Value Based - Part 1

Deep Reinforcement Learning

Mohammad Hossein Rohban, Ph.D.

Courtesy: Some slides are adopted from CS 285 Berkeley, and CS 234 Stanford, and Pieter Abbeel's compact series on RL.

RIML Lab, CE Department, Sharif University of Technology

Spring 2026

Value Function

- $V^*(s)$ = expected utility starting in s, and acting optimally in all subsequent actions.

$$V^*(s) = \max_{a_i' s} \mathbb{E}\left(\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) \middle| s_0 = s\right)$$

Value Iteration

- $V_0^*(s)$ = optimal value for state s when H=0
  - ☑ $V_0^*(s) = 0 \forall s$
- $V_1^*(s)$ = optimal value for state s when H=1
  - ☑ $V_1^*(s) = \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_0^*(s'))$
- $V_2^*(s)$ = optimal value for state s when H=2
  - ☑ $V_2^*(s) = \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_1^*(s'))$
- $V_k^*(s)$ = optimal value for state s when H = k
  - ☑ $V_k^*(s) = \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_{k-1}^*(s'))$

Value Iteration

Algorithm:

Start with $V_0^*(s) = 0$ **for all s.**

For k = 1, … , H:

For all states s in S:

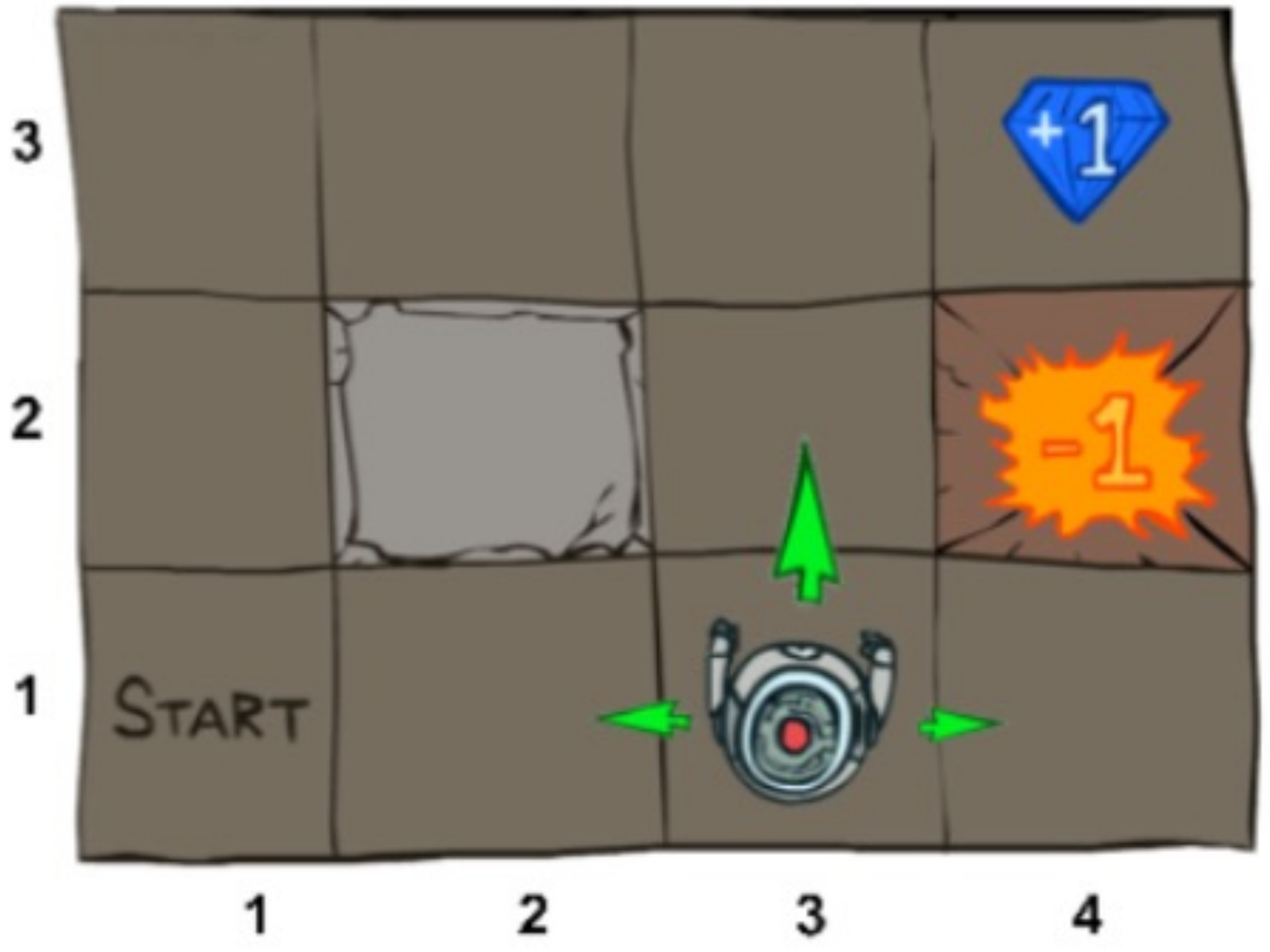
$$V_k^*(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_{k-1}^*(s'))$$

$$\pi_k^*(s) \leftarrow \mathrm{arg}\max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_{k-1}^*(s'))$$

☑ This is called a value update or Bellman update/back-up

Value Iteration

$$V_0(s) \leftarrow 0$$



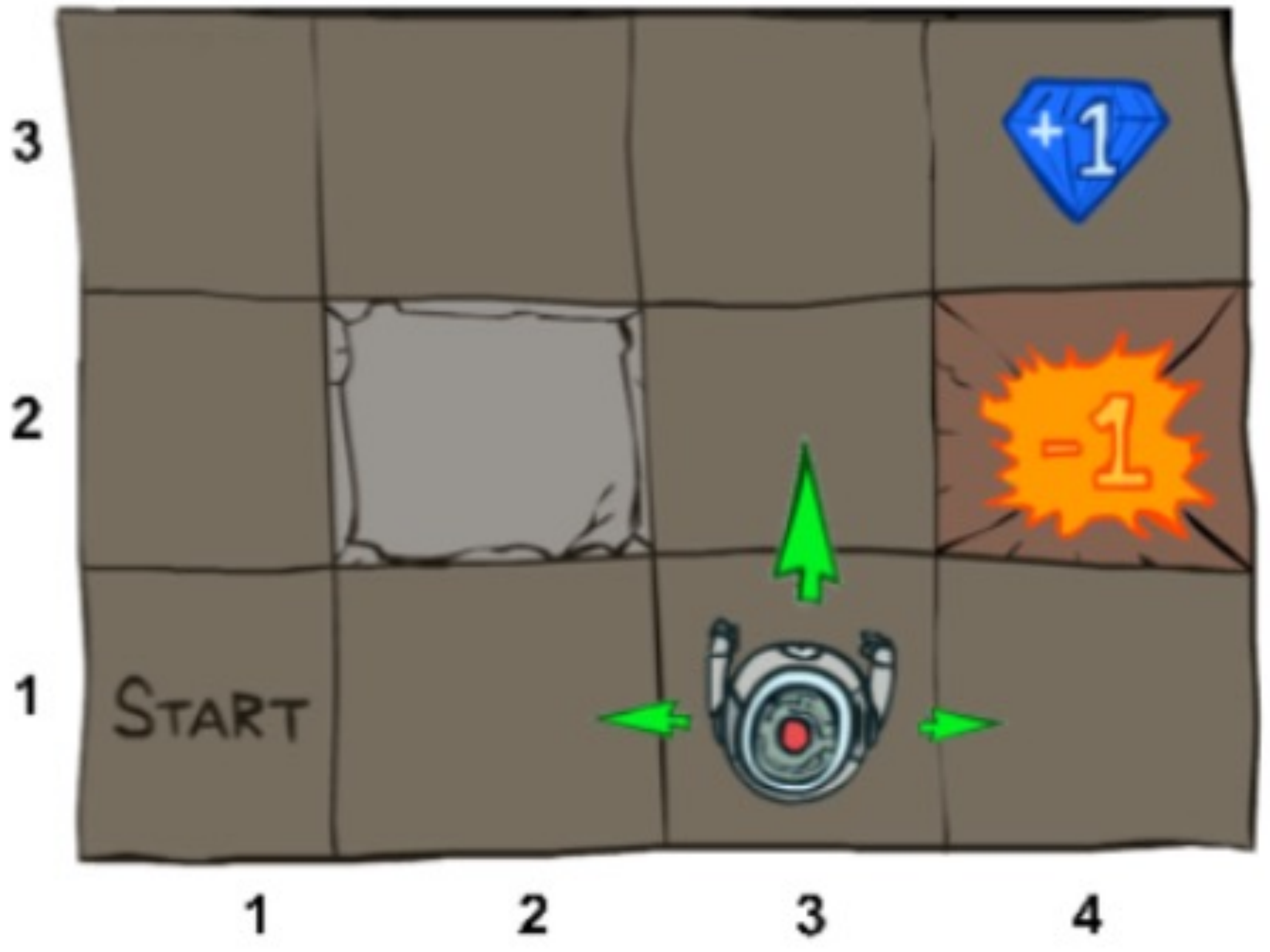
Noise = 0.2
Discount = 0.9

k = 0

Value Iteration

$$V_1(s) = \max_a \sum_{s'} P(s'|s,a)\left(R(s,a,s') + \gamma V_0(s')\right)$$

k = 0

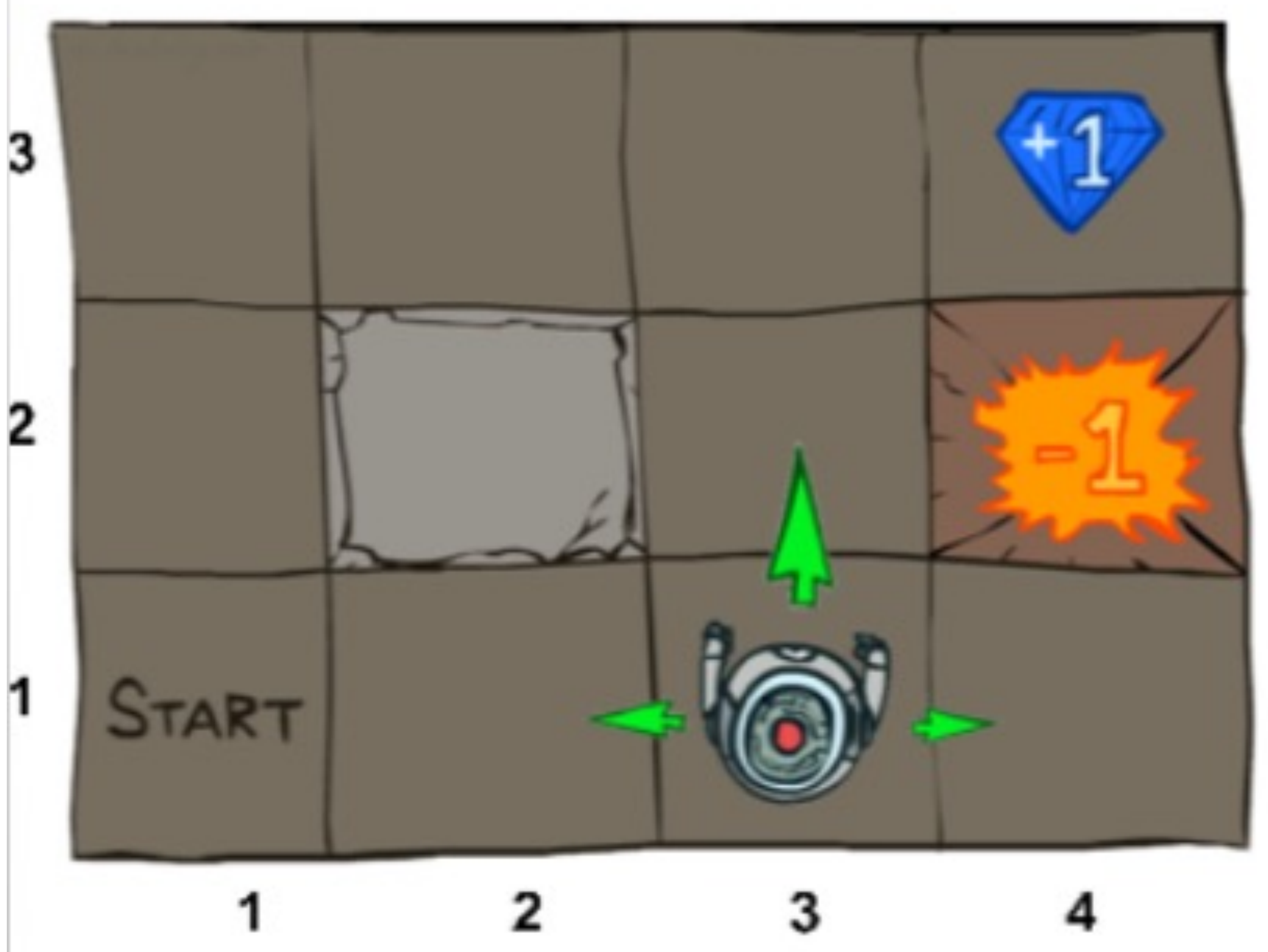


Noise = 0.2

Discount = 0.9

Value Iteration

$$V_2(s) = \max_a \sum_{s'} P(s'|s,a)\left(R(s,a,s') + \gamma V_1(s')\right)$$



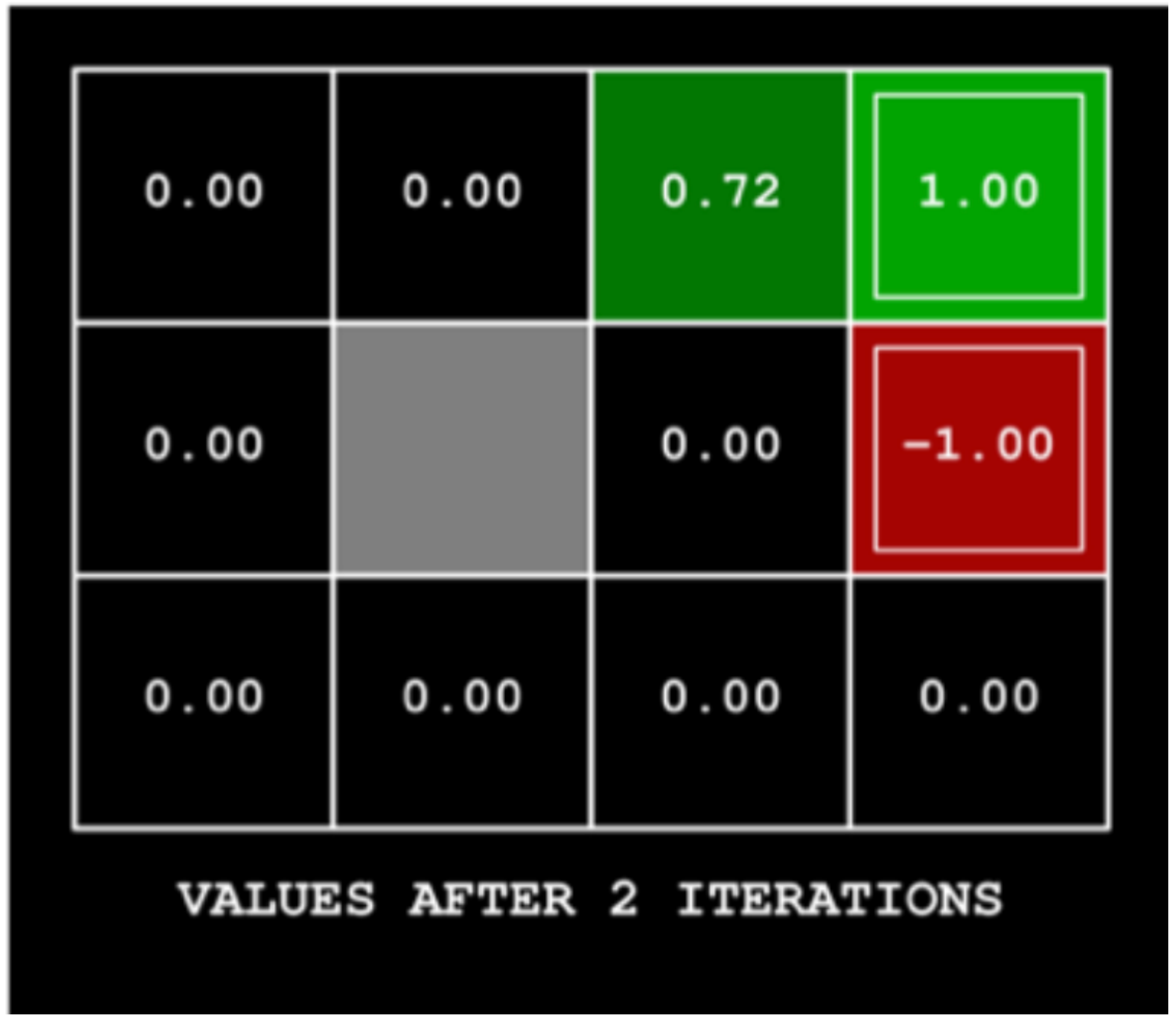
Noise = 0.2
Discount = 0.9

k = 1

Value Iteration

$$V_2(s) = \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_1(s'))$$

k = 2

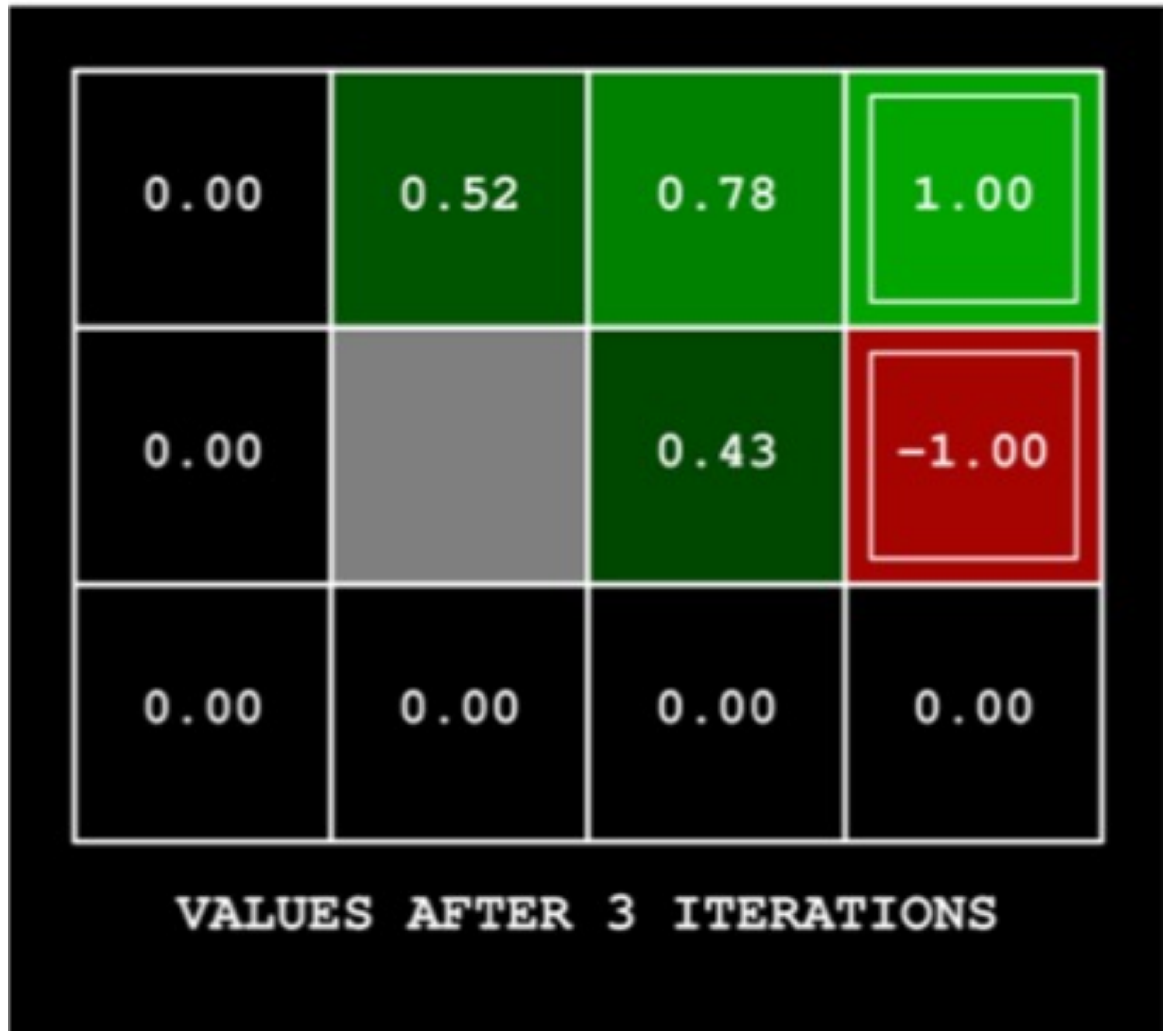


Noise = 0.2
Discount = 0.9

Value Iteration

$$V_{k+1}(s) = \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_{\mathrm{k}}(s'))$$

k = 3



Noise = 0.2
Discount = 0.9

Value Iteration

$$V_{k+1}(s) = \max_{a} \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_{k}(s'))$$

k = 4

Noise = 0.2
Discount = 0.9

Value Iteration

$$V_{k+1}(s) = \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 5

Noise = 0.2
Discount = 0.9

Value Iteration

$$V_{k+1}(s) = \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_{\mathrm{k}}(s'))$$

