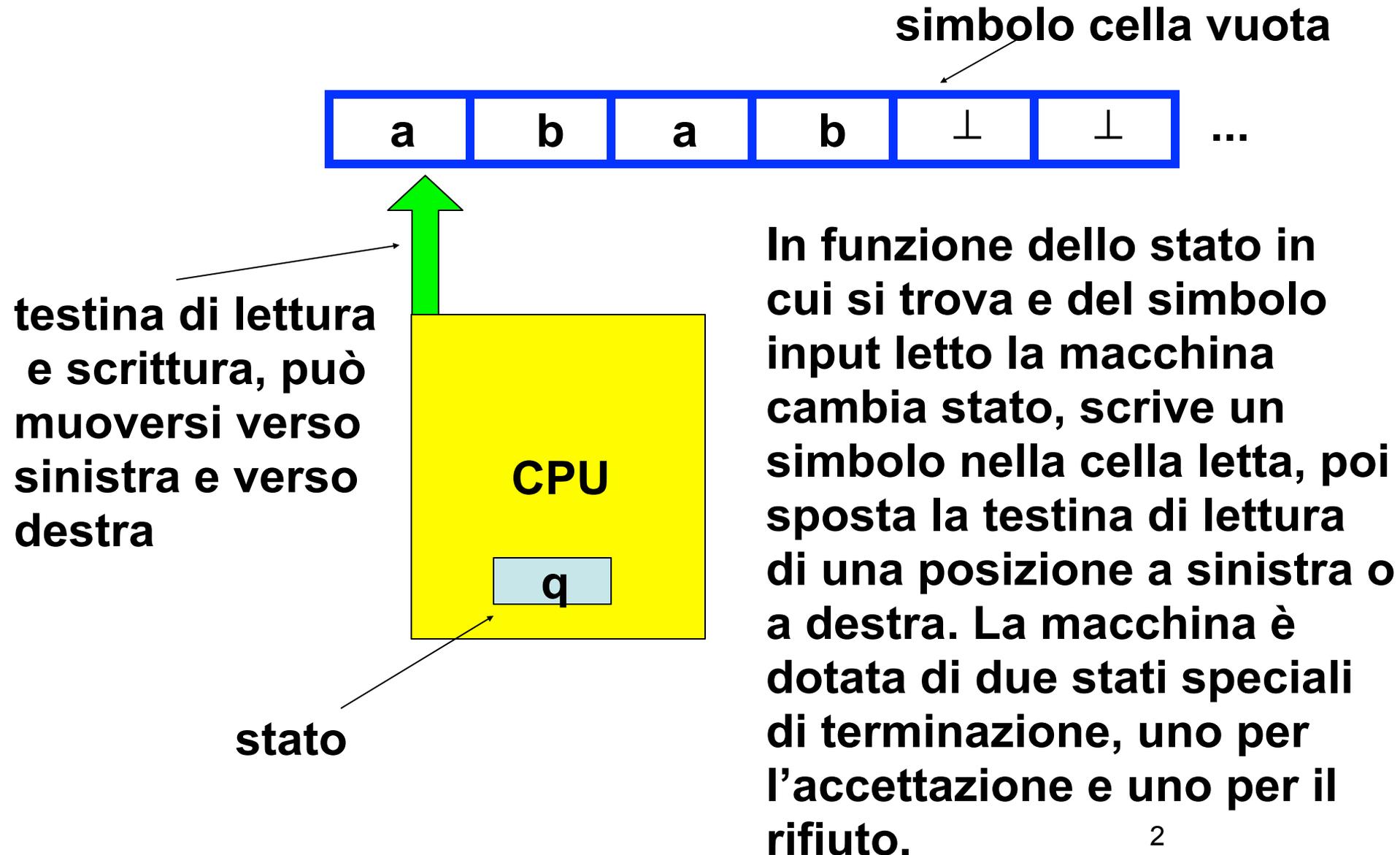


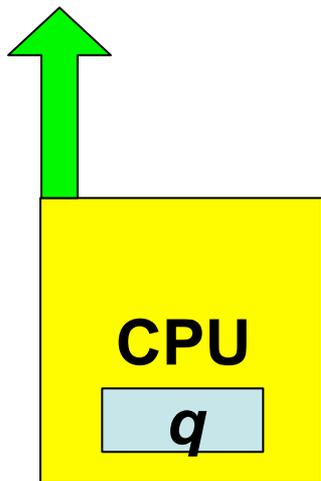
# Sommario

- **Definizione Macchine di Turing (deterministiche)**
- **esempi**
- **Tesi di Church-Turing**
- **Proprietà elementari delle macchine di Turing**

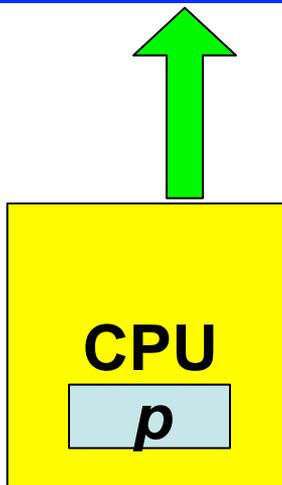
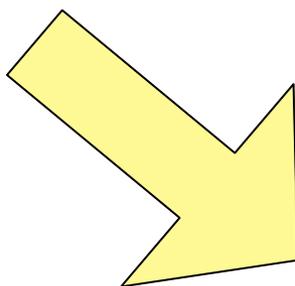
# Macchine di Turing: il modello mentale



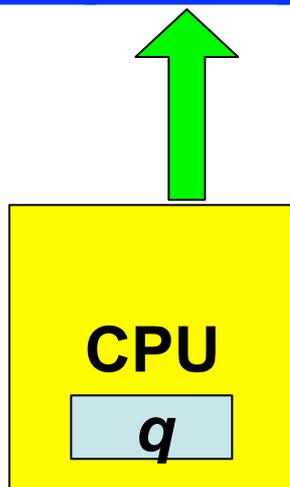
# Macchine di Turing: una mossa a destra



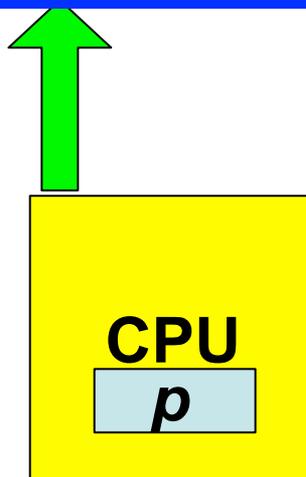
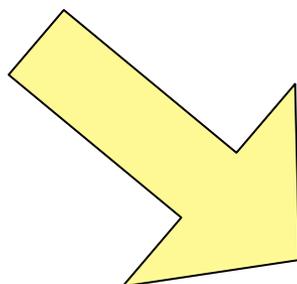
leggendo *a* nello stato *q*  
la TM passa nello stato *p*  
scrive *c* nella casella e  
sposta la testina di una  
posizione a destra



# Macchine di Turing: una mossa a sinistra



leggendo *b* nello stato *q*  
la TM passa nello stato *p*  
scrive *c* nella casella e  
sposta la testina di una  
posizione a sinistra



La macchina termina se  
entra in uno stato  
speciale di accettazione  
o in uno stato speciale di  
rifiuto, dai quali non ci  
sono mosse da eseguire.

# Esempio di TM 1

Una TM che accetta  $L = \{0^n 1^n \mid n > 0\}$

## L'idea:

**passo 1:** se si legge uno 0 lo si sostituisce con X, se si legge 1 si rifiuta, se si legge Y si va al 4.

**passo 2:** si scorre il nastro verso destra, se si trova un 1 lo si sostituisce con Y, altrimenti si rifiuta

**passo 3:** si scorre il nastro verso sinistra fino al primo X si torna a destra e si ripete dal passo 1

**passo 4:** si scorre a destra, se si leggono solo Y e il simbolo di cella vuota, allora si accetta altrimenti si rifiuta.

