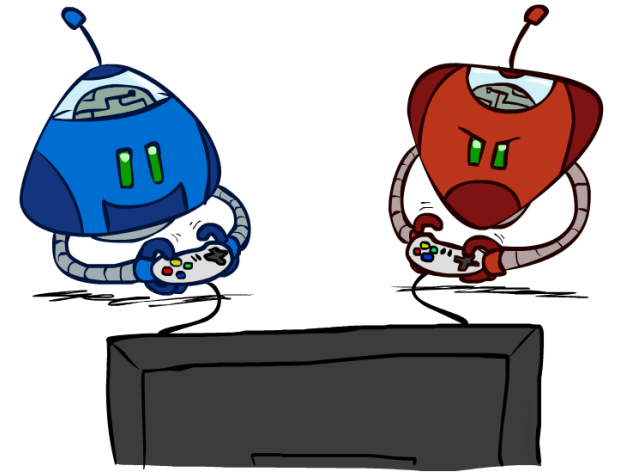


Adversarial Search

CSE 473: Introduction to Artificial Intelligence



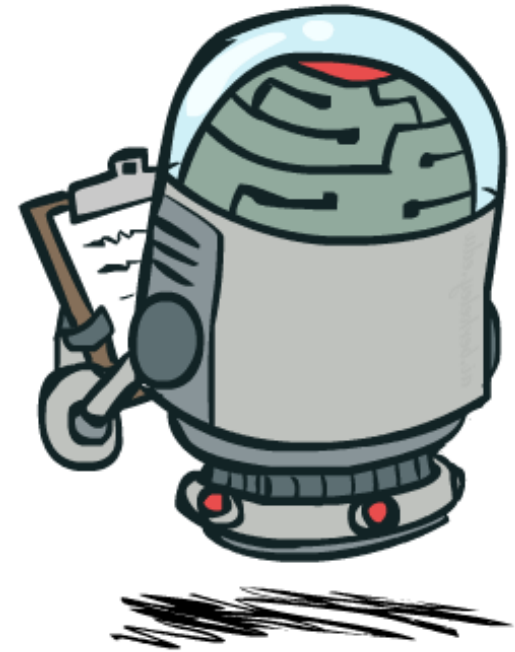
Administrivia

ANNOUNCEMENTS

- Office hours increased!
- Small change to practice problem scoring to allow 2 sick days
- You may work with friends (or come to office hours and work with new friends!) – but makes sure you can understand and complete everything you turn in yourself.

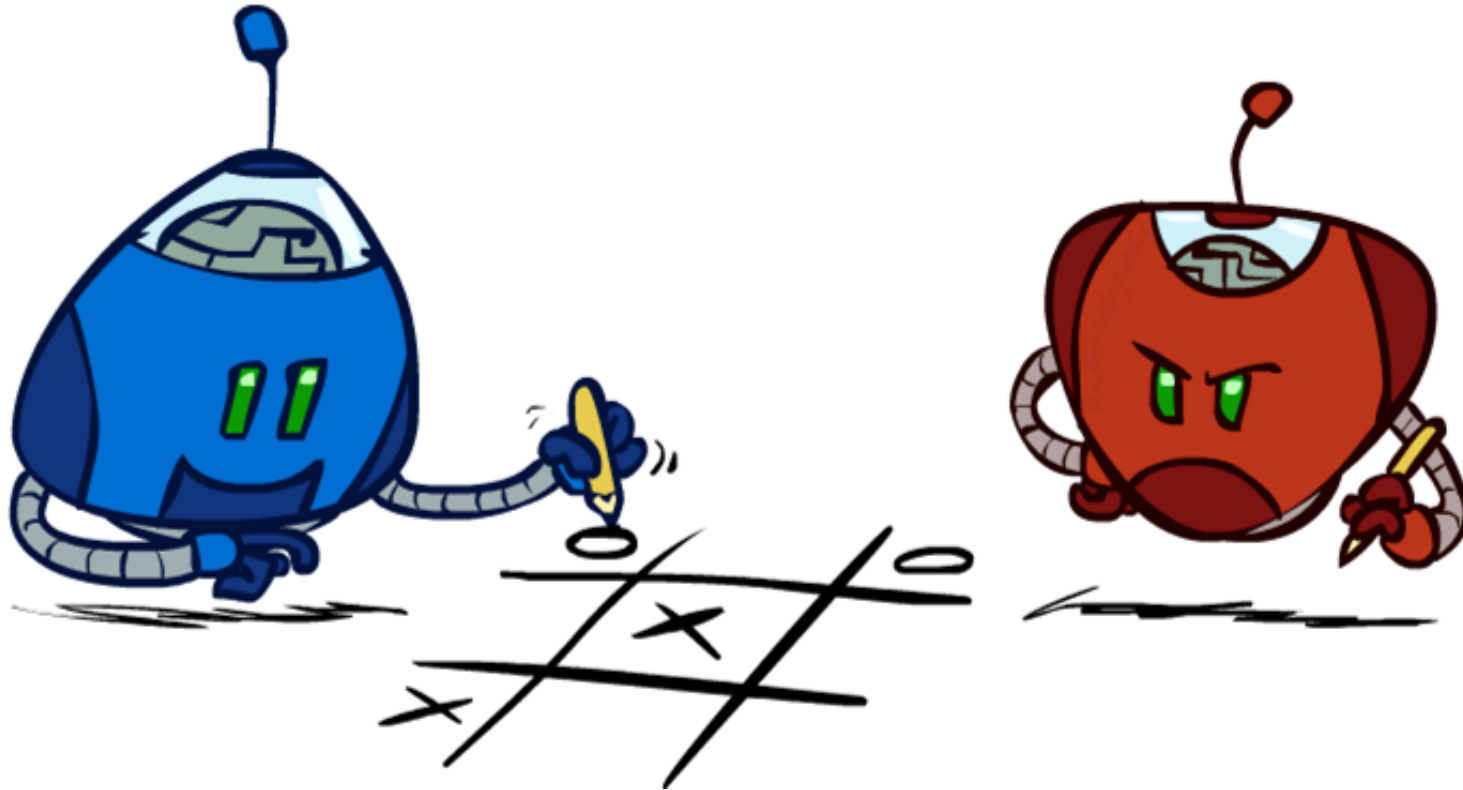
ASSIGNMENTS

- **Homework 1** due Monday (10.12)
- **Project 1** due next Monday (10.19)



Games

FOUNDATIONS



Types of Games

FOUNDATIONS

Games model environments with more than one agent

- Characteristics
 - deterministic vs stochastic?
 - fully observable vs. partially observable?
 - number of players?
 - turn-taking vs. simultaneous?
 - zero sum?
- *How to plan around opponent's moves?*



A brief history



Checkers:

- 1950: First computer player.
- 1959: Samuel's self-taught program.
- 1994: First computer world champion: Chinook defeats Tinsley
- 2007: Checkers solved! Endgame database of 39 trillion states

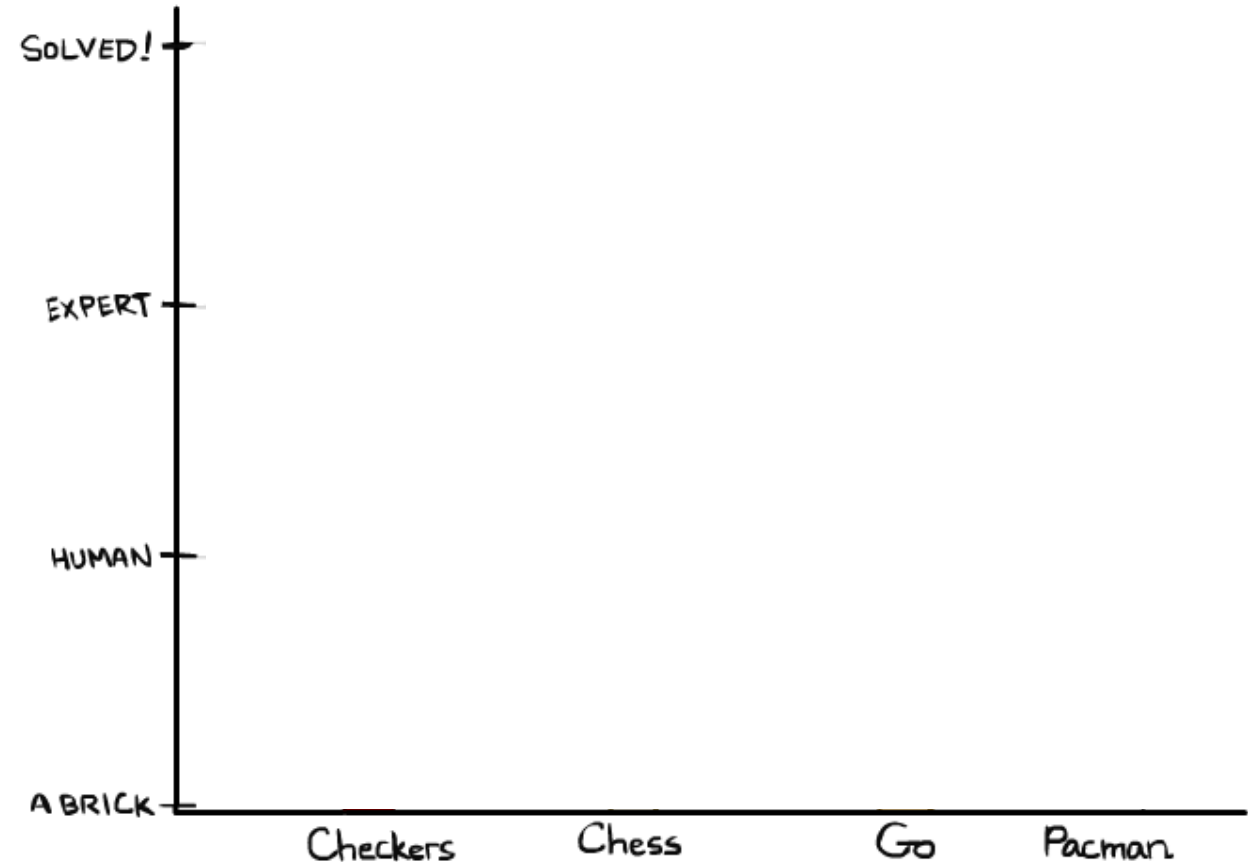
Chess:

- 1945-1960: Zuse, Wiener, Shannon, Turing, Newell & Simon, McCarthy.
- 1960s onward: gradual improvement under "standard model"
- 1997: Deep Blue defeats human champion Gary Kasparov
- 2021: Stockfish rating 3551 (vs 2870 for Magnus Carlsen).

Go:

- 1968: Zobrist's program plays legal Go, barely (b>300!)
- 1968-2005: various ad hoc approaches tried, novice level
- 2005-2014: Monte Carlo tree search -> strong amateur
- 2016-2017: AlphaGo defeats human world champions

Pacman



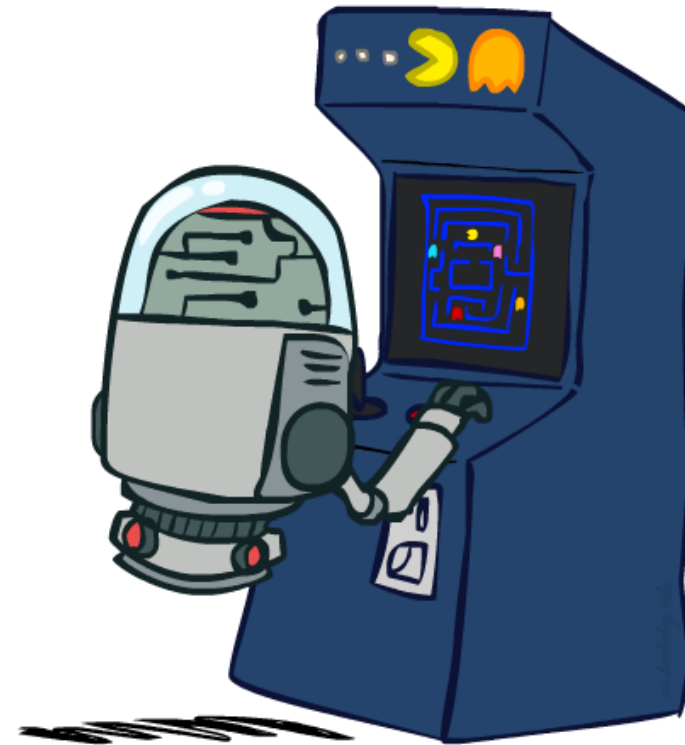
“Standard” Games

DEFINITION

Deterministic, observable, two-player, turn-taking, and zero-sum games

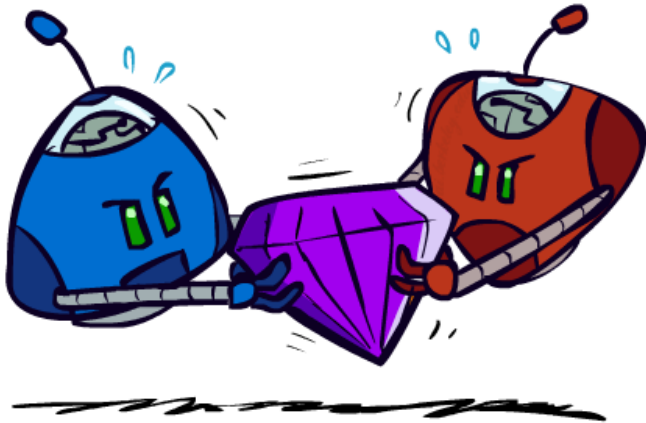
- Game Formulation

- Initial state: s_0
- Players: **player(s)** indicating turn
- Actions: **actions(s)** for player on move
- Transition model: **T(s, a)**
- Terminal (goal) test: **terminal-test(s)**
- Terminal values: **utility(s, p)**



Zero-Sum Games

FOUNDATIONS



Zero-Sum Games

- Agents have *opposite* utilities
- One agent **maximizes**, the other **minimizes**

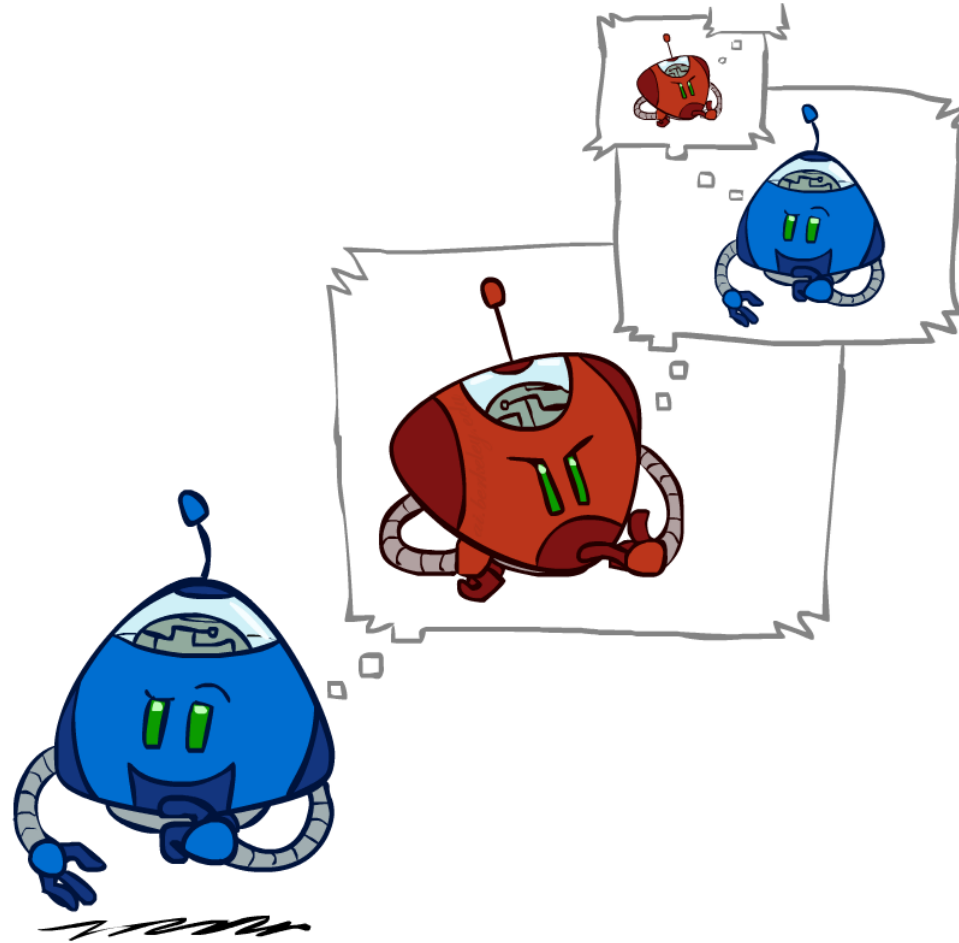


General Games

- Agents have *independent* utilities
- Cooperation, indifference, competition, etc. are all possible

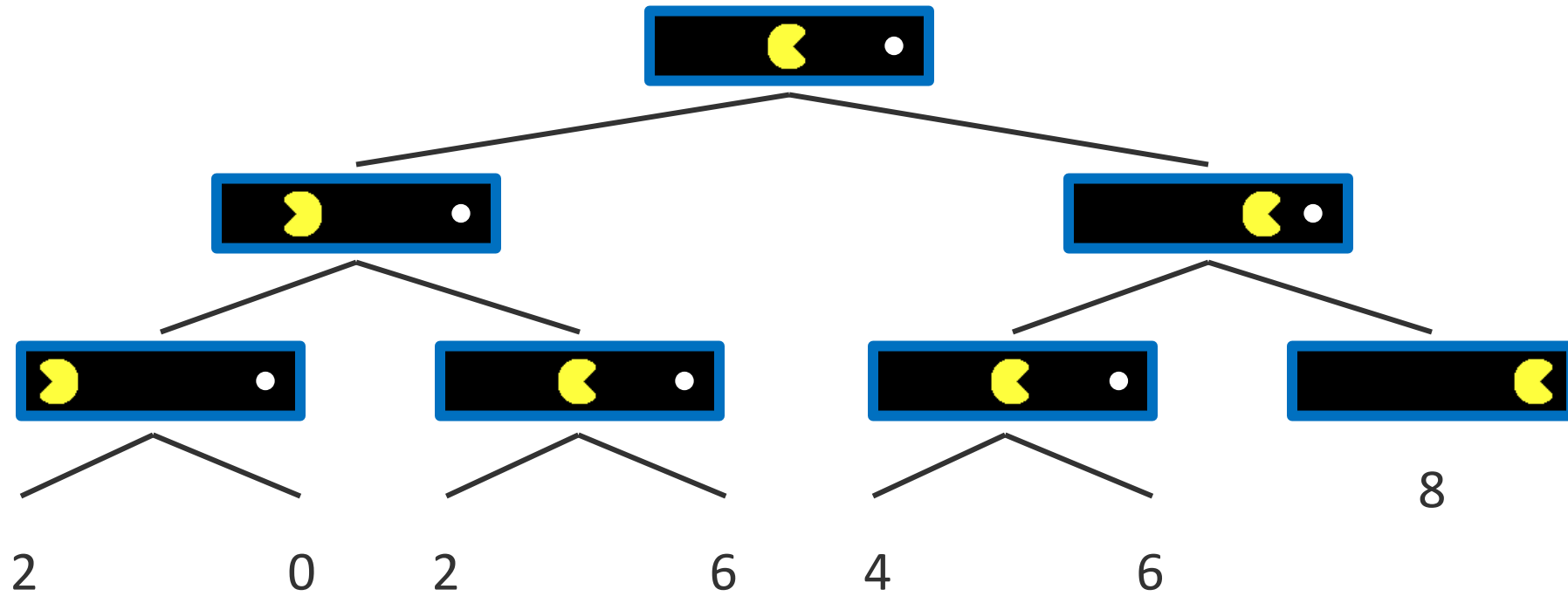
Adversarial Thinking

FOUNDATIONS



Single-Agent Trees

FOUNDATIONS

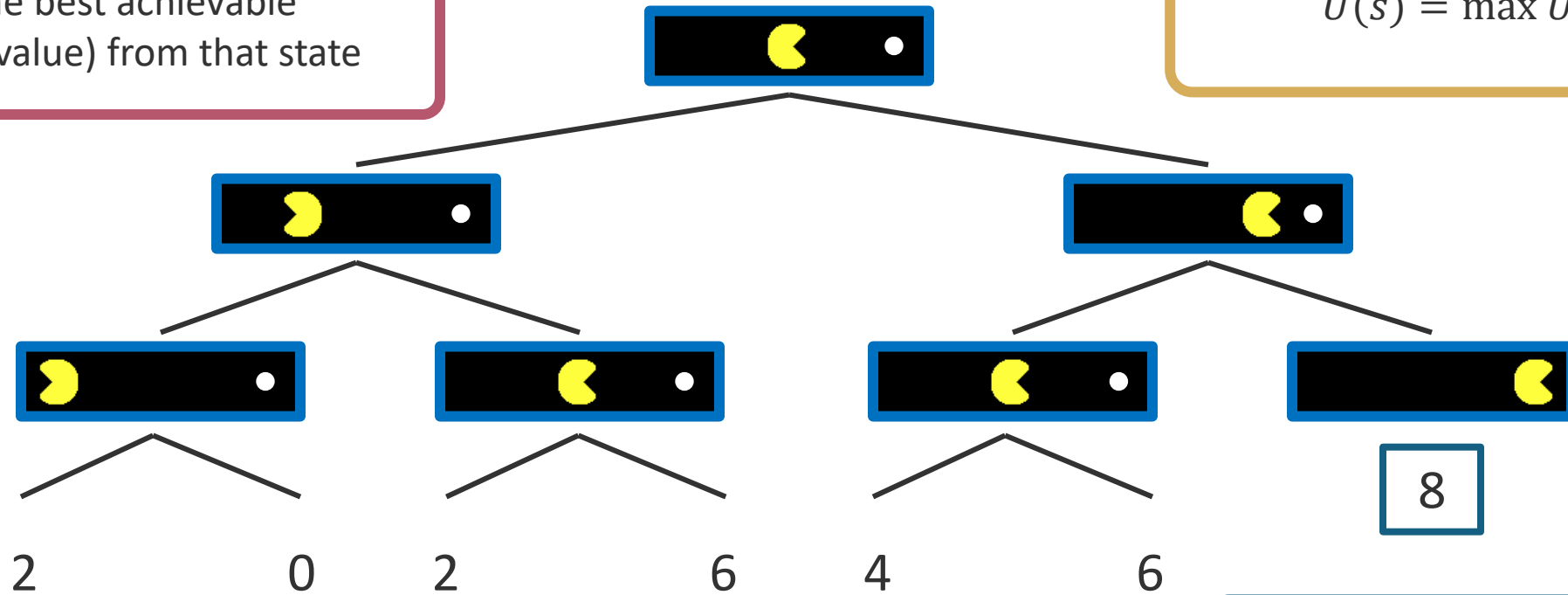


Utility

FOUNDATIONS

Utility is the best achievable outcome (value) from that state

Non-Terminal States:
 $U(s) = \max U(s')$



Terminal States:
 $U(s)$ is known

Tic-Tac-Toe

EXAMPLE



MAX (X)



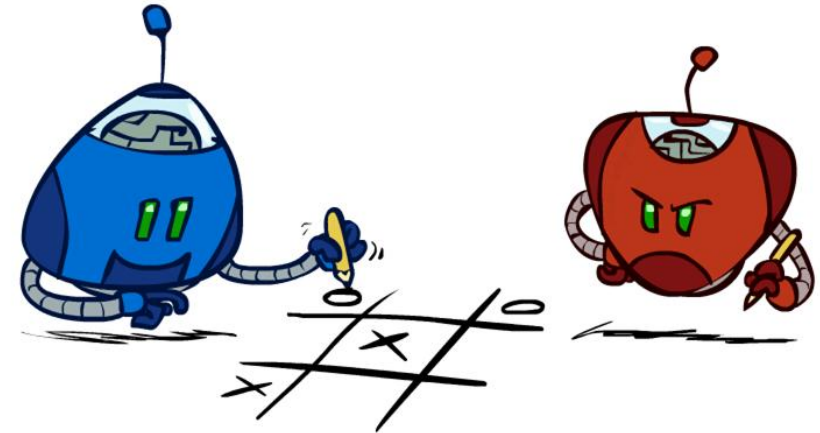
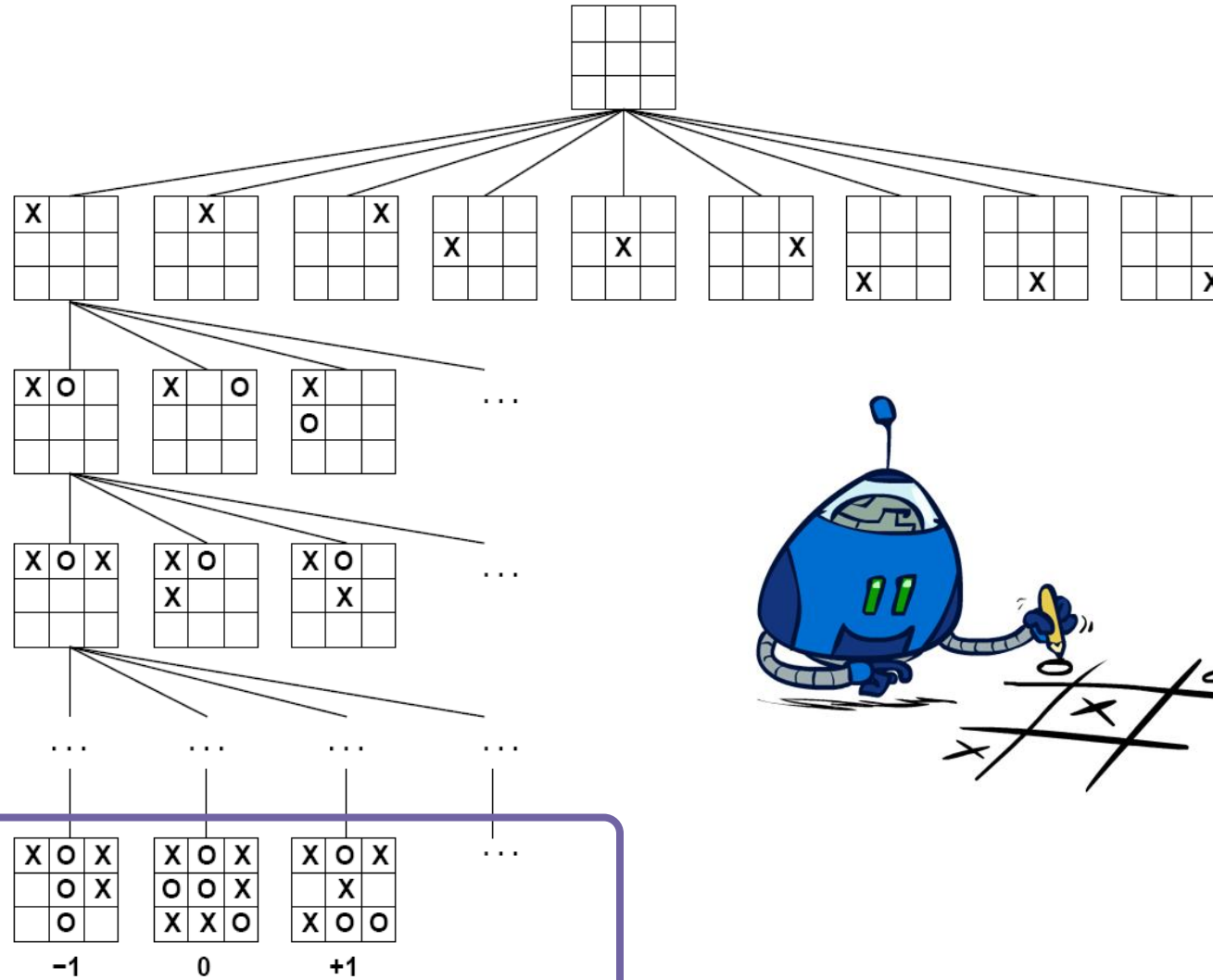
MIN (O)



MAX (X)

