Programación declarativa **(9–Febrero–2005)**

Apellidos:

Nombre:

Notas

1. *En la evaluación de los ejercicios se tendrá en cuenta la corrección, simplicidad y eficiencia de la respuesta.*
2. *Hay que describir las definiciones auxiliares (menos las del sistema).*

Ejercicio 1 [2 puntos] Definir la relación `comprimida(+L1,-L2)` que se verifique si `L2` es la lista obtenida a partir de la lista `L1` comprimiendo las ocurrencias sucesivas de un elemento a una única ocurrencia de dicho elemento. Por ejemplo,

```
?- comprimida([a,a,a,a,b,c,c,a,a,d,e,e,e,e],X).
X = [a,b,c,a,d,e]
```

Dar dos definiciones, una sin corte y otra con corte.

.....

Solución: La definición sin corte es

```
comprimida([],[]).
comprimida([X],[X]).
comprimida([X,X|L1],L2) :-
   comprimida([X|L1],L2).
comprimida([X,Y|L1],[X|L2]) :-
   not(X = Y),
   comprimida([Y|L1],L2).
```

La definición con corte es

```
comprimida([],[]).
comprimida([X],[X]).
comprimida([X,X|L1],L2) :-
   !,
   comprimida([X|L1],L2).
comprimida([X,Y|L1],[X|L2]) :-
   % not(X = Y),
   comprimida([Y|L1],L2).
```

Ejercicio 2 [2 puntos] Definir la relación `ocurrencias(+C,+E,-N)` que se verifique si `N` es el número de ocurrencias de la constante `C` en la expresión aritmética `E`. Por ejemplo,

```
?- ocurrencias(a,(a+(2-b)*(a+3))*a,N).
N = 3
?- ocurrencias(3,(a+(2-b)*(a+3))*a,N).
N = 1
```

.....

Solución: La definición de `ocurrencias` es

```
ocurrencias(X,X,1).
ocurrencias(X,E,0) :-
   X \= E,
   atomic(E).
ocurrencias(X,E,N) :-
   E =..[_,A,B],
   ocurrencias(X,A,N1),
   ocurrencias(X,B,N2),
   N is N1+N2.
```

Ejercicio 3 [3 puntos] Los grafos dirigidos se pueden representar por listas de pares de elementos denotando una flecha del primero al segundo. Por ejemplo, la lista `[a-b,b-c]` representa un grafo en el que se puede ir de `a` a `b` y de `b` a `c`; nótese que, por ser dirigido, hay un camino de `a` a `c` pero no de `c` a `a`.

Definir la relación `ciclos_minimales(+G,-L)` que se verifique si `L` es el conjunto de los ciclos minimales del grafo dirigido `G`. Por ejemplo,

```
?- ciclos_minimales([a-b,b-c,c-c,a-d,d-a],L).
L = [[a, d, a], [c, c]]
?- ciclos_minimales([a-b,b-c],L).
L = []
?- ciclos_minimales([a-b,b-c,c-a],L).
L = [[a, b, c, a]]
?- ciclos_minimales([a-b,b-c,c-a,d-b,b-e,e-d],L).
L = [[a, b, c, a], [b, e, d, b]]
```

.....

Solución: La definición es

```
ciclos_minimales(G,L) :-
   findall(C,(ciclo_minimal(G,C),ciclo_principal(C)),L).
```

La relación `ciclo_minimal(+G,-C)` se verifica si `C` es un ciclo minimal del grafo dirigido `G`. Por ejemplo,

```
?- ciclo_minimal([a-b,b-c,c-c,a-d,d-a],L).
L = [a, d, a] ;
L = [c, c] ;
L = [d, a, d] ;
No
```

y se define por

```
ciclo_minimal(G,C) :-
   nodo(X,G),
   camino(X,X,G,C).
```

La relación `nodo(?X,+G)` se verifica si `X` es un nodo del grafo `G`. Por ejemplo,

```
?- nodo(X,[a-b,b-c,b-d,c-d]).
X = a ;
X = b ;
X = c ;
X = d ;
No
```

y se define por

```
nodo(X,G) :-
   nodos(G,L),
   member(X,L).
```

La relación `nodos(+G,?L)` se verifica si `L` es el conjunto de los nodos del grafo `G`. Por ejemplo,

```
?- nodos([a-b,b-c,b-d,c-d],X).
X = [a, b, c, d]
```

y se define por

```
nodos(G,L) :-
   setof(X,Yˆadyacente(G,X,Y),L).
```

La relación `adyacente(+G,?X,?Y)` se verifica si `X` e `Y` son nodos adyacentes en el grafo `G`. Por ejemplo,

```
?- adyacente([a-b,b-c,b-d,c-d],c,Y).
Y = d ;
Y = b ;
No
```

y se define por

```
adyacente(G,X,Y) :-
   member(X-Y,G) ; member(Y-X,G).
```

La relación `camino(+A,+Z,+G,-C)` se verifica si `C` es un camino de `A` a `Z` en el grafo `G`. Por ejemplo,

```
?- camino(a,d,[a-b,b-c,b-d,c-d],C).
C = [a, b, d] ;
C = [a, b, c, d] ;
No
```

y se define por

```
camino(A,Z,G,C) :-
   camino_aux(A,[Z],G,C).
```

La relación `camino_aux(+A,+CP,+G,-C)` se verifica si `C` es una camino en `G` compuesto de un camino desde `A` hasta el primer elemento del camino parcial `CP` (con nodos distintos a los de `CP`) junto `CP`. Se define por

```
camino_aux(A,[Y|C1],G,[A,Y|C1]) :-
   member(A-Y,G).
camino_aux(A,[Y|C1],G,C) :-
   member(X-Y,G),
   not(member(X,[Y|C1])),
   camino_aux(A,[X,Y|C1],G,C).
```

La relación `ciclo_principal(+C)` se verifica si el primer elemento del ciclo `C` es su menor elemento. Por ejemplo,

```
?- ciclo_principal([a,b,c,a]).
Yes
?- ciclo_principal([b,c,a,b]).
No
```

y se define por

```
ciclo_principal([X|L]) :-
   sort([X|L],[X|_]).
```

Segunda parte [3 puntos] En el laboratorio.
