Macchine di Turing: *istruzioni per l'uso*

di GABRIELE LOLLI *con la collaborazione di* STEFANO BARATELLA *e* DOMENICO LUMINATI

Come funziona una macchina di Turing? Vediamolo in breve con qualche spiegazione, qualche esempio, e, per chi vorrà provare, qualche esercizio…

Per prima cosa, introduciamo un formalismo per descrivere le istruzioni eseguibili da una di quelle macchine di Turing (MT) di cui abbiamo già detto nell'articolo di p. 26. Possiamo pensare a questo formalismo come ad un rudimentale linguaggio di programmazione in cui esprimere comandi elementari quali la ri-scrittura del contenuto di una cella di memoria o l'accesso ad una specifica cella di memoria. Come in ogni linguaggio di programmazione, dobbiamo specificare l'*alfabeto* in cui le istruzioni sono scritte: esso è composto da due insiemi finiti, quello dei simboli di lettura e scrittura e quello degli stati, e inoltre dalle lettere D (per "destra") e S (per "sinistra").
I simboli per gli stati e quelli di lettura e scrittura possono essere scelti in modo arbitrario, ma conviene tenerli distinti. Tra i simboli di lettura e scrittura si include sempre il simbolo speciale $*$, che segnala quando una cella è vuota.
Per convenzione, all'inizio di una computazione, la macchina si trova in un particolare *stato iniziale* q_0 e c'è almeno un'istruzione che inizia con q_0.
Per l'arresto sono possibili varie soluzioni: per esempio si potrebbe chiedere che la macchina si fermi sempre in un particolare *stato finale* q_H (H per *halt*).
Un'istruzione è una parola di quattro lettere di uno di questi due tipi:

$$q\ a\ b\ r$$
$$q\ a\ M\ r,$$

dove q e r sono due stati, a e b sono simboli di lettura e scrittura e la lettera M può prendere i valori S oppure D. È da notare che nelle istruzioni q non è necessariamente diverso da r come a non è necessariamente diverso da b.
L'effetto di un'istruzione del primo tipo è il seguente: se l'unità di controllo è nello stato q e la testina è posizionata su una cella che contiene il simbolo a, allora nella cella viene sovrascritto il simbolo b e l'unità di controllo passa nello stato r.
Potete immaginare l'effetto di un'istruzione del secondo tipo: se l'unità di controllo è nello stato q e la testina è posizionata su una cella che contiene il simbolo a, allora la testina si sposta di una cella a sinistra oppure a destra (a seconda del valore di M) e l'unità di controllo passa nello stato r.
Dobbiamo anche prevedere il caso in cui l'unità di controllo si trovi nello stato q e la testina legga il simbolo a, ma nessuna delle istruzioni inizi con la coppia $q\ a$. In tal caso la macchina si ferma.

Richiediamo infine che una MT sia non contraddittoria, cioè che non contenga due istruzioni diverse che iniziano con la stessa coppia. (Altrimenti la macchina non saprebbe quale tra le due istruzioni eseguire!)
Vediamo un esempio semplice: supponiamo di avere un alfabeto di lettura e scrittura con due soli simboli | e $*$ e due soli stati: uno stato iniziale q_0 (che per un calcolatore reale potrebbe corrispondere allo stato di inizio dell'esecuzione di un programma) e uno stato finale q_H (che, continuando l'analogia con un calcolatore reale, potrebbe corrispondere allo stato di fine nell'esecuzione di un programma). A partire dalla cella del nastro su cui è posizionata la testina, la MT

$$q_0\ *\ D q_0$$
$$q_0\ |\ *\ q_H$$

esamina una ad una tutte le celle a destra di questa e, se ne trova uno, cancella il primo | che incontra, poi si ferma. Se non ci sono | sul nastro, la macchina continua ad esaminare celle sempre più lontane alla destra di quella iniziale, senza mai fermarsi.

AVANTI IL PROSSIMO!

Come alfabeto di simboli per i calcoli numerici è sufficiente un insieme di due simboli, la sbarretta | e l'asterisco $*$. Questa affermazione è un profondo e difficile risultato della teoria delle MT o più precisamente del lavoro di Gödel: i numeri naturali 0, 1, 2, . . . sono uno strumento di codifica universale, e i numeri naturali si possono rappresentare con due soli simboli: la sbarretta | che serve a scrivere n come

$$\underbrace{|\ |\ \ldots\ |\ |}_{\text{n+1 volte}}$$

e l'asterisco $*$ come segno divisorio. Per questa ragione, d'ora in poi lavoreremo sempre con l'alfabeto di simboli formato da | e $*$.
Una coppia di numeri (m, n) è rappresentata in input da $|\ |\ |\ \cdots\ |\ *\ |\ |\ \cdots\ |$, con m+1 sbarrette seguite da $*$ e da n+1 sbarrette. Ad esempio, la sequenza $|\ |\ |\ *\ |\ |$ rappresenta la coppia (2, 1), mentre $|\ |\ |\ |\ *\ |$ rappresenta (3, 0).

(Vedendo quest'ultimo esempio, dovrebbe essere chiaro perché usiamo n+1 sbarrette per rappresentare il numero naturale n.).
Nella pratica, è conveniente concordare che, all'inizio di una computazione, la testina sia posizionata sul primo (a sinistra) simbolo del nastro diverso da $*$.
Quando la macchina si ferma, per l'interpretazione del risultato numerico sono possibili diverse alternative, la cui adozione influenza la scrittura della macchina: di solito si richiede che il numero sia rappresentato in output da una parola $||\cdots|$ e che queste siano le uniche celle non vuote (ossia, non contrassegnate da $*$) del nastro.
Ecco un esempio di MT: le istruzioni seguenti

$$q_0\ |\ D q_0$$
$$q_0 * |\ q_1$$

costituiscono una MT con due stati (ricordiamo che q_0 è lo stato iniziale della nostra macchina) che, ricevuta in input la rappresentazione del numero naturale n, restituisce come output quella di n+1, dopo essersi fermata nello stato q_1 con la testina posizionata sulla | più a destra del risultato. Dunque la macchina computa il successore di ogni numero naturale.
La successione dei nastri e degli stati per il calcolo del successore di 2 è quella di Figura 1.
Se aggiungiamo

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

otteniamo una macchina con tre stati che computa la stessa funzione, ma si ferma nello stato q_2 posizionata sulla prima | di sinistra del risultato.
Poiché una MT è l'insieme delle sue istruzioni, le due macchine sopra sono chiaramente diverse, ma si possono chiamare equivalenti.
Ecco un esempio di una MT che su certi input non si ferma. La macchina data dalle istruzioni

$$q_0\ |\ D q_1$$
$$q_1\ |\ D q_0$$
$$q_0 * D q_0$$

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.
Ora un esercizio per voi: sapete scrivere una MT che non si fermi su nessun input? La richiesta è troppo alta? Ok, eccovi qualche suggerimento: potreste scrivere le istruzioni in modo che la testina continui a spostarsi a destra di un passo, qualunque sia il simbolo letto, oppure – se volete risparmiare nastro – in modo che continui a muoversi avanti e indietro tra due celle adiacenti, indipendentemente dal loro contenuto, ma di sicuro se ci pensate sapete trovare tante altre possibili soluzioni: un po' di fantasia e inventiva e il gioco è fatto! Buon lavoro!

Una Macchina di Turing per la somma

Lavoriamo con l'alfabeto di simboli formato da | e $*$. La seguente MT calcola la somma $m + n$, secondo una strategia iterativa:

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

Per capire il suo funzionamento, consideriamo l'esempio 2 + 3, per cui la successione di nastri e stati è quella nelle Figure 2, 3, 4 e 5.

Eppure i millepiedi non inciampano...

Millepiedi x email.jpg di Angelo Bonomo, http://www.flickr.com/photos/32414901@N05/3030184337

