Informatica Teorica - Appendice chicche e pinzillacchere

Andrea Asperti

Department of Computer Science, University of Bologna
Mura Anteo Zamboni 7, 40127, Bologna, ITALY
asperti@cs.unibo.it

Table of contents

Sull'esistenza del minimo

Sull'esistenza del minimo

Ogni sottoinsieme non vuoto di $\mathcal{N}$ ha un elemento minimo.

$$\exists x, x \in A \Rightarrow \exists m, (m \in A \wedge \forall z, (z \in A \Rightarrow m \leq z))$$

Manipoliamo un po' la formula:

$$\forall m, \neg(m \in A \wedge \forall z, (z \in A \Rightarrow m \leq z)) \Rightarrow \forall x, x \in \overline{A}$$
$$\forall m, (m \in \overline{A} \vee \neg(\forall z, (z \in A \Rightarrow m \leq z))) \Rightarrow \forall x, x \in \overline{A}$$
$$\forall m, (\forall z, (z \in A \Rightarrow m \leq z)) \Rightarrow m \in \overline{A}) \Rightarrow \forall x, x \in \overline{A}$$
$$\forall m, (\forall z, (z < m \Rightarrow z \in \overline{A}) \Rightarrow m \in \overline{A}) \Rightarrow \forall x, x \in \overline{A}$$

Se supporre che ogni naturale più piccolo di m stia in $\overline{A}$ é sufficiente per concludere che anche m sta in $\overline{A}$, allora $\overline{A}$ contiene tutti gli elementi.

che è una possibile formulazione del **principio di induzione**!

Sull'esistenza del minimo

Ogni sottoinsieme non vuoto di $\mathcal{N}$ ha un elemento minimo.

$$\exists x, x \in A \Rightarrow \exists m, (m \in A \wedge \forall z, (z \in A \Rightarrow m \leq z))$$

Manipoliamo un po' la formula:

$$\forall m, \neg(m \in A \wedge \forall z, (z \in A \Rightarrow m \leq z)) \Rightarrow \forall x, x \in \overline{A}$$
$$\forall m, (m \in \overline{A} \vee \neg(\forall z, (z \in A \Rightarrow m \leq z))) \Rightarrow \forall x, x \in \overline{A}$$
$$\forall m, (\forall z, (z \in A \Rightarrow m \leq z)) \Rightarrow m \in \overline{A}) \Rightarrow \forall x, x \in \overline{A}$$
$$\forall m, (\forall z, (z < m \Rightarrow z \in \overline{A}) \Rightarrow m \in \overline{A}) \Rightarrow \forall x, x \in \overline{A}$$

Se supporre che ogni naturale più piccolo di m stia in $\overline{A}$ é sufficiente per concludere che anche m sta in $\overline{A}$, allora $\overline{A}$ contiene tutti gli elementi.

che è una possibile formulazione del **principio di induzione**!

Ricorsione primitiva e ricorsione di coda

Ricorsione di coda

Una funzione si dice in forma *ricorsiva di coda* se la chiamata ricorsiva è l'ultima operazione eseguita dalla funzione chiamante.

Teorema

Ogni funzione primitiva ricorsiva può essere espressa in forma ricorsiva (non necessariamente primitiva) di coda.

Ricorsione di coda - dimostrazione

Data la funzione primitiva ricorsiva f si consideri la funzione ricorsiva di coda f'

$$\begin{cases} f(0,\vec{x}) = g(\vec{x}) \\ f(y+1,\vec{x}) = h\,(y, f(y,\vec{x}),\vec{x}) \end{cases} \qquad \begin{cases} f'(0,\vec{x},i,acc) = acc \\ f'(y+1,\vec{x},i,acc) = f'(y,\vec{x},i+1,h(i,acc,\vec{x})) \end{cases}$$

Dimostriamo per induzione su y che, per ogni i

$$f'(y,\vec{x},i,f(i,\vec{x})) = f(i+y,\vec{x})$$

Se $y = 0$,

$$f'(0,\vec{x},i,f(i,\vec{x})) = f(i,\vec{x}) = f(i+0,\vec{x})$$

Nel caso $y+1$, abbiamo

$$\begin{array}{rcl} f'(y+1,\vec{x},i,f(i,\vec{x})) & = & f'(y,\vec{x},i+1,h(i,f(i,\vec{x}),\vec{x})) \\ & = & f'(y,\vec{x},i+1,f(i+1,\vec{x})) \\ & = & f(i+1+y,\vec{x}) \text{ per ipotesi induttiva} \\ & = & f(i+y+1,\vec{x}) \end{array}$$

In particolare, $f(y,\vec{x}) = f'(y,\vec{x},0,g(\vec{x}))$

Ricorsione di coda e funzioni for-calcolabili

L'interesse delle funzioni ricorsive di coda consiste nel fatto che possono essere ricondotte banalmente a meccanismi di tipo iterativo, cioè esprimibili mediante un costrutto di tipo "for".

