Programación Declarativa: Lógica y restricciones

García Martín, Ignacio

12m071

Índice

Introducción

Enunciado

Los cuerpos de las serpientes están formados por anillos. En la víbora de Montegancedo estos anillos siguen un patrón de color que se repite a lo largo del cuerpo. Si se representan los colores por letras, el cuerpo de una víbora de Montegancedo con 15 anillos que siguen el patrón *abcd* tiene el aspecto de la secuencia *abcdabcdabcdabc*.

Para dormir, la víbora de Montegancedo recoge el cuerpo en forma de rectángulo: estira el cuerpo hacia la derecha hasta un punto en que se dobla y lo estira de nuevo hacia la izquierda, en paralelo a sí misma, hasta llegar a la cabeza, se vuelve a doblar sobre sí misma y así sucesivamente. La víbora anterior tendría el siguiente aspecto mientras duerme:

abcda
badcb
cdabc

La cabeza es siempre el primer anillo 'a', el cuerpo continua hacia la derecha hasta la siguiente 'a', se dobla en la 'b' que hay debajo y continua hacia la izquierda. Y así sucesivamente.

Programar un predicado **vibora/4** que se verifica cuando el cuarto argumento es el rectángulo que forma una víbora para dormir; dicho rectángulo se representa con una lista de listas, cada una de las cuales representa una fila del rectángulo con los colores de los anillos correspondientes. El primer argumento (en forma de lista) es el patrón de color de la víbora; dicha víbora se recoge para dormir formando filas con tantos anillos como elementos tiene la lista del segundo argumento y columnas con tantos anillos como elementos tiene la lista del tercer argumento.

Por ejemplo, para el caso de la víbora mencionada más arriba, tendríamos:

```
?- vibora([a,b,c,d],[_,_,_,_],[_,_,_],R).
R = [[a,b,c,d,a],[b,a,d,c,b],[c,d,a,b,c]] ?
```

Nótese que el patrón de una fila (segundo argumento) indica cuántas columnas tiene el rectángulo y el patrón de una columna (tercer argumento) indica el número de filas que tiene el rectángulo. Puesto que no se especifica la longitud de la víbora, se supone que es igual al producto de las longitudes del segundo y tercer argumentos, por lo que siempre se "rellena" todo el rectángulo.

El programa debe de ser puro, pero no se puede utilizar aritmética de Peano. Debe funcionar en cualquier modo de llamada siempre que segundo y tercer argumentos sean listas completas no vacías, es decir, con un número exacto de elementos distinto de cero. Los elementos de estas listas son indiferentes. Por ejemplo, las llamadas "admisibles" pueden tener en dichos argumentos listas como [_,_,_] o [a,a,a,a,a] o [0,X,a,p(b)] pero no [_|X] ni X si X es una variable libre. El programa puede no funcionar correctamente con llamadas "no admisibles" como las mencionadas, pero debe hacerlo en cualquier otro caso.

Código

Explicación

Para la realización de la práctica, después de unas cuantas lecturas del enunciado. Dividí en un primer momento la práctica en objetivos más pequeños de forma separada. El primero de ellos era comprender como rellenar/3 una lista con un patrón dado:

```
rellenar(_,[],[]).
rellenar([X|Xs],[_|Ys],[X|Ls]):-
    concatenar(Xs,[X],L1),
    rellenar(L1,Ys,Ls).
```

Siendo el primer argumento una lista con el patrón a seguir, el segundo argumento el número de elementos que contendrá la lista solución y el tercer argumento la lista solución.

Para que no nos quedemos sin la lista del patrón, vamos añadiendo el primer elemento al final de esta lista y la renombramos como L1. Para poder llamarla de forma recursiva. Hasta que los elementos de la segunda lista se agoten. Momento en el que la lista solución también estará vacía. Esto es nuestro caso base para rellenar.

Lo que estamos realizando con la lista patrón es un rotación del primer elemento de la lista al último elemento de la misma lista. A partir de ahora utilizamos una función rotar/2 para simplificar la lectura.

```
rotar([X|Xs],L):-concatenar(Xs,[X],L).
```

De forma automática, el siguiente paso era obtener una lista con la parte del patrón que no se utiliza en rellenar.

```
reres(_,[],_).
reres(X,Y,L):-
    concatenar(X,L,L1),
    rellenar(X,Y,L1).
```

La dinámica para este algoritmo es concatenar la lista patrón junto con la lista solución para obtener la lista que se obtiene al rellenar. El objetivo era poder obtener estos restos para poder pasárselos a una función que completara la siguiente fila con las columnas suficientes.

