Sapienza Università di Roma

Facoltà di Ingegneria dell'Informazione, Informatica e Statistica
Dipartimento di Informatica

Automi Calcolabilità e Complessità

Autore:
Simone Lidonnici

27 settembre 2024

Indice

1
Linguaggi regolari

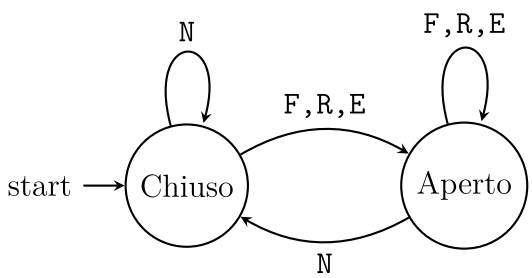
Definizione di linguaggio

Dato un **alfabeto** Σ, cioè un insieme di elementi, un **linguaggio** Σ^* è l'insieme di tutte le strighe ottenibili usando l'alfabeto Σ.

1.1 Automi Deterministici a Stati Finiti (DFA)

Il modello usato per definire i **linguaggi regolari** è l'**automa a stati finiti**, cioè una macchina che permette tramite l'input di passare da uno stato ad un altro, che ha memoria limitata e gestione dell'input limitata, ma è molto semplice.
Esempio:
Una porta che si apre tramite dei sensori può essere descritta tramite un automa con due stati (Aperta e Chiusa) e quattro input dati dai sensori (`N, F, R, E`):

- `N`: se non ci sono persone da nessun lato della porta
- `F`: se c'è una persona davanti alla porta
- `R`: se c'è una persona dietro la porta
- `E`: se ci sono persone da entrambi i lati della porta

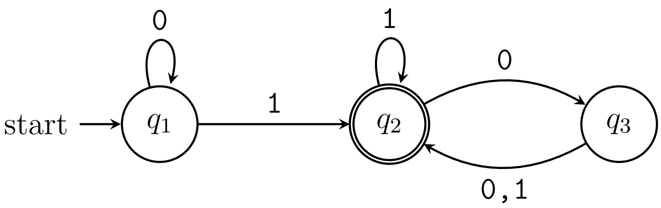


Automa Deterministico a Stati Finiti (DFA)

Un **DFA (Deterministic Finite Automaton)** è una tupla $(Q, \Sigma, \delta, q_o, F)$ in cui:

- Q è l'insieme degli stati dell'automa
- Σ è l'alfabeto dell'automa
- $\delta : Q \times \Sigma \to Q$ è la funzione di transizione degli stati
- $q_0 \in Q$ è lo stato iniziale dell'automa
- $F \subseteq Q$ è l'insieme di **stati accettanti** dell'automa

Esempio:
Preso il seguente DFA:



In questo caso:

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

δ	q_1	q_2	q_3
0	q_1	q_3	q_2
1	q_2	q_2	q_2

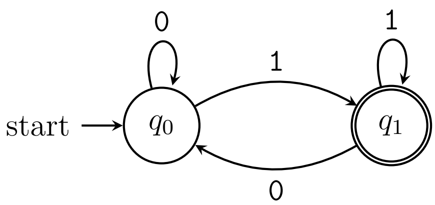
- q_1 è lo stato iniziale dell'automa
- $F = \{q_2\}$ è l'insieme degli stati accettanti

Funzione di transizione estesa

Dato un DFA D, definiamo una **funzione di transizione estesa** $\delta^* : Q \times \Sigma^* \to Q$ in modo ricorsivo:

$$\begin{cases} \delta^*(q, \varepsilon) = \delta(q, \varepsilon) = q \\ \delta^*(q, ax) = \delta^*(\delta(q, a), x) \quad a \in \Sigma, x \in \Sigma^* \end{cases}$$

Esempio:
Preso il seguente DFA:



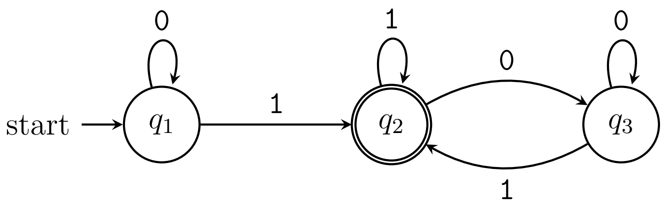
In questo DFA:

- $\delta^*(q_0, 011) = \delta^*(\delta(q_0, 0), 11) = \delta^*(q_0, 11) = \delta^*(\delta(q_0, 1), 1) = \delta^*(q_1, 1) = \delta^*(\delta(q_1, 1), \varepsilon) = \delta^*(q_1, \varepsilon) = q_1$

Linguaggio di un automa

Il **linguaggio di un DFA** D è l'insieme delle stringhe in input che l'automa accetta, cioè quelle per cui l'automa termina in uno stato accettante $q_0 \in F$.
Ogni automa ha un solo linguaggio che si scrive $L(D) = \{x \in \Sigma^* | D \text{ accetta } x\}$. Una stringa $x \in \Sigma^*$ sarà accettata da una DFA se $\delta^*(q_0, x) = q \in F$

Esempio:
Preso il seguente DFA:



Il linguaggio di questa DFA è l'insieme di tutte le stringhe che finiscono con 1:

$$L(D) = \{x \in \{0, 1\}^* | x = y1 \wedge y \in \{0, 1\}^*\}$$

1.1.1 Configurazione di un DFA

Configurazione di un DFA

Dato un DFA D, una **configurazione** di D è una coppia $Q \times \Sigma^*$ che indica:

- lo stato attuale dell'automa
- l'input ancora da leggere

La configurazione iniziale è sempre (q_0, x).

Passo di computazione di un DFA

Un **passo di computazione** è una relazione binaria con simbolo $\vdash_D$ per cui:

$$(q_1, ax) \vdash_D (q_2, x) \iff \delta(q_1, a) = q_2$$

La **chiusura per riflessione e transitività** di $\vdash_D$, scritta come $\vdash_D^*$, ha delle proprietà:

1. $(q_1, ax) \vdash_D (q_2, x) \implies (q_1, ax) \vdash_D^* (q_2, x)$
2. $\forall q, x \ (q, x) \vdash_D^* (q, x)$
3. $(q_1, abc) \vdash_D (q_2, bc) \vdash_D (q_3, c) \implies (q_1, abc) \vdash_D^* (q_3, c)$

Una stringa $x \in \Sigma^*$ sarà accettata da un DFA se:

$$\exists q \in F | (q_0, x) \vdash_D^* (q, \varepsilon)$$

1.2 Automi Non Deterministici a Stati Finiti (NFA)

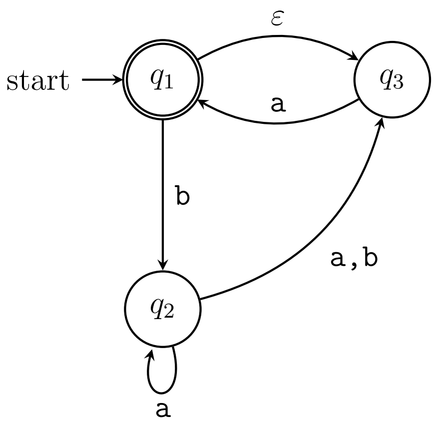
Automa Non Deterministico a Stati Finiti (NFA)

Un **NFA (Non-deterministic Finite Automaton)** è una tupla $(Q, \Sigma, \delta, q_o, F)$ in cui:

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

A differenza del DFA la funzione δ ha come uno dei parametri $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$ e come ritorno un insieme di stati, contenuto nell'insieme delle parti di Q, cioè $\mathcal{P}(Q)$. Inoltre per considerare una stringa accettata da un NFA basta che in uno dei rami della computazione la stringa venga accettata.

