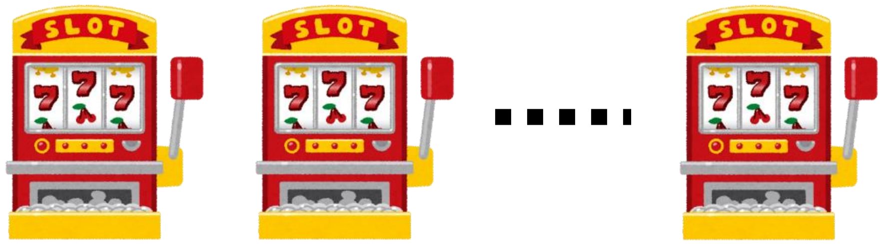
Brain-Inspired Artificial Intelligence
2: Markov Decision Process

Eiji Uchibe
Dept. of Brain Robot Interface
ATR Computational Neuroscience Labs.

Reminder: Bandit Problem

- One state, many actions
  - The goal is to find an optimal policy $\pi(a)$ that maximize the expected reward
  - The value based approach estimates the value function $Q^{\pi}(a)$
  - The policy based approach directly search the optimal policy

$R \sim p(\cdot|\, a)$



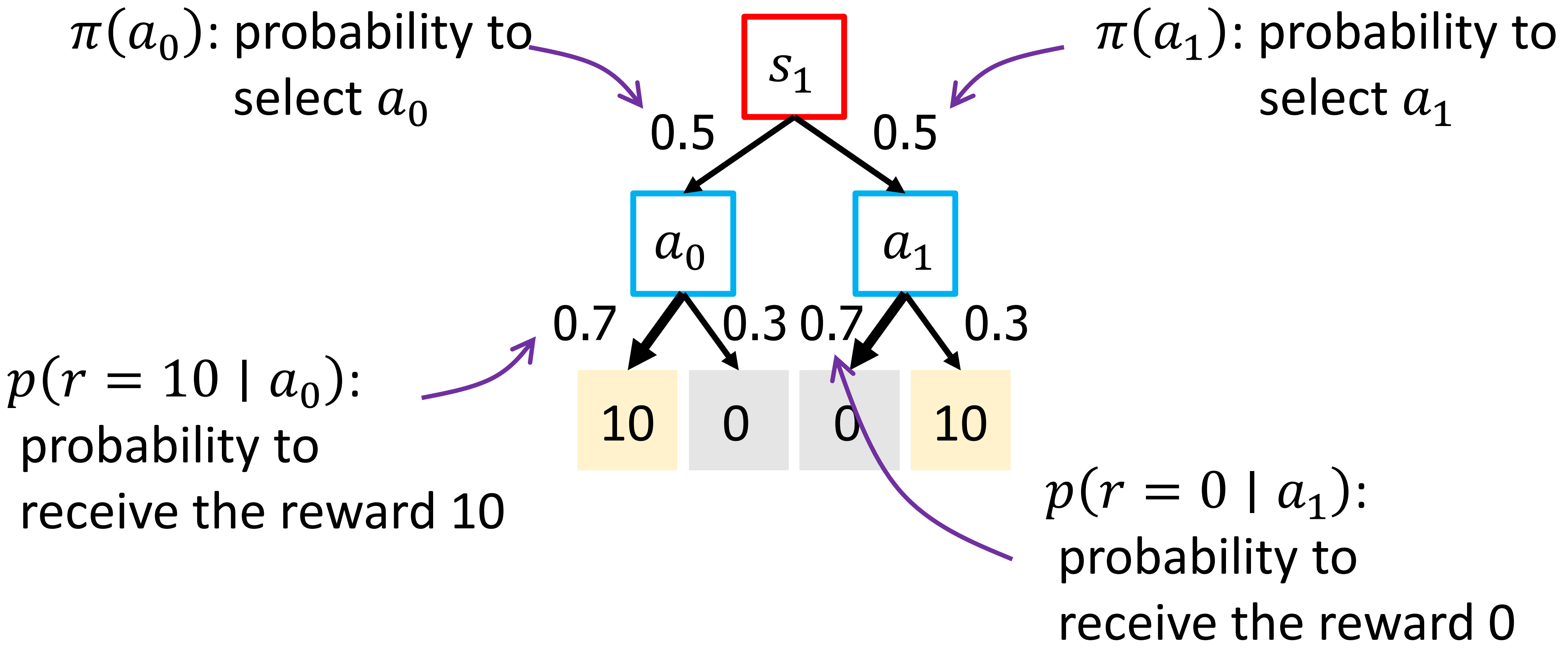
a_i

- Now, we will study Markov Decision Process (MDP) for sequential decision making
  - MDP formally describes an environment for reinforcement learning

ONE-STAGE MARKOV DECISION PROCESS

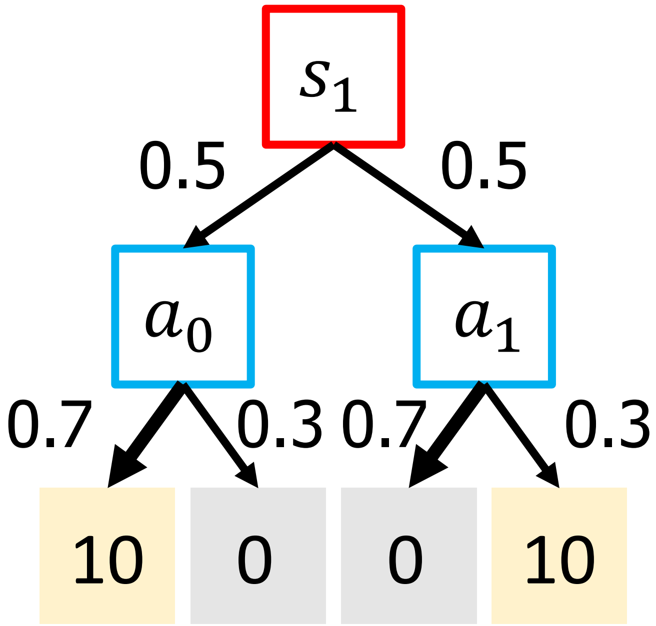
One-stage MDP

- 1-state 2-action task



Compute the action-values

- Expected value of reward
  - $Q(a_0) = \sum_r rp(r \mid a_0)$
    $= 10 \times 0.7 + 0 \times 0.3 = 7$
  - $Q(a_1) = 0 \times 0.7 + 10 \times 0.3 = 3$



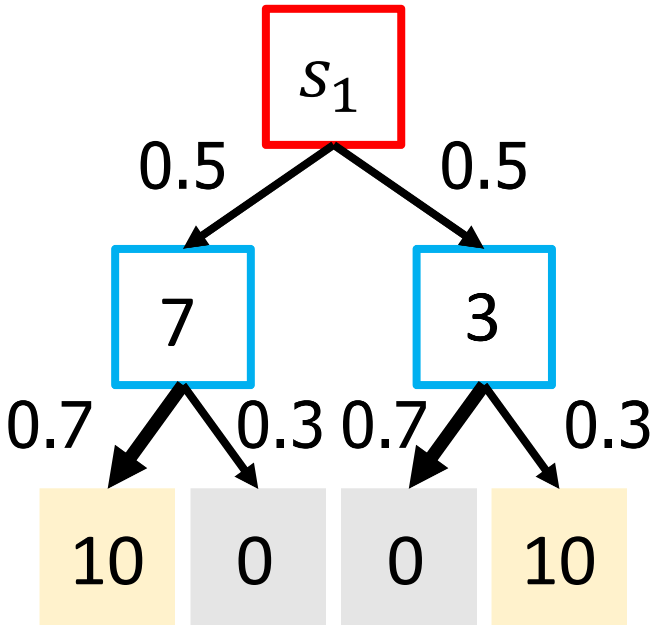
Compute the action-values

- Expected value of reward
  - $Q(a_0) = \sum_r rp(r \mid a_0)$

    $= 10 \times 0.7 + 0 \times 0.3 = 7$

  - $Q(a_1) = 0 \times 0.7 + 10 \times 0.3 = 3$

- More formally,
  - $Q(a) = \mathbb{E}_{r \sim p(r|a)}[r]$

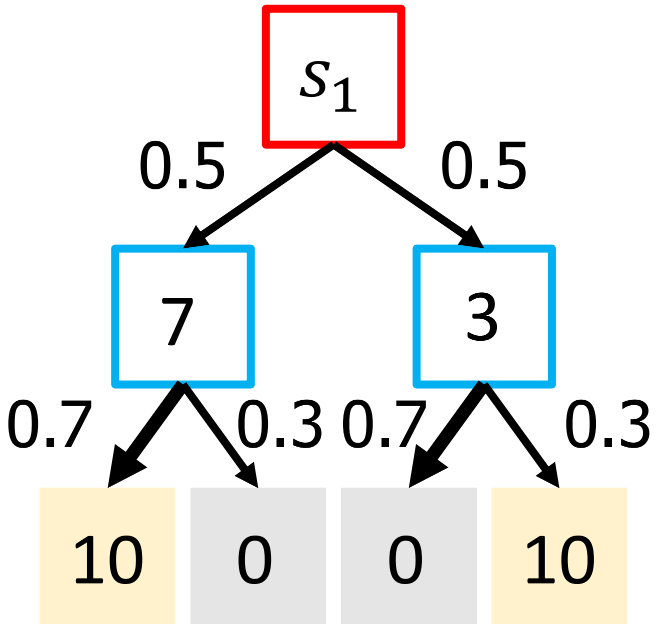


Compute the state-value

- The value of state s_1 depends on the policy. Suppose the policy is given by

$$\pi(a_0) = \pi(a_1) = 0.5.$$

- Possible transitions
  - $a_0 \rightarrow 10$ with probability 0.5×0.7
  - $a_0 \rightarrow 0$ with probability 0.5×0.3
  - $a_1 \rightarrow 0$ with probability 0.5×0.7
  - $a_1 \rightarrow 10$ with probability 0.5×0.3



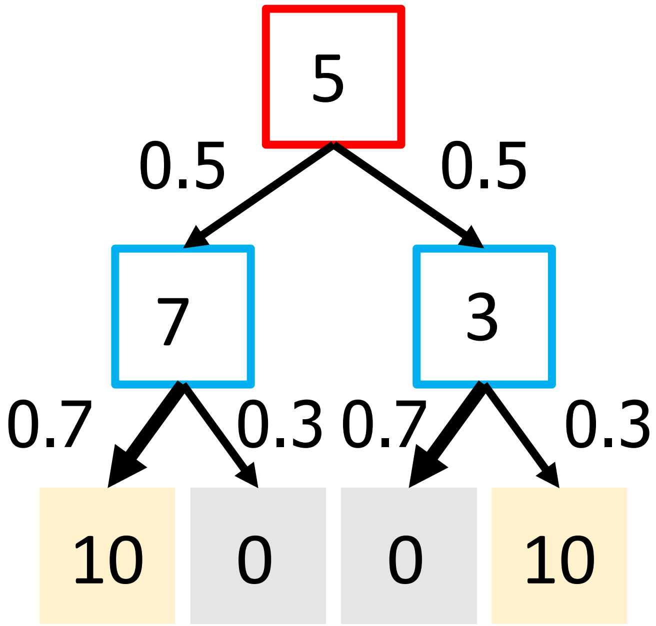
$$V(s_1) = 10 \times 0.5 \times 0.7 \\ +10 \times 0.5 \times 0.3 \\ = 5$$

Compute the state-value from the action-values

- The action values $Q(a_1)$ and $Q(a_2)$ can be used to compute the value of state
- $V(s_1) = \mathbb{E}_{a \sim \pi(a)}[Q(a)] = \sum_a Q(a)\pi(a)$

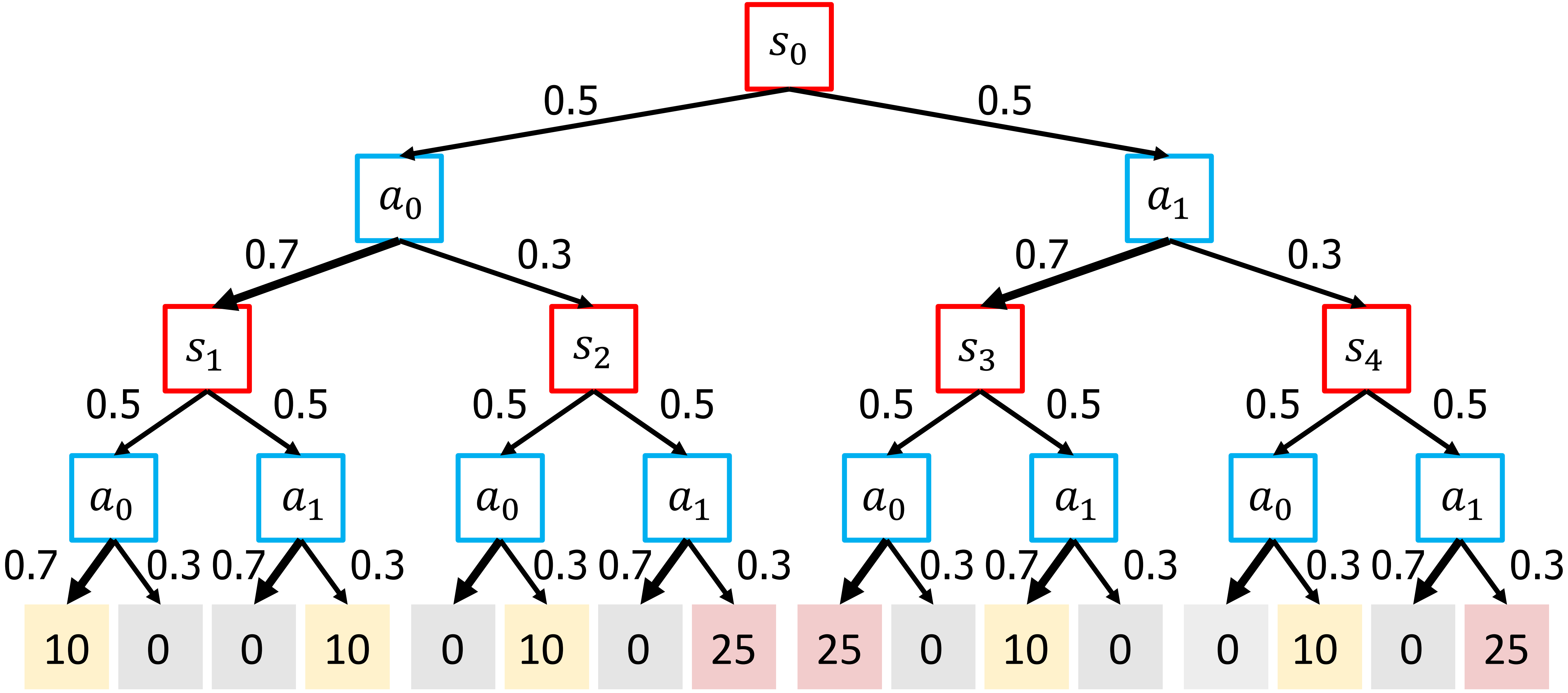
  $= 7 \times 0.5 + 3 \times 0.5 = 5$

- Hereafter, to clarify the dependency of π, the state-value function is denoted by $V^{\pi}(s)$



