Práctica 1: Programación Lógica pura.

Vidal Peña, Arturo

5 de abril de 2019

Índice

1. Código empleado y las explicaciones:

1. Definición de la base de hechos de los colores de las piezas de LEGO™:

   Disponemos una base de hechos con los colores amarillo (am), verde (v), azul (a) y rojo (r) para hacer las comprobaciones de los colores de las piezas.

```
%Colores:
color(am).
color(v).
color(a).
color(r).
```

2. Definición de predicados auxiliares comunes a los especificados en la práctica:

   Definimos una serie de predicados auxiliares:

   a) nat/1: comprueba que el número pasado como argumento es un número natural. Partiendo de 0 (que definimos natural), un número X será natural si X-1 lo es.

   b) menor_igual/2, que será cierto si el primer argumento es menor o igual que el segundo. Se llama recursivamente con el número de Peano anterior a cada argumento hasta que el primero sea 0 y el segundo, mayor que 0.

   c) mayor_igual/2, su funcionamiento es igual que el anterior, sólo que comprueba si el primer argumento es mayor o igual que el segundo.

```
%0 es natural.
nat(0).
%Si X es natural, X+1 tambien.
nat(s(X)) :- nat(X).

menor_igual(0,X).
menor_igual(s(X),s(Y)) :- menor_igual(X,Y).

mayor_igual(s(X),0) :- nat(X).
mayor_igual(s(X),s(Y)) :- mayor_igual(X,Y).
```

d) suma/3, que será cierto cuando el tercer argumento sea la suma aritmética de los dos primeros.

e) resta/3, que será cierto cuando el tercer argumento sea la resta aritmética de los dos primeros.

f) iguales/2, será cierto cuando ambos argumentos pasados como números de Peano tengan el mismo valor. Se llama recursivamente comprobando que lleguen a 0 al mismo tiempo.

g) par/1: será cierto si, restando 2 al número pasado por argumento recursivamente, se llega a 0.

h) impar/1: igual que el anterior, sólo que comprobando que llega a 1.

```
suma(0,X,X). %X = X+0.
suma(s(X),Y,s(Z)) :- suma(X,Y,Z). %Z+1 = X+1+Y

resta(X,0,X).
resta(s(X),s(Y),Z) :- resta(X,Y,Z). %Z=X-Y

iguales(s(0),s(0)).
iguales(s(A),s(B)) :- iguales(A,B).

par(0).
par(s(s(X))) :- par(X).

impar(s(0)).
impar(s(s(X))) :- impar(X).
```

i) esPieza/4: comprueba si los argumentos *Altura*, *Anchura* y *Profundidad* pasados como números de Peano son naturales, y si *Color* está en la base de hechos.

j) p/3: siendo pasada una lista como primer argumento, será cierto si el segundo argumento es la cabeza y el tercero, la cola de la lista.

k) member/2: comprueba si el primer argumento se encuentra en la lista que se pasa en el segundo argumento.

```
esPieza(Anchura,Altura,Profundidad,Color) :-
        nat(Anchura),
        nat(Altura),
        nat(Profundidad),
        color(Color).

p([H|T],H,T).

member(X,[X|_]).

member(X,[_|T]) :- member(X,T).
```

3. Definición del predicado esTorre/1:

   El caso base se afirma si la torre se compone de una única pieza, llamando a esPieza/1.

   En el caso de tener más de una pieza, comprueba recursivamente que las piezas de la torre son válidas con esPieza/1 y que la pieza en cabeza es más pequeña en *Anchura* y *Profundidad* que la siguiente en la torre, llegando al caso base con la última pieza.

```
esTorre([pieza(Anchura,Altura,Profundidad,Color)|[]]) :-
        esPieza(Anchura,Altura,Profundidad,Color).

esTorre([pieza(Anchura1,Altura1,Profundidad1,Color1),pieza(Anchura2,Altura2,Profundidad2,Color2)|Ps]) :-
        esPieza(Anchura1,Altura1,Profundidad1,Color1),
        esPieza(Anchura2,Altura2,Profundidad2,Color2),
        menor_igual(Anchura1,Anchura2),
        menor_igual(Profundidad1,Profundidad2),
        esTorre([pieza(Anchura2,Altura2,Profundidad2,Color2)|Ps]).
```

4. Definición del predicado alturaTorre/2:

   El caso base se afirma si la torre de una única pieza es válida (esTorre/1), si el segundo parámetro en natural, y si este segundo parámetro vale 0 al ser restado por la altura de la pieza que conforma la torre.

   En el caso de haber más de una pieza, comprueba que la lista es una torre válida, comprueba que el segundo parámetro es un natural, y llama al predicado auxiliar exclusivo de alturaTorre/2, pasando los mismos argumentos. Este predicado sacarAltura/2 va restando recursivamente al segundo parámetro las alturas de las piezas que conforman la torre, llegando al caso base con la última pieza.

```
alturaTorre([pieza(Anchura,Altura,Profundidad,Color)|[]],X) :-
        esTorre([pieza(Anchura,Altura,Profundidad,Color)|[]]),
        nat(X),
        resta(X,Altura,Y),
        iguales(s(Y),s(0)).

alturaTorre(Torre,X) :-
        esTorre(Torre),
        nat(X),
        sacarAltura(Torre,X).

sacarAltura([pieza(Anchura[pieza(Anchura1,Altura1,Profundidad1,Color1),pieza(Anchura2,Altura2,Profundidad2,Color2)|Ps],X) :-
        resta(X,Altura1,Y),
        alturaTorre([pieza(Anchura2,Altura2,Profundidad2,Color2)|Ps],Y).
```

5. Definición del predicado coloresTorre/2:

   El caso base se afirma si la torre de una única pieza es válida (esTorre/1), y si el color de la pieza pertenece a la lista de colores pasada en el segundo parámetro.

   En el caso de haber más de una pieza, comprueba si la lista es una torre válida, si el segundo parámetro es un número natural, y luego llama al predicado auxiliar exclusivo de coloresTorre/2, pasando los mismos argumentos. Este predicado sacarColores/2 comprueba recursivamente que los colores de las piezas de la torre pertenecen a la lista de colores del segundo parámetro, llegando al caso base con la última pieza.

```
coloresTorre([pieza(Anchura,Altura,Profundidad,Color)|[]],[Colores|[]]) :-
        esTorre([pieza(Anchura,Altura,Profundidad,Color)|[]]),
        member(Color,[Colores]).

coloresTorre(Torre,Colores) :-
        esTorre(Torre),
        sacarColores(Torre, Colores).

sacarColores([pieza(Anchura1,Altura1,Profundidad1,Color1),pieza(Anchura2,Altura2,Profundidad2,Color2)|Ps],[Colores1,Colores2|Cs]) :-
        member(Color1,[Colores1,Colores2|Cs]),
        coloresTorre([pieza(Anchura2,Altura2,Profundidad2,Color2)|Ps],[Colores2|Cs]).
```

6. Definición del predicado coloresIncluidos/2, junto a sus auxiliares:

   Primero comprueba si ambas torres pasadas como parámetros son válidas, y luego llama al predicado auxiliar exclusivo de coloresIncluidos, pasando la primera torre en el primer parámetro y la segunda, en el segundo y tercero.

   Este predicado (comprobarColores/3), compara recursivamente el primer color de Torre1 con los colores de Torre2. Cuando acaba, copia la Torre2 y vuelve a empezar con el siguiente color de Torre1.

```
coloresIncluidos(Torre1,Torre2) :-
    esTorre(Torre1),
    esTorre(Torre2),
    comprobarColores(Torre1,Torre2,Torre2).

comprobarColores([],_,_).

comprobarColores([pieza(_,_,_,Color)|Torre1],[pieza(_,_,_,Color)|_],Torre2):-
        comprobarColores(Torre1,Torre2,Torre2).

comprobarColores(Torre1,[_|Torre2],T):-
        comprobarColores(Torre1,Torre2,T).
```

7. Definición del predicado esEdificioPar/1, junto a sus auxiliares:

   El caso base de esEdificioPar/1 es si la construcción sólo tiene una línea, en cuyo caso será cierto si esa línea es par, llamando al predicado lineaPar/1. En el caso de haber más de una línea, comprobará con lineaPar/1 si todas las líneas de la construcción son pares.

   Este predicado auxiliar llama a longitudLinea/2, pasando la línea y una variable en la que unificar la longitud de esta. Este predicado llama a su vez a longitudSinBlanco/3, pasando la línea, inicializando N1 a 0 y la variable a unificar en N2.