Esempio:
Preso il seguente NFA:



In questo caso:

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

δ	q_1	q_2	q_3
a	$\emptyset$	$\{q_2, q_3\}$	q_1
b	q_2	q_3	$\emptyset$
ε	q_3	$\emptyset$	$\emptyset$

- q_1 è lo stato iniziale dell'automa
- $F = \{q_1\}$ è l'insieme degli stati accettanti

1.2.1 Configurazione di un NFA

Configurazione di un NFA

Dato un NFA N, una **configurazione** di N è una coppia $Q \times \Sigma_\varepsilon^*$ che indica:

- lo stato attuale dell'automa
- l'input ancora da leggere

La configurazione iniziale è sempre (q_0, x).

Passo di computazione di un NFA

Un **passo di computazione** è una relazione binaria con simbolo $\vdash_N$ per cui:

$$(q_1, ax) \vdash_D (q_2, x) \iff q_2 \in \delta(q_1, a)$$

La **chiusura per simmetria e transitività** di $\vdash_D$, scritta come $\vdash_N^*$, ha delle proprietà:

1. $(q_1, ax) \vdash_N (q_2, x) \implies (q_1, ax) \vdash_N^* (q_2, x)$
2. $(q_1, abc) \vdash_N (q_2, bc) \vdash_N (q_3, c) \implies (q_1, abc) \vdash_N^* (q_3, c)$

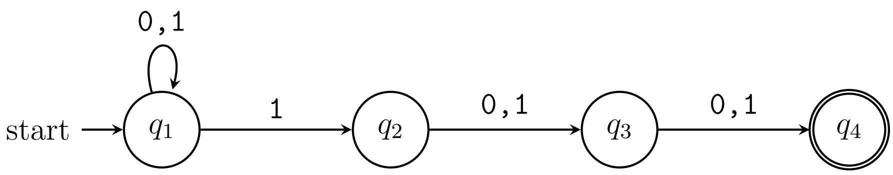
Una stringa $x \in \Sigma^*$ sarà accettata da un NFA se:

$$\exists q \in F | (q_0, x) \vdash_N^* (q, \varepsilon)$$

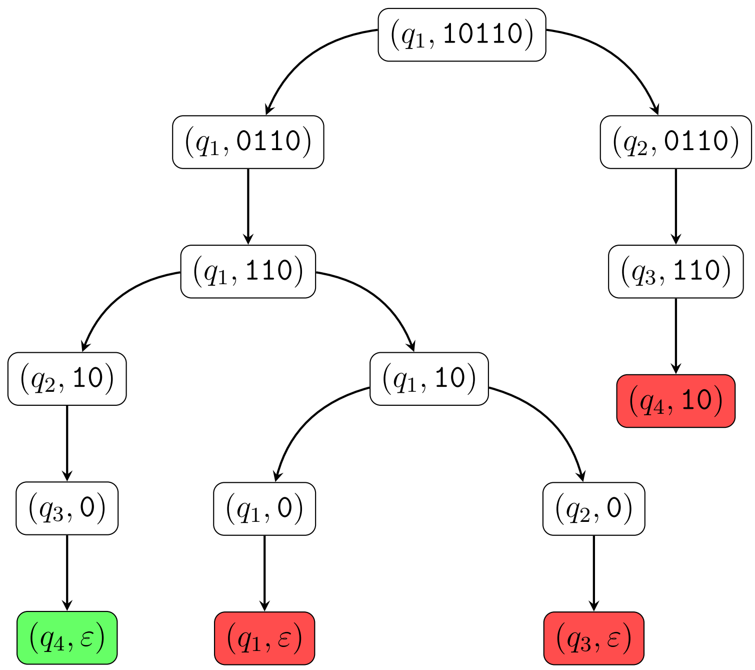
Oppure se:

$$x = y_1 \dots y_n \wedge \exists \underbrace{q_0 \dots q_n}_{\text{sequenza di stati}} | q_{i+1} = \delta(q_i, y_{i+1}) \wedge q_n \in F$$

Esempio:
Preso il seguente NFA:



Data la stringa `10110` la computazione sarà:



La stringa `10110` viene quindi accettata dal NFA.

