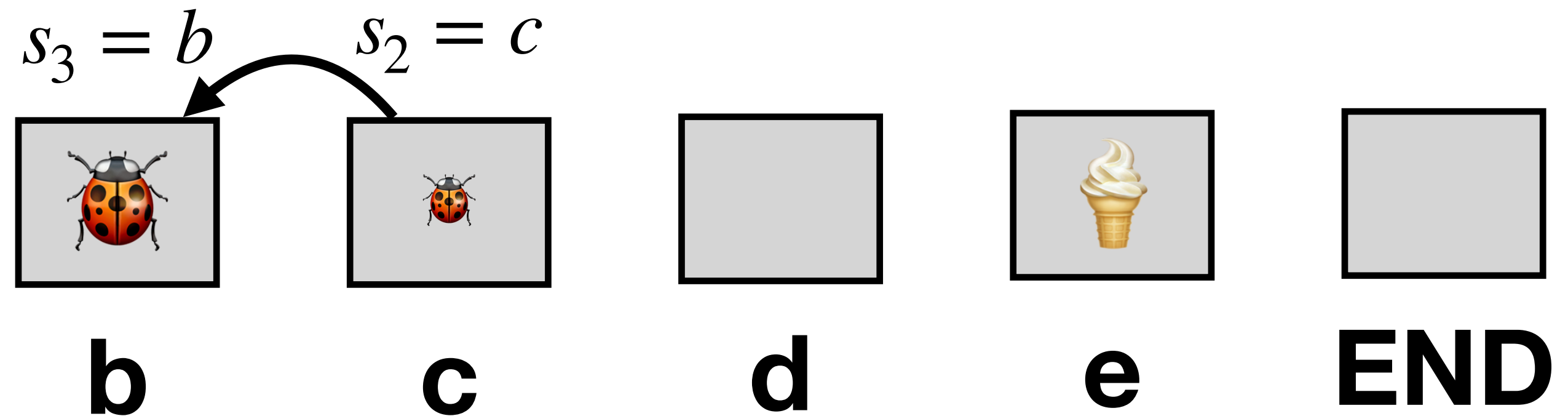
We observe:  $r_2 = 1$  and  $s_3 = b$

Q(s,a)	<i>left</i>	<i>right</i>	<i>eat</i>
<i>b</i>	0	0	0
<i>c</i>	0	0	0
<i>d</i>	0.1	0	0
<i>e</i>	0	0	0

# Running Example ( $t = 2$ )

Let's run SARSA on our running example ( $\gamma = 0.5$ ):

We will use  $\epsilon_t = 1/t$ .  
 $t = 2, \epsilon = 0.5, \alpha = 0.1$



**We take the action  $a_2 = \text{left}$**

**We observe:  $r_2 = 1$  and  $s_3 = b$**

Q(s,a)	<i>left</i>	<i>right</i>	<i>eat</i>
<i>b</i>	0	0	0
<i>c</i>	0	0	0
<i>d</i>	0.1	0	0
<i>e</i>	0	0	0

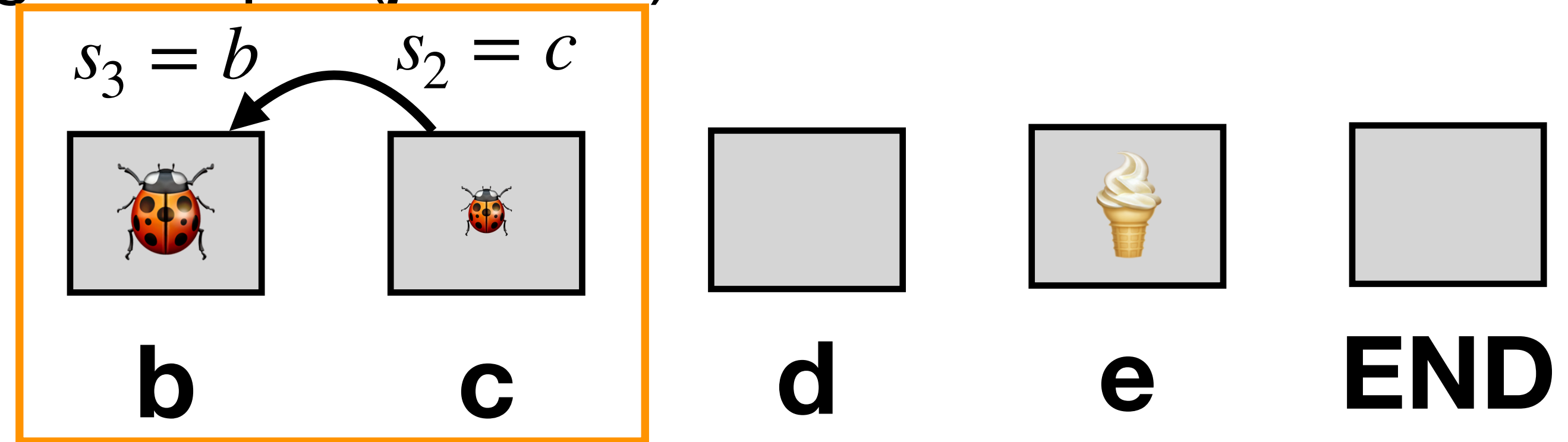
# Running Example ( $t = 2$ )

Let's run SARSA on our running example ( $\gamma = 0.5$ ):

