Trabajo Practico
Lenguajes segun sus caracteristicas

Martion Alejandro Melo

26 de octubre de 2015

1. Rehacer las entregas

1.1. Entrega 1

Definir Free Variables(FV)
$FV(x) = x$
$FV(t\ u) = FV(t) \cup FV(u)$
$FV(t \otimes u) = FV(t) \cup FV(u)$
$FV(\mathsf{fun}\ x : t) = FV(t) \backslash x$
$FV(\mathsf{fix}\ x : t) = FV(t) \backslash x$
$FV(\mathsf{Let}\ x = t\ \mathsf{in}\ u) = FV(t) \cup (FV(u) \backslash x)$
$FV(\mathsf{ifZ}\ t\ \mathsf{then}\ u\ \mathsf{else}\ v) = FV(t) \cup FV(u) \cup FV(v)$

1.2. Entrega 2

Extender PCF para disponer de Arboles Binarios

Sintaxis de Tipos:

$A = \mathsf{Nat}\ | A \Rightarrow A | A \wedge A | \mathsf{Tree}\ A | \mathsf{NoLeaf}\ A$

Sintaxis de terminos:

$t = ... | Leaf\ t | Tree\ t\ t\ t | ifLeaf\ t\ \mathsf{then}\ t\ \mathsf{else}\ t | CL\ t | CR\ t | Node\ t$

Semantica Operacional:

$\mathsf{CL\ Tree}\ t\ u\ v \rightarrow u$
$\mathsf{CR\ Tree}\ t\ u\ v \rightarrow v$
$\mathsf{ifLeaf\ Tree}\ t\ u\ v\ \mathsf{then}\ u\ \mathsf{else}\ v \rightarrow v$
$\mathsf{ifLeaf\ Leaf}\ t\ \mathsf{then}\ u\ \mathsf{else}\ v \rightarrow u$
$\mathsf{Node\ Tree}\ t\ u\ v \rightarrow t$
$\mathsf{Node\ Leaf}\ t \rightarrow t$

Relacion definida inductivamente:

$$\frac{\Gamma \vdash t : \mathsf{NoLeaf}\ A}{\Gamma \vdash t : \mathsf{Tree}\ A}$$

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash \mathsf{Leaf}\ t : \mathsf{Tree}\ A}$$

$$\frac{\Gamma \vdash t : \mathsf{NoLeaf}}{\Gamma \vdash \mathsf{CL}\ t : \mathsf{Tree}\ A}$$

$$\frac{\Gamma \vdash t : \mathsf{NoLeaf}}{\Gamma \vdash \mathsf{CR}\ t : \mathsf{Tree}\ A}$$

$$\frac{\Gamma \vdash t : \mathsf{Tree}\ A}{\Gamma \vdash \mathsf{Node}\ t : A}$$

$$\frac{\Gamma \vdash l : \mathsf{Tree}\ A \quad \Gamma \vdash r : \mathsf{Tree}\ A \quad \Gamma \vdash t : A}{\Gamma \vdash \mathsf{Tree}\ t\ r\ l : \mathsf{NoLeaf}\ A}$$

$$\frac{\Gamma \vdash l : B \quad \Gamma \vdash r : B \quad \Gamma \vdash t : \mathsf{Tree}\ A}{\Gamma \vdash \mathsf{ifLeaf}\ t\ r\ l : B}$$

1.3. Entrega 3

Extender Algoritmo de Hindley y Robinson para Arboles Binarios y derivar el tipo para el termino:

Para Hindley:

$$\frac{\Gamma \vdash t \rightsquigarrow A, \tau}{\Gamma \vdash \mathsf{Leaf}\ t \rightsquigarrow A \to \mathsf{Tree}\ A, \tau}$$

$$\frac{\Gamma \vdash t \rightsquigarrow A, \tau}{\Gamma \vdash \mathsf{CL}\ t \rightsquigarrow \mathsf{Tree}\ B, \tau \cup \{A = \mathsf{NoLeaf}\ B\}}$$

$$\frac{\Gamma \vdash t \rightsquigarrow A, \tau}{\Gamma \vdash \mathsf{CR}\ t \rightsquigarrow \mathsf{Tree}\ B, \tau \cup \{A = \mathsf{NoLeaf}\ B\}}$$

$$\frac{\Gamma \vdash t \rightsquigarrow A, \tau}{\Gamma \vdash \mathsf{Node}\ t \rightsquigarrow B, \tau \cup \{A = \mathsf{Tree}\ B\}}$$

$$\frac{\Gamma \vdash t \rightsquigarrow A, \tau \quad \Gamma \vdash u \rightsquigarrow B, \beta \quad \Gamma \vdash v \rightsquigarrow C, \alpha}{\Gamma \vdash \mathsf{Tree}\ t\ u\ v \rightsquigarrow \mathsf{NoLeaf}\ A, \tau \cup \beta \cup \alpha \cup \{B = \mathsf{Tree}\ A, C = \mathsf{Tree}\ A\}}$$

$$\frac{\Gamma \vdash t \rightsquigarrow \mathsf{Tree}\ A, \tau \quad \Gamma \vdash u \rightsquigarrow B, \beta \quad \Gamma \vdash v \rightsquigarrow B, \alpha}{\Gamma \vdash \mathsf{ifLeaf}\ t\ \ \mathsf{then}\ u\ \ \mathsf{else}\ v \rightsquigarrow B, \tau \cup \beta \cup \alpha \cup \{B = C, A = \mathsf{Tree}\ X\}}$$

Para Robinson:
(1) $(A \Rightarrow B = C \Rightarrow D) \Rightarrow A = C \ y \ B = D$.
$\mathsf{Tree}\ A = \mathsf{Tree}\ B \Rightarrow A = B$.
$\mathsf{NoLeaf}\ A = \mathsf{NoLeaf}\ B \Rightarrow A = B$.
(3) Se agrega que reglas del tipo $\mathsf{Tree}\ A = B \Rightarrow C \ y \ \mathsf{NoLeaf}\ A = B \Rightarrow C$ daran error.
Derivar el termino: CL ((fun x.x) (Leaf 2))

$$\dfrac{\dfrac{\dfrac{\overline{\Gamma x : A \vdash x \rightsquigarrow A, \emptyset}}{\vdash \mathsf{fun}\ x.x \rightsquigarrow A \Rightarrow A, \emptyset} \quad \dfrac{\overline{\vdash 2 \rightsquigarrow \mathsf{Nat}\ , \emptyset}}{\vdash \mathsf{Leaf}\ 2 \rightsquigarrow \mathsf{Tree\ Nat}\ , \emptyset}}{\vdash (\mathsf{fun}\ x.x)(\mathsf{Leaf}\ 2) \rightsquigarrow X, \{A \Rightarrow A = \mathsf{Tree\ Nat}\ \Rightarrow X\}}}{\mathsf{CL}\ ((\mathsf{fun}\ x.x)(\mathsf{Leaf}\ 2)) \rightsquigarrow \mathsf{Tree}\ B, \{A \Rightarrow A = \mathsf{Tree\ Nat}\ \Rightarrow X, X = \mathsf{NoLeaf}\ B\}}$$

Robinson

$$\begin{cases} A \Rightarrow A = \mathsf{Tree\ Nat}\ \Rightarrow X \\ X = \mathsf{NoLeaf}\ B \end{cases}$$

$\Rightarrow$ (1)

$$\begin{cases} A = \mathsf{Tree\ Nat}\ ) \\ A = X \\ X = \mathsf{NoLeaf}\ B \end{cases}$$

$\Rightarrow$

$$\begin{cases} \mathsf{Tree\ Nat}\ = \mathsf{Tree\ Nat} \\ \mathsf{Tree}\ A = \mathsf{NoLeaf}\ B \\ \mathsf{NoLeaf}\ B = \mathsf{NoLeaf}\ B \end{cases}$$

$\Rightarrow$

$$\{ \quad \mathsf{Tree}\ A = \mathsf{NoLeaf}\ B$$

ERROR en el ultimo paso $\mathsf{Tree}\ A \neq \mathsf{NoLeaf}\ B$

1.4. Entrega 4

Derivar tipo con Hindley y Robinson a: Node ((fun x.leaf x) (fun y.y)).

