Laboratorio Inteligencia Artificial I

Sesión 7: Sistemas de reglas con encadenamiento hacia delante.

PRÁCTICA **4**
Fecha de entrega de la práctica completa: 20 de diciembre

El primer objetivo de esta práctica consiste en conocer las técnicas de representación de conocimiento basadas en reglas y, más concretamente, en aprender las características básicas del lenguaje CLIPS.
CLIPS fue un sistema experto desarrollado por la NASA durante la década de los ochenta, que tuvo una gran repercusión, ya que proporcionaba un entorno completo para la construcción de sistemas expertos basados en reglas e incluía también la posibilidad de trabajar con jerarquías de objetos. Al principio CLIPS se utilizó como una herramienta de entrenamiento para la construcción de sistemas expertos que también servía para su desarrollo y ejecución. Tras completarse el desarrollo de su primera versión y ser utilizado durante un año como periodo de prueba se demostró que era una alternativa de razonamiento más eficiente a otras herramientas que hacían el mismo trabajo. A mediados del año 1986 CLIPS (ya en su versión 3.0) fue liberado para grupos y usuarios que no perteneciesen a la NASA haciendo que se diese a conocer y fuese ganando más popularidad poco a poco. Con los años este sistema experto se ha continuado usando gracias al trabajo de sus desarrolladores, que han seguido actualizándolo y añadiendo algunas mejoras a pesar de que éste sea dominio público.

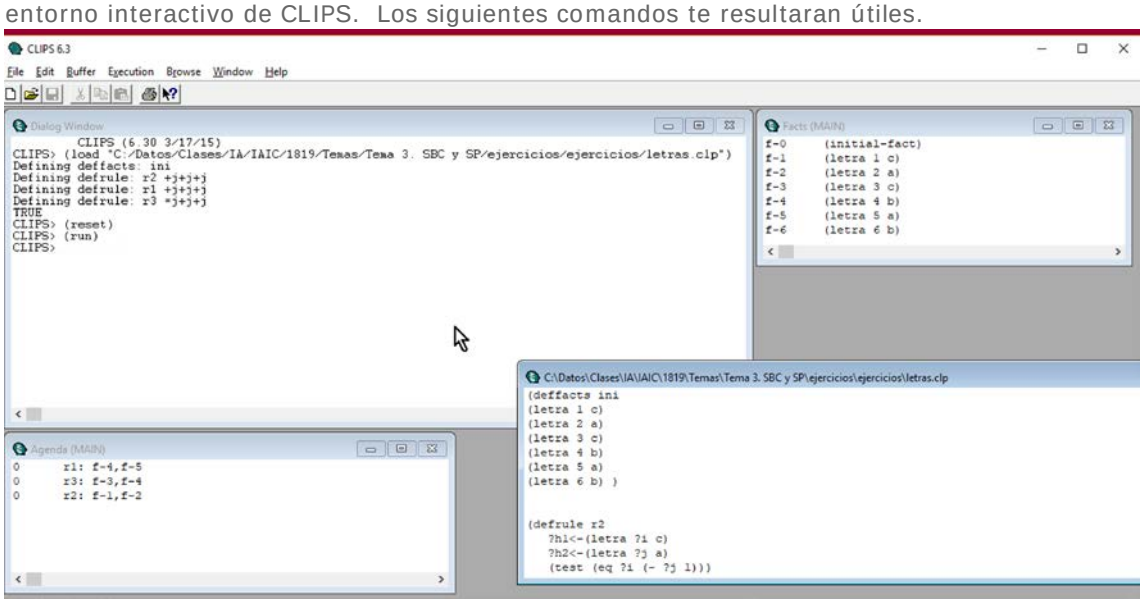
La sintaxis de CLIPS no es parecida a Python aunque hay algunas librerías que permiten integrarlo. Se puede por ejemplo incrustar código Python dentro del lenguaje CLIPS haciendo que sea muy fácil de extender. CLIPS tiene una sintaxis parecida al lenguaje LISP. La ventaja de CLIPS es que es un motor C sólido que puede integrarse completamente con otros sistemas, incluyendo Python (por ejemplo, clipspy o Pyknow), o Java (Jess). La ventaja es que las reglas se separan y cargan en tiempo de ejecución sin la necesidad de reiniciar el motor de inferencia. En la práctica no vamos a usar librerías ni sintaxis Python sino que vamos a utilizar sintaxis CLIPS y un entorno IDE interactivo para tomar contacto con el lenguaje y con el motor de inferencia.

