



● Game Playing

Types of Games in AI:

	Deterministic	Chance Moves
Perfect information	Chess, Checkers, go, Othello	Backgammon, monopoly
Imperfect information	Battleships, blind, tic-tac-toe	Bridge, poker, scrabble, nuclear war

Perfect information: A game with the perfect information is that in which agents can look into the complete board. Agents have all the information about the game, and they can see each other's moves also. Examples are Chess, Checkers, Go, etc.

Imperfect information: If in a game agents do not have all information about the game and are not aware of what's going on, such types of games are called the game with imperfect information, such as tic-tac-toe, Battleship, blind, Bridge, etc.

Deterministic games: Deterministic games are those games which follow a strict pattern and set of rules for the games, and there is no randomness associated with them. Examples are chess, Checkers, Go, tic-tac-toe, etc.

Non-deterministic games: Non-deterministic are those games which have various unpredictable events and have a factor of chance or luck. This factor of chance or luck is introduced by either dice or cards. These are random, and each action response is not fixed. Such games are also called stochastic games.

Example: Backgammon, Monopoly, Poker, etc.

Zero-Sum Game

- Zero-sum games are adversarial search which involves pure competition.
- In Zero-sum games each agent's gain or loss of utility is exactly balanced by the losses or gains of utility of another agent.
- One player of the game tries to maximize one single value, while another player tries to minimize it.



- Each move by one player in the game is called ply.
- Chess and tic-tac-toe are examples of a Zero-sum game.

Zero-sum game: Embedded thinking

The Zero-sum game involved embedded thinking in which one agent or player is trying to figure out:

- What to do.
- How to decide the move
- Needs to think about his opponent as well
- The opponent also thinks what to do

Each of the players is trying to find out the response of his opponent to their actions. This requires embedded thinking or backward reasoning to solve the game problems in AI.

Formalization of the problem:

A game can be defined as a type of search in AI which can be formalized of the following elements:

- Initial state: It specifies how the game is set up at the start.
- Player(s): It specifies which player has moved in the state space.
- Action(s): It returns the set of legal moves in state space.
- Result(s, a): It is the transition model, which specifies the result of moves in the state space.
- Terminal-Test(s): Terminal test is true if the game is over, else it is false at any case. The state where the game ends is called terminal states.
- Utility(s, p): A utility function gives the final numeric value for a game that ends in terminal states s for player p . It is also called payoff function. For Chess, the outcomes are a win, loss, or draw and its payoff values are +1, 0, $\frac{1}{2}$. And for tic-tac-toe, utility values are +1, -1, and 0.



