Game Playing as Search

MiniMax Strategy

GAME PLAYING AS SEARCH

In games, there is an opponent.

Given a board configuration, search for/make the **best move;**

opponent responds by making a move creating a new board configuration, **search again** from the new board configuration for best move, make a move…….

Time is limited to find goal in each search

- In games, the objective is not only to **find the best way to the goal** but also **beat the opponent**

GAMES: PROBLEM FORMULATION

Problem formulation

- Initial state: initial board position + whose move it is
- Operators: legal moves a player can make
- Goal (terminal test): game over?
- Utility (payoff) function: measures the outcome of the game and its desirability

Search objective:

- Find the sequence of player's decisions (moves) maximizing its utility (payoff)
- Consider the opponent's moves and their utility

TYPES OF GAMES

Perfect information:

- Each player has complete information about the opponent's position and available choices
  - Deterministic: chess, checkers
  - Chance: Backgammon, monopoly

Imperfect information:

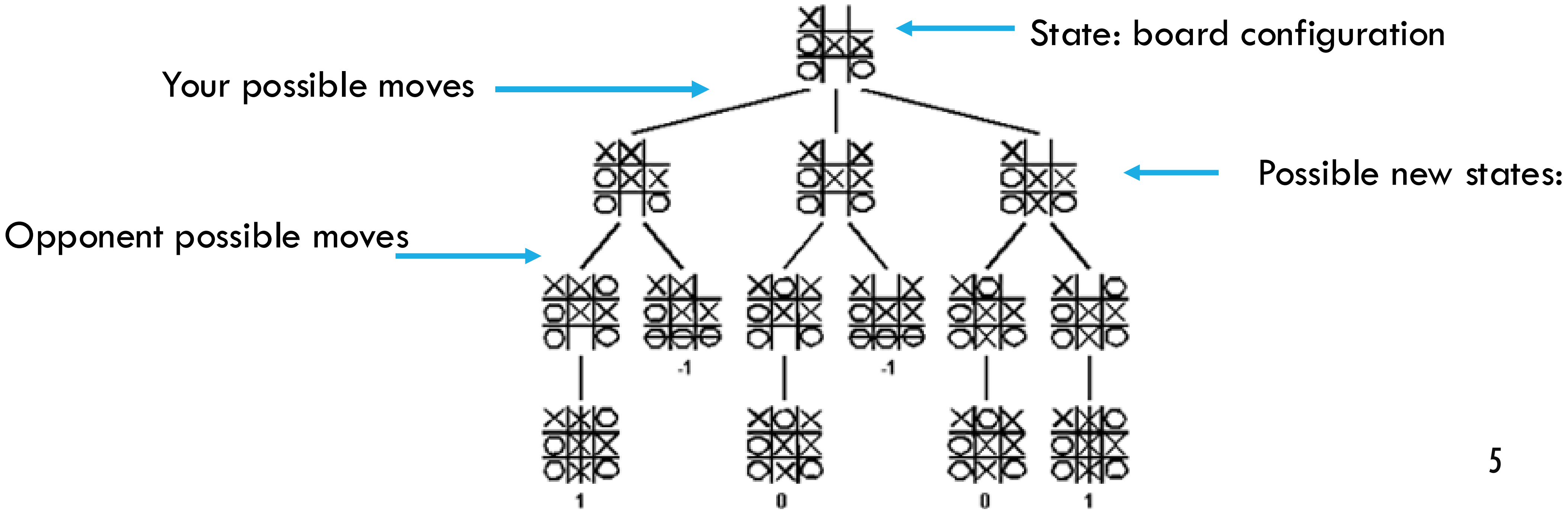
- Each player does not have complete information about the opponent's position and available choices
- Chance: poker, bridge

Two players or multiple players

TWO PLAYERS, PERFECT INFORMATION

Player and Opponent.

Ideally, the player expands the ***game tree*** taking into consideration all the possible moves of the opponent till end of the game with leaf nodes as win, lose, draw.



MIN/MAX

How to deal with the contingency problem?

- Assuming the opponent is rational who optimizes its behavior (opposite to you), consider the best opponent's response/move.

- **The minimax algorithm determines the best move**

MINMAX STRATEGY

Player – MAX

Opponent – MIN.

Commonly used with zero sum games (whenever, one player wins, the other one loses).

Minimax (sometimes **MinMax** or **MM**]) is a decision rule used in decision theory, game theory, statistics and philosophy for *mini*mizing the possible loss for a worst case (*max*imum loss) scenario.

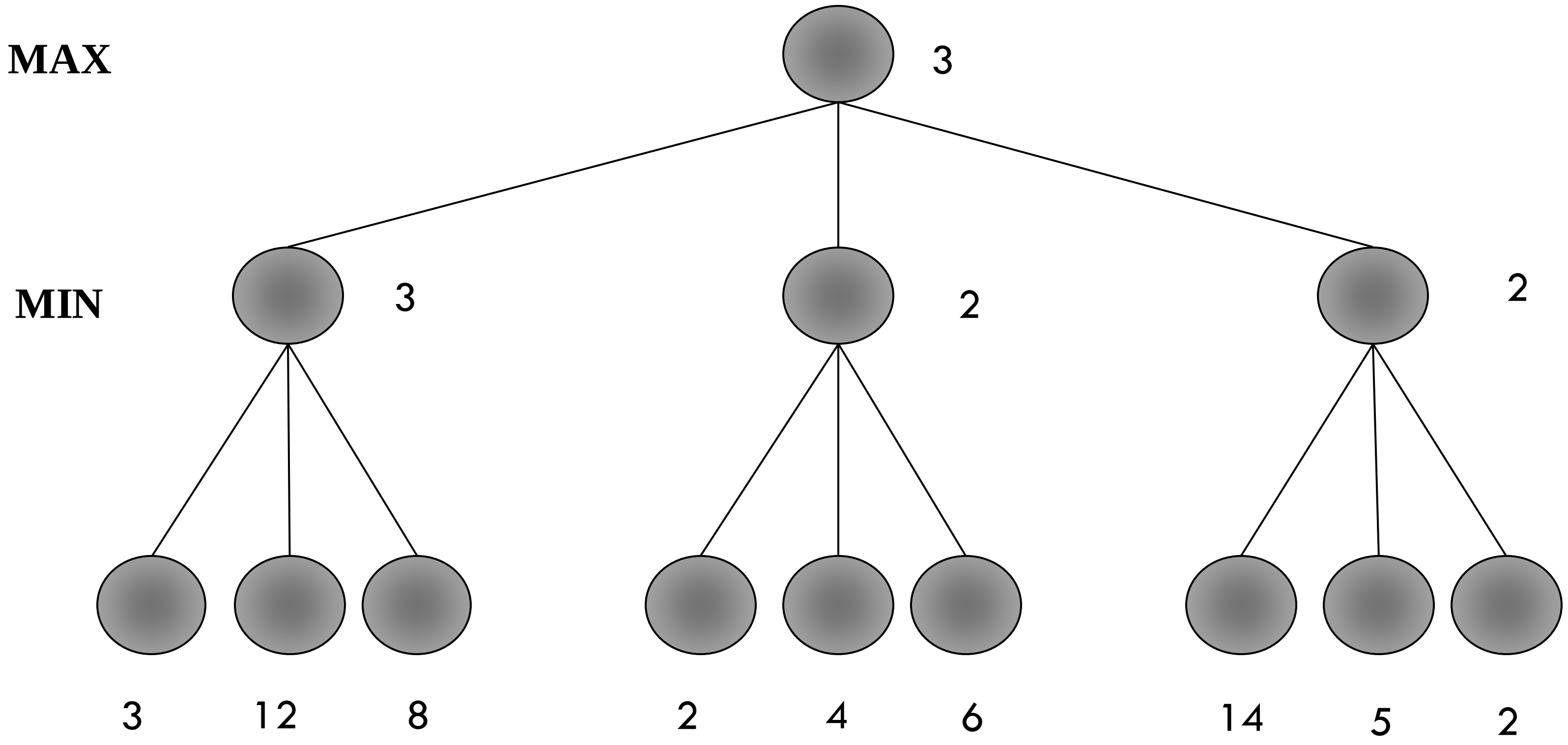
When dealing with gains, it is referred to as "maximin"—to maximize the minimum gain. Originally formulated for two-player zero-sum game theory, covering both the cases where players take alternate moves and those where they make simultaneous moves, it has also been extended to more complex games and to general decision-making in the presence of uncertainty

Given a game tree, the optimal strategy can be determined by examining the minimax value of each node, which we write as MINIMAX- VALUE(^).