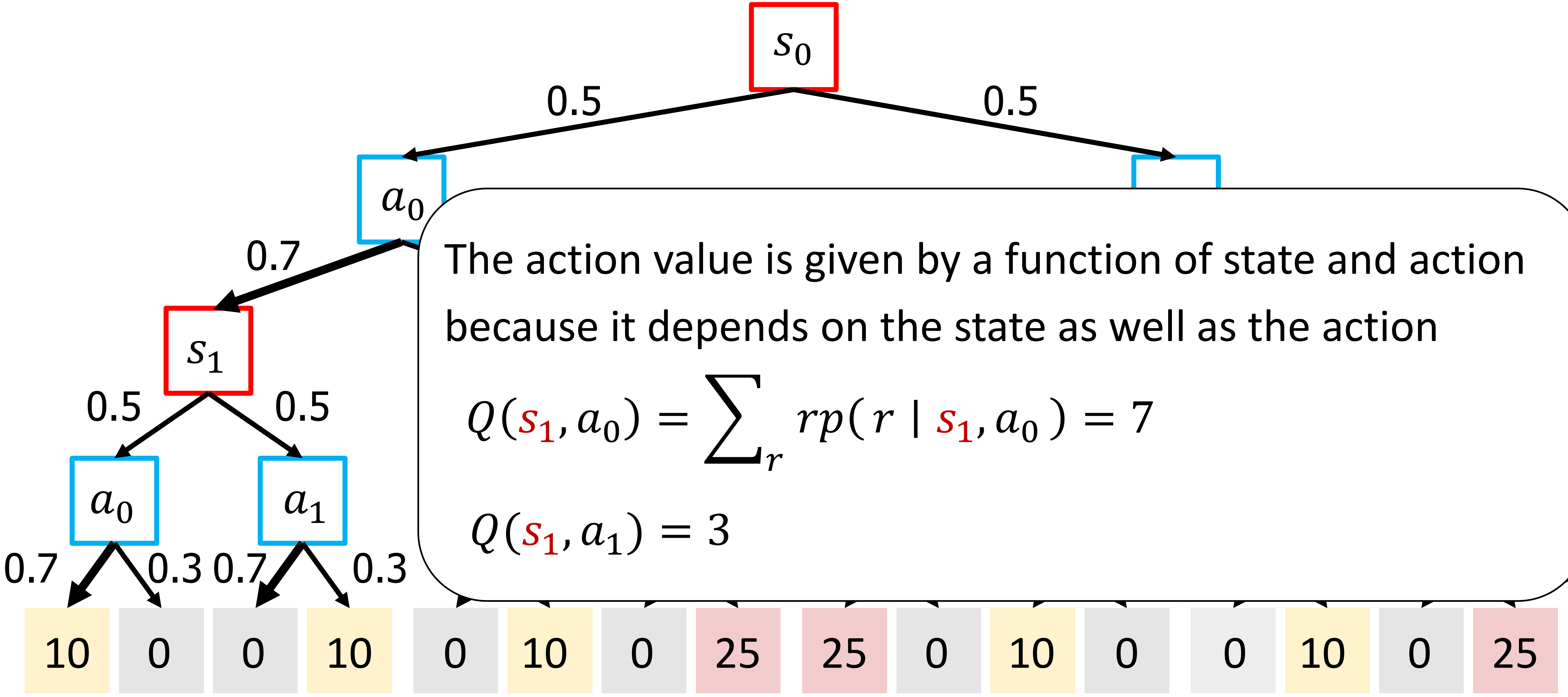
TWO-STAGE MARKOV DECISION PROCESS

Extension to a two-stage task



J. Gläscher, N. Daw, P. Dayan, and J.P. O'Doherty. (2010). States versus Rewards: Dissociable Neural Prediction Error Signals Underlying Model-Based and Model-Free Reinforcement Learning. Neuron, vol. 66, no. 4: 585–95.

Compute the action-values at the second stage

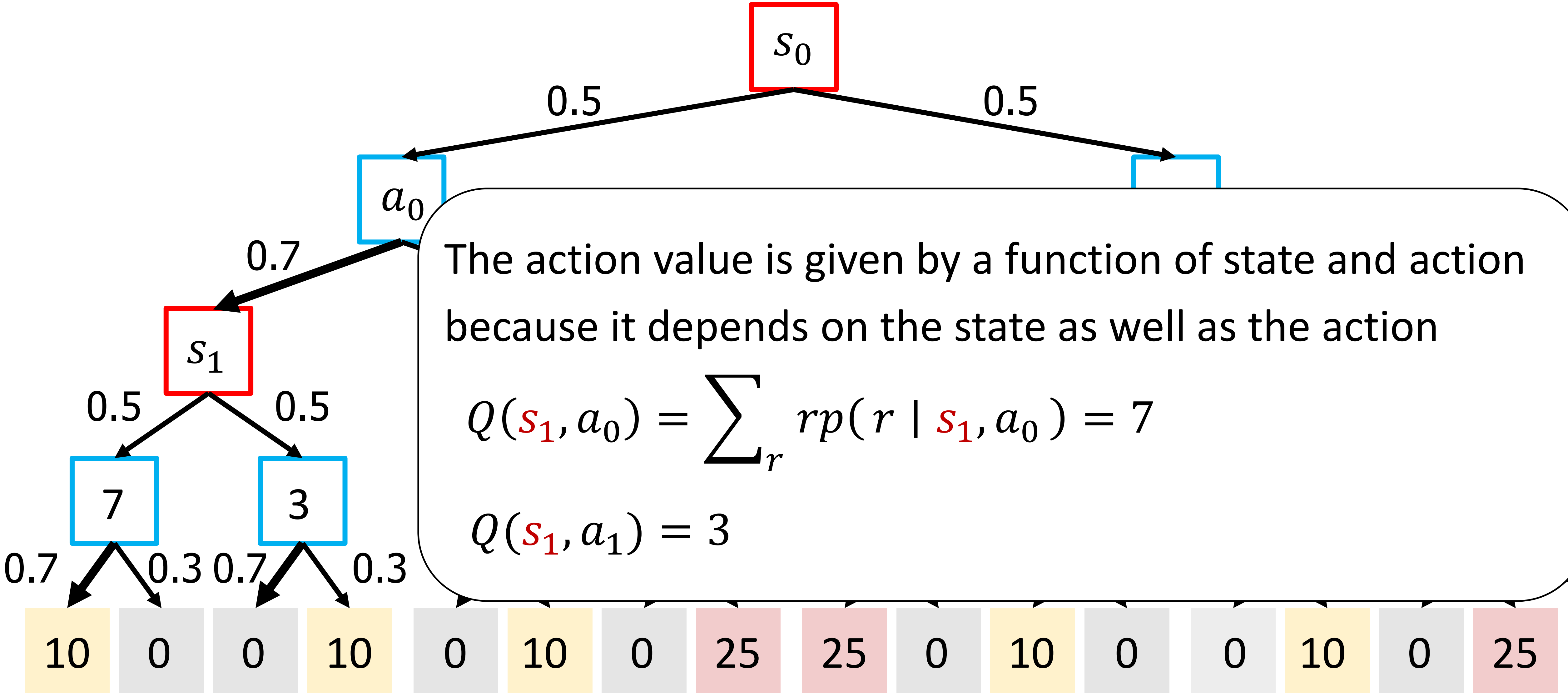


The action value is given by a function of state and action because it depends on the state as well as the action

$$Q(s_1, a_0) = \sum_r rp(r \mid s_1, a_0) = 7$$

$$Q(s_1, a_1) = 3$$

Compute the action-values at the second stage

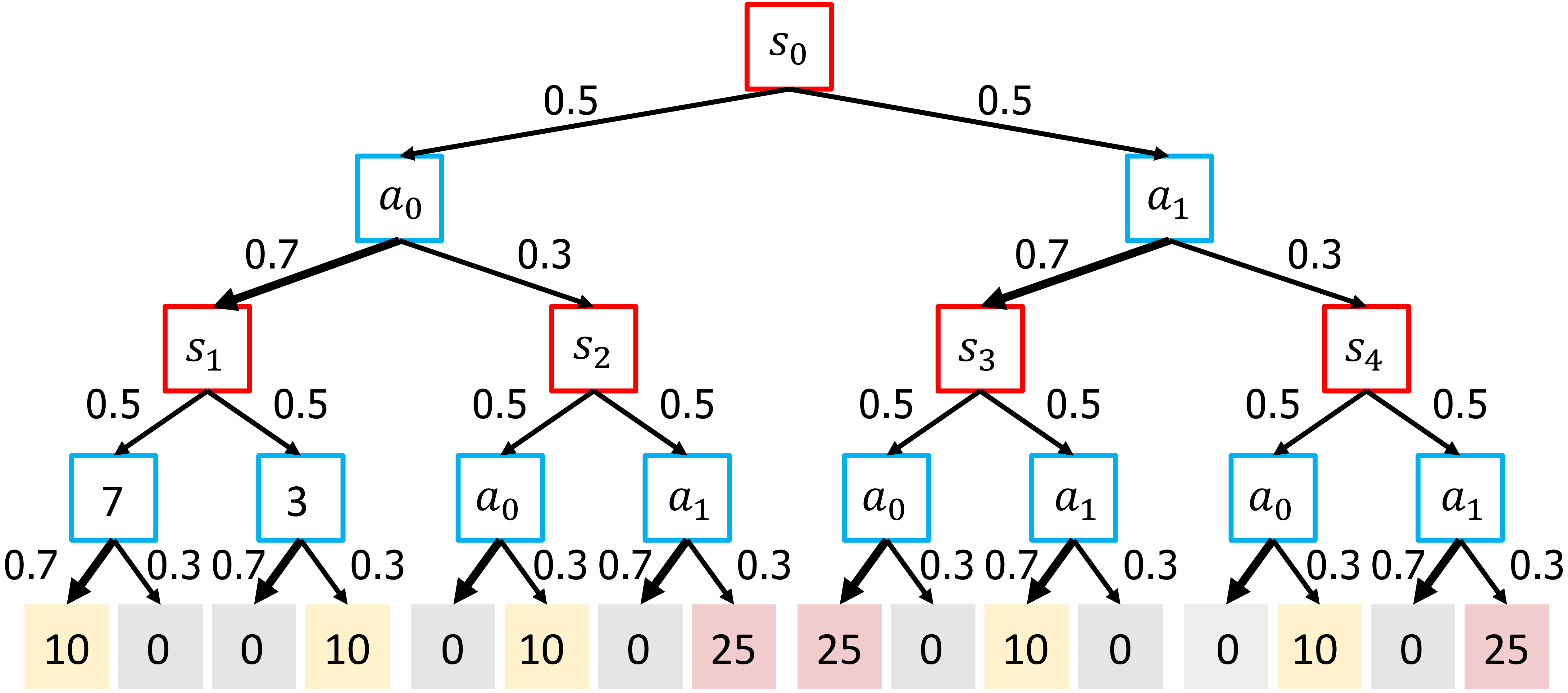


The action value is given by a function of state and action because it depends on the state as well as the action

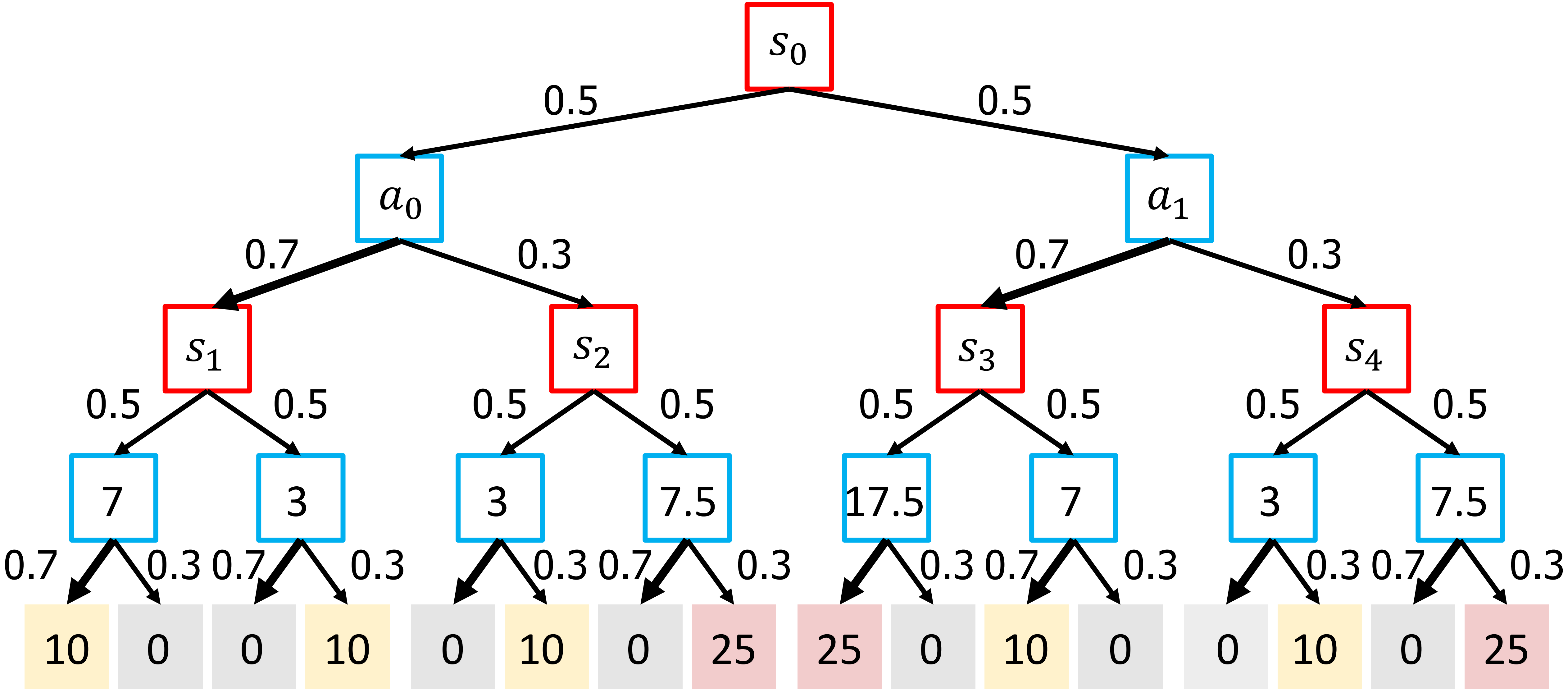
$$Q(s_1, a_0) = \sum_r rp(r \mid s_1, a_0) = 7$$

$$Q(s_1, a_1) = 3$$

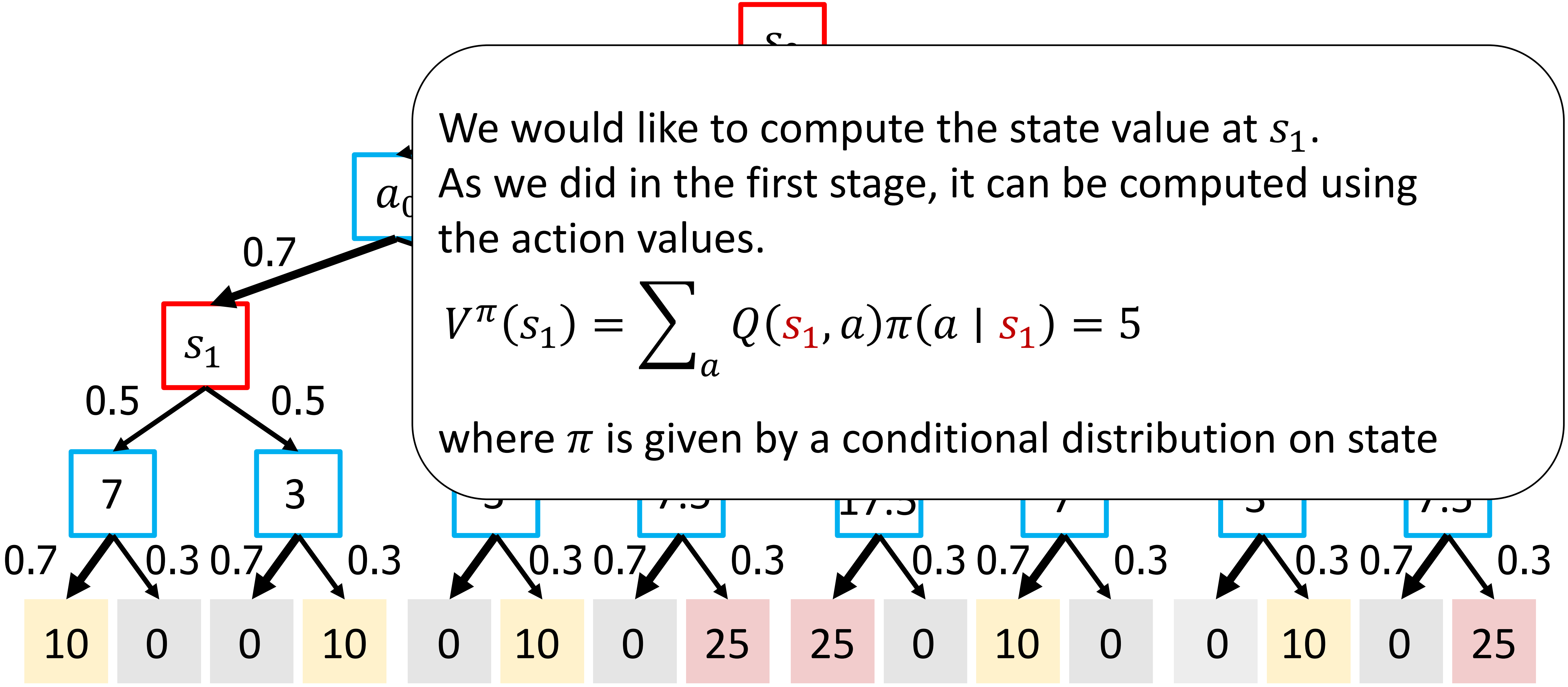
Compute the action-values at the second stage



