Universidad Simón Bolívar

Departamento de Computación y Tecnología de la Información

CI3661 – Laboratorio de Lenguajes de Programación

Trimestre: Septiembre - Diciembre 2023

Profesor: Ricardo Monascal

Estudiante: Junior Miguel Lara Torres, Carnet: 17-10303

Tarea 3: Programación Lógica y Prolog (15 pts)

IMPLEMENTACION

- Parte 1: (5 puntos) Considere la siguiente definición de los números de Church.

```
a) Predicado <suma> de los numeros de Church.

suma(*, X, X).
suma(up(X), Y, up(Z)) :- suma(X, Y, Z).

-------------------------------------------------

b) Predicado <resta> de los numeros de Church.

resta(X, *, X) :- !.
resta(up(X), up(Y), Z) :- resta(X, Y, Z).

-------------------------------------------------

c) Predicado <producto> de los numeros de Church.

producto(*, _, *).
producto(up(X), Y, Z) :-
    producto(X, Y, W),
    suma(W, Y, Z).
```

- Parte 2: (5 puntos) Considere hechos de la forma arco(A, B) en ProLog, representando una conexión dirigida desde un nodo A hasta un nodo B.

```
a) Predicado < hermano(A, B) >.

hermano(A, B) :- arco(C, A), arco(C, B).

--------------------------------------------------------------------------------

b) Predicado < alcanzable(A, B) >.

alcanzable(X,Y) :- arco(X,Y).
alcanzable(X,Y) :- arco(X,Z), alcanzable(Z,Y).

--------------------------------------------------------------------------------

c) Predicado < lca(A, B, C) >.

lca(A, A, A) :- !.
lca(A, B, C) :-
    alcanzable(C, A),
    alcanzable(C, B),
    not((alcanzable(C, Z),
        alcanzable(Z, A),
        alcanzable(Z, B),
        Z \= C )).

--------------------------------------------------------------------------------

d) Predicado < tree(A) >.

concatenar([], Y, Y).
concatenar([A|X], Y, [A|Z]) :- concatenar(X,Y,Z).

tree(A) :-
    findall(X, alcanzable(A, X), NodosT),
    concatenar([A], NodosT, Nodos),
    findall(X-Y, (member(X,Nodos), member(Y,Nodos), X\=Y, arco(X,Y)), Aristas),
    length(Nodos, LN),
    length(Aristas, LA),
    LA is LN - 1.
```

En resumen, podemos decir para cada apartado que

(a) Es directo.

(b) Es por el análisis de recursión ya que, si X alcanza a Y y no es un arco directo, entonces existe un arco entre X y un Z y a su vez ese Z alcanza a Y.

(c) Acá pensamos en usar el predicado “alcanzable” tal que dicho C alcance tanto A como B, luego debemos verificar que no existe un Z distinto de C tal que alcance A y B.

(d) Definimos la concatenación de dos listas primero. Luego el predicado tree establece que debemos calcular la cantidad de nodos en un árbol, este lo calculamos con las 2 primeras líneas (findall… y concatenar…) donde findall busca todos los nodos que son alcanzable desde la raíz, pues es un árbol se cumple que la raíz alcance a todos los demás nodos, luego a dicha lista NodosT se la concatena la raíz, teniendo así todos los nodos del árbol. Luego un segundo findall… que arma los arcos almacenándolas en una lista Aristas. Luego debemos verificar que la cantidad de aristas es igual a la cantidad de nodos menos 1.

INVESTIGACION

- Parte 1: (5 puntos) Decimos que una formula lógica es *ground* si no hay variables libres que ocurren en dicha formula. Por ejemplo, si consideramos que las letras mayúsculas son variables y las minúsculas son constantes, *f(a)* y *g(b; f(c))* son formulas ground, mientras que *f(X)* y *g(Y; f(c))* no lo son. Tomando en cuenta estas definiciones, responda las siguientes preguntas:

  a) Hay 3 valores posibles para X: a, b, c y 3 valores posibles para Y: a, b, c Como la fórmula es p(X) ^ q(Y), hay 3*3 = 9 combinaciones posibles de valores para X e Y que hacen la fórmula ground. Por lo tanto, la respuesta es 9 fórmulas ground posibles.

  b) Como f y g se agregan al dominio ya existente {a,b,c}, esto garantiza un conjunto potencialmente infinito, es decir por cada combinación de f se generan nuevas de g y viceversa, para esas nuevas de g se generan para f e incluso para sigo mismo de ambas partes. Por lo que la cantidad de fórmulas ground son infinitas.

  c) Se propone la siguiente definición del conjunto que contiene los valores tanto para X o Y, esto mediante recursión e inducción se tiene:

  Sea **Do = {a, b, c}** el dominio base.

  Entonces, el conjunto de valores posibles para X e Y se define como:

  $D_0 = \{a, b, c\}$

  $D_1 = D_0 \cup \{f(x) \mid x \in D_0\}$

  $D_2 = D_1 \cup \{g(x,y) \mid x,y \in D_1\}$

  ...

  $D_n = D_{n-2} \cup \{f(x) \mid x \in D_{n-2}\} \cup \{g(x,y) \mid x,y \in D_{n-1}\}$ con $n>=2$

  d) No existe, siempre quedara Y en q(g(f(f(b)), Y), c) tal que no sea hecho.

  Primera sustitución de r(c)
  p(f(X)) :- q(X, c).
  q(g(X, Y), c) :- r(X), r(c), q(f(c), a).
  q(X, a).

  Siguiente sustitución con Z = c, r(c) == true y X = f(c) para q,
  p(f(X)) :- q(X, c).
  q(g(X, Y), c) :- r(X), r(c), q(f(c), a).
  q(f(c), a).
  Siguiente sustitución con X = f(f(b)) para r(X)

p(f(X)) :- q(X, c).
q(g(f(f(b)), Y), c) :- r(f(f(b))).

Siguiente sustitución con X como g(f(f(b)).
p(f(g(f(f(b)))) :- q(g(f(f(b)), Y), c).
q(g(f(f(b)), Y), c)

e) Los predicados que son hechos sin ejecutar el programa son r(f(f(b))) y r(c).

f) Se tiene 3 hecho para este caso, es r(f(f(b))), r(c) y q(f(c), a).

g) Se propone la ecuación recursiva Hk como:

$$Hk = Hk\text{-}1 \cup \text{Resolución}(Hk\text{-}1, \text{Reglas})$$

Donde:

- Hk-1 es el conjunto de predicados ground ciertos en la iteración anterior (k-1).
- Reglas es el conjunto de reglas de Horn que define el programa.
- Resolución(Hk-1, Reglas) representa el conjunto de nuevos predicados ground derivados al aplicar resolución entre Hk-1 y Reglas.

De esta forma, en cada paso k:

- Parto del conjunto Hk-1 previo
- Le aplico resolución con las reglas para derivar nuevos predicados ground.
- Estos nuevos predicados se unen a Hk-1 para obtener Hk.

PD: Presiento es que muy simple y directo, pero la verdad es lo ultimo que se me ocurre luego de lo pesada que esta la semana. Esperemos pegue el flechazo.

h) (SIN RESPUESTA)

i) El análisis de niveles son los siguientes:

- r(c):
  - No depende de otros predicados, entonces Nivel 0 para r.
- q(X, Y):
  - Depende de r(X) negado y como r está en el nivel 0, entonces q debe estar a lo sumo en el nivel 1, por lo tanto, Nivel 1 para q.

- q(X, a):
  - No depende de otros predicados, entonces Nivel 0 para q.

- p(X):
  - Depende de q(X, Y) positivo en nivel 0 y q(X, X) negado en nivel 1, entonces p debe estar a lo sumo en el nivel 1, por tanto, Nivel 1 para p.

j) (SIN RESPUESTA)

k) (SIN RESPUESTA)