Programación Declarativa

Sesión de laboratorio 7

Curso 2021/22

PARTE 1. Para experimentar.

Considera la siguiente especificación:
elimina$(L, X, NL) \leftrightarrow NL$ es la lista resultante de eliminar de la lista L las apariciones de X.
Estudia las diferencias entre las siguientes definiciones de este predicado escritas en Prolog.

- Usando igualdad sintáctica:

```
elimina1([ ],X,[ ]).
elimina1([X|R],Y,NR) :- Y == X, elimina1(R,Y, NR).
elimina1([X|R],Y,[X|NR]) :- Y \== X, elimina1(R,Y,NR).
```

- Usando unificación:

```
elimina2([ ],X,[ ]).
elimina2([X|R],Y,NR) :- Y = X, elimina2(R,Y, NR).
elimina2([X|R],Y,[X|NR]) :- Y \= X, elimina2(R,Y,NR).
```

- Combinando las dos anteriores:

```
elimina3([ ],X,[ ]).
elimina3([X|R],X,NR) :- elimina3(R,X,NR).
elimina3([X|R],Y,[X|NR]) :- Y \== X, elimina3(R,Y,NR).
```

Ejecuta los siguientes objetivos en cada una de las tres versiones, pidiendo todas las respuestas posibles.
`?- eliminai([a,b,a,c],a,L). (i= 1,2,3)`
`?- eliminai([a,b,a,c],X,L). (i= 1,2,3)`
Compara los resultados. ¿Qué puedes concluir acerca de cada una de las tres versiones, según el modo de uso de los argumentos?

PARTE 2. Para programar y entregar.

Realiza los siguientes ejercicios en un mismo fichero con extensión .pl.
Utiliza el predicado del corte siempre que sea útil.

1. Utilizando la estructura de árbol binario definida en clase y el *if then else* de Prolog, programa el predicado:
   maximo$(A, X) \leftrightarrow A$ es un árbol binario de números enteros positivos dado y X es el elemento máximo de sus nodos. X vale 0 si el árbol es vacío.

2. Define en Prolog el predicado:
   sublista(SXs,Xs) $\leftrightarrow$ *SXs* es una sublista de la lista *Xs*.

   Usando los metapredicados de recolección de respuestas programa:
   sublistas(LSX,Xs) $\leftrightarrow$ *LSX* es la lista que contiene a todas las listas de *Xs*.

3. Programa en Prolog el siguiente predicado sin utilizar predicados predefinidos:

   aplana$(Xss, Xs) \leftrightarrow Xs$ es la lista que resulta de eliminar **todos** los corchetes en los subtérminos de la lista `Xss` que sean listas. Equivale a iterar `concat` de Haskell, pero ten en cuenta que en Prolog los elementos de las listas no tienen por qué ser todos del mismo tipo. Por ejemplo, el objetivo `aplana([[1],[a,b],[],2,[[c],3]],Xs).` da como resultado `Xs = [1,a,b,2,c,3]`.

4. Programa en Prolog una versión recursiva para resolver el problema de las torres de Hanoi. Para ello, define un predicado:

   hanoi(N, A, B, C, M) donde N es el número de fichas que hay que mover de la torre inicial, A es el nombre de la torre inicial, B es el nombre de la torre final, C es el nombre de la torre auxiliar y M es la secuencia de pares de movimientos de una torre a otra, hasta conseguir la traslación de todas las fichas. Por ejemplo, para trasladar dos fichas de las torres con nombres, `ini, fin, aux`, respectivamente, la lista de movimientos sería: `[(ini, aux), (ini, fin), (aux, fin)]`.