LINGUAGGI REGOLARI

Automa → modello matematico di calcolo (macchina teorica o reale) che può elaborare informazioni e agire in modo automatico seguendo una serie di stati predefiniti

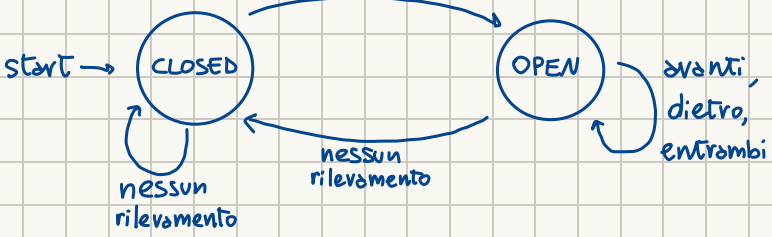
AUTOMA A STATI FINITI DETERMINISTICO (DFA)

Deterministic Finite Automaton

Possiede un numero finito di stati:
- processa input bit a bit sequenzialmente

esempio: sistema di controllo di una porta automatica

- ha dei sensori "avanti" e "dietro" (lati porta)



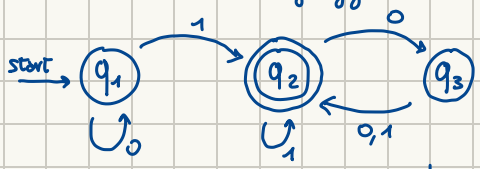
DFA (def. formale)

Un DFA è una quintupla $(Q, \Sigma, \delta, q_0, F)$, dove:

- Q : insieme finito degli stati
- Σ : insieme finito dei simboli in input (alfabeto)
- $\delta : Q \times \Sigma \to Q$: funzione di transizione (permette il passaggio di stato)
- q_0 : stato iniziale
- $F \subseteq Q$: insieme degli stati di accettazione (o "finali") — si indicano così ◎ o così ▨

↳ ogni automa, ricevuto un input, lo elabora e restituisce un output. L'output è "ACCETTA" se l'automa termina in uno stato di accettazione, o "RIFIUTA" altrimenti

es. 2 - DFA di un linguaggio binario



Se riceve in input $w = 11101$, l'automa accetta

stati → $q_1 \xrightarrow{1} q_2 \xrightarrow{1} q_2 \xrightarrow{1} q_2 \xrightarrow{0} q_3 \xrightarrow{1} q_2$ finale

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

	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

- $q_0 = q_1$
- $F = \{q_2\}$

LINGUAGGIO

Se M è un DFA, l'insieme di stringhe riconosciute ("accettate") da M si denota con $L(M)$

per esempio, il DFA dell'esempio sopra ha come linguaggio:

$$A = \{ w \mid w \text{ contiene almeno un "1" e un numero pari di "0" segue l'ultimo "1"} \}$$

quindi, $L(M) = A$ o, equivalentemente, "M riconosce A"

Per definire formalmente un linguaggio, si introduce la

funzione di transizione estesa δ^* (a più di un carattere)

- $\delta^* : Q \times \Sigma^* \longrightarrow Q$
- $\delta^*(q, \varepsilon) = \delta(q, \varepsilon)$
- $\delta^*(q, ax) = \delta^*(\delta(q,a), x)$ — si computa il primo simbolo e poi tutto il resto (ricorsivamente)

con Σ^* = insieme di stringhe in input, $x \in \Sigma^*$, $a \in \Sigma$

Si introduce anche la:

configurazione ⟶ data da una tupla $(①, ②) \in Q \times \Sigma^*$, con elementi

① lo stato ② quello che resta da leggere

Dato $x \in \Sigma^*$, la configurazione iniziale è (q_0, x)

Un passo di computazione porta da una configurazione ad un'altra

↳ relazione binaria tra configurazioni $\vdash$

si ha: $(p, ax) \vdash_M (q, x) \iff \delta(p, a) = q$ con $p, q \in Q$, $a \in \Sigma$, $x \in \Sigma^*$

dalla conf. di sinistra si passa, con un passo di computazione, a quella di destra

La relazione binaria $\vdash_M$ si può estendere considerando chiusura riflessiva e transitiva $\vdash_M^*$

- $(q, x) \vdash_M^* (q, x)$ (non legge nessun input)
- $(q, aby) \vdash_M (p, by) \wedge (p, by) \vdash_M (r, y) \implies (q, aby) \vdash_M^* (r, y)$

con $q, p, r \in Q$ $a, b \in \Sigma$, $y \in \Sigma^*$

Possiamo usare questi concetti per formalizzare la definizione di linguaggio accettato

linguaggio accettato (def. formale)

