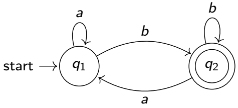
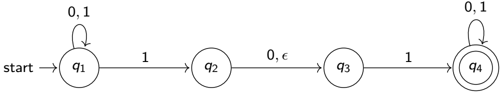
Computazione deterministica. Quando una macchina deterministica si trova in un dato stato e legge un certo simbolo, lo stato successivo è **univocamente determinato**.



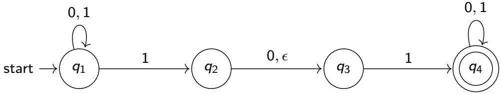
DFA $M = (Q, \Sigma, \delta, q_0, F)$: per ogni stato $q \in Q$ e **per ogni simbolo** $a \in \Sigma$ letto, esiste un **unico** stato (**univocamente determinato**) in cui si transisce: $\delta : Q \times \Sigma \to Q$.

Computazione non deterministica. Il non determinismo è una generalizzazione del determinismo. Un automa finito non deterministico (**NFA**) permette di avere

- 0, 1 o anche più archi uscenti per ciascun simbolo dell'alfabeto
- 0, 1 o anche più archi uscenti con etichetta ϵ (*epsilon*-transizioni)



Automi finiti non deterministici



Come computa un NFA?

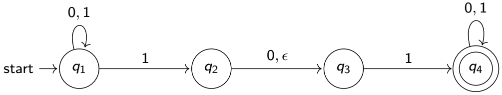
Se un NFA è in uno stato e, leggendo un simbolo a dell'alfabeto, si trova davanti a più scelte (più archi uscenti etichettati con a):

- si divide in più copie di se stesso: una copia per ciascuna scelta;
- ogni copia segue la computazione in parallelo indipendentemente dalle altre.

Se una copia di un NFA legge un simbolo che non si trova su alcun arco uscente da quello stato, allora quella copia della macchina **muore** insieme a tutto il suo ramo di computazione.

Accettazione.

- Se almeno una copia giunge in uno stato accettante, la stringa è accettata.
- Se nessuna copia giunge in uno stato accettante, allora la stringa non è accettata.

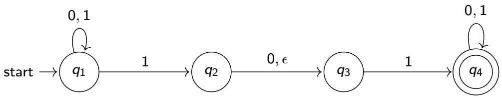


Come computa un NFA?

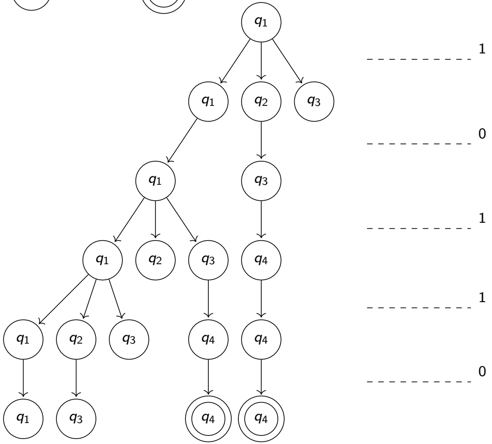
Se un NFA è in uno stato con uno o più archi etichettati con ϵ, si comporta come segue:

- senza leggere alcun simbolo dalla stringa input, la macchina si divide in più copie: una che segue ciascun arco etichettato con ϵ e una che resta nello stato corrente;
- ogni copia segue la computazione in parallelo indipendentemente dalle altre.

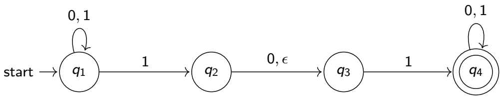
Esempio 1



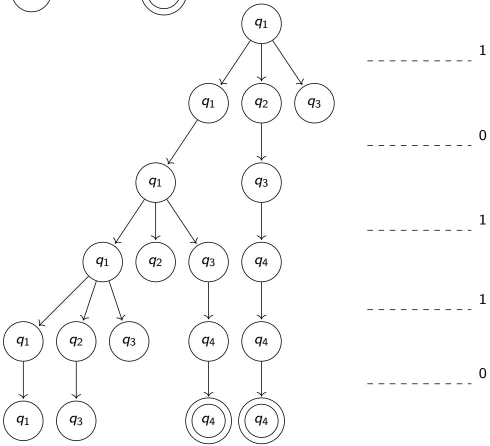
Stringa input: `10110`



Esempio 1

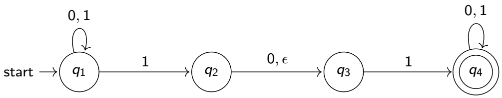


Stringa input: 10110

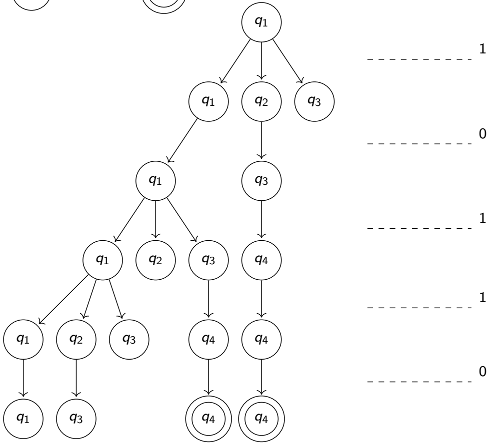


Quali stringhe accetta?

Esempio 1

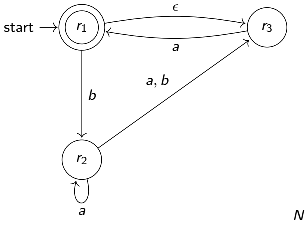


Stringa input: 10110



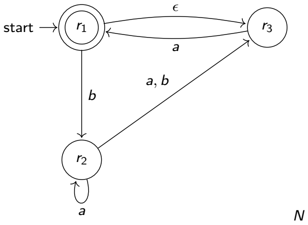
Quali stringhe accetta? Tutte le stringhe che hanno 101 oppure 11 come sottostringa.

Esempio 2



Stringa input: ϵ

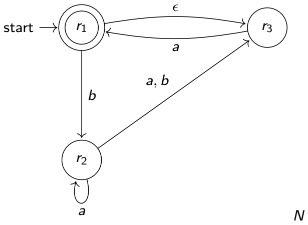
Esempio 2



Stringa input: ϵ

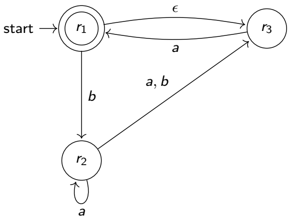
accettata perché r_1 è accettante.

Esempio 2



Stringa input: `a`

Esempio 2



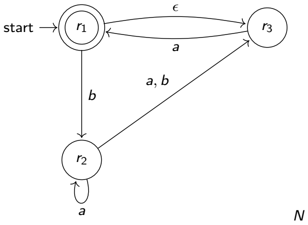
N

Stringa input: `a`

accettata perché:

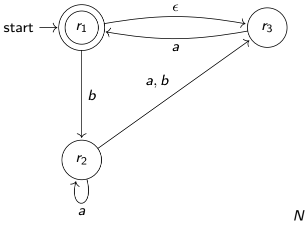
- r_1 ha una ϵ-transizione verso r_3: una copia N_A di N resta su r_1 e un'altra N_B va su r_3
- la copia N_A su r_1 legge `a` dall'input e muore;
- la copia N_B su r_3 legge `a` dall'input e va in r_1;
- input finito: N_B si trova sullo stato accettante r_1: la stringa è accettata.

Esempio 2



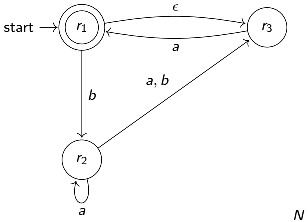
Stringa input: `aa`: accettata.

Esempio 2



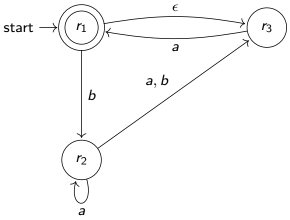
Stringa input: `baa`: `accettata`.

Esempio 2



Stringa input: `baba`: `accettata`.

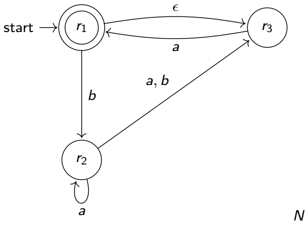
Esempio 2



N

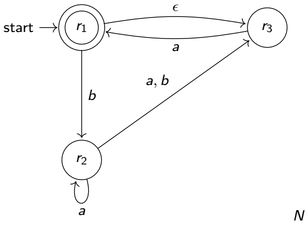
Stringa input: `b`: rifiutata.

Esempio 2



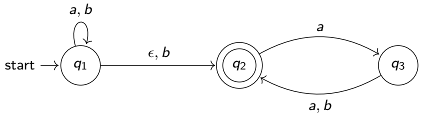
Stringa input: `ba`: rifiutata.

Esempio 2



Stringa input: `bb`: rifiutata.

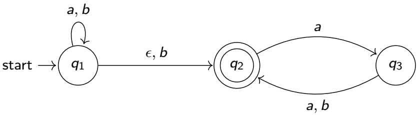
Automa finito non deterministico: definizione formale



Sia $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$

Un automa finito **non deterministico** è una quintupla $N = (Q, \Sigma, \delta, q_0, F)$.

