"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
Simone Bianco

7 ottobre 2023

Indice

Informazioni e Contatti

Appunti e riassunti personali raccolti in ambito del corso di *Automi, Calcolabilità e Complessità* offerto dal corso di laurea in Informatica dell'Università degli Studi di Roma "La Sapienza".

Ulteriori informazioni ed appunti possono essere trovati al seguente link: **https://github.com/Exyss/university-notes**. Chiunque si senta libero di segnalare incorrettezze, migliorie o richieste tramite il sistema di Issues fornito da GitHub stesso o contattando in privato l'autore :

- Email: **bianco.simone@outlook.it**
- LinkedIn: **Simone Bianco**

Gli appunti sono in continuo aggiornamento, pertanto, previa segnalazione, si prega di controllare se le modifiche siano già state apportate nella versione più recente.

Prerequisiti consigliati per lo studio:

Apprendimento del materiale relativo al corso *Progettazione di Algoritmi*.

Licence:

1

Linguaggi e Automi

1.1 Linguaggi

Definizione 1: Alfabeto

Definiamo come **alfabeto** un insieme finito di elementi detti **simboli**

Esempio:

- L'insieme $\Sigma = \{0, 1, \mathrm{x}, \mathrm{y}, \mathrm{z}\}$ è un alfabeto
- L'insieme $\Sigma = \{0, 1\}$ è un alfabeto. In particolare, tale alfabeto viene detto **alfabeto binario**

Definizione 2: Stringa

Dato un alfabeto Σ, definiamo come **stringa di** Σ una sequenza di simboli $x_1x_2 \ldots x_n$ dove $x_1, \ldots, x_n \in \Sigma$ e $n \in \mathbb{N}$.

In particolare, indichiamo come ε la **stringa vuota**

Esempio:

- Dato l'alfabeto $\Sigma = \{0, 1, \mathrm{x}, \mathrm{y}, \mathrm{z}\}$, una stringa di Σ è 0x1yyy0

Definizione 3: Linguaggio

Dato un alfabeto Σ, definiamo come **linguaggio di** Σ, indicato come Σ^*, l'insieme delle stringhe di Σ.

In particolare, notiamo che $\varepsilon \in \Sigma^*$ per qualsiasi linguaggio Σ^*

Definizione 4: 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** la seguente operazione:

$$xy = x_1 \dots x_n y_1 \dots y_n$$

Definizione 5: Potenza

Data la stringa $x \in \Sigma^*$ e dato $n \in \mathbb{N}$, definiamo come **potenza** la seguente operazione:

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

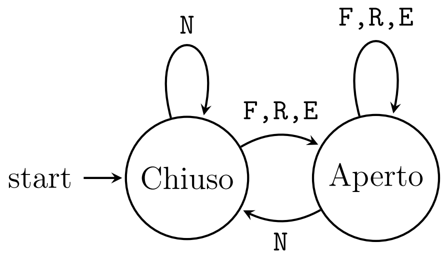
1.2 Determinismo

Definizione 6: Automa

Un **automa** è un meccanismo di controllo (o macchina) progettato per seguire automaticamente una sequenza di operazioni o rispondere a istruzioni predeterminate, mantenendo informazioni relative allo **stato** attuale dell'automa stesso ed agendo di conseguenza, **passando da uno stato all'altro**.

Esempio:

- Un sensore che apre e chiude una porta può essere descritto tramite il seguente automa, dove `Chiuso` e `Aperto` sono gli stati dell'automa e `N`, `F`, `R` e `E` sono le operazioni di transizione tra i due stati indicanti rispettivamente:
  - `N`: il sensore non rileva alcuna persona da entrambi i lati della porta
  - `F`: il sensore rileva qualcuno nel lato frontale della porta
  - `R`: il sensore rileva qualcuno nel lato retrostante della porta
  - `E`: il sensore rileva qualcuno da entrambi i lati della porta



- L'automa appena descritto è in grado di interpretare una **stringa in input** che ne descriva la sequenza di operazioni da svolgere (es: la stringa `NFNNNFRR` terminerà l'esecuzione dell'automa sullo stato `Aperto`)

