Calcolabilità e complessità
Modelli di computazione

Informazioni generali sul corso

Docente: Giovanni Pani

Laboratorio: Graziella De Martino

giovanni.pani@uniba.it

Ricevimento: Giovedì 11.30

o per appuntamento

Libro 1 : Introduzione alla teoria della computazione

Michael Sipser

Libro 2 : Linguaggi modelli complessità

Ausiello, d'Amore, Gambosi,

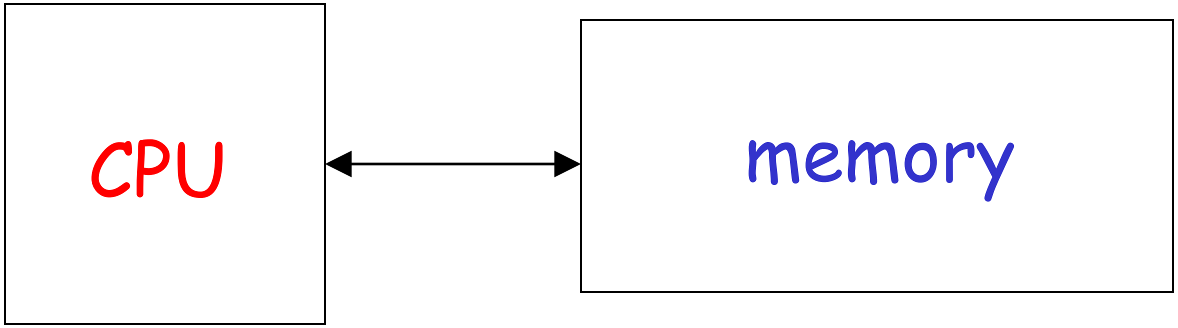
Lunedi pomeriggio
Laboratorio, portare
personal.

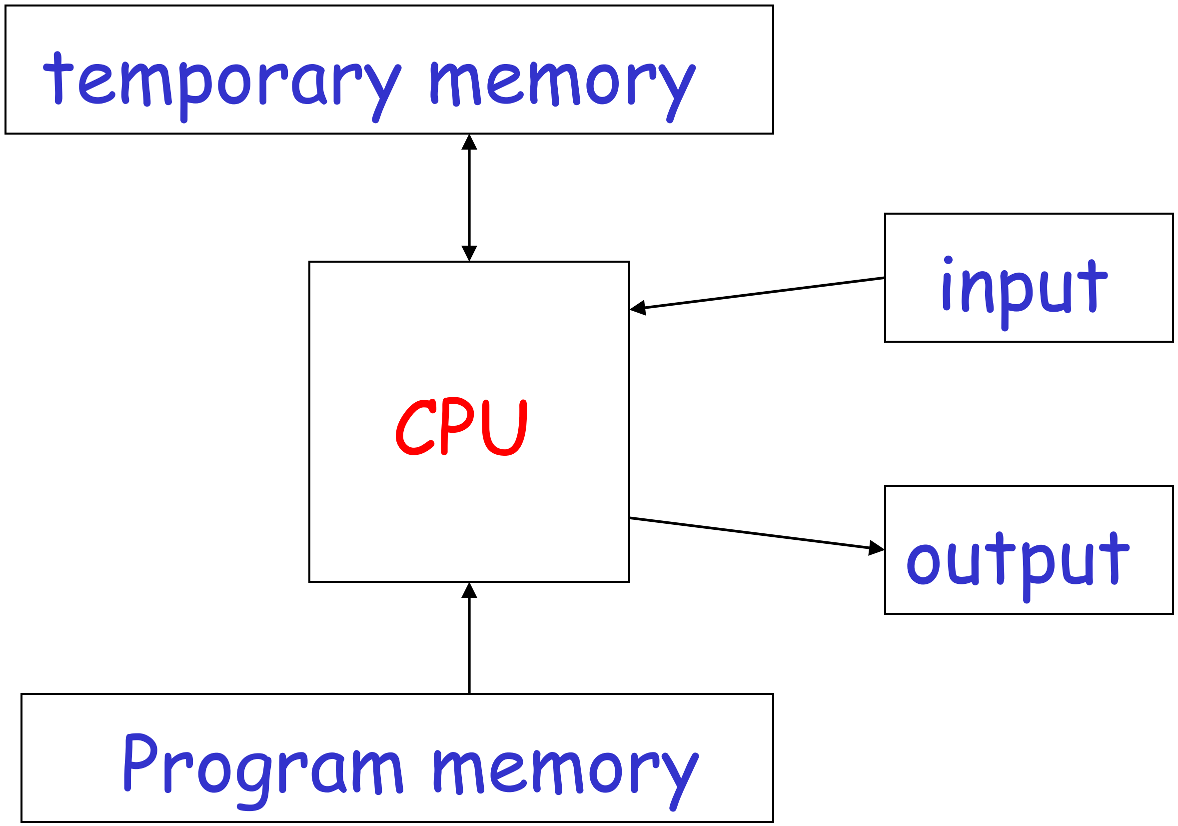
Esonero: 8 Aprile , su tutta la parte fatta.

Ada di.uniba.it
http://informatica2.di.uniba.it/
Psw CC-INF1920

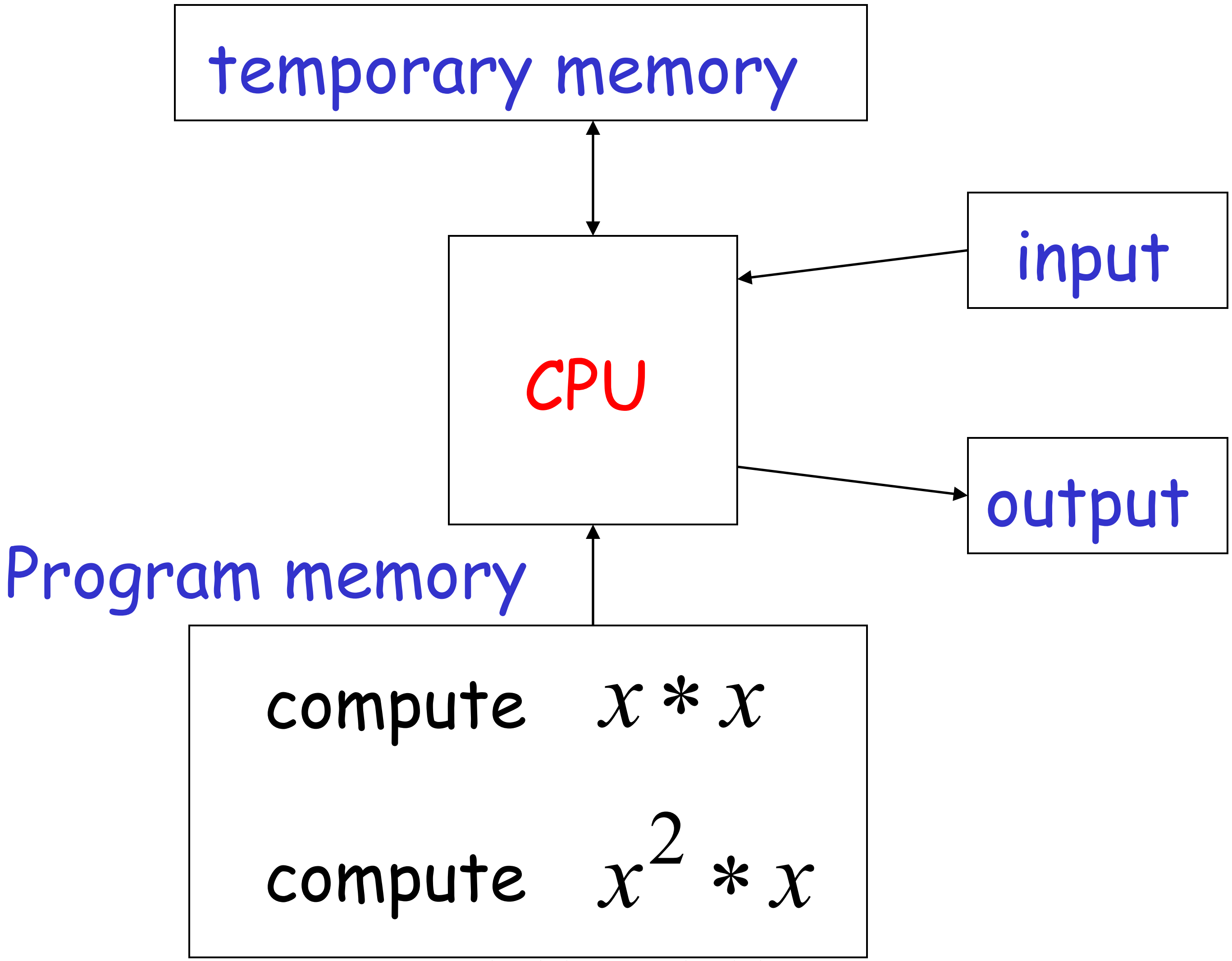
Contenuto del corso

Calcolo

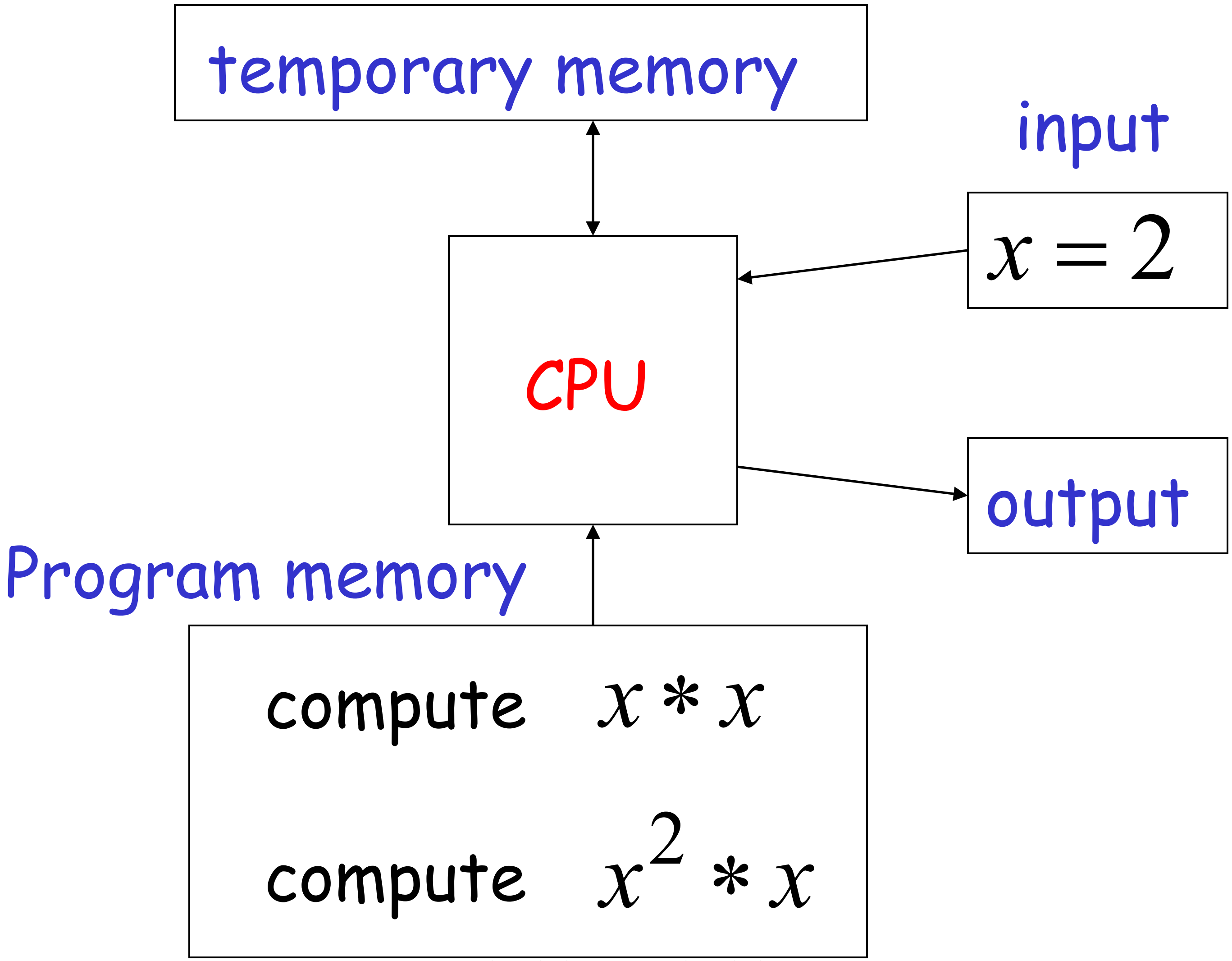



temporary memory
input
CPU
output
Program memory

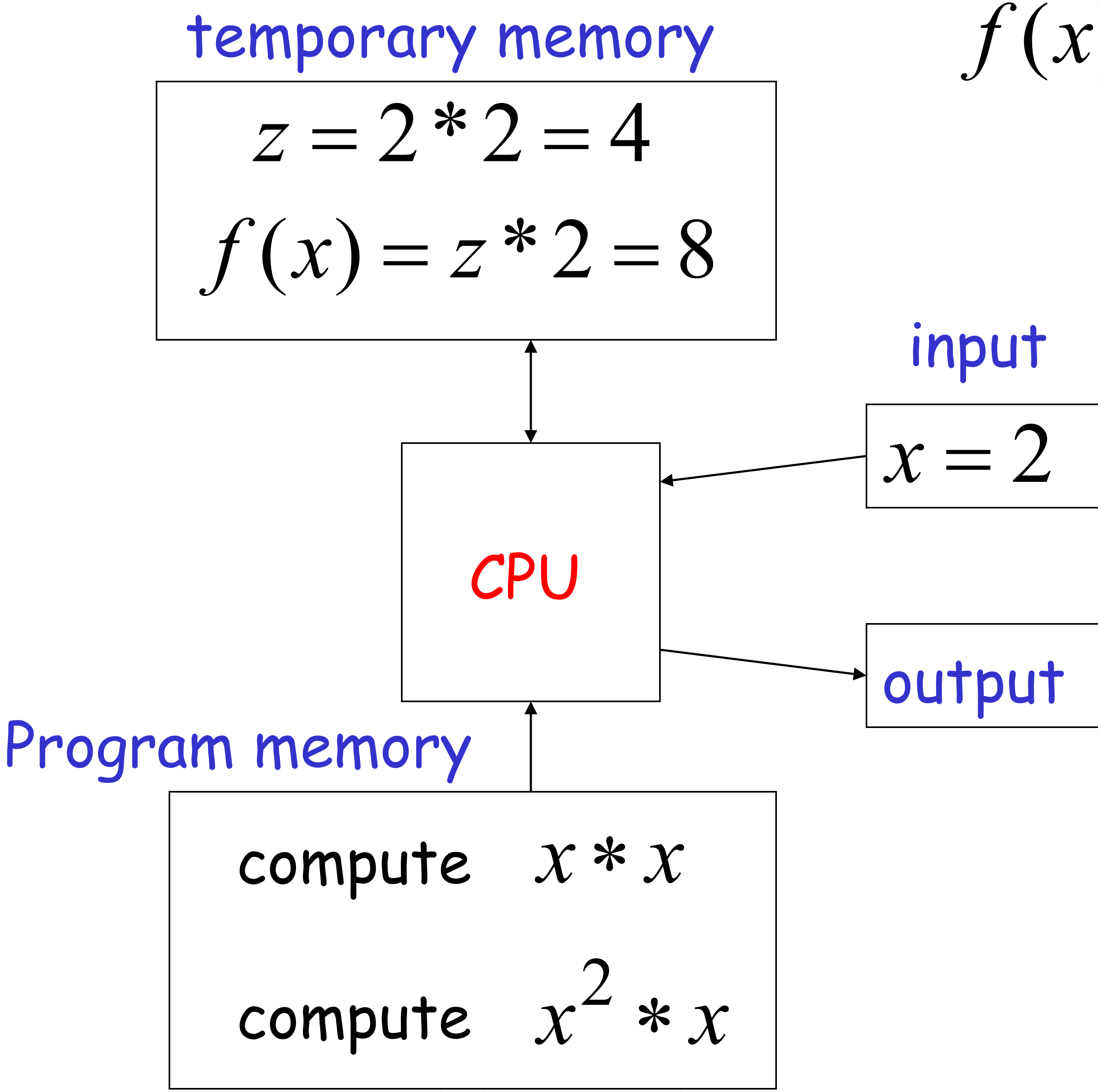
Esempio: $f(x) = x^3$



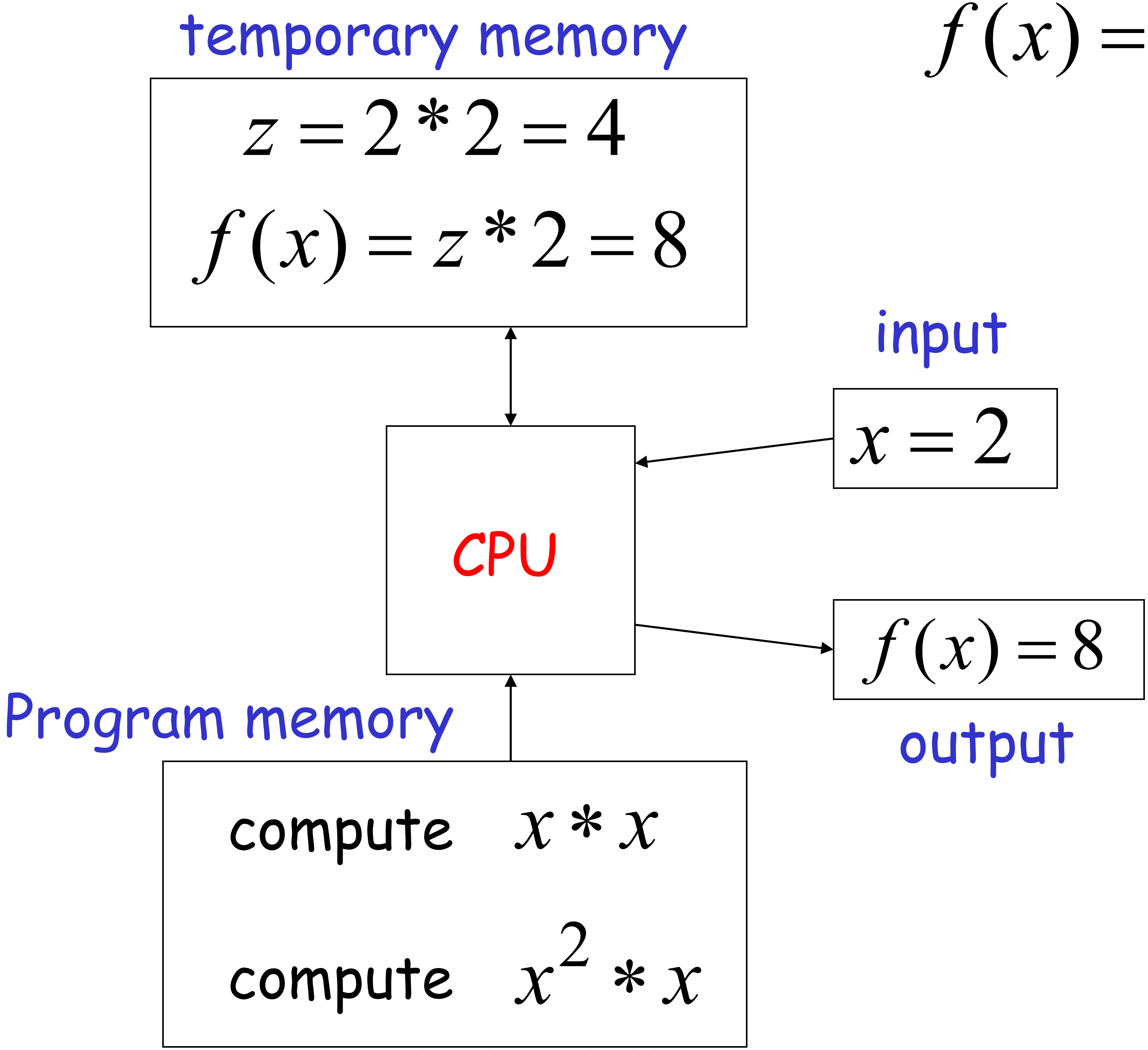
$$f(x) = x^3$$


temporary memory
input
$x = 2$
CPU
output
Program memory
compute $x * x$
compute $x^2 * x$

$$f(x) = x^3$$

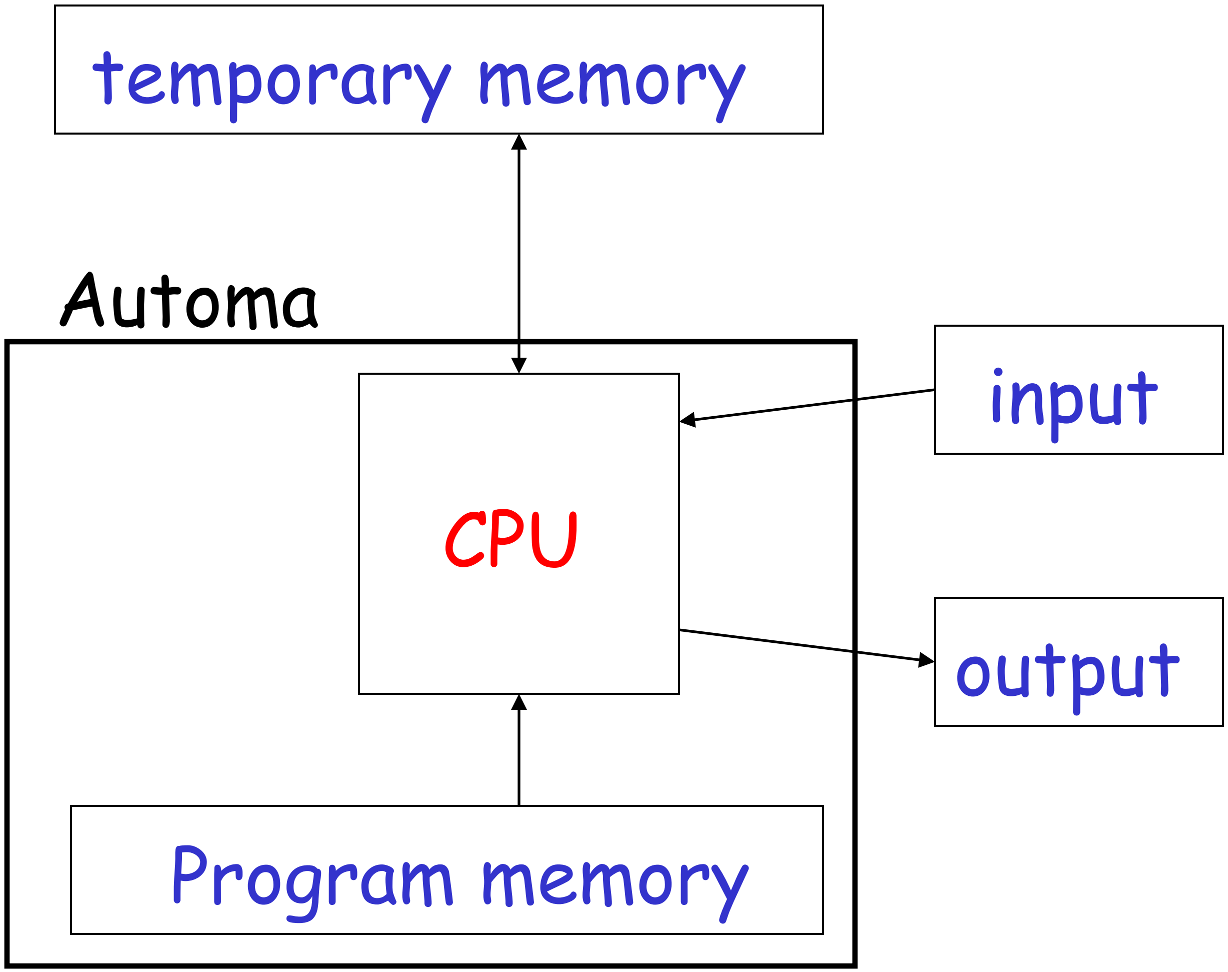

temporary memory
$z = 2 * 2 = 4$
$f(x) = z * 2 = 8$
input
$x = 2$
CPU
output
Program memory
compute $x * x$
compute $x^2 * x$

$$f(x) = x^3$$

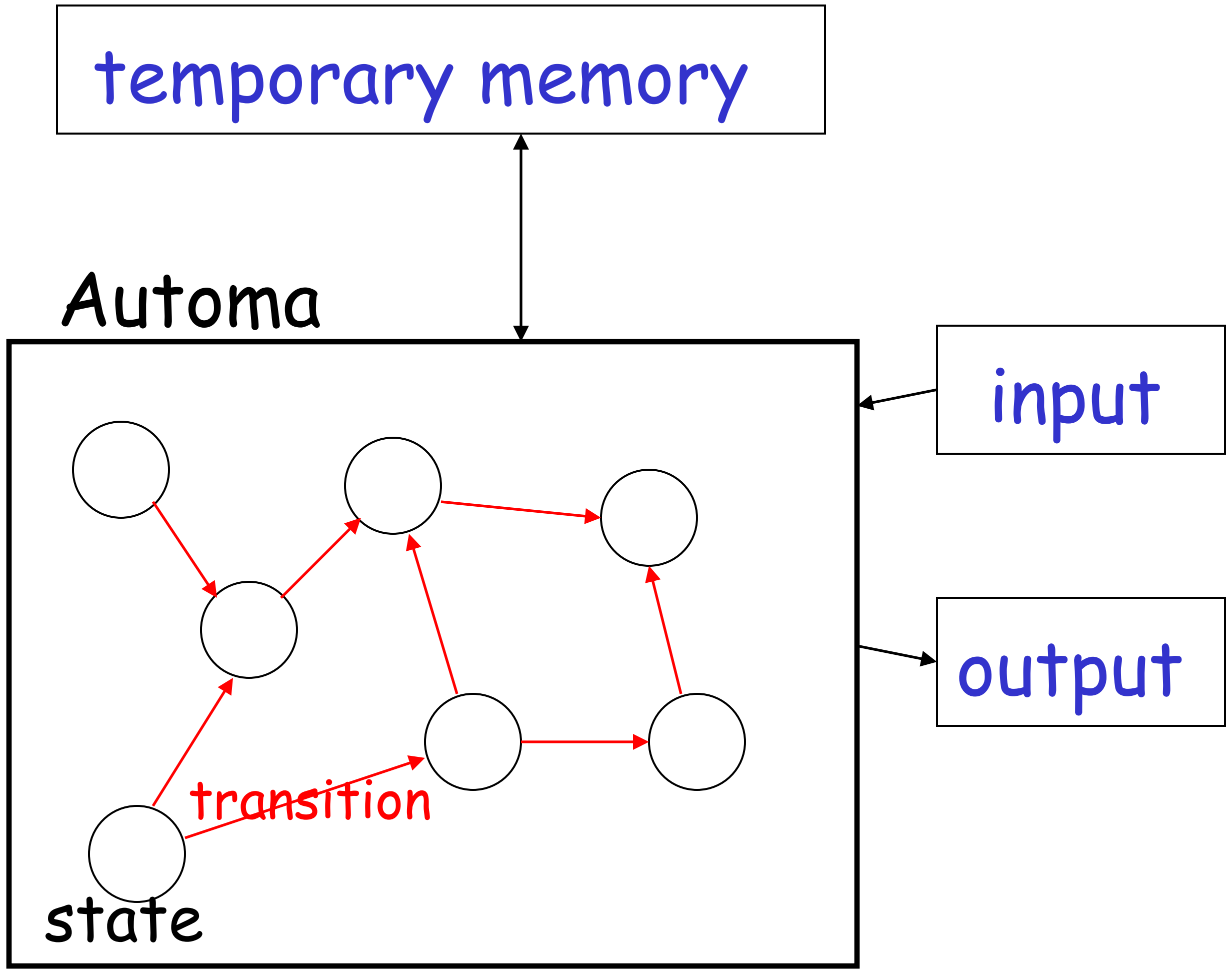


temporary memory
$z = 2*2 = 4$
$f(x) = z*2 = 8$
input
$x = 2$
CPU
$f(x) = 8$
output
Program memory
compute $x*x$
compute x^2*x

Automa



Automa

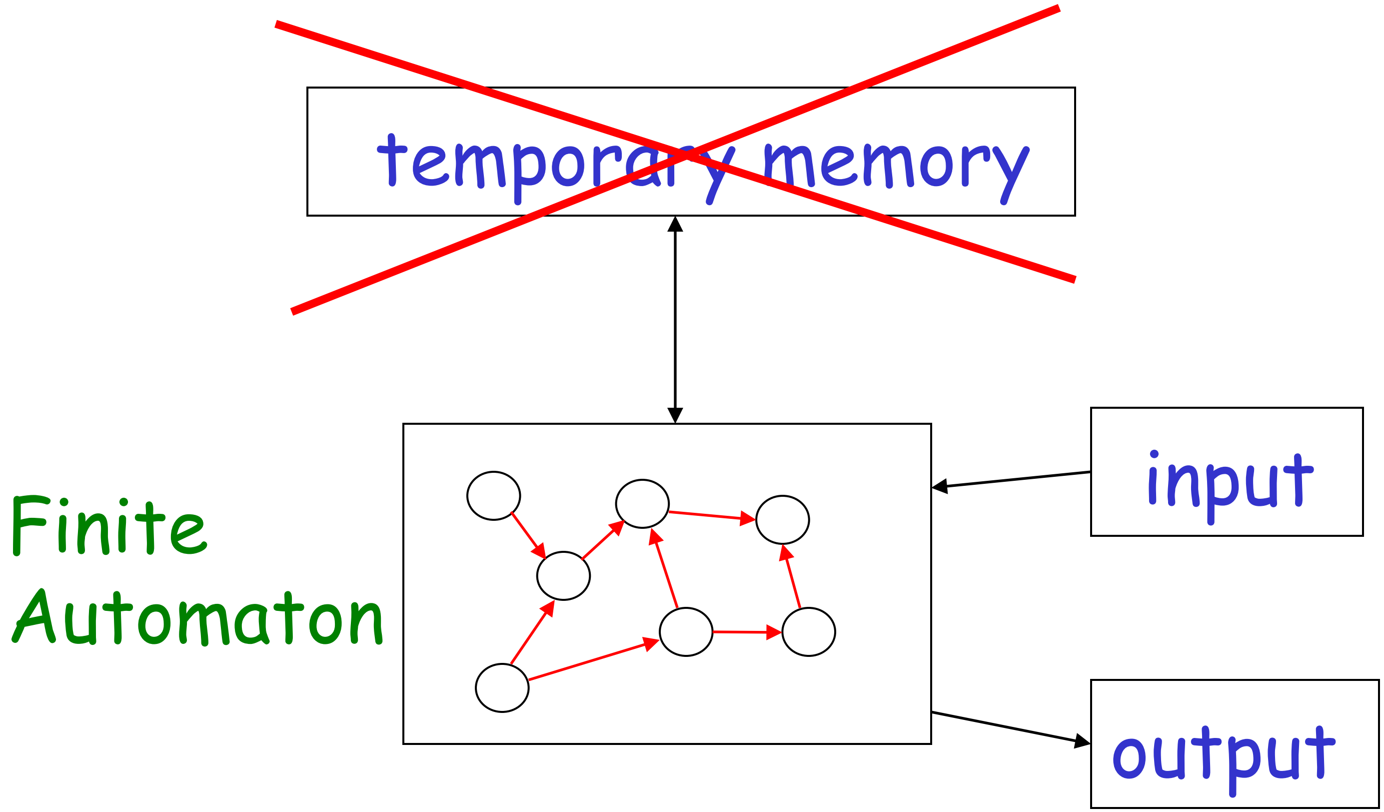


Automata

Automata si distinguono secondo il tipo di memoria

- **Finite Automata:** nessuna memoria
- **Pushdown Automata:** stack
- **Turing Machines:** random access memory

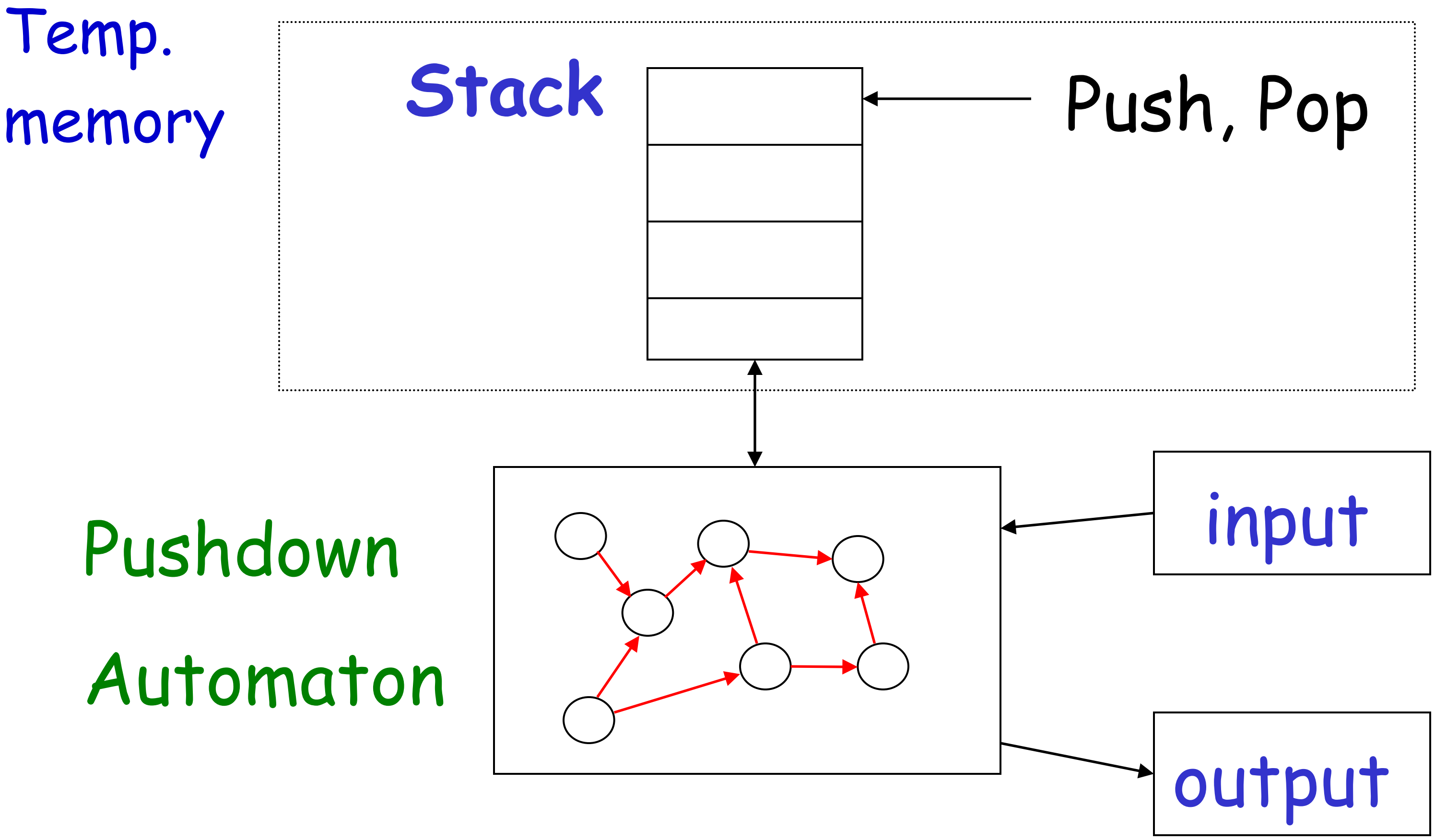
Finite Automata



Esempio: ascensori, macchine per il caffe

(piccolo potere di computazione)

Pushdown Automata

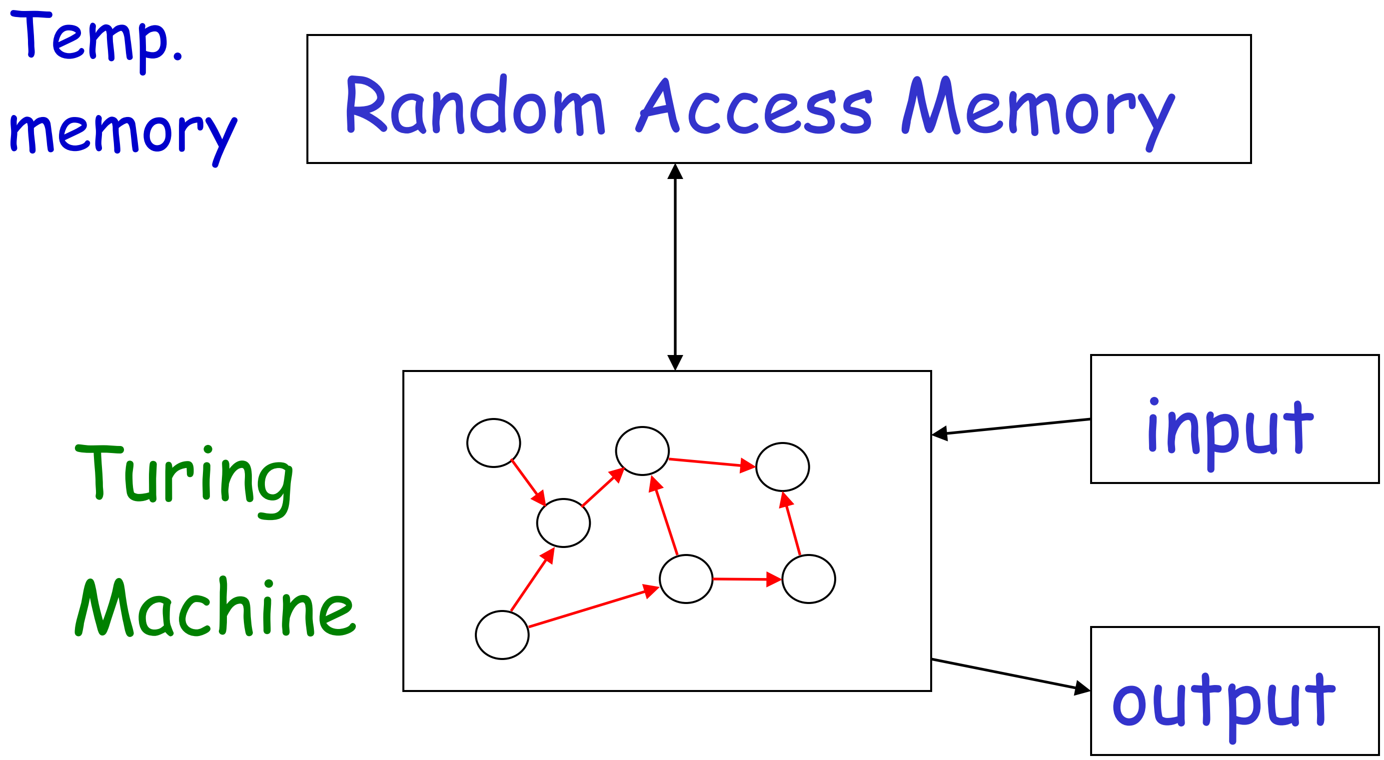


Esempio:

Compilatori per linguaggi di programmazione

(medio potere di calcolo)

Turing Machine



Esempio: qualsiasi algoritmo

(il più alto potere di calcolo)

Power of Automata

Semplici problemi	Problemi più complessi	Problemi complicati Hardest problems

Finite Automata < Pushdown Automata < Turing Machine

Meno potere ⟶ Più potere

Risolvere più problemi di calcolo

Turing Machine è il modello di calcolo più potente che è stato definito

Domanda: Esistono problemi di calcolo che non possono essere risolti?

Risposta: Si (problemi irrisolvibili)

Complessità temporale dei problemi di calcolo:

NP-complete problems

Si crede che occorre un tempo esponenziale per calcolarli

P problems

Risolti in tempo polinomiale

Preliminari matematici

Preliminari matematici

- Insiemi
- Funzioni
- Relazioni
- Grafi
- Tecniche di dimostrazioni

SETS

A insieme è una collezione di elementi

$$A = \{1, 2, 3\}$$

$$B = \{train, bus, bicycle, airplane\}$$

Scriveremo:

$$1 \in A$$

$$ship \notin B$$

Rappresentazione degli insiemi

C = { a, b, c, d, e, f, g, h, i, j, k }

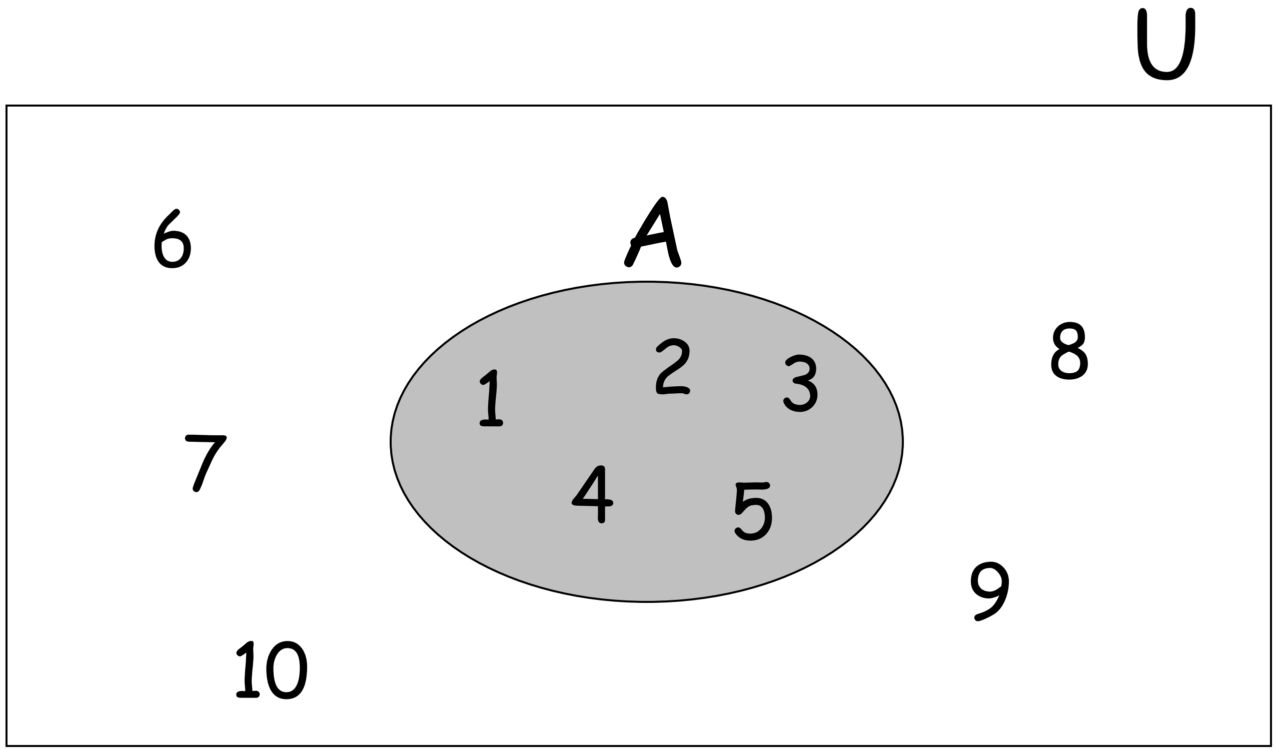
C = { a, b, ..., k } → *Insieme finito*

S = { 2, 4, 6, ... } → *Insieme infinito*

S = { j : j > 0, e j = 2k per qualchek>0 }

S = { j : j è non negativo e pari}

$A = \{1, 2, 3, 4, 5\}$

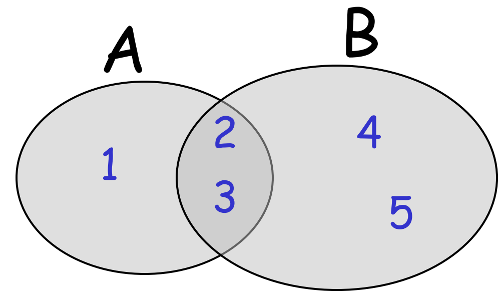


Insiemi universale: tutti gli elementi possibili

$U = \{1, \ldots, 10\}$

Operazione sugli insiemi

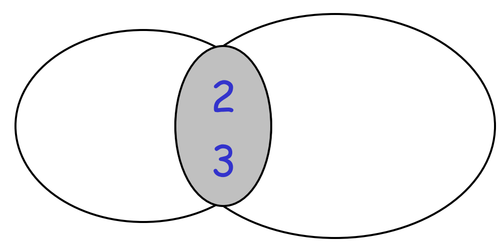
$A = \{ 1, 2, 3 \}$ $B = \{ 2, 3, 4, 5 \}$



- Unione

$A \cup B = \{ 1, 2, 3, 4, 5 \}$

- Intersezione

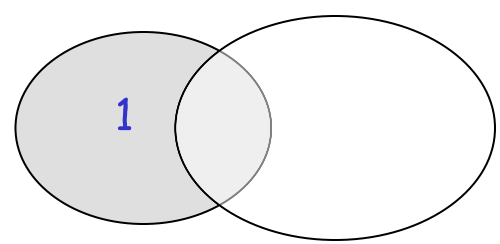


$A \cap B = \{ 2, 3 \}$

- Differenza

$A - B = \{ 1 \}$

$B - A = \{ 4, 5 \}$

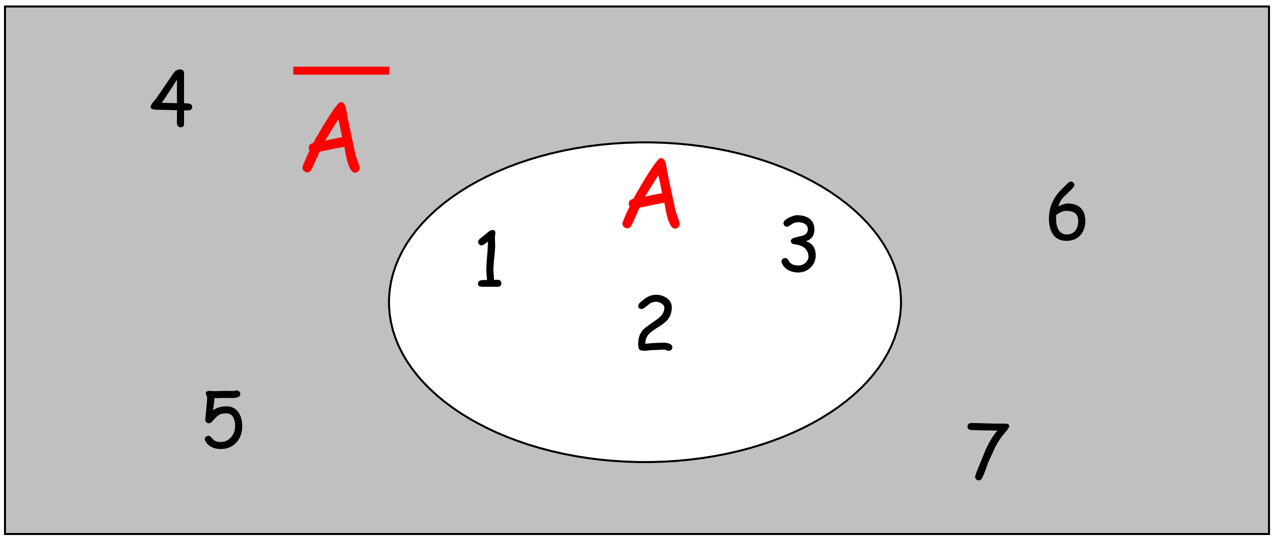


Venn diagrams

- Complemento

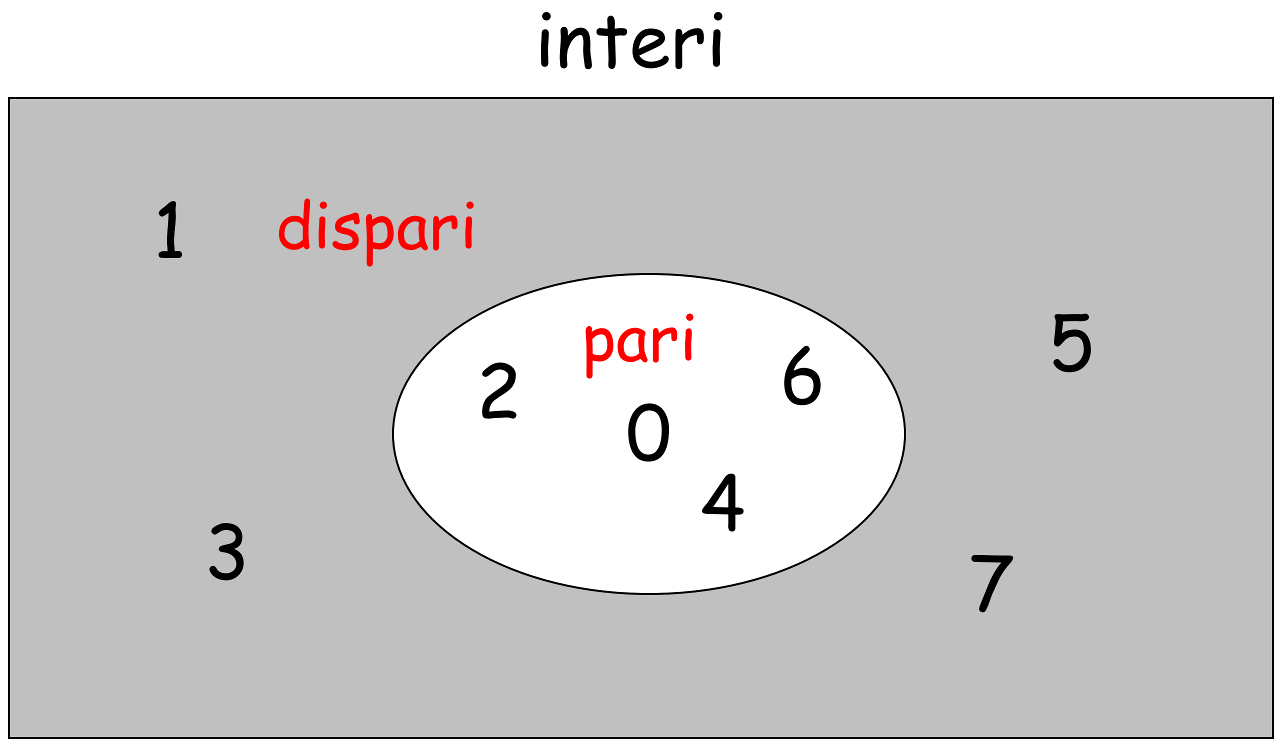
Insieme universale= {1, …, 7}

$A = \{1, 2, 3\} \Rightarrow \overline{A} = \{4, 5, 6, 7\}$



$$\overline{\overline{A}} = A$$

$$\overline{\{\text{ interi pari}\}} = \{\text{ interi dispari}\}$$



Leggi di DeMorgan

$$\overline{A \cup B} = \overline{A} \cap \overline{B}$$

$$\overline{A \cap B} = \overline{A} \cup \overline{B}$$

Vuoto, insieme nullo: $\emptyset$

$\emptyset = \{\ \}$

$S \cup \emptyset = S$

$S \cap \emptyset = \emptyset$

$\overline{\emptyset}$ = Universal Set

$S - \emptyset = S$

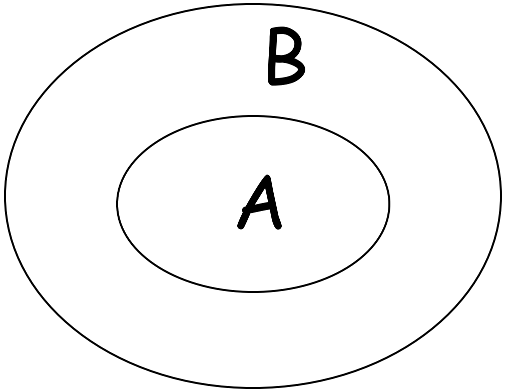
$\emptyset - S = \emptyset$

Sottoinsieme

$A = \{ 1, 2, 3\}$ $B = \{ 1, 2, 3, 4, 5 \}$

$$A \subseteq B$$

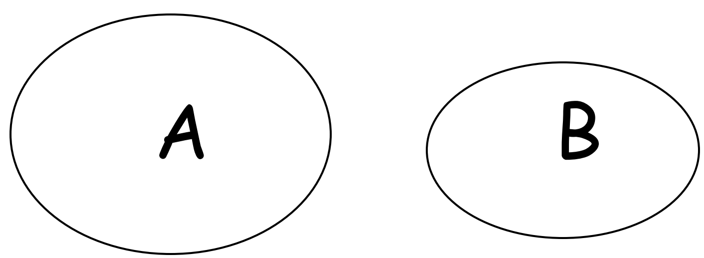
Sottoinsieme proprio: $A \subset B$



Insieme disgiunti

$A = \{ 1, 2, 3 \}$ $B = \{ 5, 6 \}$

$$A \cap B = \emptyset$$



Cardinalità

- per gli insiemi finiti

$A = \{ 2, 5, 7 \}$

$|A| = 3$

(dimensione dell'insieme)

Insieme potenza

Un insieme potenza è un insieme di insiemi

$S = \{ a, b, c \}$

Potenza di S = l'insieme di tutti I sottoinsiemi di S

$2^S = \{ \emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\} \}$

Osservazione: $| 2^S | = 2^{|S|}$ $(8 = 2^3)$

Prodotto Cartesiano

$A = \{ 2, 4 \}$ $B = \{ 2, 3, 5 \}$

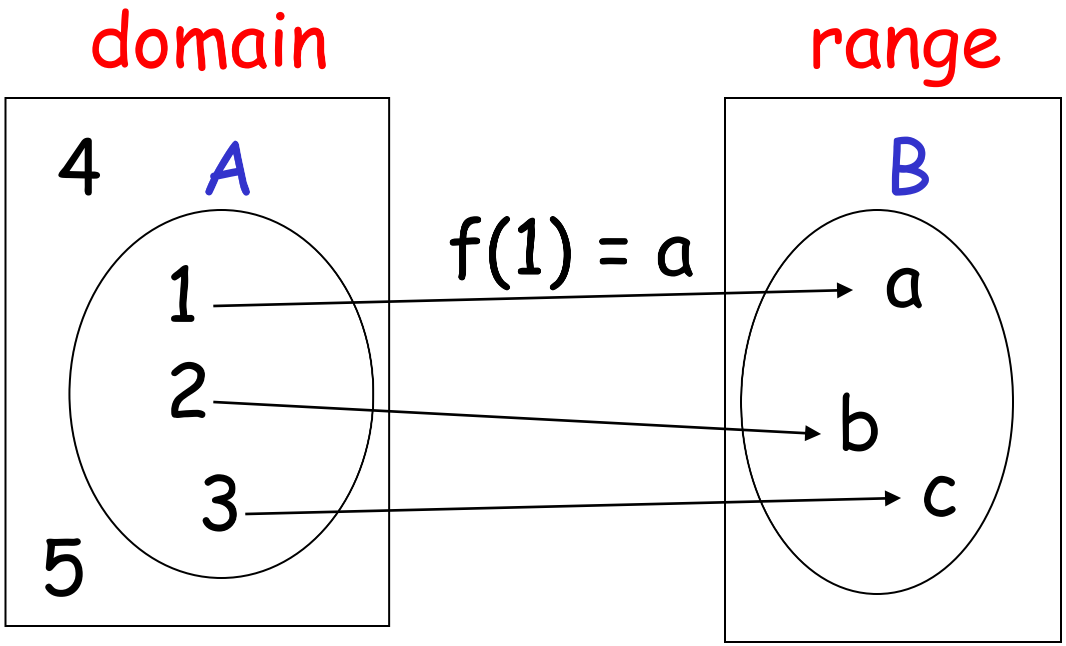
$A \times B = \{ (2, 2), (2, 3), (2, 5), (4, 2), (4, 3), (4, 5) \}$

$|A \times B| = |A| \, |B|$

Possiamo generalizzarlo a più insiemi

$A \times B \times \ldots \times Z$

Funzioni



f : A -> B

Se A = dominio

allora f è una funzione totale

altrimenti f è una funzione parziale

Relazioni

$R = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots\}$

$x_i \, R \, y_i$

per esempio . se R = '>': $2 > 1$, $3 > 2$, $3 > 1$

Relazioni di equivalenza

- Riflessiva: $x \; R \; x$
- Simmetrica: $x \; R \; y \Rightarrow y \; R \; x$
- Transitiva: $x \; R \; y$ and $y \; R \; z \Rightarrow x \; R \; z$

Esempio: R = '='

- $x = x$
- $x = y \Rightarrow y = x$
- $x = y$ e $y = z \Rightarrow x = z$

Classi di equivalenza

Data la relazione di equivalenza R

la classe di equivalenza per $x = \{y : x\ R\ y\}$

Esempio:

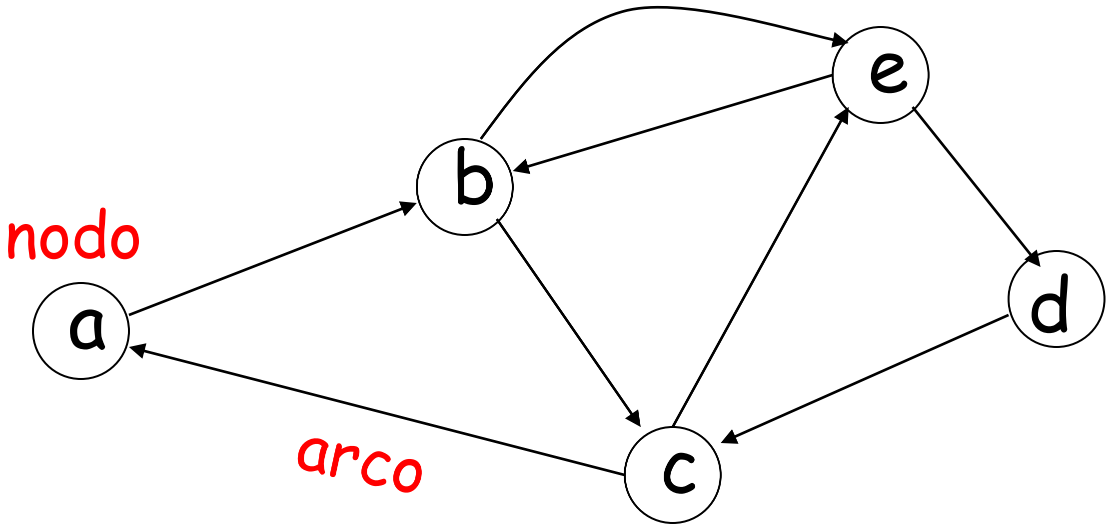
$$R = \{ (1, 1), (2, 2), (1, 2), (2, 1), (3, 3), (4, 4), (3, 4), (4, 3) \}$$

classe di equivalenza per $1 = \{1, 2\}$

classe di equivalenza per $3 = \{3, 4\}$

Grafi

Grafo diretto



- Nodi (Vertici)

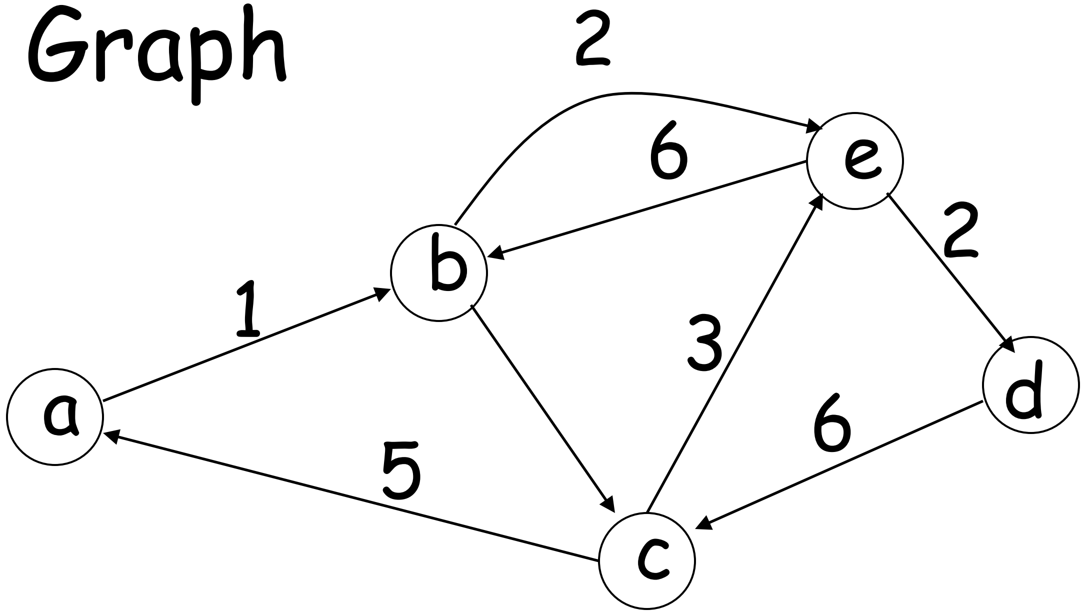
$$V = \{ a, b, c, d, e \}$$

- Archi

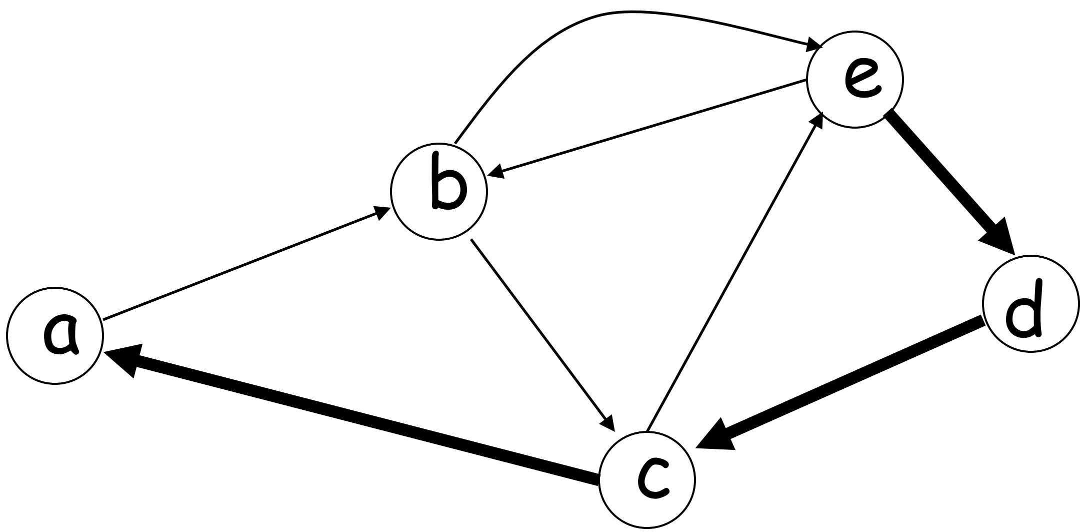
$$E = \{ (a,b), (b,c), (b,e), (c,a), (c,e), (d,c), (e,b), (e,d) \}$$

Grafo con etichette

Labeled Graph



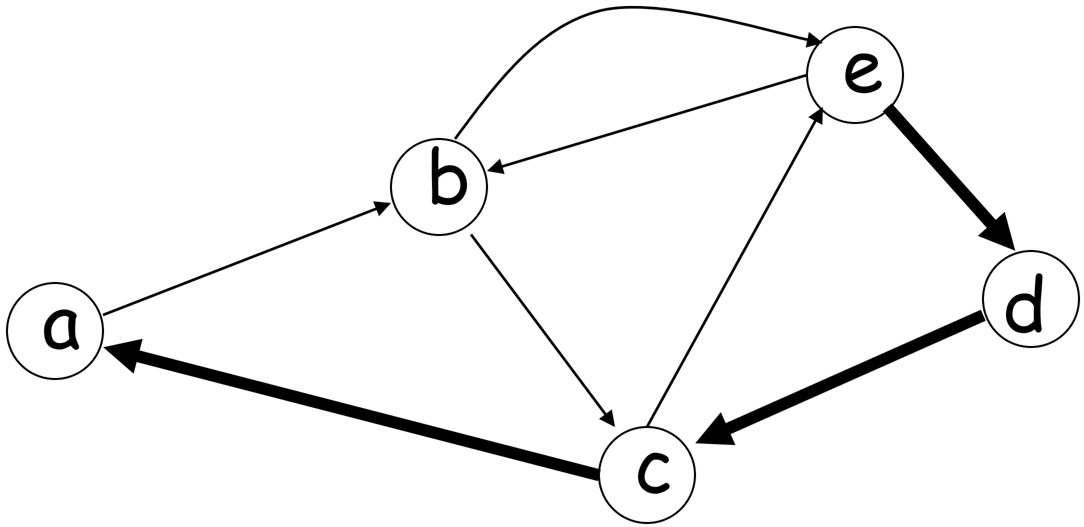
Cammino



Un cammino è una sequenza di archi adiacenti

(e, d), (d, c), (c, a)

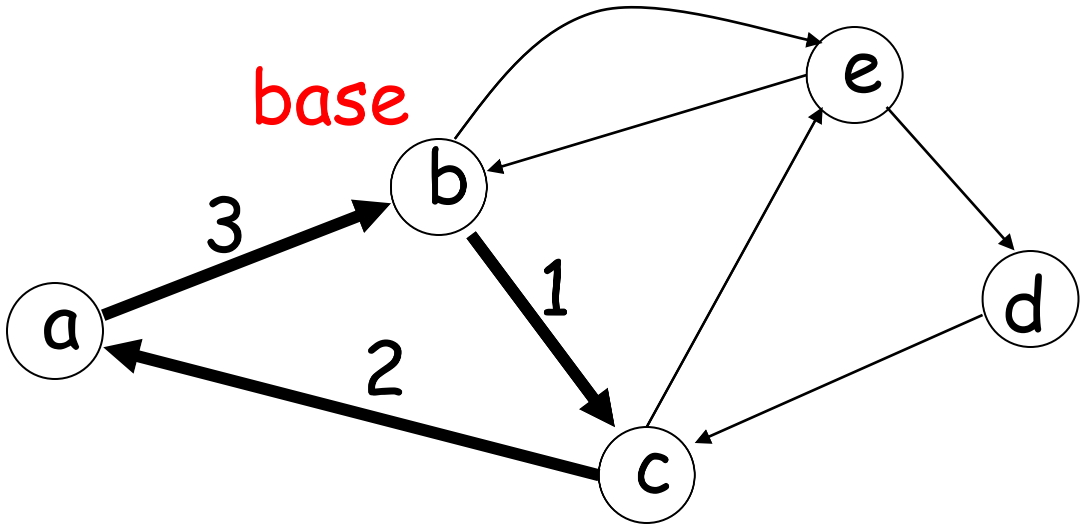
Path



Path è un cammino in cui nessun arco è ripetuto

Simple path : nessun nodo è ripetuto

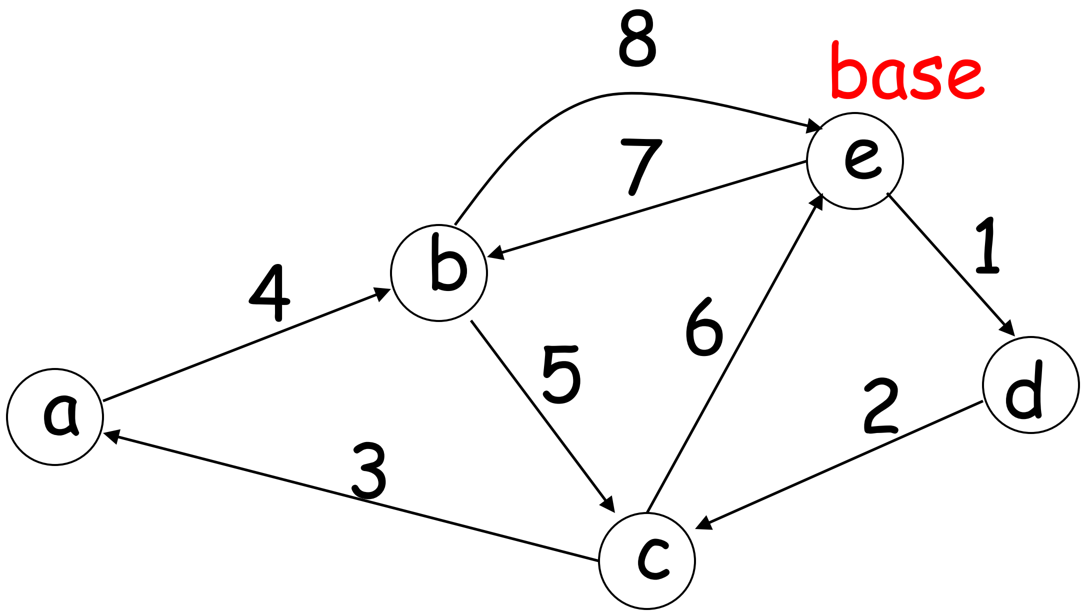
Ciclo



Ciclo: un cammino da un nodo(base) a se stesso

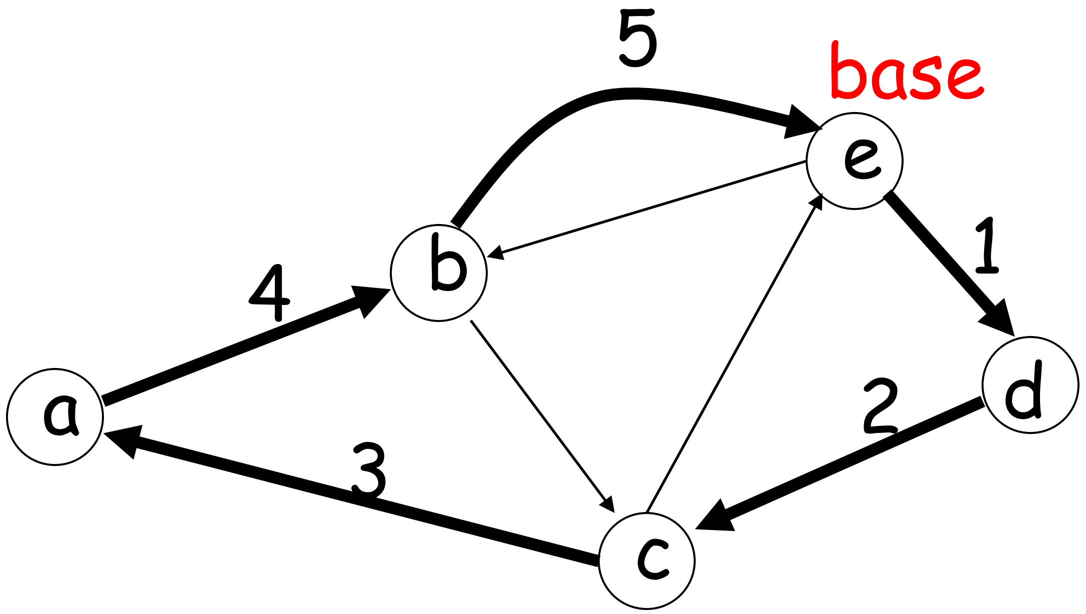
Ciclo semplice: solo la base è ripetuta

Euler Tour



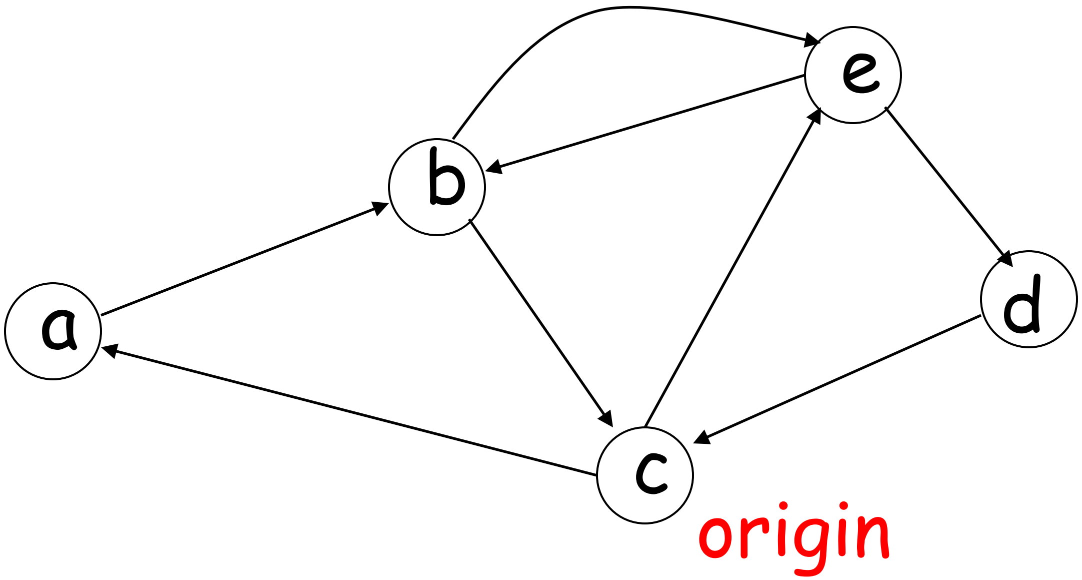
Un ciclo che contiene ogni arco una sola volta

Ciclo Hamiltonian

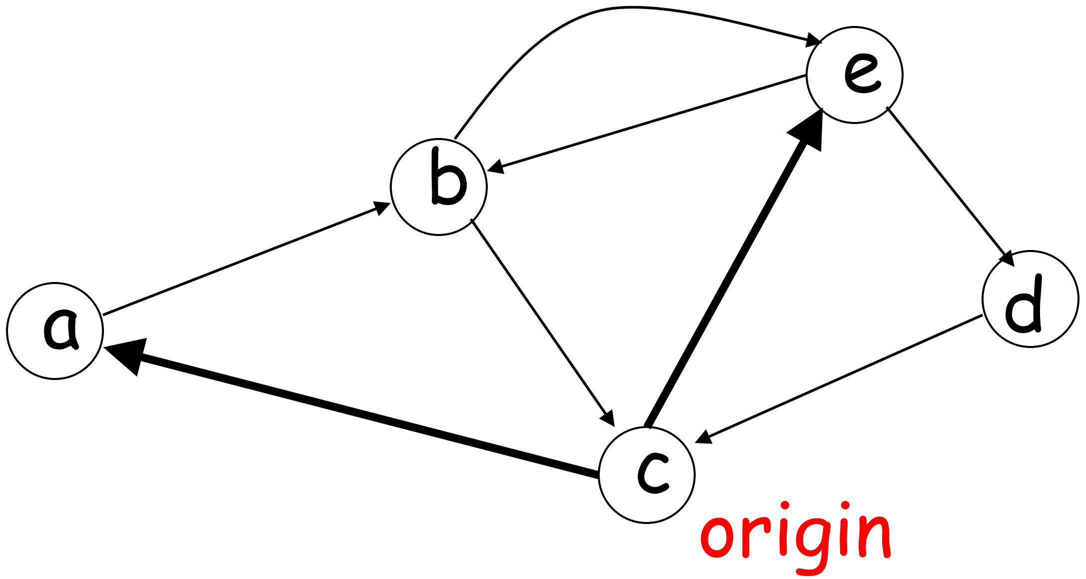


Un ciclo semplice che contiene tutti i nodi

Trovare tutti I path semplici



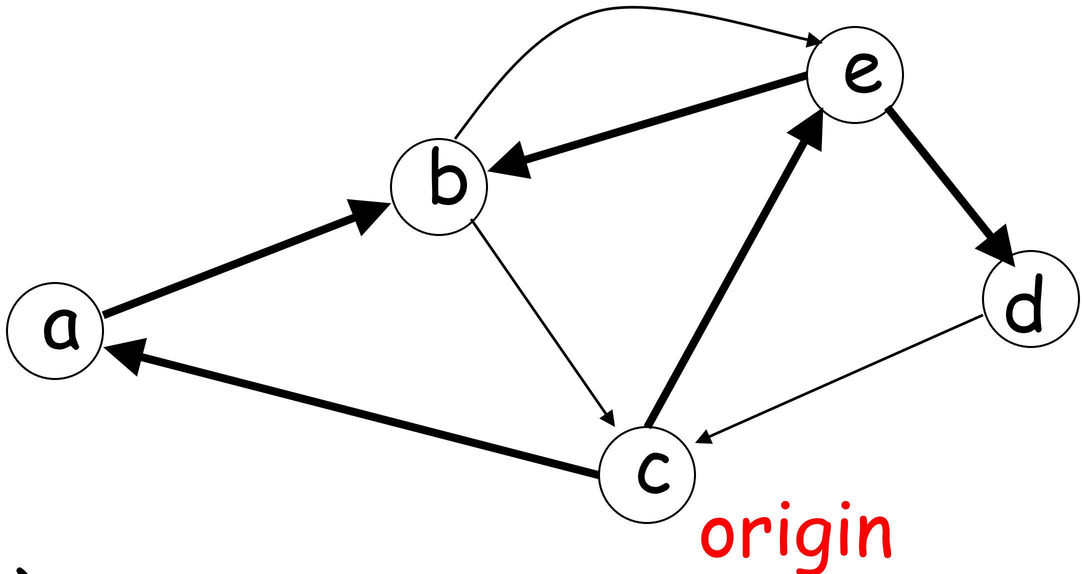
Step 1



(c, a)

(c, e)

Step 2



(c, a)

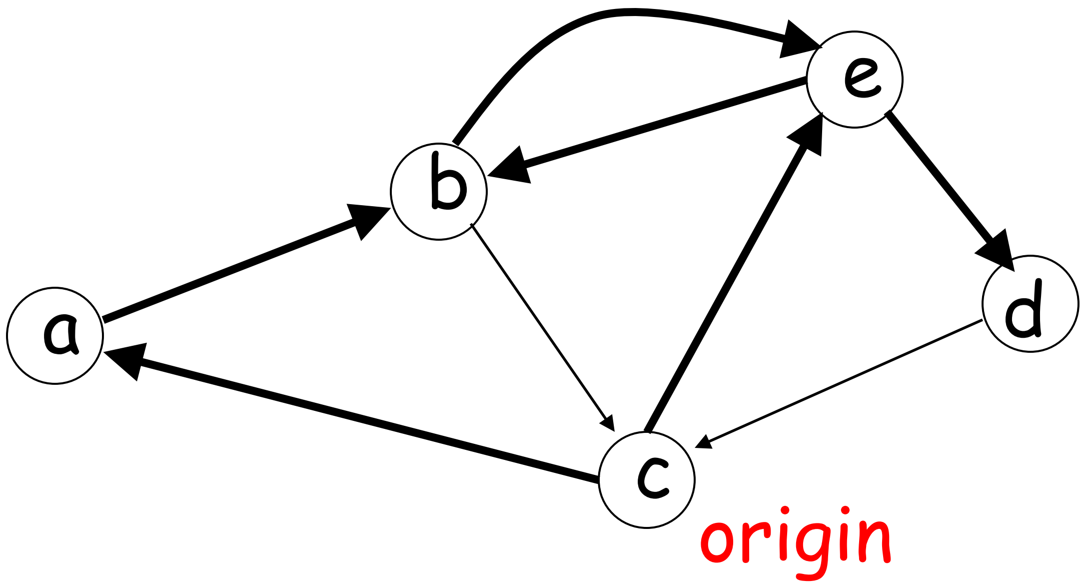
(c, a), (a, b)

(c, e)

(c, e), (e, b)

(c, e), (e, d)

Step 3



(c, a)

(c, a), (a, b)

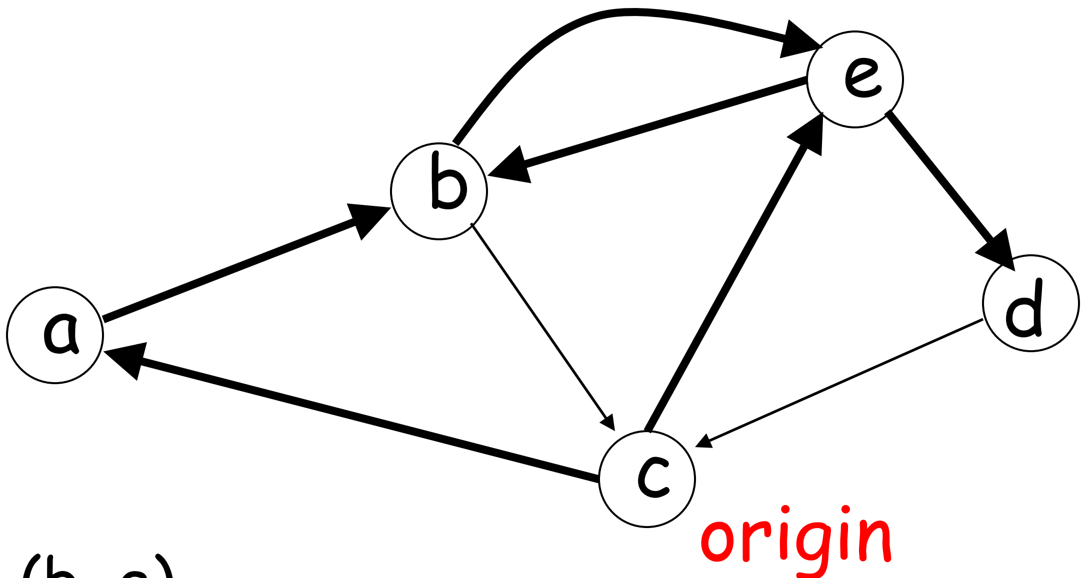
(c, a), (a, b), (b, e)

(c, e)

(c, e), (e, b)

(c, e), (e, d)

Step 4



(c, a)

(c, a), (a, b)

(c, a), (a, b), (b, e)

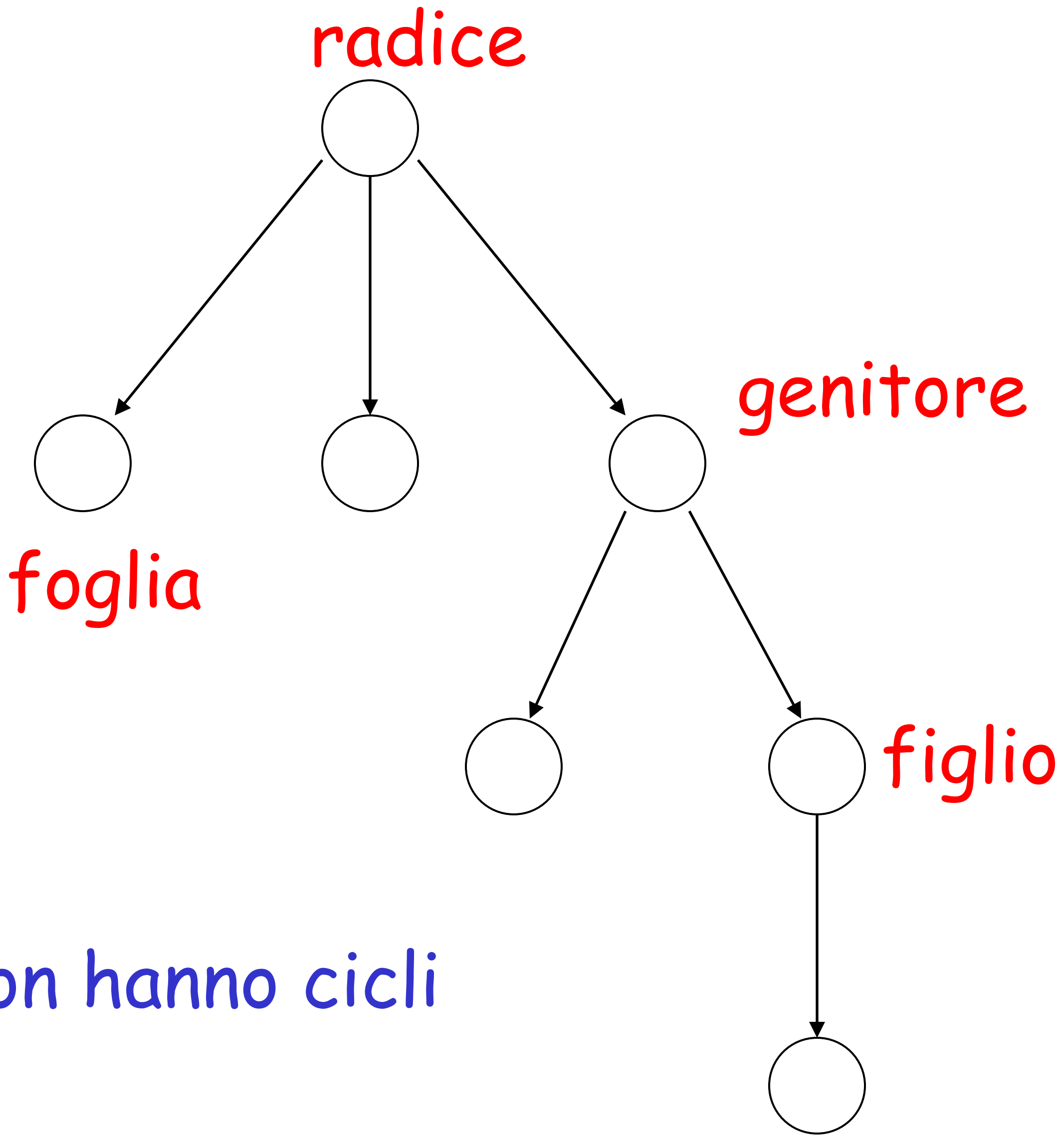
(c, a), (a, b), (b, e), (e,d)

(c, e)

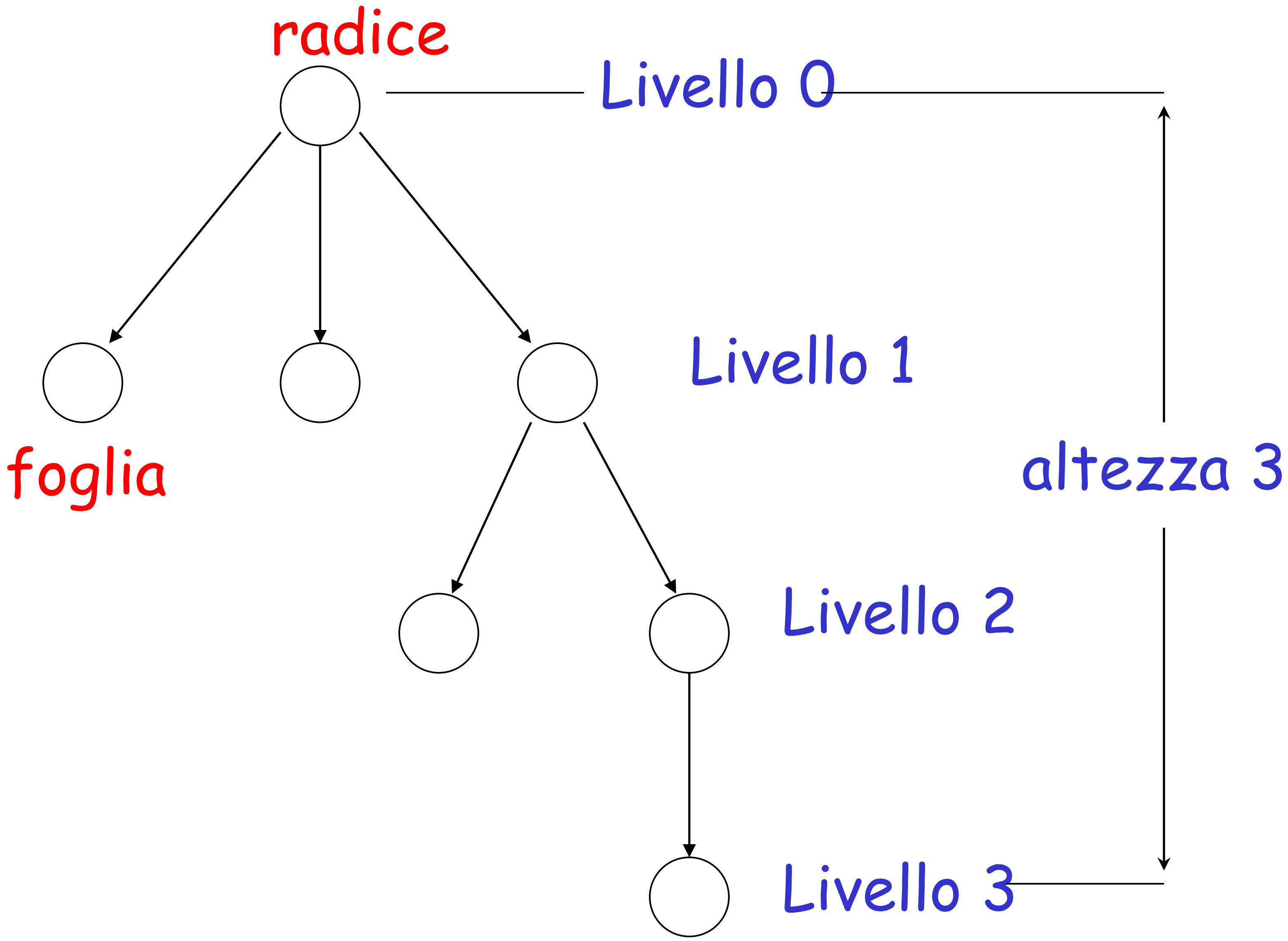
(c, e), (e, b)

(c, e), (e, d)

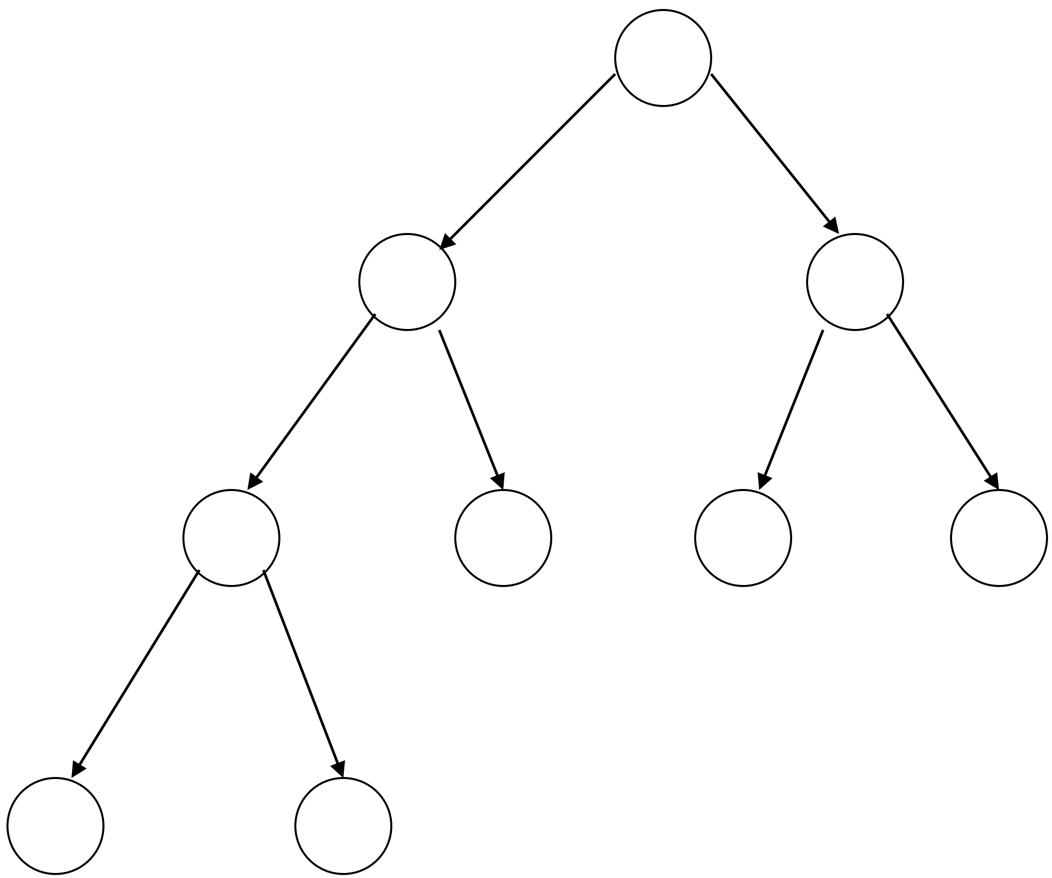
Alberi



Alberi non hanno cicli


radice
Livello 0
Livello 1
foglia
altezza 3
Livello 2
Livello 3

Alberi binari



Tecniche di dimostrazione

- dimostrazione per induzione

- dimostrazione per assurdo

Induzione

Abbiamo una serie di affermazioni ordinate

$$P_1, P_2, P_3, \ldots$$

Se sappiamo

- per qualche b that $P_1, P_2, \ldots, P_b$ sono vere
- per ogni k >= b che

$$P_1, P_2, \ldots, P_k \text{ implica } P_{k+1}$$

Then

allora P_i è vera

Dimostrazione per induzione

- Base induttiva

  trovare $P_1, P_2, \ldots, P_b$ che sono vere

- Ipotesi induttiva

  Asssumiamo che $P_1, P_2, \ldots, P_k$ sono vere,

  Per ogni k >= b

- Passo induttivo

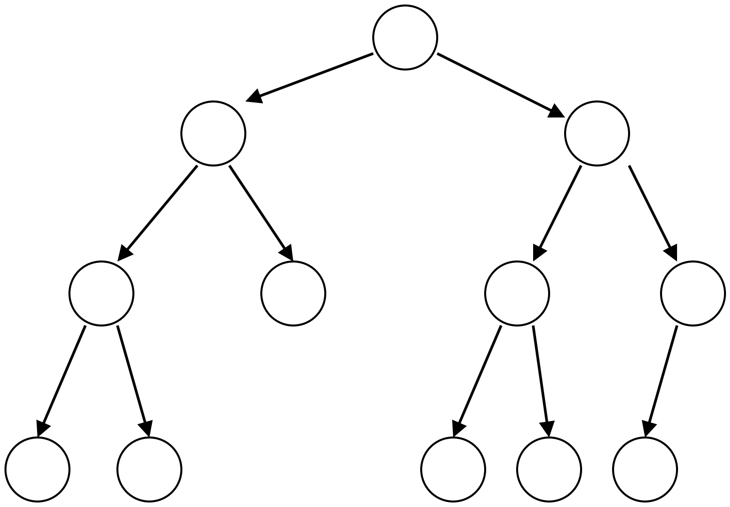
  Dimostrare che P_{k+1} è vera

Esempio

Theorem: Un albero binario di altezza n ha al massimo 2^n foglie.