k = 6

Noise = 0.2
Discount = 0.9

Value Iteration

$$V_{k+1}(s) = \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 7

Noise = 0.2
Discount = 0.9

Value Iteration

$$V_{k+1}(s) = \max_{a} \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 8

Noise = 0.2
Discount = 0.9

Value Iteration

$$V_{k+1}(s) = \max_{a} \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_{k}(s'))$$

k = 9

Noise = 0.2
Discount = 0.9

Value Iteration

$$V_{k+1}(s) = \max_{a} \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_{\mathrm{k}}(s'))$$

k = 10

Noise = 0.2
Discount = 0.9

Value Iteration

$$V_{k+1}(s) = \max_{a} \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_{\mathrm{k}}(s'))$$

k = 11

Noise = 0.2
Discount = 0.9

Value Iteration

$$V_{k+1}(s) = \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 12

Noise = 0.2
Discount = 0.9

Value Iteration

$$V_{k+1}(s) = \max_{a} \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_{\mathrm{k}}(s'))$$

k = 100

Noise = 0.2
Discount = 0.9

Q-Values

- $Q^*(s,a)$ = expected utility starting in s, taking action a, and (thereafter) acting optimally

$$V^*(s) = \max_{a'} Q^*(s,a')$$

- Bellman Equation:

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

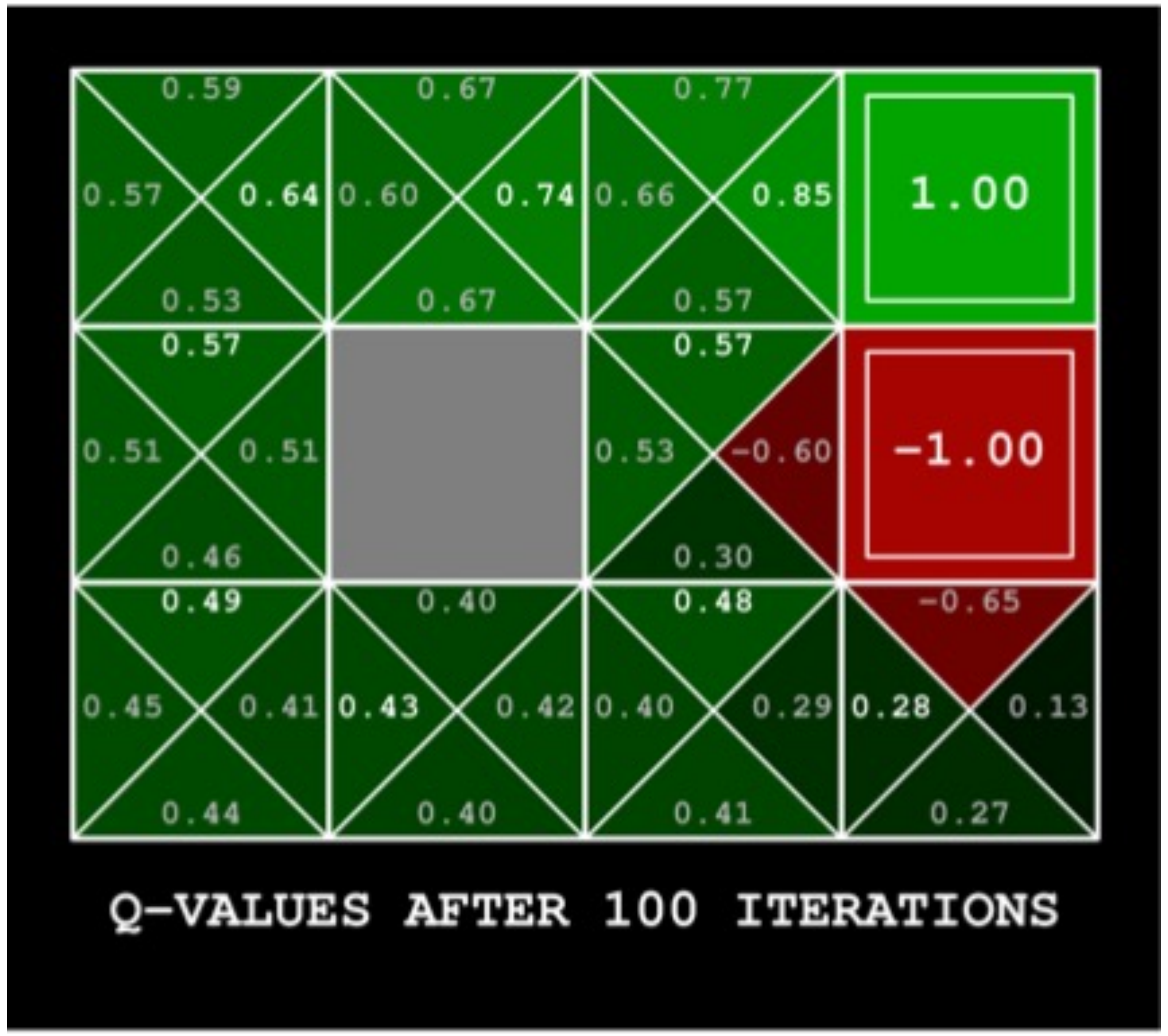
- Q-value Iteration:

$$Q^*_{k+1}(s,a) \leftarrow \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma \max_{a'} Q^*_k(s',a'))$$

Q-Value Iteration

$$Q_{k+1}^{*}(s,a) \leftarrow \sum_{s'} P(s'|s,a)\left(R(s,a,s') + \gamma \max_{a'} Q_k^{*}(s',a')\right)$$

k = 100



Noise = 0.2
Discount = 0.9

Policy Evaluation

- Recall value iteration:
  - ☑ $V_k^*(s) \leftarrow max_a \sum_{s'} P(s'|s,a)\big(R(s,a,s') + \gamma V_{k-1}^*(s')\big)$
- Policy evaluation for a given $\pi(s)$:
  - ☑ $V_k^\pi(s) \leftarrow \sum_{s'} P(s'|s,\pi(s))(R(s,\pi(s),s') + \gamma V_{k-1}^\pi(s'))$
- At convergence:
  - ☑ $\forall s V^\pi(s) \leftarrow \sum_{s'} P(s'|s,\pi(s))(R(s,\pi(s),s') + \gamma V^\pi(s))$

Policy Iteration

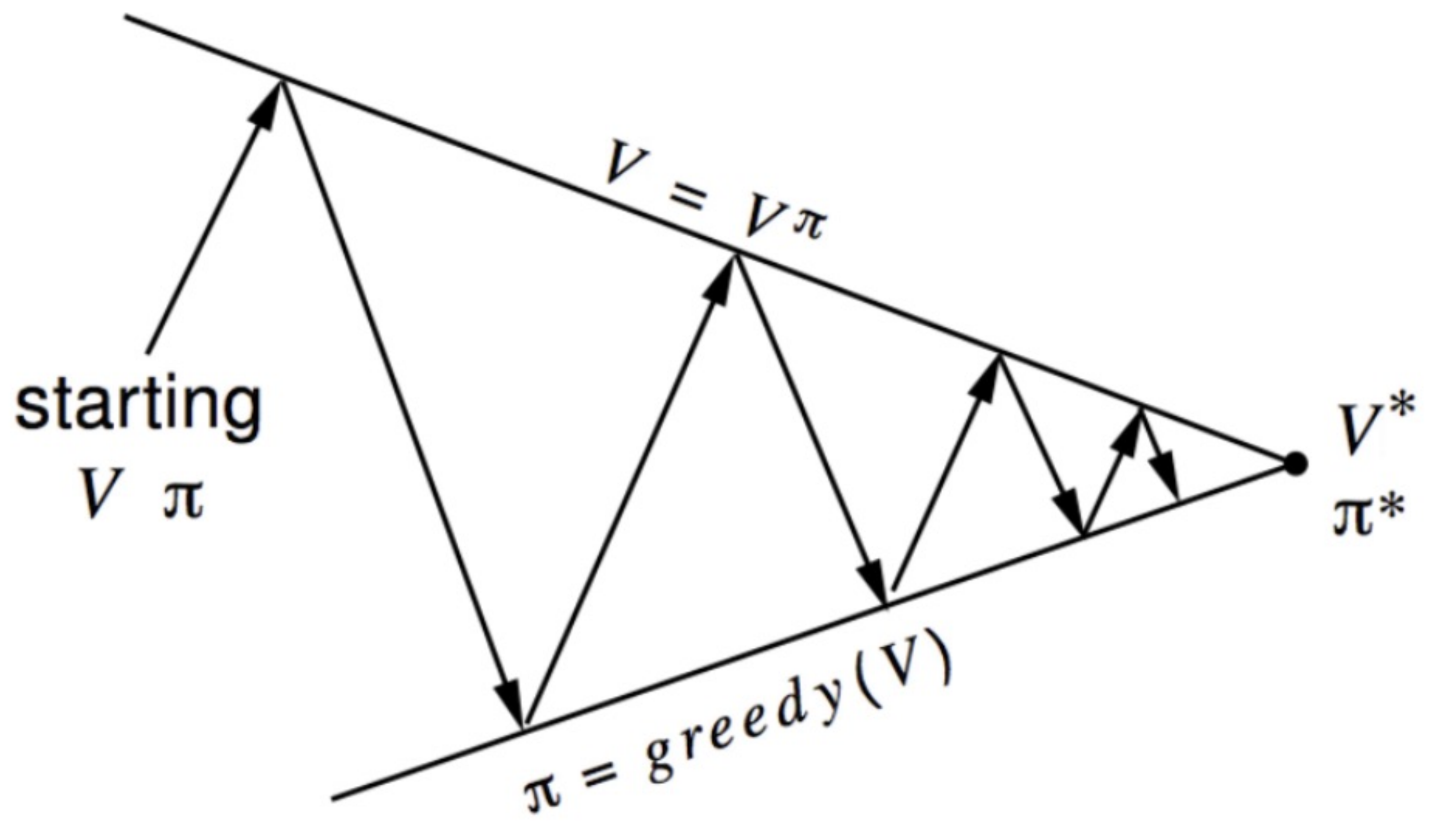
- One iteration of policy iteration
  - Policy evaluation for current policy π_k. Iterate until convergence

$$V_{i+1}^{\pi_k}(s) \leftarrow \sum_{s'} P(s'|s, \pi_k(s))[R(s, \pi_k(s), s') + \gamma V_i^{\pi_k}(s')]$$

  - Policy improvement: find the best action according to one-step look-ahead

$$\pi_{k+1}(s) \leftarrow \mathrm{arg}\max_a \sum_{s'} P(s'|s, a)[R(s, a, s') + \gamma V^{\pi_k}(s')]$$

- Repeat until policy converges
  - At convergence: optimal policy; and converges faster than value iteration under some conditions



One-step look ahead improves the policy

- Consider an alternative policy $\pi^{(1)}_{(k+1)}(t,s)$ that takes the prescribed actions in $\pi_{k+1}(s)$ only at time t = 0; and stays the same as $\pi_k(s)$ in later times.
- The value function V(s) for this new time-dependent policy is larger than or equal to V(s) for the original policy $\pi_k(s)$ for all s. Why?
- Now let $\pi^{(2)}_{(k+1)}(t,s)$, which takes the prescribed action in $\pi_{k+1}(s)$ only at times t = 0 and t = 1, and stays the same as $\pi_k(s)$ in later times.
- Similarly, V(s) gets improved for $\pi^{(2)}_{(k+1)}(t,s)$ compared to $\pi^{(1)}_{(k+1)}(t,s)$ for all s.
- Repeating this argument $\pi^{(\infty)}_{(k+1)}(t,s)$ becomes the same as $\pi_{k+1}(s)$.

An Example

- Let this be the initial policy π_0, show how policy improvement, makes this policy better.

+1	→	+1
-1	↑	-1
-1	→	-1
-1	↑	-1
-1	←	-1
-1	↑	-1
-1	→	-1

Planning vs. Learning

- Assumed to have access to the dynamics $P(s'|s,a)$.
- We don't have access to this in the real world.
- We need to estimate (or learn) the value functions.

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

Monte-Carlo Prediction

Monte Carlo Methods - Introduction

- Experience samples to learn without a model

- MC methods require only experience— sample sequences of states, actions, and rewards from actual or simulated interaction with an environment.

- We can learn with samples: episodes!

We don't have access to $P(s'|s,a)$ → Model Free Learning!

Monte-Carlo prediction

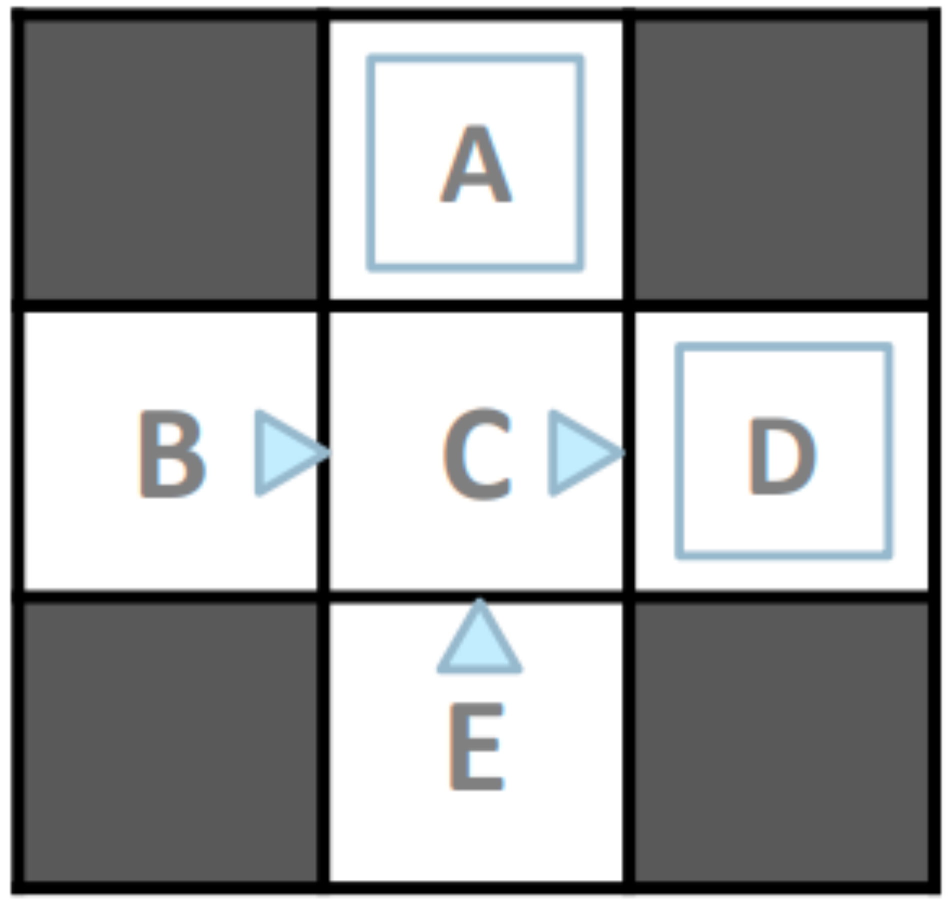
- Suppose we wish to estimate $V_\pi(s)$, the value of a state s under policy π.
- The first-visit mc method estimates $V_\pi(s)$ as the average of the returns following first visits to s.

First-visit MC prediction, for estimating $V \approx v_\pi$

```
Input: a policy π to be evaluated
Initialize:
    V(s) ∈ ℝ, arbitrarily, for all s ∈ 𝒮
    Returns(s) ← an empty list, for all s ∈ 𝒮
Loop forever (for each episode):
    Generate an episode following π: S_0, A_0, R_1, S_1, A_1, R_2, ..., S_{T-1}, A_{T-1}, R_T
     G ← 0
    Loop for each step of episode, t = T-1, T-2, ..., 0:
        G ← γG + R_{t+1}
        Unless S_t appears in S_0, S_1, ..., S_{t-1}:
            Append G to Returns(S_t)
             V(S_t) ← average(Returns(S_t))
```

Episodes: another example

Input Policy π



Assume: $\gamma = 1$

Observed Episodes (Training)

Episode 1

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 2

B, east, C, -1
C, east, D, -1
D, exit, x, +10

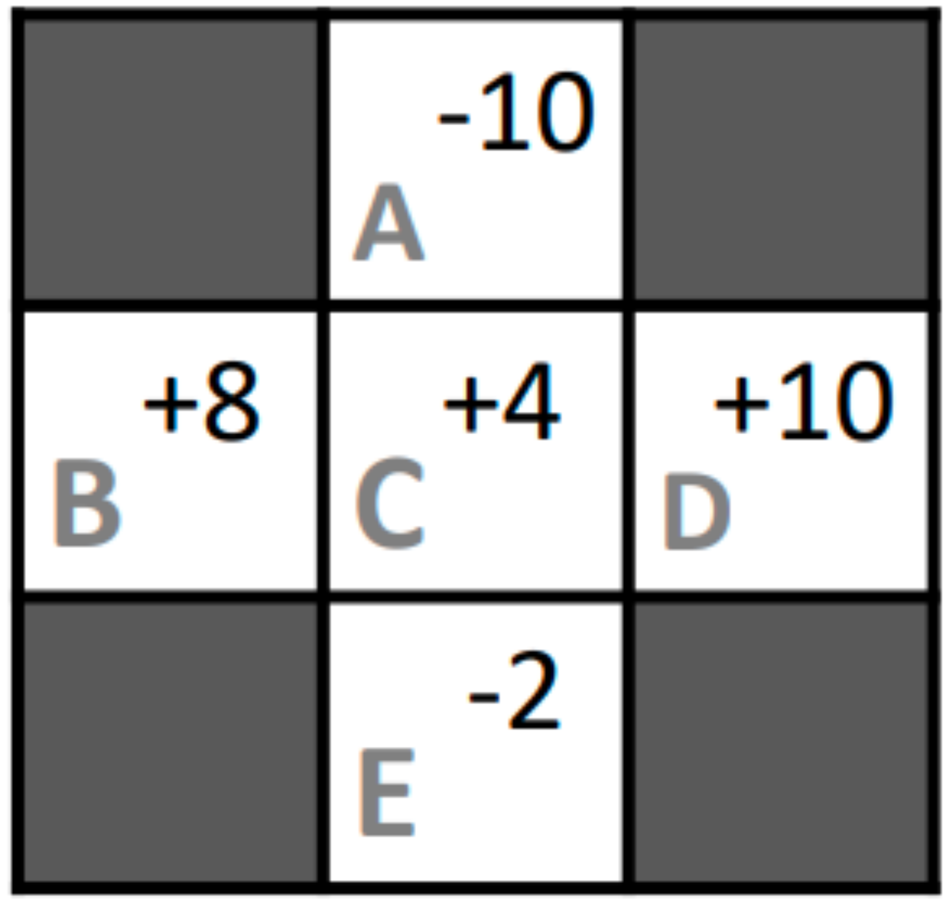
Episode 3

E, north, C, -1
C, east, D, -1
D, exit, x, +10

Episode 4

E, north, C, -1
C, east, A, -1
A, exit, x, -10

Output Values



Every Visit Monte-Carlo Policy

Every Visit Monte-Carlo Policy

Initialize $N(s) = 0, G(s) = 0 \forall s \in S$

Loop

Sample episode $i = s_{i,1}, a_{i,1}, r_{i,1}, s_{i,2}, a_{i,2}, r_{i,2}, \ldots, s_{i,T_i}$

Define $G_{i,t} = r_{i,t} + \gamma r_{i,t+1} + \gamma^2 r_{i,t+2} + \cdots \gamma^{T_i - 1} r_{i,T_i}$ **as return from time step** t **onwards in i'th episode.**

