Práctica 6
Razonamiento y búsqueda con PROLOG

El primer objetivo de esta práctica es aprender a representar conocimiento y a resolver problemas de búsqueda en el espacio de estados.

Breve introducción a SWI-PROLOG

SWI-Prolog es un editor e intérprete de PROLOG, aunque para editar tus programas puedes usar un editor de texto si lo prefieres.

Para crear un programa nuevo, seleccionar File → New. Los archivos creados tendrán la extensión "`.pl`".

Para editar un programa existente, seleccionar File → Edit. Una vez que lo tengas editado elige la opción "`Save buffer`" para guardar los cambios.

Para ejecutar un programa tienes varias opciones:

- Seleccionar File → Consult
- Hacer doble click en los archivos "`.pl`" en tu explorador de archivos
- Ejecutar `[mi_programa]` en el intérprete de SWI-PROLOG si el directorio de trabajo contiene el archivo mi`_programa.pl`

Tras la ejecución se devolverá `true/false` indicando el éxito del proceso y una lista de warnings y errores como resultado de la interpretación del programa.

Si el programa cargó con éxito puedes hacer consultas en el intérprete. No olvides acabarlas con '`.`' y si quieres obtener respuestas adicionales puedes usar '`;`' que forzará el fallo y activará la vuelta atrás (backtracking) para intentar obtener más respuestas. Cuando no haya respuestas el intérprete te responderá con `false`.

1. Parentescos familiares

Construir un sistema de reglas en Prolog que permita obtener los distintos parentescos entre los miembros de una familia, asumiendo que los nombres de las personas son identificadores únicos. Se partirá de un conjunto de hechos iniciales con 2 tipos de predicados, uno que establece los progenitores de una persona y otro que establece el sexo.

- progenitores (Padre/Madre1, Padre/Madre2, Hijo/a).
    - progenitores (juan, maria, rosa).
    - progenitores (maria, juan, luis).
- persona (Nombre, Sexo)
    - persona (juan, hombre).

Partiendo de lo indicado en el archivo InstalacionSWI-Prolog (disponible en la carpeta Materiales Prolog, en el Campus Virtual) y del programa familia0.pl (en la misma carpeta),

añadir a este último las reglas necesarias para definir los predicados correspondientes a los siguientes parentescos: padre(X,Y), madre(X, Y), hijo(X, Y), hija(X, Y), hermano(X, Y), hermana(X, Y), abuelo(X, Y), abuela(X, Y), primo(X, Y), prima(X, Y) y ascendiente(X, Y). En cualquiera de estos predicados, p(X, Y), adoptaremos la semántica "X es p de Y" (por ejemplo, abuelo(X, Y) significa que X es abuelo de Y). Diremos que una persona X es un ascendiente de Y si es su padre/madre, abuelo/a, bisabuelo/a, etc. (la regla debe servir para cualquier nivel de ascendencia, no sólo los existentes en el ejemplo).

Ten en cuenta que en que el nombre de los padres puede aparecer en cualquier orden en el predicado *progenitores*. Por ello, es conveniente definir un predicado auxiliar progenitor(X, Y) que indica que X es el padre o la madre de Y.

Al construir las reglas correspondientes a padre, madre, hijo, hija, hermano y hermana se utilizarán como condiciones los hechos iniciales y el predicado progenitor. Para el resto de los parentescos no se utilizarán los hechos iniciales sino algunos de los nuevos predicados que se hayan definido (padre, madre, hermano, etc.).

Puede utilizarse la condición X\=Y para determinar si las variables X e Y tienen valores distintos. Los comentarios comienzan con un /* y terminan con */.

2. Las garrafas

Se tienen dos jarras vacías con capacidades de 3 y 4 litros, respectivamente, pero sin ninguna marca de medida parcial. Las jarras pueden vaciarse o llenarse completamente de agua, así como verter el contenido de una a otra. El objetivo consiste en tener exactamente 2 litros de agua en la jarra de 4 litros. Se pide:

- Representar el problema en Prolog, según el paradigma del espacio de estados, estableciendo una representación adecuada para el estado del problema y definiendo el estado inicial, el estado final y las operaciones permitidas (llenar la jarra de 3 litros, llenar la jarra de 4 litros, vaciar la jarra de 3 litros, vaciar la jarra de 4 litros, verter agua de la jarra de 3 litros a la de 4 litros hasta que no quede nada en la de 3 o hasta que no quepa más en la de 4, verter agua de la jarra de 4 litros a la de 3 litros hasta que no quede nada en la de 4 o hasta que no quepa más en la de 3).

- Construir un predicado Prolog que permita encontrar una solución para pasar del estado inicial al estado final, evitando la repetición de estados. El predicado devolverá la lista de operaciones que es necesario realizar y la profundidad de la solución encontrada. Definir el predicado consulta que resuelva el problema e imprima los resultados.

3. Las garrafas generalizadas

Considera ahora una nueva versión del problema en el que las capacidades de las garrafas son C1 y C2 litros respectivamente y el objetivo es conseguir exactamente L litros en cualquiera de ellas. Las dos garrafas empezarán siempre vacías.

Crea un nuevo programa que resuelva esta versión generalizada teniendo en cuenta que los tres parámetros deben poder establecerse en la consulta.

4. Memoria

La memoria de la práctica deberá incluir una justificación breve de la implementación de las distintas partes de la práctica, además ejemplos de consultas y los resultados de su ejecución.

Se debe incluir en la memoria el nombre de los integrantes del grupo.

5. Entrega de la práctica

La práctica se entregará a través del Campus Virtual. Se habilitará una nueva tarea: **Entrega de la Práctica 6** que permitirá subir un zip que contendrá los dos ficheros pl con la solución de la práctica, debidamente documentados por medio de comentarios, y la memoria.

El fichero subido deberá tener el siguiente nombre: **Practica6GXX.zip**, siendo XX el nombre del grupo. Por ejemplo, *Practica6G03.zip*.

Uno sólo de los miembros del grupo será el encargado de subir la práctica.

No se corregirá ninguna práctica que no cumpla estos requisitos.