$$\dfrac{\dfrac{\dfrac{\overline{x : B \vdash x \rightsquigarrow B, \emptyset}}{\Gamma \vdash \mathsf{fun}\ x.\mathsf{Leaf}\ x \rightsquigarrow B \Rightarrow \mathsf{Tree}\ B, \emptyset} \quad \dfrac{\overline{y : A \vdash y \rightsquigarrow A, \emptyset}}{\Gamma \vdash \mathsf{fun}\ y.y \rightsquigarrow A \Rightarrow A, \emptyset}}{\Gamma \vdash (\mathsf{fun}\ x.\mathsf{Leaf}\ x)(\mathsf{fun}\ y.y) \rightsquigarrow X, \{B \Rightarrow \mathsf{Tree}\ B = (A \Rightarrow A) \Rightarrow X\}}}{\Gamma \vdash \mathsf{Node}\ ((\mathsf{fun}\ x.\mathsf{Leaf}\ x)(\mathsf{fun}\ y.y)) \rightsquigarrow C, \{B \Rightarrow \mathsf{Tree}\ B = (A \Rightarrow A) \Rightarrow X, X = \mathsf{Tree}\ C\}}$$

Robinson

$$\begin{cases} B \Rightarrow \mathsf{Tree}\ B = (A \Rightarrow A) \Rightarrow X \\ \mathrm{X} = \mathsf{Tree}\ \mathrm{C} \end{cases}$$

$\Rightarrow (1)$

$$\begin{cases} B = (A \Rightarrow A) \\ \mathsf{Tree}\ B = X \\ \mathrm{X} = \mathsf{Tree}\ \mathrm{C} \end{cases}$$

$\Rightarrow (1)$

$$\begin{cases} B = (A \Rightarrow A) \\ \mathsf{Tree}\ B = \mathsf{Tree}\ B \\ \mathsf{Tree}\ B = \mathsf{Tree}\ C \end{cases}$$

$\Rightarrow (4)$

$$\begin{cases} B = (A \Rightarrow A) \\ \mathsf{Tree}\ B = \mathsf{Tree}\ C \\ B = C \end{cases}$$

$\Rightarrow (1)$

$$\begin{cases} (A \Rightarrow A) = (A \Rightarrow A) \\ \mathsf{Tree}\ (A \Rightarrow A) = \mathsf{Tree}\ C \\ (A \Rightarrow A) = C \end{cases}$$

$S = [(A \Rightarrow A)/C]$
El tipo es: $(A \Rightarrow A)$

Derivar tipos con polimorfismo a: Fun X.Let y=x in y y.

$$\dfrac{\dfrac{\overline{x : [B] \vdash x : [B]} \quad \dfrac{\overline{x : [B], y : [B] \vdash y : [B \Rightarrow (A \Rightarrow A)]} \quad \overline{x : [B], y : [B] \vdash y : [B]}}{x : [B], y : [B] \vdash yy : [A \Rightarrow A]}}{x : [B] \vdash \mathsf{Let}\ y = x \mathsf{in}\ yy : [A \Rightarrow A]}}{\vdash \mathsf{fun}\ x.\mathsf{Let}\ y = x \mathsf{in}\ yy : [B \Rightarrow (A \Rightarrow A)]}$$

1.5. Entrega 5

$$(\text{fun } x.\text{fun } y.(1+x)+4)((\text{fun } z.z*z)2)$$

CBV fuerte y debil

$$\begin{aligned}(\text{fun } x.\text{fun } y.(1+x)+4)((\text{fun } z.z*z)2) &\to (\text{fun } x.\text{fun } y.(1+x)+4)(2*2)\\ &\to (\text{fun } x.\text{fun } y.(1+x)+4)4\\ &\to \text{fun } y.(1+4)+4 \textit{ fin debil}\\ &\to \text{fun } y.\ 5+4\\ &\to \text{fun } y.\ 9 \textit{ fin fuerte}\end{aligned}$$

CBN fuerte y debil

$$\begin{aligned}(\text{fun } x.\text{fun } y.(1+x)+4)((\text{fun } z.z*z)2) &\to \text{fun } y.(1+((\text{fun } z.z*z)2))+4 \textit{ fin debil}\\ &\to \text{fun } y.(1+(2*2))+4\\ &\to \text{fun } y.(1+4)+4\\ &\to \text{fun } y.\ 5+4\\ &\to \text{fun } y.\ 9 \textit{ fin fuerte}\end{aligned}$$

