Programación Lógica

Laboratorio 2 – 2017

Facultad de Ingeniería
Instituto de Computación
Grupo de Procesamiento de Lenguaje Natural

El objetivo de este obligatorio es modelar un juego en Prolog e implementar una inteligencia artificial para que lo juegue.

Nota previa - IMPORTANTE

Se debe cumplir íntegramente el "Reglamento del Instituto de Computación ante Instancias de No Individualidad en los Laboratorios", disponible en el EVA del curso.

En particular está prohibido utilizar documentación de otros grupos o de otros años, de cualquier índole, o hacer público código a través de cualquier medio (EVA, correo, papeles sobre la mesa, etc.).

Fútbol en papel

Fútbol en papel (en inglés *Paper soccer*) [1] es un juego para dos jugadores por turnos donde el objetivo es hacer un gol (uno solo) en el arco del rival. En la web se encuentran varias aplicaciones que implementan este juego [4, 5, 6, 7], que pueden usarse para tener una mejor idea de cómo es. Existen diferentes variantes del juego, a continuación describiremos las reglas de la variante que utilizaremos en el presente obligatorio, a la cual denominaremos *Futlog*.

El juego cuenta con un tablero de 10 filas y 8 columnas de casilleros, más dos arcos que ocupan dos casilleros cada uno y se ubican arriba y abajo. La pelota comienza en el centro del tablero y el jugador que comienza moviendo es el que ataca hacia arriba. El estado inicial del juego se muestra en la Figura 1.

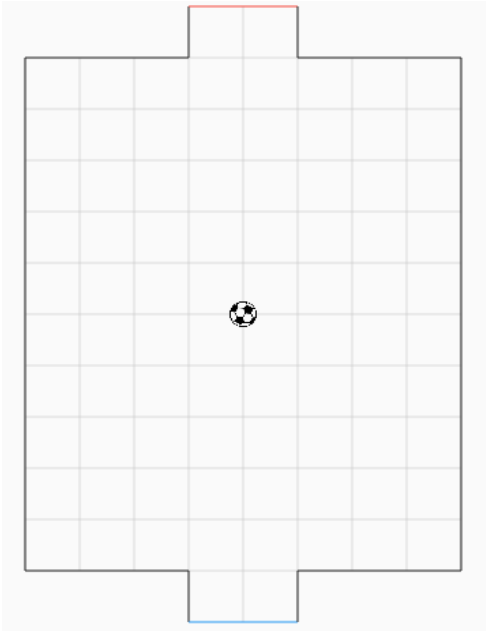

Figura 1: Estado inicial del juego

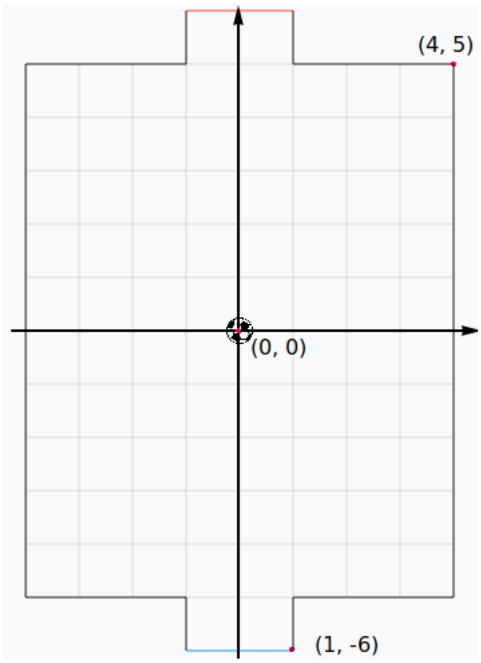

Figura 2: Sistema de coordenadas de las posiciones, con algunos puntos de ejemplo.

Movimientos

Las posiciones válidas son los vértices de los casilleros que están dentro del tablero, incluyendo sus bordes. La Figura 2 muestra el sistema de coordenadas que define las posiciones. Dos posiciones son adyacentes si se encuentran a distancia 1 en horizontal, vertical o diagonal. Los movimientos son caminos entre posiciones adyacentes que marcan (o visitan) las aristas por las que pasan, y que no repiten aristas anteriormente marcadas. Las aristas que delimitan los bordes del tablero se consideran marcadas desde el inicio. En cada turno, un jugador hace un movimiento. El jugador 1 deja trazos azules cuando mueve y el 2 deja trazos rojos.

