Microchess – Version 1.0

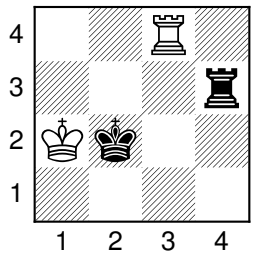
CO 456 Project, Fall 2017

November 8, 2017

1 Game Rules

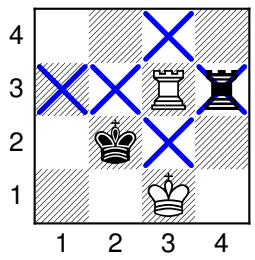
We introduce a chess-like game "Microchess". The game is played on a 4×4 grid by two players: black and white. In the beginning of the game, each player has two pieces: a king and a rook. The four pieces start in distinct cells of the grid. These positions are not fixed, that is, they can be different on each game.

The columns of the board are numbered 1 to 4 (from left to right) and the rows are numbered 1, 2, 3, 4 (from bottom to top). The figure below shows a possible starting configuration for the game.

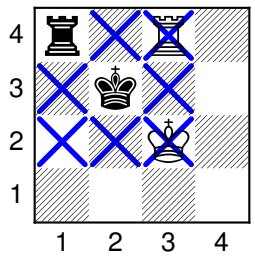


Note that a cell of the board can be represented by a pair (c, r), where $c, r \in \{1, 2, 3, 4\}$ represent the column and row respectively. For example, in the board above, the white king is at cell $(1, 2)$, and the black rook is at cell $(4, 3)$.

A rook can move horizontally or vertically, through any (nonzero) number of empty cells. If a rook can reach a piece of the opposite player in this manner, then it can *capture* that piece by occupying its cell. For example, in the following board, the white rook can move to any of the 5 cells marked with a blue cross. If it moves to cell $(4, 3)$, then the captured black rook will be removed from the board.



A king is allowed to move to an adjacent cell in any direction (horizontally, vertically, or diagonally), unless that cell is occupied by the same color rook. If an opposite color piece was there, it is captured. For example, in the following board, the black king can move to any of the seven cells marked with a blue cross. Note that if it moves to cells $(3, 2)$ or $(3, 4)$, then the captured white piece will be removed from the board.



The players alternately move one of their pieces, with the white player making the first move. When a player captures the king of the adversary, the adversary is allowed to make one last move with its rook (if it has not been captured before this time). After this the game comes to an end. We can then represent the outcome of the game by a pair of characters $(z_{\text{white}}, z_{\text{black}})$, where $z_{\text{white}} = \text{Y}$ if the white player has captured the black king, and $z_{\text{white}} = \text{N}$ otherwise. Similarly, we define $z_{\text{black}} \in \{\text{Y}, \text{N}\}$ to indicate whether the black player has captured the white king. The payoffs $(u_{\text{white}}, u_{\text{black}})$ are indicated in the following table:

W \ B	Y	N
Y	$(2, 2)$	$(3, 0)$
N	$(0, 3)$	$(1, 1)$

We say the game is a *win for white* with outcome (Y, N); a *win for black* with outcome (N, Y); and a *tie* for outcome (Y, Y). The outcome (N, N) is a *draw*; this happens when a tournament-specific move limit has been reached.

2 Tournament Rules

There will be $2n > 0$ teams entered in a tournament. A team with name Example will be entered via a file named TeamExample.java; this will contain a Java class representing a Micro-chess player. This file must have size at most $SizeLimit = 10mb$. There may be hard-coded data included at the end of this file, but no other file input/output is allowed. A single instance of each team's player will be created with the intention of playing the whole tournament; if problems arise, it may be necessary to create a new instance to finish the tournament. Upon creation of a player, the class constructor can initialize itself with pre-computations, with a time limit of $TimeLimit = 60s$. A team's player is limited to $MemoryLimit = 100mb$ of memory in total. Before each tournament, the values $SizeLimit$, $TimeLimit$ and $MemoryLimit$ will be specified, as well as penalties for exceeding these.

Once all teams are entered, the players will be paired off. Each player's $prepareForSeries$ method will be called, for clean-up and initialization. Then for each pair of players the following *series* of approximately $SeriesLength = 20$ *mirror matches* will be played, with draws being declared after $GameLength = 30$ total moves in a game:

- A random Microchess board with all four pieces is selected, and a random player is assigned to white.
- Each player's $prepareForMatch$ method is called.
- The game is played out using each player's $chooseMove$ method, and payoffs are awarded.
- The same board is selected, except the other player is assigned to white.
- Each player's $prepareForMatch$ method is called.
- The game is played out using each player's $chooseMove$ method, and payoffs are awarded.

