

MARKOV DECISION PROCESSES

Volker Krüger

Slides partially borrowed from a presentation by David Silver



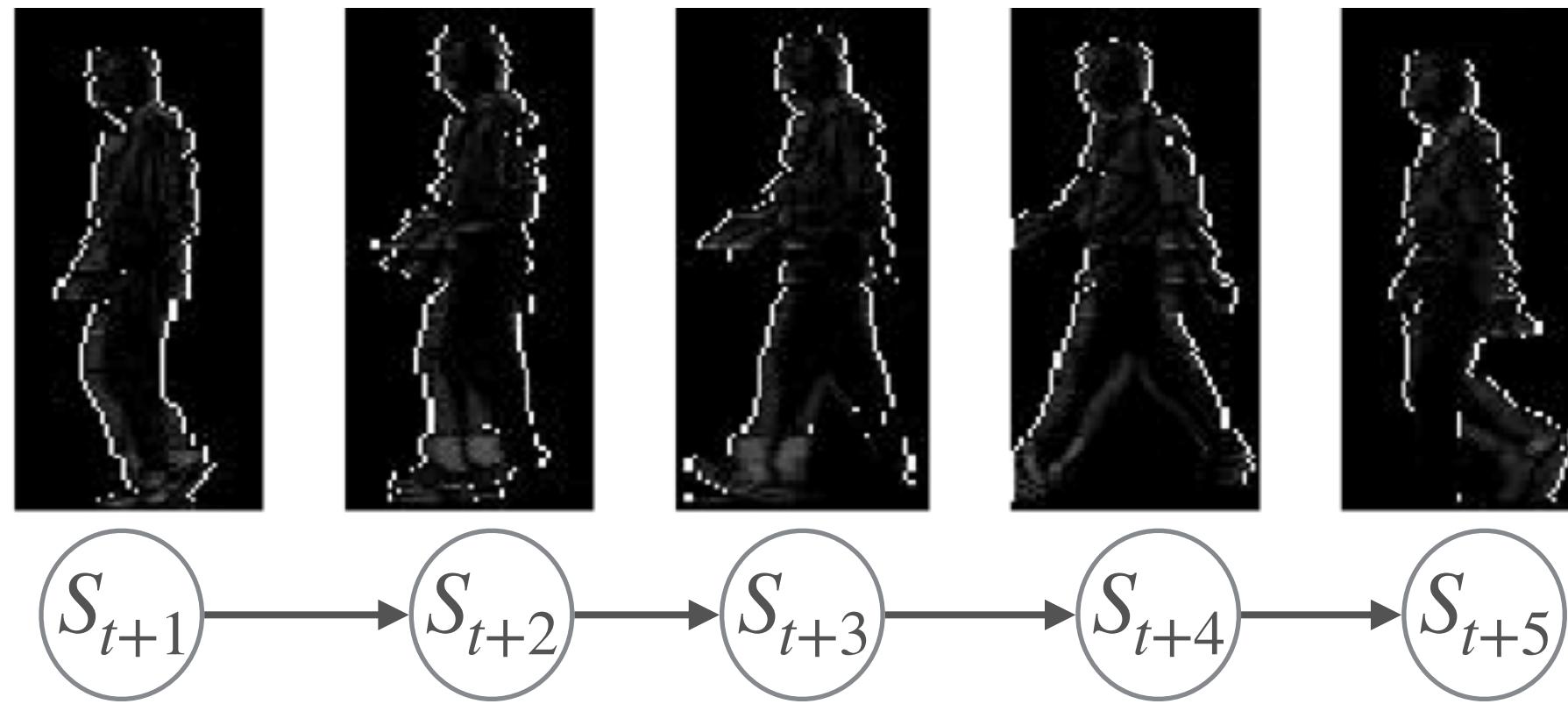
LUND
UNIVERSITY

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



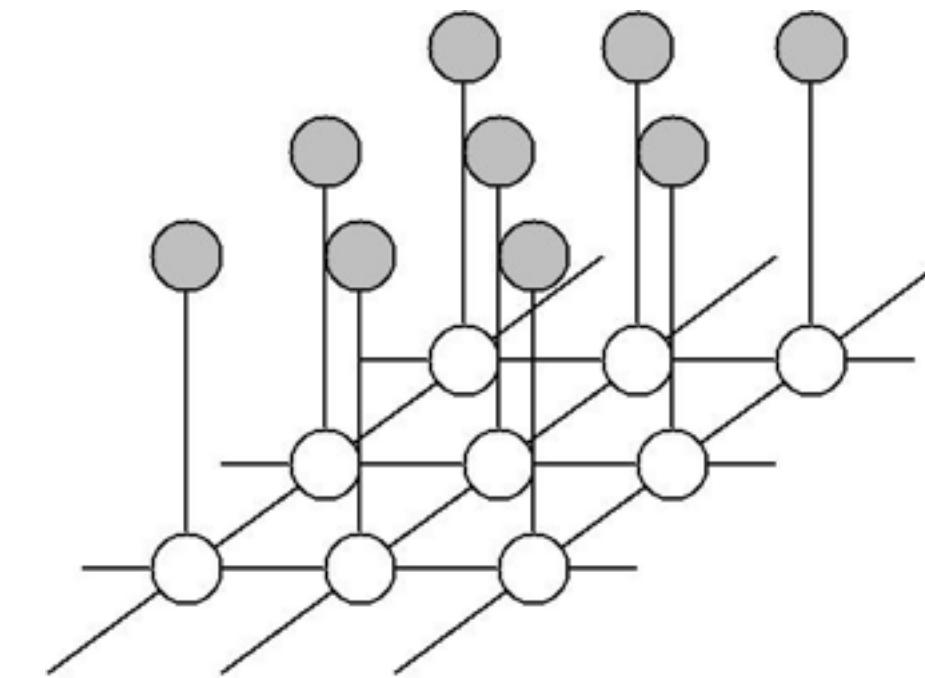
MARKOV PROCESSES



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

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

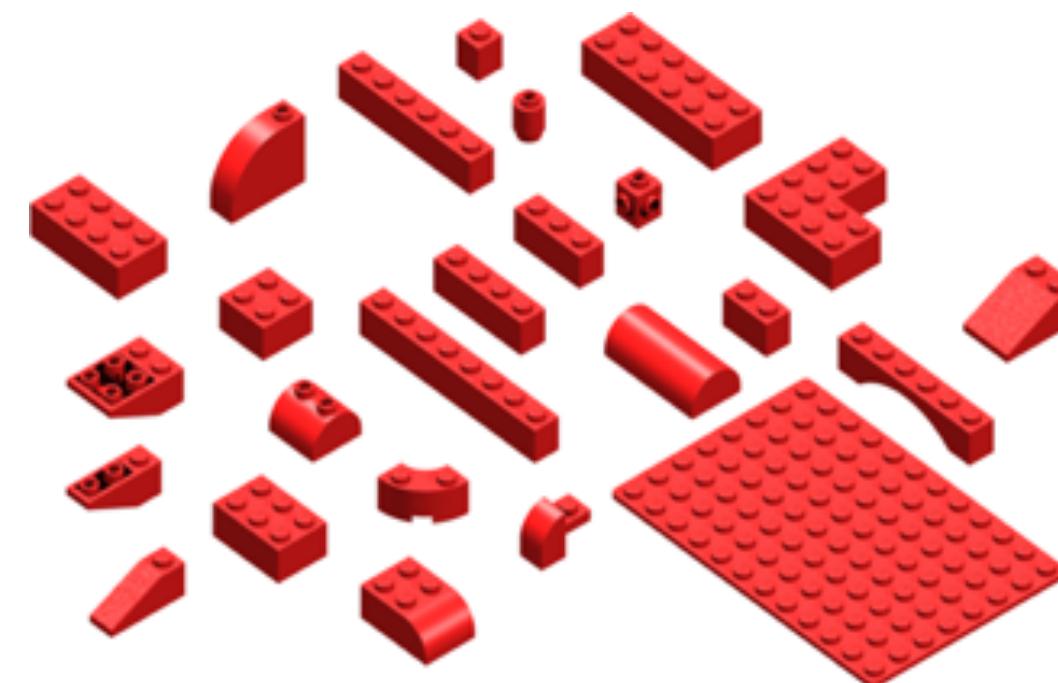
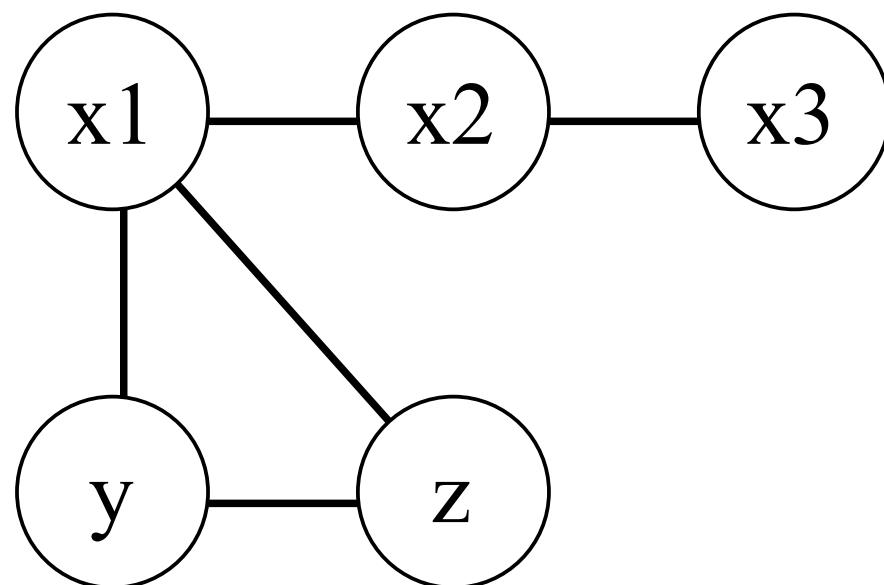
MARKOV RANDOM FIELDS



LUND
UNIVERSITY

GRAPHICAL MODELS TINKER TOYS TO BUILD COMPLEX PROBABILITY DISTRIBUTIONS

- Circles represent random variables.
- Lines represent statistical dependencies.
- There is a corresponding equation that gives $P(x_1, x_2, x_3, y, z)$ but often it's easier to understand things from the picture.
- These tinker toys for probabilities let you build up, from simple, easy-to-understand pieces, complicated probability distributions involving many variables.



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.



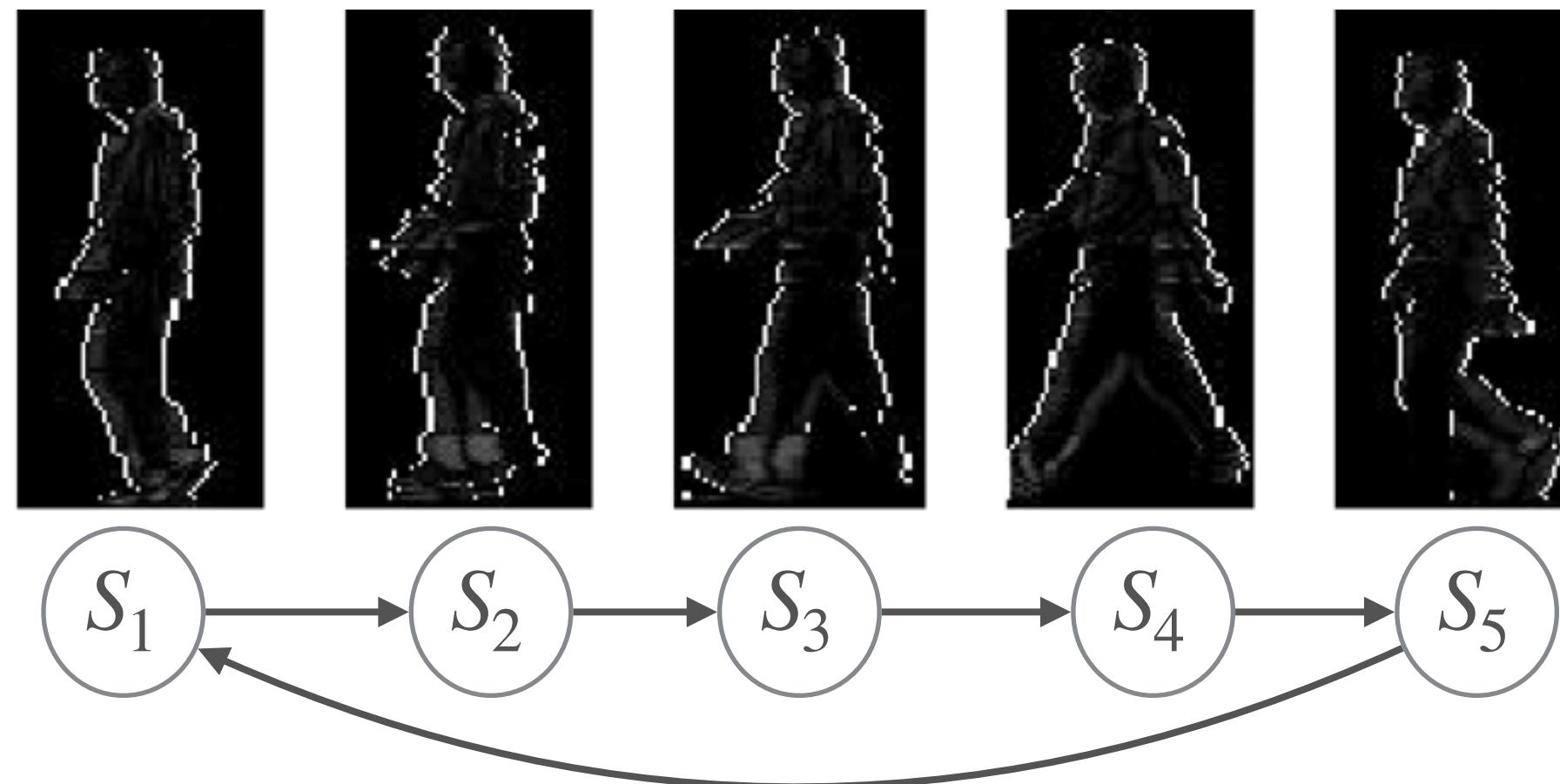
MARKOV PROCESS

- 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 (S, P)
 - S is a (finite) set of states
 - P is a state transition probability matrix, with

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

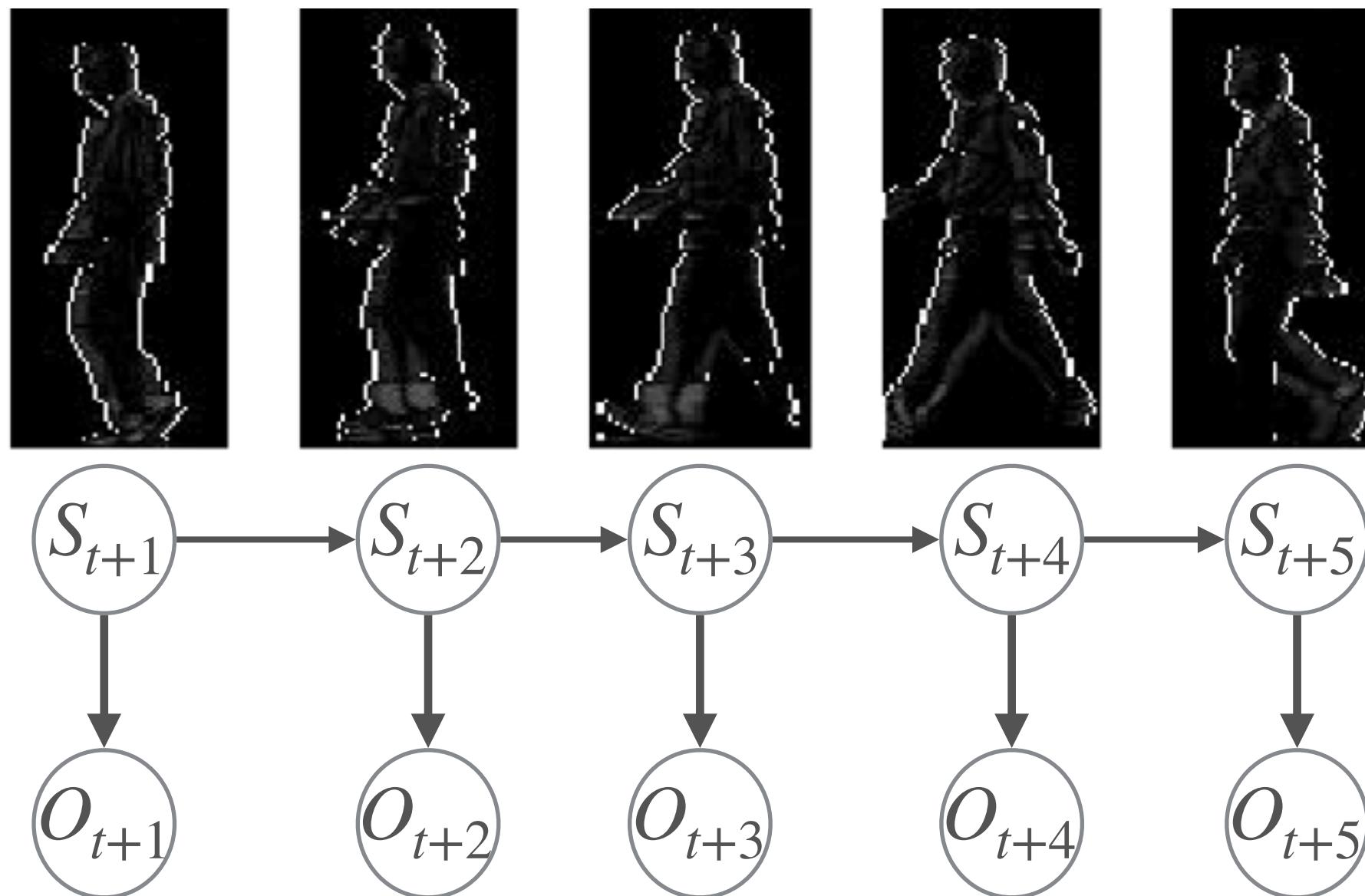


EXAMPLE GAIT MARKOV CHAIN



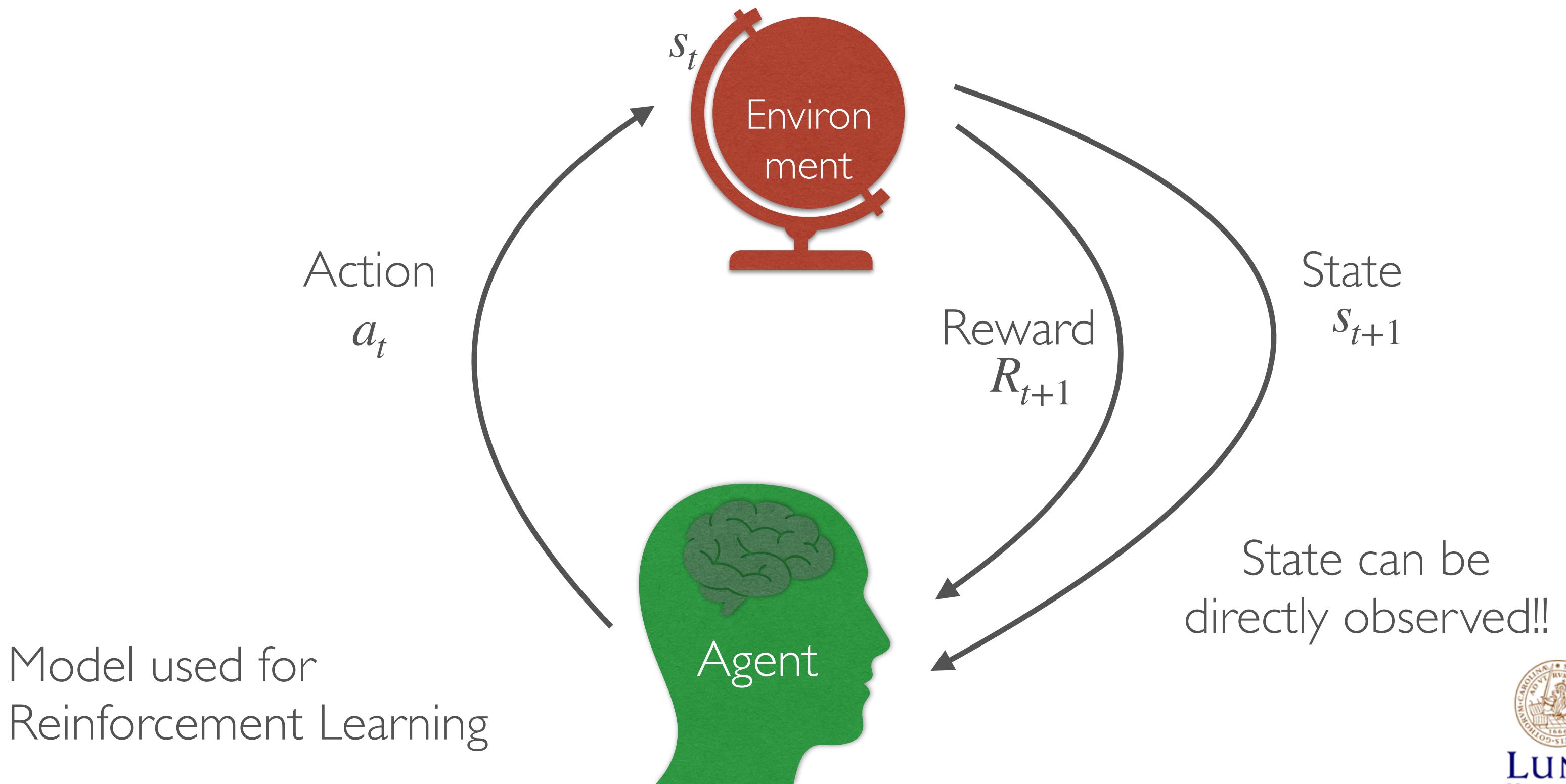
LUND
UNIVERSITY

HIDDEN MARKOV MODELS

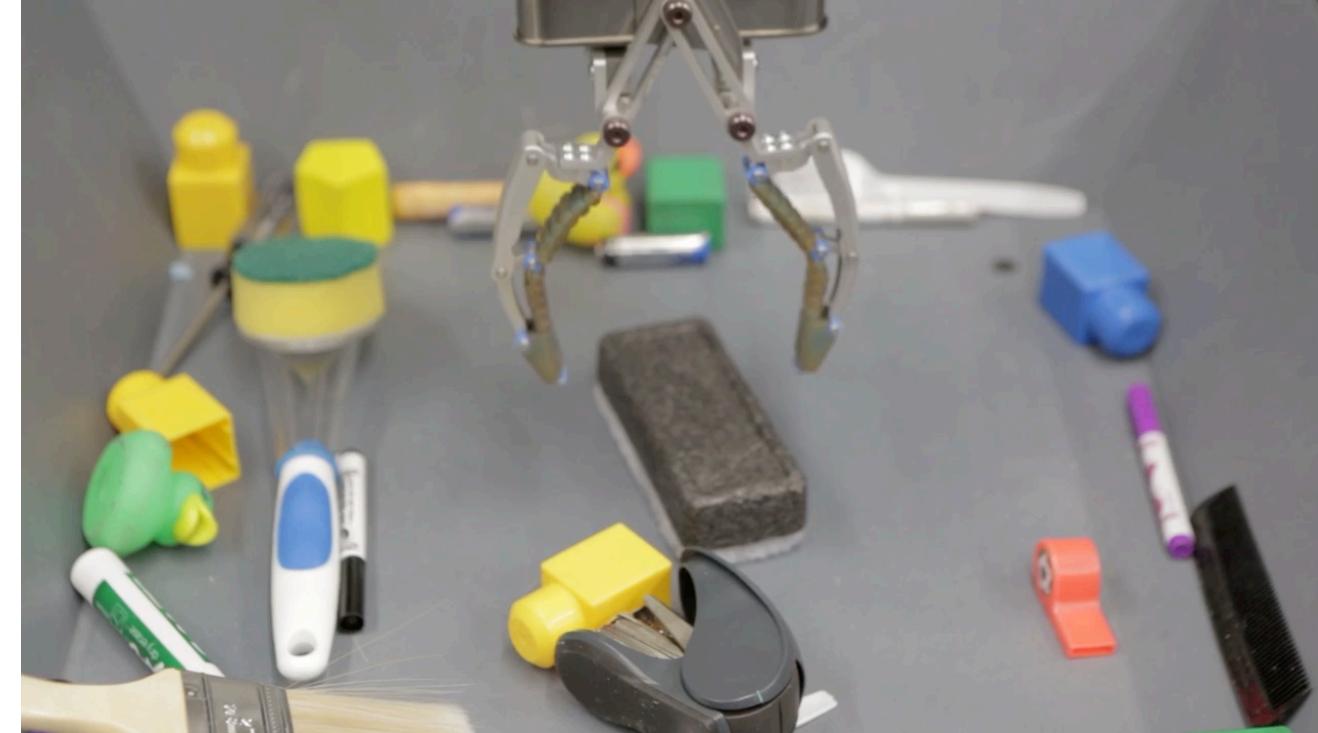


- Extension: Hidden Markov Model
 - From last lecture: Likelihood of an observation O_t , given that we are in state S_t

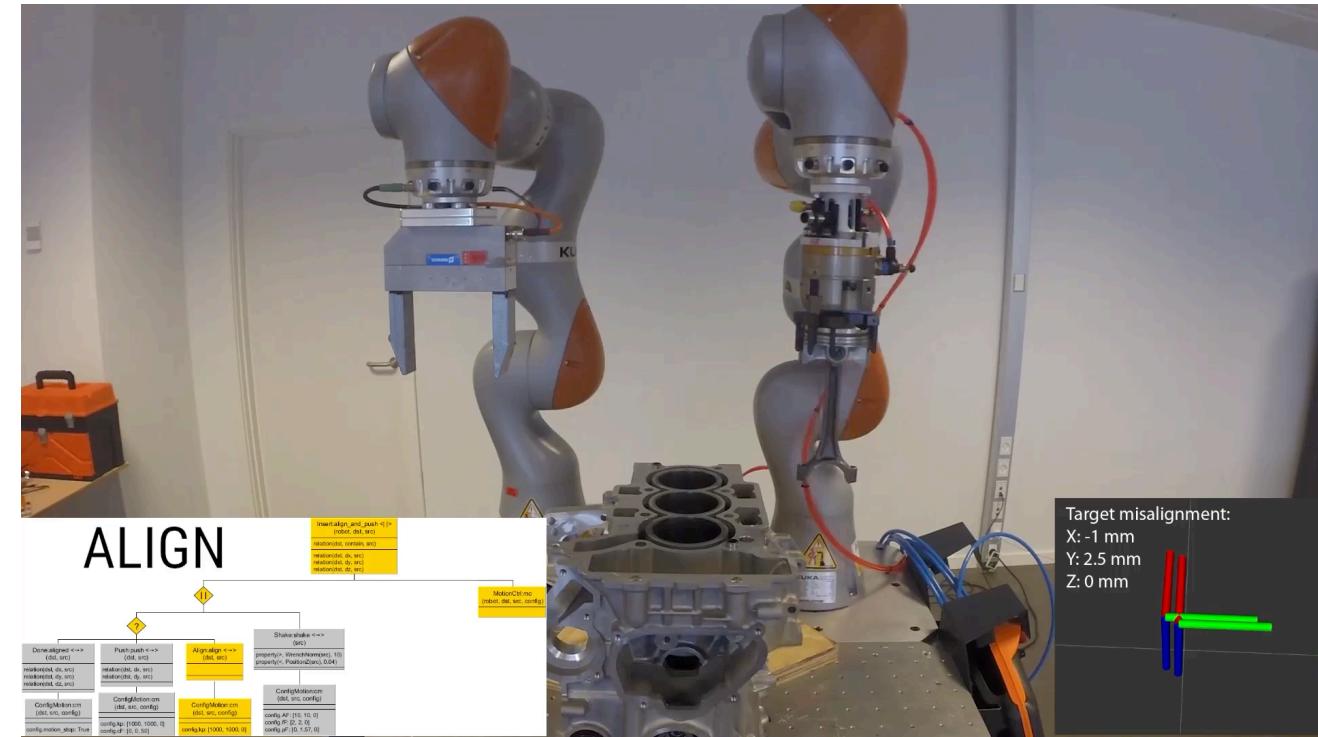
MARKOV DECISION PROCESS



REINFORCEMENT LEARNING



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



LUND
UNIVERSITY

EXAMPLE: PICKING MARKOV CHAIN

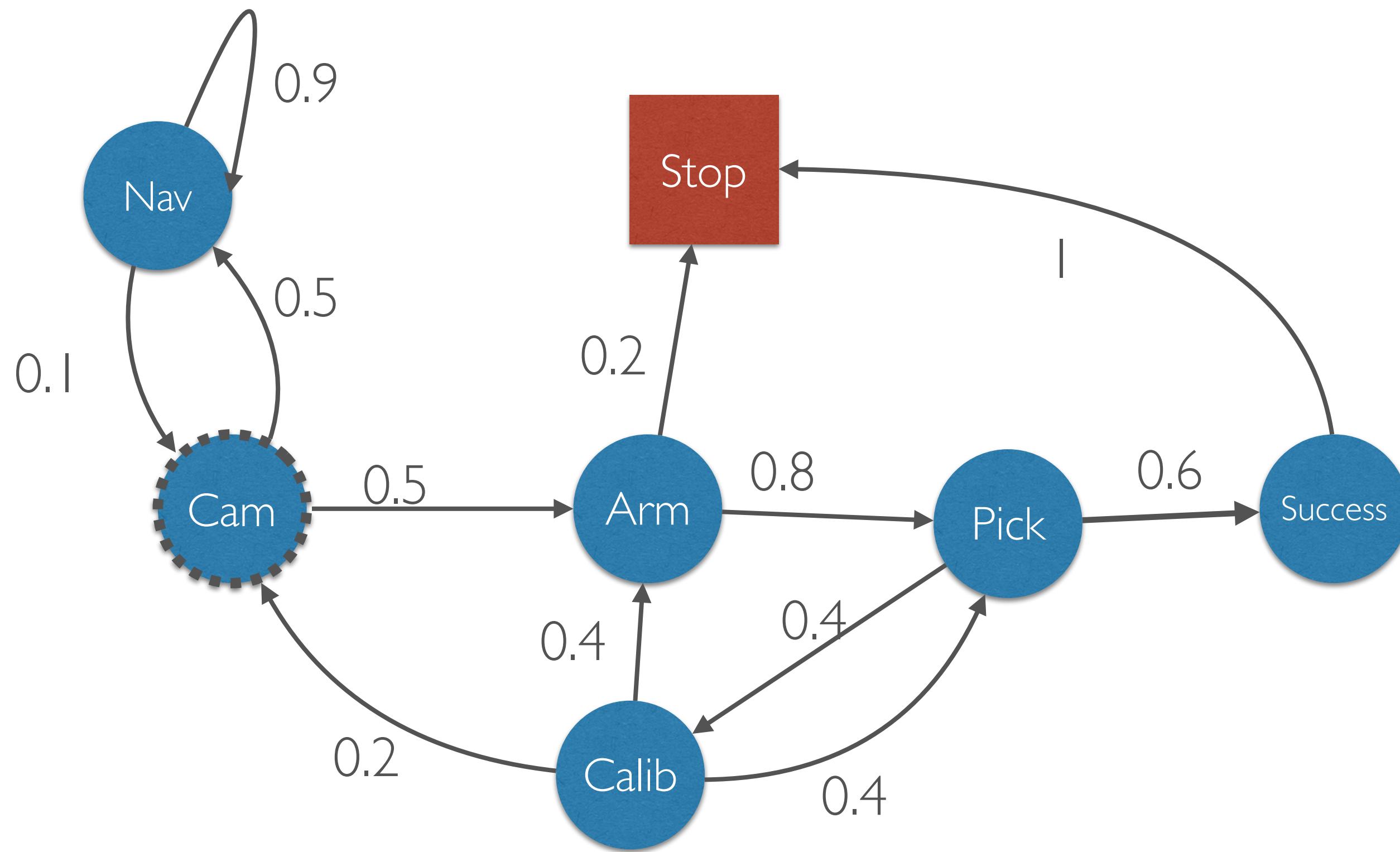


EXAMPLE: PICKING MARKOV CHAIN

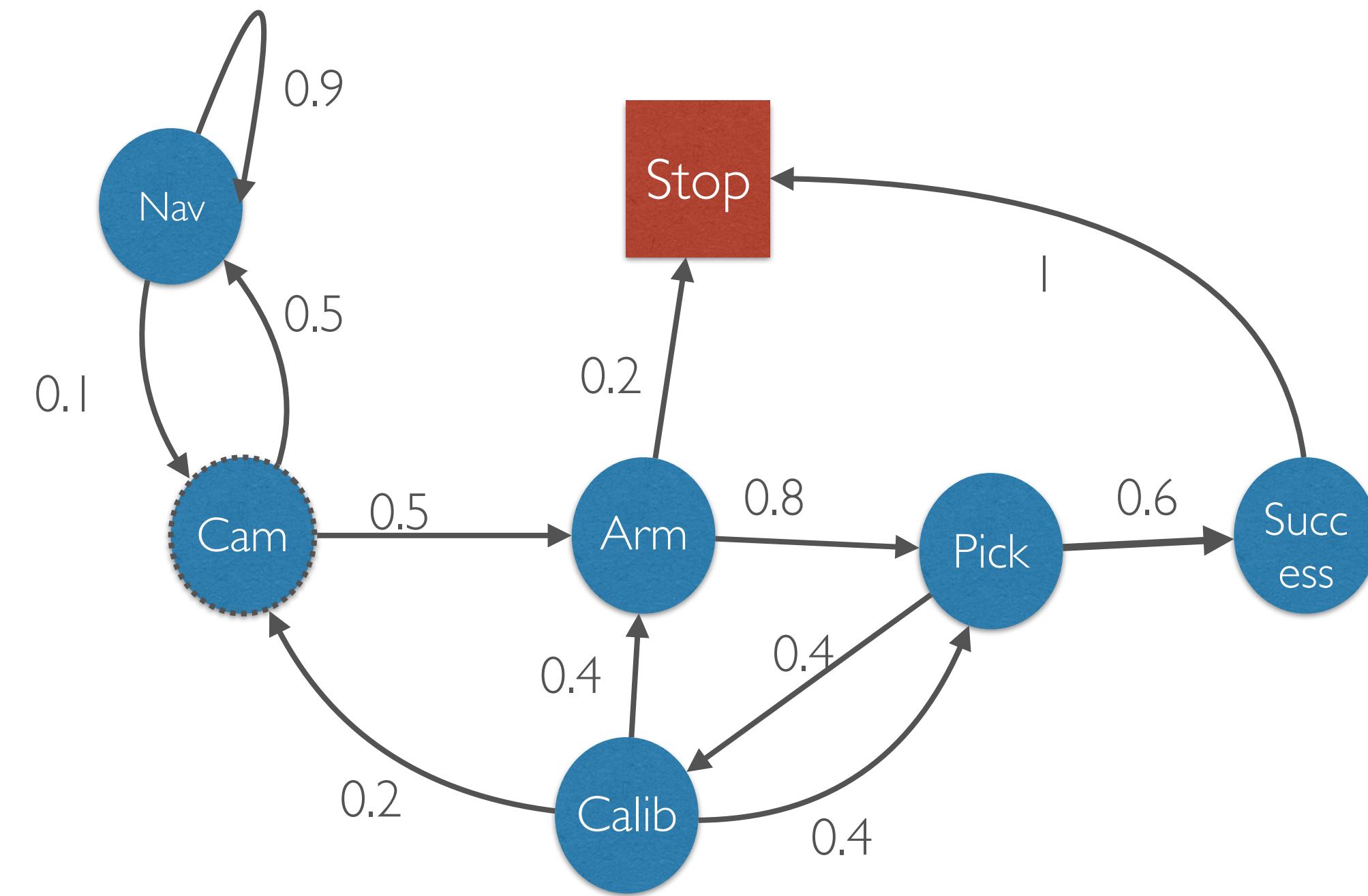


LUND
UNIVERSITY

EXAMPLE: PICKING MARKOV CHAIN



EXAMPLE: PICKING MARKOV CHAIN



Sample **episodes**:

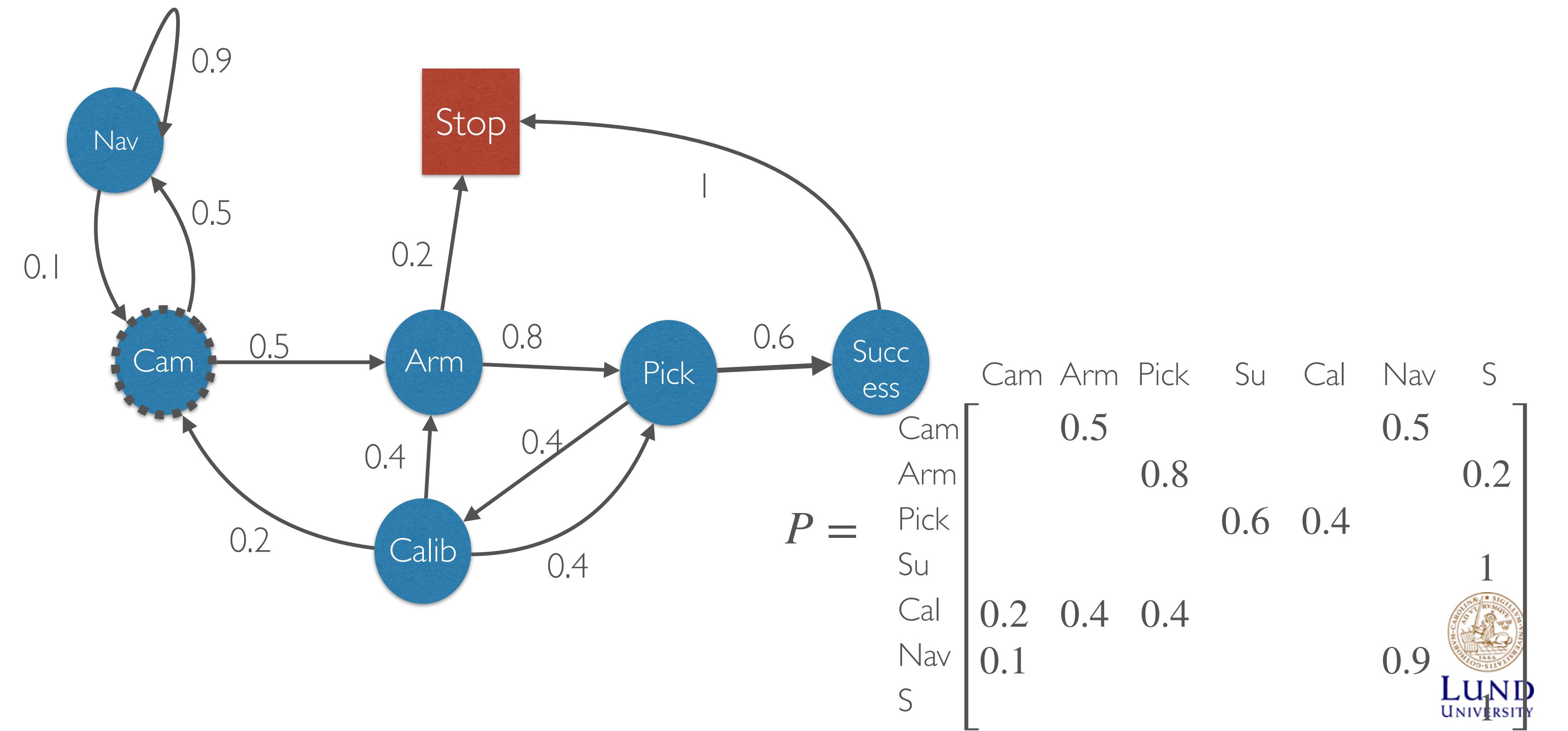
Start state $S_1 = \text{Cam}$

S_1, S_2, \dots, S_T

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



EXAMPLE: PICKING MARKOV CHAIN



MARKOV REWARD PROCESSES



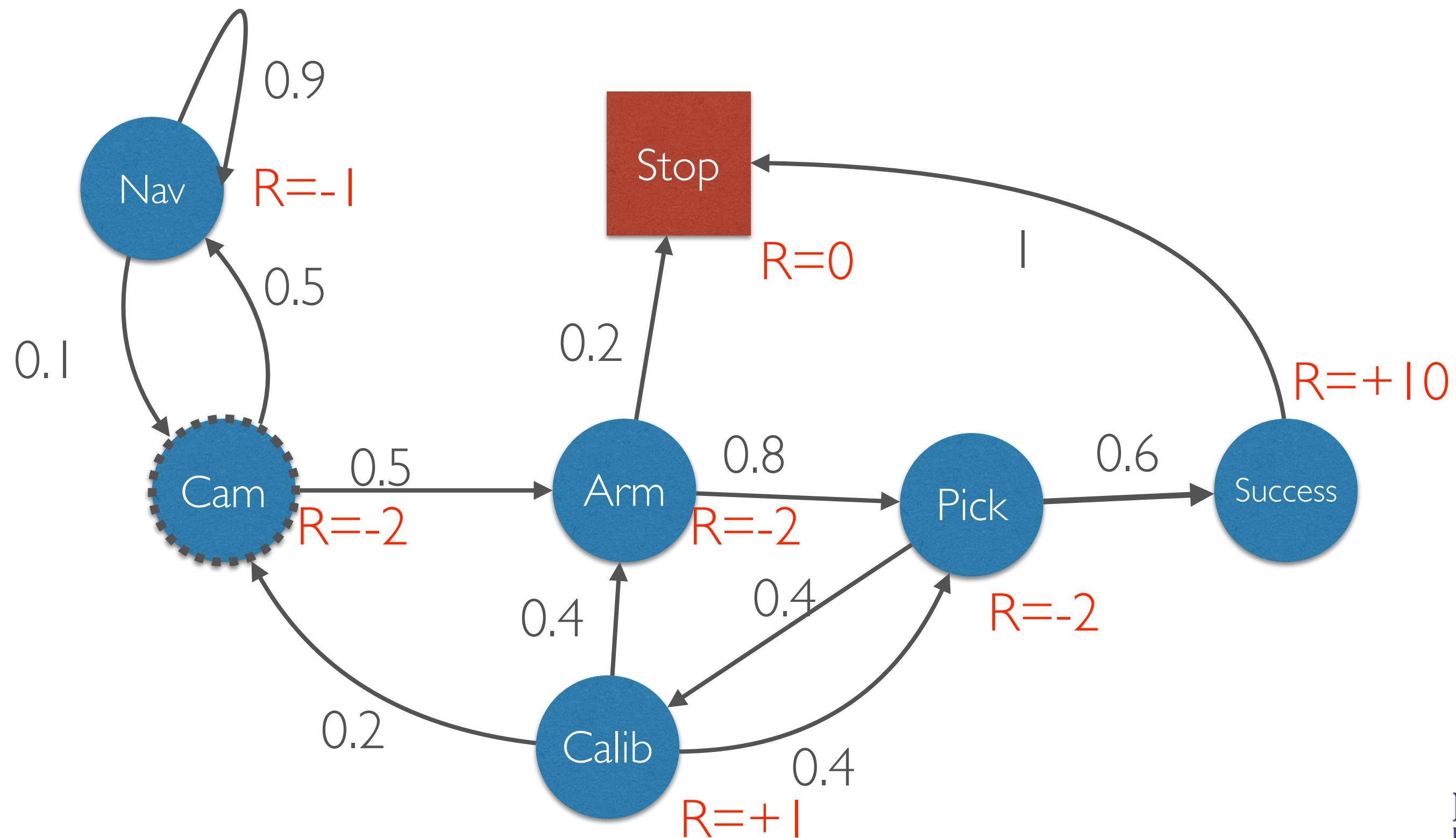
LUND
UNIVERSITY

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 $P_{ss'} = P(S_{t+1} = s' | S_t = s)$
 - \mathbf{R} is a reward function $R_s = E(R_{t+1} | S_t = s)$
 - γ 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 .

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

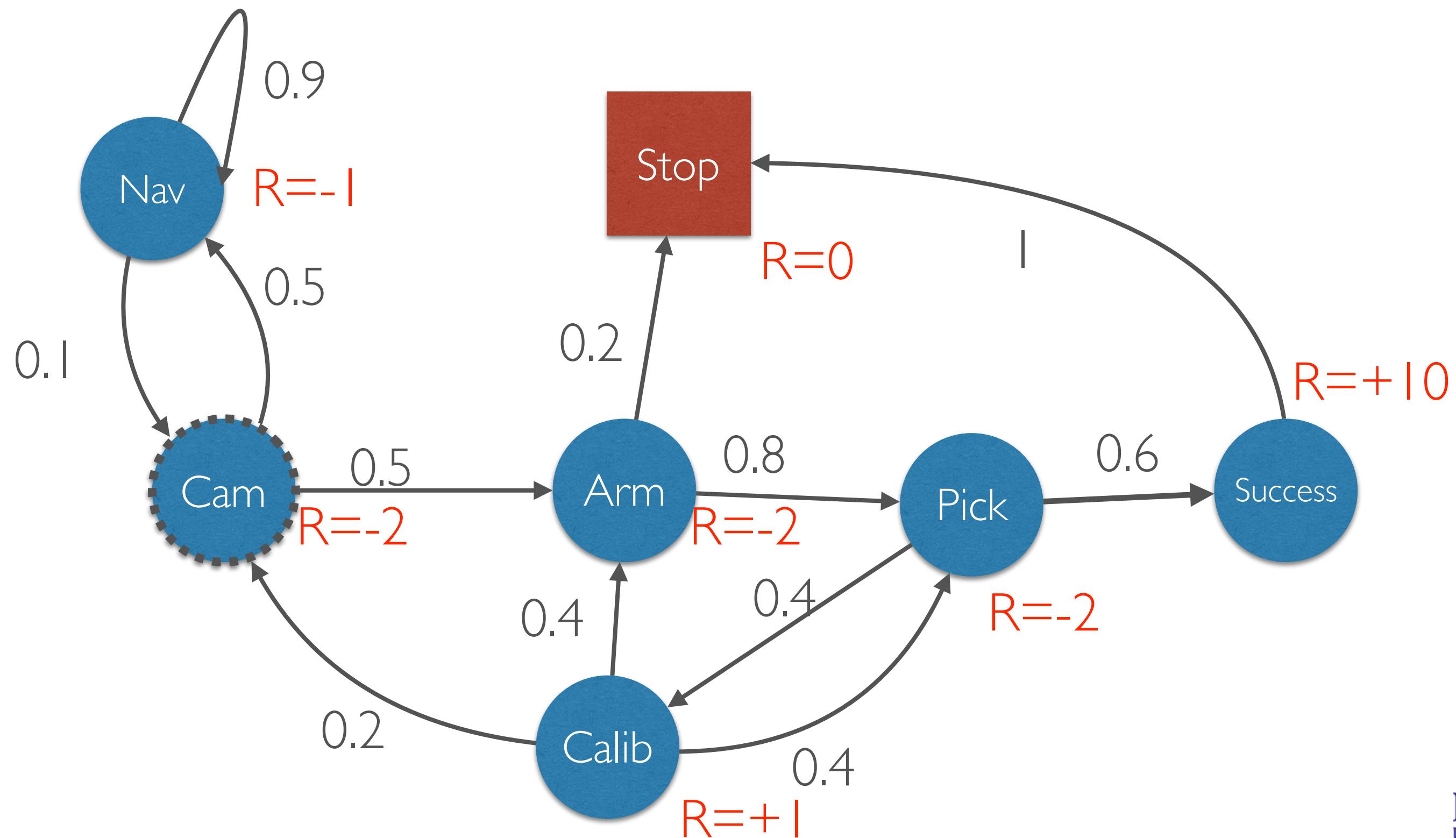
- The *discount* $\gamma \in [0,1]$ weights the future rewards.
- The value of a reward R that is $k+1$ steps into the future is $\gamma^k R$
 - Short-sighted: γ close to 0
 - Far-sighted: γ close to 1

ROLE OF THE DISCOUNT FACTOR

- Mathematically convenient as γ^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**:

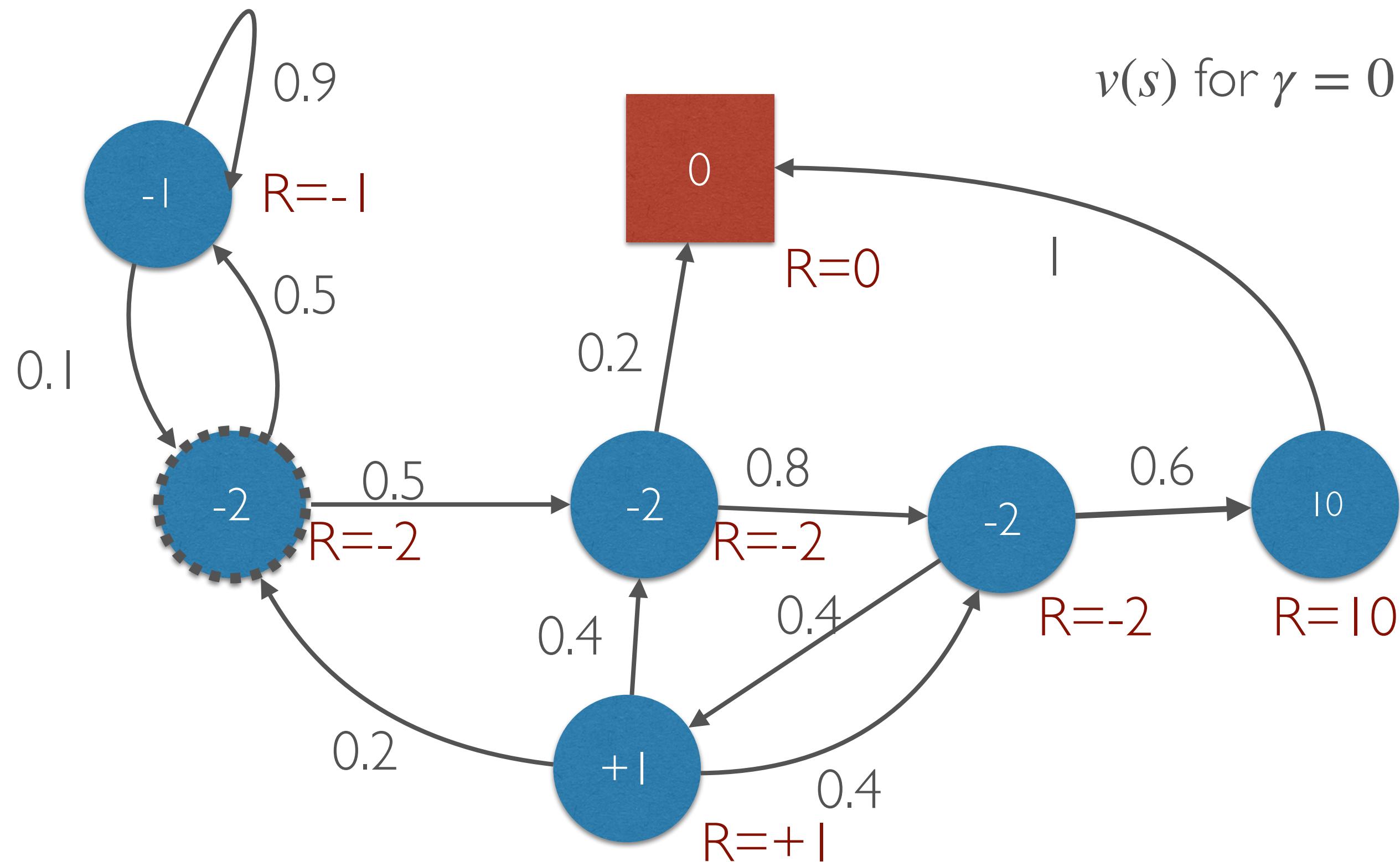
- Cam, Nav, Cam, Arm, Pick, Su, S
- Cam, Arm, Pick, Cal, Pick, Su, S
- Can, Nav, Cam, Arm, S

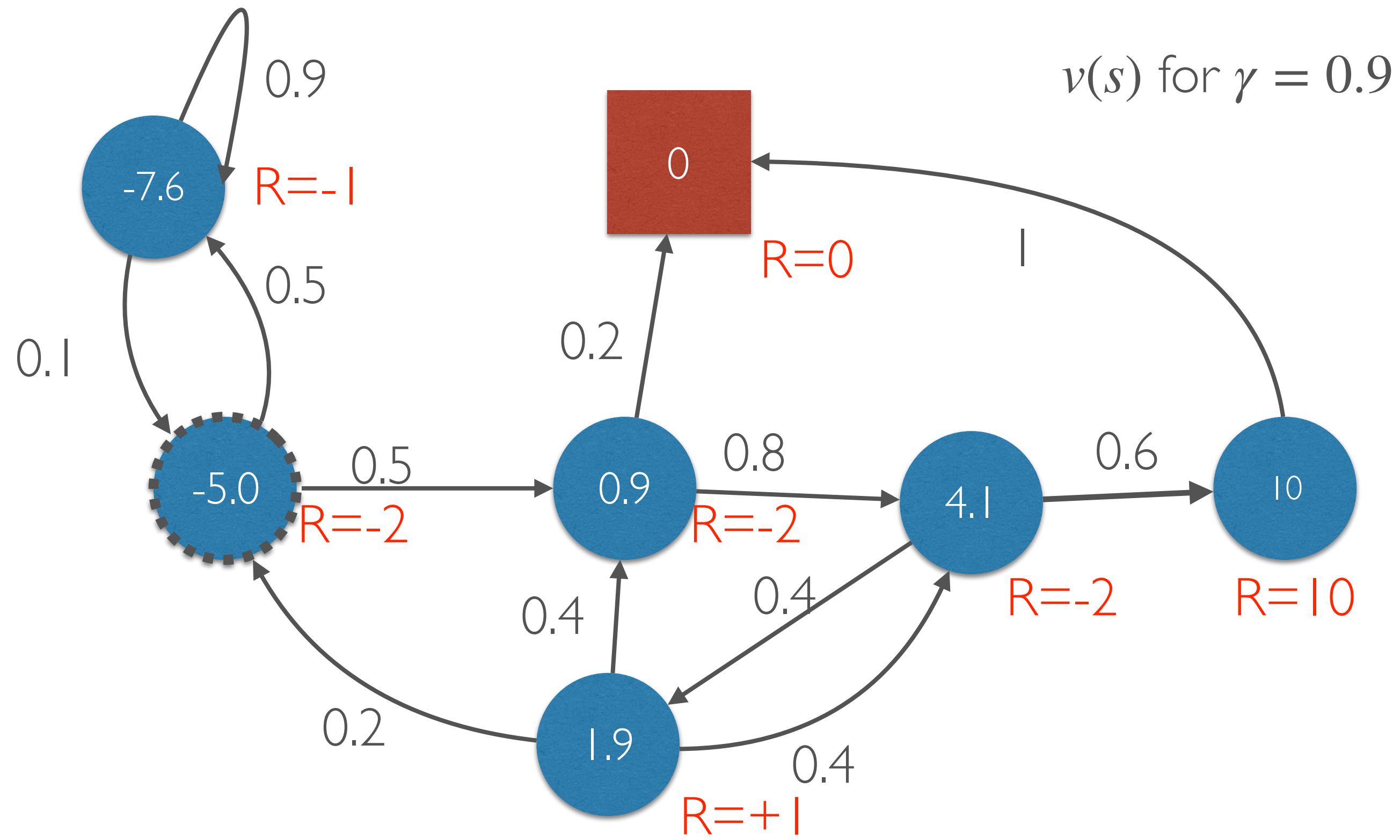
$$v_1 = -2 + \frac{1}{2}(-1) + \frac{1}{4}(-2) + \frac{1}{8}(-2) + \frac{1}{16}(-2) + \frac{1}{32}10 = -3.28$$

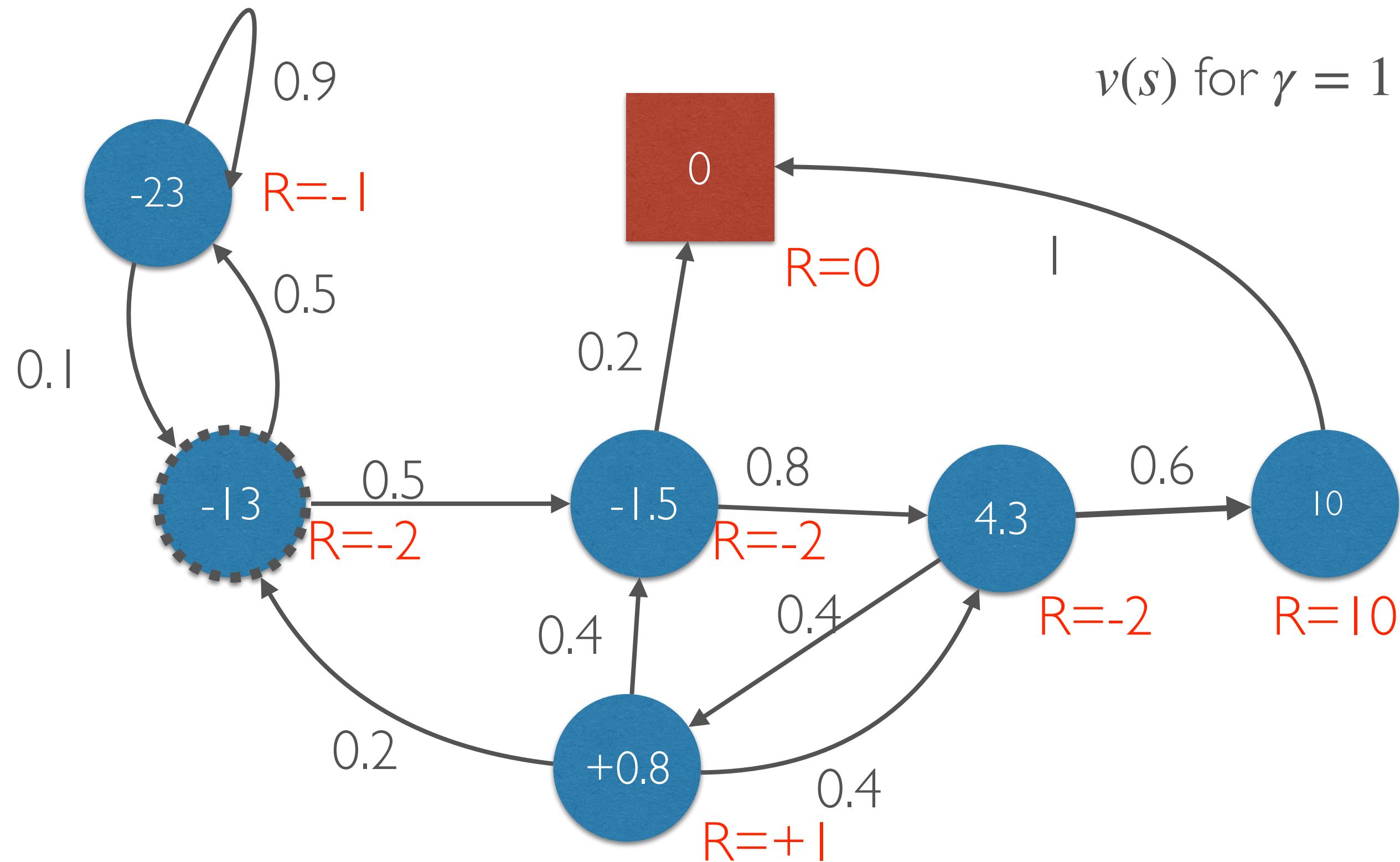
$$v_1 = -2 + \frac{1}{2}(-2) + \frac{1}{4}(-2) + \frac{1}{8}(1) + \frac{1}{16}(-2) + \frac{1}{32}10 = -3.18$$

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









BELLMAN EQUATION FOR MRP

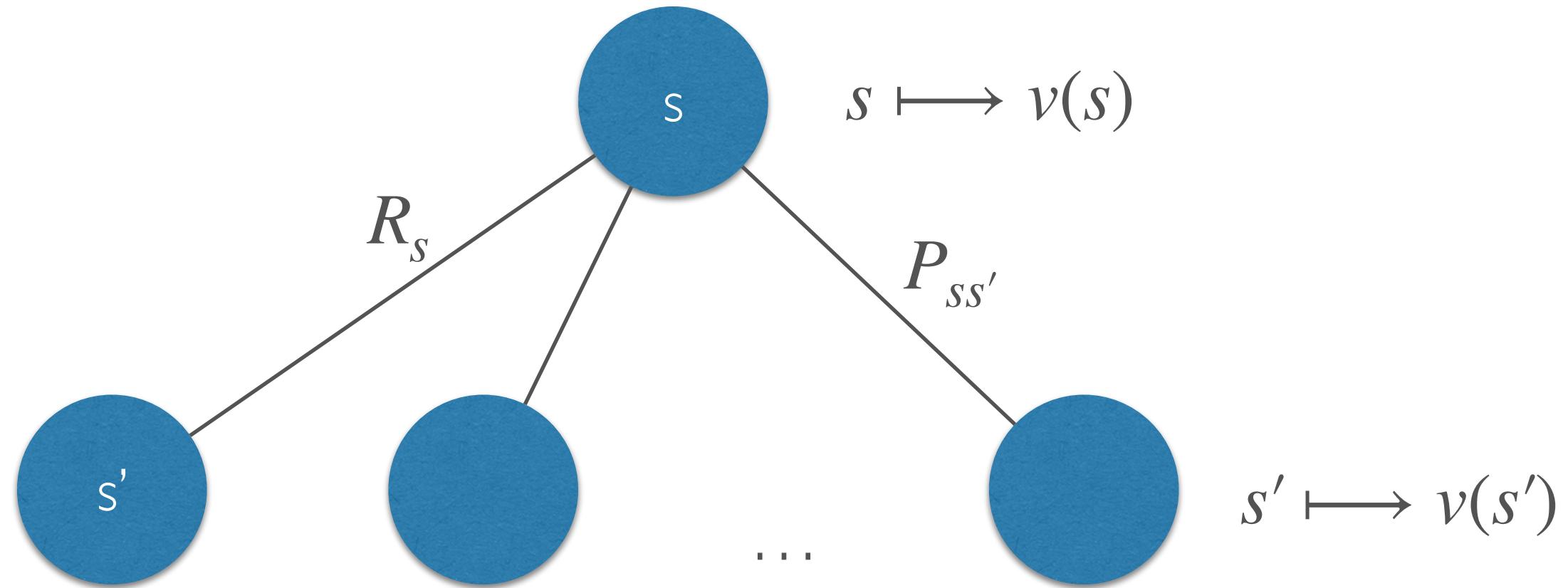
- The value function can be compute recursively

- The immediate reward R_{t+1}

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

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

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

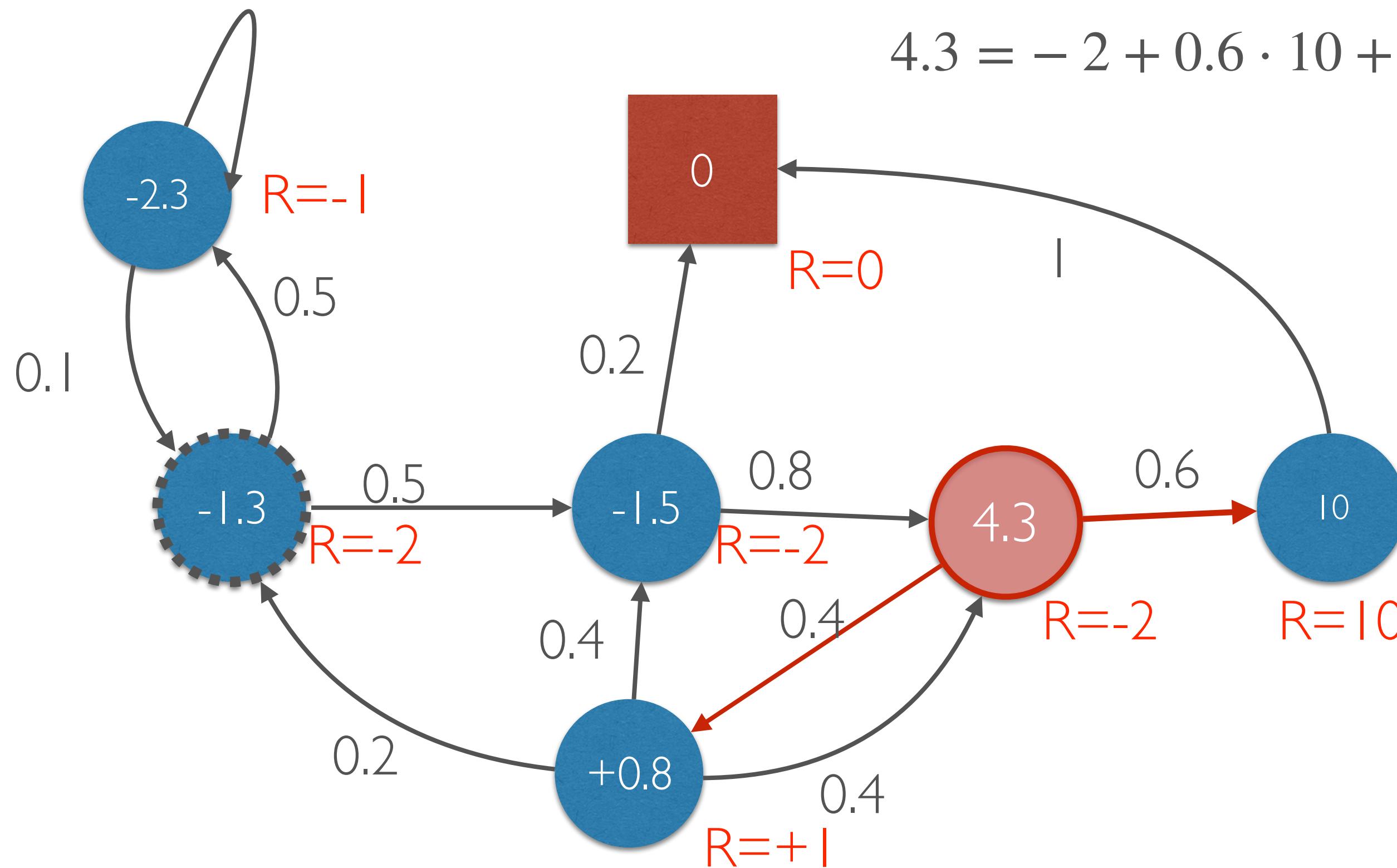


Marginalizing over the next possible states:

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

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

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



The Bellman equation can be expressed using matrices

$$v = R + \gamma Pv$$

where v is a column vector with one entry for each state

$$\begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix} = \begin{bmatrix} R_1 \\ \vdots \\ R_n \end{bmatrix} + \gamma \begin{bmatrix} P_{11} & \dots & P_{1n} \\ \vdots & & \\ P_{n1} & \dots & P_{nn} \end{bmatrix} \begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix}$$



- The Bellman equation is linear. It can be solved directly

$$v = R + \gamma Pv$$

$$(I - \gamma R) v = R$$

$$v = (I - \gamma P)^{-1} R$$

- Computational complexity is $\mathcal{O}(n^3)$ for n states
- Direct solution only possible for small MRPs
- Many iterative methods for solving large MRPs



MARKOV DECISION PROCESSES



LUND
UNIVERSITY

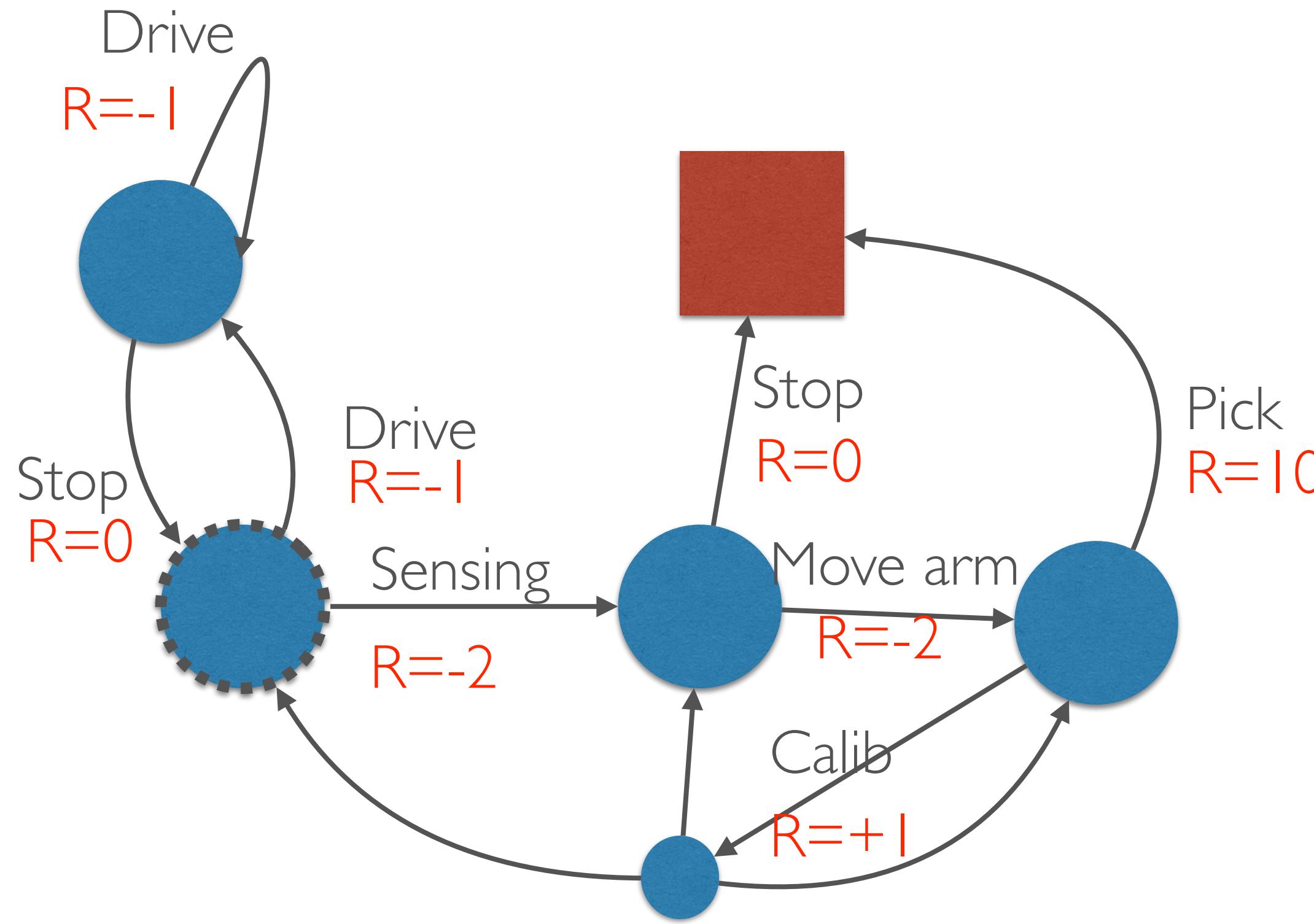
MARKOV DECISION PROCESS

- 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, γ)

- 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 $R_s^{\textcolor{red}{a}} = E(R_{t+1} | S_t = s, A_t = \textcolor{red}{a})$
- γ is a discount factor $\gamma \in [0,1]$





LUND
UNIVERSITY

POLICIES:WHAT DO TO WHEN

Definition: A **policy** π 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 π
- If we marginalise out (average over) the actions

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

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

- 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 π

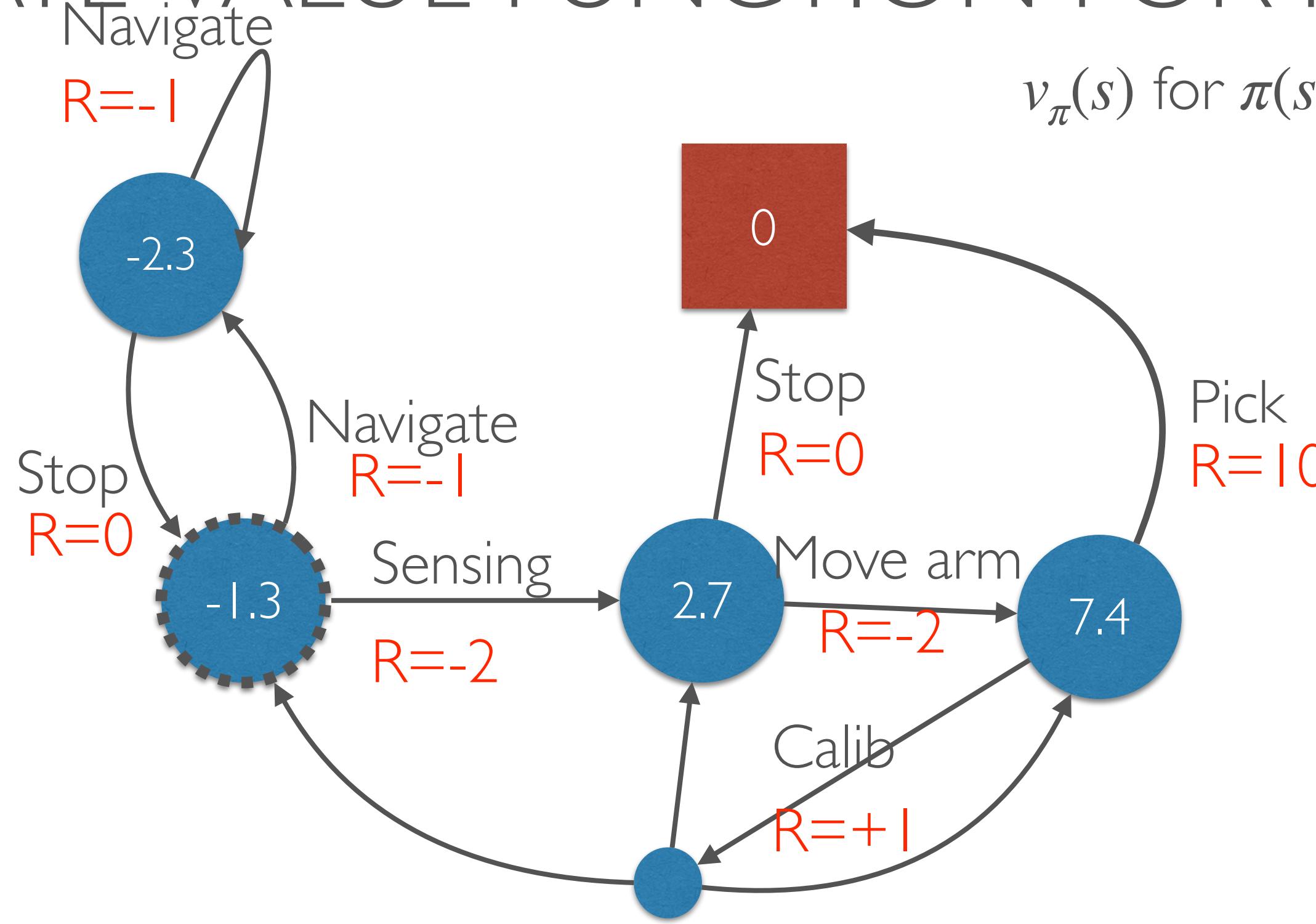
$$v_\pi(s) = E_\pi \{ G_t | S_t = s \}$$

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 π

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



STATE-VALUE FUNCTION FOR PICKING



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



BELLMAN EXPECTATION EQUATION

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

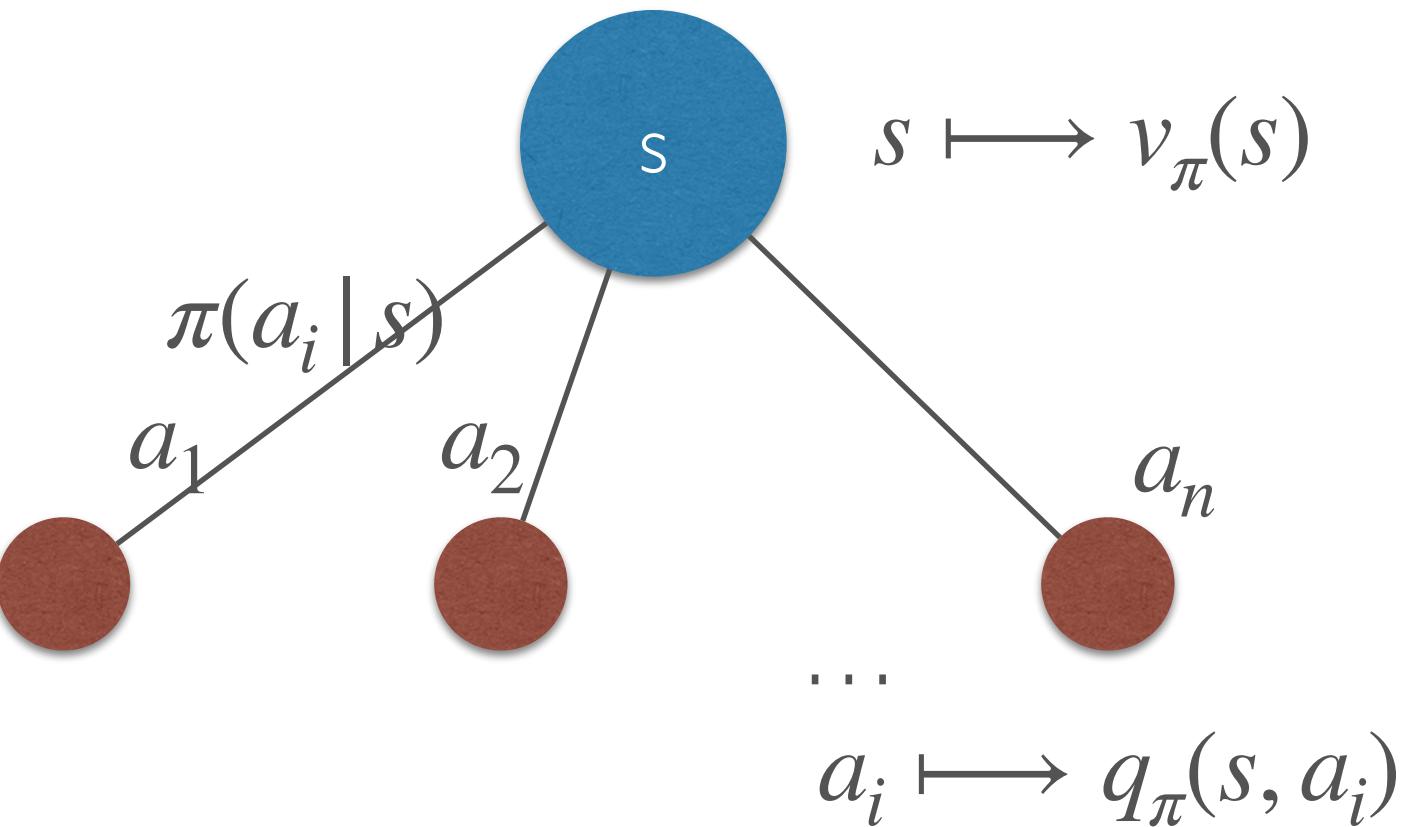
$$v_{\pi}(s) = E_{\pi} \{ R_{t+1} + \gamma v_{\pi}(S_{t+1}) \mid S_t = s \}$$

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



BELLMAN EXPECTATION EQUATION

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



Averaging / marginalising over a

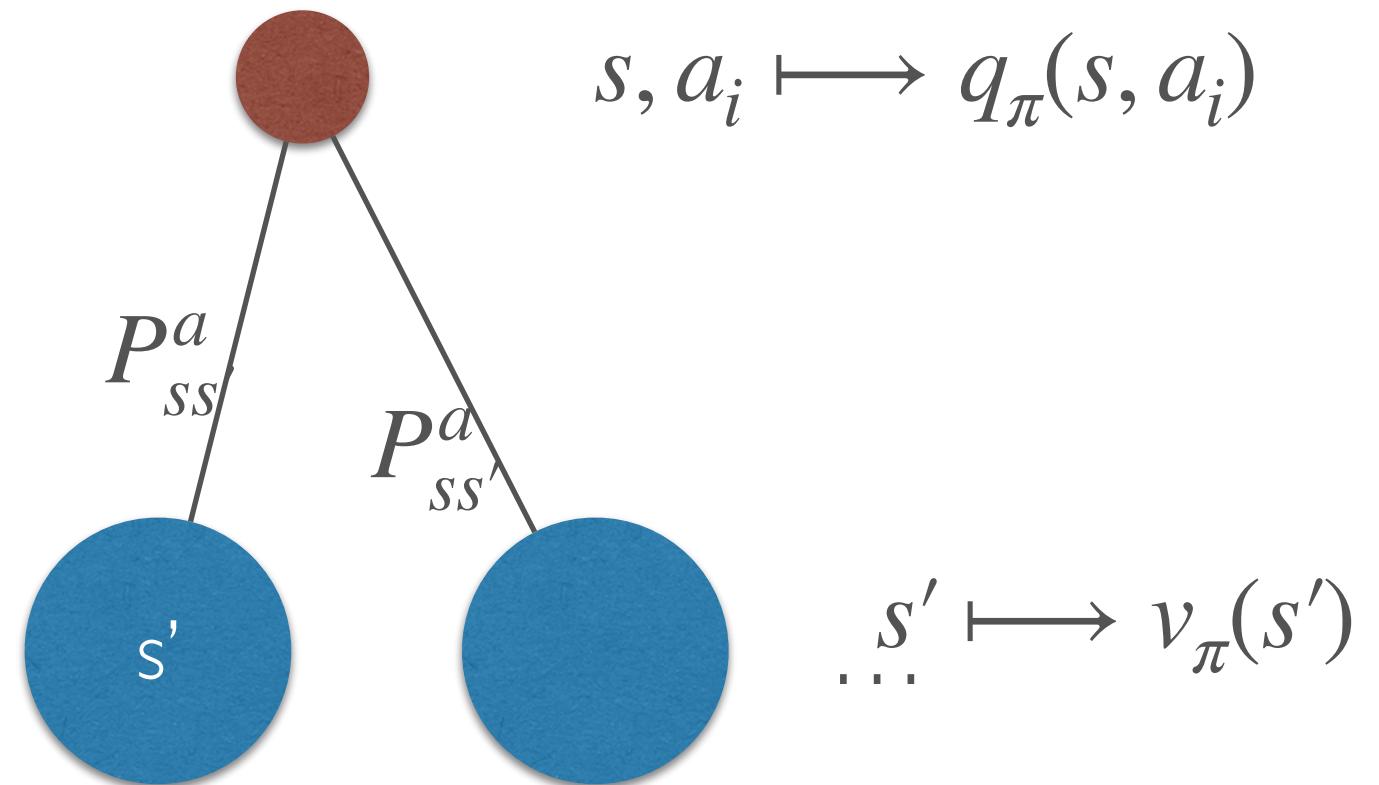
$$v_\pi(s) = \sum_{a \in A} \pi(a | s) q_\pi(s, a)$$



BELLMAN EXPECTATION EQUATION

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

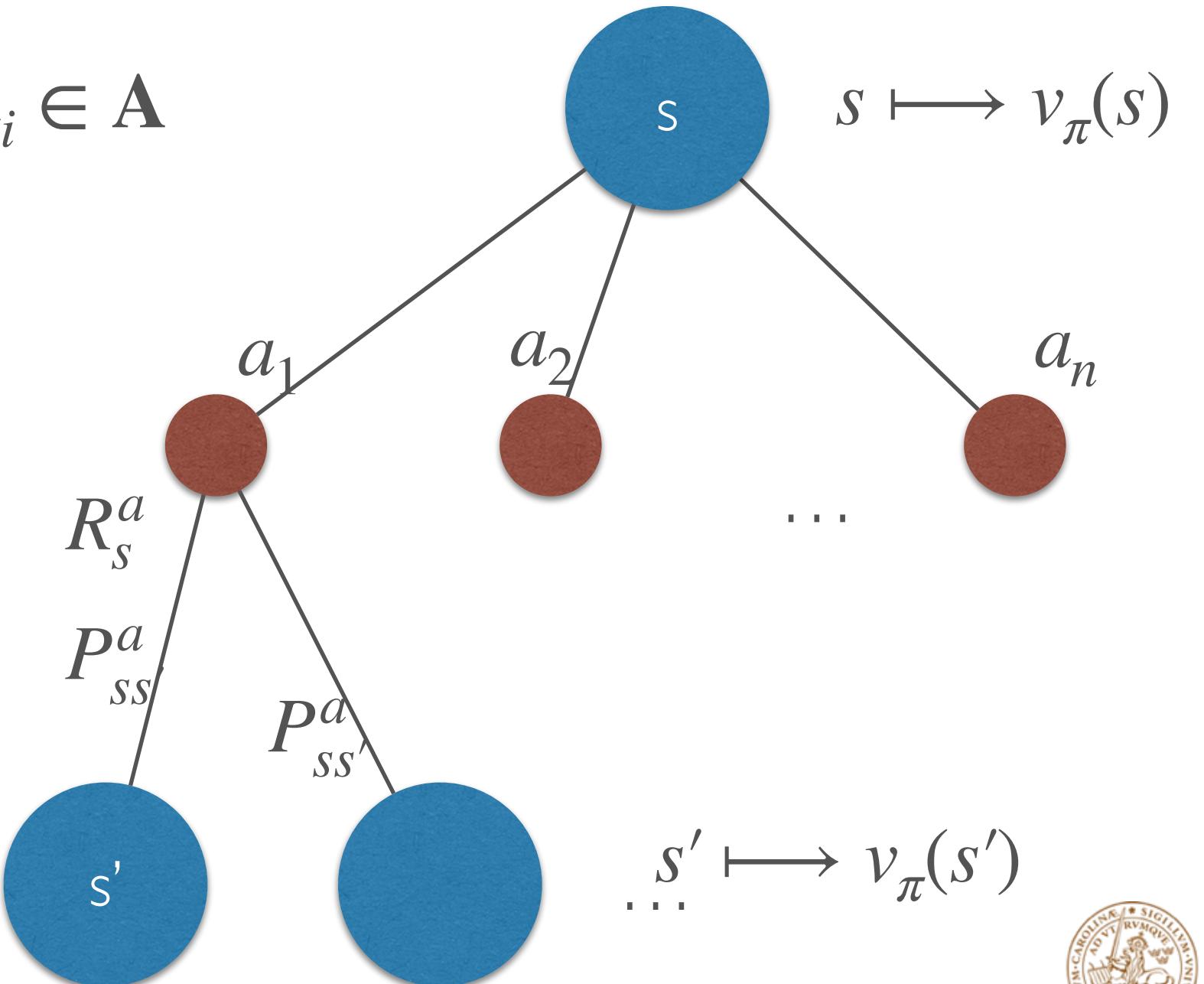
$$q_{\pi}(s, a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_{\pi}(s')$$



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

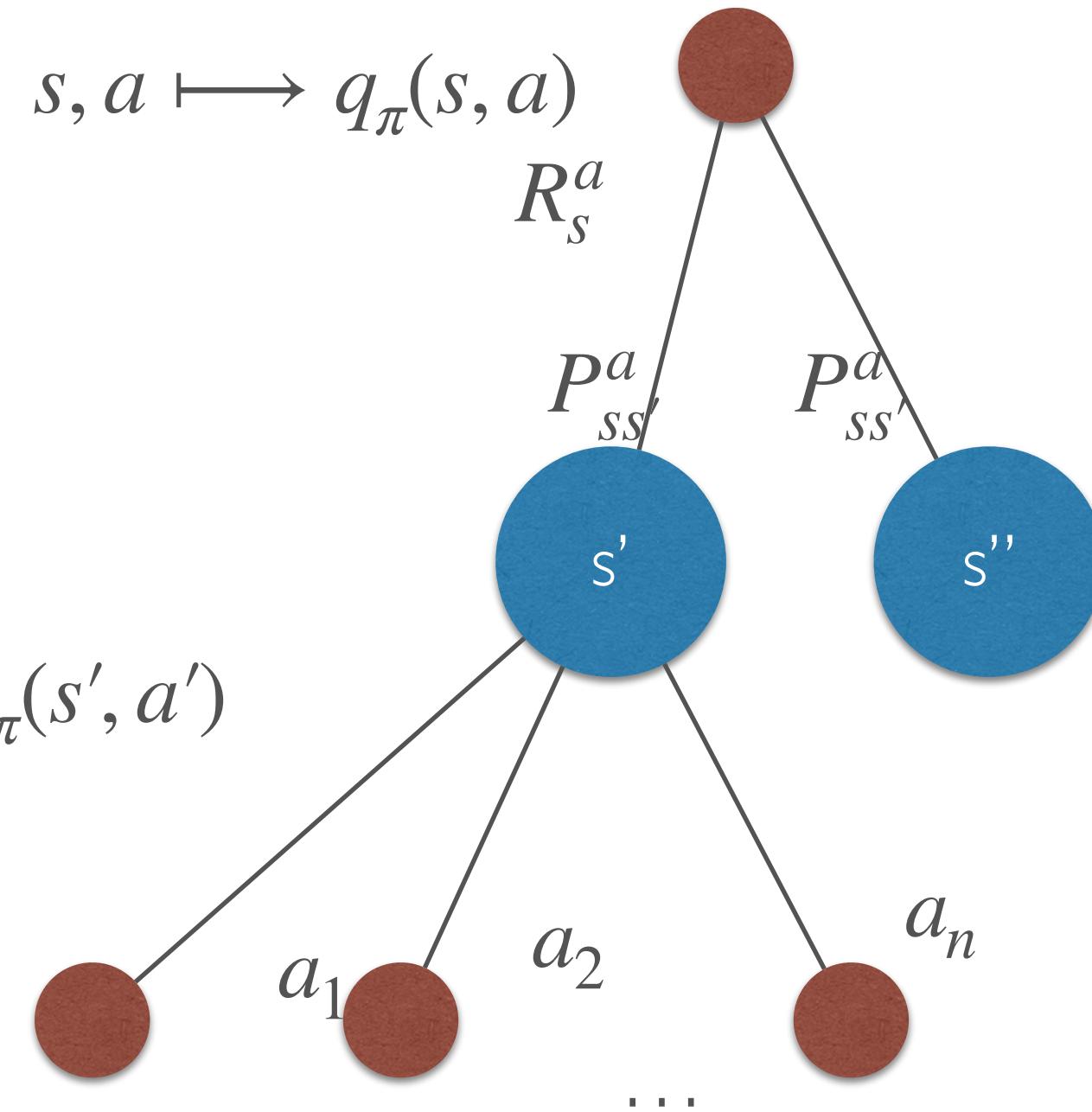
$$v_\pi(s) = \sum_{a \in A} \pi(a | s) \left(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s') \right)$$



BELLMAN EXPECTATION EQUATION

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

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

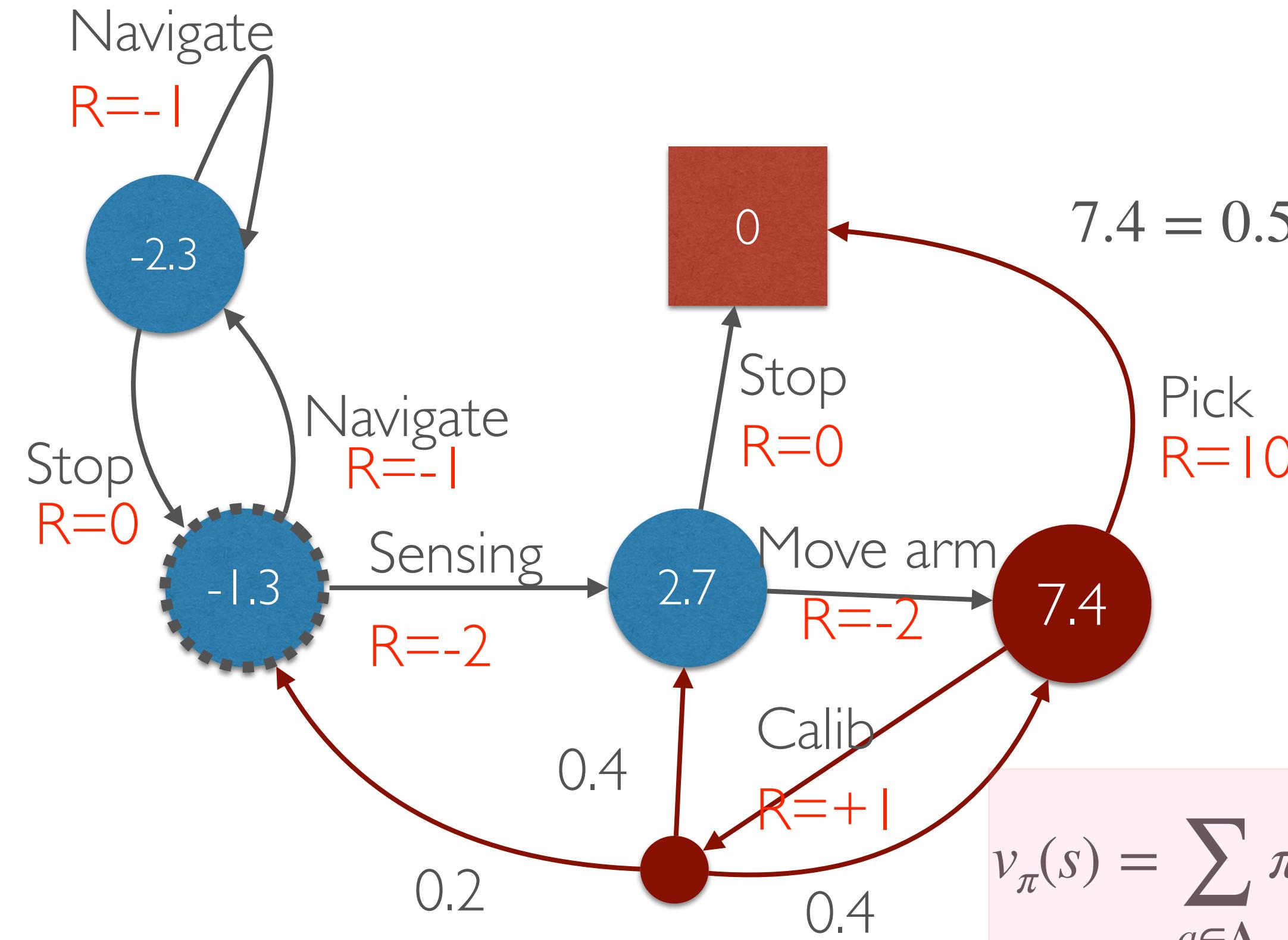


$$s, a \longmapsto q_{\pi}(s, a)$$



BELLMAN EXPECTATION EQUATION

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



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

OPTIMAL VALUE FUNCTION

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

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

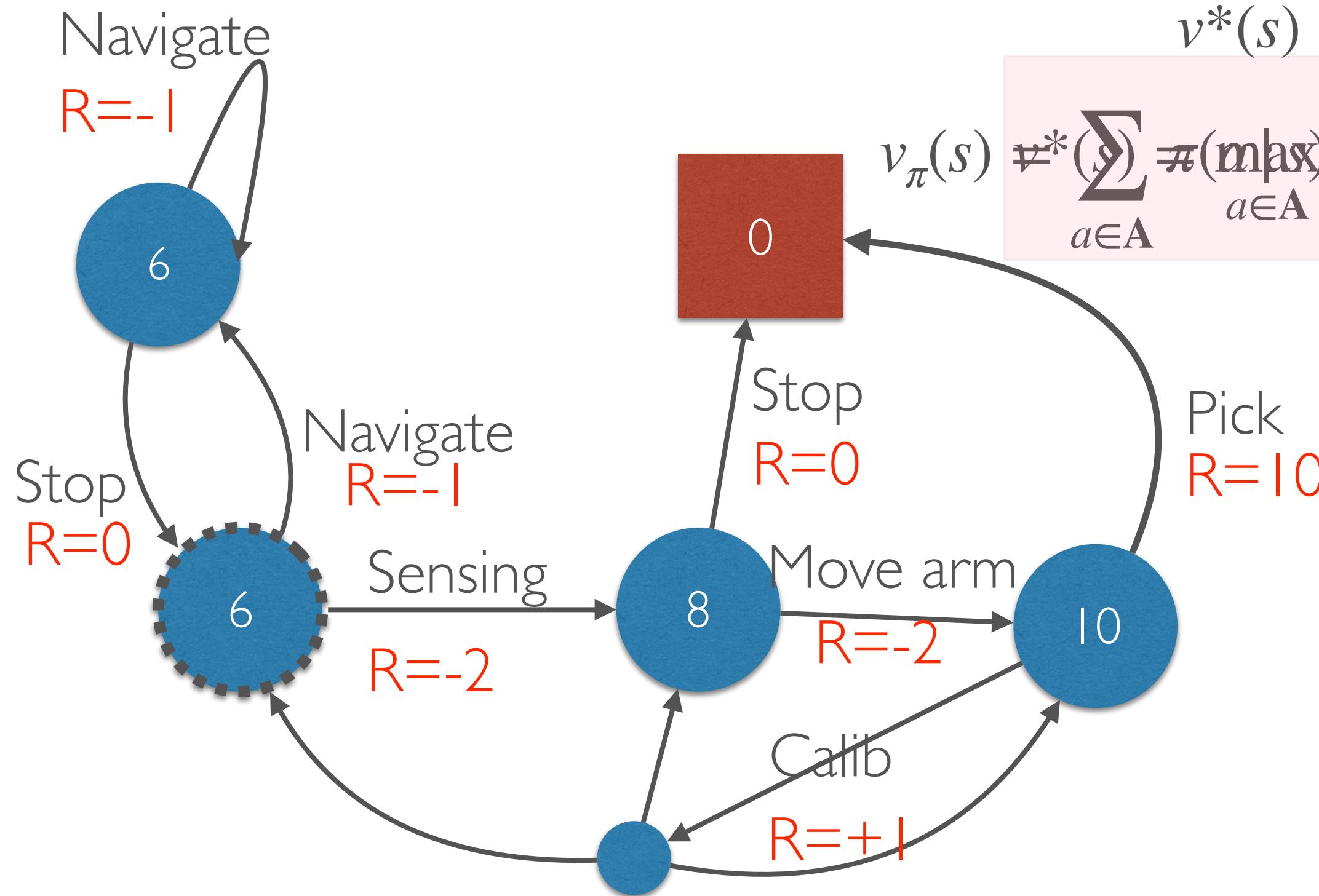
Definition: $v^*(s)$ is called the ***optimal state-value*** function

- Likewise, we can find the π 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

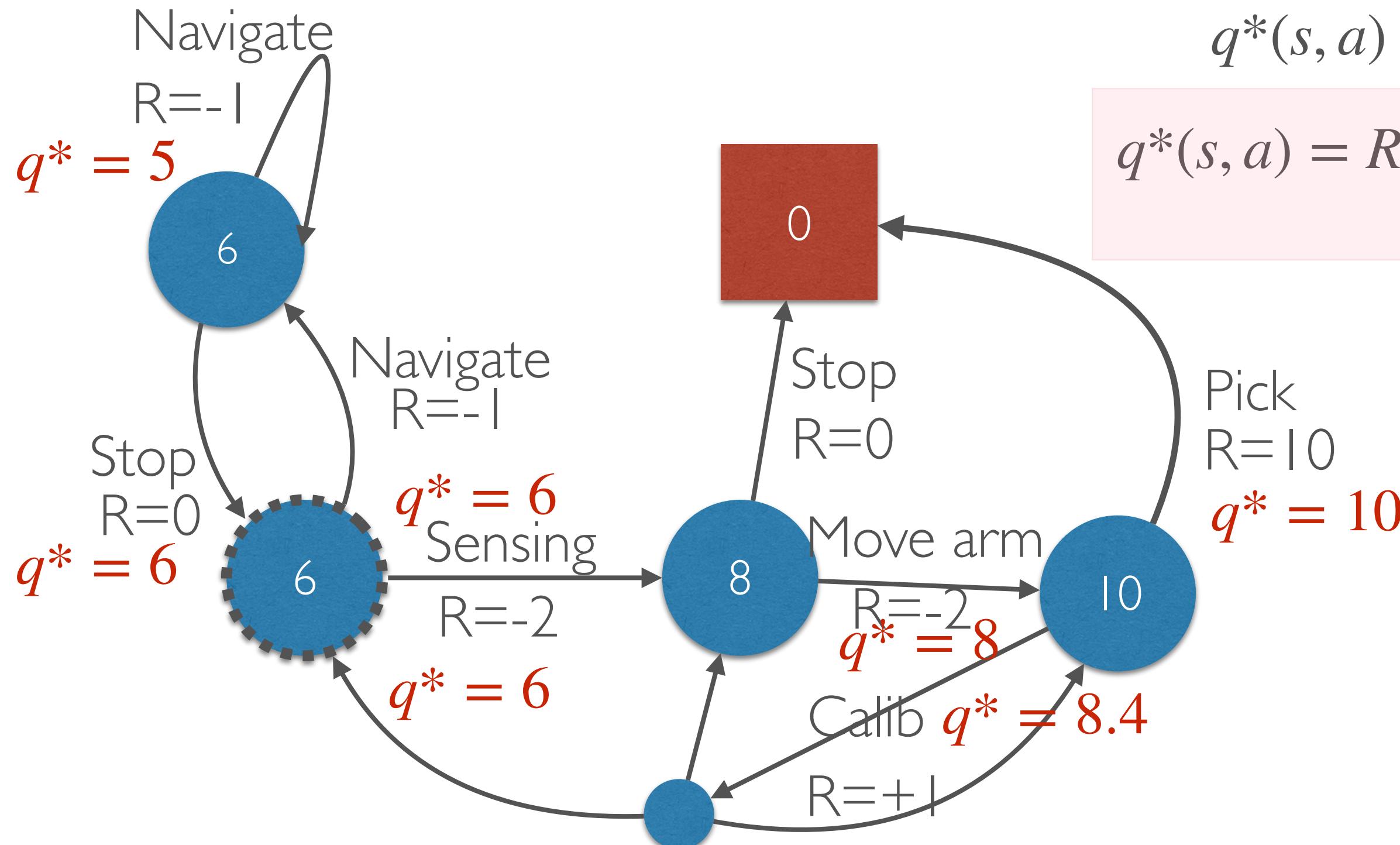


$v^*(s)$ for $\gamma = 1$

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



OPTIMAL ACTION-VALUE FUNCTION FOR PICKING



$q^*(s, a)$ for $\gamma = 1$

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



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 π^* 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 $q_{\pi^*}(s, a) = q^*(s, a)$



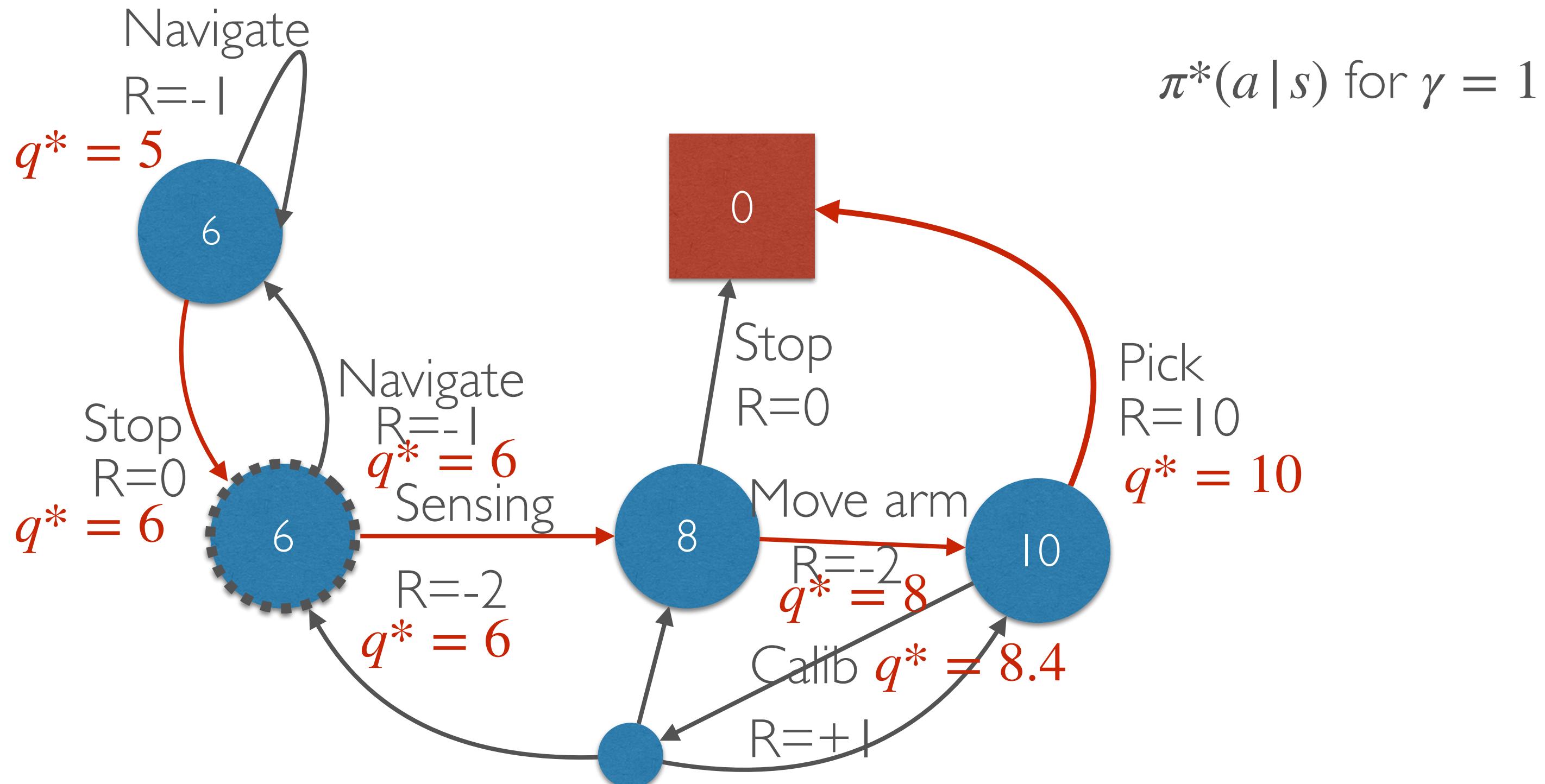
FINDING AN OPTIMAL POLICY

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

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

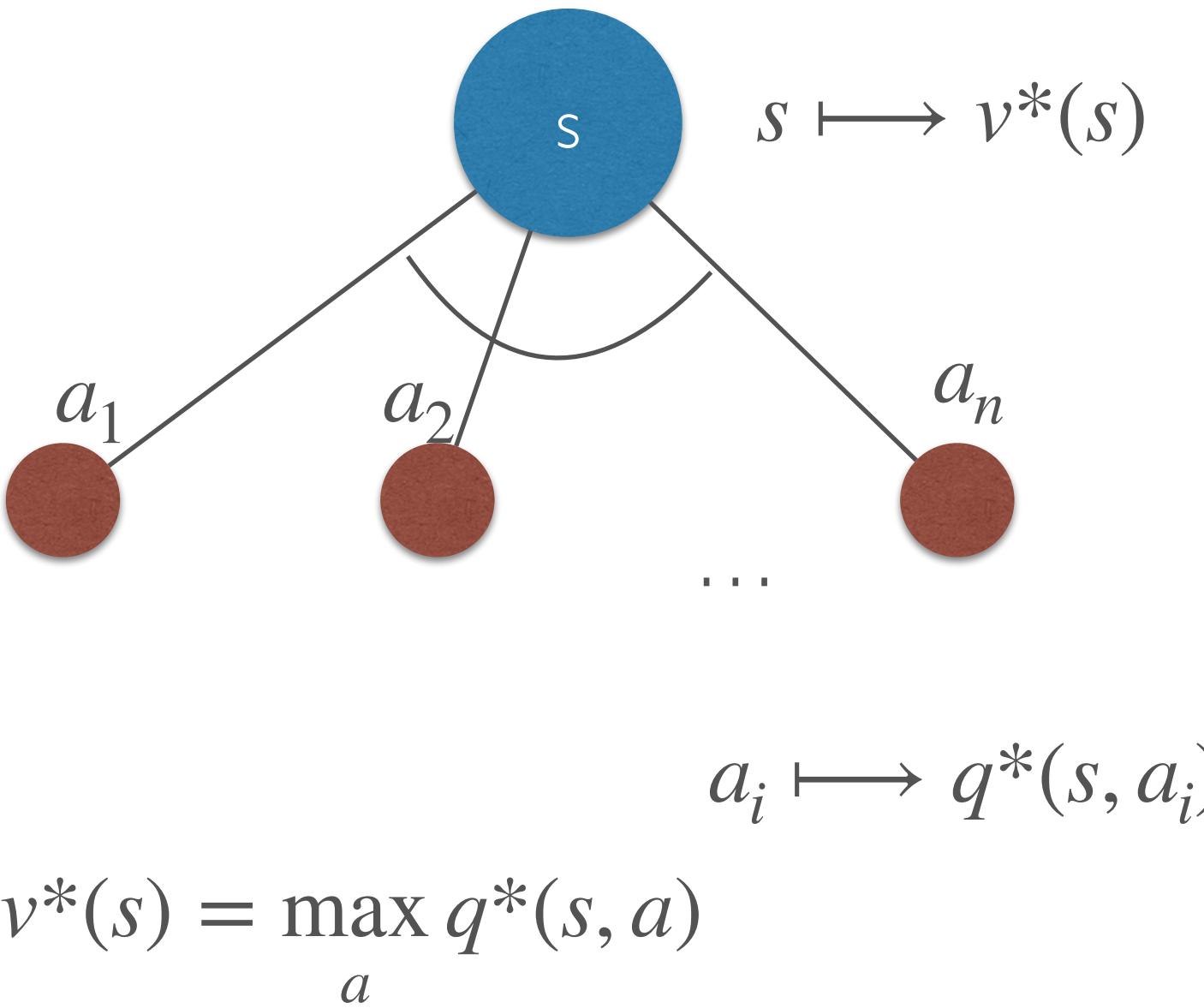
- There is always a deterministic optimal policy for any MDP
- If we know $q^*(s, a)$, we immediately have the optimal policy.





BELLMAN OPTIMALITY EQUATION

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

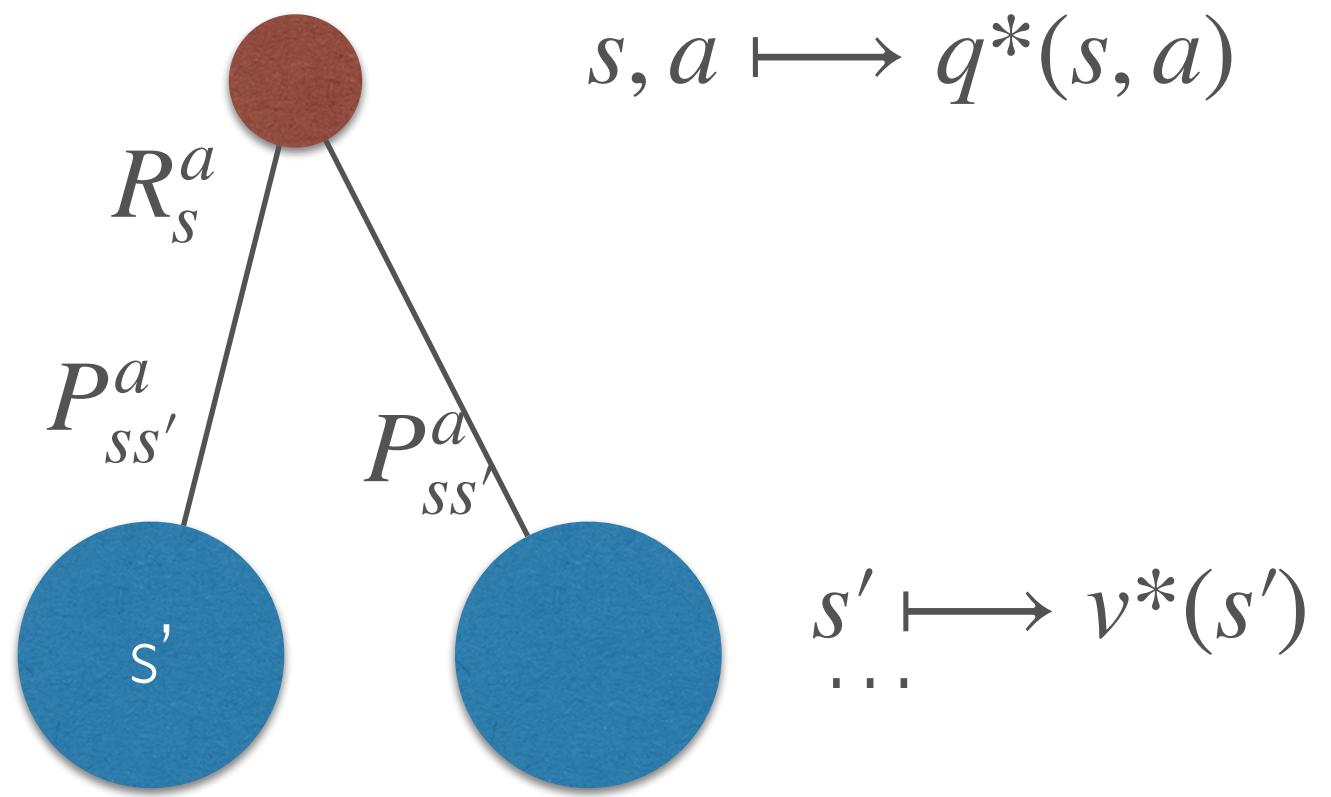


Maximizing over a

BELLMAN OPTIMALITY EQUATION

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

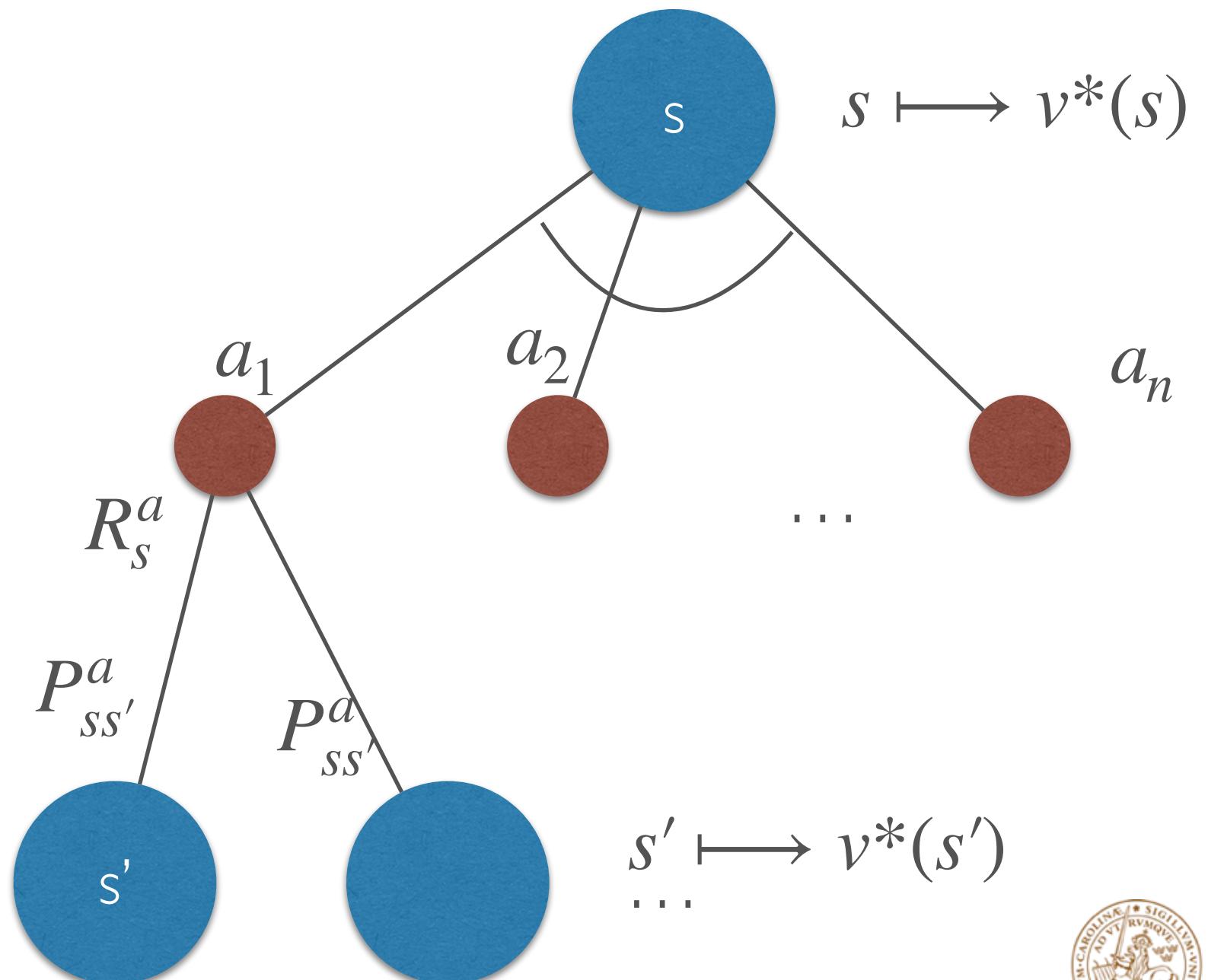
$$q^*(s, a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v^*(s')$$



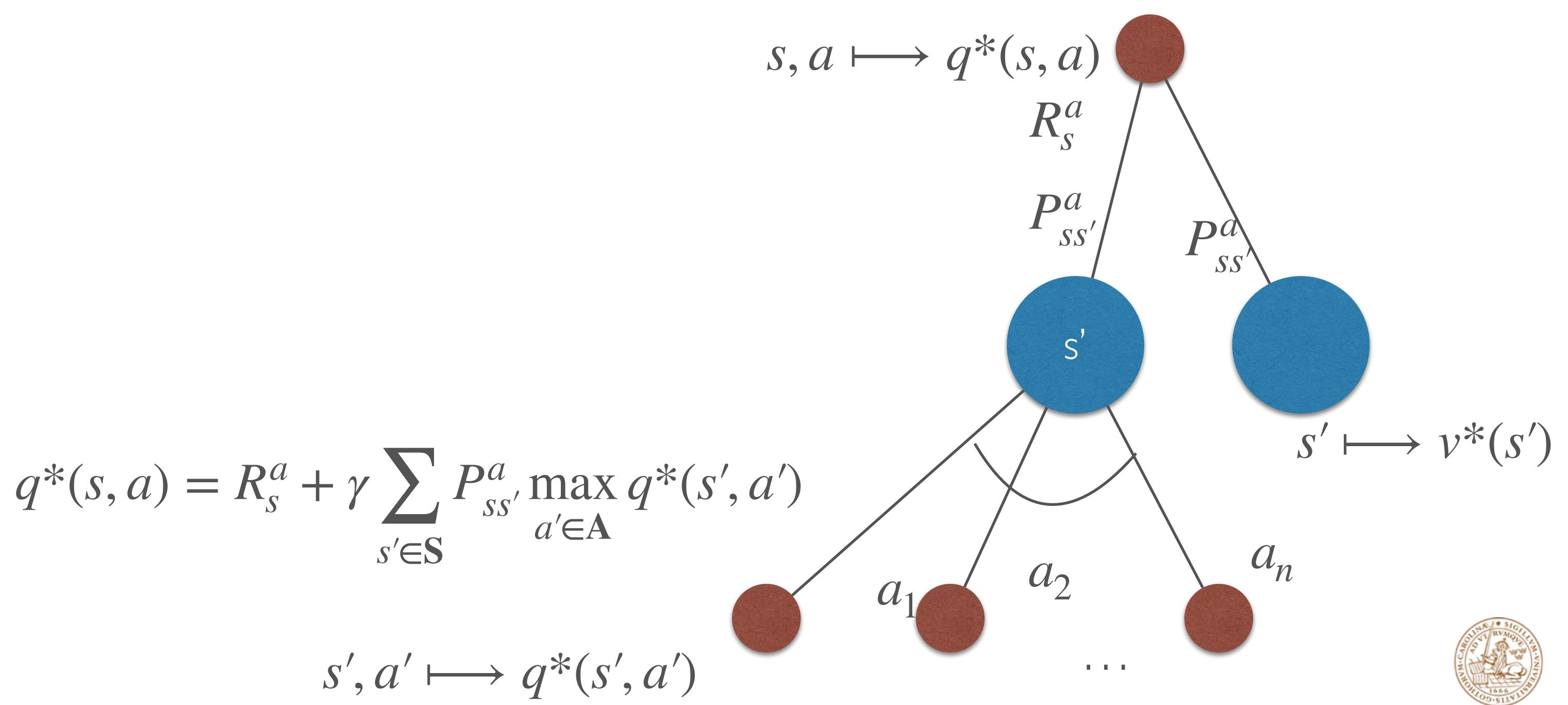
BELLMAN OPTIMALITY EQUATION

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

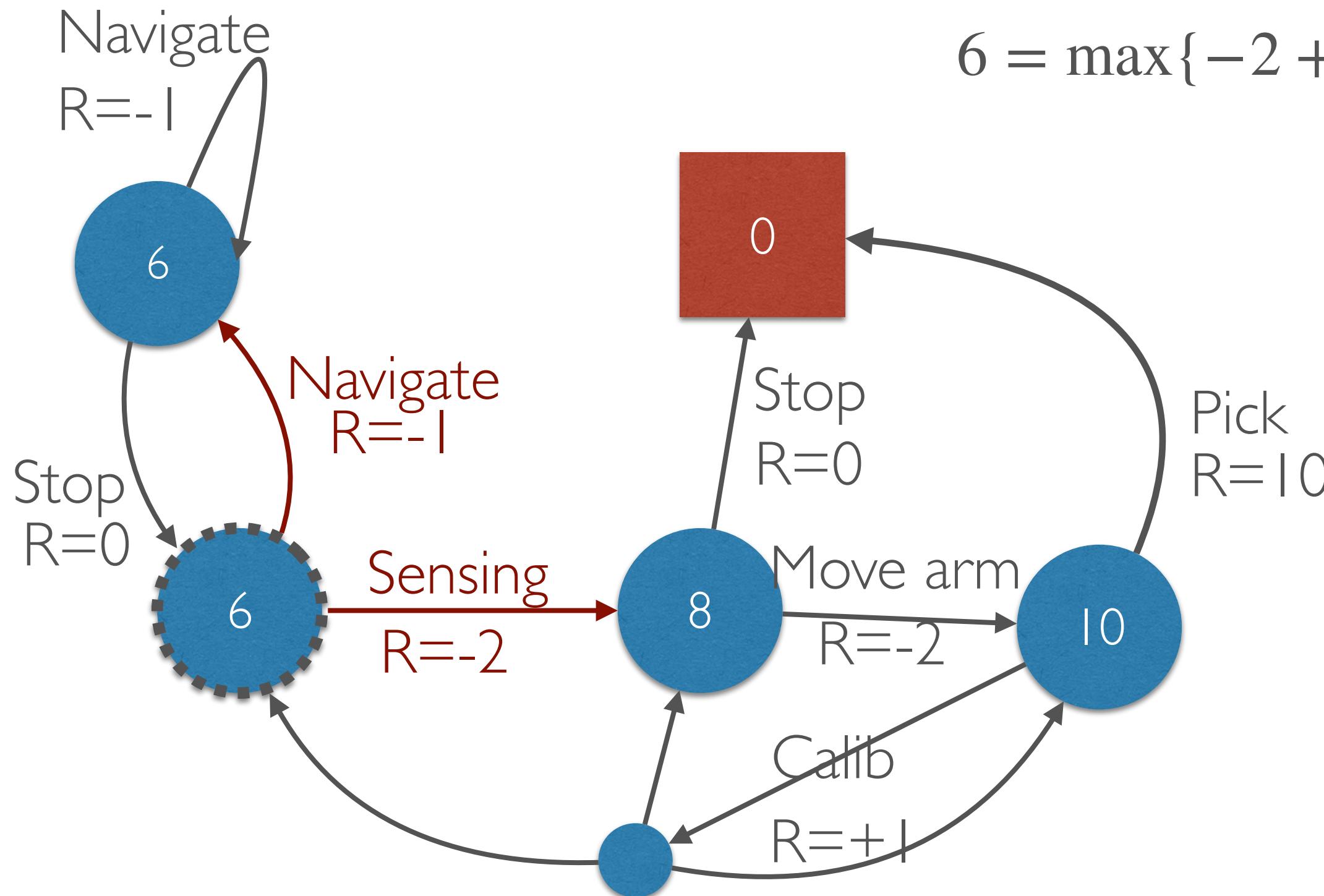
$$v_*(s) = \max_{a \in \mathbf{A}} R_s^a + \gamma \sum_{s' \in \mathbf{S}} P_{ss'}^a v^*(s')$$



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



POLICY ITERATION



LUND
UNIVERSITY

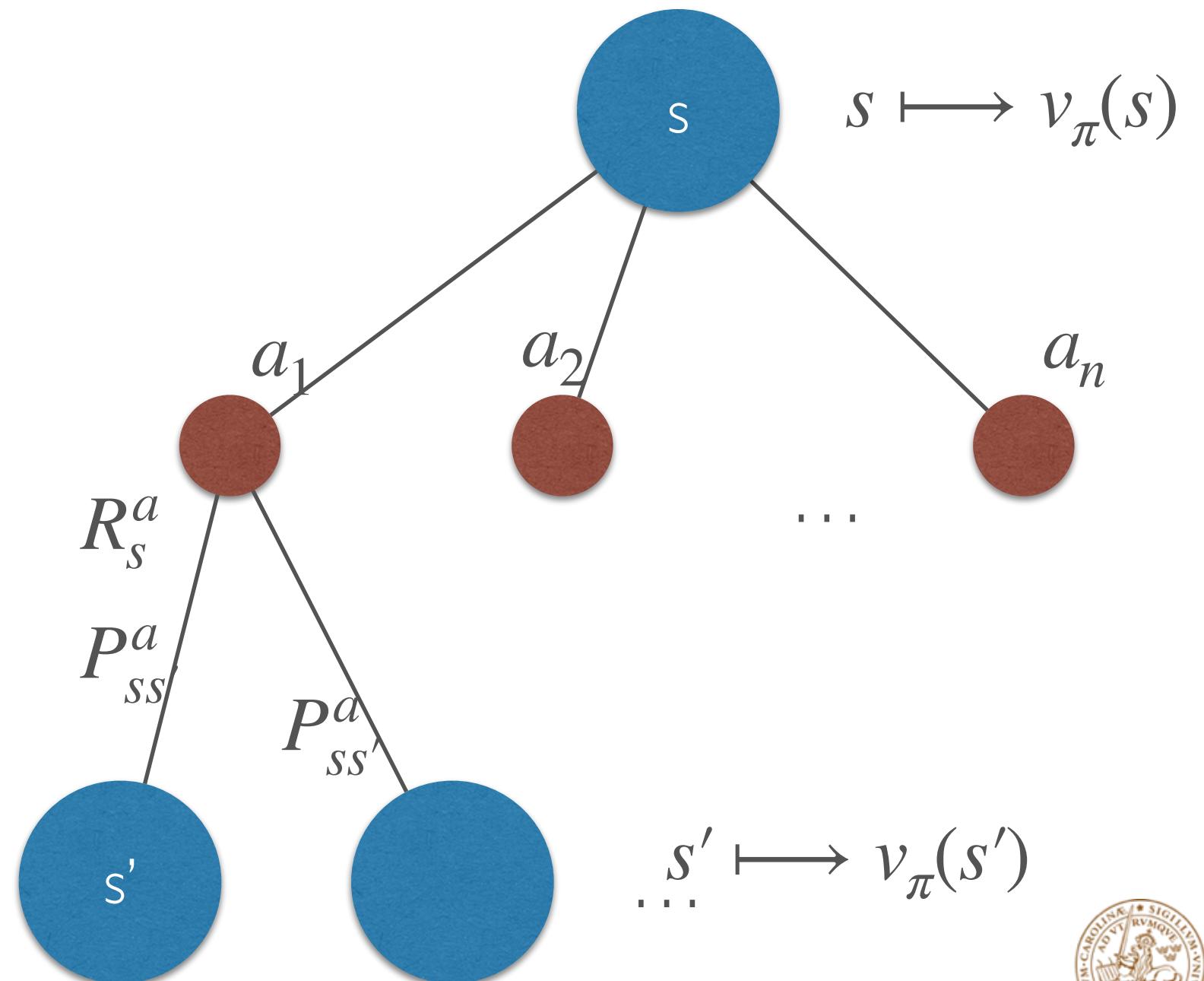
ITERATIVE POLICY EVALUATION

- Goal: evaluate a given policy π
- Solution: Iterative application of the Bellman expectation equation
- $v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_\pi$
- At each iteration $k + 1$
 - For all states $s \in S$
 - Update $v_{k+1}(s)$ from $v_k(s')$ where s is a successor state of s'

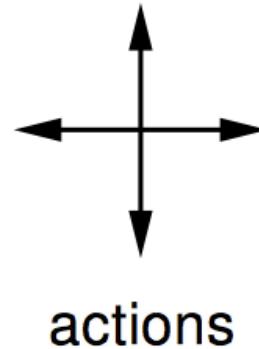


$$v_{\pi}(s) = \sum_{a \in A} \pi(a | s) \left(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_{\pi}(s') \right)$$

$$\mathbf{v}_{k+1} = \mathbf{R}^{\pi} + \gamma \mathbf{P}^{\pi} \mathbf{v}_k$$



EXAMPLE: RANDOM POLICY IN THE SMALL GRIDWORLD



	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

$$r = -1 \quad \gamma = 1$$

for all transition

- Non-terminal states 1...14
- terminal state in grey
- Actions leading out of the grid leave state unchanged
- Reward is -1 until the terminal state is reached
- Agent follows uniform random policy

$$\pi(n, \cdot) = \pi(e, \cdot) = \pi(s, \cdot) = \pi(w, \cdot)$$

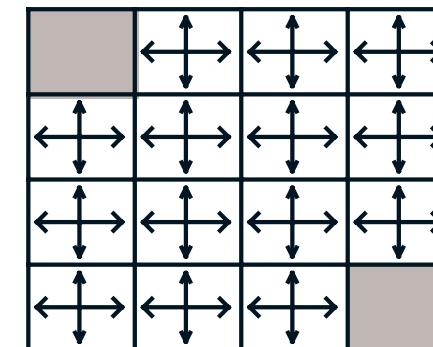
EXAMPLE: RANDOM POLICY IN THE SMALL GRIDWORLD

v_k for the
Random Policy

$k = 0$

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

Greedy Policy
w.r.t. v_k

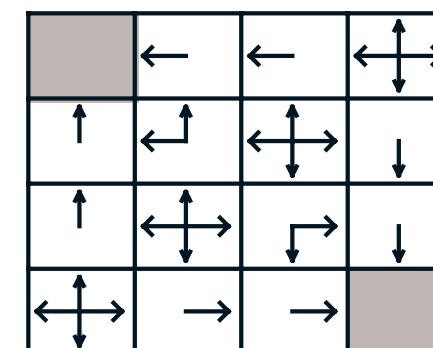
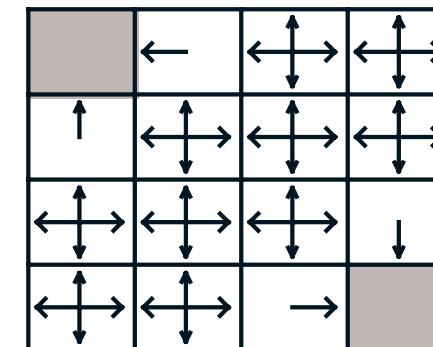


$k = 1$

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

$k = 2$

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0



EXAMPLE: RANDOM POLICY IN THE SMALL GRIDWORLD

$k = 3$

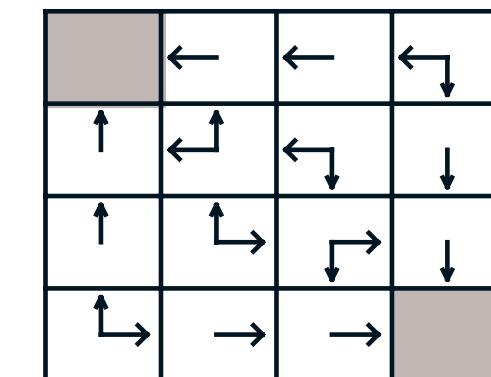
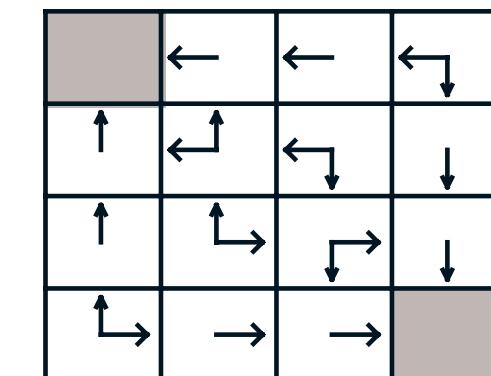
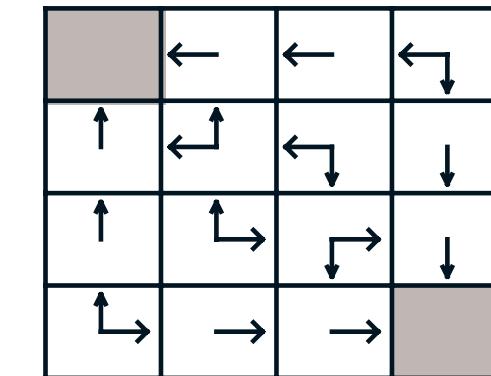
0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0

$k = 10$

0.0	-6.1	-8.4	-9.0
-6.1	-7.7	-8.4	-8.4
-8.4	-8.4	-7.7	-6.1
-9.0	-8.4	-6.1	0.0

$k = \infty$

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0



optimal
policy

HOW TO IMPROVE THE POLICY

- Given a policy.
 - Evaluate the policy

$$v_{\pi} = E \{ R_{t+1} + \gamma R_{t+2} + \dots | S_t = s \}$$

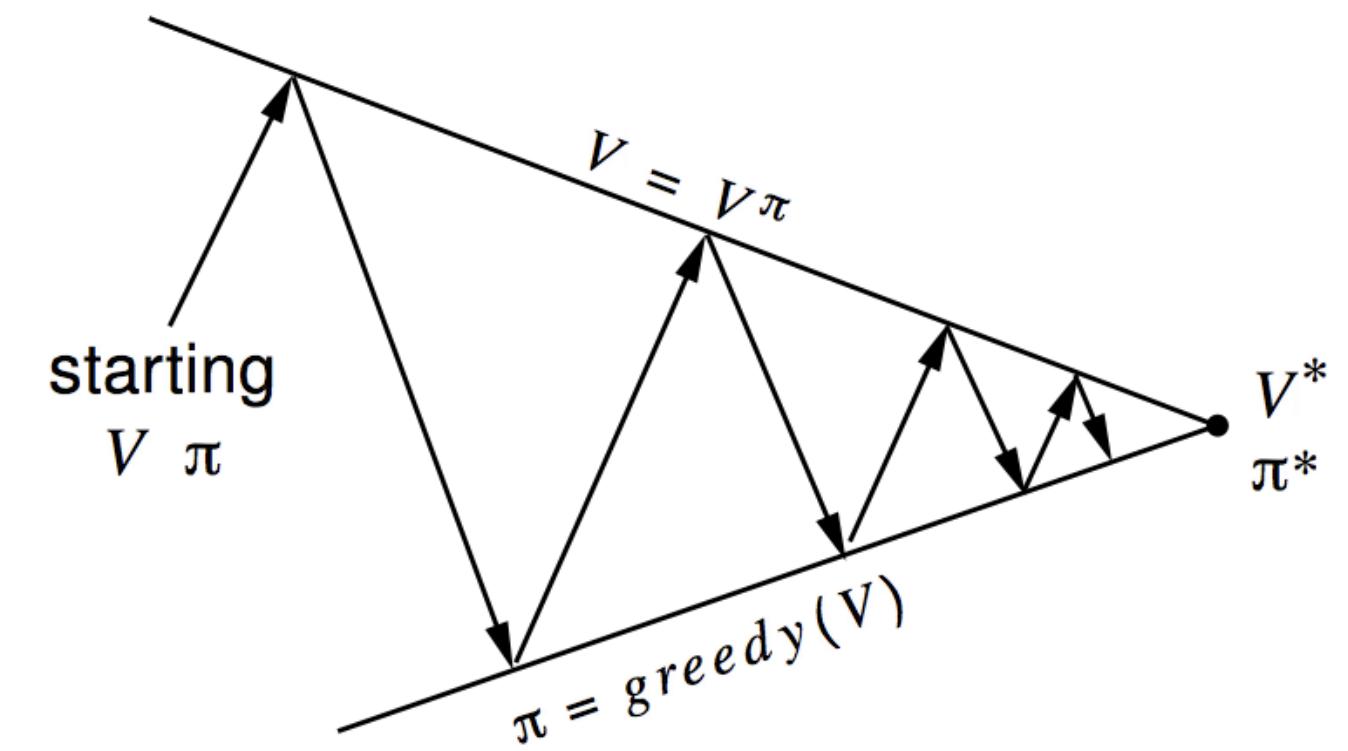
- Improve the policy by acting greedily with respect to

$$\pi' = \text{greedy}(v_{\pi})$$

- In the Gridworld the improved policy was optimal $\pi' = \pi^*$
- In general, more iterations are needed
- But this process of **policy iteration** always converges to π^*

POLICY ITERATION

- Policy Evaluation: Estimate v_π
 - Iterative policy evaluation
- Policy improvement: Generate $\pi' \geq \pi$
 - Greedy policy improvement



JACK'S CAR RENTAL

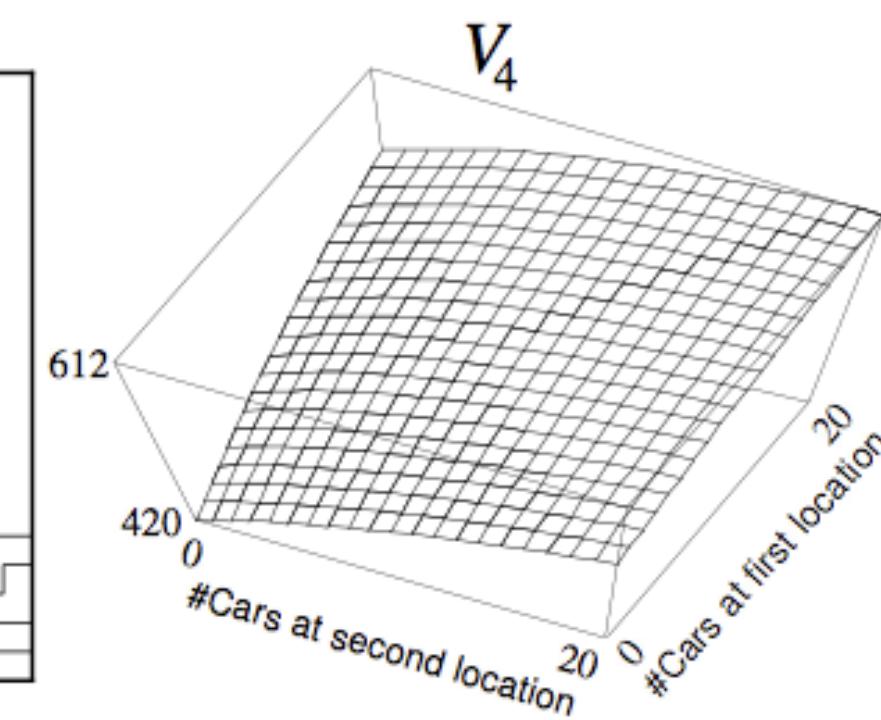
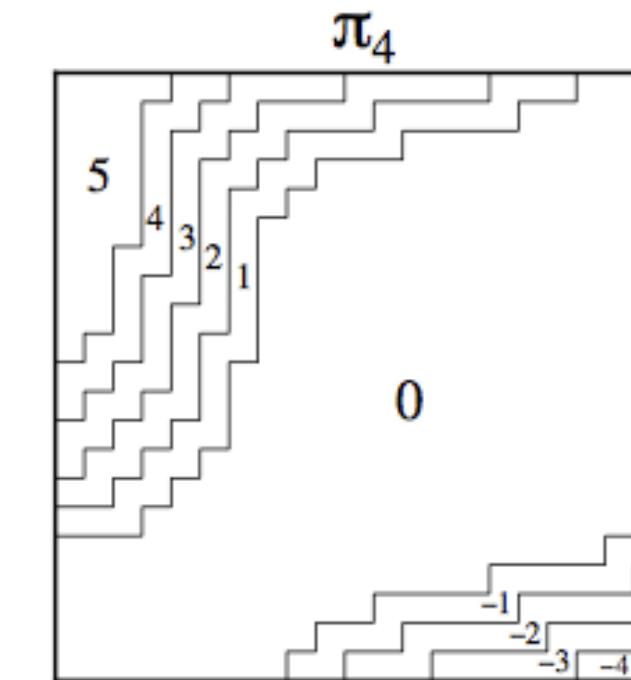
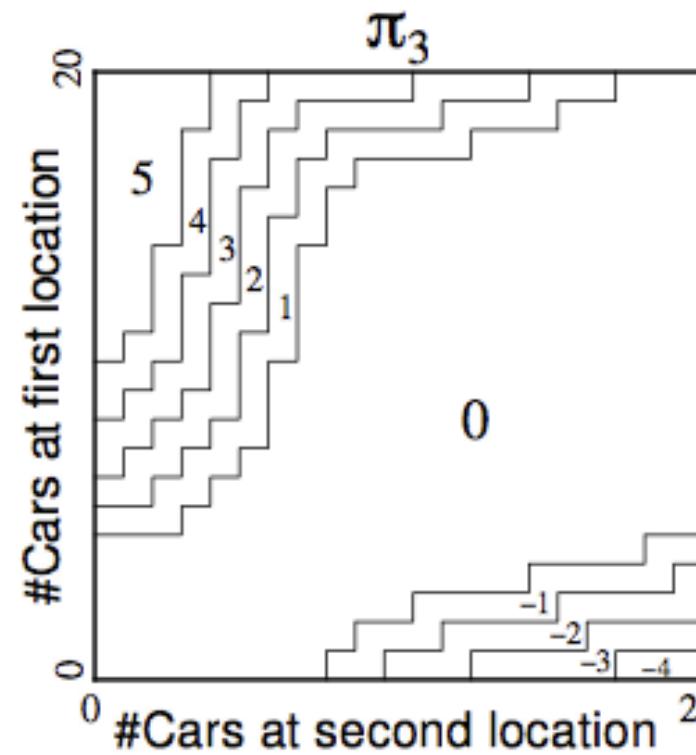
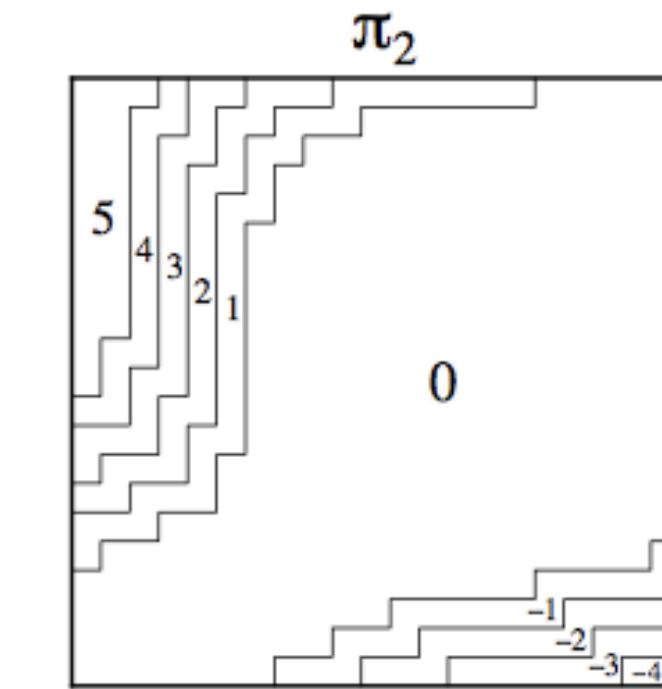
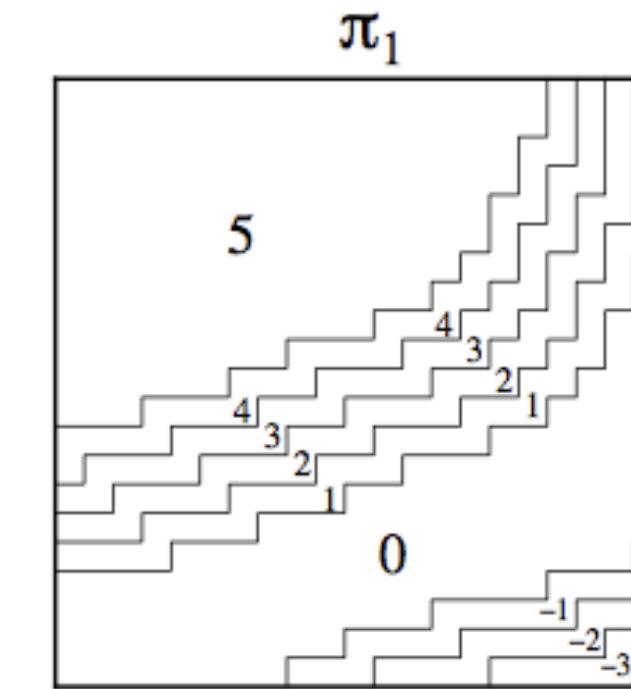
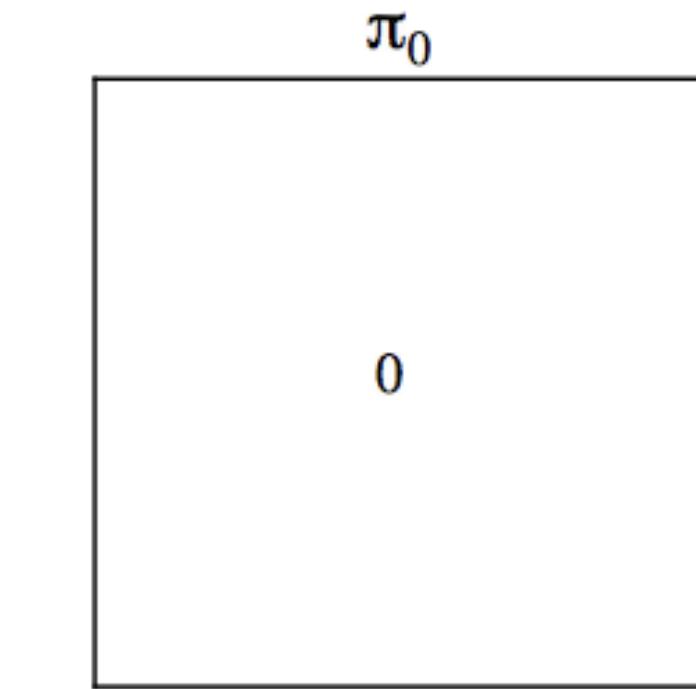


- States: Two locations, maximum 20 cars at each location
- Actions: Move up to 5 cars between locations overnight
- Reward: 10€ for each car rented if available
- Transitions: Cars returned and requested randomly
 - Poisson distribution, n returns/requests with probability $\frac{\lambda^n}{n!} e^{-\lambda}$
 - 1st location: average requests=3, average returns=3
 - 2nd location: average requests=4, average returns =2



LUND
UNIVERSITY

POLICY ITERATION IN JACK'S CAR RENTAL



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
- Policy Iteration for finding the optimal policy.
 - Very simple approach, but still finds π^*
 - the MDP needs to be known completely



VOLUNTARY EXERCISES

- Gridworld and Jack's car rental are exercises in the Book by Sutton and Barto, Sect 3 and beginning of Sect. 4
 - Correct results are in these slides.



LUND
UNIVERSITY