Automa finito non deterministico: definizione formale

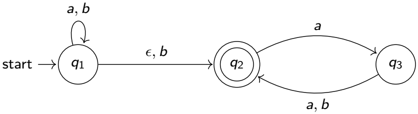


Sia $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$

Un automa finito **non deterministico** è una quintupla $N = (Q, \Sigma, \delta, q_0, F)$.

- Q è un insieme finito chiamato l'**insieme degli stati**

Automa finito non deterministico: definizione formale

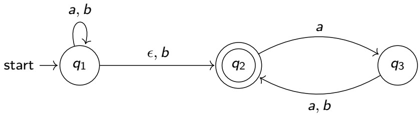


Sia $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$

Un automa finito **non deterministico** è una quintupla $N = (Q, \Sigma, \delta, q_0, F)$.

- Q è un insieme finito chiamato l'**insieme degli stati**
- Σ è un insieme finito chiamato l'**alfabeto**

Automa finito non deterministico: definizione formale

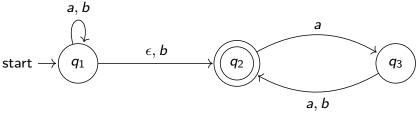


Sia $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$

Un automa finito **non deterministico** è una quintupla $N = (Q, \Sigma, \delta, q_0, F)$.

- Q è un insieme finito chiamato l'**insieme degli stati**
- Σ è un insieme finito chiamato l'**alfabeto**
- $\delta : Q \times \Sigma_\epsilon \to \mathcal{P}(Q)$ è la **funzione di transizione**

Automa finito non deterministico: definizione formale

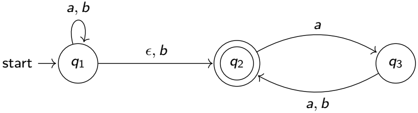


Sia $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$

Un automa finito **non deterministico** è una quintupla $N = (Q, \Sigma, \delta, q_0, F)$.

- Q è un insieme finito chiamato l'**insieme degli stati**
- Σ è un insieme finito chiamato l'**alfabeto**
- $\delta : Q \times \Sigma_\epsilon \to \mathcal{P}(Q)$ è la **funzione di transizione**
- $q_0 \in Q$ è lo **stato iniziale**

Automa finito non deterministico: definizione formale

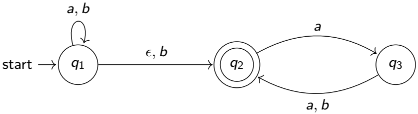


Sia $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$

Un automa finito **non deterministico** è una quintupla $N = (Q, \Sigma, \delta, q_0, F)$.

- Q è un insieme finito chiamato l'**insieme degli stati**
- Σ è un insieme finito chiamato l'**alfabeto**
- $\delta : Q \times \Sigma_\epsilon \to \mathcal{P}(Q)$ è la **funzione di transizione**
- $q_0 \in Q$ è lo **stato iniziale**
- $F \subseteq Q$ è l'**insieme degli stati accettanti** (o **finali**)

Automa finito non deterministico: definizione formale



Sia $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$

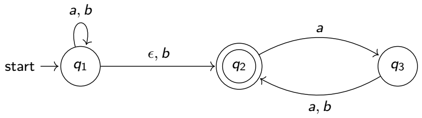
Un automa finito **non deterministico** è una quintupla $N = (Q, \Sigma, \delta, q_0, F)$.

- Q è un insieme finito chiamato l'**insieme degli stati**
- Σ è un insieme finito chiamato l'**alfabeto**
- $\delta : Q \times \Sigma_\epsilon \to \mathcal{P}(Q)$ è la **funzione di transizione**
- $q_0 \in Q$ è lo **stato iniziale**
- $F \subseteq Q$ è l'**insieme degli stati accettanti** (o **finali**)

La differenza tra DFA e NFA sta nella definizione della funzione di transizione. In un NFA la funzione δ

- ammette mosse di tipo ϵ (ϵ-transizioni);
- restituisce un insieme di stati (notare che $\emptyset \in \mathcal{P}(Q)$: nell'esempio $\delta(q_2, b) = \emptyset$).

Automa finito non deterministico: definizione formale



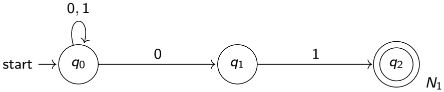
Sia $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$

Un automa finito **non deterministico** è una quintupla $N = (Q, \Sigma, \delta, q_0, F)$.

- Q è un insieme finito chiamato l'**insieme degli stati**
- Σ è un insieme finito chiamato l'**alfabeto**
- $\delta : Q \times \Sigma_\epsilon \to \mathcal{P}(Q)$ è la **funzione di transizione**
- $q_0 \in Q$ è lo **stato iniziale**
- $F \subseteq Q$ è l'**insieme degli stati accettanti** (o **finali**)

Osservazione

Il nondeterminismo è una generalizzazione del determinismo: ogni DFA è anche un NFA.

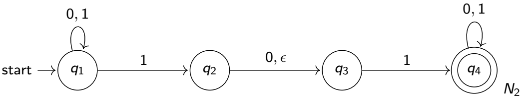


$N_1 = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_2\})$.

La funzione di transizione δ è così definita:

	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\emptyset$	$\{q_2\}$
$* \; q_2$	$\emptyset$	$\emptyset$

Esempio 2



$N_2 = (\{q_1, q_2, q_3, q_4\}, \{0, 1\}, \delta, q_0, \{q_2\})$.

La funzione di transizione δ è così definita:

	0	1	ϵ
$\rightarrow q_1$	$\{q_1\}$	$\{q_1, q_2\}$	$\emptyset$
q_2	$\{q_3\}$	$\emptyset$	$\{q_3\}$
q_3	$\emptyset$	$\{q_4\}$	$\emptyset$
$* q_4$	$\{q_4\}$	$\{q_4\}$	$\emptyset$

Come computa un NFA?

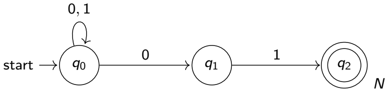
Sia $N = (Q, \Sigma, \delta, q_0, F)$ un NFA. Consideriamo la stringa input $w = w_1 w_2 \cdots w_n$, dove $w_i \in \Sigma$ per $i = 1, 2, \ldots, n$.

N accetta la stringa w se w può essere scritto come $w = y_1 y_2 \cdots y_m$, dove ciascun $y_i \in \Sigma_\epsilon$, e esiste una sequenza di stati $r_0, r_1, \ldots, r_m$ tale che:

- $r_0 = q_0$
- $r_{i+1} \in \delta(r_i, y_{i+1})$ per $i = 0, \ldots, m-1$
- $r_m \in F$

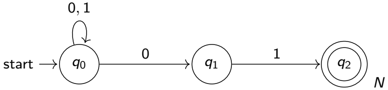
Se A è l'insieme di tutte le stringhe che la macchina accetta, diciamo che A è il **linguaggio della macchina** N e scriviamo $L(N) = A$.
Diciamo che N **riconosce** A.

Esempio



Quale linguaggio riconosce l'automa N?

Esempio

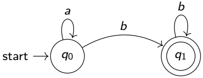


Quale linguaggio riconosce l'automa N? Accetta tutte le stringhe che finiscono con 01.
$L(N) = \{x01 | \ x \in \{0,1\}^*\}$

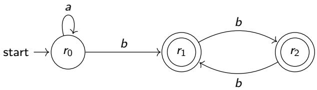
Equivalenza tra DFA e NFA

Definizione

Due macchine sono equivalenti se riconoscono lo stesso linguaggio.



N_1



N_2

$$L(N_1) = L(N_2) = \{a^n b^m |\ n \in \mathbb{N},\ m \in \mathbb{N}^+\}$$

Teorema

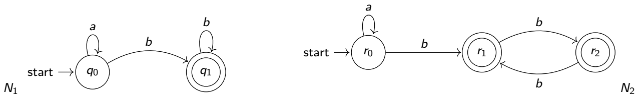
Ogni NFA N ha un equivalente DFA M; cioè, se N è un NFA, allora esiste un DFA M tale che $L(M) = L(N)$.

Come dimostrarlo?

Equivalenza tra DFA e NFA

Definizione

Due macchine sono equivalenti se riconoscono lo stesso linguaggio.



$L(N_1) = L(N_2) = \{a^n b^m |\ n \in \mathbb{N},\ m \in \mathbb{N}^+\}$

Teorema

Ogni NFA N ha un equivalente DFA M; cioè, se N è un NFA, allora esiste un DFA M tale che $L(M) = L(N)$.

Come dimostrarlo? Bisogna far vedere che dato un qualunque NFA N, siamo in grado di construire un DFA M che riesca a "simulare il comportamento di N".

Equivalenza tra DFA e NFA

1. Partiamo da un arbitrario NFA $N = (Q_N, \Sigma, \delta_N, q_N, F_N)$.
2. Costruiamo un corrispondente DFA $M = (Q_M, \Sigma, \delta_M, q_M, F_M)$ tale che $L(M) = L(N)$.

Equivalenza tra DFA e NFA

1. Partiamo da un arbitrario NFA $N = (Q_N, \Sigma, \delta_N, q_N, F_N)$.
2. Costruiamo un corrispondente DFA $M = (Q_M, \Sigma, \delta_M, q_M, F_M)$ tale che $L(M) = L(N)$.

Supponiamo prima il caso più semplice: N non ha ϵ-archi.

Equivalenza tra DFA e NFA

1. Partiamo da un arbitrario NFA $N = (Q_N, \Sigma, \delta_N, q_N, F_N)$.
2. Costruiamo un corrispondente DFA $M = (Q_M, \Sigma, \delta_M, q_M, F_M)$ tale che $L(M) = L(N)$.

