

# Linguaggi di Programmazione

## Docente: Cataldo Musto

Capitolo 7 – Linguaggi regolari,  
espressioni regolari e teorema di  
Kleene.

Si ringraziano il Prof. Giovanni Semeraro, il Prof. Marco de Gemmis e il Prof. Pasquale Lops per la concessione del materiale didattico di base

# Descrizione algebrica dei linguaggi lineari destri

Grammatiche generative	Automi a stati finiti accettori	
<i>Generatori</i>	<i>Riconoscitori</i>	
procedure effettive per		
<i>Costruire stringhe che appartengono al linguaggio</i>	<i>Stabilire se una stringa appartiene o no ad un linguaggio</i>	
<i>Definizioni operazionali</i>		

# Descrizione algebrica dei linguaggi lineari destri

Grammatiche generative	Automi a stati finiti accettori	Espressioni regolari
<i>Generatori</i>	<i>Riconoscitori</i>	
procedure effettive per		
<i>Costruire stringhe che appartengono al linguaggio</i>	<i>Stabilire se una stringa appartiene o no ad un linguaggio</i>	
<i>Definizioni operazionali</i>	<i>Definizione denotazionale</i>	

# Espressioni regolari

## ■ Definizione di espressione regolare

Sia  $X$  un alfabeto finito. Una stringa  $R$  di alfabeto  $X \cup \{ \lambda, +, *, ., \emptyset, (, ) \}$  (con  $X \cap \{ \lambda, +, *, ., \emptyset, (, ) \} = \emptyset$ ) è una **espressione regolare** di alfabeto  $X$  se e solo se vale una delle seguenti condizioni:

- (i)  $R = \emptyset$
- (ii)  $R = \lambda$
- (iii)  $R = a$ , per ogni  $a \in X$

# Espressioni regolari

## ■ Definizione di espressione regolare

Sia  $X$  un alfabeto finito. Una stringa  $R$  di alfabeto  $X \cup \{ \lambda, +, *, \cdot, \emptyset, (, ) \}$  (con  $X \cap \{ \lambda, +, *, \cdot, \emptyset, (, ) \} = \emptyset$ ) è una **espressione regolare** di alfabeto  $X$  se e solo se vale una delle seguenti condizioni:

- (i)  $R = \emptyset$
- (ii)  $R = \lambda$
- (iii)  $R = a$ , per ogni  $a \in X$
- (iv)  $R = (R_1 + R_2)$ , con  $R_1, R_2$  espressioni regolari di alfabeto  $X$
- (v)  $R = (R_1 \cdot R_2)$ , con  $R_1, R_2$  espressioni regolari di alfabeto  $X$
- (vi)  $R = (R_1)^*$ , con  $R_1$  espressione regolare di alfabeto  $X$

# Espressioni regolari

## ■ Esempi di espressioni regolari

- b
  - aa
  - aa+b
  - $(aa)^* + b^*$
  - $a(a+b)b$
  - $b^*a+c$
  - etc.
- (i)  $R = \emptyset$
  - (ii)  $R = \lambda$
  - (iii)  $R = a$ , per ogni  $a \in X$
  - (iv)  $R = (R_1 + R_2)$ , con  $R_1, R_2$
  - (v)  $R = (R_1 \cdot R_2)$ , con  $R_1, R_2 \in \Sigma^*$
  - (vi)  $R = (R_1)^*$ , con  $R_1$  espressa in forma regolare

# Espressioni regolari

## ■ Esempi di espressioni regolari

- b
- aa
- aa+b
- $(aa)^* + b^*$
- $a(a+b)b$
- $b^*a+c$
- etc.

- (i)  $R = \emptyset$
- (ii)  $R = \lambda$
- (iii)  $R = a$ , per ogni  $a \in X$
- (iv)  $R = (R_1 + R_2)$ , con  $R_1, R_2$
- (v)  $R = (R_1 \cdot R_2)$ , con  $R_1, R_2 \in \Sigma^*$
- (vi)  $R = (R_1)^*$ , con  $R_1$  espressa da una E.R.

## ■ Perché sono interessanti per noi?

- Perché le espressioni regolari forniscono un meccanismo per denotare (descrivere) i linguaggi

# Linguaggi regolari ed espressioni regolari

## ■ Definizione di linguaggio regolare

Sia  $X$  un alfabeto finito. Un *linguaggio*  $L \subseteq X^*$  è *regolare* se:

- $L$  è finito;

# Linguaggi regolari ed espressioni regolari

## ■ Definizione di linguaggio regolare

Sia  $X$  un alfabeto finito. Un *linguaggio*  $L \subseteq X^*$  è *regolare* se:

- $L$  è finito;

oppure

- $L$  può essere ottenuto per induzione utilizzando una delle seguenti operazioni:

- $L = L_1 \cup L_2$ , con  $L_1, L_2$  regolari;
- $L = L_1 \cdot L_2$ , con  $L_1, L_2$  regolari;
- $L = L_1^*$ , con  $L_1$  regolare.

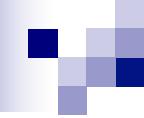
Si noti che  $\emptyset$  e  $\{\lambda\}$  sono linguaggi regolari.

Denotiamo con  $\mathcal{L}_{REG}$  la classe dei linguaggi regolari.

# Espressioni regolari e linguaggi regolari

- Ad ogni espressione regolare  $R$  corrisponde un linguaggio regolare  $S(R)$  definito nel modo seguente:

<i>Espressione regolare</i>	<i>Linguaggio regolare corrispondente</i>
$\emptyset$	$\emptyset$
$\lambda$	$\{\lambda\}$
$a$	$\{a\}$
$(R_1 + R_2)$	$S(R_1) \cup S(R_2)$
$(R_1 \cdot R_2)$	$S(R_1) \cdot S(R_2)$
$(R_1)^*$	$(S(R_1))^*$



# Espressioni regolari e Linguaggi Regolari

## ■ Convertiamo nei corrispondenti linguaggi regolari

- $R = b$   $\rightarrow S = \{b\}$
  - $R = aa$   $\rightarrow S = \{aa\}$
  - $R = aa+b$   $\rightarrow S = \{aa\} \cup \{b\}$
  - $R = (aa)^*+b^*$
  - $R = a(a+b)b$
  - $R = b^*a+c$
  - etc.



# Espressioni regolari e Linguaggi Regolari

## ■ Convertiamo nei corrispondenti linguaggi regolari

- $R = b$   $\rightarrow S = \{b\}$
  - $R = aa$   $\rightarrow S = \{aa\}$
  - $R = aa+b$   $\rightarrow S = \{aa\} \cup \{b\}$
  - $R = (aa)^*+b^*$   $\rightarrow S = \{aa\}^* \cup b^*$
  - $R = a(a+b)b$   $\rightarrow$
  - $R = b^*a+c$   $\rightarrow$
  - etc.

# Espressioni regolari e Linguaggi Regolari

## ■ Convertiamo nei corrispondenti linguaggi regolari

- |   |  |
|---|--|
| <input type="checkbox"/> $R = b$          | $\rightarrow S = \{b\}$                |
| <input type="checkbox"/> $R = aa$         | $\rightarrow S = \{aa\}$               |
| <input type="checkbox"/> $R = aa+b$       | $\rightarrow S = \{aa\} \cup \{b\}$    |
| <input type="checkbox"/> $R = (aa)^*+b^*$ | $\rightarrow S = \{aa\}^* \cup b^*$    |
| <input type="checkbox"/> $R = a(a+b)b$    | $\rightarrow S = \{aab\} \cup \{abb\}$ |
| <input type="checkbox"/> $R = b^*a+c$     | $\rightarrow S = b^*a \cup c$          |
| <input type="checkbox"/> etc.             |  |

Ogni espressione regolare denota un linguaggio

# Espressioni regolari e linguaggi regolari

- Da una espressione regolare si possono eliminare le coppie di parentesi superflue, tenuto conto che le operazioni ai punti (iv), (v) e (vi) sono elencate in ordine crescente di priorità.
  - **Iterazione:** priorità massima
  - **Concatenazione:** priorità alta
  - **Unione:** priorità più bassa
- Esempio
  - $((b^*)(a))|(c) = b^*a|c$

# Proposizione

- **Un linguaggio su  $X$  è regolare** se e solo se corrisponde ad una espressione regolare su  $X$ .  
Quindi, denotato con  $\mathcal{R}$  l'insieme delle espressioni regolari di alfabeto  $X$ , definiamo la funzione:

$$S : \mathcal{R} \rightarrow 2^{X^*}$$

che ad ogni espressione regolare  $R$  associa il corrispondente linguaggio regolare  $S(R)$ . Si ha dunque:

$$\mathcal{L}_{REG} = \left\{ L \in 2^{X^*} \mid \exists R \in \mathcal{R} : L = S(R) \right\}$$

# Esercizi Guidati - 1

$ba^*$

$S(ba^*) = S(b) \cdot S(a^*) = S(b) \cdot (S(a))^* = \{b\} \cdot \{a\}^* =$   
= insieme di tutte le parole su  $X = \{a, b\}$  che  
cominciano con una  $b$  seguita eventualmente  
soltanto da delle  $a$ .

## Esercizi Guidati - 2

$ba^*$

$S(ba^*) = S(b) \cdot S(a^*) = S(b) \cdot (S(a))^* = \{b\} \cdot \{a\}^* =$   
= insieme di tutte le parole su  $X = \{a, b\}$  che  
cominciano con una  $b$  seguita eventualmente  
soltanto da delle  $a$ .

$a^* \cdot b \cdot a^* \cdot b \cdot a^*$

$S(a^* \cdot b \cdot a^* \cdot b \cdot a^*) = S(a^*) \cdot S(b) \cdot S(a^*) \cdot S(b) \cdot S(a^*) =$   
 $= (S(a))^* \cdot S(b) \cdot (S(a))^* \cdot S(b) \cdot (S(a))^* =$   
 $= \{a\}^* \cdot \{b\} \cdot \{a\}^* \cdot \{b\} \cdot \{a\}^* =$

# Esempi

$ba^*$

$S(ba^*) = S(b) \cdot S(a^*) = S(b) \cdot (S(a))^* = \{b\} \cdot \{a\}^* =$   
= insieme di tutte le parole su  $X = \{a, b\}$  che  
cominciano con una  $b$  seguita eventualmente  
soltanto da delle  $a$ .

$a^* \cdot b \cdot a^* \cdot b \cdot a^*$

$S(a^* \cdot b \cdot a^* \cdot b \cdot a^*) = S(a^*) \cdot S(b) \cdot S(a^*) \cdot S(b) \cdot S(a^*) =$   
 $= (S(a))^* \cdot S(b) \cdot (S(a))^* \cdot S(b) \cdot (S(a))^* =$   
 $= \{a\}^* \cdot \{b\} \cdot \{a\}^* \cdot \{b\} \cdot \{a\}^* =$   
= insieme di tutte le parole su  $X = \{a, b\}$   
che contengono esattamente due  $b$ .

# Esempi

$$(a+b)^* \quad S((a+b)^*) = (S(a+b))^* = (S(a) \cup S(b))^* = (\{a\} \cup \{b\})^*$$

# Esempi

$$(a+b)^* \quad S((a+b)^*) = (S(a+b))^* = (S(a) \cup S(b))^* = (\{a\} \cup \{b\})^* \\ = \{a,b\}^* = \text{insieme di tutte le parole su } X = \{a,b\}.$$

$$(b+abb)^* \quad S((b+abb)^*) = (S(b+abb))^* = (S(b) \cup S(abb))^* =$$

# Esempi

$$(a+b)^* \quad S((a+b)^*) = (S(a+b))^* = (S(a) \cup S(b))^* = (\{a\} \cup \{b\})^* \\ = \{a,b\}^* = \text{insieme di tutte le parole su } X = \{a,b\}.$$

$$(b+abb)^* \quad S((b+abb)^*) = (S(b+abb))^* = (S(b) \cup S(abb))^* = \\ = (S(b) \cup (S(a) \cdot S(b) \cdot S(b)))^* = \\ = (\{b\} \cup (\{a\} \cdot \{b\} \cdot \{b\}))^* = \\ = (\{b\} \cup \{abb\})^* = \{b,abb\}^* =$$

# Esempi

$$(a+b)^* \quad S((a+b)^*) = (S(a+b))^* = (S(a) \cup S(b))^* = (\{a\} \cup \{b\})^* \\ = \{a,b\}^* = \text{insieme di tutte le parole su } X = \{a,b\}.$$

$$(b+abb)^* \quad S((b+abb)^*) = (S(b+abb))^* = (S(b) \cup S(abb))^* = \\ = (S(b) \cup (S(a) \cdot S(b) \cdot S(b)))^* = \\ = (\{b\} \cup (\{a\} \cdot \{b\} \cdot \{b\}))^* = \\ = (\{b\} \cup \{abb\})^* = \{b, abb\}^* =$$

= insieme di tutte le parole su  $X = \{a,b\}$  in cui  
ogni  $a$  è immediatamente seguita da  
almeno due  $b$ .

# Osservazione

- $S : \mathfrak{R} \rightarrow 2^{X^*}$  non è una funzione iniettiva.

# Osservazione

- $S : \mathfrak{R} \rightarrow 2^{X^*}$  non è una funzione iniettiva.
- Un linguaggio può essere denotato **da più espressioni regolari**.

# Esempio

- Il linguaggio costituito da tutte le parole su  $X = \{a, b\}$  con  $a$  e  $b$  che si alternano (e che cominciano e terminano con  $b$ ) può essere descritto sia dall'espressione regolare

$$b \cdot (ab)^*$$

sia da

$$(ba)^* b$$

## Definizione di espressioni regolari equivalenti

- Due espressioni regolari  $R_1$  e  $R_2$  su  $X$  sono *equivalenti* (per abuso di notazione, scriviamo  $R_1 = R_2$ ) se e solo se  $S(R_1) = S(R_2)$ .

Relativamente all'esempio precedente, si ha, pertanto:

$$b \cdot (ab)^* = (ba)^* b$$

# Proprietà delle espressioni regolari

- Siano  $R_1$ ,  $R_2$  ed  $R_3$  espressioni regolari di alfabeto  $X$ , risulta:
  1.  $(R_1 + R_2) + R_3 = R_1 + (R_2 + R_3) = R_1 + R_2 + R_3$  Prop. associativa
  2.  $R_1 + R_2 = R_2 + R_1$  Prop. commutativa
  3.  $R_1 + \emptyset = \emptyset + R_1 = R_1$   $\emptyset$  è l'elemento neutro rispetto all'operazione “+” definita sulle espressioni regolari

# Proprietà delle espressioni regolari

- Siano  $R_1$ ,  $R_2$  ed  $R_3$  espressioni regolari di alfabeto  $X$ , risulta:
  1.  $(R_1 + R_2) + R_3 = R_1 + (R_2 + R_3) = R_1 + R_2 + R_3$  Prop. associativa
  2.  $R_1 + R_2 = R_2 + R_1$  Prop. commutativa
  3.  $R_1 + \emptyset = \emptyset + R_1 = R_1$   $\emptyset$  è l'elemento neutro rispetto all'operazione “+” definita sulle espressioni regolari
  4.  $R_1 + R_1 = R_1$  Idempotenza
  5.  $(R_1 \cdot R_2) \cdot R_3 = R_1 \cdot (R_2 \cdot R_3) = R_1 \cdot R_2 \cdot R_3$  Prop. associativa
  6.  $R_1 \cdot R_2 \neq R_2 \cdot R_1$  In generale
  7.  $R_1 \cdot \lambda = \lambda \cdot R_1 = R_1$   $\lambda$  è l'elemento neutro rispetto all'operazione “.” definita sulle espressioni regolari

# Proprietà delle espressioni regolari

- Siano  $R_1$ ,  $R_2$  ed  $R_3$  espressioni regolari di alfabeto  $X$ , risulta:
  8.  $R_1 \cdot \emptyset = \emptyset \cdot R_1 = \emptyset$   $\emptyset$  è l'elemento assorbente rispetto all'operazione “.”.
  9.  $R_1 \cdot (R_2 + R_3) = (R_1 \cdot R_2) + (R_1 \cdot R_3)$  Proprietà distributiva di “.” rispetto all'operazione “+” (distributività sinistra).

# Proprietà delle espressioni regolari

- Siano  $R_1$ ,  $R_2$  ed  $R_3$  espressioni regolari di alfabeto  $X$ , risulta:
  8.  $R_1 \cdot \emptyset = \emptyset \cdot R_1 = \emptyset$   $\emptyset$  è l'elemento assorbente rispetto all'operazione “.”.
  9.  $R_1 \cdot (R_2 + R_3) = (R_1 \cdot R_2) + (R_1 \cdot R_3)$  Proprietà distributiva di “.” rispetto all'operazione “+” (distributività sinistra).
  10.  $(R_1 + R_2) \cdot R_3 = (R_1 \cdot R_3) + (R_2 \cdot R_3)$  Proprietà distributiva di “.” rispetto all'operazione “+” (distributività destra).
  11.  $(R_1)^* = (R_1)^* \cdot (R_1)^* = ((R_1)^*)^* = (\lambda + R_1)^*$
  12.  $(\emptyset)^* = (\lambda)^* = \lambda$

# Proprietà delle espressioni regolari

- Siano  $R_1$ ,  $R_2$  ed  $R_3$  espressioni regolari di alfabeto  $X$ , risulta:

$$13. \quad (R_1)^* = \lambda + R_1 \cdot R_1^* = \lambda + R_1^* \cdot R_1$$

$$\begin{aligned} 14. \quad (R_1 + R_2)^* &= (R_1^* + R_2^*)^* = (R_1^* \cdot R_2^*)^* = \\ &= (R_1^* \cdot R_2)^* \cdot R_1^* = R_1^* \cdot (R_2 \cdot R_1^*)^* \end{aligned}$$

$$15. \quad (R_1 + R_2)^* \neq R_1^* + R_2^* \quad \text{In generale}$$

$$16. \quad R_1^* \cdot R_1 = R_1 \cdot R_1^*$$

$$17. \quad R_1 \cdot (R_2 \cdot R_1)^* = (R_1 \cdot R_2)^* \cdot R_1$$

# Proprietà delle espressioni regolari

- Siano  $R_1$ ,  $R_2$  ed  $R_3$  espressioni regolari di alfabeto  $X$ , risulta:

$$18. \left( R_1^* \cdot R_2 \right)^* = \lambda + (R_1 + R_2)^* \cdot R_2$$

$$19. \left( R_1 \cdot R_2^* \right)^* = \lambda + R_1 \cdot (R_1 + R_2)^*$$

- 20. Supponiamo che:  $\lambda \notin S(R_2)$

$$R_1 = R_2 \cdot R_1 + R_3 \text{ se e solo se } R_1 = R_2^* \cdot R_3$$

$$R_1 = R_1 \cdot R_2 + R_3 \text{ se e solo se } R_1 = R_3 \cdot R_2^*$$

# Dimostrazione proprietà espressioni regolari

## ■ Per esercizio

- Aiuto: le 1) - 5) e le 7) - 14) si dimostrano ricorrendo alla funzione  $S$ ;
- comunque la maggior parte delle 1) - 20) può essere provata con una tecnica generale, detta *dimostrazione mediante riparsificazione*.

# Dimostrazione mediante riparsificazione

- Illustriamo questa tecnica dimostrando la proprietà 17)

$$R_1 \cdot (R_2 \cdot R_1)^* = (R_1 \cdot R_2)^* \cdot R_1$$

Si consideri una qualunque parola:

$$w \in S(R_1 \cdot (R_2 \cdot R_1)^*), \quad w = r_1^0 (r_2^1 \cdot r_1^1) \cdot (r_2^2 \cdot r_1^2) \cdot \dots \cdot (r_2^n \cdot r_1^n), \quad n \geq 0$$

- ove:

$$r_1^i \in S(R_1), \quad i = 0, 1, 2, \dots, n$$

$$r_2^j \in S(R_2), \quad j = 0, 1, 2, \dots, n$$

# Dimostrazione mediante riparsificazione

- Illustriamo questa tecnica dimostrando la proprietà 17)

$$R_1 \cdot (R_2 \cdot R_1)^* = (R_1 \cdot R_2)^* \cdot R_1$$

Si consideri una qualunque parola:

$$w \in S(R_1 \cdot (R_2 \cdot R_1)^*), \quad w = r_1^0 (r_2^1 \cdot r_1^1) \cdot (r_2^2 \cdot r_1^2) \cdot \dots \cdot (r_2^n \cdot r_1^n), \quad n \geq 0$$

*Riparsificando w ed utilizzando la proprietà associativa della concatenazione, si ha:*  $w = (r_1^0 \cdot r_2^1) \cdot (r_1^1 \cdot r_2^2) \cdot \dots \cdot (r_1^{n-1} \cdot r_2^n) \cdot r_1^n$

- dunque:  $w \in S((R_1 \cdot R_2)^* \cdot R_1)$
- da cui:  $S(R_1 \cdot (R_2 \cdot R_1)^*) \subseteq S((R_1 \cdot R_2)^* \cdot R_1)$
- In modo analogo si dimostra:  $S(R_1 \cdot (R_2 \cdot R_1)^*) \supseteq S((R_1 \cdot R_2)^* \cdot R_1)$  35/169

# Dimostrazione proprietà espressioni regolari

- Un'altra tecnica comune per dimostrare tali proprietà è semplicemente quella di utilizzare proprietà già note.
- Mostreremo ora come si usano le proprietà delle espressioni regolari per provare l'equivalenza di espressioni regolari.

