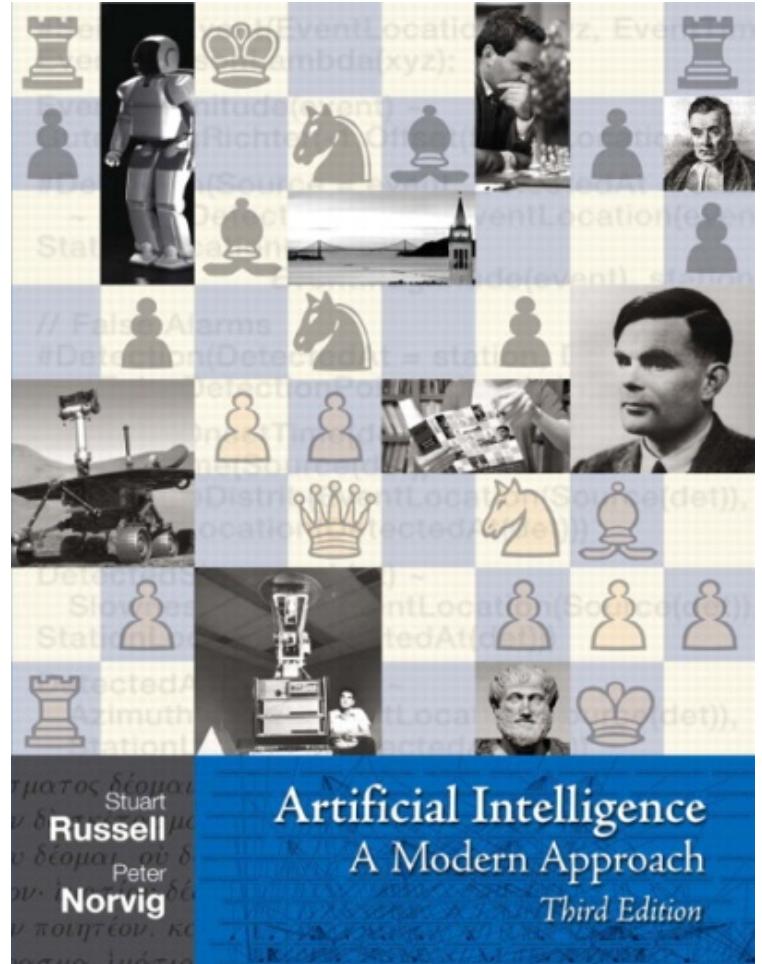


# Expectimax and Utilities

AIMA 5.1-5.5,  
AIMA 16.1-16.3

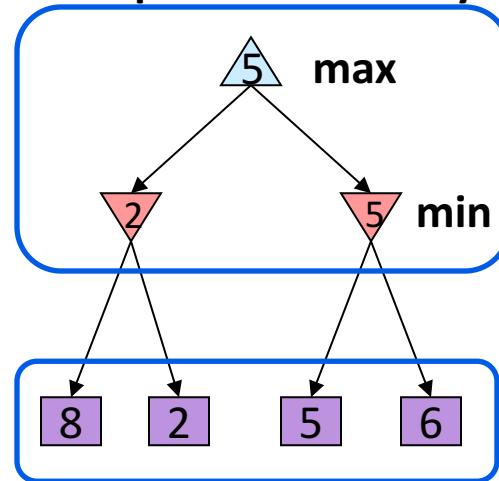
Many of today's slides are courtesy  
of Dan Klein and Pieter Abbeel  
University of California, Berkeley



# Review: Adversarial Search (Minimax)

- **Minimax search:**
  - A state-space search tree
  - Players alternate turns
  - Compute each node's **minimax value**: the best achievable utility against a rational (optimal) adversary

Minimax values:  
computed recursively



Terminal values:  
part of the game

# Review: Minimax Implementation

```
def max-value(state):
```

if the state is a terminal state:  
    return the state's utility

initialize  $v = -\infty$

for each successor of state:

$v = \max(v, \text{min-value(successor)})$

return  $v$



```
def min-value(state):
```

if the state is a terminal state:  
    return the state's utility

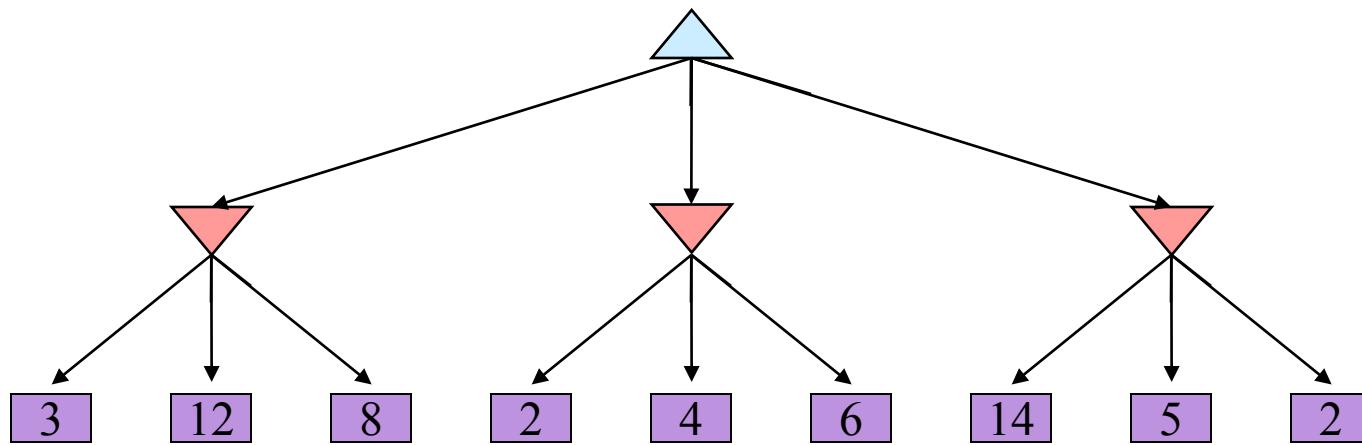
initialize  $v = +\infty$

for each successor of state:

$v = \min(v, \text{max-value(successor)})$

return  $v$

# Review Minimax Example



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



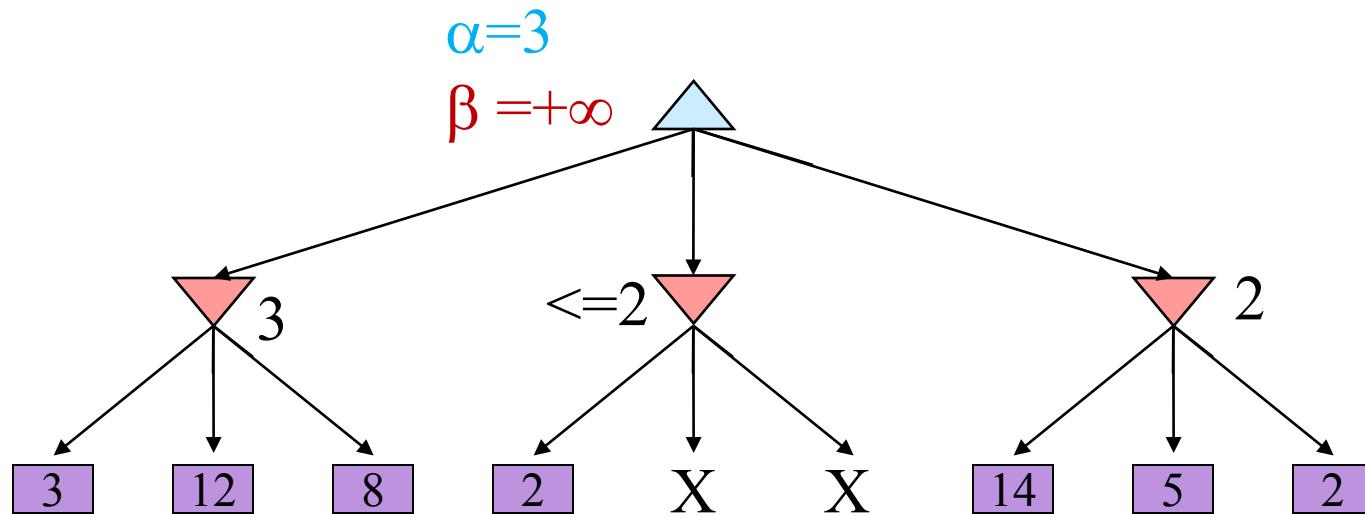
```
def min-value(state):  
    initialize v = +∞  
    for each successor of state:  
        v = min(v, max-value(successor))  
    return v
```

# Review: Alpha-Beta Pruning

---

- During Minimax, keep track of two additional values:
  - $\alpha$ : MAX's current *lower* bound on MAX's outcome
  - $\beta$ : MIN's current *upper* bound on MIN's outcome
- MAX will never allow a move that could lead to a worse score (for MAX) than  $\alpha$
- MIN will never allow a move that could lead to a better score (for MAX) than  $\beta$
- Therefore, stop evaluating a branch whenever:
  - When evaluating a MAX node: a value  $v \geq \beta$  is backed-up
    - MIN will never select that MAX node
  - When evaluating a MIN node: a value  $v \leq \alpha$  is found
    - MAX will never select that MIN node

# Review Alpha-Beta Pruning



$\alpha$ : MAX's best option on path to root  
 $\beta$ : MIN's best option on path to root

# Review: Evaluation functions

---

- Evaluates how good a ‘board position’ is
- Based on **static features** of that board alone
- Zero-sum assumption lets us use one function to describe goodness for both players.
  - $f(n) > 0$  if MAX is winning in position  $n$
  - $f(n) = 0$  if position  $n$  is tied
  - $f(n) < 0$  if MIN is winning in position  $n$
- Build using expert knowledge,
  - Tic-tac-toe:  $f(n) = (\# \text{ of } 3 \text{ lengths open for MAX}) - (\# \text{ open for MIN})$

(AIMA 5.4.1)

# Review: Chess Evaluation Functions

---

- Chess needs an evaluation function since it is impossible to search the game tree deeply enough to reach the terminal nodes
- $f(n) = (\text{sum of A's piece values}) - (\text{sum of B's piece values})$
- More complex: weighted sum of positional features:
$$\sum w_i \text{feature}_i(n)$$
- $f(n)$  can be a weighted linear function

|        |      |
|--------|------|
| Pawn   | 1.0  |
| Knight | 3.0  |
| Bishop | 3.25 |
| Rook   | 5.0  |
| Queen  | 9.0  |

Pieces values for a simple evaluation function often taught to novice chess players

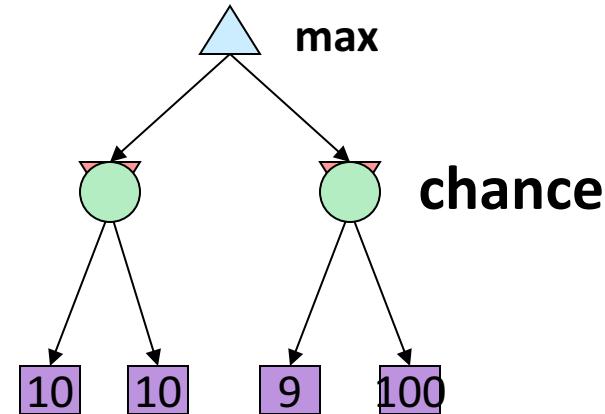
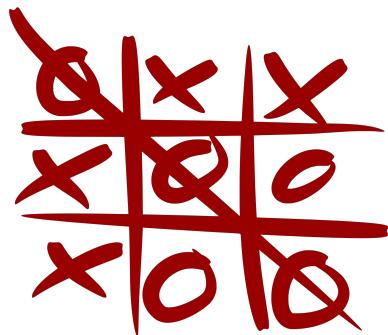
# Uncertain Outcomes

---



# Uncertain Outcomes

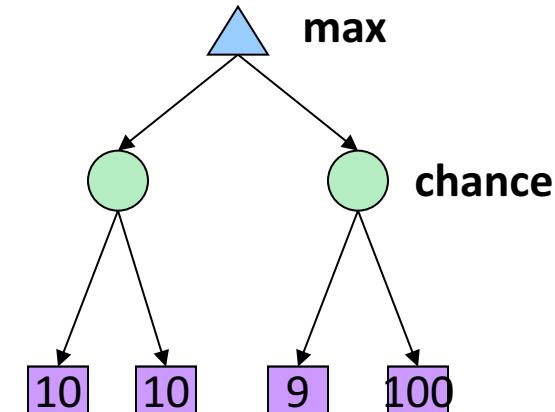
---



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  $v = -\infty$

    for each successor of state:

$v = \max(v, \text{value}(\text{successor}))$

    return  $v$

```
def exp-value(state):
```

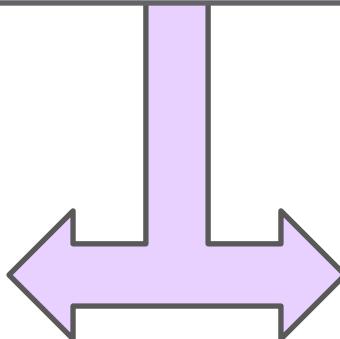
    initialize  $v = 0$

    for each successor of state:

$p = \text{probability}(\text{successor})$

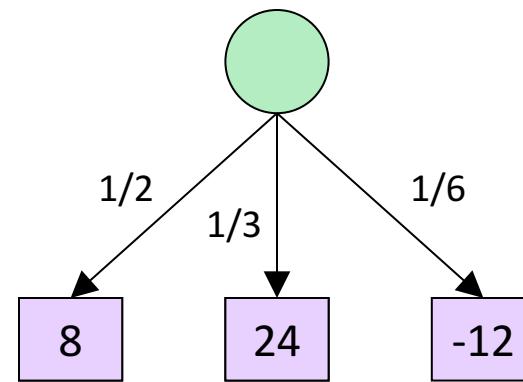
$v += p * \text{value}(\text{successor})$

    return  $v$



# Expectimax Pseudocode

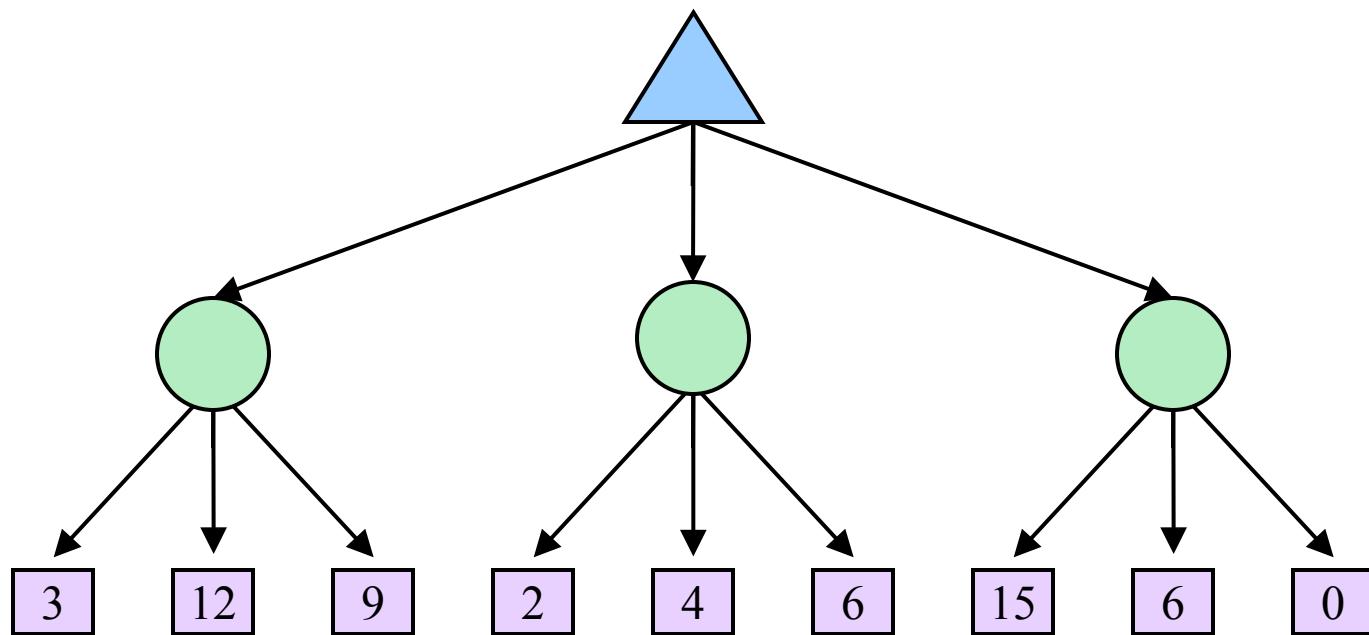
```
def exp-value(state):
    initialize v = 0
    for each successor of state:
        p = probability(successor)
        v += p * value(successor)
    return v
```



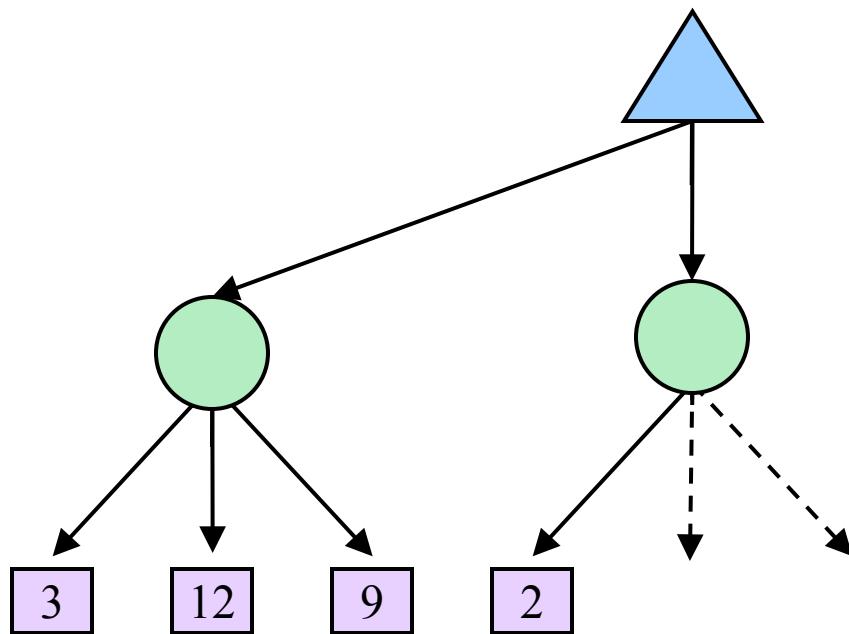
$$v = (1/2)(8) + (1/3)(24) + (1/6)(-12) = 10$$