000111

X00111

X00Y11

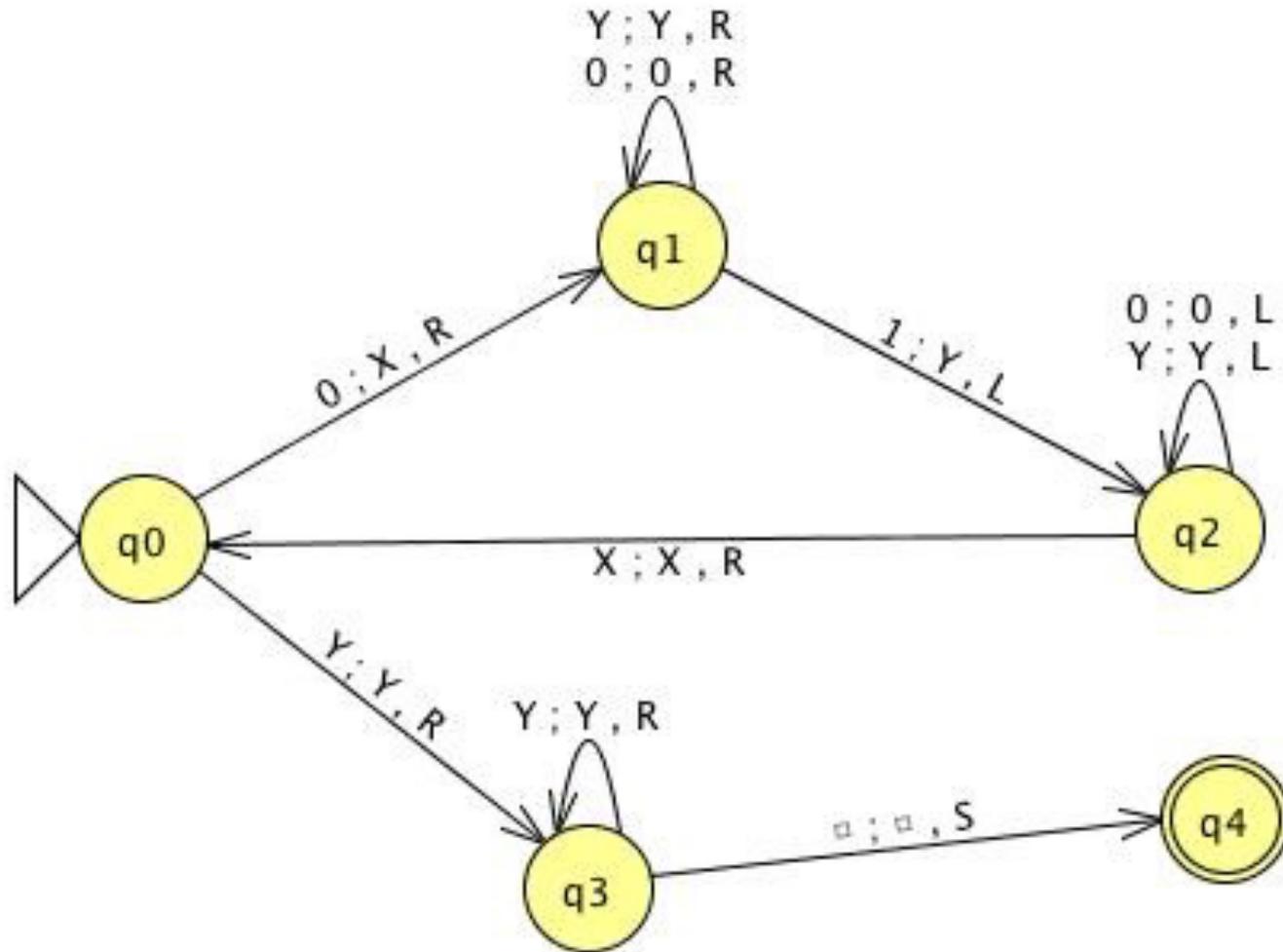
XX0Y11

XX0YY1

XXXYY1

XXXYYY

# La TM nel dettaglio



Ogni mossa non rappresentata porta a uno stato di rifiuto

La TM deve decidere  $L = \{0^{2^n} \mid n \geq 0\}$ .

L'idea si basa sull'osservazione che un numero è una potenza di 2 sse diviso per 2 dà ancora una potenza di due fino a ottenere 1.

