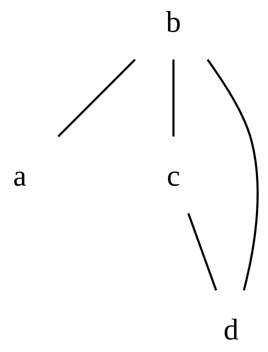
Examen de *Programación declarativa* del 23 de Enero de 2007

José A. Alonso Jiménez

Grupo de Lógica Computacional
Dpto. de Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
Sevilla, 22 de enero de 2007

Ejercicio 1 *El grado de un nodo de un grafo es el número de nodos adyacentes. Por ejemplo, en el siguiente grafo el grado de* `a` *es 1 y el de* `b` *es 3.*



Definir la relación `nodos_ordenados(+G,-L)` *que se verifique si* `L` *es la lista de los nodos del grafo* `G` *ordenados de menor a mayor grado. Por ejemplo,*

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

Solución:

```
nodos_ordenados(G,L) :-
   nodos(G,L1),
   findall(N-X,(member(X,L1), grado(X,G,N)), L2),
   sort(L2,L3),
   findall(X,member(N-X,L3),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 su definición es

```
nodos(G,L) :-
   setof(X,Y^adyacente(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 su definición es

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

La relación `grado(+X,+G,-N)` se verifica si `N` es el grado del nodo `X` en le grafo `G`. Por ejemplo,

```
?- grado(a,[a-b,b-c,b-d,c-d],N).
N = 1
?- grado(b,[a-b,b-c,b-d,c-d],N).
N = 3
```

Su definición es

```
grado(X,G,N) :-
   adyacentes(X,G,L),
   length(L,N).
```

La relación `adyacentes(+X,+G,-L)` se verifica si `L` es la lista de los nodos adyacentes al nodo `X` en el grafo `G`. Por ejemplo,

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

Su definición es

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

Ejercicio 2 *Los árboles de análisis sintáctico se representan por términos. Por ejemplo, el árbol de análisis*

```
               oración
                  |
          +-------+-------+
          |               |
   sintagma_nominal sintagma_verbal
          |               |
      +---+---+       +---+------+
      |       |       |          |
   artículo nombre  verbo  sintagma_nominal
      |       |       |          |
      el     gato   come       nombre
                                 |
                              pescado
```

se representa por el término `o(sn(art(el),n(gato)),sv(v(come),sn(n(pescado))))`. *Definir la relación* `categorías(+A,-L)` *que se verifique si* `L` *es el conjunto de las categorías gramaticales del árbol de análisis sintáctico* `A`. *Por ejemplo,*

```
?- categorías(o(sn(art(el),n(gato)),sv(v(come),sn(n(pescado)))),L).
L = [o, art, sv, v, sn, n]
?- categorías(e(+(dos,*(+(dos,-(tres))))), L).
L = [e, *, +, -]
```

Solución:

La definición de `categorías/2` es

```
categorías(A,[]) :-
   atomic(A), !.
categorías(A,L) :-
   A =.. [C|L1],
   categorías_lista(L1,L2),
   union([C],L2,L).
```

La relación `categorías_lista(+L1,-L2)` se verifica si `L2` es el conjunto de categorías gramaticales que aparecen en la lista de árboles de análisis `L1`. Por ejemplo,

```
?- categorías_lista([sn(art(el),n(gato)), sv(v(come),sn(n(pescado)))], L).
L = [art, sv, v, sn, n]
```

Su definición es

```
categorías_lista([],[]).
categorías_lista([A|L1],L2) :-
   categorías(A,L3),
   categorías_lista(L1,L4),
   union(L3,L4,L2).
```

Ejercicio 3 *Un rectángulo se divide en 4 rectángulos de base y altura entera. El problema consiste en determinar la superficie del subrectángulo superior derecho a partir de las superficies de los subrectángulos inferior izquierdo, inferior derecho y superior izquierdo.*

Definir la relación `rectángulo(+S1,+S2,+S3,-S4)` *que se verifique si* `S1` *es la superficie del subrectángulo inferior izquierdo,* `S2` *la del inferior derecho,* `S3` *la del superior izquierdo y* `S4` *la del superior derecho. Por ejemplo,*

```
?- rectángulo(6,10,21,S).
S = 35
?- rectángulo(10,6,35,S).
S = 21
?- rectángulo(6,10,35,S).
No
```

Dar dos definiciones, una usando programación lógica con restricciones y otra sin restricciones.

Solución:

Una imagen del problema es

```
    +----+----+
D   | S3 | S4 |
    +---+-----+
C   | S1 | S2 |
    +----+----+
      A    B
```

La definición de `rectángulo/4` usando restricciones es

```
:- use_module(library('clp/bounds')).

rectángulo_1(S1,S2,S3,S4) :-
   máximo(S1,S2,S3,S),
   [A,B,C,D] in 1..S,
   S1 #= A*C,
   S2 #= B*C,
   S3 #= A*D,
   label([A,B,C,D]),
   S4 is B*D.
```

La relación `máximo(+S1,+S2,+S3,-S)` se verifica si `S` es el máximo de `S1`, `S2`, `S3` y `S4`. Por ejemplo,

```
?- máximo(6,10,21,S).
S = 21
```

La definición de `máximo/4` es

```
máximo(S1,S2,S3,S) :-
   member(S,[S1,S2,S3]),
   not((member(T,[S1,S2,S3]), S<T)).
```

La definición de `rectángulo/4` sin usar restricciones es

```
rectángulo_2(S1,S2,S3,S4) :-
   máximo(S1,S2,S3,S),
   between(1,S,A),
   between(1,S,C),
   S1 =:= A*C,
   between(1,S,B),
   S2 =:= B*C,
   between(1,S,D),
   S3 =:= A*D,
   S4 is B*D.
```