Introduction to Artificial Intelligence

Week 4

Learning by Searching

Your Car is Lost in a Carpark

- How would you find it?
- Walk at random
  - No guarantee you will find it
  - Could be lost forever
- Walk down each row
  - Guarantee you will find it
  - Might take a long time
- Walk to the place you think it was last, then expand your search
  - Guarantee you will find it
  - As long as your memory is good – on average it will be less than every row, worst case just as good
- Walk to the place you think it was last, hit the alarm button, then expand search
  - Guarantee you will find it
  - Alarm can be heard from a long way away, and if you press it to no effect the car is not local to you (radius of activation)

Search Space

- Many problems define a set of actions on an environment which can be seen as a search space of states
  - We are at one point in the environment with a destination, there is a list of action states which will bring us to the destination
- Each search type is a method of navigation of the space
- We will rarely want to go to the same point in the search space
  - Represents a cycle in the search, which are not useful in terms of state spaces
- Solution – most searches will avoid this by using tree structures
  - Each node in the tree is a state within the search
  - Search Trees, Game Trees

Types of Searches

Uninformed Search

- Have no sense of the problem domain
- Generally applicable in all cases

Informed Search

- Use a heuristic function developed for the domain
- Applicable in their own domain

Goal-Based Agent

- Four Steps in the Agent:

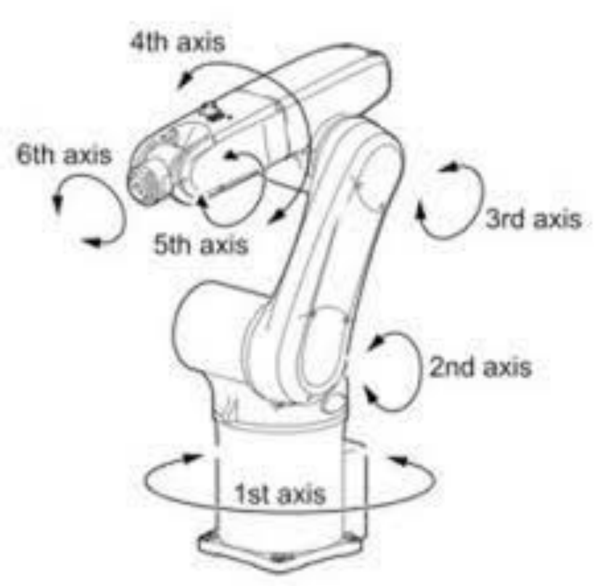
1. Define Goals
   - What are the states which are considered to be satisfying the goal?
2. Problem Formulation
   - What actions are available to move towards the goal?
   - What states are available to move towards the goal?
3. Search
   - Determine the pathway of states in order to meet with the goal
4. Execute
   - Move though the series of states

Example: Painting Robot (simplified)

1. Goal
   - All show surfaces of the part are covered in 1mm of paint
2. Problem Formulation
   - A sequence of points about the part for the robot arm to move into in order to ensure a dispersal about the entire part
3. Search
   - Examine all pathways about the part, taking into account the movements on each part of the arm
   - A pathway is good as long as it covered the part with 1mm of paint
4. Execute
   - The arm moves though the sequence of positions

State Space of the Robot

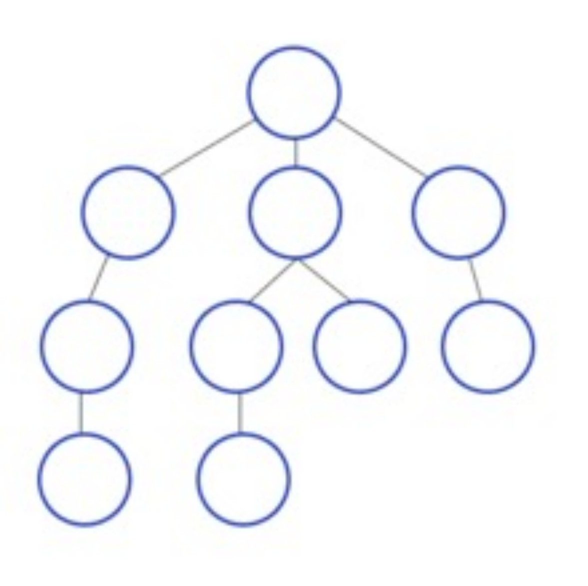
- State space is defined as the location of all moving elements on the robot
  - Relative positioning on all joints from a home position
  - Might also include the rate of spray on the nozzle – from 0 to full
- Transitions of states could be broken down into a sequence of movements in the joints on each rotational axis (hence a 6-axis robot)
  - Move(Elbow, X degrees)



