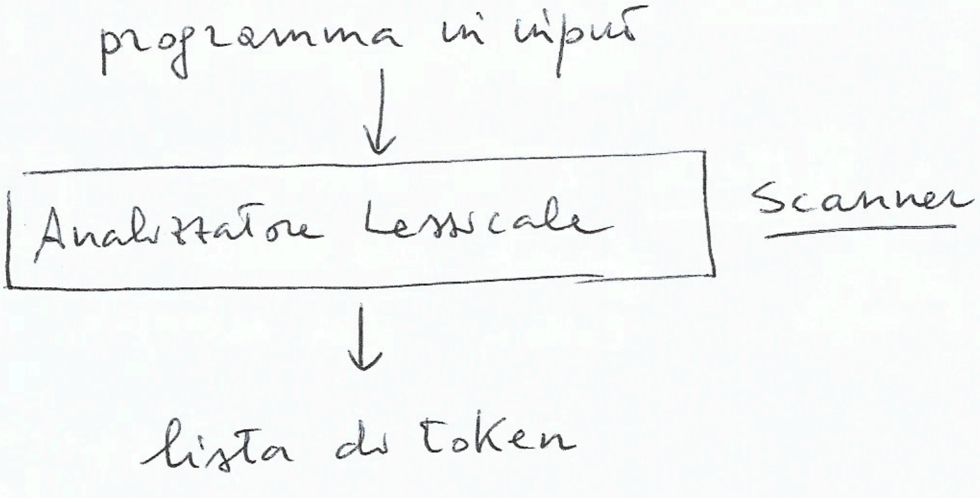
Capitolo 3

Analisi Lessicale – Linguaggi Regolari

Analisi Lessicale: riconoscere nella stringa in ingresso gruppi/sequenze di simboli che corrispondono a specifiche categorie sintattiche (ad es: identificatori, parole riservate, operatori aritmetici, ...)

La stringa in input è trasformata in una sequenza di simboli astratti, detti token



Cos'è un token?

Token = coppia (nome, valore)

- nome = simbolo astratto che rappresenta una categoria sintattica
- valore = una sequenza di simboli del testo in ingresso

ad es.: $\langle Ide, x_1 \rangle$

Dato il token $\langle Ide, x_1 \rangle$, diciamo che

- Nome (Ide) è l'informazione che identifica una classe di token (identificatori)
- Valore (x_1) è l'informazione che identifica uno specifico token (l'identificatore x_1)
- Pattern è la descrizione generale della forma dei valori di una classe di token. Ad esempio, $\underbrace{(x|y)(x|y|0|1)^*}_{\text{espressione regolare (che vedremo)}}$
- Lessema è una stringa istanza di un pattern.
  Nel nostro esempio x_1 è un lessema istanza del pattern $(x|y)(x|y|0|1)^*$

Vedremo che ad ogni "nome" di categoria sintattica è associato un "pattern" che specifica i possibili "valori" che possono essere presi per quel nome, come "lessemi".

<u>Esempio</u>

Dalla stringa C

if (x = = 0) printf ("zero")

un analizzatore lessicale potrebbe produrre la seguente sequenza di token

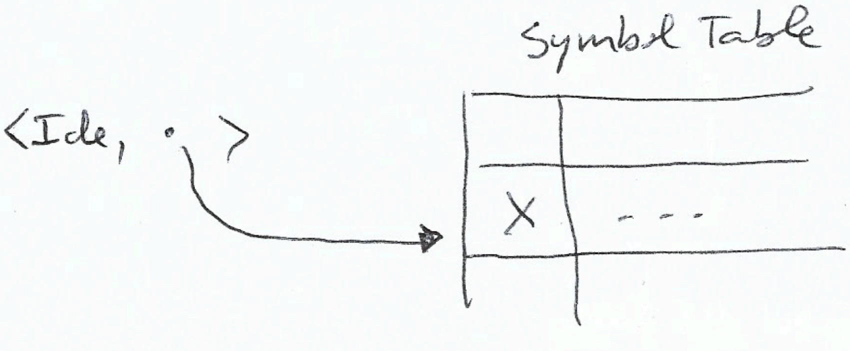
<IF> <(> <Ide, x> <OPREL, ==>

<CONST-NUM, 0> <)> <Ide, printf>

<(> <CONST-STRING, zero> <)>

In realtà, normalmente lo scanner associa agli identificatori un indirizzo nella <u>tabella dei simboli</u>

<Ide, x> è in realtà <Ide, puntatore alla Tabella dei simboli>



Espressioni Regolari

Fissato un alfabeto $A = \{a_1, a_2, \dots, a_n\}$, definiamo le espressioni regolari su A con la seguente BNF

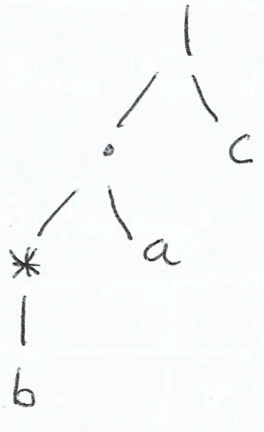
$$r ::= \emptyset \mid \varepsilon \mid \underset{\substack{\uparrow \\ \forall a \in A}}{a} \mid r \cdot r \mid r|r \mid r^*$$

(Questa è una sintassi astratta ambigua; per disambiguare, possiamo usare le parentesi)

Per semplicità, si assume che:

- la concatenazione, la disgiunzione (e anche la ripetizione) associano a sx
- la precedenza tra gli operatori sia: $* > \cdot > |$
- la concatenazione $\cdot$ è di solito omessa

per cui, ad esempio, $b^*a|c$ corrisponde all'albero sintattico



$$\underbrace{(((b)^*)\cdot(a))|(c)}_{\text{secondo la sintassi del libro}}$$

Linguaggio denotato da una espressione regolare

Dato l'alfabeto A, definiamo la funzione

$$\mathcal{L} : \text{Exp-Reg} \longrightarrow \mathcal{P}(A^*)$$

come segue:

simbolo di espressione regolare

$\mathcal{L}[\phi] = \phi$ ← linguaggio vuoto

$\mathcal{L}[\varepsilon] = \{\varepsilon\}$ ← ling. che contiene una sola stringa, quella vuota

$\mathcal{L}[a] = \{a\}$

$\mathcal{L}[r_1 \cdot r_2] = \mathcal{L}[r_1] \cdot \mathcal{L}[r_2]$

$\mathcal{L}[r_1 | r_2] = \mathcal{L}[r_1] \cup \mathcal{L}[r_2]$

$\mathcal{L}[r^*] = (\mathcal{L}[r])^*$

Se usate le parentesi: $\mathcal{L}[(r)] = \mathcal{L}[r]$

Ricordo che:

$L_1 \cdot L_2 = \{ xy \mid x \in L_1, y \in L_2 \}$

$L_1 \cup L_2 = \{ x \mid x \in L_1 \text{ o } x \in L_2 \}$

$L^0 = \{\varepsilon\} \qquad L^{n+1} = L \cdot L^n$

$L^* = \bigcup_{n \geq 0} L^n \qquad L^+ = \bigcup_{n \geq 1} L^n$

Linguaggio Regolare

Def: Un linguaggio $L \subseteq A^*$ è detto regolare sse $\exists$ una espressione regolare r tale che

$$L = \mathcal{L}[r]$$

Prop: Ogni linguaggio finito è regolare

Ad es: $L = \{a, bc\}$ $\quad r = a|bc$

$$\begin{aligned} \mathcal{L}[a|bc] &= \mathcal{L}[a] \cup \mathcal{L}[bc] = \\ &= \{a\} \cup \mathcal{L}[b] \cdot \mathcal{L}[c] = \\ &= \{a\} \cup \{b\} \cdot \{c\} = \\ &= \{a, bc\} = L \end{aligned}$$

(Non serve mai l'operatore $*$ di ripetizione)