Vale anche il viceversa e quindi ne discende il seguente risultato:

Teorema

Le funzioni primitive ricorsive sono *tutte e solo* quelle `for`-calcolabili, cioè esprimibili in un linguaggio imperativo (del primo ordine) utilizzando come unici costrutti di controllo di flusso l'if-then-else, il for e la chiamata di funzione non ricorsiva.

Ordinamenti ben fondati

Ordinamenti ben fondati

Una relazione d'ordine si dice **ben fondata** se non ammette catene discendenti infinite.

Definiamo la seguente relazione d'ordine tra coppie di numeri naturali (detta relazione lessicografica):

$$\langle m_1, n_1 \rangle \prec \langle m_2, n_2 \rangle \quad \text{se e solo se} \quad \begin{cases} m_1 < m_2 \quad \text{oppure} \\ m_1 = m_2 \wedge n_1 < n_2 \end{cases}$$

dove $<$ indica l'usuale relazione d'ordine tra numeri naturali.

Ordinamenti ben fondati

La relazione di ordinamento lessicografico è ben fondata.
Dimostrazione Data la coppia $\langle m, n \rangle$ dobbiamo dimostrare che le catene discendenti che partono da essa hanno lunghezza finita. Procediamo per induzione su m.

- il caso $m = 0$ è ovvio poiché le catene hanno al più n elementi.
- Suppponiamo che il risultato sia vero per ogni coppia $\langle m, n \rangle$ e dimostriamolo per $\langle m + 1, p \rangle$. Percorrendo la catena discendente incontreremo il primo elemento della forma $\langle m, n \rangle$ dopo al più p passi. Visto che le catene che partono da quesst'ultimo hanno lunghezza finita, anche quelle di $\langle m + 1, p \rangle$ soddisfano la stessa proprietà.

Ordinamenti ben fondati

N.B. Non è detto che in un ordinamento ben fondato il numero dei punti minori di un punto dato sia finito.

Ad esempio, nell'ordinamento lessicografico, abbiamo

$$\langle 2, n \rangle \prec \langle 3, 5 \rangle$$

qualunque sia n.
Quindi, benchè le catene discendenti abbiano lunghezza finita, non siamo in grado dare un bound a tale lunghezza.

Se utilizziamo l'ordine lessicografico tra gli argomenti per garantire la terminazione delle chiamate ricorsive, non possiamo evincere da questo un upper bound al numero delle chiamate ricorsive che saranno effettuate (e dunque al tempo di calcolo)

Il sistema T di Gödel

Il sistema T di Gödel

TIPI: $\mathcal{B}, \mathcal{N}, T_1 \times T_2, T_1 \to T_2$

TERMINI:

variabili : $x, y, z, \ldots$;

costanti : 0, S, True, False, Fst, Snd, If, Rec.

coppie $< M, N >$

applicazione $(M\ N)$

astrazione $\lambda x : T.M$

Regole di tipizzazione per T (costanti)

$$
\begin{array}{lcl}
0 & : & \mathcal{N} \\
S & : & \mathcal{N} \to \mathcal{N} \\
True & : & \mathcal{B} \\
False & : & \mathcal{B} \\
Fst & : & T_1 \times T_2 \to T_1 \\
Snd & : & T_1 \times T_2 \to T_2 \\
If & : & T \to T \to \mathcal{B} \to T \\
Rec & : & T \to (\mathcal{N} \to T \to T) \to \mathcal{N} \to T
\end{array}
$$

Regole di tipizzazione per T (termini strutturati)

Predicato di buona tipizzazione

$$\Gamma \vdash M : T$$

leggi: il termine M ha tipo T nel contesto Γ.
Il contesto è un insieme di dichiarazioni di tipo della forma $x : T_x$.
Ogni variabile libera in M deve essere dichiarata nel contesto.

coppia
$$\frac{\Gamma \vdash M : T_1 \qquad \Gamma \vdash N : T_2}{\Gamma \vdash < M, N > : T_1 \times T_2}$$

applicazione
$$\frac{\Gamma \vdash M : T_1 \to T_2 \qquad \Gamma \vdash N : T_1}{\Gamma \vdash (M\ N)\ : T_2}$$

astrazione
$$\frac{\Gamma, x : T_1 \vdash M : T_2}{\Gamma \vdash \lambda x : T_1 . M : T_1 \to T_2}$$

