Capitolo 7:
Teoria generale della calcolabilità

Differenti nozioni di calcolabilità (che seguono da differenti modelli di calcolo) portano a definire la stessa classe di funzioni.

Le tecniche di simulazione fra modelli introducono i concetti di configurazione, calcolo e codifica. Possiamo formulare una teoria astratta della calcolabilità, indipendente dal modello di calcolo adottato??

Tesi di Church-Turing: ogni funzione calcolabile rispetto ad una qualunque nozione di algoritmo (Turing, Kleene, λ-calcolo, Markov, Post) è calcolabile secondo Turing.

Quindi si congettura che la classe delle funzioni ricorsive sia la classe di funzioni più generale per la quale si possa dare un metodo di calcolo algoritmico.

Se si accetta tale tesi, si possono utilizzare descrizioni informali di algoritmi per dimostrare la calcolabilità di funzioni senza esibire una macchina che le calcola.

Una enumerazione delle macchine a registri elementari

Abbiamo visto che ogni funzione calcolata tramite una macchina a registri elementare è ricorsiva (associando numeri naturali agli stati della macchina e alle sequenze di stati).

Mostriamo che anche i programmi per macchine a registri elementari possono essere codificati (cioè esiste una *aritmetizzazione* o *gödelizzazione*). Ricordiamo che ogni programma elementare è formato da un numero finito di istruzioni del tipo:

$Ri = Ri + 1$ $\qquad\qquad$ $i \geq 1$, $n \geq 0$
$Ri = Ri \dot{-} 1$
IF $Ri = 0$ GOTO n

Per codificare una istruzione abbiamo bisogno:
✓del suo posto nella sequenza di istruzioni,
✓del suo tipo,
✓dei numeri i ed n che essa contiene.

Un modo per codificare queste informazioni è il seguente:

$$\begin{array}{lcl} Ri = Ri + 1 & \longrightarrow & 3 \cdot (i \dot{-} 1) + 1 \\ Ri = Ri \dot{-} 1 & \longrightarrow & 3 \cdot (i \dot{-} 1) + 2 \\ \text{IF } Ri = 0 \text{ GOTO } n & \longrightarrow & 3 \cdot J(i \dot{-} 1, n) \end{array}$$

dove $J : \mathbb{N} \times \mathbb{N} \longmapsto \mathbb{N}^+$ è la biiezione

$$J(x,y) = \frac{(x+y+1)(x+y)}{2} + x + 1$$

Il codice dell'istruzione m-esima è usato come esponente per il numero primo p_m ($p_1 = 2$); se il programma ha k istruzioni il suo codice sarà $2^{y_1} \ldots p_k^{y_k}$, oppure $\langle y_1, \ldots, y_k \rangle$, dove $y_1, \ldots, y_k$ sono i codici delle istruzioni.

Esempio 1 *Codifichiamo il seguente programma:*

$$\begin{array}{lcl} \textit{IF } R2 = 0 \textit{ GOTO } 0 & \to & 3 \cdot J(1,0) = 9 \\ R2 = R2 \dot{-} 1 & \to & 3 \cdot 1 + 2 = 5 \\ R1 = R1 + 1 & \to & 3 \cdot 0 + 1 = 1 \\ \textit{IF } R3 = 0 \textit{ GOTO } 1 & \to & 3 \cdot J(2,1) = 27 \end{array}$$

e quindi il codice è $2^9 \cdot 3^5 \cdot 5^1 \cdot 7^{27}$, *oppure* $\langle 9, 5, 1, 27 \rangle$.

La codifica e la decodifica sono entrambe funzioni biiettive (perche?); ad ogni intero

$$x = 2^{y_1} \dots p_m^{y_m} \text{ dove } y_i \begin{cases} \geq 0 & \text{se } 1 \leq i \leq m, \\ > 0 & \text{se } i = m \end{cases}$$

corrisponde un programma di m istruzioni, in cui la j-esima istruzione è ottenuta dividendo y_i per 3 ed individuando resto e quoziente.

Esempio 2 *A 600 ($= 2^3 \cdot 3 \cdot 5^2$) corrisponde il programma:*

IF $R1 = 0$ *GOTO* 0
$R1 = R1 + 1$
$R1 = R1 \dot{-} 1$

Quale funzione calcola questo programma?

Importante: la codifica di un programma produce interi $x = 2^{y_1}, \dots, p_k^{y_k}$, con $y_1, \dots, y_k > 0$. Invece, la decodifica di un numero naturale può produrre situazioni in cui all'istruzione j-esima corrisponde $y_j = 0$.
Per evitare questo problema, introduciamo l'istruzione *NOPE* (con codice 0) che non fa nulla.

Esercizio: quale programma è rappresentato dal numero $2^9 \cdot 5^5 \cdot 7 \cdot 11^{27}$??

Si può definire una macchina a registri elementare U che dati in input due naturali x e y, determini l'output che una macchina a registri elementare con codifica x ha per input y (*macchina a registri elementare universale*).