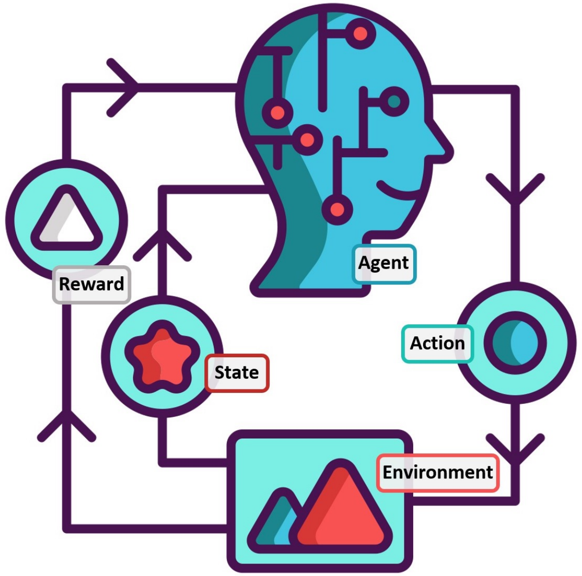
ARTIFICIAL INTELLIGENCE, MACHINE LEARNING AND CONTROL RESEARCH GROUP

Reinforcement Learning
2025/2026

RL2 Reinforcement Learning Research Lab @ UniPD

Lecture #04 Markov Decision Processes & Bellman Equations

Gian Antonio Susto



Announcements before starting

- The TA forgot to record the labs last week… :facepalm:
- We have uploaded last year recordings
- Please remind the TA to record the lab!
- Next lab tomorrow at 14:30 on bandits

Recap: Markov Decision Processes (MDPs)

Markov Decision Processes (MDPs) formally describe an environment for Reinforcement Learning

i. Markov Processes $\langle \mathcal{S}, \mathcal{P} \rangle$

ii. Markov Reward Processes $\langle \mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

iii. Markov Decision Processes $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- We have seen the formal introduction of state $\mathcal{S}$, transition probability $\mathcal{P}$

- We will see other elements (the reward function $\mathcal{R}$, return G, the value function v, the value function q, the discount factor γ, the action space $\mathcal{A}$)

Recap: Markov Decision Processes (MDPs)

Markov Decision Processes (MDPs) formally describe an environment for Reinforcement Learning

i. Markov Processes $\langle \mathcal{S}, \mathcal{P} \rangle$

ii. Markov Reward Processes $\langle \mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

iii. Markov Decision Processes $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- We have seen the formal introduction of state $\mathcal{S}$, transition probability $\mathcal{P}$

- We will see other elements (the reward function $\mathcal{R}$, return G, the value function v, the value function q, the discount factor γ, the action space $\mathcal{A}$)

MDPs are underlying formalization of a RL problem, but some of the elements $(\mathcal{P}, \mathcal{R})$ will not be known by the agent

Recap: Markov Decision Processes (MDPs)

Markov Decision Processes (MDPs) formally describe an environment for Reinforcement Learning

i. Markov Processes $\langle \mathcal{S}, \mathcal{P} \rangle$

ii. Markov Reward Processes $\langle \mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

iii. Markov Decision Processes $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- We have seen the formal introduction of state $\mathcal{S}$, transition probability $\mathcal{P}$

- We will see other elements (the reward function $\mathcal{R}$, return G, the value function v, the value function q, the discount factor γ, the action space $\mathcal{A}$)

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

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

Recap: Markov Decision Processes (MDPs)

Markov Decision Processes (MDPs) formally describe an environment for Reinforcement Learning

i. Markov Processes $\langle \mathcal{S}, \mathcal{P} \rangle$

ii. Markov Reward Processes $\langle \mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

iii. Markov Decision Processes $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- We have seen the formal introduction of state $\mathcal{S}$, transition probability $\mathcal{P}$

- We will see other elements (the reward function $\mathcal{R}$, return G, the value function v, the value function q, the discount factor γ, the action space $\mathcal{A}$)

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

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

$$\boxed{\mathcal{R}_s = \mathbb{E}\left[R_{t+1} | S_t = s\right]}$$

ii. Markov Reward Processes

Let's add rewards: a Markov Reward process is a Markov Chain with reward values

Definition

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

- $\mathcal{S}$ is a finite set of states
- $\mathcal{P}$ is a state transition probability matrix with entries

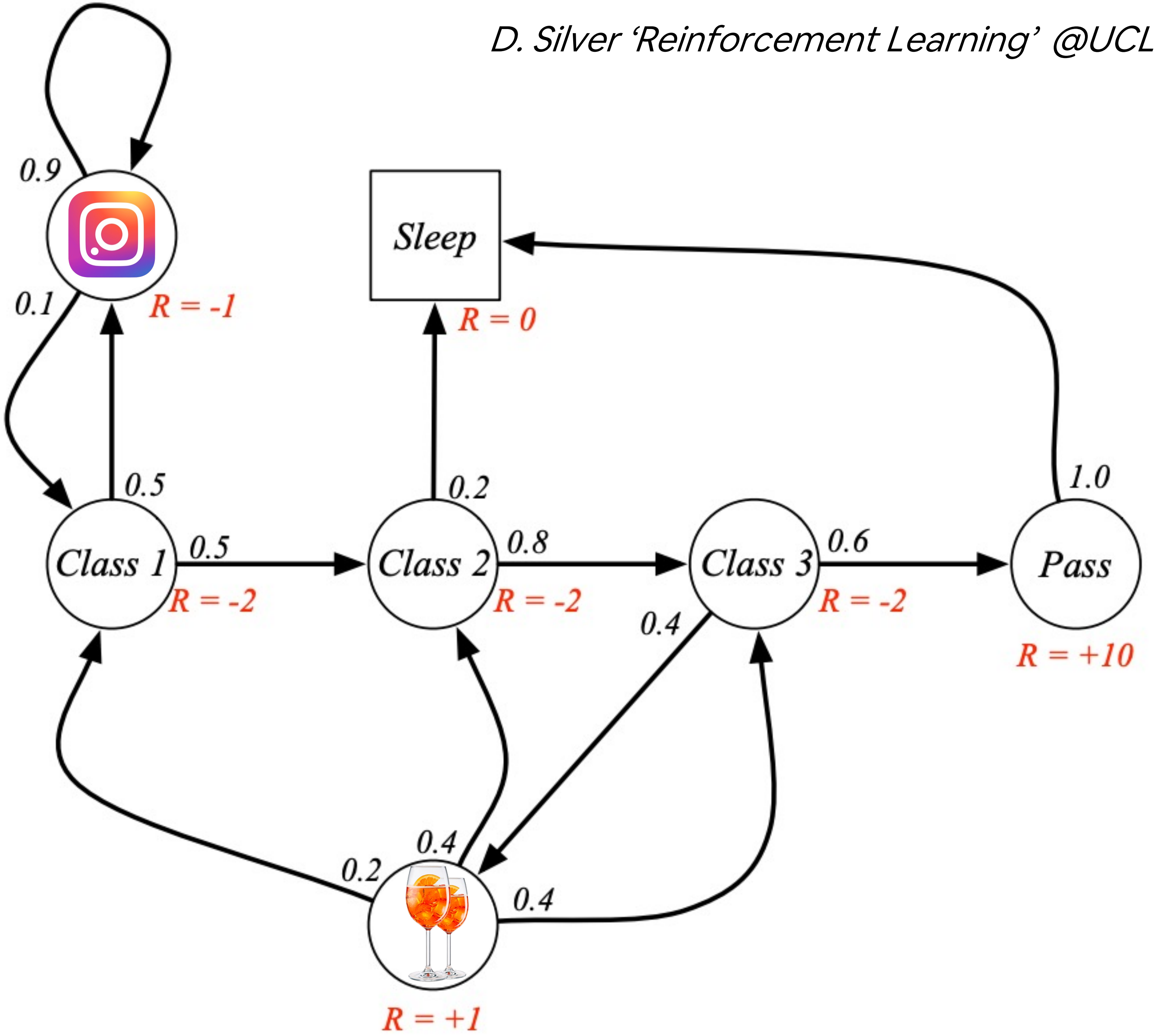
$$\mathcal{P}_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s\right]$$

- $\mathcal{R}$ is a reward function, $\mathcal{R}_s = \mathbb{E}\left[R_{t+1} | S_t = s\right]$ (it is just the immediate reward, in that specific state)
- γ is a discount factor, $\gamma \in [0,1]$

ii. Markov Reward Processes : Student Markov Chain

still no agency

D. Silver 'Reinforcement Learning' @UCL



ii. Markov Reward Processes: Return

Definition

The Return G_t is the total discounted reward from time-step t

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

where γ is a discount factor, $\gamma \in [0,1]$ (0 = myopic, 1 = far-sigthed)

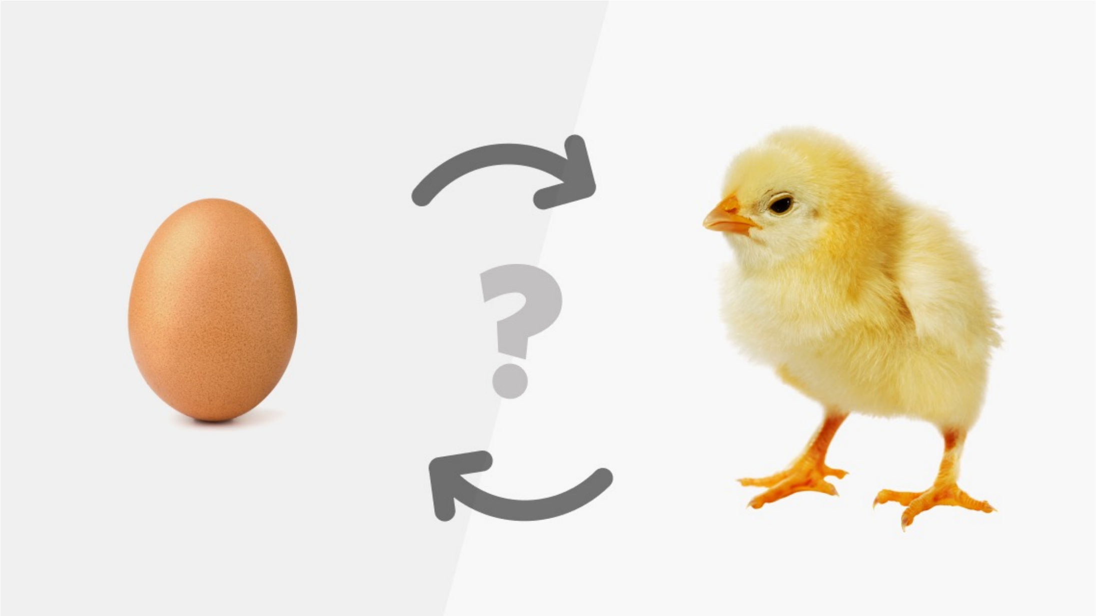
- In RL we are not interested in maximizing the value of a single step, but we want to maximize the return (return = goal of RL).
- The discount is the present value of future rewards:

$\gamma = 0$ is the 'myopic' case (we give value only to present reward)

$\gamma = 1$ is the 'far-sighted' case (all rewards are important, even if far away in the future)

ii. Markov Reward Processes: Return

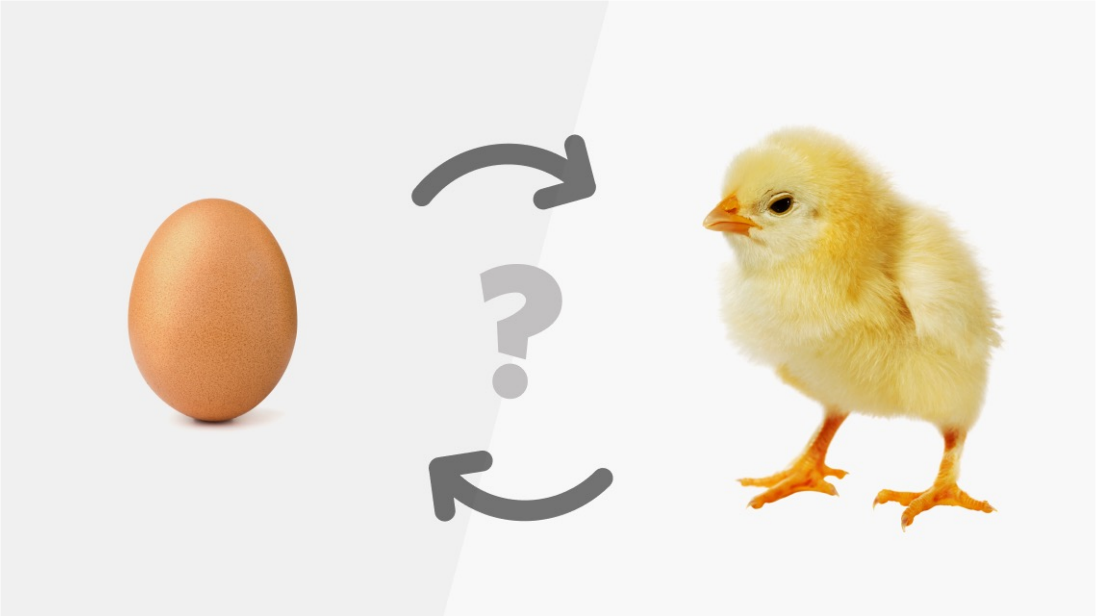
Why discounts?



ii. Markov Reward Processes: Return

Why discounts?

- Mathematically convenient: analysis can be simplified by the presence of rewards (for example we can avoid infinite returns in Markov Processes with cycles)
- Uncertainty about the future may not be fully represented
- Financial inspiration: immediate rewards may earn more interest than delayed rewards
- Animals and humans show preference for immediate rewards
- It is a general formulation: if $\gamma = 1$ we are considering the undiscounted Markov reward processes



ii. Markov Reward Processes: Value Function

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

Definition

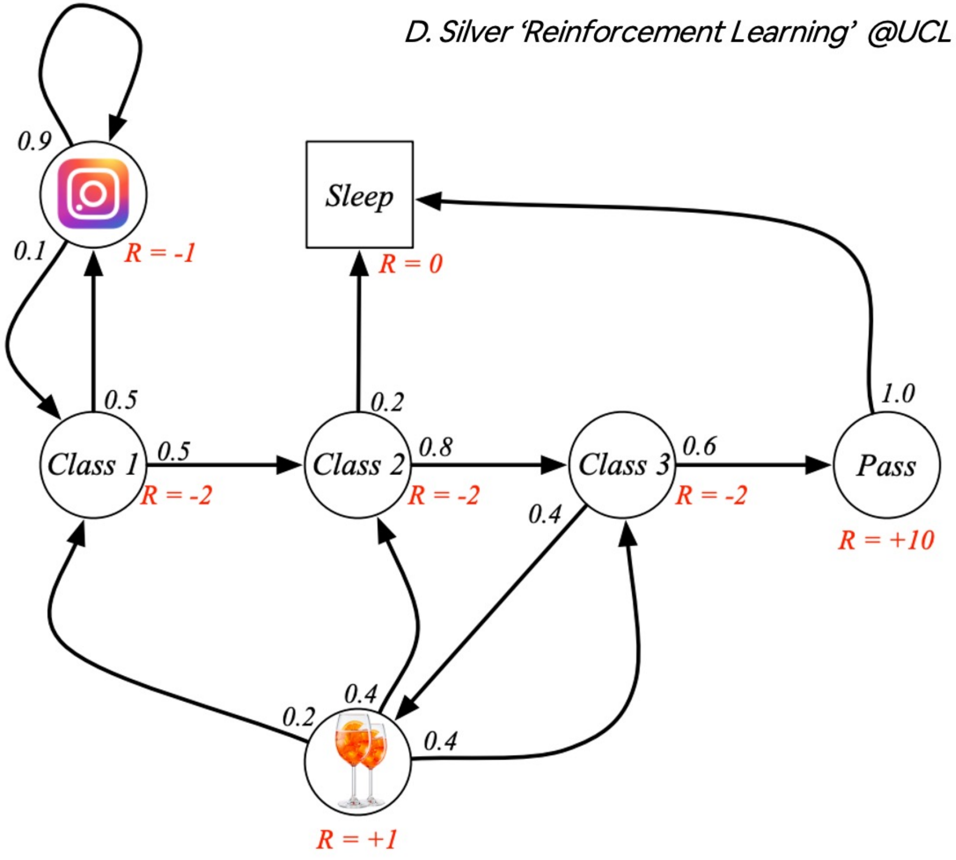
The state value function $v(s)$ of a Markov Reward Process is the expected return starting from state s

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

Please note the expectation: this is fundamental since we are in stochastic settings

ii. Markov Reward Processes : Student Markov Chain

D. Silver 'Reinforcement Learning' @UCL



Preview of Chapter 5 - How to compute state value functions from a Markov Reward Process?