N Queens and Incremental Goals

- Place N queens onto an NxN chess board such that no queen attacks any other queen; a more complicated problem is a Costas Array
- States
  - Place a queen down
- Goal
  - No queen attacks any other queen
  - Hard sometimes to see a good pathway to the goal
- Need a measure in order to see a good pathway
  - Minimize number of queens attacking

Tree Search



- Recursive definition of a tree search as DFS
- SearchTree(State, Move, visited list)
  - Apply Move to the New State
  - If (New State == goal) return TRUE
  - Else
    - If (neighbouring states are empty) return FALSE
    - For each of the MOVE on neighbouring states not in visited list
      - If SearchTree(Neighbouring states, MOVE, visited list + State) return TRUE;

Backtracking Searches

- Move towards the goal, if you reach a position which is going to be less successful – stop and move back up the tree (backtrack, retrace your steps)
- Recursive definition of a depth-first search of a backtrack:
- SearchTreeBacktrack(State, Move, visited list)
  - Apply Move to the New State
  - If (New State == goal) return TRUE
  - If (New State causes an invalid path to goal/or costs too much) return FALSE
  - Else
    - If (neighbouring states are empty) return FALSE
    - For each of the MOVE on neighbouring states not in visited list
      - If SearchTreeBacktrack(Neighbouring states, MOVE, visited list + State) return TRUE;

Sudoku

	1		4		2		5	
5								6
			3		1			
7		5				4		8
2		8				5		9
			9		6			
6								2
	7		1		3		4	

Backtrack on Sudoku

- isValid
  - Check if we have not violated the rules
    - 1-9 in each row
    - 1-9 in each column
    - 1-9 in each box
- Choice at each empty box 1-9
- Multiple Solutions?

5	3	1	2	7	6	8	9	4
6	2	4	1	9	5	2		
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

