Coloreado de mapas

El problema del coloreado de mapas es un problema matemático clásico del siglo XIX, muy relacionado con la teoría de grafos.

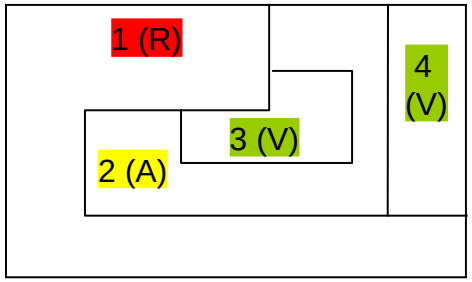
Dados un plano dividido en regiones y una paleta de colores, el problema consiste en asignar un color a cada una de las regiones del plano, de tal forma que dos regiones adyacentes no tengan asignado el mismo color.

Aunque para mapas “simples” tres colores son suficientes, el conocido como *Teorema de los cuatro colores* prueba que cuatro colores son necesarios (y suficientes) para resolver el problema en el caso general (la prueba es controvertida debido a que está basada en un programa de ordenador).

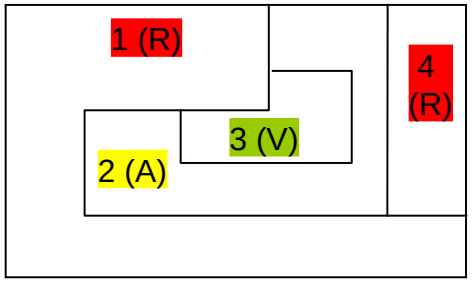
Coloreado de mapas

EJEMPLO

Dados los colores rojo (R), verde (V) y amarillo (A), y el siguiente plano dividido en cuatro regiones (numeradas del 1 al 4), la figura (a) representa una solución correcta al problema y la figura (b) una solución incorrecta (puesto que las regiones 1 y 4, que son adyacentes, están ambas coloreadas en rojo).



(a)



(b)

Coloreado de mapas

- Se desea resolver el problema del coloreado de mapas en Prolog, para lo cual se diseña un programa con dos partes diferenciadas:
  - La parte de *datos* en la que se incluirán los predicados necesarios para la descripción de un problema dado: qué regiones hay, cuáles son adyacentes (tienen frontera) y qué colores hay disponibles.
  - La parte de *conocimiento* en la que se resuelve el problema del coloreado de mapas mediante el predicado principal **solucion/1**, con un único parámetro de salida representando la asignación de colores a las distintas regiones mediante una lista de pares "(*X*,*Y*)", donde *X* es una región del plano e *Y* es su color.

Nota: se incluyen todos los predicados juntos en un único fichero para simplificar, pero lo adecuado sería tener los datos en un fichero aparte y añadir al predicado **solucion** un parámetro adicional, de entrada, que recibiría la ruta del fichero de datos, de forma que lo primero que haría el predicado **solucion/2** sería cargar ese fichero en Prolog (mediante el predicado predefinido consult/1).

Coloreado de mapas

- Por ejemplo, la siguiente consulta permitiría obtener todas las soluciones al problema presentado previamente como ejemplo:

```
?- solucion(L).

L = [ (1, verde), (2, azul), (3, rojo), (4, rojo)] ;
L = [ (1, azul), (2, verde), (3, rojo), (4, rojo)] ;
L = [ (1, verde), (2, rojo), (3, azul), (4, azul)] ;
L = [ (1, rojo), (2, verde), (3, azul), (4, azul)] ;
L = [ (1, azul), (2, rojo), (3, verde), (4, verde)] ;
L = [ (1, rojo), (2, azul), (3, verde), (4, verde)]
```

Coloreado de mapas

- Los datos del problema se representan mediante los predicados `regiones/1`, `colores/1` y `frontera/2`

```
% regiones(?X)
% X es una lista incluyendo las regiones del plano
% existentes
...

% colores(?X)
% X es una lista incluyendo los distintos colores
% disponibles
...

% frontera(?X,?Y)
% cierto si X e Y son dos regiones del plano fronterizas
...
```

Coloreado de mapas

- De esta manera, los datos correspondientes al problema descrito anteriormente serían los siguientes:

```
% regiones del plano
regiones([1,2,3,4]).

% colores disponibles
colores([rojo,azul,verde]).

% relaciones fronterizas
frontera(1,2).
frontera(1,3).
frontera(1,4).
frontera(2,3).
frontera(2,4).

% note que el predicado frontera se usa sin necesidad de hacer explícita
su simetría, de forma que para indicar que "a" y "b" son fronterizos basta
con el hecho "frontera(a,b)" o bien el hecho "frontera(b,a)"
(indistintamente). Esto tendrá que tenerse en cuenta a la hora de
comprobar fronteras (ver predicado "asignacion" más adelante).
```