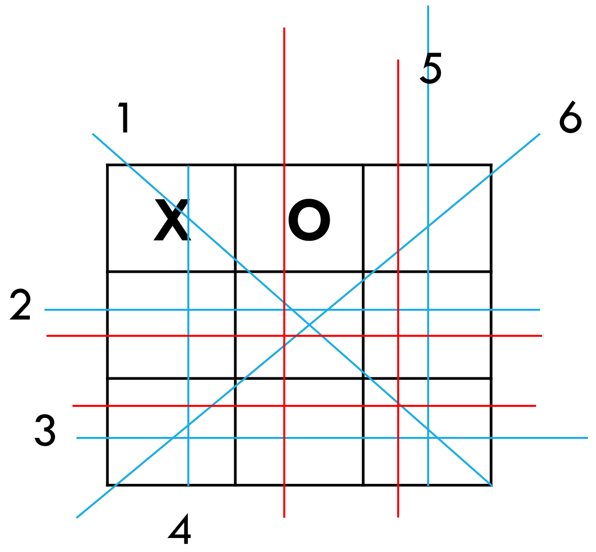
MINMAX STRATEGY

- The minimax value of a node is the ***utility*** of being in the corresponding state, assuming that both players play ***optimally*** from there to the end of the game.
- The minimax value of a terminal state is just its utility.
- Label each level in game tree with **MAX** (player) and **MIN** (opponent)
- Label leaves with evaluation of player
- Go through the game tree
  - if father node is **MAX** then label the node with the maximal value of its successors
  - if father node is **MIN** then label the node with the minimal value of its successors

MAX MIN BACKED UP VALUES



TIC-TAC-TOE

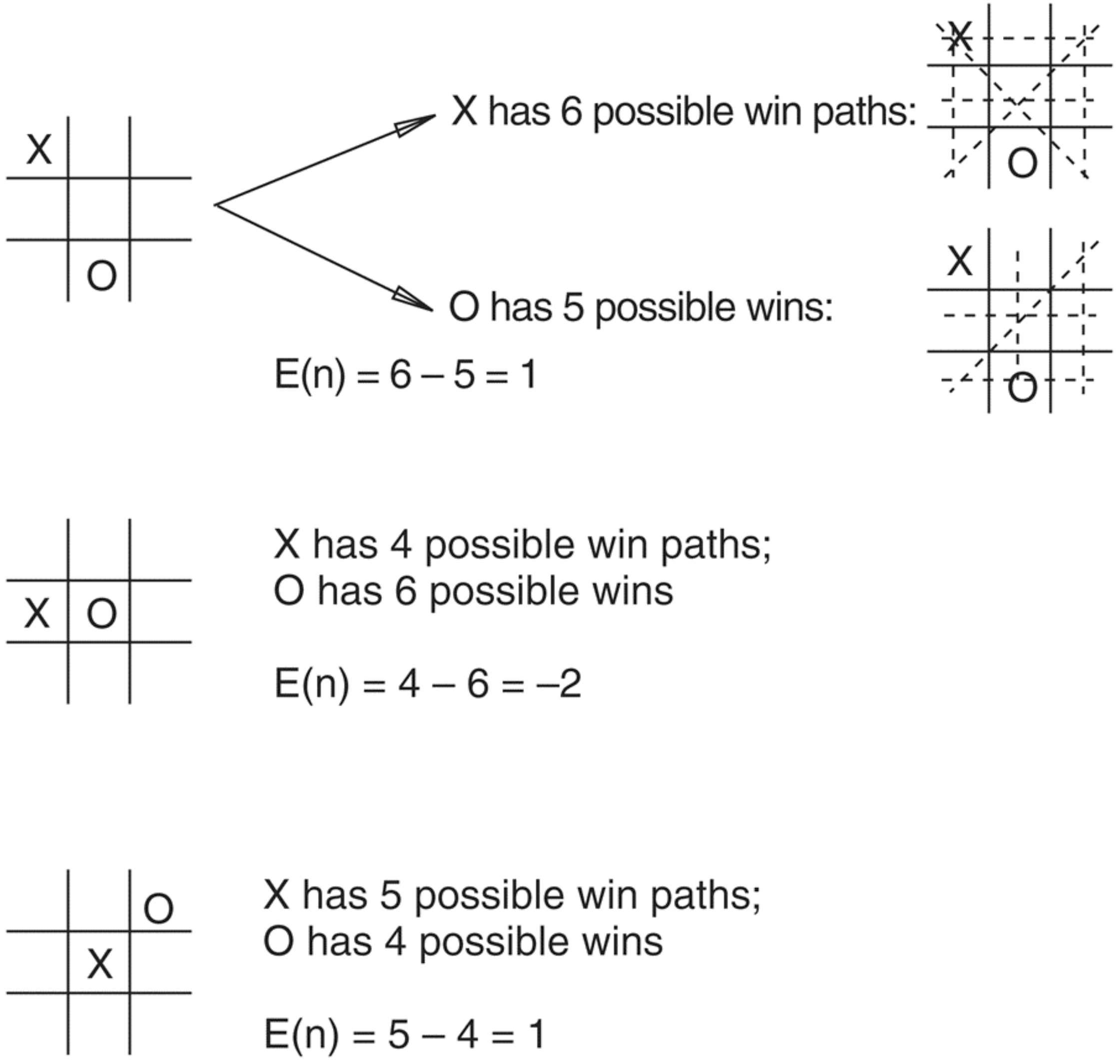


e(p) = 6 - 5 = 1

- **Initial State:** Board position of 3x3 matrix with 0 and X.
- **Operators:** Putting 0's or X's in vacant positions alternatively
- **Terminal test:** Which determines game is over
- **Utility function:**

  e(p) = (No. of complete rows, columns or diagonals are still open for player) – (No. of complete rows, columns or diagonals are still open for opponent)

HEURISTIC MEASURING CONFLICT



CALCULATION OF THE HEURISTIC

- $E(n) = M(n) – O(n)$ where
  - $M(n)$ is the total of My (MAX) possible winning lines
  - $O(n)$ is the total of Opponent’s (MIN) possible winning lines
  - $E(n)$ is the total evaluation for state n
-

MINIMAX ALGORITHM

Generate the game tree

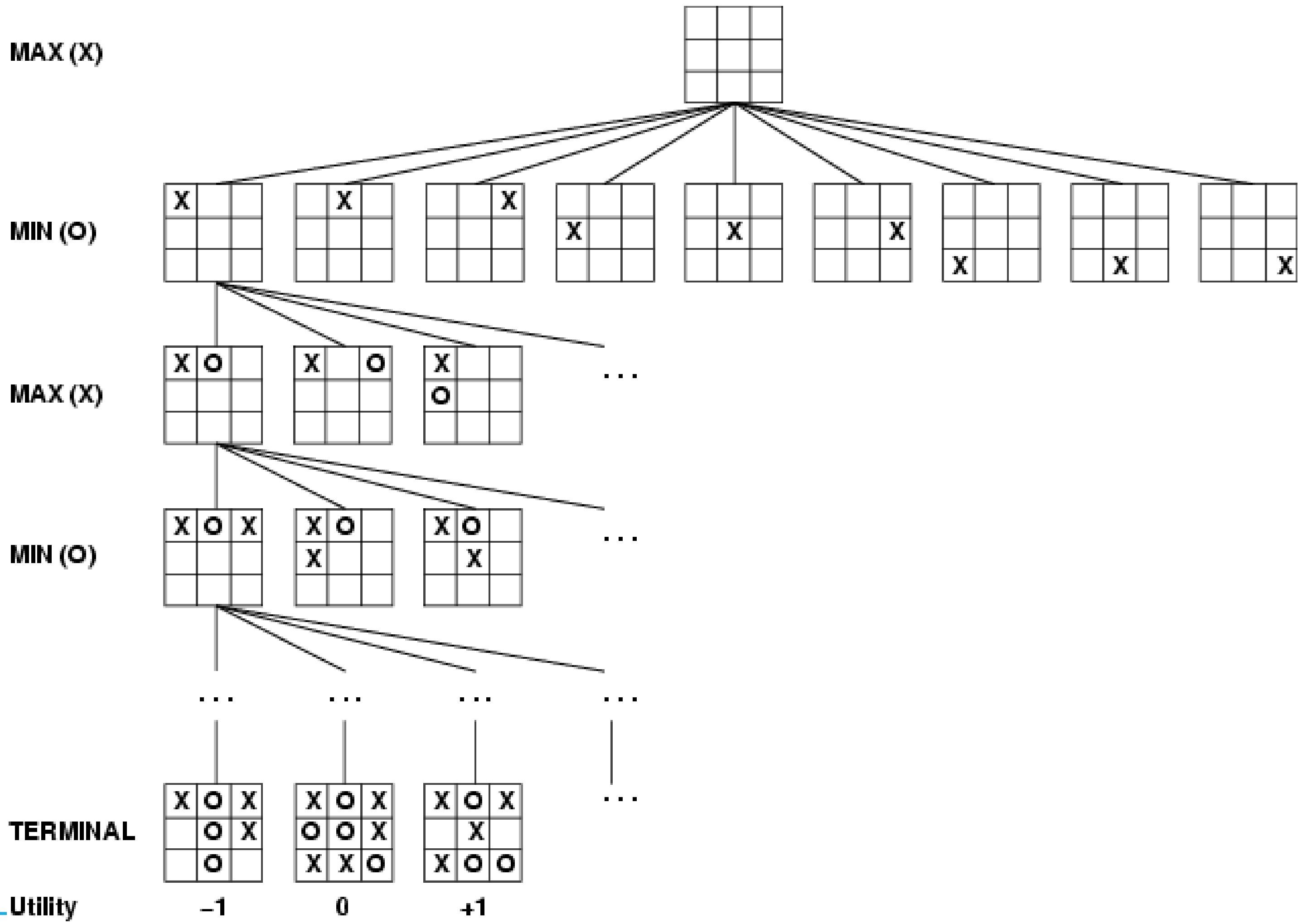
Apply the utility function to each terminal state to get its value

Use these values to determine the utility of the nodes one level higher up in the search tree