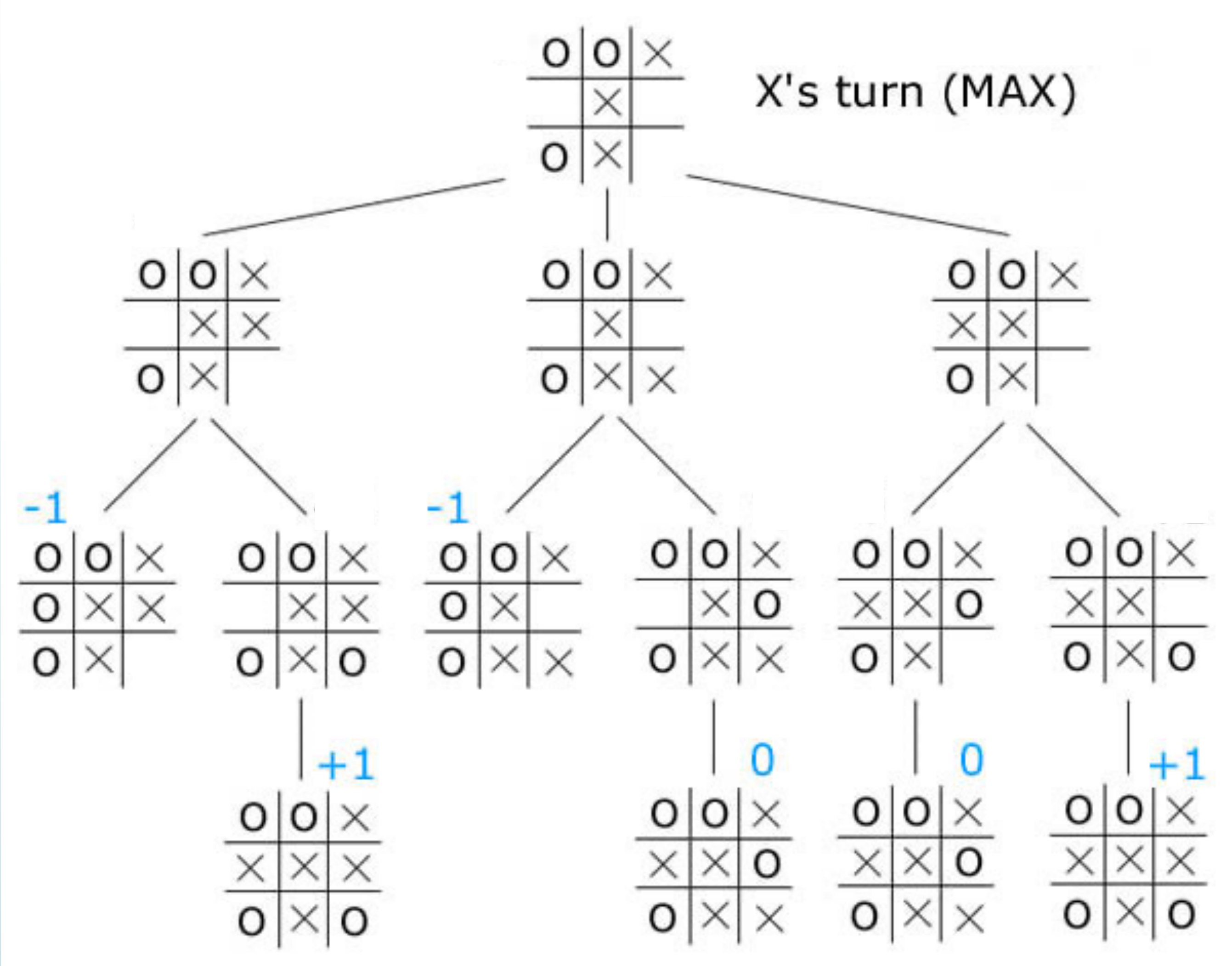
Heuristic Search

- Word comes from the Greek for discovery
- Heuristics use an educated guess on where to move the search next – rules-based movement about a search space
- Not necessarily deterministic – may have stochastic generation methods
- Trial and Error
  - Experimental and Experiential
  - Does not guarantee the most optimal solution
- Examples
  - Greedy and Hill Climbers are heuristics

Minimax

- Opponent goal is to reduce winnings (MINI-Move) and player goal is to increase winnings (MAX-Move)
- Game with a scoring system or win lose condition which can be evaluated
  - GO
  - Chess
- Game Tree
  - Selection of plays in a turn-based game can be reduced to looking at a pathway on a tree
  - Used as part of the theoretical foundations of GAME THEORY

