Macchine di Turing: prime varianti

20 aprile 2023

Varianti della MdT

Esistono diverse varianti della definizione di macchina di Turing deterministica.

Vedremo:
- lo Stayer
- 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: $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, S, R\}$
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.

Equivalenza di modelli

Formalizziamo questo discorso.

Chiamiamo $\mathcal{T}_{(L,R)}$ l'insieme delle macchine di Turing che abbiamo definito, cioè quelle con funzione di transizione

$$\delta : (Q \setminus \{q_{accept}, q_{reject}\}) \times \Gamma \to Q \times \Gamma \times \{L, R\}$$

$\mathcal{T}_{(L,R,S)}$ è l'insieme delle macchine M tali che
$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ è una settupla in cui
$Q, \Sigma, \Gamma, q_0, q_{accept}, q_{reject}$ sono definiti come in una MdT deterministica e la funzione di transizione δ è definita al modo seguente:

$$\delta : (Q \setminus \{q_{accept}, q_{reject}\}) \times \Gamma \to Q \times \Gamma \times \{L, R, S\}$$

Equivalenza di modelli

Se $\delta(q, \gamma) = (q', \gamma', d)$ e se M si trova nello stato q con la testina posizionata su una cella contenente γ, alla fine della transizione:

- M si troverà nello stato q',
- $\gamma' \in \Gamma$ sarà il simbolo scritto sulla cella del nastro su cui la testina si trovava all'inizio della transizione (contenente γ),
- la testina si troverà sulla stessa cella cui si trovava all'inizio della computazione se $d = S$, si sarà spostata sulla cella di sinistra se (tale cella esiste e se) $d = L$, si sarà spostata sulla cella di destra se $d = R$.

Le nozioni di configurazione, di linguaggio deciso e di linguaggio riconosciuto da $M' \in \mathcal{T}_{(L,R)}$ sono estese in maniera ovvia alle macchine M in $\mathcal{T}_{(L,R,S)}$.

Equivalenza di MdT e Stayer

I modelli in $\mathcal{T}_{(L,R)}$ hanno lo stesso potere computazionale dei modelli in $\mathcal{T}_{(L,R,S)}$.

Prova

Ogni MdT classica, in particolare è uno stayer.

Viceversa se $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject}) \in \mathcal{T}_{(L,R,S)}$ definiamo $M' = (Q \cup Q^c, \Sigma, \Gamma, \delta', q_0, q_{accept}, q_{reject}) \in \mathcal{T}_{(L,R)}$ dove:

- se $\delta(q_i, \gamma) = (q_j, \gamma', d)$ e $d \in \{L, R\}$ allora $\delta'(q_i, \gamma) = \delta(q_i, \gamma)$
- se $\delta(q_i, \gamma) = (q_j, \gamma', S)$ allora $\delta'(q_i, \gamma) = (q_j^c, \gamma', R)$ e $\delta'(q_j^c, \eta) = (q_j, \eta, L)$ per ogni $\eta \in \Gamma$.
- $Q^c = \{q^c \mid (q, \gamma, S) \in \delta((Q \setminus \{q_{accept}, q_{reject}\}) \times \Gamma)\}$.

Anche in questo caso $L(M) = L(M')$.

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.