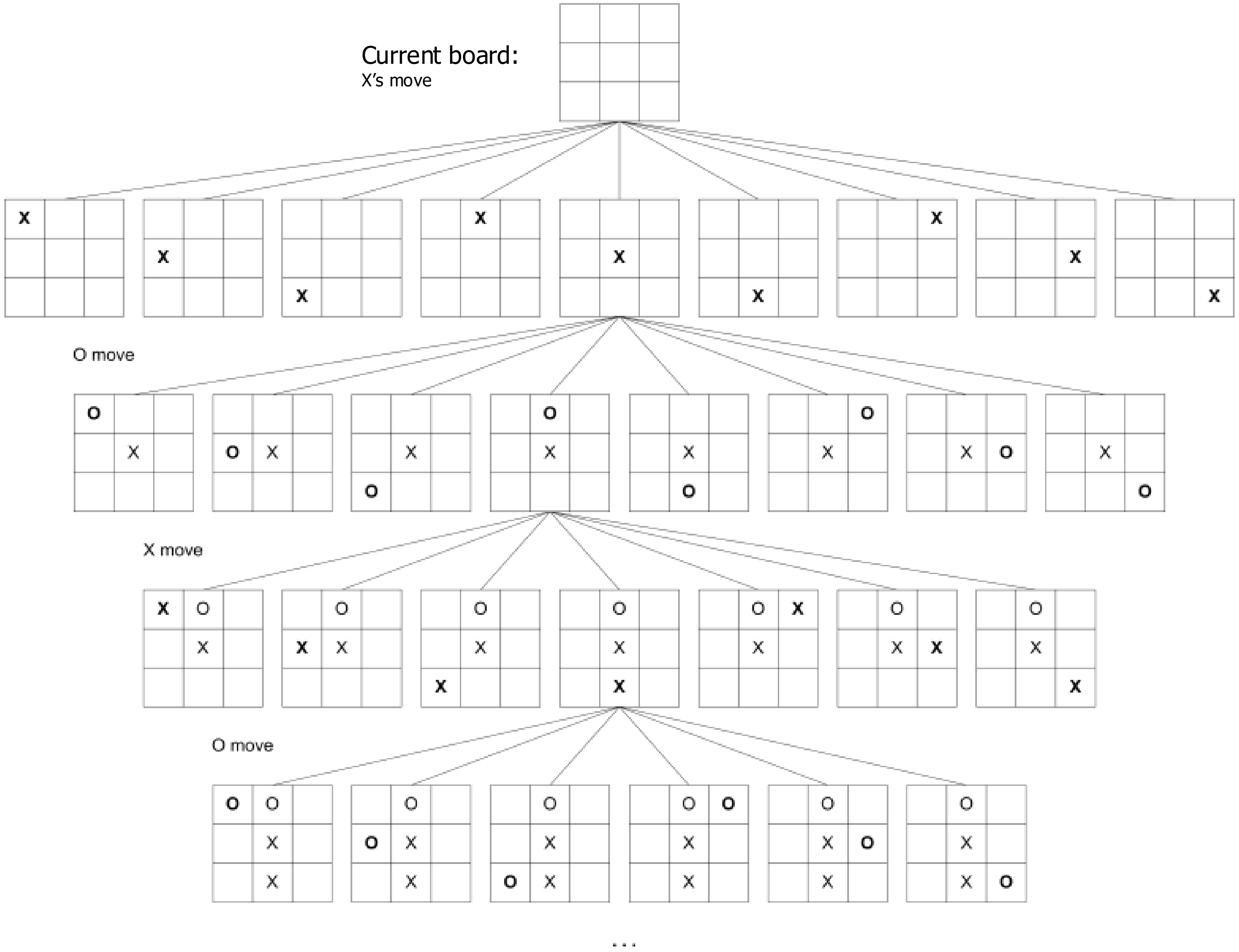
- From bottom to top
- For a max level, select the maximum value of its successors
- For a min level, select the minimum value of its successors

From root node select the move which leads to highest value

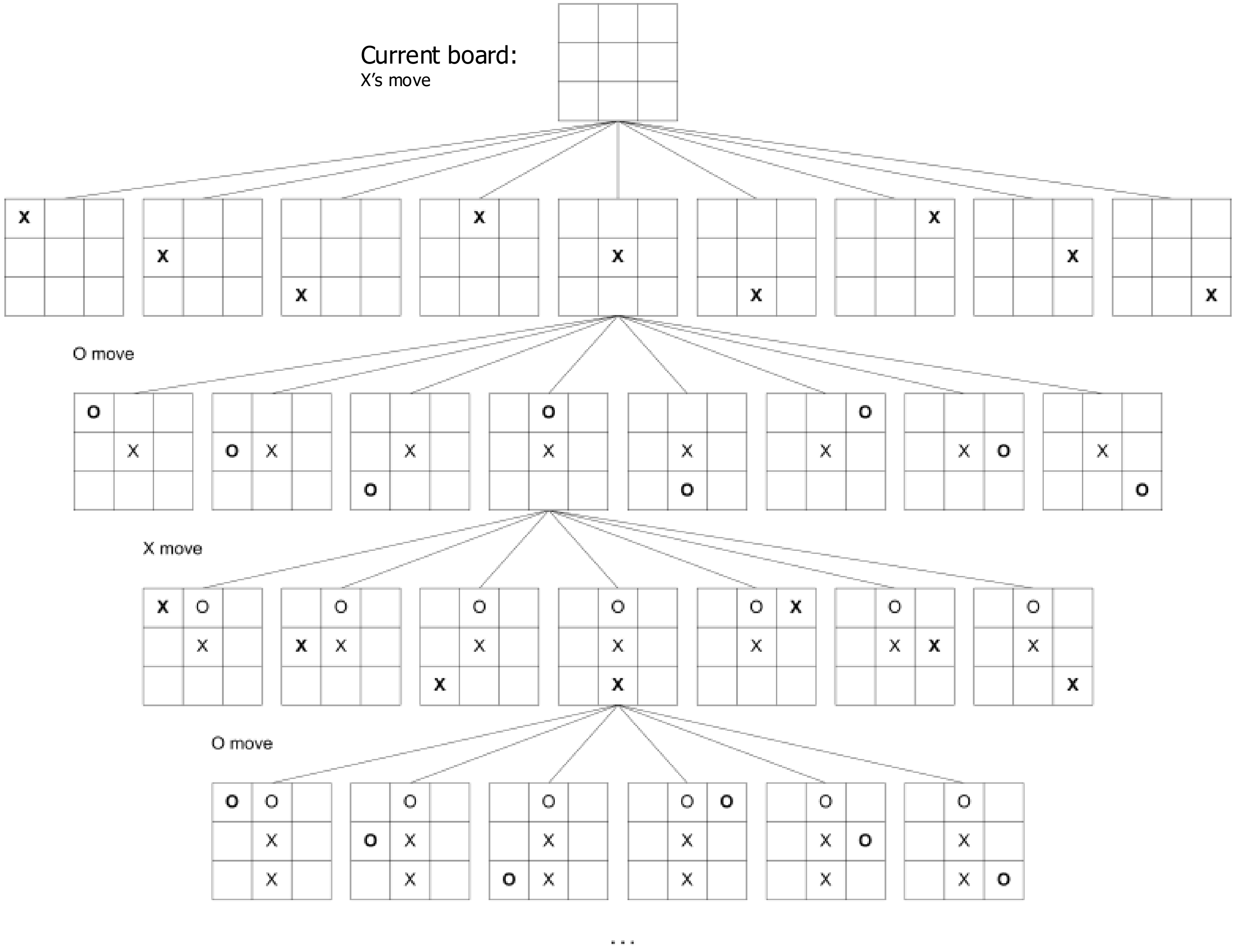
MAX MIN FOR TIC TAC TOE



GAME TREE



GAME TREE



How many nodes?

LIMITED DEPTH AND EVALUATION FUNCTION

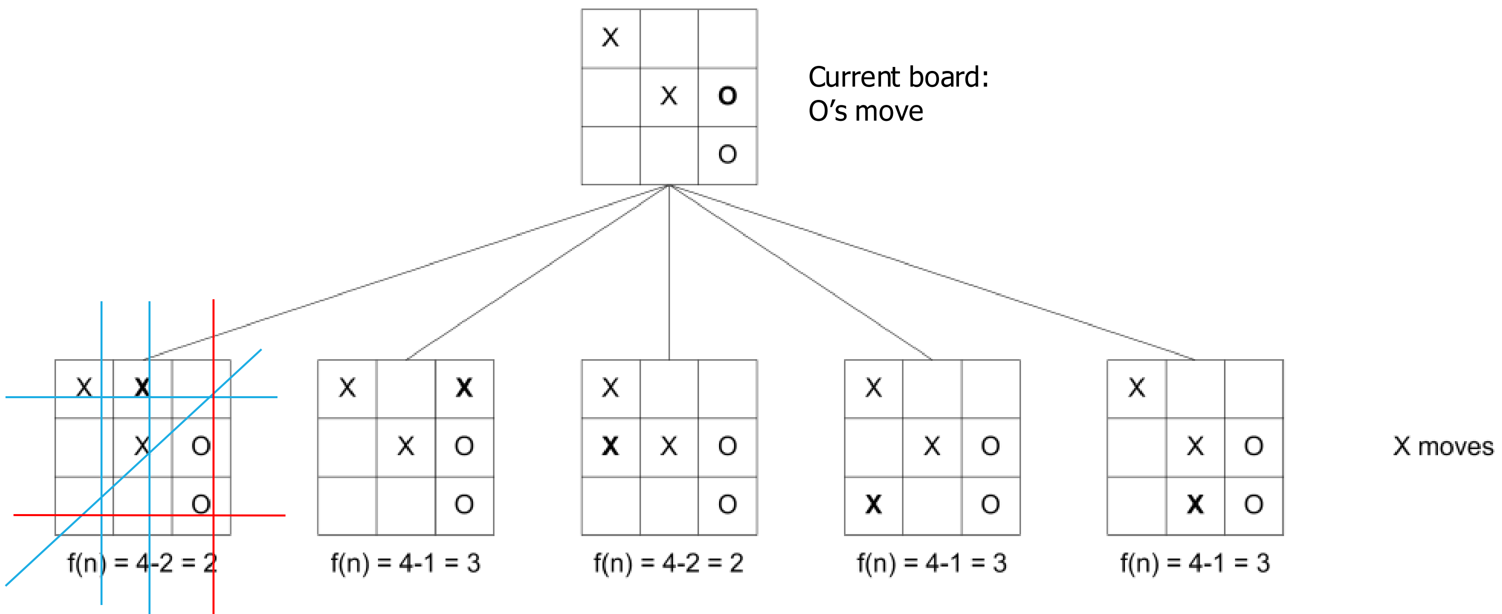
Expanding the complete game tree is only feasible for simple games.

- Tic Tac Toe has average branching factor ***b = 5*** and average (moves) depth ***d = 9***.
  - Total states of 9! => 362,880 states in the game tree

- Checkers has *b* =35 and *d* =*100*.
  - Taking into consideration symmetrical and repeated states the game tree may be even less.
  - Still for checkers 10^{40}.

For Chess, it is only feasible to explore a ***limited depth*** of the game tree and to use a board evaluation function to estimate the worth of this node to the player.

EVALUATION FUNCTION



- Evaluation function *f*(*n*) measures "***goodness***" of board configuration *n*.
- Assumed to be better estimate as search is deepened (i.e., at lower levels of game tree).
- **Evaluation function:** "Number of possible wins (rows, columns, diagonals) not blocked by opponent, minus number of possible wins for opponent not blocked by current player."

APPLYING MINIMAX TO TIC-TAC-TOE

The static evaluation function heuristic

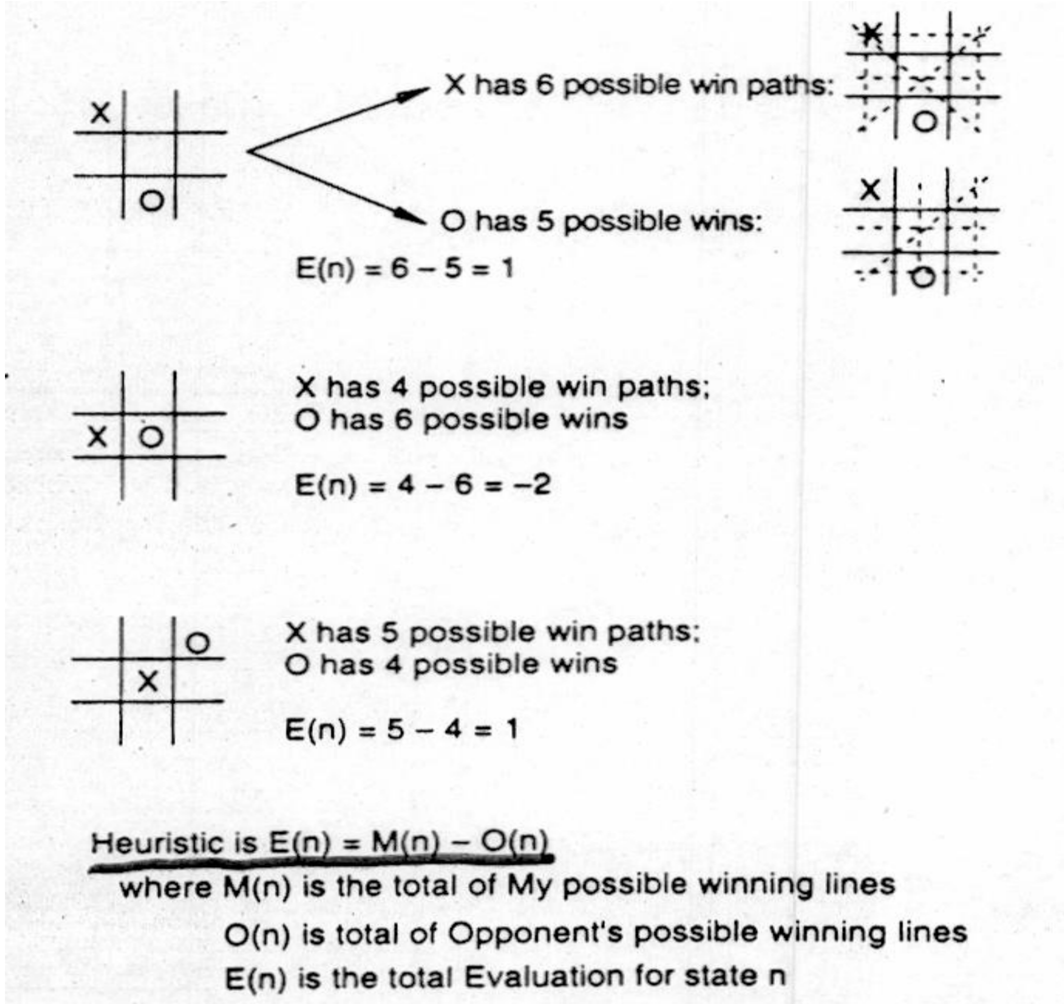
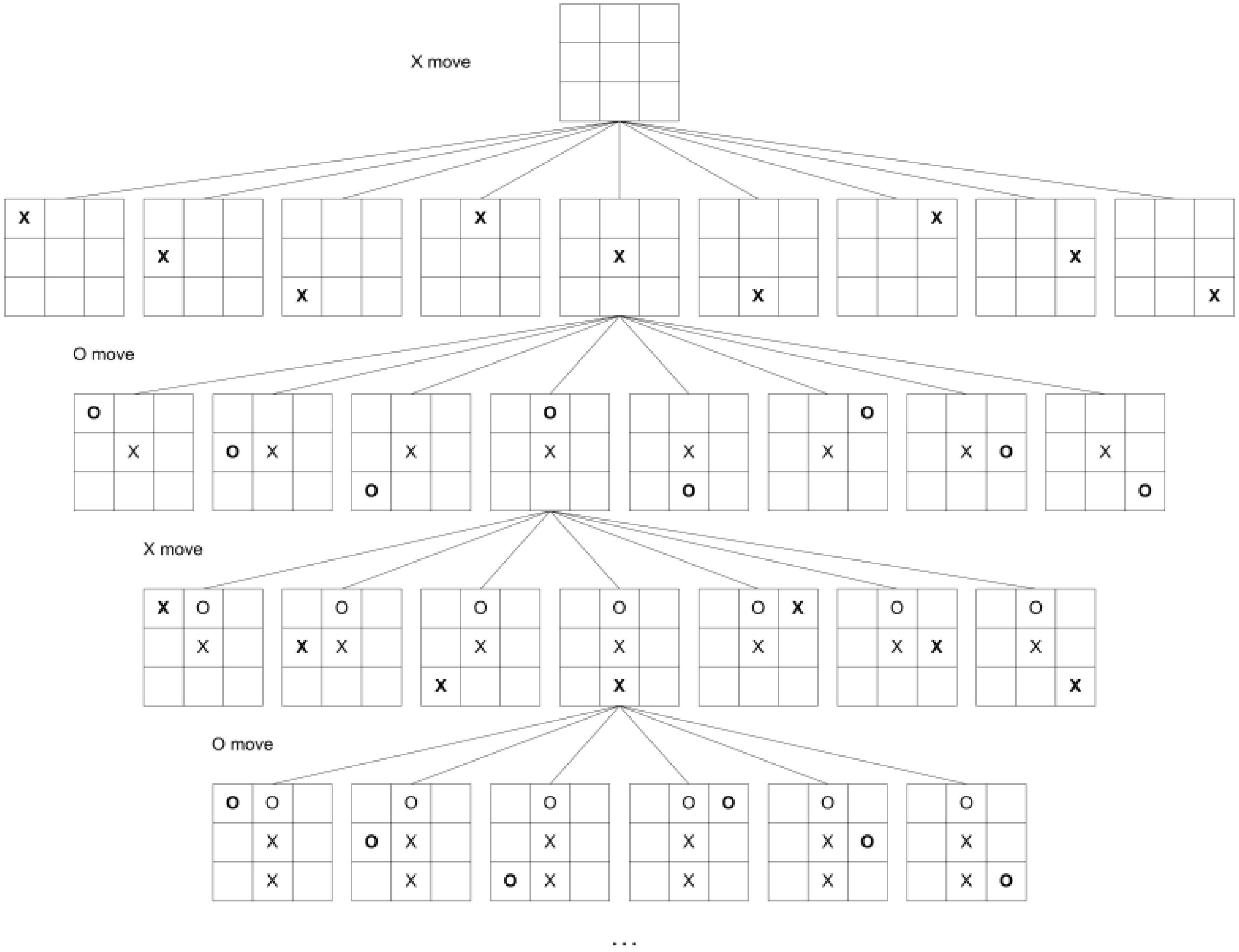


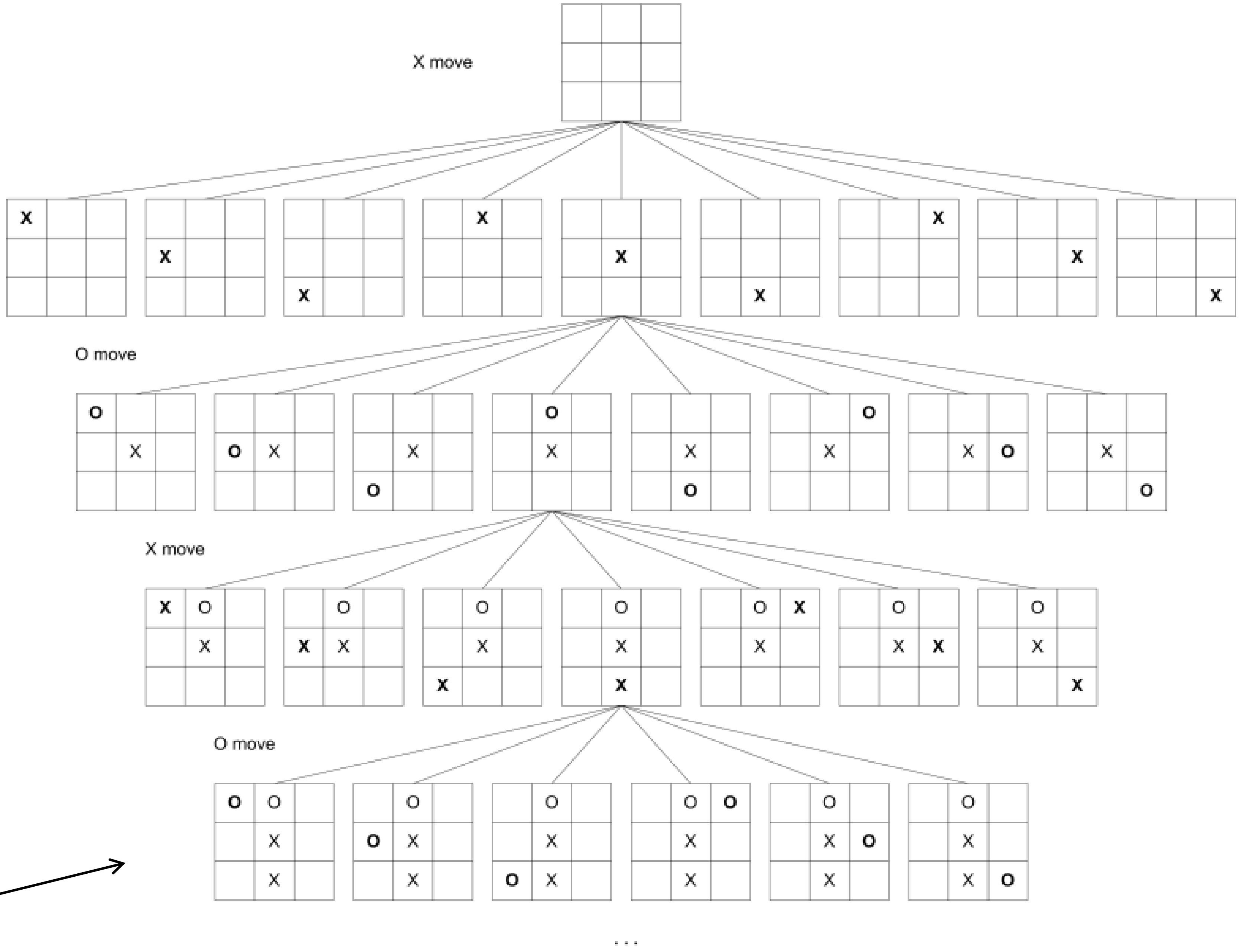
Figure 4.16 Heuristic measuring conflict applied to states of tic-tac-toe.

MINIMAX SEARCH



- Minimax search: Expand the game tree by *m* ply (levels in game tree) in a ***limited depth-first*** search. Then apply evaluation function at lowest level, and ***propagate*** results back up the tree.

MINIMAX SEARCH



Calculate f(n)

EXAMPLE: TIC-TAC-TOE

e (evaluation function $\rightarrow$ integer) = number of available rows, columns, diagonals for MAX - number of available rows, columns, diagonals for MIN

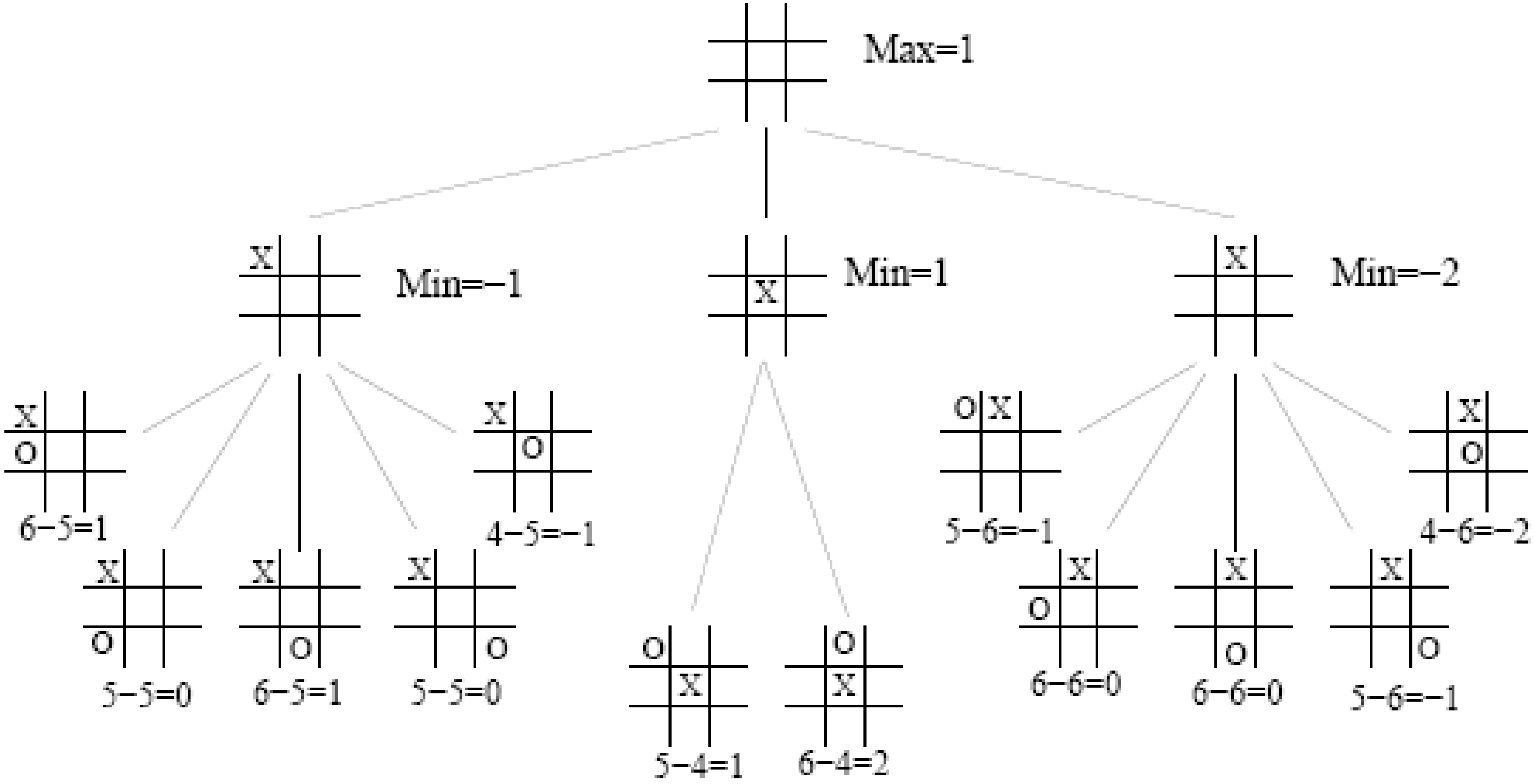
MAX plays with “X” and desires maximizing e.

MIN plays with “O” and desires minimizing e.

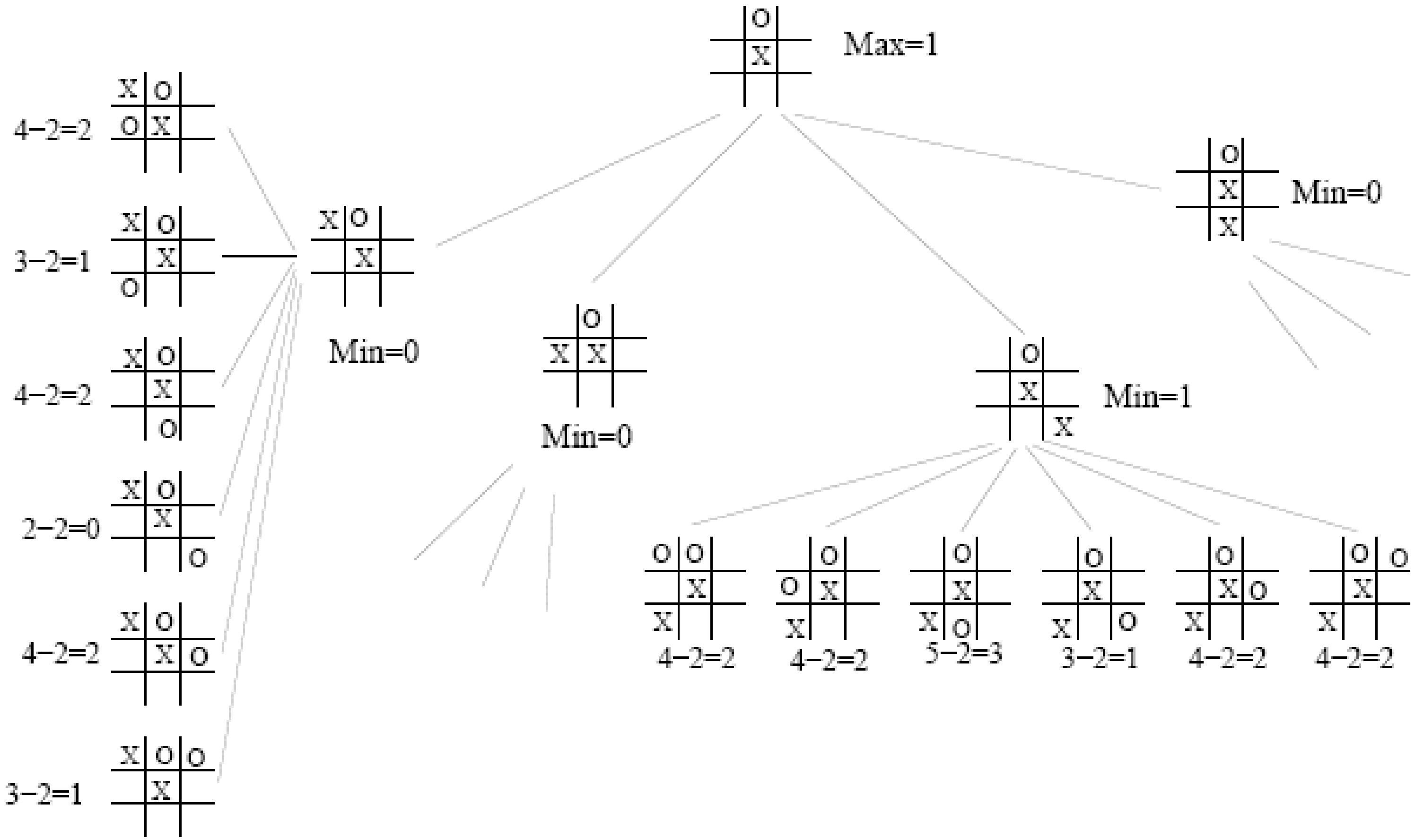
Symmetries are taken into account.

A depth limit is used (2, in the example).

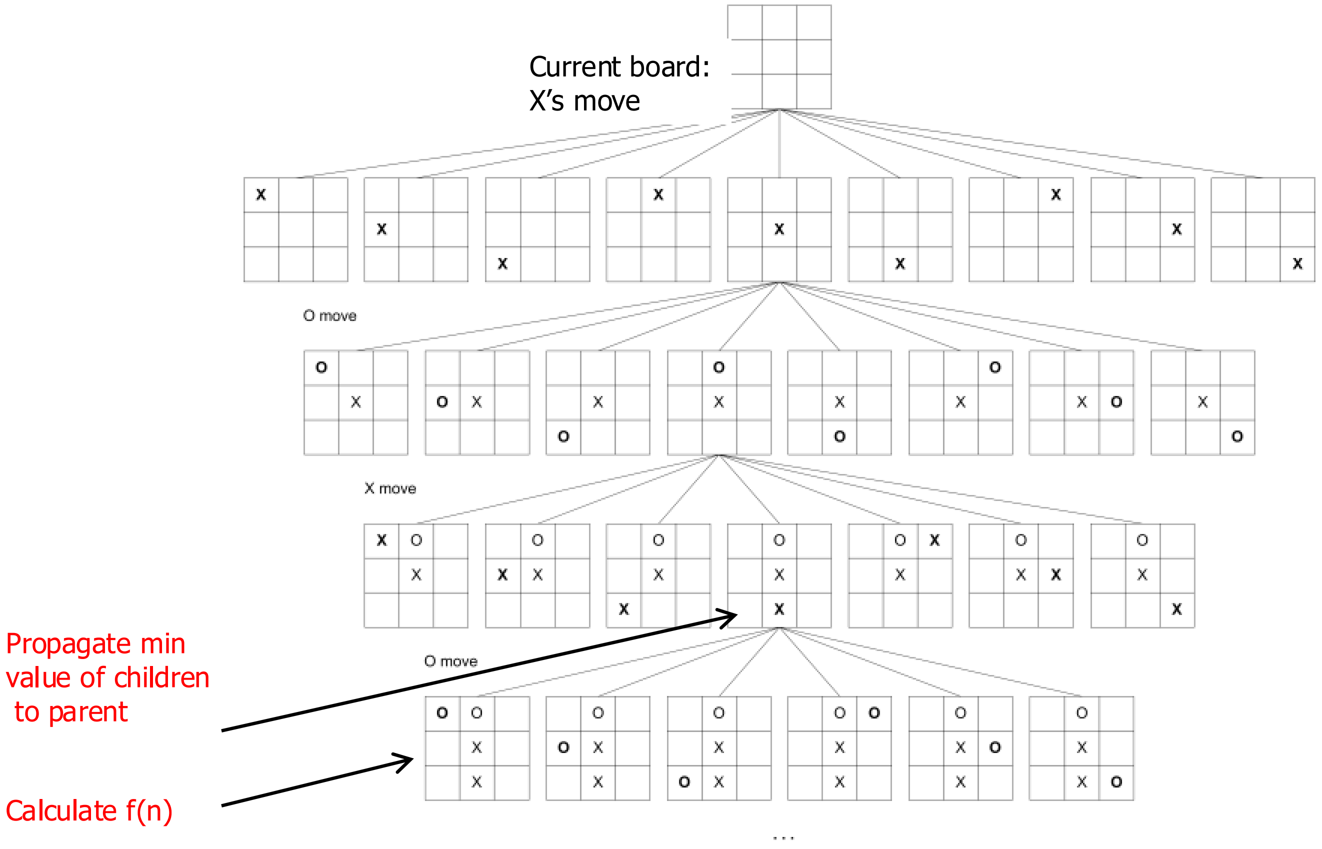
EXAMPLE: TIC-TAC-TOE



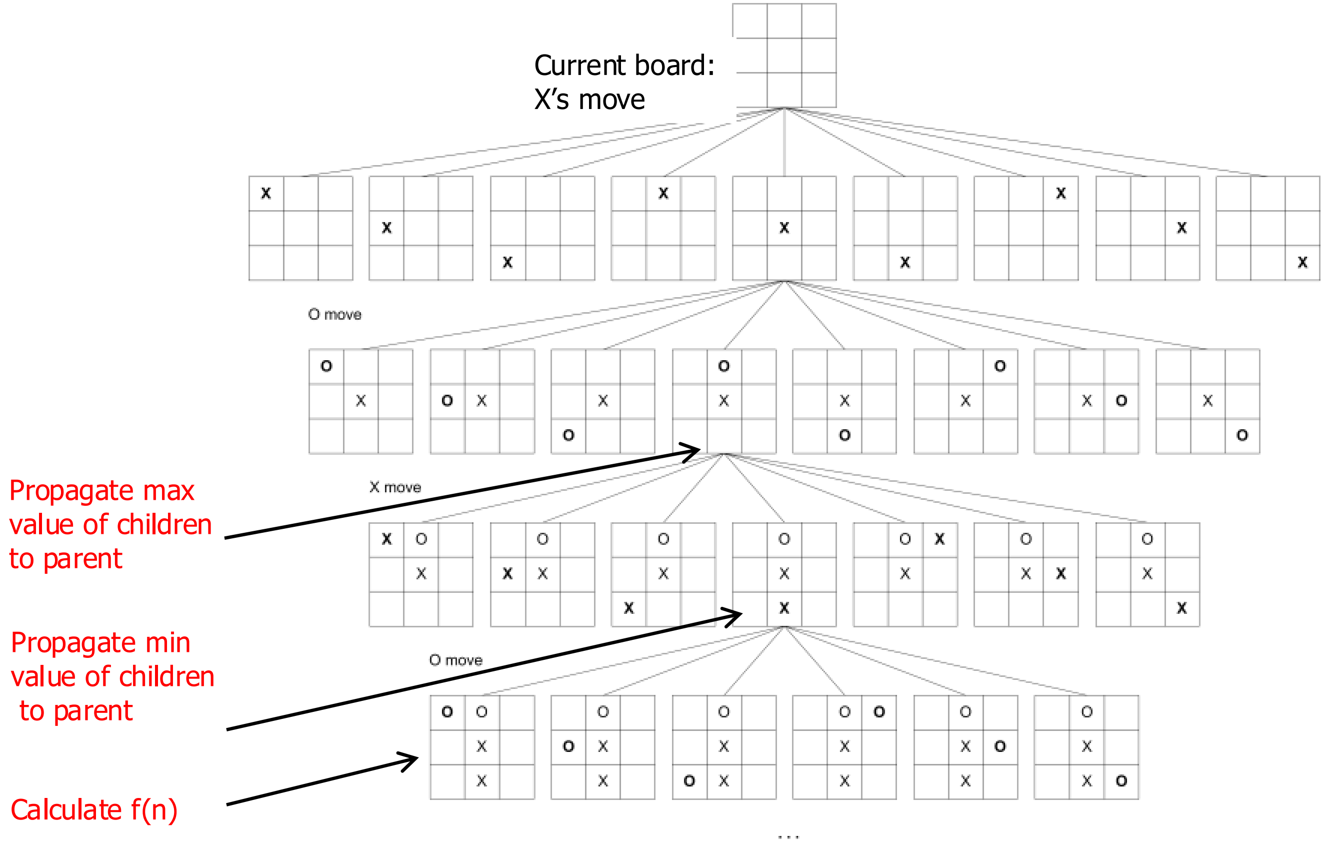
EXAMPLE: TIC-TAC-TOE



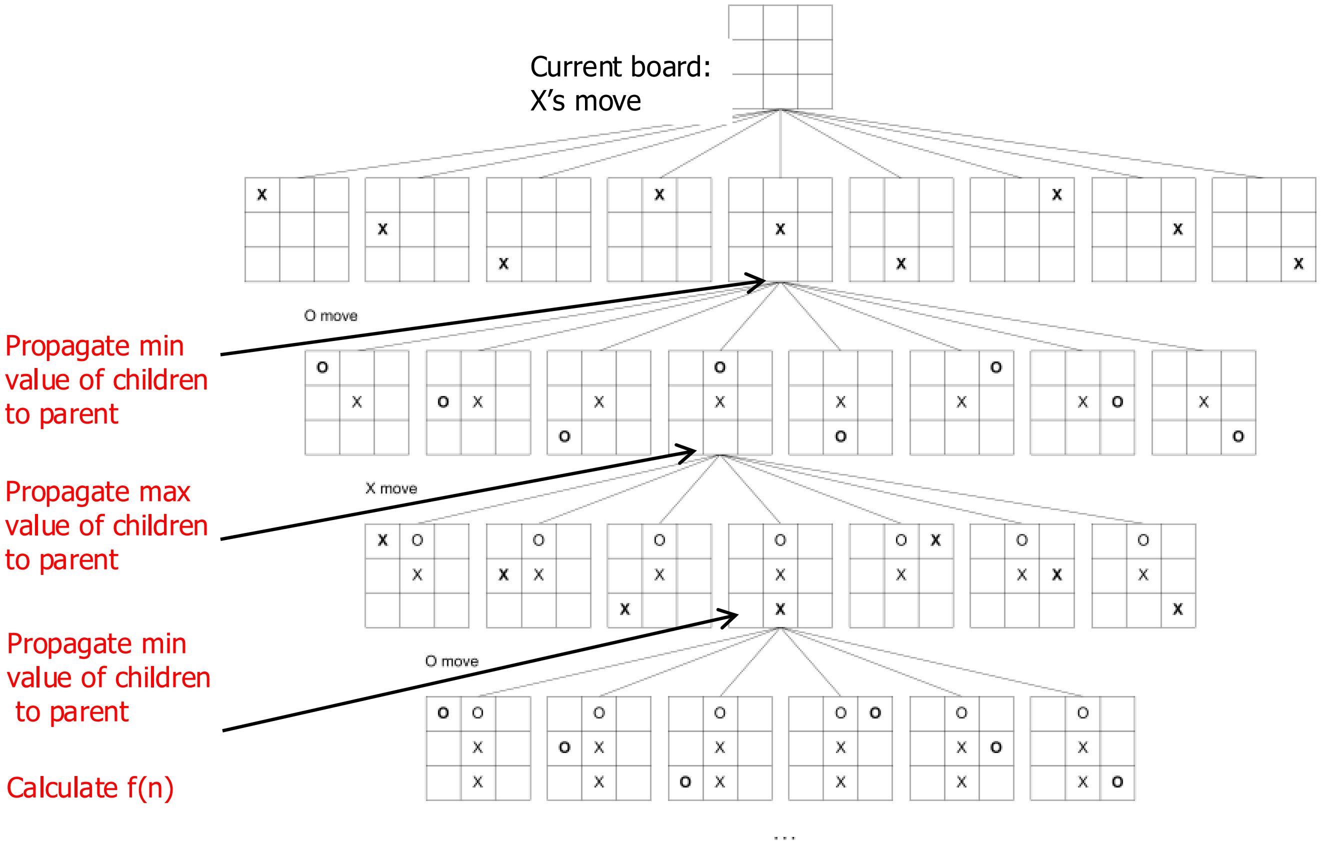
MINIMAX SEARCH



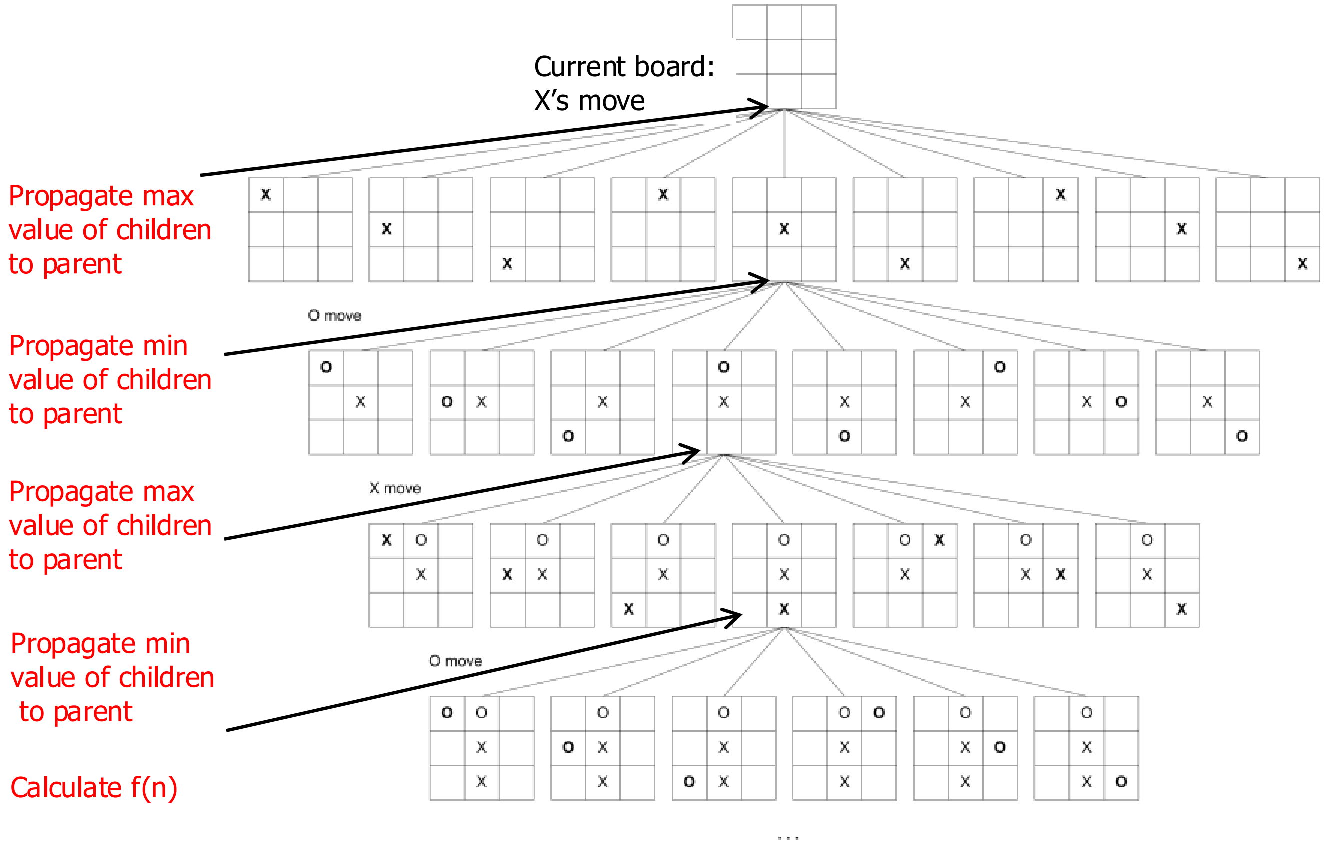
MINIMAX SEARCH



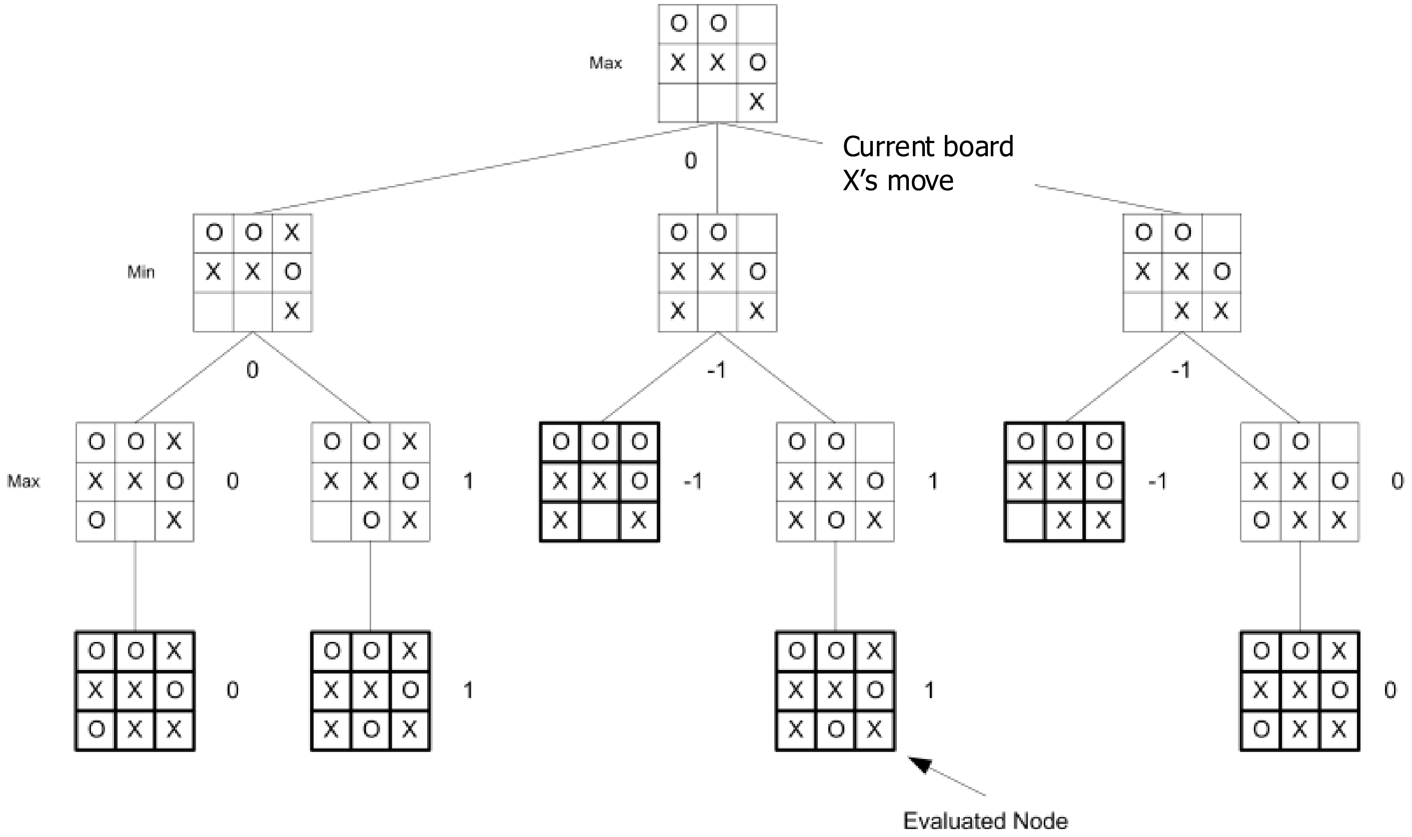
MINIMAX SEARCH



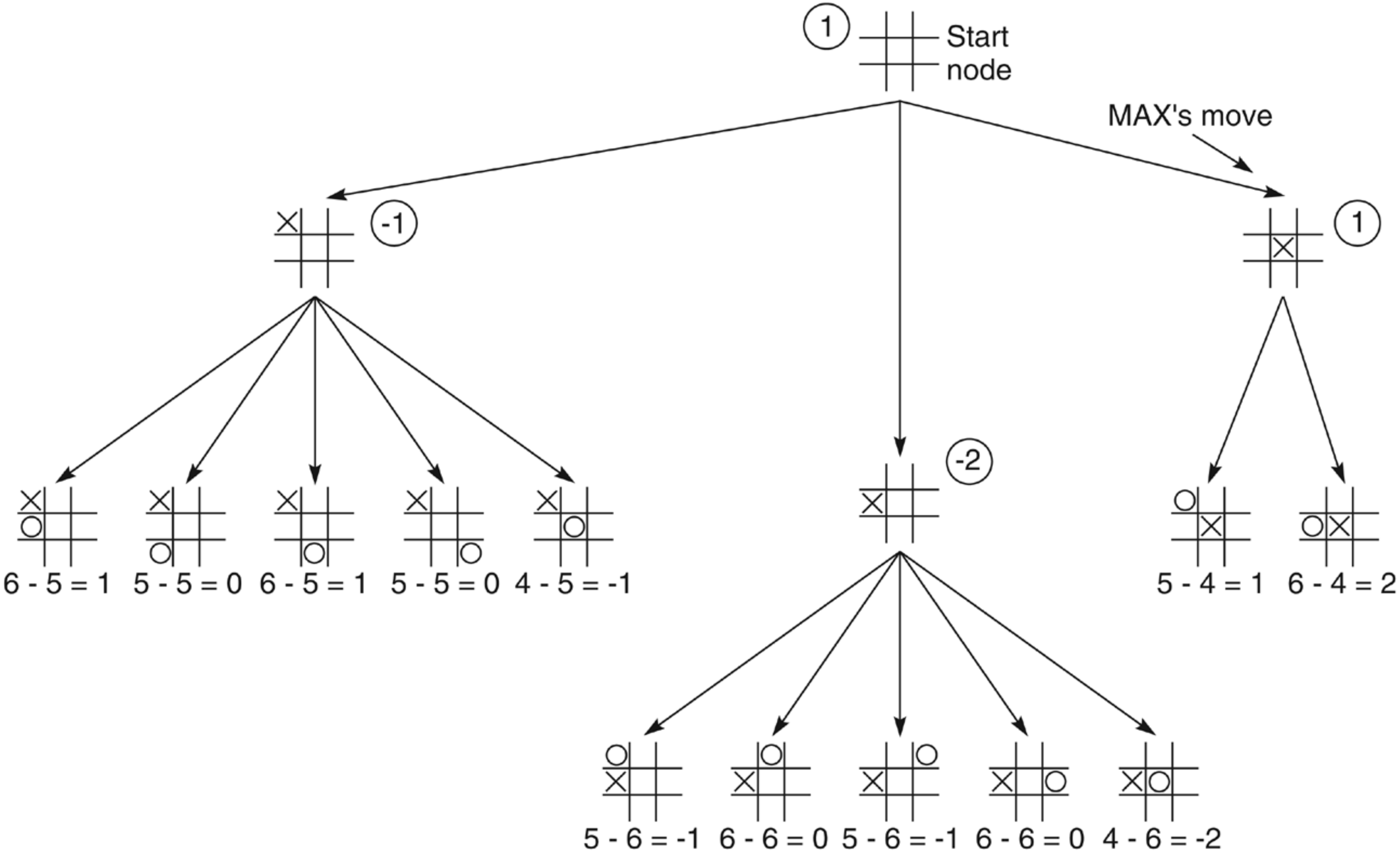
MINIMAX SEARCH



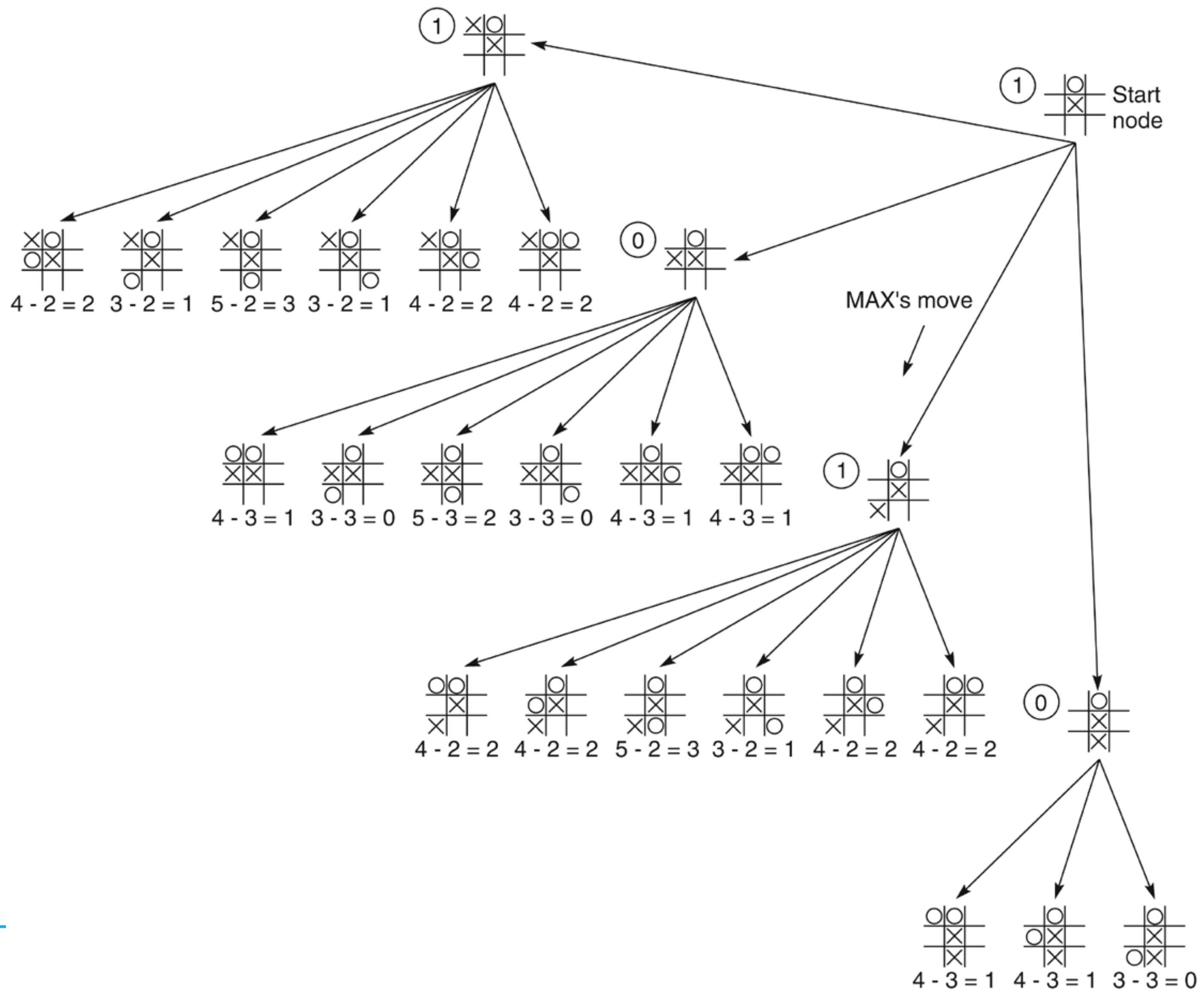
MINIMAX ALGORITHM: EXAMPLE



TWO-PLAYER MINIMAX APPLIED TO THE OPENING MOVE OF TIC-TAC-TOE (NILSSON, 1971)



TWO-PLY MINIMAX AND ONE OF TWO POSSIBLE SECOND MAX MOVES (NILSSON, 1971)



MAX MIN STRATEGY

Complete: Yes (if tree is finite)

Optimal: Yes (against an optimal opponent)

Time complexity: $O(b^d)$

Space complexity: O(bd) (depth-first exploration)