Labos LI: Práctica 5. Prolog-2 Otoño 2018

Problema(s) A:

A.1. "Hacer Aguas": disponemos de un grifo de agua, un cubo de 5 litros y otro de 8 litros. Se puede verter el contenido de un cubo en otro, llenar un cubo, o vaciar un cubo del todo, y queremos saber la secuencia mínima de operaciones para obtener exactamente 4 litros de agua en el cubo de 8 litros.

A.2. Dado un natural $n > 0$, una posición inicial $(Fila_I, Columna_I)$, una posición final $(Fila_F, Columna_F)$, y un número de pasos P, encontrar un camino de $(Fila_I, Columna_I)$ a $(Fila_F, Columna_F)$, en un tablero de ajedrez de $n \times n$ en exactamente P pasos de caballo. El programa ha de fallar si para la n en cuestión no existe tal camino.

Usa (es obligatorio) el siguiente esquema Prolog para resolver los dos problemas:

```
camino( E,E, C,C ).
camino( EstadoActual, EstadoFinal, CaminoHastaAhora, CaminoTotal ):-
  unPaso( EstadoActual, EstSiguiente ),
  \+member(EstSiguiente,CaminoHastaAhora),
  camino( EstSiguiente, EstadoFinal, [EstSiguiente|CaminoHastaAhora], CaminoTotal ).

solucionOptima:-
  nat(N),                          % Buscamos solución de "coste" 0; si no, de 1, etc.
  camino([0,0],[0,4],[[0,0]],C), % En "hacer aguas": -un estado es [cubo5,cubo8], y
  length(C,N),                     % -el coste es la longitud de C.
  write(C).
```

Problema B: Un programa escrito en el lenguaje *sumbol* tiene la siguiente sintaxis:

```
<programa>   -->    begin  <instrucciones>  end

<instrucciones>   -->    <instruccion>
<instrucciones>   -->    <instruccion> ; <instrucciones>

<instruccion>   -->    <variable> = <variable> + <variable>
<instruccion>   -->    if <variable> = <variable> then <instrucciones> else <instrucciones>  endif
<variable>-->    x
<variable>-->    y
<variable>-->    z
```

Tres ejemplos de programas *sumbol*:

```
  begin x=x+z end
  begin x=x+y; z=z+z; x=y+x end
  begin x=y+z; if z=z then x=x+z; y=y+z else z=z+y endif; x=x+z end
```

Se pide: escribir en Prolog un sencillo analizador sintáctico para el lenguaje *sumbol*, es decir, una cláusula programa(P) que se satisface si la lista de átomos P contiene un programa *sumbol* sintácticamente correcto, y que falla en caso contrario. Ejemplos:

```
?- programa( [begin, z, =, x, +, y, end] ).
yes

?- programa( [begin, z, =, x, +, y, ;, x, =, z, z, end] ).
no
```

(en el segundo ejemplo falta un -"). Para ello, hacer corresponder una cláusula a cada regla de la gramática.

Problema C: Mastermind es un juego donde un jugador (defensor) se inventa un código secreto y el otro jugador (atacante) tiene que averiguarlo. El código es una secuencia de 4 colores a escoger entre rojo, azul, verde, lila, naranja y marrón. El atacante tiene un número finito de intentos para romper el código. En cada intento, el atacante preguntará por una secuencia de 4 colores y el defensor responderá con dos números (E,D), siendo E el número de colores que el atacante ha acertado en la posición Exacta, y D el número de colores que ha acertado pero en una posición Distinta. Ejemplo:

```
El defensor escoge el código [r,n,n,a].

Intento 1: el defensor pregunta por [r,a,v,l]. La respuesta es [1,1].
El primer  1 es debido a la primera posición (r).
El segundo 1 es debido a la a, que no está en la posición correcta.
Intento 2: el defensor pregunta por [m,n,v,l]. La respuesta es [1,0].
Intento 3: el defensor pregunta por [v,l,v,l]. La respuesta es [0,0].
Intento 4: el defensor pregunta por [r,a,m,m]. La respuesta es [1,1].
Intento 5: el defensor pregunta por [r,n,a,n]. La respuesta es [2,2].
Intento 6: el defensor pregunta por [r,n,n,a]. La respuesta es [4,0]. El atacante gana el juego.
```

C.1. Construye un predicado respuesta(Codigo,Intento,E,D) que, dado un Codigo y un Intento calcule los los números [E,D] de la respuesta. Ejemplos:

```
?- respuesta([r,n,n,a],[m,n,v,l],E,D).
E = 1,
D = 0.

?- respuesta([r,n,n,a],[r,n,a,n],E,D).
E = D, D = 2.

?- respuesta([r,a,v,l],[r,n,a,n],E,D).
E = D, D = 1.
```

C.2. Queremos ahora ayudar al atacante a ganar el juego, sugiriéndole intentos. Asume que nos dan una cláusula de la forma:

```
intentos([ [ [r,a,v,l], [1,1] ], [ [m,n,v,l], [1,0] ], [ [v,l,v,l], [0,0] ],
           [ [r,a,m,m], [1,1] ], [ [r,n,a,n], [2,2] ]]).
```

que representa el histórico de los intentos hechos por el atacante. Construye un nuevo predicado nuevoIntento(A) que genera un nuevo intento A tal que, si A fuera el código a descubrir, entonces todos los intentos en el histórico hubieran tenido la misma respuesta. Ejemplo:

```
?- nuevoIntento(A).
A = [r, n, n, a]
```

Puedes probar algún Mastermind online y utilizar este predicado, posiblemente adaptando los colores, para ganar en un número razonable de pasos.