Master in Comunicazione scientifica 2002-03

Macchine di Turing

Gabriele Lolli

Dipartimento di Matematica
Università di Torino

Una macchina di Turing[1] (MT) è un insieme finito e non contraddittorio di istruzioni; l'alfabeto in cui sono scritte è composto da due insiemi finiti non vuoti disgiunti, quello dei *simboli* di lettura e scrittura $\{\alpha_1, \ldots, \alpha_m\}$ e quello degli *stati* $\{q_0, \ldots, q_n\}$, e dalle lettere D e S. Tra i simboli si include sempre un simbolo speciale $*$ (*blank*).

Usiamo le lettere greche α, β come metavariabili sui simboli, e M come metavariabile su $\{D, S\}$.

Un'istruzione è una quadrupla di uno di questi due tipi:

$$q_i\ \alpha\ \beta\ q_j$$
$$q_i\ \alpha\ M\ q_j$$

e un insieme di istruzioni si dice non contraddittorio se non contiene due istruzioni diverse che iniziano con la stessa coppia $q_i\ \alpha$.

L'effetto della macchina è quello di trasformare parole di un tipo particolare in parole dello stesso tipo; si tratta di parole in cui compare un solo stato e gli altri elementi sono simboli, come in $\alpha_{i_1}, \ldots, \alpha_{i_j}, q_h, \alpha_{i_{j+1}}, \ldots, \alpha_{i_r}$.

Un'interpretazione intuitiva fisicamente figurata è quella di pensare a un nastro potenzialmente infinito, cioè sempre finito ma prolungabile in entrambi i sensi in caso di necessità. Il nastro è diviso in caselle, in cui ogni casella contiene un simbolo; una testina di lettura e scrittura in ogni istante è posizionata su di una casella, nell'esempio della parola di prima la casella

[1] A. M. Turing introdusse questa nozione nel 1936 come definizione matematica precisa del concetto intuitivo di "funzione effettivamente calcolabile" per dimostrare l'esistenza di problemi indecidibili (tra cui quello della validità logica).

su cui è scritto $\alpha_{i_{j+1}}$ e la macchina si trova in un particolare stato q_h. Si può pensare al nastro come nastro infinito, ma tale che in ogni istante solo un numero finito di caselle non siano vuote. Una casella si dice vuota se è marcata con *blank*.

La macchina fisica, oltre a una testina che le permette di leggere (nel senso di reagire in modo diverso a simboli diversi) e scrivere, ha un corpo che è una scatola nera, di cui si sa solo che può assumere un numero finito di configurazioni o stati distinti, in funzione della storia precedente di attività svolte e successione di stati assunti. La storia è riassunta a tutti gli effetti dall'ultimo stato assunto (anche se questo, con un numero finito di stati e storie arbitrariamente lunghe, non è del tutto evidente).

L'effetto delle istruzioni $q_i\,\alpha\,\beta\,q_j$ è il seguente: se la macchina in un dato istante si trova nello stato q_i e (è posizionata su una casella su cui) legge α, scrive al suo posto β e all'istante successivo passa allo stato q_j; l'effetto delle istruzioni $q_i\,\alpha\,M\,q_j$ è che se la macchina in un dato istante si trova nello stato q_i e legge α, lo lascia scritto e si muove di una casella a destra o a sinistra a seconda che M sia D o S, passando allo stato q_j.

Se la macchina in un dato istante si trova nello stato q_i e legge α, e tra le sue istruzioni non ce ne è nessuna che inizi con la coppia $q_i\ \alpha$, la macchina si ferma. La macchina si fermerebbe anche se, data la situazione descritta, esistesse un'istruzione $q_i\ \alpha\ \alpha\ q_i$. Si noti che nelle istruzioni q_i non è necessariamente diverso da q_j come α non necessariamente diverso da β, si può cambiare stato e lasciare lo stesso simbolo, oppure cambiare simbolo e restare nello stesso stato.

Come alfabeto di simboli per i calcoli numerici è sufficiente l'insieme di due simboli, la sbarretta | e $*$. Questa affermazione è però un profondo e difficile risultato della teoria delle MT. Di solito è più comodo avere a disposizione altri simboli, in particolare ad esempio simboli che servono a delimitare il campo finito significativo, ed estendibile spostando i delimitatori, del nastro entro cui si lavora.

Diverse convenzioni standardizzano la scrittura delle macchine di Turing:

- si conviene che ci sia sempre almeno una istruzione che inizia con q_0, che è detto *stato iniziale*, e che la macchina all'istante iniziale sia nello stato q_0 e la testina sia posizionata sul primo simbolo non *blank* a sinistra;

- per l'arresto abbiamo già detto quale soluzione adottiamo, anche se sono possibili altre soluzioni, come richiedere che la macchina si fermi

sempre in un particolare stato finale q_H, H per *halt*, oppure ammettere anche insiemi contraddittori con la condizione che la macchina si ferma se ha due istruzioni contrastanti;

- per la rappresentazione dei numeri interi positivi, se i simboli sono $|, *$, il numero n è rappresentato in input da una successione ininterrotta di $n+1$ sbarrette $|$ (il numero 0 da $|$);
- una coppia di numeri $\langle m, n\rangle$ sarà rappresentata in input da $|||\cdots| * ||\cdots|$ con $m+1$ sbarrette seguite da $*$ e da $n+1$ sbarrette:

$$\underbrace{|||\cdots|}_{m+1} * \underbrace{||\cdots|}_{n+1};$$

- quando la macchina si ferma, per l'interpretazione del risultato numerico sono possibili diverse alternative, che possono variare anche da macchina a macchina, ma devono essere dichiarate (e la loro adozione influenza la scrittura della macchina): si può pretendere che il numero sia rappresentato come in input da una parola $||\cdots|$ e che queste siano le uniche caselle non *blank* del nastro; oppure che ci possano anche essere altre caselle non *blank* ma che la testina sia posizionata su una delle caselle del blocco di sbarrette consecutive che rappresentano il risultato, o all'inizio o alla fine o in altre posizione intermedie; oppure si può convenire che quando la macchina si ferma un opportuno decodificatore conti il numero totale finito di $|$ sul nastro (per questo è necessario che sia ben delimitata da simboli speciali la porzione di nastro da controllare), e altre soluzioni sono ancora possibili.

Osservazione: In alcune presentazioni, le istruzioni delle MT invece che quadruple sono quintuple. Due istruzioni come

$$q_i\ \alpha\beta\ q_j$$
$$q_j\beta M q_k$$

vengono rimpiazzate dalla quintupla

$$q_i\alpha\beta M q_k.$$

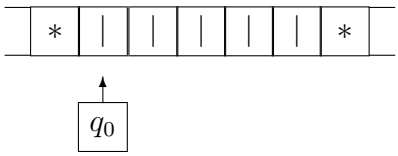
e viceversa.

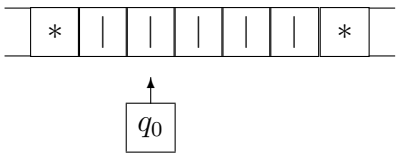
Nel seguito, per esercizio, per ogni MT scritta con quadruple si scriva la corrispondente versione con quintuple.

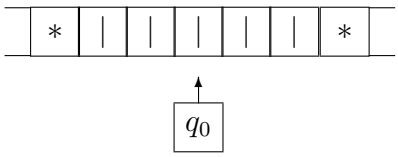
Esempio: Le seguenti istruzioni

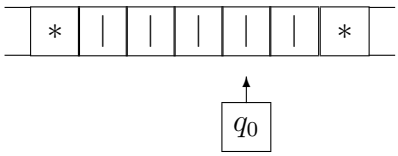
$$q_0 \mid D\ q_0$$
$$q_0\ * \mid q_1$$

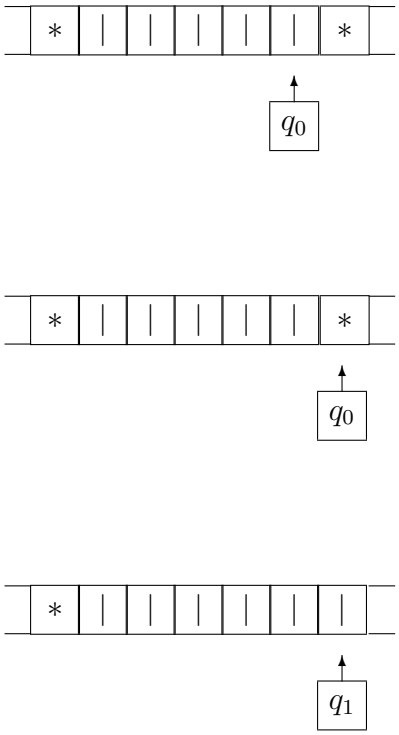
costituiscono una MT che computa il successore di ogni numero; la macchina si ferma nello stato q_1 posizionata sulla | più a destra del risultato. La successione dei nastri e degli stati è la seguente:











Se si aggiungono

$$q_1 \mid S\, q_1$$
$$q_1 * D\, q_2$$

si ottiene un'altra macchina che computa la stessa funzione, ma si ferma nello stato q_2 posizionata sulla prima | di sinistra del risultato[2].

Osservazione 1. *Esistono macchine diverse che eseguono lo stesso compito.*

Poichè una MT *è* l'insieme delle sue istruzioni, le due macchine di sopra sono chiaramente diverse, ma si possono chiamare equivalenti; è anche facile

[2]L'ultima potrebbe anche essere $q_1 * D\, q_H$.

immaginare che ne esistono infinite diverse tra di loro ed equivalenti (basta aggiungere istruzioni inoffensive, ad esempio che iniziano con uno stato che non viene mai raggiunto partendo da q_0).

Esercizio: Si supponga che l'alfabeto contenga, oltre a | e ad $*$, anche due simboli speciali B per "inizio" ed E per "fine", e che la parola iniziale sia presentata come $B\, q_0 \,|\, |\, \ldots \,|\, E$. Si scriva una MT che computa il successore.