ii. Markov Reward Processes : Student Markov Chain

Let's consider $\gamma = 1/2$ and the return obtain with the available samples

C1 C2 C3 Pass Sleep -> v(C1) = -2-2/2-2/4+10/8 = -2.25

C1 IG IG C1 C2 Sleep -> v(C1) = -2-1/2-1/4-2/8-2/16 = -3.125

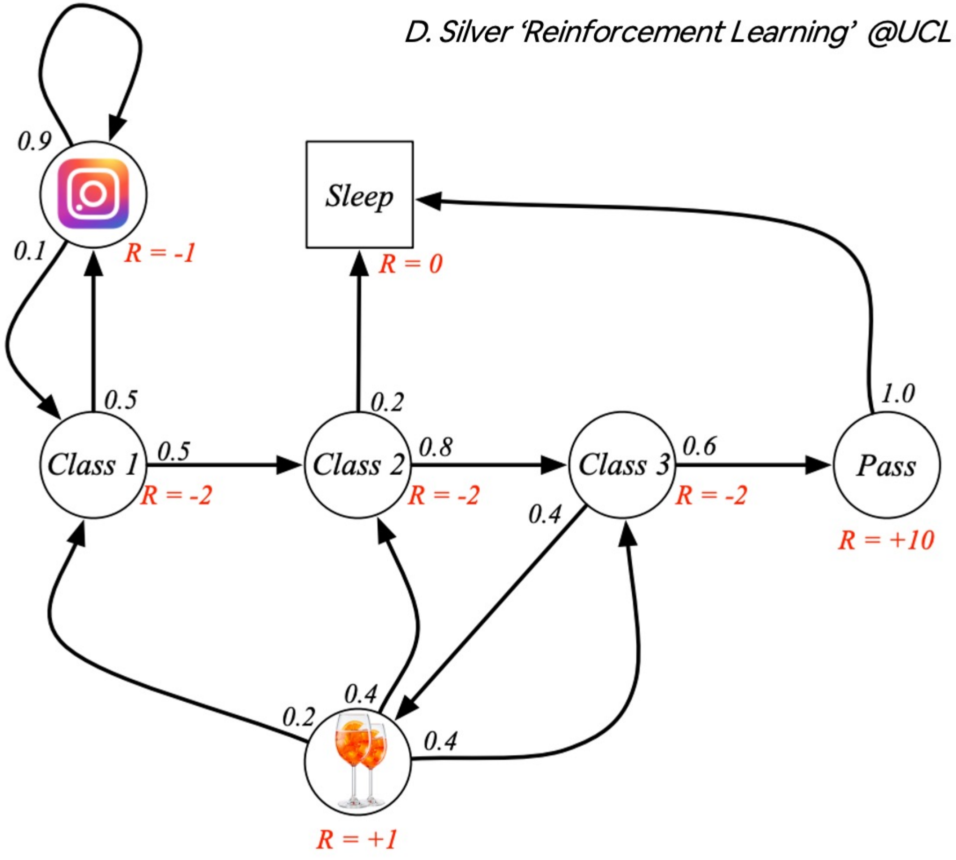
C1 C2 C3 Spritz C2 C3 Pass Sleep ->

v(C1) = -2-2/2-2/4+1/8-2/16-2/32+10/64 = -3.41

C1 IG IG C1 C2 C3 Spritz C1 IG IG IG C1 C2 C3 Spritz C2 Sleep

v(C1) = -2-1/2-1/4-2/8-2/16+ ... = -3.20

D. Silver 'Reinforcement Learning' @UCL



ii. Markov Reward Processes : Student Markov Chain

Let’s consider $\gamma = 1/2$ and the return obtain with the available samples

C1 C2 C3 Pass Sleep -> v(C1) = -2-2/2-2/4+10/8 = -2.25

C1 IG IG C1 C2 Sleep -> v(C1) = -2-1/2-1/4-2/8-2/16 = -3.125

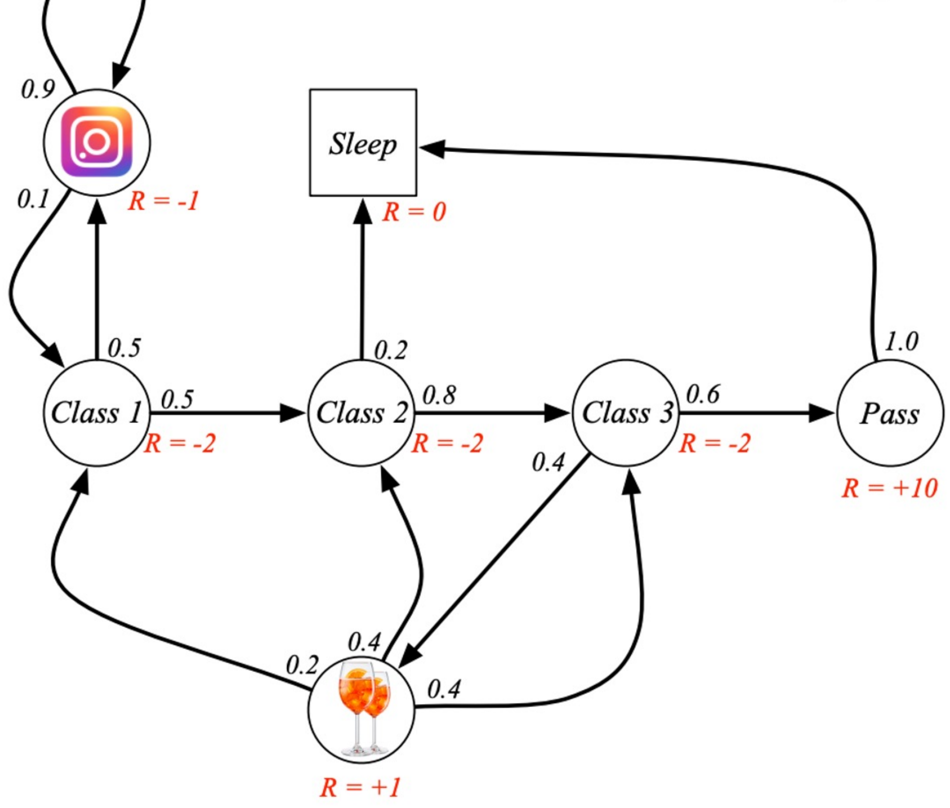
C1 C2 C3 Spritz C2 C3 Pass Sleep ->

v(C1) = -2-2/2-2/4+1/8-2/16-2/32+10/64 = -3.41

C1 IG IG C1 C2 C3 Spritz C1 IG IG IG C1 C2 C3 Spritz C2 Sleep

v(C1) = -2-1/2-1/4-2/8-2/16+ … = -3.20

D. Silver ‘Reinforcement Learning’ @UCL

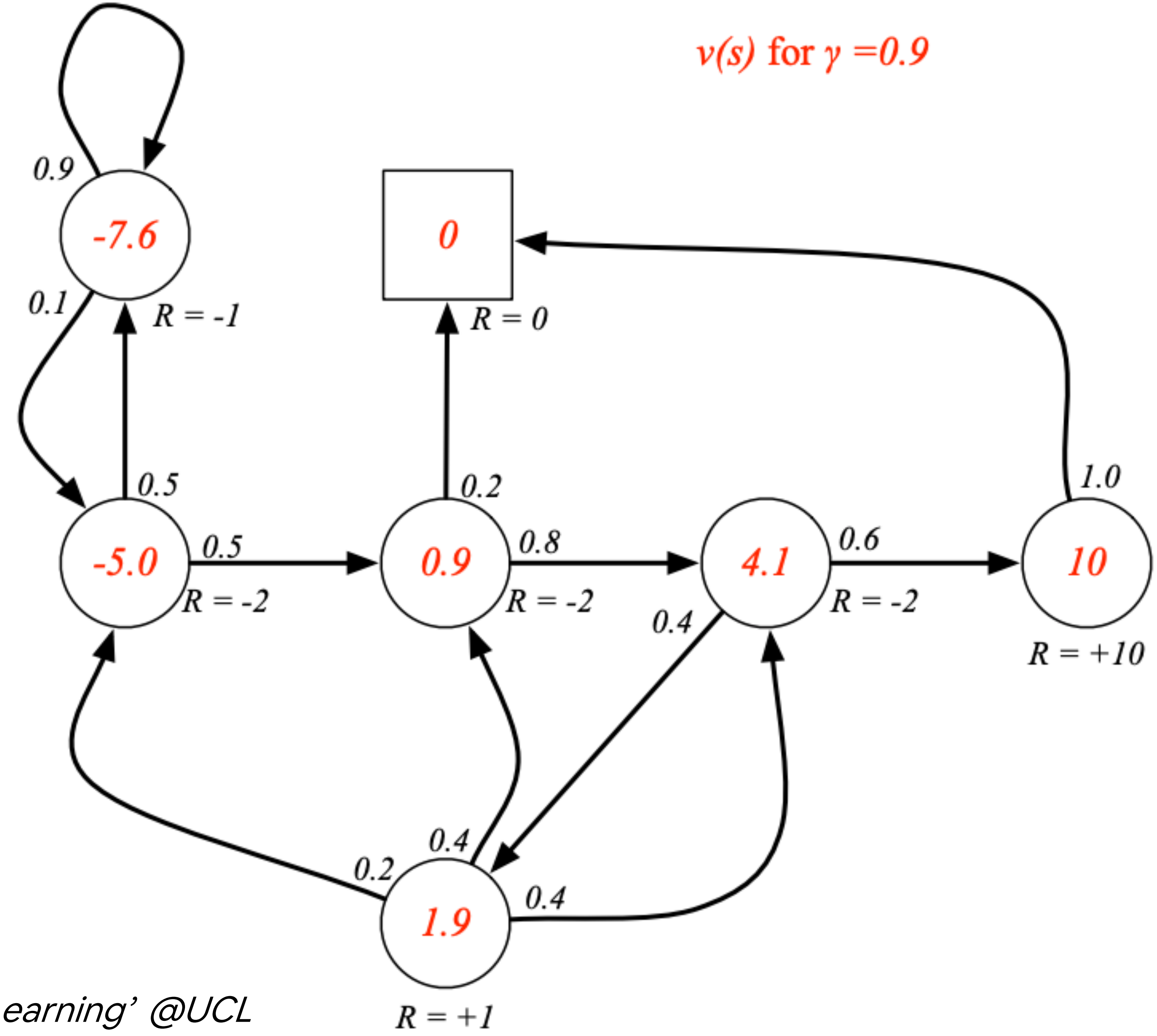


If we have samples, an estimate of the value function for state s is provided by the sampled average of the returns seen from that state.

Ie. In this case v(C1) = (-2.25-3.125-3.41-3.20)/4 = -3

ii. Markov Reward Processes : Student Markov Chain

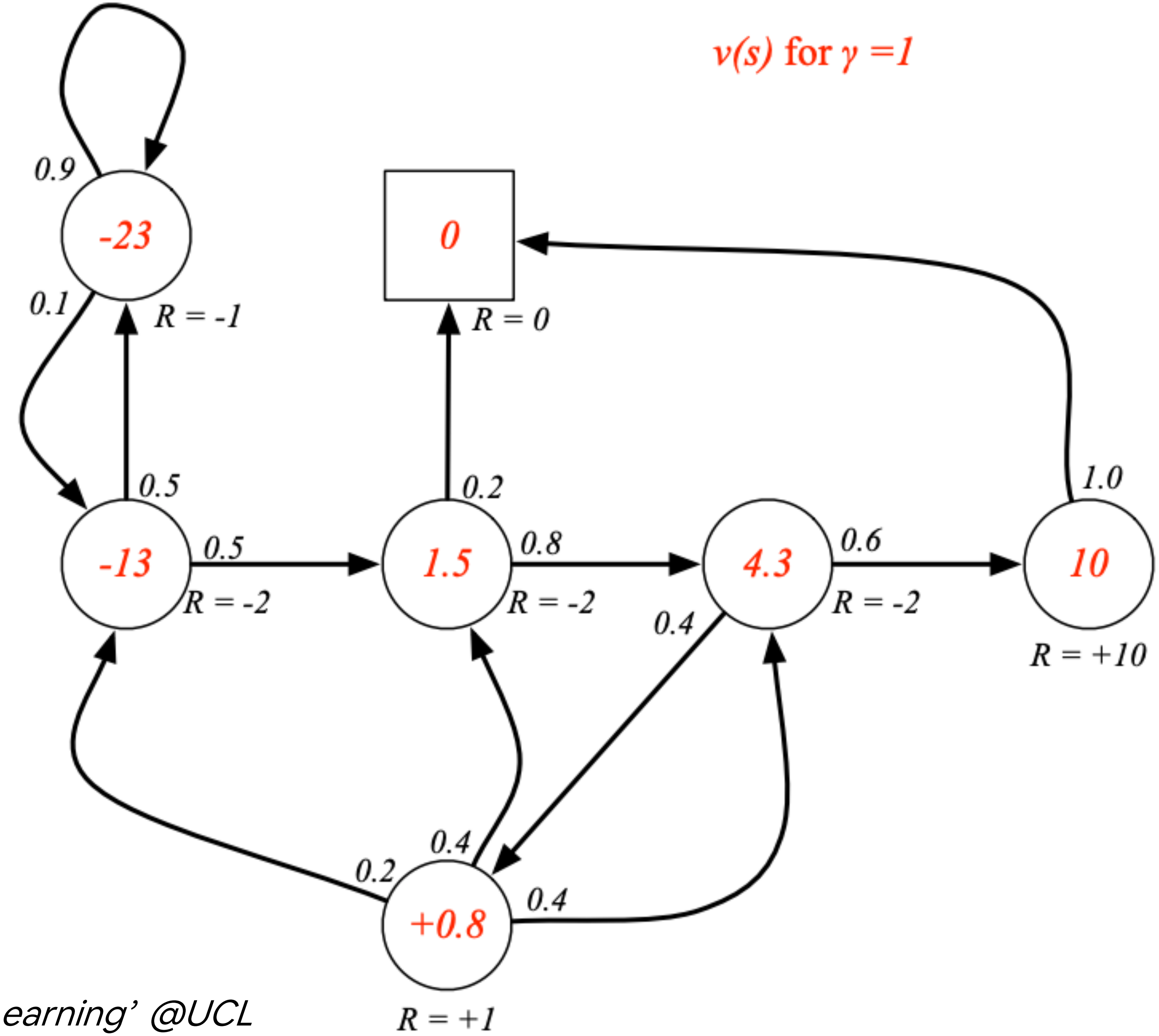
Different discounts values = different state values functions!



D. Silver 'Reinforcement Learning' @UCL

ii. Markov Reward Processes : Student Markov Chain

Different discounts values = different state values functions!



D. Silver 'Reinforcement Learning' @UCL

The Bellman Equations

Richard E. Bellman

Bellman equations are our tool to 'solve' Markov Reward Processes (MRPs) and MDPs thanks to their recursive nature:

MRP	Bellman equation: for finding value functions	Linear: we can use it for 'small' MRPs. We need to resort to iterative approaches for 'large' MRPs
MDP	Bellman expectation equation: for finding value functions and action-value functions	Linear: we can use it for small MDPs. We need to resort to iterative approaches for 'large' MDPs
MDP	Bellman optimality equation: for finding optimal value functions and optimal action-value functions	Non-linear: we need iterative approaches even for small MDPs.

Richard E. Bellman

The Bellman Equations

Bellman equations are our tool to 'solve' Markov Reward Processes (MRPs) and MDPs thanks to their recursive nature:

Bellman equations are fundamental tools to understand *v* in MRPs and *(v,q)* in MDPs

When we will consider the 'true' RL problems, the algorithms that we will use, exploti Bellman equations

		Linear: we can use it for 'small' MRPs. We need to resort to iterative approaches for 'large' MRPs
	e	Linear: we can use it for small MDPs. We need to resort to iterative approaches for 'large' MDPs
	al value	Non-linear: we need iterative approaches even for small MDPs.

The Bellman Equations

Richard E. Bellman

Bellman equations are our tool to 'solve' Markov Reward Processes (MRPs) and MDPs thanks to their recursive nature:

MRP	Bellman equation: for finding value functions	Linear: we can use it for 'small' MRPs. We need to resort to iterative approaches for 'large' MRPs
MDP	Bellman expectation equation: for finding value functions and action-value functions	Linear: we can use it for small MDPs. We need to resort to iterative approaches for 'large' MDPs
MDP	Bellman optimality equation: for finding optimal value functions and optimal action-value functions	Non-linear: we need iterative approaches even for small MDPs.

ii. Markov Reward Processes: Bellman Equation

The value function $v(s)$ can be decomposed into 2 parts:

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

Exploiting

$$\begin{aligned} G_t &\doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\ &= R_{t+1} + \gamma \left(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \cdots \right) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

ii. Markov Reward Processes: Bellman Equation

The value function $v(s)$ can be decomposed into 2 parts:

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

Exploiting

$$\begin{aligned} G_t &\doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\ &= R_{t+1} + \gamma \left(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \cdots \right) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

We have that (law of iterated expectations)

$$\begin{aligned} v(s) = \mathbb{E}[G_t | S_t = s] &= \mathbb{E}[R_{t+1} + \gamma G_{t+1} | S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma v(S_{t+1}) | S_t = s] \end{aligned}$$

ii. Markov Reward Processes: Bellman Equation

The value function $v(s)$ can be decomposed into 2 parts:

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

Exploiting

$$\begin{aligned} G_t &\doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\ &= R_{t+1} + \gamma\left(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \cdots\right) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

We have that (law of iterated expectations)

$$\begin{aligned} v(s) &= \mathbb{E}[G_t | S_t = s] = \mathbb{E}[R_{t+1} + \gamma G_{t+1} | S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma v(S_{t+1}) | S_t = s] \end{aligned}$$

The value of a state is in relationship with the next one!

We will use 'iterative definitions' many times throughout the course!

ii. Markov Reward Processes: Bellman Equation

The value function $v(s)$ can be decomposed into 2 parts:

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

Pay Attention: we will have 3 different 'Bellman Equations' today

Exploiting

$$\begin{aligned} G_t &\doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\ &= R_{t+1} + \gamma\left(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \cdots\right) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

We have that (law of iterated expectations)

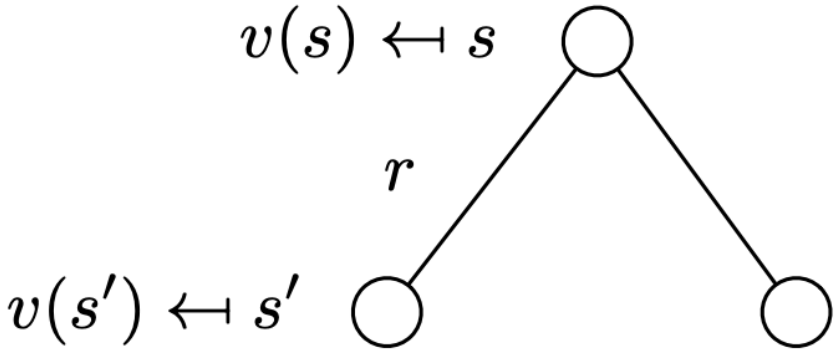
$$\begin{aligned} v(s) = \ &\mathbb{E}[G_t | S_t = s] = \ \mathbb{E}[R_{t+1} + \gamma G_{t+1} | S_t = s] \\ = \ &\mathbb{E}[R_{t+1} + \gamma v(S_{t+1}) | S_t = s] \end{aligned}$$

ii. Markov Reward Processes: Bellman Equation

The definition of the value-function from the Bellman Equation can be seen a 1-step look ahead search

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

Backup diagrams: visual representations of different algorithms and models in RL

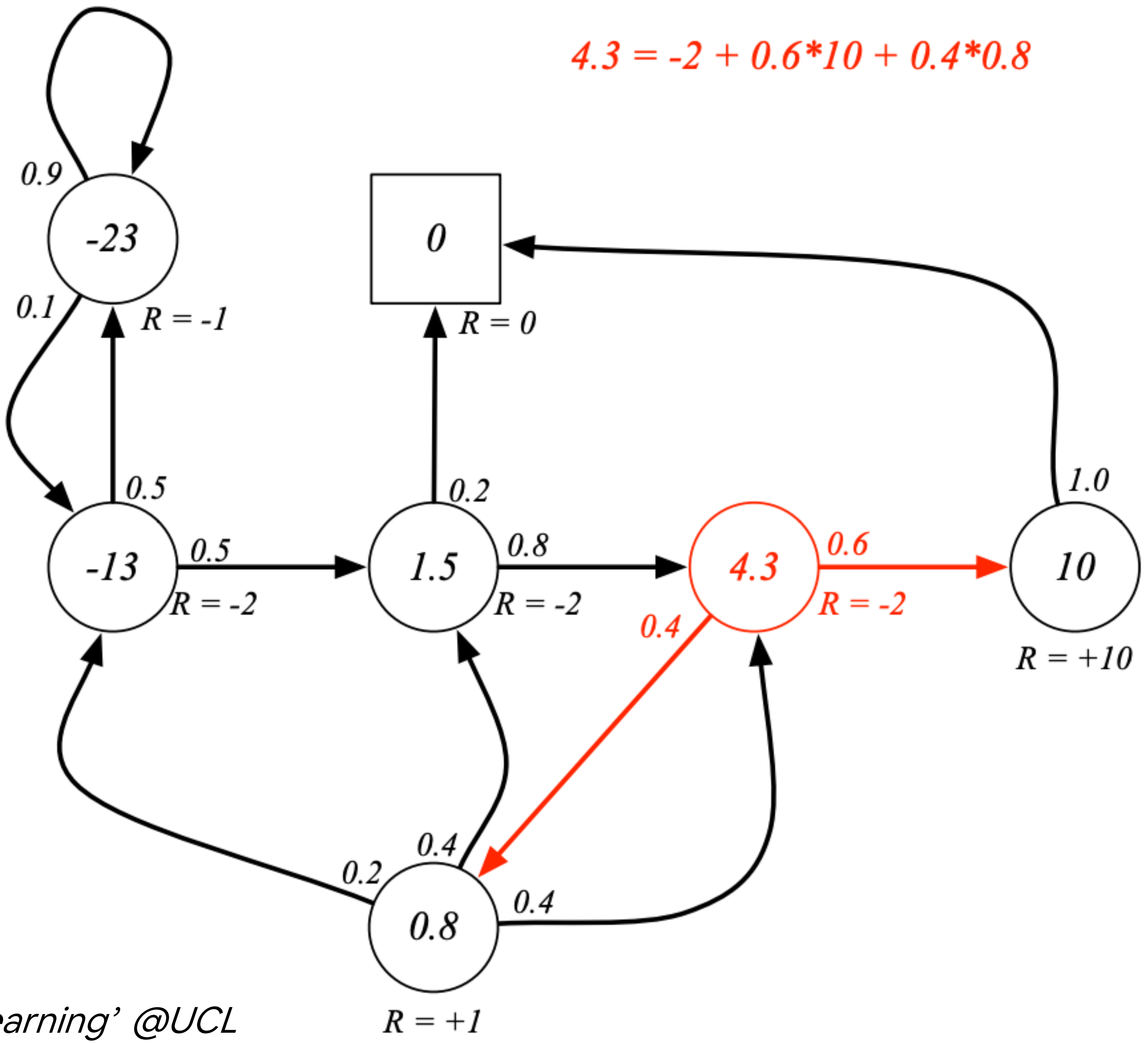


We need to consider all possible successor states with the related transition probabilities

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

ii. Markov Reward Processes : Student Markov Chain

- Undiscounted case (γ = 1)
- If state value functions are provided we can use the Bellman Equation to verify if they are true



D. Silver 'Reinforcement Learning' @UCL

ii. Markov Reward Processes: Bellman Equation in Matrix Form

The Bellman Equation can be written concisely in matrix form

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

Where v is a column vector with one entry per state

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

ii. Markov Reward Processes: Solving the Bellman Equation

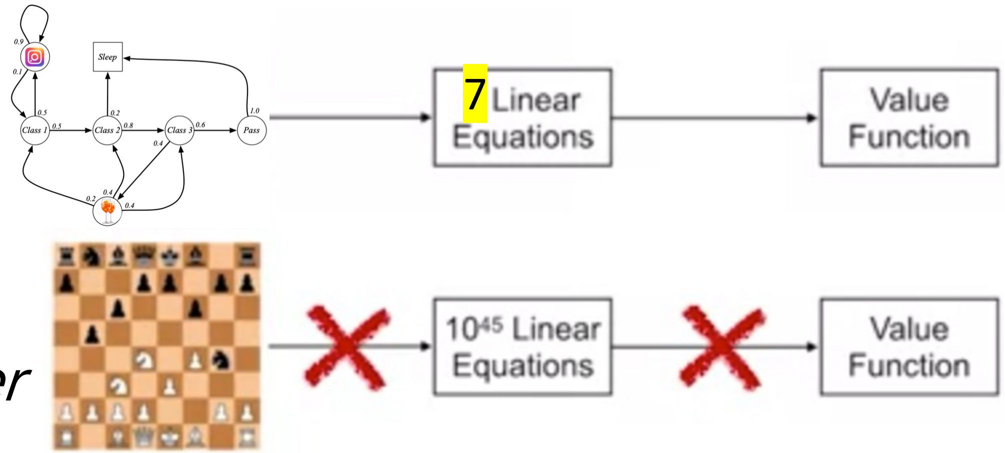
- The Bellman Equation (when we are dealing with Markov Reward Processes*) is linear
- We can solve the previous matrix directly
- If we have n states, the computational cost is $O[n^3]$: affordable only with small Markov Reward Processes (MRPs)
- For large MRPs we will look for efficient methods (Dynamic Programming, Monte-Carlo evaluation, Temporal-Difference learning)

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

$$(I - \gamma \mathcal{P})\, v = \mathcal{R}$$

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

This will not be true when we will deal with optimization and maximization (when we will consider an agent making decisions!) in Markov Decision Processes



A. White, M. White 'Fundamentals of Reinforcement Learning'

ii. Markov Reward Processes: Solving the Bellman Equation

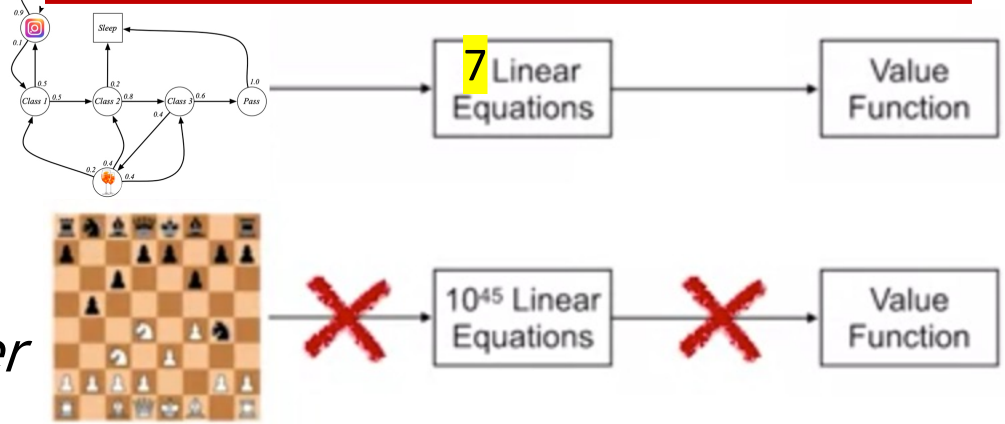
- The Bellman Equation (when we are dealing with Markov Reward Processes*) is linear
- We can solve the previous matrix directly
- If we have n states, the computational cost is $O[n^3]$: affordable only with small Markov Reward Processes (MRPs)
- For large MRPs we will look for efficient methods (Dynamic Programming, Monte-Carlo evaluation, Temporal-Difference learning)

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

$$(I - \gamma \mathcal{P})\, v = \mathcal{R}$$

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

Moreover, in true RL problems $\mathcal{R}$ and $\mathcal{P}$ are not given!



This will not be true when we will deal with optimization and maximization (when we will consider an agent making decisions!) in Markov Decision Processes

A. White, M. White 'Fundamentals of Reinforcement Learning'

iii. Markov Decision Processes (MDPs)

Let's add actions and decisions (a true RL problem!): a Markov Decision Process is a Reward Process with decisions. MPD is an environment in which all states are Markov.

Definition

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

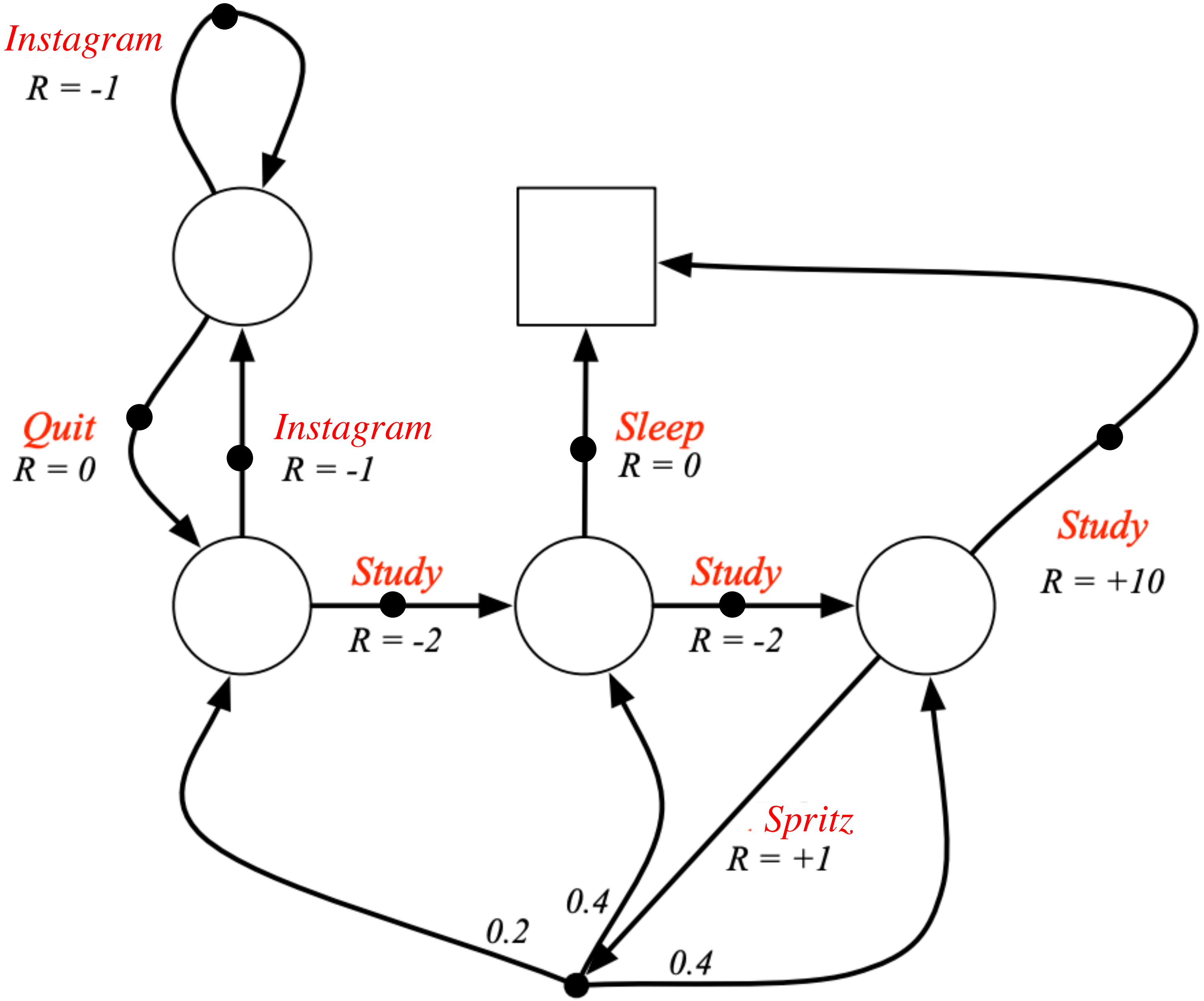
- $\mathcal{S}$ is a finite set of states
- $\mathcal{A}$ is a finite set of actions
- $\mathcal{P}$ is a state transition probability matrix with entries

$$\mathcal{P}_{ss'}^{a} = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a]$$

- $\mathcal{R}$ is a reward function, $\mathcal{R}_s^a = \mathbb{E}\,[R_{t+1} | S_t = s, A_t = a]$ (it is just the immediate reward, in that specific state)
- γ is a discount factor, $\gamma \in [0,1]$

iiI. Markov Decision Processes: Student Markov Chain

- Pay attention: we are reporting actions (typically indicated with a black dot)
- States are different here
- Now there is control and agency! We should define a policy!
- Now we can try to maximize our reward!



D. Silver 'Reinforcement Learning' @UCL

iii. MDPs: (Stochastic) Policies

Definition

A Policy π is a distribution over actions given that we are in a state:

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

- A policy fully defines the behaviour of an agent
- MDP policies depend on the current state (we are considering Markov states in MDPs, history doesn't matter)
- In MDP policies are stationary (do not depend on *t*), however we can change our policy in future episodes
- We consider stochastic policies: this allow us for example to deal with exploration!
- Please note that there is no reward here: the policy can be given or we may have 'learned' the policy with a dedicated procedure

iii. MDPs: (Stochastic) Policies

Given an MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ + a policy π:

- The state sequence $S_1, S_2, \ldots$ is a Markov Process $\langle \mathcal{S}, \mathcal{P}^{\pi} \rangle$ where

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

- The state and reward sequence $S_1, R_2, S_2, \ldots$ is a Markov Reward Process $\langle \mathcal{S}, \mathcal{P}^{\pi}, \mathcal{R}^{\pi}, \gamma \rangle$ where

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

iii. MDPs: (Stochastic) Policies

Given an MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ + a policy π:

- The state sequence $S_1, S_2, \ldots$ is a Markov Process $\langle \mathcal{S}, \mathcal{P}^{\pi} \rangle$ where

$$\mathcal{P}^{\pi}_{s,s'} = \sum_{a \in \mathcal{A}} \underbrace{\boxed{\pi(a|s)}}_{\text{AGENT}} \underbrace{\boxed{\mathcal{P}^{a}_{s,s'}}}_{\text{ENVIRONMENT}}$$

- The state and reward sequence $S_1, R_2, S_2, \ldots$ is a Markov Reward Process $\langle \mathcal{S}, \mathcal{P}^{\pi}, \mathcal{R}^{\pi}, \gamma \rangle$ where

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

iii. MDPs: Value Function

We had value functions with Markov Reward Processes, but now that we have agency, value of a state depends on the policy!

Definitions

The state-value function $v_\pi(s)$ of an MDP is the expected return from state s if we follow policy π :

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

The action-value function $q_\pi(s, a)$ of an MDP is the expected return from state s if we take action a and then we follow policy π :

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

Now in this setting we have agency so the expectation
is on the randomness of transition (and policy when not deterministic) but depends on the policy pi

iii. MDPs: Value Function

We had value functions with Markov Reward Processes, but now that we have agency, value of a state depends on the policy!

Definitions

The state-value function $v_\pi(s)$ of an MDP is the expected return from state s if we follow policy π :

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

This is defined for all actions, also for the ones that are taken by π in state s

The action-value function $q_\pi(s, a)$ of an MDP

from state s if we take action a and then we follow policy π :

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

iii. MDPs: Value Function – Why it depends on π

iii. MDPs: Value Function – Why it depends on π

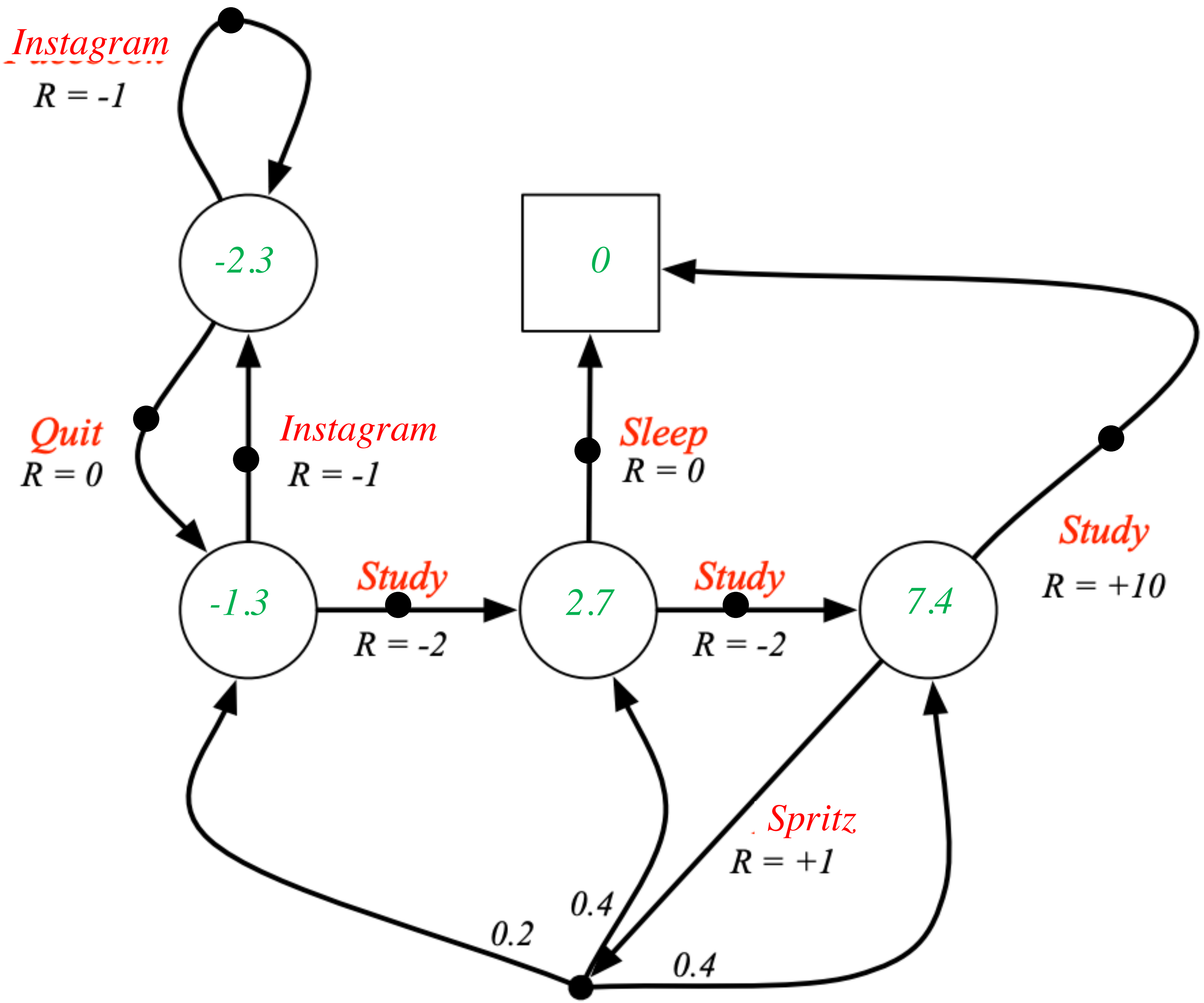
iii. MDPs: Value Function – Why it depends on π

$$V_{Magnus} \neq V_{GianAntonio}$$

iii. Markov Decision Processes: Student Markov Chain

We consider the undiscounted MDP ($\gamma = 1$) and a uniform random policy: for each state (C1, IG, C2, C3) there are two possible actions, each one with probability 0.5

$$\pi(a|s) = 0.5 \text{ for all } a, s$$



D. Silver 'Reinforcement Learning' @UCL

iii. MDPs: Bellman Expectation Equation

The value function $v_\pi(s)$ can again be decomposed into 2 parts:

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

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

Similarly for the action-value function

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

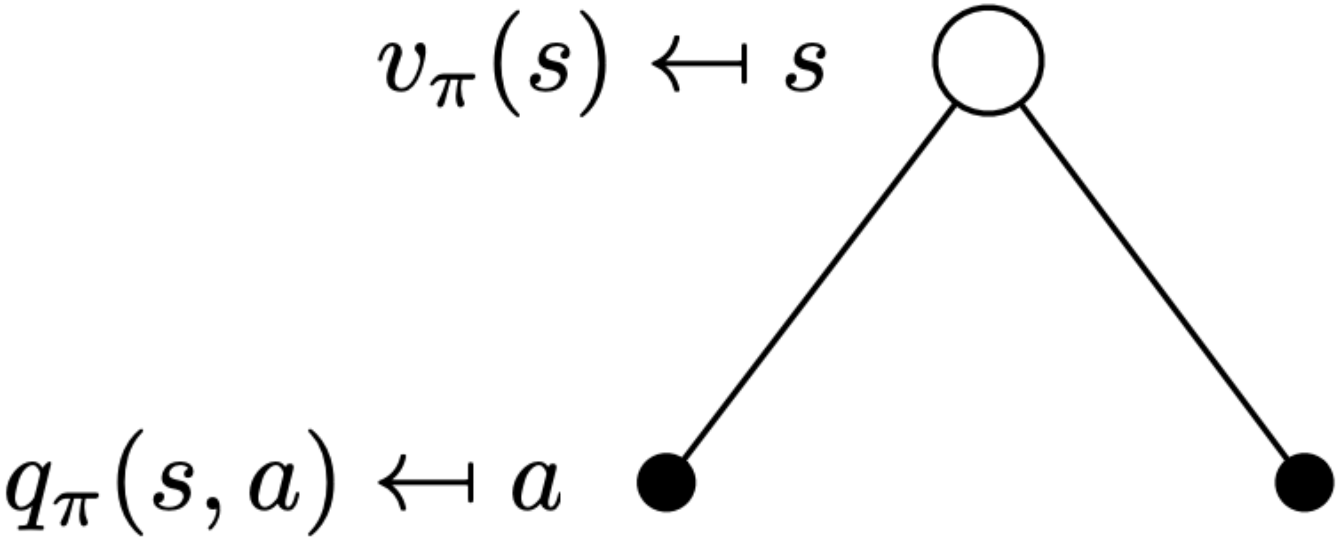
<u>* Please note that previously(for the Bellman Equation) we have considered $v(s)$</u>

iii. MDPs: Bellman Expectation Equation

Let's see the relation between q_π and v_π (1 of 4)

states: ○

actions: ●



$$\boxed{\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi[G_t | S_t = s] \\ q_\pi(s,a) &= \mathbb{E}_\pi[G_t | S_t = s, A_t = a] \end{aligned}}$$

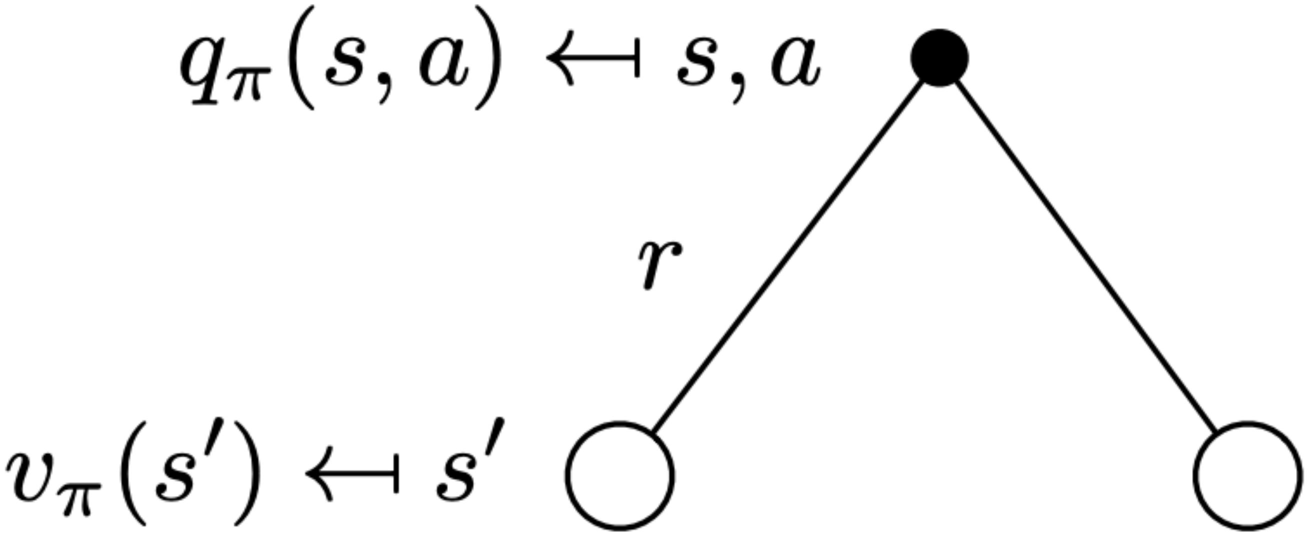
$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s,a)$$

iii. MDPs: Bellman Expectation Equation

Let's see the relation between q_π and v_π (2 of 4)

states: ○

actions: ●

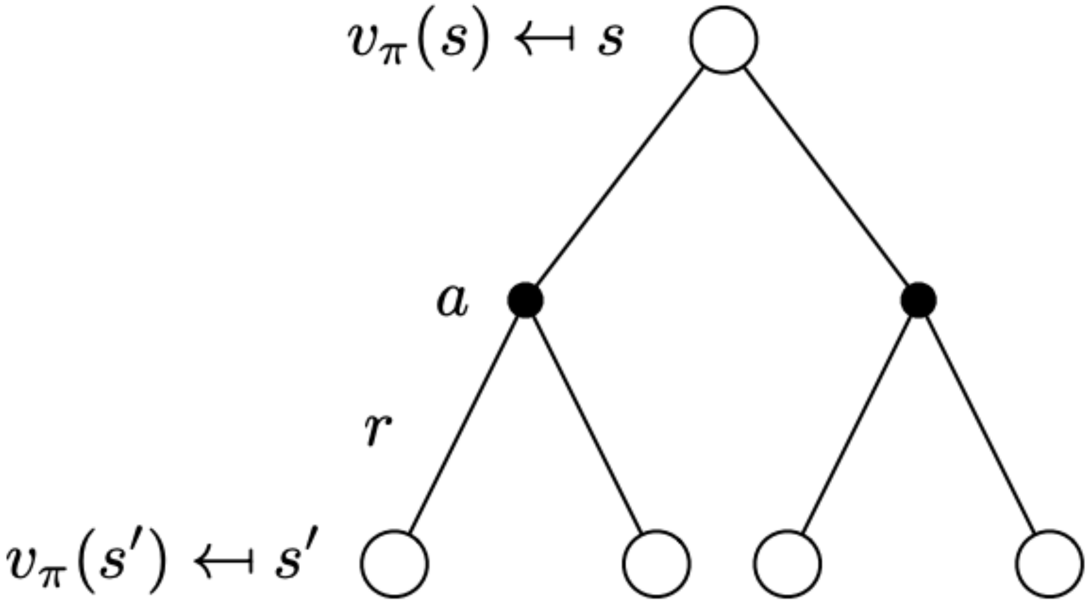


$$v_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s]$$
$$q_\pi(s,a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a]$$

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

iii. MDPs: Bellman Expectation Equation

Let’s see the relation between q_π and v_π (3 of 4)



states: ○

actions: ●

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

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

We can obtain a recursive description of v_π

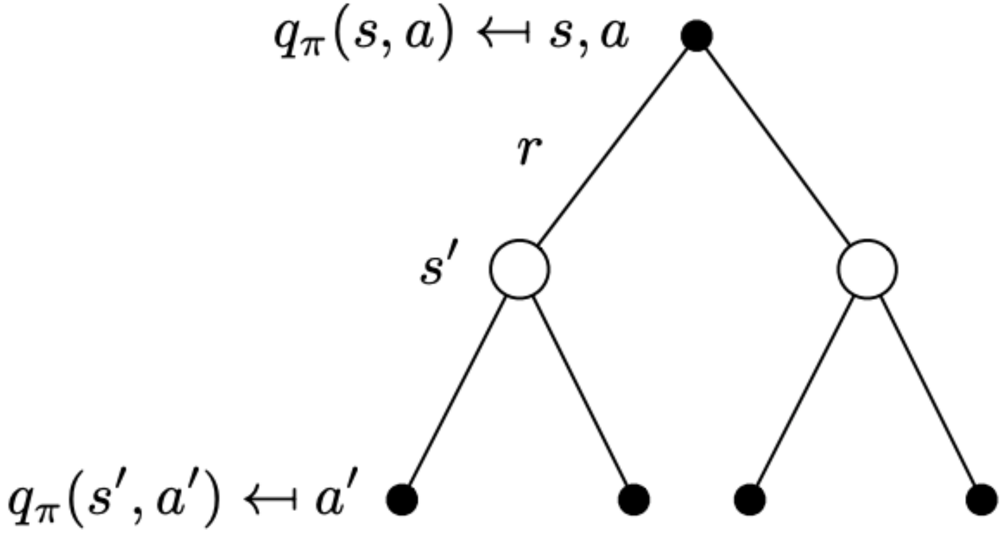
$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s') \right)$$

iii. MDPs: Bellman Expectation Equation

Let's see the relation between q_π and v_π (4 of 4)