Equivalenza di modelli di computazione

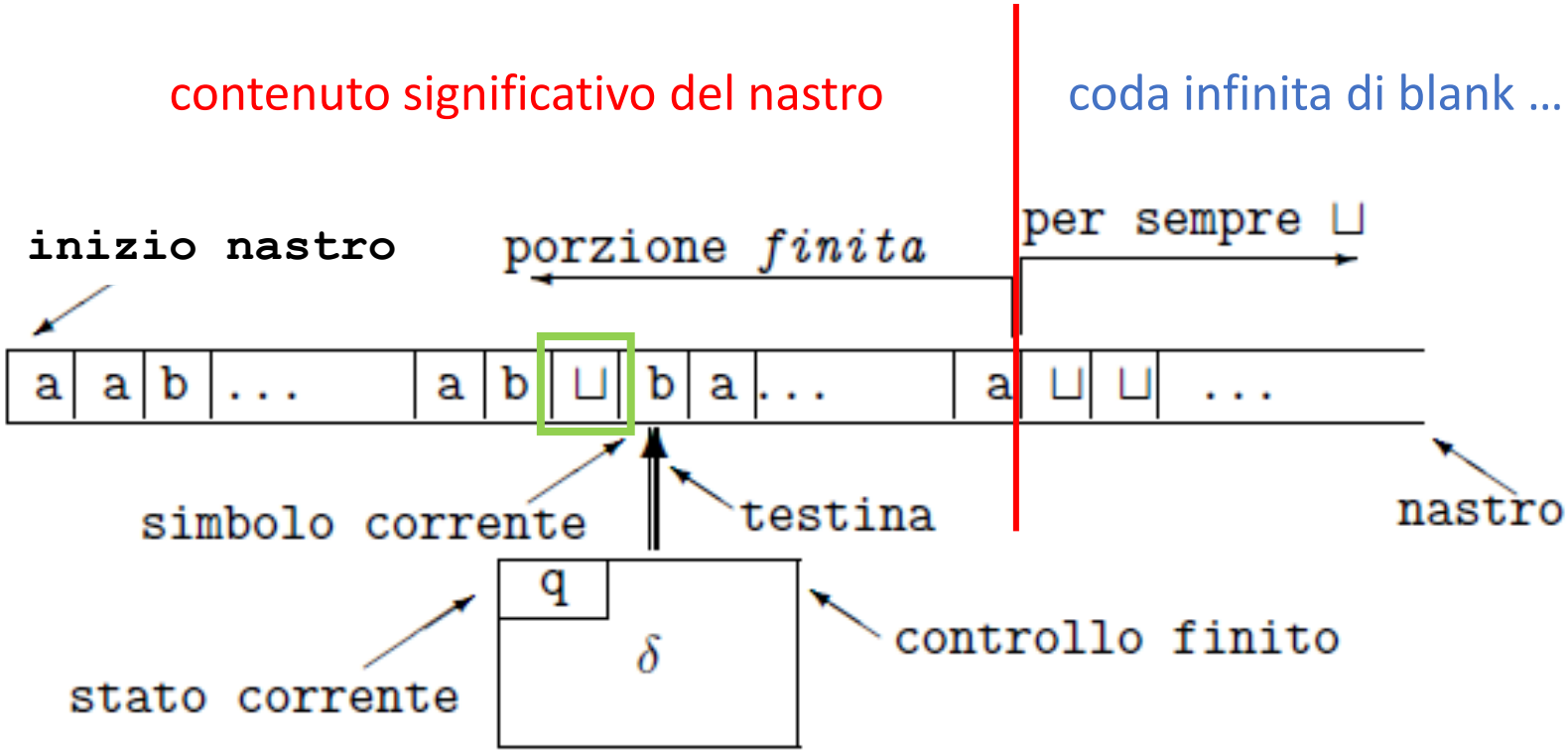
Tutti i modelli «ragionevoli» con un accesso non restrittivo ad una memoria illimitata risultano essere equivalenti.

I modelli di computazione immaginabili sono tanti, ma la classe di «algoritmi» che essi descrivono rimane la stessa!

Analogia con i linguaggi di programmazione: non esiste un «algoritmo» che può essere programmato in C e non in Java, o viceversa.

I linguaggi di programmazione immaginabili sono tanti, ma la classe di «algoritmi» che essi descrivono rimane la stessa!

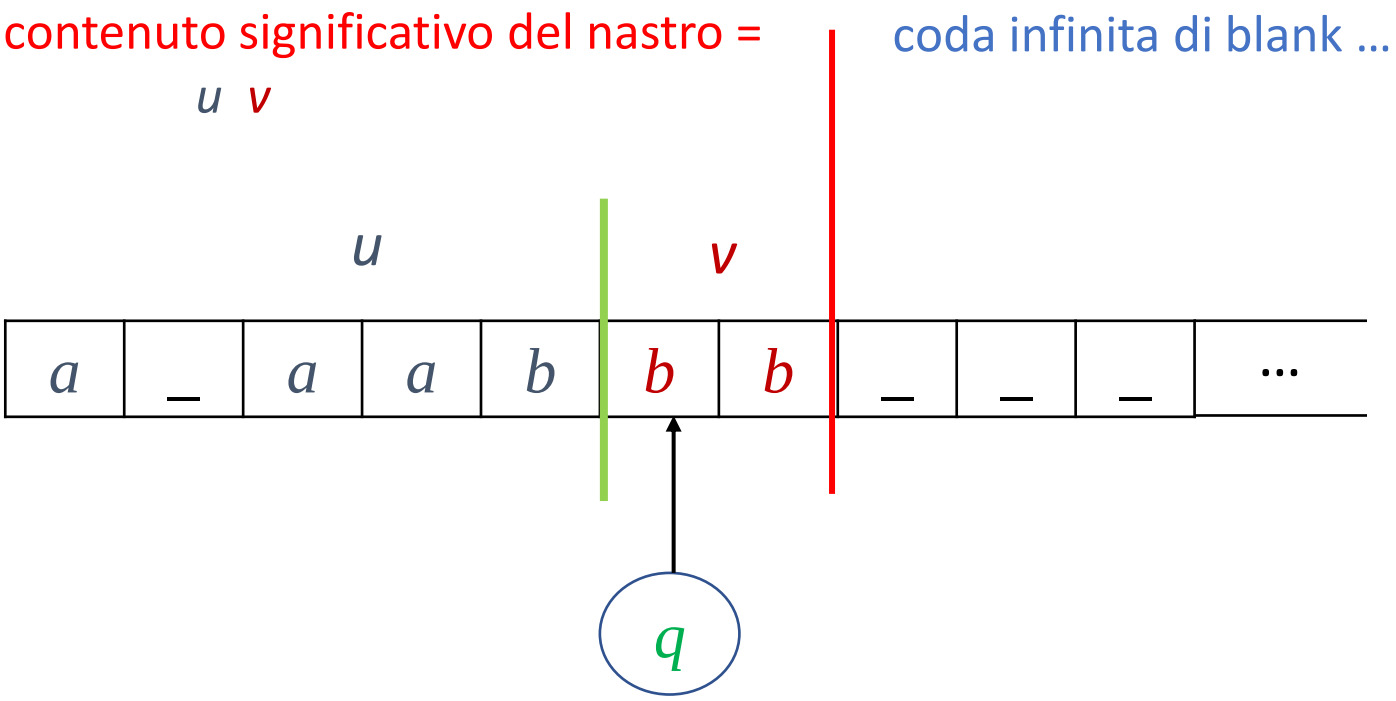
Una MdT



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.

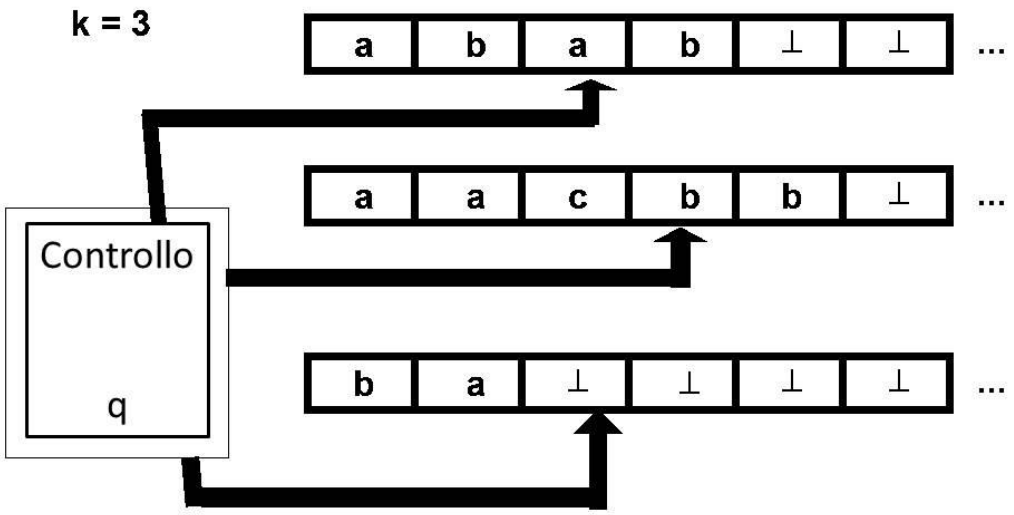
Configurazione di una MdT

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



Macchina di Turing multinastro

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:

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

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 $t = 1 \ldots, k$

Computazione di una MdTM

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_0w, q_0, … , 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, ... , 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

