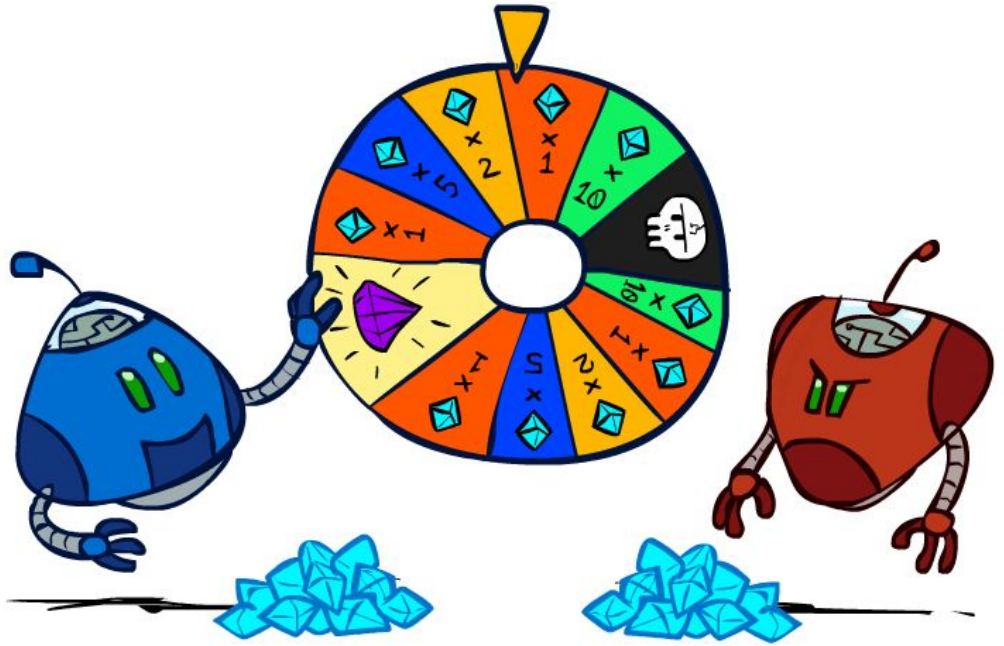
CS 106: Artificial Intelligence

Expectimax, Uncertainty, and Utilities



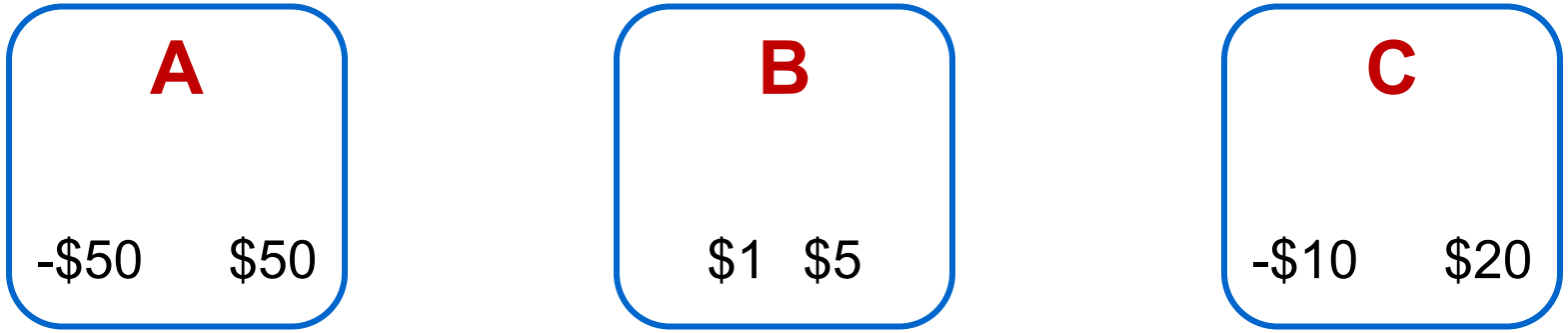
Instructor: Ngoc-Hoang LUONG, PhD.

University of Information Technology (UIT), VNU-HCM

[These slides were adapted from the slides created by Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley. http://ai.berkeley.edu.]

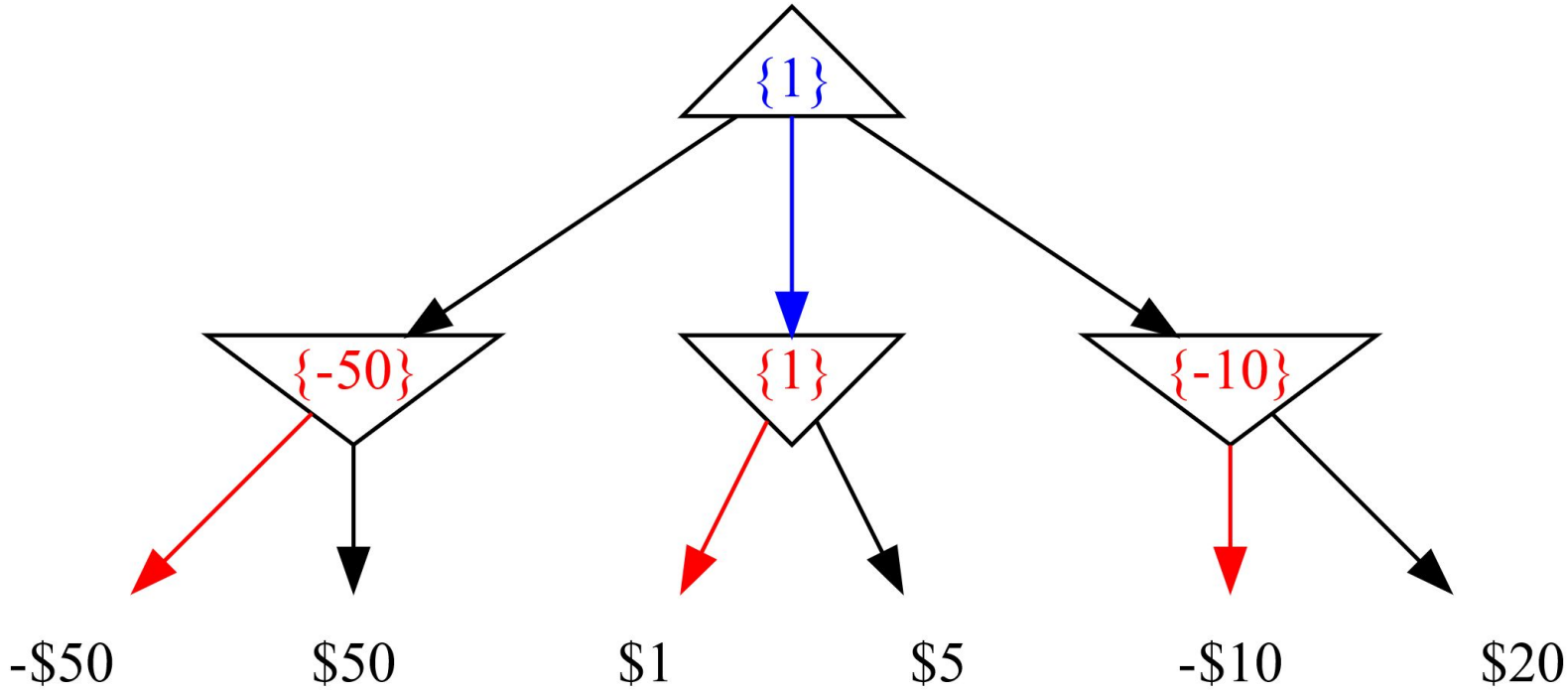
A simple game

- You choose one of the three bins.
- I choose a number from that bin.
- Your goal is to maximize the chosen number.

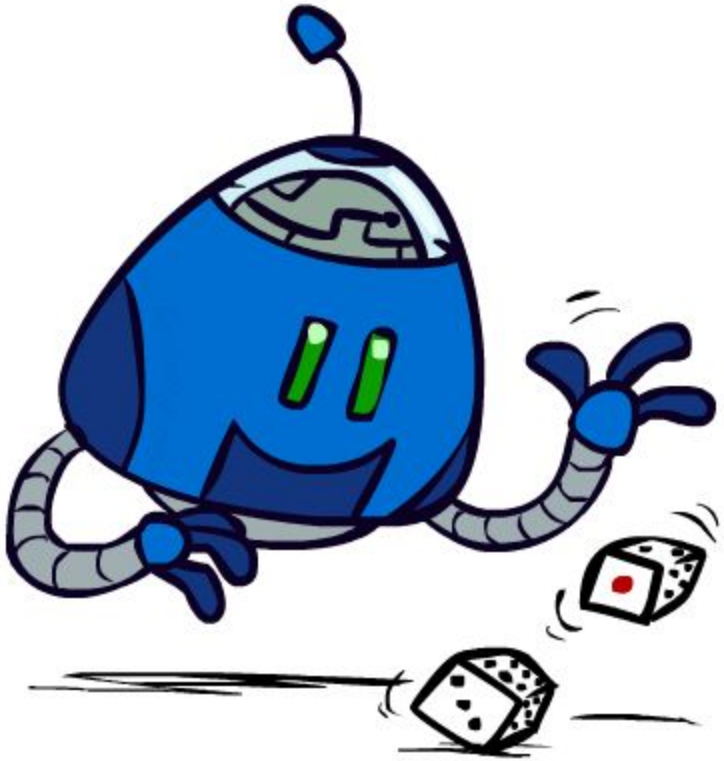


Game tree

- Each node is a decision point for a player.
- Each root-to-leaf path is a possible outcome of the game.
- Your goal is to maximize the chosen number.

