Macchine di Turing multinastro

21 aprile 2022

Varianti della MdT

Esistono diverse varianti della definizione di macchina di Turing deterministica.

Vedremo:

- la Macchina di Turing multinastro
- la Macchina di Turing non deterministica

Tali macchine di Turing hanno lo stesso potere computazionale (o espressivo) delle MdT deterministiche, cioè riconoscono la stessa classe di linguaggi.

Esistono anche altre varianti della macchina di Turing deterministica che hanno lo stesso potere espressivo.

Chiamiamo "robustezza" questa invarianza ad alcune variazioni nella definizione. Essa è una conferma che si tratta di un buon modello per la definizione di algoritmo.

Equivalenza di modelli

Siano $\mathcal{T}_1$ e $\mathcal{T}_2$ due famiglie di modelli computazionali. Per dimostrare che i modelli in $\mathcal{T}_1$ hanno lo stesso potere computazionale dei modelli in $\mathcal{T}_2$ occorre far vedere che per ogni macchina $M_1 \in \mathcal{T}_1$ esiste $M_2 \in \mathcal{T}_2$ equivalente ad M_1 e viceversa.

Abbiamo già dimostrato che la classe dei DFA ha lo stesso potere computazionale della classe degli NFA.

Come per le classi dei DFA e degli NFA, consiste spesso nel dimostrare che per ogni macchina di una classe ne esiste una dell'altra classe capace di simularla.

Come nel caso dei DFA e degli NFA, in genere una delle direzioni della prova è evidente.

Equivalenza di modelli

Per illustrare la robustezza del modello di macchina di Turing cerchiamo di variare il tipo di funzione di transizione consentito.

Nella nostra definizione, la funzione di transizione forza la testina a spostarsi verso sinistra o destra ad ogni passo; la testina non può restare ferma.

Supponiamo di aver permesso alla macchina di Turing la capacità di restare ferma.

Stayer: MdT in cui la testina può rimanere sulla stessa cella del nastro durante una transizione
Funzione di transizione: $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, S, R\}$
dove S serve ad indicare che la testina rimane ferma

Equivalenza di modelli

Questa caratteristica **non** permette alle macchine di Turing di riconoscere ulteriori linguaggi, cioè non aggiunge potere computazionale al modello scelto.

Per poter fare questa affermazione occorre trovare per ogni macchina di un tipo una equivalente dell'altro tipo.

La parte "non ovvia" è mostrare che possiamo trasformare qualsiasi macchina di Turing che ha la possibilità di "restar ferma" in una macchina di Turing equivalente che non ha tale capacità.

Lo facciamo costruendo una MdT in cui sostituiamo ogni transizione "resta ferma" con due transizioni, una che sposta la testina a destra e una che la riporta a sinistra.

Altra variante di MdT

E una MdT con «S = resta ferma» al posto di «L = muovi a sinistra»?

Questa variante NON è equivalente.

L’accesso alla memoria/nastro è restrittivo

Altre varianti equivalenti di MdT

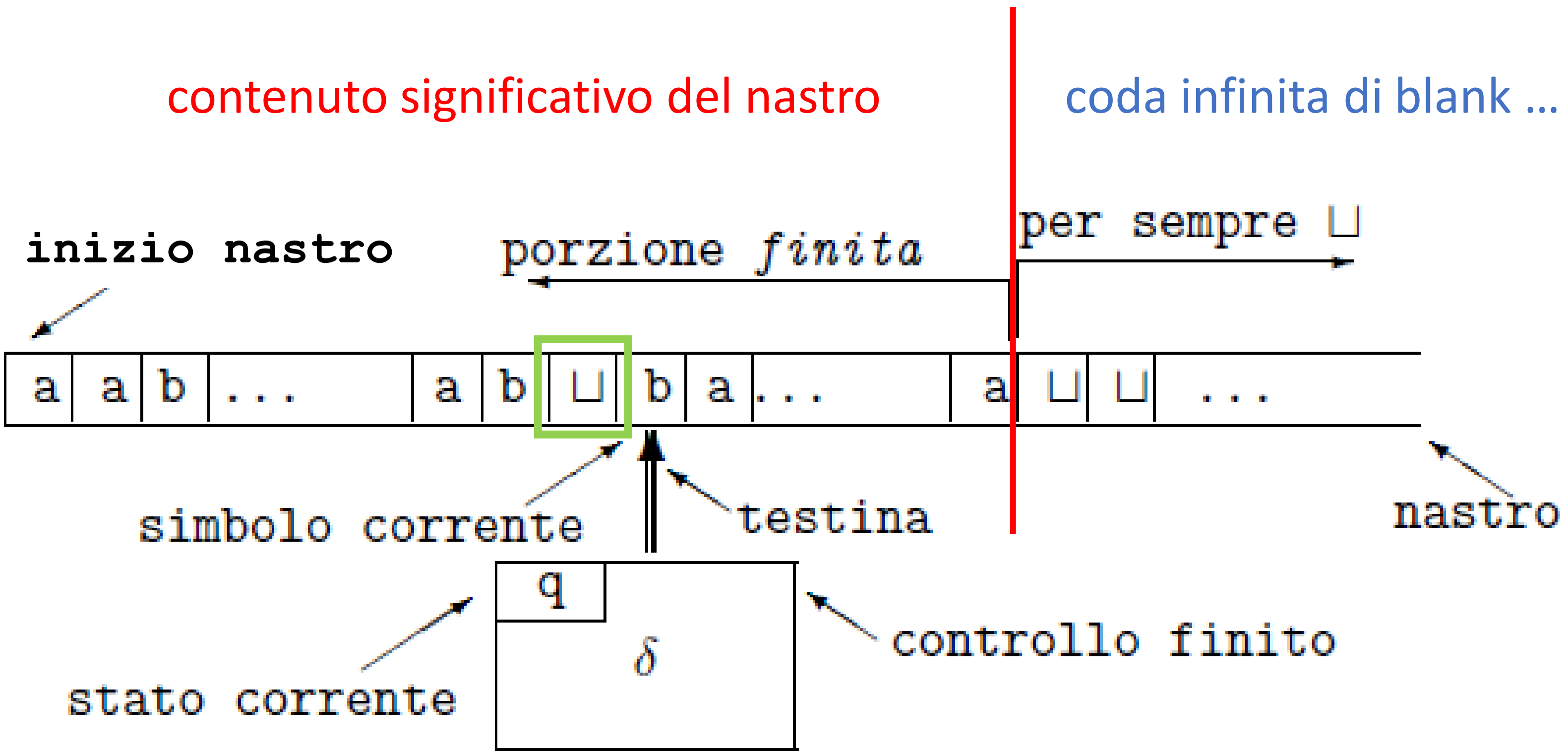
- MdT a sola scrittura (Ex. 3.17)
- MdT a nastro doppiamente infinito (ex. 3.18)
- MdT con Reset a sinistra (Ex. 3.19)

Sono tutte equivalenti alla MdT.

Sono stati proposti molti altri modelli di computazione (+ o – simili alle MdT). Sorprendentemente:

Tutti i modelli «ragionevoli» con un accesso non restrittivo ad una memoria illimitata risultano essere equivalenti.

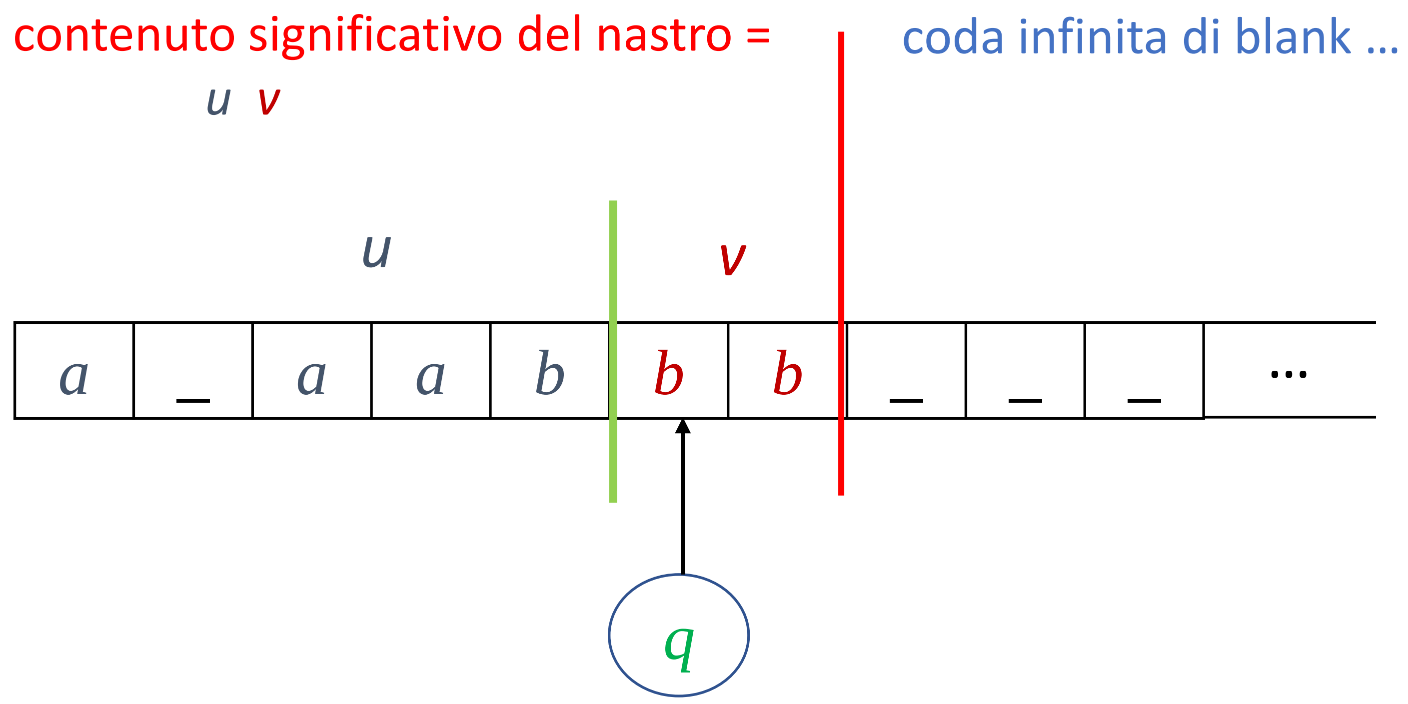
Una MdT



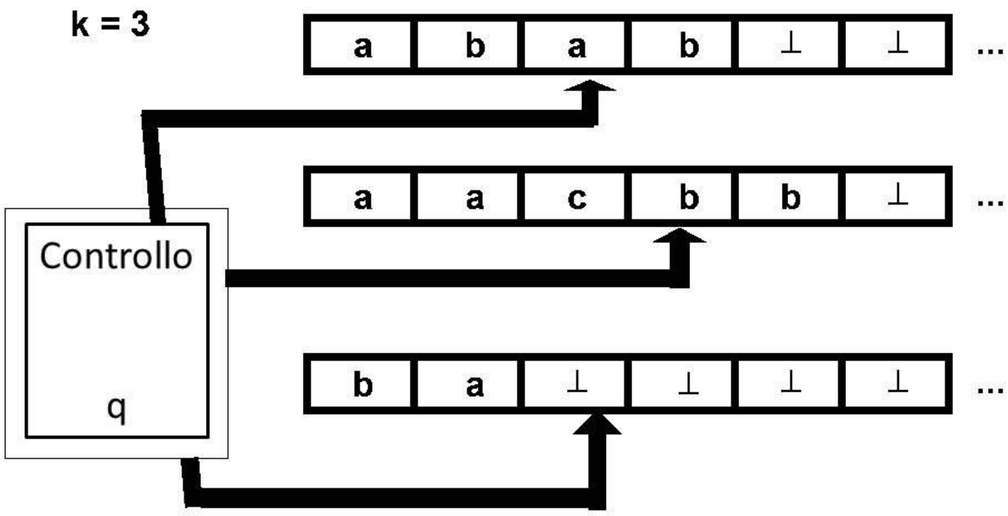
Il contenuto significativo del nastro è una stringa $w \in \Gamma^*$, con la convenzione che il suo ultimo carattere (se $w \neq \varepsilon$) non sia blank.
La stringa w PUO' contenere blank al suo interno.
A volte nel progetto di una MdT si definiscono delle transizioni per scrivere un carattere speciale nella prima cella del nastro, per meglio individuarla.

Configurazione di una MdT

La configurazione $C = u\ q\ v$ corrisponde a



Una macchina di Turing multinastro (abbreviata MdTM) è una macchina di Turing in cui si hanno più nastri contemporaneamente accessibili in scrittura e lettura che vengono aggiornati tramite più testine (una per nastro).



Macchina di Turing multinastro

Definizione (MdT a k nastri)

Dato un numero intero positivo k, **una macchina di Turing con k nastri** *è una settupla*

$$(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$$

dove $Q, \Sigma, \Gamma, q_0, q_{accept}, q_{reject}$ sono definiti come in una MdT deterministica e la funzione di transizione δ è definita al modo seguente:

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

Nota: Γ^k è il prodotto cartesiano di k copie di Γ e $\{L, R, S\}^k$ è il prodotto cartesiano di k copie di $\{L, R, S\}$.

Definizione (MdT multinastro)

Una **macchina di Turing multinastro** *è una macchina di Turing a k nastri, dove k è un numero intero positivo.*

Funzione di transizione di una MdTM

Espressione

$$\delta(q_i, a_1, \ldots, a_k) = (q_j, b_1, \ldots, b_k, D_1, \ldots, D_k)$$

indica che se la MdT a k nastri M

- si trova nello stato q_i
- la testina t legge a_t per $t = 1 \ldots, k$

allora

- M va nello stato q_j
- la testina t scrive b_t e si muove nella direzione $\mathsf{D}_t \in \{L, S, R\}$, per $t = 1 \ldots, k$

Le nozioni di configurazione, passo di computazione, di linguaggio deciso e di linguaggio riconosciuto da una MdT sono estese in maniera ovvia alle macchine MdTM.

Ad esempio, se $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ è una macchina di Turing con k nastri, una configurazione ha la forma

$$(u_1 q v_1, \ldots, u_k q v_k)$$

dove

- q è **lo** stato corrente di M
- $u_i\ q\ v_i$ è la configurazione del nastro i-esimo, i = 1, … , k

Computazione di una MdTM

Se $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ è una macchina di Turing con k nastri,

$(q_0 w, q_0, \ldots, q_0)$ è una configurazione iniziale con input w

Ovvero all'inizio M

- Si trova nello stato q_0
- Il primo nastro contiene w, posizionato a partire dalla prima cella (e seguito da tutti blank)
- Tutti gli altri nastri sono vuoti (blank)
- Tutte le k testine puntano alla prima cella del nastro corrispondente

Computazione di una MdTM

Se $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ è una macchina di Turing con k nastri,

$(q_0 w, q_0, \ldots, q_0)$ è una configurazione iniziale con input w

$(u_1 q_{accept} v_1, \ldots, u_k q_{accept} v_k)$ è una configurazione di accettazione.

Il linguaggio $L(M)$ riconosciuto da M è

$$\begin{aligned} L(M) &= \{w \in \Sigma^* \mid \exists u_t, v_t \in \Gamma^*, t \in \{1, \ldots, k\} : \\ &\quad (q_0 w, \ldots, q_0) \to^* (u_1 q_{accept} v_1, \ldots, u_k q_{accept} v_k)\} \end{aligned}$$

Esempio: MdT a 2 nastri per $\{0^n 1^n \mid n > 0\}$

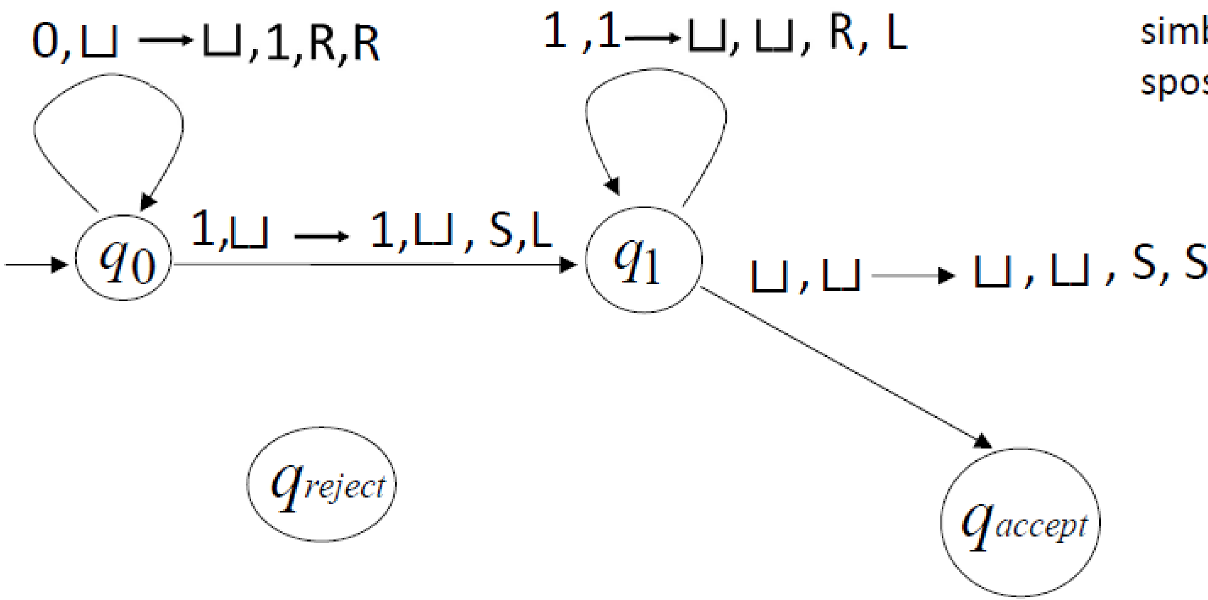
1. Scorre il primo nastro verso destra fino al primo 1: per ogni 0, scrive un 1 sul secondo nastro
2. Scorre il primo nastro verso destra e il secondo nastro verso sinistra: se i simboli letti non sono uguali, termina in q_{reject}
3. Se legge $\sqcup$ su entrambi i nastri, termina in q_{accept}

stato attuale	**simbolo letto**	**Valore funzione δ**
q_0	$(0, \sqcup)$	$\mathrm{q}_0, (\sqcup, 1), (R, R)$
q_0	$(1, \sqcup)$	$\mathrm{q}_1, (1, \sqcup), (S, L)$
q_1	$(1, 1)$	$\mathrm{q}_1, (\sqcup, \sqcup), (R, L)$
q_1	$(\sqcup, \sqcup)$	$\mathrm{q}_{\mathrm{accept}}, (\sqcup, \sqcup), (S, S)$

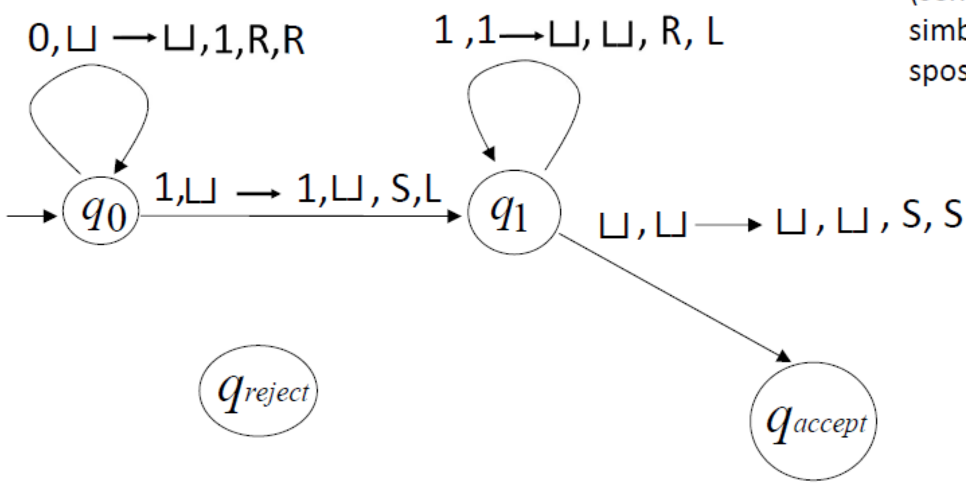
Nota: Se $\delta(q, \gamma, \gamma')$ non è presente nella tabella, con $q \in Q \setminus \{q_{accept}, q_{reject}\}$ allora $\delta(q, \gamma, \gamma') = (q_{reject}, \gamma, \gamma', S, S)$.

Esempio: MdT a 2 nastri per $\{0^n 1^n \mid n > 0\}$

Transizioni omesse vanno in q_{reject} (senza cambiare i simboli letti o spostare la testina)



Esempio: MdT a 2 nastri per $\{0^n1^n \mid n > 0\}$



Transizioni omesse vanno in q_{reject} (senza cambiare i simboli letti o spostare la testina)

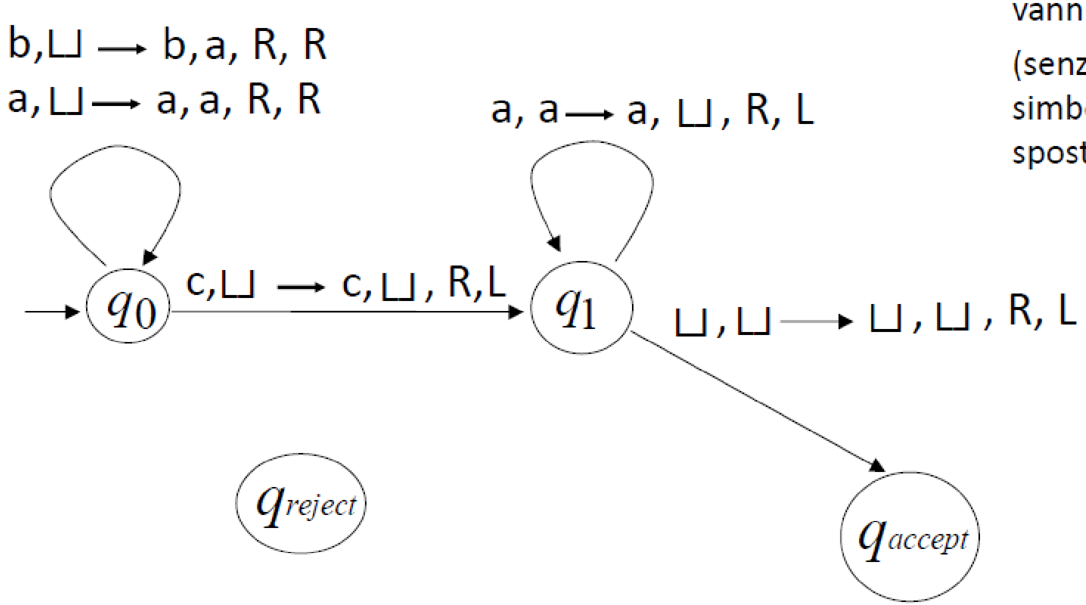
Esempio computazione su w = 0 0 1 1

Esempio: MdT a 2 nastri per $\{wca^{|w|} \mid w \in \{a, b\}^*\}$

Una macchina di Turing deterministica M a due nastri che decide il linguaggio

$$\{wca^{|w|} \mid w \in \{a, b\}^*\}$$

sull'alfabeto $\Sigma = \{a, b, c\}$.



Transizioni omesse vanno in q_{reject} (senza cambiare i simboli letti o spostare la testina)

Notare analogie con la precedente

Il modello di MdT «potenziato» con la possibilità di avere più di un nastro, permette di riconoscere altri linguaggi?

Abbiamo dei vantaggi?

Livelli di descrizione di MdT

La MdT è nata come modello preciso di «algoritmo», «procedura effettiva di calcolo».

Tre livelli di descrizione:

1. Descrizione ad alto livello
2. Descrizione implementativa (come muove la testina, memorizza i dati, …)
3. Descrizione formale, precisa (come settupla)

TM – example revisited

TM M recognizing $B = \{\mathrm{a}^k\mathrm{b}^k\mathrm{c}^k \mid k \geq 0\}$

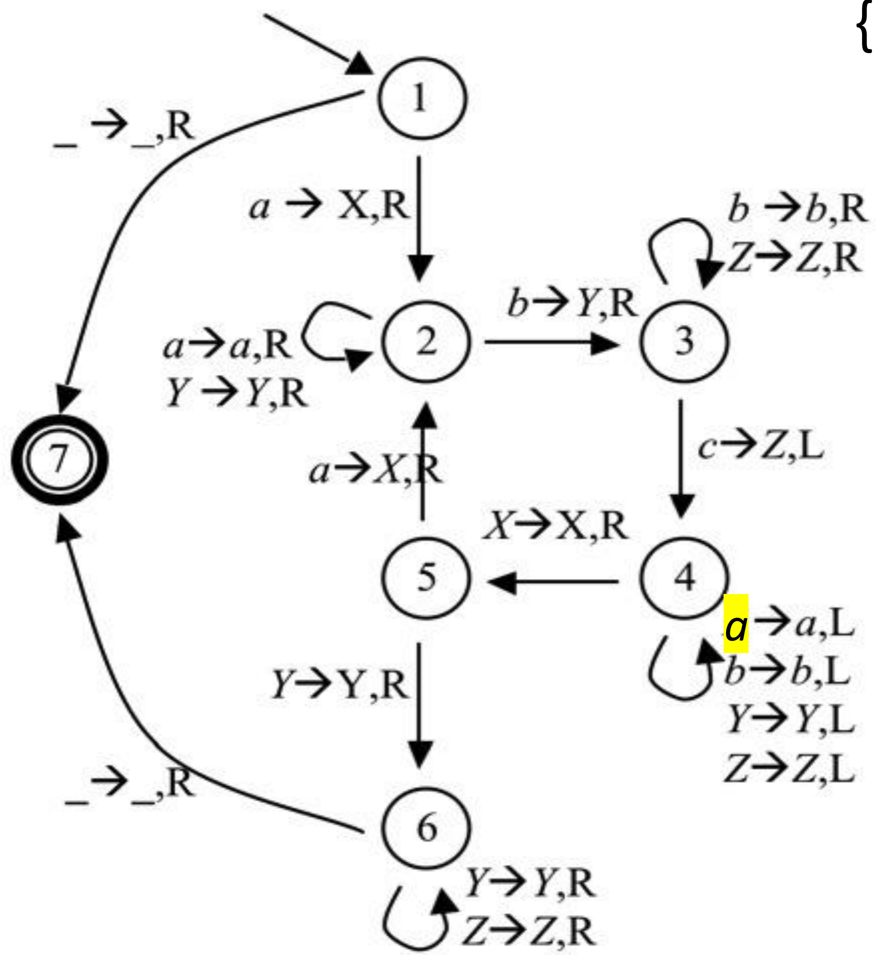
$M =$ "On input w

1. Check if $w \in \mathrm{a}^*\mathrm{b}^*\mathrm{c}^*$, *reject* if not.
2. Count the number of a's, b's, and c's in w.
3. *Accept* if all counts are equal; *reject* if not."

High-level description is ok.

You do not need to manage tapes, states, etc…

$\{ a^n b^n c^n \}$



1
2
3
4
5
6
7
_ → _,R
a → X,R
a→a,R
Y → Y,R
b→Y,R
b →b,R
Z→Z,R
c→Z,L
a→X,R
X→X,R
a→a,L
b→b,L
Y→Y,L
Z→Z,L
Y→Y,R
_→_,R
Y→Y,R
Z→Z,R

Esempio

Consideriamo il linguaggio

$$L = \{0^{2^n} \mid n > 0\}$$

insieme stringhe di 0 la cui lunghezza è potenza di 2

Nota. Il linguaggio non è regolare

Vogliamo costruire una MdT M_2 che lo decide.

Esempio

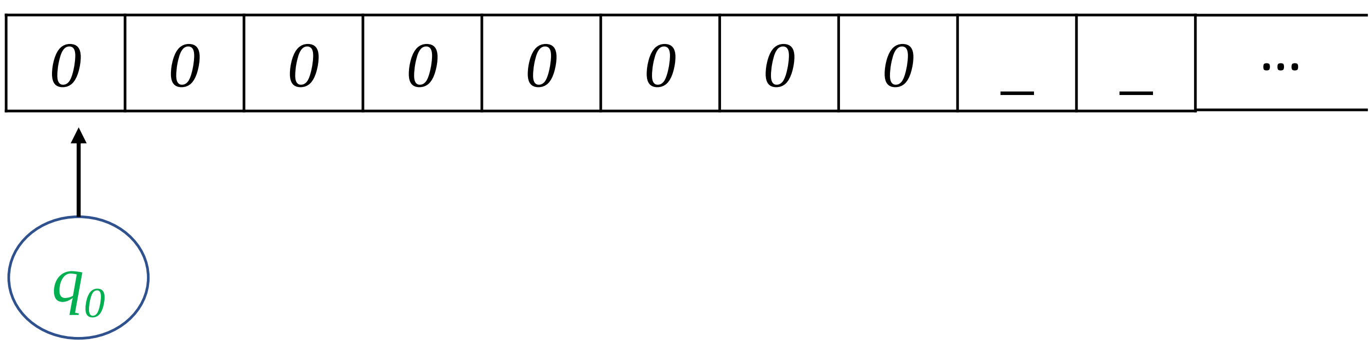
Come riconoscere se il numero di 0 è una potenza di 2?

1, 2, 4, 8, 16, 32, ….

Innanzitutto deve essere pari; se dispari rifiuto.

Se pari?

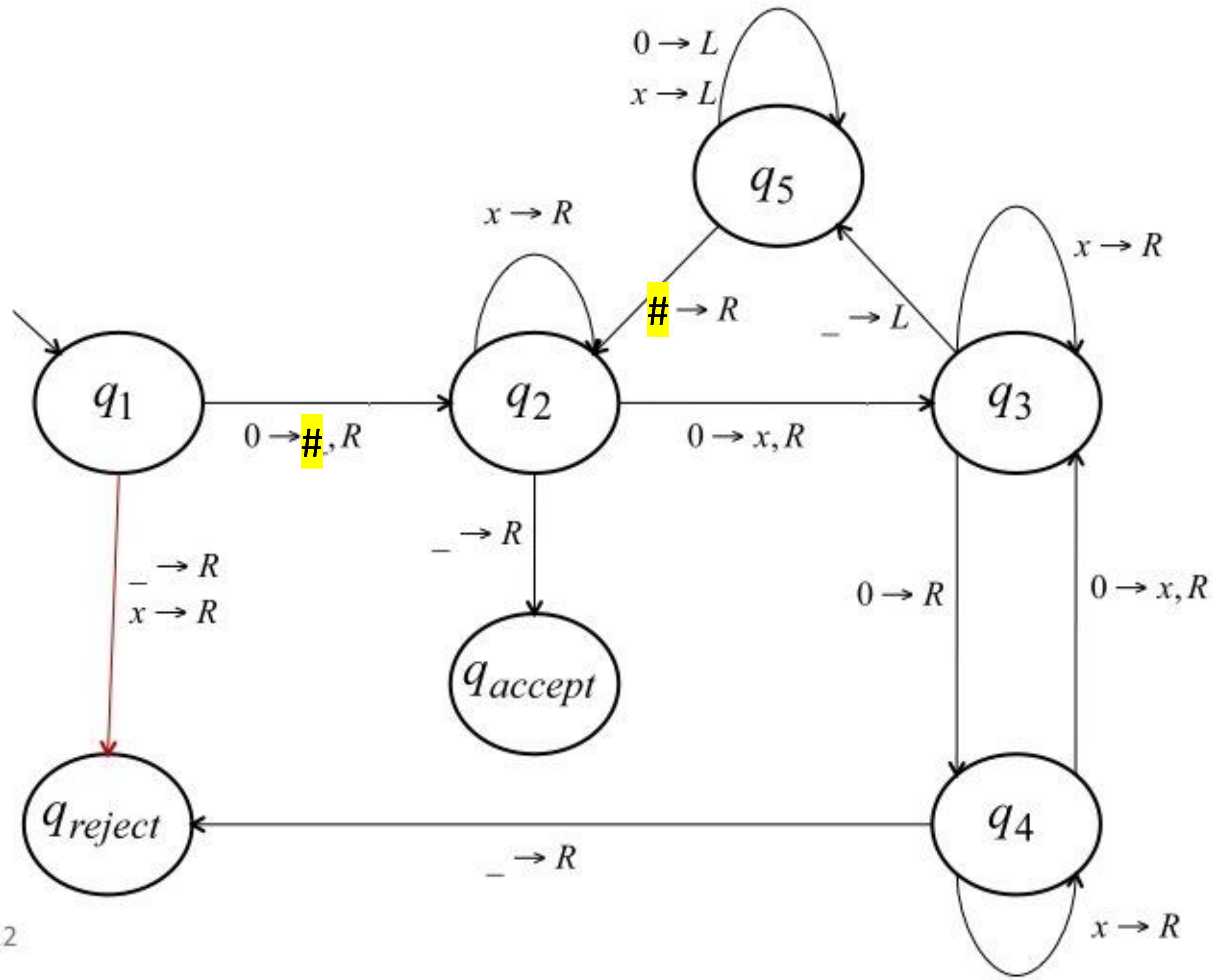
Le potenze di 2 hanno la caratteristica che dividendo ripetutamente per 2 trovo sempre numeri pari, fino ad arrivare ad 1.



MdT per 0^{2^n}

Algoritmo 1

Segno con x le posizioni pari degli 0 presenti. Se ultima è pari, ripeto sugli 0 rimasti, finché rimane un solo 0, il primo marcato con #



Esercizio

Progettare una MdT che decida il linguaggio

$$L = \{ 0^{2^n} \mid n \geq 0\}$$

e che sia **concettualmente diversa** dal precedente algoritmo.

Algoritmo 2

Cancello gli 0 della «seconda metà». Se pari, ripeto finché rimane un solo 0, il primo marcato con Y.

Discussa a lezione, ma diagramma non del tutto funzionante.

Esercizio

Progettare una MdT che decida il linguaggio

$$L = \{ 0^{2^n} \mid n \geq 0\}$$

e che sia **concettualmente diversa** dagli algoritmi precedenti.

Esistono almeno 3 algoritmi:

1. Divisioni successive per 2 fino ad arrivare ad 1 solo 0: segno gli 0 in pos. pari
2. Divisioni successive per 2 fino ad arrivare ad 1 solo 0: cancello gli 0 della seconda metà
3. Moltiplicazioni successive per 2: marco con X il primo 0; raddoppio gli 0 marcati X, copiando X^k dopo X^k ottenendo così X^{2k}.
   Esempio: 0000, X000, XX00, XXXX accetto.

E con una MdT a 2 (o più) nastri, avremmo «vantaggi»?

3.8 Give implementation-level descriptions of Turing machines that decide the following languages over the alphabet $\{0,1\}$.

[A]**a.** $\{w \mid w$ contains an equal number of 0s and 1s$\}$

b. $\{w \mid w$ contains twice as many 0s as 1s$\}$

c. $\{w \mid w$ does not contain twice as many 0s as 1s$\}$

3.8 (a) "On input string w:

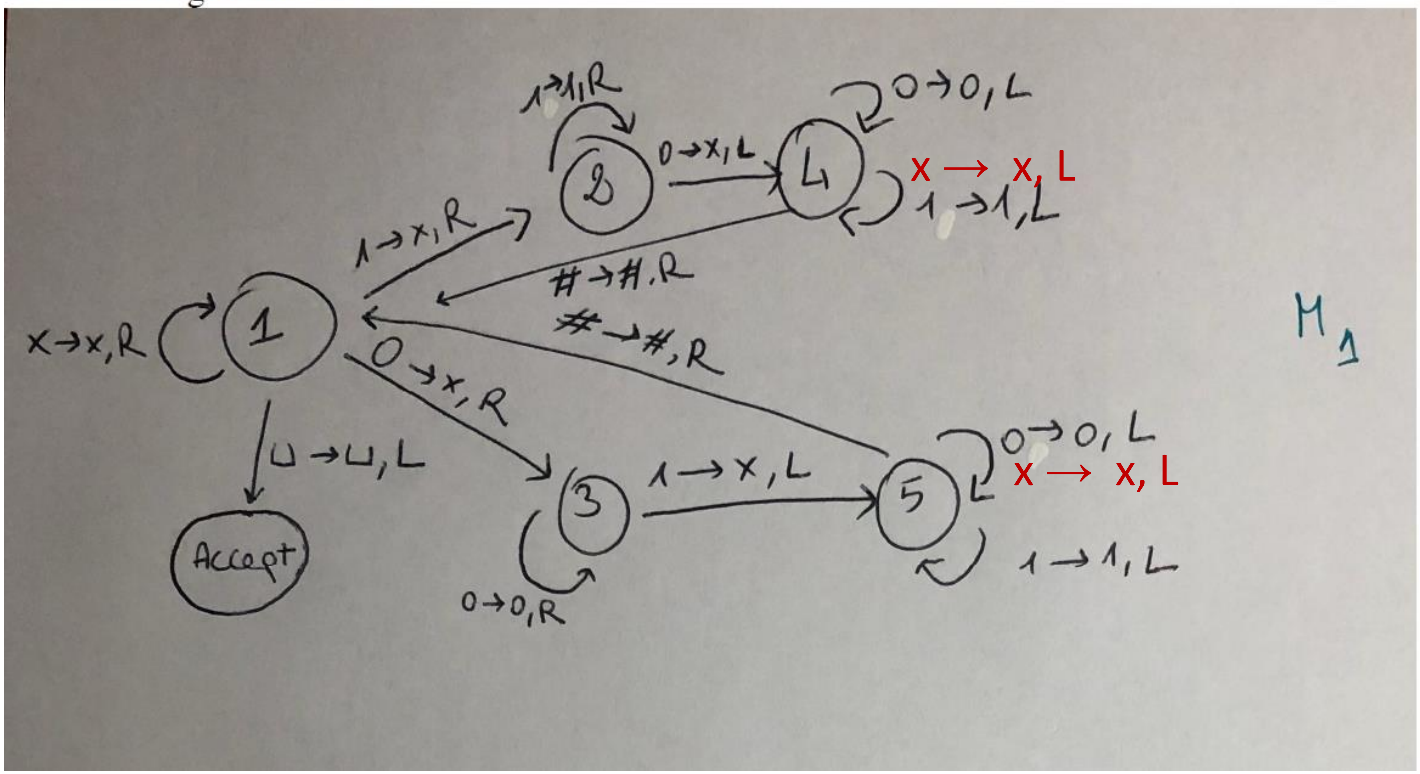
1. Scan the tape and mark the first 0 that has not been marked. If no unmarked 0 is found, go to stage 4. Otherwise, move the head back to the front of the tape.
2. Scan the tape and mark the first 1 that has not been marked. If no unmarked 1 is found, *reject*.
3. Move the head back to the front of the tape and go to stage 1.
4. Move the head back to the front of the tape. Scan the tape to see if any unmarked 1s remain. If none are found, *accept*; otherwise, *reject*."

Sipser, es. 3.8 (a)

Descrizione verbale:

Segno il primo simbolo 0 o 1 che incontro. Scorro a destra cercando la prima istanza di simbolo diverso da quello segnato. Se lo trovo, lo segno e ripeto. Altrimenti (ovvero incontro un blank) rifiuto.

Possibile diagramma di stato:



Presuppone che il simbolo nella prima cella sia marcato con #.

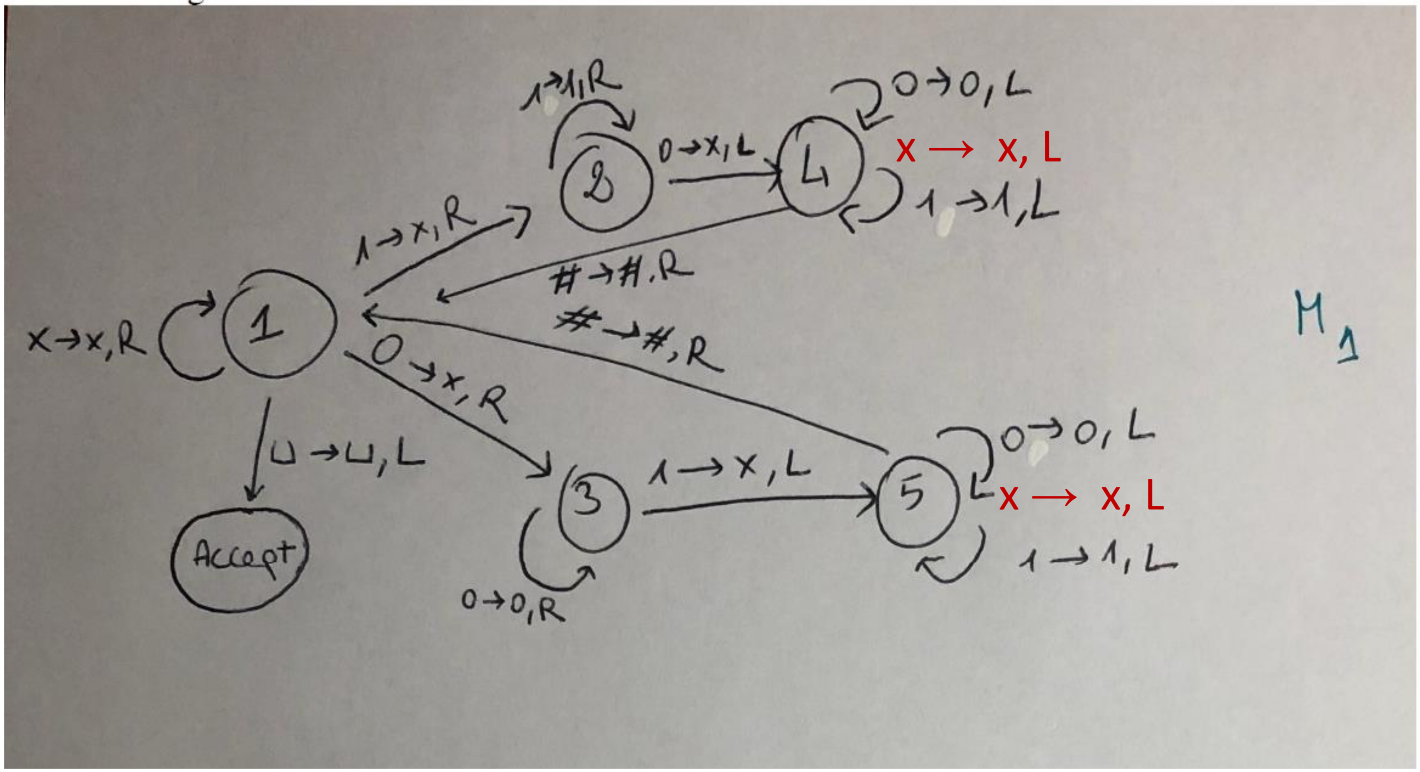
Come inserire un meccanismo che consenta ciò?

Sipser, es. 3.8 (a)

Descrizione verbale:

Segno il primo simbolo 0 o 1 che incontro. Scorro a destra cercando la prima istanza di simbolo diverso da quello segnato. Se lo trovo, lo segno e ripeto. Altrimenti (ovvero incontro un blank) rifiuto.

Possibile diagramma di stato:



Nota: il ciclo 1, 2, 4, 1 è una «replica» del ciclo 1, 3, 5, 1, ma riferito ad un primo simbolo 1, anziché 0.

Implementazione della memoria

Nota: questa è una tecnica utilizzabile in generale

Se vogliamo che M memorizzi una sequenza di k simboli possiamo

- avere una replica per ogni possibile sequenza di k simboli
- M si sposta sulla replica che corrisponde alla sequenza mentre la legge

3.8 Give implementation-level descriptions of Turing machines that decide the following languages over the alphabet $\{0,1\}$.

[A]**a.** $\{w \mid w \text{ contains an equal number of 0s and 1s}\}$

b. $\boxed{\{w \mid w \text{ contains twice as many 0s as 1s}\}}$

c. $\{w \mid w \text{ does not contain twice as many 0s as 1s}\}$

Descrizione verbale:

1. Leggo il primo carattere non segnato. Se è blank, accetto. Altrimenti vado al punto 2 e 3.
2. Se c'è un 1: cerco due 0. Rifiuto se non li trovo. Torno al punto 1.
3. Se c'è uno 0: cerco un altro zero e un altro uno. Rifiuto se non li trovo. Torno al punto 1.

3.8 Give implementation-level descriptions of Turing machines that decide the following languages over the alphabet $\{0,1\}$.

[A] **a.** $\{w \mid w$ contains an equal number of 0s and 1s$\}$

b. $\{w \mid w$ contains twice as many 0s as 1s$\}$

c. $\{w \mid w$ does not contain twice as many 0s as 1s$\}$

Uguale al secondo punto, ma la macchina rifiuta al posto di accettare e accetta invece di rifiutare.

3.8 Give implementation-level descriptions of Turing machines that decide the following languages over the alphabet $\{0,1\}$.

[A] **a.** $\{w \mid w$ contains an equal number of 0s and 1s$\}$

b. $\{w \mid w$ contains twice as many 0s as 1s$\}$

c. $\{w \mid w$ does not contain twice as many 0s as 1s$\}$

E con una MdT a 2 (o più) nastri, avremmo «vantaggi»?

1. Progettare una macchina di Turing che sposta l'input a destra di una casella.

2. Progettare una macchina di Turing che calcola il successore in binario.

Questi esercizi ci mostrano cosa può fare una MdT: copiare, spostare, calcolare successore, somma, differenza (in unario o binario), …
contare, ovvero riconoscere occorrenze di ugual numero, lunghezza pari ad una potenza di 2, …..