Lógica para Computación

Año:2024

PRÁCTICO DE MÁQUINA DE PROLOG

Fecha de entrega: 07/06/24

Dado el siguiente programa Prolog que define grafos y algunas de sus propiedades:

```
grafo(g1,[a,b,c,d],[arista(a,b),arista(b,c),arista(b,d),arista(c,d)]).
grafo(g2,[a,b,c,d],[arista(a,b),arista(c,d)]).
grafo(g3,[a,b,c,d],[arista(a,b),arista(b,c),arista(b,d)]).
grafo(g4,[a,b,c,d],[arista(a,b),arista(b,c)]).
adyacente(G,X,Y) :- grafo(G,_,L),
                (member(arista(X,Y),L); member(arista(Y,X),L)).
cadena(G,A,Z,C) :-cadena_aux(G,A,[Z],C).
cadena_aux(_,A,[A|C1],[A|C1]).
cadena_aux(G,A,[Y|C1],C) :-adyacente(G,X,Y),
                           not(member(X,[Y|C1])),
                           cadena_aux(G,A,[X,Y|C1],C).
```

Donde **grafo(G,V,L),** se verifica si G es el nombre de un grafo, V es la lista de vértices del grafo, y L es la lista de aristas del grafo representados con el objeto arista(X,Y).

adyacente(G,X,Y) se verifica si los vértices X e Y son adyacentes, es decir, están unidos por una arista en G.

cadena(G,A,Z,C) se verifica si C es una cadena elemental (es decir, no repite vértices) que va del vértice A al vértice Z.

cadena_aux(G,A,CP,C) se verifica si C es una cadena en G compuesta por la concatenación de una cadena desde A hasta el primer elemento de la cadena parcial CP (con vértices distintos a los de CP) seguida de la cadena CP.

Se pide definir las siguientes relaciones y propiedades en Prolog. Para ello puede utilizar la relación **member(X,L)** definida en Prolog (se verifica si X pertenece a la lista L), las cláusulas presentes en el programa definido anteriormente y/o cualquier cláusula que considere necesaria definida por usted:

1. **vertice(G,X)**: se verifica cuando X es un vértice del grafo G.

```
?-vertice(g1,a).
yes

?-vertice(g1,f).
no
```


```
?-vertice(g1,X).
X=a;
X=b;
X=c;
X=d;
No
```

2. **conexo(G)**: se verifica si existe una cadena para todo par de vértices de G.

```
?- conexo(g1).
true.

?- conexo(g2).
false.

?- conexo(g3).
true.
```

3. **tiene_ciclos(G)**: se verifica si G tiene al menos un ciclo, es decir, una cadena cerrada (tiene el mismo vértice como origen y destino). Ej: g1 tiene un ciclo formado por los vértices b, c y d.

```
?- tiene_ciclos(g1).
true.

?- tiene_ciclos(g2).
false.

?- tiene_ciclos(g3).
false.
```

4. **es_arbol(G**): se verifica si G es un árbol, es decir, es un grafo conexo y sin ciclos.

```
?- es_arbol(g1).
false.

?- es_arbol(g2).
false.

?- es_arbol(g3).
true.
```

5. **grado(G,X,R)**: Es la cantidad de aristas que tienen a X como uno de sus vértices.

```
?-grado(g1,b,R).
R=3

?-grado(g1,a,R).
R=1
```