Compute the action-values at the second stage



Compute the state-values at the second stage

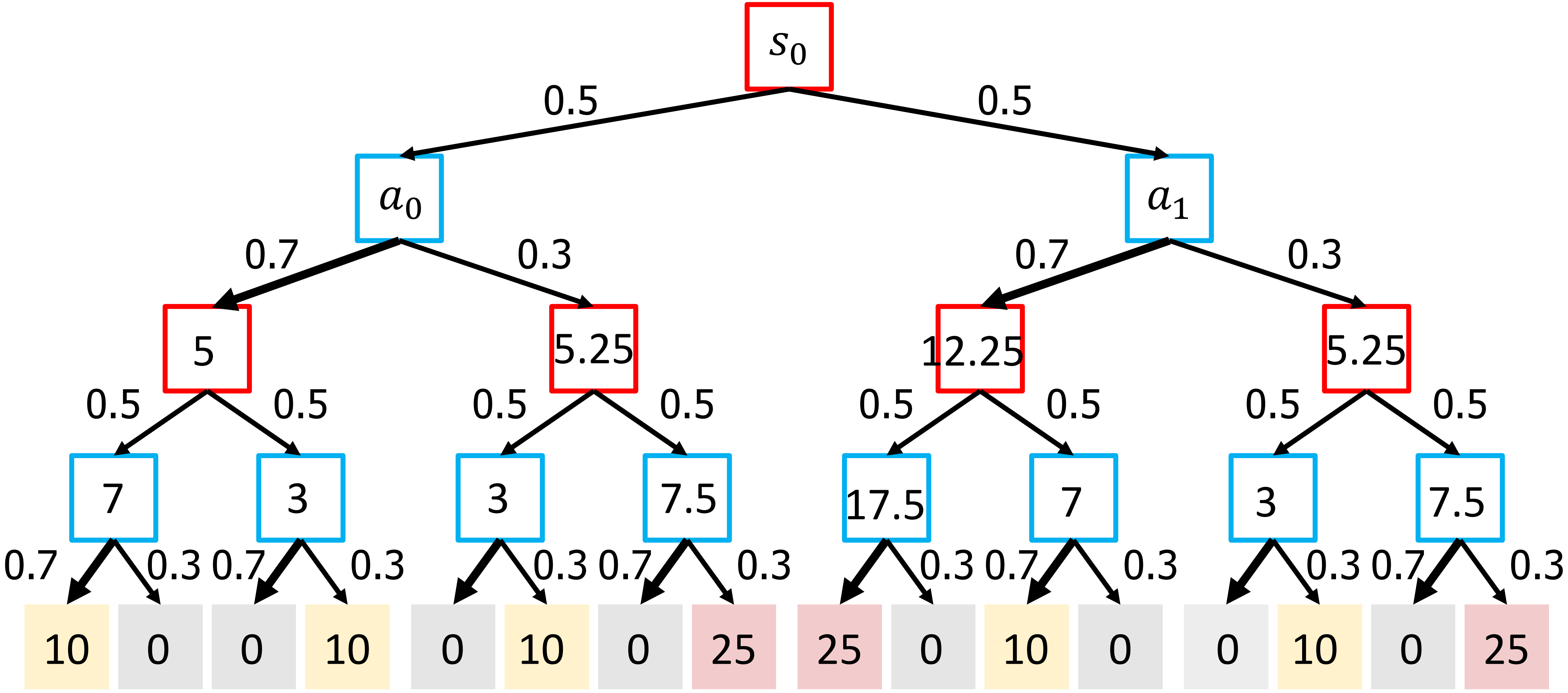


We would like to compute the state value at s_1.
As we did in the first stage, it can be computed using the action values.

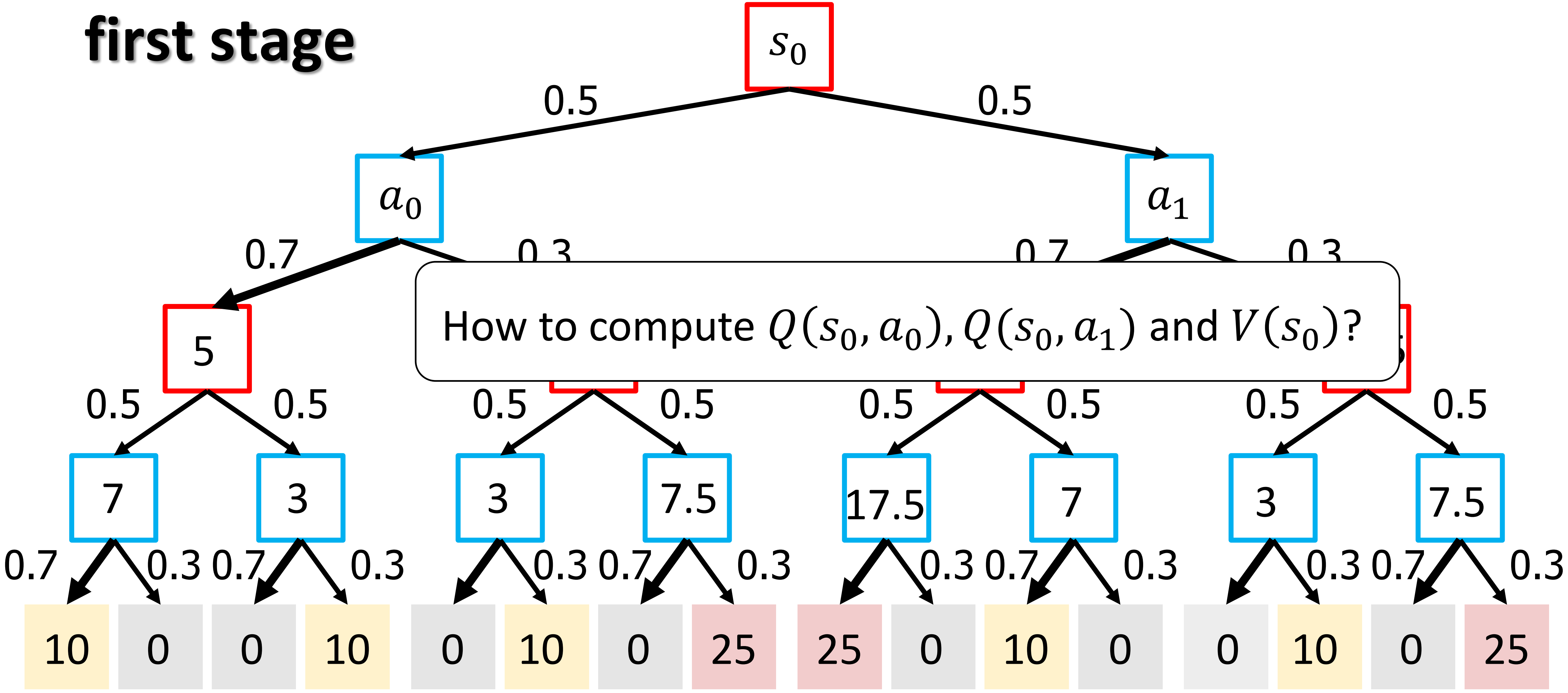
$$V^{\pi}(s_1) = \sum_a Q(s_1, a)\pi(a \mid s_1) = 5$$

where π is given by a conditional distribution on state

Compute the state-values at the second stage

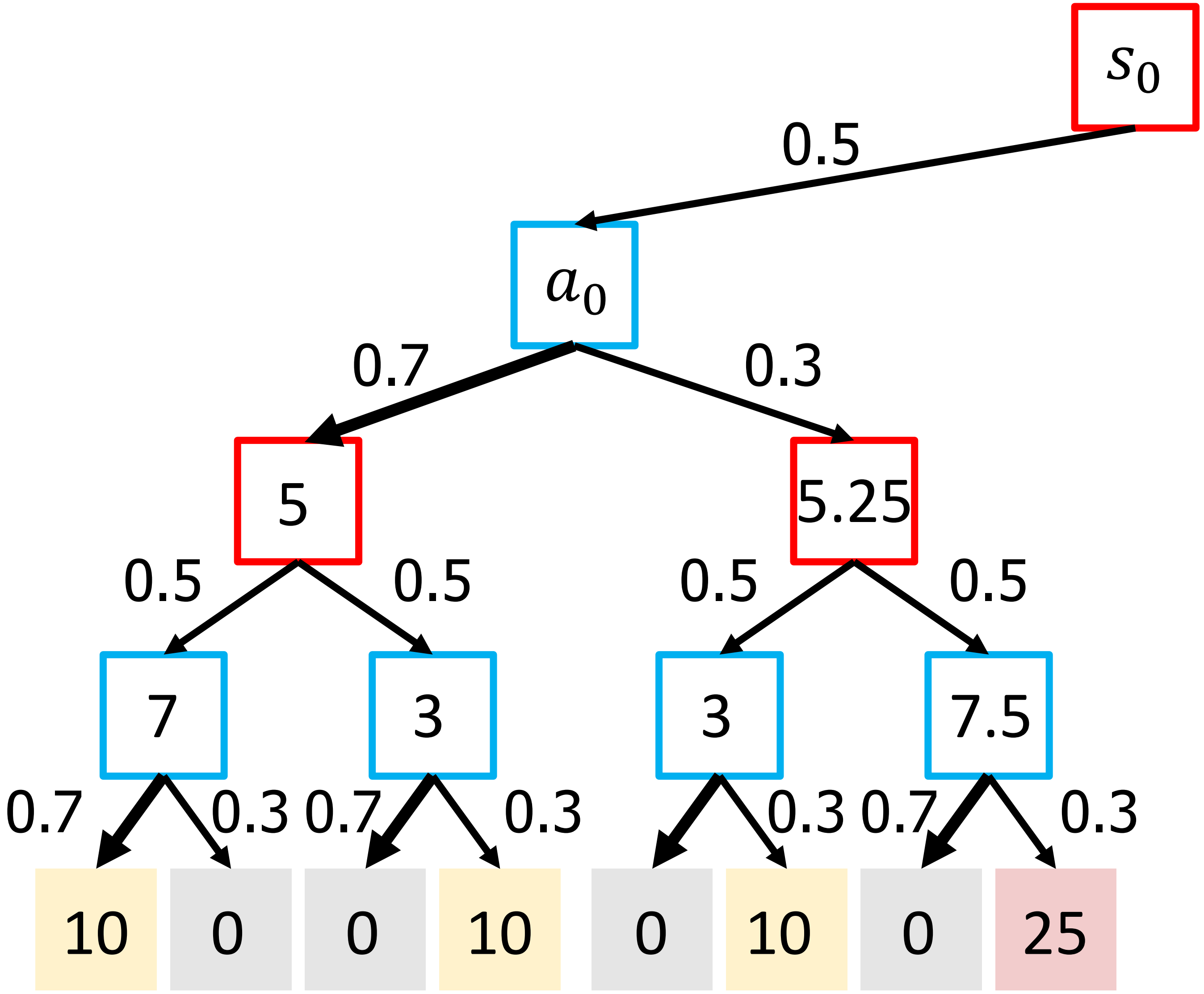


Compute the state- and action-values at the first stage



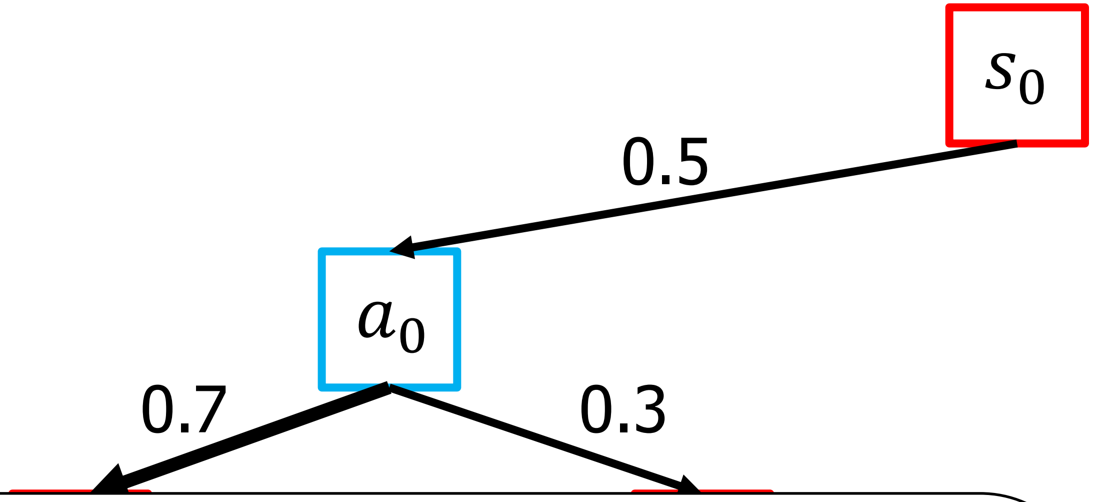
How to compute $Q(s_0, a_0)$, $Q(s_0, a_1)$ and $V(s_0)$?