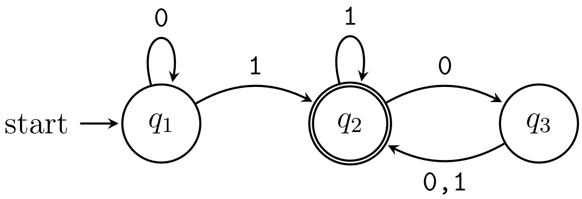
Definizione 7: Deterministic Finite Automaton (DFA)

Un **Deterministic Finite Automaton (DFA)** (o *Automa Deterministico a Stati Finiti*) è una quintupla $(Q, \Sigma, \delta, q_0, F)$ dove:

- Q è l'**insieme finito degli stati** dell'automa
- Σ è l'**alfabeto** dell'automa
- $\delta : Q \times \Sigma \to Q$ è la **funzione di transizione degli stati** dell'automa
- $q_0 \in Q$ è lo **stato iniziale** dell'automa
- $F \subseteq Q$ è l'**insieme degli stati accettanti** dell'automa, ossia l'insieme degli stati su cui, a seguito della lettura di una stringa in input, l'automa accetta la corretta terminazione

Esempio:

- Consideriamo il seguente DFA



dove:

  - $Q = \{q_1, q_2, q_3\}$ è l'insieme degli stati dell'automa
  - $\Sigma = \{0, 1\}$ è l'alfabeto dell'automa
  - $\delta : Q \times \Sigma \to Q$ definita come

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

    è la funzione di transizione degli stati dell'automa

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

Definizione 8: Funzione di transizione estesa

Sia $D := (Q, \Sigma, \delta, q_0, F)$ un DFA. Definiamo $\delta^* : Q \times \Sigma^* \to Q$ come **funzione di transizione estesa di** D la funzione definita ricorsivamente come:

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

Definizione 9: Stringa accettata

Sia $D := (Q, \Sigma, \delta, q_0, F)$ un DFA. Data una stringa $x \in \Sigma^*$, diciamo che x è **accettata da** D se $\delta^*(q_0, x) \in F$, ossia l'interpretazione di tale stringa **termina su uno stato accettante**

Esempio:

- Consideriamo ancora il DFA dell'esempio precedente.
- La stringa 0101 è accettata da tale DFA, poiché:

$$\delta^*(q_1, 0101) = \delta^*(\delta(q_1, 0), 101) = \delta^*(q_2, 101) = \delta^*(\delta(q_2, 1), 01) = \delta^*(q_2, 01) =$$

$$= \delta^*(\delta(q_2, 0), 1) = \delta^*(q_3, 1) == \delta^*(\delta(q_3, 1), \varepsilon) = \delta^*(q_2, \varepsilon) = q_2 \in F$$

- La stringa 1010, invece, non è accettata dal DFA, poiché:

$$\delta^*(q_1, 1010) = \delta^*(q_2, 010) = \delta^*(q_3, 10) = \delta^*(q_2, 0) = \delta^*(q_3, \varepsilon) = q_3 \notin F$$

Definizione 10: Linguaggio di un DFA

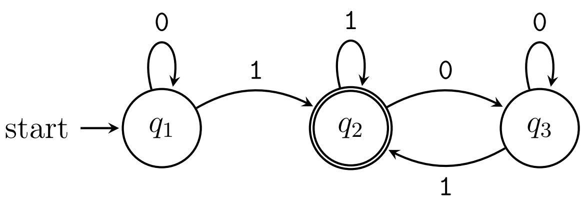
Sia $D := (Q, \Sigma, \delta, q_0, F)$ un DFA. Definiamo come **linguaggio di** D, indicato come $L(D)$, l'insieme di stringhe accettate da D

$$L(D) = \{x \in \Sigma^* \mid \delta^*(q_0, x) \in F\}$$

Inoltre, diciamo che D **riconosce** $L(D)$

Esempi:

1. - Consideriamo il seguente DFA D



- Il linguaggio riconosciuto da tale DFA corrisponde a

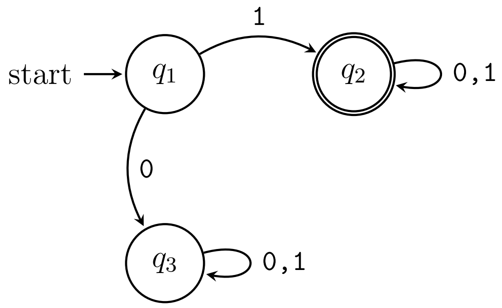
$$L(D) = \{x \in \{0,1\}^* \mid x := y1, \exists y \in \{0,1\}^*\}$$

ossia al linguaggio composto da tutte le stringhe terminanti con 1

2. • Consideriamo il seguente linguaggio

$$L = \{x \in \{0,1\}^* \mid 1y, \exists y \in \{0,1\}^*\}$$

- Un DFA in grado di riconoscere tale linguaggio corrisponde a

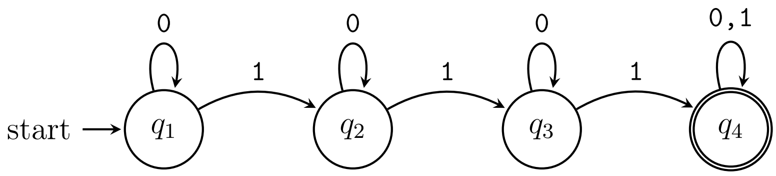


3. • Consideriamo il seguente linguaggio

$$L = \{x \in \{0,1\}^* \mid w_H(x) \geq 3\}$$

dove w_H è il **peso di Hamming** (ossia $w_H(x) = \texttt{numero di "1" in } x$)

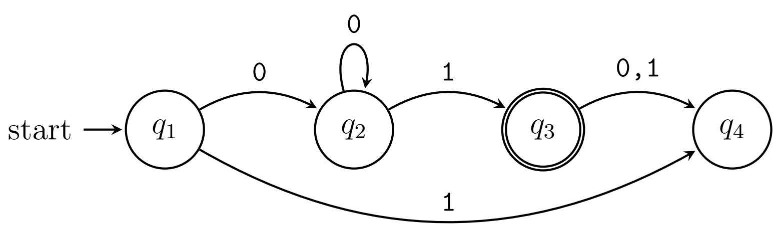
- Un DFA in grado di riconoscere tale linguaggio corrisponde a



4. • Consideriamo il seguente linguaggio

$$L = \{x \in \{0,1\}^* \mid 0^n 1, n \in \mathbb{N} - \{0\}\}$$

- Un DFA in grado di riconoscere tale linguaggio corrisponde a



Definizione 11: Configurazione di un DFA

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

Definizione 12: Passo di computazione

Definiamo come **passo di computazione** la relazione binaria definita come

$$(p, ax) \vdash_D (q, x) \iff \delta(p, a) = q$$

Definizione 13: Computazione deterministica

Definiamo una computazione come **deterministica** se ad ogni passo di computazione segue un'unica configurazione:

$$\forall (q, ax) \;\; \exists!(p, x) \mid (q, ax) \vdash_D (p, x)$$

Proposizione 1: Chiusura del passo di computazione

Sia $D := (Q, \Sigma, \delta, q_0, F)$ un DFA. La **chiusura riflessiva e transitiva** di $\vdash_D$, indicata come $\vdash_D^*$, gode delle seguenti proprietà:

- $(p, ax) \vdash_D (q, x) \implies (p, ax) \vdash_D^* (q, x)$
- $\forall q \in Q, x \in \Sigma^* \;\; (q, x) \vdash_D^* (q, x)$
- $(p, aby) \vdash_D (q, by) \wedge (q, by) \vdash_D (r, y) \implies (p, aby) \vdash_D^* (r, y)$

Osservazione 1

Sia $D := (Q, \Sigma, \delta, q_0, F)$ un DFA. Dati $q_i, q_f \in Q, x \in \Sigma^*$, si ha che

$$\delta^*(q_i, x) = q_f \iff (q_i, x) \vdash_D^* (q_f, \varepsilon)$$

(*dimostrazione omessa*)

1.3 Non determinismo

Definizione 14: Alfabeto epsilon

Dato un alfabeto Σ, definiamo $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$ come **alfabeto epsilon di** Σ

Definizione 15: Non-deterministic Finite Automaton (NFA)

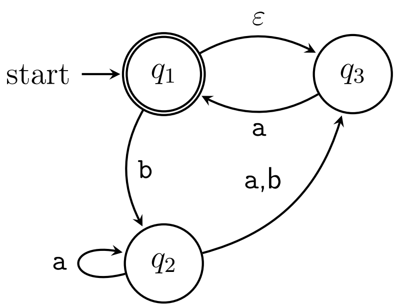
Un **Non-deterministic Finite Automaton (NFA)** (o *Automa Non-deterministico a Stati Finiti*) è una quintupla $(Q, \Sigma, \delta, q_0, F)$ dove:

- Q è l'**insieme finito 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** dell'automa
- $q_0 \in Q$ è lo **stato iniziale** dell'automa
- $F \subseteq Q$ è l'**insieme degli stati accettanti** dell'automa

Nota: $\mathcal{P}(Q)$ è l'insieme delle parti di Q, ossia l'insieme contenente tutti i suoi sottoinsiemi possibili

Esempio:

- Consideriamo il seguente NFA



dove:

- $Q = \{q_1, q_2, q_3\}$ è l'insieme degli stati dell'automa
- $\Sigma_\varepsilon = \{\varepsilon, \}$ è l'alfabeto dell'automa
- $\delta : Q \times \Sigma \to Q$ definita come

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

è la funzione di transizione degli stati dell'automa

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

Proposizione 2: Stringa accettata in un NFA

Sia $N := (Q, \Sigma_\varepsilon, \delta, q_0, F)$ un NFA. Data una stringa $x := x_0 \dots x_k \in \Sigma_\varepsilon^*$, diciamo che x è **accettata** se esiste una sequenza di stati $r_0, r_1, \dots, r_k \in Q$ tali che:

- $r_0 = q_0$
- $\forall i \in [0, k] \;\; r_{i+1} \in \delta(r_i, x_0)$
- $r_m \in F$

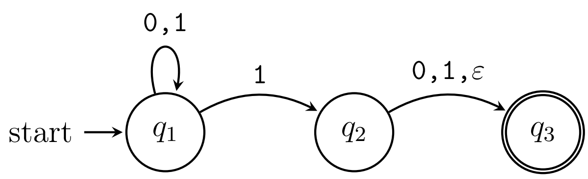
Osservazione 2: Computazione in un NFA

Sia $N := (Q, \Sigma, \delta, q_0, F)$ un NFA. Data una stringa $x \in \Sigma_\varepsilon$ in ingresso, la **computazione** viene eseguita nel seguente modo:

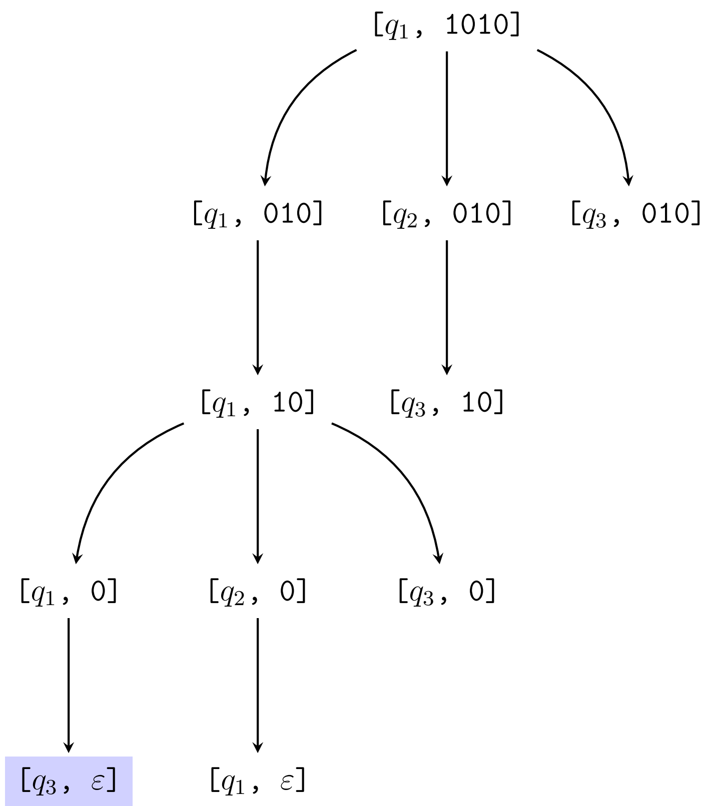
- Tutte le volte che uno stato potrebbe avere più transizioni per diversi simboli dell'alfabeto, l'automa N si duplica in **più copie**, ognuna delle quali segue il suo corso. Si vengono così a creare più **rami di computazione** indipendenti che sono eseguiti in **parallelo**.
- Se il prossimo simbolo della stringa da computare non si trova su nessuna delle transizioni uscenti dello stato attuale di un ramo di computazione, l'intero ramo **termina la sua computazione** (terminazione incorretta).
- Se almeno una delle copie di N termina correttamente su uno stato di accettazione, l'automa **accetta la stringa di partenza**.
- Quando a seguito di una computazione ci si ritrova in uno stato che possiede un ε**-arco** in uscita, la macchina si duplica in più copie: quelle che seguono gli ε-archi e quella che rimane nello stato raggiunto.

