GOAL FORMULATION

The goal of all versions of the tic-tac-toe-playing agent is to form one of the winning configurations before the opponent, thus winning the game; this is the only goal of the level 0 agent. The levels 1, 2, and 3 agents have the additional goal of actively preventing their opponent from completing a winning configuration, and the levels 4 and 5 agents additionally aim to win in the least number of turns.

PROBLEM FORMULATION

As the level 0 agent chooses its moves randomly, it is only concerned with determining the list of positions where its next token can be placed. Meanwhile, the levels 1, 2, and 3 agents search for sequences of moves that would lead to a winning configuration, which may necessitate preventing the opponent from obtaining such a configuration. In cases where multiple move sequences may produce a win, the levels 4 and 5 agents additionally find the shortest sequences that would lead to a win.

AGENT STATES

All versions of the agent have three states: the initial state where the board is still blank, the active state where neither player has won yet and thus takes turns placing their tokens on the board, and the terminal state, whereupon additional tokens can no longer be placed.

Moreover, the terminal state can be divided into three categories. A *win*, or goal state, is reached when the agent completes one of the winning configurations before the opponent. A *loss* is reached if the opponent completes a winning configuration before the agent, and a *draw* is reached if both players failed to achieve a winning configuration after the entire board has been occupied. Symbolically,

State = $(x_{1,1}, x_{1,2}, x_{1,3}, x_{2,1}, x_{2,2}, x_{2,3}, x_{3,1}, x_{3,2}, x_{3,3})$, where $x_{i,j}$ is the token on the ith row and jth column of the board. Since it is turn-based, the number of tokens of the first player is always equal to or exactly one greater than the number of tokens of the other player.

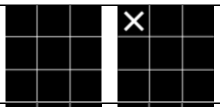
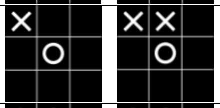
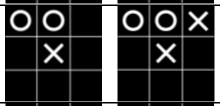
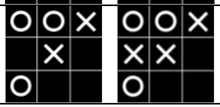
Initial State = (0, 0, 0, 0, 0, 0, 0, 0, 0). In this technical report, '0' is used to denote a blank tile.

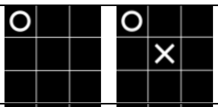
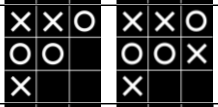
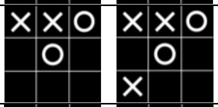
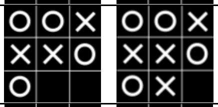
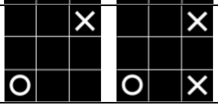
Goal State = one of the eight states presented below, where 'X' represents the agent's token and * refers to any valid value:

(X, X, X, *, *, *, *, *, *)	First row	(*, X, *, *, X, *, *, X, *)	Second column
(*, *, *, X, X, X, *, *, *)	Second row	(*, *, X, *, *, X, *, *, X)	Third column
(*, *, *, *, *, *, X, X, X)	Third row	(X, *, *, *, X, *, *, *, X)	Upper-left to lower-right diagonal
(X, *, *, X, *, *, X, *, *)	First column	(*, *, X, *, X, *, X, *, *)	Upper-right to lower-left diagonal

ACTIONS AND TRANSITION TABLE

All levels of rationality of the agent can perform one of nine actions, i.e., placing the agent's token on one of the positions on the board. Assume that 'X' is the agent's token and 'O' is the opposing token.

Name	Condition	Transition	Illustration	Effect
Place on (1, 1)	$x_{1,1} = 0$ and not at terminal state	$(0, x_{1,2}, x_{1,3}, x_{2,1}, x_{2,2}, x_{2,3}, x_{3,1}, x_{3,2}, x_{3,3}) \rightarrow$ $(X, x_{1,2}, x_{1,3}, x_{2,1}, x_{2,2}, x_{2,3}, x_{3,1}, x_{3,2}, x_{3,3})$		Place token on (1, 1).
Place on (1, 2)	$x_{1,2} = 0$ and not at terminal state	$(x_{1,1}, 0, x_{1,3}, x_{2,1}, x_{2,2}, x_{2,3}, x_{3,1}, x_{3,2}, x_{3,3}) \rightarrow$ $(x_{1,1}, X, x_{1,3}, x_{2,1}, x_{2,2}, x_{2,3}, x_{3,1}, x_{3,2}, x_{3,3})$		Place token on (1, 2).
Place on (1, 3)	$x_{1,3} = 0$ and not at terminal state	$(x_{1,1}, x_{1,2}, 0, x_{2,1}, x_{2,2}, x_{2,3}, x_{3,1}, x_{3,2}, x_{3,3}) \rightarrow$ $(x_{1,1}, x_{1,2}, X, x_{2,1}, x_{2,2}, x_{2,3}, x_{3,1}, x_{3,2}, x_{3,3})$		Place token on (1, 3).
Place on (2, 1)	$x_{2,1} = 0$ and not at terminal state	$(x_{1,1}, x_{1,2}, x_{1,3}, 0, x_{2,2}, x_{2,3}, x_{3,1}, x_{3,2}, x_{3,3}) \rightarrow$ $(x_{1,1}, x_{1,2}, x_{1,3}, X, x_{2,2}, x_{2,3}, x_{3,1}, x_{3,2}, x_{3,3})$		Place token on (2, 1).

