Tarea Programada Número 2

José Castro, Inteligencia Artificial
2^{do} Semestre

13 de octubre de 2012

1. Introducción

Los juegos de dos contrincantes han sido una de las tareas principales en el desarrollo inicial de la IA. En esta tarea ud. tendrá que programar un agente para jugar othello implementando el algoritmo de minimax o alguna de sus variantes.

2. Objetivos

Familiarizar al estudiante con la programación de minimax para su uso en juegos de 2 adversarios.

3. Especificación

3.1. El Juego de Othello

El Othello es un juego para dos contrincantes, A y B, donde el objetivo de cada jugador es acomodar la mayor cantidad de fichas de su color en el tablero.

El tablero de Othello es una cuadrícula de 8×8 en la cual inicialmente se colocan cuatro fichas, dos negras y dos blancas, en el centro del tablero, alternando las fichas blancas con la negras, de tal manera que las fichas negras conforman una diagonal, y las fichas blancas también.

Los adversarios se turnan para colocar una ficha en el tablero, esta ficha solo puede ir en puntos específicos del tablero para. Para ver las reglas exactas del Othello pueden consultar la págita `www.britishothello.org.uk/rules.html`, o bien jugar con el othello que se encuentra incluido como parte de los documentos de la tarea.

3.2. Los programas incluidos

En la página del curso encontrará varios archivos que debe utilizar para esta tarea, estos son:

- `server.erl` archivo principal del servidor de Othello. El servidor de Othello implementa las siguientes funciones en su interfaz:
  - `server:start()` inicial el servidor, e instala el proceso servidor en la memoria. Esto registra un proceso con el nombre de `oserver` (*Othello Server*).
  - `server:stop()` para el proceso servidor.
  - `server:connect(Color)` registra el proceso `self()` como el proceso a cargo del color `Color` el cual puede ser `white` o `black`. Cada vez que el juego tiene un cambio de estado el servidor notifica a los procesos/jugadores participantes del nuevo estado, con tal de que aquellos a cargo de las blancas o negras tomen las acciones requeridas en su turno.
  - `server:disconnect()`, converso de la llamada anterior.
  - `get_status()` retorna una tupla con la siguiente estructura: {`ok, Status`}, donde `Status` es un registro cuyo formato se encuentra en el archivo `othello.hrl`.
  - `server:make_move(Color, Pos)` Solicita al servidor ubicar una ficha del color textttColor (el cual puede ser `white` o `black`.
- `client.beam` archivo compilado con el ejemplo de como debe correr su tarea. Este archivo es un códio ya compilado que hace lo que su programa debe hacer. Salvo el requerimiento adicional de hacer deducción de la próxima movida.

- `othello.hrl` estructura de datos (registro) que contiene la información del juego. Aquí se encuentra el registro que contiene la información del juego:

```
-record(game,
{id      = othello, % constante
 black   = none,    % proceso que juega con negras (ie: <0.33.0>)
 white   = none,    % proceso que juega con blancas
 current = black,   % identificador de turno (white|black)
 pass    = 0,       % times in a row players have passed (< 2).
 seconds = 5,       % segundos que dura el turno
 timer   = none,    % identidad del proceso timer
 board   = none,    % tupla con el tablero actual
 border  = none     % lista con indices de posicion del borde
}).
```

- `othello.erl` archivo que contiene funciones utiliarias para poder programar el código del juego.

3.3. Su tarea

Ud. debe hacer un agente de Othello que se conecte al servidor y asuma el papel de jugar con blancas o negras Estos programas estarán sujetos a un límite de tiempo para efectuar su movida.

4. Calendario y Entregables

La fecha de entrega es el viernes 19 de Octubre, debe presentar un documento corto detallando que variante de MiniMax utilzó y como la implemento en erlang.