Una vez llegado hasta aquí, empezamos con un acercamiento a víbora juntando varias de estas funciones. La primera de ellas es vib/3 el primer argumento se refiere a la lista que contiene el patrón, el segundo argumento la lista se con las posiciones que tendrá la lista solución que es el tercer argumento.

```
vib(_,[],[]).
vib([P|Ps],[_|Cs],[P|Z]):-
    concatenar(Ps,[P],H),
    vib(H,Cs,Z).
```

La siguiente versión (vib2) introduce el número de filas a recorrer. Luego la lista solución devuelve la víbora en línea recta agrupada por las filas en las que se doblaría.

```
vib2(_,_,[],[]).
vib2([P|Ps],C,[_|Fs],[Z|T]):-
    vib([P|Ps],C,Z),
    concatenar(Ps,[P],H1),
    vib2(H1,C,Fs,T).
```

Tras varios intentos de intentar hacer una inversión dentro de una misma función utilizando la inversa, como se puede apreciar en vib4. Creamos la víbora en la que llamaba a dos funciones para realizar el objetivo, la primea de ellas es mec, similar a vib2 e invertir.

```
vib4(_,_,[],[]).
vib4([P|Ps],C,[_,_|Fs],[Z,Z1|T]):-
    duplica2(C,L),
    vib([P|Ps],C,Z),
    concatenar(Ps,[P],H1),
    vib(H1,C,Z11),
    inversa(Z11,Z1),
    concatenar(H1,Ps,H2),
    vib4(H2,L,Fs,T).
```

Funciones auxiliares

Para realizar funciones auxiliares, ha sido de gran ayuda las que se encontraban resueltas en las diapositivas. Como pueden ser :

```
concatenar([],L,L).
concatenar([X|Y],Z,[X|U]):- concatenar(Y,Z,U).

ultimo(X,[X]).
ultimo(X,[_|R]):- ultimo(X,R).

inversa([],[]).
inversa([X|R],L):-inversa(R,L1),
    concatenar(L1,[X],L).

duplica([],[]).
duplica([X|Xs],[X,X|L]):-duplica(Xs,L).
```

Empezamos con el predicado rellenarColumnas/3, tiene tres argumentos: lista con el patrón de colores, lista con las columnas y la lista solución, es decir, la lista compuesta por tantos colores como columnas tiene siguiendo el patrón. El funcionamiento es similar a vib/3

```
rellenarColumnas(_,[],[]).
rellenarColumnas([P|Ps],[_|Cs],[P|Z]):-
    concatenar(Ps,[P],H),
    rellenarColumnas(H,Cs,Z).
```

El siguiente predicado rotaciones/3, contiene tres argumentos: lista con el patrón de colores, lista con las columnas y lista con las rotaciones. Esta función llama de forma recursiva a rotar/2 y almacena lo que devuelve rotar en la cabeza de la lista de rotaciones

```
rotaciones(_,[],[]).
rotaciones(P,[_|Cs],[L|L1]):-
    rotar(P,L),
    rotaciones(L,Cs,L1).
```

El predicado invertir de aridad 2, el primer argumento es la secuencia de la víbora en línea recta, el segundo parámetro es la secuencia de la víbora enrollada.

```
invertir([],_).
invertir(A,A).
invertir([F,F1|Fs],[F,F2|[T]]):-
    inversa(F1,F2),
    invertir(Fs,T).
```

Cogemos los dos primeros elementos, que se corresponden con las dos primeras filas de la víbora(F,F1), obtenemos la inversa de la fila F1 y la hacemos corresponder con el segundo término de la lista solución. De tal forma que la lista solución tenga por primer término el mismo que teníamos y como segundo la lista invertida. Llamamos de forma recursiva al predicado con el resto de la primera lista y el de la segunda lista hasta llegar al caso de llegar a la primera lista vacía, es decir, no queden más tuplas de listas por corregir. O bien, al caso de que solo quede una fila, en cuyo caso es equivalente a la lista solución.

El predicado rellenarFilas de aridad 4, siendo el primer argumento la lista patrón, el segundo argumento la lista de columnas, el tercero la lista de filas y el cuarto la víbora “despierta”.

```
rellenarFilas(_,_,[],[]).
rellenarFilas([P|Ps],C,[_|Fs],[Z|T]):-
    rellenarColumnas([P|Ps],C,Z),
    rotaciones([P|Ps],C,X),
    ultimo(Y,X),
    rellenarFilas(Y,C,Fs,T).
```

Llamamos de forma recursiva a rellenarColumnas/4, rotaciones/3 y a ultimo. El funcionamiento es bastante intuitivo, vamos rellenando fila a fila llamando a rellenarColumnas. Una vez que llenamos una fila, nos encargamos de pasar el patrón con el que ha de empezar la siguiente fila mediante la combinación de rotaciones/3 y cogiendo la última rotación con ultimo/2. Todo este proceso hasta agotar el número de filas en las que la víbora se quedará enrollada.

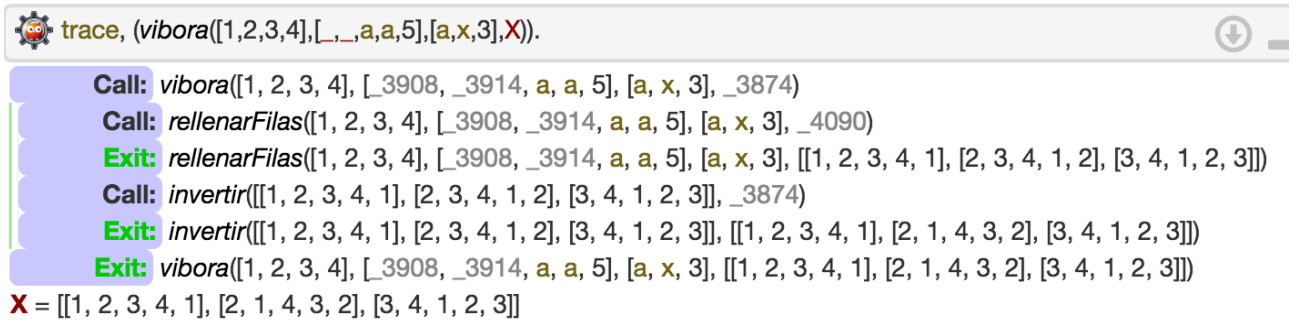
El predicado víbora de aridad 4,tiene los mismos cuatro argumentos que rellenarFilas/4. El funcionamiento es la llamada a rellenarFilas/4 para obtener la víbora segmentada en filas y después corrigiendo las filas impares para presentar la víbora “dormida”

```
vibora(P,C,F,R):-
    rellenarFilas(P,C,F,Rc),
    invertir(Rc,R).
```

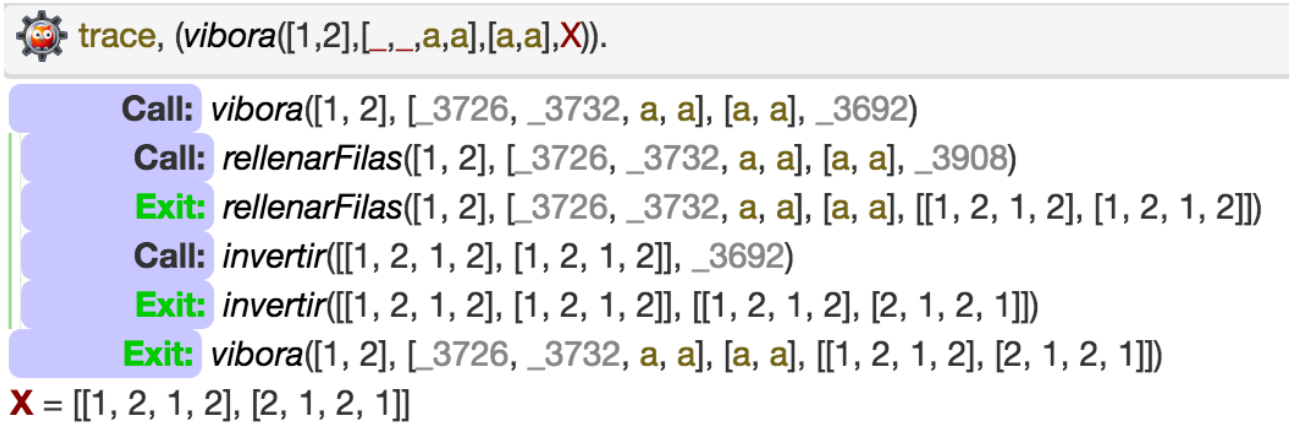
Pruebas de código

A continuación presento unas cuantas víboras, con la ayuda del debugger para mostrar como se va genenerando la solución y comprobar el correcto funcionamiento del código. Me permitiré el uso de números en vez de letras en estas pruebas para seguir de forma más fácil el patrón de color.

Patrón de cuatro colores, víbora de 15 anillos, agrupados en cinco columnas y tres filas:



Patrón de dos colores, víbora de 8 anillos, agrupados en cuatro columnas y dos filas:



Patrón de tres colores, víbora de 48 anillos, agrupados en ocho columnas y seis filas: