Proyecto del curso de Programación Funcional 2022

Martian Chess

El proyecto de curso de Programación Funcional en 2022 será implementar en Haskell una versión del juego *Martian Chess*. Se trata de un juego de tablero de estrategia abstracta para dos jugadores, creado por Andrew Looney en 1997, y publicado por Looney Labs.

Figure 1: Set de *Martian Chess*

Reglas del juego

Los juego se juega en un tablero rectangular de 4x8 casillas, dividido en dos zonas de 4x4 casillas. Las 18 piezas no tienen color. Cada jugador controla una zona del tablero, y todas las piezas que se encuentren en ésta. El tablero inicia con 9 piezas de cada lado: 3 *peones*, 3 *drones* y 3 *reinas* (en orden creciente de tamaño).

Los jugadores toman turnos moviendo una pieza a la vez. Las *reinas* pueden mover cualquier cantidad de casillas en cualquier dirección sin saltar (como en el Ajedrez estándar). Los *drones* mueven una o dos casillas horizontal o verticalmente. Los *peones* pueden mover un espacio en diagonal.

Para capturar una pieza, la pieza atacante siempre tiene que moverse de una zona del tablero a la otra (a través del *"canal"*). El jugador que captura una pieza gana puntos: 1 por peón, 2 por dron y 3 por reina. El juego termina cuando una zona del tablero queda vacía (i.e. sin piezas). Gana el jugador que tiene más puntos. Si ambos jugadores tienen la mismo puntaje, gana el que dejó su zona vacía.

Planteo

Primero se debe definir una estructura de datos para almacenar la información del estado del juego en cualquier momento de la partida. Dicha estructura de datos se implementará como un tipo `data` de Haskell llamado `MartianChessGame`.

También se deben definir dos tipos `data` de Haskell más. Uno llamado `MartianChessAction` para representar los posibles movimientos de los jugadores. Los jugadores se definen de la siguiente forma:

```haskell
data MartianChessPlayer = DeimosPlayer | PhobosPlayer deriving (Eq, Show, Enum)
```

Figure 2: Partida de Martian Chess

Los posibles resultados de la partida se representan con el siguiente tipo:

```
data GameResult p = Winner p | Loser p | Draw deriving (Eq, Show)
```

Las funciones a implementar son las siguientes:

- `beginning :: MartianChessGame`: El estado inicial del juego de *Martian Chess*.
- `activePlayer :: MartianChessGame -> Maybe MartianChessPlayer`: Esta función determina a cuál jugador le toca mover, dado un estado de juego. Si ninguno de los jugadores puede mover, e.g. porque el juego está terminado, se retorna `Nothing`.
- `actions :: MartianChessGame -> [(MartianChessPlayer, [MartianChessAction])]`: La lista debe incluir una y solo una tupla para cada jugador. Si el jugador está activo, la lista asociada debe incluir todos sus posibles movimientos para el estado de juego dado. Sino la lista debe estar vacía.
- `next :: MartianChessGame -> MartianChessPlayer -> MartianChessAction -> MartianChessGame`: Esta función aplica una acción sobre un estado de juego dado, y retorna el estado resultante. Se debe levantar un error si el jugador dado no es el jugador activo, si el juego está terminado, o si la acción no es realizable.
- `result :: MartianChessGame -> [GameResult MartianChessPlayer]`: Si el juego está terminado retorna el resultado de juego para cada jugador. Si el juego no está terminado, se debe retornar una lista vacía.
- `showGame :: MartianChessGame -> String`: Convierte el estado de juego a un texto que puede ser impreso en la consola para mostrar el tablero y demás información de la partida.
- `showAction :: MartianChessAction -> String`: Convierte una acción a un texto que puede ser impreso en la consola para mostrarla.
- `readAction :: String -> MartianChessAction`: Obtiene una acción a partir de un texto que puede haber sido introducido por el usuario en la consola.

La cátedra entregará código de referencia para facilitar las pruebas del código solicitado. Éste contiene: definiciones de algunos tipos, esqueletos de funciones a implementar, y una función `main` para poder probar la implementación.