Supponiamo prima il caso più semplice: N non ha ϵ-archi.

- $Q_M = \mathcal{P}(Q_N)$: ogni stato di M corrisponde ad un insieme di stati di N.

Equivalenza tra DFA e NFA

1. Partiamo da un arbitrario NFA $N = (Q_N, \Sigma, \delta_N, q_N, F_N)$.
2. Costruiamo un corrispondente DFA $M = (Q_M, \Sigma, \delta_M, q_M, F_M)$ tale che $L(M) = L(N)$.

Supponiamo prima il caso più semplice: N non ha ϵ-archi.

- $Q_M = \mathcal{P}(Q_N)$: ogni stato di M corrisponde ad un insieme di stati di N.
- Quando M si trova in uno stato $R \in Q_M$ e legge un simbolo $a \in \Sigma$, transisce nello stato $R' = \{q \in Q_N |\ q \in \delta_N(r, a),\ r \in R\}$, cioè

$$\delta_M(R, a) = \bigcup_{r \in R} \delta_N(r, a).$$

Equivalenza tra DFA e NFA

1. Partiamo da un arbitrario NFA $N = (Q_N, \Sigma, \delta_N, q_N, F_N)$.
2. Costruiamo un corrispondente DFA $M = (Q_M, \Sigma, \delta_M, q_M, F_M)$ tale che $L(M) = L(N)$.

Supponiamo prima il caso più semplice: N non ha ϵ-archi.

- $Q_M = \mathcal{P}(Q_N)$: ogni stato di M corrisponde ad un insieme di stati di N.
- Quando M si trova in uno stato $R \in Q_M$ e legge un simbolo $a \in \Sigma$, transisce nello stato $R' = \{q \in Q_N |\ q \in \delta_N(r, a),\ r \in R\}$, cioè

$$\delta_M(R, a) = \bigcup_{r \in R} \delta_N(r, a).$$

- $q_M = \{q_N\}$.

Equivalenza tra DFA e NFA

1. Partiamo da un arbitrario NFA $N = (Q_N, \Sigma, \delta_N, q_N, F_N)$.
2. Costruiamo un corrispondente DFA $M = (Q_M, \Sigma, \delta_M, q_M, F_M)$ tale che $L(M) = L(N)$.

Supponiamo prima il caso più semplice: N non ha ϵ-archi.

- $Q_M = \mathcal{P}(Q_N)$: ogni stato di M corrisponde ad un insieme di stati di N.
- Quando M si trova in uno stato $R \in Q_M$ e legge un simbolo $a \in \Sigma$, transisce nello stato $R' = \{q \in Q_N |\ q \in \delta_N(r, a),\ r \in R\}$, cioè

$$\delta_M(R, a) = \bigcup_{r \in R} \delta_N(r, a).$$

- $q_M = \{q_N\}$.
- $F_M = \{R \in Q_M |\ R \cap F_N \neq \emptyset\}$.

Equivalenza tra DFA e NFA

1. Partiamo da un arbitrario NFA $N = (Q_N, \Sigma, \delta_N, q_N, F_N)$.
2. Costruiamo un corrispondente DFA $M = (Q_M, \Sigma, \delta_M, q_M, F_M)$ tale che $L(M) = L(N)$.

Vediamo ora il caso generale: N può avere ϵ-archi.

Dato uno stato $R \in Q_M$, definiamo l'insieme di tutti gli stati che possono essere raggiunti dagli elementi di R usando 0 o più ϵ-archi (include gli elementi di R):

$$E(R) = \{q \in Q_N | \ q \text{ può essere raggiunto da } R \text{ usando 0 o più } \epsilon\text{-archi}\}.$$

Equivalenza tra DFA e NFA

1. Partiamo da un arbitrario NFA $N = (Q_N, \Sigma, \delta_N, q_N, F_N)$.
2. Costruiamo un corrispondente DFA $M = (Q_M, \Sigma, \delta_M, q_M, F_M)$ tale che $L(M) = L(N)$.

Vediamo ora il caso generale: N può avere ϵ-archi.

Dato uno stato $R \in Q_M$, definiamo l'insieme di tutti gli stati che possono essere raggiunti dagli elementi di R usando 0 o più ϵ-archi (include gli elementi di R):

$$E(R) = \{q \in Q_N |\ q \text{ può essere raggiunto da } R \text{ usando 0 o più } \epsilon\text{-archi}\}.$$

- $Q_M = \mathcal{P}(Q_N)$ (**come prima**).

Equivalenza tra DFA e NFA

1. Partiamo da un arbitrario NFA $N = (Q_N, \Sigma, \delta_N, q_N, F_N)$.
2. Costruiamo un corrispondente DFA $M = (Q_M, \Sigma, \delta_M, q_M, F_M)$ tale che $L(M) = L(N)$.

Vediamo ora il caso generale: N può avere ϵ-archi.

Dato uno stato $R \in Q_M$, definiamo l'insieme di tutti gli stati che possono essere raggiunti dagli elementi di R usando 0 o più ϵ-archi (include gli elementi di R):

$$E(R) = \{q \in Q_N | \ q \text{ può essere raggiunto da } R \text{ usando 0 o più } \epsilon\text{-archi}\}.$$

- $Q_M = \mathcal{P}(Q_N)$ (**come prima**).
- Quando M si trova in uno stato $R \in Q_M$ e legge un simbolo $a \in \Sigma$, transisce nello stato $R' = \{q \in Q_N | \ q \in E(\delta_N(r, a)), \ r \in R\}$, cioè

$$\delta_M(R, a) = \bigcup_{r \in R} E\left(\delta_N(r, a)\right).$$

1. Partiamo da un arbitrario NFA $N = (Q_N, \Sigma, \delta_N, q_N, F_N)$.
2. Costruiamo un corrispondente DFA $M = (Q_M, \Sigma, \delta_M, q_M, F_M)$ tale che $L(M) = L(N)$.

Vediamo ora il caso generale: N può avere ϵ-archi.

Dato uno stato $R \in Q_M$, definiamo l'insieme di tutti gli stati che possono essere raggiunti dagli elementi di R usando 0 o più ϵ-archi (include gli elementi di R):

$$E(R) = \{q \in Q_N | \ q \text{ può essere raggiunto da } R \text{ usando } 0 \text{ o più } \epsilon\text{-archi}\}.$$

- $Q_M = \mathcal{P}(Q_N)$ (**come prima**).
- Quando M si trova in uno stato $R \in Q_M$ e legge un simbolo $a \in \Sigma$, transisce nello stato $R' = \{q \in Q_N | \ q \in E(\delta_N(r, a)), \ r \in R\}$, cioè

$$\delta_M(R, a) = \bigcup_{r \in R} E\left(\delta_N(r, a)\right).$$

- $q_M = E(\{q_N\})$.

Equivalenza tra DFA e NFA

1. Partiamo da un arbitrario NFA $N = (Q_N, \Sigma, \delta_N, q_N, F_N)$.
2. Costruiamo un corrispondente DFA $M = (Q_M, \Sigma, \delta_M, q_M, F_M)$ tale che $L(M) = L(N)$.

Vediamo ora il caso generale: N può avere ϵ-archi.

Dato uno stato $R \in Q_M$, definiamo l'insieme di tutti gli stati che possono essere raggiunti dagli elementi di R usando 0 o più ϵ-archi (include gli elementi di R):

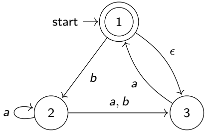
$$E(R) = \{q \in Q_N |\ q \text{ può essere raggiunto da } R \text{ usando 0 o più } \epsilon\text{-archi}\}.$$

- $Q_M = \mathcal{P}(Q_N)$ (**come prima**).
- Quando M si trova in uno stato $R \in Q_M$ e legge un simbolo $a \in \Sigma$, transisce nello stato $R' = \{q \in Q_N |\ q \in E(\delta_N(r, a)),\ r \in R\}$, cioè

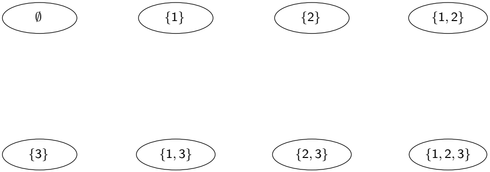
$$\delta_M(R, a) = \bigcup_{r \in R} E\left(\delta_N(r, a)\right).$$

- $q_M = E(\{q_N\})$.
- $F_M = \{R \in Q_M |\ R \cap F_N \neq \emptyset\}$ (**come prima**).

