

Paradigmas de Programación

Cálculo- λ

2do cuatrimestre de 2024

Departamento de Computación

Facultad de Ciencias Exactas y Naturales

Universidad de Buenos Aires

Introducción

Cálculo- λ : sintaxis y tipado

Cálculo- λ : semántica operacional

¿Qué es el cálculo- λ ?

Lenguaje de programación definido de manera rigurosa.

Se basa sólo en dos operaciones: construir funciones y aplicarlas.

Históricamente

- ▶ Concebido en la década de 1930 por Alonzo Church para formalizar la noción de función efectivamente computable.
- ▶ Usado desde la década de 1960 para estudiar semántica formal de lenguajes de programación.

Actualmente

- ▶ Núcleo de lenguajes de programación funcionales y asistentes de demostración.
LISP, OCAML, HASKELL, COQ, AGDA, LEAN,
- ▶ Laboratorio para investigar nuevas características de lenguajes.
- ▶ Fuertemente conectado con la teoría de la demostración, matemática constructiva, teoría de categorías, ...

Introducción

Cálculo- λ : sintaxis y tipado

Cálculo- λ : semántica operacional

El cálculo- λ^b

Sintaxis de los tipos

$$\begin{array}{lcl} \tau, \sigma, \rho, \dots & ::= & \text{bool} \\ & | & \tau \rightarrow \sigma \end{array}$$

Asumimos que el constructor de tipos “ \rightarrow ” es asociativo a derecha:

$$\tau \rightarrow \sigma \rightarrow \rho \quad = \quad \tau \rightarrow (\sigma \rightarrow \rho) \quad \neq \quad (\tau \rightarrow \sigma) \rightarrow \rho$$

El cálculo- λ^b

Suponemos dado un conjunto infinito numerable de variables:

$$\mathcal{X} = \{x, y, z, \dots\}$$

Sintaxis de los términos

M, N, P, \dots	$::=$	x	variable
		$\lambda x : \tau. M$	abstracción
		$M N$	aplicación
		true	verdadero
		false	falso
		if M then N else P	condicional

Asumimos que la aplicación es asociativa a izquierda:

$$M N P = (M N) P \neq M (N P)$$

La abstracción y el “if” tienen menor precedencia que la aplicación:

$$\lambda x : \tau. M N = \lambda x : \tau. (M N) \neq (\lambda x : \tau. M) N$$

El cálculo- λ^b

Ejemplos de términos

- ▶ $\lambda x : \text{bool}. x$
- ▶ $\lambda x : \text{bool} \rightarrow \text{bool}. x$
- ▶ $(\lambda x : \text{bool}. x) \text{ false}$
- ▶ $(\lambda x : \text{bool} \rightarrow \text{bool}. x) (\lambda y : \text{bool}. y)$
- ▶ $(\lambda x : \text{bool}. \lambda y : \text{bool} \rightarrow \text{bool}. y x) \text{ true}$
- ▶ $\lambda x : \text{bool}. \text{if } x \text{ then false else true}$
- ▶ true true
- ▶ $\text{if } \lambda x : \text{bool}. x \text{ then false else true}$

Variables libres y ligadas

Una ocurrencia de x está **ligada** si aparece adentro de una abstracción “ λx ”. Una ocurrencia de x está **libre** si no está ligada.

Ejemplo

Marcar ocurrencias de variables libres y ligadas:

$$(\lambda x : \text{bool} \rightarrow \text{bool}. \lambda y : \text{bool}. x y) (\lambda y : \text{bool}. x y) y$$

Ejercicio

Definir el conjunto de variables libres $\text{fv}(M)$ de M .

Alfa equivalencia

Los términos que difieren sólo en el nombre de variables *ligadas* se consideran iguales:

$$\begin{aligned} \lambda x : \tau. \lambda y : \sigma. x &= \lambda y : \tau. \lambda x : \sigma. y = \lambda a : \tau. \lambda b : \sigma. a \\ \lambda x : \tau. \lambda y : \sigma. x &\neq \lambda x : \tau. \lambda y : \sigma. y = \lambda x : \tau. \lambda x : \sigma. x \end{aligned}$$

Sistema de tipos

La noción de “tipabilidad” se formaliza con un sistema deductivo.

Problema

¿Qué tipo tiene x ?

Contextos de tipado

Un **contexto de tipado** es un conjunto finito de pares $(x_i : \tau_i)$:

$$\{x_1 : \tau_1, \dots, x_n : \tau_n\}$$

sin variables repetidas ($i \neq j \Rightarrow x_i \neq x_j$).

Se nota con letras griegas mayúsculas (Γ, Δ, \dots) .

Juicios de tipado

El sistema de tipos predica sobre **juicios de tipado**, de la forma:

$$\Gamma \vdash M : \tau$$

Sistema de tipos

Reglas de tipado

$$\frac{}{\Gamma \vdash \text{true} : \text{bool}} \text{T-TRUE}$$

$$\frac{}{\Gamma \vdash \text{false} : \text{bool}} \text{T-FALSE}$$

$$\frac{\Gamma \vdash M : \text{bool} \quad \Gamma \vdash N : \tau \quad \Gamma \vdash P : \tau}{\Gamma \vdash \text{if } M \text{ then } N \text{ else } P : \tau} \text{T-IF}$$

$$\frac{}{\Gamma, x : \tau \vdash x : \tau} \text{T-VAR}$$

$$\frac{\Gamma, x : \tau \vdash M : \sigma}{\Gamma \vdash \lambda x : \tau. M : \tau \rightarrow \sigma} \text{T-ABS}$$

$$\frac{\Gamma \vdash M : \tau \rightarrow \sigma \quad \Gamma \vdash N : \tau}{\Gamma \vdash MN : \tau} \text{T-APP}$$

Sistema de tipos

Ejemplo — derivaciones de juicios de tipado

Derivar, si es posible, juicios de tipado para los siguientes términos:

1. $\lambda x : \text{bool}. \text{if } x \text{ then false else } x$
2. $\lambda y : \text{bool} \rightarrow \text{bool} \rightarrow \text{bool}. \lambda z : \text{bool}. y (y x z)$
3. $x y (x z)$
4. $\text{true} (\lambda x : \text{bool}. x)$
5. $x x$

Propiedades del sistema de tipos

Teorema (Unicidad de tipos)

Si $\Gamma \vdash M : \tau$ y $\Gamma \vdash M : \sigma$ son derivables, entonces $\tau = \sigma$.

Teorema (Weakening + Strengthening)

Si $\Gamma \vdash M : \tau$ es derivable y $\text{fv}(M) \subseteq \Gamma \cap \Gamma'$ entonces $\Gamma' \vdash M : \tau$ es derivable.

Introducción

Cálculo- λ : sintaxis y tipado

Cálculo- λ : semántica operacional

Semántica formal

El sistema de tipos indica cómo se construyen los programas.
Queremos además darles **significado** (semántica).

Distintas maneras de dar semántica formal

1. **Semántica operacional.**

Indica cómo se ejecuta el programa hasta llegar a un resultado.

Semántica *small-step*: ejecución paso a paso.

Semántica *big-step*: evaluación directa al resultado.

2. **Semántica denotacional.**

Interpreta los programas como objetos matemáticos.

3. **Semántica axiomática.**

Establece relaciones lógicas entre el estado del programa antes y después de la ejecución.

4. ...

Vamos a trabajar con semántica operacional *small-step*.

Semántica operacional *small-step*

Programas

Un **programa** es un término M tipable y *cerrado* ($\text{fv}(M) = \emptyset$):

- El juicio de tipado $\vdash M : \tau$ debe ser derivable para algún τ .

Juicios de evaluación

La semántica operacional predica sobre **juicios de evaluación**:

$$M \rightarrow N$$

donde M y N son programas.

Valores

Los **valores** son los posibles resultados de evaluar programas:

$$V ::= \text{true} \mid \text{false} \mid \lambda x : \tau. M$$

Semántica operacional *small-step*

Reglas de evaluación para expresiones booleanas

$$\frac{}{\text{if true then } M \text{ else } N \rightarrow M} \text{E-IFTRUE}$$

$$\frac{}{\text{if false then } M \text{ else } N \rightarrow N} \text{E-IFFALSE}$$

$$\frac{M \rightarrow M'}{\text{if } M \text{ then } N \text{ else } P \rightarrow \text{if } M' \text{ then } N \text{ else } P} \text{E-IF}$$

Semántica operacional *small-step*

Ejemplo

1. Derivar el siguiente juicio:

if (if false then false else true) then false else true
→ if true then false else true

2. ¿Para qué términos M vale que $\text{true} \rightarrow M$?
3. ¿Es posible derivar el siguiente juicio?

if true then (if false then false else false) else true
→ if true then false else true

Semántica operacional *small-step*

Reglas de evaluación para funciones (abstracción y aplicación)

$$\frac{M \rightarrow M'}{M N \rightarrow M' N} \text{E-APP1}$$

$$\frac{N \rightarrow N'}{(\lambda x : \tau. M) N \rightarrow (\lambda x : \tau. M) N'} \text{E-APP2}$$

$$\frac{}{(\lambda x : \tau. M) V \rightarrow M\{x := V\}} \text{E-APPAbs}$$

Sustitución

La operación de **sustitución**:

$$M\{x := N\}$$

denota el término que resulta de reemplazar todas las ocurrencias libres de x en M por N .

Sustitución

Definición de sustitución

$$\begin{aligned}x\{x := N\} &\stackrel{\text{def}}{=} N \\a\{x := N\} &\stackrel{\text{def}}{=} a \text{ si } a \in \{\text{true}, \text{false}\} \cup \mathcal{X} \setminus \{x\} \\(\text{if } M \text{ then } P \text{ else } Q)\{x := N\} &\stackrel{\text{def}}{=} \begin{array}{l} \text{if } M\{x := N\} \\ \text{then } P\{x := N\} \\ \text{else } Q\{x := N\} \end{array} \\(M_1 M_2)\{x := N\} &\stackrel{\text{def}}{=} M_1\{x := N\} M_2\{x := N\} \\(\lambda y : \tau. M)\{x := N\} &\stackrel{\text{def}}{=} \begin{cases} \lambda y : \tau. M & \text{si } x = y \\ \lambda y : \tau. M\{x := N\} & \text{si } x \neq y, y \notin \text{fv}(N) \\ \lambda z : \tau. M\{y := z\}\{x := N\} & \text{si } x \neq y, y \in \text{fv}(N), \\ & z \notin \{x, y\} \cup \text{fv}(M) \cup \text{fv}(N) \end{cases}\end{aligned}$$

Sustitución

Definición de sustitución (alternativa)

$$\begin{aligned}x\{x := N\} &\stackrel{\text{def}}{=} N \\a\{x := N\} &\stackrel{\text{def}}{=} a \text{ si } a \in \{\text{true}, \text{false}\} \cup \mathcal{X} \setminus \{x\} \\(\text{if } M \text{ then } P \text{ else } Q)\{x := N\} &\stackrel{\text{def}}{=} \begin{aligned} &\text{if } M\{x := N\} \\ &\text{then } P\{x := N\} \\ &\text{else } Q\{x := N\} \end{aligned} \\(M_1 M_2)\{x := N\} &\stackrel{\text{def}}{=} M_1\{x := N\} M_2\{x := N\} \\(\lambda y : \tau. M)\{x := N\} &\stackrel{\text{def}}{=} \lambda y : \tau. M\{x := N\} \\ &\quad \text{asumiendo } y \notin \{x\} \cup \text{fv}(N)\end{aligned}$$

La asunción se puede cumplir siempre, renombrando la variable ligada y en caso de conflicto.

Semántica operacional *small-step*

Ejemplo — evaluación

Reducir repetidamente el siguiente término hasta llegar a un valor:

$$(\lambda x : \text{bool}. \lambda f : \text{bool} \rightarrow \text{bool}. f (f x)) \text{true} (\lambda x : \text{bool}. x)$$

Propiedades de la evaluación

Teorema (Determinismo)

Si $M \rightarrow N_1$ y $M \rightarrow N_2$ entonces $N_1 = N_2$.

Teorema (Preservación de tipos)

Si $\vdash M : \tau$ y $M \rightarrow N$ entonces $\vdash N : \tau$.

Teorema (Progreso)

Si $\vdash M : \tau$ entonces:

1. O bien M es un valor.
2. O bien existe N tal que $M \rightarrow N$.

Teorema (Terminación)

Si $\vdash M : \tau$, entonces no hay una cadena infinita de pasos:

$$M \rightarrow M_1 \rightarrow M_2 \rightarrow \dots$$

Propiedades de la evaluación

Corolario (Canonicidad)

1. Si $\vdash M : \text{bool}$ es derivable, entonces la evaluación de M termina y el resultado es true o false.
2. Si $\vdash M : \tau \rightarrow \sigma$ es derivable, entonces la evaluación de M termina y el resultado es una abstracción.

Slogan

Well typed programs cannot go wrong.

(Robin Milner)

! ! ! ! ! ! ! ! ! ? ? ? ? ? ? ? ?