PROGRAMACIÓN DECLARATIVA

LABORATORIO 4 (Arboles)

Para medir lo aprendido durante la Práctica 4, los alumnos deberán enviar los programas y soluciones a las preguntas que a continuación se indican. El envío se realizará en un archivo ZIP de nombre:

gN_Apellidos_Lab4.ZIP

donde N es el número del equipo de trabajo. El archivo ZIP contendrá los ficheros:
Apellidos_Lab4.pl con el código Prolog de todos los programas solicitados.
Apellidos_Lab4.txt con los datos personales de la persona que hace el envío y el enunciado y solución de aquéllas preguntas que (no pudiéndose responder vía programa) se formulen en cada uno de los ejercicios del laboratorio.

En todos los casos, "**Apellidos**" son los apellidos del alumno que hace el envío.

ENUNCIADO

Los árboles n-arios de tipo T o rosadelfas se caracterizan por ser:
1.Bien una estructura vacía,
2.o bien una estructura constituida por un elemento de tipo T, denominado nodo, junto con m (siendo 0 =< m =< n) subconjuntos disjuntos de elementos; estos subconjuntos son a su vez árboles n-arios de tipo T, que se denominan subárboles del árbol original.
El "grado" de un nodo es su número de hijos y el "grado" un árbol, el grado máximo de sus nodos.

Ejercicio 14.
Las rosadelfas se han representado en Prolog mediante los constructores:
1.nil;
2.hoja(X), donde X elemento de tipo T;
3.nodo(X, [T_1, ..., T_M]), donde X es un elemento de tipo T, T_i es un árbol n--ario de tipo T, y M =< N.
El predicado esRosadelfa(O) permite determinar si un objeto O es o no una rosadelfa.

```
% esRosadelfa(O), el objeto O es una rosadelfa.
esRosadelfa(nil).
esRosadelfa(hoja(_)).
esRosadelfa(nodo(_, Rosadelfas)) :-
esListaRosadelfas(Rosadelfas).

esListaRosadelfas([R]) :- esRosadelfa(R).
esListaRosadelfas([R|Rosadelfas]) :- esRosadelfa(R),
```

```
esListaRosadelfas(Rosadelfas).
```

Sin embargo, en la definición anterior no se hace ninguna comprobación del tipo de los elementos o sobre el número de hijos (o grado) de cada nodo. Modificar el predicado anterior y definir un predicado esRosadelfa(O, N) que determine si un objeto O es una rosadelfa de grado N y realice las comprobaciones de tipo oportunas.

Ejercicio 15.
Defina los siguientes predicados acerca de los árboles n-arios:
a) **peso(A, P)** que calcule el peso P de un árbol n-ario A, entendiendo por "peso" el número de nodos que contiene dicho árbol. Esta magnitud también recibe el nombre de "tamaño" del árbol.
b) **grado(A, G)** que calcule el grado G de un árbol n-ario A; el "grado" de un nodo es el número de hijos de ese nodo y el "grado" de un árbol es el grado máximo de los nodos que lo componen.
c) **frontera(A, F)** que permite determinar la frontera F del árbol n-ario A. La frontera de un árbol es la lista de sus hojas.
d) **preorden(A, L)** que permita recorrer los nodos del árbol n-ario A en orden preorden, obteniendo la lista L de nodos visitados y en el orden que fueron visitados. Un recorrido preorden consiste en visitar primero la raíz y después los subárboles de izquierda a derecha.

Ejercicio 16.
Al definir el predicado **construirRosadelfa(L, G, R)** se pretende construir una rosadelfa R, de grado G a partir de los elementos de una lista L. Más abajo se presenta una solución que aparece en la página 218 del libro PROGRAMACIÓN LÓGICA, TEORÍA Y PRÁCTICA, cuyos autores son María Alpuente y Pascual Julián. En realidad, dicha solución no hace lo que se pretende. Ante un objetivo como:

```
?- construirRosadelfa([1,2,3,4,5,6,7,8,9,10,11,12,13], 2, R).
```

El interprete responde construyendo el árbol:

```
R =
nodo(1,
[nodo(2, [hoja(3)]),
nodo(4, [hoja(5)]),
```

```
nodo(6, [hoja(7)]),
nodo(8, [hoja(9)]),
nodo(10, [hoja(11)]),
nodo(12, [hoja(13)]),
nil
]
)
```

que no es de grado 2.

El objetivo de este ejercicio es modificar el código del programa para que se atenga a su especificación. Esto es construya una rosadelfa de grado G.

```
% construirRosadelfa(L, G, R), construye una rosadelfa R, a
partir de los elementos de una lista L.
construirRosadelfa([], _, nil).
construirRosadelfa([X], _, hoja(X)).
construirRosadelfa([X|L], G, nodo(X, [R|Rosadelfas])):-
partir(L, G, [L1|GListas]),
construirRosadelfa(L1, G, R),
construirRosadelfas(GListas, G, Rosadelfas).

% inspecciona una lista de listas y por cada lista inspeccionada
crea una rosadelfa
construirRosadelfas([], _, []).
construirRosadelfas([L|GListas], G, [R|Rosadelfas]) :-
construirRosadelfa(L, G, R),
construirRosadelfas(GListas, G, Rosadelfas).
% partir(L, G, [L1|GListas]), divide la lista L en una secuencia
de listas de longitud
% menor o igual que G, que se devuelve como una lista de listas
en el segundo argumento.
partir(L, G, [L]):- length(L,N), N < G.
partir(L, G, [L1|GListas]):- length(L,N), N >= G,
length(L1, G),
append(L1, LL, L),
partir(LL, G, GListas).
```

La definición del predicado partir, en su formulación actual, trata dos casos: i) si la longitud de la lista L es menor que G, entonces es un resto que debe unirse directamente a la secuencia de listas; ii) en caso contrario, se fragmenta la lista L en un prefijo de longitud G y un resto de lista LL, que se sigue partiendo, mediante la llamada recursiva a partir. Observe que la llamada length(L1,G) puede utilizarse para crear una lista L1 que contenga G variables, que posteriormente se instanciarán. Si lanzamos el objetivo "**`?- partir([1,2,3,4,5,6,7,8,9,10,11,12,13], 2, L).`**", se obtiene como respuesta:

```
L = [[1, 2], [3, 4], [5, 6], [7, 8], [9, 10], [11, 12], [13]]
```

Ahí podría estar la fuente del error. Se sugiere modificar el comportamiento del predicado "partir" para que construya la lista:

```
L = [[1, 2, 3, 4, 5, 6, 7], [8, 9, 10, 11, 12, 13]]
```

y el resto de predicados pueda generar una rosadelfa equilibrada de grado G=2.

Última modificación: jueves, 13 de marzo de 2014, 19:25:25