Matrice di transizione e grafo di transizione

La funzione di transizione si presta ad una rappresentazione tabellare, detta *matrice di transizione* dove le colonne corrispondono ai simboli ai caratteri osservabili sotto la testina (elementi di $\overline{\Gamma}$) e le righe ai possibili stati interni della macchina (elementi di Q). Gli elementi della matrice sono triple che rappresentano il nuovo stato, il carattere che viene scritto e lo spostamento della testina.

La funzione di transizione si presta anche ad una rappresentazione a grafo, detta *grafo di transizione*, dove i nodi sono etichettati con gli stati e gli archi con tripla composta dal carattere letto, il carattere scritto e lo spostamento della testina.

L'applicazione della funzione di transizione ad una configurazione si chiama *passo* o *mossa*.

Computazione di una macchina di Turing

La funzione di transizione permette di definire la relazione di transizione $\vdash$ tra configurazioni, che associa ad una configurazione la configurazione successiva.

Definizione 5 *Data una macchina di Turing* $\mathcal{M} = \langle \Gamma, \not{b}, Q, q_0, F, \delta \rangle$ *e due configurazioni* c *e* c'*, avremo che* $(c \vdash_{\mathcal{M}} c')$ *se e solo se l'applicazione di* δ *a* c *produce la configurazione* c'*.*

Definizione 6 *Una computazione di una macchina di Turing* $\mathcal{M} = \langle \Gamma, \not{b}, Q, q_0, F, \delta \rangle$ *una sequenza, eventualmente infinita, di configurazioni* $\langle c_1, c_2, \ldots, c_i, \ldots \rangle$ *tali che:*

$c_1 \vdash_{\mathcal{M}} c_2 \vdash_{\mathcal{M}} \ldots \vdash_{\mathcal{M}} c_i \vdash_{\mathcal{M}} \ldots$

Usiamo la chiusura riflessiva e transitiva di $\vdash$, indicata con $\vdash^*$, per denotare la relazione tra due configurazioni c e c' che esprime l'esistenza di una computazione che da c porta a c' tramite un numero finito (eventualmente nullo) di transizioni.

Convenzione. In ogni computazione può esistere al più una configurazione finale; se la macchina raggiunge una configurazione finale il calcolo termina.

Una computazione infinita non ha configurazioni finali.

Definizione 7 *Una computazione finita* $c_1 \vdash_{\mathcal{M}} c_2 \vdash_{\mathcal{M}} \ldots \vdash_{\mathcal{M}} c_n$ è massimale *se non esiste una configurazione* c *tale che* $c_n \vdash c$.

Una computazione massimale si conclude o con una configurazione finale o con una configurazione in cui non è definita la funzione di transizione.

Riconoscimento di linguaggi

Le macchine di Turing deterministiche rappresentano uno strumento teorico potente e generale per affrontare in maniera adeguata il problema del riconoscimento di linguaggi. In tale contesto si parla della macchine di Turing come di dispositivi *riconoscitori*.

Definizione 8 *Una computazione massimale* $c_0 \vdash_{\mathcal{M}} c_2 \vdash_{\mathcal{M}} \ldots \vdash_{\mathcal{M}} c_n$ è accettante *se* c_0 *è iniziale e* c_n *è finale.*

Definizione 9 *Una computazione massimale* $c_0 \vdash_{\mathcal{M}} c_2 \vdash_{\mathcal{M}} \ldots \vdash_{\mathcal{M}} c_n$ è rifiutante *se* c_0 *è iniziale e* c_n *non è finale.*

Convenzioni.
Computazione accettante: responso affermativo.
Computazione rifiutante: responso negativo.
Computazione non terminante: nessun responso.

Formalmente:

Definizione 10 *Data una macchina di Turing* $\mathcal{M} = \langle \Gamma, \not b, Q, q_0, F, \delta \rangle$, *e dato un alfabeto di input* $\Sigma \subseteq \Gamma$, *una stringa* $x \in \Sigma^*$ *è* accettata *(*rifiutata*) da* $\mathcal{M}$ *se esiste una computazione di accettazione (di rifiuto) con* $c_0 = q_0 x$.

Questa definizione implica che la macchina puó anche non terminare.
Si puó sapere se una computazione termina?
In altri termini, esiste una macchina $\overline{\mathcal{M}}$ che puó dire se $\mathcal{M}$ termina per input x (*problema della terminazione*)?

Dato che la computazione di una macchina di Turing può terminare oppure no, appare necessario distinguere i concetti di *riconoscimento* e di *accettazione* di un linguaggio.

Informalmente

- una macchina di Turing $\mathcal{M} = \langle \Gamma, \not b, Q, q_0, F, \delta \rangle$ *riconosce* un linguaggio L se per ogni $x \in \Sigma^*$, $\mathcal{M}$ è in grado di stabilire se $x \in L$ oppure no.

- una macchina di Turing $\mathcal{M} = \langle \Gamma, \not b, Q, q_0, F, \delta \rangle$ *accetta* un linguaggio L se per tutte e sole le $x \in L$, $\mathcal{M}$ è in grado di stabilire tale appartenenza, ma se $x \notin L$, $\mathcal{M}$ non garantisce un comportamento prestabilito.

