λ-calcolo

Modello computazionale orientato alla produzione di valori sulla base di applicazioni di funzione tramite legami e sostituzioni di variabili.

Sintassi

$$\begin{array}{rcll} e & ::= & x & \textit{variabile} \\ & | & (\lambda x\,.\,e) & \textit{astrazione} \\ & | & (e\ e) & \textit{applicazione} \end{array}$$

o, con regole di inferenza:

$$\frac{x \in Var}{x \in Exp} \qquad \frac{x \in Var, e \in Exp}{(\lambda x.e) \in Exp} \qquad \frac{e_1 \in Exp, e_2 \in Exp}{(e_1\ e_2) \in Exp}$$

Semplifichiamo la scrittura eliminando le parentesi quando non necessarie seguendo le seguenti regole:

- il corpo del costruttore λ si estende il più a destra possibile;
- l'applicazione associa a sinistra.

Variabili libere

Una variabile è libera se compare in un punto in cui non è legata da una λ-astrazione.

$$\begin{aligned} \mathrm{FV}(x) &= \{x\} \\ \mathrm{FV}(\lambda x.e) &= \mathrm{FV}(e) \setminus \{x\} \\ \mathrm{FV}(e_1 e_2) &= \mathrm{FV}(e_1) \cup \mathrm{FV}(e_2) \end{aligned}$$

Un'espressione chiusa è detta *combinatore*.

Funzioni di ordine superiore

Nel λ-calcolo le funzioni sono valori di prima classe. Questo permette di implementare il passaggio di più parametri tramite currying, e di codificare valori come numeri, booleani e coppie.

Valutazione

La macchina astratta del λ-calcolo è definita dalla β-riduzione. L'idea è simile a quella di un compilatore JIT: ogni passo $e_1 \to e_2$ prende in input un programma (espressione) e_1 e ne restituisce un'altro, finché non si giunge ad un valore in forma normale β (se la valutazione termina).

Scoping statico

$$(\lambda x.x(\lambda x.x))z \equiv_\alpha (\lambda x.x(\lambda y.y))$$

Possiamo definire variabili locali tramite λ-astrazioni e applicazioni.
`let x = e1 in e2` diventa $(\lambda x.e_2)e_1$.