Macchine di Turing non deterministiche ($MTND$)
Si può estendere il modello delle macchine di Turing deterministiche consentendo transizioni che non sono determinate univocamente. Formalmente ciò è ottenuto imponendo che la funzione (parziale) di transizione δ_N sia definita sull'insieme delle parti del codominio della funzione di transizione deterministica.

Definizione 24 *Una macchina di Turing non deterministica ($MTND$) è una sestupla $\mathcal{M} = \langle \Gamma, \not{b}, Q, q_0, F, \delta \rangle$, dove:*
Γ è l'alfabeto dei simboli di nastro
$b \notin \Gamma$ è il carattere speciale di cella vuota
Q è un insieme non vuoto e finito di stati
$q_0 \in Q$ è lo stato iniziale
$F \subseteq Q$ è l'insieme degli stati finali
δ è la funzione di transizione (parziale), definita come
$\delta : Q \times \overline{\Gamma} \longmapsto \mathcal{P}(Q \times \overline{\Gamma} \times \{d, s, i\})$.

Per ogni MTND è sempre possibile determinare il numero massimo di terne aventi gli stessi primi due elementi. Tale valore $\nu(\mathcal{M})$, chiamato *grado di non determinismo di MTDN*, si deriva in modo immediato esaminando la lista di triple nelle celle della matrice di transizione.

$\nu(\mathcal{M}) = max|\delta_N(q_i, \sigma_j)|$

Date due configurazioni c e c', indicheremo ancora con $c \vdash_{\mathcal{M}} c'$ il fatto che c e c' siano legate dalla relazione di transizione definita dalla MTND $\mathcal{M}$, cioè che c' è ottenuta in corrispondenza ad una *scelta di applicazione di δ_N su c*.

Durante il suo funzionamento una macchina *non deterministica* dà luogo ad un numero di computazioni deterministiche distinte che può crescere esponenzialmente in funzione del numero di istruzioni eseguite ad ogni passo, cioè in funzione del grado di non determinismo.

Ognuna di queste computazioni (finita o infinita) parte nello stato iniziale della macchina di Turing non deterministica e differisce dalle altre per almeno un'istruzione; inoltre, se una computazione deterministica termina, allora essa deve trovarsi in uno stato finale.

Le computazioni possono essere rappresentate mediante un *albero*, i cui rami sono costituiti da tutte le computazioni che la macchina realizza a partire dalla configurazione iniziale.

Nel corso considereremo soltanto MTND del tipo riconoscitore, applicabili a problemi di tipo decisionale.

Definizione 25 *Dato una alfabeto* $\Sigma \subseteq \Gamma$*, una stringa* $x \in \Sigma^*$ *è* accettata *dalla macchina* $\mathcal{M}$ *se esiste una computazione accettante* $c_0, \ldots, c_n$ *di* $\mathcal{M}$ *, con* $c_0 = q_0 x$*.*

Definizione 26 *Dato una alfabeto* $\Sigma \subseteq \Gamma$*, una stringa* $x \in \Sigma^*$ *è* rifiutata *dalla macchina* $\mathcal{M}$ *se tutte le computazioni di* $\mathcal{M}$ *sono rifiutanti.*

Una $MTND$ rifiuta il suo input se perviene a configurazioni non finali sulle quali non si puó applicare la δ_N.

Osservazione 1: le condizioni di accettazione e di rifiuto sono asimmetriche. Nel primo caso è sufficiente raggiungere *una* configurazione finale, nel secondo caso, è necessario che *tutte* le computazioni portino a configurazioni che non siano finali.

Osservazione 2: guardando alle MTND come trasduttori, cioè come dispositivi per il calcolo di funzioni, si ha che partendo dalla stessa configurazione iniziale si potrebbe pervenire a risultati differenti operando diverse scelte nell'albero delle computazioni (*calcolo di funzioni multivalore*).

Le macchine di Turing non deterministiche hanno un potere computazionale maggiore di quelle deterministiche?

Equivalenza fra MDT e $MTND$

Teorema 27 *Per ogni macchina di Turing non deterministica $\mathcal{M}$ esiste una macchina di Turing deterministica $\mathcal{M}_D$ a 3 nastri equivalente.*

Il numero di passi richiesto da $\mathcal{M}_D$ per simulare k passi di $\mathcal{M}$ $O(kd^k)$.