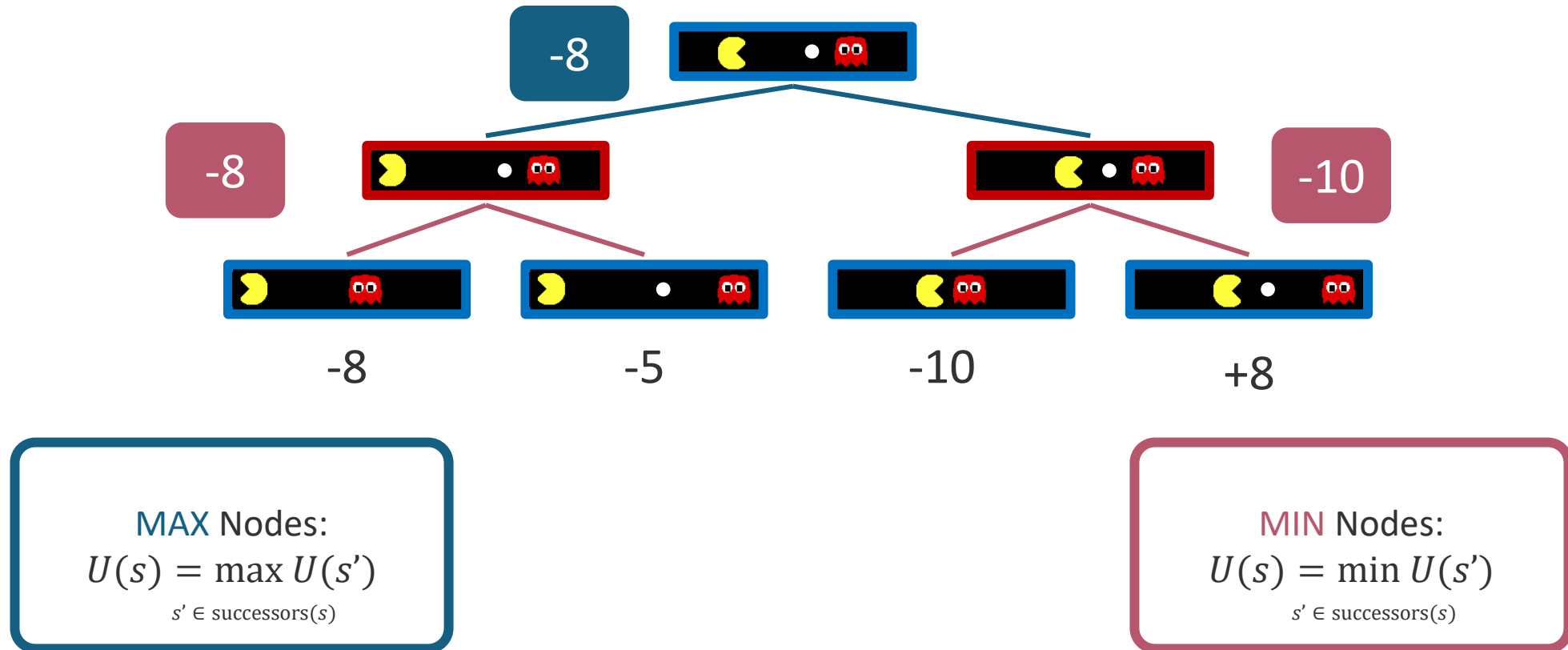


MIN (O)



Minimax Values

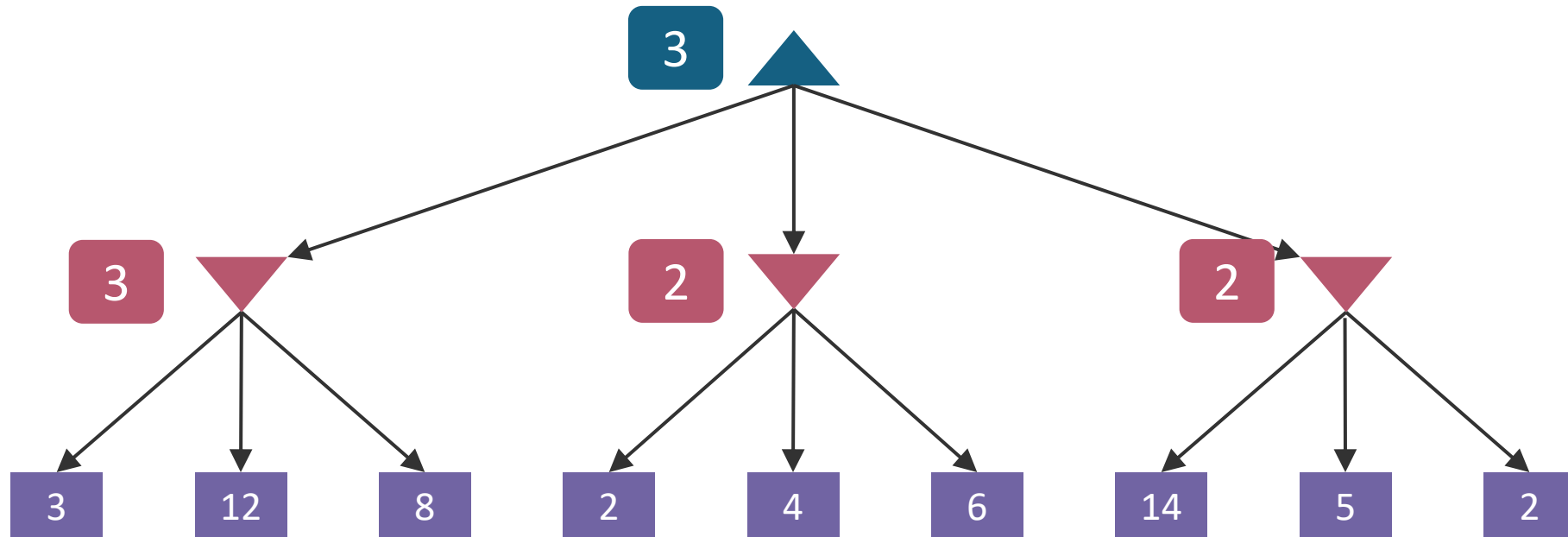
FOUNDATIONS



Minimax Tree



EXAMPLE



Minimax



ALGORITHM

```
function MINIMAX(s) returns action
  return action a in actions(s) with highest minimax-values(T(s,a))
```

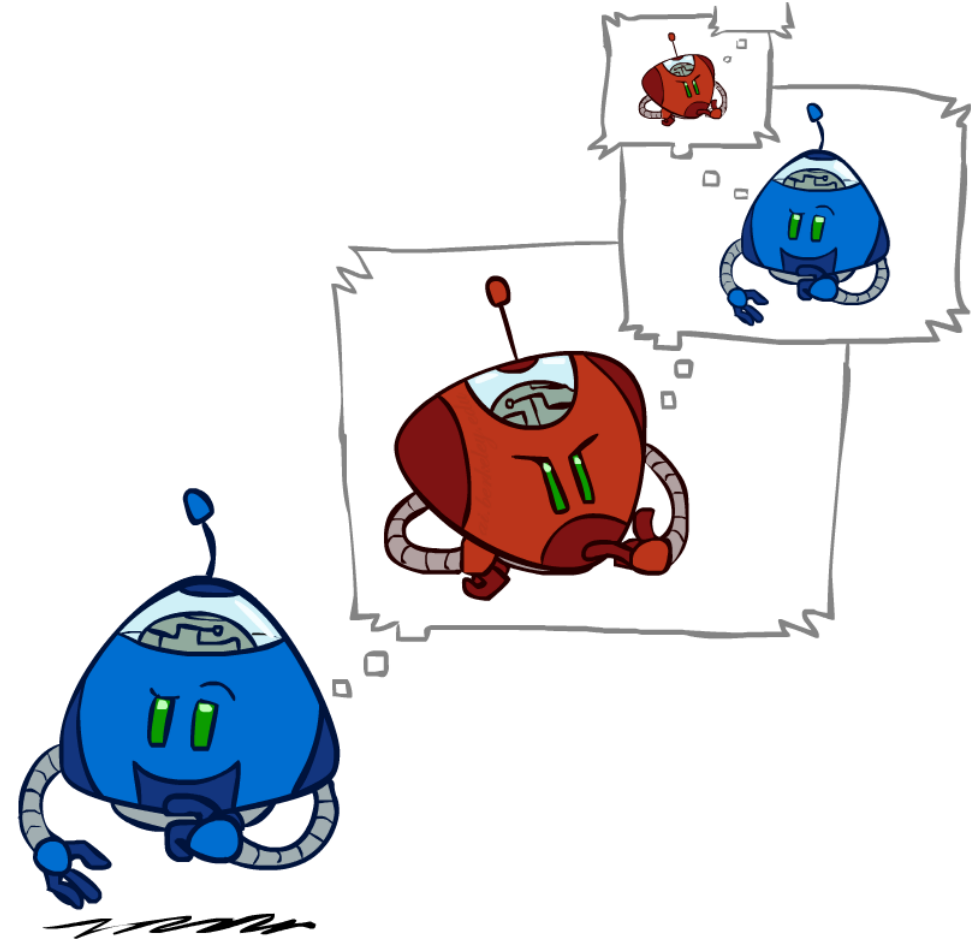
```
function MINIMAX-VALUES(s) returns value
  if terminal-test(s) then return utility(s)
  if player(s) == MAX then return maxa in actions(s) minimax-values(T(s,a))
  if player(s) == MIN then return mina in actions(s) minimax-values(T(s,a))
```

Questions?

Is It Efficient?

