TP FINAL ALP

Franco García Cillero

March 31, 2023

1 Proyecto

El proposito de este proyecto es facilitar el aprendizaje y la práctica del ajedrez japones o también conocido como shogi, al permitirle al usuario tanto jugar el juego como poder "reproducir" una partida previa paso por paso.

2 Gramática

Int ::= 1 | 2 | ... | 9
Char ::= "a" | "b" | ... | "Z"
String ::= Char | Char String
Pieces ::= King | Rook | Bishop | Golden | Silver | Knight | Lancer | Pawn | Dragon | Horse | PromotedSilver | PromotedKnight
| PromotedLancer | PromotedPawn
Player ::= White | Black
TakenPieces ::= Pieces | Pieces TakenPieces
Position ::= (Int,Int)
Board ::= (Player,Position,Pieces) | (Player,Position,Pieces) Board
Game ::= Board | Board Game
Actions = MoveP Position Position | DropP Pieces Position | PromP Position | ShowB | ShowP | SaveG FileName | NewG | LoadG FileName | PlayRec FileName

3 Manual de uso

Este programa tiene dos etapas, primero se define si se quiere comenzar una partida nueva con el tablero inicial escribiendo "New" o "n", cargar una partida previa guardada en la carpeta Games escribiendo "$Load$" o "l", o reproducir una partida previa paso a paso escribiendo "$Record$" o "r". Si se opta por esta ultima opción se procede a imprimir el tablero de cada turno guardado y se pasa apretando enter hasta el turno final. En los otros casos se carga el tablero base o el del último turno y se pasa a jugar. Los comando principales para afectar el tablero son "$Move$" o "m" que toma dos posiciones para mover una pieza de una posición a otra y "$Drop$" o "d" que pone una pieza capturada del oponente en una posición dada. "$Move$" tiene un efecto agregado de si se mueve una pieza a un área especifica esta pieza puede promoverse y cambiar sus

movimientos posibles. Las otras funciones son para asistir a los jugares, ende son ajenas al juego propiamente dicha. "*ShowBoard*" o "*sb*" muestra el estado del tablero actual, "*ShowTakenPieces*" o "*sp*" muestra las piezas capturadas por el jugador actual y "*Save*" o "*sg*" guarda el estado actual de la partida con un nombre dado, también se llama automaticamente cuando un jugador gana la partida. En cualquier momento se puede escribir "*exit*" si se desea salir de la partida actual sin guardar nada.

4 Instrucciones de ejecución

Se utiliza directamente stack ende no se necesita nada más que este para hacer correr todo. Con hacer stack build se descarga todas las librerias necesarias o directamente stack run que hace el build y procede a correr el código.
Si no se tiene stack instalado se puede descargar con
wget -qO- https://get.haskellstack.org/ | sh o
curl -sSL https://get.haskellstack.org/ | sh

5 Desiciones de implementación

El estado global de GameState contiene varios campos booleanos, playerInCheck, wonGame y moveCanPromote que se usan principalmente para simplificar las interacciones entre distintas partes del código, principalmente entre main y eval y también evitar llamar funciones muy costosas como checkCheckmate para wonGame pero podría llegar a hacerse una implementación sin estos.
Se hicieron dos evaluadores a pesar de haber un solo tipo en la gramática porque se quiso mantener completamente separadas las dos etapas del juego y no depender de variable globales extra para ver si ya se inicio una partida.

6 Referencias e inspiraciones

Como este trabajo práctico se hizo a la par con el compilador de la materia de compiladores se termino tomando la estructura general y muchas de las librerias usadas allí como me resultaban más familiares y correctas.