# AUTOMI E LINGUAGGI

## • DFA : Deterministic Finite Automaton

E' una macchina astratta $M = (Q, \Sigma, \delta, q_0, F)$, con:

1) $Q$ : insieme degli stati interni;
2) $\Sigma$ : alfabeto di simboli che l'automa può prendere in input;
3) $\delta : Q \times \Sigma \longmapsto Q$ : funzione di transizione;
   $\forall (q, \sigma) \in Q \times \Sigma,\ \exists q' \in Q$ tale che: $\delta(q, \sigma) = q'$;
4) $q_0 \in Q$ : "start state";
5) $F \subseteq Q$ : insieme degli stati d'accettazione.

*** NOTA** : Ogni automa riconosce (accetta) 1 UNICO LINGUAGGIO !

• ACCETTAZIONE DI UN AUTOMA:

$M = (Q, \Sigma, \delta, q_0, F)$ accetta la stringa $w \in \Sigma^*$ se ESISTE una SEQUENZA DI STATI $r_0, r_1, \dots, r_m \in Q$, tale che:

(1) $r_0 = q_0$;

(2) $\forall i = 0, 1, \dots m$ : $\delta(r_i, w_{i+1}) = r_{i+1}$; (transizioni lecite)

(3) $r_m \in F$.

## • LINGUAGGIO REGOLARE :

Un linguaggio $A$ è detto "REGOLARE", se esiste un DFA che lo riconosce:
se $\exists M$, tale che $\mathcal{L}(M) = A$.

Operazioni su Regular Languages:

• UNIONE : $A \cup B = \{w \mid w \in A \vee w \in B\}$

• INTERSEZIONE : $A \cap B = \{w \mid w \in A \wedge w \in B\}$

• CONCATENAZIONE : $A \circ B \{w = xy \mid x \in A \wedge y \in B\}$

• KLEENE'S STAR : $A^* = \{x_1 \dots x_k \mid k \geq 0,\ x_i \in A, \forall i = 1, \dots, k\}$
→ insieme di tutte le possibili concatenazioni ($\varepsilon \in A^*$ SEMPRE!)

(1) A, B regolari $\Rightarrow$ $A \cup B$ regolare :

$M_1 = (Q_1, \Sigma_1, \delta_1, q_1, F_1)$ t. che $\mathcal{L}(M_1) = A$

$M_2 = (Q_2, \Sigma_2, \delta_2, q_2, F_2)$ t. che $\mathcal{L}(M_2) = B$

- $Q = Q_1 \times Q_2$ ; • $q_0 = (q_1, q_2)$ ; • assumiamo $\Sigma_1 = \Sigma_2 = \Sigma$ ;
- $\delta : Q \times \Sigma \longmapsto Q$ , cioè : $\delta : (Q_1 \times Q_2) \times \Sigma \longmapsto (Q_1 \times Q_2)$ tale che :

$\delta((q_i, q_j), \sigma) \longmapsto (\delta_1(q_i, \sigma) , \delta_2(q_j, \sigma))$ $\Rightarrow$ "simulazione in parallelo"

- $\boxed{F = (F_1 \times Q_2) \cup (Q_1 \times F_2)}$

Per costruzione, se $w \in \Sigma$ era accettata da $M_1$ o $M_2$, sarà accettata anche da $M$.

$\Rightarrow$ $A \cup B$ è un LINGUAGGIO REGOLARE.

(2) $A \cap B$ è regolare :

Stessa dimostrazione, solo che : $\boxed{F = F_1 \times F_2}$ , poiché sia $M_1$ che $M_2$ devono accettare.

- NFA : Nondeterministic Finite Automaton :

È sempre una quintupla $M = (Q, \Sigma, \delta, q_0, F)$, ma con una FUNZIONE DI TRANSIZIONE $\delta$ più "libera" :

$$\boxed{\delta : Q \times (\Sigma \cup \{\varepsilon\}) \longmapsto \mathcal{P}(Q) \equiv 2^Q}$$

$\delta$ porta una coppia (stato, simbolo), volendo anche senza consumare simboli di input, non leggendo nulla, cioè $\varepsilon$, in un SOTTOINSIEME di stati di $Q$.

