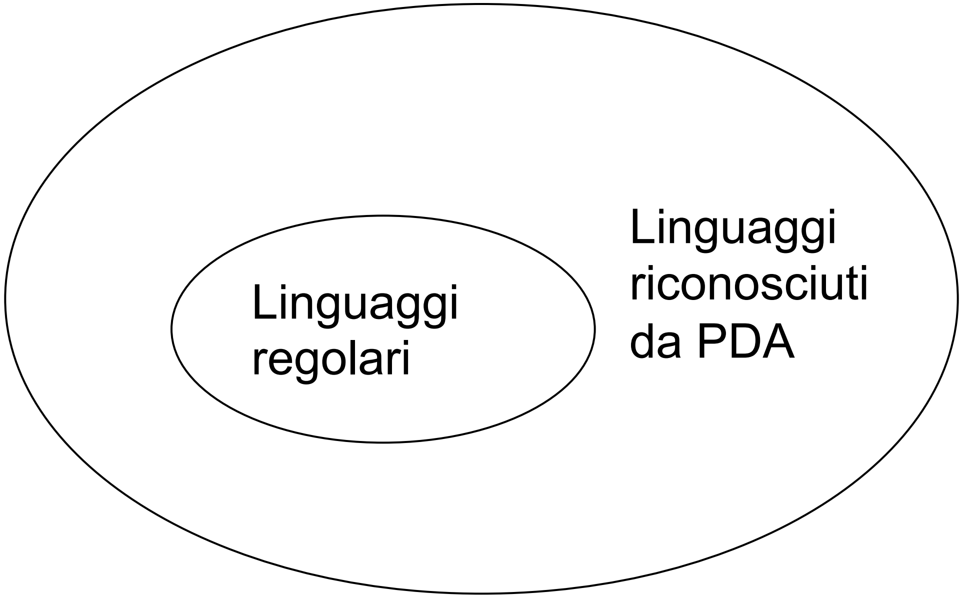
# Linguaggi




Ci sono linguaggi che non possono essere riconosciuti da PDA?

# Unione

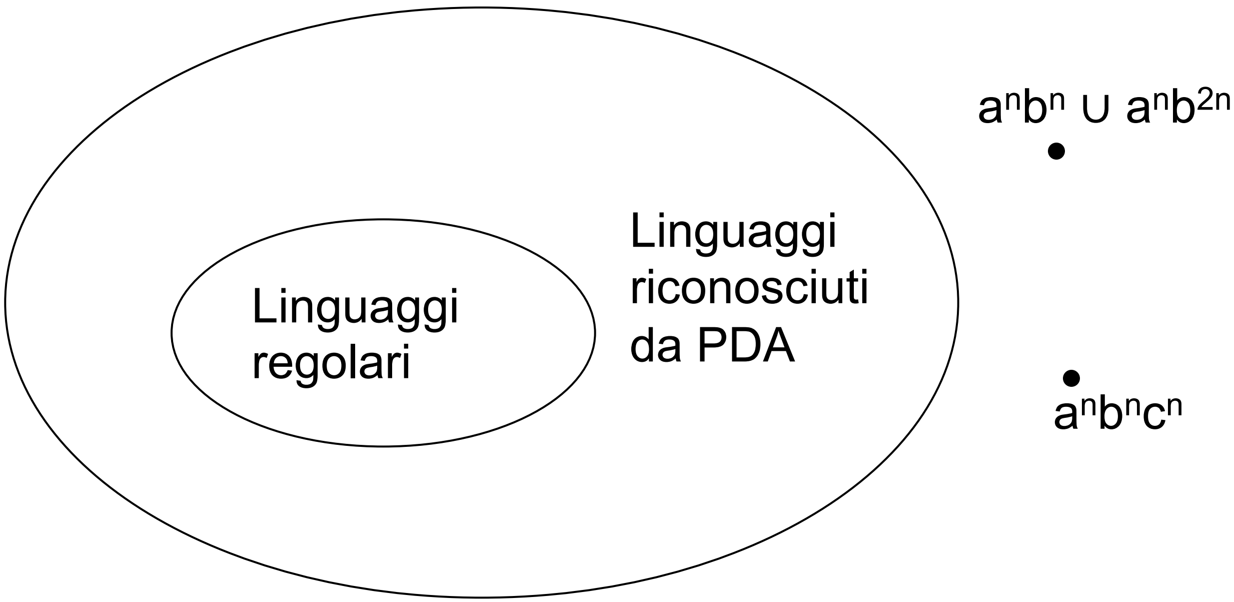
- Abbiamo visto che l'unione di alcuni linguaggi riconosciuti da PDA non può essere riconosciuta da alcun PDA
  - La classe dei PDA non è chiusa rispetto all'unione
- Esempio:
  - $L_1=\{a^n b^n | n \geq 1\}$
  - $L_2=\{a^n b^{2n} | n \geq 1\}$
  - … ma $L_1 \cup L_2$ non è riconoscibile da alcun PDA

# Altro esempio

$L=\{a^n b^n c^n | n>0\}$

- La pila può essere usata per contare le a
- I simboli sulla pila possono essere usati per controllare che il numero di b sia uguale al numero di a
- Come si può ricordare n per controllare il numero di c?

# Linguaggi




Qual è il limite dei PDA?

# Note

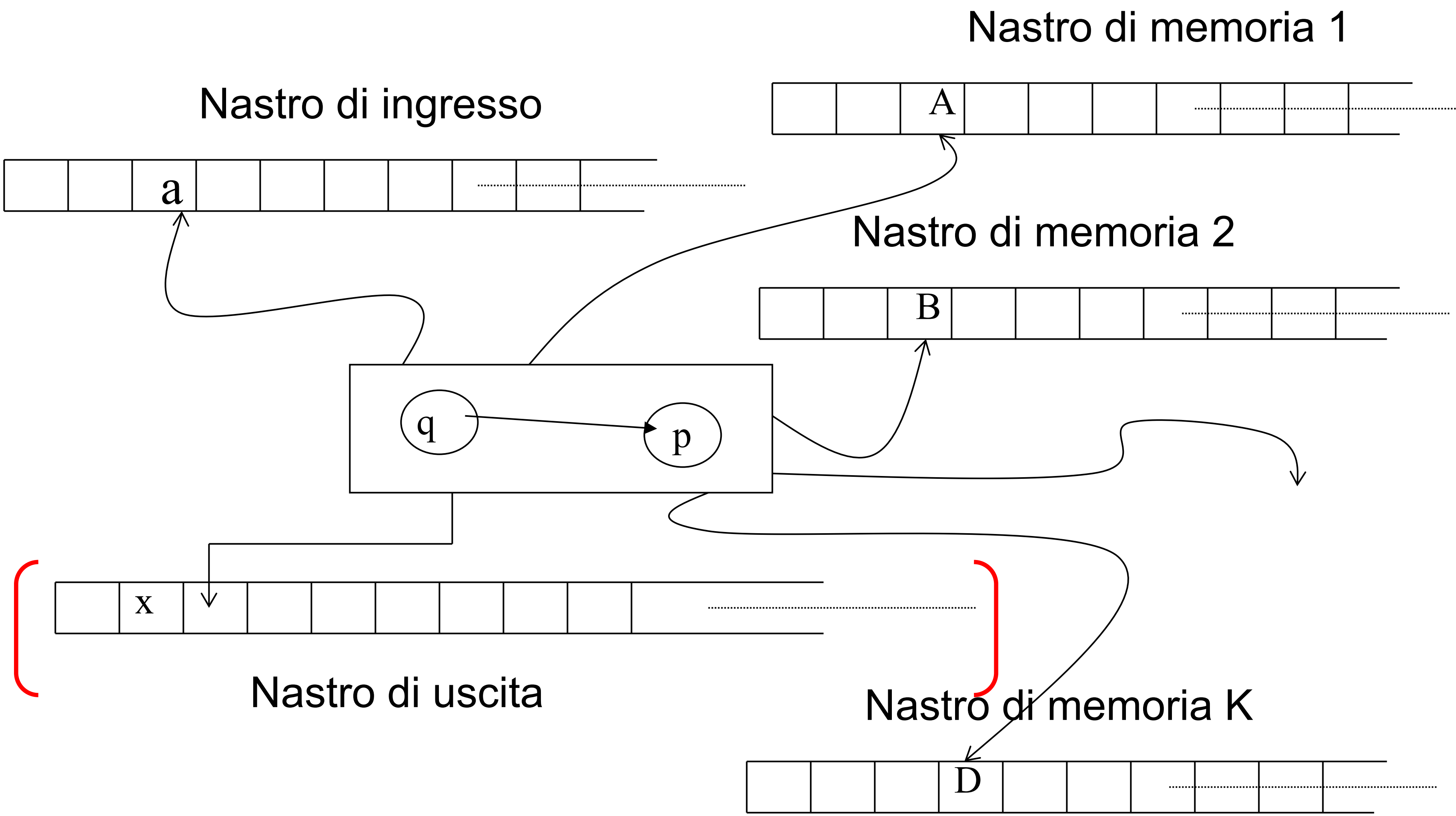
- La pila è una memoria distruttiva
  - Una volta che un simbolo è letto, viene distrutto
- La limitazione della pila può essere mostrata formalmente attraverso una generalizzazione del pumping lemma
- E' necessario usare una memoria persistente

  $\rightarrow$ <u>nastri di memoria</u>

# Macchina di Turing

- La macchina di Turing (Turing machine o TM) è il modello storico del computer
  - semplice
  - concettualmente importante
- Le TM usano i nastri come memorie
  - I nastri non sono distruttivi
  - Possono essere letti molte volte

# Modello generale



# Descrizione informale

- Gli stati e l'alfabeto sono come negli altri automi
  - Ingresso
  - Uscita
  - Dispositivo di controllo
  - Alfabeto di memoria
- I nastri sono rappresentati come sequenze infinite di celle con un simbolo speciale detto "blank" (o anche 'ƀ', '_' o '-')
  - In ogni momento, i nastri contengono solo un numero finito di simboli non-blank

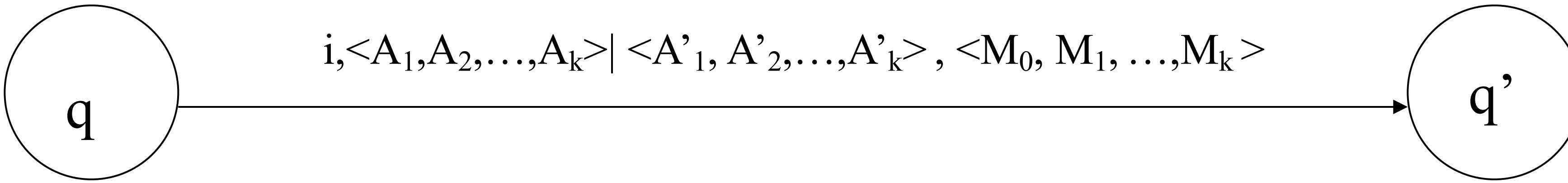
# Mosse

- Le mosse sono basate su
  - un simbolo letto dal nastro di ingresso
  - K simboli, uno per ogni nastro di memoria
  - stato del dispositivo di controllo
- Azioni
  - Cambio di stato
  - Scrittura di un simbolo che rimpiazza quello letto dal nastro di memoria
  - Spostamento delle K+1 testine

# Mosse delle testine

- Le testine della memoria e dell'ingresso possono essere spostate in tre direzioni
  - A destra di una posizione (R)
  - A sinistra di una posizione (L)
  - Ferme (S)
- La direzione di ogni testina dev'essere specificata esplicitamente

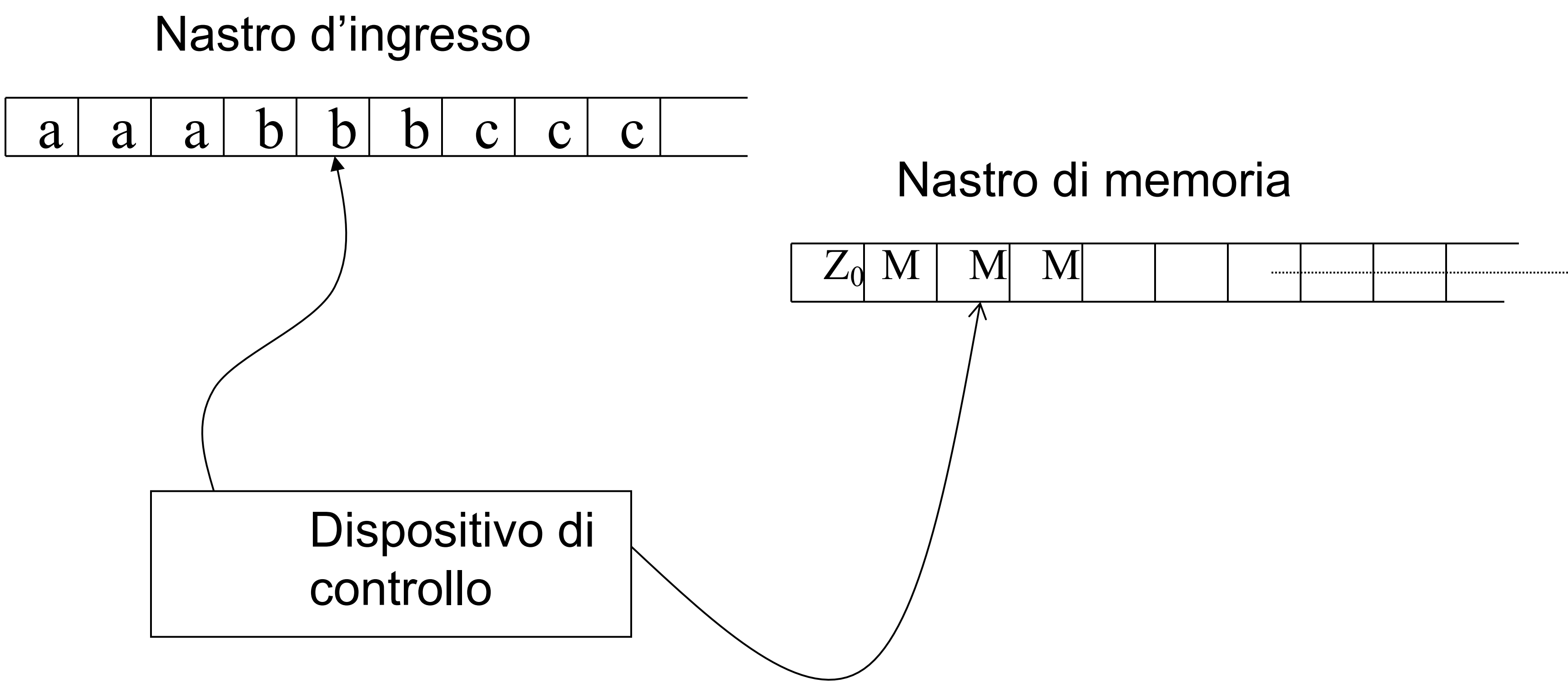
# Graficamente

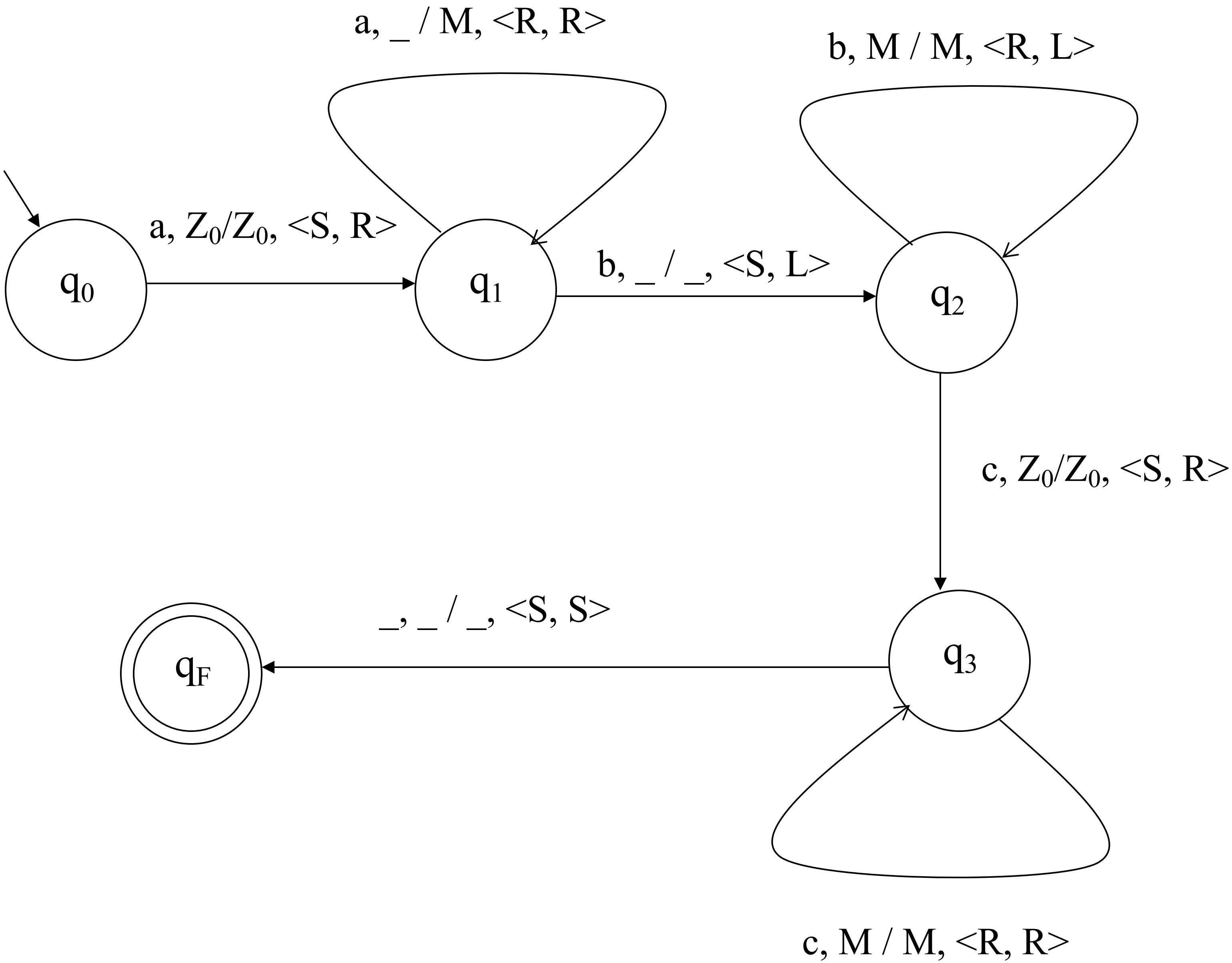



- i è il simbolo di ingresso
- $A_j$ ($1 \leq j \leq k$) è il simbolo letto dal j-esimo nastro di memoria
- $A_j$' ($1 \leq j \leq k$) è il simbolo che rimpiazza $A_j$
- $M_0$ è la direzione della testina del nastro d'ingresso
- $M_j$ ($1 \leq j \leq k$) è la direzione della testina del j-esimo nastro

# Esempio

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




a, _ / M, <R, R>
b, M / M, <R, L>
a, Z0/Z0, <S, R>
b, _ / _, <S, L>
q0
q1
q2
c, Z0/Z0, <S, R>
_, _ / _, <S, S>
qF
q3
c, M / M, <R, R>

# Formalmente

- Una TM con K nastri è una tupla di 7 elementi $M=\langle Q,I,\Gamma,\delta,q_0,Z_0,F\rangle$
  - Q è un insieme finito di stati
  - I è l'alfabeto di ingresso
  - $\Gamma$ è l'alfabeto di memoria
  - $\delta$ è la funzione di transizione
  - $q_0 \in Q$ è lo stato iniziale
  - $Z_0 \in \Gamma$ è il simbolo iniziale di memoria
  - $F \subseteq Q$ è l'insieme di stati finali

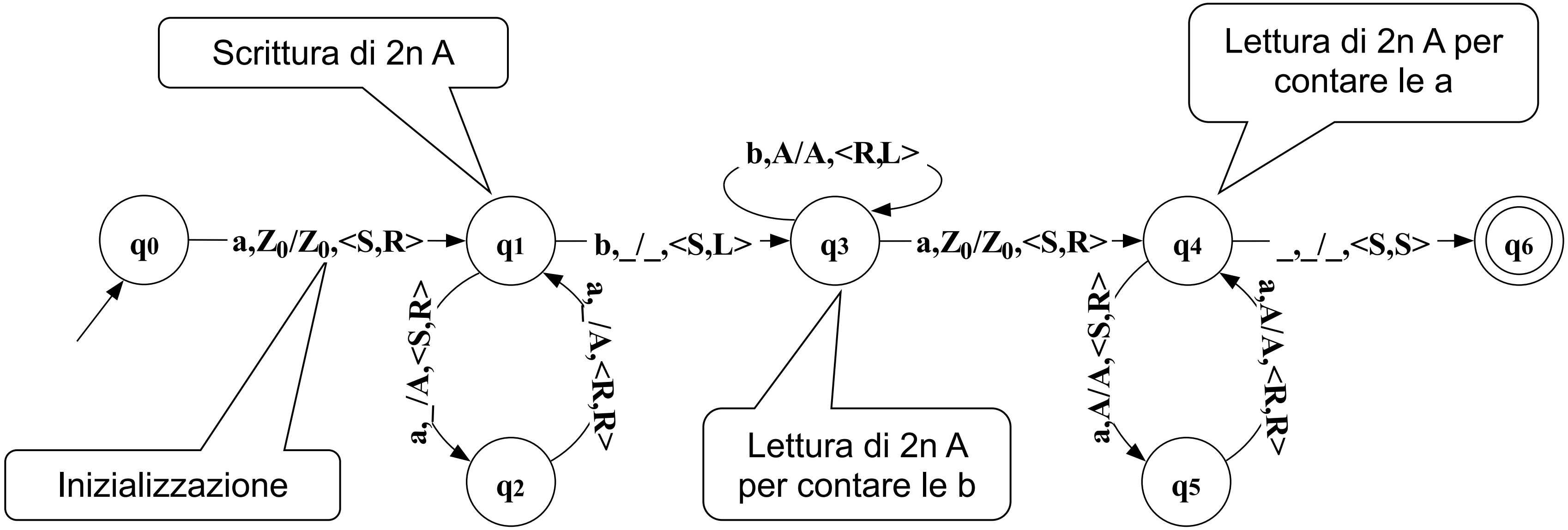
# Funzione di transizione

- La funzione di transizione è definita come
  $\delta$: $(Q\text{-}F) \times I \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{R,L,S\}^{k+1}$
- Nota
  – La funzione può essere parziale

# Esempio

$L = \{a^n b^{2n} a^n \mid n \geq 1\}$

- Scriviamo 2n A su un nastro di memoria e le usiamo per controllare le b e poi le a



# Informalmente

- Una <u>configurazione</u> di una TM è una fotografia della macchina
- Una configurazione deve includere:
  - lo stato del dispositivo di controllo
  - la stringa sul nastro di ingresso e la posizione della testina
  - la stringa e la posizione della testina per ogni nastro di memoria

# Definizione

Una configurazione c di una TM con K nastri di memoria è la seguente (K+2)-tupla:

$$c=<q,\ x\uparrow iy,\ \alpha_1\uparrow A_1\beta_1,\ \ldots,\ \alpha_K\uparrow A_K\beta_K>$$

dove

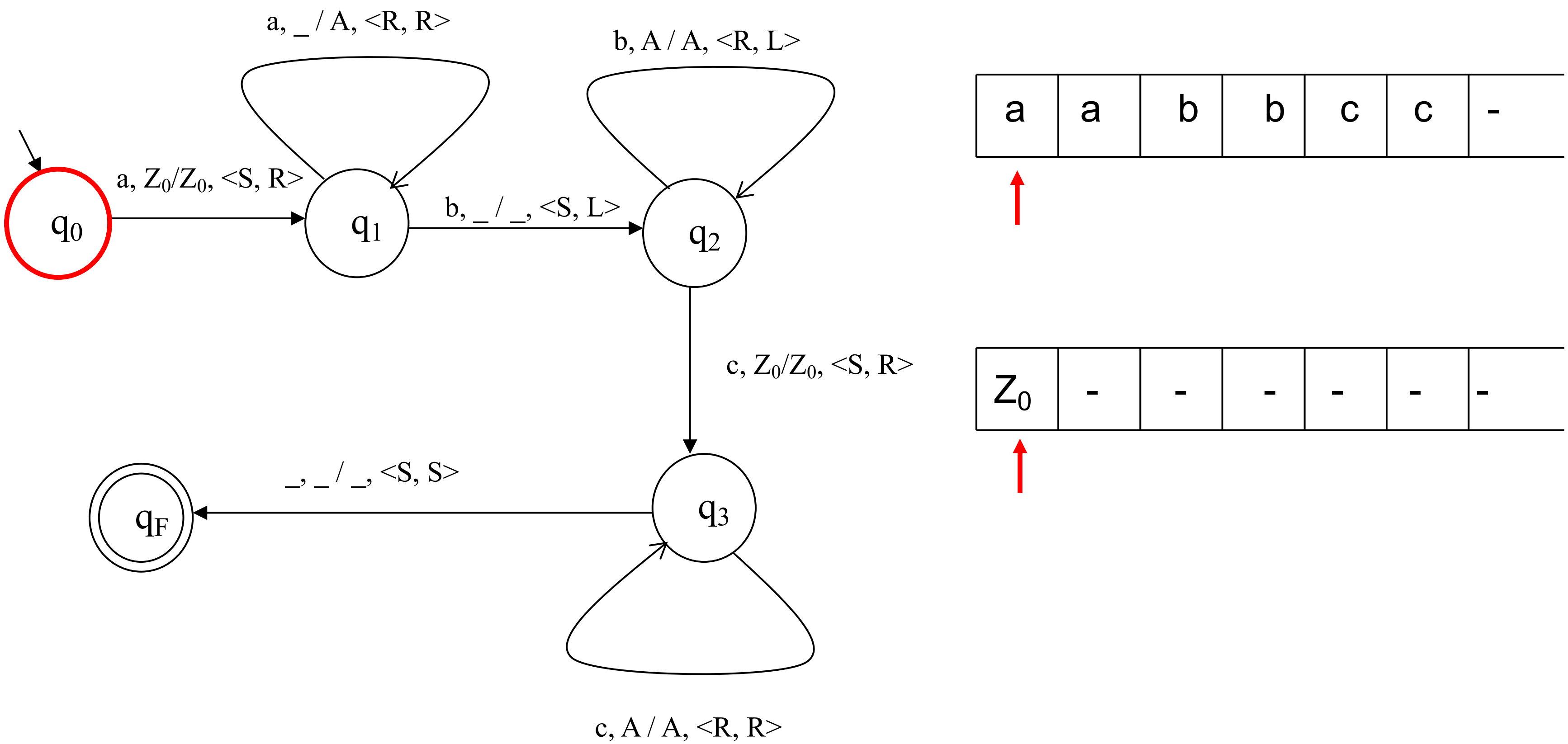
- $q \in Q$
- $x,y \in I^*$, $i \in I$
- $\alpha_r,\beta_r \in \Gamma^*$, $A_r \in \Gamma$ $\forall r\ 1 \leq r \leq K$
- $\uparrow \notin I \cup \Gamma$

# Configurazione iniziale

$$c_0 = \langle q_0, \uparrow iy, \uparrow Z_0, \ldots, \uparrow Z_0 \rangle$$

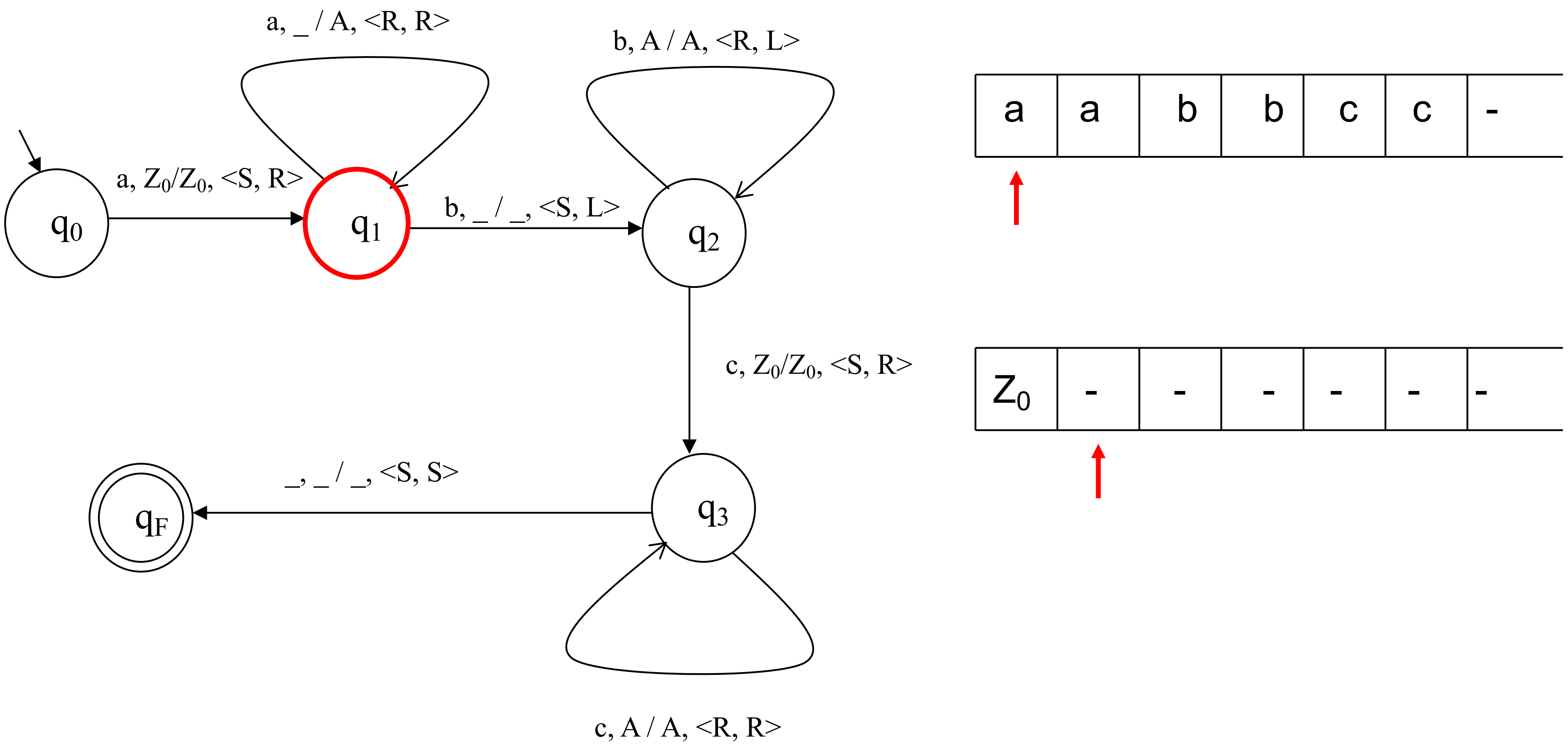
- Formalmente:
  - $x = \varepsilon$
  - $\alpha_r, \beta_r = \varepsilon$, $A_r = Z_0$ $\forall r$ $1 \leq r \leq K$
- Informalmente:
  - Il dispositivo di controllo è nello stato iniziale
  - Tutte le testine sono all'inizio del nastro corrispondente

# Esempio



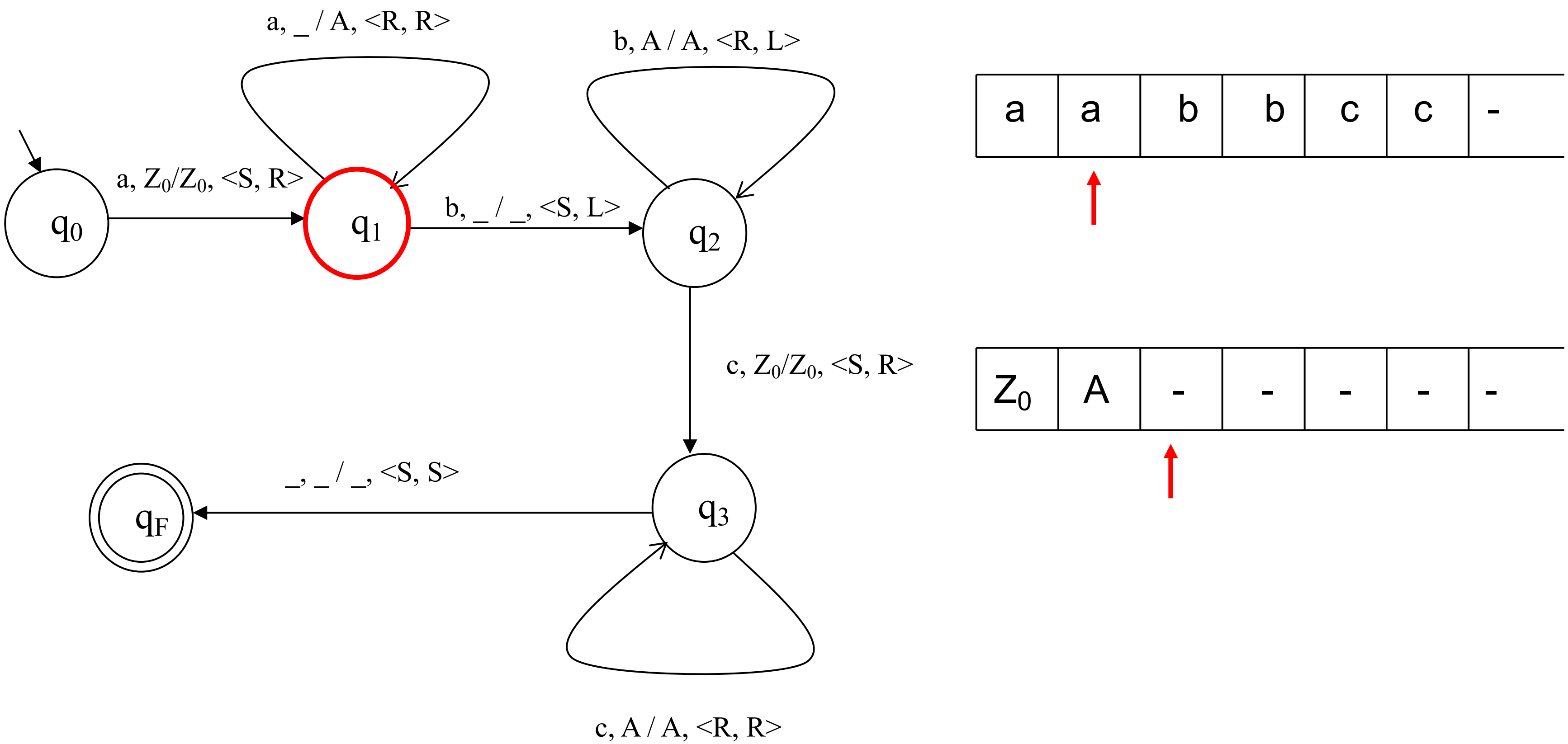

$$c=\langle q_0, \uparrow aabbcc, \uparrow Z_0 \rangle$$

# Esempio



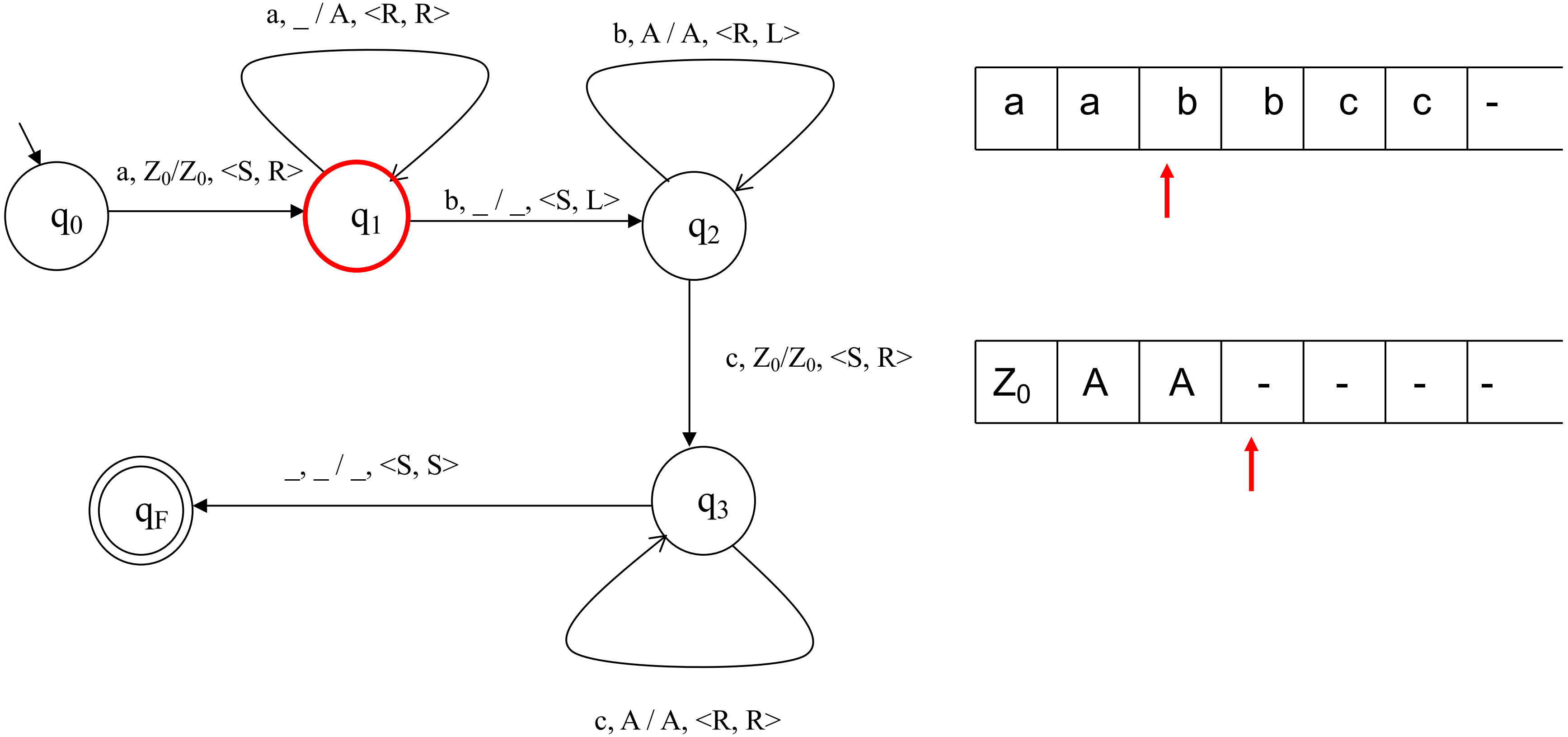

$$c=\langle q_1, \uparrow aabbcc, Z_0\uparrow\rangle$$

# Esempio




$$c=\langle q_1, a\uparrow abbcc, Z_0A\uparrow\rangle$$

# Esempio




c=<$q_1$, aa↑bbcc, $Z_0$AA↑>

# Transizione (1)

- $\delta(q,i,A_1,\dots,A_k)=<p,C_1,\dots,C_k,N,N_1,\dots,N_k>$
  - $p \in Q$
  - $N \in \{L,S,R\}$
  - $C_r \in \Gamma$, $N_r \in \{L,S,R\}$ $\forall r\ 1 \le r \le K$
- $c=<q,\ x\uparrow iy,\ \alpha_1\uparrow A_1\beta_1,\ \dots,\ \alpha_K\uparrow A_K\beta_K>$
  - $x=\underline{x}\underline{i}$, $y=\underline{j}\underline{y}$, $\alpha_r = \underline{\alpha}_r\underline{A}_r$, $\beta_r = \underline{B}_r\underline{\beta}_r$
- $c'=<q',\ x'\uparrow i'y',\ \alpha'_1\uparrow A'_1\beta'_1,\ \dots,\ \alpha'_K\uparrow A'_K\beta'_K>$

$$\mathbf{c \vdash_M c'}$$

Se e solo se valgono le condizioni seguenti

# Transizione (2)

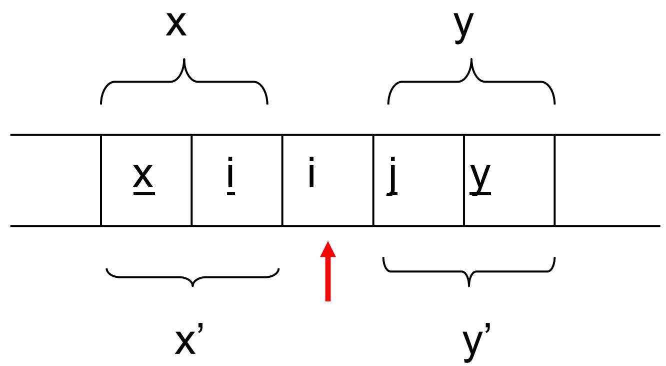
1. p=q’
2. Vale una (e solo una) delle condizioni seguenti

   2.1 N=S, x’=x, i’=i, y’=y

   2.2 N=R, x’=xi, i’=$\underline{j}$, y’=$\underline{y}$

   if y=ε then i’=_ and y’=ε

   2.3 N=L, x’=$\underline{x}$, i’=$\underline{i}$, y’=iy




$\boxed{\delta(q,i,A_1,\ldots,A_k)=<p,C_1,\ldots,C_k,N,N_1,\ldots,N_k>}$

# Transizione (2)

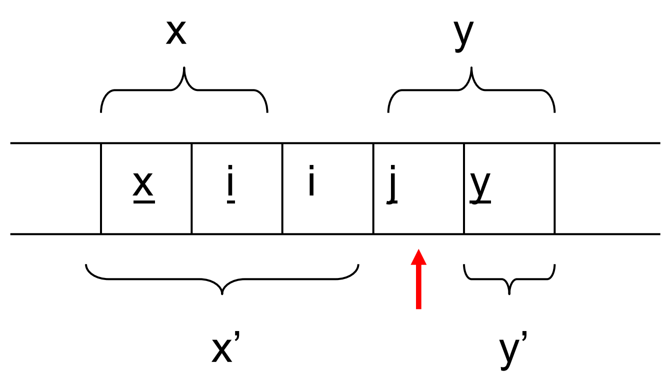
1. p=q’
2. Vale una (e solo una) delle condizioni seguenti

   2.1 N=S, x’=x, i’=i, y’=y

   2.2 N=R, x’=xi, i’=$\underline{\text{j}}$, y’=$\underline{\text{y}}$

   if y=ε then i’=_ and y’=ε

   2.3 N=L, x’=$\underline{\text{x}}$, i’=$\underline{\text{i}}$, y’=iy




$$\boxed{\delta(q,i,A_1,\ldots,A_k)=\langle p,C_1,\ldots,C_k,N,N_1,\ldots,N_k\rangle}$$

# Transizione (2)

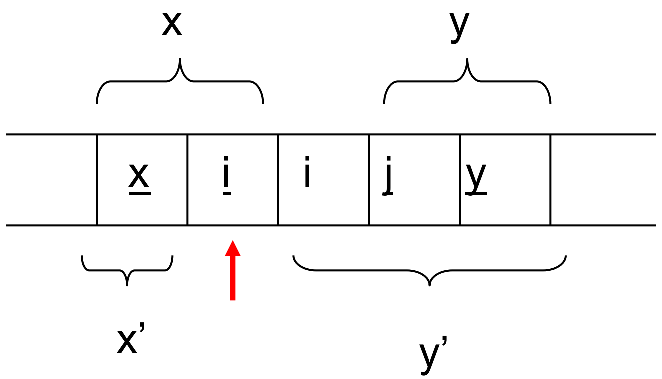
1. p=q’
2. Vale una (e solo una) delle condizioni seguenti

   2.1 N=S, x’=x, i’=i, y’=y

   2.2 N=R, x’=xi, i’=$\underline{j}$, y’=$\underline{y}$

   if y=ε then i’=_ and y’=ε

   2.3 N=L, x’=$\underline{x}$, i’=$\underline{i}$, y’=iy




$$\boxed{\delta(q,i,A_1,\ldots,A_k)=\langle p,C_1,\ldots,C_k,N,N_1,\ldots,N_k\rangle}$$

# Transizione (3)

3. Vale una (e solo una) delle condizioni seguenti $\forall r\ 1 \leq r \leq K$

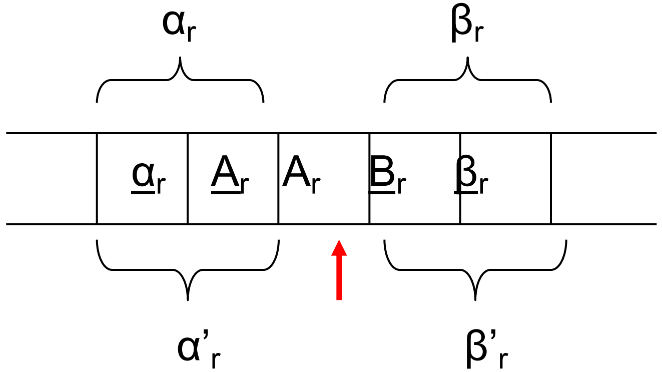
    3.1 $N_r=S$, $\alpha'_r=\alpha_r$, $A'_r=C_r$, $\beta'_r=\beta_r$

    3.2 $N_r=R$, $\alpha'_r=\alpha_rC_r$, $A'_r=\underline{B}_r$, $\beta'_r=\underline{\beta}_r$

    if $\beta_r=\varepsilon$ then $A'_r=\_$ and $\beta'_r=\varepsilon$

    3.3 $N_r=L$, $\alpha'_r=\underline{\alpha}_r$, $A'_r=\underline{A}_r$, $\beta'_r=C_r\beta_r$

$$\boxed{\delta(q,i,A_1,\ldots,A_k)=<p,C_1,\ldots,C_k,N,N_1,\ldots,N_k>}$$

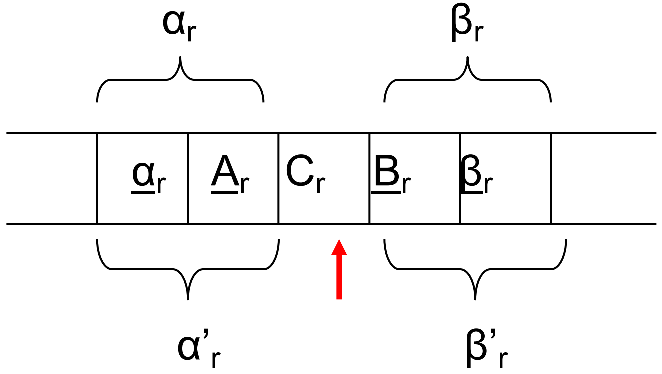


# Transizione (3)

3. Vale una (e solo una) delle condizioni seguenti $\forall r\ 1 \leq r \leq K$
   - 3.1 $N_r = S$, $\alpha'_r = \alpha_r$, $A'_r = C_r$, $\beta'_r = \beta_r$
   - 3.2 $N_r = R$, $\alpha'_r = \alpha_r C_r$, $A'_r = \underline{B}_r$, $\beta'_r = \underline{\beta}_r$

     if $\beta_r = \varepsilon$ then $A'_r = \_$ and $\beta'_r = \varepsilon$
   - 3.3 $N_r = L$, $\alpha'_r = \underline{\alpha}_r$, $A'_r = \underline{A}_r$, $\beta'_r = C_r \beta_r$

$$\boxed{\delta(q,i,A_1,\ldots,A_k)=<p,C_1,\ldots,C_k,N,N_1,\ldots,N_k>}$$

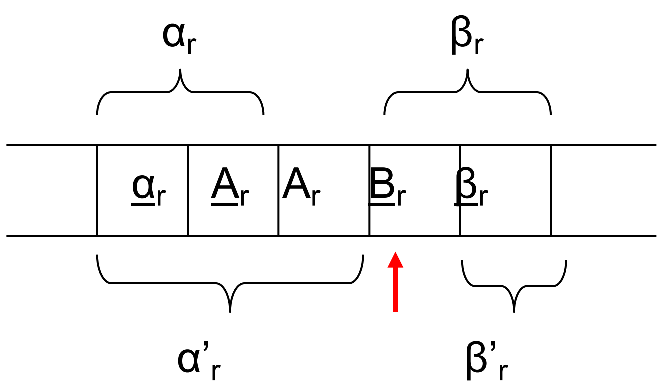


# Transizione (3)

3. Vale una (e solo una) delle condizioni seguenti $\forall r \; 1 \leq r \leq K$
   - 3.1 $N_r = S$, $\alpha'_r = \alpha_r$, $A'_r = C_r$, $\beta'_r = \beta_r$
   - 3.2 $N_r = R$, $\alpha'_r = \alpha_r C_r$, $A'_r = \underline{B}_r$, $\beta'_r = \underline{\beta}_r$

     if $\beta_r = \varepsilon$ then $A'_r = \_$ and $\beta'_r = \varepsilon$
   - 3.3 $N_r = L$, $\alpha'_r = \underline{\alpha}_r$, $A'_r = \underline{A}_r$, $\beta'_r = C_r \beta_r$

$\boxed{\delta(q,i,A_1,\ldots,A_k) = <p,C_1,\ldots,C_k,N,N_1,\ldots,N_k>}$



# Transizione (3)

3. Vale una (e solo una) delle condizioni seguenti $\forall r\ 1 \le r \le K$

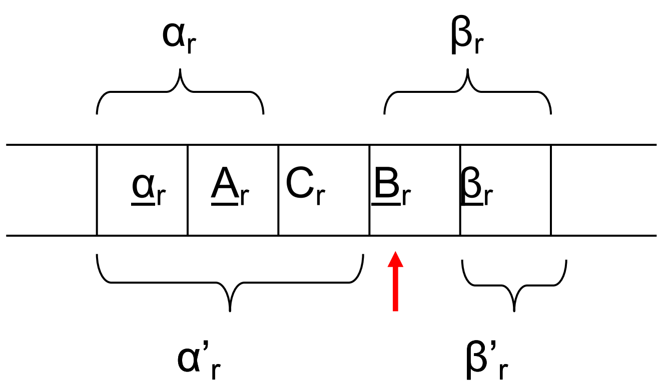
   3.1 $N_r = S$, $\alpha'_r = \alpha_r$, $A'_r = C_r$, $\beta'_r = \beta_r$

   3.2 $N_r = R$, $\alpha'_r = \alpha_r C_r$, $A'_r = \underline{B}_r$, $\beta'_r = \underline{\beta}_r$

   if $\beta_r = \varepsilon$ then $A'_r = \_$ and $\beta'_r = \varepsilon$

   3.3 $N_r = L$, $\alpha'_r = \underline{\alpha}_r$, $A'_r = \underline{A}_r$, $\beta'_r = C_r \beta_r$

$\boxed{\delta(q,i,A_1,\ldots,A_k) = \langle p, C_1,\ldots,C_k, N, N_1,\ldots,N_k \rangle}$



# Transizione (3)

3. Vale una (e solo una) delle condizioni seguenti $\forall r\ 1 \leq r \leq K$

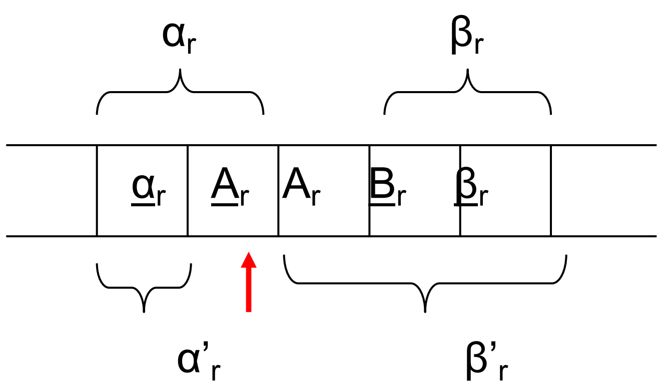
   3.1 $N_r$=S, $\alpha'_r=\alpha_r$, $A'_r=C_r$, $\beta'_r=\beta_r$

   3.2 $N_r$=R, $\alpha'_r=\alpha_r C_r$, $A'_r=\underline{B}_r$, $\beta'_r=\underline{\beta}_r$

   if $\beta_r=\varepsilon$ then $A'_r$=_ and $\beta'_r=\varepsilon$

   3.3 $N_r$=L, $\alpha'_r=\underline{\alpha}_r$, $A'_r=\underline{A}_r$, $\beta'_r=C_r\beta_r$

$\delta(q,i,A_1,\ldots,A_k)=<p,C_1,\ldots,C_k,N,N_1,\ldots,N_k>$

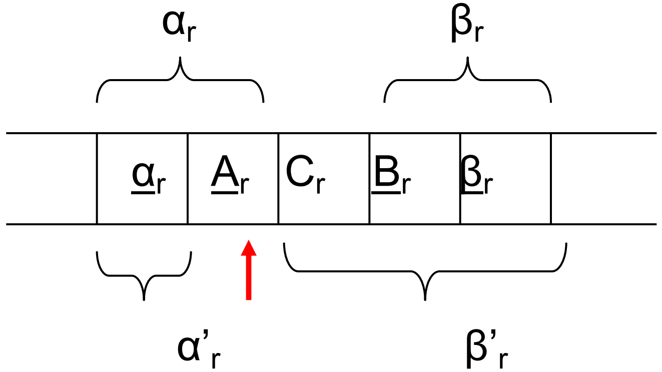


# Transizione (3)

3. Vale una (e solo una) delle condizioni seguenti $\forall r\ 1 \leq r \leq K$
   - 3.1 $N_r=S$, $\alpha'_r=\alpha_r$, $A'_r=C_r$, $\beta'_r=\beta_r$
   - 3.2 $N_r=R$, $\alpha'_r=\alpha_r C_r$, $A'_r=\underline{B}_r$, $\beta'_r=\underline{\beta}_r$

     if $\beta_r=\varepsilon$ then $A'_r=\_$ and $\beta'_r=\varepsilon$
   - 3.3 $N_r=L$, $\alpha'_r=\underline{\alpha}_r$, $A'_r=\underline{A}_r$, $\beta'_r=C_r\beta_r$

$\boxed{\delta(q,i,A_1,\ldots,A_k)=<p,C_1,\ldots,C_k,N,N_1,\ldots,N_k>}$



# Condizione di accettazione

- Una stringa $x \in I^*$ è accettata da una TM M con K nastri di memoria se e solo se

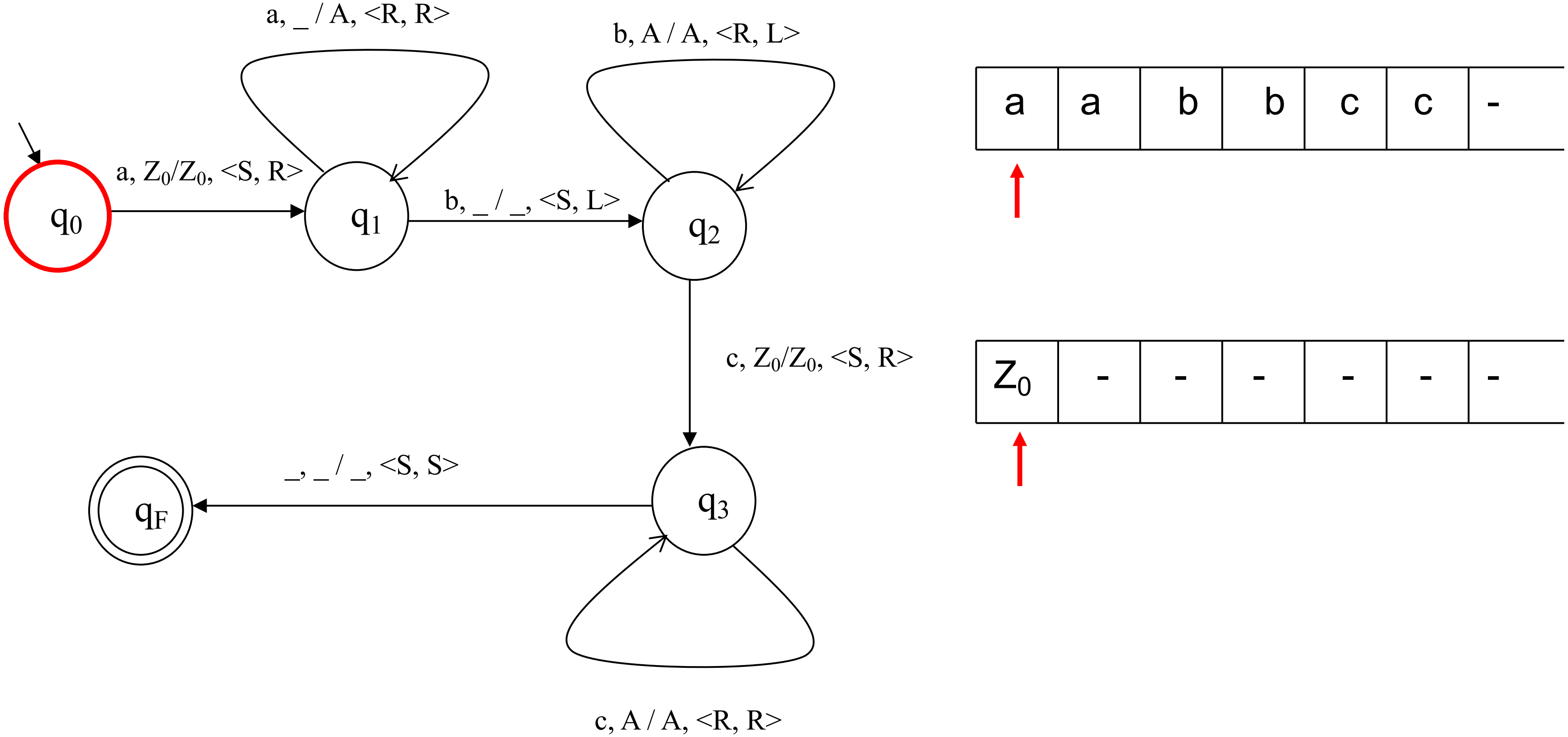
$$\exists q \exists x' \exists i \exists y \exists \alpha_1 \exists A_1 \exists \beta_1 \ldots \exists \alpha_k \exists A_K \exists \beta_K$$

$$c_0 = \langle q_0, \uparrow x, \uparrow Z_0, \ldots, \uparrow Z_0 \rangle \ \vdash^*_M$$

$$c_F = \langle q, x' \uparrow iy, \alpha_1 \uparrow A_1 \beta_1, \ldots, \alpha_K \uparrow A_K \beta_K \rangle \text{ con } q \in F \text{ (e } x = x'iy)$$

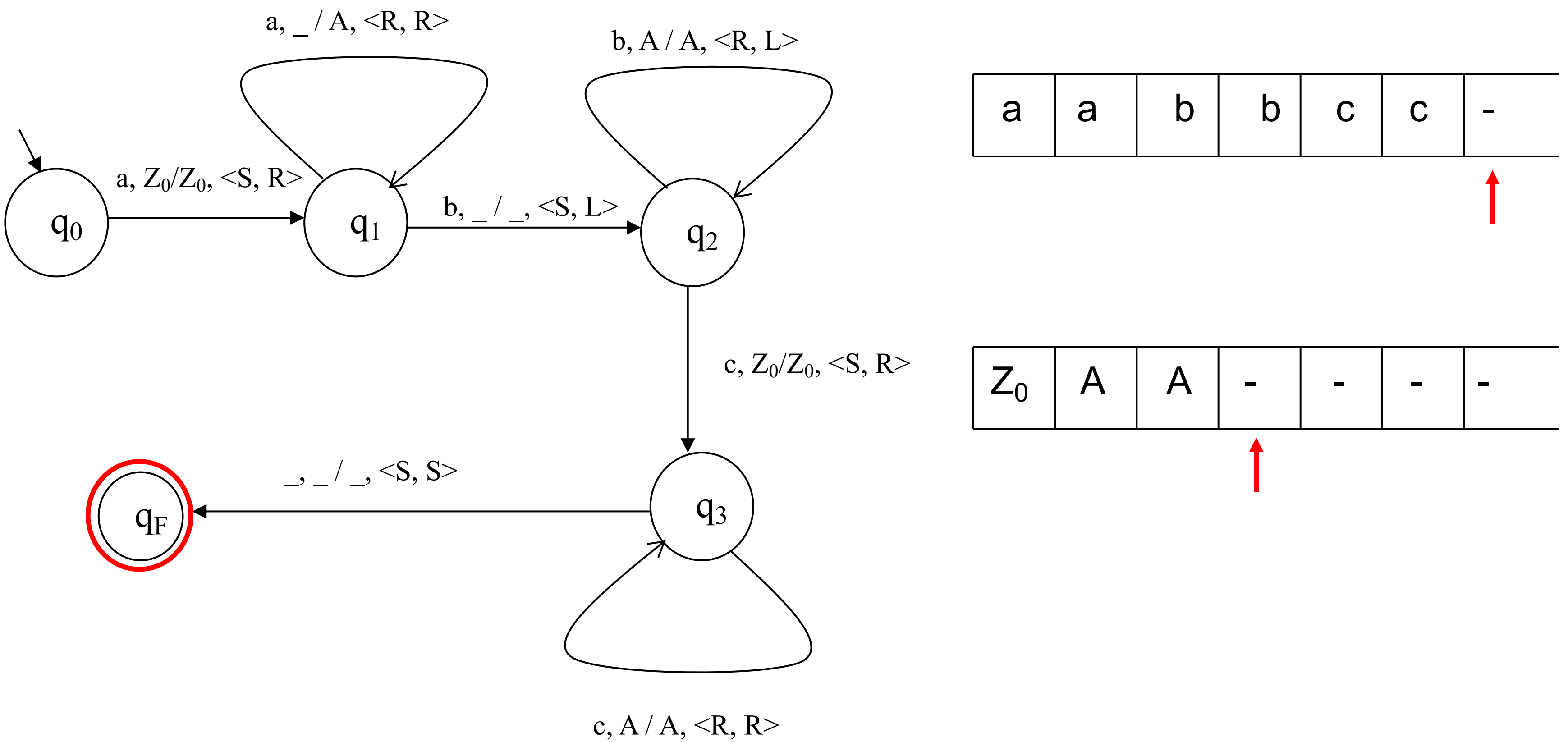
  - $c_F$ è detta configurazione finale
  - $\vdash^*_M$ è la chiusura riflessiva e transitiva di $\vdash_M$
- $L(M) = \{x \mid x \in I^* \text{ e } x \text{ è accettata da } M\}$

# Esempio




$c_0$=<$q_0$, ↑aabbcc, ↑$Z_0$>

# Esempio




$c_0$=<$q_0$, ↑aabbcc, ↑$Z_0$>

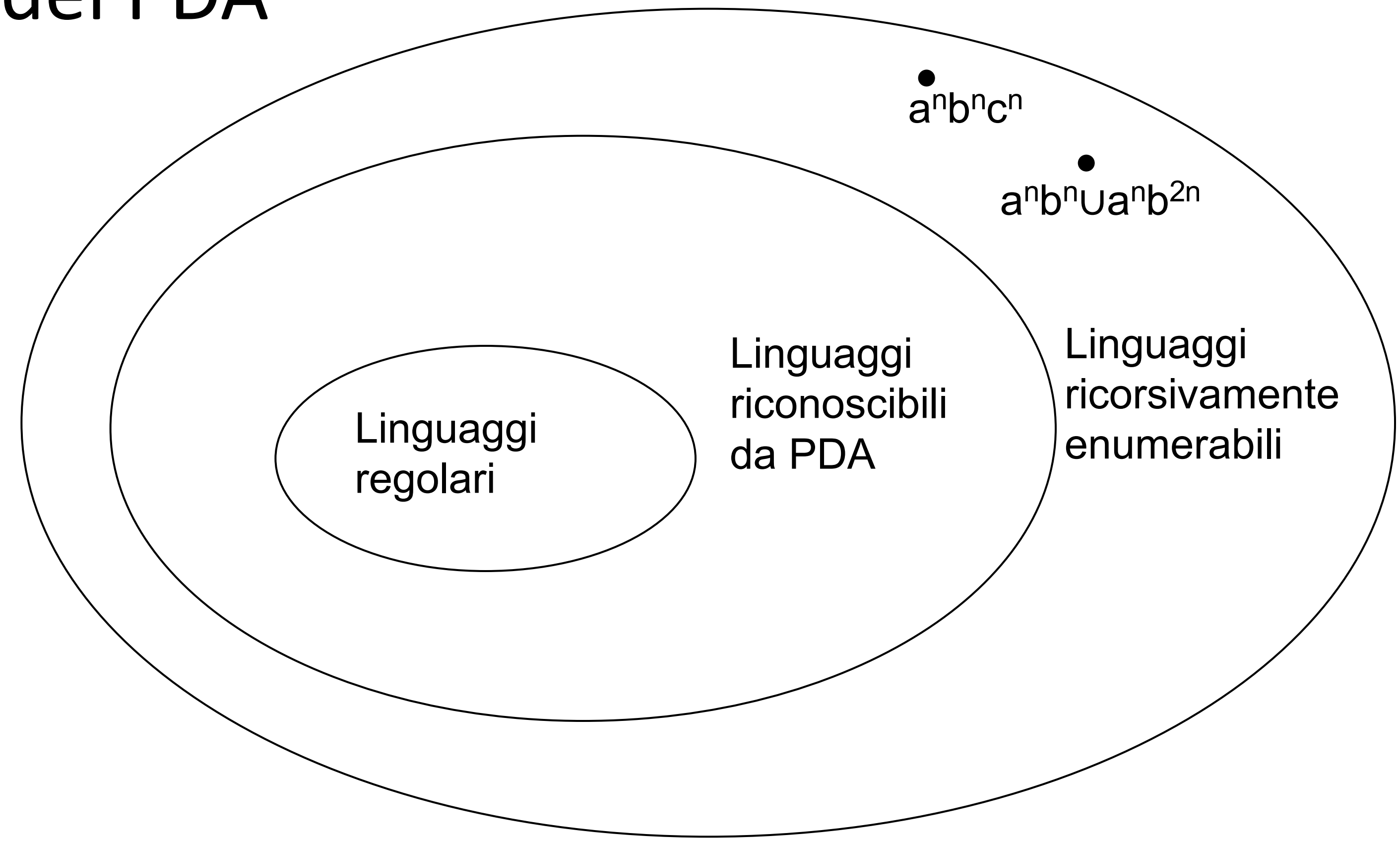
$c_F$=<$q_F$, aabbcc↑, $Z_0$AA↑>

# TM e PDA

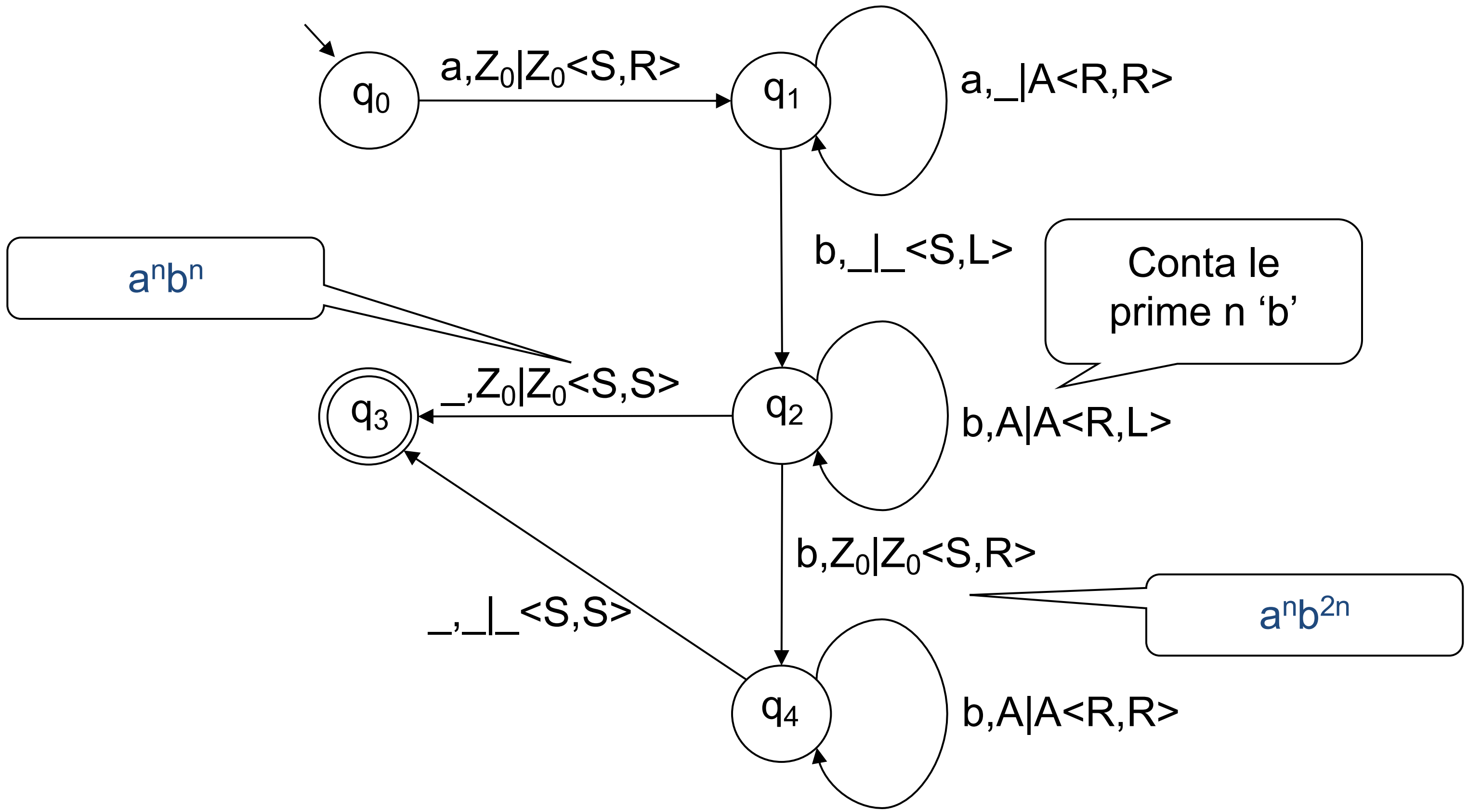
- Sappiamo che $a^n b^n c^n$ o $a^n b^n \cup a^n b^{2n}$ non possono essere riconosciuti da alcun PDA
- ma possono essere riconosciuti da una TM
  - Abbiamo visto una TM per $a^n b^n c^n$
- Ogni linguaggio riconoscibile da un PDA può essere riconosciuto da una TM
  - Si può sempre costruire una TM che usa un nastro di memoria come se fosse una pila
- I linguaggi accettati dalle TM sono detti **ricorsivamente enumerabili**

# Il bersaglio

- Le TM hanno un potere espressivo superiore a quello dei PDA



# Esempio: $a^n b^n U a^n b^{2n}$



# TM e macchine di Von Neumann

- Le TM possono simulare una macchina di Von Neumann (VNM)
  - E’ un modello astratto di computer
- La differenza riguarda l’accesso alla memoria
  - TM: sequenziale
  - VNM: diretto
- Il tipo di accesso alla memoria non influenza il potere espressivo di una macchina
  - Non cambia la classe di problemi risolvibili con tale macchina
  - Può cambiarne la complessità

# Operazioni sulle TM (1)

- Le TM sono chiuse rispetto a
  - Intersezione
  - Unione
  - Concatenazione
  - Stella di Kleene
- Le TM non sono chiuse rispetto al complemento
  - E nemmeno rispetto alla differenza (perché?)

# Operazioni sulle TM (2)

- Chiusura rispetto a unione, intersezione, …: risposta positiva
- Una TM può facilmente simulare due altre TM
  - In serie o
  - In parallelo

  e questo spiega la chiusura

# Complemento

- Le TM acicliche (se esistessero) sarebbero chiuse rispetto al complemento?
  - Sì: basterebbe definire l'insieme di stati di arresto e partizionarli in stati di accettazione e di non accettazione

→ I problemi nascono dalle computazioni che <u>non terminano</u> (da discutere in seguito)

# Definizione

- Una TM trasduttrice con K nastri è una tupla di 9 elementi $M=\langle Q,I,\Gamma,O,\delta,\eta,q_0,Z_0,F\rangle$
  - Q è un insieme finito di stati
  - …
  - $F \subseteq Q$ è l'insieme di stati finali
  - O è l'alfabeto di uscita
  - η è la funzione di uscita

# Funzione di uscita

- La funzione di uscita è definita come
  $\eta$: (Q-F) x I x $\Gamma^k \rightarrow$ O x {R,S}
- Note
  - La funzione può essere parziale
  - E' definita laddove è definita $\delta$
  - Le testine di uscita si possono muovere solo in due direzioni
    - A destra di una posizione (R)
    - Ferme (S)

# Configurazione

Una configurazione c di una TM con K nastri di memoria è la seguente (K+3)-tupla:

$$c=<q, x\uparrow iy, \alpha_1\uparrow A_1\beta_1, \ldots, \alpha_K\uparrow A_K\beta_K, \boxed{u\uparrow o}>$$

dove

- $q \in Q$
- …
- $u \in O^*$, $o \in O$
- $\uparrow \notin I \cup \Gamma \boxed{\cup O}$

# Configurazione iniziale

$$c_0 = \langle q_0, \uparrow iy, \uparrow Z_0, \ldots, \uparrow Z_0, \boxed{\uparrow \_} \rangle$$

- Formalmente:
  - $x = \varepsilon$
  - $\alpha_r = \beta_r = \varepsilon$, $A_r = Z_0$ $\forall r$ $1 \le r \le K$
  - $\boxed{u = \varepsilon, o = \_}$
- Informalmente:
  - Il dispositivo di controllo è nello stato iniziale
  - Tutte le testine sono all'inizio del nastro corrispondente

# Transizione tra configurazioni (1)

- $c=\langle q, x\uparrow iy, \alpha_1\uparrow A_1\beta_1, \ldots, \alpha_K\uparrow A_K\beta_K, \boxed{u\uparrow o}\rangle$
  - $x=\underline{x}i$, $y=j\underline{y}$, $\alpha_r = \underline{\alpha_r}\underline{A_r}$, $\beta_r = \underline{B_r}\underline{\beta_r}$
- $c'=\langle q', x'\uparrow i'y', \alpha'_1\uparrow A'_1\beta'_1, \ldots, \alpha'_K\uparrow A'_K\beta'_K, \boxed{u'\uparrow o'}\rangle$
- $\delta(q,i,A_1,\ldots,A_k)=\langle p,C_1,\ldots,C_k,N,N_1,\ldots,N_k\rangle$
  - …
- $\eta(q,i,A_1,\ldots,A_k)=\langle v,M\rangle$
  - $v\in O$
  - $M\in[S,R]$

$$\mathbf{c}\vdash_{\mathbf{M}}\mathbf{c'}$$

Se e solo se valgono le condizioni seguenti

# Transizione tra configurazioni (2)

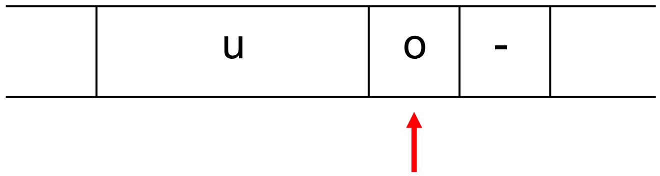
$\eta(q,i,A1,\ldots,Ak)=\langle v,M\rangle$

…

4. Vale una (e solo una) delle condizioni seguenti

   4.1 M=S, u’=u, o’=v

   4.2 M=R, u’=uv, o’=_



# Transizione tra configurazioni (2)

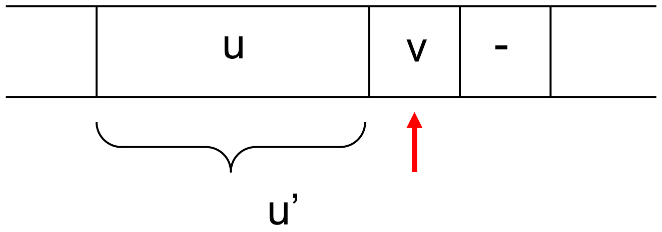
η(q,i,A1,…,Ak)=<v,M>

…

4. Vale una (e solo una) delle condizioni seguenti

   4.1 M=S, u'=u, o'=v

   4.2 M=R, u'=uv, o'=_



# Transizione tra configurazioni (2)

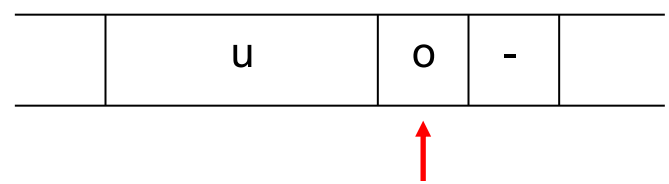
η(q,i,A1,…,Ak)=<v,M>

…

4. Vale una (e solo una) delle condizioni seguenti

    4.1 M=S, u’=u, o’=v

    4.2 M=R, u’=uv, o’=_

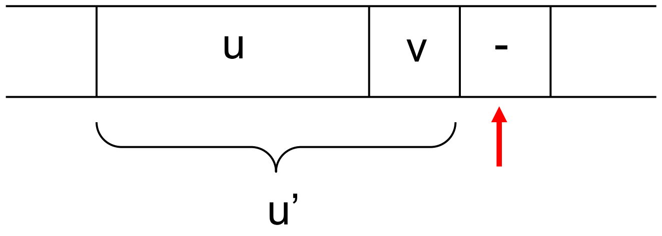


# Transizione tra configurazioni (2)

$\eta(q,i,A1,\ldots,Ak)=\langle v,M\rangle$

...

4. Vale una (e solo una) delle condizioni seguenti
   - 4.1 M=S, u'=u, o'=v
   - 4.2 M=R, u'=uv, o'=_



# Traduzione

- Una TM multinastro M definisce una <u>traduzione</u> $\tau_M$:$I^* \rightarrow O^*$ secondo la regola seguente:

  $$\tau_M(x)=y \text{ se e solo se}$$

  $$\exists q \exists x' \exists i \exists y' \exists \alpha_1 \exists A_1 \exists \beta_1 \ldots \exists \alpha_k \exists A_K \exists \beta_K \exists w \exists o$$

  $c_0$=<$q_0$, ↑x, ↑$Z_0$, …, ↑$Z_0$, ↑-> $\vdash^*_M$

  $c_F$=<q, x' ↑iy', $\alpha_1$↑$A_1\beta_1$, …, $\alpha_k$↑$A_K\beta_K$, w↑o>

  – con $q \in F$ e $y=wo$
- Nota: la traduzione è indefinita se $q \notin F$ o M non si ferma mai quando opera su x

# Come usare le TM?

- Le TM possono
  - Riconoscere linguaggi (accettori)
  - Tradurre linguaggi accettati (trasduttori)
- … ma anche calcolare funzioni
  - Sono equivalenti alle VNM

→ Possiamo pensare alla TM come a un modello astratto di "computer" con accesso sequenziale alla memoria

# Note

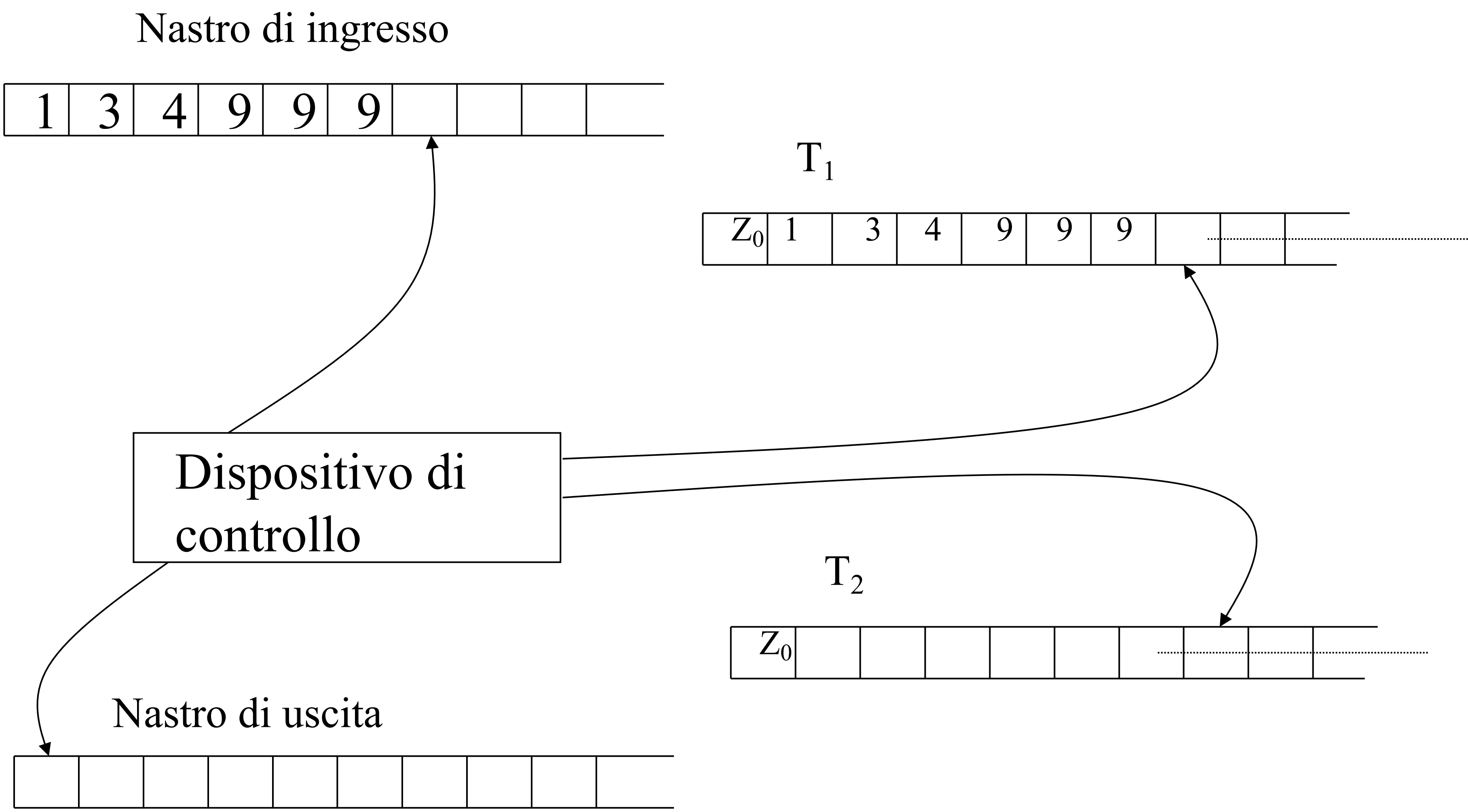
- Come per i problemi seguenti
  - Definire una TM che riconosca un dato linguaggio L
  - Descrivere una TM che traduca un linguaggio L1 in un linguaggio L2

  non c'è una "ricetta" per definire una TM che calcoli una funzione data
- E' importante capire il modello della TM e il problema che vogliamo risolvere
  - Come per "implementare un algoritmo che…"

# TM per calcolare il successore

- Problema: dato un numero n (in notazione decimale), scrivere una TM M che calcoli n+1
- Soluzione:
  - M copia n sul nastro di memoria $T_1$ e sposta la testina su $T_2$ a destra dello stesso numero di posizioni
  - M legge le cifre su $T_1$ da destra a sinistra e scrive corrispondentemente in $T_2$ le cifre modificate
    - 9 diventa 0
    - La prima cifra d (≠9) diventa d+1
  - M copia il numero in $T_2$ sul nastro di uscita

# Punto di vista meccanico



# Notazione

*: cifra decimale

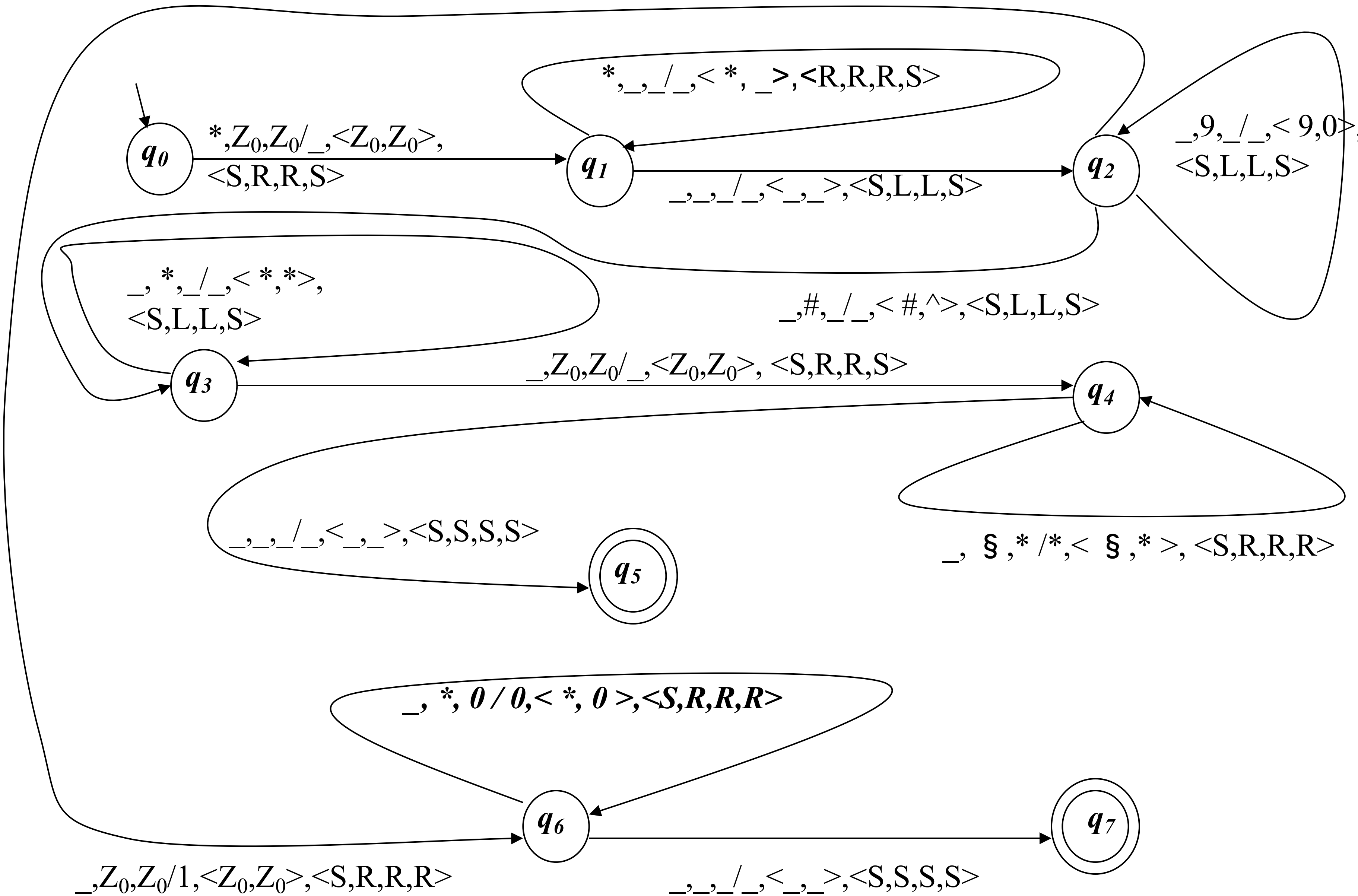
§ : cifra decimale (eventualmente diversa da *)

_ : blank

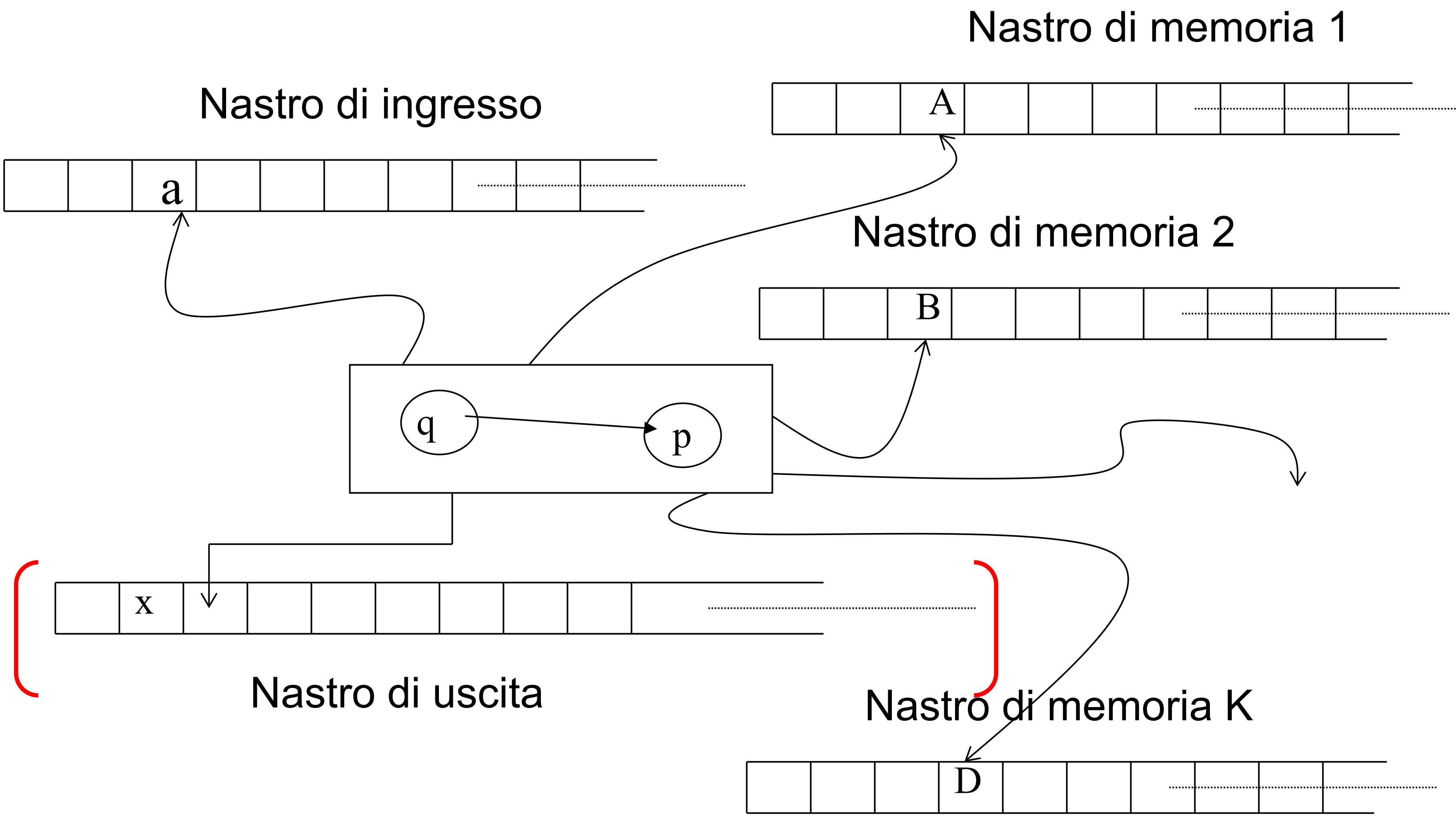
#: cifra decimale tranne 9

^: successore della cifra indicata con # (nella stessa transizione)

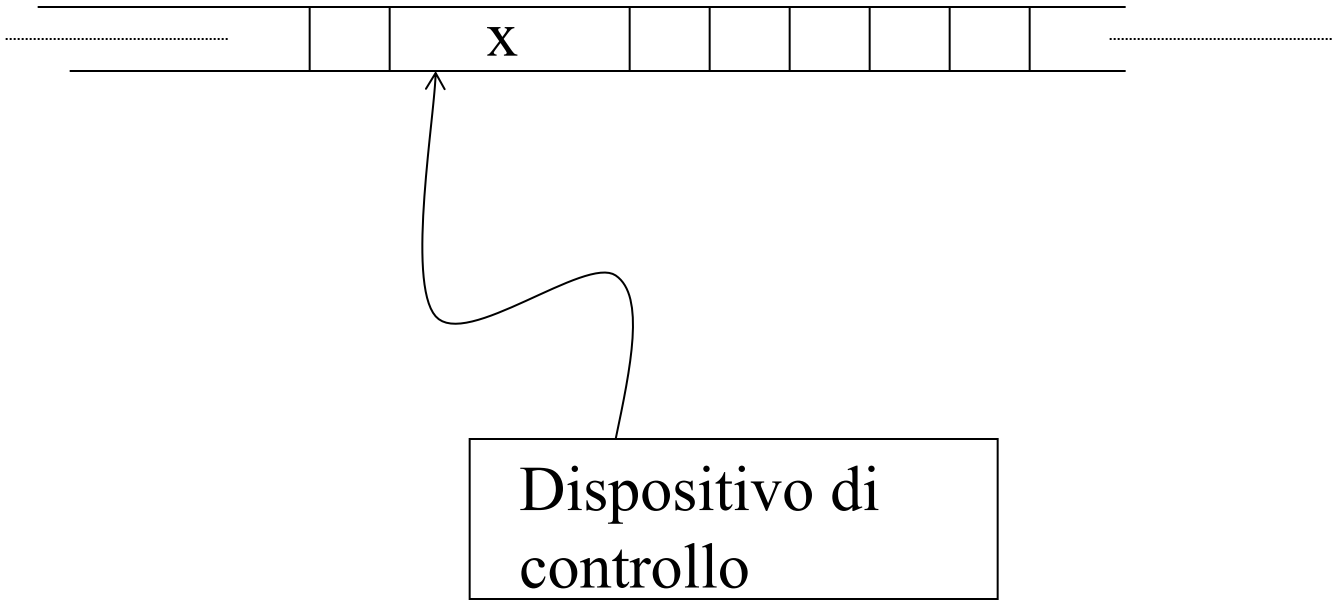
NB: occorrenze multiple dello stesso simbolo in una transizione indicano la stessa cifra


$q_0$
$q_1$
$q_2$
$q_3$
$q_4$
$q_5$
$q_6$
$q_7$
*,_,_/_,< *, _>,<R,R,R,S>
*,$Z_0$,$Z_0$/_,<$Z_0$,$Z_0$>,
<S,R,R,S>
_,_,_/_,<_,_>,<S,L,L,S>
_,9,_/_,< 9,0>,
<S,L,L,S>
_, *,_/_,< *,*>,
<S,L,L,S>
_,#,_/_,< #,^>,<S,L,L,S>
_,$Z_0$,$Z_0$/_,<$Z_0$,$Z_0$>, <S,R,R,S>
_,_,_/_,<_,_>,<S,S,S,S>
_, § ,* /*,< § ,* >, <S,R,R,R>
_, *, 0 / 0,< *, 0 >,<S,R,R,R>
_,$Z_0$,$Z_0$/1,<$Z_0$,$Z_0$>,<S,R,R,R>
_,_,_/_,<_,_>,<S,S,S,S>

# Modello della TM



# TM a nastro singolo

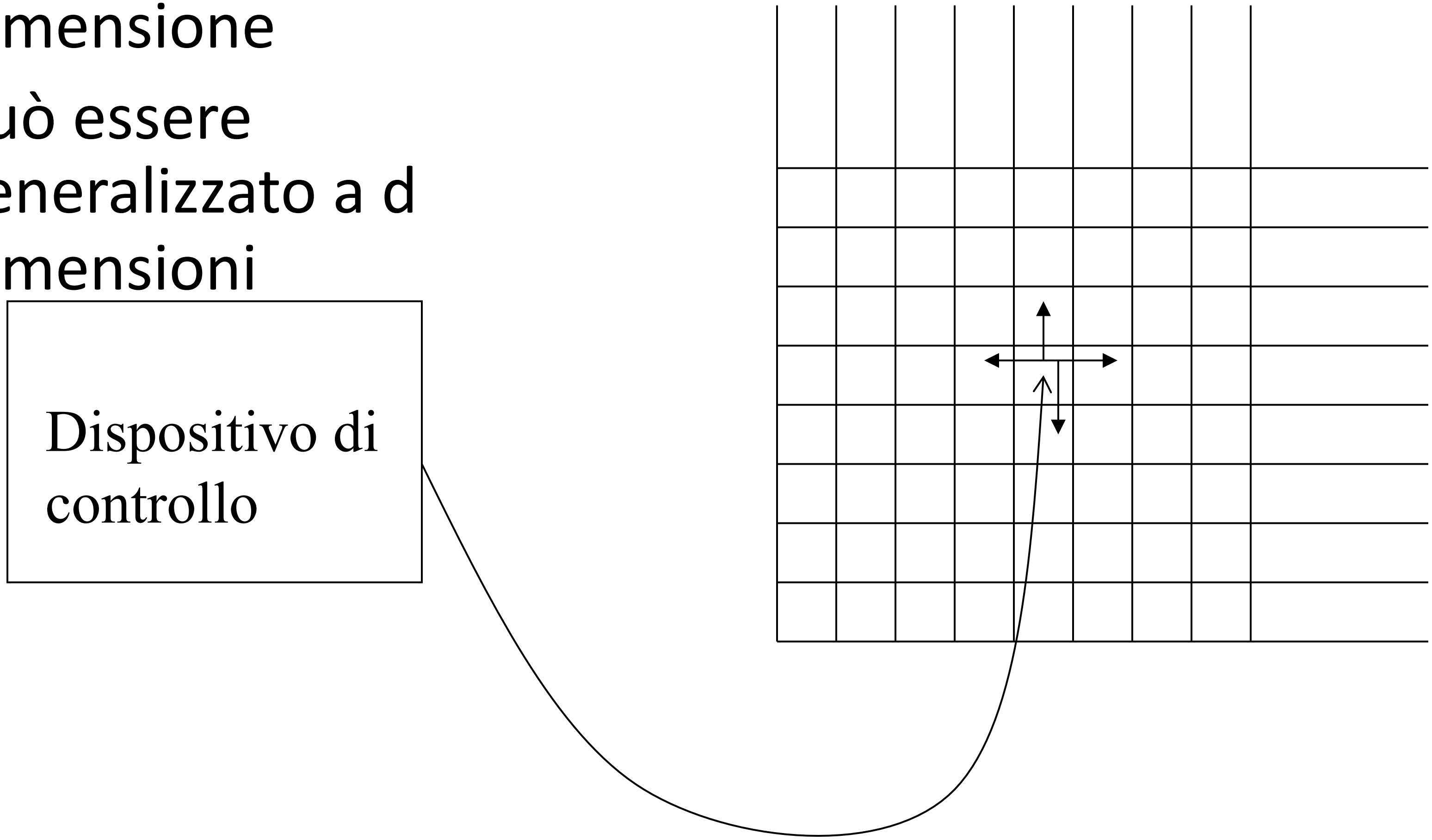



Nastro singolo

- Solitamente illimitato in ambo le direzioni
- Serve da ingresso, memoria e uscita

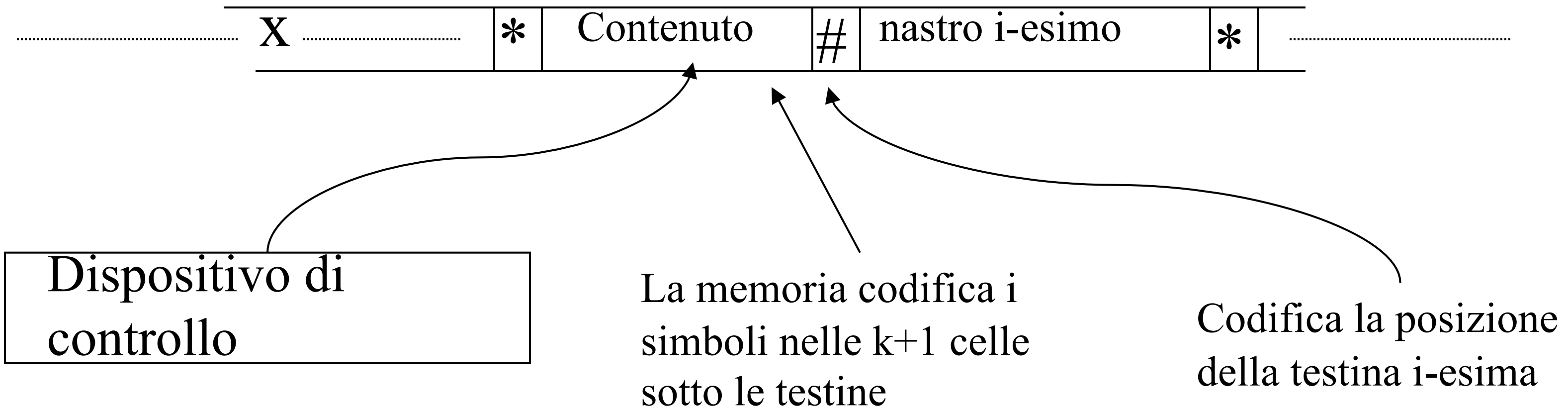
# TM con nastro bidimensionale

- Una testina per ogni dimensione
- Può essere generalizzato a d dimensioni



# Relazione tra i diversi modelli

- Sia la TM multinastro sia quella a nastro singolo possono essere dotate di nastri d-dimensionali
- Tutti questi modelli di TM sono equivalenti
  - Riconoscono la stessa classe di linguaggi



# Esempio

Progettare una TM a nastro singolo (illimitato a destra) che riconosca $L = \{a^n b^n \mid n \geq 1\}$