UNIVERSITY OF THE WITWATERSRAND, JOHANNESBURG
**School of Computer Science & Applied Mathematics**

# COMS4033A/7044A Assignment
# Reconnaissance Blind Chess Overview

## 1 Introduction

The most famous AI challenge of the 20th century was to defeat humans in the game of chess. Beginning with Claude Shannon's paper in 1950,[1] this goal was finally achieved in 1997, when IBM's *Deep Blue* defeated world champion Garry Kasparov. More recently, AI researchers have been thinking about how to make board games relevant in a modern context. To this end, researchers in 2019 proposed a variant of chess known as Reconnaissance Blind Chess (RBC).

RBC adds the aspect of imperfect information to the game of chess—whereas standard chess agents are capable of seeing all pieces on the board, in RBC, the player is unable to see their opponents' pieces and must keep track of where they could possibly be. Additionally, each time a player moves, they are allowed a "sensing" move, where they can observe a small area of the board to confirm what pieces are located there.

In this assignment, you will be tasked with building an agent to play the game against baseline agents. At the end, you will be required to compile a report detailing the methods you used in creating your agent. A portion of the mark will also be dedicated to the final performance of the agent. You may complete this assignment individually, or in groups of two.

## 2 Reconnaissance Blind Chess

RBC was developed by researchers at John Hopkins University in 2019. The main idea is that the game follows the same rules of chess, but prevents players from seeing where the opponents' pieces are located on the board. The main webpage and online resource for this assignment is `https://rbc.jhuapl.edu/`.

## 3 Rules

The rules of RBC are almost identical to that of chess. Those who are unfamiliar with the rules of chess should briefly review them (`https://en.wikipedia.org/wiki/Rules_of_chess`) although, as we shall see, it is not strictly necessary to be good at chess to tackle this assignment. In the rest of the document, it will be assumed that the reader is familiar with the basic rules and terminology

[1] `https://bit.ly/3tThS31`

The rules of RBC can be summarised here: `https://rbc.jhuapl.edu/gameRules`. Note that the objective of RBC, unlike chess, is to capture the opponent's king. You are encouraged to get a feel for the game by challenging a bot here: `https://rbc.jhuapl.edu/challenge`. Be sure to select the "Challenge a bot" game mode before starting.

To briefly summarise the rules, the game starts in the standard starting position, which means both players have perfect knowledge of all the pieces' locations on the board. Players take turns making moves, which consist of two phases. In phase one, a player selects a square on the board to sense. The game reveals a $3 \times 3$ window around this square, which shows any pieces located at those squares. This allows a player to gain information about that area of the board (although there may be uncertainty about other regions of the board). Then, the player makes a standard chess move (or passes). If the chess move is illegal (e.g. trying to capture a non-existent piece or moving a piece onto a square already occupied by one's own piece), then the player is informed of this and they pass their turn.

# 4 Requirements

In this assignment, you will be required to build an agent to play RBC. You may complete the assignment in **groups of between 2–3**. However, the primary focus is not on how to play good chess moves, but rather how to overcome the partial observability in the domain. You can read about some existing attempts to play the game in this paper here: `https://proceedings.mlr.press/v176/perrotta22a.html`.

In particular, you should start by reading the documentation here: `https://reconchess.readthedocs.io/en/latest/`. As a first step, install the Python package and copy the *Trout* bot example. Then, follow the instructions here (`https://reconchess.readthedocs.io/en/latest/bot_play.html`) to play a game between your *Trout* bot and a random agent.

The *Trout* bot will serve as the base from which to build your final agent. Note that, like many other submissions to the RBC tournament, the bot makes use of *Stockfish*, the strongest standard chess engine to select moves. Your bot will do the same, which means that you will primarily be dealing with how best to sense and make use of your board estimate, as opposed to playing chess.

The basic requirements are to create an agent that plays the game as follows:

- Your agent should keep track of all possible board configurations that the game could be in, given the uncertainty about the opponent's pieces.
- When making a sensing move, it is done by selecting a square uniformly at random.
- The sensing information should be used to whittle down the list of possible boards (much as one would rule out potential models in a propositional knowledge base). Other information can also be used to rule out potential states.
- To play a move, run *Stockfish* on all potential board states. Since there will typically be more than one possible board, the move to play should be the most popular move suggested by *Stockfish* across all the boards (with ties broken randomly).

The above strategy is similar to the *Oracle* bot described in the Perrotta el al. (2022) paper, except here sensing is done randomly.

## 5 Grading

Marks will be assigned as follows:

- Part 1:
    1. Board representation: 5%
    2. Move execution: 5%
- Part 2:
    1. Next move prediction: 5%
    2. Next state prediction: 5%
    3. Next state prediction with captures: 5%
    4. Next state prediction with sensing: 10%
- Part 3:
    1. Move generation: 5%
    2. Multiple move generation: 5%
- Part 4:
    1. Baseline implementation: 20%
    2. Report detailing improvements and round-robin tournament: 25%
    3. Competitive portion: 10%