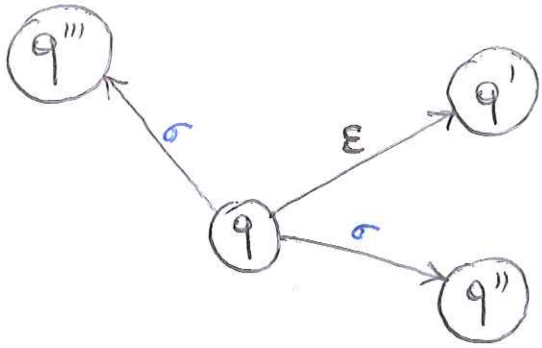
- COMPUTAZIONE ACCETTANTE :

Se esiste una sequenza di stati $r_0, r_1, \dots, r_m \in Q$ tali che :

(1) $r_0 = q_0$ ;

(2) $\forall i = 0, \dots, m-1$ , $r_{i+1} \in \delta(r_i, y_{i+1})$ , dove $y_{i+1} = \varepsilon$ oppure $y_{i+1} = w_{i+1}$ ;

(3) $r_m \in F$ .




* Il NONDETERMINISMO sta sia nella ε-arrow, sia nel poter andare in 2 stati differenti consumando lo stesso simbolo σ.
Di fatto l'automa processa entrambi i percorsi "IN PARALLELO"

- **DFA vs NFA:**

**DFA**
- Ogni stato ed ogni simbolo letto determinano lo stato successivo;
- Non è possibile cambiare di stato senza leggere simboli;
- Procede LINEARMENTE fino alla fine.

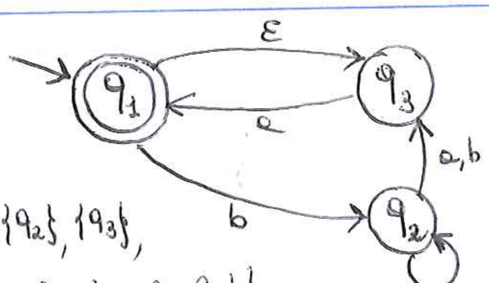
**NFA**
- Si può "scegliere" di andare in più stati (PIU' FLESSIBILITA');
- L'automa accetta se ALMENO un ramo di computazione termina in uno stato d'accettazione;
- L'automa cerca le sequenze accettante in PARALLELO;
- Il numero di percorsi è ESPONENZIALE!

## TEOREMA DI EQUIVALENZA

Ogni NFA N ha un DFA M equivalente, tale che $L(M) \equiv L(N)$.

- Esempio:

Convertire il seguente NFA in un DFA:



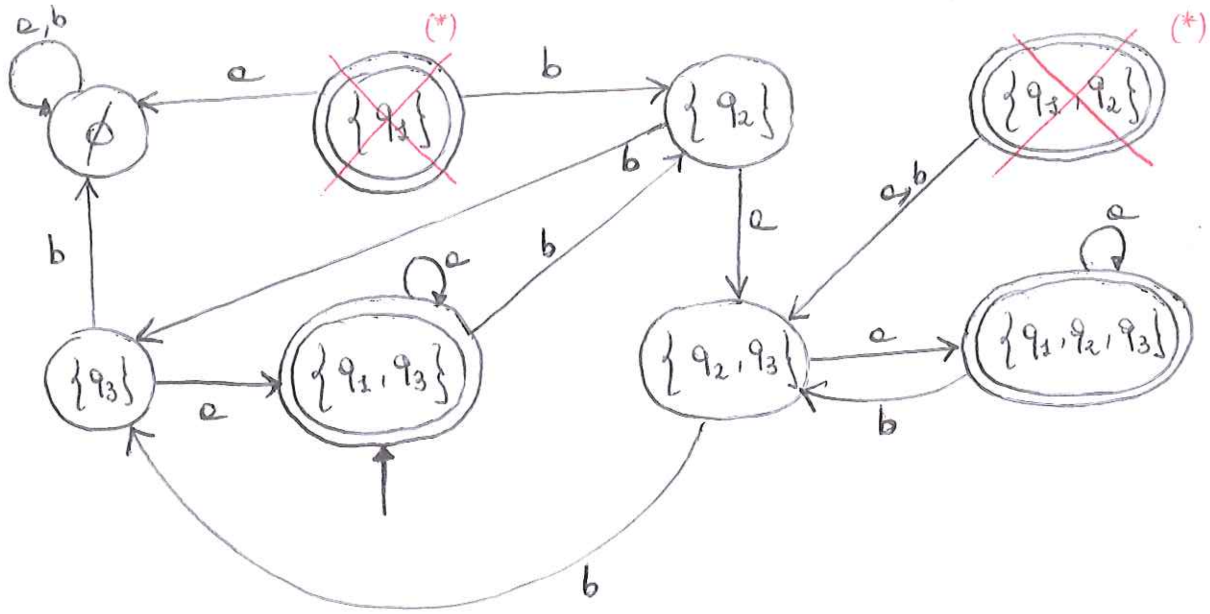

- $Q = \{q_1, q_2, q_3\} \longrightarrow Q' = 2^Q = \{\emptyset, \{q_1\}, \{q_2\}, \{q_3\}, \{q_1,q_2\}, \{q_1,q_3\}, \{q_2,q_3\}, \{q_1,q_2,q_3\}\}$

- $F = \{q_1\} \longrightarrow F' = \{\{q_1\}, \{q_1,q_2\}, \{q_1,q_3\}, \{q_1,q_2,q_3\}\}$ → Tutti gli stati che contengono $q_1$ sono accettanti

* C'è una ε-arrow, definiamo la FUNZIONE DI CHIUSURA; sono influenzati solo gli stati contenenti $q_1$:

$E(\{q_1\}) = \{q_1, q_3\}$ ; $E(\{q_1,q_2\}) = \{q_1,q_2,q_3\}$

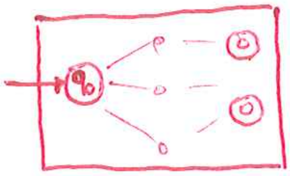
- $q_0 = q_1 \longrightarrow q_0' = E(\{q_1\}) = \{q_1, q_3\}$ → START STATE



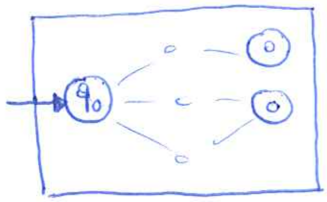

* **NOTA**: gli stati $\{q_1\}$ e $\{q_1,q_2\}$ <u>non</u> sono raggiungibili da alcuna freccia; possono quindi essere rimossi. (*)

## (1) $A \cup B$ è REGOLARE:

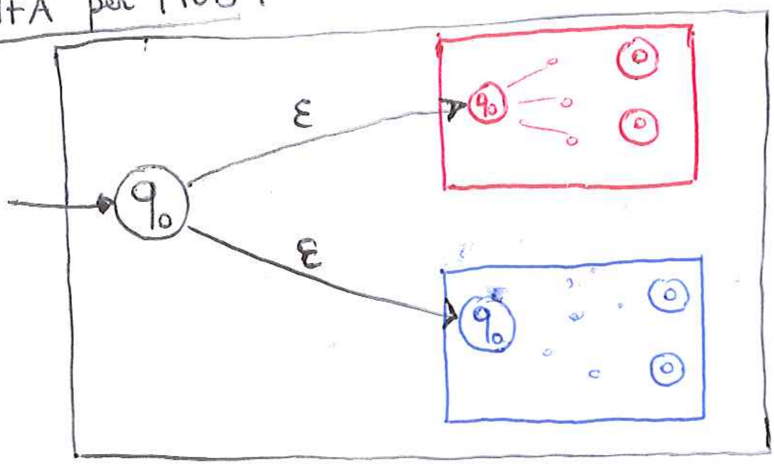
A:




B:



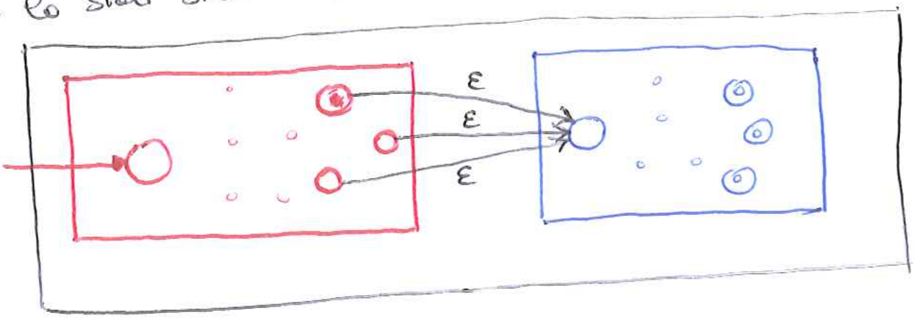

NFA per $A \cup B$:




Aggiungere un nuovo START STATE $q_0$ con 2 $\varepsilon$-arrows verso gli start state di $M_1$ ed $M_2$; computazione in parallelo.

## (3) $A \circ B$ è REGOLARE:

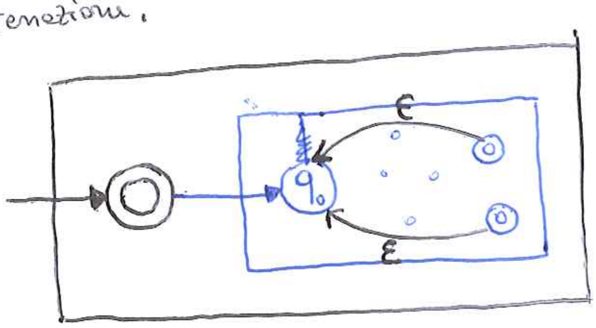
Gli stati di accettazione di A diventano stati normali, ma con delle ε-arrows verso lo start state di B:




$A \circ B$

## (4) $A^*$ è REGOLARE:

C'è uno stato di accettazione iniziale, per poter accettare ε. Gli stati di accettazione di A hanno una ε-arrow che riporta allo stato $q_0$ per valutare le successive concatenazioni.




$A^* = \{x_1 x_2 \dots x_k \mid k \geq 0 \land x_i \in A\}$

## • Espressioni Regolari (REX):

Sono definite induttivamente. Sia $\Sigma$ un alfabeto e siano $R_1$ ed $R_2$ espressioni regolari, allora:

1) $a$ è una REX, se $a \in \Sigma$;
2) $\varepsilon$ è una REX;
3) $\emptyset$ è una REX;
4) $(R_1 \cup R_2)$ è una REX;
5) $(R_1 \circ R_2)$ è una REX;
6) $(R_1)^*$ è una REX.

Ad ogni REX è associato un <u>LINGUAGGIO</u>:

- $\mathcal{L}(a) = \{a\}$, $\forall a \in \Sigma$
- $\mathcal{L}(\varepsilon) = \{\varepsilon\}$
- $\mathcal{L}(\emptyset) = \emptyset$
- $\mathcal{L}((R_1 \cup R_2)) = \mathcal{L}(R_1) \cup \mathcal{L}(R_2)$
- $\mathcal{L}((R_1 \circ R_2)) = \mathcal{L}(R_1) \circ \mathcal{L}(R_2)$
- $\mathcal{L}((R_1)^*) = (\mathcal{L}(R_1))^*$

**Notazioni:**

$R_1 \cup R_2 \longrightarrow R_1 + R_2$

$R_1 \circ R_2 \longrightarrow R_1 R_2$

$R^+ := RR^* \longrightarrow$ tutte le sequenze formate da 1 o più elementi di $R$

