Universidad Simón Bolívar
Dpto. de Computación y Tecnología de la Información
CI2661 - Laboratorio de Lenguajes I
Enero-Abril 2009

Segundo Proyecto:
Programación funcional en Prolog

1 Evaluación de funciones en programación funcional

Desde la perspectiva de la programación funcional, un programa interprete de un lenguaje funcional es un evaluador que calcula las funciones que le introducimos. La tarea del evaluador es simplificar la expresión de una función hasta alcanzar una forma irreductible, que se muestra como solución o respuesta. Cada uno de los pasos que da el evaluador en el proceso de simplificación se llama **reducción**, y lo denotaremos con el símbolo $\Rightarrow$. En cada paso de reducción, el evaluador reconoce una parte de la expresión, llamada **redex** (del inglés *reducible expression*), y la simplifica. Cuando una expresión no puede reducirse más se dice que ha alcanzado su **forma normal**. El proceso de simplificación es en sí ambiguo, como lo muestra el siguiente ejemplo.

Ejemplo 1: dados el tipo de datos

$$\text{data } \mathbb{N} = Cero \mid Suc\ \mathbb{N}$$

y las siguientes funciones

$$\begin{aligned} mas: &\quad \mathbb{N} \longrightarrow \mathbb{N} \longrightarrow \mathbb{N} \\ mas\ m\ Cero = &\quad m \\ mas\ m\ (Suc\ n) = &\quad Suc\ (mas\ m\ n) \end{aligned}$$

$$\begin{aligned} doble: &\quad \mathbb{N} \longrightarrow \mathbb{N} \\ doble\ n = &\quad mas\ n\ n \end{aligned}$$

se puede simplicar la expresión $doble\ (doble\ (Suc\ Cero))$ de varias maneras. Mostramos dos de ellas, enmarcando los redexes a ser reducidos en el siguiente paso y anotando la definición de función y la sustitución adecuadas. En la primera, el criterio es resolver antes los redexes internos, los más anidados y en caso de que haber redexes paralelos (no contenidos unos en otros) se reducen de izquierda a derecha

$$
\begin{array}{ll}
 & doble\ \left(\boxed{\ doble\ (Suc\ Cero)\ }\right) \\
\Rightarrow & \{[doble\ n = mas\ n\ n]\ (n \leftarrow Suc\ Cero)\} \\
 & doble\ \left(\boxed{\ mas\ (Suc\ Cero)\ (Suc\ Cero)\ }\right) \\
\Rightarrow & \left\{[mas\ m\ (Suc\ n) = Suc\ (mas\ m\ n)] \begin{pmatrix} m \leftarrow Suc\ Cero \\ n \leftarrow Cero \end{pmatrix}\right\} \\
 & doble\ \left(Suc\ \left(\boxed{\ mas\ (Suc\ Cero)\ Cero\ }\right)\right) \\
\Rightarrow & \{[mas\ m\ Cero = m]\ (m \leftarrow Suc\ Cero)\} \\
 & \boxed{\ doble\ (Suc\ (Suc\ Cero))\ } \\
\Rightarrow & \{[doble\ n = mas\ n\ n]\ (n \leftarrow Suc\ (Suc\ Cero))\} \\
 & \boxed{\ mas\ (Suc\ (Suc\ Cero))\ (Suc\ (Suc\ Cero))\ } \\
\Rightarrow & \left\{[mas\ m\ (Suc\ n) = Suc\ (mas\ m\ n)] \begin{pmatrix} m \leftarrow Suc\ (Suc\ Cero) \\ n \leftarrow Suc\ Cero \end{pmatrix}\right\} \\
 & Suc\ \left(\boxed{\ mas\ (Suc\ (Suc\ Cero))\ (Suc\ Cero)\ }\right) \\
\Rightarrow & \left\{[mas\ m\ (Suc\ n) = Suc\ (mas\ m\ n)] \begin{pmatrix} m \leftarrow Suc\ (Suc\ Cero) \\ n \leftarrow Cero \end{pmatrix}\right\} \\
 & Suc\ \left(Suc\ \left(\boxed{\ mas\ (Suc\ (Suc\ Cero))\ Cero\ }\right)\right) \\
\Rightarrow & \{[mas\ m\ Cero = m]\ (m \leftarrow Suc\ (Suc\ Cero))\} \\
 & Suc\ (Suc\ (Suc\ (Suc\ Cero)))
\end{array}
$$

Esta estrategia de reducción se conoce como **paso de parámetros por valor** porque los parámetros de una función se reducen antes que la función misma. Presenta dos inconvenientes serios: se pueden efectuar reducciones que no son necesarias para calcular el valor de una expresión, y puede no conducir a la forma normal de la expresión.