Pensavate che le macchine di Turing facessero cose strabilianti e siete delusi di trovare qui solo esempi da… neonati? Fare la somma di due numeri naturali, trovare il successore di un numero naturale dato: queste macchine di Turing così complesse vi sembrano uno spreco di tempo e di riflessione?
E invece San Bernardo diceva che noi siamo nani seduti sulle spalle di giganti, ovvero che le nostre conoscenze sono frutto delle conoscenze e dei progressi fatti da coloro che ci hanno preceduto. E che i risultati che hanno ottenuto, che ci sembrano così semplici, spesso hanno rappresentato un passo concettuale più profondo e importante di tanti dei nostri presunti progressi scientifici.
Questa frase si adatta perfettamente al mondo dell'informatica: le incredibili capacità dei processori moderni si appoggiano sul fatto che le operazioni di base come la somma o il prodotto sono state via via "interiorizzate" nel funzionamento delle nostre macchine, così che non "vediamo più" come queste "ragionano" Fare una somma non è semplice, o meglio, è semplice perché abbiamo imparato a farlo! E abbiamo imparato a farlo perché, inconsciamente, il nostro cervello ha trovato un processo (un algoritmo) che gli permette di fare rapidamente un tale calcolo.
Questo algoritmo (parlando in termini informatici, potremmo chiamarlo macroistruzione) ci permette di fare la somma di due numeri e noi, sommando 2 con 3, non ci accorgiamo della complessità del calcolo...
Per spiegare meglio questo fatto consideriamo la somma: 1+2+3=6. Come l'abbiamo calcolato? Abbiamo preso in considerazione 1 e 2, ne abbiamo calcolato la somma, l'abbiamo a sua volta sommata a 3 e abbiamo ottenuto 6. Oppure abbiamo sommato 2 e 3, abbiamo ottenuto 5, l'abbiamo sommato a 1 per ottenere ancora 6. Questi due processi sono differenti, ma portano allo stesso risultato perché la somma di interi è un'operazione associativa.
E noi tutto questo ragionamento lo facciamo inconsciamente, magari senza saperlo: se dovessimo procedere intenzionalmente, la mancanza di abitudine a controllare tutti i passi ci porterebbe probabilmente a calcolare molto più lentamente e a commettere un sacco di errori! Normale: come dice Jan Stewart in *Terribili simmetrie* (Bollati Boringhieri, 1995) se un millepiedi dovesse riflettere sull'algoritmo che regola la sua camminata, inciamperebbe quasi ad ogni passo..