We will use  $\epsilon_t = 1/t$ .  
 $t = 2, \epsilon = 0.5, \alpha = 0.1$

We take the action  $a_2 = \text{left}$

We observe:  $r_2 = 1$  and  $s_3 = b$

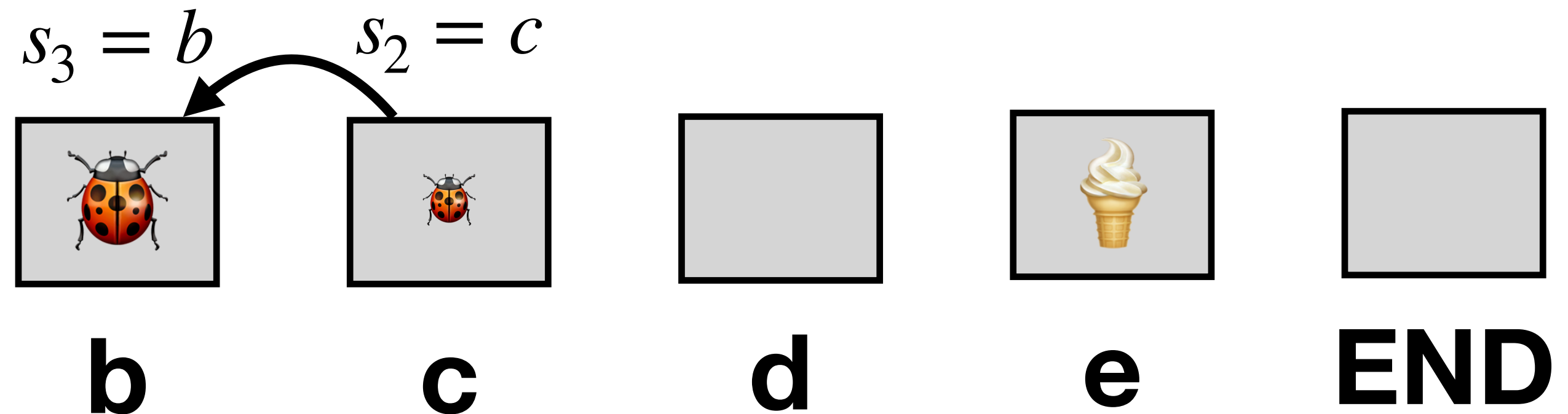


Q(s,a)	<i>left</i>	<i>right</i>	<i>eat</i>
<i>b</i>	0	0	0
<i>c</i>	0	0	0
<i>d</i>	0.1	0	0
<i>e</i>	0	0	0

# Running Example ( $t = 2$ )

Let's run SARSA on our running example ( $\gamma = 0.5$ ):

We will use  $\epsilon_t = 1/t$ .  
 $t = 2, \epsilon = 0.5, \alpha = 0.1$



We have:  $r_2 = 1$  and  $s_3 = b$

We sample  $a_3$  (we are not taking it yet)

$$\pi_1(a | b) = \begin{cases} 1 - 0.5 + 1/6 = 2/3 & a = \text{left} \\ 1/6 & a = \text{right} \\ 1/6 & a = \text{eat} \end{cases}$$

**What happened here:** Even though we did not update the estimates of the Q-function for the state  $c$ , the policy changed. Recall that we break ties (we have the preference  $\text{eat} < \text{right} < \text{left}$  and recall how we define greedy and  $\epsilon$ -greedy policies.

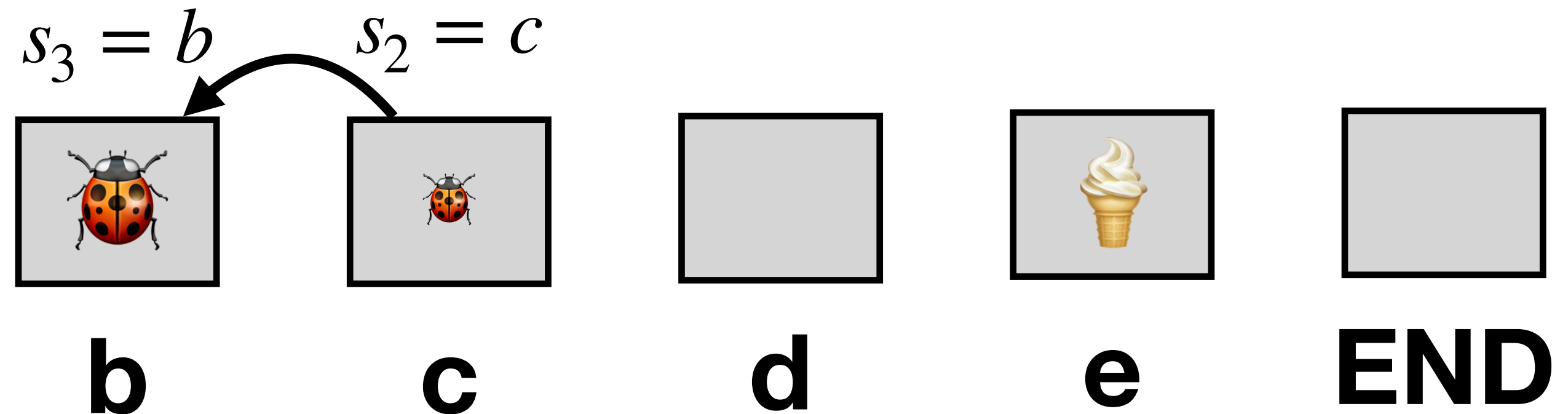
Say, it is  $a_3 = \text{right}$ .

Q(s,a)	left	right	eat
<b>b</b>	0	0	0
<b>c</b>	0	0	0
<b>d</b>	0.1	0	0
<b>e</b>	0	0	0

# Running Example ( $t = 2$ )

Let's run SARSA on our running example ( $\gamma = 0.5$ ):

We will use  $\epsilon_t = 1/t$ .  
 $t = 2, \epsilon = 0.5, \alpha = 0.1$



**We have:**  $r_2 = 1$  and  $s_3 = b$

**We sample**  $a_3$  (we are not taking it yet)

$$\pi_1(a | b) = \begin{cases} 1 - 0.5 + 1/6 = 2/3 & a = \text{left} \\ 1/6 & a = \text{right} \\ 1/6 & a = \text{eat} \end{cases}$$

**What happened here:** Even though we did not update the estimates of the Q-function for the state  $c$ , the policy changed. Recall that we break ties (we have the preference  $\text{eat} < \text{right} < \text{left}$  and recall how we define greedy and  $\epsilon$ -greedy policies.