En longitudSinBlanco/1, si la cabeza de Linea es un blanco, se llama recursivamente con la cola de Linea como primer argumento. En otro caso, comprueba que la cabeza es un color válido, y se llama recursivamente con la cola como primer argumento, y el siguiente a N1 como segundo, siendo N2 siempre el tercero. Al acabar, cuando la línea está vacía, iguala N2 a N1.

Al unificar N2 con el segundo argumento de longitudLinea/2, comprueba que sea un número par con par/1. Si es cierto, la línea es par.

```
esEdificioPar([Linea|[]]) :-
        lineaPar(Linea).

esEdificioPar([Linea1|RestoLineas]) :-
        lineaPar(Linea1),
        esEdificioPar(RestoLineas).

lineaPar([]).

lineaPar(Linea) :-
        longitudLinea(Linea,Longitud),
        par(Longitud).

longitudLinea(Linea,N) :-
        longitudSinBlanco(Linea,0,N).

longitudSinBlanco([],N1,N2) :-
        N2 = N1.

longitudSinBlanco([b|Resto],N1,N2) :-
        longitudSinBlanco(Resto,N1,N2).

longitudSinBlanco([C|Resto],N1,N2) :-
        color(C),
        longitudSinBlanco(Resto,s(N1),N2).
```

2. Pruebas realizadas:

Todas las pruebas a continuación han dado los valores esperados.

1. nat/1:
   - nat(0),
   - nat(s(0)),
   - nat(s(s(s(0)))),
   - nat(t)
2. menor_igual/2:
   - menor_igual(s(0),s(0)),
   - menor_igual(s(s(0)),s(s(0))),
   - menor_igual(s(0),s(s(0)))
3. resta/2:
   - resta(s(0),0,s(0)),
   - resta(s(0),s(0),0),
   - resta(s(s(s(s(0)))),s(s(s(s(0)))),0)
4. par/1:
   - par(0)
   - par(s(s(0))),
   - par(s(s(s(s(0)))))
5. iguales/2:
   - iguales(s(0),s(0)),
   - iguales(s(0),s(0)),
   - iguales(s(s(s(s(0)))),s(s(s(s(0))))),
6. color/1:
   - color(r),
   - color(v),
   - color(am),
   - color(a),
   - color(b)
7. esTorre/1:
   - esTorre([pieza(s(0),s(s(0)),s(0),r)]),

- esTorre([pieza(s(0),s(s(0)),s(0),r),pieza(s(0),s(0),s(0),a)]),
- esTorre([pieza(s(s(s(0))),s(s(s(0))),s(s(s(0))),am),pieza(s(s(s(0))),s(s(s(0))),s(s(s(0))),v)])

8. alturaTorre/2:
   - alturaTorre([pieza(s(0),s(0),s(0),r)],s(0)),
   - alturaTorre([pieza(s(0),s(0),s(0),r),pieza(s(s(0)),s(s(0)),s(0),a)],s(s(s(0)))),
   - alturaTorre([pieza(s(0),s(0),s(0),r),pieza(s(0),s(0),s(0),r),pieza(s(0),s(s(0)),s(0),r)],s(s(s(s(0)))))
9. coloresTorre/2:
   - coloresTorre([pieza(s(0),s(0),s(0),r)],[r]),
   - coloresTorre([pieza(s(0),s(0),s(0),r),pieza(s(0),s(s(0)),s(0),a)],[r,a]),
   - coloresTorre([pieza(s(0),s(s(0)),s(0),am),pieza(s(0),s(0),s(0),r),pieza(s(0),s(s(0)),s(0),a)],[am,r,a])
10. coloresIncluidos/2:
    - coloresIncluidos([pieza(s(0),s(0),s(0),r)],[pieza(s(0),s(0),s(0),r)]),
    - coloresIncluidos([pieza(s(0),s(0),s(0),a),pieza(s(0),s(0),s(0),a)],[pieza(s(s(0)),s(0),s(s(0)),a)]),
    - coloresIncluidos([pieza(s(0),s(0),s(0),r),pieza(s(0),s(0),s(0),v),pieza(s(0),s(0),s(0),a)])
11. esEdificioPar/2:
    - esEdificioPar([[a,a],[v,v],[a,a],[v,v]]),
    - esEdificioPar([[a,a,r,v,am,r],[v,r,v,v,am,r],[v,r,v,r,a,r],[v,r,a,r,v,r],[v,r,a,r,v,am]]),
    - esEdificioPar([[a,a,am,r],[v,v,am,r],[v,r,a,r],[v,r,a,r],[v,r,a,am]])