



SAPIENZA  
UNIVERSITÀ DI ROMA

“SAPIENZA” UNIVERSITÀ DI ROMA  
INGEGNERIA DELL'INFORMAZIONE,  
INFORMATICA E STATISTICA  
DIPARTIMENTO DI INFORMATICA

---

# Automi: Calcolabilità e Complessità

---

Appunti integrati con il libro "Introduzione alla teoria della computazione",  
Michael Sipser

*Author*  
Alessio Bandiera

9 settembre 2023

# Indice

<b>Informazioni e Contatti</b>	<b>1</b>
<b>1 Linguaggi regolari</b>	<b>2</b>
1.1 Automi finiti . . . . .	2
1.1.1 Linguaggi . . . . .	2
1.1.2 Automi finiti . . . . .	3
1.2 Non determinismo . . . . .	6
1.2.1 Definizioni . . . . .	6
1.3 Operazioni regolari . . . . .	9
1.3.1 Definizioni . . . . .	9

# Informazioni e Contatti

## Prerequisiti consigliati:

- TODO: DA DECIDERE

## Segnalazione errori ed eventuali migliorie:

Per segnalare eventuali errori e/o migliorie possibili, si prega di utilizzare il **sistema di Issues fornito da GitHub** all'interno della pagina della repository stessa contenente questi ed altri appunti (link fornito al di sotto), utilizzando uno dei template già forniti compilando direttamente i campi richiesti.

Gli appunti sono in continuo aggiornamento, pertanto, previa segnalazione, si prega di controllare se l'errore sia ancora presente nella versione più recente.

## Licenza di distribuzione:

These documents are distributed under the [GNU Free Documentation License](#), a form of copyleft intended to be used on manuals, textbooks or other types of document in order to assure everyone the effective freedom to copy and redistribute it, with or without modifications, either commercially or non-commercially.

## Contatti dell'autore e ulteriori link:

- Github: <https://github.com/ph04>
- Email: [alessio.bandiera02@gmail.com](mailto:alessio.bandiera02@gmail.com)
- LinkedIn: [Alessio Bandiera](#)

# 1

## Linguaggi regolari

### 1.1 Automi finiti

#### 1.1.1 Linguaggi

##### Definizione 1.1.1.1: Alfabeto

Si definisce **alfabeto** un qualsiasi insieme finito, non vuoto; i suoi elementi sono detti **simboli**.

**Esempio 1.1.1.1** (Alfabeto).  $\Sigma = \{0, 1, x, y, z\}$  è un alfabeto, composto da 5 simboli.

##### Definizione 1.1.1.2: Stringa

Sia  $\Sigma$  un alfabeto; una **stringa su  $\Sigma$**  è una sequenza finita di simboli di  $\Sigma$ ; la **stringa vuota** appartiene ad ogni alfabeto, ed è denotata con  $\varepsilon$ .

- Data una stringa  $w$  di  $\Sigma$ , allora  $|w|$  è la lunghezza di  $w$ .
- Se  $w$  ha lunghezza  $n \in \mathbb{N}$ , allora è possibile scrivere che  $w = w_1 w_2 \cdots w_n$  con  $w_i \in \Sigma$  e  $i \in [1, n]$ .

**Esempio 1.1.1.2** (Stringa). Sia  $\Sigma = \{0, 1, x, y, z\}$  un alfabeto; allora una sua possibile stringa è  $w = x1y0z$ .

##### Definizione 1.1.1.3: Stringa inversa

Sia  $\Sigma$  un alfabeto, e  $w = w_1 w_2 \cdots w_n$  una sua stringa; allora si definisce l'**inversa** di  $w$  come segue:  $w^{\mathcal{R}} = w_n w_{n-1} \cdots w_1$

**Definizione 1.1.1.4: Concatenazione**

Sia  $\Sigma$  un alfabeto, e  $x = x_1x_2 \cdots x_n, y = y_1y_2 \cdots y_n$  due sue stringhe; allora  $xy$  è la stringa ottenuta attraverso la **concatenazione** di  $x$  ed  $y$ .

Per indicare una stringa concatenata con se stessa  $k$  volte, si utilizza la notazione  $x^k = \underbrace{xx \cdots x}_k$ .

**Definizione 1.1.1.5: Prefisso**

Sia  $\Sigma$  un alfabeto, ed  $x, y$  due sue stringhe; allora  $x$  è detto essere un **prefisso** di  $y$ , se  $\exists z \mid xz = y$ , con  $z$  stringa in  $\Sigma$ .

**Esempio 1.1.1.3** (Prefisso). Sia  $\Sigma = \{a, b, c\}$  un alfabeto; allora la stringa  $x = ab$  è prefisso della stringa  $y = abc$ , poiché esiste una stringa  $z = c$  tale per cui  $xz = y$ .

**Definizione 1.1.1.6: Linguaggio**

Sia  $\Sigma$  un alfabeto; si definisce **linguaggio** un insieme di stringhe di  $\Sigma$ . Un linguaggio è detto **prefisso**, se nessun suo elemento è prefisso di un altro.

**1.1.2 Automi finiti****Definizione 1.1.2.1: Automa finito**

Un **automa finito**, o **macchina**, è una quintupla  $(Q, \Sigma, \delta, q_0, F)$ , dove

- $Q$  è l'**insieme degli stati** dell'automa, un insieme *finito*
- $\Sigma$  è l'**alfabeto** dell'automa, un insieme *finito*
- $\delta : Q \times \Sigma \rightarrow Q$  è la **funzione di transizione**, che definisce la relazione tra gli stati
- $q_0 \in Q$  è lo **stato iniziale**
- $F \subseteq Q$  è l'**insieme degli stati accettanti**, sui quali le stringhe possono terminare

**Esempio 1.1.2.1** (Automa finito). Un esempio di automa finito è il seguente:

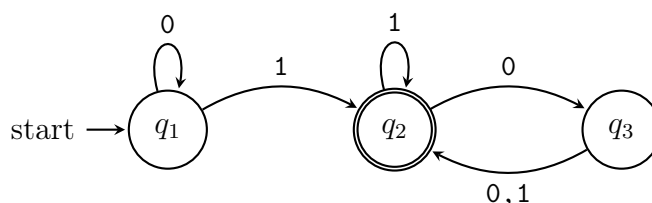


Figura 1.1: Un automa finito.

esso può essere descritto secondo la quintupla  $(Q, \Sigma, \delta, q_0, F)$  come segue:

- $Q = \{q_1, q_2, q_3\}$
- $\Sigma = \{0, 1\}$
- $\delta$  è la seguente:

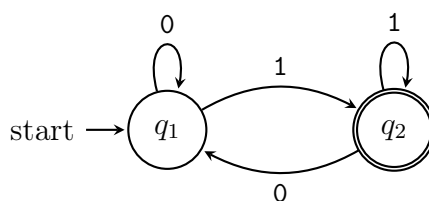
	0	1
$q_1$	$q_1$	$q_2$
$q_2$	$q_3$	$q_2$
$q_3$	$q_2$	$q_2$

- $q_1$  è lo stato iniziale
- $F = \{q_2\} \subseteq Q$

### Definizione 1.1.2.2: Linguaggio di un automa

Sia  $M$  un automa; allora il **linguaggio** di  $M$  è un insieme  $L(M)$  contenente tutte le stringhe accettate da  $M$ ; simmetricamente, si dice che  $M$  **riconosce**  $L(M)$ .

**Esempio 1.1.2.2** (Linguaggio di un automa). Si consideri il seguente automa  $M_1$ :

Figura 1.2: Un automa  $M_1$ .

sapendo che  $\Sigma = \{0, 1\}$ , che  $q_1$  è lo stato iniziale, e che  $F = \{q_2\}$ , ci si convince facilmente che

$$L(M_1) = \{w \mid w = w_1 w_2 \cdots w_{n-1} 1, n \in \mathbb{N}\}$$

ovvero,  $M_1$  accetta tutte e sole le stringhe che terminano per 1.

**Definizione 1.1.2.3: Stringhe accettate**

Sia  $M = (Q, \Sigma, \delta, q_0, F)$  un automa, e sia  $w = w_1 \cdots w_n$  una stringa tale per cui  $\forall i \in [1, n] \quad w_i \in \Sigma$ ; allora,  $M$  **accetta**  $w$  se esiste una sequenza di stati  $r_0, \dots, r_n \in Q$  tali per cui

- $r_0 = q_0$
- $\forall i \in [1, n-1] \quad \delta(r_i, w_{i+1}) = r_{i+1}$
- $r_n \in F$

Dato un linguaggio  $A$ , si ha che  $M$  **riconosce**  $A$  se e solo se  $A = \{w \mid M \text{ accetta } w\}$ .

**Definizione 1.1.2.4: Linguaggio regolare**

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

## 1.2 Non determinismo

### 1.2.1 Definizioni

#### Definizione 1.2.1.1: NFA

Un **NFA** (*Nondeterministic Finite Automaton*) è un automa in cui possono esistere varie scelte per lo stato successivo in ogni punto. Durante la computazione, ogni volta che viene incontrata una scelta, la macchina si *divide*, e ognuno dei vari automi risultanti computa le varie scelte indipendentemente.

Formalmente, un NFA è una quintupla  $(Q, \Sigma, \delta, q_0, F)$ , dove

- $Q$  è l'**insieme degli stati**, un insieme *finito*
- $\Sigma$  è l'**alfabeto** dell'automata, un insieme *finito*
- $\delta : Q \times \Sigma_\epsilon \rightarrow \mathcal{P}(Q)$  è la **funzione di transizione**, che definisce la relazione tra gli stati
- $q_0 \in Q$  è lo **stato iniziale**
- $F \subseteq Q$  è l'**insieme degli stati accettanti**

dove  $\Sigma_\epsilon := \Sigma \cup \{\epsilon\}$ .

Se il simbolo di input successivo non compare su alcuno degli archi uscenti dallo stato occupato da una copia della macchina, quella copia cessa di proseguire; inoltre, se *una qualunque copia* della macchina è in uno stato accettante, l'NFA accetta la stringa di input. Si noti che questa divisione è descritta dall'insieme potenza  $\mathcal{P}(Q)$ , poiché da ogni stato si può arrivare ad un *insieme* di stati.

Gli automi deterministici vengono anche detti **DFA** (*Deterministic Finite Automaton*). Si noti che il determinismo è un caso particolare di non determinismo, dunque un DFA è sempre anche un NFA.

**Esempio 1.2.1.1 (NFA).** Un esempio di NFA è il seguente:

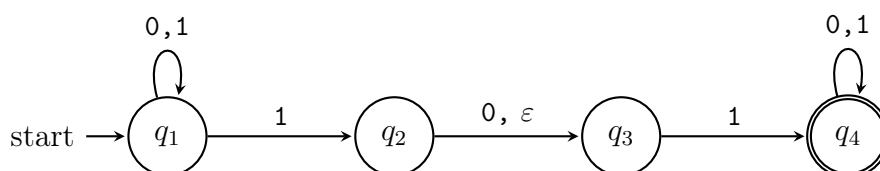


Figura 1.3: Un NFA.

esso può essere descritto secondo la quintupla  $(Q, \Sigma, \delta, q_0, F)$  come segue:

- $Q = \{q_1, q_2, q_3, q_4\}$
- $\Sigma = \{0, 1\}$

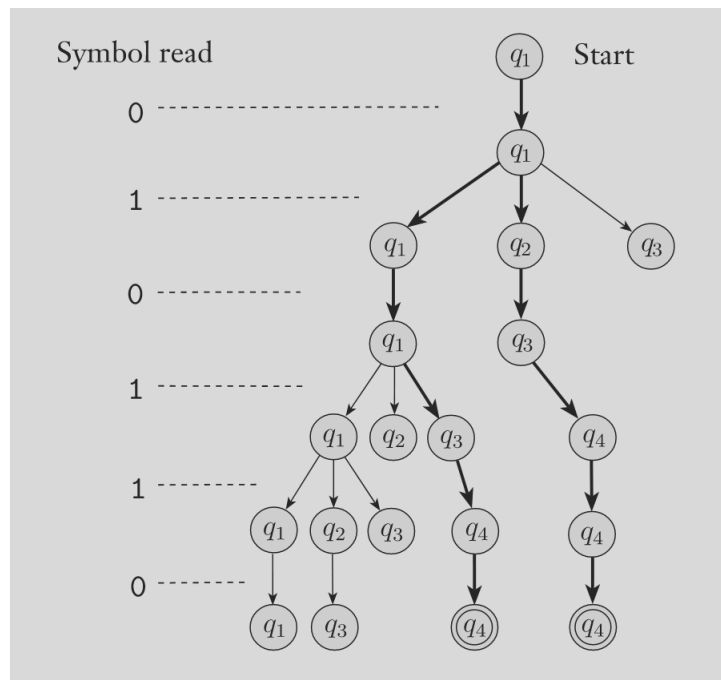


- $\delta$  è la seguente:

	0	1	$\varepsilon$
$q_1$	$\{q_1\}$	$\{q_1, q_2\}$	$\emptyset$
$q_2$	$\{q_3\}$	$\emptyset$	$\{q_3\}$
$q_3$	$\emptyset$	$\{q_4\}$	$\emptyset$
$q_4$	$\{q_4\}$	$\{q_4\}$	$\emptyset$

- $q_1$  è lo stato iniziale
- $F = \{q_4\} \subseteq Q$

L'esecuzione dell'NFA mostrato è la seguente:



Si noti che nel momento in cui vengono incontrati  $\varepsilon$ -archi si giunge allo stato successivo nello stesso step dell'input incontrato, dunque gli  $\varepsilon$ -archi *non* producono uno step di esecuzione, ma vengono eseguiti non appena vengono incontrati, e vengono accorpati immediatamente.

#### Definizione 1.2.1.2: Stringhe accettate

Sia  $N = (Q, \Sigma, \delta, q_0, F)$  un automa, e sia  $w = w_1 \cdots w_n$  una stringa tale per cui  $\forall i \in [1, n] \quad w_i \in \Sigma$ ; allora,  $M$  **accetta**  $w$  se esiste una sequenza di stati  $r_0, \dots, r_n \in Q$  tali per cui

- $r_0 = q_0$
- $\forall i \in [1, n-1] \quad r_{i+1} \in \delta(r_i, w_{i+1})$
- $r_n \in F$

**Definizione 1.2.1.3: Equivalenza tra automi**

Due macchine si dicono **equivalenti** se e solo se riconoscono lo stesso linguaggio.

**Teorema 1.2.1.1: DFA equivalente ad NFA**

Sia  $N$  un NFA; allora, esiste un DFA ad esso equivalente.

*Dimostrazione.* Sia  $N = (Q, \Sigma_\varepsilon, \delta, q_0, F)$  l'NFA in ipotesi, tale da riconoscere un linguaggio  $A$ . Inoltre, si definisca

$$\forall k \geq 0 \quad E(R) := \bigcup_{r \in R} \delta^k(r, \varepsilon)$$

l'insieme degli stati raggiungibili da stati in  $R$ , applicando (anche ripetutamente) un numero arbitrario di  $\varepsilon$ -archi (si noti che per  $k = 0$  si ha che  $R \subseteq E(R)$ ).

Allora, sia  $M = (Q', \Sigma_\varepsilon, \delta', q'_0, F)$  il DFA definito come segue:

- $Q' := \mathcal{P}(Q)$ , scelto tale da rappresentare ogni possibile stato di  $N$ ;
- $\forall R \in Q', a \in \Sigma \quad \delta'(R, a) := \bigcup_{r \in R} E(\delta(r, a))$ , scelta tale in quanto, per un certo insieme di stati  $R \in Q'$  di  $N$ , a  $\delta'(R, a)$  viene assegnata l'unione degli stati che sarebbero stati raggiunti in  $N$  dagli  $r \in R$  con  $a$ , calcolati dunque attraverso  $\delta(r, a)$ , aggiungendo infine i possibili  $\varepsilon$ -archi;
- $q'_0 := E(\{q_0\})$ , scelto tale da far iniziare  $M$  esattamente dove aveva inizio  $N$ , comprendendo anche i possibili  $\varepsilon$ -archi iniziali;
- $F' := \{R \in Q' \mid \exists r \in R : r \in F\}$ , che corrisponde all'insieme degli insiemi di stati di  $N$  contenenti almeno uno stato accettante in  $N$ .

Allora  $M$  è in grado di riconoscere  $A$  per costruzione, poiché il DFA costruito emula l'NFA di partenza, tenendo anche in considerazione gli  $\varepsilon$ -archi. Dunque, sia  $N$  che  $M$  riconoscono  $A$ , e per definizione sono di conseguenza equivalenti.  $\square$

**Corollario 1.2.1.1: Linguaggi regolari con NFA**

Sia  $A$  un linguaggio; allora,  $A$  è regolare se e solo se esiste un NFA che lo riconosce.

*Dimostrazione.*

*Prima implicazione.* Per definizione,  $A$  è regolare, dunque esiste un DFA che lo riconosce. Allora, poiché un DFA è sempre anche NFA, segue la tesi.

*Seconda implicazione.* Sia  $A$  un linguaggio tale da essere riconosciuto da un NFA; allora, per il TODO FIXA STA ROBA NON FUNZIONA ??, esiste un DFA equivalente all'NFA che riconosce  $A$ , e dunque per definizione  $A$  è regolare.

$\square$

## 1.3 Operazioni regolari

### 1.3.1 Definizioni

#### Definizione 1.3.1.1: Unione

Siano  $A$  e  $B$  due linguaggi; allora, si definisce l'**unione** di  $A$  e  $B$  il seguente linguaggio:

$$A \cup B = \{x \mid x \in A \vee x \in B\}$$

**Esempio 1.3.1.1 (Unione).** Sia  $\Sigma = \{a, \dots, z\}$  l'alfabeto composto da 26 lettere, e siano  $A = \{\text{uno}, \text{due}\}$  e  $B = \{\text{tre}, \text{quattro}\}$  due linguaggi su  $\Sigma$ . Allora, si ha che

$$A \cup B = \{\text{uno}, \text{due}, \text{tre}, \text{quattro}\}$$

#### Proposizione 1.3.1.1: Chiusura dell'unione

Siano  $A$  e  $B$  due linguaggi regolari su un alfabeto  $\Sigma$ ; allora  $A \cup B$  è regolare.

*Dimostrazione I.* Per definizione,  $A$  e  $B$  sono linguaggi regolari, dunque esistono due automi finiti

$$\begin{aligned} M_1 &= (Q_1, \Sigma, \delta_1, q_1, F_1) \\ M_2 &= (Q_2, \Sigma, \delta_2, q_2, F_2) \end{aligned}$$

tali da riconoscere rispettivamente  $A$  e  $B$ . Allora, sia  $M = (Q, \Sigma, \delta, q_0, F)$  l'automa finito definito come segue:

- $Q := Q_1 \times Q_2 = \{(r_1, r_2) \mid r_1 \in Q_1 \wedge r_2 \in Q_2\}$ , scelto tale in quanto permette di avere tutte le possibili combinazioni di stati dei due automi di partenza;
- $\forall (r_1, r_2) \in Q, a \in \Sigma \quad \delta((r_1, r_2), a) := (\delta_1(r_1, a), \delta_2(r_2, a))$ , scelta tale in quanto permette di simulare entrambi gli automi di partenza contemporaneamente, mandando ogni stato di  $M_1$  ed  $M_2$  dove sarebbe andato nei rispettivi automi di appartenenza;
- $q_0 := (q_1, q_2)$ , scelto tale in quanto deve essere lo stato in cui entrambe gli automi in ipotesi iniziavano;
- $F := (F_1 \times Q_2) \cup (Q_1 \times F_2) = \{(r_1, r_2) \mid r_1 \in F_1 \vee r_2 \in F_2\}$ , scelto tale in quanto permette di simulare gli stati accettanti di entrambi gli automi, e vanno prese tutte le coppie che vedono almeno uno dei due stati come accettanti poiché altrimenti non si accetterebbero delle stringhe accettate da  $M_1$  ed  $M_2$  in partenza.

Allora, poiché  $M$  è in grado di simulare  $M_1$  ed  $M_2$  contemporaneamente, per costruzione accetterà ogni stringa di  $A$  e di  $B$ , dunque riconoscendo  $A \cup B$ , e di conseguenza  $A \cup B$  è regolare per definizione.  $\square$

*Dimostrazione II.* Siano  $N_1$  ed  $N_2$  i seguenti NFA, tali da riconoscere rispettivamente i due linguaggi  $A$  e  $B$  in ipotesi:

$$N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$$

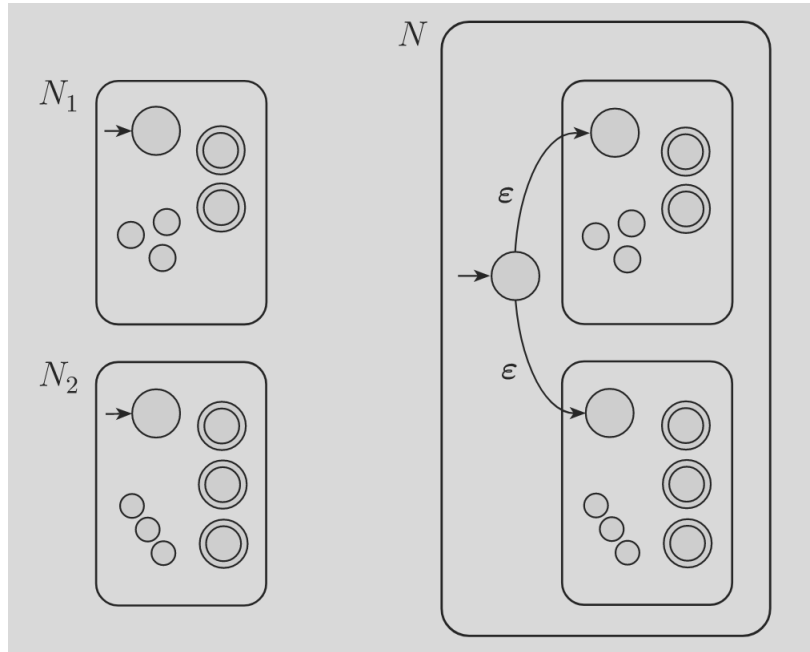
$$N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$$

Allora, sia  $N = (Q, \Sigma, \delta, q_0, F)$  l'NFA costruito come segue:

- $Q = \{q_0\} \cup Q_1 \cup Q_2$ , dove  $q_0$  è un nuovo stato, e  $Q$  è scelto tale da includere sia  $N_1$  che  $N_2$ ;
- $\forall q \in Q, a \in \Sigma_\varepsilon \quad \delta(q, a) = \begin{cases} \delta_1(q, a) & q \in Q_1 \\ \delta_2(q, a) & q \in Q_2 \\ \{q_1, q_2\} & q = q_0 \wedge a = \varepsilon \\ \emptyset & q = q_0 \wedge a \neq \varepsilon \end{cases}$ , scelta tale da poter eseguire contemporaneamente gli NFA  $N_1$  ed  $N_2$ , definendo per casi la funzione di transizione;
- $q_0$  è il nuovo stato, che rappresenta lo stato iniziale di  $N$ ;
- $F = F_1 \cup F_2$ , scelto tale da costruire  $N$  in modo che accetti una stringa se e solo se la accetterebbero  $N_1$  o  $N_2$ .

Allora, l'NFA risultante  $N$  è in grado di computare contemporaneamente  $N_1$  ed  $N_2$ , ed è dunque in grado di riconoscere  $A$  e  $B$  contemporaneamente; di conseguenza,  $N$  riconosce  $A \cup B$ , che risulta dunque essere regolare per definizione.

La foto seguente raffigura la dimostrazione costruttiva appena descritta:



□

**Definizione 1.3.1.2: Concatenazione**

Siano  $A$  e  $B$  due linguaggi; allora, si definisce la **concatenazione** di  $A$  e  $B$  il seguente linguaggio:

$$A \circ B = \{xy \mid x \in A \wedge y \in B\}$$

**Esempio 1.3.1.2** (Concatenazione). Sia  $\Sigma = \{\mathbf{a}, \dots, \mathbf{z}\}$  l'alfabeto composto da 26 lettere, e siano  $A = \{\mathbf{uno}, \mathbf{due}\}$  e  $B = \{\mathbf{tre}, \mathbf{quattro}\}$  due linguaggi su  $\Sigma$ . Allora, si ha che

$$A \circ B = \{\mathbf{unotre}, \mathbf{unoquattro}, \mathbf{duetre}, \mathbf{duequattro}\}$$

**Proposizione 1.3.1.2: Chiusura della concatenazione**

Siano  $A$  e  $B$  due linguaggi regolari su un alfabeto  $\Sigma$ ; allora  $A \circ B$  è regolare.

*Dimostrazione.* TODO

□

**Definizione 1.3.1.3: Star**

Sia  $A$  un linguaggio; allora, si definisce l'operazione unaria **star** che definisce il seguente linguaggio:

$$A^* = \{x_1x_2 \cdots x_k \mid k \geq 0 \wedge \forall i \in [1, k] \quad x_i \in A\}$$

Si noti che  $k = 0 \implies \varepsilon \in A^*$  per ogni linguaggio  $A$ .

**Esempio 1.3.1.3** (Star). Sia  $\Sigma = \{\mathbf{a}, \dots, \mathbf{z}\}$  l'alfabeto composto da 26 lettere, e siano  $A = \{\mathbf{uno}, \mathbf{due}\}$  e  $B = \{\mathbf{tre}, \mathbf{quattro}\}$  due linguaggi su  $\Sigma$ . Allora, si ha che

$$A^* = \{\varepsilon, \mathbf{uno}, \mathbf{due}, \mathbf{unouno}, \mathbf{unodue}, \mathbf{dueuno}, \mathbf{duedue}, \dots\}$$

**Proposizione 1.3.1.3: Chiusura dell'operazione star**

Siano  $A$  e  $B$  due linguaggi regolari su un alfabeto  $\Sigma$ ; allora  $A^*$  è regolare.

*Dimostrazione.* Per definizione,  $A$  e  $B$  sono linguaggi regolari, dunque per l'?? esistono due NFA

$$N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$$

$$N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$$

tali da riconoscere rispettivamente  $A$  e  $B$ . Allora, sia  $N = (Q, \Sigma, \delta, q_0, F)$  l'NFA costruito come segue:

- TODO

□