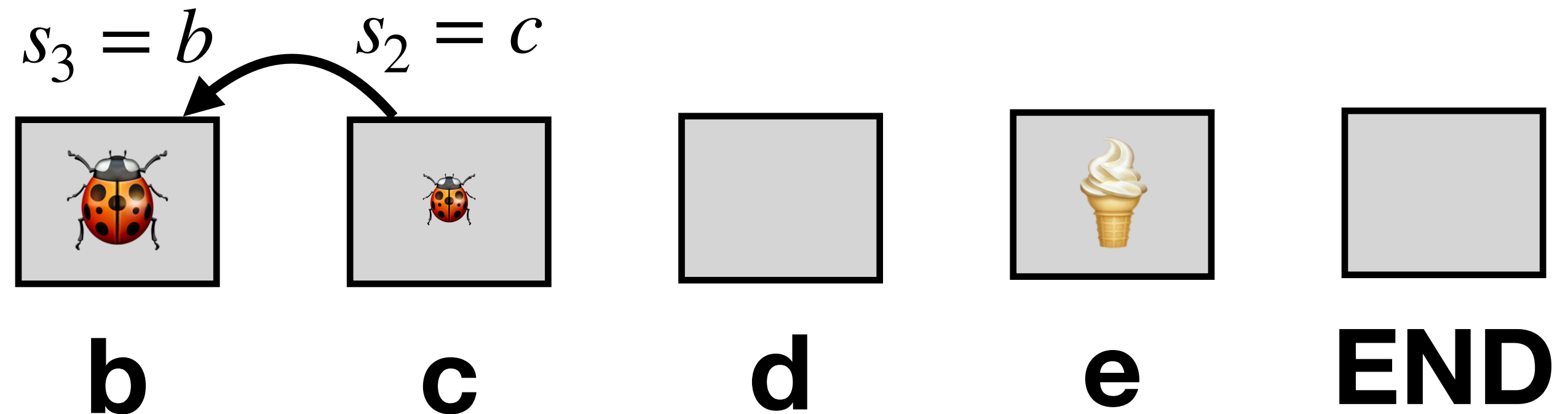
Say, it is  $a_3 = \text{right}$ .

Q(s,a)	left	right	eat
<b>b</b>	0	0	0
<b>c</b>	0	0	0
<b>d</b>	0.1	0	0
<b>e</b>	0	0	0

# Running Example ( $t = 2$ )

Let's run SARSA on our running example ( $\gamma = 0.5$ ):

We will use  $\epsilon_t = 1/t$ .  
 $t = 2, \epsilon = 0.5, \alpha = 0.1$



**We have:**  $r_2 = 1$  and  $s_3 = b$

**We sample**  $a_3$  (we are not taking it yet)

$$\pi_1(a | b) = \begin{cases} 1 - 0.5 + 1/6 = 2/3 & a = \text{left} \\ 1/6 & a = \text{right} \\ 1/6 & a = \text{eat} \end{cases}$$

Q(s,a)	left	right	eat
<b>b</b>	0	0	0
<b>c</b>	0	0	0
<b>d</b>	0.1	0	0
<b>e</b>	0	0	0

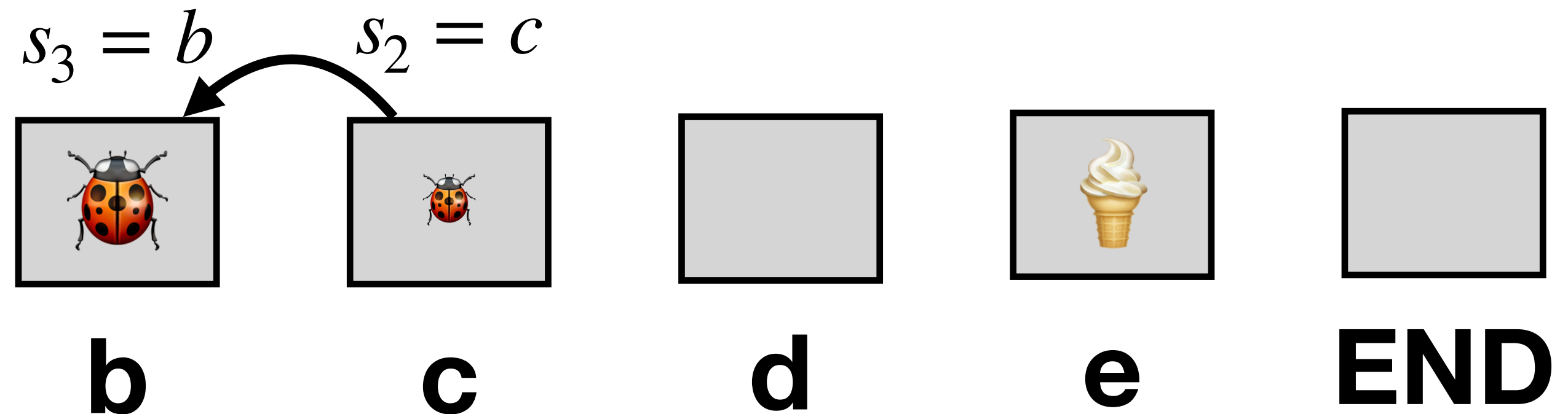
**What happened here:** Even though we did not update the estimates of the Q-function for the state  $c$ , the policy changed. Recall that we break ties (we have the preference  $\text{eat} < \text{right} < \text{left}$  and recall how we define greedy and  $\epsilon$ -greedy policies.

Say, it is  $a_3 = \text{right}$ .

# Running Example ( $t = 2$ )

Let's run SARSA on our running example ( $\gamma = 0.5$ ):

We will use  $\varepsilon_t = 1/t$ .  
 $t = 2, \varepsilon = 0.5, \alpha = 0.1$



We have:  $r_2 = 1$  and  $s_3 = b$

We now update the Q-function:

$$Q(c, \text{left}) := 0 + 0.1 (1 + 0.5 \cdot 0 - 0) = 0.1$$

$$Q(s_t, a_t) := Q(s_t, a_t) + \alpha (r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$$

Q(s,a)	left	right	eat
<b>b</b>	0	0	0
<b>c</b>	0.1	0	0
<b>d</b>	0.1	0	0
<b>e</b>	0	0	0

# Running Example ( $t = 3$ )

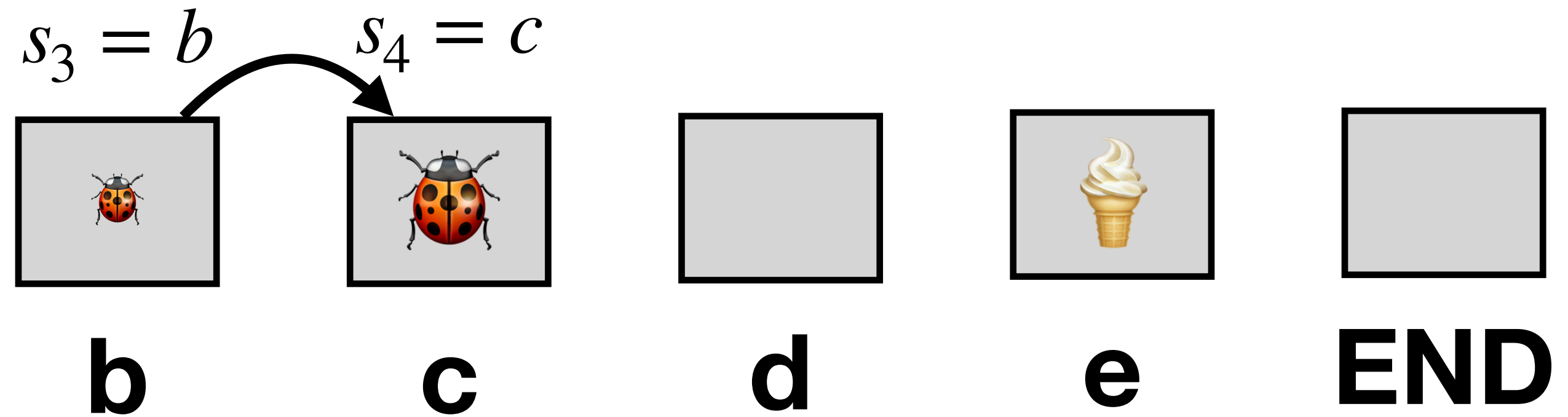
Let's run SARSA on our running example ( $\gamma = 0.5$ ):

We will use  $\varepsilon_t = 1/t$ .

$t = 1, \varepsilon = 1, \alpha = 0.1$

**We take the action  $a_3 = \text{right}$**

**We observe:  $r_3 = 1$  and  $s_4 = c$ .**



Q(s,a)	<i>left</i>	<i>right</i>	<i>eat</i>
<i>b</i>	0	0	0
<i>c</i>	0.1	0	0
<i>d</i>	0.1	0	0
<i>e</i>	0	0	0

# Running Example ( $t = 3$ )

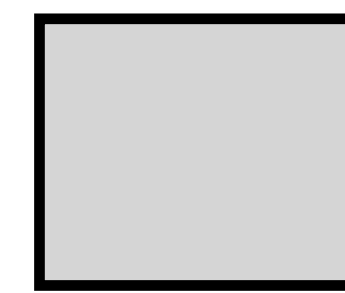
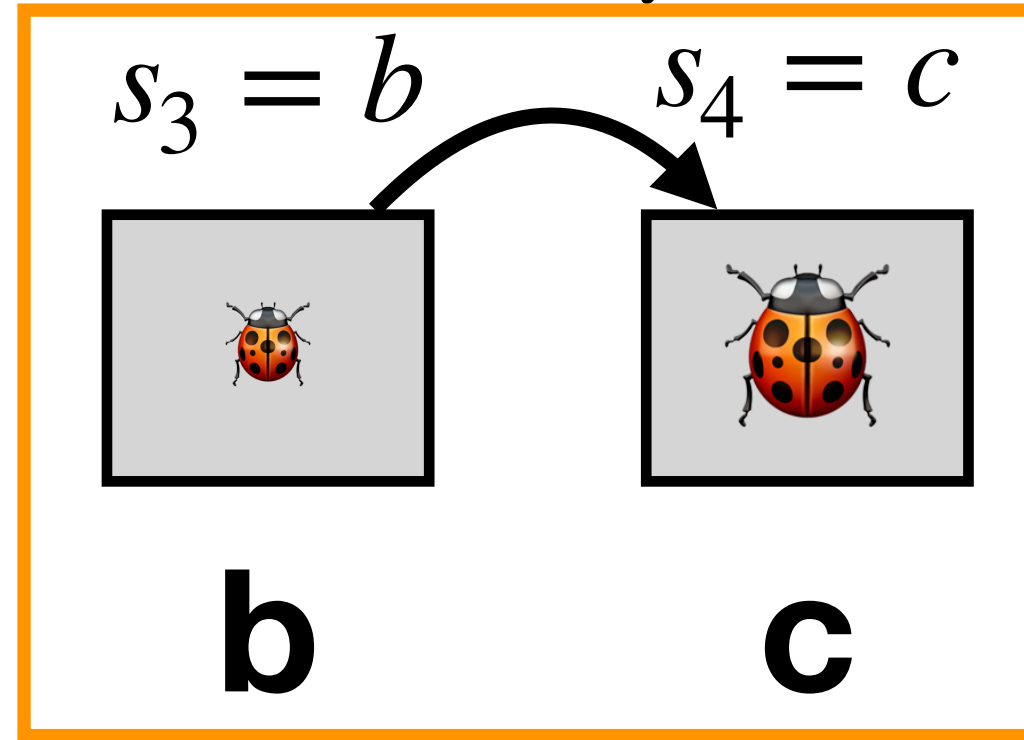
Let's run SARSA on our running example ( $\gamma = 0.5$ ):

We will use  $\epsilon_t = 1/t$ .

$t = 1, \epsilon = 1, \alpha = 0.1$

We take the action  $a_3 = \text{right}$

We observe:  $r_3 = 1$  and  $s_4 = c$ .



**d**



**e**



**END**

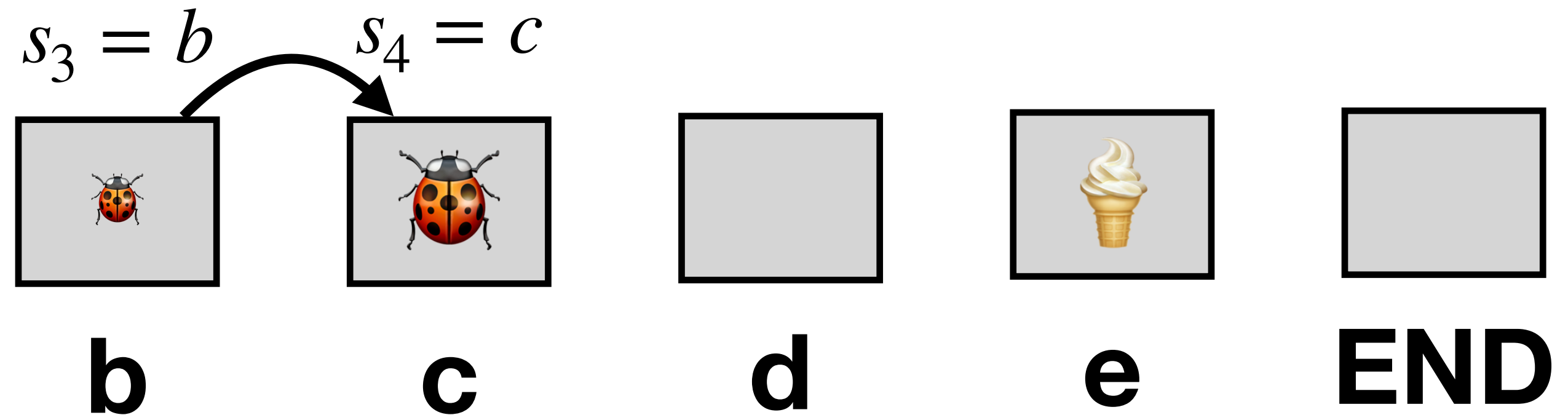
Q(s,a)	left	right	eat
<b>b</b>	0	0	0
<b>c</b>	0.1	0	0
<b>d</b>	0.1	0	0
<b>e</b>	0	0	0

# Running Example ( $t = 3$ )

Let's run SARSA on our running example ( $\gamma = 0.5$ ):

We will use  $\varepsilon_t = 1/t$ .  
 $t = 1, \varepsilon = 1, \alpha = 0.1$

We have:  $r_3 = 1$  and  $s_4 = c$



We sample  $a_4$  (we are not taking it yet)

$$\pi_1(a | c) = \begin{cases} 7/9 & a = \text{left} \\ 1/9 & a = \text{right} \\ 1/9 & a = \text{eat} \end{cases}$$

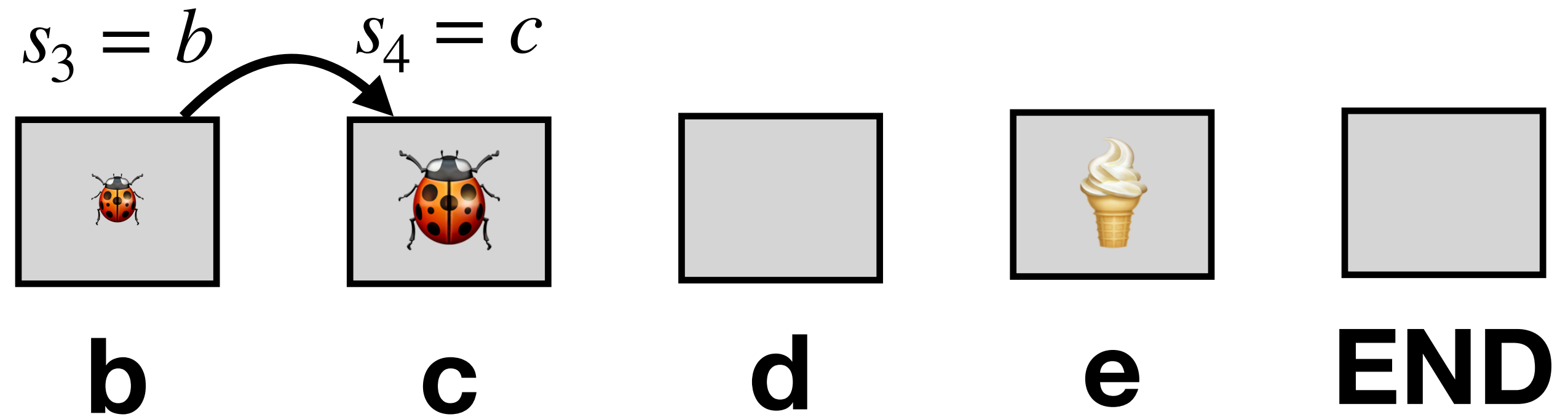
Say, it is  $a_4 = \text{left}$ .

Q(s,a)	left	right	eat
<b>b</b>	0	0	0
<b>c</b>	0.1	0	0
<b>d</b>	0.1	0	0
<b>e</b>	0	0	0

# Running Example ( $t = 3$ )

Let's run SARSA on our running example ( $\gamma = 0.5$ ):

We will use  $\epsilon_t = 1/t$ .  
 $t = 1, \epsilon = 1, \alpha = 0.1$



**We have:**  $r_3 = 1$  and  $s_4 = c$

**We sample**  $a_4$  (we are not taking it yet)

$$\pi_1(a | c) = \begin{cases} 7/9 & a = \text{left} \\ 1/9 & a = \text{right} \\ 1/9 & a = \text{eat} \end{cases}$$

