MEMORIA DEL PROYECTO - Diego Forniés y Unai Salaberria

Introducción

Este proyecto tiene como objetivo implementar el juego de mesa *Backgammon* utilizando el lenguaje de programación CLIPS. Además, crearemos un agente inteligente capaz de jugarlo.

Cómo jugar

Para poder jugar al Backgammon que hemos desarrollado junto con su respectivo agente inteligente, es necesario seguir los siguientes pasos:

1. Abrir una terminal completamente vacía de CLIPS.
2. Hacer 'load' del archivo llamado "agente_inteligente.clp".
3. Hacer 'load' del archivo "representacion_backgammon.clp".
4. Hacer 'reset' para introducir los hechos iniciales. Al hacer esto, se abrirá una ventana para verificar que quieres hacer 'reset'. Hay que clickar 'Sí'.
5. Hacer 'run' para empezar a jugar.

Hemos creado un pequeño menú donde el usuario puede elegir el modo de juego. Hay tres distintas opciones: humano contra humano, humano contra IA e IA contra IA (para la competición). Este último modo permite introducir los dados obtenidos por la IA rival.

A la hora de introducir las posiciones y los movimientos será el propio programa el encargado de verificar que estos sean correctos, y no incumplan ninguna regla.

Estructura del juego

En primer lugar, para la representación del juego hemos decidido utilizar un multicampo de 28 valores. Las primeras 24 posiciones hacen referencia al tablero, y las últimas 4 guardan las fichas comidas y las fichas en casa de cada jugador. Para el primer jugador se guardarán valores positivos, es decir, si tiene 3 fichas en la posición 10 el valor de la respectiva posición en el multicampo será +3. En cambio, se utilizan valores negativos para el segundo jugador.

Para empezar la partida hemos creado un 'fact' en el cual se dan los valores iniciales del tablero.

Hemos definido una regla para iniciar cada ronda, que nos sirve también para dar comienzo a la partida. Esta regla imprimirá el estado del tablero, indicará el turno y tirará los dados. Para tirar los dados hemos creado una función que devuelve dos valores aleatorios entre 1 y 6.

Reglas para la partida

Una vez ejecutada esta parte, tenemos tres reglas diferentes en lo que a las jugadas respecta.

La primera de ellas será la encargada de distinguir si en la tirada se ha obtenido doble o no, ya que éste es un caso especial. En caso de no ser dobles, se procederá pidiendo al usuario la posición de la ficha que quiere mover y el número de movimientos que quiere realizar. Antes de actualizar el tablero, hay que comprobar si la posición y el número de movimientos cumplen las normas del juego. Es decir, que el usuario no intente hacer movimientos ilegales o mover fichas inexistentes.

La segunda se ejecutará solamente cuando después del primer movimiento, el usuario aún puede mover otra ficha. Dicho esto, también se comprueba en cada una de las reglas si el jugador tiene alguna posibilidad de mover fichas, ya que, en caso contrario, el programa se estancaría.

La tercera regla contempla el caso de los dados dobles, donde el jugador tendrá una combinación más amplia de movimientos posibles. Para este caso hemos decidido cambiar la instancia de dados para poder guardar 2 o más.

Además, todas estas reglas obligan al jugador a sacar las fichas comidas en caso de haberlas, y de ser posible. De no ser posible ésto, se permite mover otras fichas del tablero. En caso de que sólo quede un movimiento posible y el jugador tenga una ficha comida que pueda mover, el movimiento se realiza de forma automática.

Reglas auxiliares

También hemos incluido ciertas reglas para marcar casos especiales o cambios de turno.

La regla de cambio de turno simplemente actualizará el valor del jugador una vez el otro haya terminado su jugada.

Las otras dos reglas que se han utilizado tienen relación con el final de la partida. Una de ellas informará sobre el resultado de la partida, y la otra preguntará al usuario si quiere volver a jugar.

Funciones

Para el desarrollo de las jugadas hemos utilizado varias funciones que nos servirán para comprobar que la partida se desarrolla de forma correcta. Además, muchas de las funciones están pensadas para la posterior implementación de un agente inteligente capaz de jugar al juego.

