Reinforcement Learning and Automated Planning

Class 1: Introduction

Ronen I. Brafman

Department of Computer Science

אוניברסיטת בן-גוריון בנגב
جامعة بن غوريون في النقب
Ben-Gurion University of the Negev

Course Goals

Understand models and algorithms developed, mostly in AI, to support autonomous systems, and to automate and support decision making processes.

Systems We Want to Build

Autonomous Systems

- Systems that can figure out how to perform a task without being explicitly programmed to do so
- Systems that can make decisions on their own without being explicitly programmed to do so
- Examples: Autonomous robots and droned, Autonomous Cars, Smart houses (IoT)

Decision Support Systems

- Systems that help us make decisions
- Examples: medical expert systems, logistics planning systems

Planning and RL

We will consider two cases:

- Automated Planning: We have a model of the system and we want to use it to make good decisions.
- Reinforcement Learning: We don't have a model of the system, and we want to learn to make good decisions through trial and error.

Planning for NASA's Mars Exploration Rovers

Autonomous Vehicle for Border Patrol

- Continuously monitor longer borders (fences)
- Able to recognize bombs planted and safely remove them
- Either fully autonomously, or operated remotely
- Saves lives!
- Joint project with IAI

The AOS – Tic Tac Toe

https://www.youtube.com/watch?v=ZV5-6NZ7H70

The AOS – Armadilo Simulation

https://www.youtube.com/watch?v=10sTQ8a_N6c

Some Well-Known RL Success Stories

- https://www.youtube.com/watch?v=0JL04JJjocc
- https://www.youtube.com/watch?v=dltN4MxV1RI
- https://www.youtube.com/watch?v=pkGa8ICQJS8
- https://www.youtube.com/watch?v=l9U8X6I1vow

General Information

- Instructor: Prof. Ronen Brafman
  - Office: 37/209
  - Office Hours: Sundays 14-16 on zoom
    - `https://us02web.zoom.us/j/5842390812`
  - Email: brafman@cs.bgu.ac.il

Course Outline

1. Introduction
2. Markov Processes; Markov decision processes (MDPs);
3. Classical MDP solution algorithms;
4. Specifying MDPs: factored models, dynamic Bayesian networks (DBN), deterministic models, and the AOS language;
5. Search-based solution algorithms for deterministic and stochastic MDPs: A*,AO*, RTDP, MCTS;
   - Assignment: Specify MDP using AOS language. Solve using AO*.
6. Model-Based Reinforcement Learning and Rmax;
   - Assignment: Use model-based RL to learn toy problem model in OpenAI Gym. Solve using value or policy iteration.

Course Outline - Continued

7. Model-Free Prediction;
8. Model-Free Control;
   - Assignment: Solve a problem using Q-learning and SARSA.
9. Value-Function Approximation;
10. Neural nets;
11. Deep RL: DQN and its descendants;
12. Policy Gradient methods;
    - Implement a PG algorithm.
13. POMDPs: Basic properties, specification with the AOS;
    - Assignment: Specify POMDP model using AOS language.
14. Point-based algorithms, Particle Filters, POMCP;
15. Dec-POMDPs;
    - Written or programming assignment on POMDPs.

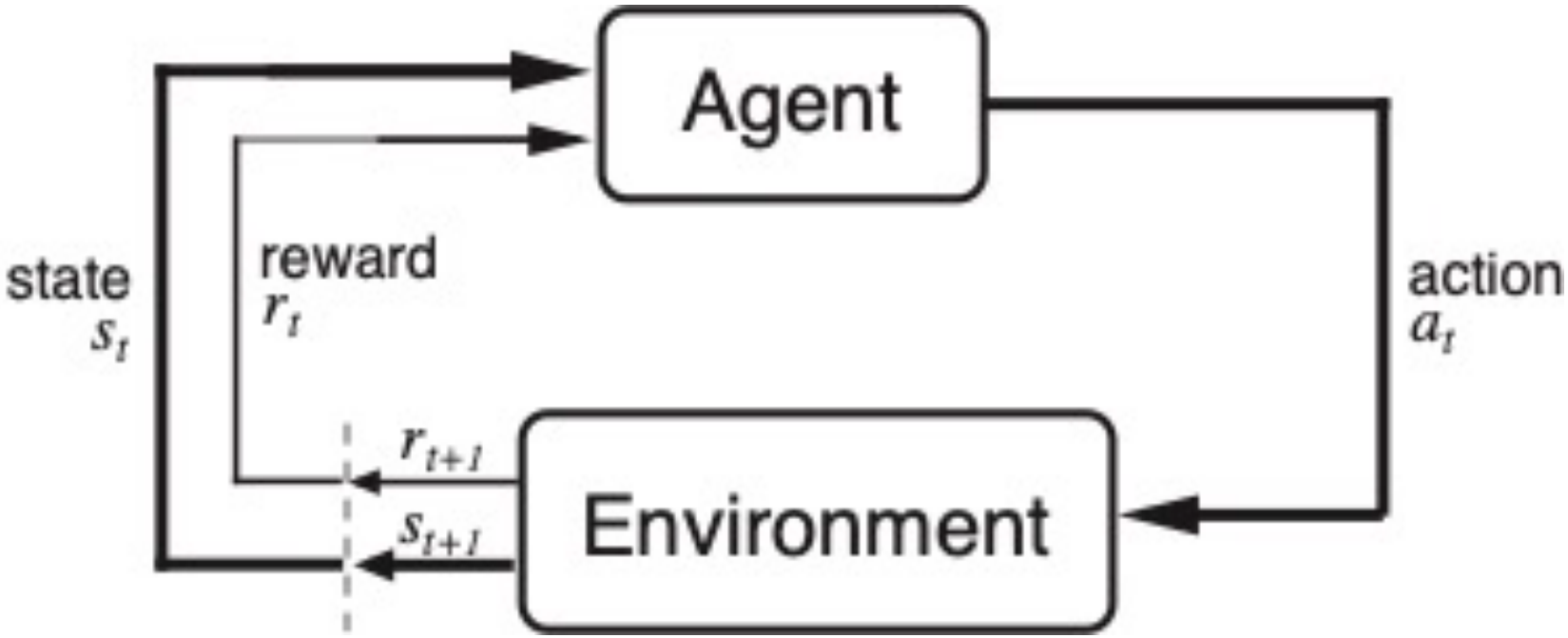
Workload and Grade

- Homework Assignments: 55%
- Exam 40% – Must pass to pass course.
- Physical Attendance 5% (0.715 pts per lecture – up to 7 lectures)
  - Reserve duty: 1 week = 2 lectures. Starting 17/12.

Resources

- David Silver's RL Course (on youtube)
- Sergey Levine's RL Course (on youtube)
- Book: Reinforcement Learning: An Introduction (2^{nd} edition). Sutton and Barto.

Sequential Decision Making: The Basic View



Three Key Concepts

Key Concepts

1. Models – what we mean (semantics)
2. Languages – how we specify it
3. Algorithms – how we solve it

Example: Finite State Machine

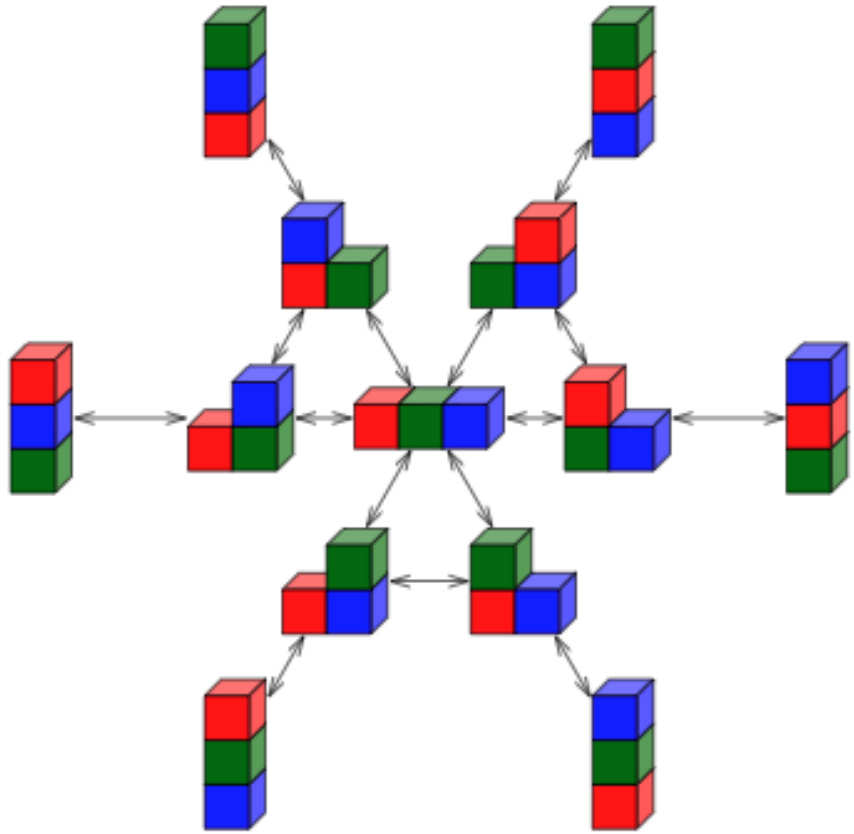
1. Model = Finite State Machine
2. Language = Finite State Machine $\langle S, \Sigma, Tr, s_0, A \rangle$
3. Algorithms = Find an accepted word

FSM is a Decision model

- A basic decision model
- Replace $\langle S, \Sigma, Tr, s_0, A \rangle$ by $\langle S, A, Tr, s_0, G \rangle$ where
  - A is a set of actions or possible decisions
  - G is a set of goal states
- Challenge: What actions should we take to reach a goal state?

Example: The Block's World

- Model $\rightarrow$
- Language = Model (= FSM)
- Algorithm = Dijkstra's



So What's the Language For?

If Language = Model why do we need a language?

So What's the Language For?

If Language = Model why do we need a language?

blocks	states
1	1
2	3
3	13
4	73
5	501
6	4051
7	37633
8	394353
9	4596553
10	58941091

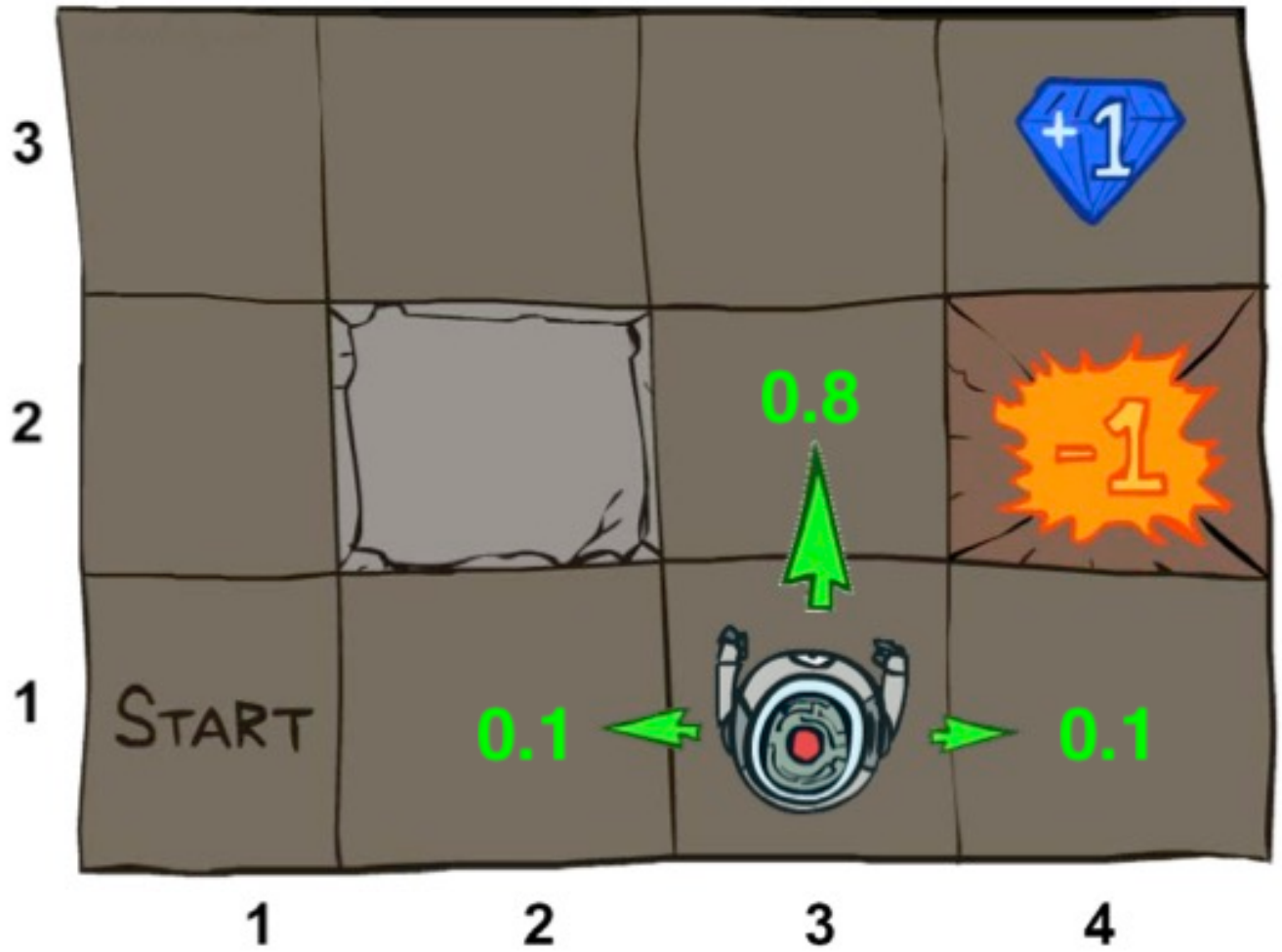
So What's the Language For?

