Una enumerazione delle macchine di Turing

Realizziamo una corrispondenza biunivoca fra macchine di Turing e numeri naturali. Sia data una macchina $\mathcal{M} = \langle \Gamma, \not b, Q, q_0, F, \delta \rangle$, la cui funzione di transizione è

$$\delta : (Q - F) \times (\Gamma \cup \{\not b\}) \longmapsto Q \times (\Gamma \cup \{\not b\}) \times \{d, s, i\}.$$

Ogni coppia (q, a) per cui la δ non è definita è codificata come $(q, a, \perp, \perp, \perp)$. Elenchiamo in ordine lessicografico le quintuple che costituiscono la funzione di transizione, introducendo un ordinamento arbitrario sugli insiemi $Q \cup \{\perp\}$, $\Gamma \cup \{\not b, \perp\}$, $\{d, s, i, \perp\}$.

Questo ordinamento induce un ordinamento fra le quintuple e, di conseguenza, sulle sequenze di quintuple e sulle codifiche di macchine di Turing.

Enumerazione delle funzioni ricorsive

I metodi definiti determinano una corrispondenza naturali e macchine (di Turing o a registri), ma anche fra funzioni ricorsive e numeri naturali.

Sia $\mathcal{E} : \mathbb{N} \longmapsto \{\mathcal{M}\}$ una biiezione fra i naturali e le descrizioni di macchine di Turing, tale che $\mathcal{E}(x) = \mathcal{M}_x$, la $(x+1)$-esima macchina dell'enumerazione. Data una codifica $\mathcal{C}$ dei naturali su un opportuno alfabeto (quello di nastro della macchina), definiamo

$$\varphi_x(y) = \begin{cases} z \in \mathbb{N} & \text{se la macchina } \mathcal{M}_x, \text{ inizializzata nella} \\ & \text{configurazione iniziale } q_0\mathcal{C}(y), \\ & \text{si arresta nella configurazione finale } \mathcal{C}(y)\ \not b q_F \mathcal{C}(z) \\ \text{indefinita} & \text{altrimenti.} \end{cases}$$

Si ottiene una enumerazione delle funzioni ricorsive ad un argomento (domanda: quale posto occupa la funzione φ_x in questa enumerazione? la corrispondenza fra naturali e funzioni ricorsive a un posto è una biiezione?)
Si può estendere l'enumerazione a funzioni a più argomenti attraverso una opportuna codifica (definirne una).

Il valore x associato alla funzione φ_x si dice *indice della funzione* (calcolata dalla macchina $\mathcal{M}_x$).

Dato che x è il numero naturale che codifica la macchina $\mathcal{M}_x$, esso è anche chiamato il *programma* che calcola la funzione φ_x.

Proprietà di enumerazioni di funzioni ricorsive

L'enumerazione ci consente di formulare i risultati della teoria della ricorsività in modo *indipendente dalla macchina*.

Tutta l'informazione relativa ad una particolare classe di macchine o programmi è contenuta negli indici delle funzioni, e dà origine ad una enumerazione (magari diversa dalle precedenti) che gode sempre delle stesse proprietà generali.

Una dimostrazione fornita assumendo una enumerazione per un certo modello di calcolo diventa valida per tutte le enumerazioni associate ad altri modelli di calcolo.

Teorema 3 *(esistenza della funzione universale) Sia data una qualsiasi enumerazione delle funzioni ricorsive. Per ogni* $k \in \mathbb{N}$ *esiste* z *tale che, per ogni* $x_1, \ldots, x_k$*:*

$$\varphi_z(x_1, \ldots, x_k, y) = \begin{cases} \varphi_y(x_1, \ldots, x_k) & \text{se essa è definita,} \\ \text{indefinita} & \text{altrimenti.} \end{cases}$$

Dimostrazione. Essa esiste nel caso di macchine a registri elementari. Basta realizzare la codifica dei k argomenti in un unico numero naturale e ricorrere alla MREL universale.
Inoltre si osserva che dati due modelli di calcolo $\mathcal{MC}_1$ e $\mathcal{MC}_2$ dotati di potere computazionale equivalente alle macchine di Turing, esiste una funzione calcolabile che associa al codice di una macchina in $\mathcal{MC}_1$ il codice di una macchina in $\mathcal{MC}_2$ che calcola la stessa funzione. Quindi l'esistenza di una funzione universale in $\mathcal{MC}_2$ implica l'esistenza di una funzione universale in $\mathcal{MC}_1$.