Formalmente:

Definizione 11 *Sia* $\mathcal{M} = \langle \Gamma, \not{b}, Q, q_0, F, \delta \rangle$ *una macchina di Turing deterministica.* $\mathcal{M}$ riconosce *(*decide*) un linguaggio* $L \subseteq \Sigma^*$ *(con* $\Sigma \subseteq \Gamma$*) sse*
per ogni $x \in \Sigma^*$ *esiste una computazione massimale* $q_0 x \vdash_{\mathcal{M}} wqz$*, con* $w \in \Gamma\overline{\Gamma}^* \cup \{\varepsilon\}$*,* $z \in \overline{\Gamma}^*\Gamma \cup \{\not{b}\}$*, e dove* $q \in F$ *sse* $x \in L$*.*

Nota bene. Non è detto che una macchina di Turing $\mathcal{M}$ possa sempre riconoscere un linguaggio. Infatti ciò è possibile se e solo se $\mathcal{M}$ si ferma per qualunque input x.

Definizione 12 *Sia* $\mathcal{M} = \langle \Gamma, \not{b}, Q, q_0, F, \delta \rangle$ *una macchina di Turing deterministica.* $\mathcal{M}$ accetta *un linguaggio* $L \subseteq \Sigma^*$ *(con* $\Sigma \subseteq \Gamma$*) sse*
$L = \{x \in \Sigma^* | q_0 x \vdash_{\mathcal{M}} wqz\}$*, con* $w \in \Gamma\overline{\Gamma}^* \cup \{\varepsilon\}$*,* $z \in \overline{\Gamma}^*\Gamma \cup \{\not{b}\}$*, e* $q \in F$*.*

Un linguaggio accettato da una macchina di Turing è detto semidecidibile.

Nota bene. Se un linguaggio è decidibile, allora è anche semidecidibile.

Calcolo di funzioni

Le macchine di Turing deterministiche possono essere viste anche come *trasduttori*, cioè come dispositivi generali capaci di realizzare il calcolo di funzioni parziali, definite su domini qualunque.

Definizione 13 *Sia* $\mathcal{M} = \langle \Gamma, \not b, Q, q_0, F, \delta \rangle$ *una* MDT *deterministica (un trasduttore) e* $f : \Sigma^* \longmapsto \Sigma^*$, *(*$\Sigma \subseteq \Gamma$*);* $\mathcal{M}$ calcola *la funzione* f *sse per ogni* $x \in \Gamma^*$*:*
1. se $x \in \Sigma^*$ *e* $f(x) = y$ *allora* $q_0 x \vdash_{\mathcal{M}} x \not b q y$, *con* $q \in F$*;*
2. se $x \notin \Sigma^*$, *oppure se* $x \in \Sigma^*$ *e* $f(x)$ *non è definita, allora non esistono computazioni massimali, oppure esistono computazioni massimali che non terminano in uno stato finale.*

Codifica dei dati

Se $\mathcal{D}$ è il dominio e $\mathcal{C}$ il codominio della funzione f, x ed $f(x)$ sono rispettivamente rappresentazioni (*codifiche*) degli elementi di $\mathcal{D}$ e $\mathcal{C}$ nell'alfabeto della macchina.

Esempio. Nel caso del calcolo di una funzione intera, si può usare un qualunque alfabeto Σ e codificare i numeri in base $|\Sigma|$. Nel calcolo del valore $m = f(n)$ la configurazione iniziale sarà:

$$q_0 x$$

dove x è la codificazione dell'intero n nell'alfabeto prescelto Σ. Al termine della computazione sul nastro vi sarà la rappresentazione codificata di m.

Calcolabilitá secondo Turing

Formalizziamo il concetto di calcolo secondo Turing (avendo a disposizione una definizione formale di algoritmo).

Definizione 14 *Un linguaggio è* decidibile secondo Turing *(T-decidibile) se esiste una macchina di Turing che lo riconosce.*

Il concetto di decidibilità può essere esteso alla valutazione di *predicati*, vale a dire di funzioni con codominio {vero, falso}.

Definizione 15 *Un linguaggio è* semidecidibile secondo Turing *(T-semidecidibile) se esiste una macchina di Turing che lo accetta.*

Anche il concetto di semidecidibilità può essere esteso alla valutazione di *predicati*.

Problema: Esistono linguaggi T-semidecidibili non T-decidibili?

Definizione 16 *Una funzione è detta* calcolabile secondo Turing *(T-calcolabile) se esiste una macchina di Turing che la calcola.*

I concetti di decidibilità e di calcolabilità basati sul modello di calcolo introdotto da Turing hanno una validità che travalica il modello stesso.

Tutti i tentativi fatti nell'ambito della logica matematica e dell'informatica teorica di definire modelli di calcolo basati su modelli e paradigmi alternativi alle macchine di Turing (e.g., macchine a registri, funzioni ricorsive, ecc.) hanno condotto a caratterizzare classi di insiemi decidibili e di funzioni calcolabili equivalenti a quelle introdotte da Turing.

Tesi di Church. Ogni procedimento algoritmico espresso nell'ambito di un qualunque modello di calcolo è realizzabile mediante una macchina di Turing.

Quindi si pu parlare di linguaggio decidibile e di funzione calcolabile senza fare esplicito riferimento alla macchina di Turing.

Macchine di Turing multinastro

Il modello di macchina di Turing a singolo nastro non è agevole (e.g., non esiste una chiara separazione fra nastro di input e memoria di lavoro).

In una macchina di Turing *multinastro* si hanno più nastri contemporaneamente accessibili in scrittura e lettura che vengono consultati e aggiornati tramite più testine.

Il multinastro non offre maggior potere computazionale, ma consente di affrontare agilmente determinate classi di problemi.

Esempio: un nastro per l'input (a sola lettura), uno per l'output (a sola scrittura), più nastri a lettura /scrittura. Quelli di input e di output sono accessibili in una sola direzione.

Definizione 17 *Una macchina di Turing a $k \geq 2$ nastri (MTM) è una sestupla $\mathcal{M}^{(k)} = \langle \Gamma, \not b, Q, q_0, F, \delta^{(k)} \rangle$, dove $\Gamma = \cup_{i=1}^{k} \Gamma_i$ è l'unione dei k alfabeti di nastro $\Gamma_1 \ldots \Gamma_k$.*
La funzione di transizione è definita come

$$\delta^{(k)} : (Q - F) \times \overline{\Gamma}_1 \times \ldots \times \overline{\Gamma}_k \longmapsto Q \times \overline{\Gamma}_1 \times \ldots \times \overline{\Gamma}_k \times \{d, s, i\}^k.$$

La macchina esegue una transizione a partire da uno stato interno q_i e con le k testine (una per nastro) posizionate sui caratteri $a_{i_1}, \ldots, a_{i_k}$, e con $\delta^{(k)}(q_i, a_{i_1}, \ldots, a_{i_k}) = (q_j, a_{j_1}, \ldots, a_{j_k}, z_{j_1}, \ldots, z_{j_k})$, con $z_{j_l} \in \{d, s, i\}$.

Configurazioni e transizioni di MTM

La *configurazione istantanea* di una MTM deve descrivere lo stato, i nastri e i caratteri osservati.

Definizione 18 *Una* configurazione istantanea *di una macchina di Turing multinastro è una stringa del tipo*

$$q \sharp \alpha_1 \uparrow \beta_1 \sharp \alpha_2 \uparrow \beta_2 \sharp \ldots \sharp \alpha_k \uparrow \beta_k$$

dove $\alpha_i \in \Gamma_i \overline{\Gamma}_i^* \cup \{\varepsilon\}$ e $\beta_i \in \overline{\Gamma}_i^* \Gamma_i \cup \{\not b\}$, con $\uparrow$ il simbolo che indica la posizione di ogni testina e $\sharp$ è un separatore.

Definizione 19 *Una configurazione* $q\sharp\alpha_1 \uparrow \beta_1\sharp \ldots \sharp\alpha_k \uparrow \beta_k$ *si dice* iniziale *se* $\alpha_i = \varepsilon$, $\beta_1 \in \Gamma_1^*$, $\beta_i = Z_0$ $(i = 2, \ldots, k)$, *e* $q = q_0$.

Definizione 20 *Una configurazione* $q\sharp\alpha_1 \uparrow \beta_1\sharp \ldots \sharp\alpha_k \uparrow \beta_k$ *si dice* finale *se* $q \in F$.

Definizione 21 *L'applicazione della funzione di transizione* $\delta^{(k)}$ *ad una configurazione si dice* transizione o mossa o passo computazionale *di una* MTM.

Estensione delle definizioni per MDT. In particolare:

Definizione 22 *Una* MTM $\mathcal{M}$ *calcola la funzione* $f(x)$ *se*

$$q_0\sharp \uparrow x\sharp \uparrow Z_0\sharp \ldots \sharp \uparrow Z_0 \vdash_{\mathcal{M}}^* q\sharp \uparrow x\sharp \uparrow f(x)\sharp \ldots \sharp \uparrow \not b, \text{ con } q \in F.$$

Equivalenza fra MDT **e** MTM

Macchine di Turing e macchine di Turing multinastro hanno differente potere computazionale?
In altri termini, i linguaggi accettati (risp., le funzioni calcolate) da una MDT sono gli stessi accettati (calcolate) da un MTM??

Teorema 23 *Data una macchina di Turing* $\mathcal{M}^{(k)} = \langle \Gamma, \not b, Q, q_0, F, \delta^k \rangle$, *esiste una macchina a un nastro che simula* t *passi di* $\mathcal{M}$ *in* $O(t^2)$ *transizioni usando un alfabeto di dimensione* $O((2|\Gamma|)^k)$.

Come conseguenza del teorema, MDT e MTM hanno lo stesso potere computazionale.