El trabajo debe realizarse en equipo. Se entregará vía Webasignatura hasta el 8 de julio a las 18:00 horas. Inmediatamente luego de la entrega se tomará una defensa a todos los miembros de cada equipo.

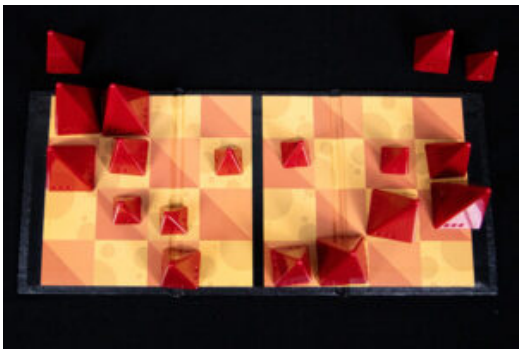

Figure 3: Partida de Martian Chess 2

Extras

Las siguientes tareas extra pueden ser realizadas por los equipos para agregar puntos a su calificación, por encima de los 100 puntos que vale el proyecto sin éstos.

Variantes de juego.

Se pretende habilitar variantes al juego original, agregando las siguientes definiciones para modificar al juego original.

```haskell
data MartianChessConfig = MartianChessConfig {
  -- Valores de variantes
  } deriving (Eq, Show)

beginning :: MartianChessConfig -> MartianChessGame
```

Al tipo `MartianChessConfig` se le podrán ir agregando datos para soportar las modificaciones a implementar. Esto implicará cambios en `beginning` y el resto de las funciones solicitadas. Las posibles modificaciones son:

- *Variante de puntaje por pieza* (5 puntos). El juego original indica un puntaje fijo que cada pieza. Agregar a `MartianChessConfig` el valor `points :: [Int]`, para indicar el puntaje de cada pieza. Para el juego original sería `[1, 2, 3]`.
- *Variante de límite de movimientos* (5 puntos). El juego original no tiene una cantidad de movimientos máxima. Agregar a `MartianChessConfig` el valor `maxRounds :: Int`, para forzar a la partida a terminar después de una cierta cantidad de rondas (i.e. un movimiento para cada jugador).

Jugador Inteligente.

La plantilla de código entregada contiene una mínima implementación de juego. Se define un tipo para agentes jugadores, y se implementan dos: el *jugador aleatorio* (que elige sus movimientos al azar) y el *jugador de consola* (que toma sus movimientos de la entrada estándar).

Una tarea extra de hasta 15 puntos consiste en implementar un jugador inteligente. Se considerará correcto si el jugador le gana significativamente al jugador aleatorio.

Referencias

- *Martian Chess* @ Looney Labs:
  - `https://www.looneylabs.com/games/martian-chess`
- Manual de *Martian Chess*:
  - `https://www.looneylabs.com/sites/default/files/literature/MartianChess_Rules5.pdf`
- *Martian Chess* @ Wikipedia:
  - `https://en.wikipedia.org/wiki/Martian_Chess`.

- *Martian Chess* @ Board Game Geek:
  - `https://boardgamegeek.com/boardgame/19803/martian-chess`.

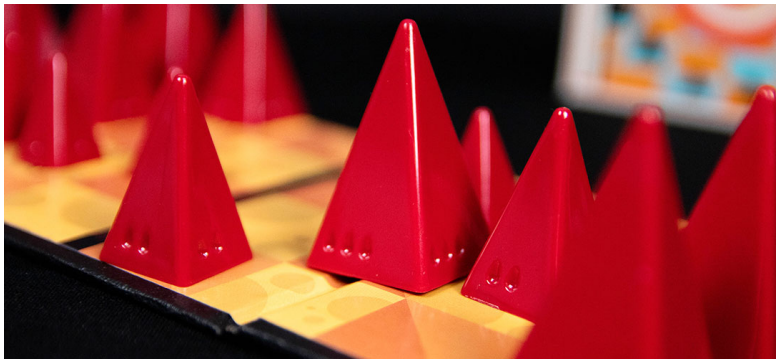

Figure 4: Partida de Martian Chess 3