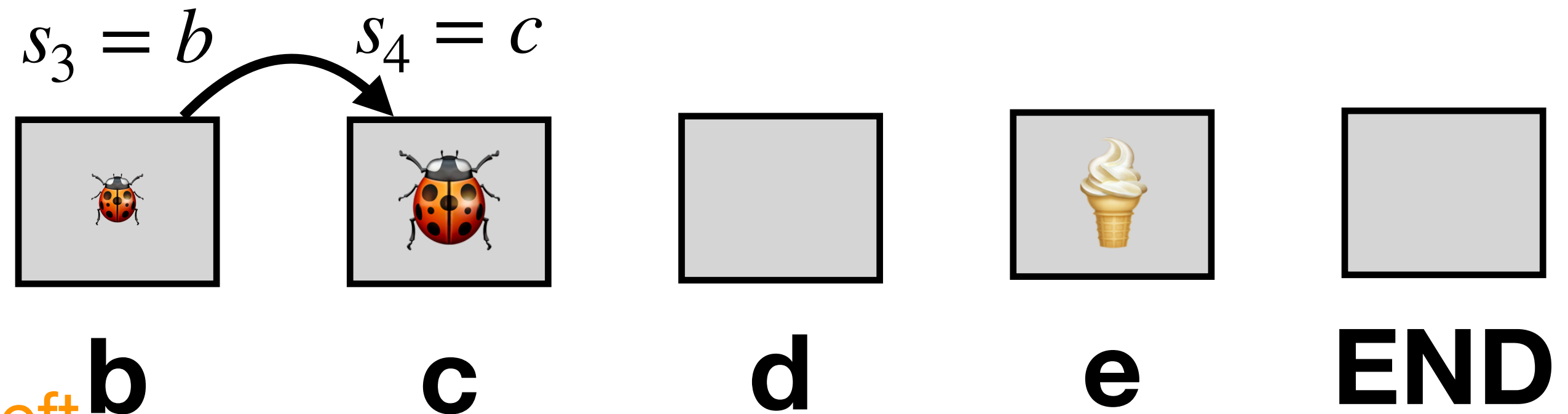
Say, it is  $a_4 = \text{left}$ .

Q(s,a)	left	right	eat
<b>b</b>	0	0	0
<b>c</b>	0.1	0	0
<b>d</b>	0.1	0	0
<b>e</b>	0	0	0

# Running Example ( $t = 3$ )

Let's run SARSA on our running example ( $\gamma = 0.5$ ):

We will use  $\epsilon_t = 1/t$ .  
 $t = 1, \epsilon = 1, \alpha = 0.1$



We have:  $r_3 = 1$  and  $s_4 = c$ ,  $a_4 = \text{left}$

We now update the Q-function:

$$Q(b, \text{right}) := 0 + 0.1 \cdot (1 + 0.5 \cdot 0.1 - 0.1) = 0.095$$

$$Q(s_t, a_t) := Q(s_t, a_t) + \alpha (r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$$

Q(s,a)	left	right	eat
<b>b</b>	0	0.095	0
<b>c</b>	0.1	0	0
<b>d</b>	0.1	0	0
<b>e</b>	0	0	0

AND SO ON....

# Note: Breaking Ties

**It is usually suggested as a good idea to break ties randomly.**

*Indeed, as we saw in our example, without tie breaking our Q-values were preferring some actions in states we have not even visited yet, just because of the arbitrary tie breaking.*

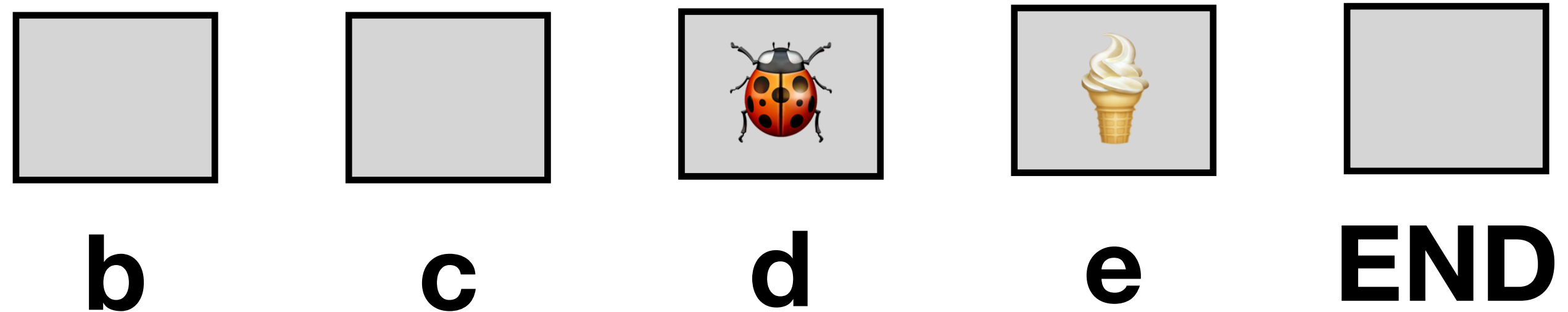
Let us rerun the example where we define the greedy policy with random tie breaking and  $\varepsilon$ -greedy policy as:

$$\pi_{\varepsilon}(a | s) = (1 - \varepsilon) \cdot \pi_{\text{greedy}}(a | s) + \frac{\varepsilon}{|A|}.$$

Note: We will not be showing all details of the updates in the next slides (that would be redundant to what we already saw). Focus mostly on the  $\varepsilon$ -policies.