Compute the action-values at the first stage



- Possible transitions
  - $s_1 \to a_0 \to 10$ with $0.7 \times 0.5 \times 0.7$
  - $s_1 \to a_0 \to 0$ with $0.7 \times 0.5 \times 0.3$
  - $s_1 \to a_1 \to 0$ with $0.7 \times 0.5 \times 0.7$
  - $s_1 \to a_1 \to 10$ with $0.7 \times 0.5 \times 0.3$
  - $s_2 \to a_0 \to 0$ with $0.3 \times 0.5 \times 0.7$
  - $s_2 \to a_0 \to 10$ with $0.3 \times 0.5 \times 0.3$
  - $s_2 \to a_1 \to 0$ with $0.3 \times 0.5 \times 0.7$
  - $s_2 \to a_1 \to 25$ with $0.3 \times 0.5 \times 0.3$

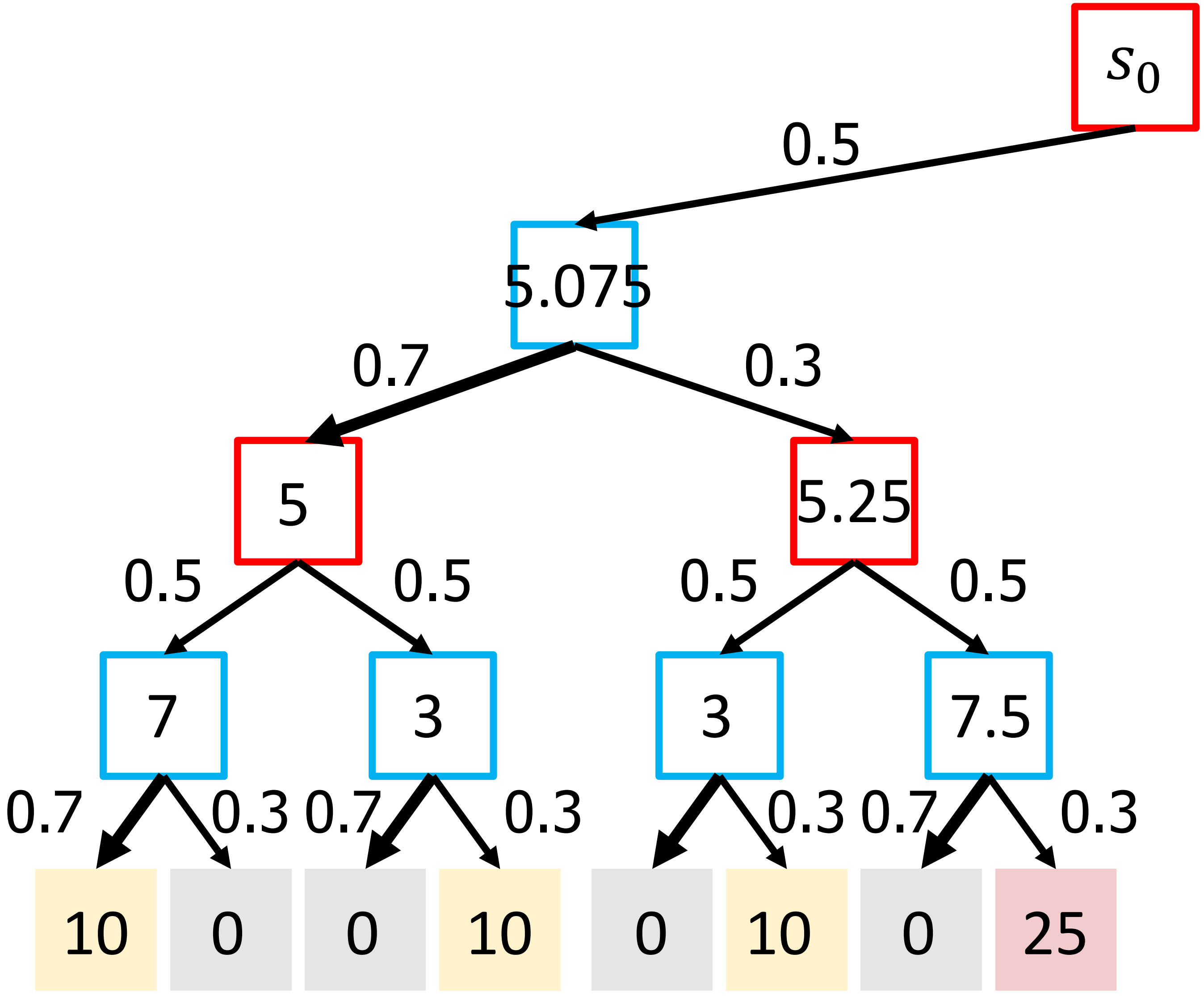
Compute the action-values at the first stage



$$
\begin{aligned}
Q(s_0, a_0) &= 10 \times 0.7 \times 0.5 \times 0.7 \\
&\quad +10 \times 0.7 \times 0.5 \times 0.3 \\
&\quad +10 \times 0.3 \times 0.5 \times 0.3 \\
&\quad +25 \times 0.3 \times 0.5 \times 0.3 \\
&= 5.075
\end{aligned}
$$

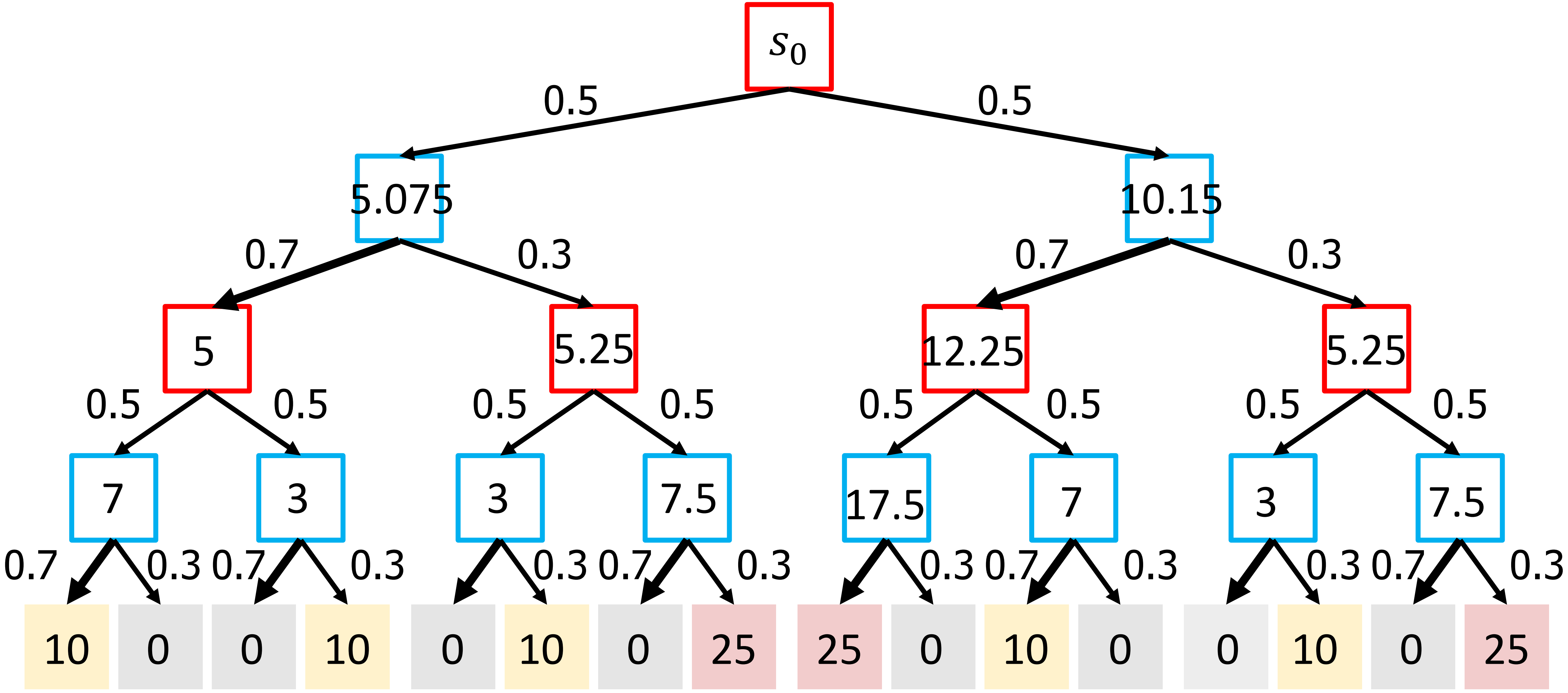
- Possible transitions
  - $s_1 \rightarrow a_0 \rightarrow 10$ with $0.7 \times 0.5 \times 0.7$
  - $s_1 \rightarrow a_0 \rightarrow 0$ with $0.7 \times 0.5 \times 0.3$
  - $s_1 \rightarrow a_1 \rightarrow 0$ with $0.7 \times 0.5 \times 0.7$
  - $s_1 \rightarrow a_1 \rightarrow 10$ with $0.7 \times 0.5 \times 0.3$
  - $s_2 \rightarrow a_0 \rightarrow 0$ with $0.3 \times 0.5 \times 0.7$
  - $s_2 \rightarrow a_0 \rightarrow 10$ with $0.3 \times 0.5 \times 0.3$
  - $s_2 \rightarrow a_1 \rightarrow 0$ with $0.3 \times 0.5 \times 0.7$
  - $s_2 \rightarrow a_1 \rightarrow 25$ with $0.3 \times 0.5 \times 0.3$

Compute the action-values at the first stage



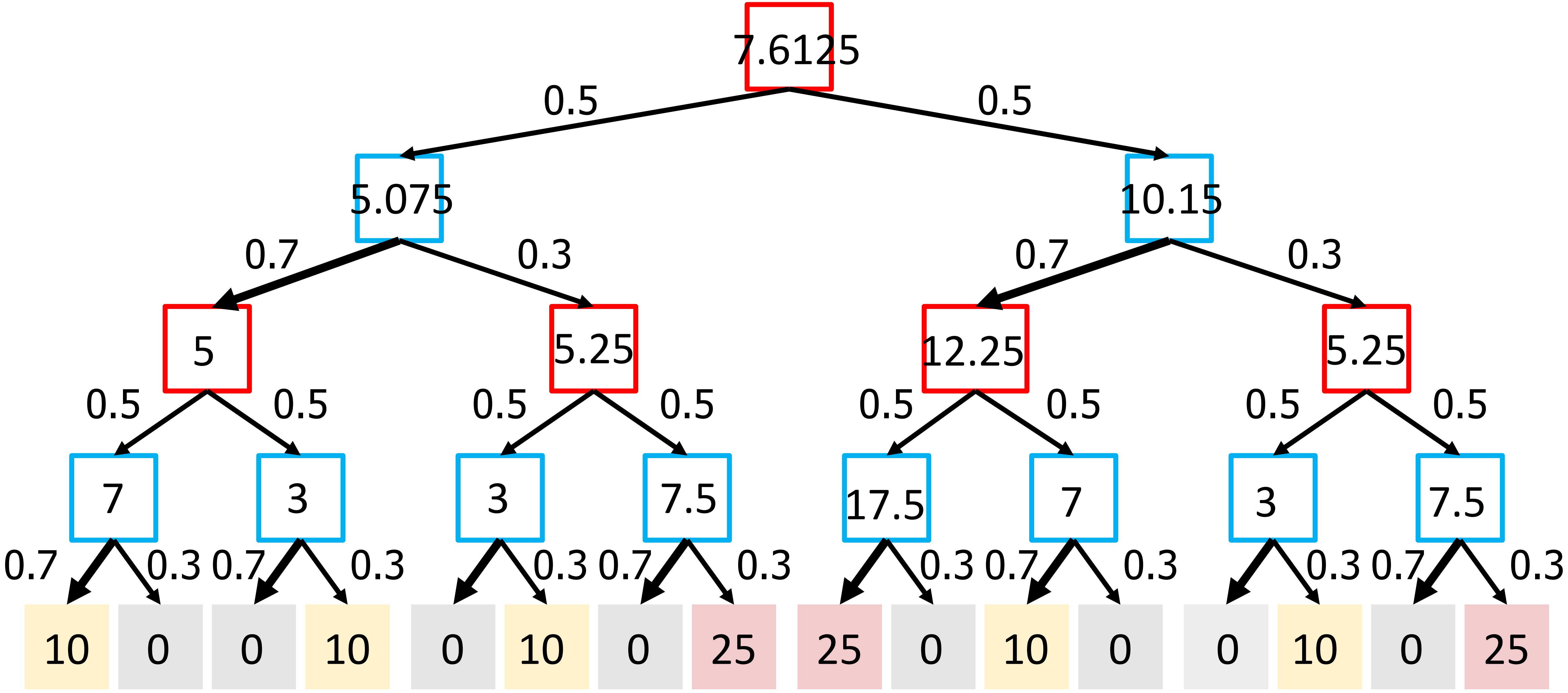
- Possible transitions
  - $s_1 \rightarrow a_0 \rightarrow 10$ with $0.7 \times 0.5 \times 0.7$
  - $s_1 \rightarrow a_0 \rightarrow 0$ with $0.7 \times 0.5 \times 0.3$
  - $s_1 \rightarrow a_1 \rightarrow 0$ with $0.7 \times 0.5 \times 0.7$
  - $s_1 \rightarrow a_1 \rightarrow 10$ with $0.7 \times 0.5 \times 0.3$
  - $s_2 \rightarrow a_0 \rightarrow 0$ with $0.3 \times 0.5 \times 0.7$
  - $s_2 \rightarrow a_0 \rightarrow 10$ with $0.3 \times 0.5 \times 0.3$
  - $s_2 \rightarrow a_1 \rightarrow 0$ with $0.3 \times 0.5 \times 0.7$
  - $s_2 \rightarrow a_1 \rightarrow 25$ with $0.3 \times 0.5 \times 0.3$

Compute the action-values at the first stage



Compute the state-value at the second stage

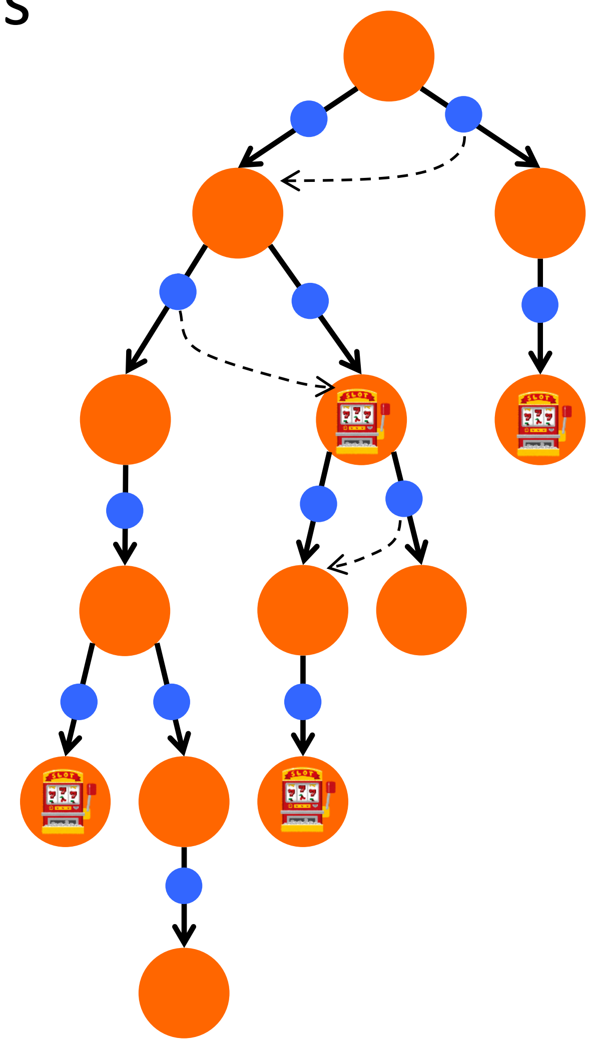
Open in Colab



GENERAL FORM OF MARKOV DECISION PROCESS

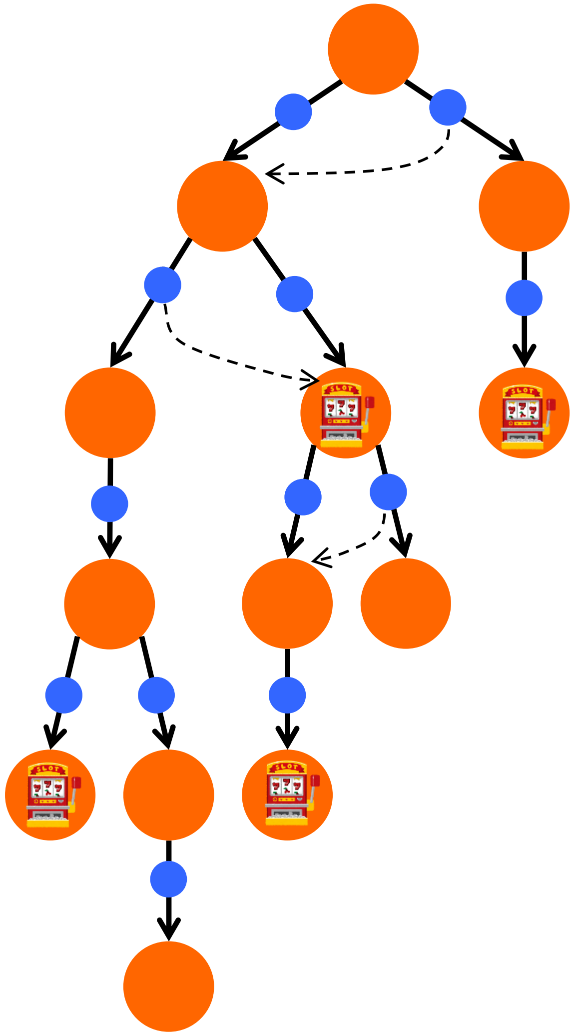
Extension to Sequential Decision Making Problems