# Esercizio

■ Dimostrare la seguente equivalenza:

$$(b + aa^*b) + (b + aa^*b) \cdot (a + ba^*b)^* \cdot (a + ba^*b) = a^*b \cdot (a + ba^*b)^*$$

$$(b + aa^*b) + (b + aa^*b) \cdot (a + ba^*b)^* \cdot (a + ba^*b) =$$

9)

## Proprietà 9

$$R1(R2+R3) = (R1^*R2 + R1^*R3)$$

## Suggerimento

$$R2 = \lambda$$

## Suggerimento 2

E' come se stessimo mettendo  
R1 a fattor comune

# Esercizio

■ Dimostrare la seguente equivalenza:

$$(b + aa^*b) + (b + aa^*b) \cdot (a + ba^*b)^* \cdot (a + ba^*b) = a^*b \cdot (a + ba^*b)^*$$

$$(b + aa^*b) + (b + aa^*b) \cdot (a + ba^*b)^* \cdot (a + ba^*b) \stackrel{9)}{=} (b + aa^*b) \cdot [ \lambda + (a + ba^*b)^* (a + ba^*b) ] =$$

## Proprietà 9

$$R1(R2+R3) = (R1^*R2 + R1^*R3)$$

## Suggerimento

$$R2 = \lambda$$

## Suggerimento 2

E' come se stessimo mettendo  
R1 a fattor comune

# Esercizio

■ Dimostrare la seguente equivalenza:

$$(b + aa^*b) + (b + aa^*b) \cdot (a + ba^*b)^* \cdot (a + ba^*b) = a^*b \cdot (a + ba^*b)^*$$

$$(b + aa^*b) + (b + aa^*b) \cdot (a + ba^*b)^* \cdot (a + ba^*b) \stackrel{9)}{=} (b + aa^*b) \cdot [\lambda + (a + ba^*b)^* (a + ba^*b)] \stackrel{13)}{=}$$

**Proprietà 13**

$$(R1)^* = (\lambda + R1^* + R1)$$

# Esercizio

■ Dimostrare la seguente equivalenza:

$$(b + aa^*b) + (b + aa^*b) \cdot (a + ba^*b)^* \cdot (a + ba^*b) = a^*b \cdot (a + ba^*b)^*$$

$$\begin{aligned} (b + aa^*b) + (b + aa^*b) \cdot (a + ba^*b)^* \cdot (a + ba^*b) &\stackrel{9)}{=} (b + aa^*b) \cdot [\lambda + (a + ba^*b)^*(a + ba^*b)] = \\ &\stackrel{13)}{=} (b + aa^*b) \cdot (a + ba^*b)^* = \end{aligned}$$

**Proprietà 13**

$$(R1)^* = (\lambda + R1^* + R1)$$

# Esercizio

■ Dimostrare la seguente equivalenza:

$$(b + aa^*b) + (b + aa^*b) \cdot (a + ba^*b)^* \cdot (a + ba^*b) = a^*b \cdot (a + ba^*b)^*$$

$$\begin{aligned} (b + aa^*b) + (b + aa^*b) \cdot (a + ba^*b)^* \cdot (a + ba^*b) &\stackrel{9)}{=} (b + aa^*b) \cdot [ \lambda + (a + ba^*b)^* (a + ba^*b) ] = \\ &\stackrel{13)}{=} (b + aa^*b) \cdot (a + ba^*b)^* = \\ &\stackrel{10)}{=} \dots \end{aligned}$$

## Proprietà 10

$$(R1+R2)R3 = (R1^*R3 + R2^*R3)$$

## Suggerimento

$$R1 = \lambda$$

## Suggerimento 2

E' come se stessimo mettendo  
R3 a fattor comune

# Esercizio

■ Dimostrare la seguente equivalenza:

$$(b + aa^*b) + (b + aa^*b) \cdot (a + ba^*b)^* \cdot (a + ba^*b) = a^*b \cdot (a + ba^*b)^*$$

$$\begin{aligned} (b + aa^*b) + (b + aa^*b) \cdot (a + ba^*b)^* \cdot (a + ba^*b) &\stackrel{9)}{=} (b + aa^*b) \cdot [\lambda + (a + ba^*b)^*(a + ba^*b)] = \\ &\stackrel{13)}{=} (b + aa^*b) \cdot (a + ba^*b)^* = \\ &\stackrel{10)}{=} (\lambda + aa^*) \cdot b \cdot (a + ba^*b)^* = \\ &\dots \end{aligned}$$

**Proprietà 10**

$$(R1+R2)R3 = (R1^*R3 + R2^*R3)$$

**Suggerimento**

$$R1 = \lambda$$

**Suggerimento 2**

E' come se stessimo mettendo  
R3 a fattor comune

# Esercizio

■ Dimostrare la seguente equivalenza:

$$(b + aa^*b) + (b + aa^*b) \cdot (a + ba^*b)^* \cdot (a + ba^*b) = a^*b \cdot (a + ba^*b)^*$$

$$\begin{aligned} (b + aa^*b) + (b + aa^*b) \cdot (a + ba^*b)^* \cdot (a + ba^*b) &\stackrel{9)}{=} (b + aa^*b) \cdot [\lambda + (a + ba^*b)^* (a + ba^*b)] = \\ &\stackrel{13)}{=} (b + aa^*b) \cdot (a + ba^*b)^* = \\ &\stackrel{10)}{=} (\lambda + aa^*) \cdot b \cdot (a + ba^*b)^* = \\ &\stackrel{13)}{=} \end{aligned}$$

**Proprietà 13**

$$(R1)^* = (\lambda + R1^* + R1)$$

# Esercizio

■ Dimostrare la seguente equivalenza:

$$(b + aa^*b) + (b + aa^*b) \cdot (a + ba^*b)^* \cdot (a + ba^*b) = a^*b \cdot (a + ba^*b)^*$$

$$\begin{aligned}(b + aa^*b) + (b + aa^*b) \cdot (a + ba^*b)^* \cdot (a + ba^*b) &\stackrel{9)}{=} (b + aa^*b) \cdot [\lambda + (a + ba^*b)^*(a + ba^*b)] = \\ &\stackrel{13)}{=} (b + aa^*b) \cdot (a + ba^*b)^* = \\ &\stackrel{10)}{=} (\lambda + aa^*) \cdot b \cdot (a + ba^*b)^* = \\ &\stackrel{13)}{=} a^*b \cdot (a + ba^*b)^*\end{aligned}$$

## Proprietà 13

$$(R1)^* = (\lambda + R1^* + R1)$$

C.V.D.

Questo è un modo alternativo che si può applicare per dimostrare che  
Due linguaggi sono equivalenti

# Teorema di Kleene

- $\mathcal{L}_3 \equiv \mathcal{L}_{FSL} \equiv \mathcal{L}_{REG}$

# Teorema di Kleene

- $\mathcal{L}_3 \equiv \mathcal{L}_{FSL} \equiv \mathcal{L}_{REG}$

## Dimostrazione

Lo schema della dimostrazione è il seguente:

- $\square \mathcal{L}_3 \subset \mathcal{L}_{FSL}$   $(\mathcal{L}_{FSL} \subset \mathcal{L}_3)$
- $\square \mathcal{L}_{FSL} \subset \mathcal{L}_{REG}$
- $\square \mathcal{L}_{REG} \subset \mathcal{L}_3$

# Teorema di Kleene

- $\mathcal{L}_3 \equiv \mathcal{L}_{FSL} \equiv \mathcal{L}_{REG}$

## Dimostrazione

Lo schema della dimostrazione è il seguente:

- $\mathcal{L}_3 \subset \mathcal{L}_{FSL}$   $(\mathcal{L}_{FSL} \subset \mathcal{L}_3)$
- $\mathcal{L}_{FSL} \subset \mathcal{L}_{REG}$
- $\mathcal{L}_{REG} \subset \mathcal{L}_3$
- Dunque dimostriamo che:
  - Ogni linguaggio di tipo tre può essere riconosciuto da un automa
  - Ogni automa a stati finiti può essere denotato da una espressione regolare
  - Il linguaggio denotato da ogni espressione regolare può essere generato da una grammatica di tipo tre

# Dimostrazione teorema di Kleene

- $\mathcal{L}_3 \subset \mathcal{L}_{FSL}$
- Sia  $L \in \mathcal{L}_3 \overset{def}{\Leftrightarrow} \exists G = (X, V, S, P)$ ,  $G$  di tipo '3':  $L(G) = L$ .

# Dimostrazione teorema di Kleene

- $\mathcal{L}_3 \subset \mathcal{L}_{FSL}$
- Sia  $L \in \mathcal{L}_3 \overset{def}{\iff} \exists G = (X, V, S, P)$ ,  $G$  di tipo ‘3’:  $L(G) = L$ .
  - (1)  $A \rightarrow bC$  con  $A, C \in V$  e  $b \in X$ ;
  - (2)  $A \rightarrow b$  con  $A \in V$  e  $b \in X \cup \{\lambda\}$ .
- Vogliamo costruire un automa a stati finiti  
 **$M = (Q, \delta, q_0, F)$  tale che  $T(M) = L(G)$ .**

Allo scopo si fornisce il seguente algoritmo per la costruzione di un automa a stati finiti non deterministico che riconosce il linguaggio generato da una fissata grammatica lineare destra.

## Algoritmo: Costruzione di un automa a stati finiti non deterministico equivalente ad una grammatica lineare destra

- Data una grammatica lineare destra:

$$G = (X, V, S, P)$$

l'automa accettore a stati finiti equivalente ( $T(M) = L(G)$ ) viene costruito come segue:

$$M = (Q, \delta, q_0, F)$$

- (I)  $X$  come alfabeto di ingresso;
- (II)  $Q = V \cup \{q\}$ ,  $q \notin V$
- (III)  $q_0 = S$
- (IV)  $F = \{q\} \cup \{B \mid B \rightarrow \lambda \in P\}$
- (V)  $\delta : Q \times X \rightarrow 2^Q$      $\exists' V.a \quad \forall B \rightarrow aC \in P, C \in \delta(B, a)$   
 $V.b \quad \forall B \rightarrow a \in P, q \in \delta(B, a)$

## Esempio semplicissimo

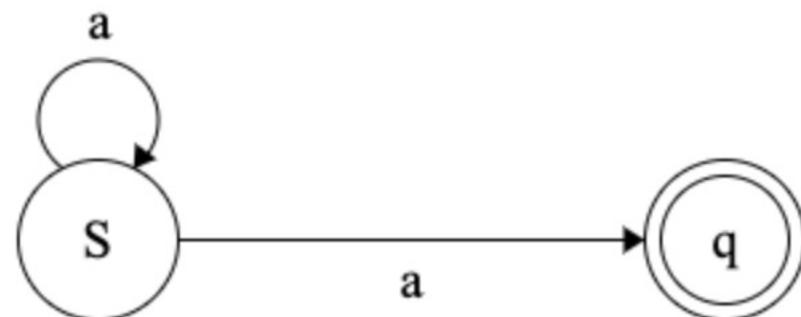
- Dato il seguente linguaggio
  - $L = \{w \in X^* | w = a^n, n > 0\}$
- Riconosciuto dalla seguente grammatica

## Esempio semplicissimo

- Dato il seguente linguaggio
  - $L = \{w \in X^* | w = a^n, n > 0\}$
- Riconosciuto dalla seguente grammatica
  - $G = (X, V, S, P)$ ,  $X = \{a\}$ ,  $V = \{S\}$ , ... etc
  - $P = \{S \rightarrow a \mid aS\}$
- Applicando l'algoritmo può essere trasformato nel seguente automa

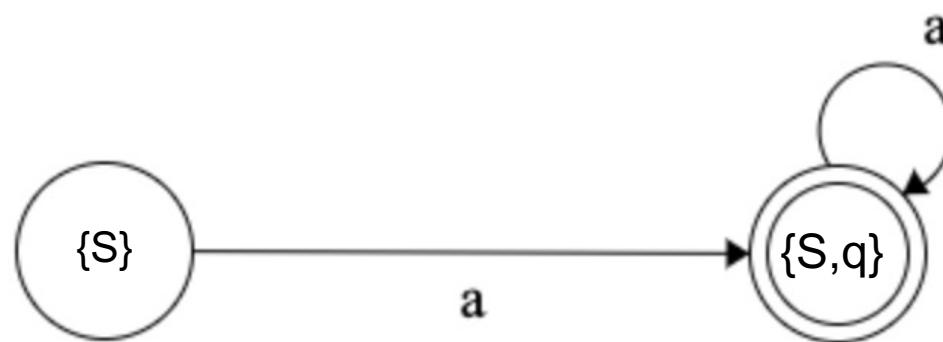
## Esempio semplicissimo

- Dato il seguente linguaggio
  - $L = \{w \in X^* | w = a^n, n > 0\}$
- Riconosciuto dalla seguente grammatica
  - $G = (X, V, S, P)$ ,  $X = \{a\}$ ,  $V = \{S\}$ , ... etc
  - $P = \{S \rightarrow a \mid aS\}$
- Applicando l'algoritmo può essere trasformato nel seguente automa



## Esempio semplicissimo

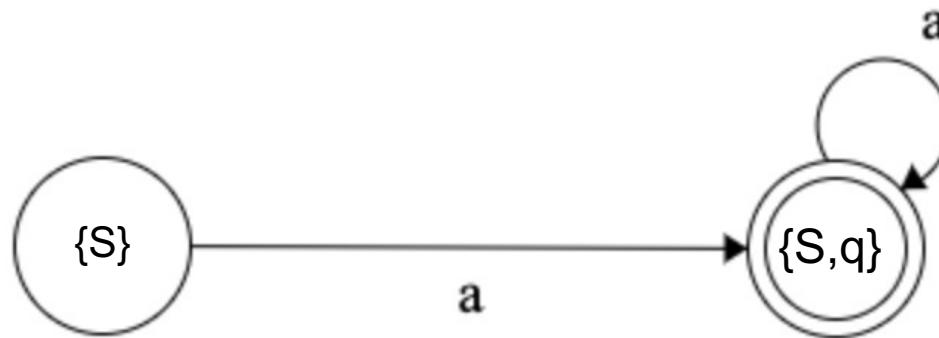
- Dato il seguente linguaggio
  - $L = \{w \in X^* | w = a^n, n > 0\}$
- Riconosciuto dalla seguente grammatica
  - $G = (X, V, S, P)$ ,  $X = \{a\}$ ,  $V = \{S\}$ , ... etc
  - $P = \{S \rightarrow a \mid aS\}$
- Che può a sua volta essere convertito nell'automa deterministico



## Esempio semplicissimo

- Dato il seguente linguaggio
  - $L = \{w \in X^* | w = a^n, n > 0\}$
- Riconosciuto dalla seguente grammatica
  - $G = (X, V, S, P)$ ,  $X = \{a\}$ ,  $V = \{S\}$ , ... etc
  - $P = \{S \rightarrow a \mid aS\}$
- Che può a sua volta essere convertito nell'automa deterministico

Si può facilmente verificare empiricamente che  $L(G) = T(M)$



## Algoritmo: Costruzione di un automa a stati finiti non deterministico equivalente ad una grammatica lineare destra

- L'algoritmo può generare un automa *non deterministic*o per effetto dei passi V.a. e V.b. Si può facilmente constatare che, se:  $w = x_1x_2\dots x_n \in L(G)$   
 $w$  può essere generata da una derivazione del tipo:

$$S \Rightarrow x_1X_2 \Rightarrow x_1x_2X_3 \Rightarrow \dots \Rightarrow x_1x_2\dots x_{i-1}X_i \Rightarrow x_1x_2\dots x_n$$

## Algoritmo: Costruzione di un automa a stati finiti non deterministico equivalente ad una grammatica lineare destra

- L'algoritmo può generare un automa *non deterministico* per effetto dei passi V.a. e V.b. Si può facilmente constatare che, se:  $w = x_1x_2\dots x_n \in L(G)$   
 $w$  può essere generata da una derivazione del tipo:  
$$S \Rightarrow x_1X_2 \Rightarrow x_1x_2X_3 \Rightarrow \dots \Rightarrow x_1x_2\dots x_{i-1}X_i \Rightarrow x_1x_2\dots x_n$$
- Dalla definizione data, l'automa  $M$ , esaminando la stringa  $w = x_1x_2\dots x_n$  compie una serie di mosse (o transizioni) che lo portano dallo stato  $S$  ad  $X_2, X_3, \dots, X_i$  e  $q$ ; pertanto  $L(G) \subseteq T(M)$ .

## Algoritmo: Costruzione di un automa a stati finiti non deterministico equivalente ad una grammatica lineare destra

- In modo del tutto analogo, ogni  $w$  in  $T(M)$  comporta una sequenza di mosse dell'automa a cui corrisponde una derivazione in  $G$ , e pertanto  $T(M) \subseteq L(G)$ .  
Se ne deduce che:  $L(G) = T(M)$  *c.v.d.*

## Algoritmo: Costruzione di un automa a stati finiti non deterministico equivalente ad una grammatica lineare destra

- Sebbene non sia strettamente necessario per la dimostrazione del Teorema di Kleene, per il suo interesse pratico si riporta di seguito l'algoritmo per **la costruzione di una grammatica lineare destra che genera il linguaggio accettato da un automa a stati finiti.**
- Tale algoritmo costituisce una dimostrazione costruttiva del seguente risultato:

$$\mathcal{L}_{FSL} \subset \mathcal{L}_3$$

## Algoritmo: Costruzione di una grammatica lineare destra equivalente ad un automa accettore a stati finiti

- Sia dato un automa accettore a stati finiti:

$$M = (Q, \delta, q_0, F)$$

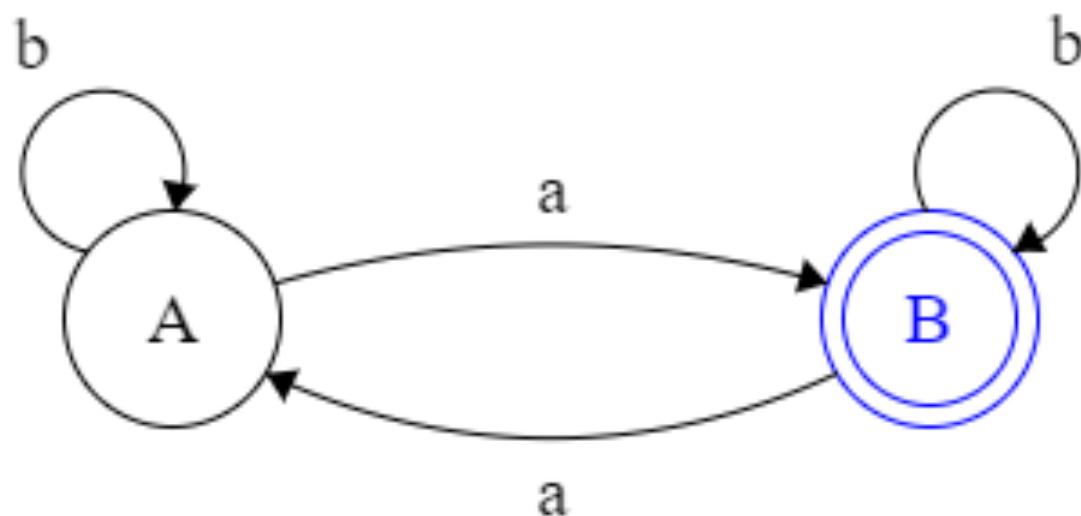
con alfabeto di ingresso  $X$ .

**La grammatica lineare destra  $G=(X,V,S,P)$  equivalente a  $M$ , ossia tale che  $L(G) = T(M)$ , si costruisce come segue:**

- (I)  $X = \text{alfabeto di ingresso di } M$
- (II)  $V = Q;$
- (III)  $S = q_0;$
- (IV)  $P = \{q \rightarrow xq' \mid q' \in \delta(q, x)\} \cup \{q \rightarrow x \mid \delta(q, x) \in F\} \cup \{q_0 \rightarrow \lambda \mid q_0 \in F\}$

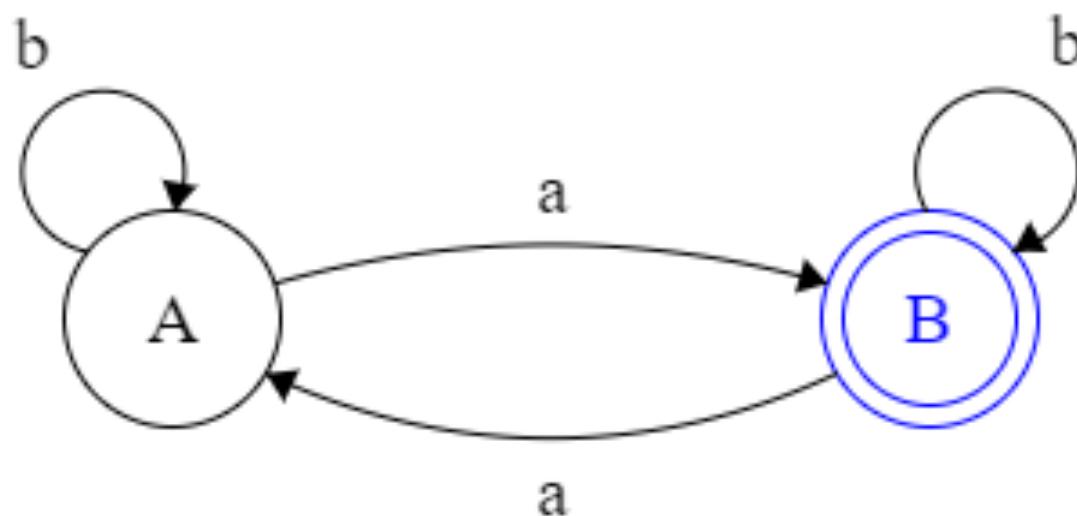
## Esempio Semplicissimo

- Creare una grammatica lineare destra equivalente all'automa a stati finiti



## Esempio Semplicissimo

- Creare una grammatica lineare destra equivalente all'automa a stati finiti



$$\begin{aligned}A &\rightarrow bA \mid a \mid aB \\B &\rightarrow aA \mid b \mid bB\end{aligned}$$

# Teorema di Kleene – Nota conclusiva

- $\mathcal{L}_3 \equiv \mathcal{L}_{FSL} \equiv \mathcal{L}_{REG}$

## Dimostrazione

Lo schema della dimostrazione è il seguente:

- $\mathcal{L}_3 \subset \mathcal{L}_{FSL}$   $(\mathcal{L}_{FSL} \subset \mathcal{L}_3)$
- $\mathcal{L}_{FSL} \subset \mathcal{L}_{REG}$
- $\mathcal{L}_{REG} \subset \mathcal{L}_3$
- Le altre inclusioni non sono dimostrate formalmente. I **meccanismi di conversione saranno però utilizzati negli esercizi.**

# Esercizio 7.1

$$\mathcal{L}_{REG} \subset \mathcal{L}_3$$

## Esercizio 7.1

Determinare una grammatica lineare destra che genera il linguaggio descritto dalla seguente espressione regolare:

$$b^* + (ab)^*$$

# Esercizio 7.1

## Esercizio 7.1

Determinare una grammatica lineare destra che genera il linguaggio descritto dalla seguente espressione regolare:

$$b^* + (ab)^*$$

Il linguaggio relativo all'espressione regolare è:

$$\begin{aligned} S(b^* + (ab)^*) &= S(b^*) \cup S((ab)^*) = (S(b))^* \cup (S(ab))^* = \{b\}^* \cup (S(a) \cdot S(b))^* = \\ &= \{b\}^* \cup (\{a\} \cdot \{b\})^* = \{b\}^* \cup \{ab\}^* \end{aligned}$$

# Esercizio 7.1

Il linguaggio relativo all'espressione regolare è:

$$\begin{aligned} S(b^* + (ab)^*) &= S(b^*) \cup S((ab)^*) = (S(b))^* \cup (S(ab))^* = \{b\}^* \cup (S(a) \cdot S(b))^* = \\ &= \{b\}^* \cup (\{a\} \cdot \{b\})^* = \{b\}^* \cup \{ab\}^* \end{aligned}$$

A questo punto, risolviamo **l'esercizio costruendo prima le due grammatiche** che generano i linguaggi denotati delle singole espressioni regolari e poi **costruendo la loro unione**

# Esercizio 7.1

Il linguaggio relativo all'espressione regolare è:

$$\begin{aligned} S(b^* + (ab)^*) &= S(b^*) \cup S((ab)^*) = (S(b))^* \cup (S(ab))^* = \{b\}^* \cup (S(a) \cdot S(b))^* = \\ &= \{b\}^* \cup (\{a\} \cdot \{b\})^* = \{b\}^* \cup \{ab\}^* \end{aligned}$$

Costruiamo dapprima la grammatica  $G_1$  tale che

$$L(G_1) = S(b^*) = \{b\}^*$$

$$G_1 = (X_1, V_1, S_1, P_1)$$

ove:

$$X_1 = \{b\} \quad V_1 = \{S_1\} \quad P_1 = \{S_1 \rightarrow bS_1 \mid \lambda\}.$$

# Esercizio 7.1

Il linguaggio relativo all'espressione regolare è:

$$\begin{aligned} S(b^* + (ab)^*) &= S(b^*) \cup S((ab)^*) = (S(b))^* \cup (S(ab))^* = \{b\}^* \cup (S(a) \cdot S(b))^* = \\ &= \{b\}^* \cup (\{a\} \cdot \{b\})^* = \{b\}^* \cup \{ab\}^* \end{aligned}$$

Costruiamo ora la grammatica  $G_2$  tale che

$$L(G_2) = S(ab) = \{ab\}$$

$$G_2 = (X_2, V_2, S_2, P_2)$$

ove:

$$X_2 = \{a, b\} \quad V_2 = \{S_2, B_2\} \quad P_2 = \{S_2 \rightarrow aB_2, B_2 \rightarrow b\} .$$

# Esercizio 7.1

Il linguaggio relativo all'espressione regolare è:

$$\begin{aligned} S(b^* + (ab)^*) &= S(b^*) \cup S((ab)^*) = (S(b))^* \cup (S(ab))^* = \{b\}^* \cup (S(a) \cdot S(b))^* = \\ &= \{b\}^* \cup (\{a\} \cdot \{b\})^* = \{b\}^* \cup \{ab\}^* \end{aligned}$$

La grammatica  $G_3$  tale che

$$L(G_3) = S((ab)^*) = \{ab\}^*$$

è dunque (si veda Capitolo 5, Tavola 1):

$$G_3 = (X_3, V_3, S_3, P_3)$$

ove:

$$\begin{aligned} X_3 &= X_2 = \{a, b\} & V_3 &= V_2 \cup \{S_3\} \\ P_3 &= \{S_3 \rightarrow \lambda\} \cup (P_2 - \{S_2 \rightarrow \lambda\}) \cup \{S_3 \rightarrow w \mid S_2 \rightarrow w \in P_2\} \cup \\ &\quad \cup \{A \rightarrow bS_3 \mid A \rightarrow b \in P_2\} = \\ &= \{S_3 \rightarrow \lambda\} \cup P_2 \cup \{S_3 \rightarrow aB_2\} \cup \{B_2 \rightarrow bS_3\} = \\ &= \{S_3 \rightarrow \lambda, S_2 \rightarrow aB_2, B_2 \rightarrow b, S_3 \rightarrow aB_2, B_2 \rightarrow bS_3\} = \\ &= \{S_3 \rightarrow \lambda \mid aB_2, S_2 \rightarrow aB_2, B_2 \rightarrow b \mid bS_3\}. \end{aligned}$$

# Esercizio 7.1

Il linguaggio relativo all'espressione regolare è:

$$\begin{aligned} S(b^* + (ab)^*) &= S(b^*) \cup S((ab)^*) = (S(b))^* \cup (S(ab))^* = \{b\}^* \cup (S(a) \cdot S(b))^* = \\ &= \{b\}^* \cup (\{a\} \cdot \{b\})^* = \{b\}^* \cup \{ab\}^* \end{aligned}$$

Si osservi che il nonterminale  $S_2$  non compare nella parte destra di nessuna produzione in  $P_3$  ( $S_2$  è un nonterminale inutile - si veda Definizione 8.12), dunque l'intera produzione ( $S_2 \rightarrow aB_2$ ) può essere rimossa da  $P_3$  senza alterare il linguaggio generato da  $G_3$ .

Quindi  $P_3$  diventa:

$$P_3 = \{S_3 \rightarrow \lambda \mid aB_2, \quad B_2 \rightarrow b \mid bS_3\}.$$

# Esercizio 7.1

La grammatica  $G$  tale che:

$$L(G) = S(b^*) \cup S((ab)^*) = S(b^* + (ab)^*)$$

è dunque:

$$G = (X, V, S, P)$$

ove:

$$X = X_1 \cup X_3 = \{a, b\}$$

$$V = V_1 \cup V_3 = V_1 \cup V_2 \cup \{S_3\} \cup \{S\} = \{S, S_1, B_2, S_3\}$$

$$\begin{aligned} P &= \{S \rightarrow w \mid S_1 \rightarrow w \in P\} \cup \{S \rightarrow w \mid S_3 \rightarrow w \in P_3\} \cup P_1 \cup P_3 = \\ &= \{S \rightarrow bS_1 \mid \lambda\} \cup \{S \rightarrow aB_2 \mid \lambda\} \cup \{S_1 \rightarrow bS_1 \mid \lambda\} \cup \\ &\quad \cup \{S_3 \rightarrow \lambda \mid aB_2, B_2 \rightarrow b \mid bS_3\} = \\ &= \{S \rightarrow bS_1 \mid aB_2 \mid \lambda, S_1 \rightarrow bS_1 \mid \lambda, S_3 \rightarrow aB_2 \mid \lambda, B_2 \rightarrow bS_3 \mid b\} \end{aligned}$$

## Esercizio 7.1 – Grammatica finale

$= \{S \rightarrow bS_1 \mid aB_2 \mid \lambda, S_1 \rightarrow bS_1 \mid \lambda, S_3 \rightarrow aB_2 \mid \lambda, B_2 \rightarrow bS_3 \mid b\}$

# Esercizio 7.2

$$\mathcal{L}_{REG} \subset \mathcal{L}_3$$

## Esercizio 7.2

Data la seguente grammatica lineare destra:

$$G = (X, V, S, P)$$

ove:

$$X = \{a, b, c\} \quad V = \{S, A, B\}$$
$$P = \left\{ S \xrightarrow{(1)} bA \mid aS \mid b, \quad A \xrightarrow{(2)} aB \mid cS \mid a, \quad B \xrightarrow{(3)} bA \mid cB \mid c \right\}$$

Determinare un'espressione regolare che denota il linguaggio  $L(G)$ .

# Esercizio 7.2

## Esercizio 7.2

Data la seguente grammatica lineare destra:

$$G = (X, V, S, P)$$

ove:

$$\begin{aligned} X &= \{a, b, c\} & V &= \{S, A, B\} \\ P &= \left\{ S \xrightarrow{(1)} bA \mid aS \mid b, \quad A \xrightarrow{(2)} aB \mid cS \mid a, \quad B \xrightarrow{(3)} bA \mid cB \mid c \right\} \end{aligned}$$

Determinare un'espressione regolare che denota il linguaggio  $L(G)$ .

Senza rischio di confusione, denotiamo con  $A$ ,  $B$  ed  $S$  gli insiemi delle stringhe derivabili in  $G$  dai nonterminali  $A$ ,  $B$  ed  $S$ , rispettivamente.

Dalla produzione (2) risulta:

$$A = \{a\} \cdot B \cup \{c\} \cdot S \cup \{a\}$$

che, per brevità, scriviamo nella forma:

$$A = aB \cup cS \cup a \tag{2'}$$

## Esercizio 7.2

Sostituendo in (1) la (2'), si ha:

$$S = b \cdot (aB \cup cS \cup a) \cup aS \cup b$$

che, per la proprietà distributiva della concatenazione rispetto all'unione, è uguale a:

$$S = baB \cup bcS \cup ba \cup aS \cup b$$

## Esercizio 7.2

Sostituendo in (1) la (2'), si ha:

$$S = b \cdot (aB \cup cS \cup a) \cup aS \cup b$$

che, per la proprietà distributiva della concatenazione rispetto all'unione, è uguale a:

$$S = baB \cup bcS \cup ba \cup aS \cup b$$

Applicando la proprietà commutativa per l'unione, la proprietà associativa per la concatenazione e la distributività della concatenazione rispetto all'unione si ha:

$$S = baB \cup (bc \cup a)S \cup (ba \cup b) \quad (1')$$

Sostituendo in (3) la (2'), si ottiene:

$$B = b \cdot (aB \cup cS \cup a) \cup cB \cup c = (ba \cup c)B \cup (bcS \cup ba \cup c) \quad (3')$$

## Esercizio 7.2

Osserviamo la (1') e la (3').

Se denotiamo con  $A$ ,  $B$  ed  $S$  tre espressioni regolari (oltre che gli insiemi di stringhe derivabili dai nonterminali  $A$ ,  $B$  ed  $S$ ), si ha che l'identità tra espressioni regolari corrispondente alla (1') è:

$$S = baB + (bc + a)S + (ba + b) \quad (1'')$$

mentre l'identità tra espressioni regolari corrispondente alla (3') è:

$$B = (ba + c)B + (bcS + ba + c) \quad (3'')$$

## Esercizio 7.2

Osserviamo la (1') e la (3').

Se denotiamo con  $A$ ,  $B$  ed  $S$  tre espressioni regolari (oltre che gli insiemi di stringhe derivabili dai nonterminali  $A$ ,  $B$  ed  $S$ ), si ha che l'identità tra espressioni regolari corrispondente alla (1') è:

$$S = baB + (bc + a)S + (ba + b) \quad (1'')$$

mentre l'identità tra espressioni regolari corrispondente alla (3') è:

$$B = (ba + c)B + (bcS + ba + c) \quad (3'')$$

In entrambi i casi ci troviamo di fronte ad una identità del tipo:

$$R_1 = R_2 \cdot R_1 + R_3 \quad \text{con} \quad R_2 \neq \lambda .$$

Per la proprietà 20) sulle espressioni regolari, si ha:

$$R_1 = {R_2}^* \cdot R_3 .$$

## Esercizio 7.2

Dunque la (3'') diventa:

$$B = (ba + c)^* \cdot (bcS + ba + c) \quad (3''')$$

Sostituendo la (3'') nella (1''), si ottiene:

$$S = \left[ ba \cdot \underline{(ba + c)^*} \cdot (bcS + ba + c) \right] + \underline{(bc + a)S} + \underline{(ba + b)} =$$

## Esercizio 7.2

Dunque la (3'') diventa:

$$B = (ba + c)^* \cdot (bcS + ba + c) \quad (3''')$$

Sostituendo la (3''') nella (1''), si ottiene:

$$\begin{aligned} S &= \left[ \underline{ba} \cdot (ba + c)^* \cdot (bcS + ba + c) \right] + \underline{(bc + a)}S + \underline{(ba + b)} = \\ &= \underline{ba} \cdot (ba + c)^* bcS + \underline{(a + bc)}S + \underline{ba} \cdot (ba + c)^* \cdot (ba + c) + \underline{ba + b} = \\ &= \left( ba \cdot (ba + c)^* bc + a + bc \right) S + ba \cdot (ba + c)^* \cdot (ba + c) + ba + b = \end{aligned}$$

## Esercizio 7.2

Dunque la (3'') diventa:

$$B = (ba + c)^* \cdot (bcS + ba + c) \quad (3''')$$

Sostituendo la (3''') nella (1''), si ottiene:

$$\begin{aligned} S &= \left[ \underline{ba} \cdot (ba + c)^* \cdot (bcS + ba + c) \right] + \underline{(bc + a)}S + \underline{(ba + b)} = \\ &= \underline{ba} \cdot (ba + c)^* bcS + \underline{(a + bc)}S + \underline{ba} \cdot (ba + c)^* \cdot (ba + c) + \underline{ba + b} = \\ &= \left( ba \cdot (ba + c)^* bc + a + bc \right) S + ba \cdot (ba + c)^* \cdot (ba + c) + ba + b = \end{aligned}$$

per la proprietà 20) sulle espressioni regolari

$$= \left( ba \cdot (ba + c)^* bc + a + bc \right)^* \cdot \left( ba \cdot (ba + c)^* (ba + c) + ba + b \right).$$

# Esercizio 7.3

## Esercizio 7.3

Sia  $L$  il linguaggio denotato dalla seguente espressione regolare:

$$(aa + aaa)^*$$

- 1) Trovare un automa a stati finiti che riconosce  $L$ .
- 2) Trasformare l'automa non deterministico trovato al punto 1) in un automa deterministico equivalente.

# Esercizio 7.3

## Esercizio 7.3

Sia  $L$  il linguaggio denotato dalla seguente espressione regolare:

$$(aa + aaa)^*$$

- 1) Trovare un automa a stati finiti che riconosce  $L$ .
- 2) Trasformare l'automa non deterministico trovato al punto 1) in un automa deterministico equivalente.

### **Schema di risoluzione**

- 1) Trovare la grammatica che generi  $(aa)$  e  $(aaa)$
- 2) Trovare la grammatica UNIONE
- 3) Trovare la grammatica ITERAZIONE (utilizzando gli schemi)
- 4) Costruire l'automa a partire da grammatica iterazione di tipo 3
- 5) Trasformare in automa deterministico

# Esercizio 7.3

## Esercizio 7.3

Sia  $L$  il linguaggio denotato dalla seguente espressione regolare:

$$(aa + aaa)^*$$

- 1) Trovare un automa a stati finiti che riconosce  $L$ .
  - 2) Trasformare l'automa non deterministico trovato al punto 1) in un automa deterministico equivalente.
- 
- 1) Determiniamo innanzitutto due grammatiche lineari destre  $G_1$  e  $G_2$  tali che:

$L(G_1) = S(aa) = \{aa\}$	$L(G_2) = S(aaa) = \{aaa\}$
$G_1 = (X, V_1, S_1, P_1)$	$G_2 = (X, V_2, S_2, P_2)$
$X = \{a\}$	
$V_1 = \{S_1, A\}$	$V_2 = \{S_2, B, C\}$
$P_1 = \{S_1 \rightarrow aA, \ A \rightarrow a\}$	$P_2 = \{S_2 \rightarrow aB, \ B \rightarrow aC, \ C \rightarrow a\}$

## Esercizio 7.3

Determiniamo poi la grammatica lineare destra  $G_3$  tale che:

$$L(G_3) = L(G_1) \cup L(G_2) = S(aa) \cup S(aaa) = S(aa + aaa)$$

$$G_3 = (X, V_3, S_3, P_3)$$

dove:

$$V_3 = V_1 \cup V_2 \cup \{S_3\} = \{S_1, S_2, S_3, A, B, C\}.$$

Le produzioni di  $G_3$  si ottengono come segue:

$$P_3 = \{S_3 \rightarrow w \mid S_1 \rightarrow w \in P_1\} \cup \{S_3 \rightarrow w \mid S_2 \rightarrow w \in P_2\} \cup P_1 \cup P_2.$$

Dunque si ha:

$$P_3 = \{S_3 \rightarrow aA \mid aB, \quad S_1 \rightarrow aA, \quad A \rightarrow a, \quad S_2 \rightarrow aB, \quad B \rightarrow aC, \quad C \rightarrow a\}$$

## Esercizio 7.3

da cui possiamo eliminare le produzioni che presentano  $S_1$  o  $S_2$  nella parte sinistra ( $S_1$  ed  $S_2$  sono nonterminali inutili perché non sono presenti nella parte destra di nessuna produzione e né  $S_1$  né  $S_2$  sono il simbolo distintivo di  $G_3$ ). Per cui  $P_3$  diventa:

$$P_3 = \{S_3 \rightarrow aA \mid aB, \ A \rightarrow a, \ B \rightarrow aC, \ C \rightarrow a\}$$

## Esercizio 7.3

da cui possiamo eliminare le produzioni che presentano  $S_1$  o  $S_2$  nella parte sinistra ( $S_1$  ed  $S_2$  sono nonterminali inutili perché non sono presenti nella parte destra di nessuna produzione e né  $S_1$  né  $S_2$  sono il simbolo distintivo di  $G_3$ ). Per cui  $P_3$  diventa:

$$P_3 = \{S_3 \rightarrow aA \mid aB, \ A \rightarrow a, \ B \rightarrow aC, \ C \rightarrow a\}$$

Determiniamo ora la grammatica lineare destra  $G$  tale che:

$$\begin{aligned} L(G) &= (L(G_3))^* = S((aa + aaa)^*) \\ G &= (X, V, S, P) \end{aligned}$$

dove:

$$V = V_3 \cup \{S\} = \{S, S_3, A, B, C\}$$

Le produzioni di  $G$  si ottengono come segue (si veda Tavola 1):

$$\begin{aligned} P &= \{S \rightarrow \lambda\} \cup (P_3 - \{S_3 \rightarrow \lambda\}) \cup \{S \rightarrow w \mid S_3 \rightarrow w \in P_3\} \cup \\ &\quad \cup \{A \rightarrow bS \mid A \rightarrow b \in P_3\} \cup \\ &\quad \cup \{A \rightarrow bS \mid A \rightarrow bB \in P_3, \ b \neq \lambda, \ B \rightarrow \lambda \in P_1\}. \end{aligned}$$

## Esercizio 7.3

$$P_3 = \{S_3 \rightarrow aA \mid aB, \ A \rightarrow a, \ B \rightarrow aC, \ C \rightarrow a\}$$

Le produzioni di  $G$  si ottengono come segue (si veda Tavola 1):

$$\begin{aligned} P = & \{S \rightarrow \lambda\} \cup (P_3 - \{S_3 \rightarrow \lambda\}) \cup \{S \rightarrow w \mid S_3 \rightarrow w \in P_3\} \cup \\ & \cup \{A \rightarrow bS \mid A \rightarrow b \in P_3\} \cup \\ & \cup \{A \rightarrow bS \mid A \rightarrow bB \in P_3, \ b \neq \lambda, \ B \rightarrow \lambda \in P_1\}. \end{aligned}$$

## Esercizio 7.3

$$P_3 = \{S_3 \rightarrow aA \mid aB, A \rightarrow a, B \rightarrow aC, C \rightarrow a\}$$

Le produzioni di  $G$  si ottengono come segue (si veda Tavola 1):

$$\begin{aligned} P = & \{S \rightarrow \lambda\} \cup (P_3 - \{S_3 \rightarrow \lambda\}) \cup \{S \rightarrow w \mid S_3 \rightarrow w \in P_3\} \cup \\ & \cup \{A \rightarrow bS \mid A \rightarrow b \in P_3\} \cup \\ & \cup \{A \rightarrow bS \mid A \rightarrow bB \in P_3, b \neq \lambda, B \rightarrow \lambda \in P_1\}. \end{aligned}$$

Dunque si ha:

$$\begin{aligned} P = & \{S \rightarrow \lambda\} \cup \{S_3 \rightarrow aA \mid aB, A \rightarrow a, B \rightarrow aC, C \rightarrow a\} \cup \\ & \cup \{S \rightarrow aA \mid aB\} \cup \{A \rightarrow aS, C \rightarrow aS\} \end{aligned}$$

da cui possiamo eliminare le produzioni che presentano  $S_3$  nella parte sinistra, ottenendo:

$$P = \{S \rightarrow aA \mid aB \mid \lambda, A \rightarrow aS \mid a, B \rightarrow aC, C \rightarrow aS \mid a\}.$$

## Esercizio 7.3

$$P = \{S \rightarrow aA \mid aB \mid \lambda, \ A \rightarrow aS \mid a, \ B \rightarrow aC, \ C \rightarrow aS \mid a\}.$$

L'automa non deterministico che riconosce  $L(G)$  si costruisce come segue:

$$M = (Q, \delta, q_0, F) \quad \text{con alfabeto di ingresso } X$$

ove, per il relativo algoritmo di trasformazione (Algoritmo 7.1), si ha:

- 1)  $Q = V \cup \{q\} = \{S, A, B, C, q\};$
- 2)  $q_0 = S;$
- 3)  $F = \{q, S\};$

## Esercizio 7.3

$$P = \{S \rightarrow aA \mid aB \mid \lambda, \ A \rightarrow aS \mid a, \ B \rightarrow aC, \ C \rightarrow aS \mid a\}.$$

4)  $\delta$  è definita dalla seguente tavola di transizione:

$\delta$	$S$	$A$	$B$	$C$	$q$
$a$					

Quindi il grafo degli stati è

## Esercizio 7.3

$$P = \{S \rightarrow aA \mid aB \mid \lambda, \ A \rightarrow aS \mid a, \ B \rightarrow aC, \ C \rightarrow aS \mid a\}.$$

4)  $\delta$  è definita dalla seguente tavola di transizione:

$\delta$	$S$	$A$	$B$	$C$	$q$
$a$	$\{A, B\}$	$\{S, q\}$	$\{C\}$	$\{S, q\}$	-

Quindi il grafo degli stati è

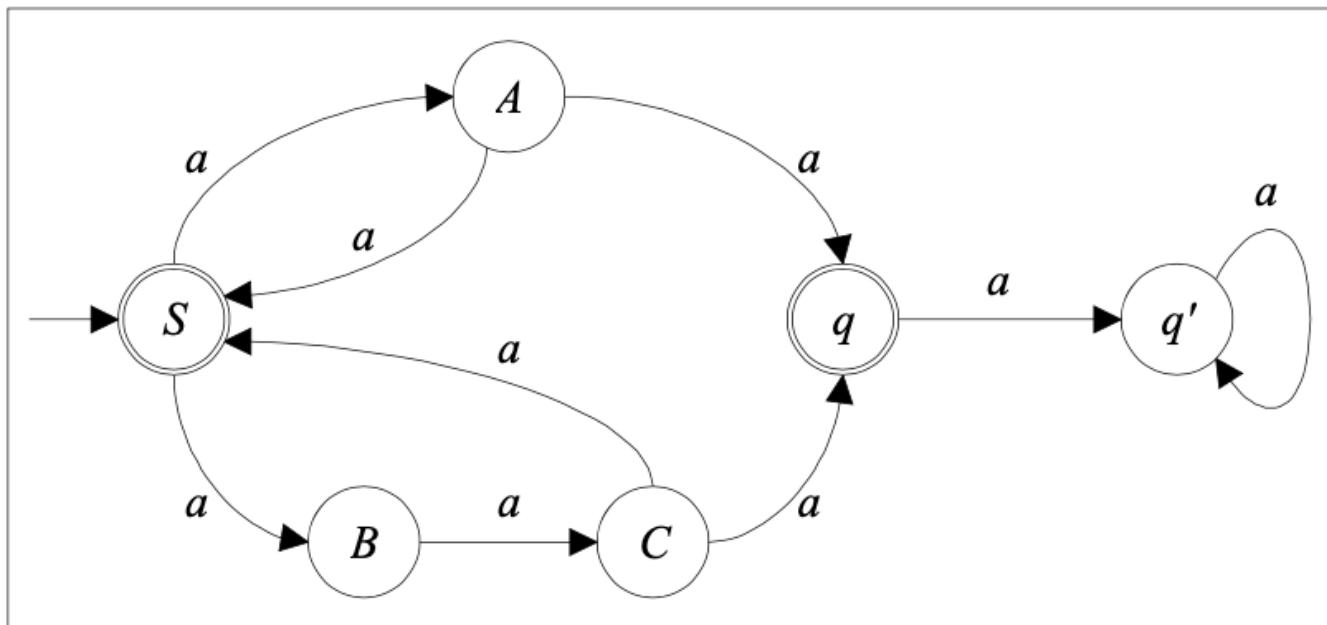
## Esercizio 7.3

$$P = \{S \rightarrow aA \mid aB \mid \lambda, A \rightarrow aS \mid a, B \rightarrow aC, C \rightarrow aS \mid a\}.$$

4)  $\delta$  è definita dalla seguente tavola di transizione:

$\delta$	$S$	$A$	$B$	$C$	$q$
$a$	$\{A, B\}$	$\{S, q\}$	$\{C\}$	$\{S, q\}$	-

Quindi il grafo degli stati è



## Esercizio 7.3

2) L'automa accettore deterministico  $M'$  equivalente ad  $M$  (ossia tale che  $T(M') = T(M)$ ) si costruisce, secondo l'Algoritmo 6.1, come segue:

$$M' = (Q', \delta', q'_0, F')$$

ove:

- a)  $Q' = 2^Q$ ;
- b)  $q'_0 = \{q_0\}$ ;
- c)  $F' = \{p \subseteq Q \mid p \cap F \neq \emptyset\}$ ;
- d)  $\delta' : Q' \times X \rightarrow Q'$   $\exists'$

$\forall q' = \{q_1, q_2, \dots, q_i\} \in Q', \forall x \in X :$

$$\delta'(q', x) = \delta'(\{q_1, q_2, \dots, q_i\}, x) =$$

$$= \delta(q_1, x) \cup \delta(q_2, x) \cup \dots \cup \delta(q_i, x) = \bigcup_{j=1}^i \delta(q_j, x)$$

## Esercizio 7.3

Dunque si ha:

$$M' = (Q', \delta', q'_0, F')$$

con:

- a)  $Q' = 2^Q = 2^{\{S, A, B, C, q\}}$ ,  $|Q'| = 32$ ;
- b)  $q'_0 = \{S\}$ ;
- c)  $F' = \{\{q\}, \{S\}, \{q, A\}, \{q, B\}, \{q, C\}, \dots, \{S, A\}, \{S, B\}, \{S, C\}, \dots, \{q, S\}\}$ .

## Esercizio 7.3

Dunque si ha:

$$M' = (Q', \delta', q'_0, F')$$

con:

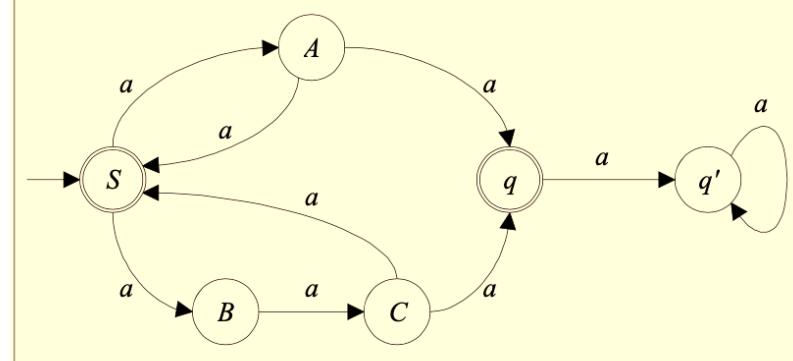
- a)  $Q' = 2^Q = 2^{\{S, A, B, C, q\}}, |Q'| = 32;$
- b)  $q'_0 = \{S\};$
- c)  $F' = \{\{q\}, \{S\}, \{q, A\}, \{q, B\}, \{q, C\}, \dots, \{S, A\}, \{S, B\}, \{S, C\}, \dots, \{q, S\}\}.$

Nella pratica, non è necessario considerare tutti gli stati di  $Q'$ , in quanto molti sono irraggiungibili. Per questo motivo, iniziamo a definire  $\delta'$  dal nuovo stato iniziale e proseguiamo definendo  $\delta'$  per un nuovo stato di  $Q'$  non appena esso viene generato.

# Esercizio 7.3

Per cui si ha:

$$\delta'(\{S\}, a) = \delta(S, a) = \{A, B\}$$

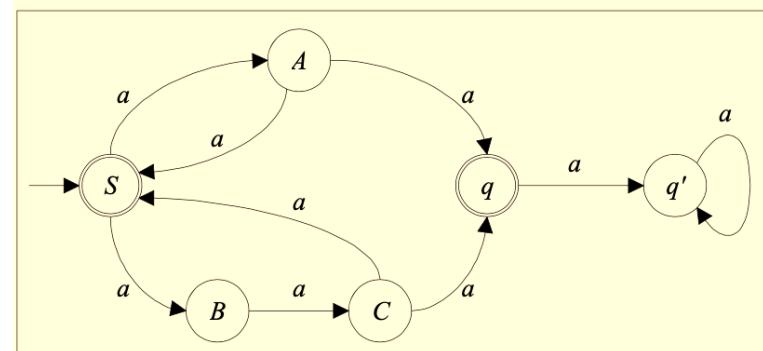


## Esercizio 7.3

Per cui si ha:

$$\delta'(\{S\}, a) = \delta(S, a) = \{A, B\}$$

$$\delta'(\{A, B\}, a) = \delta(A, a) \cup \delta(B, a) = \{S, q\} \cup \{C\} = \{S, C, q\}$$



# Esercizio 7.3

Per cui si ha:

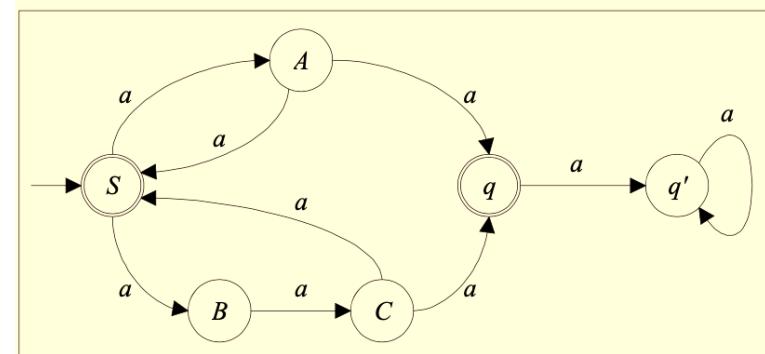
$$\delta'(\{S\}, a) = \delta(S, a) = \{A, B\}$$

$$\delta'(\{A, B\}, a) = \delta(A, a) \cup \delta(B, a) = \{S, q\} \cup \{C\} = \{S, C, q\}$$

$$\begin{aligned}\delta'(\{S, C, q\}, a) &= \delta(S, a) \cup \delta(C, a) \cup \delta(q, a) = \\ &= \{A, B\} \cup \{S, q\} \cup \emptyset = \{S, A, B, q\}\end{aligned}$$

$$\begin{aligned}\delta'(\{S, A, B, q\}, a) &= \delta(S, a) \cup \delta(A, a) \cup \delta(B, a) \cup \delta(q, a) = \\ &= \{A, B\} \cup \{S, q\} \cup \{C\} \cup \emptyset = \\ &= \{S, A, B, C, q\}\end{aligned}$$

$$\delta'(\{S, A, B, C, q\}, a) = \delta(\{S, A, B, q\}, a) \cup \delta(C, a) = \{S, A, B, C, q\}$$



# Esercizio 7.3

Per cui si ha:

$$\delta'(\{S\}, a) = \delta(S, a) = \{A, B\}$$

$$\delta'(\{A, B\}, a) = \delta(A, a) \cup \delta(B, a) = \{S, q\} \cup \{C\} = \{S, C, q\}$$

$$\delta'(\{S, C, q\}, a) = \delta(S, a) \cup \delta(C, a) \cup \delta(q, a) =$$

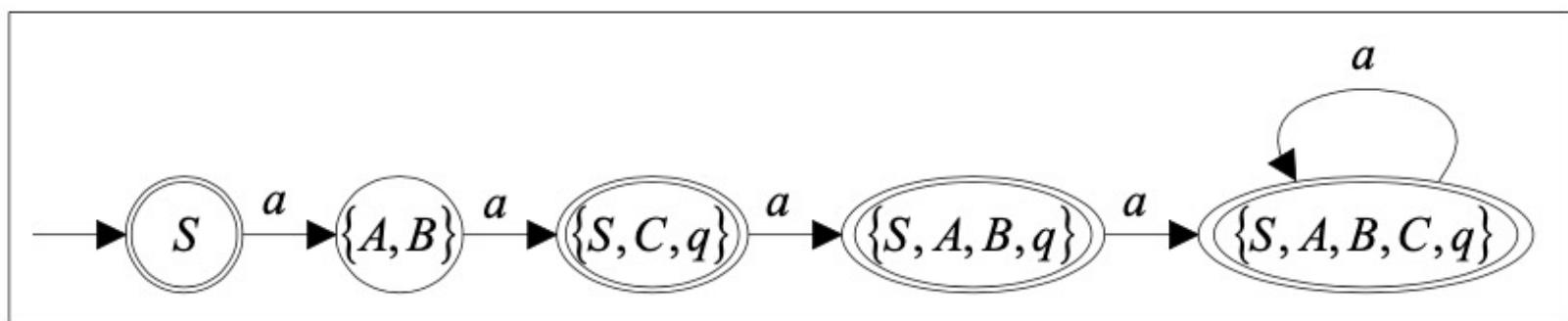
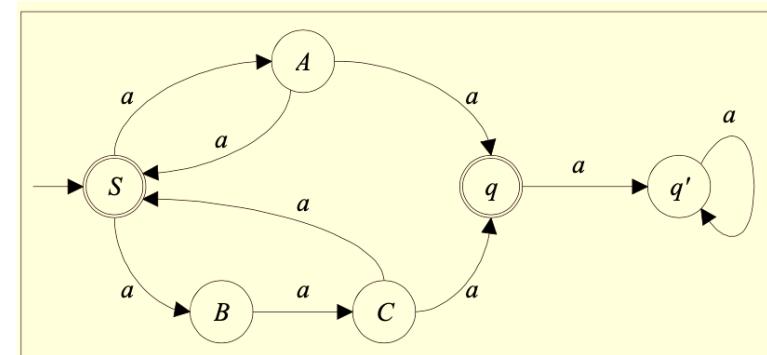
$$= \{A, B\} \cup \{S, q\} \cup \emptyset = \{S, A, B, q\}$$

$$\delta'(\{S, A, B, q\}, a) = \delta(S, a) \cup \delta(A, a) \cup \delta(B, a) \cup \delta(q, a) =$$

$$= \{A, B\} \cup \{S, q\} \cup \{C\} \cup \emptyset =$$

$$= \{S, A, B, C, q\}$$

$$\delta'(\{S, A, B, C, q\}, a) = \delta(\{S, A, B, q\}, a) \cup \delta(C, a) = \{S, A, B, C, q\}$$



## Esercizio 7.3 – Risoluzione Alternativa

Un modo alternativo è il seguente.

$$\left( (aa + aaa)^* \right)$$

Osserviamo che:

$$S\left( (aa + aaa)^* \right) = \{ \lambda, aa, aaa, aaaa, aaaaa, \dots \} = \cdot$$

## Esercizio 7.3 – Risoluzione Alternativa

Un modo alternativo è il seguente.

$$\left( (aa + aaa)^* \right)$$

Osserviamo che:

$$S\left( (aa + aaa)^* \right) = \{ \lambda, aa, aaa, aaaa, aaaaa, \dots \} = \{ a \}^* - \{ a \}$$

## Esercizio 7.3 – Risoluzione Alternativa

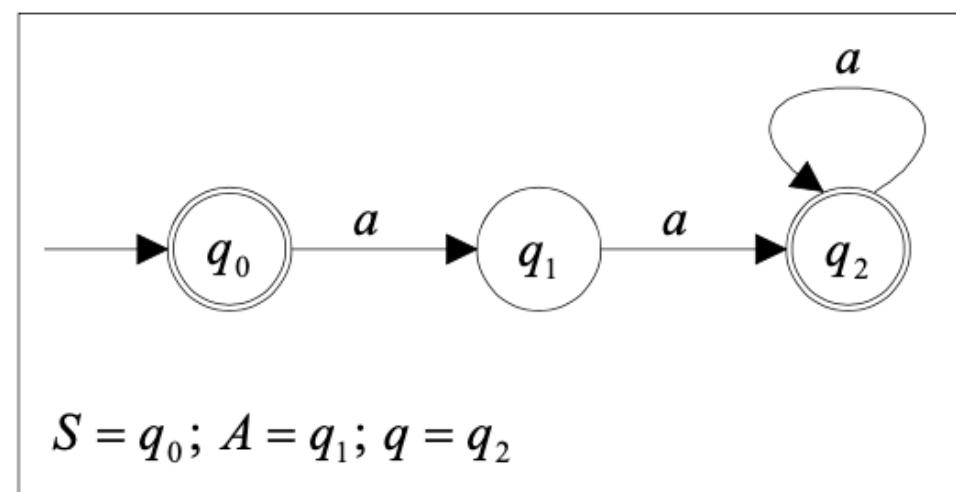
Un modo alternativo è il seguente.

$$((aa + aaa)^*)$$

Osserviamo che:

$$S((aa + aaa)^*) = \{\lambda, aa, aaa, aaaa, aaaaa, \dots\} = \{a\}^* - \{a\}$$

L'automa (minimo e deterministico) che riconosce  $\{a\}^* - \{a\}$  è



# Esercizio 7.4

## Esercizio 7.4

Con riferimento all'Esercizio 6.1, determinare una grammatica lineare destra che genera il linguaggio:

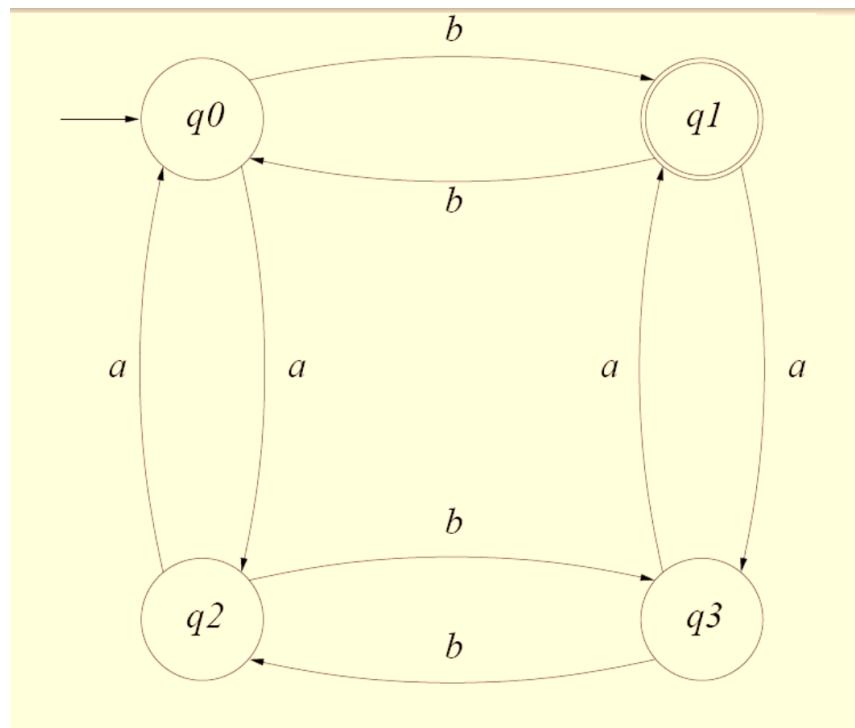
$$L = \left\{ w \mid w \in \{a,b\}^*, \text{ } w \text{ ha un numero pari di } a \text{ ed un numero dispari di } b \right\}$$

# Esercizio 7.4

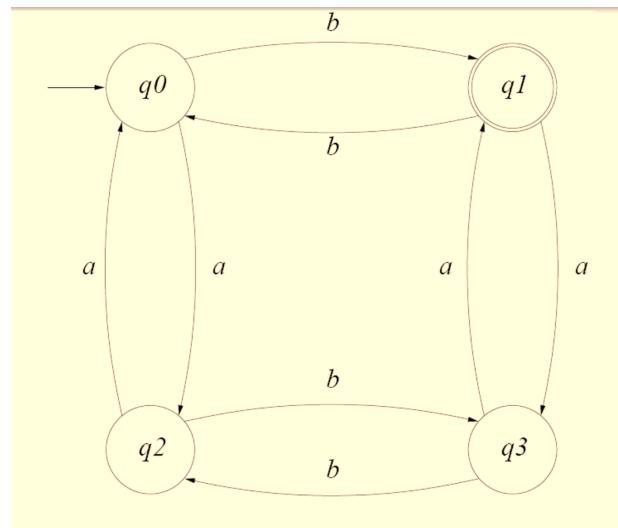
## Esercizio 7.4

Con riferimento all'Esercizio 6.1, determinare una grammatica lineare destra che genera il linguaggio:

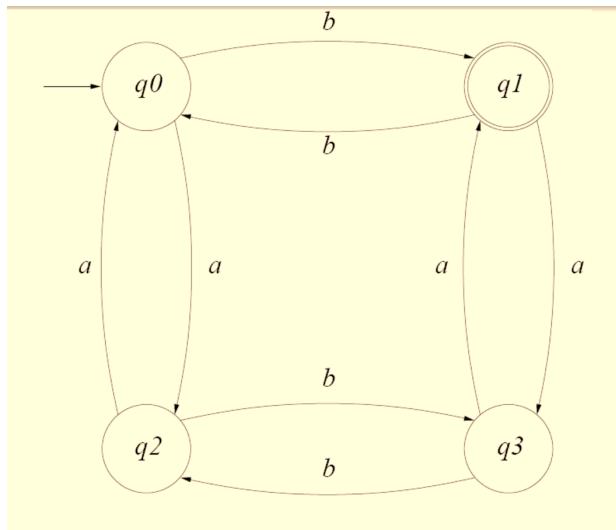
$$L = \left\{ w \mid w \in \{a, b\}^*, \text{ } w \text{ ha un numero pari di } a \text{ ed un numero dispari di } b \right\}$$



## Esercizio 7.4



## Esercizio 7.4



Una grammatica lineare destra che genera il linguaggio  $L$  può essere costruita a partire dall'automa accettore a stati finiti determinato nell'Esercizio 6.1 utilizzando l'Algoritmo 7.2, come segue:

$$G = (X, V, S, P)$$

dove:

- 1)  $X = \{a, b\}$ ;
- 2)  $V = Q = \{q_0, q_1, q_2, q_3\}$ ;
- 3)  $S = q_0$ ;
- 4)  $P = \{q_0 \rightarrow aq_2 \mid bq_1 \mid b, q_1 \rightarrow aq_3 \mid bq_0, q_2 \rightarrow aq_0 \mid bq_3, q_3 \rightarrow aq_1 \mid bq_2 \mid a\}$ .

# Esercizio 7.5

## Esercizio 7.5

Sia  $L$  il linguaggio denotato dalla seguente espressione regolare:

$$ab(bb)^*c$$

- 1) Trovare un automa a stati finiti che riconosce  $L$ .
- 2) Trasformare l'automa non deterministico al punto 1) in un automa deterministico equivalente.

### **Schema di Risoluzione:**

- 1) Trovare il linguaggio denotato dall'espressione regolare
- 2) Costruire le grammatiche per i singoli linguaggi
- 3) Utilizzare le proprietà di chiusura per combinare grammatiche semplici e ottenere grammatiche più complesse (iterazione, concatenazione) dello stesso tipo
- 4) Applicare il teorema di Kleene per costruire un automa che riconosca il linguaggio di tipo 3
- 5) Trasformare l'automa da non deterministico in deterministico

# Esercizio 7.5

## Esercizio 7.5

Sia  $L$  il linguaggio denotato dalla seguente espressione regolare:

$$ab(bb)^*c$$

- 1) Trovare un automa a stati finiti che riconosce  $L$ .
- 2) Trasformare l'automa non deterministico al punto 1) in un automa deterministico equivalente.
- 1) Determiniamo una grammatica lineare destra che genera  $L$ .

Poiché:

$$\begin{aligned} S(ab(bb)^*c) &= S(a) \cdot S(b) \cdot S((bb)^*) \cdot S(c) = \\ &= \{a\} \cdot \{b\} \cdot \{bb\}^* \cdot \{c\} = \\ &= \{ab\} \cdot \{bb\}^* \cdot \{c\} \end{aligned}$$

possiamo determinare una grammatica che genera  $L$  sfruttando le proprietà di chiusura dei linguaggi di tipo ‘3’.

# Esercizio 7.5

Una grammatica che genera  $\{ab\}$  è:

$$G_1 = (X, V_1, S_1, P_1)$$

dove:

- $X = \{a, b\}$  ;
- $V_1 = \{S_1, A\}$  ;
- $P_1 = \{S_1 \rightarrow aA, A \rightarrow b\}$  .

# Esercizio 7.5

Una grammatica che genera  $\{ab\}$  è:

$$G_1 = (X, V_1, S_1, P_1)$$

dove:

- $X = \{a, b\}$ ;
- $V_1 = \{S_1, A\}$ ;
- $P_1 = \{S_1 \rightarrow aA, A \rightarrow b\}$ .

Una grammatica  $G_3$  che genera  $\{bb\}^*$  si ottiene per iterazione dalla grammatica  $G_2$  che genera  $\{bb\}$ :

$$G_2 = (X, V_2, S_2, P_2)$$

dove:

- $X = \{b\}$ ;
- $V_2 = \{S_2, B\}$ ;
- $P_2 = \{S_2 \rightarrow bB, B \rightarrow b\}$ .

## Esercizio 7.5

$$G_3 = (X, V_3, S_3, P_3)$$

dove:

- $X = \{b\}$ ;
- $V_3 = V_2 \cup \{S_3\} = \{S_2, B, S_3\}$ ;
- $P_3 = \{S_3 \rightarrow bB \mid \lambda, B \rightarrow bS_3 \mid b, \cancel{S_2 \rightarrow bB}\}^1$ .

Una grammatica che genera  $\{c\}$  è:

$$G_4 = (X, V_4, S_4, P_4)$$

dove:

- $X = \{c\}$ ;
- $V_4 = S_4$ ;
- $P_4 = \{S_4 \rightarrow c\}$ .

## Esercizio 7.5

Una grammatica che genera  $\{ab\} \cdot \{bb\}^*$  è:

$$G_5 = (X, V_5, S_5, P_5)$$

dove:

- $X = \{a, b\}$ ;
- $V_5 = V_1 \cup V_3 = \{S_1, A, S_3, B\}$ ;
- $S_5 = S_1$ ;
- $P_5 =$

## Esercizio 7.5

Una grammatica che genera  $\{ab\} \cdot \{bb\}^*$  è:

$$G_5 = (X, V_5, S_5, P_5)$$

dove:

- $X = \{a, b\}$ ;
- $V_5 = V_1 \cup V_3 = \{S_1, A, S_3, B\}$ ;
- $S_5 = S_1$ ;
- $P_5 = \{S_1 \rightarrow aA, A \rightarrow bS_3, S_3 \rightarrow bB \mid \lambda, B \rightarrow bS_3 \mid b\}$ .

## Esercizio 7.5

Infine, una grammatica che genera il linguaggio  $L = \{ab\} \cdot \{bb\}^* \cdot \{c\}$  è:

$$G = (X, V, S, P)$$

dove:

- $X = \{a, b, c\}$ ;
- $V = V_4 \cup V_5 = \{S_4, S_1, A, S_3, B\}$ ;
- $S = S_1$ ;
- $P = \{S_1 \rightarrow aA, B \rightarrow bS_3 \mid bS_4, A \rightarrow bS_3 \mid bS_4, S_3 \rightarrow bB, S_4 \rightarrow c\}$ .

## Esercizio 7.5

L'automa che riconosce  $L(G)$  si costruisce come segue, in base all'Algoritmo 7.1:

$M = (Q, \delta, q_0, F)$  con alfabeto di ingresso  $X$

- $Q = V \cup \{q\}$ ,  $q \notin V$ ;
- $q_0 = S_1$ ;
- $F = \{q\}$ ;

e il diagramma di transizione è dato

$$P = \{S_1 \rightarrow aA, B \rightarrow bS_3 \mid bS_4, A \rightarrow bS_3 \mid bS_4, S_3 \rightarrow bB, S_4 \rightarrow c\}.$$

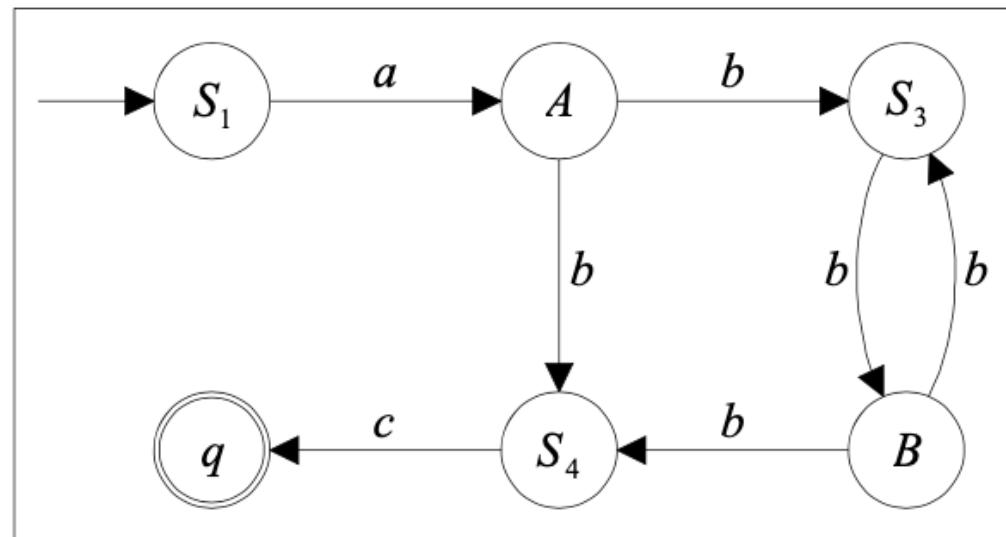
## Esercizio 7.5

L'automa che riconosce  $L(G)$  si costruisce come segue, in base all'Algoritmo 7.1:

$$M = (Q, \delta, q_0, F) \text{ con alfabeto di ingresso } X$$

- $Q = V \cup \{q\}$ ,  $q \notin V$ ;
- $q_0 = S_1$ ;
- $F = \{q\}$ ;

e il diagramma di transizione è dato



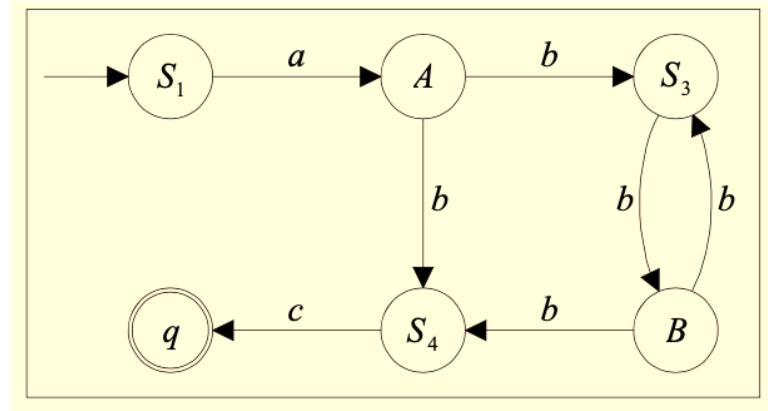
# Esercizio 7.5

2) L'automa deterministico  $M'$  equivalente ad  $M$  si costruisce come segue, in base all'Algoritmo 6.1:

$$M' = (Q', \delta', q'_0, F')$$

dove:

- $Q' = 2^Q$ ;
- $q'_0 = \{S_1\}$ ;
- $F' = \{p \subseteq Q \mid p \cap F \neq \emptyset\}$  con  $|F'| = 32$ ;



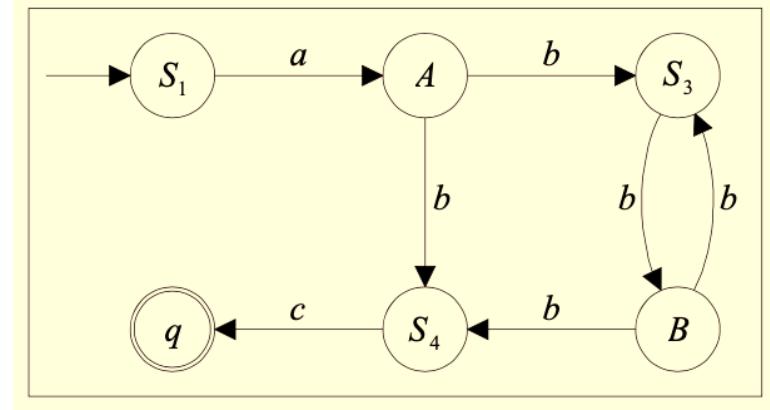
# Esercizio 7.5

2) L'automa deterministico  $M'$  equivalente ad  $M$  si costruisce come segue, in base all'Algoritmo 6.1:

$$M' = (Q', \delta', q'_0, F')$$

dove:

- $Q' = 2^Q$ ;
- $q'_0 = \{S_1\}$ ;
- $F' = \{p \subseteq Q \mid p \cap F \neq \emptyset\}$  con  $|F'| = 32$ ;



- ◆  $\delta'(\{S_1\}, a) = \delta(S_1, a) = \{A\}$ ;
- ◆  $\delta'(\{S_1\}, b) = \delta(S_1, b) = \text{non è definita}$ ;
- ◆  $\delta'(\{S_1\}, c) = \delta(S_1, c) = \text{non è definita}$ ;
- ◆  $\delta'(\{A\}, a) = \delta(A, a) = \text{non è definita}$ ;
- ◆  $\delta'(\{A\}, b) = \delta(A, b) = \{S_3, S_4\}$ ;
- ◆  $\delta'(\{A\}, c) = \delta(A, c) = \text{non è definita}$ ;

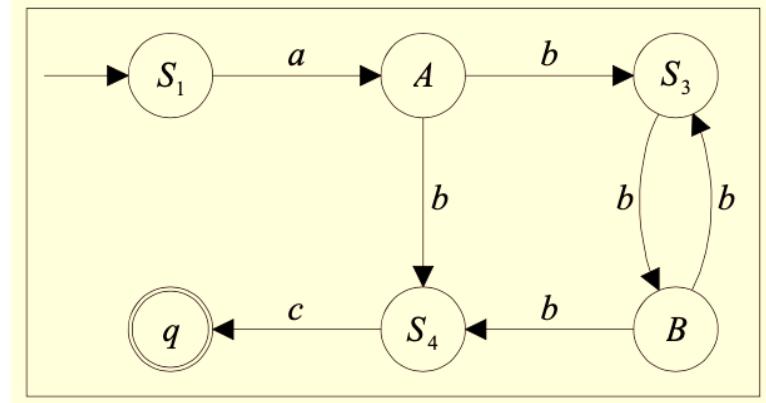
# Esercizio 7.5

2) L'automa deterministico  $M'$  equivalente ad  $M$  si costruisce come segue, in base all'Algoritmo 6.1:

$$M' = (Q', \delta', q'_0, F')$$

dove:

- $Q' = 2^Q$ ;
- $q'_0 = \{S_1\}$ ;
- $F' = \{p \subseteq Q \mid p \cap F \neq \emptyset\}$  con  $|F'| = 32$ ;



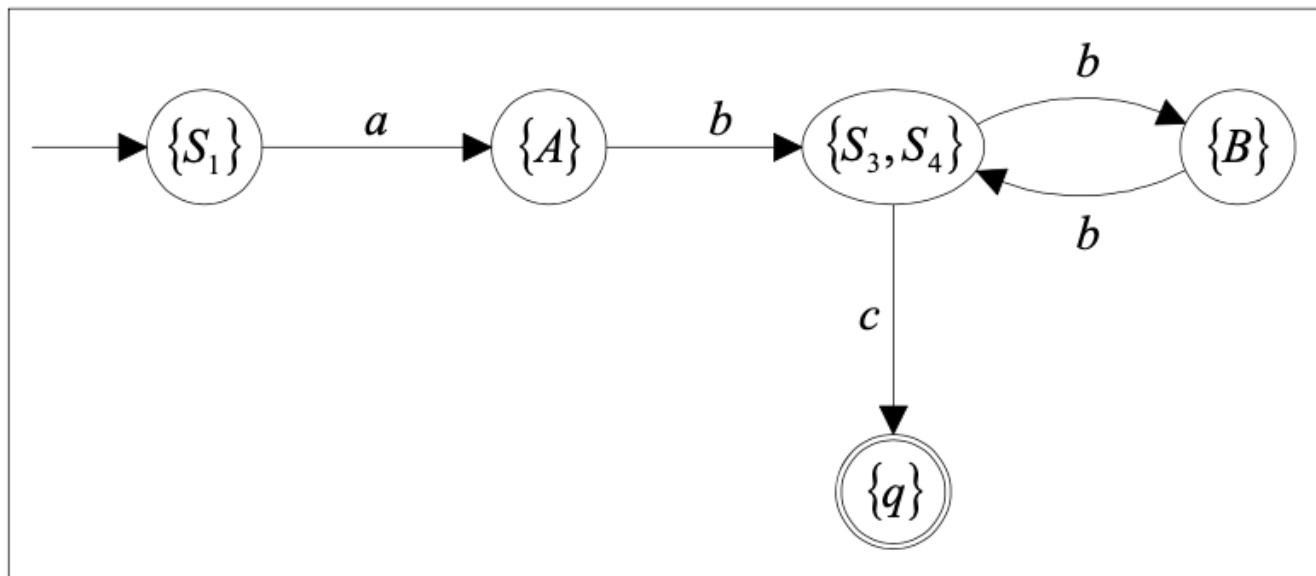
- ◆  $\delta'(\{S_1\}, a) = \delta(S_1, a) = \{A\}$ ;
- ◆  $\delta'(\{S_1\}, b) = \delta(S_1, b) = \text{non è definita}$ ;
- ◆  $\delta'(\{S_1\}, c) = \delta(S_1, c) = \text{non è definita}$ ;
- ◆  $\delta'(\{A\}, a) = \delta(A, a) = \text{non è definita}$ ;
- ◆  $\delta'(\{A\}, b) = \delta(A, b) = \{S_3, S_4\}$ ;
- ◆  $\delta'(\{A\}, c) = \delta(A, c) = \text{non è definita}$ ;
- ◆  $\delta'(\{S_3, S_4\}, a) = \delta(S_3, a) \cup \delta(S_4, a) = \text{non è definita}$ ;
- ◆  $\delta'(\{S_3, S_4\}, b) = \delta(S_3, b) \cup \delta(S_4, b) = \{B\}$ ;
- ◆  $\delta'(\{S_3, S_4\}, c) = \delta(S_3, c) \cup \delta(S_4, c) = \{q\}$ ;
- ◆  $\delta'(\{B\}, a) = \delta(B, a) = \text{non è definita}$ ;
- ◆  $\delta'(\{B\}, b) = \delta(B, b) = \{S_3, S_4\}$ ;
- ◆  $\delta'(\{B\}, c) = \delta(B, c) = \text{non è definita}$ ;

# Esercizio 7.5

- ◆  $\delta'(\{S_1\}, a) = \delta(S_1, a) = \{A\}$ ;
- ◆  $\delta'(\{S_1\}, b) = \delta(S_1, b) =$  non è definita;
- ◆  $\delta'(\{S_1\}, c) = \delta(S_1, c) =$  non è definita;
- ◆  $\delta'(\{A\}, a) = \delta(A, a) =$  non è definita;
- ◆  $\delta'(\{A\}, b) = \delta(A, b) = \{S_3, S_4\}$ ;
- ◆  $\delta'(\{A\}, c) = \delta(A, c) =$  non è definita;
- ◆  $\delta'(\{S_3, S_4\}, a) = \delta(S_3, a) \cup \delta(S_4, a) =$  non è definita;
- ◆  $\delta'(\{S_3, S_4\}, b) = \delta(S_3, b) \cup \delta(S_4, b) = \{B\}$ ;
- ◆  $\delta'(\{S_3, S_4\}, c) = \delta(S_3, c) \cup \delta(S_4, c) = \{q\}$ ;
- ◆  $\delta'(\{B\}, a) = \delta(B, a) =$  non è definita;
- ◆  $\delta'(\{B\}, b) = \delta(B, b) = \{S_3, S_4\}$ ;
- ◆  $\delta'(\{B\}, c) = \delta(B, c) =$  non è definita;

# Esercizio 7.5

- ◆  $\delta'(\{S_1\}, a) = \delta(S_1, a) = \{A\}$ ;
- ◆  $\delta'(\{S_1\}, b) = \delta(S_1, b) = \text{non è definita}$ ;
- ◆  $\delta'(\{S_1\}, c) = \delta(S_1, c) = \text{non è definita}$ ;
- ◆  $\delta'(\{A\}, a) = \delta(A, a) = \text{non è definita}$ ;
- ◆  $\delta'(\{A\}, b) = \delta(A, b) = \{S_3, S_4\}$ ;
- ◆  $\delta'(\{A\}, c) = \delta(A, c) = \text{non è definita}$ ;
- ◆  $\delta'(\{S_3, S_4\}, a) = \delta(S_3, a) \cup \delta(S_4, a) = \text{non è definita}$ ;
- ◆  $\delta'(\{S_3, S_4\}, b) = \delta(S_3, b) \cup \delta(S_4, b) = \{B\}$ ;
- ◆  $\delta'(\{S_3, S_4\}, c) = \delta(S_3, c) \cup \delta(S_4, c) = \{q\}$ ;
- ◆  $\delta'(\{B\}, a) = \delta(B, a) = \text{non è definita}$ ;
- ◆  $\delta'(\{B\}, b) = \delta(B, b) = \{S_3, S_4\}$ ;
- ◆  $\delta'(\{B\}, c) = \delta(B, c) = \text{non è definita}$ ;



# Esercizio 7.6

Si consideri la seguente grammatica lineare destra:

$$G = (X, V, S, P)$$

con

$$\begin{aligned} X &= \{a, b\} & V &= \{S, B\} \\ P &= \{S \rightarrow aB, \quad B \rightarrow aB \mid bS \mid a\} \end{aligned}$$

Determinare un automa deterministico  $M$  tale che:

$$L(G) = T(M)$$

# Esercizio 7.6

Si consideri la seguente grammatica lineare destra:

$$G = (X, V, S, P)$$

con

$$\begin{aligned} X &= \{a, b\} & V &= \{S, B\} \\ P &= \{S \rightarrow aB, \quad B \rightarrow aB \mid bS \mid a\} \end{aligned}$$

Determinare un automa deterministico  $M$  tale che:

$$L(G) = T(M)$$

Facciamo riferimento all'algoritmo per la costruzione dell'automa accettore a stati finiti non deterministico equivalente ad una grammatica lineare destra (Algoritmo 7.1).

$$M = (Q, \delta, q_0, F) \text{ con}$$

- 1)  $X = \{a, b\}$  alfabeto di ingresso;
- 2)  $Q = \{S, B, q\}$  ;
- 3)  $q_0 = S$  ;
- 4)  $F = \{q\}$  ;

$\delta$  è definita come segue:

## Esercizio 7.6

$$P = \{S \rightarrow aB, B \rightarrow aB \mid bS \mid a\}$$

$\delta$  è definita come segue:

5)  $\delta : Q \times X \rightarrow 2^Q$

5.a)  $S \rightarrow aB$  dà origine a :

$B \rightarrow aB$  dà origine a :

$B \rightarrow bS$  dà origine a :

5.b)  $B \rightarrow a$  dà origine a :

# Esercizio 7.6

$$P = \{S \rightarrow aB, B \rightarrow aB \mid bS \mid a\}$$

$\delta$  è definita come segue:

5)  $\delta : Q \times X \rightarrow 2^Q$

5.a)  $S \rightarrow aB$  dà origine a:  $B \in \delta(S, a)$ ;

$B \rightarrow aB$  dà origine a:  $B \in \delta(B, a)$ ;

$B \rightarrow bS$  dà origine a:  $S \in \delta(B, b)$ ;

5.b)  $B \rightarrow a$  dà origine a:  $q \in \delta(B, a)$ .

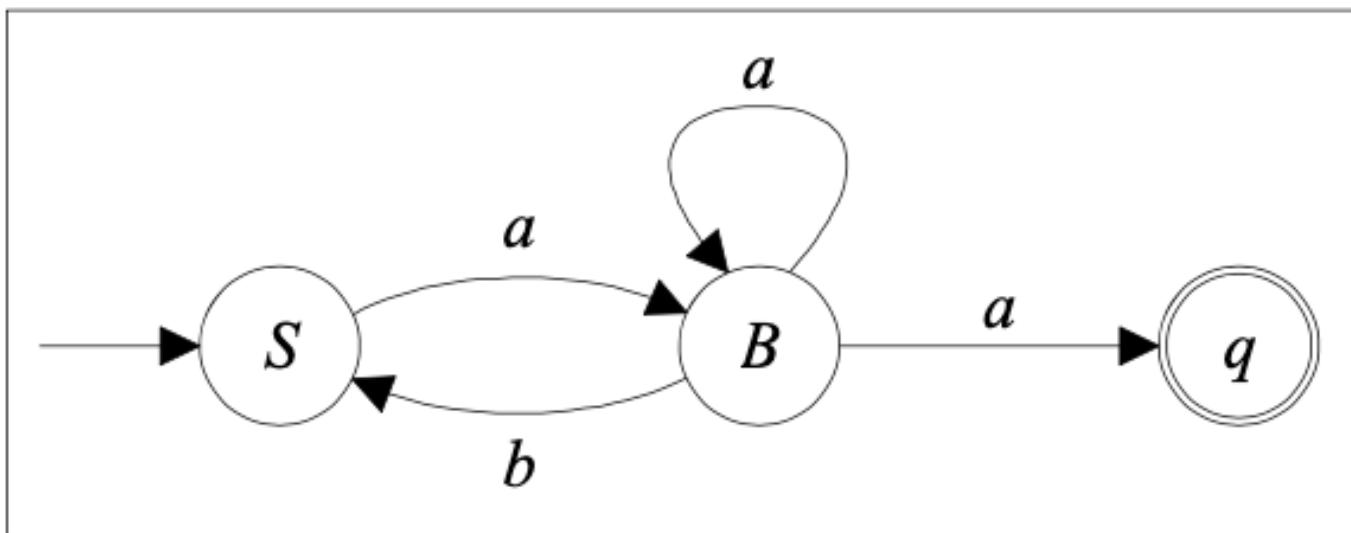
Per cui, la tavola di transizione che riassume la definizione della funzione  $\delta$  è la seguente:

$\delta$	$S$	$B$	$q$
$a$	$\{B\}$	$\{B, q\}$	—
$b$	—	$\{S\}$	—

## Esercizio 7.6

$$P = \{S \rightarrow aB, B \rightarrow aB \mid bS \mid a\}$$

$\delta$	$S$	$B$	$q$
$a$	$\{B\}$	$\{B, q\}$	-
$b$	-	$\{S\}$	-



# Esercizio 7.6

L'automa accettore a stati finiti deterministico  $M'$  equivalente ad  $M$  si costruisce come segue (Algoritmo 6.1):

$$M' = (Q', \delta', q'_0, F') \quad X = \{a, b\} \text{ alfabeto di ingresso}$$

con:

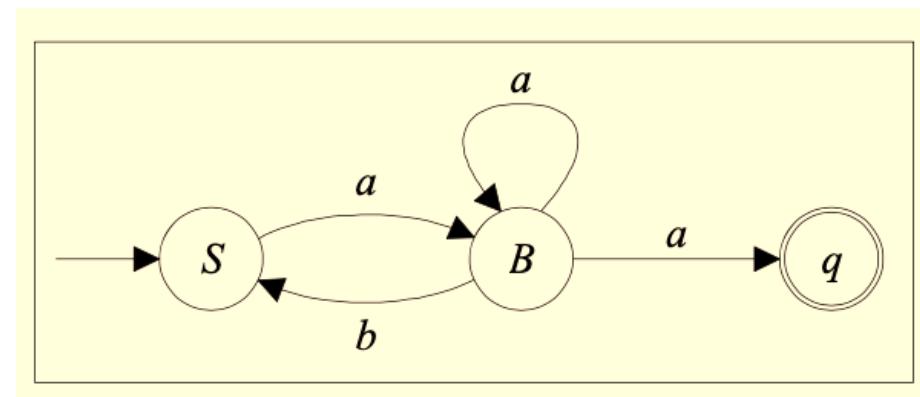
- i)  $Q' = 2^Q = 2^{\{S, B, q\}}$ ;
- ii)  $q'_0 = \{S\}$ ;
- iii)  $F' = \{\{q\}, \{q, S\}, \{q, B\}, \{q, S, B\}\}$ ;

# Esercizio 7.6

iv)  $\delta'$  è definita come segue:

$$\delta': Q' \times X \rightarrow Q'$$

- $\delta'(\{S\}, a) = \delta(S, a) =$
- $\delta'(\{S\}, b) = \delta(S, b) =$
- $\delta'(\{B\}, a) = \delta(B, a) =$
- $\delta'(\{B\}, b) = \delta(B, b) =$
- $\delta'(\{B, q\}, a) = \delta(B, a) \cup \delta(q, a) =$
- $\delta'(\{B, q\}, b) = \delta(B, b) \cup \delta(q, b) =$

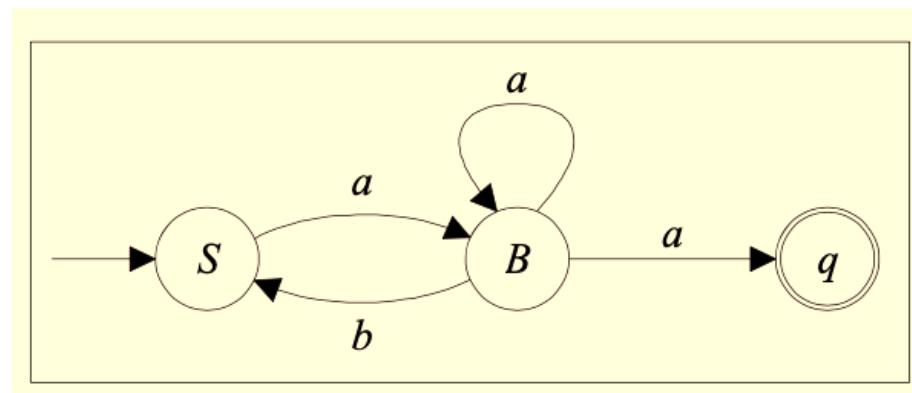


# Esercizio 7.6

iv)  $\delta'$  è definita come segue:

$$\delta': Q' \times X \rightarrow Q'$$

- $\delta'(\{S\}, a) = \delta(S, a) = \{B\}$ ;
- $\delta'(\{S\}, b) = \delta(S, b) = \text{non definita}$ ;
- $\delta'(\{B\}, a) = \delta(B, a) = \{B, q\}$ ;
- $\delta'(\{B\}, b) = \delta(B, b) = \{S\}$ ;
- $\delta'(\{B, q\}, a) = \delta(B, a) \cup \delta(q, a) = \{B, q\}$ ;
- $\delta'(\{B, q\}, b) = \delta(B, b) \cup \delta(q, b) = \{S\}$ .

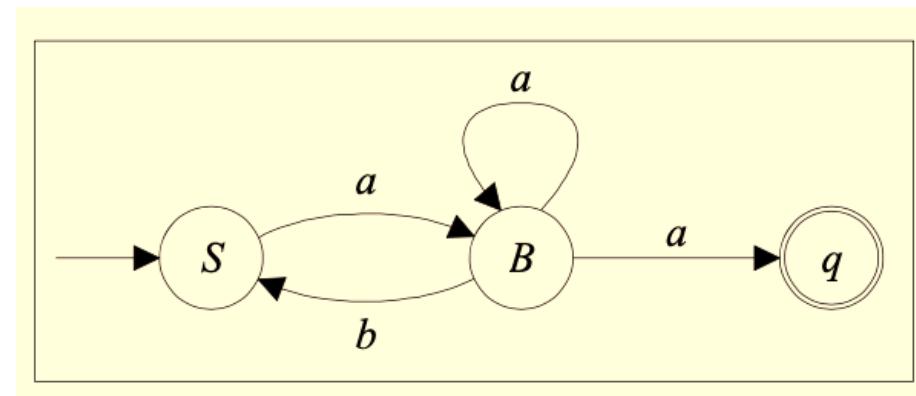


# Esercizio 7.6

iv)  $\delta'$  è definita come segue:

$$\delta': Q' \times X \rightarrow Q'$$

- $\delta'(\{S\}, a) = \delta(S, a) = \{B\}$ ;
- $\delta'(\{S\}, b) = \delta(S, b) = \text{non definita}$ ;
- $\delta'(\{B\}, a) = \delta(B, a) = \{B, q\}$ ;
- $\delta'(\{B\}, b) = \delta(B, b) = \{S\}$ ;
- $\delta'(\{B, q\}, a) = \delta(B, a) \cup \delta(q, a) = \{B, q\}$ ;
- $\delta'(\{B, q\}, b) = \delta(B, b) \cup \delta(q, b) = \{S\}$ .



$\delta'$	$\{S\}$	$\{B\}$	$\{B, q\}$
$a$	$\{B\}$	$\{B, q\}$	$\{B, q\}$
$b$	-	$\{S\}$	$\{S\}$

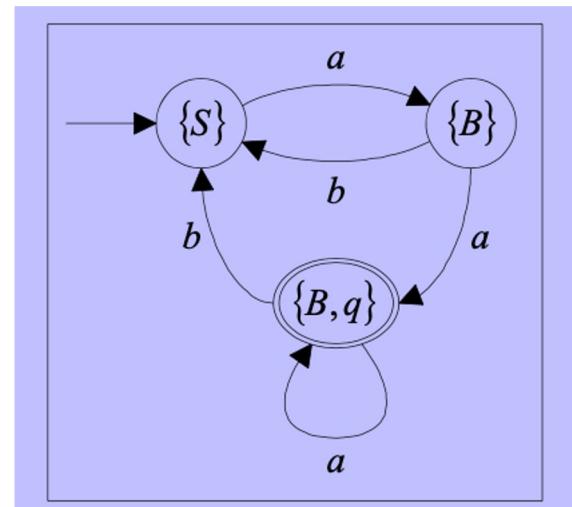
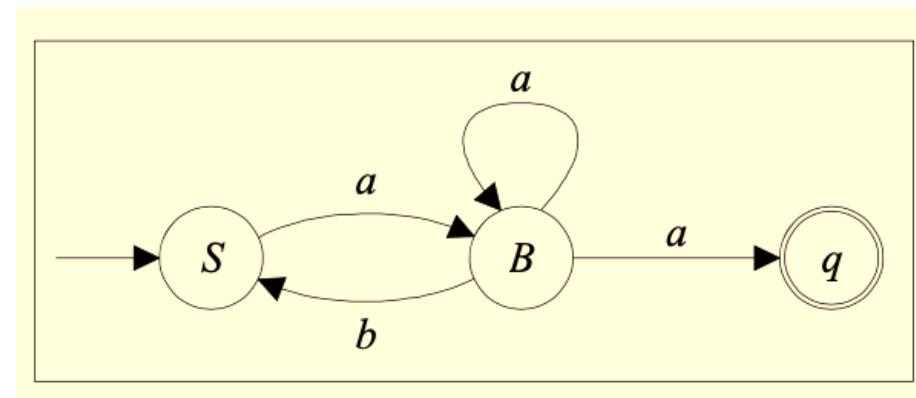
# Esercizio 7.6

iv)  $\delta'$  è definita come segue:

$$\delta': Q' \times X \rightarrow Q'$$

- $\delta'(\{S\}, a) = \delta(S, a) = \{B\}$ ;
- $\delta'(\{S\}, b) = \delta(S, b) = \text{non definita}$ ;
- $\delta'(\{B\}, a) = \delta(B, a) = \{B, q\}$ ;
- $\delta'(\{B\}, b) = \delta(B, b) = \{S\}$ ;
- $\delta'(\{B, q\}, a) = \delta(B, a) \cup \delta(q, a) = \{B, q\}$ ;
- $\delta'(\{B, q\}, b) = \delta(B, b) \cup \delta(q, b) = \{S\}$ .

$\delta'$	$\{S\}$	$\{B\}$	$\{B, q\}$
$a$	$\{B\}$	$\{B, q\}$	$\{B, q\}$
$b$	-	$\{S\}$	$\{S\}$



# Pumping Lemma per i linguaggi regolari

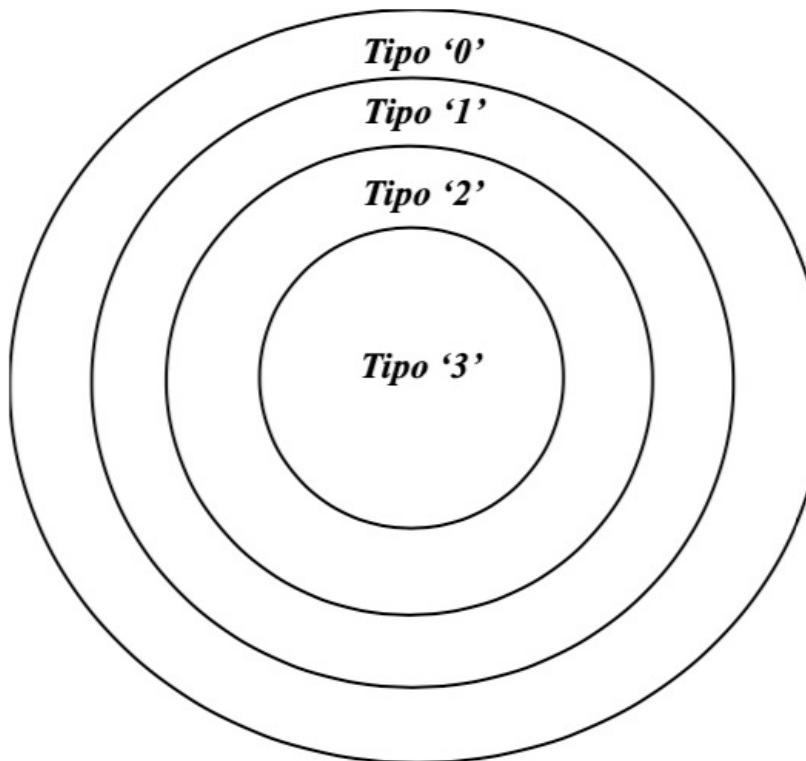
- A cosa serve il pumping lemma per i linguaggi liberi?

# Pumping Lemma per i linguaggi regolari

- A cosa serve il pumping lemma per i linguaggi liberi?
  - A dimostrare (per assurdo) che un linguaggio non è di tipo 2 (dunque è di tipo 1)
- Il pumping lemma per linguaggi regolari svolge lo stesso scopo: **dimostrare che un linguaggio non è di tipo 3**

# Pumping Lemma per i linguaggi regolari

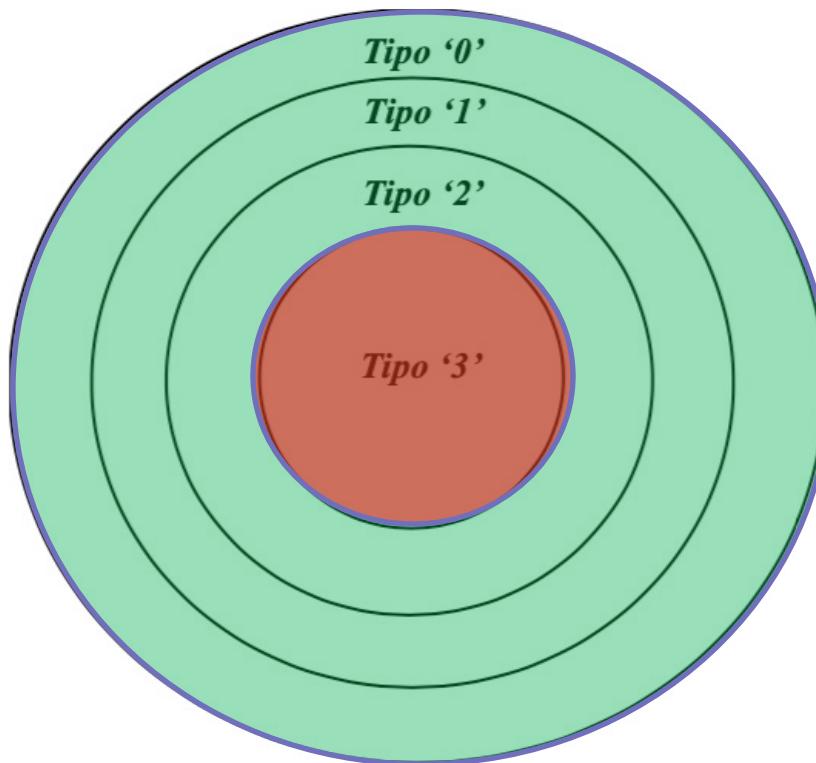
A dimostrare che un **linguaggio non è regolare**



Non è di tipo 3 (ma –ad esempio – di tipo 2)

# Pumping Lemma per i linguaggi regolari

A dimostrare che un **linguaggio non è regolare**



Non è di tipo 3 (ma –ad esempio – di tipo 2)

# Pumping Lemma per i linguaggi regolari

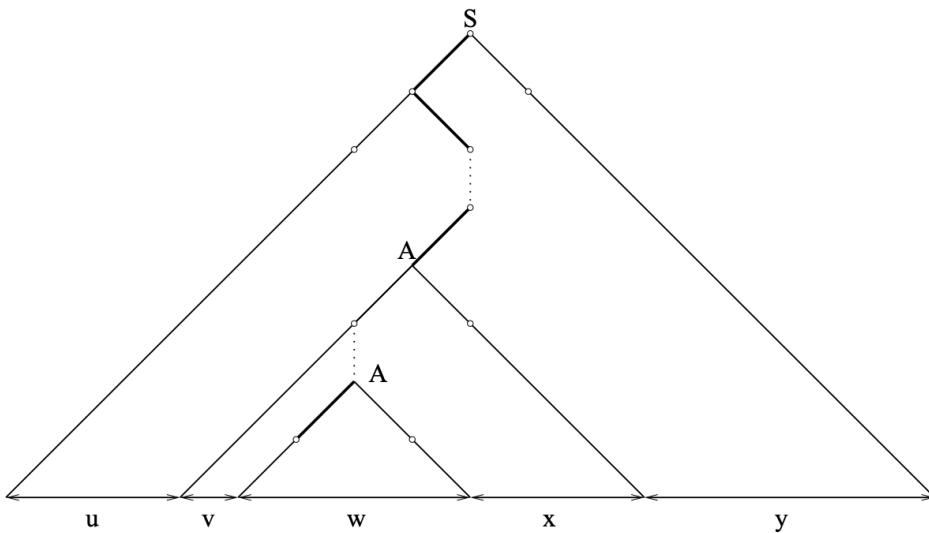
- A cosa serve il pumping lemma per i linguaggi liberi?
  - A dimostrare (per assurdo) che un linguaggio non è di tipo 2 (dunque è di tipo 1)
- **Differenza importante:** aver confutato il PL per linguaggio liberi rende automatico il fatto che un linguaggio sia di tipo 1. Confutare il PL per linguaggi regolari non rende automatico che il linguaggio sia di tipo 2 (potrebbe anche essere di tipo 1!)

# Pumping Lemma per i linguaggi regolari

- Così come il pumping lemma per i linguaggi liberi, anche quello per i linguaggi regolari **evidenzia una caratteristica che tutti i linguaggi regolari devono avere**
- Quale era la caratteristica dei linguaggi liberi?

# Pumping Lemma per i linguaggi regolari

- Così come il pumping lemma per i linguaggi liberi, anche quello per i linguaggi regolari **evidenzia una caratteristica che tutti i linguaggi regolari devono avere**
- Quale era la caratteristica dei linguaggi liberi?



Un sottoinsieme di parole cresce in **maniera costante** (applicando il principio di sostituzione dei sottoalberi)

# Pumping Lemma per i linguaggi regolari

- Così come il pumping lemma per i linguaggi liberi, anche quello per i linguaggi regolari **evidenzia una caratteristica che tutti i linguaggi regolari devono avere**
- Quale era la caratteristica dei linguaggi liberi?

Passiamo a studiare le caratteristiche dei **linguaggi regolari**  
Quale caratteristica hanno le parole dei linguaggi regolari?

# Pumping Lemma per i linguaggi regolari

- Così come il pumping lemma per i linguaggi liberi, anche quello per i linguaggi regolari **evidenzia una caratteristica che tutti i linguaggi regolari devono avere**
- Quale era la caratteristica dei linguaggi liberi?

Passiamo a studiare le caratteristiche dei **linguaggi regolari**  
Quale caratteristica hanno le parole dei linguaggi regolari?

- (1)  $A \rightarrow bC$  con  $A, C \in V$  e  $b \in X$ ;
- (2)  $A \rightarrow b$  con  $A \in V$  e  $b \in X \cup \{\lambda\}$ .

**Cosa notiamo?**

# Pumping Lemma per i linguaggi regolari

- Così come il pumping lemma per i linguaggi liberi, anche quello per i linguaggi regolari **evidenzia una caratteristica che tutti i linguaggi regolari devono avere**
- Quale era la caratteristica dei linguaggi liberi?

Passiamo a studiare le caratteristiche dei **linguaggi regolari**  
Quale caratteristica hanno le parole dei linguaggi regolari?

- (1)  $A \rightarrow bC$  con  $A, C \in V$  e  $b \in X$ ;
- (2)  $A \rightarrow b$  con  $A \in V$  e  $b \in X \cup \{\lambda\}$ .

**Cosa notiamo?**

**Ad ogni passo della derivazione la lunghezza delle parole cresce di uno**

# Pumping Lemma per i linguaggi regolari

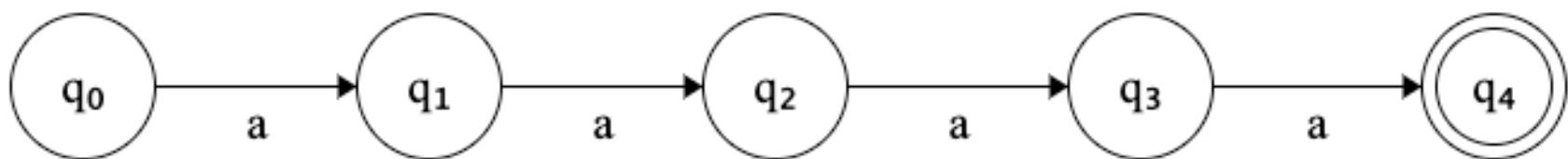
- Ad ogni passo della derivazione, una parola appartenente ad un linguaggio regolare (o, identicamente, generabile a partire da una grammatica regolare/di tipo 3) **cresce di uno**.
- **Domanda:**
  - Quanti stati servono a riconoscere una parola di lunghezza  $N$ ?

# Pumping Lemma per i linguaggi regolari

- Ad ogni passo della derivazione, una parola appartenente ad un linguaggio regolare (o, identicamente, generabile a partire da una grammatica regolare/di tipo 3) **cresce di uno**.
- **Domanda:**

- Quanti stati servono a riconoscere una parola di lunghezza  $N$ ?

$N+1$



# Pumping Lemma per i linguaggi regolari

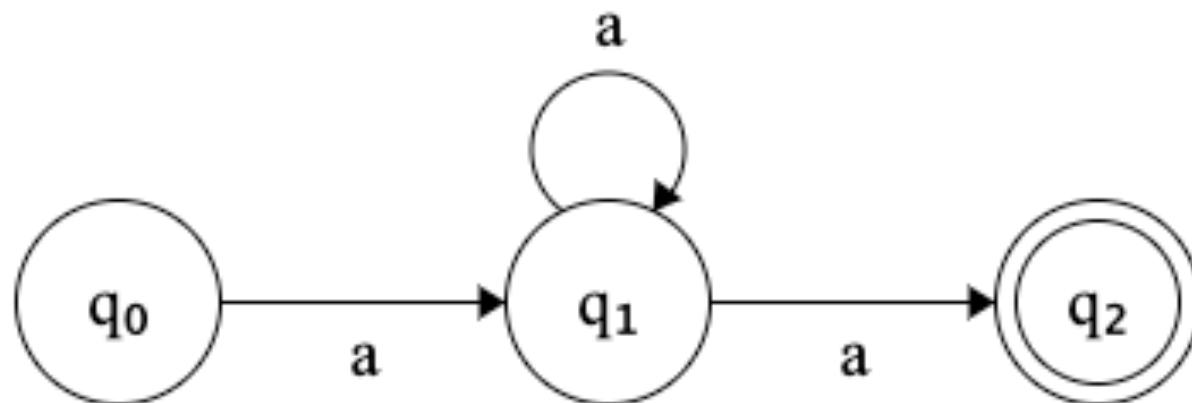
- Ad ogni passo della derivazione, una parola appartenente ad un linguaggio regolare (o, identicamente, generabile a partire da una grammatica regolare/di tipo 3) **cresce di uno**.
- **Domanda (formulazione diverse):**
  - Se un automa ha  $N$  stati, come faccio a riconoscere una parola di lunghezza  $\geq N$ ?

# Pumping Lemma per i linguaggi regolari

- Ad ogni passo della derivazione, una parola appartenente ad un linguaggio regolare (o, identicamente, generabile a partire da una grammatica regolare/di tipo 3) **cresce di uno**.
- **Domanda (formulazione diverse):**
  - Se un automa ha  $N$  stati, come faccio a riconoscere una parola di lunghezza  $\geq N$ ?
  - Assumendo che ogni stato legge 1 carattere, ci dovrà essere almeno **uno stato ripetuto**
    - **Un ciclo!**

# Pumping Lemma per i linguaggi regolari

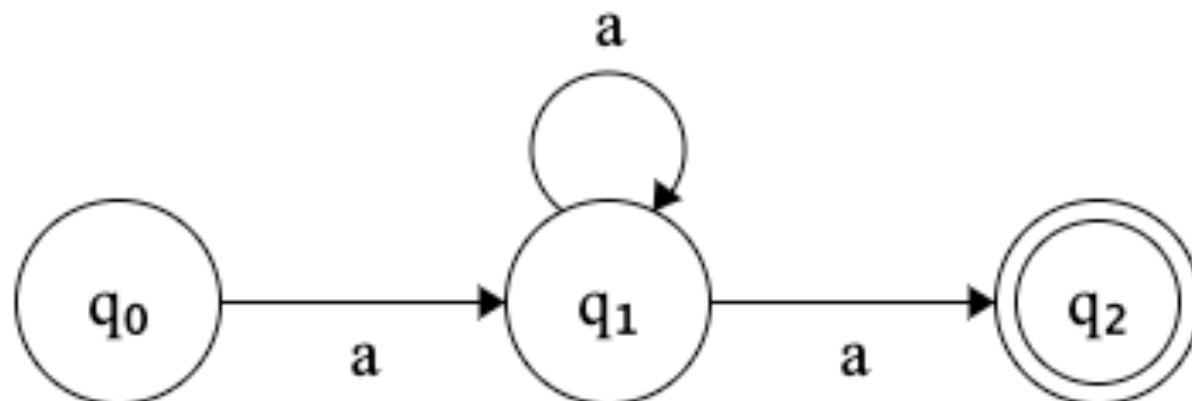
- Se un automa ha  $N$  stati, come faccio a riconoscere parole di lunghezza  $\geq N$ ?
  - Ho bisogno che nell'automa ci sia almeno uno stato ripetuto (cioè un ciclo)
- Esempio



# Pumping Lemma per i linguaggi regolari

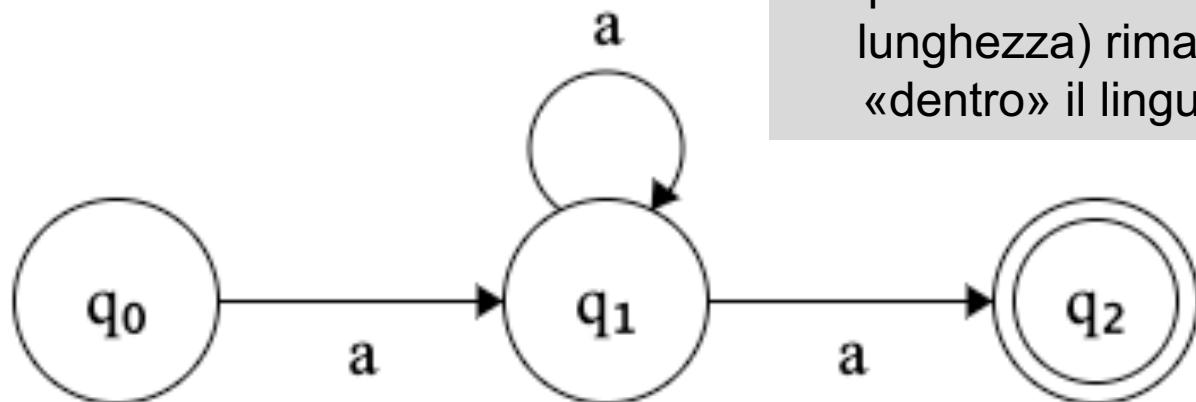
- Se un automa ha  $N$  stati, come faccio a riconoscere parole di lunghezza  $\geq N$ ?
  - Ho bisogno che nell'automa ci sia almeno uno stato ripetuto (cioè un ciclo)
- Esempio

Cosa notiamo?



# Pumping Lemma per i linguaggi regolari

- Se un automa ha  $N$  stati, come faccio a riconoscere parole di lunghezza  $\geq N$ ?
  - Ho bisogno che nell'automa ci sia almeno uno stato ripetuto (cioè un ciclo)
- **Esempio**



Una volta introdotto un ciclo però, tutte le parole che passano da quel ciclo (a prescindere dalla loro lunghezza) rimarranno «dentro» il linguaggio!

# Pumping Lemma per i linguaggi regolari

## ■ Ricapitoliamo

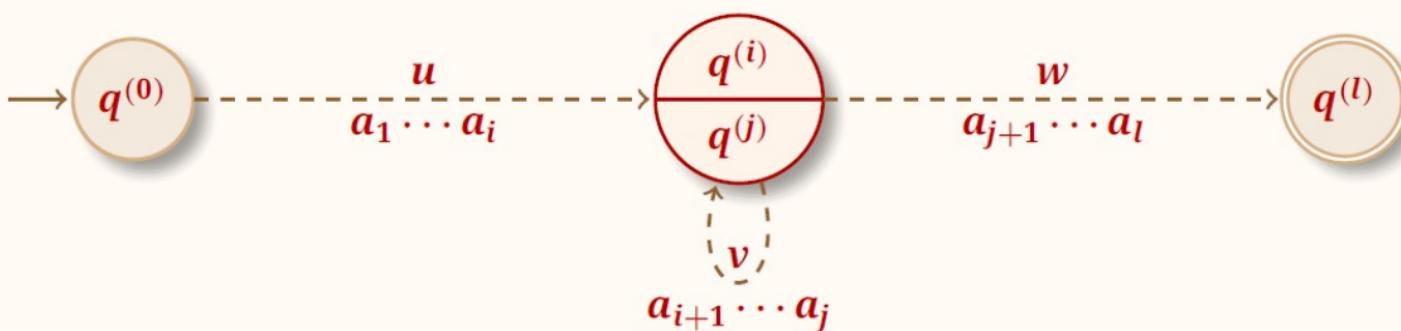
- I linguaggi di tipo tre hanno la caratteristica di far crescere la derivazione di uno ad ogni passaggio
- Gli automi riconoscono un simbolo ad ogni transizione
- Se un automa ha N stati, può riconoscere (senza cicli) al più una parola di lunghezza  $N-1$
- Se una parola ha una lunghezza  $z \geq N$  ( $N =$  numero degli stati) è necessario che uno stato si ripeta
  - Stato ripetuto = introdurre un ciclo
- Una volta introdotto il ciclo, esso può essere percorso infinite volte (è il pompaggio, come nel principio di sostituzione dei sotto-alberi!). **Tutte le parole resteranno dentro il linguaggio.**

# Pumping Lemma per i linguaggi regolari

- Sia  $M = (Q, \delta, q_0, F)$  un automa accettore a stati finiti con  $n$  stati ( $|Q|=n$ ) e sia  $z \in T(M)$ ,  $|z| \geq n$ .  
Allora  $z$  può essere scritta come  $uvw$ , e  $uv^*w \subset T(M)$  (ossia  $\forall i, i \geq 0 : uv^i w \in T(M)$ ).

# Pumping Lemma per i linguaggi regolari

- Sia  $M = (Q, \delta, q_0, F)$  un automa accettore a stati finiti con  $n$  stati ( $|Q|=n$ ) e sia  $z \in T(M)$ ,  $|z| \geq n$ . Allora  $z$  può essere scritta come  $uvw$ , e  $uv^*w \subset T(M)$  (ossia  $\forall i, i \geq 0 : uv^i w \in T(M)$ ).



# Pumping Lemma per i linguaggi regolari

- Sia  $M = (Q, \delta, q_0, F)$  un automa accettore a stati finiti con  $n$  stati ( $|Q|=n$ ) e sia  $z \in T(M)$ ,  $|z| \geq n$ .  
Allora  $z$  può essere scritta come  $uvw$ , e  $uv^*w \subset T(M)$  (ossia  $\forall i, i \geq 0 : uv^i w \in T(M)$ ).
- → Ogni parola appartenente a un linguaggio regolare può essere scritta **come sequenza di tre segmenti (u,v,w)** – Con il segmento «v» che può essere pompato un numero arbitrario di volte, **rimanendo sempre nel linguaggio**.

# Pumping Lemma per i linguaggi regolari

- Una formulazione alternativa è la seguente:

Sia  $L = T(M)$  un linguaggio regolare con

$M = (Q, \delta, q_0, F)$  un automa accettore a stati finiti.

Allora  $\exists n = |Q|$  t.c.  $\forall z \in L, |z| \geq n : z = uvw$  e:

- $|uv| \leq n$
- $v \neq \lambda$
- $uv^i w \in L, \forall i, i \geq 0$

# Pumping Lemma per i linguaggi regolari

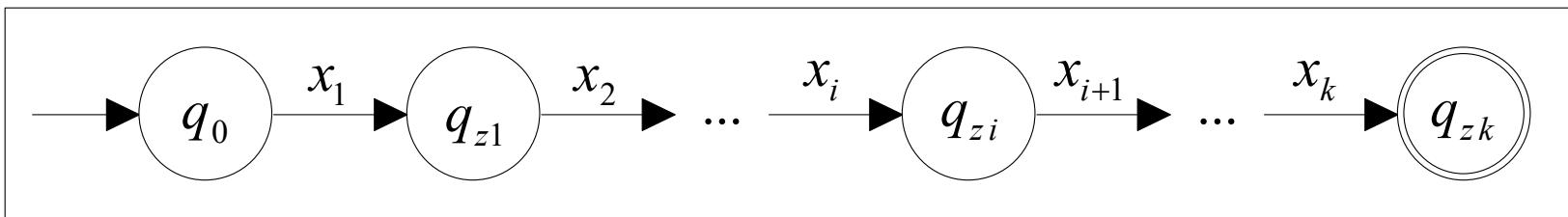
	P.L. per Linguaggi Regolari	P.L. per Linguaggi Liberi
Parole	Crescono di <b>uno</b>	Crescono in maniera <b>costante</b>
Cosa vogliamo «forzare»	I cicli sull'automa	Avere due non-terminali sullo stesso cammino, <b>in modo da poter applicare il principio di sostituzione dei sottoalberi</b>
Struttura delle parole	$Z = uvw$	$Z = uvxwy$

# Pumping Lemma per i linguaggi regolari

## ■ Dimostrazione

Sia  $z = x_1 x_2 \dots x_k$ ,  $z \in T(M)$ .