Minimax Example

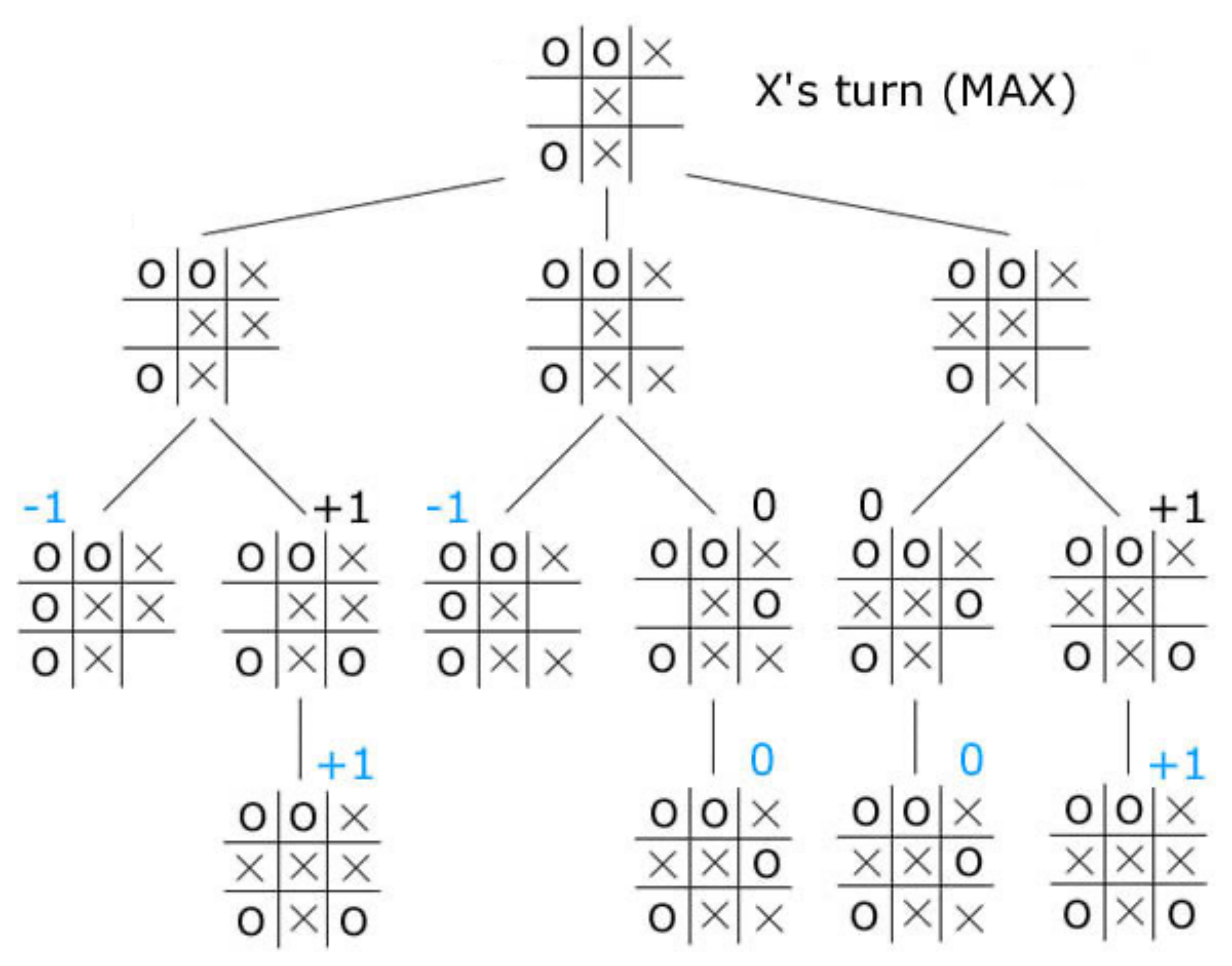


X’s turn (MAX)

0’s turn (MIN)

X’s turn (MAX)