states: ○

actions: ●



We obtain a recursive description of q_π

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

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

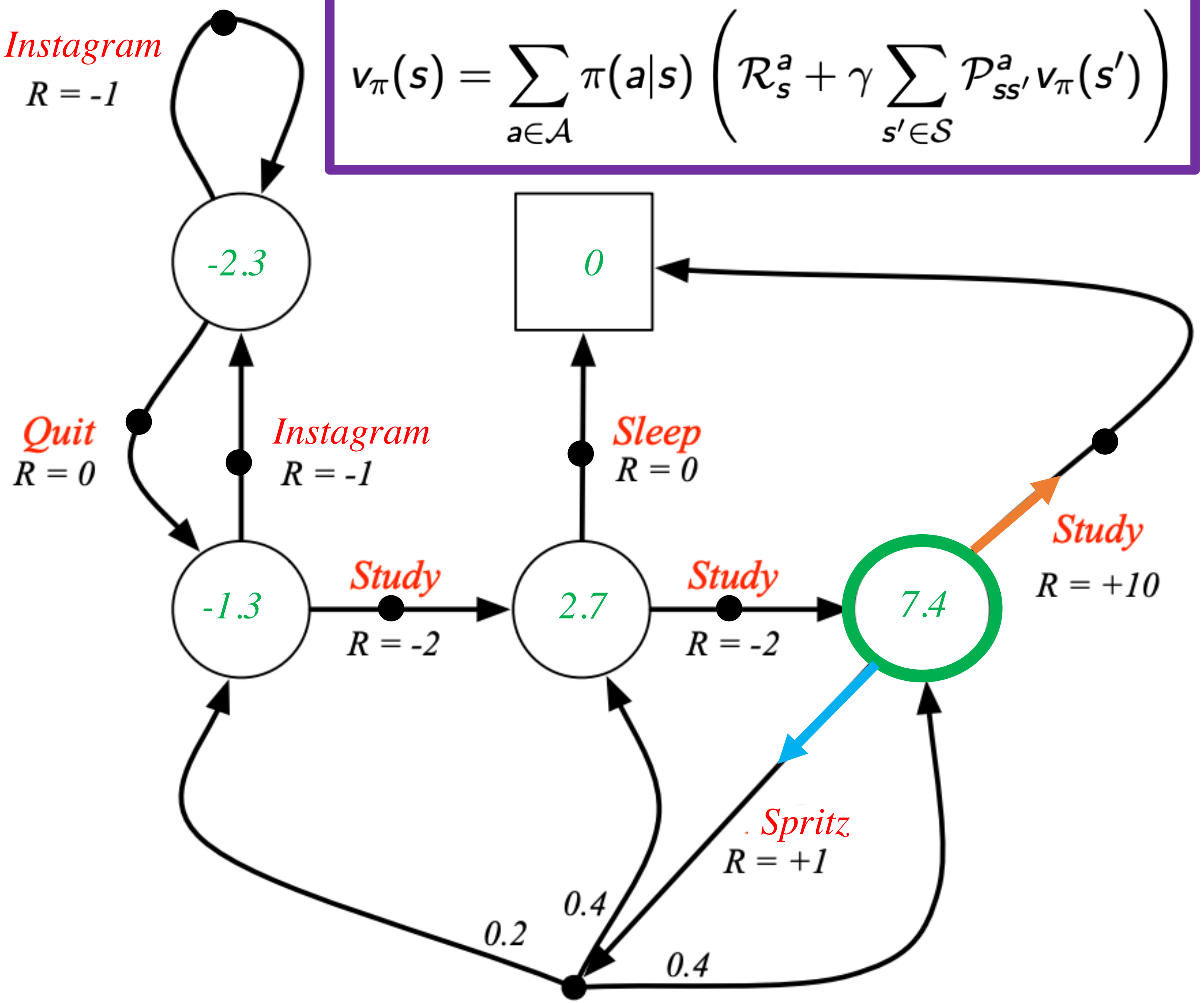
$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \sum_{a' \in \mathcal{A}} \pi(a'|s') q_\pi(s',a')$$

iiI. Markov Decision Processes: Student Markov Chain

Let's consider the previous case (undiscounted, uniform random policy) and let's verify with the recursive definition that $v_\pi(C3) = 7.4$

7.4 = 0.5*10+0.5*(1+0.4*7.4-0.2*1.3+0.4*2.7)

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



D. Silver 'Reinforcement Learning' @UCL

iii. Markov Decision Processes: Bellman Expectation Equation in Matrix Form

Also the Bellman Expectation Equation can be written concisely in matrix form (we are resorting to the induced Markov Reward Process by using $\mathcal{R}_s^\pi = \sum_{a \in \mathcal{A}} \pi(a|s)\, \mathcal{R}_s^a$ seen before):

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

That can be solved:

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

As said, a good part of the course will be dedicated to find efficient ways to avoid computing such set of linear equations.

Optimal Value Function

Definition

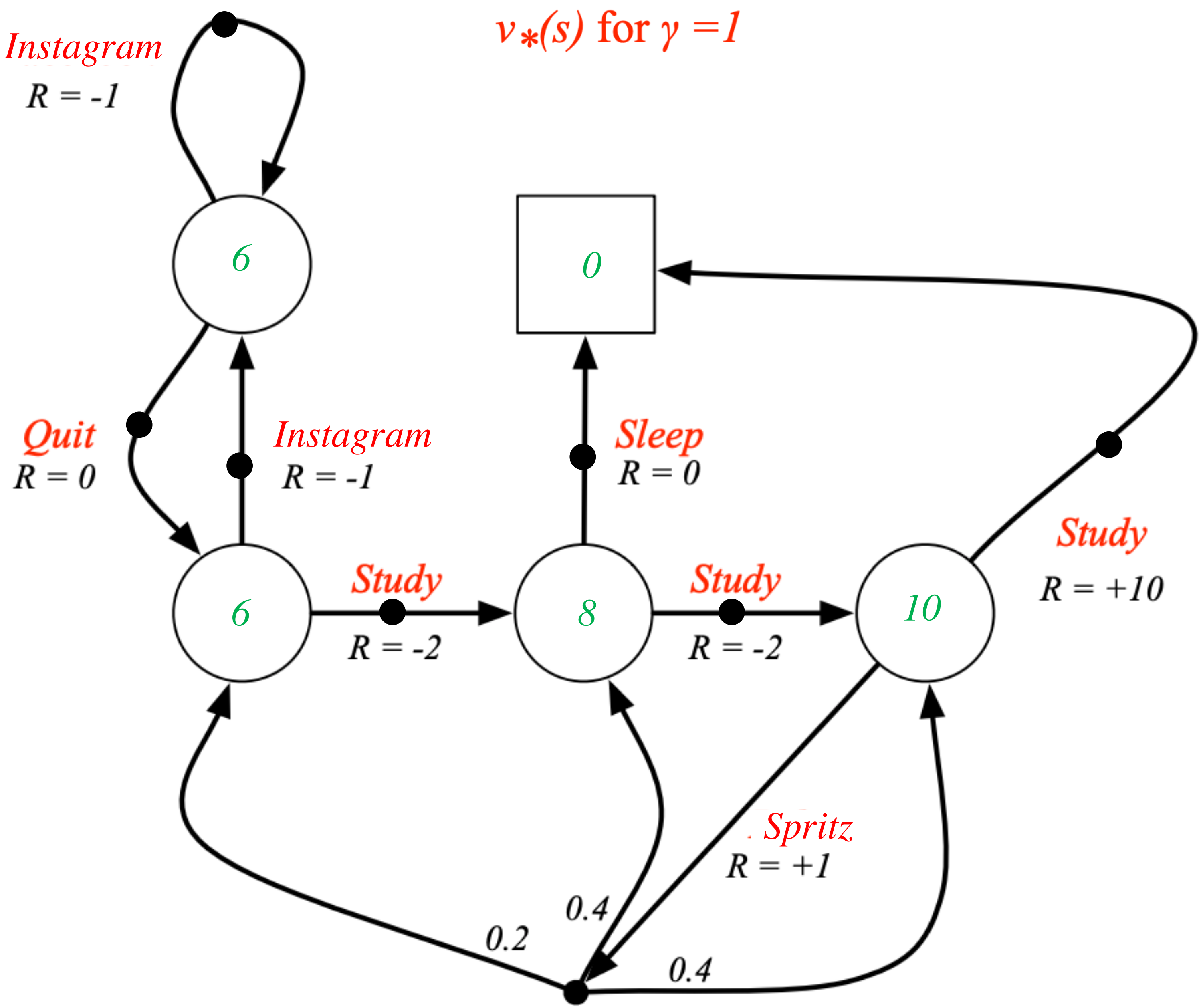
The optimal state-value function $v_*(s)$ is the maximum state-value function over all policies:

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

Optimal value function on the Student Markov Chain

- Since here we have the deterministic case (one action from one state lead you to a deterministic successor state) we can easily compute the value functions for each state going backwards

- We still don't have an explicit indication on how to behave



D. Silver 'Reinforcement Learning' @UCL

Optimal Value Function

Definition

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

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

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

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

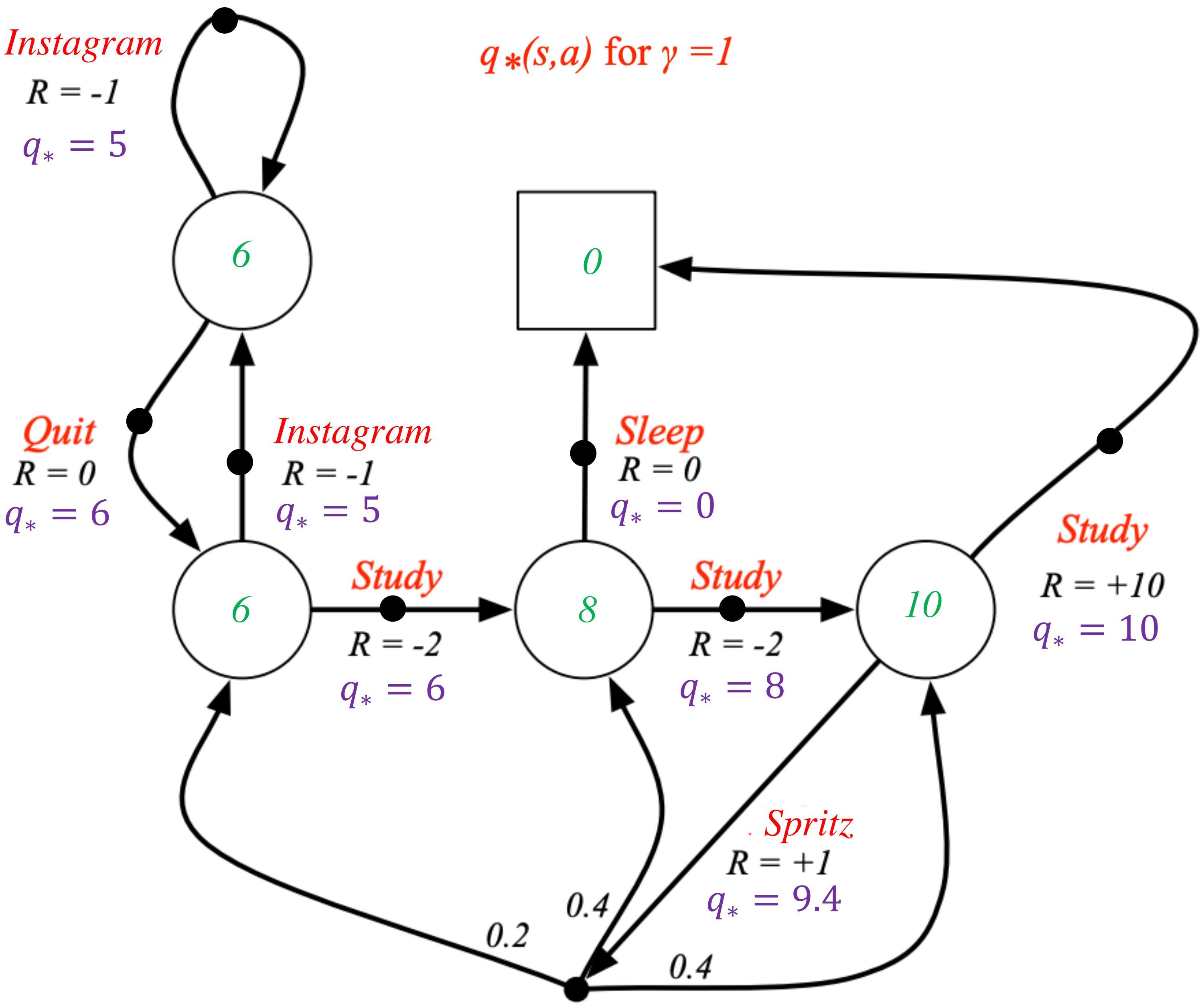
An MDP can be considered 'solved' on a RL perspective when the optimal action-value function is known: we always know the optimal action to take from a certain state

Optimal action value function on the Student Markov Chain

The following holds also for the optimal policy:

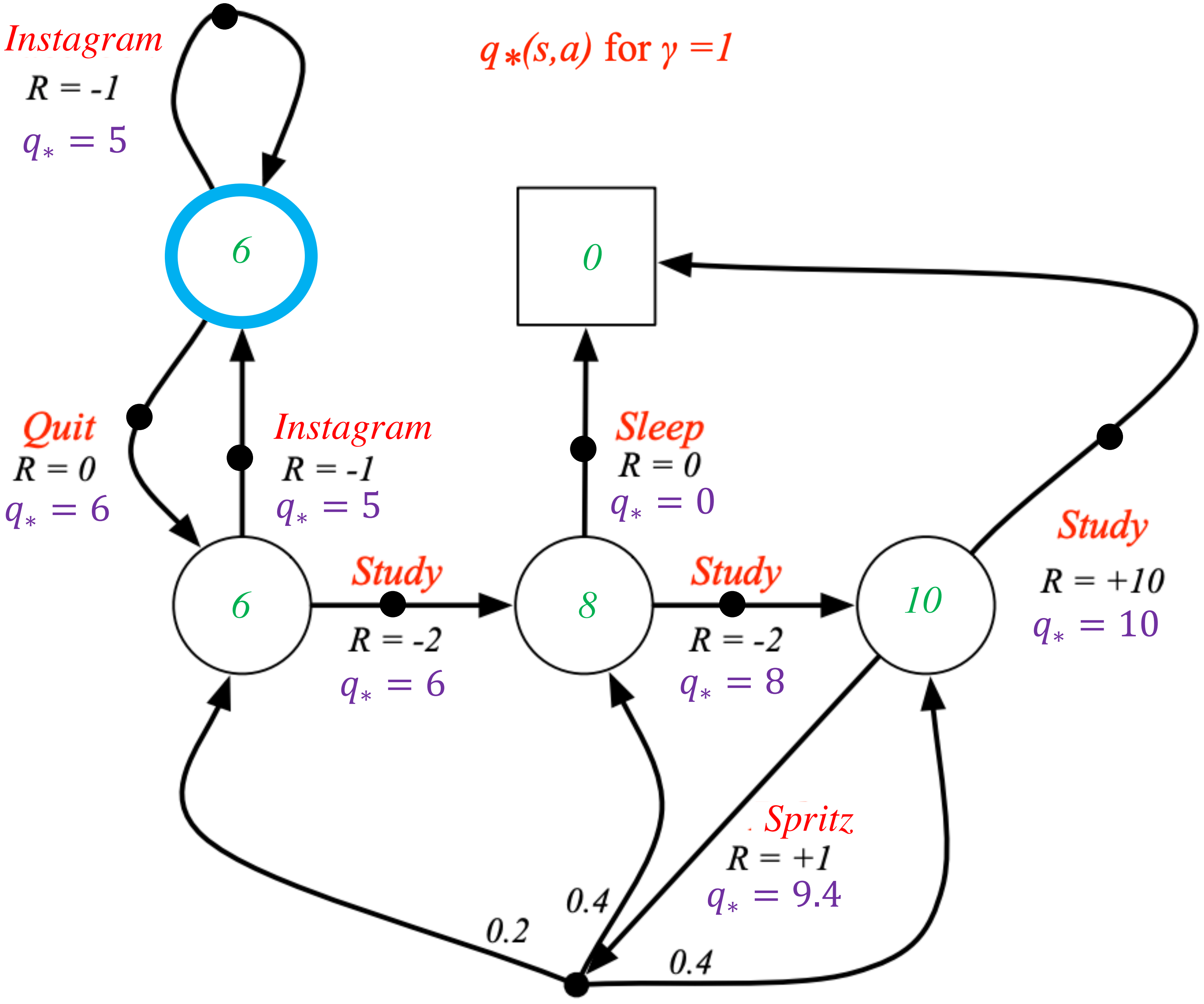
$$q_*(s,a) = \max_\pi q_\pi(s,a)$$

Action value function allows us to take decisions...