1.3 Linguaggi regolari

Insieme dei Linguaggi regolari

Dato un alfabeto Σ, l'insieme dei **linguaggi regolari** di Σ, scritto come REG, è l'insieme dei linguaggi per cui esiste una DFA che li accetta:

$$\text{REG} = \{L \subseteq \Sigma^* | \exists D \; L(D) = L\}$$

1.3.1 Proprietà dei linguaggi regolari

I linguaggi sono insiemi di stringhe di un alfabeto Σ, quindi dati due linguaggi $L_1, L_2 \subseteq \Sigma^*$ possiamo definire le operazioni:

- **Unione:**
$$L_1 \cup L_2 = \{x \in \Sigma^* | x \in L_1 \lor x \in L_2\}$$
- **Intersezione:**
$$L_1 \cap L_2 = \{x \in \Sigma^* | x \in L_1 \land x \in L_2\}$$
- **Complemento:**
$$\neg L = \{x \in \Sigma^* | x \notin L\}$$
- **Concatenazione:**
$$L_1 \circ L_2 = \{xy \in \Sigma^* | x \in L_1 \land y \in L_2\}$$
- **Potenza:**
$$L^n = \begin{cases} \{\varepsilon\} & n = 0 \\ L \circ L^{n-1} & n > 0 \end{cases}$$
- **Star di Kleene:**
$$L^* = \{x_1 \dots x_k \in \Sigma^* | x_i \in L\} = \bigcup_{n \geq 0} L^n$$

1.3.2 Chiusura dei linguaggi regolari

Chiusura dell'unione in REG

L'unione è chiusa in REG, cioè:

$$\forall L_1, L_2 \in \text{REG} \implies L_1 \cup L_2 \in \text{REG}$$

Dimostrazioni:

- **Tramite DFA:**
  Dati $L_1, L_2 \in \text{REG}$, allora esistono:
  - $D_1 = (Q_1, \Sigma, \delta_1, q_1, F_1) | L(D_1) = L_1$
  - $D_2 = (Q_2, \Sigma, \delta_2, q_2, F_2) | L(D_2) = L_2$

  Creo il DFA $D_0 = (Q_0, \Sigma, \delta_0, q_0, F_0)$ per cui:
  - $Q_0 = Q_1 \times Q_2 = \{(q_x, q_y) | q_x \in Q_1 \wedge q_y \in Q_2\}$
  - $q_0 = (q_1, q_2)$
  - $F_0 = (F_1 \times Q_2) \cup (Q_1 \times F_2) = \{(q_x, q_y) | q_x \in Q_1 \vee q_y \in Q_2\}$
  - $\forall (q_x, q_y) \in Q_0, a \in \Sigma$:
  $$\delta((q_x, q_y), a) = (\delta_1(q_x, a), \delta_2(q_y, a))$$

  Quindi $x \in L(D_0) \iff x \in L_1 \vee x \in L_2$, quindi $L_1 \cup L_2 \in \text{REG}$.

- **Tramite NFA:**
  Dati $L_1, L_2 \in \text{REG}$, allora esistono:
  - $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1) | L(N_1) = L_1$
  - $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2) | L(N_2) = L_2$

  Creo il NFA $N_0 = (Q_0, \Sigma, \delta_0, q_0, F_0)$per cui:
  - $Q_0 = Q_1 \cup Q_2 \cup \{q_0\}$
  - q_0 è un nuovo stato iniziale
  - $F_0 = F_1 \cup F_2$
  - $\forall q \in Q_0, a \in \Sigma$:
  $$\delta(q, a) = \begin{cases} \delta_1(q, a) & q \in Q_1 \\ \delta_2(q, a) & q \in Q_2 \\ \{q_1, q_2\} & q = q_0 \wedge a = \varepsilon \\ \emptyset & q = q_0 \wedge a \neq \varepsilon \end{cases}$$

  Quindi $x \in L(N_0) \iff x \in L_1 \vee x \in L_2$, quindi $L_1 \cup L_2 \in \text{REG}$.

Chiusura dell'intersezione in REG

L'intersezione è chiusa in REG, cioè:

$$\forall L_1, L_2 \in \text{REG} \implies L_1 \cap L_2 \in \text{REG}$$

Dimostrazione:
Dati $L_1, L_2 \in \text{REG}$, allora esistono:

- $D_1 = (Q_1, \Sigma, \delta_1, q_1, F_1) | L(D_1) = L_1$
- $D_2 = (Q_2, \Sigma, \delta_2, q_2, F_2) | L(D_2) = L_2$

Creo il DFA $D_0 = (Q_0, \Sigma, \delta_0, q_0, F_0)$ per cui:

- $Q_0 = Q_1 \times Q_2 = \{(q_x, q_y) | q_x \in Q_1 \wedge q_y \in Q_2\}$
- $q_0 = (q_1, q_2)$
- $F_0 = F_1 \times F_2$
- $\forall (q_x, q_y) \in Q_0, a \in \Sigma$:
$$\delta((q_x, q_y), a) = (\delta_1(q_x, a), \delta_2(q_y, a))$$

Quindi $x \in L(D_0) \iff x \in L_1 \wedge x \in L_2$, quindi $L_1 \cap L_2 \in \text{REG}$.

Chiusura del complemento in REG

Il complemento è chiuso in REG, cioè:

$$\forall L \in \text{REG} \implies \neg L \in \text{REG}$$

Dimostrazione:
Dato $L \in \text{REG}$, esiste $D = (Q, \Sigma, \delta, q_0, F) | L(D) = L$.
Creo il DFA $D^* = (Q, \Sigma, \delta, q_0, Q - F)$, uguale a D ma con gli stati accettanti invertiti, quindi $x \in L(D^*) \iff x \notin L(D)$, quindi $\neg L \in \text{REG}$.

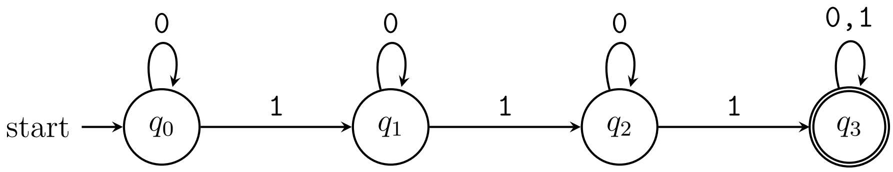
E

Esercizi

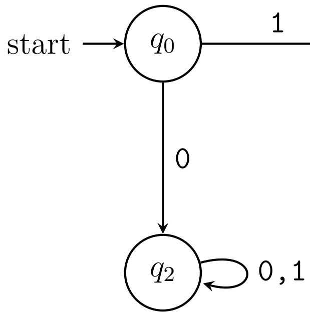
E.1 Esercizi sui linguaggi regolari

E.1.1 Costruire un DFA da un linguaggio

1. Dato un linguaggio $L(D) = \{x \in \{0,1\}^* | w_H(x) \geq 3\}$, per cui $w_H(x) = \{\text{numero di 1 in } x\}$, costruire un DFA che accetta questo linguaggio:



2. Dato un linguaggio $L(D) = \{x \in \{0,1\}^* | x = 1y \wedge y \in 0,1^*\}$, costruire un DFA che accetta questo linguaggio:



3. Dato un linguaggio $L(D) = \{x \in \{0,1\}^* | x = 0^n1\}$, costruire un DFA che accetta questo linguaggio: