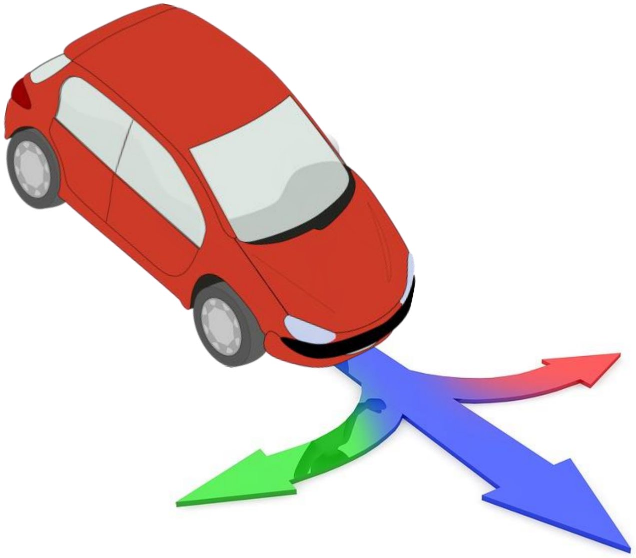


Macchine di Turing non-deterministiche

27 aprile 2023

MdT 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.**

MdT 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:*

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

MdT 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:*

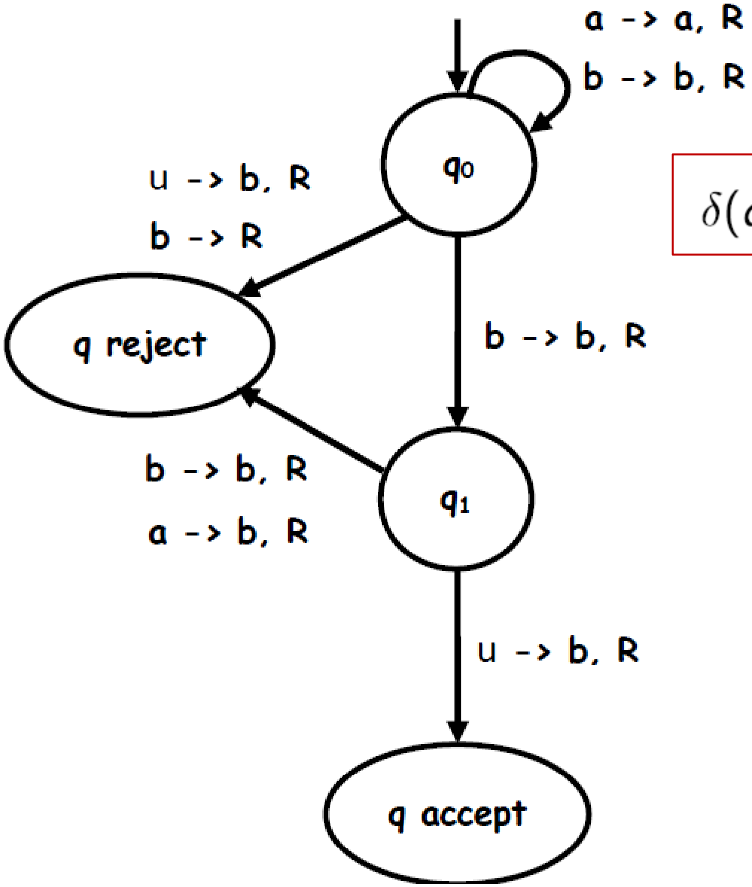
- $Q, \Sigma, \Gamma, q_0, q_{accept}, q_{reject}$ *sono definiti come in una MdT deterministica*
- *la funzione di transizione* δ *è definita al modo seguente:*