En la segunda, el criterio es resolver antes los redexes externos, los menos anidados, y en caso de haber redexes paralelos se reducen de izquierda a derecha

$$
\begin{array}{ll}
 & \boxed{\ doble\ (doble\ (Suc\ Cero))\ } \\
\Rightarrow & \{[doble\ n = mas\ n\ n]\,(n \leftarrow Suc\ Cero)\} \\
 & mas\ \left(\boxed{\ doble\ (Suc\ Cero)\ }\right)\ (doble\ (Suc\ Cero)) \\
\Rightarrow & \{[doble\ n = mas\ n\ n]\,(n \leftarrow Suc\ Cero)\} \\
 & mas\ (mas\ (Suc\ Cero)\ (Suc\ Cero))\ \left(\boxed{\ doble\ (Suc\ Cero)\ }\right) \\
\Rightarrow & \{[doble\ n = mas\ n\ n]\,(n \leftarrow Suc\ Cero)\} \\
 & mas\ \left(\boxed{\ mas\ (Suc\ Cero)\ (Suc\ Cero)\ }\right)\ (mas\ (Suc\ Cero)\ (Suc\ Cero)) \\
\Rightarrow & \left\{[mas\ m\ (Suc\ n) = Suc\ (mas\ m\ n)]\begin{pmatrix} m \leftarrow Suc\ Cero \\ n \leftarrow Cero \end{pmatrix}\right\} \\
 & mas\ (Suc\ (mas\ (Suc\ Cero)\ Cero))\ \boxed{\ (mas\ (Suc\ Cero)\ (Suc\ Cero))\ } \\
\Rightarrow & \left\{[mas\ m\ (Suc\ n) = Suc\ (mas\ m\ n)]\begin{pmatrix} m \leftarrow Suc\ Cero \\ n \leftarrow Cero \end{pmatrix}\right\} \\
 & \boxed{\ mas\ (Suc\ (mas\ (Suc\ Cero)\ Cero))\ (Suc\ (mas\ (Suc\ Cero)\ Cero))\ } \\
\Rightarrow & \left\{[mas\ m\ (Suc\ n) = Suc\ (mas\ m\ n)]\begin{pmatrix} m \leftarrow (Suc\ (mas\ (Suc\ Cero)\ Cero)) \\ n \leftarrow mas\ (Suc\ Cero)\ Cero \end{pmatrix}\right\} \\
 & Suc\ \left(mas\ (Suc\ (mas\ (Suc\ Cero)\ Cero))\ \boxed{\ (mas\ (Suc\ Cero)\ Cero)\ }\right) \\
\Rightarrow & \{[mas\ m\ Cero = m]\,(m \leftarrow Suc\ Cero)\} \\
 & Suc\ \left(\boxed{\ mas\ (Suc\ (mas\ (Suc\ Cero)\ Cero))\ (Suc\ Cero)\ }\right) \\
\Rightarrow & \left\{[mas\ m\ (Suc\ n) = Suc\ (mas\ m\ n)]\begin{pmatrix} m \leftarrow (Suc\ (mas\ (Suc\ Cero)\ Cero)) \\ n \leftarrow Cero \end{pmatrix}\right\} \\
 & Suc\ \left(Suc\ \left(\boxed{\ mas\ (Suc\ (mas\ (Suc\ Cero)\ Cero))\ Cero\ }\right)\right) \\
\Rightarrow & \{[mas\ m\ Cero = m]\,(m \leftarrow Suc\ (mas\ (Suc\ Cero)\ Cero))\} \\
 & Suc\ \left(Suc\ \left(Suc\ \left(\boxed{\ mas\ (Suc\ Cero)\ Cero\ }\right)\right)\right) \\
\Rightarrow & \{[mas\ m\ Cero = m]\,(m \leftarrow Suc\ Cero)\} \\
 & Suc\ (Suc\ (Suc\ (Suc\ Cero)))
\end{array}
$$

Esta estrategia se conoce como **paso de parámetros por nombre** porque se pueden pasar como argumentos de la función, en vez de valores, expresiones que no han sido reducidas aún. Presenta el inconveniente de tener que repetir ciertas reducciones, pero posee la importante propiedad de siempre alcanzar la forma normal. Además, como no se reducen aquellas expresiones que no son necesarias para alcanzar la forma normal, es posible trabajar con estructuras de datos infinitas.

La tercera estrategia, llamada **paso de parámetros por necesidad**, también conocida como **evaluación perezosa**, posee las ventajas del paso por nombre y la eficiencia del paso por valor: consiste en utilizar paso por nombre y recordar los valores de los argumentos ya calculados para evitar repetir las

reducciones

$$
\begin{array}{ll}
 & \boxed{doble\ (doble\ \ (Suc\ Cero))} \\
\Rightarrow & \{[doble\ n = mas\ n\ n]\,(n \leftarrow Suc\ Cero)\} \\
 & mas\ x\ x \text{ where } x = \boxed{doble\ \ (Suc\ Cero)} \\
\Rightarrow & \{[doble\ n = mas\ n\ n]\,(n \leftarrow Suc\ Cero)\} \\
 & mas\ x\ x \text{ where } x = \boxed{mas\ y\ y} \text{ where } y = Suc\ Cero \\
\Rightarrow & \left\{[mas\ m\ (Suc\ n) = Suc\ \ (mas\ m\ n)] \begin{pmatrix} m \leftarrow y \\ n \leftarrow Cero \end{pmatrix}\right\} \\
 & mas\ x\ x \text{ where } x = Suc\ (\boxed{mas\ \ (Suc\ Cero)\ \ Cero}) \\
\Rightarrow & \{[mas\ m\ Cero = m]\,(m \leftarrow Suc\ Cero)\} \\
 & \boxed{mas\ x\ x} \text{ where } x = Suc\ \ (Suc\ Cero) \\
\Rightarrow & \left\{[mas\ m\ (Suc\ n) = Suc\ \ (mas\ m\ n)] \begin{pmatrix} m \leftarrow x \\ n \leftarrow Suc\ Cero \end{pmatrix}\right\} \\
 & Suc\ (\boxed{mas\ \ (Suc\ y)\ \ y}) \text{ where } y = Suc\ Cero \\
\Rightarrow & \left\{[mas\ m\ (Suc\ n) = Suc\ \ (mas\ m\ n)] \begin{pmatrix} m \leftarrow Suc\ y \\ n \leftarrow Cero \end{pmatrix}\right\} \\
 & Suc\ \big(Suc\ \ (\boxed{mas\ \ (Suc\ \ (Suc\ Cero))\ \ Cero})\big) \\
\Rightarrow & \{[mas\ m\ Cero = m]\,(m \leftarrow Suc\ \ (Suc\ Cero))\} \\
 & Suc\ (Suc\ \ (Suc\ \ (Suc\ Cero)))
\end{array}
$$

2 Reducciones más externas en los sistemas de reescritura

- Una **signatura** es un conjunto no vacío $\mathcal{F}$ de nombres de función. Asociado con cada $f \in \mathcal{F}$ hay un número natural, llamado **aridad**, denotando la cantidad de argumentos, con lo cual

$$\mathcal{F} = \bigcup_{n \geq 0} \mathcal{F}_n$$

donde cada $\mathcal{F}_n$ agrupa los nombres de las funciones con n argumentos. Las funciones con aridad $n = 0$ se llaman **constantes**.

Ejemplo 2: Sea la firma $\mathcal{F}$ que contiene los nombres de funciones $cero, suc, mas, por$ con aridades respectivas $0, 1, 2, 2$. Entonces $\mathcal{F}_0 = \{cero\}$, $\mathcal{F}_1 = \{suc\}$, $\mathcal{F}_2 = \{mas, por\}$ y, para todo $n > 2$, $\mathcal{F}_n = \varnothing$, con lo cual

$$\mathcal{F} = \mathcal{F}_0 \cup \mathcal{F}_1 \cup \mathcal{F}_2$$

- Sea $\mathcal{X}$ un conjunto numerable de símbolos que representan variables, tal que $\mathcal{F} \cap \mathcal{X} = \varnothing$. Un **término** es una variable, una constante o una función cuyos argumentos son términos. El **conjunto de términos** $\mathcal{T}(\mathcal{F}, \mathcal{X})$ sobre $\mathcal{F}$ y $\mathcal{X}$ se define recursivamente:

  i) son términos las variables

$$\mathcal{X} \subset \mathcal{T}(\mathcal{F}, \mathcal{X})$$

ii) son términos las constantes (funciones de aridad 0)

$$\mathcal{F}_0 \subset \mathcal{T}(\mathcal{F}, \mathcal{X})$$

iii) son términos las funciones cuyos argumentos son términos

$$(\forall n, f : n > 0 \wedge f \in \mathcal{F}_n : t_1, \ldots, t_n \in \mathcal{T}(\mathcal{F}, \mathcal{X}) \Rightarrow f(t_1, \ldots, t_n) \in \mathcal{T}(\mathcal{F}, \mathcal{X}))$$

Un término es **lineal** si cada variable ocurre a lo sumo una vez en él, es decir, cuando no tiene variables repetidas.

Ejemplo 3: Sea $X = \{x_i \mid i \in \mathbb{N}\}$. Entonces,

$$\begin{array}{rcl} \mathcal{T}(\mathcal{F}, \mathcal{X}) & = & \mathcal{X} \cup \{cero\} \cup \\ & & \left\{ \begin{array}{c} suc(cero), suc(x_0), suc(suc(cero)), suc(suc(x_0)), \\ suc(suc(mas(cero, cero)), suc(suc(mas(cero, x_0))... \end{array} \right\} \end{array}$$

- La **función de posición** devuelve la lista de las posiciones dentro del término

$$\begin{array}{rl} pos: & \mathcal{T}(\mathcal{F}, \mathcal{X}) \longrightarrow \mathcal{P}(\mathbb{N}^*) \\ pos(t) = & \left\{ \begin{array}{ll} \{\lambda\} & ; \text{ si } t \in \mathcal{X} \cup \mathcal{F}_0 \\ \{\lambda\} \cup \{ip \in \mathbb{N}^* \mid p \in pos(t_i), 1 \leq i \leq n\} & ; \text{ si } t = f(t_1, \ldots, t_n) \end{array} \right. \end{array}$$

El **conjunto de posiciones** de un término t es la $pos(t)$. Las posiciones son secuencias de números naturales distintos de cero con λ denotando la secuencia vacía. En Prolog, una posición se puede representar como una lista de enteros positivos con $\lambda = []$.

El **orden de prefijos** en $\mathbb{N}^*$ definido por

$$p \leq q \text{ si, y sólo si, } (\exists p' : p' \in \mathbb{N}^* : p \cdot p' = q)$$

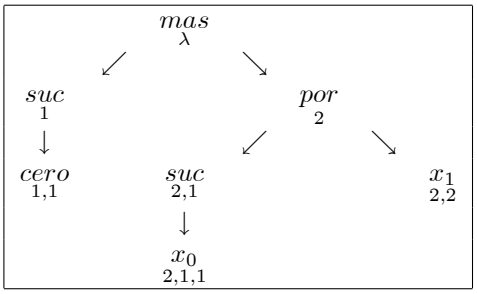
donde $\cdot$ es la concatenación de secuencias, es un orden parcial en el conjunto de posiciones.

Un **contexto** (t, p) es el par formado por un término $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ y una posición p de t, es decir, tal que $p \in pos(t)$.

Ejemplo 4: Sea el término lineal

$$t = mas(suc(cero), por(suc(x_0), x_1))$$

cuya representación en árbol con la indicación de sus posiciones es



y la representación como una lista de posiciones es

$$pos\,(t) = [[],[1],[2],[1,1],[2,1],[2,2],[2,1,1]]$$

Nótese que, por ejemplo, $(2) \leq (2,1,1)$ porque existe $(1,1)$ tal que $(2) \cdot (1,1) = (2,1,1)$, pero $(1,1) \not\leq (2,1)$ y $(2,1) \not\leq (1,1)$, con lo cual $(1,1)$ y $(2,1)$ son incomparables.

Los contextos de t son

$$(\lambda, t)\,((1),t),((2),t),((1,1),t),((2,1),t),((2,2),t) \text{ y } ((2,2,1),t)$$

- La **función de subtérminos** extrae el subtérmino en la posición p de un término dado t

$$\begin{array}{rl} subterm: & \mathbb{N}^* \times \mathcal{T}(\mathcal{F},\mathcal{X}) \longrightarrow \mathcal{T}(\mathcal{F},\mathcal{X}) \\ subterm\,(p,t) = & \begin{cases} t & ;\text{ si } (p=\lambda) \vee (p \notin pos\,(t)) \\ subterm\,(q,t_i) & ;\text{ si } (t = f\,(t_1,\ldots,t_n)) \wedge (\exists i : 1 \leq i \leq n : p = iq) \end{cases} \end{array}$$

Ejemplo 5: Los subtérminos del término del ejemplo anterior son

$$\begin{array}{rcl} subterm\,(\lambda,t) & = & t \\ subterm\,((1),t) & = & suc(cero) \\ subterm\,((2),t) & = & por\,(suc\,(x_0),x_1) \\ subterm\,((1,1),t) & = & cero \\ subterm\,((2,1),t) & = & suc(x_0) \\ subterm\,((2,2),t) & = & x_1 \\ subterm\,((2,1,1),t) & = & x_0 \end{array}$$

Si la posición p no pertenece a $post(t)$ la función devuelve el mismo término

$$subterm\,((1,2),t) = t$$

- La **función de variables** extrae sin repetición las variables de un témino

$$\begin{array}{rl} var: & \mathcal{T}(\mathcal{F},\mathcal{X}) \longrightarrow \mathcal{P}\,(\mathcal{X}) \\ var\,(t) = & \{x \in \mathcal{X} \mid (\exists p : p \in pos(t) : subterm\,(p,t) = x)\} \end{array}$$

Ejemplo 6: siguiendo con el término del ejemplo 4;

$$var(t) = \{x_0, x_1\}$$

- La **función de remplazo** toma dos términos y una posición para modificar el primero colocando el segundo en la posición posición dada

$$\begin{array}{rl} repl: & \mathbb{N}^* \times \mathcal{T}(\mathcal{F},\mathcal{X}) \times \mathcal{T}(\mathcal{F},\mathcal{X}) \longrightarrow \mathcal{T}(\mathcal{F},\mathcal{X}) \\ repl\,(p,t,s) = & \begin{cases} s & ;\text{ si } p = \lambda \\ t & ;\text{ si } p \notin pos\,(t) \\ f\,(t_1,\ldots,repl(q,t_i,s),\ldots,t_n) & ;\text{ si } (t = f\,(t_1,\ldots,t_n)) \wedge (\exists i : 1 \leq i \leq n : p = iq) \end{cases} \end{array}$$

Ejemplo 7: Sean los términos t del ejemplo 4, $s = por(x_2, suc(x_3))$ y las posiciones λ, $(1), (1,2)$ y $(2,2)$ de modo que

$$\begin{aligned} repl(\lambda, t, s) &= s \\ repl((1,2), t, s) &= t \\ repl((1), t, s) &= mas(por(x_2, suc(x_3)), por(suc(x_0), x_1)) \\ repl((2,2), t, s) &= mas(suc(cero), por(suc(x_0), por(x_2, suc(x_3)))) \end{aligned}$$

- La **función de sustitución** es una función, definida a partir de un subconjunto finito $\mathcal{D}$ de $\mathcal{X}$, que asigna a cada variable en $\mathcal{D}$ un término y al resto infinito de las variables a sí mismas

$$\begin{aligned} sust_D : &\quad \mathcal{X} \longrightarrow \mathcal{T}(\mathcal{F}, \mathcal{X}) \\ sust_D(x) = &\quad \begin{cases} x & ; \text{si } x \notin \mathcal{D} \\ t & ; \text{si } x \in \mathcal{D} \wedge t \in \mathcal{T}(\mathcal{F}, \mathcal{X}) \setminus \mathcal{D} \wedge x \notin var(t) \end{cases} \end{aligned}$$

La **función extendida de sustitución** toma una dominio finito de variables y construye un conjunto finito de sustituciones, una por cada variable

$$\begin{aligned} sust : &\quad \mathcal{P}_{\text{fin}}(\mathcal{X}) \longrightarrow \mathcal{P}(\mathcal{X} \times \mathcal{T}(\mathcal{F}, \mathcal{X})) \\ sust(\mathcal{D}) = &\quad \{(x, sust_{\mathcal{D}}(x)) \mid x \in D\} \end{aligned}$$

Los pares (x, t) de la imagen de la función sustitución extendida se denotan con $x \leftarrow t$.

Ejemplo 8: Sea $\mathcal{D} = \{x_0, x_1, x_2\}$ y sea la función de sustitución

$$\begin{aligned} sust_{\{x_0,x_1,x_2\}} : &\quad \mathcal{X} \longrightarrow \mathcal{T}(\mathcal{F}, \mathcal{X}) \\ sust_{\{x_0,x_1,x_2\}}(x) = &\quad \begin{cases} suc(cero) & ; \text{si } x = x_0 \\ mas(x_0, suc(x_3)) & ; \text{si } x = x_1 \\ por(x_3, suc(x_1)) & ; \text{si } x = x_2 \\ x & ; \text{otro modo} \end{cases} \end{aligned}$$

cuya función extendida de sustitución es

$$\begin{aligned} sust : &\quad \mathcal{P}_{\text{fin}}(\mathcal{X}) \longrightarrow \mathcal{P}(\mathcal{X} \times \mathcal{T}(\mathcal{F}, \mathcal{X})) \\ sust(\{x_0, x_1, x_2\}) = &\quad \{x_0 \leftarrow suc(cero),\ x_1 \leftarrow mas(x_0, suc(x_3)),\ x_2 \leftarrow por(x_3, suc(x_1))\} \end{aligned}$$

Dado un conjunto finito de variables, la imagen de la función de sustitución extendida se llama **conjunto de sustituciones**, y denotaremos genericamente con Φ. Denotamos por $\mathcal{S}ust$ al rango de la función $sust$, es decir, al conjunto de todos los conjuntos de sustituciones posibles

$$\mathcal{S}ust = sust(\mathcal{P}_{\text{fin}}(\mathcal{X})) \subset \mathcal{P}(\mathcal{X} \times \mathcal{T}(\mathcal{F}, \mathcal{X}))$$

- La **función de instanciación** toma un conjunto de sustituciones Φ y un término t para devolver la **instancia** de t con los remplazos que especifica Φ

$$inst: \quad \mathcal{S}ust \times \mathcal{T}(\mathcal{F},\mathcal{X}) \longrightarrow \mathcal{T}(\mathcal{F},\mathcal{X})$$

$$inst\,(\Phi,t) = \begin{cases} t & ;\text{ si } (t \in \mathcal{F}_0) \vee (\forall\,(x,t) : (x,t) \in \Phi : x \notin var(t)) \\ s & ;\text{ si } (t = x) \wedge ((x,s) \in \Phi) \\ f\,(inst\,(\Phi,t_1),\ldots,inst\,(\Phi,t_n)) & ;\text{ si } (t = f\,(t_1,\ldots,t_n)) \end{cases}$$

Ejemplo 9: Sea el conjunto de sustituciones del ejemplo 8, y el término t el ejemplo 4, entonces

$$inst\,(sust\,(\{x_0,x_1,x_2\}),t) = mas\,(suc\,(cero),por\,(suc\,(suc(cero)),mas\,(x_0,suc\,(x_3))))$$

Un **redex** es una instancia del término izquierdo de una regla de reescritura.

- La **función de unificación** es una función booleana que es *true* cuando el conjunto de sustituciones iguala las instancias de dos términos,

$$unif: \quad \mathcal{S}ust \times \mathcal{T}(\mathcal{F},\mathcal{X}) \times \mathcal{T}(\mathcal{F},\mathcal{X}) \longrightarrow \{true, false\}$$

$$unif\,(\Phi,t,s) = \quad (inst\,(\Phi,t) = inst\,(\Phi,s))$$

- Un **sistema de reescritura** es un par $(\mathcal{F},\mathcal{R})$ formado por una signatura $\mathcal{F}$ y una conjunto finito $\mathcal{R} \subset \mathcal{T}(\mathcal{F},\mathcal{X}) \times \mathcal{T}(\mathcal{F},\mathcal{X})$ de **reglas de reescritura** cuyos pares (t,s), denotados por $t \longrightarrow s$, cumplen las condiciones:

  i) $t \notin \mathcal{X}$

  ii) $var(s) \subseteq var(t)$

  Un sistema de reescritura es **ortogonal** si

  1) todos los terminales izquierdos de sus reglas son lineales,

  2) ningún términal izquierdo en las reglas unifica con algún subtérmino (no variable y renombrado) de los terminales izquierdos de alguna regla, incluyendo la misma regla.

  Ejemplo 10: Sea la signatura $\mathcal{F}$ del ejemplo 2 y sea el conjunto $\mathcal{R}$ de reglas de reescritura

  1– $mas(x_0,cero) \longrightarrow x_0$
  2– $mas\,(x_0,suc(x_1)) \longrightarrow suc(mas(x_0,x_1))$
  3– $mas(x_0,x_1) \longrightarrow mas(x_1,x_0)$
  4– $mas(mas(x_0,x_1),x_2) \longrightarrow mas(x_0,mas(x_1,x_2))$
  5– $por\,(x_0,cero) \longrightarrow cero$
  6– $por\,(x_0,suc(x_1)) \longrightarrow mas(x_0,por(x_0,x_1))$
  7– $por(x_0,x_1) \longrightarrow por(x_1,x_0)$
  8– $por(por(x_0,x_1),x_2) \longrightarrow por(x_0,por(x_1,x_2))$
  9– $por(x_0,mas(x_1,x_2)) \longrightarrow mas(por(x_0,x_1),por(x_0,x_2))$

  El sistema de reescritura $(\mathcal{F},\mathcal{R})$ en la base para el cálculo aritmético (con suma y producto) en los naturales.

