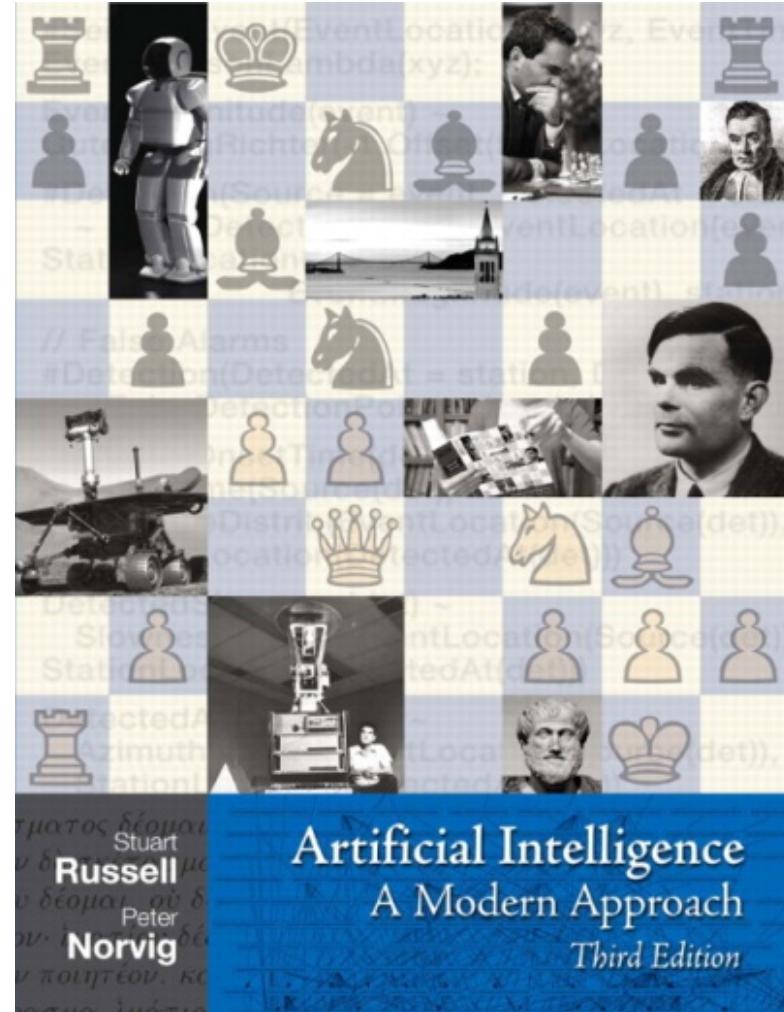


Expectimax, Utilities and MDPs

Read AIMA Chapter 16.1-16.3,
and AIMA Chapter 17.1-17.3,
and Sutton and Barto, Chapter 3

Slides courtesy of Dan Klein and Pieter Abbeel

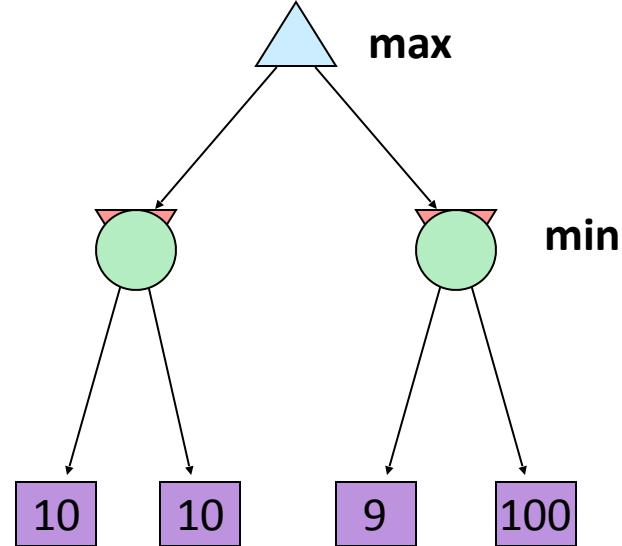
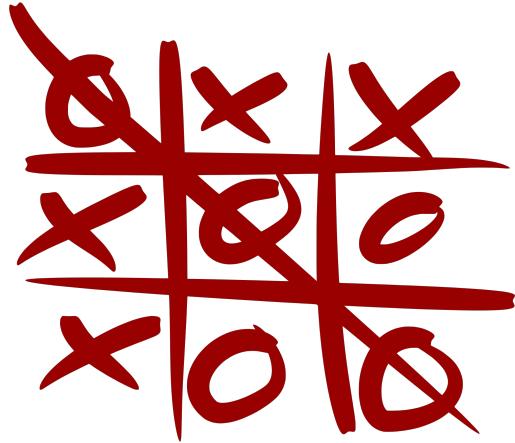
University of California, Berkeley



Uncertain Outcomes



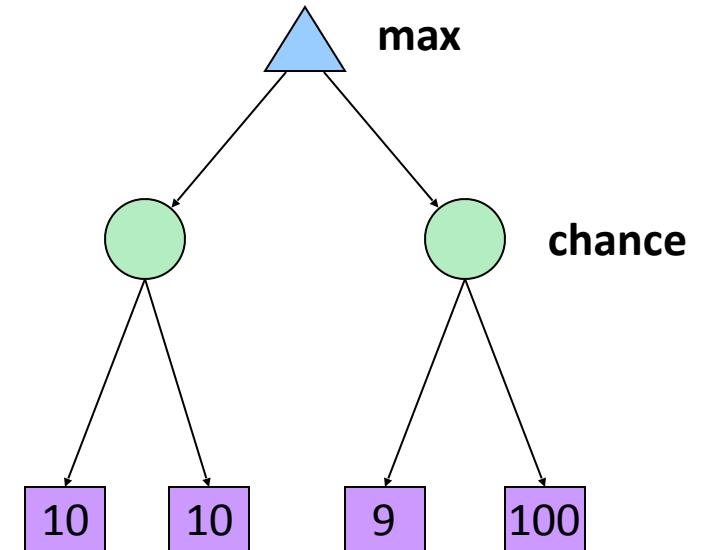
Worst-Case vs. Average Case



Idea: Uncertain outcomes controlled by chance, not an adversary!

Expectimax Search

- Why wouldn't we know what the result of an action will be?
 - Explicit randomness: rolling dice
 - Unpredictable opponents: the opponent isn't optimal
 - Actions can fail: when moving a robot, wheels might slip
- Values should now reflect average-case (expectimax) outcomes, not worst-case (minimax) outcomes
- 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
- Later, we'll learn how to formalize the underlying uncertain-result problems as **Markov Decision Processes**

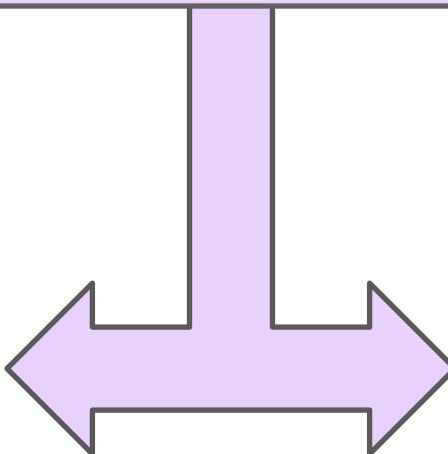


Expectimax Pseudocode

```
def value(state):
    if the state is a terminal state: return the state's utility
    if the next agent is MAX: return max-value(state)
    if the next agent is EXP: return exp-value(state)
```

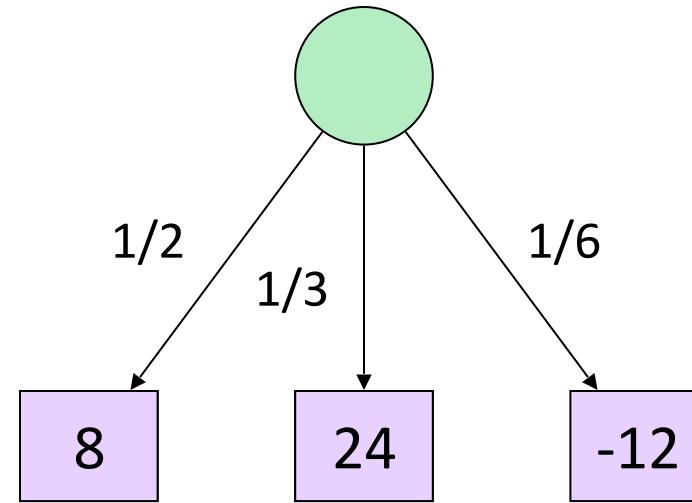
```
def max-value(state):
    initialize v = -∞
    for each successor of state:
        v = max(v, value(successor))
    return v
```

```
def exp-value(state):
    initialize v = 0
    for each successor of state:
        p =
            probability(successor)
        v += p * value(successor)
    return v
```



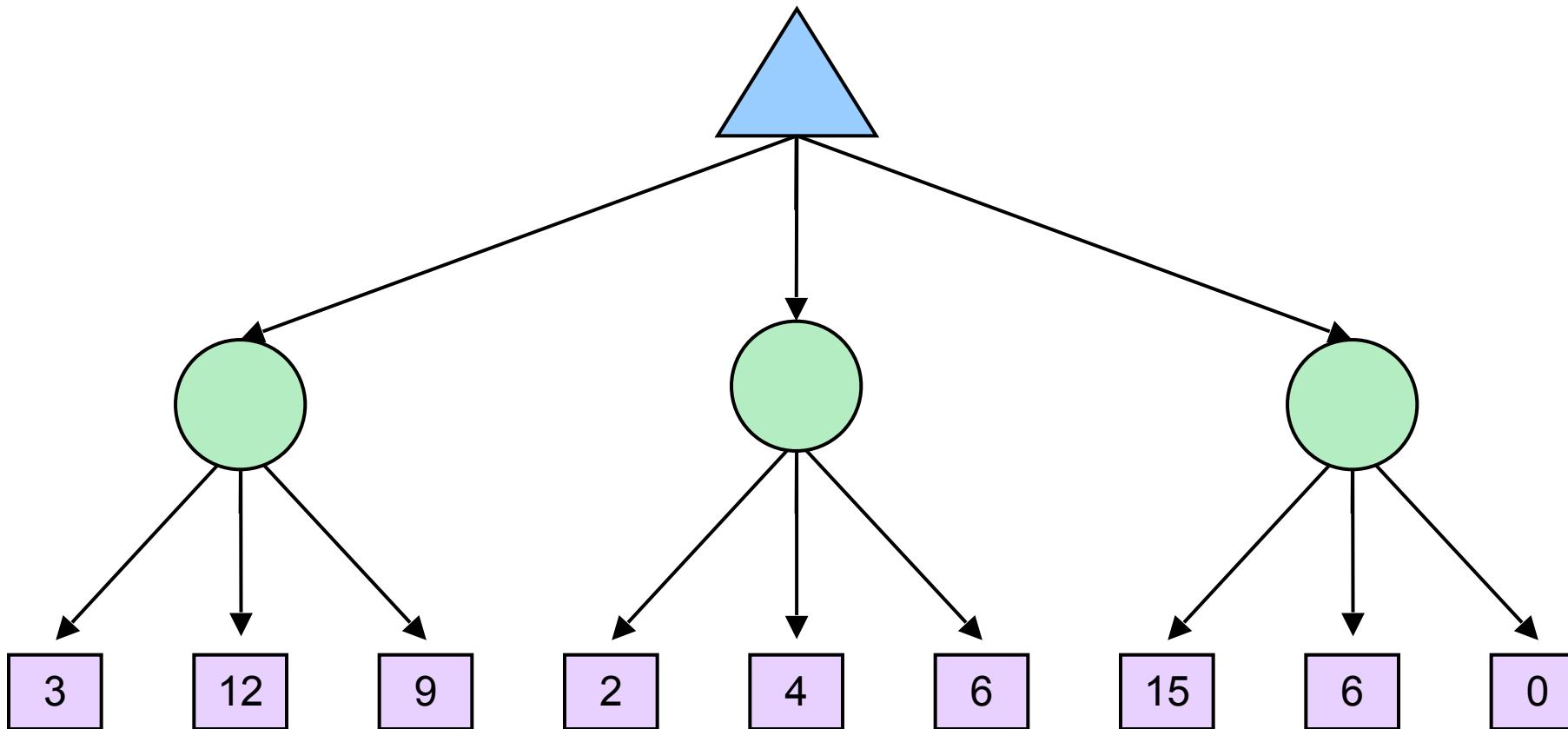
Expectimax Pseudocode

```
def exp-value(state):
    initialize v = 0
    for each successor of state:
        p =
            probability(successor)
        v += p * value(successor)
    return v
```

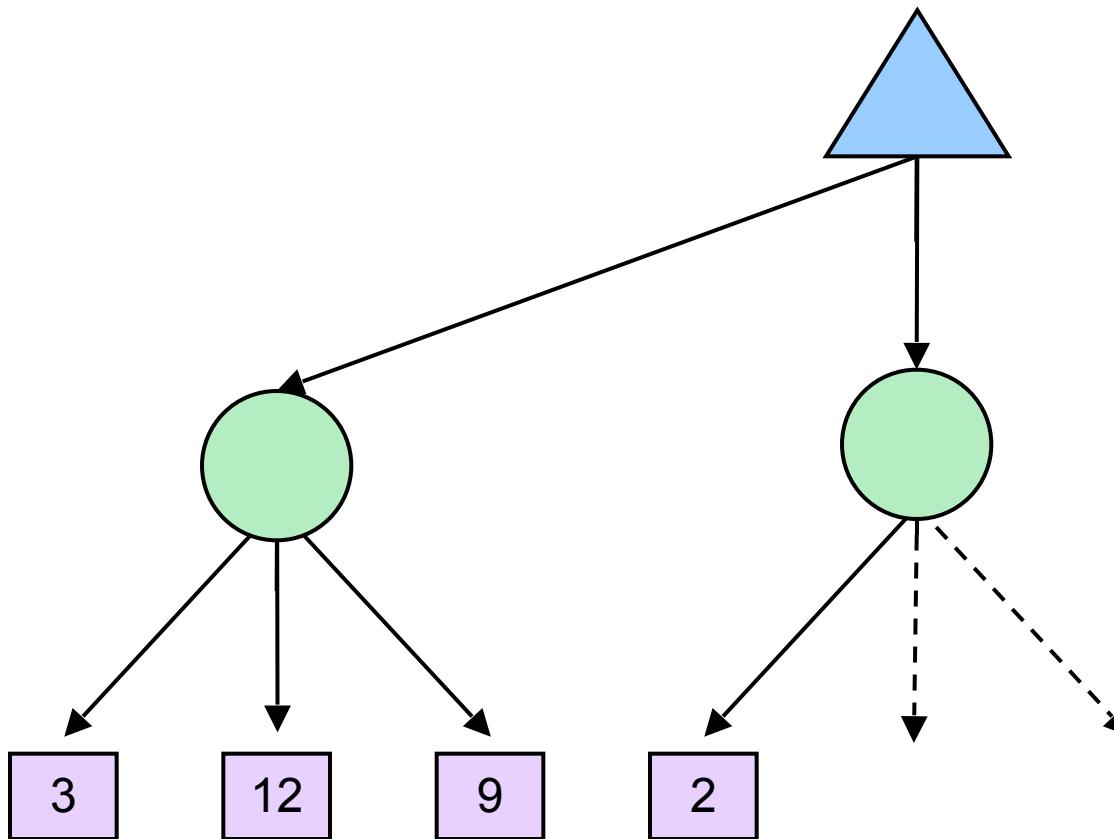


$$v = (1/2)(8) + (1/3)(24) + (1/6)(-12) = 10$$

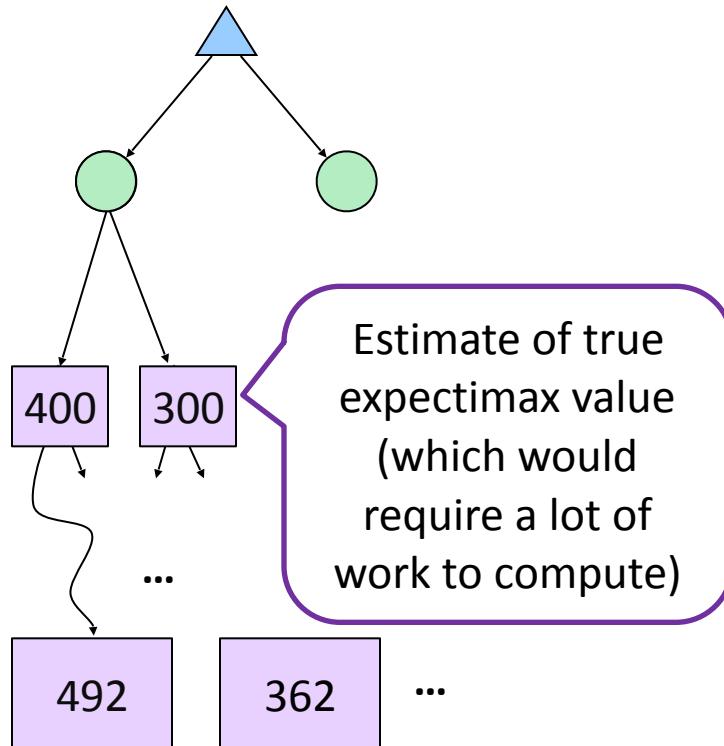
Expectimax Example



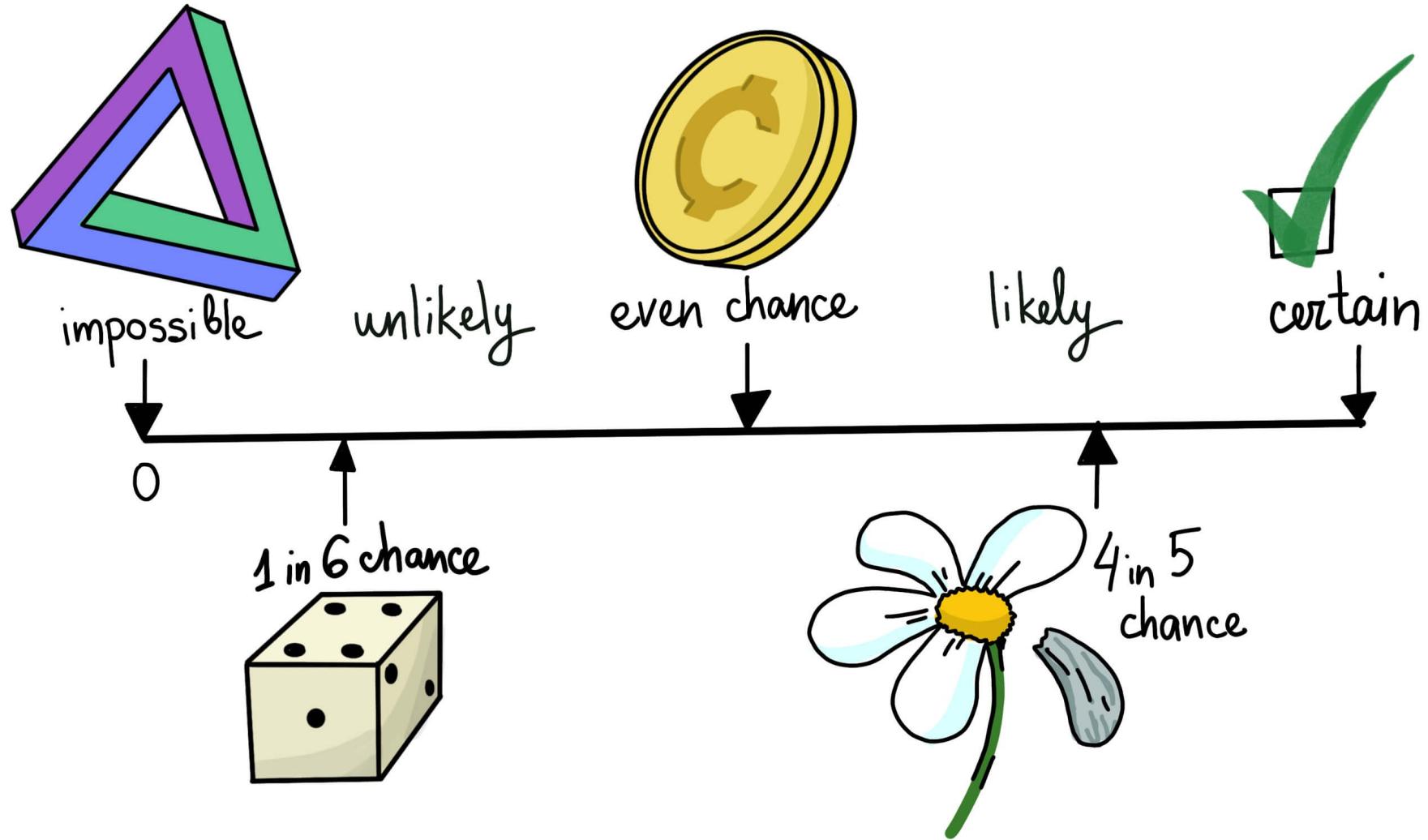
Expectimax Pruning?



Depth-Limited Expectimax



Probabilities



Probabilities

- A **random variable** represents an event whose outcome is unknown
- A **probability distribution** is an assignment of weights to outcomes
- Example: Traffic on freeway
 - Random variable: T = whether there's traffic
 - Outcomes: $T \in \{\text{none}, \text{light}, \text{heavy}\}$
 - Distribution: $P(T=\text{none}) = 0.25$, $P(T=\text{light}) = 0.50$, $P(T=\text{heavy}) = 0.25$
- Some laws of probability (more later):
 - Probabilities are always non-negative
 - Probabilities over all possible outcomes sum to one
- As we get more evidence, probabilities may change:
 - $P(T=\text{heavy}) = 0.25$, $P(T=\text{heavy} \mid \text{Hour}=8\text{am}) = 0.60$
 - We'll talk about methods for reasoning and updating probabilities later



0.25



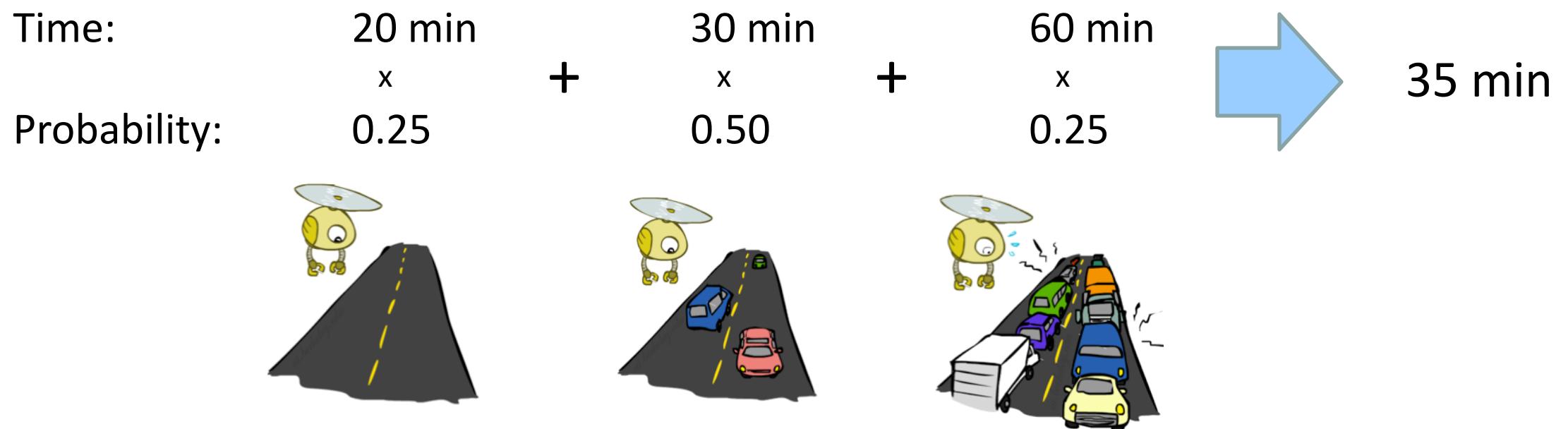
0.50



0.25

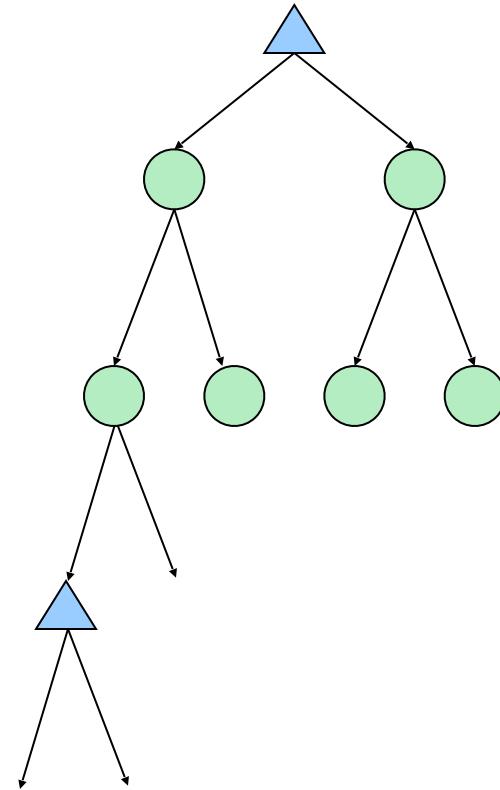
Expectations

- The expected value of a function of a random variable is the average, weighted by the probability distribution over outcomes
- Example: How long to get to the airport?



What Probabilities to Use?

- In expectimax search, we have a probabilistic model of how the opponent (or environment) will behave in any state
 - Model could be a simple uniform distribution (roll a die)
 - Model could be sophisticated and require a great deal of computation
 - We have a chance node for any outcome out of our control: opponent or environment
 - The model might say that adversarial actions are likely!
- For now, assume each chance node magically comes along with probabilities that specify the distribution over its outcomes



Having a probabilistic belief about another agent's action does not mean that the agent is flipping any coins!

What are Probabilities?

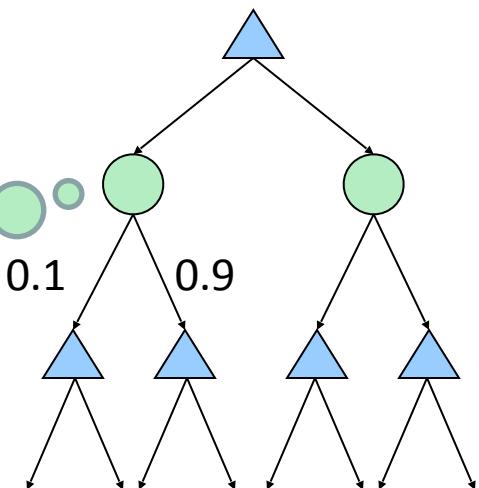
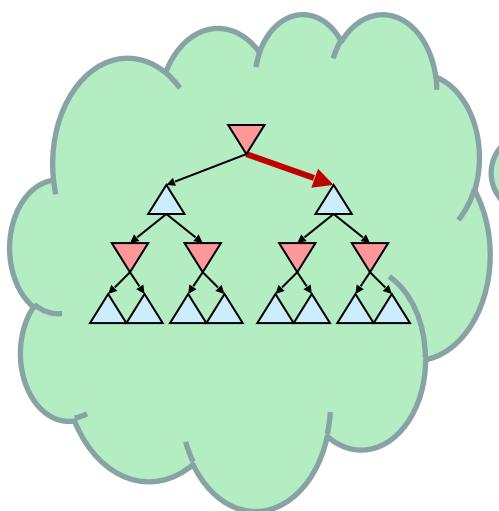
- Objectivist / frequentist answer:
 - Averages over repeated *experiments*
 - E.g. empirically estimating $P(\text{rain})$ from historical observation
 - Assertion about how future experiments will go (in the limit)
 - New evidence changes the *reference class*
 - Makes one think of *inherently random* events, like rolling dice

- Subjectivist / Bayesian answer:
 - Degrees of belief about unobserved variables
 - E.g. an agent's belief that it's raining, given the temperature
 - E.g. agent's belief how an opponent will behave, given the state
 - Often *learn* probabilities from past experiences (more later)
 - New evidence *updates beliefs* (more later)



Quiz: Informed Probabilities

- Let's say you know that your opponent is actually running a depth 2 minimax, using the result 80% of the time, and moving randomly otherwise
- Question: What tree search should you use?



- Answer: Expectimax!

- To figure out EACH chance node's probabilities, you have to run a simulation of your opponent
- This kind of thing gets very slow very quickly
- Even worse if you have to simulate your opponent simulating you...
- ... except for minimax, which has the nice property that it all collapses into one game tree

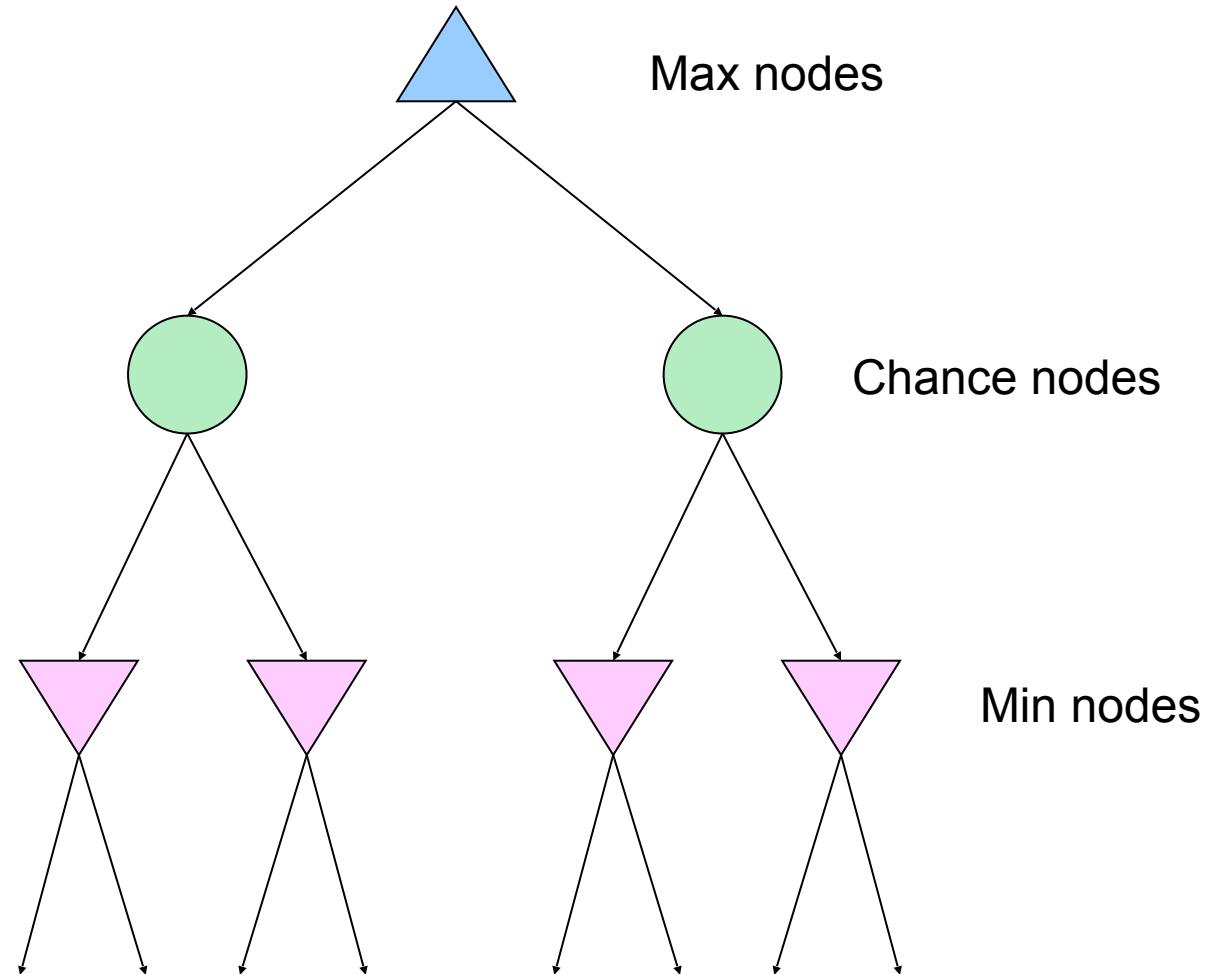
Example: Backgammon

- 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!



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

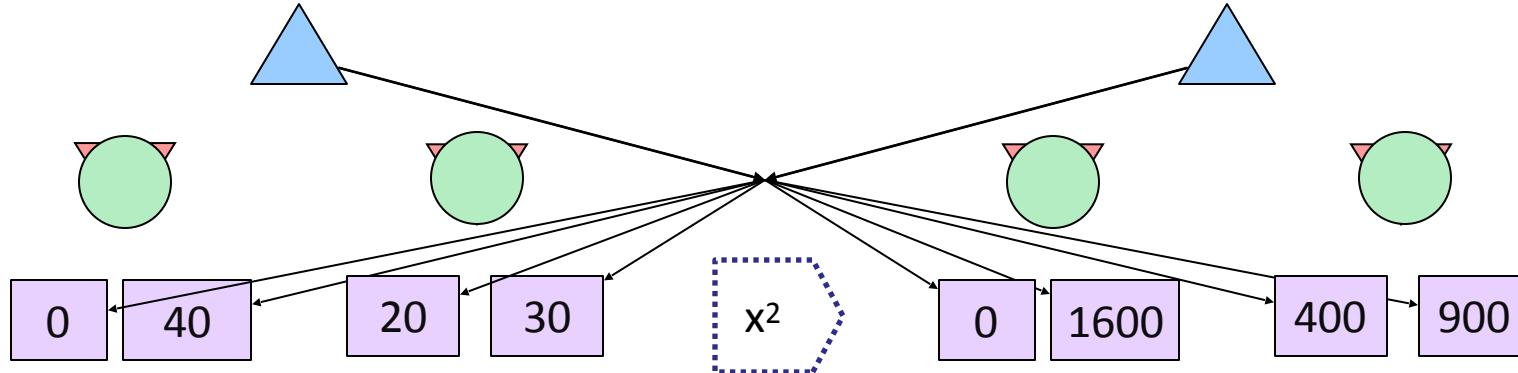


Utilities

Maximum Expected Utility

- Why should we average utilities? Why not minimax?
- Principle of maximum expected utility:
 - A rational agent should chose the action that **maximizes its expected utility, given its knowledge**
- Questions:
 - Where do utilities come from?
 - How do we know such utilities even exist?
 - How do we know that averaging even makes sense?
 - What if our behavior (preferences) can't be described by utilities?

What Utilities to Use?



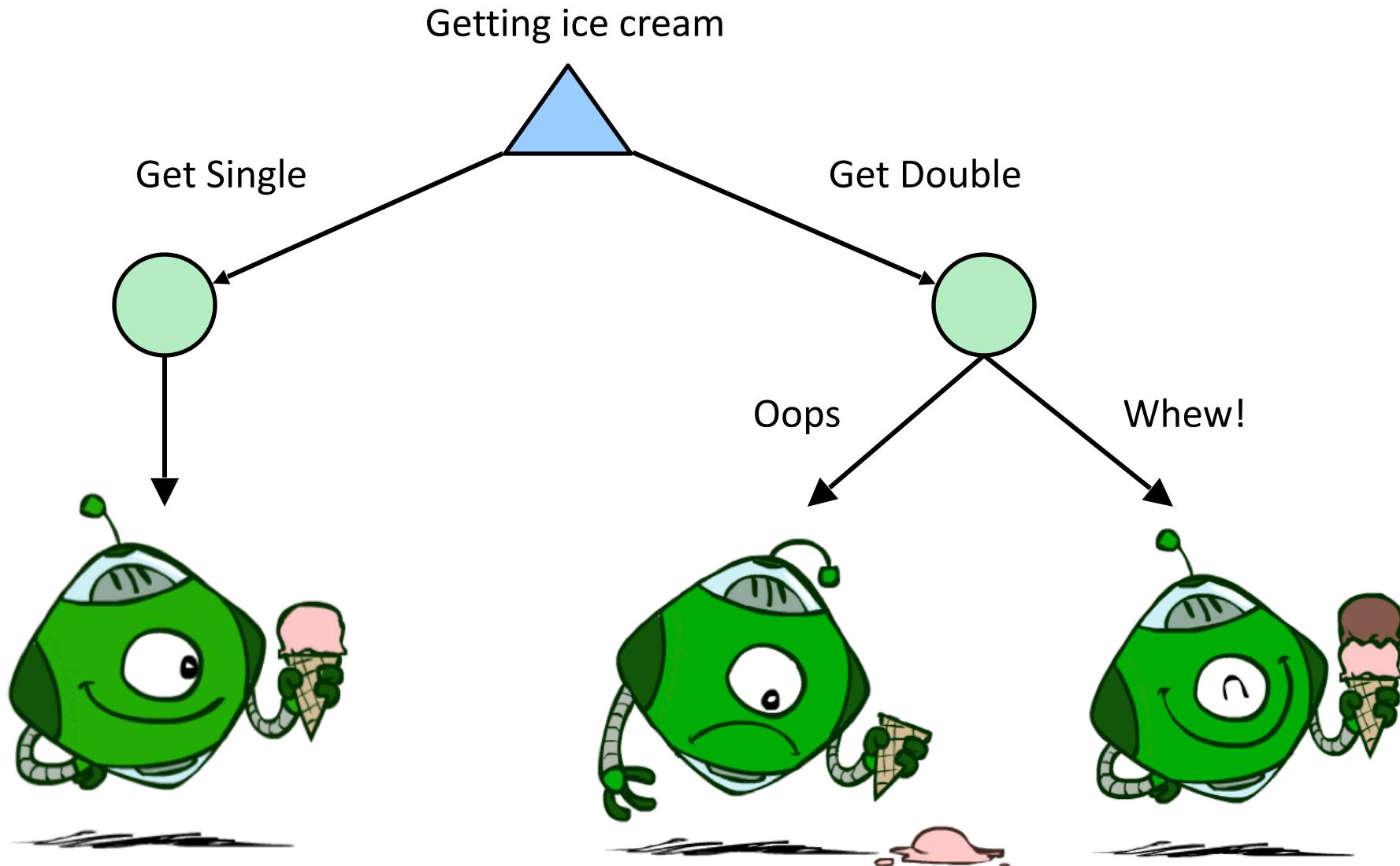
- For worst-case minimax reasoning, terminal function scale doesn't matter
 - We just want better states to have higher evaluations (get the ordering right)
 - We call this **insensitivity to monotonic transformations**
- For average-case expectimax reasoning, we need *magnitudes* to be meaningful

Utilities

- Utilities are functions from outcomes (states of the world) to real numbers that describe an agent's preferences
- Where do utilities come from?
 - In a game, may be simple (+1/-1)
 - Utilities summarize the agent's goals
 - Theorem: any “rational” preferences can be summarized as a utility function
- We hard-wire utilities and let behaviors emerge
 - Why don't we let agents pick utilities?
 - Why don't we prescribe behaviors?



Utilities: Uncertain Outcomes



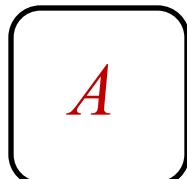
Preferences

- An agent must have preferences among:

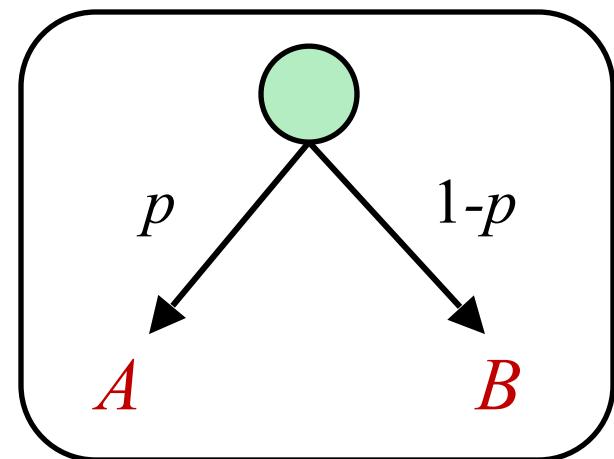
- Prizes: A , B , etc.
- Lotteries: situations with uncertain prizes

$$L = [p, A; (1-p), B]$$

A Prize



A Lottery

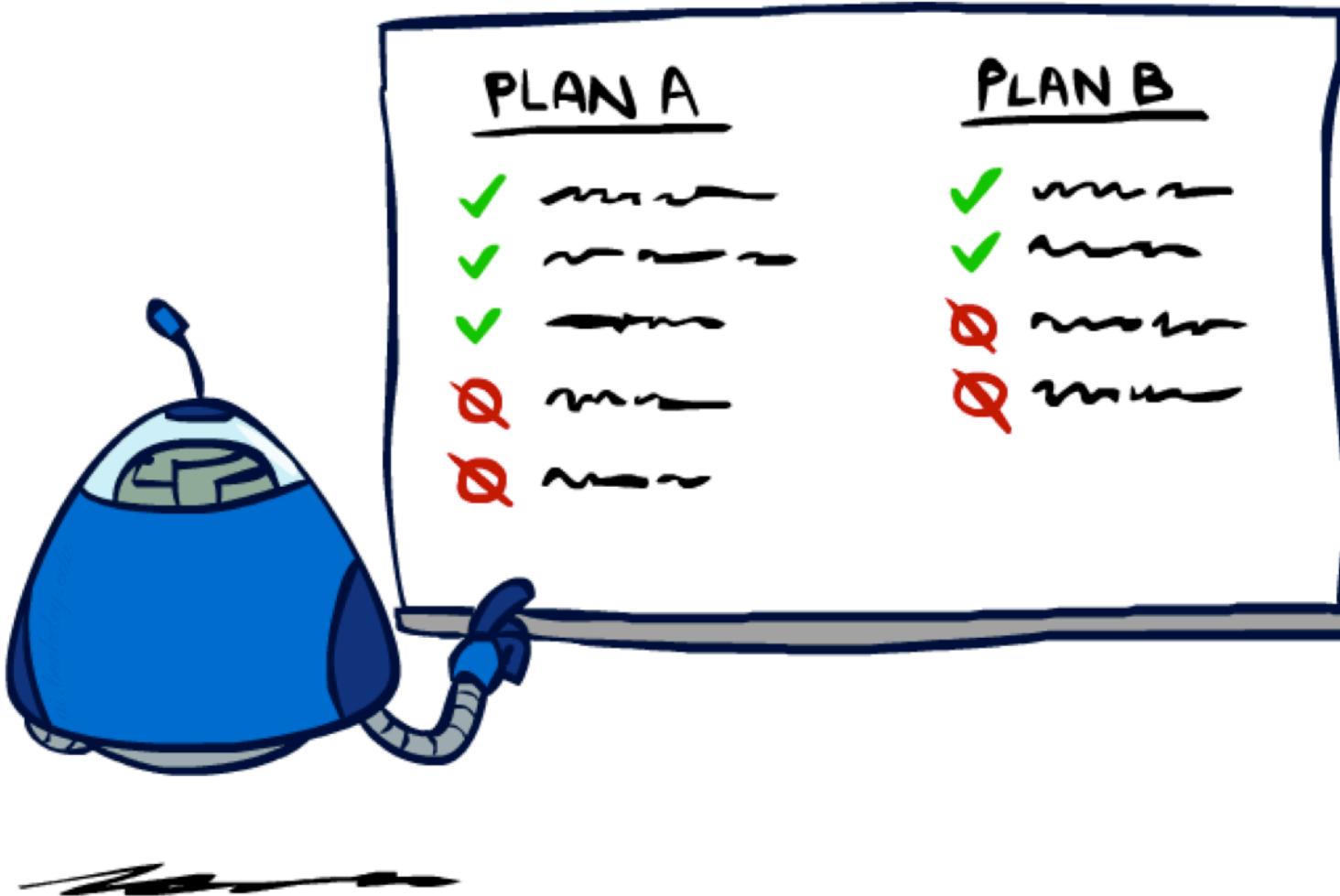


- Notation:

- Preference: $A \succ B$
- Indifference: $A \sim B$



Rationality

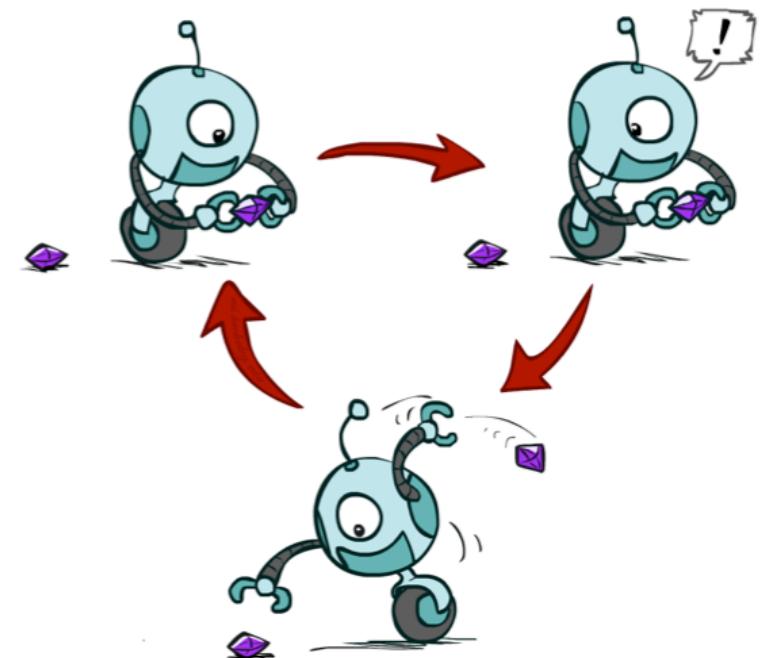


Rational Preferences

- We want some constraints on preferences before we call them rational, such as:

Axiom of Transitivity: $(A \succ B) \wedge (B \succ C) \Rightarrow (A \succ C)$

- For example: an agent with **intransitive preferences** can be induced to give away all of its money
 - If $B > C$, then an agent with C would pay (say) 1 cent to get B
 - If $A > B$, then an agent with B would pay (say) 1 cent to get A
 - If $C > A$, then an agent with A would pay (say) 1 cent to get C



Rational Preferences

The Axioms of Rationality

Orderability

$$(A \succ B) \vee (B \succ A) \vee (A \sim B)$$

Transitivity

$$(A \succ B) \wedge (B \succ C) \Rightarrow (A \succ C)$$

Continuity

$$A \succ B \succ C \Rightarrow \exists p [p, A; 1-p, C] \sim B$$

Substitutability

$$A \sim B \Rightarrow [p, A; 1-p, C] \sim [p, B; 1-p, C]$$

Monotonicity

$$A \succ B \Rightarrow$$

$$(p \geq q \Leftrightarrow [p, A; 1-p, B] \succeq [q, A; 1-q, B])$$



Theorem: Rational preferences imply behavior describable as maximization of expected utility

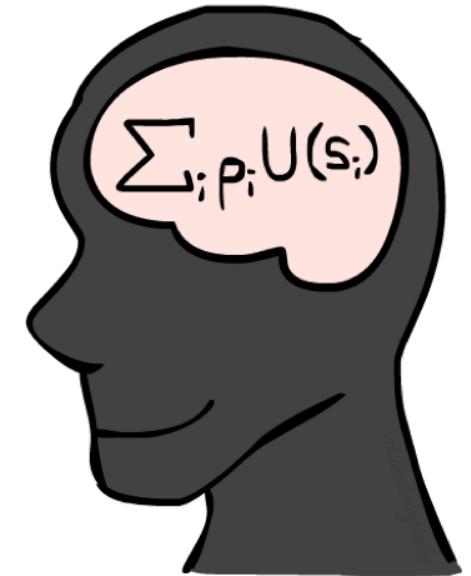
MEU Principle

- Theorem [Ramsey, 1931; von Neumann & Morgenstern, 1944]
 - Given any preferences satisfying these constraints, there exists a real-valued function U such that:

$$U(A) \geq U(B) \Leftrightarrow A \succeq B$$

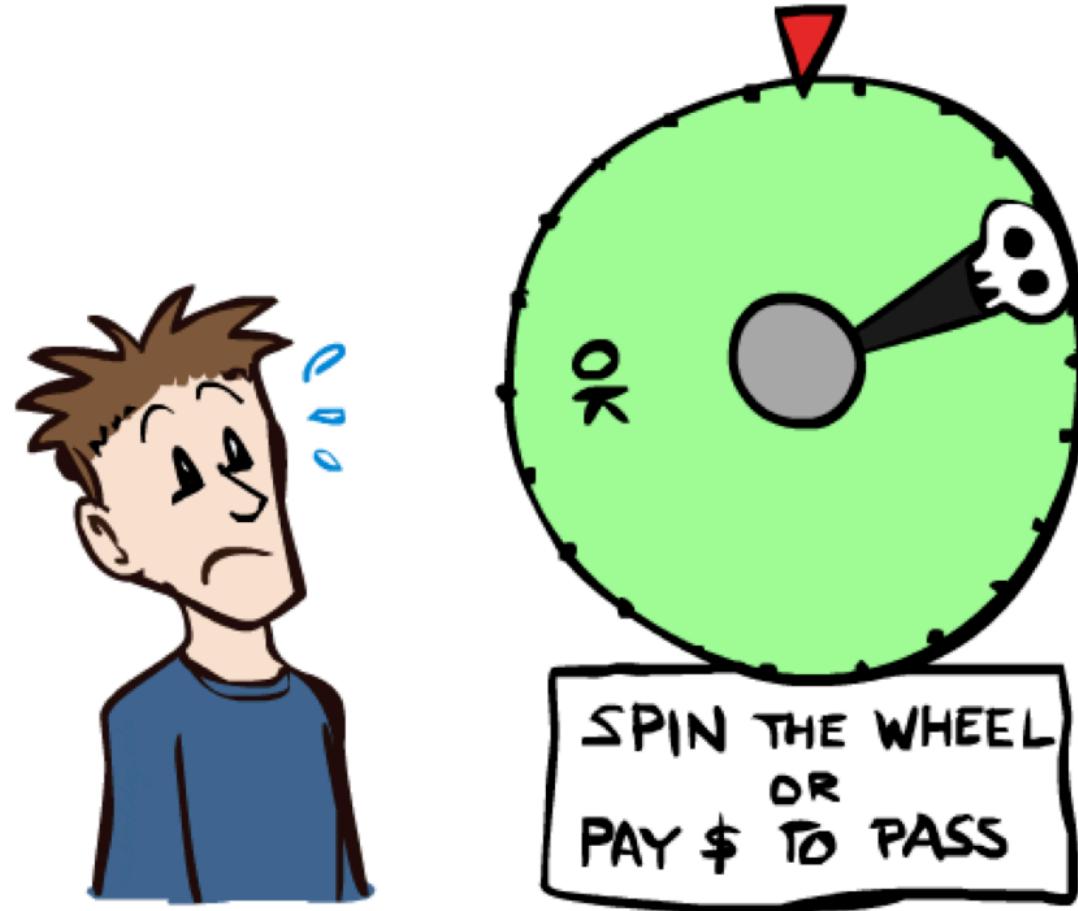
$$U([p_1, S_1; \dots; p_n, S_n]) = \sum_i p_i U(S_i)$$

- I.e. values assigned by U preserve preferences of both prizes and lotteries!



- Maximum expected utility (MEU) principle:
 - Choose the action that maximizes expected utility
 - Note: an agent can be entirely rational (consistent with MEU) without ever representing or manipulating utilities and probabilities
 - E.g., a lookup table for perfect tic-tac-toe, a reflex vacuum cleaner

Human Utilities



Utility Scales

- Normalized utilities: $u_+ = 1.0, u_- = 0.0$
- Micromorts: one-millionth chance of death, useful for paying to reduce product risks, etc.
- QALYs: quality-adjusted life years, useful for medical decisions involving substantial risk
- Note: behavior is invariant under positive linear transformation

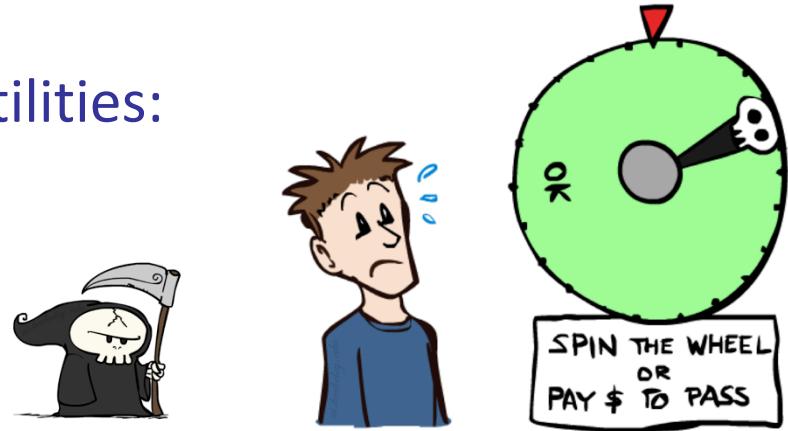
$$U'(x) = k_1 U(x) + k_2 \quad \text{where } k_1 > 0$$

- With deterministic prizes only (no lottery choices), only ordinal utility can be determined, i.e., total order on prizes



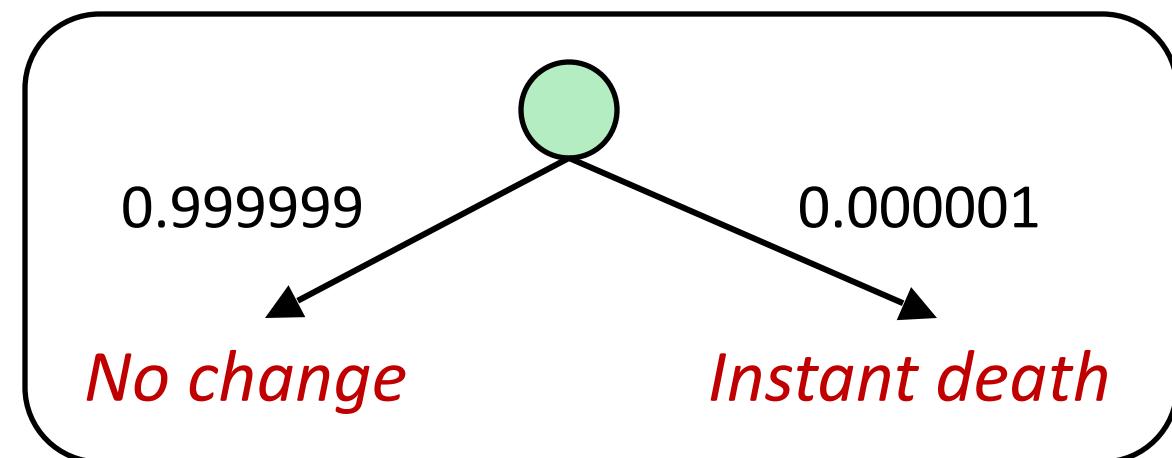
Human Utilities

- Utilities map states to real numbers. Which numbers?
- Standard approach to assessment (elicitation) of human utilities:
 - Compare a prize A to a **standard lottery** L_p between
 - “best possible prize” u_+ with probability p
 - “worst possible catastrophe” u_- with probability $1-p$
 - Adjust lottery probability p until indifference: $A \sim L_p$
 - Resulting p is a utility in $[0,1]$



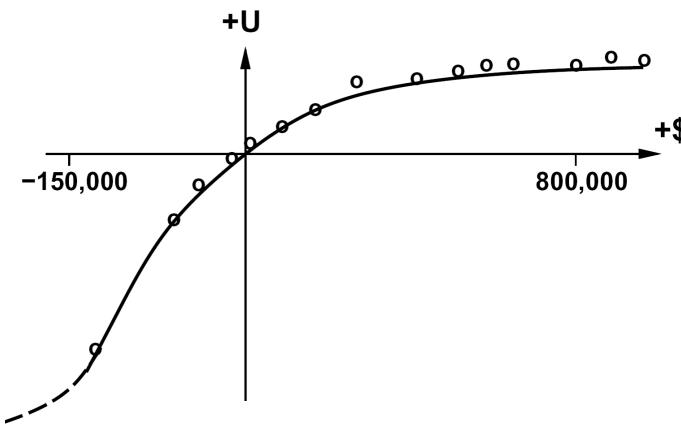
Pay \$30

~



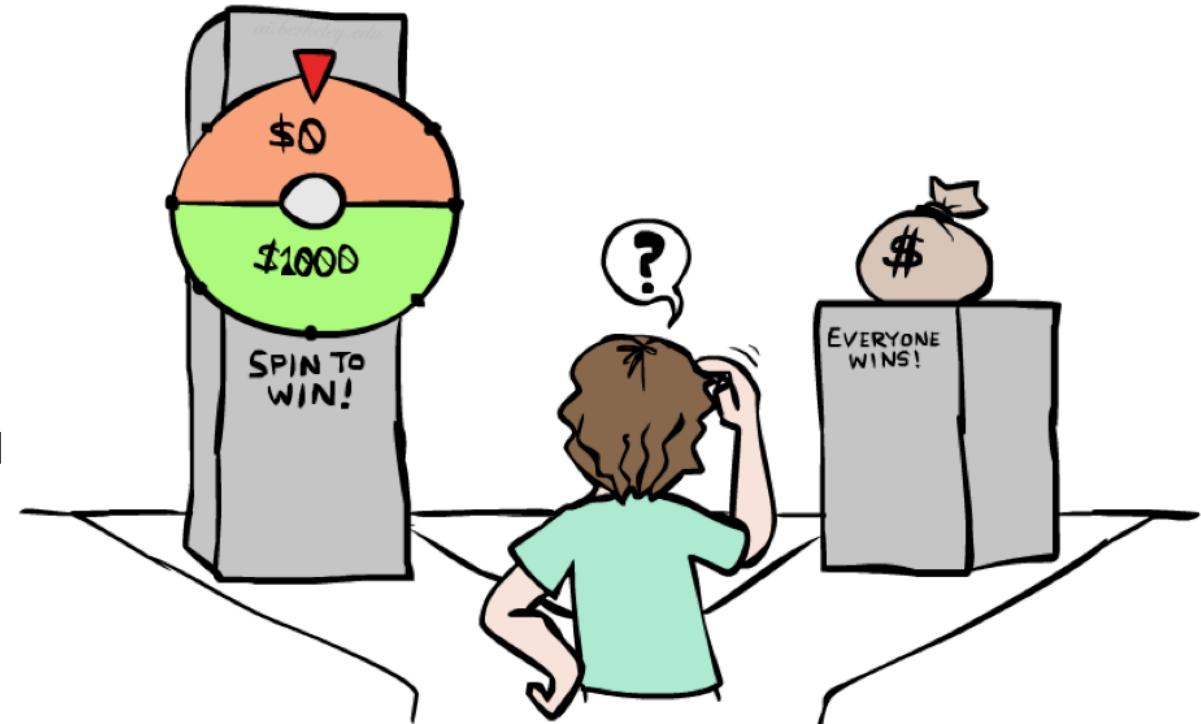
Money

- Money does not behave as a utility function, but we can talk about the utility of having money (or being in debt)
- Given a lottery $L = [p, \$X; (1-p), \$Y]$
 - The **expected monetary value** $EMV(L)$ is $p*X + (1-p)*Y$
 - $U(L) = p*U(\$X) + (1-p)*U(\$Y)$
 - Typically, $U(L) < U(EMV(L))$
 - In this sense, people are **risk-averse**
 - When deep in debt, people are **risk-prone**



Example: Insurance

- Consider the lottery [0.5, \$1000; 0.5, \$0]
 - What is its **expected monetary value**? (\$500)
 - What is its **certainty equivalent**?
 - Monetary value acceptable in lieu of lottery
 - \$400 for most people
 - Difference of \$100 is the **insurance premium**
 - There's an insurance industry because people will pay to reduce their risk
 - If everyone were risk-neutral, no insurance needed!
 - It's win-win: you'd rather have the \$400 and the insurance company would rather have the lottery (their utility curve is linear and they have many lotteries)



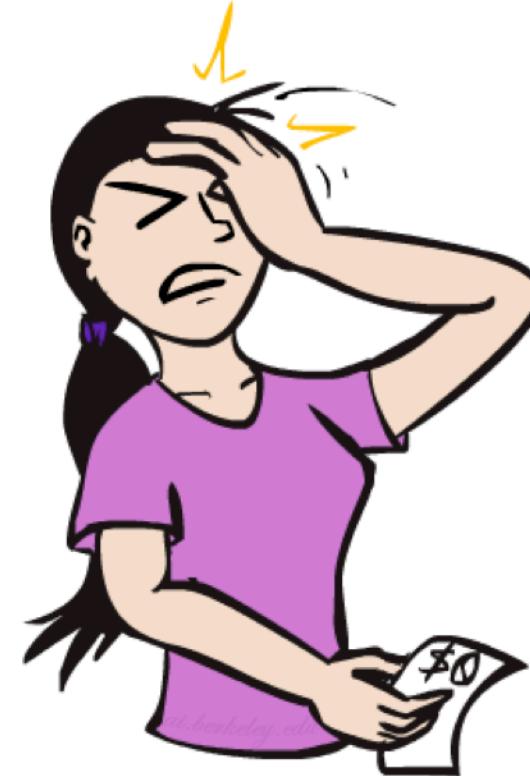
Example: Human Rationality?

- Famous example of Allais (1953)

- A: [0.8, \$4k; 0.2, \$0] ←
- B: [1.0, \$3k; 0.0, \$0]
- C: [0.2, \$4k; 0.8, \$0]
- D: [0.25, \$3k; 0.75, \$0]

- Most people prefer B > A, C > D

- But if $U(\$0) = 0$, then
 - $B > A \Rightarrow U(\$3k) > 0.8 U(\$4k)$
 - $C > D \Rightarrow 0.8 U(\$4k) > U(\$3k)$



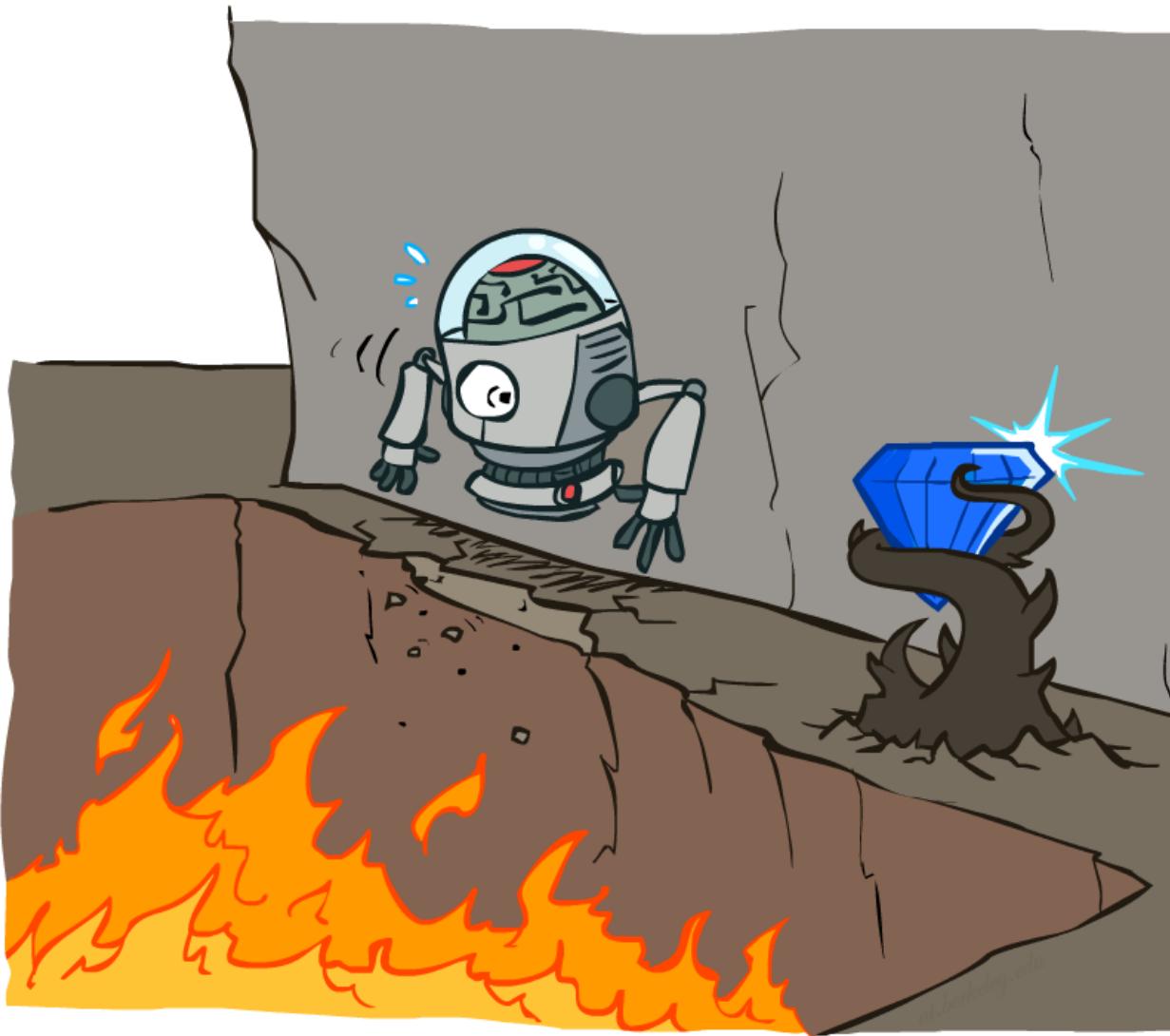
Markov Decision Processes



Slides courtesy of Dan Klein and Pieter Abbeel

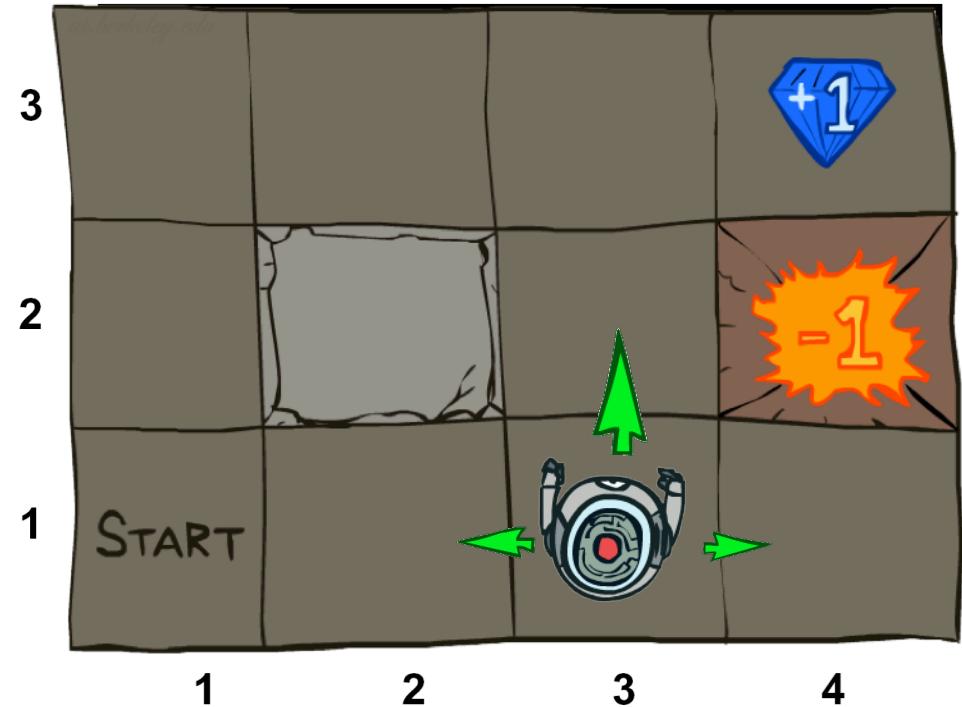
University of California, Berkeley

Non-Deterministic Search



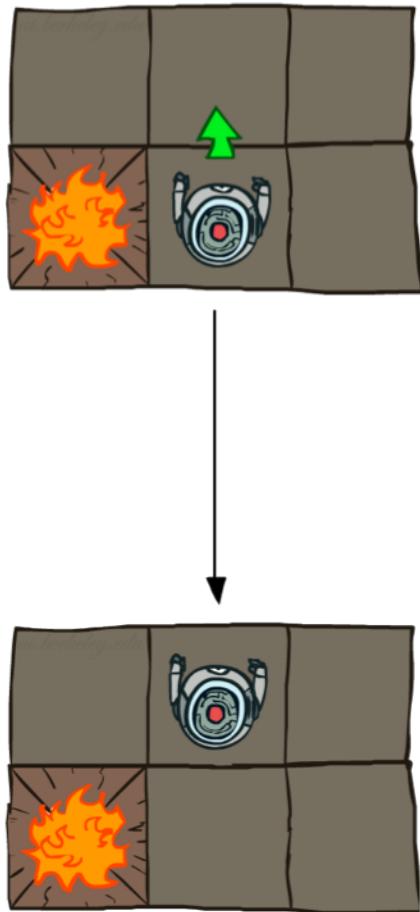
Example: Grid World

- 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)
 - Big rewards come at the end (good or bad)
- Goal: maximize sum of rewards

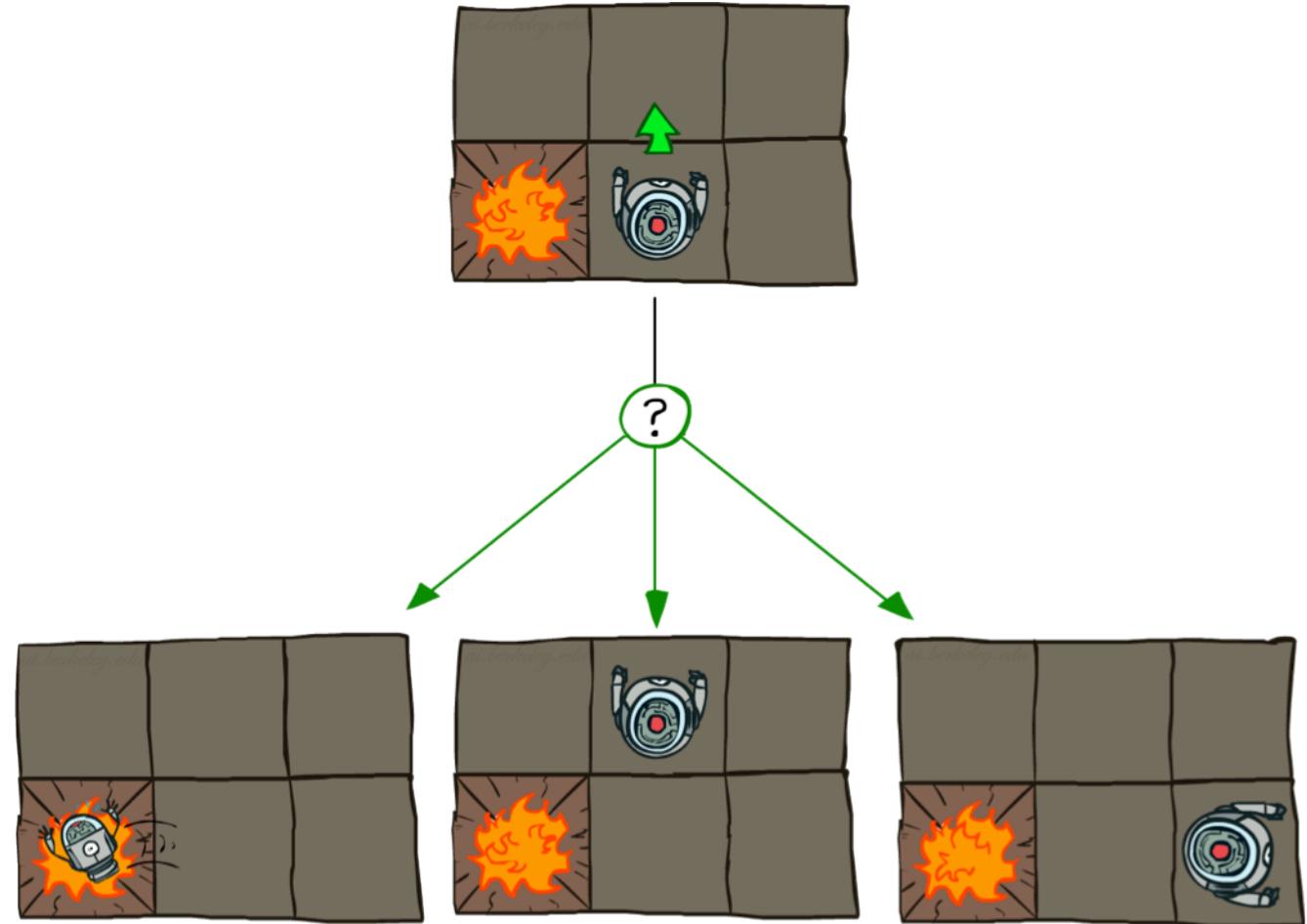


Grid World Actions

Deterministic Grid World

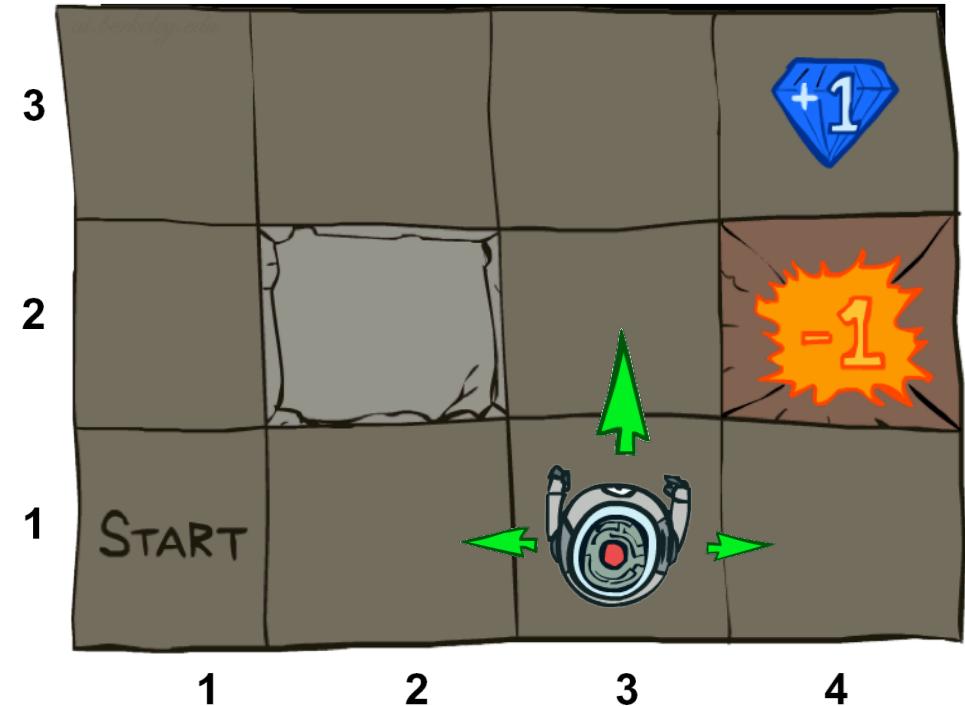


Stochastic Grid World



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



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

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

=

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

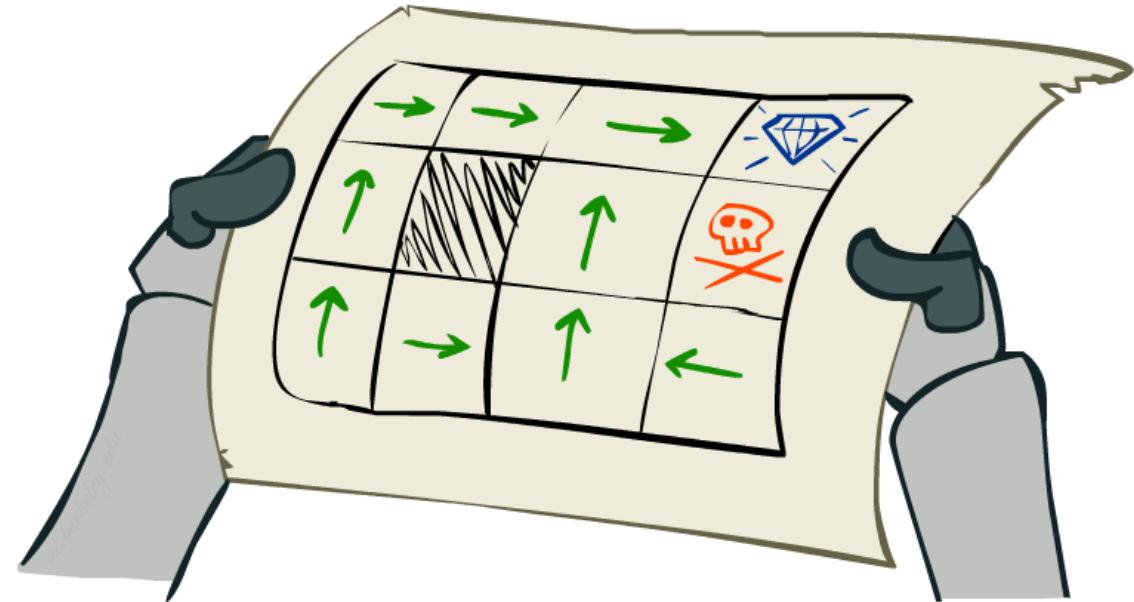


Andrey Markov
(1856-1922)

- This is just like search, where the successor function could only depend on the current state (not the history)

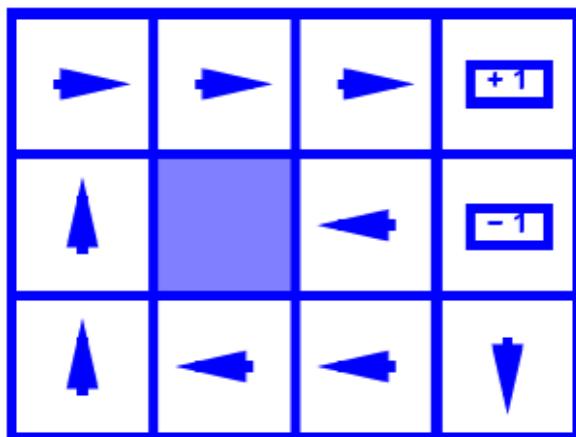
Policies

- In deterministic single-agent search problems, we wanted 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
 - An explicit policy defines a reflex agent
- Expectimax didn't compute entire policies
 - It computed the action for a single state only

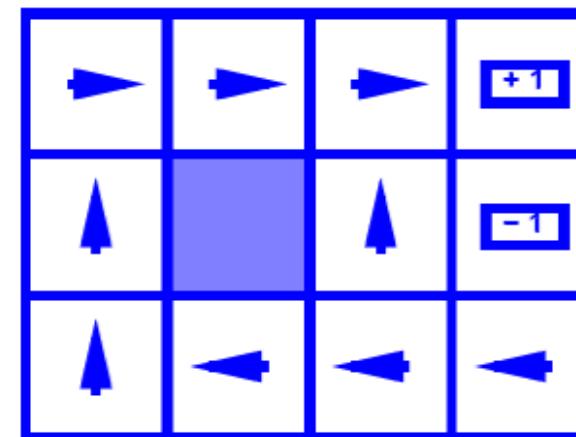


Optimal policy when $R(s, a, s') = -0.03$
for all non-terminals s

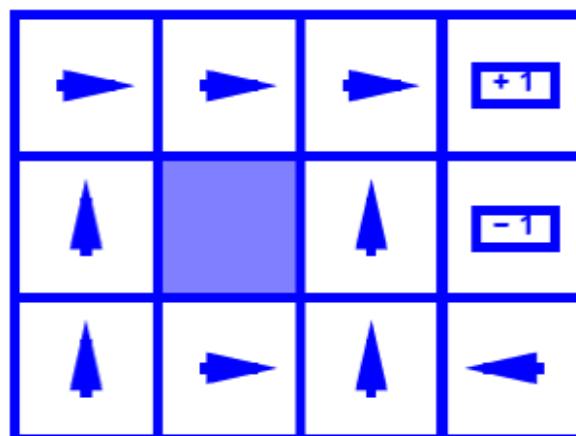
Optimal Policies



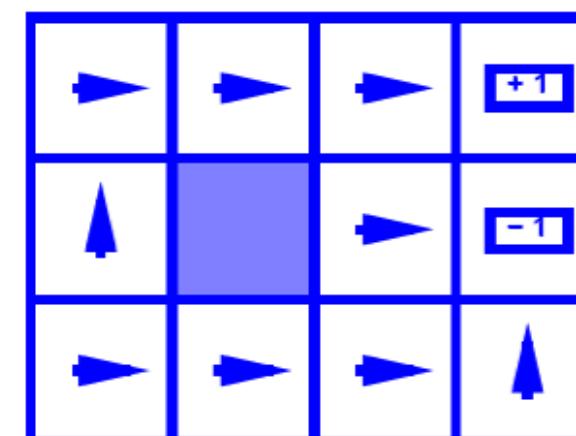
$$R(s) = -0.01$$



$$R(s) = -0.03$$

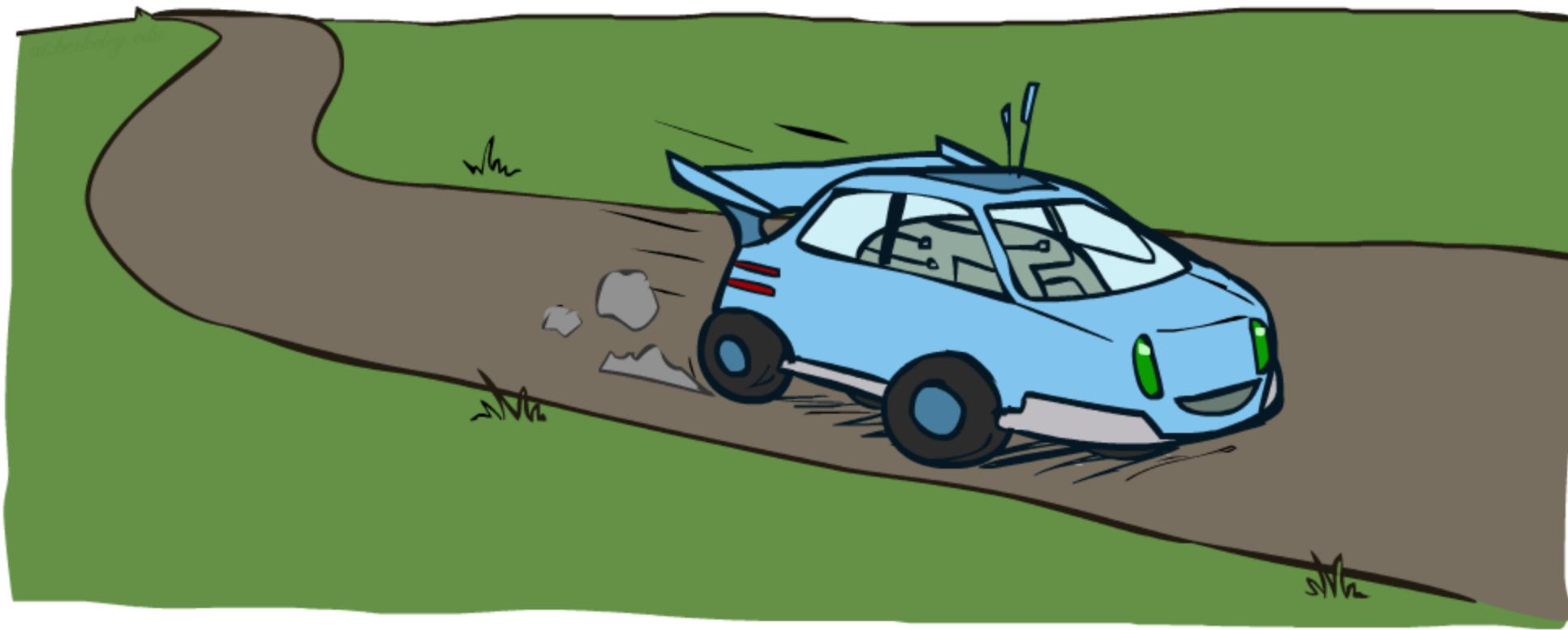


$$R(s) = -0.4$$



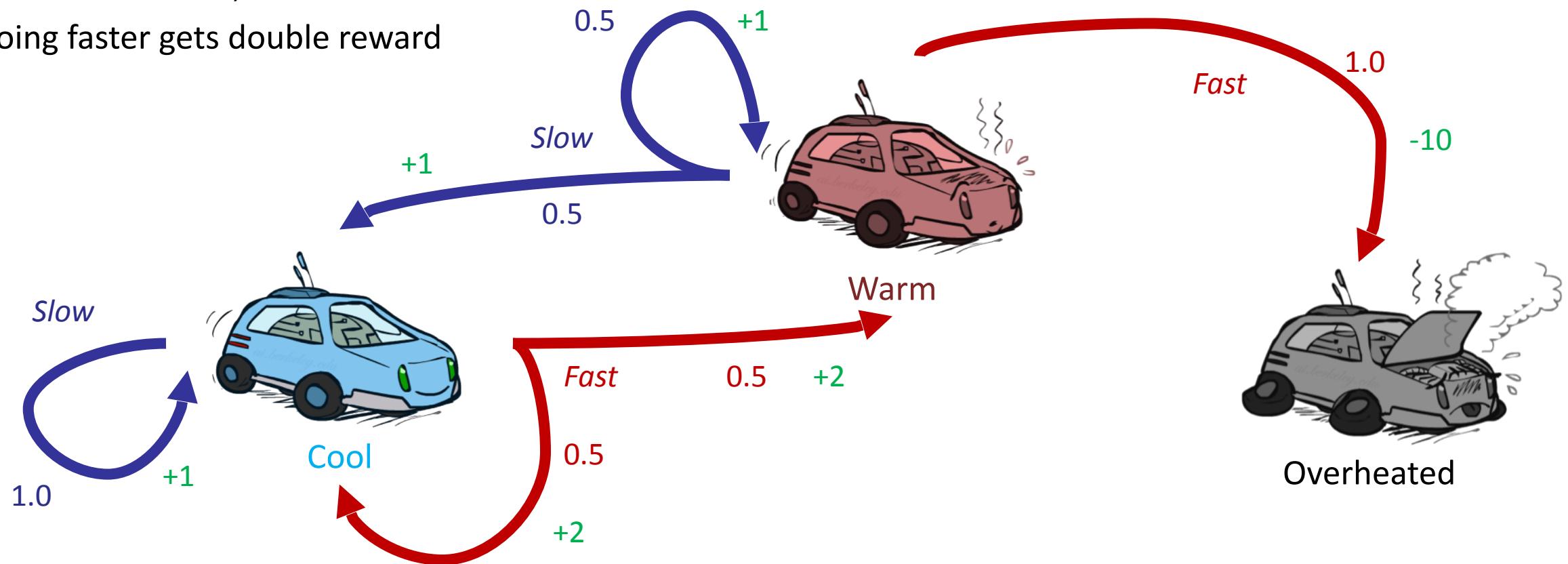
$$R(s) = -2.0$$

Example: Racing

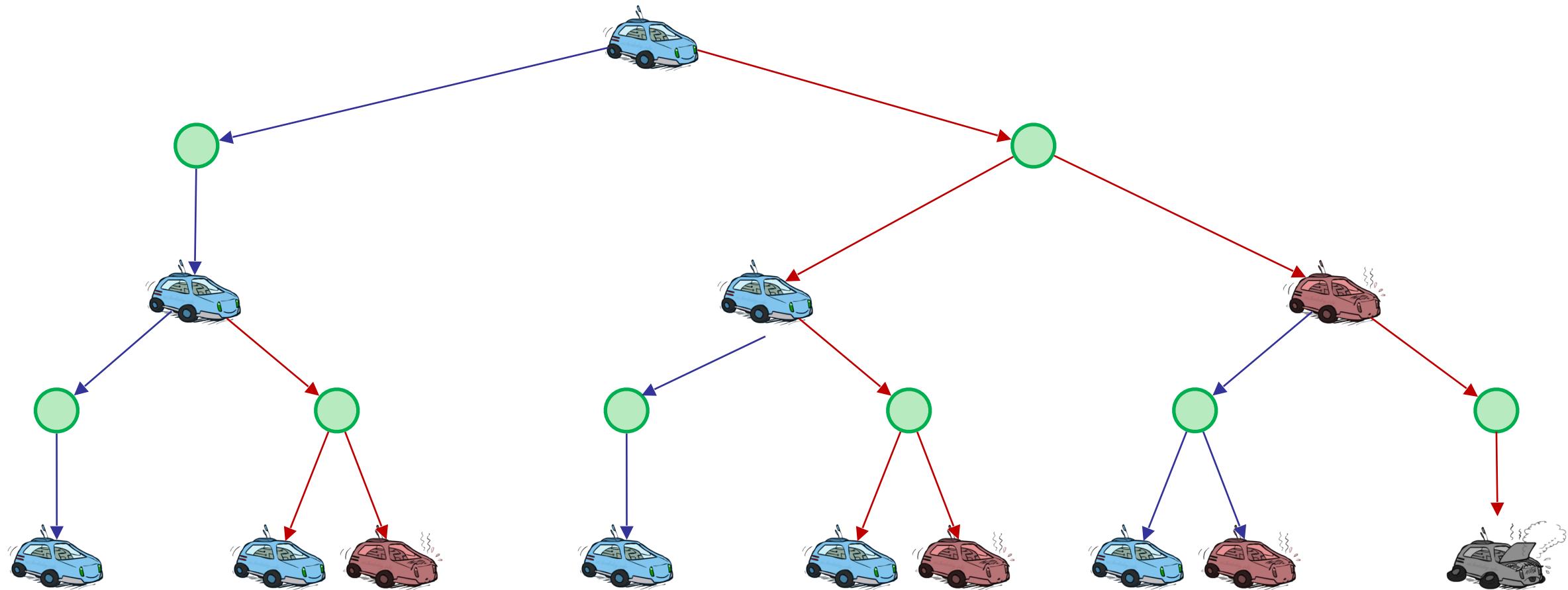


Example: Racing

- A robot car wants to travel far, quickly
- Three states: **Cool**, **Warm**, **Overheated**
- Two actions: *Slow*, *Fast*
- Going faster gets double reward

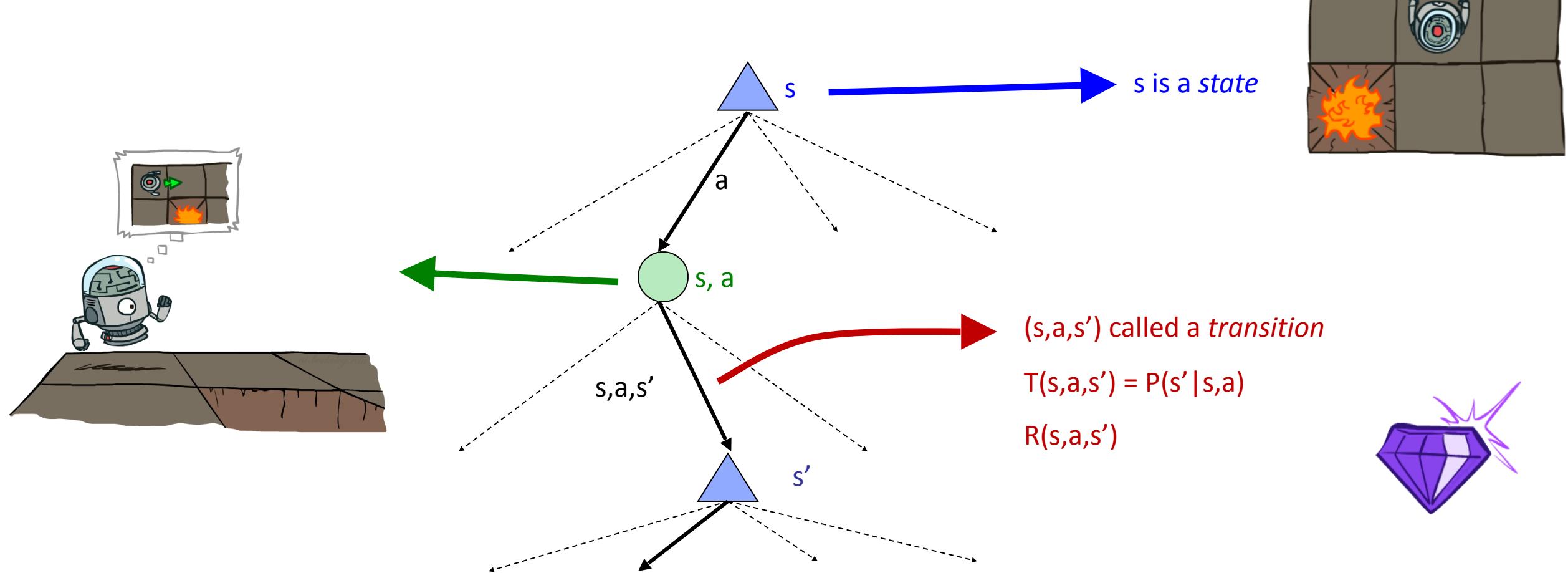


Racing Search Tree

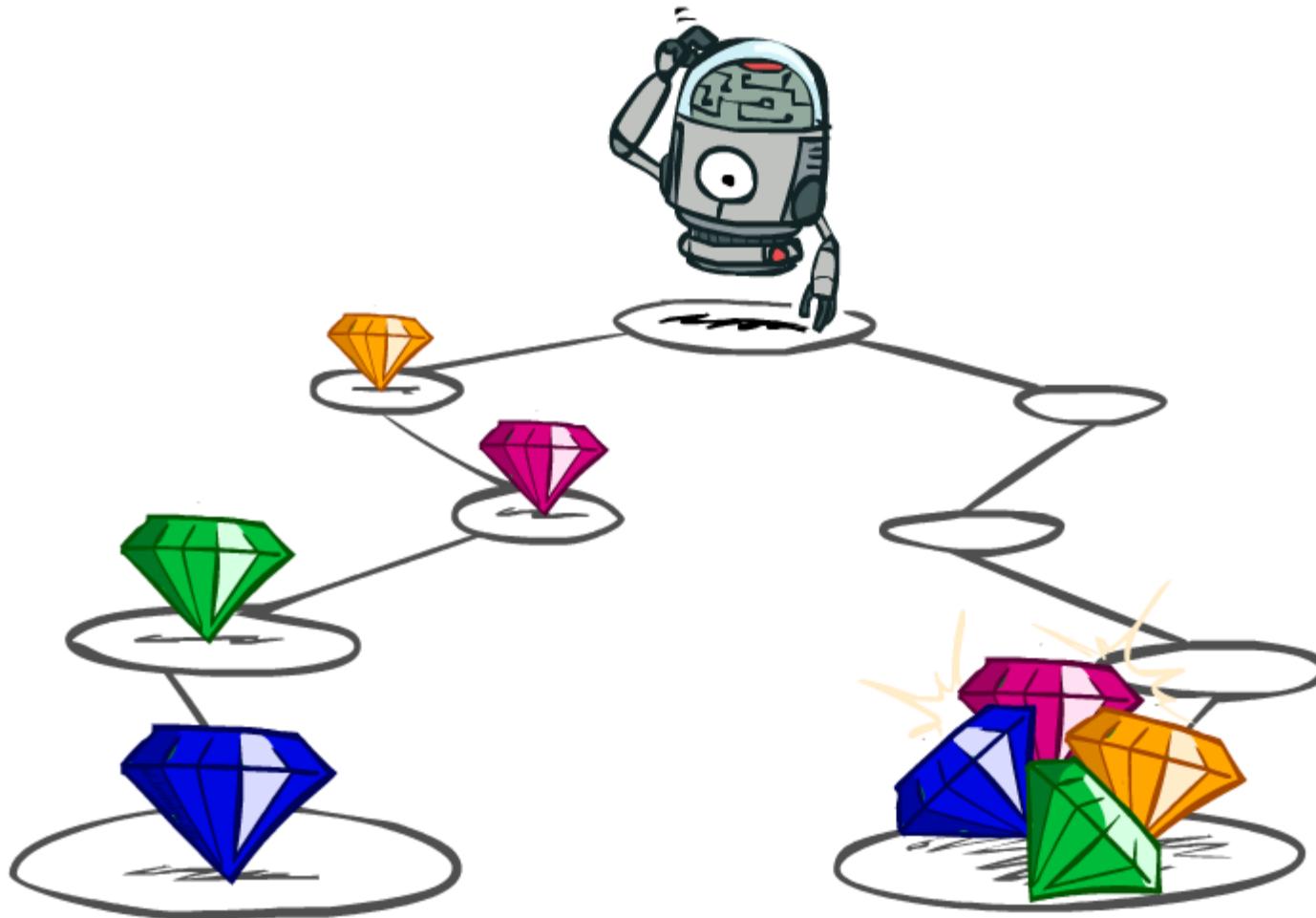


MDP Search Trees

- Each MDP state projects an expectimax-like search tree

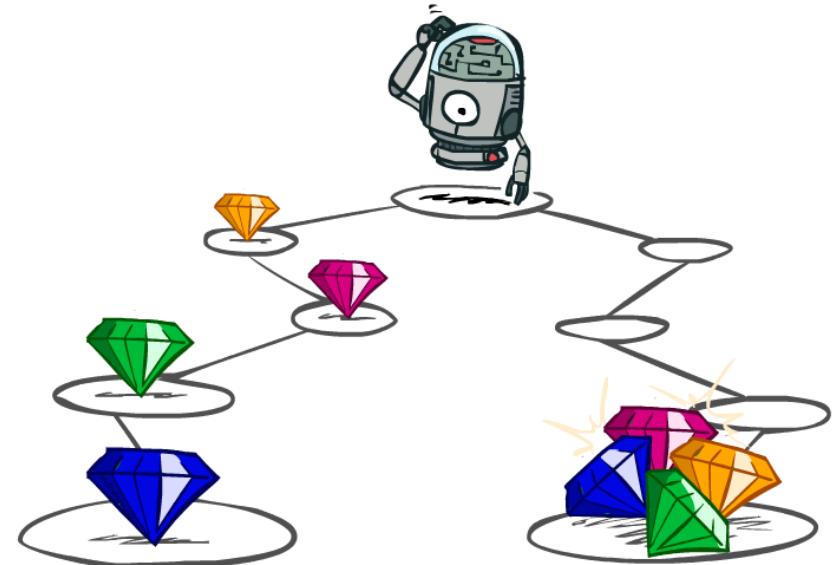


Utilities of Sequences



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]



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



γ^2

Worth In Two Steps

Discounting

- How to discount?

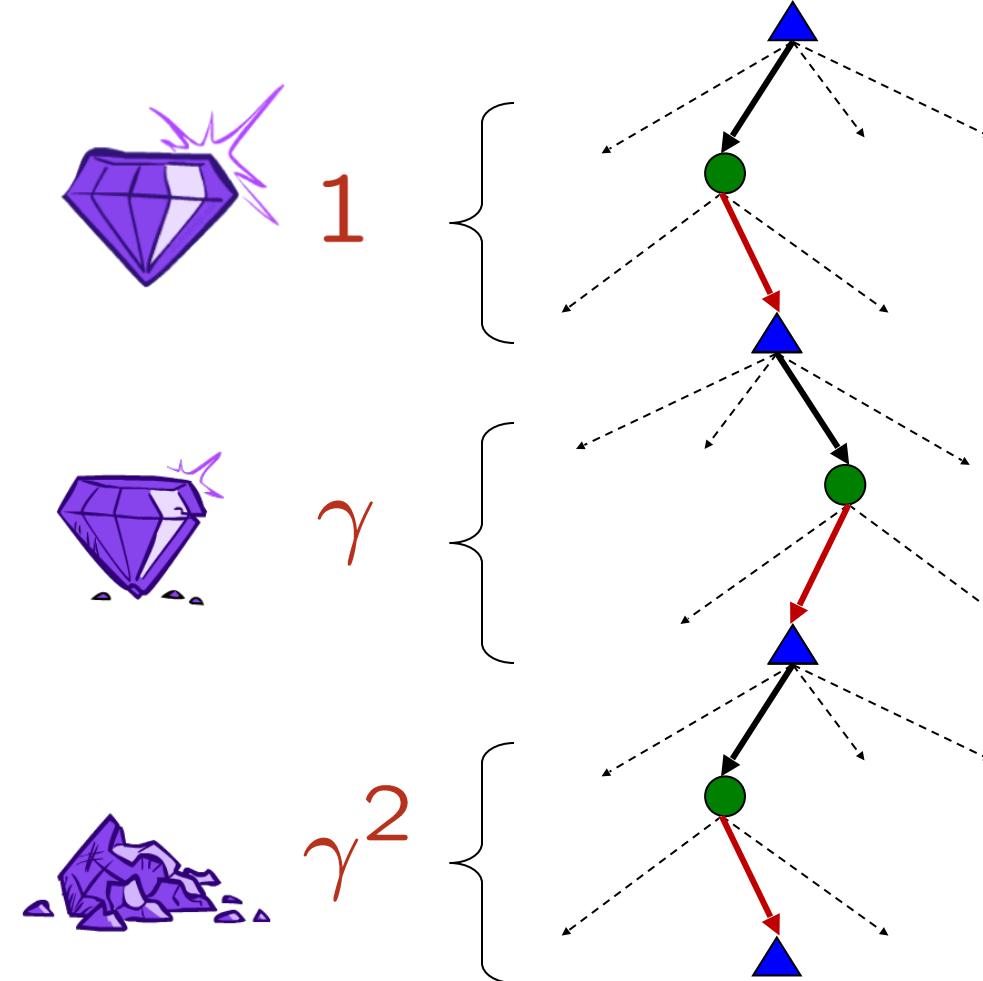
- Each time we descend a level, we multiply in the discount once

- Why discount?

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

- Example: discount of 0.5

- $U([1,2,3]) = 1*1 + 0.5*2 + 0.25*3$
- $U([1,2,3]) < U([3,2,1])$



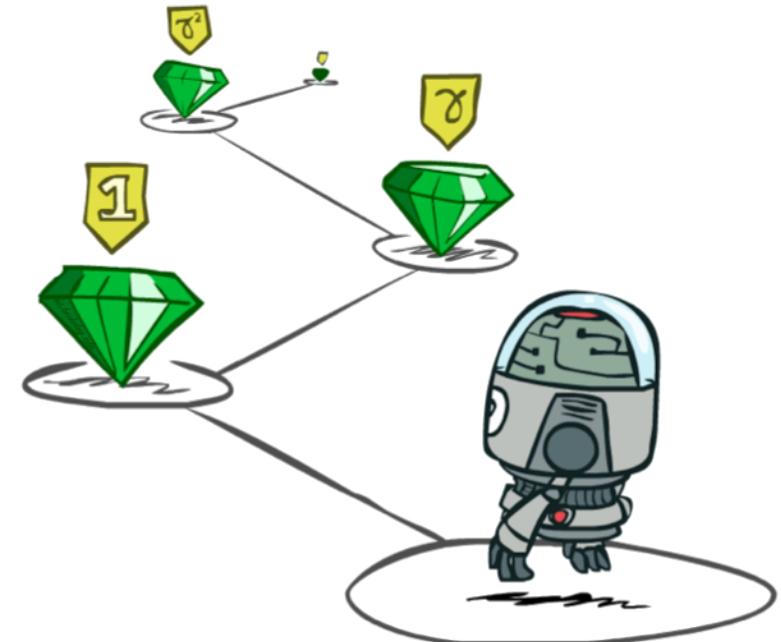
Stationary Preferences

- Theorem: if we assume stationary preferences:

$$[a_1, a_2, \dots] \succ [b_1, b_2, \dots]$$

\Updownarrow

$$[r, a_1, a_2, \dots] \succ [r, b_1, b_2, \dots]$$



- Then: there are only two ways to define utilities

- Additive utility:

$$U([r_0, r_1, r_2, \dots]) = r_0 + r_1 + r_2 + \dots$$

- Discounted utility:

$$U([r_0, r_1, r_2, \dots]) = r_0 + \gamma r_1 + \gamma^2 r_2 \dots$$

Infinite Utilities?!

- Problem: What if the game lasts forever? Do we get infinite rewards?

- Solutions:

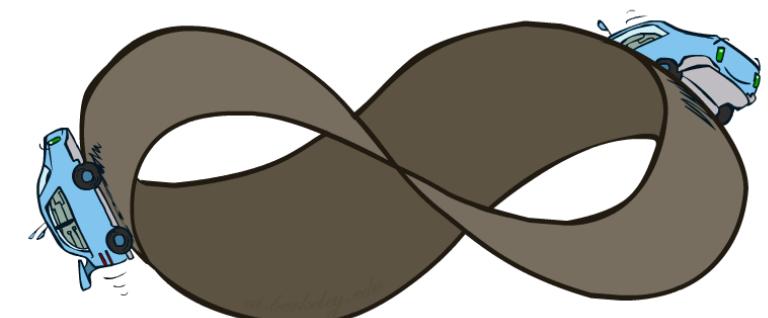
- Finite horizon: (similar to depth-limited search)

- Terminate episodes after a fixed T steps (e.g. life)
 - Gives nonstationary policies (π depends on time left)

- Discounting: use $0 < \gamma < 1$

$$U([r_0, \dots r_\infty]) = \sum_{t=0}^{\infty} \gamma^t r_t \leq R_{\max}/(1 - \gamma)$$

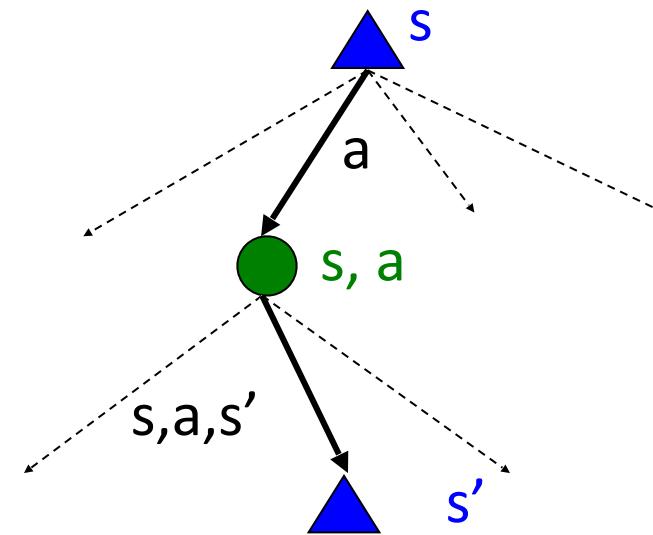
- Smaller γ means smaller “horizon” – shorter term focus
 - Absorbing state: guarantee that for every policy, a terminal state will eventually be reached (like “overheated” for racing)



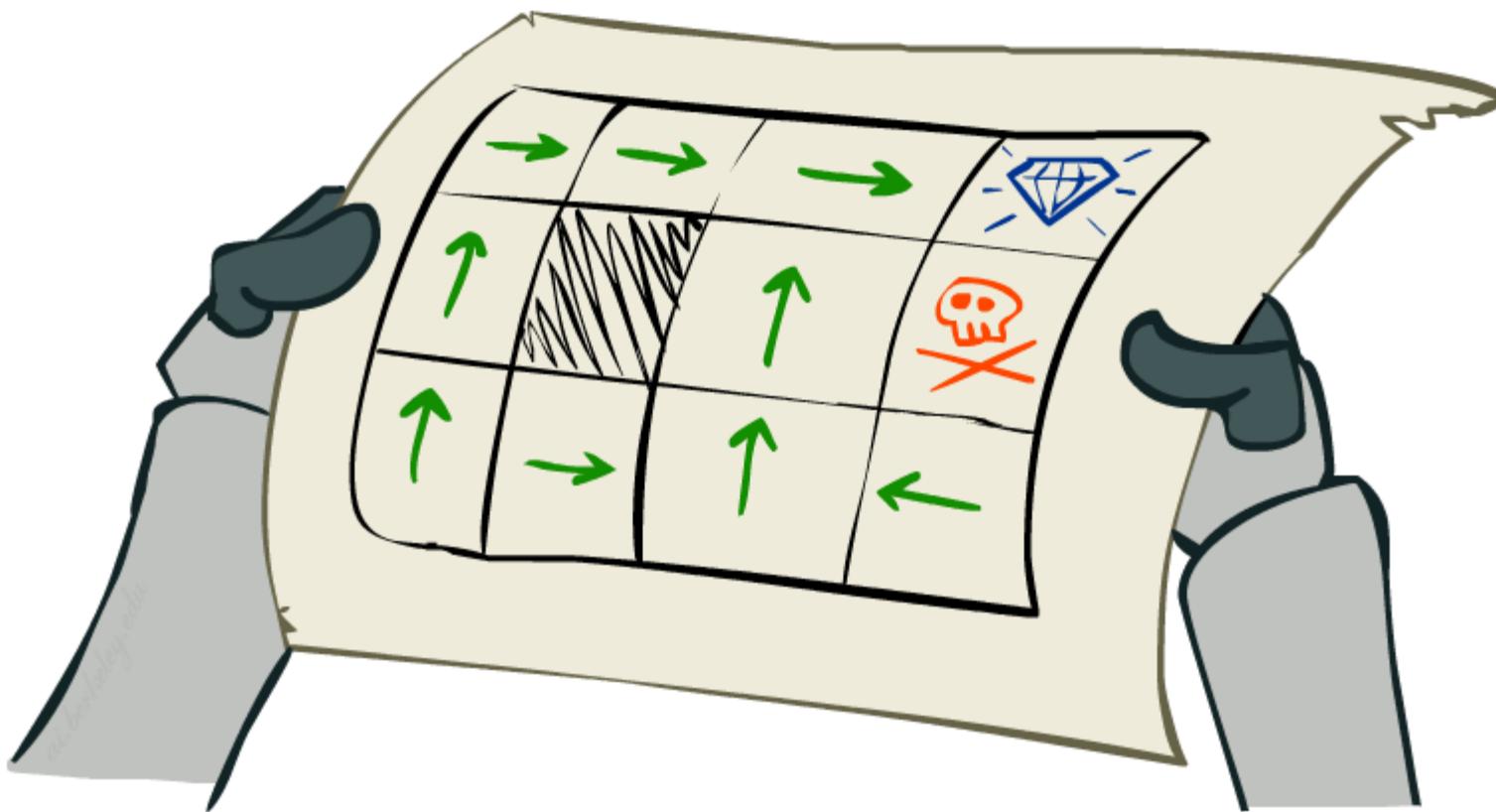
Recap: Defining MDPs

- Markov decision processes:
 - Set of states S
 - Start state s_0
 - Set of actions A
 - Transitions $P(s'|s,a)$ (or $T(s,a,s')$)
 - Rewards $R(s,a,s')$ (and discount γ)

- MDP quantities so far:
 - Policy = Choice of action for each state
 - Utility = sum of (discounted) rewards



Solving MDPs



Optimal Quantities

- The value (utility) of a state s :

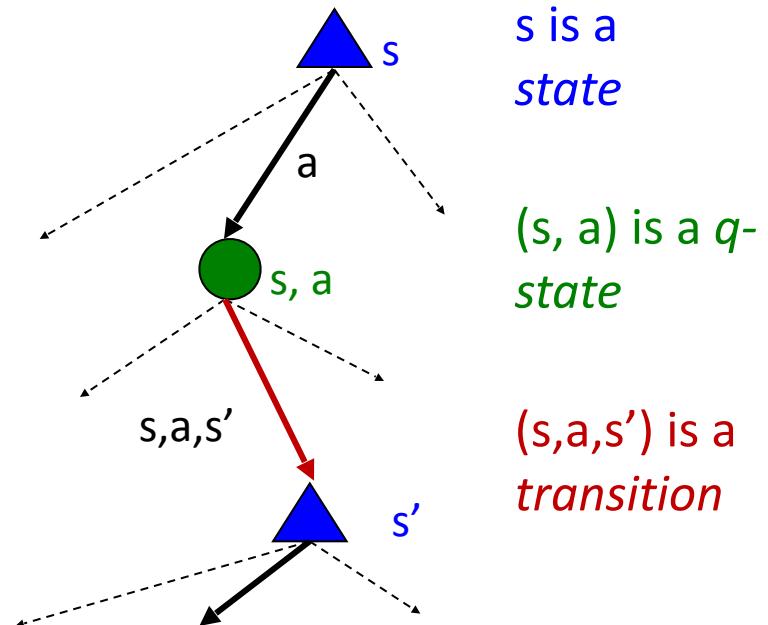
$V^*(s)$ = expected utility starting in s and acting optimally

- The value (utility) of a q-state (s,a) :

$Q^*(s,a)$ = expected utility starting out having taken action a from state s and (thereafter) acting optimally

- The optimal policy:

$\pi^*(s)$ = optimal action from state s



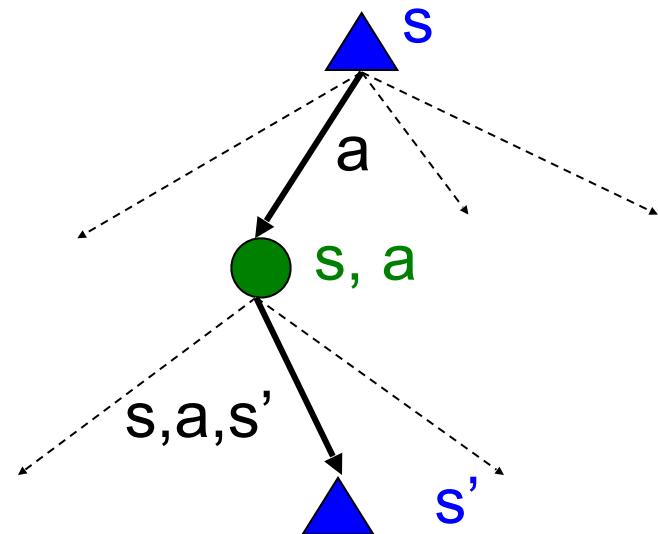
Values of States

- Fundamental operation: compute the (expectimax) value of a state
 - Expected utility under optimal action
 - Average sum of (discounted) rewards
 - This is just what expectimax computed!
- Recursive definition of value:

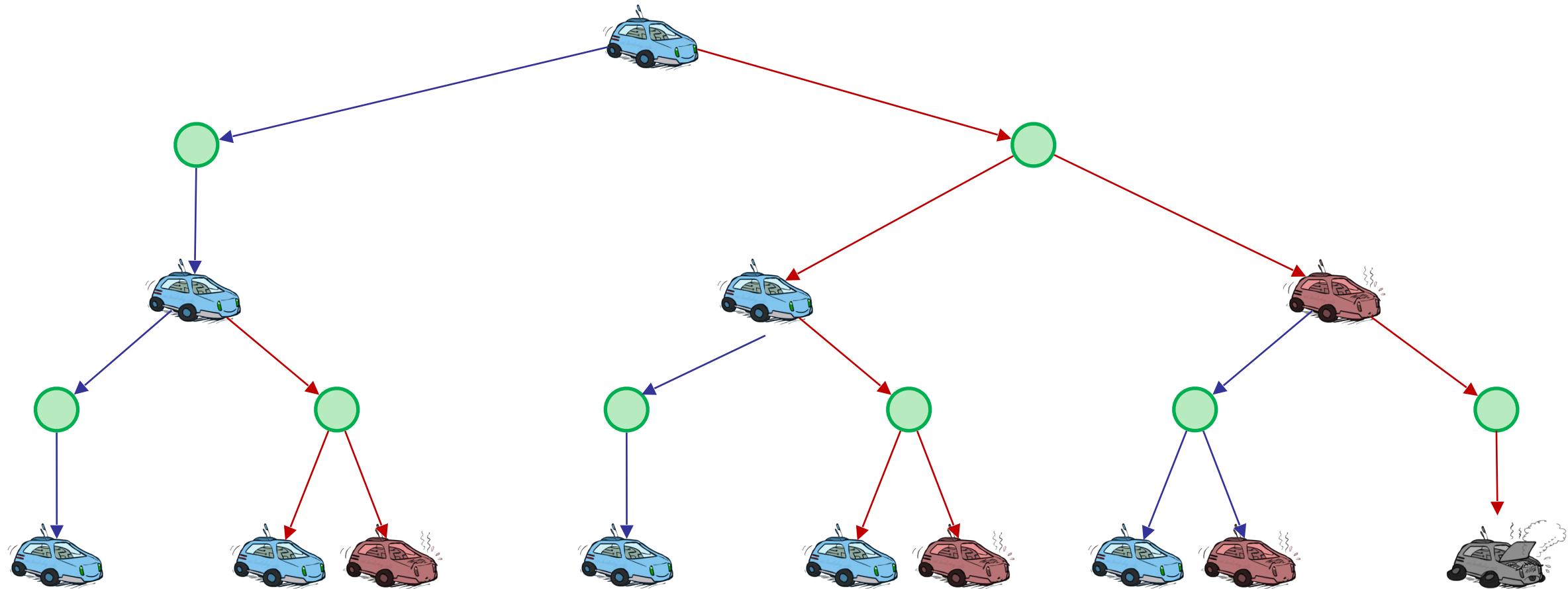
$$V^*(s) = \max_a Q^*(s, a)$$

$$Q^*(s, a) = \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^*(s')]$$

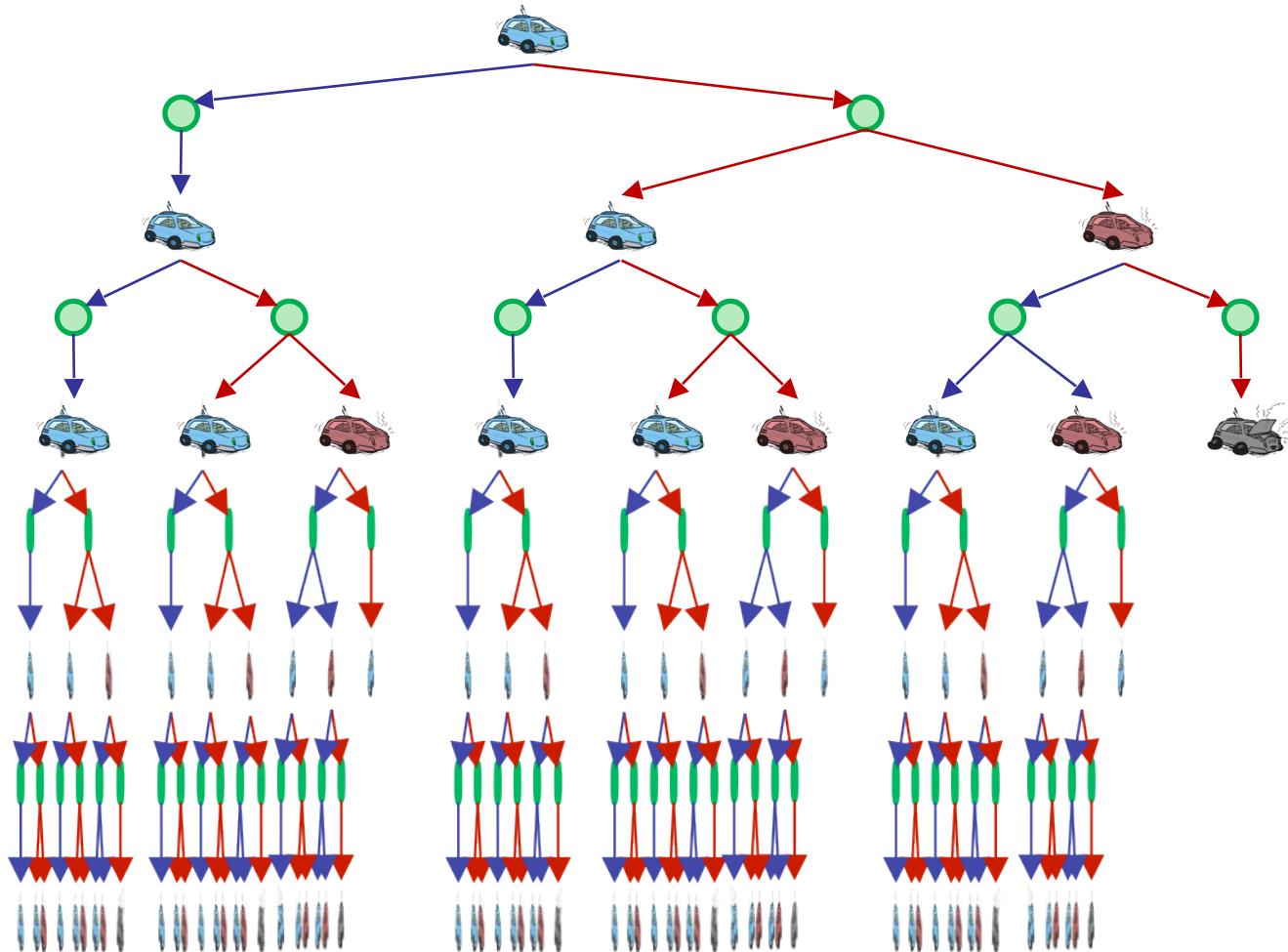
$$V^*(s) = \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^*(s')]$$



Racing Search Tree

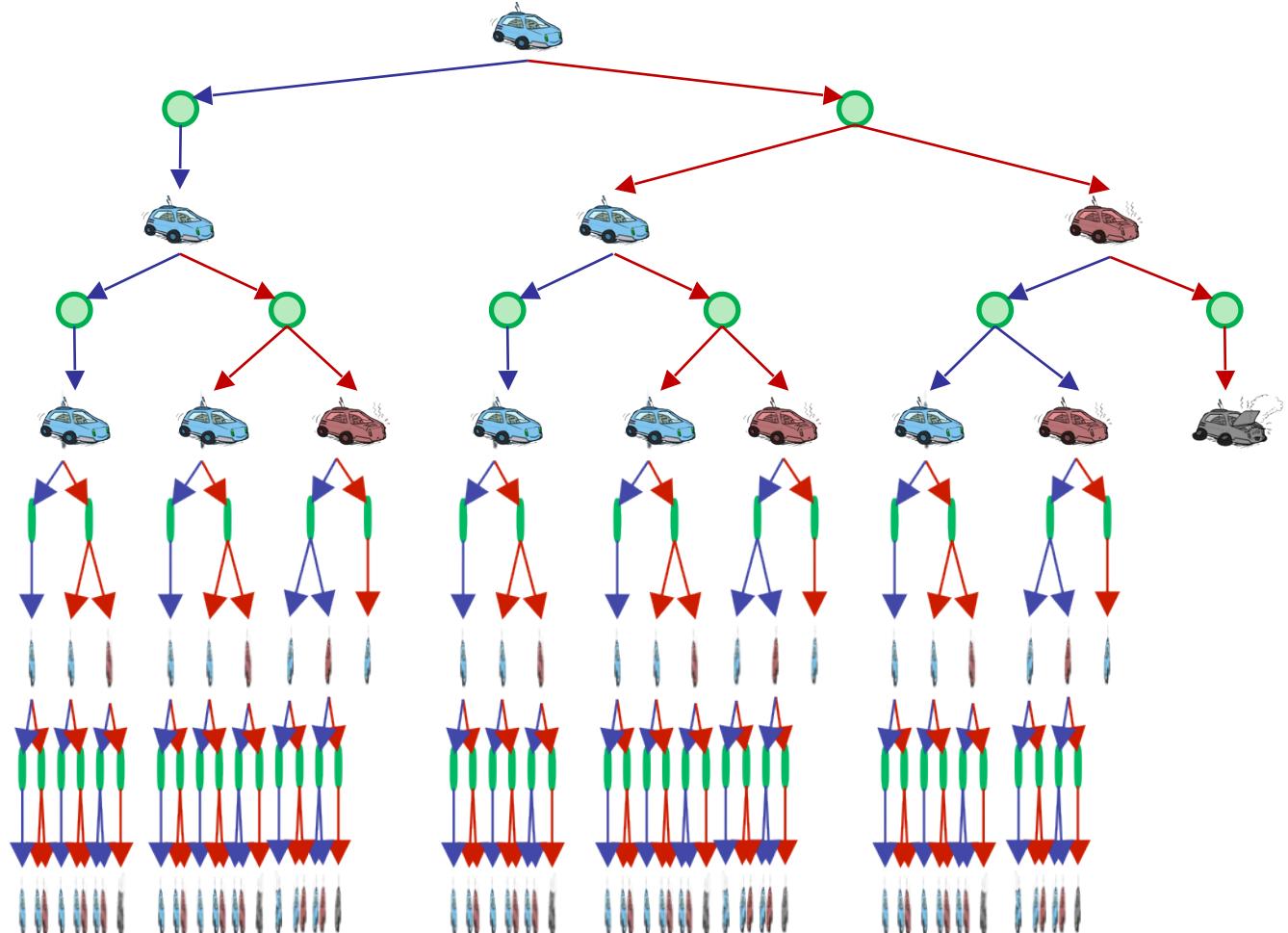


Racing Search Tree



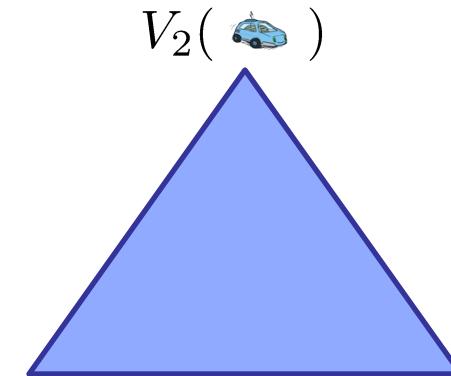
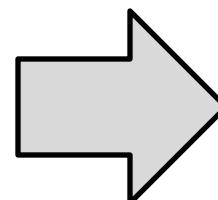
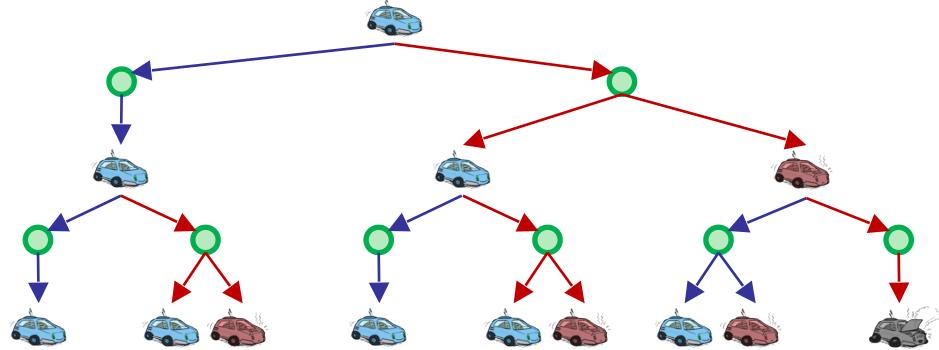
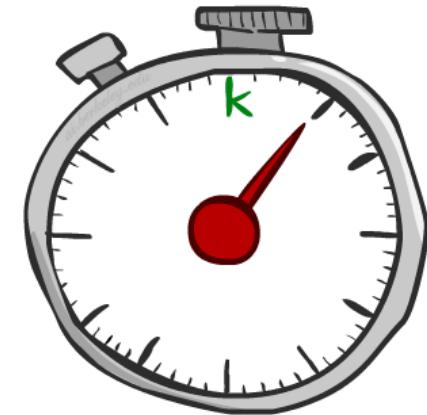
Racing Search Tree

- We're doing way too much work with expectimax!
- Problem: States are repeated
 - Idea: Only compute needed quantities once
- Problem: Tree goes on forever
 - Idea: Do a depth-limited computation, but with increasing depths until change is small
 - Note: deep parts of the tree eventually don't matter if $\gamma < 1$

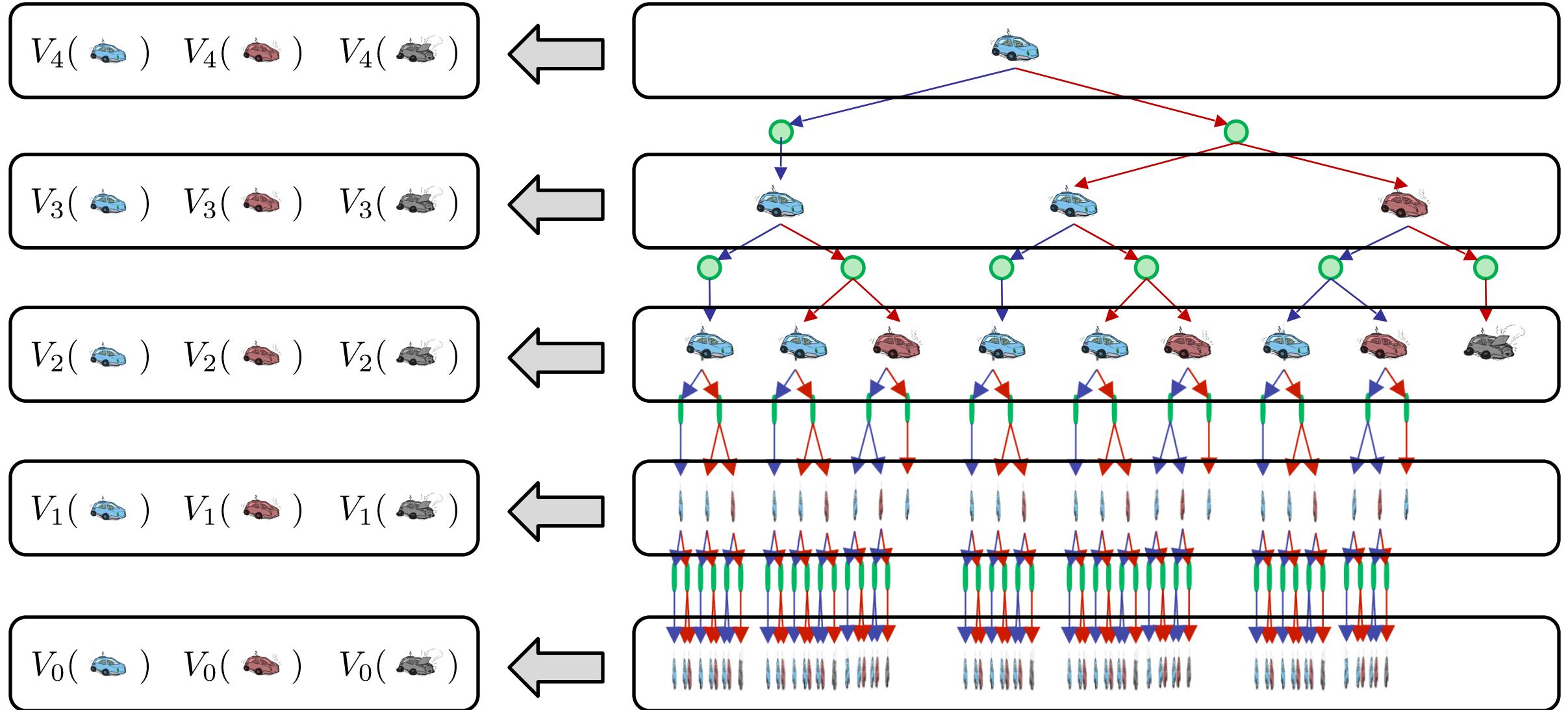


Time-Limited Values

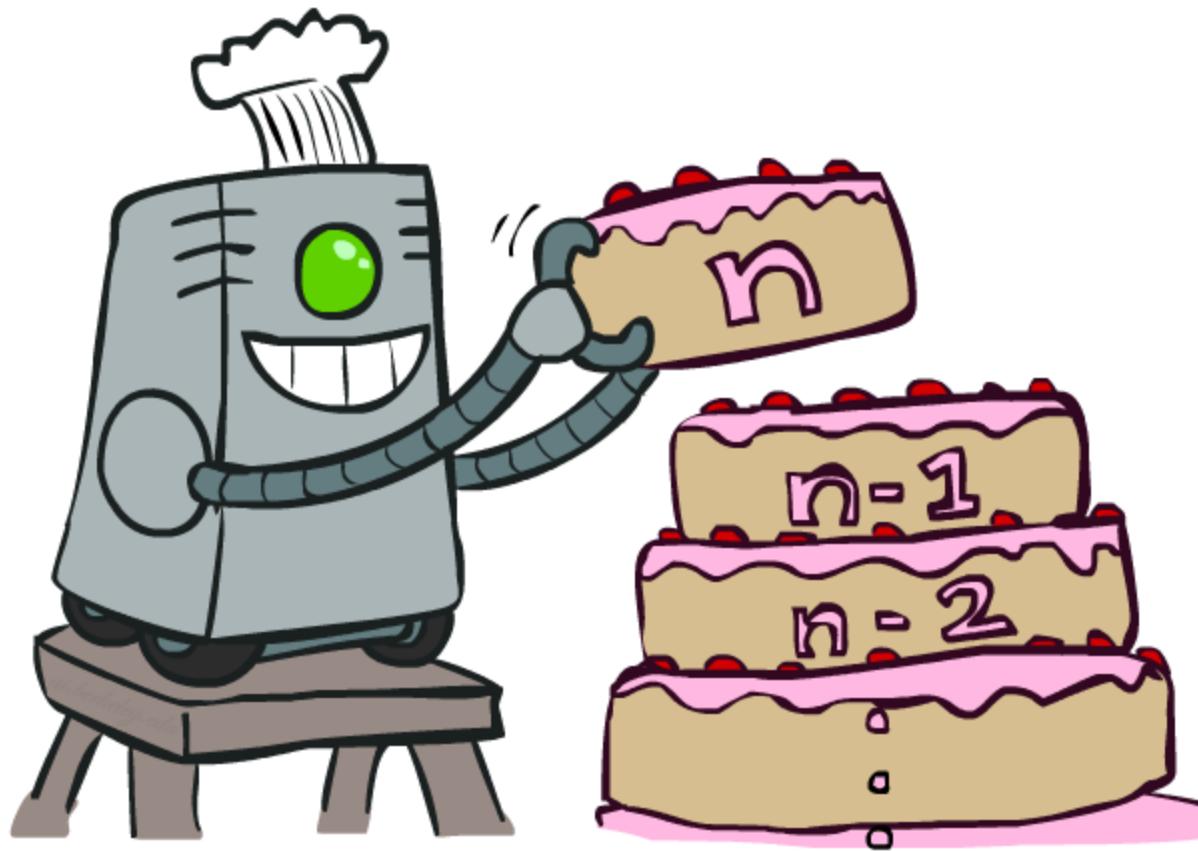
- Key idea: time-limited values
- Define $V_k(s)$ to be the optimal value of s if the game ends in k more time steps
 - Equivalently, it's what a depth- k expectimax would give from s



Computing Time-Limited Values



Value Iteration

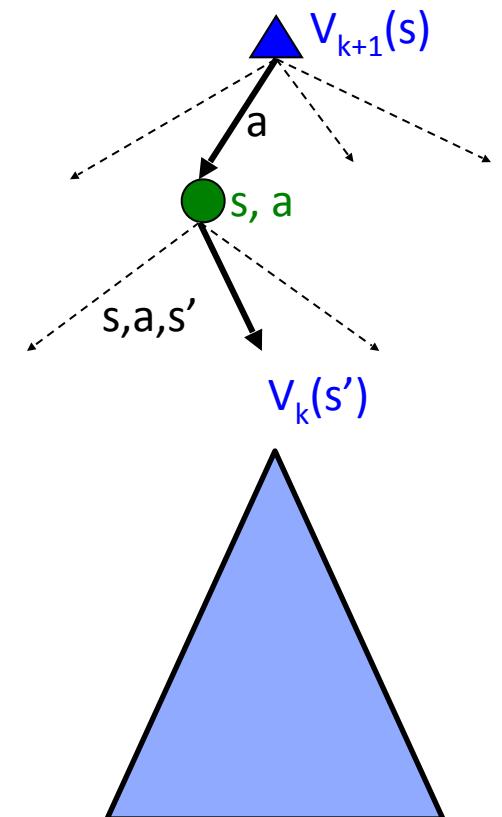


Value Iteration

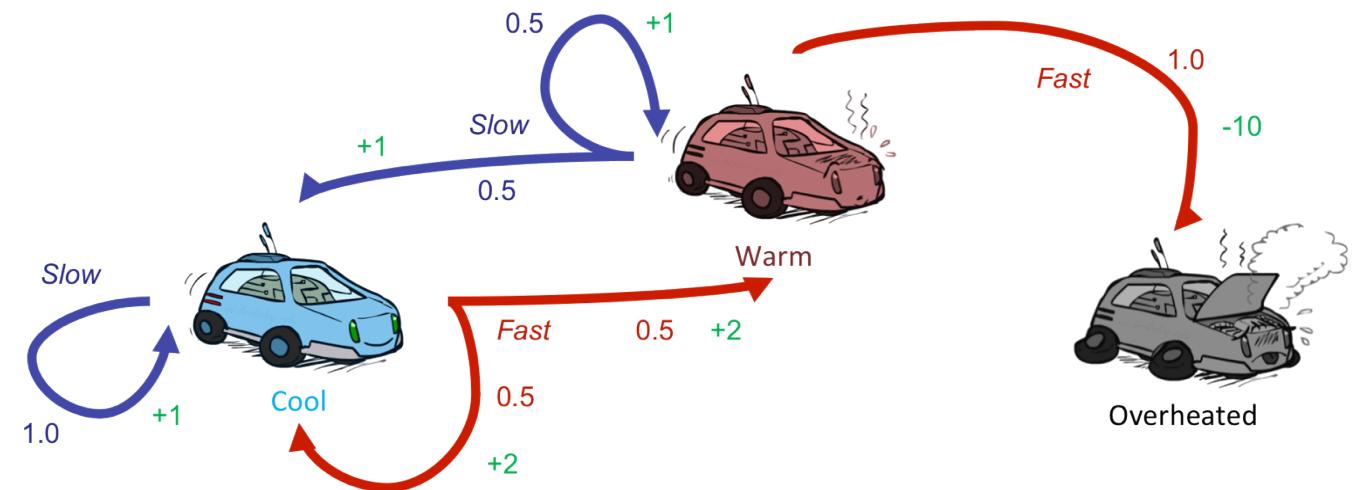
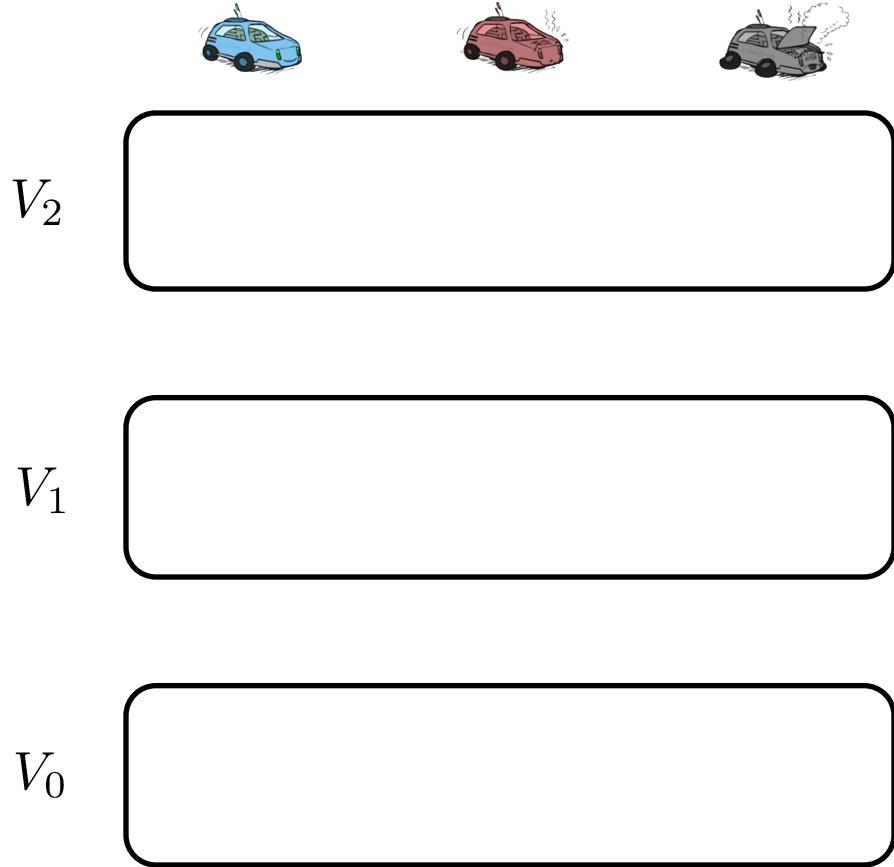
- Start with $V_0(s) = 0$: no time steps left means an expected reward sum of zero
- Given vector of $V_k(s)$ values, do one ply of expectimax from each state:

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V_k(s')]$$

- Repeat until convergence
- Complexity of each iteration: $O(S^2A)$
- Theorem: will converge to unique optimal values**
 - Basic idea: approximations get refined towards optimal values
 - Policy may converge long before values do



Example: Value Iteration

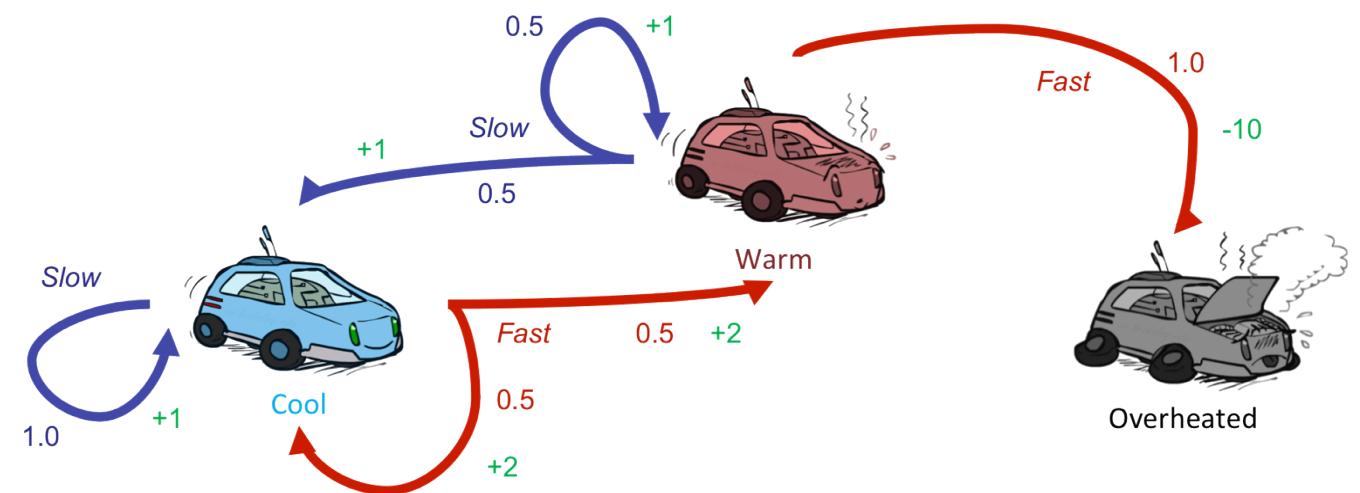


Assume no discount!

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V_k(s')]$$

Example: Value Iteration

V_2	3.5	2.5	0
V_1	2	1	0
V_0	0	0	0

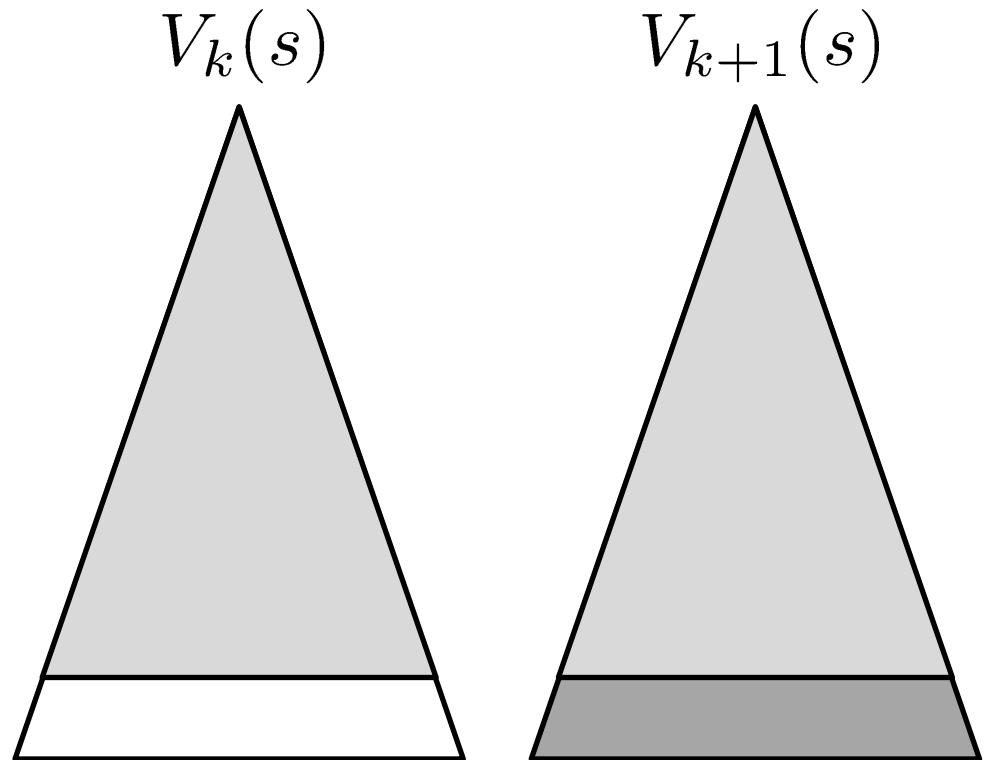


Assume no discount!

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V_k(s')]$$

Convergence*

- How do we know the V_k vectors are going to converge?
- Case 1: If the tree has maximum depth M , then V_M holds the actual untruncated values
- Case 2: If the discount is less than 1
 - Sketch: For any state V_k and V_{k+1} can be viewed as depth $k+1$ expectimax results in nearly identical search trees
 - The difference is that on the bottom layer, V_{k+1} has actual rewards while V_k has zeros
 - That last layer is at best all R_{MAX}
 - It is at worst R_{MIN}
 - But everything is discounted by γ^k that far out
 - So V_k and V_{k+1} are at most $\gamma^k \max|R|$ different
 - So as k increases, the values converge



Next Time: Policy-Based Methods
