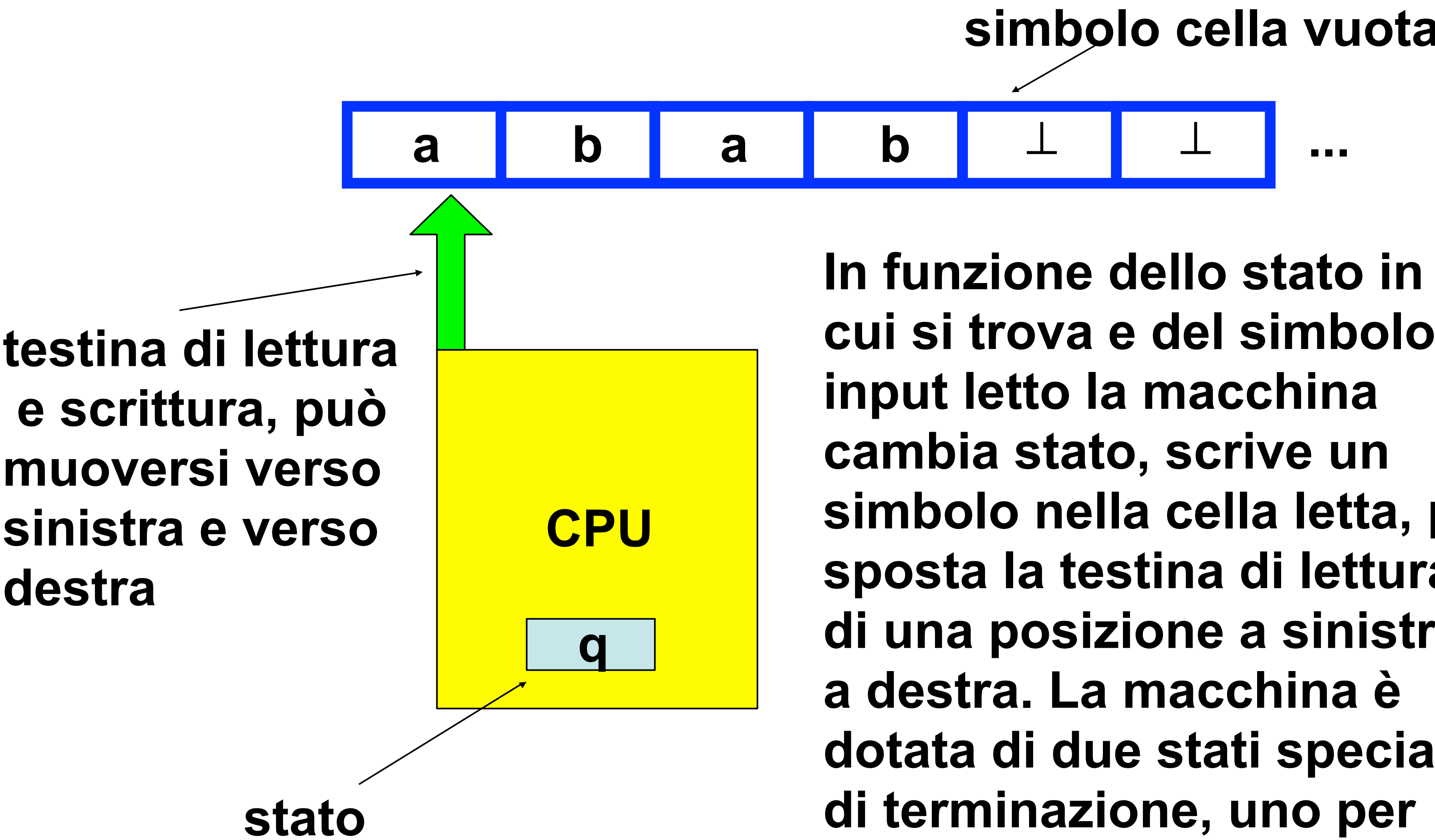
# Sommario

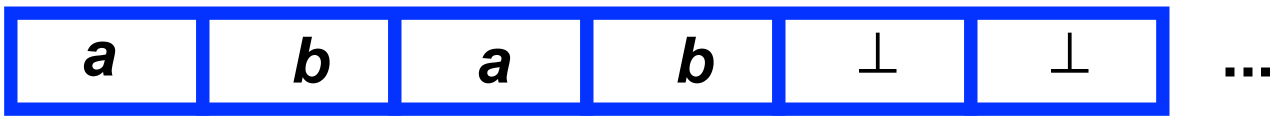
- Definizione Macchine di Turing, TM
- esempi
- Tesi di Church-Turing
- Proprietà elementari delle TM

