

Cálculo Lambda

Segunda parte

Paradigmas (de Lenguajes) de Programación

Departamento de Ciencias de la Computación
Universidad de Buenos Aires

30 de septiembre de 2025

Extensión con pares

- $\sigma, \tau ::= \dots \mid \sigma \times \tau$
- $M, N ::= \dots \mid \langle M, N \rangle \mid \pi_1(M) \mid \pi_2(M)$
- Definir como macro la función $\text{curry}_{\sigma, \tau, \delta}$ 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?

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

Extensión con pares

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

Segunda extensión

Extensión con uniones disjuntas

$$\begin{aligned}\sigma &::= \dots \mid \sigma + \sigma \\ M &::= \dots \mid \text{left}_\sigma(M) \mid \text{right}_\sigma(M) \mid \\ &\quad \text{case } M \text{ of } \text{left}(x) \rightsquigarrow M \parallel \text{right}(y) \rightsquigarrow M\end{aligned}$$

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

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.

Extensión con **árboles binarios**

- $\sigma ::= \dots \mid \text{AB}_\sigma$
- $M, N, O ::= \dots \mid \text{Nil}_\sigma \mid \text{Bin}(M, N, O) \mid \text{raíz}(M) \mid \text{der}(M) \mid \text{izq}(M) \mid \text{esNil}(M)$
- Definir como ejemplo un árbol de funciones $\text{Bool} \rightarrow \text{Bool}$.
- 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

- $M, N, O ::= \dots \mid Nil_\sigma \mid Bin(M, N, O) \mid$
case M of $Nil \rightsquigarrow N ; Bin(i, r, d) \rightsquigarrow O$

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.

Árboles binarios bis

- Reducir el siguiente término:
$$\text{case if } (\lambda x : \text{Bool}.x) \text{ True then Bin}(\text{Nil}_{\text{Nat}}, \underline{1}, \text{Nil}_{\text{Nat}}) \text{ else Nil}_{\text{Nat}}$$

$$\text{of Nil} \rightsquigarrow \text{False} ; \text{Bin}(i, r, d) \rightsquigarrow \text{iszero}(r)$$
- 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 $\text{map}(M, N)$, donde N es un árbol y M una función que se aplicará a cada uno de los elementos de N .