a. States can represent board positions/states(the marks and empty boxes on the board), action will be the move that the agent makes. Which in Tic-Tac-Toe would be to mark an empty box with either ‘x’ or ‘o’. The current state of the game affects the actions in a way that the opponents previous move can significantly affect the current action agent will take. For example in a situation where based on the opponent's move, the opponent could be 1 mark away from reaching the goal state, that will and should lead the agent to make a move that prevents the opponent from reaching the goal state.

b. The reward function takes the board(with its current state), and a mark type (either ‘x’ or ‘o’). Then it searches for 3 consecutive matches of a mark type. It will be searching for vertical, horizontal and diagonal matches. If it finds a match then it will compare a mark from the match to the input mark type, if it is the same it will return 1, if it differs then -1. If it cannot find a match then it returns 0. Note that this function does not differentiate between draw state and a mid game or start game state, it returns 0 in either of those cases. This function is simple but it captures the essence of a reward function for Tic-Tac-Toe. The objective of the reward function for Tic-Tac-Toe is to determine whether the board is in a winning state or a losing state or neither for the queried player. Provided function captures all possible states, and returns a representative value for each state. Given the objective of the reward function, or through the lens of the objective function, there isn’t a difference between a board state that's a draw, a midgame or start game scenario.

c. Instead of searching for an end state or a goal state, the function could be adjusted to analyze the current state and somehow try to quantify the current state. For example, Chess is a much more complicated game than Tic-Tac-Toe that's played on a larger board. Instead of trying to analyze a sequence of best moves that will eventually lead to an end state, the function could look for opportunities or threats that would put the agent in an advantageous or disadvantageous position. In the simplest terms, the function could search for certain patterns that create advantages or disadvantages. As a specific example, one thing the function could be searching for can be forks. A fork is a basic chess tactic in which a piece attacks multiple enemy pieces simultaneously. Overall, the essence of my suggestion is to design a function that evaluates the reward by trying to quantify advantages and disadvantages that the current state or the current state and next few states holds.
    i. To further clarify, this concept function takes the board and quantifies a score for the queried player. In a practical application, one could iterate through all the possible next actions by temporarily adding it to the board and querying this function, and eventually choosing the one with the highest score.