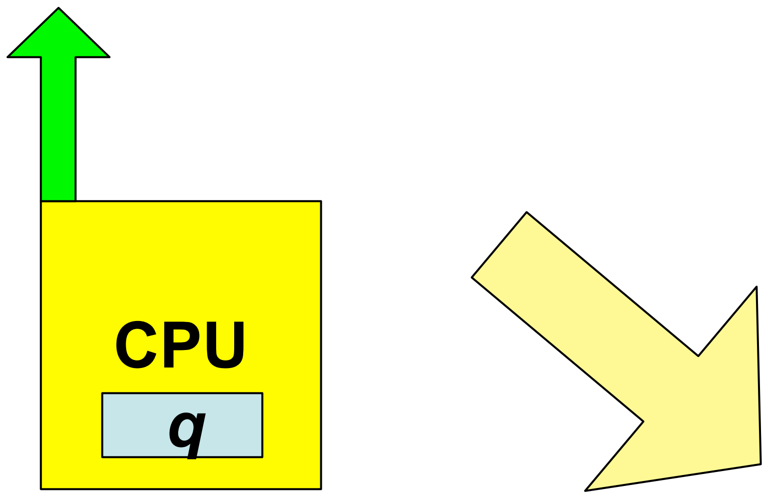
# Macchine di Turing: il modello mentale



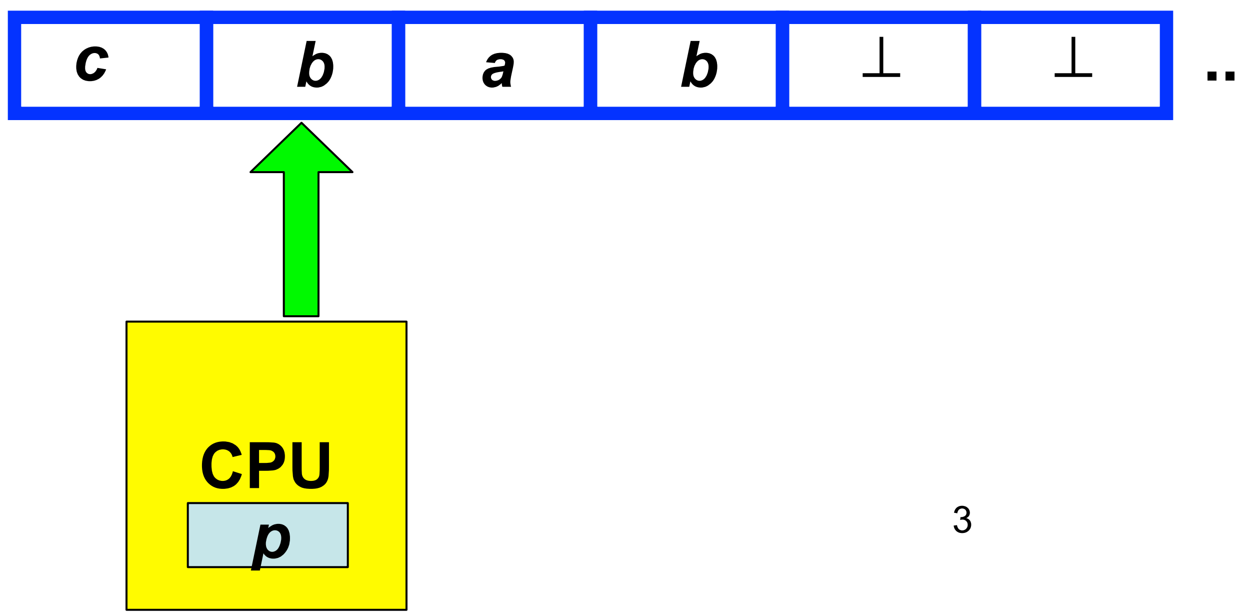

In funzione dello stato in cui si trova e del simbolo input letto la macchina cambia stato, scrive un simbolo nella cella letta, poi sposta la testina di lettura di una posizione a sinistra o a destra. La macchina è dotata di due stati speciali di terminazione, uno per l'accettazione e uno per il rifiuto.