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

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

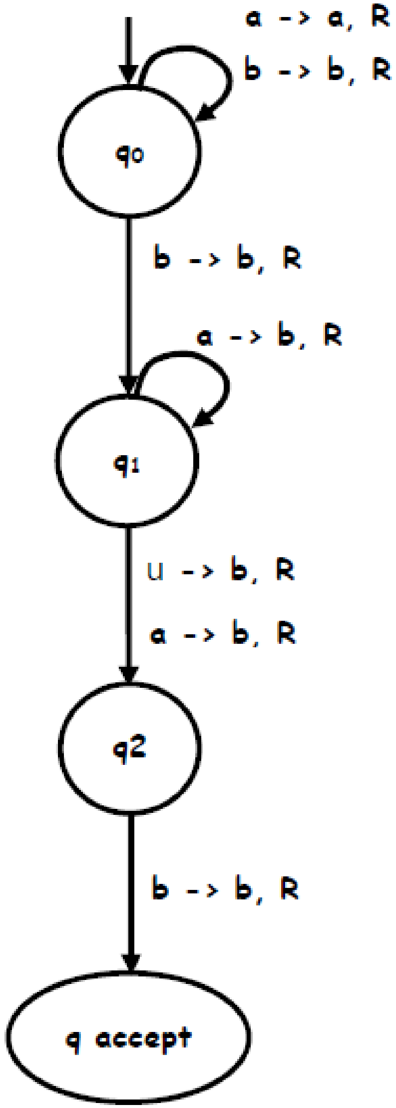
Nota: Se $\delta(q, a)$ è l'insieme vuoto, la macchina si muove nello stato q_{reject} e si ferma.

Esempio – M_1



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

Esempio – M_2



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

Computazioni

- Le configurazioni hanno la stessa forma uqv come per le MdT (deterministiche)
- Un passo di computazione $C_1 \rightarrow C_2$ e una computazione $C \rightarrow^* C'$ sono definite analogamente

- Invece, per MdT non-deterministiche, una configurazione uqv può produrre più configurazioni in un solo passo.
- Come per un NFA , le computazioni possono essere organizzate in un albero
- Una computazione è completamente determinata da una sequenza di scelte fra le varie configurazioni raggiungibili, cioè da un cammino nell’albero.

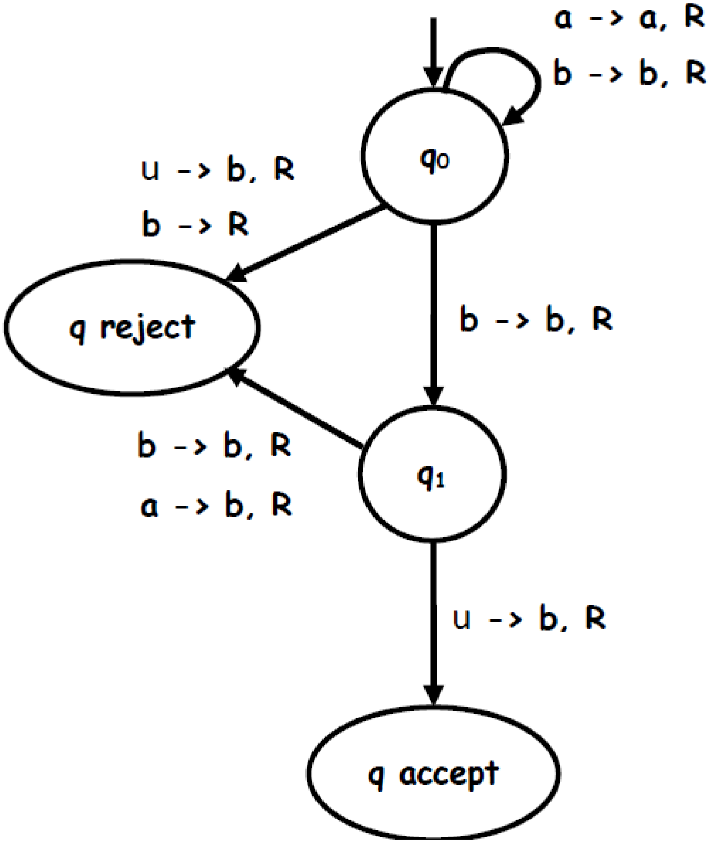
Macchina di Turing non deterministica

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)**.