Proof by induction:

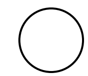
Sia L(i) il massimo numero di foglie di ogni sottoalbero di altezza i



Vogliamo dimostrare che : $L(i) <= 2^i$

- Base induttiva

- $L(0) = 1$ (nodo radice)



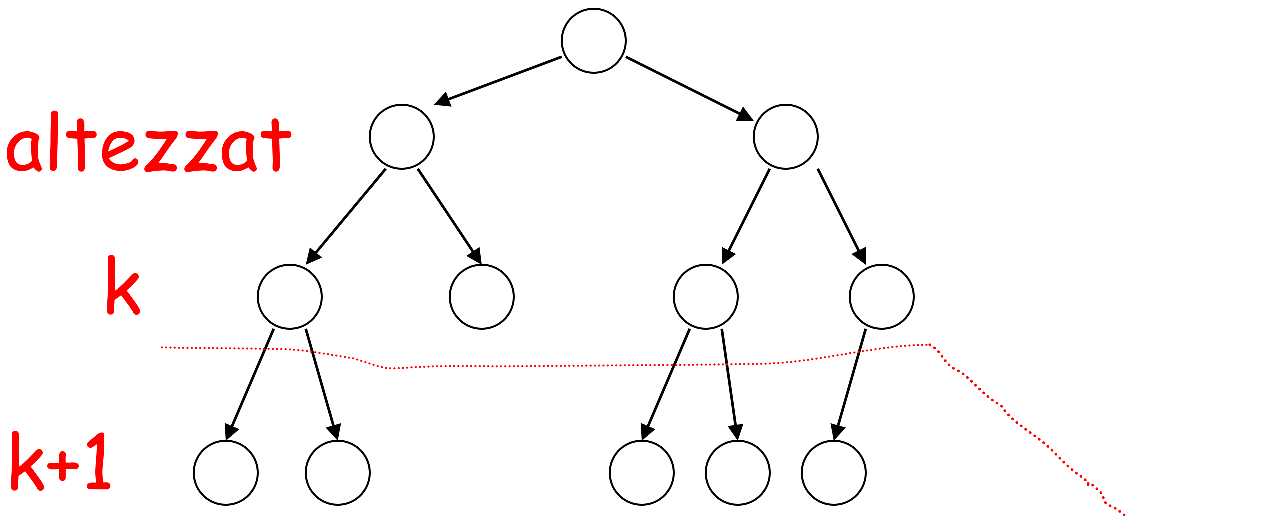
- Ipotesi induttiva

- Assumiamo che $L(i) <= 2^i$ for all $i = 0, 1, \ldots, k$

- Step induttivo

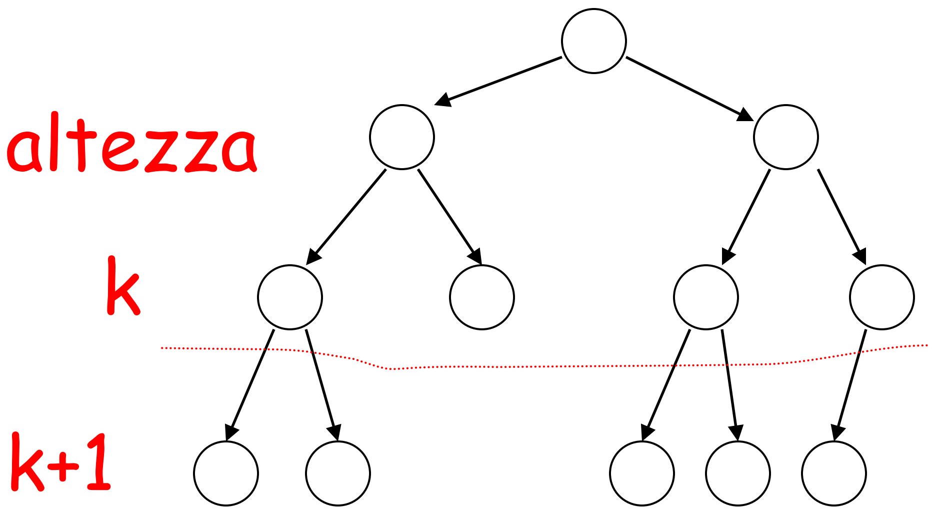
- Dobbiamo dimostrare che $L(k + 1) <= 2^{k+1}$

Step induttivo



Per ipotesi induttiva: $L(k) <= 2^k$

Step induttivo



$L(k) <= 2^k$

$$L(k+1) <= 2 * L(k) <= 2 * 2^k = 2^{k+1}$$

(possiamo addizionare al massimo due nodi per ogni Foglia di livello k)

Remark

La ricorsione è un altra cosa

Esempio di funzione ricorsiva:

$$f(n) = f(n-1) + f(n-2)$$

$$f(0) = 1, \quad f(1) = 1$$

Dimostrazione per assurdo

Vogliamo provare che P è vero

- Assumiamo che P è falso
- arriviamo ad una conclusione sbagliata
- quindi, P deve essere vero.

Esempio

Teorema: $\sqrt{2}$ non è razionale

Dimostrazione:

Assumiamo per assurdo che sia razionale

$\sqrt{2}$ = n/m

n e m non devono avere fattori comuni

Proviamo che questa affermazione è impossibile

$\sqrt{2} = n/m \Rightarrow 2\, m^2 = n^2$

quindi, n^2 è pari
quindi n è pari
(quadrato di dispari è dispari) $\Rightarrow$ n è pari

$n = 2\, k$

$2\, m^2 = 4k^2 \Rightarrow m^2 = 2k^2 \Rightarrow$ m è pari

$m = 2\, p$

Allora, m e n hanno come fattore comune 2

Contradizione!

Linguaggi

Linguaggio: un insieme di stringhe

Stringa: una sequenza di simboli da un alfabeto

Esempio:

Stringhe: gatto, cane, casa

Linguaggio: {gatto, cane, casa}

Alfabeto: $\Sigma = \{a, b, c, \ldots, z\}$

Linguaggi sono usati per descrivere problemi di calcolo:

$$PRIMI = \{2,3,5,7,11,13,17,\ldots\}$$

$$Pari = \{0,2,4,6,\ldots\}$$

Alfabeto: $\Sigma = \{0,1,2,\ldots,9\}$

Alfabeti e Stringe

Un alfabeto è un insieme di simboli

Esempio Alfabeto: $\Sigma = \{a, b\}$

Una stringa è una sequenza di simboli da un alfabeto

Esempio Stringhe

a

ab

$abba$

$aaabbbaabab$

$u = ab$

$v = bbbaaa$

$w = abba$

Alfabeto dei numeri decimali

$$\Sigma = \{0,1,2,\ldots,9\}$$

102345 567463386

Alfabeto dei numeri binari $\Sigma = \{0,1\}$

100010001 101101111

Alfabeto dei numeri unari $\Sigma = \{1\}$

Numeri unari :	11	111	1111	11111
Numeri decimali:	1	2	3	4

Zero?

Operazioni su stringhe

$w = a_1 a_2 \cdots a_n$ $\qquad$ $abba$

$v = b_1 b_2 \cdots b_m$ $\qquad$ $bbbaaa$

Concatenazione

$wv = a_1 a_2 \cdots a_n b_1 b_2 \cdots b_m$ $\qquad$ $abbabbbaaa$

$$w = a_1 a_2 \cdots a_n$$

$$ababaaabbb$$

Inverso

$$w^R = a_n \cdots a_2 a_1$$

$$bbbaaababa$$

Lunghezza di una stringa

$$w = a_1 a_2 \cdots a_n$$

Lunghezza: $$|w| = n$$

Esempi: $$|abba| = 4$$

$$|aa| = 2$$

$$|a| = 1$$

Lunghezza della concatenazine

$$|uv| = |u| + |v|$$

Esempio: $u = aab, \quad |u| = 3$

$$v = abaab, \quad |v| = 5$$

$$|uv| = |aababaab| = 8$$

$$|uv| = |u| + |v| = 3 + 5 = 8$$

Stringa vuota

Una stringa con nessuna lettera è denotata:

$$\lambda \text{ o } \varepsilon$$

Osservaziones:

$$|\lambda| = 0$$

$$\lambda w = w\lambda = w$$

$$\lambda abba = abba\lambda = ab\lambda ba = abba$$

Sottostringa

Sottostringa di una stringa:

Una sequenza consecutiva di caratteri:

Stringa	Sottostringa
$\underline{ab}bab$	ab
$\underline{abba}b$	$abba$
$ab\underline{b}ab$	b
$a\underline{bbab}$	$bbab$

Prefisso e Suffisso

$$abbab$$

Prefisso	Suffisso
λ	$abbab$
a	$bbab$
ab	bab
abb	ab
$abba$	b
$abbab$	λ

$$w = \underbrace{u}_{\text{Prefisso}}\underbrace{v}_{\text{Suffisso}}$$

Altre operazioni

$$w^n = \underbrace{ww \cdots w}_{n}$$

Esempio: $(abba)^2 = abbaabba$

Definizione: $w^0 = \lambda$

$$(abba)^0 = \lambda$$

L'operazione *

Σ^* : L'insieme di tutte le possibili stringhe che è possibile generare a partire dall'alfabeto Σ

$$\Sigma = \{a, b\}$$

$$\Sigma^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$$

L'operazione +

Σ^+ : L'insieme di tutte le possibili stringe che è possibile generare a partire dall'alfabeto Σ eccetto λ

$$\Sigma = \{a, b\}$$

$$\Sigma^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$$

$$\Sigma^+ = \Sigma^* - \lambda$$

$$\Sigma^+ = \{a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$$

Linguaggi. estensionali

Un linguaggio su un alfabeto Σ

È un qualsiasi sottoinsieme di Σ^*

Esempio:

$$\Sigma = \{a, b\}$$

$$\Sigma^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, \ldots\}$$

linguaggio : $\{\lambda\}$

linguaggio : $\{a, aa, aab\}$

linguaggio : $\{\lambda, abba, baba, aa, ab, aaaaaa\}$

Esempi di linguaggi

Alfabeto $\Sigma = \{a, b\}$

Un linguaggio infinito $L = \{a^n b^n : n \geq 0\}$

$$\left.\begin{array}{l} \lambda \\ ab \\ aabb \\ aaaaabbbbb \end{array}\right\} \in L \qquad abb \notin L$$

Numeri primi

alfabeto $\Sigma = \{0,1,2,\ldots,9\}$

Linguaggio:

$$PRIMES = \{x : x \in \Sigma^* \text{ and } x \text{ is prime}\}$$

$$PRIMES = \{2,3,5,7,11,13,17,\ldots\}$$

Numeri pari e dispari

alfabeto $\Sigma = \{0,1,2,\ldots,9\}$

$$EVEN = \{x : x \in \Sigma^* \text{ e } x \text{ è pari}\}$$

$$EVEN = \{0,2,4,6,\ldots\}$$

$$ODD = \{x : x \in \Sigma^* \text{ e } x \text{ è dispari}\}$$

$$ODD = \{1,3,5,7,\ldots\}$$

Somma unaria

alfabeto: $\Sigma = \{1, +, =\}$

Linguaggio:

$$ADDITION = \{x + y = z \ : x = 1^n, y = 1^m, z = 1^k, \\ n + m = k\}$$

$$11 + 111 = 11111 \in ADDITION$$

$$111 + 111 = 111 \notin ADDITION$$

Radici

Alfabeto: $\Sigma = \{1, \#\}$

Linguaggio:

$$SQUARES = \{x\#y \ : x = 1^n, y = 1^m, m = n^2\}$$

$$11\#1111 \in SQUARES$$

$$111\#1111 \notin SQUARES$$

Nota che :

Insieme vuoto $\emptyset = \{\ \} \neq \{\lambda\}$

Dimensione insiemi

$$|\{\ \}| = |\emptyset| = 0 \qquad |\{\lambda\}| = 1$$

Lunghezza di una stringa

$$|\lambda| = 0$$

Operazioni sui linguaggi

Le stesse degli insiemi

$$\{a, ab, aaaa\} \bigcup \{bb, ab\} = \{a, ab, bb, aaaa\}$$

$$\{a, ab, aaaa\} \bigcap \{bb, ab\} = \{ab\}$$

$$\{a, ab, aaaa\} - \{bb, ab\} = \{a, aaaa\}$$

Complemento: $\overline{L} = \Sigma^* - L$

$$\overline{\{a, ba\}} = \{\lambda, b, aa, ab, bb, aaa, \ldots\}$$

Inverso

Definizione: $L^R = \{w^R : w \in L\}$

Esempio: $\{ab, aab, baba\}^R = \{ba, baa, abab\}$

$$L = \{a^n b^n : n \geq 0\}$$

$$L^R = \{b^n a^n : n \geq 0\}$$

Concatenazione

Definizione: $L_1 L_2 = \{xy : x \in L_1, y \in L_2\}$

Esempio: $\{a, ab, ba\}\{b, aa\}$

$$= \{ab, aaa, abb, abaa, bab, baaa\}$$

Altre operazioni

Definizione: $$L^n = \underbrace{LL\cdots L}_{n}$$

$$\{a,b\}^3 = \{a,b\}\{a,b\}\{a,b\} =$$
$$\{aaa, aab, aba, abb, baa, bab, bba, bbb\}$$

Casi speciale: $L^0 = \{\lambda\}$

$$\{a, bba, aaa\}^0 = \{\lambda\}$$

$$L = \{a^n b^n : n \geq 0\}$$

$$L^2 = \{a^n b^n a^m b^m : n, m \geq 0\}$$

$$aabbaaabbb \in L^2$$

Star-Closure-intensione (Kleene *)

Tutte le stringhe che possono essere costruite da L

Definizione: $$L^* = L^0 \cup L^1 \cup L^2 \cdots$$

Esempio:

$$\{a, bb\}^* = \begin{Bmatrix} \lambda, \\ a, bb, \\ aa, abb, bba, bbbb, \\ aaa, aabb, abba, abbbb, \ldots \end{Bmatrix}$$

Chiusure

Definizione: $L^{+} = L^{1} \cup L^{2} \cup \cdots$

Lo stesso come L^{*} without the λ

$$\{a, bb\}^{+} = \left\{ \begin{array}{l} a, bb, \\ aa, abb, bba, bbbb, \\ aaa, aabb, abba, abbbb, \ldots \end{array} \right\}$$

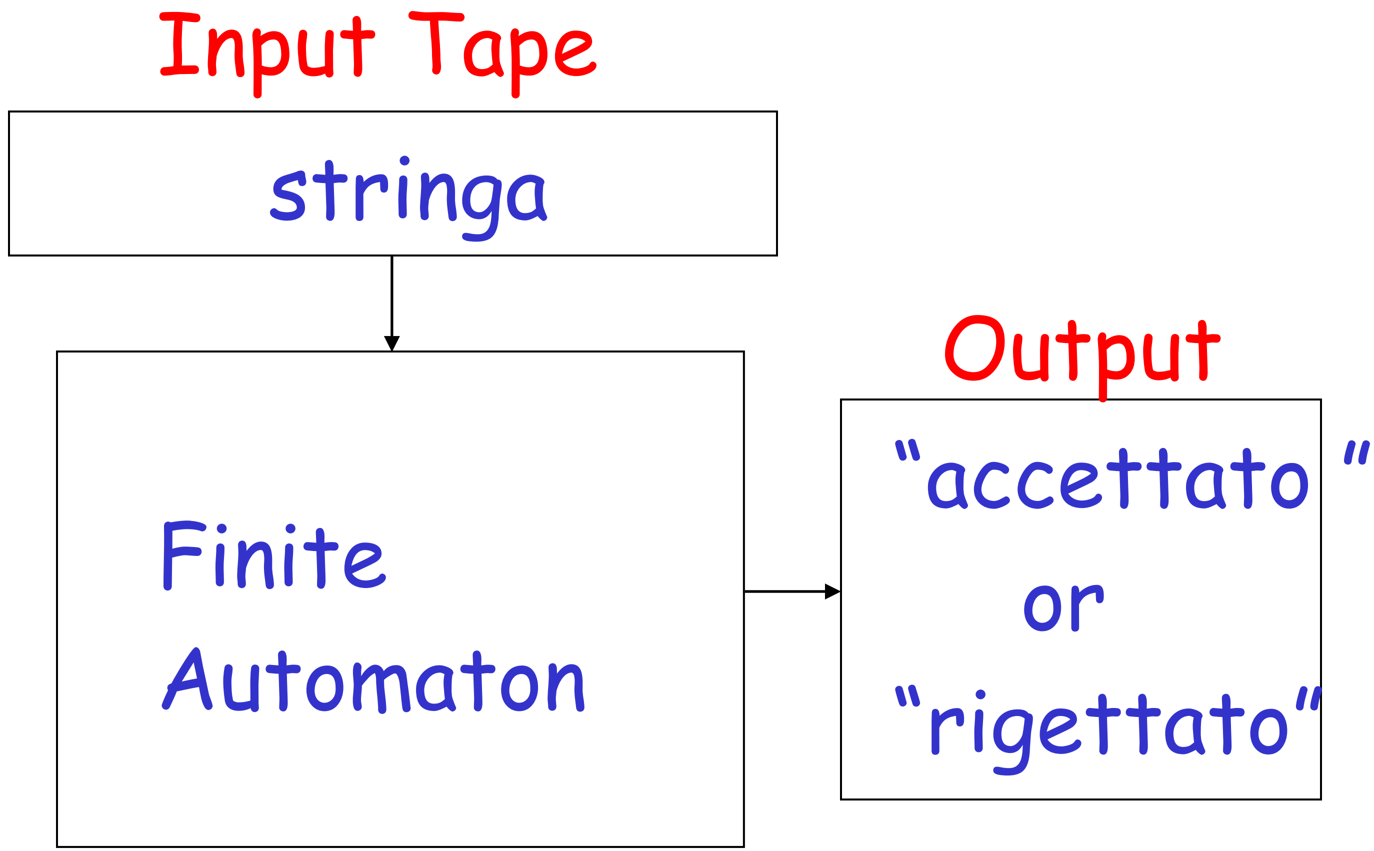
Deterministic

Finite Automata

E linguaggi regolari

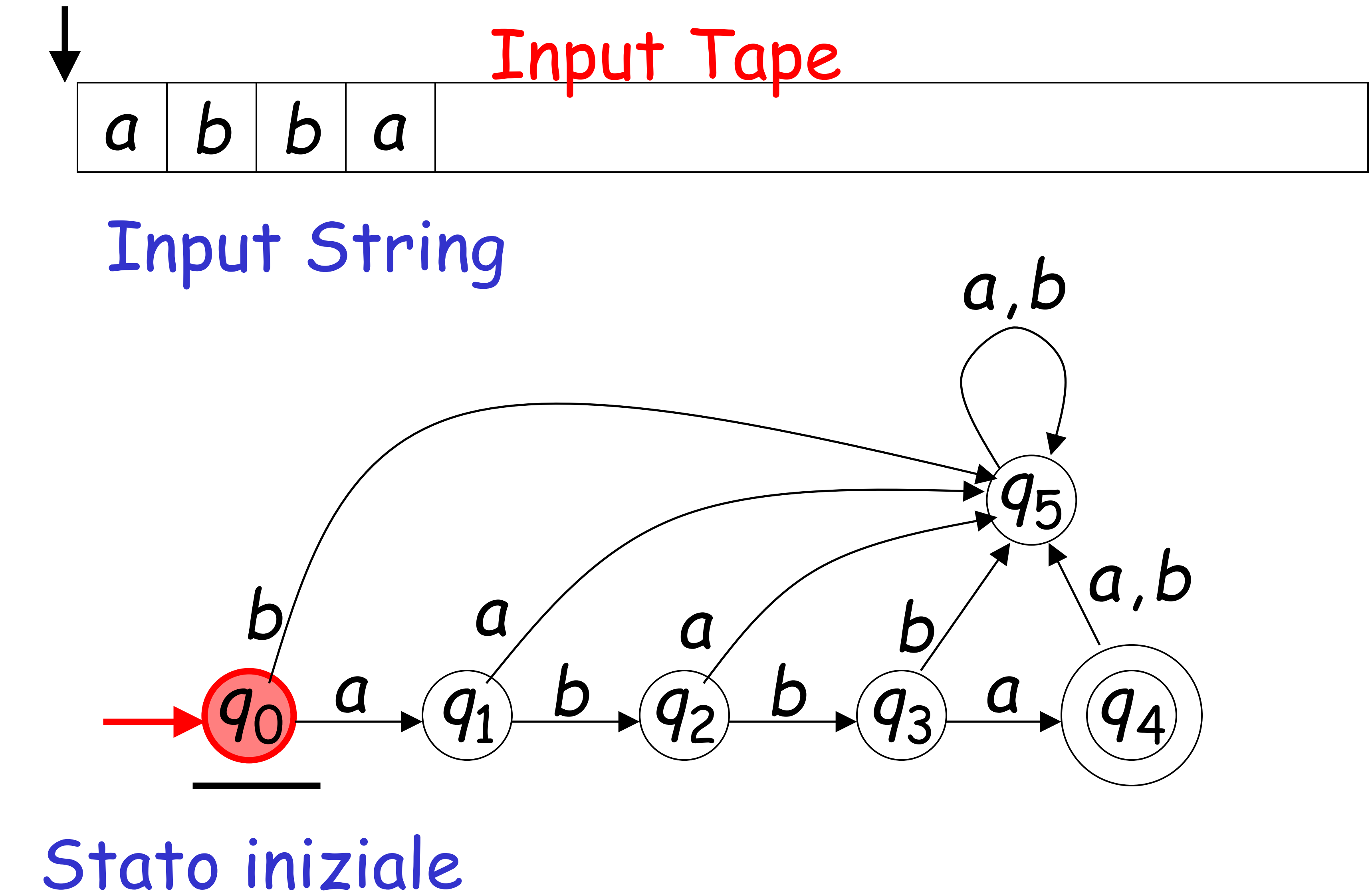
Simulatore http://www.jflap.org/

Deterministic Finite Automaton (DFA)

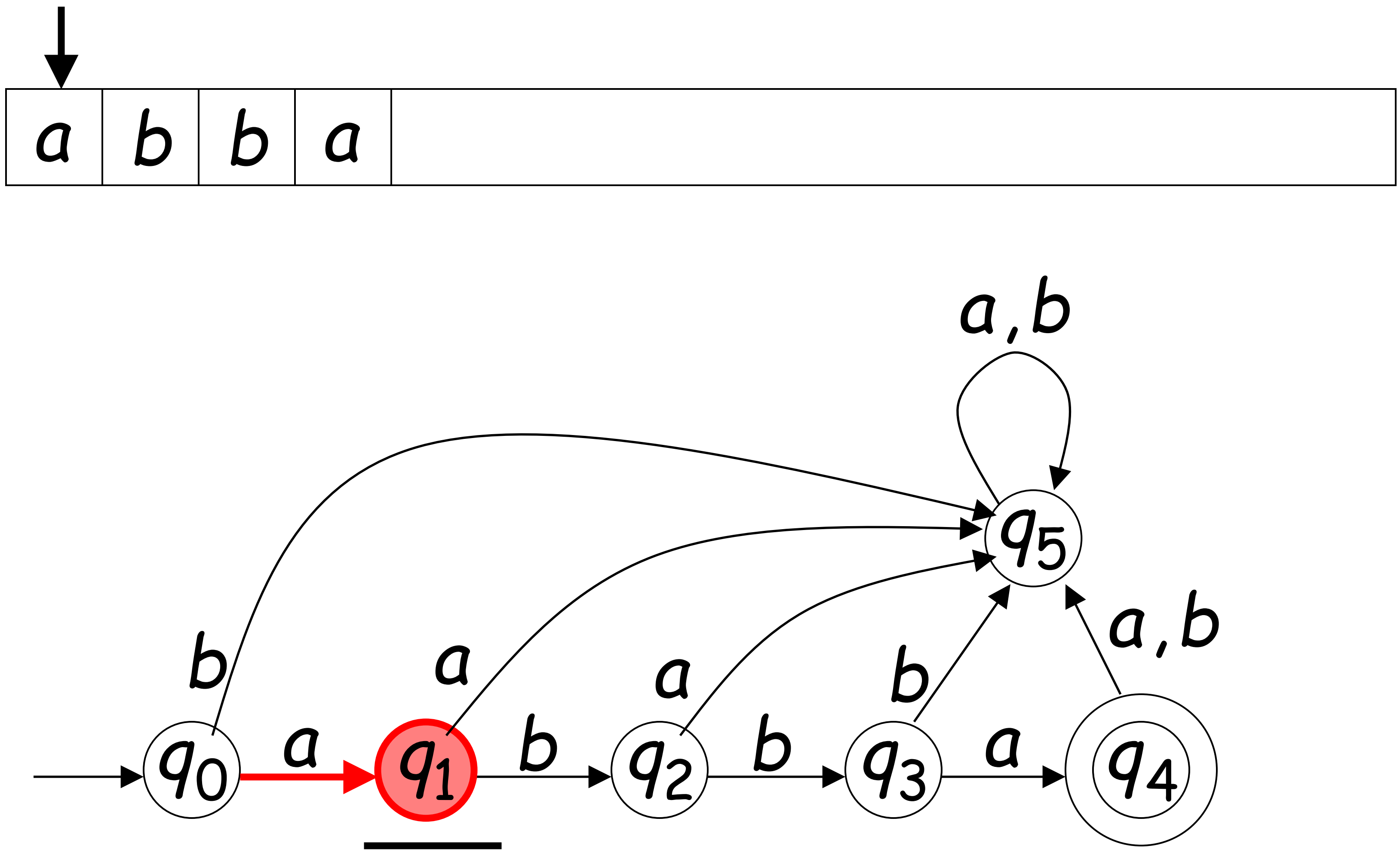


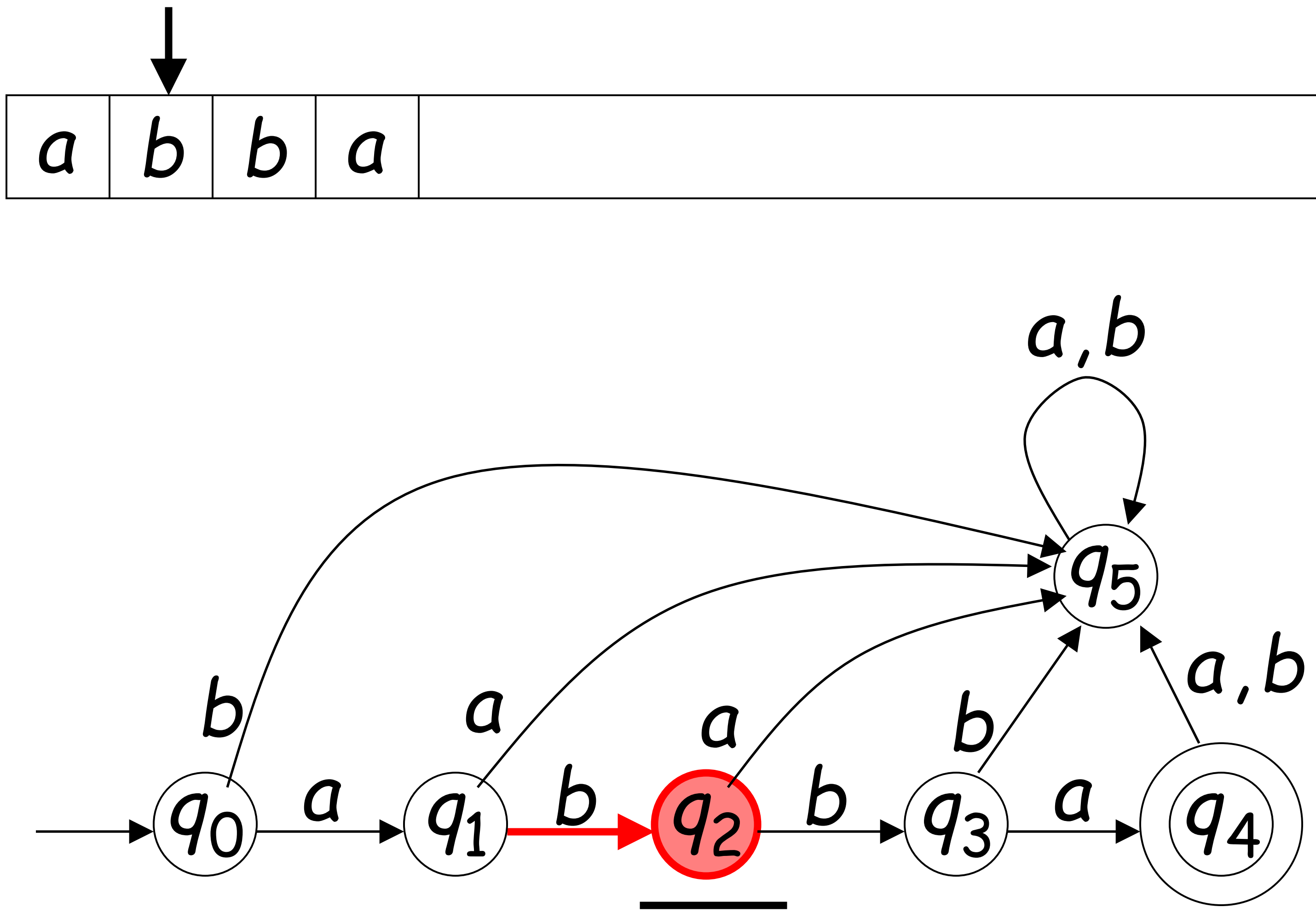
Configurazione iniziale

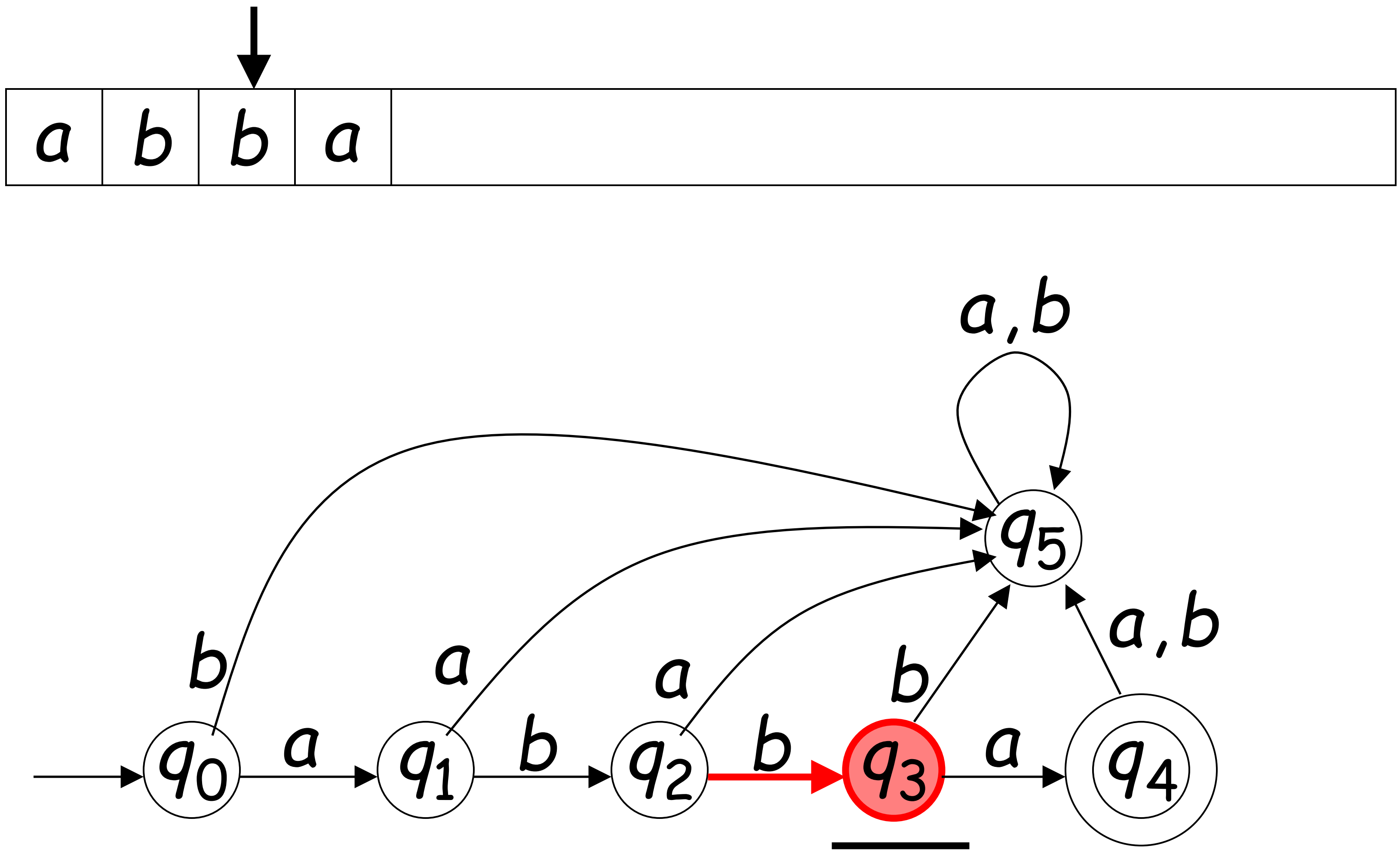
testa



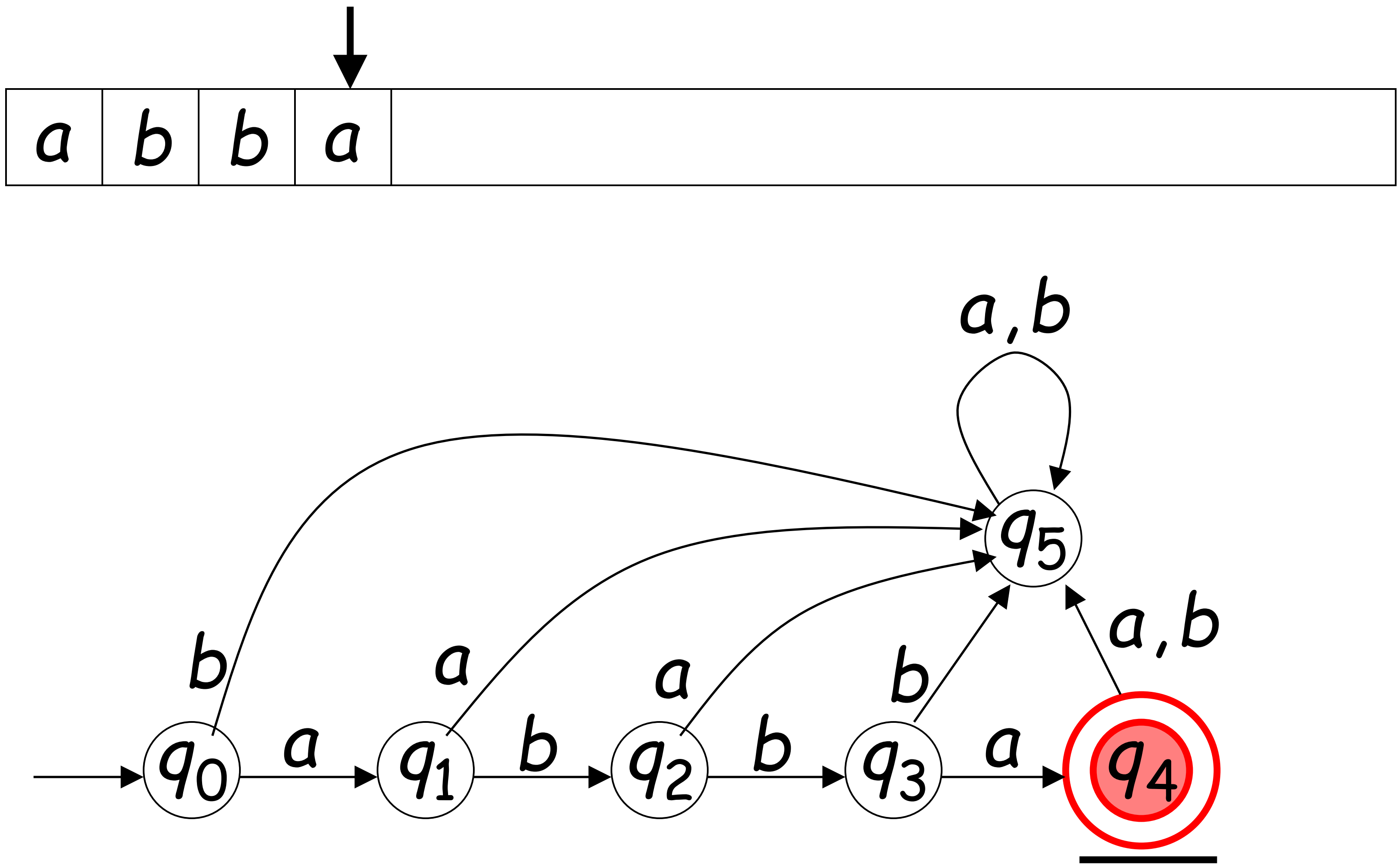
Analizzare l'Input




a b b a
a,b
q5
b
a
a
b
a,b
q0
a
q1
b
q2
b
q3
a
q4

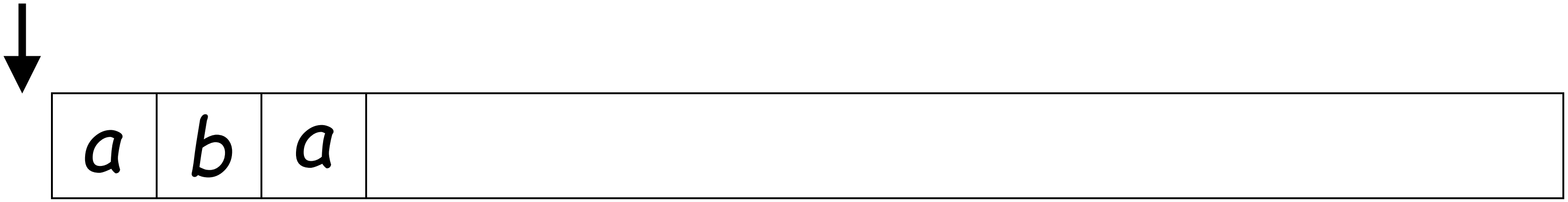

a
b
b
a
a,b
q5
b
a
a
b
a,b
q0
a
q1
b
q2
b
q3
a
q4

Input finito

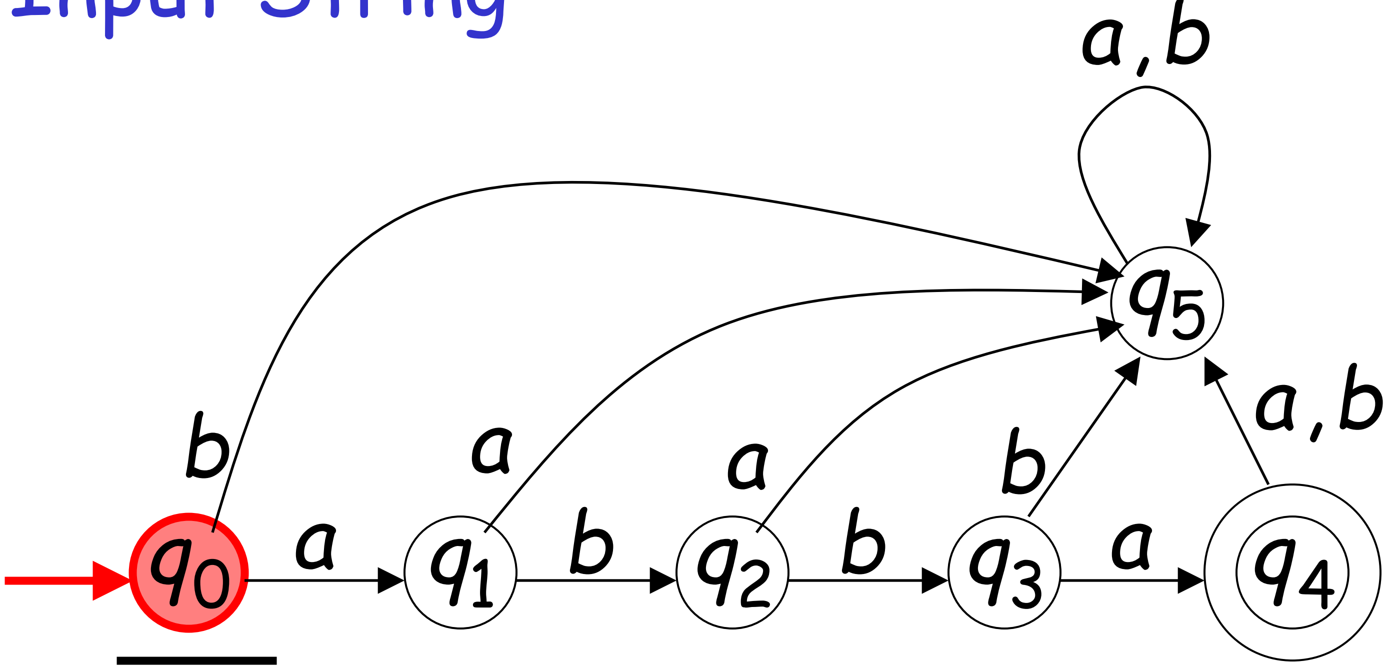


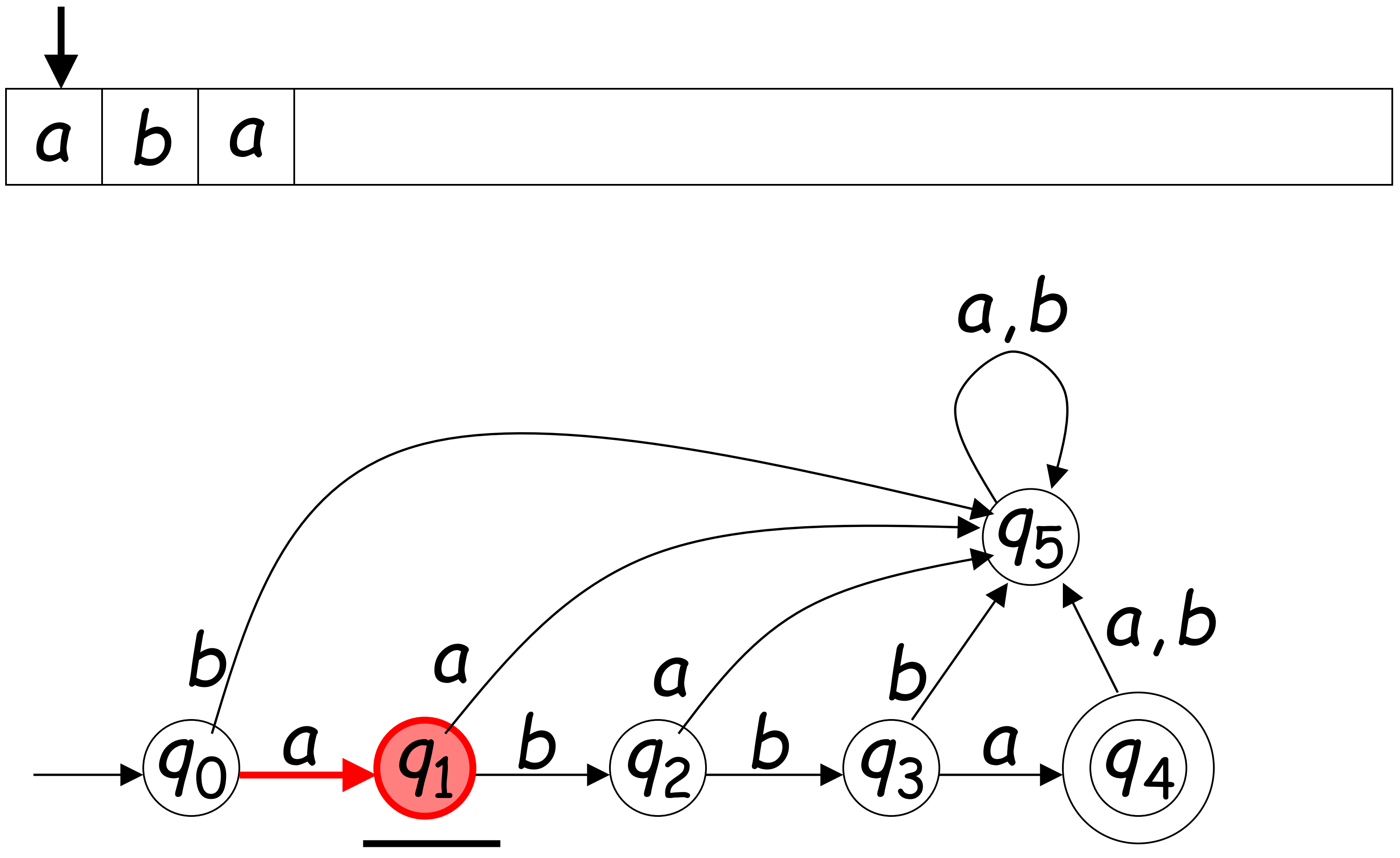
accettato

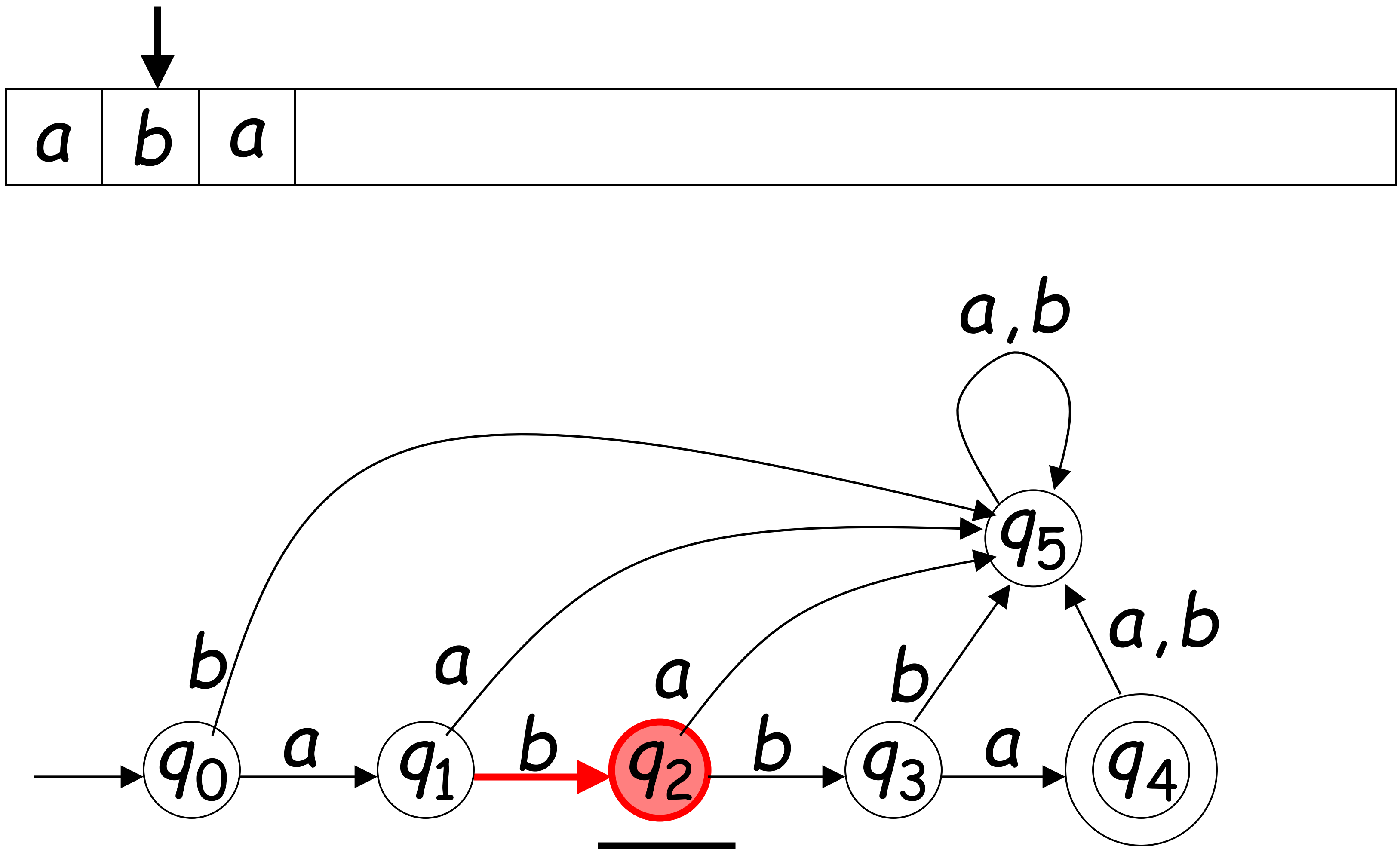
Un caso rigettato



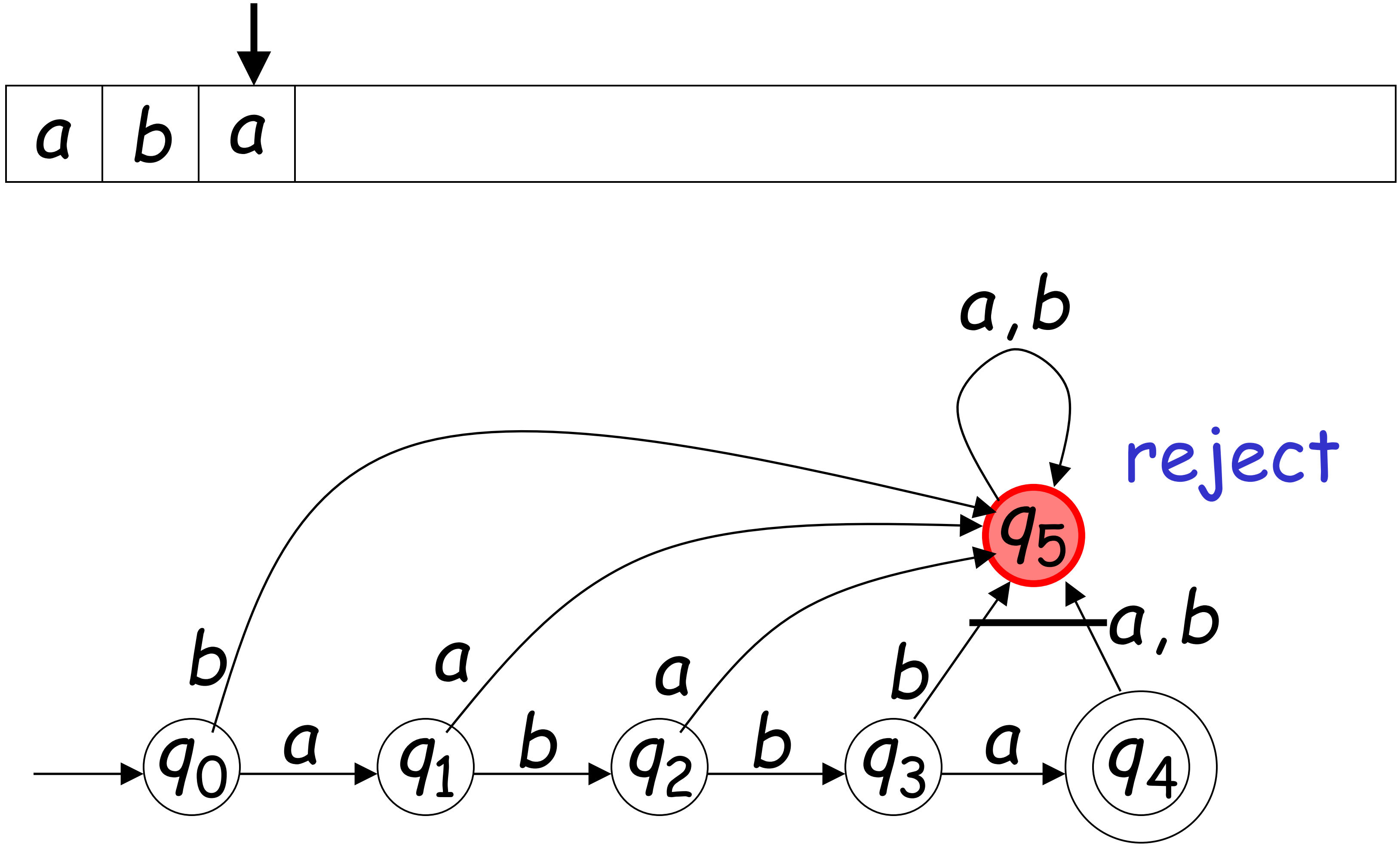
Input String




a b a
a,b
q5
a,b
b
a
a
b
q0
a
q1
b
q2
b
q3
a
q4


a
b
a
a,b
b
a
a
b
a,b
q5
b
a
b
b
a
q0
q1
q2
q3
q4

Input finito

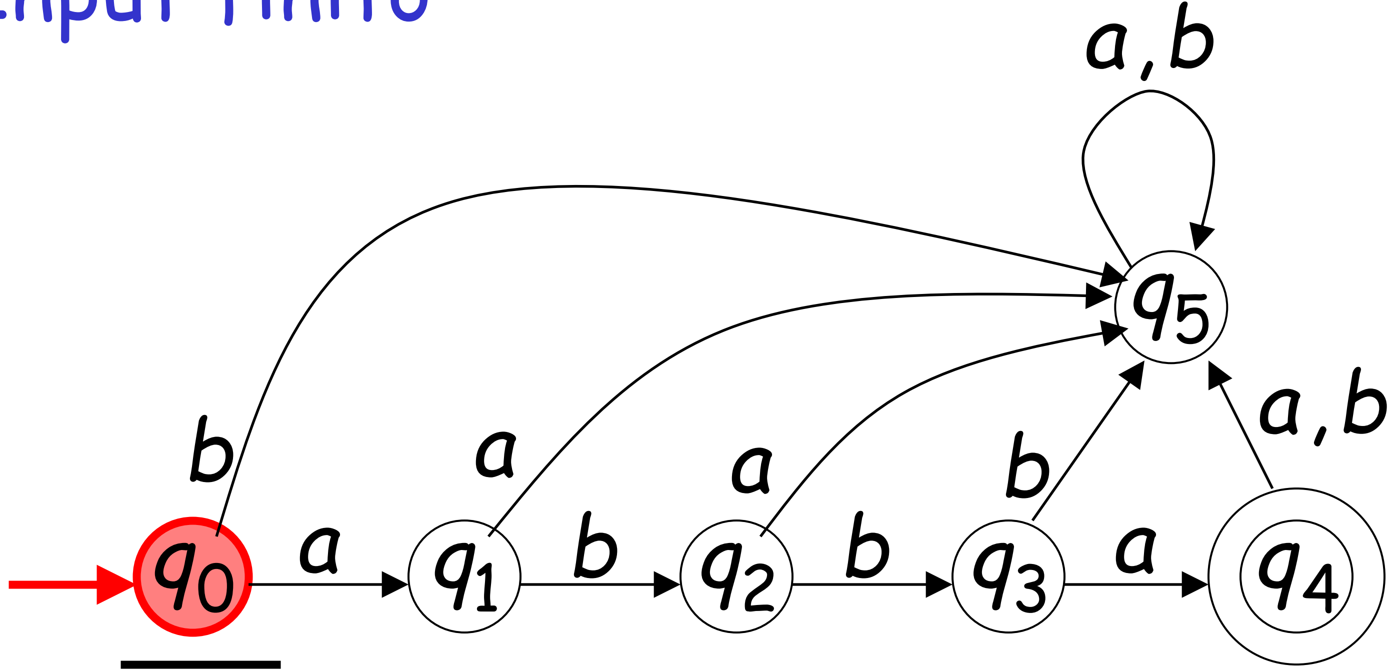


Un altro caso rigettato

Nastro vuoto

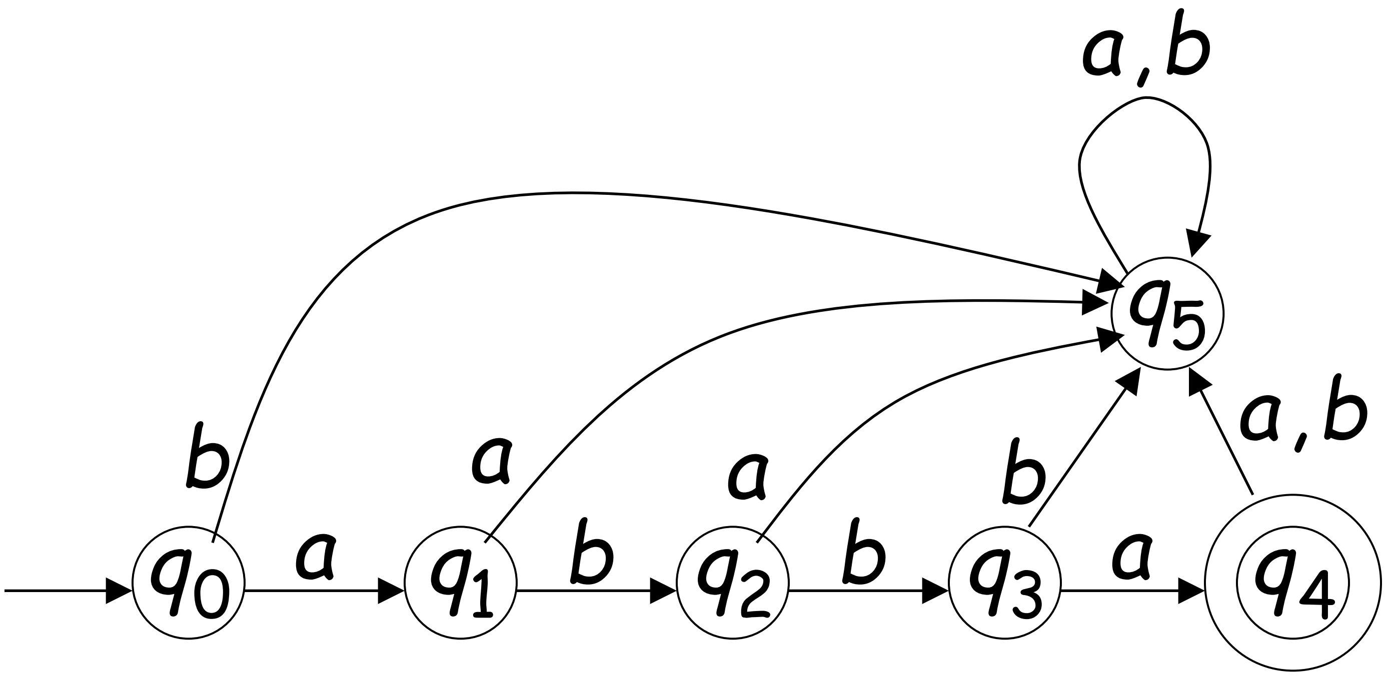
(λ)

Input finito



rigettato

Linguaggio accettato: $L = \{abba\}$



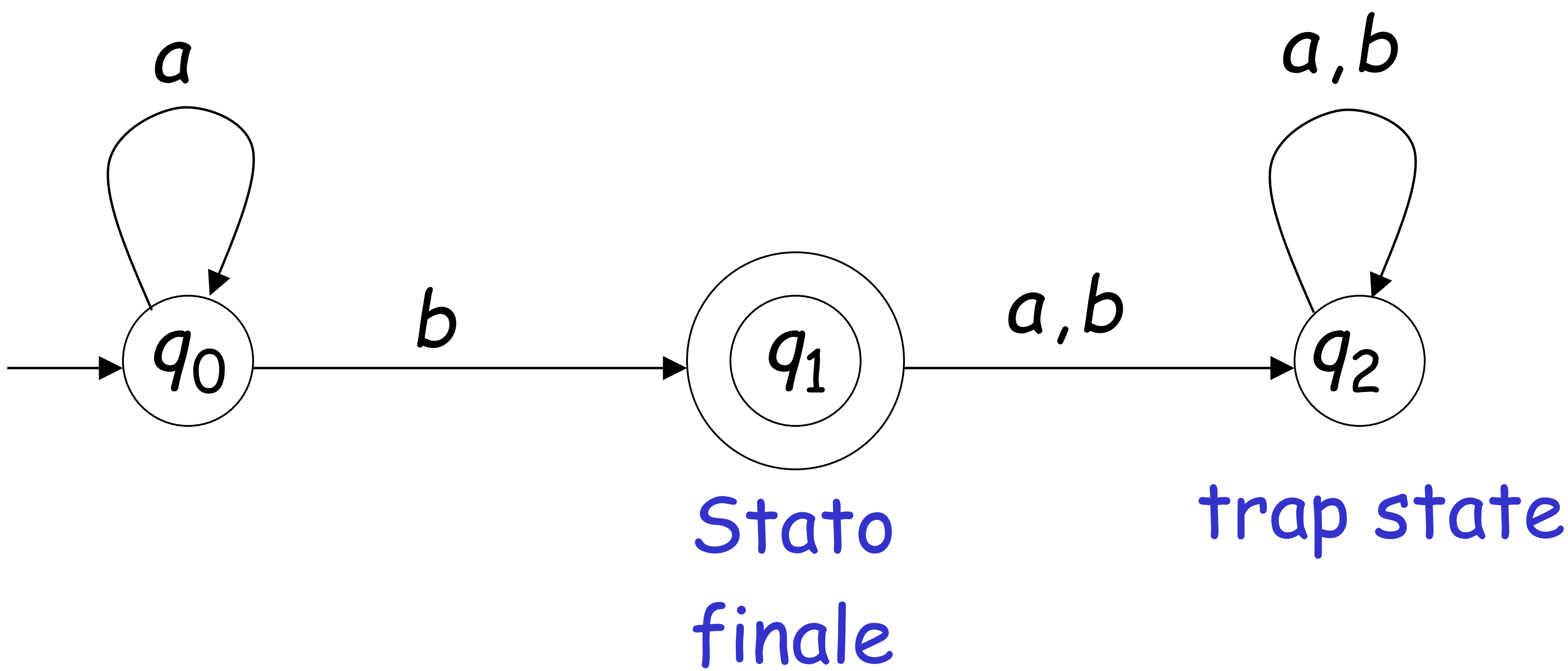
Per accettare una stringa :

Devono essere esaminati tutti i caratteri di Input e l'ultimo stato è uno stato finale

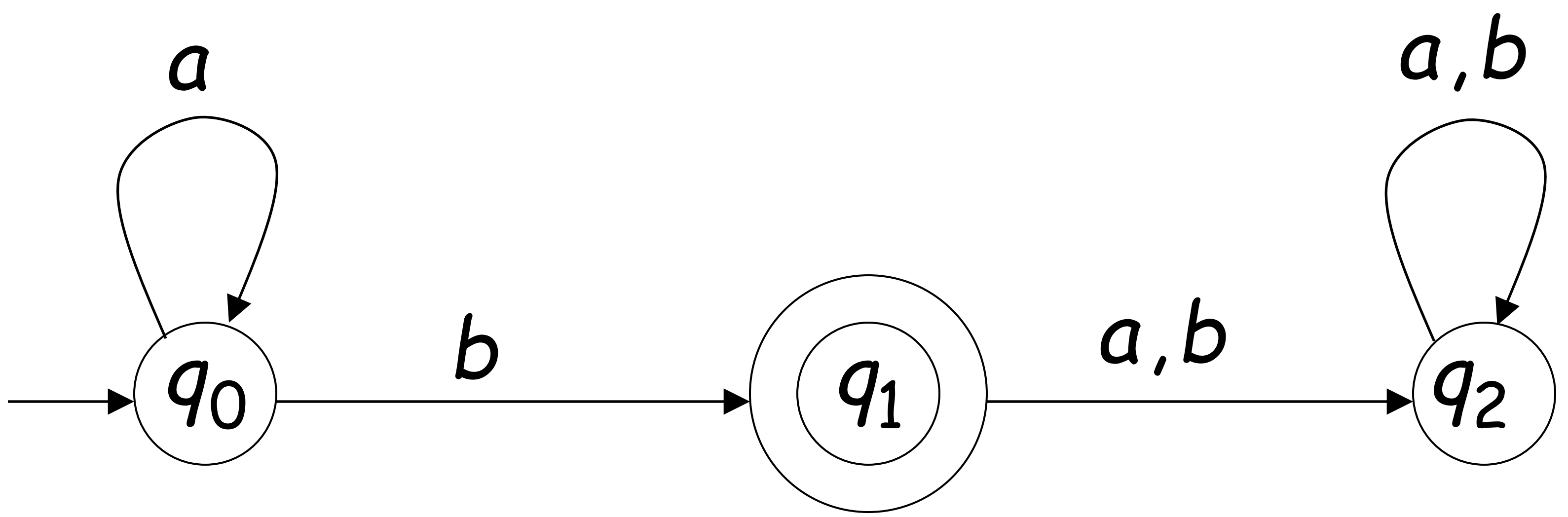
Per rigettare una stringa :

Tutti i caratteri di input sono stati esaminati E non si è raggiunto uno stato finale

Un altro esempio

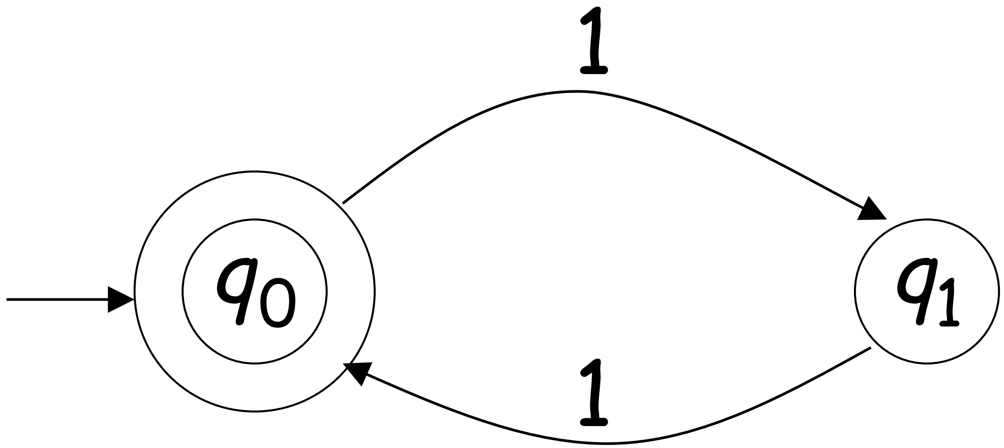


Language Accepted: $L = \{a^n b : n \geq 0\}$



Un altro esempio

Alfabeto: $\Sigma = \{1\}$



Linguaggio accettato:

$$EVEN = \{x : x \in \Sigma^* \text{ and } x \text{ is even}\}$$
$$= \{\lambda, 11, 1111, 111111, \ldots\}$$

Definizione formale

un automa deterministico formale(DFA)

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

Q : insieme degli stati

Σ : alfabeto di input $\lambda \notin \Sigma$

δ : funzione di transizione

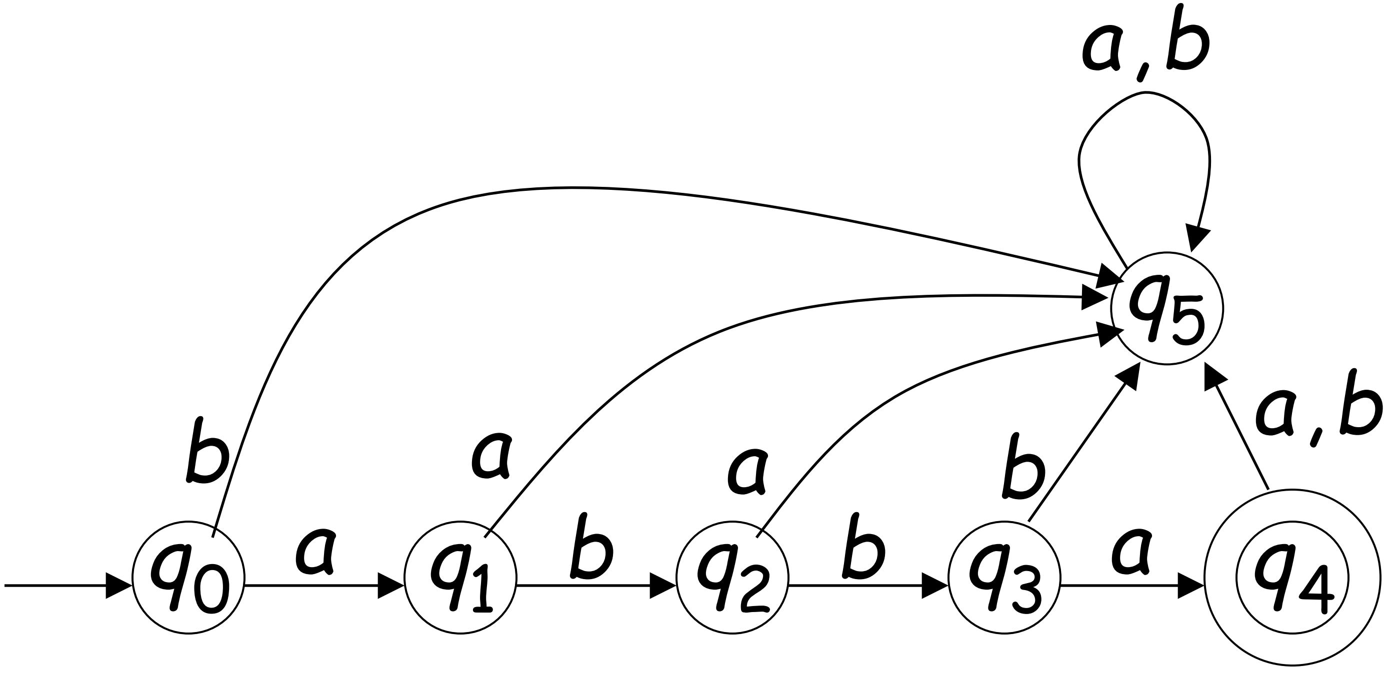
q_0 : stato iniziale

F : insieme degli stati di accettazione (finale)

Insieme degli stati Q

esempio

$$Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$$

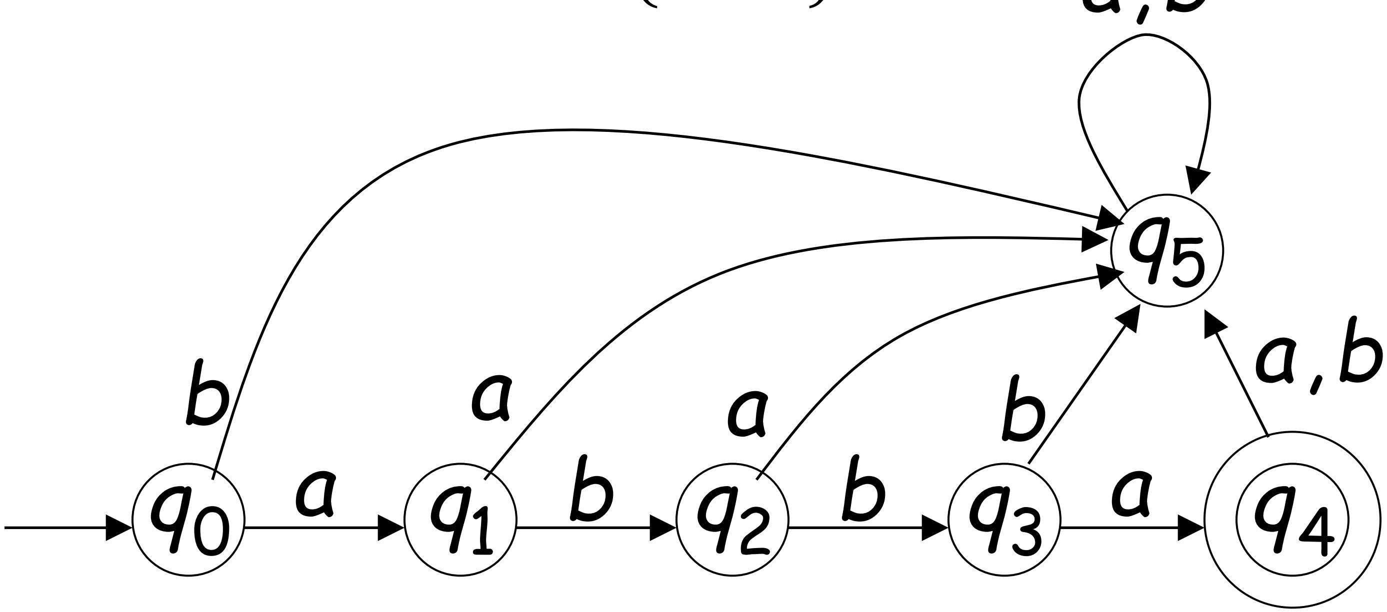


Alfabeto di input Σ

$\lambda \notin \Sigma$:l'alfabeto di input non contene λ

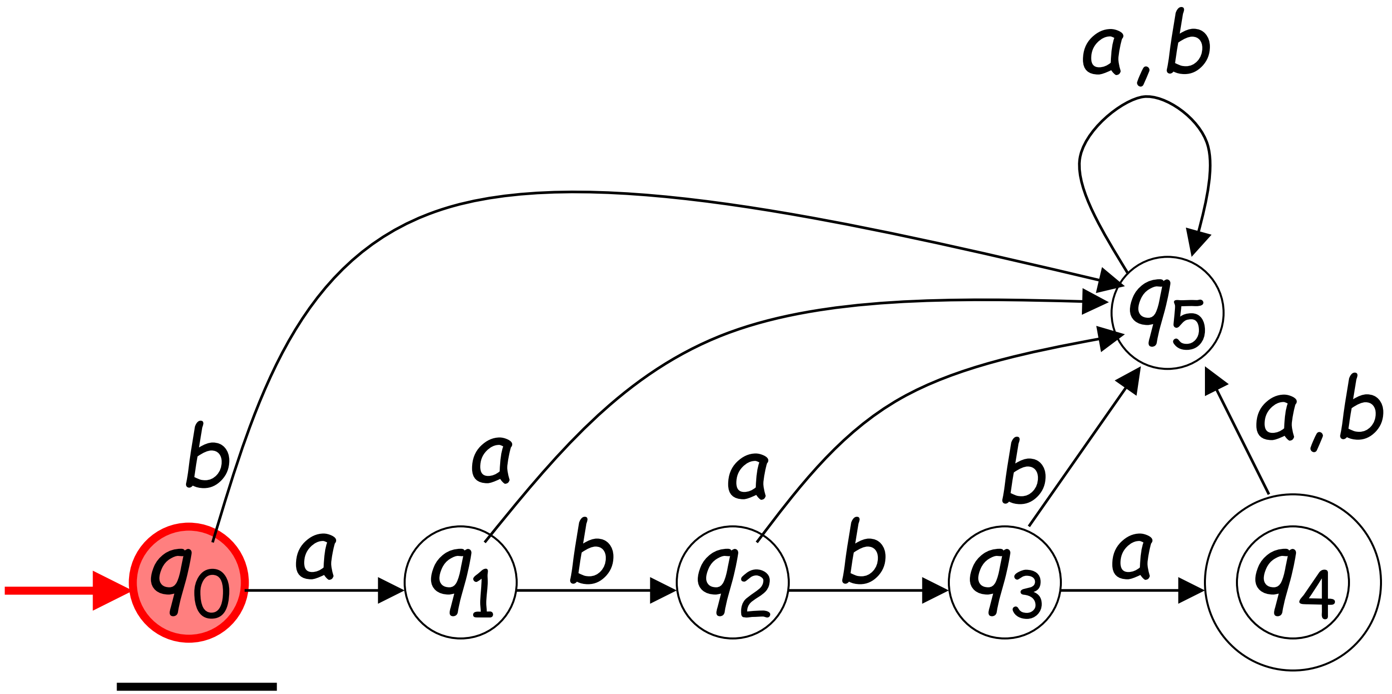
esempio

$$\Sigma = \{a, b\}$$



Stato iniziale q_0

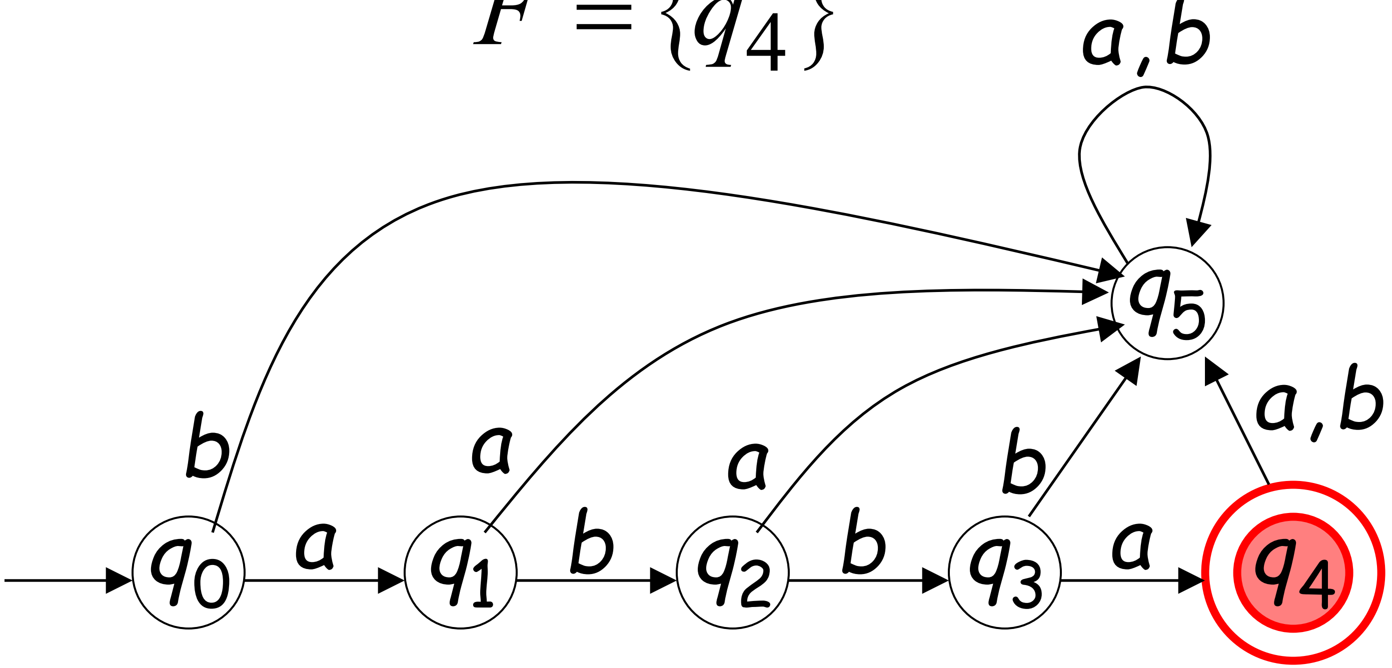
esempio



Insieme stati finali $F \subseteq Q$

esempio

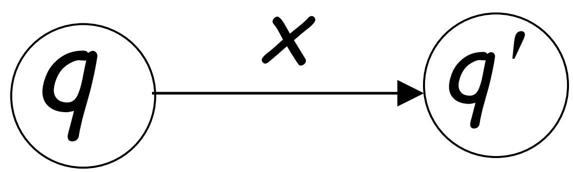
$$F = \{q_4\}$$



Funzione di transizione

$$\delta : Q \times \Sigma \rightarrow Q$$

$$\delta(q, x) = q'$$



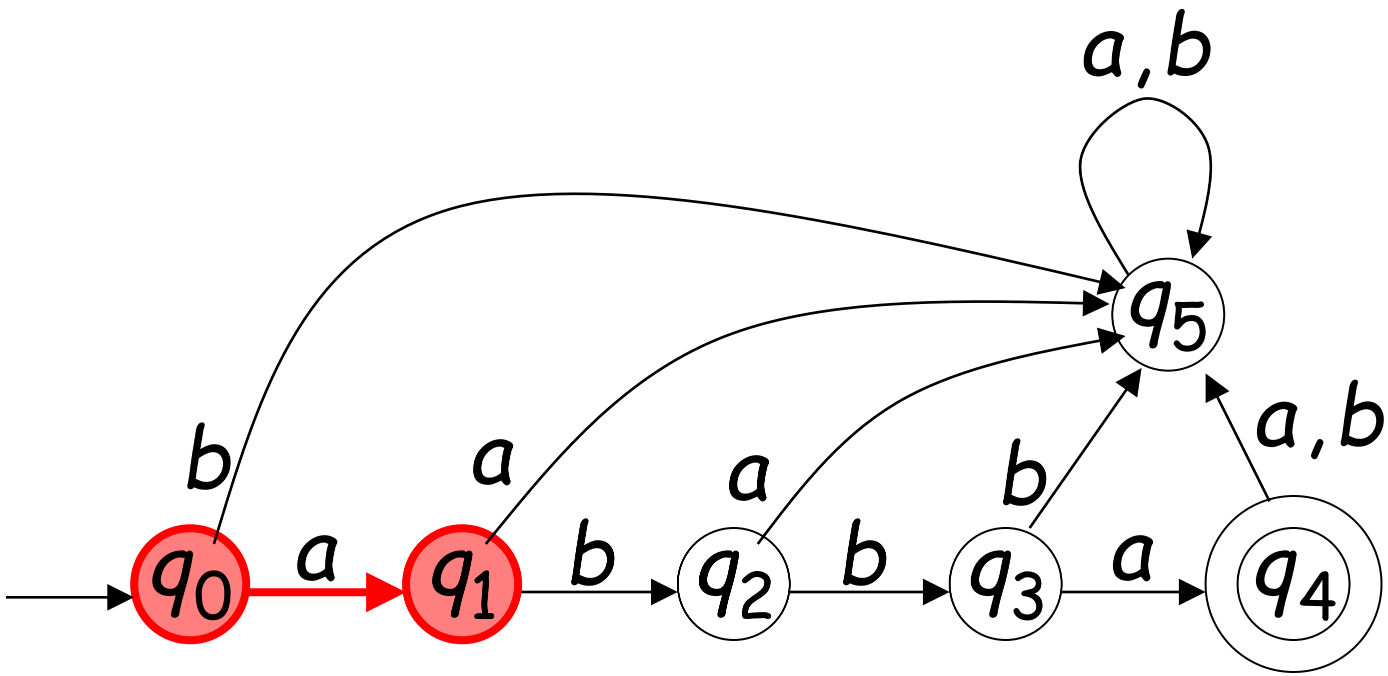
Descrive il risultato della

Transizione dallo stato q

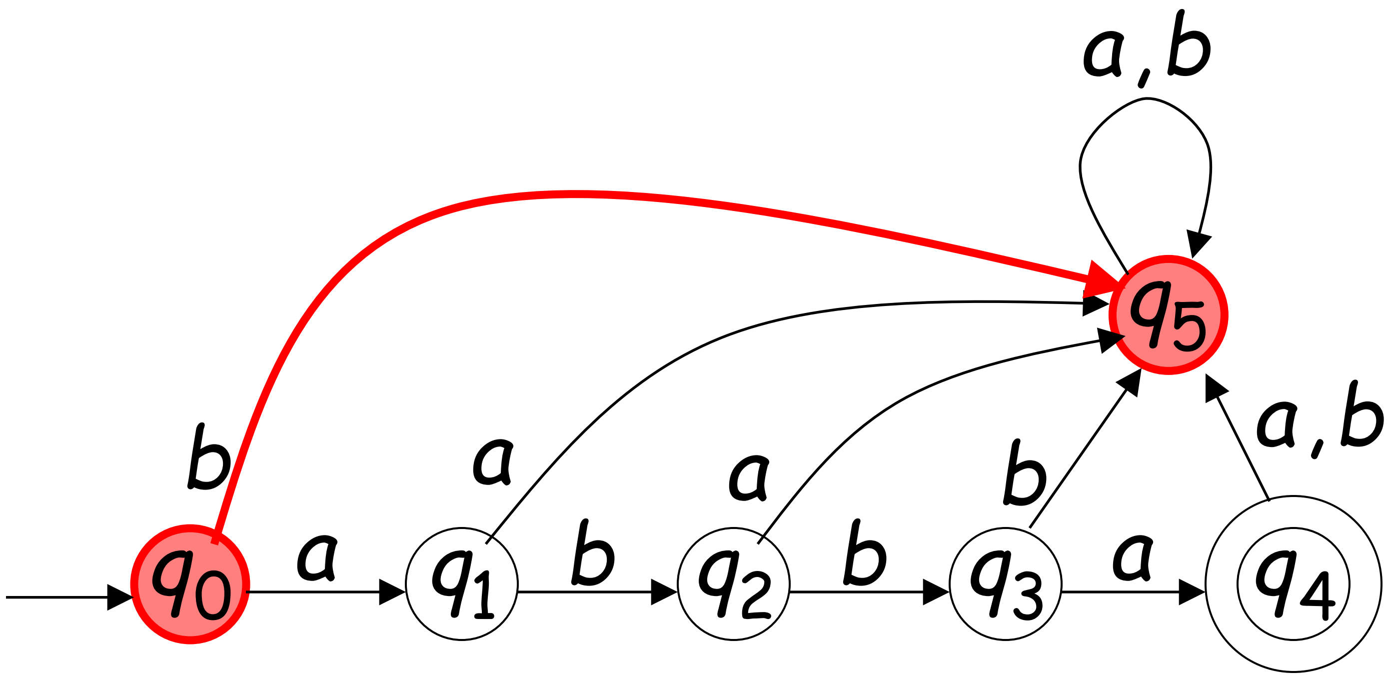
Con simbolo x

esempio:

$$\delta(q_0, a) = q_1$$



$$\delta(q_0, b) = q_5$$



$$\delta(q_2, b) = q_3$$

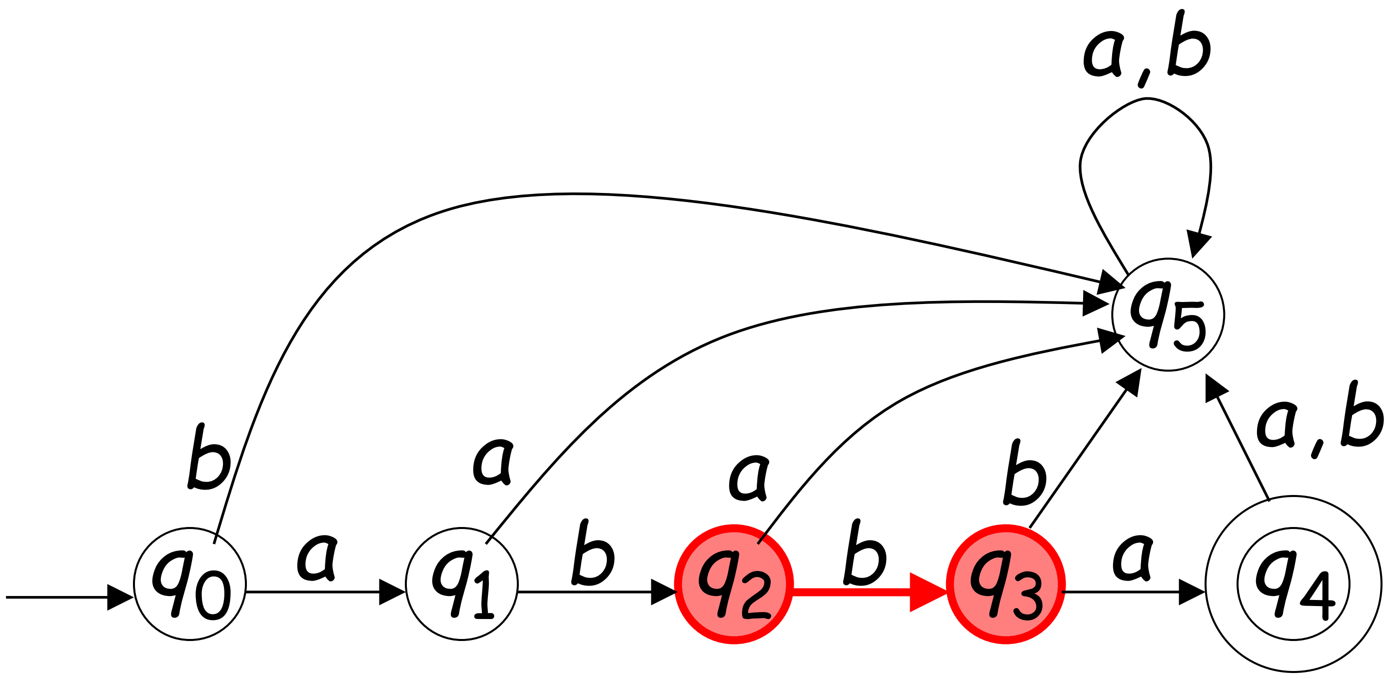
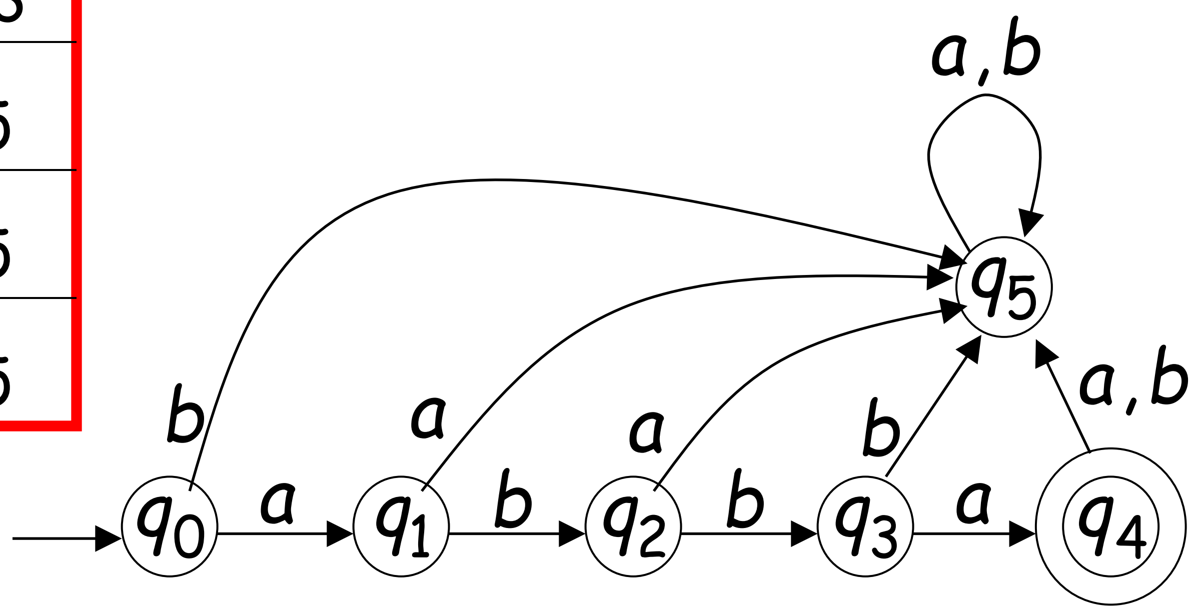


Tavola di transizione per δ

symbols

states

δ	a	b
q_0	q_1	q_5
q_1	q_5	q_2
q_2	q_5	q_3
q_3	q_4	q_5
q_4	q_5	q_5
q_5	q_5	q_5



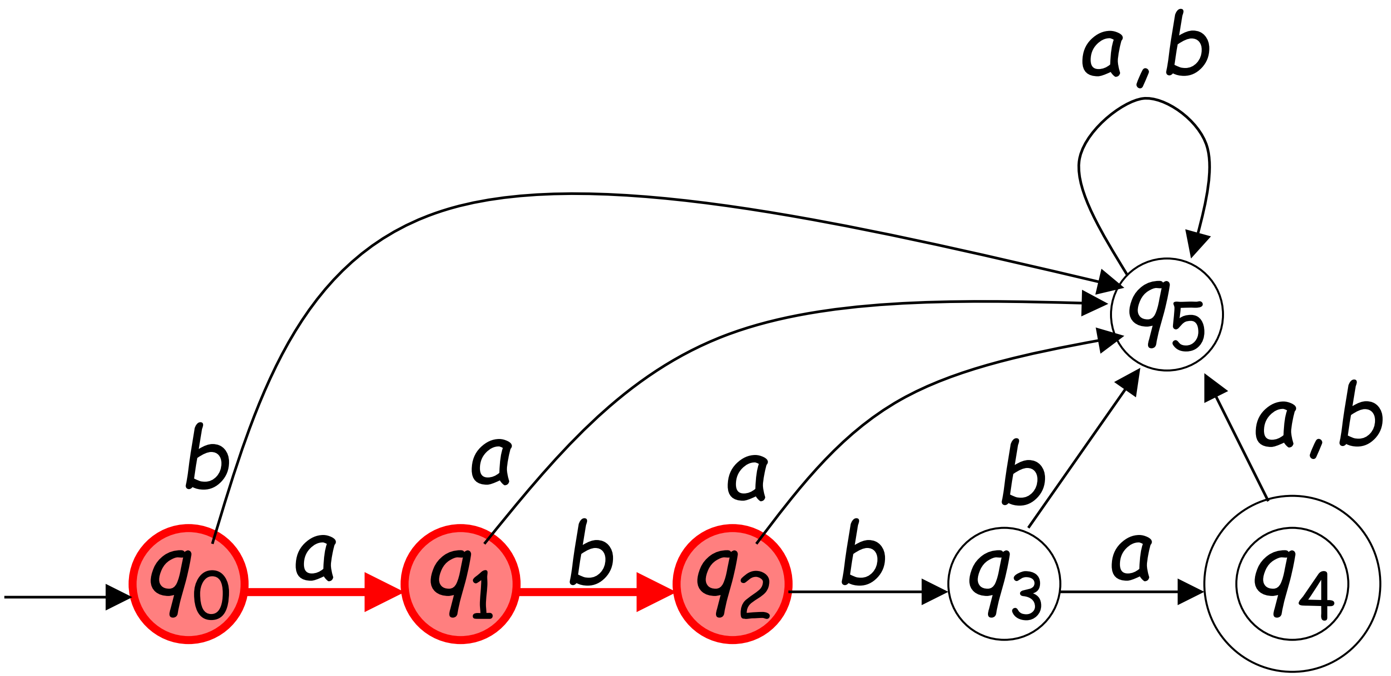
Funzione estesa di transizione

$$\delta^* : Q \times \Sigma^* \to Q$$

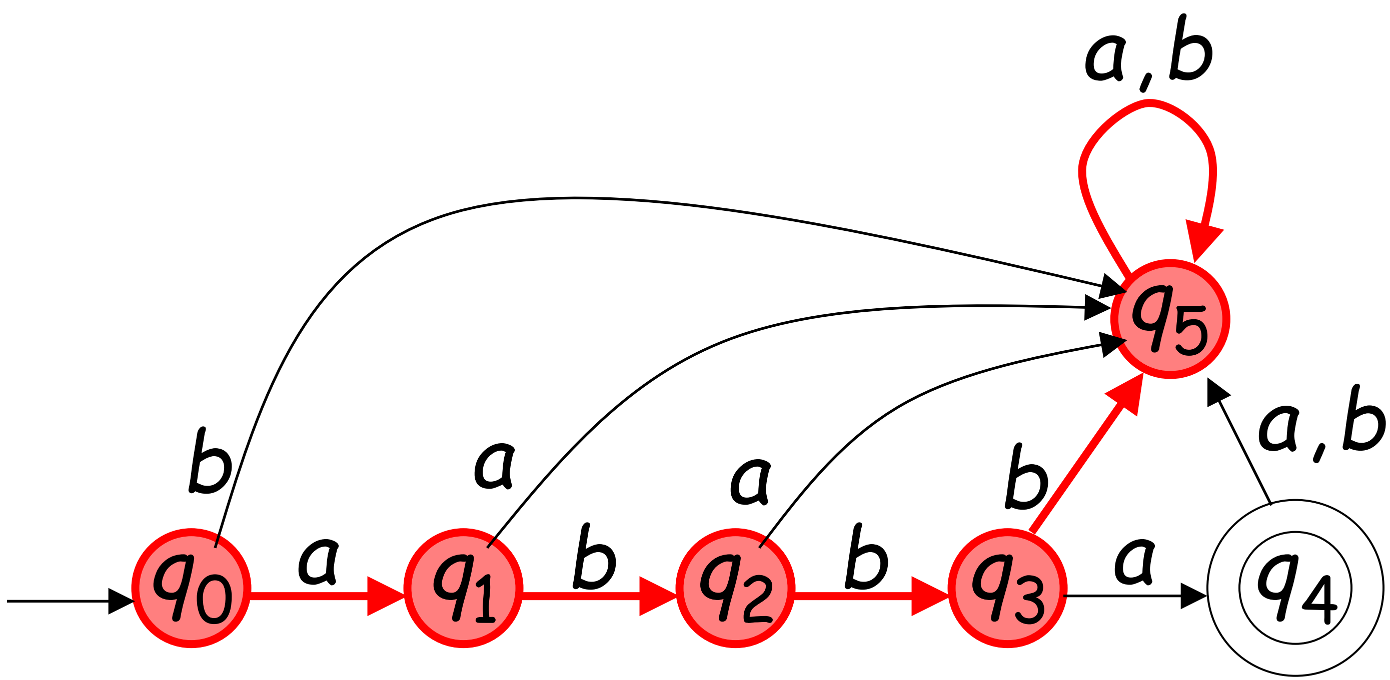
$$\delta^*(q, w) = q'$$

Descrive lo stato che risulta dopo aver
Esaminata la stringa w
a partire dallo stato q

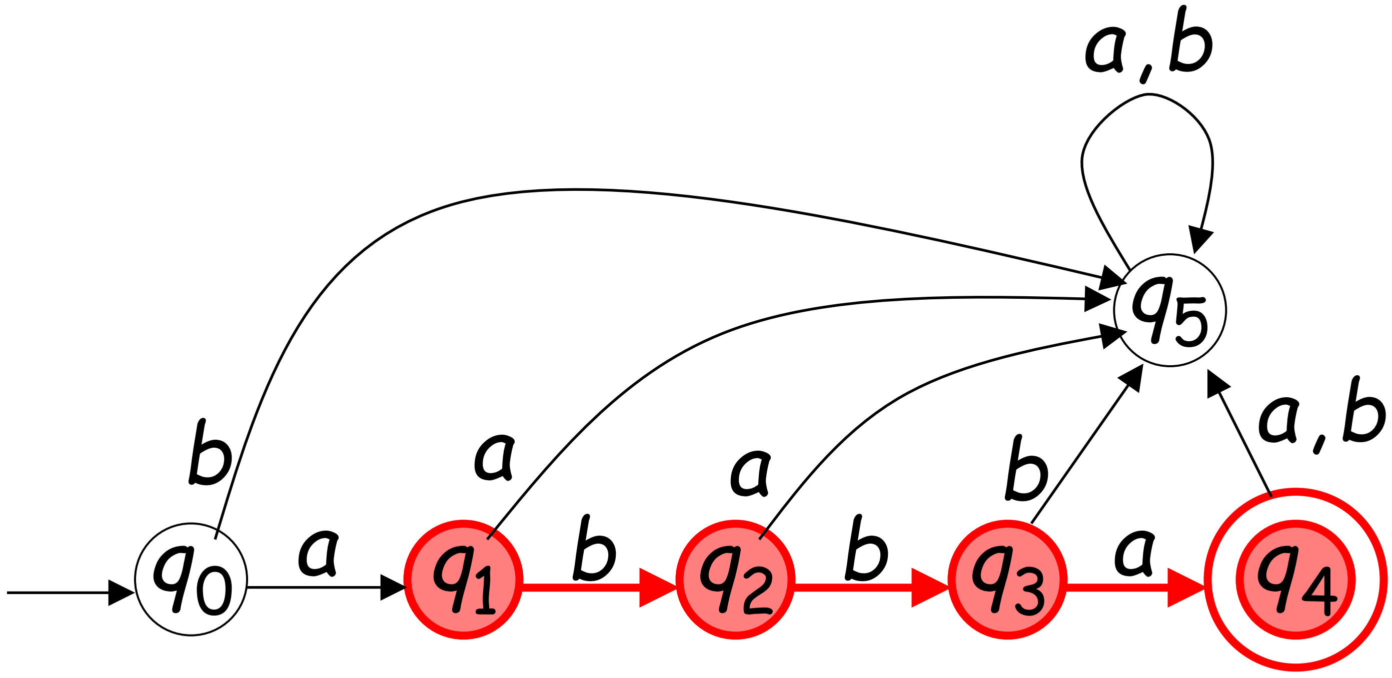
esempio: $\delta^*(q_0, ab) = q_2$



$$\delta^*(q_0, abbbaa) = q_5$$



$$\delta^*(q_1, bba) = q_4$$



Caso speciale:

Delta*(q, aw)=Delta*(delta(q,a), w);
Delta*(q, Lambda)=q

Per ogni stato q

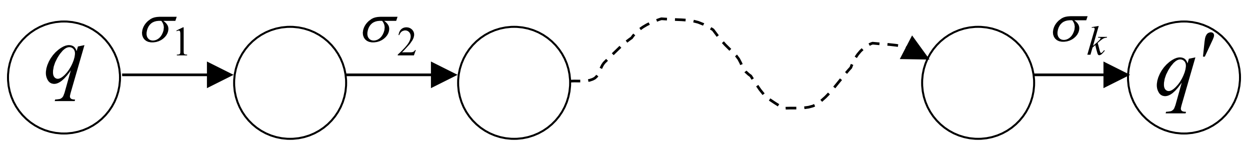
$$\delta^*(q, \lambda) = q$$

:

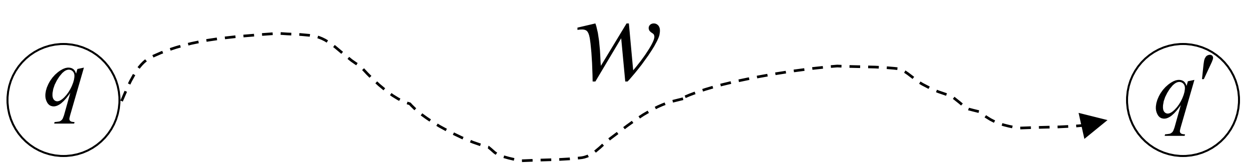
$$\delta^*(q,w) = q'$$

Implica che vi è un cammino di transizione

$$w = \sigma_1 \sigma_2 \cdots \sigma_k$$

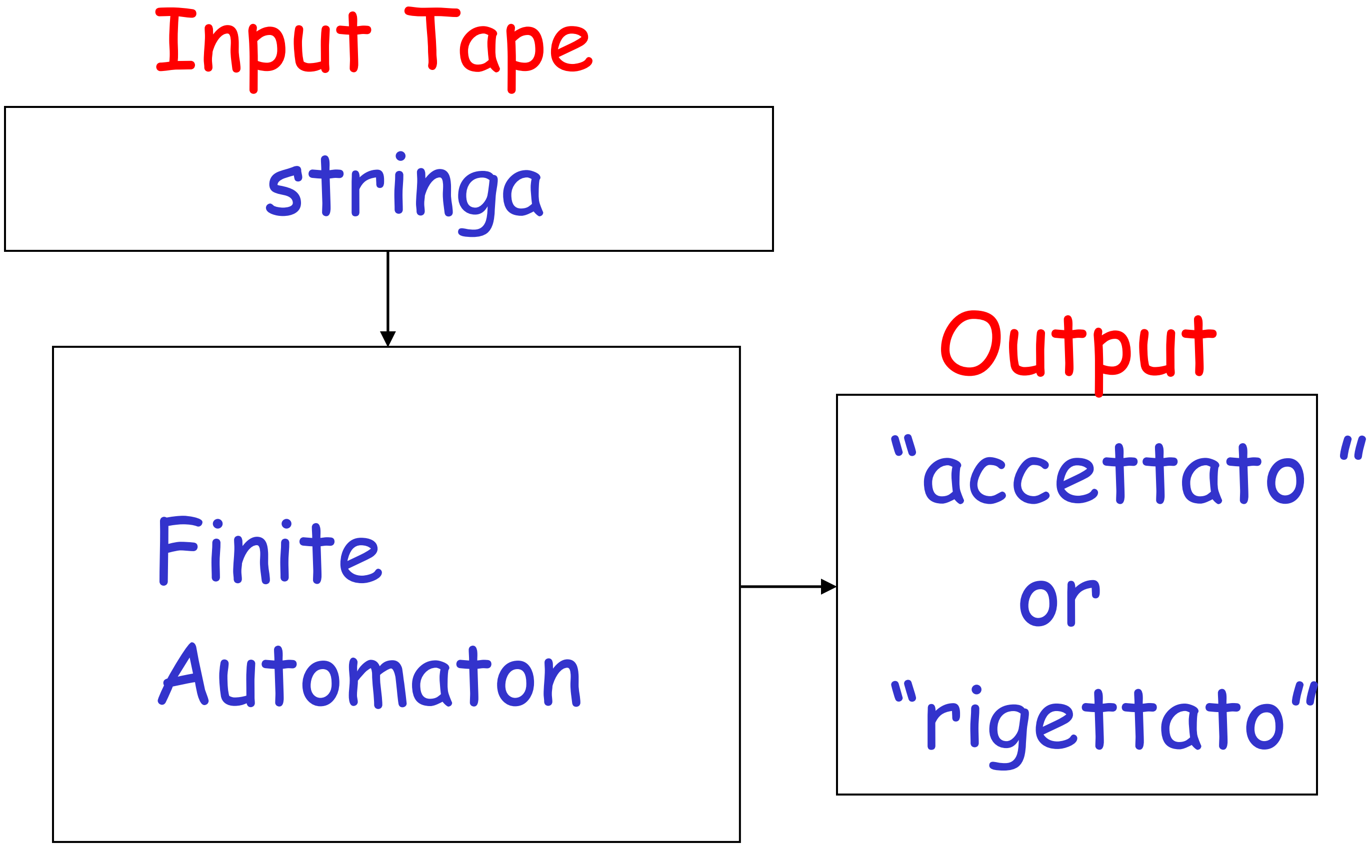


Alcuni stati possono essere ripeturi



Complessità costante sull'input

Deterministic Finite Automaton (DFA)



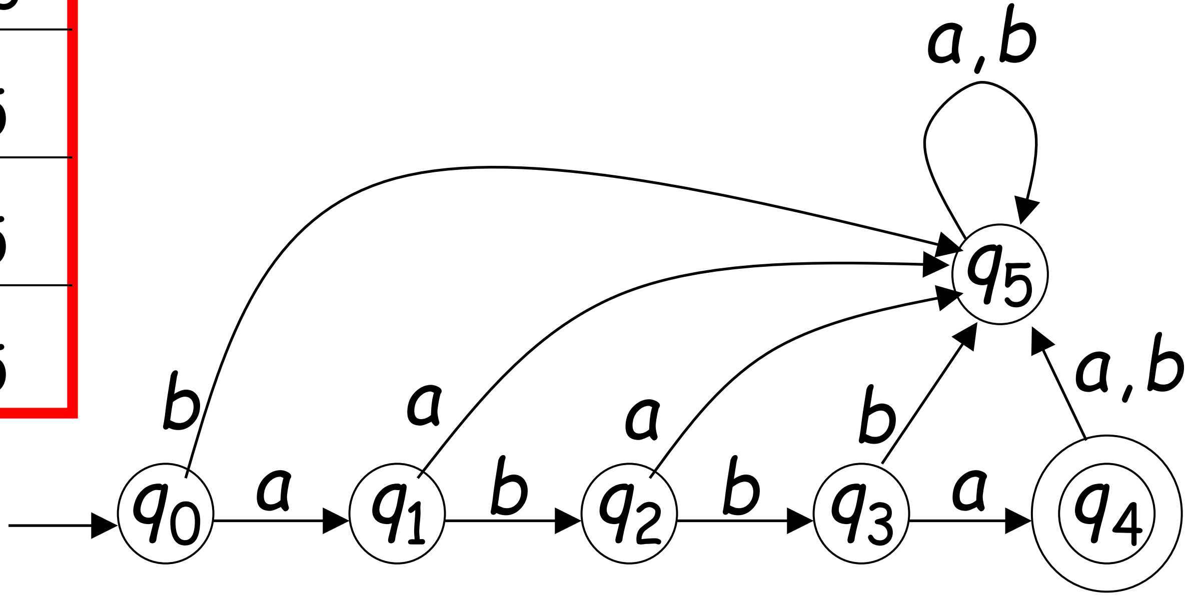
U(automa, input)=automa(input)

symbols

states

δ	a	b
q_0	q_1	q_5
q_1	q_5	q_2
q_2	q_5	q_3
q_3	q_4	q_5
q_4	q_5	q_5
q_5	q_5	q_5

δ



U(automa, input)=automa(input)

Linguaggio accettato da un DFA

Linguaggio di un DFA: M

È denotato come $L(M)$

E contiene tutte le stringhe

Accettate da M

Un linguaggio L'

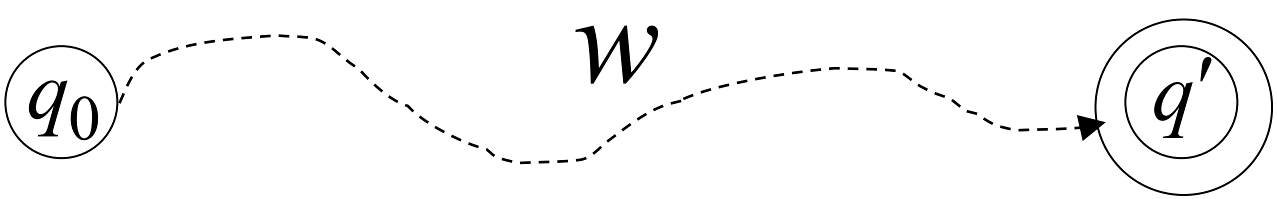
È accettato (o riconosciuto)

Da un DFA M se $L(M) = L'$

Per un DFA $M = (Q, \Sigma, \delta, q_0, F)$

Il linguaggio accettato da M:

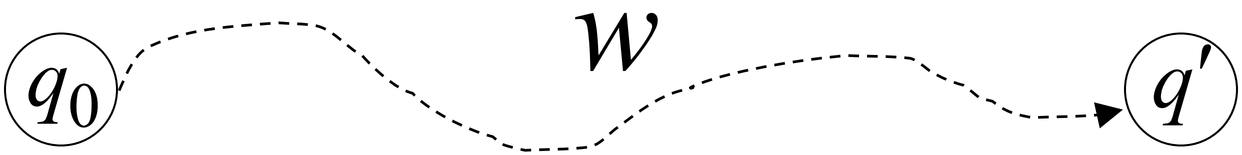
$$L(M) = \{ w \in \Sigma^* : \delta^*(q_0, w) \in F \}$$



$q' \in F$

Linguaggio rifiutato da M :

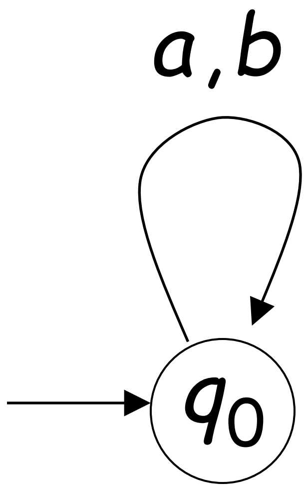
$$\overline{L(M)} = \{ w \in \Sigma^* : \delta^*(q_0, w) \notin F \}$$



$q' \notin F$

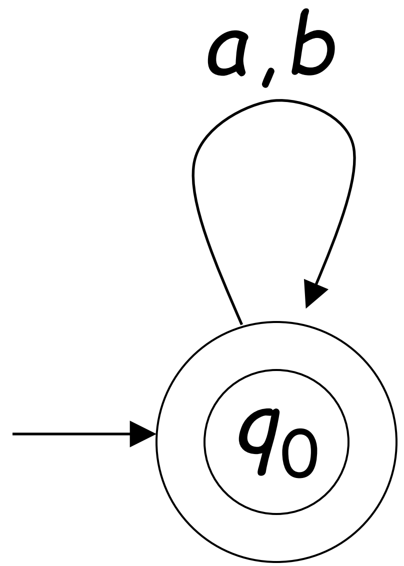
DFA esempi

$$\Sigma = \{a, b\}$$



$$L(M) = \{\ \}$$

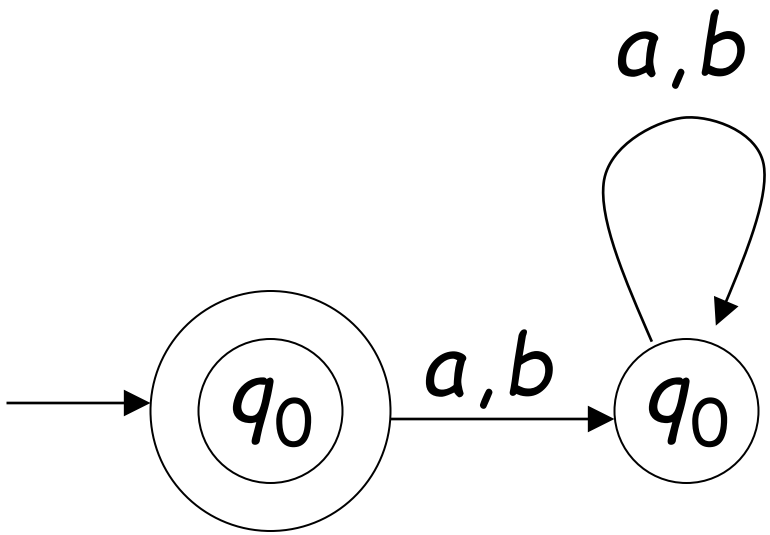
Linguaggio vuoto



$$L(M) = \Sigma^*$$

Tutte le stringhe

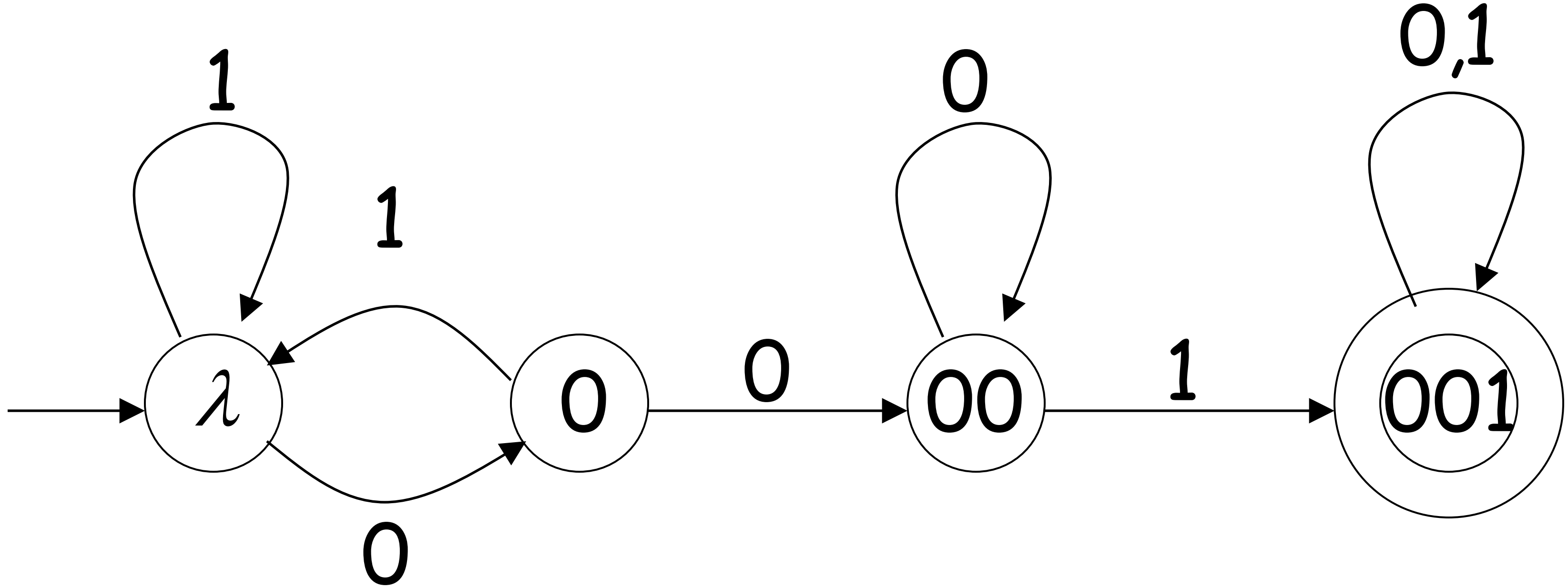
$$\Sigma = \{a, b\}$$



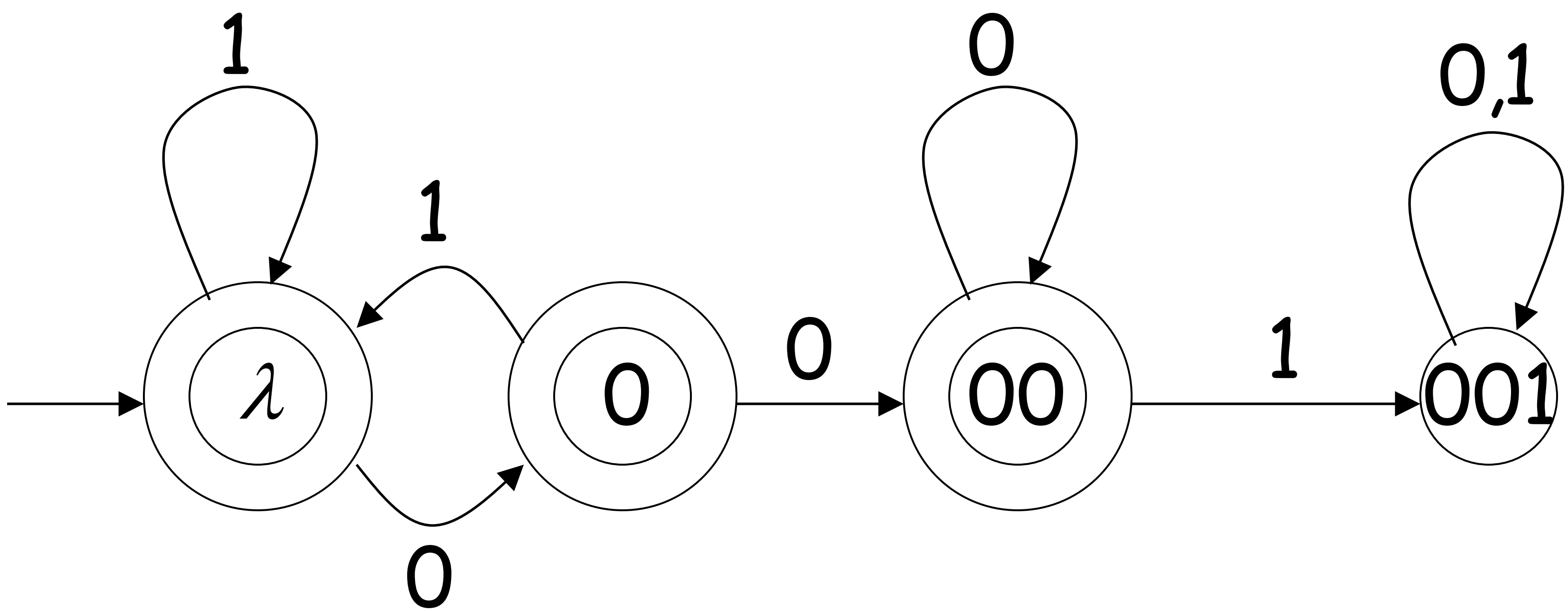
$$L(M) = \{\lambda\}$$

Linguaggio che riconosce le Stringa vuota

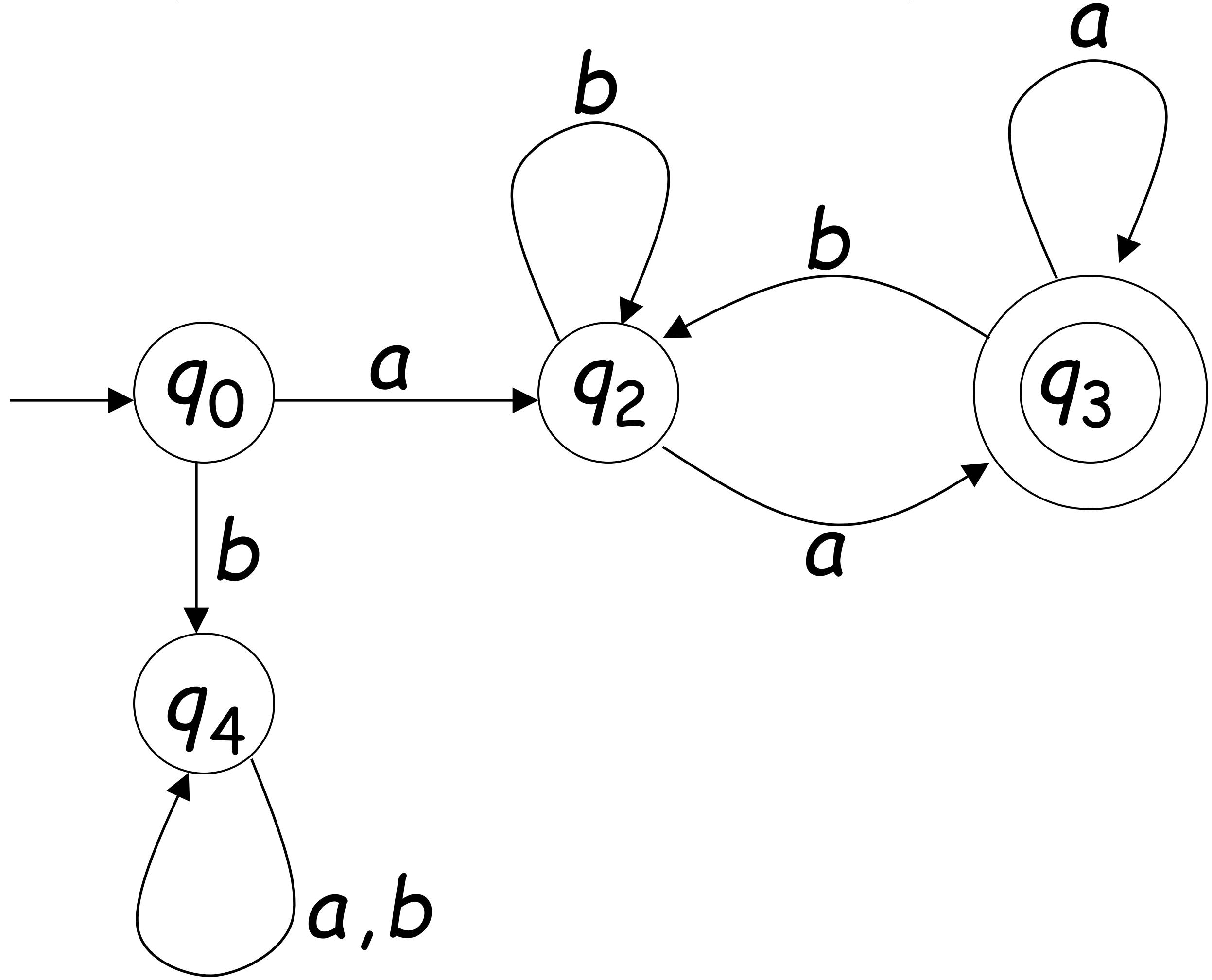
$L(M)$ = { tutte le stringhe binarie che contengono la sottostringa 001}



$L(M)$ = { tutte le stringhe binarie che non Contengono 001 }



$$L(M) = \{awa : w \in \{a,b\}^*\}$$



Linguaggi regolari

Definizione:

Un linguaggio L è regolare se esiste un DFA M che lo accetta ($L(M) = L$)

I linguaggi accettati da tutti i DFA formano la famiglia dei linguaggi regolari

Esempi di linguaggi regolari:

$\{abba\}$ $\{\lambda, ab, abba\}$

$\{a^n b : n \geq 0\}$ $\{awa : w \in \{a,b\}^*\}$

{ tutte stringhe {a,b}* con prefisso ab}

{ all binary strings without substring 001}

$\{x : x \in \{1\}^*$ and x is even$\}$

$\{\ \}$ $\{\lambda\}$ $\{a,b\}^*$

Abbiamo visto in precedenza gli automi regolari che li definiscono

Esitono linguaggi che non sono regolari:

$$L = \{a^n b^n : n \geq 0\}$$

$$ADDITION = \{x + y = z \;\; : x = 1^n, y = 1^m, z = 1^k, \\ n + m = k\}$$

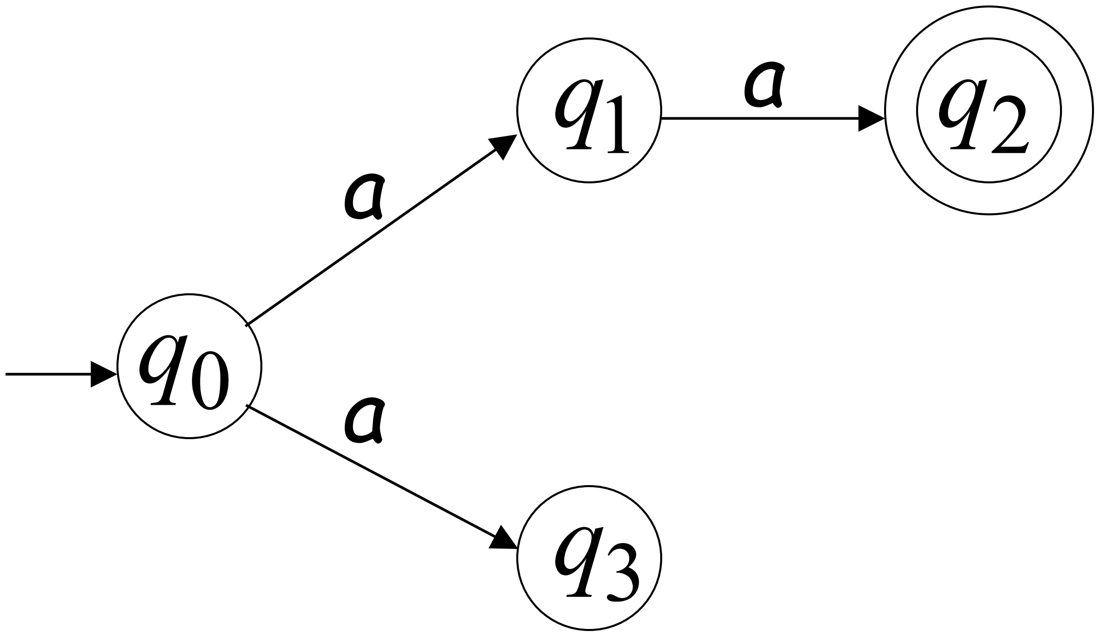
Non esiste nessun DFA che accetta
Questo linguaggio (vedremo più avanti)

Non-Deterministic Finite Automata

Automa finito deterministico
calcolo finito e deterministico
sequenziale, un segmento di lng
dell'input

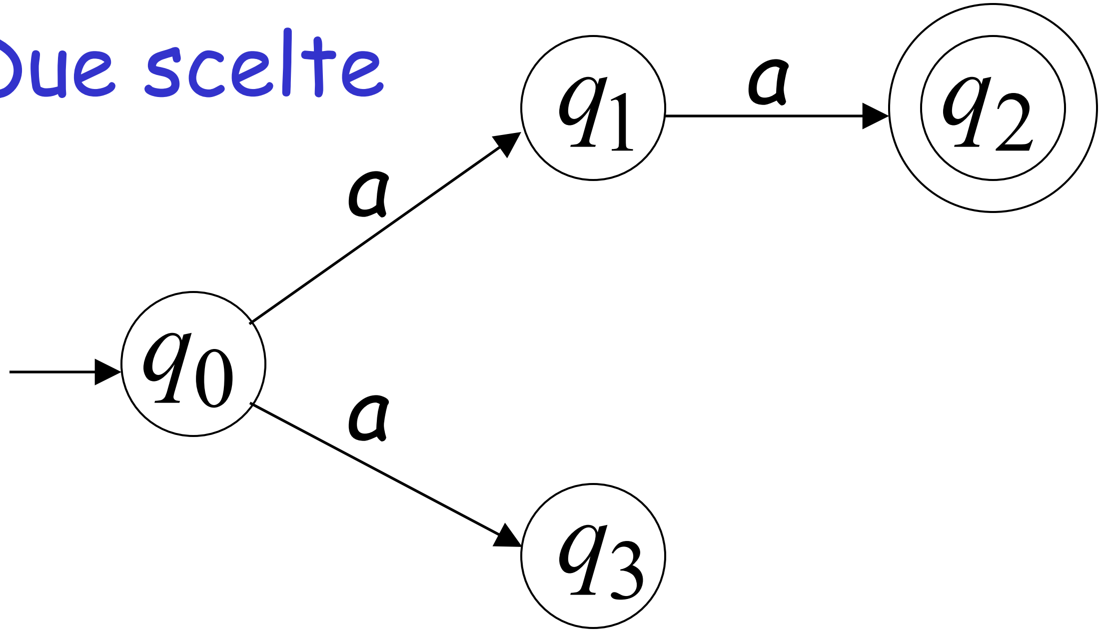
Automi non deterministici (NFA)

alfabeto = $\{a\}$

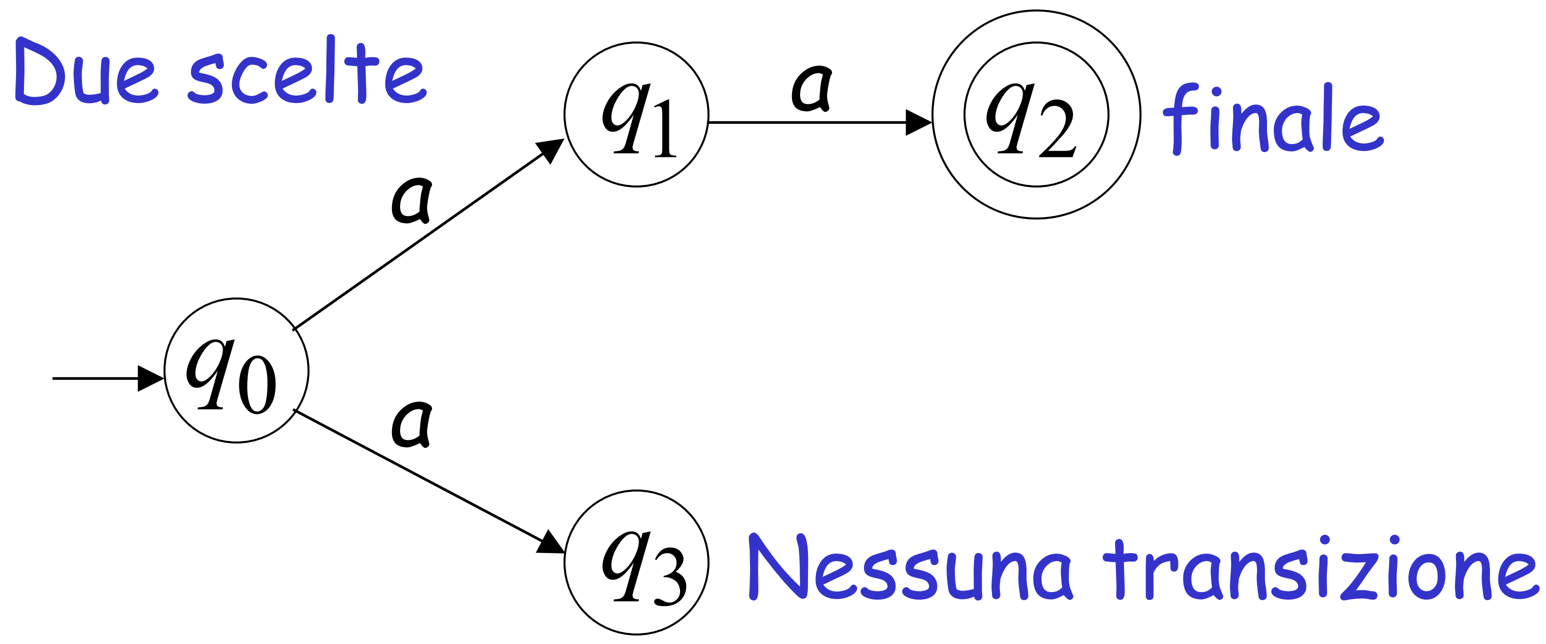


alfabeto = $\{a\}$

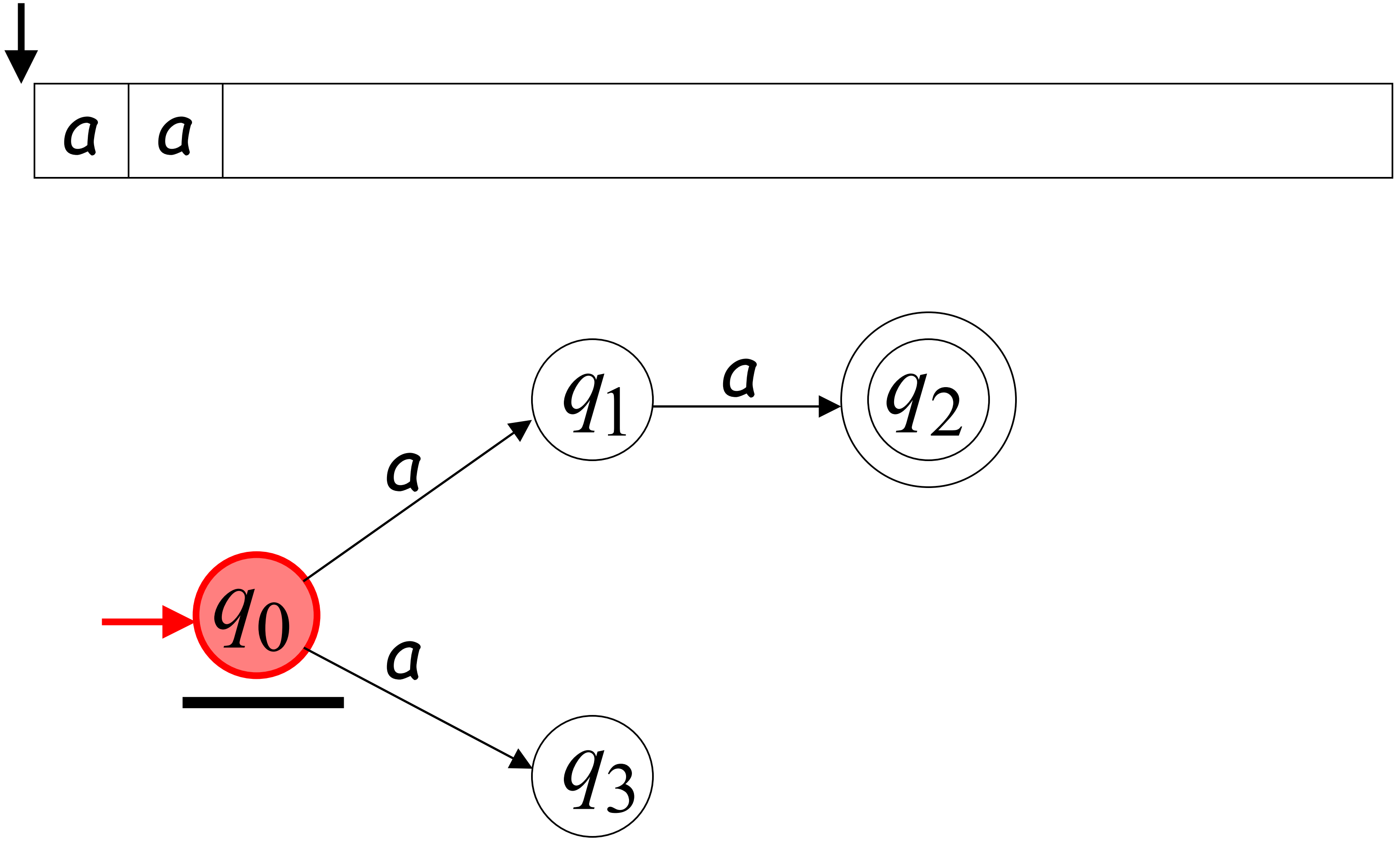
Due scelte



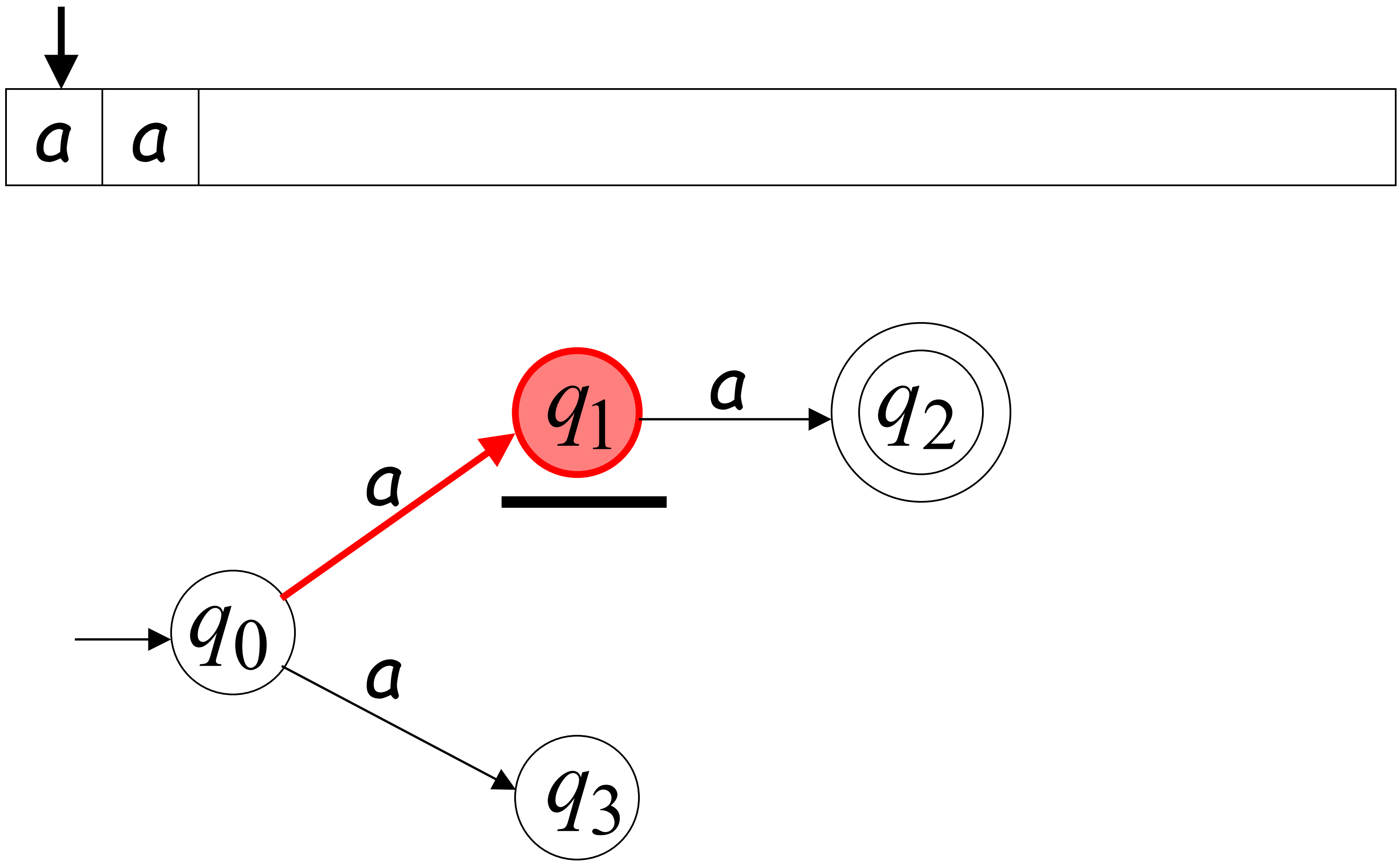
alfabeto = $\{a\}$



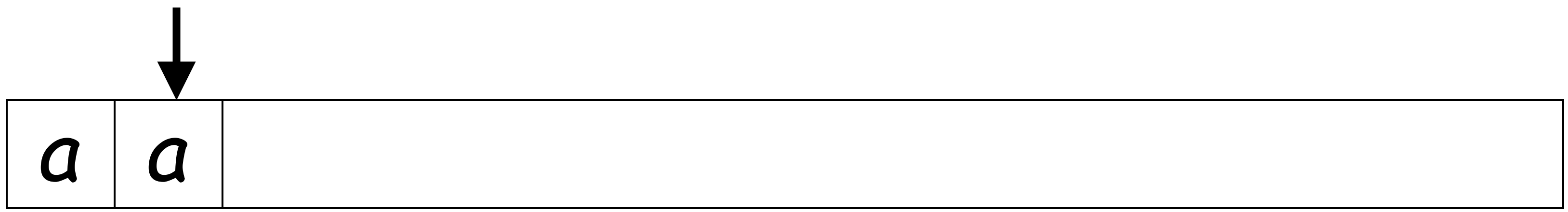
Prima delle due scelte



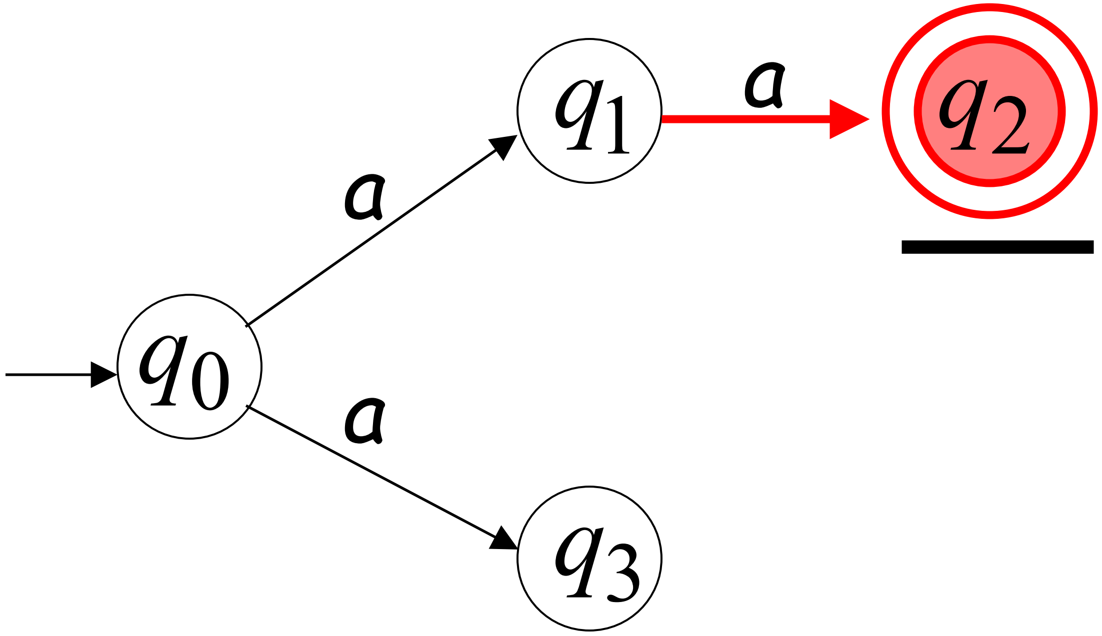
Prima scelta



Prima scelta

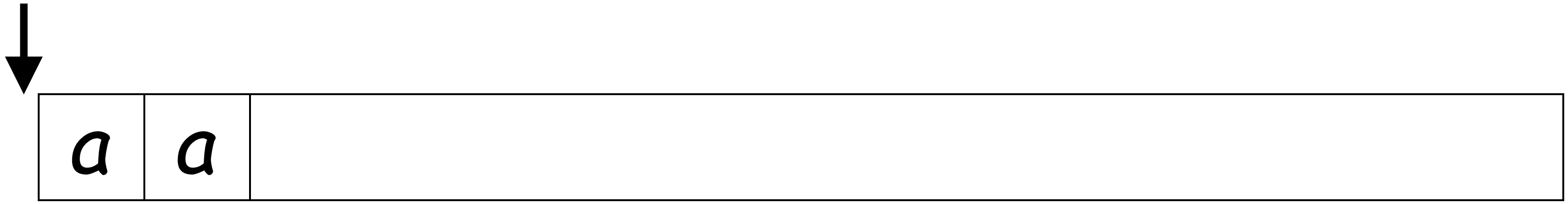


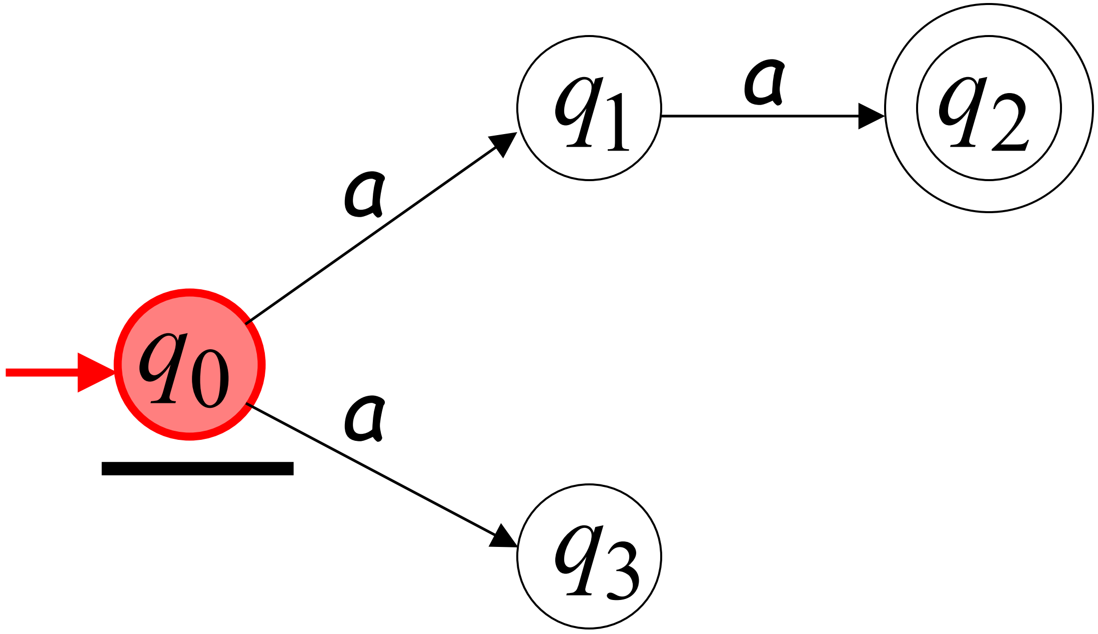
Abbiamo consumato tutto l'input



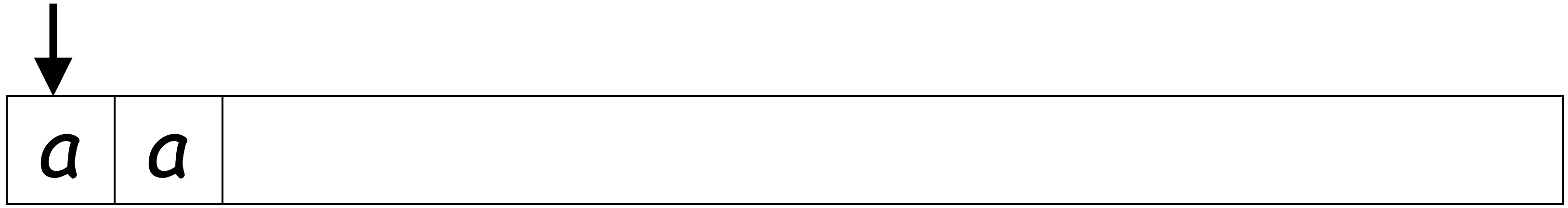
"accettato"

Seconda scelta

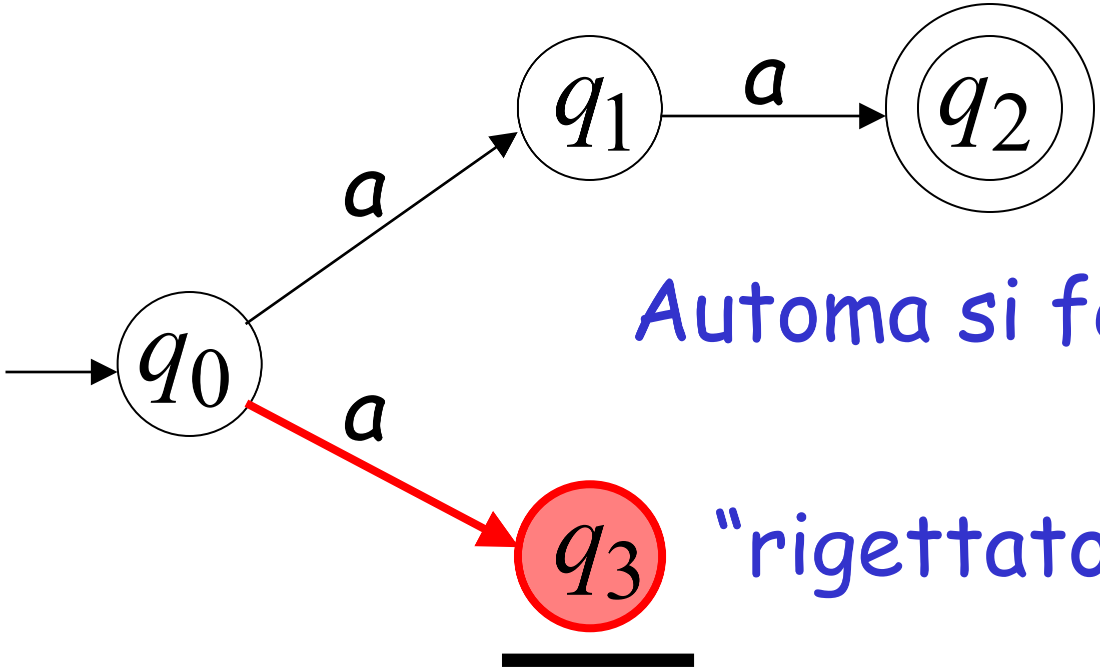




Seconda scelta



Input non può essere tutto usato



Automa si ferma

"rigettato"

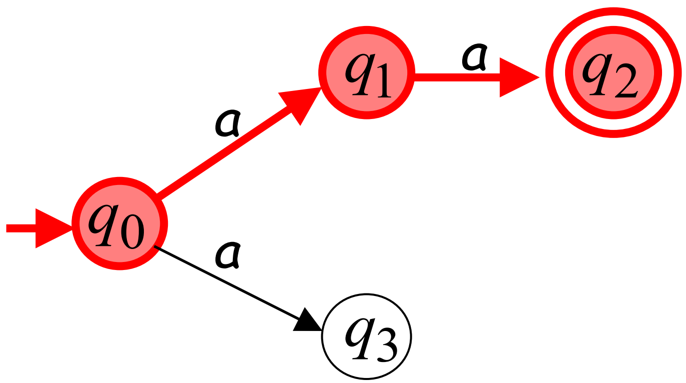
un NFA accetta una stringa:

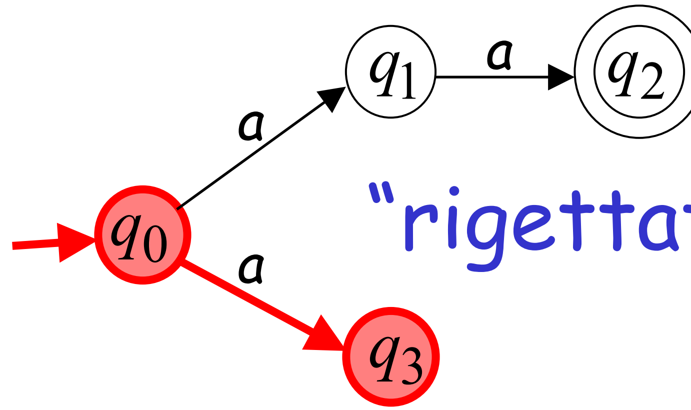
Se esiste una computazione che accetta la stringa

Tutta la stringa di input è stata letta e l'automa Si trova in uno stato finale

aa È accettato dal NFA:

"accettato"



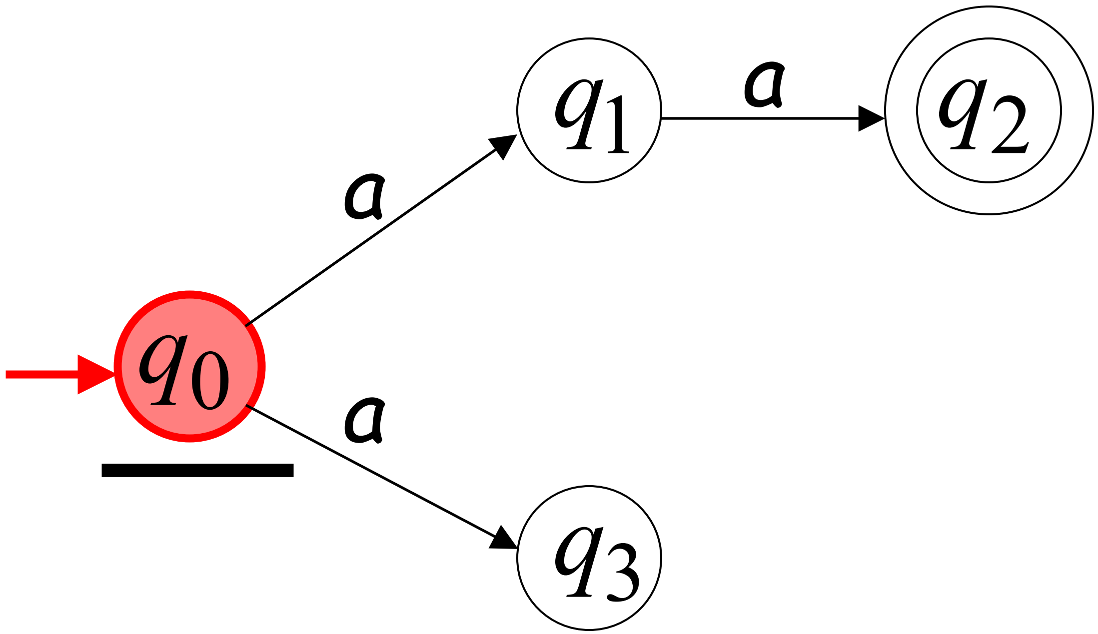


"rigettato"

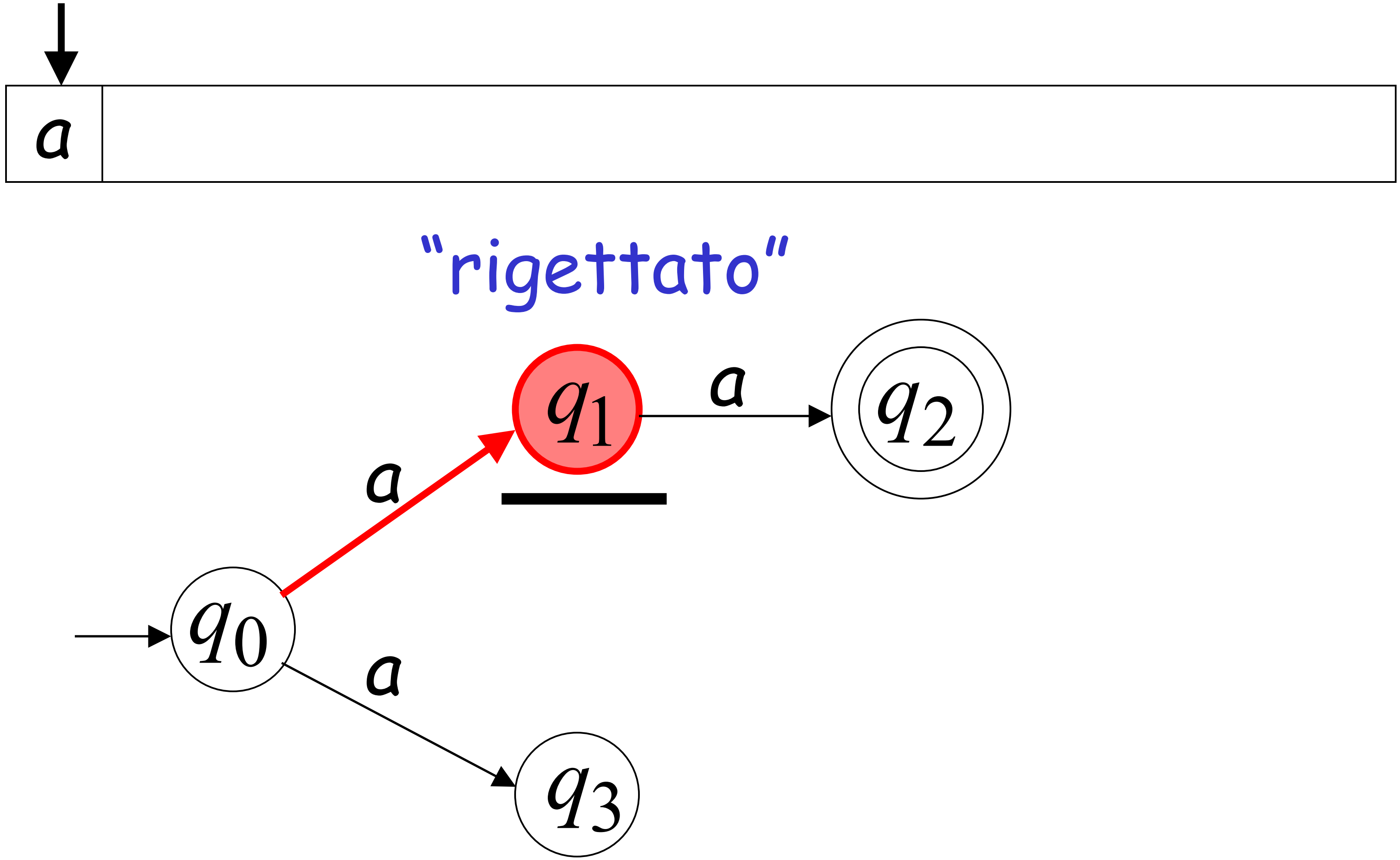
Perchè la Computazione accetta aa

Questa computazione è ignorata

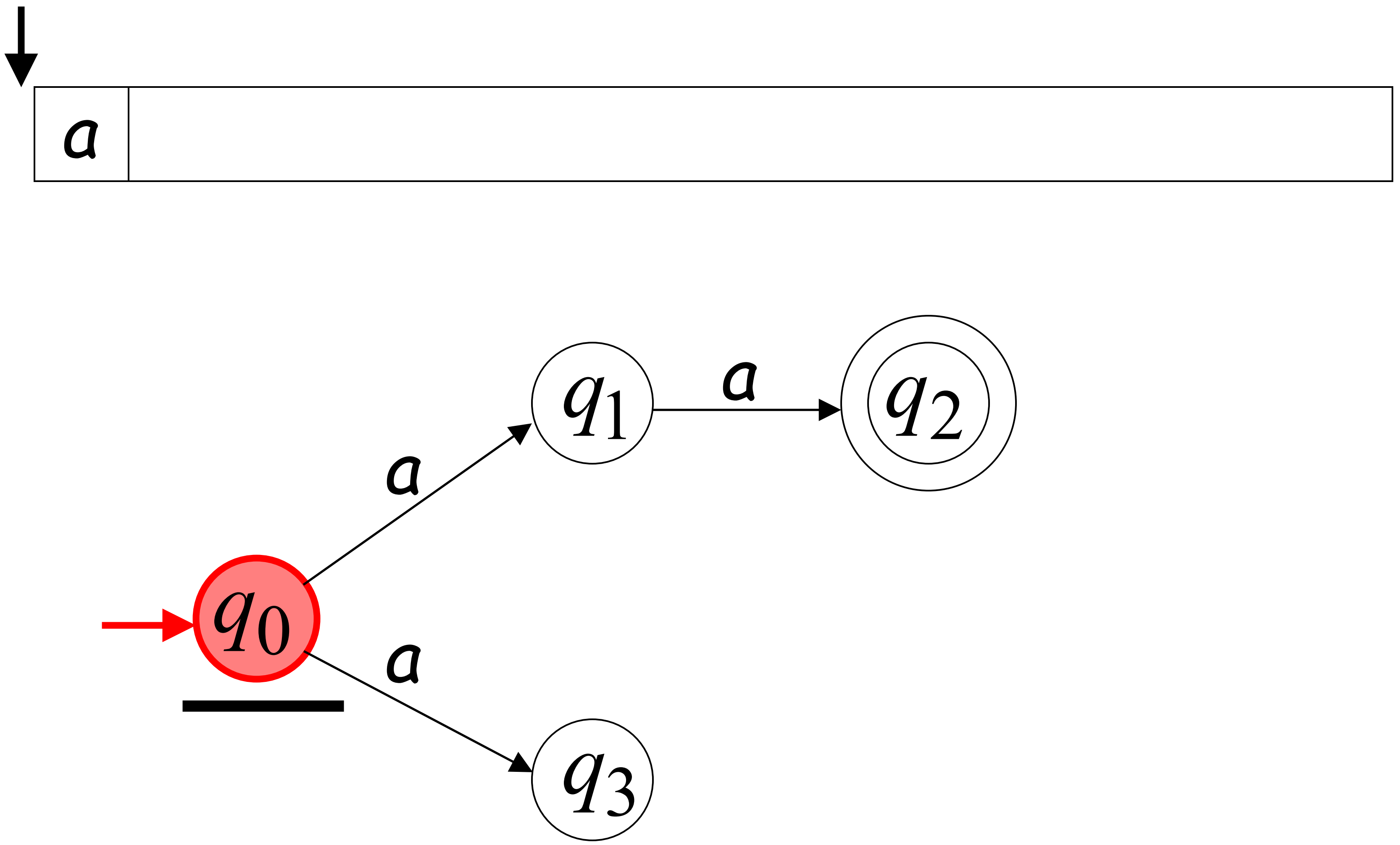
Esempio computazione che rigettà



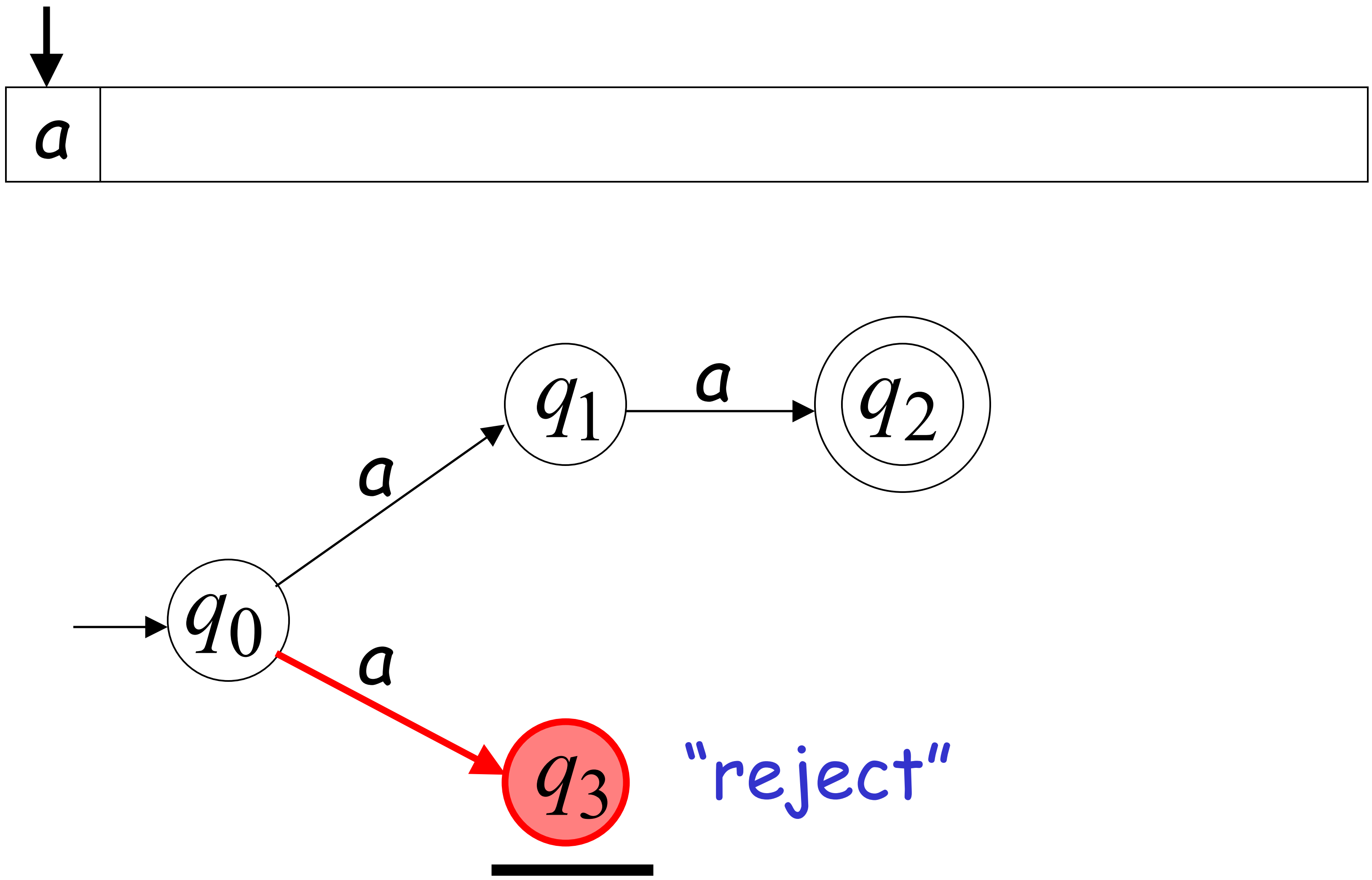
Prima scelta



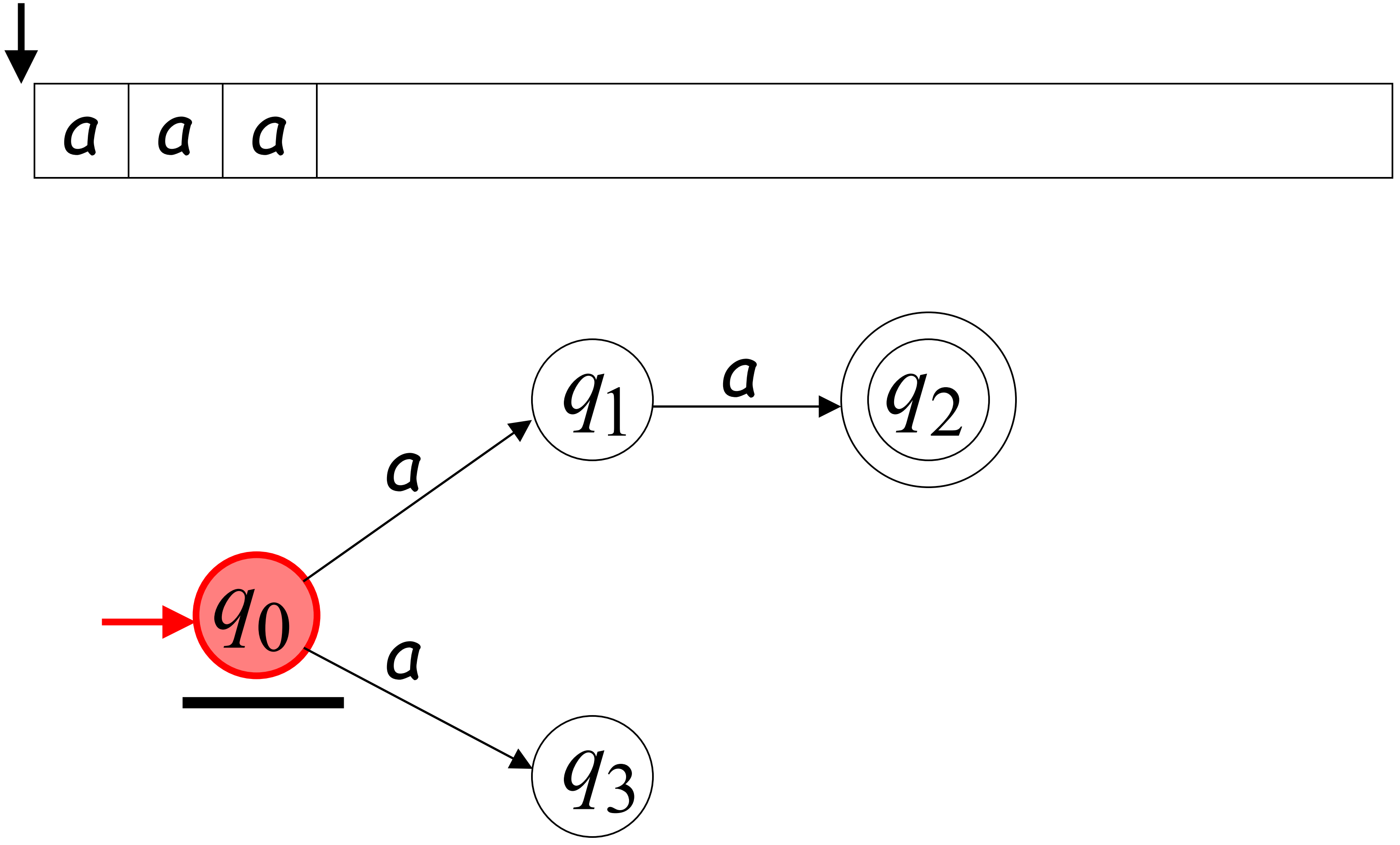
Seconda scelta



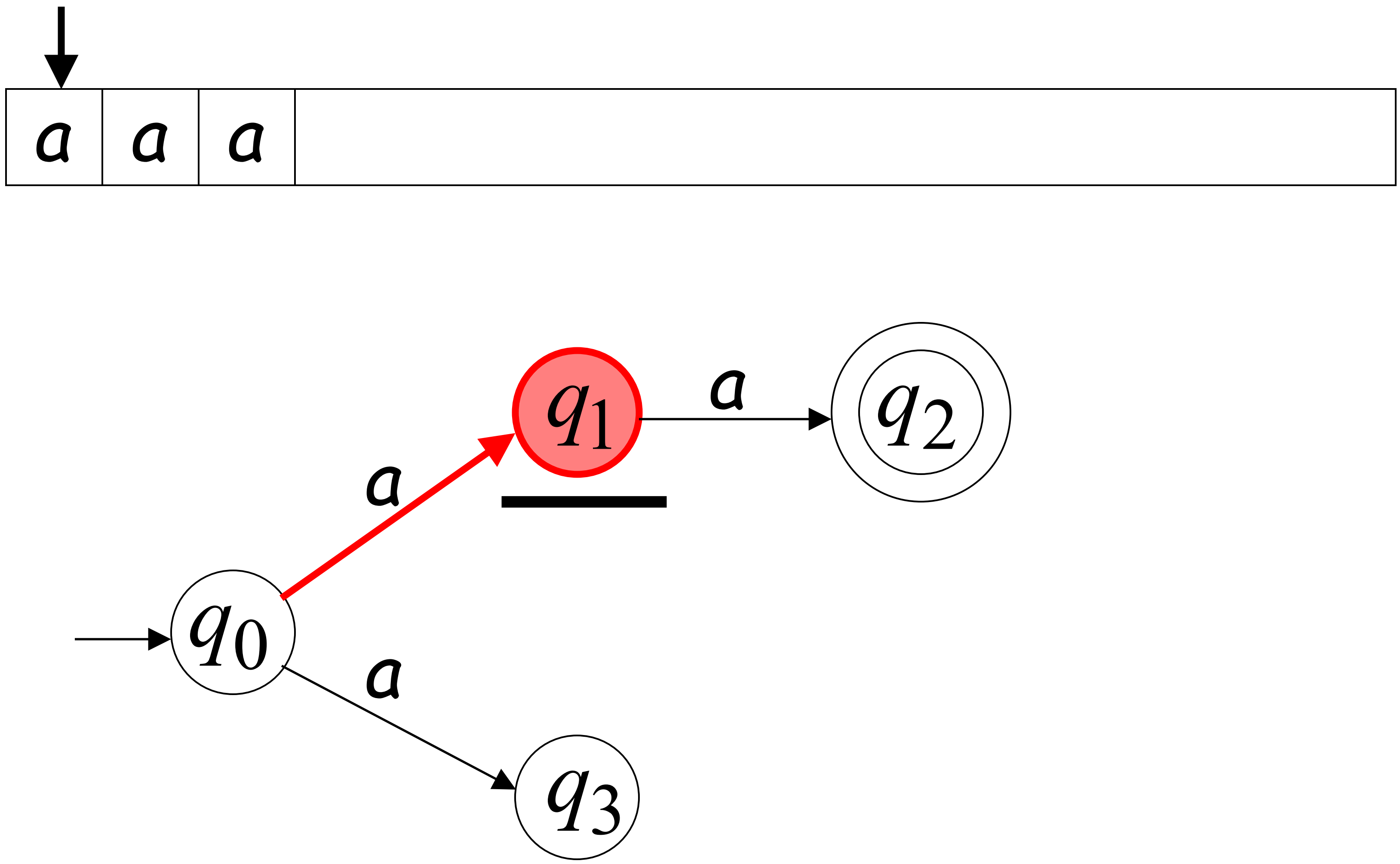
Seconda scelta



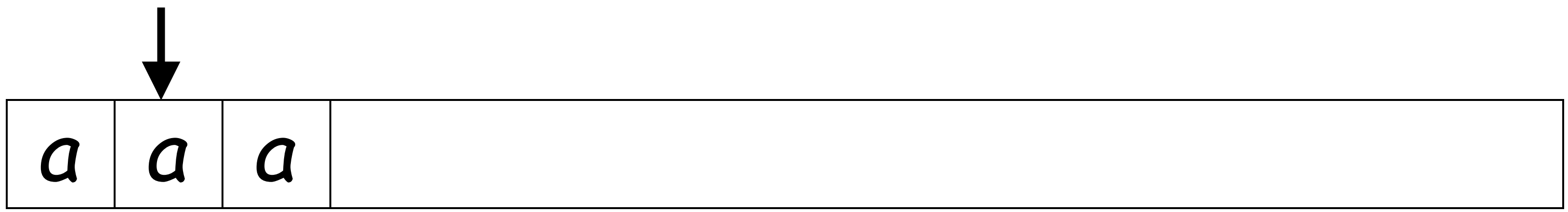
Un altro esempio



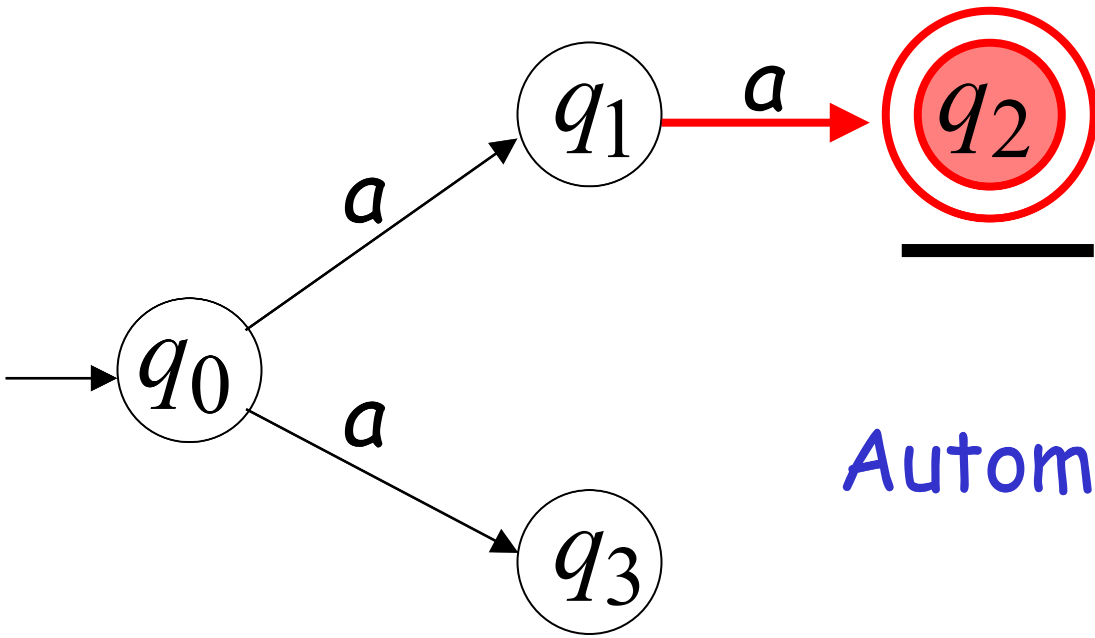
Prima scelta



First Choice



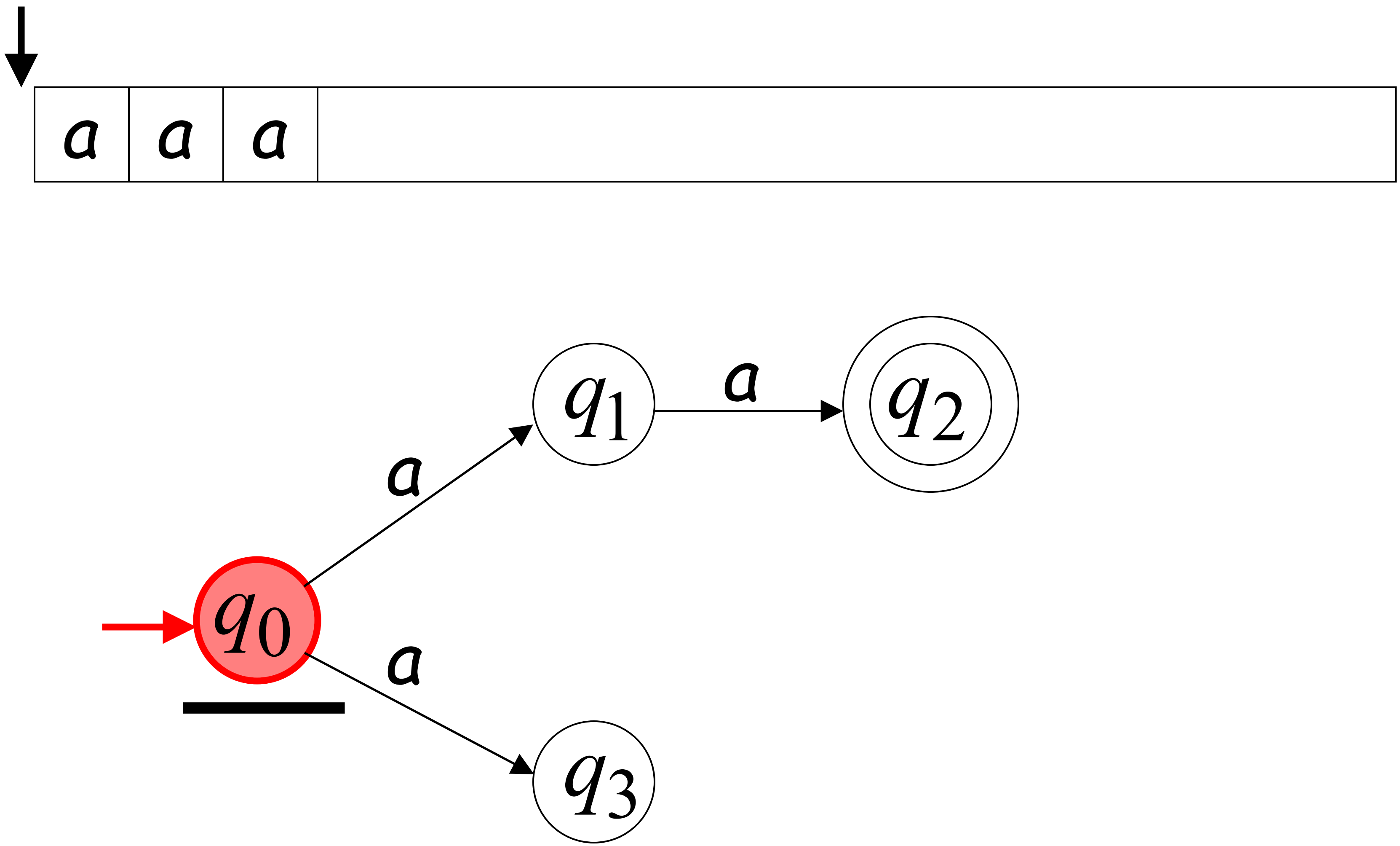
Input cannot be consumed



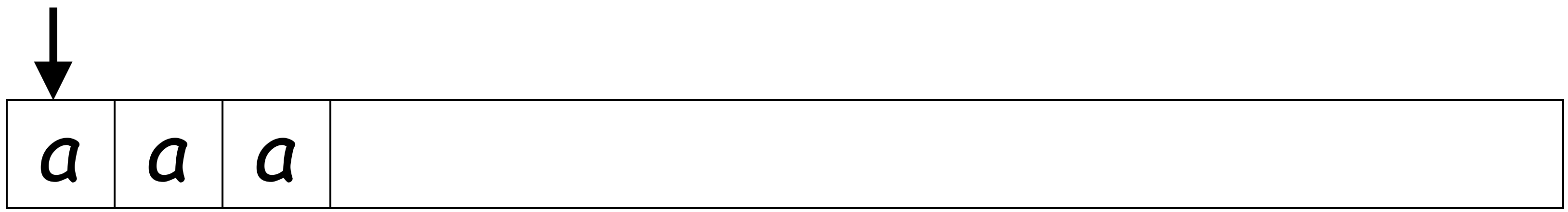
"reject"

Automaton halts

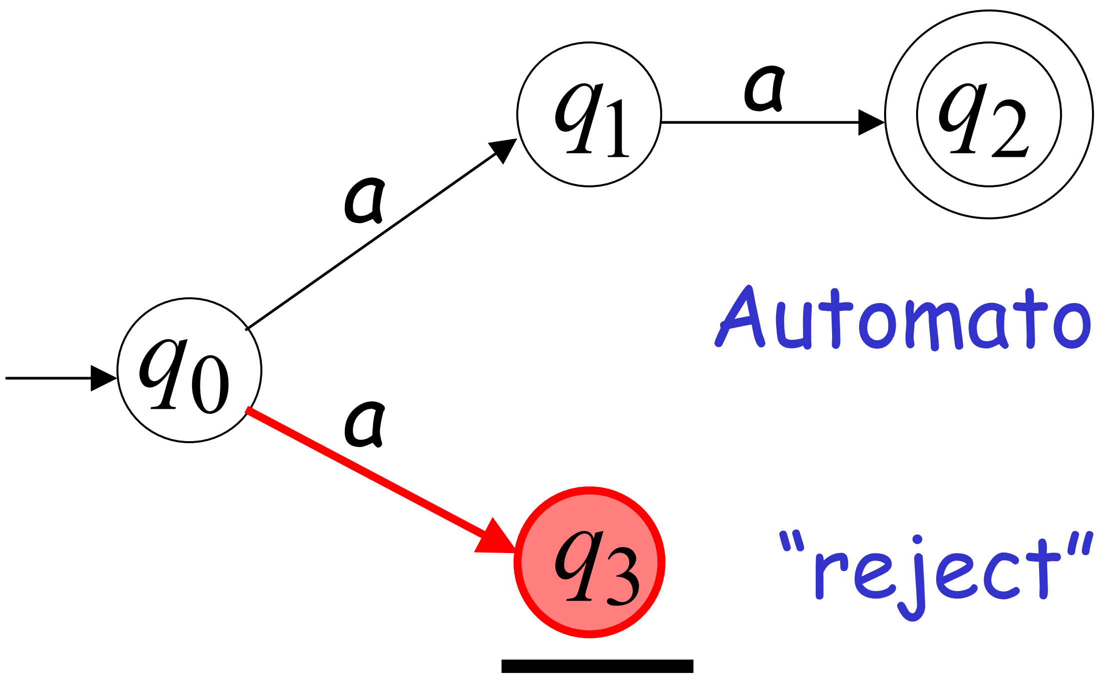
Second Choice



Second Choice



Input non viene tutto consumato



Automaton halts

"reject"

An NFA rejects a string:

Se non vi è una computazione del NFA
che accetta la stringa.