- A learning agent is not in front of gambling machines
- A current position is considered as a state
  - The state is a sufficient statistics to describe the dynamics of the environment
  - Bandits are MDPs with one state
- The action of the agent influences the environment, causing a state transition



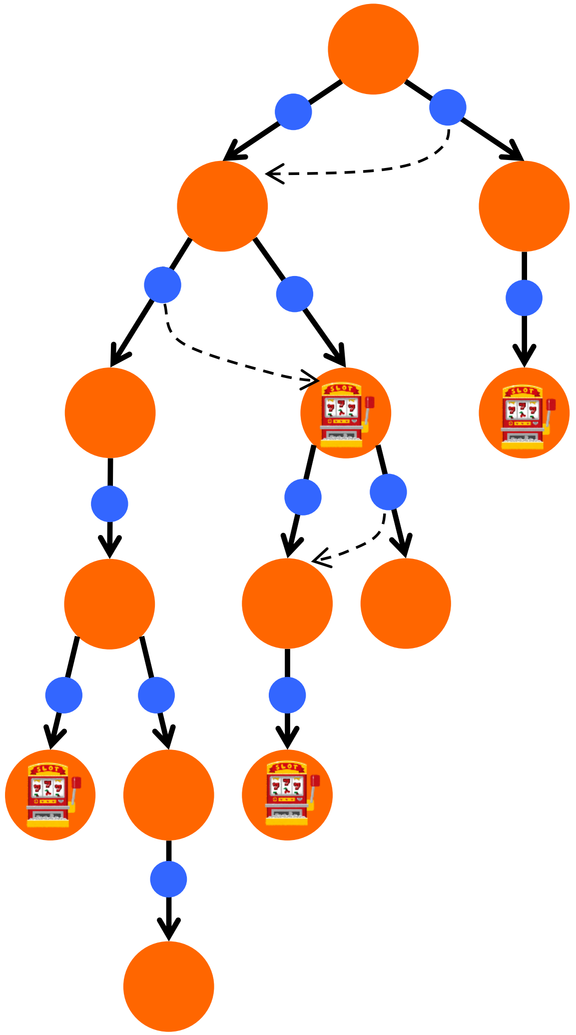
Extension to Sequential Decision Making Problems

- s: state,
- a: action,
- State set: $\mathcal{S} = \{s_1, s_2, \ldots, s_{|\mathcal{S}|}\}$
- Action set: $\mathcal{A} = \{a_1, a_2, \ldots, a_{|\mathcal{A}|}\}$



Extension to Sequential Decision Making Problems

- $p_0(s)$: initial state distribution
- $p(s', r \mid s, a)$: stochastic environmental dynamics
  probability of occurring state s' and reward r
  when executing action a at state s
- $\pi(\, a \mid s \,)$: stochastic policy
  probability to select action a at state s
- $\pi(s) \in \mathcal{A}$: deterministic policy
  action at state s



Markov property

- Consider a sequence of states, action, and rewards

$$s_0, a_0, r_1, s_1, a_1, r_2, \dots, s_{t-1}, a_{t-1}, r_t, s_t, a_t, s_{t+1}, r_{t+1}$$

  where the subscript t represents the time index

- How can we model the next state s_{t+1} and the reward r_{t+1}?
  The most general way is to introduce a conditional probability distribution

$$\Pr(s_{t+1} = s', r_{t+1} = r \mid s_0, a_0, r_1, \dots, s_{t-1}, a_{t-1}, r_t, s_t, a_t)$$

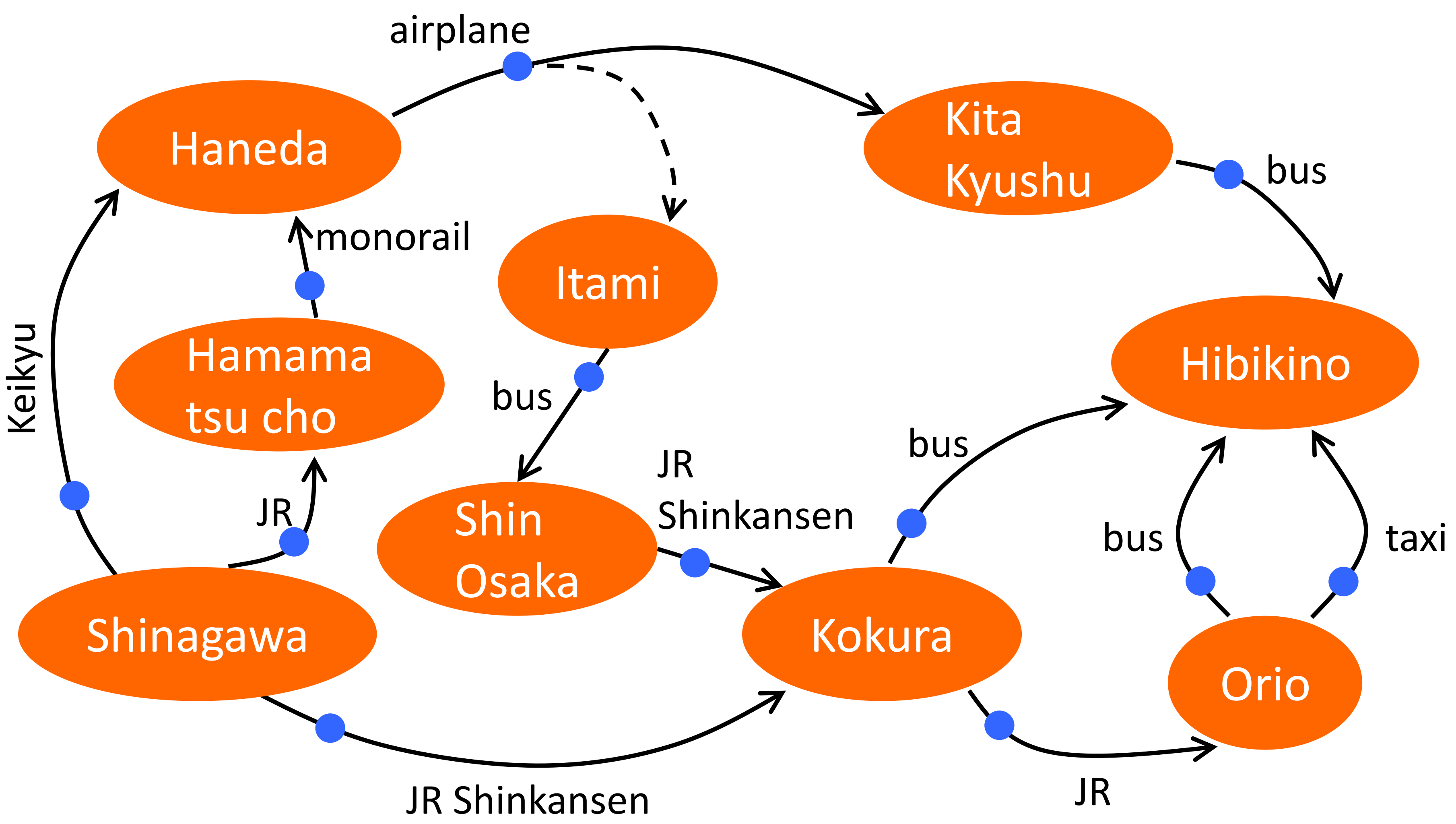
Markov property

- It is too complicated! We usually **assume** that the environment satisfies the following Markov property:

$$p(s', r \mid s, a) = \Pr(S_{t+1} = s', R_{t+1} = r \mid S_t = s, A_t = a)$$

- We have already utilized the Markovian assumption in the previous two-stage task

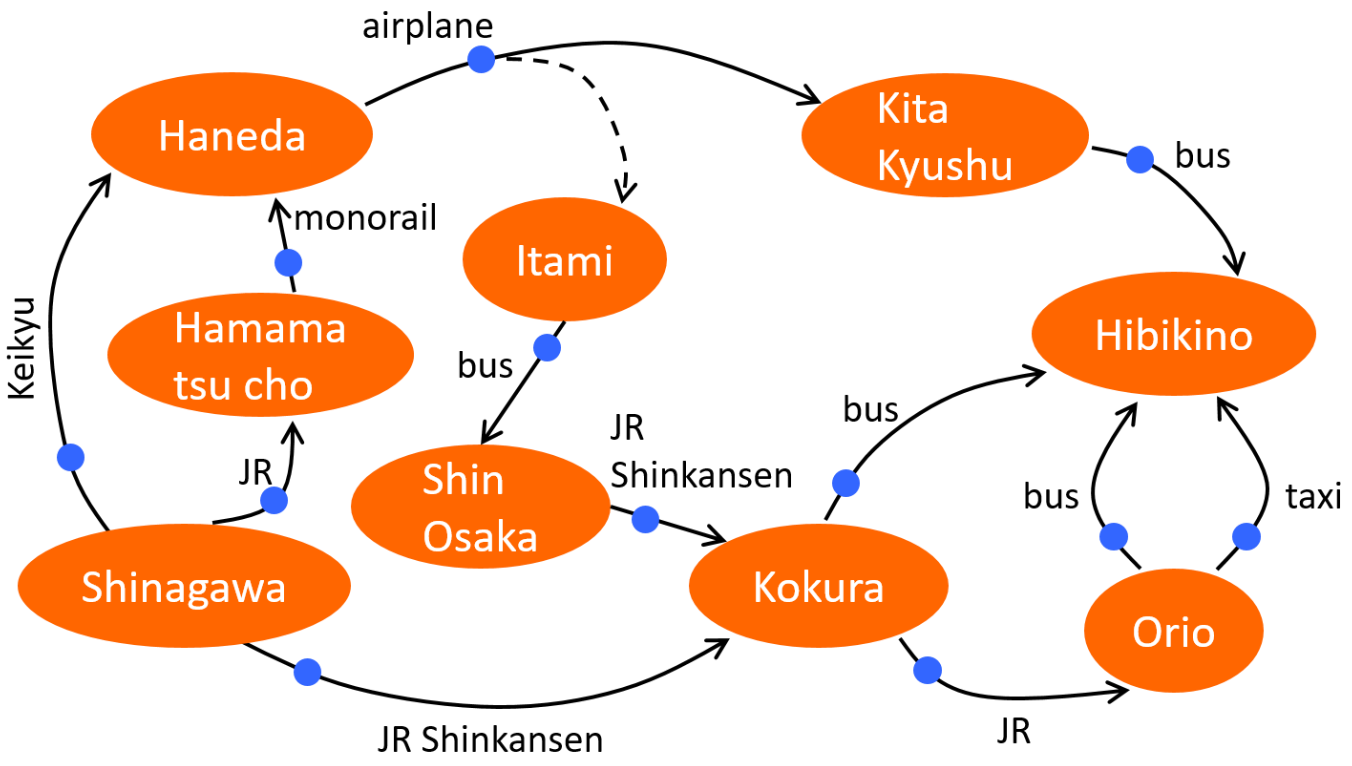
Example: From Shinagawa to Hibikino Campus



States and Actions in the Example

- $\mathcal{S}$={Shinagawa, Hamamatsu cho, Haneda, Itami, Shin Osaka, Kitakyushu, Kokura, Orio, Hibikino}

- Action set is state dependent in this case
  - $\mathcal{A}$(Shinagawa) = {Keikyu, JR, JR Shinkansen}
  - $\mathcal{A}$(Kokura) = {bus, JR}
  - $\mathcal{A}$(Haneda) = {Airplane}
  - $\vdots$



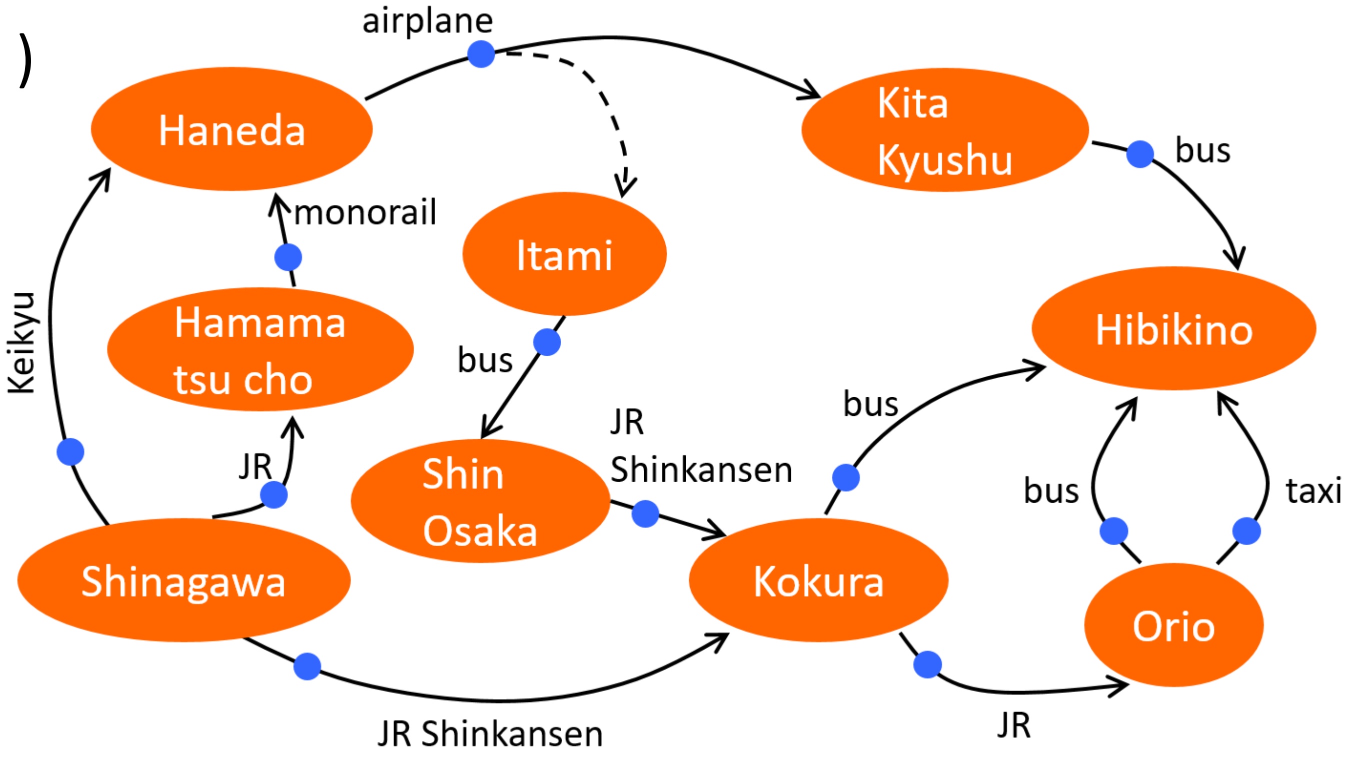
State Representation is Important

- Markovian property
  - Pr(Kita Kyushu|Haneda, airplane, Hamamatsu-cho, monorail, Shinagawa, JR)