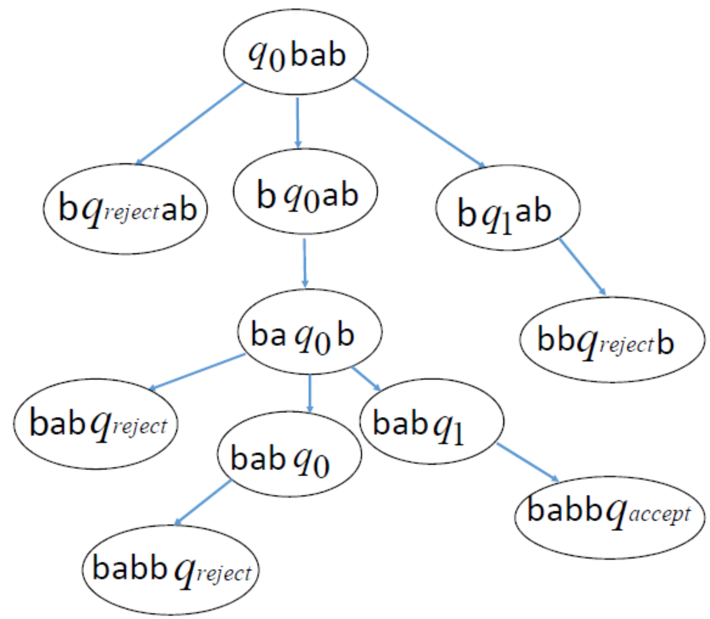
I nodi sono configurazioni e i figli di ogni nodo rappresentano le possibili configurazioni raggiungibili da quel nodo in un passo di computazione.

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

Esempio – M_1



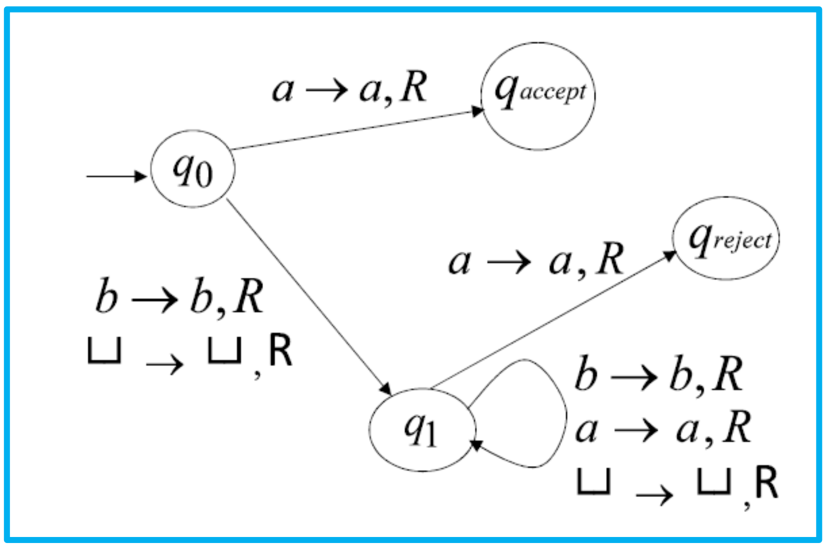
Albero delle computazioni per bab

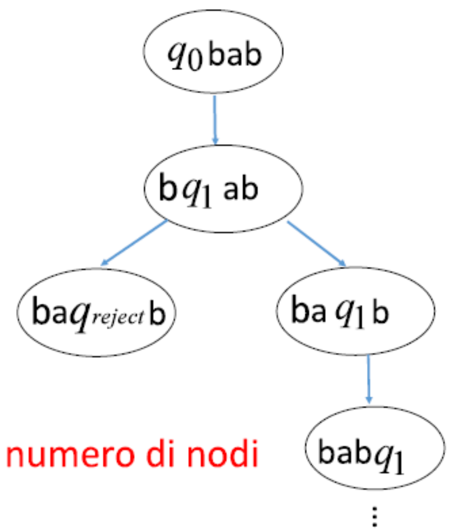


$$\delta(q_0, b) = \{(q_0, b, R), (q_1, b, R), (q_{reject}, b, R)\}$$

Computazioni che non terminano

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.





Albero con infinito numero di nodi

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

MdT non deterministica

Sia N una macchina di Turing non deterministica e sia w una stringa.

- N accetta w se e solo se esiste almeno una computazione $q_0 w \to^* u q_{accept} v$, 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

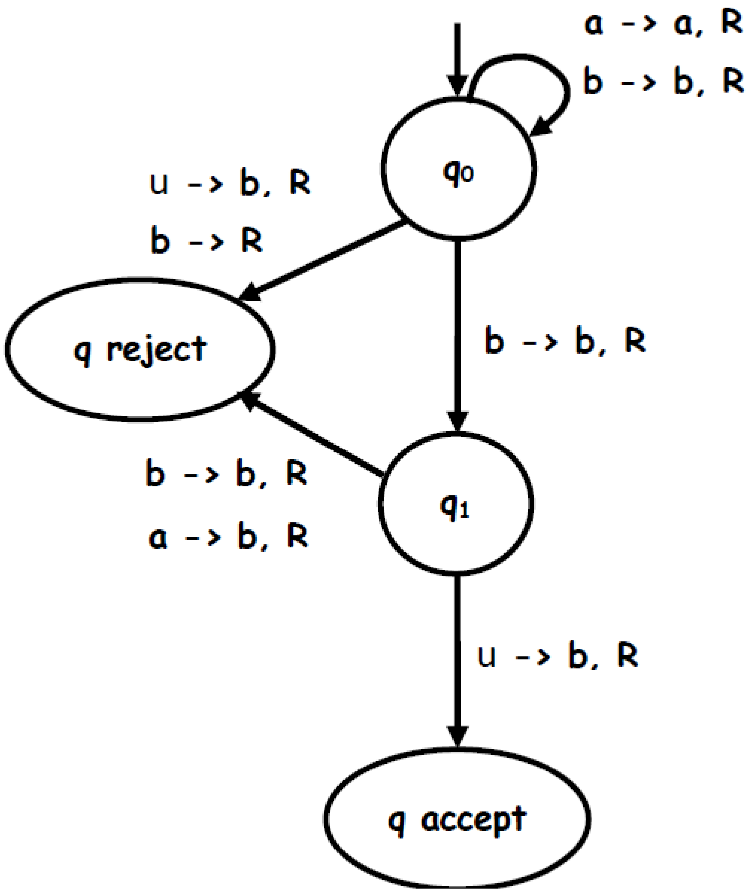
Definizione

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:*

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

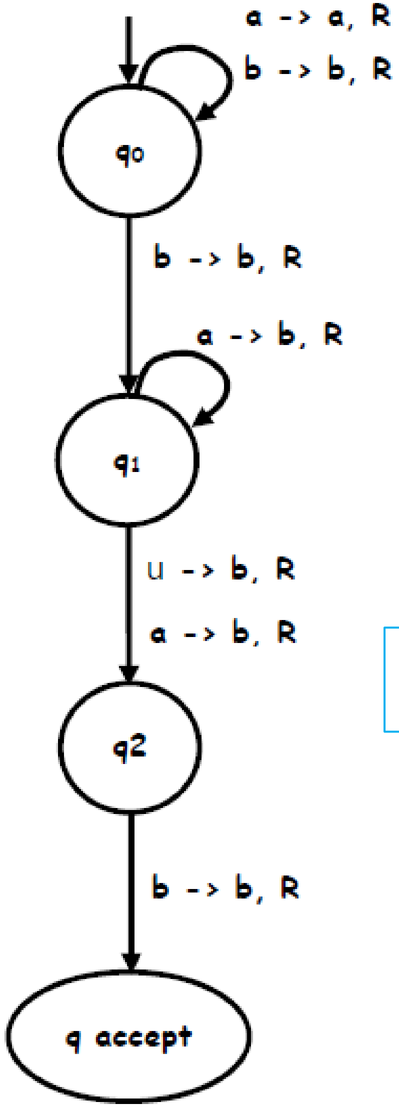
Nota: non considereremo funzioni calcolate

Esempio – M_1



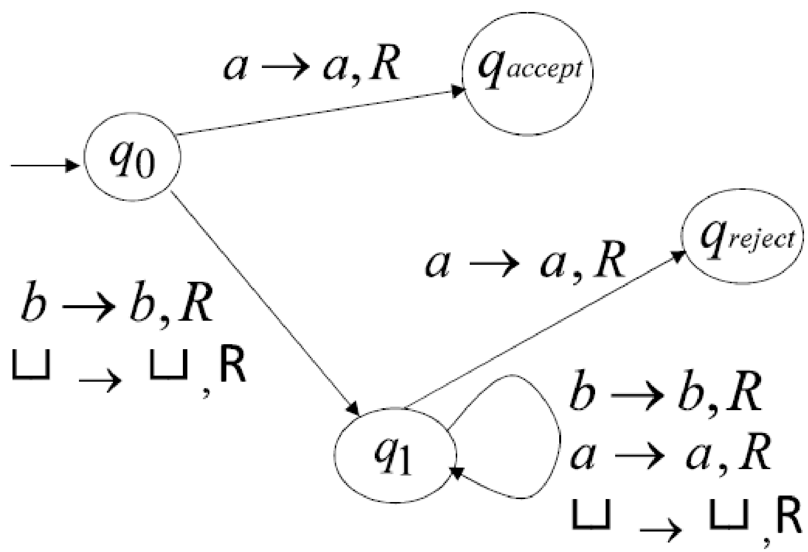
$$L(M_1) = \{wb \mid w \in \{a, b\}^*\}$$

Esempio – M_2



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

Esempio – N_1



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

Equivalenza

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 $L(D) = L(N)$.

- Una esplorazione dell'albero tramite visita BFS assicura che, se nell'albero c'è un nodo contenente una configurazione di accettazione, lo scoprirà in un numero finito di passi.

Idea della prova

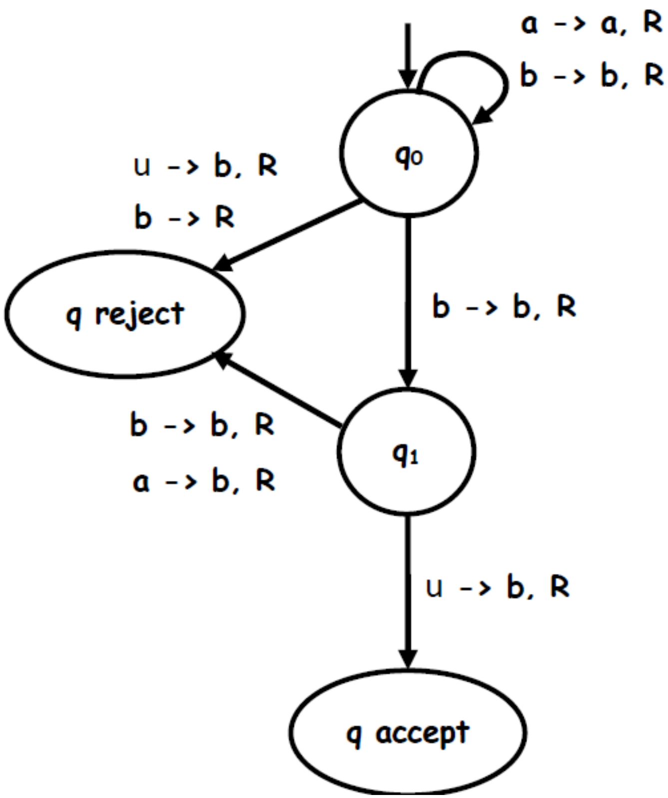
Vogliamo rappresentare le computazioni di N su una stringa w, cioè i cammini nell'albero delle computazioni di N su w mediante una stringa.

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 di *N* su ogni coppia stato-simbolo.
Ovvero B è il massimo numero di figli che può avere un nodo in un albero delle computazioni.

Denotiamo Γ_B = { 1, 2, … , B}.

