Lógica Computacional 2017-2
Boletín de ejercicios 3
Lógica de predicados

Lourdes del Carmen González Huesca   Pilar Selene Linares Arévalo

13 de marzo de 2017

Sintaxis

1. Para las siguientes fórmulas, señala el alcance de los cuantificadores. Además para cada presencia de variable indica si se trata de una variable libre o ligada.

   a) $\forall x \exists y (S(x) \to M(y,x)) \land T(y) \leftrightarrow \exists y R(x,y) \land P(y)$

   b) $\forall x \exists y S(y) \to \forall z \big(R(x,z) \land R(y,z) \land \exists x (P(x) \lor R(x, f(y)))\big)$

2. Define recursivamente las siguientes funciones sobre términos:

   a) `ncv: TERM -> N` , que calcula el número de presencias de constantes y variables.

   b) `reemp: TERM -> VAR -> TERM -> TERM`, tal que `reemp t v t1` regresa el resultado de reemplazar en `t` las apariciones de la variable `v` por el término `t1`.

3. Define recursivamente las siguientes funciones para fórmulas de lógica de primer orden:

   a) `pred: FORM -> {P}`, que calcula el conjunto de símbolos de predicado en una fórmula.

   b) `nva: FORM -> N`, que calcula el número de variables ligadas en una fórmula.

4. Sean t un término y n_i el número de presencias de símbolos de función de índice i en t.
   Sea $ncv(t)$ el número de presencias de variables y constantes en t definido en el ejercicio anterior.
   Muestra que
   $$ncv(t) = 1 + \sum_i (i-1) n_i$$

Sustitución

1. Obtener las composiciones $\sigma\rho$ y $\rho\sigma$ a partir de los siguientes pares:

   a) $\sigma = [x := h(a,b,c,y)]$, $\rho = [w := f(g(h(k(x,b))))]$

   b) $\sigma = [u,v,w := c, f(z), f(g(h(x)))]$, $\rho = [u,x,z := v, k(u), c]$

2. Realiza las siguientes sustituciones:

   a) $\Big(\forall v \forall w (P(u,v,a) \land R(a,w) \land Q(u,w))\Big)[y := f(z), z := g(a,b), x := f(y)]$

b) $\Big(\forall x(R(u,v,w)\vee P(x)) \to \exists y(P(f(y))\vee R(y,x,a))\Big)[u:=f(a), v:=g(x), w:=y, x:=k(u), y:= f(a)]$

c) $(\exists x(\forall y(R(x,y)\to Q(z))\wedge Q(x)))[x:=g(h(a,y)), y:=g(b), z:=f(y)]$

d) $(W(g(y))\wedge \exists z V(f(x,a),g(z))\wedge U(f(x,z)))[x,y,z:=z,f(z,z),g(y,x)]$

e) $(\forall v\exists y S(v,y,z)\vee \exists z P(f(v),g(y),z))[w:=f(u), u:=g(y,z), y:=b]$

f) $(R(a,y)\leftrightarrow \exists y\exists z(P(x,y,a)\wedge R(a,z)\wedge Q(z,x)))[y:=f(z), z:=g(a,b), x:=f(y)]$

g) $\Big(\forall x\exists w(Q(w,y,x)\wedge \exists y P(f(x),w))\Big)[y,w:=f(a),f(x)]$

3. Demuestra, utilizando inducción estuctural, las propiedades de las sustituciones en fórmulas de primer orden.

   a) Si $x\notin FV(\varphi)$ entonces $\varphi[\vec{x}, x:=\vec{t}, r\,] = \varphi[\vec{x}:=\vec{t}\,]$.

   b) Si $\vec{x}\cap FV(\varphi)=\varnothing$ entonces $\varphi[\vec{x}:=\vec{t}\,]=\varphi$.

   c) $FV(\varphi[\vec{x}:=\vec{t}\,])\subseteq (FV(\varphi)-\vec{x})\cup FV(\vec{t})$.

   d) Si $x\notin \vec{x}\cup Var(\vec{t})$ entonces $x\in FV(\varphi)$ si y sólo si $x\in FV(\varphi[\vec{x}:=\vec{t}\,])$.

   e) Si $x\notin \vec{x}\cup Var(\vec{t})$ entonces $\varphi[x:=t][\vec{x}:=\vec{t}\,] = \varphi\big[\vec{x}:=\vec{t}\,\big]\big[x:=t[\vec{x}:=\vec{t}\,]\big]$.

   f) Si $x\notin \vec{x}\cup Var(\vec{t})$ entonces $\varphi[\vec{x}, x:=\vec{t}, t] = \varphi[\vec{x}:=\vec{t}\,][x:=t]$.

Especificación formal

1. Dado el lenguaje de predicados para listas que se describe a continuación, obtener las traducciones al español de las fórmulas dadas.
   Sean $\mathcal{L}=\{L^{(1)}, C^{(2)}, R^{(2)}, I^{(2)}, f^{(2)}, a\}$ y $\mathcal{M}=\langle Lista(\mathbf{N}),\mathcal{I}\rangle$ tal que $R^{\mathcal{I}}(x,y):=y$ es la reversa de x, $C^{\mathcal{I}}(x,y):=y$ es la cola de x, $L^{\mathcal{I}}(x):=x$ es una lista unitaria, $I^{\mathcal{I}}(x,y):=x$ es igual a y, $f^{\mathcal{I}}(x,y):=$ la concatenación de x con y, y $a^{\mathcal{I}}:=$ es la lista vacía.

   a) $C(x,y)\wedge L(y)$

   b) $R(f(x,y),z)\wedge R(f(y,x),z)\to I(x,y)$

   c) $I(x,y)\to R(f(x,y),z)\wedge R(f(y,x),z)$

   d) $\exists y C(x,y)\vee \exists z R(x,z)$

   e) $\exists z(R(z,a)\vee \neg I(z,a))$

   f) $\forall x(L(x)\to R(f(x,a),x)\wedge C(f(x,a),a))$

   g) $R(f(x,y),f(y,x))\wedge C(x,v)\wedge C(y,w)\to I(v,w)$

   h) $\forall x(R(f(x,a),x)\wedge C(f(x,a),a)\leftrightarrow L(x))$

   i) $(\forall x\exists y R(x,y)\to I(x,y))\vee(\forall x\exists y R(x,y)\to R(y,x))$

   j) $\forall x\exists y(L(x)\wedge C(x,y)\to I(x,f(x,y)))$

2. Realizar la especificación formal acerca de las siguientes propiedades de la función que verifica la pertenencia de un elemento a una lista dada. En cada caso dar dos especificaciones, una funcional y otra relacional. Definir primero el lenguaje a utilizar.

   a) Existe una lista, tal que no es unitaria pero su cola si es unitaria.

b) Toda lista puede verse como la concatenación de dos listas.

c) Ningún elemento pertenece a la lista vacía.

d) Si cierto elemento es la cabeza de una lista, entonces dicho elemento pertenece a esa lista.

e) La cabeza de una lista pertenece a dicha lista.

f) Si un elemento pertenece a una lista entonces o es la cabeza de esa lista o pertenece a la cola.

g) Si un elemento pertenece a una lista entonces pertenece a cualquier otra lista cuya cola es la lista anterior.

h) Si un elemento pertenece a la concatenación de dos listas entonces pertenece a alguna de estas dos listas.

i) Si un elemento pertenece a una lista entonces dicha lista es la concatenación de otras dos donde la segunda tiene como cabeza a dicho elemento.

j) Que un elemento pertenezca a la reversa de una lista equivale a que pertenezca a la lista.

3. Realizar la especificación formal de las siguientes propiedades de las funciones reversa de una lista y concatenación de dos listas. Definir primero el lenguaje a utilizar.

   a) La operación de concatenación es asociativa.

   b) Si la concatenación de dos listas es vacía entonces ambas listas son vacías.

   c) Si la concatenación de dos listas es una lista unitaria entonces una de dichas listas es vacía y la otra es la misma lista unitaria.

   d) La reversa de la concatenación de dos listas es la concatenación de la reversa de la segunda lista con la reversa de la primera lista.

   e) La reversa de la reversa de una lista es la misma lista.

   f) Si la reversa de una lista es la misma lista entonces dicha lista es vacía o unitaria.

4. En el lenguaje de programación LISP, una expresión simbólica o S-expresión es alguna de las siguientes

   a) Un símbolo.

   b) Una S-lista, donde una S-lista es una sucesión, tal vez vacía, de S-expresiones, encerrada entre paréntesis.

   Por ejemplo, las siguientes son cinco S-expresiones:

   `e, (), (e z), (b (a m)), (f (a () ) v (i o))`

   Dé una especificación formal de las S-expresiones.

5. Realizar la especificación formal acerca del tipo abstracto de datos pila considerando homogeneidad, es decir, suponiendo que todos los elementos en una pila son del mismo tipo digamos A. En cada caso dar dos especificaciones, una funcional y otra relacional. Definir primero el lenguaje a utilizar.

   a) Definición del tipo de datos pila:

      i. La vacía es una pila (de elementos de A).
      ii. El resultado de agregar un elemento de A en el tope de una pila es una pila.

iii. Son todos.

b) Si una pila no es vacía entonces el elemento en el tope es un elemento de A

c) Si una pila no es vacía entonces la pila que resulta al eliminar el elemento en el tope es una pila de elementos de A.

d) Si una pila no es vacía entonces la pila que resulta al agregar el tope de la pila dada a la pila obtenida al eliminar el tope de la pila dada es la pila dada.

e) La pila vacía no tiene elementos.

f) La pila que resulta de agregar un elemento a una pila dada tiene un elemento más que la pila dada.

g) Si la pila resultante de eliminar el tope de una pila dada es vacía entonces la pila dada tiene sólo un elemento.

6. Realizar una especificación formal de alguna estructura o tema de interés particular. Definir primero el lenguaje a utilizar. Algunas sugerencias son:

   a) Sistemas numéricos y sus propiedades:
      1) Enteros, racionales, etc..
      2) Binarios, ternarios, etc..
      3) Propiedades de conversión entre sistemas.

   b) Estructuras de datos y sus propiedades:
      1) Listas con el constructor `snoc`
      2) Listas heterogéneas con elementos de dos tipos dados.
      3) Árboles binarios
      4) Colas, tablas, arreglos, etc.
      5) Árboles rojinegros, árboles AVL, etc..
      6) Conjuntos.

   c) Micromundos:
      1) Extensión del micromundo de figuras a tres dimensiones.
      2) Mundo de bloques agregando un brazo de robot (reglas para que el brazo mueva los bloques (planeación en inteligencia artificial))
      3) Sistema de correo electrónico.
      4) Sistema de funcionamiento de un elevador.

   d) Reglas de algún juego de su interés (juegos de mesa, juegos de rol, ajedrez, sudoku, etc)

7. Tenemos seis cubos de color amarillo, azul o verde. Un cubo puede estar uno sobre otro o en el piso. Considerese la signatura $\Sigma = \{S^{(2)}, A^{(1)}, Az^{(1)}, V^{(1)}, L^{(1)}, p\}$ donde $S(x, y)$ significa “x está sobre y”; $A(x), Az(x), V(x)$ representan los colores amarillo, azul y verde; $L(x)$ significa que “x está libre”, es decir, ningún cubo está sobre x, y la constante p representa al piso.
   Simbolizar los siguiente:

   a) Hay un cubo azul sobre el piso con un cubo amarillo sobre él y un cubo verde sobre el amarillo.

   b) Ningún cubo amarillo está libre.

   c) Hay un cubo azul libre y un cubo verde y libre.

d) Cualquier cubo amarillo tiene un cubo sobre él.

e) No todos los cubos azules están libres.

f) Cualquier cubo verde está libre.

g) Todos los cubos sobre el piso son azules.

h) Cualquier cubo que esté sobre un cubo amarillo es verde o azul.

i) Hay un cubo verde sobre un cubo verde.

j) Hay un cubo amarillo libre sobre el piso.

k) Ningún cubo está sobre el piso.

l) Hay un cubo amarillo que está sobre uno azul y hay un cubo azul sobre él.

m) Todos los cubos están sobre algo.