Diciamo che $x \in \Sigma^*$ è accettato da $M = (Q, \Sigma, \delta, q_0, F)$ se

- $\delta^*(q_0, x) \in F$ (se δ^* (iniziale, input) porta ad uno stato finale)
- o anche se $(q_0, x) \vdash_{M^*} (q, \varepsilon)$ dove $q \in F$ (se attraverso $\vdash_{M^*}$, si arriva ad uno stato finale e non resta niente da leggere)

In altre parole, $L(M) = \{ x \in \Sigma^* : \delta^*(q_0, x) \in F \}$

LINGUAGGIO REGOLARE

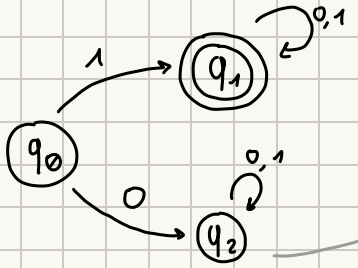
Un linguaggio è chiamato linguaggio regolare se un automa finito lo riconosce.

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

Vogliamo capire come costruire un DFA per un linguaggio

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

($\|$ = concatenazione logica ($1\|y$ sono le stringhe che iniziano con 1)



automa A

lo stato q_2 è uno "stato pozzo" (non è lo stato d'accettazione e non ha archi uscenti verso altri stati)
l'automa A è equivalente ad un automa A' senza gli archi di q_2

Vogliamo sviluppare una prova di correttezza, ovvero dimostrare che il DFA accetta $x \iff x \in L$

per l'automa A, notiamo che:
- $\delta^*(q_1, v) = q_1 \quad \forall v \in \{0,1\}^*$
- $\delta^*(q_2, v) = q_2 \quad \forall v \in \{0,1\}^*$

Costruiamo la dimostrazione per induzione:

CASO BASE: $|x| = 0$, quindi $x = \varepsilon$

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

IPOTESI INDUTTIVA Sia $n>0$. Supponiamo $|w| \leq n$.

Si ha:

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

(Sia q_1 che q_2 non hanno archi uscenti verso altri stati, quindi non importa quali siano le cifre successive

PASSO INDUTTIVO

Prendo x t.c. $|x| = n+1$. Posso scriverlo come $x = au$, con $a \in \{0,1\}$, $u \in \{0,1\}^*$

Ho quindi $\delta^*(q_0, x) = \delta^*(q_0, au) = \delta^*(\delta(q_0, a), u)$

ha 2 possibili soluzioni:

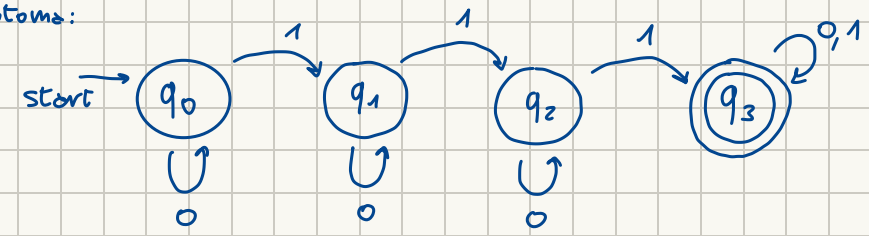
e, visto che $\delta(q_0, a) = \begin{cases} q_2 & \text{se } a = 0 \\ q_1 & \text{se } a = 1 \end{cases}$ → quindi, l'automa andrà sicuramente in q_2 o in q_1, e, poiché da entrambi non ci sono archi uscenti, vi terminerà

$\delta^*(q_0, x) = q_1 \iff a = 1$ ← si ha quindi

esercizi

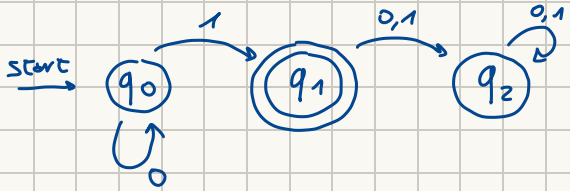
- $L = \{ x \in \{0,1\}^* \mid W_H(x) \geq 3 \}$ dove $W_H(x) = \#1$ (numero di 1 nella stringa)

automa:



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

automa:



OPERAZIONI SUI LINGUAGGI

Fissiamo $\Sigma = \{0,1\}$. Per $n \in \mathbb{N}$, $[n] = \{1, \dots, n\}$

Poiché i linguaggi sono insiemi di stringhe, possiamo fare operazioni su di essi

- 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 $\overline{L_1} = \{ x \in \Sigma^* : x \notin L_1 \}$
- CONCATENAZIONE $L_1 \circ L_2 = \{ xy : x \in L_1 \wedge y \in L_2 \}$
  - si ha $\varepsilon x = x\varepsilon = x$, $x(ya) = (xy)a = xya$

es. $\Sigma = \{a,b\}$ $L_1 = \{ a, ab, ba \}$ $L_2 = \{ab, b\}$

$L_1 \circ L_2 = \{ aab, ab, abab, abb, baab, bab \}$

- POTENZA si definisce ricorsivamente

$$\begin{cases} x^0 = \varepsilon \\ x^{n+1} = x^n x \end{cases}$$ stringhe

$$\begin{cases} L^0 = \{\varepsilon\} \\ L^{n+1} = L^n \circ L \end{cases}$$ linguaggi

es. $L = \{ a, ab, ba \}$

$L^2 = L \circ L = \{ aa, aab, aba, abab, abba, baa, baab, baba \}$

- OPERATORE $*$ $L^* = \bigcup_{n \geq 0} L^n = \{\varepsilon\} \cup L^1 \cup L^2 \dots$ (tutte le combinazioni)

es. $L = \{a,b\}$ $L^* = \{\varepsilon, a, b, aa, ab, ba, bb, aaa \dots\}$

CHIUSURA DEI LINGUAGGI REGOLARI

Vogliamo capire se i linguaggi regolari sono chiusi per le loro operazioni (se $L_1, L_2 \in REG \rightarrow L_1 \cup L_2 \in REG$) ecc.

TEOREMA

REG è chiuso per unione

TODO

CHIUSURE

NON DETERMINISMO

Nella computazione non deterministica:

- quando un automa si trova in uno stato $q \in Q$ e legge $\sigma \in \Sigma$, può andare in diversi stati
- sono ammessi "ε-archi", che consentono di aprire rami di computazione parallela senza leggere nulla
- l'automa accetta se e solo se esiste almeno un ramo che accetta

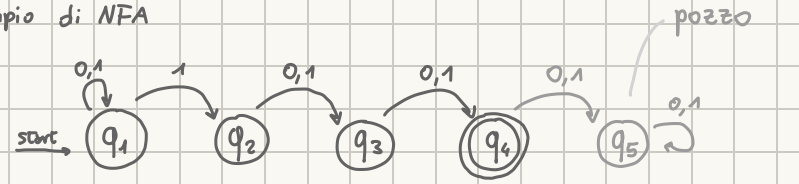
NFA

Un NFA è una tupla $(Q, \Sigma, \delta, q_0, F)$, dove

(come nei DFA)

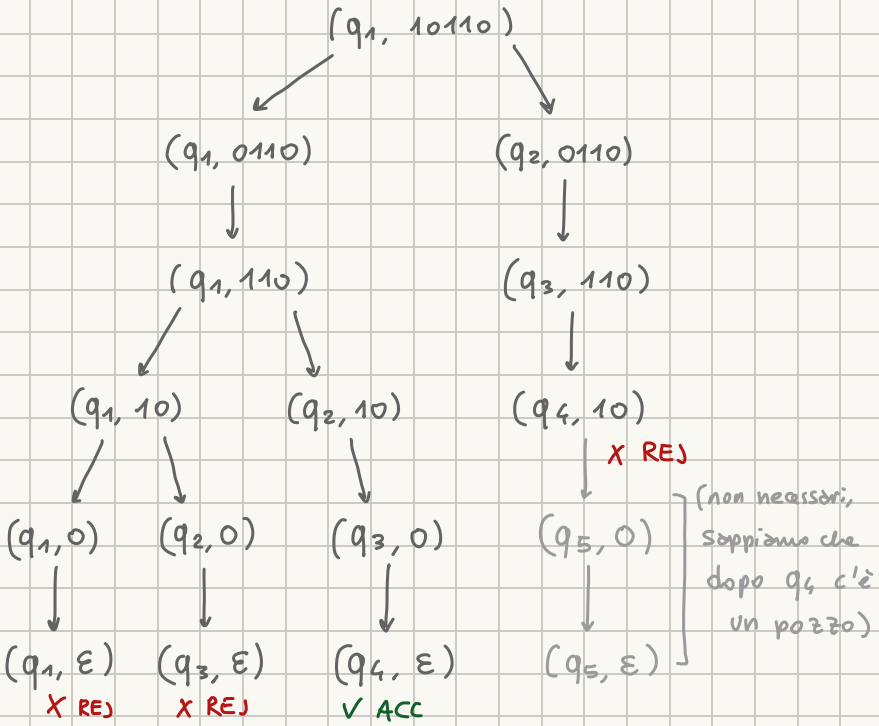
- Q: insieme finito degli stati
- Σ: alfabeto
- q_0: stato iniziale
- $F \subseteq Q$: stati di accettazione
- $\delta : Q \times \Sigma_\varepsilon \longrightarrow \mathbb{P}(Q)$ (insieme delle parti di Q: insieme di tutti i possibili sottoinsiemi)
  - $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$

