OTHELLO

Inteligencia Artificial

CLIPS

Douglas & Erik

1. Desarrollo del juego

1.1. Archivos

- loader: archivo que carga el resto de archivos
- tablero: funciones para interactuar con el tablero
- imprimir: funciones para facilitar la impresión
- cpu: funciones para la lógica del rival por CPU
- reglas: reglas para el control del juego

1.2. Templates

- Configuración: la configuración para la partida; con tamaño, profundidad y modos
- Juego: la representación actual de la partida; con turno, cantidad de piezas y tablero
- Casilla: la casilla siendo analizada; con posición y direcciones

1.3. Hechos

- inicial: punto de partida del programa
- cambiar-turno: cambia de turno o repite turno si es requerido
- human / cpu: indica el modo a emplear a continuación
- imprimir: imprime el tablero, el turno y las jugadas disponibles
- mover ?x ?y: indica dónde se quiere colocar una ficha
- is-victoria: comprueba si se ha terminado la partida

1.4. Reglas y funciones

(también variables globales)

Tipo	Nombre	Descripción
loader.clp		
function	loader	carga el resto de archivos
tablero.clp		
global	tamanoFila	tamaño de cada fila y columna del tablero
function	tablero4	crea un tablero 4x4 inicial
function	tablero8	crea un tablero 8x8 inicial
function	tablero	crea un tablero inicial respecto a *tamanoFila*
function	opuesto	devuelve el símbolo de jugador opuesto

function	parse-nth	convierte un punto a índice del multislot
function	parse-x-y	convierte de índice del multislot a punto
function	obtener	devuelve el símbolo en dicho punto
function	cambiar	coloca el símbolo en dicho punto
function	fuera	TRUE si el punto se encuentra fuera
function	posicion-valida	TRUE si la posición no está ocupada
function	direccion-valida	devuelve la cantidad de fichas que se revertirían en dicha dirección o 0 si no es válida
function	revertir	revierte todas las fichas acorde en la dirección
function	get-succesors	devuelve una lista de puntos sucesores posibles conformada por pares de *x* e *y*
imprimir.clp		
function	imprimir	imprime el tablero
function	imprimir-succesors	imprime los sucesores con el formato: x_1, y_1; x_2, y_2; x_3, y_3;
cpu.clp		
global	MIN	*infinito* negativo para minmax
global	MAX	*infinito* positivo para minmax
function	simular-movimiento	simula el movimiento en la rama actual del algoritmo minmax
function	controlar -esquinas	devuelve la puntuación extra acorde al dominio de esquinas
function	controlar-centro	devuelve la puntuación extra acorde al dominio del centro
function	voltear-mas	devuelve la puntuación extra corde a la ventaja de piezas
function	evaluar-tablero	devuelve la puntuación acorde junto a las estrategias
function	minmax	algoritmo minmax
function	min-value	parte min del algoritmo minmax, contra el rival
function	max-value	parte max del algoritmo minmax, a favor nuestra
function	mejor-movimiento	devuelve la mejor jugada empleando minmax

reglas.clp		
inicial		reglas iniciales
rule	inicial	solicita la configuración deseada
cambiar turno		reglas para cambio de turno
function	assert-quien	realiza assert del modo deseado (human / cpu)
function	quien	devuelve la puntuación del jugador
rule [11]	OtoX	cambia el turno de O a X
rule [11]	XtoO	cambia el turno de X a O
rule [10]	turno-no-valido	recupera el turno
configuracion		reglas para la configuración
rule	configuracion	aplica la configuración e inicia el juego
imprimir		reglas para la impresión de jugadas
rule [1]	imprimir	imprime el tablero, el turno y las jugadas posibles
rule [1]	clear-imprimir	elimina el hecho *imprimir*
mover		reglas para la colocación de fichas
rule	human	solicita por consola la siguiente jugada
rule	cpu	solicita a la lógica automática “cpu” la siguiente jugada
rule	casilla-valida	comprueba si la casilla es válida (posición y direcciones)
rule	casilla-no-valida	avisa la no validez por consola y solicita una nueva jugada
revertir		reglas para revertir las fichas
function	revertir-hacia	revierte en una dirección (evita la duplicación de código)
rule	revertir-# [8]	revierte hacia la dirección # especificada
rule	revertir	coloca la última ficha y completa la reversión
meta		reglas para definir el fin del juego
rule [20]	is-victoria	devuelve quién ha ganado y para el juego
rule [20]	not-victoria	elimina el hecho *is-victoria*

2. Representación empleada

X para el primer jugador y O para el segundo. Diferenciando entre modo “human” y “cpu”, cualquier otro modo se considerará de tipo “debug”.

El tablero se representa mediante un único multislot de símbolos, con los símbolos posibles: _, O y X, siendo _ para casilla vacía. Los puntos se representan mediante su valor en el eje horizontal (x) seguido de su valor en el eje vertical (y), es decir, primero columna, luego fila.

3. Idea del heurístico

El heurístico emplea cuatro estrategias del juego:

- Controlar esquinas: 10 puntos por cada esquina
- Controlar el centro en juego temprano: 1 punto extra por cada ficha dentro del 4x4 central y 1 punto menos por cada ficha fuera del 4x4
- Voltear menos en juego temprano: 1 punto por cada ficha de desventaja
- Voltear más en juego tardío: 1 punto por cada ficha de ventaja

4. Pasos a seguir

Seguir estos pasos en el IDE de CLIPS.

- File -> Load... -> "loader.clp"
- Escribir en la ventana de diálogo (Dialog Window):
(loader)

Para jugar:
(run)
TRUE

O en lugar de “TRUE”, podemos establecer la configuración que deseemos poniendo “FALSE” y a continuación los datos:

- **tamaño**: número de casillas por cada fila y columna [POR DEFECTO: *8*]
  Opciones: *4*, *6* u *8*
- **profundidad**: profundidad del algoritmo minmax para las CPU [POR DEFECTO: *3*]
  Opciones: cualquier natural, idealmente no mayor a 8
- **O**: modo para el segundo jugador [POR DEFECTO: *cpu*]
  Opciones: *human*, *cpu*. Cualquier otro modo se considerará *debug*
- **X**: modo para el primer jugador [POR DEFECTO: *human*]
  Opciones: *human*, *cpu*. Cualquier otro modo se considerará *debug*
- **imprimir**: si muestra el tablero, turno y posibles jugadas [POR DEFECTO: *TRUE*]
  Opciones: *TRUE*, *FALSE*