Esempio: La seguente macchina calcola la differenza $m - n$, secondo la seguente semplice strategia iterativa: per ogni sbarretta nel secondo numero, si sposta nel primo e ne cancella una. Il secondo argomento serve da *contatore* del numero di iterazioni necessarie. Complicazioni sono dovute al dover tener conto della grandezza relativa dei due argomenti, al fatto che solo n sbarrette delle $n + 1$ che rappresentano n devono contare per l'iterazione, e al dover mantenere invariata la parte centrale della parola input per ritrovare la stessa situazione a ogni ciclo.

$$
\begin{array}{l}
q_0 \,|\, D\, q_0 \\
q_0 * D\, q_1 \\
q_1 \,|\, D\, q_2 \\
q_2 \,|\, D\, q_3 \\
q_3 \,|\, D\, q_3 \\
q_3 * S\, q_4 \\
q_4 \,|\, * q_4 \\
q_4 * S\, q_5 \\
q_5 \,|\, S\, q_5 \\
q_5 * S\, q_6 \\
q_6 \,|\, S\, q_6 \\
q_6 * D\, q_7 \\
q_7 \,|\, * q_7 \\
q_7 * D\, q_0 \\
q_2 * S\, q_8 \\
q_8 \,|\, * q_8 \\
q_8 * S\, q_9 \\
q_9 * S\, q_{10}
\end{array}
$$

Il primo gruppo di istruzioni

$$
\begin{array}{l}
q_0 \,|\, D\, q_0 \\
q_0 * D\, q_1 \\
q_1 \,|\, D\, q_2
\end{array}
$$

$$q_2 \mid D\ q_3$$
$$q_3 \mid D\ q_3$$
$$q_3 * S\ q_4$$

porta la testina alla fine dei due gruppi di sbarrette, e la successiva istruzione cancella l'ultima. Quindi la macchina torna all'inizio nello stato q_7 con

$$q_4 * S\ q_5$$
$$q_5 \mid S\ q_5$$
$$q_5 * Sq_6$$
$$q_6 \mid S\ q_6$$
$$q_6 * D\ q_7$$

e con le due successive istruzioni cancella la prima sbarretta e si sposta a destra riposizionandosi su | nello stato q_0 pronta a iterare.

Il gruppo finale di istruzioni

$$q_2 * S\ q_8$$
$$q_8 \mid\ * q_8$$
$$q_8 * S\ q_9$$
$$q_9 * S\ q_{10}$$

entra in funzione quando a destra c'è inizialmente o è rimasta una sola sbarretta, che viene cancellata e la macchina si riposiziona sul risultato.

Quando $m \geq n$, la macchina dà come risultato $m-n$ e si ferma nello stato q_{10}, posizionata su una sbarretta del risultato. Se $m < n$ si danno due casi; se $m+1 = n$, la macchina si ferma nello stato q_{10} posizionata su una casella con $*$, e il nastro finale non è significativo, numerico, perché non contiene nessun |; se $m+1 < n$, la macchina si ferma nello stato q_1; il risultato sul nastro non è giusto; in questo caso è la lettura dello stato finale che ci avverte che non esiste il risultato: la differenza, tra numeri non negativi, è definita solo se il minuendo è maggiore o uguale del sottraendo.

Esercizi: Disegnare la successione dei nastri e degli stati nei casi $4-2$, $4-0$, $2-3$, $2-4$.

Esercizi: per ognuna delle successive macchine, disegnare la successione dei nastri e degli stati per alcuni input significativi.

Esempio: La macchina *Copy* è una macchina che dato un numero ne scrive uno uguale a partire da due caselle più a destra (della fine del primo). Usiamo un simbolo ausiliario \$ che mettiamo ripetutamente su ogni sbarretta del numero, e in corrispondenza al quale andiamo a scrivere una sbarretta

al fondo della parola. Alla fine, se si vogliono avere il numero originale e la copia, i \$ andranno risostituiti con |.

$$
\begin{array}{l}
q_0 \mid \$\ q_0 \\
q_0 \$\ D\ q_1 \\
q_1 \mid D\ q_1 \\
q_1 * D\ q_2 \\
q_2 * D\ q_3 \\
q_3 * \mid q_5 \\
q_3 \mid D\ q_4 \\
q_4 \mid D\ q_4 \\
q_4 * \mid q_5 \\
q_5 \mid S\ q_5 \\
q_5 * S\ q_6 \\
q_6 * S\ q_7 \\
q_7 \mid S\ q_8 \\
q_8 \mid S\ q_8 \\
q_8 \$\ D\ q_0 \\
q_7 \$\ D\ q_9 \\
q_9\ *\ D\ q_9
\end{array}
$$

Mancano alcune istruzioni per ripristinare le sbarrette al posto dei dollari (esercizio).

In modo analogo si può scrivere una MT che copia la parola da un'altra parte desiderata, ma precisata, del nastro, o che copi una parola qualunque che non sia un numero.

Esercizio: Progettare (programmare) una macchina che copia un numero, come quella di sopra, senza usare un simbolo ausiliario.

La macchina *Copy* è necessaria quando si distrugge uno degli input nel corso del calcolo, ma lo si deve conservare, o per il risultato, o per altri calcoli.

Esercizio: Progettare una macchina che dati due numeri m ed n rappresentati da $|\,|\,|\cdots|\ *\ |\,|\cdots|$ decide quale è il maggiore posizionandosi su (una sbarretta di) quello maggiore, o sul *blank* intermedio se sono uguali.

Esercizio: Usando la macchina precedente, disegnare una macchina che per ogni m ed n calcola $m - n$ se $m \geq n$ ed $n - m$ se $n \geq m$, cioè calcola $|m - n|$.

Esempio: Macchina che esegue la somma di due numeri:

$$q_0 \mid * \; q_0$$
$$q_0 * D \; q_1$$
$$q_1 \mid Dq_1$$
$$q_1 * \mid q_2$$
$$q_2 \mid D \; q_2$$
$$q_2 * S \; q_3$$
$$q_3 \mid * \; q_3.$$

Questa macchina non è elegante, perché è basata su un ragionamento esterno, del programmatore sul numero totale di sbarrette (spiegarlo) - il programmatore peraltro ha il dovere di trovare la soluzione più efficiente; le operazioni si calcolano di solito per iterazione, come nel caso della macchina per la sottrazione.

Per l'addizione, si usi il secondo dei due addendi come contatore, ogni volta che si cancella una sua sbarretta (meno una) si aggiunge una sbarretta davanti al primo, realizzando praticamente il dettato dell'equazione ricorsiva dell'addizione $x + y' = (x + y)'$.

Si noti che due macchine si possono combinare in modo da ottenere una macchina che calcola prima la funzione calcolata dalla prima, quindi sul risultato la funzione calcolata dalla seconda, cioè la *composizione* delle due funzioni; la composizione si ottiene semplicemente unendo i due insiemi di istruzioni dopo aver traslato gli indici degli stati della seconda macchina in modo che non interferiscano con quelli della prima, il q_0 della seconda diventando il q_h finale della prima (dopo aver eventualmente modificato la prima in modo che si fermi su uno stato di indice massimo, e posizionata sulla prima sbarretta a sinistra del risultato).

Osservazione 2. *Esistono macchine che per certi input non si fermano.*

Ad esempio la macchina data dalle istruzioni

$$q_0 \mid Dq_1$$
$$q_1 \mid Dq_0$$
$$q_0 * Dq_0$$

si ferma nello stato q_1 se il numero n in input è pari, e il risultato è n stesso, mentre continua a spostarsi a destra nello stato q_0 se il numero è dispari. (Esercizio: rappresentare le successioni di nastri corrispondenti.)

Esercizio: Scrivere una MT che come nel caso precedente si ferma se e solo se n è pari, ma in tal caso dà come risultato 1.

Esercizio: Scrivere una MT che per ogni numero n dà come risultato 1 se n è pari, e 0 se n è dispari.

Esempio: Macchina per dividere per 2 (anche questa inelegante).

$q_0 \mid D\, q_1$	
$q_1 \mid D\, q_2$	$q_1\mid$ significa: ci sono almeno due $\mid$
$q_2 \mid D\, q_2$	scavalca input
$q_2\ * D\, q_7$	
$q_7 \mid D\, q_7$	scavalca risultato parziale
$q_7\ * \mid q_8$	scrive $\mid$
$q_8 \mid S\, q_8$	torna a sinistra
$q_8\ * S\, q_9$	
$q_9 \mid S\, q_9$	
$q_9 * D\, q_{10}$	arrivata all'inizio
$q_{10} \mid\ * q_{10}$	cancella due $\mid$
$q_{10} * D\, q_{11}$	
$q_{11} \mid\ * q_{11}$	
$q_{11} * D\, q_0$	itera
$q_0 * \$\, q_0$	input numero dispari
$q_1 * \mid q_3$	rimasta una $\mid$ nell'input
$q_3 \mid S\, q_4$	
$q_4 \mid\ * q_5$	
$q_5 * D\, q_6$	

Con le ultime istruzioni a partire da $q_1 * \mid q_3$ si sposta tale sbarretta a fianco del risultato e si posiziona su di esso (funzionano anche se l'input era 0, come è allora anche il risultato).

Invece di $q_0 * \$\, q_0$, si possono dare le seguenti istruzioni, che fanno scrivere a destra il resto 1, cioè $||$.

$q_0 * D q_{12}$
$q_{12}\mid D\, q_{12}$
$q_{12} * D\, q_{13}$
$q_{13} * \mid q_{13}$
$q_{13} \mid D\, q_{14}$
$q_{14} * \mid q_{14}$

Gli ultimi esempi mostrano come tra i compiti che possono risolvere le macchine di Turing ci sia, oltre a quello di calcolare le funzioni, anche quello di riconoscere l'appartenenza o meno a insiemi o relazioni, calcolando la *funzione caratteristica*. La funzione caratteristica di un insieme X è la funzione

$$\varphi_X(n) = \begin{cases} 1 & \text{se } n \in X \\ 0 & \text{se } n \notin X \end{cases}$$

che vale 1 se l'argomento appartiene all'insieme, e 0 se non vi appartiene.

Definizione 1. *Si dice che una funzione*[3] $f : \mathbb{N} \longrightarrow \mathbb{N}$ *è calcolata da una MT M se per ogni input n la macchina M si ferma e dà come risultato il numero $f(n)$.*

Una funzione f si dice effettivamente calcolabile *se esiste una MT che la calcola.*

Si dice che una relazione $R \subseteq \mathbb{N}^n$ è effettivamente calcolabile (o decidibile*) se la sua funzione caratteristica è effettivamente calcolabile.*

Data una macchina M e un input n (per semplicità di notazione consideriamo solo questo caso), o meglio un input che è la codifica scelta per n, indicheremo con $M(n)$ il (numero che decodifica il) risultato scritto sul nastro quando M si ferma e quando esso è un risultato numerico secondo le convenzioni di decodifica. Per tenere conto dei casi in cui non si ferma, o il risultato non è numerico, introduciamo il simbolo $\uparrow$ e scriviamo $M(n) \uparrow$ per dire che M su n non dà il risultato numerico o non dà nessun risultato perché non si ferma. La freccia si legge "diverge". Possiamo anche esprimere con $M(n) \downarrow$ il semplice fatto che la macchina si ferma per l'input n.

Poiché esistono funzioni da naturali in naturali che non sono ovunque definite (ad esempio la sottrazione, la divisione), anche la scrittura $f(n)$ è ambigua, nel senso che se n appartiene al dominio di definizione di f allora $f(n)$ è un numero, altrimenti non esiste. Una funzione $f : \mathbb{N} \longrightarrow \mathbb{N}$ si dice *parziale* se il suo dominio è $\subseteq \mathbb{N}$, senza precisare se uguale o contenuto propriamente, e senza escludere che sia uguale, e la funzione quindi *totale*.

Con $M(n) \simeq f(n)$ indichiamo che o entrambi i membri sono definiti, e allora $M(n) = f(n)$, oppure entrambi sono indefiniti, cioè $f(n)$ non esiste e $M(n) \uparrow$. Ripetiamo perciò la

[3]La notazione $f : \mathbb{N} \longrightarrow \mathbb{N}$ indica una funzione che manda numeri naturali in numeri naturali. La definizione vale anche per funzioni a più argomenti, come la somma, ma non lo scriviamo per non appesantire la notazione. Con opportune codifiche numeriche ci si può ridurre a considerare solo funzioni di un argomento.

Definizione 2. *Si dice che una funzione parziale* $f : \mathbb{N} \longrightarrow \mathbb{N}$ *è calcolata da una MT* M *se per ogni input* n $M(n) \simeq f(n)$.

Una funzione parziale f *si dice* effettivamente calcolabile *se esiste una MT che la calcola.*

Questa definizione trasforma un concetto intuitivo come quello di "calcolabile mediante un metodo preciso, un procedimento effettivo, un algoritmo, una procedura mecanica" (tutti sinonimi) in una definizione precisa e rigorosa. Che la definizione debba ritenersi adeguata è il contenuto della cosiddetta

Tesi di Church[4] Le funzioni che sono calcolabili mediante un procedimento intuitivamente effettivo coincidono con le funzioni effettivamente calcolabili, cioè calcolabili mediante macchine di Turing.

La complicazione della definzione di $\uparrow$ con l'alternativa del risultato non numerico o della non terminazione è dovuta al fatto che, quando una funzione non ha un valore definito, come abbiamo visto nel caso della sottrazione, la macchina può segnalare la non eseguibilità del calcolo in vari modi, dopo un tempo finito. Resta aperto il problema se possa sempre essere cosí, cioè sia sempre possibile per funzioni parziali avere una macchina che o calcola il valore, se c'è, oppure ci segnala con qualche artificio che il valore non c'è.

L'interesse della questione è la seguente. Esistono certamente funzioni numeriche non effettivamente calcolabili, anche se è difficile darne un esempio perché ogni esempio facile non può che essere fornito da una definizione o formula esplicita per la funzione, e in questo caso la funzione risulta calcolabile. Infatti si vede, nello sviluppo della teoria delle MT, che ogni funzione che si ottenga da alcune di base come il successore, l'addizione e simili, per composizione o con l'uso delle strategie usuale per definire nuove funzioni a partire da date (come la composizione, l'iterazione, la ricerca del primo numero che soddisfa una condizione decidibile, e simili) è ancora calcolabile, e in breve si dimostra che ogni funzione definita da una formula aritmetica non troppo complicata lo è.

Questo è uno degli argomenti che vengono portati a favore della tesi di Church. Ma che esistano funzioni non effettivamente calcolabili segue

[4] Alonzo Church, logico americano, ha formulato questa tesi nel 1936; contemporaneamente una tesi analoga è stata formulata da Turing; si parla anche di Tesi di Church-Turing.

facilmente dalla teoria generale delle funzioni numeriche; le funzioni sono molte di più di quelle definibili in un qualsiasi linguaggio.

Se una funzione è effettivamente calcolabile, quindi esiste una macchina che permette di ottenere i suoi valori (quando ci sono), è sempre il caso che il dominio di questa funzione è decidibile? Il dominio è dato dagli argomenti su cui la macchina si ferma, con un risultato numerico. Se è decidbile, come nel caso della sottrazione o della divisione, le risposte della macchina nel caso che l'input non appartenga al dominio sono informazioni utili. Si può ad esempio estendere la funzione ai casi in cui non è definita ponendo dei valori convenzionali. Nel caso della sottrazione, la sottrazione calcolabile totale che viene di solito usata in informatica è la sottrazione troncata

$$m \dot{-} n = \begin{cases} m - n & \text{se } m \geq n \\ 0 & \text{se } m < n. \end{cases}$$

La divisione può venir trasformata in una funzione che dà sempre come risultato la coppia $\langle$quoziente, resto$\rangle$.

Per fare questo, l'importante è che la macchina si fermi, poi guardando il nastro si vede se l'output è un numero o no. Quindi il problema è collegato al seguente: data una macchina, si può decidere a priori quali sono gli input su cui si ferma? A priori vuol dire non ricorrendo all'ovvia soluzione di far partire la macchina e stare a vedere quando si ferma: se infatti siamo su un caso in cui non si ferma, aspettiamo le calende greche senza avere una risposta. Si potrebbe pensare che data la semplicità di ciascuna MT, e il fatto che per scriverla bisogna pensare a come si comporta sui vari input possibili significativi (0 o non zero, pari o dispari e cosí via), la cosa sia possibile.

Decidere il problema della fermata (o della terminazione, o dell'*halt*) significa deciderlo in senso tecnico, cioè mediante una macchina di Turing. Sui nastri delle macchine di Turing si possono scrivere parole che codificano anche alfabeti non mumerici. Con le macchine di Turing si possono eseguire semplici operazioni sintattiche, su alfabeti finiti, qualunque, come confrontare due parole per vedere se sono uguali, copiare una parola, riconononoscere se una parola compare in una successione di parole, e cosí via.

Basta avere due simboli a e b per rappresentare ad esempio con a, aa, aaa e cosí via gli stati $q_0, q_1, \ldots$ e con $b, bb, bbb, \ldots$ i simboli $\alpha_1, \alpha_2, \ldots$; data una macchina particolare, con tre stati e due simboli per esempio, si potrà usare $a**$, $aa*$ e aaa per i tre stati e $b*$, bb per i due simboli, oltre a dd per D e ss per S, in modo da avere una lunghezza fissa delle parole che codificano le

istruzioni; con altri opportuni simboli di separazione si potrà rappresentare ogni quadrupla della macchina e quindi tutte le istruzioni.

Se indichiamo sempre con M la codifica su nastro delle istruzioni di una macchina M, il problema dell'*halt* consiste nel chiedersi se esista una macchina di Turing H tale che per ogni M ed n

$$H(M,n) = \begin{cases} 1 & \text{se } M(n) \downarrow \\ 0 & \text{se } M(n) \uparrow \end{cases}$$

Si potrebbe anche chiedere di meno, che per ogni M ne esista una che si ferma sempre e decide la terminazione di M. Ma si vorrebbe che questa fosse ottenibile in modo effettivo a partire da M, e uniforme, cioè con la stessa costruzione per ogni M. Altrimenti non si saprebbe neanche come incominciare a dimostrarlo, se per ogni M si dovesse inventare una diversa costruzione che decide a priori la sua terminazione. Allora si può dimostrare che la richiesta non sarebbe meno impegnativa.

Decidere il problema della terminazione non deve necessariamente implicare di essere in grado di eseguire i calcoli che ogni macchina esegue. Anzi H non dovrebbe proprio comportarsi in questo modo, altrimenti quando replica un calcolo senza terminazione non termina neanche lei. Si può pensare di poter trovare un modo di analizzare le istruzioni che permetta di scoprire se è possibile che si instauri un ciclo, magari classificando i possibili gruppi di istruzioni "pericolosi" in quanto possono produrre cicli. Ma è chiaro che il solo pensare a una macchina che soddisfi le condizioni di sopra per H suggerisce di chiedersi se esista una macchina capace di replicare i calcoli di ogni altra macchina. Questo sarebbe già un primo passo nella direzione della plausibilità di H.

Una macchina di Turing - insieme di istruzioni - si può pensare in due modi: o che sia fisicamente realizzata in modo da comportarsi per ogni input secondo quanto le istruzioni prevedono, ed eseguire quindi quel solo compito, per ogni input, oppure come istruzioni da consegnare ad un interprete perché questo le esegua.

Quando, data una macchina, si mostra con esempi come la macchina si comporta su determinati input (vedi esercizi ed esempi di sopra), si dice che si stanno interpretando le istruzioni; chi le interpreta e descrive la successione conseguente di nastri e stati segue, applica, le istruzioni; il soggetto che fa questo lavoro nei nostri esempi è il soggetto umano; ma nel compiere il

lavoro tale soggetto mette in gioco solo abilità di lettura e scrittura, oltre alla comprensione di cosa vuol dire fare certi movimenti, come spostare l'occhio e l'indice (testina) a destra e a sinistra, e alla compulsione decisa a farlo di fronte alla lettura delle istruzioni. Queste abilità sono le stesse cha hanno portato a definire il concetto di Macchina di Turing. Questo significa che nell'interpretare una MT il soggetto si comporta come una macchina, mette in gioco solo le abilità che definiscono una MT. Lo stesso soggetto, con gli stessi comportamenti, può interpretare ogni MT ed eseguire ogni calcolo di ogni MT. Questa osservazione rende plausibile che esista una macchina unica capace di fare la stessa cosa. Si tratta di una *macchina di Turing universale*, o MTU.

Una macchina di Turing universale U è una MT tale che innanzi tutto nel suo linguaggio è possibile codificare con una parola ogni insieme di istruzioni per ogni possibile macchina di Turing[5]; quindi per ogni macchina di Turing M ed ogni input adeguato a M, indicata con $U(M, n)$ la macchina U messa in azione su un input costituito dalla codifica della macchina M e da un input n per M, si ha

$$U(M, n) \simeq M(n).$$

Teorema 1. *Esiste una Macchina di Turing Universale.*

Teorema 2. *Non esiste una macchina di Turing H tale che per ogni MT M ed ogni n*

$$H(M, n) = \begin{cases} 1 & \textit{se } M(n) \downarrow \\ 0 & \textit{se } M(n) \uparrow \end{cases}$$

