VARIANTI DI MACCHINE DI TURING

Varianti di 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.

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.

Ricordiamo: Due macchine sono equivalenti se riconoscono lo stesso linguaggio.

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

A differenza però degli automi finiti, alle macchine in $\mathcal{T}_1$ potrebbe corrispondere una rappresentazione delle configurazioni diversa dalla rappresentazione delle configurazioni delle macchine in $\mathcal{T}_2$.

La dimostrazione che per ogni macchina $M_1 \in \mathcal{T}_1$ esiste $M_2 \in \mathcal{T}_2$ equivalente ad M_1 (e viceversa) consiste spesso nella dimostrazione che per ogni M_1 esiste M_2 capace di **simularla** (e viceversa): sono in corrispondenza configurazioni iniziali, configurazioni intermedie ed eventuali configurazioni di arresto.

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

Un semplice esempio

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.

La funzione di transizione avrebbe allora la forma

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

Un semplice esempio

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.

Formalizziamo questo discorso.

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

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

Chiamiamo $\mathcal{T}_{(L,R,S)}$ il nuovo insieme di macchine di Turing.

$\mathcal{T}_{(L,R,S)}$ 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:

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

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 $\mathcal{T}_{(L,R,S)}$.

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

Se $M' = (Q, \Sigma, \Gamma, \delta', q_0, q_{accept}, q_{reject}) \in \mathcal{T}_{(L,R)}$, definiamo $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject}) \in \mathcal{T}_{(L,R,S)}$ con $\delta(q, \gamma) = \delta'(q, \gamma)$, per ogni $(q, \gamma) \in (Q \setminus \{q_{accept}, q_{reject}\}) \times \Gamma$.

Ovviamente $L(M) = L(M')$.

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 $L(M) = L(M')$.

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).

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:

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

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.*

Macchina di Turing multinastro

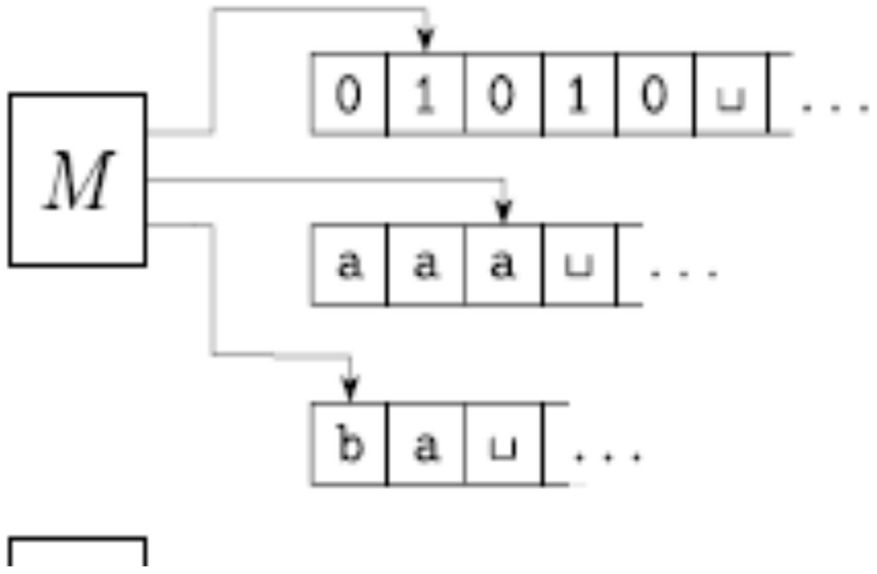


Figura tratta da M. Sipser, Introduzione alla teoria della computazione.

Funzione di transizione di una MdTM

Il codominio della funzione di transizione di una macchina di Turing a k nastri è un insieme di sequenze $(q_j, b_1, \ldots, b_k, d_1, \ldots, d_k)$ di lunghezza $(2k + 1)$ dove:

- $q_j \in Q$
- $b_t \in \Gamma$, per $t \in \{1, \ldots, k\}$
- $d_t \in \{L, R, S\}$, per $t \in \{1, \ldots, k\}$

Funzione di transizione di una MdTM

Se $\delta(q_i, a_1, \ldots, a_k) = (q_j, b_1, \ldots, b_k, d_1, \ldots, d_k)$ e se M si trova nello stato q_i con le k testine - una per nastro - posizionate su celle contenente $a_1, \ldots, a_k$ rispettivamente, alla fine della transizione:

- M si troverà nello stato q_j,
- $b_t \in \Gamma$ sarà il simbolo scritto sulla cella del t-esimo nastro su cui la testina si trovava all'inizio della transizione (contenente a_t), per $t \in \{1, \ldots, k\}$
- la testina sul t-esimo nastro si troverà sulla stessa cella cui si trovava all'inizio della computazione se $d_t = S$, si sarà spostata sulla cella di sinistra se (tale cella esiste e se) $d_t = L$, si sarà spostata sulla cella di destra se $d_t = R$, per $t \in \{1, \ldots, k\}$.

Computazione di una MdTM

Una MdTM $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ inizia la computazione:

- partendo dallo stato iniziale q_0,
- con l'input $w \in \Sigma^*$ posizionato sulla parte più a sinistra del primo nastro: le prime n celle a sinistra, se $n = |w|$ è la lunghezza dell'input (il resto delle celle conterrà il carattere $\sqcup$).
- Gli altri nastri conterranno solo il carattere $\sqcup$.
- Tutte le testine saranno posizionate sulle prime celle dei rispettivi nastri.
- In particolare, la testina del primo nastro sarà posizionata sulla cella contenente il primo simbolo di input (quello più a sinistra), se tale input è diverso dalla stringa vuota.

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 $q \in Q$ è lo stato corrente di M e, per $t \in \{1, \ldots, k\}$, $u_t q v_t \in \Gamma^* Q \Gamma^S$, con $\Gamma^S = \Gamma^*(\Gamma \setminus \{\sqcup\}) \cup \{\sqcup, \epsilon\}$; la testina del t-esimo nastro è posizionata sul primo simbolo di v_t se $v_t \neq \epsilon$, su $\sqcup$ altrimenti; $u_t v_t$ è il contenuto del t-esimo nastro, con la convenzione di aver eliminato tutti i $\sqcup$ che seguono l'ultimo carattere di v_t se $v_t \neq \epsilon$, tutti i simboli $\sqcup$ che seguono u_t altrimenti.

Computazione di una MdTM

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

$(q_0 w, \ldots, q_0)$ 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

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

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

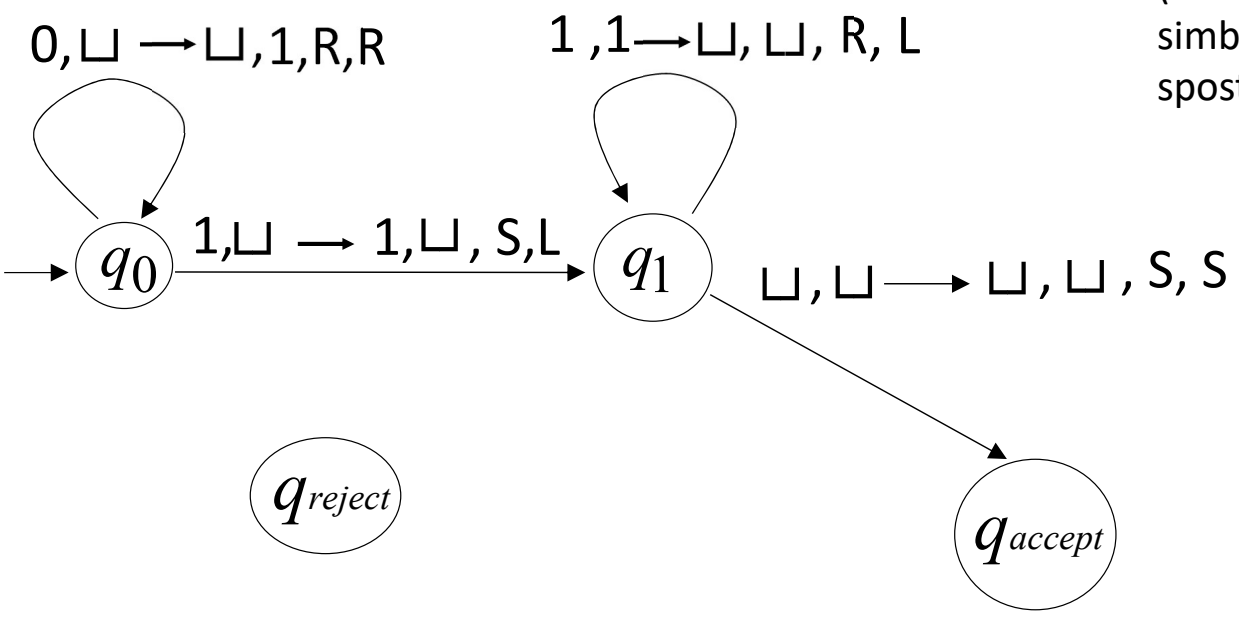
1. Scorre il primo nastro verso destra fino al primo `1`: per ogni `0`, scrive un `1` sul secondo nastro
2. Scorre il primo nastro verso destra e il secondo nastro verso sinistra: se i simboli letti non sono uguali, termina in q_{reject}
3. Se legge $\sqcup$ su entrambi i nastri, termina in q_{accept}

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

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)

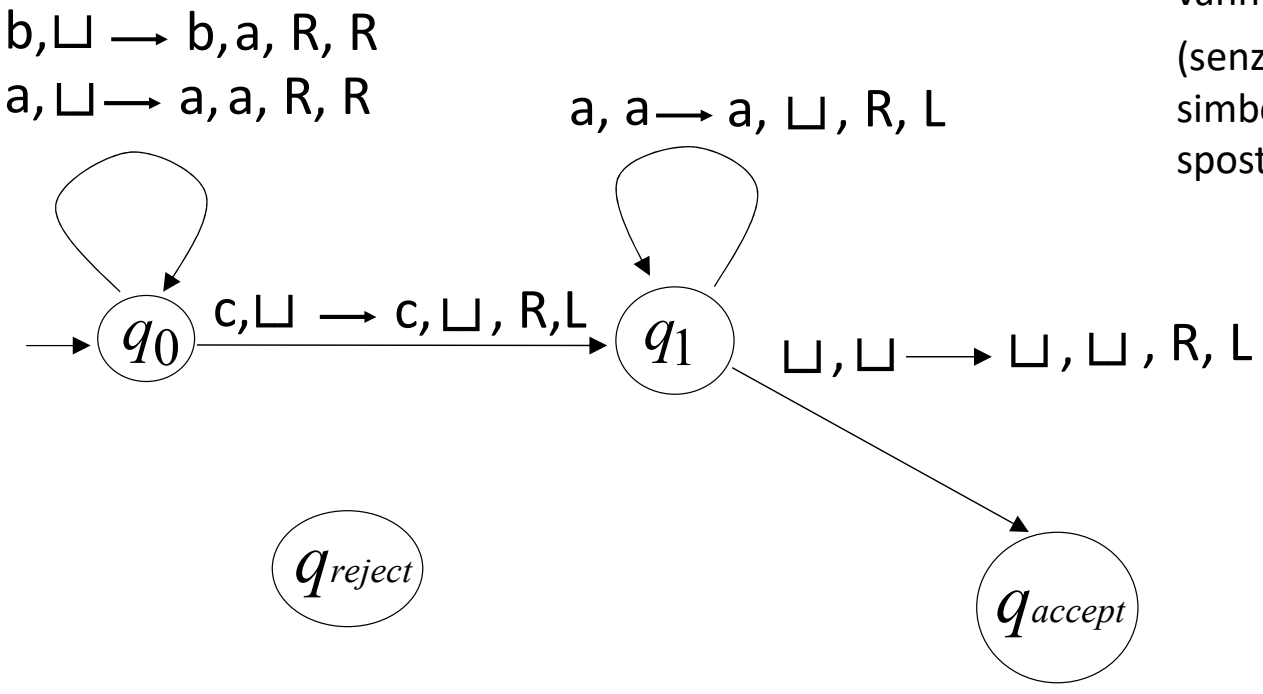
Esempio: MdT a 2 nastri per $\{wca^{|w|} \mid w \in \{a, b\}^*\}$

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

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

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

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



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

Teorema

Per ogni macchina di Turing multinastro M esiste una macchina di Turing (a nastro singolo) S equivalente a M, cioè tale che $L(M) = L(S)$.