Esempio – M_1



B = 3

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

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

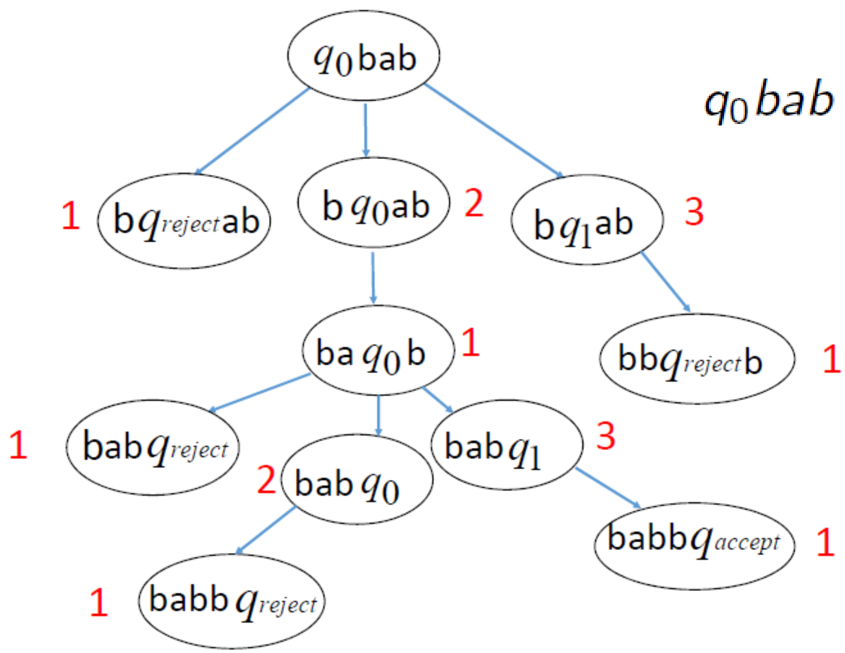
Esempio – M_1

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

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

La computazione che mostra che *bab* è accettata.

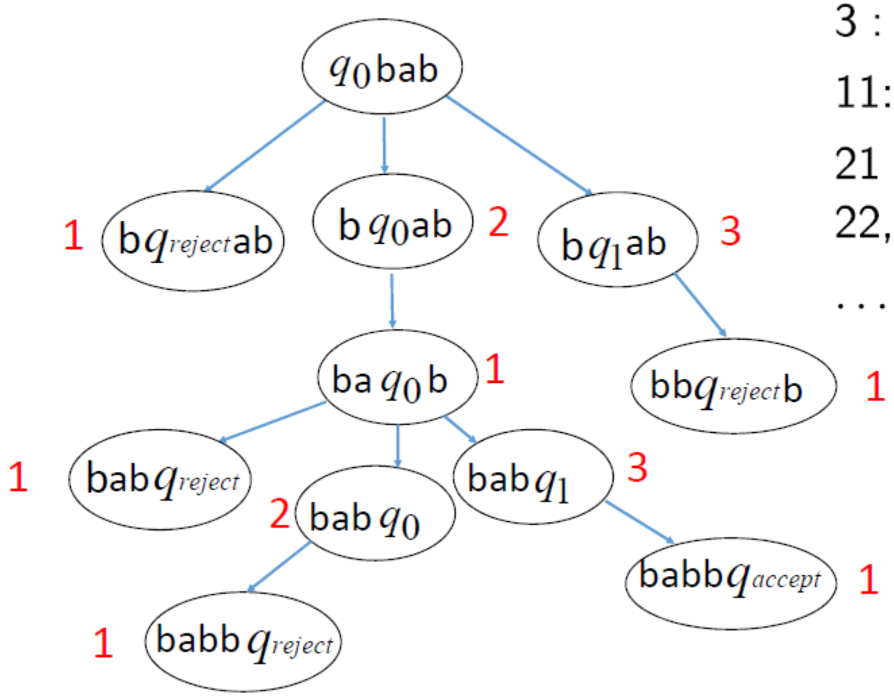
2131 :



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

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

Esempio – M_1



$1: \quad q_0 bab \rightarrow bq_{reject}ab$

$2: \quad q_0 bab \rightarrow bq_0 ab$

$3: \quad q_0 bab \rightarrow bq_1 ab$

11: senza sbocco, muore. Lo stesso per 12 e 13

$21: \quad q_0 bab \rightarrow bq_0 ab \rightarrow baq_0 b$

22, 23 senza sbocco, muore.

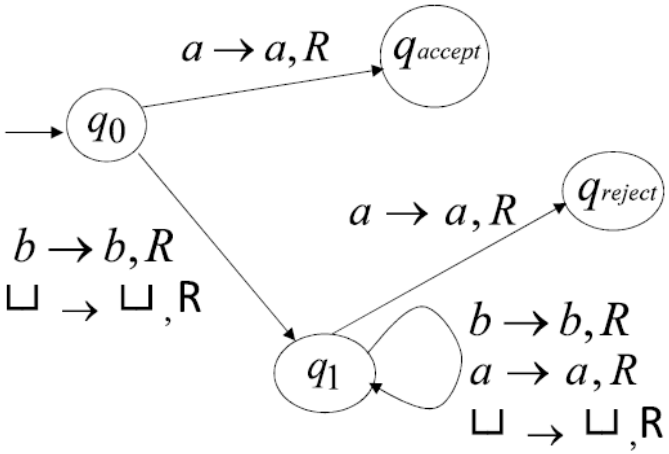
...

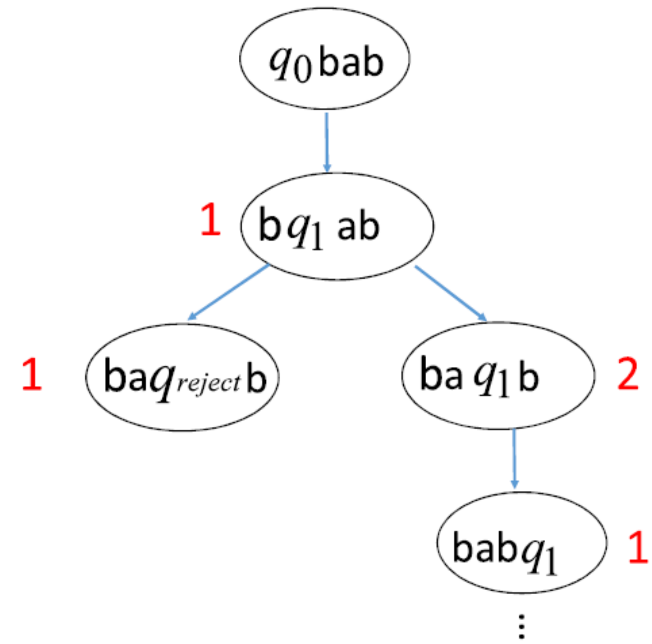
L'ordine in cui BFS scopre i nodi di un albero in cui ogni nodo ha B figli è l'ordine radix (o per lunghezza) sulle stringhe in {1, 2, … , B}.

Esempio B=3:

Epsilon, 1, 2, 3, 11, 12, 13, 21, 22, 23, 31, 32, 33, 111, 112, …….

Esempio – N_1





B = 2

$\Gamma_B = \{ 1, 2 \}$

Ordine computazioni: 1, 11, 12, 121, 1212, ...

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.

Esercizio

Descrivere una MdT che data una stringa w sull'alfabeto { 1, 2} calcoli la successiva di w nell'ordine radix.

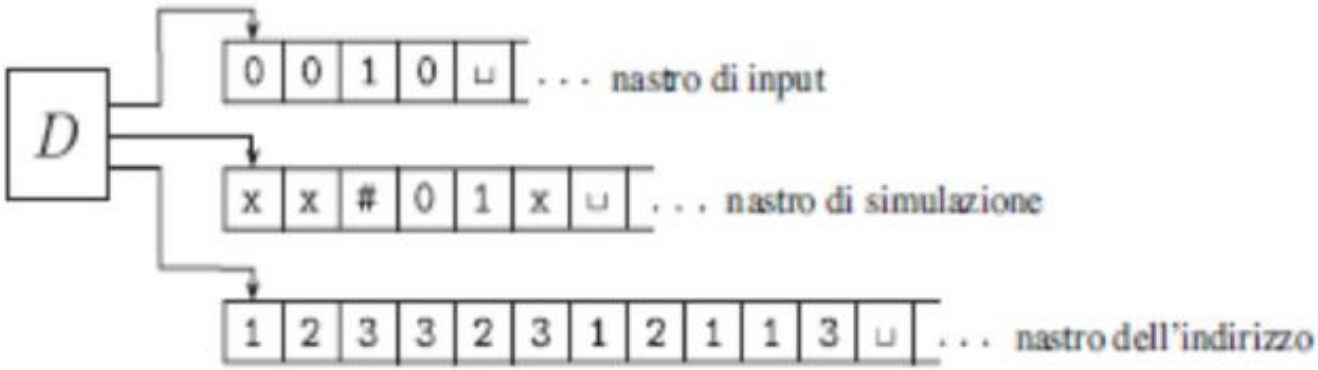
Ordine radix = ordine per lunghezza e a parità di lunghezza ordine numerico secondo 1 < 2.

Esempio

w = 1221 successore di w = 1222

w' = 222 successore di w' = 1111

Idea della prova



- 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.