Uncertainty and Utility

CSE 4617: Artificial Intelligence

Isham Tashdeed
Lecturer, CSE

Recap: Probabilities

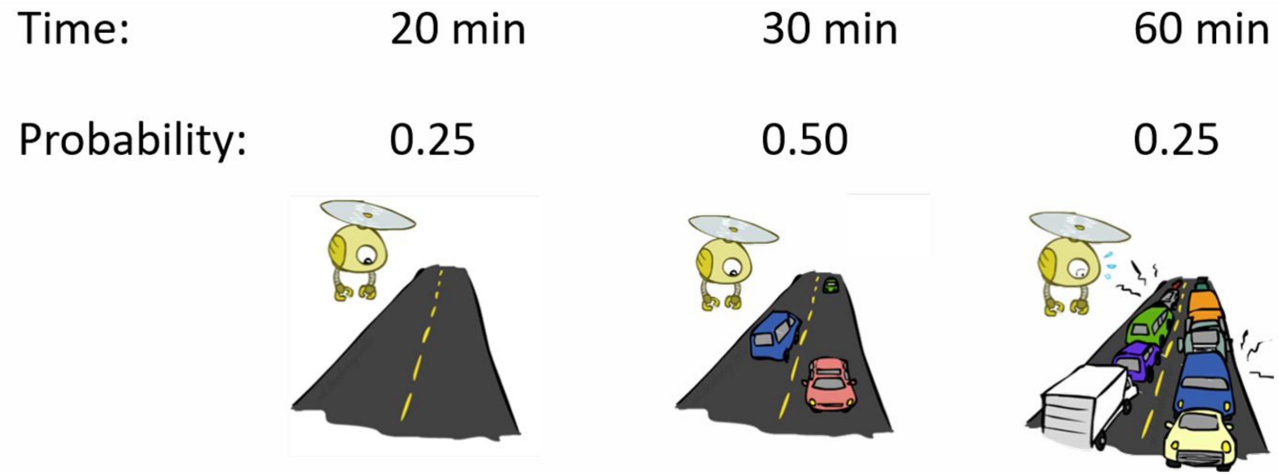
- Random Variable → An even whose outcome is unknown
- Probability Distribution → Assignment of weights to outcomes based on their chance
- Probabilities can’ be negative
- Sum of all possible outcomes = 1

Example: Getting that internship

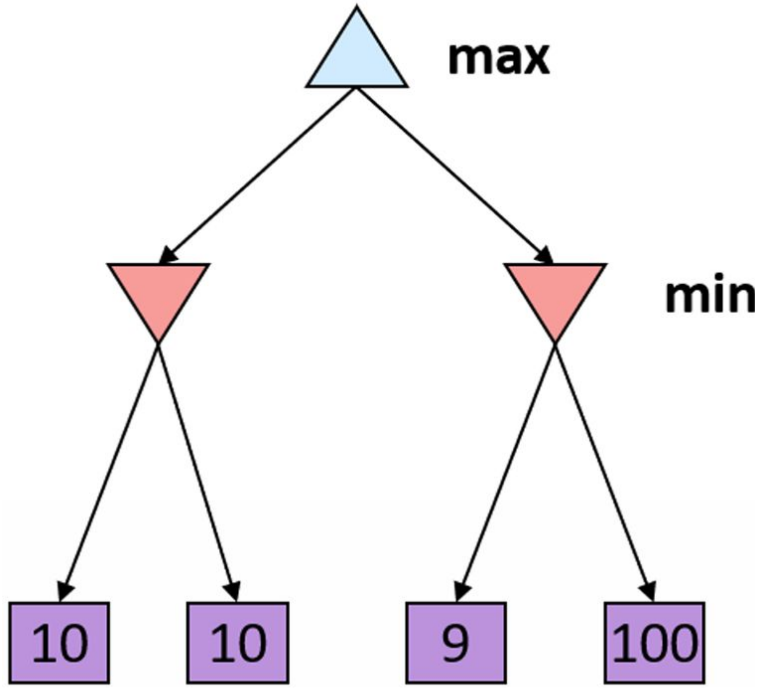
- Random variable: I → whether you got the internship
- Outcome: $I \in$ {none, good one, bad one}
- Distribution: P(I=none) = 0.1, P(I=good) = 0.4, P(I=bad) = 0.5