Dimostrazione

Faremo vedere che esiste una macchina di Turing (a nastro singolo) $S = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ equivalente a M.

Nella dimostrazione useremo il termine "contenuto del nastro": è la stringa dei simboli che restano sul nastro dopo aver eliminato tutti i simboli $\sqcup$ che seguono l'ultimo carattere del nastro diverso da $\sqcup$.

Supponiamo che M abbia k nastri.

Per ogni configurazione di M

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

la macchina S che simula M deve codificare su un solo nastro:

- il contenuto $u_1 v_1, \ldots, u_k v_k$ dei k nastri,
- la posizione di ciascuna testina su ogni nastro.

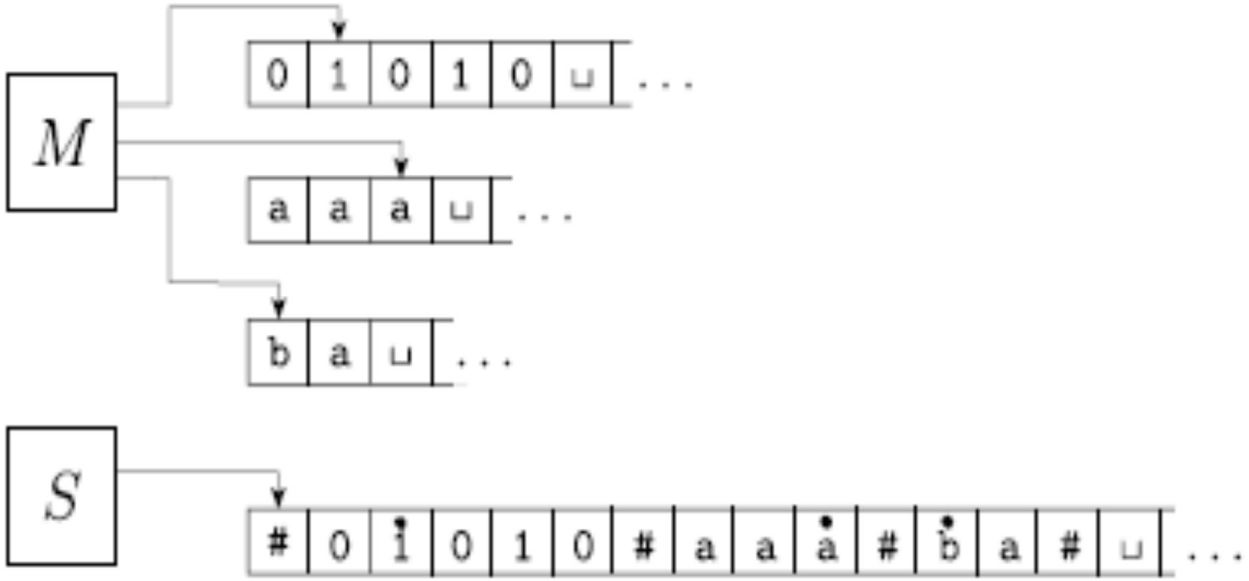


Figura tratta da M. Sipser, Introduzione alla teoria della computazione.

- Il contenuto del nastro di S sarà la concatenazione di k blocchi separati da $\#$ (seguita da $\sqcup$).
- Ogni blocco corrisponderà a un nastro di M e avrà una lunghezza variabile che dipende dal contenuto del nastro corrispondente.
- Un elemento $\dot{\gamma}$, con $\gamma \in \Gamma$, nel blocco t-esimo indica la posizione della testina del nastro t-esimo, $t \in \{1, \ldots, k\}$.

- Quindi a una configurazione di M

$$(u_1 q a_1 v_1, \ldots, u_k q a_k v_k)$$

corrisponderà la configurazione di S

$$q' \# u_1 \dot{a}_1 v_1 \# \ldots \# u_k \dot{a}_k v_k \#$$

- Nota: Se Γ è l'alfabeto dei simboli di nastro di M, l'alfabeto dei simboli di nastro di S sarà $\Gamma \cup \{\dot{\gamma} \mid \gamma \in \Gamma\} \cup \{\#, \dot{\#}\}$.

Sia $w = w_1 \cdots w_n$ una stringa input, $w_t \in \Sigma$, $t \in \{1, \ldots, k\}$.

1. **(generazione della configurazione iniziale di M)** S passa dalla configurazione iniziale $q_0 w_1 \cdots w_n$ alla configurazione

$$q' \# \dot{w}_1 \cdots w_n \# \dot{\sqcup} \# \ldots \# \dot{\sqcup} \#$$

   (**Nota:** Servono stati addizionali).
2. Per simulare un passo di computazione di M, per effetto dell'applicazione di $\delta(q, a_1, ..., a_k) = (s, b_1, ..., b_k, d_1, ..., d_k)$, la macchina S scorre il dato sul nastro dal primo $\#$ al $(k+1)$-esimo $\#$ da sinistra a destra e **viceversa** due volte:
   - la prima volta S determina quali sono i simboli correnti di M, $\dot{a}_t$, memorizzando nello stato i simboli marcati sui singoli nastri (**Nota:** Servono stati addizionali).
   - la seconda volta S esegue su ogni sezione (cioè un nastro di M) le azioni che simulano quelle di M: scrittura ed eventuale spostamento delle testine (in particolare eventuale spostamento dei marcatori).

- Nota. Nel corso della simulazione S potrebbe spostare una delle testine virtuali (puntino) su un delimitatore $\#$. Questo significa che M ha spostato la testina del nastro (corrispondente al segmento del nastro di S) sulla parte di nastro contenente solo $\sqcup$ (sulla cella a destra del carattere diverso da $\sqcup$ più a destra del suo nastro). In questo caso, S cambia $\dot{\#}$ con $\dot{\sqcup}$ e deve spostare di una posizione tutto il suffisso del nastro a partire da $\dot{\#}$ (cambiato con $\#$) fino all'ultimo $\#$ a destra.
- Poi la MdT entra nello stato che ricorda il nuovo stato di M e riposiziona la testina all'inizio del nastro.
- Infine, se M si ferma, anche S si ferma, eventualmente rimuovendo tutti i separatori $\#$ e sostituendo i caratteri $\dot{a}_t$ con a_t.

□

Teorema

Un linguaggio L è Turing riconoscibile se e solo se esiste una macchina di Turing multinastro M che lo riconosce, cioè tale che $L(M) = L$.

Dimostrazione.

Se L è Turing riconoscibile allora esiste una MdT M tale che $L(M) = L$. Poiché M è una MdT a k nastri con $k = 1$, esiste una macchina di Turing multinastro M tale che $L(M) = L$.

Viceversa, se esiste una macchina di Turing multinastro M tale che $L(M) = L$, per il teorema precedente esiste una macchina di Turing (a nastro singolo) S tale che $L(S) = L(M) = L$. Quindi L è Turing riconoscibile.

□

Osservazione

- Abbiamo dimostrato che le MdTM hanno lo stesso potere delle MdT.
- Introdurremo una misura del tempo necessario per eseguire una computazione che termina.
- Più precisamente considereremo solo MdT che si fermano su ogni input.
- Assoceremo a ogni MdT di questo tipo una funzione che stimi il tempo che le è necessario per stabilire se w appartiene al linguaggio $L(M)$ deciso da M.

Definizione

Sia $\Sigma = \{a_0, \ldots, a_k\}$ *un alfabeto e sia* $a_0 < a_1 < \ldots < a_k$ *un ordinamento degli elementi di* Σ. *Siano* $x, y \in \Sigma^*$.
Diremo che $x \leq y$ *rispetto all'***ordine radix** *se* x *e* y *verificano una delle condizioni seguenti:*

1. $|x| < |y|$.
2. $|x| = |y|$ *e* $x = zax'$, $y = zby'$, *con* $z, x', y' \in \Sigma^*$, $a, b \in \Sigma$ *e* $a \leq b$.

In "Introduzione alla teoria della computazione", l'ordine radix è chiamato **ordine per lunghezza**.
La lista dei numeri in base k in ordine crescente è una lista di stringhe su $\Sigma = \{0, \ldots, k-1\}$ in ordine radix.

Ordine radix

- Esempio. Supponiamo $a < b < \ldots < z$.

  barca $<$ aurora, castagna $<$ castello.
- La lista delle stringhe su $\Sigma = \{0, 1\}$, con $0 < 1$, in ordine radix:

$$\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, \ldots$$

- La lista delle stringhe su $\Sigma = \{a, b, c\}$, con $a < b < c$, in ordine radix:

$$\epsilon, a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc, \ldots$$

Macchina di Turing non deterministica

Introduciamo un'altra variante della macchina di Turing deterministica, detta macchina di Turing non deterministica. Essenzialmente, una macchina di Turing non deterministica differisce da una deterministica per il fatto che una configurazione può evolvere in più di una configurazione successiva.

Vedremo che le macchine non deterministiche hanno lo stesso potere computazionale di quelle deterministiche.

Tuttavia vedremo che le macchine di Turing deterministiche possono simulare quelle non deterministiche con una perdita di efficienza **esponenziale**.

Macchina di Turing non deterministica

Definizione (Macchina di Turing non deterministica)

Una Macchina di Turing **non deterministica** *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*
- *la funzione di transizione* δ *è definita al modo seguente:*

