

Automati e Linguaggi Formali

Parte 12 – Varianti di macchine di Turing

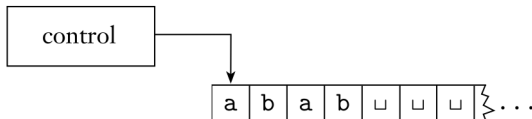
Davide Bresolin
Ultimo aggiornamento: 11 maggio 2023



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

1 Varianti di macchine di Turing

- Esistono **definizioni alternative** delle macchine di Turing
- Chiamiamo **varianti** queste alternative
- Tutte le varianti “ragionevoli” riconoscono **la stessa classe di linguaggi**
- Le Turing machine sono un modello **robusto**

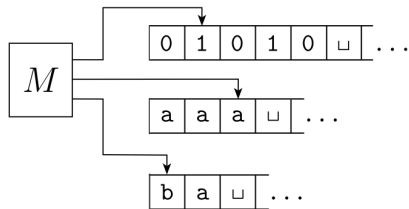


- è una TM con un nastro **infinito solo verso destra**
- l'input si trova **all'inizio del nastro**
- la testina parte dalla **posizione più a sinistra del nastro**
- se M tenta di spostare la testina a sinistra quando si trova nella prima cella del nastro, allora **la testina rimane ferma**



Theorem

- 1** *Per ogni TM a nastro semi-infinito esiste una TM a nastro infinito equivalente.*
- 2** *Per ogni TM a nastro infinito esiste una TM a nastro semi-infinito equivalente.*



- è una TM con k nastri semi-infiniti
- k testine di lettura e scrittura
- l'input si trova sul nastro 1
- ad ogni passo scrive e si muove **simultaneamente** su tutti i nastri

- funzione di transizione:

$$\delta : Q \times \Gamma^k \mapsto Q \times \Gamma^k \times \{L, R\}^k$$

$$\delta(q_i, a_1, \dots, a_k) = (q_j, b_1, \dots, b_k, L, R, \dots, L) :$$

- se lo stato è q_i e le testine leggono a_1, \dots, a_k
- allora scrivi b_1, \dots, b_k sui k nastri
- muovi ogni testina a sinistra o a destra come specificato

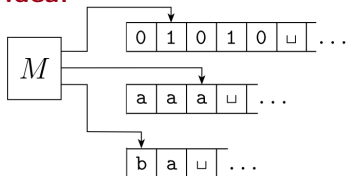
Theorem

Per ogni TM multinastro esiste una TM a singolo nastro equivalente.

Theorem

Per ogni TM multinastro esiste una TM a singolo nastro equivalente.

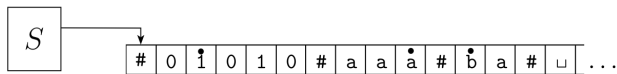
Idea:



■ S simula M

■ i nastri sono separati da #

■ un punto sopra un simbolo indica la posizione della testina





$S =$ "Su input $w = w_1 \dots w_n$:"

$S =$ "Su input $w = w_1 \dots w_n$:

- 1 Inizializza il nastro per rappresentare i k nastri:

$$\# \overset{\bullet}{w}_1 w_2 \dots w_n \# \sqcup \# \sqcup \# \dots \#$$

$S =$ "Su input $w = w_1 \dots w_n$:

- 1 Inizializza il nastro per rappresentare i k nastri:

$$\# \overset{\bullet}{w}_1 w_2 \dots w_n \# \sqcup \overset{\bullet}{\#} \sqcup \overset{\bullet}{\#} \dots \#$$

- 2 Per simulare una mossa di M , scorri il nastro per determinare i simboli puntati dalle testine virtuali

$S =$ "Su input $w = w_1 \dots w_n$:

- 1 Inizializza il nastro per rappresentare i k nastri:

$$\# \overset{\bullet}{w}_1 w_2 \dots w_n \# \overset{\bullet}{\sqcup} \# \overset{\bullet}{\sqcup} \# \dots \#$$

- 2 Per simulare una mossa di M , scorri il nastro per determinare i simboli puntati dalle testine virtuali
- 3 Fai un secondo passaggio del nastro per aggiornare i nastri virtuali secondo la funzione di transizione di M .

S = "Su input $w = w_1 \dots w_n$:

- 1 Inizializza il nastro per rappresentare i k nastri:

$$\# \overset{\bullet}{w}_1 w_2 \dots w_n \# \sqcup \overset{\bullet}{\#} \sqcup \overset{\bullet}{\#} \dots \#$$

- 2 Per simulare una mossa di M , scorri il nastro per determinare i simboli puntati dalle testine virtuali
- 3 Fai un secondo passaggio del nastro per aggiornare i nastri virtuali secondo la funzione di transizione di M .
- 4 Se S sposta una testina virtuale a destra su un $\#$, allora M ha spostato la testina sulla parte vuota del nastro. Scrivi un \sqcup e sposta il contenuto del nastro di una cella a destra

