Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información.
CI–3661 – Laboratorio de Lenguajes de Programación.
Septiembre–Diciembre 2012

Tarea III: Prolog – *Solución*

*(*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:

$$\begin{array}{ccc} * & \mapsto & 0 \\ \text{up}(*) & \mapsto & 1 \\ \text{up}(\text{up}(*)) & \mapsto & 2 \\ \text{up}(\text{up}(\text{up}(*))) & \mapsto & 3 \end{array}$$

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.

Por ejemplo:

```
suma(up(up(*)), up(*), X).
X = up(up(up(*)))
```

```
suma(*, Y, Y).

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

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.

Por ejemplo:

```
resta(up(up(*)), up(*), X).
X = up(*)
```

```
resta(X, Y, Z):-
    suma(Y, Z, X).
```

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.

Por ejemplo:

Por ejemplo:

```
producto(up(up(up(*))), up(up(*)), X).
X = up(up(up(up(up(*)))))
```

```
producto(_, *, *).
```

```
producto(X, up(Y), Z):-
    producto(X, Y, Z1),
    suma(X, Z1, Z).
```

Investigación:

a) (0.25 pts) – Considere el dominio de valores $\{a, b, c\}$. ¿Cuantas posibles fórmulas *ground* (en el dominio sugerido) se pueden construir a partir de la fórmula $P(x) \wedge Q(y)$?

$3^2 = 8$

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

Infinitas

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*).

Sea U el conjunto que se desea hallar. Se define U de manera inductiva:

1. $\{a,\ b,\ c\} \subseteq U$
2. Si $X \in U$, entonces $f(X) \in U$
3. Si $X \in U$ e $Y \in U$, entonces $g(X,\ Y) \in U$
4. Nada más pertenece a U.

Nota: Esto se conoce como 'Universo de Herbrand'.

d) (0.5 pts) – Considere el siguiente programa:

```
p(f(X)) :- q(X, Y), r(Y).

q(g(X, Y), Z) :- r(X), r(Z), q(f(Z), a).
q(X, a).

r(f(f(b))).
r(c).
```

Diga un modelo para el programa (la conjunción de todas las fórmulas).

$$\{\ p(f(g(c,a))),\ \ q(g(c,a),c),\ \ r(c),\ \ q(f(c),a)\ \}$$

e) (0.25 pts) – Considerando el mismo programa, diga cuales predicados *ground* son ciertos sin ejecutar el programa (solo hechos).

$$H_0 = \{\ r(c),\ \ r(f(f(b)))\ \}\ \cup\ \{\ q(X,a)\ \mid\ X \in U\ \}$$

f) (0.25 pts) – Considerando el mismo programa, diga cuales predicados *ground* son ciertos ejecutando a lo sumo una sola vez las reglas.

$$H_1 = \{\ q(g(X,Y),Z)\ \mid\ Y \in U\ \ \wedge\ \ X,Z \in \{c,\ \ f(f(b))\}\ \} \cup H_0$$

g) (0.5 pts) – Plantee una ecuación general recursiva H_k, que dado el conjunto de predicados *ground* que son ciertos ejecutando a lo sumo $k-1$ veces las reglas (H_{k-1}), calcule dicho conjunto ejecutando las reglas a lo sumo k veces.

$$\begin{aligned} H_k \ =\ & \{\ p(f(X),\ Y)\ \mid\ q(X,\ Y)\ \in\ H_{k-1}\ \wedge\ r(Y)\ \in\ H_{k-1}\ \} \\ & \cup\ \{\ q(g(X,\ Y),\ Z)\ \mid\ r(X)\ \in\ H_{k-1}\ \wedge\ r(Z)\ \in\ H_{k-1}\ \wedge\ q(f(Z),\ a)\ \in\ H_{k-1}\ \wedge\ Y\ \in\ U\ \} \\ & \cup\ H_{k-1} \end{aligned}$$

h) (0.25 pts) – Sea k^* el valor más bajo de k tal que $H_k = H_{k+1}$ (mínimo punto fijo de H). Diga si el modelo que encontró en la parte (d) corresponde a un subconjunto de H_{k^*}. ¿Su respuesta sería igual para cualquier otro modelo del programa?

Si, lo es. Y cualquier otro modelo también es subconjunto de H_{k^*}.

Nota: Esto se conoce como 'Base de Herbrand'.

i) (0.25 pts) – Considere el siguiente programa:

```
p(X) :- q(X, Y), not(q(X, X)), not(r(Y)).

q(X, Y) :- r(X), p(Y).
q(X, a).

r(X) :- not(q(X, b)).
r(c).
```

Se dice que un predicado p es de nivel k si todos los predicados negados que aparezcan en cada cuerpo donde p esté en la cabeza, están a lo sumo e nivel $k-1$. El nivel de un predicado que no depende de otros que estén negados es 0.

Diga el nivel correspondiente a los predicados p, q y r

nivel de $p = 2$

nivel de $q = 0$

nivel de $r = 1$

j) (0.25 pts) – Calcule H_{k^*} para el programa anterior: Tomando en cuenta primero para los predicados de nivel 0 (de haberlos), luego para los predicados de nivel 1 (de haberlos) y así en adelante.

Dado un detalle en la definición de la pregunta anterior, esta pregunta tiene varias respuestas válidas. Cualquier de ellas se considerará correcta.

k) (0.25 pts) – Si existen ciclos de predicados negados en un programa, los niveles antes mencionados no están bien definidos. Discuta como afecta esto el cálculo de H_{k^*}.

Potencialmente, no se podría llegar nunca a incluir conocimiento nuevo a H_{k^*}.