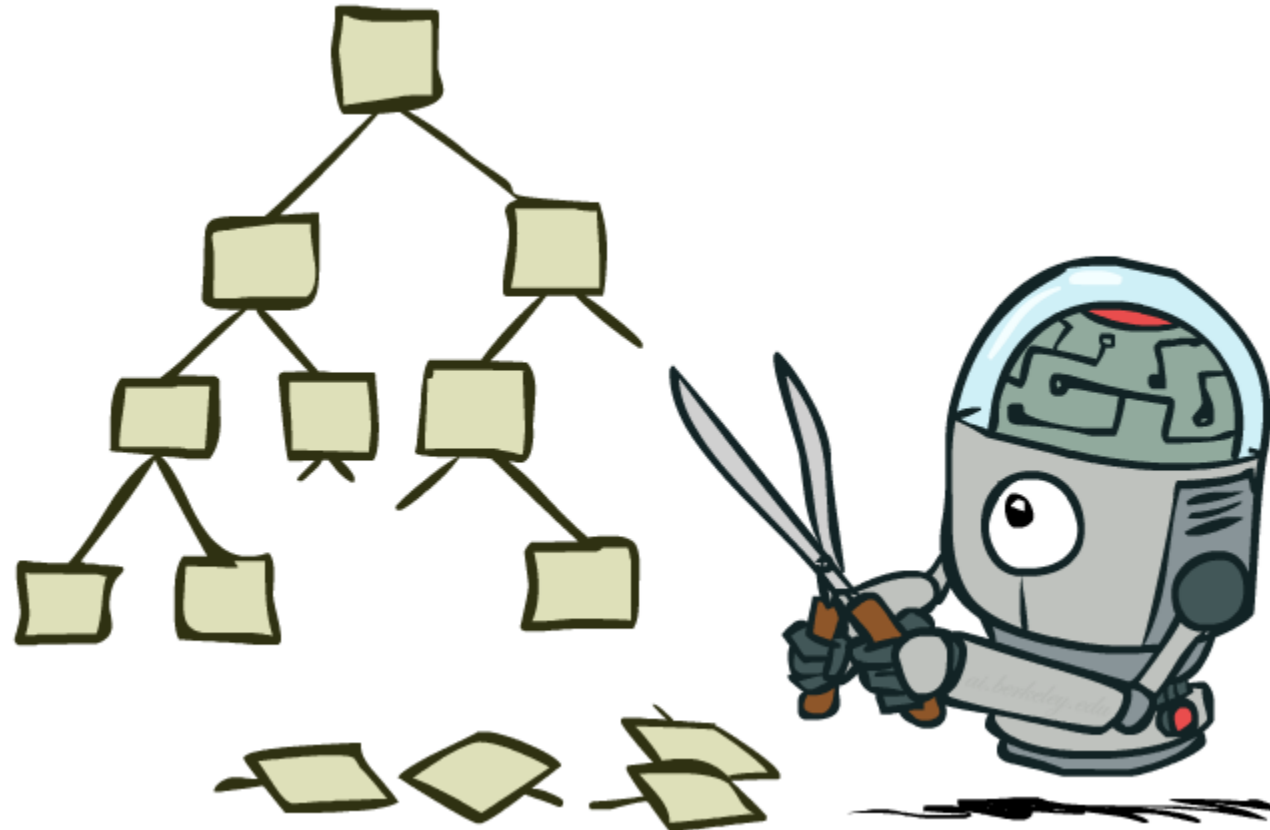
FOUNDATIONS

- How efficient is minimax?
 - Operates just like (exhaustive) DFS
 - Time Complexity: $O(b^m)$
 - Space Complexity: $O(bm)$
- Example: Chess
 - $b \approx 35$, $m \approx 100$
 - Completely infeasible
 - *Is this how humans play chess?*



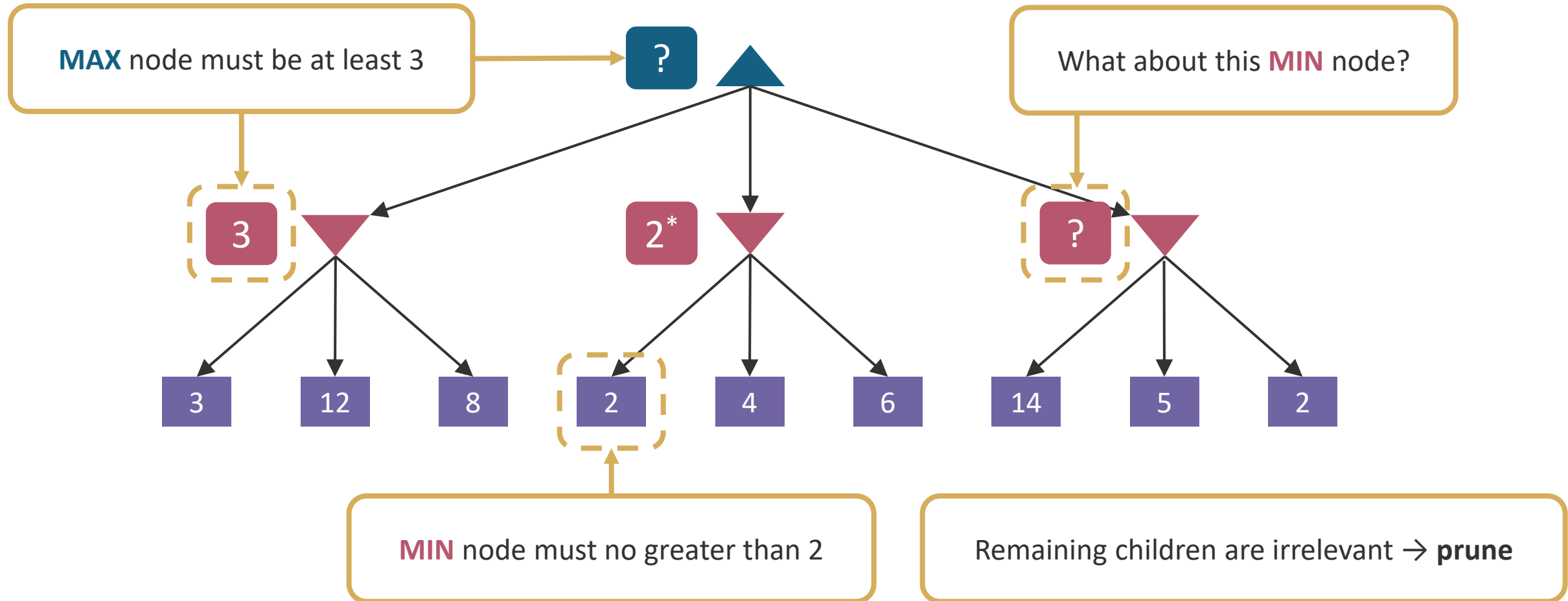
Avoiding Unnecessary Work

CONTEXT



Pruning

EXAMPLE

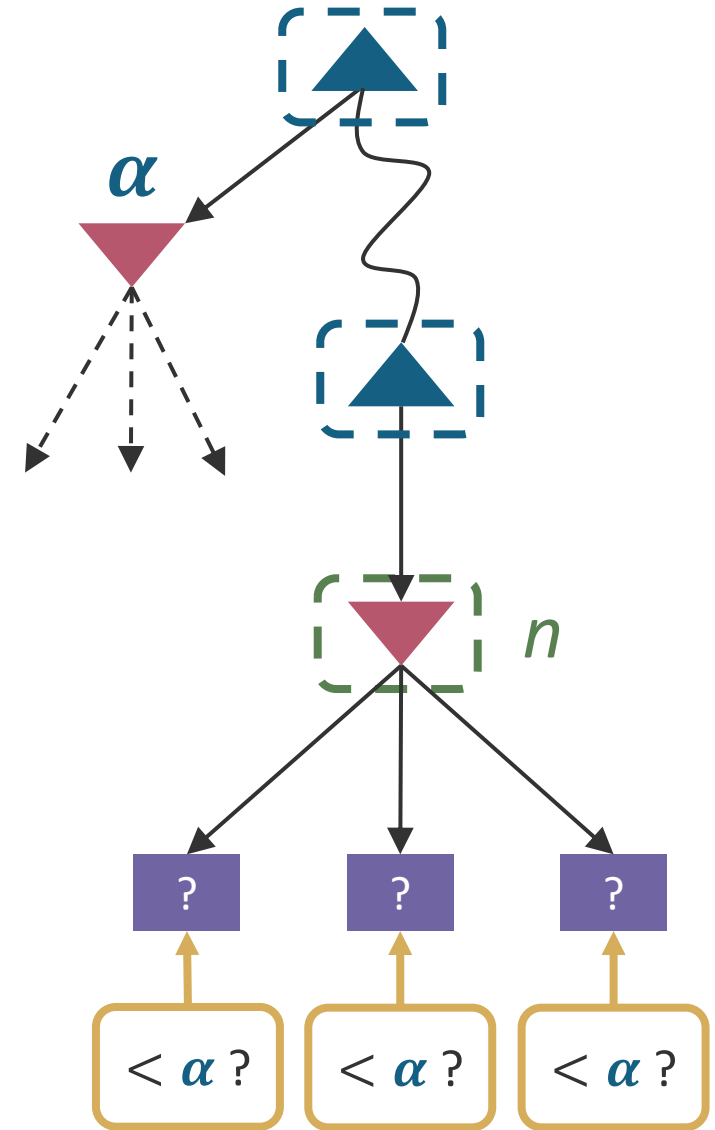


Alpha-Beta Pruning

FOUNDATIONS

W

- Idea
 - Avoid extra work by **pruning** branches when we know their utilities won't affect the decision
- General Configuration (**MIN** version)
 - To compute min-value at node n , loop over n 's children
 - n 's value is relevant for **MAX** node above
 - Let α be best value **MAX** can get at any point along current path from root
 - If n becomes worse than α , **MAX** will avoid it, so we can stop considering n 's children
- Version for **MAX** is symmetric (with β)



Alpha-Beta Implementation



ALGORITHM

```
function MINIMAX-DECISION( $s$ ) returns action
  return action  $a$  in actions( $s$ ) with highest max-value( $T(s, a)$ ,  $-\infty, +\infty$ )
```

