Paradigmas de Programación

Cálculo-λ

1er. cuatrimestre de 2025

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Introducción

Cálculo-λ^b: sintaxis y tipado

Cálculo-λ^b: semántica operacional

Cálculo-λ^{bn}: extensión con números naturales

¿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^b: sintaxis y tipado

Cálculo-λ^b: semántica operacional

Cálculo-λ^{bn}: extensión con números naturales

El cálculo-λ^b

Sintaxis de los tipos

$$\begin{array}{rcl} \tau, \sigma, \rho, \ldots & ::= & \mathsf{bool} \\ & \mid & \tau \to \sigma \end{array}$$

Asumimos que el constructor de tipos "$\to$" es asociativo a derecha:

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

El cálculo-λ^b

Suponemos dado un conjunto infinito numerable de variables:

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

Sintaxis de los términos

$$
\begin{array}{rcll}
M, N, P, \ldots & ::= & x & \text{variable} \\
& | & \lambda x : \tau.\, M & \text{abstracción} \\
& | & M\, N & \text{aplicación} \\
& | & \mathsf{true} & \text{verdadero} \\
& | & \mathsf{false} & \text{falso} \\
& | & \mathsf{if}\ M\ \mathsf{then}\ N\ \mathsf{else}\ P & \text{condicional}
\end{array}
$$

Asumimos que la aplicación es asociativa a izquierda:

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

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

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

El cálculo-λ^b

Ejemplos de términos

- $\lambda x : \mathsf{bool}.\, x$
- $\lambda x : \mathsf{bool} \to \mathsf{bool}.\, x$
- $(\lambda x : \mathsf{bool}.\, x)\, \mathsf{false}$
- $(\lambda x : \mathsf{bool} \to \mathsf{bool}.\, x)\, (\lambda y : \mathsf{bool}.\, y)$
- $(\lambda x : \mathsf{bool}.\, \lambda y : \mathsf{bool} \to \mathsf{bool}.\, y\, x)\, \mathsf{true}$
- $\lambda x : \mathsf{bool}.\, \mathsf{if}\ x\ \mathsf{then}\, \mathsf{false}\ \mathsf{else}\ \mathsf{true}$
- $\mathsf{true}\, \mathsf{true}$
- $\mathsf{if}\ \lambda x : \mathsf{bool}.\, x\ \mathsf{then}\, \mathsf{false}\ \mathsf{else}\ \mathsf{true}$

Variables libres y ligadas

Una ocurrencia de x está **ligada** si aparece adentro de una abstracción "$\lambda\, x$". Una ocurrencia de x está **libre** si no está ligada.

Ejemplo

Marcar ocurrencias de variables libres y ligadas:

$$(\lambda x : \mathsf{bool} \to \mathsf{bool}.\, \lambda y : \mathsf{bool}.\, x\, y)\,(\lambda y : \mathsf{bool}.\, x\, y)\, y$$

Ejercicio

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

Variables libres: Definición formal

$$
\begin{aligned}
\mathsf{fv}(x) &\stackrel{\text{def}}{=} \{x\} \\
\mathsf{fv}(\mathsf{true}) = \mathsf{fv}(\mathsf{false}) &\stackrel{\text{def}}{=} \emptyset \\
\mathsf{fv}(\mathsf{if}\ M\ \mathsf{then}\ N\ \mathsf{else}\ P) &\stackrel{\text{def}}{=} \mathsf{fv}(M) \cup \mathsf{fv}(N) \cup \mathsf{fv}(P) \\
\mathsf{fv}(M\ N) &\stackrel{\text{def}}{=} \mathsf{fv}(M) \cup \mathsf{fv}(N) \\
\mathsf{fv}(\lambda x : \tau.\ M) &\stackrel{\text{def}}{=} \mathsf{fv}(M) \setminus \{x\}
\end{aligned}
$$

α-equivalencia

- Dos términos M y N que difieren solamente en el nombre de sus variables ligadas se dicen α-*equivalentes* (relación $=_\alpha$)
- α-equivalencia es una relación de equivalencia
- **Ojo:** De aquí en más haremos abuso de notación y usaremos el operador $=$ para denotar las α-*equivalencias* (ojo).

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

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 $(x_i : \tau_i)$:

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

sin variables repetidas ($i \neq j \Rightarrow x_i \neq x_j$).
Se nota con letras griegas mayúsculas ($\Gamma, \Delta, \ldots$).
A veces notamos $\mathrm{dom}(\Gamma) = \{x_1, \ldots, x_n\}$.

Juicios de tipado

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

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

Sistema de tipos

Reglas de tipado

$$\frac{}{\Gamma \vdash \text{true} : \text{bool}}\,\text{T-TRUE} \qquad \frac{}{\Gamma \vdash \text{false} : \text{bool}}\,\text{T-FALSE}$$

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

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

$$\frac{\Gamma \vdash M : \tau \to \sigma \qquad \Gamma \vdash N : \tau}{\Gamma \vdash M\, N : \sigma}\,\text{T-APP}$$

Sistema de tipos

Ejemplo — derivaciones de juicios de tipado

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

1. $\lambda x : \mathsf{bool}.\,\mathsf{if}\ x\ \mathsf{then}\,\mathsf{false}\ \mathsf{else}\ x$
2. $\lambda y : \mathsf{bool} \to \mathsf{bool} \to \mathsf{bool}.\, \lambda z : \mathsf{bool}.\, y\,(y\,x\,z)$
3. $x\,z\,(y\,z)$
4. $\lambda x : \mathsf{bool} \to \mathsf{bool}.\, \lambda x : \mathsf{bool}.\, x$
5. $\mathsf{true}\,(\lambda x : \mathsf{bool}.\, x)$
6. $x\,x$

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 $\mathrm{fv}(M) \subseteq \mathrm{dom}(\Gamma \cap \Gamma')$ entonces $\Gamma' \vdash M : \tau$ es derivable.

Introducción

Cálculo-λ^b: sintaxis y tipado

Cálculo-λ^b: semántica operacional

Cálculo-λ^{bn}: extensión con números naturales

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* ($\mathsf{fv}(M) = \emptyset$):

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

Juicios de evaluación

La semántica operacional hace afirmaciones sobre **juicios de evaluación**:

$$M \to N$$

donde M y N son programas.

Valores

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

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

Semántica operacional *small-step*

Reglas de evaluación para expresiones booleanas

$$\frac{}{\text{if true then } M \text{ else } N \to M}\,\textsc{e-ifTrue}$$

$$\frac{}{\text{if false then } M \text{ else } N \to N}\,\textsc{e-ifFalse}$$

$$\frac{M \to M'}{\text{if } M \text{ then } N \text{ else } P \to \text{if } M' \text{ then } N \text{ else } P}\,\textsc{e-if}$$

Semántica operacional *small-step*

Ejemplo

1. Derivar el siguiente juicio:

$$\begin{array}{rl} & \text{if (if false then false else true) then false else true} \\ \to & \text{if true then false else true} \end{array}$$

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

$$\begin{array}{rl} & \text{if true then (if false then false else false) else true} \\ \to & \text{if true then false else true} \end{array}$$

Semántica operacional *small-step*

Reglas de evaluación para funciones (abstracción y aplicación)

$$\frac{M \to M'}{M\,N \to M'\,N}\;\textsc{e-app}1$$

$$\frac{N \to N'}{(\lambda x : \tau.\, M)\,N \to (\lambda x : \tau.\, M)\,N'}\;\textsc{e-app}2$$

$$\frac{}{(\lambda x : \tau.\, M)\,V \to M\{x := V\}}\;\textsc{e-appAbs}$$

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

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

$$
\begin{cases}
\lambda y : \tau.\, M & \text{si } x = y \\
\lambda y : \tau.\, M\{x := N\} & \text{si } x \neq y,\ y \notin \text{fv}(N) \\
\lambda z : \tau.\, M\{y := z\}\{x := N\} & \text{si } x \neq y,\ y \in \text{fv}(N), \\
 & \ z \notin \{x, y\} \cup \text{fv}(M) \cup \text{fv}(N)
\end{cases}
$$

Sustitución

Definición de sustitución (alternativa)

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

La suposición 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:

$$(\lambda x : \mathsf{bool}.\, \lambda f : \mathsf{bool} \to \mathsf{bool}.\, f\,(f\,x))\,\mathsf{true}\,(\lambda x : \mathsf{bool}.\, x)$$

Propiedades de la evaluación

Teorema (Determinismo)

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

Teorema (Preservación de tipos)

Si $\vdash M : \tau$ y $M \to N$ 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 $M \to N$.

Teorema (Terminación)

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

$$M \to M_1 \to M_2 \to \dots$$

Propiedades de la evaluación

Corolario (Canonicidad)

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

Slogan

Well typed programs cannot go wrong. (Robin Milner)

Propiedades de la evaluación

Forma normal

Una forma normal es un término que no puede evaluarse más (i.e., M tal que no existe N, $M \to N$).

Lema

Todo valor está en forma normal.

Estado de error

Estado de la evaluación, donde el término **está en forma normal**, pero no es un valor.
Representa estado en el cual el sistema de *runtime* en una implementación real generaría una excepción.
Recordar que un valor es el resultado al que puede evaluar un término bien-tipado y cerrado.

Evaluación en muchos pasos

Juicio en muchos pasos

La evaluación en muchos pasos $\twoheadrightarrow$ (también denotado $\rightarrow^*$) es la clausura reflexiva-transitiva de $\rightarrow$.
Es decir, es la menor relación tal que

1. Si $M \rightarrow M'$, entonces $M \twoheadrightarrow M'$
2. $M \twoheadrightarrow M$ para todo M
3. Si $M \twoheadrightarrow M'$ y $M' \twoheadrightarrow M''$, entonces $M \twoheadrightarrow M''$

$$
\begin{array}{ll}
 & \mathit{if}\ \mathsf{true}\ \mathit{then}\ (\mathit{if}\ \mathsf{false}\ \mathit{then}\ \mathsf{false}\ \mathit{else}\ \mathsf{true})\ \mathit{else}\ \mathsf{true} \\
\twoheadrightarrow & \mathsf{true}
\end{array}
$$

Para el cálculo de expresiones booleanas valen:

Lema (Unicidad de formas normales)

Si $M \twoheadrightarrow U$ y $M \twoheadrightarrow V$ con U, V formas normales, entonces $U = V$

Lema (Terminación)

Para todo M existe una forma normal N tal que $M \twoheadrightarrow N$

Introducción

Cálculo-λ^b: sintaxis y tipado

Cálculo-λ^b: semántica operacional

Cálculo-λ^{bn}: extensión con números naturales

El cálculo λ^{bn}

Sintaxis: tipos

$$\tau, \sigma, \ldots \quad ::= \quad \ldots \mid \mathsf{nat}$$

Sintaxis: términos

			Semántica informal
M	$::=$	$\ldots$	
	$\mid$	zero	el número cero
	$\mid$	succ(M)	el sucesor del número que representa M
	$\mid$	pred(M)	el predecesor del número que representa M
	$\mid$	isZero(M)	representa un booleano true/false, dependiendo de si M representa al cero o no

El cálculo λ^{bn}: reglas de tipado

$$\frac{}{\Gamma \vdash \mathsf{zero} : \mathsf{nat}}\textsc{T-Zero}$$

$$\frac{\Gamma \vdash M : \mathsf{nat}}{\Gamma \vdash \mathsf{succ}(M) : \mathsf{nat}}\textsc{T-Succ} \qquad \frac{\Gamma \vdash M : \mathsf{nat}}{\Gamma \vdash \mathsf{pred}(M) : \mathsf{nat}}\textsc{T-Pred}$$

$$\frac{\Gamma \vdash M : \mathsf{nat}}{\Gamma \vdash \mathsf{isZero}(M) : \mathsf{bool}}\textsc{T-IsZero}$$

El cálculo λ^{bn}: valores

Extendemos el conjunto de **valores**:

$$V ::= \ldots \mid \mathsf{zero} \mid \mathsf{succ}(V)$$

El cálculo λ^{bn}: semántica operacional

$$\frac{M \to M'}{\mathsf{succ}(M) \to \mathsf{succ}(M')}\,\textsc{e-succ}$$

$$\frac{M \to M'}{\mathsf{pred}(M) \to \mathsf{pred}(M')}\,\textsc{e-pred}$$

$$\frac{}{\mathsf{pred}(\mathsf{succ}(V)) \to V}\,\textsc{e-predSucc}$$

$$\frac{M \to M'}{\mathsf{isZero}(M) \to \mathsf{isZero}(M')}\,\textsc{e-isZero}$$

$$\frac{}{\mathsf{isZero}(\mathsf{zero}) \to \mathsf{true}}\,\textsc{e-isZeroZero}$$

$$\frac{}{\mathsf{isZero}(\mathsf{succ}(V)) \to \mathsf{false}}\,\textsc{e-isZeroSucc}$$

El cálculo λ^{bn}: semántica operacional

Ejemplo

1. Evaluar isZero(succ(pred(succ(zero)))).
2. Evaluar isZero(pred(pred(succ(zero)))). (¿Qué ocurre?)

Forma normal ("f.n.")

Un programa M es una **f.n.** si no existe M' tal que $M \to M'$.

¿Todas las f.n.'s cerradas y tipables son valores?

En el cálculo-λ^{b} **sí**.
En el cálculo-λ^{bn} **no**. (¿Qué propiedad deja de valer?)

Las f.n.'s que no son valores se llaman **términos de error**.

Propiedades de la evaluación

Teorema (Determinismo)

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

Teorema (Preservación de tipos)

Si $\vdash M : \tau$ y $M \to N$ 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 $M \to N$.

Teorema (Terminación)

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

$$M \to M_1 \to M_2 \to \dots$$

¿Todas las f.n.'s cerradas y tipables son valores?

En el cálculo-λ^{b} **sí**.
En el cálculo-λ^{bn} **no**. Deja de valer Progreso. ¿Por qué?

¿Qué podríamos modificar para que sí valga la propiedad?
¿Esto requiere cambiar el lenguaje que estamos modelando?
Si es así, ¿qué es exactamente lo que cambia?

¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ¿ ? ? ? ? ? ? ? ? ?

Lectura recomendada

Capítulos 5 y 9 del libro de Pierce.

Benjamin C. Pierce. *Types and Programming Languages*
The MIT Press, 2002.

Lectura adicional

Capítulos 1 y 3 del libro de Sørensen y Urzyczyn.

Morten Sørensen y Paweł Urzyczyn. *Lectures on the Curry–Howard Isomorphism*. Elsevier, 2006.