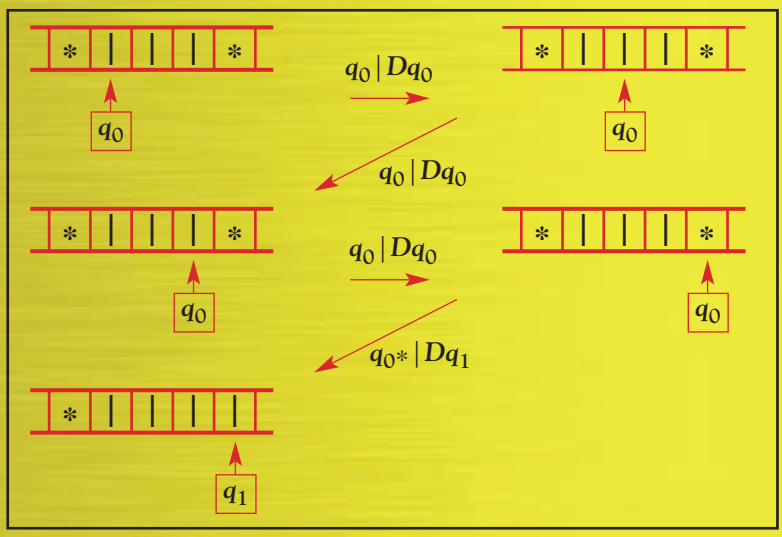

Figura 1

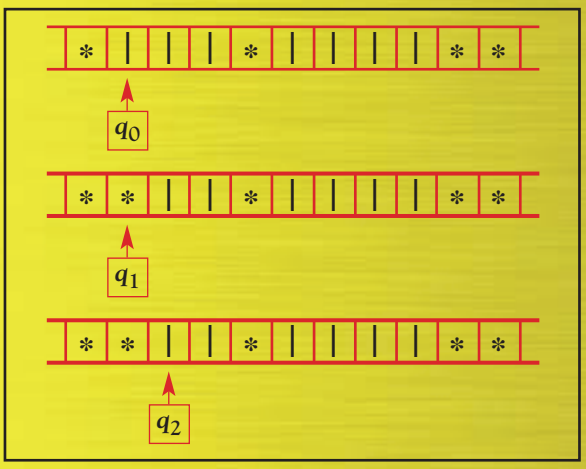

Figura 2

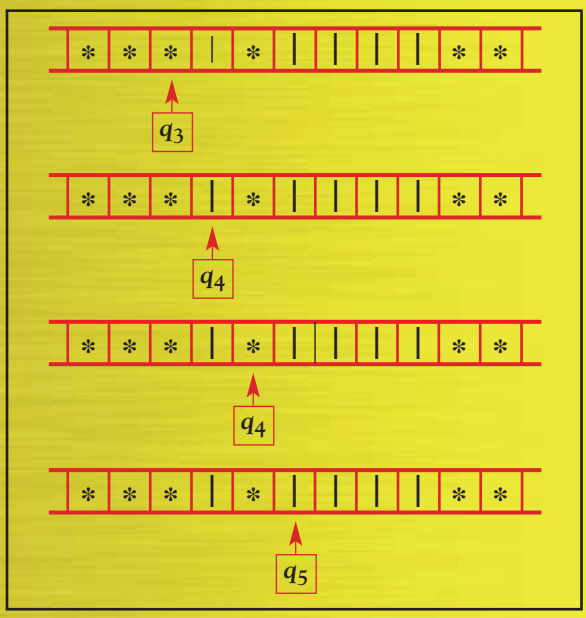

Figura 3

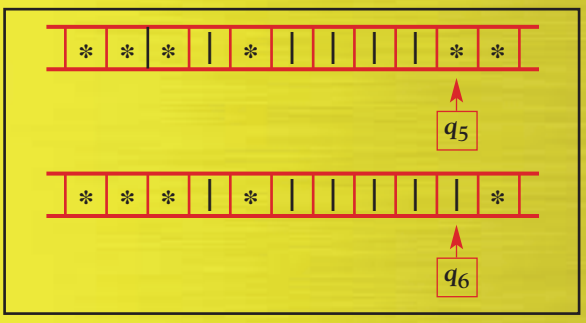

Figura 4

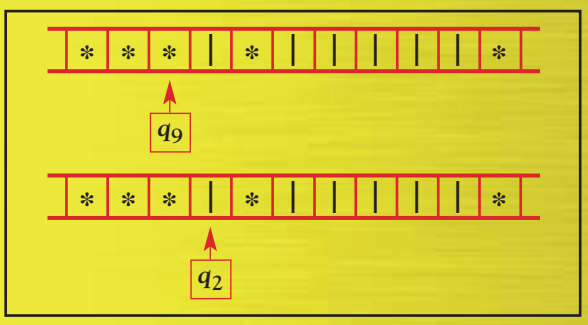

Figura 5

Ora l'idea ispiratrice del programma è che, dopo aver cancellato la prima sbarretta del primo addendo, si usa quanto rimane del primo addendo come contatore effettivo: per ogni sbarretta rimasta del primo addendo, dopo averla cancellata, si va a metterne una alla fine del secondo addendo, come si vede nelle Figure 3 e 4.
Quindi si torna a sinistra fino ad arrivare alla configurazione della Figura 5.
E si itera.
La macchina si ferma nello stato q_{10} posizionata sulla prima sbarretta del risultato.

C'è anche una macchina molto più semplice che esegue la somma di due numeri:

$$q_0 \mid * q_0$$
$$q_0 * Dq_1$$
$$q_1 * Dq_5$$
$$q_1 \mid * q_2$$
$$q_2 * Dq_3$$
$$q_3 \mid Dq_3$$
$$q_3 * \mid q_4$$
$$q_4 \mid Sq_4$$
$$q_4 * Dq_5$$

ma non è elegante, perché è basata su un ragionamento esterno, del programmatore, sul numero totale di sbarrette. Infatti, la macchina cancella le prime due sbarrette, se ce ne sono due, e ne inserisce una al posto del $*$ di separazione dei due numeri (vi suggeriamo di simularne il funzionamento sull'input (3, 2)).
Ecco un esercizio per voi: provate a modificare la MT precedente in modo che la nuova macchina, avuta in input la rappresentazione di una coppia (m, n), cancelli la sbarretta più a sinistra della rappresentazione di m, quella più a destra della rappresentazione di n e infine sostituisca il simbolo di separazione $*$ con $|$.
Qual è il risultato prodotto da questa MT a partire dalla coppia (m, n)?