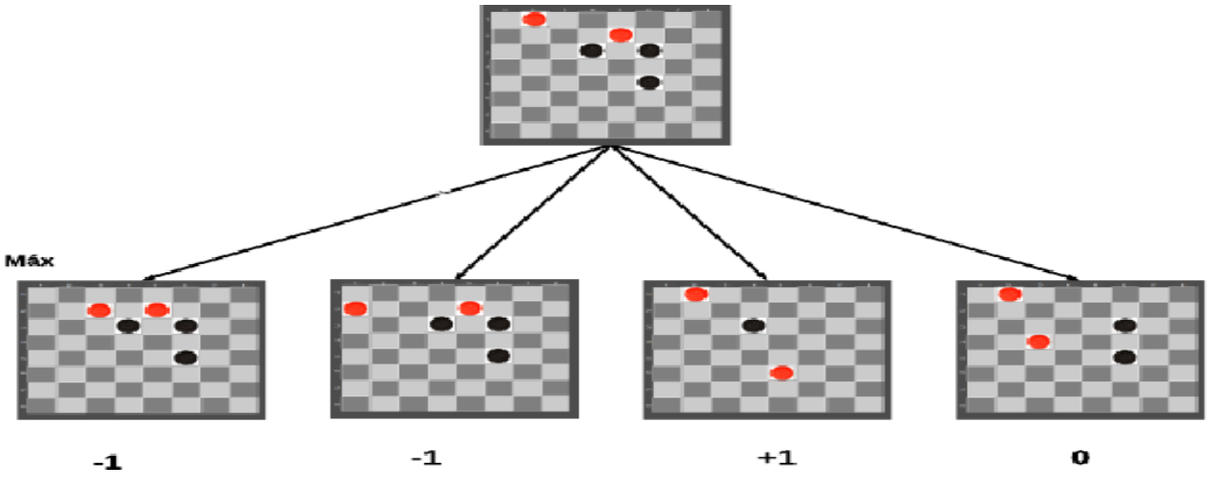
Introduction and overview



- Checkers is a very popular game all over the world. The first attempts to build the first English draughts computer program were in the early 1950s. In 2007, it was published that the program “Chinook” was able to “solve” the 8X8 board from all possible positions

- Checkers, which is also known as Draughts, is a strategy board game that involves diagonal moves of game pieces and capture of opponent’s pieces by jump moves. The game is played by two players with 12 pieces each. The name Draughts means “to draw” or “to move.” The main objective is to capture as many of the opponent’s pieces as possible until the opponent has no more potential moves. Also, another objective is to get to the opponent’s side of the board in order to crown their pieces king along the way.

- The game of checkers is considered a complicated game with 10^20 possible legal positions in the English draughts version (8*8 board) alone (much more on higher dimensions). Our approach is to create a computer agent based on the Minimax algorithm, together with possible improvements, which is in one-on-one games. We start from implementing a basic minimax player with a basic evaluation heuristic, and step by step we improve the pruning by implementing alpha-beta pruning and in addition, we improve our evaluation

function to reach better approximation of the value of the position for our computer player.

Main functionalities

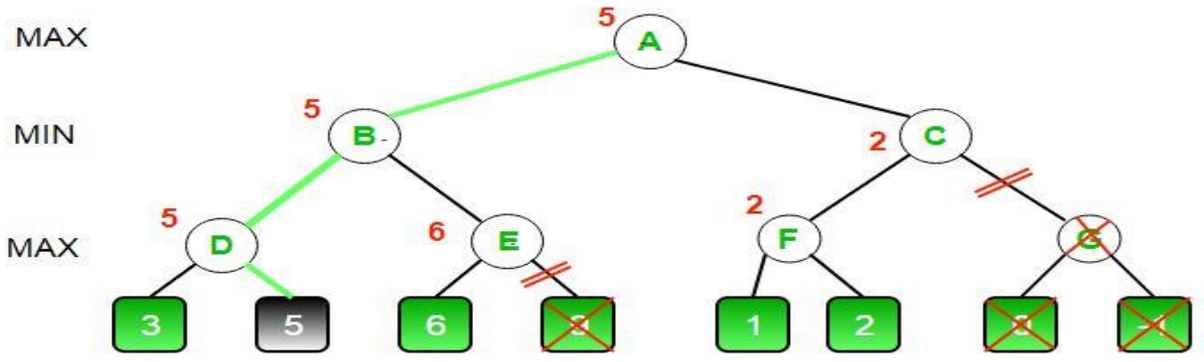
- System shall be able to play against a single player (human player).
- System shall be able to keep check of the valid moves(state space search).
- The system shall be able to keep check of the invalid moves.
- System shall be able to keep track of currently which player has turn either player 1 or player 2.
- System will keep track of the valid kill moves.
- System shall be able not to allow the player to take wrong moves.
- System shall be able to tell which player has won the game.

