

# MARKOV DECISION PROCESSES

Volker Krüger

Slides partially borrowed from a presentation by David Silver



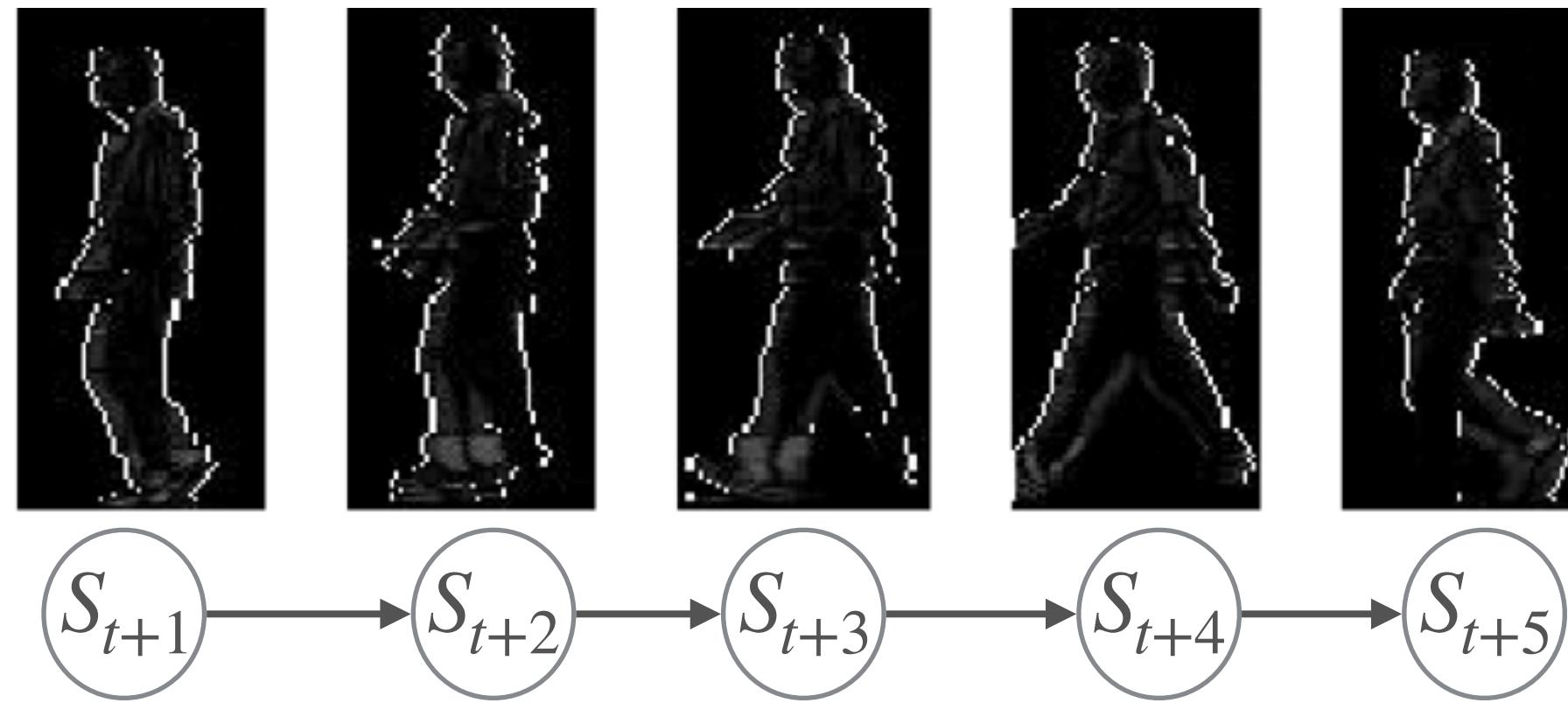
LUND  
UNIVERSITY

# OVERVIEW

- Markov Processes (MP)
- Markov Reward Processes (MRP)
- Markov Decision Processes (MDP)
  - Formal Model of Reinforcement Learning Tasks
  - Value Functions
  - Bellman Equation
  - Optimal Value Function: solving the MDP
- Policy Iteration



# MARKOV PROCESSES



- The next state is dependent only on the present state
- A random state  $S_t$  is Markov if and only if

$$P(S_{t+1} | S_t) = P(S_{t+1} | S_1, \dots, S_t)$$

# STATE TRANSITION MATRIX

- For a Markov state  $s$  and a successor state  $s'$ , the *state transition probability* is defined by

$$P_{ss'} = P(S_{t+1} = s' | S_t = s)$$

- A state transition probability matrix summarizes the possible state transitions

$$\mathbf{P} = \begin{bmatrix} P_{11} & \dots & P_{1n} \\ \vdots & & \\ P_{n1} & \dots & P_{nn} \end{bmatrix}$$

- Note that each row of  $\mathbf{P}$  sums to 1.



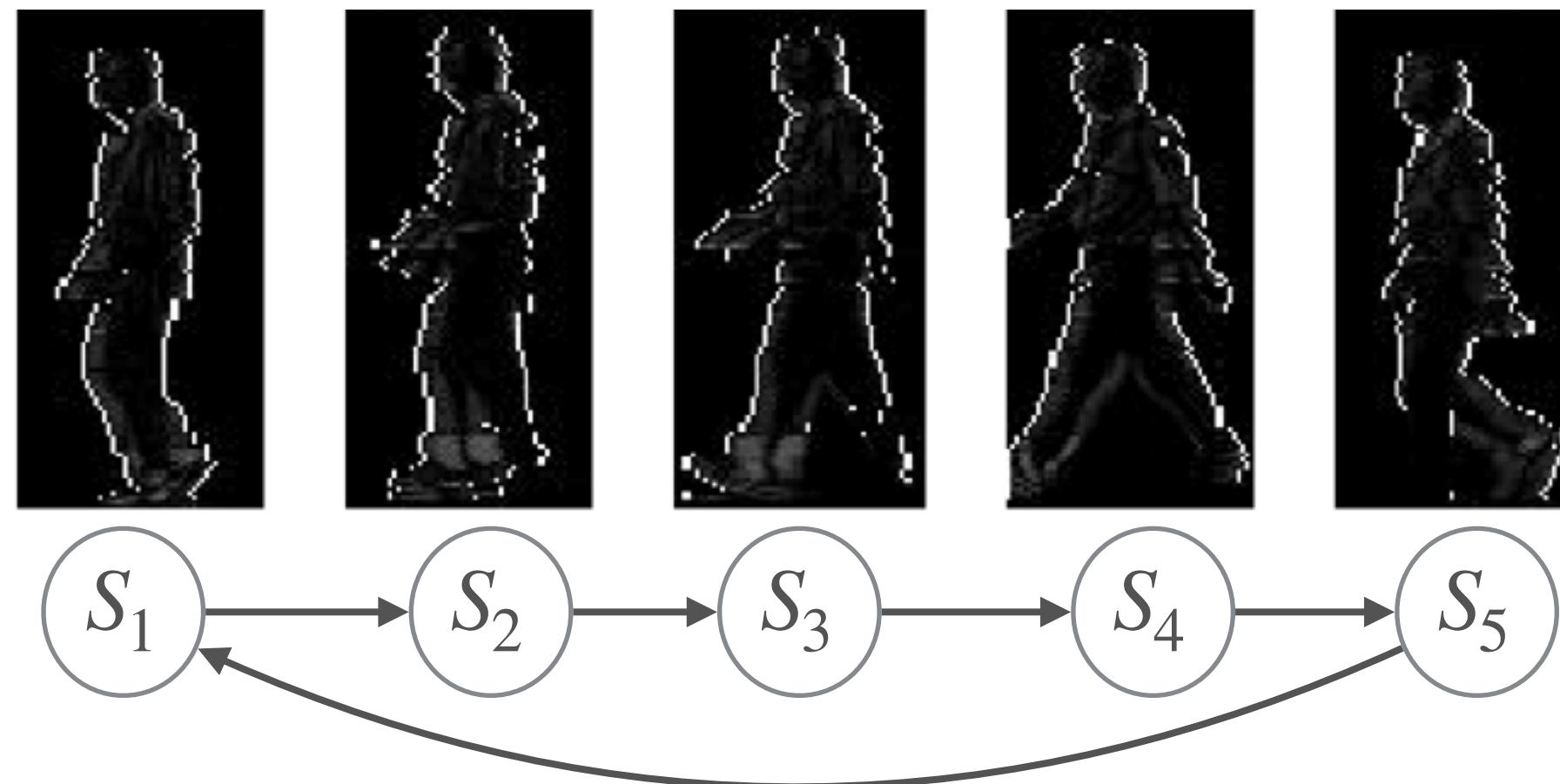
# MARKOV PROCESS

- A Markov process is a sequence  $S_1, S_2, \dots$  of Markov states, i.e., a sequence of random states that have the Markov property
- **Definition:** A Markov process (or Markov Chain) is a tuple  $(\mathbf{S}, \mathbf{P})$ 
  - $\mathbf{S}$  is a (finite) set of states
  - $\mathbf{P}$  is a state transition probability matrix, with

$$P_{ss'} = P(S_{t+1} = s' | S_t = s)$$

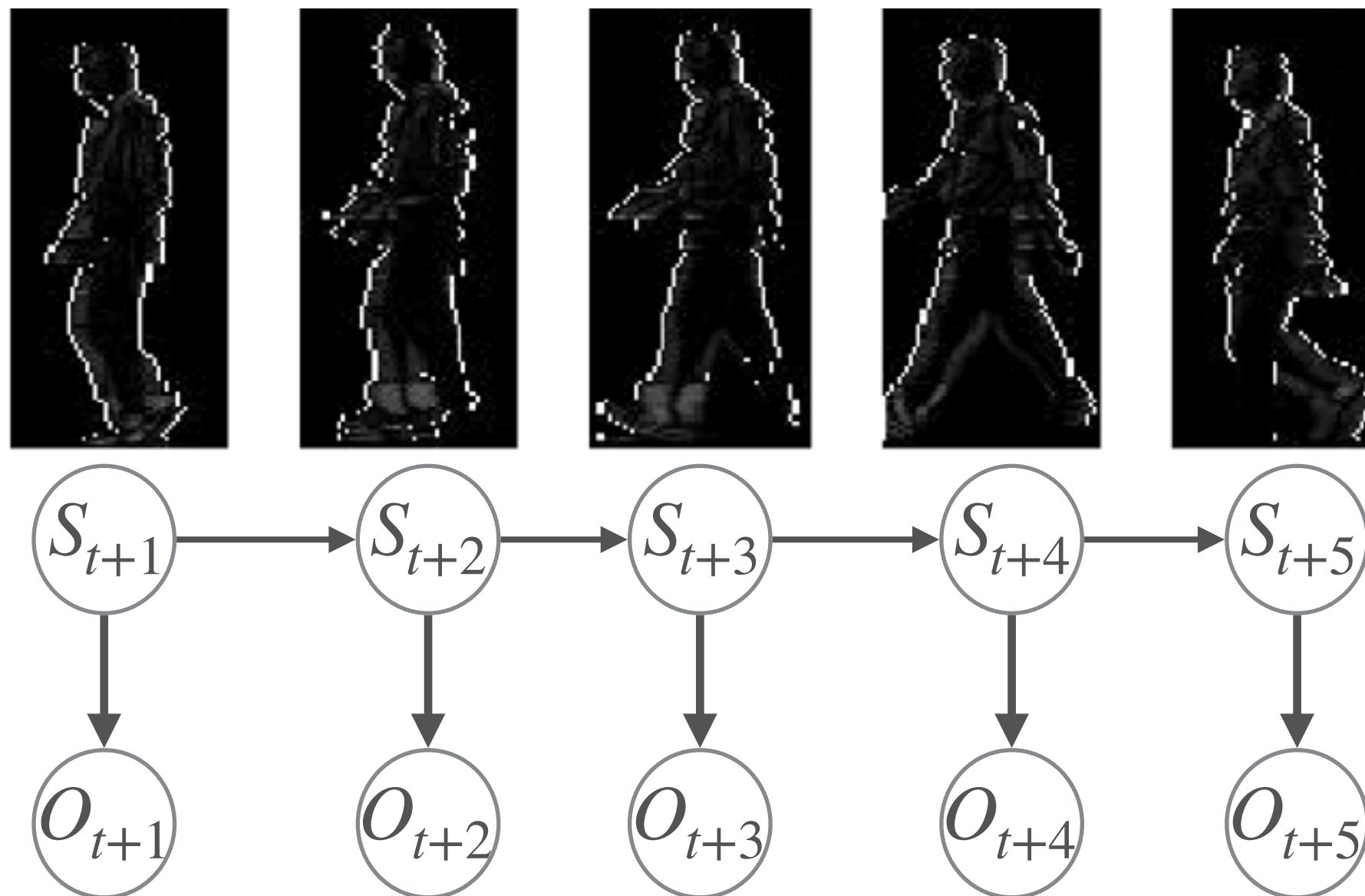


# EXAMPLE GAIT MARKOV CHAIN



LUND  
UNIVERSITY

# HIDDEN MARKOV MODELS



- Extension: Hidden Markov Model:
  - Likelihood of an observation  $o_t$ , given that we are in state  $S_t$