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

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

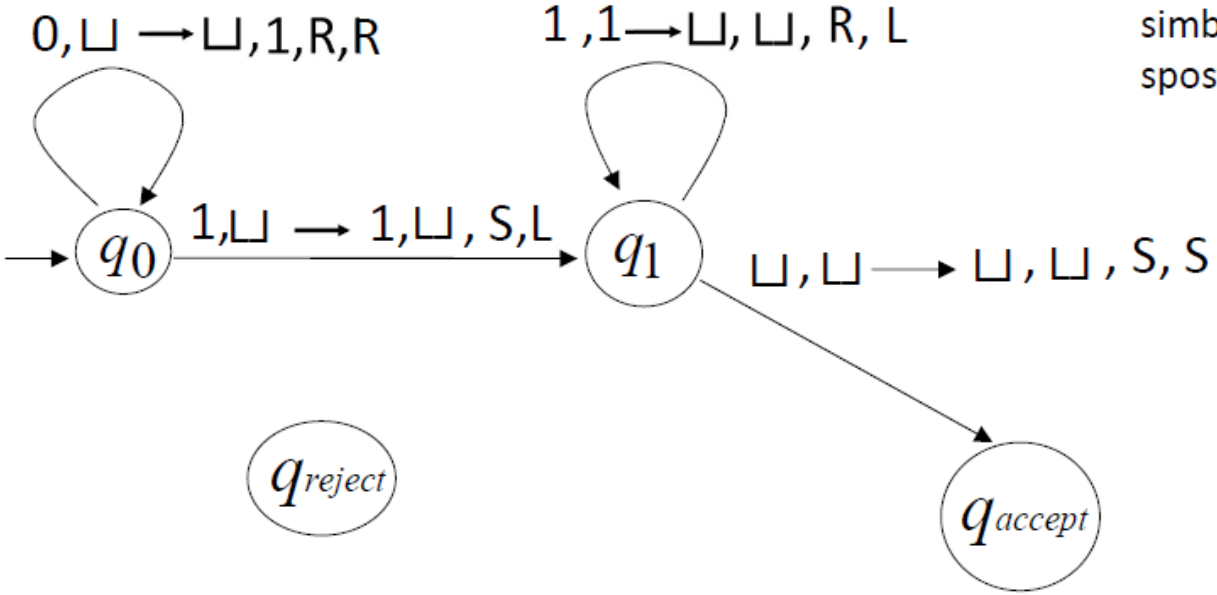
1. Scorre il primo nastro verso destra fino al primo 1: per ogni 0, scrive un 1 sul secondo nastro e cancella lo 0
2. Scorre il primo nastro verso destra e il secondo nastro verso sinistra: se i simboli letti non sono uguali, termina in q_{reject}
3. Se legge $\sqcup$ su entrambi i nastri, termina in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

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

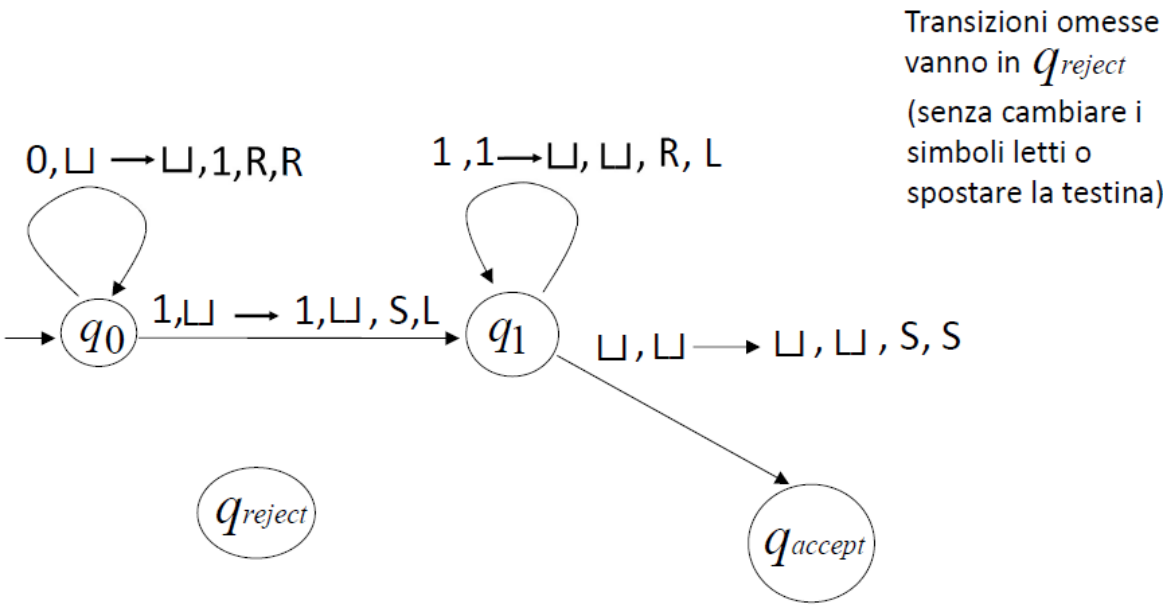
Nota: Se $\delta(q, \gamma, \gamma')$ non è presente nella tabella, con $q \in Q \setminus \{q_{accept}, q_{reject}\}$ allora $\delta(q, \gamma, \gamma') = (q_{reject}, \gamma, \gamma', S, S)$.

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

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



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



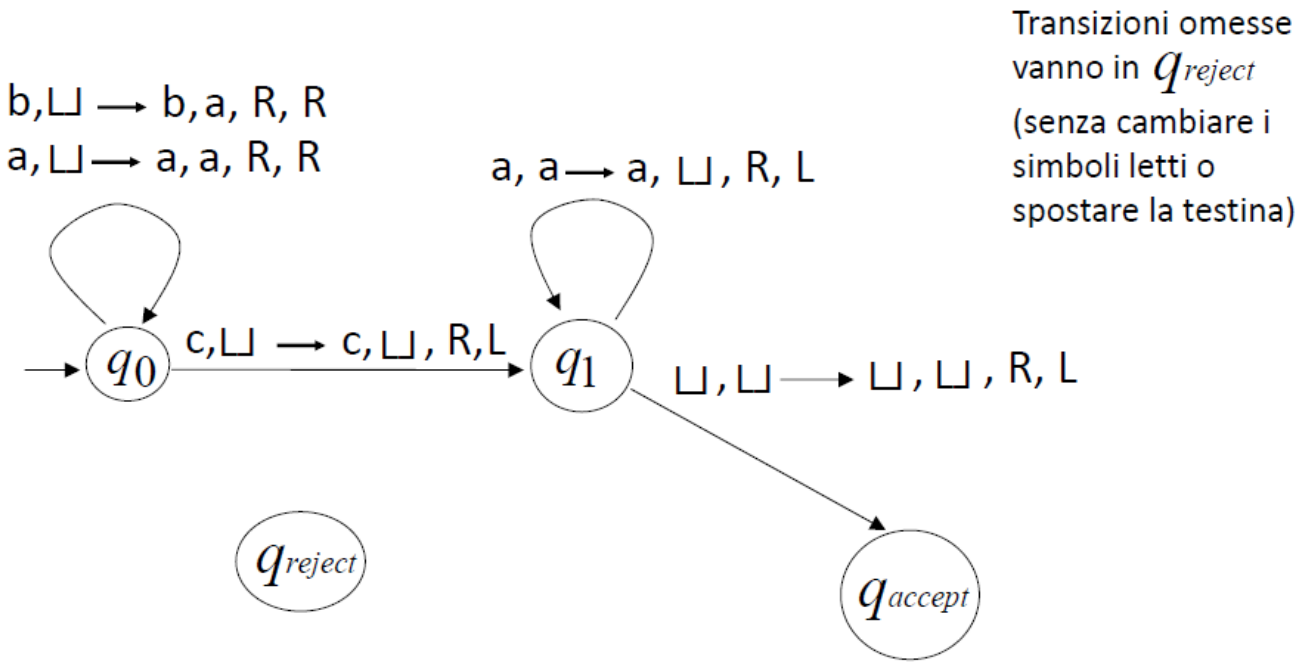
Esempio computazione su w = 0 0 1 1

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 $\Sigma = \{a, b, c\}$.



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?

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»?

Esercizio

Progettare una MdT che calcoli la funzione f(x,y), differenza intera di due interi positivi x e y

$$f(x, y) = x - y \qquad \text{se } x \geq y$$
$$f(x, y) = 0 \qquad \text{altrimenti.}$$

Si supponga che l'input sia $< x > 0 < y >$, dove $< n > = 1^n$, è la rappresentazione unaria dell'intero positivo n.

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.

3. Esercizi_ETC_lez2_definizioni.pdf

Un altro esempio

La funzione $f(x) = 2x$ e' calcolabile

x e' un intero

Macchina di Turing:

Stringa input: x in notazione unaria

Stringa output: xx in notazione unaria