



Valutazione Lazy


Aniello Murano
Università degli Studi di Napoli
"Federico II"

Murano Aniello
Fond. LP - XVII Lezione

1

Prefazione alla lezione

- Nella lezione precedente abbiamo introdotto i linguaggi (funzionali) con tipi di ordine superiore.
- Si tratta di linguaggi più articolati di IMP in cui sono ammesse anche **funzioni come valori** (è possibile scrivere una procedura che prende in input e restituisce in output una funzione).
- La progettazione di tali linguaggi si basa su funzioni matematiche.
- Per ogni termine del linguaggio è possibile calcolarne il "tipo" tramite l'uso di regole.
- La valutazione (operazionale) di un termine è data tramite l'uso di "forme canoniche".
- Esistono due tipi di valutazione per i linguaggi funzionali:
 - Valutazione **Eager** o Call by value (lezione precedente), in cui gli oggetti sono valutati il prima possibile
 - Valutazione **Lazy** o Call by name (questa lezione), in cui la valutazione degli oggetti è rinviata al momento in cui tali valori sono di fatto usati.



Murano Aniello
Fond. LP - XVII Lezione

2

Esercitazione su termini tipabili

- Mostrare che il termine seguente è tipabile utilizzando le regole (di tipo) per la determinazione del relativo tipo :

$\text{if } ((\lambda x. \lambda y. x+y) z) k \text{ then } (z, k) \text{ else } (k, z)$

- Alcuni suggerimenti per la soluzione:

1. x e y devono essere interi; z e k dello stesso tipo;
2. La guardia dell'if deve essere intera, dunque $\lambda x. \lambda y. x+y$ è di tipo $\text{int} \rightarrow \text{int} \rightarrow \text{int}$ e z di tipo intero;
3. Dunque il termine è tipabile e il tipo del termine è $\text{int} * \text{int}$



Murano Aniello
Fond. LP - XVII Lezione

3

Esercitazione su valutazione eager

- Il termine $(\lambda x. \lambda y. x+y) 5$ non è in forma canonica, mentre lo è il termine $(\lambda y. 5+y)$. "Esercizio per gli studenti"

- Valutare se il termine $(\lambda x. \lambda y. \lambda z. ((x,z), (y,z)) \text{ rec } f. \lambda x. x+1)$ è in forma canonica.

- Suggerimenti per la soluzione:

- Il termine dato è una applicazione $(t_1 t_2)$. Appliciamo dunque la regola relativa, iniziando col trovare la forma canonica dei sottotermini;
- Il termine $t_1 \equiv \lambda x. \lambda y. \lambda z. ((x,z), (y,z))$ è già in forma canonica;
- Il termine t_2 è una "finta ricorsione" e la forma canonica è $\lambda x. x+1$;
- Visto che t_1 è di tipo $\lambda x.t$, possiamo applicare la regola di valutazione. Dunque la forma canonica del termine dato è ottenuta sostituendo x in t con la forma canonica di t_2 (cioè $\lambda y. \lambda z. ((x,z), (y,z)) [x \rightarrow \lambda x. x+1]$);
- Risposta: la forma canonica del termine è $\lambda y. \lambda z. ((\lambda x. x+1, z), (y,z))$



Murano Aniello
Fond. LP - XVII Lezione

4

Valutazione Lazy

- La valutazione Lazy (conosciuta anche come valutazione ritardata) consiste nel posticipare la valutazione di una computazione fino a quando (e ogni volta) il risultato della computazione è realmente usato.
- Tra i vantaggi della valutazione Lazy ricordiamo:
 - Aumento delle performance di un programma in seguito alla non valutazione di componenti non necessari alla computazione;
 - Riduzione della possibilità di incontrare errori nella valutazione condizionale di computazioni (l'errore potrebbe essere nella diramazione di un comando "if" che non verrà mai preso);



Murano Aniello
Fond. LP - XVII Lezione

5

Un esempio "performante"

- Nella valutazione lazy di un'espressione della forma

$$\text{let } x = N$$

consiste nel rimandare la valutazione di N fino a quando non è richiesto il valore di x.

- Esempio: Si consideri il termine

$$\text{let } x = N \text{ in } 1 \text{ con } N = \sum_{i=\log e}^{\sqrt{|\pi|^4}} (\log \pi^{i/2} \cdot \int_{\sqrt{2}}^e (i \cdot x)! dx)$$

perchè calcolare N se il risultato sarà comunque 1?

- Va notato che, adottando la valutazione lazy, non sempre si risparmia "tempo di calcolo". Lo si risparmia nel caso dell'esempio precedente, ma lo si triplica per $\text{let } x = N \text{ in } (x + x + x)$.



Murano Aniello
Fond. LP - XVII Lezione

6

Linguaggio Lazy

- Come nel caso eager, esistono termini la cui valutazione può produrre valori di base (numeri), coppie di valori o funzioni.
- Per tener conto della diversa natura dei valori prodotti dalla valutazione dei termini, consideriamo anche in questo caso il concetto di "tipi" per le espressioni, indicati con τ , e definiti come segue:

$$\tau ::= \text{int} \mid \tau_1 * \tau_2 \mid \tau_1 \rightarrow \tau_2$$

dove

- int → termine valutato con numero
- $\tau_1 * \tau_2$ → termine valutato coppia
- $\tau_1 \rightarrow \tau_2$ → termine valutato funzione



Murano Aniello
Fond. LP - XVII Lezione

7

Sintassi dei termini

- Assumendo che a ciascuna variabile x di **Var** sia associato un unico tipo, dato da $\text{type}(x)$, la sintassi dei termini è data da

$t ::= x \mid$	<i>names</i>
– these are bound names, not variable locations	
$n \mid t_1 + t_2 \mid t_1 - t_2 \mid t_1 \times t_2 \mid$	<i>arithmetic</i>
if t_0 then t_1 else $t_2 \mid$	<i>conditional</i>
– tests if t_0 is zero	
$(t_1, t_2) \mid$	<i>pair formation</i>
fst (t) snd (t) 	<i>projection</i>
$\lambda x. t \mid$	<i>abstraction</i>
$t_1 t_2 \mid$	<i>application</i>
let $x \leftarrow t_1$ in $t_2 \mid$	<i>prior evaluation</i>
rec $x.t$	<i>recursive function</i>

Dunque, cambiamo solo il rec

Dunque, il corpo del rec può essere un termine t qualsiasi e non una astrazione



Murano Aniello
Fond. LP - XVII Lezione

8

Termini tipabili

- Nella valutazione lazy (come nel caso della valutazione eager) un termine t è tipabile, cioè

$$t : \tau$$

- se corrisponde a un tipo $(\text{int}, \tau_1 * \tau_2, \tau_1 \rightarrow \tau_2)$, secondo le seguenti regole (di tipo):

- Variabili, operazioni, let, prodotti e funzioni hanno le stesse regole di tipo del caso eager
- Il *rec*, invece, ha la seguente regola di tipo (a lato quella eager)

$$\frac{x : \tau \quad t : \tau}{\text{rec } x.t : \tau} \quad \boxed{\frac{y : \tau \quad \lambda x.t : \tau}{\text{rec } y.(\lambda x.t) : \tau} \text{ [T-rec]}}$$

- Le variabili libere dei termini si calcolano con le regole introdotte per eager, ad eccezione del termine *rec* che ha la regola:

- $\text{FV}(\text{rec } x.t) = \text{FV}(t) \setminus \{x\}$
- Nel caso eager si ha invece $\text{FV}(\text{rec } y.(\lambda x.t)) = \text{FV}(\lambda x.t) \setminus \{x\}$



Semantica operativa lazy

- Nella semantica operativa lazy (come nel caso eager) si utilizzano **forme canoniche** dei termini.

- $t \in C_{\tau}^l$ indica che t è una forma canonica lazy di tipo τ , ed è definita per induzione strutturale su τ nel modo seguente:

- $n \in C_{\text{int}}^l$ (i numeri sono forme canoniche)
- $(t_1, t_2) \in C_{\tau_1 * \tau_2}^l$ se $t_1 : \tau_1$ & $t_2 : \tau_2$ con t_1 e t_2 chiusi;
- $\lambda x.t \in C_{\tau_1 \rightarrow \tau_2}^l$ se $\lambda x.t : \tau_1 \rightarrow \tau_2$ e $\lambda x.t$ è chiuso

Nella lazy, la coppia è già considerata in forma canonica. Nella eager, invece, la coppia deve essere trasformata in forma canonica.

- Dunque le forme canoniche sono particolari termini chiusi
- Definiamo ora le regole di inferenza in grado di ridurre un termine chiuso tipabile t in una forma canonica c , cioè

$$t \rightarrow^l c.$$



Valutazione di operatori e condizioni (come nel caso eager)

$c \rightarrow^e c$ for $c \in \mathcal{O}_T^e$

$$\frac{t_1 \rightsquigarrow^e n_1 \quad t_2 \rightsquigarrow^e n_2}{t_1 \text{ op } t_2 \rightsquigarrow^e c_1} \text{ [op]}$$

Attenzione, il simbolo "e" sulle derivazioni va cambiata con il simbolo "l"

$$\frac{t_0 \rightsquigarrow^e 0 \quad t_1 \rightsquigarrow^e c_1}{\text{if } t_0 \text{ then } t_1 \text{ else } t_2 \rightsquigarrow^e c_1} \text{ [if-T]} \quad \text{takes zero as true}$$

$$\frac{t_0 \rightsquigarrow^e n \quad n \neq 0 \quad t_2 \rightsquigarrow^e c_2}{\text{if } t_0 \text{ then } t_1 \text{ else } t_2 \rightsquigarrow^e c_2} \text{ [if-F]}$$



Valutazione del prodotto

$$\frac{t \rightsquigarrow^l (t_1, t_2) \quad t_1 \rightsquigarrow^l c_1 \text{ [fst]}}{\text{fst}(t) \rightsquigarrow^l c_1}$$

$$\frac{t \rightsquigarrow^l (t_1, t_2) \quad t_2 \rightsquigarrow^l c_2 \text{ [snd]}}{\text{snd}(t) \rightsquigarrow^l c_2}$$

Si noti che viene valutato un solo elemento della coppia e non entrambi come invece avveniva nella eager



Valutazione di funzioni, let e rec

◆ Lazy Application

- Just substitute the argument as an expression for x

$$\frac{t_1 \rightsquigarrow^1 \lambda x. t'_1 \quad [x \mapsto t_2] t'_1 \rightsquigarrow^1 c}{t_1 t_2 \rightsquigarrow^1 c} \text{ [app]}$$

◆ Let

- simple a short-hand

$$\frac{[x \mapsto t_1] t_2 \rightsquigarrow^1 c}{\mathbf{let} \ x \leftarrow t_1 \ \mathbf{in} \ t_2 \rightsquigarrow^1 c} \text{ [let]}$$

◆ Rec

- unfolds and evaluates

$$\frac{[x \mapsto \mathbf{rec} \ x.t] t \rightsquigarrow^1 c}{\mathbf{rec} \ x.t \rightsquigarrow^1 c} \text{ [let]}$$

Si noti come la sostituzione nel Let e nelle funzioni viene fatta senza richiedere la valutazione del termine da sostituire, dunque l'idea è quella di passare oggetti non ancora valutati

Esercizio 1

- Verificare se esiste la forma canonica lazy del seguente termine:

$$(\mathbf{rec} \ f. \lambda x. (f \ x) \ 5)$$

- Soluzione: Nella lezione precedente, abbiamo mostrato che non esiste una forma canonica eager del termine. Con una valutazione analoga è possibile dimostrare che non ne esiste una lazy.

Esercizio 2

- Verificare se esiste la forma canonica (e se sì quale) eager e lazy del seguente termine:

$\text{fst}(5+5, (\text{rec } f. \lambda x. (f \ x) \ 5))$

- Soluzione:

1. Non esiste una forma canonica eager perché il secondo termine della coppia non è in forma canonica
2. Esiste la forma canonica lazy che è 10. Questo perché, a differenza della eager, nella valutazione lazy del first di una coppia, non occorre che il secondo termine sia in forma canonica.



La valutazione lazy è deterministica

- Per dimostrare che la valutazione è deterministica occorre provare che per ogni termine se $t \rightarrow^l c$ e $t \rightarrow^l c'$ allora $c \equiv c'$.
- La precedente proprietà è facilmente dimostrabile per induzione sulle regole di derivazione date (esercizio per gli studenti).

