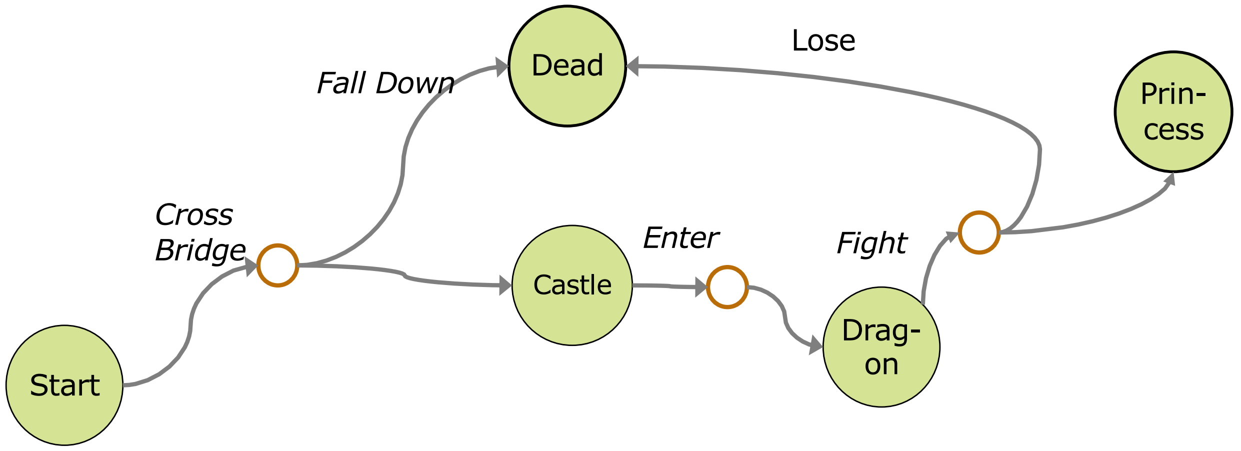
HSLU Lucerne University of Applied Sciences and Arts

Markov Decision Processes



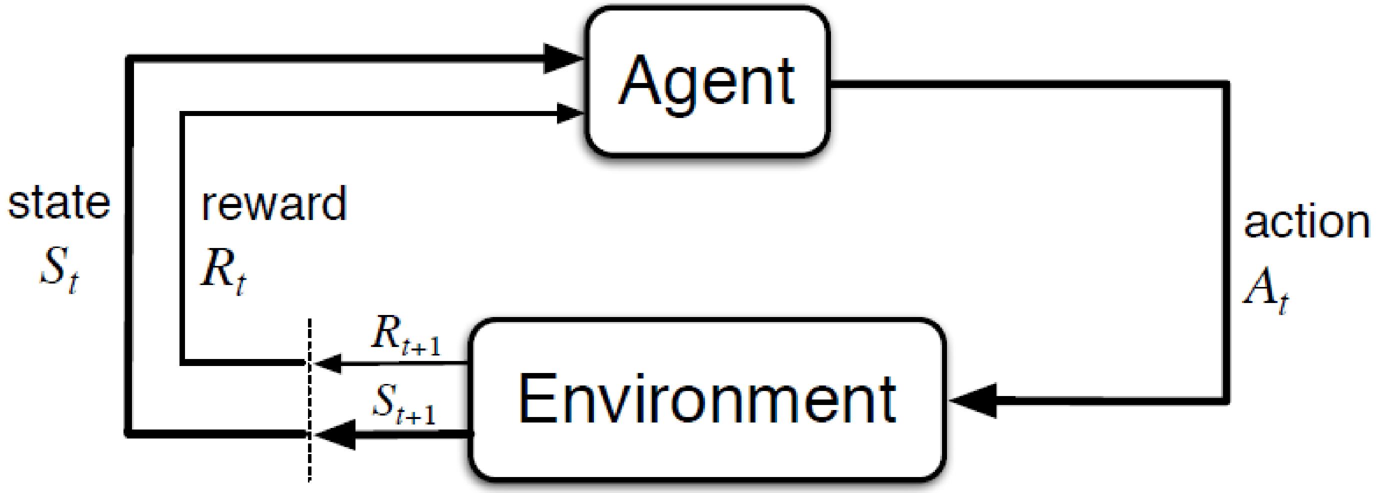
Reinforcement Learning
October 6, 2022

FH Zentralschweiz

Learning Objectives

- How to formulate a RL problem as Markov Decision Process
- Understand episodic tasks and episodes
- Differentiate between episodic and continous tasks
- Understand how discounts are used for returns in continuing tasks
- Derive the Bellman equation
- Differentiate between state-value and action-value functions
- Understand backup diagrams

