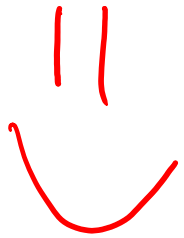
Announcements



Canvas

- Up-to-date with scores and slip days

Assignments

- HW7: Thu, 11/19, 11:59 pm

Schedule change

- Friday: Lecture in all three recitation slots
- Monday: Recitation in both lecture slots

Final exam scheduled

Study groups

Wrap Up HMMs

HMM slides from last time

Introduction to Machine Learning

Markov Decision Processes

Instructor: Pat Virtue

Learning Paradigms

Paradigm	**Data**
Supervised	$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^N$ $\quad \mathbf{x} \sim p^*(\cdot)$ and $y = c^*(\cdot)$
$\hookrightarrow$ Regression	$y^{(i)} \in \mathbb{R}$
$\hookrightarrow$ Classification	$y^{(i)} \in \{1, \ldots, K\}$
$\hookrightarrow$ Binary classification	$y^{(i)} \in \{+1, -1\}$
$\hookrightarrow$ Structured Prediction	$\mathbf{y}^{(i)}$ is a vector
Unsupervised	$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N$ $\quad \mathbf{x} \sim p^*(\cdot)$
Semi-supervised	$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{N_1} \cup \{\mathbf{x}^{(j)}\}_{j=1}^{N_2}$
Online	$\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), (\mathbf{x}^{(3)}, y^{(3)}), \ldots\}$
Active Learning	$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N$ and can query $y^{(i)} = c^*(\cdot)$ at a cost
Imitation Learning	$\mathcal{D} = \{(s^{(1)}, a^{(1)}), (s^{(2)}, a^{(2)}), \ldots\}$
Reinforcement Learning	$\mathcal{D} = \{(s^{(1)}, a^{(1)}, r^{(1)}), (s^{(2)}, a^{(2)}, r^{(2)}), \ldots\}$

Slide credit: CMU MLD, Matt Gormley

Learning Paradigms

Paradigm	**Data**
Supervised	$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^N$ $\quad \mathbf{x} \sim p^*(\cdot)$ and $y = c^*(\cdot)$
$\hookrightarrow$ Regression	$y^{(i)} \in \mathbb{R}$
$\hookrightarrow$ Classification	$y^{(i)} \in \{1, \ldots, K\}$
$\hookrightarrow$ Binary classification	$y^{(i)} \in \{+1, -1\}$
$\hookrightarrow$ Structured Prediction	$\mathbf{y}^{(i)}$ is a vector
Unsupervised	$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N$ $\quad \mathbf{x} \sim p^*(\cdot)$
Semi-supervised	$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{N_1} \cup \{\mathbf{x}^{(j)}\}_{j=1}^{N_2}$
Online	$\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), (\mathbf{x}^{(3)}, y^{(3)}), \ldots\}$
Active Learning	$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N$ and can query $y^{(i)} = c^*(\cdot)$ at a cost
Imitation Learning	$\mathcal{D} = \{(s^{(1)}, a^{(1)}), (s^{(2)}, a^{(2)}), \ldots\}$
Reinforcement Learning	$\mathcal{D} = \{(s^{(1)}, a^{(1)}, r^{(1)}), (s^{(2)}, a^{(2)}, r^{(2)}), \ldots\}$