# Macchine di Turing: una mossa a destra

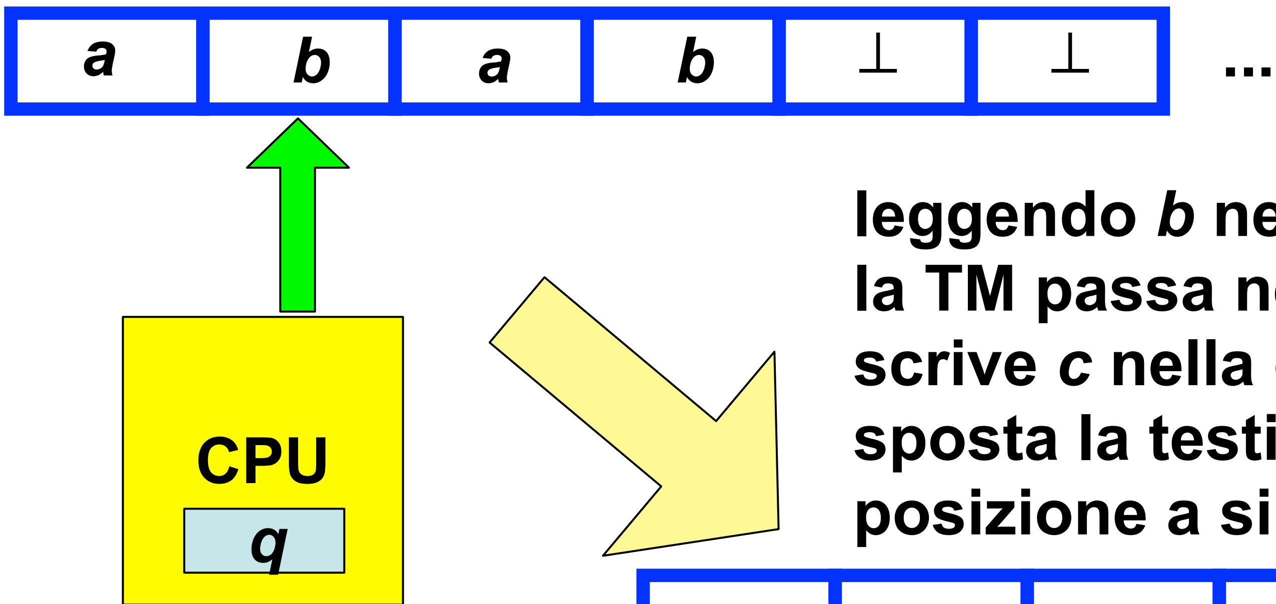


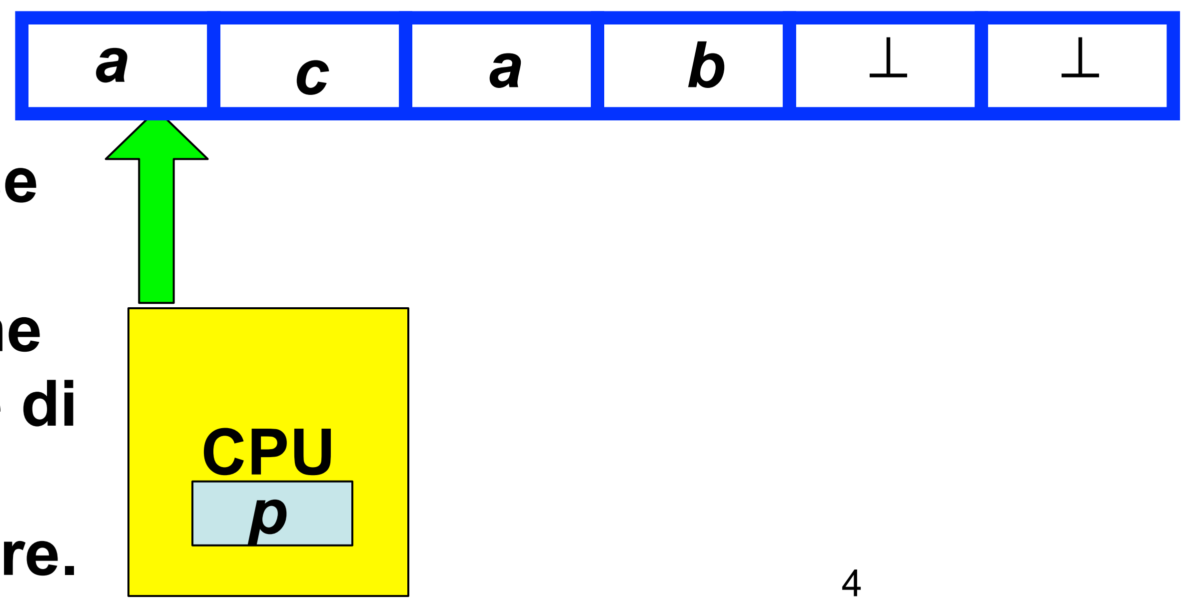

**leggendo *a* nello stato *q* la TM passa nello stato *p* scrive *c* nella casella e spostala testina di una posizione adestra**