Similar applications in the market

machine playing of two-player games

1. Tic-tac-toe
2. Chess
3. Connect 4
4. isola
5. Security games, for example in these "games" we want to use AI to find a strategy to protect an airport. It is reasonable to try and design our model under the assumption that someone else wants to break it. You could use Alpha-Beta pruning here (although in practice, more sophisticated algorithms are used).

An initial literature review of Academic publications



- Games have remained through decades an interesting topic for both classical artificial intelligence (AI) based approaches and computational intelligence-based approaches. In 1965 chess was even announced as 'the *Drosophila* of artificial intelligence' by the Russian mathematician Alexander Kronrod.

  Although most examples of the application of computational intelligence methods to game playing make use of either reinforcement learning methods or neural networks, there are also some papers devoted to evolutionary or hybrid neuro-genetic solutions

  checkers (**Aleksander, 1999; Chellapilla & Fogel, 1999**)

- an approach to the Alpha-Beta Pruning is looked. The main idea is to analyze the algorithm itself as an optimization for the Minimax.

In game theory, optimization becomes significant when the possibilities are too many to consider all of them. This minimax algorithm is used specifically for those two non-cooperative player games whereas punctuation is used to set the winner.

The game tree of the Minimax shows every possibility that the first player can do and all the counter-plays that the second player would do; in that way, the first player is trying to maximize his punctuation while the second player tries to minimize it. Among all the possibilities, there are the good ones and the bad ones that are useless for the main player; thus, it is necessary to cut those useless branches of the tree in order to avoid the program computing them unnecessarily. Here is when the Alpha-Beta Pruning becomes the optimal solution

(PDF) Alpha-Beta Pruning Algorithm: The Intelligence Behind Strategy Games. Available from: https://www.researchgate.net/publication/360872512_Alpha-Beta_Pruning_Algorithm_The_Intelligence_Behind_Strategy_Games

- Minimax Algorithm is a solution to reduce the burden on hardware in chess engine. However, more in-depth methods needed to further increase the search algorithm. One of those solutions is called Alpha-Beta Pruning algorithm. The idea is to eliminate the unnecessary nodes in the search tree

  (PDF) APPLYING ALPHA-BETA ALGORITHM IN A CHESS ENGINE. Available from: https://www.researchgate.net/publication/319390201_APPLYING_ALPHA-BETA_ALGORITHM_IN_A_CHESS_ENGINE

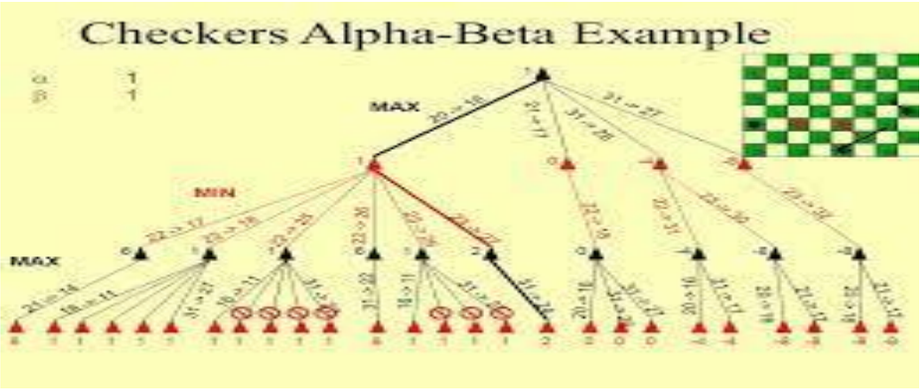
- Alpha beta pruning is an optimization technique for the minimax algorithm. Through the course of this blog, we will discuss what alpha beta pruning means, we will discuss minimax algorithm, rules to find good ordering, and more.