If Language = Model, why do we need a language?

blocks	states
1	1
2	3
3	13
4	73
5	501
6	4051
7	37633
8	394353
9	4596553
10	58941091

We need a compact way to specify the model

FSM Are Not Enough: *Get the Diamond*



We need a more general model!

Markov Decision Processes (MDPs)

- In a nutshell – an MDP is a state machine with probabilistic transitions
- Markov decision processes formally describe an environment for most reinforcement learning problems
- They are a good model for planning when your actions have stochastic effects, but you can observe their outcome
- There are three key assumptions:
  - The transition function is stochastic
  - The state of the environment is fully observable
  - The current state completely characterizes the process

Markov Property

"The future is independent of the past given the present"

Definition

A state S_t is *Markov* if and only if

$$\mathbb{P}[S_{t+1} \mid S_t] = \mathbb{P}[S_{t+1} \mid S_1, ..., S_t]$$

- The state captures all relevant information from the history
- Once the state is known, the history may be thrown away
- i.e. The state is a sufficient statistic of the future

State Transition Matrix

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

$$\mathcal{P}_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s\right]$$

State transition matrix $\mathcal{P}$ defines transition probabilities from all states s to all successor states s',

$$\mathcal{P} = \text{from} \begin{matrix} \\ \\ \\ \end{matrix} \begin{matrix} \text{to} \\ \\ \\ \end{matrix} \begin{bmatrix} \mathcal{P}_{11} & \dots & \mathcal{P}_{1n} \\ \vdots & & \\ \mathcal{P}_{n1} & \dots & \mathcal{P}_{nn} \end{bmatrix}$$

where each row of the matrix sums to 1.

Markov Process

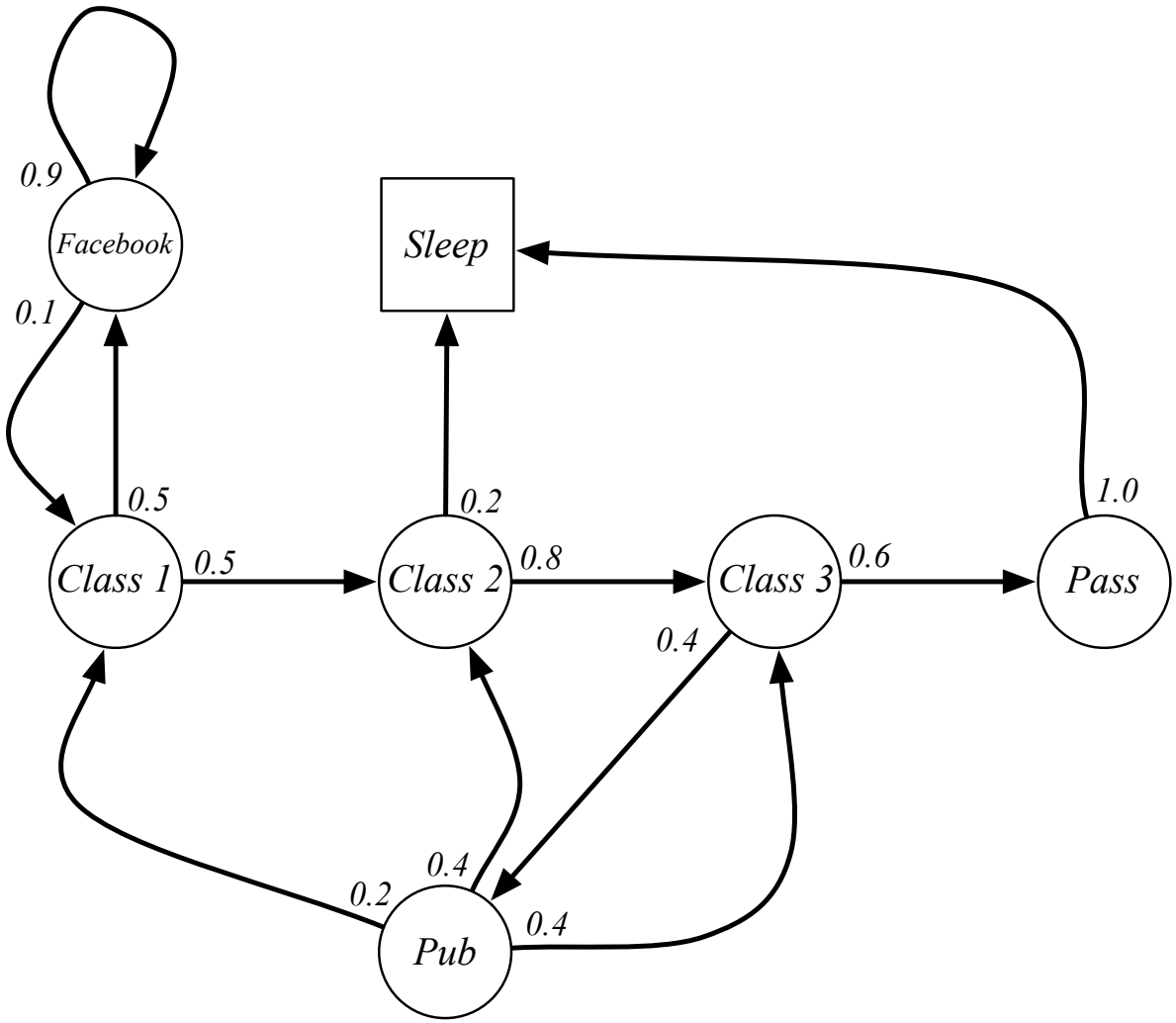
A Markov process is a memoryless random process, i.e. a sequence of random states $S_1, S_2, \ldots$ with the Markov property.

Definition

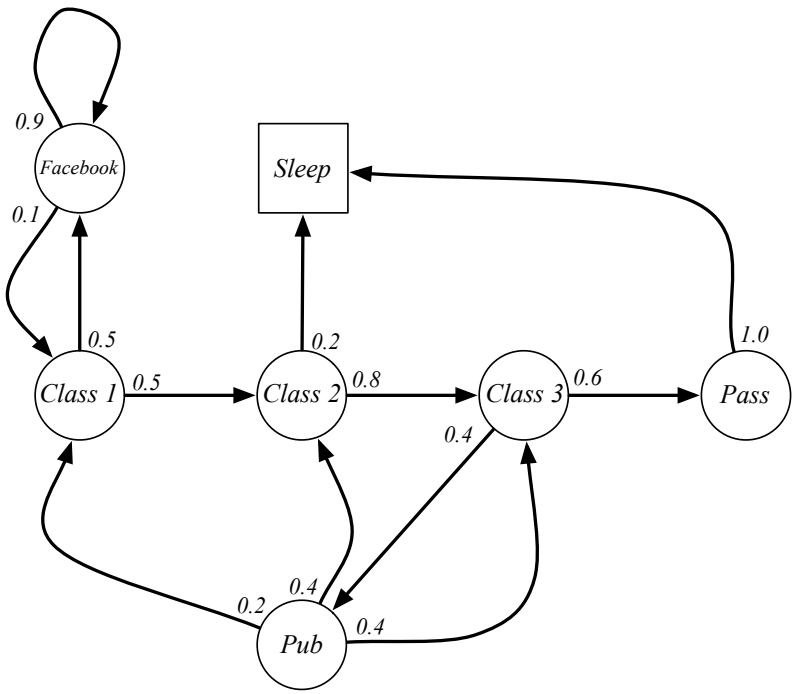
A *Markov Process* (or *Markov Chain*) is a tuple $\langle \mathcal{S}, \mathcal{P} \rangle$

- $\mathcal{S}$ is a (finite) set of states
- $\mathcal{P}$ is a state transition probability matrix,
  $\mathcal{P}_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s\right]$

Example: Student Markov Chain



Example: Student Markov Chain Episodes

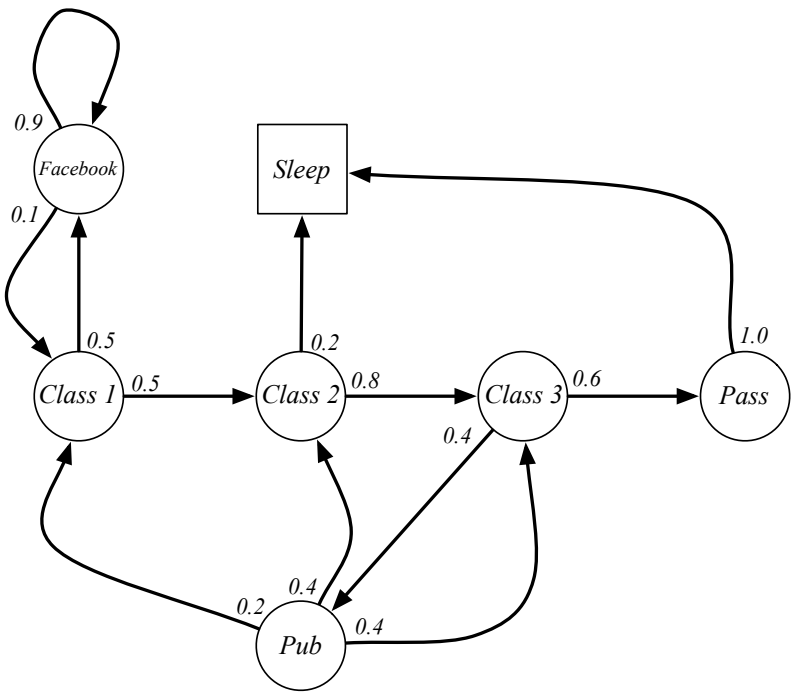


Sample episodes for Student Markov Chain starting from $S_1 = \text{C1}$

$$S_1, S_2, ..., S_T$$

- C1 C2 C3 Pass Sleep
- C1 FB FB C1 C2 Sleep
- C1 C2 C3 Pub C2 C3 Pass Sleep
- C1 FB FB C1 C2 C3 Pub C1 FB FB FB C1 C2 C3 Pub C2 Sleep

Example: Student Markov Chain Transition Matrix



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

Markov Reward Process

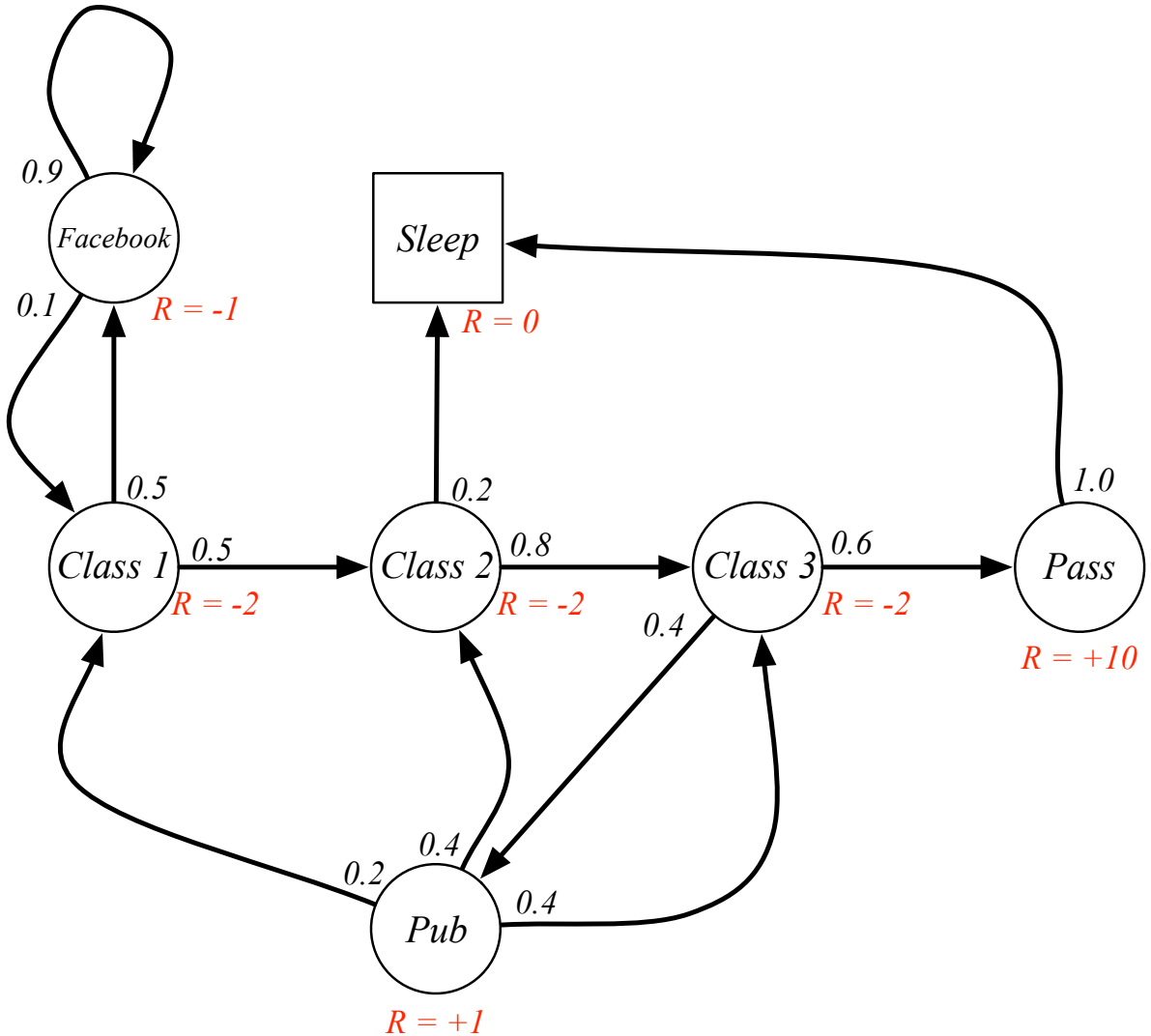
A Markov reward process is a Markov chain with values.

Definition

A *Markov Reward Process* is a tuple $\langle \mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- $\mathcal{S}$ is a finite set of states
- $\mathcal{P}$ is a state transition probability matrix, $\mathcal{P}_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s]$
- $\mathcal{R}$ is a reward function, $\mathcal{R}_s = \mathbb{E}[R_{t+1} \mid S_t = s]$
- γ is a discount factor, $\gamma \in [0, 1]$

Example: Student MRP



Return

Definition

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

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

- The *discount* $\gamma \in [0, 1]$ is the present value of future rewards
- The value of receiving reward R after $k + 1$ time-steps is $\gamma^k R$.
- This values immediate reward above delayed reward.
  - γ close to 0 leads to "myopic" evaluation
  - γ close to 1 leads to "far-sighted" evaluation

Why discount?

Most Markov reward and decision processes are discounted. Why?

- Mathematically convenient to discount rewards
- Avoids infinite returns in cyclic Markov processes
- Uncertainty about the future may not be fully represented
- If the reward is financial, immediate rewards may earn more interest than delayed rewards
- Animal/human behaviour shows preference for immediate reward
- It is sometimes possible to use *undiscounted* Markov reward processes (i.e. $\gamma = 1$), e.g. if all sequences terminate.

Value Function

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

Definition

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

$$v(s) = \mathbb{E}[G_t \mid S_t = s]$$

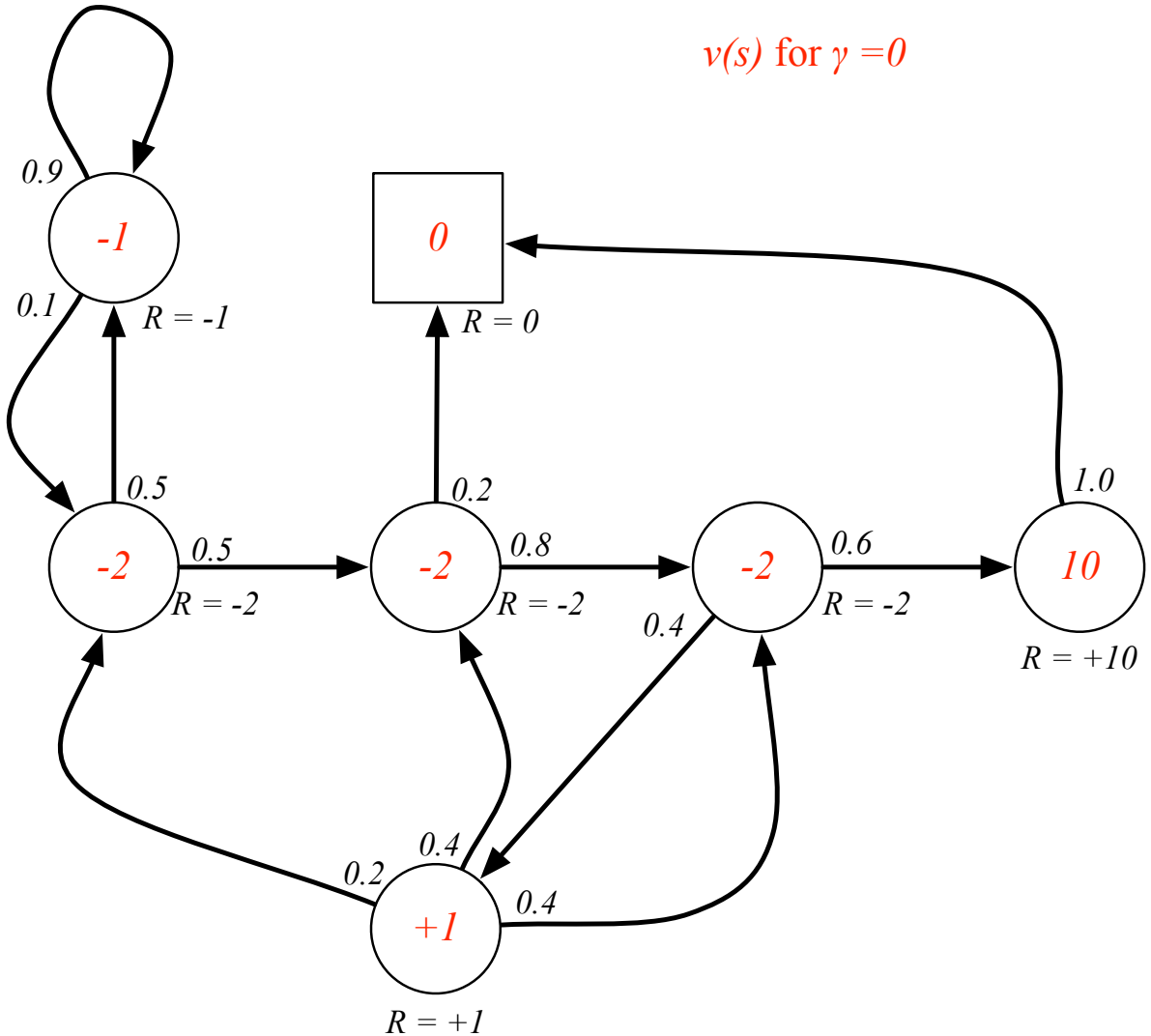
Example: Student MRP Returns

Sample returns for Student MRP:
Starting from $S_1 = \text{C1}$ with $\gamma = \frac{1}{2}$

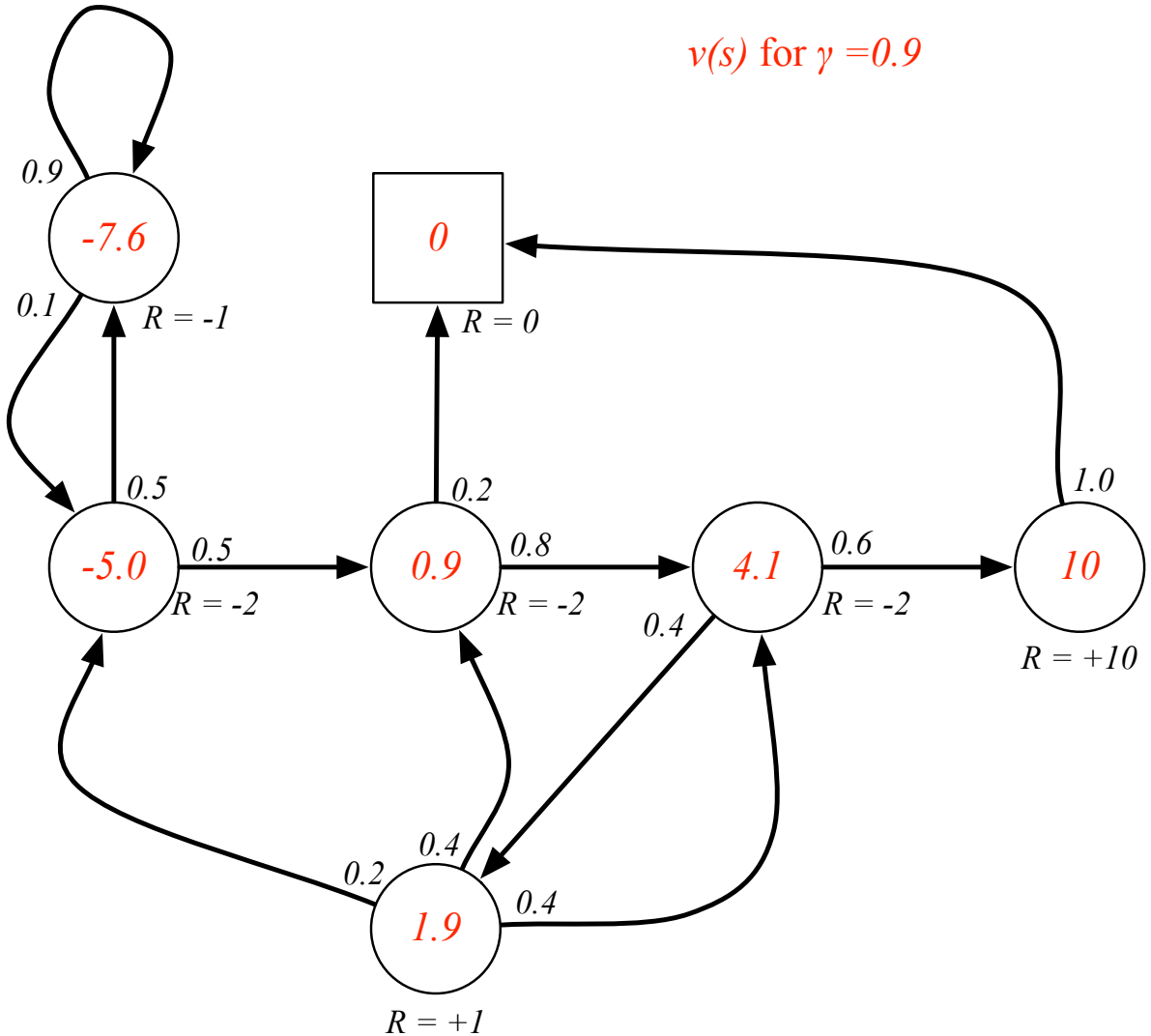
$$G_1 = R_2 + \gamma R_3 + ... + \gamma^{T-2} R_T$$