Las primeras funciones se pueden agrupar como funciones de petición, es decir, piden al usuario o a la IA que les den la posición y el número de movimientos. Siempre se comprobará que estos sean valores correctos.

Las siguientes son funciones de comprobación. Funciones booleanas que servirán para comprobar si se cumplen ciertos criterios. Por ejemplo, saber si hay fichas comidas o si el movimiento que el usuario quiere hacer es legal.

También hay dos funciones de búsqueda, que devolverán una lista con valores posibles, como el caso de buscarPosiciones, que devolverá una lista con todas las posiciones donde el jugador tiene alguna ficha.

Para terminar, tenemos dos funciones relacionadas con el tablero. Una para actualizarlo cuando sea necesario, y otra para mostrarlo por pantalla.

Implementación del agente inteligente

Primero de todo, hay que destacar que hemos decidido implementar este agente que va a jugar mediante funciones, ya que pensamos que el hilo del juego está bien implementado tal cual estaba. Por tanto, hemos decidido plantear la IA como un añadido al trabajo más general de implementar el juego. De esta forma es muy fácil modificar el juego para convertirlo de nuevo a 'user vs user' o incluso a 'IA vs IA'.

Dicho esto, a continuación vamos a presentar la implementación de nuestro agente inteligente.

El algoritmo de búsqueda que hemos elegido ha sido el Minimax, y estas son las razones de nuestra decisión:

1. El Backgammon es un juego de dos jugadores, lo que se adapta perfectamente a nuestro algoritmo, ya que de esta forma se buscará maximizar la puntuación de la IA y minimizar la del rival.
2. Disponemos de toda la información, por lo que el algoritmo podrá explorar todas las posibilidades antes de tomar una decisión.
3. Se puede implementar una función de evaluación heurística.

Ahora vamos a ver la estructura de nuestro algoritmo.

Algoritmo Minimax

El algoritmo tiene como uno de sus parámetros la profundidad máxima que visitará. Una vez se llegue a tal profundidad, se utilizará la función de evaluación sobre el tablero actual.

La función minimax cuenta con dos posibles vías, la del jugador que maximiza y la del que minimiza. Se aplica la recursividad sobre el mismo algoritmo para obtener la máxima y mínima evaluación, respectivamente.

Cada vez que la función se amplía hacia una posible combinación de movimiento y posición, se crea un tablero auxiliar actualizado con esta, para posteriormente evaluarla. Esta evaluación será posible siempre y cuando sea mayor que todas las anteriores (o

menor en el caso del jugador rival) y su correspondiente combinación cumpla las normas del juego.

Función de evaluación

Esta es la función que se utiliza para evaluar el tablero tal y como está. Ya hemos visto que se utilizará para evaluar los posibles movimientos. La función puede variar en base al objetivo que tengamos. En nuestro caso, hemos decidido implementarla de la siguiente forma:

- Se recompensa comer las fichas del rival, pero penaliza que el rival coma tus propias fichas.
- Hemos añadido una penalización por dejar fichas sueltas, que será mayor cuanto más cerca de casa estén.
- Se premia el acercar las fichas hacia las posiciones finales, donde es necesario tener todas las fichas para empezar a meterlas en casa.
- También se premia el meter las fichas en casa.

Funciones de llamada

Estas funciones serán las encargadas de llamar a las dos anteriores para saber cuál es el movimiento óptimo en base a la situación de la jugada.

Además, hemos añadido una función de actualización del tablero específica para la IA.

Conclusiones

Tras la realización del proyecto hemos llegado a la conclusión de que es muy importante entender bien la estructura de cualquier tipo de problema, como puede ser un juego u otra aplicación diferente, para poder implementar un agente inteligente.

En nuestro caso, la correcta implementación del juego nos ha llevado mucho tiempo, pero ha sido muy útil para entender bien cómo funciona. Esto ha hecho que posteriormente no haya sido complicado implementar la IA capaz de jugarlo.