After all pairs have finished their series, the players will be paired off with new opponents, and a new series will be played. This will be continued until a round robin is completed between all of the players.

The values $SeriesLength$ and $GameLength$ will be specified prior to every tournament. As initial guidelines, the functions $prepareForSeries$, $prepareForMatch$, $chooseMove$ should typically run in under $1s, 1s, 0.1s$ respectively. If barriers to practicality arise, we might modify tournament parameters up until 72 hours before the tournament begins.

3 Project Specifications

Your team of 1 or 2 students is tasked with coding a Microchess tournament player. On Learn you will find code for a Microchess tournament simulation, as well as templates and sample players to get you started. The *Alliance* of student teams will be joined by an equinumerous *Horde* of automated players, chosen by the moderating team of TAs and the instructor. Initially, the Horde will consist of some number of players who play "best" moves randomly, for some twisted notion of "best".

Instructions to get started:

- Choose a teammate, or ask to be paired, or choose to work alone. Your team may split up the work however you choose.
- Choose a team name; we will pretend this is "Example" for the following steps. It must be 4-12 letters, with the first letter capitalized.
- Download the latest version of the file MicroChess.zip from LEARN. (Beta version currently!)
- Change the name of MyPlayer.java to TeamExample.java
- In the files TeamExample.java and Tournament.java, replace "MyPlayer" with "TeamExample" (in the class name, class constructor and for the tournament entry).
- In TeamExample.java, modify the methods TeamExample, prepareForSeries, prepareForMatch, chooseMove to whatever you want!
- You can copy the code from TeamMonkey.java or TeamNihilist.java to get started with a silly player.
- More useful sample players will be provided in the near future.

- In the src directory, run "javac *.java", and then "java Tournament full" to test the tournament.
- Only the TeamExample.java file will be your project submission for tournaments.
- E-mail problems to g3gauthi@uwaterloo.ca and alinhare@uwaterloo.ca!

Grading:

- This project is worth 15 marks of your final grade. Please talk to the instructor as soon as possible if you would like to opt out, and propose an alternate project instead.
- Milestones: $7 = 1 + 3 + 3$ marks will be for completing milestones. This includes participation in two mandatory practice tournaments, and answering a few questions to get you thinking.
- Final tournament result: this is worth 8 marks. Suppose a student ranked s out of $2n$ in total payoff, with 1 being the lowest, and $2n$ being the highest. Let S be the sum of all student rankings s, and let σ denote the minimum possible value of S, and Σ denote the maximum possible value. Then that student's grade will be:

$$4 \cdot \frac{s-1}{2n-1} + 4 \cdot \frac{S-\sigma}{\Sigma-\sigma}$$

- Idea: we want you to learn as much as possible, and to help one another to achieve this. Hence the better the Alliance does against the Horde, the better the average grade. That explains the second term. The first term simply indicates that your grade depends on how well you do versus all other players, including the Horde. (Your team can opt out of the Alliance versus Horde battle, and instead the final tournament portion of your grade will be $8 \cdot \frac{s-1}{2n-1}$; this is unlikely to significantly help you, and there is no going back. Glory to the Alliance!)

Mandatory deadlines in bold:

- Wed, Nov 8: Project Specifications Version 1.0, Beta Code released
- Mon, Nov 13: **Zeroth milestone due.** Zeroth practice tournament (optional).
- Mon, Nov 20: **First milestone due. First practice tournament**
- Mon, Nov 27: **Second milestone due. Second practice tournament**
- Fri, Dec 1: Final practice tournament (optional)
- Mon, Dec 4: **Showdown: Final Tournament**

4 The Initial Horde

We define 8 initial player personalities which will likely show up during the first practice tournament, among others. None of these g have memory, and they play each game as if it was the only one. The moderating team's goal will be to challenge the students, but not to best them. There will always be some weak players in the Horde.

The first two personalities are somewhat silly. A Monkey (M) is a player who takes the best piece they can, or else plays randomly. A Nihilist (N) is a player who plays randomly, since every choice is optimal when nothing matters. (In the following, the quotes carry no meaning, they're just there for fun.)

- Monkey: captures the king if possible; else captures rook if possible; else plays randomly.