00000000

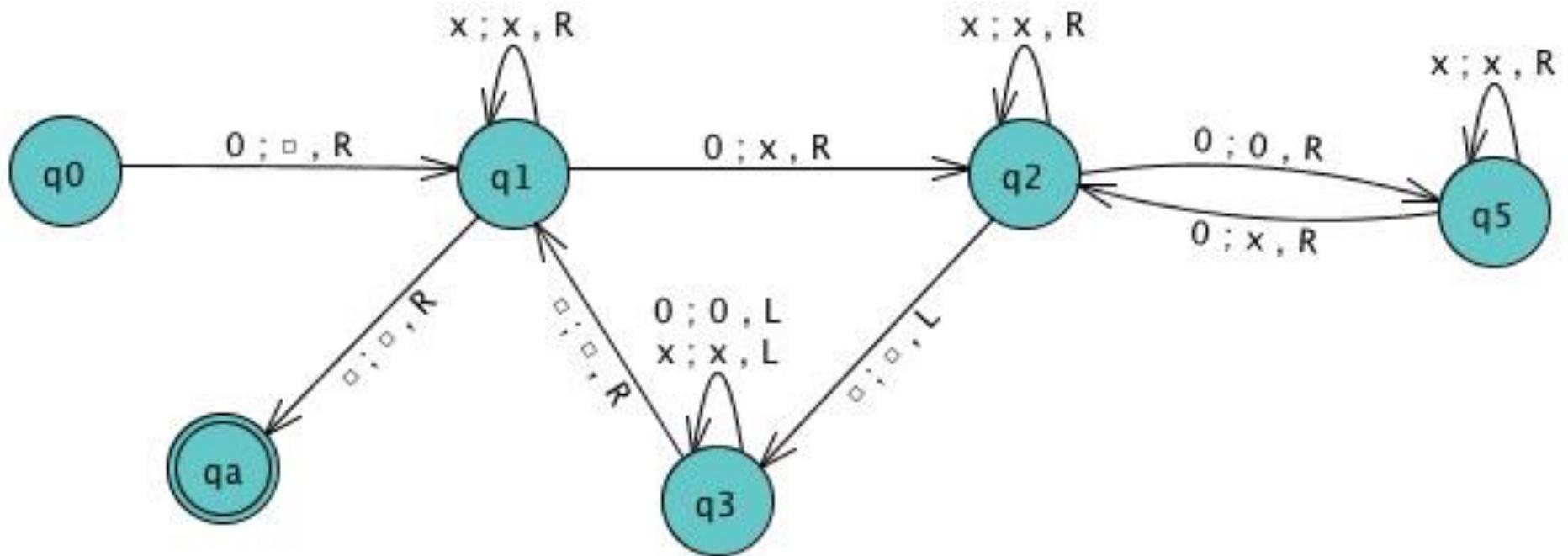
0X0X0X0X

passo 1: si marca il primo 0 e si scorre il nastro verso destra rimpiazzando uno 0 sì e uno no con X, facendo in modo che per ogni 0 che diventa X, ce ne sia uno che resta (così il numero è diviso a metà) e trascurando eventuali X incontrate, poi si ritorna all'inizio del nastro

0XXX0XXX

0XXXXXXXX

passo 2: se, dopo l'esecuzione una o più volte del passo 1 o fin dall'inizio scorrendo a destra si incontra un solo 0, allora si accetta.



**Qui lo 0 “marcato” all’inizio è rimpiazzato con un blank.  
 Ogni mossa non rappresentata porta a uno stato di rifiuto**

# Esempio di TM 3

La TM deve decidere  $L = \{w\#w \mid w \text{ è in } \{0,1\}^* \text{ e } |w| > 0\}$ .

**passo 1:** si marca come cella vuota il primo 0 o il primo 1, i passi successivi sono divisi in due casi, a seconda se si è letto uno 0 o un 1.

**010#010**

**B10#010**

## caso 0

**passo 2:** si scorre il nastro a destra fino a trovare #

**B10#X10**

**passo 3:** si scorre ancora a destra, trascurando eventuali simboli già marcati, se si trova uno 0 lo si marca a X, altrimenti si rifiuta