# Running Example (Initialization)

We will use  $\varepsilon_t = 1/t$ .  
 $t = 1, \varepsilon = 1, \alpha = 0.1$



**With random tie-breaking:**

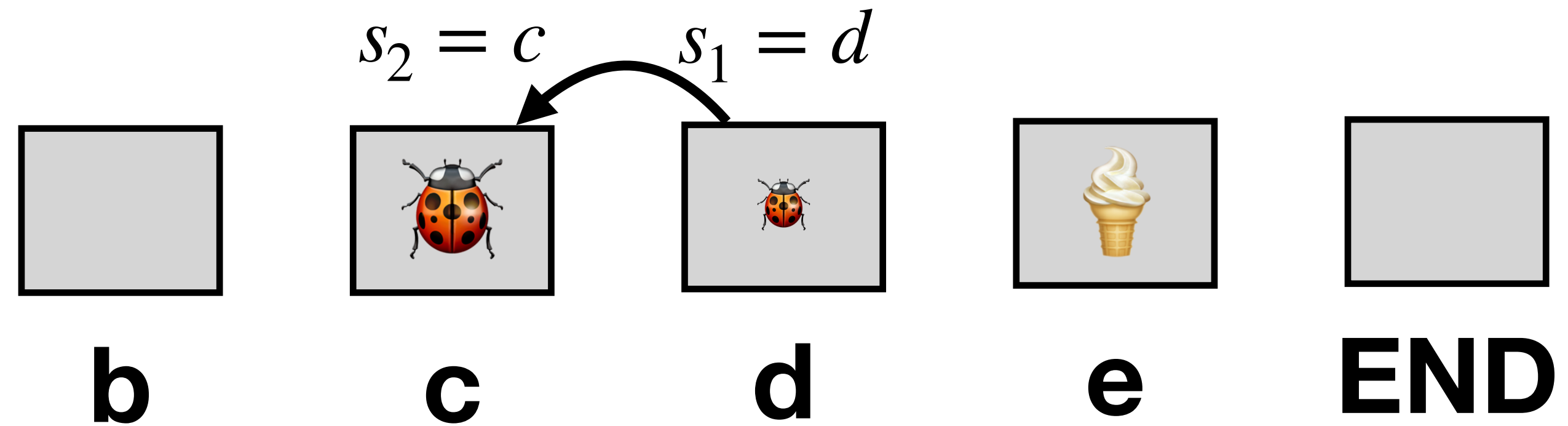
$$a_1 \sim \pi(a | d) = \begin{cases} 1/3 & a = \text{left} \\ 1/3 & a = \text{right} \\ 1/3 & a = \text{eat} \end{cases}$$

**Without random tie-breaking:**

$$a_1 \sim \pi(a | d) = \begin{cases} 1/3 & a = \text{left} \\ 1/3 & a = \text{right} \\ 1/3 & a = \text{eat} \end{cases}$$

<b>Q(s,a)</b>	<b>left</b>	<b>right</b>	<b>eat</b>
<b>b</b>	0	0	0
<b>c</b>	0	0	0
<b>d</b>	0	0	0
<b>e</b>	0	0	0

# Running Example ( $t = 1$ )



**With random tie-breaking:**

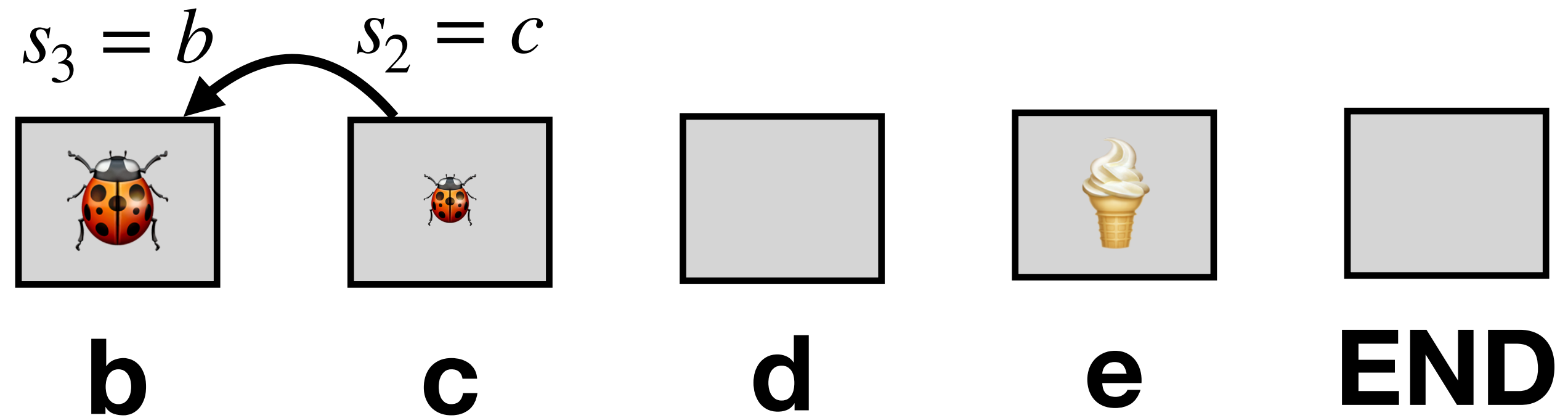
$$a_2 \sim \pi(a | c) = \begin{cases} 1/3 & a = \text{left} \\ 1/3 & a = \text{right} \\ 1/3 & a = \text{eat} \end{cases}$$

**Without random tie-breaking:**

$$a_2 \sim \pi(a | c) = \begin{cases} 1/3 & a = \text{left} \\ 1/3 & a = \text{right} \\ 1/3 & a = \text{eat} \end{cases}$$

Q(s,a)	<i>left</i>	<i>right</i>	<i>eat</i>
<b><i>b</i></b>	0	0	0
<b><i>c</i></b>	0	0	0
<b><i>d</i></b>	0	0	0
<b><i>e</i></b>	0	0	0