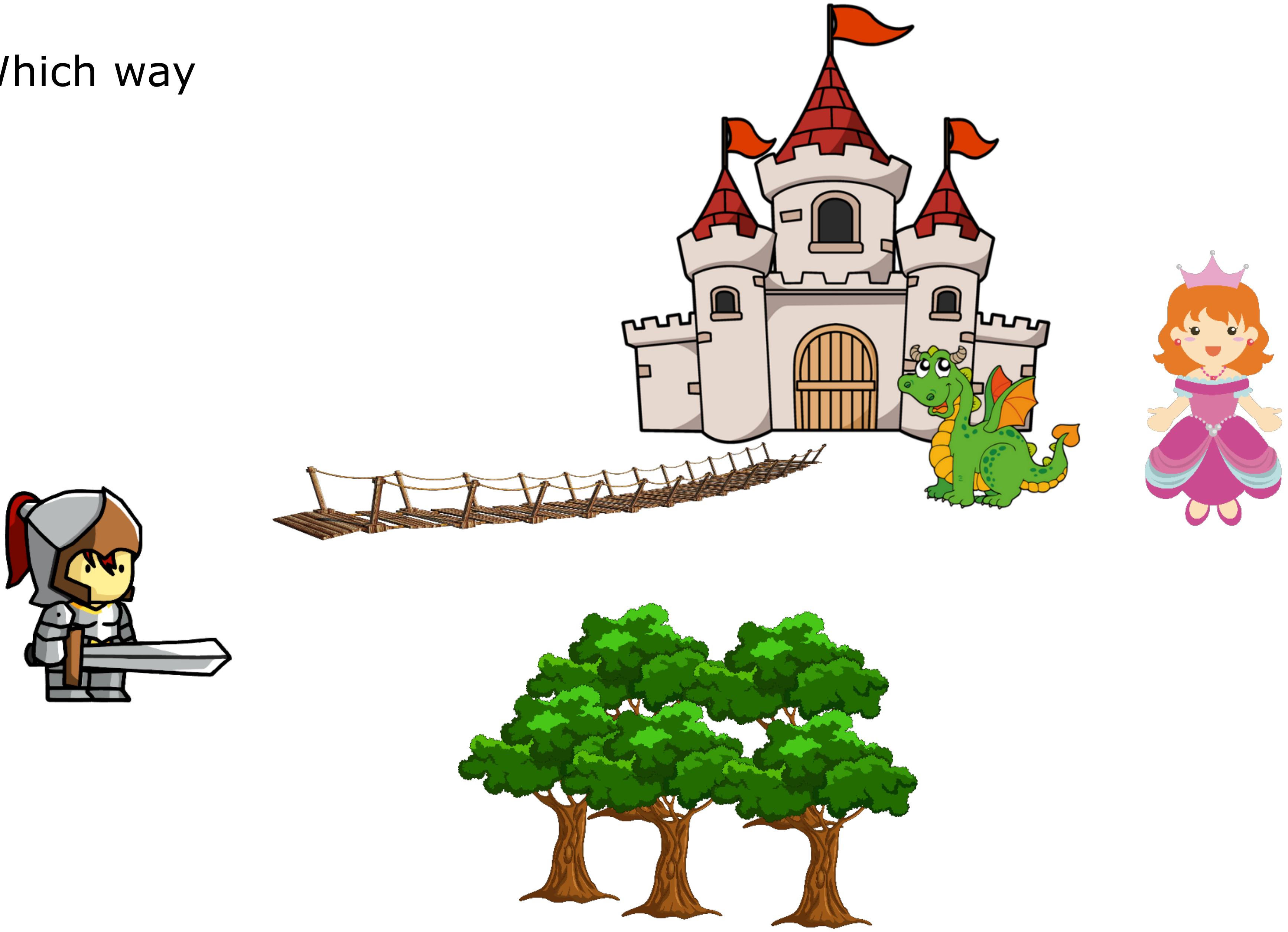
Agent and Environement



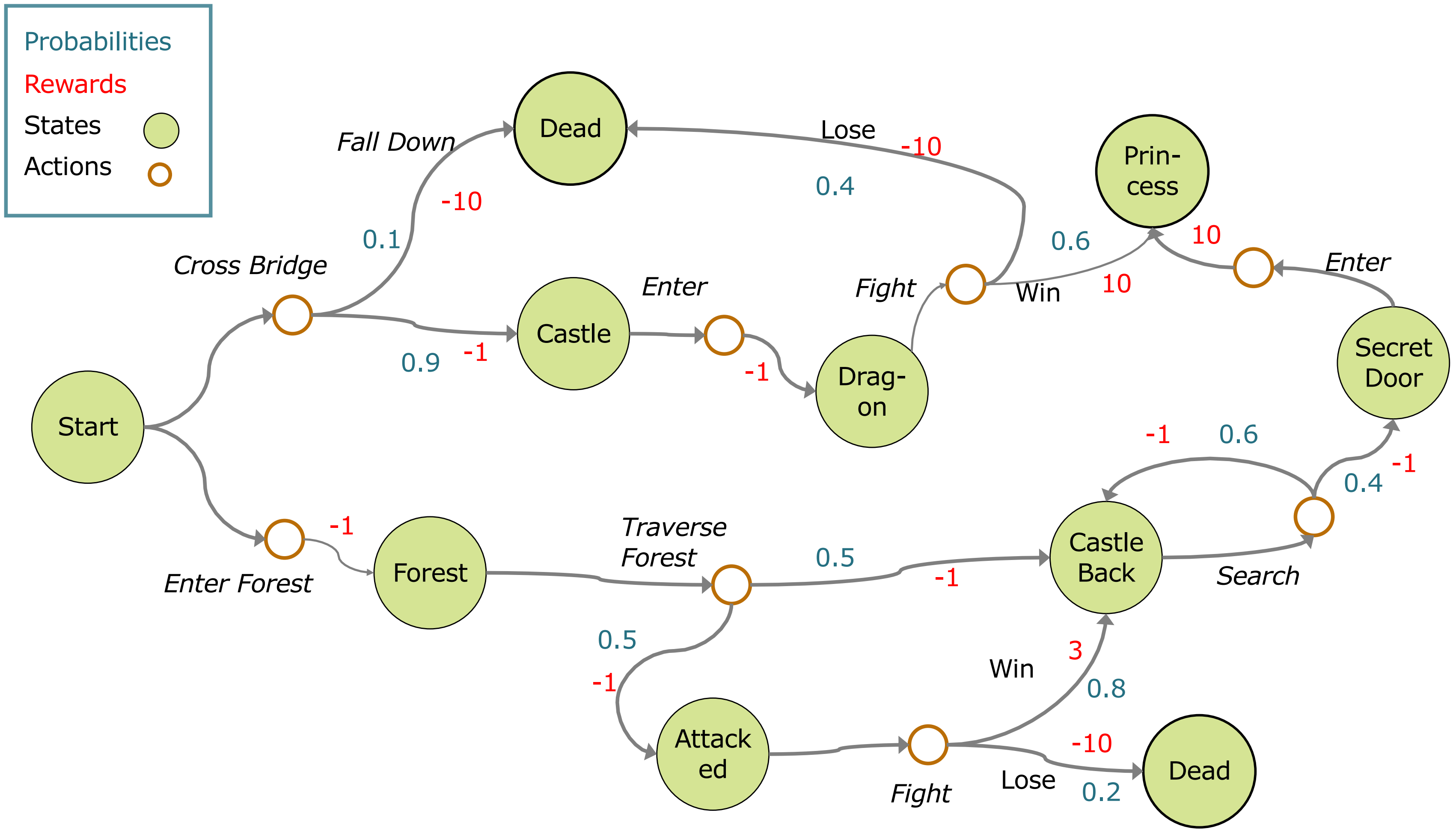
Markov Decision Process (MDP)

- (not Markov Process, which has no decisions))
- Markov property: *The future is independent of the past given the present*
- An Environment consists of states S_t
- An action (decision) a_t that can be taken in each state
- The action will give a reward and a new state

Which way



Markov Decision Process



Examples

Sample episodes:

Start → *Cross Bridge* → Castle → *Enter* → Dragon → *Fight* → Princess

Start → *Enter Forest* → Forest → *Traverse Forest* → Castle Back → *Search* → Castle Back → *Search* → Princess

What are the rewards?

How to solve this?