- La **relación de reducción** $\Rightarrow_{\mathcal{R}} \subseteq \mathcal{T}(\mathcal{F},\mathcal{X}) \times \mathcal{T}(\mathcal{F},\mathcal{X})$ relativa al sistema de reescritura $(\mathcal{F},\mathcal{R})$ se define de la siguiente manera: $t \Rightarrow_{\mathcal{R}} s$ si, y sólo si,

  $$(\exists \Phi,(l,r),p:\Phi \in \mathcal{S}ust,(l,r) \in \mathcal{R} \wedge p \in var(t):subterm\,(p,t)=inst(\Phi,l) \wedge s=repl\,(p,t,inst(\Phi,s)))$$

  en cuyo caso diremos que s **reescribe** t al **contraer** el redex $inst(\Phi,l)$. Cuando se quiere especificar la regla de reescritura y la posición de remplazo se denota la relación de reducción con $\Rightarrow_n^p$ donde n es el número de la regla y p la posición.

  Ejemplo 11: Sea el sistema de reescritura del ejemplo 10 los términos

  $$t = mas\,(suc\,(cero)\,,por\,(suc\,(cero)\,,mas\,(cero,suc(cero))))$$

  y

  $$s = mas\,(suc\,(cero)\,,mas(por(suc\,(cero)\,,cero),por(suc\,(cero)\,,suc\,(cero))))$$

  Entonces, tenemos que $t \Rightarrow_9^{(2)} s$ en virtud de la existencia de

  i) el conjunto de sustituciones en $\mathcal{S}ust$

  $$\Phi = \{x_0 \leftarrow suc\,(cero)\,,\ x_1 \leftarrow cero,\ x_2 \leftarrow suc(cero)\}$$

  ii) la regla de reescritura en $\mathcal{R}$

  $$por(x_0,mas(x_1,x_2)) \longrightarrow mas(por(x_0,x_1),por(x_0,x_2))$$

  iii) la posición (2) en t de manera que

  $$\begin{aligned} subterm\,((2),t) &= por\,(suc\,(cero)\,,mas\,(cero,suc(cero))) \\ &= inst(\Phi,por(x_0,mas(x_1,x_2))) \end{aligned}$$

  y

  $$\begin{aligned} s &= mas\,(suc\,(cero)\,,mas(por(suc\,(cero)\,,cero),por(suc\,(cero)\,,suc\,(cero)))) \\ &= repl\,((2),t,inst(\Phi,s)) \end{aligned}$$

  donde

  $$inst(\Phi,s) = mas(por(suc\,(cero)\,,cero),por(suc\,(cero)\,,suc\,(cero)))$$

  En conclusión, s reescribe t al contraer el redex $por\,(suc\,(cero)\,,mas\,(cero,suc(cero)))$.

- La composición de la relación $\Rightarrow_{\mathcal{R}}$define el **proceso de reducción**. Un término $t \in \mathcal{T}(\mathcal{F},\mathcal{X})$ se dice en **forma normal** respecto a $\Rightarrow_{\mathcal{R}}$si no existe término $s \neq t$ tal que $t \Rightarrow_{\mathcal{R}} s$.

Ejemplo 12: el término irreductible $suc\,(suc\,(cero))$ es la forma normal del término t del ejemplo 11. En el proceso de reducción hemos marcado en un cuadro las redexes que son remplazadas en el paso de reducción siguiente;

$$
\begin{aligned}
& mas\left(suc\,(cero)\,,\boxed{por\,(suc\,(cero)\,,mas\,(cero,suc(cero)))}\right)\\
\Rightarrow\ & {}^{(2)}_{9}\ mas\left(suc\,(cero)\,,mas\,(\boxed{por(suc\,(cero)\,,cero)}\,,por\,(suc\,(cero)\,,suc\,(cero)))\right)\\
\Rightarrow\ & {}^{(2,1)}_{5}\ mas\left(suc\,(cero)\,,\boxed{mas\,(cero,por(suc\,(cero)\,,suc\,(cero)))}\right)\\
\Rightarrow\ & {}^{(2)}_{3}\ mas\left(suc\,(cero)\,,\boxed{mas(por(suc\,(cero)\,,suc\,(cero)),cero)}\right)\\
\Rightarrow\ & {}^{(2)}_{1}\ mas\left(suc\,(cero)\,,\boxed{por(suc\,(cero)\,,suc\,(cero))}\right)\\
\Rightarrow\ & {}^{(2)}_{6}\ mas\left(suc\,(cero)\,,mas(suc\,(cero)\,,\boxed{por(suc\,(cero)\,,cero)}\right)\\
\Rightarrow\ & {}^{(2,2)}_{5}\ mas\left(suc\,(cero)\,,\boxed{mas(suc\,(cero)\,,cero)}\right)\\
\Rightarrow\ & {}^{(2)}_{1}\ \boxed{mas\,(suc\,(cero)\,,suc\,(cero))}\\
\Rightarrow\ & {}^{(\lambda)}_{2}\ suc\left(\boxed{mas(suc\,(cero)}\,,cero\right)\\
\Rightarrow\ & {}^{(1)}_{1}\ suc\,(suc\,(cero))
\end{aligned}
$$

- Una **estrategía de reducción** es un criterio fijo de elección para remplazo entre las diferentes redexes en un mismo término. La estrategía de reducción más externa consiste en elegir siempre la redex cuya posición sea menor según el orden de prefijos, o si hay varias redexes cuyas posiciones son incomparables, se remplazan de iquierda a derecha. Es un hecho demostrado que la estrategía de reducción más externa en sistemas de reescritura ortogonales son **normalizantes**, es decir, siempre se alcanza una forma normal.