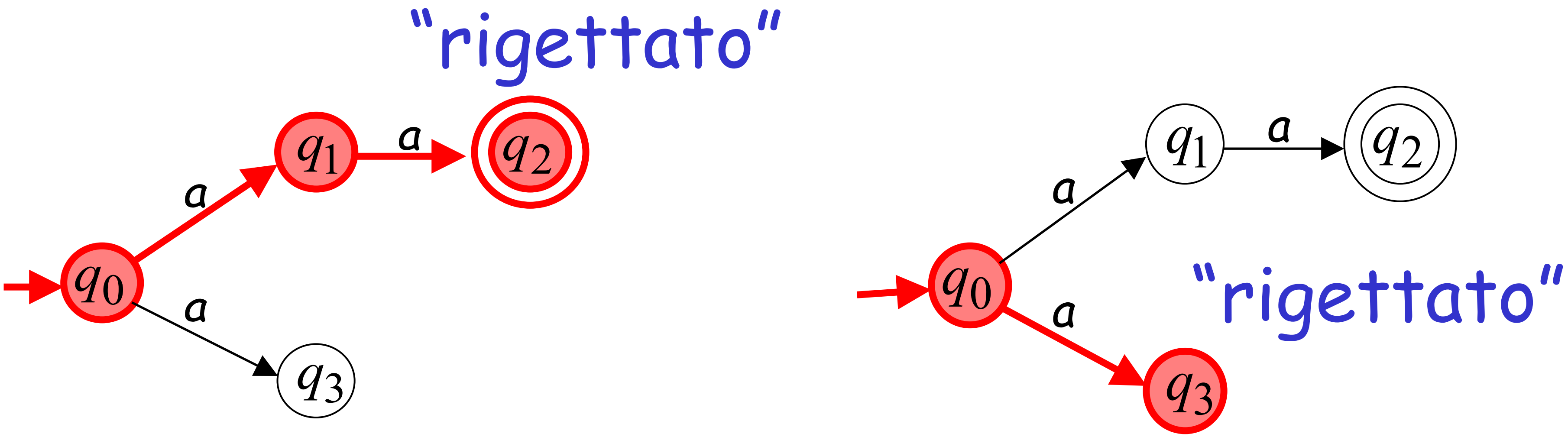
Per ogni computazione:

- tutto l'input è consumato e l'automa
- non ha raggiunto uno stato finale

O

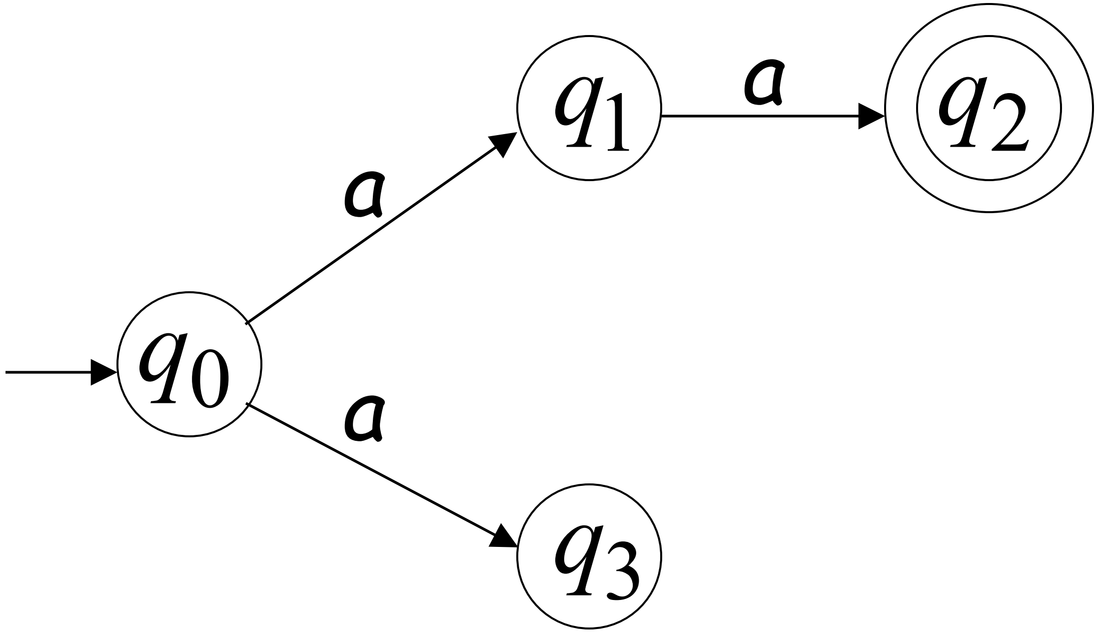
- L'input non è stato tutto consumato

aaa È rigettato dal NFA:

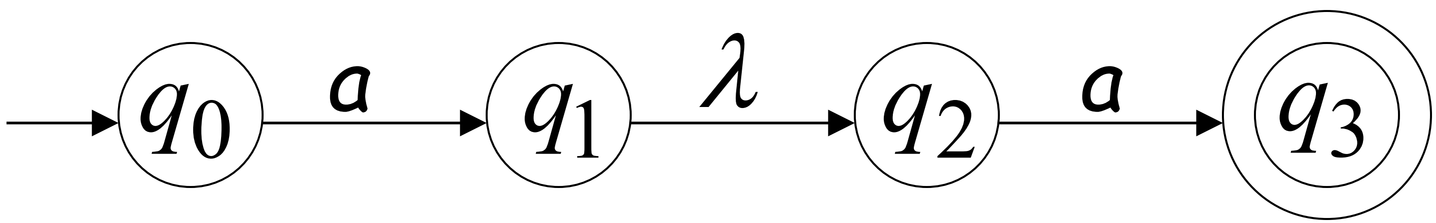


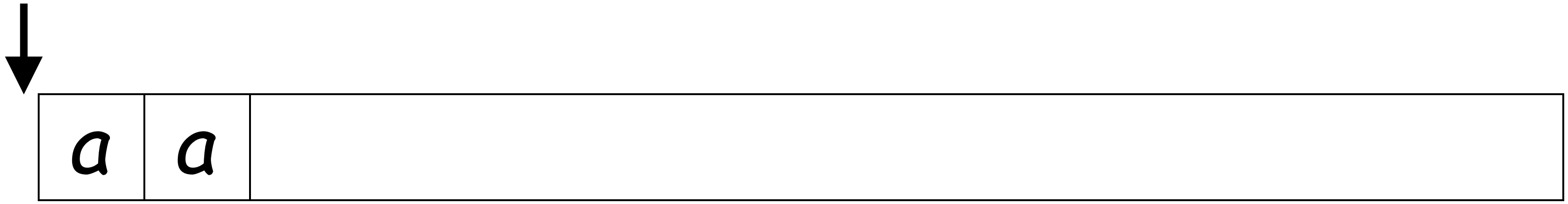
Tutte le possibili computazioni
non raggiungono uno stato finale

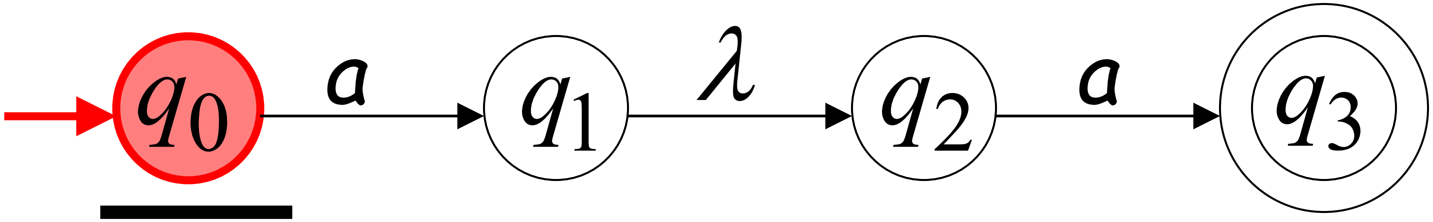
Linguaggio accettato: $L = \{aa\}$

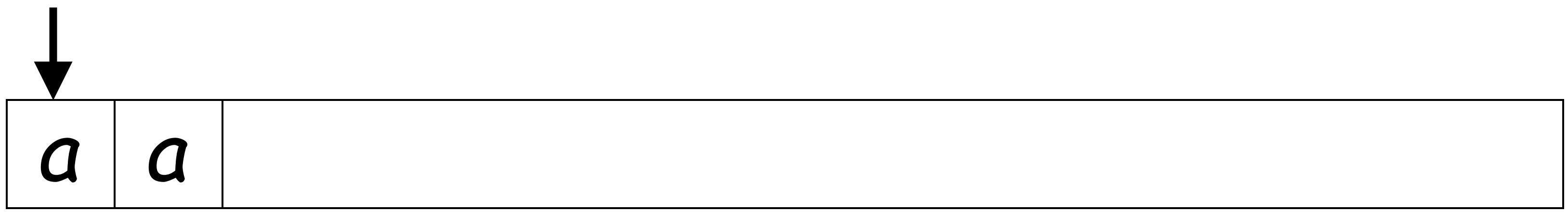


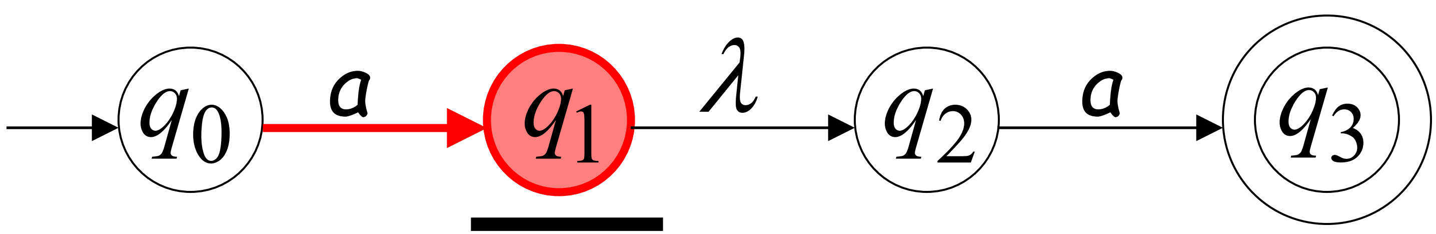
Lambda transizione



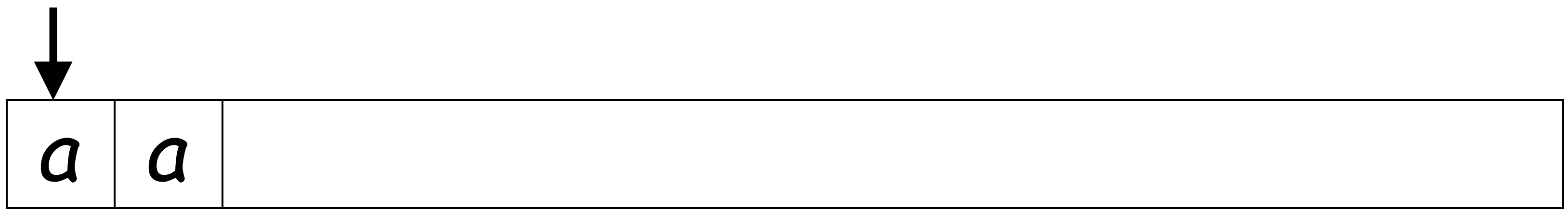

a
a

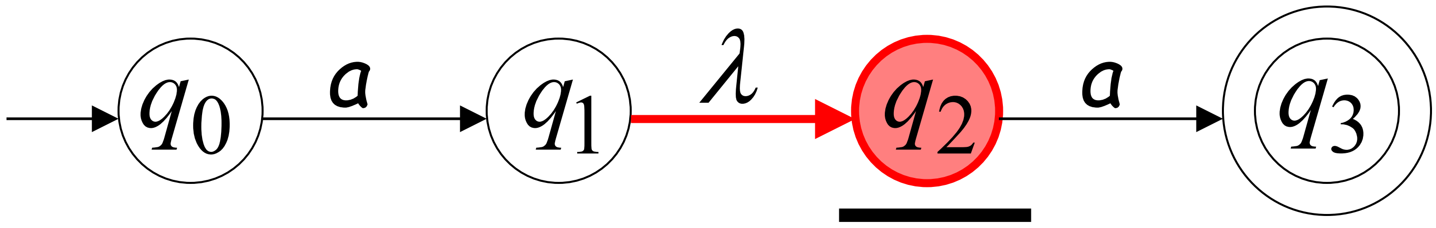

q_0
a
q_1
λ
q_2
a
q_3


a
a

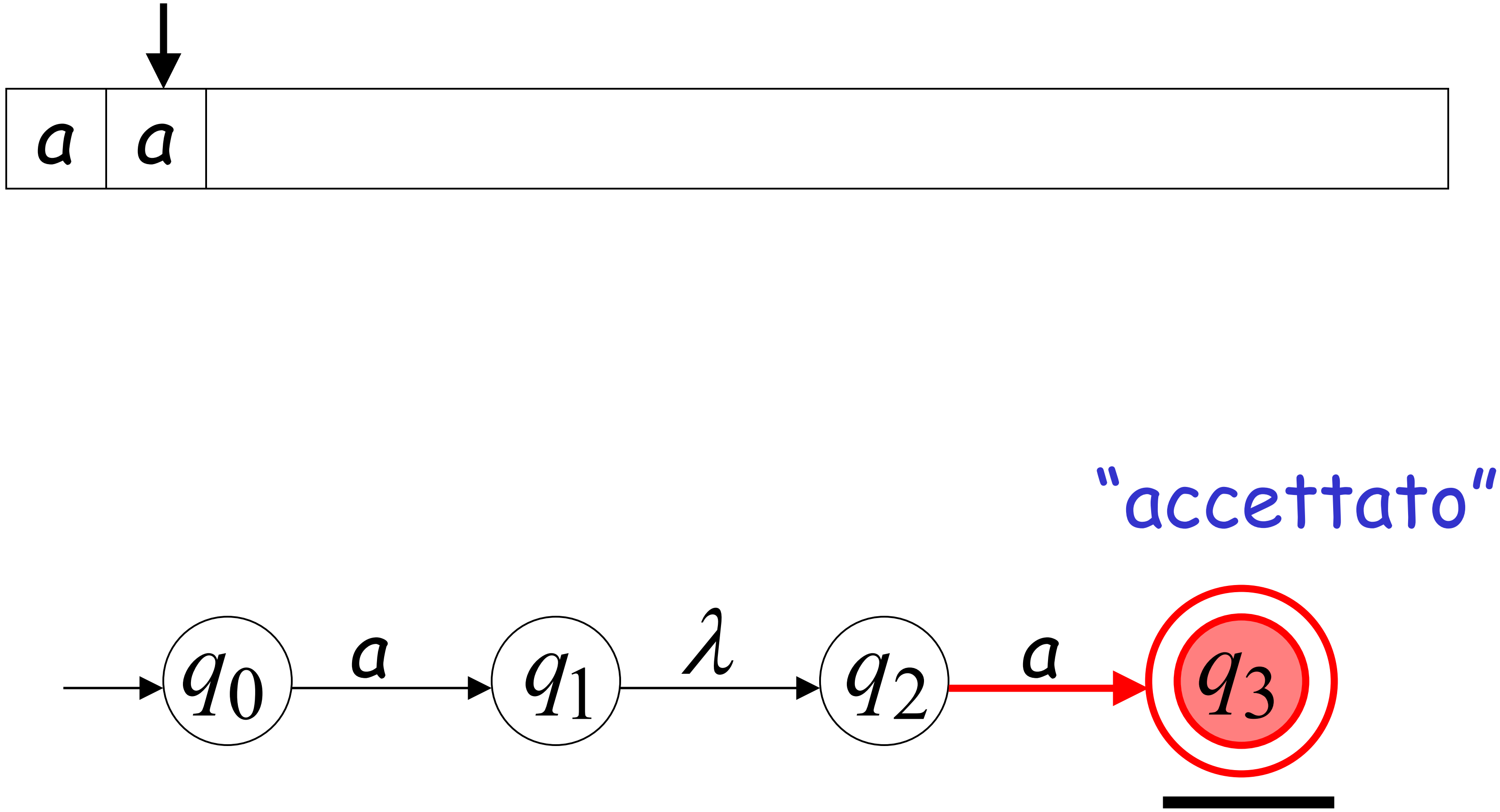

q_0
a
q_1
λ
q_2
a
q_3

La testina dell'input non si muove



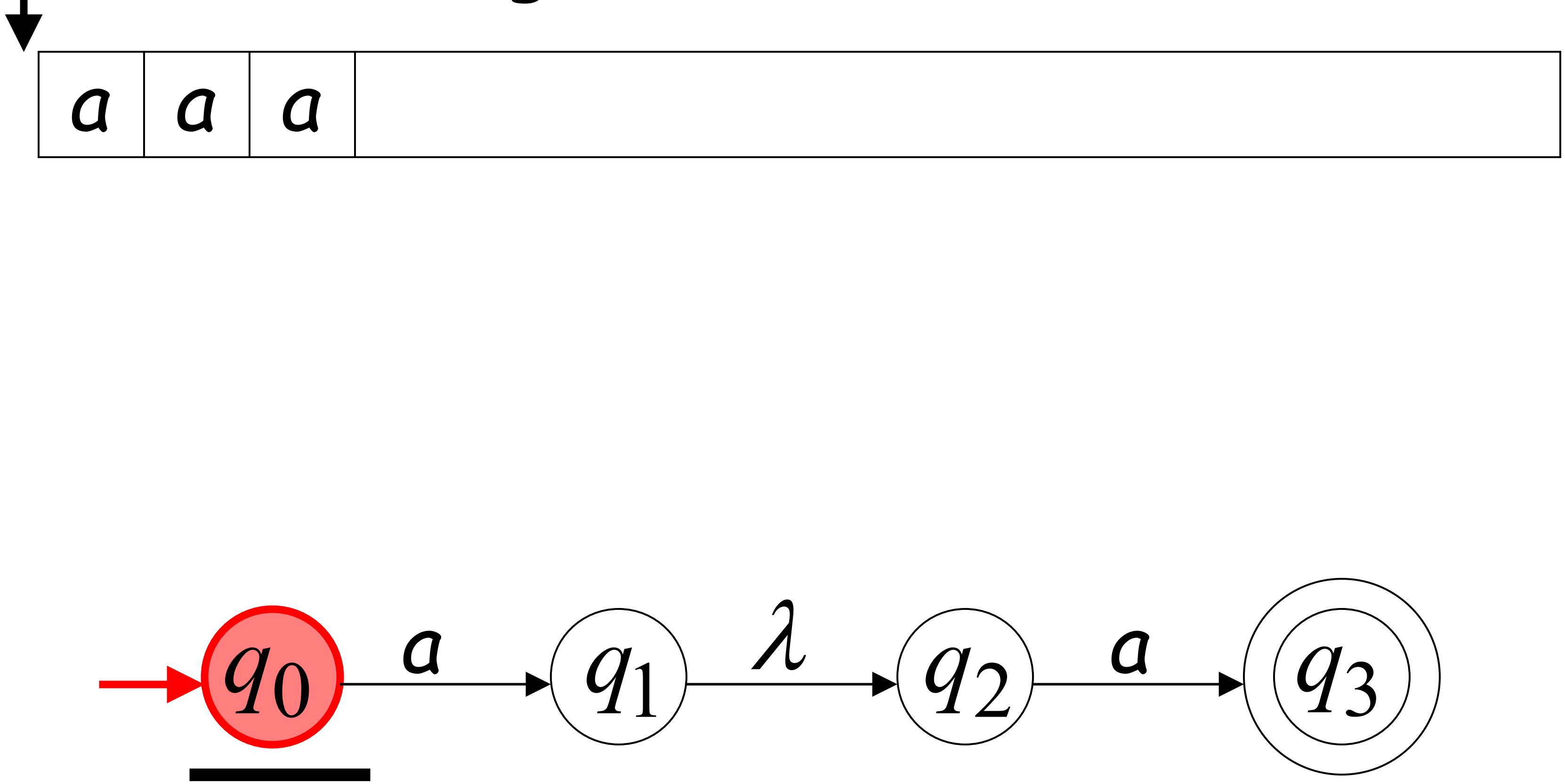


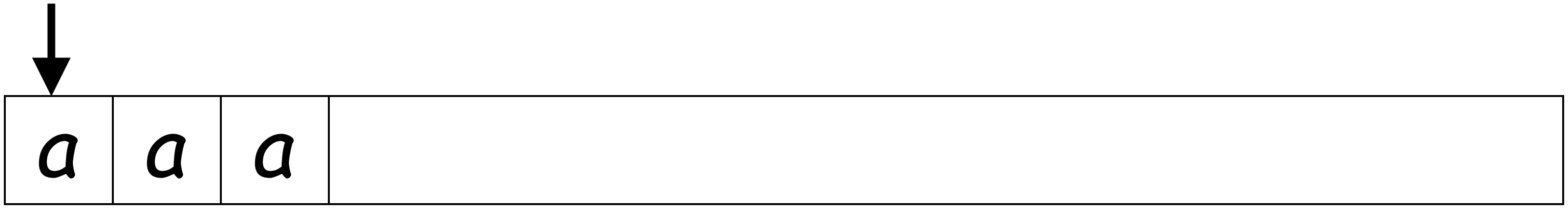
Tutto l'input è esaminato

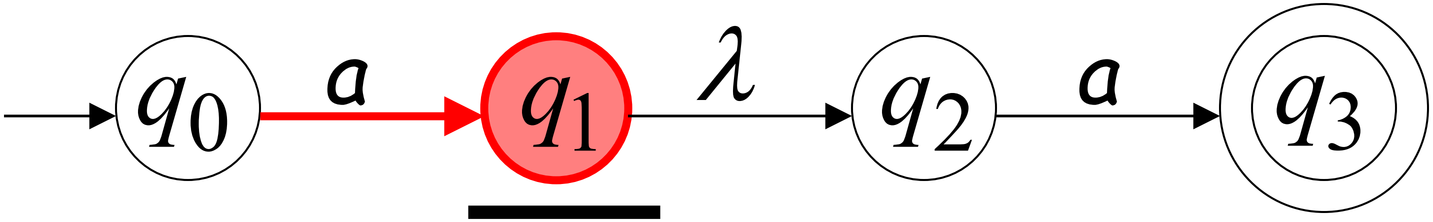


stringa aa è accettata

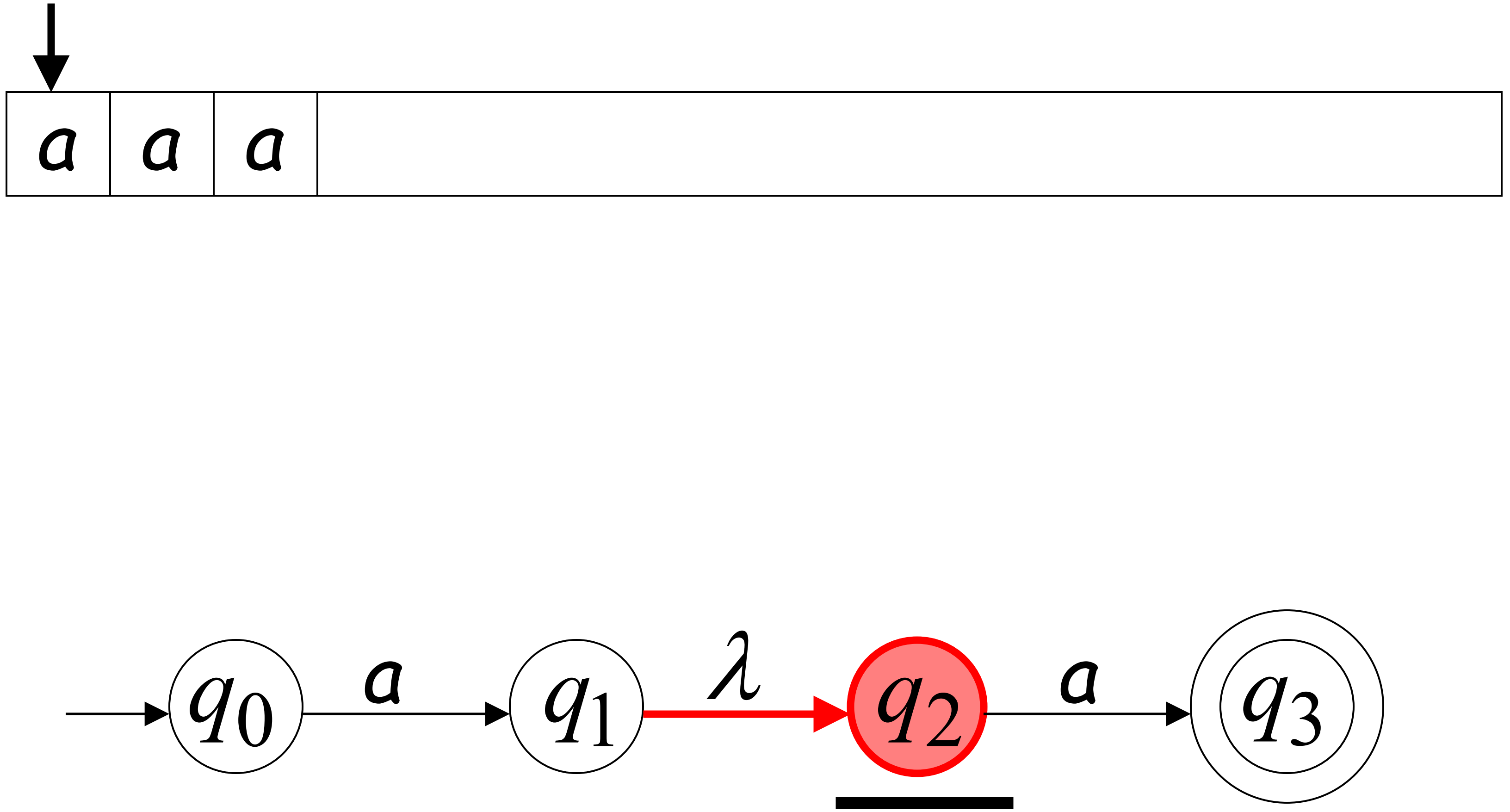
Esempio di non accettazione (rigettato)



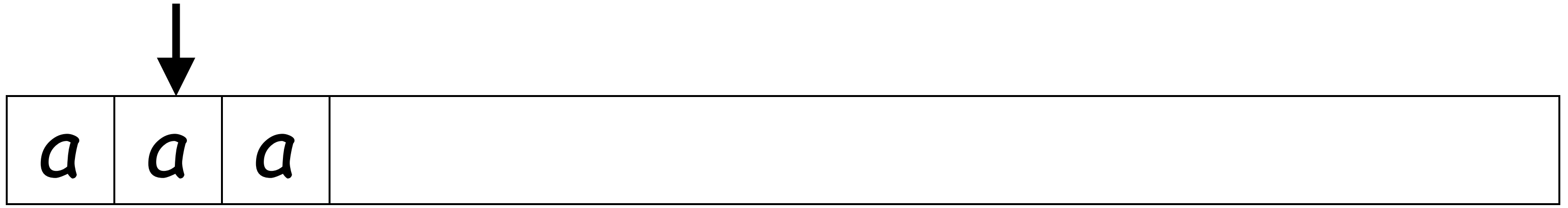

a a a


q_0
a
q_1
λ
q_2
a
q_3

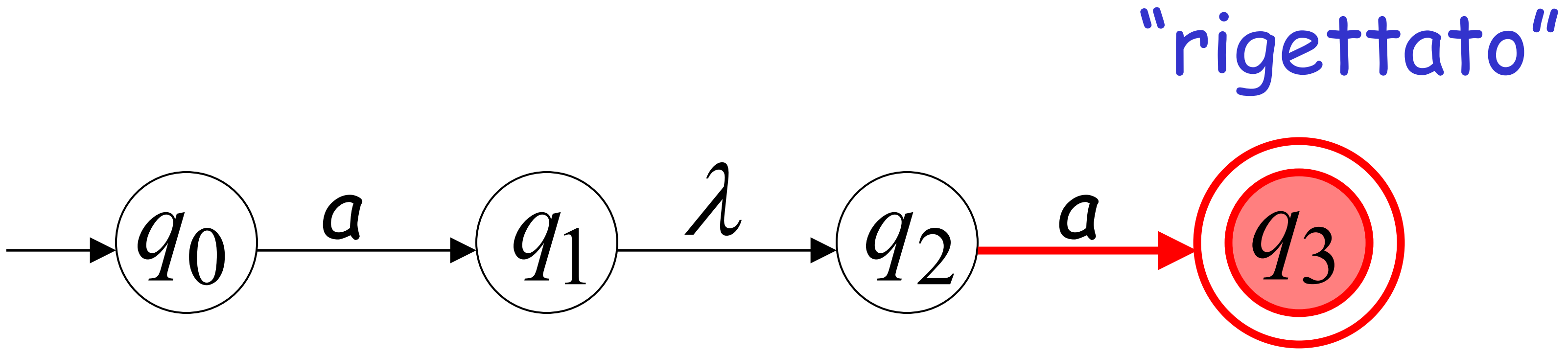
(la testina non si muove)



Input non viene analizzato tutto

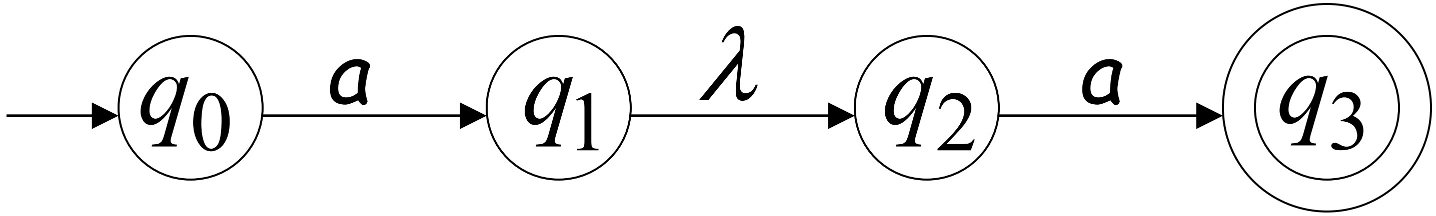


Automa si ferma



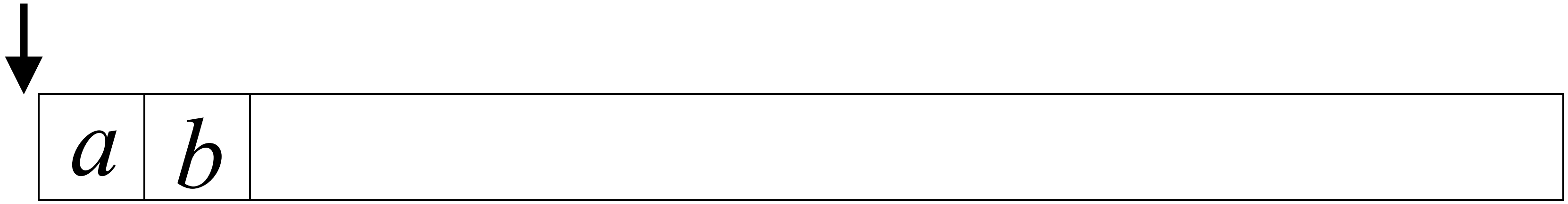
stringa aaa è rigettata

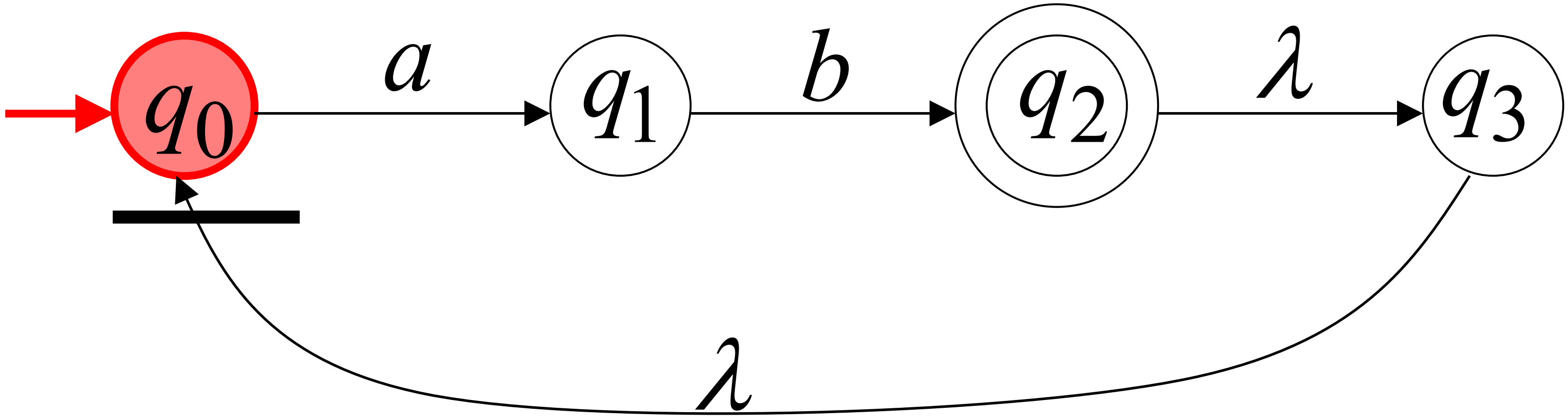
Linguaggio accettato: $L = \{aa\}$

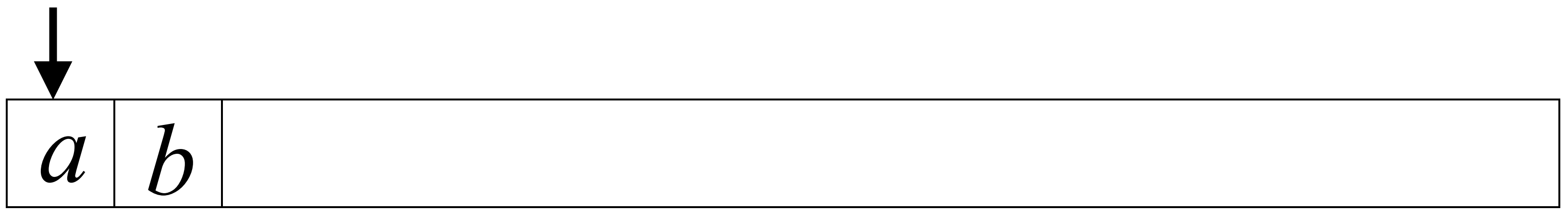


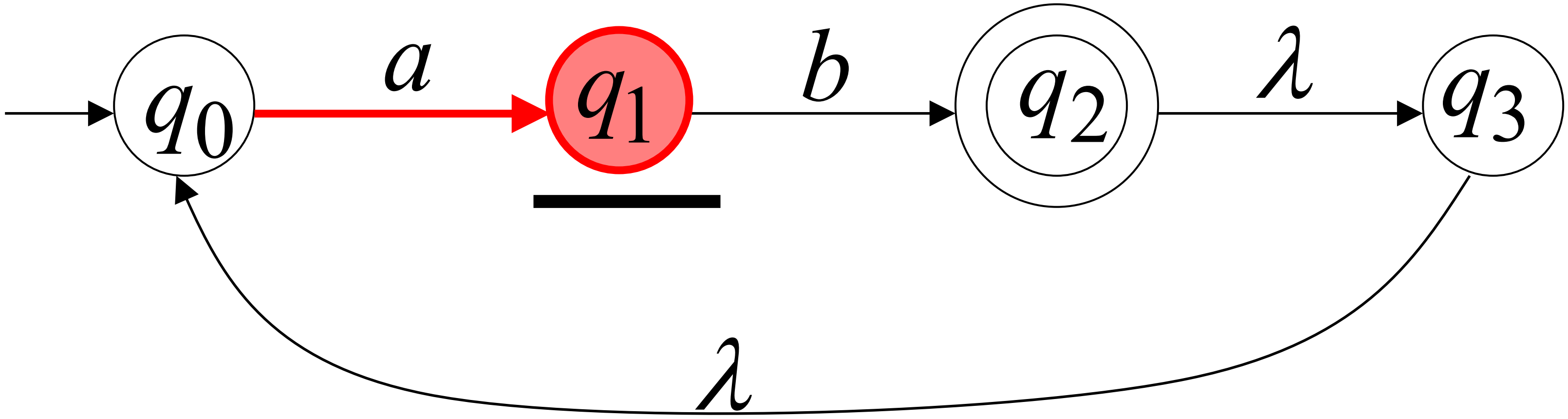
Esiste una computazione si
Per ogni computazione no

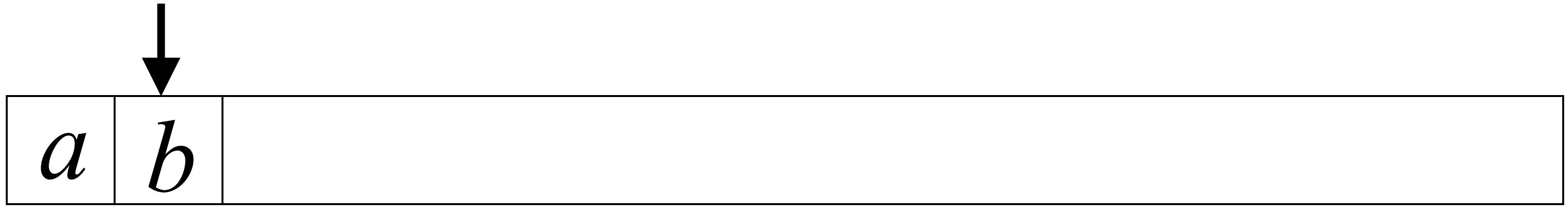
Un altro NFA

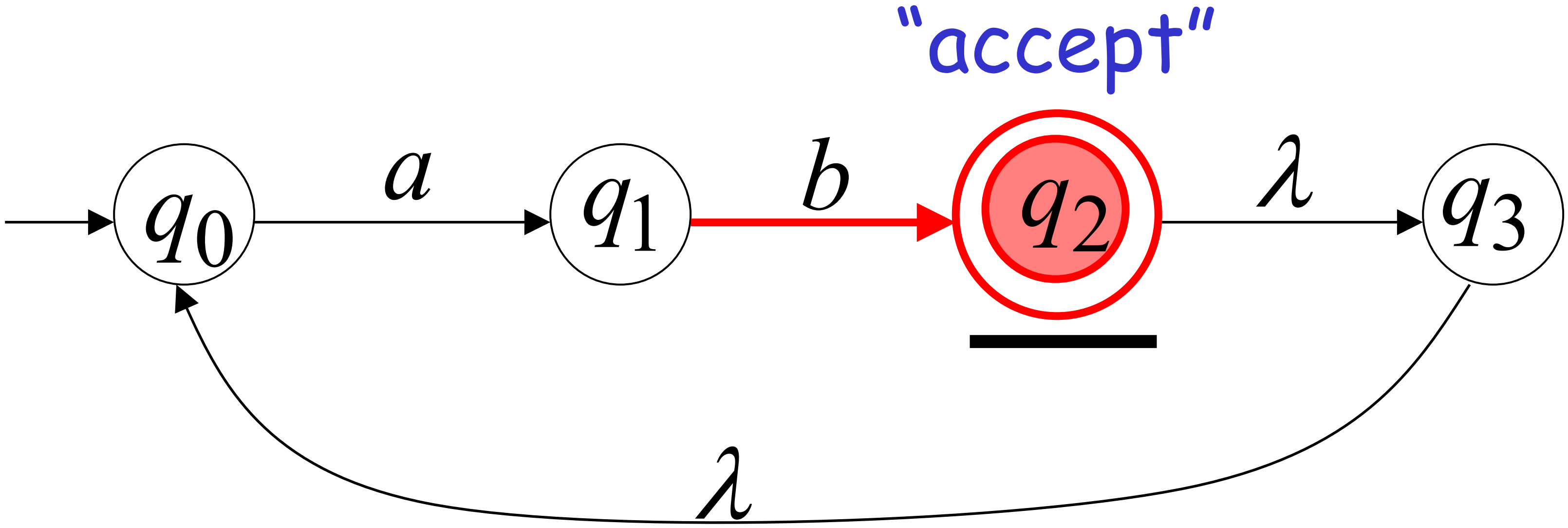




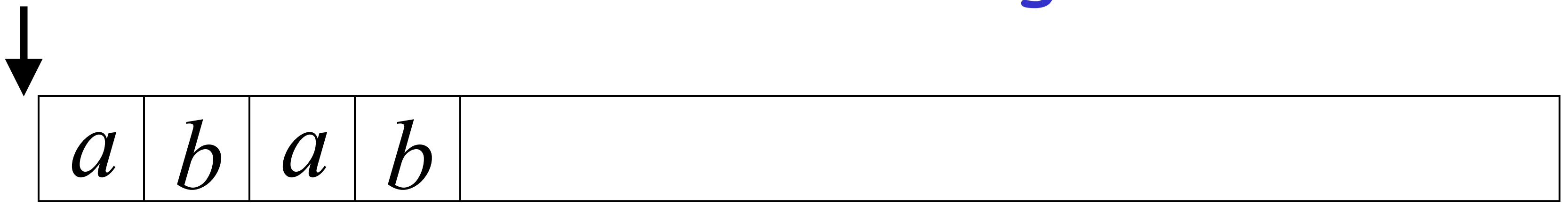

a
b

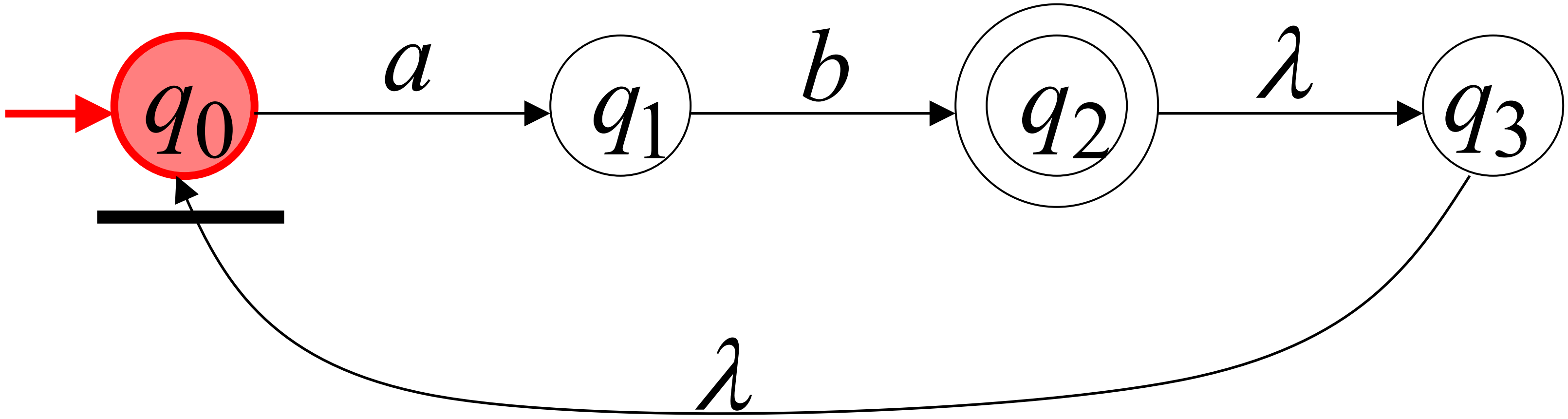

q_0
a
q_1
b
q_2
λ
q_3
λ

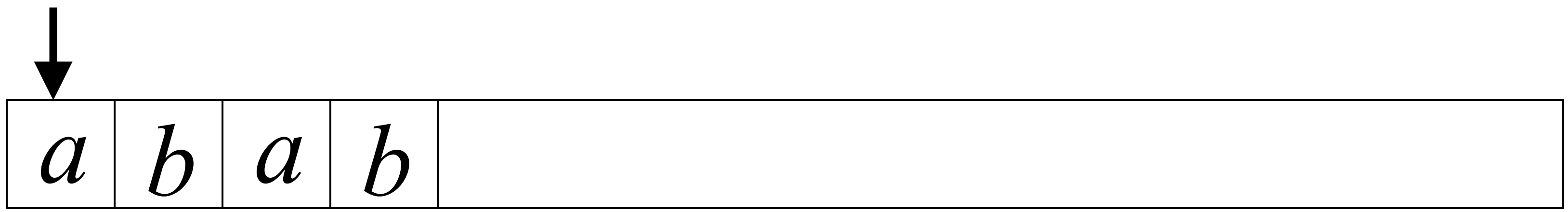

a
b

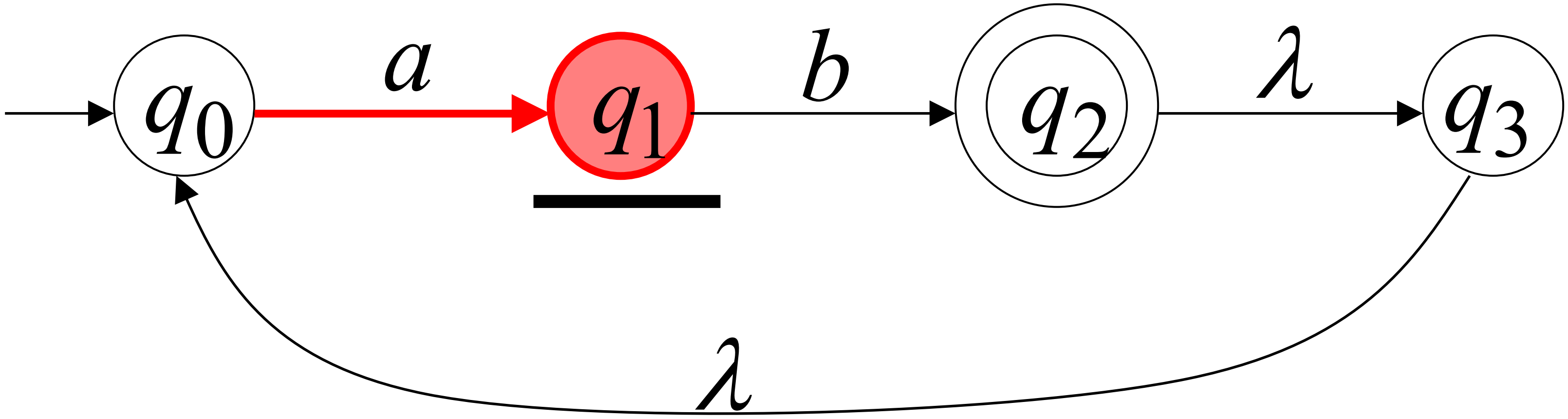

"accept"
a
b
λ
λ
q_0
q_1
q_2
q_3

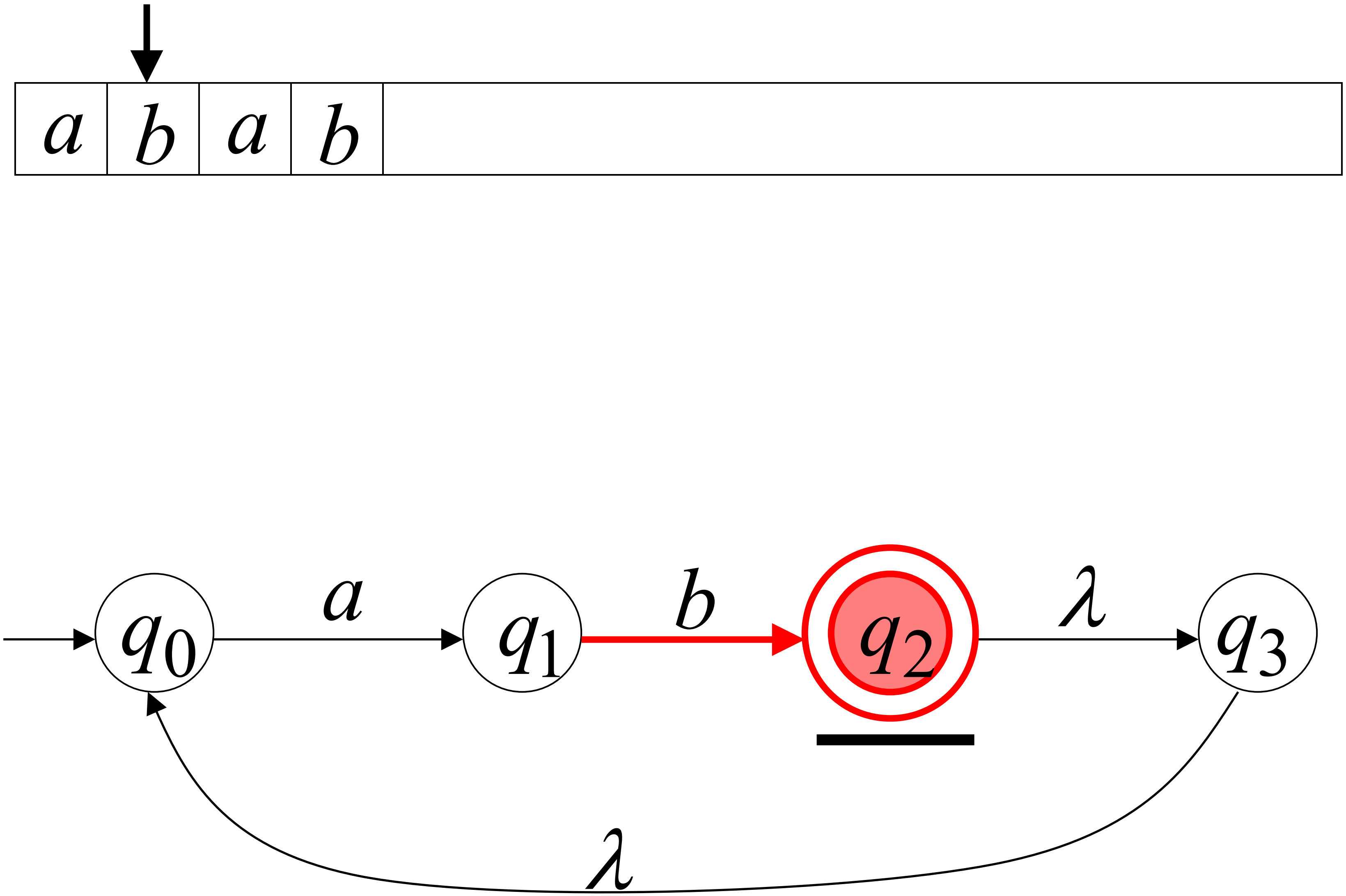
Un altra stringa



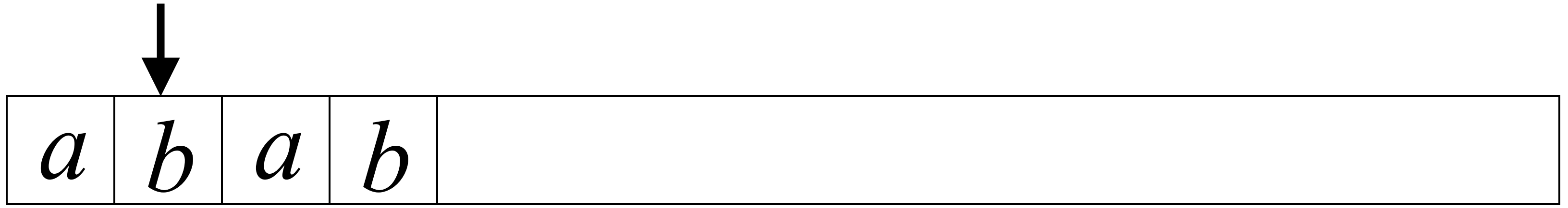


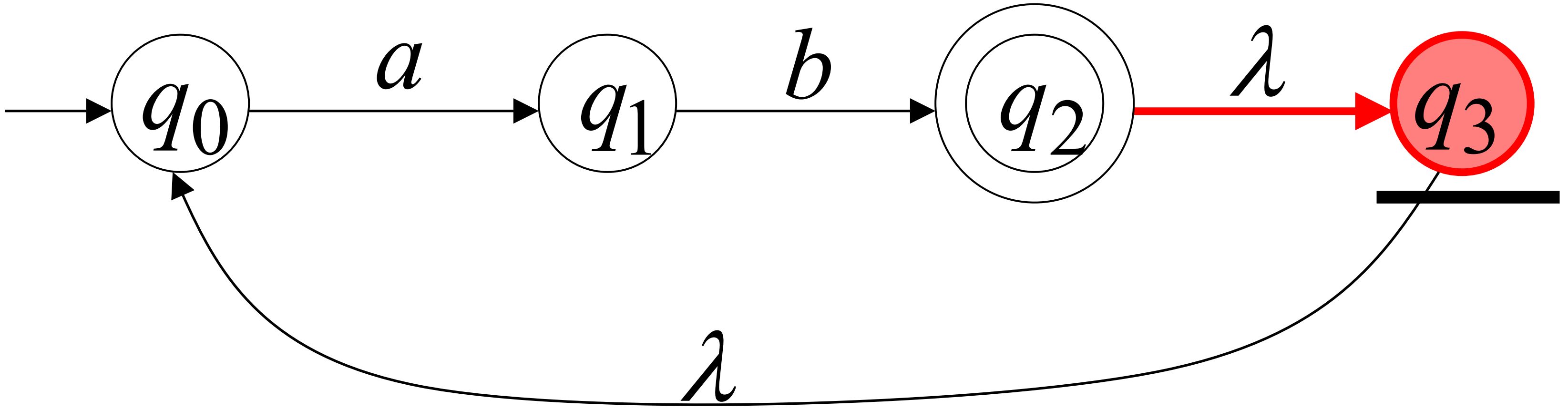

a
b
a
b

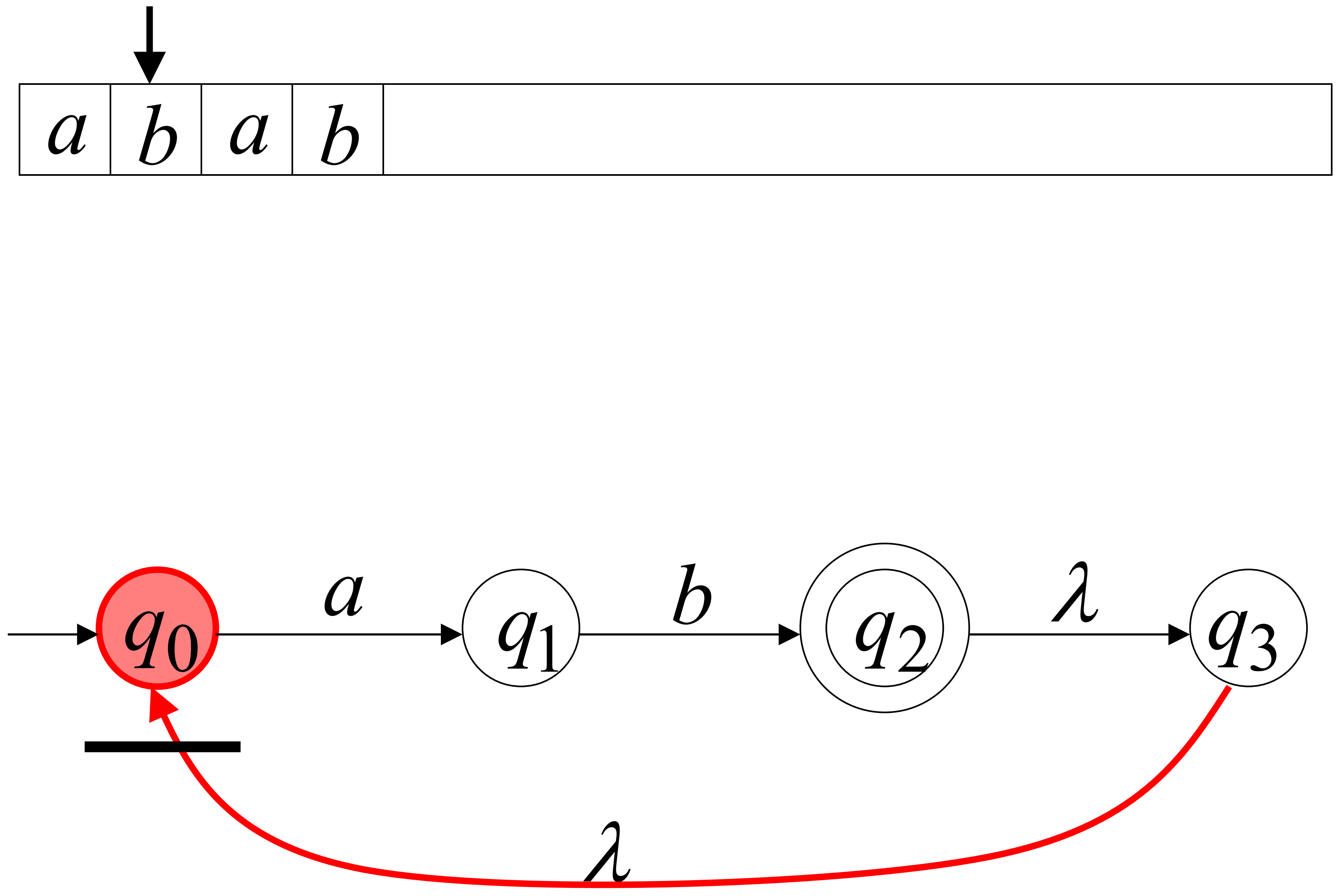

q_0
a
q_1
b
q_2
λ
q_3
λ

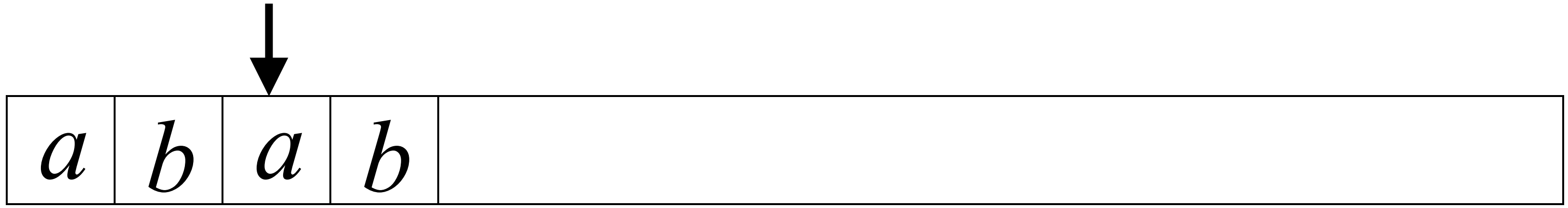


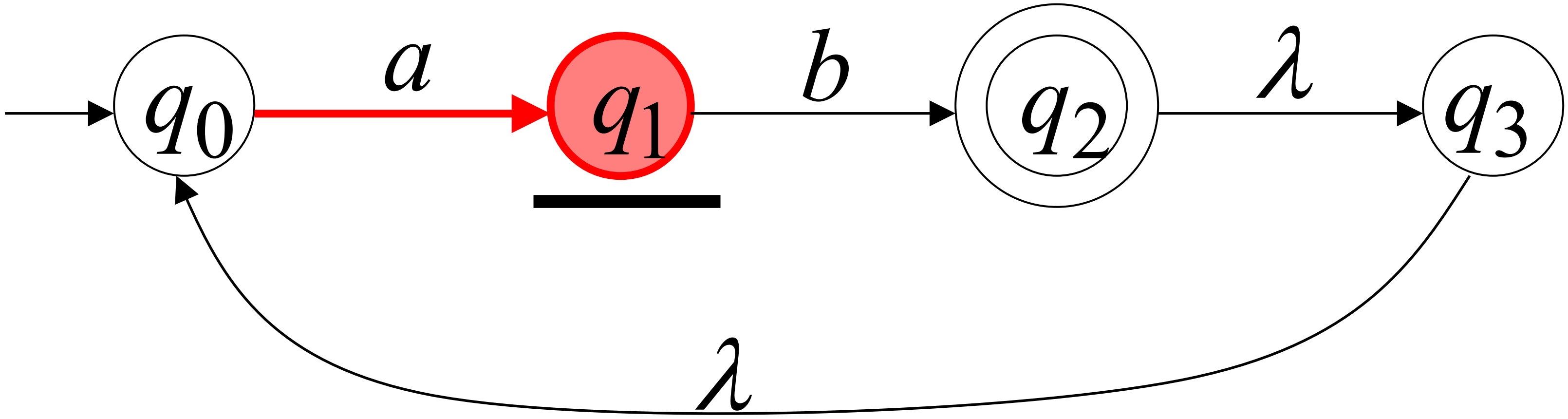
a
b
a
b
q_0
a
q_1
b
q_2
λ
q_3
λ

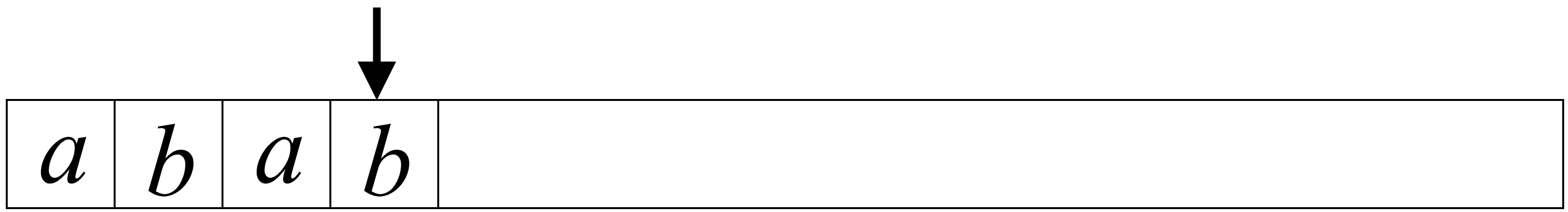

a
b
a
b

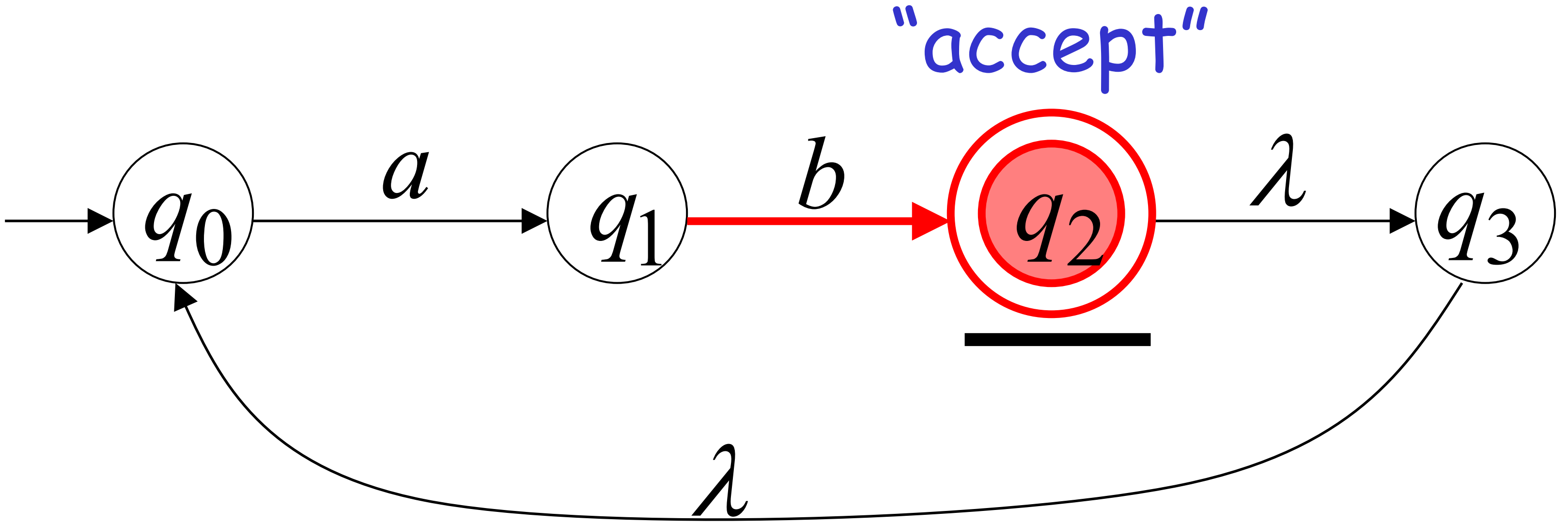

q_0
a
q_1
b
q_2
λ
q_3
λ


a
b
a
b
a
b
λ
λ
q0
q1
q2
q3


a
b
a
b


q0
a
q1
b
q2
λ
q3
λ

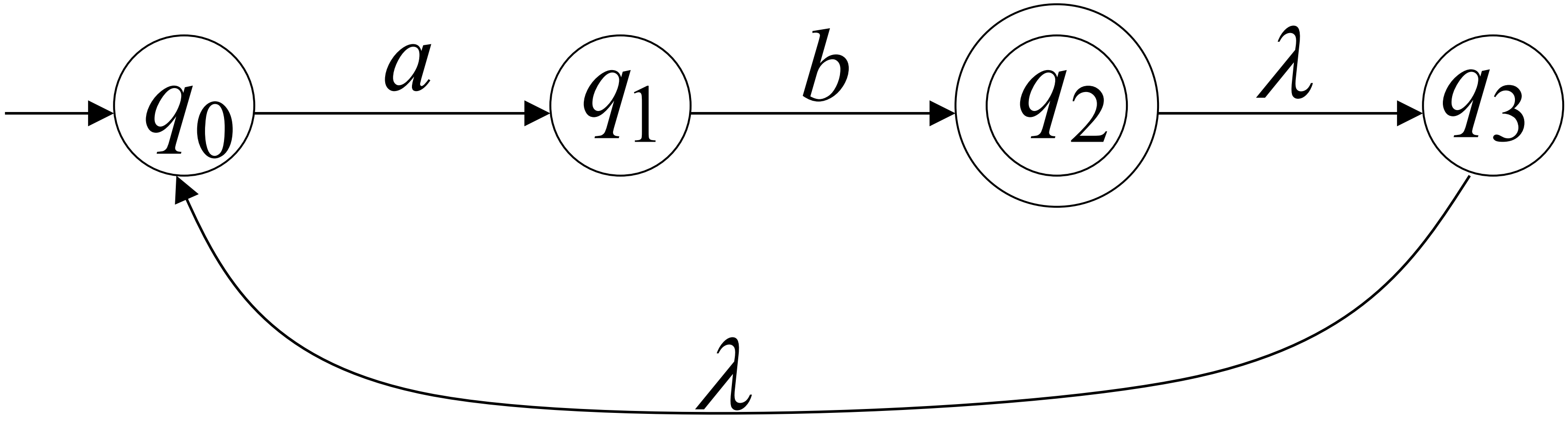

a
b
a
b


"accept"
q_0
a
q_1
b
q_2
λ
q_3
λ

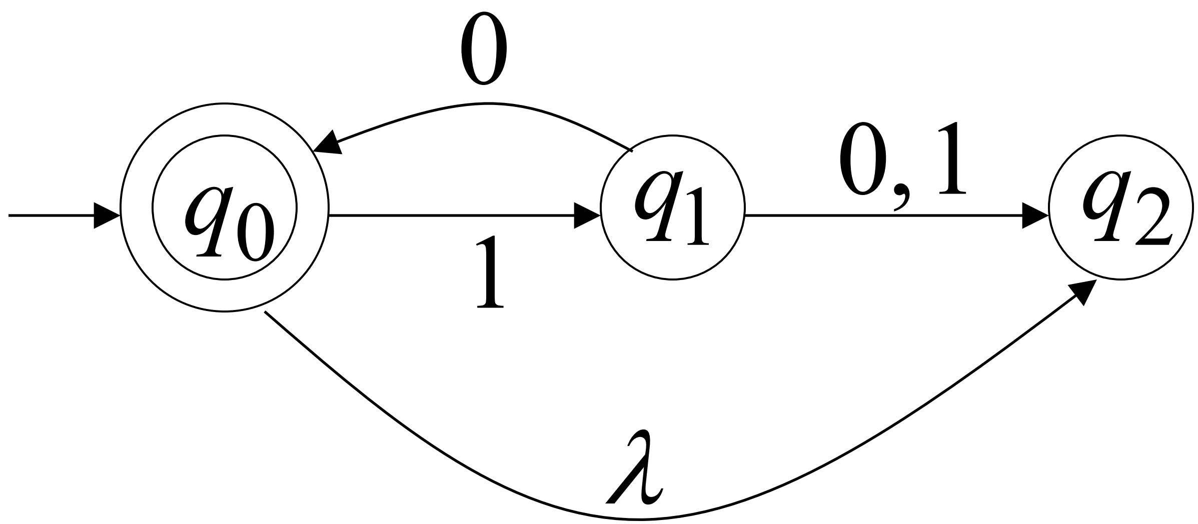
Linguaggio accettato

$$L = \{ab,\ abab,\ ababab,\ \ldots\}$$

$$= \{ab\}^{+}$$



NFA esempio

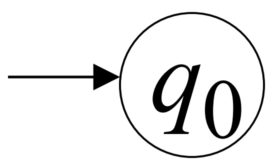


Remarks:

- Il simbolo λ non appare mai
- sul nastro di input

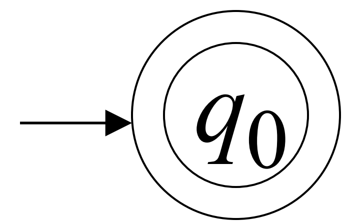
- Semplici automata:

M_1



$$L(M_1) = \{\}$$

M_2



$$L(M_2) = \{\lambda\}$$

Formal Definition of NFAs

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

$Q:$ Set of states, i.e. $\{q_0, q_1, q_2\}$

$\Sigma:$ Input aplhabet, i.e. $\{a, b\}$ $\qquad \lambda \notin \Sigma$

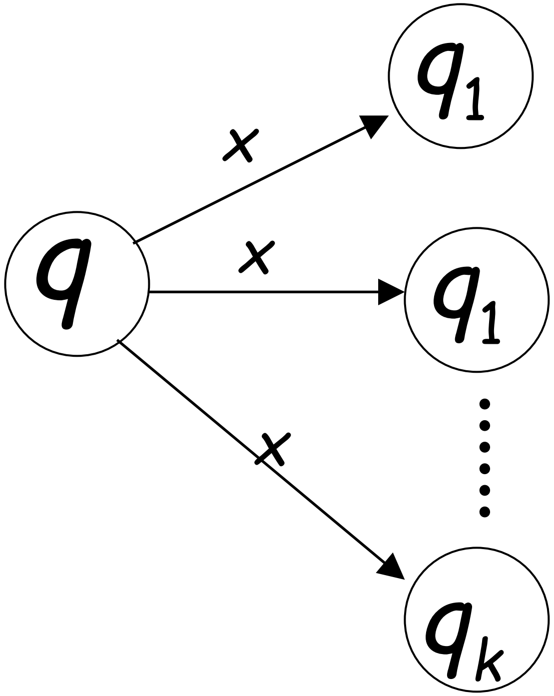
$\delta:$ Transition function

$q_0:$ Initial state

$F:$ Accepting states

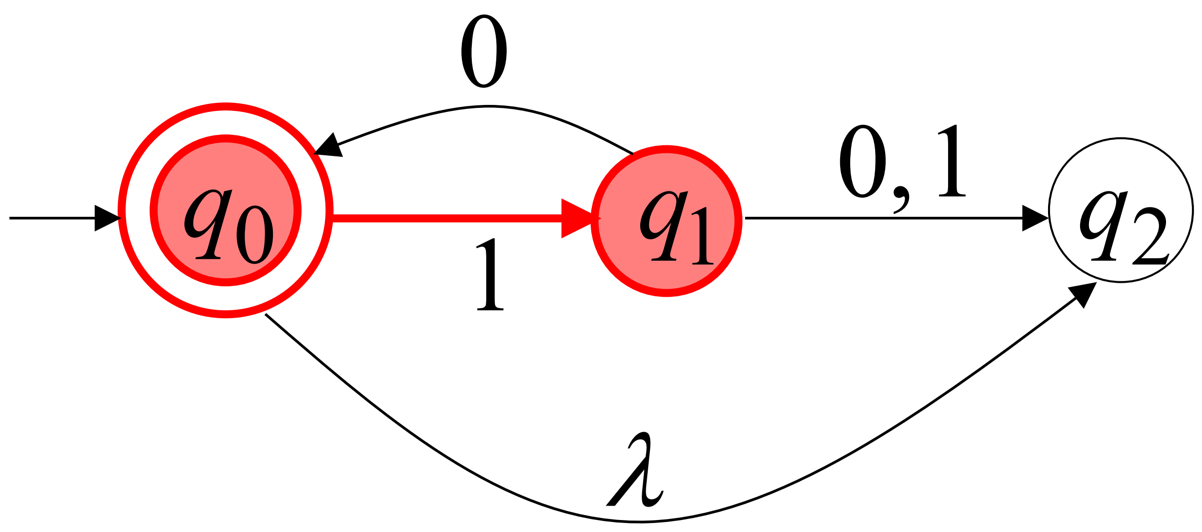
Funzione di transizione δ

$$\delta(q, x) = \{q_1, q_2, \ldots, q_k\}$$

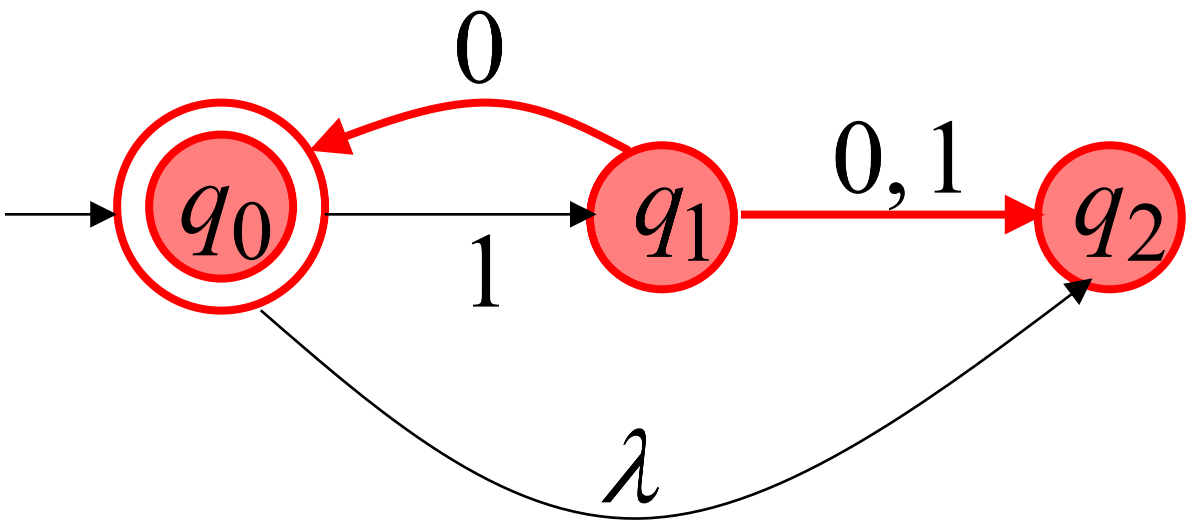


Stati risultanti con una transizione con simbolo x

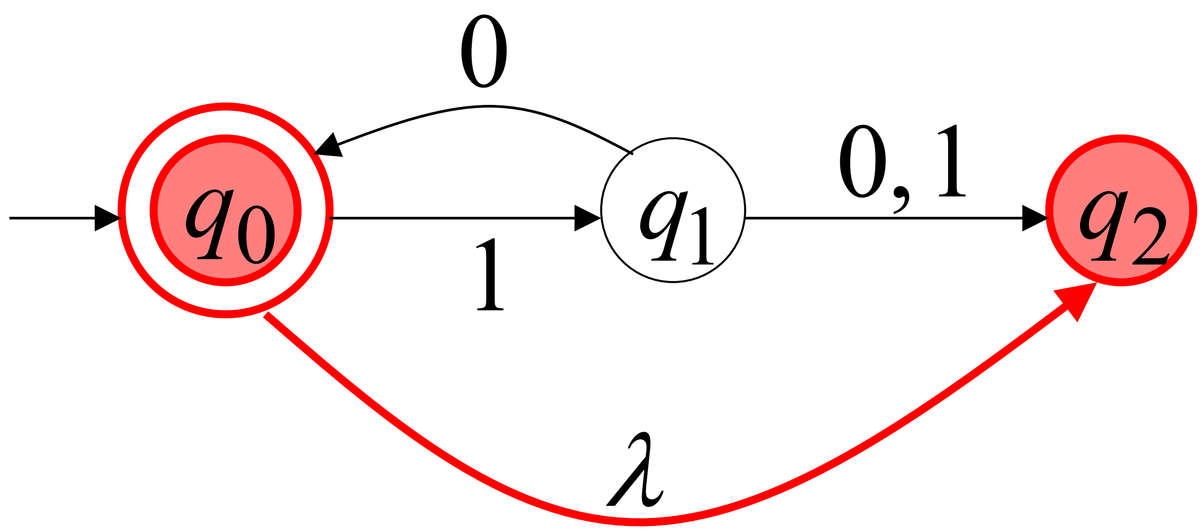
$$\delta(q_0, 1) = \{q_1\}$$



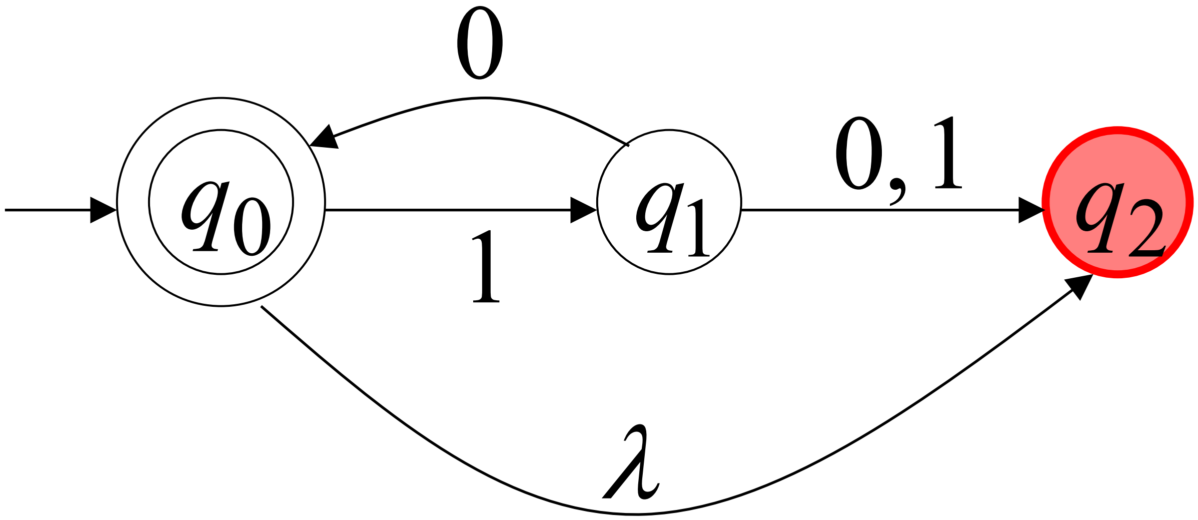
$$\delta(q_1, 0) = \{q_0, q_2\}$$



$$\delta(q_0, \lambda) = \{q_2\}$$



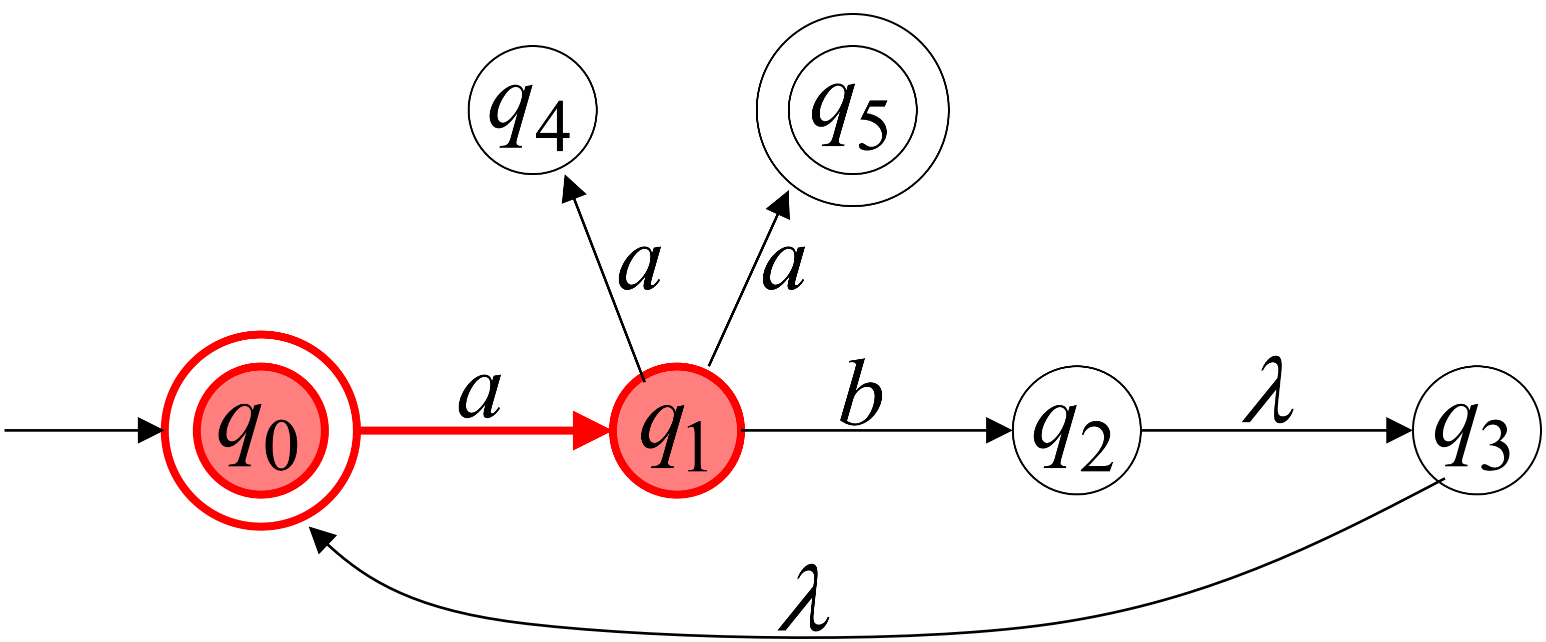
$$\delta(q_2, 1) = \varnothing$$



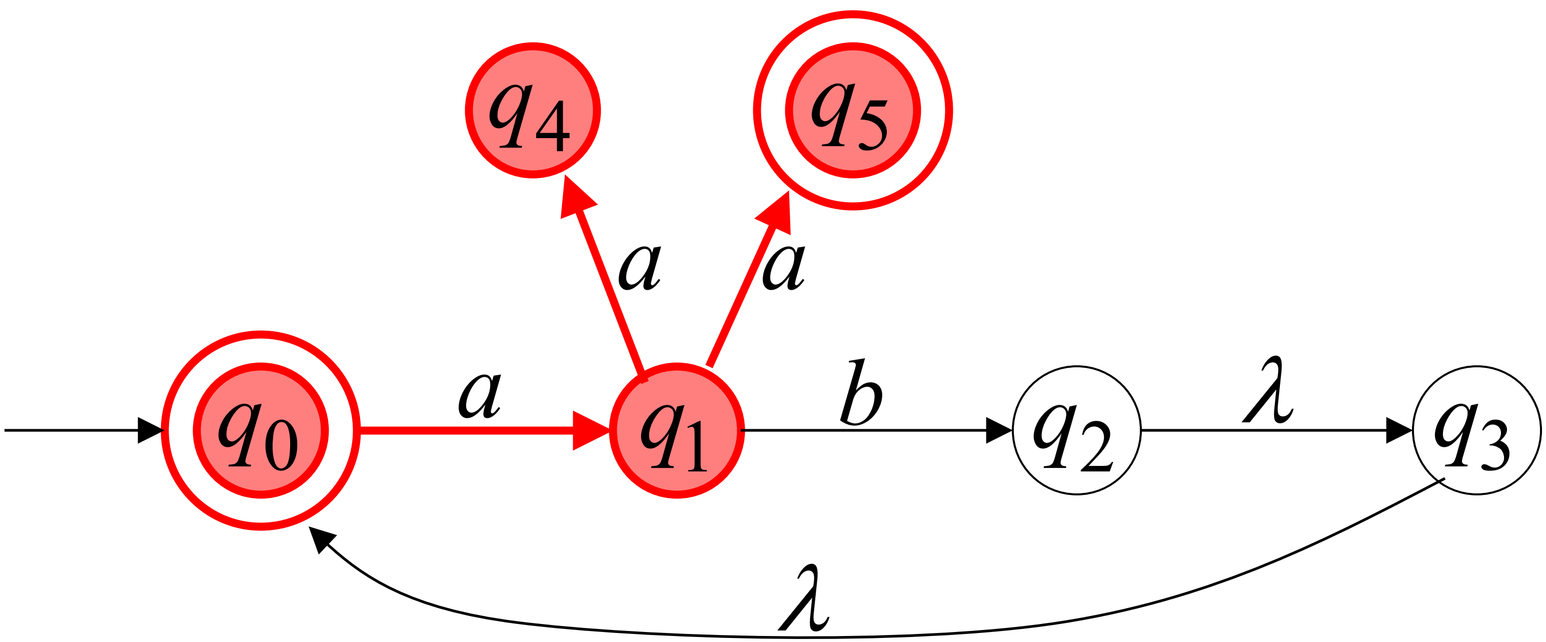
Funzione di transizione estesa δ^*

La stessa cosa δ ma applicata a stringhe

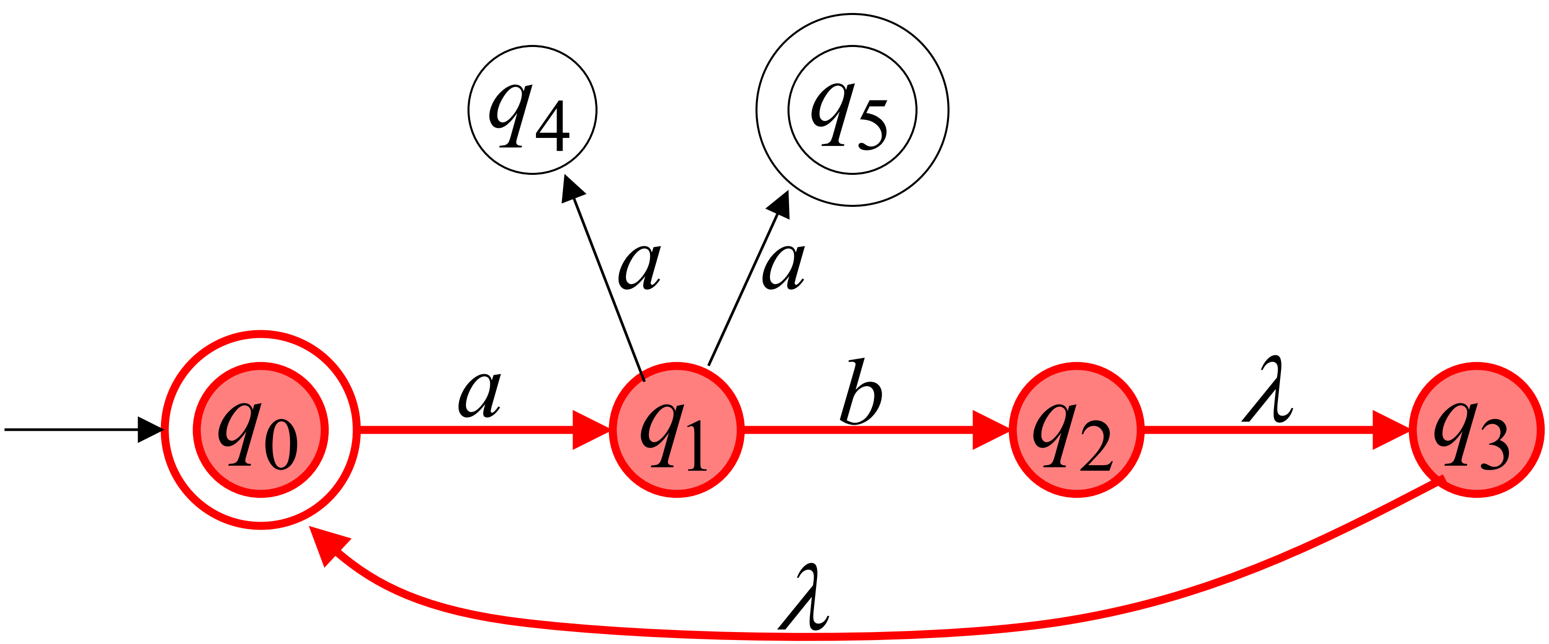
$$\delta^*(q_0, a) = \{q_1\}$$



$$\delta^*(q_0, aa) = \{q_4, q_5\}$$

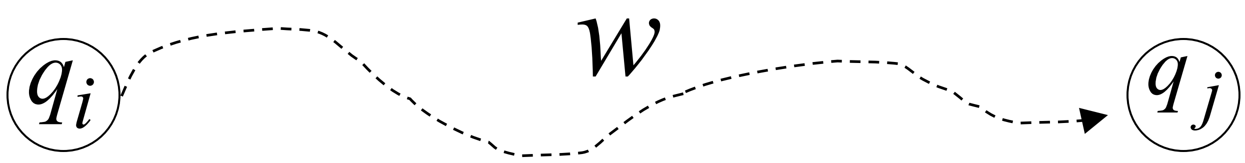


$$\delta^*(q_0, ab) = \{q_2, q_3, q_0\}$$

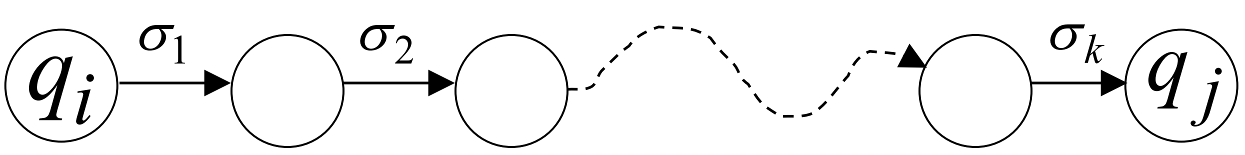


In generale

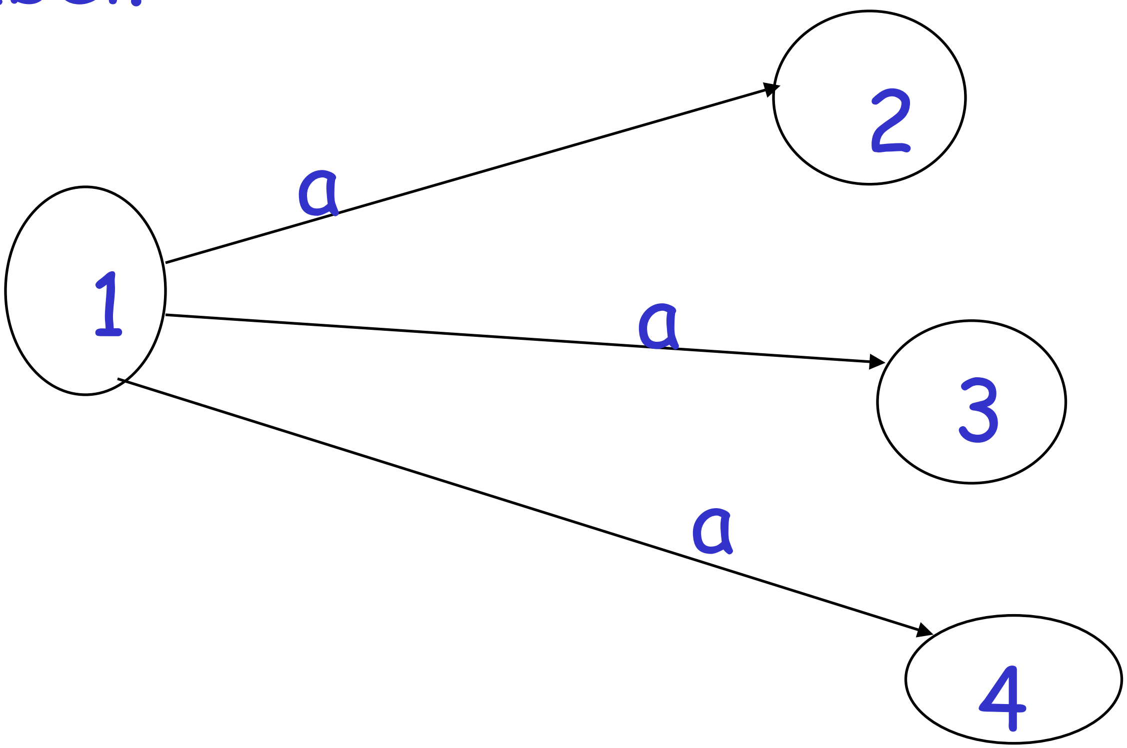
$q_j \in \delta^*(q_i, w)$: vi è un cammino da q_i a q_j con label w



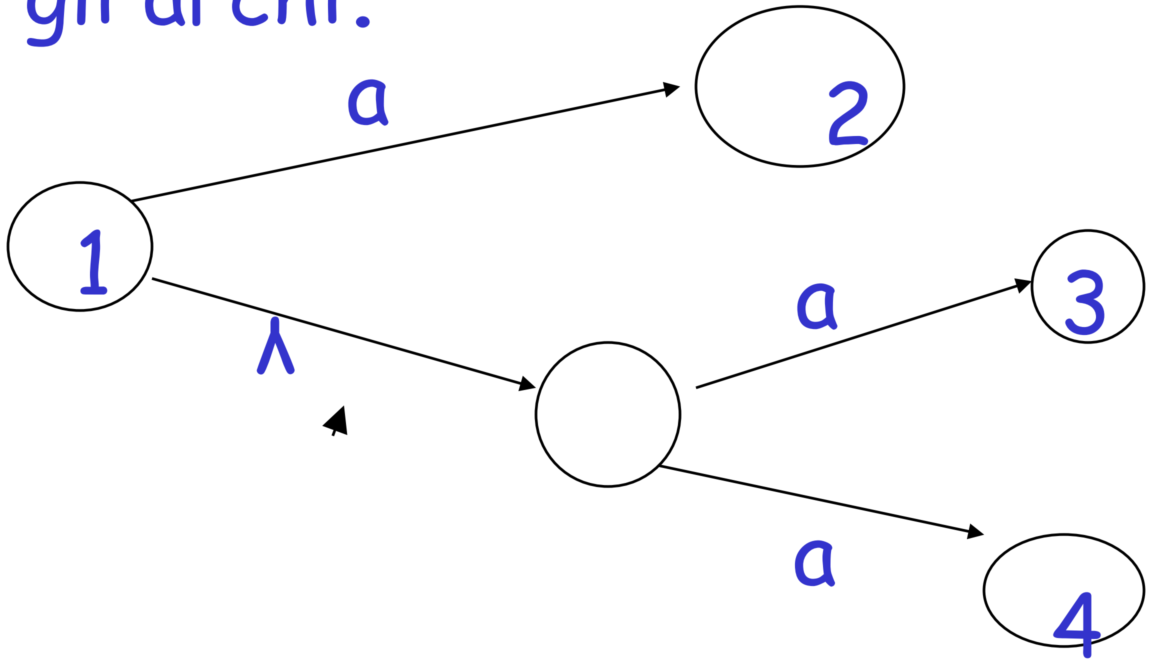
$$w = \sigma_1 \sigma_2 \cdots \sigma_k$$



Grado di non determinismo di un nodo per ogni nodo il numero di archi con la stessa label.



Grado di non determinismo di un automa, il grado massimo di non determinismo di tutti gli archi.



The Language of an NFA M

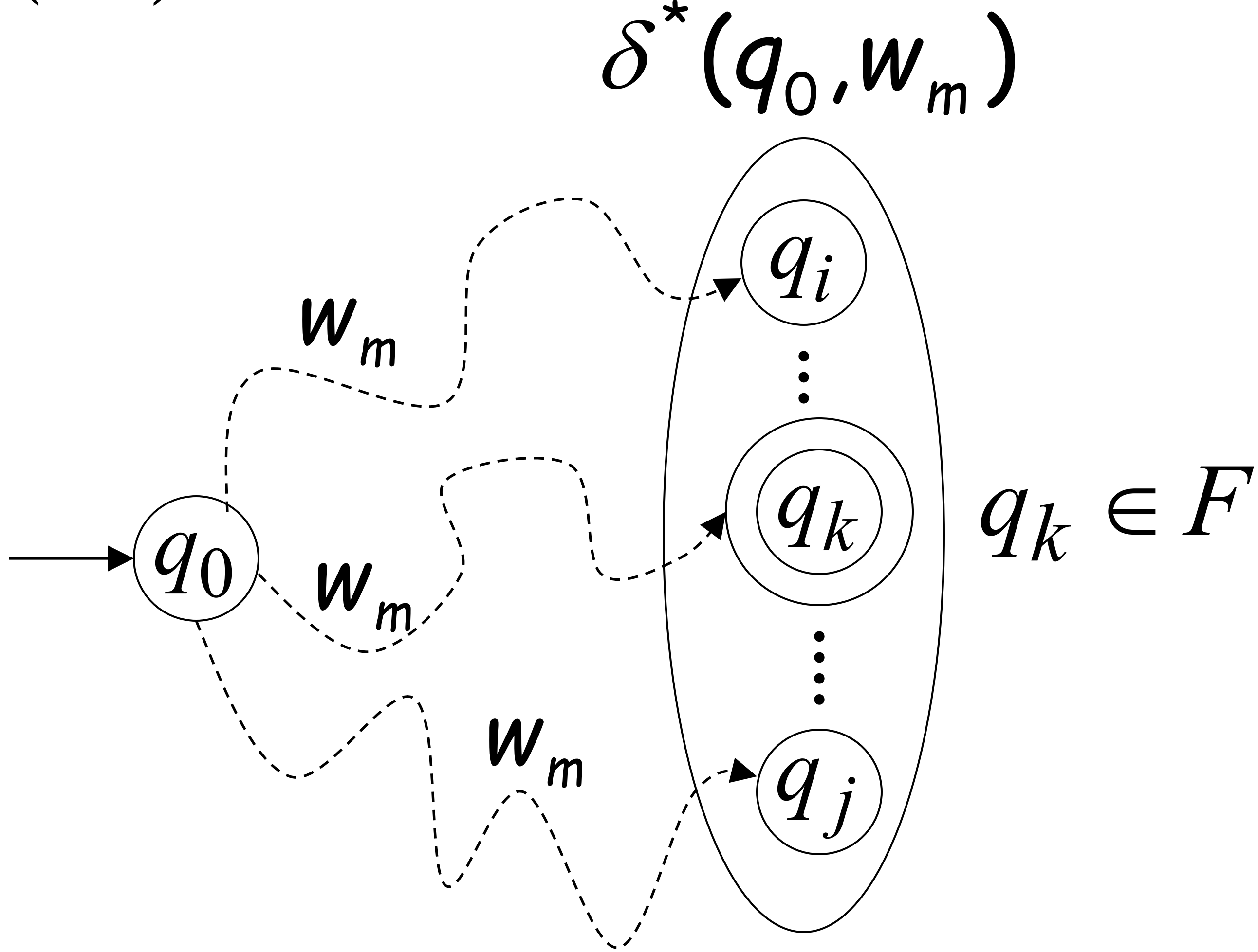
Il linguaggio accettato da M è:

$$L(M) = \{w_1, w_2, \ldots w_n\}$$

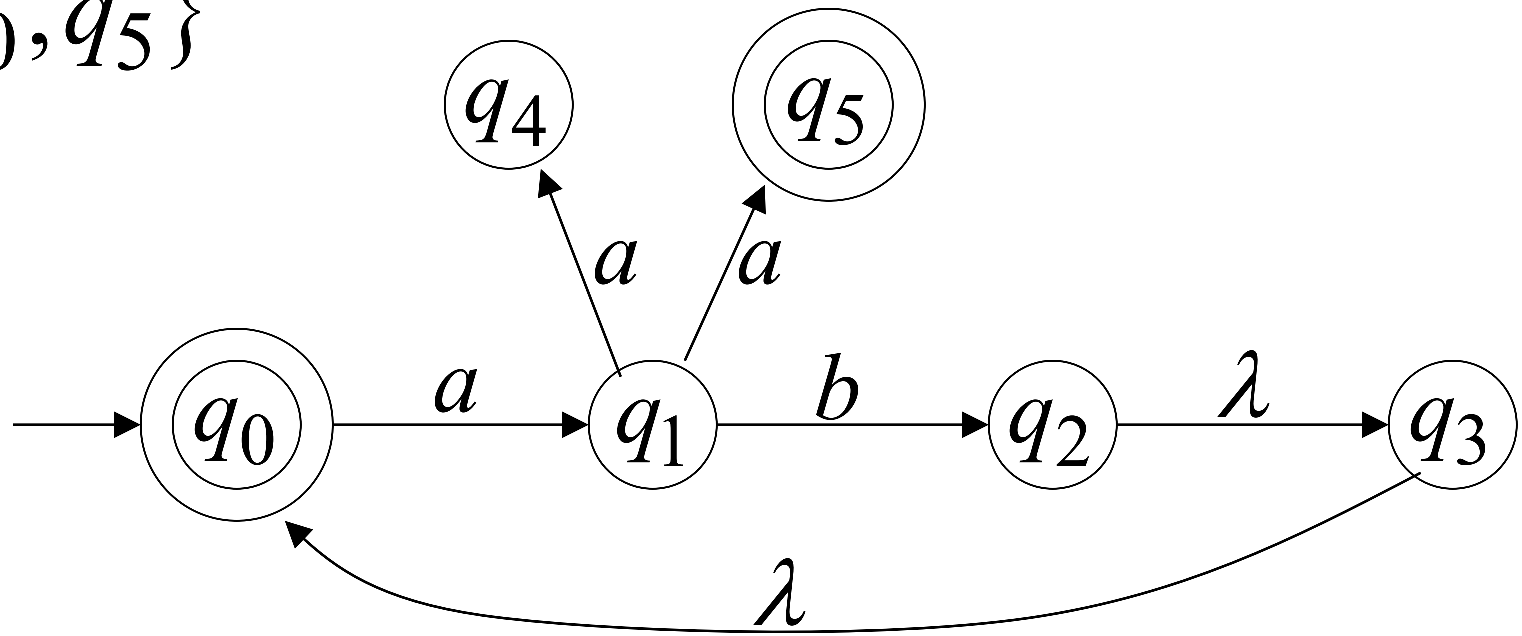
dove $\delta^*(q_0, w_m) = \{q_i, \ldots, q_k, \ldots, q_j\}$

E vi è un $q_k \in F$ (stato finale)

$w_m \in L(M)$



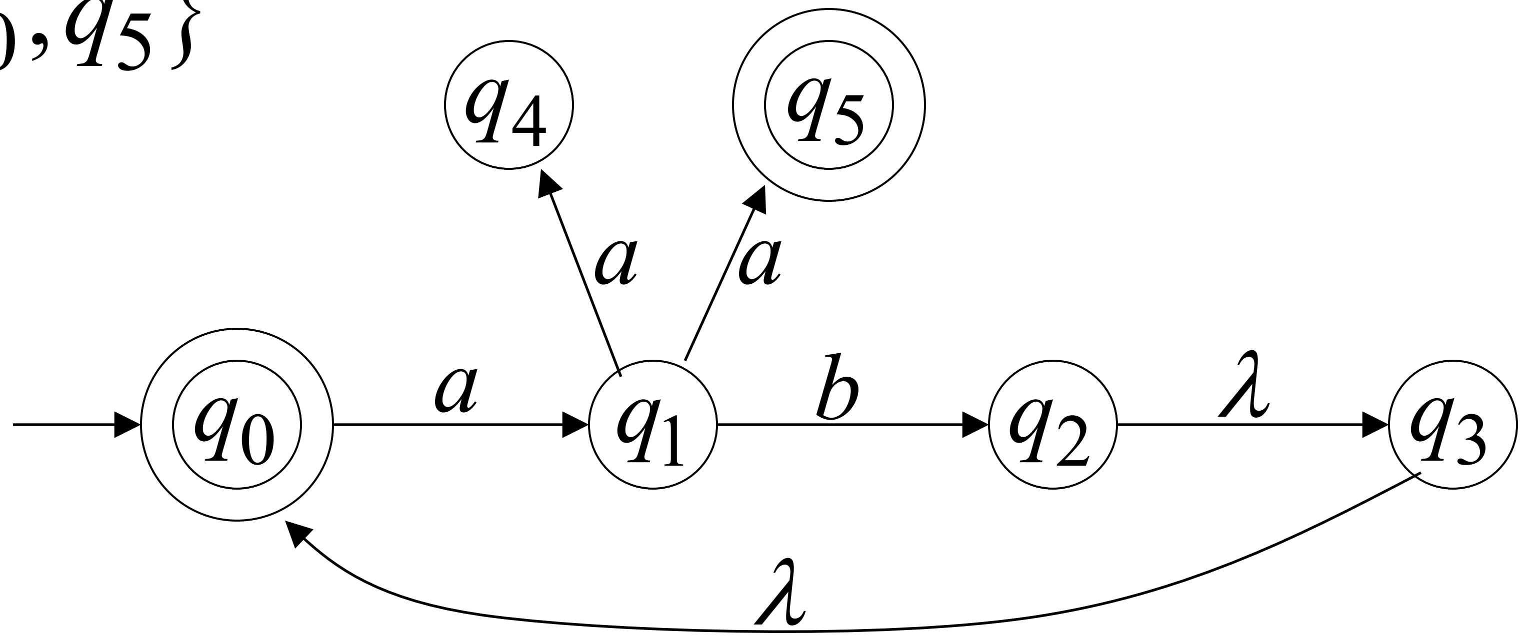
$$F = \{q_0, q_5\}$$



$$\delta^*(q_0, aa) = \{q_4, \underline{q_5}\} \Longrightarrow aa \in L(M)$$

$$q_5 \in F$$

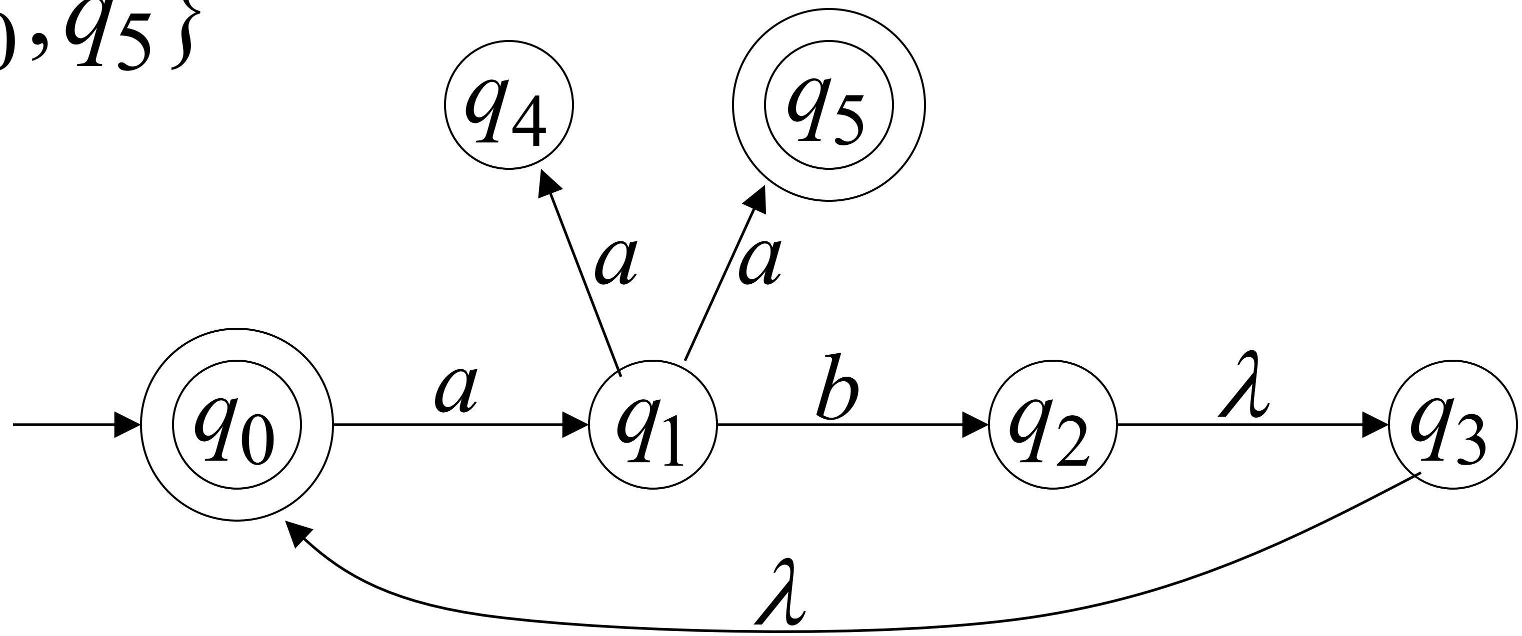
$$F = \{q_0, q_5\}$$



$$\delta^*(q_0, ab) = \{q_2, q_3, \underline{q_0}\} \Longrightarrow ab \in L(M)$$

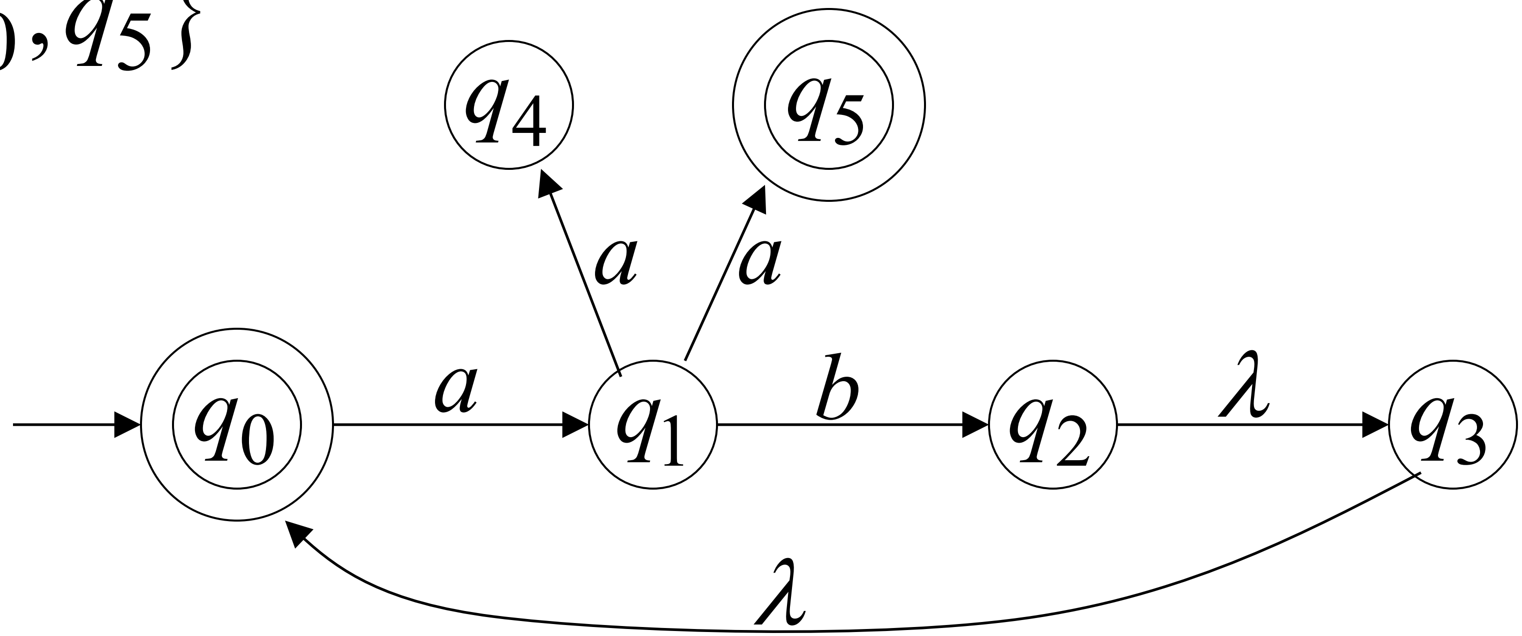
$$\underline{q_0} \searrow \in F$$

$$F = \{q_0, q_5\}$$



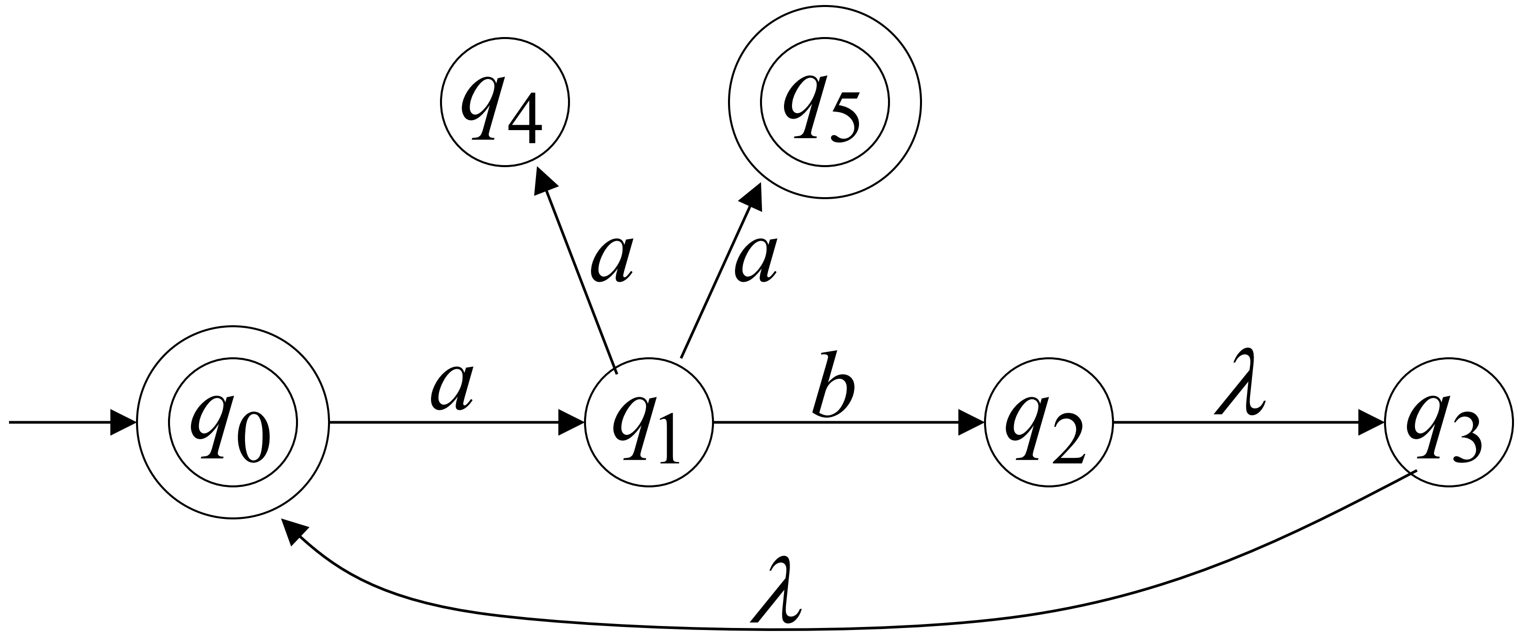
$$\delta^*(q_0, abaa) = \{q_4, \underbrace{q_5}_{\in F}\} \Longrightarrow aaba \in L(M)$$

$F = \{q_0, q_5\}$



$\delta^*(q_0, aba) = \{q_1\}$ $\Longrightarrow$ $aba \notin L(M)$

$\notin F$



$$L(M) = \{ab\}^* \cup \{ab\}^* \{aa\}$$

δ*(stato, cW)=

{

delta_*(q,W)

con q elemento dell'insieme {delta(stato, c)}

}

$$q \in \delta^{*}(q, \lambda)$$ Per ogni stato

1 $\delta^*(stato, cW)=$

{

$\delta^*(q,W)$

con $q \in \{\delta(stato, c)\}$

}

2 $q \in \delta^*(q, \lambda)$ Per ogni stato

NFA accettano i linguaggi regolari

Equivalenza tra macchine

Definizione:

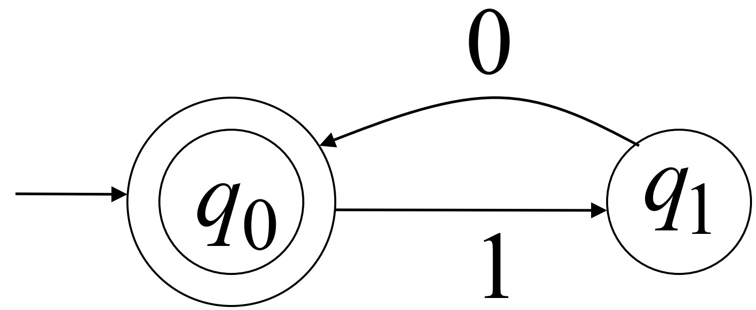
macchina M_1 è equivalente alla macchina M_2

se $L(M_1) = L(M_2)$

Esempio di macchine equivalenti

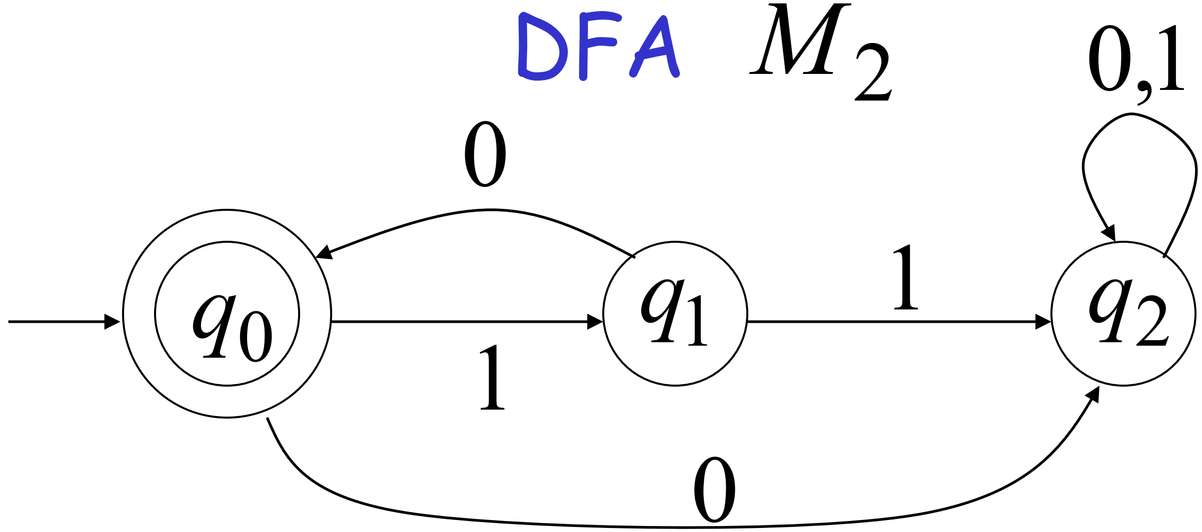
NFA M_1

$$L(M_1) = \{10\}^*$$



DFA M_2

$$L(M_2) = \{10\}^*$$



Teorema:

$$\left\{ \begin{array}{l} \text{Linguaggi} \\ \text{Accettati} \\ \text{da NFA} \end{array} \right\} = \left\{ \begin{array}{l} \text{Linguaggi} \\ \text{regolari} \end{array} \right\}$$

Linguaggi
Accettati da un
DFA

NFA e DFA hanno lo stesso potere di computazione,
Accettano gli stessi inguaggi.

dimostrazione: mostreremo

$$\left\{ \begin{array}{l} \text{Linguaggi a} \\ \text{Accettati} \\ \text{da NFA} \end{array} \right\} \supseteq \left\{ \begin{array}{l} \text{Linguaggi} \\ \text{regolari} \end{array} \right\}$$

AND

$$\left\{ \begin{array}{l} \text{Linguaggi a} \\ \text{Accettati} \\ \text{da NFA} \end{array} \right\} \subseteq \left\{ \begin{array}{l} \text{Linguaggi} \\ \text{regolari} \end{array} \right\}$$

Parte prima

$$\left\{ \begin{array}{l} \text{Linguaggi a} \\ \text{Accettati} \\ \text{da NFA} \end{array} \right\} \supseteq \left\{ \begin{array}{l} \text{Linguaggi} \\ \text{regolari} \end{array} \right\}$$

ogni DFA è banalmente un NFA

$\Downarrow$

Ogni linguaggio L accettato da un DFA
È anche accettato da un NFA

Parte seconda

$$\left\{ \begin{array}{l} \text{Linguaggi a} \\ \text{Accettati} \\ \text{da NFA} \end{array} \right\} \subseteq \left\{ \begin{array}{l} \text{Linguaggi} \\ \text{regolari} \end{array} \right\}$$

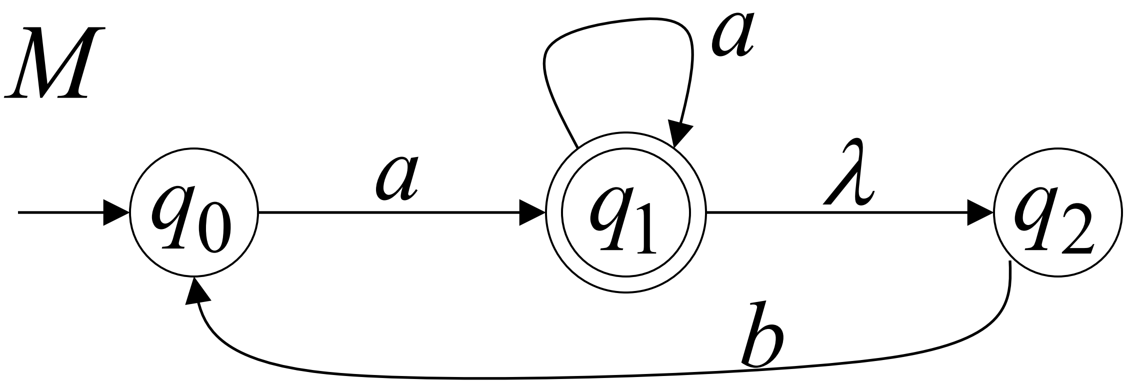
Ogni nfa può essere tradotto in un nfa

$\Downarrow$

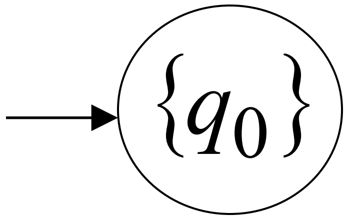
Ogni linguaggio L accettato da un NFA
È anche accettato da un DFA

Conversione da NFA a DFA

NFA M

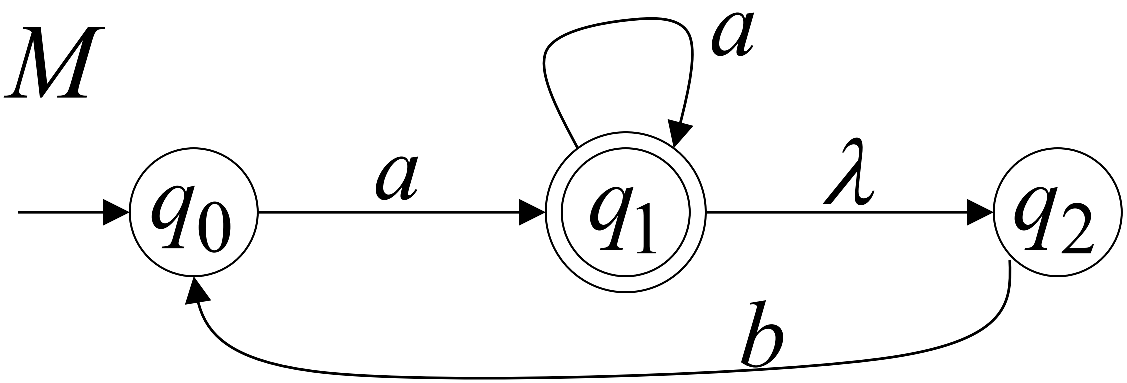


DFA M'

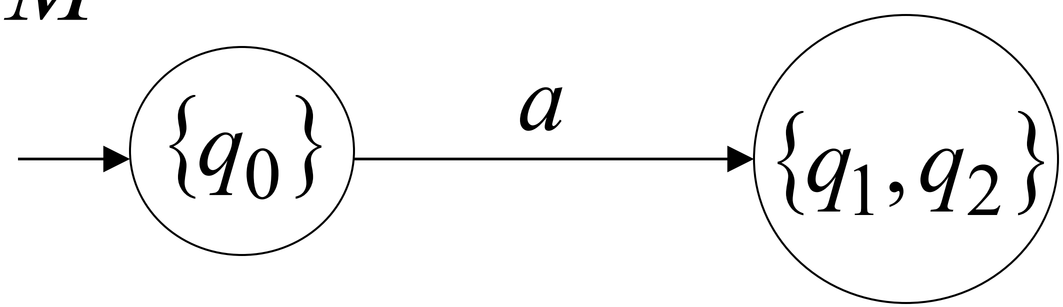


$$\delta^*(q_0, a) = \{q_1, q_2\}$$

NFA M

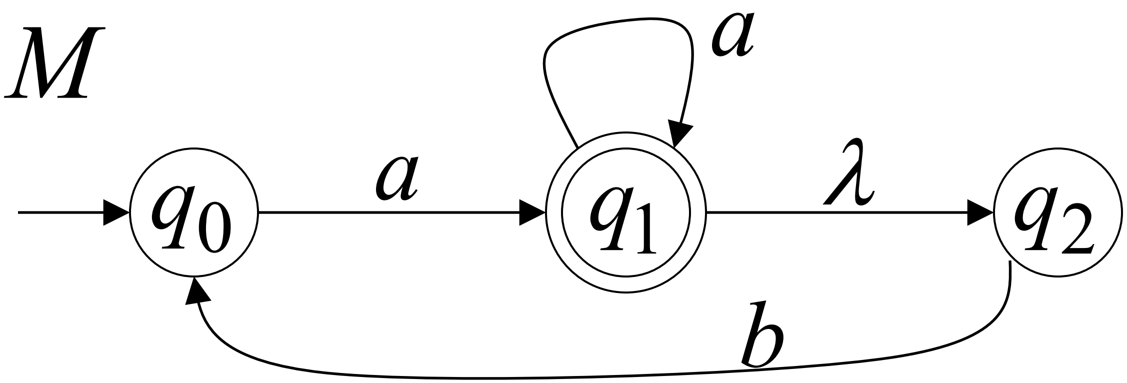


DFA M'

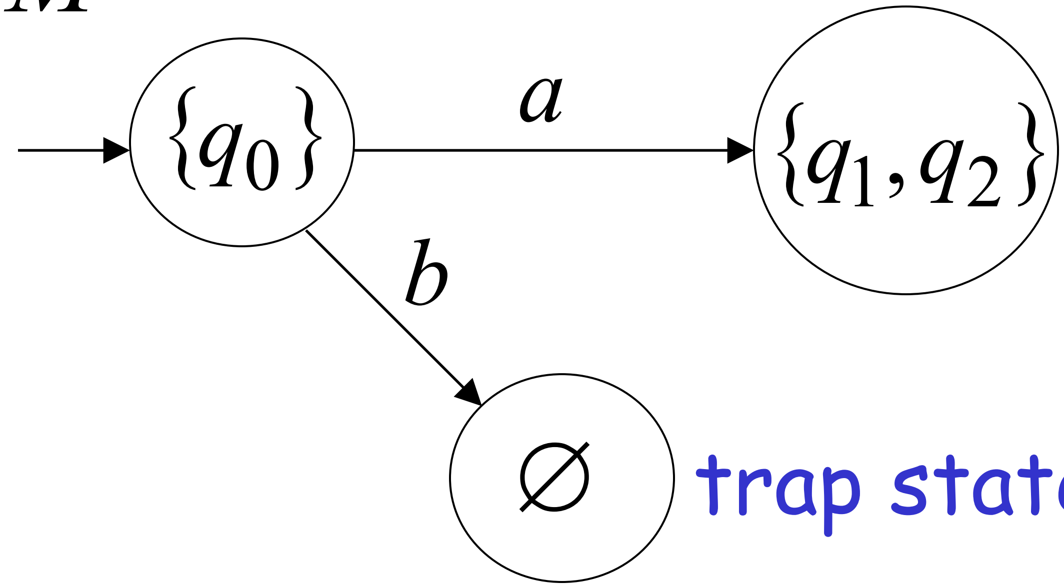


$$\delta^*(q_0, b) = \emptyset \quad \text{Insieme vuoto}$$

NFA M



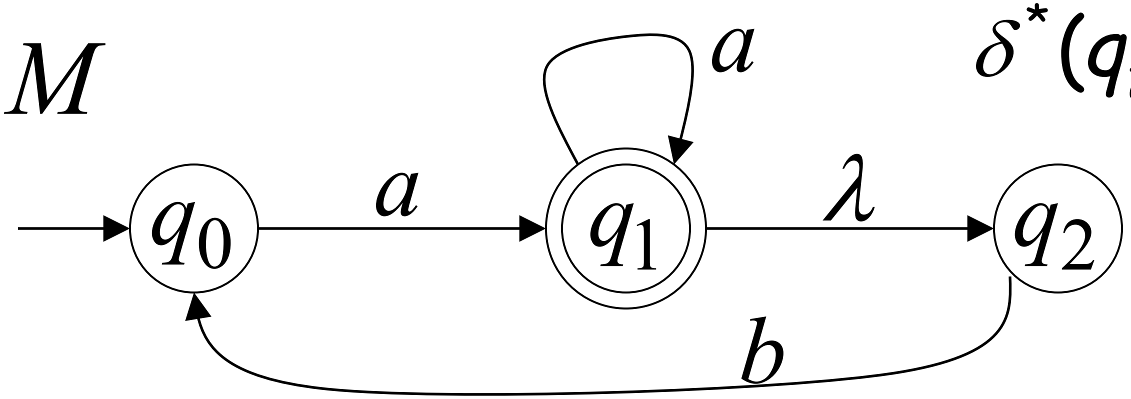
DFA M'

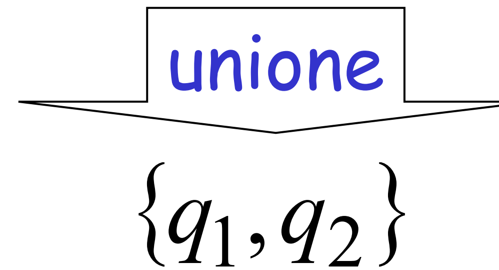


trap state

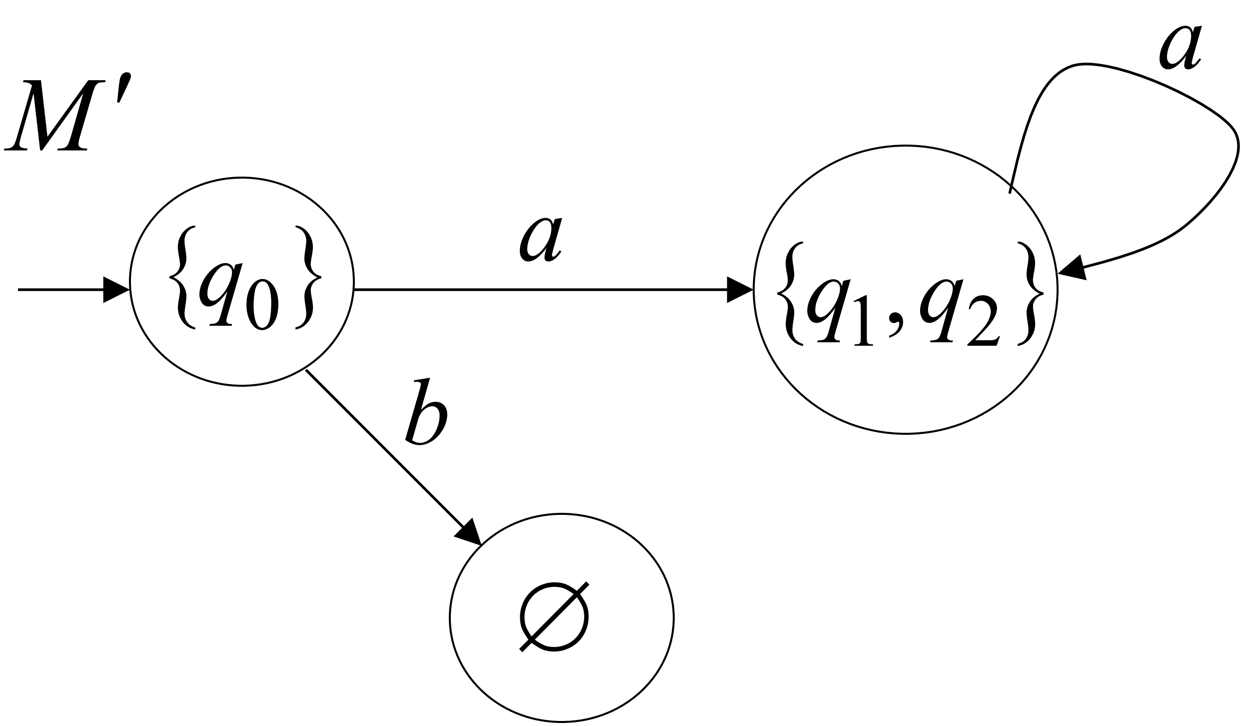
NFA M

$$\delta^*(q_1, a) = \{q_1, q_2\}$$
$$\delta^*(q_2, a) = \varnothing$$





DFA M'

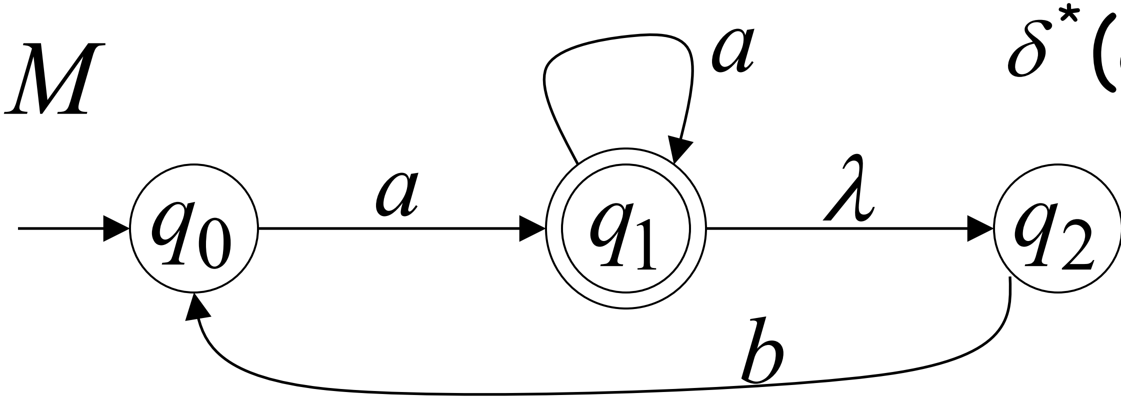


$$\delta^*(q_1, b) = \{q_0\}$$
$$\delta^*(q_2, b) = \{q_0\}$$

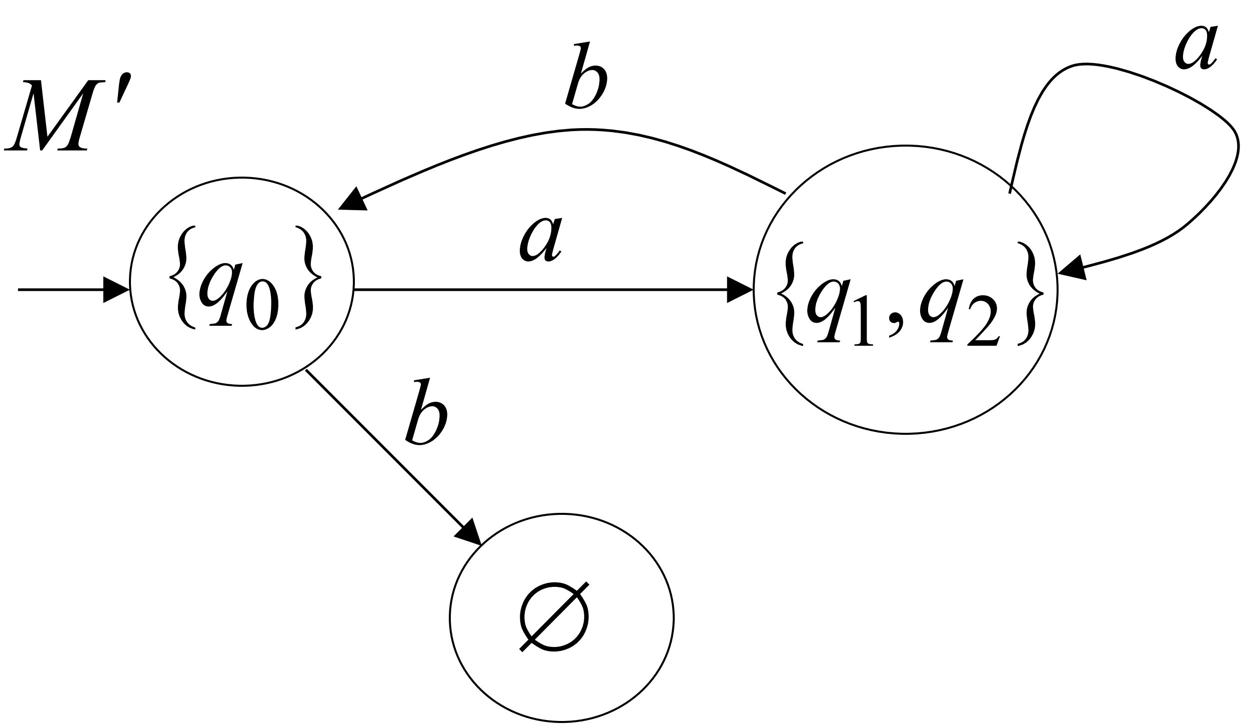
unione

$$\{q_0\}$$

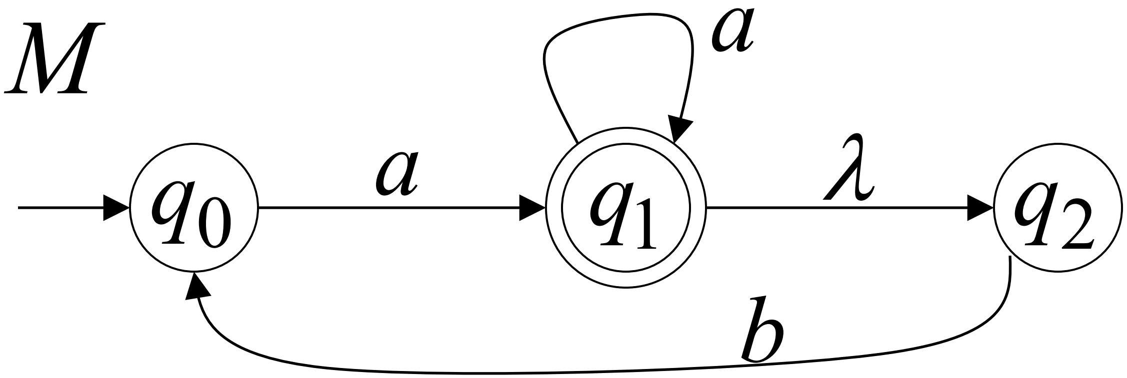
NFA M



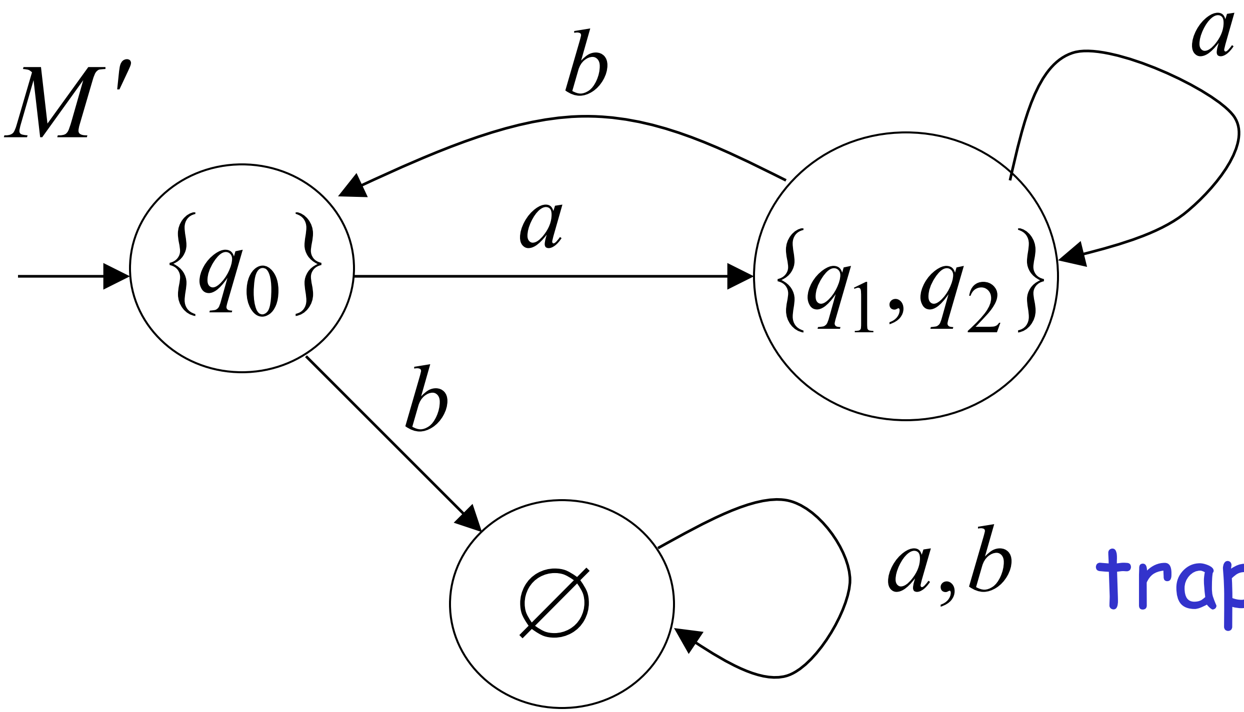
DFA M'



NFA M



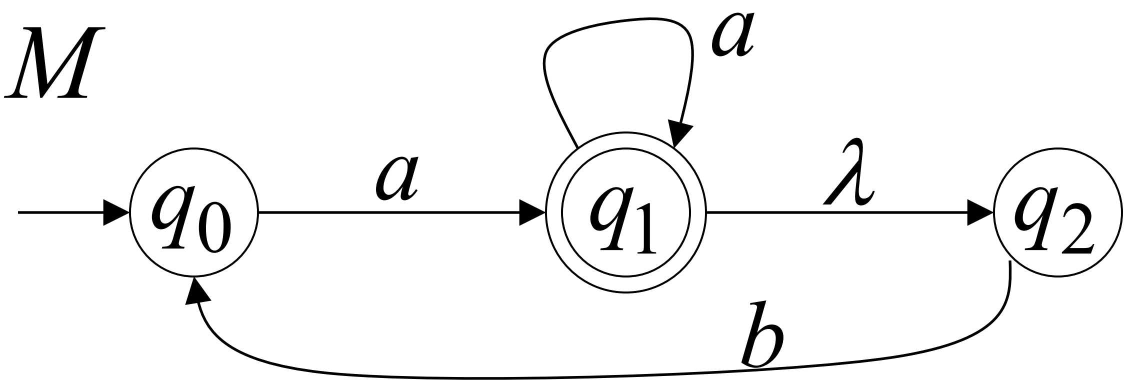
DFA M'



trap state

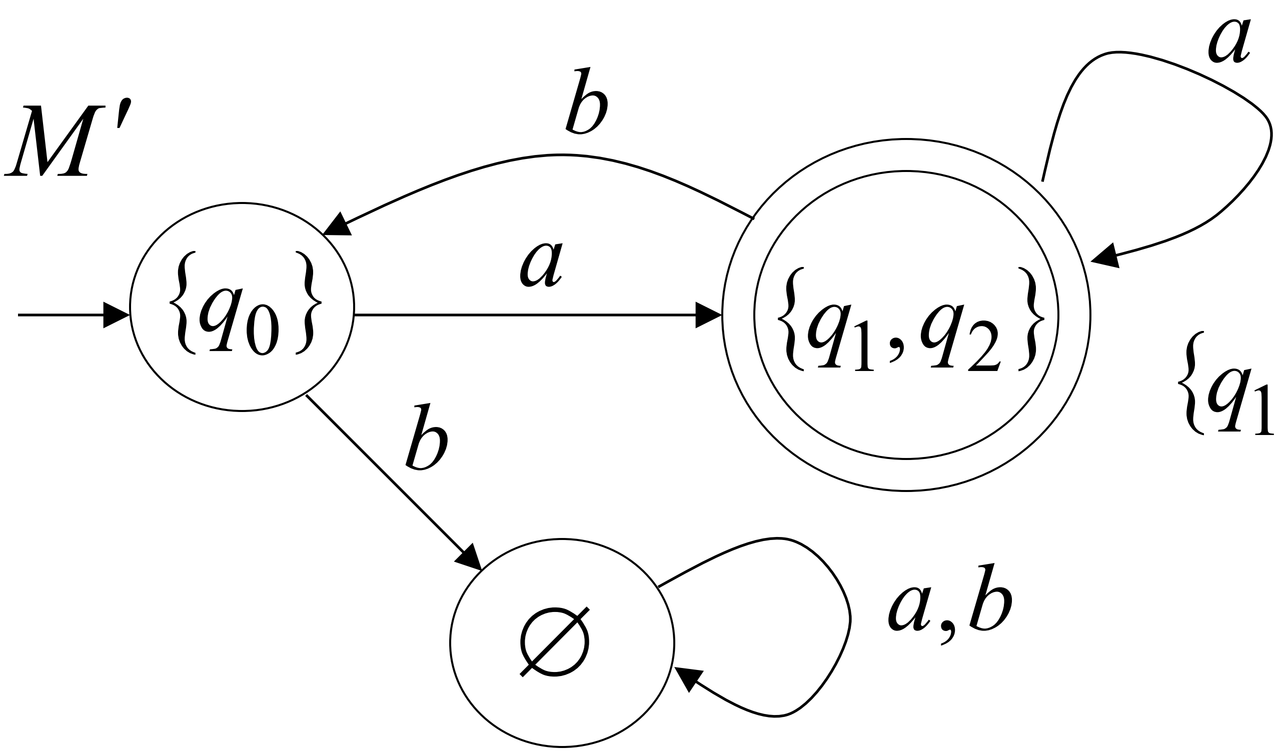
Fine della costruzione

NFA M



$q_1 \in F$

DFA M'



$\{q_1, q_2\} \in F'$

Procedura generale

Input: NFA M

Output: un equivalente DFA M'

con $L(M) = L(M')$

NFA ha gli stati $$q_0, q_1, q_2, \ldots$$

DFA ha gli stati definiti dall'insieme delle parti

$$\emptyset, \{q_0\}, \{q_1\}, \{q_0, q_1\}, \{q_1, q_2, q_3\}, \ldots.$$

Step della procedura

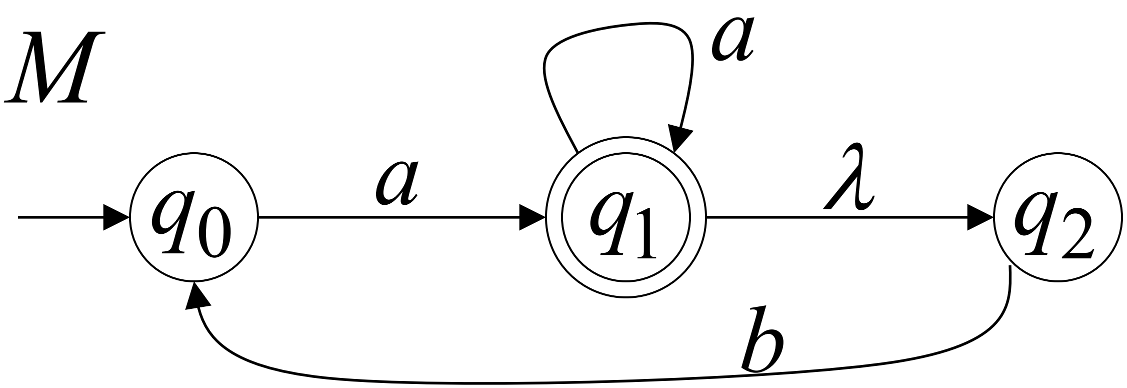
step

1. Stato iniziale NFA: q_0

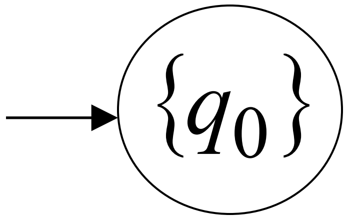
stato iniziale del DFA: $\{q_0\}$

esempio

NFA M



DFA M'



step

2. per ogni stato DFA $\{q_i, q_j, \ldots, q_m\}$

calcolo nel NFA

$$\left.\begin{aligned} &\delta^*(q_i, a) \\ \cup\, &\delta^*(q_j, a) \\ &\ldots \\ \cup\, &\delta^*(q_m, a) \end{aligned}\right\} = \{q'_k, q'_l, \ldots, q'_n\}$$

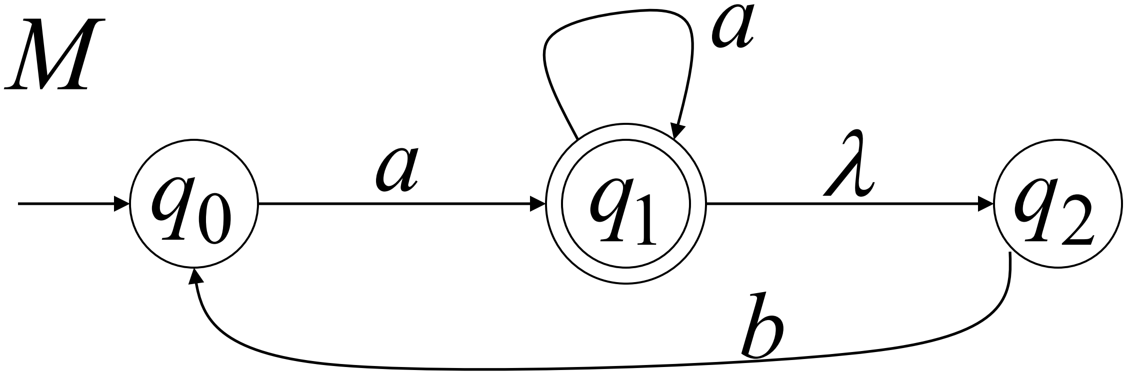
unione

addiziona questa nuova transizione al DFA

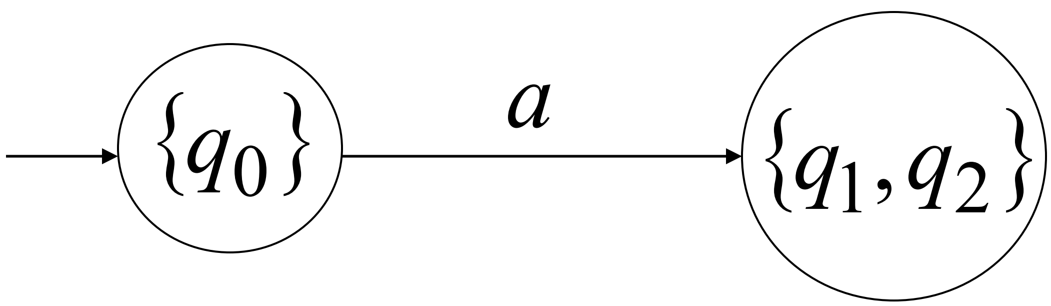
$$\delta(\{q_i, q_j, \ldots, q_m\},\ a) = \{q'_k, q'_l, \ldots, q'_n\}$$

esempio $\delta^*(q_0, a) = \{q_1, q_2\}$

NFA M



DFA M' $\delta(\{q_0\}, a) = \{q_1, q_2\}$

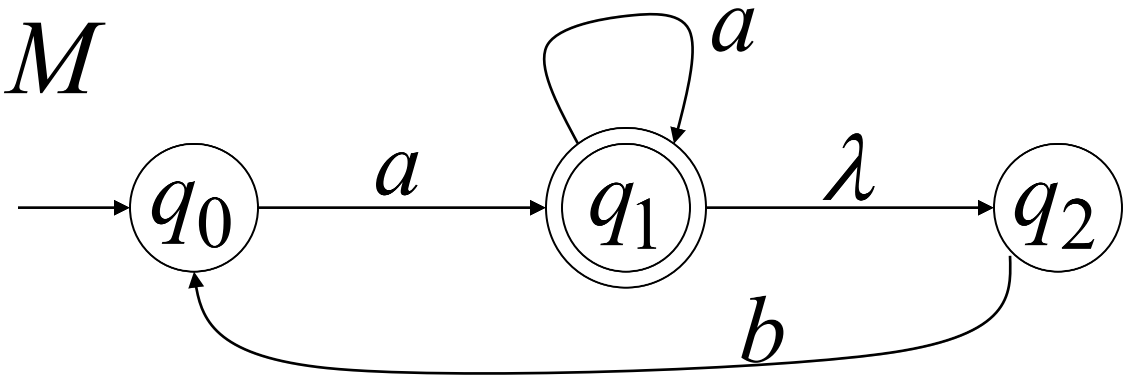


step

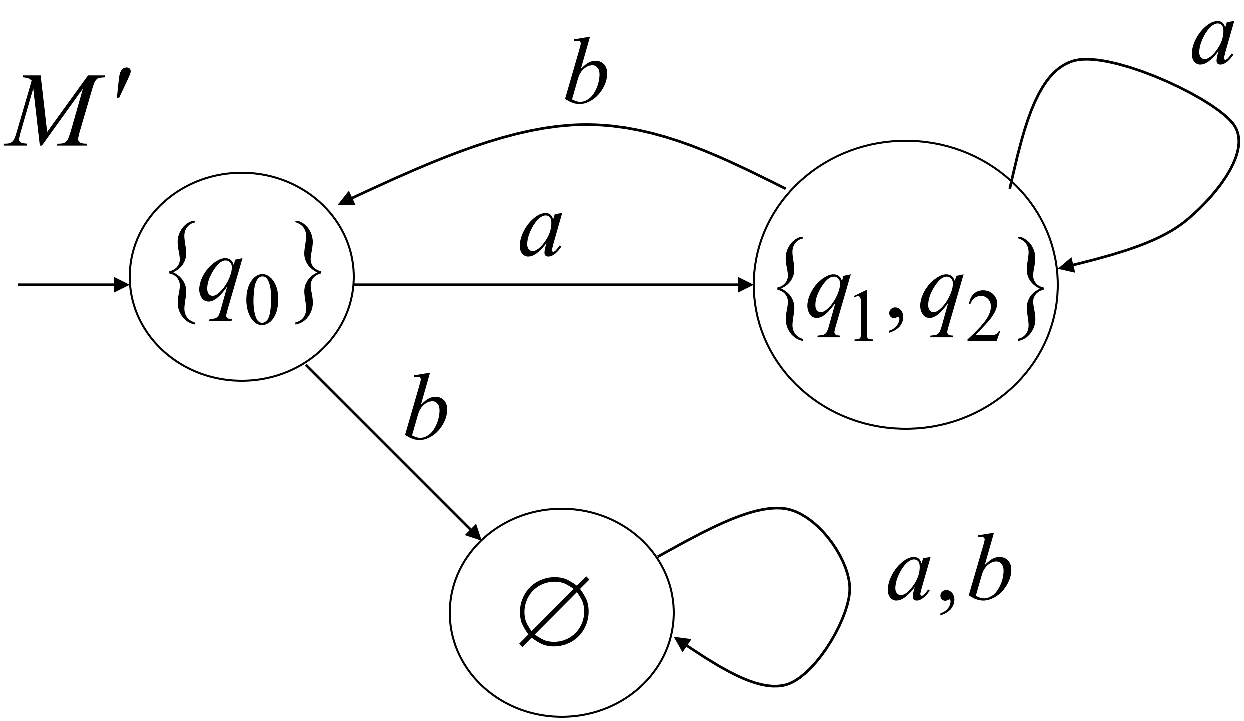
3. Ripeti lo step 2 per ogni stato nel DFA e simboli nell'alfabeto finchè non vi sono più stati che possono essere addizionati al DFA

esempio

NFA M



DFA M'



step

4.

$$\{q_i, q_j, \ldots, q_m\}$$

Per ogni stato DFA

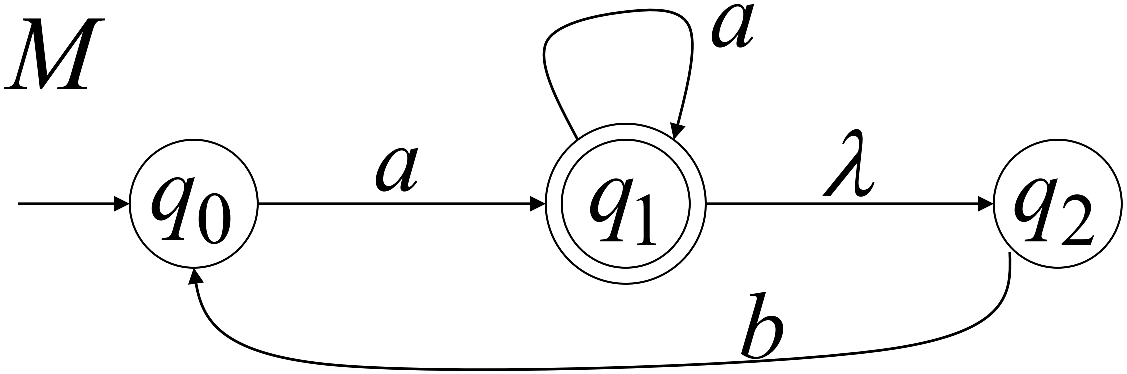
Se qualche q_j è uno stato di accettazione del NFA

Allora $\{q_i, q_j, \ldots, q_m\}$

è uno stato di accettazione del DFA

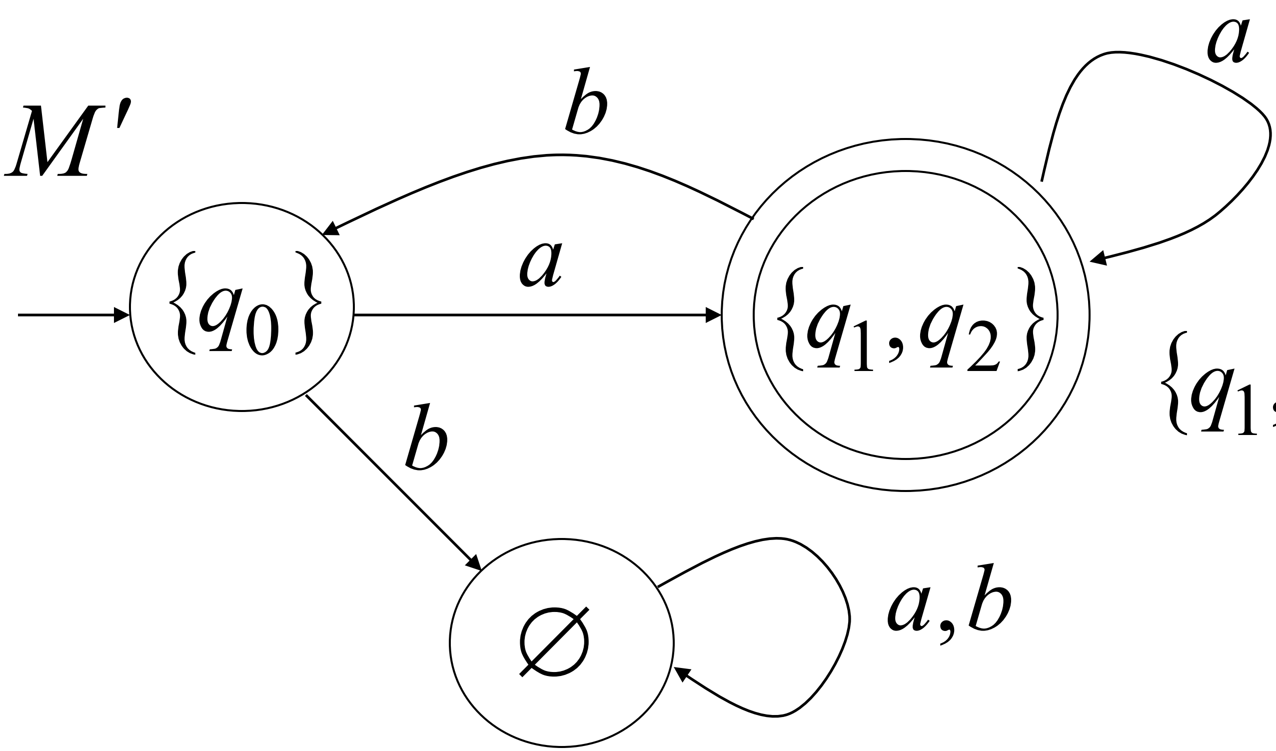
Example

NFA M



$q_1 \in F$

DFA M'



$\{q_1, q_2\} \in F'$

Step della procedura

step

1. Stato iniziale NFA: q_0

stato iniziale del DFA: $\{q_0\}$

step

2. per ogni stato DFA $\{q_i, q_j, \ldots, q_m\}$

calcolo nel NFA

$$\left.\begin{array}{r} \delta^*(q_i, a) \\ \cup\, \delta^*(q_j, a) \\ \ldots \\ \cup\, \delta^*(q_m, a) \end{array}\right\} = \{q'_k, q'_l, \ldots, q'_n\}$$

unione

addiziona questa nuova transizione al DFA

$$\delta(\{q_i, q_j, \ldots, q_m\},\ a) = \{q'_k, q'_l, \ldots, q'_n\}$$

step

3. Ripeti lo step 2 per ogni stato nel DFA e simboli nell'alfabeto finchè non vi sono più stati che possono essere addizionati al DFA

step

4. Per ogni stato del DFA $\{q_i, q_j, \ldots, q_m\}$

se è presente uno stato q_j finale, accettante, del NFA

allora, $\{q_i, q_j, \ldots, q_m\}$ è uno stato accettante del DFA

Lemma:

Se traduciamo un NFA M in un DFA M'
Allora i due automata sono equivalenti:

$$L(M) = L(M')$$

dimostrazione:

Dobbiamo dimostrare che: $L(M) \subseteq L(M')$

e

$$L(M) \supseteq L(M')$$

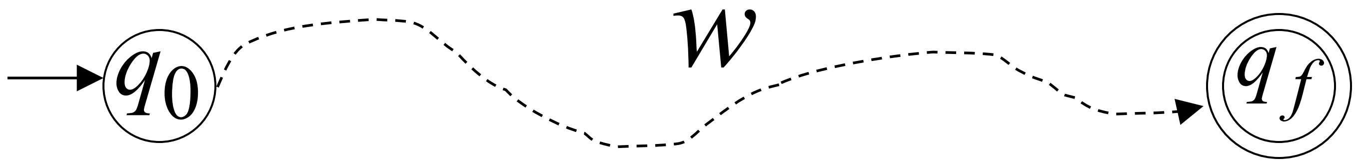
Mostriamo che: $L(M) \subseteq L(M')$

NFA contenuto in DFA

Dobbiamo provare che:

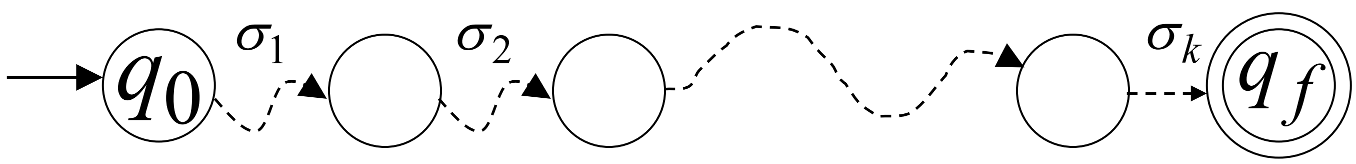
$$w \in L(M) \quad \Longrightarrow \quad w \in L(M')$$

considera $w \in L(M)$ NFA



simboli

$$w = \sigma_1 \sigma_2 \cdots \sigma_k$$

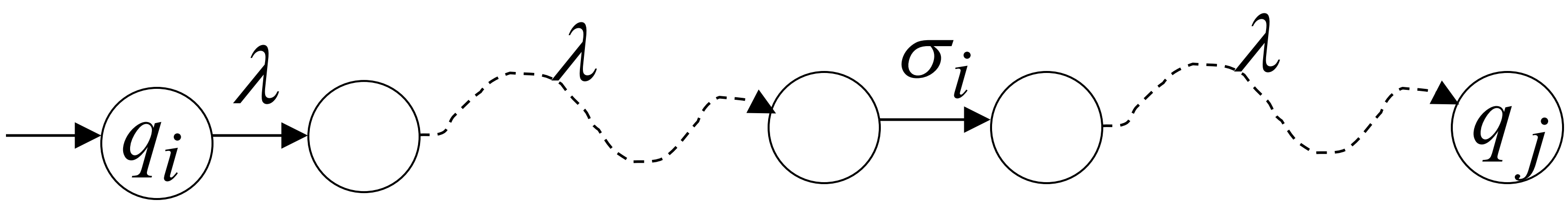


ricordiamo

Simboli, lng 1

Denota un sotto cammino tale che

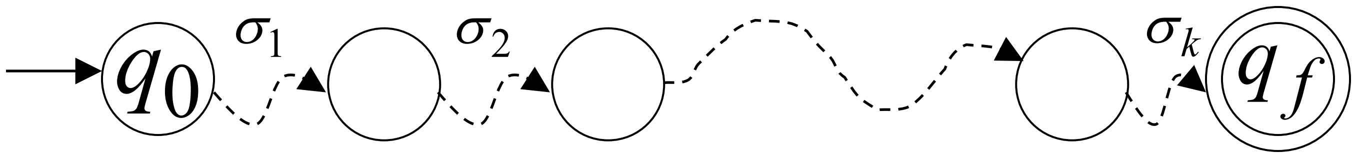
simboli



Mostriamo che se $w \in L(M)$

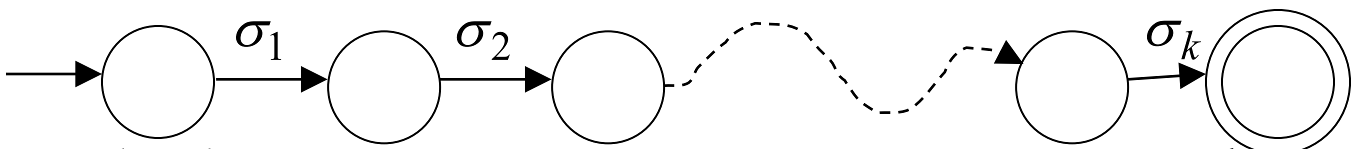
$$w = \sigma_1 \sigma_2 \cdots \sigma_k$$

NFA $M:$



allora

DFA $M':$



$\{q_0\}$

state label

$w \in L(M')$

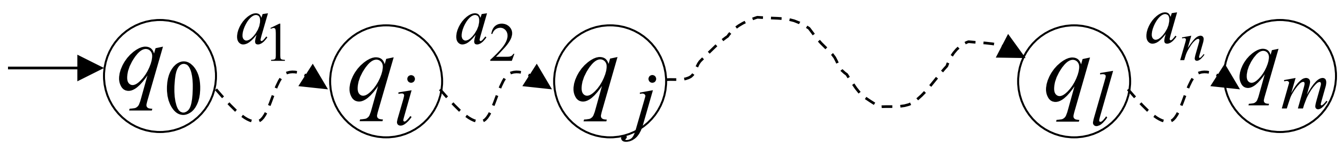
$\{q_f, \ldots\}$

state label

In modo piu generale , mostreremo che se in M :

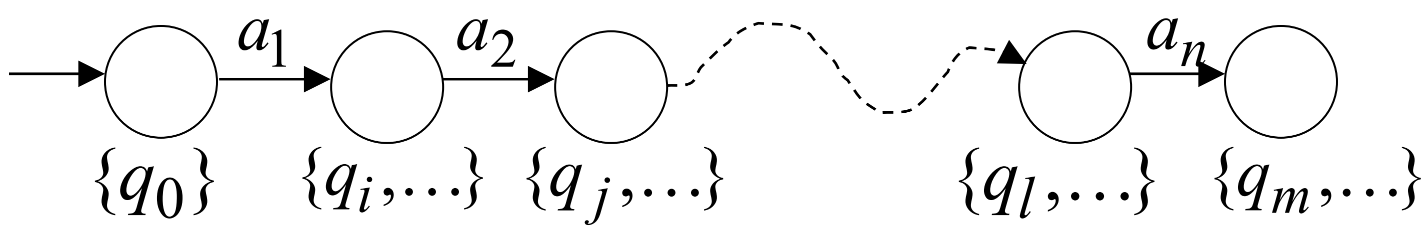
(stringa arbitraria) $v = a_1 a_2 \cdots a_n$

NFA M :



allora

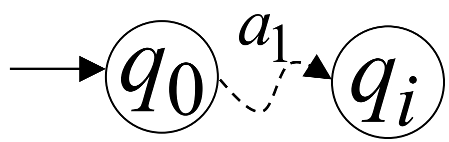
DFA M' :



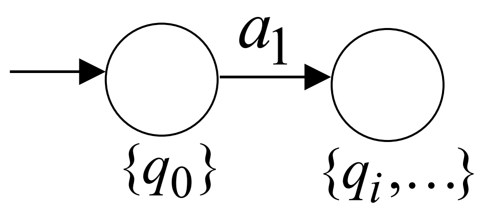
Dimostrazione per induzione su $|v|$

Base induzione: $|v| = 1$ $\qquad v = a_1$

NFA M :



DFA M' :



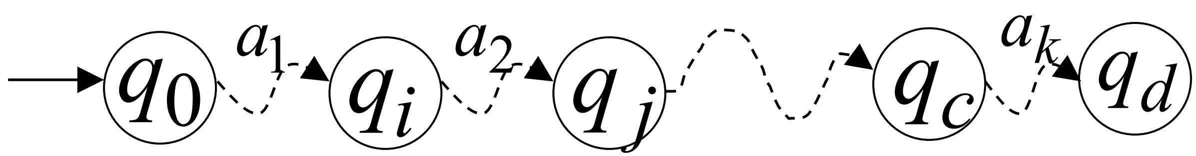
[vero per come costruito M']

Ipotesi induttiva: $1 \leq |v| \leq k$

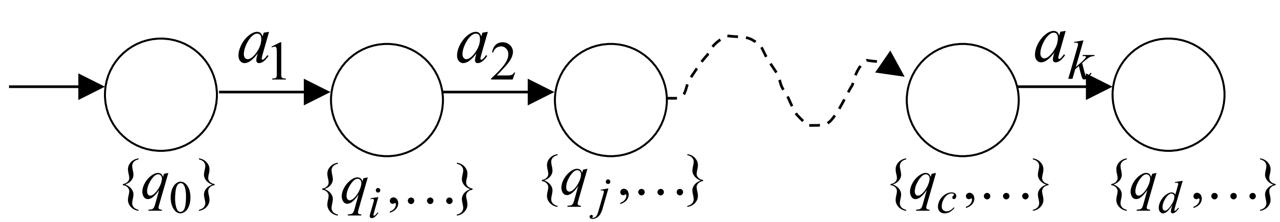
$$v = a_1 a_2 \cdots a_k$$

Supponiamo valga

NFA $M:$



DFA $M':$

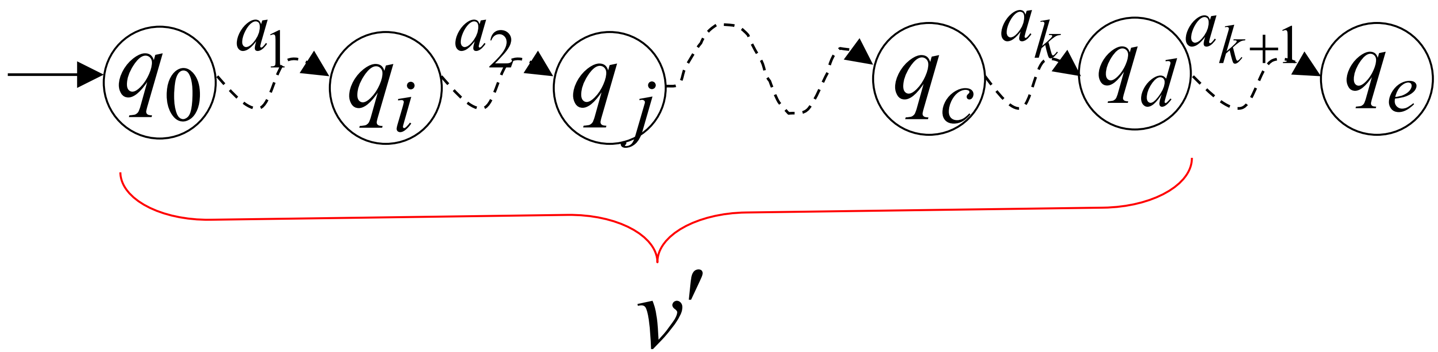


Step induttivo: $|v| = k+1$

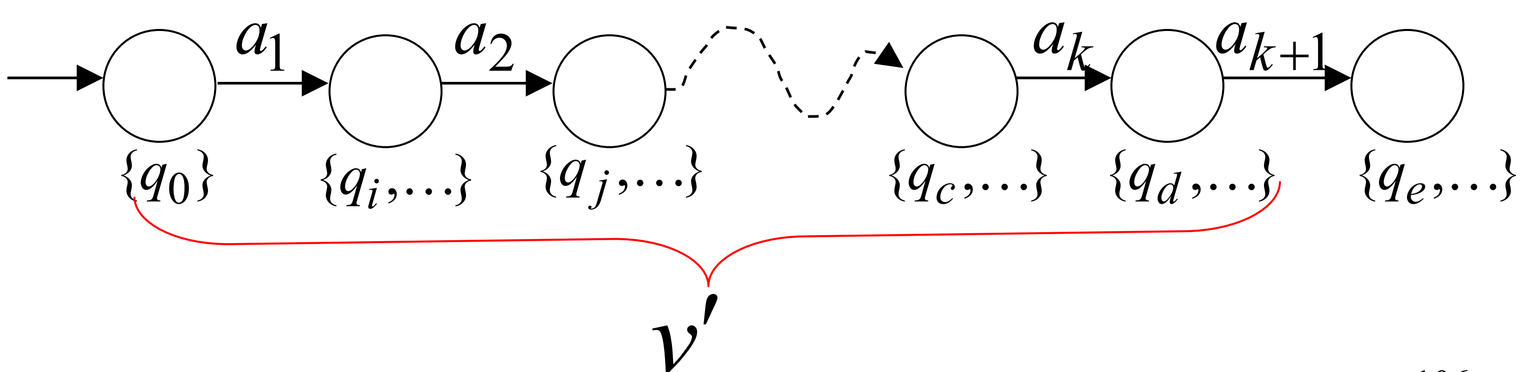
$$v = \underbrace{a_1 a_2 \cdots a_k}_{v'} a_{k+1} = v' a_{k+1}$$

Vero per costruzione di M'

NFA M:



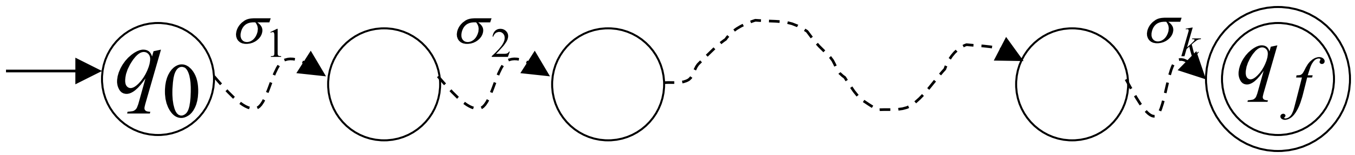
DFA M':



Quindi se $w \in L(M)$

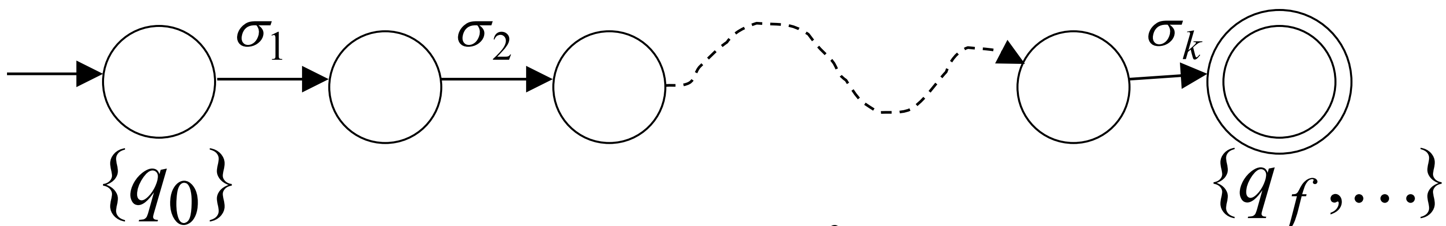
$w = \sigma_1 \sigma_2 \cdots \sigma_k$

NFA $M:$



allora

DFA $M':$



$w \in L(M')$

allora: $L(M) \subseteq L(M')$ dimostrato

e $L(M) \supseteq L(M')$ banale

quindi: $L(M) = L(M')$

Fine lemma

Espressioni regolari

Ricordo: un linguaggio è regolare se è riconosciuto da un NFA (=DFA)

Definizione sintattica

L'espressioni regolari di base : $\emptyset, \quad \lambda, \quad \alpha$

Date le espressioi regolari r_1 e r_2

$$\left.\begin{array}{l} r_1 + r_2 \\ r_1 \cdot r_2 \\ r_1{}^* \\ (r_1) \end{array}\right\} \text{Sono espressioni regolari}$$

Una semantica: Linguaggi associati alle espressioni regolari

Per le espressioni regolari di base:

$$L(\varnothing) = \varnothing$$

$$L(\lambda) = \{\lambda\}$$

$$L(a) = \{a\}$$

passo

per le espressioni regolari r_1 e r_2

$$L(r_1 + r_2) = L(r_1) \cup L(r_2)$$

$$L(r_1 \cdot r_2) = L(r_1)\, L(r_2)$$

$$L(r_1 *) = (L(r_1))*$$

$$L((r_1)) = L(r_1)$$

Linguaggi associati alle espressioni regolari

$L(r)$: linguaggio associato all'espressione r

esempio

$$L((a+b\cdot c)^*)=\{\lambda,a,bc,aa,abc,bca,...\}$$

Espressioni regolari definiscono solo e soltanto i linguaggi regolari?

proprietà dei
linguaggi regolari e automi

per linguaggi regolari L_1 e L_2 dimostreremo che:

Unione: $L_1 \cup L_2$

Concatenatione: $L_1 L_2$

Star: $L_1 {}^*$

Reversal: $L_1{}^R$

Complemento: $\overline{L_1}$

Intersezione: $L_1 \cap L_2$

sono linguaggi regolari

diremo: linguaggi regolari sono **chiusi sotto**

Unione: $L_1 \cup L_2$

Concatenatione: $L_1 L_2$

Star: $L_1 *$

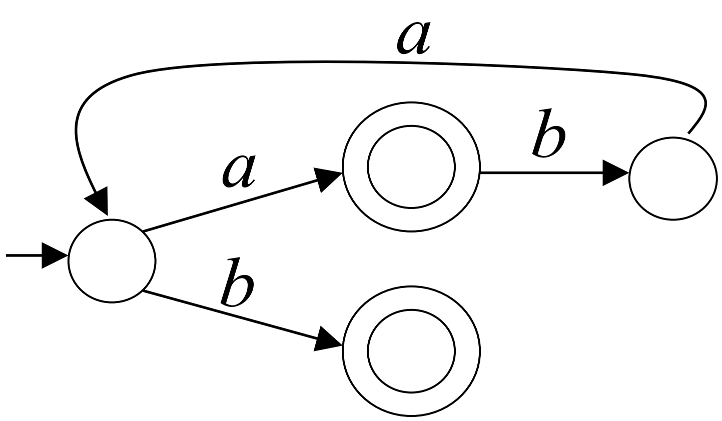
Reversal: L_1^R

Complemento: $\overline{L_1}$

Intersezione: $L_1 \cap L_2$

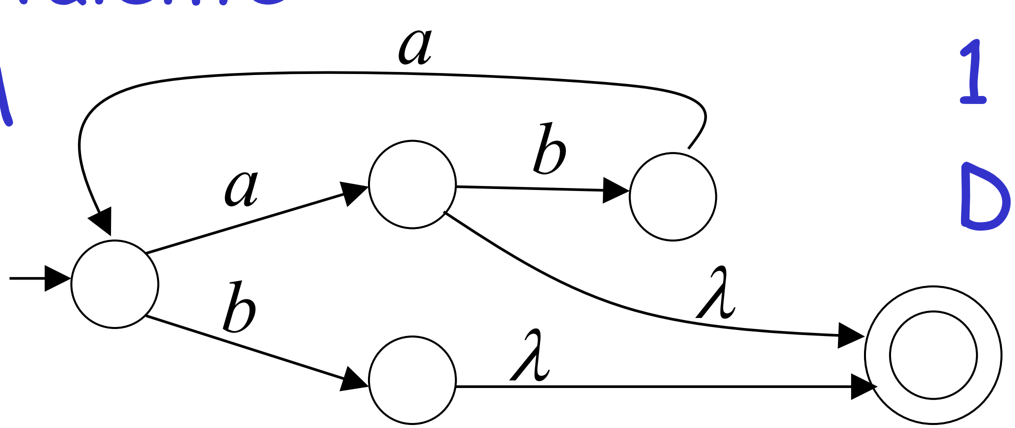
Useremo nfa con un solo stato finale

NFA



2 stati di accettazione

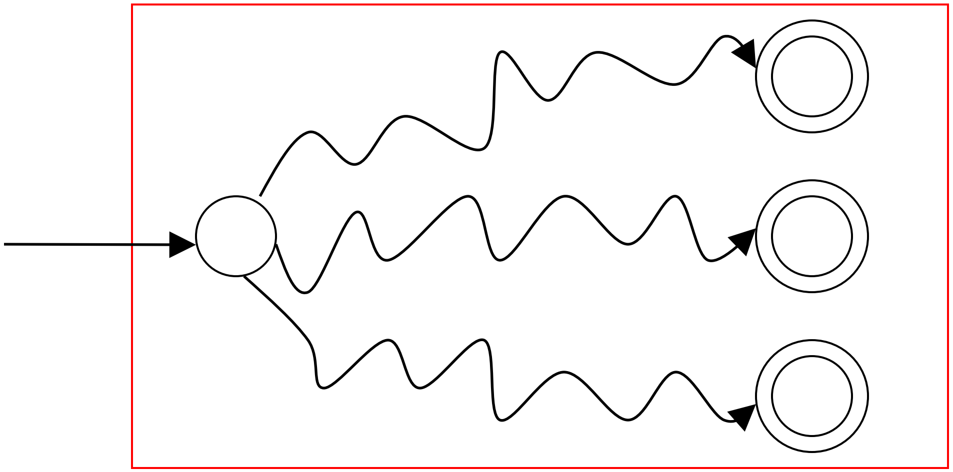
Equivalente NFA



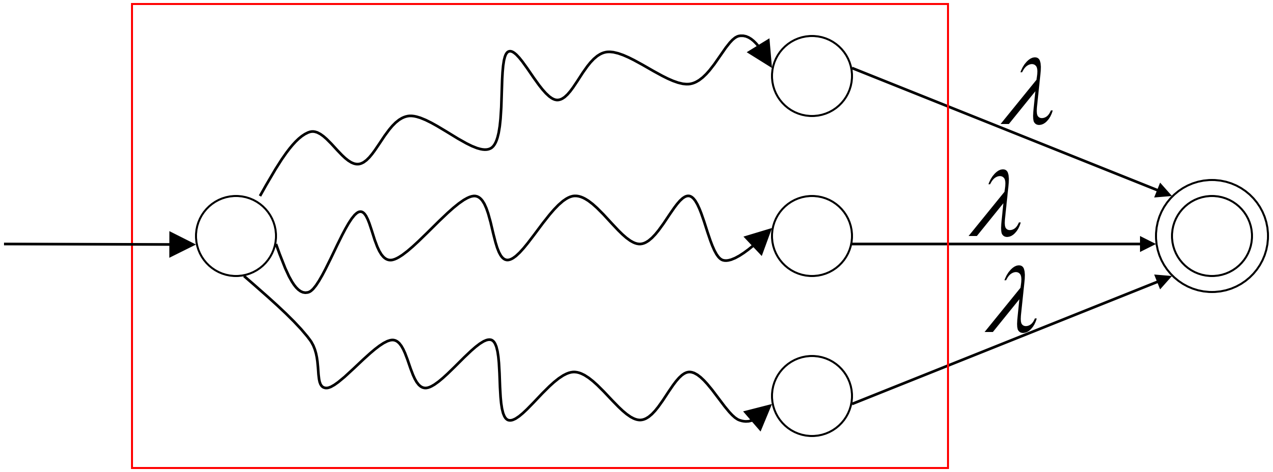
1 solo stato Di accettazione

In Generale

NFA



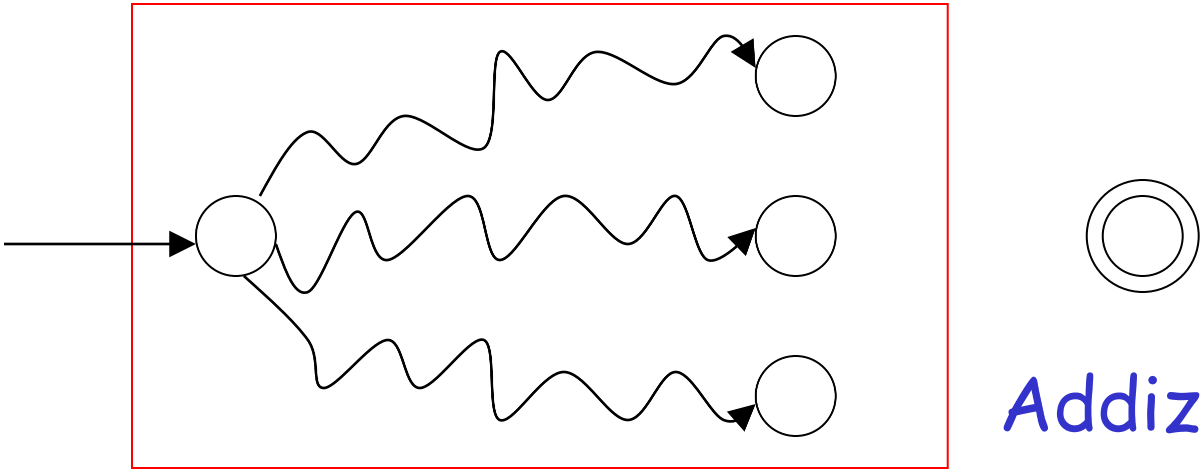
Equivalent NFA



Un solo
Stato di
accettazione

Caso estremo

NFA senza stato di accettazione



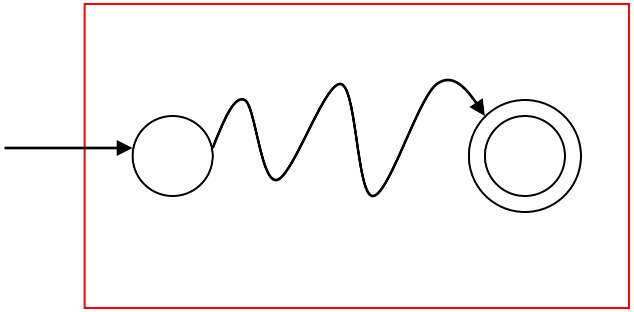
Addizioniamo
Uno stato di
Accettazione
Senza transizione

Prendiamo due linguaggi

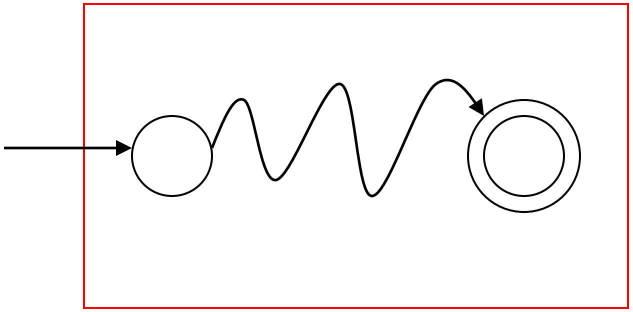
Linguaggio regolare L_1 linguaggio regolare L_2

$$L(M_1) = L_1 \qquad\qquad L(M_2) = L_2$$

NFA M_1



NFA M_2



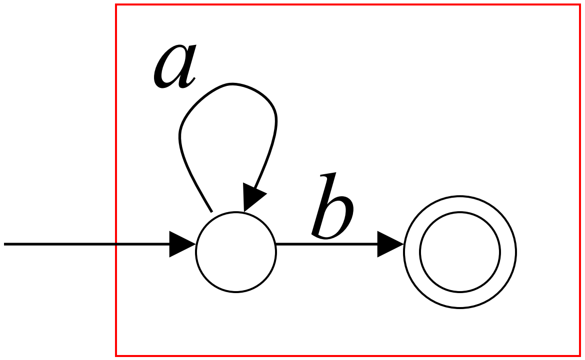
Un solo stato di accettazione Un solo stato di accettazione

Esempio

$n \geq 0$

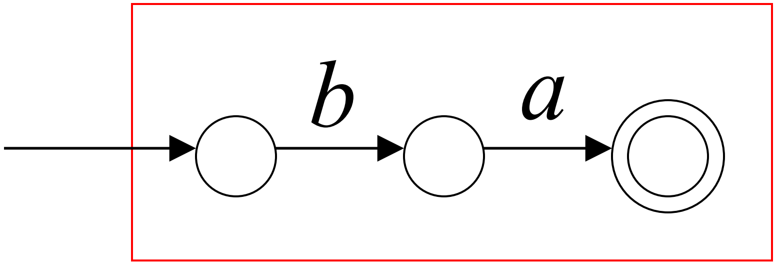
$L_1 = \{a^n b\}$

M_1



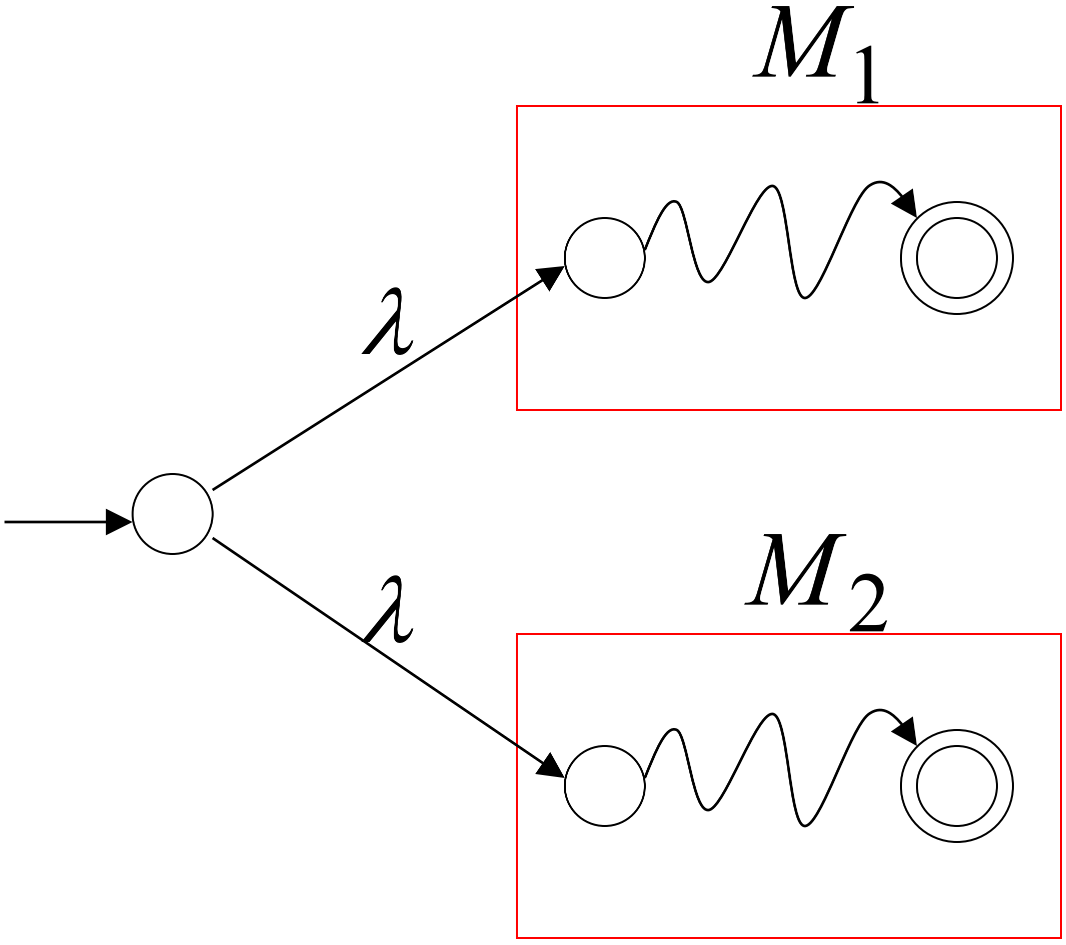
$L_2 = \{ba\}$

M_2



Unione

NFA per $L_1 \cup L_2$

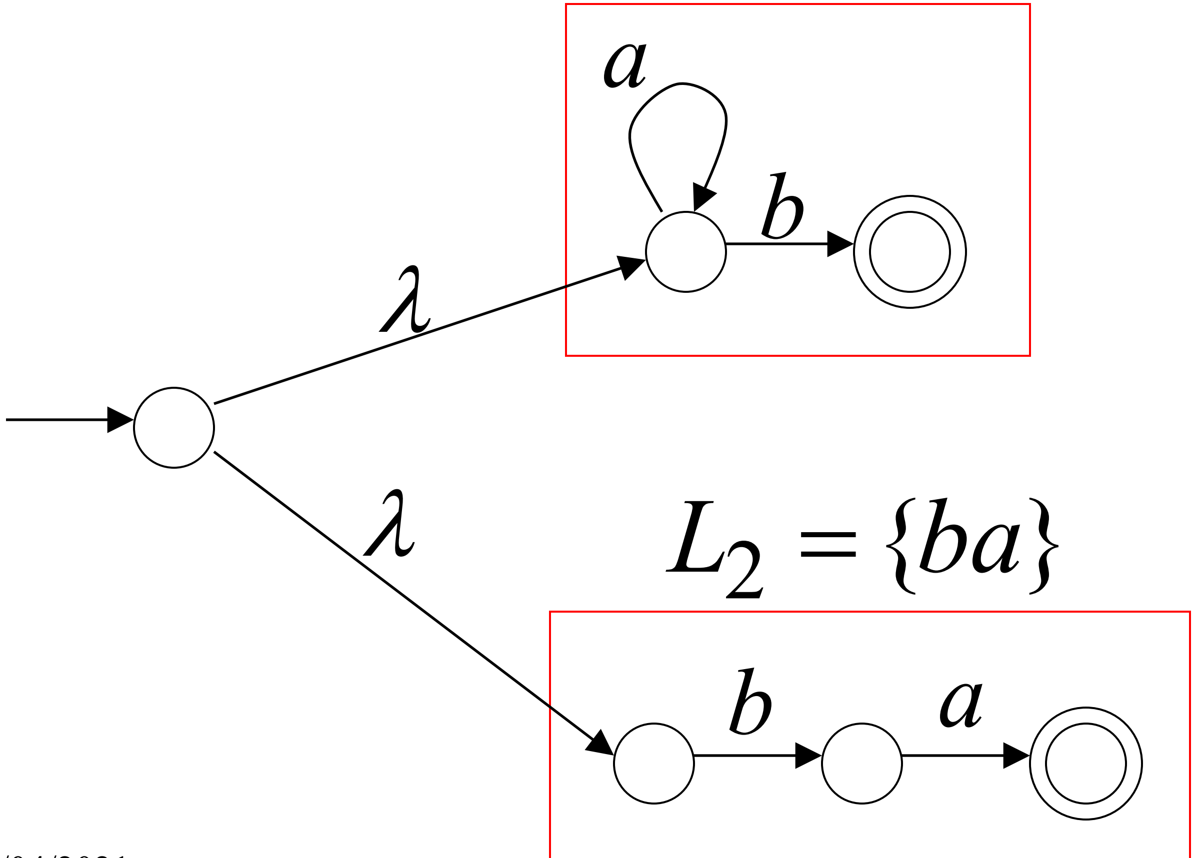


Esempio

NFA per $L_1 \cup L_2 = \{a^n b\} \cup \{ba\}$

$L_1 = \{a^n b\}$

$L_2 = \{ba\}$



Evitiamo le transizioni con le lambda transizioni.
Mostriamo che a partire da due automi (N_1,N_2) , si può costruire l'automa unione dei due linguaggi definiti dagli automi precedenti.

Gli stati del nuovo automa sono l'unione degli stati precedenti, K_1 e K_2, più un nuovo stato iniziale q'_0.

Funzione transizione dell'automa unione, N, a partire dalle delta di N_1 e N_2.

$$\delta_N(q,a) = \delta_{N_1}(q,a), q \in K_1, a \in \Sigma_1$$

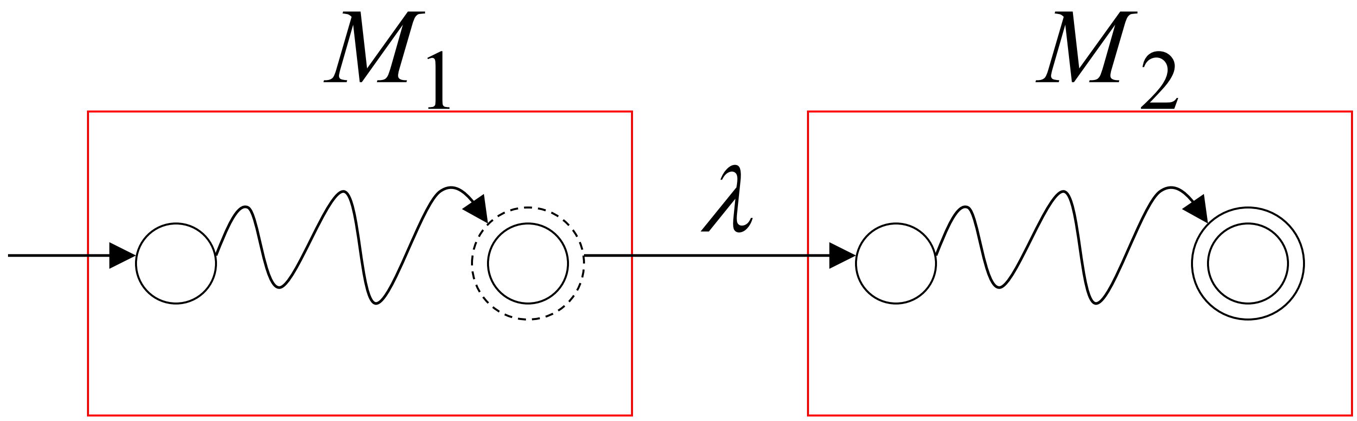
$$\delta_N(q,a) = \delta_{N_2}(q,a), q \in K_2, a \in \Sigma_2$$

$$\delta_N(q'_0,a) = \delta_{N_1}(q_{0_1},a) \bigcup \delta_{N_2}(q_{0_2},a), a \in \Sigma$$

Provare che la definizione precedente definisce l'unione di due automi.

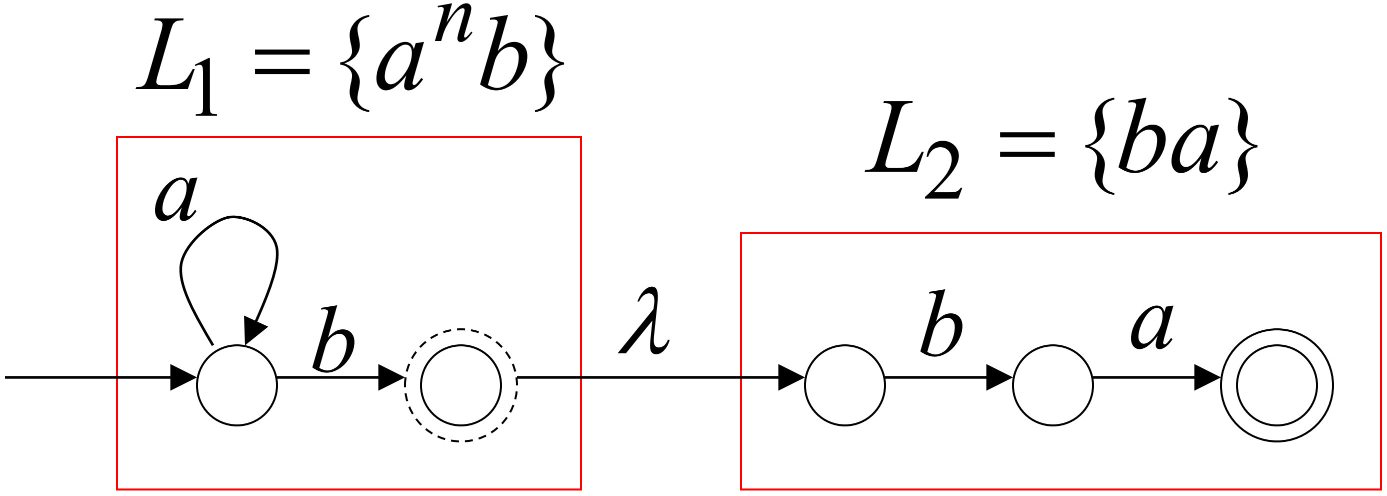
Concatenazione

NFA per $L_1 L_2$



esempio

NFA per $L_1 L_2 = \{a^n b\}\{ba\} = \{a^n bba\}$

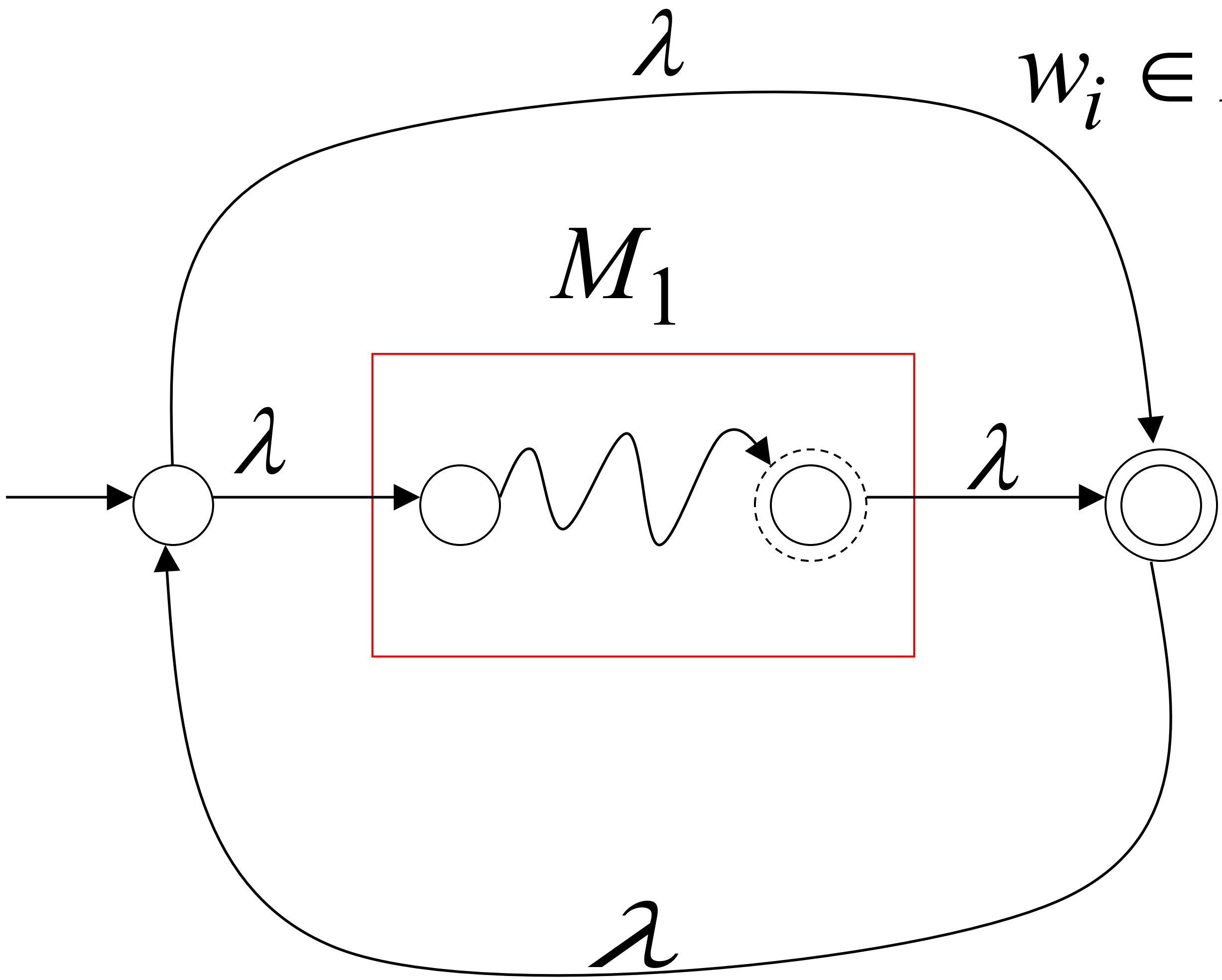


Star

NFA per $L_1{}^*$

$w = w_1 w_2 \cdots w_k$

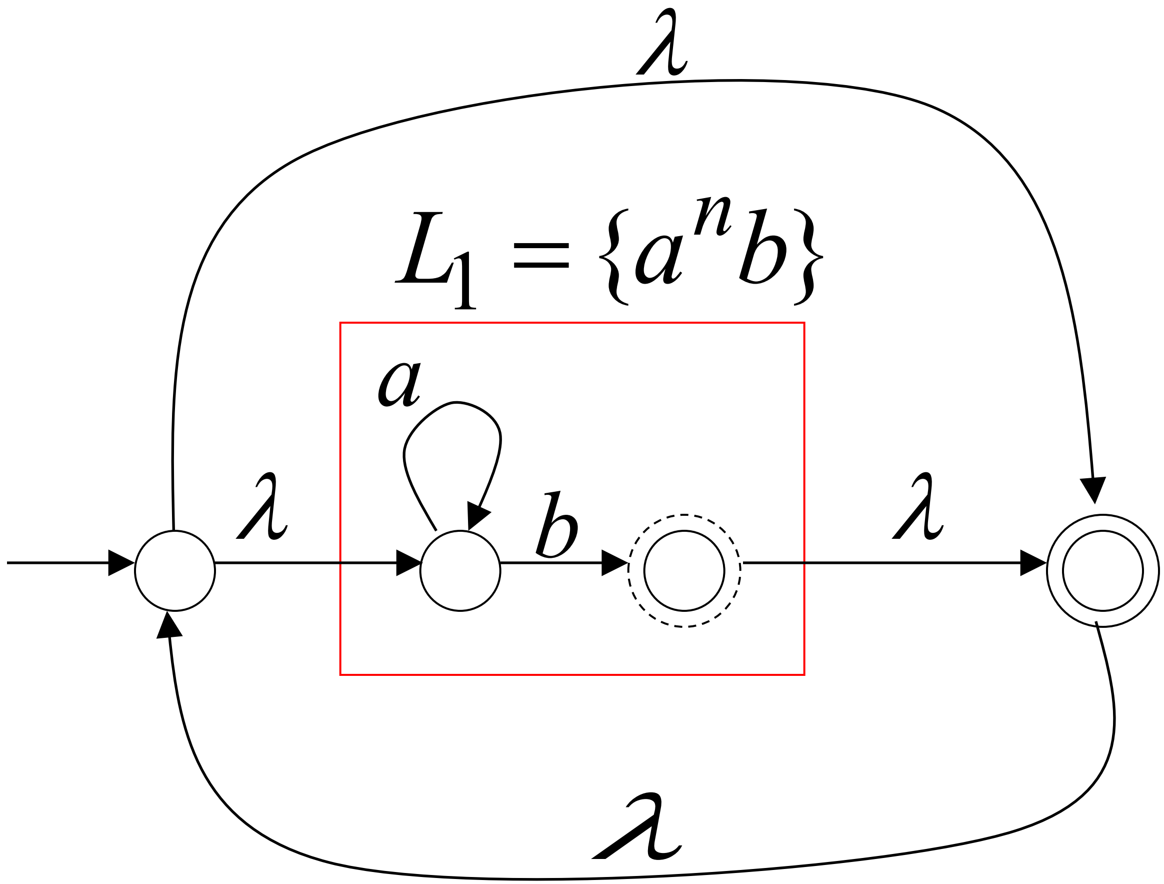
$w_i \in L_1$



$\lambda \in L_1{}^*$

esempio

NFA per $L_1{}^* = \{a^n b\}^*$



Reverse

NFA per L_1^R

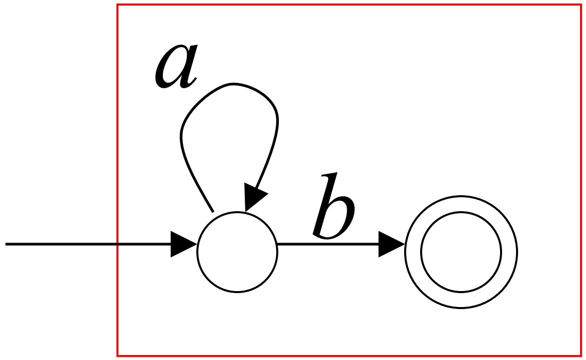
1. Reverse tutte le transizioni

2. Stato iniziale quello finale,
quello finale stato iniziale

esempio

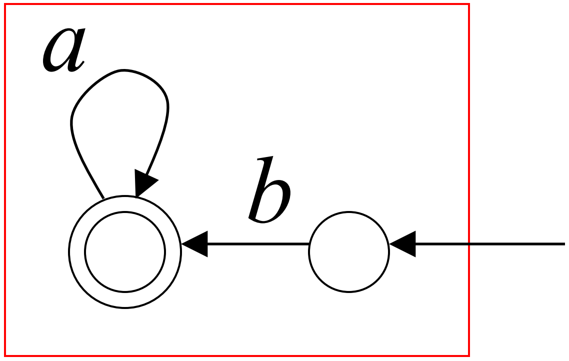
M_1

$L_1 = \{a^n b\}$

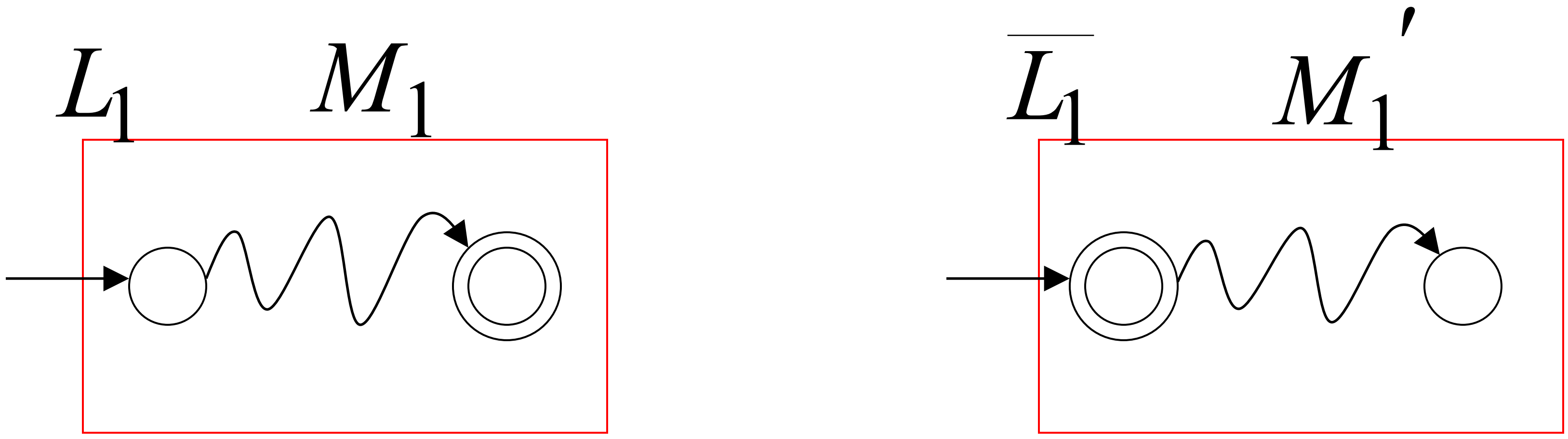


M_1'

$L_1^R = \{ba^n\}$



Complemento

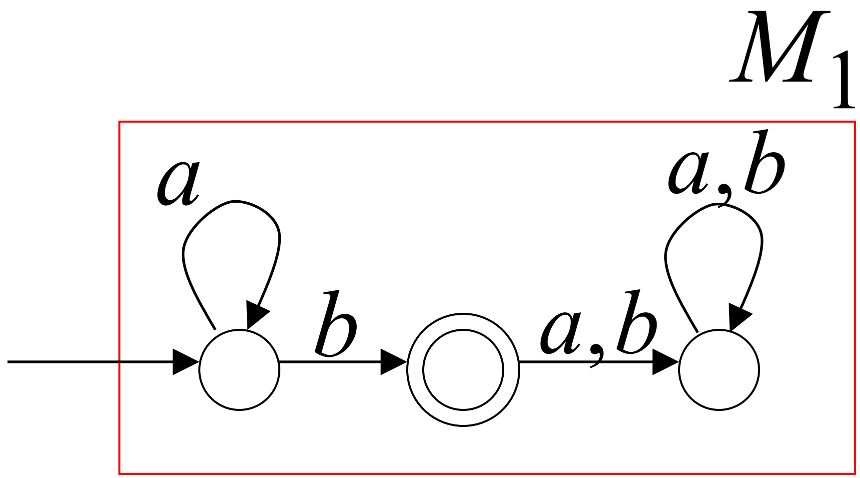


prendiamo il DFA che accetta L_1

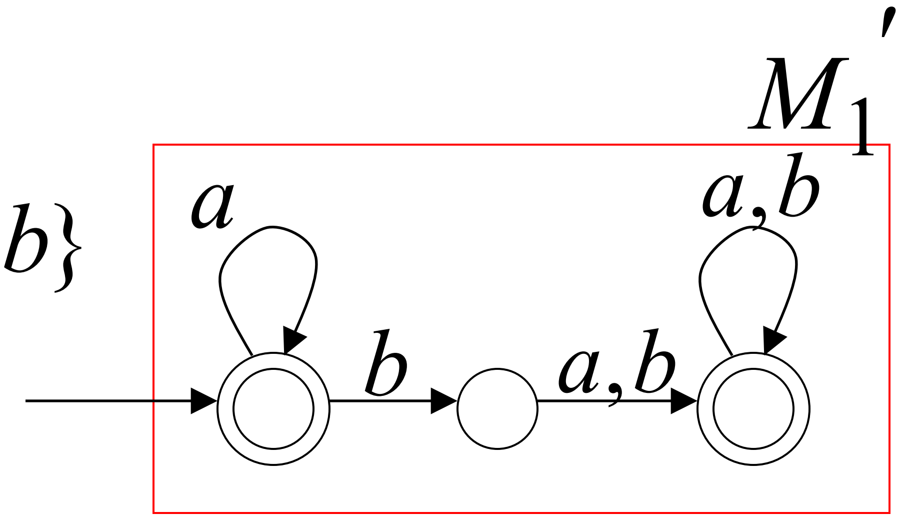
1. Stati non finale diventano finale, e vice-versa, resta lo stato iniziale.

esempio

$$L_1 = \{a^n b\}$$



$$\overline{L_1} = \{a,b\}^* - \{a^n b\}$$



Intersezione

L_1 regolari

L_2 regolari

mostriamo $L_1 \cap L_2$ regolari

leggi DeMorgan : $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$

L_1 , L_2 regolari

$\Rightarrow$ $\overline{L_1}$, $\overline{L_2}$ regolari

$\Rightarrow$ $\overline{L_1} \cup \overline{L_2}$ regolari

$\Rightarrow$ $\overline{\overline{L_1} \cup \overline{L_2}}$ regolari

$\Rightarrow$ $L_1 \cap L_2$ regolari

esempio

$$L_1 = \{a^n b\}$$ regolari

$$L_2 = \{ab, ba\}$$ regolari

$$\Rightarrow L_1 \cap L_2 = \{ab\}$$

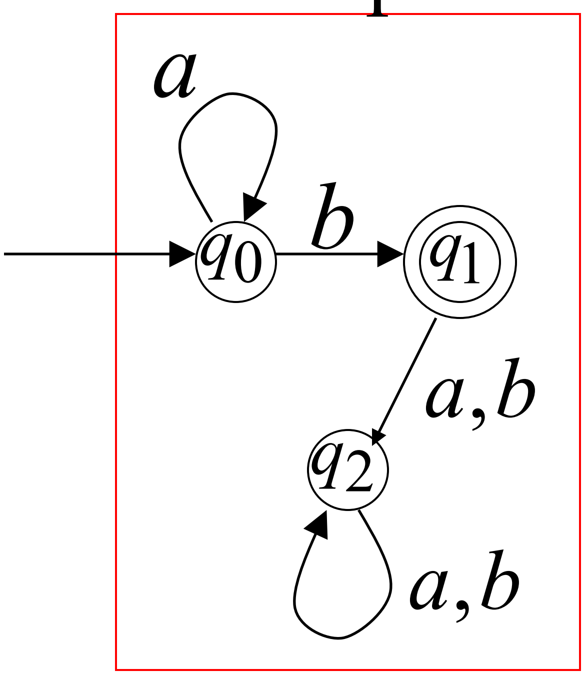
regolari

esempio:

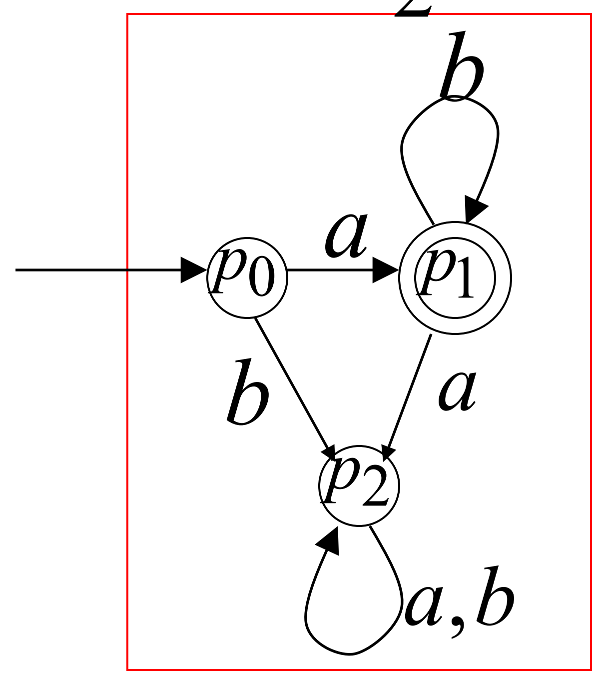
$$n \geq 0 \qquad\qquad m \geq 0$$

$$L_1 = \{a^n b\} \qquad\qquad L_2 = \{ab^m\}$$

M_1

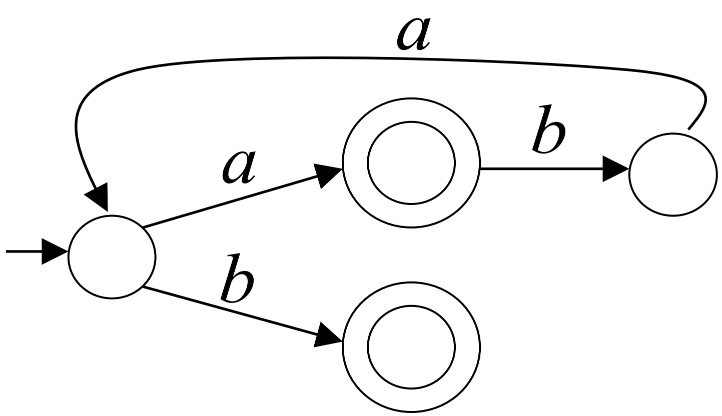


M_2



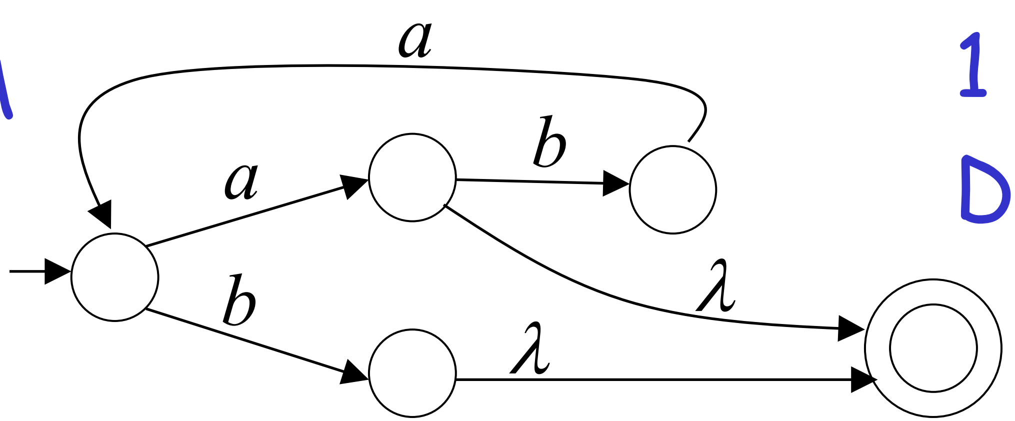
Useremo nfa con un solo stato finale

NFA



2 stati di
accettazione

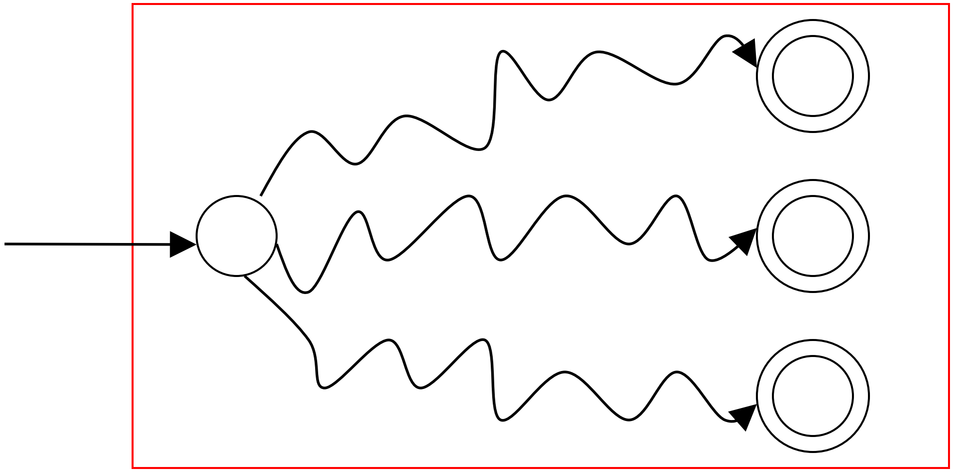
Equivalente
NFA



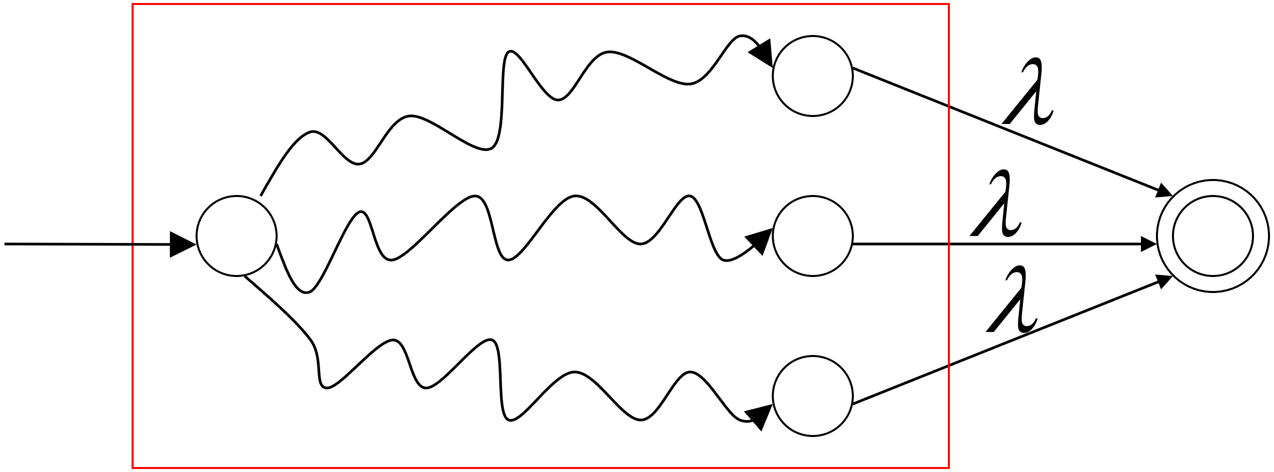
1 solo stato
Di accettazione

In Generale

NFA

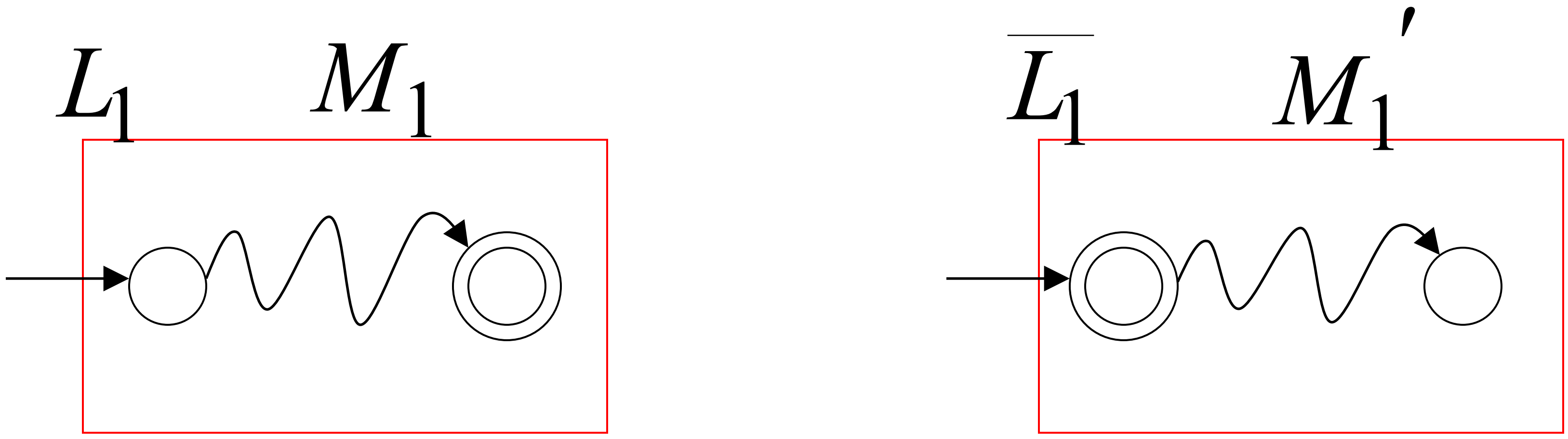


Equivalent NFA



Un solo
Stato di
accettazione

Complemento



1 prendiamo il **DFA** che accetta L_1

2. Stati non finale diventano finale, e vice-versa

Chiusura rispetto intersezione

macchina M_1

DFA per L_1

macchina M_2

DFA per L_2

Costruiamo un DFA M che accetta $L_1 \cap L_2$

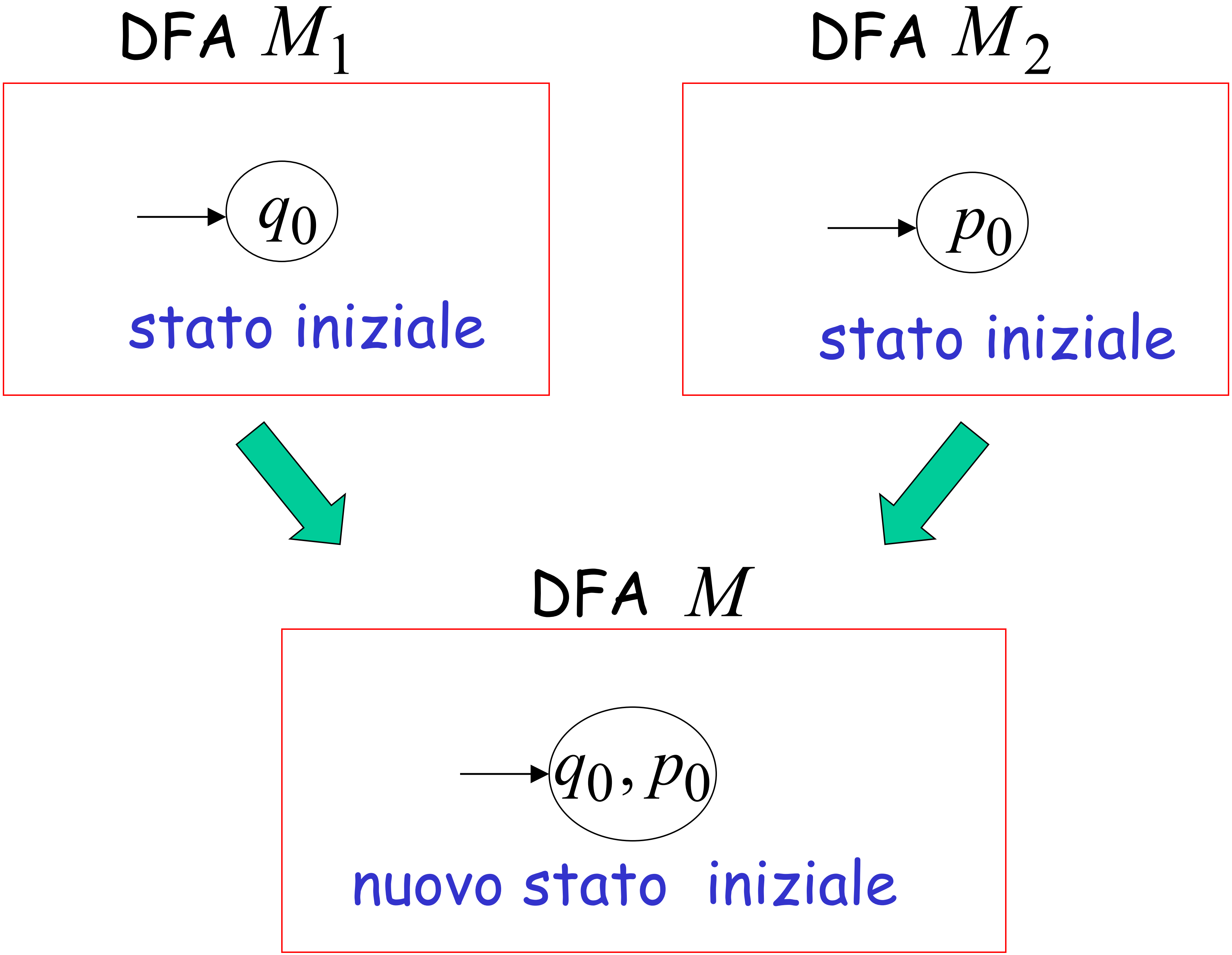
M Simula in parallelo M_1 e M_2

Stati in M

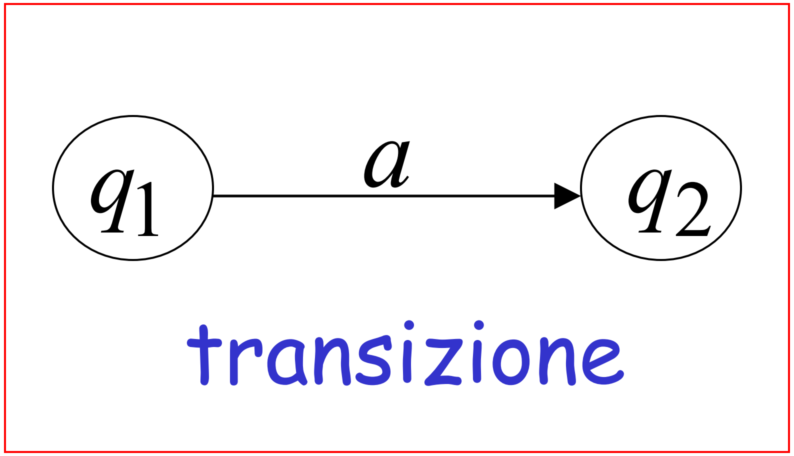
q_i, p_j

Stato in M_1

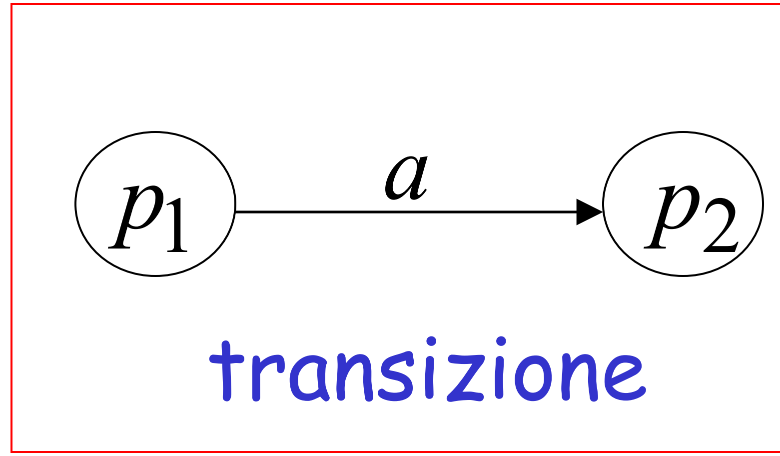
Stato in M_2


DFA M_1
q_0
stato iniziale
DFA M_2
p_0
stato iniziale
DFA M
q_0, p_0
nuovo stato iniziale

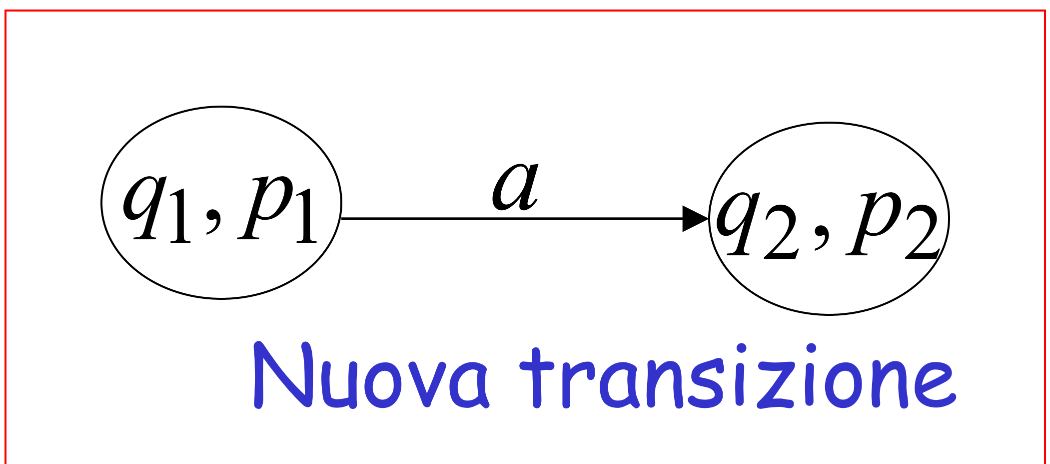
DFA M_1

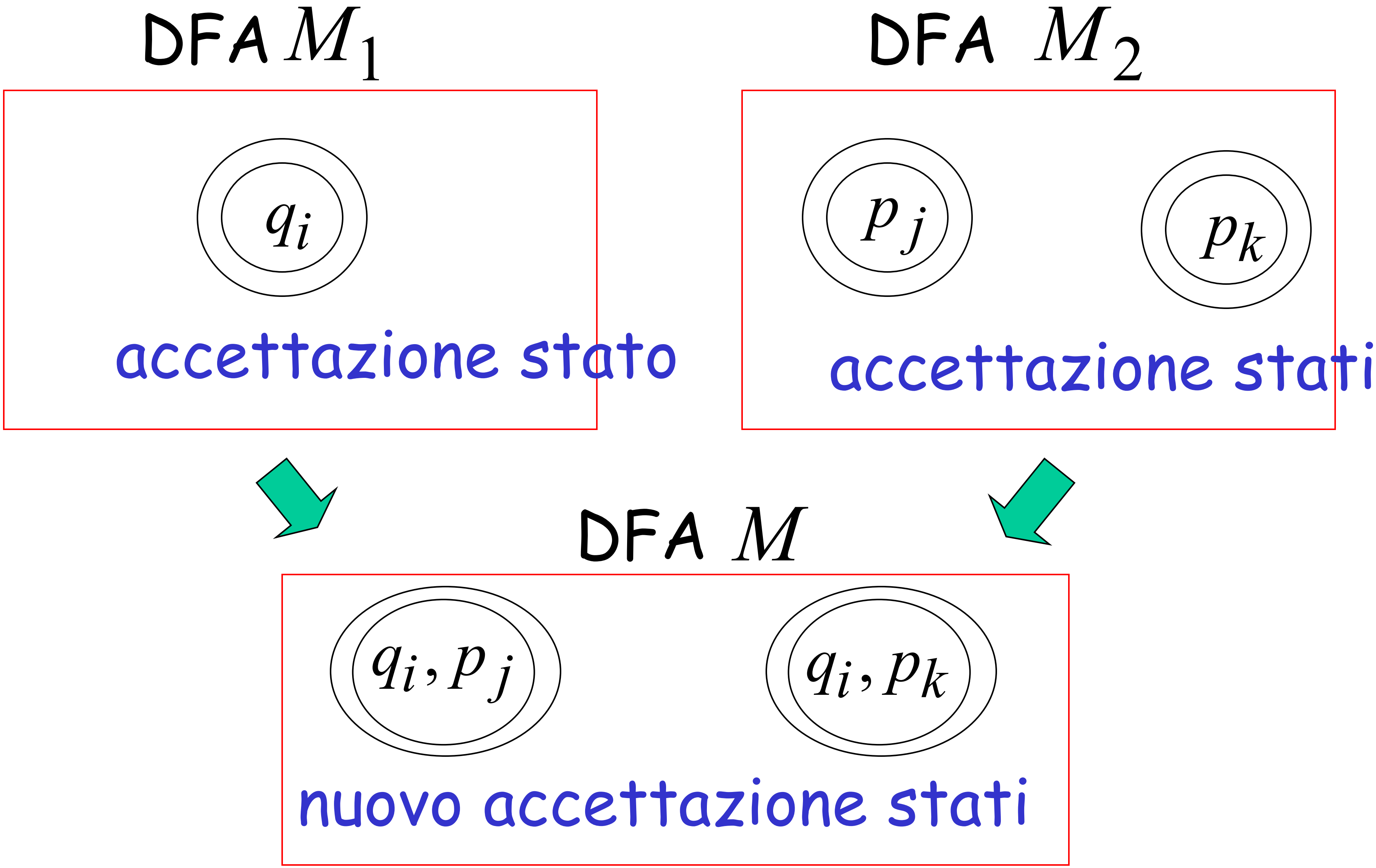


DFA M_2



DFA M



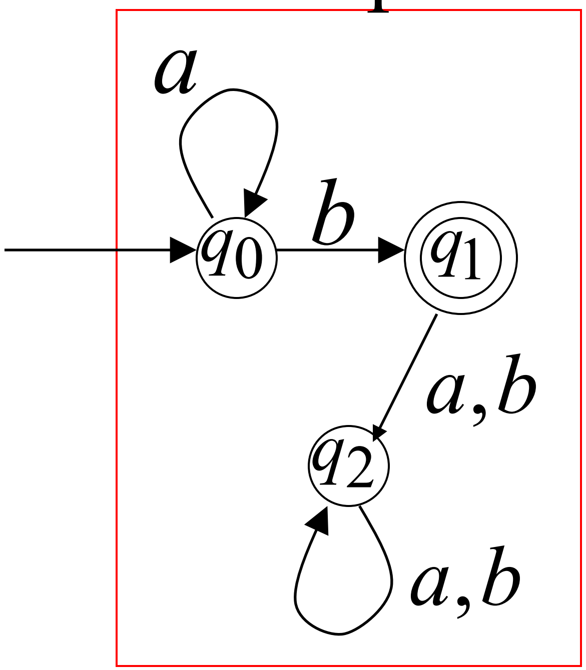

DFA M_1
q_i
accettazione stato
DFA M_2
p_j
p_k
accettazione stati
DFA M
q_i, p_j
q_i, p_k
nuovo accettazione stati

esempio:

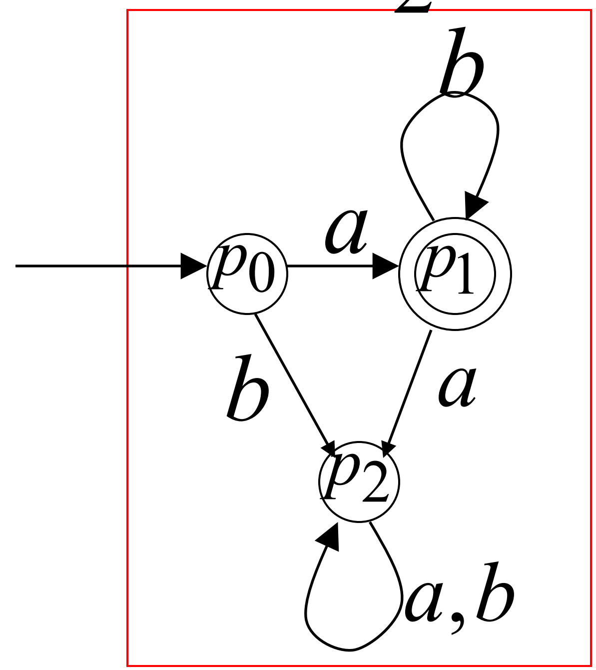
$$L_1 = \{a^n b\} \quad n \geq 0$$

$$L_2 = \{ab^m\} \quad m \geq 0$$

M_1

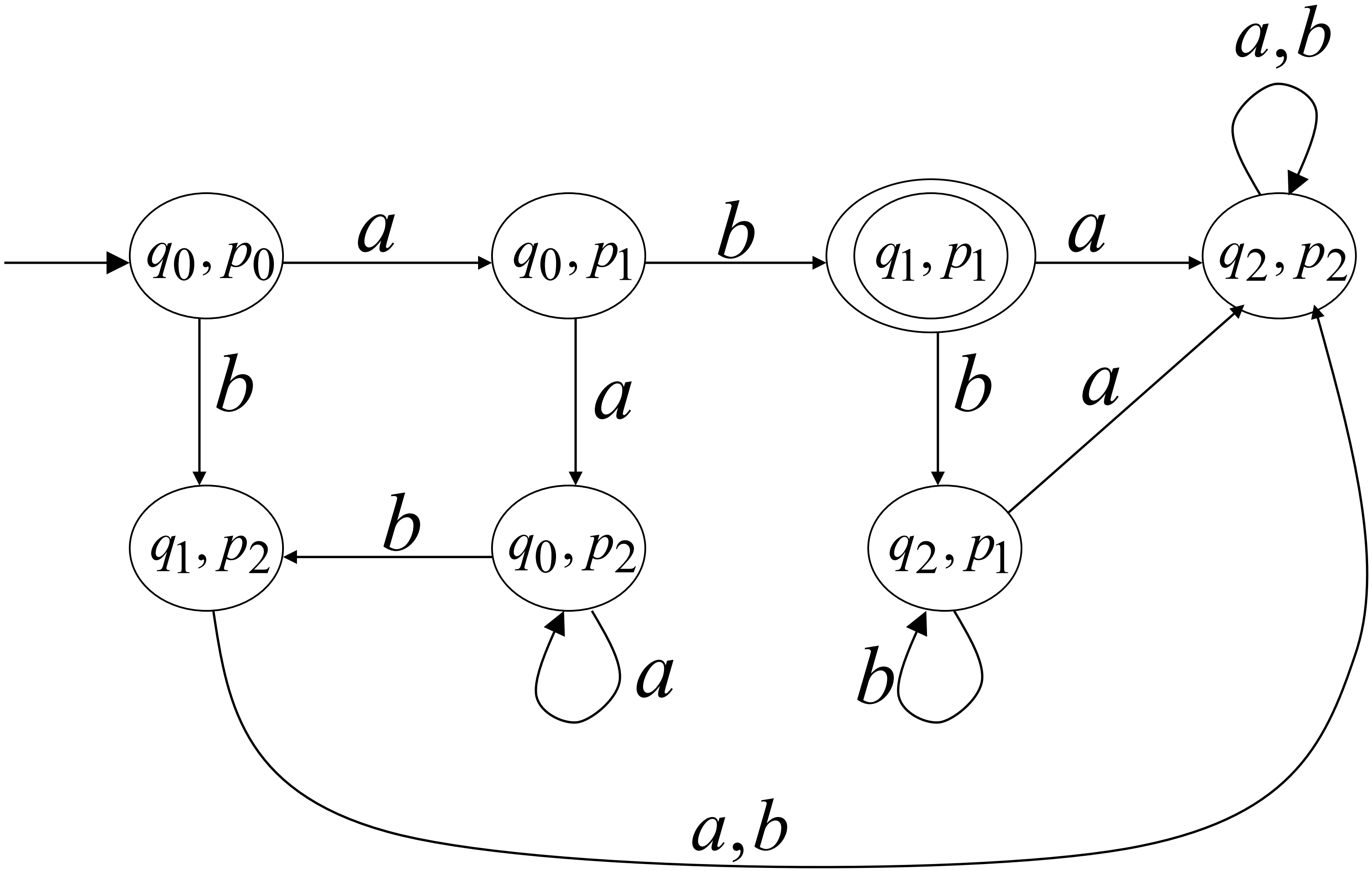


M_2



Intersezione automata

$$L = \{a^n b\} \cap \{ab^n\} = \{ab\}$$



Se appariene ad entrambi

Sia la stringa di lunghezza n
Esistono due cammini di lunghezza n, uno per ogni automa. Dallo stato iniziale a quello finale.
Se lng 1 vero, dimostrare vero per n+1.
Ultimo tratto da n a n+1 arco nei due automa e arco automa costruito. Considera stringa n e considera come stato finale quello prima dello stato finale, vedi arco che riconosce il carattere n, simula i due automi. Continua ad andare indietro fino a raggiungere lo stato iniziale.

Nell'automa costruito esiste un cammino di lng n che li simula

M Simula in parallelo M_1 e M_2

M accettazione stringa w Se e solo se:

M_1 accetta w string

e M_2 accetta w string

$$L(M) = L(M_1) \cap L(M_2)$$

Espressioni regolari e linguaggi regolari

Teorema

$$\left\{\begin{array}{l}\text{Linguaggi}\\ \text{Generati da}\\ \text{Espressioni regolari}\end{array}\right\} = \left\{\begin{array}{l}\text{Linguaggi}\\ \text{regolari}\end{array}\right\}$$

Dimostrazione - Parte 1

$$\left\{\begin{array}{l}\text{Linguaggi}\\ \text{Generati da}\\ \text{Espressioni regolari}\end{array}\right\} \subseteq \left\{\begin{array}{c}\text{Linguaggi}\\ \text{regolari}\end{array}\right\}$$

per ogni espressione regolare r

il linguaggio $L(r)$ è regolare

Dimostrazione per induzione sulla lunghezza

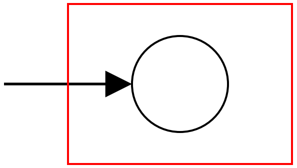
r

Base induzione

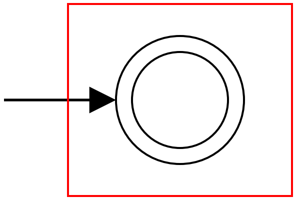
Espressioni regolari di base: $\emptyset, \quad \lambda, \quad \alpha$

corrispondente

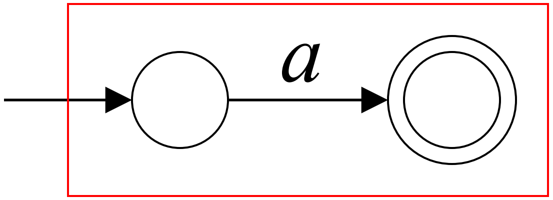
NFAs



$$L(M_1) = \emptyset = L(\emptyset)$$



$$L(M_2) = \{\lambda\} = L(\lambda)$$



$$L(M_3) = \{a\} = L(a)$$

Linguaggi regolari

Ipotesi induttiva

supponi

Per le espressioni regolari r_1 e r_2,

$L(r_1)$ e $L(r_2)$ sono linguaggi regolari.

Esistono due automi uno per ogni linguaggio

Passo induttivo

Proviamo che:

$$L(r_1 + r_2)$$

$$L(r_1 \cdot r_2)$$

$$L(r_1 *)$$

$$L((r_1))$$

Sono linguaggi regolari

Ricorda che, per def. di espressione regolare

$$L(r_1 + r_2) = L(r_1) \cup L(r_2)$$

$$L(r_1 \cdot r_2) = L(r_1)\,L(r_2)$$

$$L(r_1 *) = (L(r_1))*$$

$$L((r_1)) = L(r_1)$$

Per ipotesi induttiva :

$L(r_1)$ e $L(r_2)$ sono linguaggi regolari

Inoltre sappiamo, slides precedenti:

I linguaggi regolari sono chiusi rispetto:

Unione $L(r_1) \cup L(r_2)$

Concatenatzione $L(r_1)L(r_2)$

Star $(L(r_1))^*$

Usando la chiusura dele operazioni
Possiamo costruire un NFA M tale che:

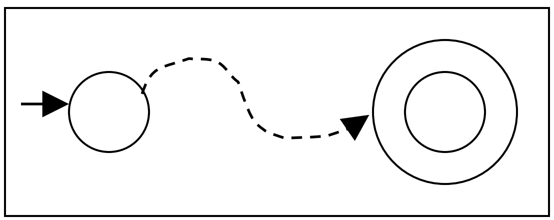
$$L(M) = L(r)$$

---

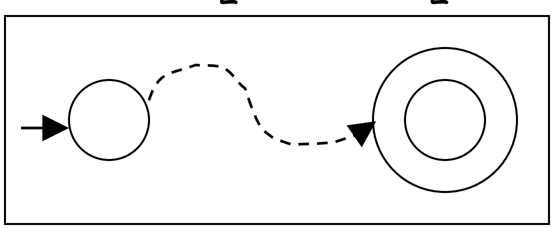
esempio: $r = r_1 + r_2$

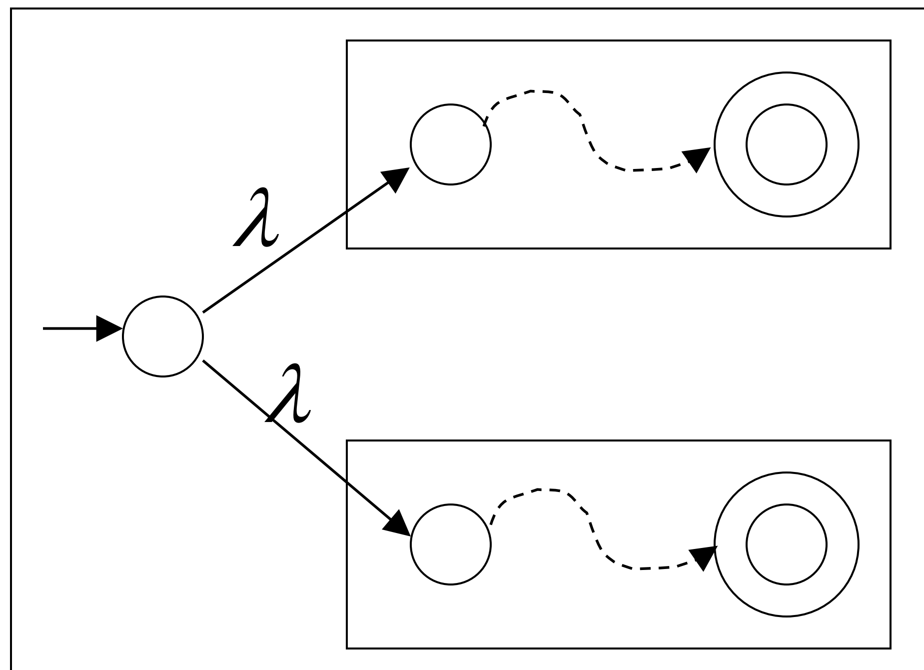
$$L(M) = L(r)$$

$$L(M_1) = L(r_1)$$



$$L(M_2) = L(r_2)$$





Stella e puntino. esercizio

Stella: torna indietro con lambda.
Puntino: collega i finali del primo con l'iniziale con un lambda.

dim - Part 2

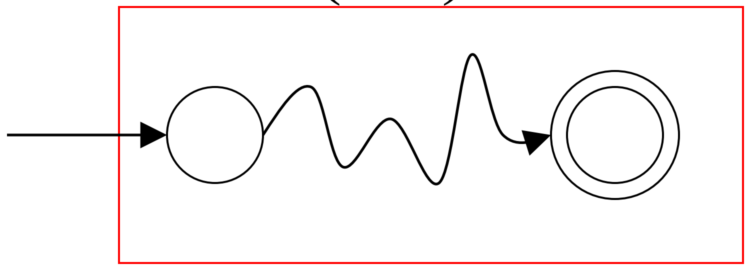
$$\left\{\begin{array}{l}\text{Linguaggi}\\\text{Generati da}\\\text{Espressioni regolari}\end{array}\right\} \supseteq \left\{\begin{array}{l}\text{Linguaggi}\\\text{regolari}\end{array}\right\}$$

Per ogni linguaggio regolare L esiste
una espressione regolare r con $L(r) = L$

Convertiremo un NFA che accetta L
In una espressione regolare

Poichè L è regolare , allora esiste un NFA M che lo accettà

$$L(M) = L$$



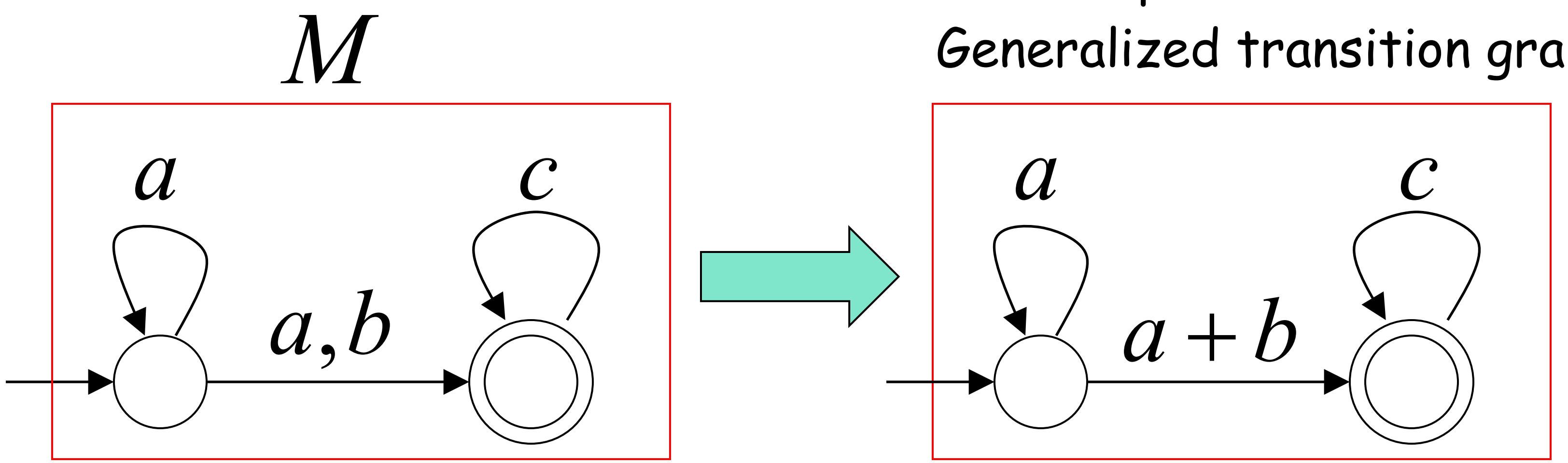
Prendiamo l'automa con un solo stato finale da M costruiamo l'equivalente

Generalized Transition Graph

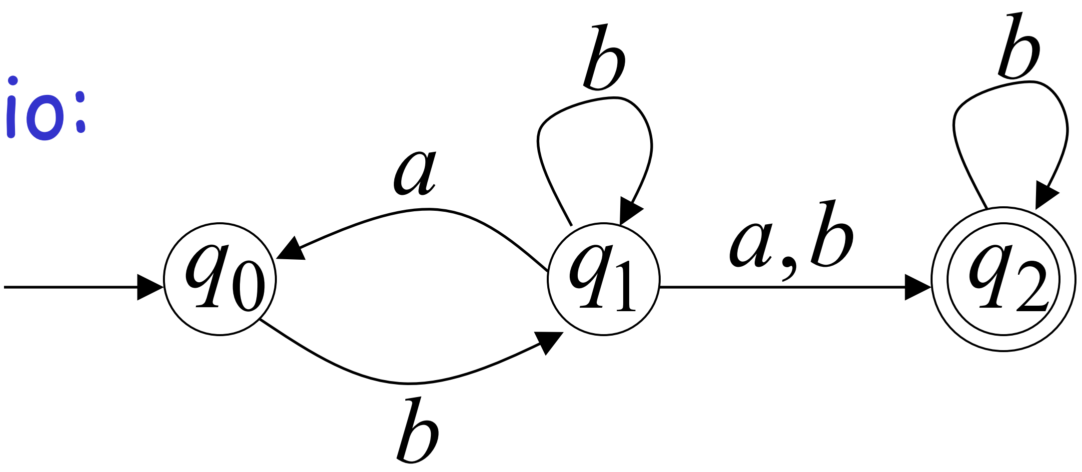
Nel quale i caratteri di transizione, transition labels, sono espressioni regolari

Esempio:

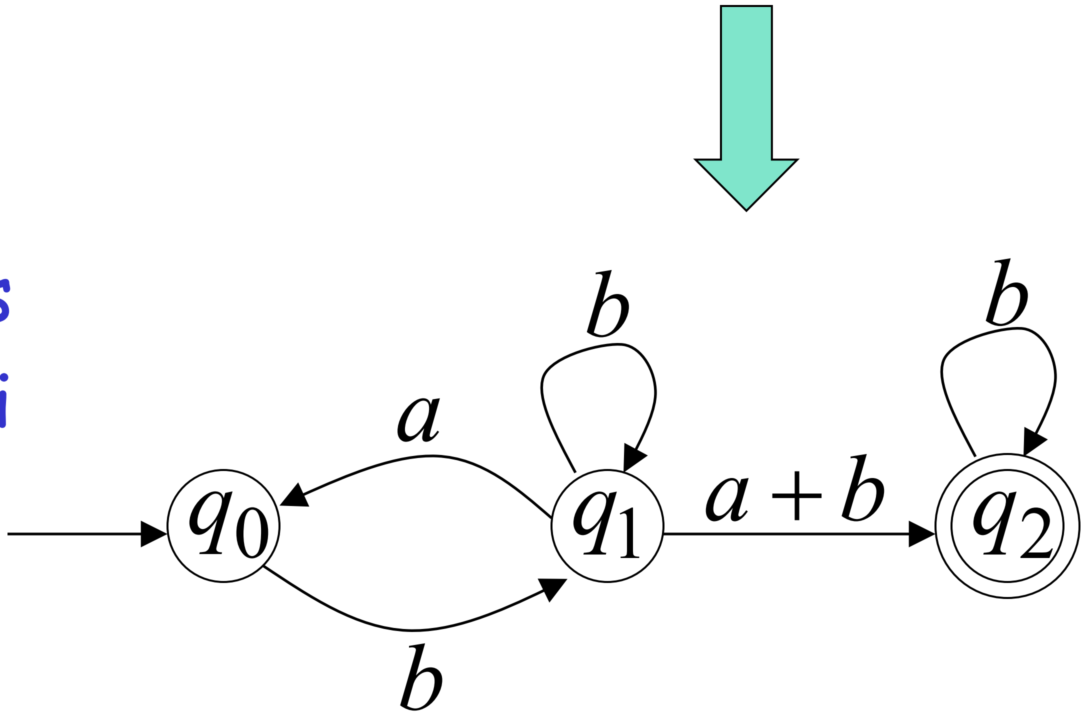
Il corrispondente
Generalized transition graph



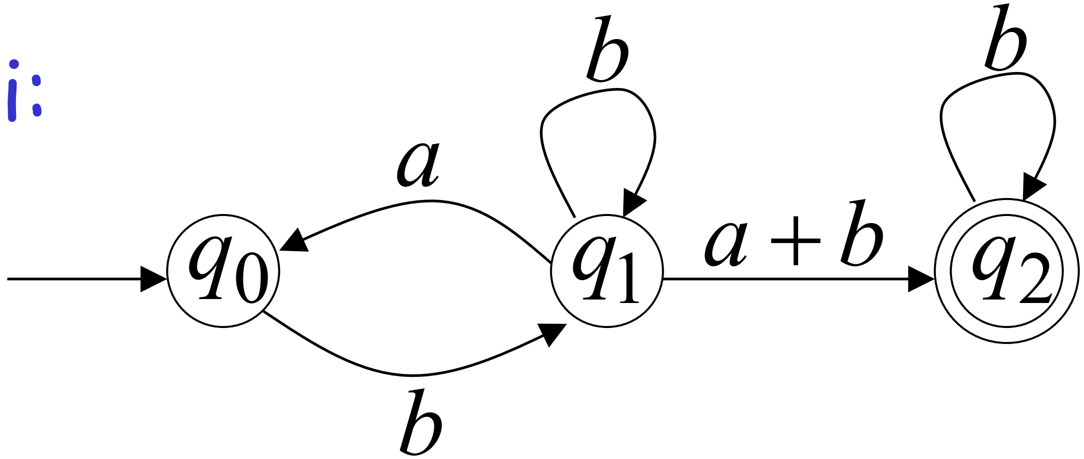
Un altro esempio:



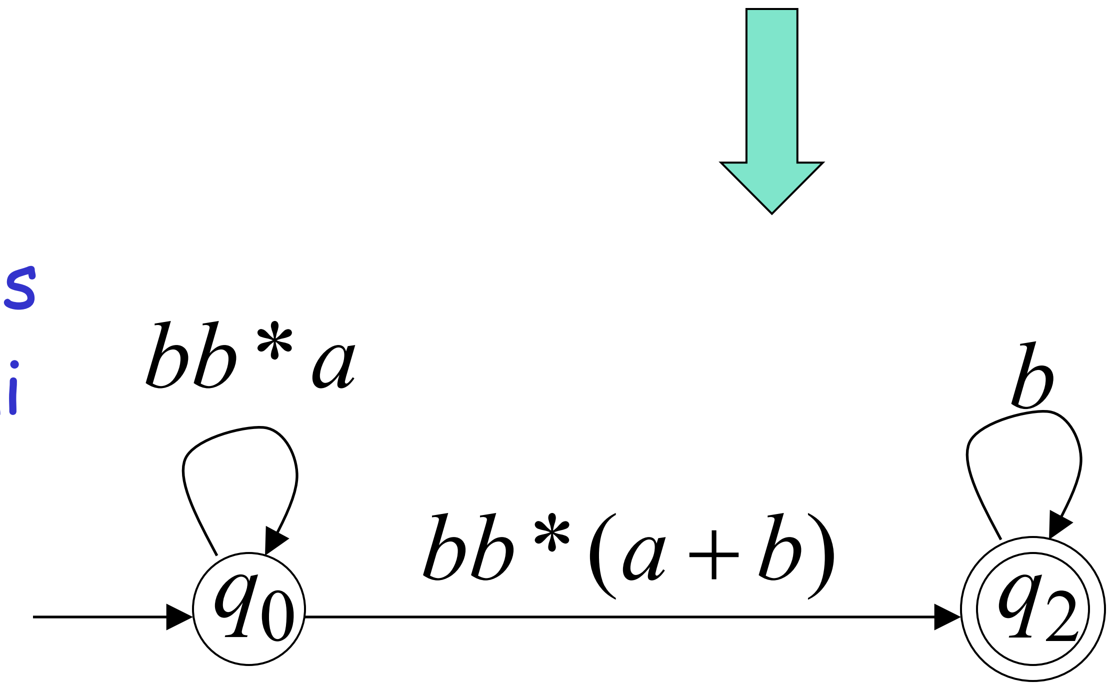
Transition labels
Sono espressioni
regolari



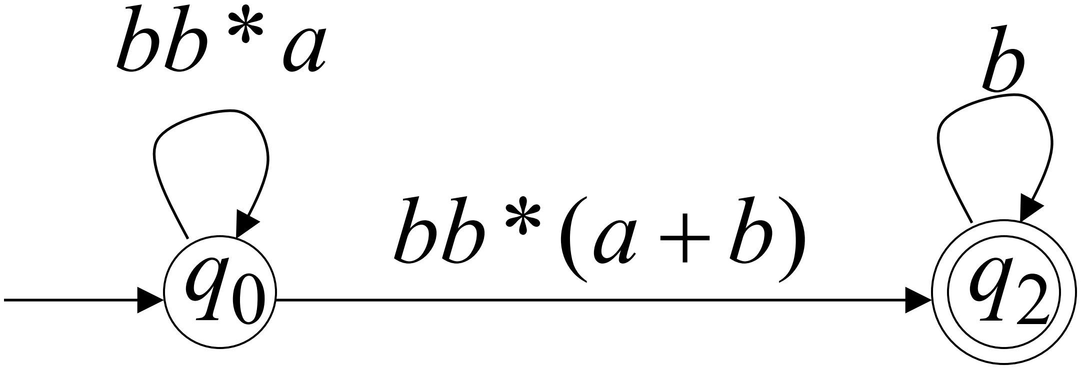
Ridurre gli stati:



Transition labels sono espressioni regolari



Espressione regolare che si ottiene:



$$r = (bb^*a)^*bb^*(a+b)b^*$$

$$L(r) = L(M) = L$$

Stato iniziale solo archi uscenti, nessuno rientrante

Solo uno finale tutti entranti e nessun uscente.