**BX0#X10**

**passo 4:** si torna all'inizio del nastro,

**BX0#XX0**

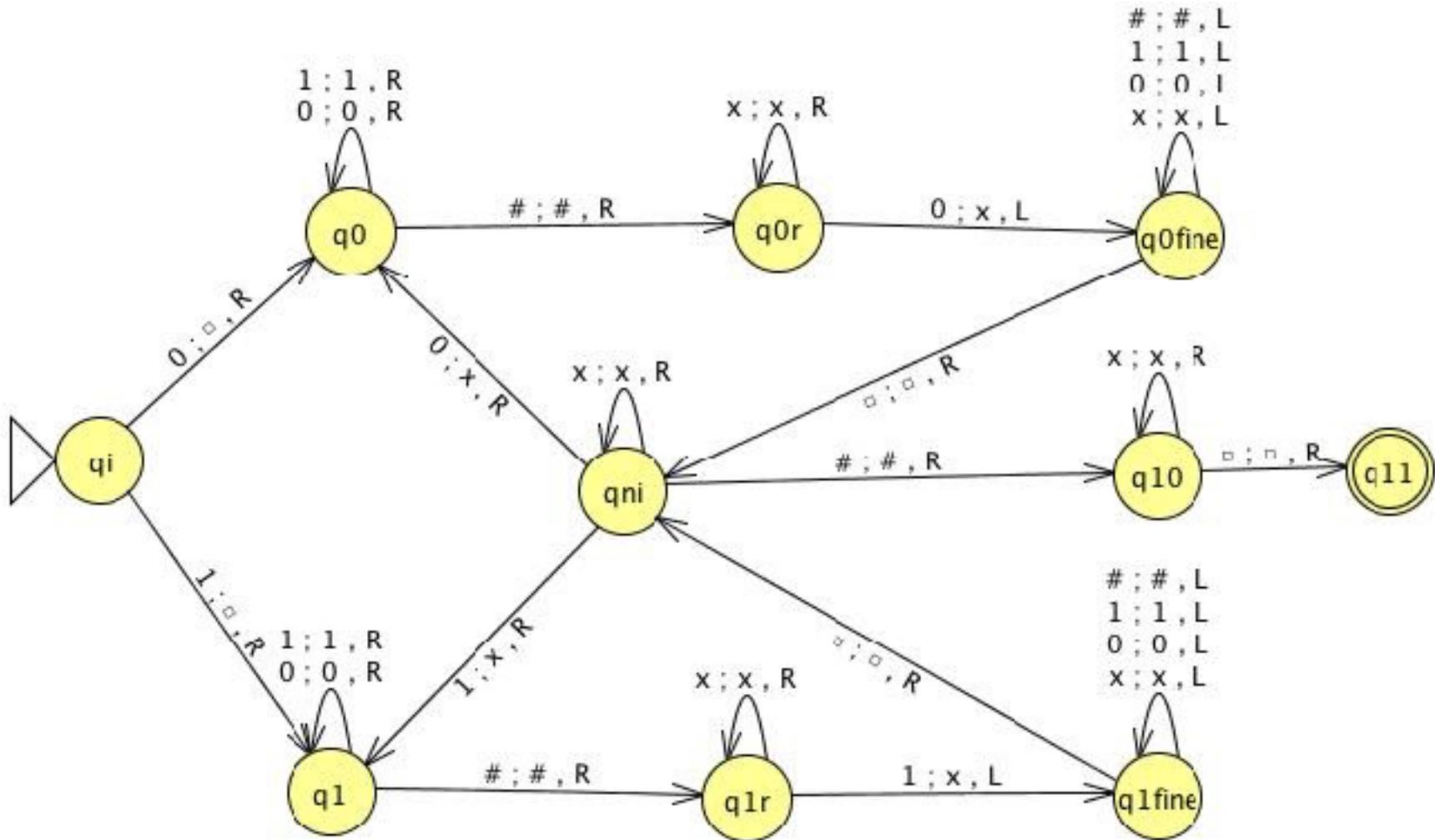
**passo 5:** si ritorna verso destra, trascurando eventuali simboli già marcati, e se si trova uno 0 lo si marca come X e si torna al passo 2, se si trova un 1 si va al passo 2 del caso 1, altrimenti si controlla se anche dopo l'occorrenza di # ci sono solo X e in tal caso si accetta.

**BXX#XXX**

## caso 1

identico, scambiando 0 in 1

# La TM nel dettaglio



Ogni mossa non rappresentata porta a uno stato di rifiuto

# Modello formale

Una **Machina di Turing** (Turing Machine) **deterministica**, è una settupla  $M = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$  dove

- $Q, \Sigma$  e  $\Gamma \supset \Sigma$  sono insiemi finiti, rispettivamente degli stati, dei simboli di input e dell'alfabeto di nastro; lo speciale simbolo di cella vuota  $\perp$  è in  $\Gamma$ , ma non in  $\Sigma$
- $\delta : Q - \{q_a, q_r\} \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$  è la funzione di transizione;
- $q_0$  è lo stato iniziale;
- $q_a$  e  $q_r$  sono rispettivamente lo stato di accettazione e quello di rifiuto, con  $q_a \neq q_r$

# Configurazioni

Una configurazione deve informare sul contenuto del nastro, lo stato della macchina e la posizione della testina di lettura.

Queste informazioni si possono ottenere sinteticamente da una sequenza del tipo

$$\alpha qa\beta \text{ in } \Gamma^*Q\Gamma^*$$

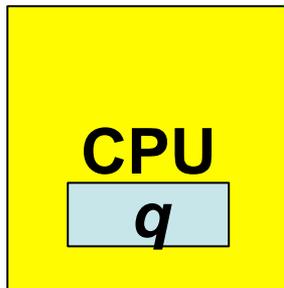
dove  $\alpha$  e  $\beta$  sono stringhe su  $\Gamma$ ,  $a$  è un simbolo di  $\Gamma$ ,  $\alpha a\beta$  è il contenuto del nastro,  $a$  è il simbolo in lettura e  $q$  è lo stato della macchina.

La sequenza

$$q_0x$$

dove  $x$  è una stringa input, e  $q_0$  è lo stato iniziale è detta configurazione iniziale.

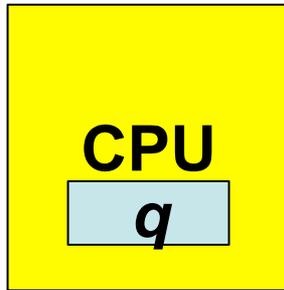
# Configurazioni esempio 1



La TM si trova nella configurazione

*abqabb*

## Configurazioni esempio 2



La TM si trova nella configurazione  
*ababbq*

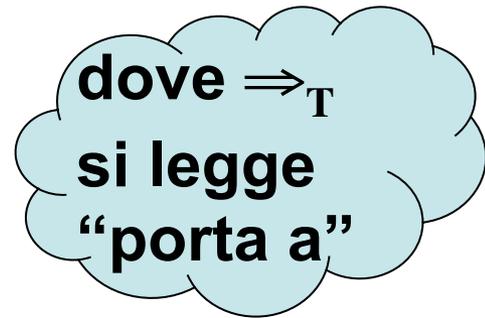
Una configurazione  $\alpha aq$  è *equivalente a*  $\alpha aq\perp$ , cioè quella in cui la testina è posizionata su una cella vuota.

## Mosse

a destra:

se  $\delta(q, a) = (p, b, R)$  allora

$$\alpha q a c \beta \Rightarrow_T \alpha b p c \beta$$



a sinistra:

se  $\delta(q, c) = (p, b, L)$  allora

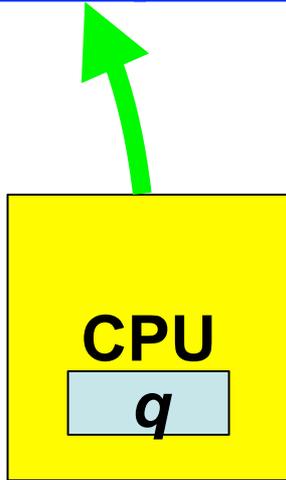
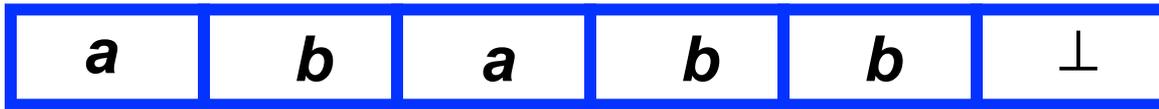
$$\alpha a q c \beta \Rightarrow_T \alpha p a b \beta$$

Una configurazione del tipo  $\alpha q_a \beta$  è detta di accettazione.

Una configurazione del tipo  $\alpha q_r \beta$  è detta di rifiuto.

Queste configurazioni sono di terminazione

# Configurazioni e mosse



La TM si trova nella configurazione

$qababb$

Se  $\delta(q,a) = (p,b,L)$  allora  $qababb \Rightarrow_T pbbabb$

Se la macchina si trova in una configurazione del tipo  $qa\beta$ , cioè ha la testina di lettura sulla prima cella del nastro, e se la mossa prevista in questo caso dovesse comportare uno spostamento a sinistra della testina allora per convenzione la testina non si muove.

## Linguaggio accettato

Sia  $M = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$ , e sia  $C(x)$  l'insieme delle configurazioni raggiungibili da quella iniziale  $c_0 = q_0 x$ .

Il linguaggio **accettato** (riconosciuto) è

$$\begin{aligned} L(M) &= \{x \mid x \in \Sigma^* \text{ e } \exists c \in C(x) \text{ e } c \text{ è di accettazione}\} \\ &= \{x \mid x \in \Sigma^* \text{ e } q_0 x \Rightarrow^* \alpha q_a \beta \text{ con } \alpha, \beta \in \Gamma^*\} \end{aligned}$$

Il linguaggio **rifiutato** è

$$\begin{aligned} R(M) &= \{x \mid x \in \Sigma^* \text{ e } \exists c \in C(x) \text{ e } c \text{ è di rifiuto}\} \\ &= \{x \mid x \in \Sigma^* \text{ e } q_0 x \Rightarrow^* \alpha q_r \beta \text{ con } \alpha, \beta \in \Gamma^*\} \end{aligned}$$

**In generale**  $L(M) \cup R(M) \subseteq \Sigma^*$ !

Se  $L(M) \cup R(M) = \Sigma^*$  allora vuol dire che la TM si ferma sempre, in tal caso  $L(M)$  è il linguaggio **deciso** dalla TM.

