CS 232: Artificial Intelligence

Fall 2023

Prof. Carolyn Anderson
Wellesley College

Reminders

Start early on these assignments!

- I have help hours today (4-5:30)
- Lyra has help hours tomorrow

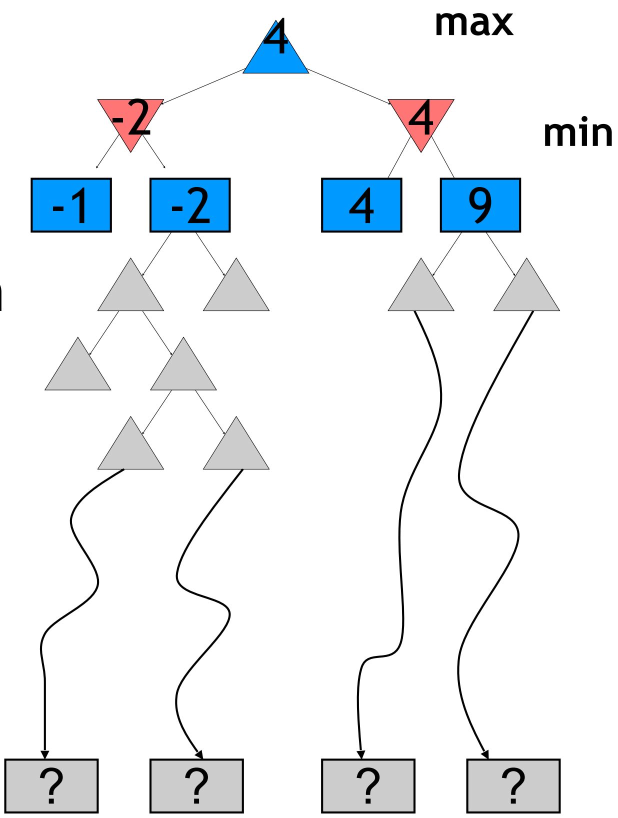
Recap

Minimax Summary

- Rank final game states by their final scores (for tic-tac-toe or chess: win, draw, loss).
- Rank intermediate game states by whose turn it is and the available moves.
  - If it's X's turn, set the rank to that of the *maximum* move available. If a move will result in a win, X should take it.
  - If it's O's turn, set the rank to that of the *minimum* move available. If a move will result in a loss, X should avoid it.

Resource Limits

- Problem: In realistic games, cannot search to leaves!
- Solution: Depth-limited search
  - Instead, search only to a limited depth in the tree
  - Replace terminal utilities with an evaluation function for non-terminal positions



Uncertain Outcomes

Stochastic Transition Model

$T(s, a) \rightarrow s'$

$T(s, a, s') \rightarrow p(s' | s, a)$

In our search algorithms so far, the transition model was deterministic and described the outcome of each action in each state.

The transition function is sometimes written as T(s, a, s'), or explicitly as a probability:

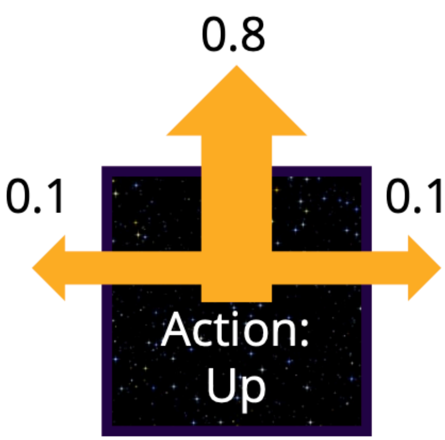
The probability of

arriving in state **s'**

p(s' | s, a)

given that

we are in state **s** and we selected action **a**

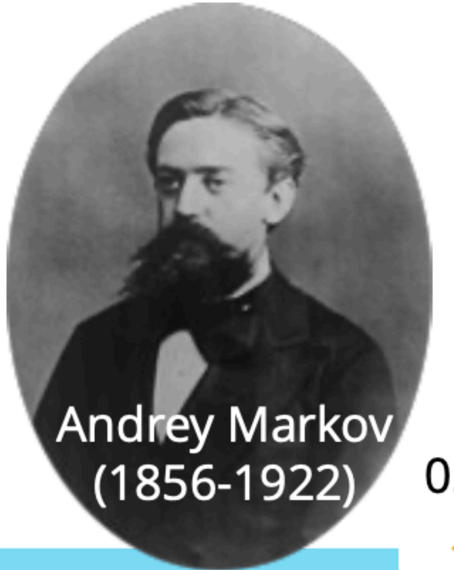


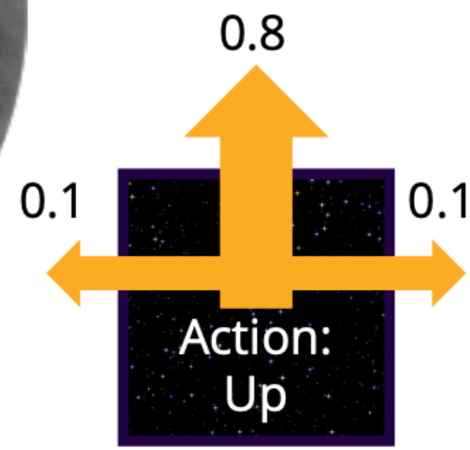
Slides adapted from Chris Callison-Burch

Stochastic Transition Model

In our search algorithms so far, the transition model was deterministic and described the outcome of each action in each state.

The transition function is sometimes written as T(s, a, s'), or explicitly as a probability:



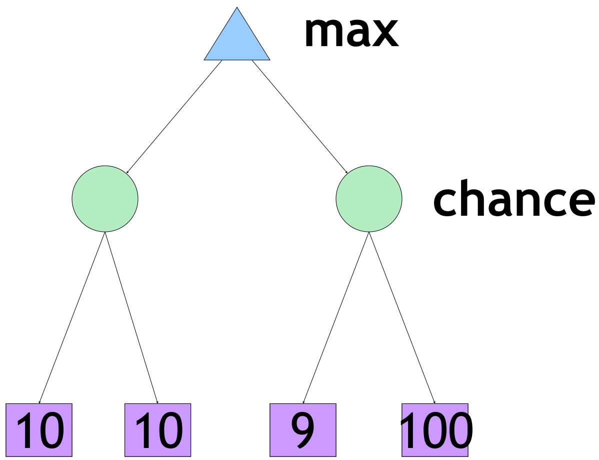


p(s' | s, a)

Transitions are **Markovian**: the probability of arriving in s' only depends on s and not the history of earlier states.

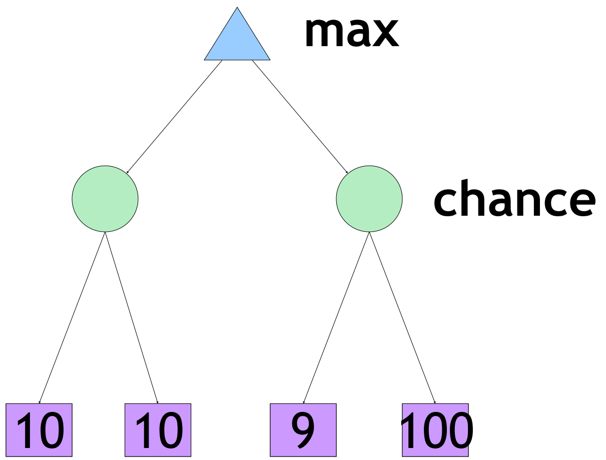
Expectimax Search

- Many reasons that outcomes are unpredictable:
  - Explicit randomness: rolling dice
  - Unpredictable opponents: the ghosts respond randomly
  - Actions can fail: when moving a robot, wheels might slip