Minimax Example

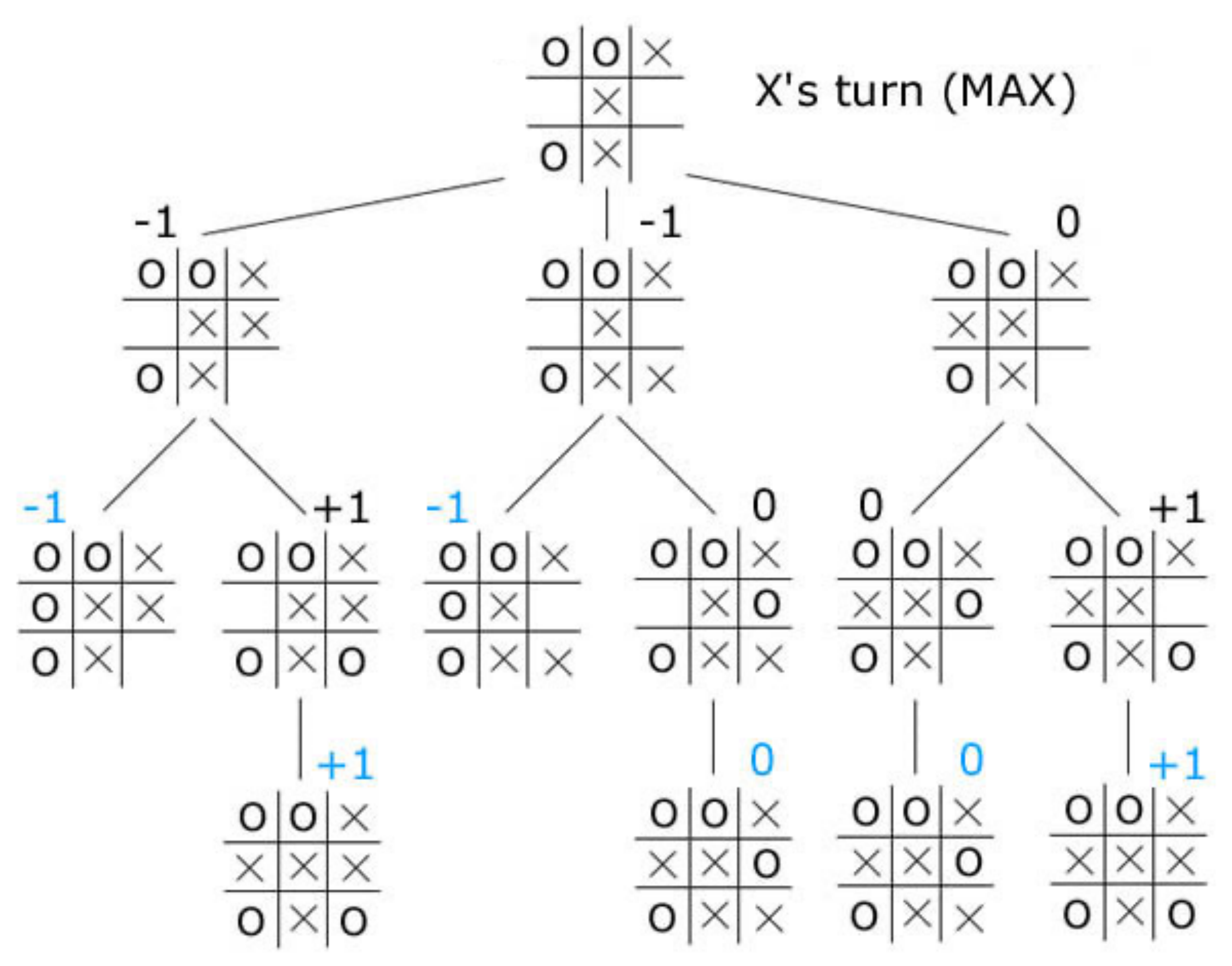


X’s turn (MAX)

0’s turn (MIN)

X’s turn (MAX)

Minimax Example

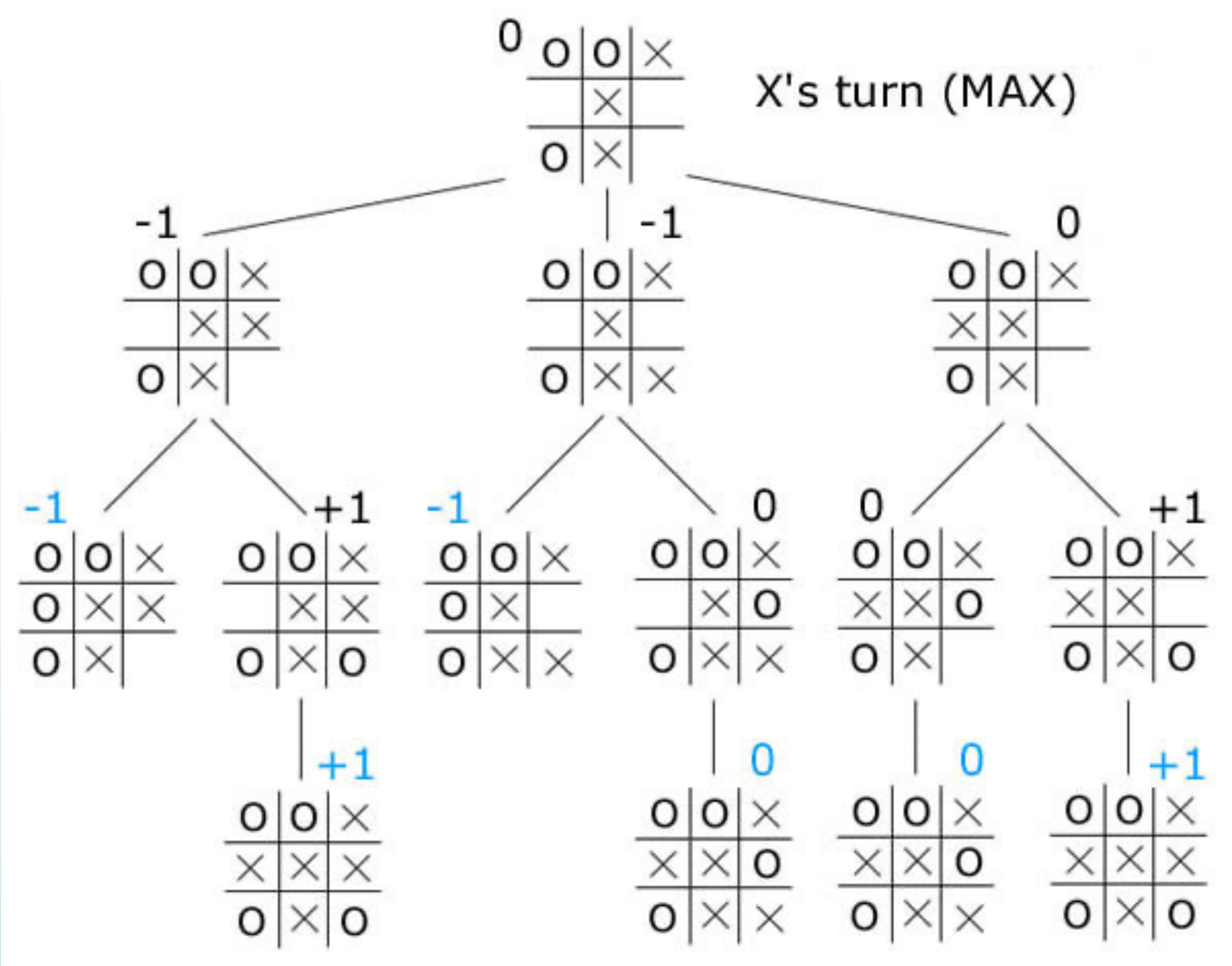


X's turn (MAX)

0's turn (MIN)

X's turn (MAX)

Minimax Example



X’s turn (MAX)

0’s turn (MIN)

X’s turn (MAX)

Backtracking

- Simulate the entire game
- Assume rational/perfect play from the opponent
- Theoretically there is a perfect game play method if we search the entire game
- Solved Games
  - Tic Tac Toe
  - Checkers
  - Two player heads-up limit poker – Recent discovery by researchers in University of Alberta
- Unsolved
  - Chess
  - Go

Backtracking with Alpha Beta Cuts

- Heuristic search based on backtracking
- Used in situations where there is a value to the solution at each point to allow for an evaluation
- Applied for Game Trees
- Alpha Cut
  - Maximum lower bound (Max Plays - Black)
- Beta Cut
  - Minimum upper bound (Min Play - Red)

Cut Examples