Cálculo Lambda

Segunda parte

Paradigmas (de Lenguajes) de Programación

Departamento de Ciencias de la Computación
Universidad de Buenos Aires

6 de mayo de 2025

Primera extensión

Extensión con **pares**

- $\sigma, \tau ::= \cdots \mid \sigma \times \tau$
- $M, N ::= \cdots \mid \langle M, N \rangle \mid \pi_1(M) \mid \pi_2(M)$
- Definir como macro la función $curry_{\sigma,\tau,\delta}$ que sirve para currificar funciones que reciben pares como argumento.
- Enunciar las nuevas reglas de tipado.
- Extender el conjunto de valores y determinar las nuevas reglas de semántica.
- ¿Qué problema introduce agregar la siguiente regla?

$$\pi_1(\langle M, N \rangle) \longrightarrow M$$

Primera extensión

Extensión con **pares**

- Verificar el siguiente juicio de tipado:
  $\emptyset \vdash \pi_1((\lambda x : \mathit{Nat}.\langle x, \mathrm{True}\rangle)\ 0) : \mathit{Nat}$
- Reducir el siguiente término a un valor:
  $\pi_1((\lambda x : \mathit{Nat}.\langle x, \mathrm{True}\rangle)\ 0)$

Segunda extensión

Extensión con **uniones disjuntas**

$$
\begin{array}{lcl}
\sigma & ::= & \ldots \mid \sigma + \sigma \\
M & ::= & \ldots \mid \mathsf{left}_\sigma(M) \mid \mathsf{right}_\sigma(M) \mid \\
 & & \mathsf{case}\ M\ \mathsf{of}\ \mathsf{left}(x) \rightsquigarrow M \parallel \mathsf{right}(y) \rightsquigarrow M
\end{array}
$$

- $\sigma + \tau$ representa el tipo de la unión disjunta entre σ y τ, similar al tipo Either σ τ de Haskell,
- $\mathsf{left}_\sigma(M)$ y $\mathsf{right}_\sigma(M)$ inyectan un valor en la unión, y
- $\mathsf{case}\ M\ \mathsf{of}\ \mathsf{left}(x) \rightsquigarrow N \parallel \mathsf{right}(y) \rightsquigarrow O$ efectúa un análisis de casos del término M comparándolo con los patrones $\mathsf{left}_\sigma(x)$ y $\mathsf{right}_\sigma(y)$.

Ejercicio

- Marcar subtérminos y anotaciones de tipos.
- Enunciar las nuevas reglas de tipado y extender el conjunto de valores y las reglas de semántica de la nueva extensión.

Tercera extensión

Extensión con **árboles binarios**

- $\sigma ::= \cdots \mid \mathsf{AB}_\sigma$
- $M, N, O ::= \cdots \mid \mathsf{Nil}_\sigma \mid \mathsf{Bin}(M, N, O) \mid \mathsf{raíz}(M) \mid \mathsf{der}(M) \mid \mathsf{izq}(M) \mid \mathsf{esNil}(M)$
- Definir como ejemplo un árbol de funciones Bool $\rightarrow$ Bool.
- Definir las nuevas reglas de tipado.
- Determinar las reglas de semántica y nuevos valores.

Otra forma de proyectar/observar

Otra forma de representar proyectores u observadores más prolija y que requiere menos reglas (aunque una construcción más sofisticada):

Árboles bis

- $M, N, O ::= \cdots \mid Nil_\sigma \mid \mathrm{Bin}(M, N, O) \mid$
  $\text{case } M \text{ } of \text{ } Nil \leadsto N \text{ ; } \mathrm{Bin}(i, r, d) \leadsto O$

  Importante: las minúsculas i, r y d representan los nombres de las variables que pueden aparecer libres en O.
- Marcar subtérminos y anotaciones de tipos.
- Modificar las reglas de tipado para soportar la nueva extensión.
- ¿Se modifica el conjunto de valores?
- Agregar las reglas de semántica necesarias.

Otra forma de proyectar/observar

Árboles binarios bis

- Reducir el siguiente término:
  case if $(\lambda x : Bool.x)$ True then $\text{Bin}(\text{Nil}_{Nat}, \underline{1}, \text{Nil}_{Nat})$ else Nil_{Nat}
  $of\ \ Nil \rightsquigarrow$ False ; $\text{Bin}(i, r, d) \rightsquigarrow \text{iszero}(r)$
- Definir como macros las funciones raíz_σ, der_σ, izq_σ y esNil_σ, que al aplicarlas a un árbol binario emulan los términos de la extensión original.
- Definir una nueva extensión que incorpore expresiones de la forma $map(M, N)$, donde N es un árbol y M una función que se aplicará a cada uno de los elementos de N.