Para poder definir más precisamente los movimientos, primero hay que distinguir dos tipos de posiciones: las que tienen alguna arista adyacente marcada (posiciones visitadas) y las posiciones que no tienen ninguna de sus aristas marcadas (posiciones no visitadas). Un movimiento consiste en mover la pelota desde donde está hacia una posición adyacente (sin pasar por una arista marcada), sucesivamente, y acaba cuando se cumple alguna de las siguientes condiciones: se llegó a una posición no visitada, se hizo un gol o el jugador se quedó sin movimientos (“atrapado”). En otras palabras, se debe seguir moviendo la pelota si se está en una posición visitada. La Figura 3 muestra movimientos válidos.

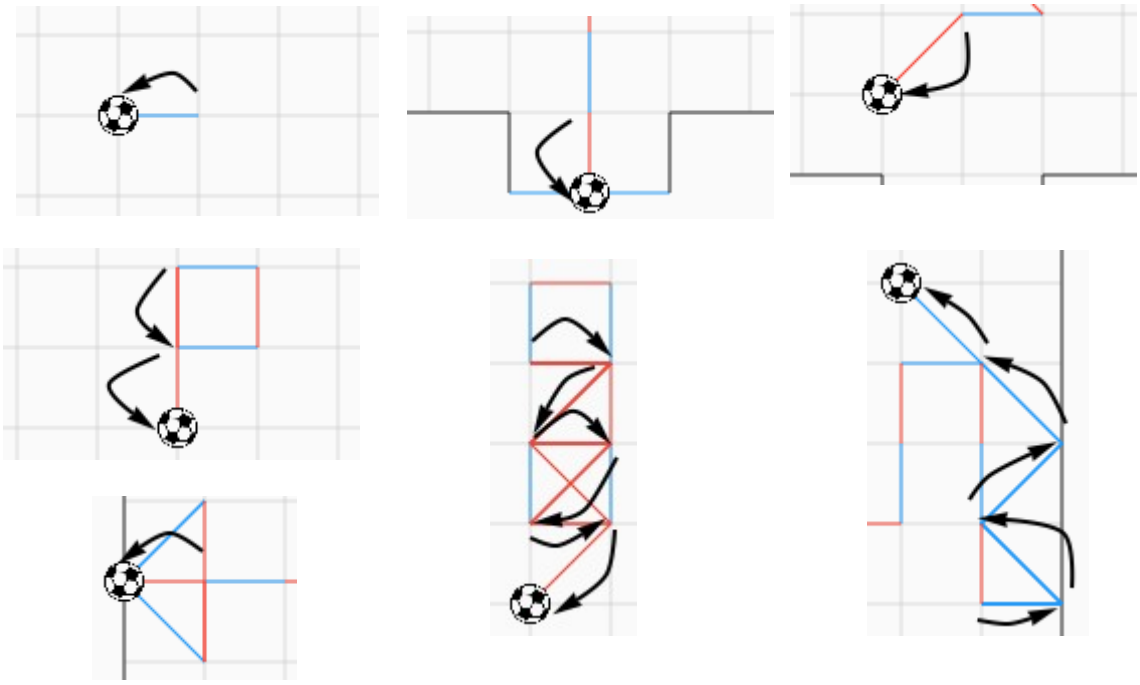

Figura 3: Ejemplos de movimientos válidos

Fin del juego

El juego termina cuando ocurre una de las siguientes condiciones:

- Un jugador mueve la pelota hacia uno de los 3 vértices del arco. En este caso hay *gol,* ganando la partida el jugador que tenía que hacer el gol a ese arco (“mete gol, gana”). Notar que el jugador 1, azul, ataca hacia arriba (hacia el arco rojo) y el jugador 2, rojo, ataca hacia abajo (hacia el arco azul).
- No hay movimientos para hacer. En este caso pierde el jugador que llegó a dicha posición.

Requerimientos a implementar

El juego intenta simular un escenario de implantación real, con una aplicación web consultando servicios implementados en Prolog (por ejemplo porque provee una forma simple de modelar un problema). Consiste de una interfaz web y una parte hecha en SWI-Prolog. La parte hecha en Prolog a su vez consta de 3 servidores web que aceptan pedidos HTTP: una que controla la lógica del juego y una inteligencia para cada jugador. Estos servicios usan una parte en común: los módulos *Juego* e *Inteligencia*, como muestra la Figura 4. Los servidores son independientes entre sí: son programas Prolog distintos y no se pueden interferir entre sí. Si bien utilizan los mismos módulos, potencialmente podrían ser distintos, permitiendo que jueguen dos inteligencias distintas entre sí y a la vez que la lógica de juego sea otra que las usadas por las inteligencias. Adicionalmente, los servidores recargan los módulos cada vez que se invoca un servicio para evitar tener que volver a levantarlos si se hacen cambios en los archivos Prolog. Esto facilita el desarrollo.

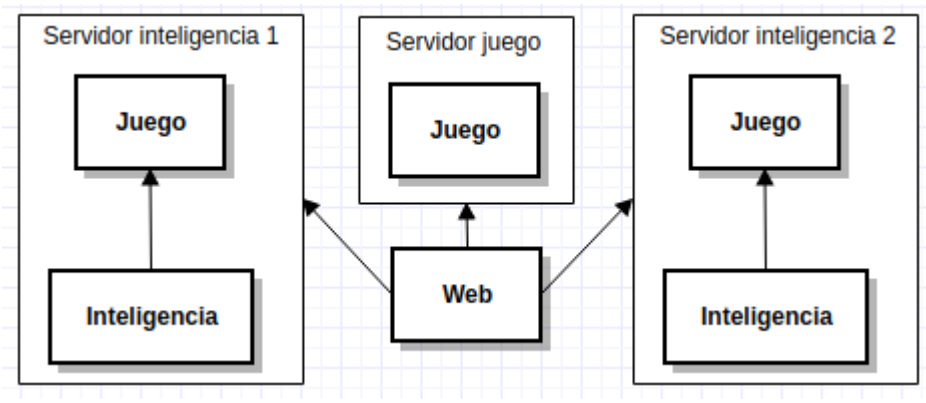

Figura 4: Componentes del sistema

La parte web se encarga de mostrar lo que ocurre en el juego y también de controlar al jugador Humano cuando juegue. Una captura de pantalla se muestra en la Figura 5. La parte web guarda un estado para cada servidor y comienza inicializándolos. Luego, entra en un ciclo, preguntando primero al servidor del juego a quién le toca jugar, y le pide al jugador que haga una jugada. Si es humano, la lógica se hará desde la web, llamando a predicados que permiten saber si el movimiento hecho hasta el momento es válido y si el movimiento en su totalidad también lo es. Si es la máquina, llamará al servidor de inteligencia correspondiente y le pedirá que haga una jugada. Luego de esto, verifica con el servidor de juego que la jugada sea correcta y le avisará al servidor del otro jugador qué movimiento hizo el rival. Cuando se da una condición de fin de juego, la web avisará y terminará el juego. El juego puede terminar por alguno de los siguientes motivos: hubo gol, un jugador se quedó sin movimientos, un jugador de tipo inteligencia agotó el tiempo de espera de 40 segundos (timeout) o debido a que un jugador de tipo inteligencia hizo un movimiento inválido según el servidor de juego.

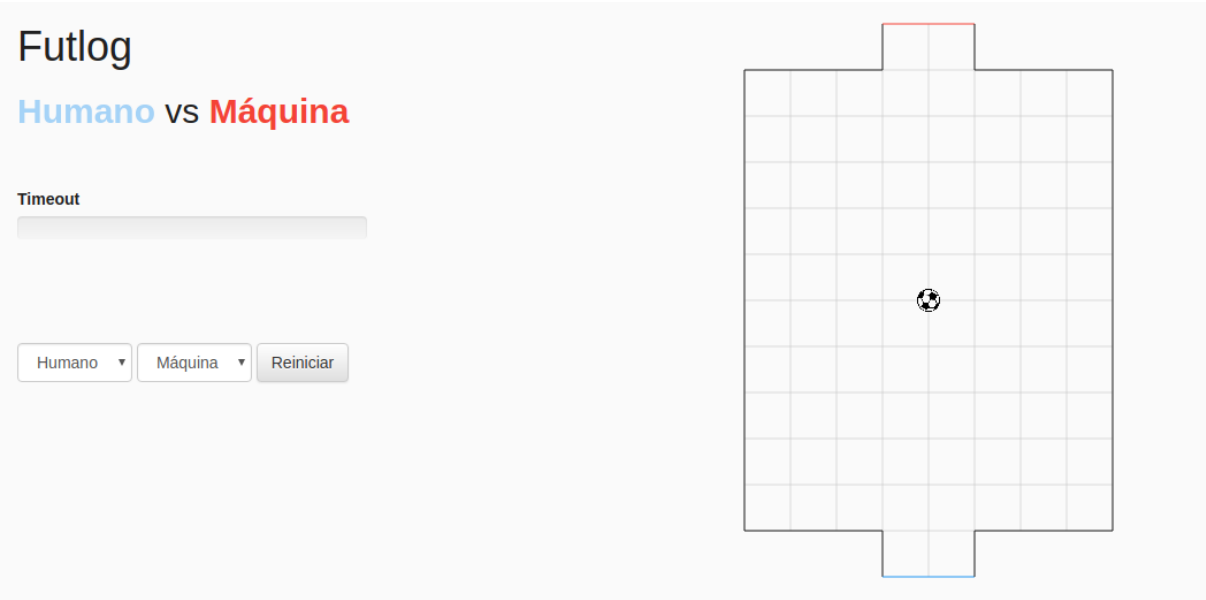

Figura 5: Captura de pantalla de la parte web

Se debe trabajar sólo en la parte Prolog, en donde se deben implementar los módulos **Juego** e **Inteligencia.** En los insumos liberados junto con la letra se encuentran definidos los predicados a implementar para estos módulos. La implementación debe realizarse de manera que pueda ser ejecutada en la plataforma SWI-Prolog.

Se recomienda implementar tests automatizados para los predicados definidos, de modo de poder comprobar en todo momento que la implementación funcione como debe, incluso para casos borde. Para esto, SWI-Prolog provee un framework llamado *Prolog Unit Tests* [3]. Los docentes del curso promovemos que compartan los tests que hagan con el resto de los grupos a través del EVA, mientras cumplan con el Reglamento indicando al comienzo del documento.

Inteligencia de la máquina

Dentro del módulo *Inteligencia*, se debe implementar el predicado **hacer_jugada** con una estrategia *buena* y *eficiente*. Se sugiere utilizar el algoritmo *Minimax* [2] para implementarla. La cantidad de niveles a utilizar en el algoritmo debe tomarse como un parámetro del sistema, utilizando el predicado **niveles_minimax** en *Inteligencia*.

El predicado debe poder dar una jugada a realizar en menos de 40 segundos, que es el tiempo límite que da la parte web para esperar una respuesta. Aunque el tiempo total para dar una jugada sea grande, se espera que la jugada promedio demore significativamente menos, ya que el juego se vuelve tedioso si siempre demora mucho en responder la inteligencia.

Ejecución del ambiente

El ambiente está pensado para funcionar en Windows y Linux.

Para levantar los servicios en Linux, usar el script **servicios**:

```
./prolog/servicios start
```

En Windows usar el script **servicios.bat**:

```
prolog\servicios.bat start
```

El script levanta los tres servicios web Prolog.

Para usar la parte web, alcanza con abrir el archivo **index.html**. Deben estar los servicios corriendo para que funcione correctamente.

Predicados a implementar

Los predicados a implementar en *Juego* son:

- **estado_inicial(?E)**: E es el estado inicial del juego, de tamaño AltoxAncho unificado por cantidad_casilleros(Ancho,Alto). La pelota comienza en la posición p(0,0), que es la posición central del tablero. El turno inicial es del jugador 1, que es el que patea hacia arriba. Los grupos son libres de modelar el estado del juego como quieran, mientras cumplan los requerimientos de los predicados.
- **posicion_pelota(+E,?P)**: P es la posición de la pelota para el estado E.
- **mover(+E,?LP,?E2)**: E2 el estado resultante de hacer un movimiento con la pelota, a través de las posiciones de la lista LP en el estado E y de cambiar el turno.
- **gol(+E,?Njugador)**: La pelota está en situación de gol a favor del jugador NJugador para el estado E.
- **turno(+E,?Njugador)**: NJugador es el jugador que tiene que mover en el siguiente turno para el estado E.
- **prefijo_movimiento(+E,+LP)**: LP es una lista no vacía de posiciones que constituyen el prefijo de un movimiento para el estado E, sin llegar a formar un movimiento. Este predicado permite validar jugadas del jugador *humano* mientras va eligiendo posiciones, para luego finalmente concretar el movimiento con *mover*.