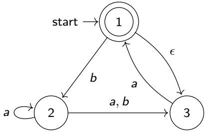
Esempio 1



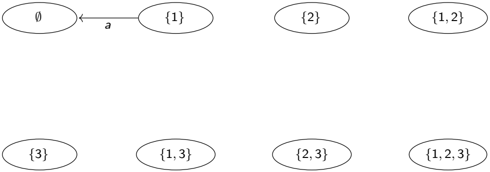
$$Q_M = \mathcal{P}(Q_N) = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}.$$



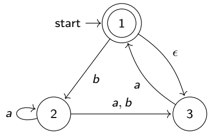
Esempio 1



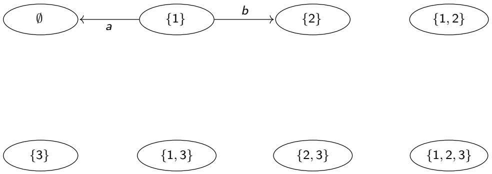
$$\delta_M(\{1\}, a) = E\left(\delta_N(1, a)\right) = E\left(\emptyset\right) = \emptyset$$



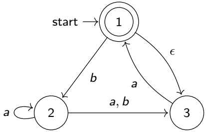
Esempio 1



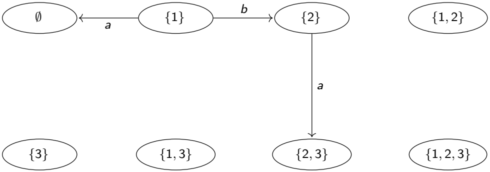
$$\delta_M(\{1\}, b) = E\left(\delta_N(1, b)\right) = E\left(\{2\}\right) = \{2\}$$



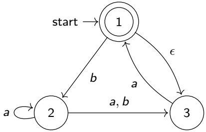
Esempio 1



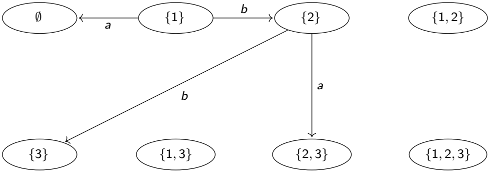
$$\delta_M(\{2\}, a) = E\left(\delta_N(2, a)\right) = \{2, 3\}$$



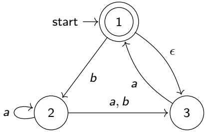
Esempio 1



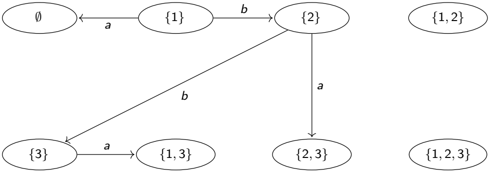
$$\delta_M(\{2\}, b) = E\left(\delta_N(2, b)\right) = \{3\}$$



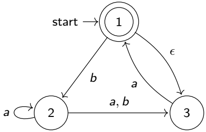
Esempio 1



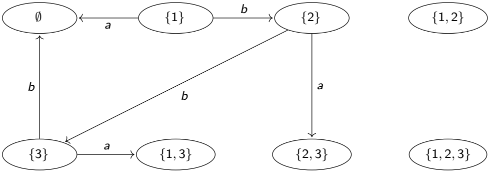
$$\delta_M(\{3\}, a) = E\left(\delta_N(3, a)\right) = E\left(\{1\}\right) = \{1, 3\}$$



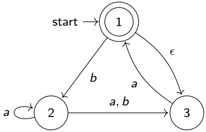
Esempio 1



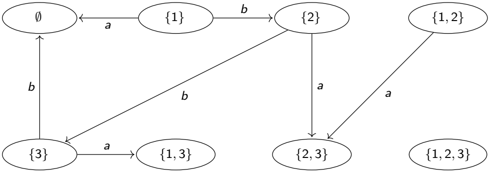
$$\delta_M(\{3\}, b) = E\left(\delta_N(3, b)\right) = E\left(\emptyset\right) = \emptyset$$



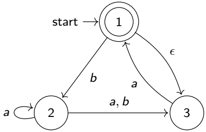
Esempio 1



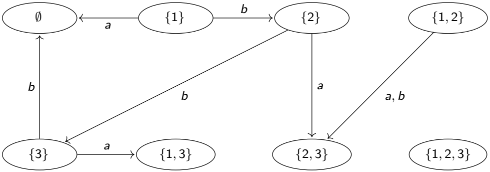
$$\delta_M(\{1,2\}, a) = E\left(\delta_N(1,a)\right) \cup E\left(\delta_N(2,a)\right) = E\left(\emptyset\right) \cup E\left(\{2,3\}\right) = \emptyset \cup \{2,3\} = \{2,3\}$$



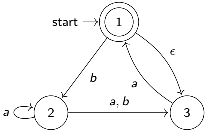
Esempio 1



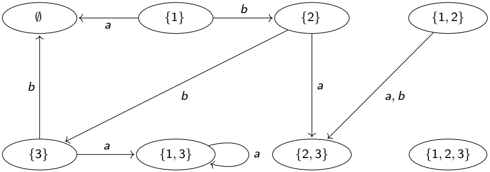
$$\delta_M(\{1,2\},b) = E\left(\delta_N(1,b)\right) \cup E\left(\delta_N(2,b)\right) = E\left(\{2\}\right) \cup E\left(\{3\}\right) = \{2\} \cup \{3\} = \{2,3\}$$



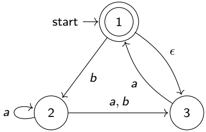
Esempio 1



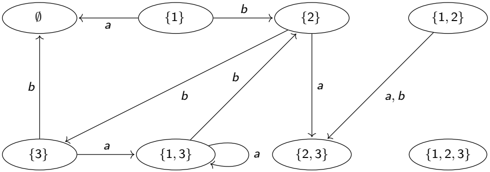
$$\delta_M(\{1,3\},a) = E\left(\delta_N(1,a)\right) \cup E\left(\delta_N(3,a)\right) = E\left(\emptyset\right) \cup E\left(\{1\}\right) = \emptyset \cup \{1,3\} = \{1,3\}$$



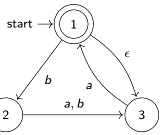
Esempio 1



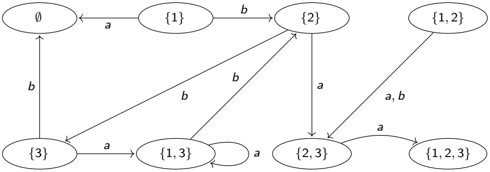
$$\delta_M(\{1,3\},b) = E\left(\delta_N(1,b)\right) \cup E\left(\delta_N(3,b)\right) = E\left(\{2\}\right) \cup E\left(\emptyset\right) = \{2\} \cup \emptyset = \{2\}$$



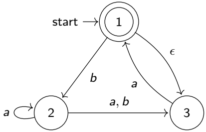
Esempio 1



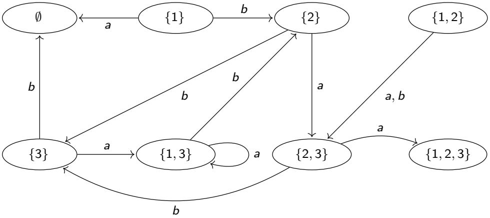
$$\delta_M(\{2,3\}, a) = E\left(\delta_N(2,a)\right) \cup E\left(\delta_N(3,a)\right) = E\left(\{2,3\}\right) \cup E\left(\{1\}\right) = \{2,3\} \cup \{1,3\} = \{1,2,3\}$$



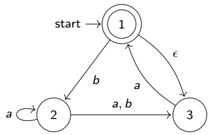
Esempio 1



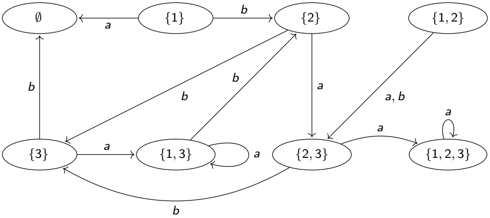
$$\delta_M(\{2,3\}, b) = E\left(\delta_N(2,b)\right) \cup E\left(\delta_N(3,b)\right) = E\left(\{3\}\right) \cup E\left(\emptyset\right) = \{3\} \cup \emptyset = \{3\}$$



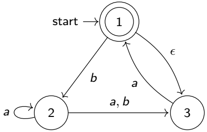
Esempio 1



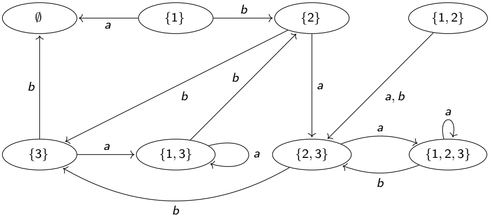
$$\delta_M(\{1,2,3\},a) = E\left(\delta_N(1,a)\right) \cup E\left(\delta_N(2,a)\right) \cup E\left(\delta_N(3,a)\right) =$$
$$E\left(\emptyset\right) \cup E\left(\{2,3\}\right) \cup E\left(\{1\}\right) = \emptyset \cup \{2,3\} \cup \{1,3\} = \{1,2,3\}$$



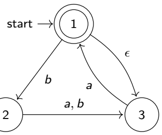
Esempio 1

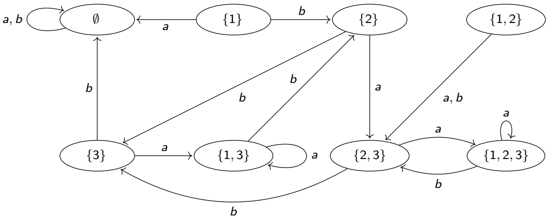


$\delta_M(\{1,2,3\},b) = E\left(\delta_N(1,b)\right) \cup E\left(\delta_N(2,b)\right) \cup E\left(\delta_N(3,b)\right) =$
$E\left(\{2\}\right) \cup E\left(\{3\}\right) \cup E\left(\emptyset\right) = \{2\} \cup \{3\} \cup \emptyset = \{2,3\}$