Teorema 4 *(*s*-*m*-*n*) Sia data una qualsiasi enumerazione delle funzioni ricorsive. Per ogni* $m, n \geq 1$ *esiste una funzione ricorsiva* s *tale che, per ogni* $x, y_1, \ldots, y_m, z_1, \ldots, z_n$*:*

$$\varphi_x(y_1, \ldots, y_m, z_1, \ldots, z_n) = \varphi_{s(x, y_1, \ldots, y_m)}(z_1, \ldots, z_n)$$

Dimostrazione. Assumiamo $m = n = 1$. Nel caso di una enumerazione associata alle macchine a registri, dimostriamo che esiste s tale che: $\varphi_x(y, z) = \varphi_{s(x,y)}(z)$. Dati x e y generiamo la macchina a registri di indice x e la trasformiamo in una nuova macchina che inizialmente inserisce y nel primo registro, quindi si comporta esattamente come la precedente. Infine, determiniamo il codice di tale nuova macchina $x' = s(x, y)$.
L'estensione a m,n qualunque è immediata.
L'estensione a tutte le enumerazioni segue da quanto detto nella precedente dimostrazione.

Teorema 5 *(forma normale di Kleene) Sia data una qualsiasi enumerazione delle funzioni ricorsive. Esistono una funzione ricorsiva U e un insieme di funzioni ricorsive T_k tali che, per ogni $k \in \mathbb{N}$, tutte le funzioni ricorsive di k argomenti sono esprimibili come segue:*

$$\varphi_i(x_1, \ldots, x_k) = U(\mu t[T_k(i, x_1, \ldots, x_k, t) = 0]).$$

Il teorema mostra che una funzione ricorsiva è calcolabile attraverso il calcolo di due funzioni: la prima (il predicato di Kleene) verifica che esista una computazione ammissibile; la seconda, che spacchetta il codice della computazione per restituire il risultato.

Dimostrazione. Supponiamo che l'enumerazione delle funzioni ricorsive sia quella derivante dall'enumerazione delle MREL.
Dato ogni programma π siamo in grado di fornire una funzione ricorsiva T_π tale che se f è la funzione calcolata mediante π allora:

$$f(x_1, \ldots, x_k) = U(\mu t[T_\pi(x_1, \ldots, x_k, t) = 1])$$

Sia π_i il programma MREL che calcola φ_i. Si può costruire una funzione ricorsiva T_k tale che, assumendo in input il codice i di π_i, determina tale programma e, dati i valori $x_1, \ldots, x_n, t$ calcola il valore $T_{\pi_i}(x_1, \ldots, x_k, t)$.

La generalizzazione del risultato ad una qualunque enumerazione è simile al teorema della esistenza di una funzione universale.

Funzioni non calcolabili

Abbiamo già introdotto i concetti di funzione calcolabile, non calcolabile, di insieme decidibile e semidecidibile.

Possiamo riformulare quanto visto in modo indipendente dal modello di calcolo, avendo introdotto il concetto di enumerazione di funzioni ricorsive.

Teorema 6 *Sia data una qualsiasi enumerazione delle funzioni ricorsive. Non esiste alcuna funzione ricorsiva g tale che per ogni x e y:*

$$g(x,y) = \begin{cases} 1 & \textit{se } \varphi_x(y) \textit{ è definita,} \\ 0 & \textit{altrimenti.} \end{cases}$$

Dimostrazione. Se la funzione g fosse ricorsiva lo sarebbe anche:

$$f(x) = \begin{cases} 1 & \text{se } g(x,x) = 0, \\ indefinita & \text{altrimenti.} \end{cases}$$

Sia e un indice per la funzione f, allora:
$f(e) = \varphi_e(e) = 1$ se e solo se $g(e,e) = 0$
ma per la definizione di g:
$g(e,e) = 0$ se e solo se $\varphi_e(e)$ non è definita.
Ad analoga contraddizione si perviene se $f(e) = \varphi_e(e) = 0$.
c.d.v.

Teorema 7 *Non esiste alcuna funzione ricorsiva g' tale che per ogni x:*

$$g'(x) = \begin{cases} 1 & \text{se } \varphi_x(x) \text{ è definita,} \\ 0 & \text{altrimenti.} \end{cases}$$

Corollario 8 *Sia data una qualsiasi enumerazione delle funzioni ricorsive. Gli insiemi*

$$K = \{x | \varphi_x(x) \text{è definita}\} \text{ e } K' = \{\langle x, y\rangle | \varphi_x(y) \text{è definita}\}$$

non sono decidibili.

✓decidere se un generico programma si arresta su un generico input
✓decidere se un generico programma si arresta avendo come input il proprio codice
✓decidere se un generico programma si arresta con input prefissato
sono problemi con la stessa difficoltà.

Teorema 9 *Sia data una qualsiasi enumerazione delle funzioni ricorsive. Non esiste una funzione ricorsiva g tale che per ogni x:*

$$g(x) = \begin{cases} 1 & \text{se } \varphi_x(0) \text{ è definita,} \\ 0 & \text{altrimenti.} \end{cases}$$

Dimostrazione. Sia

$$f(x,y) = \begin{cases} 0 & \text{se } \varphi_x(x) \text{ è definita,} \\ indefinita & \text{altrimenti.} \end{cases}$$

La f è ricorsiva. In base al teorema s-m-n esiste s ricorsiva tale che $\varphi_{s(x)}(y) = f(x,y)$. Per definizione della f le varie $\varphi_{s(x)}(y)$ saranno o identicamente nulle oppure sempre indefinite. Se g fosse ricorsiva potremmo definire una funzione ricorsiva $g' = g \cdot s$:

$$g'(x) = g(s(x)) = \begin{cases} 1 & \text{se } \varphi_{s(x)}(0) \text{ è definita,} \\ 0 & \text{altrimenti.} \end{cases}$$

Quindi la funzione:

$$g'(x) = \begin{cases} 1 & \text{se } \varphi_x(0) \text{ è definita,} \\ 0 & \text{altrimenti.} \end{cases}$$

è ricorsiva (contrariamente a quanto dimostrato precedentemente.

Applicando le tecniche precedenti, si provano i seguenti risultati:

Teorema 10 *Sia data una qualsiasi enumerazione delle funzioni ricorsive. Non esiste una funzione ricorsiva g tale che per ogni x:*

$$g(x) = \begin{cases} 1 & \textit{se } \varphi_x \textit{ è costante,} \\ 0 & \textit{altrimenti.} \end{cases}$$

Dimostrazione. Basta definire g' come segue:

$$g'(x) = \begin{cases} 1 & \text{se } \varphi_{s(x)} \text{ è costante,} \\ 0 & \text{altrimenti.} \end{cases}$$

È chiaro che $\varphi_{s(x)}$ è costante (ed uguale a zero) se e solo se $\varphi_x(x)$ è definita. Ma la funzione binaria $g'(x)$ che restituisce 1 se e solo se $\varphi_x(x)$ è definita non è ricorsiva.

Teorema 11 *Sia data una qualsiasi enumerazione delle funzioni ricorsive. Non esiste una funzione ricorsiva g tale che per ogni x, y, z:*

$$g(x,y,z) = \begin{cases} 1 & \textit{se } \varphi_x(y) = z, \\ 0 & \textit{altrimenti.} \end{cases}$$

Dimostrazione. Se g fosse ricorsiva, lo sarebbe anche:

$$g'(x) = g(s(x),0,0) = \begin{cases} 1 & \text{se } \varphi_{s(x)}(0) = 0, \\ 0 & \text{altrimenti.} \end{cases}$$

Ma abbiamo già dimostrato che g' non può essere ricorsiva.

Teorema 12 *Sia data una qualsiasi enumerazione delle funzioni ricorsive. Non esiste una funzione ricorsiva g tale che per ogni x, y:*

$$g(x,y) = \begin{cases} 1 & \textit{se } \varphi_x = \varphi_y, \\ 0 & \textit{altrimenti.} \end{cases}$$

Dimostrazione. Sia e un indice per la funzione $\lambda x[0]$. Allora avremo ancora che:

$$g'(x) = g(s(x),e) = \begin{cases} 1 & \text{se } \varphi_{s(x)} = \varphi_e, \\ 0 & \text{altrimenti.} \end{cases}$$

è ricorsiva, in contraddizione con i risultati precedenti.

Dimostreremo che ogni proprietà non banale è indecidibile.

Una proprietà è non banale se è valida per un insieme non vuoto di istanze, ma non per l'intero universo delle istanze stesse.

N.B.: Tutte le dimostrazioni precedenti si riconducono alla indecidibilità del problema della terminazione. Un metodo alternativo consiste nell'utilizzare la diagonalizzazione (rifare le dimostrazioni con questo metodo).

Teorema 13 *Data una qualunque enumerazione delle funzioni ricorsive, non esiste una funzione ricorsiva g tale che, per ogni x,*

$$g(x) = \begin{cases} 1 & \textit{se } \varphi_x \textit{ è totale,} \\ 0 & \textit{altrimenti.} \end{cases}$$

Dimostrazione. Se g fosse ricorsiva allora lo sarebbe anche:

$$f(x) = \begin{cases} \varphi_x(x) + 1 & \text{se } g(x) = 1, \text{ cioè se } \varphi_x \text{ è totale,} \\ 0 & \text{altrimenti.} \end{cases}$$

Chiaramente f sarebbe totale. Se e fosse un suo indice si avrebbe la seguente contraddizione:

$$\varphi_e(e) = f(e) = \varphi_e(e) + 1$$

Indecidibilità in matematica e informatica

Esistono numerosi problemi indecidibili in diversi settori della matematica e dell'informatica. Ad esempio, nella teoria dei linguaggi formali:

✓il problema della ambiguità di una grammatica context-free è indecidibile.

✓date due grammatiche context-free $\mathcal{G}_1$ e $\mathcal{G}_2$, $L(\mathcal{G}_1) = L(\mathcal{G}_2)$?

✓date due grammatiche context-free $\mathcal{G}_1$ e $\mathcal{G}_2$, $L(\mathcal{G}_1) \cap L(\mathcal{G}_2) = \emptyset$?

✓data una grammatica contenstuale $\mathcal{G}$, $L(\mathcal{G})$ è vuoto?

Nell'ambito della teoria della programmazione (riferendoci ad un linguaggio specifico, e non ad un modello di calcolo generico):

✓un programma che contiene una dichiarazione di proceduram chiamerà la procedura stessa?

✓una variabile di programma assumerà un particolare valore durante l'esecuzione del programma?

✓un programma, in presenza di un particolare input, fornisce un determinato output?

✓due programmi calcolano la stessa funzione?

✓un programma calcola una funzione costante?

✓un programma calcola una funzione totale?

Nel campo della matematica:

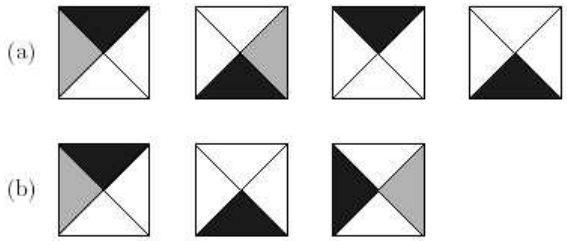
✓data una formula del calcolo dei prodicati, tale formula è un teorema?

✓data una formula dell'aritmetica, tale formula è un teorema della teoria?

✓Dato un sistema di equazioni lineari a coefficienti ed esponenti interi (e.g., $x^2 + y^2 = z^2$), il sistema ammette soluzioni intere?

Problema di pavimentazione (*tiling*)

Dato un insieme di mattonelle quadrate con il bordo colorato, ciascuna suddivisa in quattro triangoli e disponibili in numeri arbitrario, è possibile pavimentare un pavimento di qualsiasi forma?



1 Due insiemi di base di mattonelle. (a) Caso in cui è possibile pavimentare. (b) Caso in cui non è possibile.