  "In fair Verona, wheaor45j93mico,49" – Monkey

- Nihilist: moves are chosen randomly.

  "I can't go on, I'll go on." – Samuel Beckett

All of the other personalities are rational, and assume that their opponents are rational... But they have varying goals, and make strong (and mostly incorrect) assumptions about their opponent's goals. Let u_1, u_2 denote the utilities derived at the end of a game of Microchess for a player 1 and their opponent 2.

The Realist (R) plays to maximize their payoff $u_{1,R} = u_1$, and assumes that their opponent will do the same for their payoff $u_{2,R} = u_2$. That is, a Realist plays the game as it was designed! This is the type of player we normally consider in chess-like games. So a Realist always chooses a winning move if possible, or else tries to tie the game, or else tries to draw; and assumes the opponent has similar motivations. Any player satisfying this condition will be considered a Realist.

On the other hand, an Optimist (O) thinks everyone is out to help them; so they plays best moves assuming that $u_{1,O} = u_1$ and $u_{2,O} = u_1$ as well! Note that the Nihilist (N) is in fact a rational player with $u_{1,N} = 0 = u_{2,N}$. The other personalities are similarly listed below:

- Optimist: $u_{1,O} = u_1$ and $u_{2,O} = u_1$

  "I feel like I'm too busy writing history to read it." – Kanye West

- Pessimist: $u_{1,P} = u_1$ and $u_{2,P} = -u_1$

  "If you expect the worst, you'll never be disappointed." – Sarah Dessen

- Queller: $u_{1,Q} = -u_2$ and $u_{2,Q} = u_2$

  "A ain't got time to bleed." – Blain from Predator

- Realist: $u_{1,R} = u_1$ and $u_{2,R} = u_2$

  "Our true passions are selfish." – Stendahl

- Scrapper: $u_{1,S} = -u_2$ and $u_{2,S} = -u_1$

  "When somebody challenges you, fight back. Be brutal, be tough." – Donald Trump

- Truster: $u_{1,T} = 3u_1 + 2u_2$ and $u_{2,T} = 2u_1 + 3u_2$

  "The best way to find out if you can trust somebody is to trust them." – Ernest Hemingway

- Utilitarian: $u_{1,U} = u_1 + u_2$ and $u_{2,U} = u_1 + u_2$

  "The end may justify the means" – Leon Trotsky

It will often be the case that Horde players play randomly when faced with choices giving equivalent payoffs; and they assume that their opponents will do the same.

5 Zeroth Milestone, due Monday November 13

E-mail your answers to both g3gauthi@uwaterloo.ca and André Linhares (alinhare@uwaterloo.ca) by 9:30AM, Monday November 13. You may also optionally submit a code fragment to the designated dropbox folder on Learn, to participate in the zeroth tournament, with parameters: $SizeLimit = 10mb$, $TimeLimit = 120s$, $MemoryLimit = 100mb$, $SeriesLength = 20$, $GameLength = 30$.

1. Choose one of the following**, as well as a team name of 4-12 letters, capitalized**:

   - Please pair me up with another student. My comfort level with coding/Java is /10.
   - I have chosen a teammate, with the following name and student number. (Both students need to list the other.)
   - I would like to work alone.

2. Choose zero or more of the following:

   - I would like to opt out of this project. I have discussed this with the instructor, and we have agreed on an alternate project. I realize that I cannot go back on this choice.
   - I would like to go solo, and have my final tournament result determine 8% of my final grade. I realize that I cannot go back on this choice.

3. A *board position* is any game board that can be reached in Microchess, along with an indicator $I \in \{B, W, E\}$ meaning "Black to play", "White to play" or "Game ended" respectively. Equivalently, a subset of the four pieces are placed and I selected subject to:

   - At least 1 piece remains.
   - If both kings remain, $I \neq E$.
   - If exactly one king (of color $C \in \{B, W\}$) is missing and the C rook remains, then $I \in \{E, C\}$
   - If exactly one king and the same color rook are missing, then $I = E$.
   - If both kings are missing, then $I = E$.

   How many board positions are possible? You should not use a computer to compute this, but you can verify your answer that way. [Hint: break it down into cases depending on which pieces remain.]

6 First Milestone, due Monday November 20

E-mail your answers to both g3gauthi@uwaterloo.ca and André Linhares (alinhare@uwaterloo.ca) by 9:30AM, Monday, November 20.

1. You must submit a code fragment to the designated dropbox folder on Learn, to participate in the first tournament. Your mark is based on having your code run without crashing. The parameters $SizeLimit$, $TimeLimit$, $MemoryLimit$, $SeriesLength$, $GameLength$ are to be determined, and you will be warned if your code exceeds the limits. You cannot use a sample player without modification.

2. [TBD]

3. [TBD]

7 Second Milestone, due Monday November 27

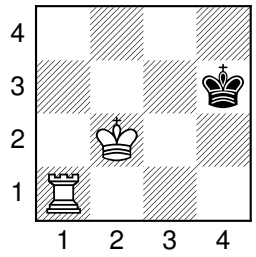
E-mail your answers to both g3gauthi@uwaterloo.ca and André Linhares (alinhare@uwaterloo.ca) by 9:30AM, Monday, November 27.

1. You must submit a code fragment to the designated dropbox folder on Learn, to participate in the first tournament. Your mark is based on having your code run without crashing, and respecting resource limits. The parameters $SizeLimit$, $TimeLimit$, $MemoryLimit$, $SeriesLength$, $GameLength$ are to be determined. You cannot use a sample player without modification.
2. [TBD]
3. [TBD]

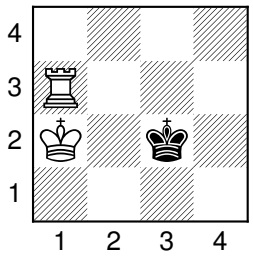
8 Food for Thought

You are encouraged to think about these questions. Some Milestone questions will be similar.

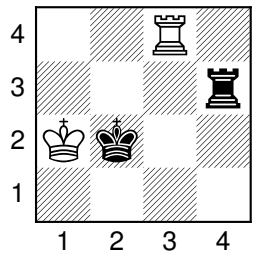
1. Player Personalities!
   (a) Can a player be both a Realist and a Utilitarian?
   (b) Are any of the listed personalities identical?
   (c) If you believe that you are playing against a certain personality, how can you use this to your advantage?
   (d) How can you take advantage of an opponent's belief about your player's personality?
2. [TENTATIVE] Consider the following position:



   (a) Show that white can win regardless of who starts, and regardless of black's moves.
   (b) Show on any board with the same three pieces, if white starts and their king is on $(2, 2)$, then white can win regardless of black's moves.
   (c) Show that white can force a win here as well, regardless of who starts:



3. [TENTATIVE] Suppose you play as white for a single game in the following starting position, and the black player (P) is a Pessimist. That is, (P) tries to rationally maximize (P)'s payoff, and assumes that you will rationally play to minimize (P)'s payoff (with no regard for your own payoff).



   (a) What move should you play to maximize your payoff?

   (b) Aside from that move, find the next best move for a Realist. How will (P) respond?

9 Cooperation and Academic Rules

The whole class will benefit from a stronger overall standing. Toward this goal, we encourage the following in the lead-up to the tournament:

- Having inclusive meetings between teams to discuss the project.
- Online discussion on Piazza.
- Cooperation in overcoming conceptual or technical difficulties.
- Sharing knowledge and expertise.
- Using and modifying the code from the sample players we provide.
- Finding inspiration from publicly available code, **with attribution**.

During the tournament, you are all expected to be honestly competing for a higher placement. The following are not allowed:

- Coordination between different teams. For example:
  - Secret handshakes or similar methods to identify a team during the tournament.
  - Forming cliques and purposefully excluding other teams.
- Testing the limits of the rules. For example:
  - Willfully exceeding resource limits.
  - "Black box" inspiration, where you use ideas that you do not understand.
  - Nearly copying an outside source, with or without attribution.

If we suspect infringement, we will ask you to rectify the issue. Our secondary tool to combat these issues will be to tweak the rules. Notably, we might modify the Horde to move things in an intended direction. Rest assured that we will not make accusations lightly, and we aim to err on the side of caution.

In extreme cases, we may seek disciplinary action for academic offenses. For example, this might arise if the aforementioned rules are broken repeatedly and willfully. To be sure, the following must be avoided entirely:

- Plagiarism: you cannot copy code from other teams, or from outside sources.
- Lack of attribution: very nearly copying an outside source, without attribution.
- Exploitative code: purposefully attempting to read/write to memory or elsewhere when no access was intended.
- Malware: abusing our trust by including malicious code.

Of course, all academic regulations must be respected, as is always the case.