# Running Example ( $t = 2$ )



**With random tie-breaking:**

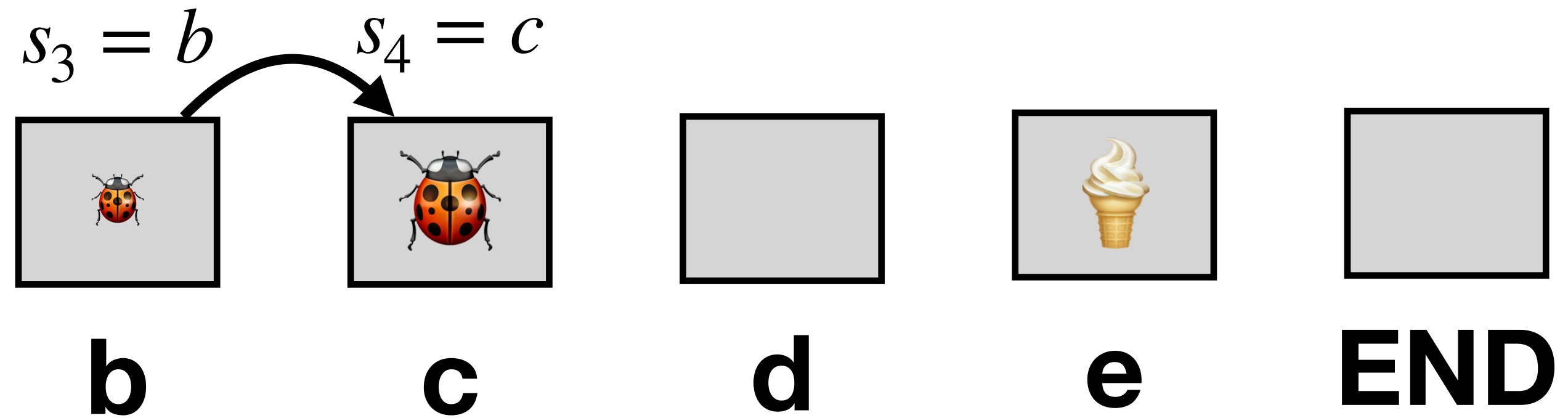
$$a_3 \sim \pi(a | b) = \begin{cases} 1/3 & a = \text{left} \\ 1/3 & a = \text{right} \\ 1/3 & a = \text{eat} \end{cases}$$

**Without random tie-breaking:**

$$a_3 \sim \pi(a | b) = \begin{cases} 2/3 & a = \text{left} \\ 1/6 & a = \text{right} \\ 1/6 & a = \text{eat} \end{cases}$$

$Q(s,a)$	<i>left</i>	<i>right</i>	<i>eat</i>
<b><i>b</i></b>	0	0	0
<b><i>c</i></b>	0	0	0
<b><i>d</i></b>	0.1	0	0
<b><i>e</i></b>	0	0	0

# Running Example ( $t = 3$ )



**With random tie-breaking:**

$$a_4 \sim \pi(a | c) = \begin{cases} 7/9 & a = \text{left} \\ 1/9 & a = \text{right} \\ 1/9 & a = \text{eat} \end{cases}$$

**Without random tie-breaking:**

$$a_4 \sim \pi(a | c) = \begin{cases} 7/9 & a = \text{left} \\ 1/9 & a = \text{right} \\ 1/9 & a = \text{eat} \end{cases}$$

Q(s,a)	<i>left</i>	<i>right</i>	<i>eat</i>
<i>b</i>	0	0	0
<i>c</i>	<b>0.1</b>	0	0
<i>d</i>	0.1	0	0
<i>e</i>	0	0	0

**AND SO ON....**

# Note: Optimistic Initialization

What happens if we initialize the Q values differently?

For instance, what would happen if we started with:

$Q(s,a)$	<i>left</i>	<i>right</i>	<i>eat</i>
<i>b</i>	5	5	5
<i>c</i>	5	5	5
<i>d</i>	5	5	5
<i>e</i>	5	5	5

**Answer:** The agent would be “exploring” more than with the initialization we used.

This is a general property. If you want to promote exploration, initialize higher estimate of the Q function.

# Convergence (SARSA)

- SARSA converges to the optimal state-value function  $Q^*$  if the following conditions are satisfied:
  1. The sequence of policies  $\pi_t$  satisfies the GLIE conditions (enough to have  $\epsilon_t = 1/t$ ).
  2. Step-sizes satisfy the Robbins-Monro conditions:

$$\sum_{t=1}^{\infty} \alpha_t = \infty,$$

$$\sum_{t=1}^{\infty} \alpha_t^2 < \infty.$$

# Note: Why “SARSA”?

**Why the name?** Because of the update rule

$$Q(s_t, a_t) := Q(s_t, a_t) + \alpha (r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$$

which uses the tuple  $s_t, a_t, r_t, s_{t+1}, a_{t+1} \sim \text{s a r s a}$ .

# Q-Learning (1/2)

- The **Optimal Bellman Equation** (we have not talked about it yet but it is similar to what we already saw):

$$Q^*(s, a) = R(s, a) + \gamma \sum_{s_{t+1} \in \mathcal{S}} P(s_{t+1} | s_t, a_t) \cdot \max_{a_{t+1} \in \mathcal{A}} Q^*(s_{t+1}, a_{t+1}).$$

$$\mathbb{E} \left[ \max_{a_{t+1} \in \mathcal{A}} Q^*(X_{t+1}, a_{t+1}) \mid X_t = s_t, A_t = a_t \right]$$

- Q-Learning update rule:

$$Q(s_t, a_t) := Q(s_t, a_t) + \alpha \left( r_t + \gamma \max_{a \in \mathcal{A}} Q(s_{t+1}, a) - Q(s_t, a_t) \right)$$

# Q-Learning (2/2)

- 1. Initialize:** set  $\pi$  to be some  $\varepsilon$ -greedy policy, set  $t = 1$
- 2. Observe the initial state**  $s_1$
- 3. While**  $s_t$  is not a terminal state:
  - 1. Take** action  $a_t \sim \pi(s_t)$  and observe  $r_t, s_{t+1}$ .
  - 2.** 
$$Q(s_t, a_t) := Q(s_t, a_t) + \alpha \left( r_t + \gamma \max_{a \in A} Q(s_{t+1}, a) - Q(s_t, a_t) \right)$$
  - 3.**  $\pi := \varepsilon$ -greedy( $Q$ )
  - 4.** Set  $t := t + 1$ . Update  $\varepsilon, \alpha$  /\* see next slides \*/

# Convergence (Q-Learning)

- For convergence of the state-value Q-function, we need only the Robbins-Monro conditions + every state-action pair needs to be visited infinitely often (with probability 1).
- For convergence of the policy to the optimal policy, we need GLIE (i.e. it needs to also be greedy in the limit...).

# On-Policy and Off-Policy Methods

- **On-policy methods:** samples must be from the policy that we are learning. **Example:** SARSA, MC Policy Iteration.
- **Off-policy methods:** samples do not have to be from the policy that we are learning. **Example:** Q-Learning.

**END OF SLIDES**