  The word ‘pruning’ means cutting down branches and leaves. In data science pruning is a much-used term which refers to post and pre-pruning in decision trees and random forest

  Alpha-beta pruning is nothing but the pruning of useless branches in decision trees. This alpha-beta pruning algorithm was discovered independently by researchers in the 1900s. (Article by Great learning team).

- Many game-playing programs must search very large game trees. Use of the alpha-beta pruning algorithm instead of the simple minimax search reduces by a large factor the number of bottom positions which must be examined in the search. An analytical expression for the expected number of bottom positions examined in a game tree using alpha-beta pruning is derived, subject to the assumptions that the branching factor N and the depth D of the tree are arbitrary but fixed, and the bottom positions are a random permutation of (N sub-D) unique values. A simple approximation to the growth rate of the expected number of bottom positions examined is suggested, based on a Monte Carlo simulation for large values of N and D. The behavior of the model is compared with the behavior of the alpha-beta algorithm in a chess playing program and the effects of correlation and non-unique bottom position values in real game trees are examined. (Book is Analysis of the Alpha-beta Pruning Algorithm by S. H. Fuller, J. G. Gaschnig, J. J. Gillogly)

Details of the Alpha-Beta Depth-First algorithm



Minimax is a recursive algorithm which is used to choose an optimal move for a player assuming that the other player is also playing optimally.

the key to the Minimax algorithm is a back and forth between the two players, where the player whose "turn it is" desires to pick the move with the maximum score. In turn, the scores for each of the available moves are determined by the opposing player deciding which of its available moves has the minimum score

It is used in games such as tic-tac-toe, go, chess, Isola, checkers, and many other two-player games.

Such games are called games of perfect information because it is possible to see all the possible moves of a particular game.

There are two players involved in a game, called MIN and MAX. The player MAX tries to get the highest possible score and MIN tries to get the lowest possible score, i.e., MIN and MAX try to act opposite of each other.