    = Pr(Kita Kyushu|Haneda, airplane, Shinagawa, Keikyu)

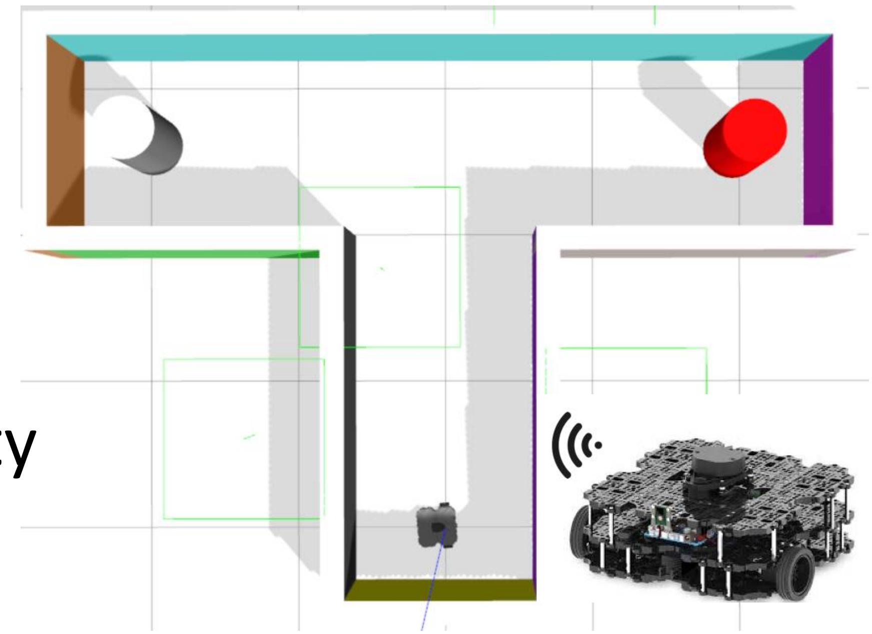
    = Pr(Kita Kyushu|Haneda, airplane)

independent on the route



State Representation is Important

- The task of the robot is to move from the start position to the goal (red cylinder)
- The robot cannot see the red cylinder at the corner
- If a state is given by a position and an orientation of the robot, simple RL algorithms cannot achieve this task because the state representation does not satisfy Markov property
- Memory is needed



J. Wang et al. (2020). Modular deep reinforcement learning from reward and punishment for robot navigation. Neural Networks, 135: 115-126.

Components

- State transition probability: $p(\, s' \mid s, a\,) = \sum_r p(s', r \mid s, a)$

- Expected reward for state-action pairs:

$$r(s,a) = \sum_r rp(\, r \mid s, a\,) \quad \text{where} \quad p(\, r \mid s, a\,) = \sum_{s'} p(s', r \mid s, a)$$

- Expected reward for state-action-next-state triples

$$r(s,a,s') = \sum_r rp(\, r \mid s, a, s'\,) \quad \text{where} \quad p(\, r \mid s, a, s'\,) = \frac{p(\, s', r \mid s, a\,)}{p(s' \mid s, a)}$$

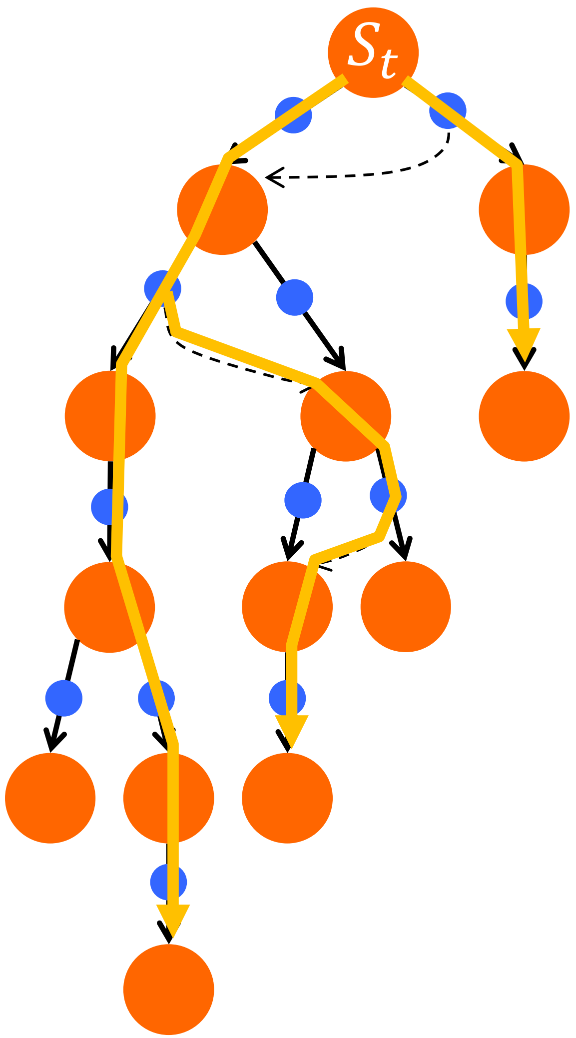
Return

- The return G_t is defined as the sum of discounted rewards from time-step t

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

  – G_t is a random variable

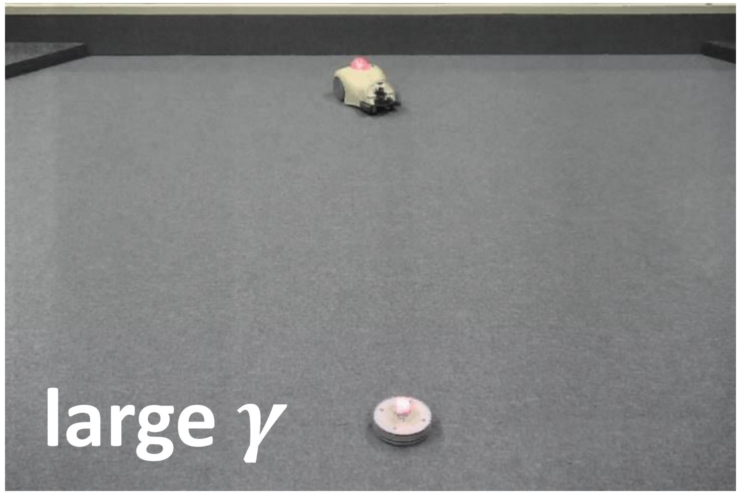
- $\gamma \in [0, 1)$ is the discount rate
  – γ close to 0: myopic evaluation
  – γ close to 1: far-sighted evaluation



Why Discount?

- The learned behavior is affected by the choice of γ

The robot does not move towards the battery



The robot tries to catch the battery

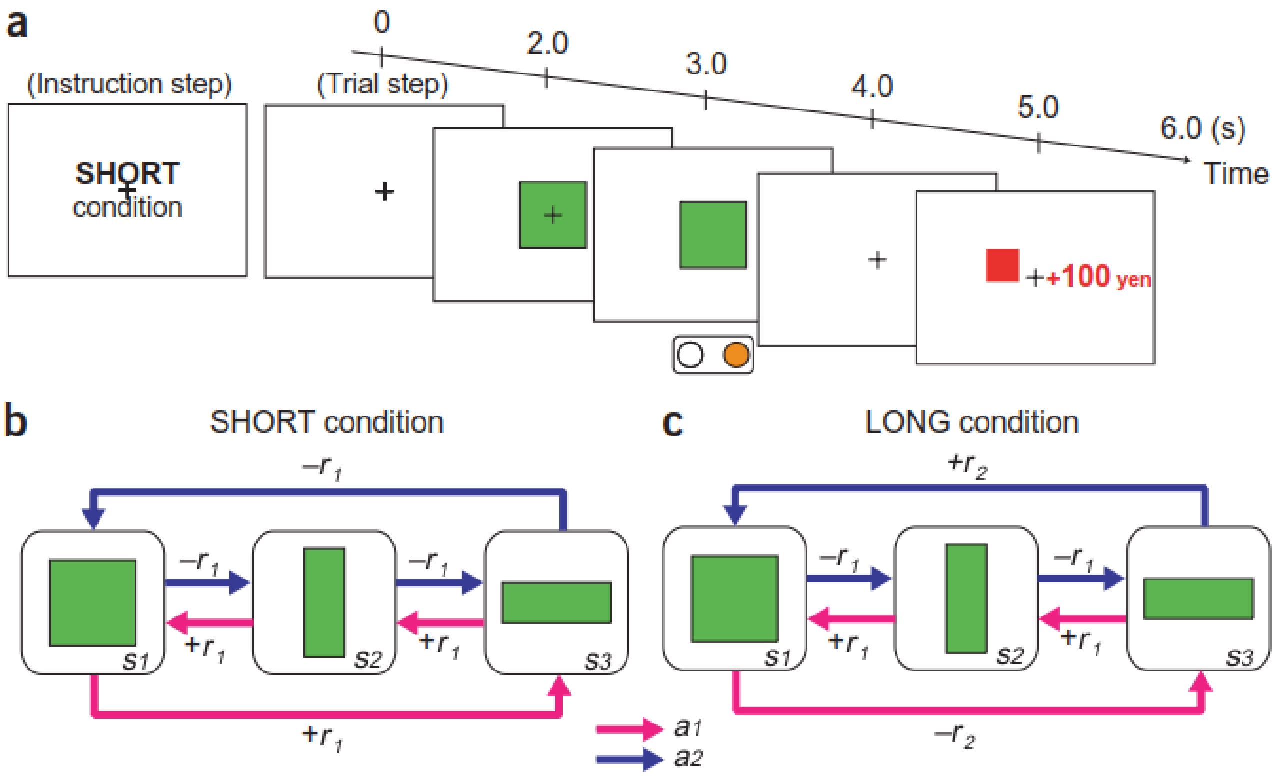
- If the reward is bounded, the return is also bounded:

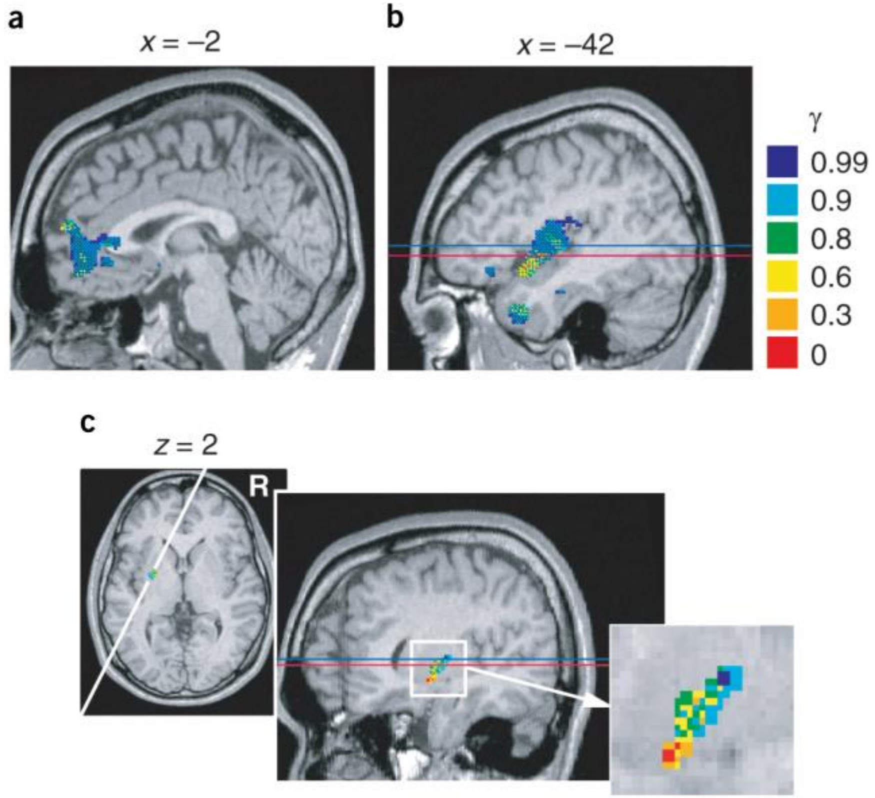
$$|R| \leq R_{\max} \Rightarrow |G| \leq \frac{R_{\max}}{1-\gamma}$$

  – Useful for convergence

Why Discount?

- Prediction of immediate and future rewards differentially recruits cortico-basal ganglia loops





S.C. Tanaka, K. Doya, G. Okada, K. Ueda, Y. Okamoto, and S. Yamawaki. (2004). Prediction of immediate and future rewards differentially recruits cortico-basal ganglia loops. Nature Neuroscience, 7(8): 887-893.

State-Value Function

- A state value function evaluates the policy in each state
- For a given stationary policy π, consider a state-action-reward sequence

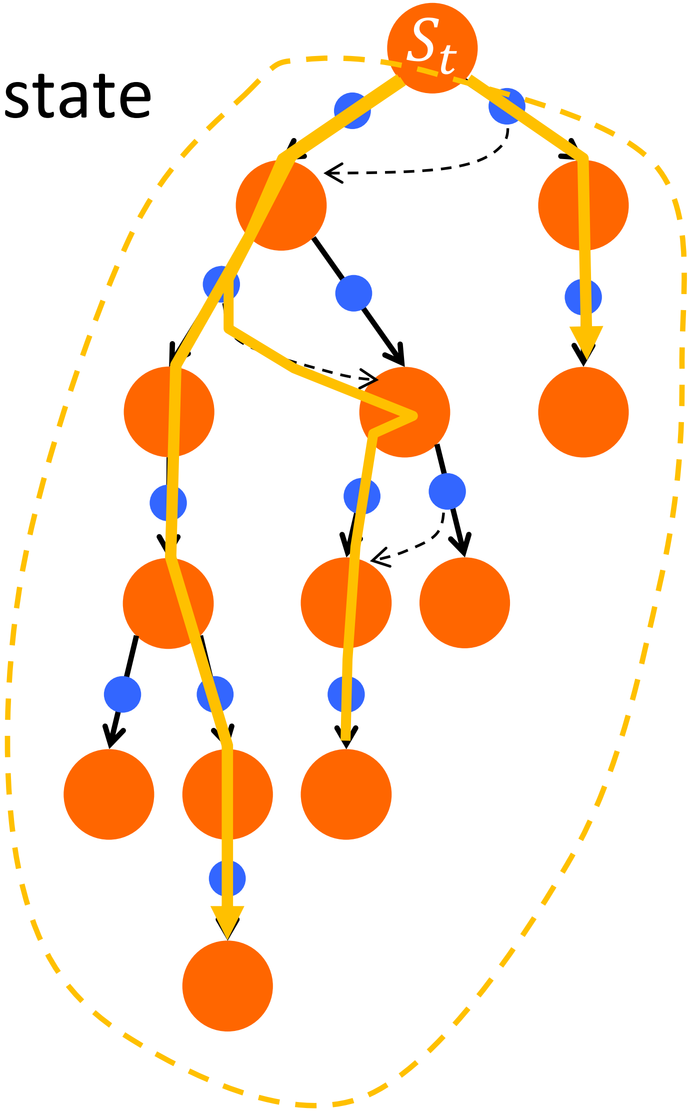
$$s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1}, r_{t+2}, s_{t+2} \dots$$

generated by

$$a_t \sim \pi(a \mid s_t), \; s_{t+1}, r_{t+1} \sim p(s', r \mid s_t, a_t)$$

- The state value function is the expected return given by

$$V^\pi(s) \triangleq \mathbb{E}_\pi[G_t \mid s_t = s, a_{t:\infty} \sim \pi]$$



State-Value Function

- The value of a state s under a policy π is defined by

$$V^{\pi}(s) \triangleq \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \,|s_t = s\right]$$

- $\mathbb{E}_{\pi}\{\quad\}$ denotes the expectation which is taken over the probability distribution

$$P(\, a_t, r_{t+1}, s_{t+1}, a_{t+1}, r_{t+2}, s_{t+2}, \ldots \mid s_t = s\,)$$
$$= \prod_{k=0}^{\infty} \pi(a_{t+k}|s) P_T(s_{t+k+1}, r_{t+k+1}|s_{t+k}, a_{t+k})$$

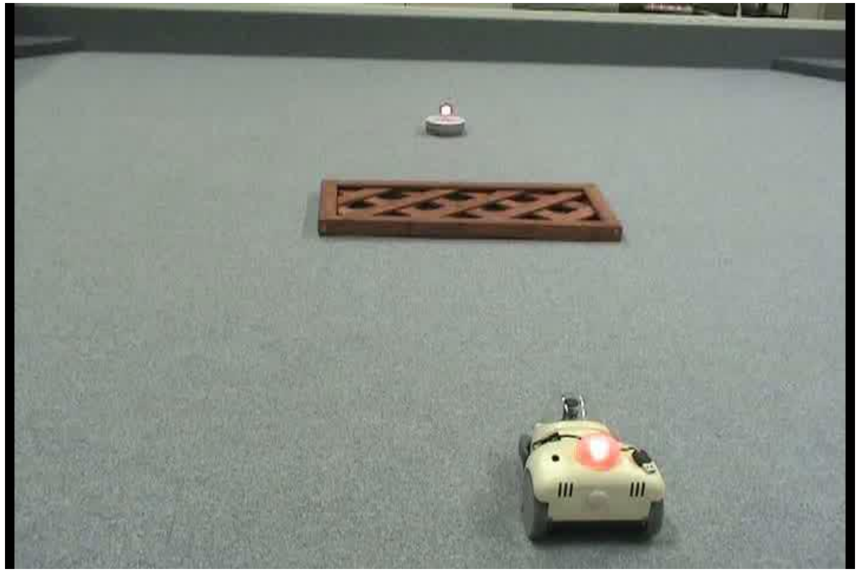
- Therefore,

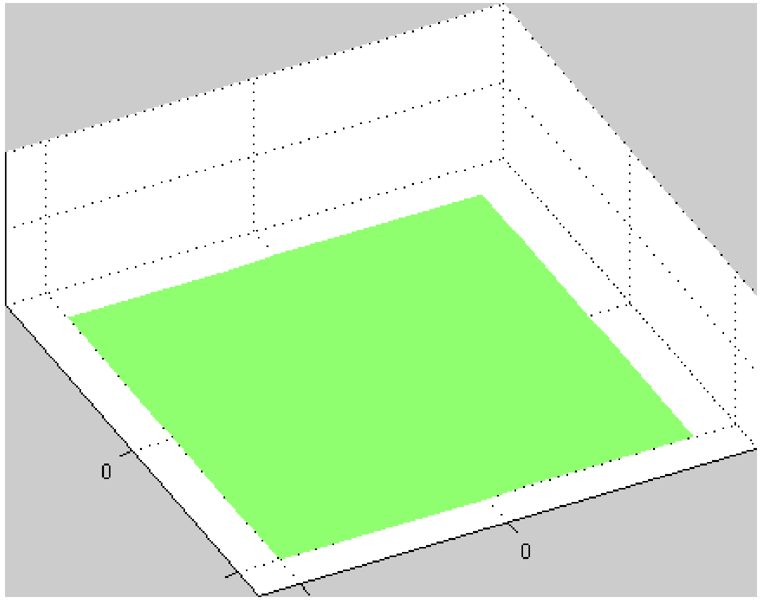
$$V^{\pi}(s) = \sum_{a_t, r_{t+1}, s_{t+1}, \ldots} \Pr(a_t, r_{t+1}, s_{t+1}, \ldots |s_t = s) \left[\sum_{k=0}^{\infty} \gamma^k\, r_{t+k+1}\right]$$

Example of State-Value Function

- Task: to get the battery pack while avoiding collisions with an obstacle
  - Positive reward for successful battery catching
  - Negative reward for collisions with obstacles

The goal state has the highest value



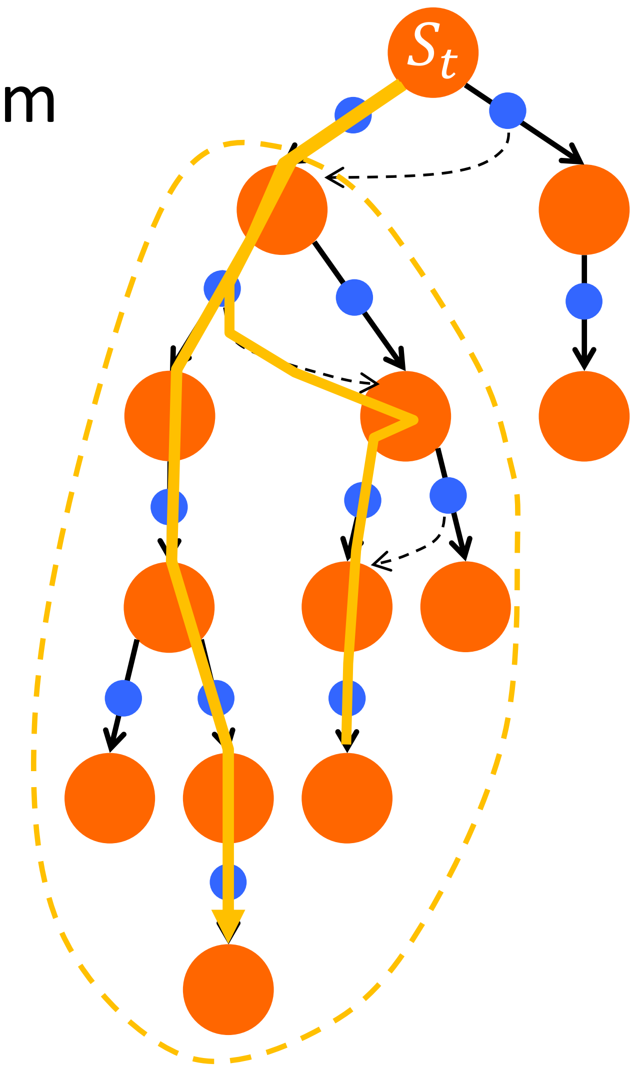


state value function

Action-Value Function

- An action-value function used in the bandit problem is extended to a function of state and action
- It is the expected return starting from s, taking the action a, and thereafter following policy π
  - $s_{t+1}, r_{t+1} \sim p(s', r \mid, s_t, a_t), a_{t+1} \sim \pi(a \mid s_{t+1})$
- The state-action value function is the expected return given by

$$Q^{\pi}(s,a) \triangleq \mathbb{E}_{\pi}[G_t \mid s_t = s, a_t = a]$$



Action-Value Function

- The value of a state s under a policy π is defined by

$$Q^{\pi}(s,a) \triangleq \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^{k} r_{t+k+1} \,|\, s_t = s, a_t = a\right]$$

- $\mathbb{E}_{\pi}\{\quad\}$ denotes the expectation which is taken over the probability distribution

$$P(\, r_{t+1}, s_{t+1}, a_{t+1}, s_{t+2}, \dots \mid s_t = s, a_t = a\,)$$
$$= \prod_{k=0}^{\infty} p(s_{t+k+1}, r_{t+k+1} | s_{t+k}, a_{t+k}) \pi(a_{t+k+1} | s_{t+k+1})$$

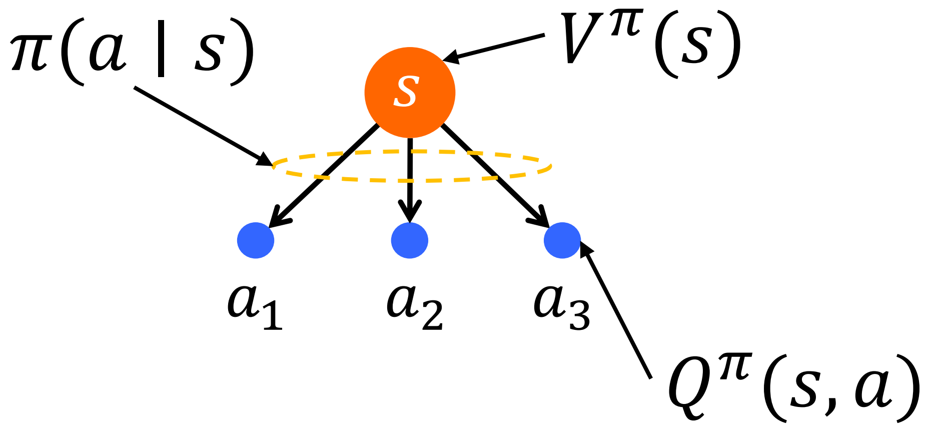
- Therefore,

$$Q^{\pi}(s,a) = \sum_{r_{t+1}, s_{t+1}, a_{t+1}, \dots} P(r_{t+1}, s_{t+1}, \dots | s_t = s, a_t = a) \left[\sum_{k=0}^{\infty} \gamma^{k} r_{t+k+1}\right]$$

Relation between V^π and Q^π

- Relation between $V^\pi(s)$ and $Q^\pi(s,a)$

$$V^\pi(s) = \sum_{a\in\mathcal{A}} \pi(a \mid s) Q^\pi(s,a)$$
$$= \mathbb{E}_\pi[Q^\pi(s,a)]$$

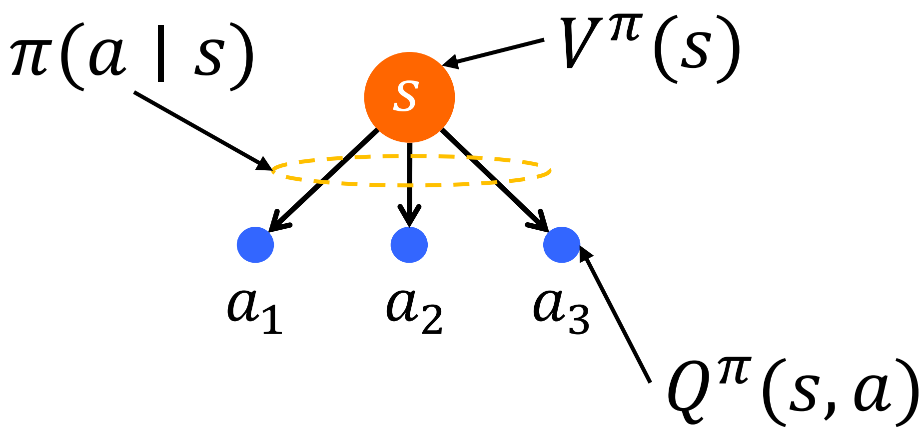


- We have already used this relation

Advantage function

- Advantage function

$$A_\pi(s,a) \triangleq q_\pi(s,a) - V^\pi(s)$$



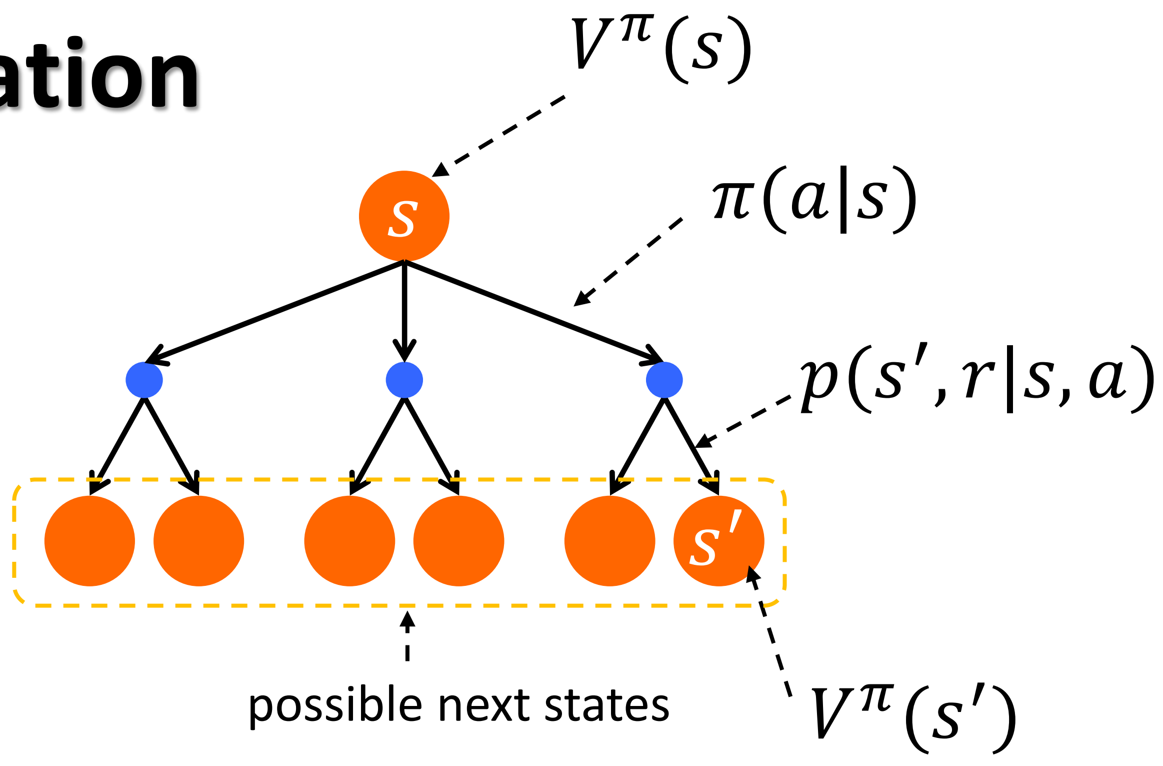
- From the definition,

$$\begin{aligned}\mathbb{E}_\pi[A_\pi(s,a)] &= \mathbb{E}_\pi[Q^\pi(s,a)] - V^\pi(s) \\ &= 0\end{aligned}$$

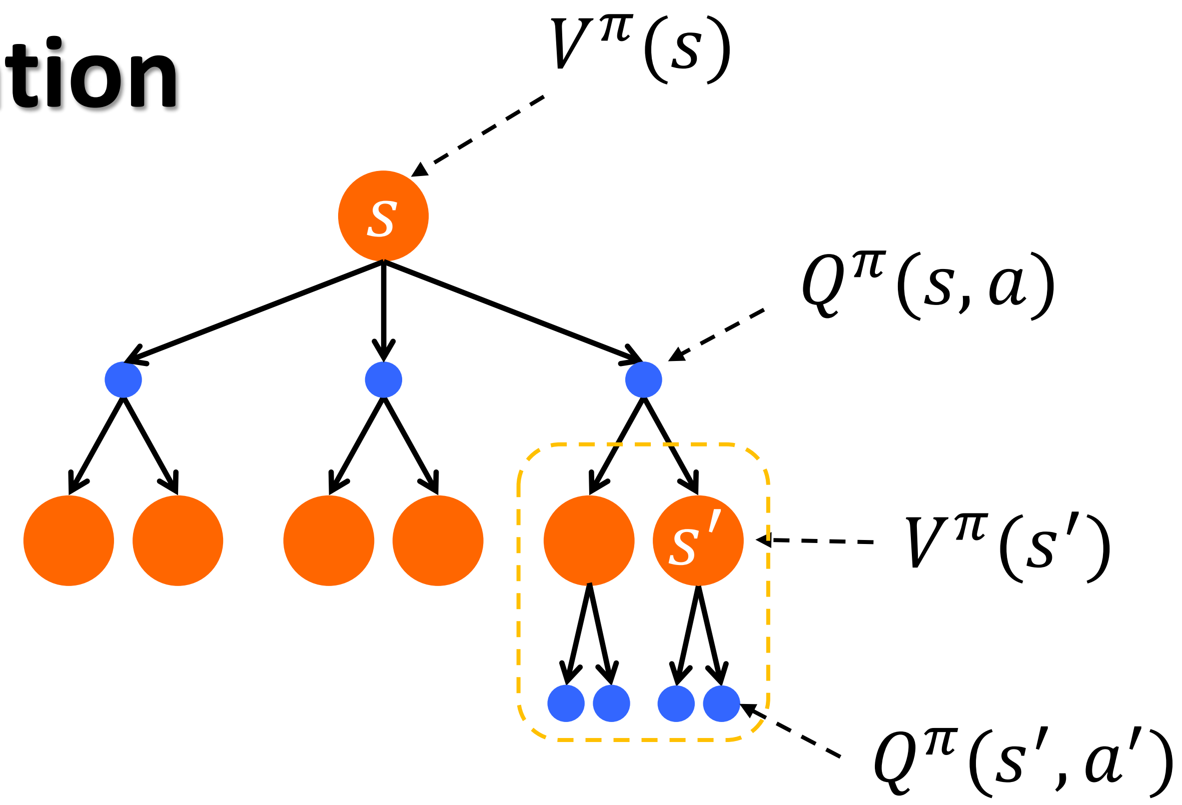
Bellman (Expectation) Equation

- Recursive relationship
- Linear with respect to V^π
- For any policy π and any state s,

$$
\begin{aligned}
V^\pi(s) &= \mathbb{E}_\pi[G_t \mid s_t = s] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid s_t = s] \\
&= \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma \mathbb{E}_\pi[\, G_{t+1} \mid s_{t+1} = s' \,]\right] \\
&= \sum_a \pi(a \mid s) \sum_{s',r} P(s', r \mid s, a)\,[r + \gamma V^\pi(s')]
\end{aligned}
$$



Bellman (Expectation) Equation



- Recursive relationships for Q^π
- Linear with respect to Q^π
- For any policy π and any state-action pair (s,a),

$$Q^\pi(s,a) = \mathbb{E}_\pi[G_t \mid s_t = s, a_t = a]$$

$$= \sum_{s',r} p(s',r \mid s,a)\left[r + \gamma \mathbb{E}_\pi[\, G_{t+1} \mid s_{t+1} = s', a_{t+1} = a'\,]\right]$$

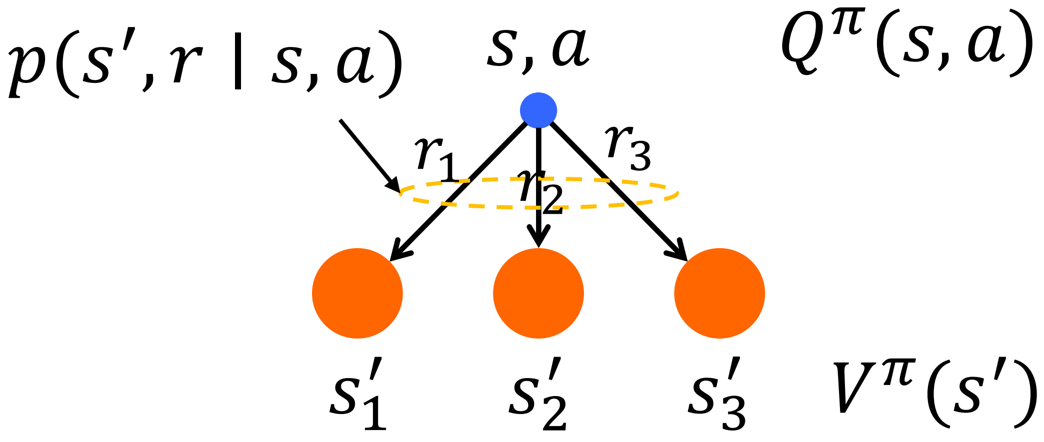
$$= \sum_{s',r} P(s',r \mid s,a)\left[r + \gamma \sum_{a'} \pi(\, a' \mid s'\,) Q^\pi(s',a')\right]$$

Relations between V^π and Q^π

- Using the previous equations, we obtain

$$Q^\pi(s,a) = \sum_{s',r} P(s',r \mid s,a)\left[r + \gamma \sum_{a'} \pi(a' \mid s')Q^\pi(s',a')\right]$$

$$= \sum\nolimits_{s',r} p(s',r \mid s,a)[r + \gamma V^\pi(s')]$$



Optimal Value Functions

- Reminder: The goal of RL is to find an optimal policy π_*
- V^* is an optimal state value function for π^* defined by

$$V^*(s) \triangleq \max_{\pi} V^{\pi}(s) \qquad \forall s \in \mathcal{S}$$

- Similarly, an optimal state-action value function is defined by

$$Q^*(s,a) \triangleq \max_{\pi} Q^{\pi}(s,a) \qquad \forall s \in \mathcal{S}, \forall a \in \mathcal{A}$$

Optimal Value Functions

Open in Colab

Results

Here, p0 is the probability to select the action a_0 at the state s_0. Then, the probability to select a_1 at s_0 is given by $1-$ p0. Similarly, p1 is the probability to select a_0 at s_1. The first array shows the state-value function, $V^{\pi}(s)$. The second 2-D array shows the action-value function, $Q^{\pi}(s,a)$.

```
env = TwoStageMDPEnv()
interact(compute_value, env=fixed(env), p0=(0, 1, 0.1), p1=(0, 1, 0.1), p2=(0, 1, 0.1), p3=(0, 1, 0.1), p4=(0, 1, 0.1))
```

p0 0.50
p1 0.50
p2 0.50
p3 0.50
p4 0.50

$\pi(a_0 \mid s)$

```
(array([[ 7.6125],
        [ 5.    ],
        [ 5.25  ],
        [12.25  ],
        [ 5.25  ],
        [ 0.    ]]), array([[ 5.075, 10.15 ],
        [ 7.   ,  3.   ],
        [ 3.   ,  7.5  ],
        [17.5  ,  7.   ],
        [ 3.   ,  7.5  ],
        [ 0.   ,  0.   ]]))
<function __main__.compute_value(env, p0=0.5, p1=0.5, p2=0.5, p3=0.5, p4=0.5)>
```

$V^{\pi}(s)$

$Q^{\pi}(s,a)$

Try finding the optimal policy

```
[4]
```

Theorem

- A state-value function defines a partial ordering over policies

$$\pi \geq \pi' \quad \text{if} \quad V^{\pi}(s) \geq V^{\pi'}(s), \quad \text{for all } s$$

Theorem

- For any Markov Decision Process

1. There exists an optimal policy π^* that is better than or equal to all other policies, $\pi^* \geq \pi$ for all π

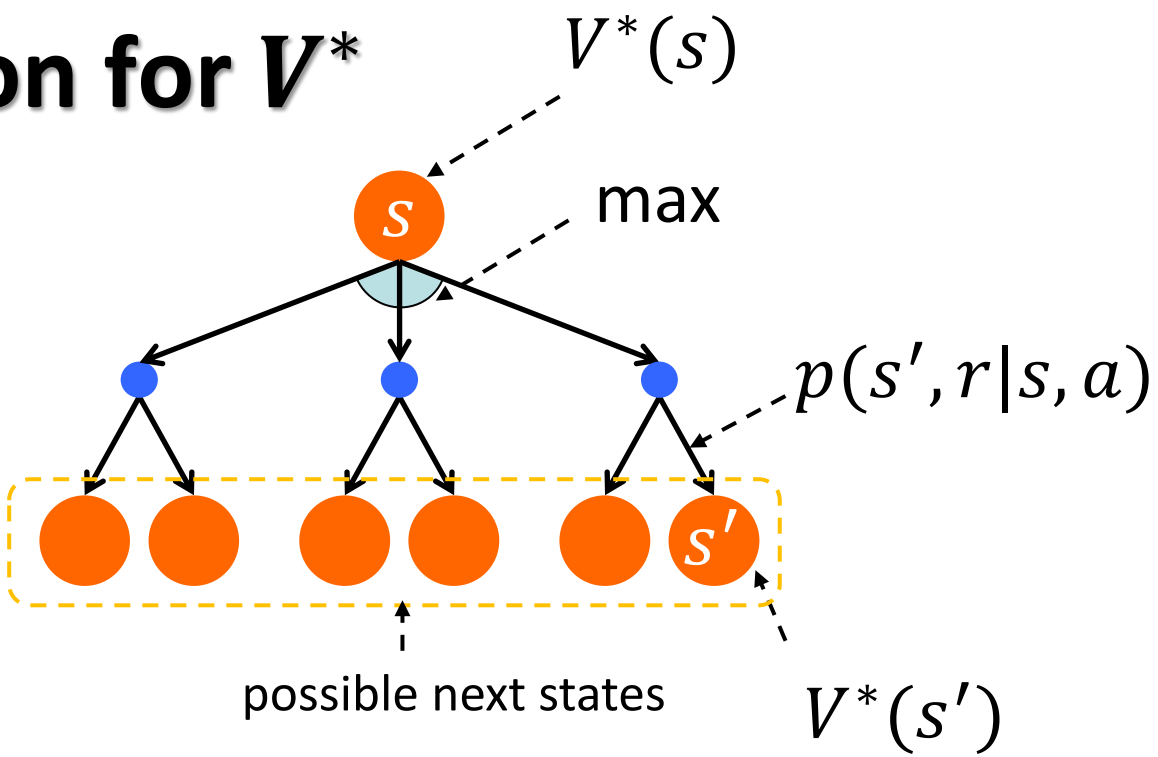
2. All optimal policies achieve the optimal value function,

$$V^{\pi^*}(s) = V^*(s)$$

3. All optimal policies achieve the optimal state-action value function

$$\underbrace{Q^{\pi^*}(s,a)}_{\text{action-value function of the optimal policy}} = \underbrace{Q^*(s,a)}_{\text{optimal action-value function}}$$

Bellman Optimality Equation for V^*



- Recursive relationships
- Nonlinear with respect to V^*
- For any state s,

$$V^*(s) = \max_a Q^*(s,a)$$

$$= \max_a \mathbb{E}_{\pi^*}[G_t \mid s_t = s, a_t = a] = \max_a \mathbb{E}_{\pi^*}[r_{t+1} + \gamma G_t \mid s_t = s, a_t = a]$$

$$= \max_a \mathbb{E}[r_{t+1} + \gamma V^*(s_{t+1}) \mid s_t = s, a_t = a]$$

$$= \max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma V^*(s')]$$

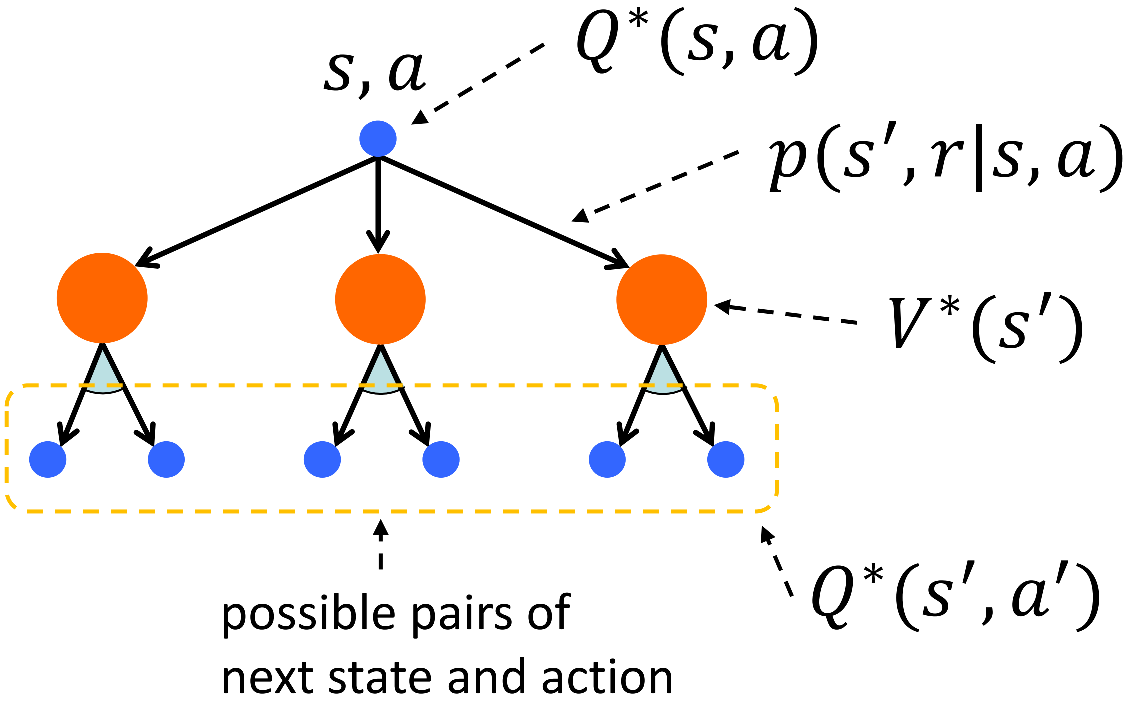
Bellman Optimality Equation for Q^*

- Similarly, the Bellman optimality equation is given by

$$Q^*(s,a) = \mathbb{E}\left[r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a') \mid s_t = s, a_t = a\right]$$

$$= \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma \max_{a'} Q^*(s', a')\right]$$

- This is also a nonlinear equation with respect to Q^* due to the max operator



Optimal Value Functions

- There exists the following relationships:

$$V^*(s) = \mathbb{E}\left[r_{t+1} + \gamma \max_a Q^*(s_{t+1}, a) \mid s_t = s\right]$$

$$Q^*(s, a) = \mathbb{E}[r_{t+1} + \gamma V^*(s_{t+1}) \mid s_t = s, a_t = a]$$

Summary

- Bellman expectation equation

$$V^{\pi}(s) = \sum_{a} \pi(a \mid s) \sum_{s',r} p(s',r \mid s,a) \left[r + \gamma V^{\pi}(s')\right]$$

$$Q^{\pi}(s,a) = \sum_{s',r} p(s',r \mid s,a) \left[r + \gamma \sum_{a'} \pi(a' \mid s') Q^{\pi}(s',a')\right]$$

- Bellman optimality equation

$$V^{*}(s) = \max_{a} \sum_{s',r} p(s',r \mid s,a)\left[r + \gamma V^{*}(s')\right]$$

$$Q^{*}(s,a) = \sum_{s',r} p(s',r \mid s,a)\left[r + \gamma \max_{a'} Q^{*}(s',a')\right]$$