Paradigmas de Programación

Cálculo-λ

2do cuatrimestre de 2024
Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

¿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, …

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

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

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

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

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 $\mathsf{fv}(M)$ de M.

Alfa equivalencia

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

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

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 $(\Gamma, \Delta, \ldots)$.

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

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

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

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

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

Sistema de tipos

Ejemplo — derivaciones de juicios de tipado

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

1. $\lambda x : \mathsf{bool}.\, \mathsf{if}\ x\ \mathsf{then}\, \mathsf{false}\ \mathsf{else}\ x$
2. $\lambda y : \mathsf{bool} \to \mathsf{bool} \to \mathsf{bool}.\, \lambda z : \mathsf{bool}.\, y\,(y\,x\,z)$
3. $x\,y\,(x\,z)$
4. $\mathsf{true}\,(\lambda x : \mathsf{bool}.\, x)$
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 $\mathsf{fv}(M) \subseteq \Gamma \cap \Gamma'$ entonces $\Gamma' \vdash M : \tau$ es derivable.

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 predica 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:

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

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

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

Semántica operacional *small-step*

Reglas de evaluación para funciones (abstracción y aplicación)

$$\frac{M \to M'}{M\,N \to M'\,N}\textsc{E-App1}$$

$$\frac{N \to N'}{(\lambda x : \tau.\, M)\, N \to (\lambda x : \tau.\, M)\, N'}\textsc{E-App2}$$

$$\frac{}{(\lambda x : \tau.\, M)\, V \to M\{x := V\}}\textsc{E-AppAbs}$$

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 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:

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

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 $\vdash M : \text{bool}$ 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)