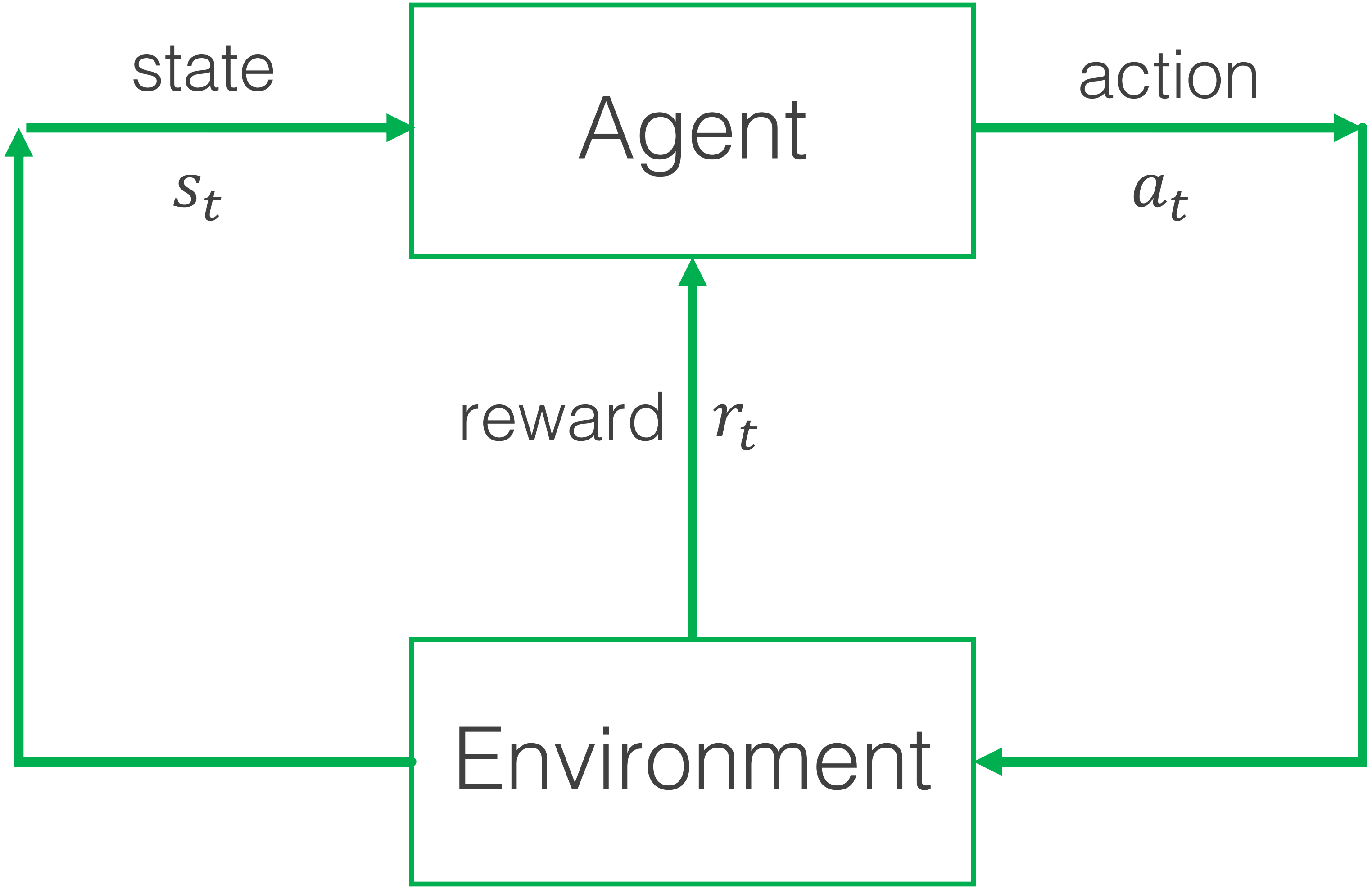
Reinforcement Learning II

Agent-environment Interaction



Agent at each step t…

Encounters state, s_t
Executes action a_t
Receives scalar reward, r_{t+1}

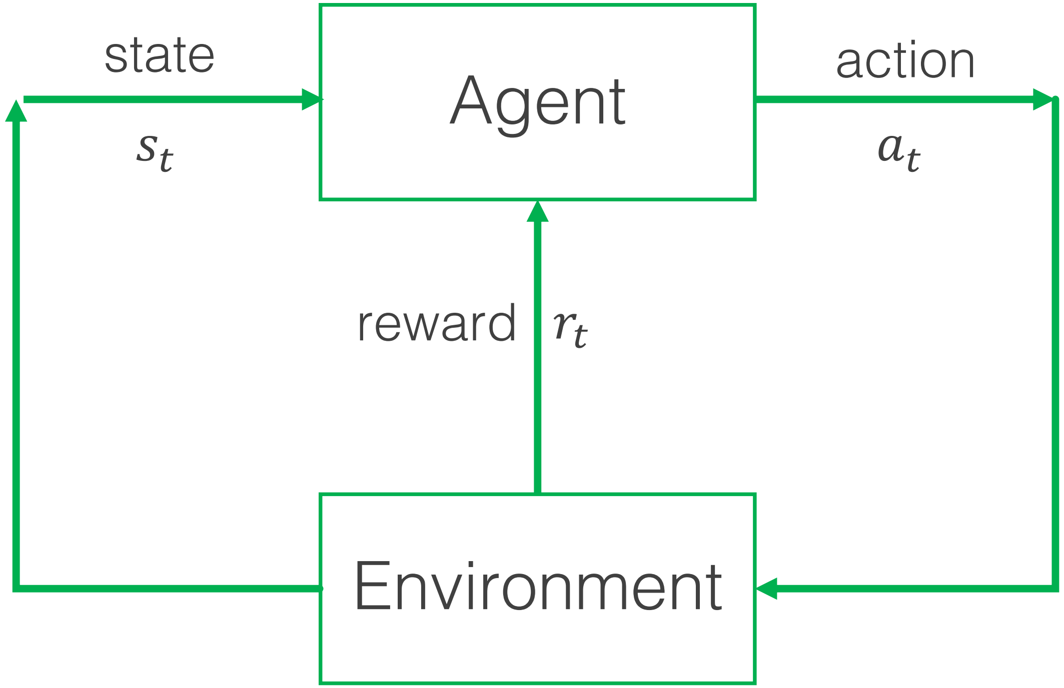
Environment at each step t…

Receives action a_t
Transitions to state, s_{t+1}
Emits scalar reward, r_{t+1}

Actions: choices made by the agent
States: basis on which choices are made
Rewards: define the agent's goals

David Silver, 2015

Reinforcement Learning Components

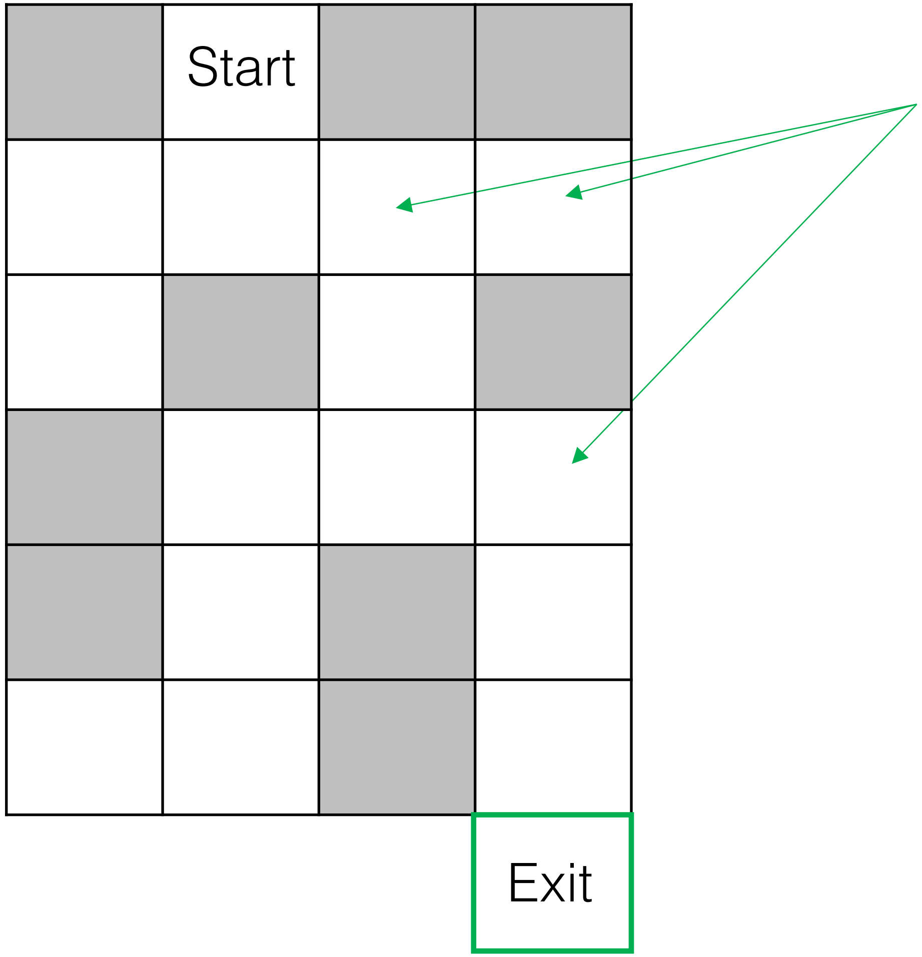


Policy (agent behavior), $\pi(s)$

Reward function (the goal), r_t

Value functions (expected returns),
$v(s)$ State value
$q(s,a)$ Action value

Maze Example: Policy, Value, and Reward



Each location in the maze represents a **state**

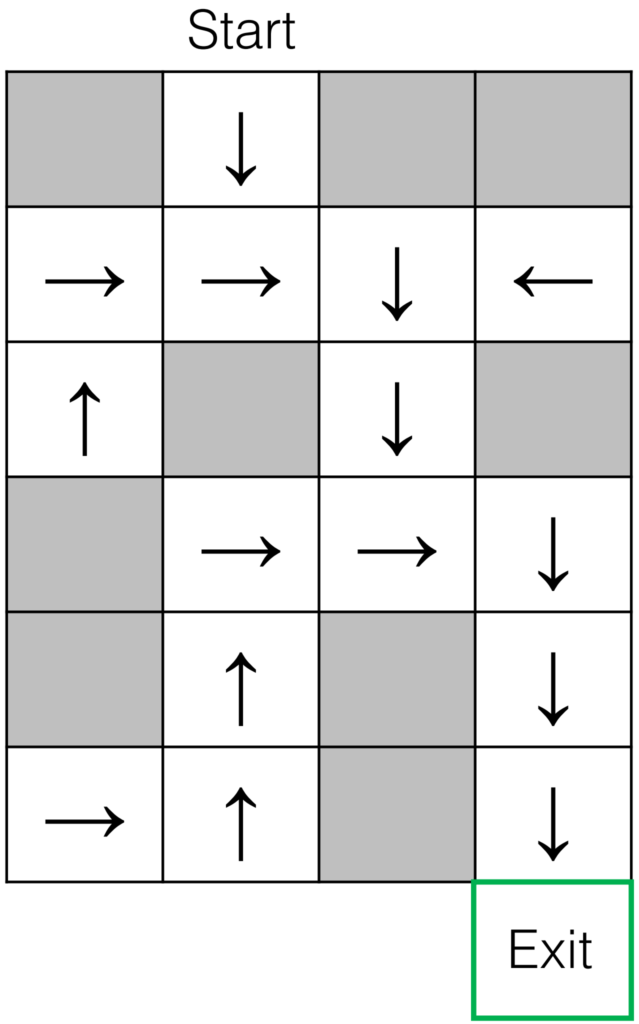
The **reward** is -1 for each step the agent is in the maze

Available **actions**: move ↑,↓,←,→ (as long as that path is not blocked)

Adapted from David Silver, 2015

Policy $\pi(s)$

(which actions to take in each state)

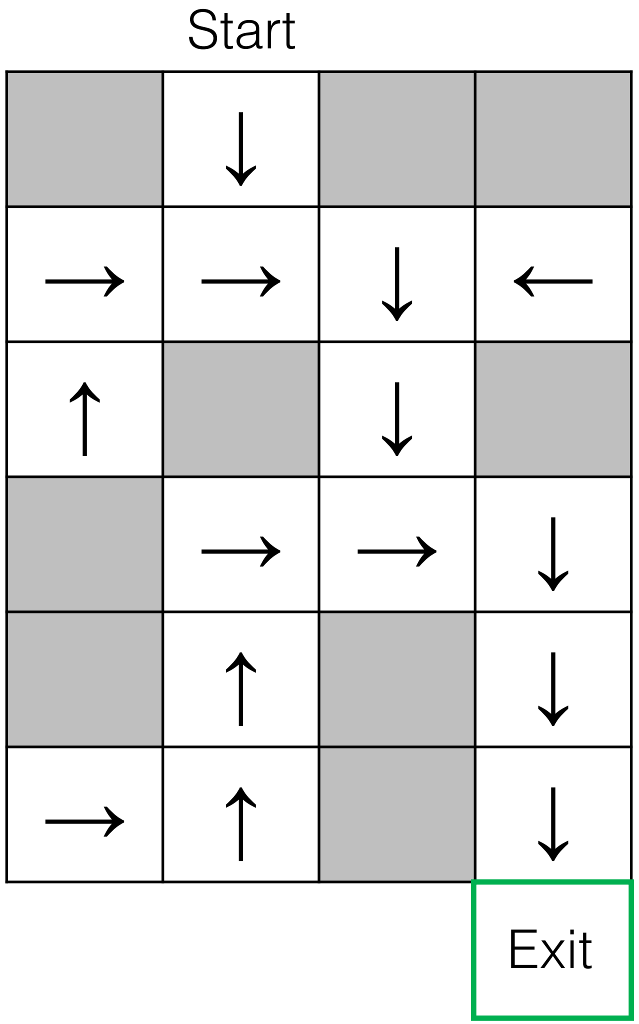


▨	↓	▨	▨
→	→	↓	←
↑	▨	↓	▨
▨	→	→	↓
▨	↑	▨	↓
→	↑	▨	↓

Adapted from David Silver, 2015

Policy $\pi(s)$

(which actions to take in each state)



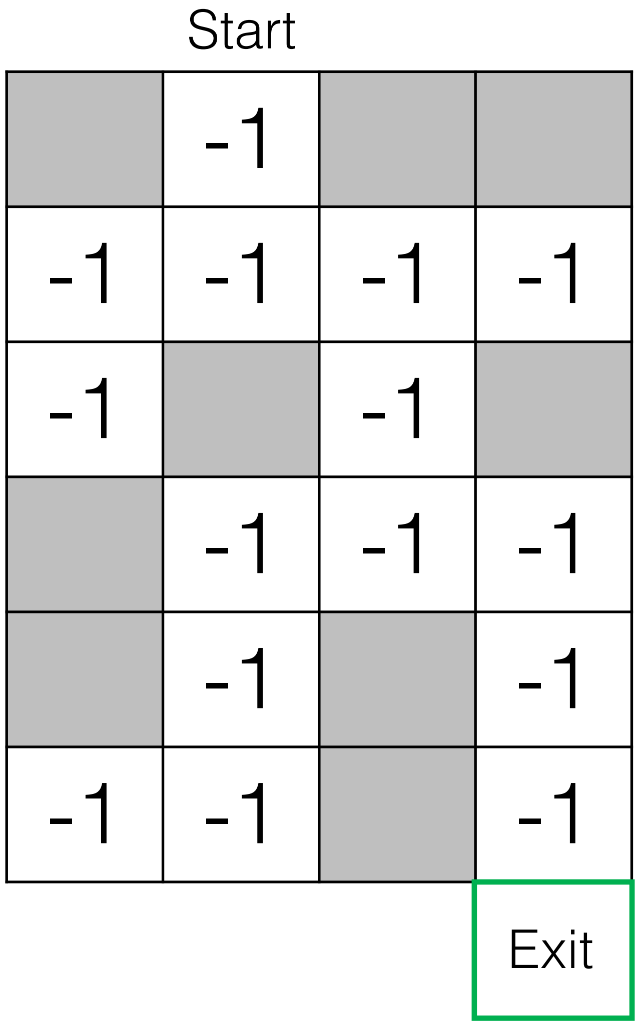
Start

▨	↓	▨	▨
→	→	↓	←
↑	▨	↓	▨
▨	→	→	↓
▨	↑	▨	↓
→	↑	▨	↓

Exit

Reward r_t

(rewards are received after actions are taken)



Start

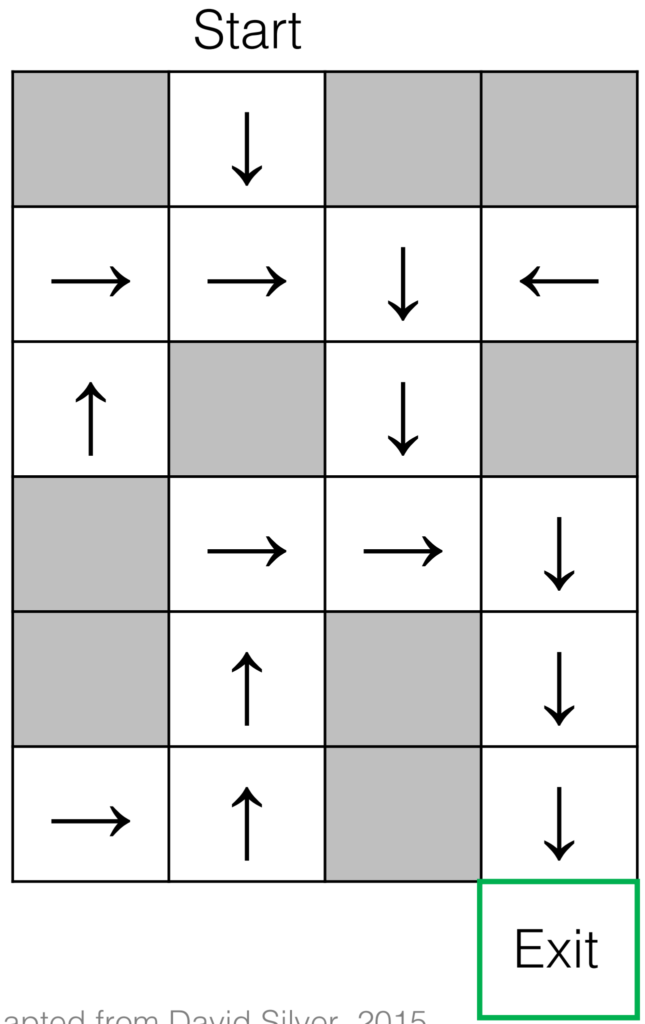
▨	-1	▨	▨
-1	-1	-1	-1
-1	▨	-1	▨
▨	-1	-1	-1
▨	-1	▨	-1
-1	-1	▨	-1

Exit

Adapted from David Silver, 2015

Policy $\pi(s)$

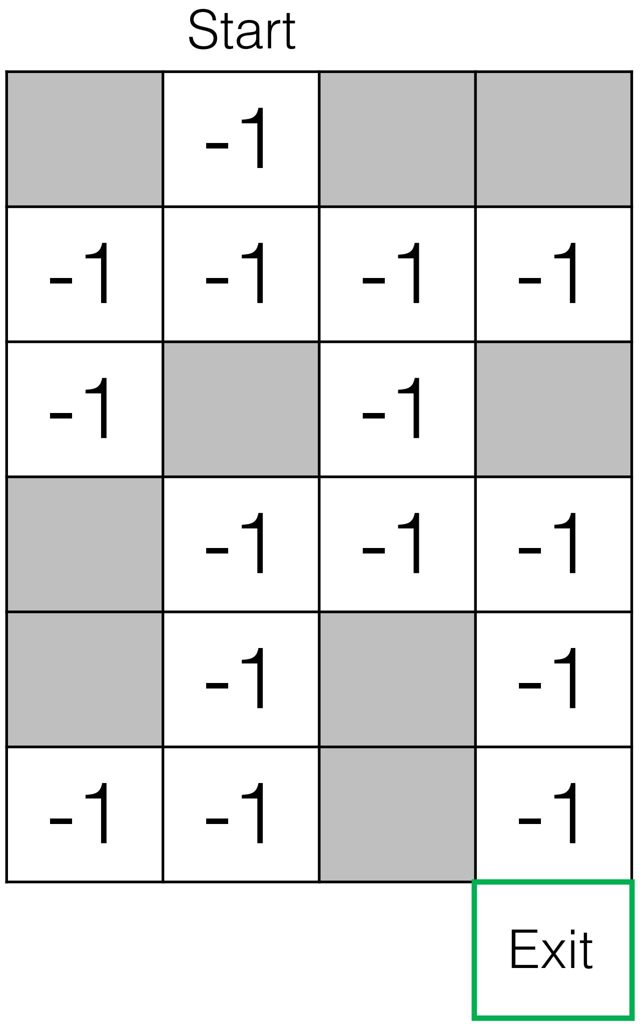
(which actions to take in each state)



	↓		
→	→	↓	←
↑		↓	
	→	→	↓
	↑		↓
→	↑		↓

Reward r_t

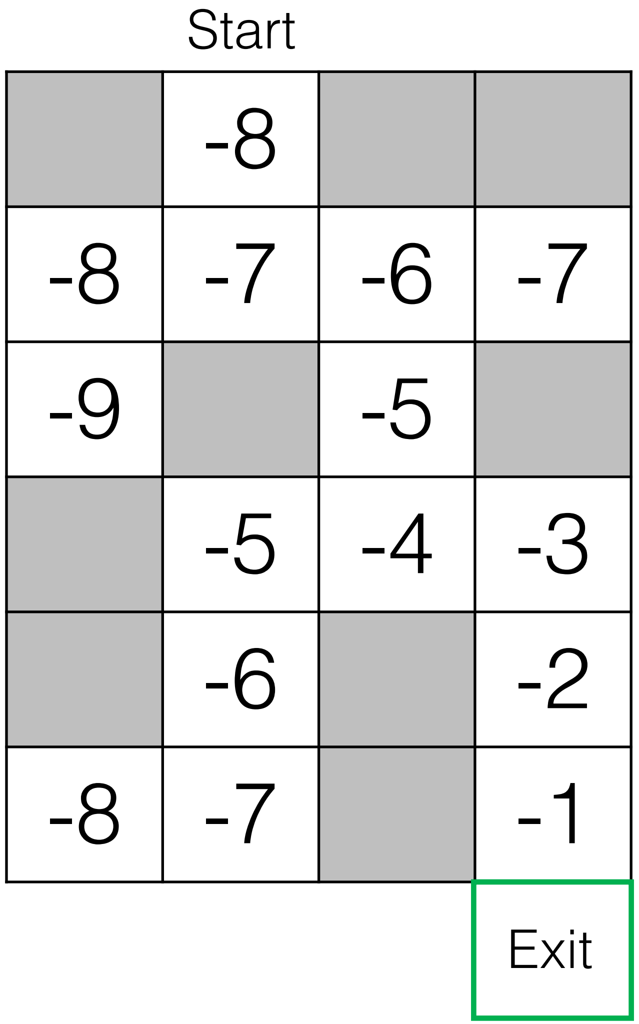
(rewards are received after actions are taken)



	-1		
-1	-1	-1	-1
-1		-1	
	-1	-1	-1
	-1		-1
-1	-1		-1

State Value $v_\pi(s)$

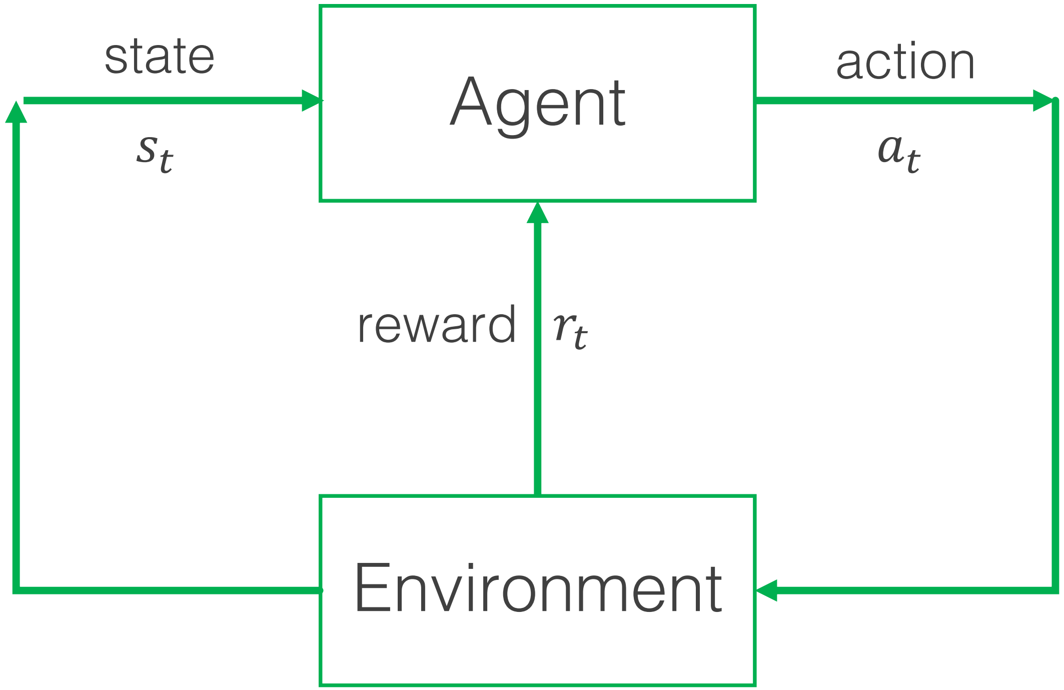
(expected cumulative rewards starting from current state **if** we follow the policy)



	-8		
-8	-7	-6	-7
-9		-5	
	-5	-4	-3
	-6		-2
-8	-7		-1

Adapted from David Silver, 2015

Policy



Policy, $\pi(s)$

- Selects an action to choose based on the state
- Determines an agent's "behavior"

Deterministic policy:

$$a = \pi(s)$$

Stochastic policy:

$$\pi(a|s) = P(a_t = a|s_t = s)$$

Helps us "explore" the state space

RL tries to learn the "best" policy

Goals and rewards

Rewards are the **only way** of communicating RL goals

Ex 1: Robot learning a maze

- 0 until it escapes, then +1 when it does
- -1 until it escapes (encourages it to escape quickly)

Ex 2: Robot collecting empty soda cans

- +1 for each empty soda can
- Negative rewards for bumping into things

Chess: what if we set +1 for capturing a piece?
(it may not win the game and still maximize rewards)

What you want achieved not **how**

Returns / cumulative reward

Episodic tasks (finite number, T, of steps, then reset)

$$G_t = r_{t+1} + r_{t+2} + \cdots + r_T$$

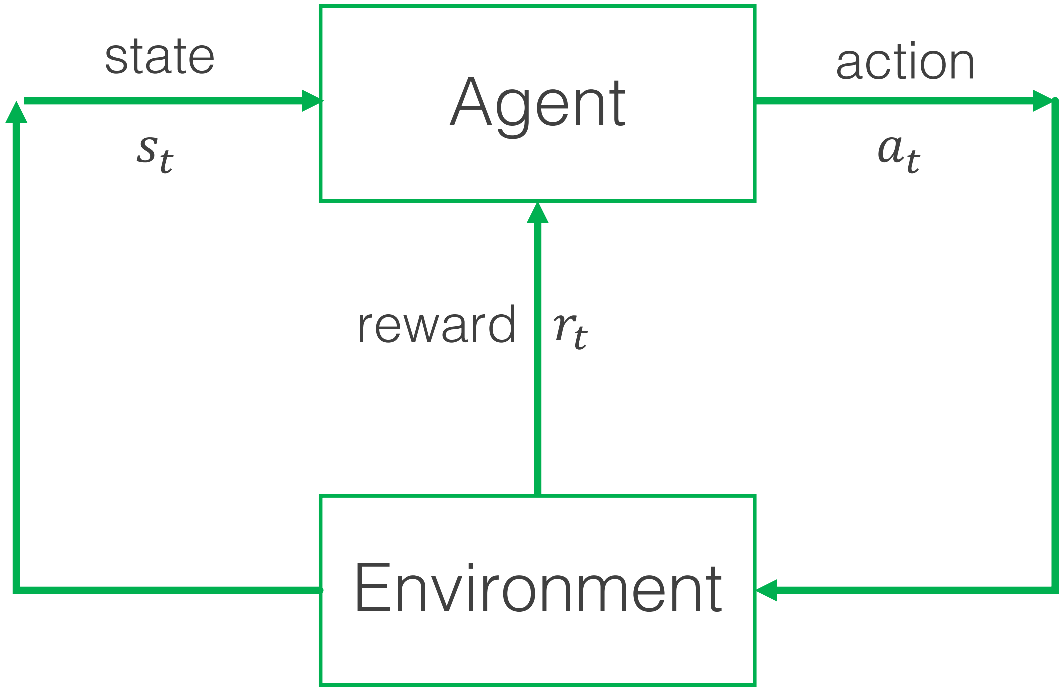
Continuing tasks with discounting ($T \rightarrow \infty$)

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

where $\mathbf{0} \leq \gamma \leq \mathbf{1}$ is the discount rate

This makes the agent care more about immediate rewards

Value functions



State Value function, $v_\pi(s)$

- How "good" is it to be in a state, s_t then follow policy π to choose actions
- Total expected rewards

$$v_\pi(s) = E_\pi[G_t | s_t = s]$$

Action Value function, $q_\pi(s, a)$

- How "good" is it to be in a state, s, take action a, then follow policy π to choose actions
- Total expected rewards

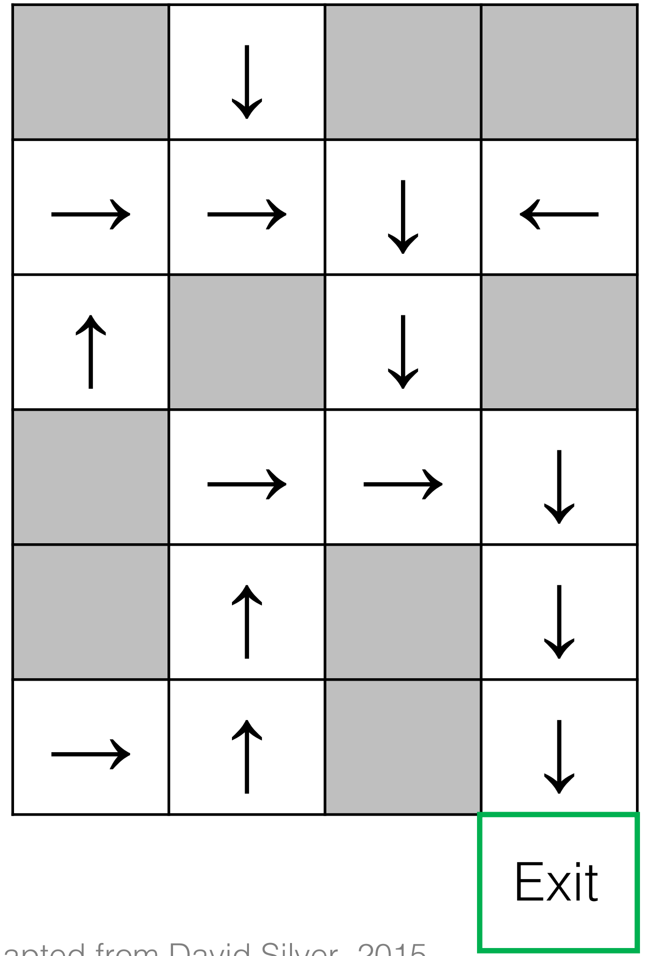
$$q_\pi(s, a) = E_\pi[G_t | s_t = s, a_t = a]$$

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

Policy $\pi(s)$

(which actions to take in each state)

Start



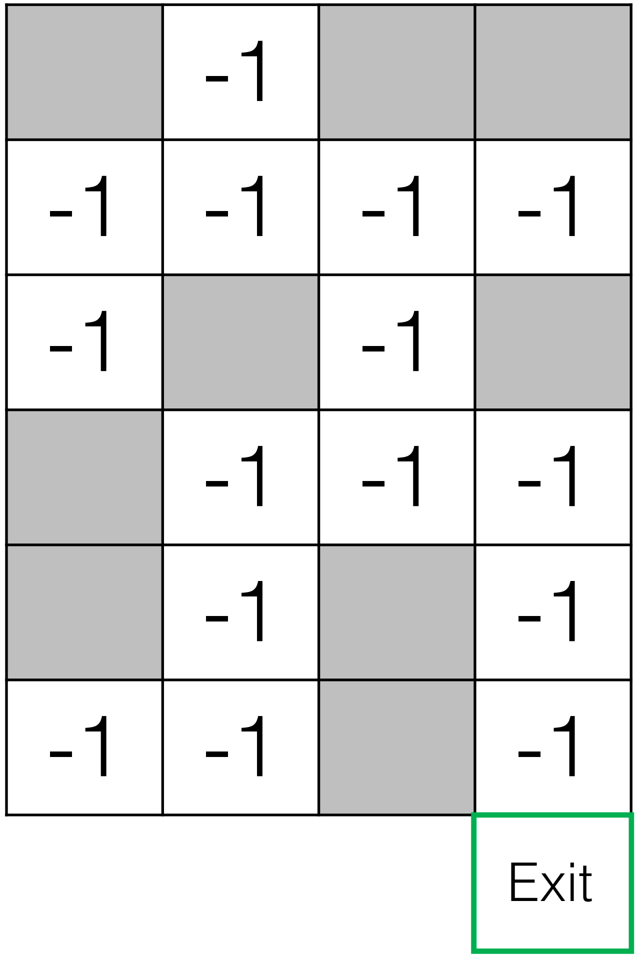
	↓		
→	→	↓	←
↑		↓	
	→	→	↓
	↑		↓
→	↑		↓

Exit

Reward r_t

(rewards are received after actions are taken)

Start



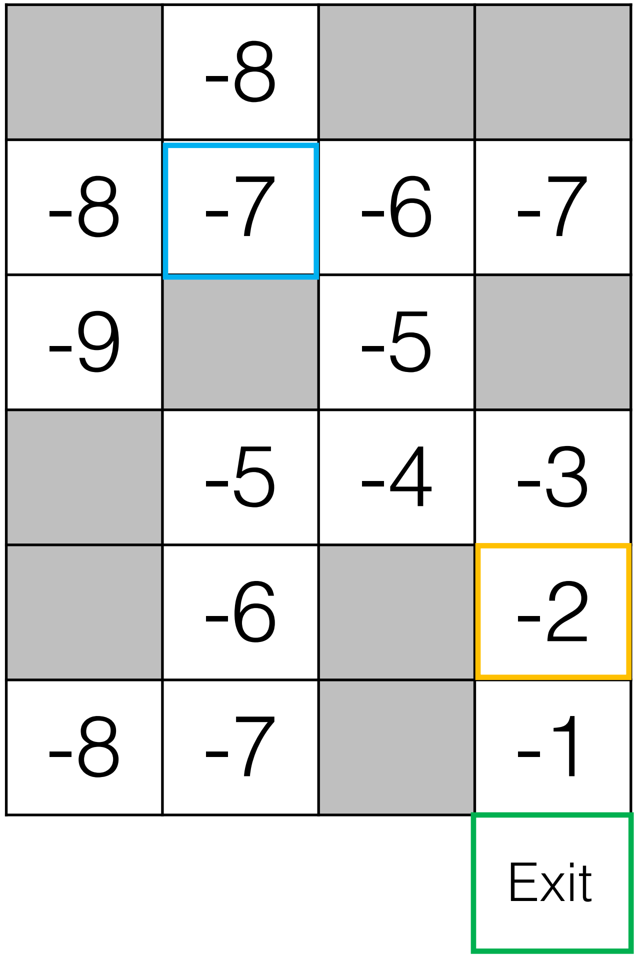
	-1		
-1	-1	-1	-1
-1		-1	
	-1	-1	-1
	-1		-1
-1	-1		-1

Exit

State Value $v_\pi(s)$

(expected cumulative rewards starting from current state **if** we follow the policy)

Start

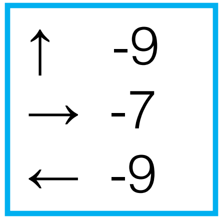


	-8		
-8	-7	-6	-7
-9		-5	
	-5	-4	-3
	-6		-2
-8	-7		-1

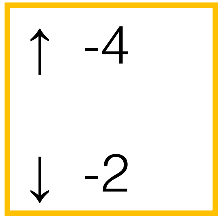
Exit

Action Value $q_\pi(s,a)$

(expected cumulative rewards starting from current state **if** we take action a then follow the policy)



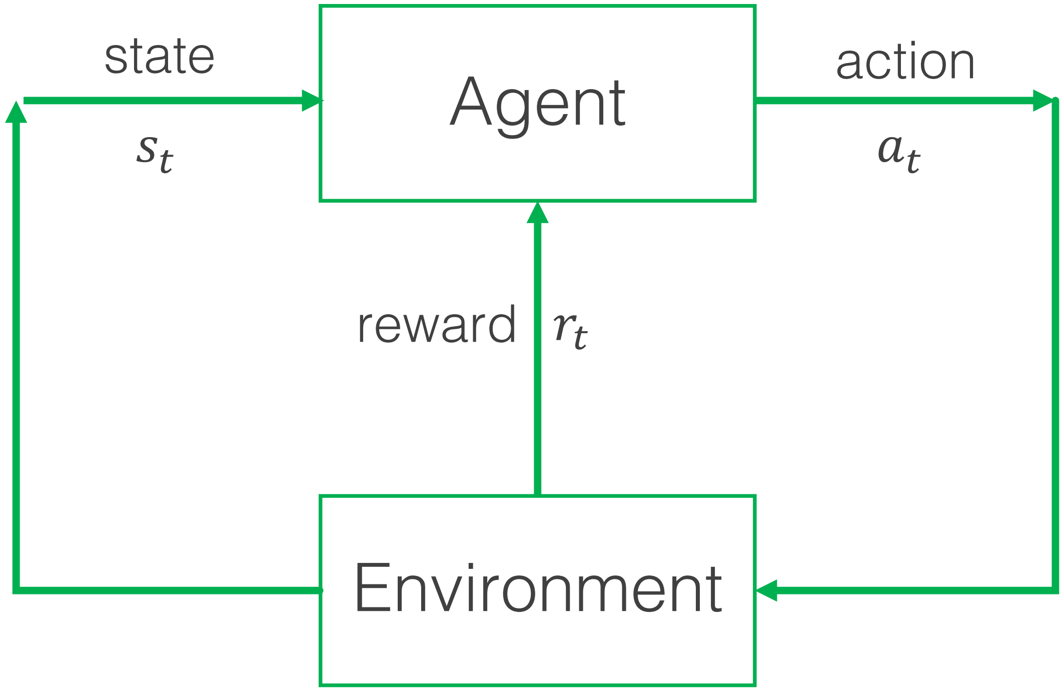
↑ -9
→ -7
← -9



↑ -4
↓ -2

Adapted from David Silver, 2015

Model



Model (of the environment)

Transitions: predicts what state the environment will transition to next

$$P_{ss'}^{a} = P(s_{t+1} = s' | s_t = s, a_t = a)$$

Rewards: predicts the next reward given an action

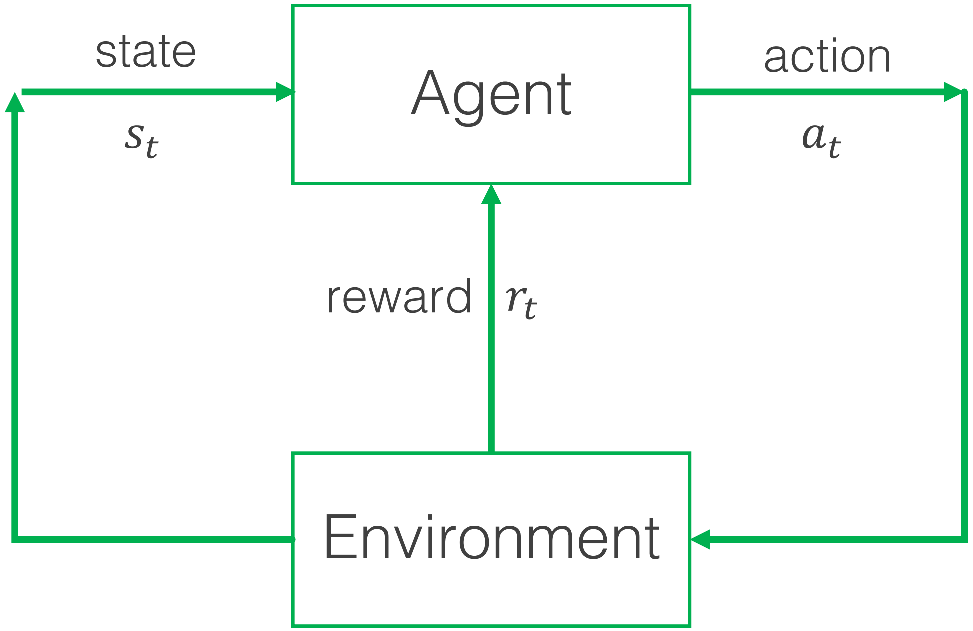
$$R_s^a = E[r_{t+1} | s_t = s, a_t = a]$$

“Planning” is the process of using a model to create or improve a policy

We don’t always have a full model of the environment

Model-based RL uses a model
Model-free RL does not use a model

Reinforcement Learning Components



Policy (determines agent behavior), $\pi(s)$

- Determines action given current state
- Agent's way of behaving at a given time

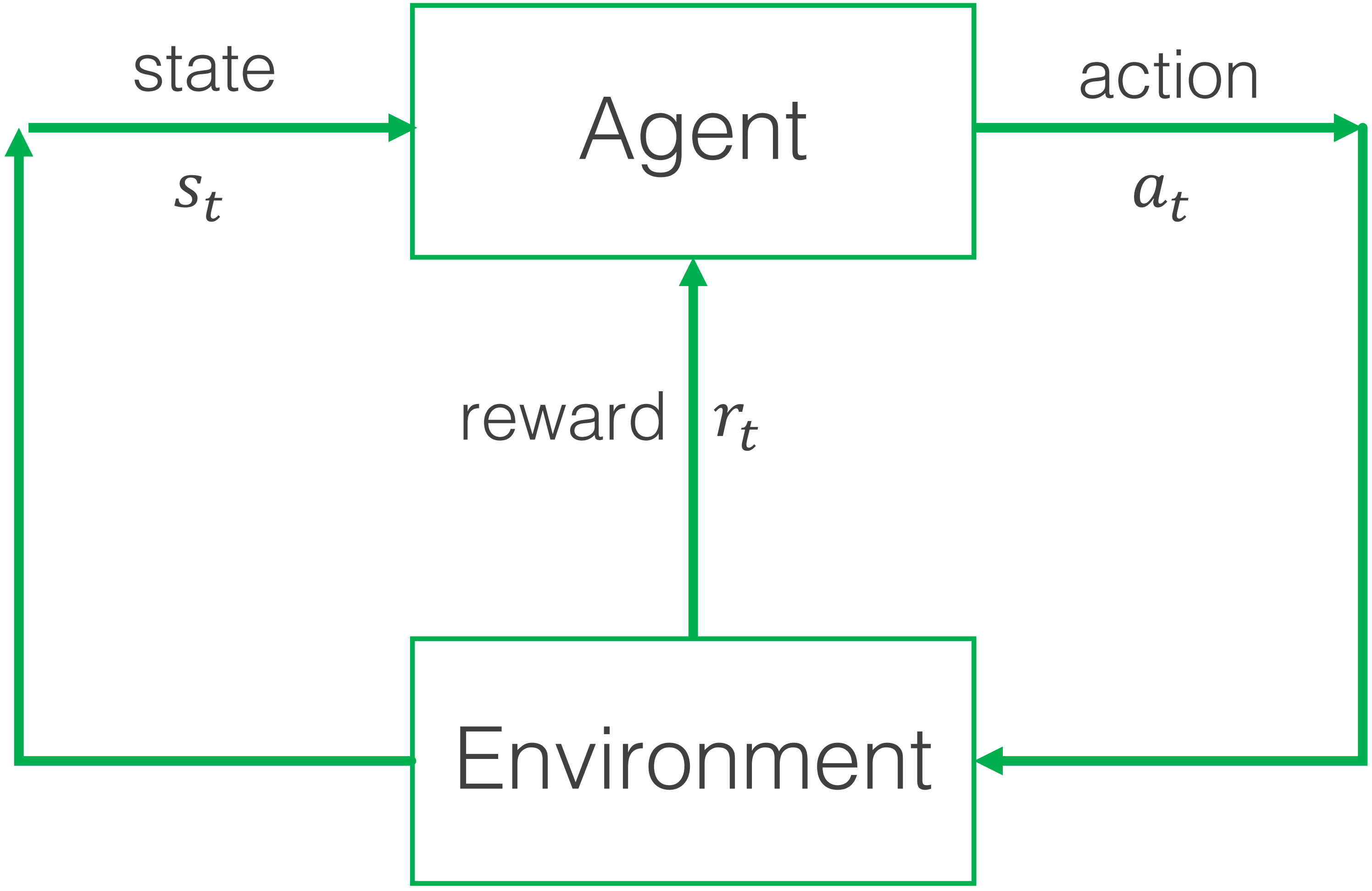
Reward function (sets the goal), r_t

- Maps state of the environment to a reward that describes the state desirability
- Objective is to **maximize total rewards**

Value (estimates expected returns), $v(s), q(s,a)$

- Expected returns from a state and following a specific policy
- How "good" is each state

Environment



state

s_t

Agent

action

a_t

reward

r_t

Environment

Markov Decision Process

(assumed form for most RL problems)

Goal

Maximize returns (expected rewards)

Find the best policy to guide our actions in an environment

Here, environment is modeled as a Markov Decision Process

Reinforcement Learning Roadmap

Knowledge of **Environment**

Perfect knowledge
Known Markov Decision Process

No knowledge
Must learn from experience

Dynamic Programming

What’s a Markov Decision Process?
How do we find optimal policies?

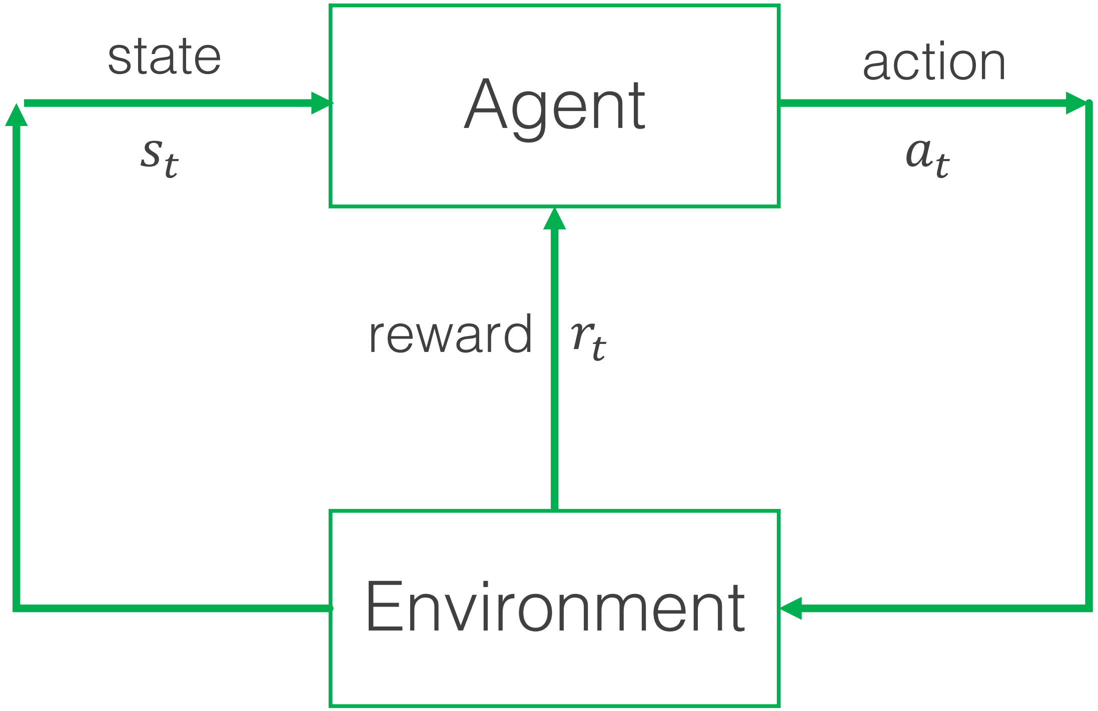
Monte Carlo Control

How do we estimate our value functions?
How do we use the value functions to choose actions?

Towards Markov Decisions Processes (MDPs)

History

The record of all that has happened in this system



Step 0: s_0, a_0

Step 1: r_1, s_1, a_1

Step 2: r_2, s_2, a_2

$\vdots$

Step T: r_t, s_t, a_t

History at time t : $H_t = \{s_t, a_t, r_{t-1}, s_{t-1}, a_{t-1}, \ldots r_1, s_1, a_1, s_0, a_0\}$

Markov property

Instead of needing the full history:

$$H_t = \{s_t, a_t, r_{t-1}, s_{t-1}, a_{t-1}, \dots r_1, s_1, a_1, s_0, a_0\}$$

We can summarize everything in the current state

$$H_t = \{s_t, a_t\}$$

The future is independent of the past given the present

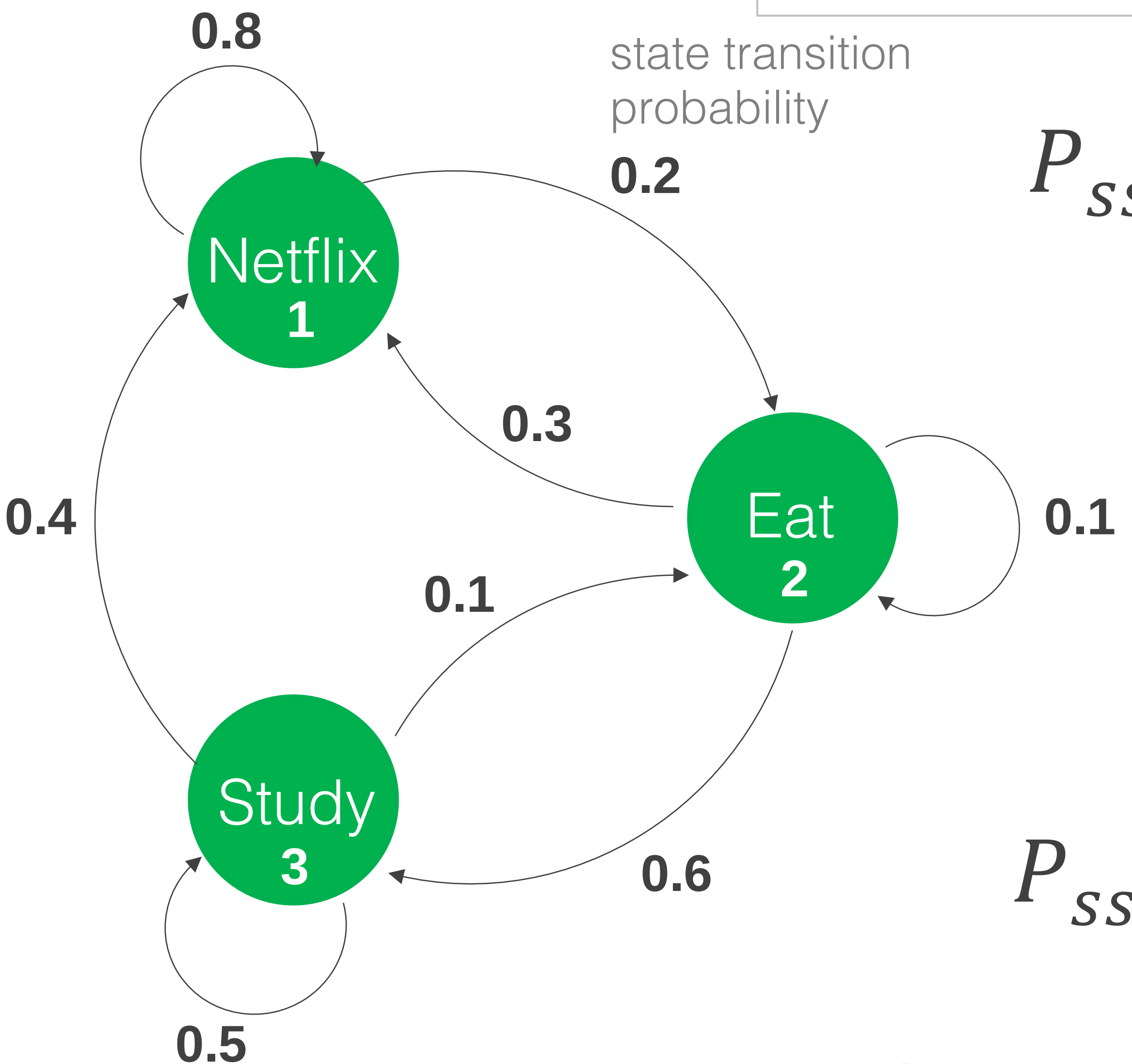
Another way of saying this is:

$$P(s_{t+1}|s_t) = P(s_{t+1}|s_t, s_{t-1}, \dots, s_1, s_0)$$

Markov Chains

Example: student life

Two components: $\{S, P\}$
State space, S
Transition matrix, P



State-space representation

State transition probabilities

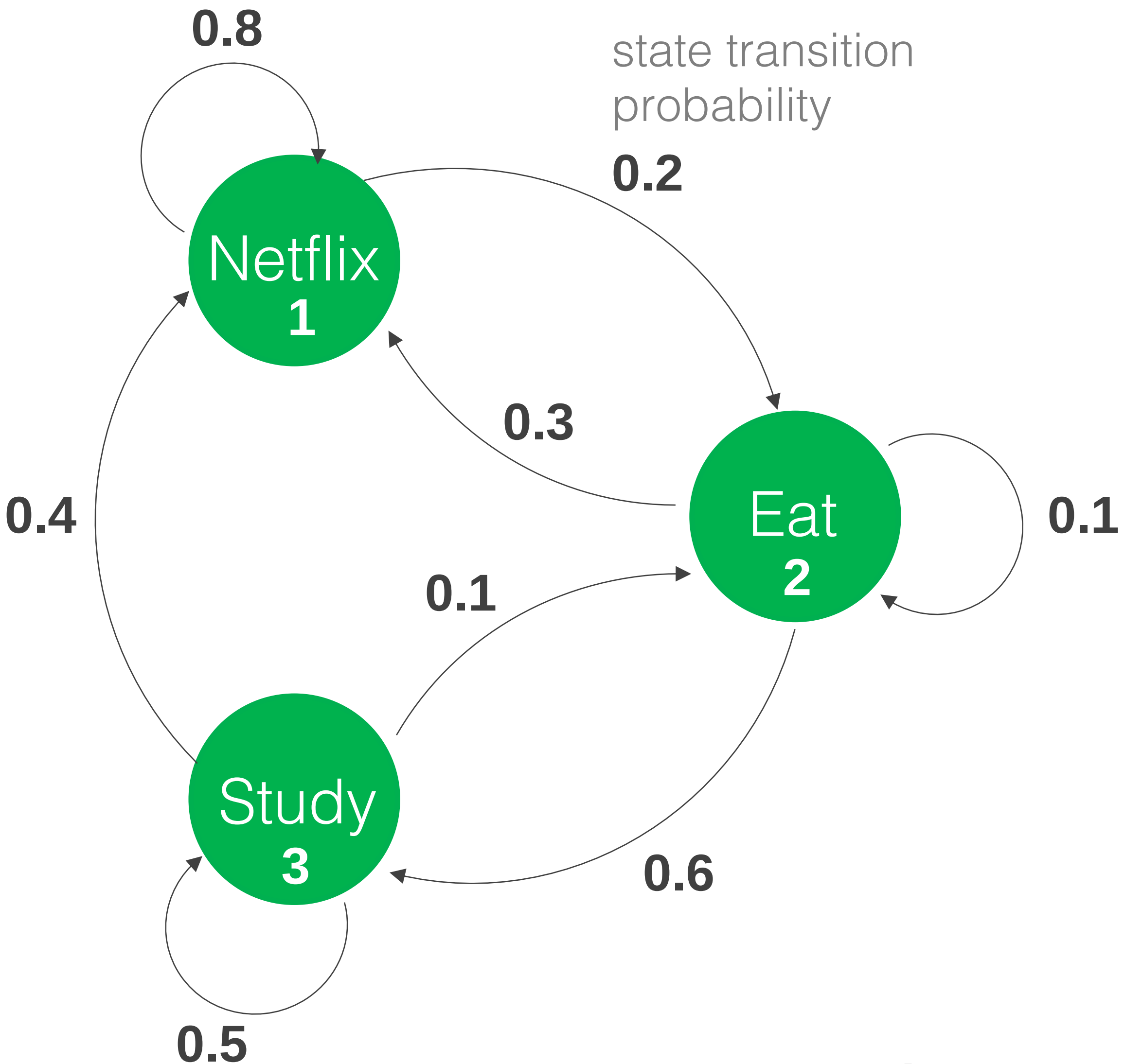
$$P_{ss'} = \begin{array}{cc} & \text{To state} \\ & \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \text{From state} \begin{array}{c} 1 \\ 2 \\ 3 \end{array} & \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} \end{array}$$

Transitions out of each state sum to 1

$$P_{ss'} = \begin{array}{cc} & \text{To state} \\ & \begin{array}{ccc} \text{Netflix} & \text{Eat} & \text{Study} \end{array} \\ \text{From state} \begin{array}{c} \text{Netflix} \\ \text{Eat} \\ \text{Study} \end{array} & \begin{bmatrix} 0.8 & 0.2 & 0 \\ 0.3 & 0.1 & 0.6 \\ 0.4 & 0.1 & 0.5 \end{bmatrix} \end{array}$$

Markov Chains

Example: student life



State-space representation

If we start in state 1, what's the probability we'll be in each state after one step?

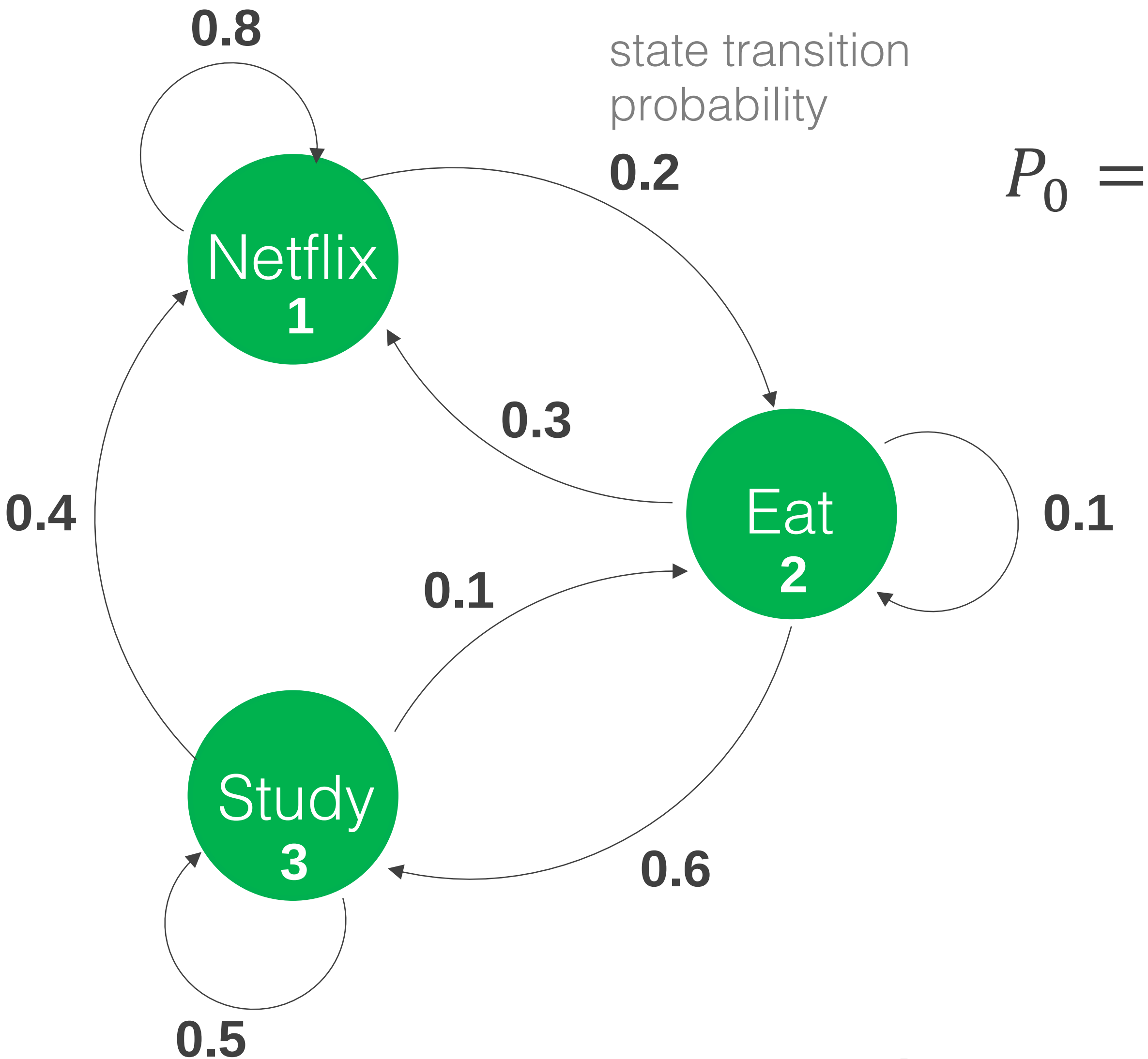
$$
\begin{array}{ccccc}
 & & 1 & 2 & 3 \\
P_1 & = & [0.8 & 0.2 & 0]
\end{array}
$$

This is the first row of the state transition probability matrix

$$
P_{ss'} = \begin{array}{c|c|ccc}
 & & & \text{To state} & \\
 & & \text{Netflix} & \text{Eat} & \text{Study} \\
\text{From state} & \text{Netflix} & 0.8 & 0.2 & 0 \\
 & \text{Eat} & 0.3 & 0.1 & 0.6 \\
 & \text{Study} & 0.4 & 0.1 & 0.5
\end{array}
$$

Markov Chains

Example: student life



State-space representation

If we started in state 1, we can calculate the probabilities of being in each state at step 1 as:

$$P_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}^T \qquad P_1 = P_0 P_{ss'}$$

$$P_1 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.8 & 0.2 & 0 \\ 0.3 & 0.1 & 0.6 \\ 0.4 & 0.1 & 0.5 \end{bmatrix}$$

$$P_1 = \begin{bmatrix} 0.8 & 0.2 & 0 \end{bmatrix}$$

To state

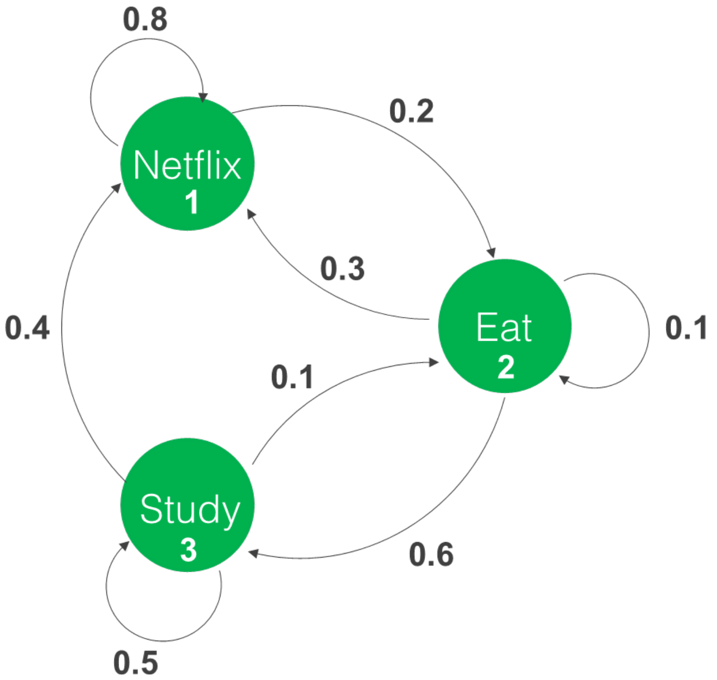
From state

$$P_{ss'} = \begin{array}{c|ccc} & \text{Netflix} & \text{Eat} & \text{Study} \\ \hline \text{Netflix} & 0.8 & 0.2 & 0 \\ \text{Eat} & 0.3 & 0.1 & 0.6 \\ \text{Study} & 0.4 & 0.1 & 0.5 \end{array}$$

① $P_1 = P_0 P_{ss'}$

$$P_1 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.8 & 0.2 & 0 \\ 0.3 & 0.1 & 0.6 \\ 0.4 & 0.1 & 0.5 \end{bmatrix}$$

$$P_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}^T$$



$$P_1 = \begin{bmatrix} 0.8 & 0.2 & 0 \end{bmatrix}$$

② $P_2 = P_1 P_{ss'} = P_0 P_{ss'} P_{ss'} = P_0 P_{ss'}^2$

$$P_2 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.8 & 0.2 & 0 \\ 0.3 & 0.1 & 0.6 \\ 0.4 & 0.1 & 0.5 \end{bmatrix} \begin{bmatrix} 0.8 & 0.2 & 0 \\ 0.3 & 0.1 & 0.6 \\ 0.4 & 0.1 & 0.5 \end{bmatrix}$$

$$P_2 = \begin{bmatrix} 0.7 & 0.18 & 0.12 \end{bmatrix}$$

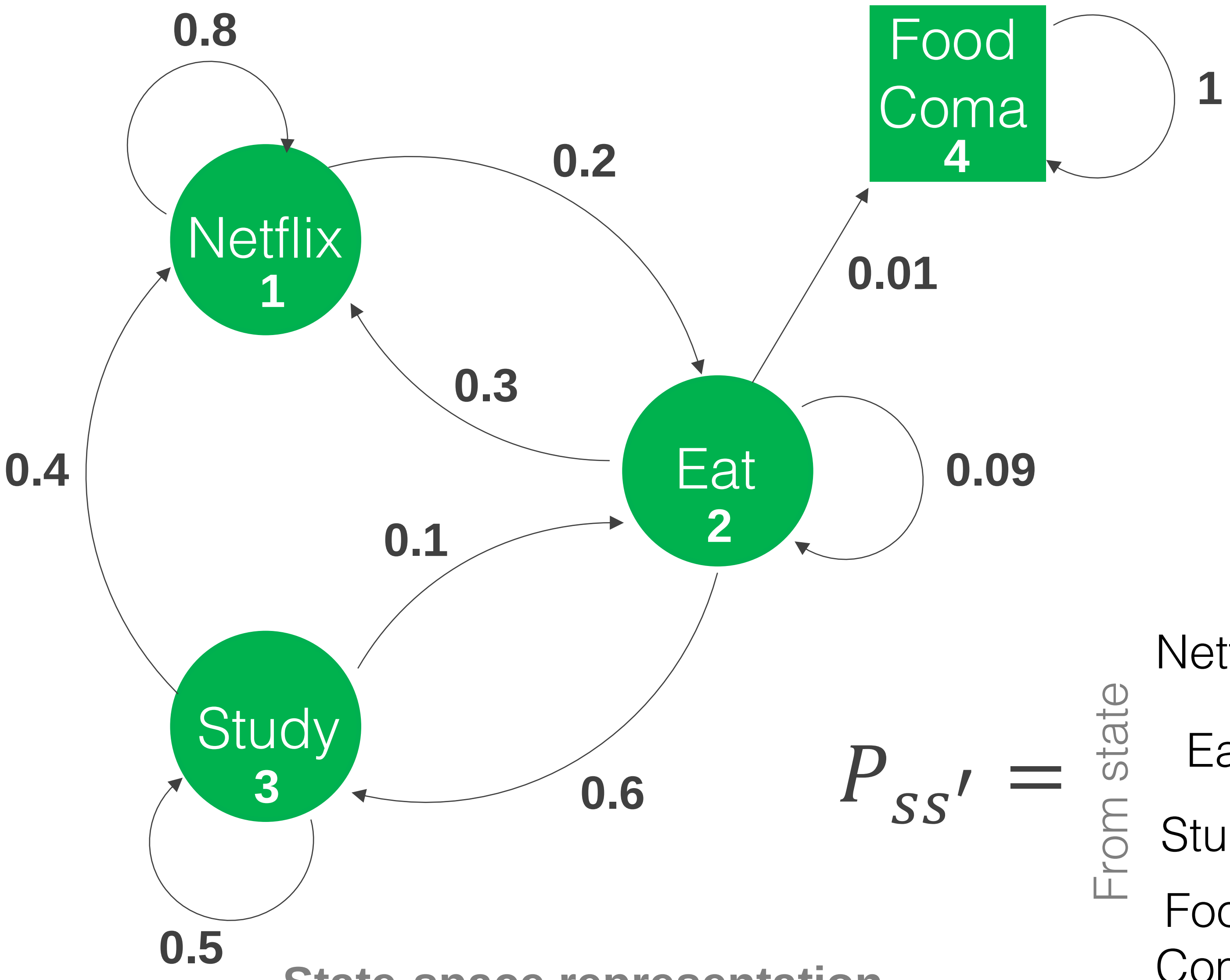
n $P_n = P_0 P_{ss'}^n$

As $n \to \infty$, we identify our steady state probabilities

$$P_\infty = \begin{bmatrix} 0.64 & 0.16 & 0.20 \end{bmatrix}$$

Markov Chains with absorbing state

Example: student life

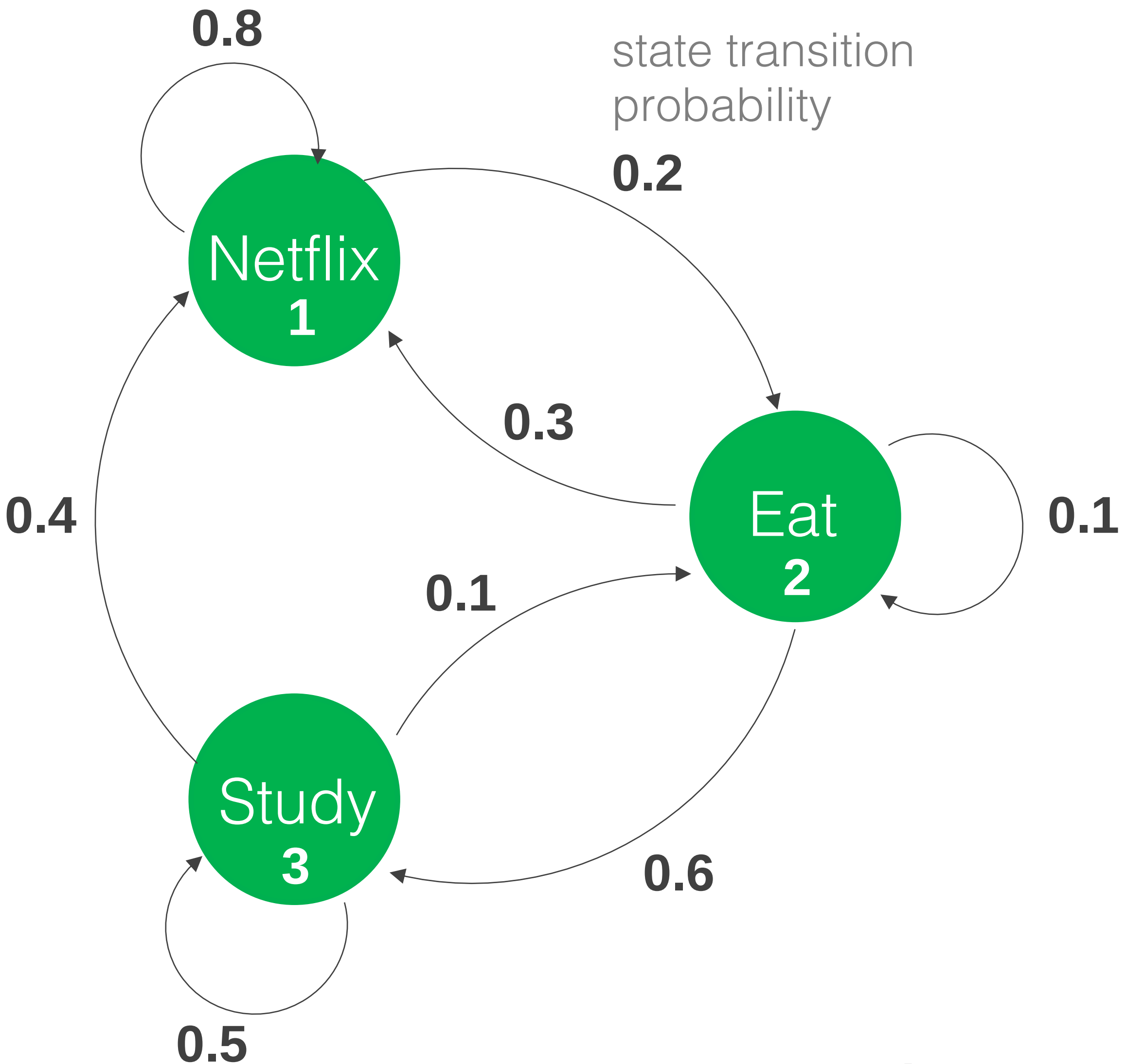


State-space representation

$$P_{ss'} = \begin{array}{cc} & \text{To state} \\ & \begin{array}{cccc} \text{Netflix} & \text{Eat} & \text{Study} & \text{Food Coma} \end{array} \\ \text{From state} \begin{array}{r} \text{Netflix} \\ \text{Eat} \\ \text{Study} \\ \text{Food Coma} \end{array} & \begin{bmatrix} 0.8 & 0.2 & 0 & 0 \\ 0.3 & 0.09 & 0.6 & 0.01 \\ 0.4 & 0.1 & 0.5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

Markov Chains

Example: student life



State-space representation

Markov chains can be used to represent sequential discrete-time data

Can estimate long-term state probabilities

Can simulate state sequences based on the model

Markov property applies

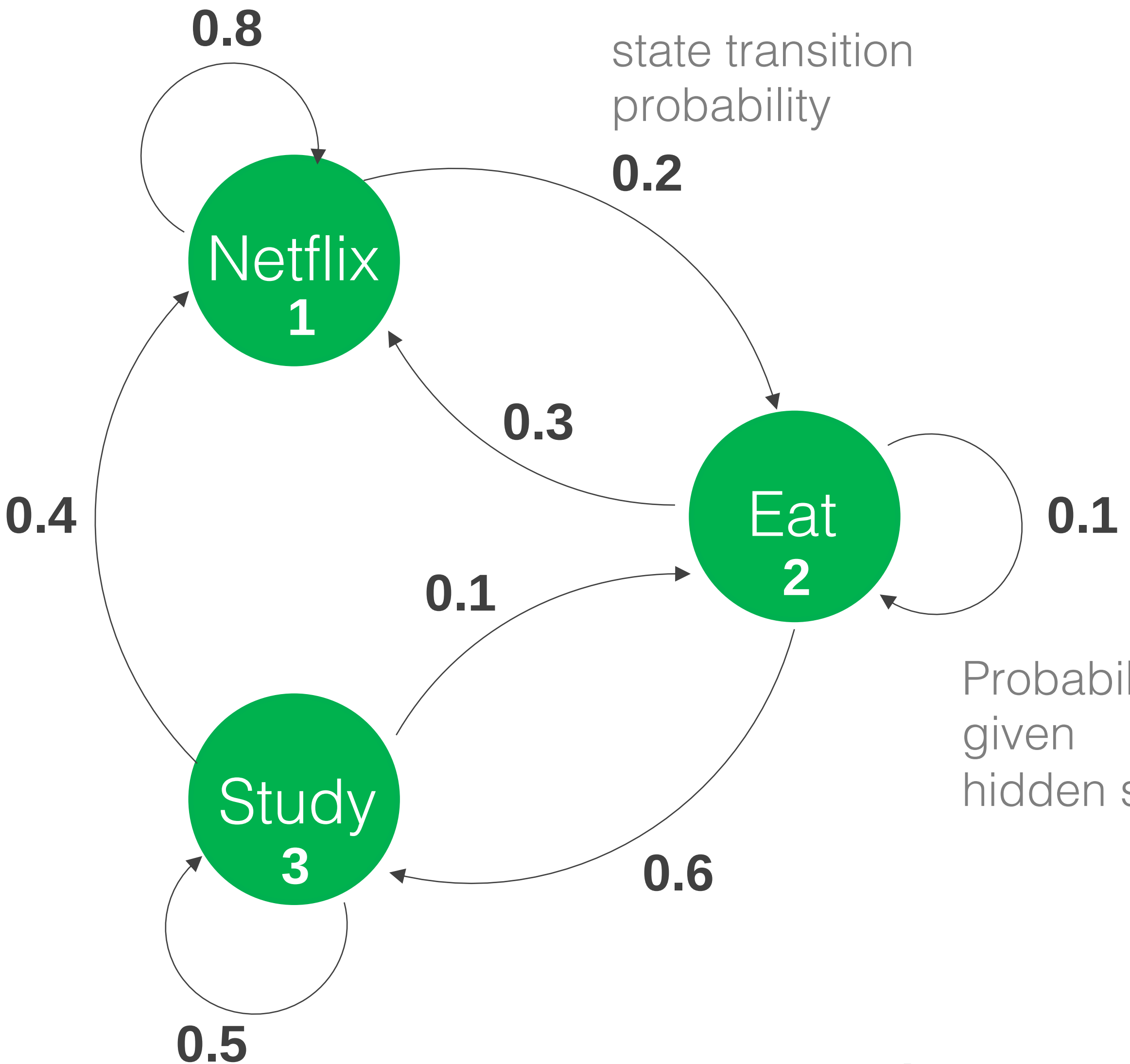
(current state gives you all the information you need about future states)

$$P(s_{t+1}|s_t) = P(s_{t+1}|s_t, s_{t-1}, \dots, s_1, s_0)$$

Valid if the system is **autonomous** and the states are **fully observable**

Hidden Markov Models

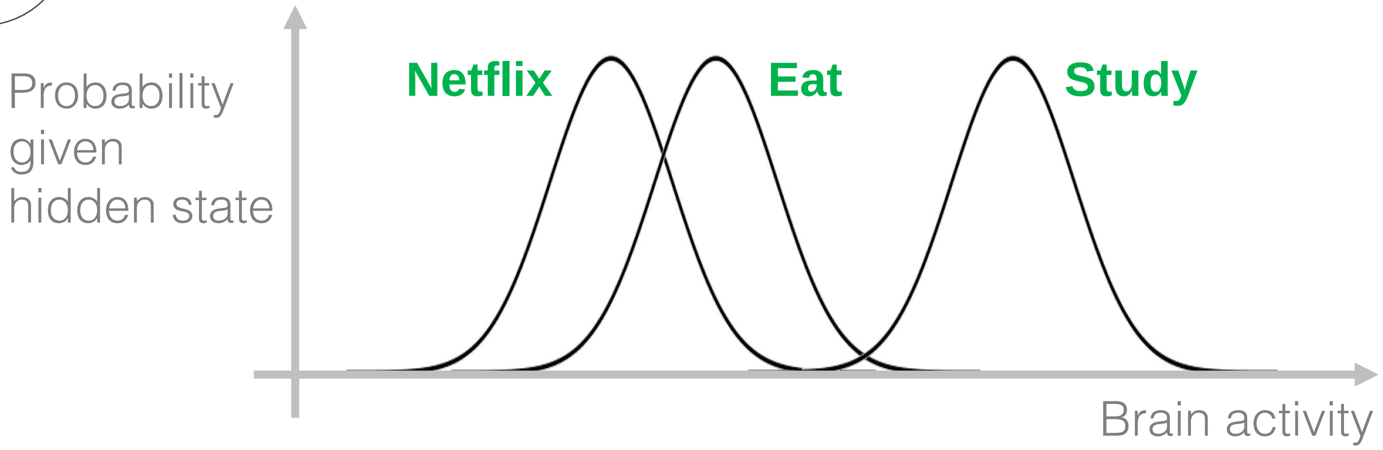
Example: student life



State-space representation

What if we **don’t directly observe what state** the system is in, but instead observe a quantity that depends on the state?

Example: the student wears an EEG monitor, and we see readings of brain activity.



States are **hidden** or **latent variables**

Markov Models

	States are **Fully Observable**	States are **Partially Observable**
Autonomous (no actions; make predictions)	Markov Chain, Markov Reward Process	Hidden Markov Model (HMM)
Controlled (can take actions)	Markov Decision Process (MDP)	Partially Observable Markov Decision Process (POMDP)

Applications

HMMs: time series ML, e.g. speech + handwriting recognition, bioinformatics

MDPs: used extensively for reinforcement learning

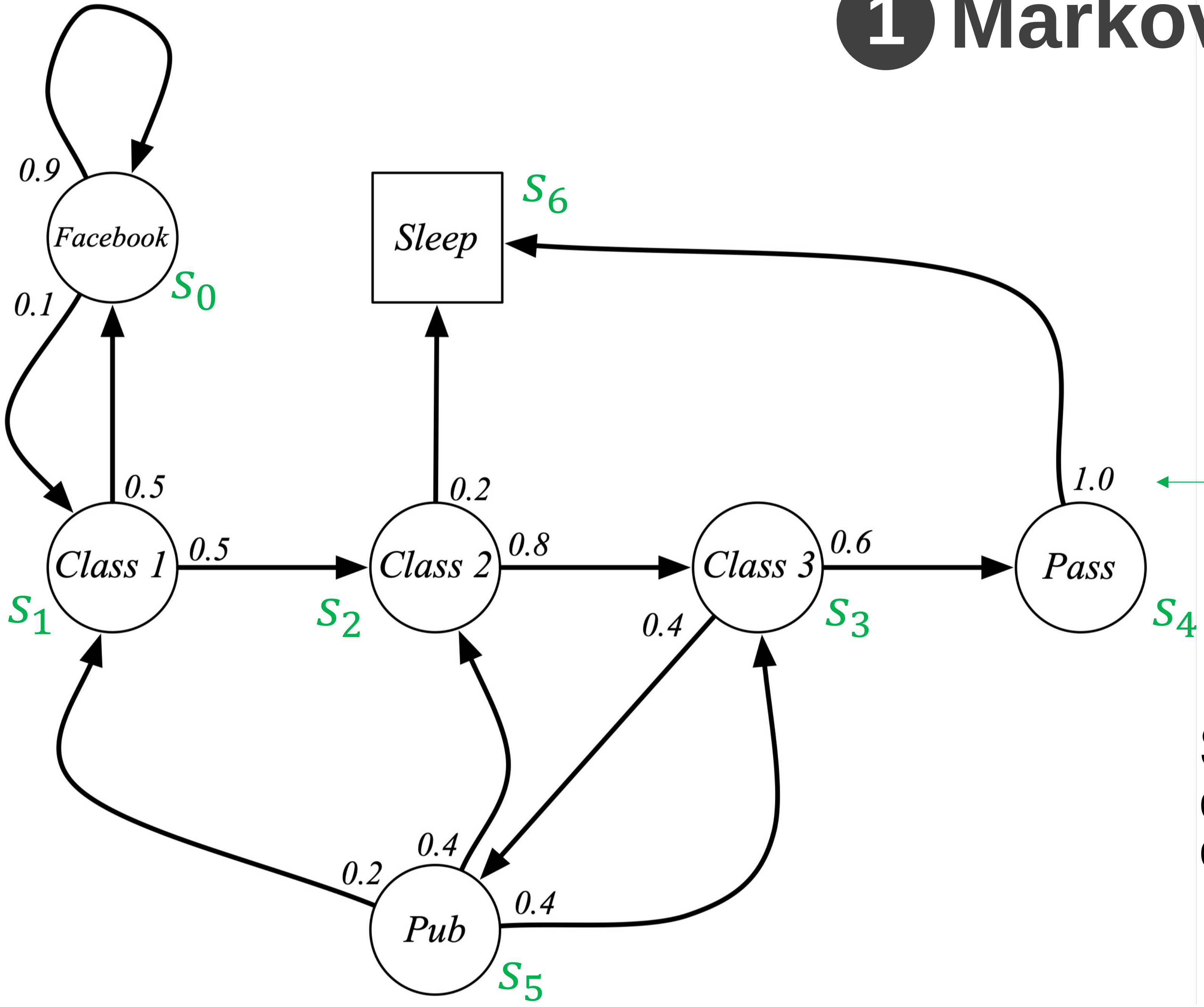
Building blocks for the full RL problem

1	Markov Chain	$\{$state space S, transition probabilities $P\}$
2	Markov Reward Process (MRP)	$\{S, P,$ + rewards R, discount rate $\gamma\}$ adds rewards (and values)
3	Markov Decision Process (MDP)	$\{S, P, R, \gamma,$ + actions $A\}$ adds decisions (i.e. the ability to control)

MDPs form the basis for most reinforcement learning environments

Adapted from David Silver, 2015

❶ Markov Chain Example



Components:

State space S,

Transition probabilities P

$$P_{46} = P_{ss'}$$

Sample Episodes:

C1,C2,Sleep

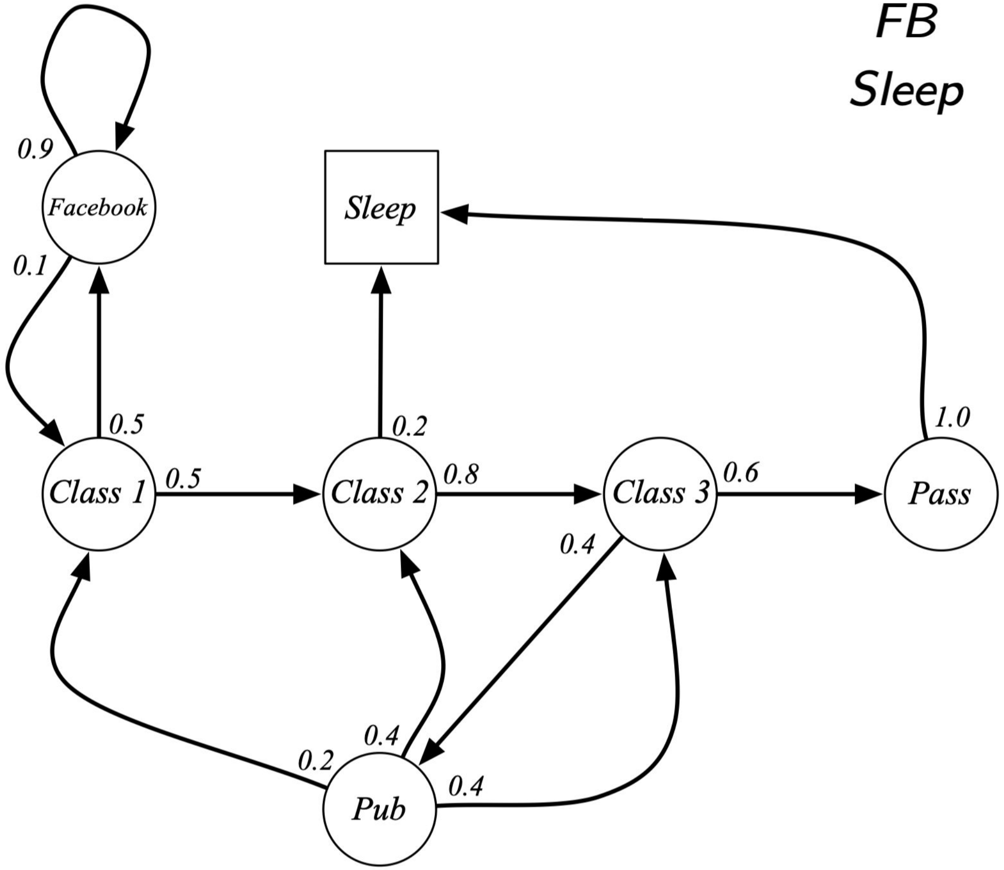
C1,FB,FB,FB,C1,C2,C3,Pass,Sleep

Example from David Silver, UCL, 2015

Markov Chain

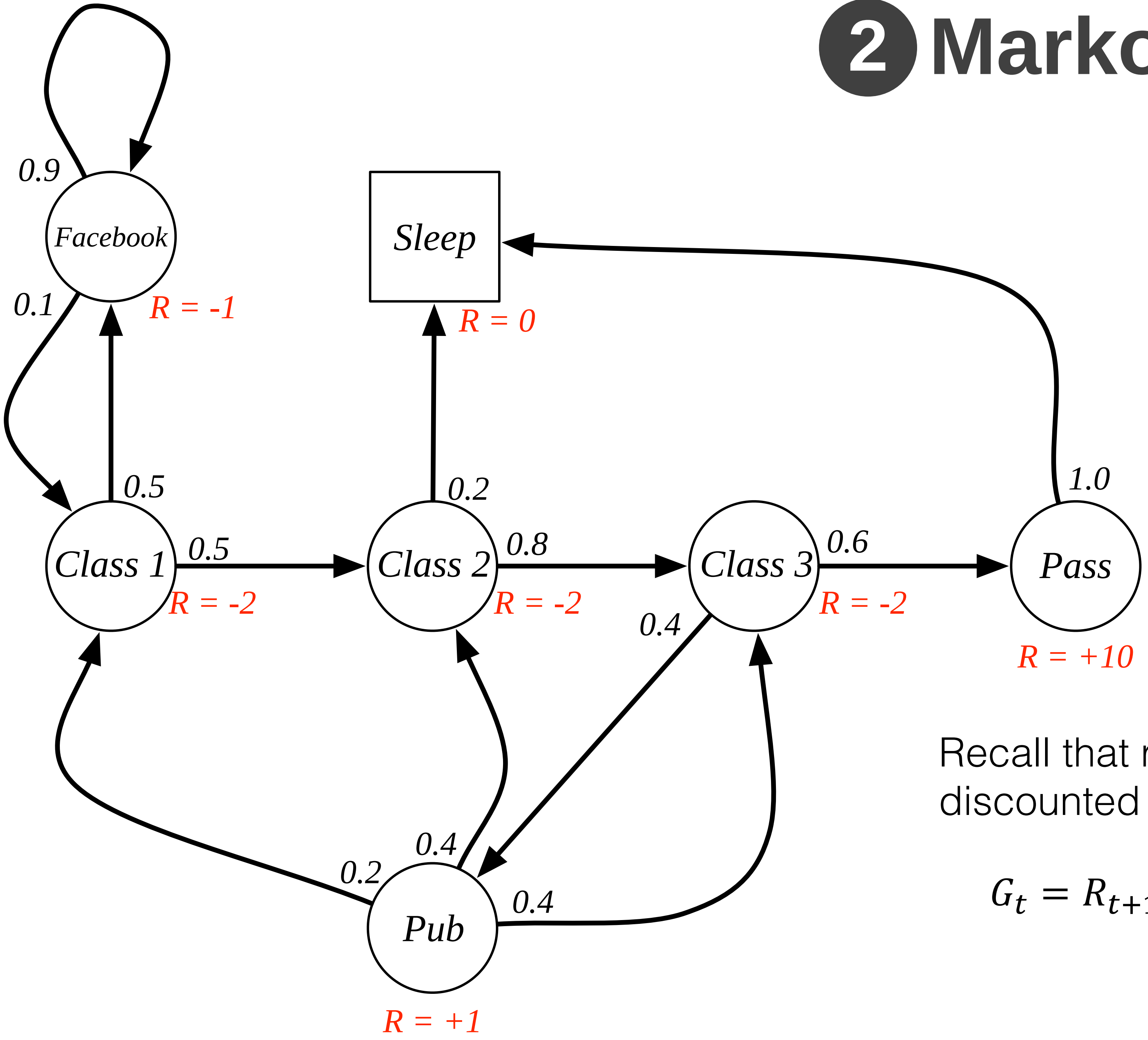
$$\mathcal{P} = \begin{array}{c|ccccccc} & C1 & C2 & C3 & Pass & Pub & FB & Sleep \\ \hline C1 & & 0.5 & & & & 0.5 & \\ C2 & & & 0.8 & & & & 0.2 \\ C3 & & & & 0.6 & 0.4 & & \\ Pass & & & & & & & 1.0 \\ Pub & 0.2 & 0.4 & 0.4 & & & & \\ FB & 0.1 & & & & & 0.9 & \\ Sleep & & & & & & & 1 \end{array}$$

State transition probability matrix, $P_{ss'}$



Example from David Silver, UCL, 2015

❷ Markov Reward Process



Components:

State space S,

Transition probabilities, P

Rewards, R

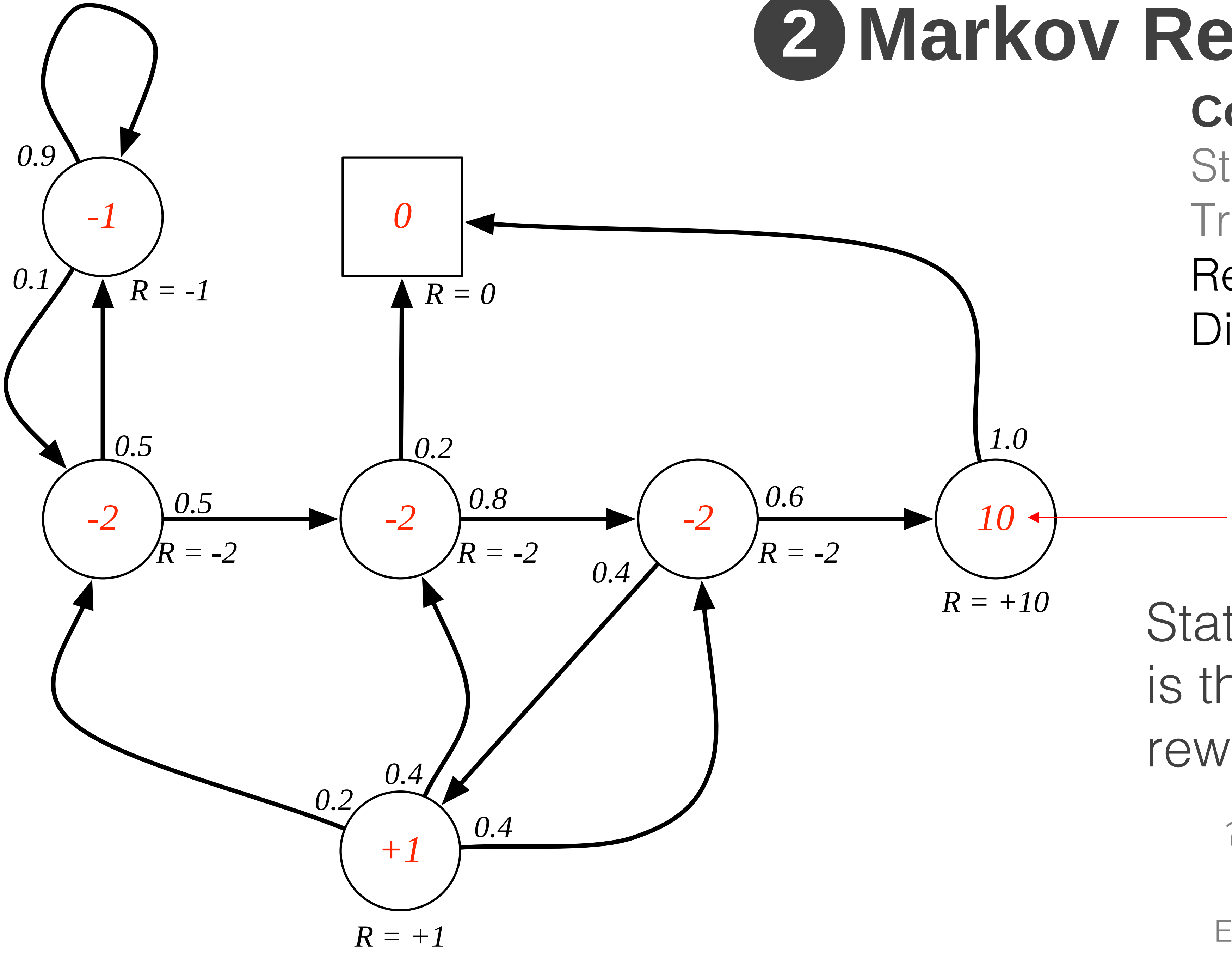
Discount rate, γ

Recall that returns, let's call G_t, are the total discounted rewards from time t:

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

Example from David Silver, UCL, 2015

❷ Markov Reward Process



$v(s)$ for $\gamma = 0$

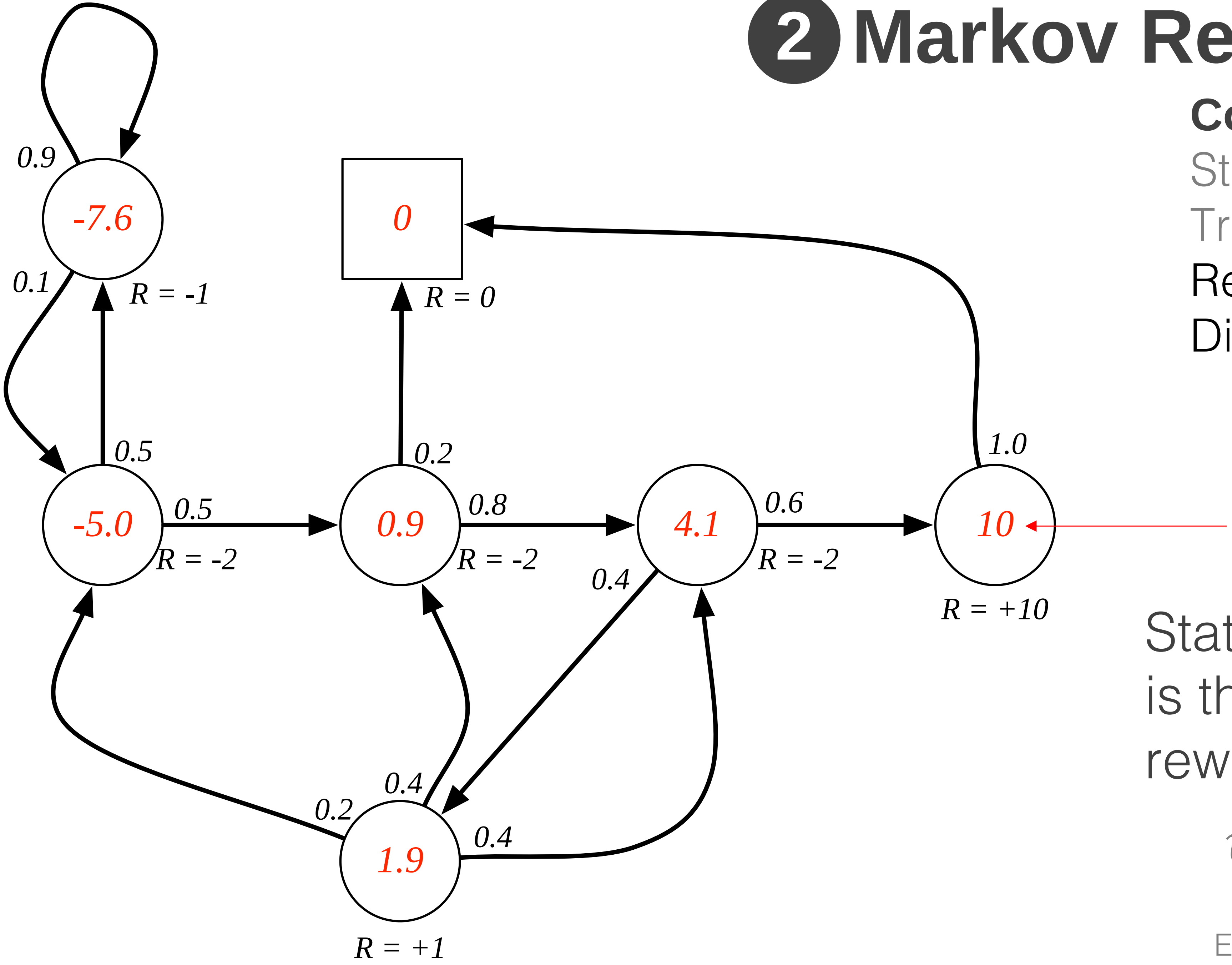
Components:
State space S,
Transition probabilities, P
Rewards, R
Discount rate, γ

State value function $v(s)$ is the expected total reward (into the future)

$$v(s) = E[G_t | S = s_t]$$

Example from David Silver, UCL, 2015

❷ Markov Reward Process



$v(s)$ for $\gamma = 0.9$

Components:

State space S,

Transition probabilities, P

Rewards, R

Discount rate, γ

State value function $v(s)$ is the expected total reward (into the future)

$$v(s) = E[G_t | S = s_t]$$

Example from David Silver, UCL, 2015

“Backup” property of state value functions

$v(s_t) = E[G_t|S = s_t]$ where $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots$

$$= E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots |S = s_t]$$

$$= E[R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} \ldots)|S = s_t]$$

$$= E[R_{t+1} + \gamma G_{t+1}|S = s_t]$$

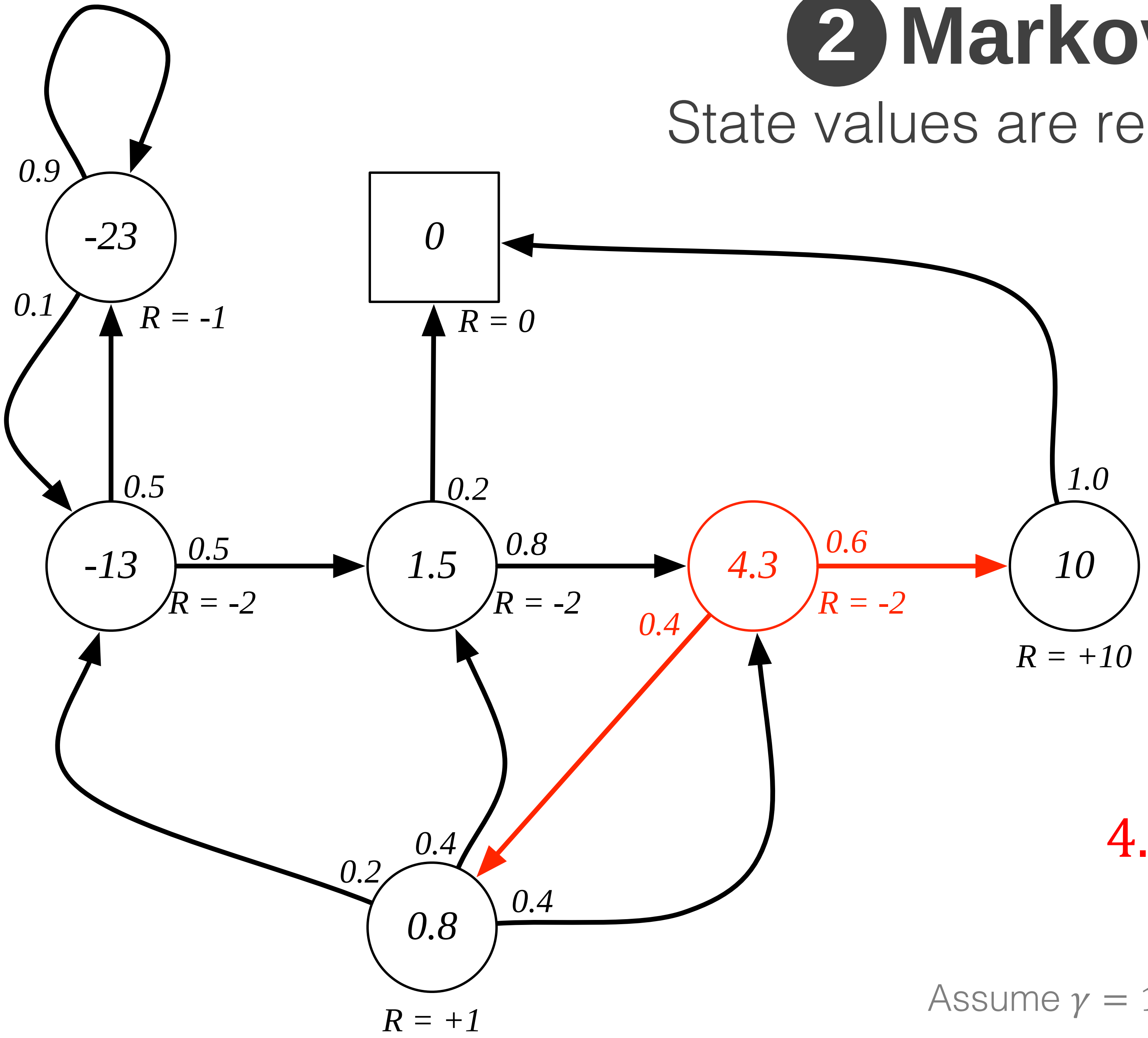
$$= E[R_{t+1} + \gamma v(s_{t+1})|S = s_t]$$

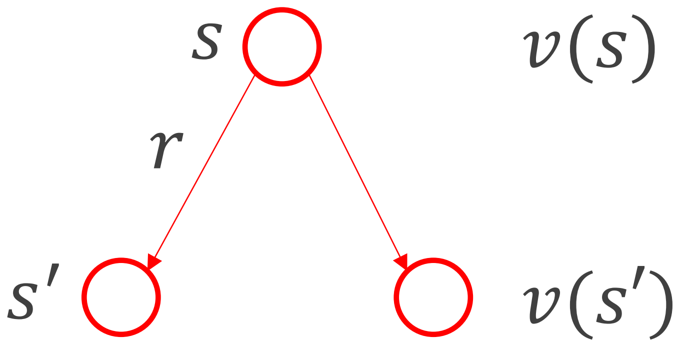
This recursive relationship is a version of the **Bellman Equation**

Example from David Silver, UCL, 2015

2 Markov Reward Process

State values are related to neighboring states





possible states we could transition to from s

$$v(s) = E[R_s + \gamma v(s')|s]$$

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

$$4.3 = -2 + 0.6\times10 + 0.4\times0.8$$

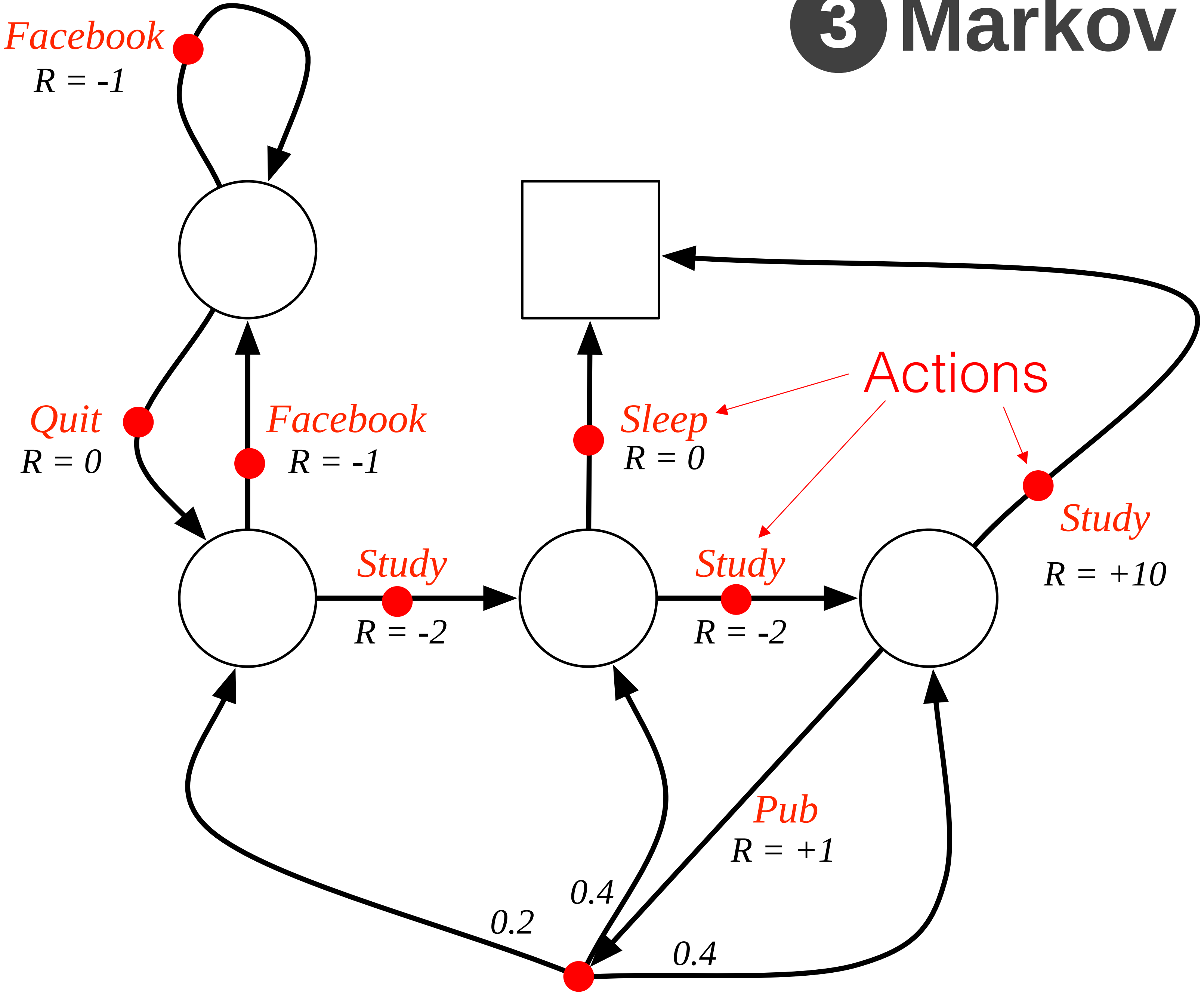
Notation: $s = s_t$ and $s' = s_{t+1}$

$R_s = E[R_{t+1}|S_t = s]$

Assume $\gamma = 1$

Example from David Silver, UCL, 2015

③ Markov Decision Process



Components:

State space S,

Transition probabilities, P

Rewards, R

Discount rate, γ

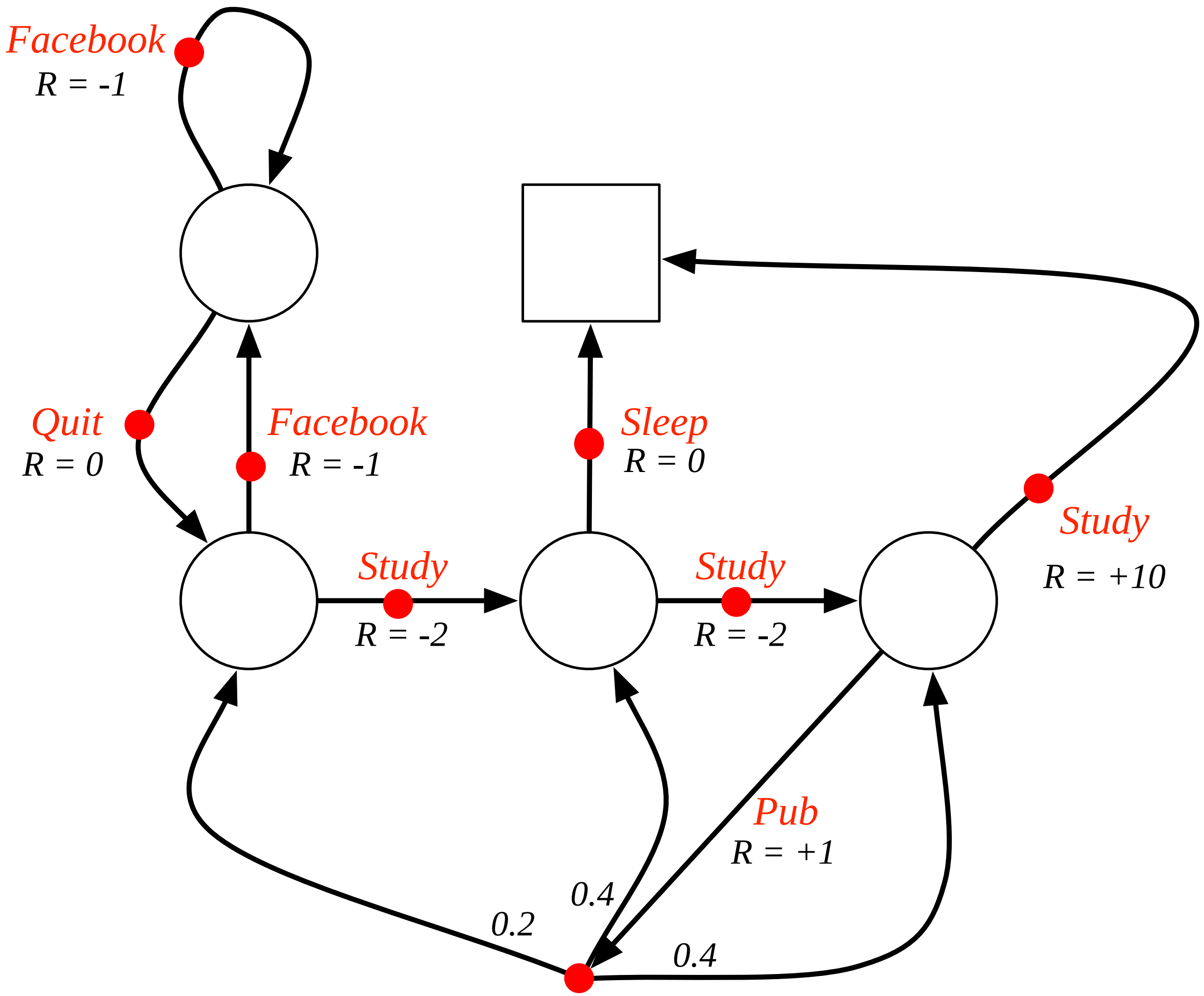
Actions, A

Adds interaction with the environment

An agent in a state chooses an action, the environment (the MDP) provides a reward and the next state

Example from David Silver, UCL, 2015

③ Markov Decision Process



Policy (how we choose actions)

(can be stochastic or deterministic)

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

State value function

(expected return from state s, and following policy π)

$$v_\pi(s) = E[G_t|s]$$

$$v_\pi(s) = E[R_s^a + \gamma v_\pi(s')|s]$$

Action value function

(expected return from state s, taking action a, and following policy π)

$$q_\pi(s,a) = E[G_t|s,a]$$

$$q_\pi(s,a) = E[R_s^a + \gamma q_\pi(s',a')|s,a]$$

$$R_s^a = E[r_{t+1}|S_t = s, A_t = a]$$

Example from David Silver, UCL, 2015

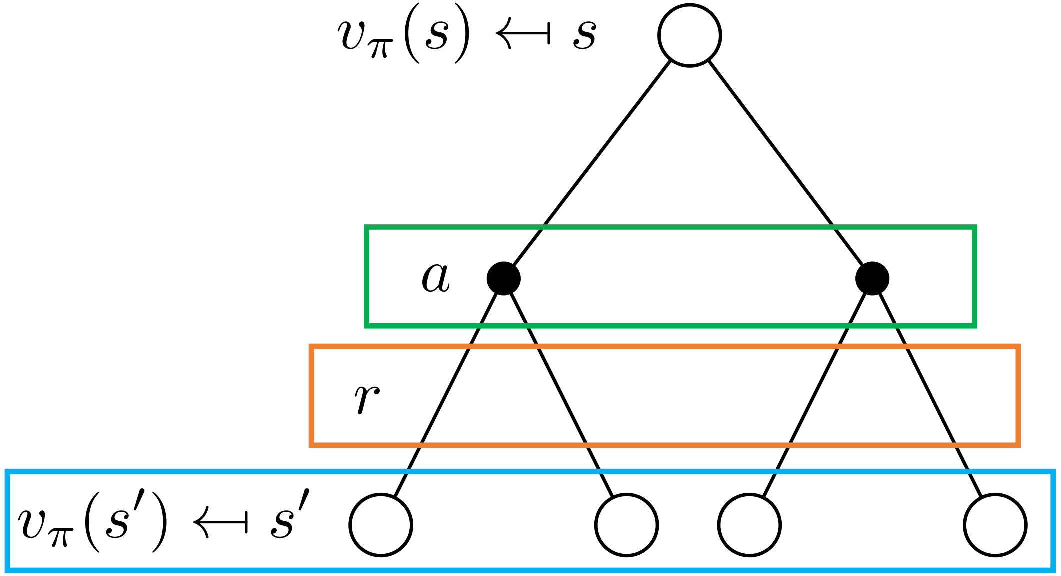
Bellman Expectation Equations for the state value function

(expected return from state s, and following policy π)

$$v_\pi(s) = E[G_t|s]$$

$$v_\pi(s) = E[R_s^a + \gamma v_\pi(s')|s]$$

$$R_s^a = E[R_{t+1}|S_t = s, A_t = a]$$



Expectation over the possible actions

Expectation over the rewards

(based on state and choice of action)

Expectation over the next possible states

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

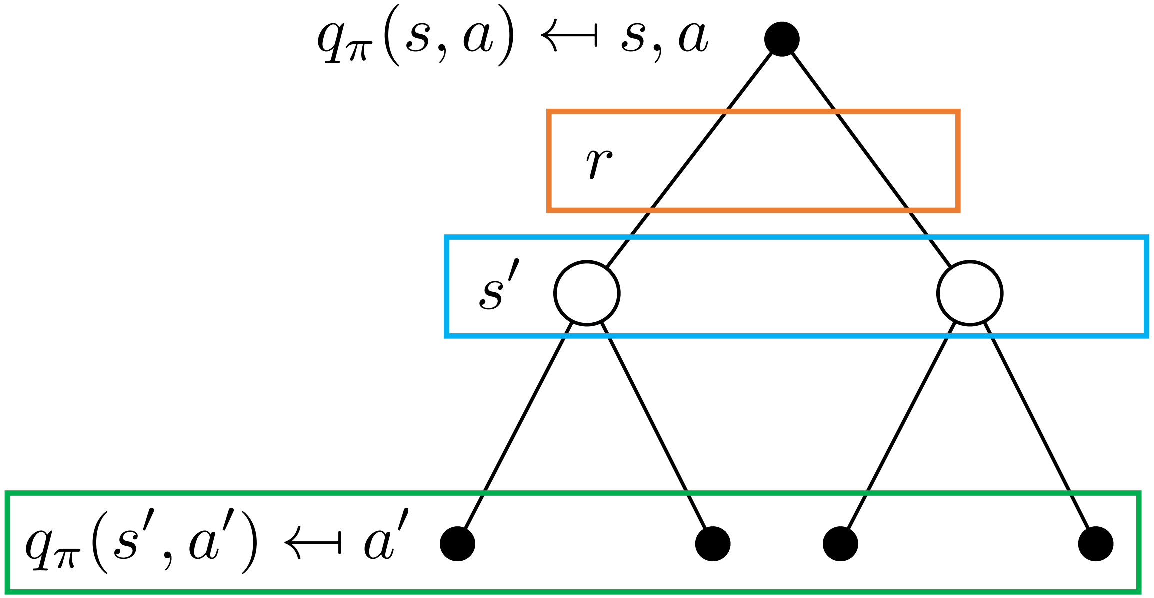
Bellman Expectation Equations for the action value function

(expected return from state s, taking action a, then following policy π)

$$q_\pi(s,a) = E[G_t|s,a]$$

$$q_\pi(s,a) = E[R_s^a + \gamma q_\pi(s',a')|s,a]$$

$$R_s^a = E[R_{t+1}|S_t = s, A_t = a]$$



Expectation over the rewards

(based on state and choice of action)

Expectation over the next possible states

Expectation over the possible actions

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

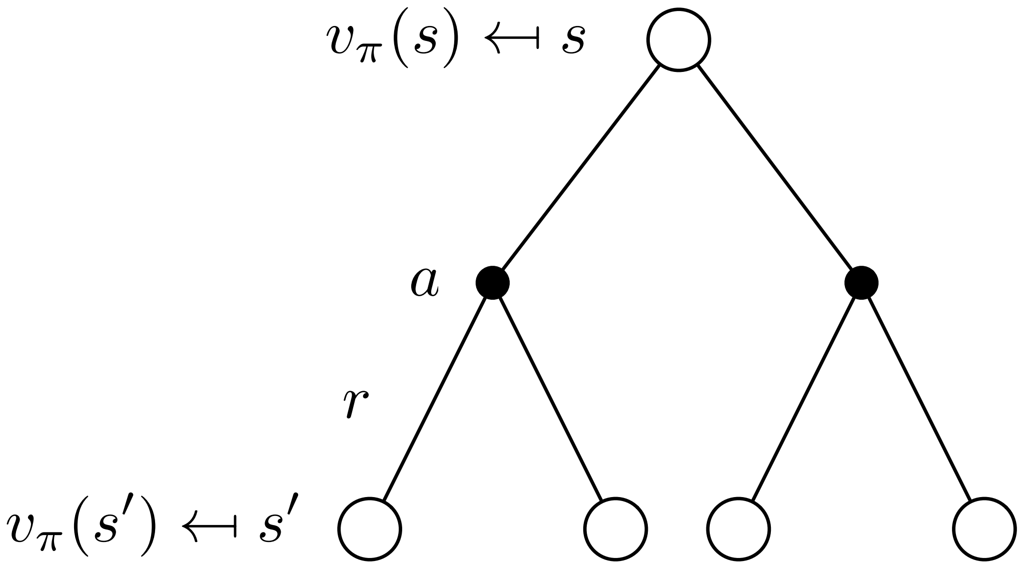
Bellman Expectation Equations

State value function

(expected return from state s, and following policy π)

$$v_\pi(s) = E[G_t|s]$$

$$v_\pi(s) = E[R_s^a + \gamma v_\pi(s')|s]$$



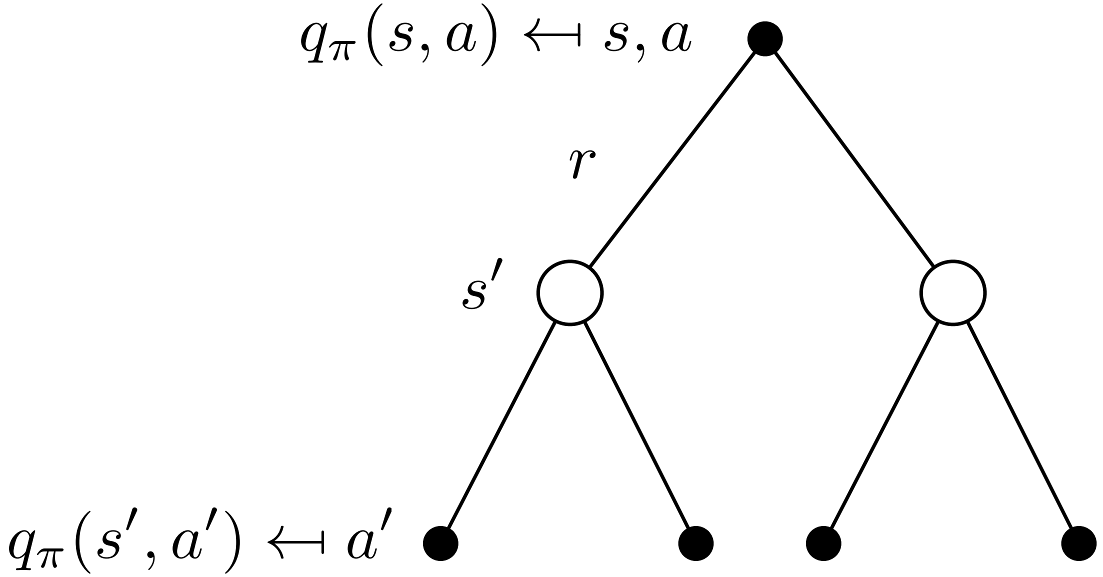
$$v_\pi(s) = \underline{\sum_a \pi(a|s)}\left(\underline{R_s^a} + \gamma \underline{\sum_{s'} P_{ss'}^a v_\pi(s')}\right)$$

Action value function

(expected return from state s, taking action a, then following policy π)

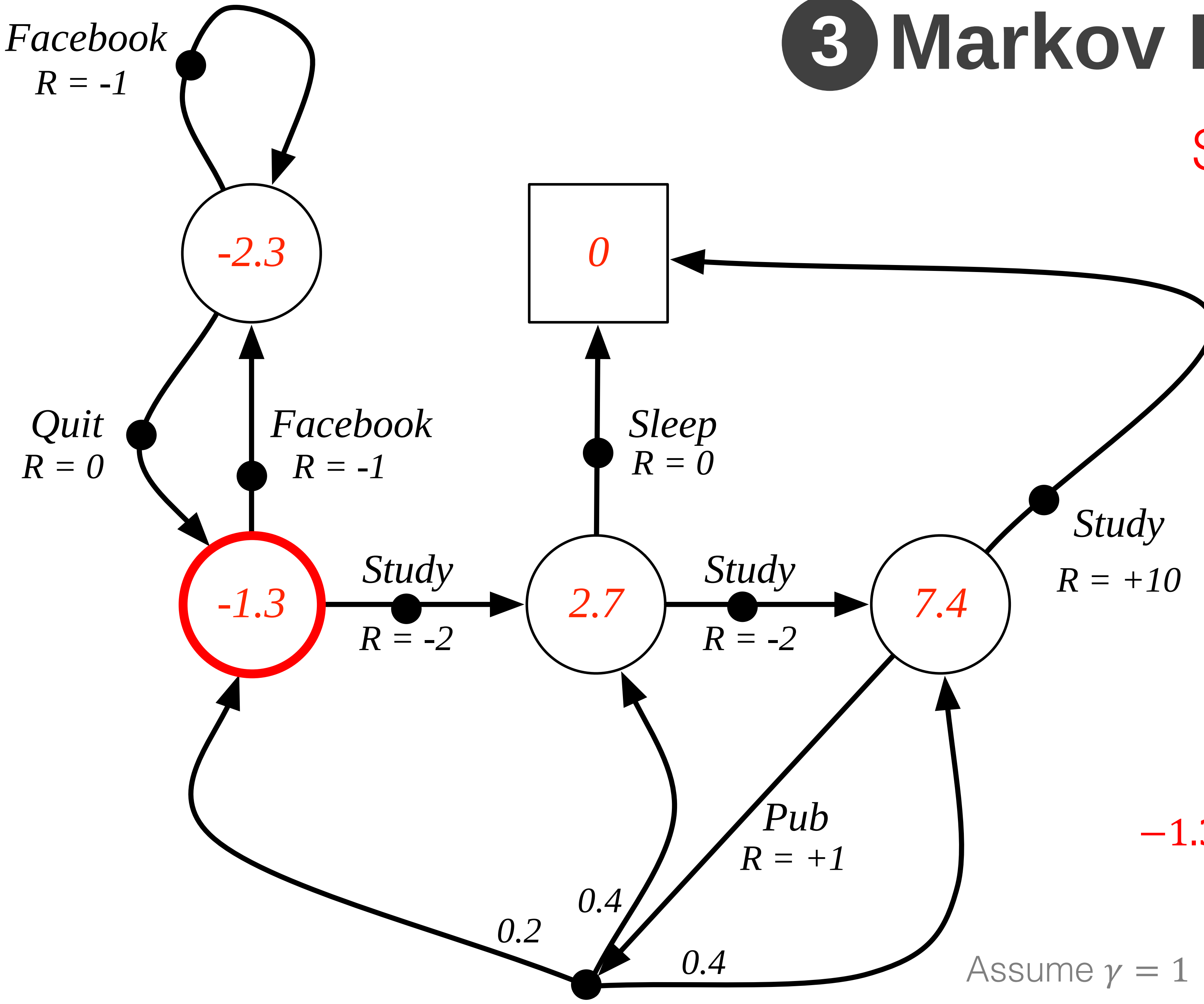
$$q_\pi(s,a) = E[G_t|s,a]$$

$$q_\pi(s,a) = E[R_s^a + \gamma q_\pi(s',a')|s,a]$$



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

❸ Markov Decision Process



Assume $\gamma = 1$

State value function, $\mathrm{v}_\pi(s)$

Assumptions:

$$\pi(a|s) = \frac{1}{\text{\# of reachable states}}$$

(randomly select an action)

$$\gamma = 1$$

(no discounting)

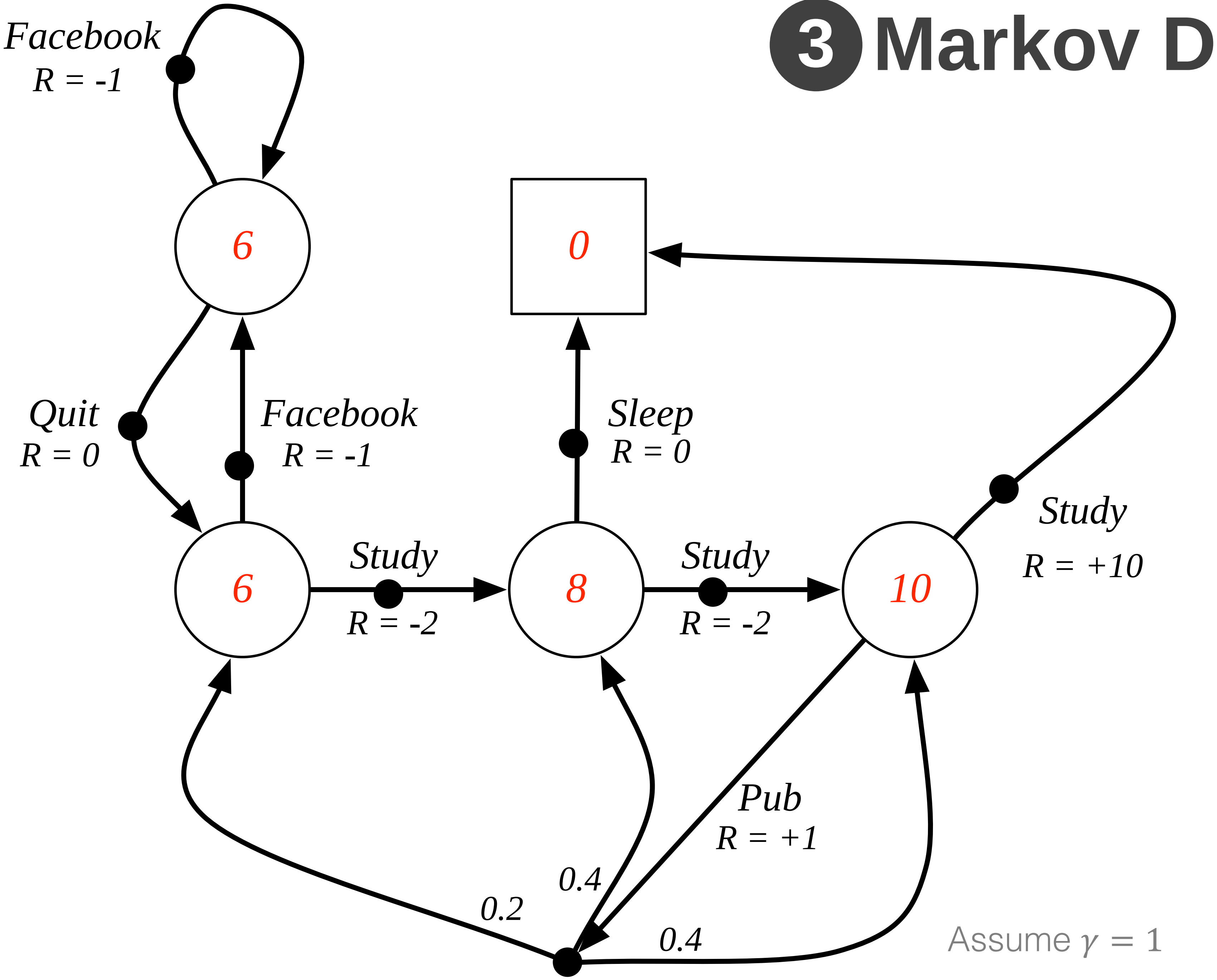
Bellman Expectation Equation

$$v(s) = E[R_s^a + \gamma v(s')|s]$$

$$-1.3 \approx 0.5\times(-1 - 2.3) + 0.5\times(-2 + 2.7)$$

Example from David Silver, UCL, 2015

3 Markov Decision Process



Assume $\gamma = 1$

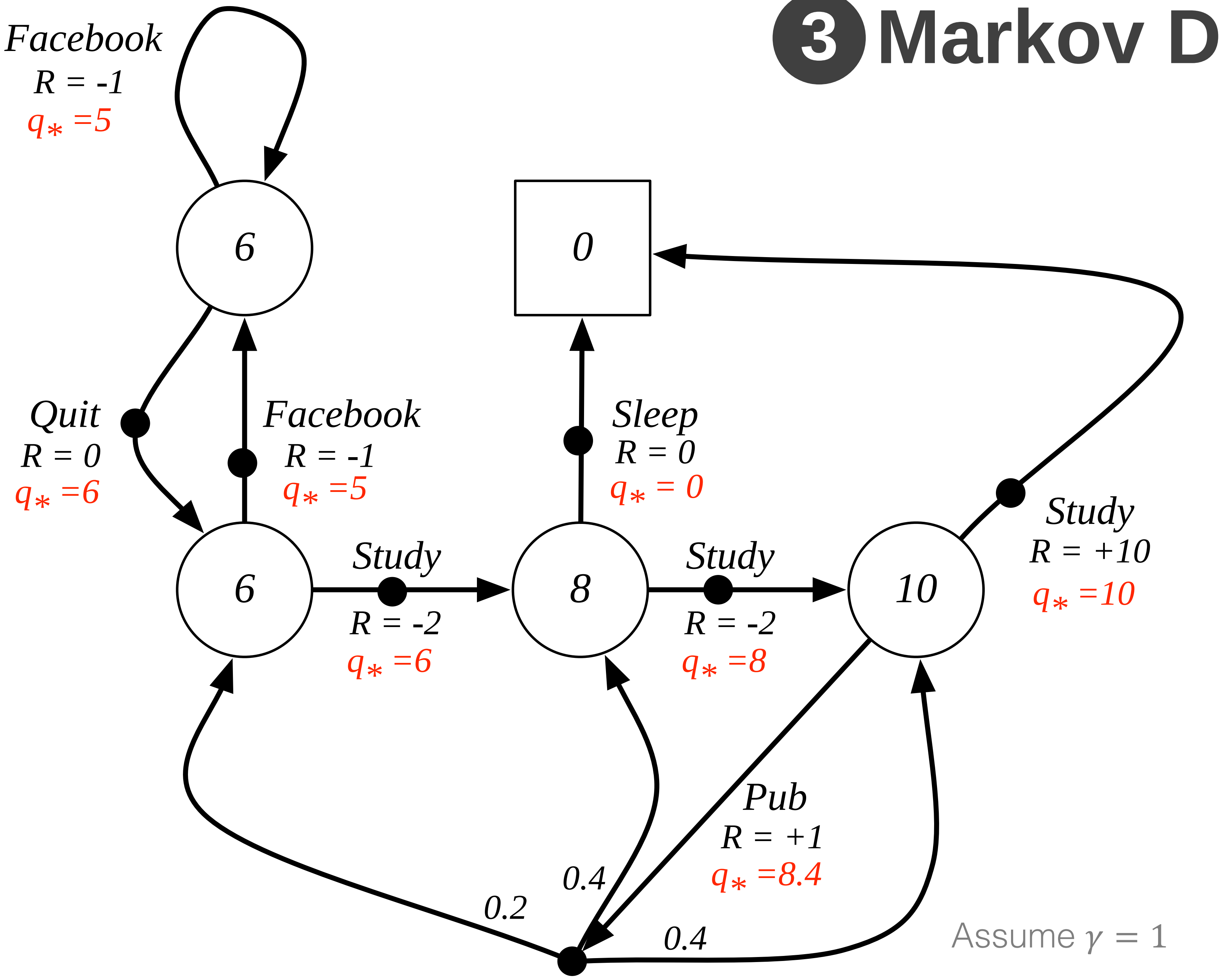
Optimal **state-value** function, $v_*(s)$

Maximum value function over all policies

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

Example from David Silver, UCL, 2015

❸ Markov Decision Process



Assume $\gamma = 1$

Optimal **state-value** function, $v_*(s)$

Maximum value function over all policies

$$v_*(s) = \max_\pi v_\pi(s)$$

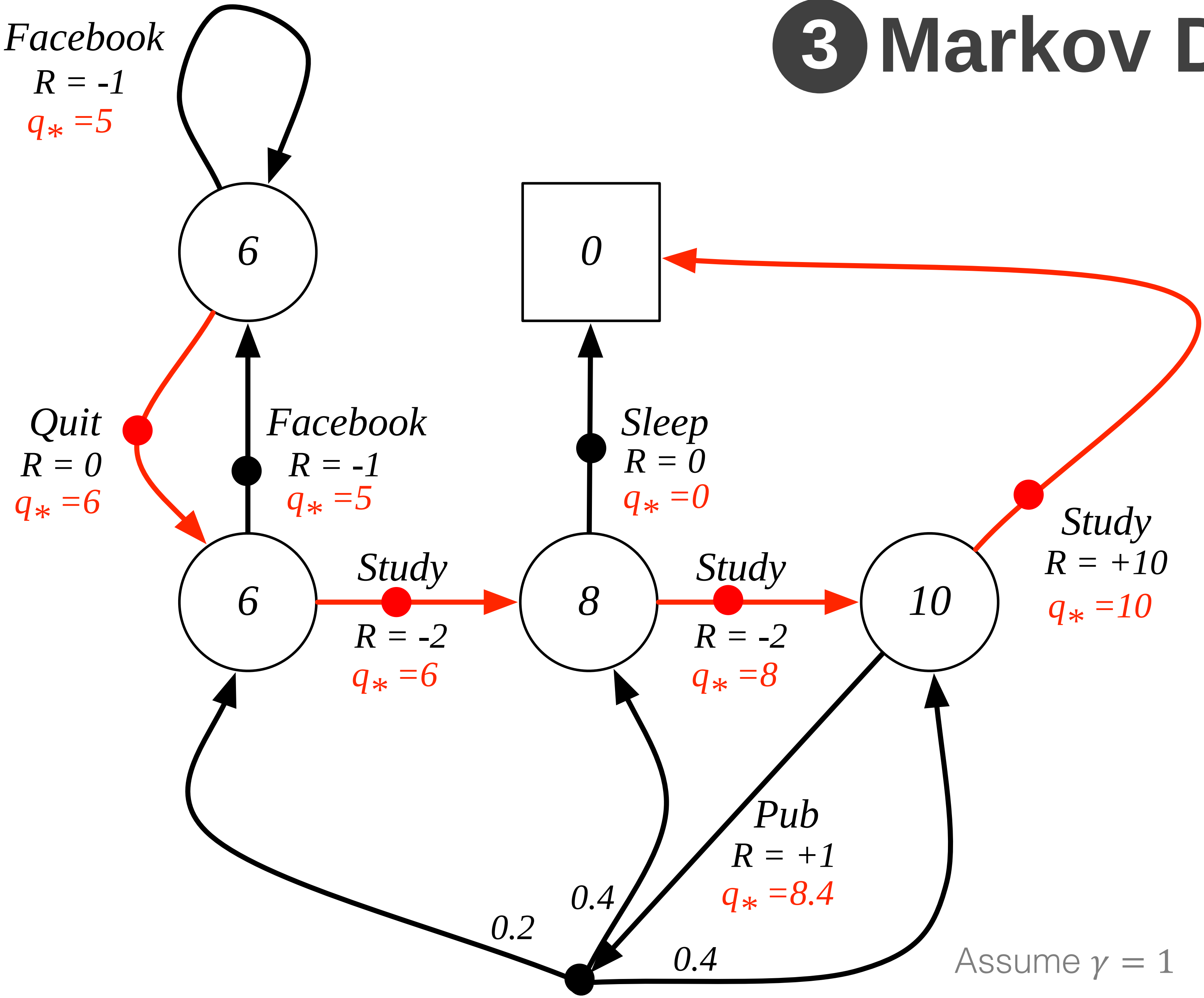
Optimal **action-value** function, $q_*(s,a)$

Maximum value function over all policies

$$q_*(s,a) = \max_\pi q_\pi(s,a)$$

Example from David Silver, UCL, 2015

③ Markov Decision Process



Optimal **policy**, $\pi_*(s)$

Which action to take at each moment

$$\pi_*(s) = \arg\max_a q_*(s, a)$$

Once we have the optimal value functions, we've "solved" the MDP!

Assume $\gamma = 1$

Example from David Silver, UCL, 2015

Reinforcement Learning Roadmap

Knowledge of **Environment**

Perfect knowledge
Known Markov Decision Process

Dynamic Programming

What's a Markov Decision Process?
How do we find optimal policies?

No knowledge
Must learn from experience

Monte Carlo Control

How do we estimate our value functions?
How do we use the value functions to choose actions?