

# Automi, Calcolabilità e Complessità

Alessio Marini, 2122855

Appunti presi durante il corso di **Automi, Calcolabilità e Complessità** nell'anno **2025/2026** del professore Daniele Venturi.

Gli appunti li scrivo principalmente per rendere il corso più comprensibile **a me** e anche per imparare il linguaggio Typst. Se li usate per studiare verificate sempre le informazioni 🙏.

## Contatti:

🐙 [alem1105](#)

✉ [marini.2122855@studenti.uniroma1.it](mailto:marini.2122855@studenti.uniroma1.it)

September 27, 2025

# Indice

1. Introduzione alla terminologia .....	3
1.1. Operazioni sulle Stringhe .....	3
2. DFA - Automa a Stati Finiti .....	5
3. Linguaggi Regolari .....	8
3.1. Operazioni sui Linguaggi .....	10
3.2. Proprietà di chiusura dei Linguaggi Naturali .....	11
3.2.1. Chiusura per Unione .....	11
3.2.2. Chiusura per Complemento .....	13
4. Non Determinismo .....	14

# 1. Introduzione alla terminologia

Introduciamo delle definizioni e delle operazioni che utilizzeremo durante il corso.

## Alfabeto

È un insieme finito di simboli, quindi ad esempio  $\Sigma = \{0, 1, x, y, z\}$ .

## Stringa

Una stringa è una sequenza di simboli che appartengono ad un alfabeto. Quindi, ad esempio, dato l'alfabeto  $\Sigma = \{0, 1, x, y, z\}$  una sua stringa è  $w = 01z$ .

## 1.1. Operazioni sulle Stringhe

### Lunghezza di una Stringa

Data una stringa  $w \in \Sigma^*$  indichiamo la lunghezza con  $|w|$  ed è definita come il numero di simboli che contiene.

### Concatenazione

Data la stringa  $x = x_1, \dots, x_n \in \Sigma^*$  e la stringa  $y = y_1, \dots, y_m \in \Sigma^*$  definiamo come **concatenazione di  $x$  con  $y$**  la stringa  $x \cdot y = x_1 \dots x_n y_1 \dots y_m$ .

### Stringa Vuota

Durante il corso indicheremo con  $\varepsilon$  la stringa vuota, ovvero una stringa tale che  $|\varepsilon| = 0$ .  
Se concateniamo una qualsiasi stringa non vuota con una stringa vuota otteniamo la prima stringa:

$$\forall w \in \Sigma^* \quad w \cdot \varepsilon = w$$

### Conteggio

Data una stringa  $w \in \Sigma^*$  e un simbolo  $a \in \Sigma$  indichiamo il conteggio di  $a$  in  $w$  con  $|w|_a$  e lo definiamo come il numero di occorrenze del carattere  $a$  nella stringa  $w$ .

### Stringa Rovesciata

Data una stringa  $w = a_1 \dots a_n \in \Sigma^*$  dove  $a_1, \dots, a_n \in \Sigma$ , definiamo la stringa rovesciata con  $w^R = a_n \dots a_1$ .

### Potenza

Data la stringa  $w \in \Sigma^*$  e dato  $n \in \mathbb{N}$  definiamo la potenza in modo ricorsivo:

$$w^n = \begin{cases} \varepsilon & \text{se } n = 0 \\ ww^{\{n-1\}} & \text{se } n > 0 \end{cases}$$

### Linguaggio

Dato un alfabeto  $\Sigma$  definiamo  $\Sigma^*$  come linguaggio di  $\Sigma$ , ovvero l'insieme di tutte le stringhe di quell'alfabeto.

## 2. DFA - Automa a Stati Finiti

Il modello di computazione che utilizzeremo per ora è un DFA, questo ha una memoria limitata e permette una gestione dell'input. La memoria gli permette di memorizzare i suoi stati e tramite gli input decide in quale stato futuro muoversi.

### Esempio - Una porta automatica

Una porta automatica avrà due stati:

- Aperta
- Chiusa

E due input:

- Rileva qualcuno
- Non rileva nessuno

Quindi lo stato iniziale sarà la porta chiusa, se rileva qualcuno va nello stato di aperta mentre se non rileva nessuno rimane chiusa.

