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

September 27, 2025

Indice

1. Introduzione alla terminologia

Alfabeto

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

Stringa

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

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

Stringa Vuota

Durante il corso indicheremo con ε 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:

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

Conteggio

Data una stringa $w \in \Sigma^*$ e un simbolo $a \in \Sigma$ indichiamo il contaggio 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 $w = a_1 ... a_n \in \Sigma^*$ dove $a_1, ..., a_n \in \Sigma$, definiamo la stringa rovesciata con $w^R = a_n ... a_1$..

Potenza

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

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

Linguaggio

Dato un alfabeto Σ definiamo Σ^* come linguaggio di Σ, 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
- Σ è l'insieme finito dei simboli in input
- $\delta : Q \times \Sigma \to Q$ 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:

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

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:

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

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 δ, 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

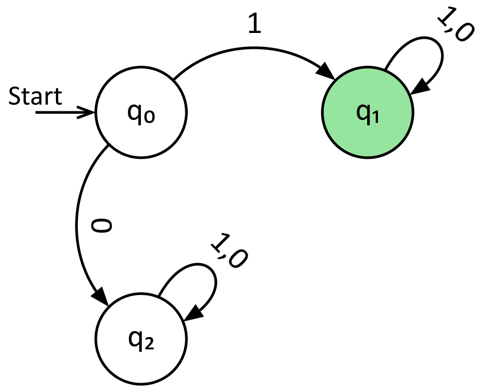
$$\text{REG} = \{L \subseteq \Sigma^* : \exists \text{ DFA M t.c. } L(M) = L\}$$

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

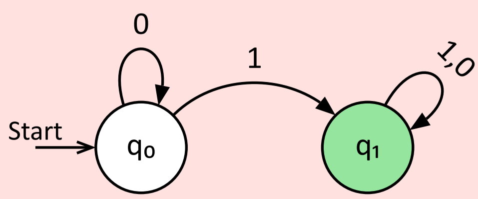
Dato il linguaggio $L = \{x \in \{0,1\}^* \text{ t.c. } x = 1y, y \in \{0,1\}^*\}$, un possibile DFA potrebbe essere:



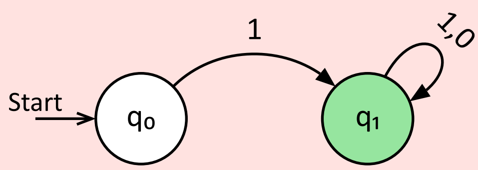
Questo DFA accetta quindi tutte le stringhe che iniziano con il simbolo 1 mentre rifiuta tutte quelle che iniziano con il simbolo 0.

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 omesso 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:

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

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 soluzione 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 sicuremante in uno dei due stati q_1 o q_2 e da lí non si muoverà più, per il ragionamento fatto all'inizio della dimostrazione.

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»

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

3.2. Proprietà di chiusura dei Linguaggi Naturali

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

Teorema - Chiusura per Unione

Come prima idea possiamo dire che:

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

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.