# 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 $L = \{a^n b^n \mid n > 0\}$**

## L'idea:

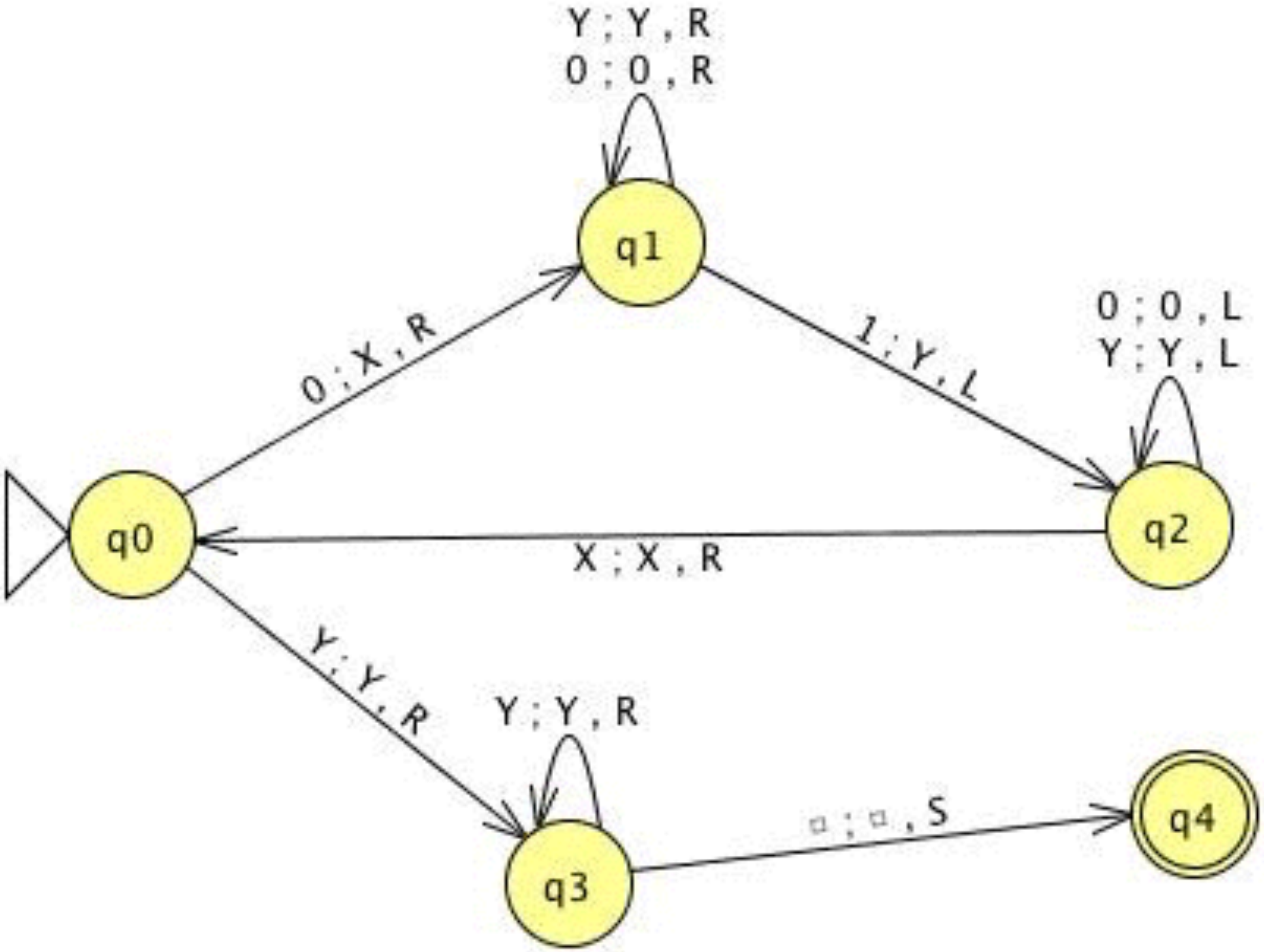
**passo 1: se si legge uno 0 lo si sostituisce con X altrimenti si va al 4.**