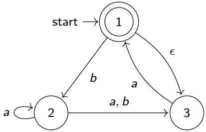


Esempio 1

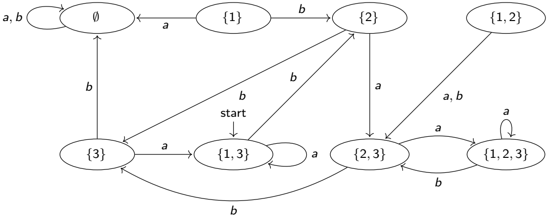




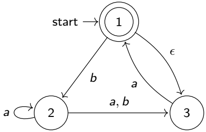
Esempio 1



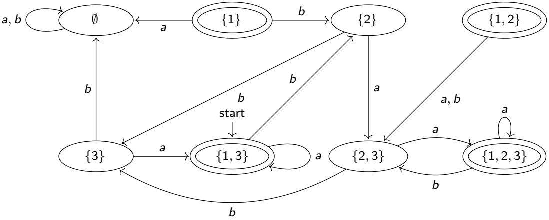
$$q_M = E(\{q_N\}) = E(\{1\}) = \{1, 3\}.$$



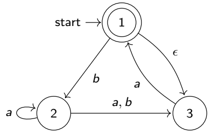
Esempio 1



$$F_N = \{R \in Q_M |\ R \cap F_N \neq \emptyset\} = \{\{1\}, \{1,2\}, \{1,3\}, \{1,2,3\}\}.$$

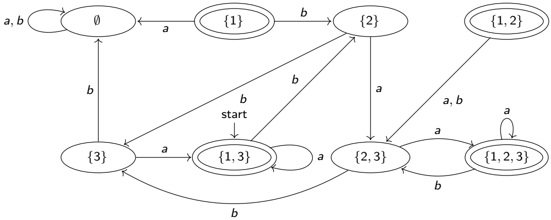


Esempio 1

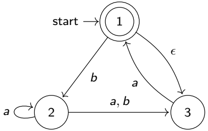


Gli stati $\{1\}$ e $\{1,2\}$ possono essere eliminati perché non hanno archi entranti. In generale, per ogni NFA N esiste un DFA M tale che

$$|Q_M| \leq |\mathcal{P}(Q_N)| = 2^{|Q_N|}.$$

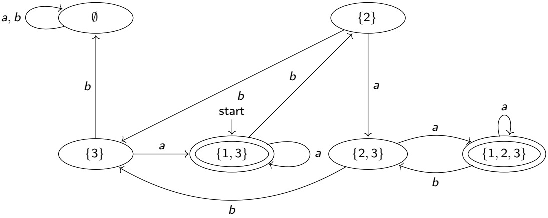


Esempio 1

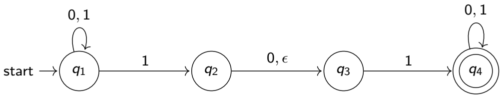


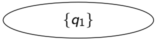
Gli stati $\{1\}$ e $\{1, 2\}$ possono essere eliminati perché non hanno archi entranti. In generale, per ogni NFA N esiste un DFA M tale che

$$|Q_M| \leq |\mathcal{P}(Q_N)| = 2^{|Q_N|}.$$

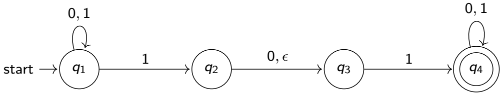


Esempio 2

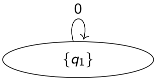




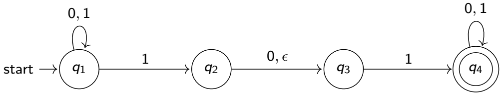
Esempio 2



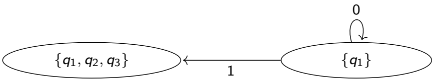
$$\delta_M(\{q_1\}, 0) = E(\delta_N(q_1, 0)) = \{q_1\}$$



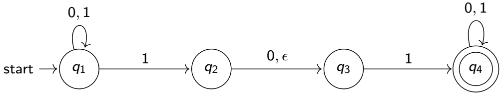
Esempio 2



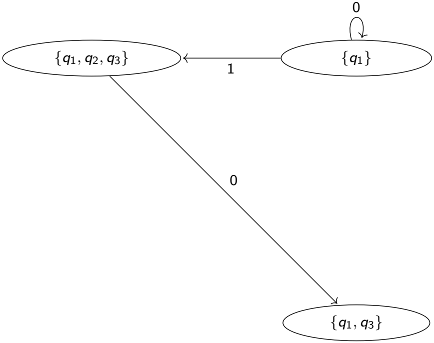
$$\delta_M(\{q_1\}, 1) = E(\delta_N(q_1, 1)) = E(\{q_1, q_2\}) = \{q_1, q_2, q_3\}$$



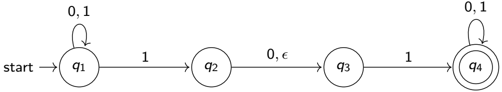
Esempio 2



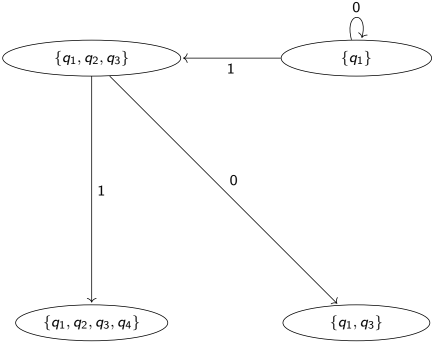
$$\delta_M(\{q_1, q_2, q_3\}, 0) = E(\delta_N(q_1, 0)) \cup E(\delta_N(q_2, 0)) \cup E(\delta_N(q_3, 0))\{q_1\} \cup \{q_3\} \cup \emptyset = \{q_1, q_3\}$$



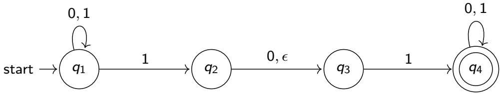
Esempio 2



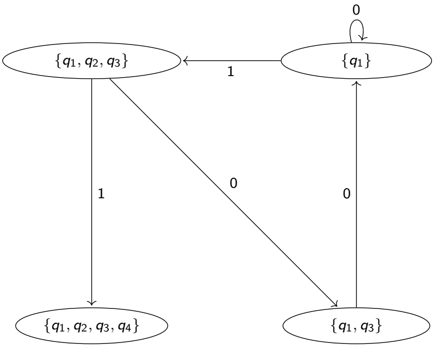
$\delta_M(\{q_1, q_2, q_3\}, 1) = E(\delta_N(q_1, 1)) \cup E(\delta_N(q_2, 1)) \cup E(\delta_N(q_3, 1)) =$
$\{q_1, q_2, q_3\} \cup \emptyset \cup \{q_4\} = \{q_1, q_2, q_3, q_4\}$



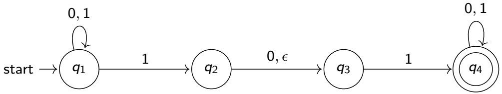
Esempio 2



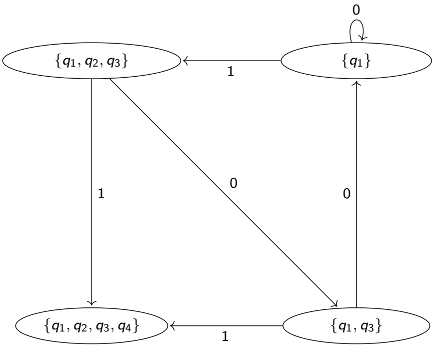
$$\delta_M(\{q_1, q_3\}, 0) = E(\delta_N(q_1, 0)) \cup E(\delta_N(q_3, 0)) = \{q_1\} \cup \emptyset = \{q_1\}$$



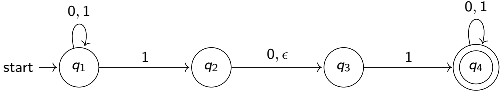
Esempio 2



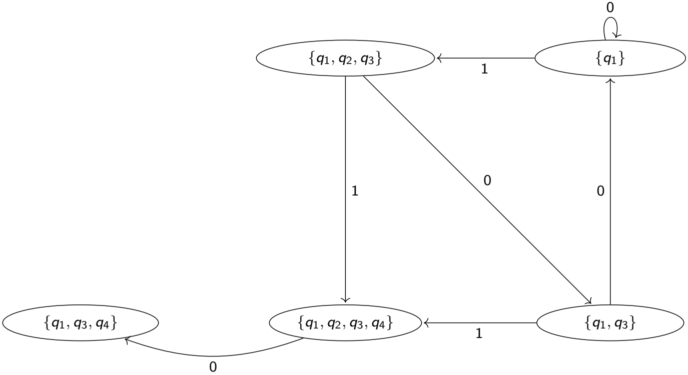
$$\delta_M(\{q_1, q_3\}, 1) = E(\delta_N(q_1, 1)) \cup E(\delta_N(q_3, 1)) = \{q_1, q_2, q_3\} \cup \{q_4\} = \{q_1, q_2, q_3, q_4\}$$