Los predicados a implementar en *Inteligencia* son:

- **nombre(?Nombre)**: Nombre es el nombre de la inteligencia, que deberá ser "Grupo XX", siendo XX el número de grupo. Esto se usa en la web para saber contra quién se está jugando.
- **niveles_minimax(?MaxNiveles)**: MaxNiveles es la cantidad máxima de niveles de Minimax de la inteligencia.
- **hacer_jugada(+E,?LP,?E2)**: E2 es el estado resultante de mover según la lista de posiciones LP de un movimiento que la inteligencia elige jugar para el estado E. El turno de jugar en el estado E es el de la inteligencia, mientras que en E2 es del otro jugador.

Insumos

La estructura de carpetas de los insumos que se proveen es la siguiente:

web/
    css/: archivos CSS
    img/: imágenes
    js/: archivos JavaScript
    index.html: página HTML utilizada para interactuar con el juego
prolog/
    inteligencia.pl: módulo Prolog que se encarga de la inteligencia para el jugador *máquina*
    juego.pl: módulo Prolog que contiene predicados para consultar y realizar movimientos, así como también para consultar y cambiar el estado del juego
    servicios: script para levantar los servicios en Linux
    servicios.bat: script para levantar los servicios en Windows
    servidor_inteligencia.pl: módulo Prolog para los servidores de inteligencia
    servidor_inteligencia_unix.pl: módulo Prolog para los servidores de inteligencia en Unix
    servidor_juego.pl: módulo Prolog para el servidor de juego
    servidor_juego_unix.pl: módulo Prolog para el servidor de juego en Unix

Forma de entrega

La entrega se realizará a través del EVA del curso. Se debe entregar un solo archivo '**grupo##.zip**', donde ## es el número del grupo que realiza la entrega, conteniendo todos los módulos de la solución y el informe de la misma, ítems indicados en el apartado **Entregable**.

Fecha de entrega

Los trabajos deberán ser entregados siguiendo el procedimiento descrito anteriormente antes del lunes 12/06/2017 a las 23:55, sin excepciones. No se aceptará ningún trabajo pasada la citada fecha.

Entregable

El archivo a entregar debe contener:

- Todos los módulos Prolog, salvo los relacionados con los servidores. Esto incluye **inteligencia.pl**, **juego.pl** y todos los módulos auxiliares que se hayan implementado.
- Informe en formato PDF detallando la estructura de los módulos, los predicados principales y las decisiones de diseño tomadas.

Competencia (opcional)

Luego de entregada la tarea, habrá una instancia de competencia entre las inteligencias implementadas por los distintos grupos. Esta instancia es opcional y cada grupo debe indicar en el informe si desea o no participar. Para que se lleve a cabo, debe haber un mínimo de 4 equipos participantes. Se hará en un día, horario y salón a definir.

Consistirá de un campeonato de Futlog entre los grupos que participen, jugando de a 2 hasta llegar a un finalista. El formato del campeonato se definirá más adelante. Luego del campeonato, el equipo ganador contará sobre su implementación y se alentará a que los demás equipos también cuenten cuál fue su forma de implementar la estrategia.

Para esta parte, las inteligencias deben cumplir estrictamente la regla de tiempo límite y de movimientos válidos. En caso de no cumplirlas, se considera al jugador del grupo como perdedor del partido.

El equipo ganador tendrá 5 puntos extra para el total del curso y el equipo que salga segundo tendrá 2 puntos extra. Estos puntos no servirán si un estudiante reprueba el curso (y por tanto no definirán escenarios en el cual a un estudiante le falten pocos puntos para salvar curso). Notar que estos puntos son extra, y sin ellos se puede llegar de todas formas a la nota máxima. Simplemente sirven para aumentar los puntos obtenidos en el curso, permitiendo potencialmente pasarse de los 100 puntos (llegando, como mucho, hasta 105).

Referencias

[1] https://en.wikipedia.org/wiki/Paper_soccer

[2] http://en.wikipedia.org/wiki/Minimax

[3] http://www.swi-prolog.org/pldoc/doc_for?object=section(%27packages/plunit.html%27)

[4] https://www.playok.com/en/soccer/ (online entre personas)

[5] http://www.kongregate.com/games/k_meleon1982/paper-soccer (contra una máquina)

[6] https://play.google.com/store/apps/details?id=towe.papersoccer (para Android)

[7] https://itunes.apple.com/us/app/best-paper-soccer/id638148895?mt=8 (para iOS)