PRÁCTICA 2: PROGRAMACIÓN EN PROLOG
Curso 2018-2019

Ordenar por un criterio arbitrario

Se pide escribir un programa para ordenar listas en el que el criterio de comparación no esté predefinido (se da en la ejecución) y que además lleve a cabo la ordenación mediante el algoritmo de "árboles flotantes" que se explica más abajo. La práctica consta de las siguientes partes:

1. Programar un predicado `menor(A,B,Comp,M)`, que devuelva en `M` el menor entre `A` y `B` usando `Comp` como criterio de comparación. `Comp` es el nombre de un predicado que determina cuándo un elemento es igual o mayor que otro. Por ejemplo, `menor(3,4,<,M)` devuelve `M=3`.

2. Programar un predicado `menor_o_igual/2` que determina si su primer argumento es menor o igual al segundo de la siguiente forma. Una variable libre es igual a cualquier otro término. Un término es menor que otro, no siendo ninguno variable libre, si su nombre es `@<` que el del otro, si los nombres son idénticos pero su aridad es `<` que la del otro, o nombres y aridades son idénticas pero alguno de los argumentos del primer término es menor que el correspondiente argumento del segundo término según el siguiente procedimiento:

   Se recorren los argumentos de cada término de izquierda a derecha y se comparan los de la misma posición. Si el del primero es menor que el del segundo el primer término es menor que el segundo. Si son iguales se continua el proceso. Si todos los argumentos son iguales, ambos términos son iguales.

3. Programar un predicado `ordenar(Lista,Comp,Orden)` tal que la lista `Orden` es la resultante de ordenar la lista `Lista` utilizando `Comp` como criterio de comparación. Para llevar a cabo la ordenación usaremos árboles flotantes. El árbol inicial contiene como hojas los elementos de la lista de entrada. En un árbol flotante cada nodo interno tiene dos hijos exactamente y su valor es el menor de los de sus hijos. El proceso de tomar el valor de la raíz del árbol flotante, reflotarlo y continuar hasta que el árbol esté vacío obtiene la lista ordenada deseada.

   Se pide seguir los siguientes pasos:

   - Programar un predicado lista_hojas(Lista,Hojas) que, dada una lista, devuelve otra con las hojas que compondrán el árbol:

     ```
     lista_hojas([1,2,3],Hojas)
     Hojas = [tree(1,void,void),tree(2,void,void),tree(3,void,void)]
     ```

   - Programar un predicado hojas_arbol(Hoja, Com, Arbol) que, dada la lista de hojas devuelve el árbol flotante inicial:

     ```
     tree(1,tree(1,tree(1,void,void),tree(2,void,void)),tree(3,void,void))
     ```

   - Programar un predicado `ordenacion(Arbol,Comp,Orden)` que, dado el árbol inicial, devuelve en `Orden` la lista ordenada.

     El valor de la raíz de este árbol es el menor de los elementos de la lista, por lo que este elemento se sitúa en primer lugar de la lista ordenada de salida. A continuación se "reflota"el árbol y su nueva raíz será el siguiente elemento de la lista ordenada. Y así sucesivamente.

     En el árbol flotante todos los nodos del camino que lleva de la raíz a la hoja que tiene su mismo valor tienen ese mismo valor, ya que es el menor de todos. Para reflotar el árbol se debe recorrer dicho camino desde la hoja hasta la raíz procediendo como sigue. El nodo hoja se elimina (sustituyéndolo por un árbol vacío). Cada nodo que tenga un solo hijo se sustituye por dicho único hijo (un nodo tal será el padre de la hoja que se ha eliminado). A cada nodo con dos hijos se le asigna como valor el menor de los de sus hijos.

   - Programar `ordenar(Lista,Comp,Orden)` con estos elementos. El segundo argumento de `ordenar/3` debe permitir ordenar según cualquier criterio deseado, siempre que exista un predicado que se pueda utilizar para definir dicho criterio. Por ejemplo, para la ordenación tradicional de listas de números se utilizaría la llamada `ordenar([...],'=<',Orden)`. O, definiendo un predicado `menor_o_igual/2` como el de arriba que determina un criterio no estándar de comparación de términos se podrían ordenar listas según dicho criterio con llamadas como `ordenar([...],menor_o_igual,Orden)`.