Slide credit: CMU MLD, Matt Gormley

Interacting with the World

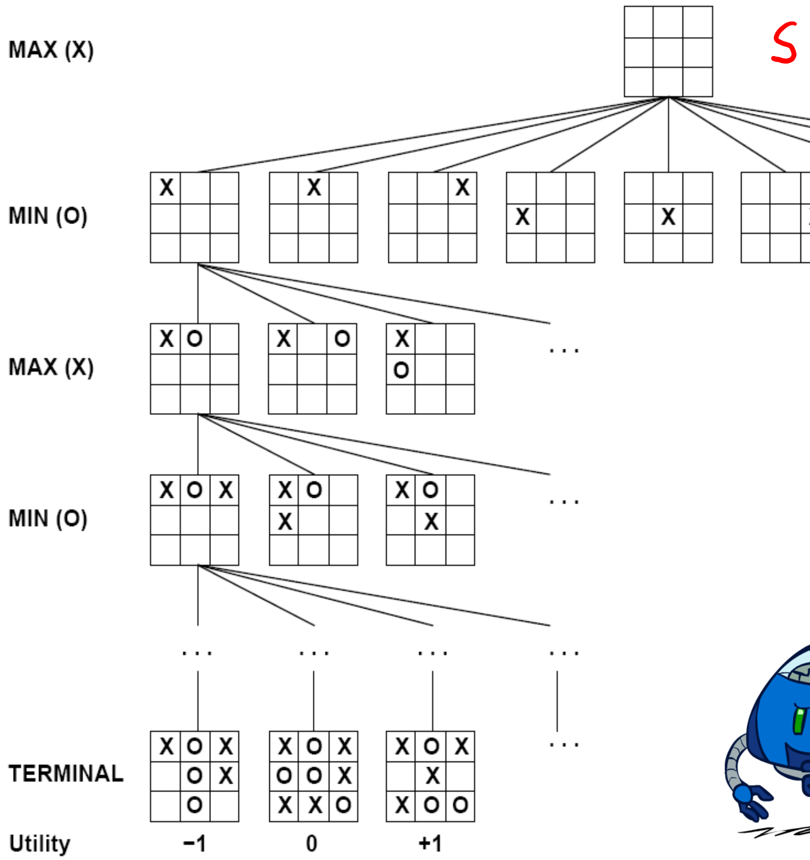
Sequential decision making

Games

- Simple: Tic-tac-toe
- Go
- Arcade games

Worlds

- Simple: Grid World
- Open AI Gym https://gym.openai.com/
- Autonomous Driving



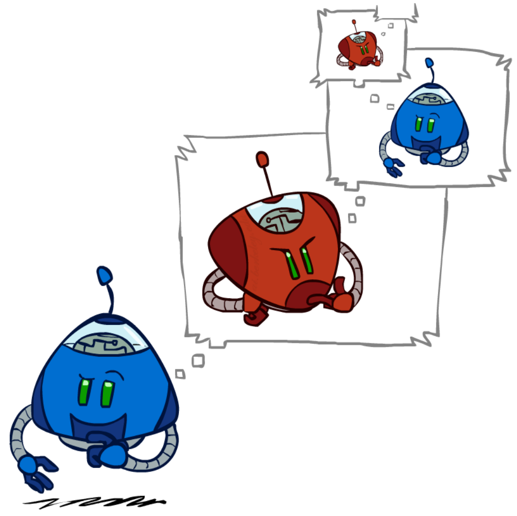


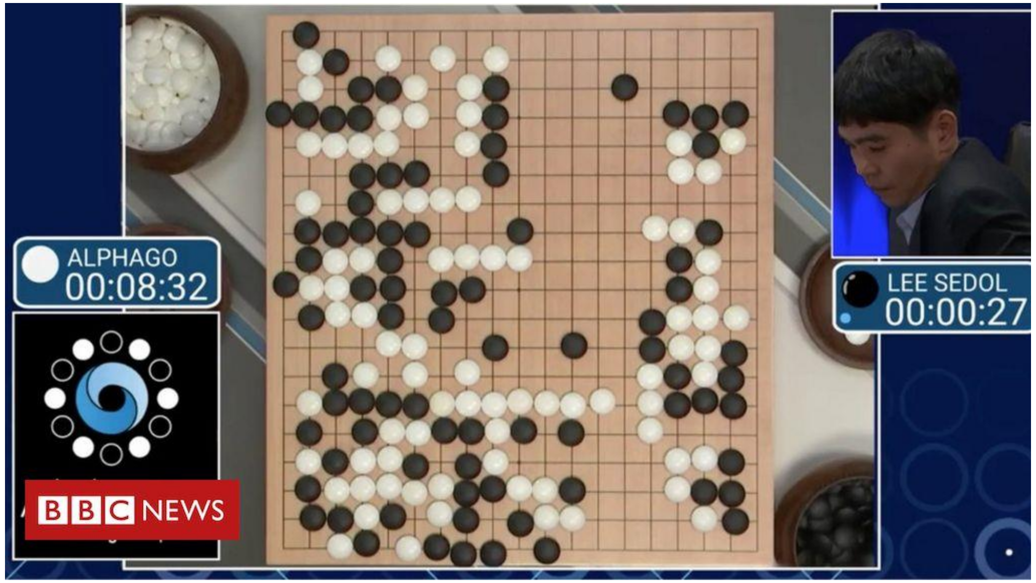
Image: ai.berkeley.edu

Interacting with the World

Sequential decision making

Games

- Simple: Tic-tac-toe
- Go
- Arcade games



Worlds

- Simple: Grid World
- Open AI Gym https://gym.openai.com/
- Autonomous Driving

Image: BBC News

Interacting with the World

Sequential decision making

Games

- Simple: Tic-tac-toe
- Go
- Arcade games

Worlds

- Simple: Grid World
- Open AI Gym https://gym.openai.com/
- Autonomous Driving

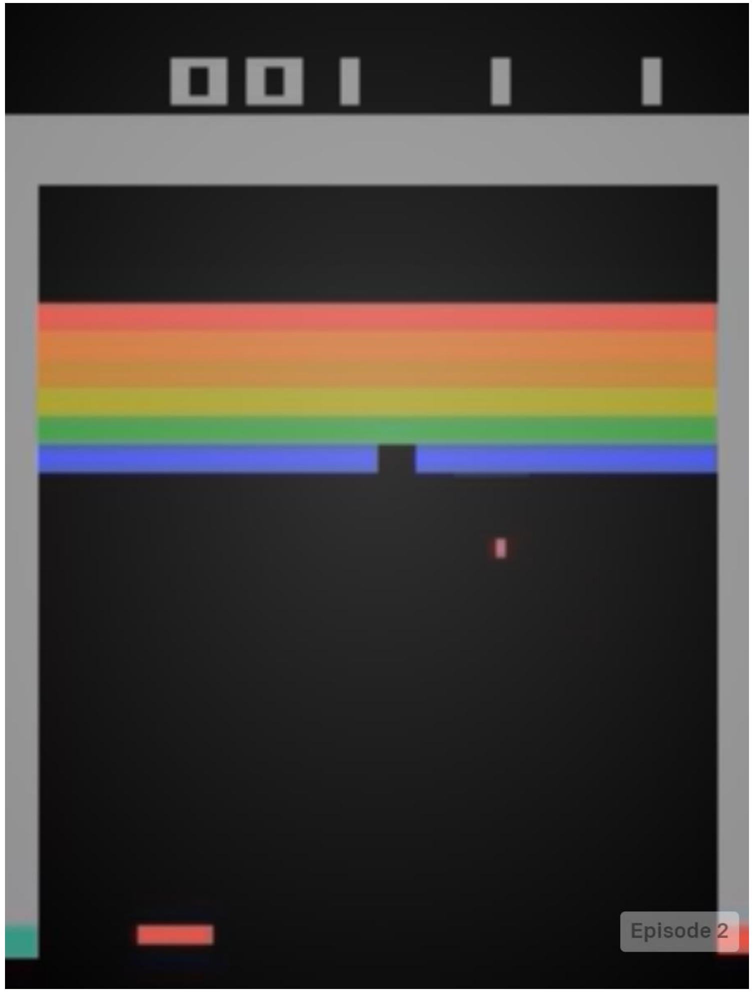


Image: https://gym.openai.com/

Interacting with the World

Sequential decision making

Games

- Simple: Tic-tac-toe
- Go
- Arcade games

Worlds

- Simple: Grid World
- Open AI Gym https://gym.openai.com/
- Autonomous Driving

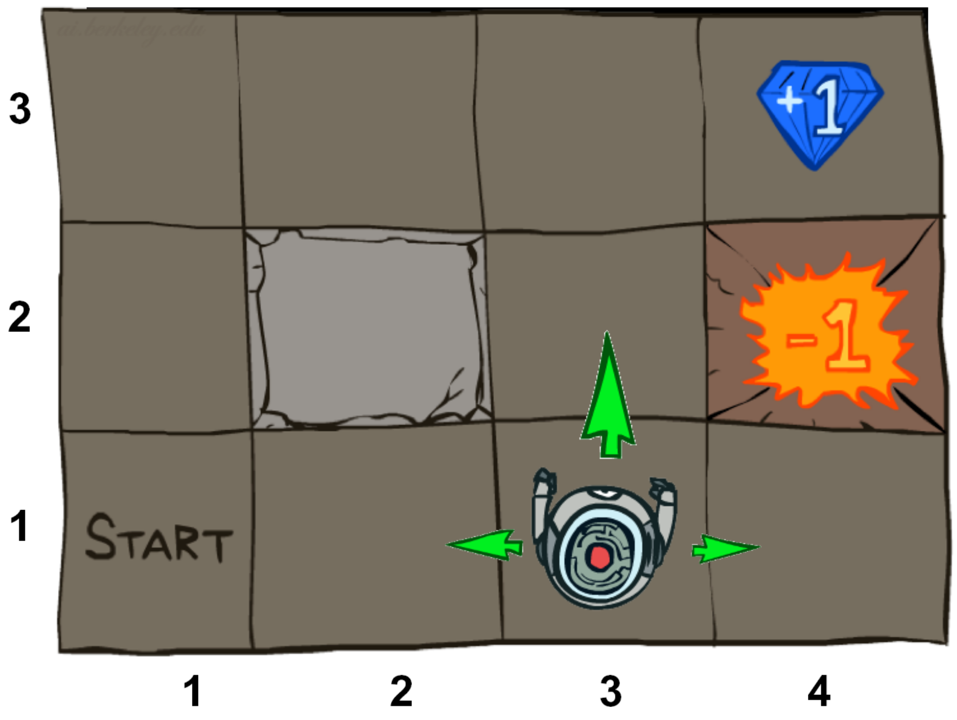


Image: ai.berkeley.edu

Interacting with the World

Sequential decision making

Games

- Simple: Tic-tac-toe
- Go
- Arcade games

Worlds

- Simple: Grid World
- Open AI Gym https://gym.openai.com/
- Autonomous Driving

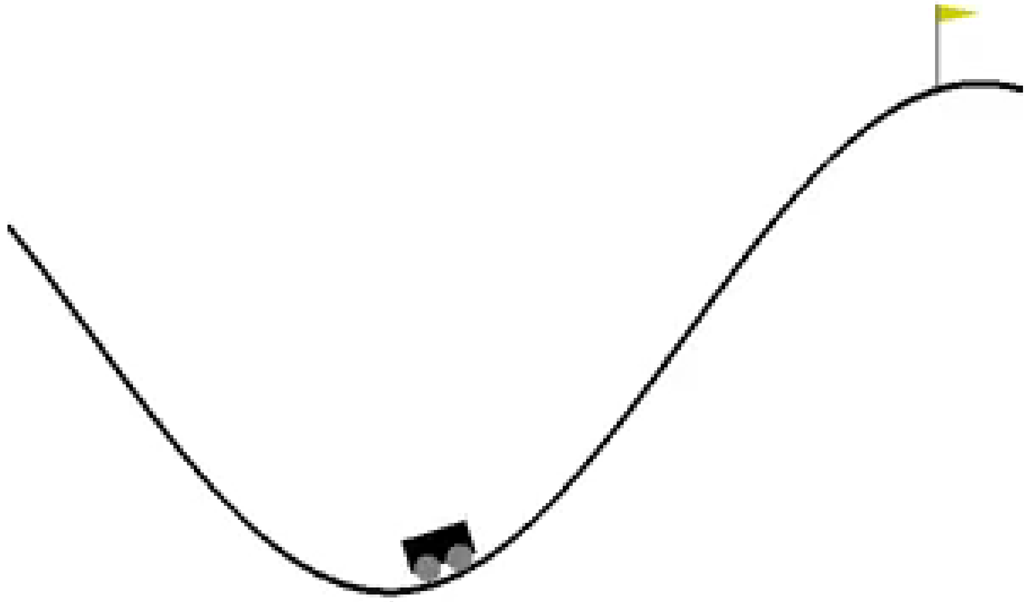


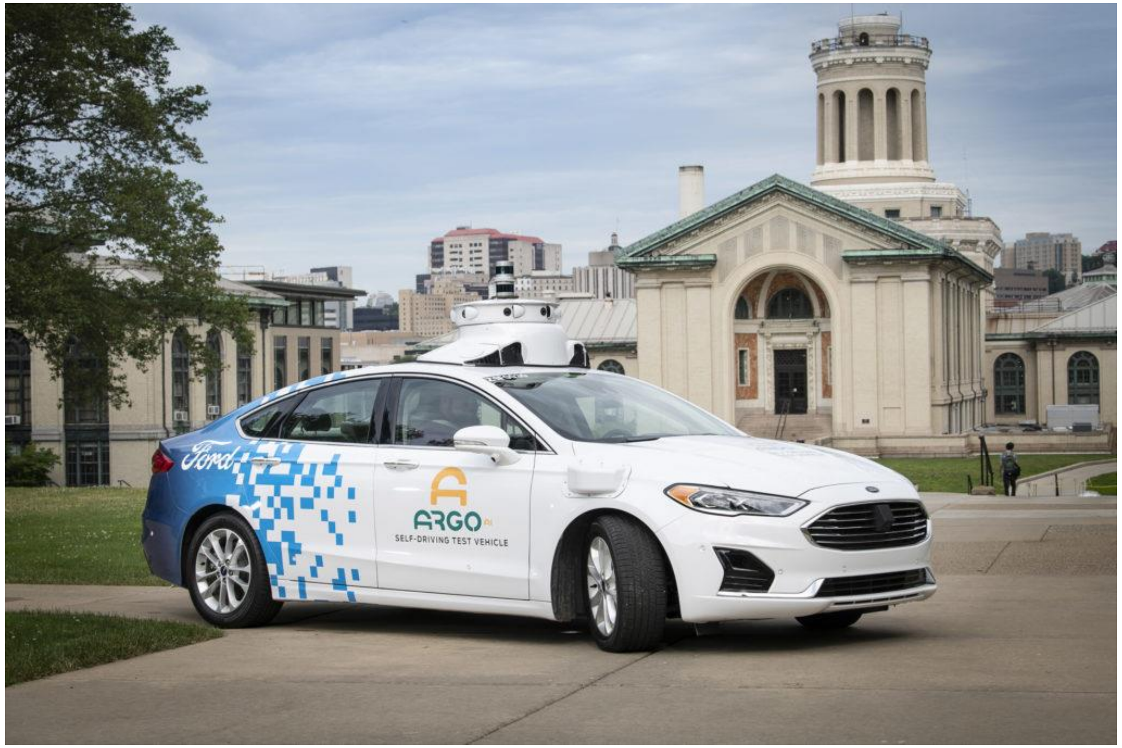
Image: https://gym.openai.com/

Interacting with the World

Sequential decision making

Games

- Simple: Tic-tac-toe
- Go
- Arcade games



Worlds

- Simple: Grid World
- Open AI Gym https://gym.openai.com/
- Autonomous Driving

Image: https://www.argo.ai/2019/06/pushing-the-self-driving-frontier-argo-ai-partners-with-carnegie-mellon-to-form-autonomous-vehicle-research-center/