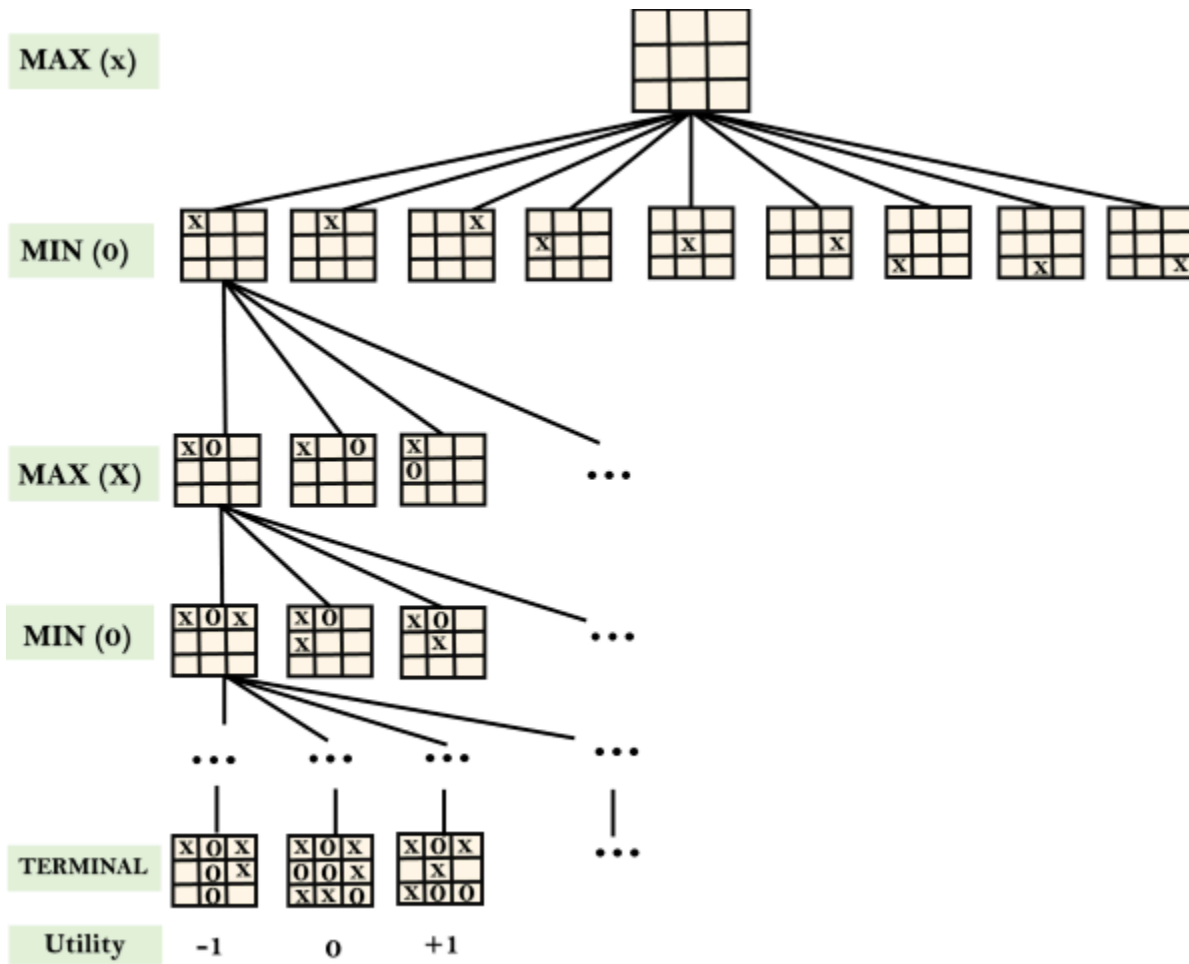
Game tree:

A game tree is a tree where nodes of the tree are the game states and Edges of the tree are the moves by players. Game tree involves initial state, actions function, and result Function.

Example: Tic-Tac-Toe game tree:

The following figure shows part of the game-tree for tic-tac-toe games. Following are some key points of the game:

- There are two players MAX and MIN.
- Players have an alternate turn and start with MAX.
- MAX maximizes the result of the game tree
- MIN minimizes the result.

Example Explanation:

From the initial state, MAX has 9 possible moves as he starts first. MAX place x and MIN place o, and both players play alternatively until we reach a leaf node where one player has three in a row or all squares are filled.

Both players will compute each node, minimax, the minimax value which is the best achievable utility against an optimal adversary.

Suppose both the players are well aware of the tic-tac-toe and playing the best play. Each player is doing his best to prevent another one from winning. MIN is acting against Max in the game.



So in the game tree, we have a layer of Max, a layer of MIN, and each layer is called as Ply. Max places x, then MIN puts o to prevent Max from winning, and this game continues until the terminal node.

In this either MIN wins, MAX wins, or it's a draw. This game-tree is the whole search space of possibilities that MIN and MAX are playing tic-tac-toe and taking turns alternately.

● Adversarial Search Techniques

Adversarial search is a search, where we examine the problem which arises when we try to plan ahead of the world and other agents are planning against us.

In previous topics, we have studied the search strategies which are only associated with a single agent that aims to find the solution which is often expressed in the form of a sequence of actions.

But, there might be some situations where more than one agent is searching for the solution in the same search space, and this situation usually occurs in game playing.

The environment with more than one agent is termed a multi-agent environment, in which each agent is an opponent of another agent and playing against each other. Each agent needs to consider the action of another agent and the effect of that action on their performance.

So, Searches in which two or more players with conflicting goals are trying to explore the same search space for the solution, are called adversarial searches, often known as Games.

Games are modeled as a Search problem and heuristic evaluation function, and these are the two main factors which help to model and solve games in AI.

● Mini-Max Algorithm in Artificial Intelligence

Mini-max algorithm is a recursive or backtracking algorithm which is used in decision-making and game theory. It provides an optimal move for the player assuming that the opponent is also playing optimally.

The Min-Max algorithm uses recursion to search through the game-tree.

The Min-Max algorithm is mostly used for game playing in AI. Such as Chess, Checkers, tic-tac-toe, go, and various two-players games. This Algorithm computes the minimax decision for the current state.

In this algorithm two players play the game, one is called MAX and other is called MIN.