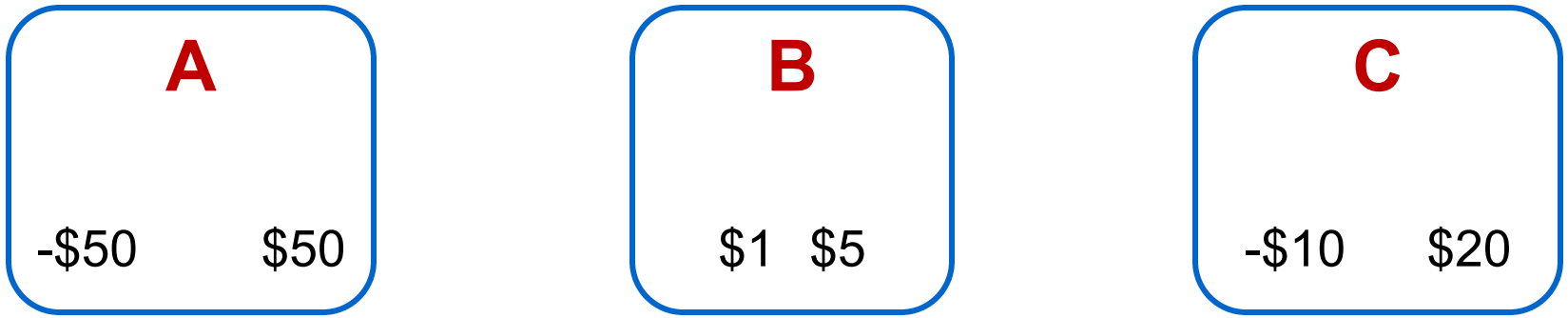


Uncertain Outcomes



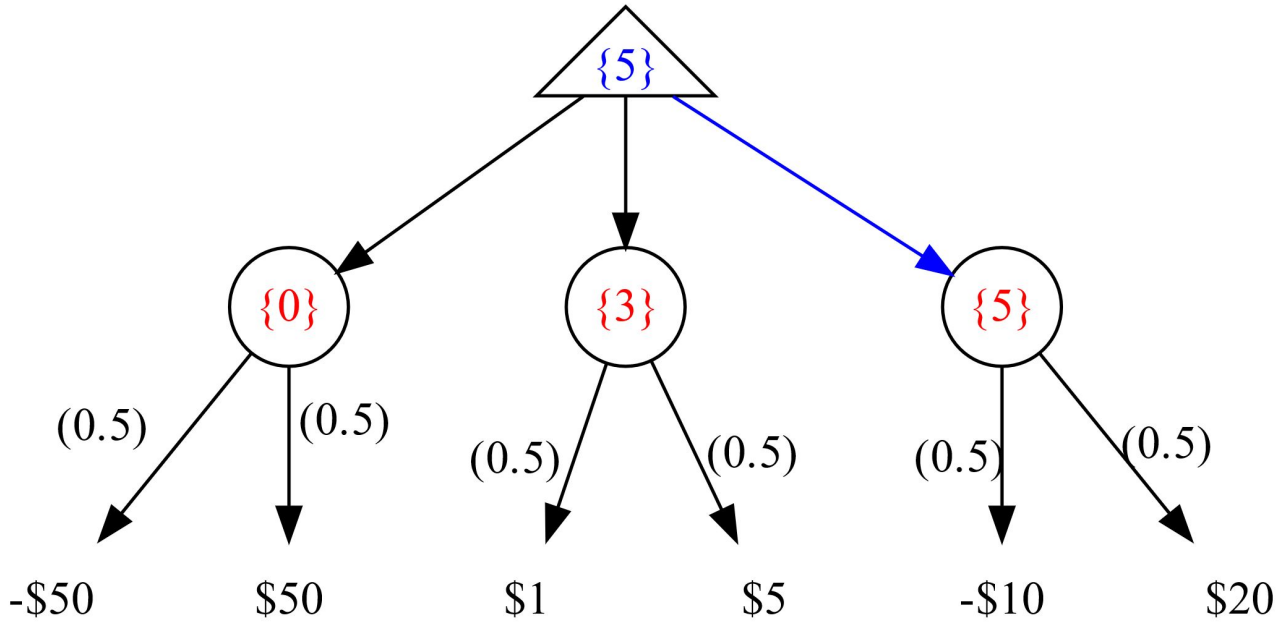
Uncertain Outcomes

- You choose one of the three bins.
- I toss a fair coin. If head, the bigger number is returned. If tail, the smaller number is returned.
- Your goal is to maximize the chosen number.

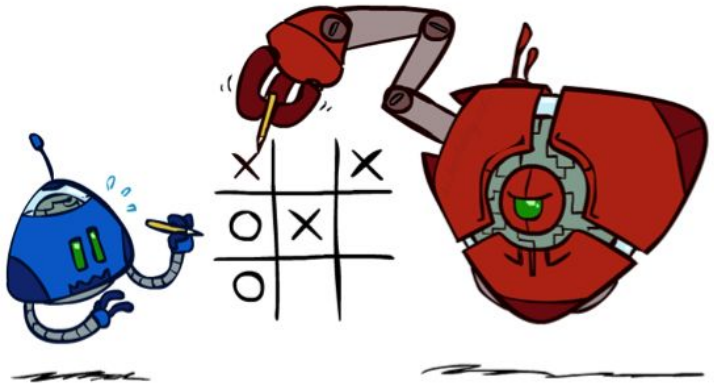


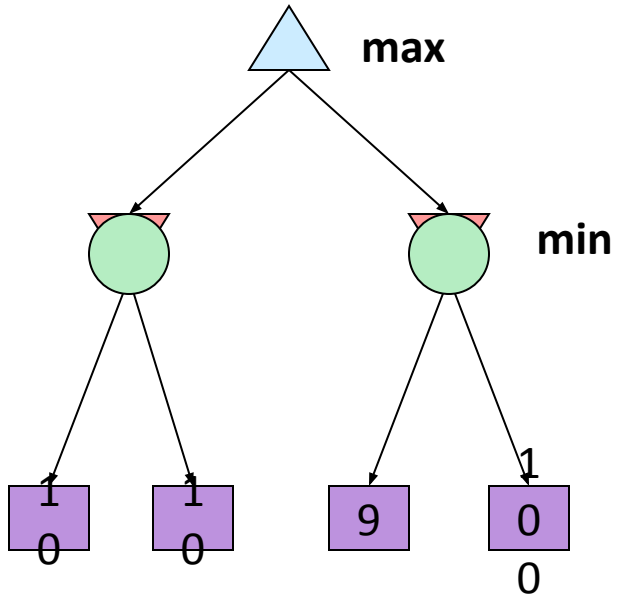
Uncertain Outcomes

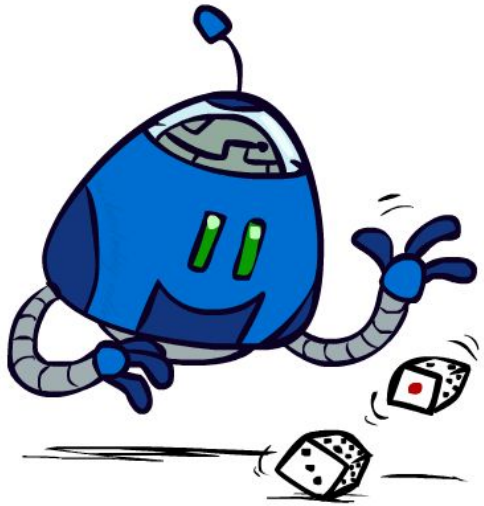
- You choose one of the three bins.
- I toss a fair coin. If head, the bigger number is returned. If tail, the smaller number is returned.
- Your goal is to maximize the chosen number.



Worst-Case vs. Average Case



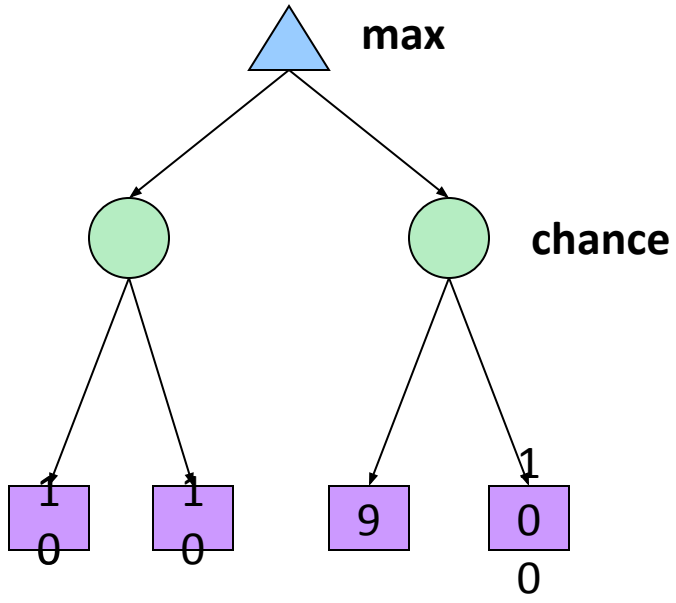




Idea: Uncertain outcomes controlled by chance, not an adversary!

Expectimax Search

- Why wouldn't we know what the result of an action will be?
  - Explicit randomness: rolling dice
  - Unpredictable opponents: the ghosts respond randomly
  - 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



[Demo: min vs exp (L7D1,2)]

Video of Demo Minimax vs Expectimax (Min)

Video of Demo Minimax vs Expectimax (Exp)

Expectimax Pseudocode

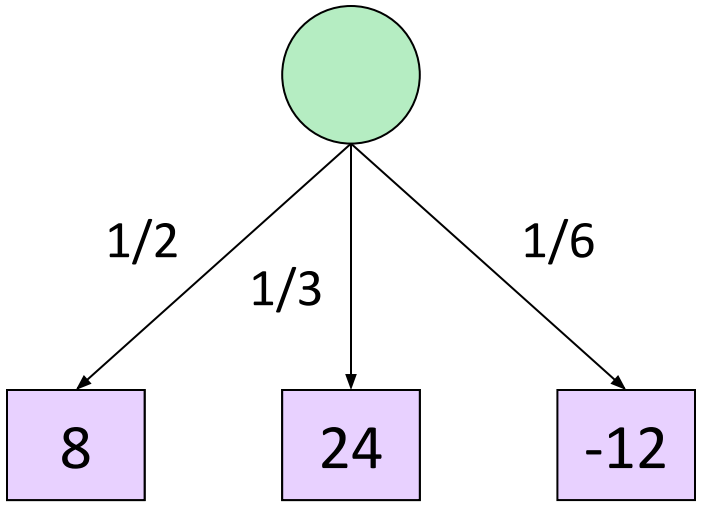
```
def value(state):
    if the state is a terminal state: return the state's utility
    if the next agent is MAX: return max-value(state)
    if the next agent is EXP: return exp-value(state)
```

```
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
```

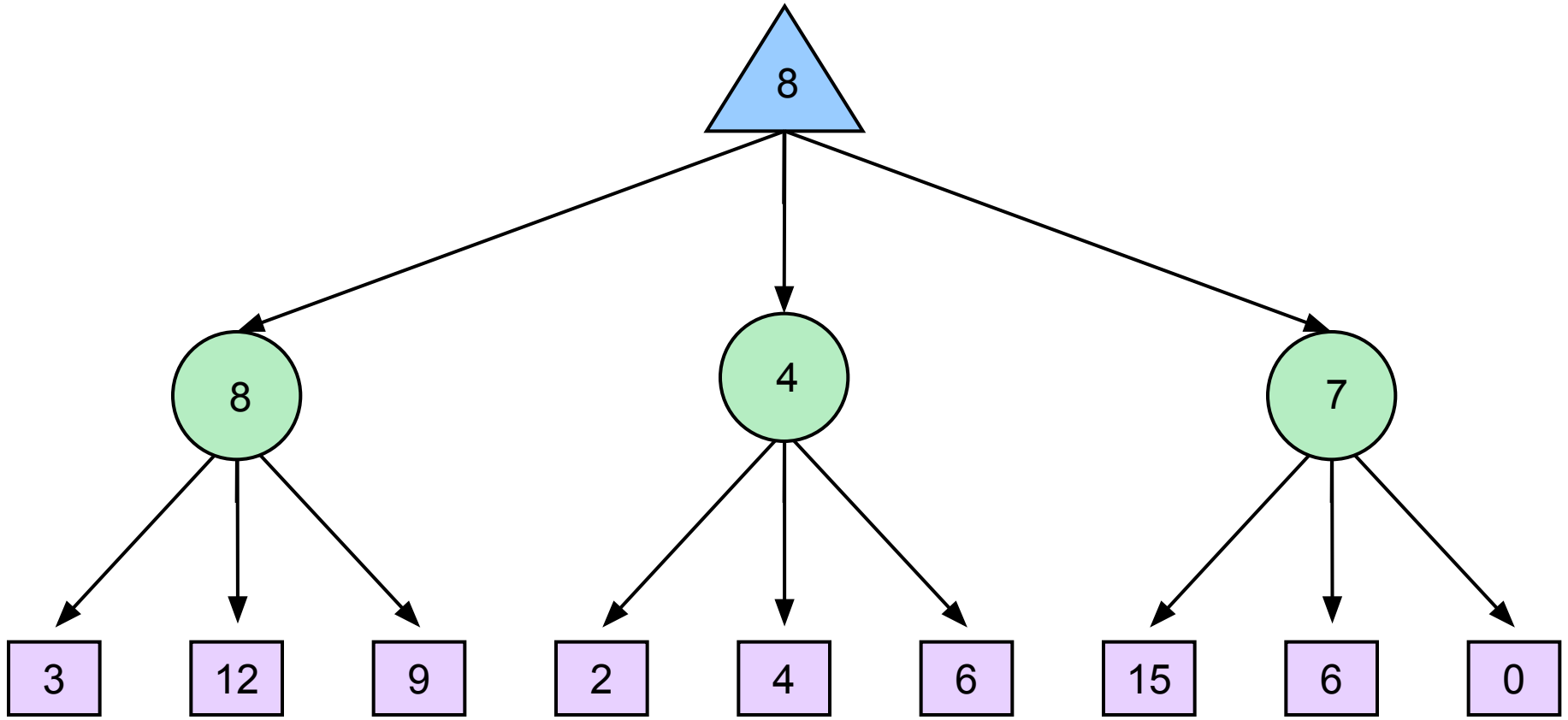
Expectimax Pseudocode

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

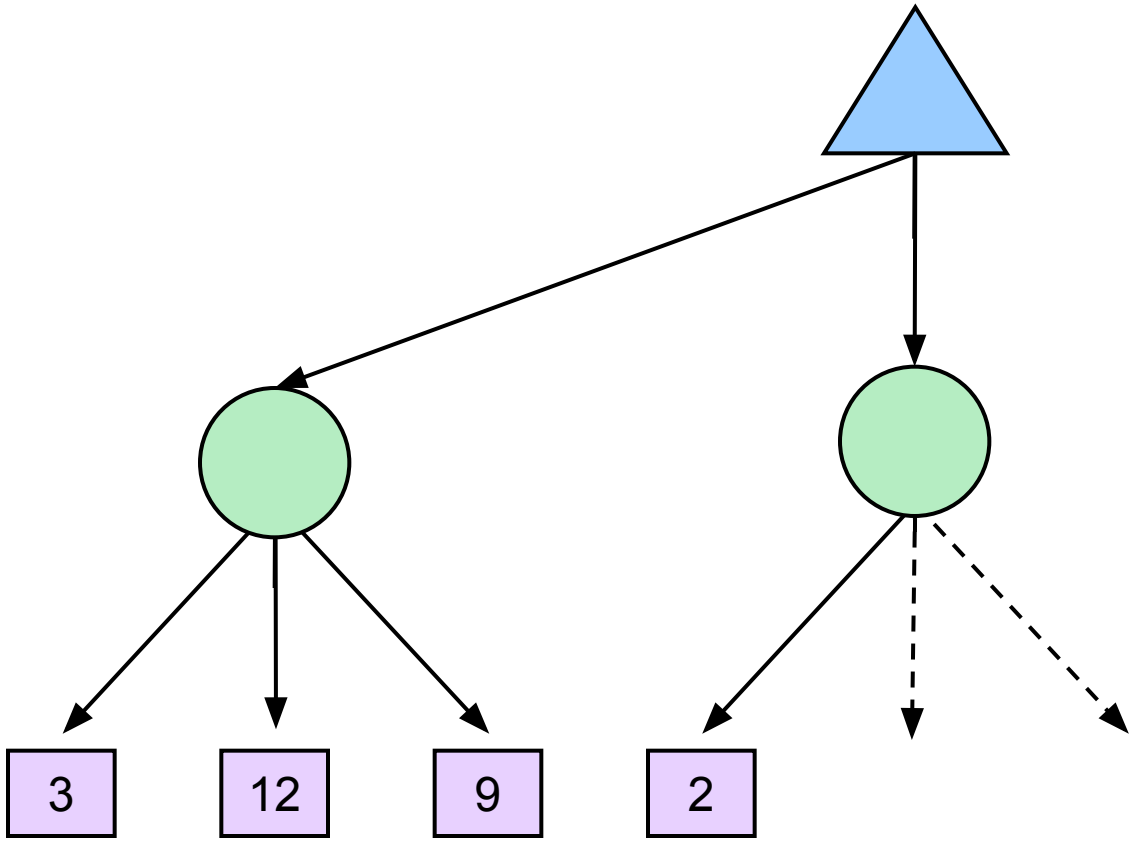


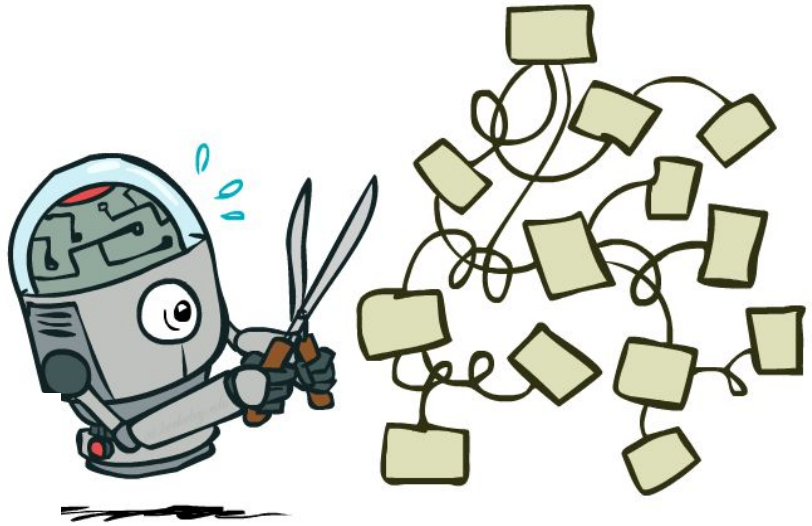
$$v = (1/2)\ (8) + (1/3)\ (24) + (1/6)\ (-12) = 10$$

Expectimax Example

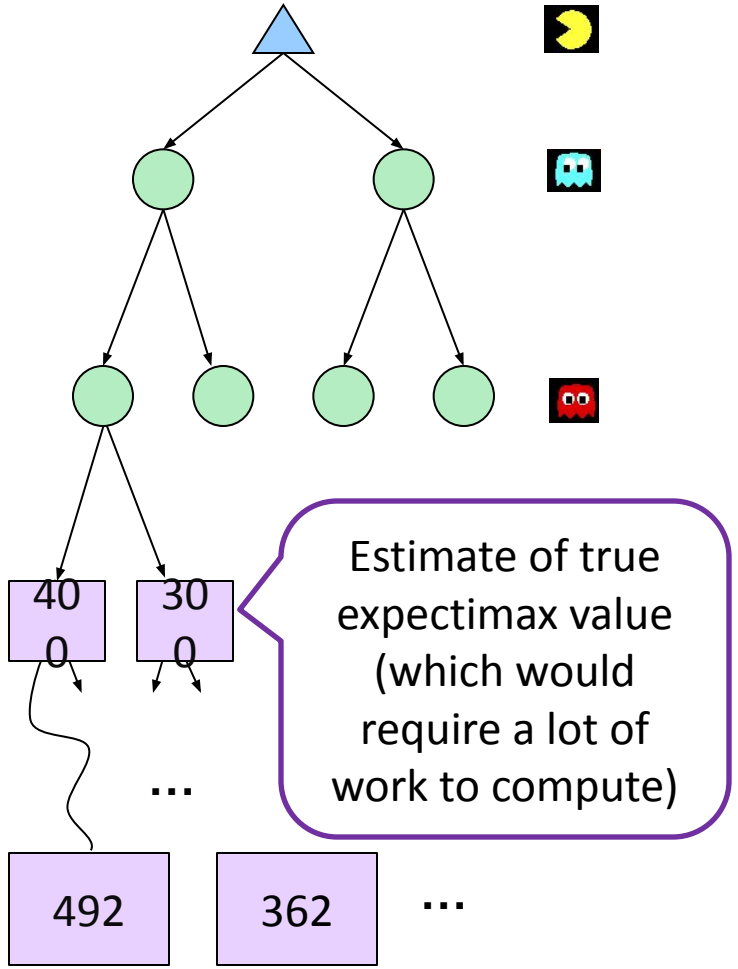


Expectimax Pruning?

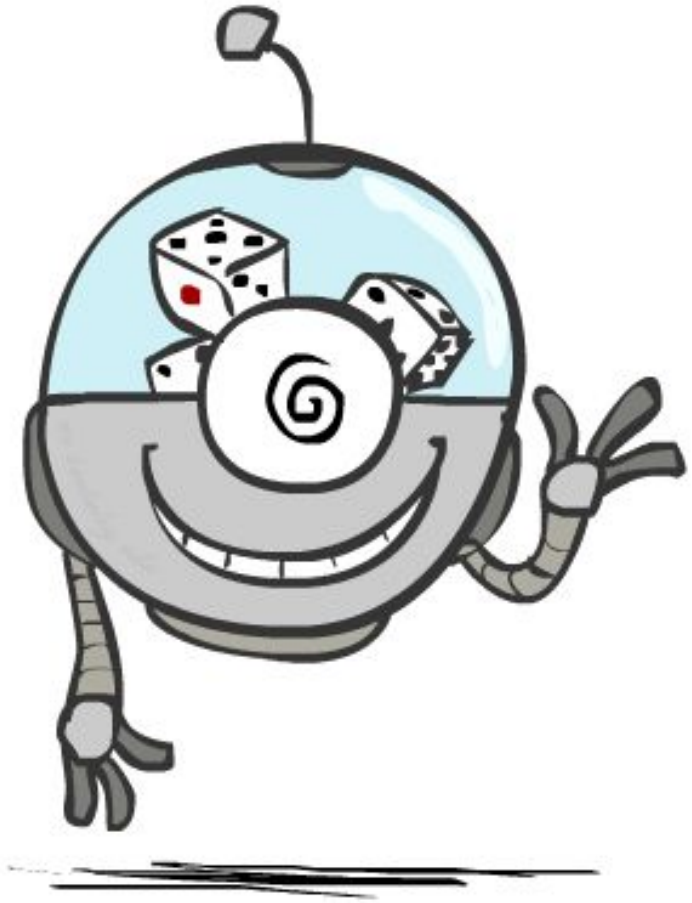




Depth-Limited Expectimax



Probabilities



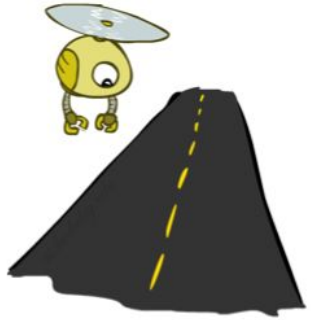
Reminder: Probabilities

- A random variable represents an event whose outcome is unknown
- A probability distribution is an assignment of weights to outcomes

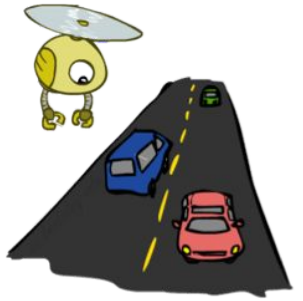
- Example: Traffic on freeway
  - Random variable: T = whether there's traffic
  - Outcomes: T in {none, light, heavy}
  - Distribution: P(T=none) = 0.25, P(T=light) = 0.50, P(T=heavy) = 0.25

- 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=heavy) = 0.25, P(T=heavy | Hour=8am) = 0.60
  - We'll talk about methods for reasoning and updating probabilities later



0.25

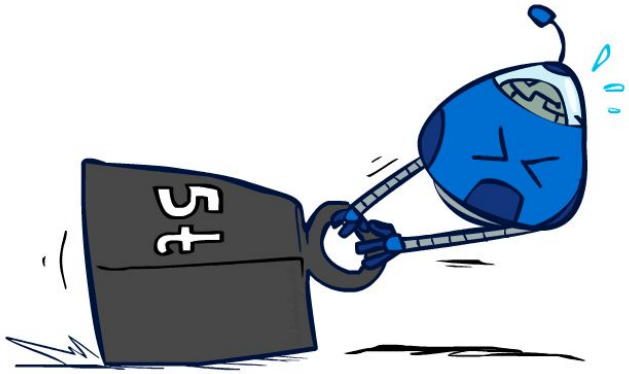


0.50

0.25

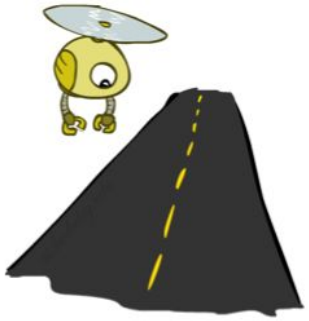
Reminder: Expectations

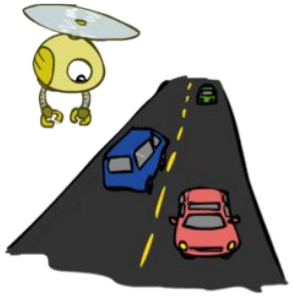
- The expected value of a function of a random variable is the average, weighted by the probability distribution over outcomes



- Example: How long to get to the airport?

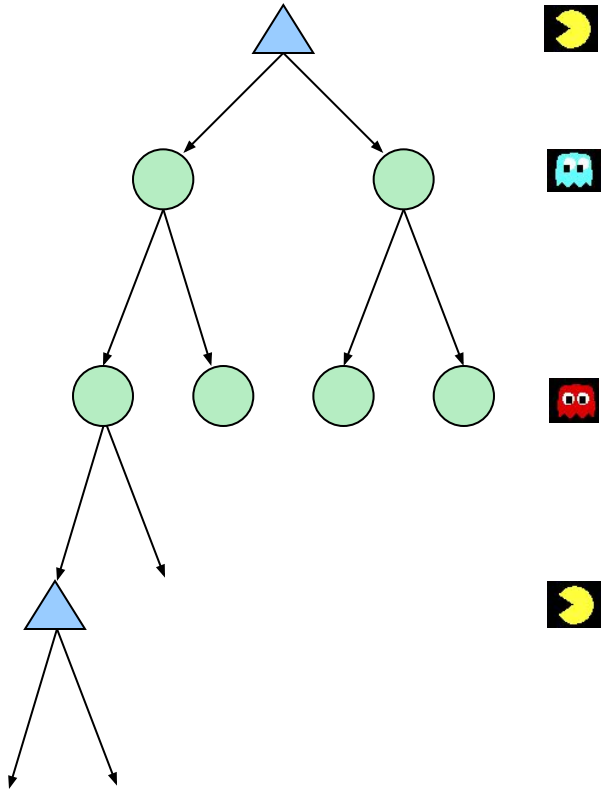
Time:	20 min		30 min		60 min		
	x	+	x	+	x		
Probability:	0.25		0.50		0.25		35 min





What Probabilities to Use?

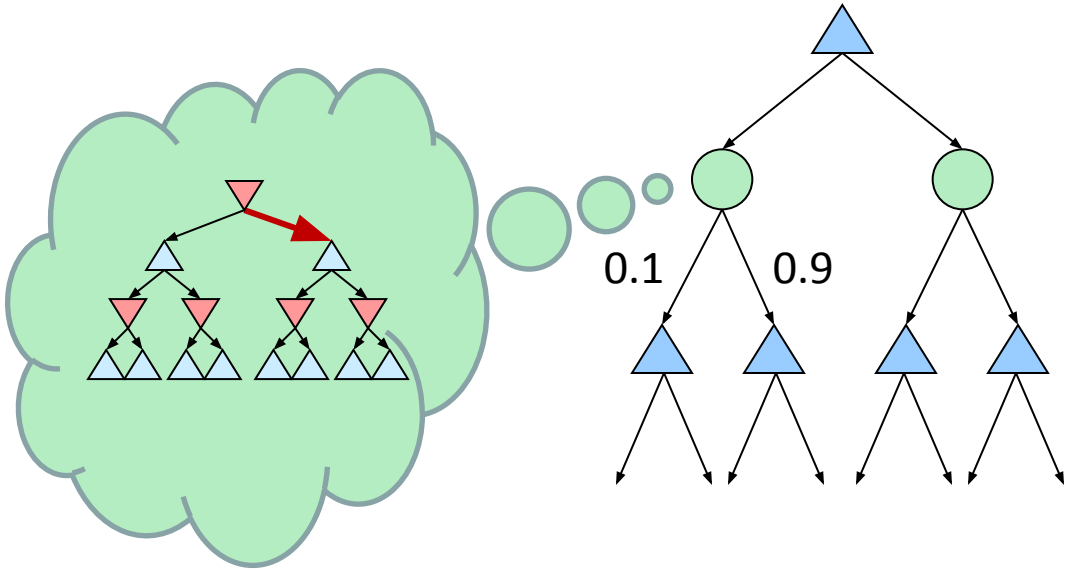
- In expectimax search, we have a probabilistic model of how the opponent (or environment) will behave in any state
  - Model could be a simple uniform distribution (roll a die)
  - Model could be sophisticated and require a great deal of computation
  - We have a chance node for any outcome out of our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



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

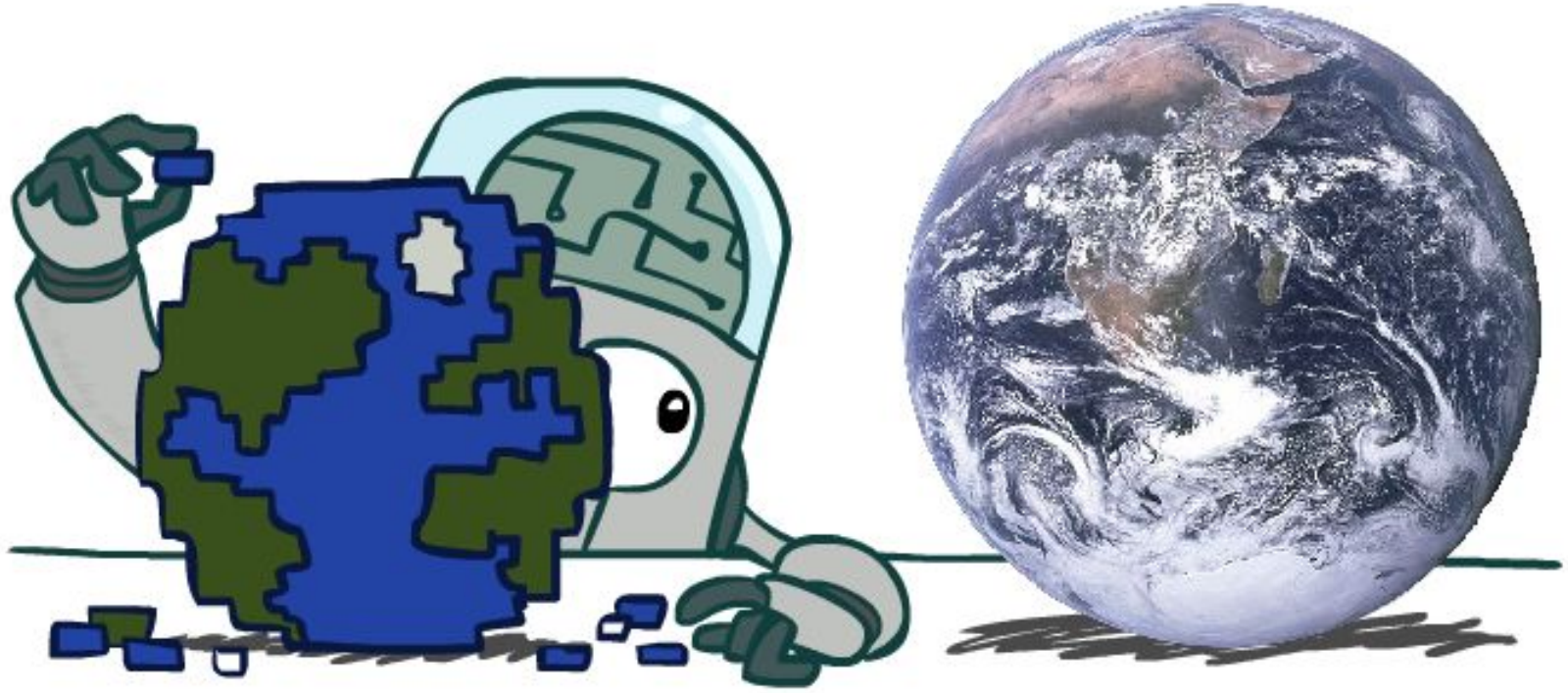
Quiz: Informed Probabilities

- 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

Modeling Assumptions



The Dangers of Optimism and Pessimism

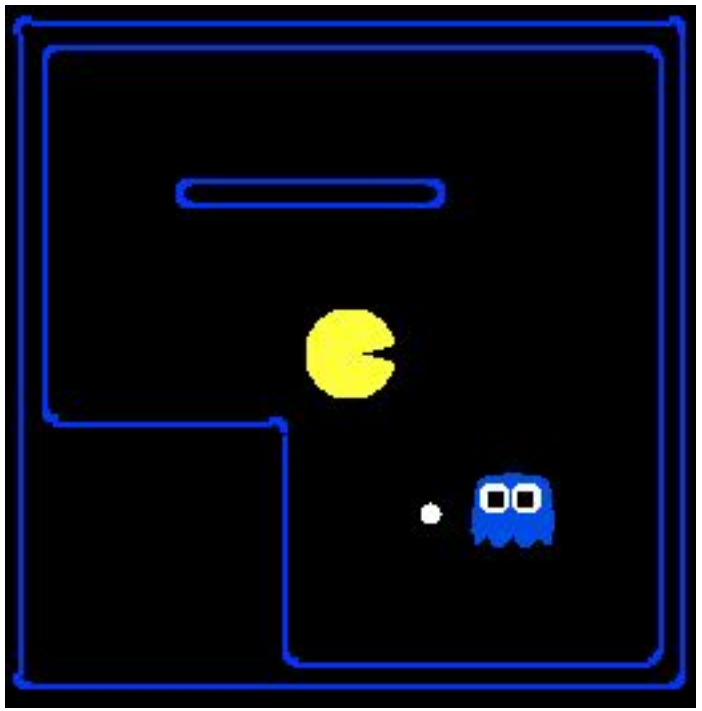
Dangerous Optimism

Assuming chance when the world is adversarial

Dangerous Pessimism

Assuming the worst case when it's not likely

Assumptions vs. Reality



	Adversarial Ghost	Random Ghost
Minimax Pacman	Won 5/5 Avg. Score: 483	Won 5/5 Avg. Score: 493
Expectimax Pacman	Won 1/5 Avg. Score: -303	Won 5/5 Avg. Score: 503

Results from playing 5 games

Pacman used depth 4 search with an eval function that avoids trouble
Ghost used depth 2 search with an eval function that seeks Pacman

[Demos: world assumptions (L7D3,4,5,6)]

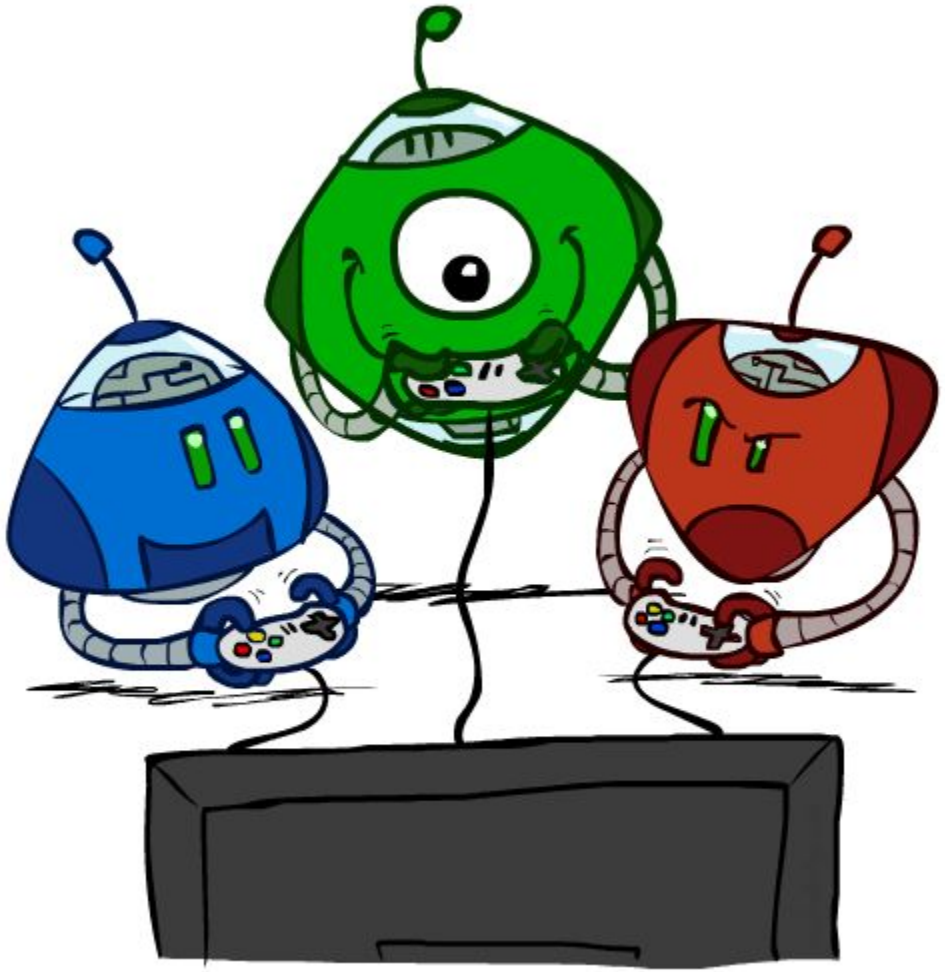
Video of Demo World Assumptions
Random Ghost – Expectimax Pacman

Video of Demo World Assumptions
Adversarial Ghost – Minimax Pacman

Video of Demo World Assumptions
Adversarial Ghost – Expectimax Pacman

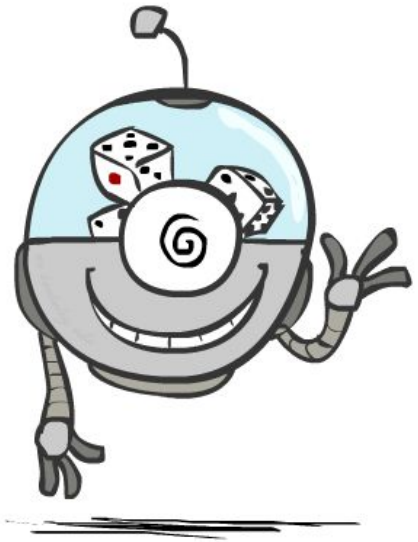
Video of Demo World Assumptions
Random Ghost – Minimax Pacman