Coloreado de mapas

- El predicado solucion/1 que resuelve el problema del coloreado de mapas es el siguiente:

```
% solucion(-Solucion)
% cierto si Solucion es una lista de pares (Region,Color)
% asignando colores a regiones

solucion(Solucion):-

        % Accedemos a las listas de regiones y colores disponibles
        regiones(Regiones),
        colores(Colores),

        % hallamos la asignación de colores a regiones
        asignacion(Regiones,Colores,Solucion).
```

Coloreado de mapas

```
% asignacion(+LR,+LC,?LA)
% cierto si LA es una lista de asignaciones (Region,Color)
% para la lista de regiones LR siendo LC la lista de colores
% disponibles

asignacion([],_,[]).
   % Si no hay ninguna región a la que asignarle color, entonces
   % la lista de asignaciones es vacía

asignacion([Region1|RestoRegiones],
           Colores,
           [(Region1,Color)|RestoAsignados]):-
   % determinamos las asignaciones para el resto de regiones
   asignacion(RestoRegiones,Colores,RestoAsignados),
   % elegimos un color para colorear la región actual (Region1)
   member(Color,Colores),
   % verificamos si ese color no ha sido ya utilizado para colorear
   % alguna región fronteriza
   \+ ((frontera(Region1,Region2); frontera(Region2,Region1)),
        member((Region2,Color),RestoAsignados)).
```

Coloreado de mapas

Recuerde que:

- El predicado predefinido `;/2` se usa en notación infija y representa la *disyunción* de sus dos argumentos, es decir, el objetivo "`P1 ; P2`" será cierto si y sólo si o bien `P1` o bien `P2` (o ambos) son ciertos.
- El predicado predefinido `,/2` se usa en notación infija y representa la *conjunción* de sus dos argumentos, es decir, el objetivo "`P1 , P2`" será cierto si y sólo si tanto `P1` como `P2` son ciertos.
- El predicado predefinido `member(?E, ?L)`, cierto si `E` pertenece a la lista `L`, es equivalente al predicado `pertenece/2` implementado en el tema PL-2.
- El predicado predefinido `\+ /1` se usa en notación prefija e implementa la negación por fallo finito (`\+ P` será cierto si y sólo si `P` falla) estudiada en el tema PL-3.

Coloreado de mapas

- Observaciones

Por lo tanto, la línea

```
\+ ((frontera(Region1,Region2); frontera(Region2,Region1)),
       member((Region2,Color),RestoAsignados)).
```

será cierta si no existe ninguna región, `Region2,` que haga frontera con `Region1` y tal que el par `(Region2,Color)` ya esté en la lista de pares asignados previamente (la disyunción ; con el predicado `frontera` se debe a que la relación `frontera` no es necesariamente simétrica, es decir, `frontera(r1,r2)` indica que las dos regiones hacen frontera, sin necesidad de añadir el hecho `frontera(r2,r1)`).

Ejercicio

- Añada al programa anterior el predicado `soluciones/0, que` escribe cuántas soluciones tiene el problema y las imprime, una por línea.

- Utilice para ello los predicados sobre listas descritos en los apuntes y en las hojas de problemas o sus equivalentes en SWI-Prolog (length, member, maplist, …).

Ejercicio

```
?- soluciones.

El problema tiene 6 soluciones, que son:

[ (1,verde), (2,azul), (3,rojo), (4,rojo)]
[ (1,azul), (2,verde), (3,rojo), (4,rojo)]
[ (1,verde), (2,rojo), (3,azul), (4,azul)]
[ (1,rojo), (2,verde), (3,azul), (4,azul)]
[ (1,azul), (2,rojo), (3,verde), (4,verde)]
[ (1,rojo), (2,azul), (3,verde), (4,verde)]
```

La siguiente diapositiva incluye la implementación de este predicado.

Ejercicio (solución)

```
soluciones :-
  findall(Asignacion, solucion(Asignacion), Soluciones),
  length(Soluciones, Cuantas),
          % equivalente a longitud/2, tema PL-2
  write('El problema tiene '),
  write(Cuantas),
  write(' soluciones, que son:'),
  nl,      % salto de línea
  nl,
  maplist(escribe, Soluciones).
         % equivalente a map/2, tema PL-3

escribe(A) :-
          write(A),
          nl.
```