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

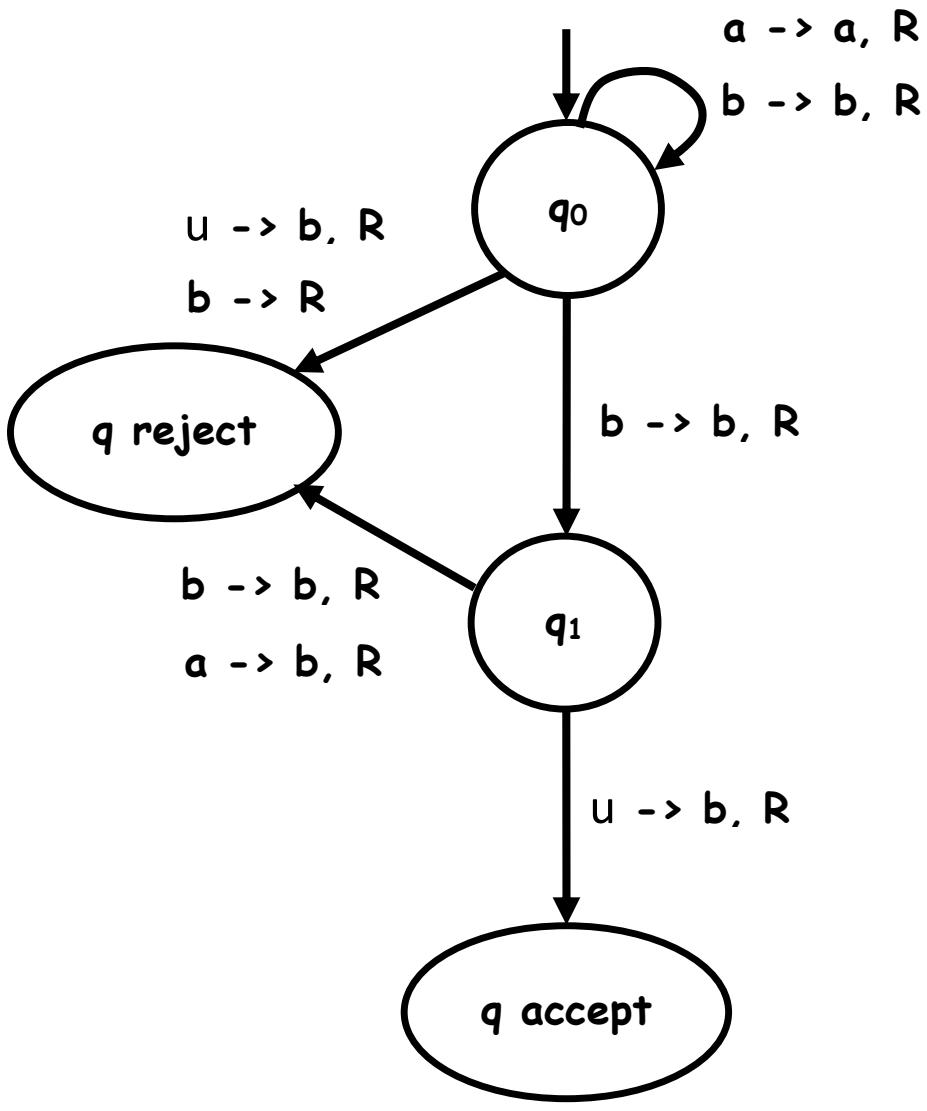
Macchina di Turing non deterministica

Quindi per ogni $q \in Q \setminus \{q_{accept}, q_{reject}\}$, per ogni $\gamma \in \Gamma$, risulta $\delta(q, \gamma) = \{(q_1, \gamma_1, d_1), \ldots, (q_k, \gamma_k, d_k)\}$, con $k \geq 0$ e $(q_j, \gamma_j, d_j) \in Q \times \Gamma \times \{L, R\}$, per $j \in \{1, \ldots, k\}$.

Macchina di Turing non deterministica

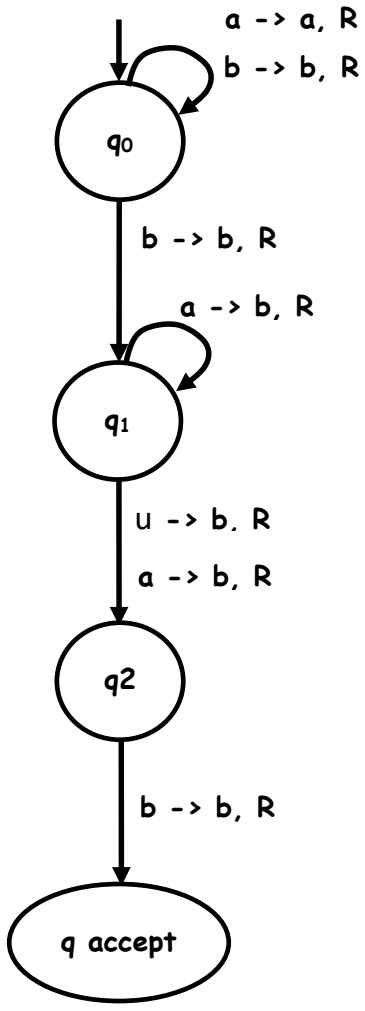
Nei tre esempi di macchine di Turing non deterministiche che seguono l'alfabeto input è $\{a, b\}$ e l'alfabeto di nastro è $\{a, b, \sqcup\}$.

Macchina di Turing M_1



$$\delta(q_0, b) = \{(q_0, b, R), (q_1, b, R), (q_{reject}, b, R)\}$$

Macchina di Turing M_2



$\delta(q_0, b) = \{(q_0, b, R), (q_1, b, R)\},$
$\delta(q_1, a) = \{(q_1, b, R), (q_2, b, R)\}$

Macchina di Turing non deterministica

- Le configurazioni hanno la stessa forma uqv di quelle definite per una Macchina di Turing deterministica.
- Allo stesso modo non cambia il passo di computazione $C_1 \to C_2$ e una computazione $C \to^* C'$ continua a essere una successione finita (**non un albero!**) di configurazioni.
- Poiché però la macchina di Turing è non deterministica, ci possono essere più configurazioni $u'q'v'$ che sono prodotte da uqv in **un solo** passo.

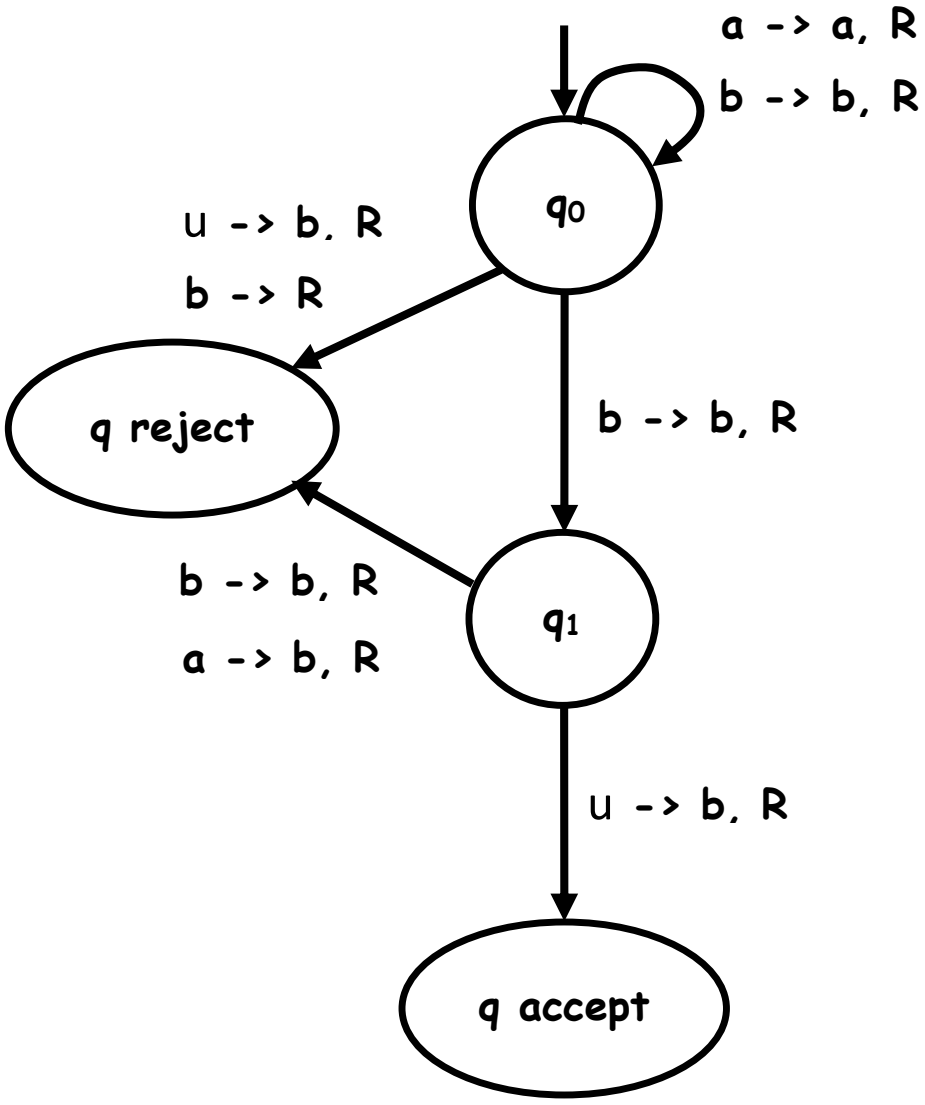
Macchina di Turing non deterministica

Quindi, come in un NFA, le computazioni (a partire da una configurazione) possono essere organizzate in un albero in cui la radice è la configurazione di partenza (tipicamente quella iniziale), i nodi sono configurazioni i cui figli rappresentano le possibili configurazioni raggiungibili da quel nodo.

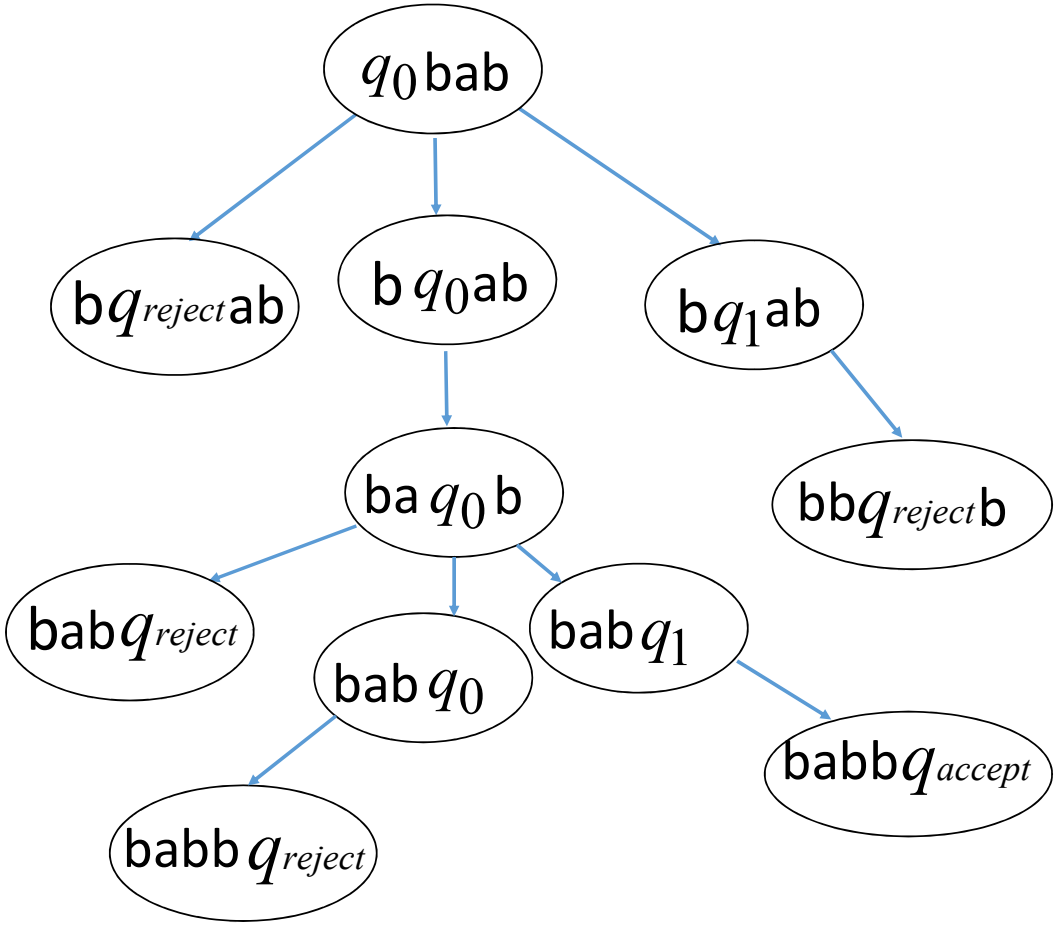
Una computazione è **completamente determinata** da una sequenza di scelte tra le varie configurazioni raggiungibili passo dopo passo.

Quindi una computazione può essere rappresentata da un cammino nell'albero.

Macchina di Turing M_1



Albero delle computazioni su *bab*



Computazioni in una Macchina di Turing non deterministica

Come per le macchine di Turing deterministiche, partendo dalla configurazione $q_0 w$, la computazione può terminare (se si raggiunge una configurazione di arresto) o non terminare.

Computazioni che non terminano

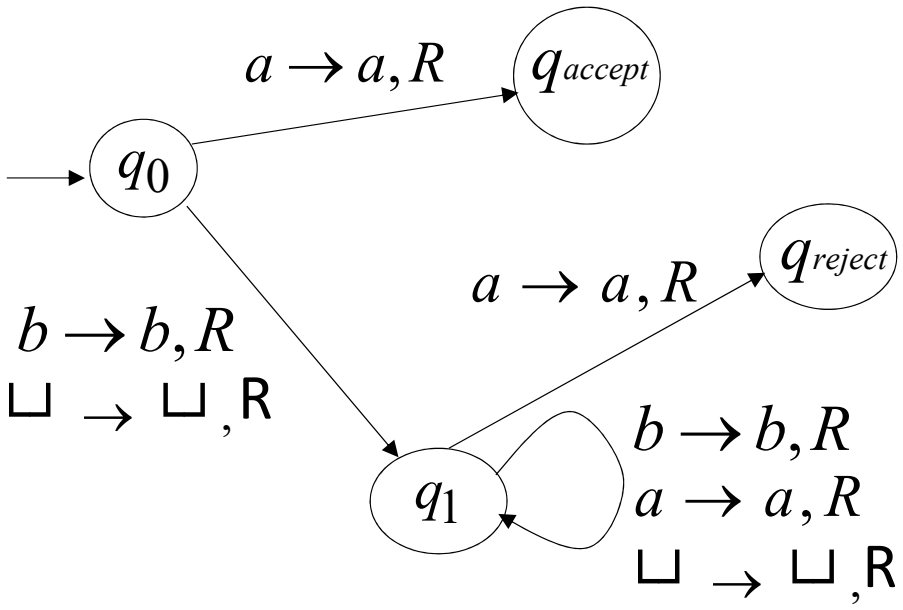
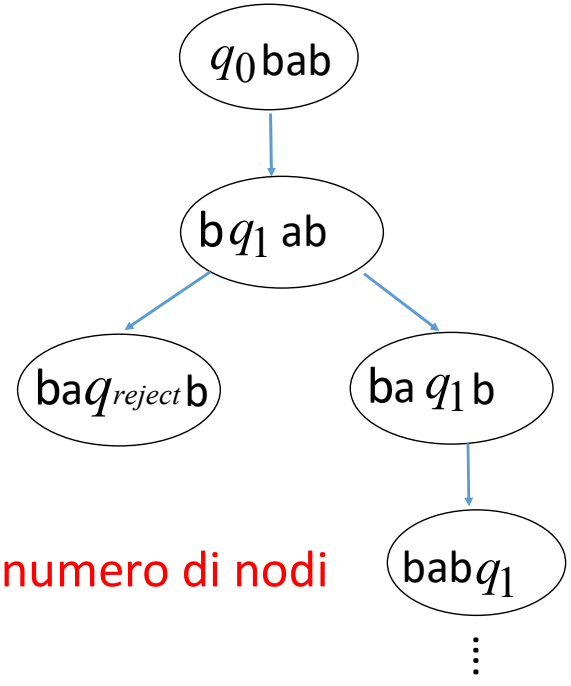


Figura: N_1

Computazioni che non terminano



Albero con infinito numero di nodi

L'albero delle computazioni di N_1 su *bab* non ha un numero finito di nodi.

Macchina di Turing non deterministica

In una macchina di Turing non deterministica, le computazioni su una stringa input w possono essere organizzate in un albero radicato in cui la radice è la configurazione iniziale $q_0 w$ **(albero delle computazioni)**.

I nodi sono configurazioni e i figli di ogni nodo rappresentano le possibili configurazioni raggiungibili da quel nodo in un passo di computazione.

In particolare, ogni configurazione in cui lo stato è q_{reject} o q_{accept} è una foglia.

Macchina di Turing non deterministica

Rispetto alla macchina di Turing deterministica, c'è un'asimmetria tra la definizione di accettazione di una stringa e quella di rifiuto di una stringa.
Sia N una macchina di Turing non deterministica e sia w una stringa.

- N accetta w se e solo se esiste almeno una computazione $q_0 w \rightarrow^* u q_{accept} v$, dove $q_0 w$ è la configurazione iniziale di N con input w e $u q_{accept} v$ è una configurazione di accettazione. L'albero delle computazioni di N su w contiene almeno una foglia con stato q_{accept}.

- N rifiuta w se e solo se ogni computazione dalla configurazione iniziale $q_0 w$ con input w termina in una configurazione di rifiuto. L'albero delle computazioni di N su w è finito e tutte le foglie sono configurazioni con stato q_{reject}.

Linguaggio riconosciuto da una MdT non deterministica

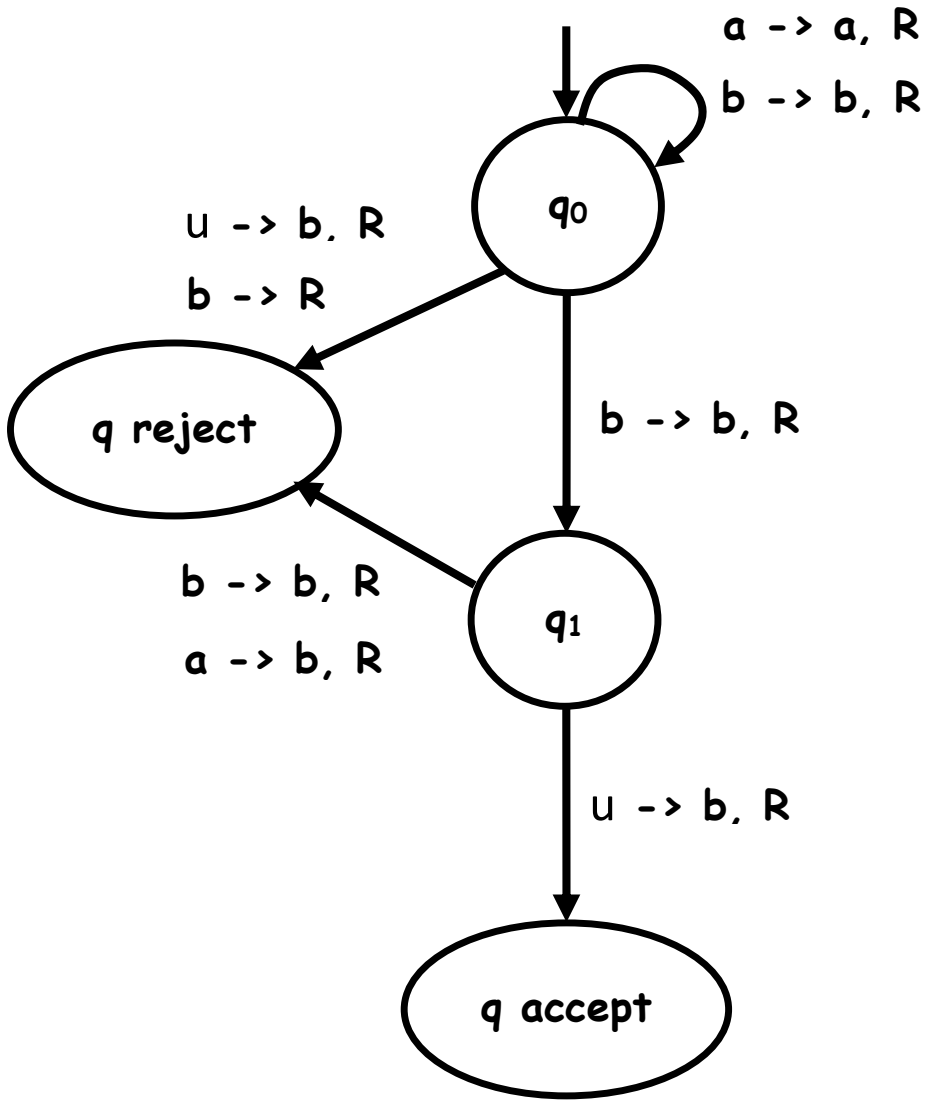
Definizione

Sia $N = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ *una MdT non deterministica. Il linguaggio* $L(N)$ *riconosciuto da* N*, è l'insieme:*

$$L(N) = \{w \in \Sigma^* \mid \text{esiste una computazione } q_0 w \rightarrow^* u q_{accept} v, \; u, v \in \Gamma^*\}$$

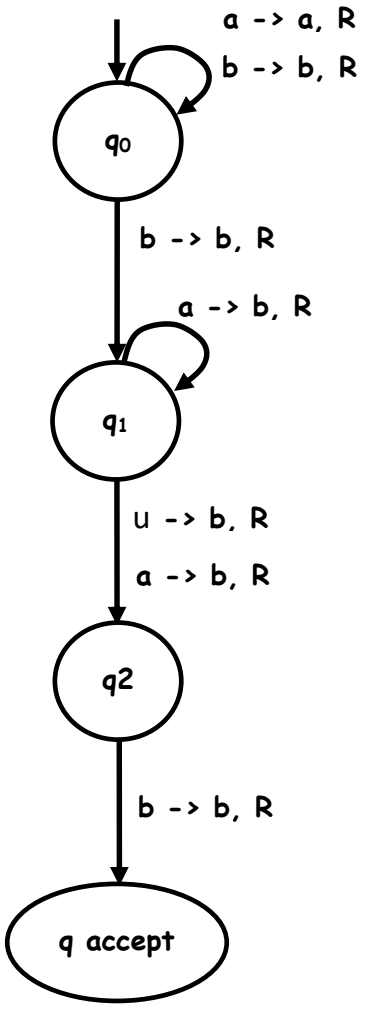
- Non considereremo funzioni calcolate da una MdT non deterministica.

Macchina di Turing M_1



$L(M_1) = \{wb \mid w \in \{a, b\}^*\}$

Macchina di Turing M_2



$$L(M_2) = \{xba^n by \mid x, y \in \{a, b\}^*, n \in \mathbb{N}, n \geq 1\}$$

Macchina di Turing N_1

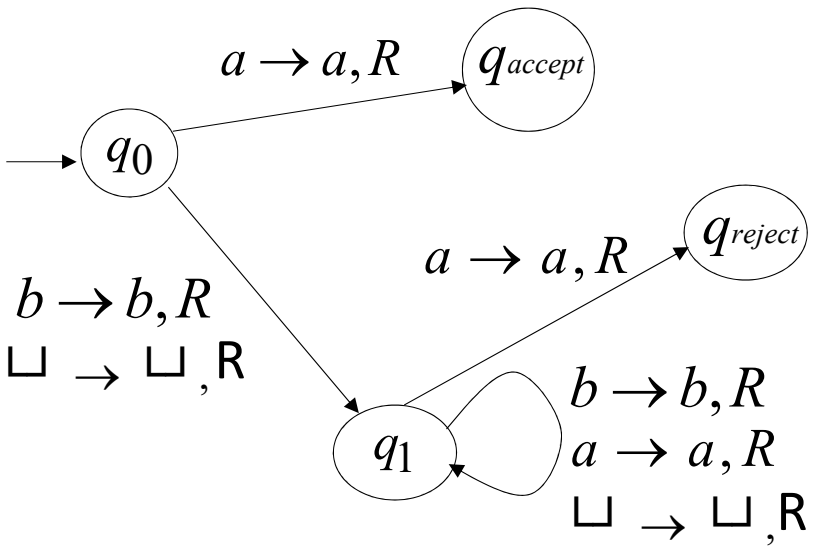


Figura: N_1

$L(N_1) = \{aw \mid w \in \{a, b\}^*\}$

Equivalenza tra il modello deterministico e quello non deterministico

Teorema

Per ogni macchina di Turing non deterministica N esiste una macchina di Turing deterministica D equivalente ad N, cioè tale che $L(N) = L(D)$.

Idea della prova

- Ogni computazione di N deriva da una sequenza di scelte che D deve riprodurre.

  D simula N provando tutte le possibili scelte che può fare N durante la sua computazione nondeterministica.
- Quindi, per ogni input w, D esplora l'albero delle computazioni di N su w.
- Se D trova lo stato di accettazione su uno qualsiasi dei rami dell'albero, allora D accetta. Altrimenti, la simulazione effettuata da D **non terminerà**.
- Questo assicura che $L(D) = L(N)$.

Idea della prova

- Una strategia sbagliata: visitare l'albero in profondità (la visita rischierebbe di rimanere "incastrata" in un cammino corrispondente a una computazione che non termina escludendo un'eventuale accettazione).
- Una strategia vincente: visitare l'albero per livelli (se esiste una computazione $q_0 w \rightarrow^* u q_{accept} v$, D prima o poi effettuerà la sequenza di scelte corrispondenti).
- D utilizza una visita per livelli dell'albero. Questo metodo garantisce che D visita ogni nodo dell'albero finché non incontra una configurazione di accettazione, se una tale configurazione è una foglia dell'albero.

Rappresentazione delle computazioni

- Vogliamo rappresentare le computazioni di N su una stringa w, cioè i cammini nell'albero delle computazioni di N su w mediante una stringa.
- **Perché?**
- **Come?**

Rappresentazione delle computazioni - Perché?

- Vogliamo rappresentare le computazioni di N su una stringa w, cioè i cammini nell'albero delle computazioni di N su w mediante una stringa.
- **Perché?**
- La MdT deterministica D che simula N ha tre nastri. La macchina D utilizza i suoi tre nastri in maniera particolare, come illustrato nella figura seguente.

Macchina di Turing non deterministica

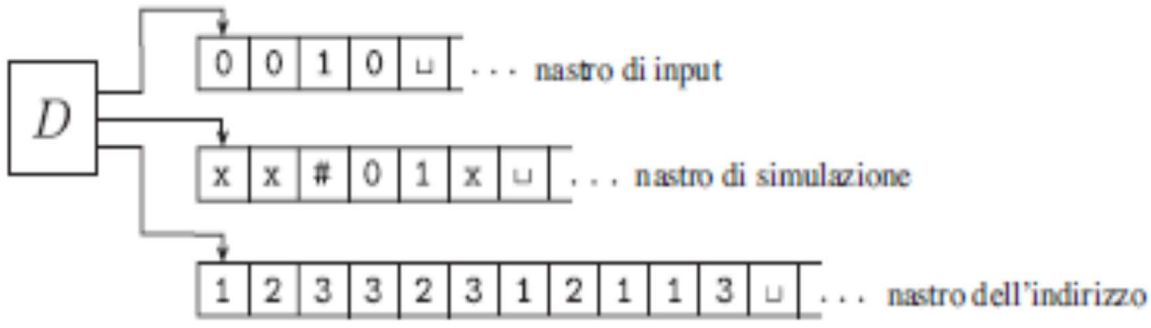


FIGURA 3.17
La TM deterministica D che simula la TM non deterministica N

Figura tratta da M. Sipser, Introduzione alla teoria della computazione.

Rappresentazione delle computazioni - Perché?

- La macchina D utilizza tre nastri.
- Il nastro 1 contiene sempre la stringa di input e non viene mai modificato.
- Il nastro 2 mantiene una copia del nastro di N corrispondente a qualche diramazione della sua computazione non deterministica.
- Il nastro 3 deve tenere traccia della posizione di D nell'albero delle computazioni di N.
- Sul nastro di una macchina di Turing possiamo scrivere solo stringhe di simboli!
  Quindi tale posizione deve essere necessariamente rappresentata mediante una stringa.

Rappresentazione delle computazioni - Come?

- Vogliamo rappresentare le computazioni di N su una stringa w, cioè i cammini nell'albero delle computazioni di N su w mediante una stringa.
- **Come?**

Rappresentazione delle computazioni

Sia $N = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ una macchina di Turing non deterministica.

Sia b il massimo numero di scelte per N su ogni stato e ogni carattere:

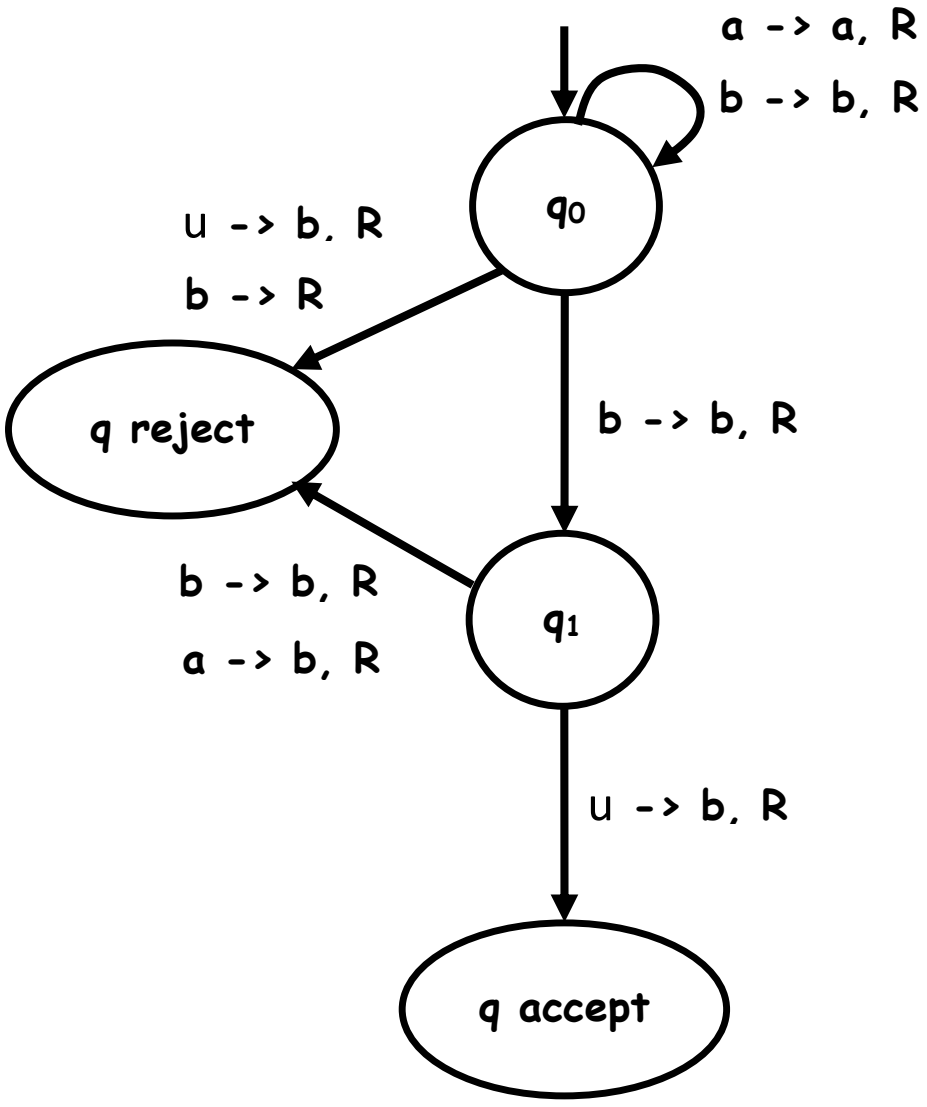
$$b = \max\{\delta(q, \sigma) \mid q \in Q, \sigma \in \Sigma\}$$

Consideriamo l'alfabeto $\Gamma_b = \{1, \ldots, b\}$.

Per ogni coppia stato-simbolo (q, σ) esistono al più b scelte a ognuna delle quali associamo un diverso simbolo in Γ_b.

Quindi per ogni configurazione $uq\sigma v$ esistono al più b successive configurazioni a ognuna delle quali risulta associato un diverso simbolo in Γ_b.

Rappresentazione dei cammini



$\Gamma_b = \{1, 2, 3\}$.

Per la MT della figura precedente risulta $\Gamma_b = \{1, 2, 3\}$ e $\delta(q_0, b) = \{(q_0, b, R), (q_1, b, R), (q_{reject}, b, R)\}$.

Associamo 1 a (q_{reject}, b, R), 2 a (q_0, b, R) e 3 a (q_1, b, R).

Associamo 1 alle altre triple, quelle in $\delta(q, \sigma)$ con $(q, \sigma) \neq (q_0, b)$.

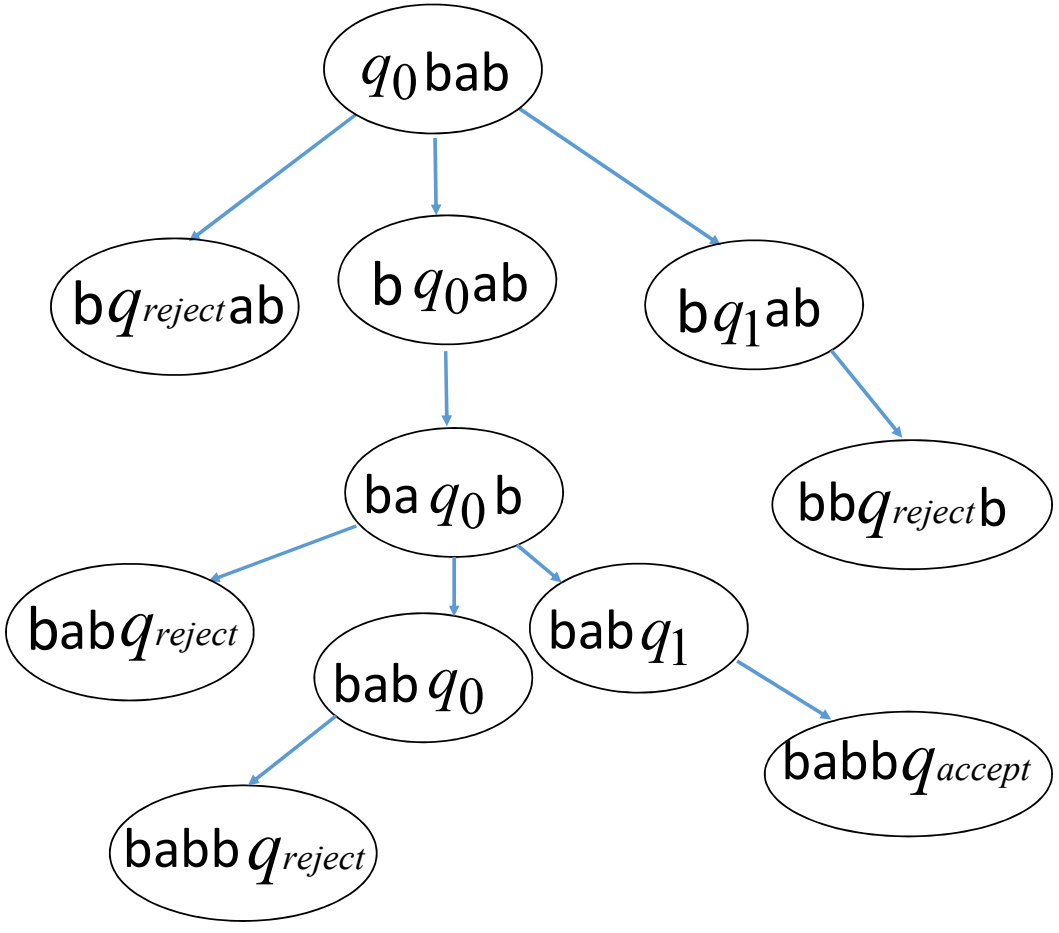
Rappresentazione delle computazioni

Ogni albero di computazione di N è un albero b-ario, cioè ogni nodo ha al più b figli.

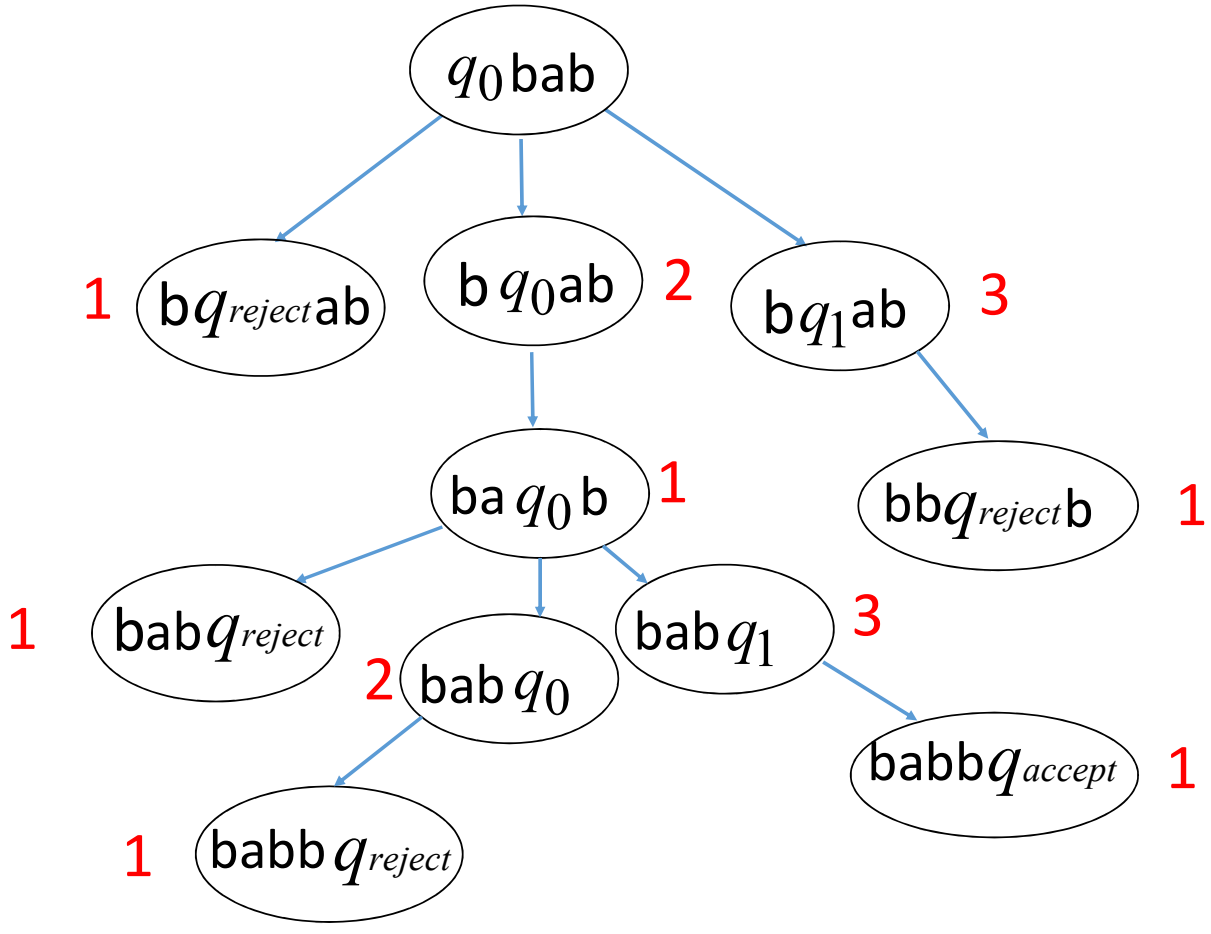
Ogni cammino in ogni albero di computazione può essere rappresentato da una stringa sull'alfabeto $\Gamma_b = \{1, \ldots, b\}$.

- Nota. Sul testo di Sipser si parla di indirizzo del nodo. La stringa che rappresenta un cammino è anche l'indirizzo del nodo finale del cammino.

Albero delle computazioni su *bab*



Rappresentazione delle computazioni su *bab*



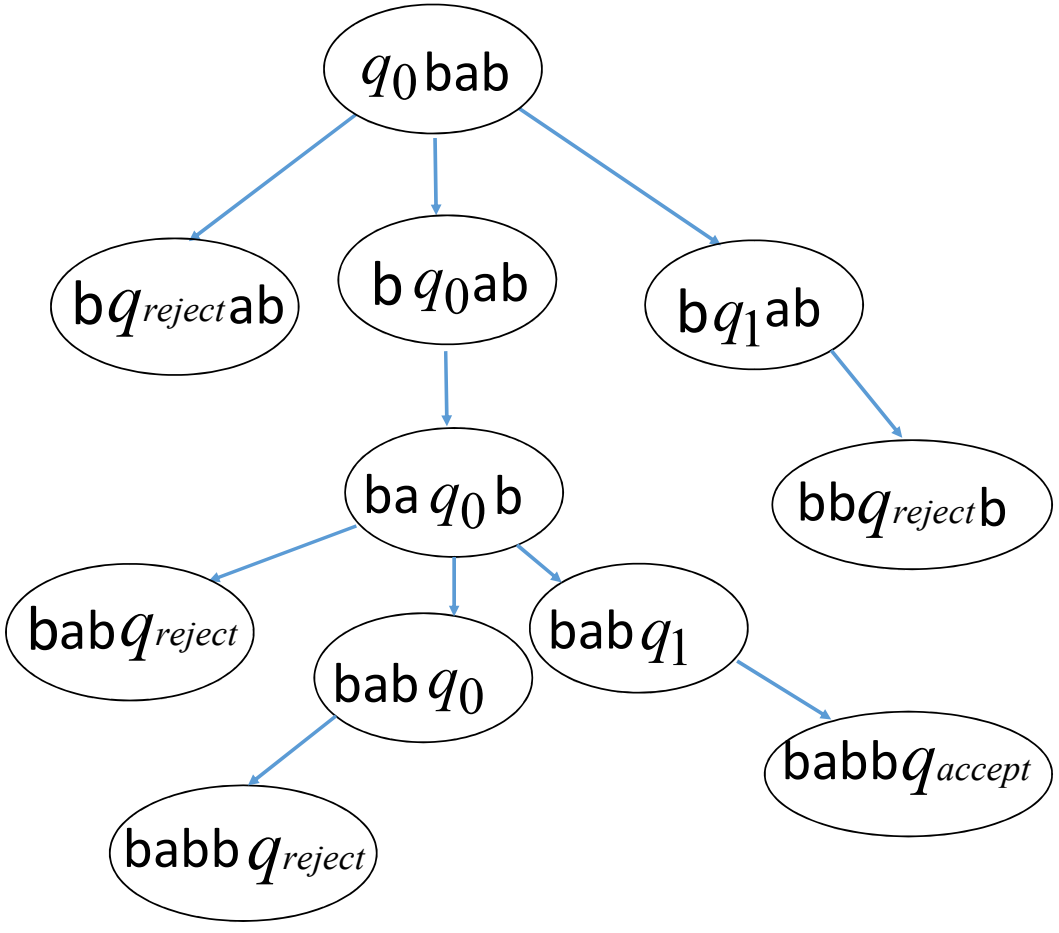
Codifica delle computazioni

Sia $\Gamma_b = \{1, \ldots, b\}$ l'alfabeto "delle scelte" di una macchina di Turing non deterministica.

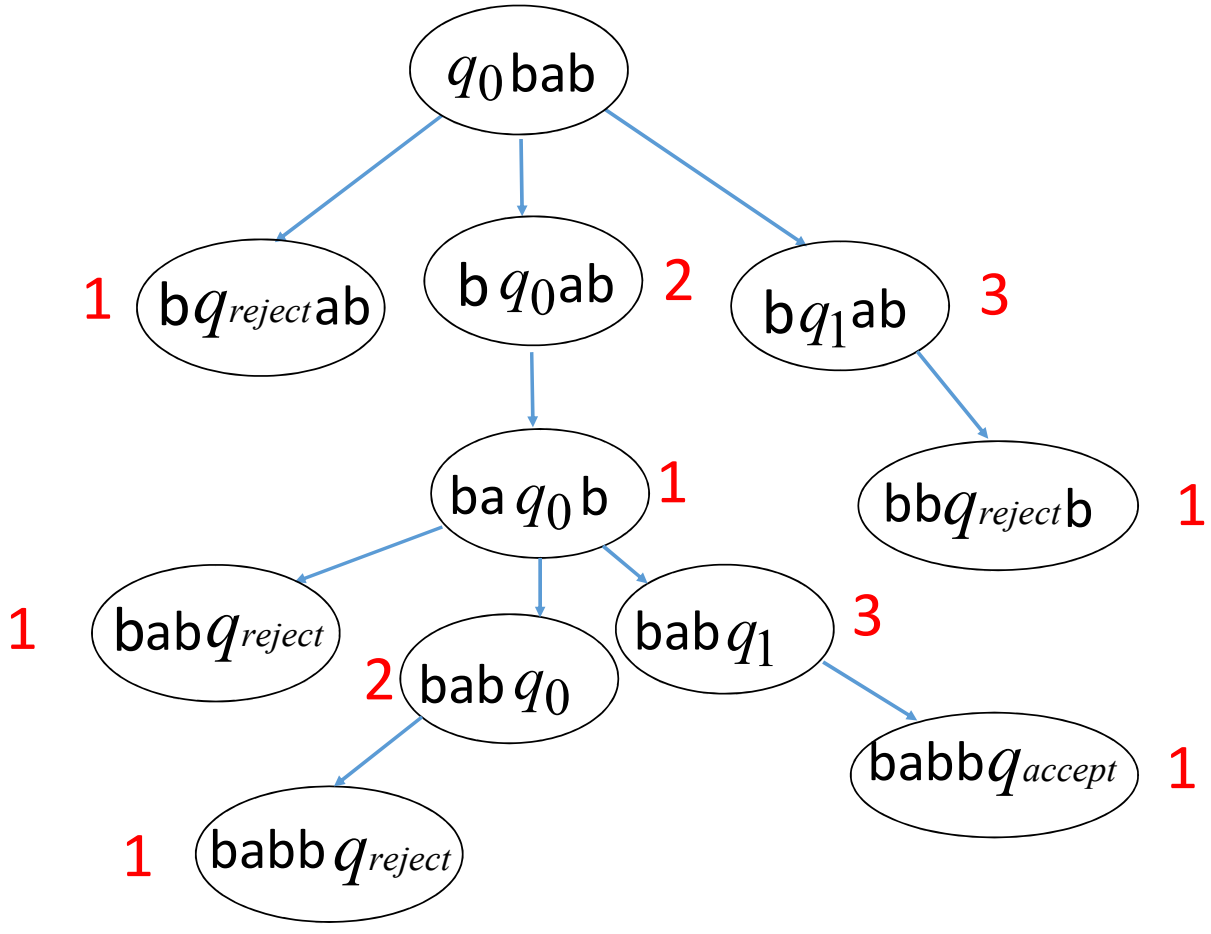
- A ogni computazione è associata una stringa su Γ_b
- Viceversa, a ogni stringa su Γ_b è associata al più una computazione

  (Non è sempre vero che una stringa rappresenta una computazione)

Albero delle computazioni su *bab*



Rappresentazione delle computazioni su *bab*



Codifica delle computazioni su *bab*

- 1 : $q_0bab \to bq_{reject}ab$
  2 : $q_0bab \to bq_0ab$
  3 : $q_0bab \to bq_1ab$
- 11: senza sbocco, muore. Lo stesso per 12 e 13.
- 21 : $q_0bab \to bq_0ab \to baq_0b$
  22, 23 senza sbocco, muore.

  ...
- La computazione che mostra che *bab* è accettata.

$$2131:$$

$$q_0bab \to bq_0ab \to baq_0b \to babq_1 \to babbq_{accept}$$

2131 codifica il cammino che termina in una foglia con stato q_{accept}. La stringa 2131 è l'indirizzo della foglia $babbq_{accept}$.

Rappresentazione dei cammini

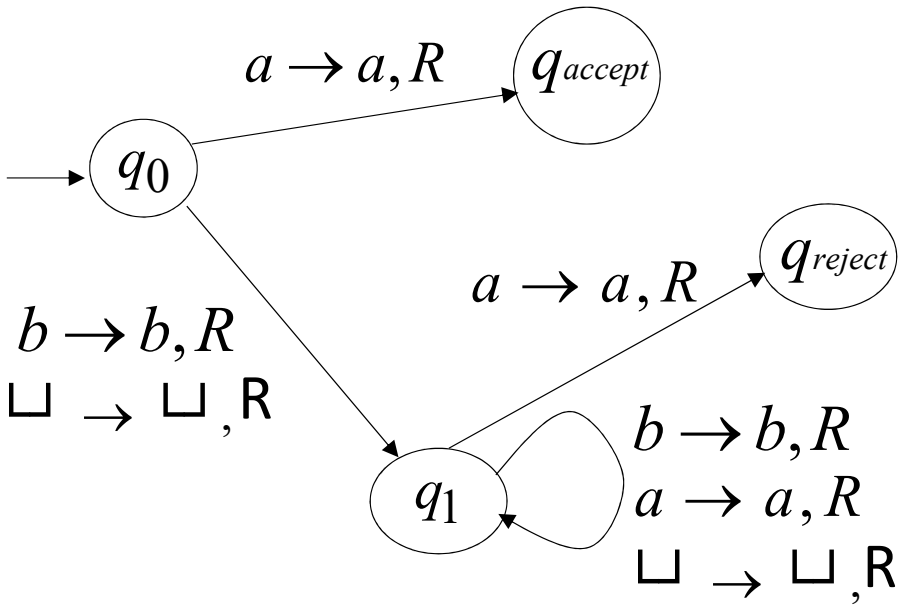
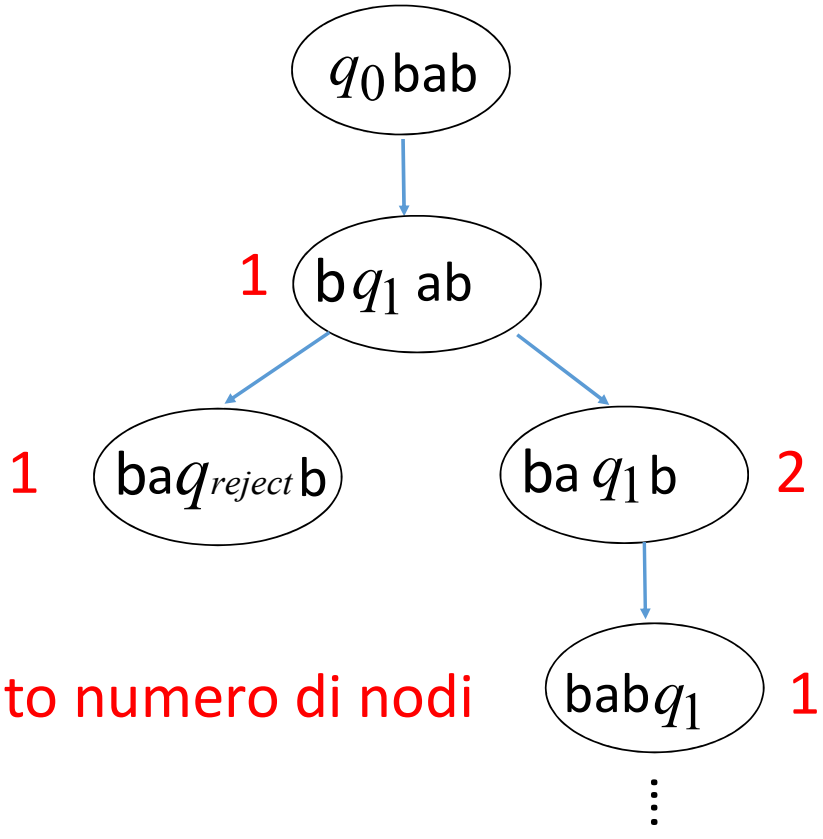


Figura: N_1

Per N_1 abbiamo $\Gamma_b = \{1, 2\}$.

L'albero delle computazioni di N_1 su *bab* non ha un numero finito di nodi.

Rappresentazione dei cammini



Albero con infinito numero di nodi

Rappresentazione dei cammini

Se una stringa rappresenta una computazione, ogni simbolo nella stringa rappresenta una scelta tra le possibili alternative proposte dalla δ in un passo di computazione.

La stringa vuota corrisponde alla configurazione iniziale, è l'indirizzo della radice dell'albero.

Una visita per livelli dell'albero corrisponde alla lista delle stringhe su Γ_b in ordine radix (in ordine di lunghezza crescente e, a parità di lunghezza, in ordine numerico).

La macchina D che simula N utilizzerà come "sottoprogramma" una macchina di Turing Sb che genera la successiva stringa binaria in ordine radix.

Macchina di Turing non deterministica

Teorema

Per ogni macchina di Turing non deterministica N esiste una macchina di Turing deterministica D equivalente ad N, cioè tale che $L(N) = L(D)$.

Dimostrazione

Data una macchina di Turing non deterministica $N = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, la macchina di Turing D che simula N ha tre nastri.

Sappiamo che esiste una macchina di Turing deterministica a un nastro equivalente ad D e quindi a N.

Macchina di Turing non deterministica

- Sul **primo nastro** è memorizzata la stringa input w (sulla parte più a sinistra) e il contenuto del primo nastro non verrà alterato dalle computazioni di D.
- Sul **secondo nastro** viene eseguita la simulazione di N. Il nastro 2 mantiene una copia del nastro di N corrispondente a qualche diramazione dell'albero delle computazioni.
- Sul **terzo nastro** vengono generate le codifiche delle possibili computazioni di N con input w.

  Il nastro 3 tiene traccia della posizione di D nell'albero delle computazioni di N.

Macchina di Turing non deterministica

Descrizione di D:

1. Inizialmente il nastro 1 contiene l'input w e i nastri 2 e 3 contengono solo $\sqcup$. (D inizializza la stringa sul nastro 3 a ϵ.)
2. D copia il contenuto del nastro 1 sul nastro 2.
3. Utilizza il nastro 2 per simulare N con input w sulla ramificazione della sua computazione non deterministica corrispondente alla stringa sul nastro 3, cioè riproduce la successione di scelte del corrispondente cammino (se tale stringa corrisponde a una computazione). Prima di ogni passo di N, consulta il simbolo successivo sul nastro 3 per determinare quale scelta fare tra quelle consentite dalla funzione di transizione di N. Se si raggiunge una configurazione di accettazione, D accetta l'input. Altrimenti (se si raggiunge una configurazione di rifiuto, se la stringa non corrisponde a una computazione, o al termine della simulazione sulla stringa) D passa al passo 4.
4. D genera sul nastro 3 la stringa successiva a quella corrente in ordine radix e torna al passo 2.

D accetta w se e solo se N accetta w, quindi $L(D) = L(N)$. $\square$

$\Gamma_b = \{1, 2\}$, $\delta(q_1, a) = \{(q_{reject}, a, R), (q_1, a, R)\}$

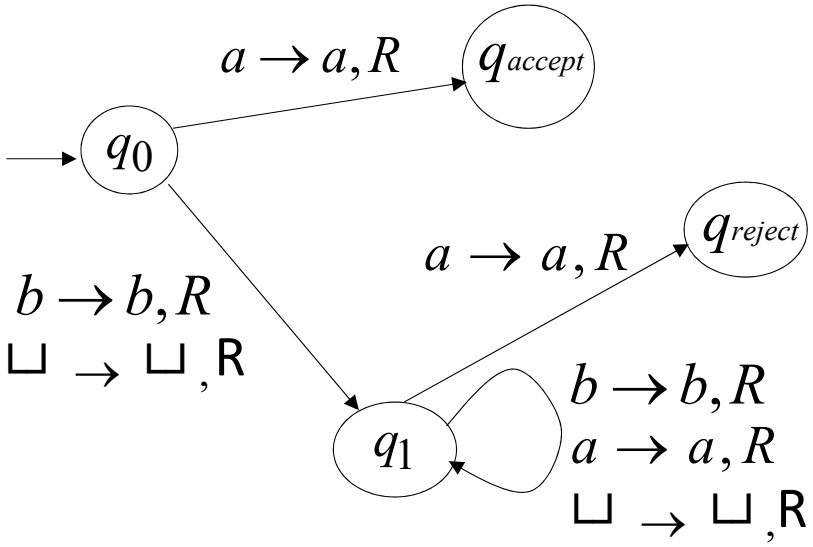


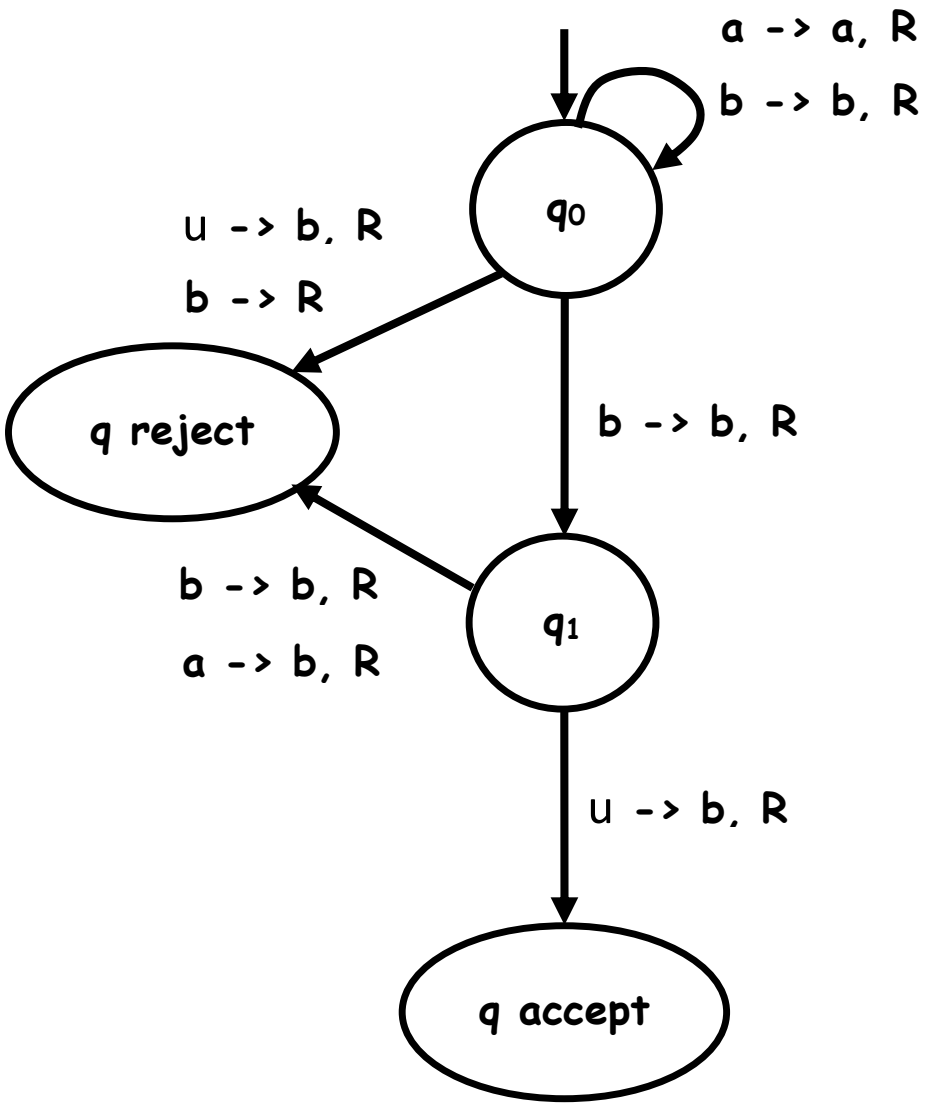
Figura: N_1

Esempio. $N_1 = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ è tale che $\delta(q_1, a) = \{(q_{reject}, a, R), (q_1, a, R)\}$. Supponiamo di associare 1 alla scelta (q_{reject}, a, R) e 2 alla scelta (q_1, a, R).

Poniamo $D = (Q', \Sigma, \Gamma', \delta', q'_0, q_{accept}, q'_{reject})$.

Sia p_1 lo stato che corrisponde a q_1 nella simulazione di N_1 (eventualmente $p_1 = q_1$). Allora
$\delta'(p_1, \sigma, a, 1) = (q_{reject}, \sigma, a, 1, S, R, R)$ e
$\delta'(p_1, \sigma, a, 2) = (p_1, \sigma, a, 2, S, R, R)$, per ogni carattere σ.

Esempio 2



$\Gamma_b = \{1, 2, 3\}$, $\delta(q_0, b) = \{(q_0, b, R), (q_1, b, R), (q_{reject}, b, R)\}$

Esempio. $M_1 = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ è tale che $\delta(q_0, b) = \{(q_0, b, R), (q_1, b, R), (q_{reject}, b, R)\}$. Supponiamo di associare 2 alla scelta (q_0, b, R), 3 alla scelta (q_1, b, R) e 1 alla scelta (q_{reject}, b, R).

Poniamo $D = (Q', \Sigma, \Gamma', \delta', q'_0, q_{accept}, q'_{reject})$.

Sia p_0 lo stato che corrisponde a q_0 nella simulazione di M_1 e p_1 lo stato che corrisponde a q_1 nella simulazione di M_1 (eventualmente $p_0 = q_0$ e $p_1 = q_1$). Allora
$\delta'(p_0, \sigma, b, 2) = (p_0, \sigma, b, 2, S, R, R)$,
$\delta'(p_0, \sigma, b, 3) = (p_1, \sigma, b, 1, S, R, R)$ e
$\delta'(p_0, \sigma, b, 1) = (q_{reject}, \sigma, b, 1, S, R, R)$, per ogni carattere σ.

Corollario

Un linguaggio L è Turing riconoscibile se e solo se esiste una macchina di Turing non deterministica N che lo riconosce, cioè tale che $L(N) = L$.

Dimostrazione

Se L è il linguaggio $L(N)$ riconosciuto da una macchina di Turing non deterministica N, per il teorema precedente esiste una MdT deterministica D tale che $L(D) = L(N) = L$.

Quindi L è Turing riconoscibile.

Macchina di Turing non deterministica

Viceversa se L è Turing riconoscibile allora esiste una macchina di Turing non deterministica $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ che lo riconosce, cioè tale che $L(M) = L$.

È facile vedere che la macchina di Turing non deterministica $M' = (Q, \Sigma, \Gamma, \delta', q_0, q_{accept}, q_{reject})$, dove $\delta'(q, \gamma) = \{\delta(q, \gamma)\}$, per ogni $q \in Q$ e $\gamma \in \Gamma$, è tale che $L(M') = L(M) = L$.

Decisore

- Una macchina di Turing non deterministica è un decisore se, per ogni stringa input w, tutte le computazioni a partire da $q_0 w$ terminano in una configurazione di arresto.

- Una macchina di Turing non deterministica N decide L se N è un decisore e $L = L(N)$.

- **Nota.** Sia N una macchina di Turing non deterministica tale che, per ogni input w, N accetta w o N rifiuta w.

  N non è necessariamente un decisore.

Linguaggio riconosciuto e linguaggio deciso da una MdT non deterministica

Anche per le macchine di Turing non deterministiche sono definite due famiglie di linguaggi:

- i linguaggi L riconosciuti da macchine di Turing non deterministiche;
- i linguaggi L decisi da macchine di Turing non deterministiche.

Macchina di Turing non deterministica

Abbiamo visto che una macchina di Turing non deterministica N può essere simulata da una MdT deterministica D. È possibile modificare la prova di questo risultato per provare che se N è una macchina di Turing non deterministica tale che, per ogni input w, N si ferma sempre su tutte le ramificazioni dell'albero delle computazioni su w, allora esiste una MdT deterministica D che simula N e che si arresta su ogni input (cioè D è un decisore). Diremo che una macchina di Turing non deterministica è un **decisore** se tutte le sue ramificazioni si fermano su ogni input.

Corollario

Un linguaggio L è decidibile se e solo se esiste una macchina di Turing non deterministica N che lo decide.

Osservazione

Sia N una macchina di Turing non deterministica che è un decisore.

Per ogni input w, l'albero delle computazioni di N su w ha un numero finito di nodi, in virtù del teorema seguente sugli alberi.

Teorema

Se ogni nodo in un albero ha un numero finito di figli e ogni cammino dell'albero ha un numero finito di nodi, allora l'albero ha un numero finito di nodi.

Corollario

Un linguaggio L è decidibile se e solo se esiste una macchina di Turing non deterministica N che lo decide.

Prova

Se un linguaggio L è decidibile, esiste una macchina di Turing deterministica M che è un decisore e che accetta L. M può essere facilmente trasformata in una macchina di Turing non deterministica che decide L.

Viceversa, se un linguaggio L è deciso da una macchina di Turing non deterministica N, definiamo una macchina di Turing deterministica D', modificando la precedente definizione di D come segue.

Descrizione di D':

1. Inizialmente il nastro 1 contiene l'input w e i nastri 2 e 3 contengono solo $\sqcup$. (D' inizializza la stringa sul nastro 3 a ϵ.)
2. D' copia il contenuto del nastro 1 sul nastro 2.
3. Utilizza il nastro 2 per simulare N con input w su una ramificazione della sua computazione non deterministica, riproducendo la successione di scelte del cammino corrispondente alla stringa sul nastro 3 (se tale stringa corrisponde a una computazione). Se si raggiunge una configurazione di accettazione, D' accetta l'input. Altrimenti (se si raggiunge una configurazione di rifiuto, se la stringa non corrisponde a una computazione, o al termine della simulazione sulla stringa) D' passa al passo 4.
4. Rifiuta se tutti i cammini dell'albero delle computazioni di N su w hanno portato a una configurazione di rifiuto. Altrimenti D' passa al passo 5.
5. D' genera sul nastro 3 la stringa successiva a quella corrente in ordine radix e torna al passo 2.

Possiamo dedurre che D' è un decisore per L.

Se N accetta il suo input w, allora D' troverà un cammino che termina in una configurazione di accettazione e D' accetta w.

Se N rifiuta il suo input w, tutte le sue computazioni su w terminano in una configurazione di rifiuto perché è un decisore. Quindi ciascuno dei cammini ha un numero finito di nodi poiché ogni arco nel cammino rappresenta un passo di una computazione di N su w. Pertanto l'intero albero di computazione di N su w è finito. Quindi, D' si fermerà e rifiuterà quando l'intero albero sarà stato esplorato.

La macchina di Turing D'

Nota che la macchina D' deve individuare se tutte le computazioni su w conducono a una configurazione di rifiuto.

Quindi D' deve avere un controllo sulle stringhe che rappresentano codifiche di computazioni su w che terminano in q_{reject}. Se x è una stringa che codifica una tale computazione, D' deve bloccare la generazione di stringhe in ordine radix con prefisso x.

Analogamente, se la stringa x è una stringa non valida, cioè non corrisponde a una computazione, D' deve bloccare la generazione di stringhe in ordine radix con prefisso x.

Altre varianti equivalenti

Altri esempi di varianti equivalenti sono:

- Macchina di Turing con nastro infinito (in entrambe le direzioni).
- Macchina di Turing con $|\Sigma| = 1$.
- Enumeratori (varianti di macchine di Turing per generare le stringhe in un linguaggio).

La definizione di algoritmo

- Algoritmi per eseguire calcoli in svariati campi erano noti già nell'antichità (esempio: Algoritmo di Euclide per il calcolo del MCD, circa 300 A.C.).
  Turing ha il merito di aver definito in modo formale il concetto di algoritmo.

- L'esigenza nacque a causa di un problema posto da Hilbert, al Congresso internazionale di Matematica del 1900 (Parigi), il decimo di una lista di 23 problemi.
- Il decimo problema di Hilbert si può riformulare come la richiesta di una soluzione algoritmica a un problema relativo ai polinomi.

Decimo problema di Hilbert

- Un polinomio è una somma di termini, dove ogni termine è il prodotto di alcune variabili e una costante, chiamata coefficiente.
- Una radice di un polinomio è un'assegnazione di valori alle sue variabili tale che il polinomio assume valore 0.
- Il decimo problema di Hilbert riguarda i polinomi a coefficienti interi e lo stabilire se hanno una radice in $\mathbb{Z}$.
- Esempio: il polinomio $6x^3yz^2 + 3xy^2 - x^3 - 10$ ammette la radice $x = 5$, $y = 3$, $z = 0$.
- Non tutti i polinomi hanno una radice intera.
  Ad esempio $x^2 + 1 = 0$ è un polinomio a coefficienti interi che non ha radici in $\mathbb{Z}$.

Decimo problema di Hilbert

- Decimo problema di Hilbert: progettare un algoritmo per decidere se un polinomio ha una radice in $\mathbb{Z}$.

  (Hilbert non usò il termine algoritmo)

- Vi era l'assunzione implicita che ogni enunciato potesse essere provato o confutato mediante un algoritmo.

La definizione di algoritmo

- Nota: per provare che un problema è risolubile mediante un algoritmo basta esibire l'algoritmo.
  Per provare che per un problema non esiste nessun algoritmo che lo risolve, è necessario definire in modo formale la nozione di algoritmo.

- Risolvendo negativamente un altro problema posto da Hilbert nel 1928 (il problema della decisione), Turing introdusse nel 1936 il modello formale che da lui prende il nome.
  Usando tale modello e basandosi sul lavoro di Davis, Putnam e Robinson, molti anni dopo (1970), Matijasevic provò l'indecidibilità del decimo problema: non esiste nessun algoritmo per decidere se un polinomio ha radici intere.

Equivalenza con altri modelli

- Molti altri modelli formali per la definizione di algoritmo furono dati negli stessi anni 30:
  - i sistemi di Post (da Post)
  - le funzioni μ-ricorsive (Godel, Herbrand)
  - il λ-calcolo (Church, Kleene)
  - la logica combinatoria (Schonfinkel, Curry)

- Fu dimostrato che tutti questi modelli erano equivalenti, cioè conducevano alla stessa classe di problemi risolvibili mediante algoritmi.

Tesi di Church

L'equivalenza dei vari modelli proposti condusse alla

Tesi di Church (o **Tesi di Church-Turing)**: La macchina di Turing è la definizione formale della nozione intuitiva di algoritmo.

La tesi di Church non è un teorema ed è indimostrabile.

Tenendo conto che essa è antecedente alla costruzione dei primi computer, costituì un enorme salto intellettuale.

L'evoluzione dei sistemi di calcolo ha generato negli ultimi anni una corrente di studi rivolti al superamento (o alla integrazione) del modello di Turing.