Possiamo rappresentare il comportamento dell'automa  $M$ , con ingresso  $z$ , come segue:



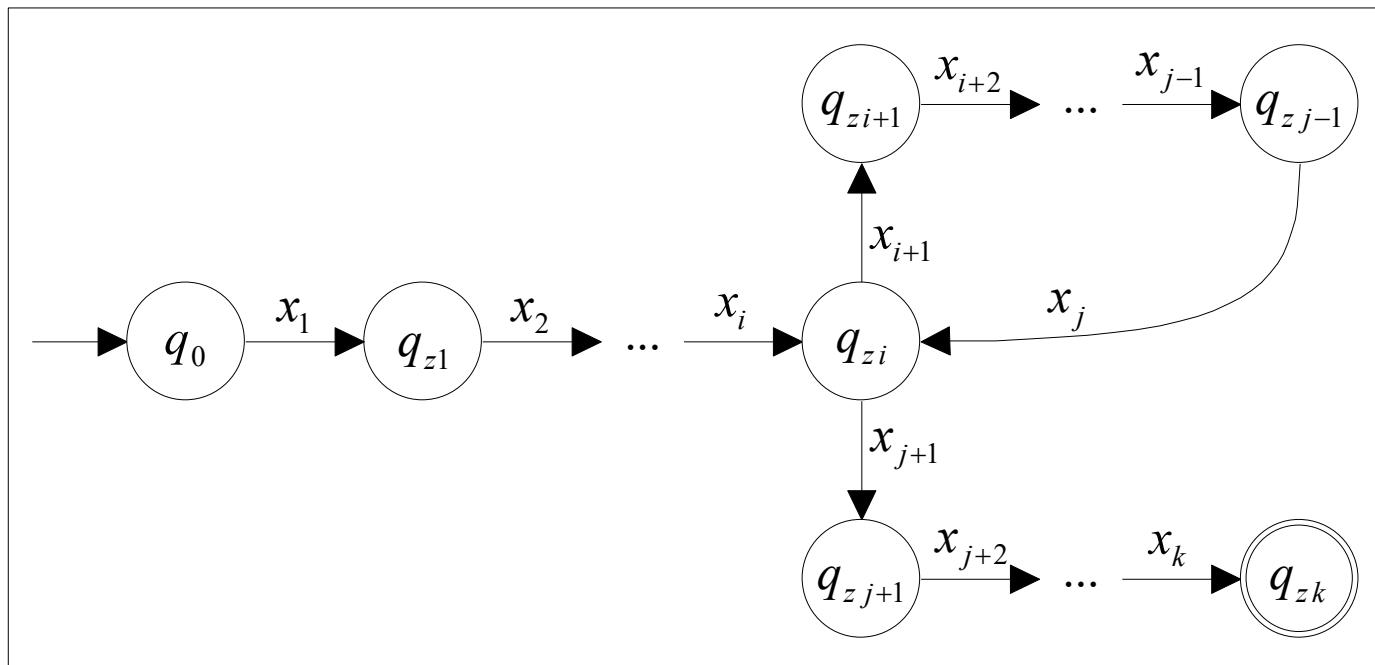
Se si ha:  $|z| \geq n$ , nella figura devono comparire almeno  $n+1$  stati ma, poiché  $M$  ha solo  $n$  stati distinti ( $|Q| = n$ ) almeno uno stato tra  $q_0, q_{z1}, q_{z2}, \dots, q_{zk}$  deve comparire due volte.

Supponiamo che si abbia  $q_{zi} = q_{zj}$ ,  $i < j$ .

# Pumping Lemma per i linguaggi regolari

## ■ Dimostrazione

Si ha dunque la situazione rappresentata in figura:

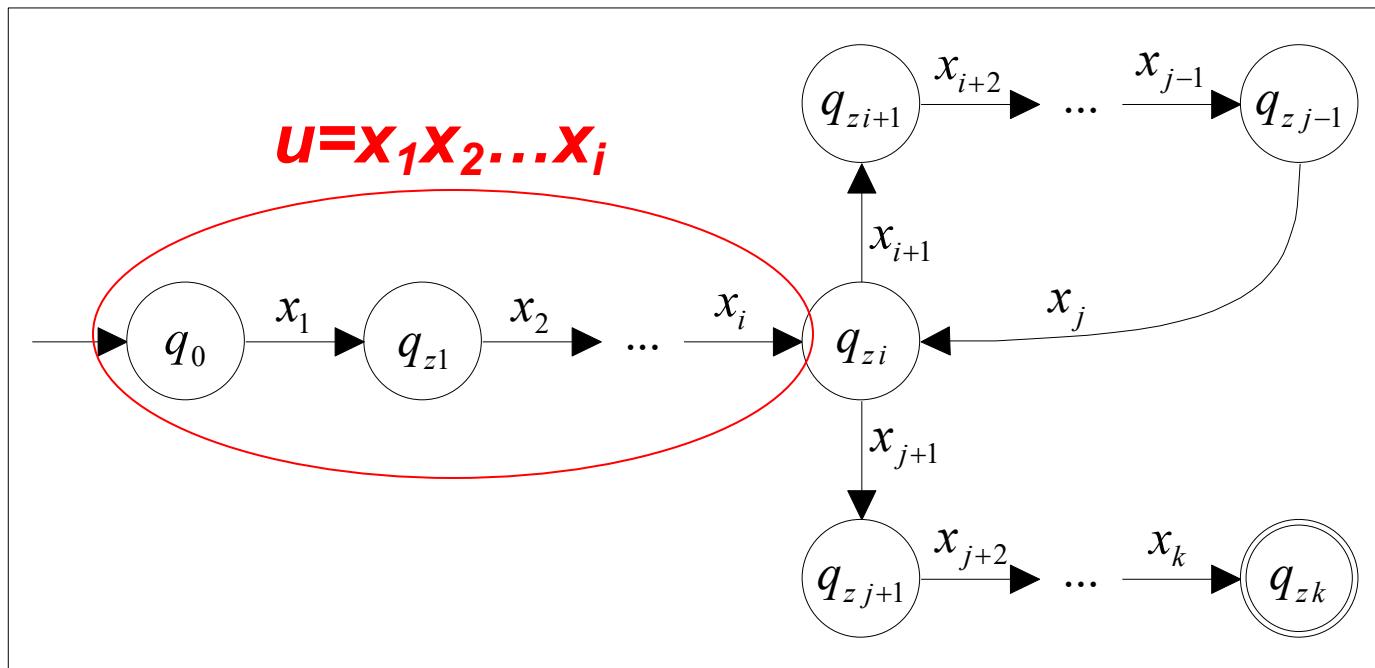


Possiamo scrivere  $z$  nella forma:  $z = uvw$

# Pumping Lemma per i linguaggi regolari

## ■ Dimostrazione

Si ha dunque la situazione rappresentata in figura:

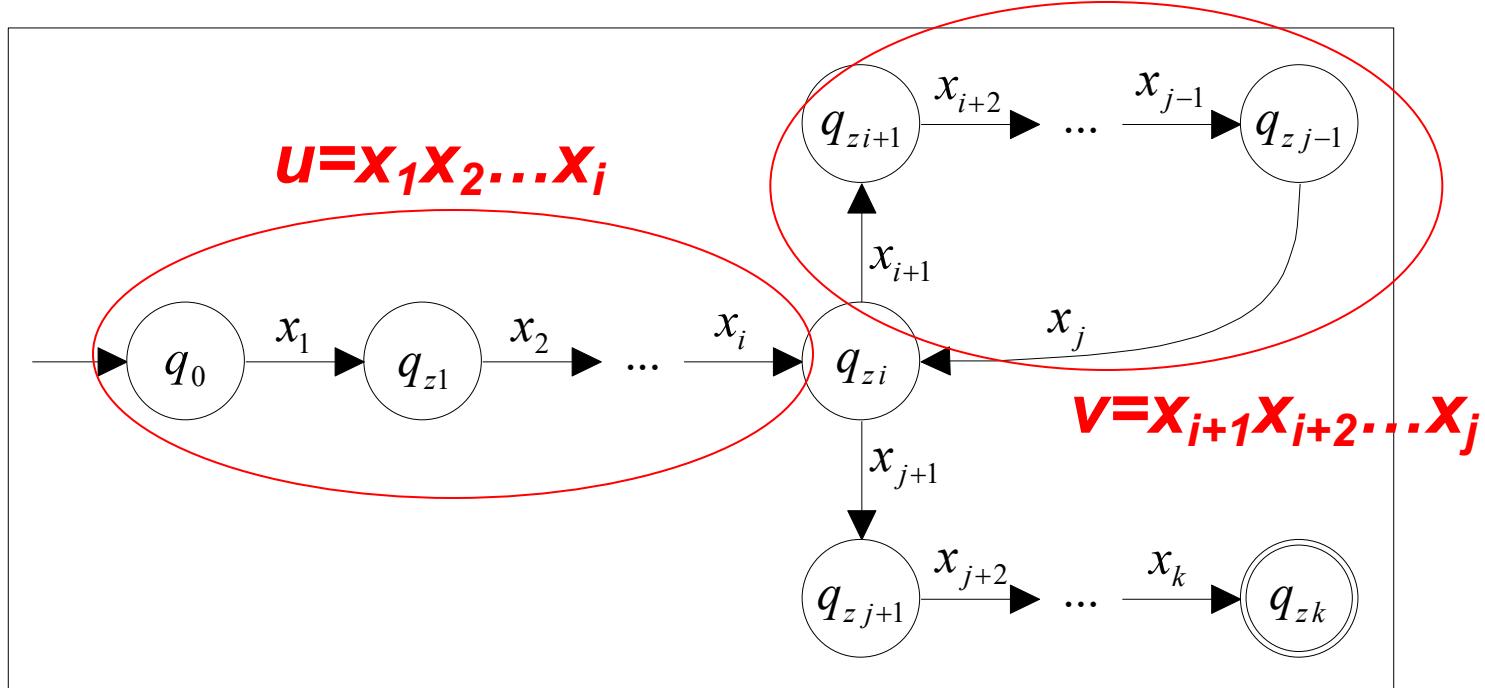


Possiamo scrivere  $z$  nella forma:  $z = uvw$

# Pumping Lemma per i linguaggi regolari

## ■ Dimostrazione

Si ha dunque la situazione rappresentata in figura:

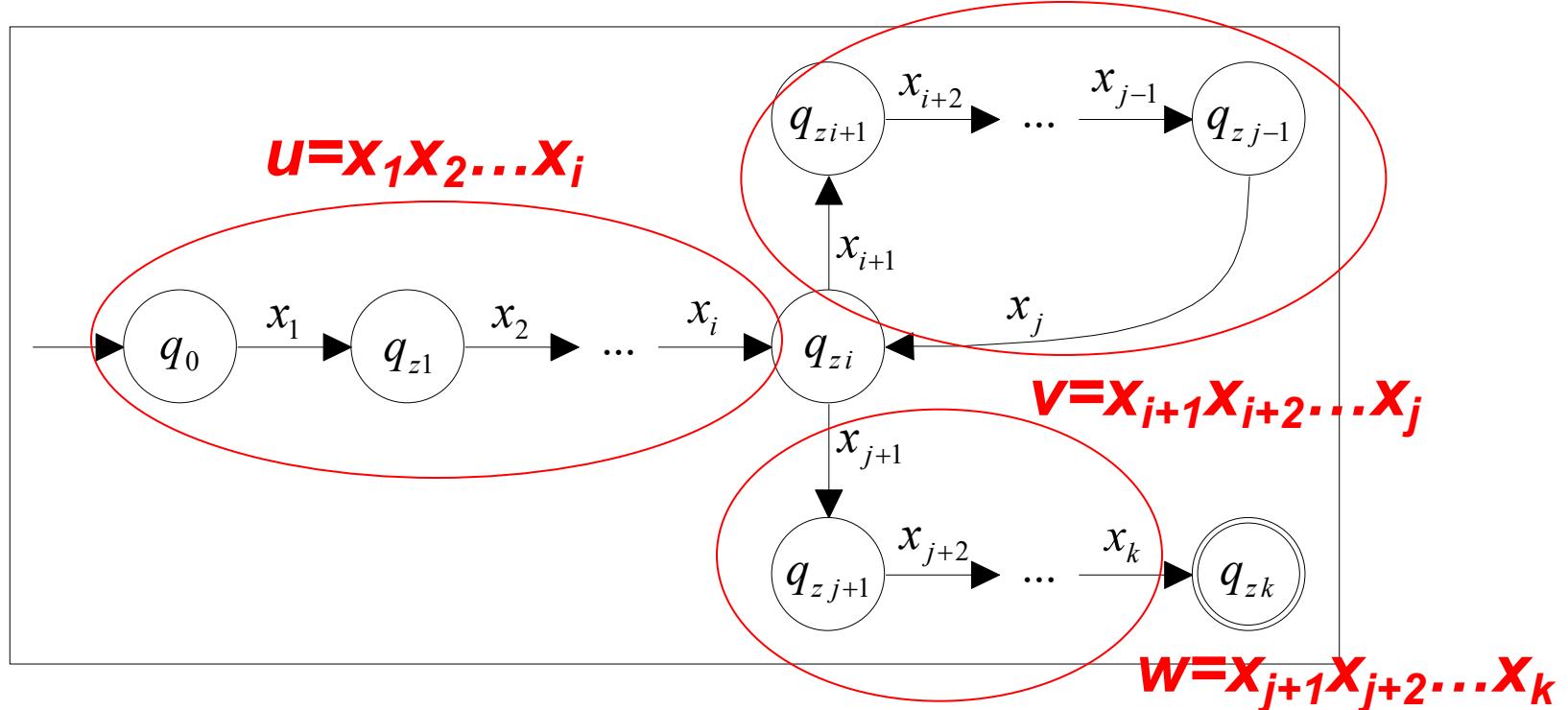


Possiamo scrivere  $z$  nella forma:  $z = uvw$

# Pumping Lemma per i linguaggi regolari

## ■ Dimostrazione

Si ha dunque la situazione rappresentata in figura:



Possiamo scrivere  $z$  nella forma:  $z = uvw$

# Pumping Lemma per i linguaggi regolari

## ■ Dimostrazione

Poiché  $z \in T(M)$ , l'automa  $M$ , per effetto dell'ingresso di  $z = uvw$ , si porta per definizione in uno stato finale ( $\delta^*(q_0, z) \in F$ ).

Ma è immediato osservare che tale stato è lo stesso in cui  $M$  si porta per effetto dell'ingresso delle parole  $uw$ ,  $uv^2w, \dots uv^i w$ ,  $i \geq 0$ .

Dunque si ha:  $uv^i w \in T(M)$ ,  $i \geq 0$ .

c.v.d.

# Esercizio

Dimostrare formalmente che il seguente linguaggio

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

non è lineare destro

# Esercizio - Soluzione

- Supponiamo, per assurdo, che  $L$  sia lineare destro
- Per il teorema di Kleene,  $\mathcal{L}_3 \equiv \mathcal{L}_{FSL} \equiv \mathcal{L}_{REG}$ 
  - Quindi, se  $L$  è lineare è anche regolare ed anche a stati finiti
- Quindi:  $\exists M = (Q, \delta, q_0, F)$  FSA su  $X = (a, b, c)$  tale che  
$$L = T(M)$$
- Sia  $p = |Q|$ , ossia  $p$  è il numero di stati di  $M$
- Per il Pumping lemma sui linguaggi regolari abbiamo che  $\forall z \in L, |z| \geq p$ 
  - $z = uvw$
  - $|uv| \leq p$
  - $v \neq \lambda$
  - $\forall k \geq 0: uv^k w \in T(M)$

## Esercizio - Soluzione

- Consideriamo una parola in  $L$

$$z = a^p b^p$$

- Avremo che

- $|z| = 2p > p$
  - $z$  deve essere accettata da  $M$

## Esercizio - Soluzione

- L'automa  $M$  parte dallo stato  $q_0$  e legge una  $a$  per volta
- Dopo aver letto la prima  $a$  si porta in  $q_1$ , dopo la seconda  $a$  in  $q_2, \dots$ , dopo la  $p$ -sima  $a$  si porta in  $q_p$

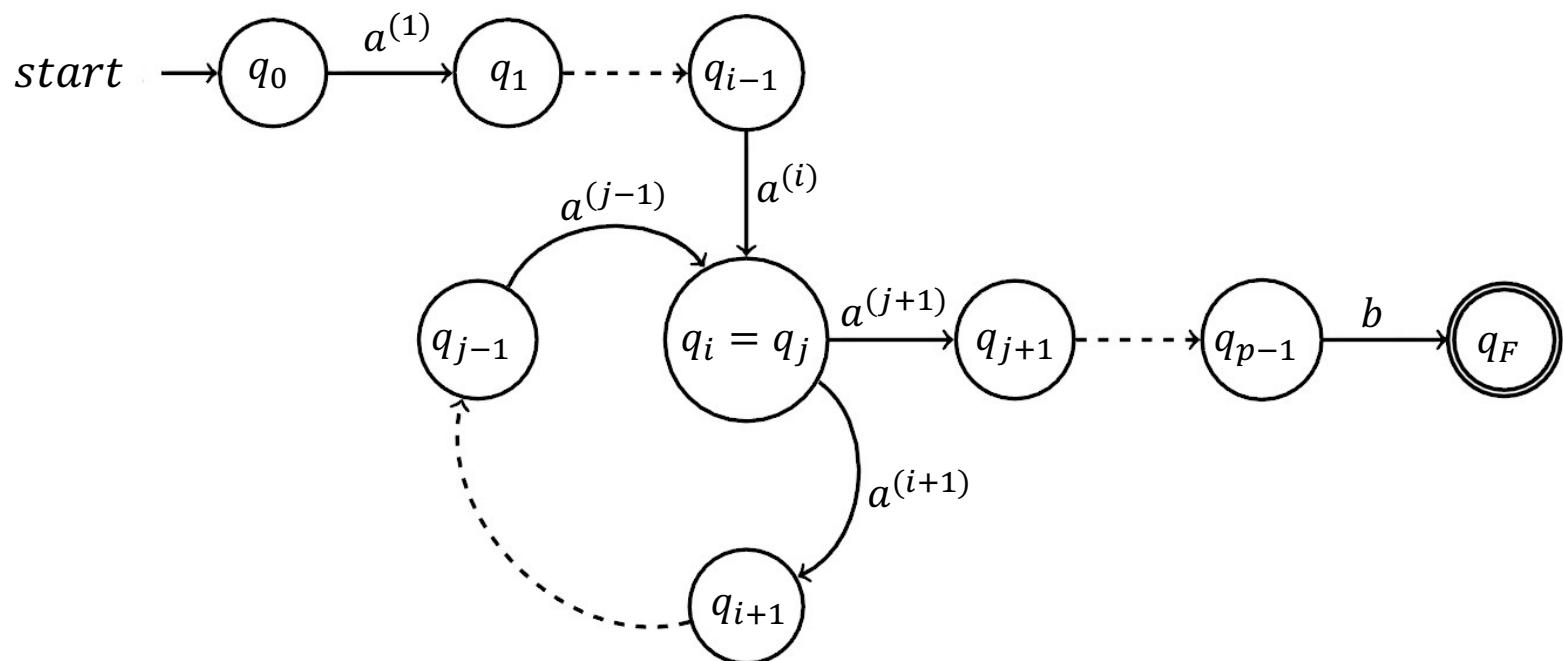
## Esercizio - Soluzione

- Per riconoscere le prime  $p a$ , l'automa  $M$  ha bisogno di transitare in  $p + 1$  stati

$$p a \left\{ \begin{array}{l} q_0 \xrightarrow{a} q_1 \\ q_1 \xrightarrow{a} q_2 \\ \dots \\ q_{p-1} \xrightarrow{a} q_p \end{array} \right.$$

- Poiché  $M$  ha solo  $p$  stati si deve verificare un ciclo

# Esercizio - Soluzione



Trace di  $M$

# Esercizio - Soluzione

■ Avremo che:

- $z$  si può scrivere come

$$z = uvw = \underbrace{a^i}_{u} \underbrace{a^{j-i}}_{v} \underbrace{a^{p-j}b^p}_{w}$$

- $|uv| = |a^i a^{j-i}| \leq p$
- $a^{j-i} \neq \lambda, 0 < j - i \leq p$

■ Sia  $k = 0$ , per il lemma abbiamo che la parola depompata  $uv^k w \in T(M)$

$$t = uv^2 w = a^{p+(j-i)} b^p \in T(M)$$

■ Osservazione:  $uv^2 w \notin L$ , dato che  $\#_a(t) > \#_b(t)$

## Esercizio - Soluzione

- Abbiamo raggiunto una **contraddizione**
- $L$  non è regolare in quanto  $\exists M$  tale che  $T(M) = L$
- $L$  non è lineare destro