Quindi la simulazione comporta un aumento esponenziale del tempo di accettazione di una stringa rispetto al tempo richiesto dalla macchina non deterministica.

Problema aperto. Esiste una simulazione più efficiente? È possibile simulare una macchina di Turing non deterministica con una deterministica in tempo polinomiale?

Ci sono valide ragioni per ritenere che non sia possibile simulare una macchina di Turing detemrinistica con un numero di passi sub-esponenziale. Ciò, comunque, non è mai stato dimostrato.

NOTA BENE: una macchina di Turing deterministica è ingrado di simulare efficientemente una MTND se è in possesso di una informazione aggiuntiva (la stringa di caratteri in $\{1,\ldots,d\}$) che la "guida" verso una configurazione di accettazione, se esiste.

Se L è il linguaggio accettato da una MDNT $\mathcal{M}$ allora, per ogni stringa $x \in L$, una sequenza di scelte $c \in \{1,\ldots,d\}^*$ che conduca $\mathcal{M}$ con input x ad una configurazione di accettazione può essere vista come una *prova* dell'appartenenza di x ad L.

Quindi, mentre la *determinazione* di una prova c per $x \in L$ può richiedere un tempo esponenziale con una macchina di Turing deterministica ceh simula $\mathcal{M}$, la *verifica* che c sia una prova di $x \in L$ può essere effettuata con un numero di passi sensibilmente inferiore.

Riduzione di MDT

Le macchine di Turing possono essere semplificate ulteriormente senza che si perda in potere computazionale.
Una prima semplificazione si ottiene limitando le caratteristiche del nastro di una macchina di Turing: non più infinito in entrambe le direzioni bensì finito in una direzione ed infinito nell'altra. In questo caso si parla di macchina di Turing con nastro *semi-infinito*.

Teorema 28 *Per ogni macchina di Turing* $\mathcal{M} = \langle \Gamma, \not{b}, Q, q_0, F, \delta \rangle$ *esiste una macchina di Turing a* $\mathcal{M}'$ *equivalente, con nastro semi-infinito.*

Dimostrazione. $\mathcal{M}'$ utilizza un nastro a 3 tracce. Il nastro infinito viene piegato in due sulla prima e terza traccia. Sulla seconda traccia un simbolo $\uparrow$ o $\downarrow$ indica se la testina della macchina si trova sulla parte semi-infinita destra o sulla parte semi-infinita sinistra.

Una seconda versione di macchina di Turing semplificata si ottiene riducendo la dimensione dell'alfabeto. In particolare, è possibile ridurre la cardinalità dell'alfabeto Γ a 1.

Teorema 29 *Per ogni MT* $\mathcal{M} = \langle \Gamma, \not{b}, Q, q_0, F, \delta \rangle$*esiste una MT* $\mathcal{M}' = \langle \Gamma', \not{b}, Q', q_0', F', \delta' \rangle$ *con* $|\Gamma'| = 1$ *equivalente*

Dimostrazione. I $|\Gamma| + 1$ caratteri di $\overline{\Gamma}$ sono codificati con i 2 caratteri di $\overline{\Gamma'}$. Inoltre, per ogni regola di transizione della macchina $\mathcal{M}$ prevediamo una serie di regole di transizione della macchina $\mathcal{M}'$ che svolgono le seguenti azioni:

- determinazione del carattere originario osservato;
- eventuale modifica di tale carattere;
- eventuale spostamento della testina in corrispondenza della codifica del carattere più a destra o più a sinistra di quello osservato correntemente.

codificazione di una MDT

È possibile descrivere una macchina di Turing con una stringa di caratteri e fornire tale descrizione come input ad un'altra macchina di Turing.
Esistono vari modi di descrivere una macchina di Turing con una stringa:

- possiamo fornire la sequenza delle quintuple che costituiscono la funzione di transizione $\sharp\sharp d_1\sharp\sharp d_2\sharp\sharp \ldots \sharp\sharp d_n$ in cui è presente la quintupla $q_i\sharp\sigma_j\sharp q_h\sharp\sigma_k\sharp t_1$ se e solo se esiste la regola di transizione $\delta(q_i, \sigma_j) = (q_h, \sigma_k, t_1)$;
- possiamo sfruttare il fatto che ogni macchina di Turing può essere realizzata come composizione di alcune macchine elementari e fornire la descrizione di una macchina come sequenza di tali macchine (*macchina di Turing linearizzata*).