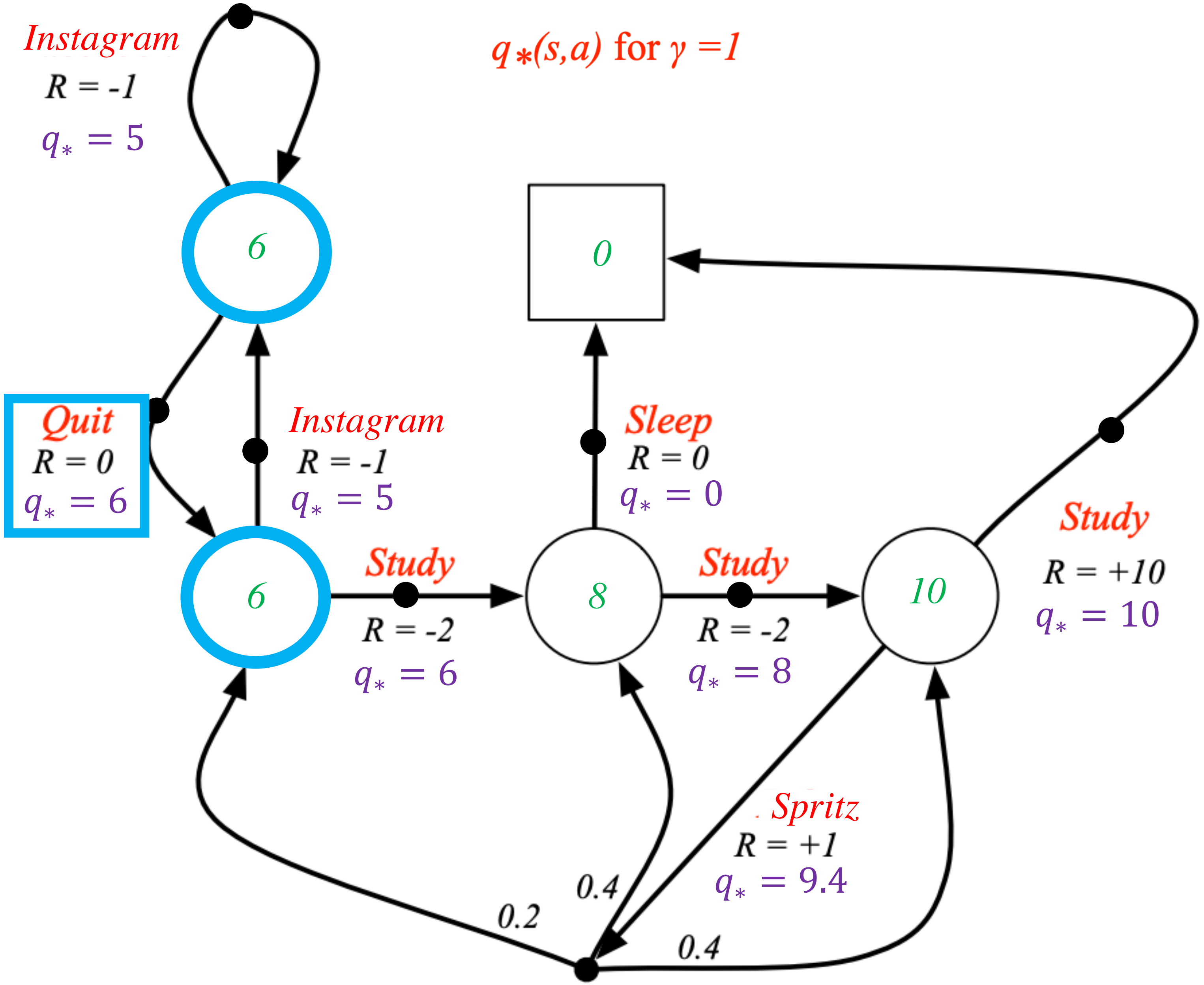
D. Silver 'Reinforcement Learning' @UCL

Optimal action value function on the Student Markov Chain



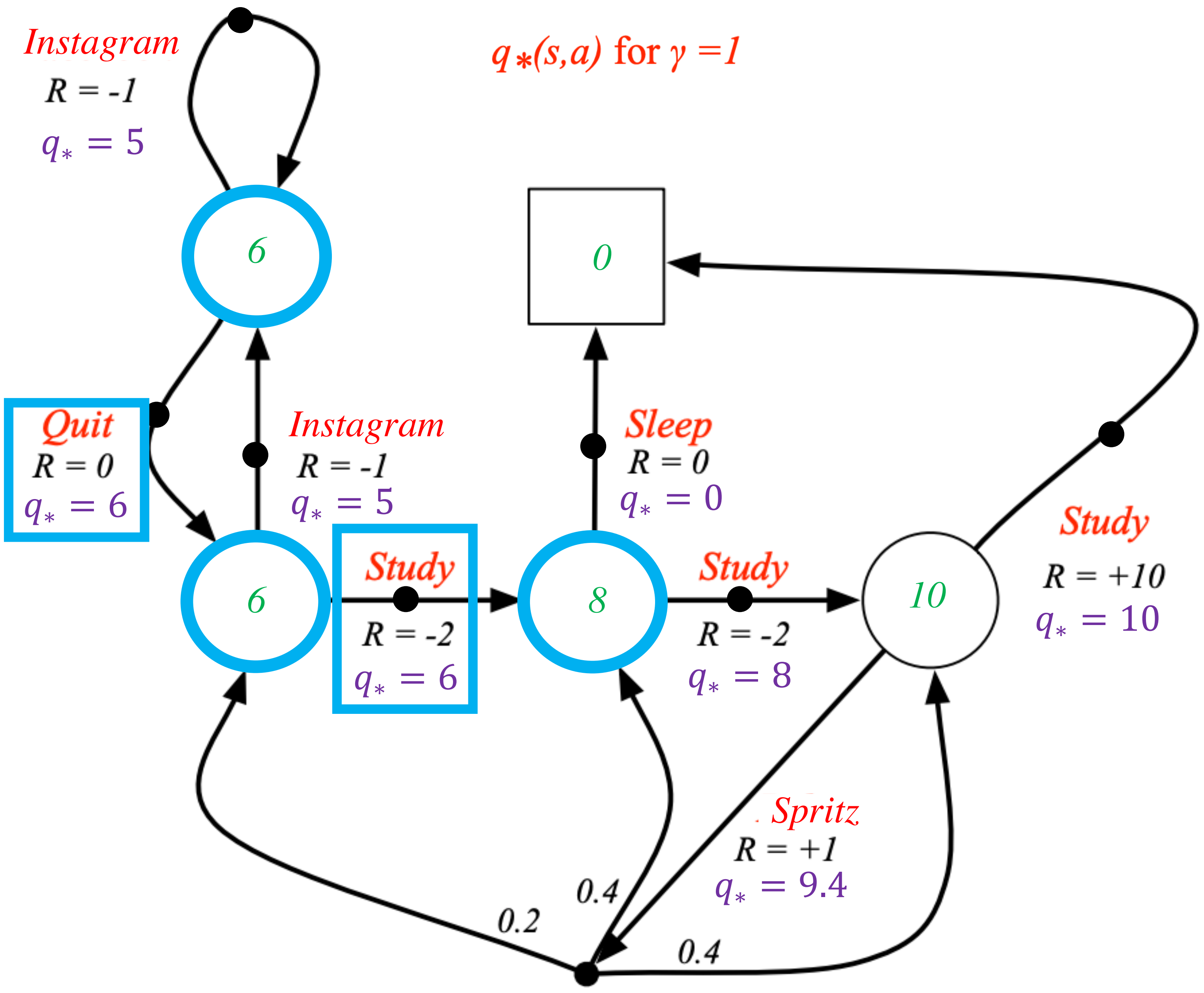
D. Silver 'Reinforcement Learning' @UCL

Optimal action value function on the Student Markov Chain



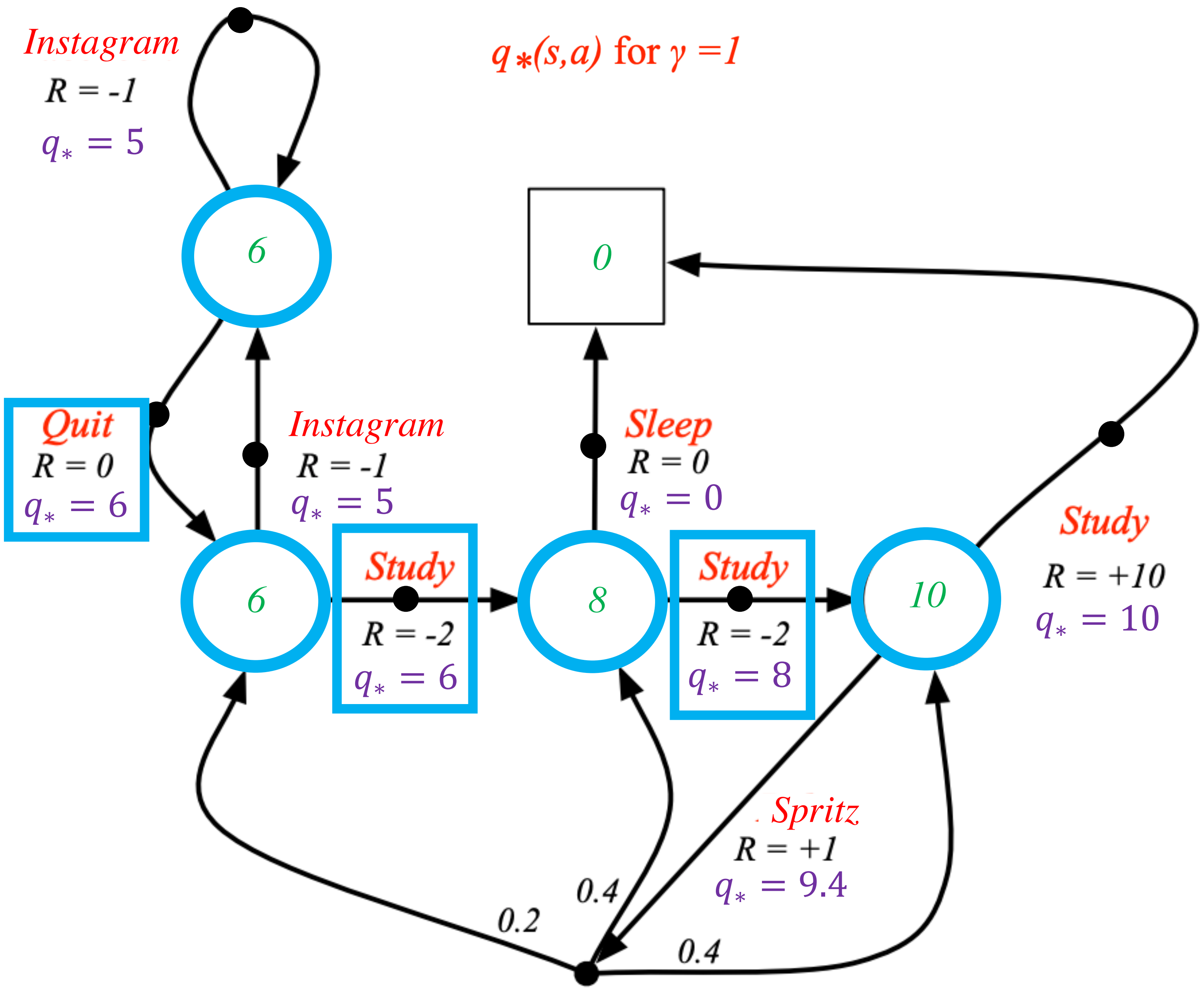
D. Silver 'Reinforcement Learning' @UCL

Optimal action value function on the Student Markov Chain



D. Silver 'Reinforcement Learning' @UCL

Optimal action value function on the Student Markov Chain

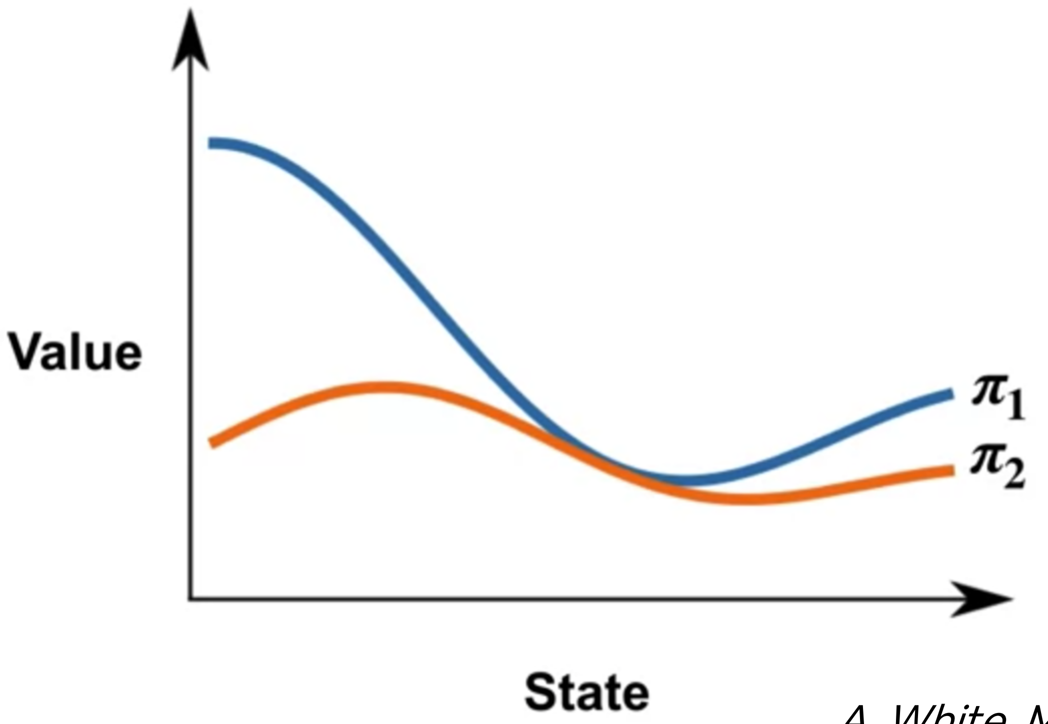


D. Silver 'Reinforcement Learning' @UCL

Optimal Policy

Our final goal is to find an optimal policy: the best way to act in a MDP!

We define an order over policies: $\pi \geq \pi'$ if $v_{\pi}(s) \geq v_{\pi'}(s)$ for all s



$\pi_1 \geq \pi_2$

A. White, M. White 'Fundamentals of Reinforcement Learning'

Optimal Policy

Theorem

For any MDP:

1. There exists an optimal policy π_* such that $\pi_* \geq \pi$ for all possible π
2. All optimal policies achieve the optimal value function

$$v_{\pi_*}(s) = v_*(s)$$

3. All optimal policies achieve the optimal value function

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

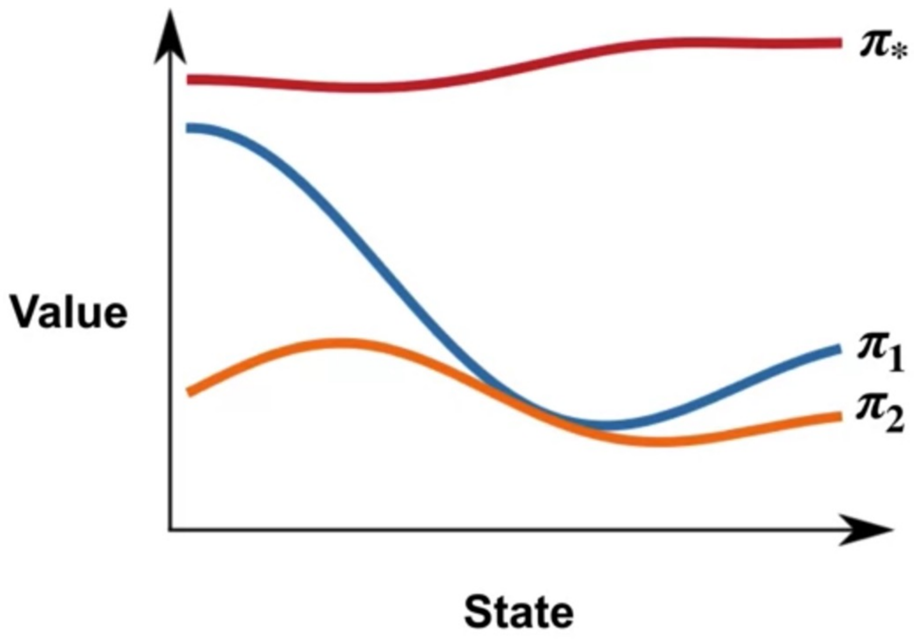
Finding an Optimal Policy

- How to operationally find an optimal policy?