Per gli altri stati sono presenti archi uscenti per tutti gli altri stati ed entranti da tutti gli altri stati e su se stesso. Se non esiste un arco da q_i a q_j creiamo un arco con label insieme vuoto ϕ

Se k=2 slide precedente

Altimenti

prendiamo lo stato da eliminare q

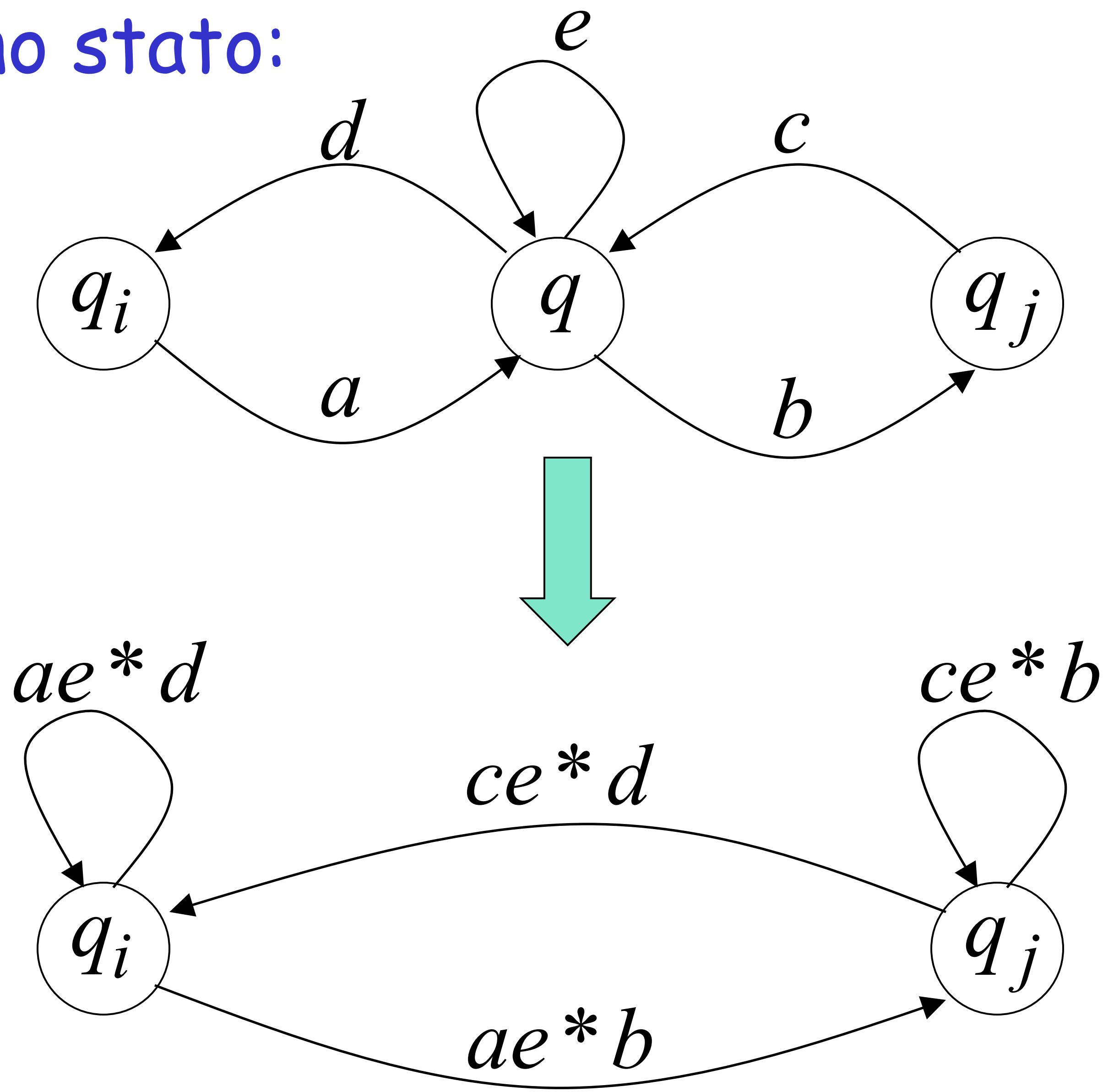
Per ogni q_i e q_j collegati via q

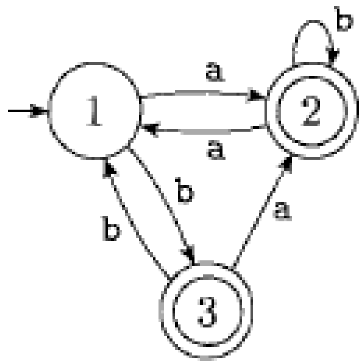
$$\delta^*\left(q\_1, q\_j\right) = (R\_1)(R\_2)^*(R\_3) \cup (R\_4)$$

vai da q_i a q, R_1
gira su q, (R_2)*
vai da q a q_j, R_3
direttamente da q_i a q_j, R_4

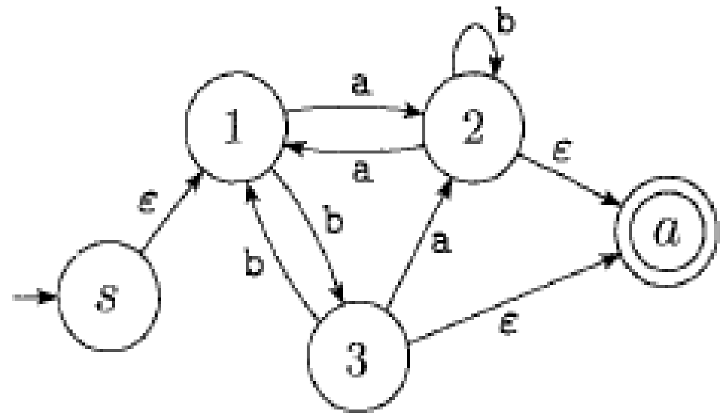
In generale

Rimuovere uno stato:



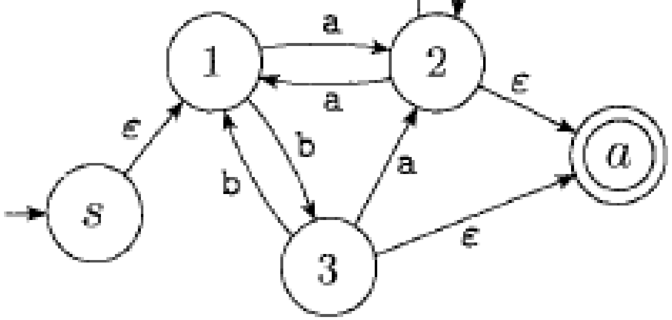


(a)

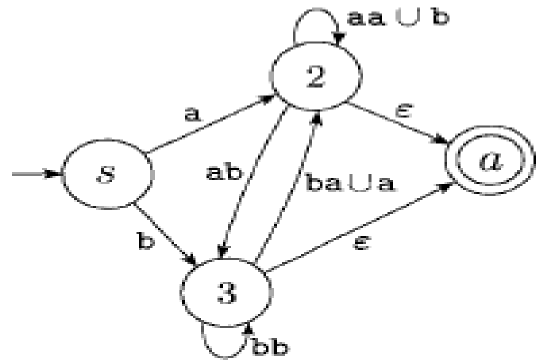


(b)

Per ogni q_i e q_j collegati via q

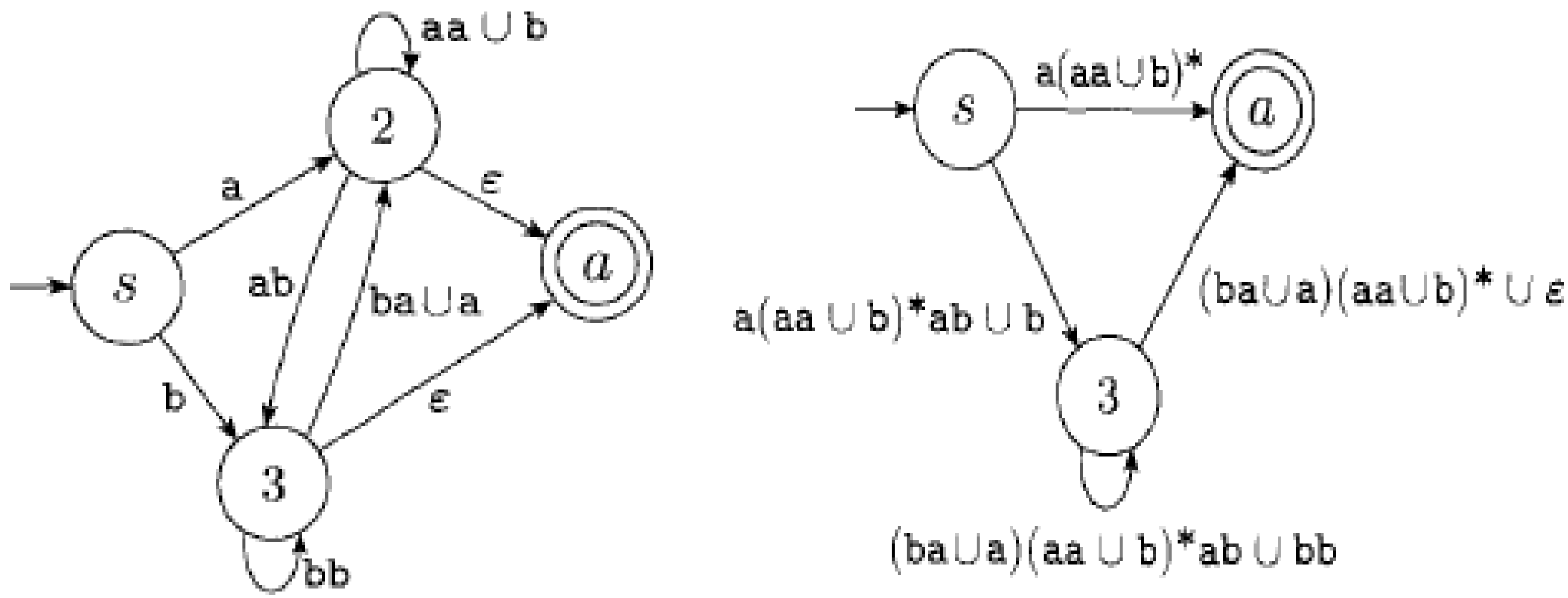


(b)

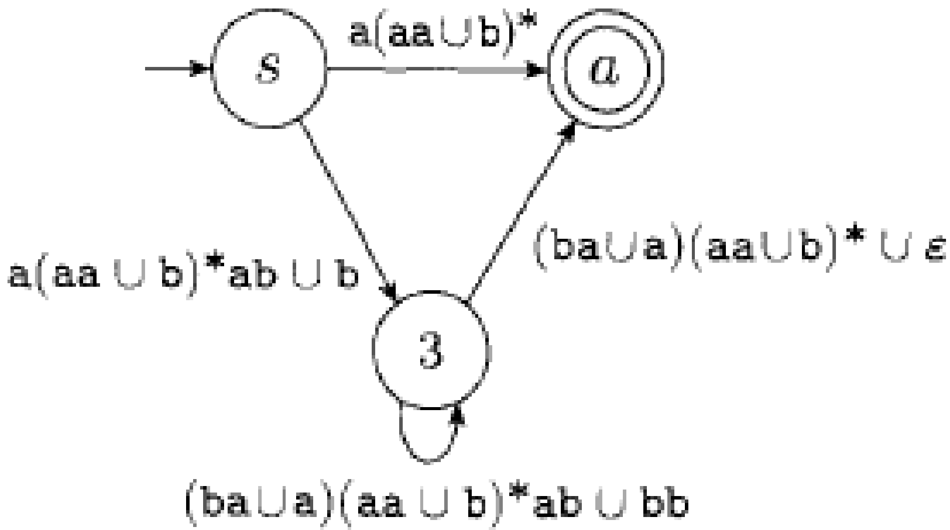


vai da q_i a q, R_1
gira su q, (R_2)*
via da q a q_j, R_3
direttamente da q_i a q_j, R_4

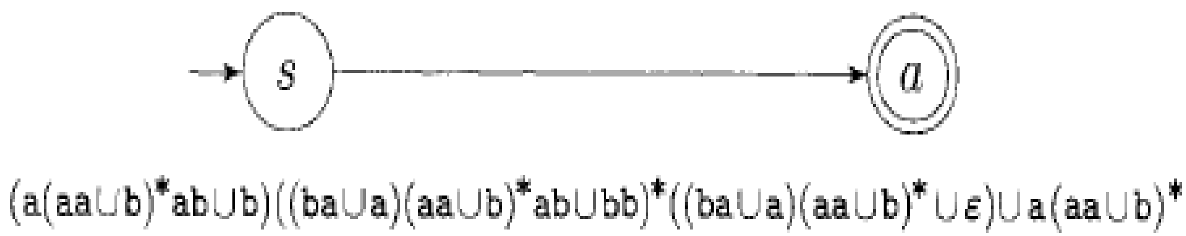
$$\delta^*(q\_1, q\_j) = (R\_1)(R\_2)^*(R\_3) \cup (R\_4)$$



vai da q_i a q, R_1
gira su q, (R_2)*
vai da q a q_j, R_3
direttamente da q_i a q_j, R_4



a(aa∪b)*
a(aa ∪ b)*ab ∪ b
(ba∪a)(aa∪b)* ∪ ε
(ba∪a)(aa ∪ b)*ab ∪ bb



(a(aa∪b)*ab∪b)((ba∪a)(aa∪b)*ab∪bb)*((ba∪a)(aa∪b)*∪ε)∪a(aa∪b)*

Algoritmo: Eliminare uno stato alla volta fino a che restano 2 stati.

Dimostrazione algoritmo funziona ovvero l'automa iniziale G e G', meno uno stato, accettano lo stesso linguaggio

G' con due stati allora otteniamo espressione regolare.

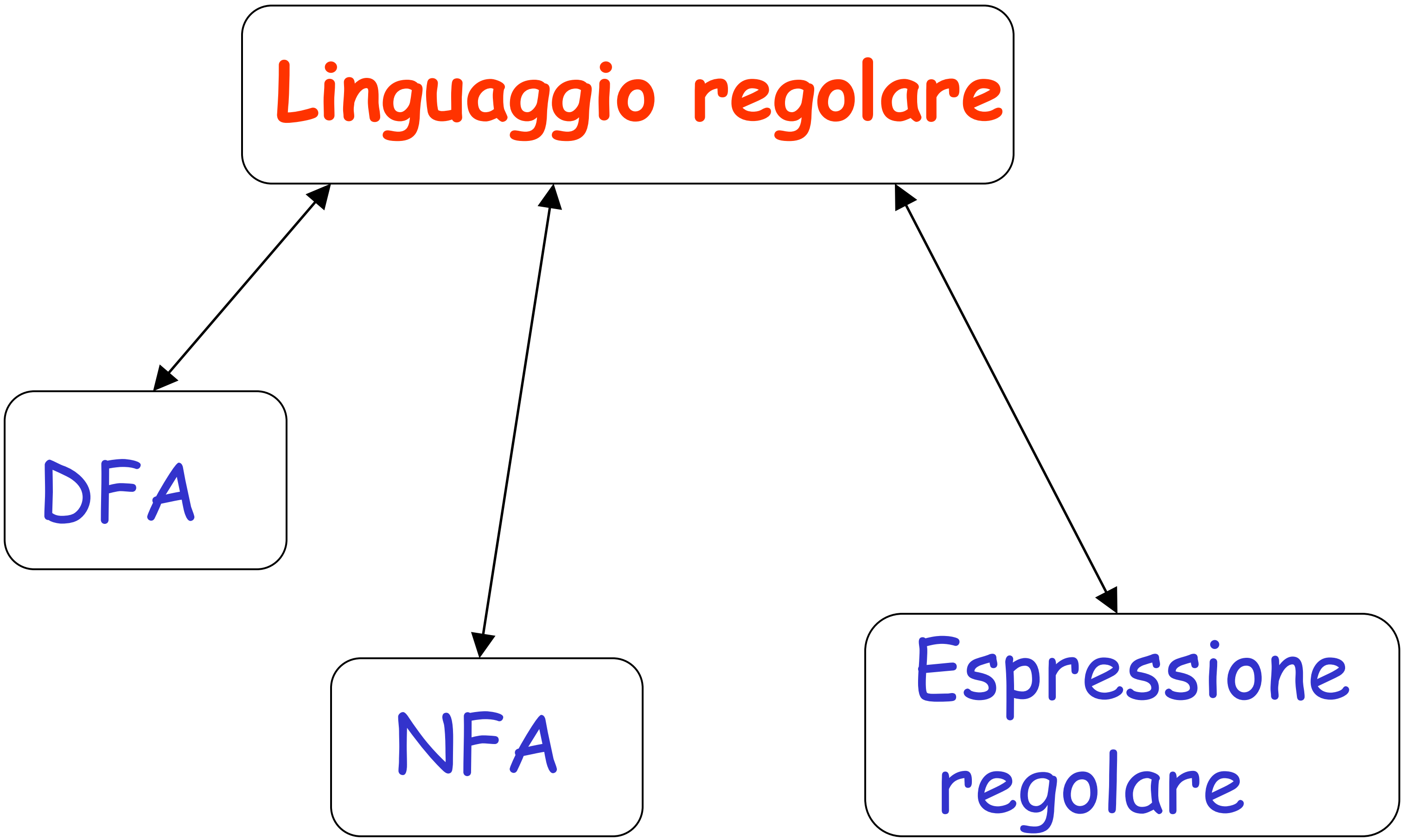
Vero per k-1 provare per k+1.

Prendiamo una stringa che viene accettata, esiste un cammino che accetta la stringa se non usa lo stato da eliminare bene *G* e *G'* accettano la stringa. In slang:

«Se *G* non usa lo stato da eliminare: bene *G'* «si fa lo stesso giro» »;

Se *G* usa lo stato da eliminare allora in *G'* lo stato in oggetto non esiste ma nei nuovi archi tutte le sottostringhe che venivano riconosciute tramite lo stato eliminato sono descritte dalle espressioni regolari sugli archi

Presentazione standard di un linguaggio regolare



esercizi
slide successive

esempio

Epressione regolare: $(a+b)\cdot a^*$

$$
\begin{aligned}
L((a+b)\cdot a^*) &= L((a+b))\,L(a^*) \\
&= L(a+b)\,L(a^*) \\
&= (L(a)\cup L(b))\,(L(a))^* \\
&= (\{a\}\cup\{b\})\,(\{a\})^* \\
&= \{a,b\}\,\{\lambda,a,aa,aaa,\ldots\} \\
&= \{a,aa,aaa,\ldots,b,ba,baa,\ldots\}
\end{aligned}
$$

Linguaggi associati alle espressioni regolari

$L(r)$: linguaggio associato all'espressione r

esempio

$$L((a+b\cdot c)^*)=\{\lambda, a, bc, aa, abc, bca, \ldots\}$$

Esempio

Espressione regolare $r = (a+b)^*(a+bb)$

$$L(r) = \{a, bb, aa, abb, ba, bbb, ...\}$$

esempio

Espressione regolare $r = (aa)^*(bb)^*b$

$$L(r) = \{a^{2n}b^{2m}b : \quad n, m \geq 0\}$$

esempio

Espressione regolare $r = (0+1)^{*}\,00\,(0+1)^{*}$

$L(r)$ = { tutte le stringhe che contengono 00 }

esempio

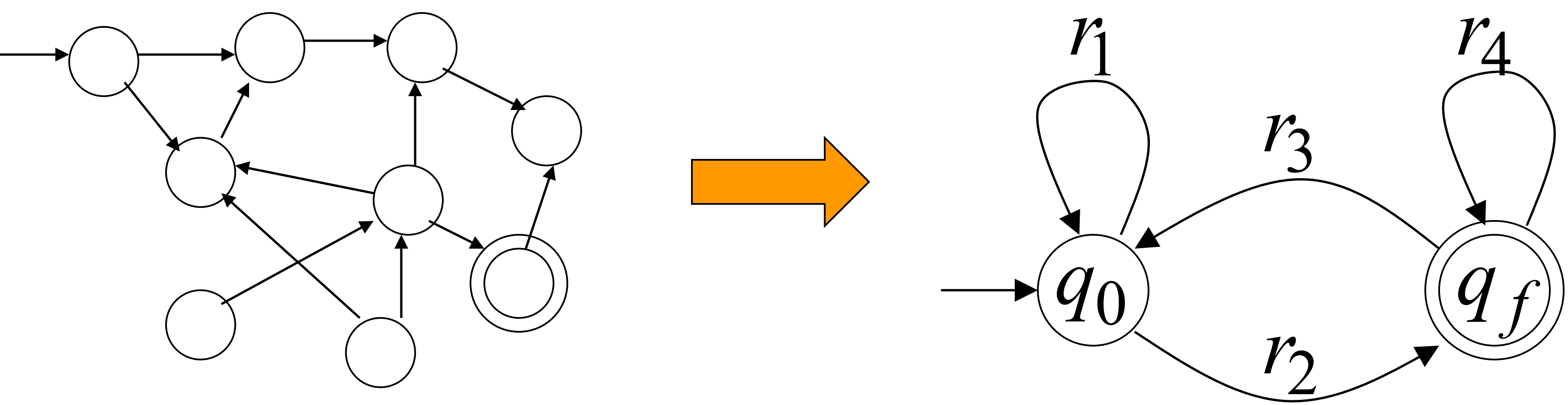
Espressione regolare $r = (1 + 01)^* (0 + \lambda)$

$L(r)$

= { tutte le stringhe senza sottostringhe 00 }

Ripetere il processo finchè
Due stati restano il grafo risultante
sarà il seguente

Grafo iniziale | Grafo risultante



L'espressione regolare risultante:

$$r = r_1{}^* r_2 (r_4 + r_3 r_1{}^* r_2)^*$$

$$L(r) = L(M) = L$$

Definizione formale

Grammatica:

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

Insieme delle variabili

Insieme simboli terminali

Start variabile

Insieme delle produzioni

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

Tutte le produzioni P sono della forma:

$$A \to s$$

Stringhe di
Variabili e non
terminali

Linguaggio di una grammatica:

Per una grammatica G con start S

$$L(G) = \{w : \quad S \overset{*}{\Longrightarrow} w, \quad w \in T^*\}$$

Stringhe di terminali o λ

terminali

Grammatica lineare

Le grammatiche con al massimo una variabile sul lato destro della produzione

Esempio:

$$S \to aSb \qquad S \to Ab$$

$$S \to \lambda \qquad A \to aAb$$

$$A \to \lambda$$

Grammatica non lineare

Grammatica G :

$$S \to SS$$

$$S \to \lambda$$

$$S \to aSb$$

$$S \to bSa$$

$$L(G) = \{w: \quad n_a(w) = n_b(w)\}$$

Numeri di a nella stringa w

Grammatica lineare

Grammatica G :

$$S \to A$$

$$A \to aB \mid \lambda$$

$$B \to Ab$$

$$L(G) = \{a^n b^n : n \geq 0\}$$

Grammatica lineare a destra

Tutte le produzioni hanno la forma

$$A \to xB$$

o

$$A \to x$$

Stringa di terminali

esempio:

$$S \to abS$$

$$S \to a$$

Grammatiche lineare sinistra

Tutte le produzioni hanno la forma:

$$A \to Bx$$

o

$$A \to x$$

Stringhe di terminali

Esempio:

$$S \to Aab$$

$$A \to Aab \mid B$$

$$B \to a$$

Grammatica regolare

Grammatiche regolari

Una grammatica regolare è qualsiasi grammatica lineare a destra o a sinistra

Esempio:

G_1

$$S \to abS$$

$$S \to a$$

G_2

$$S \to Aab$$

$$A \to Aab \mid B$$

$$B \to a$$

I linguaggi generati da una grammatica regolare è un linguaggio regolare

Examples:

$$G_1$$

$$S \to abS$$

$$S \to a$$

$$G_2$$

$$S \to Aab$$

$$A \to Aab \mid B$$

$$B \to a$$

$$L(G_1) = (ab)^* a$$

$$L(G_2) = aab(ab)^*$$

Grammatiche regolari generano linguaggi regolari

Teorema

$$\left\{ \begin{array}{l} \text{Linguaggi} \\ \text{Generati da} \\ \text{grammatiche} \\ \text{regolari} \end{array} \right\} = \left\{ \begin{array}{l} \text{Linguaggi} \\ \text{regolari} \end{array} \right\}$$

Teorema - Part 1

$$\left\{\begin{array}{l}\text{Linguaggi} \\ \text{Generati da} \\ \text{grammatiche} \\ \text{regolari}\end{array}\right\} \subseteq \left\{\begin{array}{l}\text{Linguaggi} \\ \text{regolari}\end{array}\right\}$$

Ogni grammatica regolare
Genera un liguaggio generale

Teorema - Part 2

$$\left\{ \begin{array}{l} \text{Linguaggi} \\ \text{Generati da} \\ \text{grammatiche} \\ \text{regolari} \end{array} \right\} \supseteq \left\{ \begin{array}{l} \text{Linguaggi} \\ \text{regolari} \end{array} \right\}$$

Ogni linguaggio regolare
È generato da una grammatica regolare

Proof - Part 1

$$\left\{ \text{Linguaggi Generati da grammatiche regolari} \right\} \subseteq \left\{ \text{Linguaggi regolari} \right\}$$

Il linguaggio $L(G)$ generato da
Una grammatica regolare G è regolare

Il caso della Grammatica

sia G una right-linear grammatica

proveremo: $L(G)$ è regolare

idea: costruiamo una NFA M

con $L(M) = L(G)$

Grammatica G è right-linear

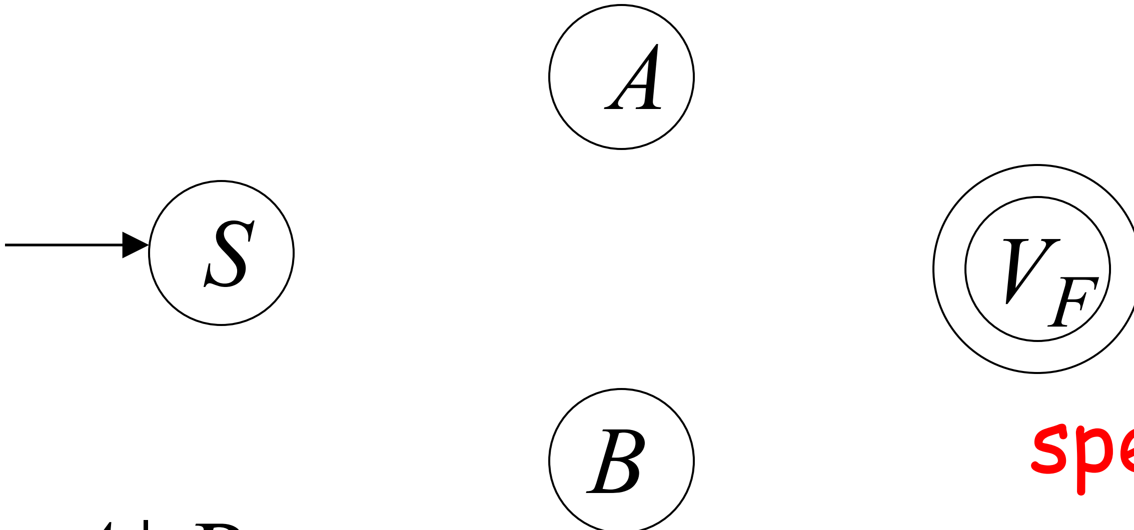
Esempio:

$$S \rightarrow aA \mid B$$

$$A \rightarrow aa\ B$$

$$B \rightarrow b\ B \mid a$$

Construiamo NFA M tale che
ogni stato è una variabile della grammatica :



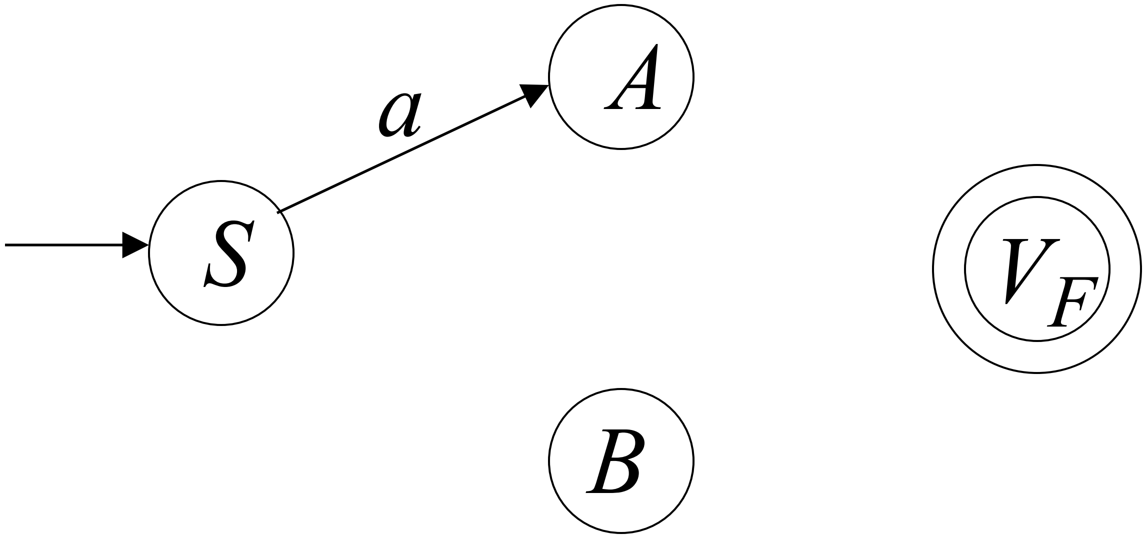
speciale
stato finale

$$S \to aA \mid B$$

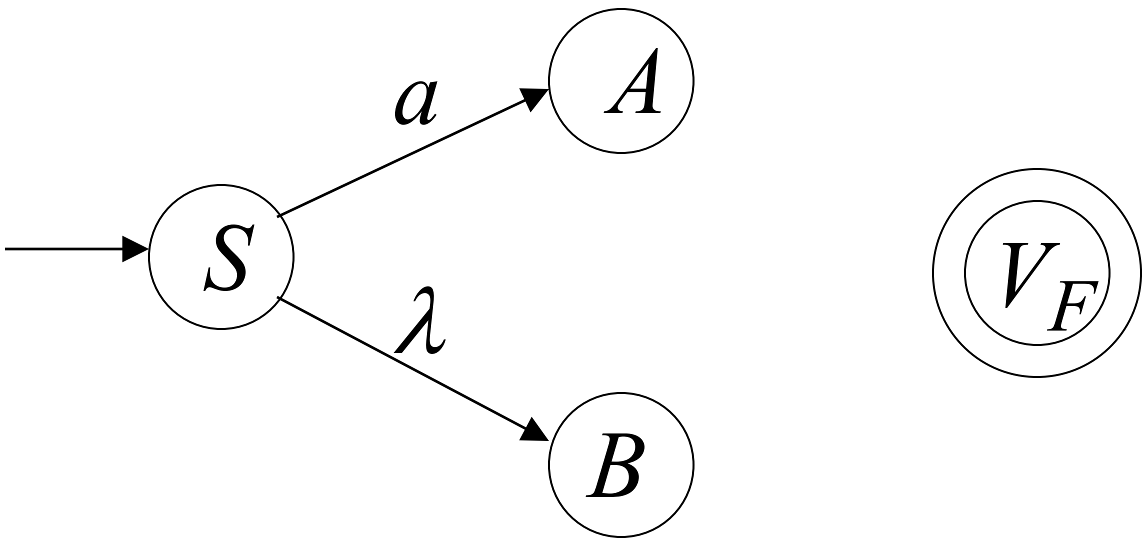
$$A \to aa\ B$$

$$B \to b\ B \mid a$$

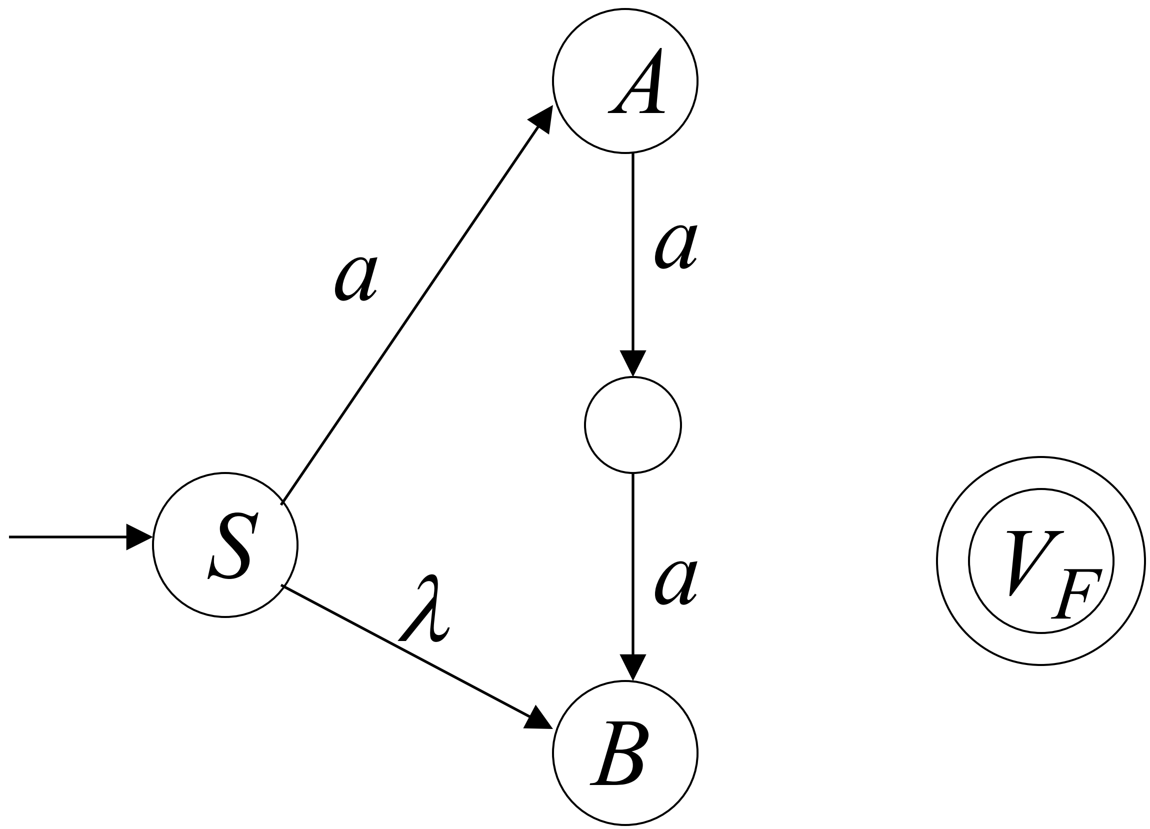
Addizioniamo un arco per ogni produzione:



$$S \rightarrow aA$$

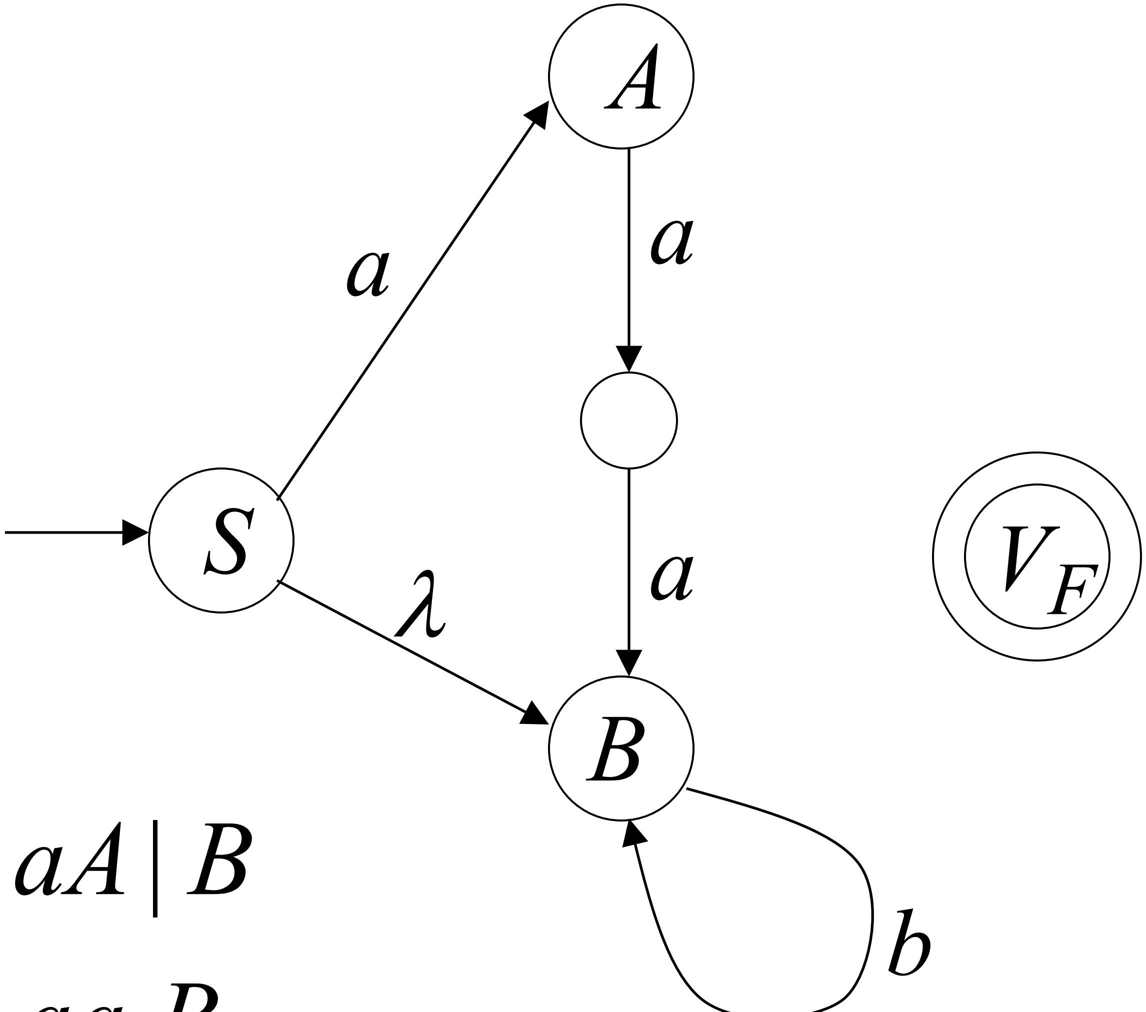


$$S \to aA \mid B$$



$$S \to aA \mid B$$

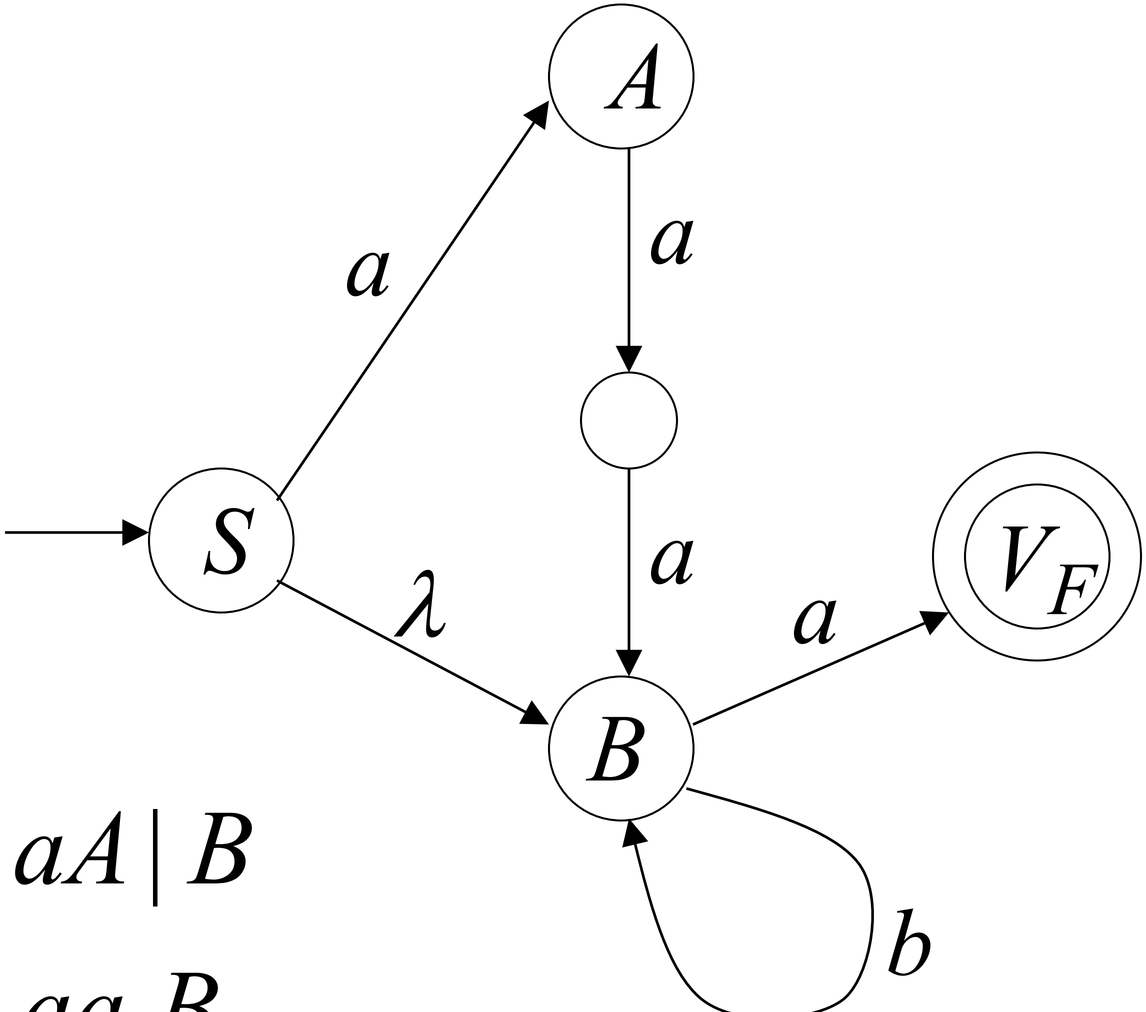
$$A \to aa\ B$$



$$S \to aA \mid B$$

$$A \to aa\ B$$

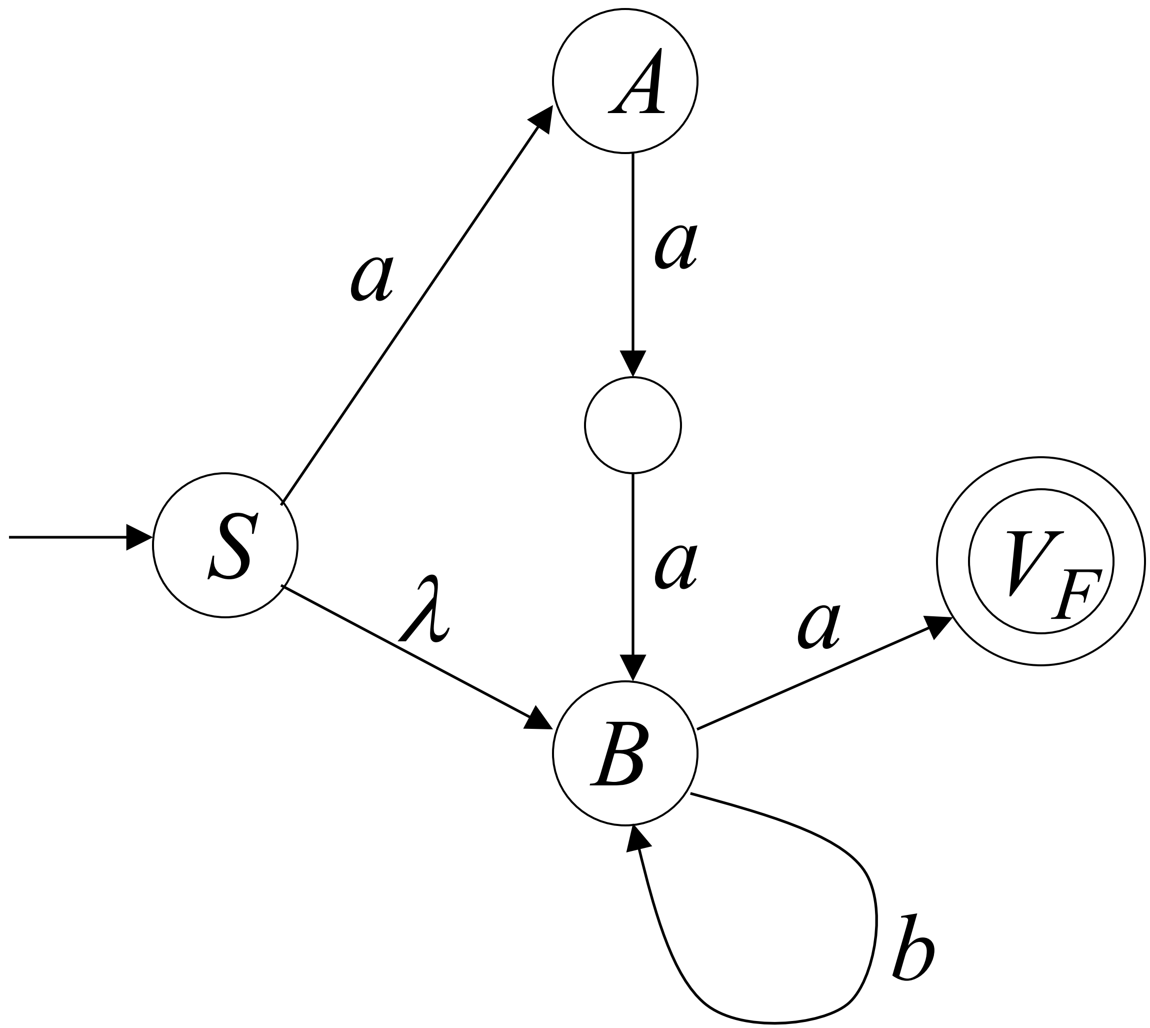
$$B \to bB$$



$$S \to aA \mid B$$

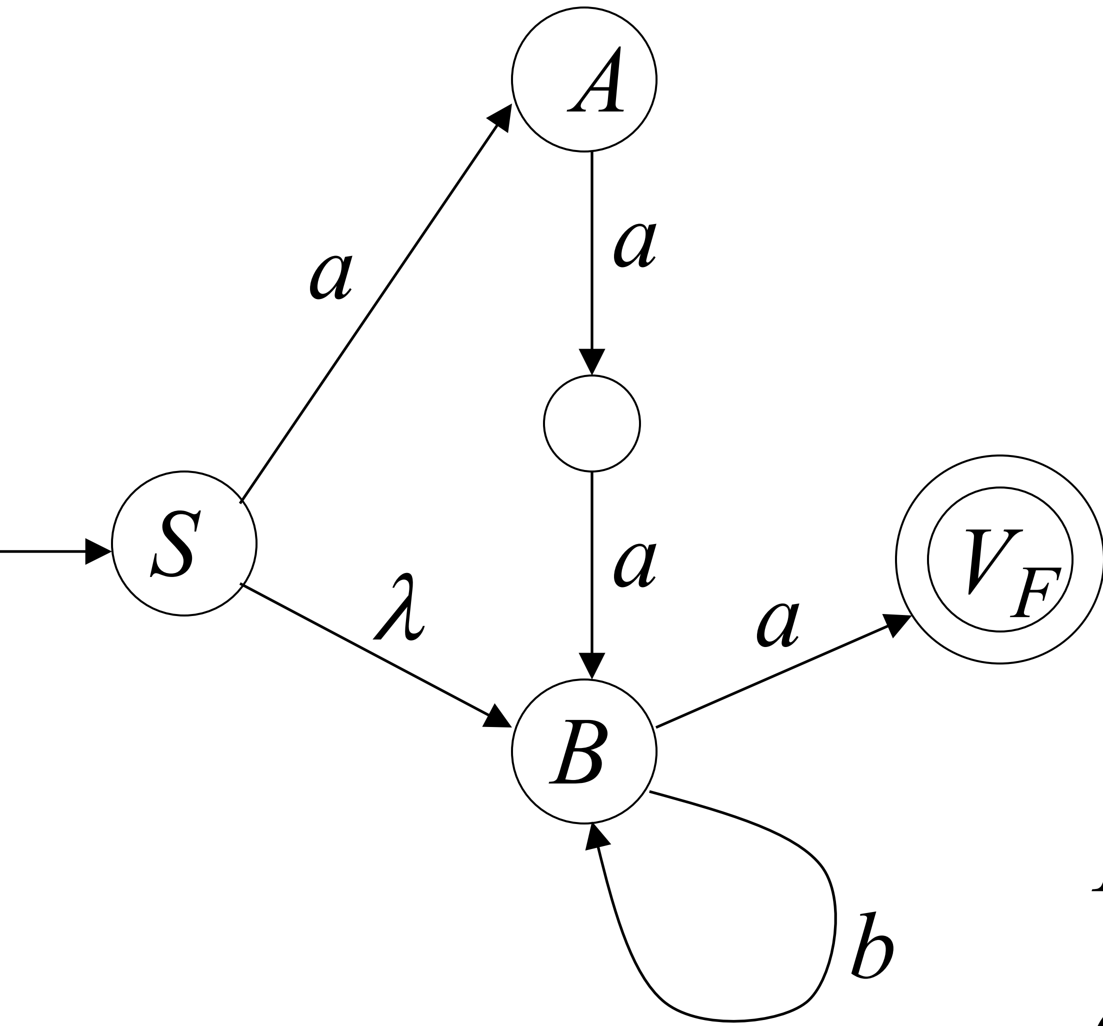
$$A \to aa\ B$$

$$B \to bB \mid a$$



$$S \Rightarrow aA \Rightarrow aaaB \Rightarrow aaabB \Rightarrow aaaba$$

NFA M



Grammatica

G

$$S \to aA \mid B$$

$$A \to aa\ B$$

$$B \to bB \mid a$$

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

$$aaab^*a + b^*a$$

In Generale

una right-linear grammatica G

Ha le variabili: $V_0, V_1, V_2, \dots$

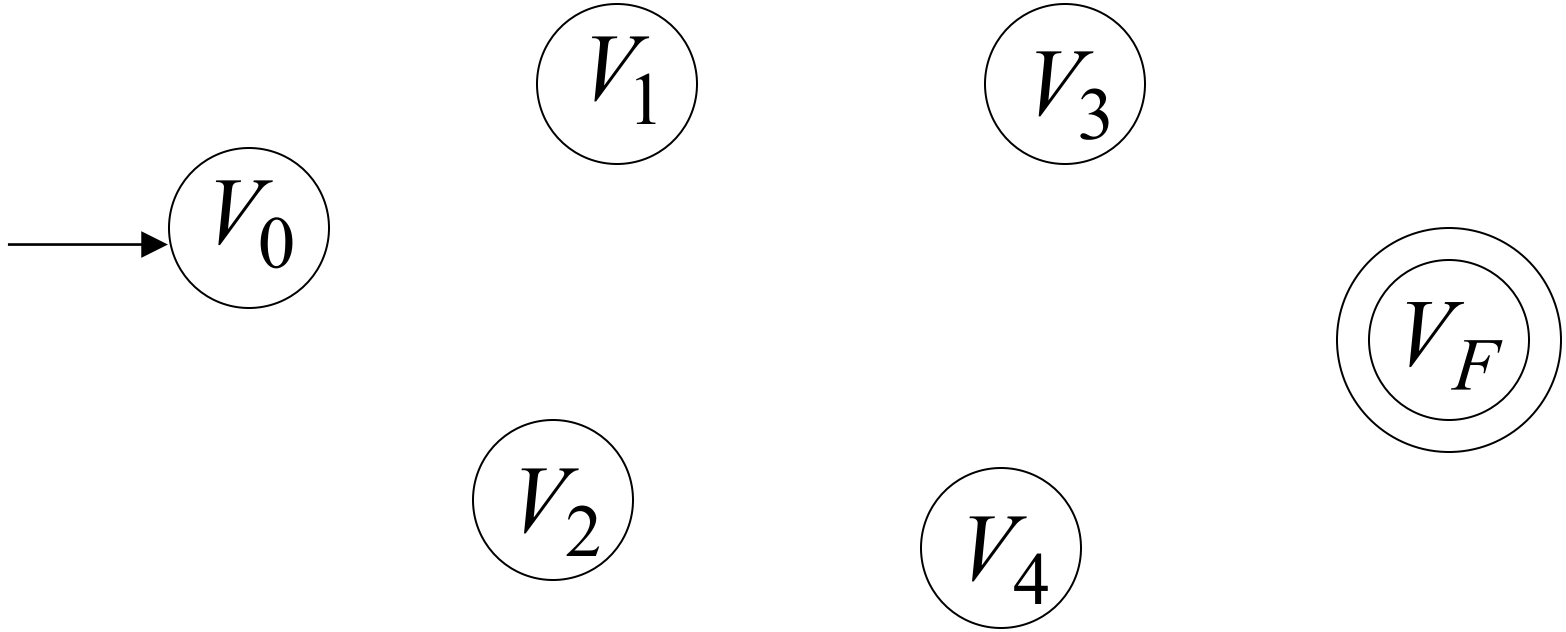
E le produzioni: $V_i \to a_1 a_2 \cdots a_m V_j$

or

$V_i \to a_1 a_2 \cdots a_m$

Costruiamo un NFA M tale che:

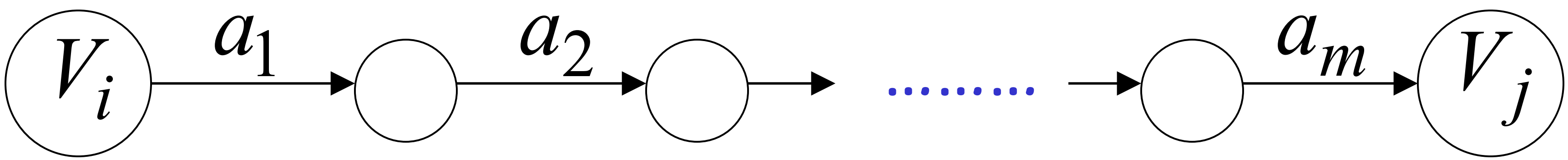
Ogni variabile V_i corrisponde ad un nodo:



stato finale

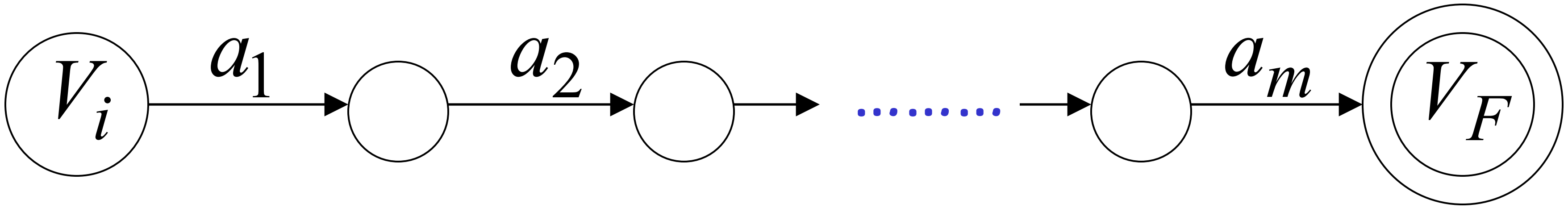
per ogni produzione: $V_i \rightarrow a_1 a_2 \cdots a_m V_j$

Addizioniamo transizioni e nodi intermedi

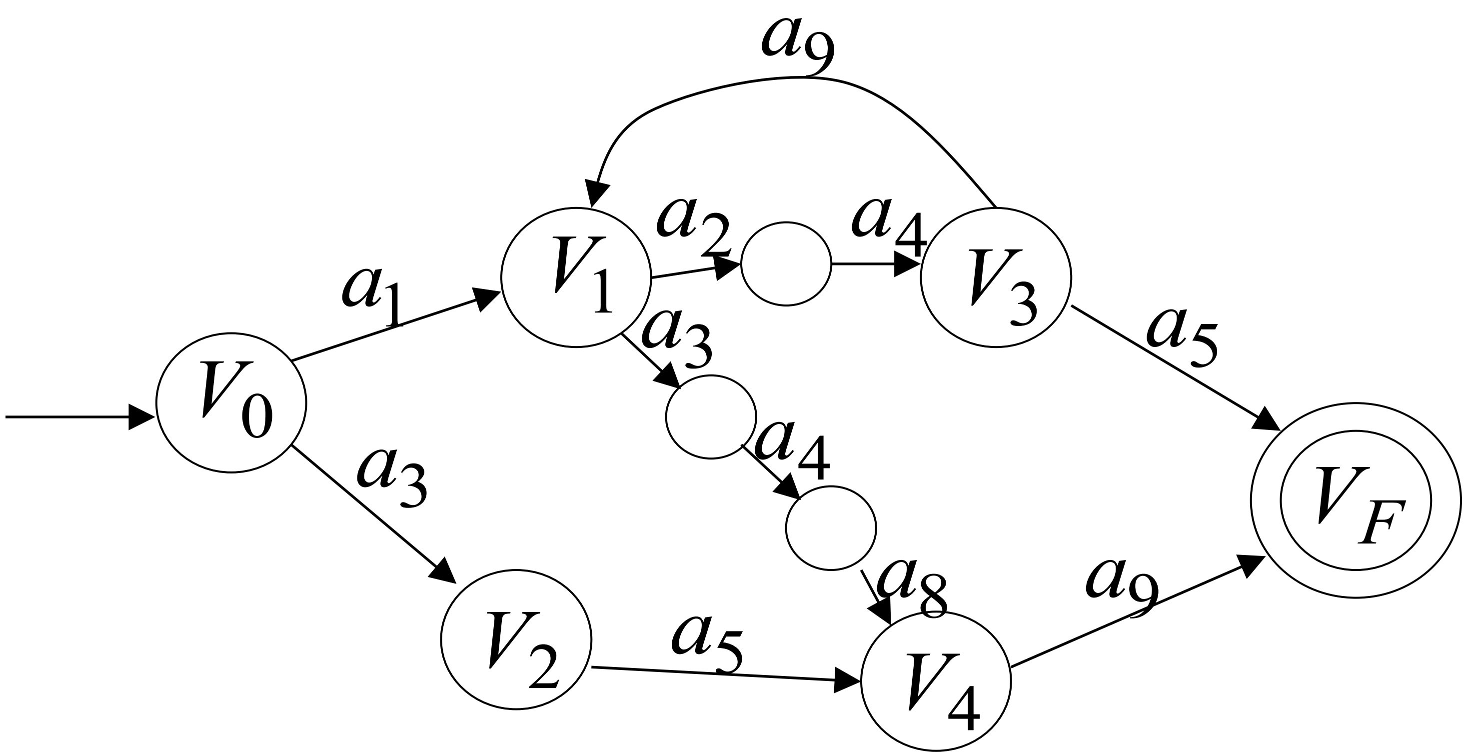


per ogni produzione: $V_i \to a_1 a_2 \cdots a_m$

Addizioniamo transizioni e nodi intermedi



otteniamo NFA M come questo:



Vale che: $$L(G) = L(M)$$

Il caso di una Left-Linear grammatica

Fate voi

dimostrazione - Part 2

$$\left\{ \begin{array}{l} \text{Linguaggi} \\ \text{Generati da} \\ \text{grammatiche} \\ \text{regolari} \end{array} \right\} \supseteq \left\{ \begin{array}{l} \text{Linguaggi} \\ \text{regolari} \end{array} \right\}$$

ogni linguaggio regolare L è generato da qualche grammatica regolare G

qualsiasi linguaggio regolare L è generato da una grammatica regolare G

idea:

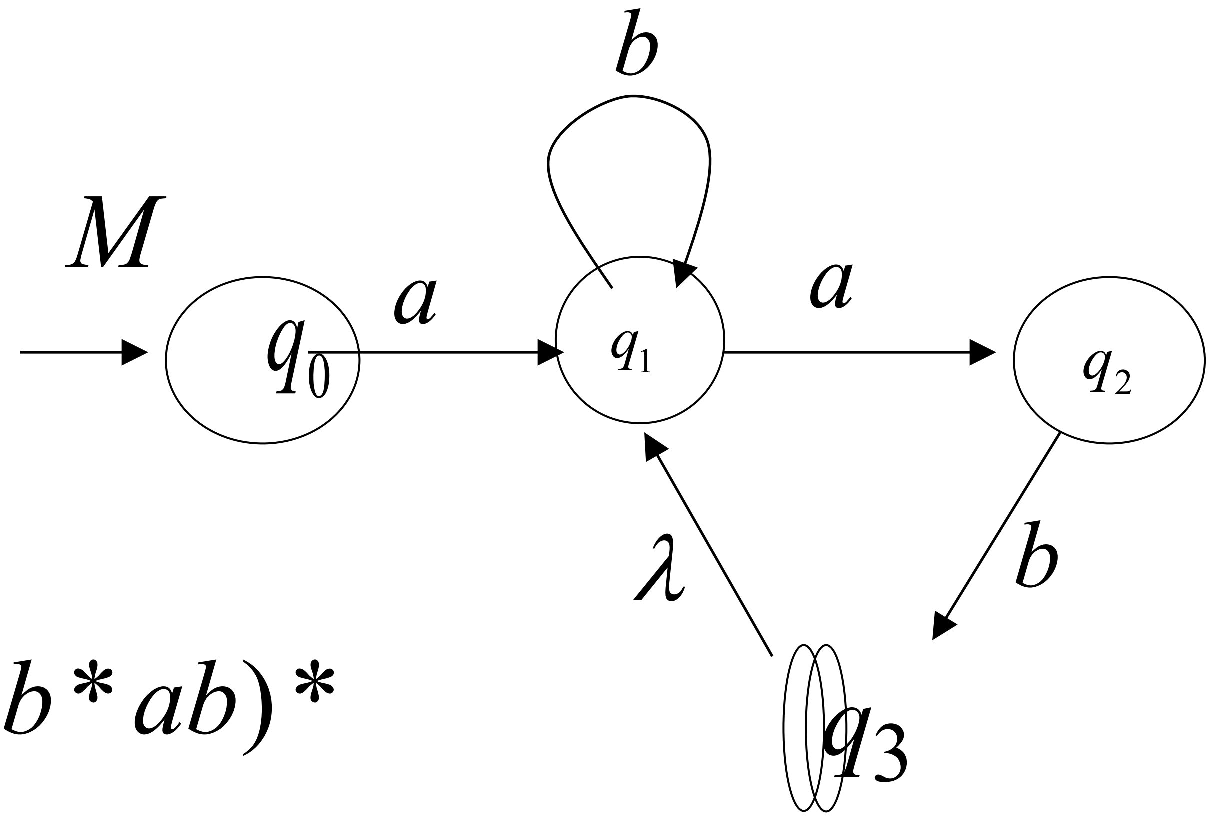
sia M NFA con $L = L(M)$.

costruiamo da M una grammatica G regolare tale che $L(M) = L(G)$

Poichè L è regolare

è un NFA M tale che $L = L(M)$

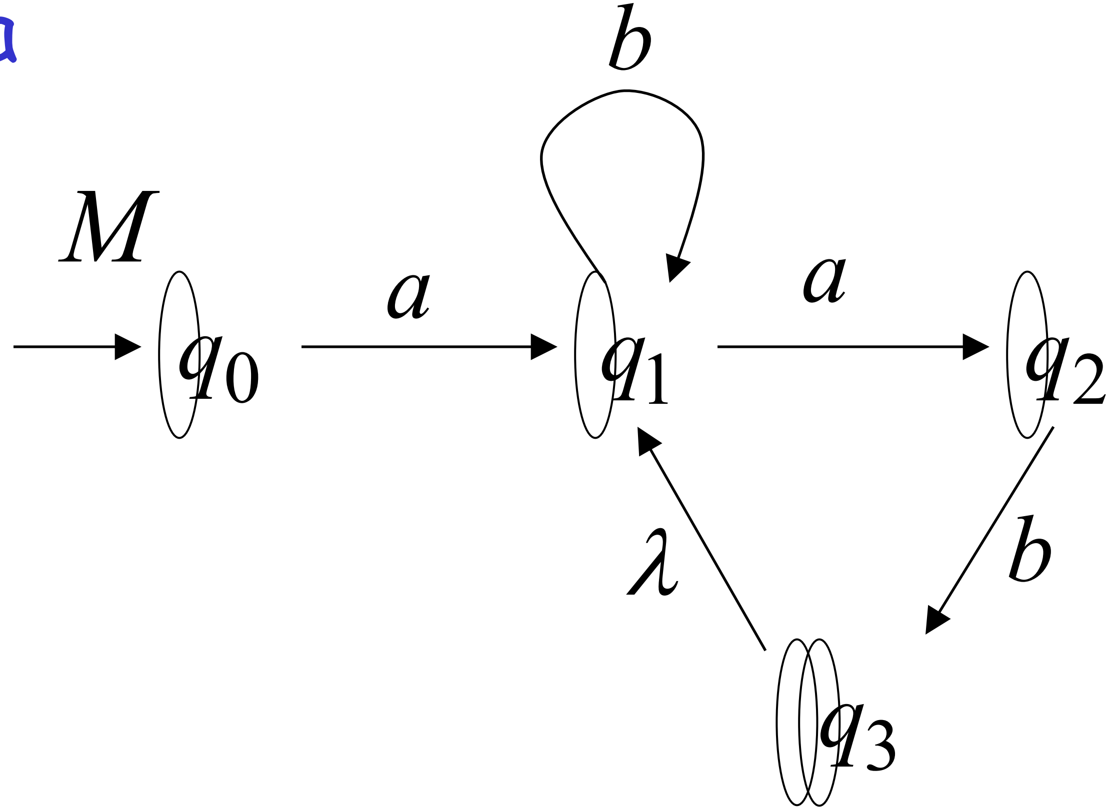
Esempio:



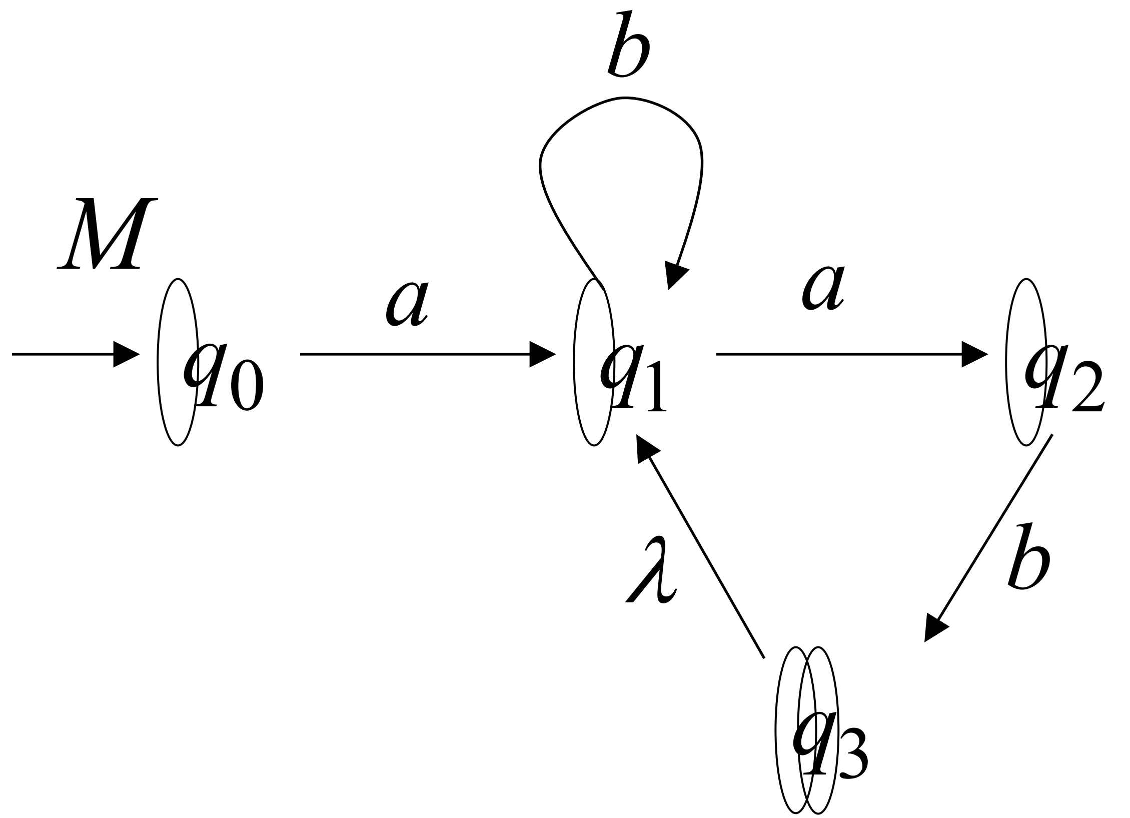
$$L = ab^*ab(b^*ab)^*$$

$$L = L(M)$$

convertiamo M in una right-linear grammatica



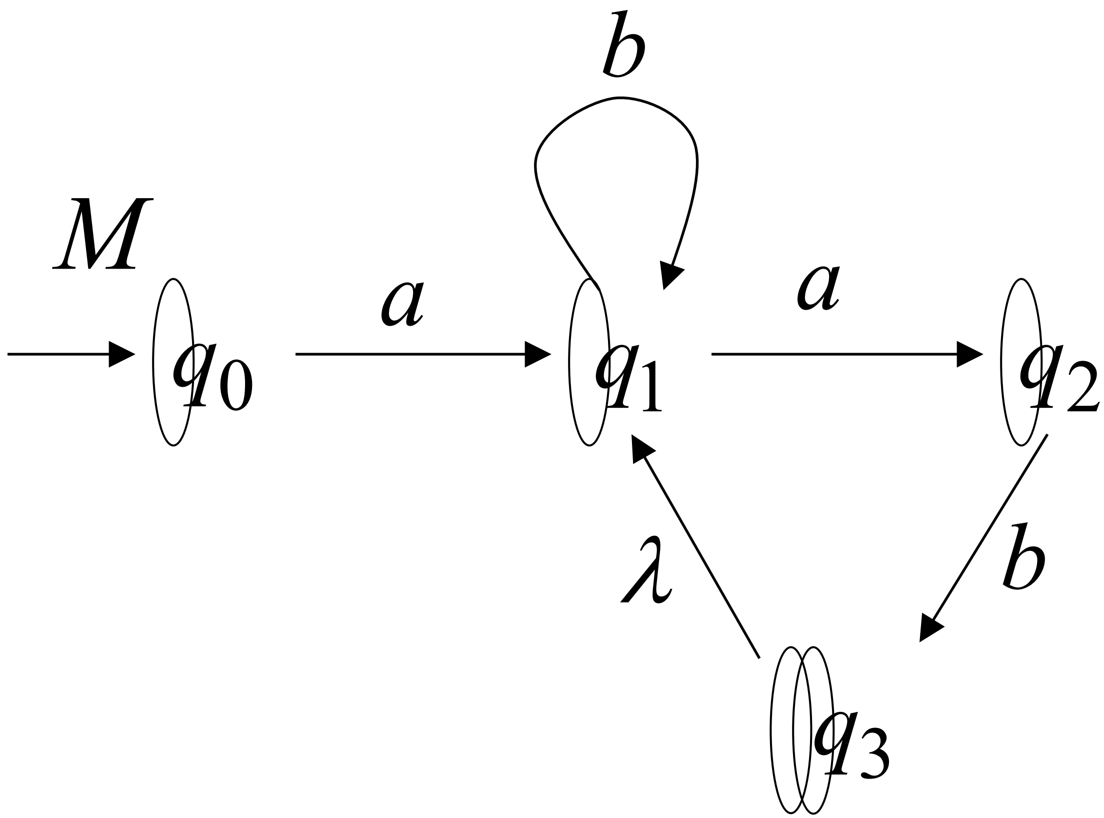
$$q_0 \rightarrow aq_1$$



$$q_0 \rightarrow aq_1$$

$$q_1 \rightarrow bq_1$$

$$q_1 \rightarrow aq_2$$



$$q_0 \to a q_1$$

$$q_1 \to b q_1$$

$$q_1 \to a q_2$$

$$q_2 \to b q_3$$

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

G

$q_0 \to aq_1$

$q_1 \to bq_1$

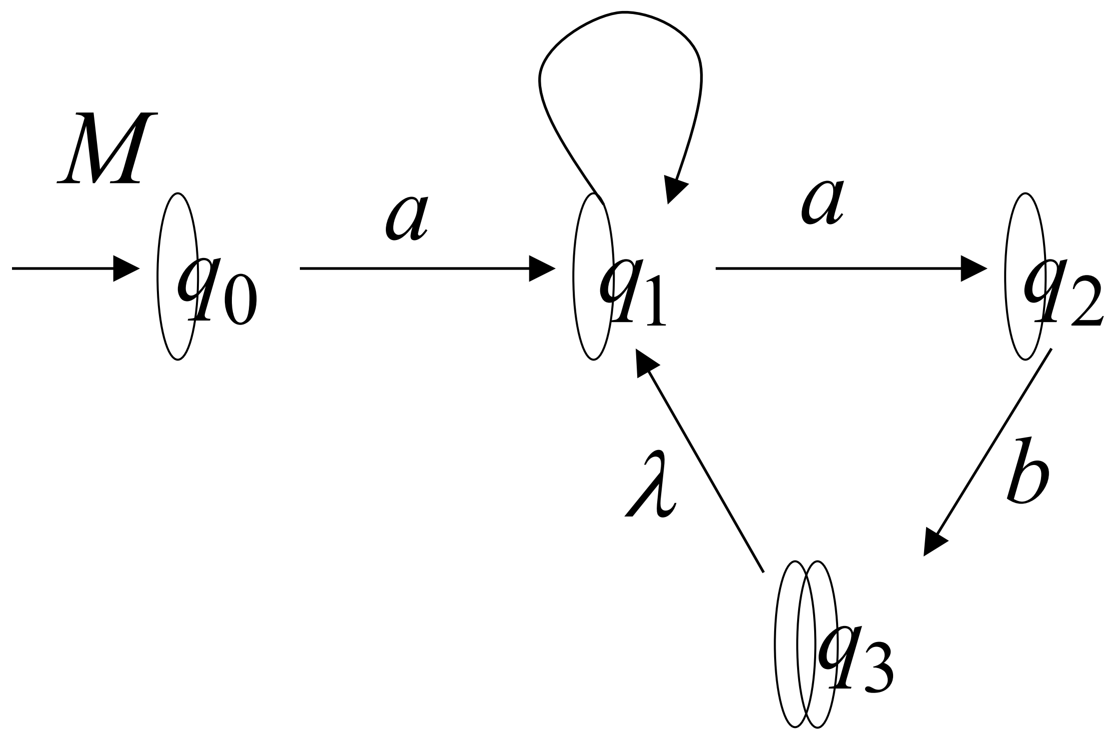
$q_1 \to aq_2$

$q_2 \to bq_3$

$q_3 \to q_1$

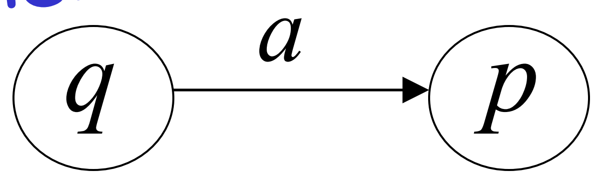
$q_3 \to \lambda$

M



In Generale

per qualsiasi transizione:

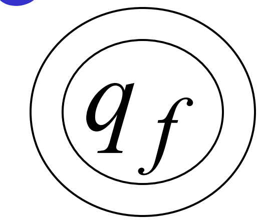


addizioniamo la produzione:

$$q \to ap$$

variable terminal variable

per qualsiasi stato finale :



Addizioniamo la produzione:

$$q_f \to \lambda$$

Since G è right-linear grammatica

G è grammatica regolare

con

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

linguaggi Non-regolari

(Pumping Lemma)

linguaggi Non-regolari

$$\{a^n b^n : \quad n \geq 0\}$$

$$\{vv^R : \quad v \in \{a,b\}^*\}$$

linguaggi regolari

$$a^*b$$

$$b^*c + a$$

$$b + c(a+b)^*$$

$etc...$

Come possiamo provare che un linguaggio L non è regolare?

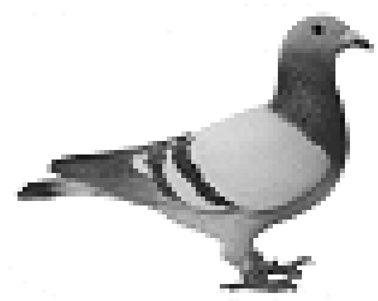
Dobbiamo provare che non vi è Nessun DFa or NFa or RE che lo accetta

Difficulty: non è facile da provare

(perchè vi sono infiniti dfa, nfa e re)

Solution: usare il Pumping Lemma !!!

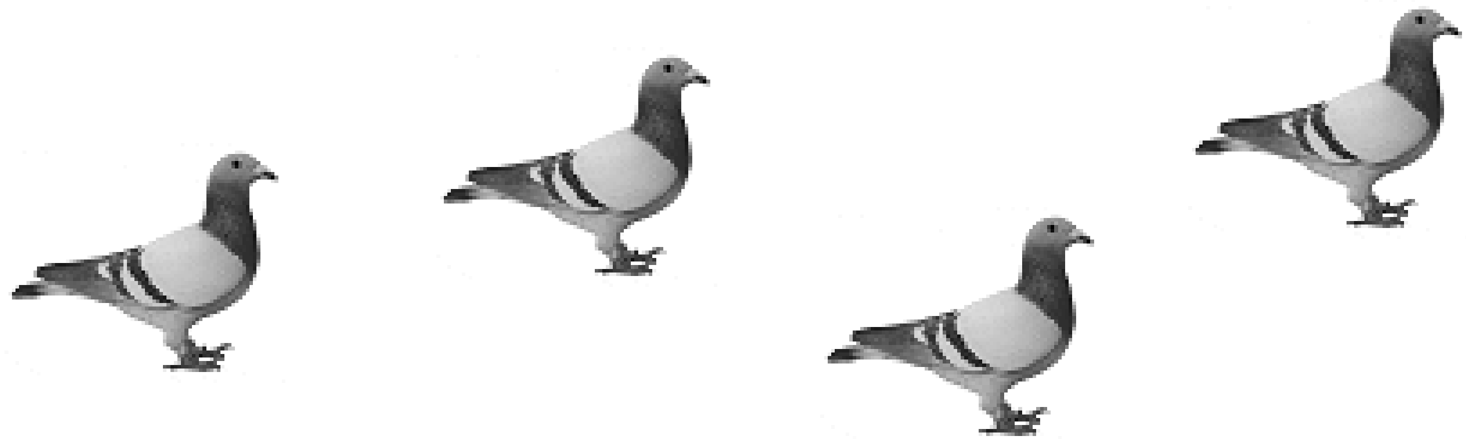
L



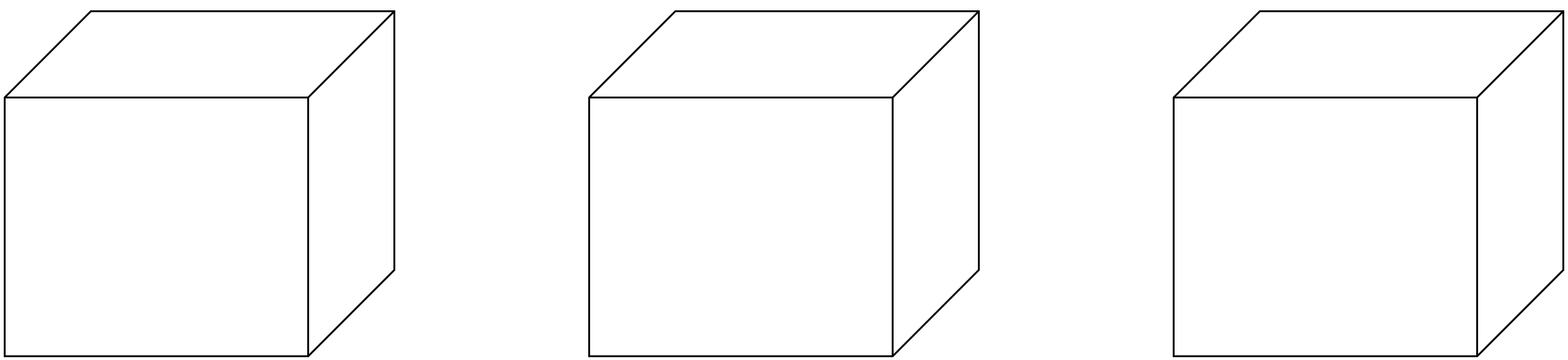
il Pigeonhole Principle

Capelli. Persone.