Esempio:

- Consideriamo il seguente NFA



- Supponiamo che venga computata la stringa `x = 1010`:



- Poiché esiste un ramo che termina correttamente, l'NFA descritto accetta la stringa `x = 1010`

1.3.1 Equivalenza tra NFA e DFA

Definizione 16: Classe dei linguaggi riconosciuti da un DFA

Dato un alfabeto Σ, definiamo come **classi dei linguaggi riconosciuti da un DFA** il seguente insieme:

$$\mathcal{L}(\text{DFA}) = \{L \subseteq \Sigma^* \mid \exists \text{DFA } D \text{ t.c } L = L(D)\}$$

Definizione 17: Classe dei linguaggi riconosciuti da un NFA

Dato un alfabeto Σ, definiamo come **classi dei linguaggi riconosciuti da un NFA** il seguente insieme:

$$\mathcal{L}(\text{NFA}) = \{L \subseteq \Sigma_\varepsilon^* \mid \exists \text{NFA } N \text{ t.c } L = L(N)\}$$

Teorema 1: Equivalenza tra NFA e DFA

Date le due classi di linguaggi $\mathcal{L}(\text{DFA})$ e $\mathcal{L}(\text{NFA})$, si ha che:

$$\mathcal{L}(\text{DFA}) = \mathcal{L}(\text{NFA})$$

Dimostrazione.

Prima implicazione.

- Dato $L \in \mathcal{L}(\text{DFA})$, sia $D := (Q, \Sigma, \delta, q_0, F)$ il DFA tale che $L = L(D)$
- Poiché il concetto di NFA è una generalizzazione del concetto di DFA, ne segue automaticamente che D sia anche un NFA, implicando che $L \in \mathcal{L}(\text{NFA})$ e di conseguenza che:
$$\mathcal{L}(\text{DFA}) \subseteq \mathcal{L}(\text{NFA})$$

Seconda implicazione.

- Dato $L \in \mathcal{L}(\text{NFA})$, sia $N := (Q_N, \Sigma_\varepsilon, \delta_N, q_{0_N}, F_N)$ il NFA tale che $L = L(N)$
- Consideriamo quindi il DFA $D := (Q_D, \Sigma, \delta_D, q_{0_D}, F_D)$ costruito tramite N stesso:
  - $Q_D = \mathcal{P}(Q_N)$
  - Dato $R \in Q_D$, definiamo l'estensione di R come:
  $$E(R) = \{q \in Q_N \mid \text{q è raggiungibile in } N \text{ da } q' \in R \text{ tramite } k \geq 0 \ \varepsilon\text{-archi}\}$$
  - $q_{0_D} = E(\{q_{0_N}\})$
  - $F_D = \{R \in Q_D \mid R \cap F_N \neq \varnothing\}$
  - Dati $R \in Q_D$ e $a \in \Sigma$, definiamo δ_D come:
  $$\delta_D = (R, a) = \bigcup_{r \in R} E(\delta_N(r, a))$$
- A questo punto, per costruzione stessa di D si ha che:
$$x \in L = L(N) \iff x \in L(D)$$
implicando dunque che $L \in \mathcal{L}(\text{DFA})$ e di conseguenza che:
$$L \in \mathcal{L}(\text{NFA}) \subseteq \mathcal{L}(\text{DFA})$$

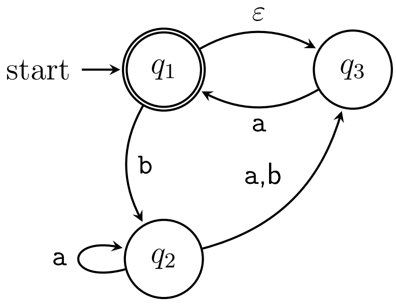
$\square$

Osservazione 3

Dato un NFA N, seguendo i passaggi della dimostrazione precedente è possibile definire un DFA D equivalente ad N

Esempio:

- Consideriamo ancora il seguente NFA



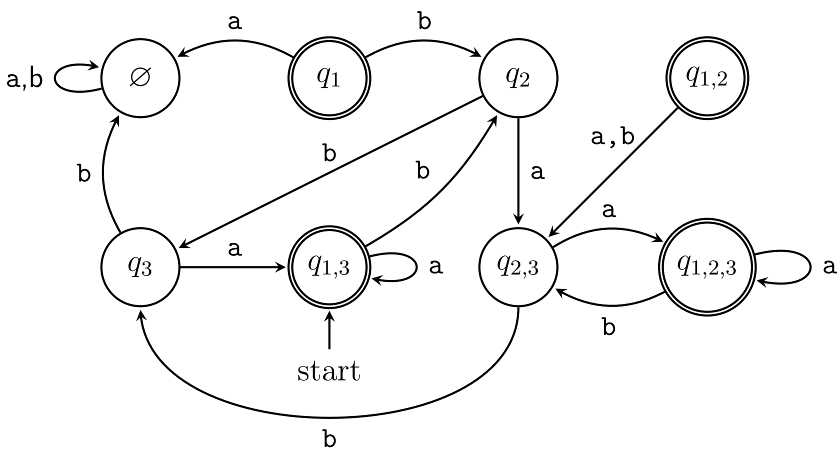
- Definiamo quindi l'insieme degli stati del DFA equivalente a tale NFA:

$$Q_D = \{\varnothing, \{q_1\}, \{q_2\}, \{q_3\}, \{q_1, q_2\}, \{q_2, q_3\}, \{q_1, q_3\}, \{q_1, q_2, q_3\}\} =$$
$$= \{\varnothing, q_1, q_2, q_3, q_{1,2}, q_{2,3}, q_{1,3}, q_{1,2,3}\}$$

- A questo punto, lo stato iniziale sarà $q_{0_D} = E(\{q_{0_N}\}) = E(\{q_1\}) = \{q_1, q_3\} = q_{1,3}$, mentre gli stati accentanti saranno:

$$F_D = \{\{q_1\}, \{q_1, q_2\}, \{q_1, q_3\}, \{q_1, q_2, q_3\}\} = \{q_1, q_{1,2}, q_{1,3}, q_{1,2,3}\}$$

- Le transizioni del DFA corrisponderanno invece a:
  - $\delta_D(\{q_1\}, a) = E(\delta_N(q_1,), a) = \varnothing$
  - $\delta_D(\{q_1\}, b) = E(\delta_N(q_1,), b) = \{q_2\} = q_2$
  - $\delta_D(\{q_2\}, a) = E(\delta_N(q_2,), a) = \{q_2, q_3\} = q_{2,3}$
  - $\delta_D(\{q_2\}, b) = E(\delta_N(q_2,), b) = \{q_2\} = q_2$
  - $\delta_D(\{q_1, q_2\}, a) = E(\delta_N(q_1, a)) \cup E(\delta_N(q_2, a)) = \varnothing \cup \{q_2, q_3\} = \{q_2, q_3\} = q_{2,3}$
  - $\delta_D(\{q_1, q_2\}, b) = E(\delta_N(q_1, b)) \cup E(\delta_N(q_2, b)) = \{q_2\} \cup \{q_3\} = \{q_2, q_3\} = q_{2,3}$
  - ...
- Il DFA equivalente corrisponde dunque a:



1.4 Linguaggi regolari

Definizione 18: Linguaggi regolari

Dato un linguaggio Σ^*, definiamo come **insieme dei linguaggi regolari di** Σ^*, indicato con REG, l'insieme delle classi dei linguaggi riconosciuti:

$$REG := \mathcal{L}(\text{DFA}) = \mathcal{L}(\text{NFA})$$

Proposizione 3: Operazioni sui linguaggi

Dati due linguaggi $L_1, L_2 \subseteq \Sigma^*$, definiamo le seguenti operazioni:

- **Operatore unione**:

$$L_1 \cup L_2 = \{x \in \Sigma^* \mid x \in L_1 \lor x \in L_2\}$$

- **Operatore intersezione**:

$$L_1 \cap L_2 = \{x \in \Sigma^* \mid x \in L_1 \land x \in L_2\}$$

- **Operatore complemento**:

$$\neg L_1 = \{x \in \Sigma^* \mid x \notin L_1\}$$

- **Operatore concatenazione**:

$$L_1 \circ L_2 = \{xy \in \Sigma^* \mid x \in L_1, x \in L_2\}$$

- **Operatore potenza**:

$$L_1^n = \begin{cases} \{\varepsilon\} & \text{se } n = 0 \\ L_1 \circ L_1^{n-1} & \text{se } n > 0 \end{cases}$$

- **Operatore star**:

$$L_1^* = \{x_1 \dots x_k \in \Sigma^* \mid k \geq 0, \forall i \in [1, k] \;\; x_i \in L_1\} = \bigcup_{n \geq 0} L^n$$

Teorema 2: Chiusura dell'unione in ***REG***

L'operatore unione è **chiuso in** REG, ossia:

$$\forall L_1, L_2 \in REG \;\; L_1 \cup L_2 \in REG$$

Dimostrazione.

- Dati due linguaggi $L_1, L_2 \in REG$, siano $D_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ e $D_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ i due DFA tali che:
$$L_1 = L(D_1) \qquad L_2 = L(D_2)$$
- Definiamo quindi il DFA $D = (Q, \Sigma, \delta, q_0, F)$ tale che:
  - $Q = Q_1 \times Q_2 = \{(r_1, r_2) \mid r_1 \in Q_1, r_2 \in Q_2\}$
  - $\forall (r_1, r_2) \in Q, a \in \Sigma$ si ha che $\delta((r_1, r_2), a) = (\delta(r_1, a), \delta(r_2, a))$
  - $F = \{(r_1, r_2) \mid r_1 \in F_1 \vee r_2 \in F_2\} = (F_1 \times Q_2) \cup (Q_1 \times F_2)$
- A questo punto, per costruzione stessa di D ne segue che:
$$x \in L_1 \cup L_2 \iff D(x) \in F \iff x \in L(D)$$
da cui concludiamo che:
$$L_1 \cup L_2 = L(D) \implies L_1 \cup L_2 \in REG$$

□

Teorema 3: Chiusura dell'intersezione in *REG*

L'operatore intersezione è **chiuso in** REG, ossia:
$$\forall L_1, L_2 \in REG \;\; L_1 \cap L_2 \in REG$$

Dimostrazione.

- Dati due linguaggi $L_1, L_2 \in REG$, siano $D_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ e $D_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ i due DFA tali che:
$$L_1 = L(D_1) \qquad L_2 = L(D_2)$$
- Definiamo quindi il DFA $D = (Q, \Sigma, \delta, q_0, F)$ tale che:
  - $Q = Q_1 \times Q_2 = \{(r_1, r_2) \mid r_1 \in Q_1, r_2 \in Q_2\}$
  - $\forall (r_1, r_2) \in Q, a \in \Sigma$ si ha che $\delta((r_1, r_2), a) = (\delta(r_1, a), \delta(r_2, a))$
  - $F = \{(r_1, r_2) \mid r_1 \in F_1 \wedge r_2 \in F_2\} = F_1 \times F_2$
- A questo punto, per costruzione stessa di D ne segue che:
$$x \in L_1 \cap L_2 \iff D(x) \in F \iff x \in L(D)$$
da cui concludiamo che:
$$L_1 \cap L_2 = L(D) \implies L_1 \cap L_2 \in REG$$

□