### DFA

Definiamo un DFA come una tupla  $(Q, \Sigma, \delta, q_0, F)$  dove:

- $Q$  è l'insieme degli stati
- $\Sigma$  è l'insieme finito dei simboli in input
- $\delta : Q \times \Sigma \rightarrow Q$  è la funzione di transizione degli stati, ovvero dato lo stato in cui si trova ed un input, restituisce lo stato in cui andremo
- $q_0 \in Q$  è lo stato iniziale dell'automa
- $F \subseteq Q$  è l'insieme degli stati di accettazione dell'automa, ovvero gli stati dove l'automa si trova dopo aver riconosciuto determinate stringhe e consente la terminazione.

Dato DFA  $M$  possiamo definire l'insieme delle stringhe riconosciute dall'automa, ovvero quelle che lo portano in uno stato di accettazione come  $L(M)$ . Da notare che può anche accadere che  $L(M) = \emptyset$ . Daremo una definizione più formale di quest'ultimo più avanti.

Dati dei DFA vogliamo iniziare a definire dei linguaggi dedicati a questi, per farlo abbiamo bisogno della **funzione di transizione estesa**.

### Funzione di Transizione Estesa

La definiamo come:

$$\delta^* = Q \times \Sigma^* \rightarrow Q$$

Quindi questa a differenza di quella classica non usa degli input singoli ma delle intere stringhe appartenenti al **linguaggio** del DFA.

È definibile in modo ricorsivo:

$$\begin{cases} \delta^*(q, \varepsilon) = \delta(q, \varepsilon) = q \\ \delta^*(q, aw) = \delta^*(\delta(q, a), w) \quad \text{con } w \in \Sigma^* \text{ e } a \in \Sigma \end{cases}$$

Quindi data una stringa, partiamo dal primo carattere a sinistra e andiamo avanti utilizzando la funzione di transizione fino ad arrivare ad una stringa vuota.

Adesso diamo le definizioni di **Configurazione** e **Passo di Computazione** che ci serviranno a definire più formalmente un **Linguaggio Accettato** del DFA.

### Configurazione

Sia  $D := (Q, \Sigma, \delta, q_0, F)$  un DFA, definiamo la coppia  $(q, w) \in Q \times \Sigma^*$  come configurazione di  $D$ . Inoltre dato un  $x \in \Sigma^*$ , la **configurazione iniziale** è  $(q_0, x)$ .

### Passo di Configurazione

Indica il passaggio da una configurazione ad un'altra rispettando la funzione di transizione  $\delta$ , il passaggio lo indichiamo con il simbolo  $\vdash_M$  dove  $M$  indica il DFA. Possiamo dire quindi che esiste una relazione binaria fra un passo di configurazione e la funzione di transizione:

$$(p, ax) \vdash_M (q, x) \Leftrightarrow \delta(p, a) = q$$

Dove  $p, q \in Q$  -  $a \in \Sigma$  e  $x \in \Sigma^*$ . Un passaggio di configurazione può avvenire, quindi, soltanto se la funzione di transizione lo permette.

Possiamo estendere questa relazione con il simbolo  $\vdash_M^*$  considerando anche la **chiusura riflessiva e transitiva**:

- **Riflessività:**  $(q, x) \vdash_M^* (q, x)$
- **Transitività:** Se  $(q, aby) \vdash_M (p, by) \wedge (p, by) \vdash_M (r, y) \Rightarrow (q, aby) \vdash_M^* (r, y)$ 
  - Dove  $q, p, r \in Q$  -  $a, b \in \Sigma$  ed  $y \in \Sigma^*$

Definiamo quindi il linguaggio accettato dal DFA.

### Linguaggio Accettato

Diciamo che  $x \in \Sigma^*$  è accettato da un automa  $M = (Q, \Sigma, \delta, q_0, F)$  se  $\delta^*(q_0, x) \in F$  oppure usando la relazione del passaggio, se  $(q_0, x) \vdash_M^* (q, \varepsilon)$  con  $q \in F$ .