Games Trees

Minimax Search

$\arg\max_a V(s')$

$s' = result(s, a)$

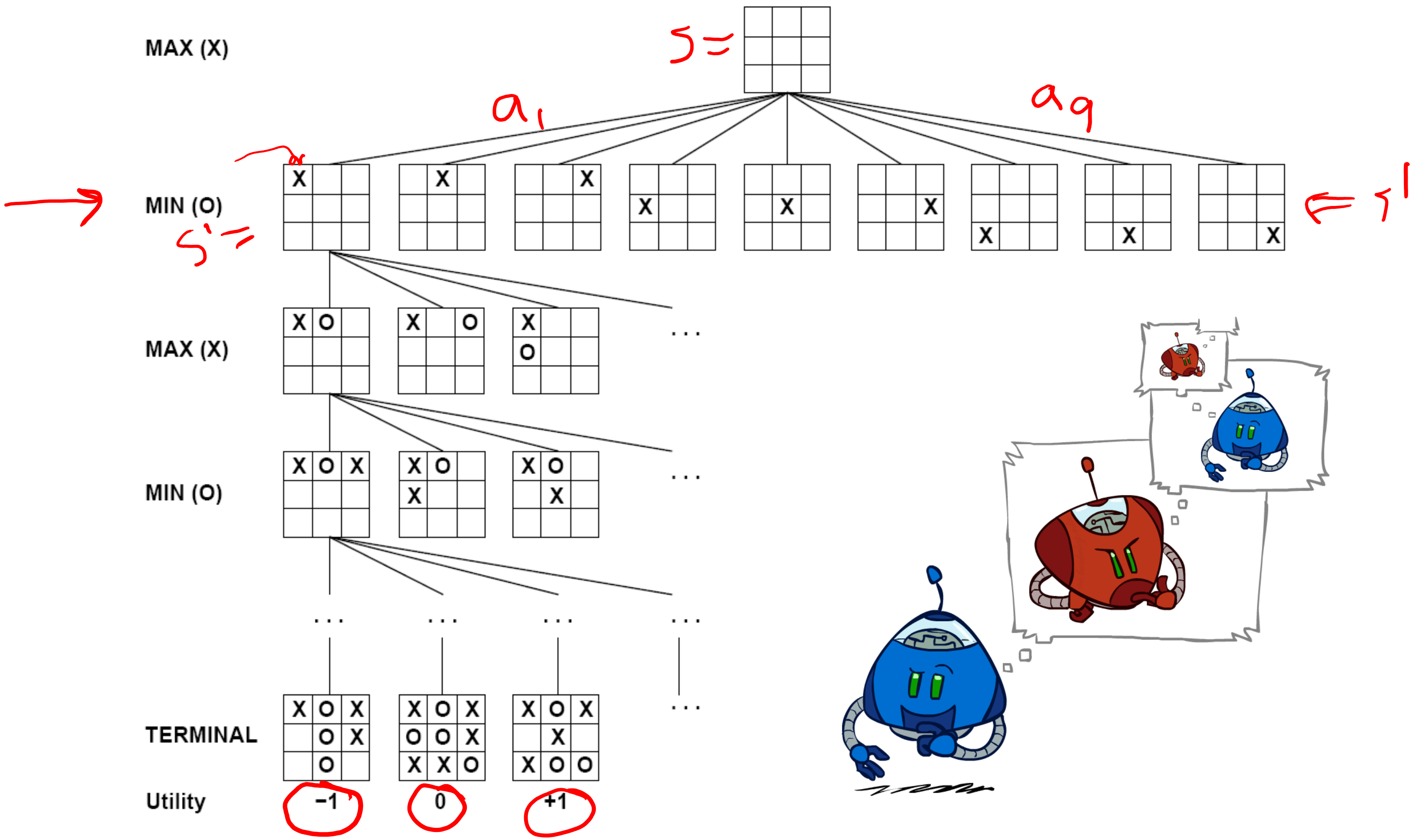
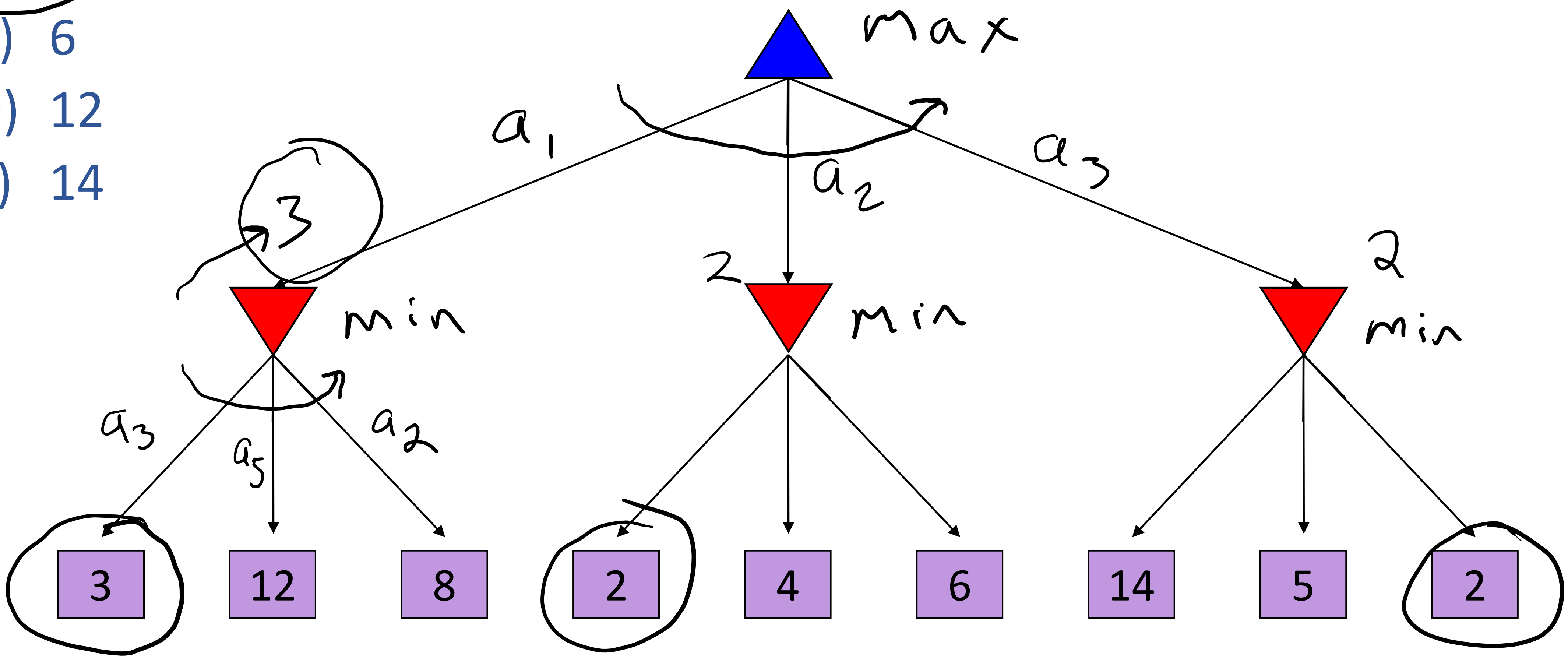


Image: ai.berkeley.edu

Piazza Poll 1

What is the minimax value at the root?

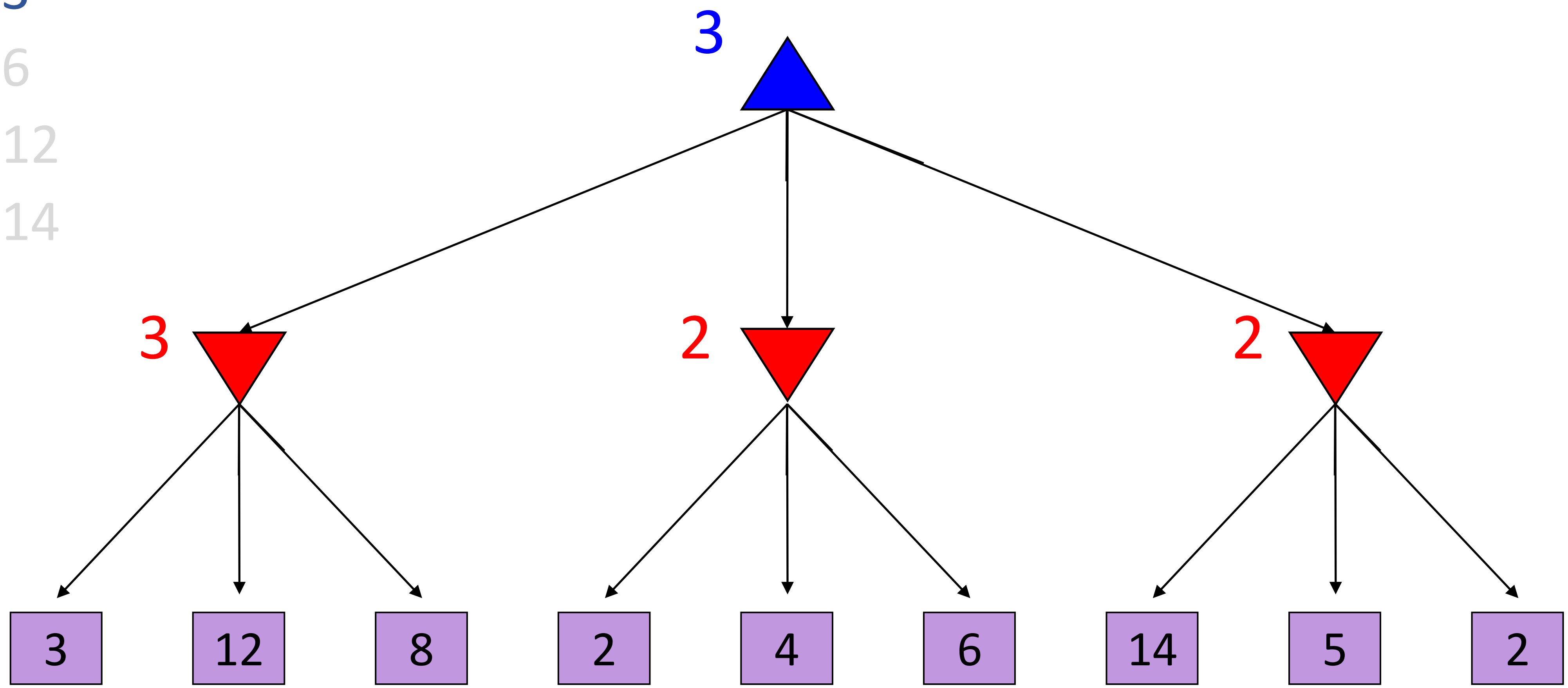
A) 2
B) 3
C) 6
D) 12
E) 14



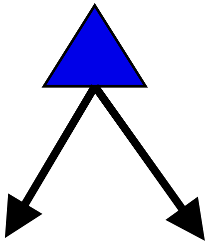
Piazza Poll 1

What is the minimax value at the root?

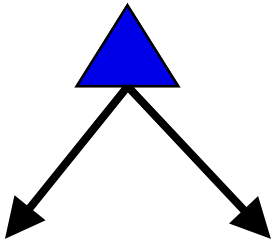
A) 2
B) 3
C) 6
D) 12
E) 14



Minimax Notation



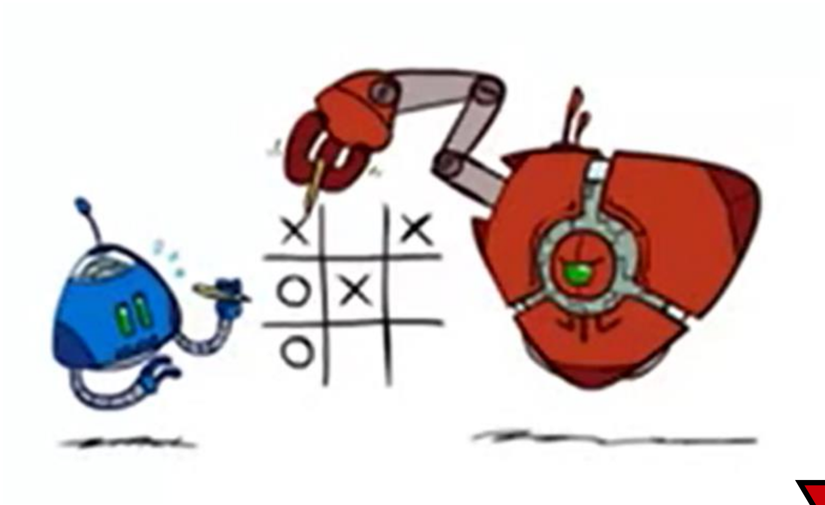
$$V(s) = \max_{a} V(s'),$$
where $s' = result(s, a)$



$$\hat{a} = \underset{a}{\operatorname{argmax}}\, V(s'),$$
where $s' = result(s, a)$

Modeling Assumptions

Know your opponent



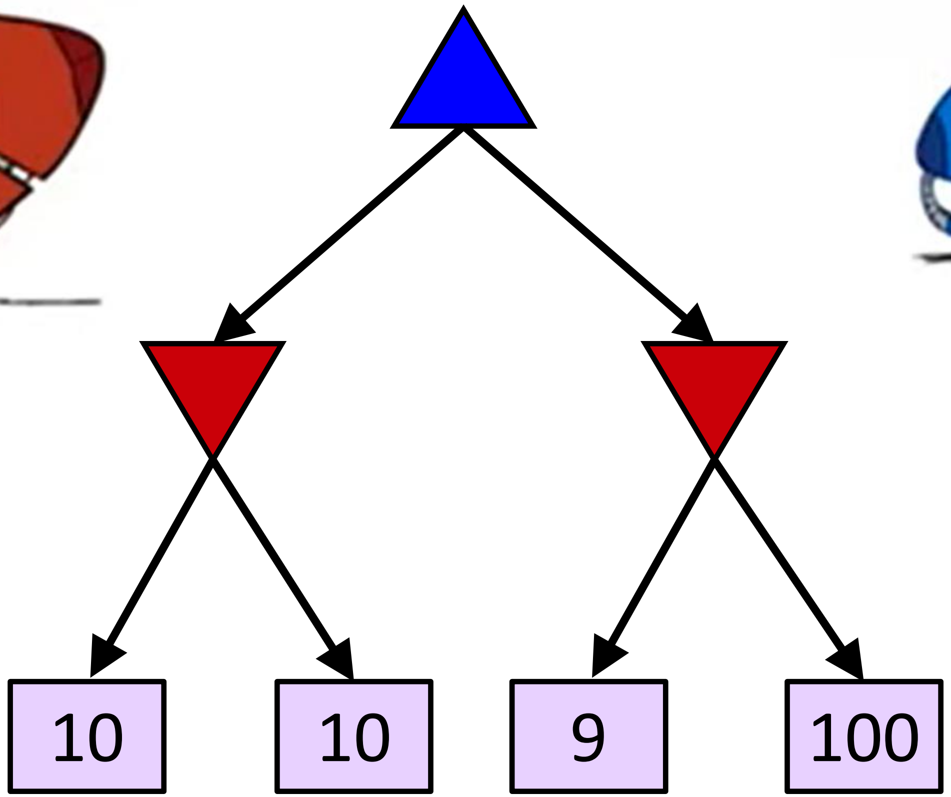


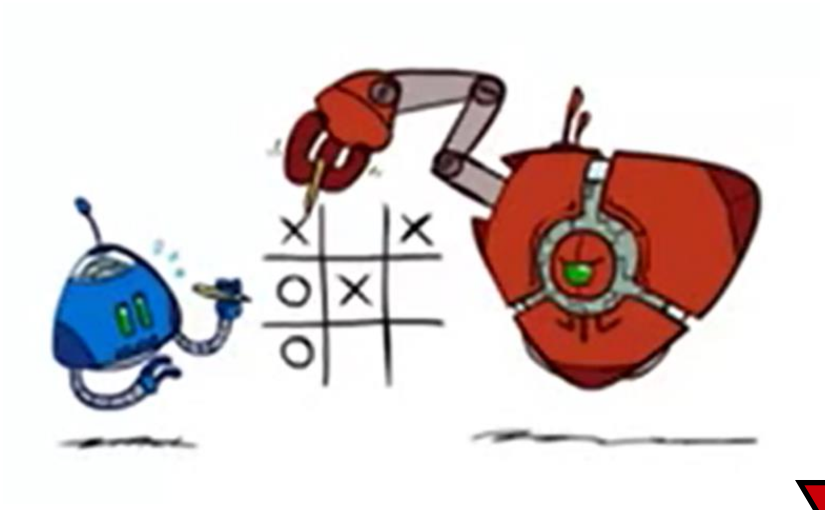
Image: ai.berkeley.edu

Minimax Driver?

Clip: How I Met Your Mother, CBS

Modeling Assumptions

Know your opponent



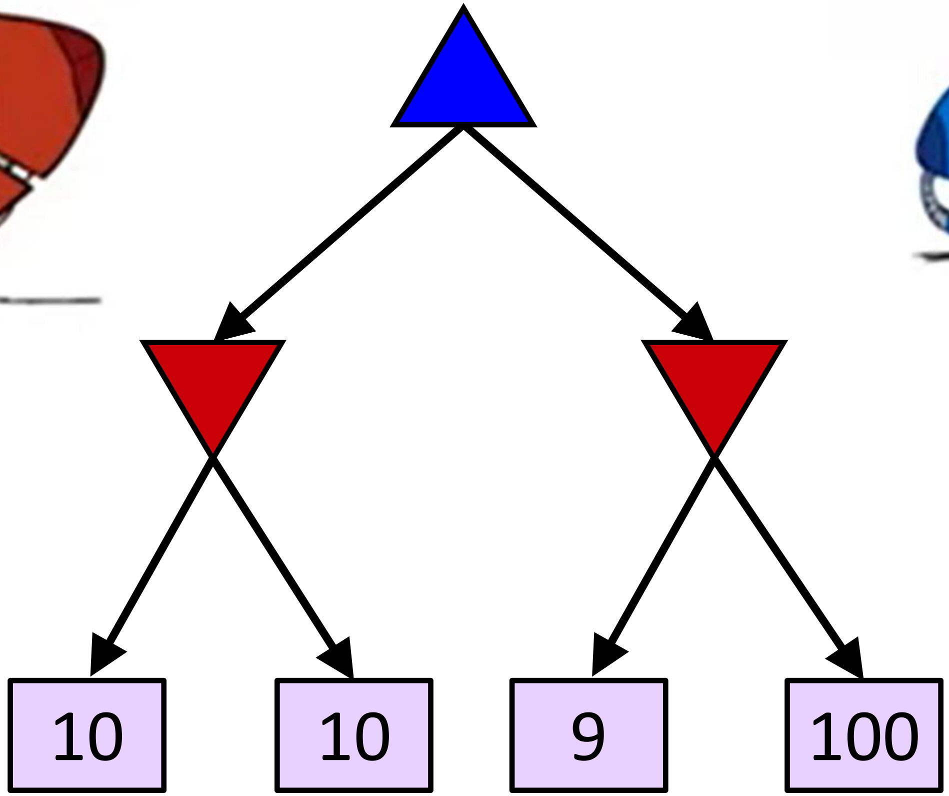
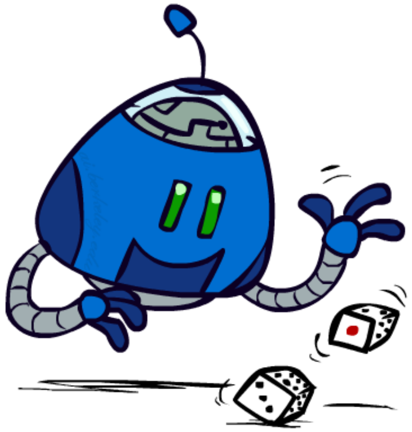


Image: ai.berkeley.edu

Modeling Assumptions

Chance nodes: Expectimax



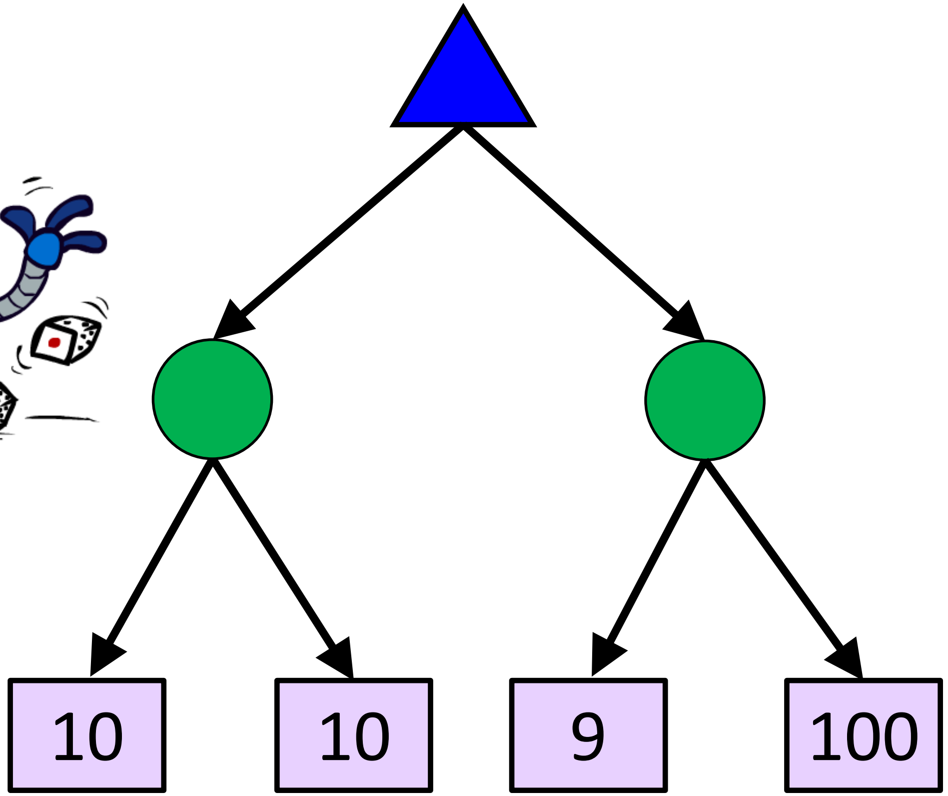
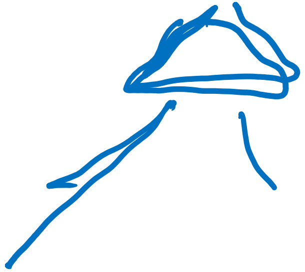
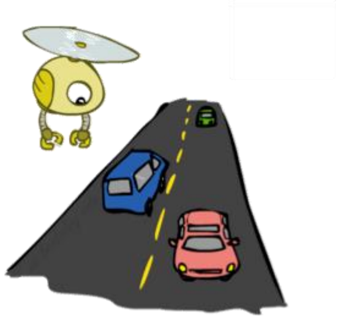


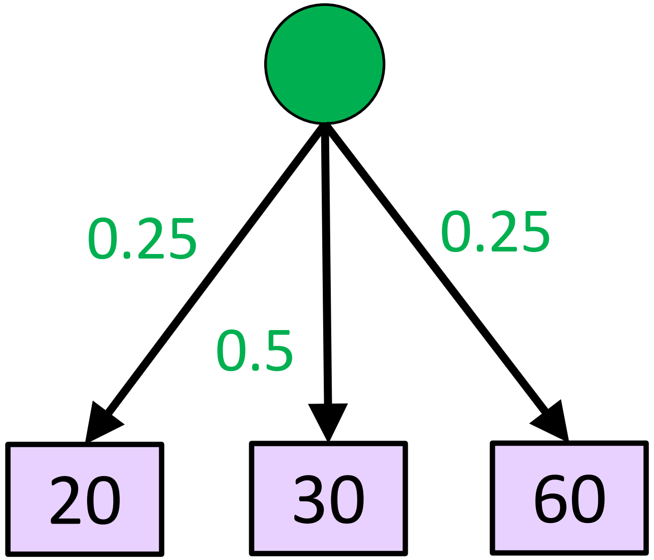
Image: ai.berkeley.edu

Expectations



Time:	20 min		30 min		60 min
	x	+	x	+	x
Probability:	0.25		0.50		0.25





Max node notation

$$V(s) = \max_{a} V(s'),$$

where $s' = result(s, a)$

Chance node notation

$$V(s) = \sum_{s'} P(s')\, V(s')$$

Image: ai.berkeley.edu

Piazza Poll 2

Expectimax tree search:

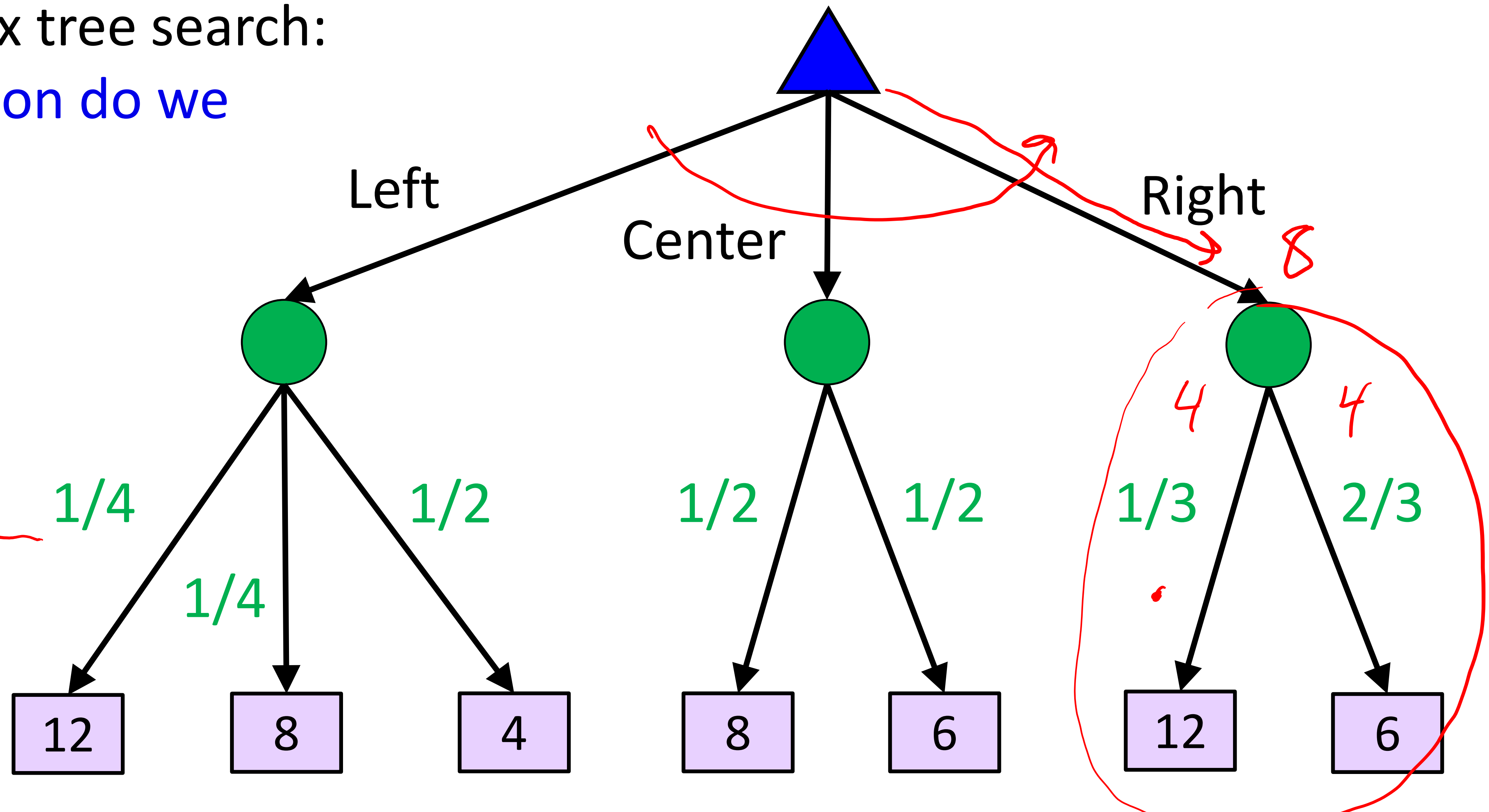
Which action do we choose?

