



Technische Hochschule  
Ingolstadt

# Principles of Autonomy and Decision Making

(AI\_PrincAutonomy\_2808)

## Week 4: Markov Decision Processes (MDPs)

### Team:

- Prof. Dr. rer. nat. Lenz Belzner
- Chidvilas Karpenahalli Ramakrishna, M.Eng.
- Adithya Mohan, M.Eng.
- Zahra Zeinaly, Ph.D.

## Week 3 Recap: Uninformed and Informed Search



- Uninformed Search
  - Depth-first search (DFS)
    - No cost consideration
  - Breadth-first search (BFS)
    - No cost consideration
  - Uniform cost search (UCS)
    - Cost is considered
- Informed Search
  - Greedy search
    - Heuristic function
  - A\* search
    - Heuristic + cost function
- What have we not considered?
  - Uncertainty
  - The above algorithms work in deterministic environments

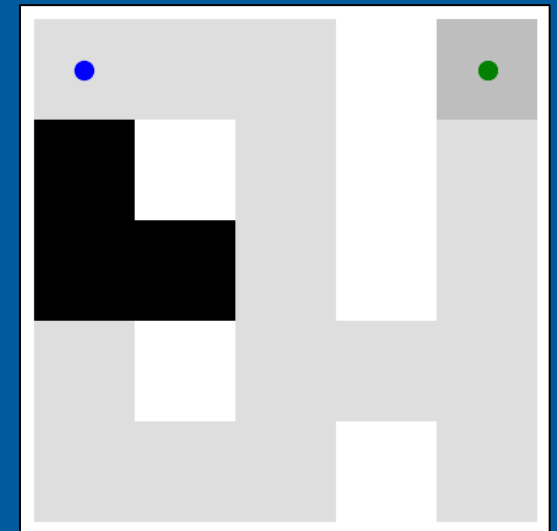


Fig. 1 A\* algorithm

## Decision Making Under Uncertainty



- Deterministic search:
  - Same input always produces the same output
  - No uncertainty in outcomes i.e.  $P(s'|s, a) = 1$
  - Example: Frozen Lake-v1 without randomness (is\_slippery=False)
- Non-deterministic search:
  - Same input does not always produce the same output
  - Uncertainty in outcomes i.e.  $P(s'|s, a) \neq 1$
  - Example: Frozen Lake-v1 with randomness (is\_slippery=True)
- How do we formulate non-deterministic search problems?
  - Markov Decision Process (MDPs)



Fig. 1 Probabilistic Frozen lake with consecutive downward actions

# Markov Decision Processes (MDPs)



- Definition:
  - „MDP is a mathematical framework used for modeling decision making in situations where outcomes are partly random and partly under the control of a decision maker. “
- Markov property:
  - Action outcomes depend only on the current state i.e. history is irrelevant
  - $P(S_{t+1} = s' | S_t = s_t, A_t = a_t, S_{t-1} = s_{t-1}, A_{t-1} = a_{t-1}, \dots, S_0 = s_0) = P(S_{t+1} = s' | S_t = s_t, A_t = a_t)$
  - All relevant information of history is embedded in the current state  $s_t$



Fig. 1 Andrey Markov

# Markov Decision Processes (MDPs)



- Components of an MDP:
  - States:
    - $s \in S$
    - A Start state  $s_0$
    - A Terminal state (optional)
  - Actions:
    - $a \in A$
  - Reward function:
    - $R(s, a, s')$  or  $R(s, a)$
    - In MDPs, costs and/or heuristics are called rewards
    - „Living reward“ and „Termination“ rewards
  - Transition function:
    - Here,  $T(s, a, s') = P(s'|s, a)$
    - Models the uncertainty

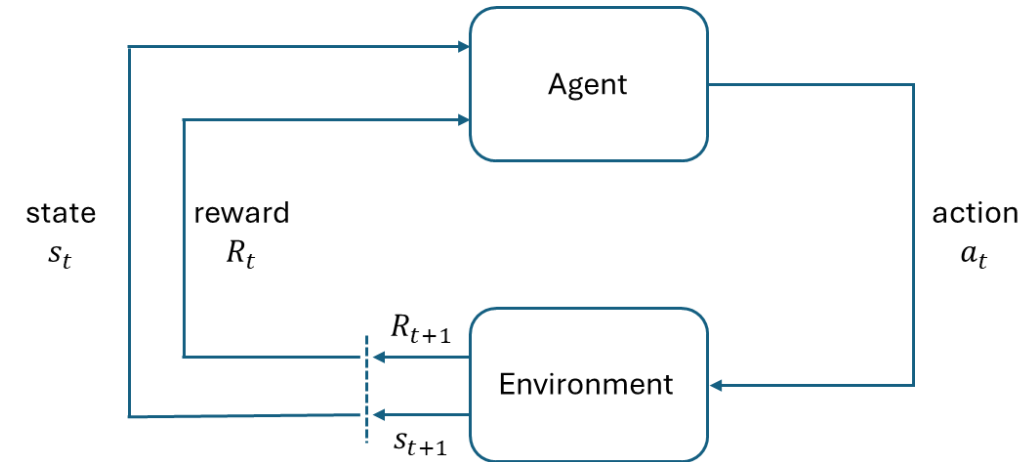


Fig. 1 Agent-environment interaction cycle

## MDP: Rewards



- Rewards:
  - $R(s, a, s')$
  - Positive, negative or zero
  - „Living“ reward and „termination“ reward
- Rewards influence the outcome
- Exercise:
  - Let's consider a simple grid-world as shown in Fig. 1 with a start state, a goal state and some hell-states
  - Aim: To gather the most rewards possible before termination
  - Decide a path in the 5x5 grid based on the following reward structures. *Which path would you take?*
    - Reward structure 1: goal=+10, hell-state=-1
    - Reward structure 2: goal=+10, living reward=-0.01, hell-state=-1
    - Reward structure 3: goal=+10, living reward=-3, hell-state=-1



Fig. 1 A simple grid-world



## MDP: Transition Function



- Definition:
  - „A transition function in an MDP describes the probability of moving from one state to another state given an action. It quantifies the dynamics of the environment. “
  - Also called as the „model“ or the „dynamics“
- A transition is denoted as  $(s, a, s')$
- A transition function is denoted as  $T(s, a, s')$ 
  - $T(s, a, s') = P(s' | s, a)$
- After transition, the agent gets a reward  $R(s, a, s')$

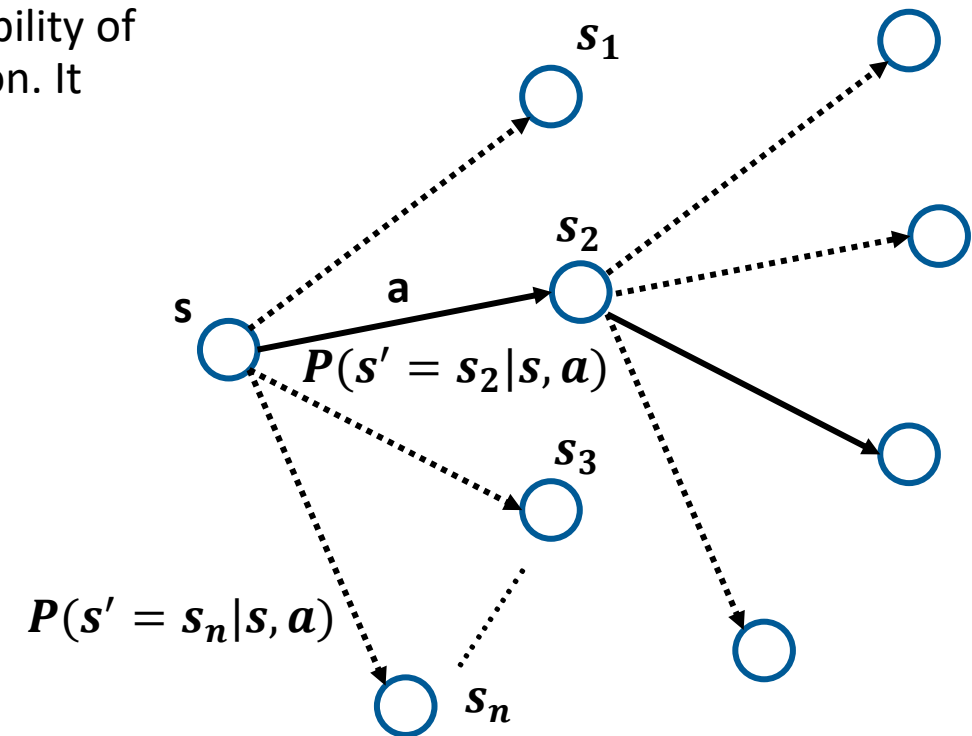


Fig 1. MDP search tree

# MDP: Policy



- Definition:
  - „A policy is a strategy used by an agent to decide the action to take in each state of an environment.
- Policies differentiate search problems from MDPs
  - A plan is a sequence of actions. It works in a search problem as everything is deterministic.
  - We cannot have a plan in MDP because it may work or may not work
  - In MDPs something analogous to a plan is what we call as a „policy“
- For an MDP, we need an **optimal policy** denoted as  $\pi^*: S \rightarrow A$
- The objective of a policy is to maximize the cumulative reward the agent receives over time
- Policy can be
  - Deterministic (one action for a state)
  - Stochastic (action sampled from a distribution)

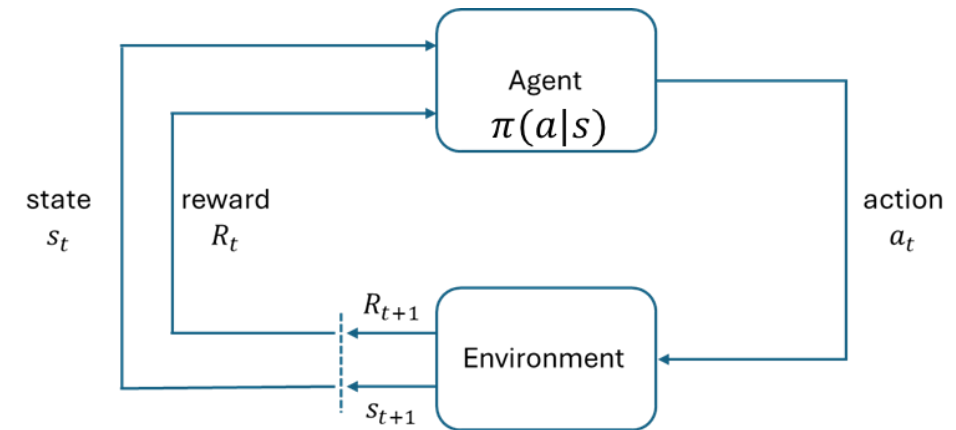


Fig 1. Agent-environment interaction cycle



## Utilities of Sequences



- Utility definition:
  - „The cumulative reward that an agent receives through a sequence of actions, states, and transitions. “
- We usually discount the rewards over time as we prefer immediate rewards over future rewards – discount factor ( $\gamma$ )
  - Stochastic environment
  - We use an **exponential** decay of rewards
- Exercise 1: Which sequence should the agent choose if the reward sequence is  $[1,2,3]$  or  $[3,2,1]$ ?
- Exercise 2: Calculate the discounted cumulative reward from state  $s_{(4,2)}$  following a policy  $\pi$  highlighted in yellow for the grid-world shown in Fig. 1
  - Let  $\gamma = 0.1$ ,  $T(s, a, s') = 1$  and the reward structure be goal=+100, living reward=-1, hell-state=-10?

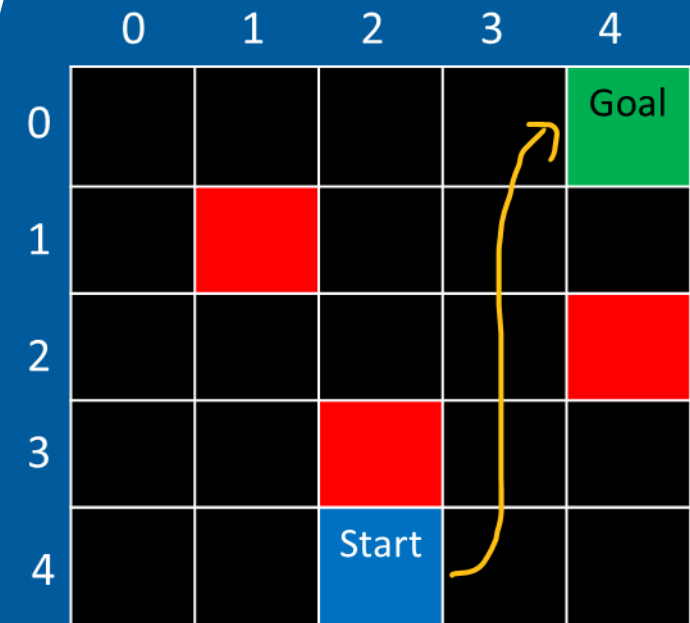


Fig. 1 A simple grid-world

## Solving MDPs



- What do we want?
  - Input: MDP
  - Output: An optimal policy  $\pi^*(s)$
- Definition of optimal quantities:
  - Optimal value of a state  $V^*(s)$ : Expected utility starting in state  $s$  and acting optimally
  - Optimal Q-value of a state-action pair  $Q^*(s, a)$ : Expected utility starting in state  $s$  and taking an action  $a$  and acting optimally thereafter
  - Optimal policy  $\pi^*(s)$ : Optimal strategy from state  $s$

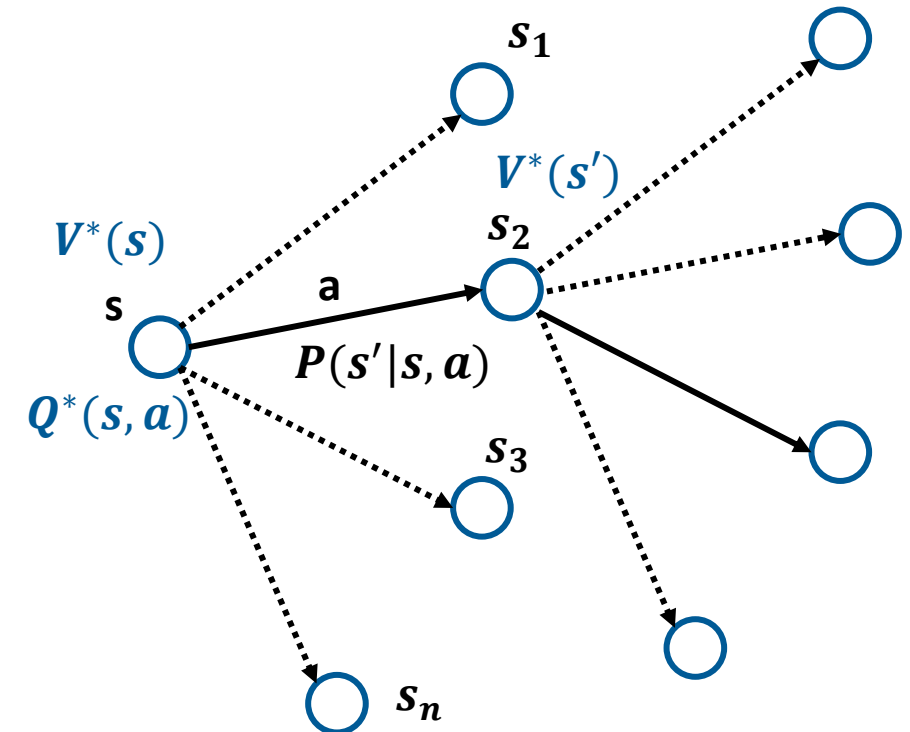


Fig 1. MDP search tree

## Solving MDPs: Value of a State



- What is the optimal value  $V^*(s)$ ?
  - $V^*(s) = \max_a Q^*(s, a)$
  - $Q^*(s, a) = \sum_{s'} T(s, a, s') [ R(s, a, s') + \gamma V^*(s') ]$
  - **Bellman equation:**
    - $V^*(s) = \max_a \sum_{s'} P(s'|a, s) [ R(s, a, s') + \gamma V^*(s') ]$
- How to solve the Bellman equation?
  - Value-iteration and policy-iteration (week 5)
  - Dynamic programming (week 6)
  - Reinforcement Learning (week 8 to week 11)

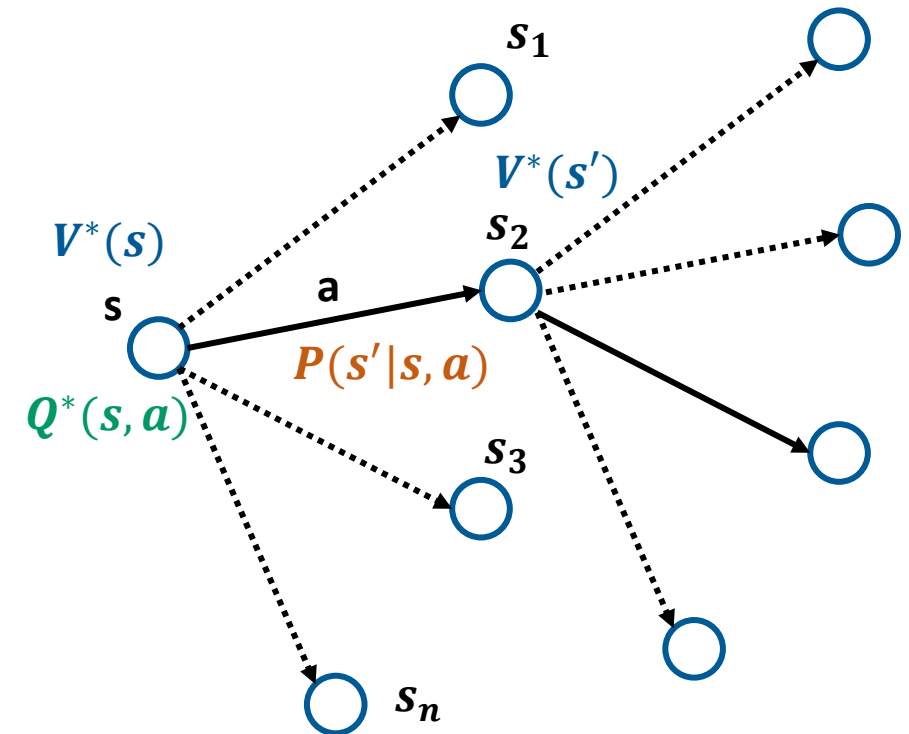


Fig 1. MDP search tree