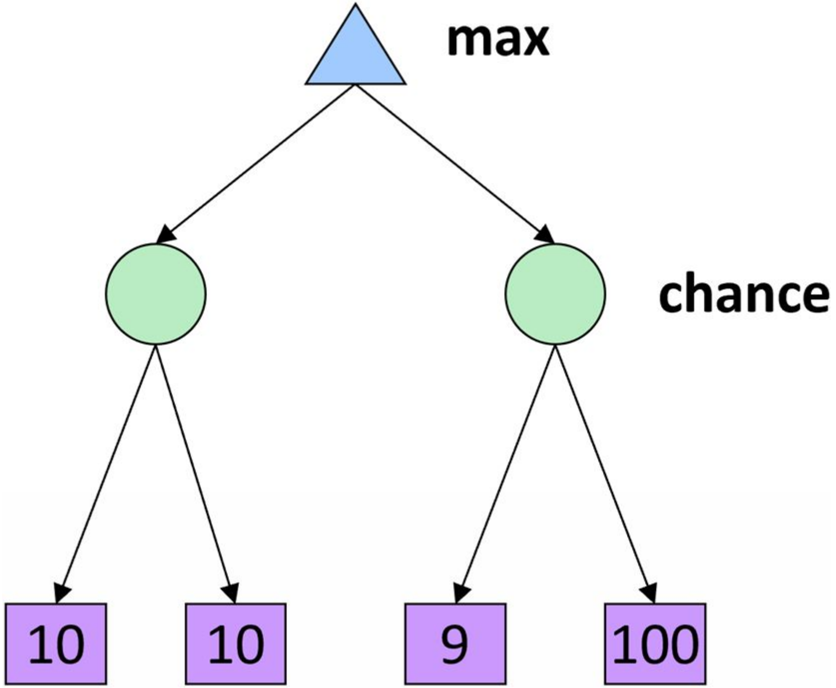
Recap: Probabilities

- Expected value → Average, weighted by the probability distribution over outcomes
- Sum of the values multiplied by their chance of occuring



Worst-case scenario vs. Average-case scenario





Worst-case scenario vs. Average-case scenario

- How wouldn't we know what the results of an action would be?
  - Explicit randomness → Dice roll, coin flip
  - Unpredictable opponents → Sometimes, opponents are not entirely rational agents
  - Failed actions → Wheel slip, traffic congestion
- Values should reflect average-case outcomes, not worst-case outcomes
  - Max nodes are the same as mini-max
  - Chance nodes are replacing min nodes but the outcome has probabilities attached with it
  - Calculate the **expected utilities** → Take weighted average of children

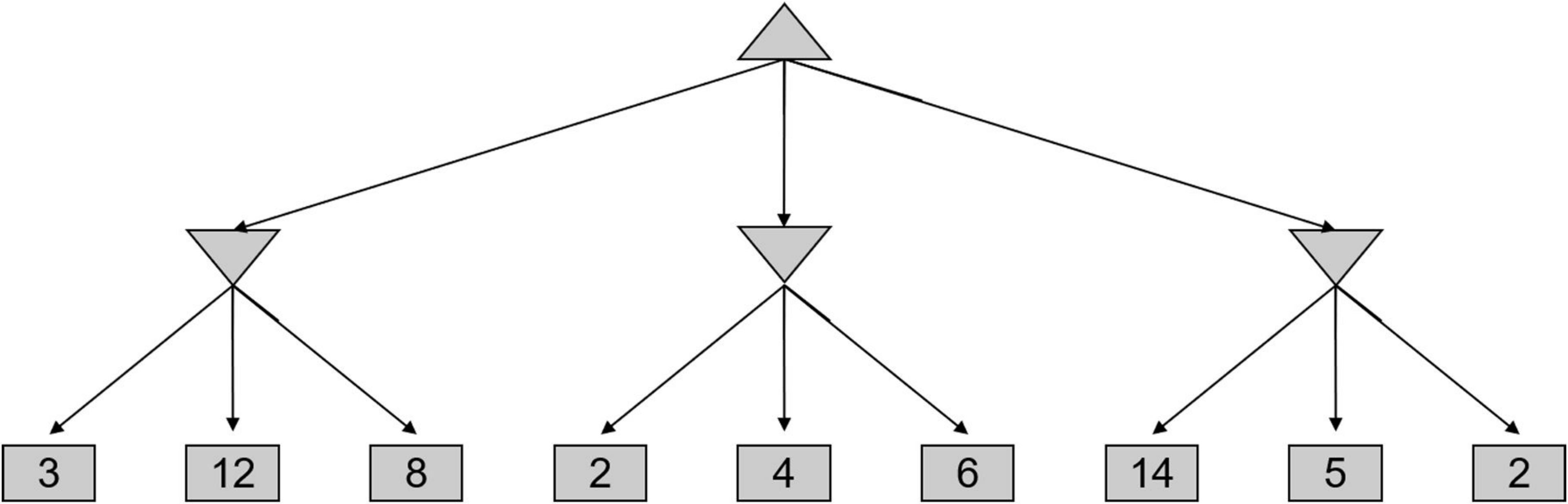
Minimax Algorithm

```
VALUE (state) → returns utility value of the state
    if state is TERMINAL: then return utility
    if state is MAX-AGENT: then return MAX-VALUE (state)
    if state is MIN-AGENT: then return MIN-VALUE (state)


MAX-VALUE (state) → returns utility value of a state
    v ← −∞
    for successor in state:
        v = max (v, VALUE (successor))
    return v


MIN-VALUE (state) → returns utility value of a state
    v ← +∞
    for successor in state:
        v = min (v, VALUE (successor))
    return v
```