A: Left

B: Center

C: Right

D: Eight



Piazza Poll 2

Expectimax tree search:

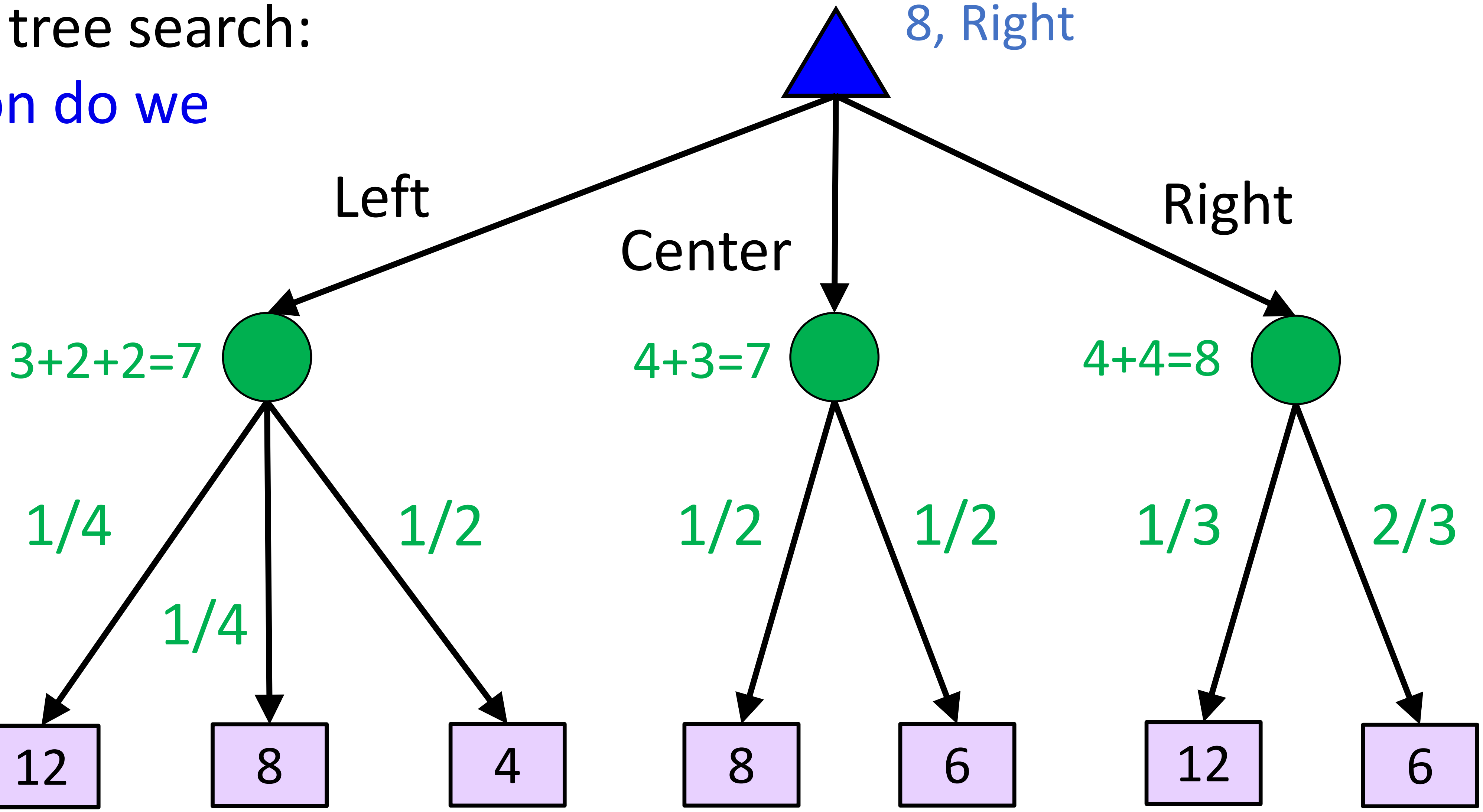
Which action do we choose?

