Programación declarativa **(29–Noviembre–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 [2.5 puntos] Se considera una base de conocimiento con propiedades de personas expresadas mediante relaciones binarias con el primer argumento el nombre de la persona y el segundo su valor. Por ejemplo,

```
altura(juan,1.70).
altura(eva,1.80).
altura(pepe,1.60).
peso(eva,82).
peso(juan,73).
peso(pepe,82).
```

1. Definir la relación `valor_medio(+P,-N)` que se verifique si `N` es el valor medio de la propiedad `P`, Por ejemplo, con la base de conocimiento anterior,

```
?- valor_medio(altura,L).
L = 1.7
?- valor_medio(peso,L).
L = 79
```

2. Definir la relación `máximo(+P,-N)` que se verifique si `N` es el máximo valor de la propiedad `P`. Por ejemplo, con la base de conocimiento anterior,

```
?- máximo(altura,N).
N = 1.8
?- máximo(peso,N).
N = 82
```

[**Nota**: La definición debe de ser aplicable a cualquier base de conocimiento del tipo del ejemplo].

.....

Solución: Solución del apartado 1:

```
valor_medio(P,N) :-
    findall(X,apply(P,[_,X]),L),
    suma_lista(L,S),
    length(L,M),
    N is S/M.
```

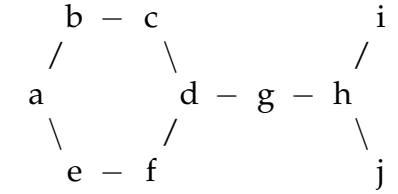
donde `suma_lista(+L,-N)` se verifica si `N` es la suma de los elementos de la lista numérica `L`.

```
suma_lista([],0).
suma_lista([X|L],N) :-
    suma_lista(L,M),
    N is X+M.
```

Solución del apartado 2:

```
máximo(P,N) :-
    apply(P,[_,N]),
    not((apply(P,[_,M]), N<M)).
```

Ejercicio 2 [2.5 puntos] Un grafo puede representarse mediante los pares de sus ejes. Por ejemplo, el grafo



puede representarse por `[a-b,a-e,b-c,c-d,d-f,d-g,e-f,g-h,h-i,h,j]`.

Definir la relación `circulo(+G,+N,+R,-L)` que se verifique si `L` es la lista de los nodos del grafo `G` que se encuentra a una distancia del nodo `N` menor o igual que `R`. Por ejemplo,

```
?- circulo([a-b,a-e,b-c,c-d,d-f,d-g,e-f,g-h,h-i,h-j],g,0,L).
L = [g]
?- circulo([a-b,a-e,b-c,c-d,d-f,d-g,e-f,g-h,h-i,h-j],g,1,L).
L = [d, g, h]
?- circulo([a-b,a-e,b-c,c-d,d-f,d-g,e-f,g-h,h-i,h-j],g,2,L).
L = [c, d, f, g, h, i, j]
```

.....

Solución:

```
circulo(G,N,R,L) :-
    findall(N1,((nodos(G,L),
                 member(N1,L),
                 distancia(G,N,N1,X),
                 X =< R)),
            L).
```

Se han usado las siguientes funciones auxiliares:

- `nodos(+G,-L)` se verifica si `L` es la lista de los nodos del grafo `G`. Por ejemplo,

```
?- nodos([a-b,a-e,b-c,c-d,d-f,d-g,e-f,g-h,h-i,h,j],L).
L = [a, b, c, d, e, f, g, h, i]
```

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

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

```
?- adyacente([a-b,a-e,b-c,c-d,d-f,d-g,e-f,g-h,h-i,h,j],g,Y).
Y = h ;
Y = d ;
No
```

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

- `distancia(+G,+N1,+N2,-D)` se verifica si `D` es la distancia del camino más corto del nodo `N1` al `N2` en el grafo `G`. Por ejemplo,

```
?- distancia([a-b,a-e,b-c,c-d,d-f,d-g,e-f,g-h,h-i,h,j],g,f,D).
D = 2
```

```
distancia(G,N1,N2,D) :-
    camino(G,N1,N2,[_|C]),
    length(C,D),
    not((camino(G,N1,N2,[_|C2]), length(C2,D2), D2<D)).
```

- `camino(+G,+A,+Z,-C)` se verifica si `C` es un camino en el grafo desde el nodo `A` al `Z`. Por ejemplo,

```
?- camino([a-b,a-e,b-c,c-d,d-f,d-g,e-f,g-h,h-i,h,j],g,f,C).
C = [g, d, f] ;
C = [g, d, c, b, a, e, f] ;
No
```

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

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

```
?- camino_aux([a-b,a-e,b-c,c-d,d-f,d-g,e-f,g-h,h-i,h,j],g,[e,f
C = [g, d, c, b, a, e, f] ;
No
```

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

Ejercicio 3 [2 puntos] Supongamos que se definen las gramáticas mediante reglas de producción con el operador `--->` como, por ejemplo, la definida en `ejemplo_gramatica.pl`

```
:- op(1200,xfx,--->).

oración             ---> sintagma_nominal, sintagma_verbal.
sintagma_nominal    ---> nombre.
sintagma_nominal    ---> artículo, nombre.
sintagma_verbal     ---> verbo, sintagma_nominal.
artículo            ---> [el].
nombre              ---> [gato].
nombre              ---> [perro].
nombre              ---> [pescado].
nombre              ---> [carne].
verbo               ---> [come].
```

Definir la relación `deriva(+C,?LD)` que se verifica si la lista de diferencias `LD` es de la categoría gramatical `C`. Por ejemplo,

```
?- ['ejemplo_gramatica.pl'].
?- deriva(oración,[el,gato,come,pescado]-[]).
Yes
?- findall(_O,deriva(oración,_O-[]),_L), length(_L,N).
N = 64
?- deriva(sintagma_nominal,SN-[]).
```

```
SN = [gato] ;
SN = [perro]
Yes
```

. . . . . .

Solución:

```
deriva(X,E0-E1) :-
    (X ---> Y),
    deriva(Y,E0-E1).
deriva((X,Y),E0-E2) :-
    deriva(X,E0-E1),
    deriva(Y,E1-E2).
deriva([X],[X|E]-E).
```

Segunda parte [3 puntos] En el laboratorio.
