Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información.
CI–3661 – Laboratorio de Lenguajes de Programación.
Abril–Julio 2023

Tarea 2: Programación Lógica y Prolog

(30 pts)

Implementación:

1. (7 puntos) – 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:

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

   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) Implemente el predicado `suma/3`, que debe ser cierto cuando el tercer argumento corresponda a la suma de los primeros dos argumentos.

   Por ejemplo:
   suma(up(up(*)), up(*), X).

   X = up(up(up(*)))

   b) Implemente el predicado `resta/3`, que debe ser cierto cuando el tercer argumento corresponda a la resta positiva de los primeros dos argumentos (nótese que la resta está indefinida si el primer argumento es menor que el segundo).

   Por ejemplo:
   resta(up(up(*)), up(*), X).

   X = up(*)

   c) Implemente el predicados `producto/3`, que debe ser cierto cuando el tercer argumento corresponda al producto de los primeros dos argumentos.
   Por ejemplo:

   Por ejemplo:
   producto(up(up(up(*))), up(up(*)), X).

   X = up(up(up(up(up(up(*))))))

2. (7 puntos) – Considere hechos de la forma `arco(A, B)` en ProLog, representando una conexión dirigida desde un nodo A hasta un nodo B.

   a) Implemente un predicado `hermano(A, B)` que se satisfaga si existe nodo C, tal que exista un arco desde C hasta A y un arco desde C hasta B.

   b) Implemente un predicado `alcanzable(A, B)` que se satisfaga si existe algún camino, siguiendo 0 o más arcos desde A hasta B. (Puede suponer que el grafo que inducen los arcos es acíclico).

   c) Implemente un predicado `lca(A, B, C)` que se satisfaga si C alcanza tanto A como B; y no existe ningún C' alcanzable desde C, que a su vez alcance también a A y B. (Puede suponer que el grafo que inducen los arcos es acíclico).

   d) Implemente un predicado `tree(A)` que se satisfaga si los nodos alcanzables desde A (en conjunto con los arcos en cuestión) forman un árbol.

3. (7 puntos) – Se desea que escriba en ProLog un resolvedor automático para el Juego Sudoku.

   El programa debe recibir como entrada un tablero 9×9, con el siguiente formato:

   - Cada fila estará en una línea, separadas por un salto de línea.
   - Los valores en una misma fila estarán separados por un espacio en blanco.
   - Cada valor puede ser un número del 1 al 9 o el caracter punto (.), que representa un espacio en blanco.

   Su programa debe imprimir todas las posibles soluciones que son compatibles con los números pasados como entrada.

   Recuerde que una solución válida debe ser tal que:

   - Cada fila debe tener los números del 1 al 9, sin repetidos.
   - Cada columna debe tener los números del 1 al 9, sin repetidos.
   - Cada bloque de 3×3 que resulta de separar el tablero en tercios, en ambas direcciones, debe tener los números del 1 al 9, sin repetidos.

Investigación:

1. (9 puntos) Decimos que una fórmula lógica es *ground* si no hay variables libres que ocurren en dicha fórmula.

   Por ejemplo, si consideramos que las letras mayúsculas son variables y las minúsculas son constantes, $f(a)$ y $g(b, f(c))$ son fórmulas *ground*, mientras que $f(X)$ y $g(Y, f(c))$ no lo son.

   Tomando en cuenta estas definiciones, responda las siguientes preguntas:

   a) Considere el dominio de valores $\{a, b, c\}$. ¿Cuantas posibles fórmulas *ground* (en el dominio sugerido) se pueden construir a partir de la fórmula $p(X) \wedge q(Y)$?

   b) Aumente ahora el dominio, con dos funcionales f (de adiciad 1) y g (de adicidad 2). ¿Cuántas posibles fórmulas *ground* (en el dominio sugerido) se pueden construir ahora a partir de la fórmula $p(X) \wedge q(Y)$?

   c) 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*).

   d) 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).
   Un modelo es un conjunto de predicados (hechos) *ground* que hacen ciertas todas las reglas.

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

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

   g) 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 (en otras palabras, H_{k-1}), calcule dicho conjunto ejecutando las reglas a lo sumo k veces.

   h) 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?

i) Considere el siguiente programa:

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

q(X, Y) :- not(r(X)), q(Y, a).
q(X, a).

r(c).
```

Se dice que un predicado p es de nivel k si todos los predicados q que aparezcan en cada cuerpo donde p esté en la cabeza, cumplen lo siguiente:

- Si q aparece positivo (sin negación), q debe estar a lo sumo en el nivel k.
- Si q aparece negativo (con negación), q debe estar a lo sumo en el nivel $k-1$.

El nivel de un predicado que no depende de otros es 0.

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

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

k) 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^*}.

Detalles de la Entrega

La entrega de la tarea consistirá de un único archivo PDF con su implementación para las funciones pedidas y respuestas para las preguntas planteadas.También debe incluir un enlace al reporitorio `git` donde se encuentra su implementación para la pregunta 3 de implementación.

La tarea deberá ser entregada al prof. Ricardo Monascal *únicamente* a su dirección de correo electrónico:
(rmonascal@gmail.com) a más tardar el Martes, 4 de Julio, a las 11:59pm. VET.