A: Left

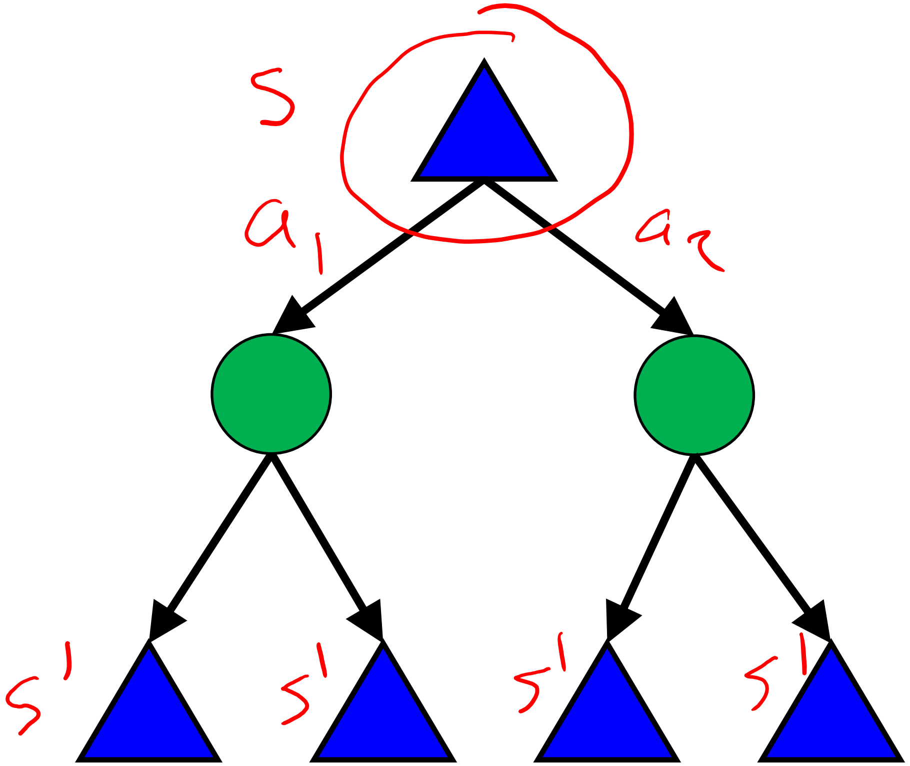
B: Center

C: Right

D: Eight



Expectimax Notation



$$V(s) = \max_{a} \sum_{s'} P(s'|s,a)\, V(s')$$

Non-Deterministic Search

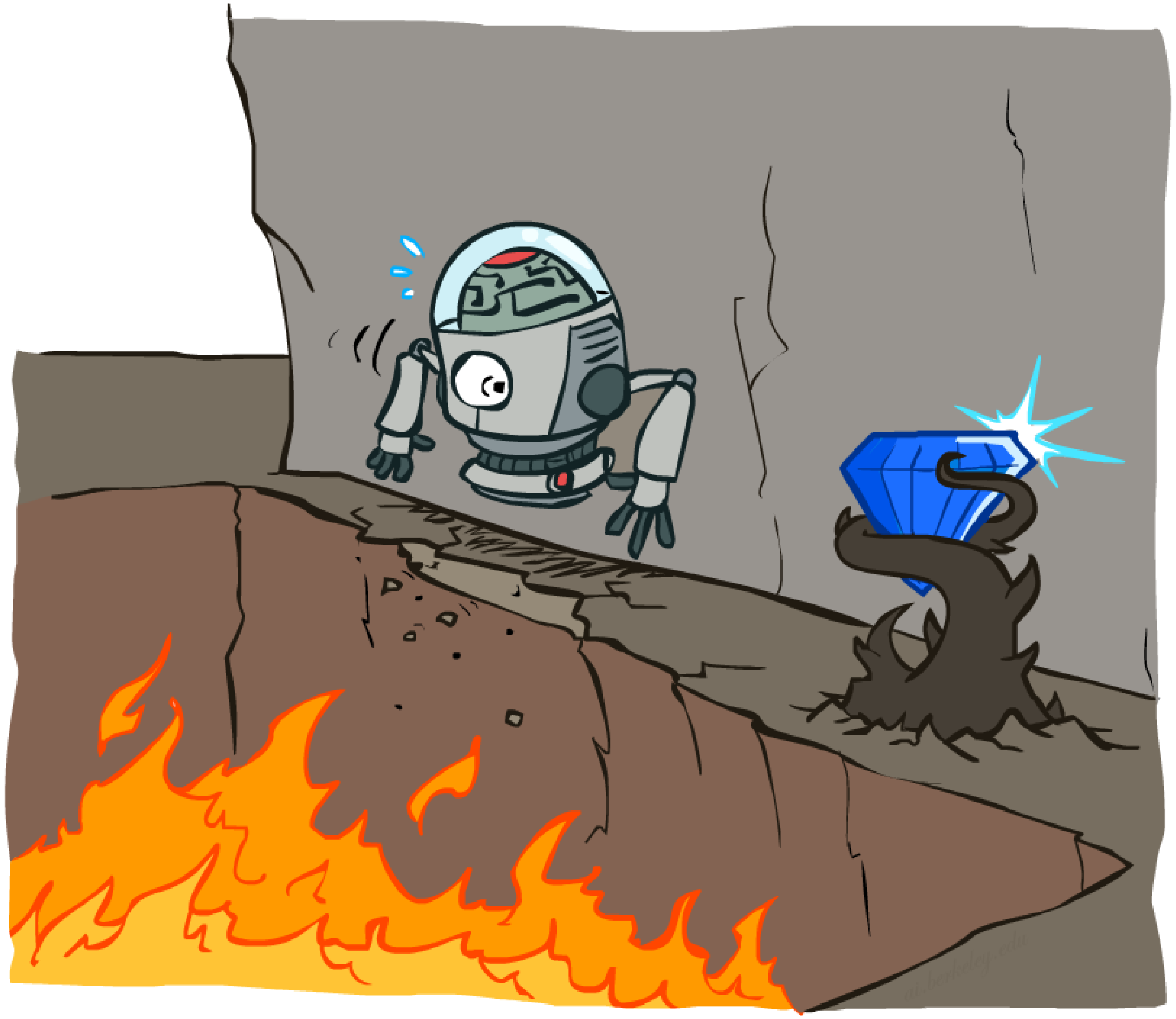
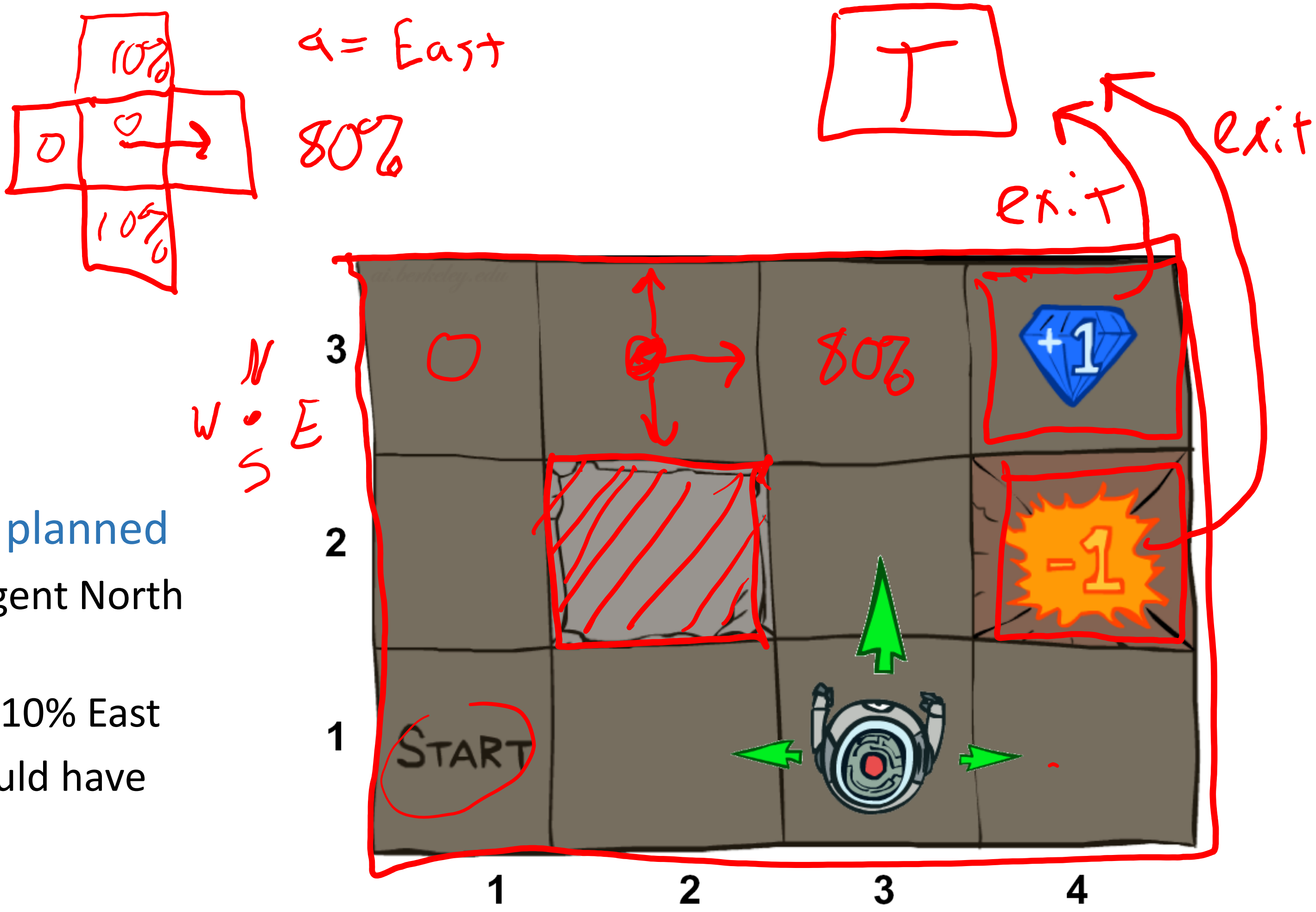


Image: ai.berkeley.edu

Example: Grid World



- A maze-like problem
  - The agent lives in a grid
  - Walls block the agent's path
- Noisy movement: actions do not always go as planned
  - 80% of the time, the action North takes the agent North (if there is no wall there)
  - 10% of the time, North takes the agent West; 10% East
  - If there is a wall in the direction the agent would have been taken, the agent stays put
- The agent receives rewards each time step
  - Small "living" reward each step (can be negative) -0.1
  - Big rewards come at the end (good or bad)
- Goal: maximize sum of rewards