A simple approach, if we have $q_*(s,a)$, if by maximizing over:

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

- There is always a deterministic optimal policy for any MDP



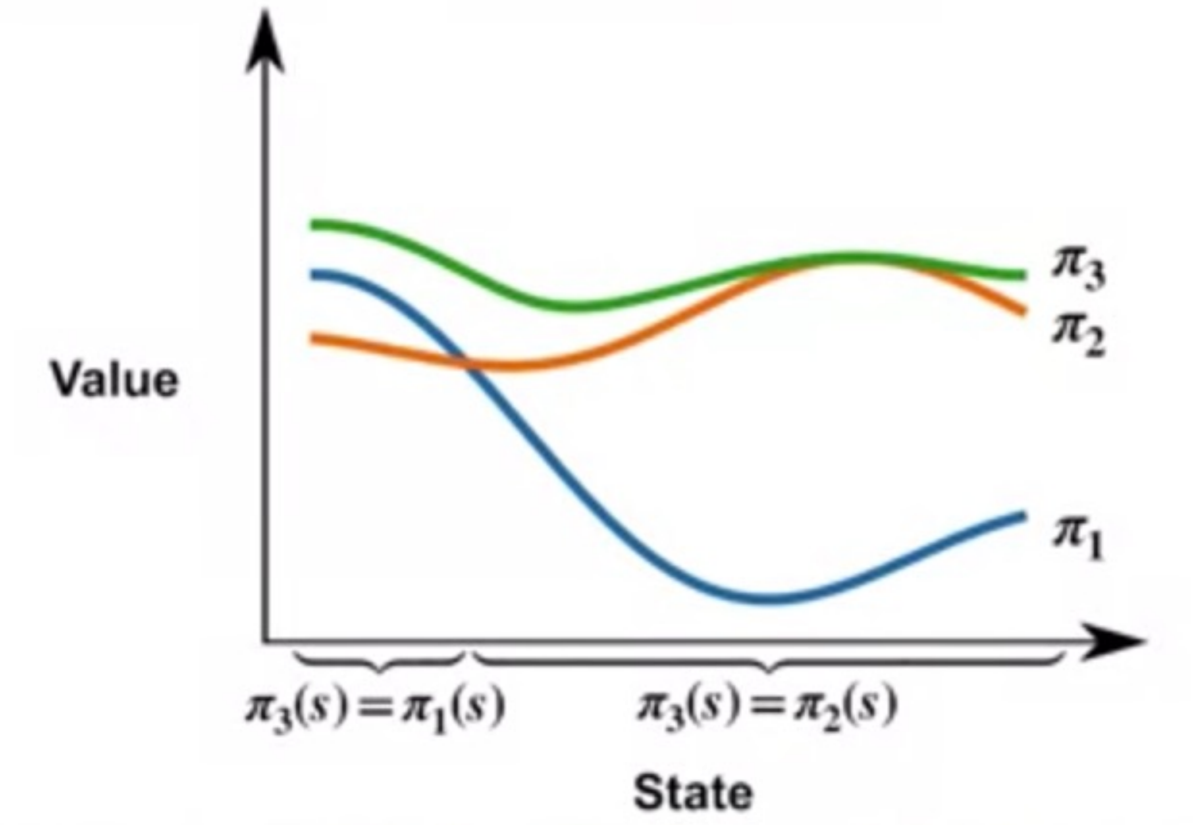
A. White, M. White 'Fundamentals of Reinforcement Learning'

Optimal Policy Existence

- Why there is always an optimal policy?

Optimal Policy Existence

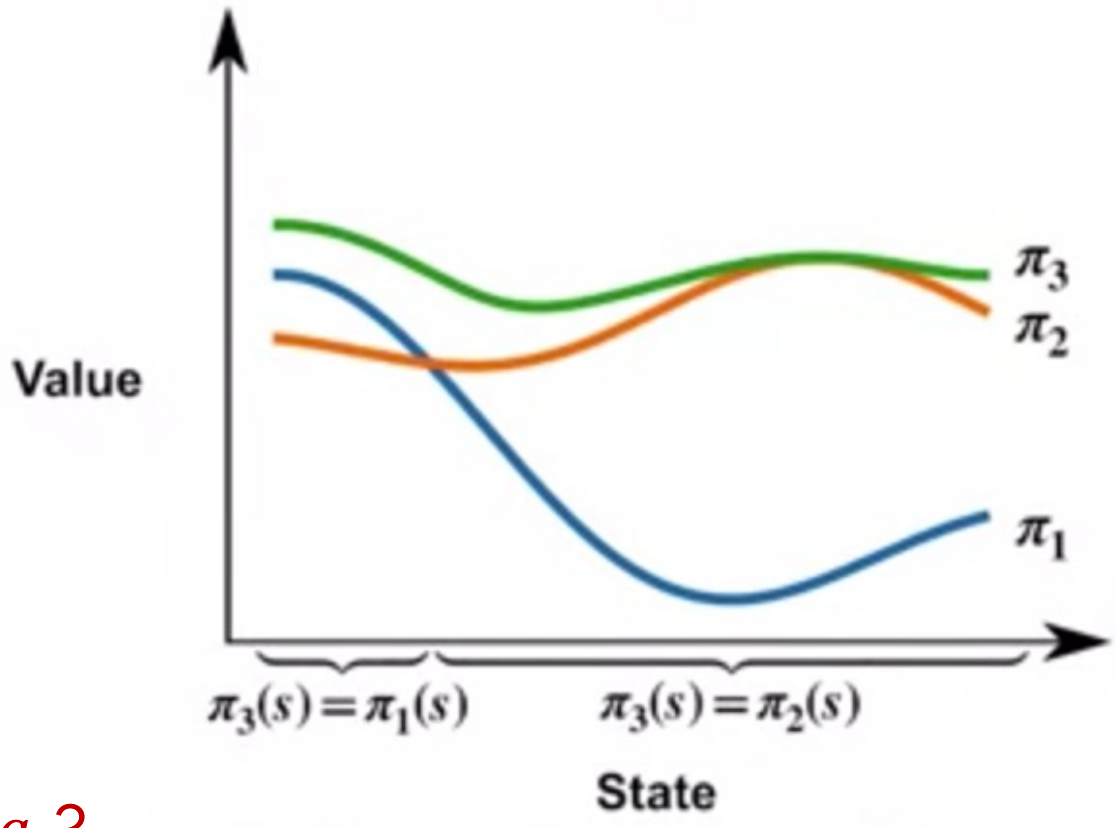
- Why there is always an optimal policy?



A. White, M. White 'Fundamentals of Reinforcement Learning'

Optimal Policy Existence

- Why there is always an optimal policy?

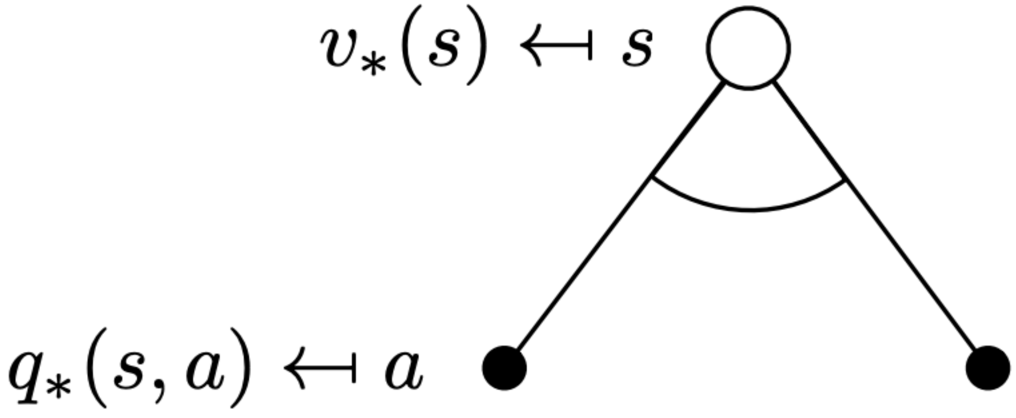


But how do we get q_*?

A. White, M. White 'Fundamentals of Reinforcement Learning'

Bellman Optimality Equation 1/3

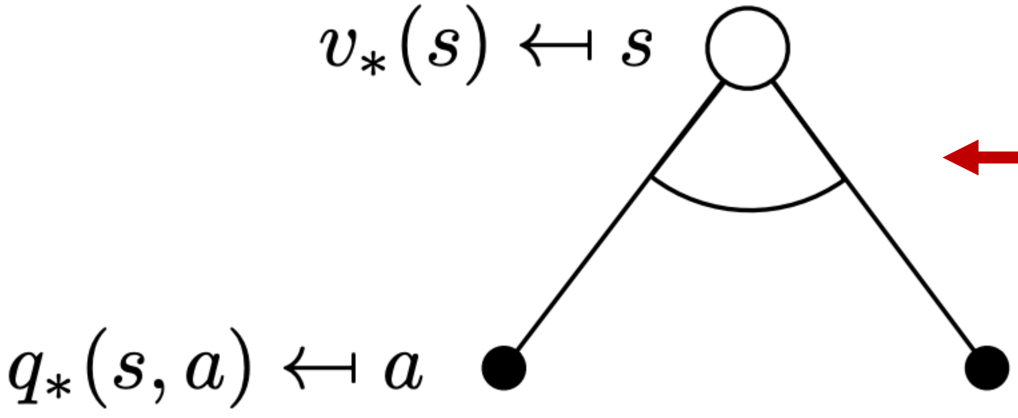
- Not to be confused with the Bellman Expectation Equation, that holds for a generic policy and it is a way to recursively define v_π and q_π
- The Bellman Optimality Equation is a way to define the optimal v_* with itself: we exploit again the 1-step look-ahead principle



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

Bellman Optimality Equation 1/3

- Not to be confused with the Bellman Expectation Equation, that holds for a generic policy and it is a way to recursively define v_π and q_π
- The Bellman Optimality Equation is a way to define the optimal v_* with itself: we exploit again the 1-step look-ahead principle

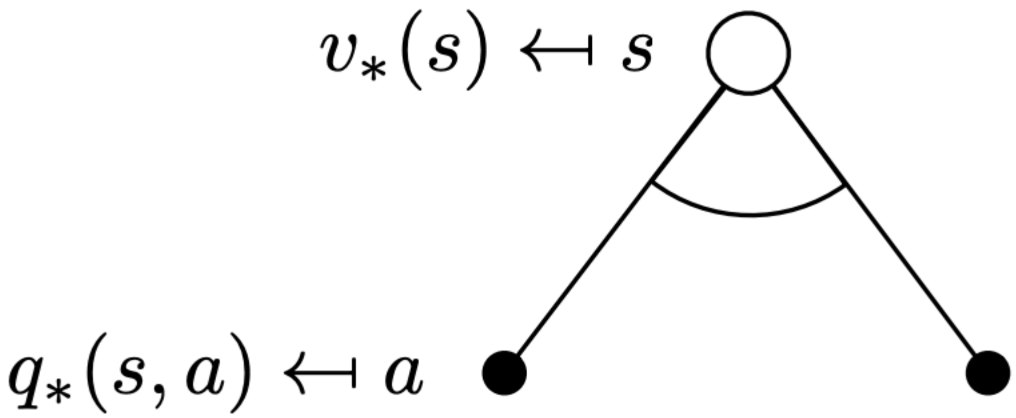


This symbol indicates the max over all possible choices for the action value: we don't need to average

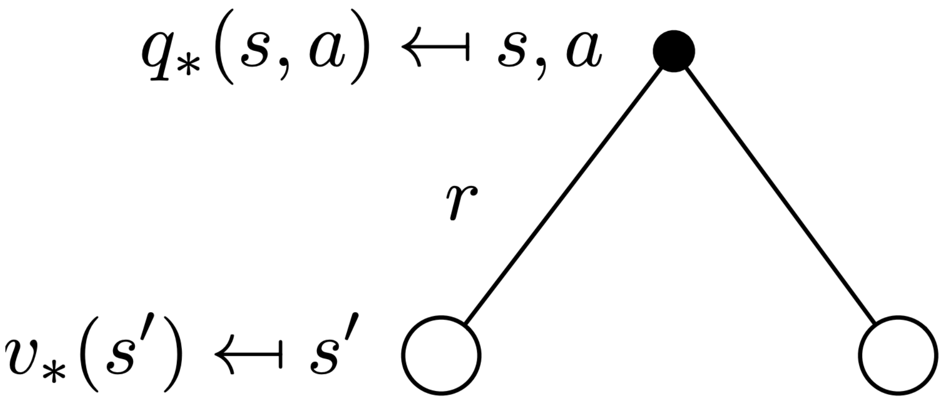
$$v_*(s) = \max_a q_*(s, a)$$

Bellman Optimality Equation 1/3

- Not to be confused with the Bellman <u>Expectation</u> Equation, that holds for a generic policy and it is a way to recursively define v_π and q_π
- The Bellman Optimality Equation is a way to define the optimal v_* with itself: we exploit again the 1-step look-ahead principle



Agent: here we pick the 'best' action

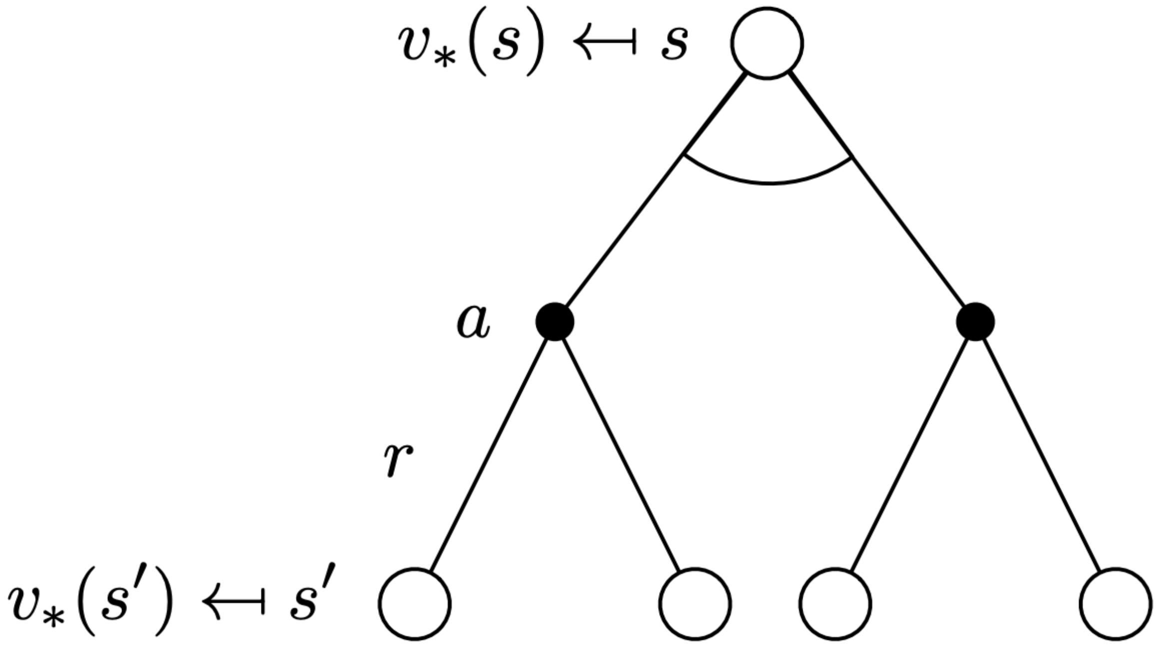


Environment: now we need to average!

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

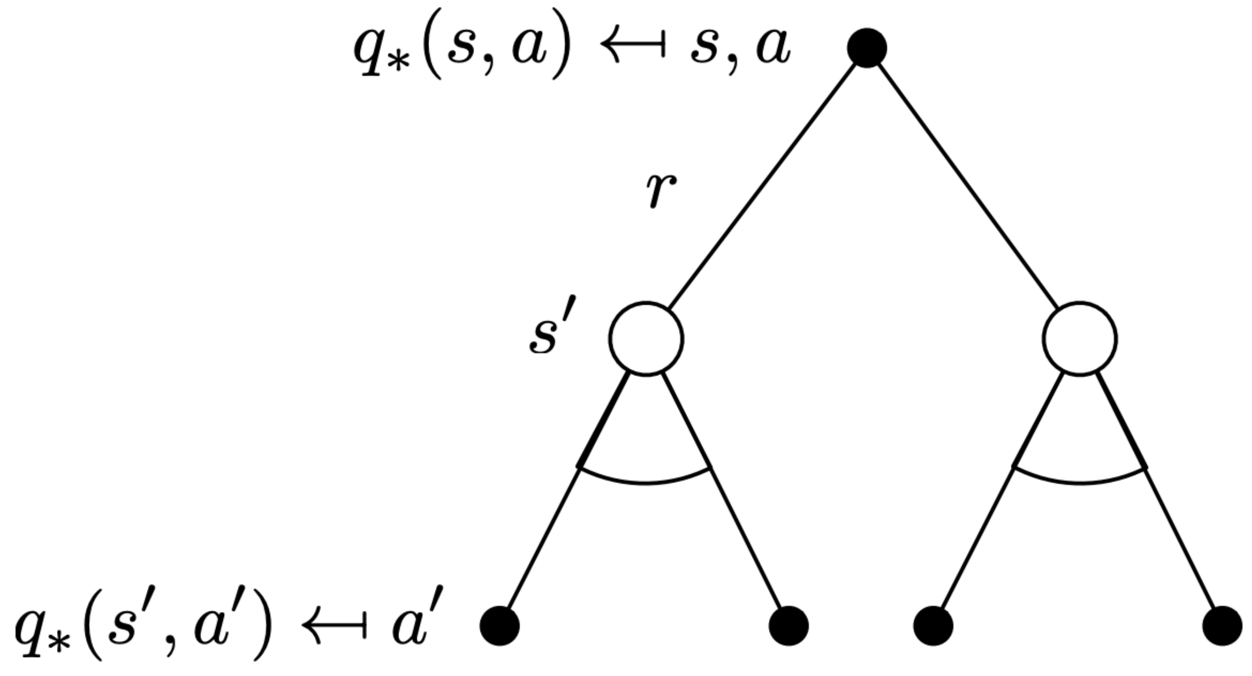
$$q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s')$$

Bellman Optimality Equation 2/3



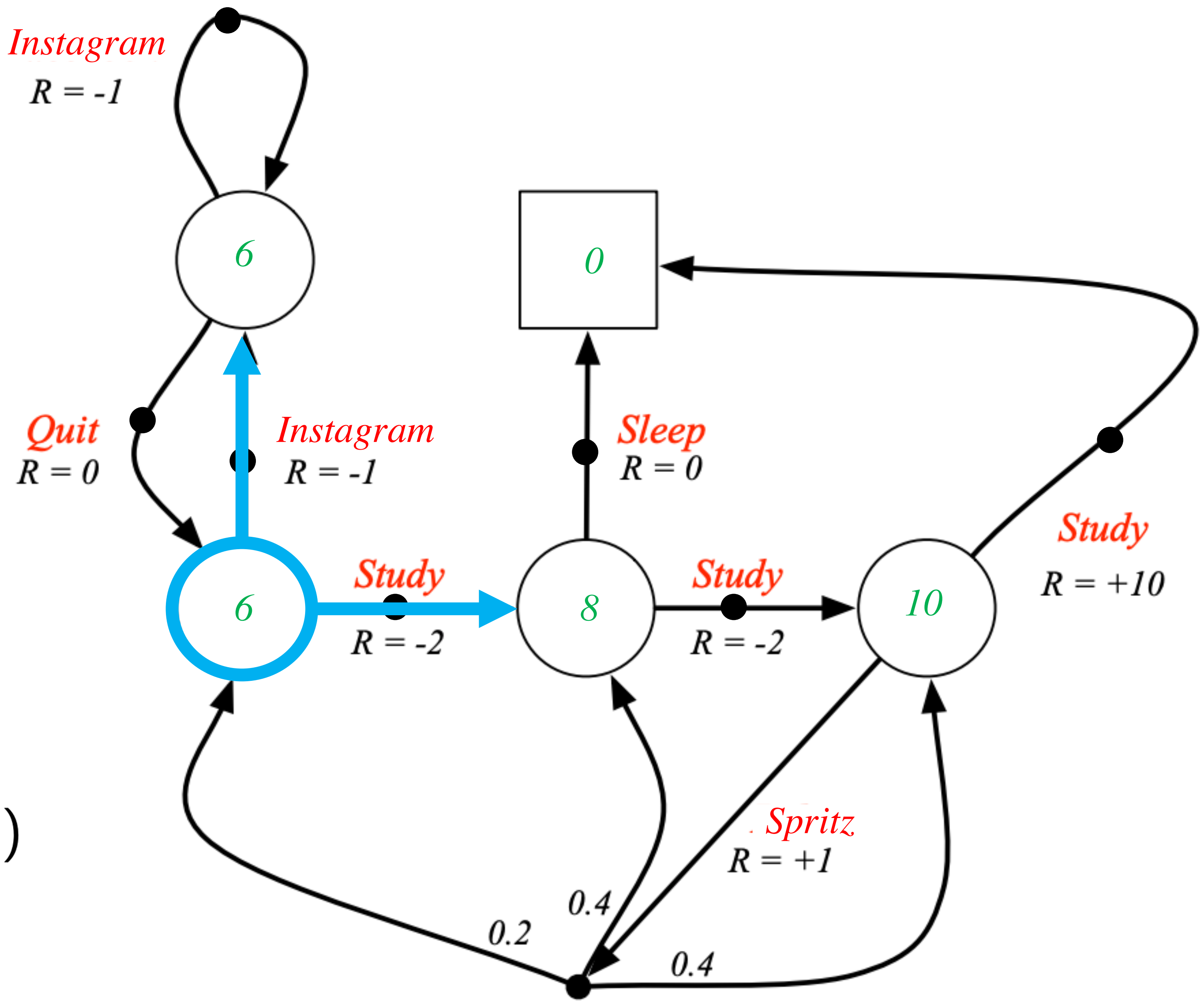
$$\boxed{v_*(s) = \max_a \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s')}$$

Bellman Optimality Equation 3/3



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

Bellman Optimal equation in the Student MDP



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

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

D. Silver 'Reinforcement Learning' @UCL

Bellman Optimality Equation

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

- Since there is a non-linear operation (a max) we don't have a closed-form solution
- We will resort to iterative methods: value iteration, policy iteration, q-learning, SARSA, ...

Summarizing

Markov Decision Processes (MDPs) formally describe an environment for Reinforcement Learning

i. Markov Processes $\langle \mathcal{S}, \mathcal{P} \rangle$

ii. Markov Reward Processes $\langle \mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

iii. Markov Decision Processes (MDPs) $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

From now on we will deal with MDPs!

Markov Reward Processes and MDPs can be ‘solved’ with respect to:

- Deriving the value function and the action-value function (in MDP w.r.t. a policy)
- Finding the best policy

Summarizing

Markov Decision Processes (MDPs) formally describe an environment for Reinforcement Learning

i. Markov Processes $\langle \mathcal{S}, \mathcal{P} \rangle$

ii. Markov Reward Processes $\langle \mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

iii. Markov Decision Processes (MDPs) $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

From now on we will deal with MDPs!

Markov Reward Processes and MDPs can be ‘solved’ with respect to:

- Deriving the value function and the action-value function (in MDP w.r.t. a policy) -> (POLICY) EVALUATION
- Finding the best policy -> CONTROL / POLICY IMPROVEMENT

Summarizing

Richard E. Bellman

Bellman equations are our tool to 'solve' Markov Reward Processes (MRPs) and MDPs thanks to their recursive nature:

MRP	Bellman equation: for finding value functions	Linear: we can use it for 'small' MRPs. We need to resort to iterative approaches for 'large' MRPs
MDP	Bellman expectation equation: for finding value functions and action-value functions	Linear: we can use it for small MDPs. We need to resort to iterative approaches for 'large' MDPs
MDP	Bellman optimality equation: for finding optimal value functions and optimal action-value functions	Non-linear: we need iterative approaches even for small MDPs.

Summarizing

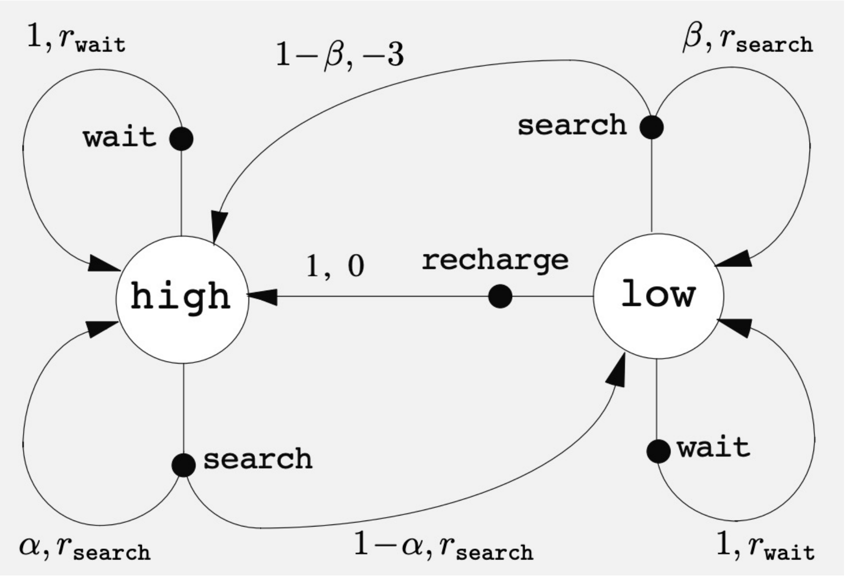
Richard E. Bellman

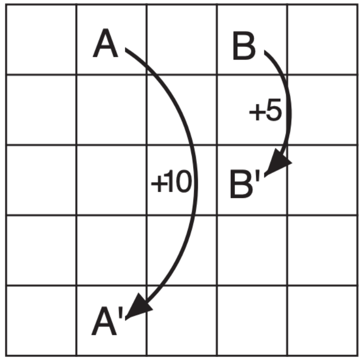
Bellman equations are our tool to 'solve' Markov Reward Processes (MRPs) and MDPs thanks to their recursive nature:

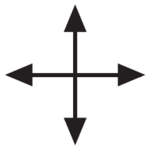
In the book

~~MRP~~	~~Bellman equation: for finding value functions~~	~~Linear: we can use it for 'small' MRPs. We need to resort to iterative approaches for 'large' MRPs~~
MDP	Bellman ~~expectation~~ equation: for finding value functions and action-value functions	Linear: we can use it for small MDPs. We need to resort to iterative approaches for 'large' MDPs
MDP	Bellman optimality equation: for finding optimal value functions and optimal action-value functions	Non-linear: we need iterative approaches even for small MDPs.

Examples in the Book (chapter 3)

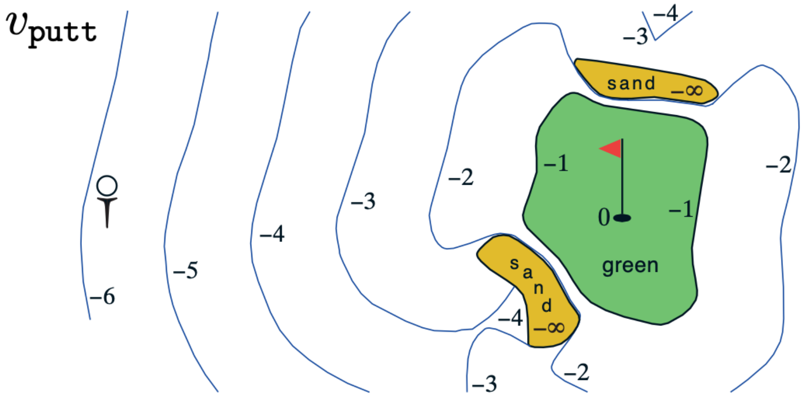


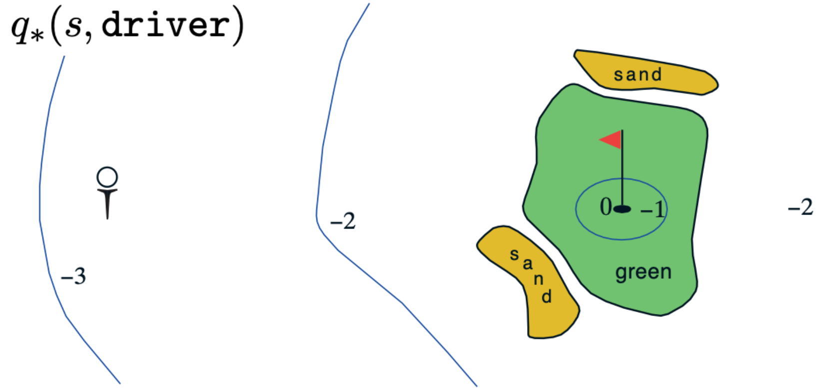




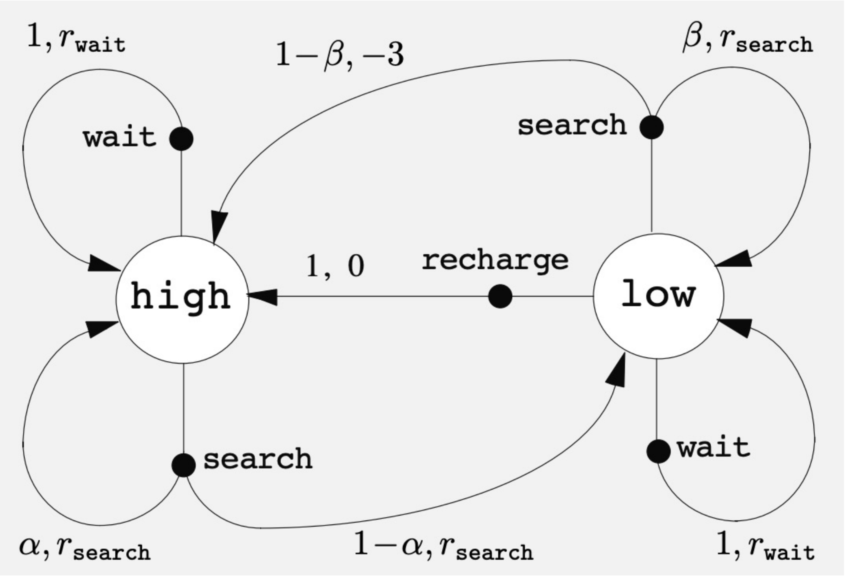
Actions

3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

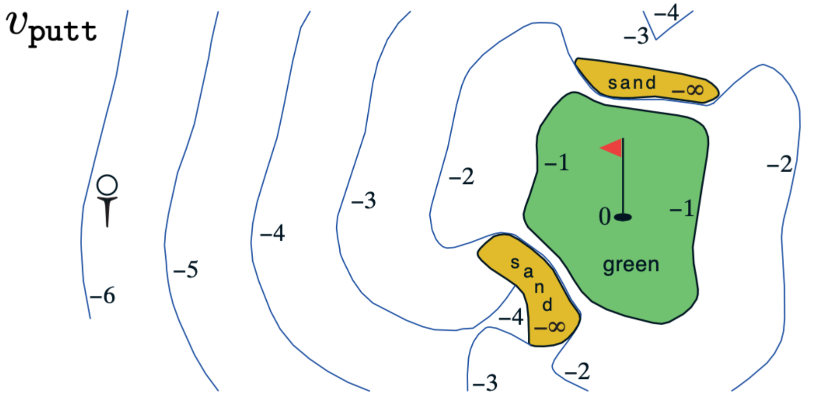


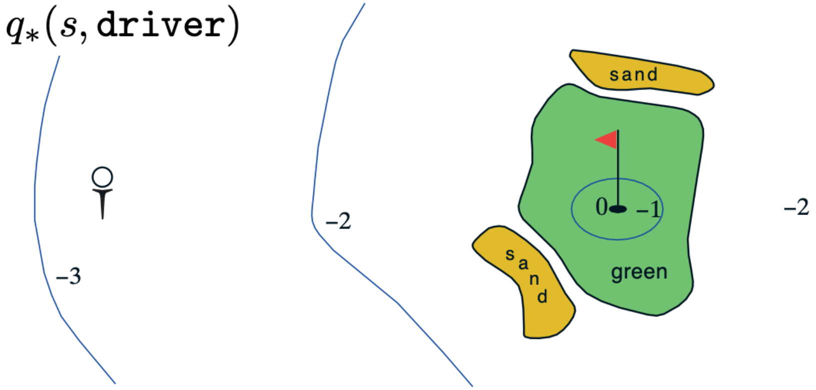


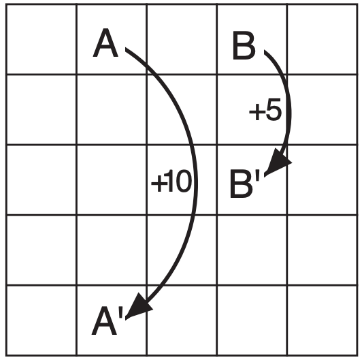
Examples in the Book (chapter 3)

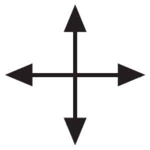


We will not see these in the laboratory (we will move directly to Chapter 4 examples): we suggest to take a look by yourselves!









Actions

3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

MDP: Exam

- All the content of Chapter 3 of the book may be exam material
- Keep in mind that the book presents directly MDPs: use these slides for a definition of Markov Processes and Markov Reward Processes

Credits

- Image of the course is taken from C. Mahoney 'Reinforcement Learning' https://towardsdatascience.com/reinforcement-learning-fda8ff535bb6
- Overall structure of the lecture and some content was inspired/adapted from D. Silver RL course a @ UCL

Thank you!
Questions?

Lecture #04
Markov Decision Processes & Bellman Equations

Gian Antonio Susto