HÁSKÓLINN Í REYKJAVÍK
REYKJAVIK UNIVERSITY

Spring 2019

T-662-ARTI

# Project Proposal: Ultimate Tic Tac Toe

Group 4


Alex Már Gunnarsson
Jóhann Egilsson
Pedro Hrafn Martinez

Stephan Schiffel
Sigurður Helgason
Svanur Jóhannesson


## Description

There exists a variant of the game Tic Tac Toe, we call this variant Ultimate Tic Tac Toe. This game is almost identical to the original game but with a major twist. Ultimate Tic Tac Toe has a 3x3 board with a 3x3 board in each cell. The first player picks a cell on the big board where he puts an X in the tiny board. The cell that he picks determines where the second player makes his next move. The move of the second player then also determines the cell the first player makes his next move and so on. When a player wins a match on a small board the board is replaced with the symbol of the winner, empty symbol if the game resulted in a tie. The goal of the game is to place your symbol three times in a horizontal, vertical og diagonal row on the big board.

Our objective would be to implement an agent using min-max algorithm with Alpha-Beta pruning. Similar to the Breakthrough programming assignment this agent will have a time limit to make a move. We will have to experiment with different time limits so that the agent can do reasonably well. The evaluation function for states will also reward 100 points for winning, 0 for draw and -100 for losing, while implementing the project we may change this evaluation but in our opinion it is a good first step.

Since we are to implement an agent here is the PEAS description of that agent in a form of a table.

| **Performance Measure** | **Environment** | **Actuators** | **Sensors** |
|---|---|---|---|
| - Placing three symbols, belonging to the agent, in a row on the big board.<br>- Prevent that the symbols of the opponent get placed in row. | - 3x3 board where each cell is another 3x3 board<br>- Game pieces (X and O's) | - Game pieces | - Move sensors, that sense that the opponent has made his move. |

Here we describe the environment of the game:

- **Fully observable**, the agent sees the full state of the game at any given time.
- **Deterministic**, there is no randomness involved in the game itself.
- **Sequential**, the current state depends on past states like which cells we have won on the big board.
- **Static**, the state of the game does not change when the agent is deliberating
- **Discrete**, there are finite number of states of the game.

The state space of the game is much bigger than in the original game tic tac toe. The state space of the original game with unreachable states is $3^9$, since for every cell in the 3x3 board it can be empty, have an X or an O. For the variant of the game we have $3^{81}$, since for every cell of the 3x3 board we have the state space $3^9$. Since there are 9 cells we get the state space $(3^9)^9$. But then again this massive number contains a lot of unreachable states.

## Goals

For this task we have several goals. Firstly we will want to see the agent win a random agent, since the random agent in a way should be an easy opponent. We will measure this in the win ratio of 10 games against the random agent. Since we do not know how two perfectly rational agents will perform against each other in this variant of tic tac toe we considered this to be good first step.

We want to implement an agent that is a worthy opponent. What this means is that he will have to be able to draw or win an opponent as good as him. This will measured in a win ratio out of 3 games against himself.

## Work plan

Here below is the task list for the project. Of course like with all projects this task list is subject to change. In the case that we come to the realization that some task is not suitable for the scope of project, maybe due to lack of time or time spent in other tasks exceeded our estimation.

| **Task** | **Time estimate, per member** | **Members assigned** |
|---|---|---|
| Implementing the agent | 10 hours | Alex, Jóhann, Pedro |
| Setting up the environment, in the form of a web application | 8 hours | Alex (Backend), Jóhann (Frontend) |
| Setting up and running the experiments | 3 hours | Alex, Jóhann, Pedro |
| Reporting on the agent | 4 hours | Alex, Jóhann Pedro |

The first two tasks can be done in parallel, the first thing is to design the environment and how we will represent the state of the game so that the web application can be implemented in parallel. The last two shall be done sequentially. Each task is assigned to a member but with respect to change, if the task fits someone else in group better.

## References or Literature

For the implementation itself we will not be using any libraries, we will implement the Min-Max algorithm with Alpha-Beta pruning using Python[1], 3.7. The solution will be implemented with Iterative Deepening. For this work we will be using the book of the course as a reference. The web interface that will accommodate the project will have a front end written in Javascript using the React Framework[2]. The project will include a list of dependencies that will follow in a *package.json* file, we do not know in an advance what dependencies will follow. The backend of the project will be written in Python using the framework Flask[3].

[1] https://www.python.org/
[2] https://reactjs.org/
[3] http://flask.pocoo.org/