Apartado 1. Toma de contacto

Para la toma de contacto se os proporcionan los siguientes archivos:

- Archivo de instalación del entorno IDE interactivo
- Ejemplos de archivos CLP para ejecutar
- Diapositivas de explicación de CLIPS

Los archivos de CLIPS tienen extensión clp. Podemos probar el comportamiento usando el entorno interactivo de CLIPS. Los siguientes comandos te resultaran útiles.



(facts): lista los hechos de MT
(clear) : eliminamos todos los hechos y construcciones de la memoria de trabajo.
(watch statistics): activa la visualización de estadísticas.
(reset): eliminamos todos los hechos de la memoria de trabajo, eliminamos las activaciones de la agenda y restauramos las condiciones iniciales: añadimos el hecho initial-fact que se utilizarán por las reglas de inicialización (reglas sin antecedente) y añade los hechos iniciales definidos con deffacts.
(set-strategy depth): cambiamos la estrategia de resolución de conflictos a Depth (muestra la estrategia que había anteriormente).

Se pide:

1. Ejecutar alguno de los ejemplos clp que os he dejado en el campus virtual observando la MT y el ciclo de reconocimiento actuación.
2. Cambiar la estrategia de resolución de conflictos observando si afecta o no la respuesta del sistema.

Apartado 2. Sistema experto y proceso de Ingeniería del conocimiento

Como hemos visto en clase, un sistema experto en Inteligencia Artificial es aquel que imita las actividades de un humano para resolver problemas que requieren experiencia humana, mediante el uso de representación del conocimiento y procedimientos de decisión. El conocimiento del experto en ese campo se organiza en una base de conocimiento, y en función de los datos disponibles de la aplicación (base de hechos) se imita la forma de actuar del experto explorando en la base de conocimientos hasta encontrar la solución (motor de inferencia). Los resultados finales y la forma en que se obtienen se expresan a través de la interfaz.

En este apartado vamos a definir un pequeño sistema experto que nos ayude a decidirnos sobre el tipo de vehículo (eléctrico, hibrido, enchufable, gasolina, diesel, gas GNC, GLP) que mejor se adecua a nuestras circunstancias y estilo de vida (economía, distancia, ciudad (puntos de recarga), viajes,..).

Se entregará el archivo .clp y un documento que explique el proceso de ingeniería del conocimiento que nos ayudará a definir qué información es relevante y qué reglas de decisión debemos incluir (se pueden usar páginas webs o entrevistas con algún experto, por ejemplo, un vendedor de un concesionario de coches).

Apartado 3.

En este apartado de la práctica ejecutaremos los archivos clp utilizando JESS que permite gestionar el motor de inferencia desde un programa JAVA. Usando un sistema de reglas se pide desarrollar los comportamientos de los 4 fantasmas y de Ms Pacman usando el simulador MsPacman vs Ghosts (más informacion en https://gaia.fdi.ucm.es/research/mspacman/). Para ello hay que utilizar los sensores y actuadores que permiten percibir y actuar con el entorno.

Ms. Pac-Man es un juego distribuido a partir de 1981 donde la señora Pac-Man tiene que conseguir el máximo número de puntos huyendo de los fantasmas Blinky, Pinky, Inky y Clyde. El objetivo del personaje es comer todos los puntos -pills- de la pantalla. Estos fantasmas son, respectivamente, de colores rojo, rosa, cian y naranja. Hay un “pasillo” a los costados del laberinto que permiten a Pac-Man o sus enemigos transportarse al costado opuesto (sale por la derecha y reingresa por la izquierda, o viceversa). Cuatro puntos más grandes de lo normal situados cerca de las esquinas del laberinto nombrados en inglés «Power Pills» (que en español lo han traducido en diversas formas como píldoras mágicas o de poder, bolas de energía o simplemente punto de poder), proporcionan a Pac-Man, durante un tiempo limitado, la habilidad de comerse él a los monstruos (todos ellos se vuelven azules mientras Pac-Man tiene esa habilidad), tras lo cual todo vuelve a ser como al principio. Después de haber sido comidos por Pac-Man, los fantasmas se regeneran en «casa» (una caja situada en el centro del laberinto).Se entregará un archivo clp con el comportamiento y se podrá visualizar el comportamiento utilizando el simulador. Se proporcionan ejemplos y material en el campus.