# Linguaggio accettato

## In generale

il linguaggio accettato,  $L(M)$ , da una TM è detto **Turing-riconoscibile** o semplicemente **riconoscibile** (ricorsivamente enumerabile, semidecidibile, in questo caso  $L(M) \cup R(M) \subseteq \Sigma^*$ )

## Se

la TM si ferma sempre  $L(M)$  è il linguaggio **deciso** dalla TM ed è detto **Turing-decidibile** o semplicemente **decidibile** (in questo caso  $L(M) \cup R(M) = \Sigma^*$ )

## Classe dei linguaggi accettati

**L'insieme di tutti i linguaggi che sono riconosciuti da una TM è così definito:**

$$\mathcal{L}(TM) = \{L \mid \exists M \in TM \text{ e } L(M) = L\}$$

# Tesi di Church - Turing

## Tesi di Church - Turing:

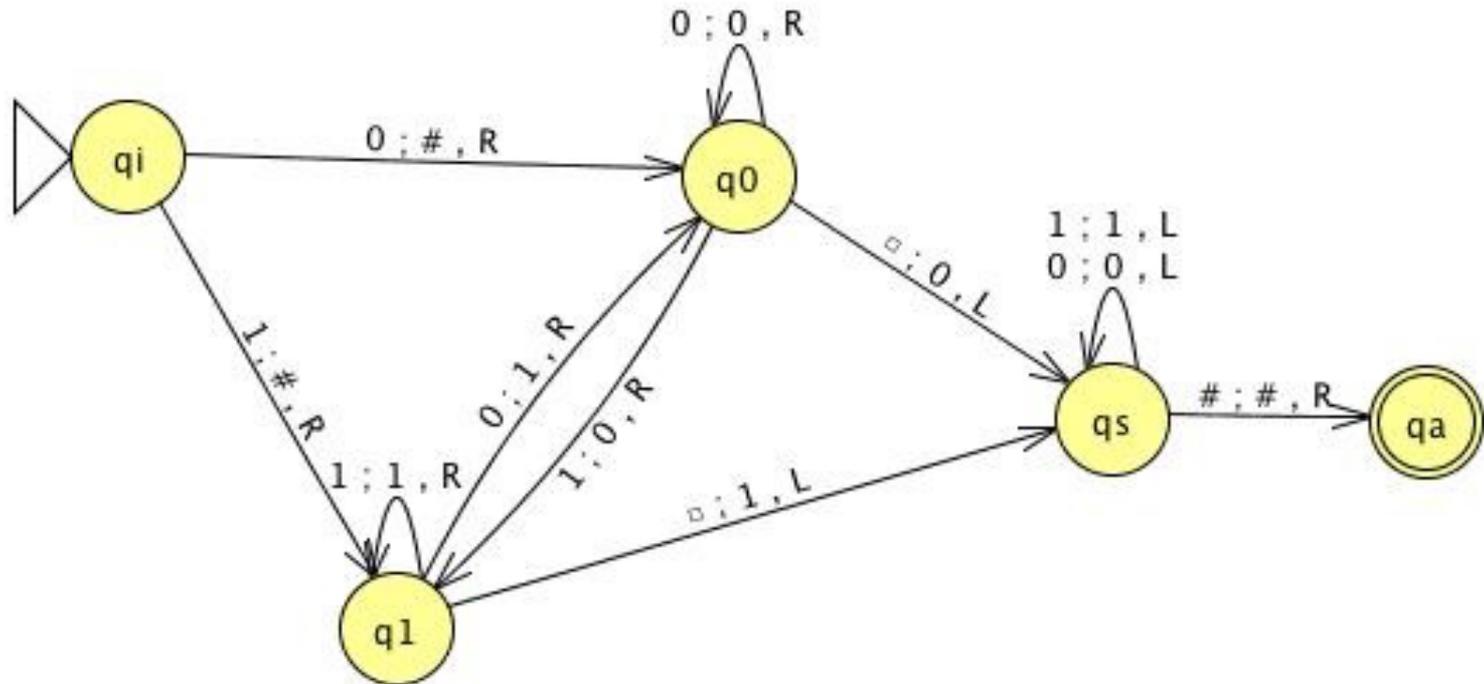
La classe delle funzioni calcolate da una **TM** è la classe delle funzioni che informalmente sono considerate effettivamente calcolabili, o equivalentemente la classe dei linguaggi riconosciuti da una **TM** corrisponde alla classe dei problemi intuitivamente effettivamente risolvibili.