Oss: Esistono linguaggi regolari infiniti

$$\begin{aligned} \mathcal{L}[a^* b] &= \mathcal{L}[a^*] \cdot \mathcal{L}[b] = (\mathcal{L}[a])^* \cdot \{b\} = \\ &= \{a\}^* \cdot \{b\} = \bigcup_{n \geq 0} \{a\}^n \cdot \{b\} = \\ &= \{\varepsilon, a, aa, \dots\} \cdot \{b\} = \{a^n b \mid n \geq 0\} \end{aligned}$$

$$\mathcal{L}[a \mid a^* b] = \mathcal{L}[a] \cup \mathcal{L}[a^* b] = \{a\} \cup \{a^n b \mid n \geq 0\}$$

$$\begin{aligned} \mathcal{L}[(a|b) \cdot b^*] &= \mathcal{L}[a|b] \cdot \mathcal{L}[b^*] = (\mathcal{L}[a] \cup \mathcal{L}[b]) \cdot (\mathcal{L}[b])^* \\ &= \{a, b\} \cdot \{b\}^* = \{ab^n \mid n \geq 0\} \cup \{b^n \mid n \geq 1\} \end{aligned}$$

Esempi di Espr. Regolari

$A = \{0, 1\}$

- $0^* 1 0^*$

  $L_1 = \{w \in A^* \mid w$ contiene un solo $1\}$

- $(0|1)^* 1 (0|1)^*$

  $L_2 = \{w \in A^* \mid w$ contiene almeno un $1\}$

- $(0|1)^* 001 (0|1)^*$

  $L_3 = \{w \in A^* \mid w$ contiene 001 come sottostringa$\}$

- $1^* (0 1 1^*)^*$

  $L_4 = \{w \in A^* \mid$ ogni occorrenza di 0 è seguita immediatamente da almeno un $1\}$

- $((0|1)(0|1))^*$

  $L_5 = \{w \in A^* \mid w$ è di lunghezza pari$\}$

- $(0|1)^* 1$

  $L_6 = \{w \in A^* \mid w$ termina con $1\}$

Altri Operatori Ausiliari

Ripetizione positiva: r^+ $(rr^* \text{ o } r^*r)$

Possibilità: $r?$ $(r|\varepsilon)$

Elenco: $[a_1, \dots, a_n]$ per $a_1, \dots, a_n \in A$
(sta per $a_1|a_2|\dots|a_n$)

se gli a_i sono ordinati in modo che $a_i < a_{i+1}$
allora
$[a_1 - a_n]$ (sta per $a_1|a_2|\dots|a_n$)

Esempio

- Numeri decimali senza segno

$$[0-9]^+(\varepsilon \mid .[0-9]^+)$$

dove l'alfabeto è $\{0, 1, \dots, 9, .\}$

Alternativamente: $[0-9]^+(.[0-9]^+)?$

Definizioni Regolari

Una definizione regolare su alfabeto A è costituita da una lista di definizioni

$$
\begin{aligned}
d_1 &:= r_1 \\
d_2 &:= r_2 \\
&\vdots \\
d_k &:= r_k
\end{aligned}
$$

dove i vari d_i sono simboli "nuovi" e ogni r_i è una espressione regolare sull'alfabeto esteso $A \uplus \{d_1, \dots, d_k\}$

Esempi

(1) Numeri decimali con segno

numconsegno := segno cifre (. cifre)?

segno := - | +

cifre := cifra$^+$

cifra := [0-9]

(2) Identificatori

identifier := letter (letter | digit)*

letter := [a-zA-Z]

digit := [0-9]

Esercizio: Provate a definire un tipo particolare di identificatore che deve soddisfare questi requisiti:

- iniziare con una lettera maiuscola
- contenere almeno una cifra
- terminare col simbolo "!"

Equivalenza Tra Espr. Regolari

Def: Due espressioni regolari r e s sono equivalenti sse $\mathcal{L}[r] = \mathcal{L}[s]$ (cioè denotano lo stesso linguaggio) e lo denotiamo con $r \simeq s$

Esistono molte leggi per $\simeq$. Alcune sono le seguenti:

$r \mid s \simeq s \mid r$	$\mid$ è commutativa
$r \mid (s \mid t) \simeq (r \mid s) \mid t$	$\mid$ è associativa
$r \mid r \simeq r$	$\mid$ è idempotente
$r \cdot (s \cdot t) \simeq (r \cdot s) \cdot t$	$\cdot$ è associativa
$\varepsilon \cdot r \simeq r \simeq r \cdot \varepsilon$	ε è l'elemento neutro per $\cdot$
$(r^*)^* \simeq r^*$	$*$ è idempotente
$r(s \mid t) \simeq rs \mid rt$	$\cdot$ distribuisce a sx sul $\mid$
$(r \mid s)t \simeq rt \mid st$	$\cdot$ distribuisce a dx sul $\mid$

$\phi^* \simeq \varepsilon$

$r \mid \phi \simeq r$

$r \cdot \phi \simeq \phi \simeq \phi \cdot r$

$(\varepsilon \mid r)^* \simeq r^*$

$(r^* s^*)^* \simeq (r \mid s)^*$

$\vdots$

Dimostrare queste leggi non è sempre facile, ma in alcuni casi lo è.

$$\mathcal{L}[r|s] = \mathcal{L}[r] \cup \mathcal{L}[s] = \mathcal{L}[s] \cup \mathcal{L}[r] = \mathcal{L}[s|r]$$

$$\mathcal{L}[r|r] = \mathcal{L}[r] \cup \mathcal{L}[r] = \mathcal{L}[r]$$

$$\mathcal{L}[r\varepsilon] = \mathcal{L}[r] \cdot \{\varepsilon\} = \mathcal{L}[r]$$

$$\mathcal{L}[\emptyset^*] = (\mathcal{L}[\emptyset])^* = \emptyset^* = \emptyset^0 \cup \emptyset^1 \cup \emptyset^2 \ldots$$
$$= \{\varepsilon\} \cup \emptyset \cup \emptyset \ldots = \{\varepsilon\} = \mathcal{L}[\varepsilon]$$

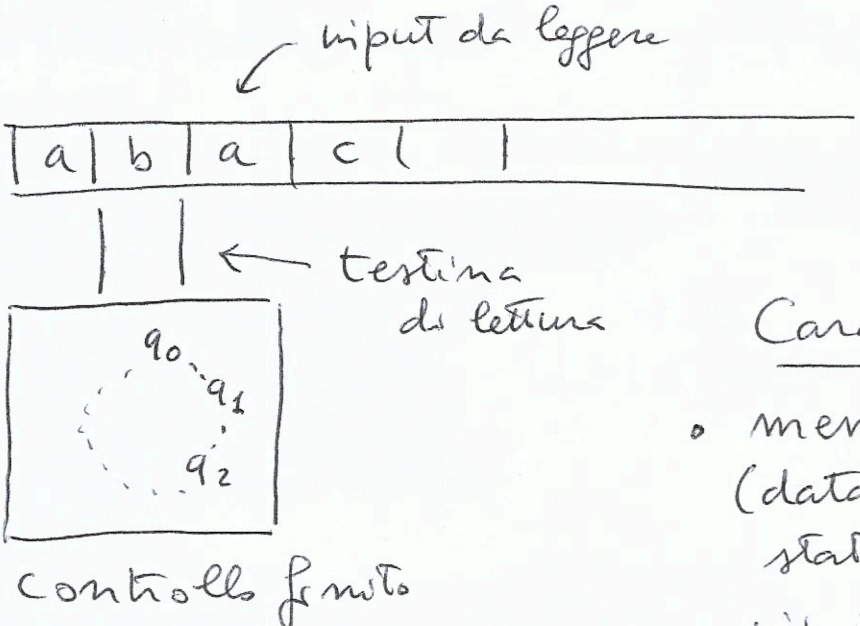
$$\mathcal{L}[r \cdot \emptyset] = \mathcal{L}[r] \cdot \mathcal{L}[\emptyset] = \mathcal{L}[r] \cdot \emptyset = \emptyset = \mathcal{L}[\emptyset]$$

Le espressioni regolari servono per specificare il pattern di una categoria sintattica, ovvero la forma dei possibili lessemi.