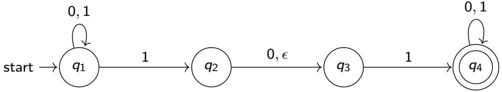
Esempio 2



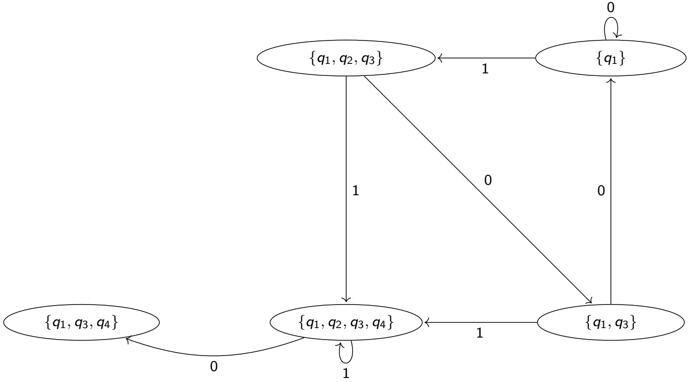
$$\delta_M(\{q_1, q_2, q_3, q_4\}, 0) = E(\delta_N(q_1, 0)) \cup E(\delta_N(q_2, 0)) \cup =$$
$$E(\delta_N(q_3, 0)) \cup E(\delta_N(q_4, 0)) = \{q_1\} \cup \{q_3\} \cup \emptyset \cup \{q_4\} = \{q_1, q_3, q_4\}$$



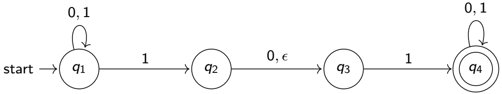
Esempio 2



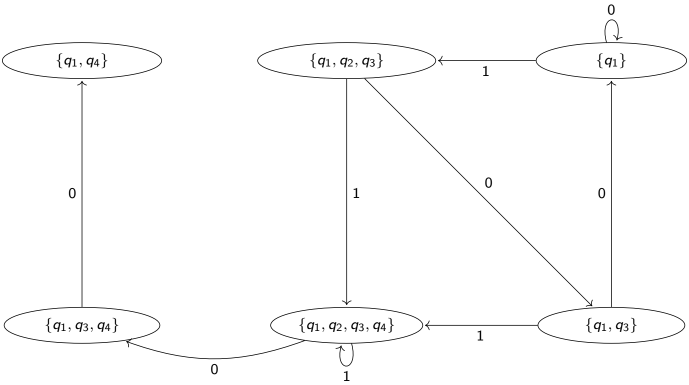
$\delta_M(\{q_1, q_2, q_3, q_4\}, 1) = E(\delta_N(q_1, 1)) \cup E(\delta_N(q_2, 1)) \cup =$
$E(\delta_N(q_3, 1)) \cup E(\delta_N(q_4, 1)) = \{q_1, q_2, q_3\} \cup \emptyset \cup \{q_4\} \cup \{q_4\} = \{q_1, q_2, q_3, q_4\}$



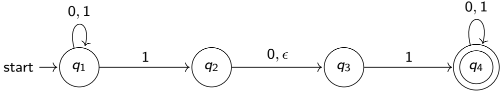
Esempio 2



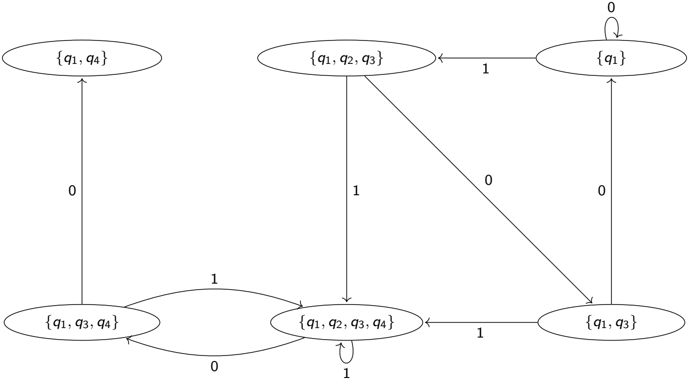
$\delta_M(\{q_1, q_3, q_4\}, 0) = E(\delta_N(q_1, 0)) \cup E(\delta_N(q_3, 0)) \cup E(\delta_N(q_4, 0)) = \{q_1\} \cup \emptyset \cup \{q_4\} = \{q_1, q_4\}$



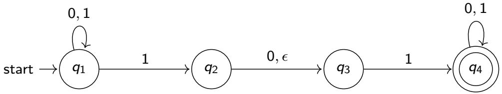
Esempio 2



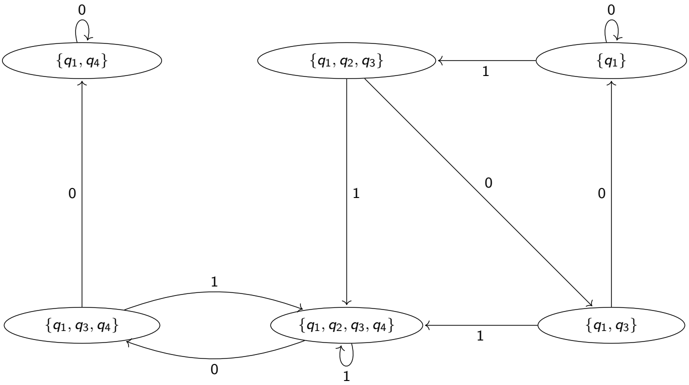
$$\delta_M(\{q_1, q_3, q_4\}, 1) = E(\delta_N(q_1, 1)) \cup E(\delta_N(q_3, 1)) \cup E(\delta_N(q_4, 1)) =$$
$$\{q_1, q_2, q_3\} \cup \{q_4\} \cup \{q_4\} = \{q_1, q_2, q_3, q_4\}$$



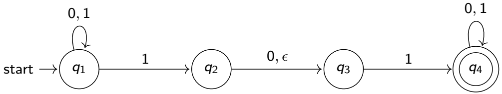
Esempio 2



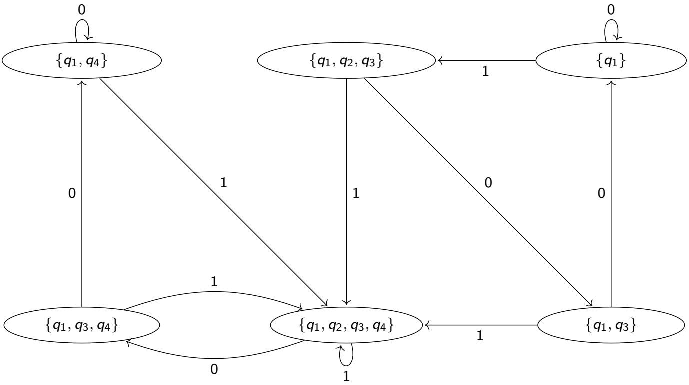
$$\delta_M(\{q_1, q_4\}, 0) = E(\delta_N(q_1, 0)) \cup E(\delta_N(q_4, 0)) = \{q_1\} \cup \{q_4\} = \{q_1, q_4\}$$



Esempio 2



$$\delta_M(\{q_1, q_4\}, 1) = E(\delta_N(q_1, 1)) \cup E(\delta_N(q_4, 1)) = \{q_1, q_2, q_3\} \cup \{q_4\} = \{q_1, q_2, q_3, q_4\}$$



Equivalenza tra DFA e NFA

1. $N = (Q_N, \Sigma, \delta_N, q_N, F_N)$.
2. Costruiamo un corrispondente DFA $M = (Q_M, \Sigma, \delta_M, q_M, F_M)$ tale che

- $Q_M = \mathcal{P}(Q_N)$.
- $\delta_M(R, a) = \bigcup_{r \in R} E\left(\delta_N(r, a)\right)$.
- $q_M = E(\{q_N\})$.
- $F_N = \{R \in Q_M \mid R \cap F_N \neq \emptyset\}$.

Dimostriamo formalmente che $L(N) = L(M)$.

Funzione di transizione estesa

La funzione di transizione δ di un DFA o un NFA può essere estesa per operare direttamente su stringhe (invece che su simboli).

Funzione di transizione estesa

La funzione di transizione δ di un DFA o un NFA può essere estesa per operare direttamente su stringhe (invece che su simboli).

Dato un DFA M, definiamo la funzione di transizione estesa δ^*_M in modo che per uno stato q e una stringa w, $\delta^*_M(q, w)$ sia lo stato raggiungibile quando, partendo da q, si abbia w come input.

Funzione di transizione estesa

La funzione di transizione δ di un DFA o un NFA può essere estesa per operare direttamente su stringhe (invece che su simboli).

Dato un DFA M, definiamo la funzione di transizione estesa δ^*_M in modo che per uno stato q e una stringa w, $\delta^*_M(q, w)$ sia lo stato raggiungibile quando, partendo da q, si abbia w come input.

Usando la funzione di transizione estesa, il linguaggio riconosciuto da un DFA $M = (Q_M, \Sigma, \delta_M, q_M, F_M)$ diventa:

$$L(M) = \{w \mid \delta^*_M(q_M, w) \in F_M\}.$$

Funzione di transizione estesa

La funzione di transizione δ di un DFA o un NFA può essere estesa per operare direttamente su stringhe (invece che su simboli).

Dato un DFA M, definiamo la funzione di transizione estesa δ_M^* in modo che per uno stato q e una stringa w, $\delta_M^*(q, w)$ sia lo stato raggiungibile quando, partendo da q, si abbia w come input.

Usando la funzione di transizione estesa, il linguaggio riconosciuto da un DFA $M = (Q_M, \Sigma, \delta_M, q_M, F_M)$ diventa:

$$L(M) = \{w \mid \delta_M^*(q_M, w) \in F_M\}.$$

Dato un NFA N, definiamo la funzione di transizione estesa δ_N^* in modo che per uno stato q e una stringa w, $\delta_N^*(q, w)$ sia l'insieme degli stati raggiungibili quando, partendo da q, si abbia w come input.

Funzione di transizione estesa

La funzione di transizione δ di un DFA o un NFA può essere estesa per operare direttamente su stringhe (invece che su simboli).