the minimax algorithm can still be inefficient and may use further optimization

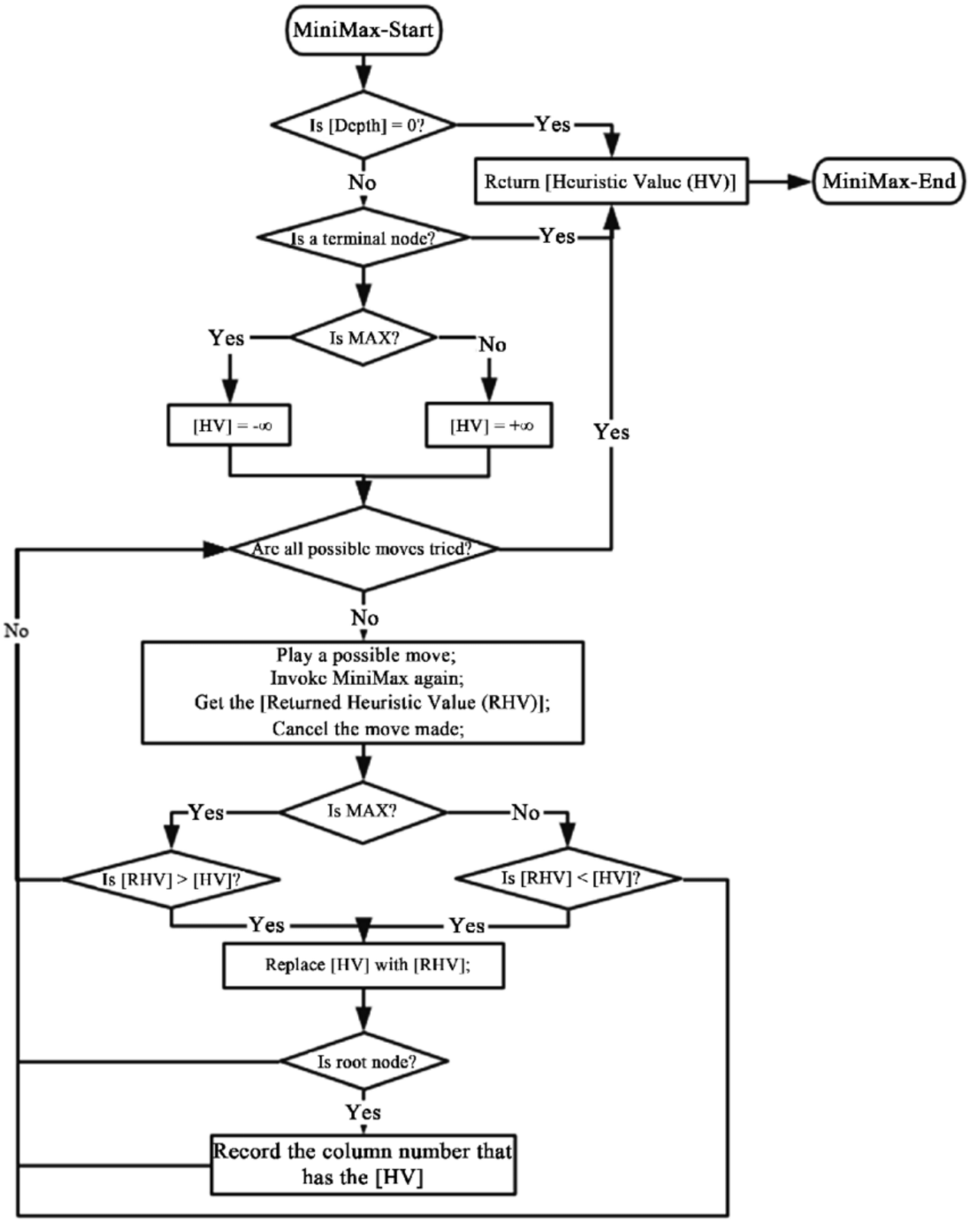
Alpha–beta pruning is a search algorithm that seeks to decrease the number of nodes that are evaluated by the minimax algorithm in its search tree, it is an adversarial search algorithm,

If we apply alpha-beta pruning to a standard minimax algorithm, it returns the same move as the standard one, but it removes (prunes) all the nodes that are possibly not affecting the final decision.

Alpha: It is the best choice so far for the player MAX. We want to get the highest possible value here.
Beta: It is the best choice so far for MIN, and it has to be the lowest possible value.

Note: Each node has to keep track of its alpha and beta values. Alpha can be updated only when it’s MAX’s turn and, similarly, beta can be updated only when it’s MIN’s chance.

How does alpha-beta pruning work?


MiniMax-Start
Is [Depth] = 0?
Yes
No
Return [Heuristic Value (HV)]
MiniMax-End
Is a terminal node?
Yes
Yes
Is MAX?
No
[HV] = -∞
[HV] = +∞
Yes
Are all possible moves tried?
No
No
Play a possible move;
Invoke MiniMax again;
Get the [Returned Heuristic Value (RHV)];
Cancel the move made;
Yes
Is MAX?
No
Is [RHV] > [HV]?
Is [RHV] < [HV]?
Yes
Yes
Replace [HV] with [RHV];
Is root node?
Yes
Record the column number that has the [HV]

```
// Search game tree to given depth, and return evaluation of
// root node.
int AlphaBeta(gamePosition, depth, alpha, beta)
{
        if (depth=0 || game is over)
        // evaluate leaf gamePositionition from
        // current player's standpoint
                return Eval (gamePosition);
         // present return value
        score = - INFINITY;
        // generate successor moves
        moves = Generate(gamePosition);
        // look over all moves
        for i =1 to sizeof(moves) do
        {
                // execute current move
                Make(moves[i]);
                // call other player, and switch sign of
                // returned value
                cur = -AlphaBeta(gamePosition, depth-1,
                                        -beta, -alpha);
                // compare returned value and score
                // value, update it if necessary
                if (cur > score) score = cur;
                // adjust the search window
                if (score > alpha) alpha = score;
                // retract current move
                Undo(moves[i]);
                 // cut off
                if (alpha >= beta) return alpha;
        }
        return score;
}
```

Block Diagram

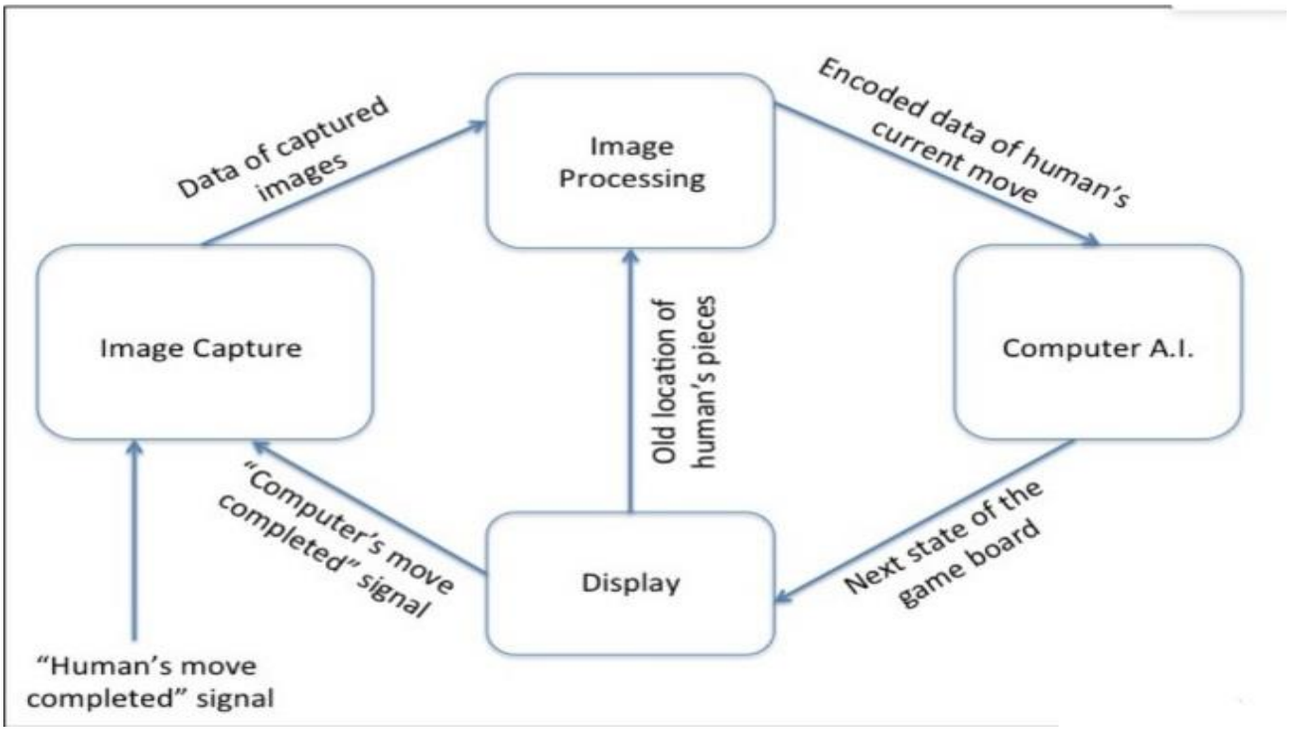
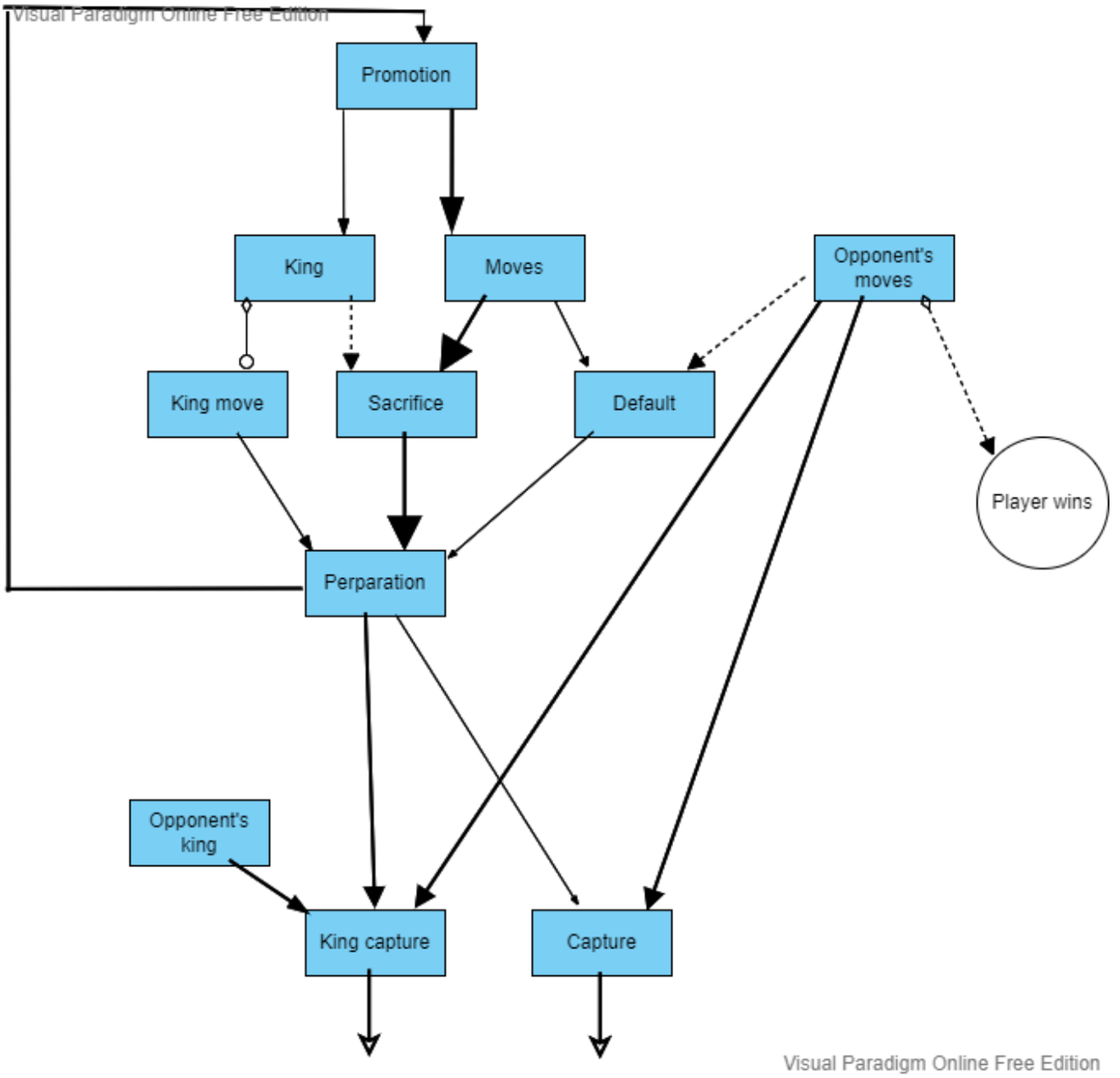
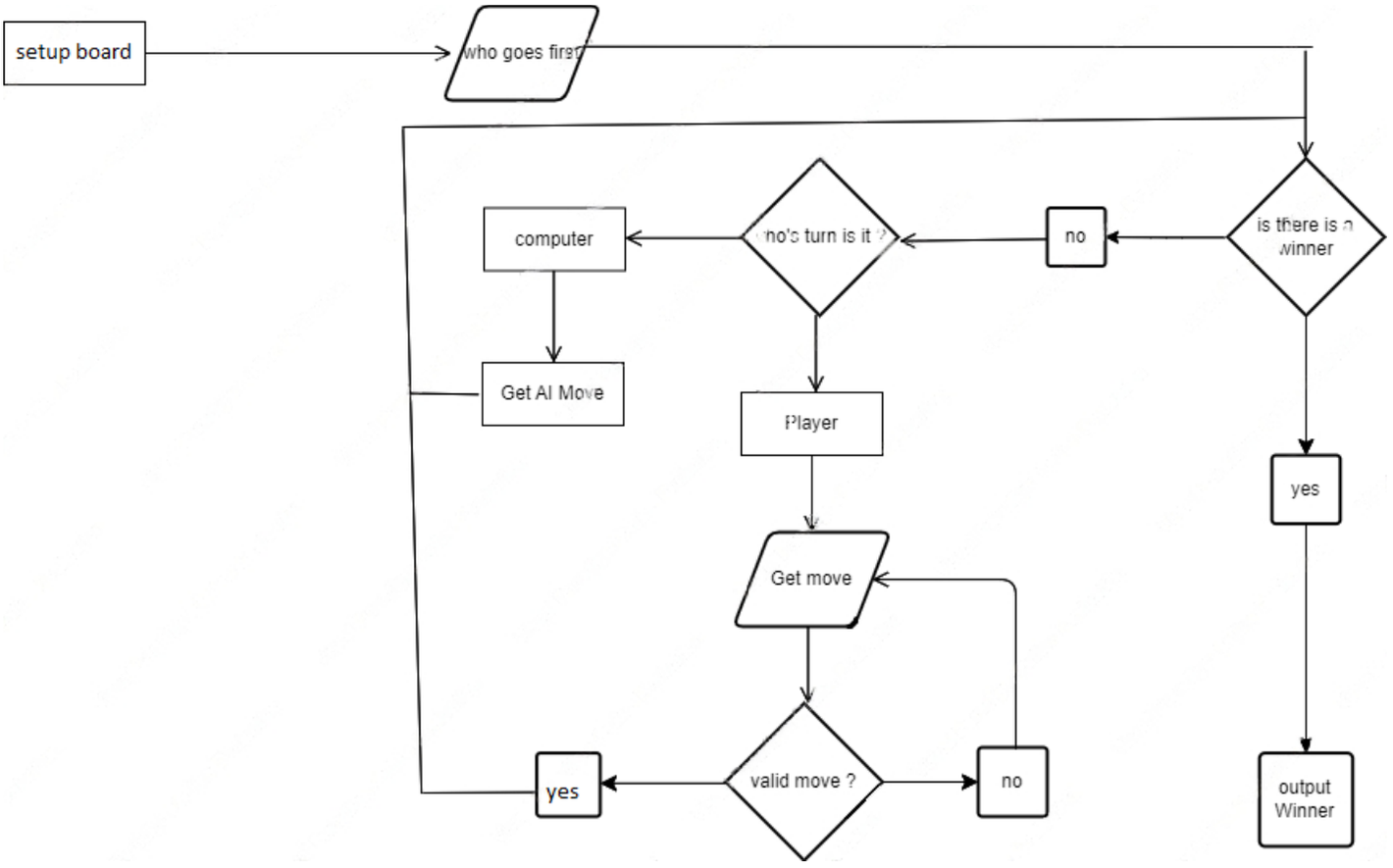


Image Processing
Data of captured images
Encoded data of human's current move
Image Capture
Computer A.I.
Old location of human's pieces
"Computer's move completed" signal
Next state of the game board
Display
"Human's move completed" signal



Flow Chart Diagram


setup board
who goes first
is there is a winner
yes
output Winner
no
who's turn is it ?
computer
Get AI Move
Player
Get move
valid move ?
no
yes

Use-Case Diagram