C1 C2 C3 Pass Sleep	$v_1 = -2 - 2 * \frac{1}{2} - 2 * \frac{1}{4} + 10 * \frac{1}{8}$	$=$	-2.25
C1 FB FB C1 C2 Sleep	$v_1 = -2 - 1 * \frac{1}{2} - 1 * \frac{1}{4} - 2 * \frac{1}{8} - 2 * \frac{1}{16}$	$=$	-3.125
C1 C2 C3 Pub C2 C3 Pass Sleep	$v_1 = -2 - 2 * \frac{1}{2} - 2 * \frac{1}{4} + 1 * \frac{1}{8} - 2 * \frac{1}{16} \dots$	$=$	-3.41
C1 FB FB C1 C2 C3 Pub C1 ... FB FB FB C1 C2 C3 Pub C2 Sleep	$v_1 = -2 - 1 * \frac{1}{2} - 1 * \frac{1}{4} - 2 * \frac{1}{8} - 2 * \frac{1}{16} \dots$	$=$	-3.20

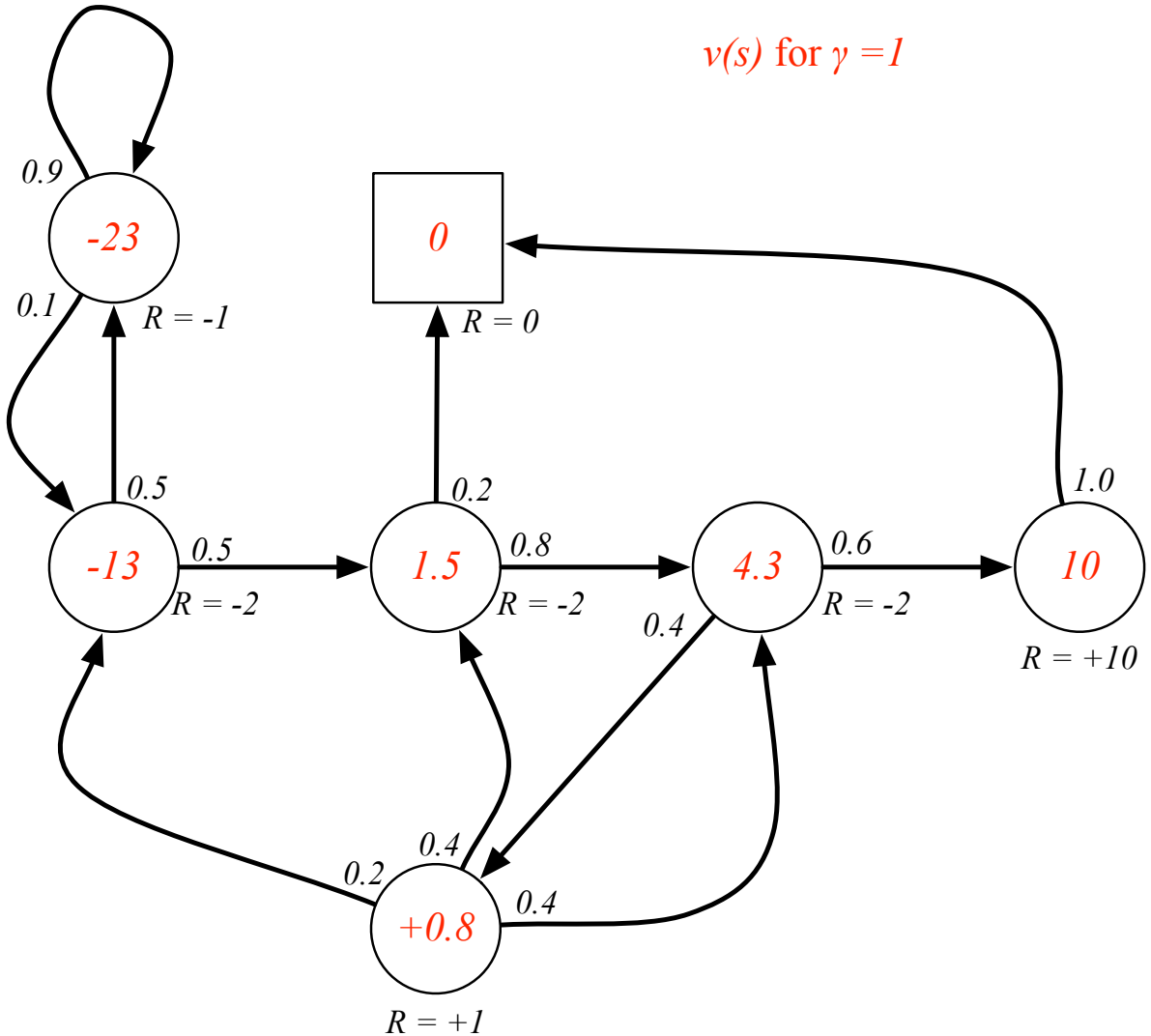
Example: State-Value Function for Student MRP (1)



Example: State-Value Function for Student MRP (2)



Example: State-Value Function for Student MRP (3)



Bellman Equation for MRPs

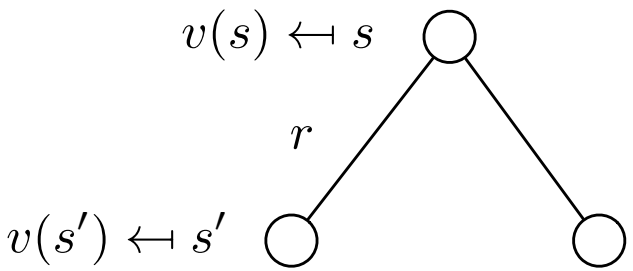
The value function can be decomposed into two parts:

- immediate reward R_{t+1}
- discounted value of successor state $\gamma v(S_{t+1})$

$$
\begin{aligned}
v(s) &= \mathbb{E}\left[G_t \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma\left(R_{t+2} + \gamma R_{t+3} + \ldots\right) \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma G_{t+1} \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s\right]
\end{aligned}
$$

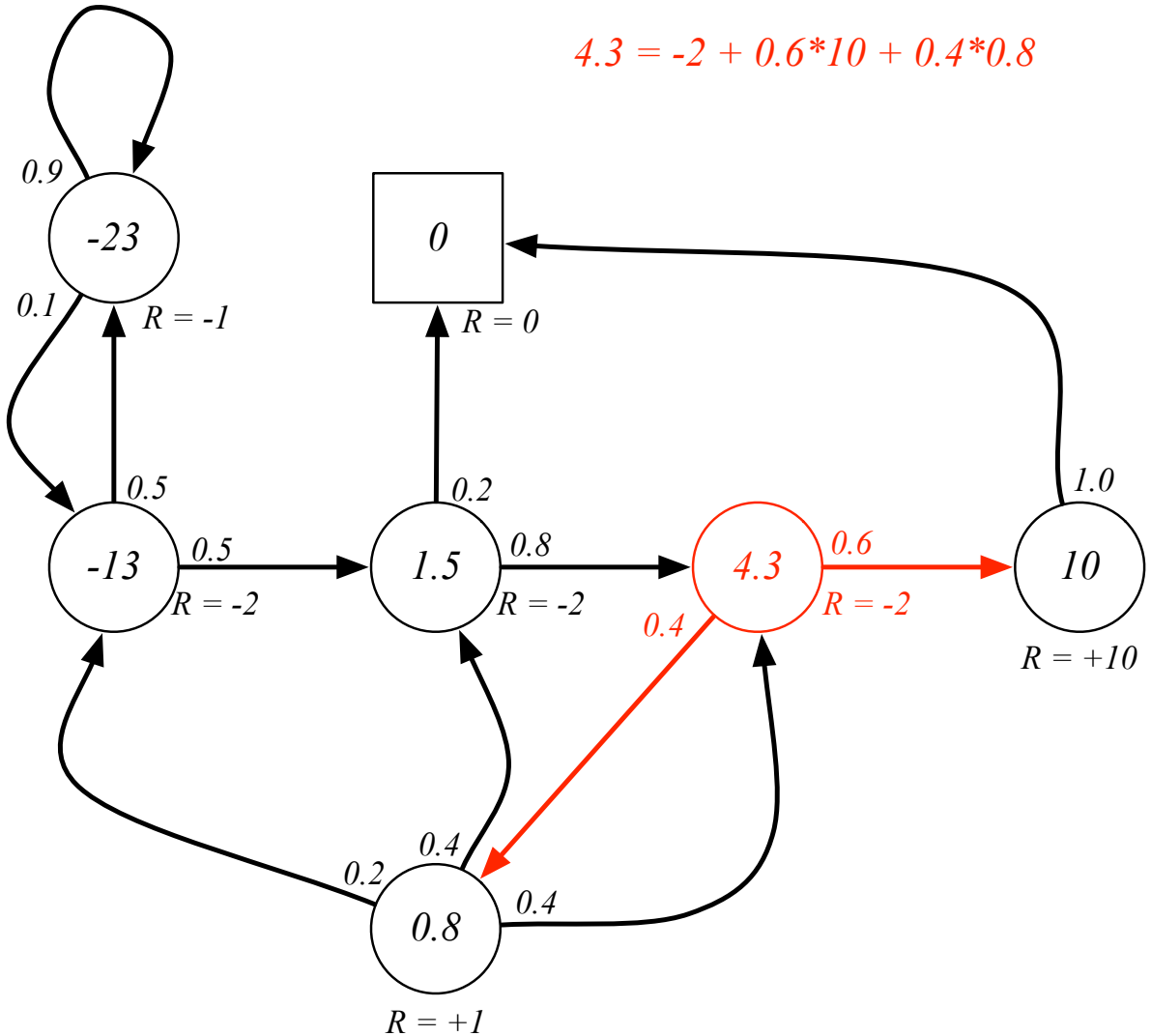
Bellman Equation for MRPs (2)

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



$$v(s) = \mathcal{R}_s + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'} v(s')$$

Example: Bellman Equation for Student MRP



Bellman Equation in Matrix Form

The Bellman equation can be expressed concisely using matrices,

$$v = \mathcal{R} + \gamma \mathcal{P} v$$

where v is a column vector with one entry per state

$$\begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix} = \begin{bmatrix} \mathcal{R}_1 \\ \vdots \\ \mathcal{R}_n \end{bmatrix} + \gamma \begin{bmatrix} \mathcal{P}_{11} & \dots & \mathcal{P}_{1n} \\ \vdots & & \\ \mathcal{P}_{11} & \dots & \mathcal{P}_{nn} \end{bmatrix} \begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix}$$

Solving the Bellman Equation

- The Bellman equation is a linear equation
- It can be solved directly:

$$
\begin{aligned}
v &= \mathcal{R} + \gamma \mathcal{P} v \\
(I - \gamma \mathcal{P})\, v &= \mathcal{R} \\
v &= (I - \gamma \mathcal{P})^{-1} \mathcal{R}
\end{aligned}
$$

- Computational complexity is $O(n^3)$ for n states
- Direct solution only possible for small MRPs
- There are many iterative methods for large MRPs, e.g.
  - Dynamic programming
  - Monte-Carlo evaluation
  - Temporal-Difference learning