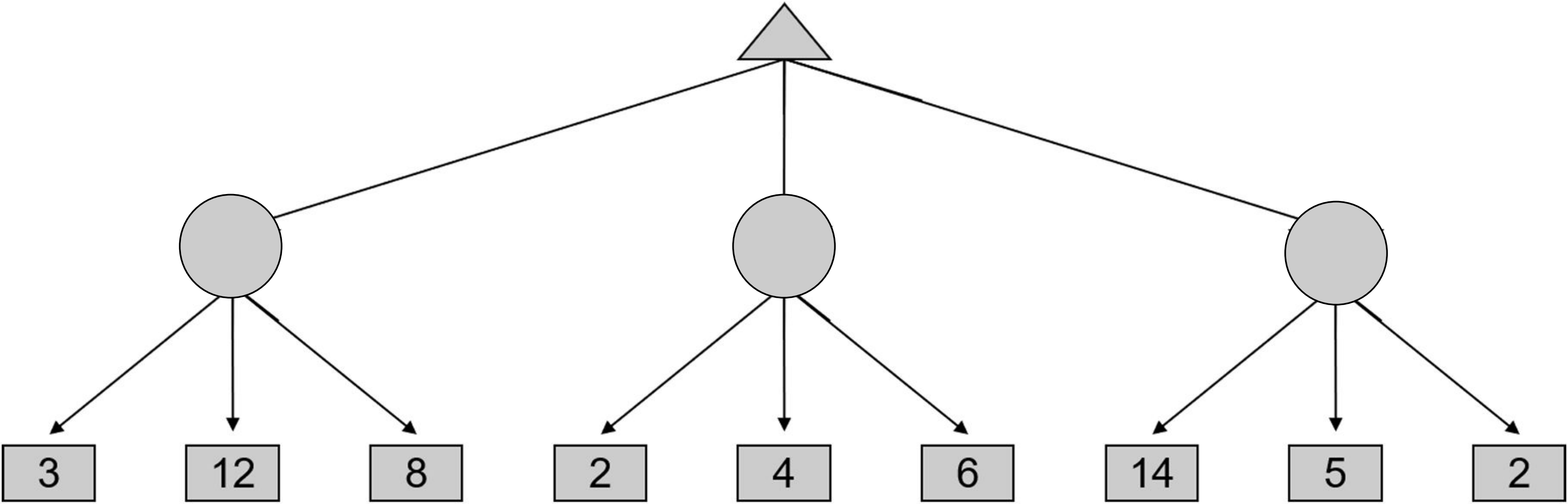
Expectimax Algorithm

```
VALUE (state) → returns utility value of the state
     if state is TERMINAL: then return utility
     if state is MAX-AGENT: then return MAX-VALUE (state)
     if state is EXP-AGENT: then return EXP-VALUE (state)


MAX-VALUE (state) → returns utility value of a state
     v ← -∞
     for successor in state:
          v = max (v, VALUE (successor))
     return v


MIN-VALUE (state) → returns expected utility value of a state
     v ← 0
     for successor in state:
          p ← probability (successor)
          v += p * VALUE (successor)
     return v
```