Dato un DFA M, definiamo la funzione di transizione estesa δ^*_M in modo che per uno stato q e una stringa w, $\delta^*_M(q, w)$ sia lo stato raggiungibile quando, partendo da q, si abbia w come input.

Usando la funzione di transizione estesa, il linguaggio riconosciuto da un DFA $M = (Q_M, \Sigma, \delta_M, q_M, F_M)$ diventa:

$$L(M) = \{w \mid \delta^*_M(q_M, w) \in F_M\}.$$

Dato un NFA N, definiamo la funzione di transizione estesa δ^*_N in modo che per uno stato q e una stringa w, $\delta^*_N(q, w)$ sia l'insieme degli stati raggiungibili quando, partendo da q, si abbia w come input.

Usando la funzione di transizione estesa, il linguaggio riconosciuto da un NFA $N = (Q_N, \Sigma, \delta_N, q_N, F_N)$ diventa:

$$L(N) = \{w \mid \delta^*(q_N, w) \cap F_N \neq \emptyset\}.$$

Funzione di transizione estesa (definizione ricorsiva)

Definizione (Funzione di transizione estesa per DFA)

Dato un DFA $(Q_M, \Sigma, \delta_M, q_M, F_M)$*, la funzione di transizione estesa* δ^*_M *è così definita:*

- $\delta^*_M(q, \epsilon) = q$ *per ogni* $q \in Q_M$*;*
- *per ogni stringa* $w = \mathbf{x}a$*, dove* $a \in \Sigma$*,*

$$\delta^*_M(q, \mathbf{x}a) = \delta_M(\delta^*_M(q, \mathbf{x}), a).$$

Funzione di transizione estesa (definizione ricorsiva)

Definizione (Funzione di transizione estesa per DFA)

Dato un DFA $(Q_M, \Sigma, \delta_M, q_M, F_M)$, *la funzione di transizione estesa* δ^*_M *è così definita:*

- $\delta^*_M(q, \epsilon) = q$ *per ogni* $q \in Q_M$;
- *per ogni stringa* $w = \mathbf{x}a$, *dove* $a \in \Sigma$,

$$\delta^*_M(q, \mathbf{x}a) = \delta_M(\delta^*_M(q, \mathbf{x}), a).$$

Definizione (Funzione di transizione estesa per NFA)

Dato un NFA $(Q_N, \Sigma, \delta_N, q_N, F_N)$, *la funzione di transizione estesa* δ^*_N *è così definita:*

- $\delta^*_N(q, \epsilon) = E(\delta_N(q, \epsilon)) = E(\{q\})$ *per ogni* $q \in Q_N$;
- *per ogni stringa* $w = \mathbf{x}a$, *dove* $a \in \Sigma$,

$$\delta^*_N(q, \mathbf{x}a) = \cup_{r \in \delta^*_N(q, \mathbf{x})} E(\delta_N(r, a)).$$

Torniamo all'equivalenza tra DFA e NFA

1. $N = (Q_N, \Sigma, \delta_N, q_N, F_N)$.
2. Abbiamo costruito un corrispondente DFA $M = (Q_M, \Sigma, \delta_M, q_M, F_M)$ tale che

- $Q_M = \mathcal{P}(Q_N)$.
- $\delta_M(R, a) = \bigcup_{r \in R} E\left(\delta_N(r, a)\right)$.
- $q_M = E(\{q_N\})$.
- $F_N = \{R \in Q_M |\ R \cap F_N \neq \emptyset\}$.

Dobbiamo dimostrare che una stringa $w \in \Sigma^*$ è accettata da N se e solo se w è accettata da M.

Usando la funzione di transizione estesa, è sufficiente dimostrare per induzione che

$$\delta_N^*(q_N, w) = \delta_M^*(q_M, w).$$

Equivalenza tra DFA e NFA

$$\delta_N^*(q_N, w) = \delta_M^*(q_M, w).$$

Induzione sulla lunghezza di w.

Base: $|w| = 0$, cioè $w = \epsilon$
$\delta_N^*(q_N, \epsilon) = E(\{q_N\})$
$\delta_M^*(q_M, \epsilon) = q_M = E(\{q_N\})$

Passo induttivo: supponiamo l'equivalenza vera per $\mathbf{x} \in \Sigma^*$ e proviamo che è anche vera per $\mathbf{x}a$, dove $a \in \Sigma$.

$$\begin{array}{rcll}
\delta_M^*(q_M, \mathbf{x}a) & = & \delta_M(\delta_M^*(q_M, \mathbf{x}), a) & \text{def. di } \delta_M^* \\
& = & \delta_M(\delta_N^*(q_N, \mathbf{x}), a) & \text{passo induttivo} \\
& = & \displaystyle\bigcup_{r \in \delta_N^*(q_N, \mathbf{x})} E(\delta_N(r, a)) & \text{def. di } \delta_M \\
& = & \delta_N^*(q_N, \mathbf{x}a) & \text{def. di } \delta_N^*.
\end{array}$$

Ancora sull'equivalenza tra DFA e NFA

Sappiamo che dato un NFA qualsiasi, possiamo sempre definire un DFA equivalente.

Ancora sull'equivalenza tra DFA e NFA

Sappiamo che dato un NFA qualsiasi, possiamo sempre definire un DFA equivalente.

Ma **attenzione**: il DFA equivalente può essere esponenzialmente meno efficiente in termini di numero di stati.

Ancora sull'equivalenza tra DFA e NFA

Sappiamo che dato un NFA qualsiasi, possiamo sempre definire un DFA equivalente.

Ma **attenzione**: il DFA equivalente può essere esponenzialmente meno efficiente in termini di numero di stati.

Possiamo dimostrare che esiste un NFA con $n + 1$ stati che non ha nessun DFA equivalente con meno di 2^n stati!

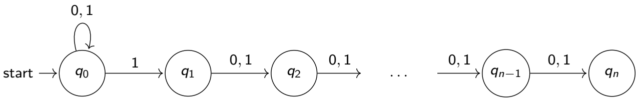
Ancora sull'equivalenza tra DFA e NFA

Sappiamo che dato un NFA qualsiasi, possiamo sempre definire un DFA equivalente.

Ma **attenzione**: il DFA equivalente può essere esponenzialmente meno efficiente in termini di numero di stati.

Possiamo dimostrare che esiste un NFA con $n + 1$ stati che non ha nessun DFA equivalente con meno di 2^n stati!

Sia $N = (Q_N, \Sigma, \delta_N, q_N, F_N)$ il seguente NFA che accetta tutte le stringhe binarie che hanno un 1 nella n-esima posizione dalla fine.



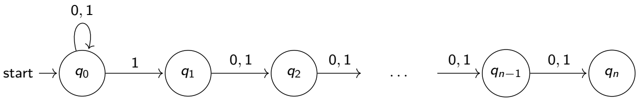
Ancora sull'equivalenza tra DFA e NFA

Sappiamo che dato un NFA qualsiasi, possiamo sempre definire un DFA equivalente.

Ma **attenzione**: il DFA equivalente può essere esponenzialmente meno efficiente in termini di numero di stati.

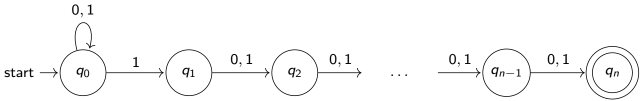
Possiamo dimostrare che esiste un NFA con $n+1$ stati che non ha nessun DFA equivalente con meno di 2^n stati!

Sia $N = (Q_N, \Sigma, \delta_N, q_N, F_N)$ il seguente NFA che accetta tutte le stringhe binarie che hanno un 1 nella n-esima posizione dalla fine.



Supponiamo per assurdo che esista un DFA M equivalente con meno di 2^n stati. Faremo vedere che esiste una stringa accettata da N ma non da M: N e M non sono equivalenti (contraddizione).

Ancora sull'equivalenza tra DFA e NFA



Supponiamo per assurdo che esista un DFA $M = (Q_M, \Sigma, \delta_M, q_M, F_M)$ equivalente a N tale che $|Q_M| < 2^n$.

Tutte le stringhe di n bit sono 2^n mentre $|Q_M| < 2^n$. Quindi, esistono due stringhe diverse di n bit, v e w, che se date in input a M fanno terminare la computazione nello stesso stato $q \in Q_M$ (entrambe accettate o entrambe rifiutate).

Sia $v = v_1 v_2 \cdots v_n$ e $w = w_1 w_2 \cdots w_n$ con $v_i, w_j \in \{0, 1\}$. Essendo diverse, v e w devono differire su almeno un bit: esiste $1 \leq i \leq n$ tale che $v_i \neq w_i$.

Ancora sull'equivalenza tra DFA e NFA

$$\begin{array}{cccccccc} v_1 & v_2 & \cdots & v_{i-1} & v_i & v_{i+1} & \cdots & v_n \\ w_1 & w_2 & \cdots & w_{i-1} & w_i & w_{i+1} & \cdots & w_n \end{array}$$

Su entrambe le stringhe, il DFA M, dopo aver letto l'ultimo simbolo, termina nello stesso stato q.

Prendiamo una qualunque stringa $x = x_1 x_2 \cdots x_i$ di i bit e appendiamola alla fine di v e w in modo che v_i e w_i siano nella n-esima posizione dalla fine.

$$\begin{array}{ccccccccccc} v_1 & v_2 & \cdots & v_{i-1} & v_i & v_{i+1} & \cdots & v_n\ x_1 & x_2 & \cdots & x_i \\ w_1 & w_2 & \cdots & w_{i-1} & w_i & w_{i+1} & \cdots & w_n\ x_1 & x_2 & \cdots & x_i \end{array}$$

Abbiamo ottenuto due nuove stringhe $v \circ x$ e $w \circ x$. Su M, in entrambi i casi, la computazione finisce nello stesso stato q: finisce in q fino a v_n, w_n e poi le stringhe seguono con gli stessi simboli. Quindi, le stringhe o sono entrambe accettate o entrambe rifiutate da M.

Supponiamo senza perdita di generalità che $v_i = 1$ e $w_i = 0$. Su N, $v \circ x$ è accettata mentre $w \circ x$ è rifiutata: contraddizione.

Teorema

Un linguaggio è **regolare** *se e solo se esiste un* **automa finito non deterministico** *che lo riconosce.*

Dimostrazione.
Sia L un linguaggio regolare. Allora (per definizione) esiste un DFA che lo riconosce.

Ma ogni DFA è anche un NFA, quindi esiste un NFA che riconosce L. Quindi:

$$L \text{ è regolare} \implies \text{esiste un NFA che riconosce } L.$$

Sia N un NFA che riconosce un linguaggio L. Abbiamo dimostrato che ogni NFA ha un equivalente DFA. Quindi esiste un DFA M che riconosce lo stesso linguaggio L. Quindi, per definizione, L è regolare. Quindi:

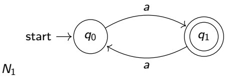
$$\text{esiste un NFA che riconosce } L \implies L \text{ è regolare }.$$

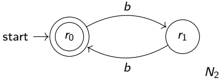
Linguaggi regolari chiusi per concatenazione

Teorema

La classe dei linguaggi regolari è chiusa per l'operazione di concatenazione. Cioè, se L_1 e L_2 sono linguaggi regolari, allora lo è anche $L_1 \circ L_2$.

Vediamo un esempio.





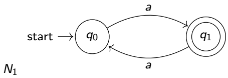
$L(N_1) = \{a^n |\ n \text{ dispari}\}$ e $L(N_2) = \{b^n |\ n \text{ pari}\}$. Vogliamo costruire un NFA N_3 che riconosca il linguaggio $L(N_1) \circ L(N_2)$.

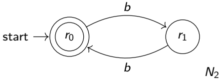
Linguaggi regolari chiusi per concatenazione

Teorema

La classe dei linguaggi regolari è chiusa per l'operazione di concatenazione. Cioè, se L_1 e L_2 sono linguaggi regolari, allora lo è anche $L_1 \circ L_2$.

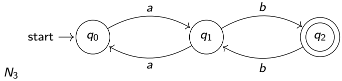
Vediamo un esempio.





$L(N_1) = \{a^n | \ n \text{ dispari}\}$ e $L(N_2) = \{b^n | \ n \text{ pari}\}$. Vogliamo costruire un NFA N_3 che riconosca il linguaggio $L(N_1) \circ L(N_2)$.

Tentativo errato:

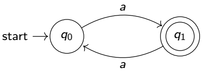


N_3 riconosce abbaab $\notin L(N_1) \circ L(N_2)$.

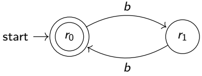
Teorema

La classe dei linguaggi regolari è chiusa per l'operazione di concatenazione. Cioè, se L_1 e L_2 sono linguaggi regolari, allora lo è anche $L_1 \circ L_2$.

Vediamo un esempio.



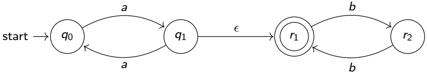
N_1



N_2

$L(N_1) = \{a^n | \; n$ dispari$\}$ e $L(N_2) = \{b^n | \; n$ pari$\}$. Vogliamo costruire un NFA N_3 che riconosca il linguaggio $L(N_1) \circ L(N_2)$.

Costruzione corretta:



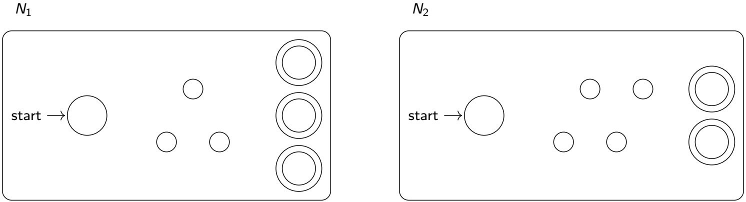
N_3

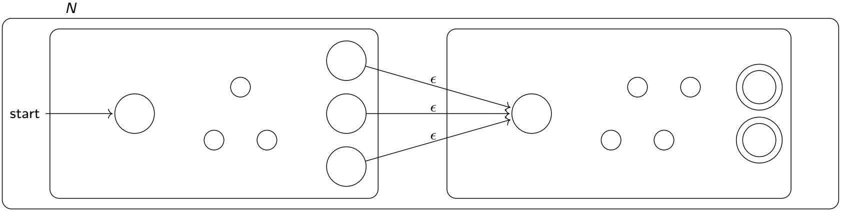
Linguaggi regolari chiusi per concatenazione

Teorema

La classe dei linguaggi regolari è chiusa per l'operazione di concatenazione. Cioè, se L_1 e L_2 sono linguaggi regolari, allora lo è anche $L_1 \circ L_2$.

Idea generale.





Linguaggi regolari chiusi per concatenazione

Teorema

La classe dei linguaggi regolari è chiusa per l'operazione di concatenazione. Cioè, se L_1 e L_2 sono linguaggi regolari, allora lo è anche $L_1 \circ L_2$.

Dimostrazione.
Siano L_1 e L_2 definiti sullo stesso alfabeto Σ (non si perde di generalità).
Sia $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ il DFA che riconosce L_1.
Sia $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ il DFA che riconosce L_2.

Costruiamo l' NFA $N = (Q, \Sigma, \delta, q_1, F_2)$ che riconosce $L_1 \circ L_2$.

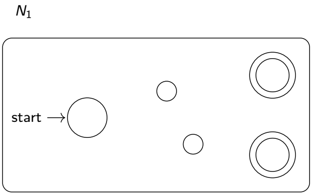
- $Q = Q_1 \cup Q_2$.
- Lo stato iniziale è lo stesso di N_1.
- Gli stati finali sono gli stessi di N_2.
- La funzione di transizione è definita in modo che per ogni $q \in Q$ e ogni $a \in \Sigma_\epsilon$:

$$\delta(q, a) = \begin{cases} \delta_1(q, a) & \text{se } q \in Q_1 \text{ e } q \notin F_1 \\ \delta_1(q, a) & \text{se } q \in F_1 \text{ e } a \neq \epsilon \\ \delta_1(q, a) \cup \{q_2\} & \text{se } q \in F_1 \text{ e } a = \epsilon \\ \delta_2(q, a) & \text{se } q \in Q_2. \end{cases}$$

Teorema

La classe dei linguaggi regolari è chiusa per l'operazione di kleene star. Cioè, se L è un linguaggio regolare, allora lo è anche L^*.

Idea. Esiste un NFA N_1 che riconosce L. Modifichiamo N_1 in modo da ottenere un NFA N che riconosca L^*: N accetta una stringa se essa può essere partizionata in varie parti, ciascuna di esse accettata da N_1. Ricordare che $\epsilon \in L^*$.



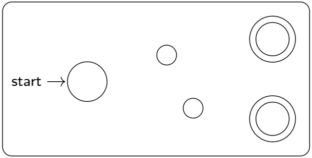
Linguaggi regolari chiusi per operazione star

Teorema

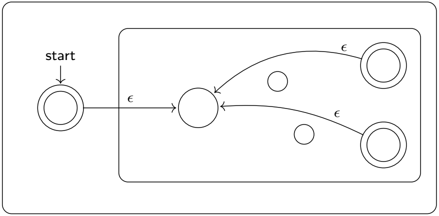
La classe dei linguaggi regolari è chiusa per l'operazione di kleene star. Cioè, se L è un linguaggio regolare, allora lo è anche L^*.

Idea. Esiste un NFA N_1 che riconosce L. Modifichiamo N_1 in modo da ottenere un NFA N che riconosca L^*: N accetta una stringa se essa può essere partizionata in varie parti, ciascuna di esse accettata da N_1. Ricordare che $\epsilon \in L^*$.

N_1



N



Linguaggi regolari chiusi per operazione star

Teorema

La classe dei linguaggi regolari è chiusa per l'operazione di kleene star. Cioè, se L è un linguaggio regolare, allora lo è anche L^*.

Dimostrazione.
Sia L un linguaggio regolare e $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ l'NFA che lo riconosce.

Costruiamo l'NFA $N = (Q, \Sigma, \delta, q_0, F)$ che riconosce L^*.

- $Q = \{q_0\} \cup Q_1$.
- Lo stato iniziale è q_0.
- $F = \{q_0\} \cup F_1$.
- La funzione di transizione è definita in modo che per ogni $q \in Q$ e ogni $a \in \Sigma_\epsilon$:

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & \text{se } q \in Q_1 \text{ e } q \notin F_1 \\ \delta_1(q,a) & \text{se } q \in F_1 \text{ e } a \neq \epsilon \\ \delta_1(q,a) \cup \{q_1\} & \text{se } q \in F_1 \text{ e } a = \epsilon \\ \emptyset & \text{se } q = q_0 \text{ e } a \neq \epsilon \\ \{q_1\} & \text{se } q = q_0 \text{ e } a = \epsilon. \end{cases}$$