S = “Su input $w = w_1 \dots w_n$:

- 1 Inizializza il nastro per rappresentare i k nastri:

$$\# \overset{\bullet}{w}_1 \overset{\bullet}{w}_2 \dots \overset{\bullet}{w}_n \# \square \# \square \# \dots \#$$

- 2 Per simulare una mossa di M , scorri il nastro per determinare i simboli puntati dalle testine virtuali
- 3 Fai un secondo passaggio del nastro per aggiornare i nastri virtuali secondo la funzione di transizione di M .
- 4 Se S sposta una testina virtuale a destra su un $\#$, allora M ha spostato la testina sulla parte vuota del nastro. Scrivi un \square e sposta il contenuto del nastro di una cella a destra
- 5 Se si raggiunge una configurazione di accettazione, **accetta**, se si raggiunge una configurazione di rifiuto, **rifiuta**, altrimenti ripeti da **2**.

Corollary

*Un linguaggio è Turing-riconoscibile se e solo se esiste una macchina di Turing **multinastro** che lo riconosce.*

Corollary

*Un linguaggio è Turing-riconoscibile se e solo se esiste una macchina di Turing **multinastro** che lo riconosce.*

- ⇒ Un linguaggio è Turing-riconoscibile se è riconosciuto da una TM con un solo nastro, che è un caso particolare di TM multinastro

Corollary

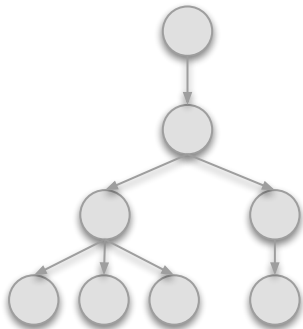
*Un linguaggio è Turing-riconoscibile se e solo se esiste una macchina di Turing **multinastro** che lo riconosce.*

- ⇒ Un linguaggio è Turing-riconoscibile se è riconosciuto da una TM con un solo nastro, che è un caso particolare di TM multinastro
- ⇐ Costruzione precedente

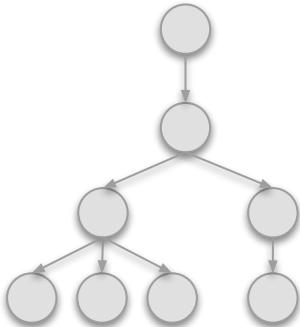
- Una TM non deterministica ha **più strade possibili** durante la computazione
- Consideriamo macchine con un solo nastro semi-infinito
- La funzione di transizione è:

$$\delta : Q \times \Gamma \mapsto 2^{(Q \times \Gamma \times \{L, R\})}$$

- la computazione è un **albero** che descrive le scelte possibili
- la macchina accetta se **esiste un ramo** che porta allo stato di accettazione



- La computazione è un **albero** i cui rami corrispondono alle diverse **possibilità** per la macchina
- **Tutti i rami** devono essere esaminati fino a quando non viene trovato uno **stato di accettazione**



- La computazione è un **albero** i cui rami corrispondono alle diverse **possibilità** per la macchina
- **Tutti i rami** devono essere esaminati fino a quando non viene trovato uno **stato di accettazione**
- Come esaminare l'albero: **in ampiezza** o **in profondità**?



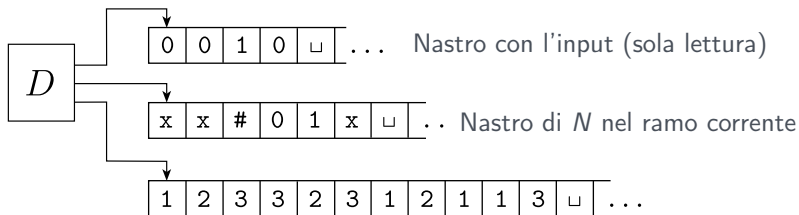
Theorem

Per ogni TM non deterministica N esiste una TM deterministica D equivalente.

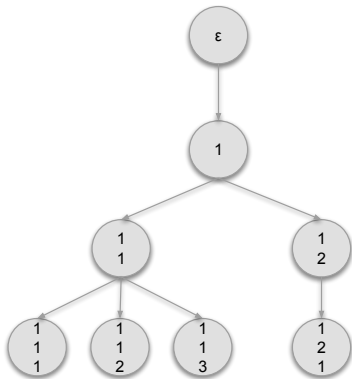
Theorem

Per ogni TM non deterministica N esiste una TM deterministica D equivalente.

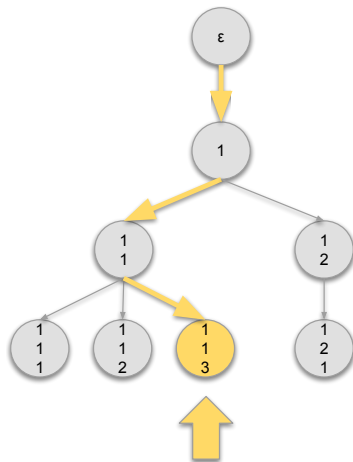
Idea:



Il terzo nastro tiene traccia delle scelte non deterministiche



- Ad ogni nodo viene assegnato un **indirizzo**: una stringa sull'alfabeto $\Gamma_b = \{1, 2, \dots, b\}$, dove b è il massimo numero di figli dei nodi dell'albero



- Ad ogni nodo viene assegnato un **indirizzo**: una stringa sull'alfabeto $\Gamma_b = \{1, 2, \dots, b\}$, dove b è il massimo numero di figli dei nodi dell'albero
- Il nodo 113 si raggiunge prendendo il **primo** figlio della radice, seguito dal **primo** figlio di quel nodo ed infine dal **terzo** figlio.
- Questo ordinamento può essere utilizzato per attraversare in modo efficiente l'albero in ampiezza.

- 1** Inizialmente il nastro 1 contiene l'input w e i nastri 2 e 3 sono vuoti

- 1** Inizialmente il nastro 1 contiene l'input w e i nastri 2 e 3 sono vuoti
- 2** Copia il nastro 1 sul nastro 2 e inizializza la stringa sul nastro 3 a ε

- 1** Inizialmente il nastro 1 contiene l'input w e i nastri 2 e 3 sono vuoti
- 2** Copia il nastro 1 sul nastro 2 e inizializza la stringa sul nastro 3 a ε
- 3** Usa il nastro 2 per simulare N con input w su un ramo di computazione.
Prima di ogni passo di N , consulta il simbolo successivo sul nastro 3 per determinare quale scelta fare (tra quelle consentite).

- 3** (cont.) Se non rimangono più simboli sul nastro 3, o se questa scelta non è valida, interrompi questo ramo e vai alla fase **4**. Vai alla fase **4** anche se si incontra una configurazione di rifiuto.
- Se viene trovata una configurazione di accettazione, **accetta**.

- 3** (cont.) Se non rimangono più simboli sul nastro 3, o se questa scelta non è valida, interrompi questo ramo e vai alla fase **4**. Vai alla fase **4** anche se si incontra una configurazione di rifiuto.
Se viene trovata una configurazione di accettazione, **accetta**.
- 4** Sostituire la stringa sul nastro 3 con la stringa successiva nell'ordine delle stringhe. Simula il ramo successivo di N andando alla fase **2**.



Corollary

*Un linguaggio è Turing-riconoscibile se e solo se esiste una macchina di Turing **non deterministica** che lo riconosce.*



Corollary

*Un linguaggio è Turing-riconoscibile se e solo se esiste una macchina di Turing **non deterministica** che lo riconosce.*

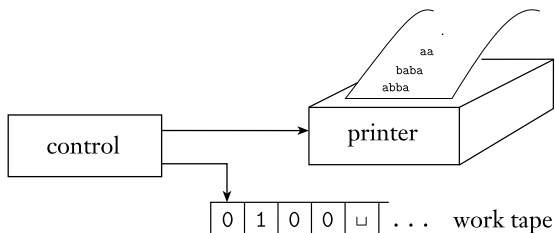
- ⇒ Un linguaggio è Turing-riconoscibile se è riconosciuto da una TM deterministica, che è un caso particolare di TM non deterministica



Corollary

*Un linguaggio è Turing-riconoscibile se e solo se esiste una macchina di Turing **non deterministica** che lo riconosce.*

- ⇒ Un linguaggio è Turing-riconoscibile se è riconosciuto da una TM deterministica, che è un caso particolare di TM non deterministica
- ⇐ Costruzione precedente



- **Enumeratore:** macchina di Turing + stampante
- Un enumeratore E inizia con **nastro vuoto**
- Di tanto in tanto, **invia una stringa alla stampante**
- Linguaggio **enumerato** da E : tutte le stringhe stampate
- E può generare le stringhe in qualsiasi ordine, anche con ripetizioni

Theorem

Un linguaggio è Turing-riconoscibile se e solo se esiste un enumeratore che lo enumera

Theorem

Un linguaggio è Turing-riconoscibile se e solo se esiste un enumeratore che lo enumera

Idea: dobbiamo mostrare che

- se esiste un enumeratore E , allora esiste una TM M che riconosce lo stesso linguaggio
- se esiste una TM M che riconosce il linguaggio, allora possiamo costruire un enumeratore

- Una macchina di Turing con “resta ferma” invece di “muovi a sinistra”
- Funzione di transizione: $\delta : Q \times \Gamma \mapsto Q \times \Gamma\{S, R\}$
- Ad ogni passo, la TM può lasciare ferma la testina o muoverla a destra
- Non può muoversi a sinistra!

Domanda

Quale classe di linguaggi riconosce?

- Esistono altri modelli di computazione universali
- Alcuni sono molto simili alle macchine di Turing
- Altri sono molto diversi
- Hanno tutti una caratteristica comune:
 - **accesso senza restrizioni ad una memoria illimitata**
- Sono **tutti equivalenti tra loro!**