Place on (2, 2)	$x_{2,2} = 0$ and not at terminal state	$(x_{1,1}, x_{1,2}, x_{1,3}, x_{2,1}, 0, x_{2,3}, x_{3,1}, x_{3,2}, x_{3,3}) \rightarrow (x_{1,1}, x_{1,2}, x_{1,3}, x_{2,1}, \text{X}, x_{2,3}, x_{3,1}, x_{3,2}, x_{3,3})$		Place token on (2, 2).
Place on (2, 3)	$x_{2,3} = 0$ and not at terminal state	$(x_{1,1}, x_{1,2}, x_{1,3}, x_{2,1}, x_{2,2}, 0, x_{3,1}, x_{3,2}, x_{3,3}) \rightarrow (x_{1,1}, x_{1,2}, x_{1,3}, x_{2,1}, x_{2,2}, \text{X}, x_{3,1}, x_{3,2}, x_{3,3})$		Place token on (2, 3).
Place on (3, 1)	$x_{3,1} = 0$ and not at terminal state	$(x_{1,1}, x_{1,2}, x_{1,3}, x_{2,1}, x_{2,2}, x_{2,3}, 0, x_{3,2}, x_{3,3}) \rightarrow (x_{1,1}, x_{1,2}, x_{1,3}, x_{2,1}, x_{2,2}, x_{2,3}, \text{X}, x_{3,2}, x_{3,3})$		Place token on (3, 1).
Place on (3, 2)	$x_{3,2} = 0$ and not at terminal state	$(x_{1,1}, x_{1,2}, x_{1,3}, x_{2,1}, x_{2,2}, x_{2,3}, x_{3,1}, 0, x_{3,3}) \rightarrow (x_{1,1}, x_{1,2}, x_{1,3}, x_{2,1}, x_{2,2}, x_{2,3}, x_{3,1}, \text{X}, x_{3,3})$		Place token on (3, 2).
Place on (3, 3)	$x_{3,3} = 0$ and not at terminal state	$(x_{1,1}, x_{1,2}, x_{1,3}, x_{2,1}, x_{2,2}, x_{2,3}, x_{3,1}, x_{3,2}, 0) \rightarrow (x_{1,1}, x_{1,2}, x_{1,3}, x_{2,1}, x_{2,2}, x_{2,3}, x_{3,1}, x_{3,2}, \text{X})$		Place token on (3, 3).

GOAL STATE DETERMINATION

To inform its succeeding moves, all levels of rationality of the agent keep a persistent record of the state of the board, which is represented as a two-dimensional 3×3 array of characters under the hood. After every move, the agent compares the contents of this memory with each of the eight possible winning configurations; if at least one of these is met, then the agent has attained the goal state.

Apart from the goal state, the agent also refers to this persistent record of the state to determine whether a draw or loss has been reached (i.e., the opponent won or the board was filled without either player winning). Upon reaching any of the three terminal states, all further actions are disabled.

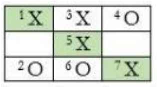
DESCRIPTION AND ANALYSIS OF LEVELS OF INTELLIGENCE

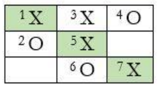
This section describes the six (6) levels of intelligence of the tic-tac-toe-playing agent. The table below shows the average decision time of each agent (in milliseconds) per level (over 10 runs) assuming that the opponent is responding with the optimal moves (with the exception of level 0).

Level	**Number of Remaining (Unoccupied) Positions**								
	9	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
0	0.0221	0.0137	0.0138	0.0185	0.0128	0.0127	0.0127	0.0119	0.0106
1	0.0162	0.1326	0.2061	0.0209	0.0223	0.0167	0.0176	0.0155	0.0909
2	97.6703	9.4607	1.1712	0.2431	0.0412	0.0264	0.0068	0.0056	0.0035
3	3.1408	0.4139	0.1397	0.0179	0.0152	0.0069	0.0052	0.0037	0.0033
4	76.2557	7.2822	1.0876	0.2129	0.0364	0.0171	0.0069	0.0042	0.0030
5	13.6244	2.2356	0.2139	0.0305	0.0151	0.0075	0.0054	0.0038	0.0029

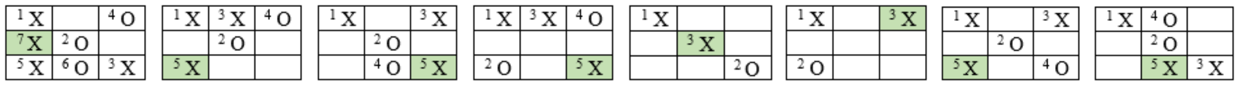
Level 0. This level is characterized by nonrationality, with the agent deciding its moves randomly as long as they are admissible per the rules. More specifically, it employs the 48-bit-seed pseudorandom linear congruential generator from Java's `ThreadLocalRandom` class, which features an $O(1)$ time complexity and registers the least decision time — albeit this is irrelevant due to its lack of intelligence.

Level 1. This marks the first level of rational behavior. In deciding its move, the agent consults a hard-coded table of optimal moves per board configuration (implemented internally as a collection of conditional constructs). Exploiting symmetry and rotations, moves are reduced to the more general categories of corner, edge, and center moves. To maximize the chances of winning, the opening move is always set to a corner [arbitrarily chosen to be (1,1)]. As pointed out by Gardner (1988), this opening leaves a single optimal response to hold a draw: a center move. Replying with an edge or corner move will result in a guaranteed win for the first player (even with perfect play from both sides), as seen in the images on the right:





Following the strategy suggested by Newell and Simon (1972), the succeeding moves are decided based on a set hierarchical order (listed and illustrated below in descending order of priority): three-in-a-lane win, block an opponent's win, generate a fork (there are two unblocked lanes for a three-in-a-lane win), block an opponent's potential fork, center move, opposite corner move, empty corner move, and side move. The forking positions were identified using an online tool by Campbell (2017).

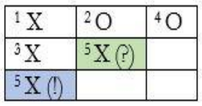


This hierarchical rule-based approach theoretically makes the agent unbeatable. Moreover, since it only involves conditionals, the time complexity of the decision-making per se is a constant $O(1)$, and *a posteriori* results show that its decision time is consistently below a millisecond regardless of the number of unoccupied positions left. However, this brute-force approach comes at the cost of manual strategic analysis and development time; thus, it may not be feasible for larger variants of tic-tac-toe.

From Level 2 onwards, the minimax algorithm (a depth-first search strategy for adversarial games) is used by the agent to ensure optimal moves leading to either a draw (assuming perfect play) or a win. To avoid the horizon effect — wherein a drastic change in the evaluation happens beyond the set maximum depth — no cutoff was enforced. Since the game tree is computationally small, a complete search is feasible; in fact, the empirical results show that no evaluation exceeds a tenth of a second.

Level 2 follows the standard minimax implementation. The utility values of each leaf node are as follows: 100 for a win, –100 for a loss, and 0 for a draw. The minimax values are then backed up the game tree via recursion. To decrease the size of the search space, alpha-beta pruning was employed in **Level 3**. The states are examined with the possible moves arranged in row-major order (the board was represented as a 3×3 array). As seen in the table, pruning results in a significant decrease in decision time. Theoretically, Russell and Norvig (2010) stated that the total number of nodes examined is roughly $O(b^{0.75m})$, where b is the branching factor and m is the maximum depth.

A peculiarity can be noticed in the move sequence on the right. Levels 2 and 3 will choose the green-colored move. Albeit not the immediate win, this move still guarantees a win for the agent since it leads to a horizontal and diagonal fork. As a consequence of the utility value being a static 100 for a winning configuration, it evaluates both green- and blue-colored moves as equally optimal but gives priority to the former due to row-major ordering.



A computationally inexpensive workaround is given in **Levels 4 and 5**, which implement depth-sensitive minimax. Typically, if the current evaluation is greater than the interim maximum (or less than the minimum), this evaluation becomes the new interim extremum. However, the variant in the last two levels sets the new interim maximum as the difference of the evaluation and depth of the leaf node (analogously, the new interim minimum is the sum of the evaluation and depth of the leaf node). As the depth of more immediate winning moves is smaller, the difference between the evaluation (100) and the depth is higher; thus, the (maximizing) agent gives preference to them. In reference to the scenario above, the blue-colored move is an immediate win, evaluated as $100 - 0 = 100$, while the green-colored move still requires an additional move and receives a lower evaluation of $100 - 1 = 99$.

Level 5 is the same as Level 4, but with the addition of alpha-beta pruning. As expected, the reduction in the number of generated subtrees curbs the decision time considerably. As a result of the depth-sensitive strategy affecting the minimax values of certain nodes and deferring the pruning of previously disregarded branches, Level 5 clocks in a longer time than Level 3. Nonetheless, this is a necessary trade-off to improve the agent's rationality as it progresses onto higher levels of intelligence.

Campbell, G. (2017). *Tic tac toe fork generator in C#.* https://gabegamedev.com/2017/05/31/first-blog-post/
Gardner, M. (1998). *Hexaflexagons and other mathematical diversions.* University of Chicago Press.
Newell, A., & Simon, H.A. (1972). *Human problem solving*. Prentice Hall.
Russell, S., & Norvig, P. (2010). *Artificial intelligence: A modern approach* (3rd ed.). Prentice Hall.