Dimostrazione. La dimostrazione è un esempio di una forma di ragionamento che si ispira alle antinomie (come "questa frase è falsa"), sfruttando un autoriferimento. L'assunzione da cui bisogna partire è che esiste una enumerazione di tutte le macchine di Turing,

$$M_0, M_1, \ldots, M_n, \ldots$$

che è effettiva, vale a dire che esiste una MT la quale, per ogni n, stampa le istruzioni di M_n. Per convincersi dell'esistenza di questa enumerazione è

[5]Per questo è utile aver dimostrato che, almeno per quel che riguarda le macchine che computano funzioni numeriche, ma di fatto per tutte, grazie alla capacità di codifica universale di $0, 1$, è possibile scrivere ogni MT in un linguaggio fissato.

sufficiente ricordare che le MT si identificano con gli insiemi finiti di istruzioni, i quali possono essere enumerati per numero crescente di istruzioni e stati coinvolti e in un ordine alfabetico. Il fatto che si considerano macchine (e le corrispondenti funzioni) non ovunque definite facilita il compito, perché non c'è da verificare nulla, se non la correttezza sintattica.

Supponendo dunque per assurdo che il problema della terminazione fosse decidibile, esisterebbe una macchina H, composta con quella che da n calcola una rappresentazione di M_n, tale che per ogni i ed n

$$H(i,n) = \begin{cases} 1 & \text{se } M_i(n) \downarrow \\ 0 & \text{se } M_i(n) \uparrow, \end{cases}$$

e ne esisterebbe allora una H' tale che per ogni i

$$H'(i) = \begin{cases} 1 & \text{se } M_i(i) \downarrow \\ 0 & \text{se } M_i(i) \uparrow. \end{cases}$$

Per costruire H' da H basta predisporre una macchina che dato i produce $\langle i, M_i \rangle$, e poi inizia a lavorare come H.

Data H', esiste allora H'' tale che per ogni i

$$H"(i) = \begin{cases} 0 & \text{se } M_i(i) \uparrow \\ \uparrow & \text{se } M_i(i) \downarrow. \end{cases}$$

Per avere H'' basta aggiungere ad H' istruzioni che quando questa si ferma fanno un test sul risultato, e se questo è 0 non fanno nulla, se invece è 1 mettono in funzione un ciclo, dato ad esempio da una coppia

$$q_r \mid Dq_s$$
$$q_s \mid Sq_r.$$

H'' è una MT, quindi occorre nella enumerazione, ad esempio è M_h. Ma ora è facile verificare che:

$$M_h(h) \downarrow \text{ se e solo se } M_h(h) \uparrow$$

un assurdo, che si può evitare solo negando l'ipotesi iniziale che esista la MT H.

Corollario 1. *Il problema della terminazione per MT è indecidibile.*

Il corollario non è che una riformulazione del teorema se "indecidibile" significa "non esiste una MT tale che ... "; è invece una conseguenza del

teorema e della Tesi di Church se "decidibile" significa disporre di un metodo effettivo qualsivoglia per rispondere a tutte le domande (infinite) che sono casi particolari del problema.

Un *problema* in questo contesto non è mai una domanda singola, ma un insieme, o una proprietà, e la soluzione del problema è un metodo che si applica uniformemente e dà la risposta (sí o no) per ogni caso possibile.

Prima di vedere un'applicazione alla logica, soffermiamoci a discutere il tipo di argomento usato nella dimostrazione del Teorema 2, che è un argomento per autoriferimento, che porta a una contraddizione. Si potrebbe pensare che con una variante dello stesso si potesse escludere l'esistenza di una macchina universale. L'argomento, per analogia, sarebbe il seguente.

Se esistesse una macchina di Turing universale, ne esisterebbe anche una, che indichiamo ora con U tale che per ogni n

$$U(n) \simeq M_n(n).$$

Ne esisterebbe quindi anche una U^* tale che per ogni n $U^*(n) \simeq U(n) + 1$. Sia e il posto di questa nell'enumerazione di tutte le macchine di Turing, $U^* = M_e$. Allora

$$U(e) \simeq M_e(e) \simeq U^*(e) \simeq U(e) + 1,$$

che sembra in effetti una contraddizione, non fosse che per il segno $\simeq$, che non è il segno $=$. Dall'uguaglianza di sopra si deduce solo che $U(e) \uparrow$.

Dall'indecidibilità del problema della terminazione segue

Corollario 2. *Il problema della validità logica per linguaggi predicativi è indecidibile.*

Dimostrazione Ricordiamo solo i passi fondamentali della dimostrazione. Il problema della fermata viene *ridotto* al problema della validità logica.

a) Innanzi tutto si dimostra che le funzioni effettivamente calcolabili coincidono con quelle che in matematica si chiamano *funzioni ricorsive parziali*, e sono caratterizzate da particolari schemi definitori (la ricorsione primitiva e la minimizzazione) a partire da funzioni di base, come il successore.

b) Quindi si dimostra che le funzioni ricorsive parziali coincidono con quelle che sono definibili da particolari sistemi di equazioni, nel senso che i loro valori si ottengono per mezzo di derivazioni, a partire dalle equazioni

definitorie, con regole di sostituzione e le proprietà riflessiva, simmetrica e transitiva dell'uguaglianza.

In un linguaggio predicativo che contenga una costante 0 e un simbolo funzionale indicato con $'$, i numeri si possono rappresentare con termini, n come $0\overbrace{''\cdots'}^{n}$ con n occorrenze di $'$.