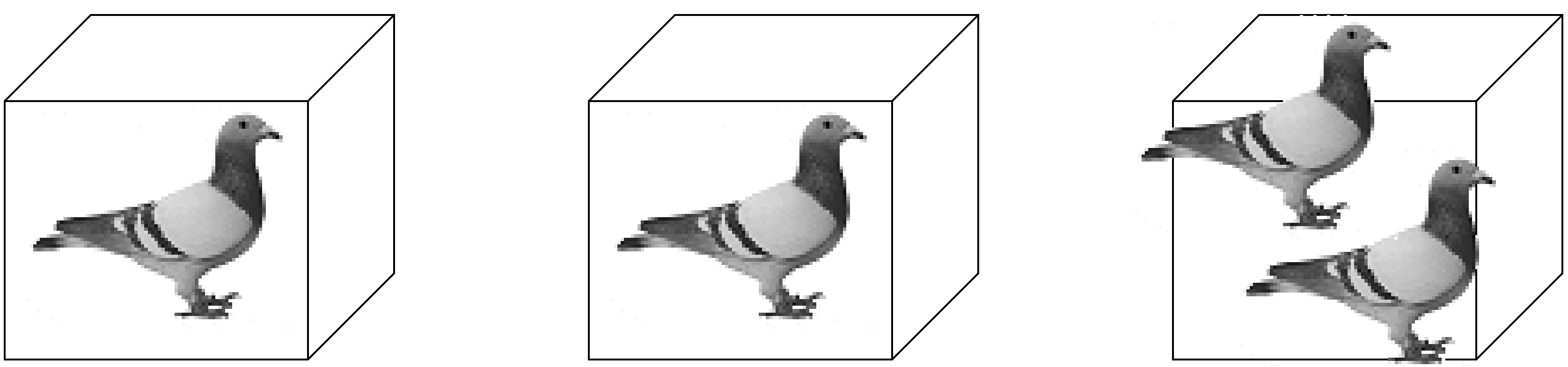
4 pigeons



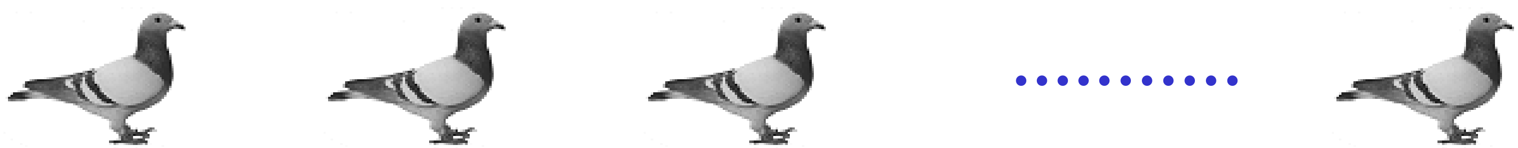
3 pigeonholes



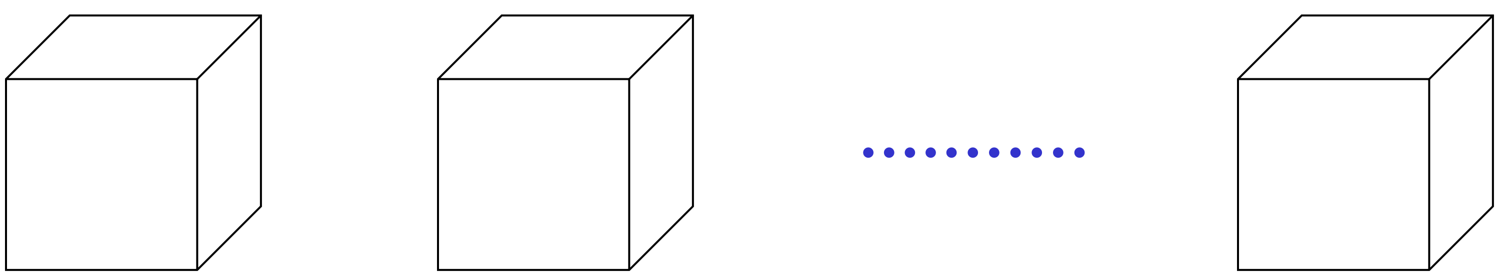
a pigeonhole deve
Contenere due pigeons



n pigeons



m pigeonholes $\qquad n > m$



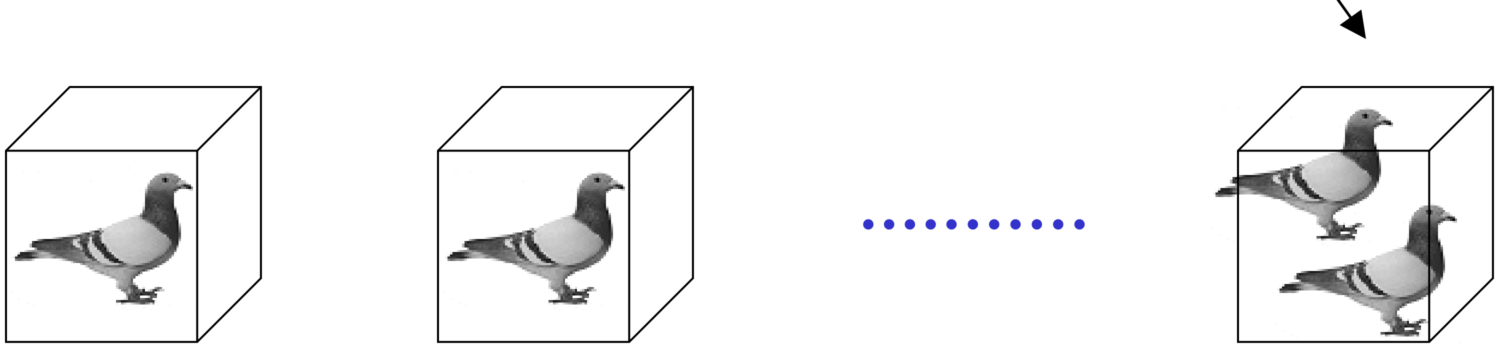
il Pigeonhole Principle

n pigeons

m pigeonholes

$$n > m$$

a pigeonhole deve
Contenere minimo
due pigeons

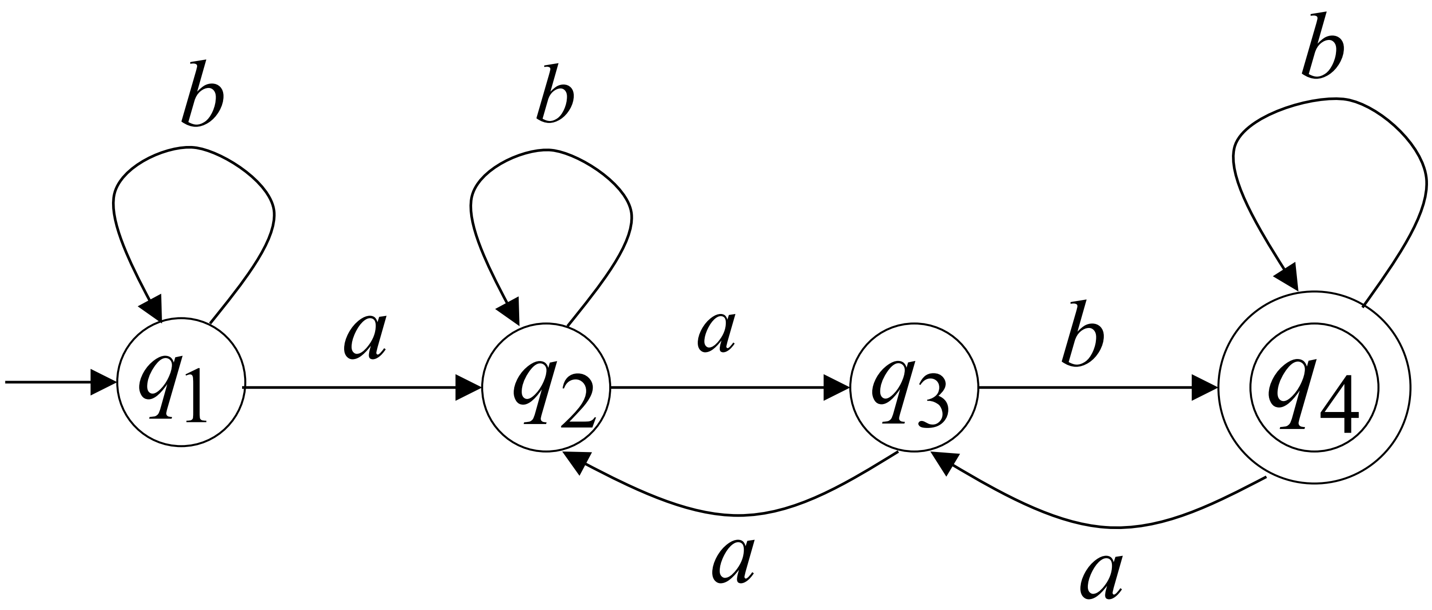


il Pigeonhole Principle

e i

DFa

considera un DFa con 4 stati

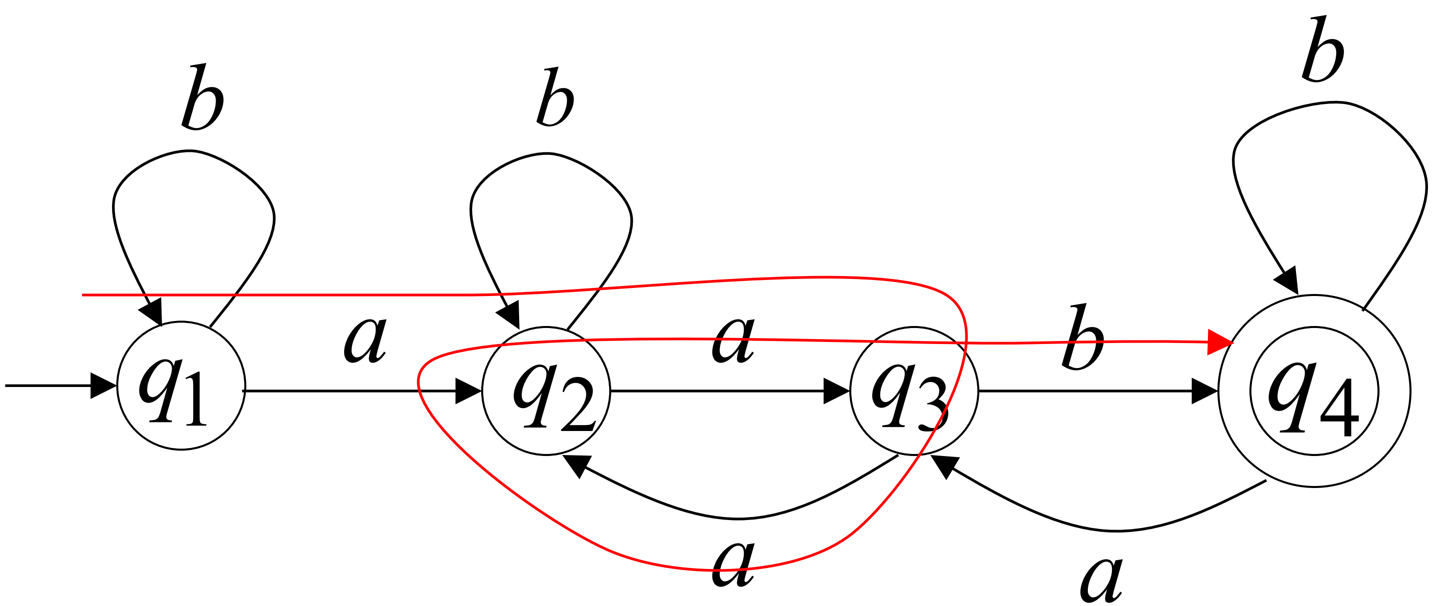


considera il cammino di una "stringa lunga" :
(lunghezza almeno 4)

$$aaaab$$

uno stato è ripetuto nel cammino di $aaaab$

$$q_1 \xrightarrow{a} q_2 \xrightarrow{a} \boxed{q_3} \xrightarrow{a} q_2 \xrightarrow{a} \boxed{q_3} \xrightarrow{b} q_4$$

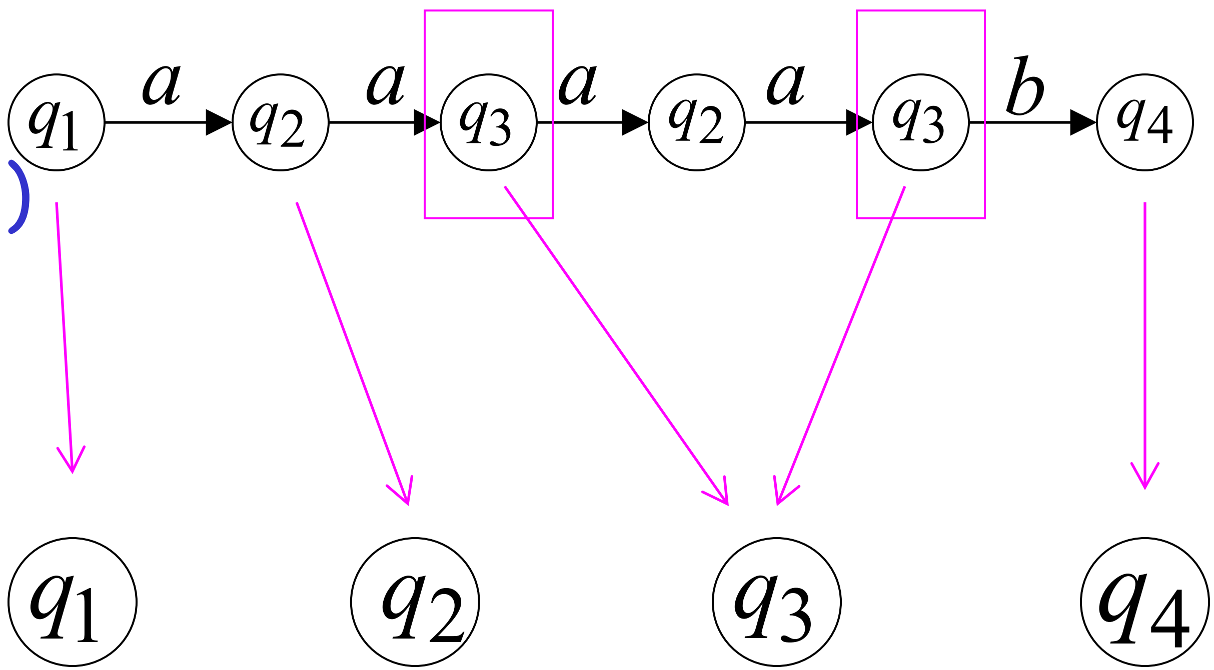


il stato è ripetuto da a, risultato del pigeonhole principle

cammino di $aaaab$

Pigeons:
(cammino stati)

Sono più degli



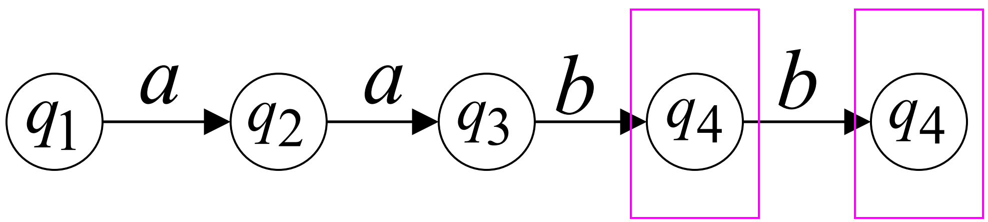
Cesti:
(stati dell'automata)

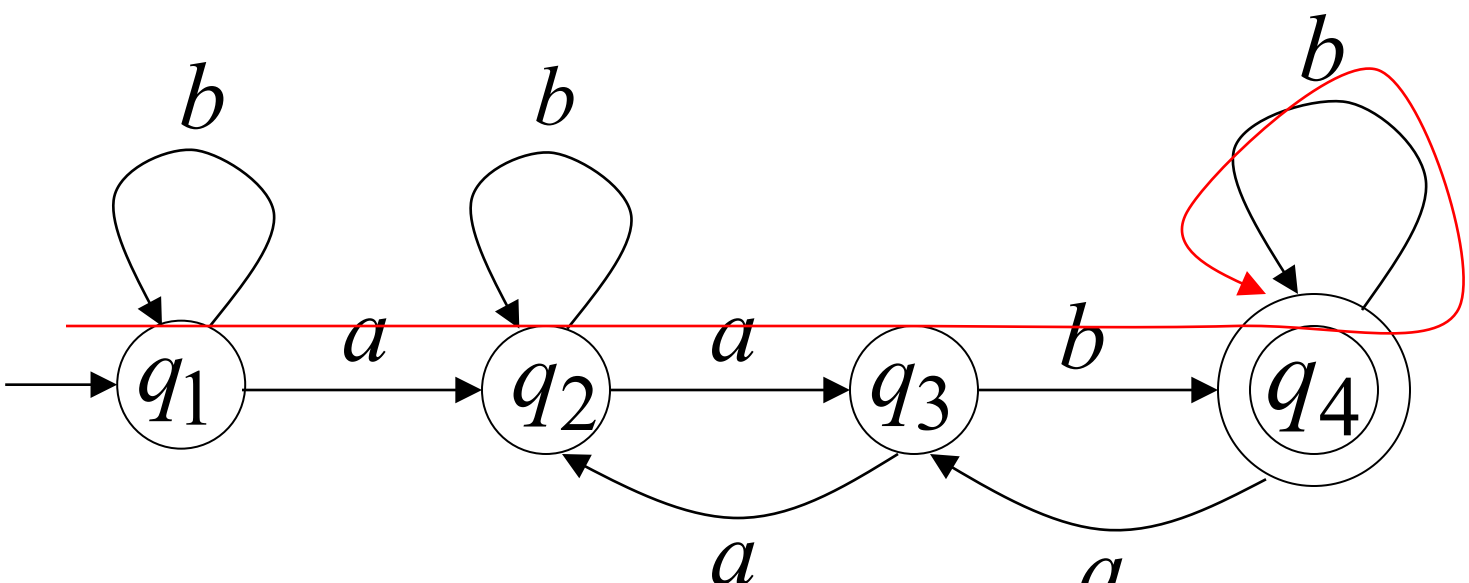
stato
Ripetuto

considera il cammino di a “long” stringa: $aabb$
(lunghezza almeno 4)

Dal pigeonhole principle:

uno stato è ripetuto nel cammino di $aabb$



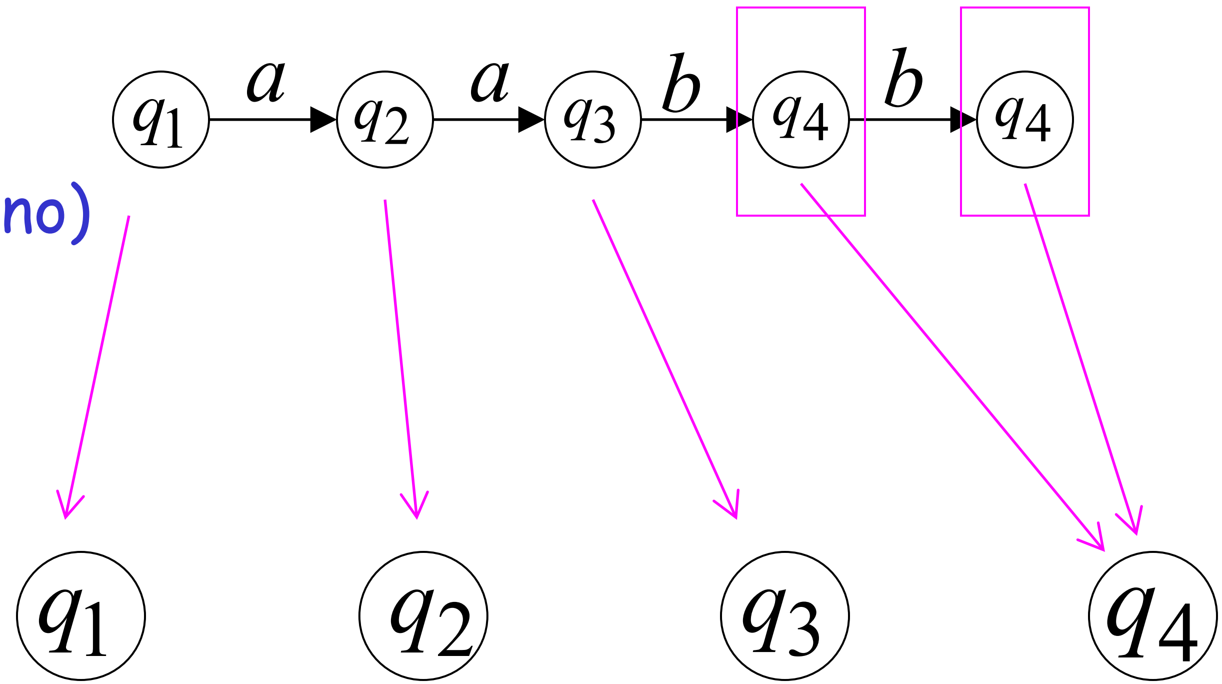


il stato è ripetuto come risultato del pigeonhole principle

cammino di $aabb$

Pigeons:
(stati del cammino)

sono di più



Nests:
(automaton stati)

Stati automa

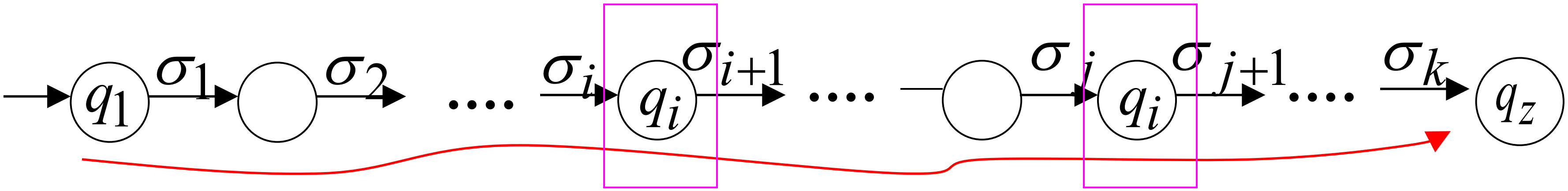
ripetuto
stato

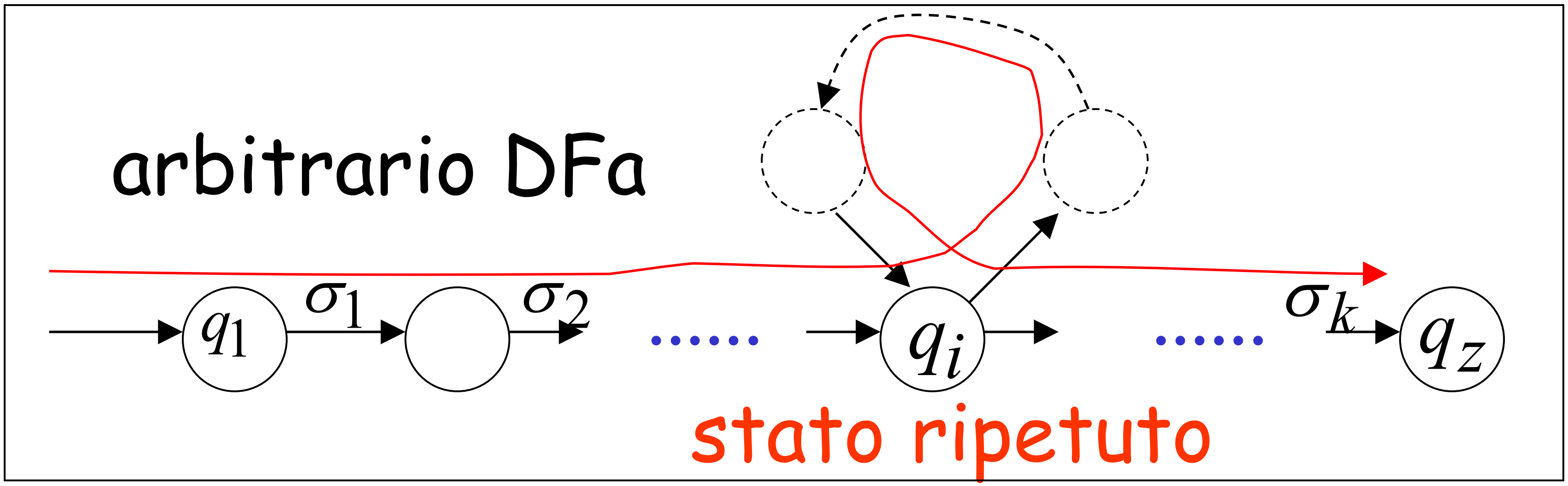
In Generale: se $|w| \geq$ #states of DFA

Per il pigeonhole principle,

uno stato è ripetuto nel cammino W

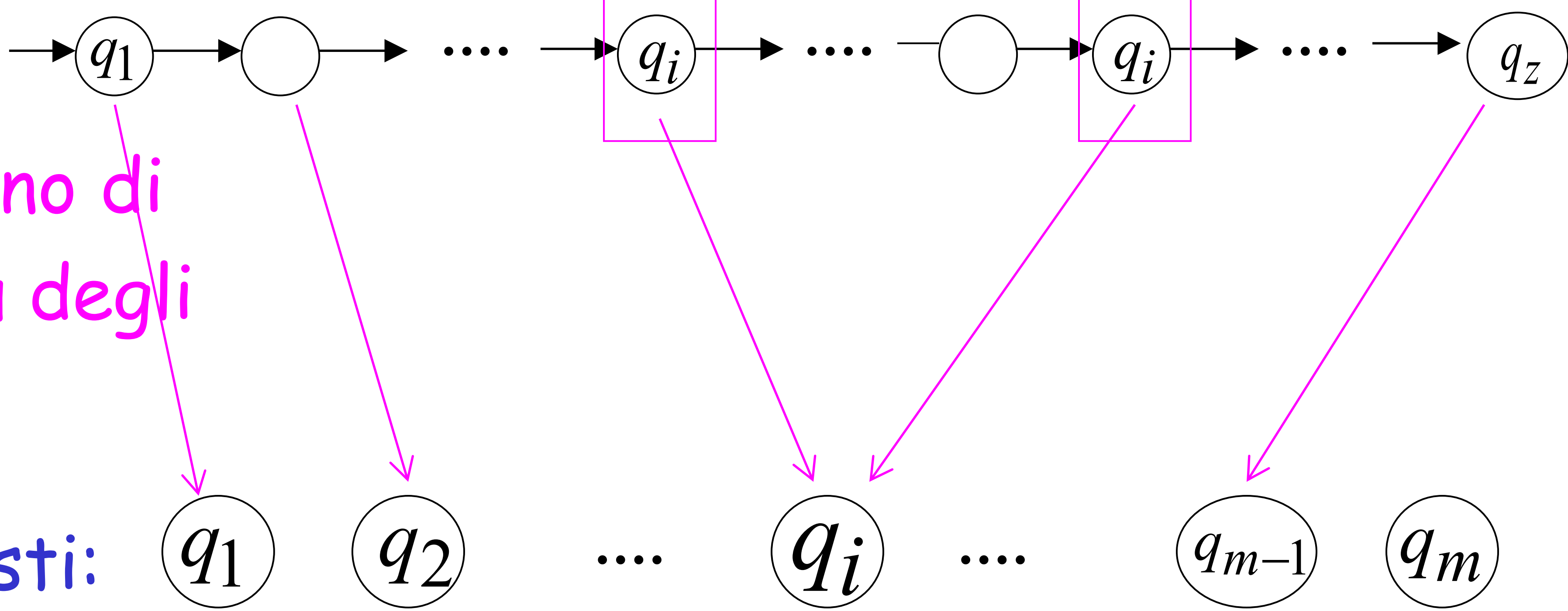
cammino di $w = \sigma_1 \sigma_2 \cdots \sigma_k$





$|w| \geq \#\text{states of DFA} = m$

Pigeons: (stati del cammino) cammino di w



Sono di Più degli

cesti: (stati dell automata)

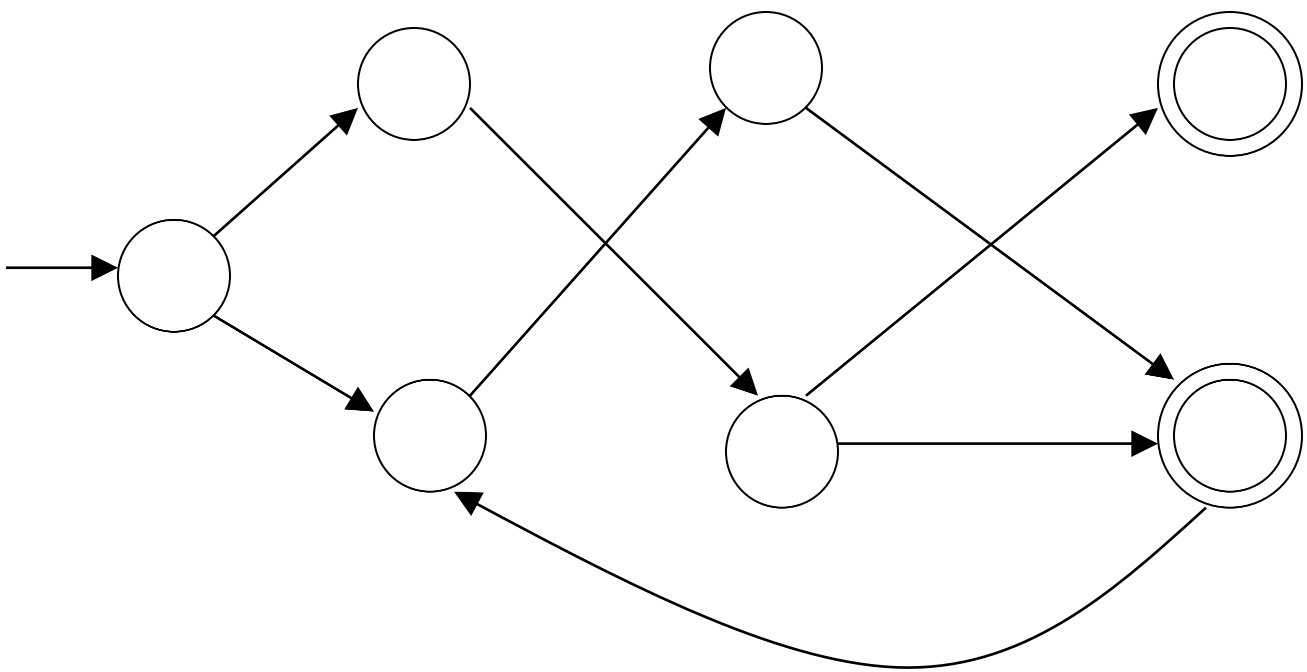
uno stato è ripetuto

il Pumping Lemma

prendi un linguaggio regolare infinito L

(contiene un numero infinito di stringhe)

Sia un DFa che accetta L



m stati

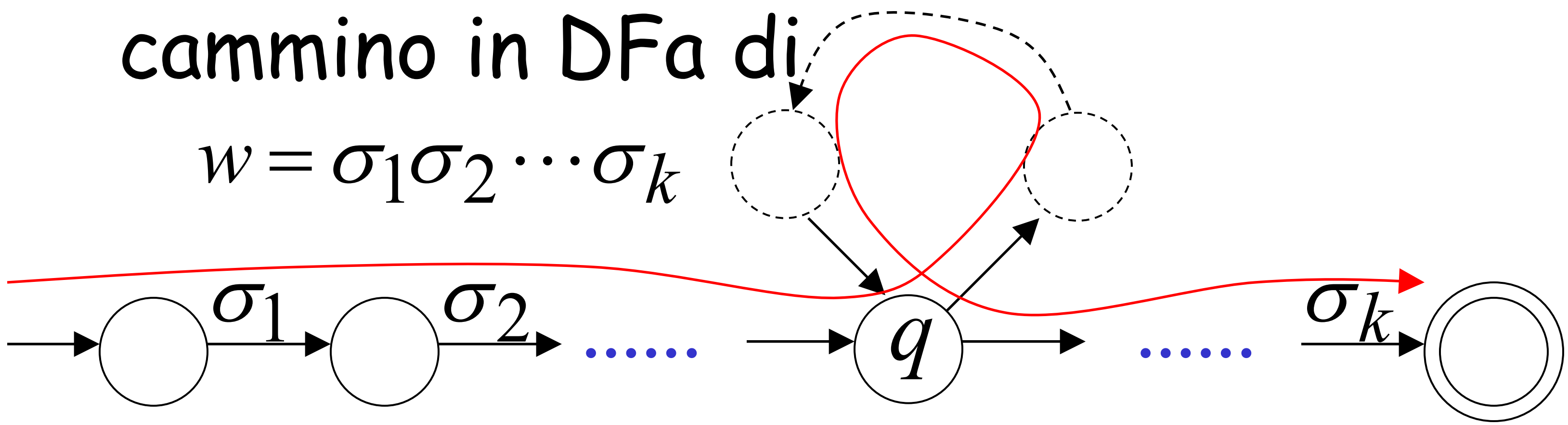
prendiamo una stringa $w \in L$ con $|w| \geq m$

(numero di stati del DFa)

Almeno uno stato è ripetuto nel cammino di w

cammino in DFa di

$$w = \sigma_1 \sigma_2 \cdots \sigma_k$$



stato ripetuto in DFa

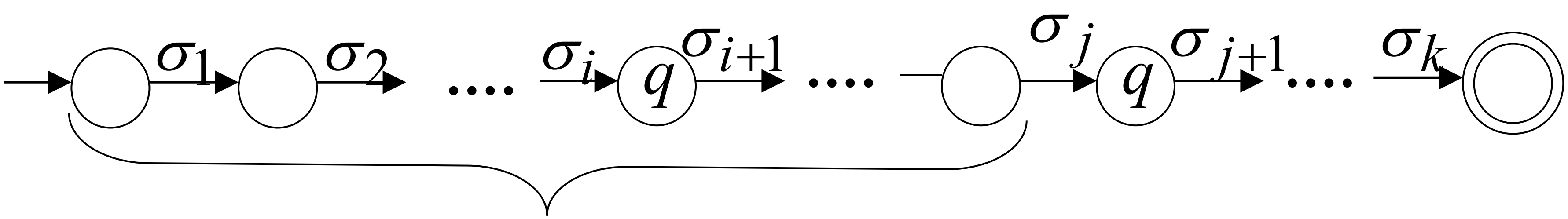
Ci saranno molti stati ripetuti

prendiamo il primo stato ripetuto q

In una dimensione il cammino di: w

prima occorrenza

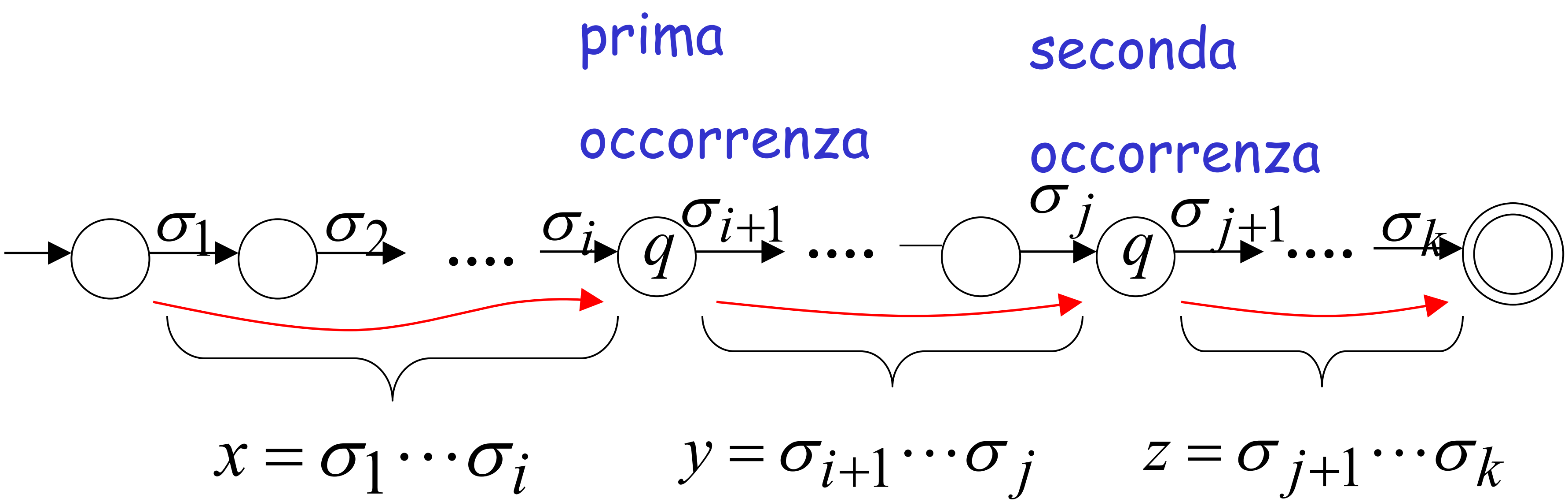
seconda occorrenza



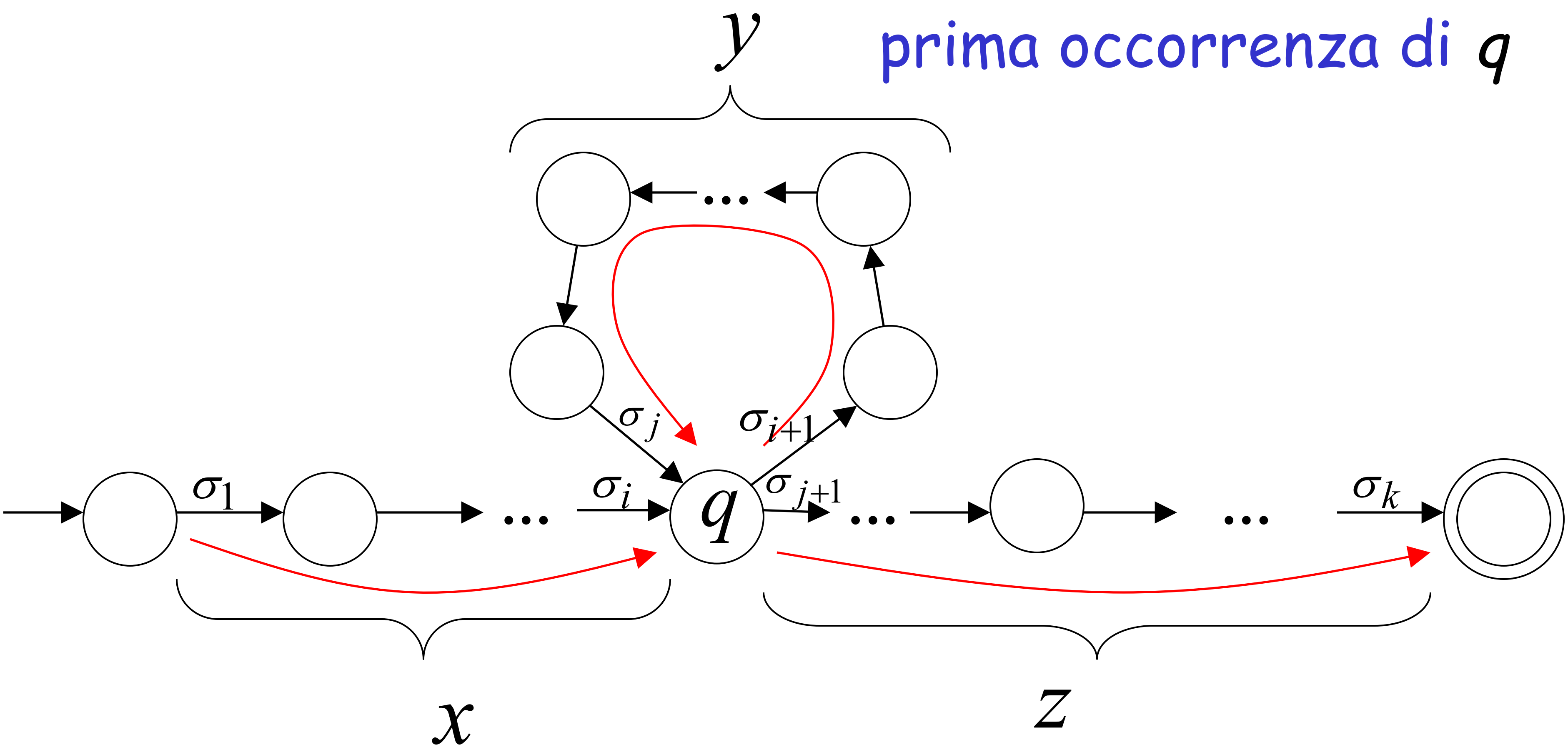
unico stato

Possiamo scrivere $w = xyz$

Una dimensione del cammino di : w

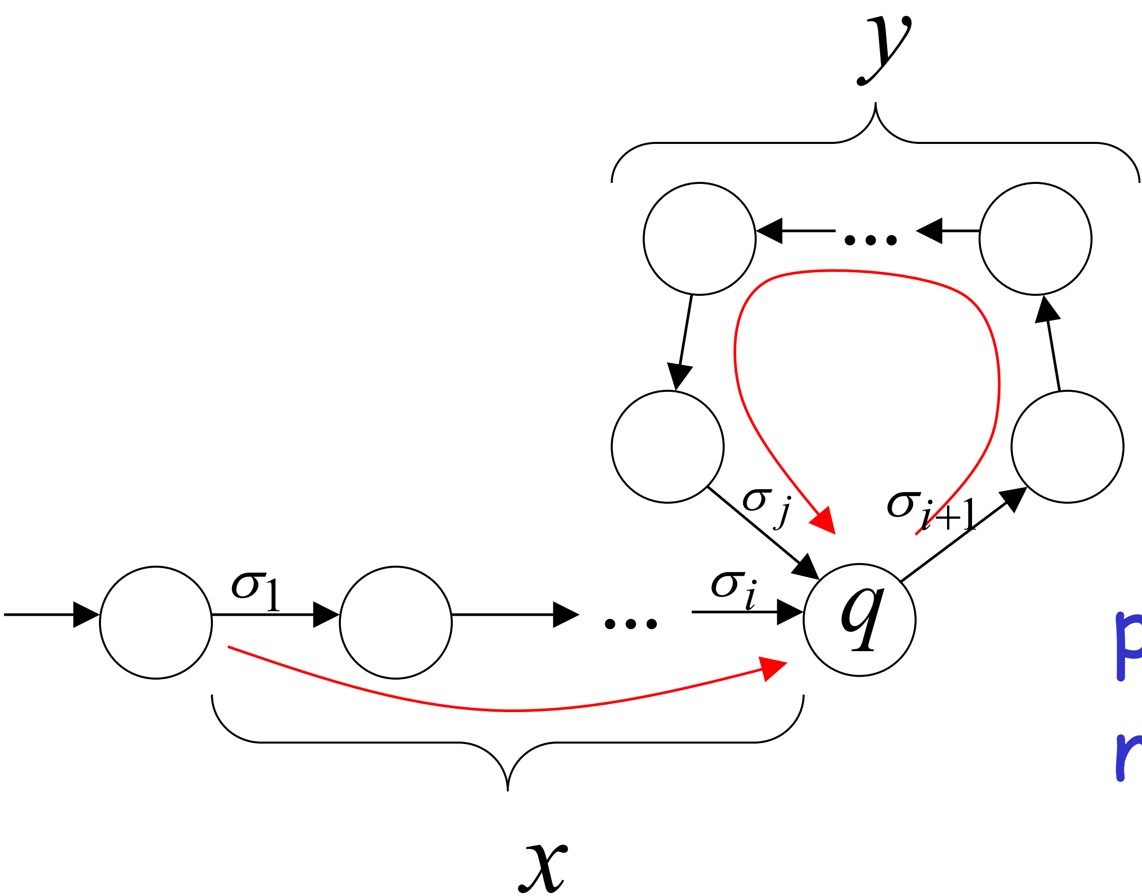


Nel DFa: $w = x\ y\ z$



y
contiene solo
prima occorrenza di q
σ_j
σ_{i+1}
σ_1
σ_i
q
σ_{j+1}
σ_k
x
z

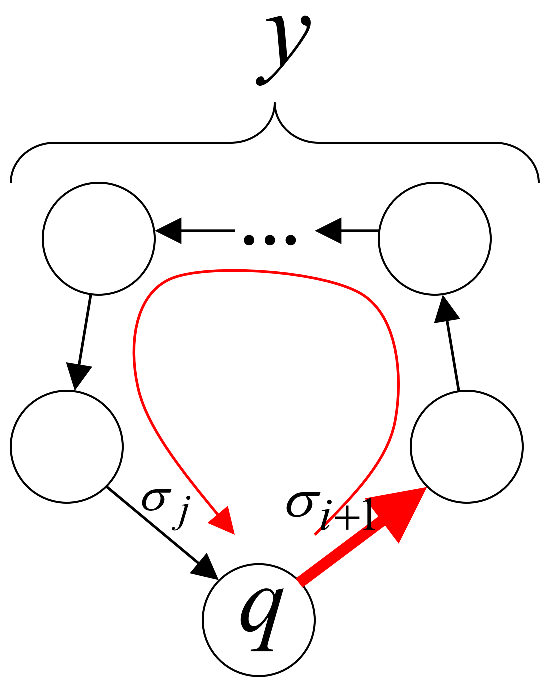
osservazione: lunghezza $|x\,y| \leq m$ numero di stati del DFa



unici stati

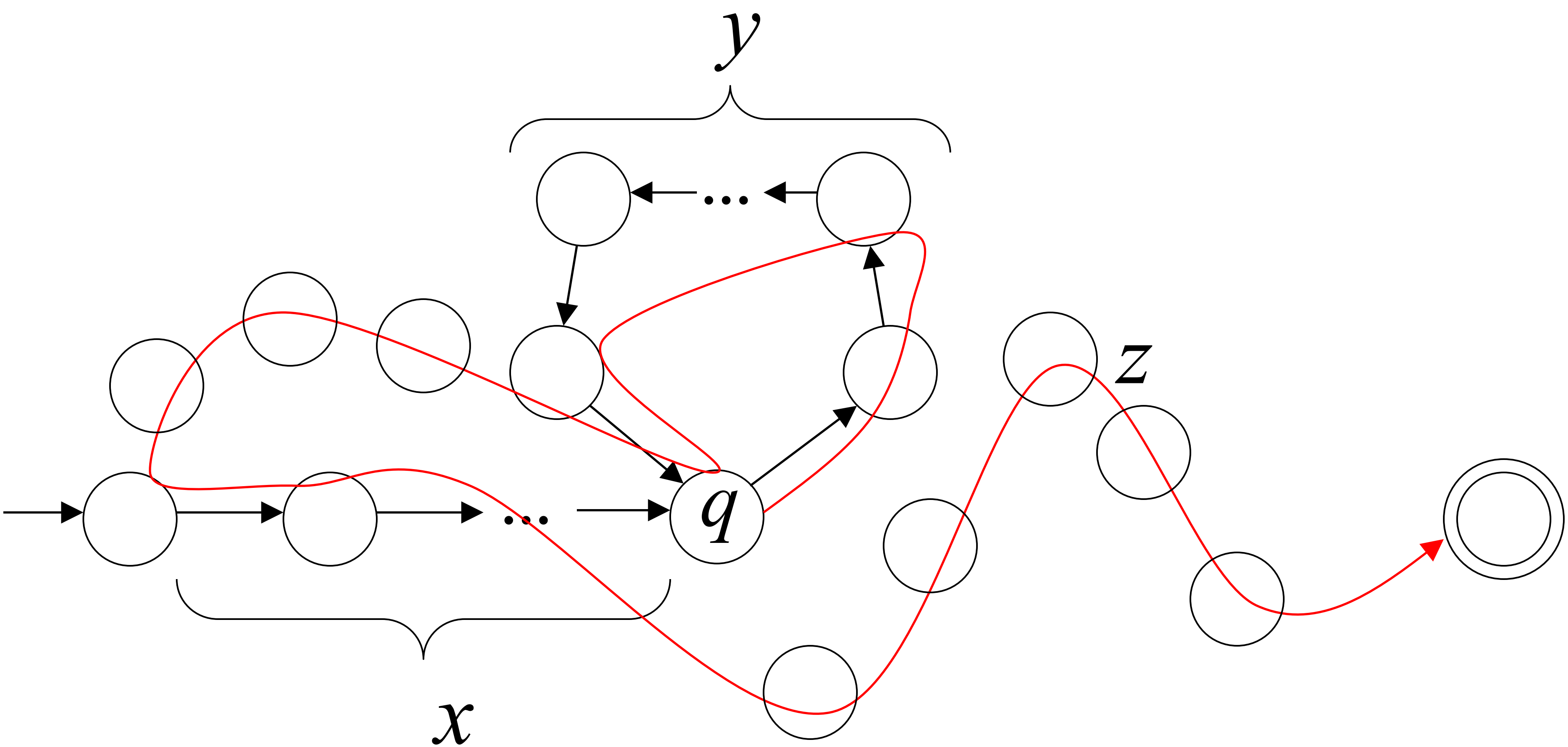
poichè, in xy nessuno è stato ripetuto (eccetto q) osservazione: lunghezza $|\,y\,| \geq 1$

Vi è almeno un loop



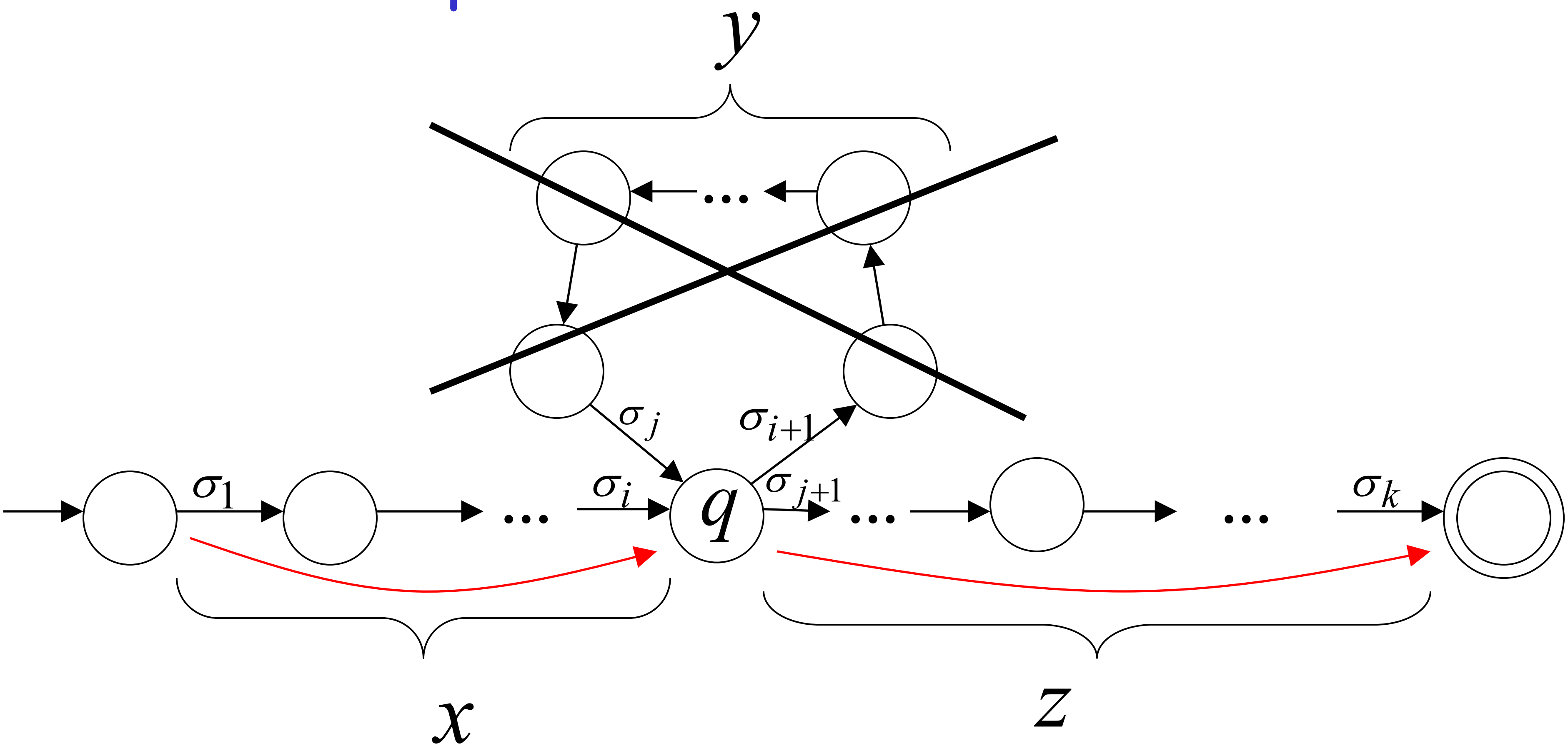
Non badiamo alla forma della stringa z

z può avere pezzi di cammino di x and y



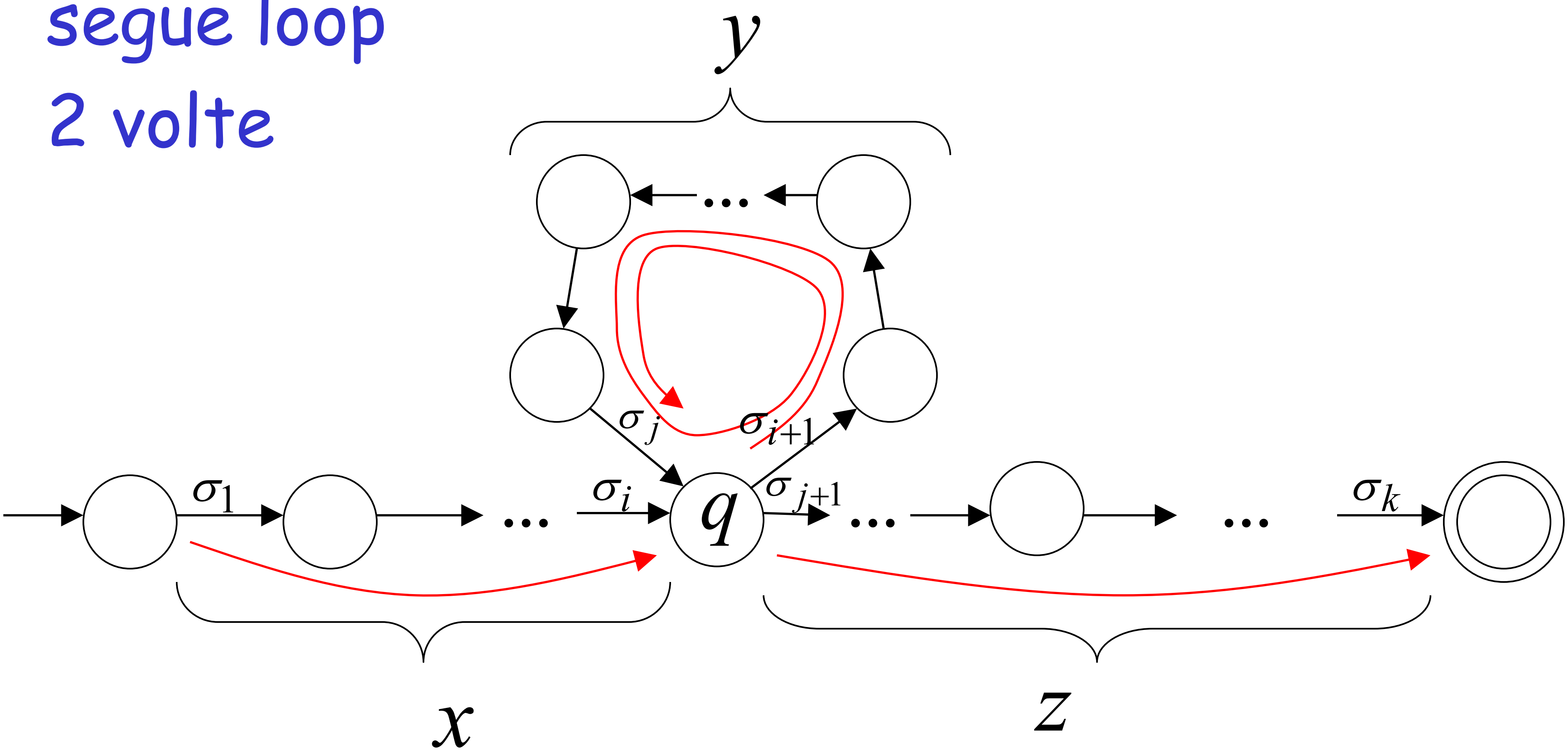
stringa addizionale: la stringa $x\,z$ è accettata

Non fa il loop



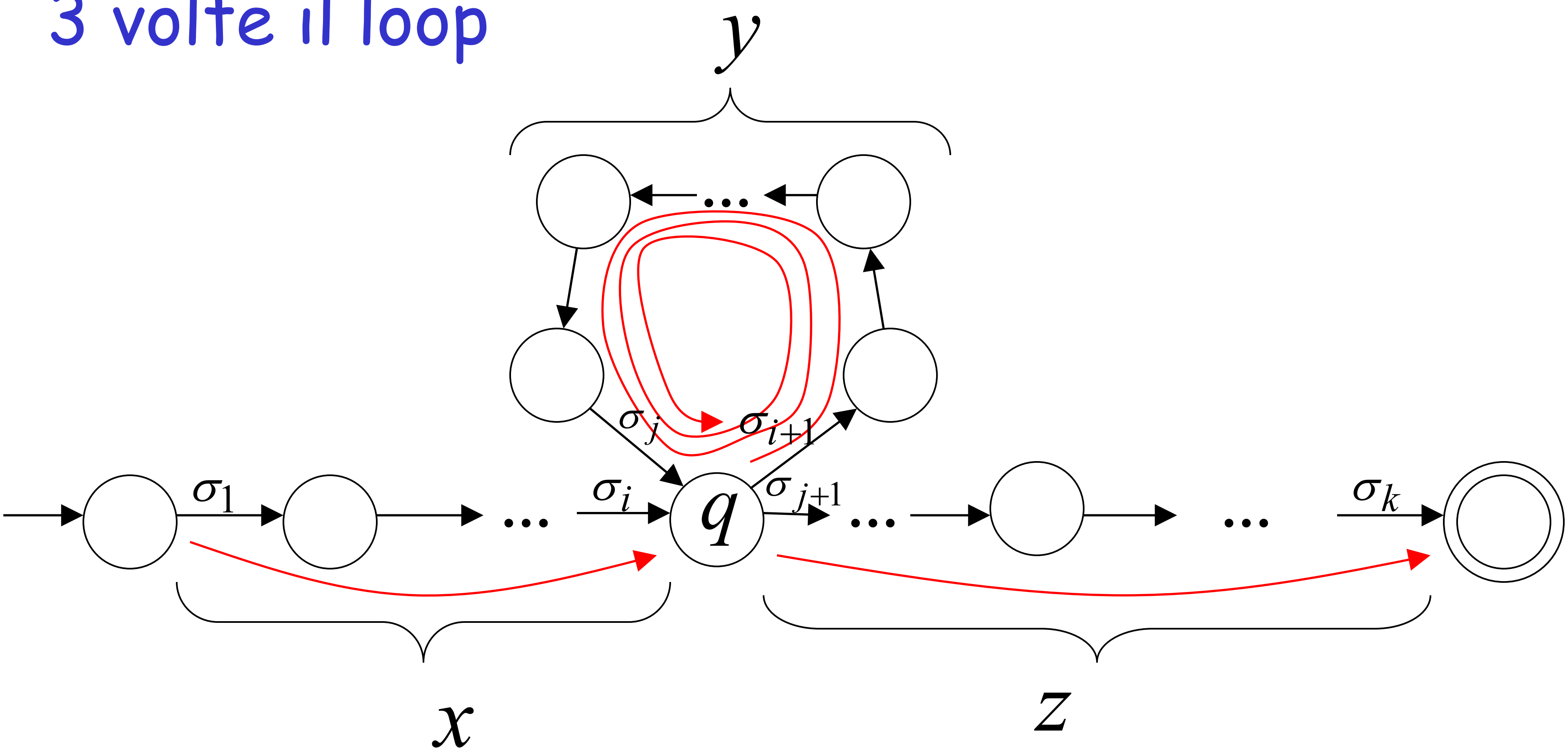
stringa addizionale : la stringa $x\ y\ y\ z$ è accettata

segue loop
2 volte



addizionale stringa: la stringa $x\ y\ y\ y\ z$ è accettata

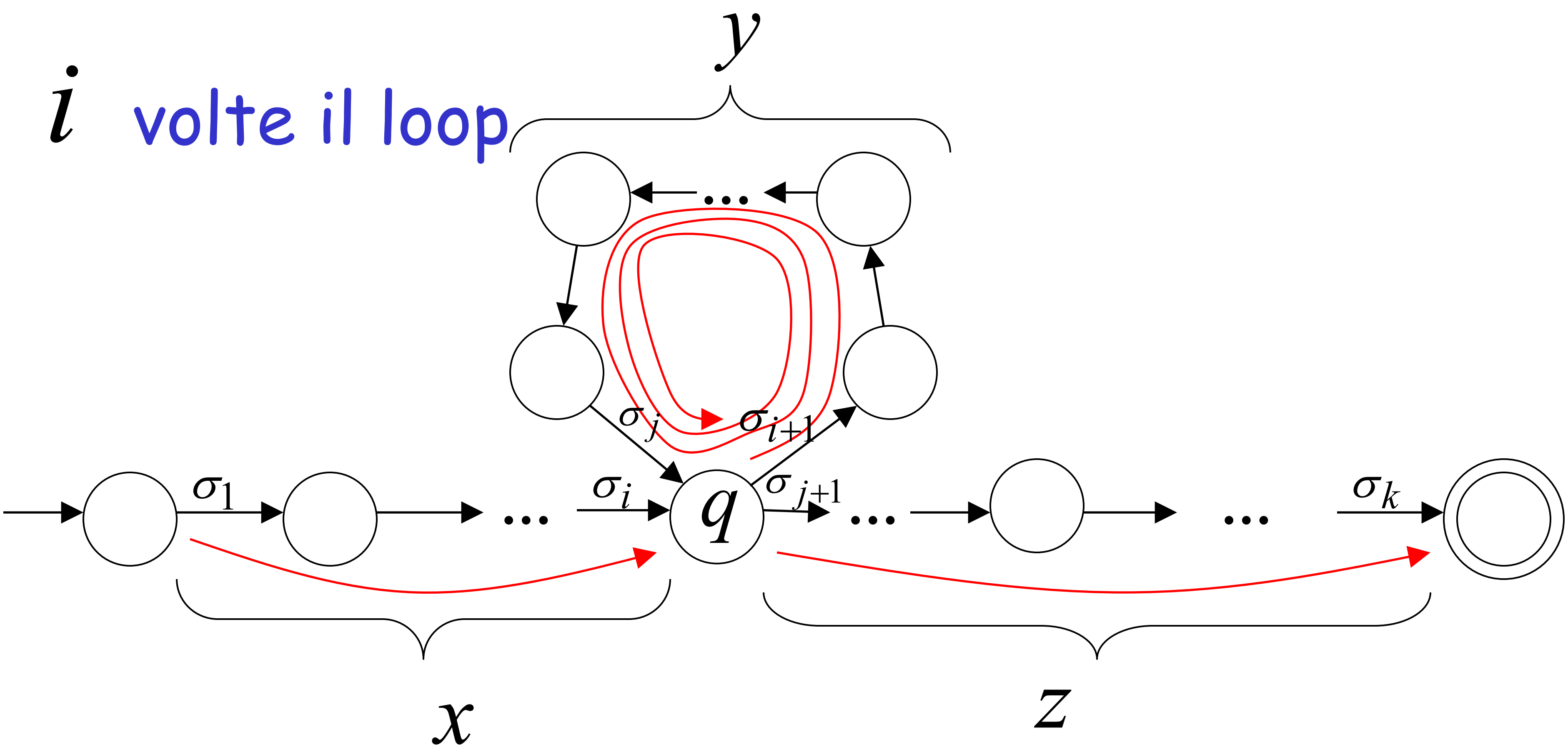
3 volte il loop



In Generale: la stringa $x\, y^i\, z$

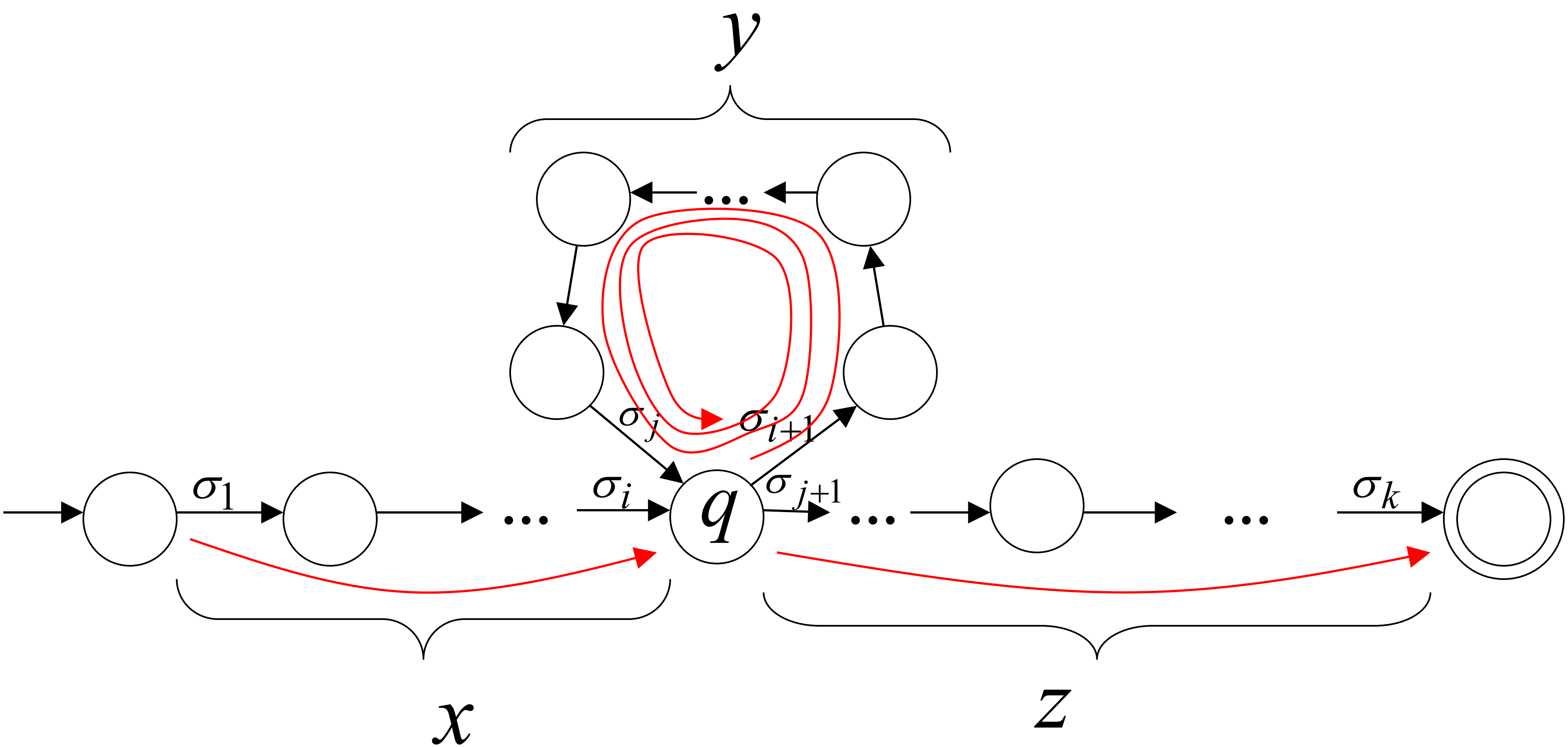
è accettata $i = 0, 1, 2, \ldots$

i volte il loop



quindi: $\quad x\, y^i\, z \in L \qquad i = 0, 1, 2, \ldots$

linguaggio accettato dal DFa



il Pumping Lemma:

- dato un linguaggio regolare infinito L
- esiste an intero m (lunghezza critica)
- per ogni stringa $w \in L$ con lunghezza $|w| \geq m$
- possiamo scrivere $w = x\, y\, z$
- con $|x\, y| \leq m$ e $|y| \geq 1$
- tale che: $x\, y^i\, z \in L \qquad i = 0, 1, 2, \ldots$

nel libro sipster :

lunghezza Critica = m lunghezza Pumping p

applicazioni applicazioni

del
Pumping Lemma

osservazione:

ogni linguaggio di dimensione finita è regolare

(possiamo facilmente costruire an NFa
che accetta ogni stringa nel linguaggio)

quindi, ogni linguaggio non-regolare
è di dimensione infinita

(contiene an infinito numero di stringhe)

supponiamo vogliamo provare che
Un linguaggio infinito L non è regolare

1. assumiamo l' opposto: L è regolare

2. il pumping lemma deve valere per L

3. usiamo il pumping lemma per ottenere una contradizione

4. quindi, L non è regolare

Spiegazione Step 3: come avere una contradizione

1. Let m sia la lunghezza critica for L

2. Scegliamo una stringa particolare $w \in L$ che soddisfa la condizione di lunghezza $|w| \geq m$

3. scrivere $w = xyz$

4. mostriamo che $w' = xy^i z \notin L$ Per qualche $i \neq 1$

5. Questo ci dà una contradizione, poichè dal pumping lemma $w' = xy^i z \in L$

Note: È sufficiente mostrare che solo una stringa $w \in L$ genera una contradizione

Non dobbiamo ottenere contradizioni per ogni $w \in L$

Esempi di applicazioni del Pumping Lemma

teorema: il linguaggio $L = \{a^n b^n : n \geq 0\}$ non è regolare

dim: Usa il Pumping Lemma

$$L = \{a^n b^n : n \geq 0\}$$

assumiamo per contradizione

che L è un linguaggio regolare

Since L è infinito

Possiamo applicare il Pumping Lemma

$$L = \{a^n b^n : n \geq 0\}$$

sia m la lunghezza critica per L

Prendiamo a stringa w such che: $w \in L$

e lunghezza $|w| \geq m$

prendiamo $w = a^m b^m$

Dal Pumping Lemma:

possiamoscrivere $w = a^m b^m = x\ y\ z$

Con lunghezza $|x\,y| \le m, \quad |y| \ge 1$

$$w = xyz = a^m b^m = \overbrace{\underbrace{a\ldots a}_{x}\underbrace{a\ldots a}_{y}a\ldots a}^{m}\overbrace{b\ldots b}^{m}$$

$$\underbrace{a\ldots a}_{y}\ \underbrace{a\ldots ab\ldots b}_{z}$$

allora: $y = a^k, \quad 1 \le k \le m$

$$x\ y\ z = a^m b^m \qquad\qquad y = a^k, \quad 1 \le k \le m$$

dal Pumping Lemma:

$$x\ y^i\ z \ \in\ L$$

$$i = 0, 1, 2, \ldots$$

allora: $x\ y^2\ z \ \in\ L$

$$x\ y\ z = a^m b^m \qquad y = a^k, \quad 1 \le k \le m$$

Dal **Pumping Lemma:** $$x\ y^2\ z \ \in\ L$$

$$xy^2z = \overbrace{\underbrace{a...a}_{x}\underbrace{a...a}_{y}\underbrace{a...a}_{y}a...a}^{m+k}\overbrace{b...b}^{m} \in L$$

(where $z = a...ab...b$)

allora: $a^{m+k}b^m \in L$

aaabbb =xyz
1 caso x=aa y=a z=bbb
aa aa bbb

2 caso x=aaab y=b z=b
aaab bb b

3 caso
x=aa y=ab z=bb
aa ababab bb

$$a^{m+k}b^m \in L \qquad\qquad k \geq 1$$

MA: $$L = \{a^n b^n : n \geq 0\}$$

$$\Downarrow$$

$$a^{m+k}b^m \notin L$$

contradizione!!!

quindi: l'assunzione che che L
è un linguaggio regolare
non è vera

Conclusione: L non è un linguaggio regolare

END dim

linguaggio Non-regolare $\{a^n b^n : \quad n \geq 0\}$

Linguaggio regolare

$L(a^* b^*)$

a*b*

aayabybb=xyz y=a

aa aa a bbb

aaa bbbbb

y=ab

aaabbbb

23922 - ROMA - Part. del frammento di bassorilievo, rappres. le Stagioni - Museo Vaticano Ripr. int. - Andersc

Precisazioni unioni e intersezione automi

Per studenti 2020

Chiusura rispetto intersezione

automa M_1 automa M_2

$\boxed{\text{DFA per } L_1}$ $\boxed{\text{DFA per } L_2}$

«Automi completi»

Costruiamo un DFA M che accetta $L_1 \cap L_2$

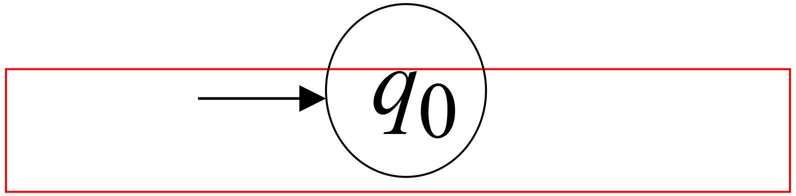
M Simula in parallelo M_1 e M_2

Stati in M

q_i, p_j

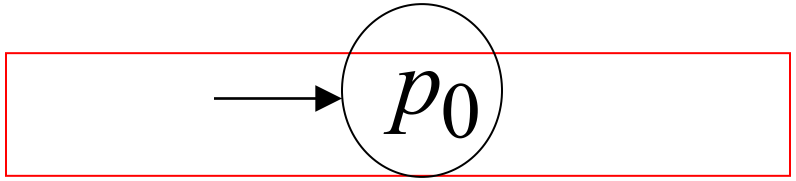
Stato in M_1 Stato in M_2

DFA M_1



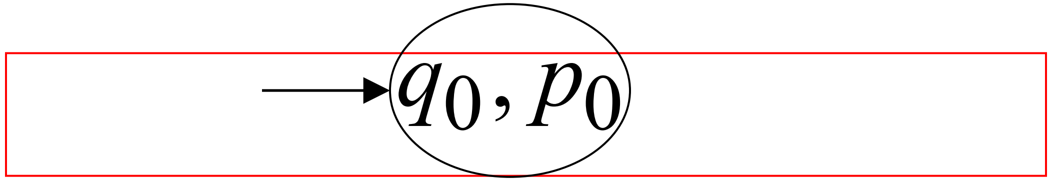
stato iniziale

DFA M_2



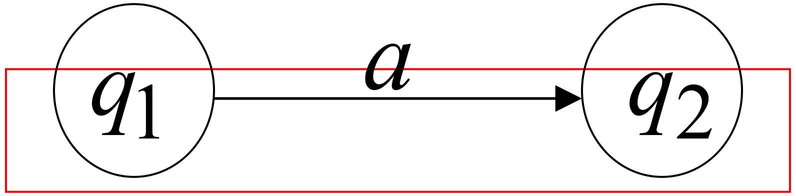
stato iniziale

DFA M



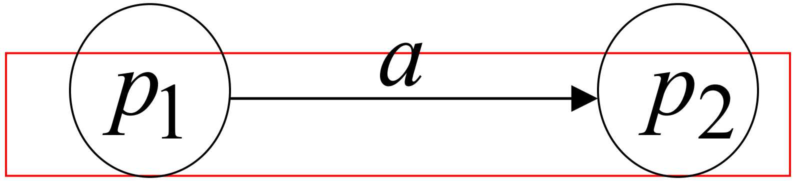
nuovo stato iniziale

DFA M_1



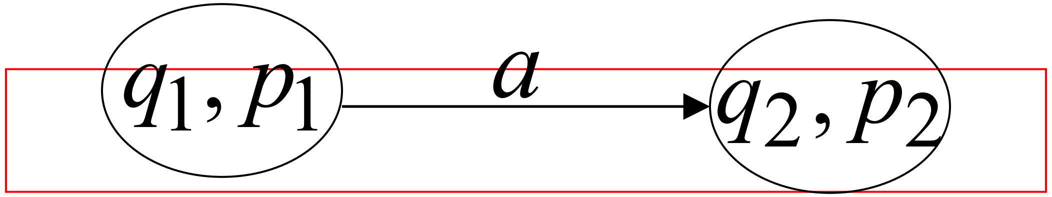
transizione

DFA M_2

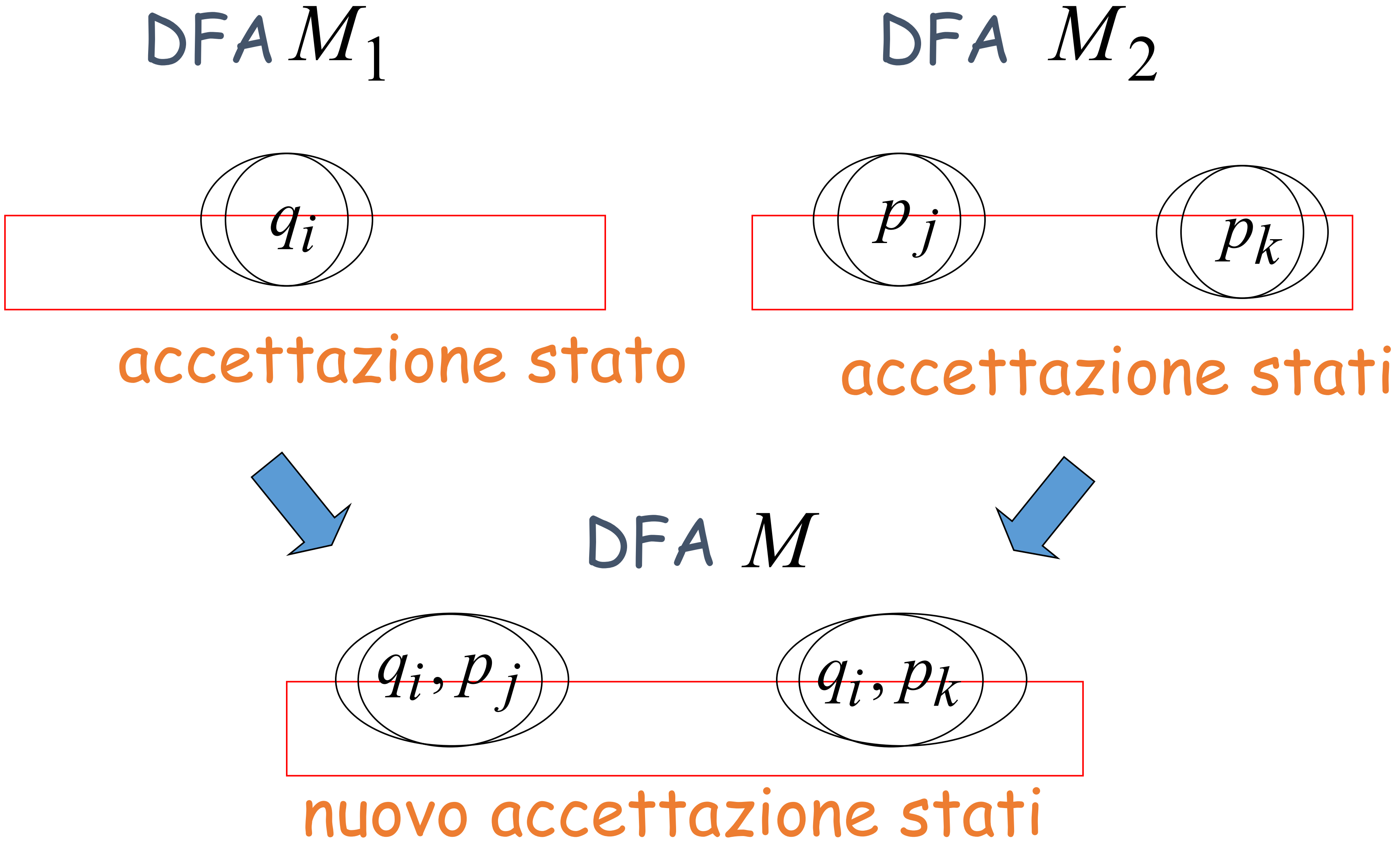


transizione

DFA M

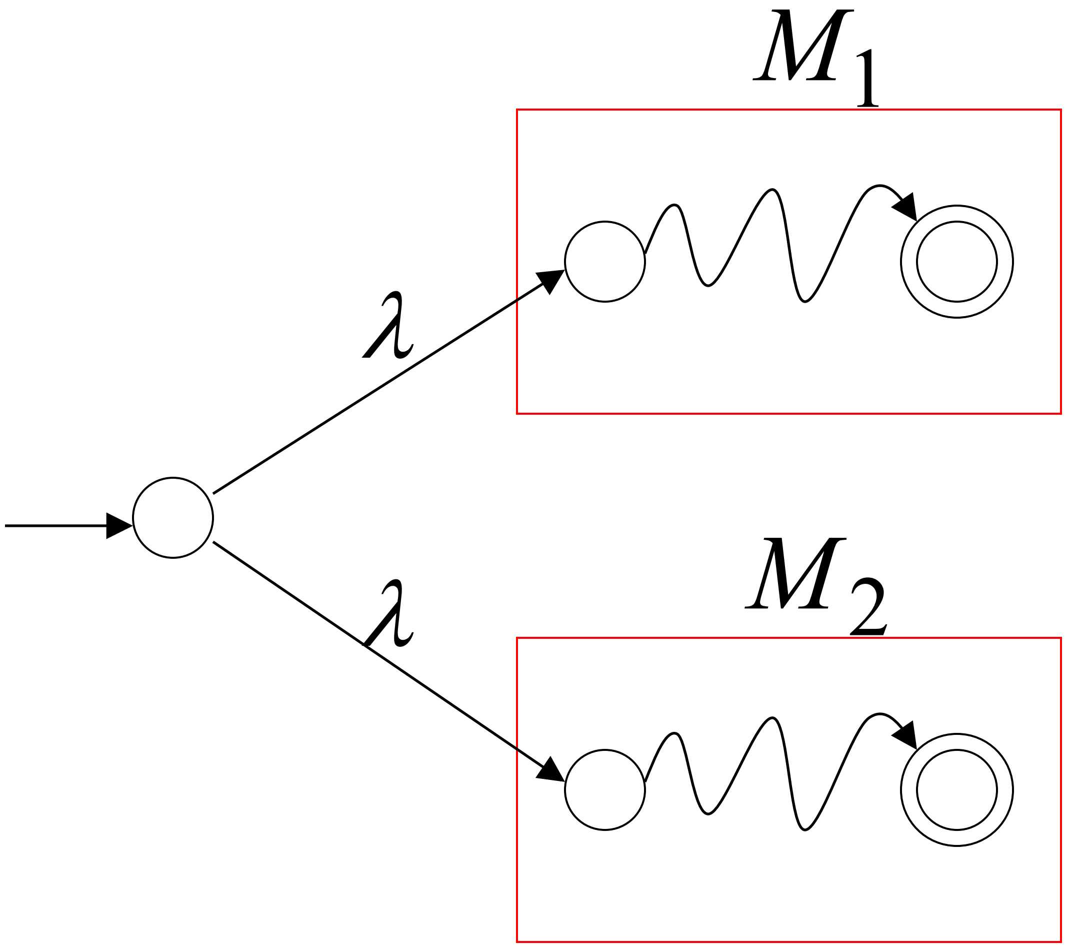


Nuova transizione


DFA M_1
DFA M_2
q_i
p_j
p_k
accettazione stato
accettazione stati
DFA M
q_i, p_j
q_i, p_k
nuovo accettazione stati

Unione $L_1 \cup L_2$

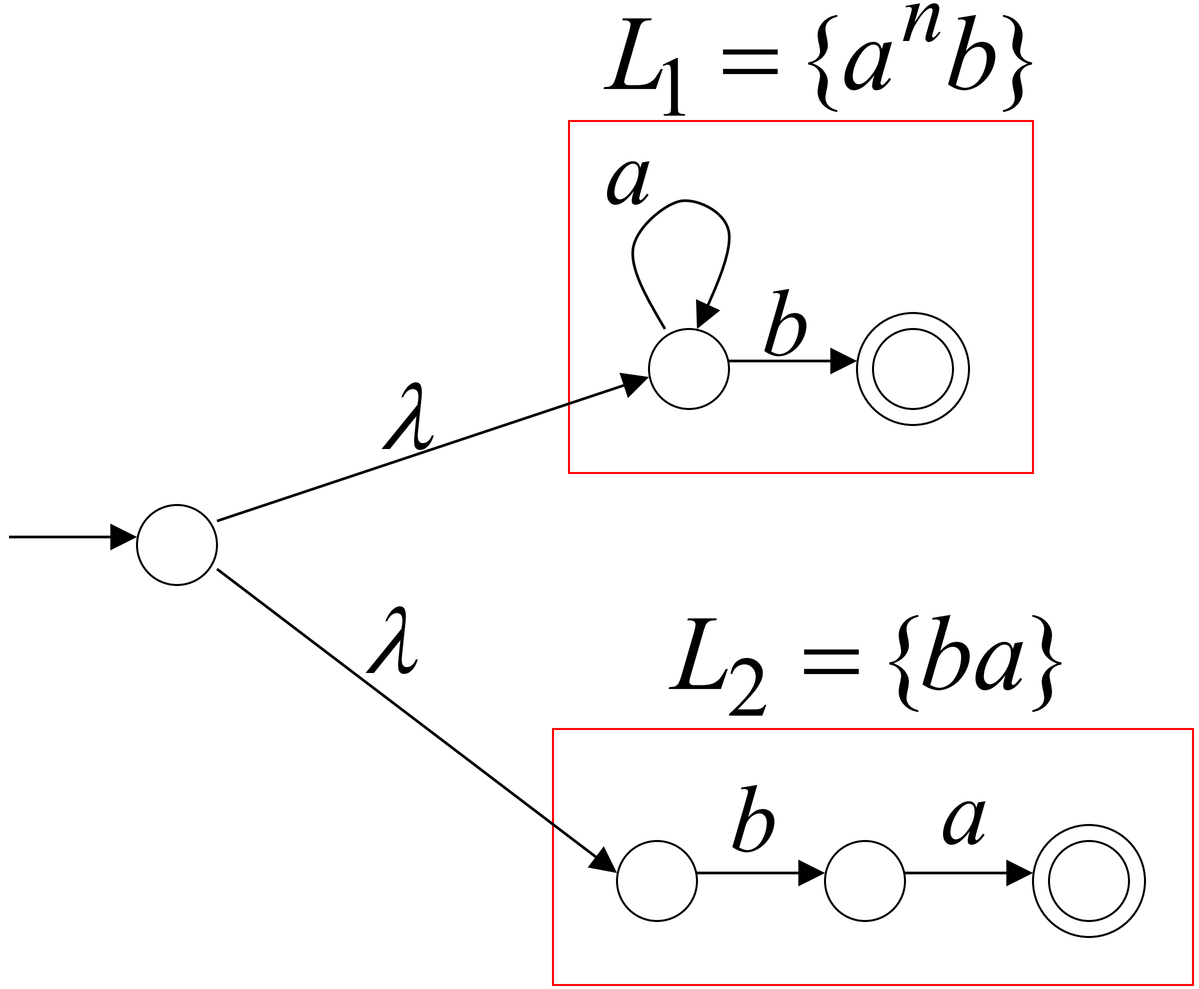
- NFA per



Esempio

NFA per $L_1 \cup L_2 = \{a^n b\} \cup \{ba\}$

-



Potremmo definire l'unione in un altro modo.
dati due automi «completi»
.

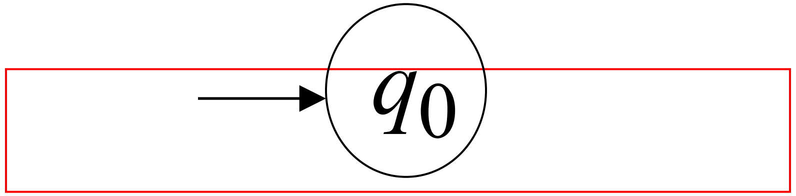
M_1 M_2

Stati in M

$$q_i, p_j$$

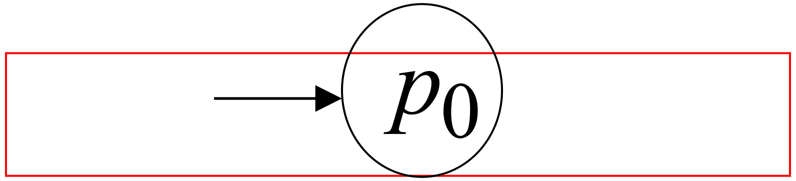
Stato in M_1 Stato in M_2

DFA M_1



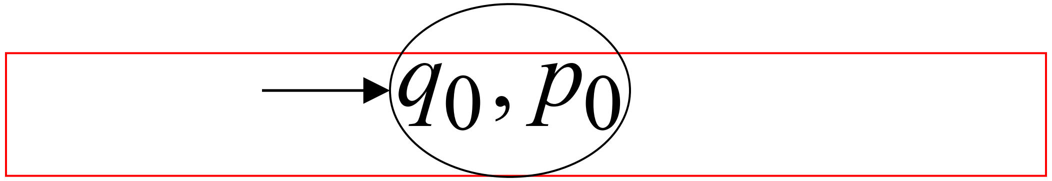
stato iniziale

DFA M_2



stato iniziale

DFA M



nuovo stato iniziale

DFA M_1

$$q_1 \xrightarrow{a} q_2$$

transizione

DFA M_2

$$p_1 \xrightarrow{a} p_2$$

transizione

DFA M

$$q_1, p_1 \xrightarrow{a} q_2, p_2$$

Nuova transizione

DFA M_1 DFA M_2

q_i p_k

accettazione stato accettazione stato

DFA M

q_i, p q, p_k

nuovi stati di accettazione

Con p uno stato qualsiasi di M_2

Con q uno stato qualsiasi di M_1

Provare che la definizione precedente definisce l'unione di due automi.

Parliamo come.