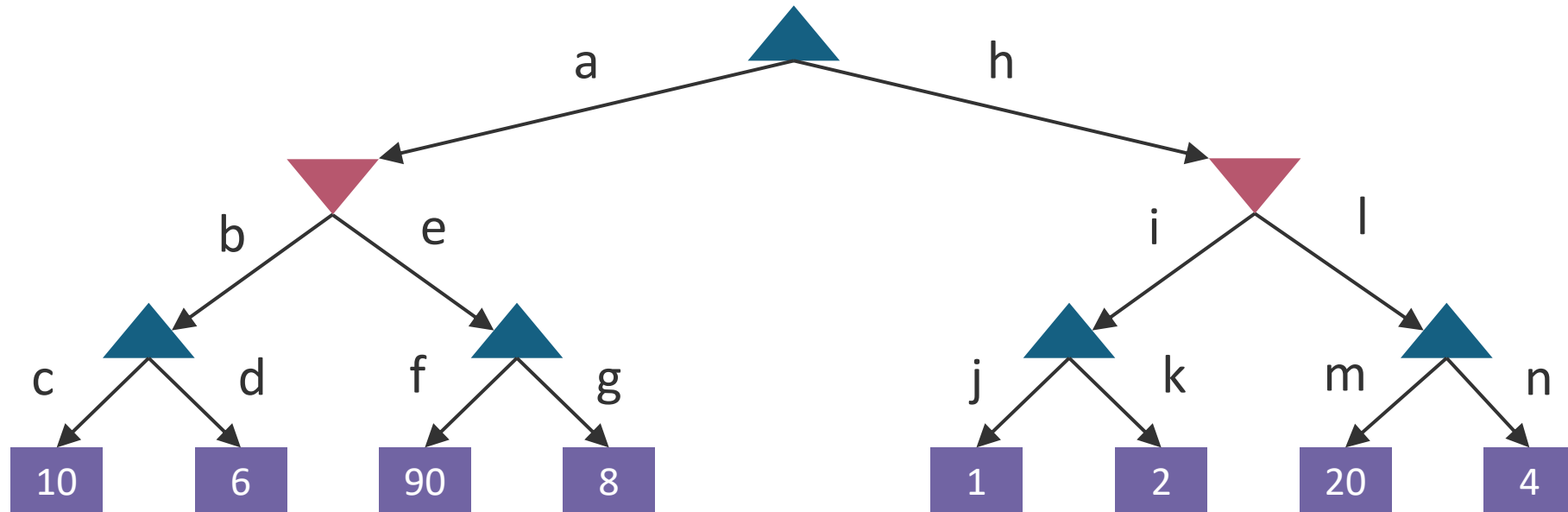
```
function MAX-VALUE( $s, \alpha, \beta$ ) returns value
   $v \leftarrow -\infty$ 
  for each  $s'$  in successors( $s$ )
     $v = \max(v, \text{min-value}(s', \alpha, \beta))$ 
    if  $v \geq \beta$ 
      return  $v$ 
   $\alpha = \max(\alpha, v)$ 
  return  $v$ 
```

```
function MIN-VALUE( $s, \alpha, \beta$ ) returns value
   $v \leftarrow +\infty$ 
  for each  $s'$  in successors( $s$ )
     $v = \min(v, \text{max-value}(s', \alpha, \beta))$ 
    if  $v \leq \alpha$ 
      return  $v$ 
   $\beta = \max(\beta, v)$ 
  return  $v$ 
```

Pruning Example

EXAMPLE

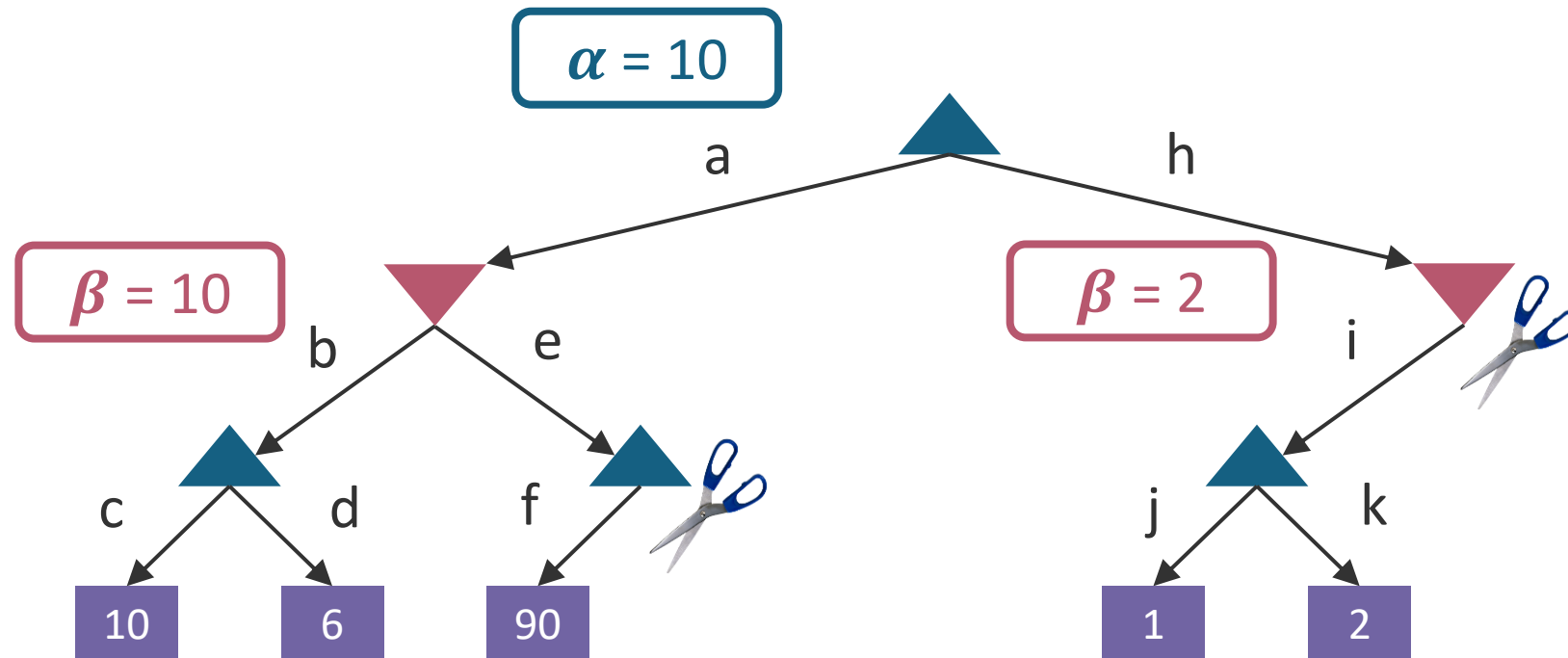
Which nodes get pruned? What are the α and β values at the end?



Pruning Example (Solution)

EXAMPLE

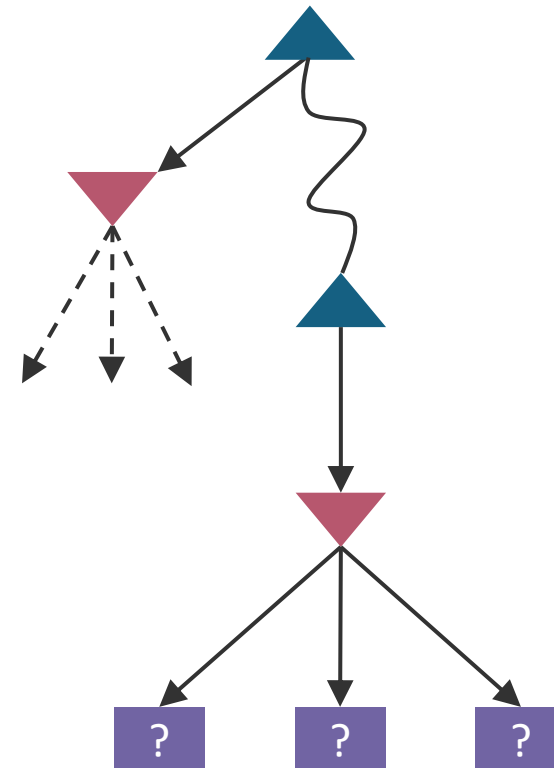
Which nodes get pruned? What are the α and β values?



Properties of Alpha-Beta Pruning

DEFINITION

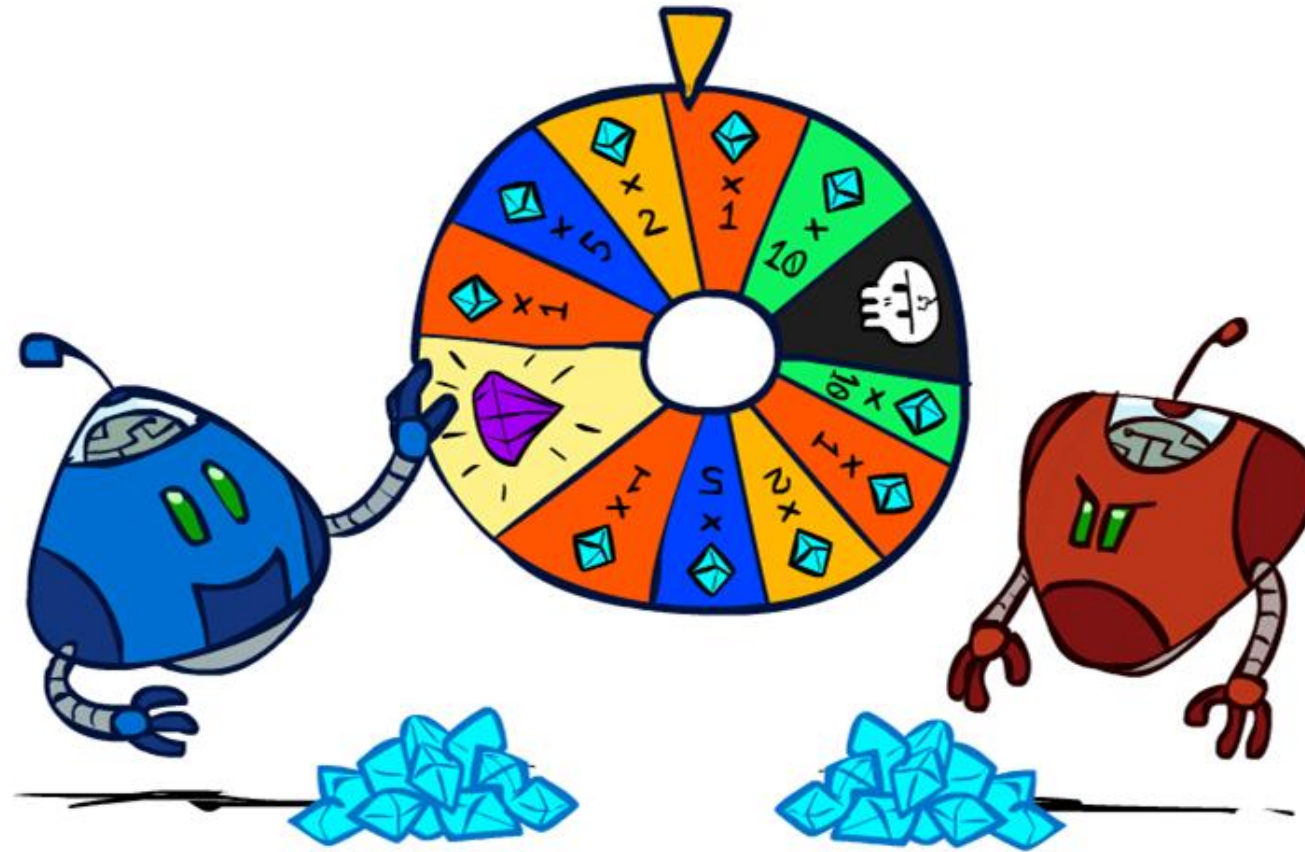
- Theorem: Alpha-Beta pruning has *no effect* on computed minimax value
- Good child ordering improves efficiency
- With “perfect ordering”:
 - Time complexity drops to $O(b^{m/2})$
 - Doubles the solvable depth!
- Example of **metareasoning**
 - Reasoning about reasoning
- For chess
 - Only 35^{50} instead of 35^{100}



Questions?

