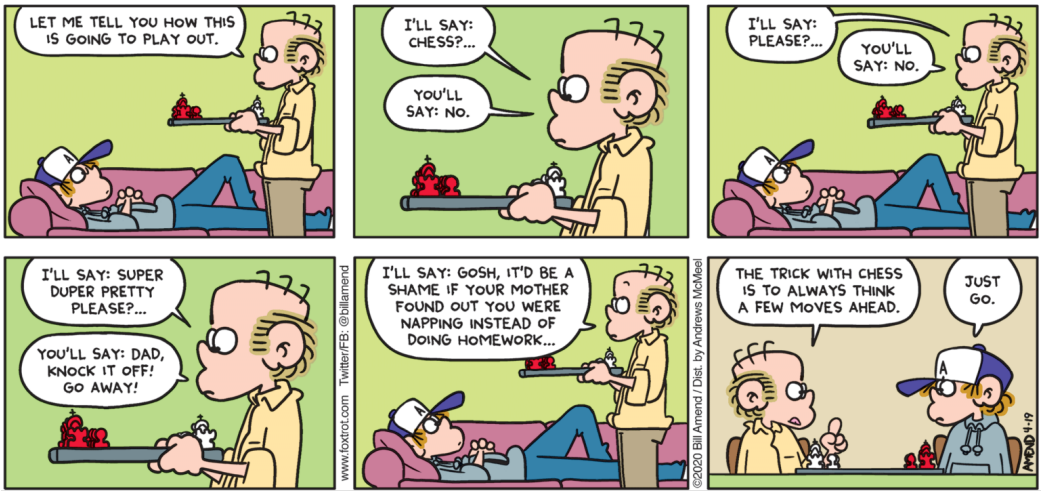
CSCI 3202

Lecture 20

October 13, 2025



Foxtrot by Bill Amend. https://foxtrot.com/

Announcements

- Midterm
  - Most students did not finish one or more problems on the midterm
  - We will grade the midterm and adjust the midterm grade by adding points to your score
    - The exact amount depends on the class averages
  - We are actively grading the midterms now
- Homework 5 is due tonight by 11:59 pm
- Homework 6 will be released on Wednesday
- Quiz 7 on Friday
  - Game Trees
  - Minimax
  - Alpha Beta pruning

Games

- We are studying 2 player, zero-sum games
- Each board position is a state
- We create a tree of all possible moves beginning at the start of the game

- We use an evaluation function to rate the strength of each state, positive means that we are winning and negative means we are losing
- We use a function from game theory called minmax() to maximize the value of our move and minimize the value of our opponent's move
- Then we search through the tree looking for the best value of the minmax() function, as far out as possible into the game
- Ideally, we would search to the end of the game, but we rarely can afford to search that far (or we don't have the patience)
- How far we can search into the game depends mainly on the amount of memory and the speed of our CPU
- That's it, but of course there are complications
- Games.pdf

Evaluation Functions

- Evaluation function examples:
- For tic-tac-toe
  - From cs.stackexchange.com
  - For non terminal positions, we use a linear evaluation function defined as

$$Eval(s) = 3 \cdot X_2(s) + X_1(s) - (3 \cdot O_2(s) + O_1(s))$$

    We define $X_n(s)$ as the number of rows, columns, and diagonals in state s with exactly n X's and no O's, and similarly define $O_n(s)$
- For Chess:
  - From Wikipedia (https://en.wikipedia.org/wiki/Evaluation_function)

An example handcrafted evaluation function for chess might look like the following:

- c_1 * material + c_2 * mobility + c_3 * king safety + c_4 * center control + c_5 * pawn structure + c_6 * king tropism + ...

Each of the terms is a weight multiplied by a difference factor: the value of white's material or positional terms minus black's.

- The material term is obtained by assigning a value in pawn-units to each of the pieces.
- Mobility is the number of legal moves available to a player, or alternately the sum of the number of spaces attacked or defended by each piece, including spaces occupied by friendly or opposing pieces. Effective mobility, or the number of "safe" spaces a piece may move to, may also be taken into account.
- King safety is a set of bonuses and penalties assessed for the location of the king and the configuration of pawns and pieces adjacent to or in front of the king, and opposing pieces bearing on spaces around the king.
- Center control is derived from how many pawns and pieces occupy or bear on the four center spaces and sometimes the 12 spaces of the extended center.
- Pawn structure is a set of penalties and bonuses for various strengths and weaknesses in pawn structure, such as penalties for doubled and isolated pawns.
- King tropism is a bonus for closeness (or penalty for distance) of certain pieces, especially queens and knights, to the opposing king.

- More recent versions of computer opponents use a neural network or something similar to evaluate a positions
- These networks improve their evaluation through reinforcement learning
  - The other components of the game are similar to the methods we are discussing in class

Question

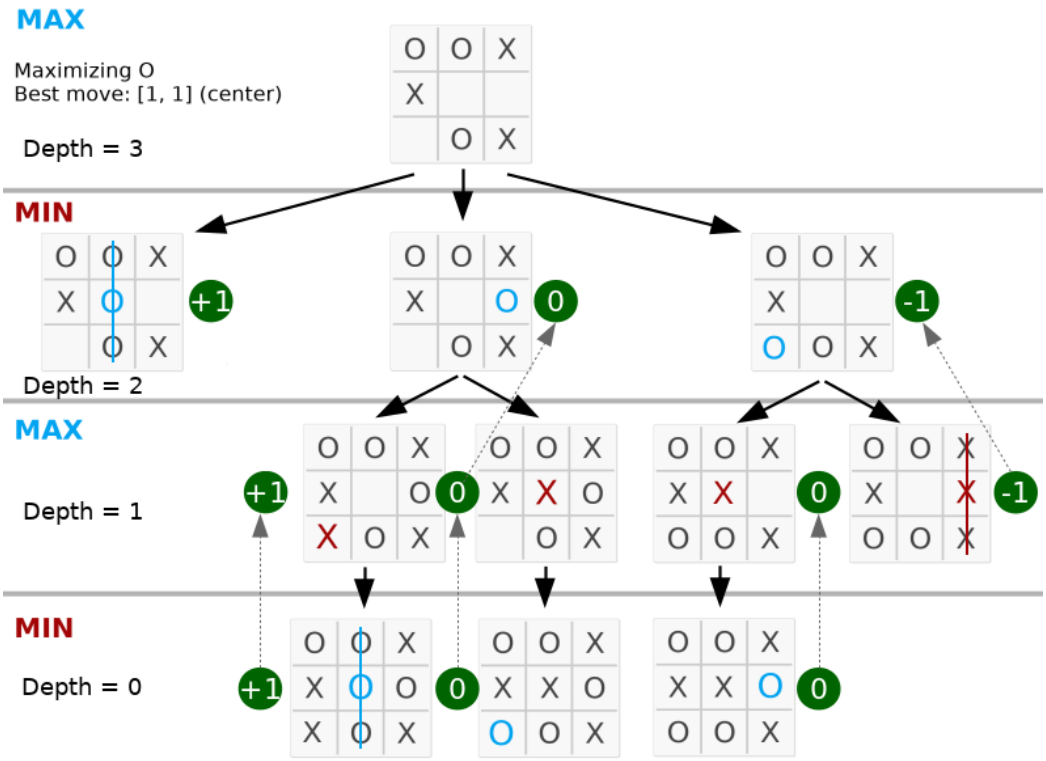
- Does the evaluation function have to return a unique value?
- What happens to the minimax result if we have ties?

An Interesting Evaluation Function

- For a game with a limited tree size we can use the following evaluation function **only at terminal nodes**
- If a node is not terminal, we don't evaluate it

$$f(x) = \begin{cases} 1 & \text{if terminal node is a win} \\ 0 & \text{if terminal node is a draw} \\ -1 & \text{if terminal node is a loss} \end{cases}$$

- We end up with a lot of ties



Alpha Beta pruning

- Minimax is a backtracking algorithm therefore we have to go through the entire tree
- The size of the tree grows exponentially with each layer
- There are some nodes or entire branches of the tree that will never be the result of minimax. We can stop evaluating these nodes or branches to save time

Upcoming

- Alpha-Beta pruning
- Project description on Friday
  - Default project is Mancala
  - Can create your own project as long as it uses one of the topics in our class
  - Can work in teams of 2 if desired