  Ejemplo 13: El proceso de reducción del ejemplo 12 obedece al criterio de reducción más interna; estrategia opuesta, como se ve en el ejemplo 1, a la más externa. A continuación mostramos el proceso de reducción con

estrategía más externa para el término t del ejemplo 11:

$$\boxed{mas\,(suc\,(cero)\,,por\,(suc\,(cero)\,,mas\,(cero,suc(cero))))}$$
$$\Rightarrow \overset{(\lambda)}{3} \boxed{mas\,(por\,(suc\,(cero)\,,mas\,(cero,suc(cero)))\,,suc\,(cero)\,,)}$$
$$\Rightarrow \overset{(\lambda)}{2} suc\,(\boxed{mas\,(por\,(suc\,(cero)\,,mas\,(cero,suc(cero)))\,,cero)})$$
$$\Rightarrow \overset{(1)}{1} suc\,(\boxed{por\,(suc\,(cero)\,,mas\,(cero,suc(cero)))})$$
$$\Rightarrow \overset{(1)}{7} suc\,(\boxed{por\,(mas\,(cero,suc(cero))\,,suc\,(cero))})$$
$$\Rightarrow \overset{(1)}{6} suc\,(mas\,(\boxed{mas\,(cero,suc(cero))}\,,por(mas\,(cero,suc(cero))\,,cero)))$$
$$\Rightarrow \overset{(1,1)}{3} suc\,(mas\,(mas\,(suc(cero),cero)\,,\boxed{por(mas\,(cero,suc(cero))\,,cero)}))$$
$$\Rightarrow \overset{(1,2)}{5} suc\,(\boxed{mas\,(mas\,(suc(cero),cero)\,,cero)})$$
$$\Rightarrow \overset{(1)}{1} suc\,(\boxed{mas\,(suc(cero),cero)})$$
$$\Rightarrow \overset{(1)}{1} suc\,(suc\,(cero))$$

3 Enunciado del proyecto

Se debe programar en Prolog los procedimientos (definiciones) que computen las funciones descritas:

- $pos(+Term, -Lpos)$ donde $Term$ es un término y Pos es la lista de posiciones.
- $subterm(+Pos, +Term1, Term2)$ donde Pos es una posición, $Term1$ es un término y $Term2$ es el subtérmino en la posición indicada.
- $var(+Term, -Lvar)$ donde $Term$ es un término y $Lvar$ es la lista sin repeticiones de las variables que pcurren en el término.
- $repl(+Term1, +Pos, +Term2, -Term3)$ donde $Term3$ es el resultado de colocar $Term2$ en lugar del subtérmino de $Term1$ en la posición Pos.
- $inst(+Sust, +Term1, -Term2)$ donde $Sust$ es una lista de pares variable-término y $Term2$ es la instancia de $Term1$ según $Sust$.
- $redex(+Term, -Lterm)$ donde $Lterm$ es la lista de redexes externos incomparables de $Term$ (si hay uno solo la lista es unitaria).
- $reduce(+Term1, -Term2)$ donde $Term2$ es el resultado de un paso de reucción en $Term1$.
- $normal(+Term1, -Fnorm)$ donde $Fnorm$ es la forma normal de $Term$.

Documentación a entregar:

- El módulo *haskell.pl* conteniendo el código fuente Prolog que implementa los procedimientos que se piden correctamente documentado.
- **Fecha de Entrega.** Martes 31 de marzo de 2009 (Semana 12).
- **Valor de Evaluación.** Treinta (30) puntos.

References

[1] Lloyd, J. W. (1984)
Foundations of Logic Programming
`Springer-Verlag`

[2] Clocksin, W. F. & Mellish, C. S. (1987)
Programming in Prolog
`Springer-Verlag`

[3] Bratko, I. (1990)
Prolog; Programming for Artificial Intelligence
`Addison-Wesley`

[4] Baader, F. & Nipkow, T. (1998)
Term Rewriting and All That
`Cambridge University Press`