La Macchina di Turing universale

Qual è il potere computazionale della macchina di Turing?
Esistono funzioni non calcolabili secondo Turing?
Si cercherà di caratterizzare la potenza delle macchine di Turing individuando prima una funzione estremamente potente che è calcolabile secondo Turing e successivamente una funzione non calcolabile secondo Turing.

Definizione 30 *Sia* $m : (\Sigma^*)^n \longmapsto \Sigma^*$ *una funzione a piú argomenti. Una macchina di Turing* $\mathcal{M}$ *calcola* m *se realizza la computazione* $q_0 x_1 \not b \ldots \not b x_n \vdash_{\mathcal{M}} x_1 \not b \ldots \not b x_n \not b q y$, *con* q *stato finale sse* $m(x_1 \ldots x_n) = y$.

Definizione 31 *Una macchina di Turing* $\mathcal{U} = \langle \Gamma, \not b, Q', q'_0, F', \delta' \rangle$ *si dice* macchina universale *se calcola una funzione* $u : (\Gamma^*)^{n+1} \longmapsto \Gamma^*$ *con la seguente proprietá:*
data una qualunque macchina di Turing $\mathcal{M} = \langle \Gamma, \not b, Q, q_0, F, \delta \rangle$ *che calcola la funzione* $m : (\Gamma^*)^n \longmapsto \Gamma^*$, *esiste una stringa* $c_{\mathcal{M}} \in \Gamma^*$ *(una codifica di* $\mathcal{M}$*) tale che* $u(c_{\mathcal{M}}, x_1, \ldots, x_n) = m(x_1, \ldots, x_n)$.

La macchina universale è quindi in grado di simulare il comportamento di ogni altra macchina di Turing. Esiste una tale macchina?

La dimostrazione è costruttiva. Si procede in questo modo:

- si prende in considerazione una MT con nastro semi-infinito e alfabeto costituito dagli unici due simboli 1 e $\not b$. Non c'è perdita di generalità.

- Si costruisce la descrizione di ogni MT e la si rappresenta mediante una stringa dell'alfabeto $\{1, \not b\}$.

- Si definisce una macchina che, assumendo in input tale descrizione, simuli il comportamento della macchina descritta.

Quindi vale il seguente

Teorema 32 *Esiste una macchina di Turing* $\mathcal{U} = \langle \Gamma, \not b, Q', q'_0, F', \delta' \rangle$ *tale che data ogni macchina di Turing* $\mathcal{M} = \langle \{1\}, \not b, Q, \delta, q_0, F \rangle$*:* $u(c_{\mathcal{M}}, x) = m(x)$ *per ogni* $x \in \{1, \not b\}^*$.

Possiamo interpretare la MT universale come

- un calcolatore e $c_{\mathcal{M}}$ e x come un programma e i suoi dati;

- un interprete di un linguaggio di programmazione.

Nel secondo caso possiamo modificare $\mathcal{U}$ in modo che non cancelli il programma (codifica $c_{\mathcal{M}}$).

Il problema della terminazione (halting problem): data una macchina di Turing $\mathcal{M}$ ed una stringa x, stabilire se $\mathcal{M}$ termina la computazione avendo x come input.

Teorema 33 *Dati un alfabeto* Γ *ed una codifica che associa ad ogni macchina* $\mathcal{M} = \langle \Gamma, \not{b}, Q, q_0, F, \delta \rangle$ *una stringa* $c_{\mathcal{M}} \in \Gamma^*$*. La funzione*

$$h(c_{\mathcal{M}}, x) = \begin{cases} 1 & \text{se } \mathcal{M} \text{ termina su input } x, \\ 0 & \text{se } \mathcal{M} \text{ non termina su input } x. \end{cases}$$

non è T-calcolabile.