Haskell Chess Engine

Name: Michael Lee

UNI: mhl2156

Given a position in chess, what does my program think is the next best move?

For this project, I would like to create a program that is able to evaluate a position in chess, and return what the next best move is.

It would take an input in of a chess board in the representation of a list of tuples, where each tuple included within the list would be a tuple of a string (representative of a chess piece, specific to a certain color) and chess coordinate position. It would also accept a parameter for which color the user would like to have evaluated.

Using movement rules hardcoded into the program, the chess bot would then create a list of every square in the 64-square board, and count on each one how many times it is attacked, as well as whether or not it is occupied with a piece. It would most likely achieve this by traversing the input "board", and then using the movement rules to jump to specific coordinates and mark them.

From this information, the program would then generate a list of potential moves for some pieces on the board. Parameters such as point differential (am I capturing something or am I putting my piece in danger?), distance from center of board, legality of move, distance from enemy king, etc are all viable options. Can expand further on these parameters as the realistic power of the chess evaluator becomes clearer.

After evaluating, the chess bot will then be able to differentiate a top move, and then will suggest this move to the user in the form of a tuple, (String, (String, String)) with the piece name and the coordinate it is meant to move to.

The evaluation itself will be the main way that parallelism is implemented, as performing many different computations will fit into the idea of parallel evaluation. The scope of this project can be scaled based on how many parameters make sense for move evaluation; Seeing as there are infinite possibilities for chess and no single objective way to play the game, design will be open to customizing. Along with this, there are many different moves that are not necessarily obvious, such as "en-passant" capturing and castling. Along with this, many chess bots are designed to prioritize potential checkmating, as well as have the ability to use bfs-type positional evaluation

to suggest moves after evaluating an entire tree of moves branching from one. This is also a potential option for increasing the scale of this project.