Quindi possiamo identificare l'idea di **algoritmo** con quella di una **TM che si ferma sempre**.

Mentre un **semialgoritmo** corrisponde a una **TM che non sempre si ferma**.

# Esempio di modulo TM

Una TM che trasforma una stringa binaria  $x$  nella stringa  $\#x$ , riposizionando la testina di lettura sulla prima cella di nastro



# Esempio: Una TM che genera la successiva stringa binaria in ordine canonico

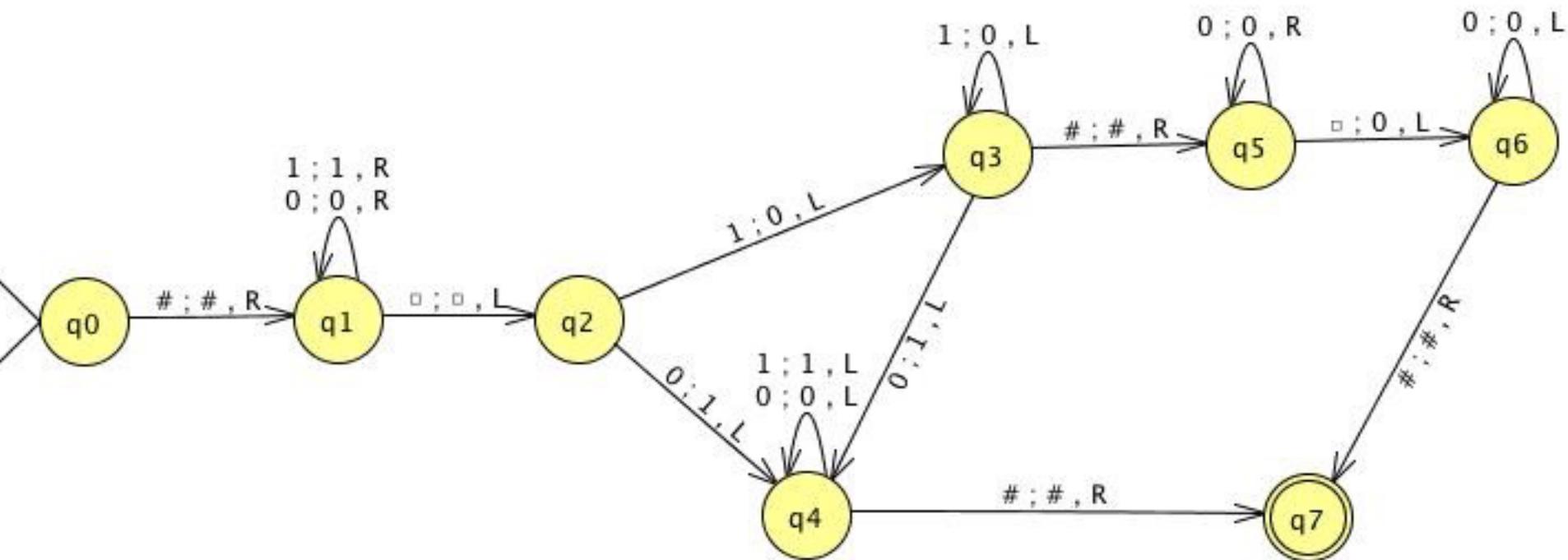
Le stringhe binarie elencate in ordine canonico :

0 1 00 01 10 11 000 001 010 011 100 101 110 111 ...

1. Data una sequenza  $x01\dots1$ , con  $x$  in  $\{0,1\}^*$ , la successiva si ottiene trasformando tutti gli 1 finali in 0 e lo 0 in 1, ottenendo  $x10\dots0$
2. Data una sequenza  $x0$ , con  $x$  in  $\{0,1\}^*$ , la successiva si ottiene trasformando lo 0 in 1, ottenendo  $x1$
3. data una sequenza  $1\dots1$  la successiva si ottiene trasformando ogni 1 in 0 e concatenando un altro 0

Ne diamo una descrizione dettagliata come file JFLAP, supponendo di avere utilizzato il modulo di TM che ha inserito il marcatore di fine nastro nella prima cella.

**Esempio: Una TM che dà in output la stringa binaria successiva in ordine canonico a quella data in input**



**Esempio: Una TM che genera in ordine canonico tutte le stringhe binarie quando inizializzata con la stringa 0.**