Markov Decision Process

08/01/24

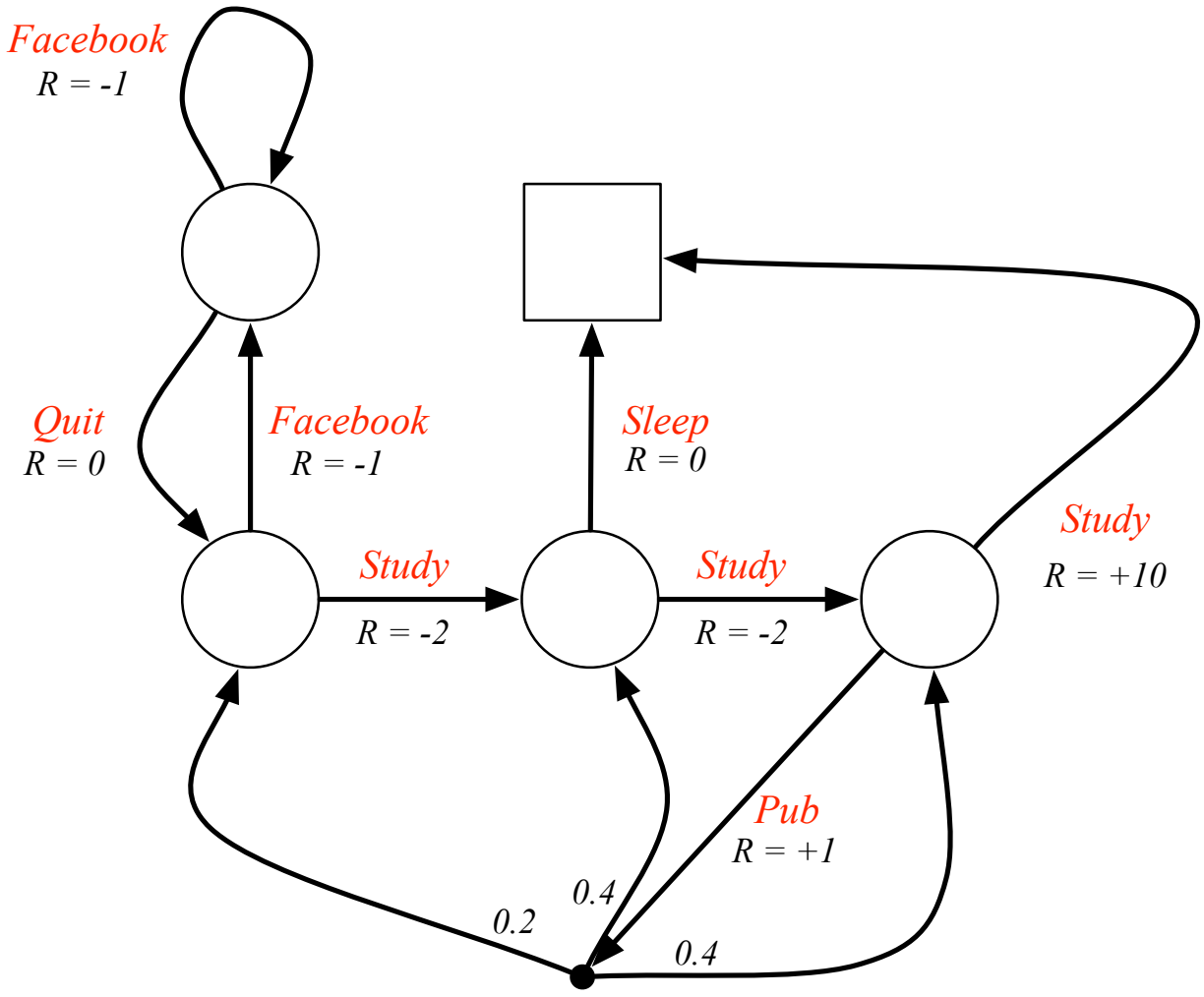
A Markov decision process (MDP) is a Markov reward process with decisions. It is an *environment* in which all states are Markov.

Definition

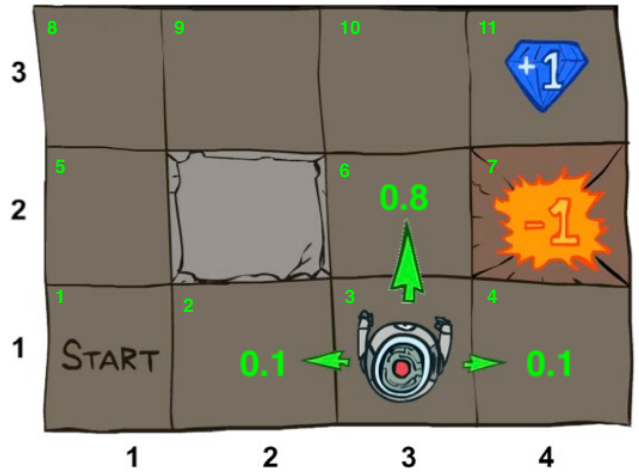
A *Markov Decision Process* is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- $\mathcal{S}$ is a finite set of states
- $\mathcal{A}$ is a finite set of actions
- $\mathcal{P}$ is a state transition probability matrix, $\mathcal{P}^{a}_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s, A_t = a\right]$
- $\mathcal{R}$ is a reward function, $\mathcal{R}^{a}_{s} = \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a\right]$
- γ is a discount factor $\gamma \in [0, 1]$.

Example: Student MDP



Formalizing *Get the Diamond*



- $S = \{s_1, s_2, \ldots, s_{11}\}$
- $A = \{U, D, L, R, S\}$
- Tr : 5 transition matrices
  - Example entires: $Tr(s_1, U, s_5) = 0.8$, $Tr(s_1, D, s_1,) = 0.9$
- $R(s, a) = -0.04 \quad \forall s \in S \setminus \{s_7, s_{11}\}$
- $R(s_7, *) = -1, R(s_1 1, *) = 1$

Policies (1)

Definition

A *policy* π is a distribution over actions given states,

$$\pi(a|s) = \mathbb{P}\left[A_t = a \mid S_t = s\right]$$

- A policy fully defines the behaviour of an agent
- MDP policies depend on the current state (not the history)
- i.e. Policies are *stationary* (time-independent),
  $A_t \sim \pi(\cdot|S_t), \forall t > 0$

Policies (2)

- Given an MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ and a policy π
- The state sequence $S_1, S_2, ...$ is a Markov process $\langle \mathcal{S}, \mathcal{P}^\pi \rangle$
- The state and reward sequence $S_1, R_2, S_2, ...$ is a Markov reward process $\langle \mathcal{S}, \mathcal{P}^\pi, \mathcal{R}^\pi, \gamma \rangle$
- where

$$\mathcal{P}^\pi_{s,s'} = \sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{P}^a_{ss'}$$

$$\mathcal{R}^\pi_s = \sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{R}^a_s$$

Value Function

Definition

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

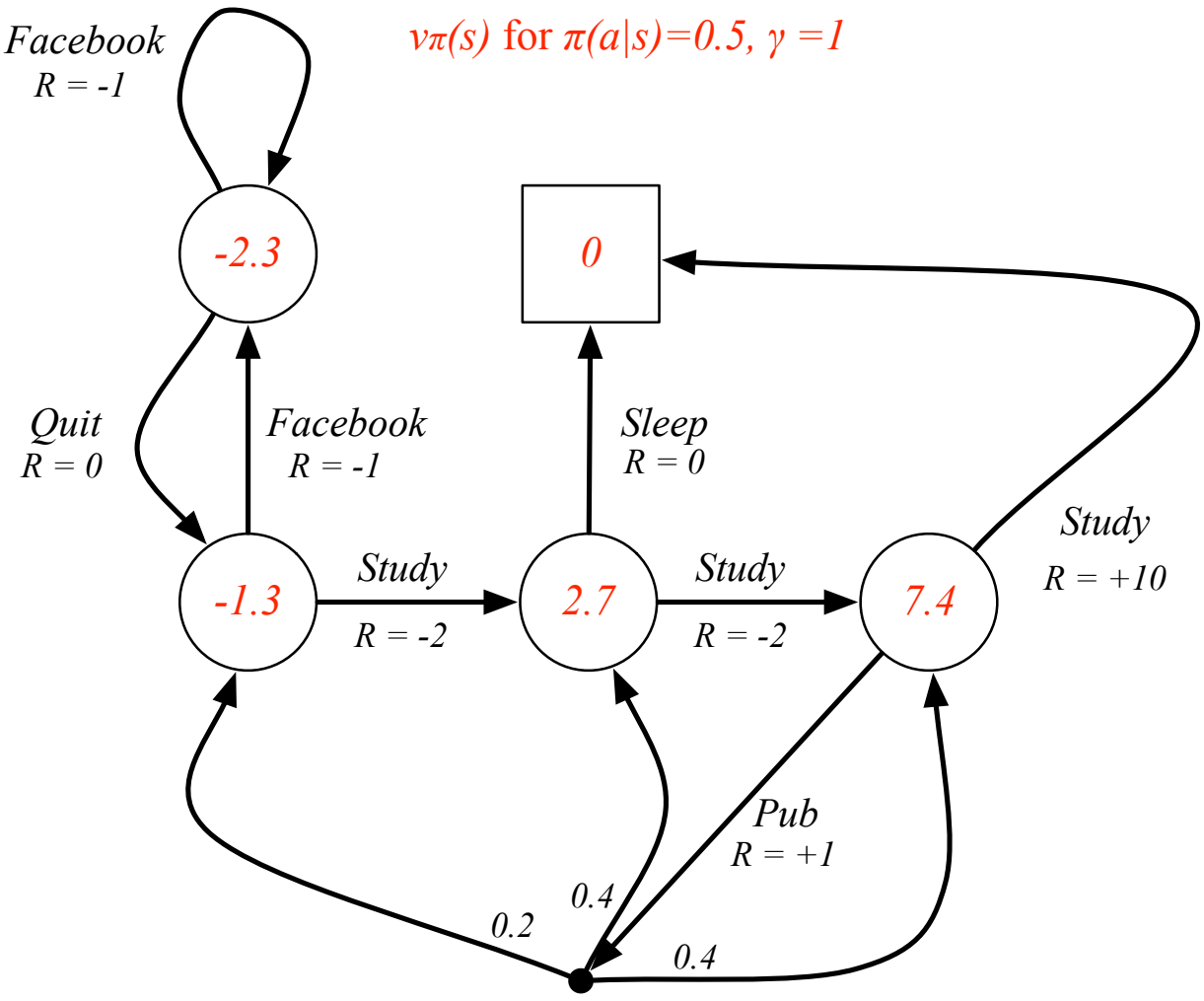
$$v_\pi(s) = \mathbb{E}_\pi \left[G_t \mid S_t = s \right]$$

Definition

The *action-value function* $q_\pi(s, a)$ is the expected return starting from state s, taking action a, and then following policy π

$$q_\pi(s, a) = \mathbb{E}_\pi \left[G_t \mid S_t = s, A_t = a \right]$$

Example: State-Value Function for Student MDP



Bellman Expectation Equation

The state-value function can again be decomposed into immediate reward plus discounted value of successor state,

$$v_\pi(s) = \mathbb{E}_\pi \left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s \right]$$

The action-value function can similarly be decomposed,

$$q_\pi(s, a) = \mathbb{E}_\pi \left[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a \right]$$

A_{t+1} is chosen by policy π and therefor $\rightarrow$ $V_\pi(S_{t+1})$

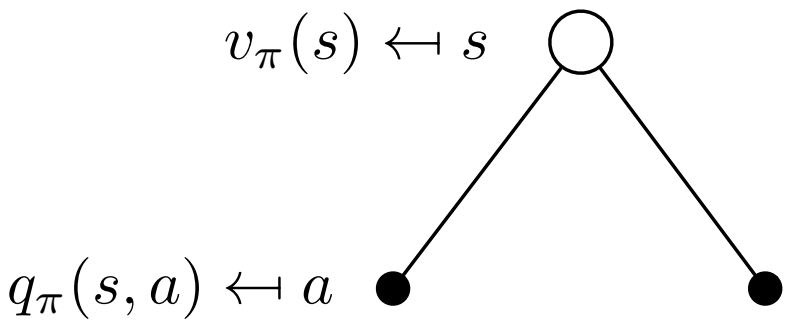
If: $\pi(s) = a'$

Then: $V_\pi(s) = q_\pi(s, a')$

But nothing guarantee that: $V_\pi(s) = q_\pi(s, a^*)$

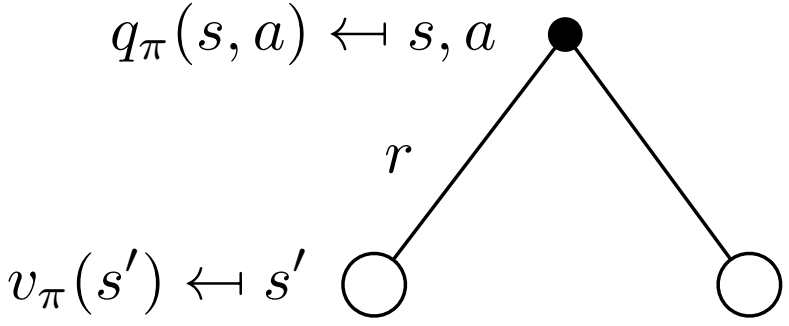
The relation between V_π and q_π is: $V_\pi(s) = q(s, \pi(s))$

Bellman Expectation Equation for V^{π}



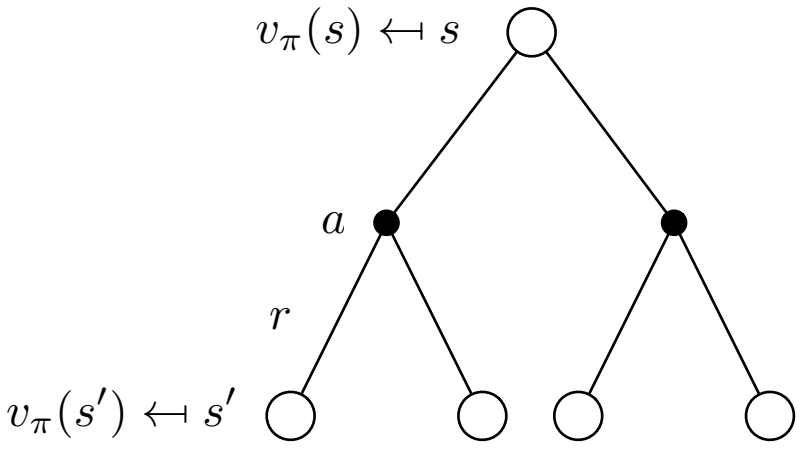
$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s,a)$$

Bellman Expectation Equation for Q^π



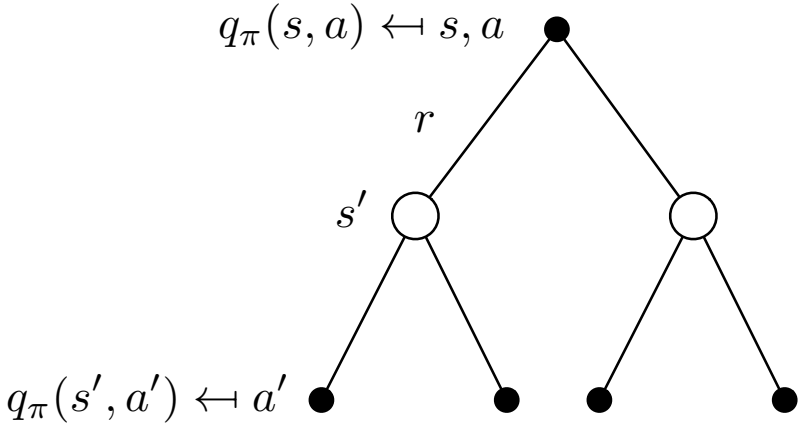
$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s')$$

Bellman Expectation Equation for v_π (2)



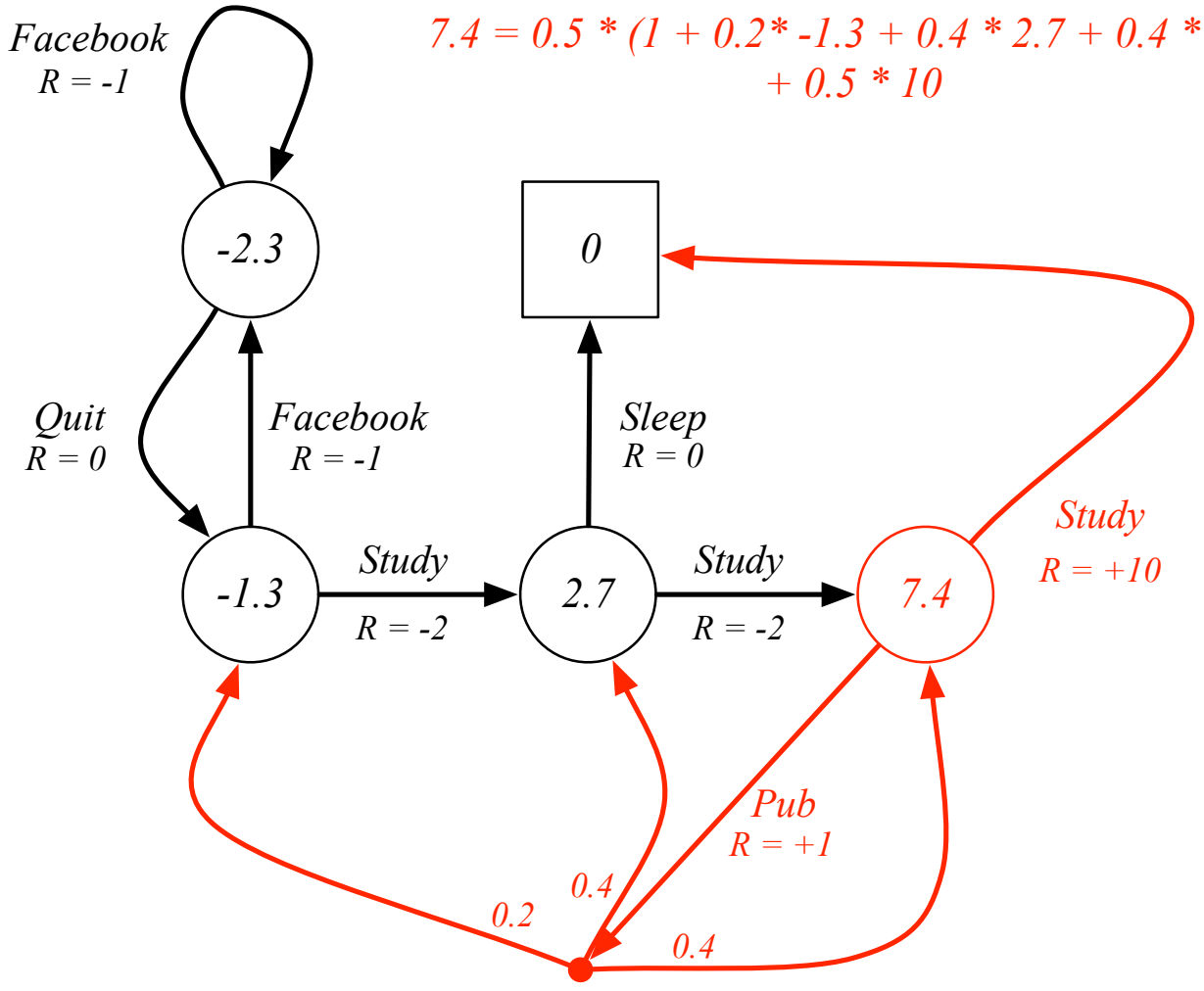
$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s') \right)$$

Bellman Expectation Equation for q_π (2)



$$q_\pi(s, a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \sum_{a' \in \mathcal{A}} \pi(a'|s') q_\pi(s', a')$$

Example: Bellman Expectation Equation in Student MDP



Bellman Expectation Equation (Matrix Form)

The Bellman expectation equation can be expressed concisely using the induced MRP,

$$v_\pi = \mathcal{R}^\pi + \gamma \mathcal{P}^\pi v_\pi$$

with direct solution

$$v_\pi = (I - \gamma \mathcal{P}^\pi)^{-1} \mathcal{R}^\pi$$

Optimal Value Function

Definition

The *optimal state-value function* $v_*(s)$ is the maximum value function over all policies

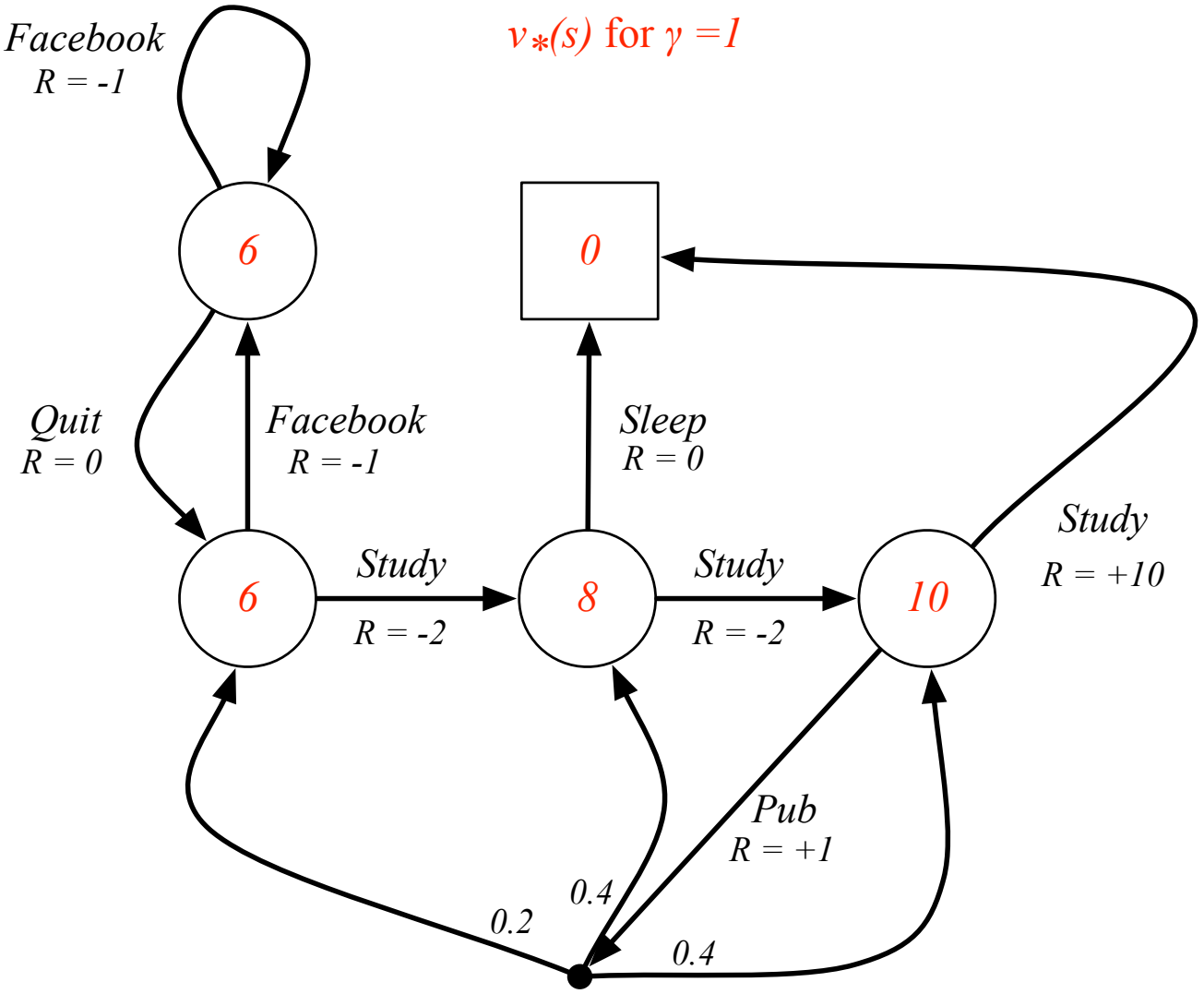
$$v_*(s) = \max_\pi v_\pi(s)$$

The *optimal action-value function* $q_*(s, a)$ is the maximum action-value function over all policies

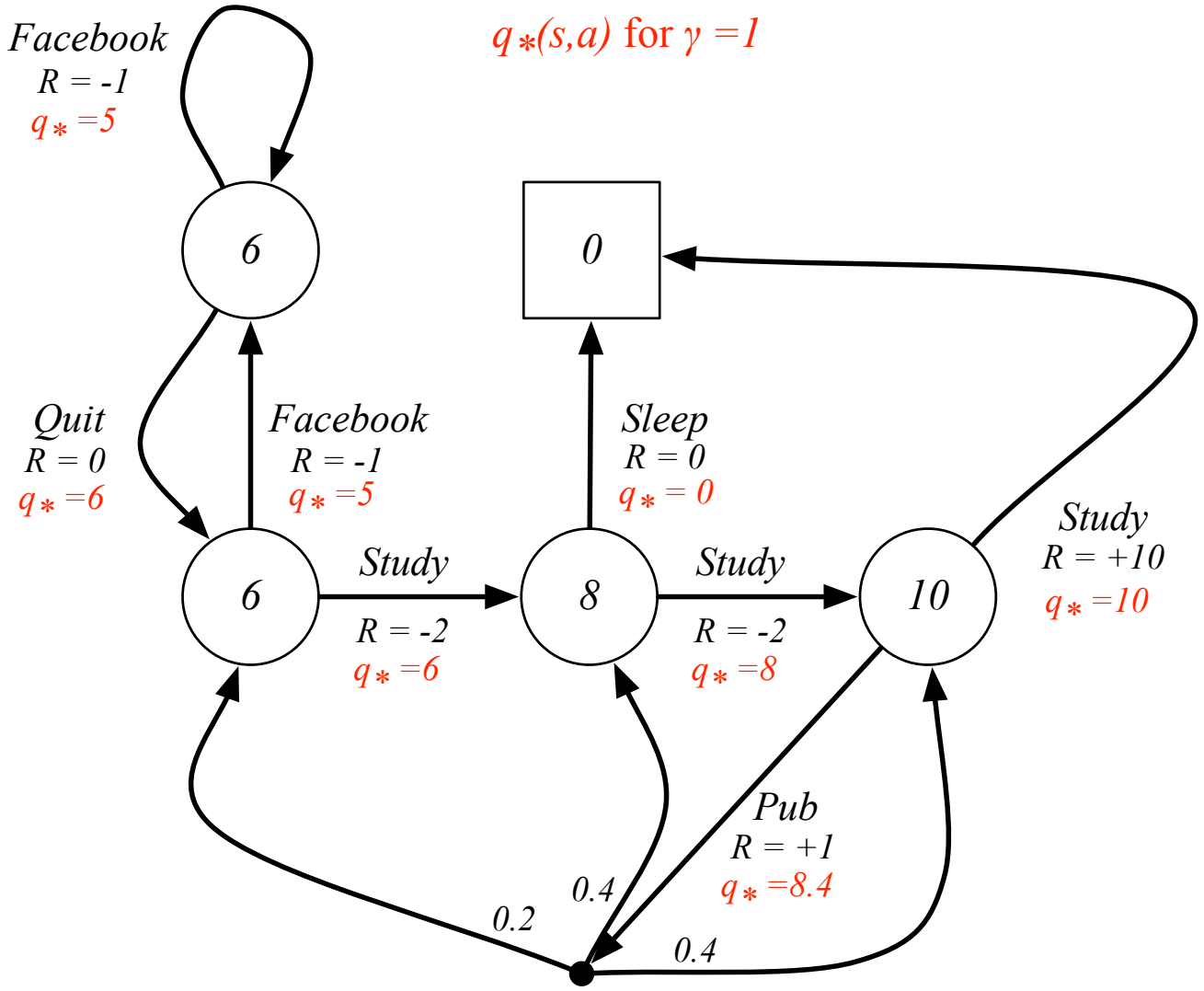
$$q_*(s, a) = \max_\pi q_\pi(s, a)$$

- The optimal value function specifies the best possible performance in the MDP.
- An MDP is "solved" when we know the optimal value fn.

Example: Optimal Value Function for Student MDP



Example: Optimal Action-Value Function for Student MDP



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, $q_{\pi_*}(s, a) = q_*(s, a)$*

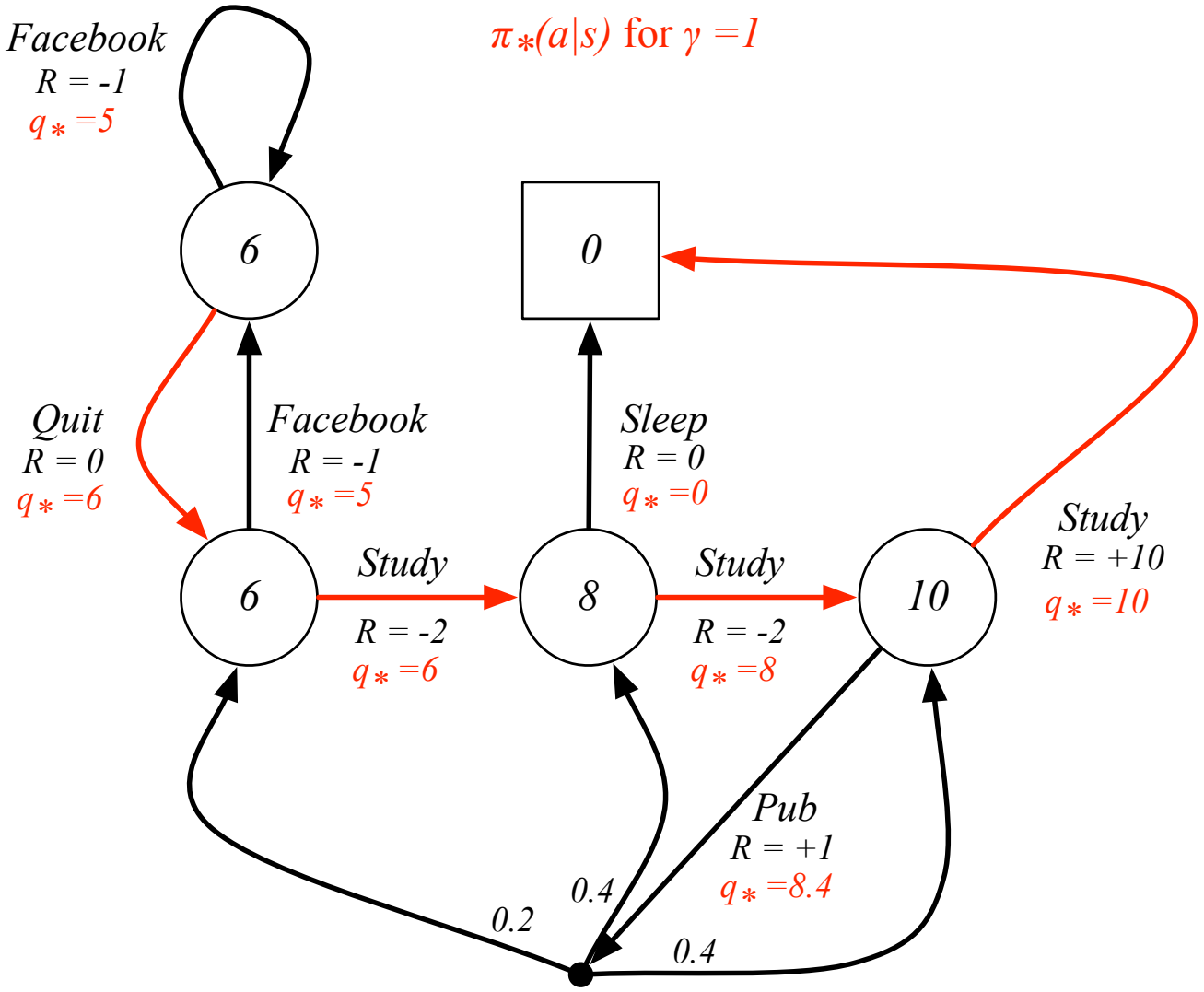
Finding an Optimal Policy

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

$$\pi_*(a|s) = \begin{cases} 1 & \text{if } a = \underset{a \in \mathcal{A}}{\text{argmax}}\ q_*(s, a) \\ 0 & \textit{otherwise} \end{cases}$$

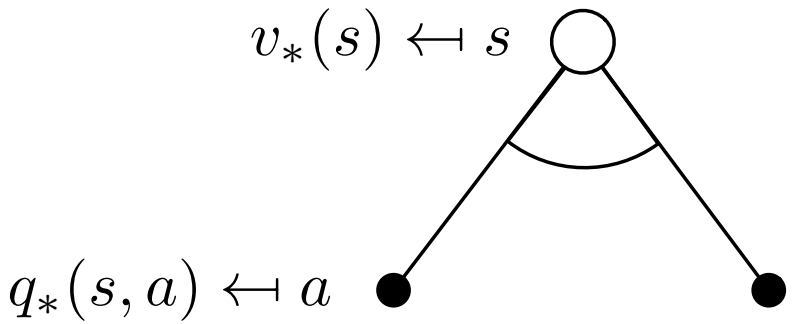
- There is always a deterministic optimal policy for any MDP
- If we know $q_*(s, a)$, we immediately have the optimal policy

Example: Optimal Policy for Student MDP



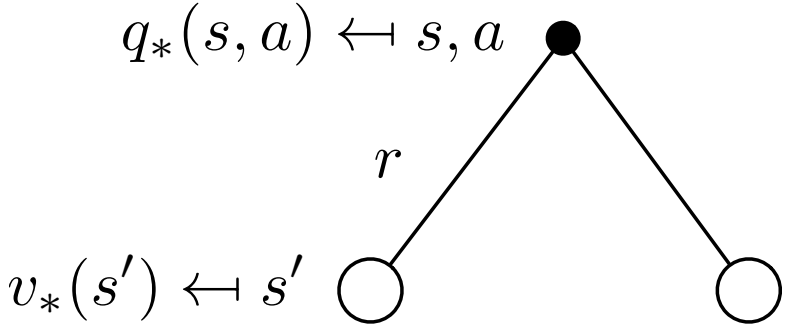
Bellman Optimality Equation for v_*

The optimal value functions are recursively related by the Bellman optimality equations:



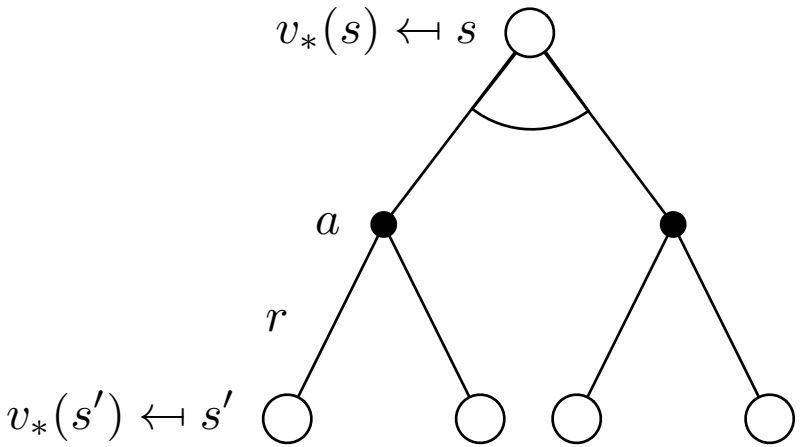
$$v_*(s) = \max_a q_*(s,a)$$

Bellman Optimality Equation for Q^*



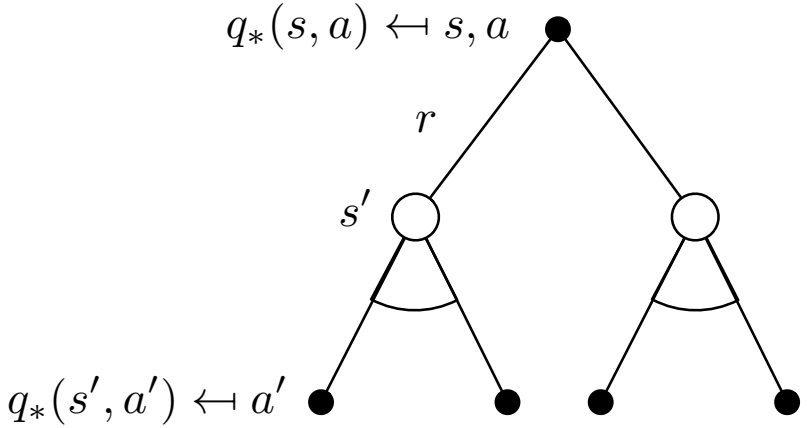
$$q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s')$$