Le equazioni che definiscono per ricorsione la somma a partire dal successore sono

$$\begin{cases} x+0 &= x \\ x+y' &= (x+y)'. \end{cases}$$

Ad esempio $4+3=7$ si deriva da queste equazioni, per $x=4$ e $y=2$, nel seguente modo:

$$\begin{aligned} x+0 &= x \\ x+y' &= (x+y)' \\ 4+0''' &= (4+0'')' \\ 4+0'' &= (4+0')' \\ 4+0''' &= (4+0')'' \\ 4+0' &= (4+0)' \\ &= 4' \\ 4+0''' &= 4'''. \end{aligned}$$

Gli apici passano progressivamente da un addendo dentro la parentesi a fuori, mentre tale addendo scende a 0.

c) Quindi si può associare in modo effettivo a ogni calcolo di una macchina di Turing (per una funzione numerica, ricorsiva parzale) una deduzione da un sistema di equazioni (formule). Infine grazie a questa corrispondenza, si vede che se il problema della validità logica, quindi quello della conseguenza logica, quindi quello della derivabilità, fossero decidibili, anche il problema della terminazione lo sarebbe.

Si dice in breve: *la logica predicativa è indecidibile*, a differenza di quella proposizionale. Questo significa che c'è almeno un linguaggio per le cui formule il problema della validità è indecidibile, quello delle equazioni usato nella dimostrazione. Per linguaggi particolari invece può esistere un metodo

di decisione; ad esempio per i linguaggi monadici il metodo delle tavole semantiche è un metodo di decisione.

La dimostrazione del teorema di indecidibilità del problema della terminazione fornisce un esempio di un insieme dalle caratteristiche particolarmente interessanti, un insieme non decidibile ma *semidecidibile*. L'insieme

$$K = \{n : M_n(n) \downarrow\}$$

è l'insieme per cui si cerca un metodo di decisione, che non esiste, quindi K non è decidibile. Ma c'è un metodo di decisione parziale intuitivo: dato n, si individua la n-esima macchina M_n, la si fa partire su n, e si aspetta. Se $n \in K$, ovvero se $M_n(n) \downarrow$, dopo un tempo finito si ha la risposta, perché la macchina si ferma. Se $n \notin K$, non si ha mai nessuna risposta. Un procedimento del genere dà la risposta, giusta, dopo un numero finito di passi, per ogni caso positivo, mentre non dà nessuna risposta per i casi che di fatto sono negativi. Si noti che il procedimento è effettivo, cioè è realizzato da una macchina (che non sempre si ferma).

Un problema per cui esista un metodo di decisione parziale come quello descritto si dice *semidecidibile*. Un insieme semidecidibile si può pensare come il dominio di una funzione parziale, una funzione parziale ricorsiva calcolata dalla macchina che implementa il procedimento sopra descritto, e per esempio quando si ferma dà come risultato 1. L'esempio di pag. 9 mostra come un insieme possa essere semidecidibile, grazie a un metodo di decisione parziale, ma questo non esclude che sia anche decidibile, se esiste un metodo migliore, che sia sempre definito e corretto.

Si vede facilmente che un insieme è decidibile se e solo se sia l'insieme sia il suo complemento sono semidecidibili: si mettono in funzione, per un n qualunque dato, sia il metodo di decisione parziale per l'insieme, sia quello per il complemento, e dopo un numero finito di passi da una della due macchine si ha la risposta.

Per gli insiemi di numeri si usa anche la terminologia *ricorsivo* per decidible e *ricorsivamente enumerabile* per semidecidibile. Il motivo di quest'ultimo termine sta nel fatto che, mettendo in funzione un procedimento di decisione parziale, si possono ottenere via via gli elementi dell'insieme, "sputati" per cosí dire dalla macchina man mano che questa si accorge che un numero appartiene all'insieme.

Un insieme che ha le stesse caratteristiche di K è l'insieme delle formule logicamente valide (di un linguaggio per cui il problema della validità è inde-

cidibile). Un metodo di decisione parziale per la validità logica è fornito dalle tavole semantiche, che si chiudono se non esiste il controesempio, ma possono andare avanti a essere sviluppate all'infinito nel caso ci sia un controesempio, ma infinito.

Se l'insieme delle formule logicamente valide è ricorsivamente enumerabile, anche quello delle formule insoddisfacibili lo è (esercizio; suggerimento: una formula è logicamente valida se e solo se la sua negazione è insoddisfacibile). Ma allora per la precedente osservazione l'insieme delle formule soddisfacibili non è neanche ricorsivamente enumerabile, e il problema della soddisfacibilità neanche semidecidible.

Esiste una gerarchia infinita di problemi sempre più complicati, o sempre meno effettivamente dominabili; e al di sopra di tutti esistono poi insiemi, come quello degli enunciati veri dell'aritmetica, che non sono neanche definibili da formule aritmetihe.

Nella teoria delle funzioni calcolabili esistono altri risultati di indecidibilità o risultati di non ottimizzazione, ad esempio

Teorema 3. *Non esiste una MT O tale che per ogni n $M_{O(n)}$ sia equivalente a M_n e con un numero di istruzioni minimale tra quelle equivalenti.*

Quasi tutti si si dimostrano con tecniche simili a quello sulla terminazione (sfruttando, ma evitando - come ha insegnato Gödel - argomenti paradossali).