1.6. Entrega 6

$$(\text{fun } x.x+x)((\text{fun } y.y)3)$$

CBN

$$\dfrac{\dfrac{}{\vdash \text{fun } x.x+x \hookrightarrow < x,x+x,\varnothing >} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\vdash \text{fun } y.y \hookrightarrow < y,y,\varnothing >} \quad \dfrac{}{y =< 3,\varnothing > \vdash 3 \hookrightarrow 3}}{\vdash (\text{fun } y.y)3 \hookrightarrow 3}}{x =< ((\text{fun } y.y)3),\varnothing > \vdash x \hookrightarrow 3} \quad \dfrac{\dfrac{\dfrac{\dfrac{}{\vdash \text{fun } y.y \hookrightarrow < y,y,\varnothing >} \quad \dfrac{}{y =< 3,\varnothing > \vdash 3 \hookrightarrow 3}}{\vdash (\text{fun } y.y)3 \hookrightarrow 3}}{x =< ((\text{fun } y.y)3),\varnothing > \vdash x \hookrightarrow 3}}{x =< ((\text{fun } y.y)3),\varnothing > \vdash x+x \hookrightarrow 6}}{\vdash (\text{fun } x.x+x)((\text{fun } y.y)3) \hookrightarrow 6}$$

1.7. Entrega 7

Agregar la multiplicacion al compilador.
Con lo cual a la definicion que ya teniamos agregariamos la operacion "mul"que multiplica el primero de la pila con el acumulador borrando el primero de la pila y poniendo el resultado en el acumulador.
La secuencia para $(1+2)*3+1$ seria la siguiente: ldi 1;push;ldi 2; add; push; ldi 3; mul; push; ldi 1; add;

2. Responder las siguientes preguntas:

1. Los tipos permiten verficar que un programa sea correcto, esto se lleva a cabo analizando las operaciones y validandolas bajo un conjunto de reglas. Dependiendo del tipo de lenguaje se podran detectar errores de tipos podran ser detectados en la etapa de compilacion, este se da en los lenguajes con tipado estatico que realizan el chequeo de tipos en la etapa de compilacion.

2. Que un lenguaje no sea tipado quiere decir que al momento de escribir el codigo no es necesario dar el tipo de datos que se utiliza en las funciones, parametros, definiciones de variables. Por ejemplo en Javascript las variables se pueden definir como "var nombre= dato"sin dar el tipo de la variable. Si bien el lenguaje es no tipado, no quiere decir que posea inferencia de tipos.

3. La inferencia de tipo es la capacidad que permite al programador obviar la definicion del tipo de las funciones, declaraciones y/o variables. Ya que el sistema de inferencia de tipos los asigna Automaticamente al momento de interpretarlo o compilarlo. Esta capacidad la poseen principal los lenguajes con chequeo estatico de tipos. Los lenguajes mas conocidos con inferencia de tipos son Hakell, OCaml y Scala.

4. El polimorfismo de tipos permite a un lenguaje poder dar la misma interfaz a elementos de distintos tipos. Existen varios tipos de polimorfismo, entre ellos se encuentra el polimorfismo parametrico el cual provee la capacidad a una funcion de recibir por parametro valores de distinto tipos. Por ejemplo una funcion que sume numeros enteros o reales. Tambien se encuentra el polimorfismo ad hoc que permite definir muchas veces la misma funcion para distintos tipos.

5. Una estrategia de evaluacion determinado cuando evaluar los argumentos de la llamada de una funcion. Tambien determina el tipo del valor que le pasa. Todos los lenguajes de programacion utilizan una estrategia de evaluacion. Las principales estrategias son: Call by value: Esta estrategia primero evalua los argumentos y luego evalua la funcion. Call by name: Esta estrategia primero evalua la funcion y luego cuando sea necesario evalua los argumentos.

   Ambas tienen sus ventajas y sus desventajas.

6. Un interprete realiza la traducion a codigo maquina en tiempo de ejecucion, instruccion por instruccion. Esto permite que tanto el debuging del programa interpretado, la programacion y distribucion del programa sea mas flexible. Esta ultima caracteristica se da ya que al ser interpretado lo unico que se necesita es un interprete especifico para cada plataforma.

7. Un compilador transforma el codigo escrito al codigo maquina previo a la ejecucion. Existen muchos tipos de compiladores, sus diferencias estan en las etapas de compilacion, el resultado final, etc. A su vez un compilador puede formar parte de un interprete.

8. Una maquina virtual interpreta un tipo de codigo especifico para el que este destinada y ejecuta sus instrucciones, su mayor ventaja es que permite la portabilidad del programa escrito, ya que el codigo esta destinado a que la maquina virtual lo entienda y no la plataforma donde esta corriendo. Con lo cual el programar se asegura que se va a obtener el mismo resultado/comportamiento en las disintas plataformas(hardware/software). Las maquinas virtuales mas conocidas son la de Java, Javascript,Smalltalks, Scala.