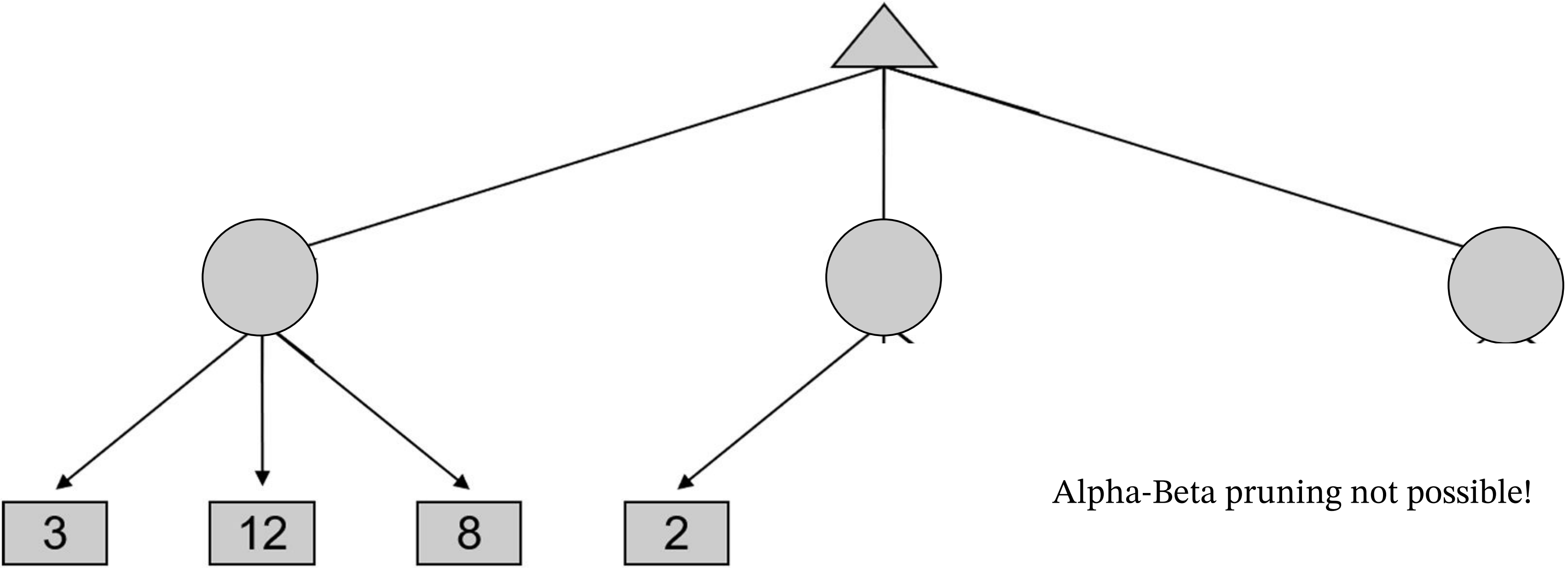
Minimax Algorithm



Expectimax Algorithm



Expectimax Pruning?



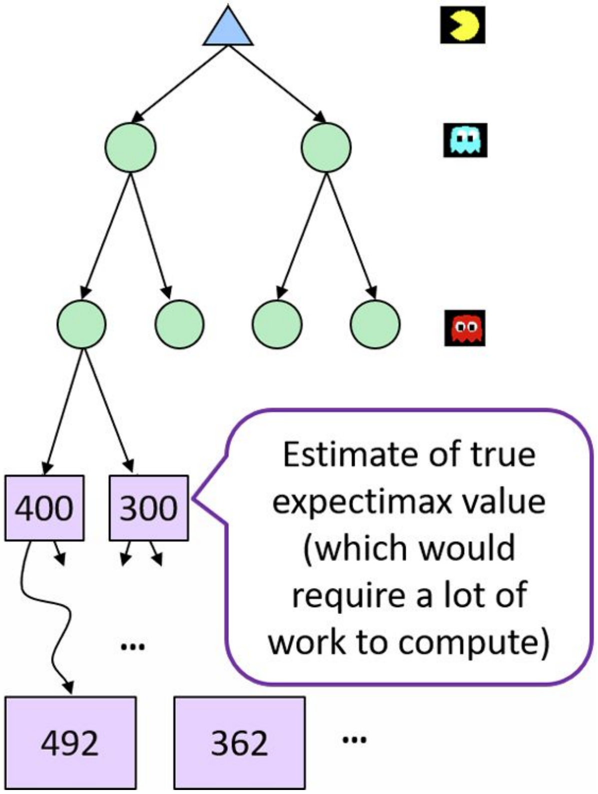
Alpha-Beta pruning not possible!

Depth Limited Expectimax

In demanding games like chess, we can not search to the leaves!

Depth-limited Search:

- Search only to a limited depth in the tree
- Replace terminal utilities with an evaluation function for non-terminal positions
- Guarantee of optimal play is gone
- The more depth you check, the better your moves become
- Use iterative deepening or a better outcome



What Probability Model 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
  - Model could be sophisticated and require a great deal o computation
  - We have a chance node for any outcome out of our control: opponent or environment
- Assume each chance node magically comes along with probabilities that specify the distribution over its outcomes
- Having a chance node to model an opponent's action does not mean that the agent is also following the same distribution → It is just an estimate!

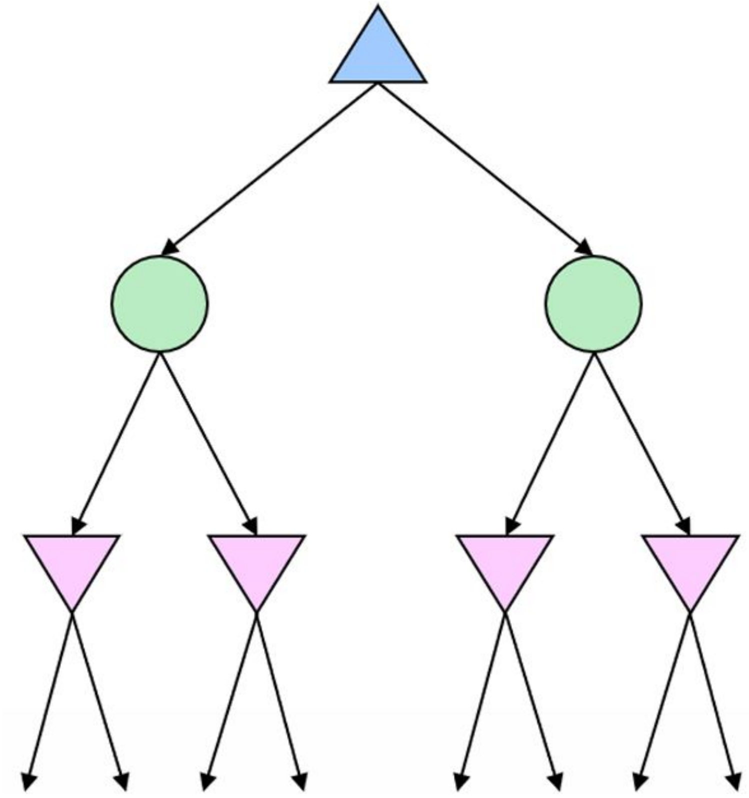
Dangers of Optimism and Pessimism

	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

Other Types of Games

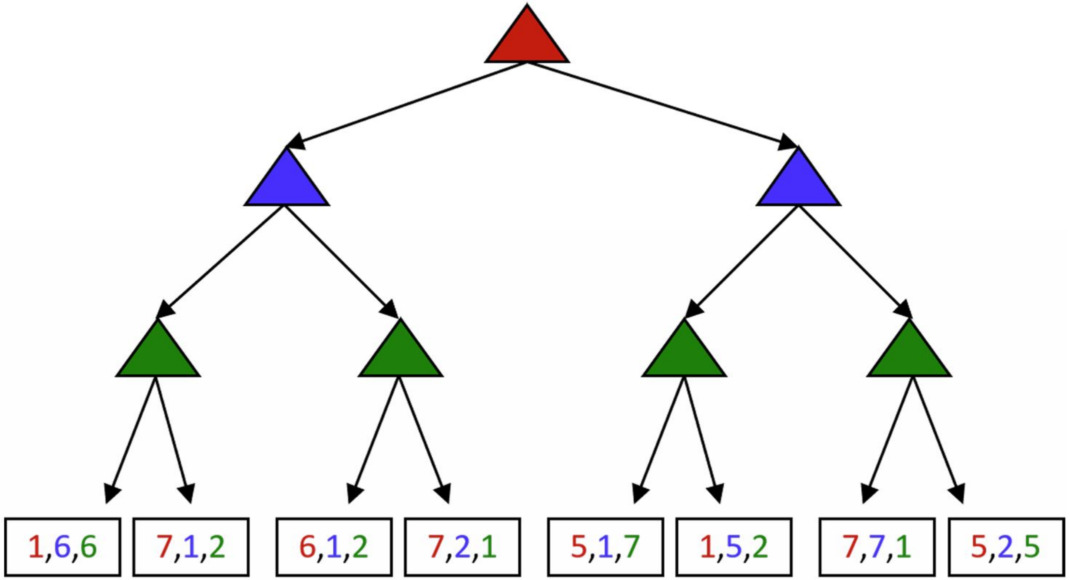
Mixed Layer Type Games

- Expectiminimax
  - Environment is an extra “random agent” player that moves after each min/max agent
  - Each node computes the appropriate combination of its children
- Examples could be:
  - Backgammon
  - Ludo
  - Snakes and Ladders
  - Uno



Mixed Layer Type Games

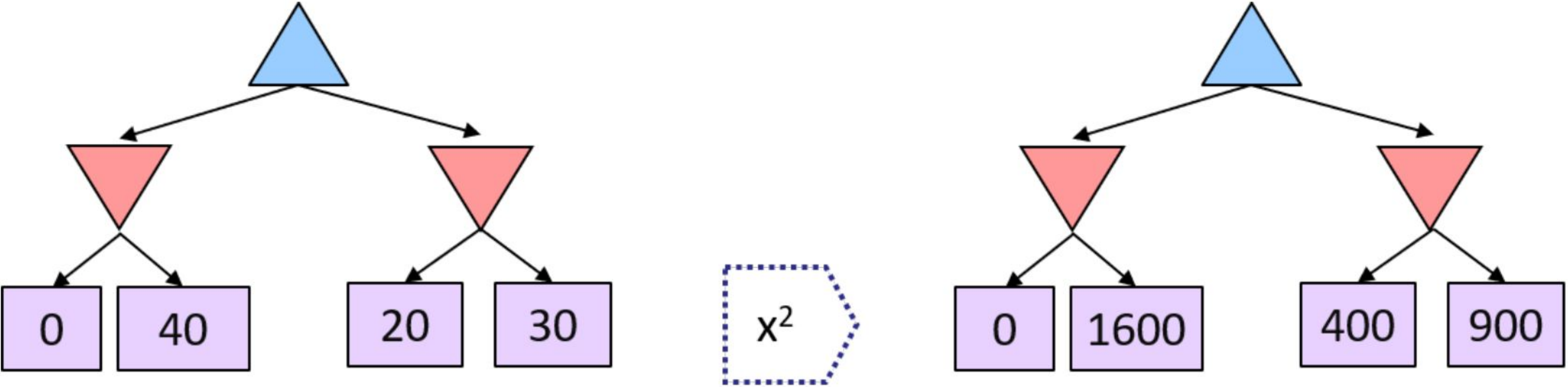
- What if a game is not zero-sum or has multiple players?
- Terminals have utility tuples
- Node values are also utility tuples
- Each player maximizes its own component → Not zero-sum



