Valutazione Lazy

Aniello Murano
Università degli Studi di Napoli
"Federico II"

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
  - Valutazone **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.

Esercitazione su termini tipabili

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

  if ((λx. λy. x+y z) k) then (z , k) 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 λx. λy. x+y è di tipo int → int →int e z di tipo intero;
  3. Dunque il termine è tipabile e il tipo del termine è int*int

Esercitazione su valutazione eager

- Il termine (λx. λy. x+y 5) non è in forma canonica, mentre lo è il termine (λy. 5+y). "Esercizio per gli studenti"
- Valutare se il termine (λx. λy. λz. ((x,z), (y,z)) rec f. λx. x+1) è in forma canonica.
- Suggerimenti per la soluzione:
  - Il termine dato è una applicazione (t_1 t_2). Applichiamo dunque la regola relativa, iniziando col trovare la forma canonica dei sottotermini;
  - Il termine $t_1 \equiv$ λx. λy. λz. ((x,z), (y,z)) è già in forma canonica;
  - Il termine t_2 è una "finta ricorsione" e la forma canonica è λx. x+1;
  - Visto che t_1 è di tipo λx.t, possiamo applicare la regola di valutazione. Dunque la forma canonica del termine dato è ottenuta sostit. x in t con la forma canonica di t_2 (cioè λy. λz. ((x,z), (y,z)) [x → λx. x+1]);
  - Risposta: la forma canonica del termine è λy. λz. ((λx. x+1,z), (y,z))

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);

Un esempio "performante"

- Nella valutazione lazy di un'espressione della forma

  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

  let x= N in 1 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 let x = N in (x + x + x).

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

Sintassi dei termini

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

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

Dunque, cambiamo solo il rec

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

Termini tipabili

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

$$t : \tau$$

- se corrisponde a un tipo (int, $\tau_1 * \tau_2$, $\tau_1 \to \tau_2$), 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)

$$\frac{x: \tau \quad t: \tau}{\text{rec } x.t : \tau}$$

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

- Le variabili libere dei termini si calcolano con le regole introdotte per eager, ad eccezione del termine rec che ha la regola:
  - FV(rec x.t)=FV(t)\{x}
  - Nel caso eager si ha invece FV(rec y.(λx. t))=FV(λ x.t)\{x}

Semantica operazionale lazy

- Nella semantica operazionale lazy (come nel caso eager) si utilizzano **forme canoniche** dei termini.
- $t \in C^l_\tau$ indica che t è una forma canonica lazy di tipo τ, ed è definita per induzione strutturale su τ nel modo seguente:
- $n \in C^l_{int}$ (i numeri sono forme canoniche)
- $(t_1,t_2) \in C^l_{\tau_1 * \tau_2}$ se $t_1:\tau_1$ & $t_2:\tau_2$ con t_1 e t_2 chiusi;
- $\lambda x.t \in C^l_{\tau_1 \to \tau_2}$ se $\lambda x.t : \tau_1 \to \tau_2$ e λx.t è chiuso
- 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

$$t \to^{l} c.$$

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

Valutazione di operatori e condizioni (come nel caso eager)

$c \rightarrow^{e} c$ for $c \in C_T^e$

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

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

$$\frac{t_0 \rightsquigarrow^{e} 0 \qquad t_1 \rightsquigarrow^{e} c_1}{\textbf{if}\ t_0\ \textbf{then}\ t_1\ \textbf{else}\ t_2 \rightsquigarrow^{e} c_1}\ [\text{if-T}]$$

takes zero as true

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

Valutazione del prodotto

$$\frac{t \rightsquigarrow^{l} (t_1, t_2) \qquad t_1 \rightsquigarrow^{l} c_1}{\textbf{fst}(t) \rightsquigarrow^{l} c_1}\ [\text{fst}] \qquad \frac{t \rightsquigarrow^{l} (t_1, t_2) \qquad t_2 \rightsquigarrow^{l} c_2}{\textbf{snd}(t) \rightsquigarrow^{l} c_2}\ [\text{snd}]$$

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
- Let
  - simple a short-hand
- Rec
  - unfolds and evaluates

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

$$\frac{[x \mapsto t_1]t_2 \rightsquigarrow^l c}{\textbf{let}\ x \Leftarrow t_1\ \textbf{in}\ t_2 \rightsquigarrow^l c}\ [let]$$

$$\frac{[x \mapsto \textbf{rec}\ x.t]t \rightsquigarrow^l c}{\textbf{rec}\ x.t \rightsquigarrow^l c}\ [let]$$

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:

  (rec f. λx. (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 si quale) eager e lazy del seguente termine:

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

- 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).