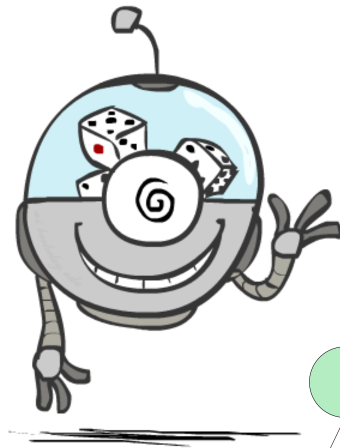
Expectimax Search

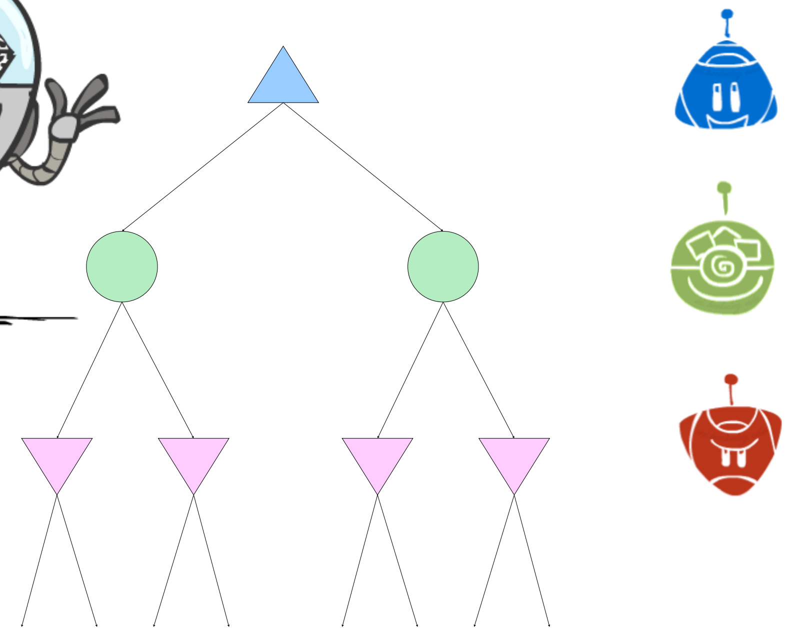
- Expectimax search: compute the average score under optimal play
  - Max nodes as in minimax search
  - Chance nodes are like min nodes but the outcome is uncertain
  - Calculate their expected utilities
  - I.e. take weighted average (expectation) of children



Mixed Layer Types

- E.g. Backgammon
- Expectiminimax
  - Environment is an extra "random agent" player that moves after each min/max agent
  - Each node computes the appropriate combination of its children





Example: Backgammon

- Dice rolls increase *b*: 21 possible rolls with 2 dice
  - Backgammon ≈ 20 legal moves
  - Depth 2 = $20 \times (21 \times 20)^3 = 1.2 \times 10^9$
- As depth increases, probability of reaching a given search node shrinks
  - So usefulness of search is diminished
  - So limiting depth is less damaging
  - But pruning is trickier…
- Historic AI: TDGammon uses depth-2 search + very good evaluation function + reinforcement learning:
  world-champion level play
- 1st AI world champion in any game!

Image: Wikipedia

Stochastic Transitions

Stochastic Transition Model

In our search algorithms so far, the transition model was deterministic and described the outcome of each action in each state.

The transition function is sometimes written as T(s, a, s'), or explicitly as a probability:

Navigating an Asteroid Field

3 terminal states (only 1 goal)

Suppose we have a **fully-observable** 4x3 environment with goal states.

The millennium falcon begins in the start state and **picks an action at each time step**.

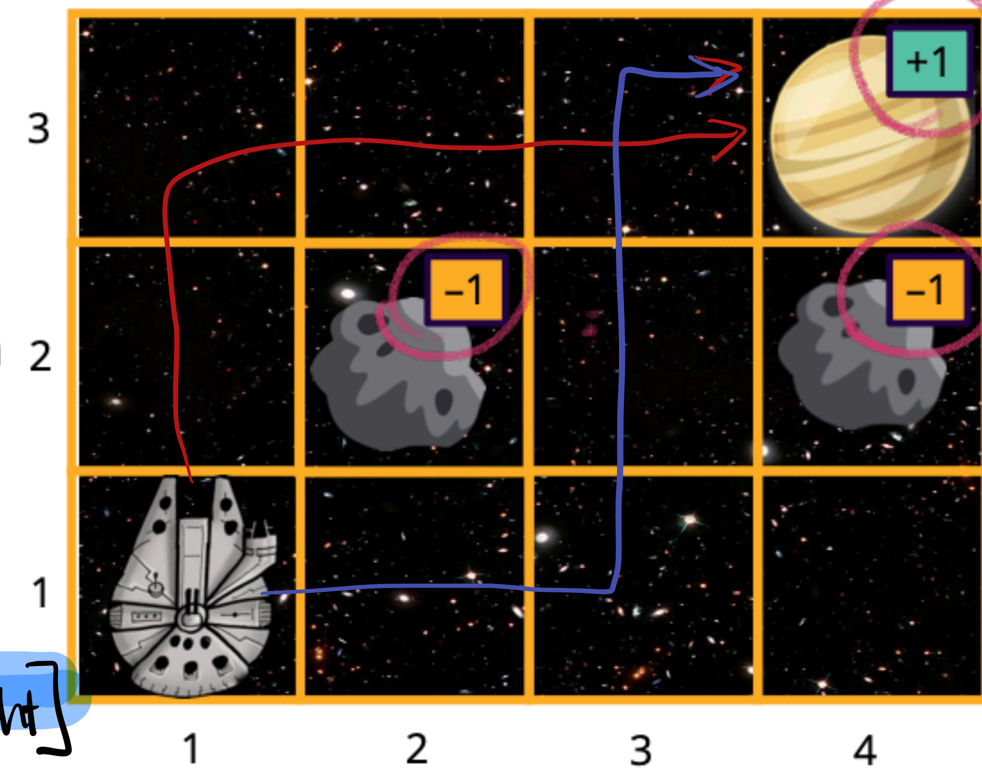
Actions: *Up, Down, Left, Right*

The game **terminates when it reaches a goal state** (+1 or -1).

If the environment were **deterministic**, the solution would be easy:

[*Up, Up, Right, Right, Right*]

[Right, Right, Up, Up, Right]



Slides adapted from Chris Callison-Burch

Navigating an Asteroid Field

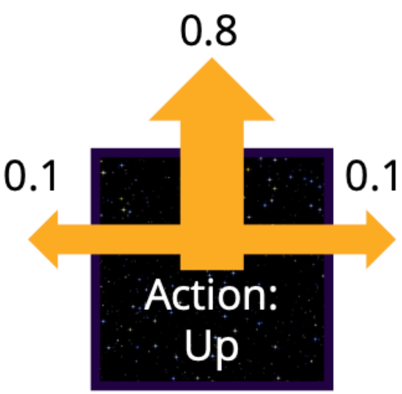
Instead of making the environment deterministic, we will make it **stochastic**.

If the Falcon selects the action *Up* then it only moves up 80% of the time.

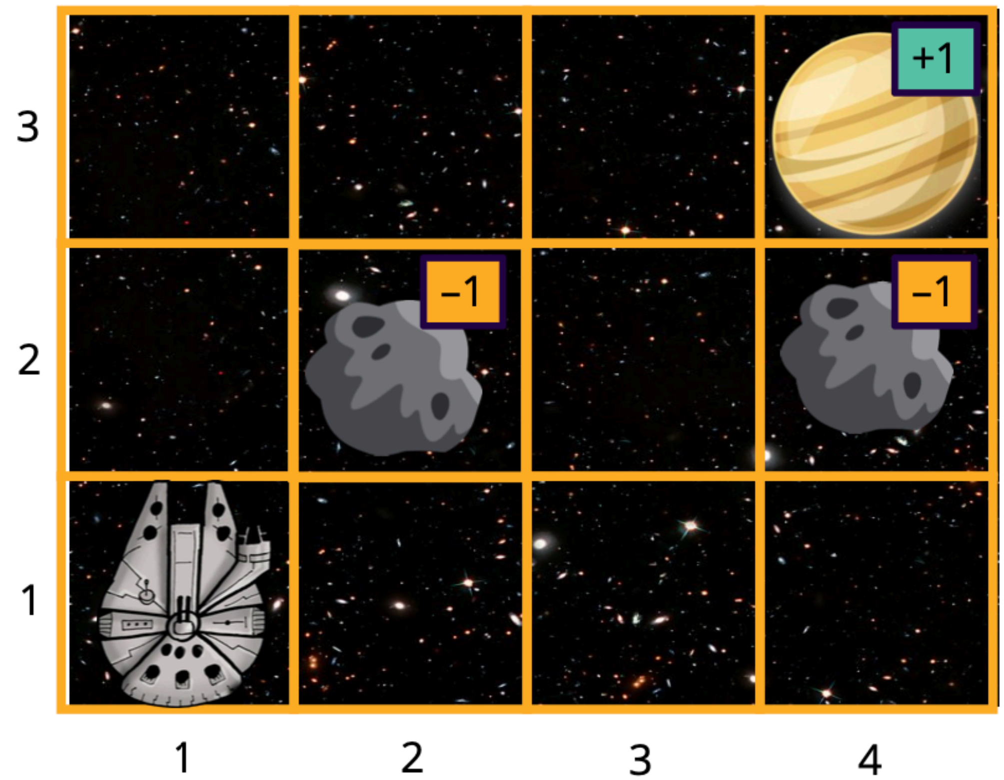
10% of the time the weird gravity fields cause it to veer off to the left or right.

stochastic

Transition Model:



Given any state s,



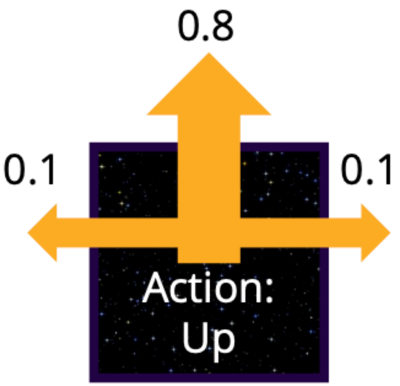
Navigating an Asteroid Field

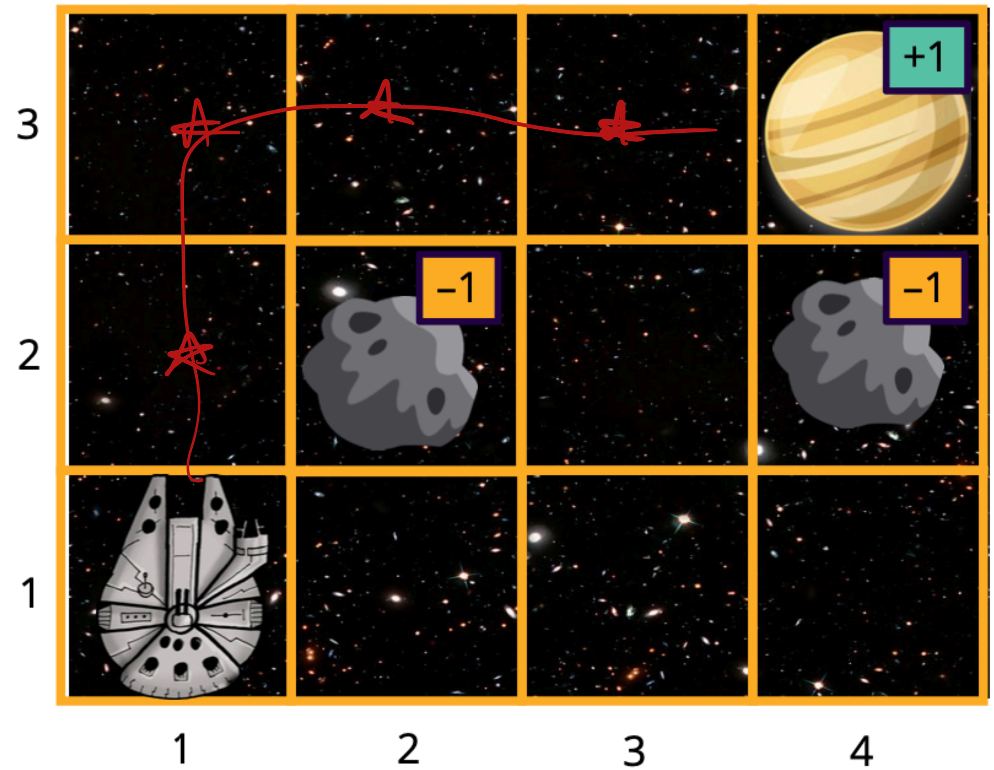
For action sequence

[Up, Up, Right, Right, Right],

what's the probability that the millennium falcon reaches the intended goal?

Transition Model:





Navigating an Asteroid Field

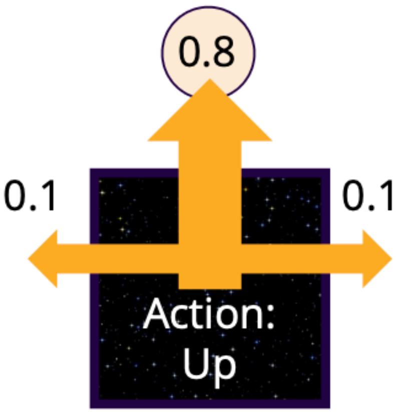
For action sequence

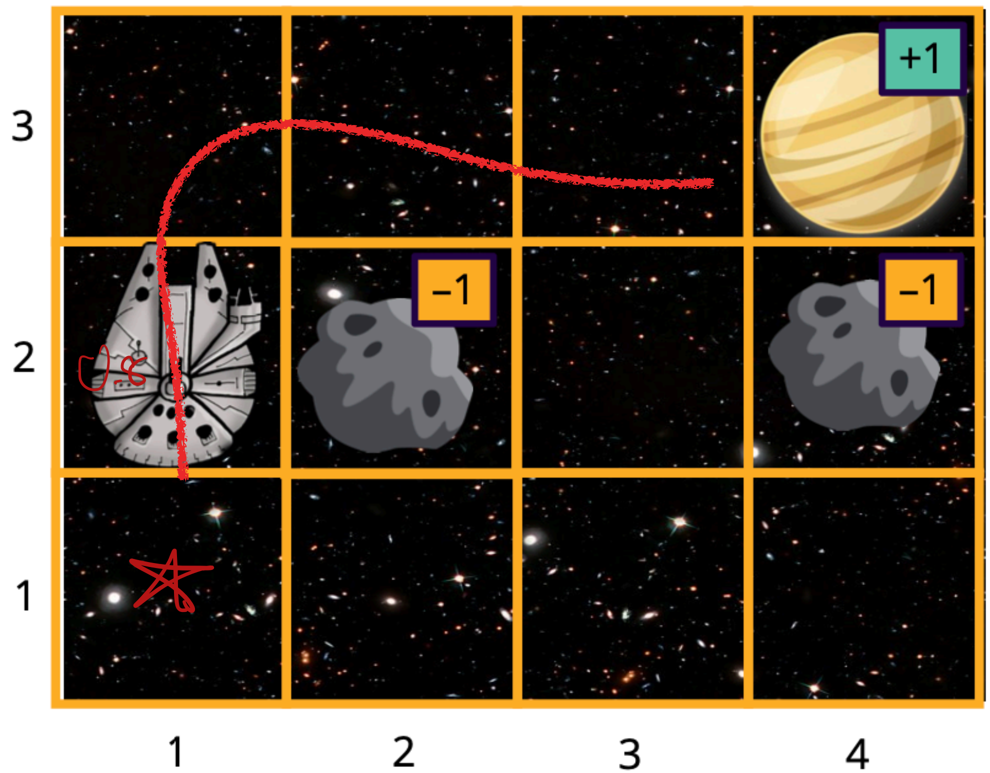
[Up, Up, Right, Right, Right],

what's the probability that the millennium falcon reaches the intended goal?

0.8

Transition Model:





Slides adapted from Chris Callison-Burch

Navigating an Asteroid Field

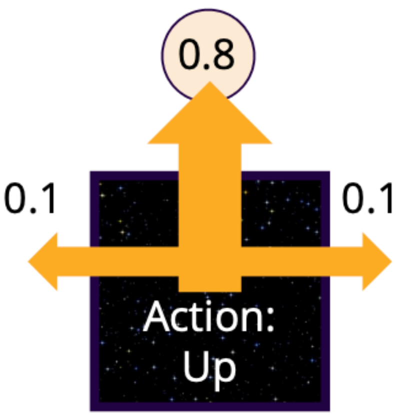
For action sequence

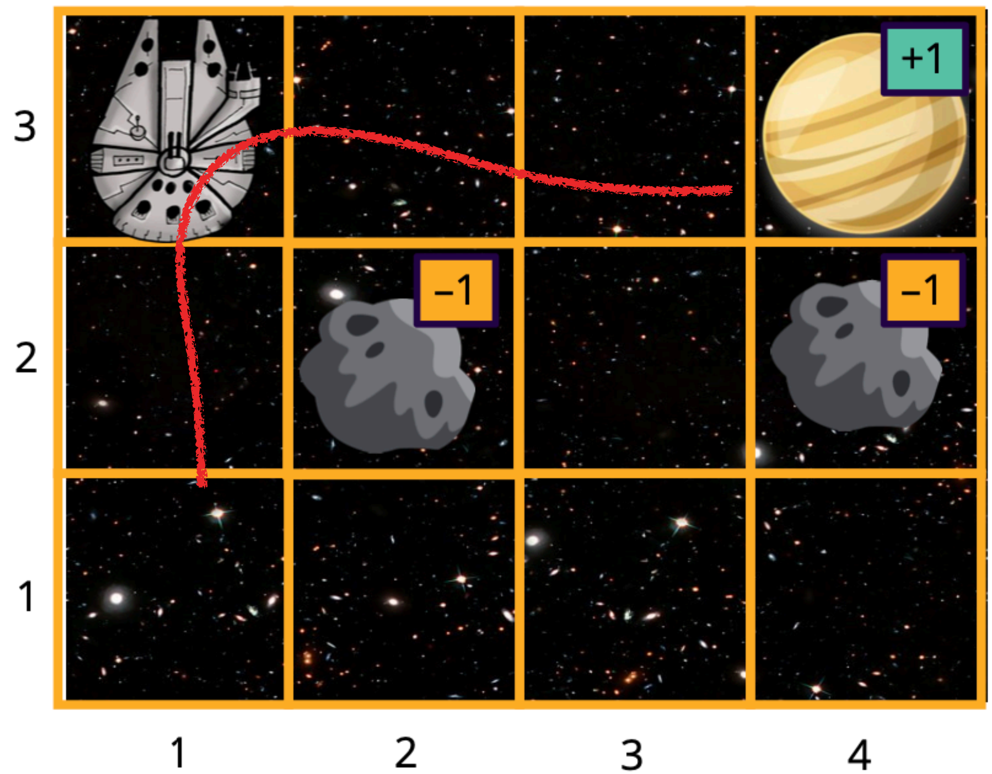
[*Up*, *Up*, *Right*, *Right*, *Right*],

what's the probability that the millennium falcon reaches the intended goal?

0.8 * 0.8

Transition Model:





Slides adapted from Chris Callison-Burch

Navigating an Asteroid Field

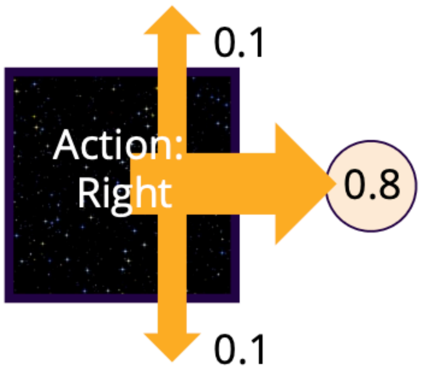
For action sequence

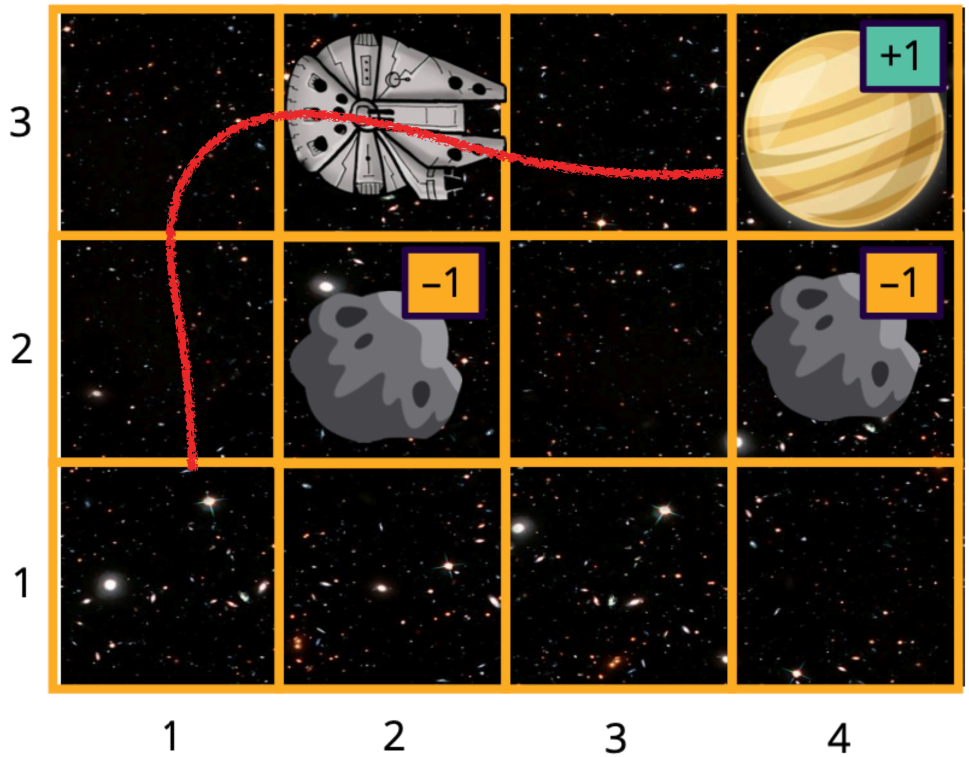
[*Up, Up, Right, Right, Right*],

what's the probability that the millennium falcon reaches the intended goal?

0.8 * 0.8 * 0.8

Transition Model:





Slides adapted from Chris Callison-Burch

Navigating an Asteroid Field

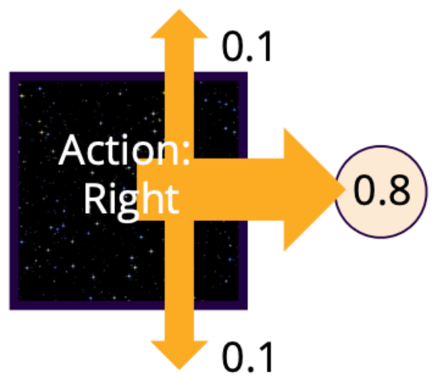
For action sequence

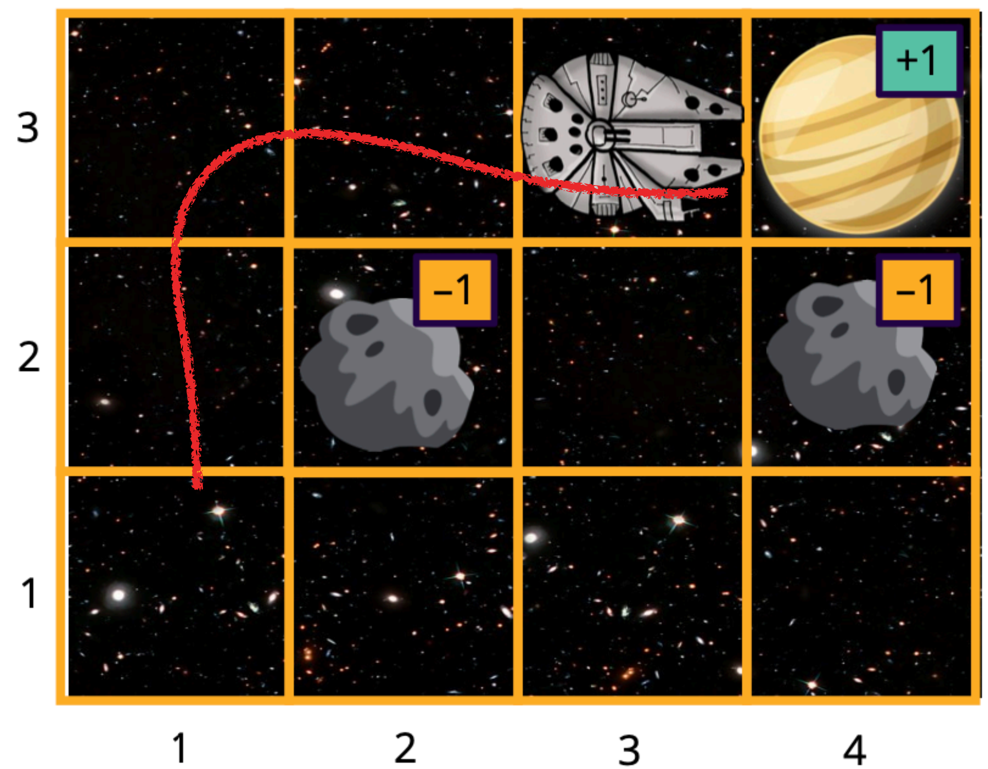
[*Up, Up, Right, Right, Right*],

what's the probability that the millennium falcon reaches the intended goal?

0.8 * 0.8 * 0.8 * 0.8

Transition Model:





Slides adapted from Chris Callison-Burch

Navigating an Asteroid Field

For action sequence

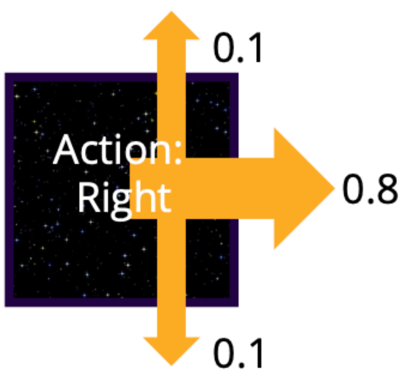
[Up, Up, Right, Right, Right],

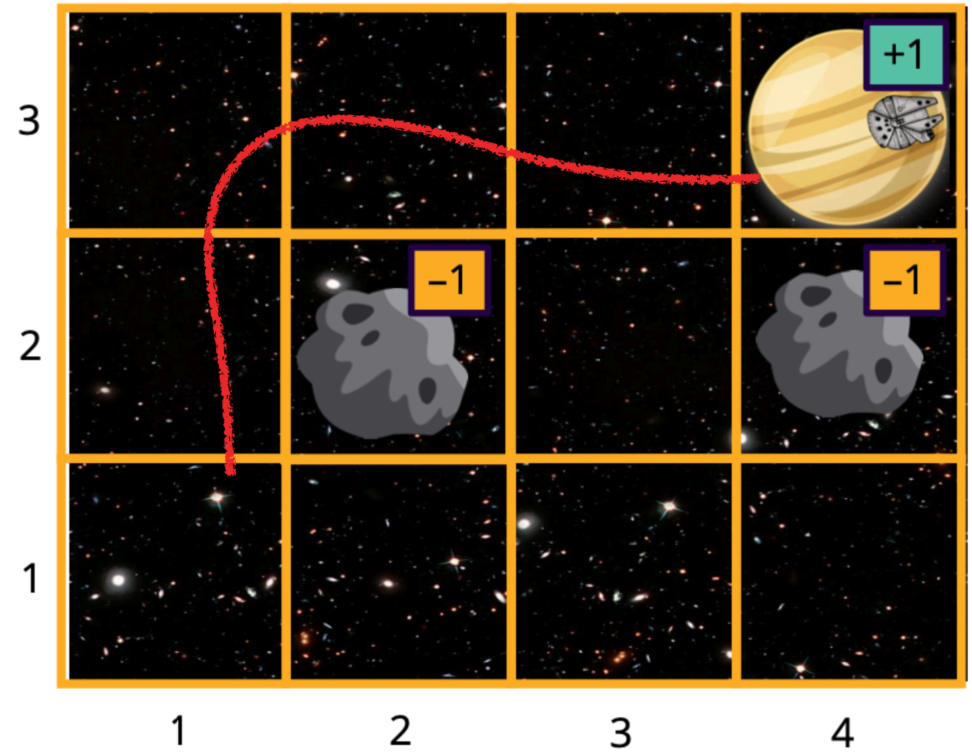
what's the probability that the millennium falcon reaches the intended goal?

0.8 * 0.8 * 0.8 * 0.8 * 0.8

= 0.32768

Transition Model:





Slides adapted from Chris Callison-Burch

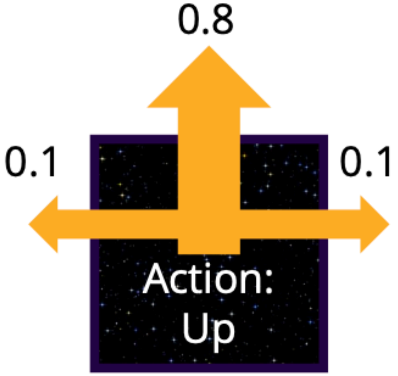
Navigating an Asteroid Field

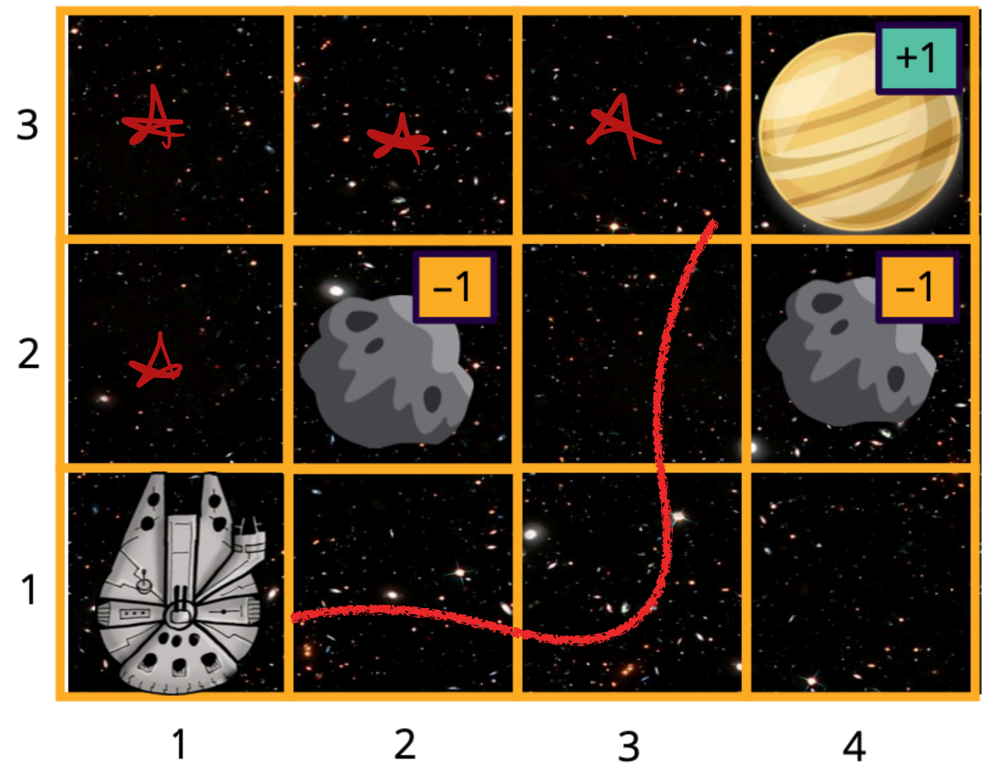
For action sequence

[*Up, Up, Right, Right, Right*],

what's the probability that the millennium falcon reaches the intended goal?

Transition Model:





Slides adapted from Chris Callison-Burch

Navigating an Asteroid Field

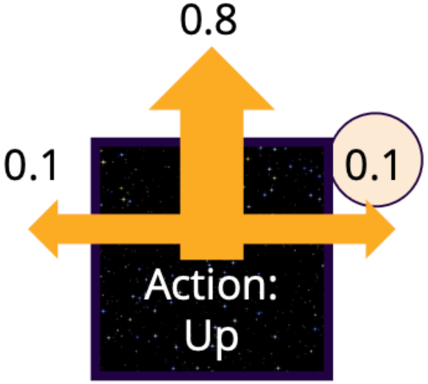
For action sequence

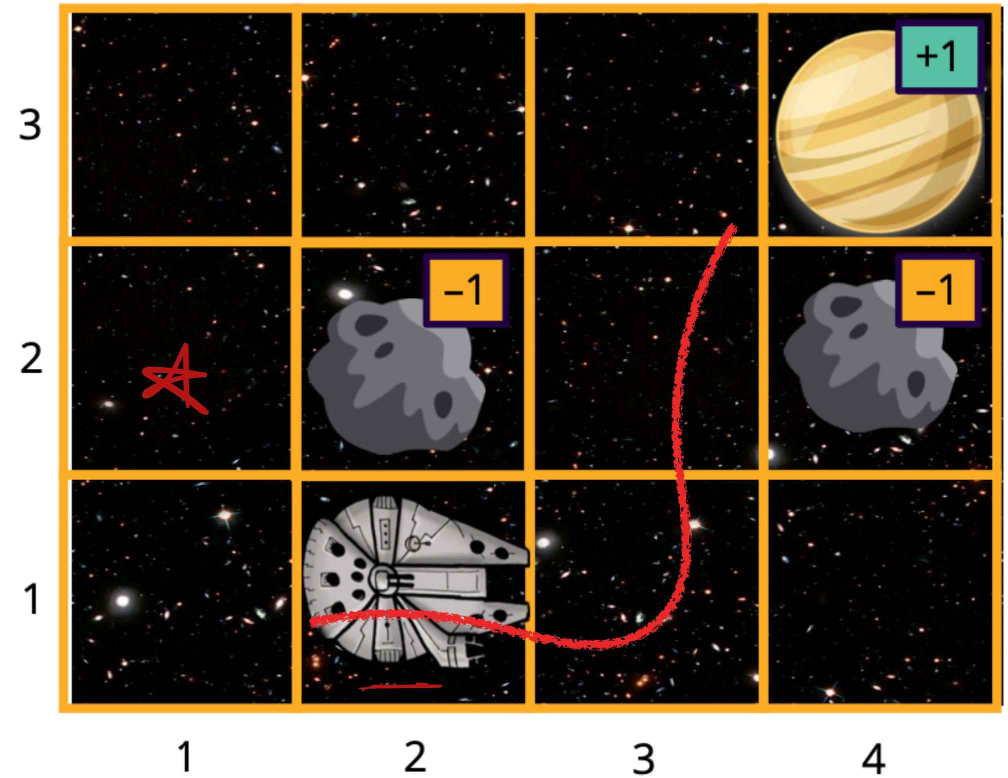
[Up, Up, Right, Right, Right],

what's the probability that the millennium falcon reaches the intended goal?

0.1

Transition Model:





Navigating an Asteroid Field

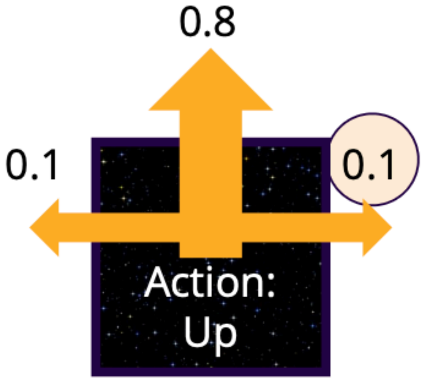
For action sequence

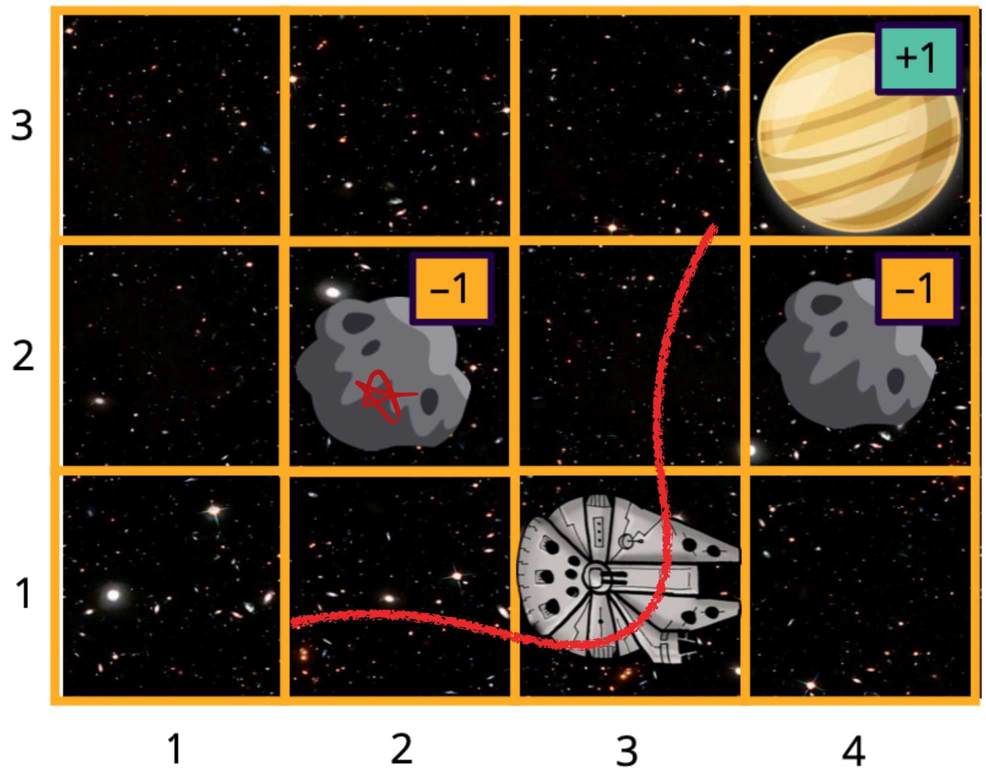
[Up, Up, Right, Right, Right],

what's the probability that the millennium falcon reaches the intended goal?

0.1 * 0.1

Transition Model:





Slides adapted from Chris Callison-Burch

Navigating an Asteroid Field

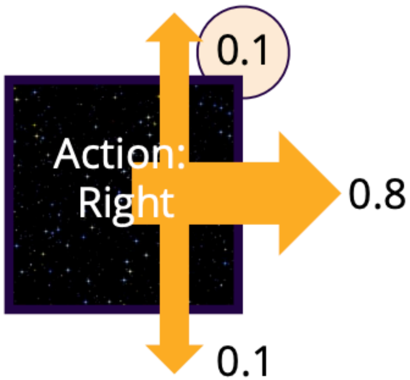
For action sequence

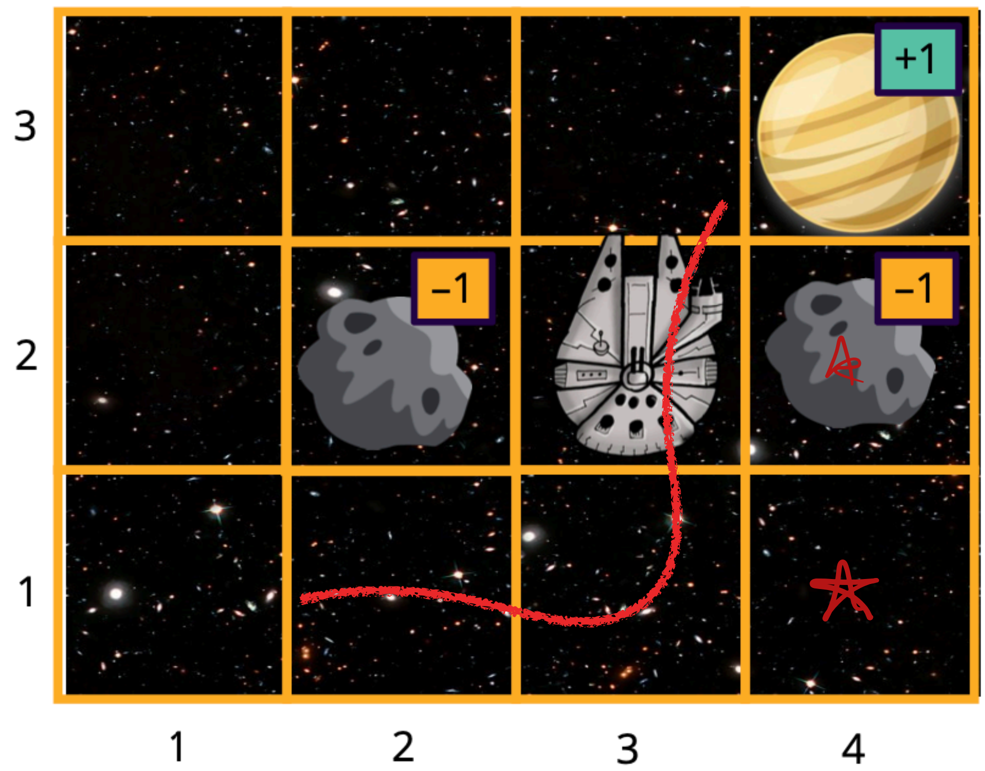
[*Up, Up, Right, Right, Right*],

what's the probability that the millennium falcon reaches the intended goal?

0.1 * 0.1 * 0.1

Transition Model:





Navigating an Asteroid Field

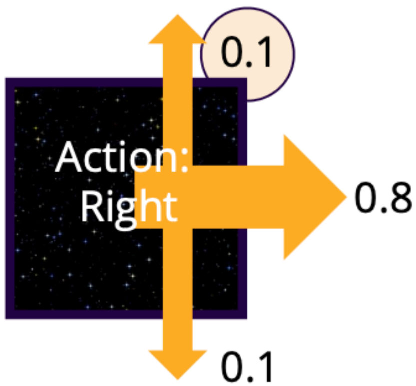
For action sequence

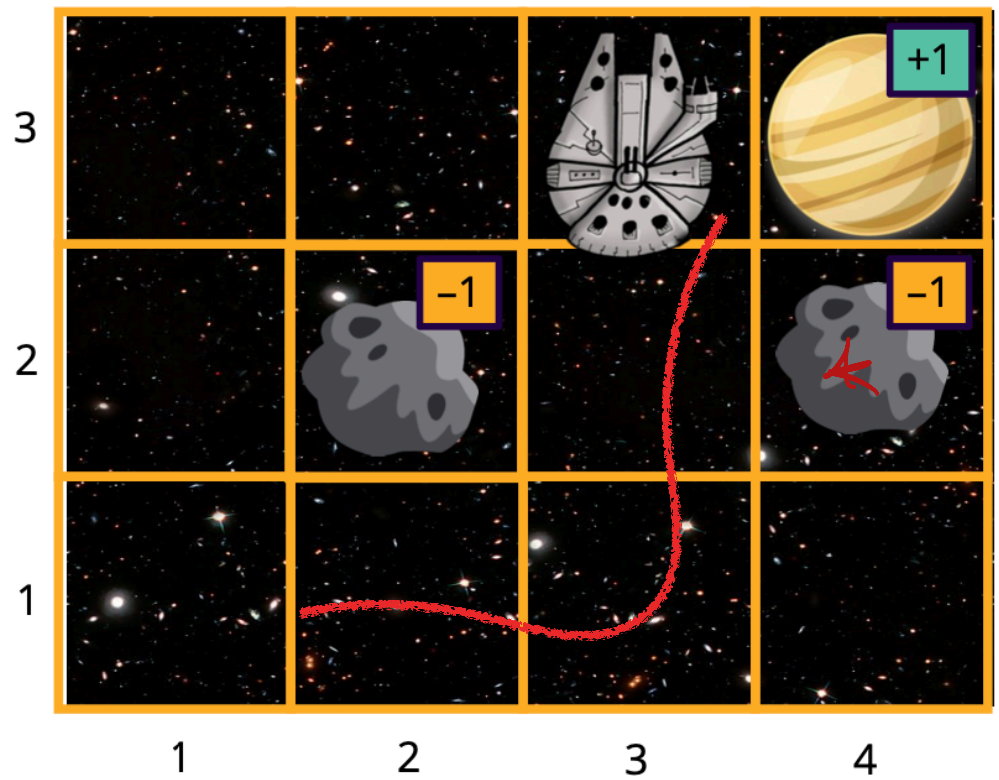
[*Up, Up, Right, Right, Right*],

what's the probability that the millennium falcon reaches the intended goal?

0.1 * 0.1 * 0.1 * 0.1

Transition Model:





Slides adapted from Chris Callison-Burch

Navigating an Asteroid Field

For action sequence

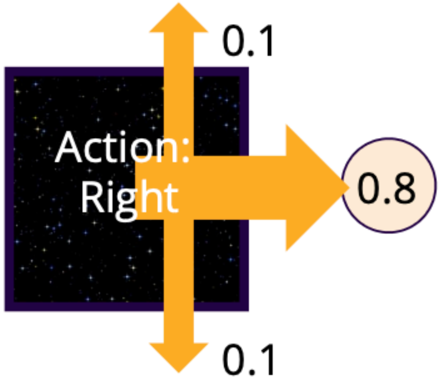
[*Up, Up, Right, Right, Right*],

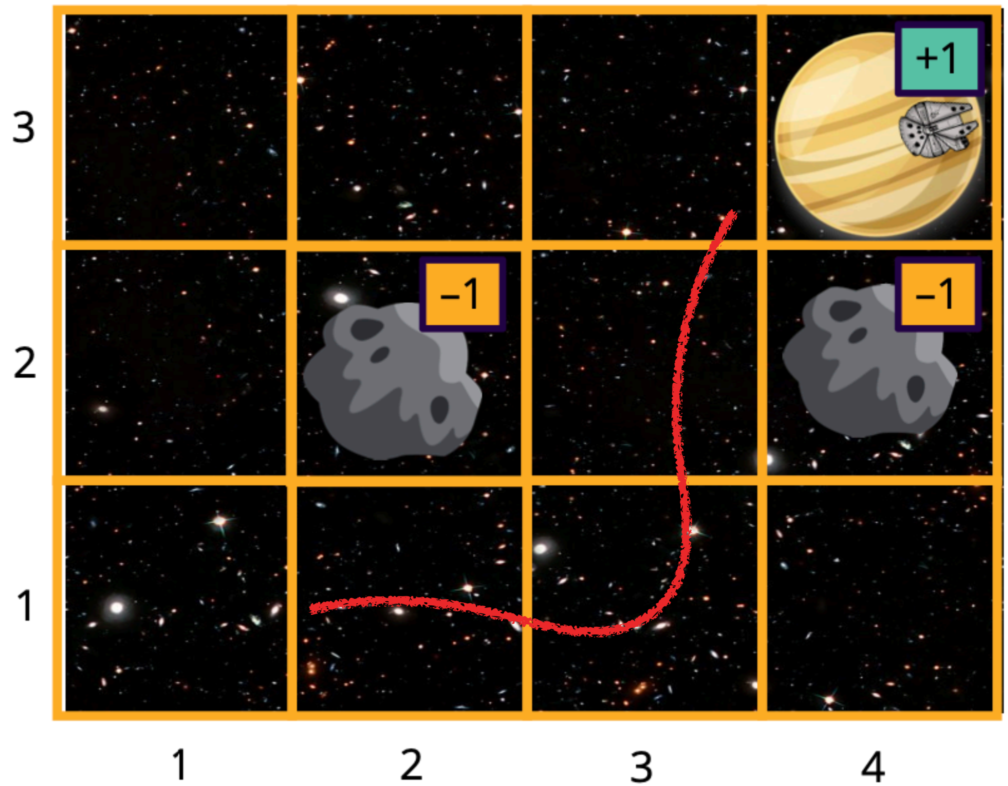
what's the probability that the millennium falcon reaches the intended goal?

0.1 * 0.1 * 0.1 * 0.1 * 0.8

= 0.00008

Transition Model:





Navigating an Asteroid Field

For action sequence

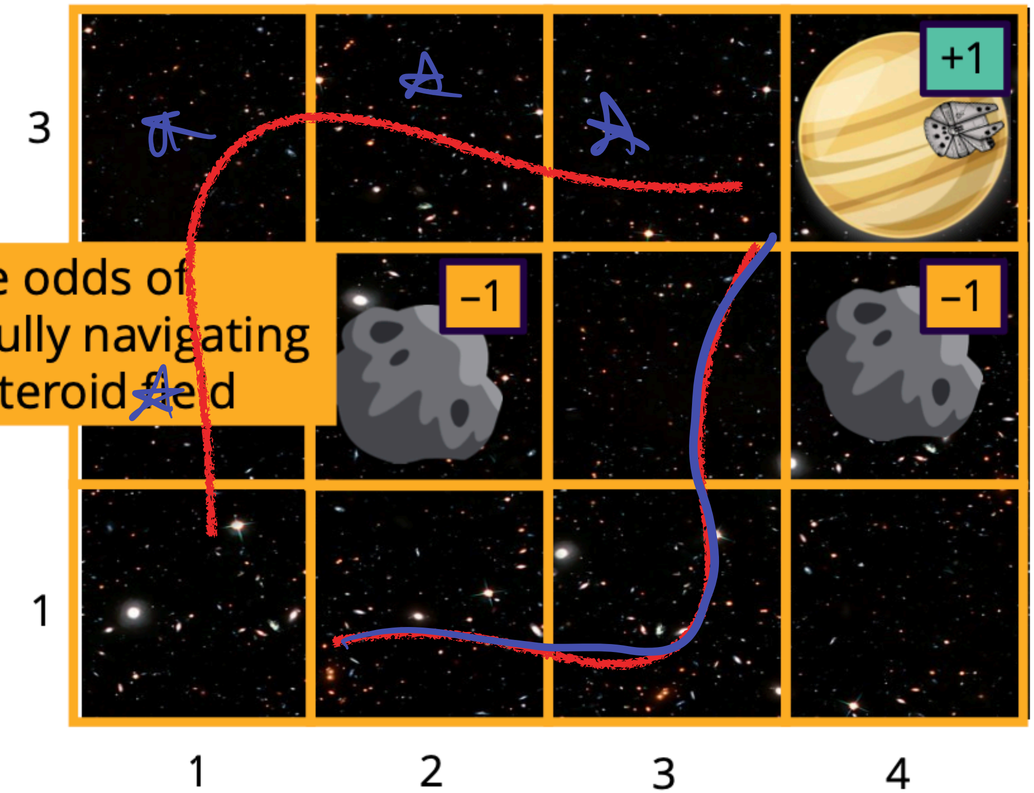
[Up, Up, Right, Right, Right],

what's the probability that the millennium falcon reaches the intended goal?

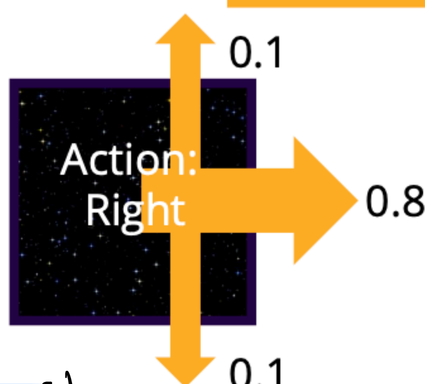
0.32768 + 0.00008

= 0.32776

The odds of successfully navigating an asteroid field



Transition Model:



[Right, Right, Up, Up, Right] = ??

= 0.32768

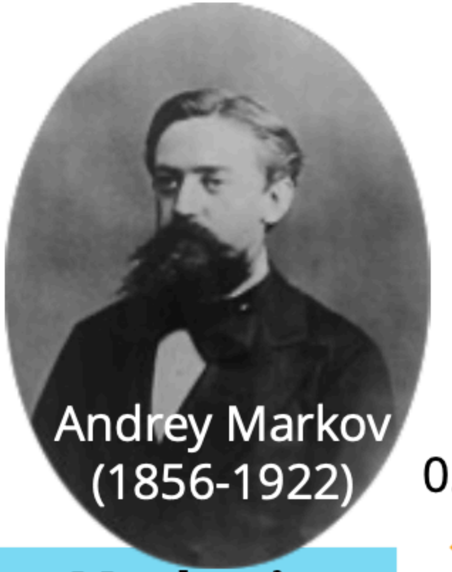
+

Markov Decision Processes

Stochastic Transition Model

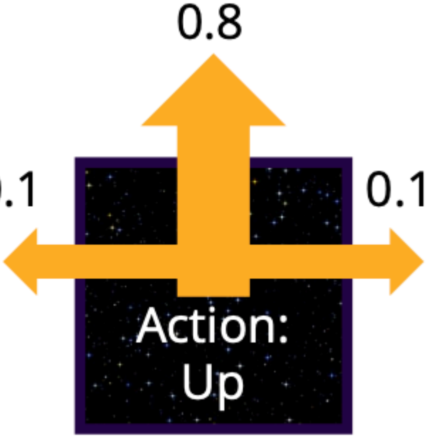
In our search algorithms so far, the transition model was deterministic and described the outcome of each action in each state.

The transition function is sometimes written as T(s, a, s'), or explicitly as a probability:



$$p(s' \mid s, a)$$

Transitions are **Markovian**: the probability of arriving in s' only depends on s and not the history of earlier states.



Markov Decision Process

A **Markov decision process** or **MDP** is

- a **sequential** decision problem
- for a **fully observable** environment
- with a **stochastic** transition model
- that has **additive rewards**

Reward function

Previously, we've rewarded our agent only when it found a solution.

In minimax, for instance, we calculated the utility of an action based on whether it lead to the goal.

For more open-ended problems, it is useful to have a reward function: our agent can be **rewarded at intermediate states**.

Reward function

We will specify a **utility or reward function** for the agent.

The "rewards" can be **positive** or **negative** but are bounded by some maximum value.

Because the decision process is **sequential,** we must specify the utility function on a sequence of states and actions.

Instead of only giving a reward at the goal states, the agent can **receive a reward at each time step**, based on its transition from **s** to **s'** via action **a**.

This is defined by a reward function

R(s, a, s')

Additive: $-0.04 + -0.04 + -0.04 + 0.04 + -0.04 + 1 =$

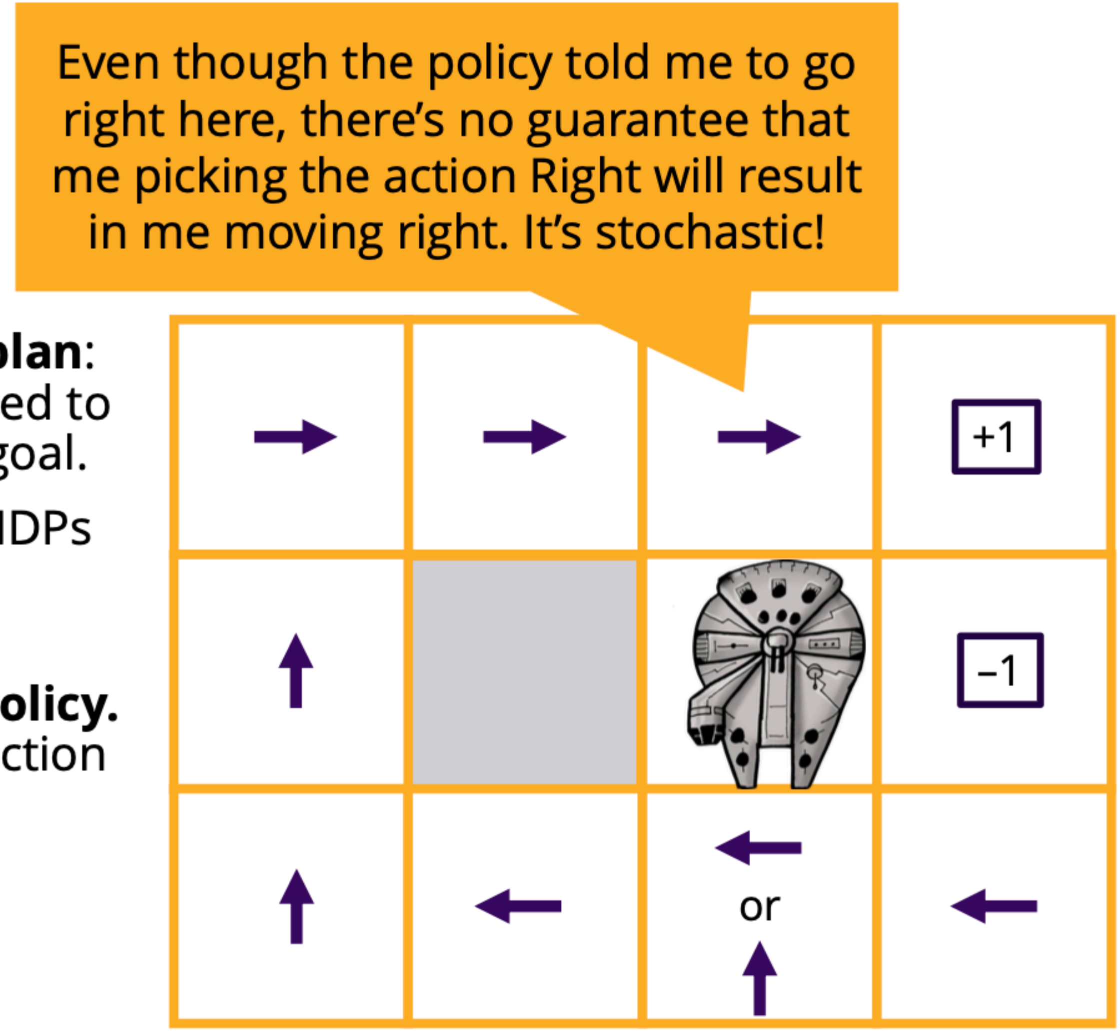
Solution == Policy

In search problems a solution was **a plan**: a sequence of action that corresponded to the shortest path from the start to a goal.

Because of the non-determinism in MDPs we cannot simply give a sequence of actions.

Instead, the solution to an MDP is a **policy.** A policy maps from a state onto the action to take if the agent is in that state.

$$\pi(s) = a$$



Slides adapted from Chris Callison-Burch

Markov Decision Process

To find a solution to an MDP, you need to define the following things:

- **A set of states** $s \in S$
- **A set of actions** $a \in A$
- A transition function **T(s, a, s')**
  - Probability that executing action **a** in **s** will lead to **s' P(s' | s, a)**
  - The probability is called **the model**
- A reward function **R(s, a, s')**
  - Sometimes just R(s) or R(s')
- An **initial state** s_0
- Optionally, one or more **terminal states**

Solution == Policy

In search problems a solution was **a plan**: a sequence of action that corresponded to the shortest path from the start to a goal.

Because of the non-determinism in MDPs we cannot simply give a sequence of actions.

Instead, the solution to an MDP is a **policy.** A policy maps from a state onto the action to take if the agent is in that state.

$$\boldsymbol{\pi}(s) = a$$

Policies and Rewards

Even if the **same policy** is executed multiple times by the agent, this may lead to different sequence of states and actions (**environment history**), and thus a **different score** under the reward function.

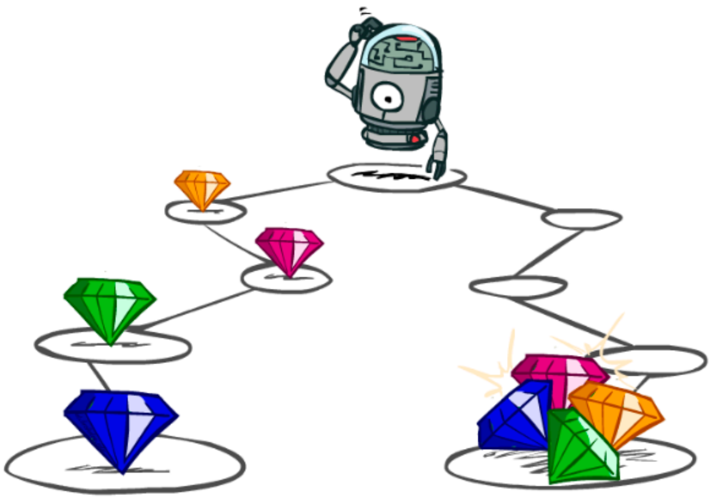
Therefore we need to compute the **expected utility** of all the possible paths generated by a policy.

Sequence Utility

Utilities of Sequences

What preferences should an agent have over reward sequences?

More or less?	[1, 2, 2]	or	[2, 3, 4]
Now or later?	[0, 0, 1]	or	[1, 0, 0]



Slides adapted from Chris Callison-Burch

Discounting

It's reasonable to maximize the sum of rewards

It's also reasonable to prefer rewards now to rewards later

One solution: values of rewards decay exponentially

1

Worth Now

γ

Worth Next Step

γ^2

Worth In Two Steps

Discounting

How to discount?

- Each time we descend a level, we multiply in the discount once

Why discount?

- Sooner rewards probably do have higher utility than later rewards
- Also helps our algorithms converge