Utilities

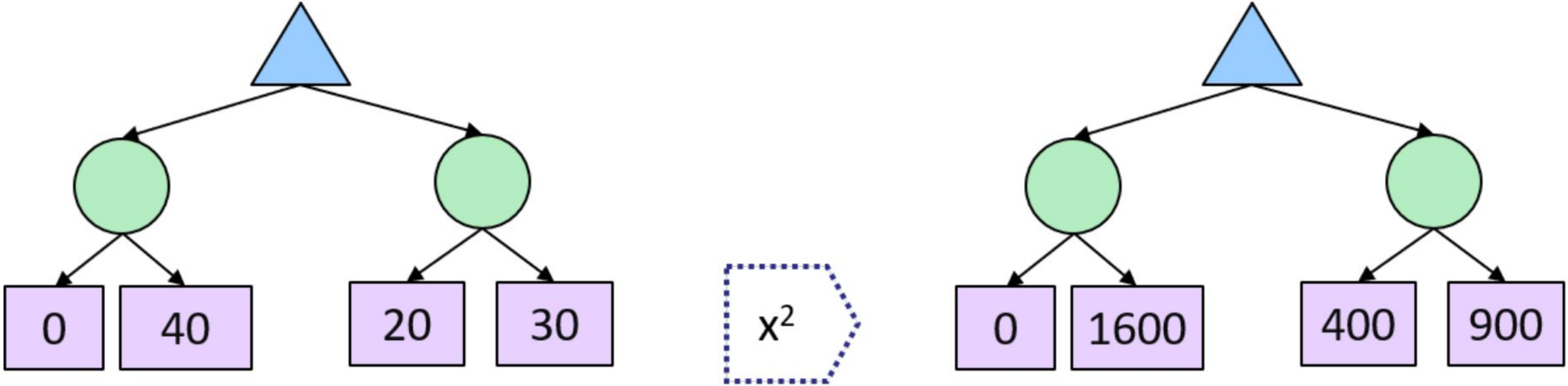
Utilities

- Principle of maximum expected utility
  - A rational agent should chose the action that maximizes its expected utility, given its knowledge as noted by its model of the world
- But what do these numbers really mean? Can we let an agent decide?



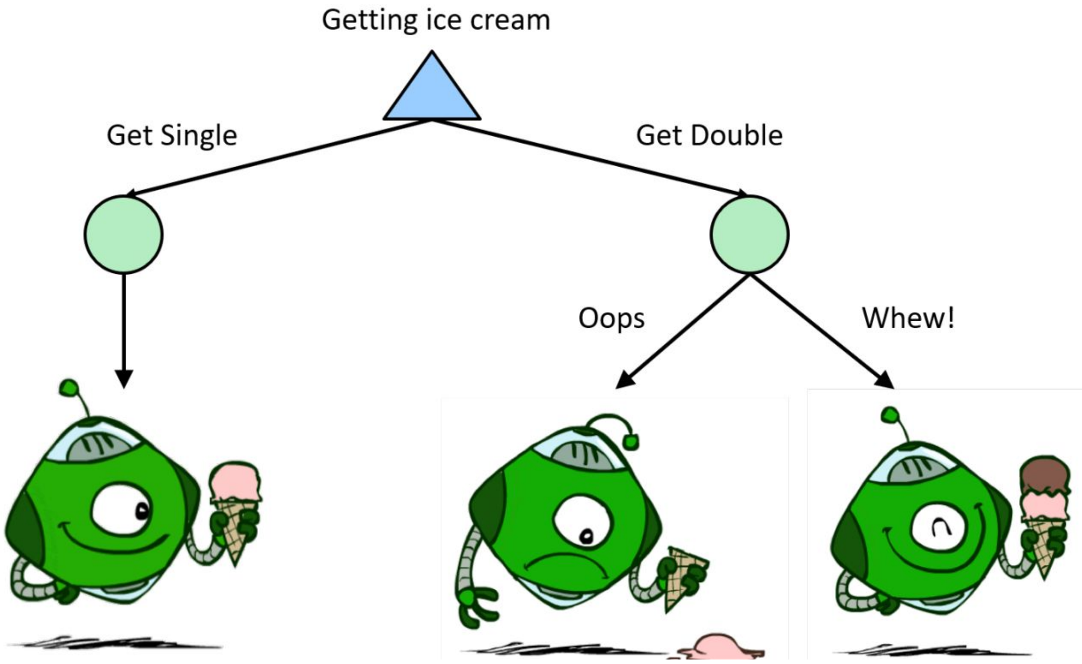
Utilities

- 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 real numbers that describe the agent's preference
- Any "rational" preference can be summarized as a utility function → Proof?



Preferences

- Prizes → A, B
- Lottery → L=[p, A; (1-p), B]
- Preference → $A \succ B$ Indifference → $A \sim B$
- Lottery doesn't mean an actual game of lottery, any event with multiple outcomes and probability values for those outcomes can be considered as a lottery

Rational Preference

- We want some constraints on preferences before we call them rational, such as:
  - Transitivity: $(A \succ B) \wedge (B \succ C) \rightarrow (A \succ C)$
  - Orderability: $(A \succ B) \vee (B \succ A) \vee (A \sim B)$
  - 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: $A \succ B \rightarrow (p \geq q \leftrightarrow [p, A; (1 - p), B] \succcurlyeq [q, A; (1 - q), B])$
- **Theorem**: Rational preferences imply behavior describable as maximization of expected utility