# Expectimax Example

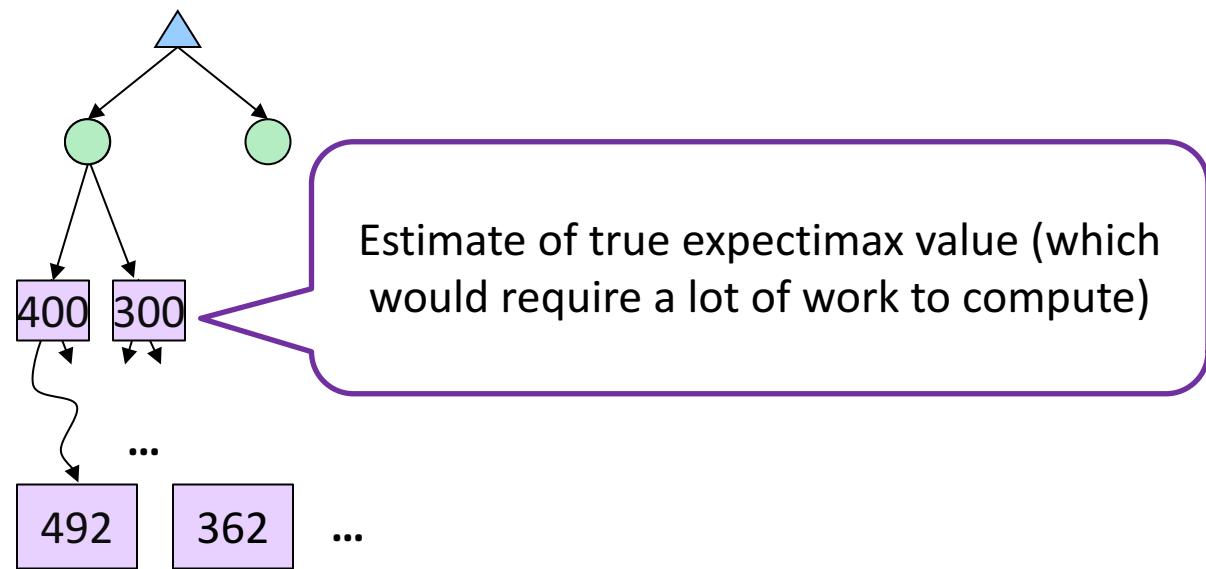
---



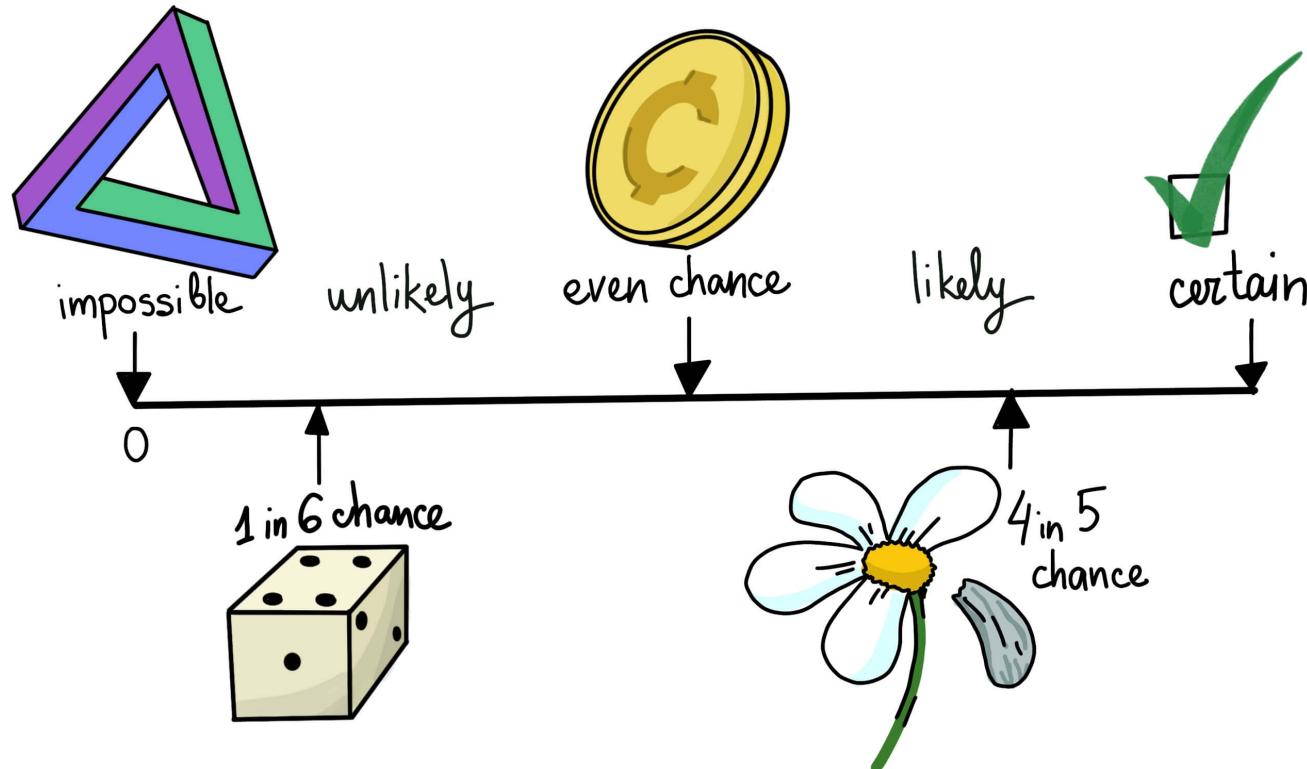
# Expectimax Pruning?



# Depth-Limited Expectimax



# Probabilities



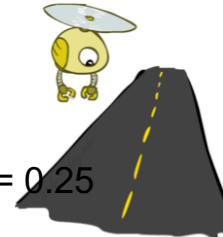
© luminousmen.com

# 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:  $T \in \{\text{none, light, heavy}\}$
- Distribution:  $P(T=\text{none}) = 0.25$ ,  $P(T=\text{light}) = 0.50$ ,  $P(T=\text{heavy}) = 0.25$



0.25

- **Some laws of probability (more later):**

- Probabilities are always non-negative
- Probabilities over all possible outcomes sum to one



0.50

- **As we get more evidence, probabilities may change:**

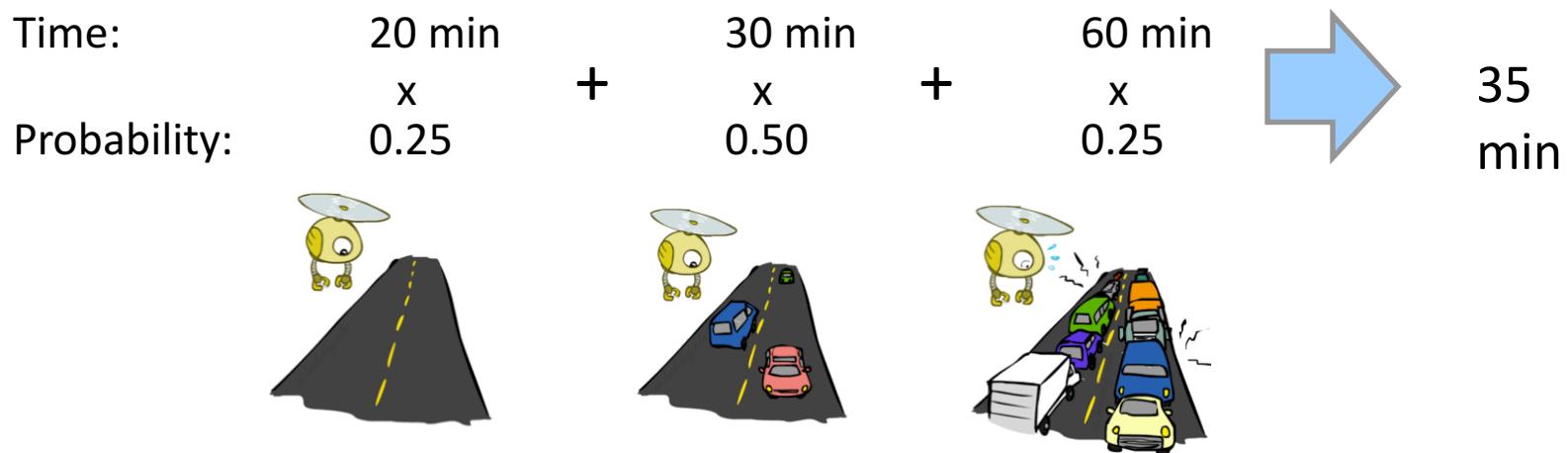
- $P(T=\text{heavy}) = 0.25$ ,  $P(T=\text{heavy} | \text{Hour}=8\text{am}) = 0.60$
- We'll talk about methods for reasoning and updating probabilities later



0.25

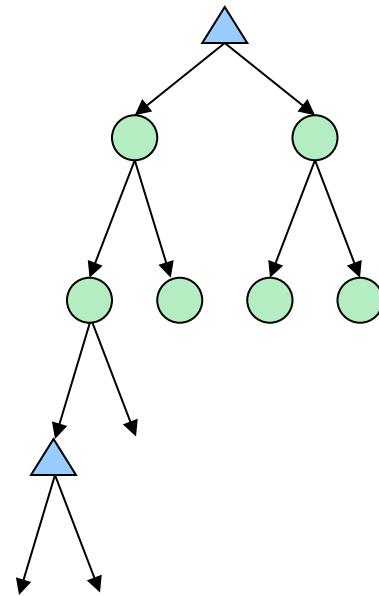
# Expectations

- 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

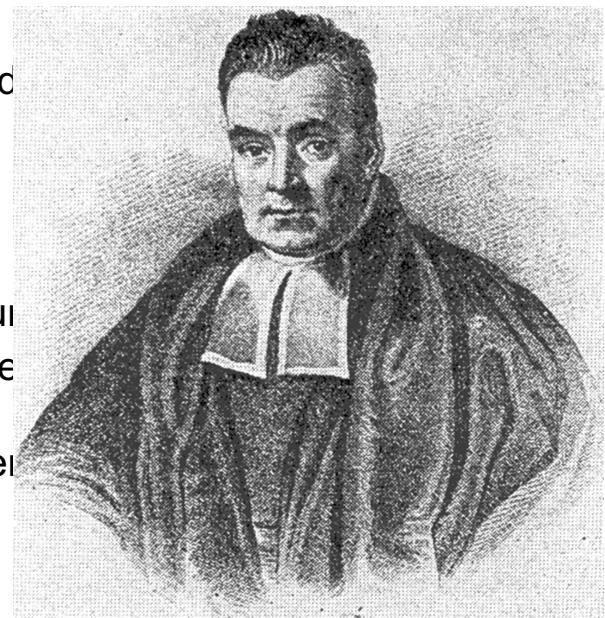


*Having a probabilistic belief about another agent's action does not mean that the agent is flipping any coins!*

# What are Probabilities?

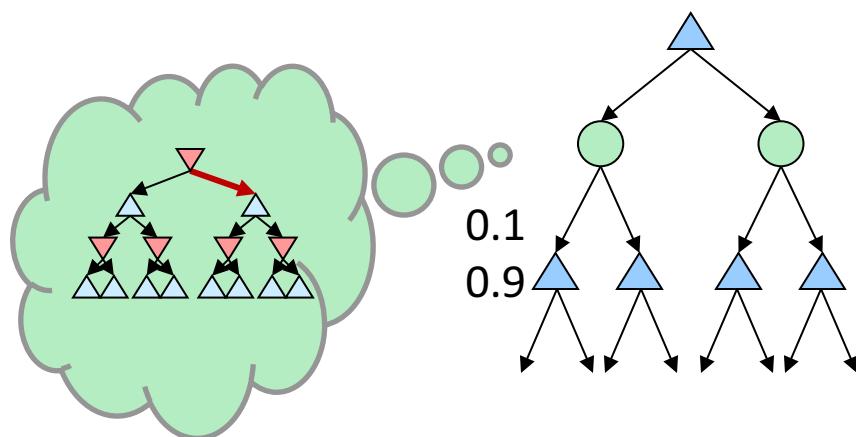
---

- **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

---

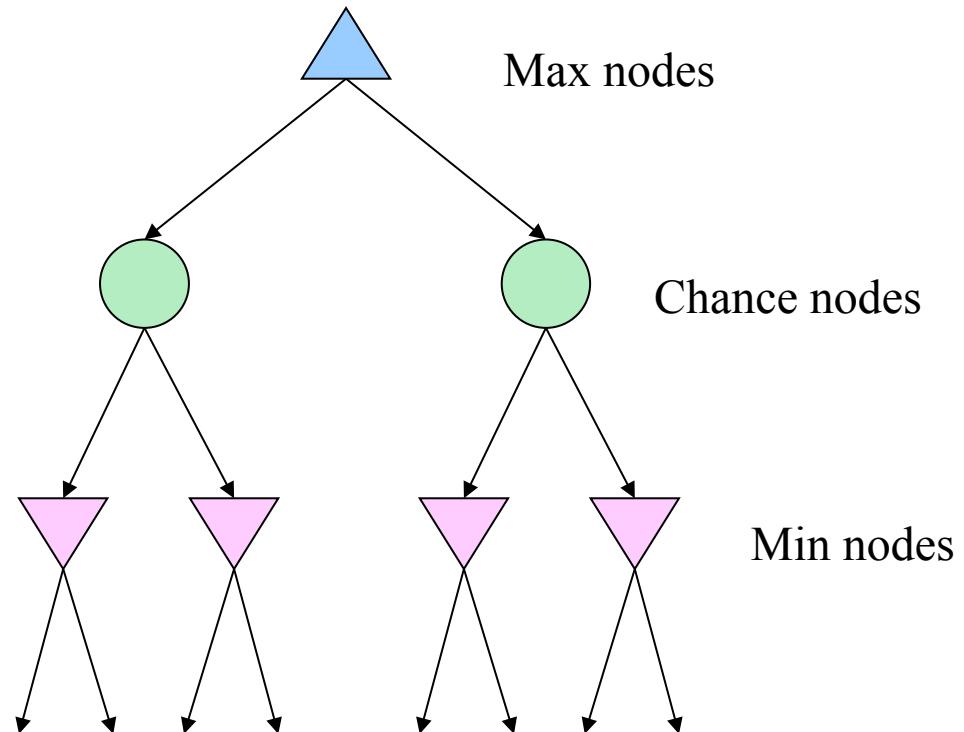
- **Dice rolls increase  $b$ : 21 possible rolls with 2 dice**
  - Backgammon  $\approx 20$  legal moves
  - Depth 2 =  $20 \times (21 \times 20)^3 = 1.2 \times 10^9$
- **As depth increases, probability of reaching a given search node shrinks**
  - So usefulness of search is diminished
  - So limiting 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**
- **1<sup>st</sup> AI world champion in any game!**



Image: Wikipedia

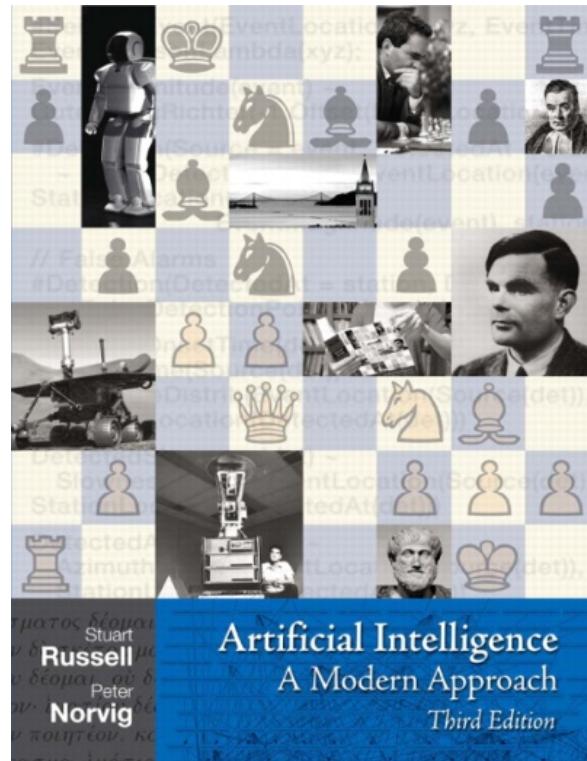
# Mixed Layer Types

- **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

# AIMA Chapter 16.1-16.3

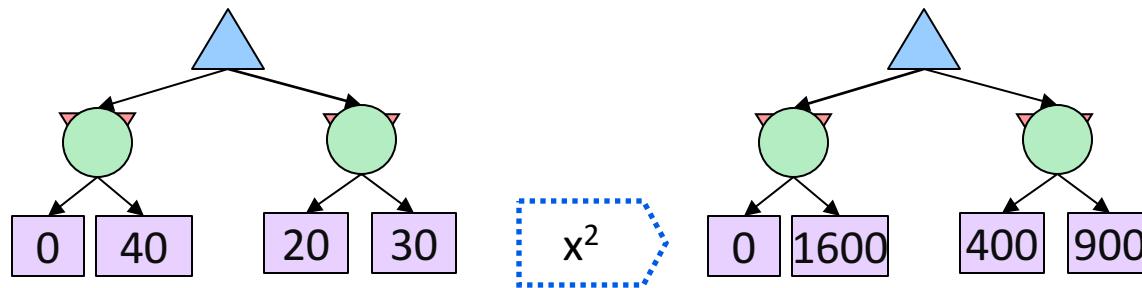


# Maximum Expected Utility

---

- 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

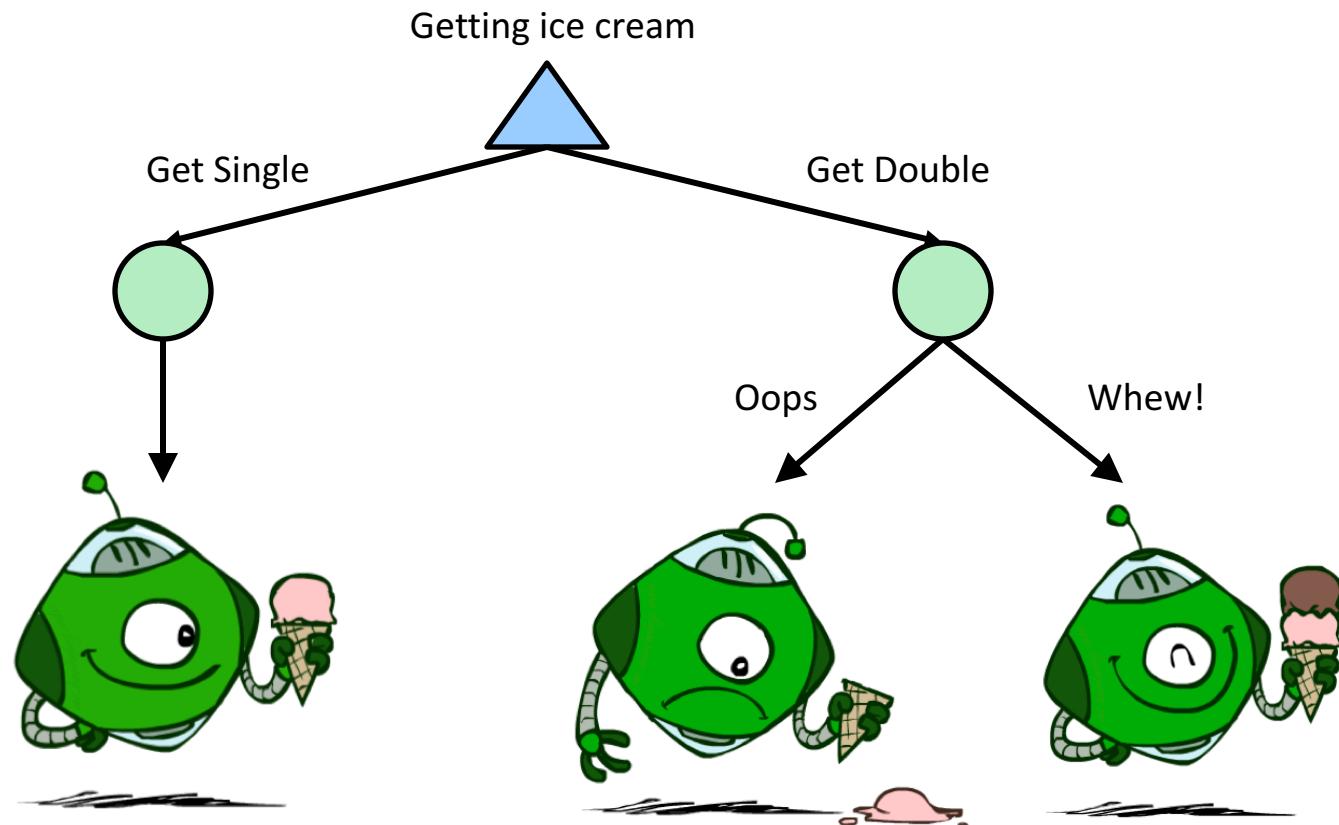
---

- 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

---



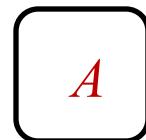
# Preferences

- An agent must have preferences among:

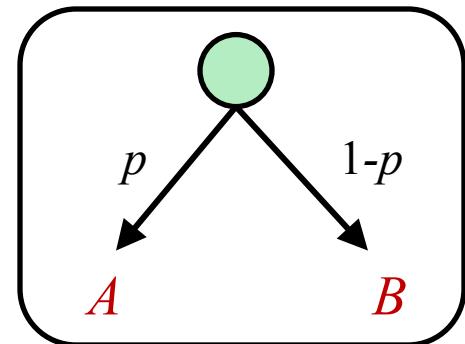
- Prizes:  $A, B$ , etc.
- Lotteries: situations with uncertain prizes