Goal: Find the sequence of best actions, i.e. maximize the cumulative reward

How would you solve the problem?

How to solve....

... depends on the information available:

- Do we know all the states, or do we have to discover them?
- Do we know the probabilities? Explicitly or can we sample from them?
- Do we have a full model of the problem?
- Can we start exploring at every state?
- ...

Definition of MDP

The dynamics of an MDP is defined as

$$p(s', r \mid s, a) \doteq \Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\}$$

(this can be viewed as a function of 4 parameters)

Goal (remains the same for MDP)

Maximizing the expected return:

$$G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T$$

Maximizing the *discounted return*:

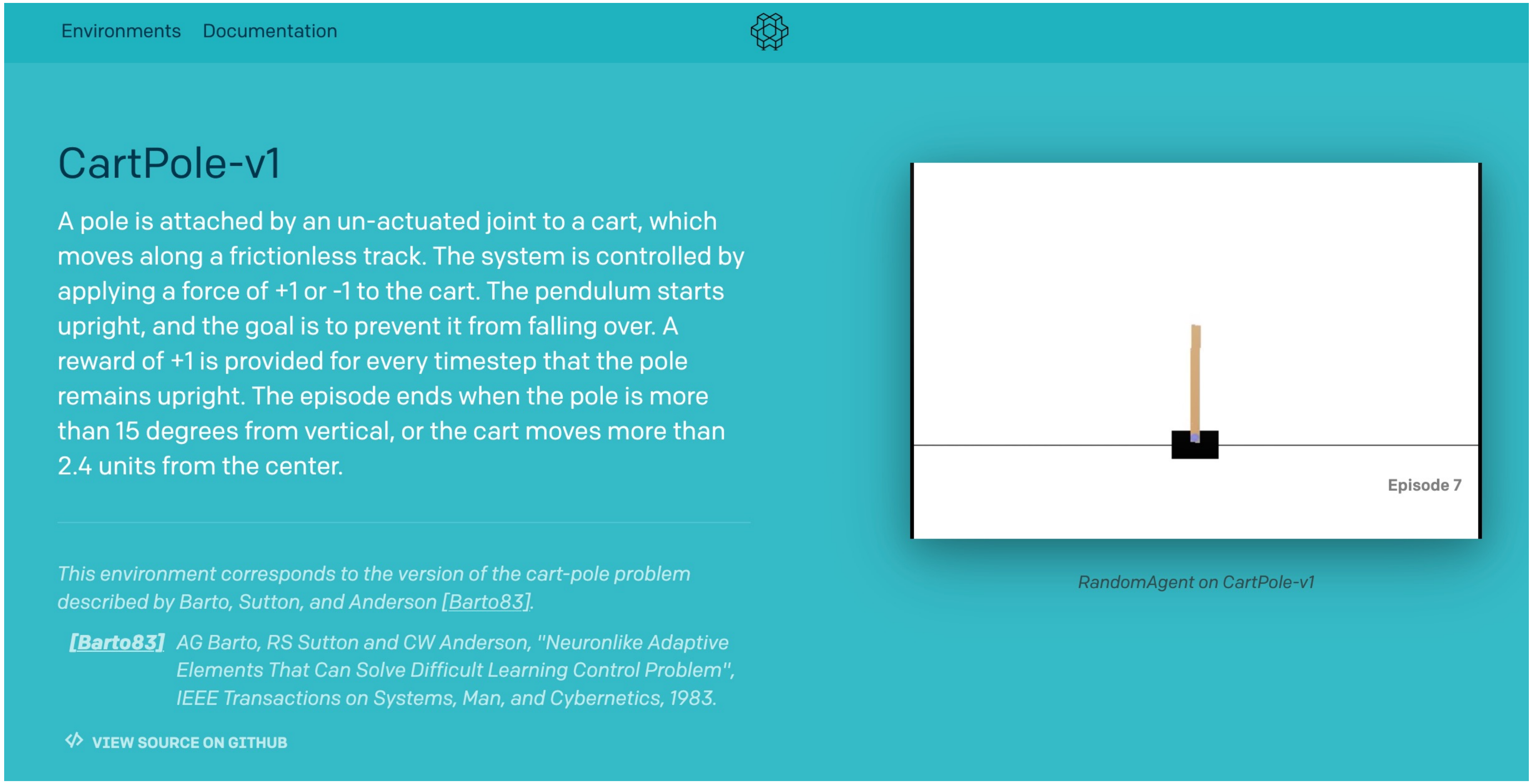
$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

$$0 \leq \gamma \leq 1$$

Episodic vs. Continuing tasks

- Many MDP are naturally episodic, where they end at a terminal state, these are *episodic tasks*
- Others are *continuous tasks* which do not end but run forever

MACH



CartPole-v1

A pole is attached by an un-actuated joint to a cart, which moves along a frictionless track. The system is controlled by applying a force of +1 or -1 to the cart. The pendulum starts upright, and the goal is to prevent it from falling over. A reward of +1 is provided for every timestep that the pole remains upright. The episode ends when the pole is more than 15 degrees from vertical, or the cart moves more than 2.4 units from the center.

This environment corresponds to the version of the cart-pole problem described by Barto, Sutton, and Anderson [Barto83].

[Barto83] *AG Barto, RS Sutton and CW Anderson, "Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problem", IEEE Transactions on Systems, Man, and Cybernetics, 1983.*

RandomAgent on CartPole-v1

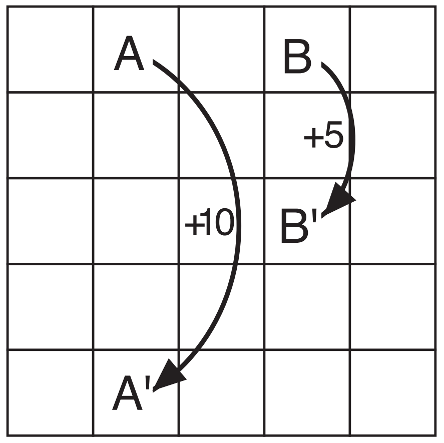
https://gym.openai.com/envs/CartPole-v1/

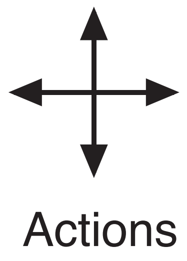
Recursive calculation of returns

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

Examples of MDPs

- Grid world environment
- After reaching A (or B) and agent is transferred to the state A' (or B') with the indicated reward





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

value function

Policy

A policy is a mapping from states to probabilities of selecting each possible action:

$$\pi(a|s) \doteq \Pr\{A_t = a | S_t = s\}$$

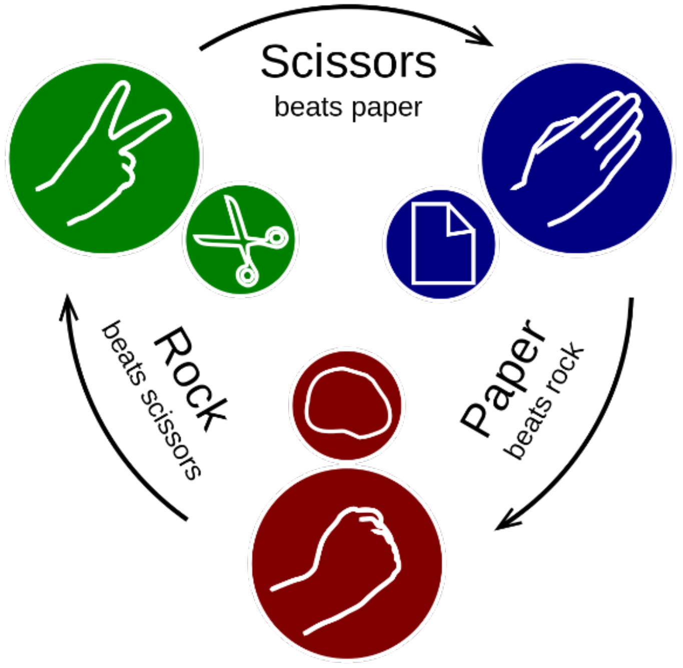
Reinforcement learning methods specify how the agent's policy is changed as a result of its experience.

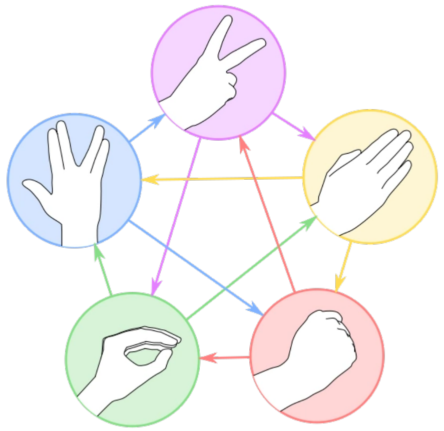
(Barton & Sutton)

Stochastic / Deterministic policies

- As defined before, a policy is generally **stochastic**, meaning that there are different possible actions taken under the policy with different probabilities for each action
- For example, the optimal policy for rock, paper, scissor is (1/3, 1/3, 1/3)

- In a **deterministic** policy, there is a single action a that is (always) taken in a state s





https://bigbangtheory.fandom.com

State-Value Function

The value function of a state s under a policy π is the expected return by

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s], \text{ for all } s \in \mathcal{S}$$

by following π from s

Action-value

The value of taking action a in state s under a policy π, is the expected return

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

by following π after taking action a

Bellman Equation

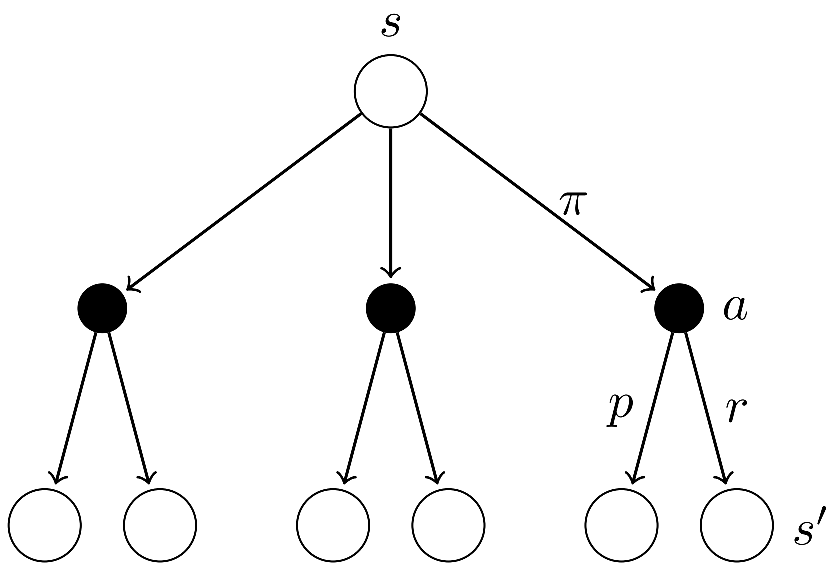
There is a fundamental, recursive calculation for the value-function:

$$v_\pi(s) \;=\; \sum_a \pi(a|s) \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma v_\pi(s')\big], \;\text{for all } s \in \mathcal{S}$$

This is the *Bellman equation (for the state-value function)*

Backup Diagram for the State-Value Function

- This recursive relationship can be visualized using a *backup diagram*:

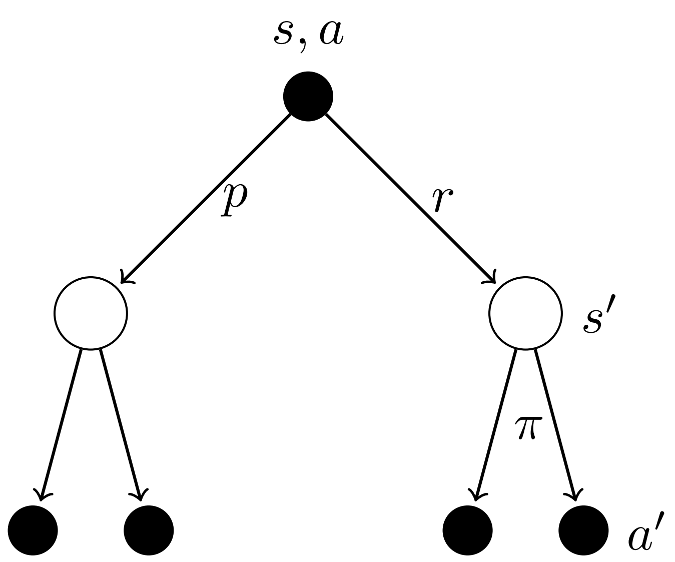


$$v_\pi(s) \;=$$

$$\sum_a \pi(a|s) \cdot$$

$$\sum_{s',r} p(s', r \mid s, a)[r + \gamma v_\pi(s')]$$

Backup Diagram for the Action-Value Function



How does the equation of the action-value function look like?

Optimal Policies

- We can compare policies and say that a policy π is better or equal to another policy π' if its expected return is greater than or equal to that of π.
- There is at least one policy that is better or equal to all the other policies. This is an *optimal* policy.

Optimal State-Value Function

- the optimal state-value function is then defined as

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

- and the optimal action-value function as

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

- which can also be written using the state-value function:

$$q_*(s,a) = \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a]$$

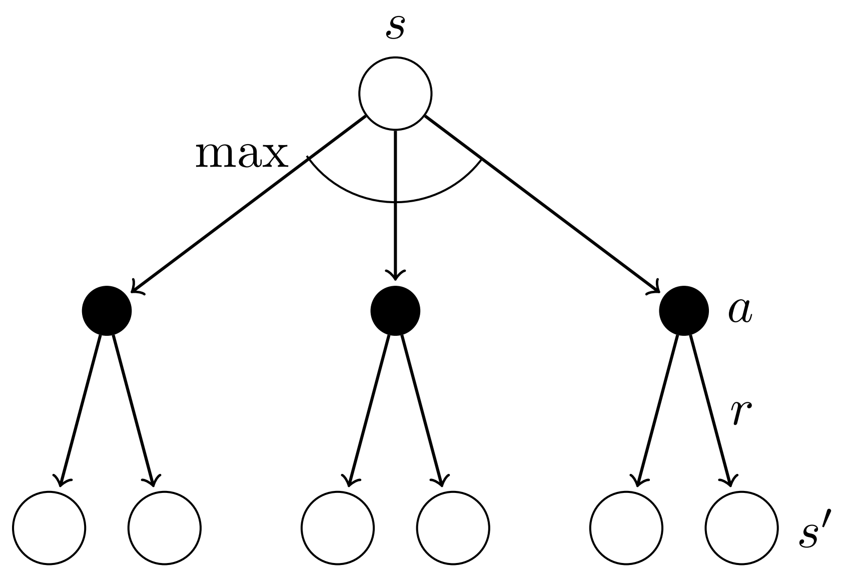
Bellman Optimality Equation for the State-Value Function

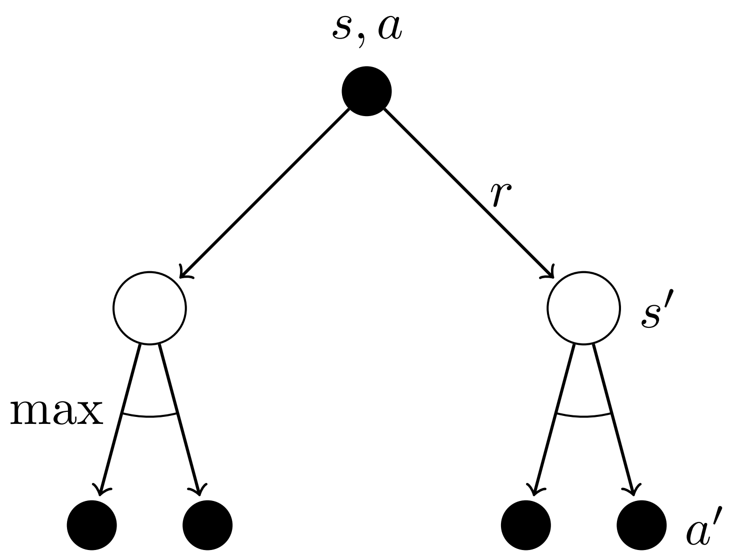
$$\begin{aligned}
v_*(s) &= \max_{a \in \mathcal{A}(s)} q_{\pi_*}(s,a) \\
&= \max_a \mathbb{E}_{\pi_*}[G_t \mid S_t = s, A_t = a] \\
&= \max_a \mathbb{E}_{\pi_*}[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a] \\
&= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a] \\
&= \max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma v_*(s')
\end{aligned}$$

Bellman Optimality Equation for the Action-Value Function

$$
\begin{aligned}
q_*(s,a) &= \mathbb{E}\Big[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \mid S_t = s, A_t = a\Big] \\
&= \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma \max_{a'} q_*(s', a')\right]
\end{aligned}
$$

Backup diagrams for the optimal functions

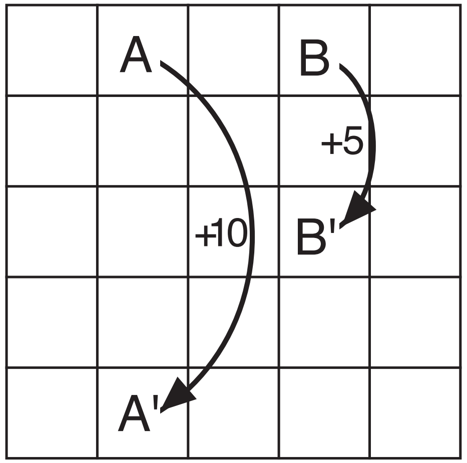




Optimal Policies

Given v_*, any policy that is *greedy* with respect to v_* is an optimal policy

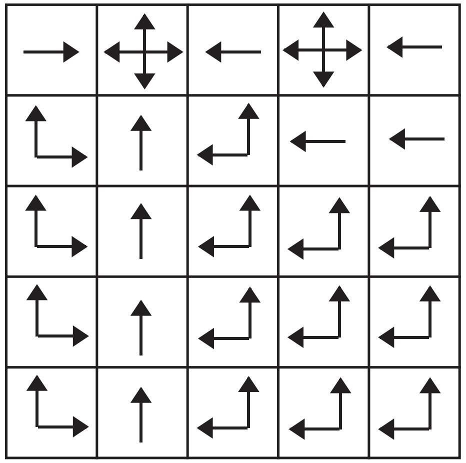
Example: Gridworld



Gridworld

22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

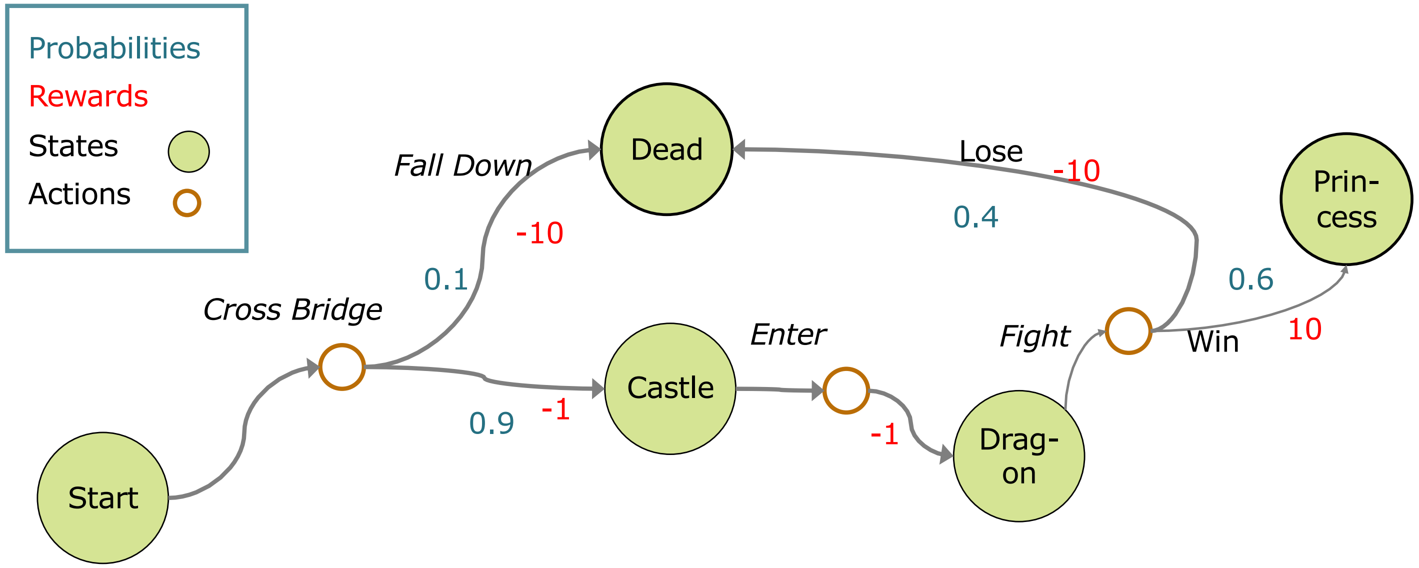
v_*



π_*

Example: Princess

What are the value functions for the states *Dragon*, *Castle* and *Start*?


Probabilities
Rewards
States ●
Actions ○
Dead
Fall Down
-10
0.1
Cross Bridge
Castle
0.9
-1
Enter
-1
Drag-on
Fight
Lose
-10
0.4
Prin-cess
0.6
Win
10
Start