### 3. Linguaggi Regolari

#### Definizione

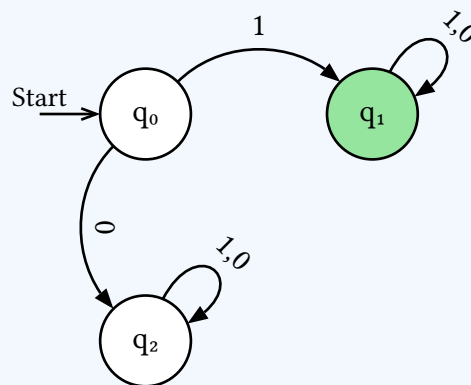
$$\text{REG} = \{L \subseteq \Sigma^* : \exists \text{ DFA } M \text{ t.c. } L(M) = L\}$$

Quindi i linguaggi regolari sono tutti quei linguaggi che sono accettati da almeno un DFA.

Uno dei nostri obiettivi nel corso è quello di, dato un linguaggio, progettare dei DFA adatti.

#### Esempio

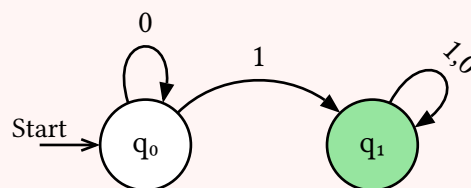
Dato il linguaggio  $L = \{x \in \{0,1\}^* \text{ t.c. } x = 1y, y \in \{0,1\}^*\}$ , un possibile DFA potrebbe essere:



Questo DFA accetta quindi tutte le stringhe che iniziano con il simbolo 1 mentre rifiuta tutte quelle che iniziano con il simbolo 0.

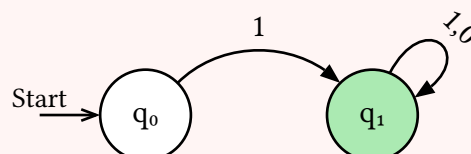
#### Attenzione - Stato Pozzo

Notiamo che è presente lo stato  $q_2$  dal quale il DFA non esce più una volta entrato, questo è necessario perchè se omissso il DFA accetterebbe tutte le stringhe:



Infatti in questo modo se in  $q_0$  riceve 0 rimane su stesso ma poi continua ad attendere input.

Il modo corretto per lasciare lo stesso significato del primo DFA ma omettere lo stato  $q_2$  è quello di omettere anche il comportamento di  $q_0$  in caso riceviamo 0, ovvero:





Adesso dobbiamo dimostrare formalmente che questo DFA accetta il linguaggio fornito, dobbiamo quindi dimostrare che:

$$\text{DFA accetta } x \Leftrightarrow x \in L$$

Innanzitutto facciamo due osservazioni, ovvero che se il DFA si trova in  $q_1$  o  $q_2$  allora non cambierà mai più stato:

- $\delta^*(q_1, u) = q_1 \quad \forall u \in \{0, 1\}^*$
- $\delta^*(q_2, u) = q_2 \quad \forall u \in \{0, 1\}^*$

Dimostriamo per induzione che il linguaggio è accettato, quindi presa una stringa dobbiamo far vedere che se inizia con 1 terminiamo in  $q_1$  altrimenti in  $q_2$ .

### Dimostrazione

#### Caso Base

Come caso base prendiamo una stringa vuota, quindi  $|x| = 0$  ovvero  $x = \varepsilon$ , abbiamo che:

$$\delta^*(q_0, \varepsilon) = \delta(q_0, \varepsilon) = q_0 \notin F$$

Infatti se abbiamo una stringa vuota il DFA non fa nulla e rimane in  $q_0$

#### Passo Induttivo

Adesso dobbiamo prendere una stringa  $w$  tale che  $|w| \leq n$  con  $n > 0$ , la funzione di transizione avrà quindi 3 risultati possibili:

$$\delta^*(q_0, w) = \begin{cases} q_0 & \text{se } w = \varepsilon \\ q_1 & \text{se } w \text{ inizia con } 1 \\ q_2 & \text{se } w \text{ inizia con } 0 \end{cases}$$

Prendiamo quindi una stringa  $x$  tale che  $|x| = n + 1$  e la costruiamo come  $x = au$  con  $a \in \{0, 1\}$  e  $u \in \{0, 1\}^*$ , la funzione di transizione ci restituirà:

$$\delta^*(q_0, x) = \delta^*(q_0, au) = \delta^* \left( \underbrace{\delta(q_0, a)}_{\text{ha 2 soluzioni}}, u \right)$$

Le due soluzioni del passaggio evidenziato sono:

- $\delta(q_0, a) = q_1$  se  $a = 1$
- $\delta(q_0, a) = q_2$  se  $a = 0$

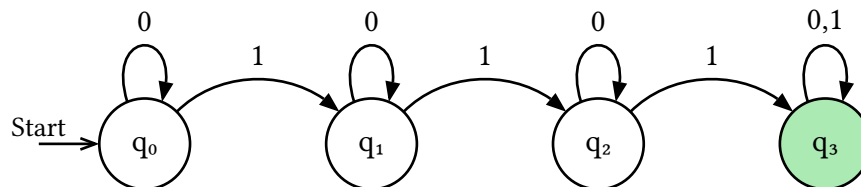
Quindi il DFA andrà sicuramente in uno dei due stati  $q_1$  o  $q_2$  e da lì non si muoverà più, per il ragionamento fatto all'inizio della dimostrazione.

## Esercizi

### DFA 1

Dato il linguaggio  $L = \{x : x \in \{0, 1\}^* \wedge W_H(x) \geq 3\}$  con  $W_H(x) = \#1$  ovvero il numero di 1 presenti nella stringa. Progettare un automa che accetta il linguaggio e dimostrarlo.

Un possibile automa potrebbe essere:

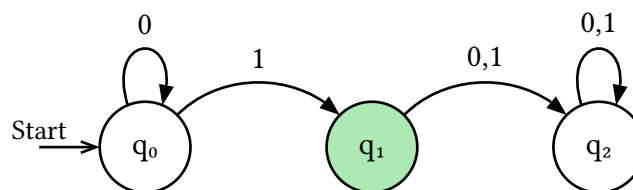


**Dimostrazione:** Copiare da iPad

### DFA 2

Dato il linguaggio  $L = \{x : x = 0^n 1 \text{ con } n \in \mathbb{N}\}$  progettare un automa che accetta il linguaggio e dimostrarlo.

Un possibile automa potrebbe essere:



**Dimostrazione:** Copiare da iPad

## 3.1. Operazioni sui Linguaggi

Definiamo adesso delle operazioni sui linguaggi che ci torneranno utili.

### Unione

$$L_1 \cup L_2 = \{x \in \Sigma^* : x \in L_1 \vee x \in L_2\}$$

### Intersezione

$$L_1 \cap L_2 = \{x \in \Sigma^* : x \in L_1 \wedge x \in L_2\}$$

### Complemento

$$\overline{L} = \{x \in \Sigma^* : x \notin L\}$$

### Concatenazione

$$L_1 \circ L_2 = \{xy : x \in L_1 \wedge y \in L_2\}$$

Da notare che questa operazione non è commutativa quindi  $L_1 \circ L_2 \neq L_2 \circ L_1$

### Potenza

Possiamo definirla ricorsivamente:

$$\begin{cases} L_0 = \varepsilon \\ L^{n+1} = L^n \circ L \end{cases}$$

### Operatore \* «star»

$$L^* = \bigcup_{n \geq 0} L^n = \{\varepsilon\} \cup L^1 \cup L^2 \cup \dots$$

## 3.2. Proprietà di chiusura dei Linguaggi Naturali

Vogliamo capire se dati due linguaggi naturali  $L_1, L_2 \in \text{REG}$  il linguaggio risultante di operazioni effettuate con questi linguaggi è naturale o no, ad esempio se  $L_1 \cup L_2 \in \text{REG}$  oppure se  $L_1 \cap L_2 \in \text{REG}$

### 3.2.1. Chiusura per Unione

#### Teorema - Chiusura per Unione

Come prima idea possiamo dire che:

$$L_1, L_2 \in \text{REG} \Rightarrow \exists M_1, M_2 \in \text{DFA t.c. } L(M_1) = L_1 \wedge L(M_2) = L_2$$

Quindi dati due linguaggi naturali esistono due automi che li hanno come linguaggi accettati. Noi dobbiamo definire un terzo automa  $M$  tale che  $L(M) = L_1 \cup L_2$ , ma data una stringa  $x$  candidata non possiamo provare a vedere prima cosa succede su  $M_1$  e se non la accetta provare  $M_2$  perchè perderemmo la sequenza corretta della stringa su  $M$ .

Quello che dobbiamo fare è testare ogni carattere di  $x$  in parallelo su  $M_1$  e  $M_2$  e in base al risultato aggiorniamo lo stato di  $M$ .

### Input Dimostrazione

Vogliamo mostrare che dati

- $M_1 = (Q_1, \Sigma, \delta_1, q_0^1, F_1)$
- $M_2 = (Q_2, \Sigma, \delta_2, q_0^2, F_2)$

Assumiamo lo stesso  $\Sigma$  per semplicità

Tali che:  $L(M_1) = L_1 \wedge L(M_2) = L_2$

Costruiamo un terzo DFA  $M(Q, \Sigma, \delta, q_0, F)$  t.c.  $L(M) = L_1 \cup L_2$

Avremo che:

- $Q = \{(r_1, r_2) : r_1 \in Q_1, r_2 \in Q_2\} = Q_1 \times Q_2$  (Tutte le coppie di stati possibili)
- $\delta : Q \times \Sigma \rightarrow Q$ 
  - $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$
- $F = \{(r_1, r_2) : r_1 \in F_1 \vee r_2 \in F_2\} = \underbrace{(F_1 \times Q_2)}_{\text{il primo stato è accettato}} \cup \underbrace{(F_2 \times Q_1)}_{\text{il secondo stato è accettato}}$

Infatti basta che soltanto uno dei due stati della coppia venga accettato per accettare la coppia.

Da notare che per l'intersezione abbiamo una situazione molto simile, infatti avremo che:

$$F = \{(r_1, r_2) : r_1 \in F_1 \wedge r_2 \in F_2\} = F_1 \times F_2$$

### Dimostrazione

Vogliamo mostrare che dato

$$\delta^*(q_0, x) = \delta^*((q_0^1, q_0^2), x)$$

Si ha che  $\forall x \in \Sigma^*$

$$= (\delta_1^*(q_0^1, x), \delta_2^*(q_0^2, x))$$

**TODO: MANCA UNA PARTE DI DIMOSTRAZIONE (Mostrare che funziona per n e 0)**

$$\forall x \in \Sigma^*, x \in L(M) \Leftrightarrow x \in L_1 \cup L_2$$

$$\Rightarrow x \in L(M) \Rightarrow \delta^*(q_0, x) \in F = F_1 \times Q_2 \cup F_2 \times Q_1 = (p, q)$$

$$\text{Dove } p = \delta_1^*(q_0^1, x), q = \delta_2^*(q_0^2, x))$$

Questo significa che

- Se  $x \in L_1$  allora  $\delta^*(q_0^1, x) \in F_1$  e quindi:

$$\delta^*(q_0, x) = (\delta_1^*(q_0^1, x), \delta_2^*(q_0^2, x)) \in F_1 \times Q_2 \Rightarrow M \text{ accetta } x$$

- Se  $x \in L_2$  allora  $\delta^*(q_0^2, x) \in F_2$  e quindi:

$$\delta^*(q_0, x) = (\delta_1^*(q_0^1, x), \delta_2^*(q_0^2, x)) \in F_2 \times Q_1 \Rightarrow M \text{ accetta } x$$

Spiegato a parole, abbiamo che la funzione di transizione dell'automa  $M$  equivale ad eseguire lo stesso input sui due automi  $M_1, M_2$ . Presa una stringa del linguaggio questa è accettata dall'automa se e solo se appartiene all'unione dei due linguaggi di  $M_1$  e  $M_2$ .

Partendo dalla sinistra dell'implicazione abbiamo che  $x \in L(M)$  quindi la stringa è accettata e allora la funzione di transizione estesa ci porta in uno stato appartenente ad  $F$ . Ricordiamo che lo stato in cui ci troviamo è in realtà una coppia di stati uno dei quali deve essere accettato o da  $M_1$  o da  $M_2$  e questo appunto significa rispettivamente che o  $x \in L(M_1)$  oppure  $x \in L(M_2)$ .

### 3.2.2. Chiusura per Complemento

Per dimostrare la chiusura per complemento abbiamo bisogno del concetto di **Non Determinismo**.

## 4. Non Determinismo

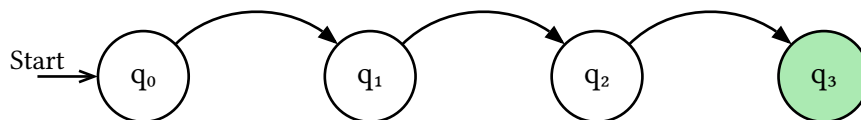
Per adesso abbiamo visto soltanto automi deterministici, questo significa che trovandoci in uno stato e ricevendo un input possiamo soltanto andare in un altro stato o rimanere fermi, ma in generale un solo movimento.

Nel **non determinismo** invece:

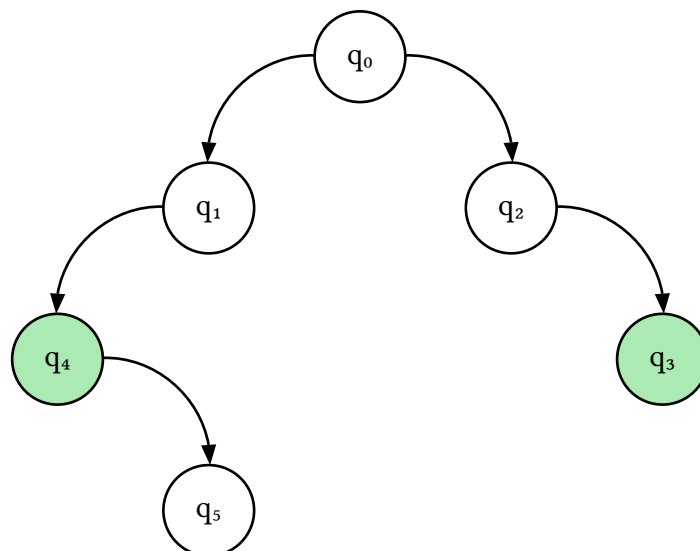
- Quando l'automa è in  $q \in Q$  e legge  $a \in \Sigma$  può andare in diversi stati
- Sono ammessi gli « $\epsilon$ -archi» ovvero l'automa può muoversi senza leggere input. Dallo stesso stato possono partire più « $\epsilon$ -archi».
- Accettazione: Se e solo se esiste un ramo che accetta, vedremo più avanti che quando studiamo un NFA avremo un albero con vari rami, se un ramo accetta allora consideriamo la stringa come accettata per il NFA.

Nel non determinismo quindi abbiamo un input che si dirama in vari stati invece che seguire un cammino di *uno stato alla volta*.

### • Determinismo



### • Non Determinismo



### Definizione - NFA

Un NFA è  $(Q, \Sigma, \delta, q_0, F)$  dove  $Q, \Sigma, q_0, F$  sono come nei DFA ma:

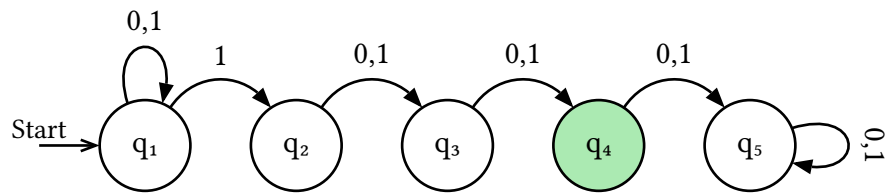
$$\delta : Q \times \Sigma_{\epsilon} \rightarrow \mathbb{P}(Q)$$

Dove  $\Sigma^{\epsilon} \cup \{\epsilon\}$

e  $\mathbb{P}$  è l'insieme delle parti.

Vediamo un esempio e capiamo come ci si muove al loro interno.

### Esempio



Che ha come linguaggio:

$$L = \{x : x \in \{0,1\}^* \text{ che hanno un '1' in terzultima posizione}\}$$

*Da notare che lo stato  $q_5$  possiamo anche ometterlo, ma non dobbiamo indicare in  $q_4$  nessun arco.*

Per muoverci, ad esempio nel NFA dell'esempio sopra, ci torna utile disegnare un albero con tutte i cammini che stiamo intraprendendo. Se ad esempio riceviamo in input la stringa «10110»:

