Capture the King Chess and other Variants

Bachelor Thesis Proposal

Andreas Abel

October 2022

Potential supervisors: Andreas Abel, Christian Sattler

Background

Chess has gained surprising popularity in recent times, possibly due to the availability of online chess servers, strong chess engines, open availability of training materials, online streamers and commentators, and maybe because of its connection to the quest for artificial intelligence.

There are numerous online chess forums that allow you to play and analyse chess games, even for some popular chess variants. However, many chess variants have not seen the same availability and exploration.

An often discussed chess variant is **Capture the King**. In this variant (family), the goal is to capture the opponents king rather than checkmate it. This sounds like a minor reformulation only (because a checkmated king would be captured in the next move), but has deep implications. For instance, there is no significant concept of *checking* the opponents king as the opponent could choose to ignore the check and lose their king (and game) in the next move. Also, one could overlook that the opponent's king is checked and make a move besides capturing the king; thus, the opponent could escape a checkmate situation. Further, *stalemate* is abolished; a stalemated king has to move into check where it can be captured (so a stalemate is a loss except for the opponent overlooking their victory). Endgames change drastically; some drawn endgames turn into wins, some into losses even for the stronger party. Further, the castling rules are simplified: there is no more reason why one should not be allowed to castle out of check, into check, or over threatened squares.

Capture the King deprives the king of its aristocratic privileges, which spared it from being actually captured. One could go further and even take the right to castle, arriving at *castle-free chess*, which also has been suggested.

There are tons of other chess variants. E.g., *denial chess*, where you can force your opponent to take back their last move and play a different one, once per move. Or one could restrict this to say only 3 such vetoes per game. An comprehensive collection of chess variants is David Pritchard's *Classified Encyclopedia of Chess Variants*, online at https://www.jsbeasley.co.uk/encyc.htm .

In general, looking at chess variants provides an opportunity to freshly analyse the game, without having to compete with existing databases of openings, endgame positions, and game collections. Implementing chess variant servers with game playing functionality and engines provides a way to evaluate these variants more thoroughly than just the manual study.

Project description

The goal is to design and implement a Chess platform for variants. This goal is very open-ended, so a first task is to define the scope of the project. Here are parts of the design space:

Scope of chess variants

While classic chess has fixed rules that can be hardwired into the implementation of a chess platform, allowing variants requires a flexible setup. A first task is to define the limits of considered chess variants: For instance a limit could be that only 8x8 boards are considered, but one could allow rectangular boards in general, or even other board topologies. Then, a limit could be that only the classic chess pieces (king, queen bishop, knight, rook, pawn) are considered, but one could also choose to let other pieces be definable. Then, one could limit the possible moves of pieces to classic chess (straight, diagonal, knight move, not passing over pieces etc.) or make other moves definable.

In the end, a data structure, library or domain-specific language should be defined that allows the definition of a chess variant.

Style of chess platform

A chess platform can take the shape of a server implementing a protocol to negotiate and start a game, to make moves and to declare the end of a game, all the while keeping time control. Clients connect to this server; both human players and computer players could connect. A client for the human player would allow input via a GUI (or even just console) and check for legal moves. Example: https://www.freechess.org/ with clients listed on https://www.freechess.org/interfaces.html .

Another type of platform would only allow human players to play on a webpage against other human players or chess engines. For instance: https://lichess.org/ .

If the focus on the project is less on the platform, a standalone program is thinkable. Here, either a human plays against the computer or another human on the same program, or two computer play against each other.

Chess engine

Chess engines have been developed for decades with ever more sophistication, from the basic alpha-beta-pruning with a manually programmed position evaluation function to Monte Carlo tree search and the self-trained AlphaZero.

Making stronger and stronger chess engines is an open-ended project; and not everything is feasible in the scope of this project with the available computing resources. However, engine development is fun, and *some* engine developed in course of the project will help evaluating chess variants.

Analysis tools

Chess engines are usually enhanced by opening and endgame tables. An opening table is likely hard to come by, since it is distilled from large game databases. However, endgames with only a few pieces left can be decided with reasonable computing power. Contemporary databases contain the results for all boards with only seven pieces in total left.

A possible component of this project is to analyse endgames for a chess variant with only a few pieces left on the board, and draw conclusions about this chess variant.

Suggested reading material

- Browse https://www.jsbeasley.co.uk/encyc.htm or other resources for chess variants
- https://en.wikipedia.org/wiki/Alpha%E2%80%93beta_pruning and other resources on game strategies

Target group

Bachelor students in DV, D, IT.

Special prerequisites

- Knowledge of chess is strongly recommended, passion for chess is certainly a plus!
- Good programming skills
- Interest in algorithms