# EXAMPLE: PICKING MARKOV CHAIN



# EXAMPLE: PICKING MARKOV CHAIN



# EXAMPLE: PICKING MARKOV CHAIN



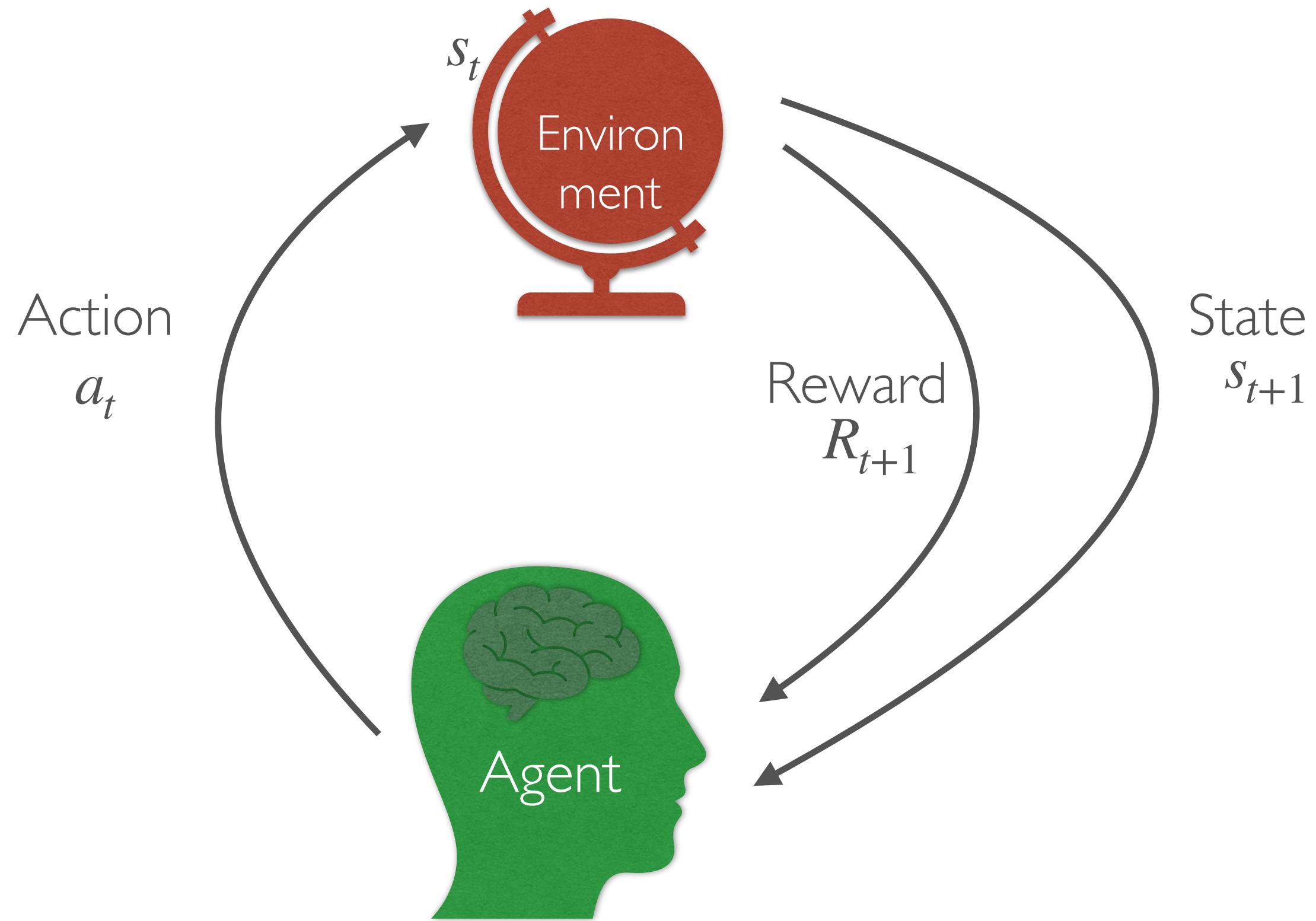
LUND  
UNIVERSITY

# EXAMPLE: PICKING MARKOV CHAIN



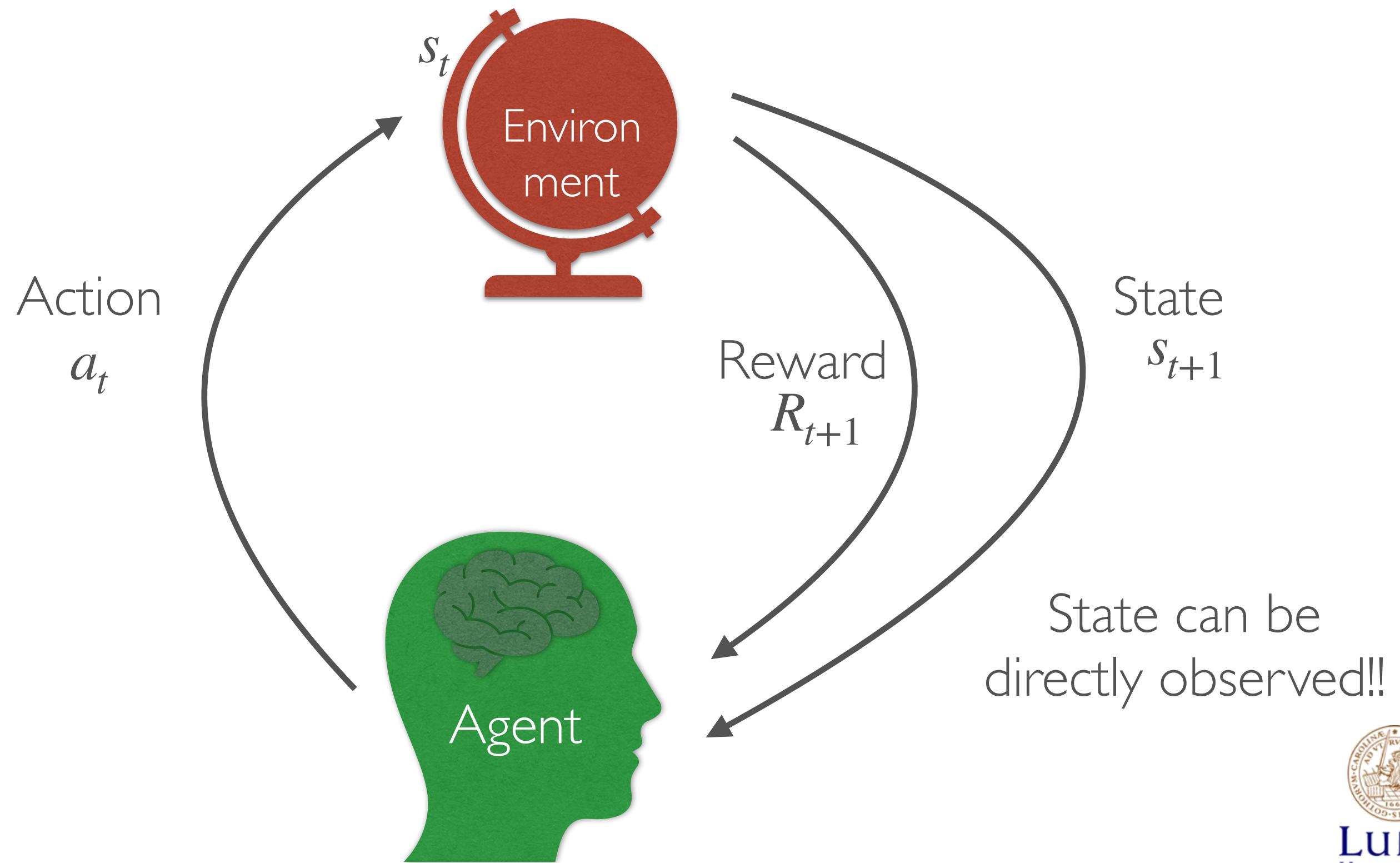
LUND  
UNIVERSITY

# MARKOV DECISION PROCESS

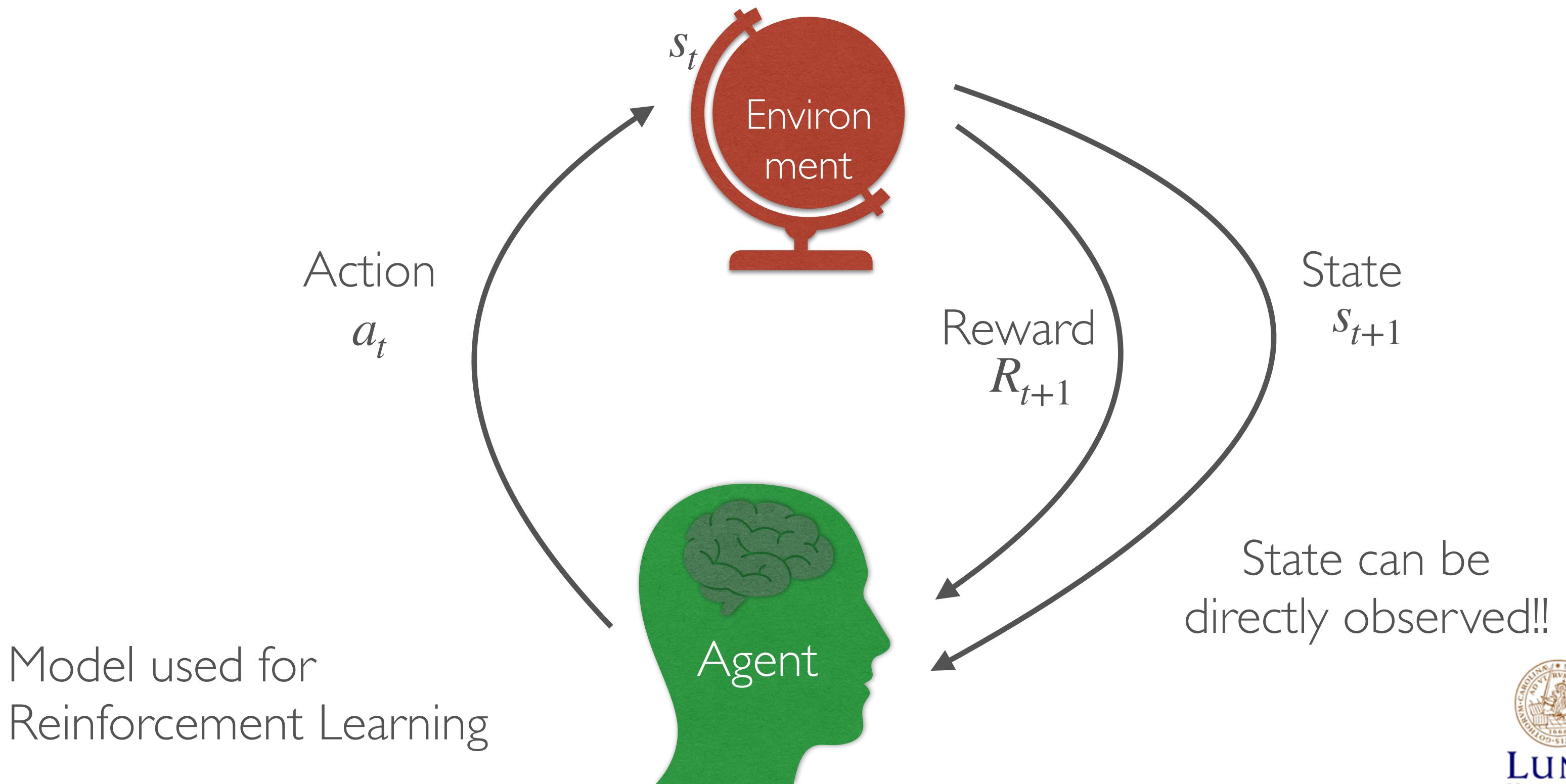


LUND  
UNIVERSITY

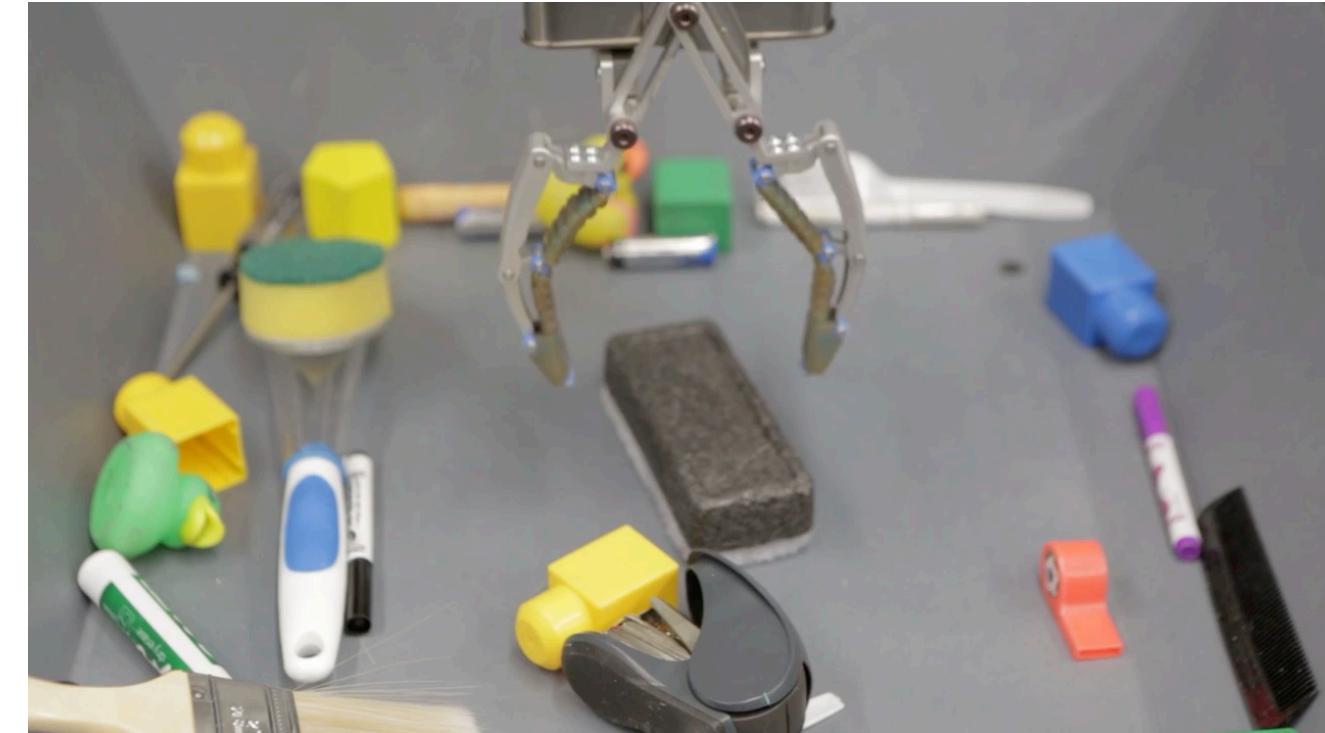
# MARKOV DECISION PROCESS



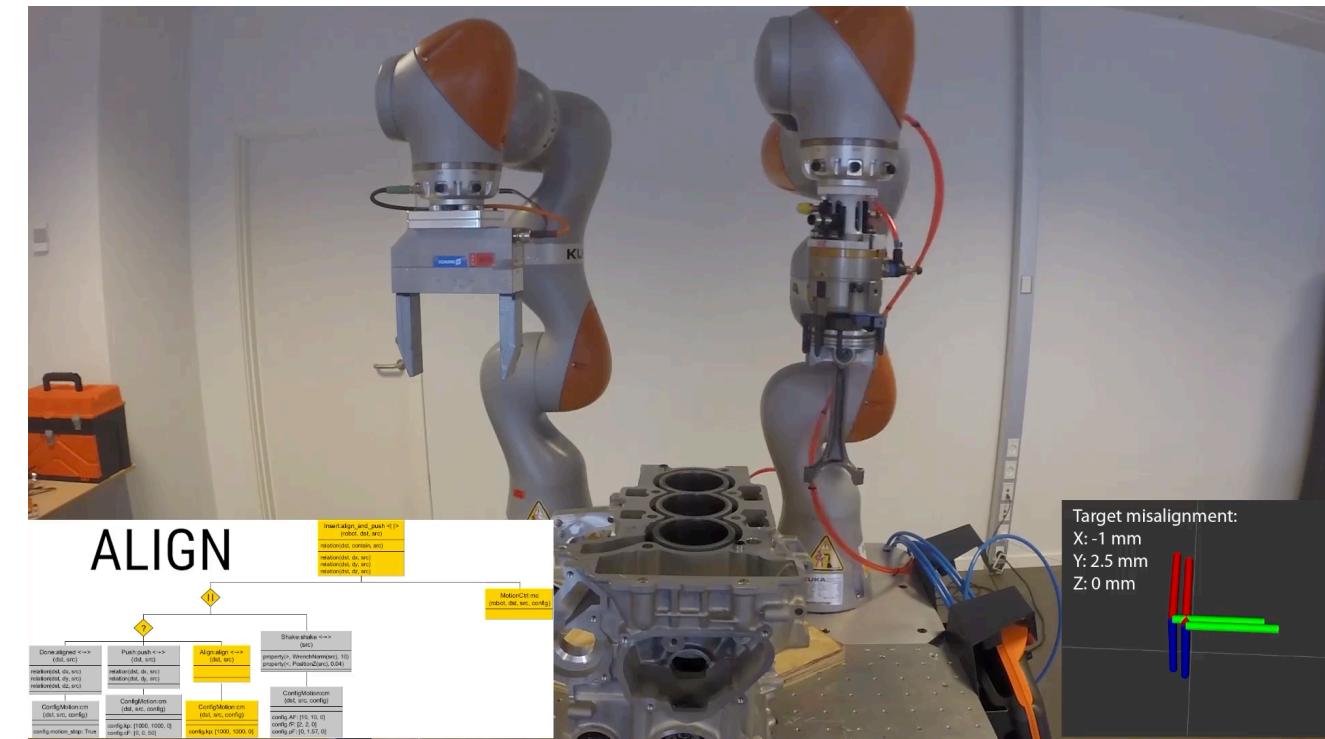
# MARKOV DECISION PROCESS



# REINFORCEMENT LEARNING

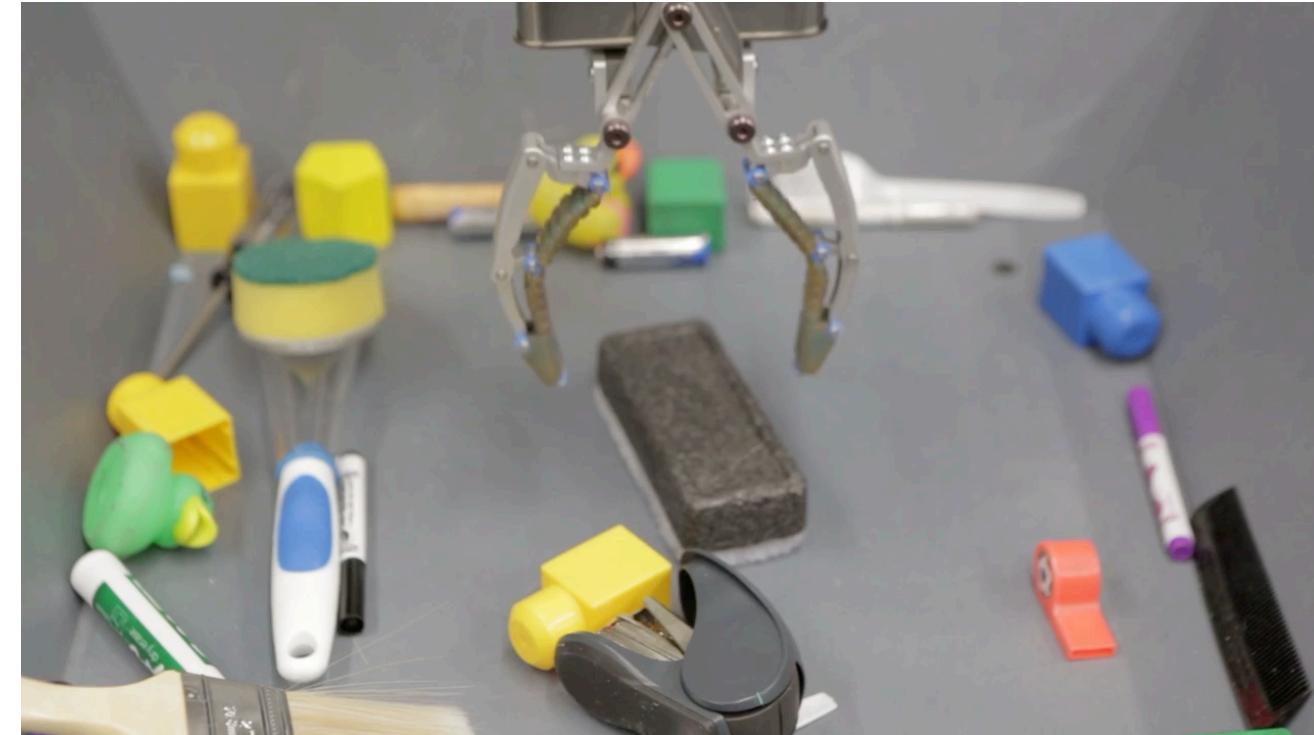


- Value function + Policy modelled using CNNs.
- 14 robots needed 3000h (800.000 grasp attempts) before something intelligent happened

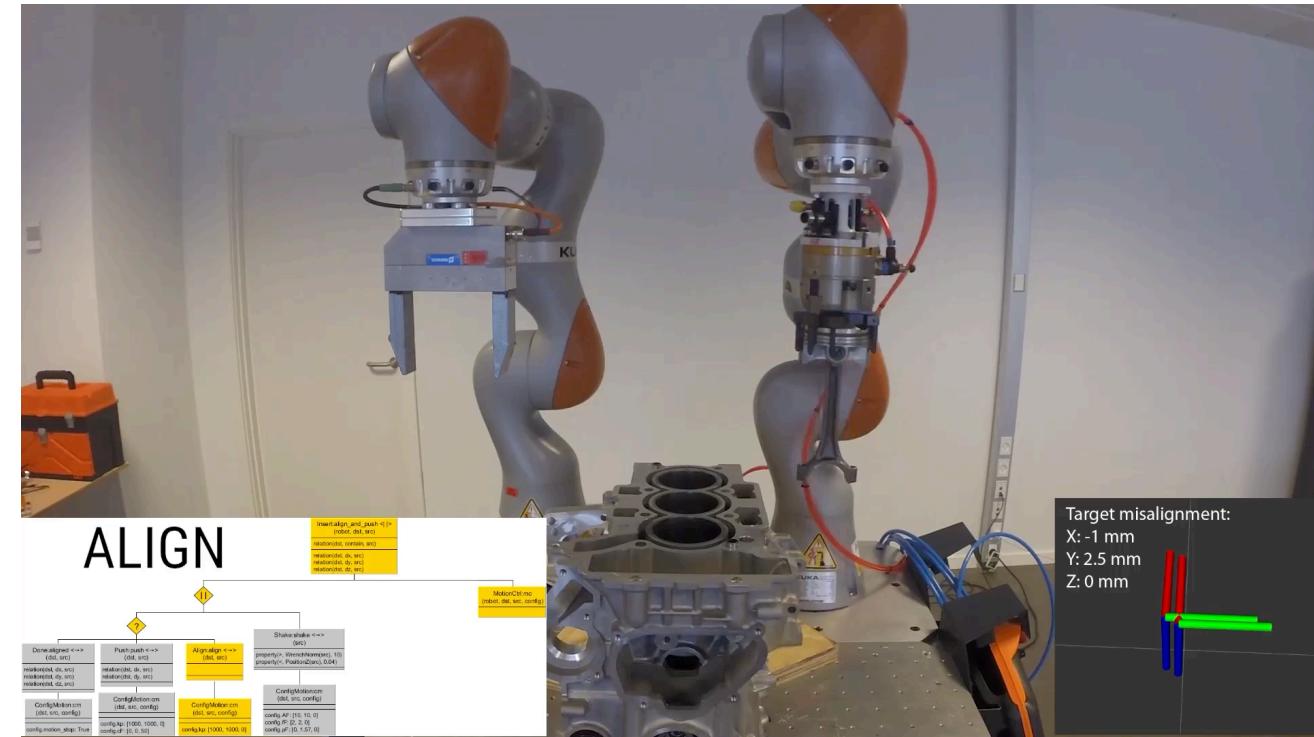


LUND  
UNIVERSITY

# REINFORCEMENT LEARNING

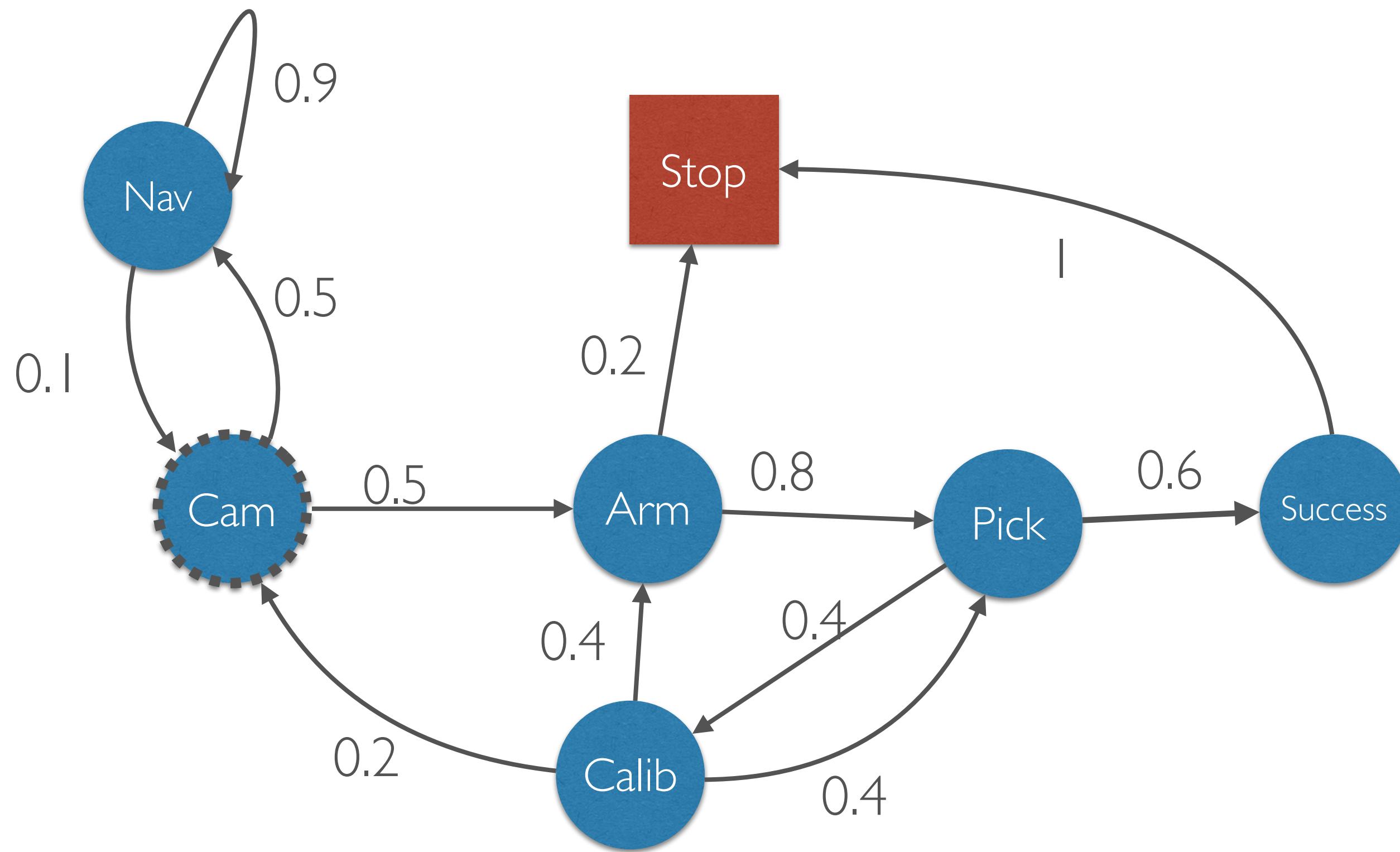


- Value function + Policy modelled using CNNs.
- 14 robots needed 3000h (800.000 grasp attempts) before something intelligent happened

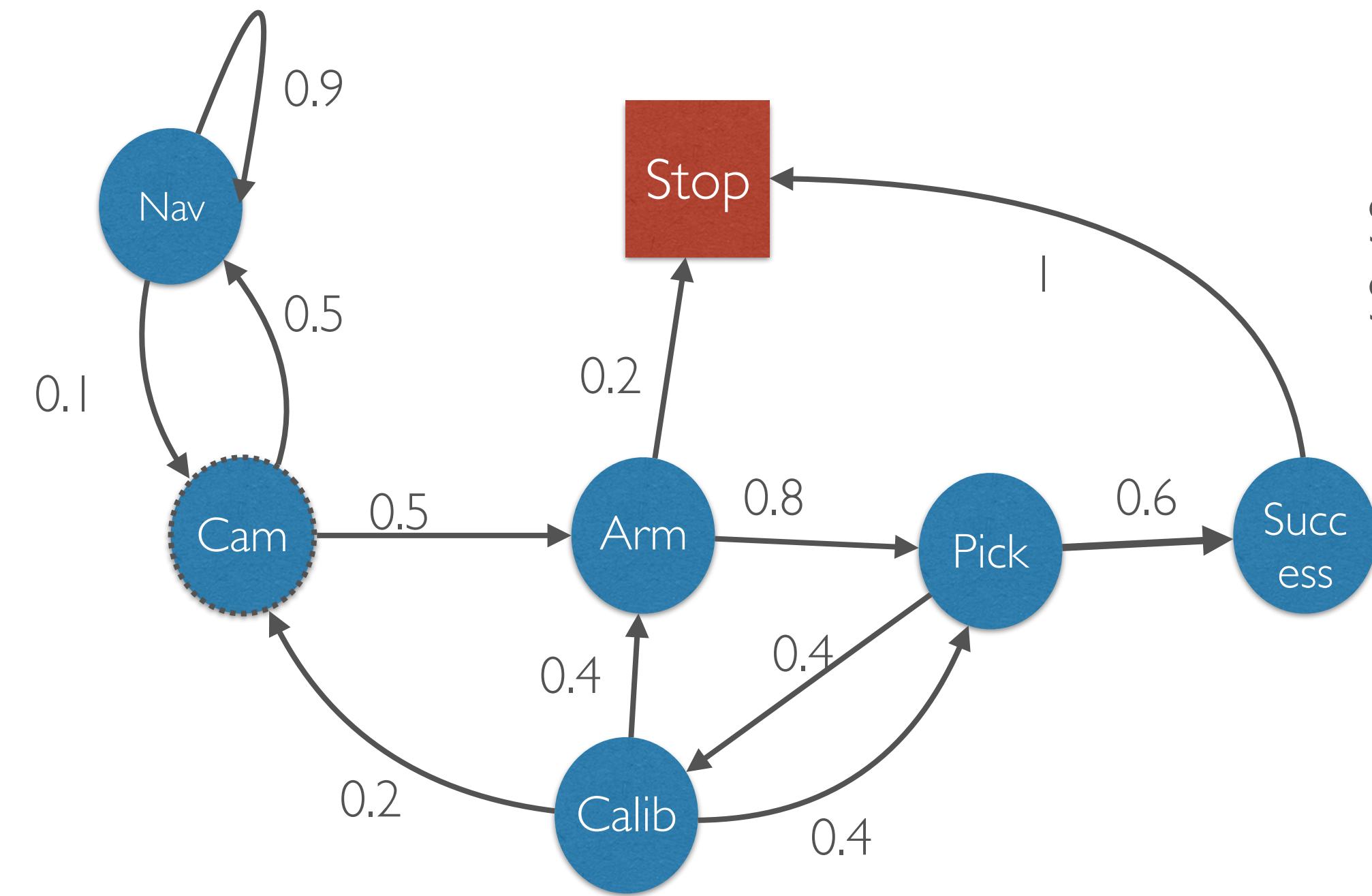


LUND  
UNIVERSITY

# EXAMPLE: PICKING MARKOV CHAIN



# EXAMPLE: PICKING MARKOV CHAIN



Sample **episodes**:

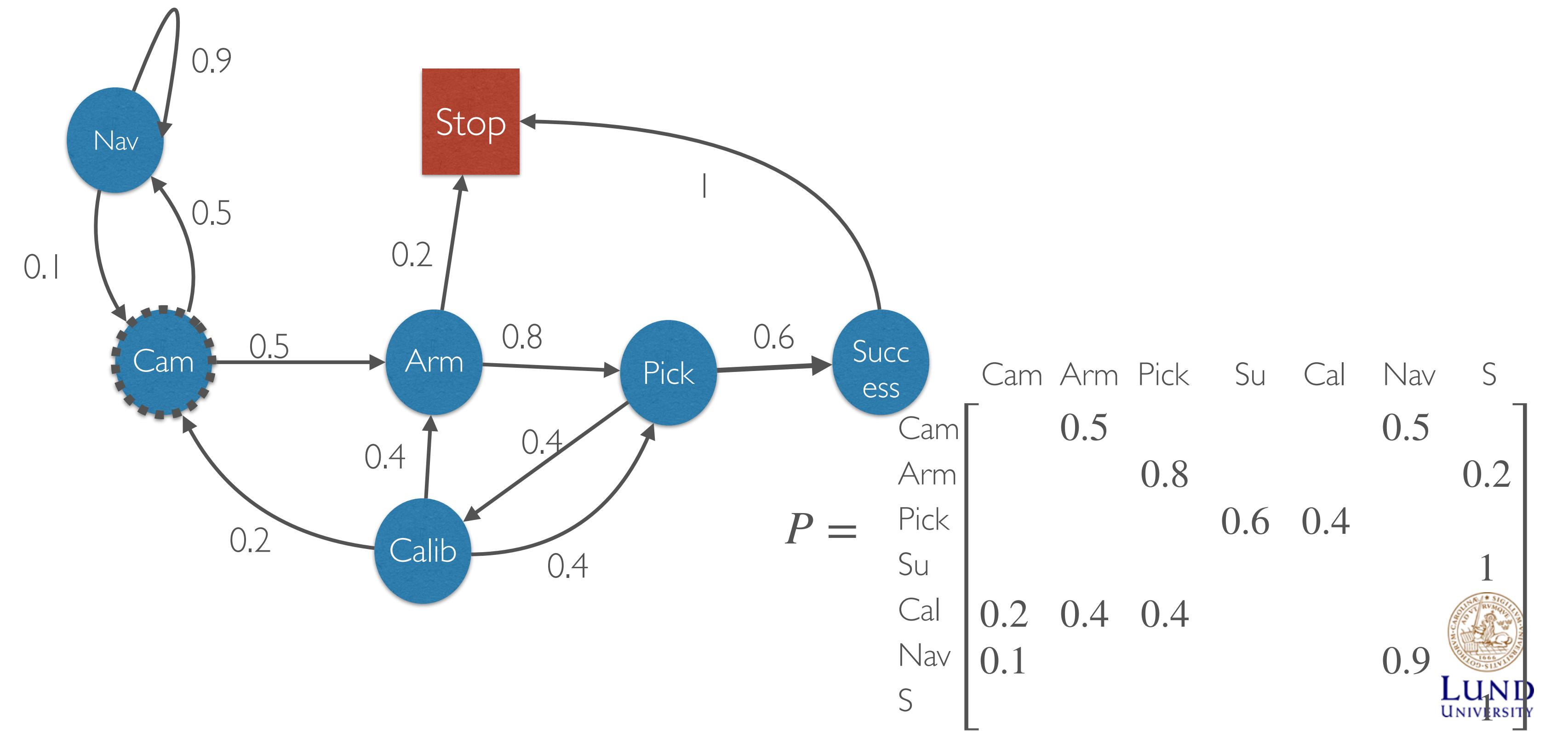
Start state  $S_1 = \text{Cam}$

$S_1, S_2, \dots, S_T$

- Cam, Nav, Cam, Arm, Pick, Su, S
- Cam, Arm, Pick, Cam, Pick, Su, S
- Can, Nav, Cam, Arm, S



# EXAMPLE: PICKING MARKOV CHAIN



# MARKOV REWARD PROCESSES



LUND  
UNIVERSITY

# MARKOV REWARD PROCESS

- **Definition:** A Markov Reward Process is a tupel  $(\mathbf{S}, \mathbf{P}, \mathbf{R}, \gamma)$ 
  - $\mathbf{S}$  is a finite set of states
  - $\mathbf{P}$  is a state transition probability matrix  $P_{ss'} = P(S_{t+1} = s' | S_t = s)$

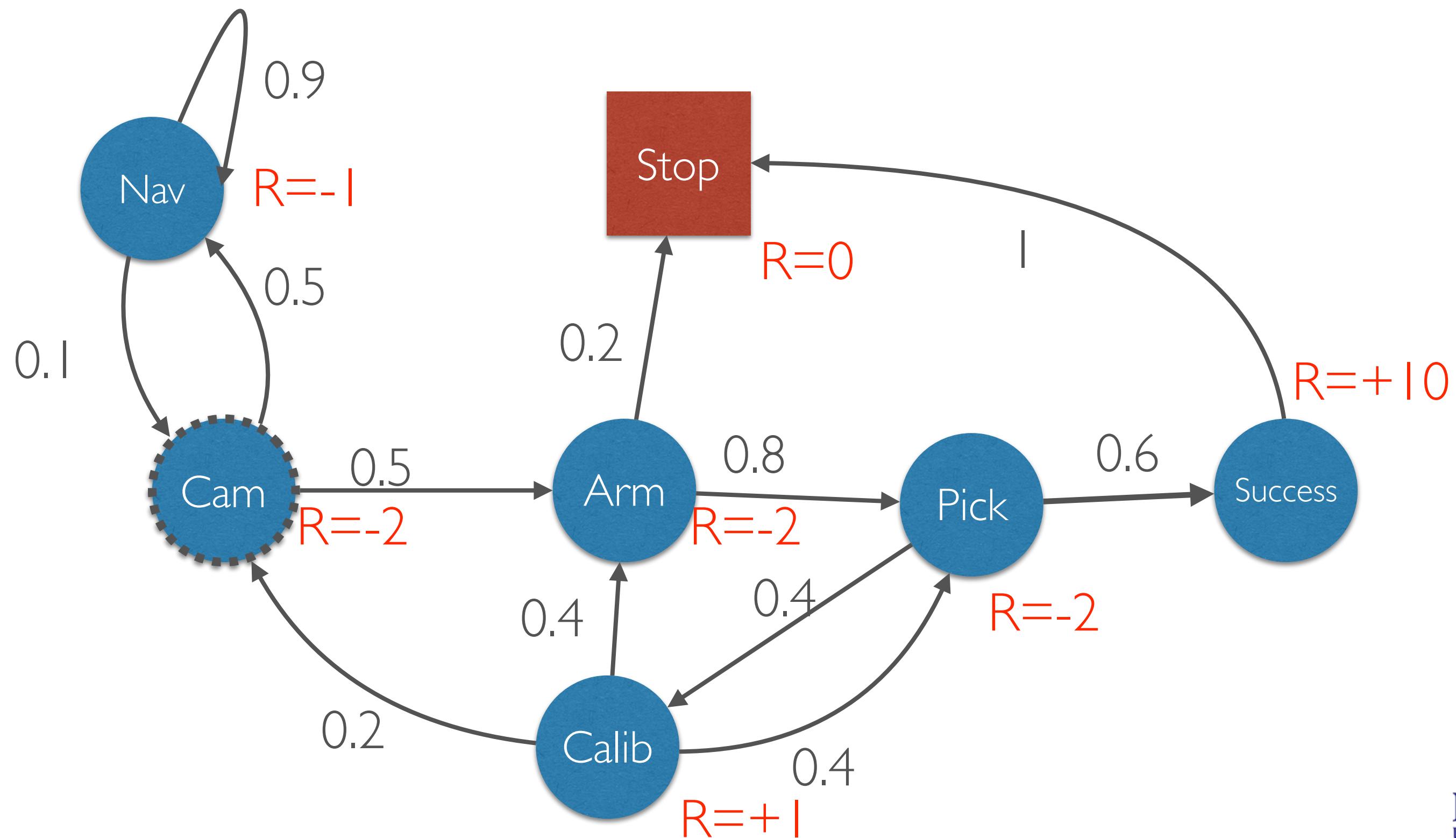


# MARKOV REWARD PROCESS

- **Definition:** A Markov Reward Process is a tupel  $(\mathbf{S}, \mathbf{P}, \mathbf{R}, \gamma)$ 
  - $\mathbf{S}$  is a finite set of states
  - $\mathbf{P}$  is a state transition probability matrix  $P_{ss'} = P(S_{t+1} = s' | S_t = s)$
  - $\mathbf{R}$  is a reward function,  $R_s = E(R_{t+1} | S_t = s)$
  - $\gamma$  is a discount factor,  $\gamma \in [0,1]$



# EXAMPLE: PICKING MARKOV CHAIN



# RETURN

**Definition:** The **return**  $G_t$  is the total discounted reward from time-step  $t$ .

$$G_t = R_{t+1} + \gamma R_{t+2} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

- The *discount*  $\gamma \in [0,1]$  weights the future rewards.



# RETURN

**Definition:** The **return**  $G_t$  is the total discounted reward from time-step  $t$ .

$$G_t = R_{t+1} + \gamma R_{t+2} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

- The *discount*  $\gamma \in [0,1]$  weights the future rewards.
- The value of a reward  $R$  that is  $k+1$  steps into the future is  $\gamma^k R$



# RETURN

**Definition:** The **return**  $G_t$  is the total discounted reward from time-step  $t$ .

$$G_t = R_{t+1} + \gamma R_{t+2} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

- The *discount*  $\gamma \in [0,1]$  weights the future rewards.
- The value of a reward  $R$  that is  $k+1$  steps into the future is  $\gamma^k R$ 
  - Short-sighted:  $\gamma$  close to 0



# RETURN

**Definition:** The **return**  $G_t$  is the total discounted reward from time-step  $t$ .

$$G_t = R_{t+1} + \gamma R_{t+2} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

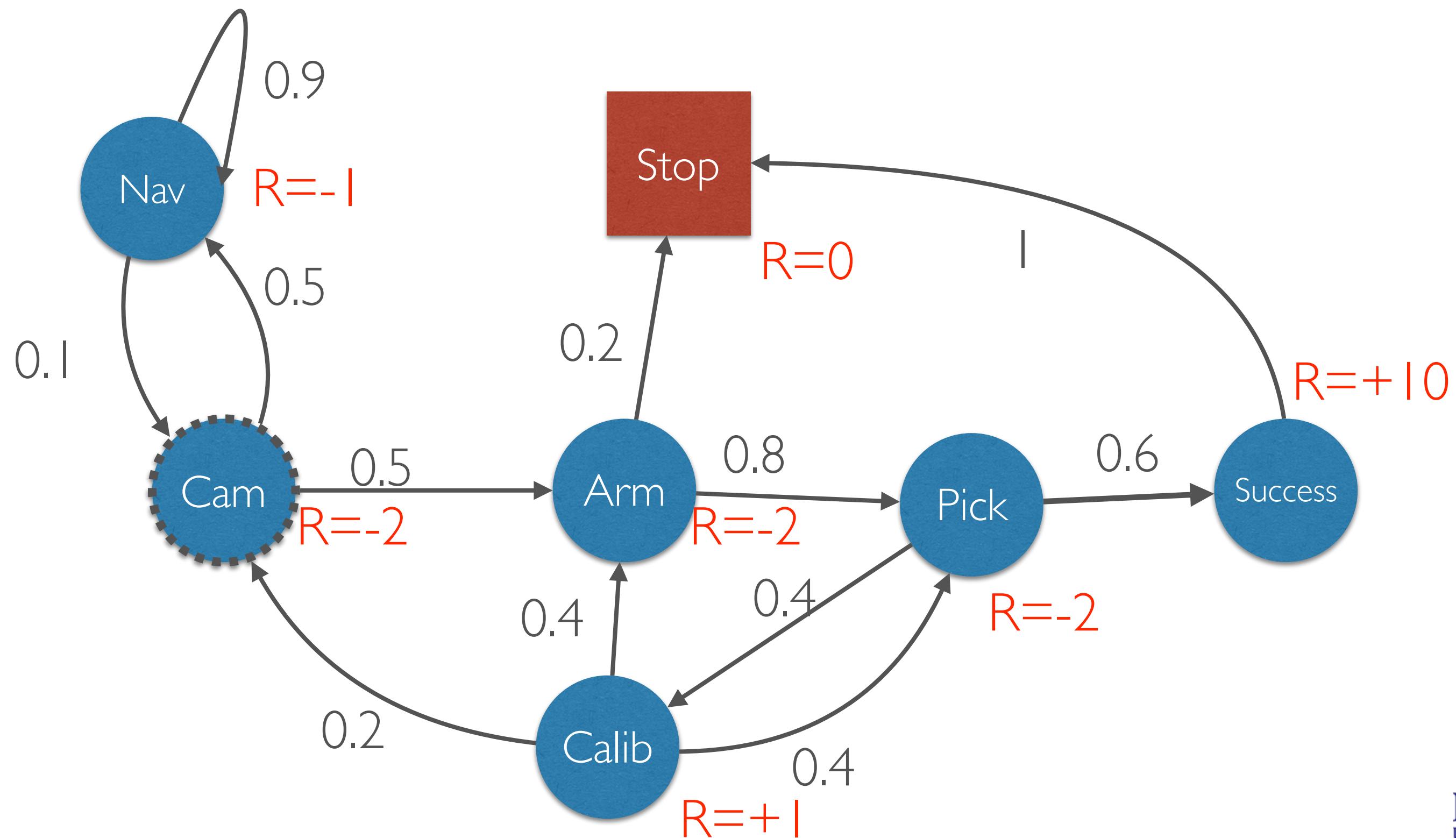
- The *discount*  $\gamma \in [0,1]$  weights the future rewards.
- The value of a reward  $R$  that is  $k+1$  steps into the future is  $\gamma^k R$ 
  - Short-sighted:  $\gamma$  close to 0
  - Far-sighted:  $\gamma$  close to 1

# ROLE OF THE DISCOUNT FACTOR

- Mathematically convenient as  $\gamma^k$  vanishes with  $k$  for  $\gamma < 1$
- Assures finite returns even in cyclic Markov processes
- Represents the uncertainty about the future rewards
- Animal/human behaviour shows preference for immediate reward
- Possible to use *undiscounted* Markov reward process ( $\gamma = 1$ ) if all sequences terminate (assures finite return)



# EXAMPLE: PICKING MARKOV CHAIN



If we get rewards, how good is it to be in a certain state



LUND  
UNIVERSITY

# STATE VALUE FUNCTION

- The state value function  $v(s)$  gives the long-term value of a state  $s$

**Definition:** The **state value function**  $v(s)$  of a MRP is the expected return starting from state  $s$

$$v(s) = E \{ G_t | S_t = s \}$$



# REWARD OF THE PICKING NETWORK

Sample **returns**:  $S_1 = \text{Cam with } \gamma = \frac{1}{2}$

$$G_1 = R_2 + \gamma R_3 + \dots + \gamma^{T-2} R_T$$

Sample **episodes**:

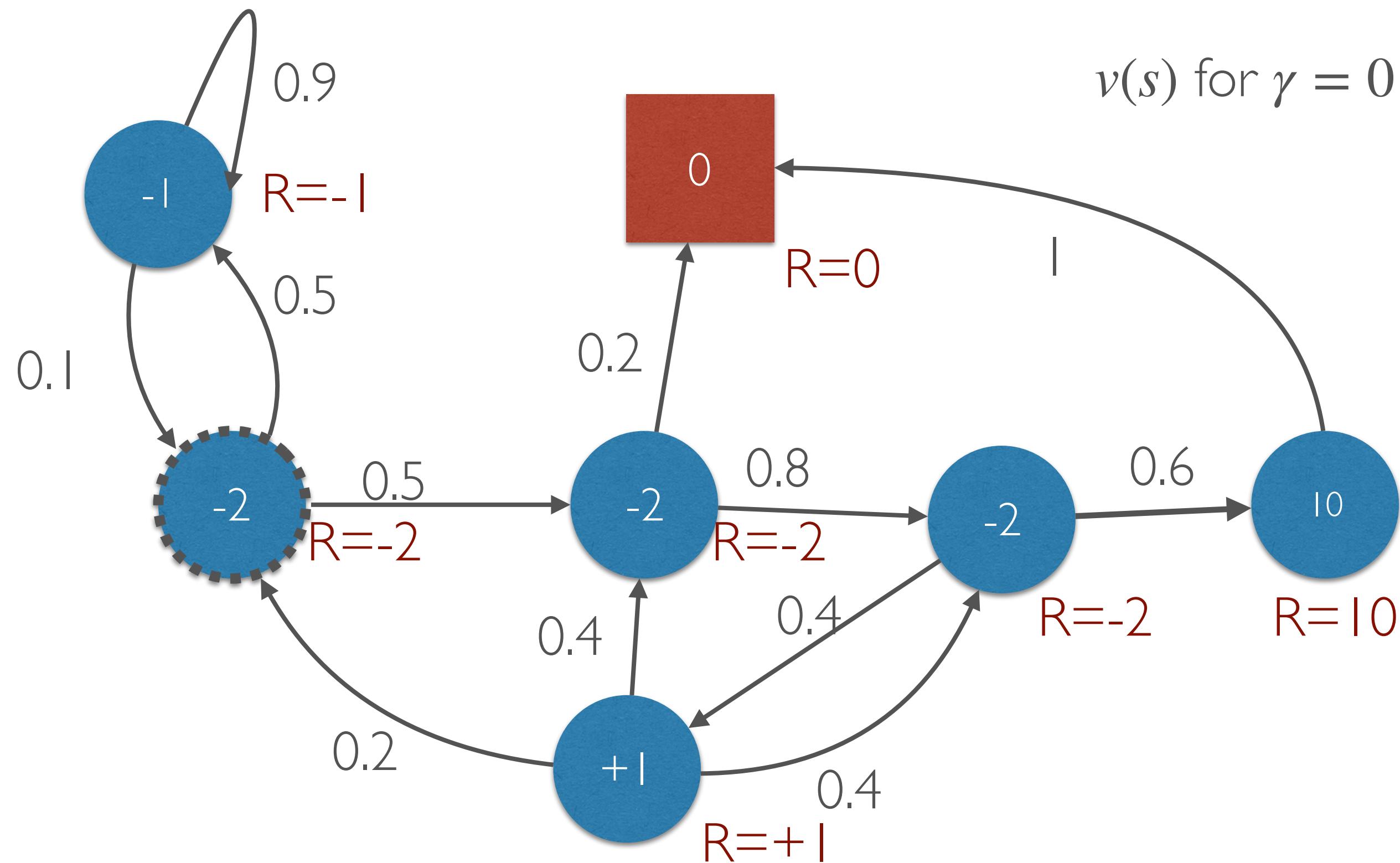
- Cam, Nav, Cam, Arm, Pick, Su, S
- Cam, Arm, Pick, Cal, Pick, Su, S
- Can, Nav, Cam, Arm, S

