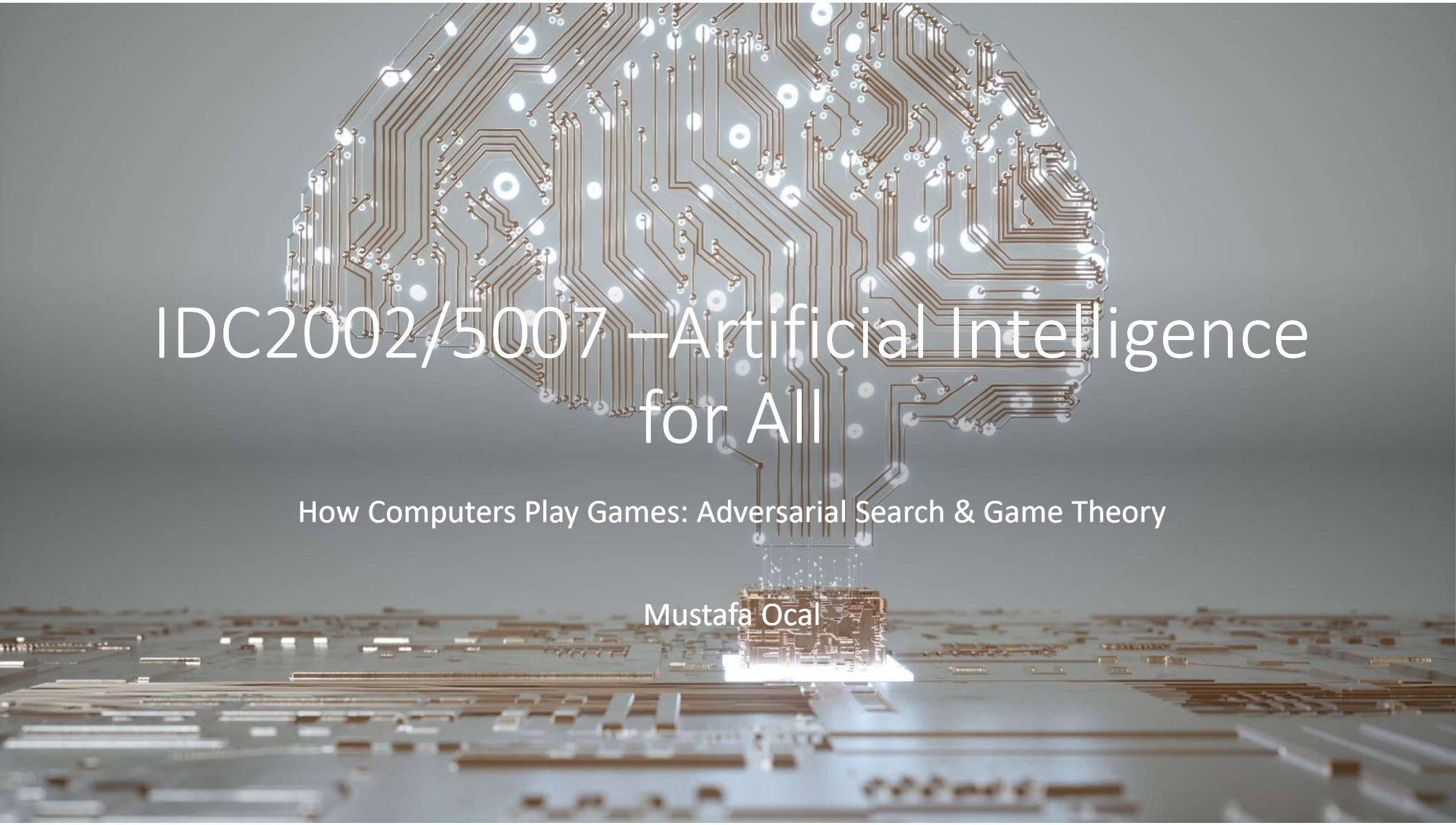


IDC2002/5007 –Artificial Intelligence for All

How Computers Play Games: Adversarial Search & Game Theory

Mustafa Ocal

Adversarial Search Problems

- Multiagent Environment
  - each agent needs to consider the actions of other agents and how they affect its own welfare.
  - The unpredictability of these other agents can introduce contingencies into the agent's problem-solving process
- Competitive Environment
  - The agents' goals are in conflict, giving rise to **adversarial search** problems—often known as **games**

Game Theory

- Two players
  - Max-min
  - Taking turns, fully observable
- Moves: Action
- Position: state
- Zero sum:
  - good for one player, bad for another
  - No win-win outcome.

Game Theory

- S_0: The initial state of the game
- TO-MOVE(s): player to move in state s.
- ACTIONS(s): The set of legal moves in state s.
- RESULT(s, a): The transition model, resulting state
- IS-TERMINAL(s): A terminal test to detect when the game is over
- UTILITY(s; p): A utility function (objective/payoff)

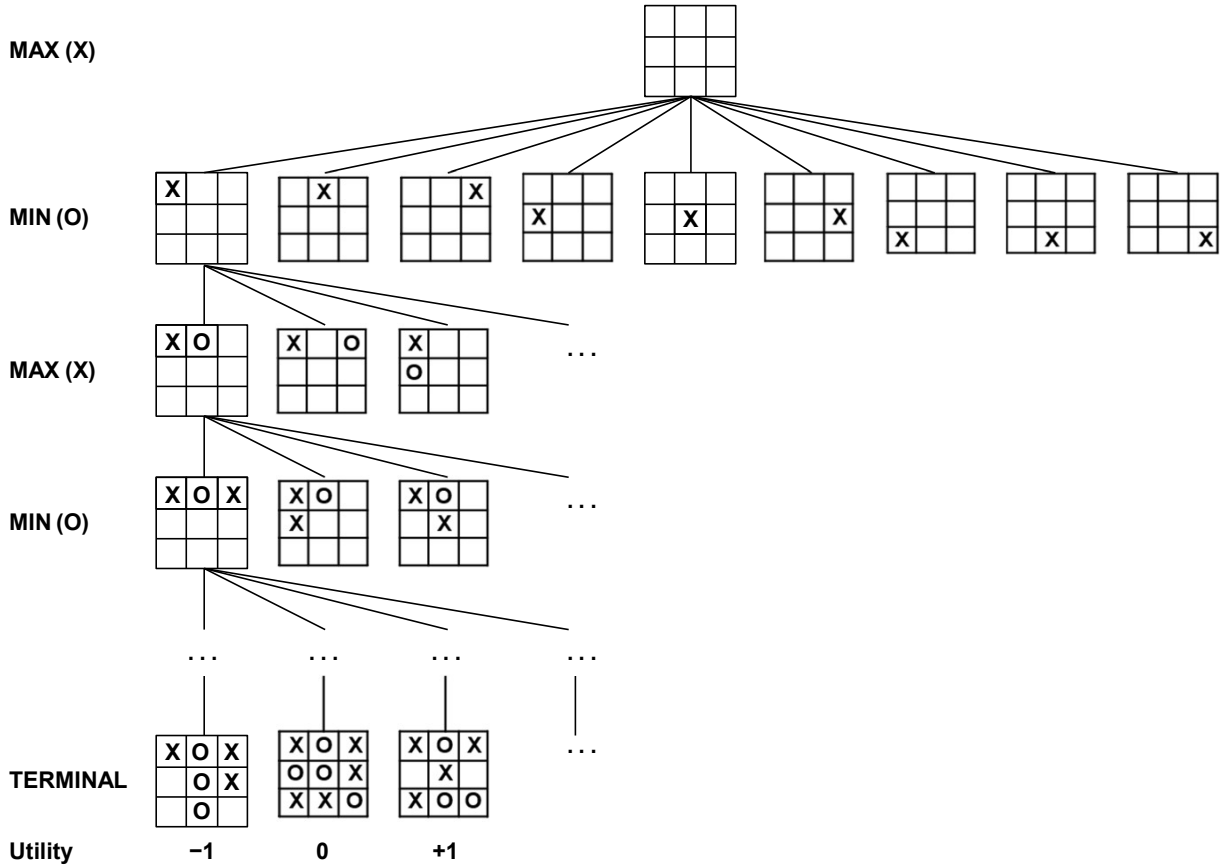
Games: Computer vs Human

• On May 11, 1997 IBM Deep Blue beat the world chess champion Garry Kasparov

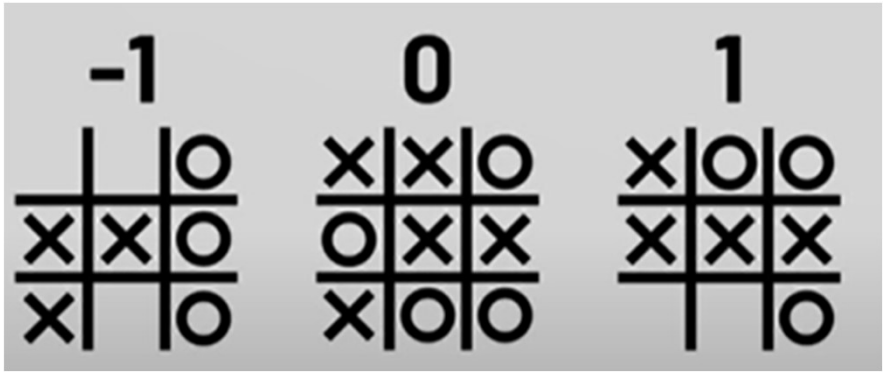
• "Unpredictable" opponent ⇒ solution is a strategy specifying a move for every possible opponent reply

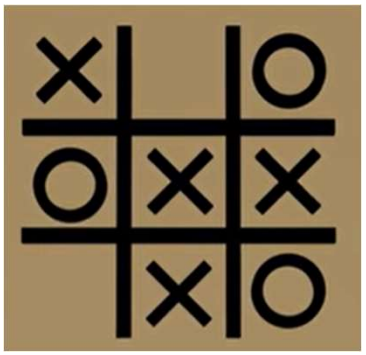
	deterministic	chance
perfect information	chess, checkers, go, othello	backgammon monopoly
imperfect information	battleships, blind tictactoe	bridge, poker, scrabble

Tic-Tac-Toe



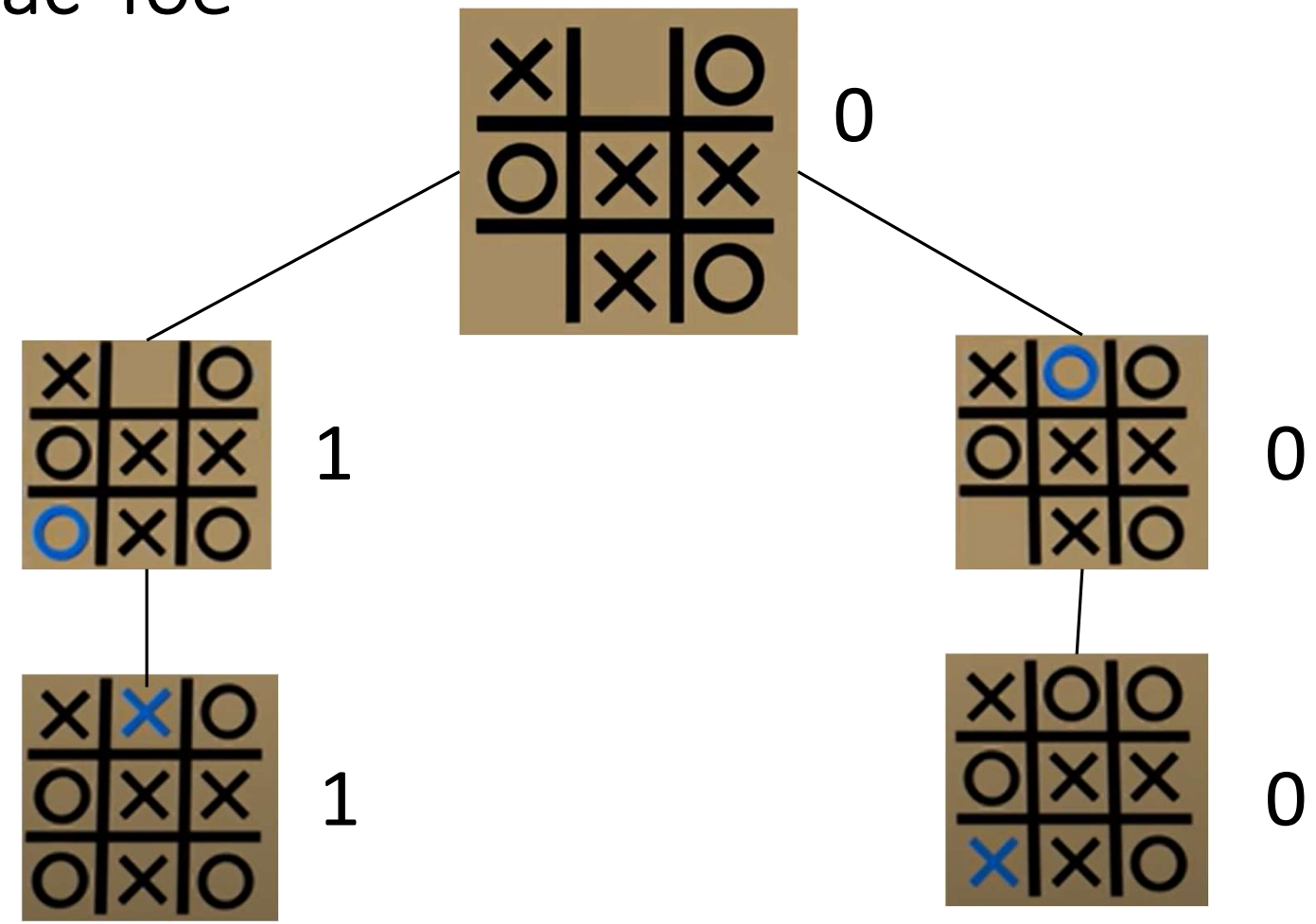
Tic-Tac-Toe





Possible moves?

Tic-Tac-Toe



Minimax Algorithm

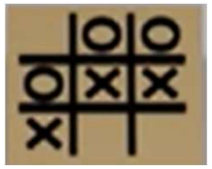
- The minimax algorithm is a decision-making algorithm used in two-player games, especially in the field of artificial intelligence and game theory.
- It's primarily employed in games with perfect information, where both players have full knowledge of the game state and can make informed decisions.
- The primary goal of the minimax algorithm is to determine the best possible move for a player, assuming that the opponent will also make the best possible moves to minimize the first player's outcome (hence the name "minimax")

Minimax Algorithm

Here's how the minimax algorithm works:

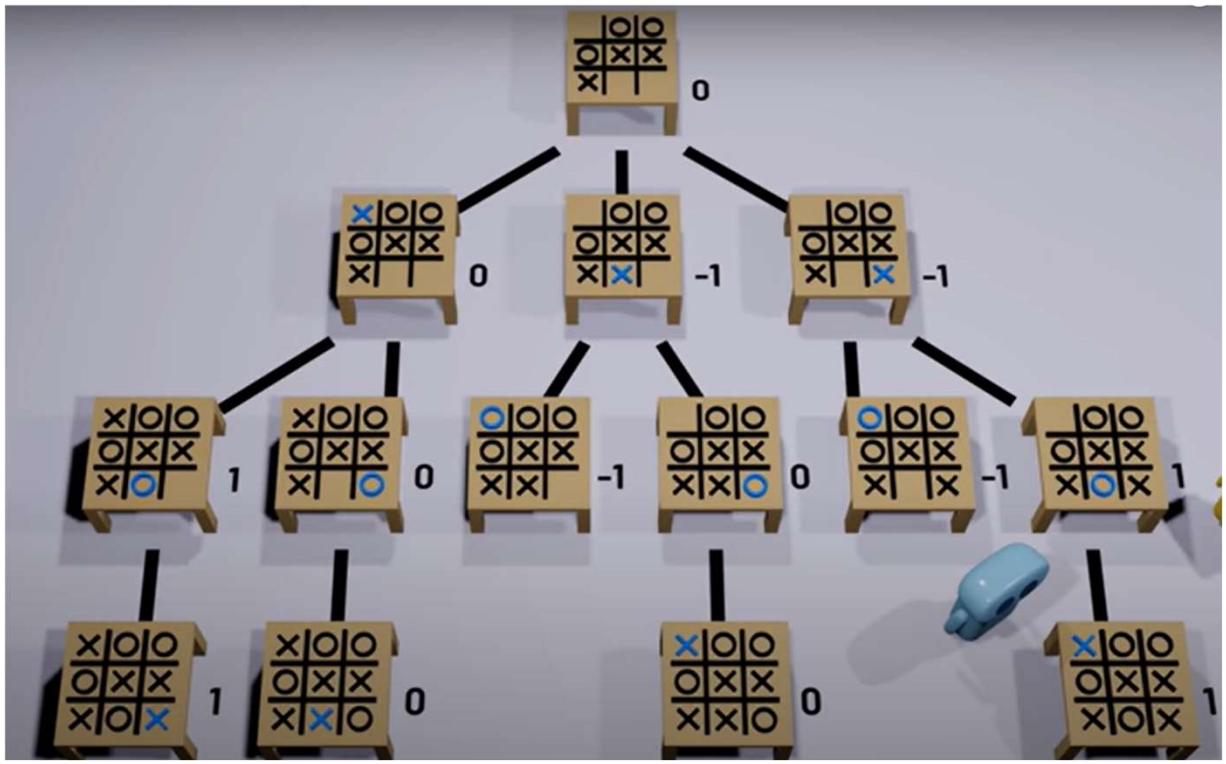
- **Tree of Game States:** The algorithm constructs a tree of possible game states, starting from the current state and branching out to all possible future states resulting from legal moves by both players. This tree is often called the game tree.
- **Evaluation Function:** At the terminal nodes of the tree (end of the game or a predefined depth), an evaluation function is applied to determine the "value" of that game state for the player. The evaluation function assigns a numerical score to each terminal node, representing how favorable or unfavorable the state is for the player.
- **Backpropagation:** The values of terminal nodes are then propagated up the tree. At each non-terminal node (i.e., a node representing a state where it's the player's turn to move), the algorithm calculates the best achievable outcome if both players play optimally. This involves taking the maximum value for the player whose turn it is to move (the "max" player) and the minimum value for the opponent (the "min" player) at each level of the tree alternately.
- **Decision:** Finally, the algorithm chooses the move that leads to the maximum value in the root node of the tree, assuming the opponent will minimize the player's outcome. This move is considered the best move to make in that game state.

Example: Tic-Tac-Toe

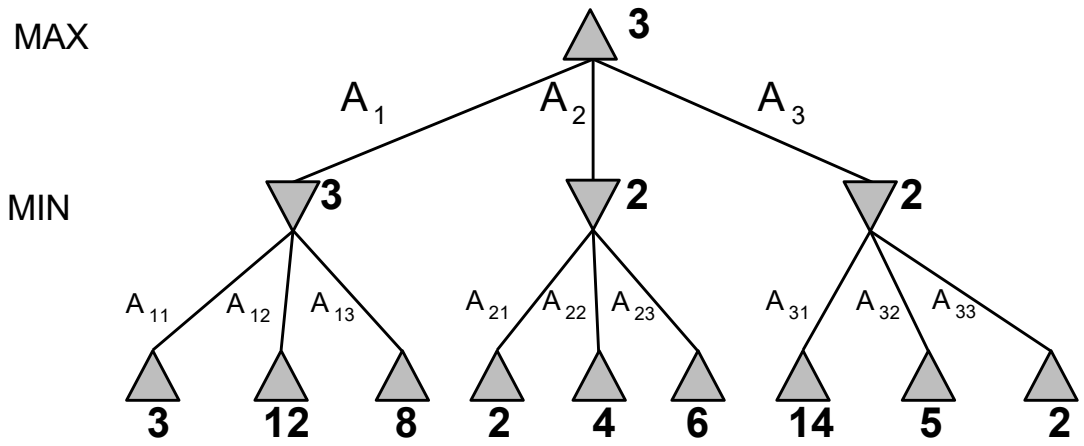


Root node
(Start state)
X’s turn

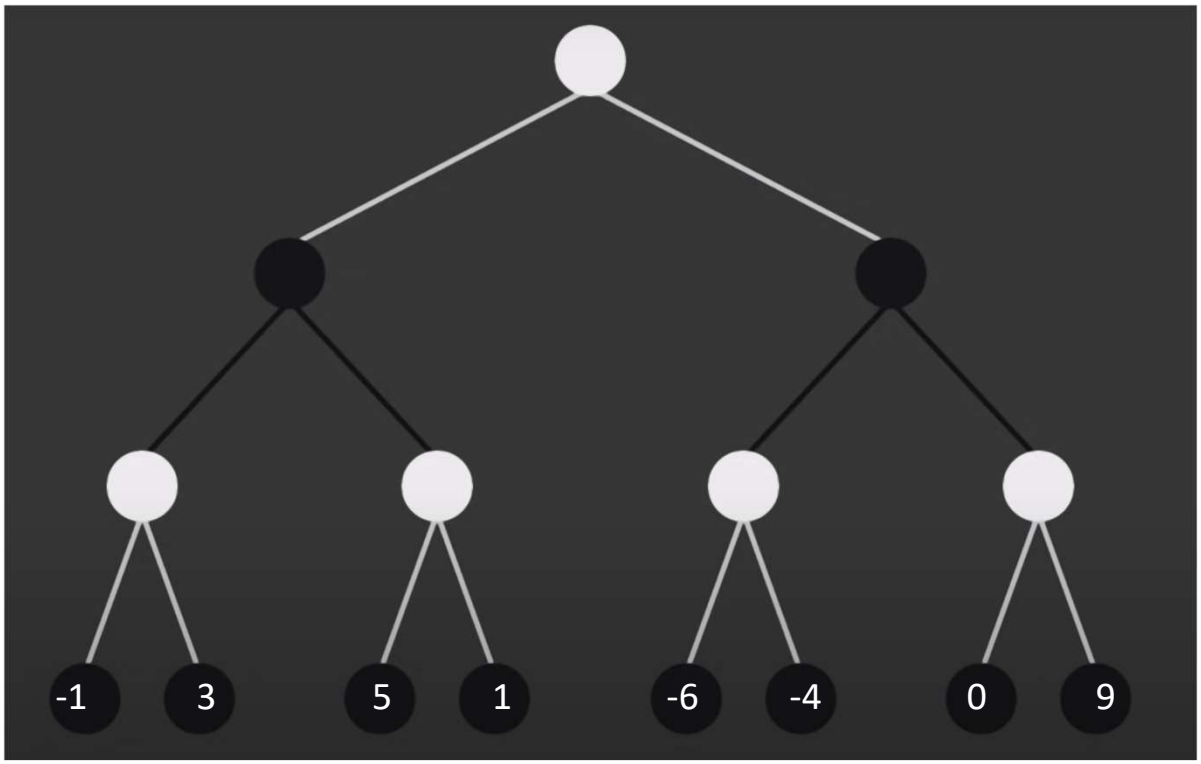
Example: Tic-Tac-Toe



Example – Simple Game



Example: Chess



Properties of minimax

- Complete?? Yes, if tree is finite (chess has specific rules for this)
- Optimal?? Yes, against an optimal opponent. Otherwise?? Time complexity?? $O(b^m)$
- Space complexity?? $O(bm)$ (depth-first exploration) For chess, $b \approx 35$, $m \approx 100$ for "reasonable" games
  - $\Rightarrow$ exact solution completely infeasible
- But do we need to explore every path?

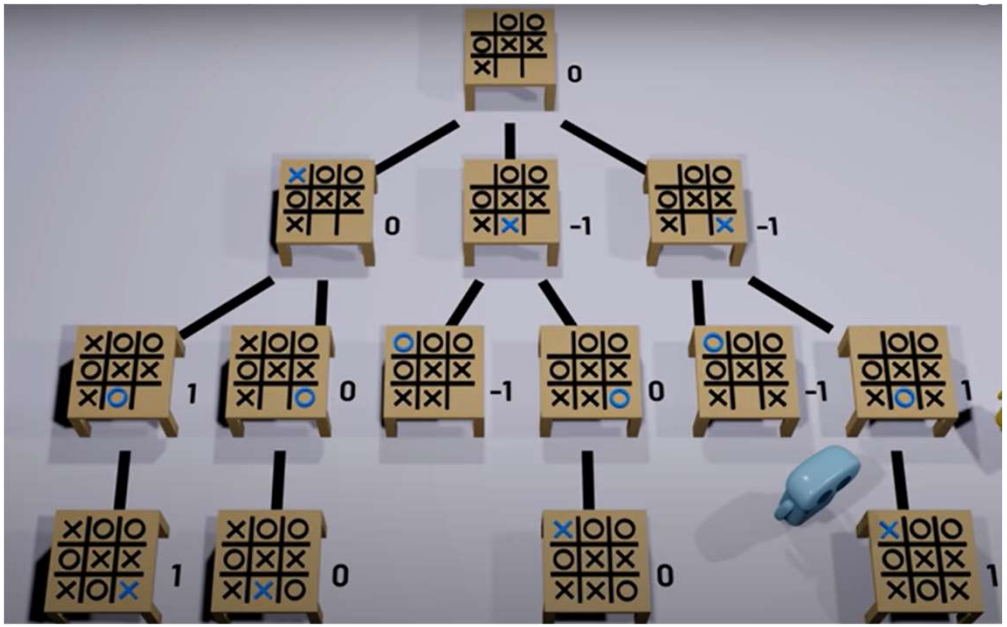
Alpha-Beta Pruning

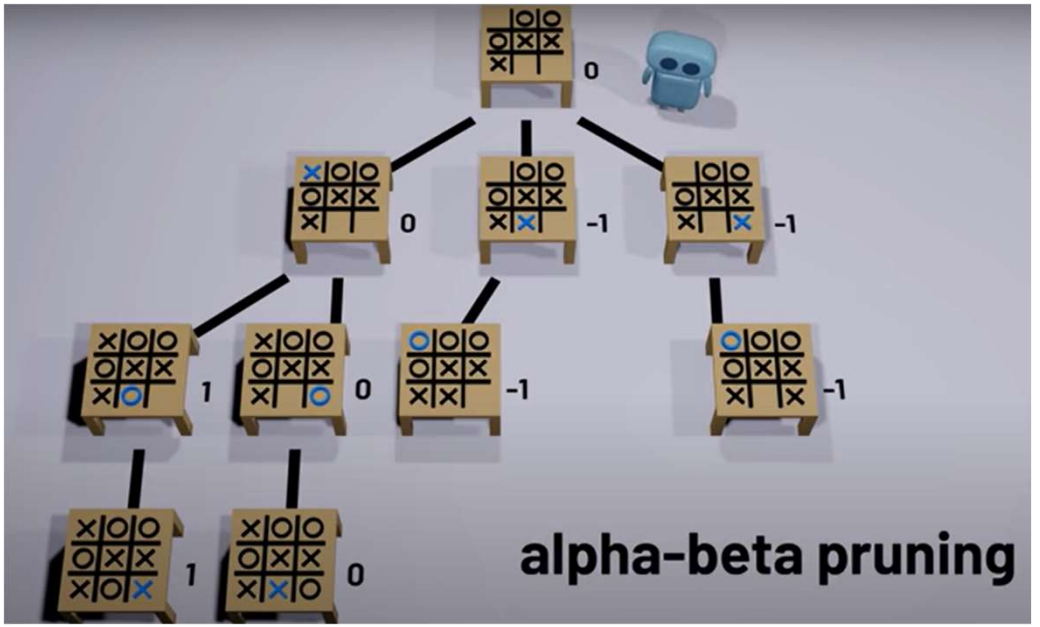
- Alpha-beta pruning is a modified version of minimax algorithm.
- It returns the same move as Minimax would, but **prunes away branches** that cannot possibly influence the final decision.
- Limitation to the minimax algorithm is overcome with alpha-beta pruning.

General Principle