**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 non si leggono 1, ma solo Y, allora si accetta altrimenti si rifiuta.**

# La TM nel dettaglio




**Ogni mossa non rappresentata porta a uno stato di rifiuto**

# Esempio di TM 2

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

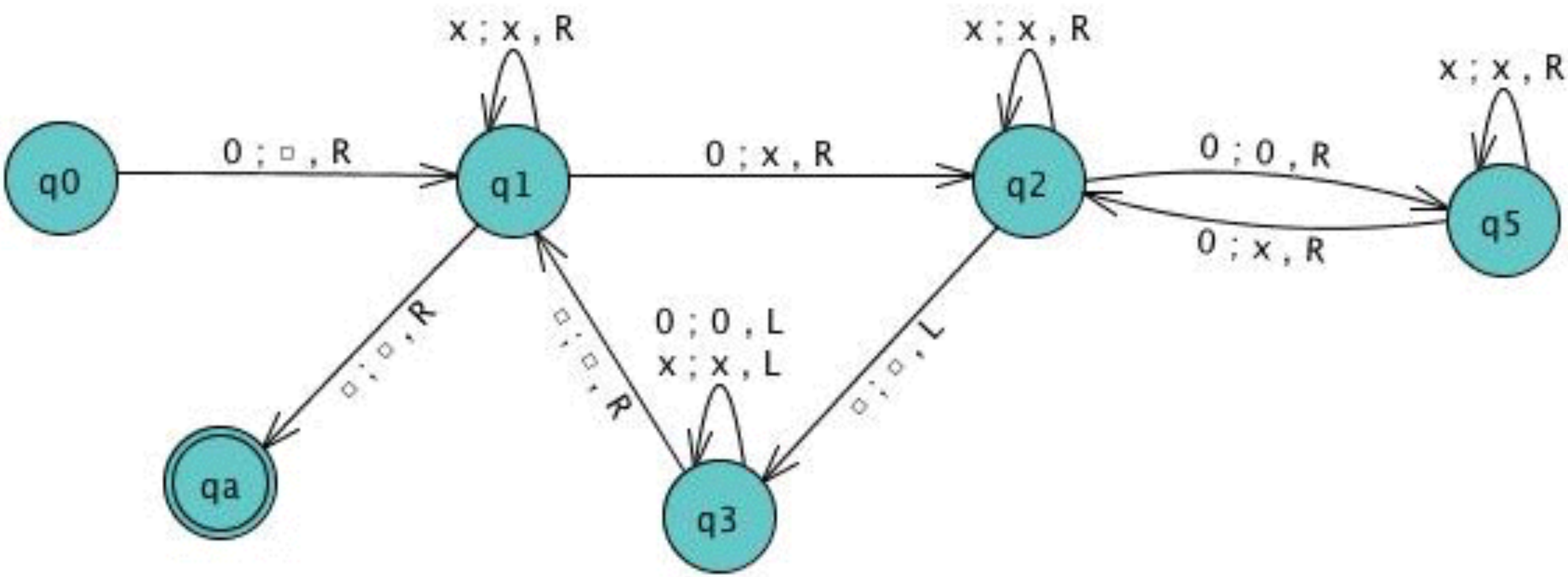
L' idea implementa l'osservazione che un numero è una potenza di 2 sse ripetutamente diviso per due dà risultato pari fino all'ultima divisione che dà 1.

passo 1: si marca la fine sinistra del nastro e si scorre il nastro verso destra rimpiazzando uno 0 sì e uno no con X, (così il numero è diviso a metà)
trascurando eventuali X incontrate;

passo 2: se al passo 1 si era incontrato un solo 0 allora si accetta, altrimenti se è dispari si rifiuta

passo 4: si torna all'inizio del nastro e si ripete dal passo 1.