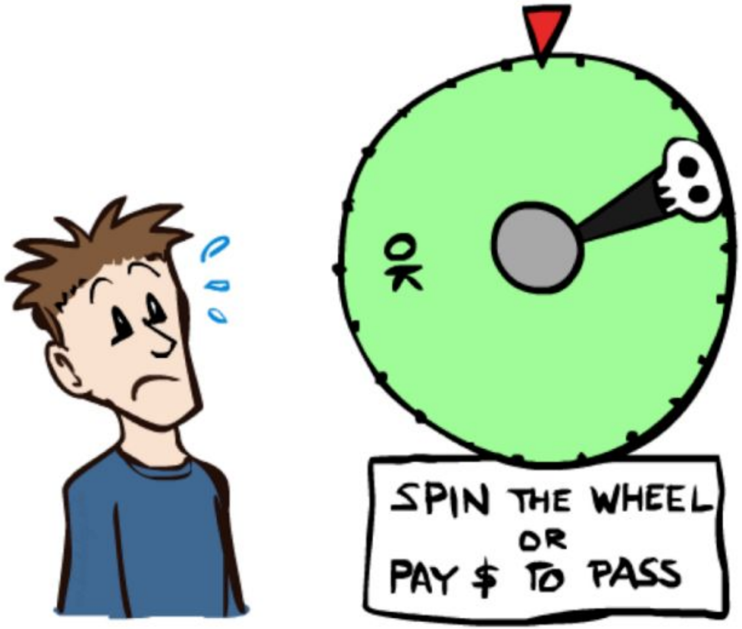
MEU Principle

- [Ramsey, 1931; von Neumann & Morgenstern, 1944]
  - Given any preferences satisfying these constraints, there exists a real-valued function U such that:
  - $A \succcurlyeq B \leftrightarrow U(A) \geq U(B)$ where, $U(p_1, S_1; p_2, S_2; \ldots) = \sum_i p_i U(S_i)$
- Choose the action that maximizes expected utility
- An agent can be entirely rational (consistent with MEU) without ever representing or manipulating utilities and probabilities

Human Utilities

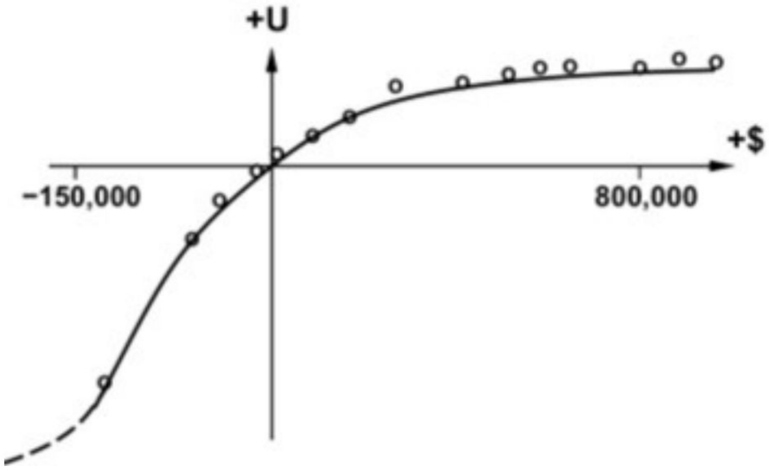
Human Utilities

- Utilities map states to real numbers
- Standard approach to asses human utilities:
  - Compare a prize A to a standard lottery L_p between:
    - Best possible prize with prob p
    - Worst possible prize with prob (1-p)
  - Adjust lottery probability p until indifference: $A \sim L_p$



Utility of Money

- Money does not behave as a utility function
- Given a lottery $L = [p, \$X; (1\text{-}p), \$Y]$
  - Expected monetary value (EMV) $\rightarrow p \times X + (1\text{-}p) \times Y$
  - Utility $U(L) \rightarrow p \times U(\$X) + (1\text{-}p) \times U(\$Y)$
  - Usually, $U(L) < U(\text{EMV}(L))$
- People are risk-averse in usual scenarios
- When deep in debt, people are risk-prone
- Useful when modeling complex functions like insurance premium



Are Humans Rational?

Thank you

Additional Resources

- AI 101: Monte Carlo Tree Search
- What's the Use of Utility Functions?
- Eliezer Yudkowsky – AI Alignment: Why It's Hard, and Where to Start