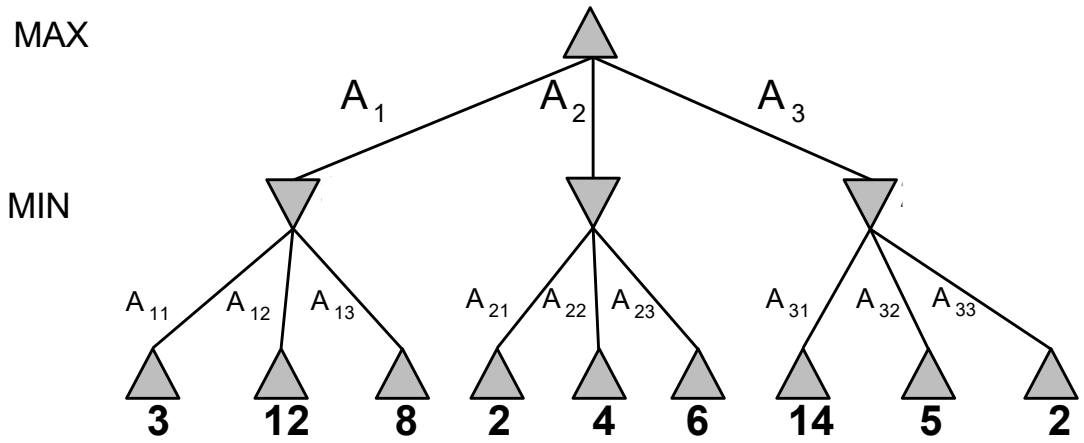
- Consider a node ‘n’ somewhere in the tree, such as that player has a choice of moving to that node.
- If a player has a better choice ‘m’ either at the parent node of ‘n’ or any choice point further up, then ‘n’ will never be reached in actual play.
- Once we found out enough about ‘n’ to reach this conclusion, we can prune it.

Alpha-Beta Pruning

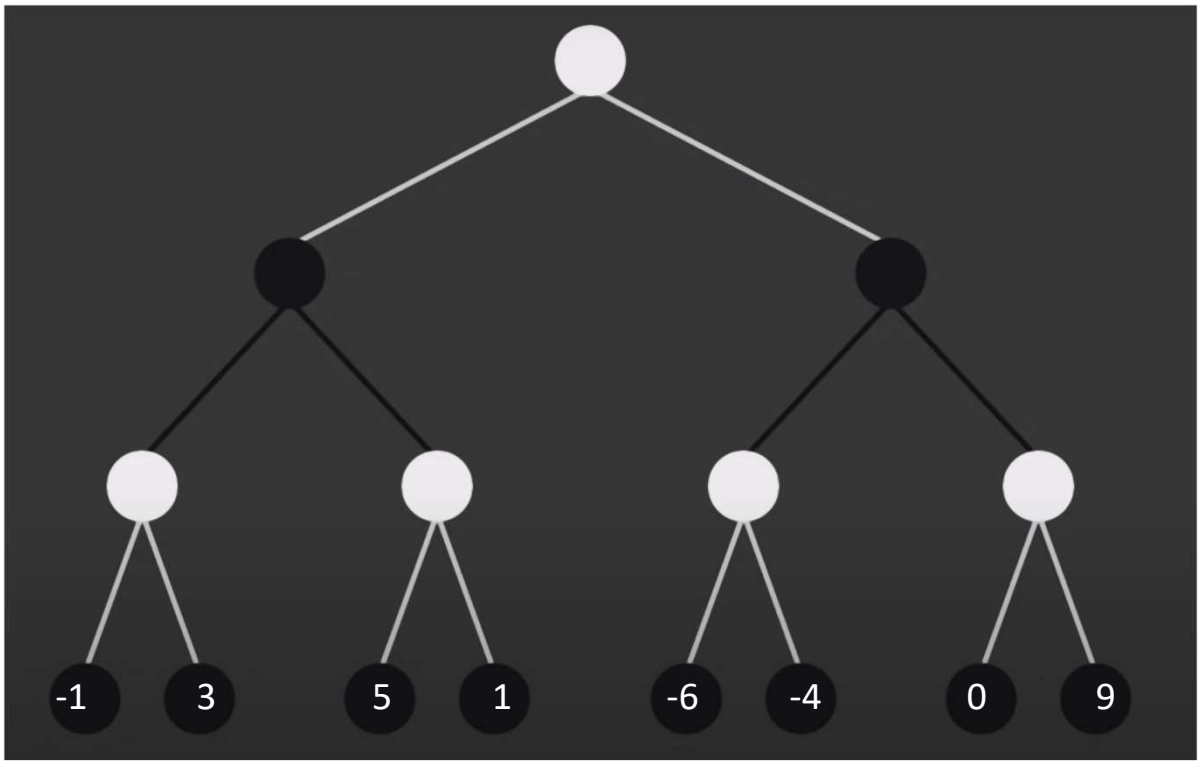




Alpha-Beta Pruning



Alpha-Beta Pruning: Chess



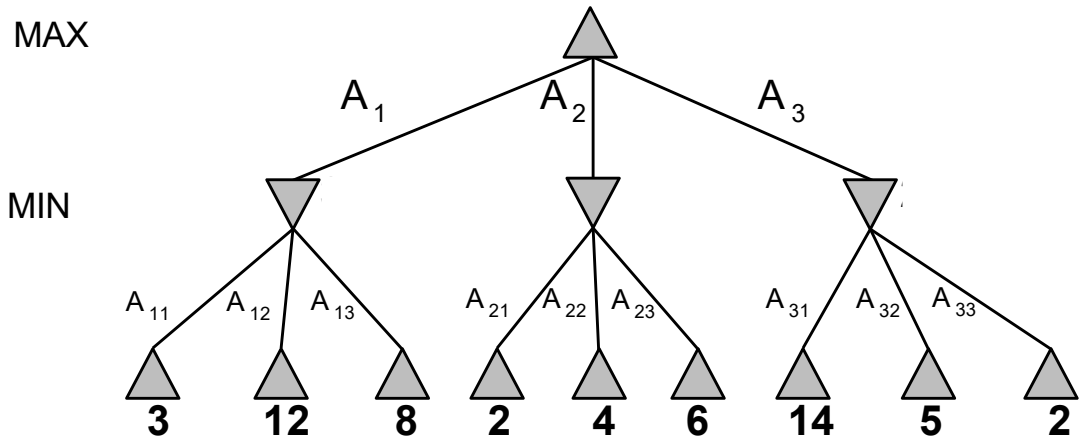
Alpha

- Alpha (α) is the best choice (best highest-value) we have found so far at any point along the path for MAX
- It is a lower bound on the value that a MAX node may ultimately be assigned.
- The MAX node will only update the value of alpha.
- Initially $\alpha = -\infty$

Beta

- Beta (β) is the best choice (best lowest-value) we have found so far at any point along the path for MIN
- It is a upper bound on the value that a MIN node may ultimately be assigned.
- The MIN node will only update the value of beta.
- Initially $\beta = +\infty$

Alpha-Beta Pruning



Alpha-Beta Pruning: Chess