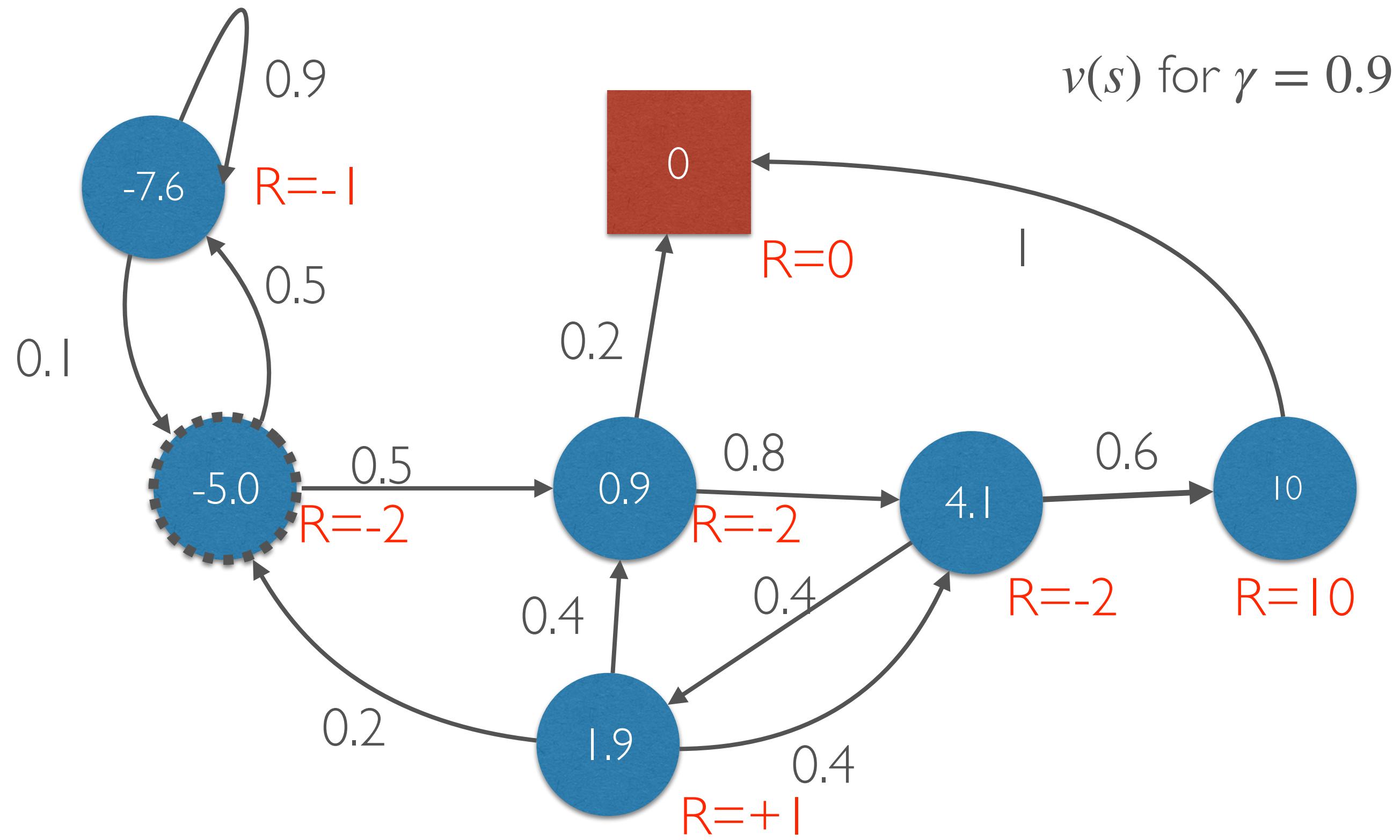
$$v_1 = -2 + \frac{1}{2}(-1) + \frac{1}{4}(-2) + \frac{1}{8}(-2) + \frac{1}{16}(-2) + \frac{1}{32}10 = -3.28$$

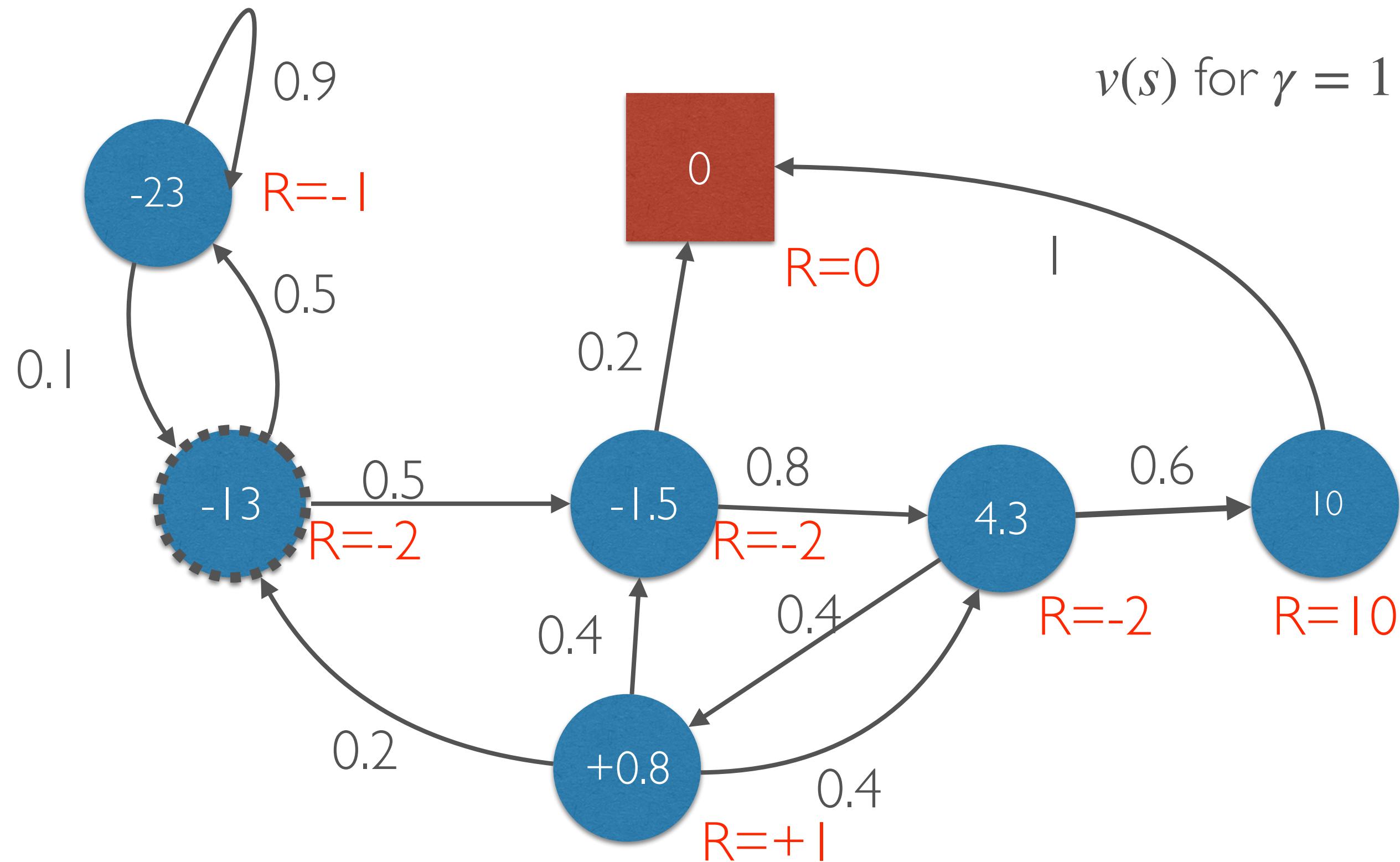
$$v_1 = -2 + \frac{1}{2}(-2) + \frac{1}{4}(-2) + \frac{1}{8}(1) + \frac{1}{16}(-2) + \frac{1}{32}10 = -3.18$$

$$v_1 = -2 + \frac{1}{2}(-2) + \frac{1}{4}(-1) + \frac{1}{8}(-2) + \frac{1}{16}(-2) = -3.62$$









# BELLMAN EQUATION FOR MRP

- The value function can be compute recursively
  - The immediate reward  $R_{t+1}$
  - The discounted future rewards  $\gamma v(S_{t+1})$



# BELLMAN EQUATION FOR MRP

- The value function can be compute recursively
  - The immediate reward  $R_{t+1}$
  - The discounted future rewards  $\gamma v(S_{t+1})$

$$v(s) = E \{ G_t | S_t = s \}$$



LUND  
UNIVERSITY

# BELLMAN EQUATION FOR MRP

- The value function can be compute recursively

- The immediate reward  $R_{t+1}$
- The discounted future rewards  $\gamma v(S_{t+1})$

$$\begin{aligned}v(s) &= E \left\{ G_t \mid S_t = s \right\} \\&= E \left\{ R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \mid S_t = s \right\}\end{aligned}$$



LUND  
UNIVERSITY

# BELLMAN EQUATION FOR MRP

- The value function can be compute recursively

- The immediate reward  $R_{t+1}$

- The discounted future rewards  $\gamma v(S_{t+1})$

$$\begin{aligned}v(s) &= E \left\{ G_t \mid S_t = s \right\} \\&= E \left\{ R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \mid S_t = s \right\} \\&= E \left\{ R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} + \dots) \mid S_t = s \right\}\end{aligned}$$



# BELLMAN EQUATION FOR MRP

- The value function can be compute recursively

- The immediate reward  $R_{t+1}$

- The discounted future rewards  $\gamma v(S_{t+1})$

$$\begin{aligned}v(s) &= E \{ G_t | S_t = s \} \\&= E \{ R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s \} \\&= E \{ R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} + \dots) | S_t = s \} \\&= E \{ R_{t+1} + \gamma (G_{t+1}) | S_t = s \}\end{aligned}$$



# BELLMAN EQUATION FOR MRP

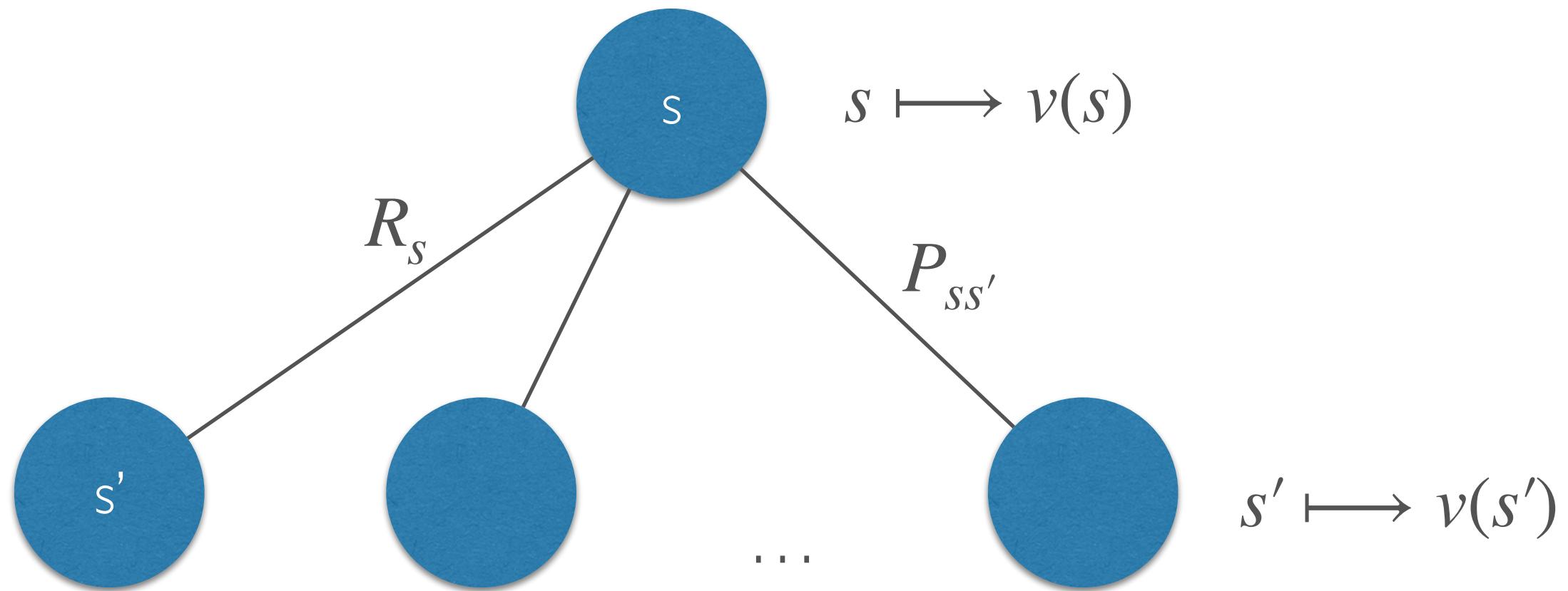
- The value function can be compute recursively

- The immediate reward  $R_{t+1}$

- The discounted future rewards  $\gamma v(S_{t+1})$

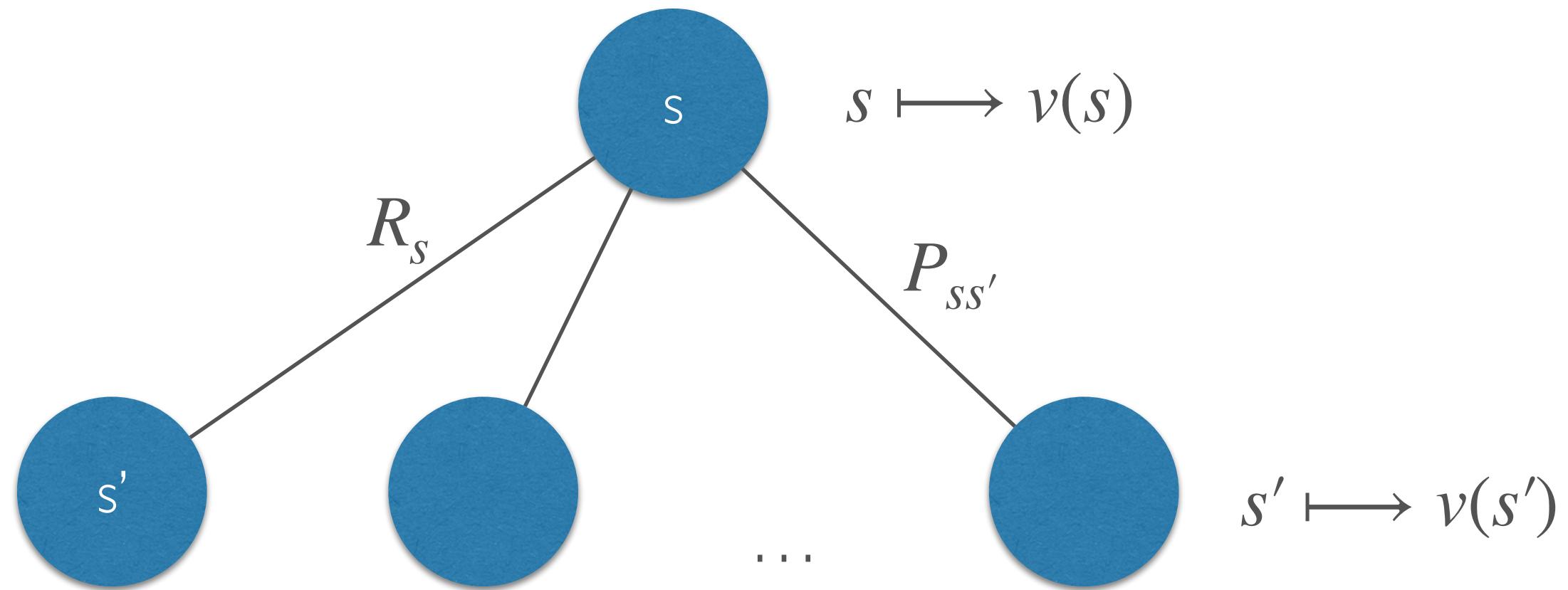
$$\begin{aligned}v(s) &= E \{ G_t | S_t = s \} \\&= E \{ R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s \} \\&= E \{ R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} + \dots) | S_t = s \} \\&= E \{ R_{t+1} + \gamma (G_{t+1}) | S_t = s \} \\&= E \{ R_{t+1} + \gamma v(S_{t+1}) | S_t = s \}\end{aligned}$$

$$v(s) = E \left\{ R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s \right\}$$



LUND  
UNIVERSITY

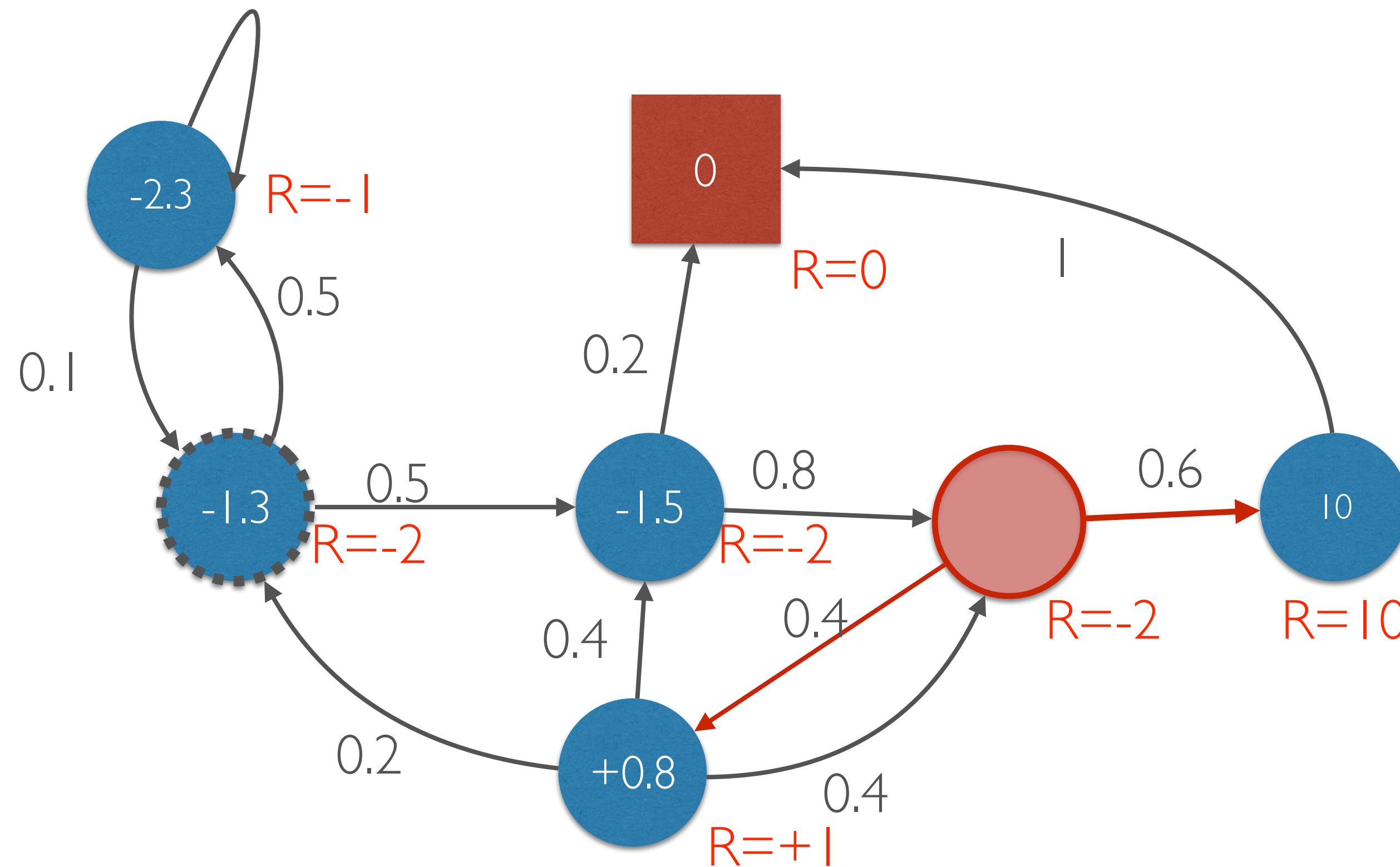
$$v(s) = E \left\{ R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s \right\}$$



Marginalizing over the next possible states:

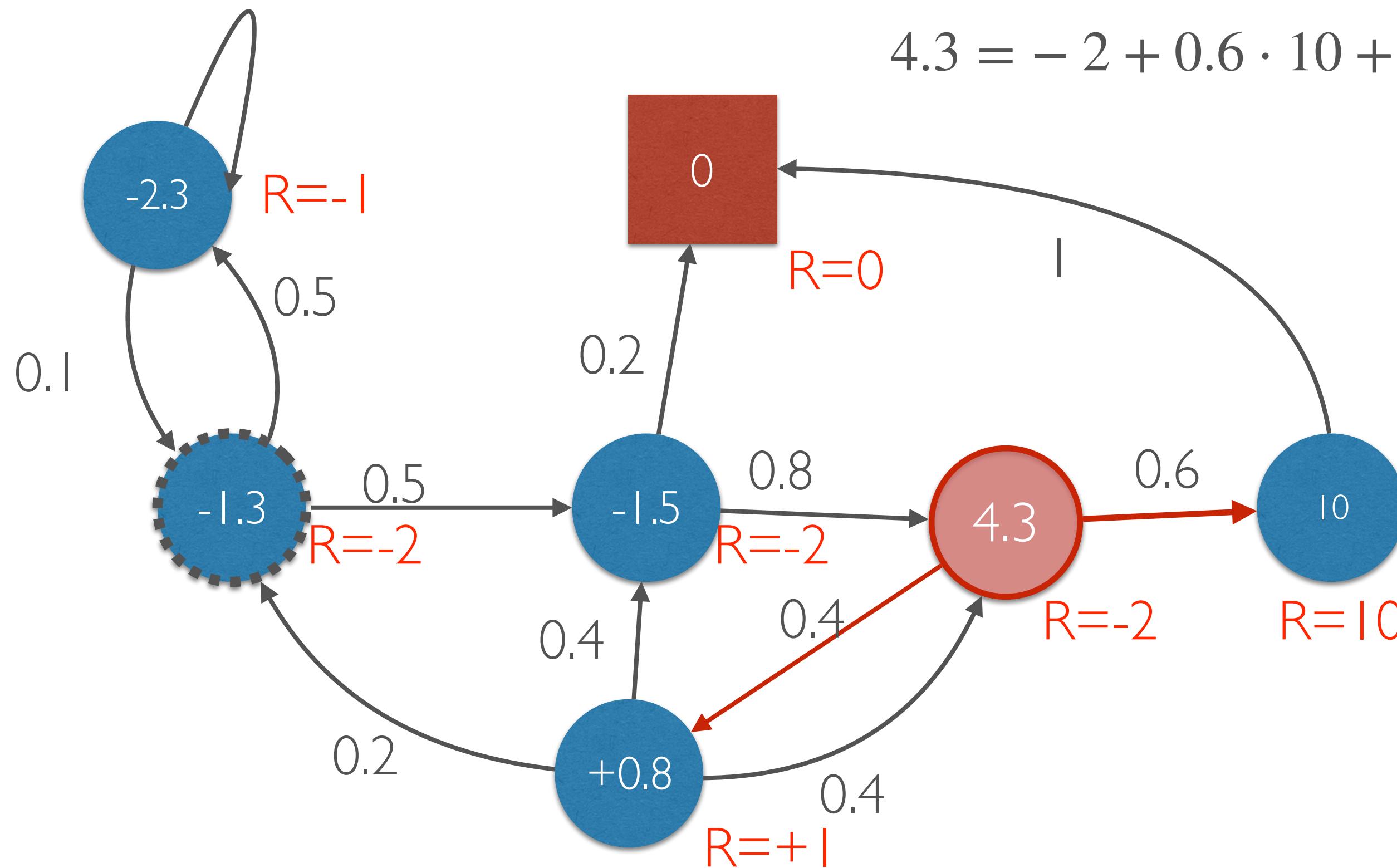
$$v(s) = R_s + \gamma \sum_{s' \in S} P_{ss'} v(s')$$

$$v(s) = R_s + \gamma \sum_{s' \in S} P_{ss'} v(s') \quad \gamma = 1$$



$$v(s) = R_s + \gamma \sum_{s' \in S} P_{ss'} v(s') \quad \gamma = 1$$

$$4.3 = -2 + 0.6 \cdot 10 + 0.4 \cdot 0.8$$



# MARKOV DECISION PROCESSES



LUND  
UNIVERSITY

# MARKOV DECISION PROCESS

- A Markov Decision Process is a Markov Reward Process with decisions on actions

**Definition:** A **Markov Decision Process** is a tupel  $(S, A, P, R, \gamma)$

- $S$  is a finite set of states
- $A$  is a finite set of actions
- $P$  is a state transition probability matrix  $P_{ss'}^{\textcolor{red}{a}} = P(S_{t+1} = s' | S_t = s, A_t = \textcolor{red}{a})$
- $R$  is a reward function  $R_s^{\textcolor{red}{a}} = E(R_{t+1} | S_t = s, A_t = \textcolor{red}{a})$
- $\gamma$  is a discount factor  $\gamma \in [0,1]$



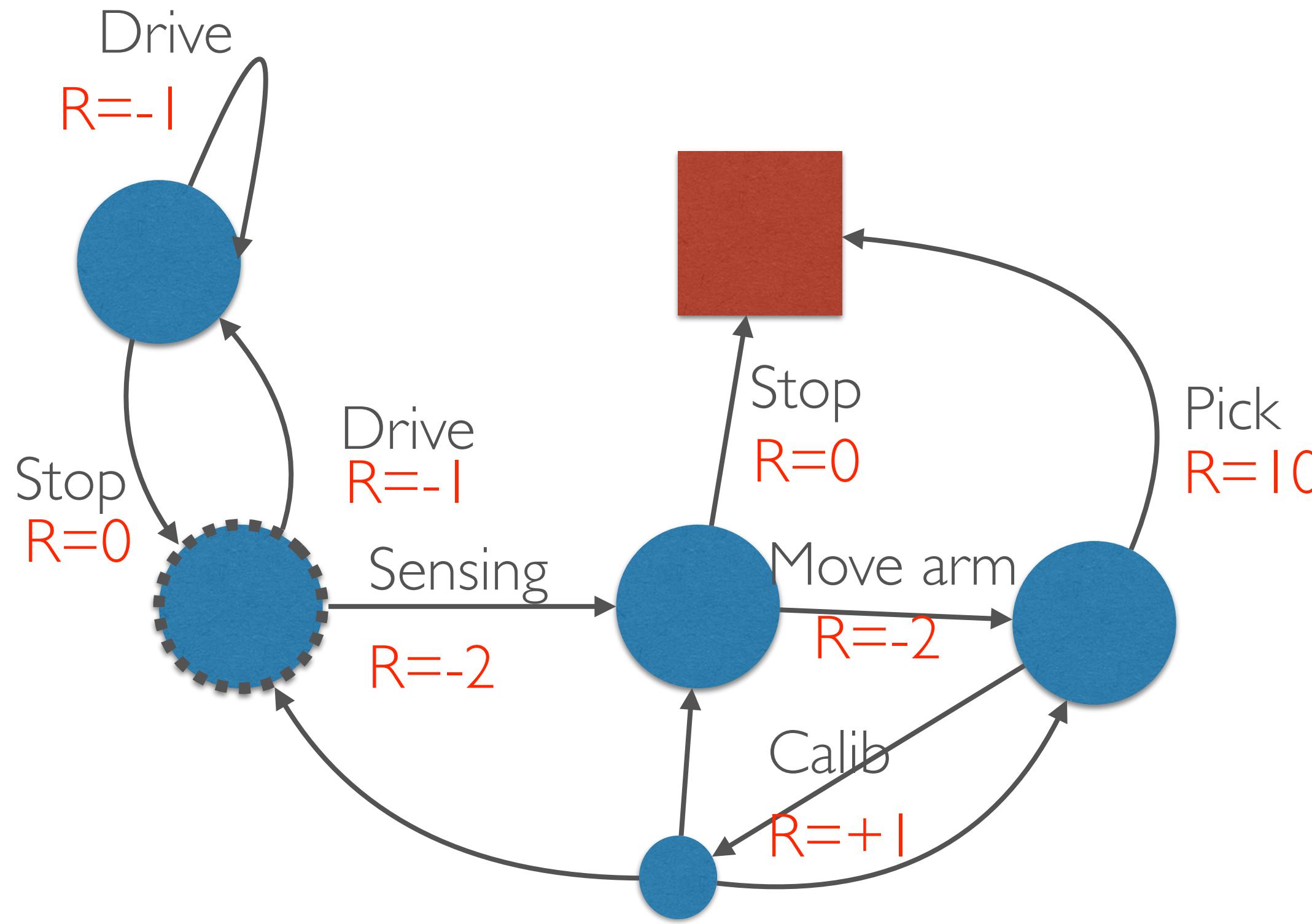
# MARKOV DECISION PROCESS

- A Markov Decision Process is a Markov Reward Process with decisions on actions

**Definition:** A **Markov Decision Process** is a tupel  $(S, A, P, R, \gamma)$

- $S$  is a finite set of states
- $A$  is a finite set of actions
- $P$  is a state transition probability matrix  $P_{ss'}^{\textcolor{red}{a}} = P(S_{t+1} = s' | S_t = s, A_t = \textcolor{red}{a})$
- $R$  is a reward function  $R_s^{\textcolor{red}{a}} = E(R_{t+1} | S_t = s, A_t = \textcolor{red}{a})$
- $\gamma$  is a discount factor  $\gamma \in [0,1]$





LUND  
UNIVERSITY

# POLICIES:WHAT DO TO WHEN

**Definition:** A **policy**  $\pi$  is a distribution over actions given states,

$$\pi(a | s) = P(A_t = a | S_t = s)$$

- Given that I am in state  $s$  what action  $a$  should I take?
- Policies define the behaviour of the agent
- Policies are stationary, i.e., they do not depend on the time

$$\sum_{a \in A} \pi(a | s) = 1$$

# POLICIES

- We can reduce a MDP to a MRP.
- Let  $\mathbf{M} = (\mathbf{S}, \mathbf{A}, \mathbf{P}, \mathbf{R}, \gamma)$  be an MDP with a policy  $\pi$
- If we marginalise out (average over) the actions

$$\hat{\mathbf{P}}_{s,s'} = \sum_{a \in A} \pi(a | s) P_{ss'}^a$$

$$\hat{\mathbf{R}}_s = \sum_{a \in A} \pi(a | s) R_s^a$$

- then  $(\mathbf{S}, \hat{\mathbf{P}}, \hat{\mathbf{R}}, \gamma)$  is a Markov Reward Process.



# VALUE FUNCTIONS

**Definition:** The state-value function  $v_\pi(s)$  of an MDP is the expected return starting from state  $s$  and following policy  $\pi$

$$v_\pi(s) = E_\pi \{ G_t | S_t = s \}$$



# VALUE FUNCTIONS

**Definition:** The state-value function  $v_\pi(s)$  of an MDP is the expected return starting from state  $s$  and following policy  $\pi$

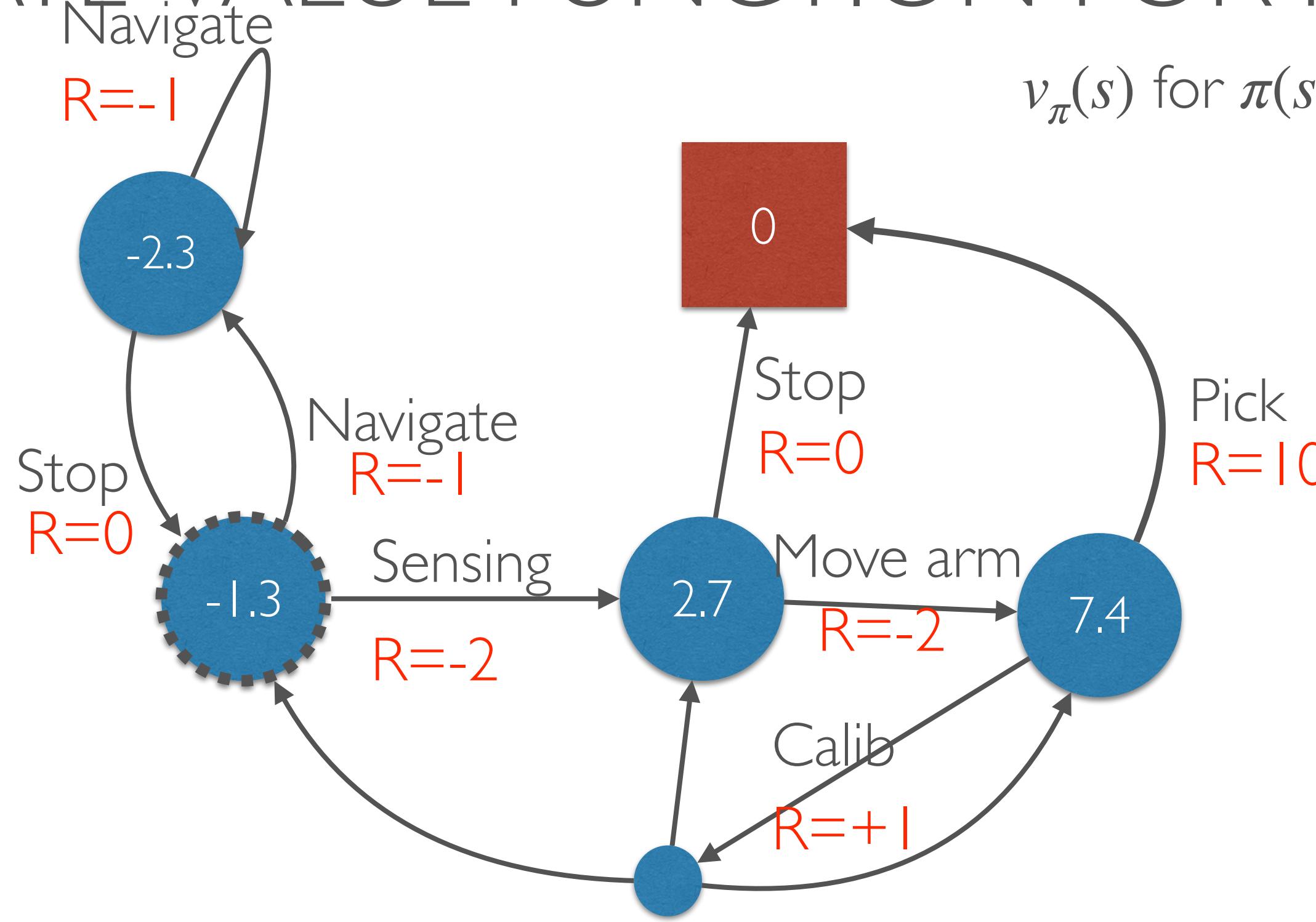
$$v_\pi(s) = E_\pi \{ G_t | S_t = s \}$$

**Definition:** The action-value function  $q_\pi(s, a)$  of an MDP is the expected return starting from state  $s$  by taking action  $a$  and following policy  $\pi$

$$q_\pi(s, a) = E_\pi \{ G_t | S_t = s, A_t = a \}$$



# STATE-VALUE FUNCTION FOR PICKING



# BELLMAN EXPECTATION EQUATION

- The state-value function and the action value function can again be recursively computed

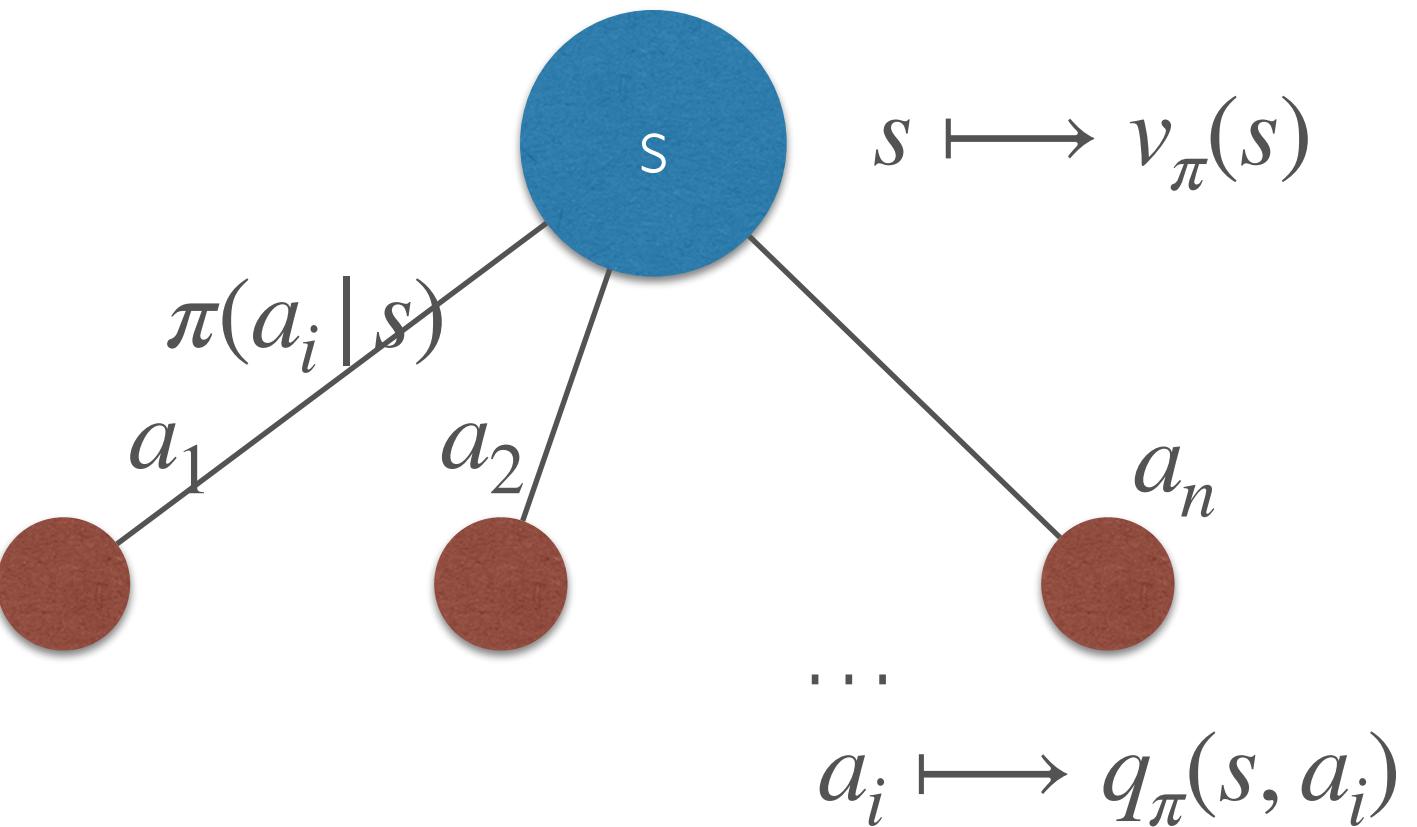
$$v_{\pi}(s) = E_{\pi} \{ R_{t+1} + \gamma v_{\pi}(S_{t+1}) \mid S_t = s \}$$

$$q_{\pi}(s, a) = E_{\pi} \{ R_{t+1} + \gamma q_{\pi}(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a \}$$



# BELLMAN EXPECTATION EQUATION

- From state  $s$ , we can take different actions  $a_i \in A$  chosen according to  $\pi(a | s)$



Averaging / marginalising over the action  $a$

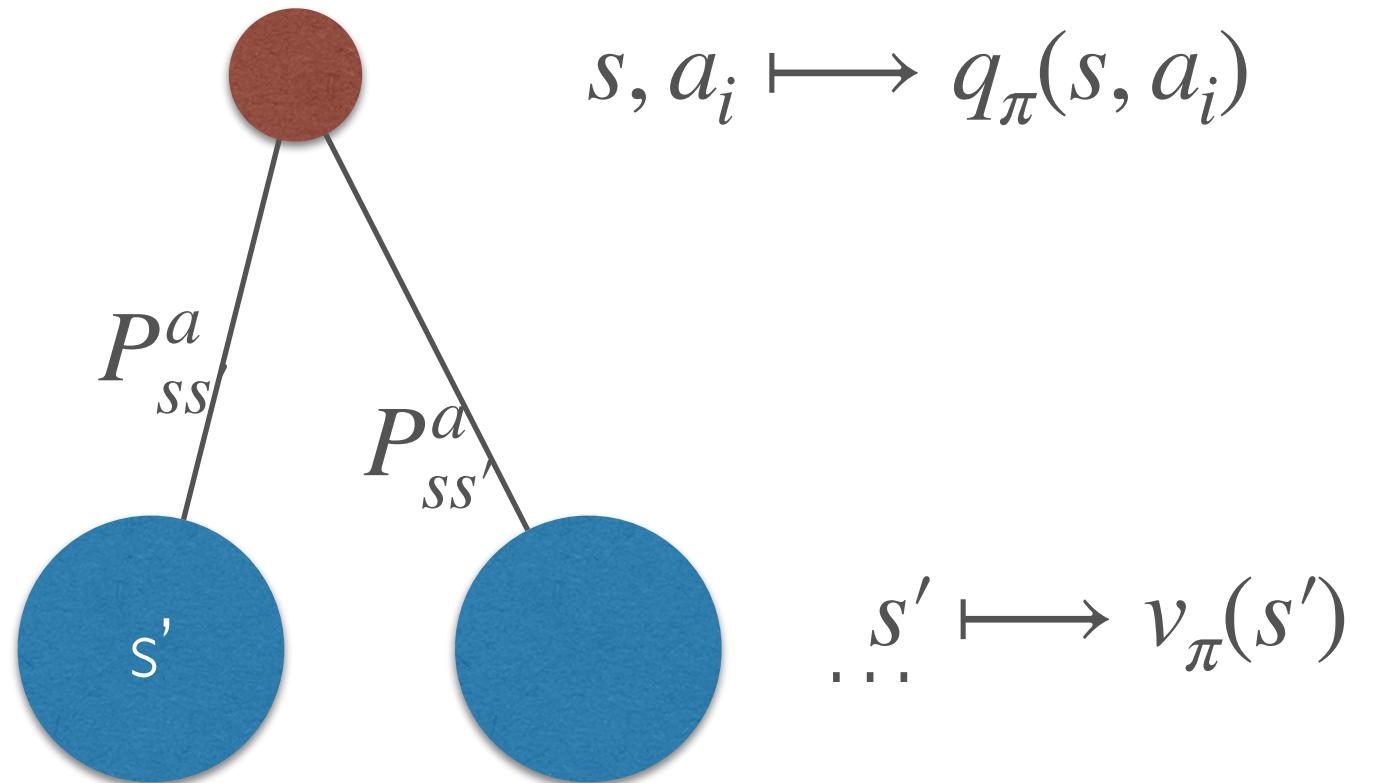
$$v_\pi(s) = \sum_{a \in A} \pi(a | s) q_\pi(s, a)$$



# BELLMAN EXPECTATION EQUATION

- Based on  $s$  and  $a$  we get a reward  $R_s^a$

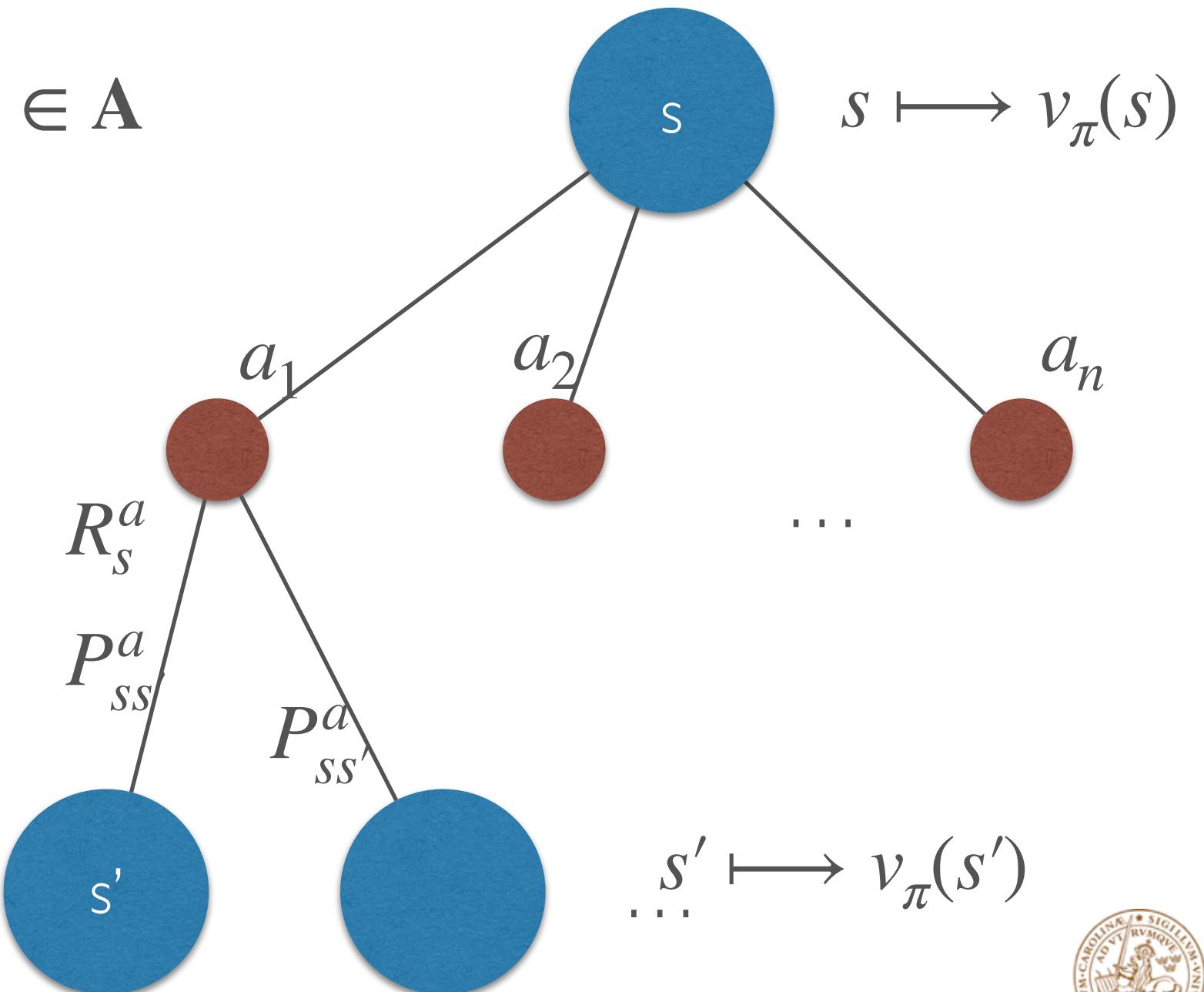
$$q_{\pi}(s, a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_{\pi}(s')$$



# BELLMAN EXPECTATION EQUATION

- From state  $s$ , we can take different actions  $a_i \in A$  chosen according to  $\pi(a | s)$
- Based on  $s$  and  $a$  we get a reward  $R_s^a$

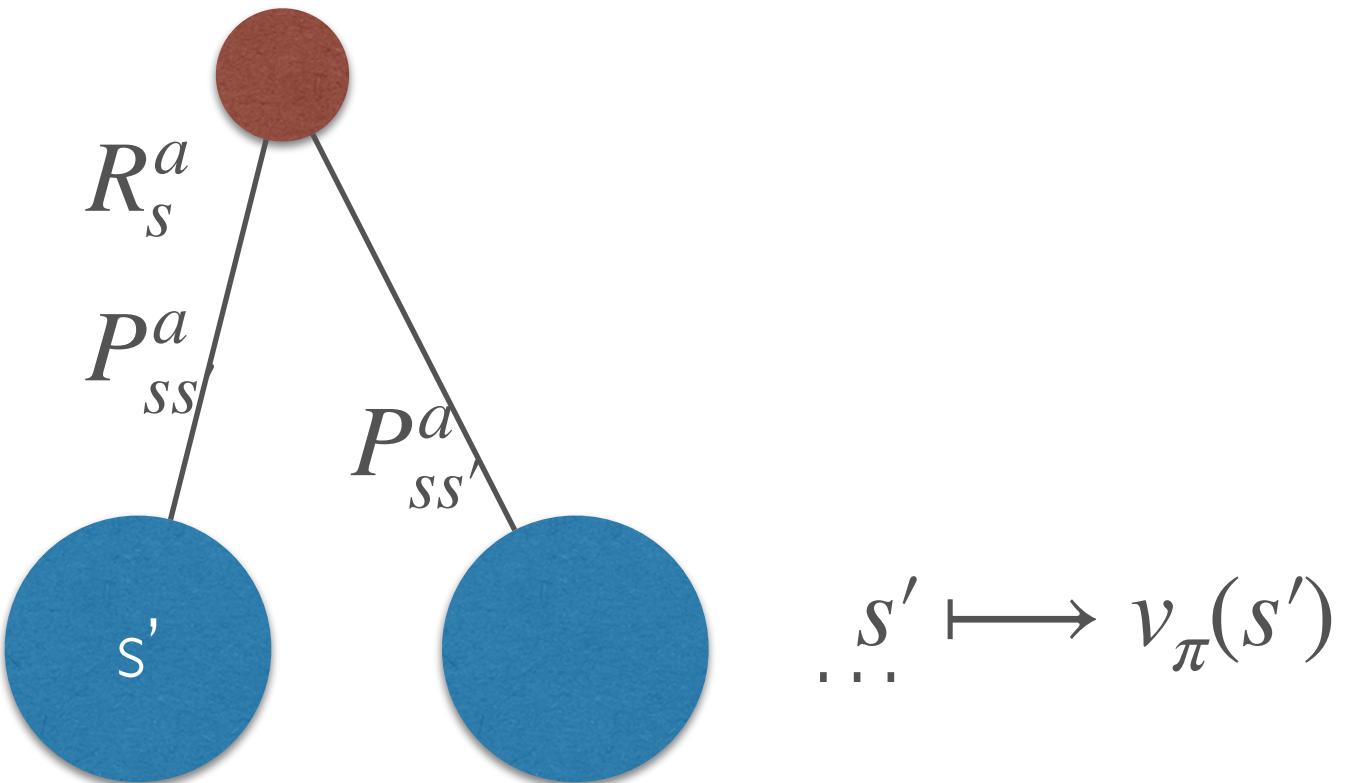
$$v_\pi(s) = \sum_{a \in A} \pi(a | s) \left( R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s') \right)$$



# BELLMAN EXPECTATION EQUATION

- From state  $s$ , we can take different actions  $a_i \in A$  chosen according to  $\pi(a | s)$
- Based on  $s$  and  $a$  we get a reward  $R_s^a$

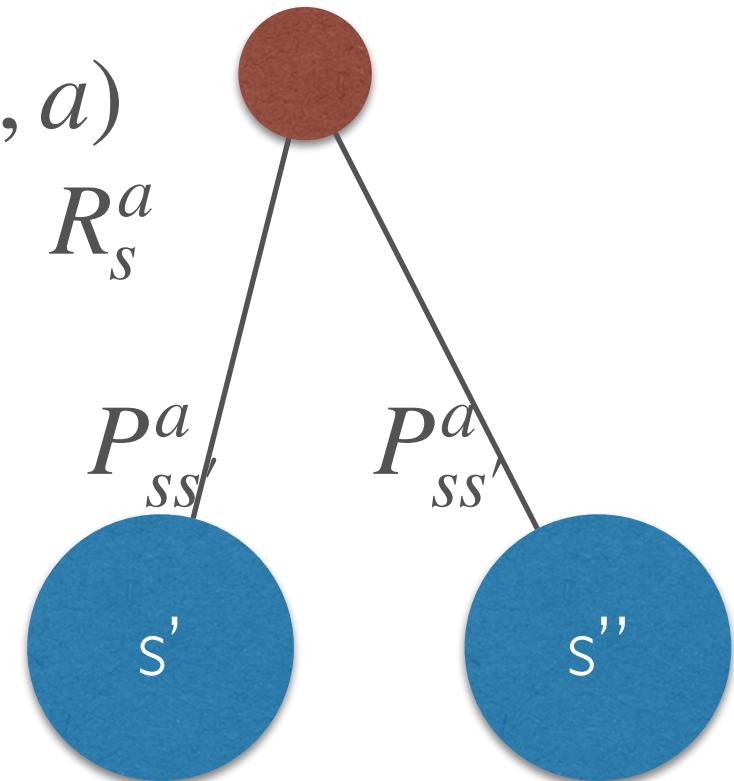
$$v_\pi(s) = \sum_{a \in A} \pi(a | s) \left( R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s') \right)$$



# BELLMAN EXPECTATION EQUATION

$$s, a \mapsto q_{\pi}(s, a)$$

$$q_{\pi}(s, a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \sum_{a' \in A} \pi(a' | s') q_{\pi}(s', a')$$



$$s', a_i \mapsto q_{\pi}(s', a_i)$$



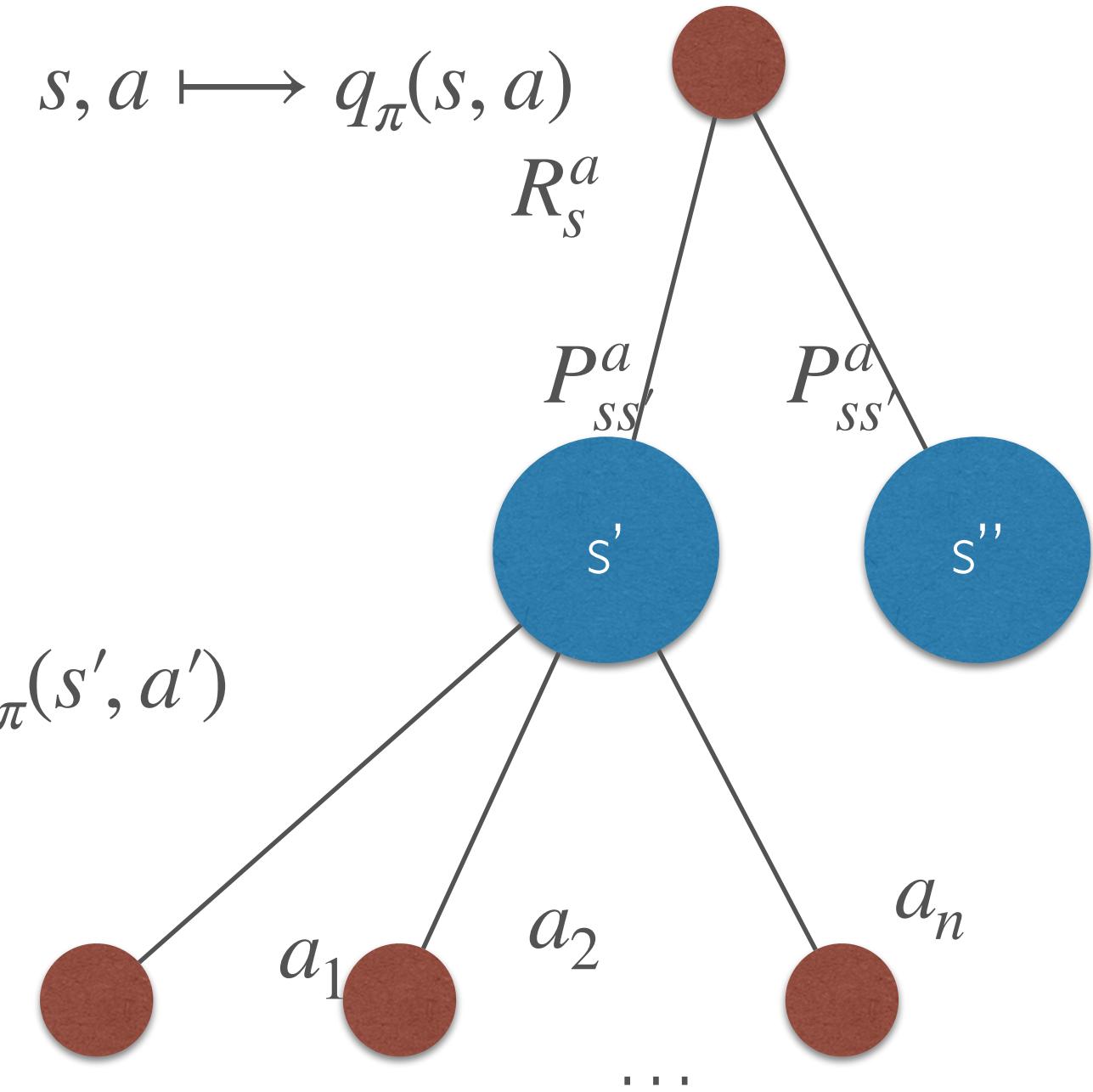
LUND  
UNIVERSITY

# BELLMAN EXPECTATION EQUATION

$$s, a \mapsto q_{\pi}(s, a)$$

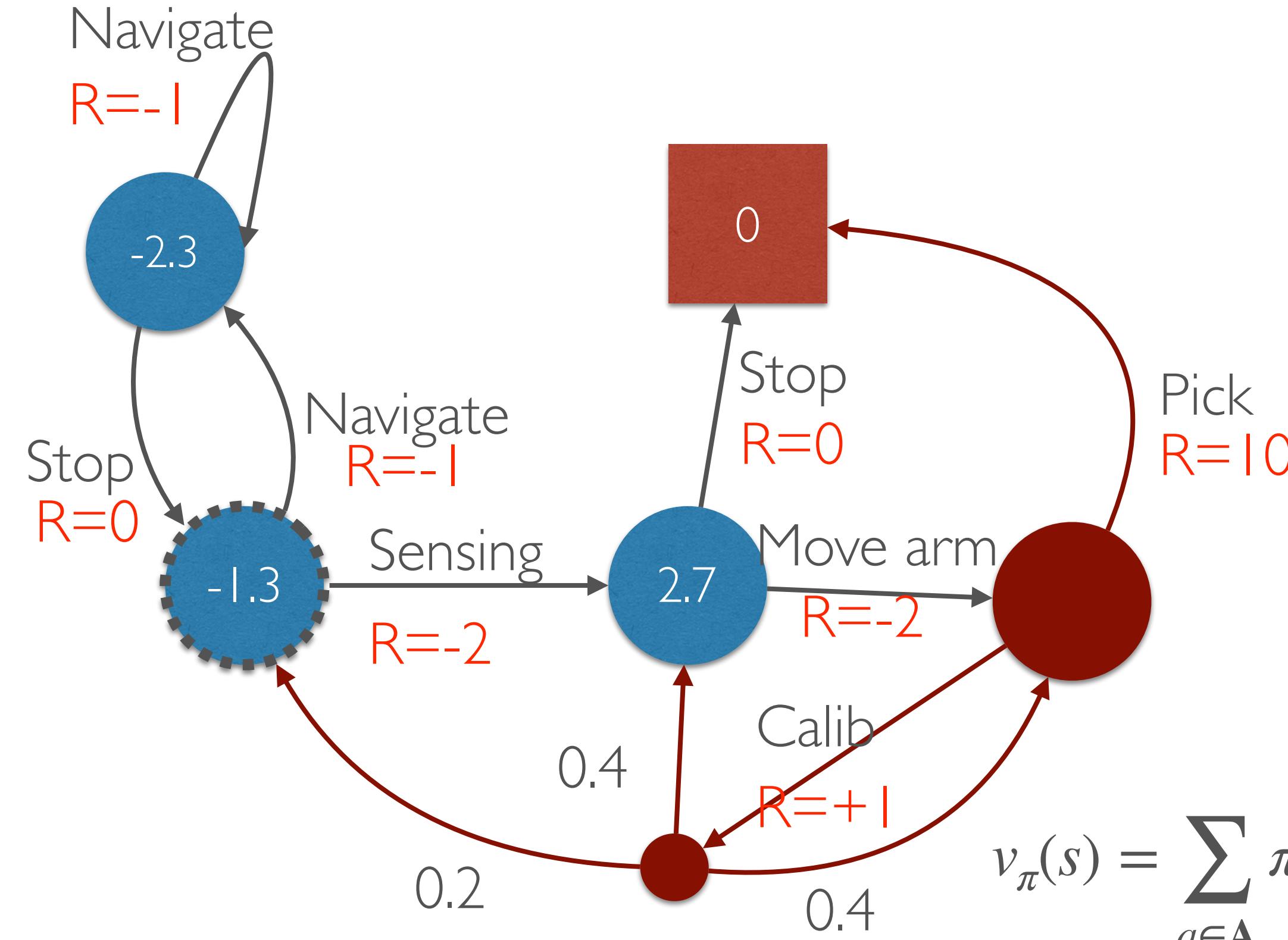
$$q_{\pi}(s, a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \sum_{a' \in A} \pi(a' | s') q_{\pi}(s', a')$$

$$s', a_i \mapsto q_{\pi}(s', a_i)$$



# BELLMAN EXPECTATION EQUATION

$v_\pi(s)$  for  $\pi(s, a) = 0.5, \gamma = 1$

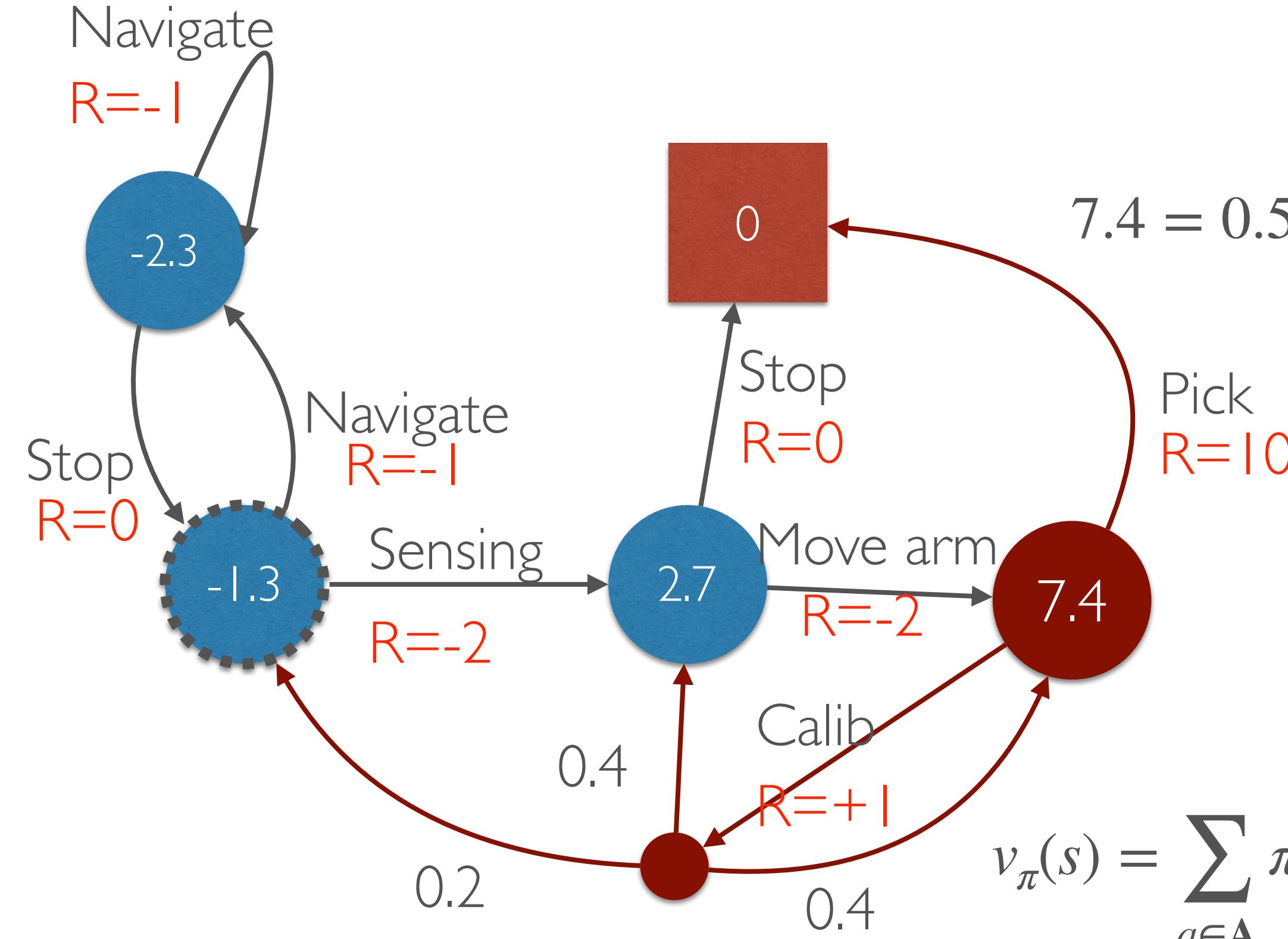


$$v_\pi(s) = \sum_{a \in A} \pi(a | s) \left( R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s') \right)$$

# BELLMAN EXPECTATION EQUATION

$$v_{\pi}(s) \text{ for } \pi(s, a) = 0.5, \gamma = 1$$

$$7.4 = 0.5 \cdot (1 + 0.2 \cdot (-1.3) + 0.4 \cdot 2.7 + 0.4 \cdot 7.4) + 0.5 \cdot 10$$



$$v_{\pi}(s) = \sum_{a \in A} \pi(a | s) \left( R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_{\pi}(s') \right)$$

# OPTIMAL VALUE FUNCTION

- So far, we have asked: Given a policy  $\pi$  what are the state values.
- however, we can also aim to optimise the policy  $\pi$  to maximise the state value function

$$v^*(s) = \max_{\pi} v_{\pi}(s)$$

**Definition:**  $v^*(s)$  is called the **optimal state-value** function

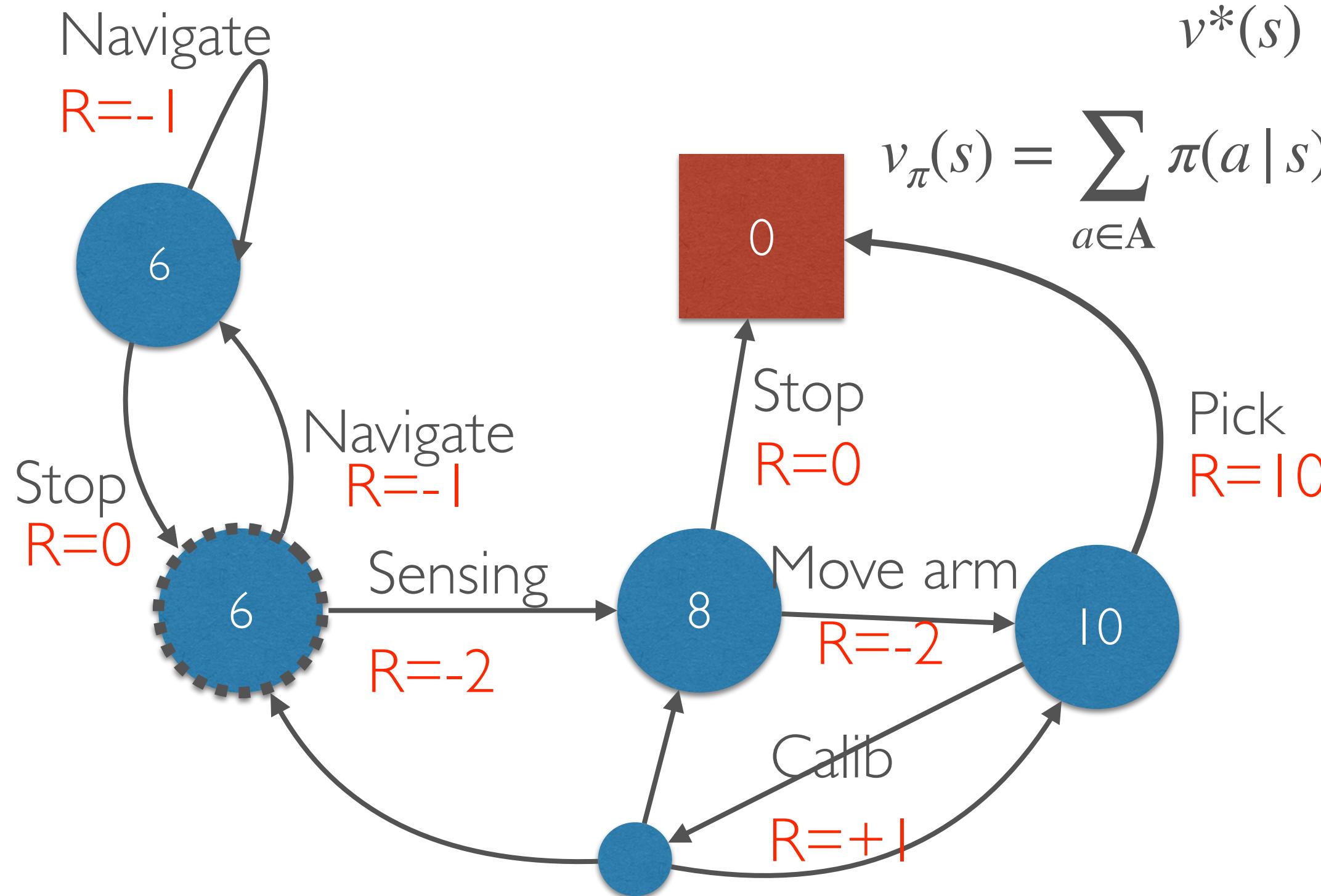
- Likewise, we can find the  $\pi$  that optimises the action-value function

$$q^*(s, a) = \max_{\pi} q_{\pi}(s, a)$$

**Definition:**  $q^*(s, a)$  is called the **optimal action-value function**



# EXAMPLE: OPTIMAL STATE-VALUE FUNCTION FOR PICKING

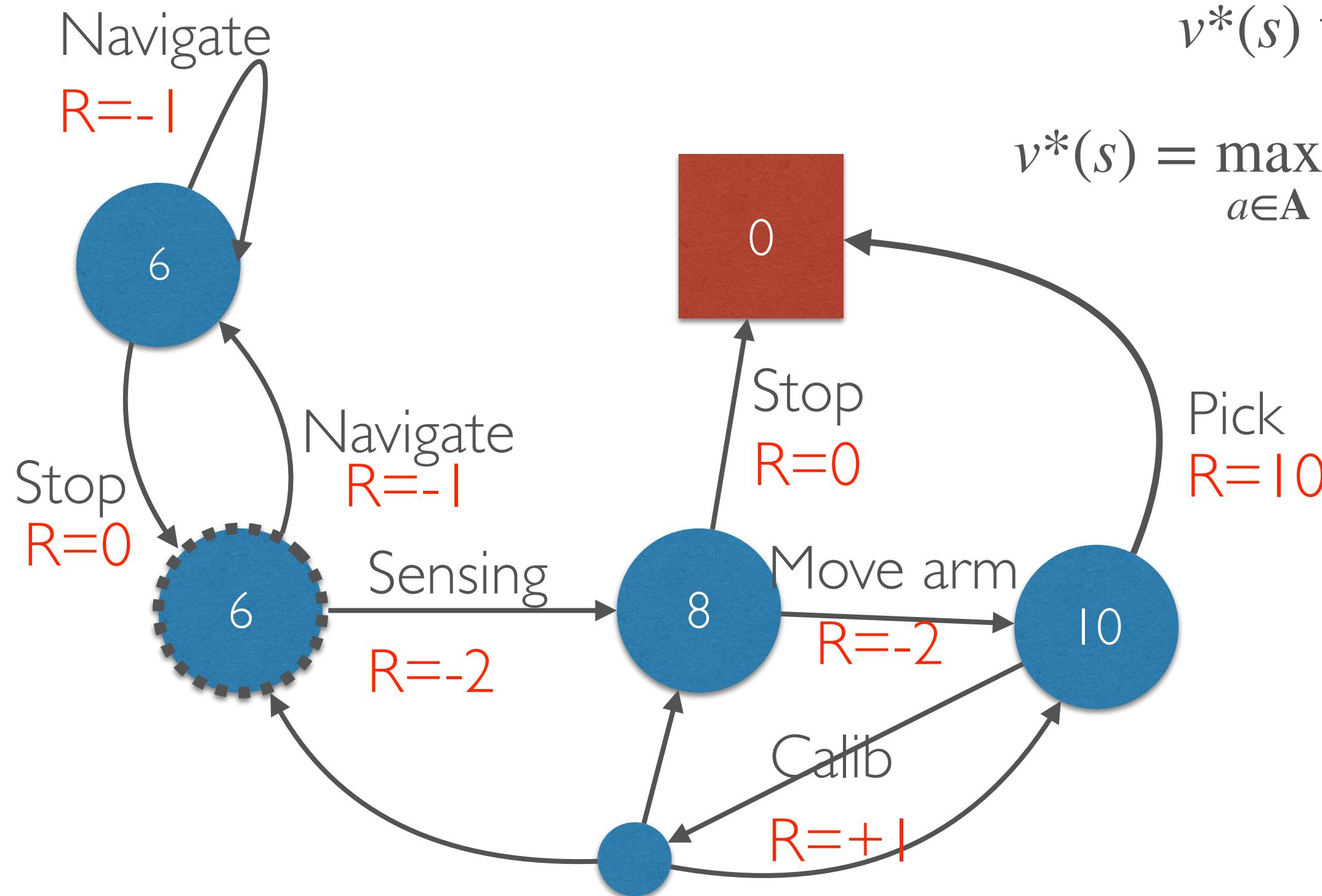


$v^*(s)$  for  $\gamma = 1$

$$v_\pi(s) = \sum_{a \in A} \pi(a | s) \left( R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s') \right)$$



# EXAMPLE: OPTIMAL STATE-VALUE FUNCTION FOR PICKING

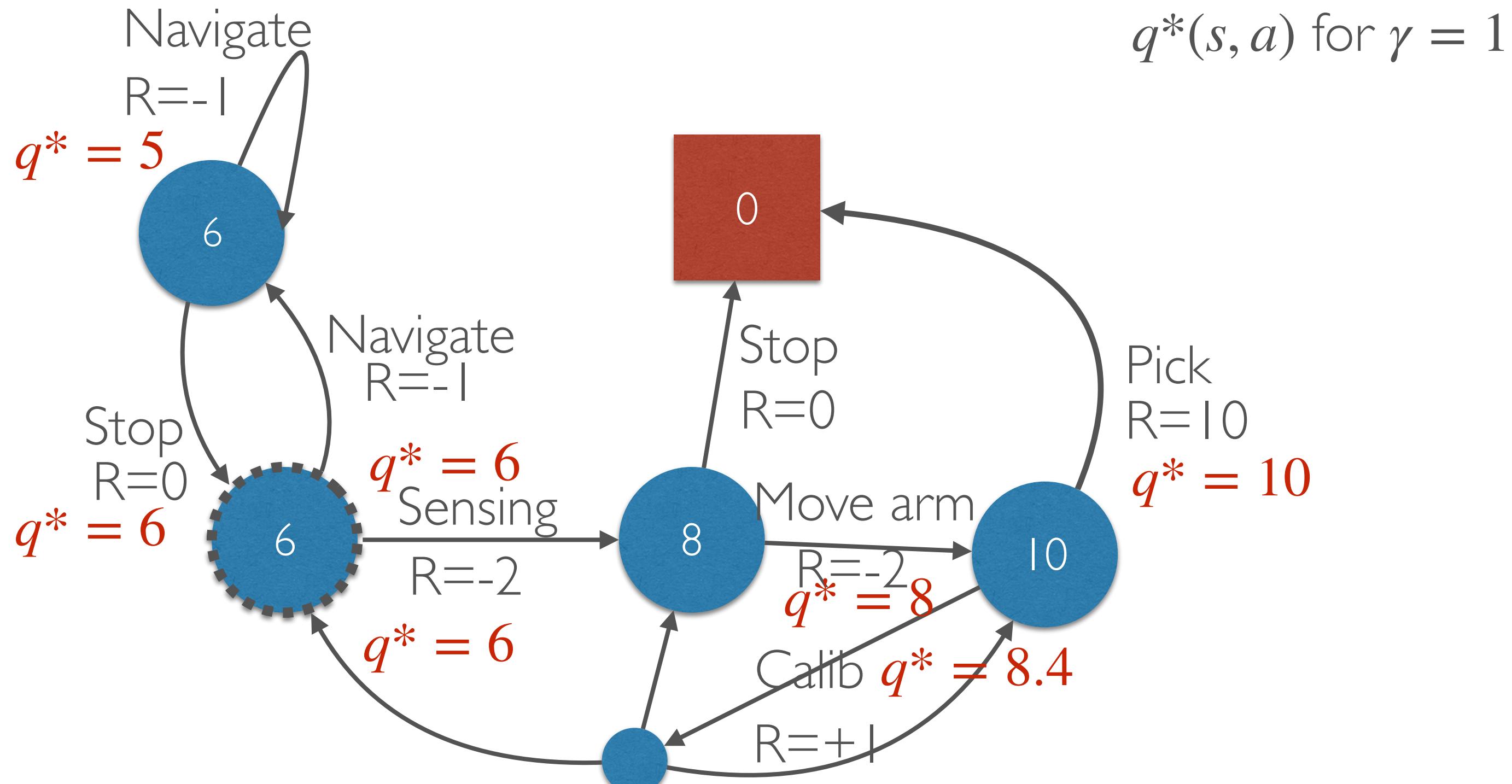


$v^*(s)$  for  $\gamma = 1$

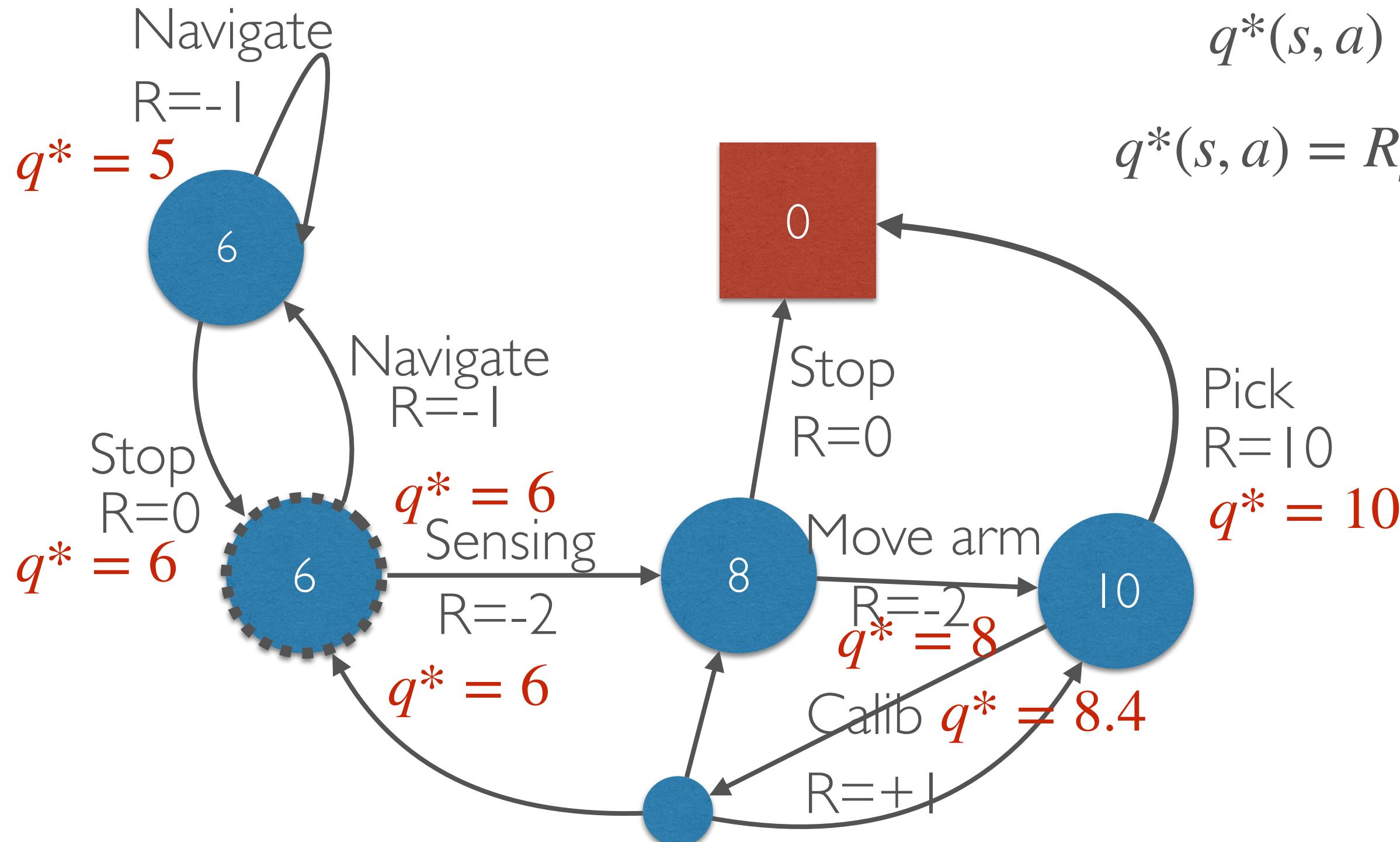
$$v^*(s) = \max_{a \in A} \left( R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v^*(s') \right)$$



# OPTIMAL ACTION-VALUE FUNCTION FOR PICKING



# OPTIMAL ACTION-VALUE FUNCTION FOR PICKING



# OPTIMAL POLICY

Define a partial ordering over policies:

$$\pi \geq \pi' \text{ if } v_\pi(s) \geq v_{\pi'}(s) \forall s$$

- **Theorem:** For any Markov Decision Process
- There exists an optimal policy  $\pi^*$  that is better than or equal to all other policies
$$\pi^* \geq \pi \forall \pi$$
- All optimal policies achieve the optimal value function  $v_{\pi^*}(s) = v^*(s)$
- All optimal policies achieve the optimal action-value function  $q_{\pi^*}(s, a) = q^*(s, a)$



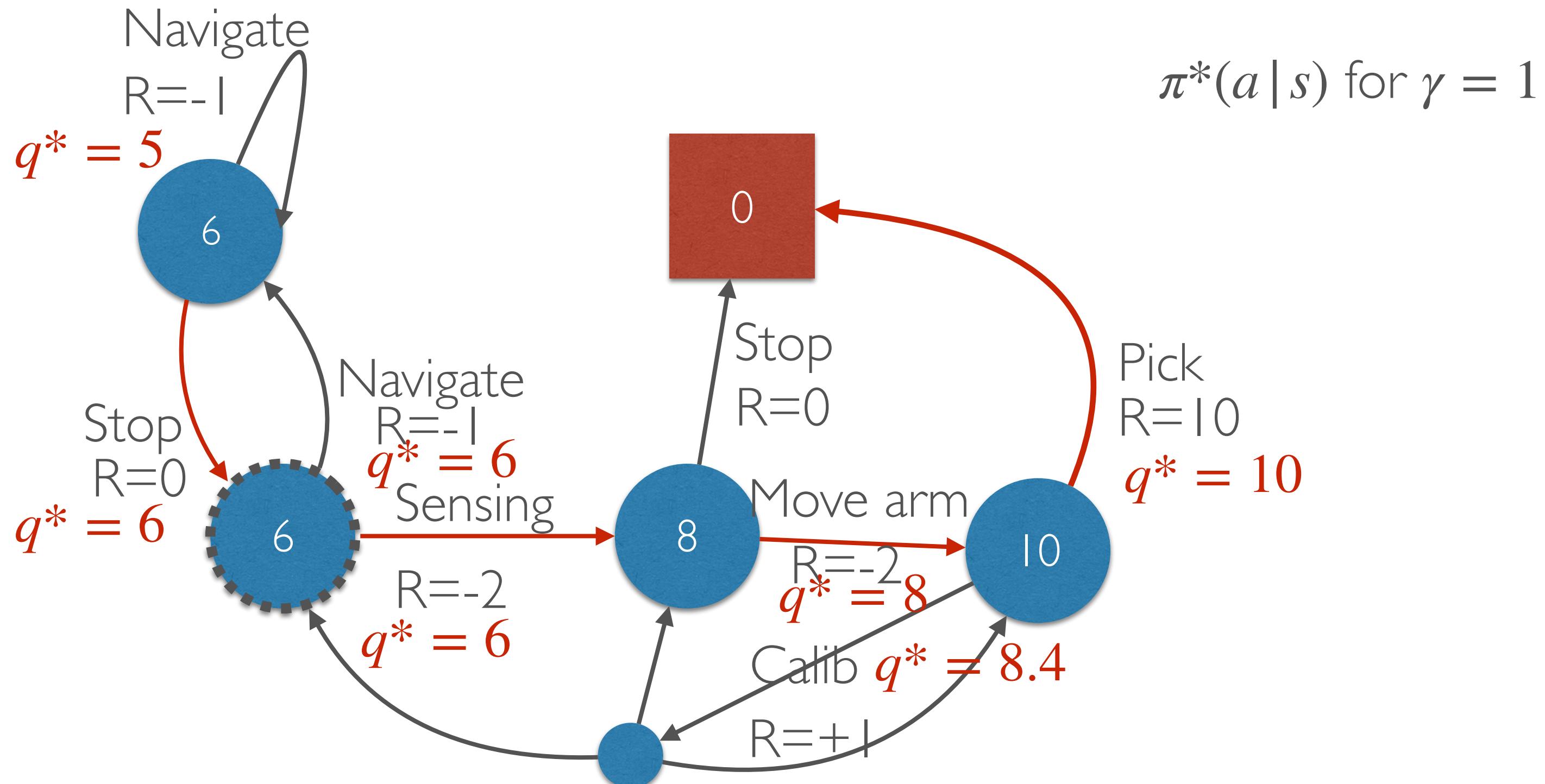
# FINDING AN OPTIMAL POLICY

- An optimal policy can be found by maximising over  $q^*(s, a)$

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

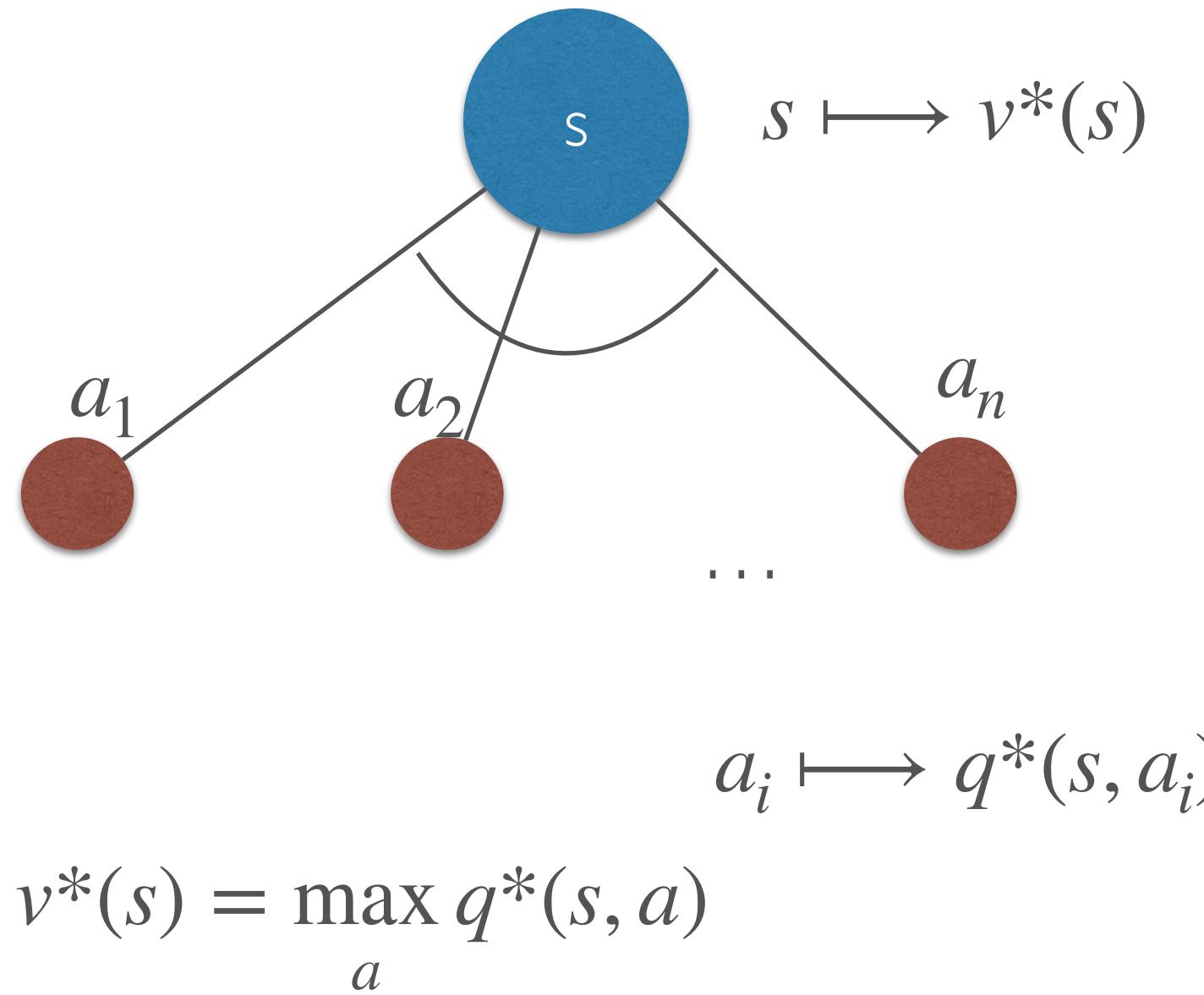
- There is always a deterministic optimal policy for any MDP
- If we know  $q^*(s, a)$ , we immediately have the optimal policy.





# BELLMAN OPTIMALITY EQUATION

- From state  $s$ , we can take action  $a_i \in A$  that has maximal value  $q(s, a)$

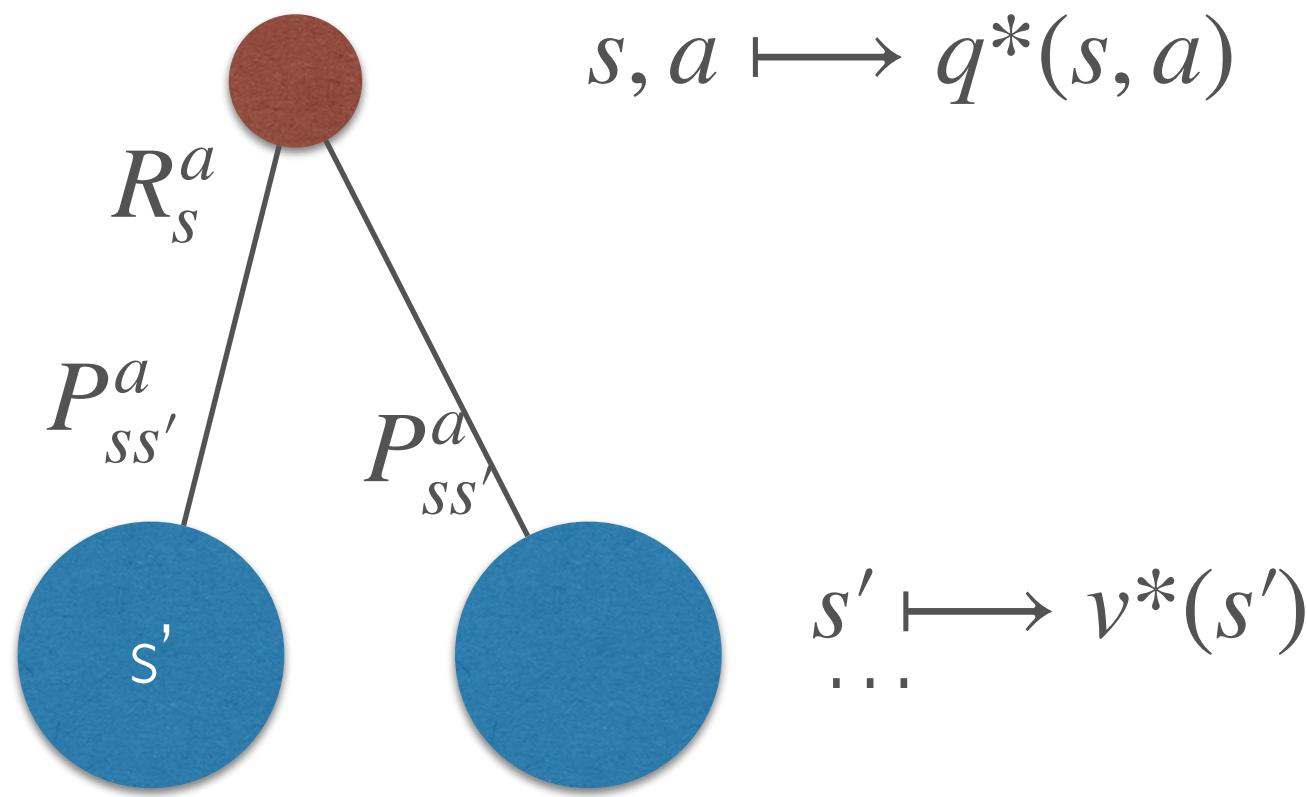


Maximizing over  $a$

# BELLMAN OPTIMALITY EQUATION

- Based on  $s$  and  $a$  we get a reward  $R_s^a$

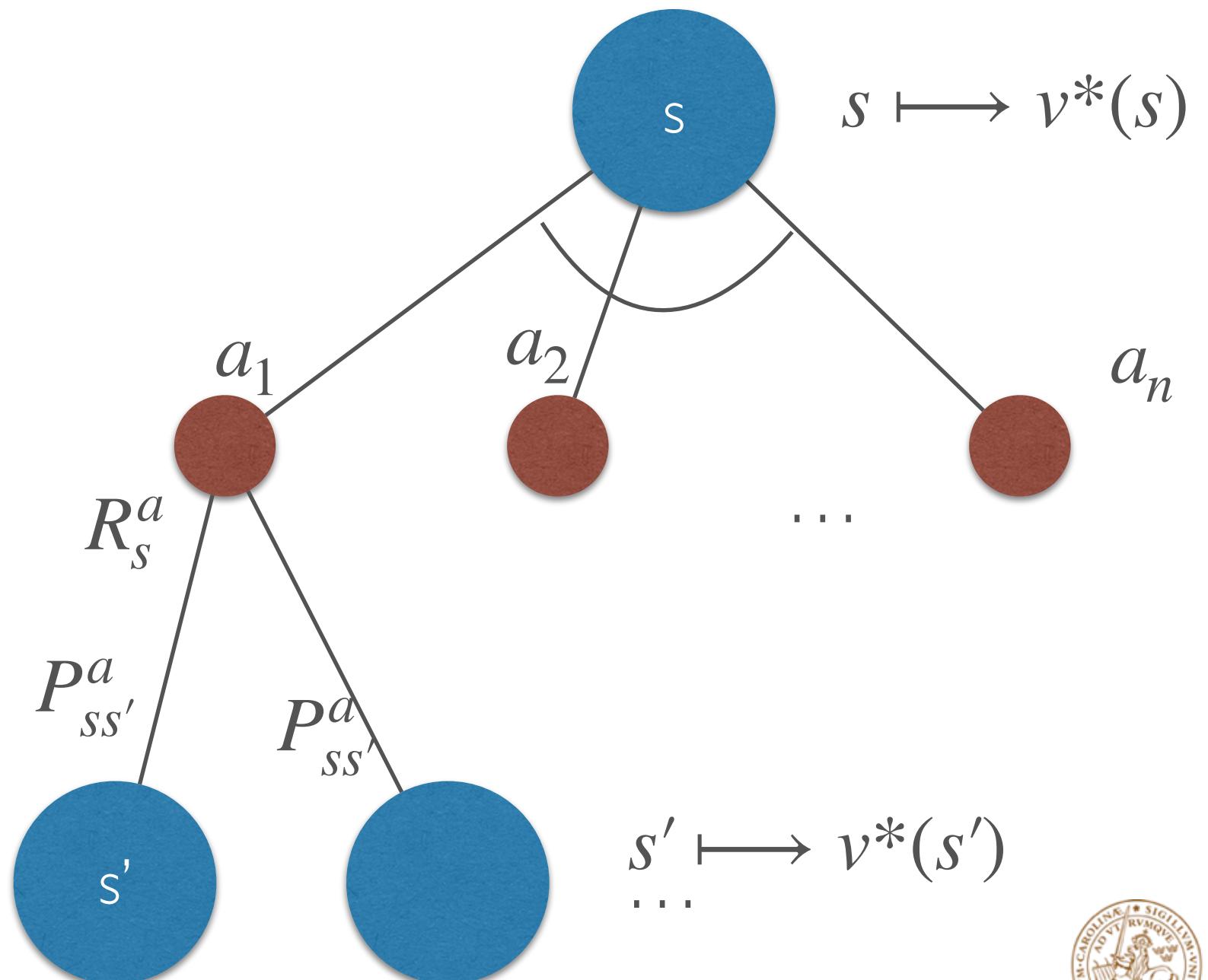
$$q^*(s, a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v^*(s')$$



# BELLMAN OPTIMALITY EQUATION

- From state  $s$ , we can action  $a_i \in A$  that has maximal reward  $q(s, a)$
- Based on  $s$  and  $a$  we get a reward  $R_s^a$

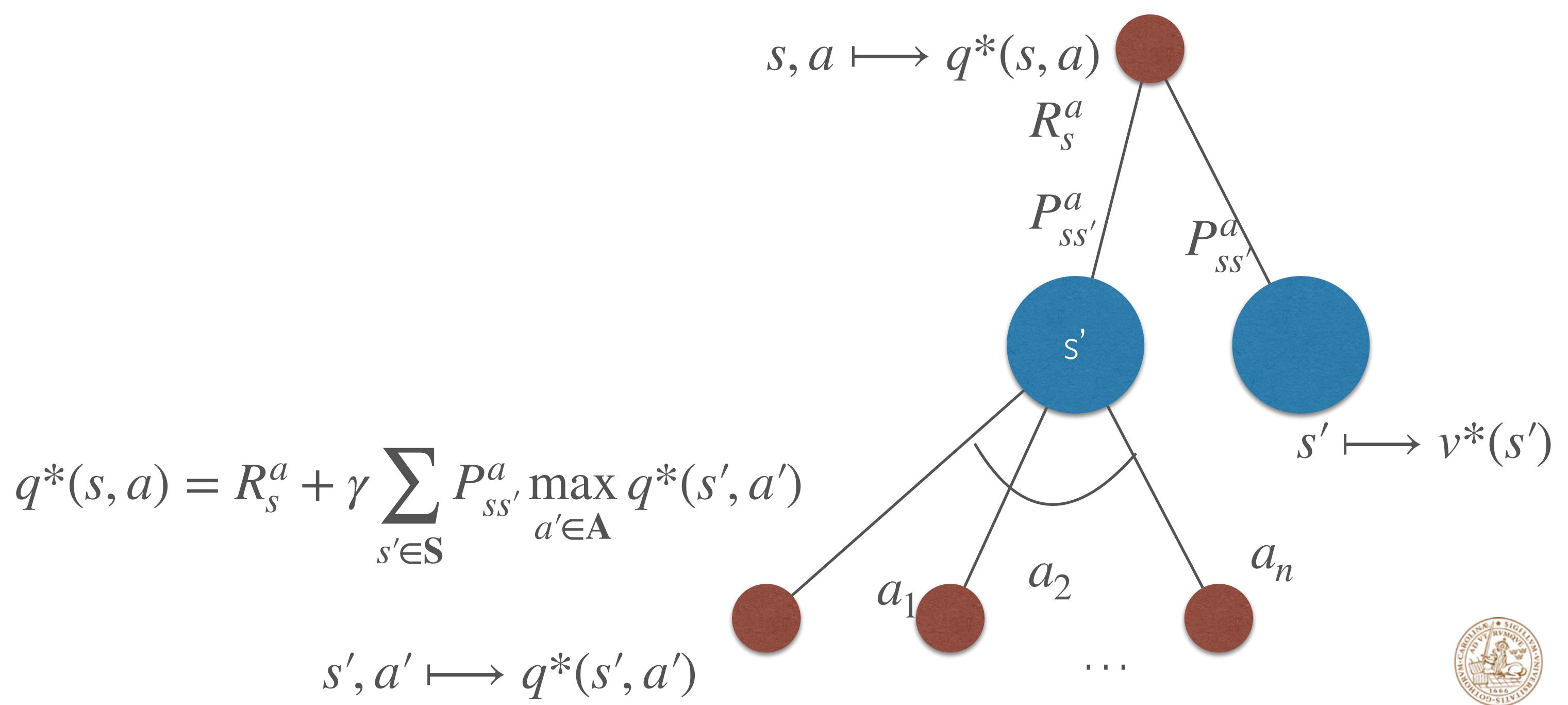
$$v_*(s) = \max_{a \in A} R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v^*(s')$$



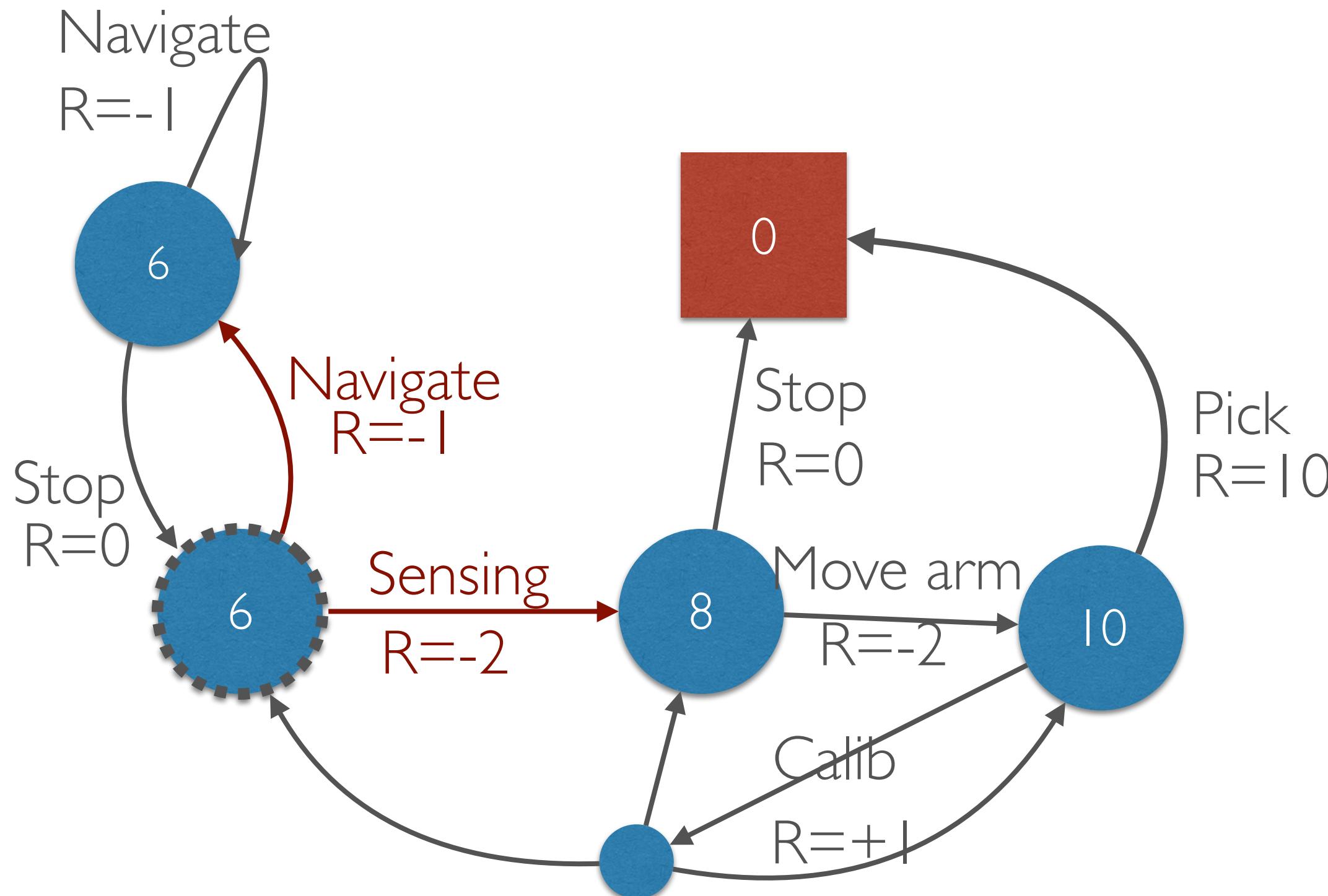
$$s' \longmapsto v^*(s')$$



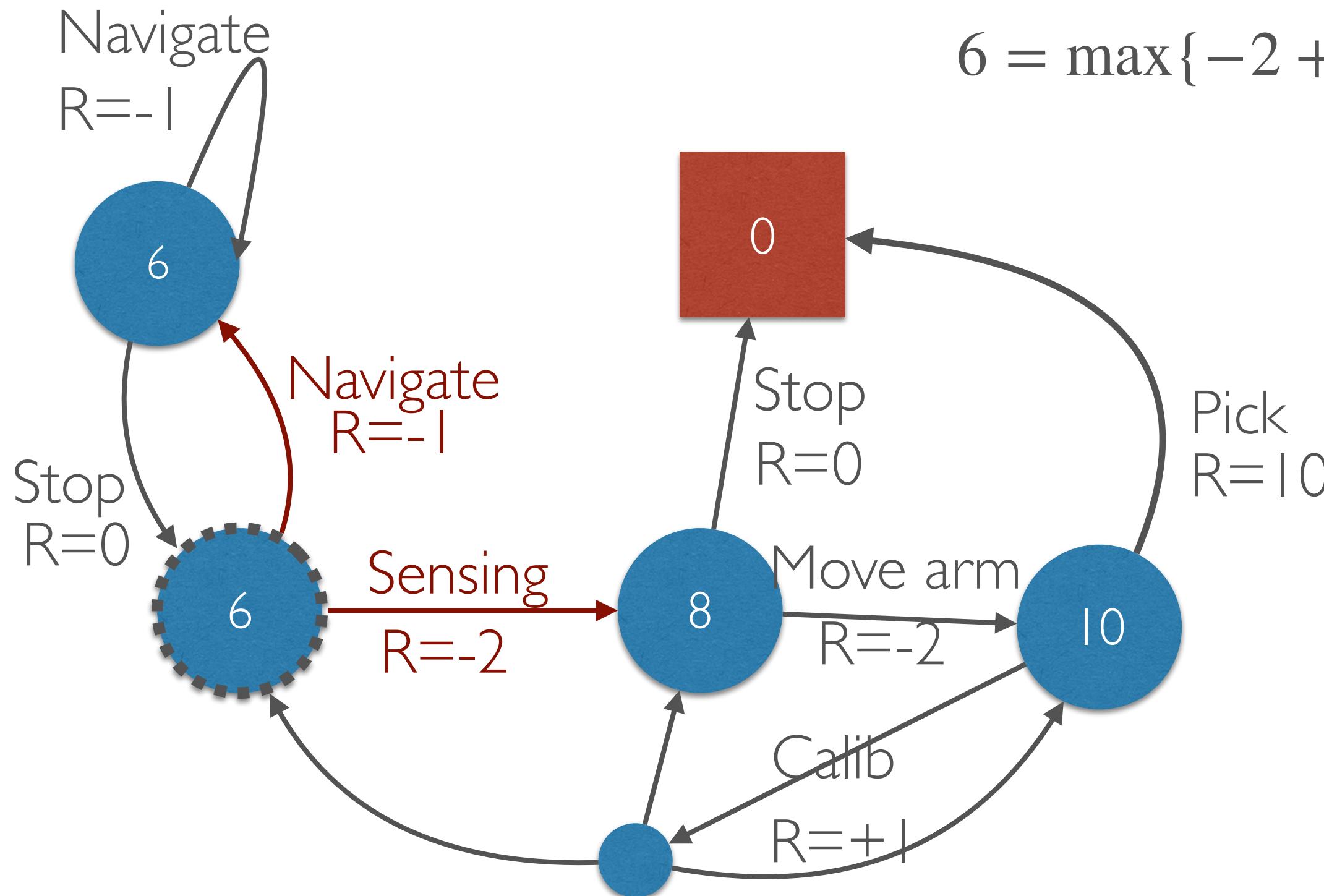
# BELLMAN OPTIMALITY EQUATION



# BELLMAN OPTIMALITY EQUATION



# BELLMAN OPTIMALITY EQUATION



$$6 = \max \{-2 + 8, -1 + 6\}$$



# SOLVING THE BELLMAN OPTIMALITY EQUATION

- Bellman Optimality Equation is non-linear
- No closed form solution (in general)
- Many iterative solution methods
  - Dynamic Programming: Value Iteration, Policy Iteration
  - Q-learning
  - Sarsa



# SUMMARY

- Markov Model, Markov Chain
- Markov Reward Process, Bellman Equation for the state value function
- Markov Decision Process, Bellman Eq. for state value and action value functions
- Identify the optimal policy, state value function and action value function
- What is the model knowledge in a MDP?