# La TM nel dettaglio



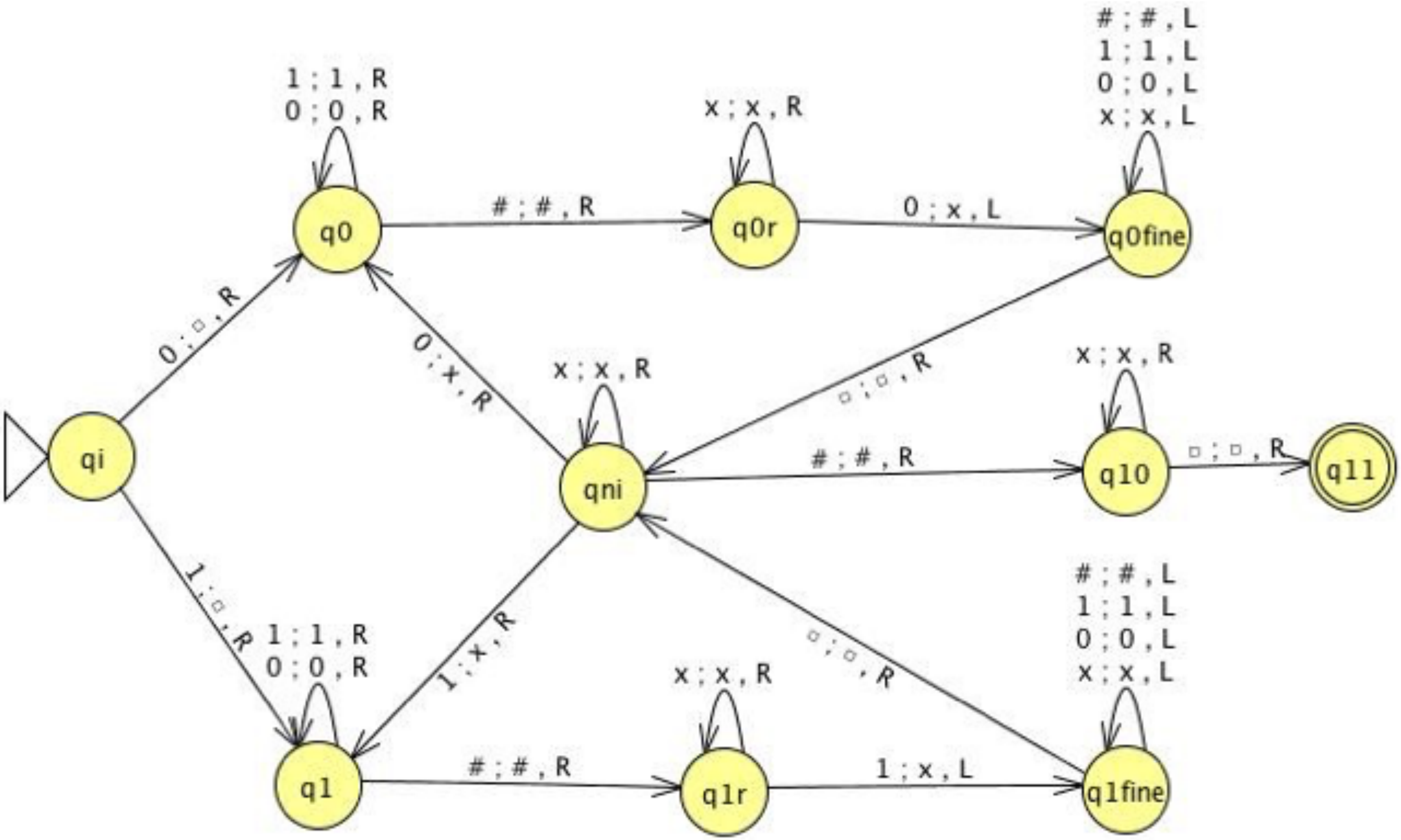

**Ogni mossa non rappresentata porta a uno stato di rifiuto**

# Esempio di TM 3

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

**passo 1: si marca come cella vuota il primo 0 o il primo 1, proseguendo su due strade identiche tranne per il simbolo da controllare (per riconoscere l'inizio del nastro).**
**caso 0**
**passo 2: si scorre il nastro a destra fino a trovare #**
**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**
**passo 4: si torna all'inizio del nastro,**
**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 la passo 2 del caso 1, altrimenti si controlla se anche dopo l'occorrenza di # ci sono solo X e in tal caso si accetta.**
**caso 1**
**identico, cambiando 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 <u>non</u> 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**

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

**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**

$$q_0 x$$

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

# Mosse

dove $\Rightarrow_T$
si legge
“porta a”