Regole di riduzione per T

$$
\begin{array}{ll}
(Fst) & (Fst < M, N >) \rightarrow M \\
(Snd) & (Snd < M, N >) \rightarrow N \\
(\beta) & (\lambda x : T.M\ N) \rightarrow M[N/x] \\
(If\_true) & (If\ M\ N\ true) \rightarrow M \\
(If\_false) & (If\ M\ N\ false) \rightarrow N \\
(Rec\_O) & (Rec\ M\ N\ O) \rightarrow M \\
(Rec\_S) & (Rec\ M\ N\ (S\ P)) \rightarrow (N\ P\ (Rec\ M\ N\ P))
\end{array}
$$

Remark: il tipo è invariante per riduzione (subject reduction).

La funzione di Ackermann in T

Siccome il sistema T è un linguaggio di ordine superiore tutti i funzionali di ricorsione e iterazione (di ogni ordine!) sono esprimibili in T.

Quindi, come visto a lezione, il seguente programma permette di calcolare Ackermann ed appartiene al linguaggio

$$\begin{aligned} \mathtt{ack}_{0,y} &= \lambda x.x + y \\ \mathtt{ack}_{1,y} &= \mathit{Ite}\ 0\ \mathtt{ack}_{0,y} \\ \mathtt{ack}_{z+1,y} &= \mathtt{Ite}\ \mathtt{ack}_{1,y}\ \mathtt{next}\ z \end{aligned}$$

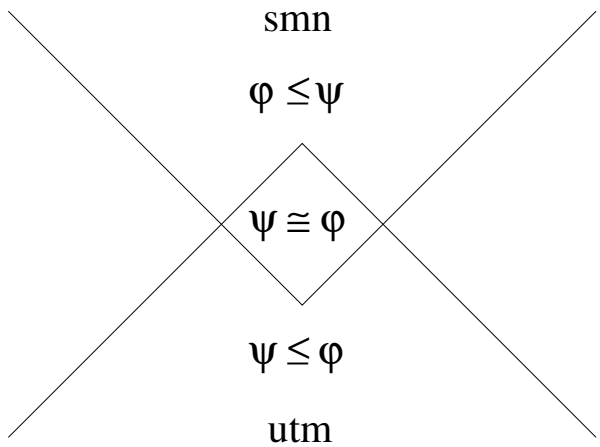
dove $\mathtt{next} = \lambda f.\mathtt{Ite}\ 1\ f$.

Esercizio: verificare che ogni termine $\mathtt{ack}_{z,y}$ è ben tipato e ha tipo $\mathcal{N} \to \mathcal{N}$.

Il teorema di equivalenza di Roger

Il teorema di Roger asserisce che **tutte le enumerazioni accettabili di funzioni parziali ricorsive sono equivalenti tra loro**. Più precisamente, è possibile dimostrare che, supposto φ accettabile,

- $\psi \leq \varphi \Leftrightarrow \psi$ soddisfa la proprietà utm
- $\varphi \leq \psi \Leftrightarrow \psi$ soddisfa la proprietà smn



Dimostrazione del teorema di Roger

Sia u l'indice di un interprete per φ.

- $\psi \leq \varphi \Leftrightarrow \psi$ soddisfa la proprietà utm:
  - $\Rightarrow$ Se esiste f t.c. tale che $\psi_n = \varphi_{f(n)}$, allora la funzione $\varphi_u(f(n), m)$ é calcolabile e deve essere enumerata da ψ. L'indice di tale funzione soddisfa utm.
  - $\Leftarrow$ Preso u' tale che $\psi_{u'}(x, y) = \psi_x(y)$, per smn esiste s t.c. tale che $\varphi_{s(x)}(y) = \psi_{u'}(x, y) = \psi_x(y)$.
- $\varphi \leq \psi \Leftrightarrow \psi$ soddisfa la proprietà smn
  - $\Rightarrow$ Sia f t.c. tale che $\varphi_n = \psi_{f(n)}$. Data una funzione calcolabile $g(n, m)$ esiste s t.c. tale che $\varphi_{s(n)}(m) = g(n, m)$; allora $f \circ s$ soddisfa smn per ψ.
  - $\Leftarrow$ Si consideri la funzione t.c. $\varphi_u(x, y) = \varphi_x(y)$; la funzione s tale che $\psi_{s(x)}(y) = \varphi_u(x, y)$ riduce φ a ψ.

Fissare una particolare enumerazione non è più rilevante