Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI3661 - Taller de Lenguajes de Programación I
Enero-Marzo 2011

Quiz Prolog

(10 puntos)

Nombre:

Carnet:

Atención: en todas sus respuestas debe utilizar *exclusivamente* predicados disponibles en el estándar GNU Prolog, **exceptuando** los cortes (*cuts*) y el predicado de negación (`\+` o `not`). Si Ud. no sabe si un predicado está o no en el conjunto estándar de GNU Prolog, puede preguntarlo *públicamente* para recibir un **sí** o **no** como respuesta y **nada** más.

1. La técnica de compresión de datos *Run Length Encoding* consiste en reemplazar secuencias de valores idénticos por una tupla con el conteo total de elementos en la secuencia y el elemento repetido en la secuencia; entonces, el valor comprimido resultante es una lista de tuplas $< conteo, valor >$. Así, si tenemos una lista Prolog cualquiera y usamos listas de dos elementos para representar las tuplas, queremos comprimirla usando un predicado `encode/2` como sigue,

   ```
   ?- encode([a,a,a,b,c,c,a,b,b,d,d,d,d,d],C).
   C = [[3,a],[1,b],[2,c],[1,a],[2,b],[5,d]]
   ```

   y luego descomprimirla usando un predicado `decode/2` como sigue

   ```
   ?- decode([[3,a],[1,b],[2,c],[1,a],[2,b],[5,d]],D).
   D = [a,a,a,b,c,c,a,b,b,d,d,d,d,d]
   ```

   a) **(3 puntos)** Implante el predicado `encode/2`.

   ```
   encode([],[]).
   encode([Value|Rest],[[N,Value]|RLE]) :-
     implode(Value,Rest,N,Next),  % Extraer N Values continuos de Rest, quedando Next.
     encode(Next,RLE).            % Codificar Next.

   implode(E,List,N,Rest) :- implode(E,List,N,Rest,1).
   % implode/5 hace el trabajo con un acumulador
   implode(_,[],A,[],A).          % Fin de la lista.
   implode(E,[E|R],N,Rest,A) :-
     A1 is A+1,                   % Se mantiene la secuencia de Es, continuamos
     implode(E,R,N,Rest,A1).      % extrayendo y acumulando.
   implode(E,[F|R],A,[F|R],A) :-
     E \= F.                      % Fin de la secuencia.
   ```

   a) **(3 puntos)** Implante el predicado `decode/2`.

   ```
   decode([],[]).
   decode([[N,Value]|RLE],List) :-
     explode(N,Value,VS),             % Unificar VS con una lista de N Values.
     decode(RLE,Rest),                % Decodificar el resto, antes de
     append(VS,Rest,List).            % concatenar los valores al principio.


   % explode/3 hace el trabajo decrementando el conteo mientras agrega
   % elementos a la lista resultante.
   explode(0,_,[]).
   explode(N,X,[X|R]) :- N > 0, N1 is N-1, explode(N1,X,R).
   ```

2. **(4 puntos)** El predicado `rotate(L,N,R)` triunfa si `R` es la lista resultante de rotar `N` posiciones los elementos de la lista `L`. Si el valor de `N` es positivo, la rotación es hacia la derecha, mientras que si el valor de `N` es negativo, la rotación es hacia la izquierda, e.g.

```
rotate([1,2,3,4],1,L)        L = [4,1,2,3]
rotate([1,2,3,4],3,L)        L = [2,3,4,1]
rotate([1,2,3,4],5,L)             no
rotate([1,2,3,4],-1,L)       L = [2,3,4,1]
rotate([1,2,3,4],-3,L)       L = [4,1,2,3]
rotate(L,2,[3,4,1,2])        L = [1,2,3,4]
rotate(L,3,R)           L = [A,B,C] R = [A,B,C]
```

   Estos ejemplos han sido escritos empleando listas de números, pero el predicado debe operar correctamente sobre listas heterogéneas arbitrarias. Implante el predicado `rotate/3`. *Pista:* no hace falta calcular la longitud de las listas que son pasadas como argumento.

```
rotate(List,0,List).
rotate(List,N,Rotated) :-
  N > 0,
  length(Suffix,N),                % Lista vacía de longitud N.
  append(Prefix,Suffix,List),      % Suffix tiene los N últimos :-)
  append(Suffix,Prefix,Rotated).   % Mover Suffix al principio.
rotate(List,N,Rotated) :-
  N < 0,
  N1 is 0 - N,                     % Con valores positivos es más fácil.
  length(Prefix,N1),               % Lista vacía de longitud N.
  append(Prefix,Suffix,List),      % Prefix tiene los N primeros :-)
  append(Suffix,Prefix,Rotated).   % Mover Prefix al final.
```

3. **(3 puntos extra)** Representaremos cualquier alfabeto Σ como una lista de átomos Prolog. También usaremos listas para representar las palabras construidas a partir del alfabeto. Implante el predicado `sigma/2` capaz de enumerar el conjunto infinito de palabras Σ^* aprovechando el *backtracking*, asegurando que las palabras se generen en orden de longitud, esto es

```
?- sigma([a,b],W).
W = [] ; W = [a] ; W = [b] ; W = [a,a] ; W = [b,a] ; W = [a,b] ; W = [b,b] ; W = [a,a,a] ...
```

   Nota: esta pregunta solamente le otorgará puntos si **aprueba** el resto del examen.

```
sigma(_,[]).
sigma(Sigma,[Letter|Word]) :- sigma(Sigma,Word), member(Letter,Sigma).
```