Programación Lógica y Funcional
Presentación del Curso

Angel García Báez
ZS24019400@estudiantes.uv.mx

Maestría en Inteligencia Artificial

IIIA Instituto de Investigaciones en Inteligencia Artificial
Universidad Veracruzana
Campus Sur, Calle Paseo Lote II, Sección 2a, No 112
Nuevo Xalapa, Xalapa, Ver., México 91097

3 de octubre de 2024

1. Cuestionario de Socrative.

Regístrese como estudiante en `https://www.socrative.com` para unirse a la sala (room) 2024PLF. Resuelva el cuestionario que ahí se presenta. Las preguntas pueden tener varias respuestas (30 puntos).

El cuestionario fue completado exitosamente en la plataforma de socrative bajo el nombre de Angel García Báez.

2. Extensión del programa de la familia.

Extienda el programa de la familia en Prolog para incluir las relaciones tío/2 y tía/2. Pruébelas con las metas:

1. **tio(X,Y).**
2. **tia(ann,Y).**

Defina una meta para computar quienes son sobrinos en esa familia (10 puntos).

Para llevar a cabo esta actividad, se utilizó el código de la familia facilitado por el Dr. Alejandro Guerra y se hizo una breve modificación en la definición de hermana, donde la clausula **mujer(Y)** fue sustituida por **mujer(X)**.

Posteriormente, para definir la relación tio(X,Y) el razonamiento fue encontrar al progenitor Z de Y y además si X es hermano del progenitor Z.

La misma lógica se aplico a tia(X,Y), donde el razonamiento fue encontrar al progenitor Z de Y y además si X es hermana del progenitor Z.

Bajo estas definiciones que se muestran en el código más adelante, al computar la meta tio(X,Y). se obtiene un false de regreso, porque para esta familia no hay nadie que sea tío de nadie. Por otro lado, para la meta tia(ann,Y) , al computar dicha meta se obtuvo que ann es tía de jim y nada más.

Finalmente, bajo estas nuevas clausulas que agregue al programa de la familia, para poder computar quienes son sobrinos de dicha familia se propuso computar:

```
1 tia(_, Y).
```

dado que las metas anteriores revelaron que únicamente hay tías en esta familia, nadie cumple la condición de tío, aunque si se quisiera ser más amplio, se puede computar la siguiente meta para incluir a los tíos en caso de que alguien cumpliera la condición:

```
1 tia(_,Y), tio(_,Y).
```

Los resultados se observan en la documentación del código a continuación:

```
%%% Universidad Veracruzana
%%% Instituto de Investigaciones en Inteligencia Artificial
%%% Maestria en Inteligencia Artificial
%%% Alejandro Guerra-Hernandez
% progenitor(X,Y).
% Verdadero si X es progenitor de Y.
% Estos predicados estan adaptados de Bratko(2012), cap1.
progenitor(pam,bob).
progenitor(tom,bob).
progenitor(tom,liz).
%progenitor(liz,rog). YO agregue un caso para probar tio, rog hijo de liz
progenitor(bob,ann).
progenitor(bob,pat).
progenitor(pat,jim).


%%%% mujer(X). %%%%
% Verdadero si X es mujer.
mujer(pam).
mujer(liz).
mujer(pat).
mujer(ann).

%%%% hombre(X). %%%%
% Verdadero si X es hombre.
hombre(tom).
hombre(bob).
hombre(jim).
%hombre(rog). rog es hombre

%%%%  vastago(X,Y). %%%%
% Verdadero si X es vastago de la persona Y.
%
% ?- vastago(bob,X).
% X = pam ;
% X = tom.
```

```
37 vastago(X,Y) :- progenitor(Y,X).
38
39 %%%% madre(X,Y). %%%%
40 % Verdadero si X es madre de Y.
41 %
42 %% ?- madre(X,bob).
43 %% X = pam ;
44 %% false.
45 madre(X,Y) :-
46     progenitor(X,Y),
47     mujer(X).
48
49 %%%% abuela(X,Y). %%%%
50 % Verdadero si X es abuela de Y.
51 %% ?- abuela(X, ann).
52 %% X = pam ;
53 %% false.
54 abuela(X,Y) :-
55     progenitor(X,Z),
56     progenitor(Z,Y),
57     mujer(X).
58
59 %%%% hermana(X,Y). %%%%
60 % Verdadero si X es hermana de Y.
61 %% ?- hermana(ann,X).
62 %% X = pat.
63 hermana(X,Y) :-
64     progenitor(Z,X),
65     progenitor(Z,Y),
66     mujer(X), % Se modifico la condicion de mujer(y)
          a mujer(x)
67     dif(X,Y).
68
69 %%%% ancestro(X,Z). %%%%
70 % Verdadero si X es ancestro/a de Z.
71 %
72 %% ?- ancestro(pam,X).
73 %% X = bob ;
74 %% X = ann ;
75 %% X = pat ;
```

```
%% X = jim ;
%% false.
ancestro(X,Z) :-
    progenitor(X,Z).
ancestro(X,Z) :-
    progenitor(X,Y),
    ancestro(Y,Z).

%%%% APORTACIONES DE ANGEL AL PROGRAMA %%%%
%%%% AGREGAR AL HERMANO %%%%
hermano(X,Y) :-
    progenitor(Z,X),
    progenitor(Z,Y),
    hombre(X), % Se modifico la condicion de que sea
        hombre(X)
    dif(X,Y). % que no sea el mismo, no puede ser
       hermano de el mismo.
%%%% AGREGAR A LA TIA %%%%
tia(X,Y):- %% X es tia de Y
    progenitor(Z,Y), % Encontrar el progenitor Z de
       Y
    hermana(X,Z). % Comprobar si X es hermana del
       progenitor Z
% tia(ann,Y). Computar la siguiente meta da como
   resultado:
% Ann es tia de Jim y nada mas
% tia(X,Y). Computar la siguiente meta da como
   resultado:
% Liz es tia de Ann
% Liz es tia de Pat
% Ann es tia de Jim y nada mas
%%%% AGREGAR AL LA TiO %%%%
tio(X,Y):- %% X es tio de Y
    progenitor(Z,Y), % Encontrar el progenitor Z de
       Y
    hermano(X,Z). % Comprobar si X es hermano del
       progenitor Z
% tio(X,Y). Computar la siguiente meta da como
   resultado:
%false
```

```
107 % Cabe mencionar que en la familia de este ejemplo
      no hay tios, solo tias,
108 % Por ello es que prolog da false, porque no hay
      ninguna clausula en el programa
109 % que satisfaga el requisito para que un miembro se
      considere tio de otro.
110 %%%% Defina una meta para computar quienes son
      sobrinos en esa familia %%%%
111 % Asumiendo que se sabe que no hay tios, bastaria
      con ejecutar:
112 % tia(_,X). dicha meta devuelve:
113 % X = ann; X = pat; X = jim, false.
114 % Suponiendo que hay tias y tios, se puede computar
      la siguiente meta:
115 % tia(_,X);tia(_,X). dicha meta devuelve:
116 % X = ann; X = pat; X = jim, false.
```

3. Ejercicio de unificación.

Aplique el algoritmo de unificación visto en clase a los términos $f(a, g(Z), Z)$ **y** $f(X, g(X), b)$ **(10 puntos).**

Para aplicar el algoritmo de la unificación sobre dichos terminos, se tomaron de bases las laminas de Guerra-Hernández (2024) como apoyo para resolver paso a paso la unificación de la siguiente manera:

$$f\{X, g(X), b\} = f\{a, g(Z), Z\}$$

A continuación se expresa paso a paso la forma en que se lleva a cabo la unificación:

$$\{X = a, g(X) = g(Z), b = Z\}$$
$$\{X = a, g(a) = g(Z), b = Z\}$$
$$\{X = a, a = Z, b = Z\}$$
$$\{X = a, Z = a, b = Z\}$$
$$\{X = a, Z = a, b = a\}$$
fallo

Dicha unificación no puede ser concretada puesto que, al momento de unificar los términos, se llega a una contradicción en donde hay 2 constantes diferentes (a y b) pero el desglose del proceso termina llevando a que b = a, lo cual en sí mismo no puede ser dado que b es una constante distinta de a, por lo que la unificación falla y no se puede concretar para estos términos.

Referencias

Guerra-Hernández, D. A. (2024). Programación Lógica y Funcional - Programación Lógica.