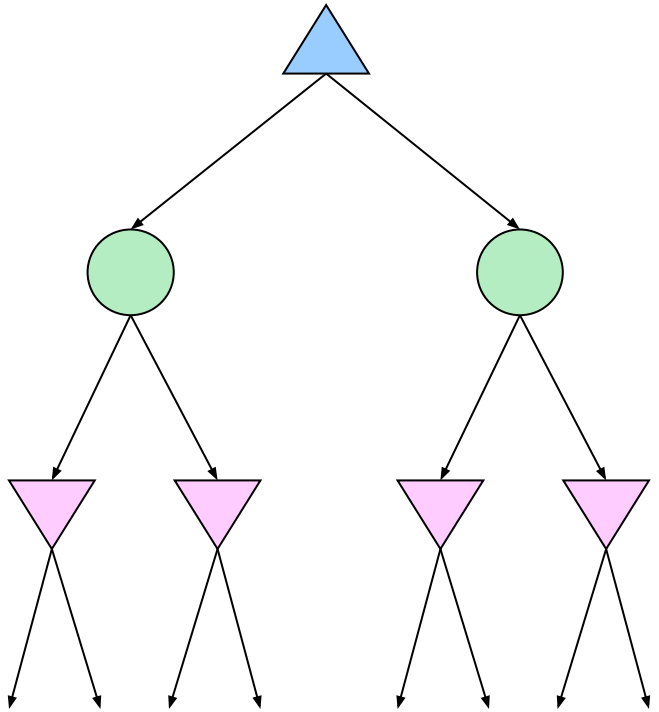
Other Game Types



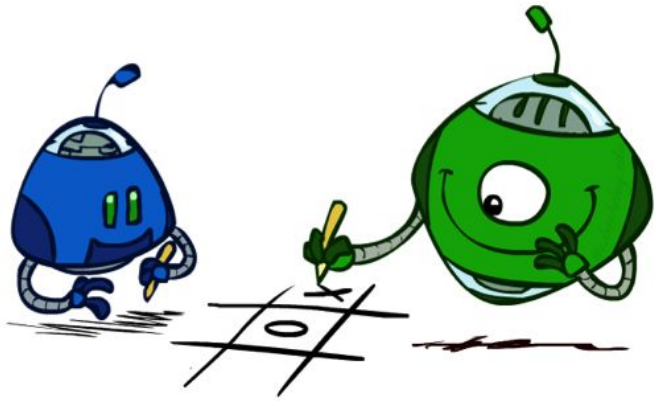
Mixed Layer Types

- 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

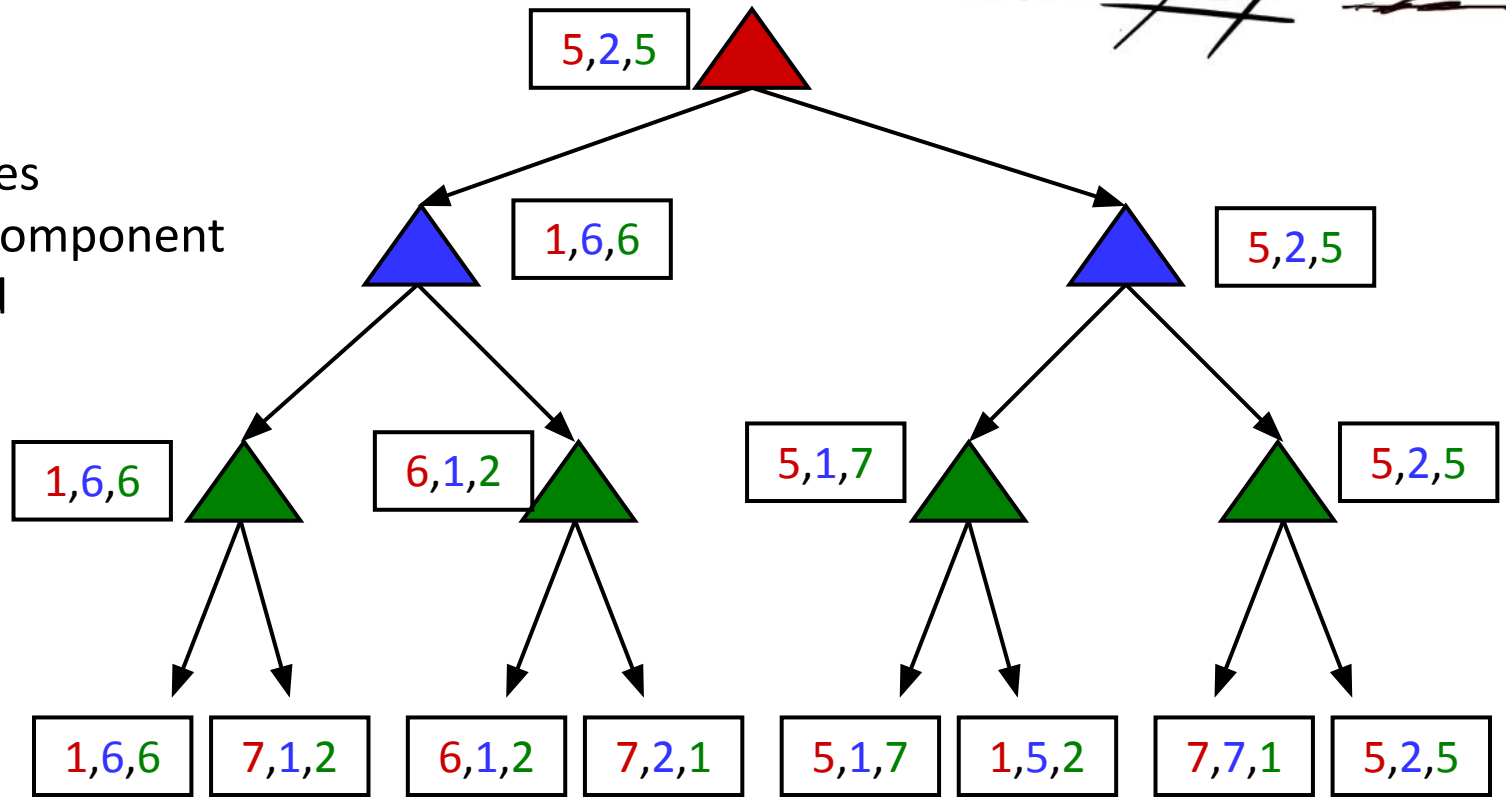


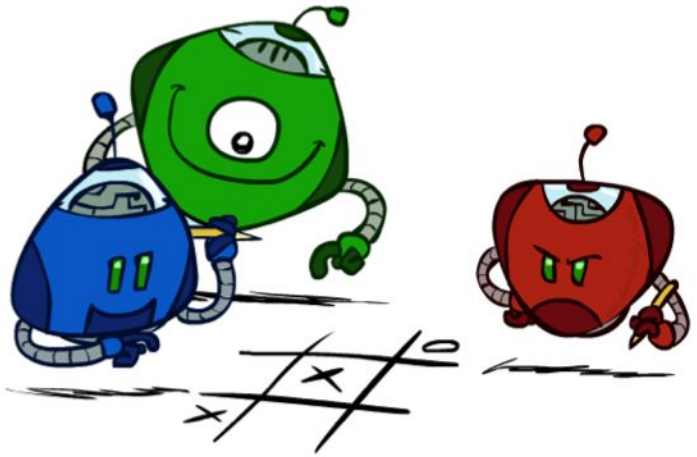


Multi-Agent Utilities

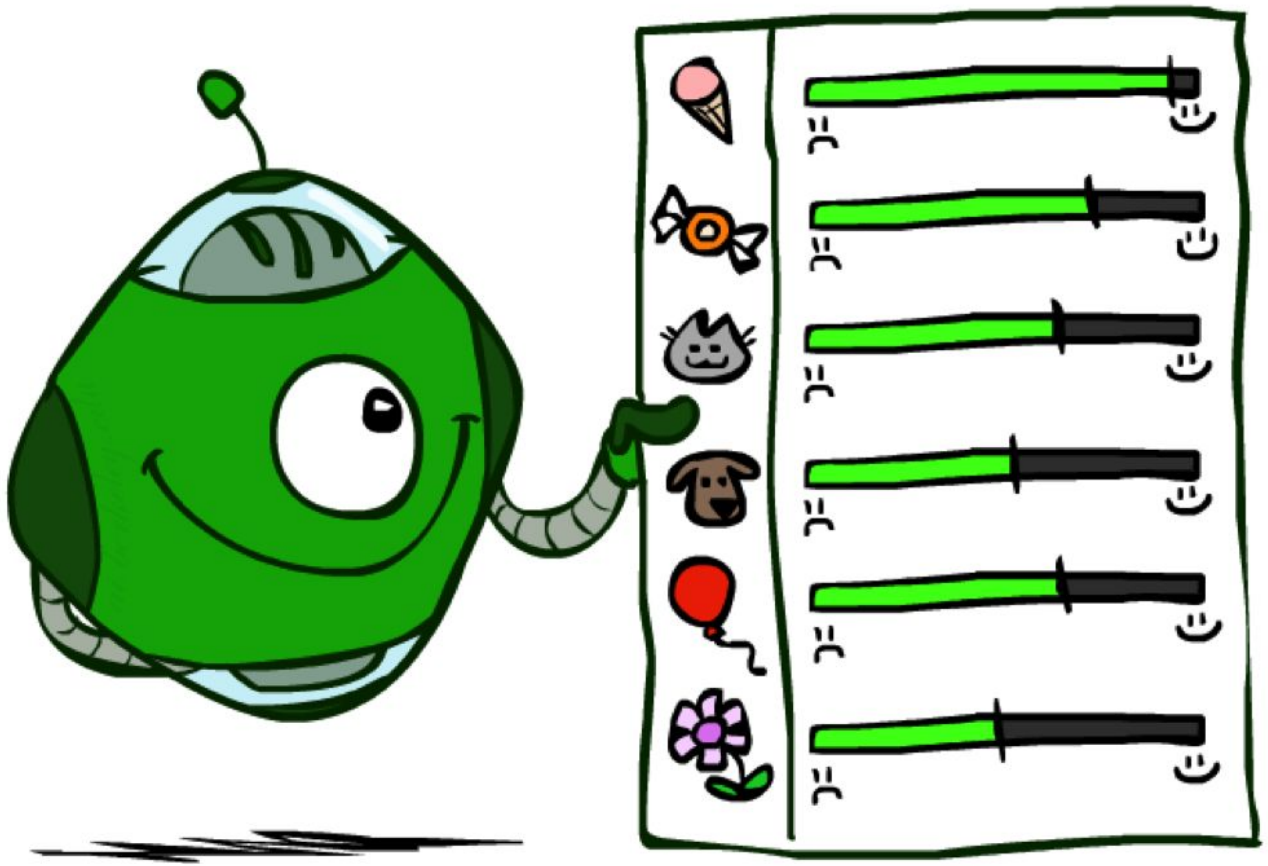


- What if the game is not zero-sum, or has multiple players?
- Generalization of minimax:
  - Terminals have utility tuples
  - Node values are also utility tuples
  - Each player maximizes its own component
  - Can give rise to cooperation and competition dynamically…



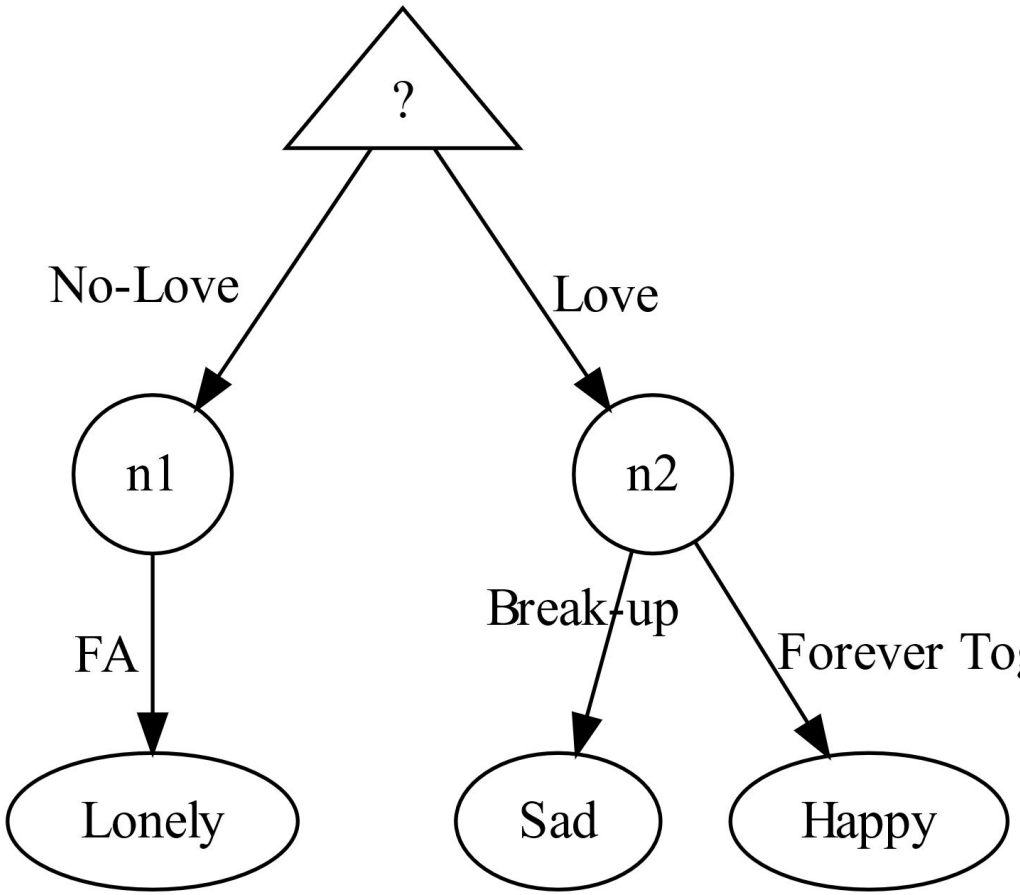


Utilities



Maximum Expected Utility

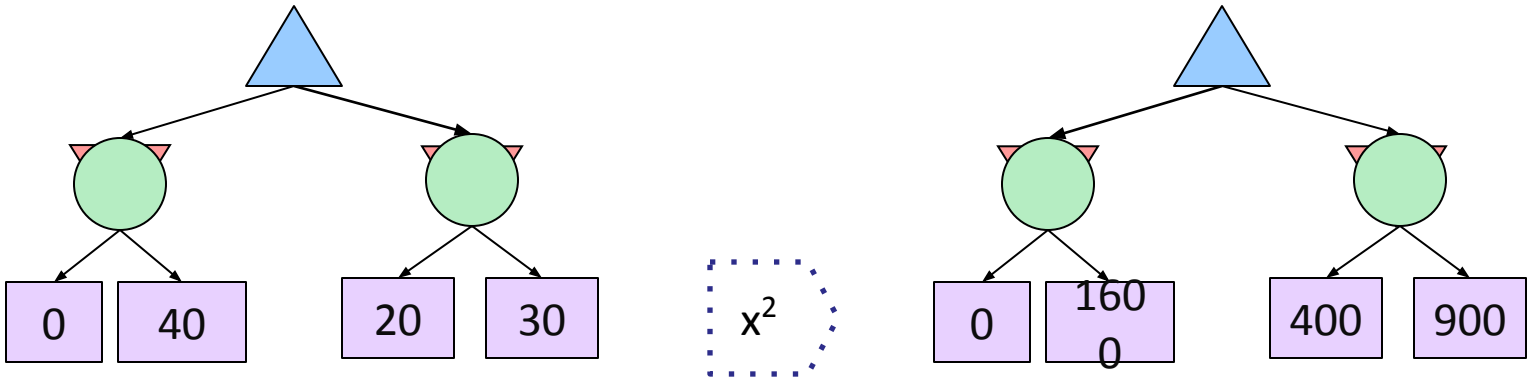
- Why should we average utilities? Why not minimax?
- Princ
  - A
    ex
- Ques
  - W
  - H
  - H
  - W



its

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

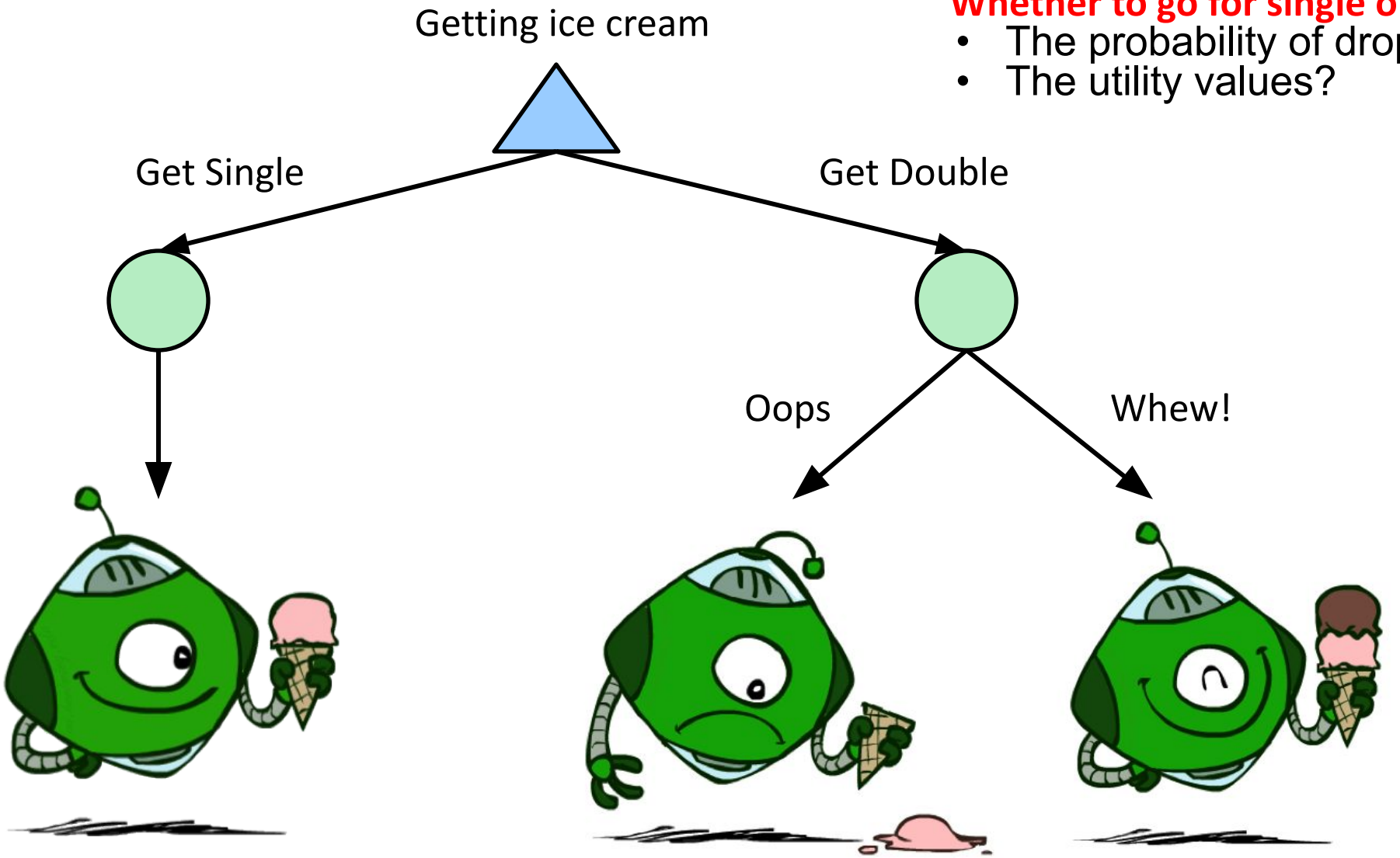
- Utilities are functions from outcomes (states of the world) to real numbers that describe an agent's preferences

- 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

Utilities: Uncertain Outcomes

Whether to go for single or double?

- The probability of dropping?
- The utility values?



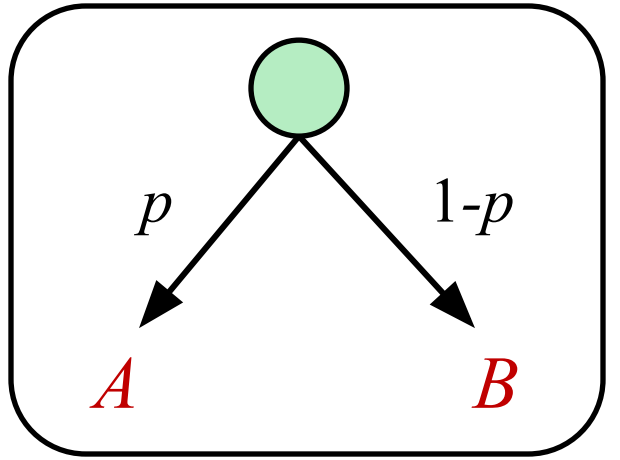
Preferences

- An agent must have preferences among:
  - Outcome □ Prizes: A, B, etc.
  - Probabilities □ Lotteries: situations with uncertain prizes

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

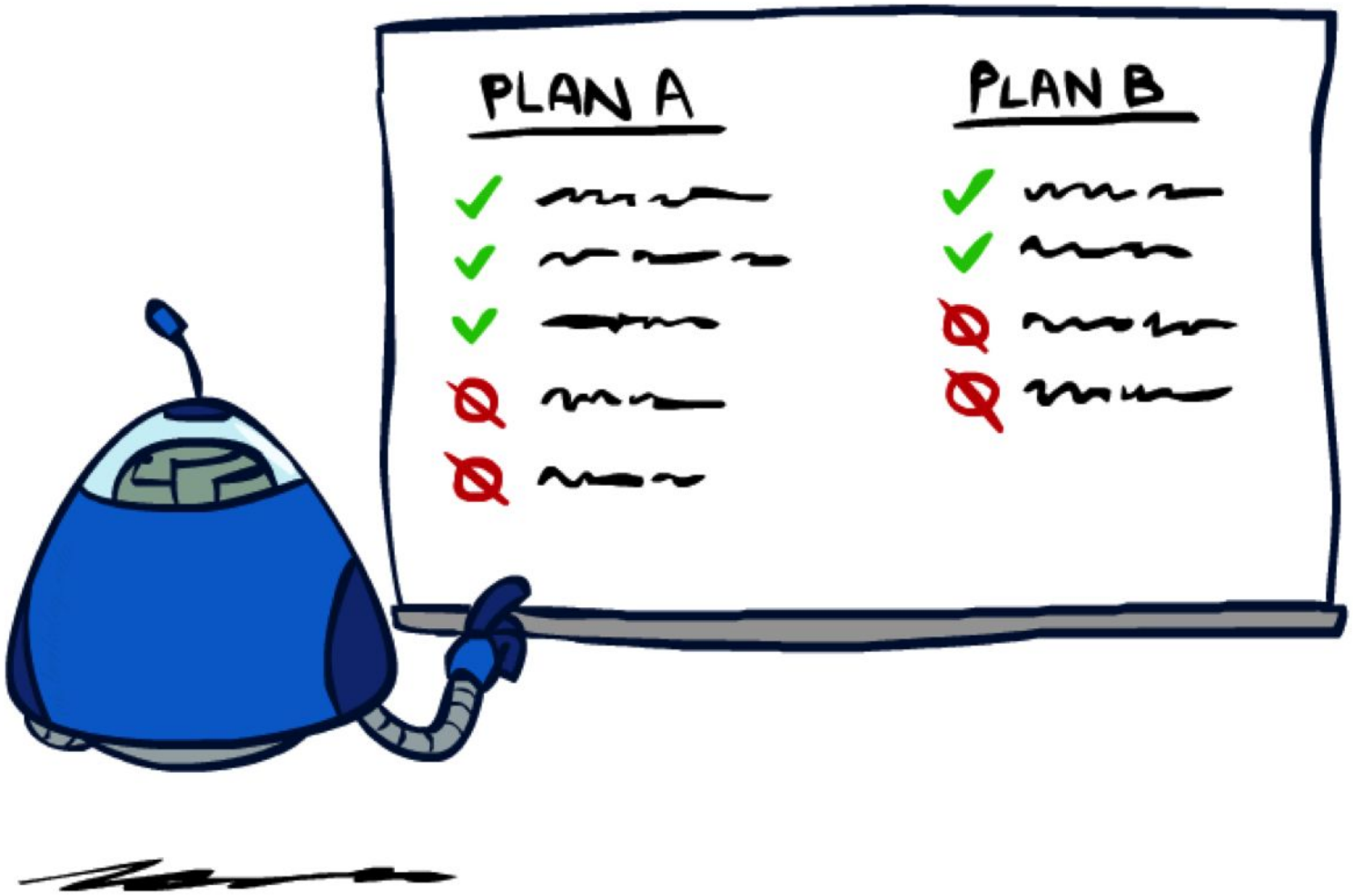
A Prize

A Lottery



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

Rationality

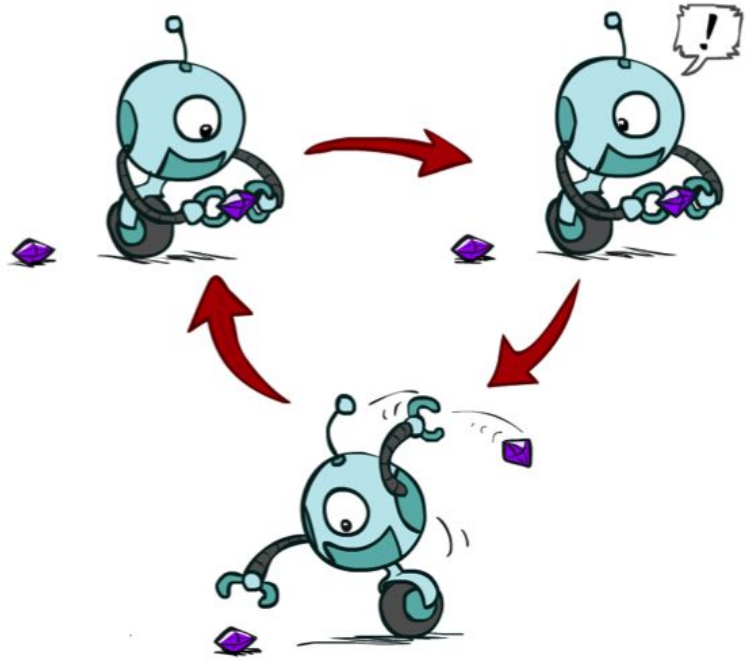


Rational Preferences

- 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

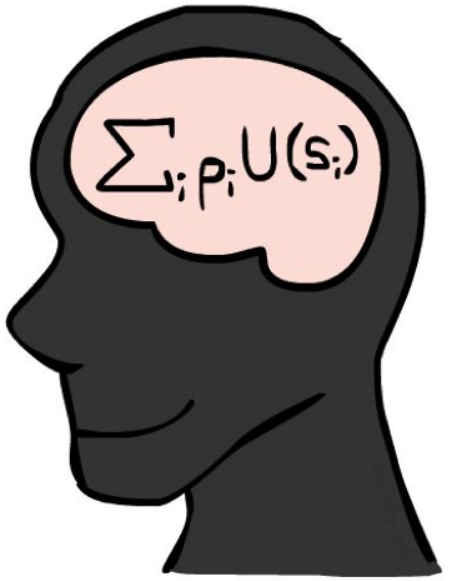
MEU Principle

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

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

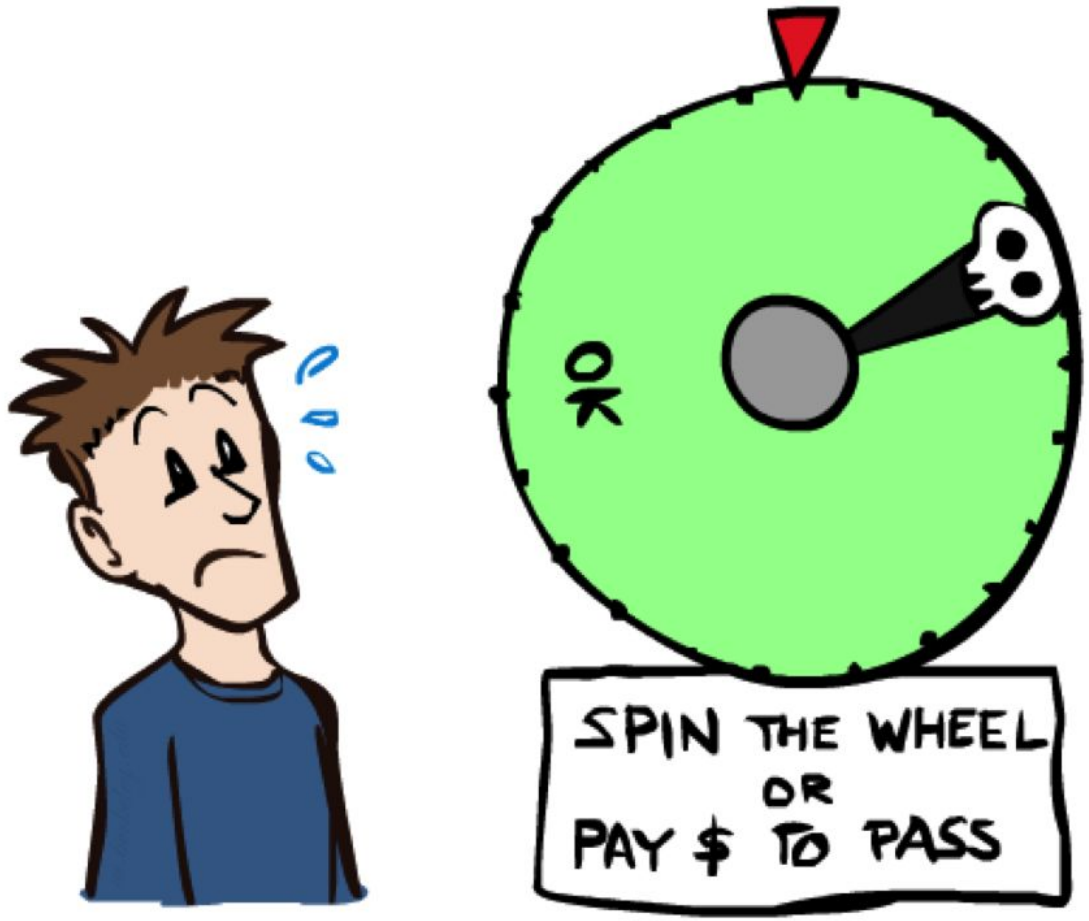
$$U([p_1, S_1;\; \ldots\; ;\; p_n, S_n]) = \sum_i p_i U(S_i)$$

  - Chance nodes of Expectimax
  - 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 ***(coincidence, no need to run Expectimax)***
  - 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

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

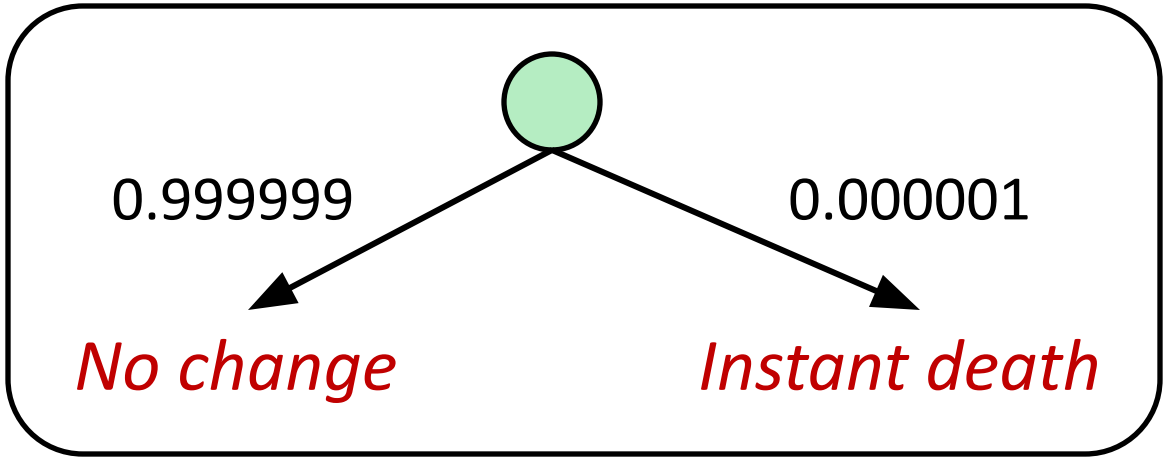
- With deterministic prizes only (no lottery choices), only ordinal utility can be determined, i.e., total order on prizes

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 ~ L_p
  - Resulting p is a utility in [0,1]

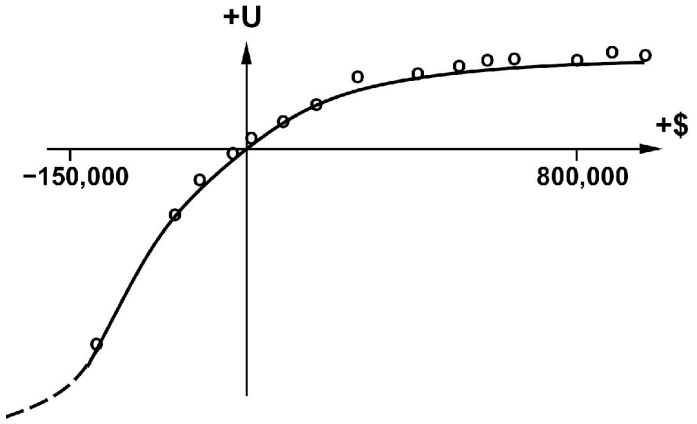
Pay $30

~



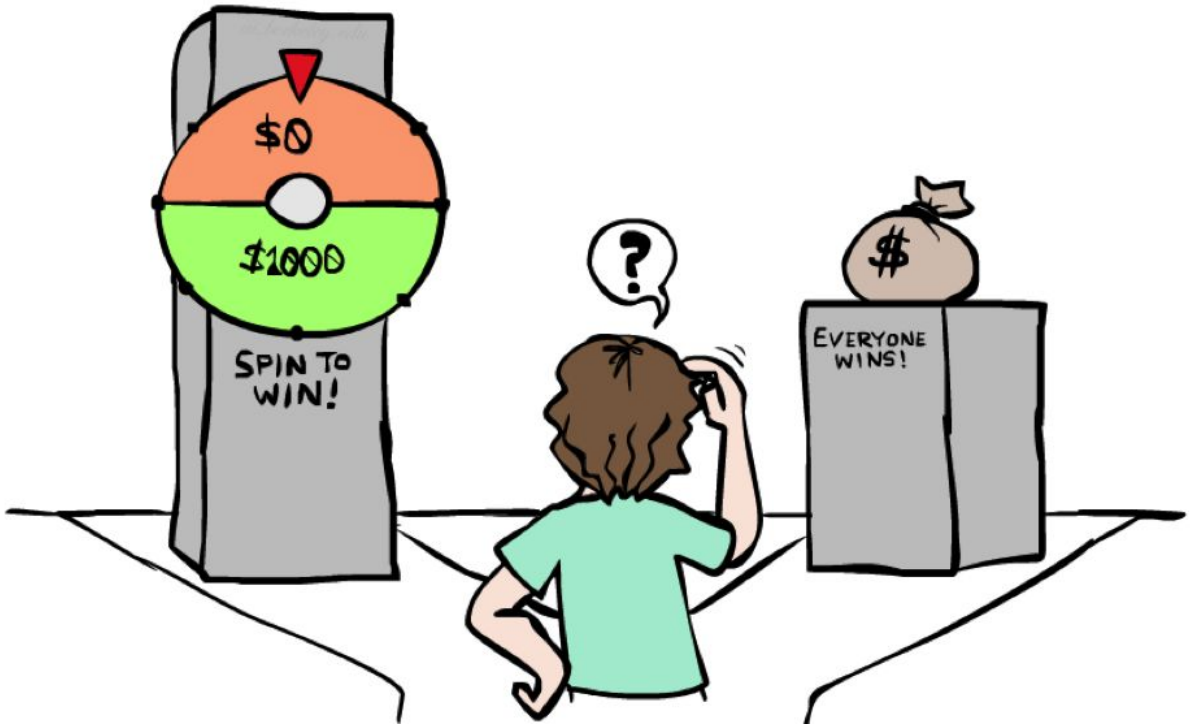
Money

- 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))
  - People often prefer to have a guarantee over the EMV in terms of dollars, rather than dealing with a lottery
  - 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 flat 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 ⇒ U($3k) > 0.8 U($4k)
  - C > D ⇒ 0.2 U($4k) > 0.25 U($3k)
    0.8 U($4k) > U($3k)