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

$$\alpha qac\beta \Rightarrow_T \alpha bpc\beta$$

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

$$\alpha aqc\beta \Rightarrow_T \alpha pab\beta$$

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

# 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**

$$L(M) = \{x \mid x \in \Sigma^* \text{ e } \exists\, c \in C(x) \text{ e c è di accettazione}\}$$
$$= \{x \mid x \in \Sigma^* \text{ e } q_0x \Rightarrow^* \alpha q_a\beta \text{ con } \alpha,\beta \in\Gamma^*\}$$

**Il linguaggio rifiutato è**

$$R(M) = \{x \mid x \in \Sigma^* \text{ e } \exists\, c \in C(x) \text{ e c è di rifiuto}\}$$
$$= \{x \mid x \in \Sigma^* \text{ e } q_0x \Rightarrow^* \alpha q_r\beta \text{ con } \alpha,\beta \in\Gamma^*\}$$

**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**

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

**e il linguaggio accettato, L(M), da una TM è detto Turing-riconoscibile o semplicemente riconoscibile (ricorsivamente enumerabile, semidecidibile)**

**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 TMed è detto Turing-decidibile o semplicemente decidibile**

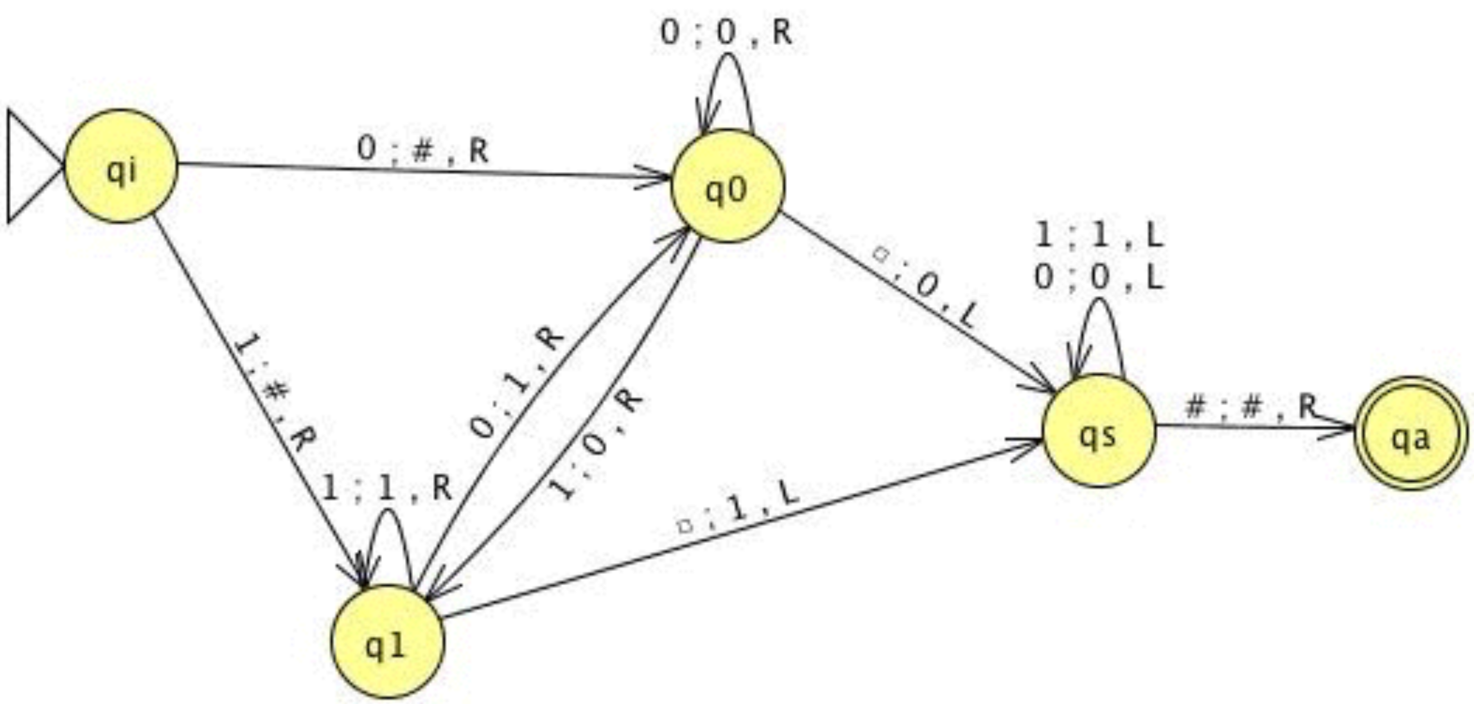
# Classe dei linguaggi accettati

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

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

# 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

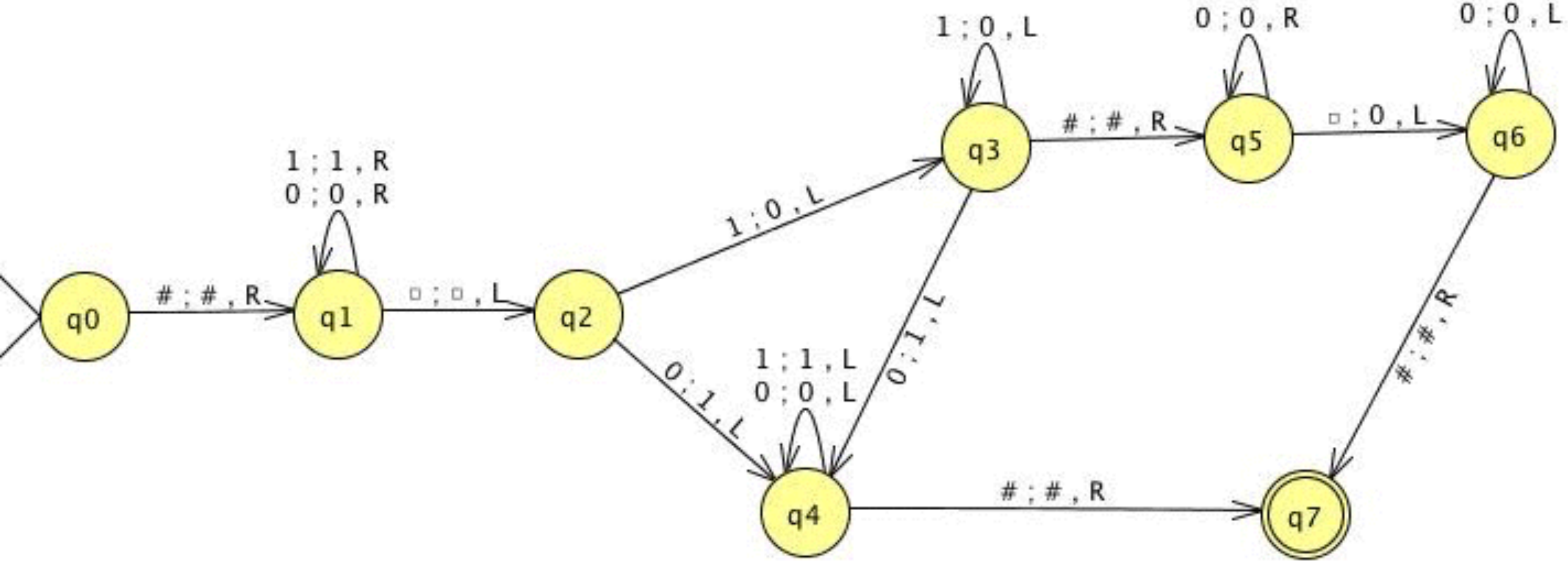
**Le stringhe binarie elencate in ordine canonico :**

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

1. **Data una sequenza x0y, con x in {0,1}* e y in 1*, la successiva si ottiene trasformando tutti gli 1 finali in 0 e lo 0 in 1, ottenendo x10...0**
2. **data una sequenza 1...1 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 genera la successiva stringa binaria in ordine canonico



# 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.**

# Semialgoritmo per l'inequivalenza per CFG

## Problema: due CFG sono inequivalenti?

**input: due CFG G1 e G2**

1. **verifica se la parola vuota è generata da entrambe, se no accetta altrimenti vai al passo 2**
2. **trasforma G1 e G2 in equivalenti ( a meno della parola vuota) CFG in forma normale di Chomsky**
3. **inizializza il nastro con la prima parola non vuota**
4. **utilizza l'algoritmo CYK per controllare se la stringa sul nastro è generata da G1 e non da G2 o viceversa**

**se SI fermati e accetta altrimenti**

**genera la stringa successiva (in ordine quasilessicografico)**

**a quella sul nastro di input e torna al punto 4.**

**É evidente che la procedura termina se G1 e G2 non sono equivalenti, mentre non si ferma altrimenti.**