esempio di NFA



$L = \{x : x \in \{0,1\}^* : x \text{ ha un "1" in terzultima posizione}\}$

computazione sulla stringa "10110"

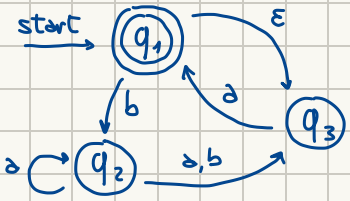


la computazione in un NFA avviene diversamente:

- quando uno stato ha più transizioni sullo stesso input, l'automa si duplica
  - ↳ si hanno quindi più rami di computazione indipendenti, che sono eseguiti in parallelo

- se l'automa si trova in uno stato che possiede un ε-arco uscente, l'automa si duplica: segue l'ε-arco e, parallelamente, rimane nello stato raggiunto

esercizio



- $w = \varepsilon$
  - (q_1, ε) ✓
    - (q_3, ε) ×
- $w = a$
  - (q_1, a) ×
  - $(q_3, a) \to (q_1, \varepsilon)$ ✓
- $w = bb$
  - (q_1, bb)
    - $(q_2, b) \to (q_3, \varepsilon)$ ×
    - (q_3, bb) ×
- $w = baabba$
  - $(q_1, baabba)$
    - $(q_2, abba) \to (q_3, bba)$ ×
    - $(q_3, baabba)$ ×

configurazione, computazione, accettazione in un NFA

- Dato un NFA N, indichiamo come configurazione una coppia $(q, x) \in Q, \Sigma_\varepsilon^*$
- Avremo

$$(p, ax) \vdash_N (q, x) \iff q \in \delta(p, a) \quad \text{con } x \in \Sigma_\varepsilon^*,\ a \in \Sigma_\varepsilon,\ p, q \in \Sigma$$

quindi, visto che $\delta: Q \times \Sigma_\varepsilon \to \mathbb{P}(Q)$, si deve avere $\delta(p, a) \in \mathbb{P}(Q)$

- Un NFA N accetta $w \in \Sigma_\varepsilon^* \iff \exists q \in F$ t.c. $(q_0, w) \vdash_N^* (q, \varepsilon)$

($\vdash_N^*$: relazione estesa (come per i DFA))

EQUIVALENZA TRA DFA E NFA

Definiamo le due classi

$$\mathcal{L}(DFA) \overset{?}{\subseteq} REG \quad (= ??)$$

$$\mathcal{L}(NFA) = \{ L : \exists \text{ NFA } N \text{ t.c. } L(N) = L \}$$

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

dim Dobbiamo dimostrare $\mathcal{L}(DFA) \subseteq \mathcal{L}(NFA)$ e $\mathcal{L}(DFA) \supseteq \mathcal{L}(NFA)$

① PRIMA IMPLICAZIONE $\mathcal{L}(DFA) \subseteq \mathcal{L}(NFA)$

Dato $L \in \mathcal{L}(DFA)$, sia $D := (Q, \Sigma, \delta, q_0, F)$ t.c. $L = L(D)$

il concetto di NFA è una generalizzazione del concetto di DFA quindi D è anche un NFA, e $L \in \mathcal{L}(NFA)$

(ogni DFA è un NFA)

② seconda implicazione $\mathcal{L}(NFA) \subseteq \mathcal{L}(DFA)$

(idea: trasformare l'NFA in un DFA che lo simula)

- sia $N = \{Q, \Sigma, \delta, q_0, F\}$ l'NFA t.c. $L(N) = A$

costruiamo

$M = \{Q_M, \Sigma, \delta_M, q_0^M, F_M\}$ t.c. $L(M) = A$

iniziamo senza considerare gli ε-archi:

- $Q_M = \mathbb{P}(Q)$ (ogni stato di M è un insieme di stati di N)
- $\delta_M(R, a) = \bigcup_{r \in R} \delta(r, a)$

ovvero con $R \in Q_M$, $a \in \Sigma$, $\delta'(R, a) = \{q \in Q \mid q \in \delta(r, a) \text{ con } r \in R\}$

(unione di tutti gli stati a cui portiamo tutti gli $r \in R$)

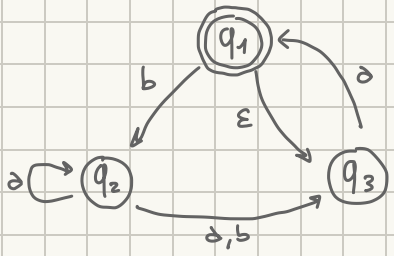
- $q_0^M = \{q_0\}$
- $F_M = \{R \in Q' \mid R \cap F \neq \emptyset\}$ stati che hanno al loro interno almeno uno stato accettante di N

introducendo gli ε-archi:

definiamo $E(R) = \{q \mid q$ può essere raggiunto da R attraverso ≥ 0 ε-archi$\}$ (0 perché includiamo se stesso)

quindi, abbiamo
- $\delta_M(R, a) = \{q \in Q \mid q \in E(\delta(r, a)), \exists r \in R\}$ (quelli di prima e quelli raggiunti con ε-archi)
- $q_0^M = E(\{q_0\})$

esempio:



notazione per $\{q_1, q_2\}$

- $Q_D = \{q_\emptyset, q_{\{1\}}, q_{\{2\}}, q_{\{3\}}, q_{\{1,2\}}, q_{\{1,3\}}, q_{\{2,3\}}, q_{\{1,2,3\}}\}$
- $q_0^D = E(\{q_{\{1\}}\}) = q_{\{1,3\}}$ (ε-arco da 1 a 3)
- $F_D = \{q_{\{1\}}, q_{\{1,2\}}, q_{\{1,3\}}, q_{\{1,2,3\}}\}$ ($q \in Q_D : q \cap \{q_1\} \neq \emptyset$)

rimane da definire δ:

stato q_1: $\delta_D(q_{\{1\}}, a) = E(\delta(q_1, a)) = \emptyset$

$\delta_D(q_{\{1\}}, b) = E(\delta(q_1, b)) = q_{\{2\}}$

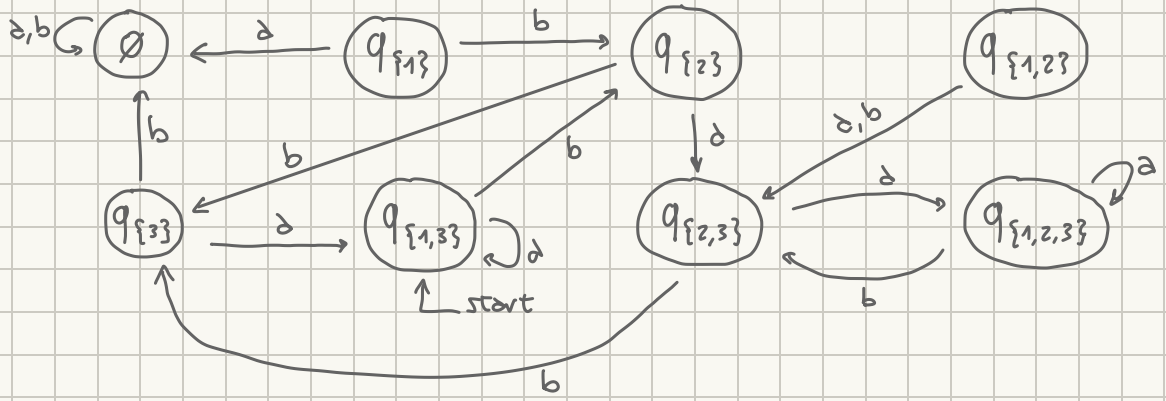
stato q_2: $\delta_D(q_{\{2\}}, a) = E(\delta(q_2, a)) = q_{\{2,3\}}$

$\delta_D(q_{\{2\}}, b) = \dots = q_{\{3\}}$

stato $q_{\{1,2\}}$: $\delta_D(q_{\{1,2\}}, a) = E(\delta(q_1, a)) \cup E(\delta(q_2, a)) = \emptyset \cup q_{\{2,3\}} = q_{\{2,3\}}$

⋮

otterremmo il diagramma



possiamo semplificare il diagramma osservando che
gli stati $q_{\{1\}}$ e $q_{\{1,2\}}$ non hanno archi entranti (e possono quindi essere rimossi)