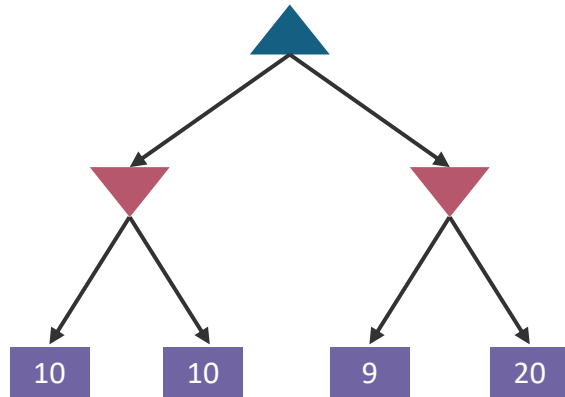
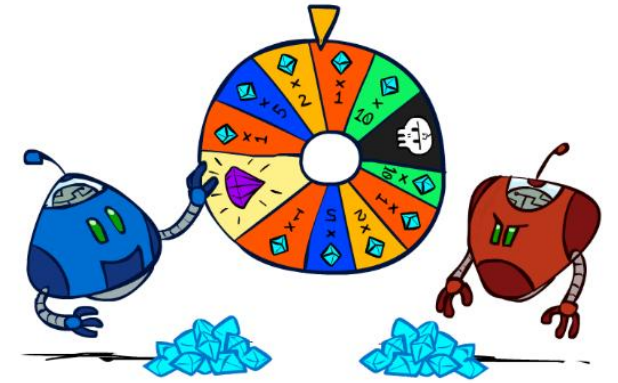
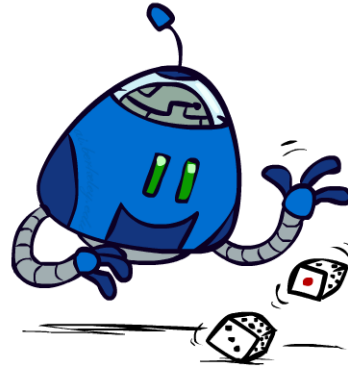
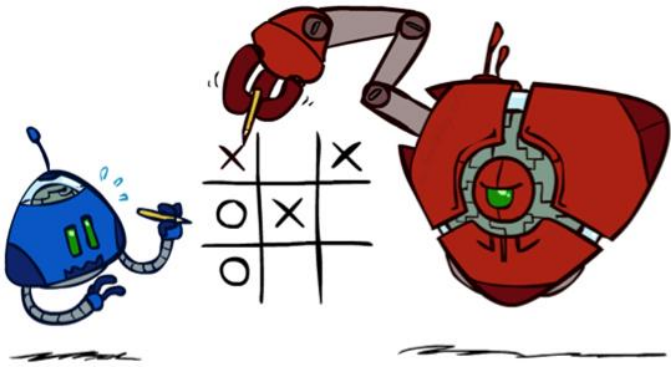
Games with Uncertainty

CONTEXT

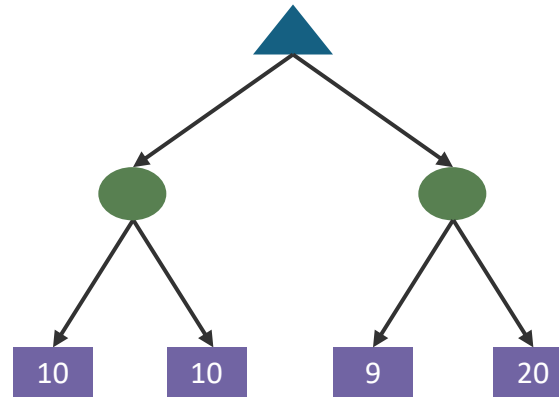


Chance Outcomes in Trees

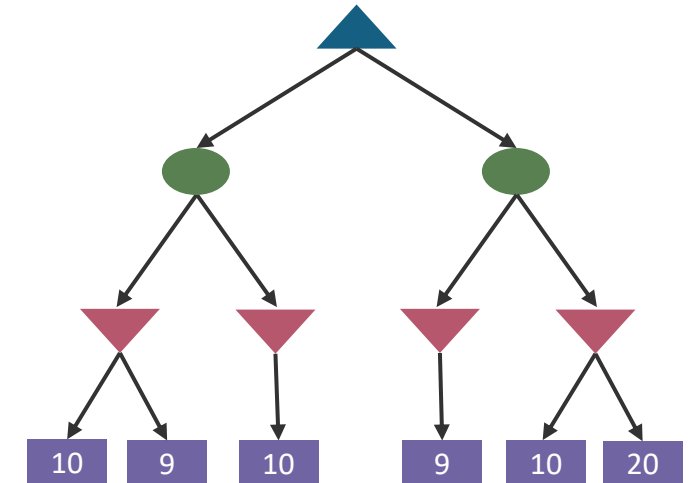
EXAMPLE



Minimax



Expectimax



Expectiminimax

Expectiminimax



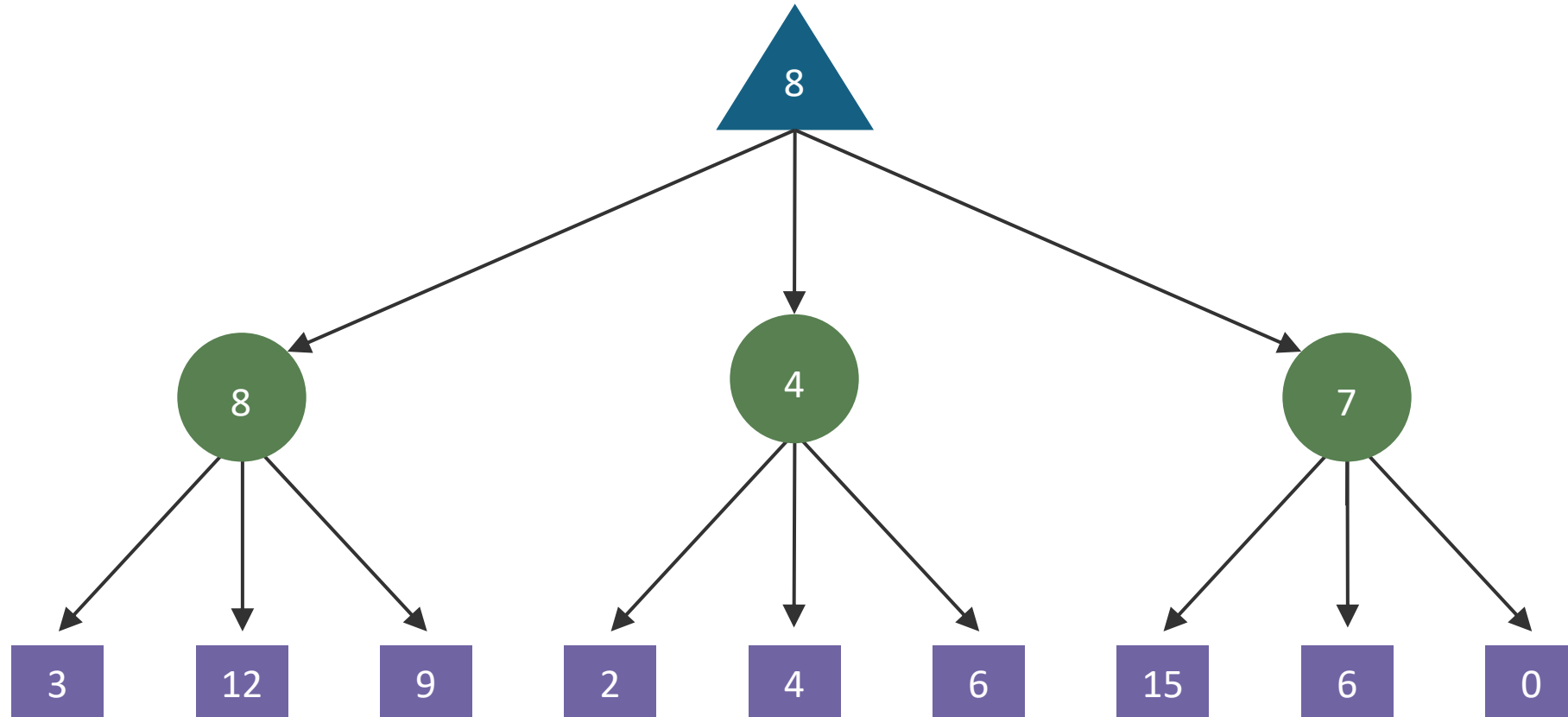
ALGORITHM

```
function DECISION( $s$ ) returns action
    return action  $a$  in actions( $s$ ) with highest value( $T(s, a)$ )
```

```
function VALUE( $s$ ) returns value
    if terminal-test( $s$ ) then return utility( $s$ )
    if player( $s$ ) == MAX then return max  $a$  in actions( $s$ ) value( $T(s, a)$ )
    if player( $s$ ) == MIN then return min  $a$  in actions( $s$ ) value( $T(s, a)$ )
    if player( $s$ ) == CHANCE then return sum  $a$  in actions( $s$ )  $P(a) * \text{value}(T(s, a))$ 
```

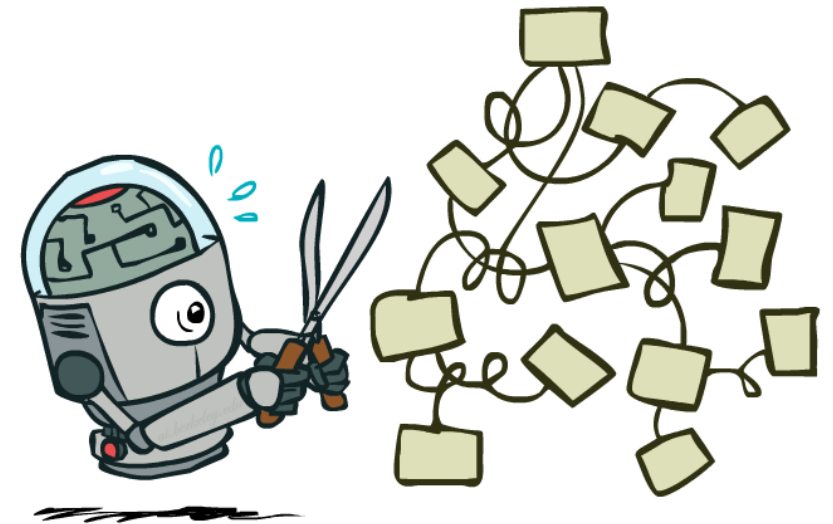
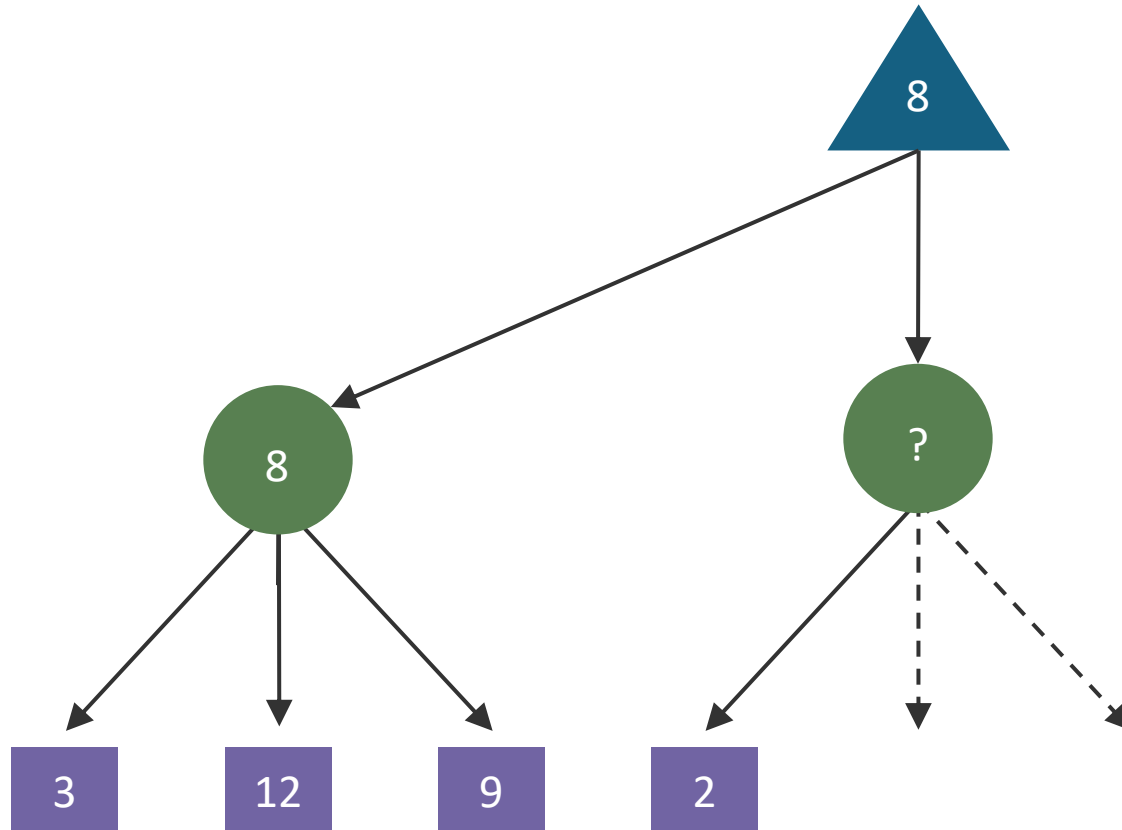
Expectimax

FOUNDATIONS



Expectimax Pruning?

EXAMPLE



Resource Limits

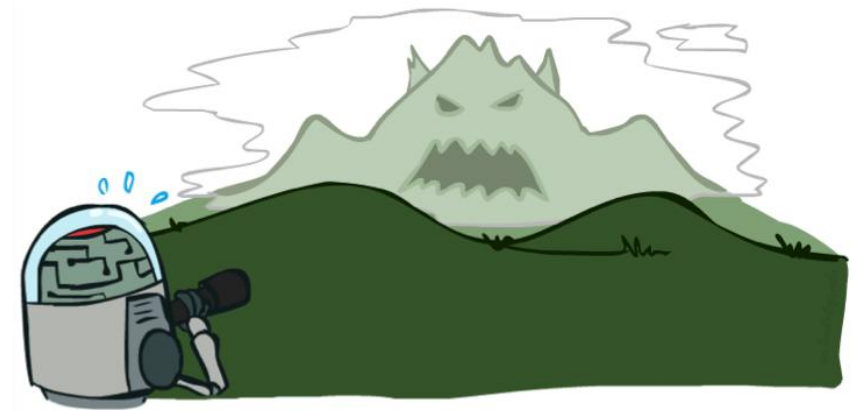
CONTEXT



Real-World Constraints

FOUNDATIONS

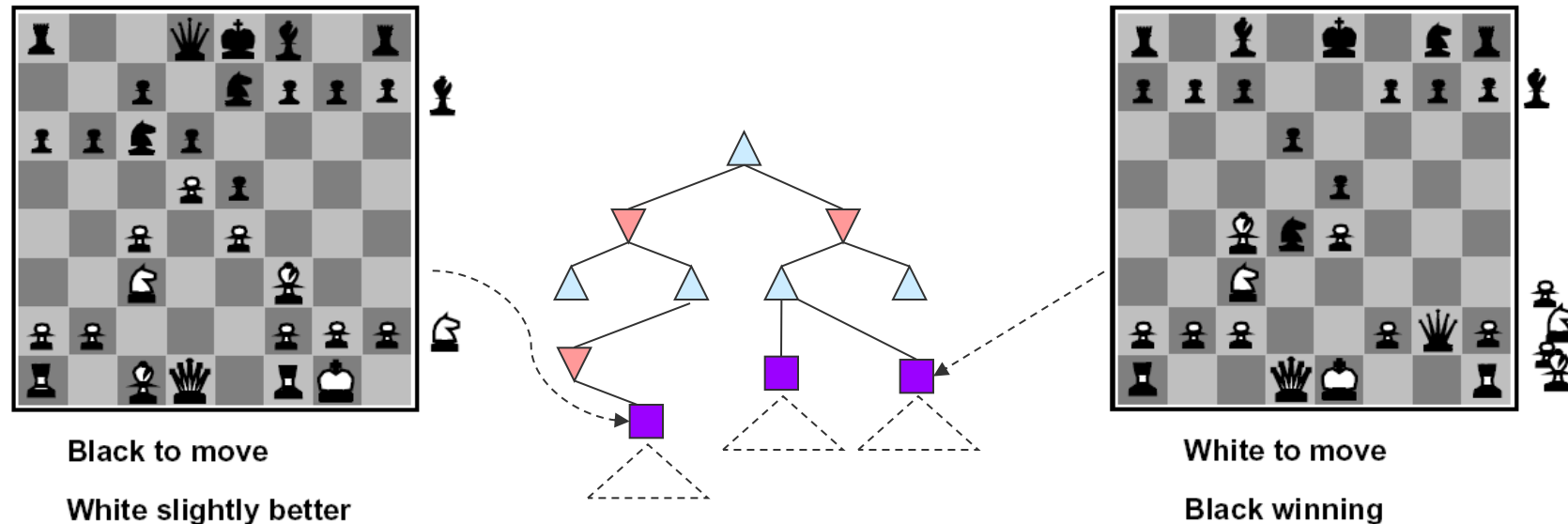
- **Problem:** In realistic games, searching to leaves is infeasible
- **Solution: Bounded Lookahead**
 - Search only to a preset **depth limit** (**horizon**)
 - Use an **evaluation function** for non-terminal utilities
 - Guarantee of optimality is gone
 - Marginal increase in depth makes a big difference!
- **Tradeoff**
 - Deeper search (usually) leads to better play, but is considerably more costly
 - Complexity of features vs. complexity of computation



Evaluation Functions



- Evaluation functions score non-terminals in depth-limited search



- Typically weighted linear sum of features:
 - $EVAL(s) = w_1 f_1(s) + w_2 f_2(s) + \dots + w_n f_n(s)$
 - E.g., $w_1 = 9$, $f_1(s) = (\text{num white queens} - \text{num black queens})$, etc.
- Or a more complex nonlinear function (e.g., NeuralNet) trained by self-play RL
- Terminate search only in *quiescent* positions, i.e., no major changes expected in feature values

Questions?

that's it for today!

SUMMARY

- Multi-Agent Search
- Adversarial search via minimax
- When transitions are deterministic and players play optimally, pruning is possible

UPCOMING

- Modeling Stochasticity
- Reinforcement Learning

REMINDERS

- Do **Homework 1**
- Do **Project 1**