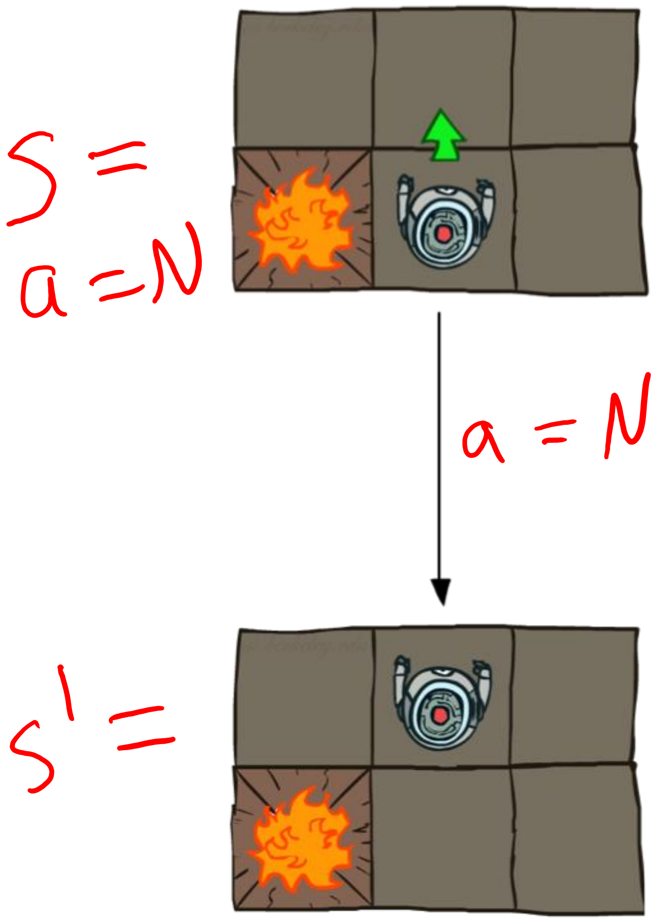
80% go the direction that I choose

20% go in an adjacent direction

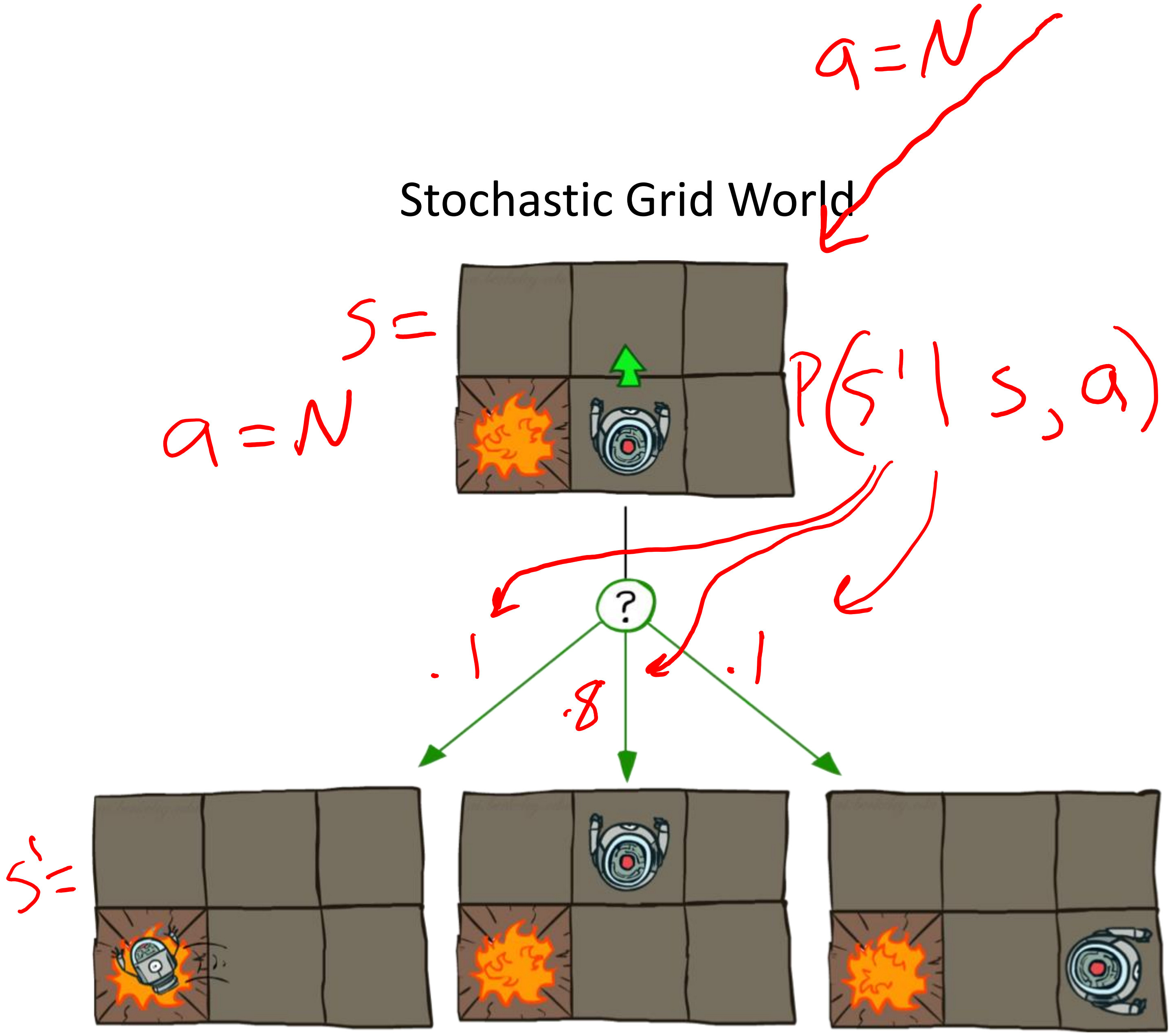
Slide: ai.berkeley.edu

Grid World Actions

Deterministic Grid World



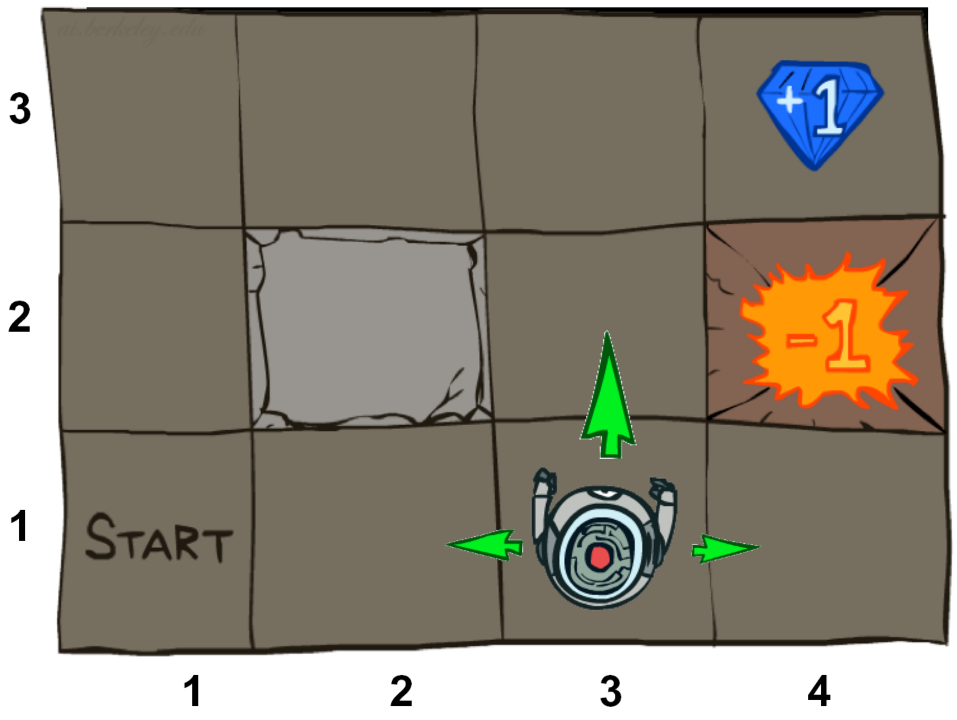
Stochastic Grid World



Markov Decision Processes

An MDP is defined by:

- A set of states $s \in S$
- A set of actions $a \in A$
- A transition function $T(s, a, s')$
  - Probability that a from s leads to s', i.e., $P(s' | s, a)$
  - Also called the model or the dynamics
- A reward function $R(s, a, s')$
  - Sometimes just $R(s)$ or $R(s')$
- A start state
- Maybe a terminal state



MDPs are non-deterministic search problems

- One way to solve them is with expectimax search
- We'll have a new tool soon

[Demo – gridworld manual intro (L8D1)]

Demo of Gridworld

What is Markov about MDPs?

"Markov" generally means that given the present state, the future and the past are independent

For Markov decision processes, "Markov" means action outcomes depend only on the current state

$$P(S_{t+1} = s' | S_t = s_t, A_t = a_t, S_{t-1} = s_{t-1}, A_{t-1}, \ldots S_0 = s_0)$$

$$=$$

$$P(S_{t+1} = s' | S_t = s_t, A_t = a_t)$$

Andrey Markov
(1856-1922)

Slide: ai.berkeley.edu

Policies

$\pi(s) \to a$

$h(x) \to y$

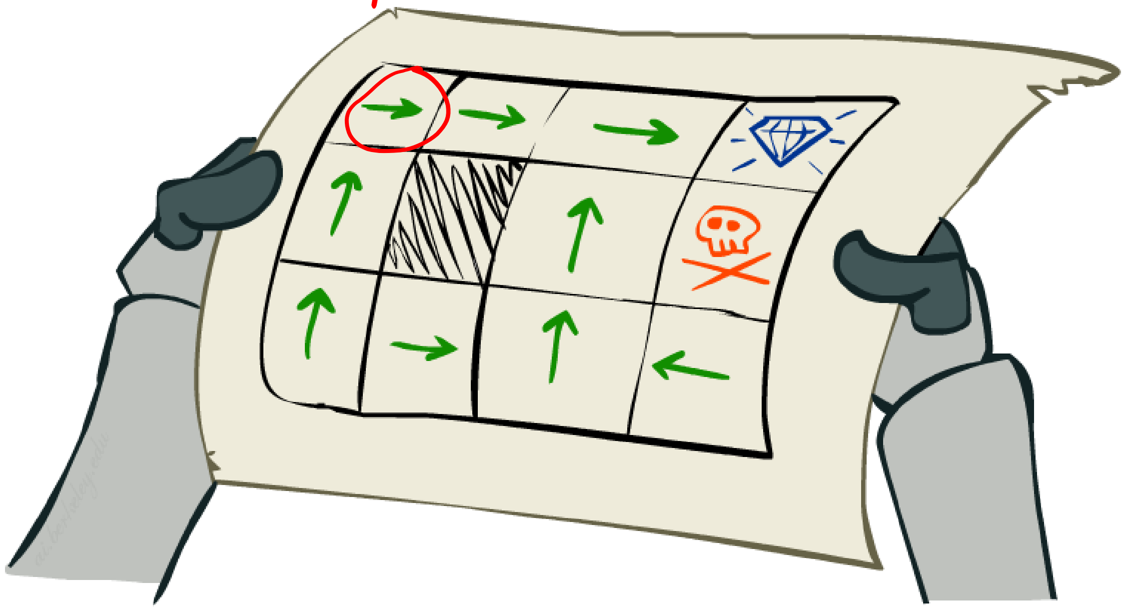
We don't just want an optimal plan, or sequence of actions, from start to a goal

For MDPs, we want an optimal policy $\pi^*: S \rightarrow A$

- A policy π gives an action for each state
- An optimal policy is one that maximizes expected utility if followed

Expectimax didn't compute entire policies

- It computed the action for a single state only



Optimal policy when R(s, a, s') = -0.03 for all non-terminals s

Slide: ai.berkeley.edu