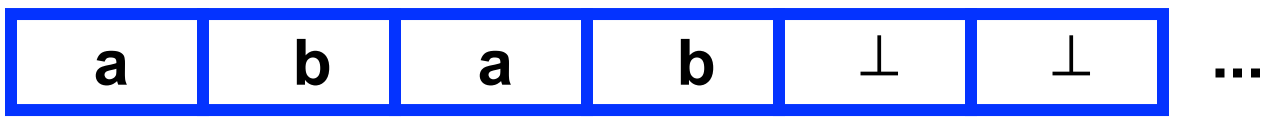
# Sommario

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

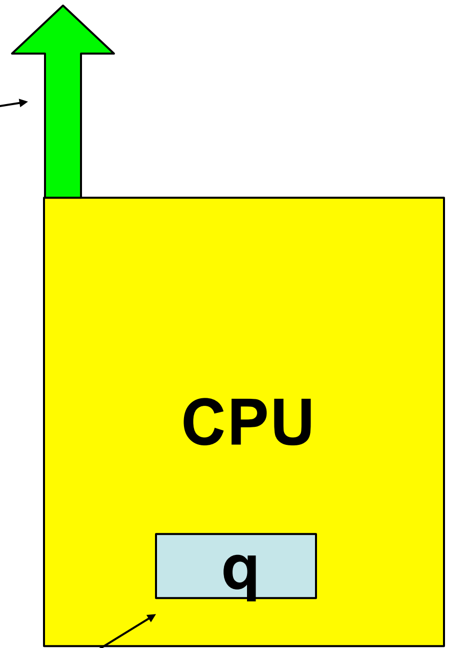
- **JFLAP**

# Macchine di Turing: il modello mentale

simbolo cella vuota



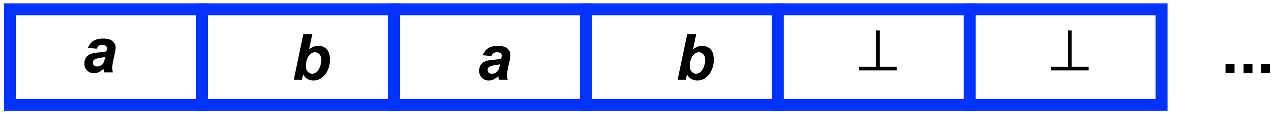

testina di lettura e scrittura, può muoversi verso sinistra e verso destra

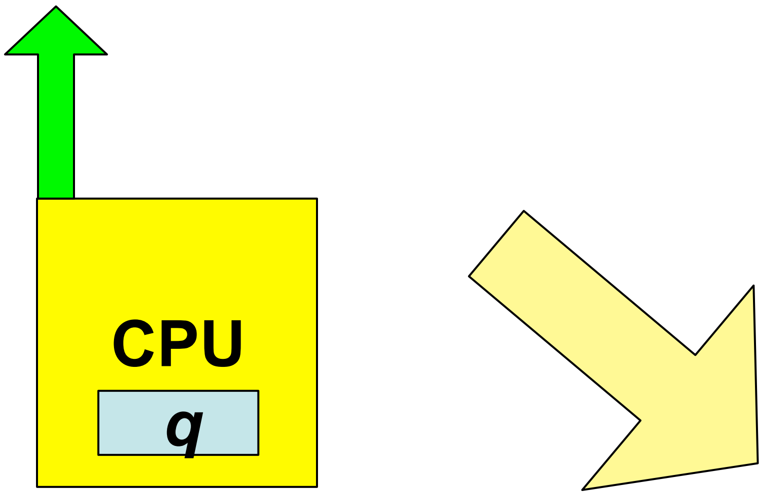



stato

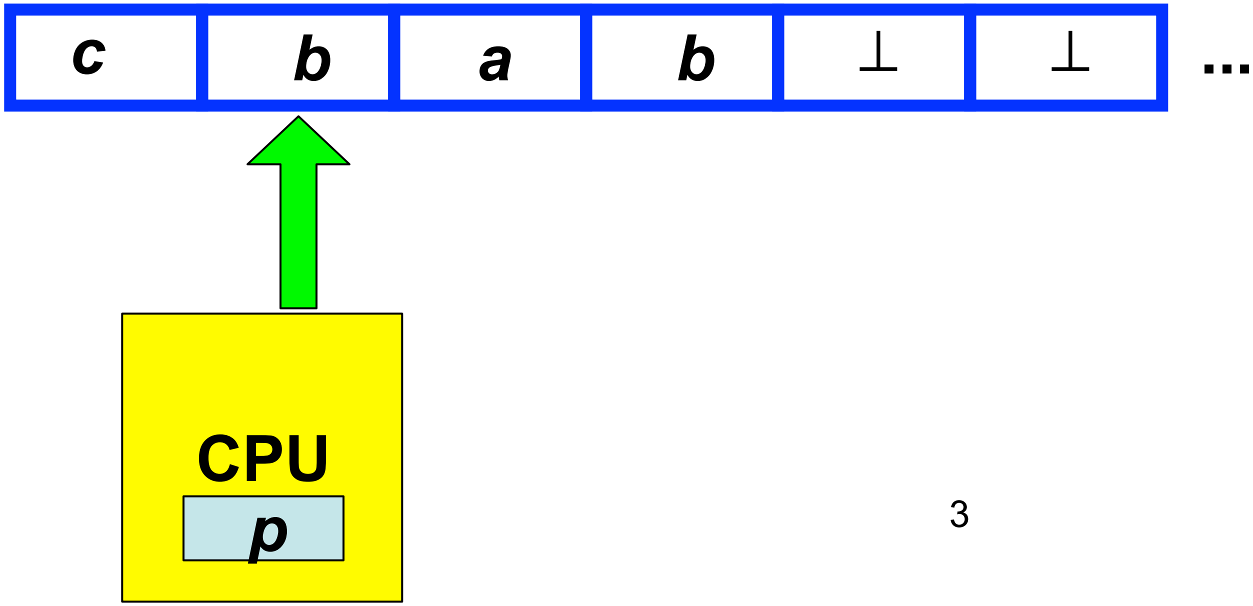
In funzione dello stato in cui si trova e del simbolo input letto la macchina cambia stato, scrive un simbolo nella cella letta, poi sposta la testina di lettura di una posizione a sinistra o a destra. La macchina è dotata di due stati speciali di terminazione, uno per l’accettazione e uno per il rifiuto.

# Macchine di Turing: una mossa a destra

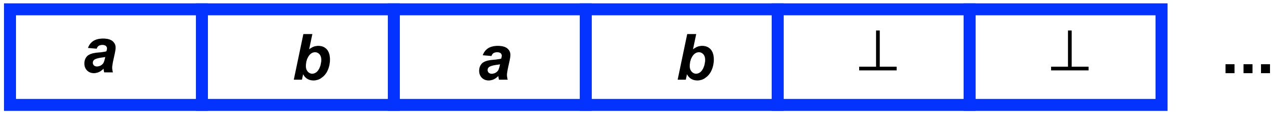


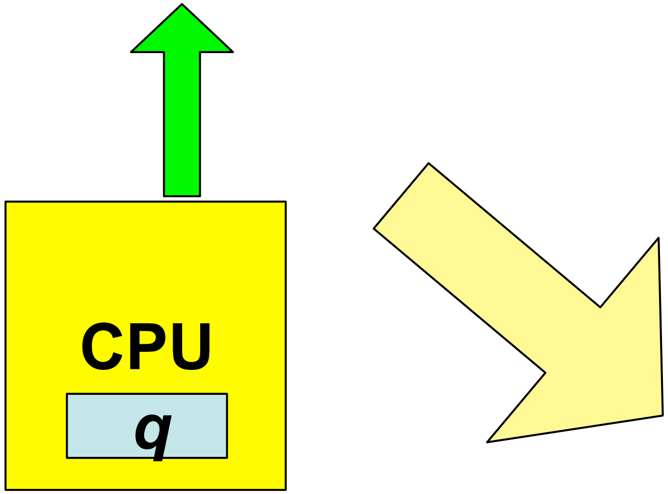




**leggendo *a* nello stato *q* la TM passa nello stato *p* scrive *c* nella casella e sposta la testina di una posizione adestra**

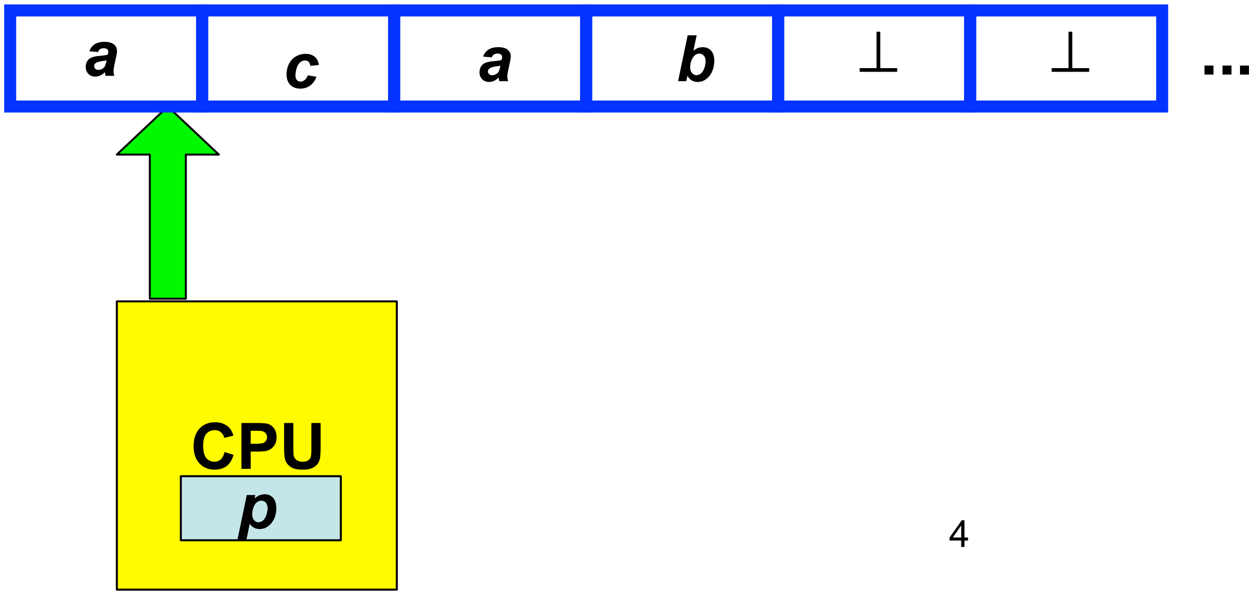


# Macchine di Turing: una mossa a sinistra




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



# Esempio di TM 1

**La TM deve decidere $\{0^n 1^n \mid n>0\}$.**

**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.**

# 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

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

a sinistra:

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

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

dove $\Rightarrow_T^*$ si legge "porta a in più passi"

se $\alpha_1 q_1 \beta_1 \Rightarrow_T \alpha_2 q_2 \beta_2 \Rightarrow_T \dots \Rightarrow_T \alpha_n q_n \beta_n$ allora $\alpha_1 q_1 \beta_1 \Rightarrow_T^* \alpha_n q_n \beta_n$

Diciamo anche che una configurazione c' è raggiungibile da una configurazione c se c $\Rightarrow_T^*$ c'.

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

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

Il linguaggio rifiutato è

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

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:**

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

**Spesso serviranno TM di "servizio", per cui ci è utile la definizione di funzione calcolata da una TM:**

**Data una funzione f : $\Sigma^* \rightarrow \Sigma^*$ una TM M calcola f se**

$$q_0 x \Rightarrow^* q_a f(x)$$

# Esempio di TM 2

La TM deve decidere $\{0^{2^n}\ n \geq 0\}$.
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 scorre il nastro verso destra rimpiazzando uno 0 sì e uno no con X (così il numero è diviso a metà)
passo 2: se al passo 1 si era incontrato un solo 0 allora si accetta
passo 3: se al passo 1 il numero di 0 non è uno ed è dispari allora si rifiuta
passo 4: si torna all'inizio del nastro e si ripete dal passo 1, trascurando le X incontrate

# 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, questo 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, e si marca a X lo 0 che li segue, se non si trova uno 0 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**

# Esempio di “modulo”

**La TM esegue solo uno slittamento a destra dell’input, inserendo un marcatore di inizio nastro, per es. #, nella prima cella. Nella specifica usiamo l’alfabeto binario**

**passo 1: si legge il simbolo nella prima cella, sostituendolo con # proseguendo su due strade identiche tranne per il simbolo memorizzato nello stato**
**caso 0**
**passo 2: si scorre il nastro a destra fino a trovare la cella vuota o un 1;**
**caso cella vuota**
**se si giunge alla cella vuota si scrive lo 0 mancante e si riporta la testina indietro fino a leggere il # inserito**
**se si giunge a leggere un 1 si rimpiazzato l‘1 con uno 0 e si va al passo 2, caso 1.**

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

**Le stringhe binarie elencate in ordine canonico (quasi lessicografico):**

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

1. **Data una sequenza x01...1, con x in {0,1}* e un numero di 1 ≥0, 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 formale come file JFLAP, supponendo di avere inserito il marcatore di fine nastro nella prima cella.**

# 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 <u>effettivamente</u> calcolabili, o equivalentemente la classe dei linguaggi riconosciuti da una TM corrisponde alla classe dei problemi intuitivamente <u>effettivamente</u> 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.**