

LAMBDA CALCOLO

λ -calcolo

Formalismo alla base della programmazione funzionale. Si possono solo definire funzioni. È Turing completo, puo' rappresentare qualsiasi costrutto dei linguaggi.

Sintassi

$$e ::= x \quad \text{con } x \in V = \{x, y, z\}$$

$$\quad | \quad \lambda x e$$

$$\quad | \quad e e$$

λ -astrazione

Funzione anonima del λ -calcolo: $\lambda x . e$

Dove λ introduce la funzione

x argomento (parametro formale)

"." separa variabile dal corpo

e corpo

es. tramite "ee" si applica un argomento al corpo

Variabili libere (FV)

Variabili non legate da alcun costruttore λ . L'insieme delle $FV(e)$ è definito ricorsivamente sulla struttura di e con le seguenti regole:

- $FV(x) = \{x\}$
 - $FV(e_1 e_2) = FV(e_1) \cup FV(e_2)$
 - $FV(\lambda x . e) = FV(e) \setminus \{x\}$
- } - scende ricorsivamente in e
 - accumula le variabili:
 - quando risale elimina da FV le variabili con λ -astrazione

$$\begin{aligned} \text{es. } FV(\lambda x . x z) &= FV(xz) / \{x\} \\ &= FV(x) \cup FV(z) \setminus \{x\} \\ &= \{x\} \cup \{z\} \setminus \{x\} \\ &= \{z\} \end{aligned}$$

α -equivаленция

$e^1 \equiv_\alpha e^2$ se un'espressione puo' essere ottenuta dall'altra rinominando una o più variabili legate (differisce solo il nome delle variabili legate)

$$\begin{aligned} \text{es. } \lambda x . \lambda y . xy z &\equiv_\alpha \lambda y . \lambda x . xy z \\ \lambda x . xy &\not\equiv_\alpha \lambda y . yy \end{aligned}$$

B-riduzione

Operazione di applicazione funzionale del λ -calcolo.

Redex

Sott'espressione su cui si esegue la B-riduzione e che viene sostituita dal risultato.

Ovvero, la struttura $(\lambda x . e_1)e_2$ da individuare nella espressione per B-ridurre

Regole di inferenza

$$\cdot (\lambda x . e_1)e_2 \longrightarrow e_1[x := e_2]$$

$$\cdot \frac{e_1 \longrightarrow e}{e_1 e_2 \longrightarrow e e_2}$$

$$\cdot \frac{e_2 \longrightarrow e'}{e_1 e_2 \longrightarrow e_1 e'}$$

$$\cdot \frac{e \longrightarrow e'}{\lambda x . e \longrightarrow \lambda x . e'}$$

$$\text{es. } (\underbrace{\lambda x . x}_{\text{funzione}})(\lambda y . y)z \longrightarrow (\lambda y . y)z \longrightarrow z$$

$$\text{idennità} \longrightarrow (\lambda x . x)(\lambda y . y) = (\lambda y . y)$$

input output valore passato

Capture avoiding substitution (CAS)

Evita che le variabili libere vengano catturate. La CAS di x in e_1 con e_2 denotato con $e_1\{x:=e_2\}$

$$\cdot x\{x:=e\} = e$$

$$\cdot y\{x:=e\} = y \text{ se } x \neq y$$

applicando $\{x:=e\}$ ad ogni espressione costituita da una sola variabile si sostituisce solo se quella variabile è proprio quella da sostituire, altrimenti rimane uguale

$$\cdot e_1 e_2 \{x:=e\} = (e_1\{x:=e\})(e_2\{x:=e\}) \rightarrow \text{se l'espressione è un'applicazione di funzione "e", si propaga sulle singole sottoespressioni}$$

$$\cdot (\lambda y. e_1)\{x:=e\} = \lambda y (e_1\{x:=e\})$$

se $y \neq x$ e $y \notin FV(e)$

y è un parametro formale, non coincide con la variabile da sostituire e non appartiene alle variabili libere di e . Si procede con la sostituzione del corpo nella λ -astrazione

$$\cdot (\lambda y. e_1)\{x:=e\} = \lambda z. ((e_1\{y:=z\})\{x:=e\})$$

se $y \neq x$, $y \in FV(e)$ e z fresca (non usata)

y parametro formale e variabile libera di e . Prima di applicare $\{x:=e\}$ si deve rinominare y usando la variabile fresca z . Si ottiene un'espressione \equiv_a .

$$\cdot (\lambda x. e_1)\{x:=e\} = \lambda x. e_1 \rightarrow x \text{ è parametro formale e coincide con la variabile da sostituire. Si arresta e non sostituisce perché si può sostituire solo le variabili libere.}$$

Forma normale di B

Espressione non ulteriormente riducibile. Si arriva a questa forma con una valutazione completa di una λ -espressione, ovvero con una sequenza di B-riduzioni. Non puo' contenere Redex, quindi sara' in due forme:

astrazione funzionale

espressione contenente variabili libere che ostacolano la riduzione.
es. $\lambda x. x$, $\lambda x. xy$, x , xy

B-equivalenza

$e_1 \equiv_b e_2$ (le espressioni sono b-equivalenti) se:

$$\cdot e_1 \equiv_b e_2$$

$$\cdot e_1 \rightarrow e_1' \equiv_b e_2 \text{ oppure } e_2 \rightarrow e_2' \equiv_b e_1 \quad \left. \begin{array}{l} \text{"\rightarrow rappresenta una sequenza} \\ \text{di riduzioni:} \end{array} \right\}$$

$$\text{es. 1) } (\lambda x. x)z \equiv_b (\lambda x. \lambda y. x)zw$$

$$2) (\lambda x. x) \equiv_b (\lambda y. y)$$

$$3) (\lambda x. \lambda y. xy)(\lambda x. x)z \equiv_b (\lambda k. k)z$$

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

$$\begin{aligned} * &= (\lambda x. \lambda y. \lambda z) (\lambda x. x) z \\ &\text{Redex } e_1\{x:=ez\} \\ &\text{Inserisco } (\lambda x. x) \text{ al posto della } x \end{aligned}$$

Teorema di Church Rosser

Proprietà di confluenza, le valutazioni confluiscono nello stesso stato finale.

Ovvero, l'ordine delle B-riduzioni non cambia il risultato. E' comunque possibile che le λ -riduzioni siano non terminanti.

Combinatore Ω

Esempio d: λ -espressione non terminante, definita come:

$$\Omega = (\lambda x. xx)(\lambda x. xx) \xrightarrow{\text{B-riduzione}} (\lambda x. xx)(\lambda x. xx) \rightarrow \text{loop}$$

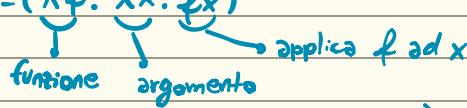
...

Può succedere che, in base al Redex scelto, l'espressione possa terminare con:

- $(\lambda x. k) \Omega \rightarrow k$
- $(\lambda x. k) \Omega \rightarrow (\lambda x. k) \Omega \rightarrow k$
- $(\lambda x. k) \Omega \rightarrow (\lambda x. k) \Omega \rightarrow \dots (\lambda x. k) \Omega \rightarrow \dots$

Funzioni d'
ordine superiore

Sono funzioni che accettano altre funzioni come argomento o risultato
es. La funzione $\text{APPLY} = (\lambda f. \lambda x. fx)$



Se argomento è k e funzione è identità $\text{Id} = (\lambda x. x)$
 $\Rightarrow \text{APPLY Id } k \rightarrow (\lambda x. \text{Id } x)k \rightarrow \text{Id } k \rightarrow k$

Funzioni Curried

Funzioni con più parametri. Possono essere applicate parzialmente.

es. Funzione somma che prende e somma 2 numeri.

$$\text{SOMMA} = (\lambda x. \lambda y. x + y) \rightarrow \text{SOMMA } 10 \ 5 \rightarrow \lambda y. 10 + y \rightarrow 10 + 5 = 15$$

Se si applica parzialmente si ottiene $\text{SOMMA } 10 \rightarrow \lambda y. 10 + y$ ovvero una funzione pronta a sommare 10 al valore passato.

Combinatore Y
(Ricorsione)

Nel λ -calcolo non esiste una definizione nativa per la ricorsione, infatti le funzioni sono anonime (non è possibile richiamarle).

Una funzione può essere resa ricorsiva tramite il combinatore Y

$$Y = \lambda f. (\lambda x. f(xx))(\lambda x. f(xx)) \quad FV(Y) = \emptyset$$

Permette ad una funzione di: richiamare sé stessa.

Soddisfa la proprietà del Punto Fisso $Yf \equiv_{\beta} f(Yf)$ dove f è una qualunque λ -espressione.

Cioè, Y applicato ad una funzione f si riduce all'applicazione di f a Yf .

Y permette quindi di applicare f un numero arbitrario di volte.

Come si effettua
la ricorsione

- 1) Si definisce una λ -astrazione al cui interno si usa un nome f per effettuare tutte le chiamate ricorsive. Di fatto, f rappresenta il corpo della funzione ricorsiva f che si intende definire
- 2) Si definisce la λ -astrazione $G = \lambda f. e$
- 3) Si definisce $F = YG$ (F è la funzione ricorsiva)

es. $e = \lambda y. (fy)y \rightarrow$ applica ricorsione f al parametro y

Definisco la λ -astrazione $G = \lambda f. e$, applico il combinatore $Y: F = YG$

Applicando f a un argomento k , la sua esecuzione sarà la seguente:

$$\begin{aligned} YGk &\Rightarrow G(\lambda x. G(xx))(\lambda x. G(xx))k \equiv_{\beta} (\lambda . ((YG)y)y)k \\ &\Rightarrow (YG)kk \\ &\Rightarrow (YG)k(k)k \end{aligned}$$

Rappresentare i dati
nel λ -calcolo

- $\text{True} = (\lambda t. \lambda f. t)$
- $\text{False} = (\lambda t. \lambda f. f)$
- $\text{If} = \lambda c. \lambda \text{then}. \lambda \text{else}. c \text{ then else}$
- $\text{Co} = \lambda s. \lambda z. z$
- $\text{Cn} = \lambda s. \lambda z. s^h(z)$
- $\text{SUCC} = \lambda n. \lambda s. \lambda z. s(nsz)$
- $\text{PLUS} = \lambda m. \lambda n. \lambda s. \lambda z. ms(nsz) = \lambda m. \lambda n. m \text{ SUCC } n$
- $\text{TIMES} = \lambda m. \lambda n. m (\text{PLUS } n) \text{ Co}$

Passaggio parametri

Criteri per la scelta del redex per ridurre, strategia di valutazione
es. doppia scelta del redex $(\lambda x. xx)((\lambda y. yy)z)$

$\xrightarrow{\text{CBN}}$

$\xrightarrow{\text{CBV}}$

Call by Value (CBV)

Valutazione lazy. Le occorrenze sono sostituite dal valore dell'espressione del parametro formale.

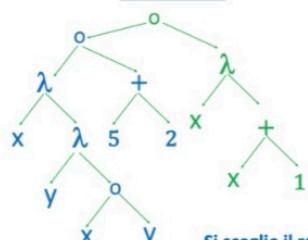
$$(\lambda x. e_1)_v \rightarrow e_1 \{x := v\}$$

$$\frac{e_1 \rightarrow e'}{e_1 e_2 \rightarrow e' e_2} \quad \frac{e_2 \rightarrow e'}{ve_2 \rightarrow ve'}$$

La precedenza deve essere data alla riduzione dell'argomento della funzione prima che all'applicazione. Si sceglie il redex più interno.

$$((\lambda x. (\lambda y. (yx)))(5 + 2)) (\lambda x. x + 1)$$

$$(\lambda x. \lambda y. y x) (5 + 2) \lambda x. x + 1 \rightarrow (\lambda x. \lambda y. y x) 7 \lambda x. x + 1$$



Si sceglie il redex più interno

$$\begin{aligned} &\rightarrow (\lambda y. y 7) \lambda x. x + 1 \\ &\rightarrow (\lambda x. x + 1) 7 \\ &\rightarrow 7 + 1 \\ &\rightarrow 8 \end{aligned}$$

Call by Name (CBN)

Valutazione eager. Le occorrenze sono sostituite dall'espressione non valutata che definisce il parametro attuale.

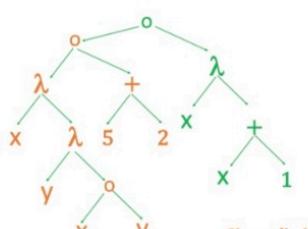
$$(\lambda x. e_1) e_2 \rightarrow e_1 \{x := e_2\}$$

$$\frac{e_1 \rightarrow e'}{e_1 e_2 \rightarrow e' e_2}$$

La precedenza nella scelta tra i diversi redex disponibili deve essere data all'applicazione della funzione che si trova più esternamente

$$((\lambda x. (\lambda y. (yx)))(5 + 2)) (\lambda x. x + 1)$$

$$(\lambda x. \lambda y. y x) (5 + 2) \lambda x. x + 1 \rightarrow (\lambda y. y (5 + 2)) \lambda x. x + 1$$



Si sceglie il redex più esterno

$$\begin{aligned} &\rightarrow (\lambda x. x + 1) (5 + 2) \\ &\rightarrow (5 + 2) + 1 \\ &\rightarrow 7 + 1 \\ &\rightarrow 8 \end{aligned}$$

Differenze B-riduzione

$e \rightarrow e'$
Non esiste la regola $\lambda x. e \rightarrow \lambda x. e'$ che permette di eseguire parte del corpo della funzione prima di applicarla, quindi per esempio $\lambda x((\lambda y. y)z)$ non può essere ridotta, mentre nella B-riduzione diventerebbe $\lambda x. z$.

Questo vuol dire che CBV e CBS non sono equivalenti alla B-riduzione.