For each time step t **until** T_i **(the end of the episode** i**):**

state s **is the state visited at time step** t **in episodes** i

Increment counter of total visits: $N(s) = N(s) + 1$

Increment total return: $G(s) = G(s) + G_{i,t}$

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

Incremental Monte-Carlo Policy

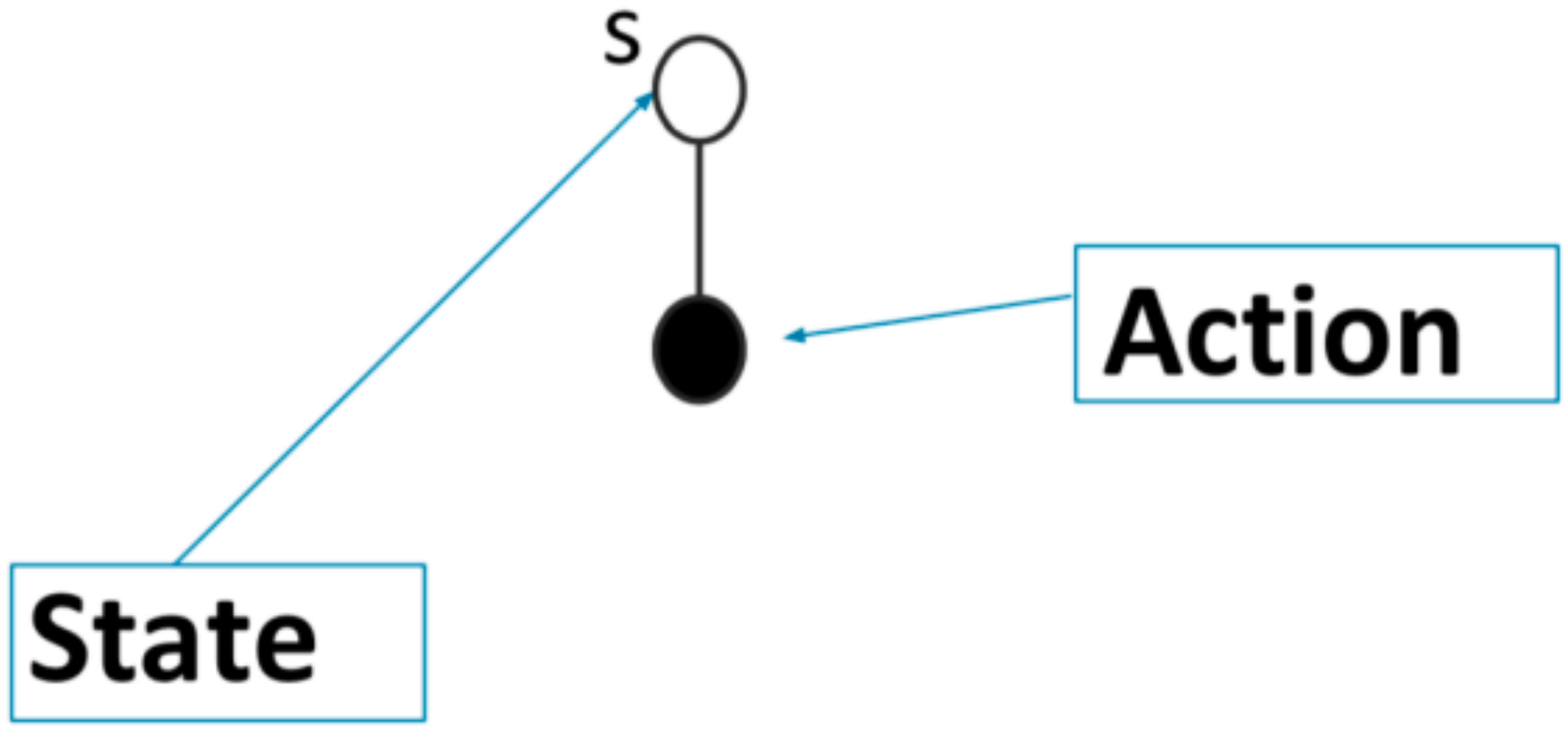
Incremental Monte-Carlo Policy

After each episode $i = s_{i,1}, a_{i,1}, r_{i,1}, s_{i,2}, a_{i,2}, r_{i,2}, \ldots$

- **Define** $G_{i,t} = r_{i,t} + \gamma r_{i,t+1} + \gamma^2 r_{i,t+2} + \cdots$ **as return from time step** t **onwards in i'th episode**
- **For state** s **visited at time step** t **in episode** i
  - **Increment counter of total visits:** $N(s) = N(s) + 1$
  - **Update estimate**

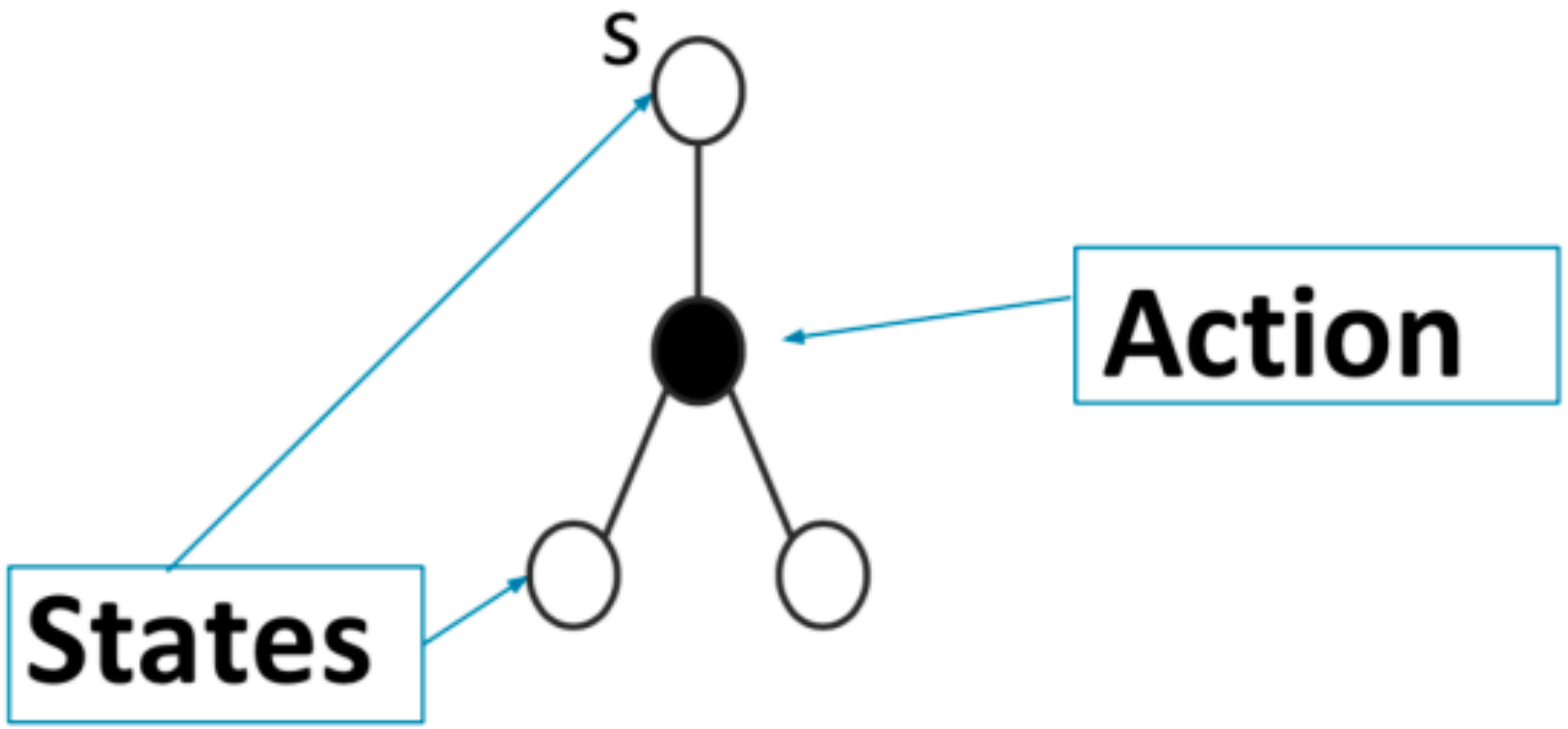
$$V^{\pi}(s) = V^{\pi}(s)\frac{N(s)-1}{N(s)} + \frac{G_{i,t}}{N(s)} = V^{\pi}(s) + \frac{1}{N(s)}\left(G_{i,t} - V^{\pi}(s)\right)$$

Policy Evaluation Diagram

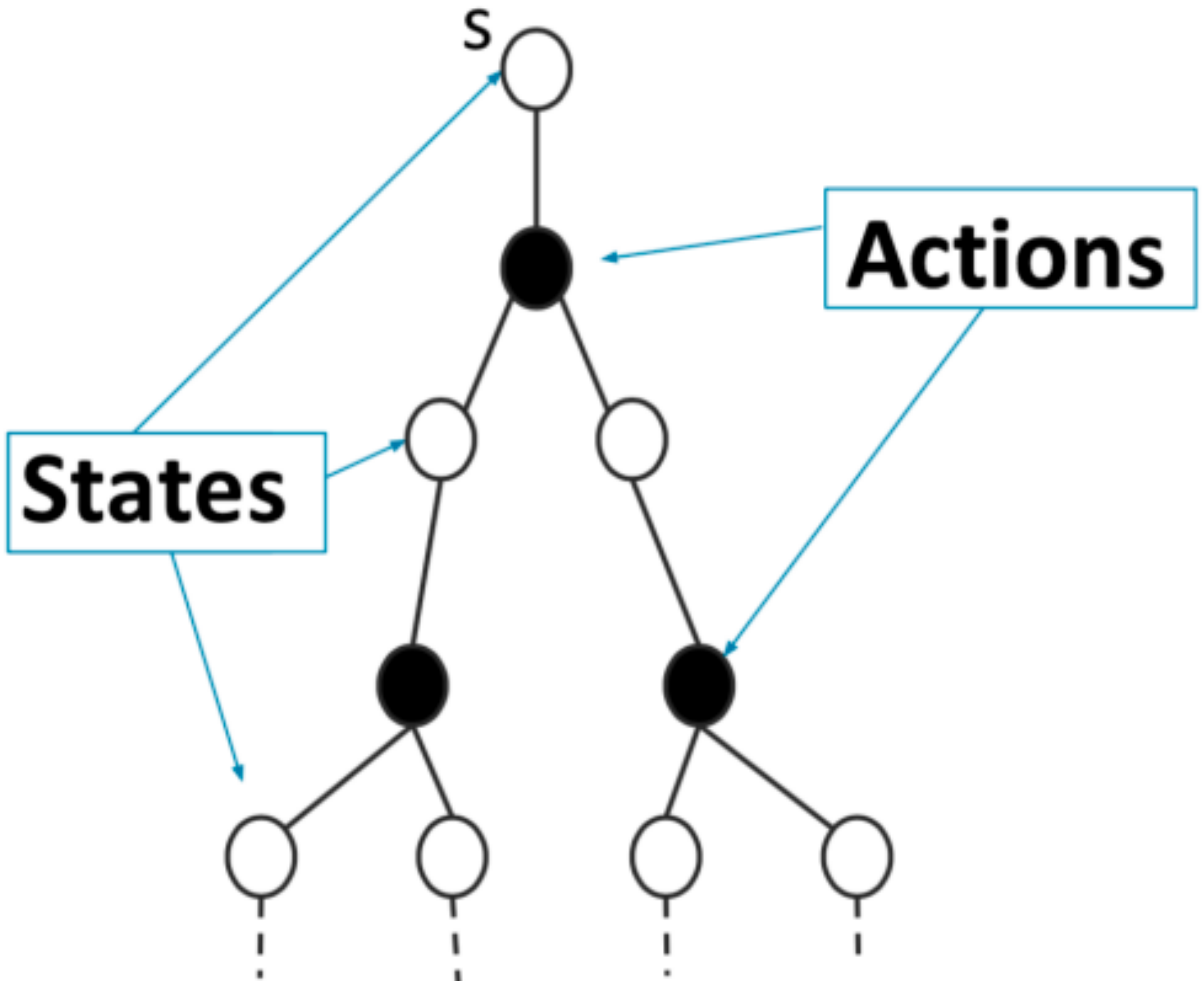


33

Policy Evaluation Diagram

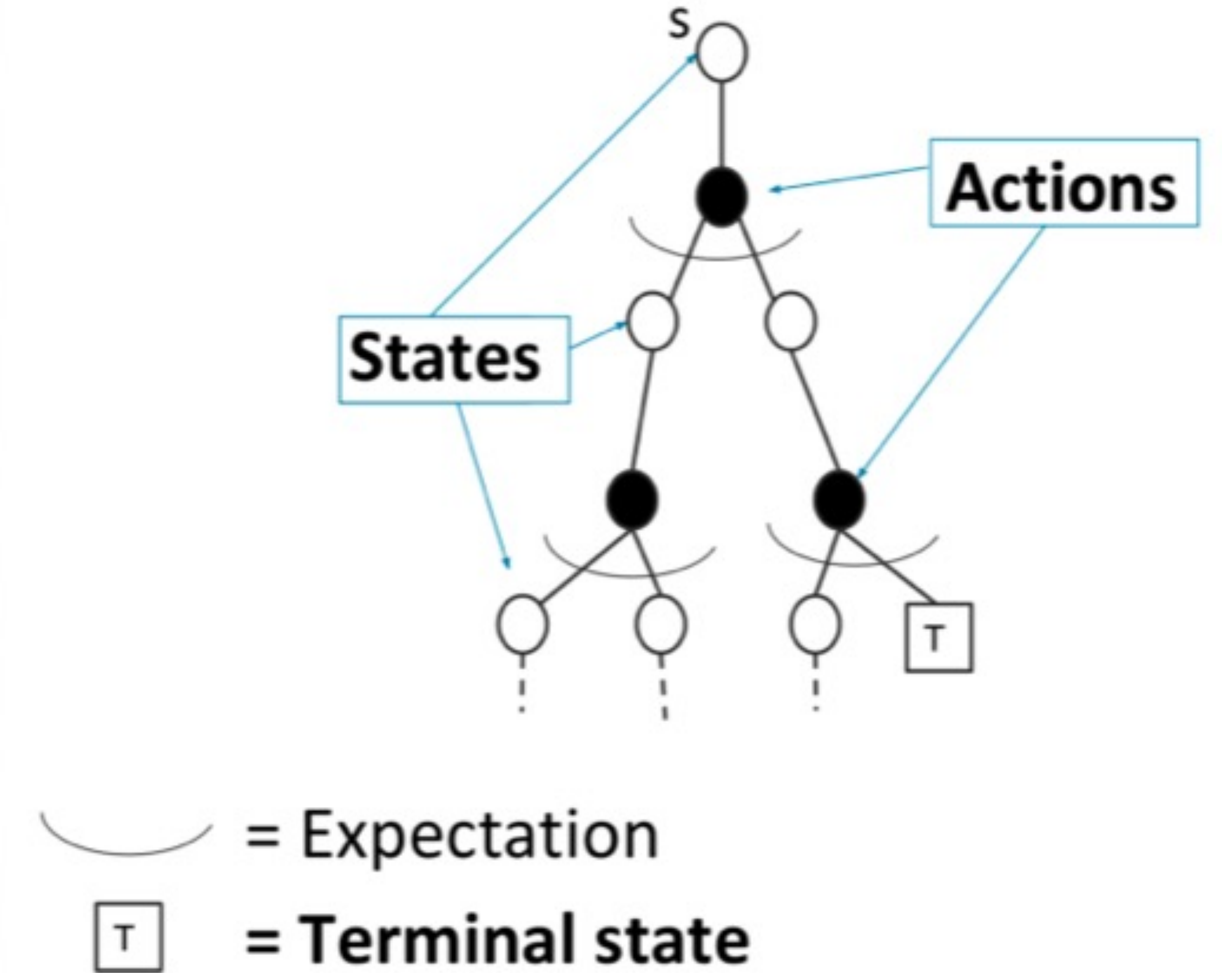


Policy Evaluation Diagram



Policy Evaluation Diagram

$$V^{\pi}(s) = V^{\pi}(s) + \alpha(G_{i,t} - V^{\pi}(s))$$



Policy Evaluation Diagram

$$V^{\pi}(s) = V^{\pi}(s) + \alpha(G_{i,t} - V^{\pi}(s))$$