Ma come <u>riconoscere</u> se una certa sequenza in ingresso è un lessema per una certa categoria sintattica?

$\Rightarrow$ Automi (a stati finiti)

Automi (a stati) Finiti



Caratteristiche:

- memoria finita (data dal numero degli stati $q_0, \dots q_n$)
- input: stringa da riconoscere
- output: 1 bit (Si/No)

Descrizione iniziale

- testina di lettura posizionata sul primo carattere dell'input
- controllo su stato iniziale q_0

Funzionamento: Ripeti

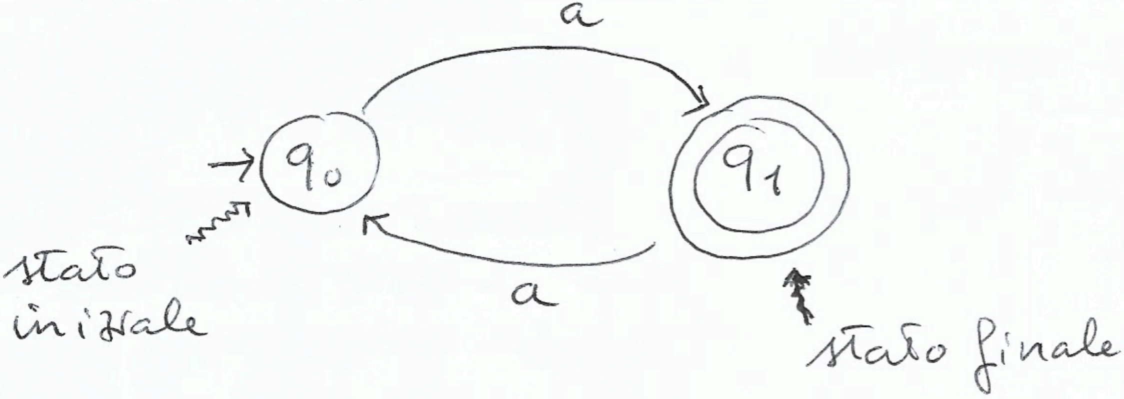
- leggi il carattere in input; in base allo stato in cui si trova decide:
  - di cambiare di stato
  - e di spostare la testina sull'input success

fino a che

- ha finito di leggere l'input (e riconosce la stringa se ha raggiunto uno stato finale)
- oppure si è bloccato prima perché per la coppia (stato corrente, input attuale) non era specificato uno stato successivo

Diagrammi di Transizione

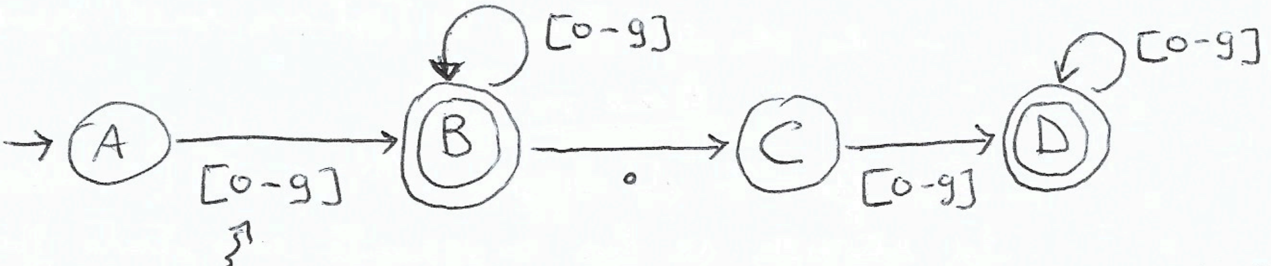
Il funzionamento di un automa finito è ben rappresentato attraverso un diagramma di transizione



Riconoscere una stringa w significa trovare un cammino etichettato w sul grafo a partire dallo stato iniziale che finisce su uno stato finale

$$L = \{ a^{2n+1} \mid n \geq 0 \} = \{ a^n \mid n \text{ è dispari} \}$$
$$= \mathcal{L}[a(aa)^*]$$

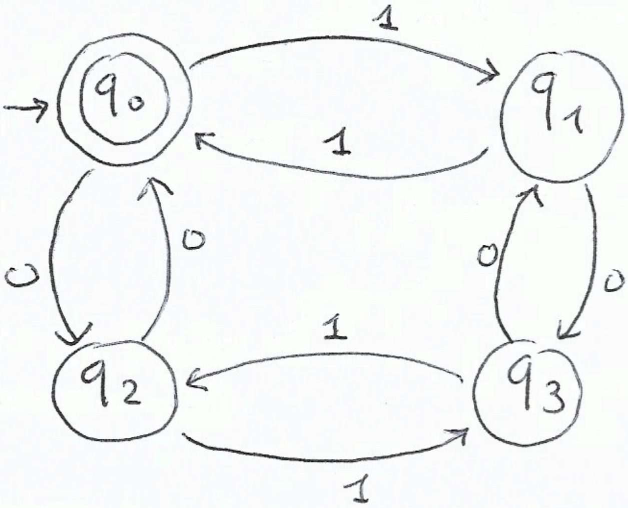
<u>N.B.</u> Se in input ho w = a b, l'automa si blocca in q_1, senza essere stata completata la lettura dell'input w



sta a riassumere 10 diverse transizioni, una per ciascun simbolo 0,..,9

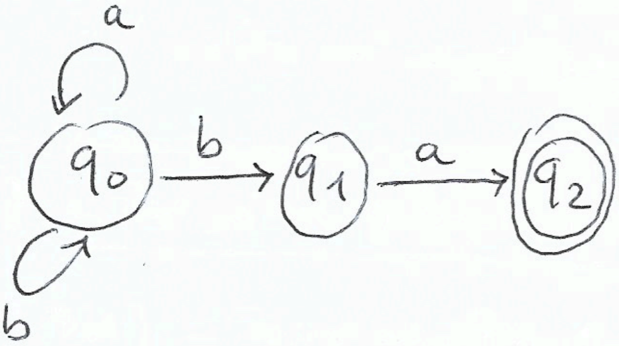
diagramma di transizione per
$[0-9]^+(\varepsilon \mid . [0-9]^+)$

Esempi



$L = \{ w \in \{0,1\}^* \mid$ in w il numero di 0 e 1 è sempre pari $\}$

N.B. Deterministico: per ogni coppia (q, s), con $q \in \{q_0, q_1, q_2, q_3\}$ e $s \in \{0,1\}$, esiste una e una sola mossa possibile

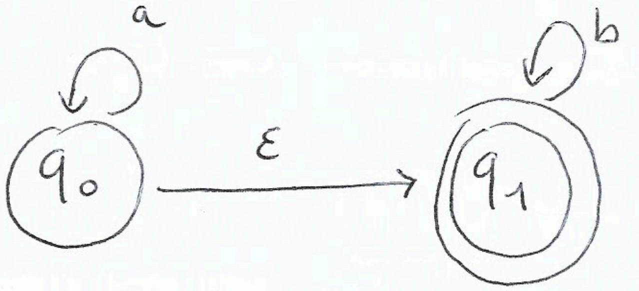


$L = \mathcal{L}[(a|b)^* ba]$

N.B: • $ba \in L[M]$ perché esiste un cammino da q_0 a q_2 etichettato ba

• NONDETERMINISTICO

- (q_0, b) offre 2 mosse o su q_0 o su q_1
- (q_1, b) non offre mosse
- $(q_2, a/b)$ non offre mosse

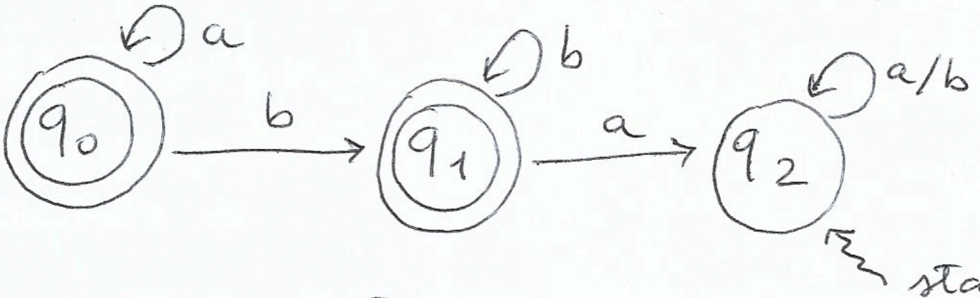


$L[M] = \mathcal{L}[a^* b^*]$

transizioni ε ⟶

- nondeterministico: in q_0 può non spostare la testina di lettura (ovvero non leggere l'input) e spostarsi in q_1

Se vogliamo un automa deterministico per $\mathcal{L}[a^*b^*]$, possiamo definire



stato pozzo (di errore)

- non ci sono transizioni etichettate ε
- da ogni stato per ognuno dei due simboli (a e b), esce una e una sola transizione

↳ questo automa è deterministico

Automi Finiti Nondeterministici (NFA)

Def: Un automa finito nondeterministico (NFA) è una quintupla $(\Sigma, Q, \delta, q_0, F)$ dove

- Σ è un alfabeto finito di simboli in input
- Q è un insieme finito di stati
- $q_0 \in Q$ è lo stato iniziale
- $F \subseteq Q$ è l'insieme degli stati finali
- δ è la funzione di transizione con tipo

$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \longrightarrow \underbrace{\mathcal{P}(Q)}_{\text{insieme delle parti di } Q}$$

$$\left(\delta(q, \sigma) = Q' \subseteq Q \right)$$

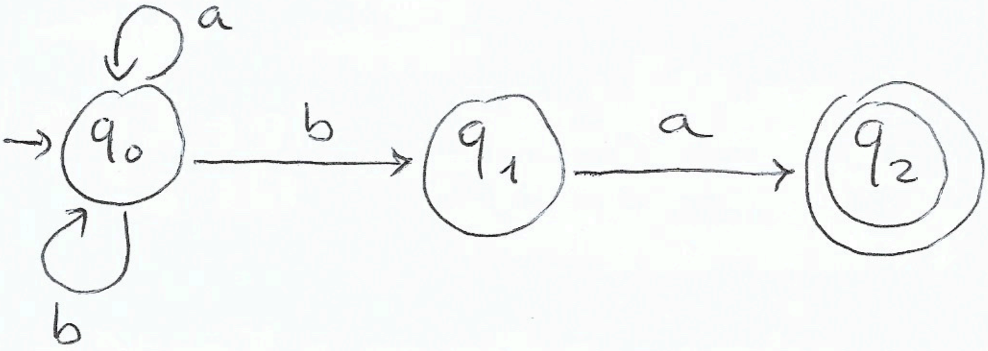
Esempio

$\Sigma = \{a, b\}$ $Q = \{q_0, q_1, q_2\}$ q_0 iniziale

$F = \{q_2\}$

δ	a	b	ε
q_0	$\{q_0\}$	$\{q_0, q_1\}$	$\emptyset$
q_1	$\{q_2\}$	$\emptyset$	$\emptyset$
q_2	$\emptyset$	$\emptyset$	$\emptyset$

Come rappresentare questo NFA con un diagramma di transizioni?



Esercizio

- Disegnate il diagramma di transizione per $M = (\Sigma, Q, \delta, q_0, F)$ dove

$\Sigma = \{0, 1\}$ $Q = \{q_0, q_1, q_2\}$ q_0 iniziale $F = \{q_1\}$

δ	0	1	ε
q_0	$\emptyset$	$\{q_0, q_2\}$	$\{q_1\}$
q_1	$\{q_1\}$	$\emptyset$	$\{q_0\}$
q_2	$\{q_2, q_0\}$	$\emptyset$	$\emptyset$

- Ricavare dai diagrammi di transizione a pg 14 la definizione di NFA come quintuple $(\Sigma, Q, \delta, q_0, F)$

Linguaggio Riconosciuto/Accettato

<u>Def</u> (informale) Un NFA $N = (\Sigma, Q, \delta, q_0, F)$ accetta $w = a_1 \ldots a_n$ sse nel diagramma di transizione esiste un cammino da q_0 ad uno stato in F nel quale la stringa che si ottiene concatenando le etichette degli archi percorsi è esattamente w.

Più formalmente:

- Descrizione istantanea: $(\underset{\substack{\uparrow \\ \text{stato corrente}}}{q}, \underset{\substack{\nwarrow \\ \text{input da leggere}}}{w})$

- Mossa

$$\frac{q' \in \delta(q, \sigma)}{(q, \sigma w) \vdash_N (q', w)} \qquad \begin{array}{l} \sigma \in \Sigma \cup \{\varepsilon\} \\ w \in \Sigma^* \end{array}$$

- Cammino (chiusura riflessiva e transitiva di $\vdash_N$)

$$\frac{}{(q, w) \vdash_N^* (q, w)} \qquad \frac{(q, w) \vdash_N^* (q', w') \quad (q', w') \vdash_N (q'', w'')}{(q, w) \vdash_N^* (q'', w'')}$$

- Accettazione / Riconoscimento

$$w \in L[N] \text{ sse } \exists q \in F.\ (q_0, w) \vdash_N^* (q, \varepsilon)$$

- Il linguaggio accettato da N, indicato con $L[N]$, è

$$L[N] = \{ w \in \Sigma^* \mid \exists q \in F.\ (q_0, w) \vdash_N^* (q, \varepsilon) \}$$

- Due NFA N_1 e N_2 si dicono <u>equivalenti</u> sse accettano lo stesso ling., cioè se $L[N_1] = L[N_2]$

Esempi / Esercizi

Costruire un NFA per i linguaggi

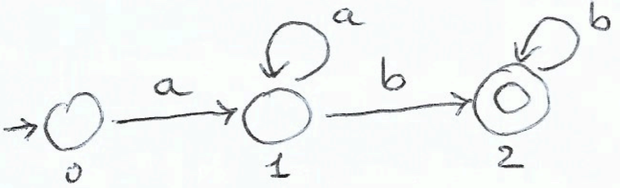
$L_1 = \{ a^n b^m \mid n, m \geq 1 \}$

$L_2 = \{ a^{3K+2} \mid K \geq 0 \}$

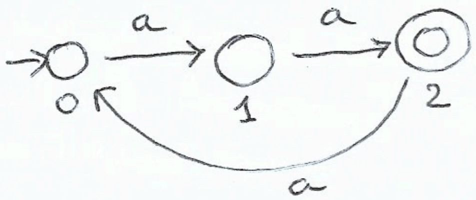
$L_3 = \{ w \in (a|b)^* \mid$ w contiene la sottostringa aba$\}$

$L_4 = \{ w \in (a|b)^* \mid$ w contiene esattamente 3 "a"$\}$

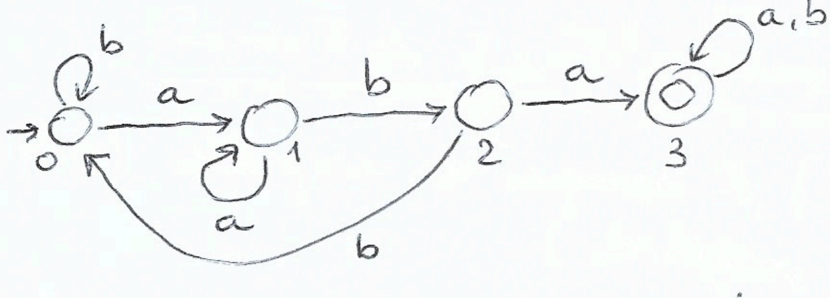
Soluzioni



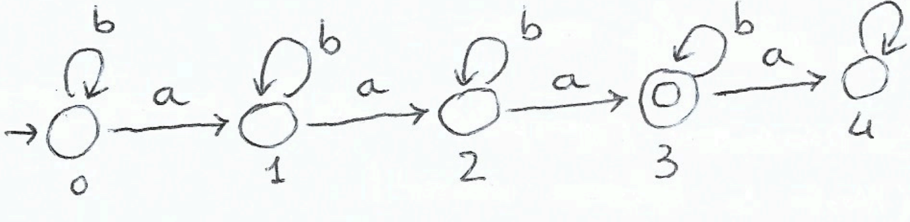
$L[N_1] = L_1$



$L[N_2] = L_2$

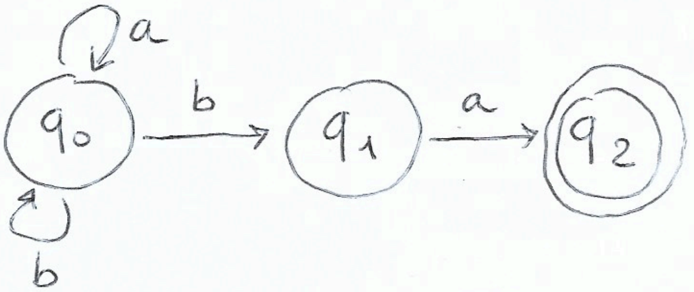


$L[N_3] = L_3$

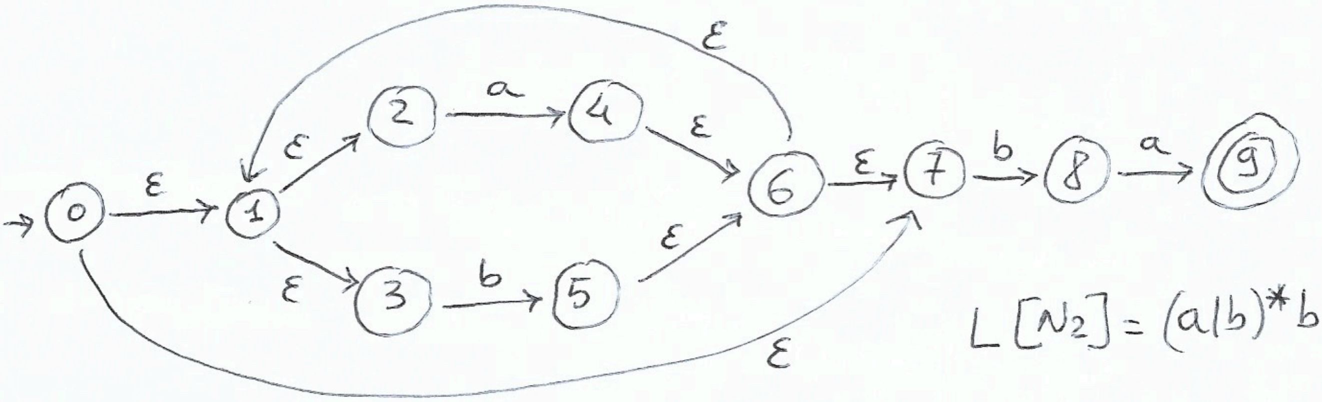


$L[N_4] = L_4$

N.B. Per un linguaggio L, se esiste un NFA N tale che $L = L[N]$, allora $\exists$ infiniti $N_1, N_2, \ldots N_K, \ldots$ tali che $L = L[N_i]$ per $i = 1, 2, \ldots$



$L[N_1] = \mathcal{L}[(a|b)^*ba]$



$L[N_2] = (a|b)^*ba$

e ne vedremo altri sempre per questo linguaggio!

NFA: - sono "comodi": facile costruirli, ma
- inefficienti: accettare w significa cercare un cammino su un grafo "nondeterministico" ⇒ tante potenziali strade alternative (possibili failure ⇒ backtracking)

DFA: Deterministic Finite Automata

- $\delta(q, \sigma)$ è sempre un singoletto (solo una mossa possibile)
- non ci sono le mosse ε

⇒ • scansione completa dell'input garantita (non si blocca mai)
• in un tempo $O(n)$ - dove n è la lunghezza di w - sappiamo se w è accettata o meno
• ma più difficili da definire

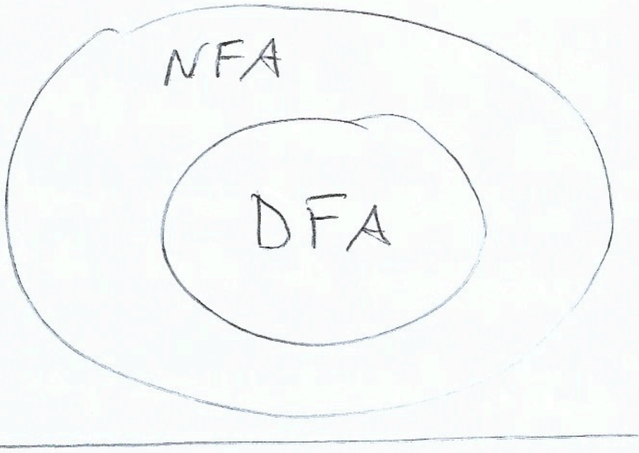
Automi Finiti Deterministici

Def Un automa finito deterministico (DFA) è una quintupla $(\Sigma, Q, \delta, q_0, F)$, dove Σ, Q, q_0 e F sono definiti come per un NFA, mentre la funzione di transizione δ ha tipo $\delta : Q \times \Sigma \to Q$

$$(\delta(q,\sigma) = q')$$

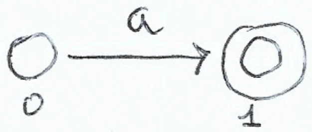
Osservazione: Un DFA è un particolare tipo di NFA tale che:

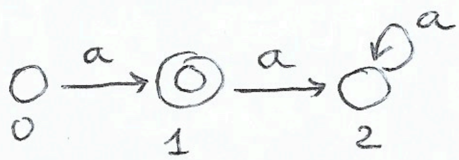
- $\forall q \in Q \quad \delta(q,\varepsilon) = \emptyset$ (non ci sono transizioni ε)
- $\forall \sigma \in \Sigma, \forall q \in Q \; \exists q' \in Q . \; \delta(q,\sigma) = \{q'\}$
  (l'insieme delle mosse possibili è sempre un singoletto)



NFA o DFA ?

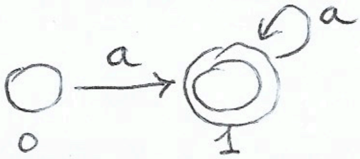
$\Sigma = \{a\}$

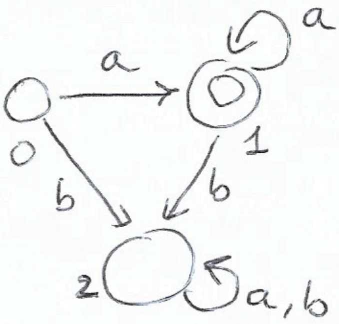




NFA o DFA ?

$\Sigma = \{a, b\}$





Vogliamo ora dimostrare che i DFA sono tanto espressivi quanto gli NFA, sebbene siano un sottoinsieme proprio degli NFA.

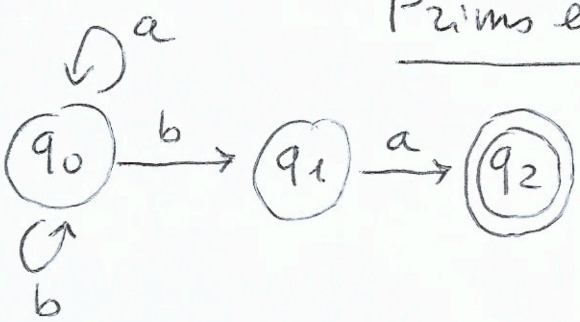
Prop: Per ogni NFA, è possibile costruire un DFA ad esso equivalente!

Come fare? Idea: "seguire" contemporaneamente tutti i possibili cammini alternativi dell'NFA
$\Rightarrow$ gli stati del DFA che andiamo a costruire sono costituiti da insiemi di stati dell'NFA

Tecnica detta "Costruzione per sottoinsiemi" (Subset Construction)

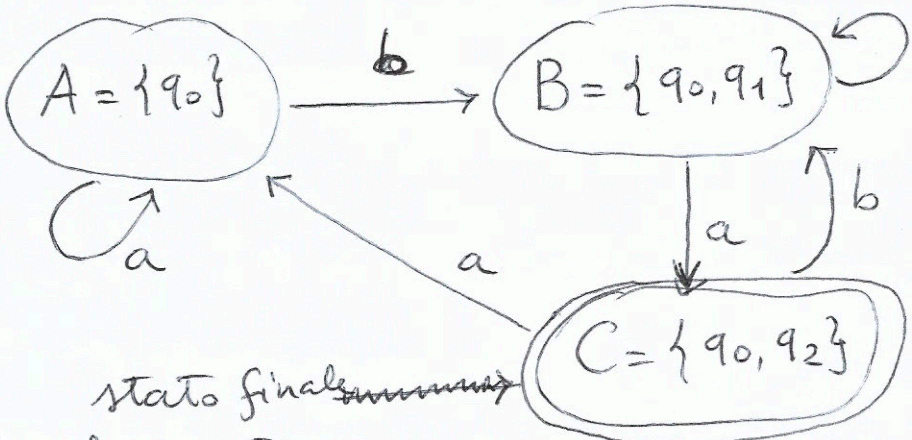
Prima di formalizzarla, vediamo un paio d'esempi.

Primo esempio semplice

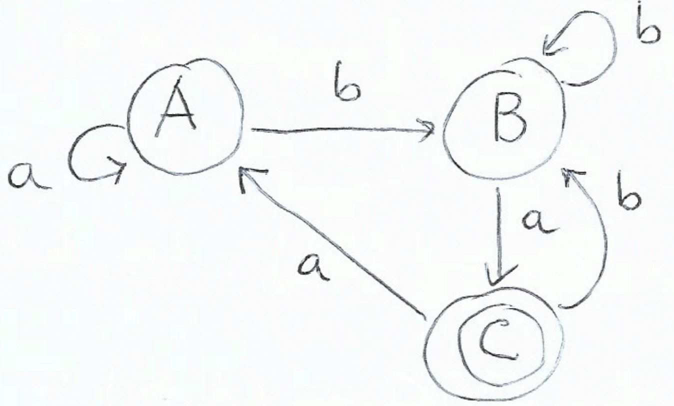


NFA che vogliamo trasformare in un equivalente DFA

Stato iniziale del DFA: $\{q_0\} = A$



stato finale perché contiene $q_2 \in F$

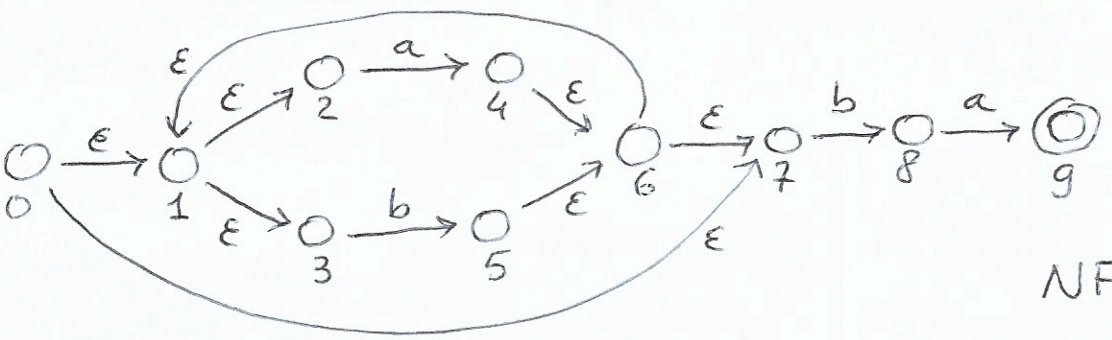


questo è un DFA del tutto equivalente

- Passando dallo stato $A = \{q_0\}$ allo stato $B = \{q_0, q_1\}$, ho seguito contemporaneamente tutte e due le transizioni etichettate b
- Passando dallo stato $B = \{q_0, q_1\}$ a $C = \{q_0, q_2\}$, ho seguito sia la transizione $q_0 \xrightarrow{a} q_0$, sia la transizione $q_1 \xrightarrow{a} q_2$

$$\text{mossa}(B, a) = \bigcup_{q \in B} \delta(q, a) = \delta(q_0, a) \cup \delta(q_1, a)$$
$$= \{q_0, q_2\} = C$$

Secondo Esempio



NFA

Lo stato iniziale del DFA è dato da 0, più tutti gli stati che possono raggiungere da 0 con mosse ε

$$\varepsilon\text{-closure}(0) = \{0, 1, 2, 3, 7\} \quad \text{Ⓐ}$$

Per calcolare lo stato che raggiungo da A leggendo "a", devo vedere quali stati in A possono fare "a" e poi farne la ε-closure

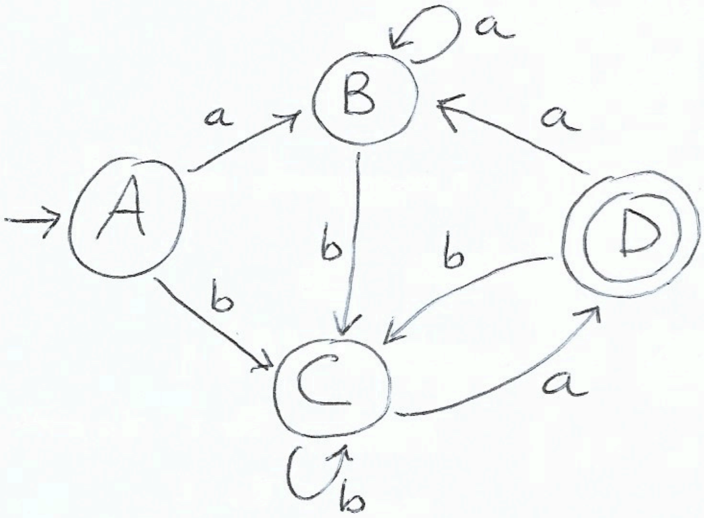
$$\Delta(A, a) = \varepsilon\text{-closure}(\text{mossa}(A, a)) = $$
$$= \varepsilon\text{-closure}\Big(\bigcup_{q \in A} \delta(q, a)\Big) =$$
$$= \varepsilon\text{-closure}(\{4\}) = \{\underline{4}, 6, 7, 1, 2, 3\} = B$$

$$\Delta(A, b) = \varepsilon\text{-closure}(\text{mossa}(A, b)) =$$
$$= \varepsilon\text{-closure}\Big(\bigcup_{q \in A} \delta(q, b)\Big) =$$
$$= \varepsilon\text{-closure}(\{5, 8\}) = \{1, 2, 3, \underline{5}, 6, 7, \underline{8}\} = C$$

$$\Delta(B, a) = \varepsilon\text{-closure}(\{4\}) = B$$
$$\Delta(B, b) = \varepsilon\text{-closure}(\{5, 8\}) = C$$
$$\Delta(C, a) = \varepsilon\text{-closure}(\{4, 9\}) = \{\underline{4}, \overset{\text{finale}}{\underline{9}}, 1, 2, 3, 6, 7\} = D$$
$$\Delta(C, b) = \varepsilon\text{-closure}(\{5, 8\}) = C$$

$$\Delta(D, a) = \varepsilon\text{-closure}(\{4\}) = B$$

$$\Delta(D, b) = \varepsilon\text{-closure}(\{5, 8\}) = C$$



DFA ottenuto
per costruzione
per sottoinsiemi

ε-closure e mosse

Sia q uno stato di un NFA. La ε-closure di q è l'insieme degli stati raggiungibili da q solo con mosse ε

$$\frac{}{\{q\} \subseteq \varepsilon\text{-closure}(q)} \qquad \frac{p \in \varepsilon\text{-closure}(q)}{\delta(p, \varepsilon) \subseteq \varepsilon\text{-closure}(q)}$$

Sia P un insieme di stati di un NFA.

$$\varepsilon\text{-closure}(P) = \bigcup_{p \in P} \varepsilon\text{-closure}(p)$$

$$\text{mossa} : \mathcal{P}(Q) \times \Sigma \to \mathcal{P}(Q)$$

$$\text{mossa}(P, a) = \bigcup_{p \in P} \delta(p, a)$$

$$\Delta(A, b) = \varepsilon\text{-closure}(\text{mossa}(A, b))$$

↑ funzione di transizione del DFA

Algoritmo per calcolare la ε-closure (P)

```
Inizializzare $T = P$;
Inizializzare $\varepsilon$-closure$(P) = P$;
while $T \neq \emptyset$ do {
        "scegli un $r \in T$ e rimuovilo da $T$"
        for each $s \in \delta(r, \varepsilon)$ do
                if $s \notin \varepsilon$-closure$(P)$ {
                        add $s$ to $\varepsilon$-closure$(P)$;
                        add $s$ to $T$;
                }
}
```

Oss: Usando ε-closure, si può definire il ling. riconosciuto da un NFA in modo elegante:

$$\hat{\delta} : Q \times \Sigma^* \to \mathcal{P}(Q)$$

$$\hat{\delta}(q, \varepsilon) = \varepsilon\text{-closure}(q)$$

$$\hat{\delta}(q, xa) = \varepsilon\text{-closure}(P)$$

dove $P = \{ p \in Q \mid \exists r \in \hat{\delta}(q, x) \text{ e } p \in \delta(r, a) \}$

$$\boxed{w \in L[N] \text{ sse } \exists p \in F \text{ tale che } p \in \hat{\delta}(q_0, w)}$$

Costruzione dei Sottoinsiemi

Dato un NFA $N=(\Sigma, Q, \delta, q_0, F)$:

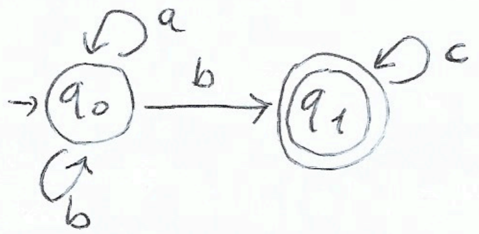
```
• Inizializza $S = \varepsilon\text{-closure}(q_0)$;   // S stato iniziale del DFA
• Inizializza $T = \{S\}$;   // T è l'insieme degli stati del DFA
                             // S non è marcato all'inizio

• Finché c'è un $P \in T$ non marcato {
      - marca P;
      - for each $a \in \Sigma$ {
            - $R = \varepsilon\text{-closure}(\text{mossa}(P, a))$;
            - if $R \notin T$ {
                  add R to T;   // R non ha marca
              }
            - definisci $\Delta(P, a) = R$;
        }
  }
```

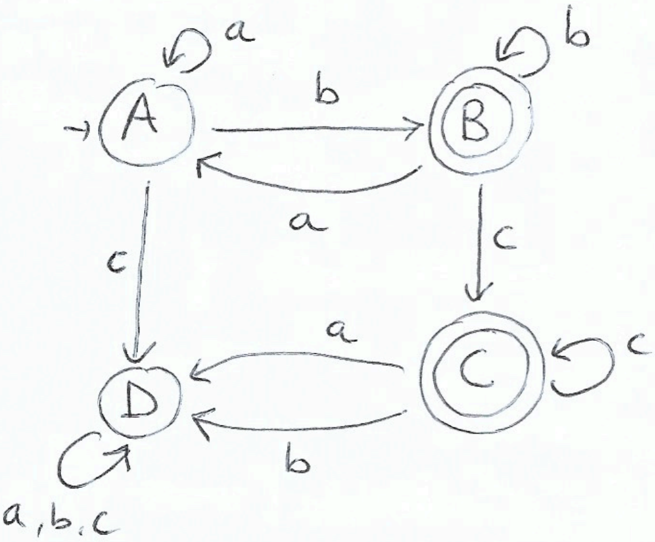
Definiamo il DFA $M_N = (\Sigma, T, \Delta, \varepsilon\text{-closure}(q_0), \mathcal{F})$ dove $R \in \mathcal{F}$ sse $\exists q \in R$ con $q \in F$.

- Nel caso pessimo $T = \mathcal{P}(Q)$, cioè il DFA M_N costruito a partire dall'NFA N, ha un numero di stati pari a 2^n, dove $n = |Q|$.
  ↑ esponenziale
- Di solito T è molto più piccolo di $\mathcal{P}(Q)$

Esempio di caso pessimo



NFA N con 2 stati



$A = \{q_0\}$

$B = \{q_0, q_1\}$

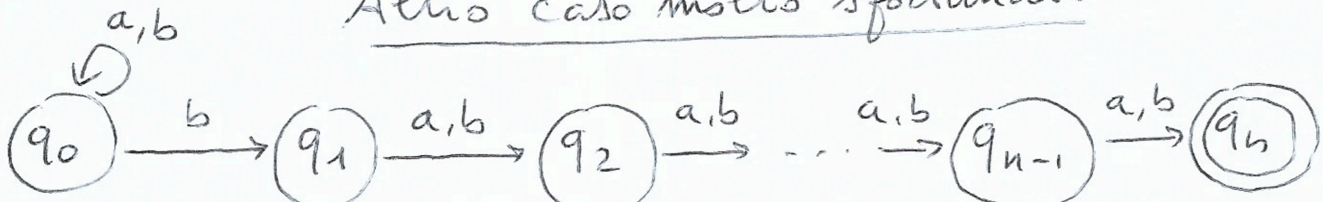
$C = \{q_1\}$

$D = \emptyset$

DFA M_N con $2^2 = 4$ stati

Il DFA M_N può essere "esponenzialmente" più grande dell'NFA N

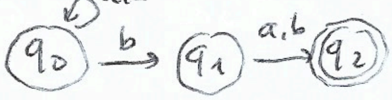
Altro caso molto sfortunato



$L[N] = \{w \in (a|b)^* \mid$ il simbolo n-esimo da dx è b$\}$

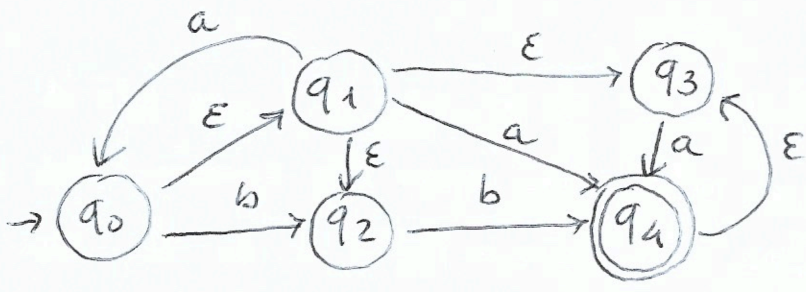
$$\begin{tikzcd}
\{q_0\} \arrow[loop above, "a"] \arrow[r, "b"] & \{q_0, q_1\} \arrow[r, "b"] \arrow[dd, "a"] & \{q_0, q_1, q_2\} \arrow[r, "b"] \arrow[d, "a"] & \{q_0, q_1, q_2, q_3\} \arrow[r, "b"] \arrow[d, "a"] & \dots \\
& & \{q_0, q_2, q_3\} & \{q_0, q_2, q_3, q_4\} & \\
& \{q_0, q_2\} \arrow[r, "b"] \arrow[d, "a"] & \{q_0, q_1, q_3\} \arrow[r, "b"] & \dots & \\
& \{q_0, q_3\} \arrow[r, "b"] & \{q_0, q_1, q_4\} \arrow[r, "b"] & \dots &
\end{tikzcd}$$

Anche in questo caso, gli stati del DFA sono 2^n mentre gli stati dell'NFA sono $n+1$

Provate a disegnare il DFA per 

vedete che avrà 2^2 stati

Esempio Corposo



NFA N

$A = \varepsilon\text{-closure}(q_0) = \{q_0, q_1, q_2, q_3\}$

$\Delta(A, a) = \varepsilon\text{-closure}(\text{mossa}(A, a)) = \varepsilon\text{-closure}(\{q_0, q_4\}) =$

$= \{q_0, q_1, q_2, q_3, q_4\} = B$

$\Delta(A, b) = \varepsilon\text{-closure}(\text{mossa}(A, b)) = \varepsilon\text{-closure}(\{q_2, q_4\}) =$

$= \{q_2, q_3, q_4\} = C$

$\Delta(B, a) = \varepsilon\text{-closure}(\text{mossa}(B, a)) = \varepsilon\text{-closure}(\{q_0, q_4\}) = B$

$\Delta(B, b) = \varepsilon\text{-closure}(\text{mossa}(B, b)) = \varepsilon\text{-closure}(\{q_2, q_4\}) = C$

$\Delta(C, a) = \varepsilon\text{-closure}(\text{mossa}(C, a)) = \varepsilon\text{-closure}(\{q_4\}) =$

$= \{q_3, q_4\} = D$

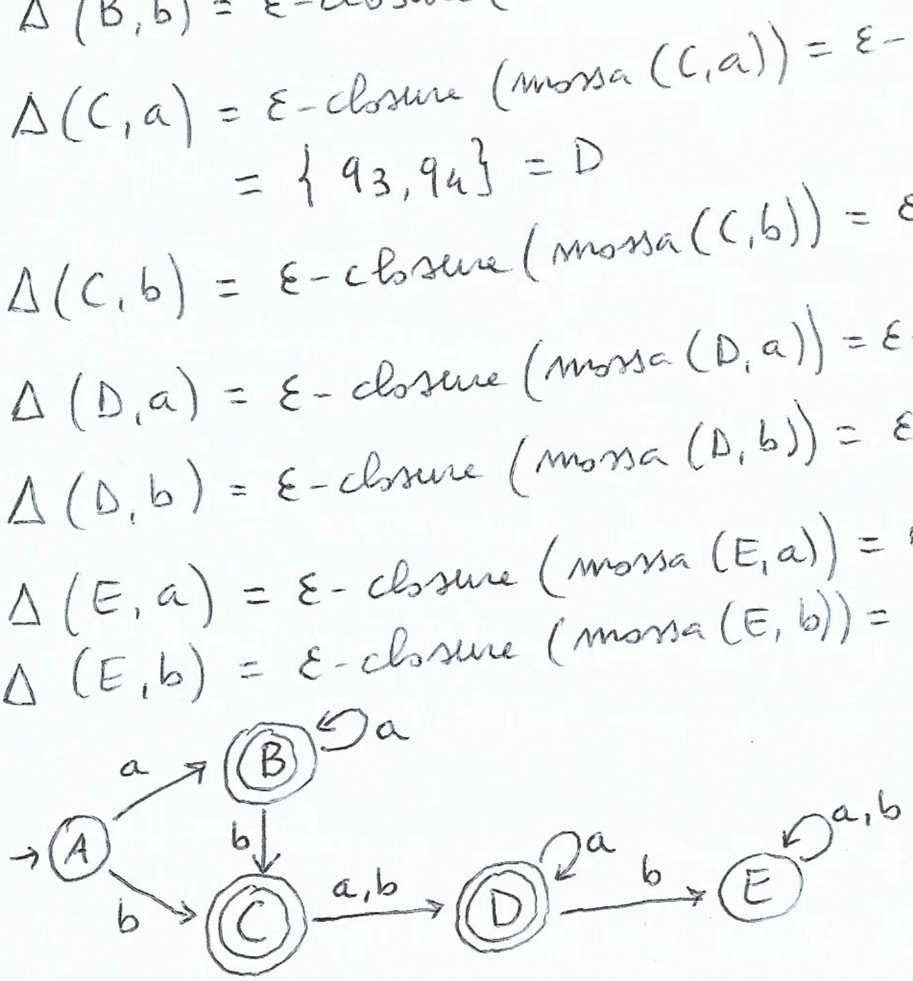
$\Delta(C, b) = \varepsilon\text{-closure}(\text{mossa}(C, b)) = \varepsilon\text{-closure}(\{q_4\}) = D$

$\Delta(D, a) = \varepsilon\text{-closure}(\text{mossa}(D, a)) = \varepsilon\text{-closure}(\{q_4\}) = D$

$\Delta(D, b) = \varepsilon\text{-closure}(\text{mossa}(D, b)) = \varepsilon\text{-closure}(\emptyset) = \emptyset = E$

$\Delta(E, a) = \varepsilon\text{-closure}(\text{mossa}(E, a)) = \varepsilon\text{-closure}(\emptyset) = \emptyset \equiv E$

$\Delta(E, b) = \varepsilon\text{-closure}(\text{mossa}(E, b)) = \varepsilon\text{-closure}(\emptyset) = \emptyset = E$



DFA M_N

Equivalenza NFA - DFA

Teorema Sia $N = (\Sigma, Q, \delta, q_0, F)$ un NFA e sia M_N l'automa ottenuto con la costruzione per sottoinsiemi. Allora M_N è un DFA e si ha che

$$L[N] = L[M_N]$$

Corollario La classe dei linguaggi riconosciuti dagli NFA coincide con la classe dei ling. riconosciuti dai DFA

Dimostrazione del Teorema:

Sia $N = (\Sigma, Q, \delta, q_0, F)$ un NFA e sia $M_N = (\Sigma, T, \Delta, \varepsilon\text{-closure}(q_0), \mathcal{F})$ l'automa ottenuto con l'algoritmo.

(1) M_N è deterministico! Infatti $\Delta(A, a)$ è definita per ogni coppia (A, a) con $A \in T$ e $a \in \Sigma$ in modo univoco, e il risultato di $\Delta(A, a)$ è un elemento di T.

(2) Quindi dobbiamo "solo" dimostrare che $L[N] = L[M_N]$

Oss: Per un DFA, $\varepsilon\text{-closure}(R) = R$ con $R \in T$, perché non ci sono mosse ε

Notazione Chiamiamo $i_M = \varepsilon\text{-closure}(q_0)$ lo stato iniziale di M_N

Vogliamo dimostrare che $\forall w \in \Sigma^*$

$$\hat{\delta}(q_0, w) = \hat{\Delta}(i_M, w)$$

(Per def. $\hat{\delta}$ vedi pg. 24)

per induzione sulla lunghezza di w.

Caso base : $|w| = 0$ cioè $w = \varepsilon$

$$\hat{\delta}(q_0, \varepsilon) = \varepsilon\text{-closure}(q_0)$$

$$\hat{\Delta}(i_M, \varepsilon) = \varepsilon\text{-closure}(i_M) = i_M = \varepsilon\text{-closure}(q_0)$$

OK

Passo induttivo $w = xa$ con $a \in \Sigma$ e $x \in \Sigma^*$

Per ipotesi induttiva, sappiamo che

$$\hat{\delta}(q_0, x) = \hat{\Delta}(i_M, x) = \{p_1, \dots, p_k\}$$

Per definizione di $\hat{\delta}$,

$$\hat{\delta}(q_0, xa) = \cancel{\Delta}\ \varepsilon\text{-closure}\left(\bigcup_{i=1}^{k} \delta(p_i, a)\right)$$

Similmente

$$\hat{\Delta}(i_M, xa) = \Delta(\{p_1 \dots p_k\}, a)$$

In base all'algoritmo, la definizione di Δ ci dice che

$$\Delta(\{p_1 \dots p_k\}, a) = \varepsilon\text{-closure}(\text{mossa}(\{p_1 \dots p_k\}, a)) =$$

$$= \varepsilon\text{-closure}\left(\bigcup_{i=1}^{k} \delta(p_i, a)\right)$$

$$= \hat{\delta}(q_0, xa)$$

OK

Infine, abbiamo che

$w \in L[N]$ sse $\exists p \in \hat{\delta}(q_0, w)$ con $p \in F$

sse $\exists p \in \hat{\Delta}(i_M, w)$ con $p \in F$

sse $\hat{\Delta}(i_M, w) \in \tilde{F}$

sse $w \in L[M_N]$ $\qquad \forall w \in \Sigma^*$

Quindi $L[N] = L[M_N]$ c.v.d.
