3X3 TIC-TAC-TOE (ALGORITHM)

PROFESSOR: TAEHYUNG (GEORGE) WANG

COMP 469 ARTIFICIAL INTELLIGENCE

PIETRO CICCIARI
MARIO CHOTO
ISRAEL SANCHEZ
RAJ KUMAR
ARMON LEE
VENKAT SAIRAM RAVALA

PROBLEM STATEMENT AND REQUIREMENTS

Problem:

Solve the Tic-Tac-Toe game using a goal-based agent.

Objective:

The agent aims to either win the game or prevent the opponent from winning.

PEAS Framework:

- Performance: The agent's performance is measured by its ability to win or, at least, not lose the game (Ideally).
- Environment: The environment is the 3x3 Tic-Tac-Toe board where the game is played.
- Actuators: The actions of the agent include placing an "X" or "O" on the board.
- Sensors: The agent uses sensors to see the current state of the board and to check where the opponent has placed their symbol.

PROPERTIES OF THE TASK ENVIRONMENT

Fully Observable: Agent can see the entire board.

Deterministic:
The outcome of placing a symbol is predictable.

Sequential:
Turn-based gameplay.

Static:
Board only changes when a player places a symbol.

Discrete and Known:
Fixed number of squares and well-known rules.

FORMAL SPECIFICATIONS

POSSIBLE STATES:

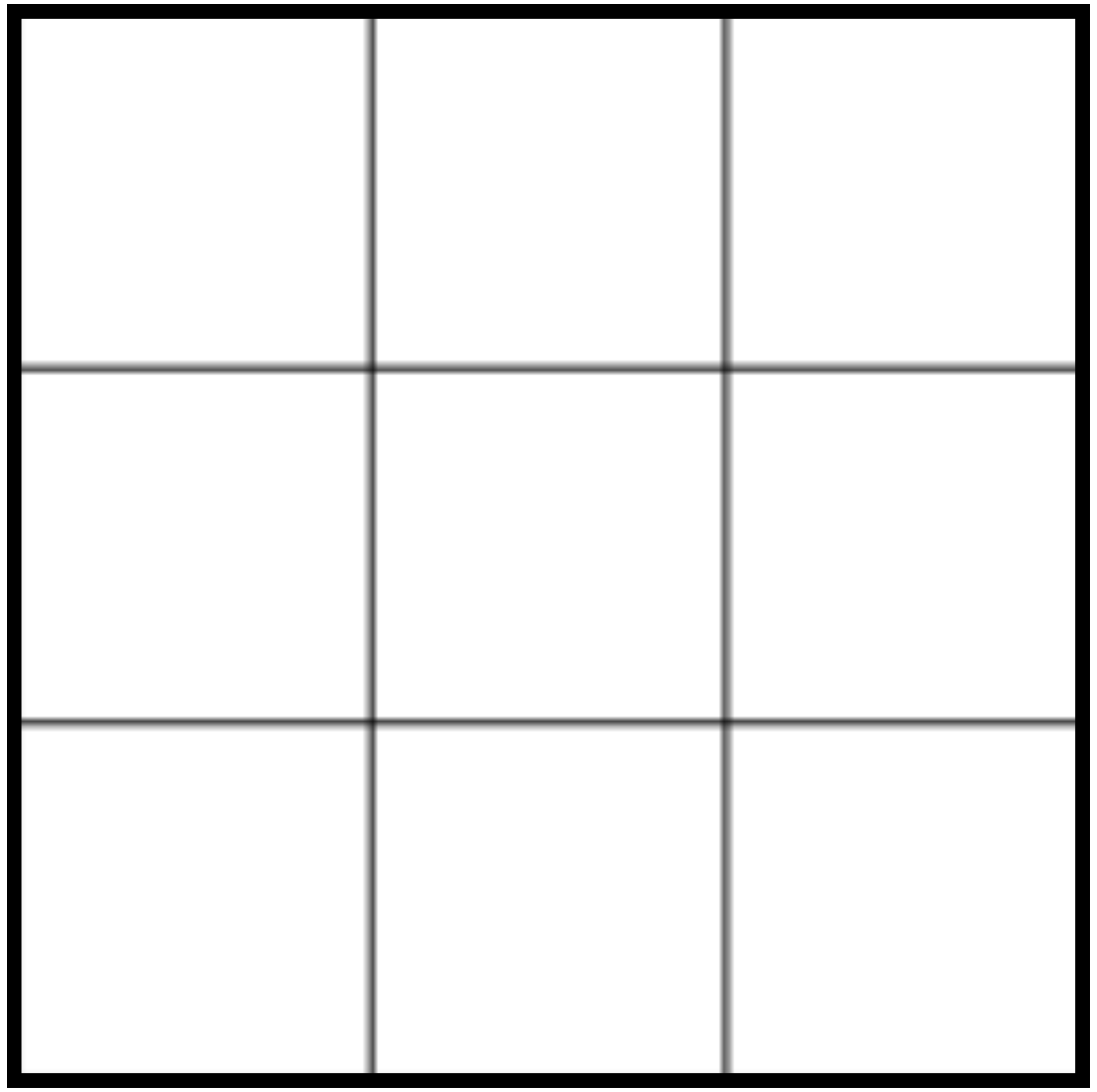
All configurations of symbols on the 3x3 grid.

INITIAL STATE

Empty board.

GOAL STATE:

Three symbols in a row, column, or diagonal.



ACTIONS

Placing a symbol in any empty square.

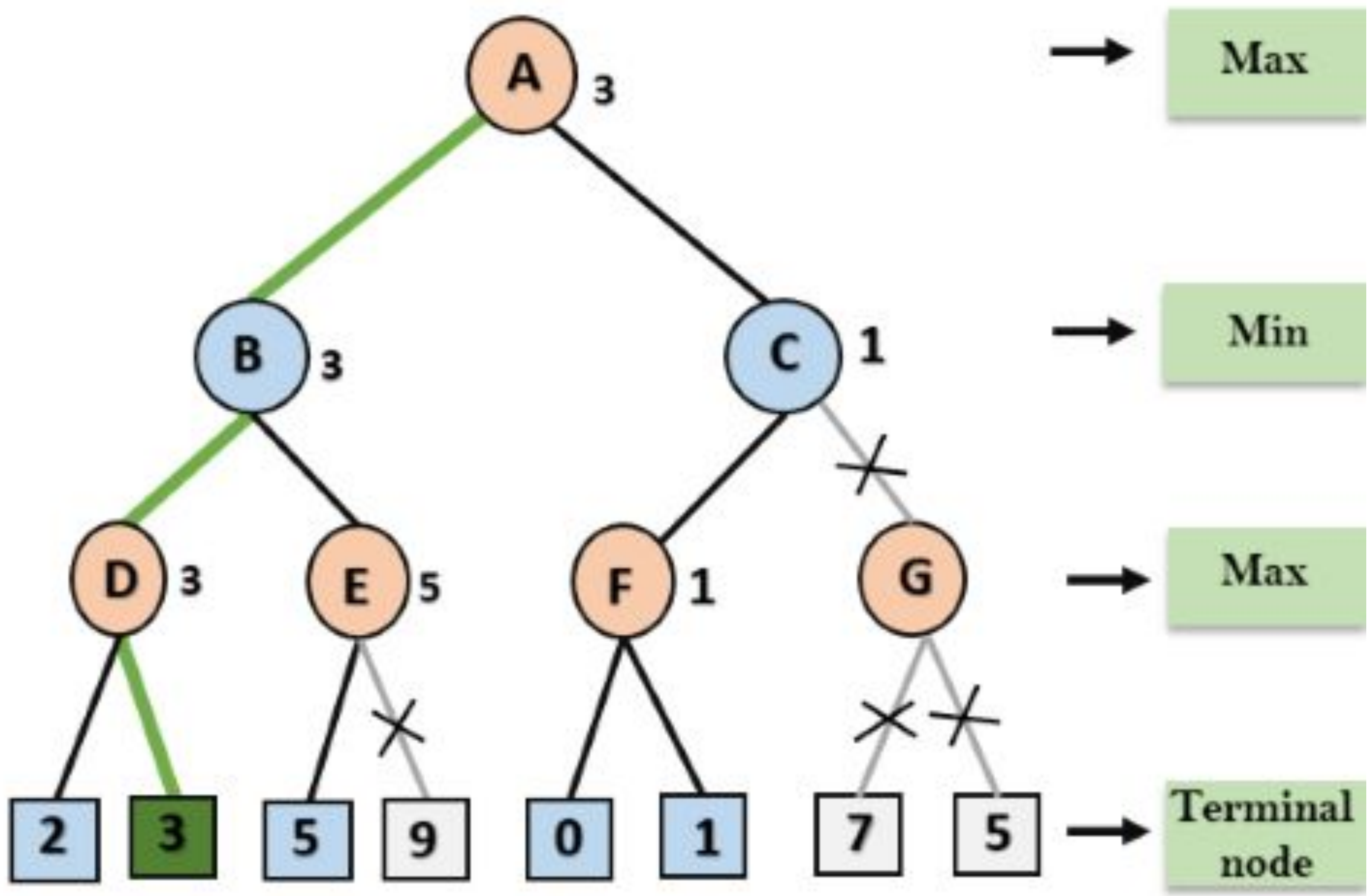
TRANSITION MODEL:

State changes when a symbol is placed.

APPROACH AND DESIGN - MINIMAX ALGORITHM

EXPLANATION:

The Minimax algorithm looks ahead at possible moves and outcomes, aiming to minimize the worst-case scenario and maximize the best-case scenario.



HEURISTIC FUNCTION:

Evaluates proximity to winning or blocking the opponent.

CODE IMPLEMENTATION OVERVIEW

IMPLEMENTATION

Code uses Minimax algorithm for decision-making.

PSEUDOCODE

General structure of the algorithm (details on the full report).

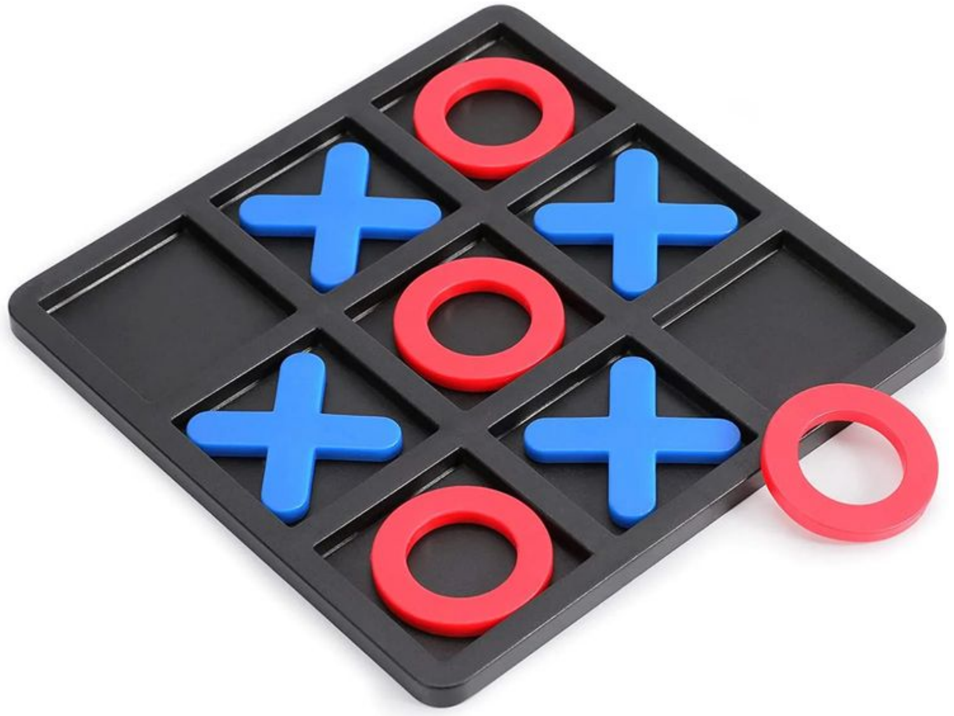
PROGRAMMING LANGUAGE:

Python (with error handling for Pygame initialization and display setup).

INPUT AND OUTPUT

INPUT PROCESSING

Player clicks on a square, program checks availability and updates the board.



OUTPUT GENERATION

Minimax algorithm determines computer's move, game displays outcomes like "Player X wins!", "Player O wins!", or "It's a draw!".

TEST CASES

2

Edge Cases:

Situations where the board is nearly full, and the computer must decide between winning, blocking, or forcing a draw.

1

Standard Cases:

Examples of typical game scenarios and how the computer responds.

PERFORMANCE ANALYSIS

TIME COMPLEXITY:

Manageable due to the small board size.

COMPLETENESS:

The algorithm always finds the best move.

Space Complexity:

Feasible to store all possible game states.

Cost Optimality:

Optimal for winning or drawing.

PERFORMANCE ANALYSIS

SUMMARY OF RESULTS:

Minimax consistently finds the best possible moves, leading to either a win or a draw.

EFFECTIVENESS AND EFFICIENCY

Well-suited for Tic-Tac-Toe, but larger games may require optimization like alpha-beta pruning

Q&A