Bellman Optimality Equation for V^* (2)



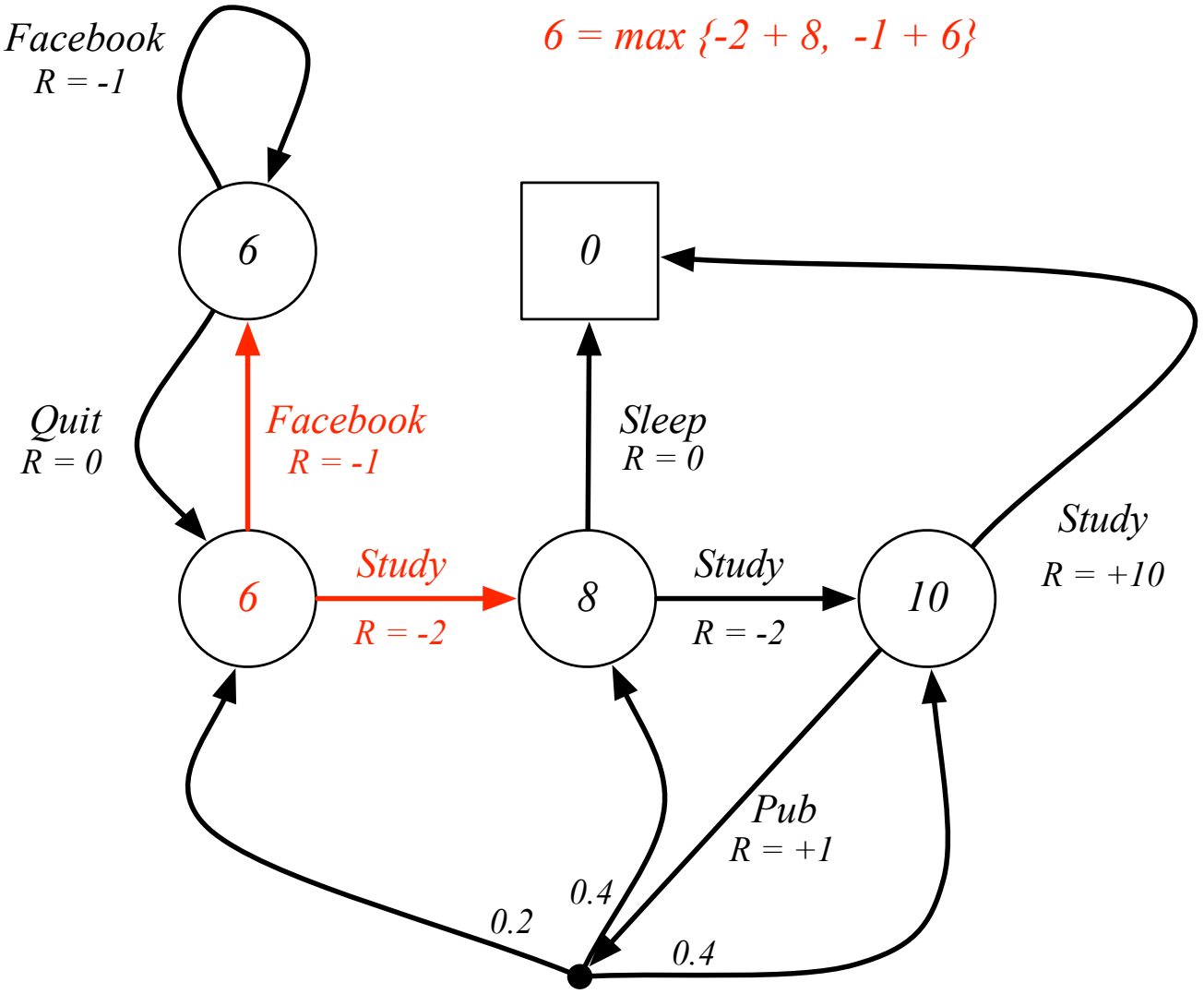
$$v_*(s) = \max_a \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s')$$

Bellman Optimality Equation for Q^* (2)



$$q_*(s, a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \max_{a'} q_*(s', a')$$

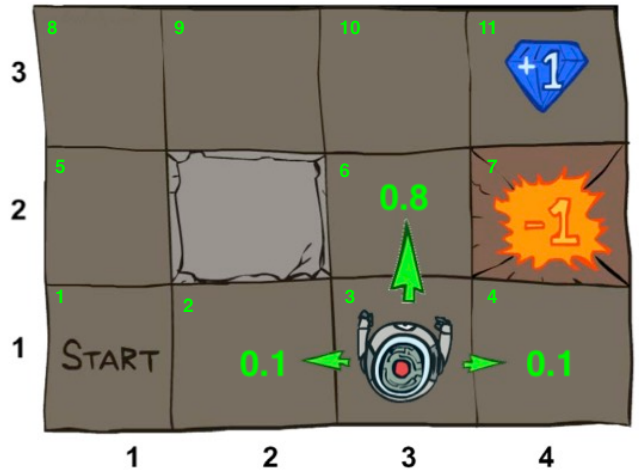
Example: Bellman Optimality Equation in Student MDP



Solving the Bellman Optimality Equation

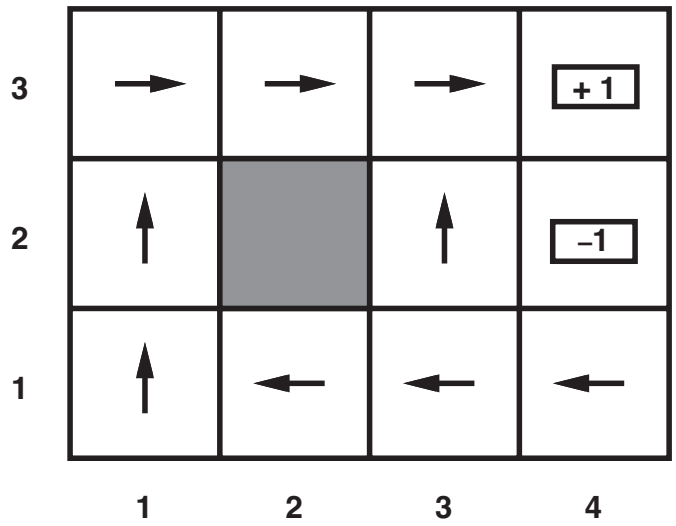
- Bellman Optimality Equation is non-linear
- No closed form solution (in general)
- Many iterative solution methods
  - Value Iteration
  - Policy Iteration
  - Q-learning
  - Sarsa

Formalizing *Get the Diamond*

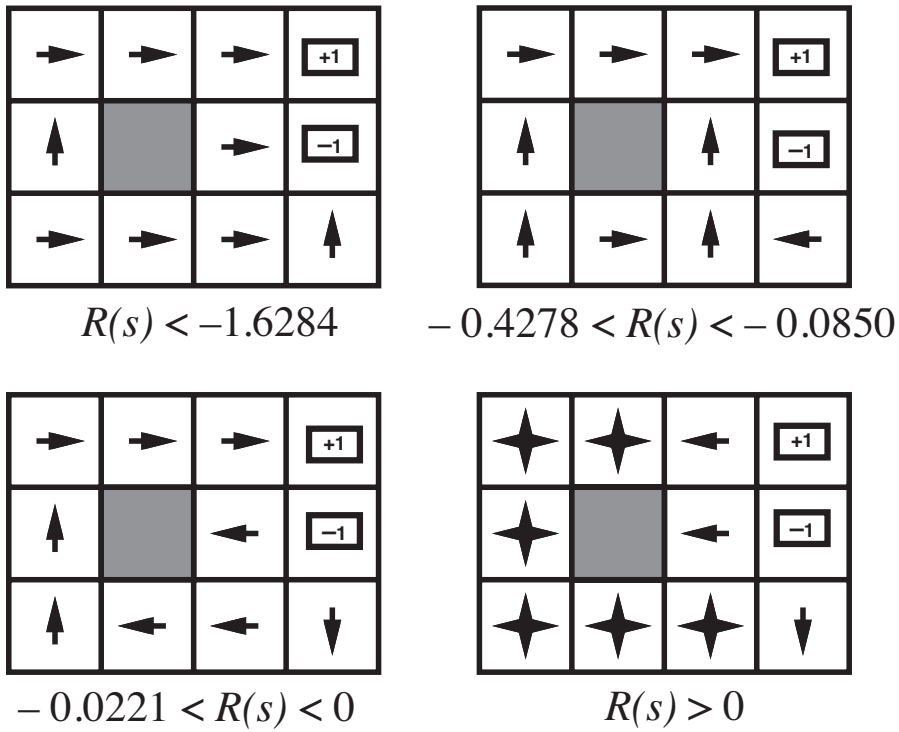


- $S = \{s_1, s_2, \ldots, s_{11}\}$
- $A = \{U, D, L, R, S\}$
- Tr : 5 transition matrices
  - Example entires: $Tr(s_1, U, s_5) = 0.8$, $Tr(s_1, D, s_1,) = 0.9$
- $R(s, a) = -0.04 \quad \forall s \in S \setminus \{s_7, s_{11}\}$
- $R(s_7, *) = -1, R(s_1 1, *) = 1$

Optimal Policies: *Get the Diamond*



(a)



(b)

R(s) = -0.04

Other values

Ergodic Markov Process (no exam)

An ergodic Markov process is

- *Recurrent*: each state is visited an infinite number of times
- *Aperiodic*: each state is visited without any systematic period

Theorem

An ergodic Markov process has a limiting stationary distribution $d^{\pi}(s)$ *with the property*

$$d^{\pi}(s) = \sum_{s' \in \mathcal{S}} d^{\pi}(s') \mathcal{P}_{s's}$$

Ergodic MDP (no exam)

Definition

An MDP is ergodic if the Markov chain induced by any policy is ergodic.

For any policy π, an ergodic MDP has an *average reward per time-step* ρ^π that is independent of start state.

$$\rho^\pi = \lim_{T \to \infty} \frac{1}{T} \mathbb{E}\left[\sum_{t=1}^{T} R_t\right]$$

POMDPs (no exam)

A Partially Observable Markov Decision Process is an MDP with hidden states. It is a hidden Markov model with actions.

Definition

A *POMDP* is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{P}, \mathcal{R}, \mathcal{Z}, \gamma \rangle$

- $\mathcal{S}$ is a finite set of states
- $\mathcal{A}$ is a finite set of actions
- $\mathcal{O}$ is a finite set of observations
- $\mathcal{P}$ is a state transition probability matrix, $\mathcal{P}^a_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s, A_t = a\right]$
- $\mathcal{R}$ is a reward function, $\mathcal{R}^a_s = \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a\right]$
- $\mathcal{Z}$ is an observation function, $\mathcal{Z}^a_{s'o} = \mathbb{P}\left[O_{t+1} = o \mid S_{t+1} = s', A_t = a\right]$
- γ is a discount factor $\gamma \in [0, 1]$.