Universidad Simón Bolívar

Departamento de Computación y Tecnología de la Información.

CI–3661 – Laboratorio de Lenguajes de Programación.

Enero–Marzo 2020

Integrantes: José Ramón Barrera Melchor **carnet:** 15-10123

Carlos Rafael Rivero Panades **carnet:** 13-11216

Tarea IV: Prolog (5 pts).

Implementación:

(1.5 pts) – Considere la siguiente definición de los números de Church:

> "El cero (0) es representado por la constante universal *, luego cada número natural se representa como la aplicación repetida de una función up sobre el cero. El número es entonces igual a la cantidad de aplicaciones de dicha función."

Por ejemplo:

$$* \rightarrow 0$$

$$up(*) \rightarrow 1$$

$$up(up(*)) \rightarrow 2$$

$$up(up(up(*))) \rightarrow 3$$

Tomando en cuenta la definición anterior, se desea que implemente en Prolog una calculadora para números de Church. De los tres argumentos, los primeros dos deben estar siempre instanciados. El tercero puede o no estarlo (Nota: No debe transformar los números de Church en números enteros).

a) (0.5 pts) – Debe implantar el predicados suma/3, que tomar los primeros dos argumentos y almacenar, en el tercer argumento, el resultado de evaluar la suma de los mismos.

b) (0.5 pts) – Debe implantar el predicados resta/3, que tomar los primeros dos argumentos y almacenar, en el tercer argumento, el resultado de evaluar la resta de los mismos.

c) (0.5 pts) – Debe implantar el predicados producto/3, que tomar los primeros dos argumentos y almacenar, en el tercer argumento, el resultado de evaluar el producto de los mismos.

```
#Definimos el 0, para que las operaciones siguientes lleguen a true
church(*).

#Definimos el "sucesor", para que todos los número lleguen a true
church(up(A)) :- church(A).

#Definimos la suma para números de church
suma(A, *, A) :- church(A).
suma(A, up(B), up(C)) :-
  suma(A, B, C).

#Definimos la resta para números de church
resta(A, *, A) :- church(A).
resta(up(A), up(B), C) :-
  resta(A, B, C).

#Definimos la multiplicación para números de church, usando la suma
producto(A, *, *) :- church(A).
producto(A, up(B), C) :-
  suma(D, A, C),
  producto(A, B, D).
```

Investigación:

a) (0.25 pts) – Considere el dominio de valores {a, b, c}. ¿Cuántas posibles fórmulas ground (en el dominio sugerido) se pueden construir a partir de la fórmula $P(x) \wedge Q(y)$?
Nota: Recuerde que una fórmula es ground cuando todas sus variables están completamente instanciadas con átomos en el dominio de valores. Por ejemplo: $P(a) \vee Q(b)$.

Existen 9 posibles formulas ground, estas son:

	a	b	c
a	P(a) ∧ Q(a)	P(a) ∧ Q(b)	P(a) ∧ Q(c)
b	P(b) ∧ Q(a)	P(b) ∧ Q(b)	P(b) ∧ Q(c)
c	P(c) ∧ Q(a)	P(c) ∧ Q(b)	P(c) ∧ Q(c)

b) (0.25 pts) – Aumente ahora el dominio, con dos funcionales f (de aridad 1) y g (de aridad 2). ¿Cuántas posibles fórmulas ground (en el dominio sugerido) se pueden construir ahora a partir de la fórmula P(x) ∧ Q(y)?

Asumiendo que las funciones f y g no se pueden componer, y que el orden de los parámetros de g si importa, serían $15^2 = 225$ posibles fórmulas ground. En caso que se puedan componer existen infinitas posibles fórmulas ground.

c) (0.5 pts) – Plantee el conjunto de posibles valores que pueden reemplazar bien sea a *x* o a *y* en la fórmula de la pregunta anterior (Nota: Puede utilizar las operaciones básicas de conjuntos, así como definiciones inductivas).

Asumiendo que las funciones f y g no se pueden componer, y que el orden de los parámetros de g si importa, tenemos el siguiente conjunto:

A es el conjunto de los posibles valores de X y Y en la fórmula anterior

$$A = \{ n \mid (n = f(v) \vee n = g(v, w) \vee n = v) \wedge (v,w \in \{a,b,c\}) \}$$

O

$$A = \{a, b, c, f(a), f(b), f(c), g(a,a), g(a,b), g(a,c), g(b,a), g(b,b), g(b,c), g(c,a), g(c,b), g(c,c)\}$$

Asumiendo que las funciones f y g se pueden componer, y que el orden de los parámetros de g si importa, en ese caso existen infinitos posibles valores para X y Y, de la forma g(A,A) o f(A).