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

A Prize



A Lottery



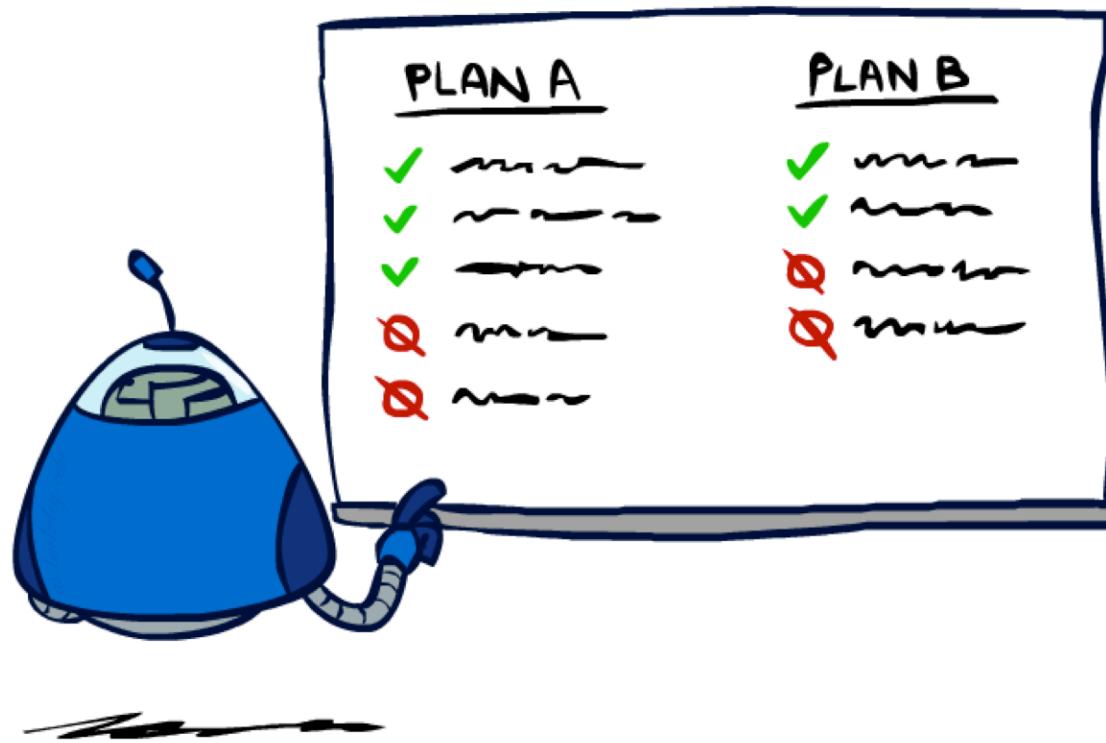
- Notation:

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



# Rationality

---



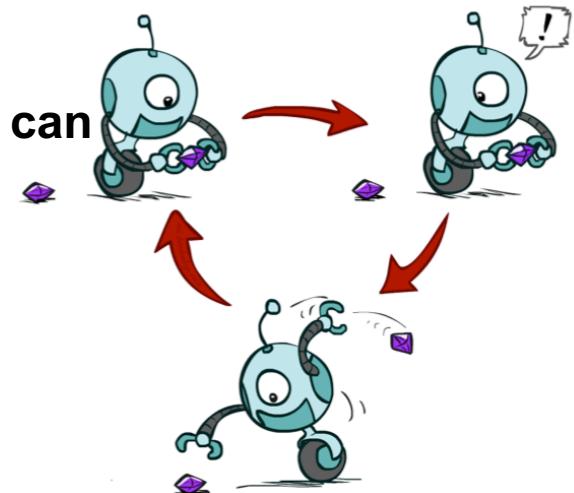
# Rational Preferences

- 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:

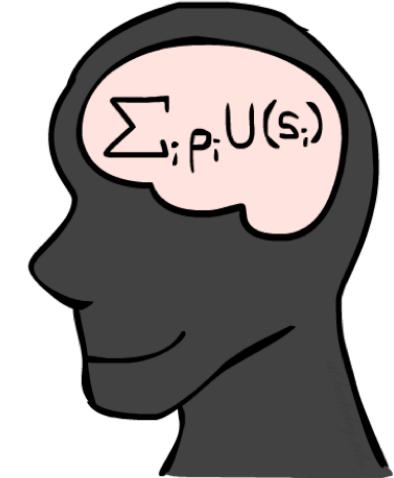
$$U(A) \geq U(B) \Leftrightarrow A \succeq B$$

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

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

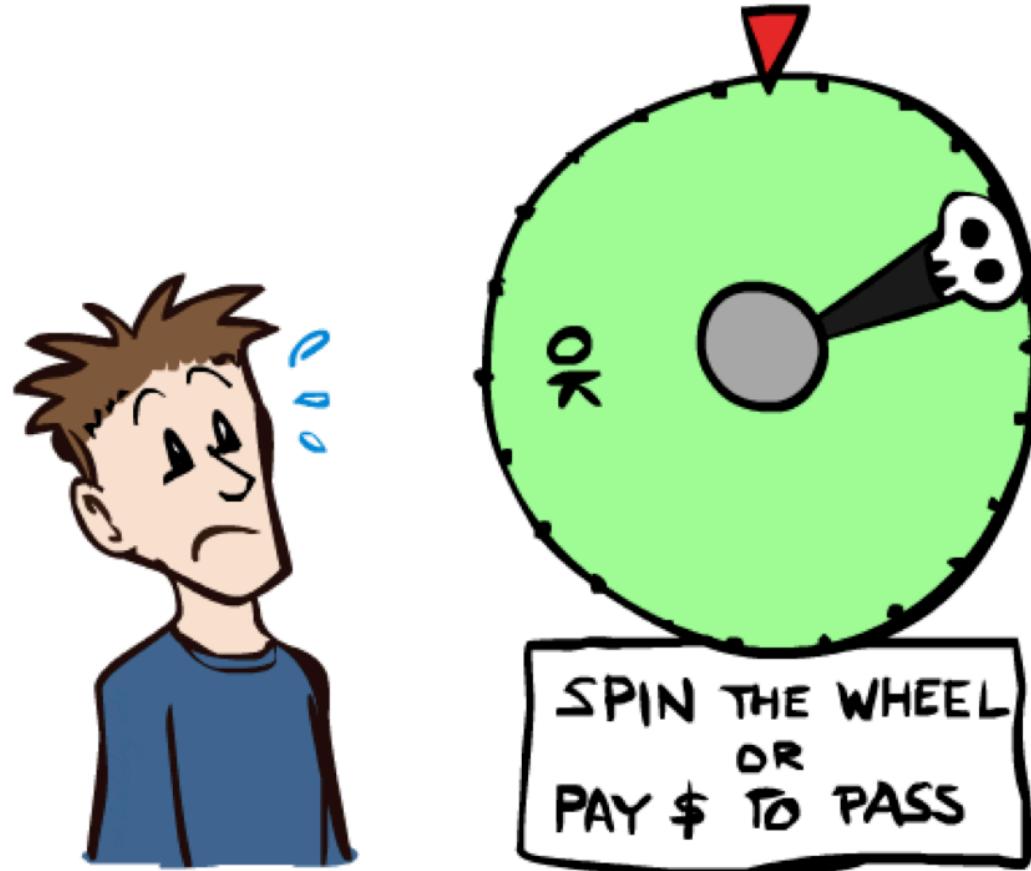
- **Maximum expected utility (MEU) principle:**

- 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



# Micromort examples

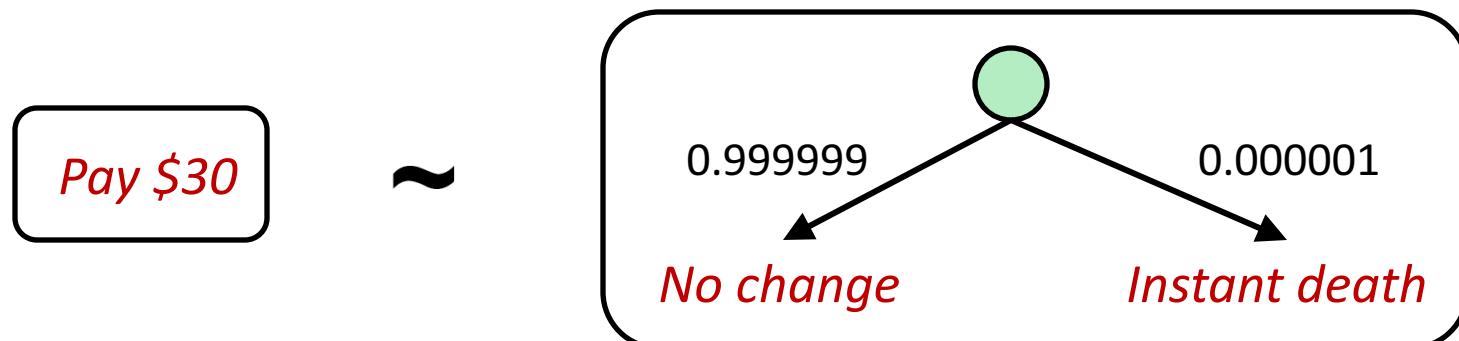
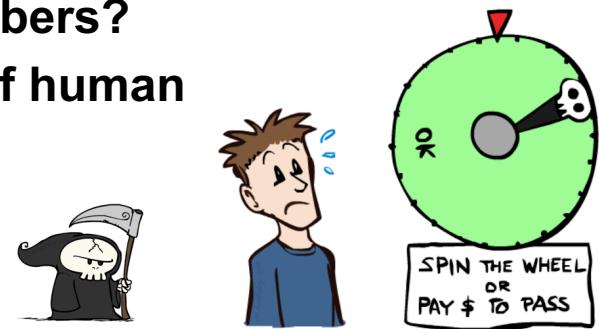
| Death from           | Micromorts per exposure |
|----------------------|-------------------------|
| Scuba diving         | 5 per dive              |
| Skydiving            | 7 per jump              |
| Base-jumping         | 430 per jump            |
| Climbing Mt. Everest | 38,000 per ascent       |

| 1 Micromort  |            |
|--------------|------------|
| Train travel | 6000 miles |
| Jet          | 1000 miles |
| Car          | 230 miles  |
| Walking      | 17 miles   |
| Bicycle      | 10 miles   |
| Motorbike    | 6 miles    |



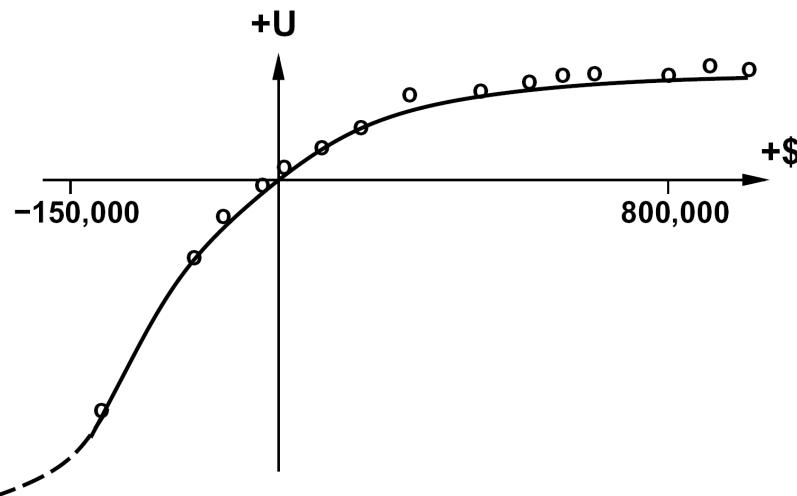
# Human Utilities

- Utilities map states to real numbers. Which numbers?
- Standard approach to assessment (elicitation) of human
  - 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  $[0,1]$



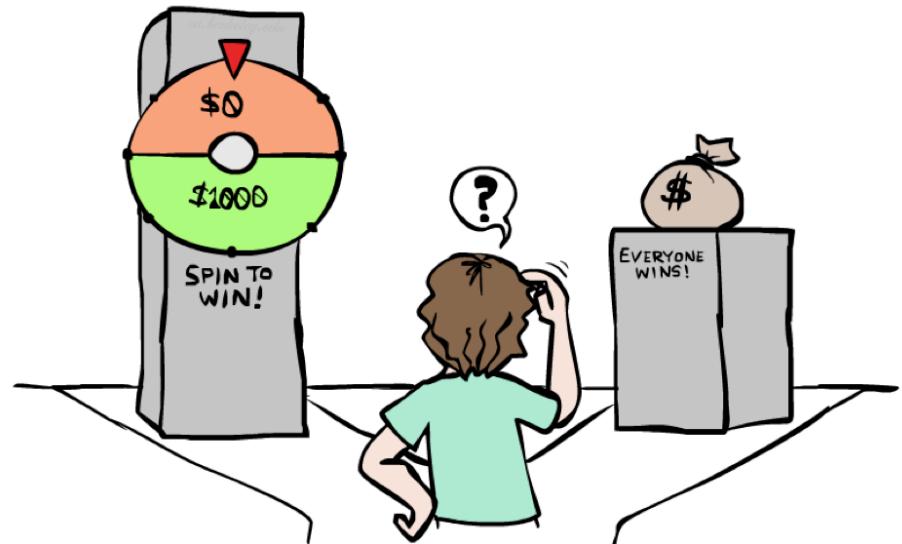
# Money

- 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
  - $B > A \Rightarrow U(\$3k) > 0.8 U(\$4k)$
  - $C > D \Rightarrow 0.8 U(\$4k) > U(\$3k)$