$R^k := \underbrace{RR\ldots R}_{k \text{ volte}} \longrightarrow$ tutte le sequenze formate da $k$ elementi di $R$

*** Nota:** non confondere $\varepsilon$ con $\emptyset$! Abbiamo le seguenti proprietà:

1. $R\varepsilon = R$;
2. $R\emptyset = \emptyset$;
3. $R \cup \varepsilon = \mathcal{L}(R) \cup \{\varepsilon\}$, può essere diverso da $\mathcal{L}(R)$!
4. $R \cup \emptyset = \mathcal{L}(R) \cup \emptyset = \mathcal{L}(R)$

**Esempio:**

REX: $\emptyset^*$. Qual è il linguaggio associato?

$$\mathcal{L}(\emptyset^*) = (\mathcal{L}(\emptyset))^* = \emptyset^* = \{\varepsilon\} \neq \emptyset$$

## • Teorema di Equivalenza (REX):

$\boxed{\text{Un linguaggio } \mathcal{L} \text{ è REGOLARE} \iff \exists \text{ REX } R \text{ tale che: } \mathcal{L}(R) = \mathcal{L}}$

• Esempio: Costruzione di McNaughton-Yamada:

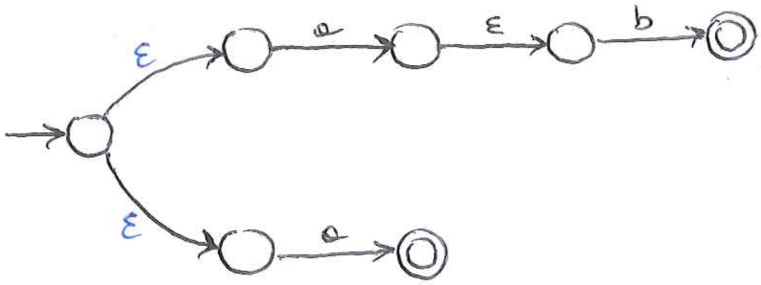
REX = $(ab \cup a)^*$, con $\Sigma = \{a, b\}$

Vogliamo dimostrare che $\mathcal{L}((ab \cup a)^*)$ è REGOLARE, esibendo l'automa che lo riconosce.

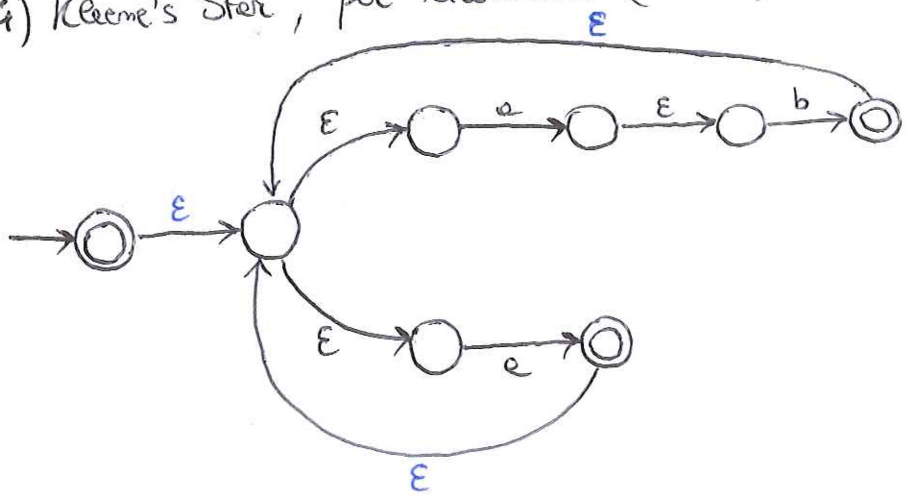
(1) Per riconoscere a e b:


(2) Per riconoscere ab, devo concatenare:


(3) Unione, per riconoscere $ab \cup a$:

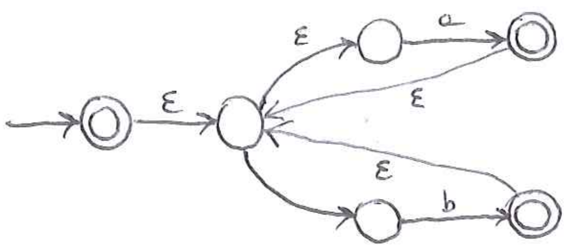



(4) Kleene's Star, per riconoscere $(ab \cup a)^*$:

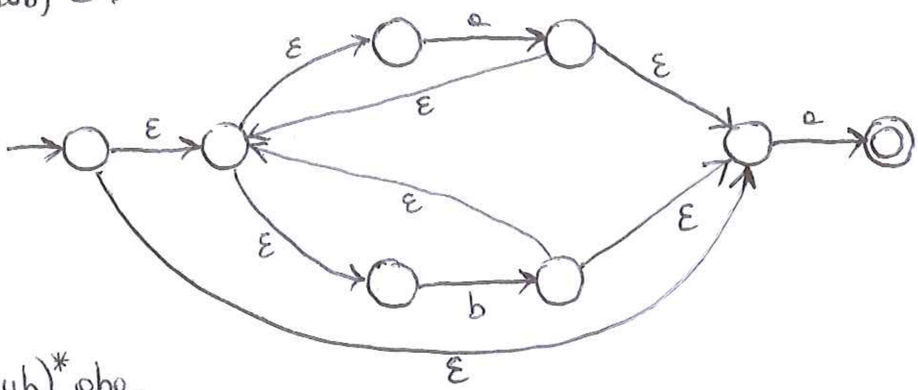


Esercizio: $R = (a \cup b)^* aba$ , con $\Sigma = \{a, b\}$ . Costruire l'automa (NFA).

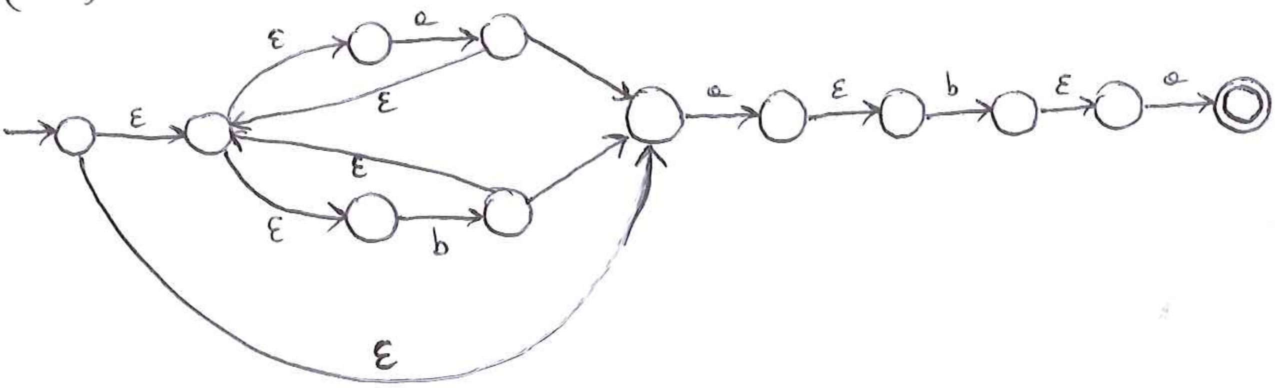
1. $(a \cup b)^*$:




2. $(a \cup b)^* a$ :

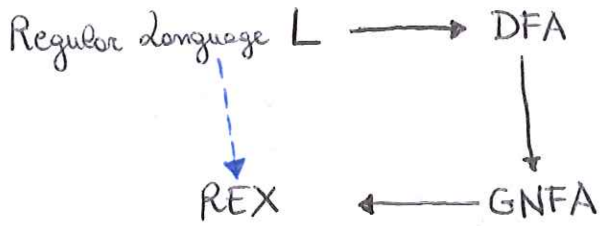



3. $(a \cup b)^* aba$




## • Da Automa a REX :

Bisognerà fare il seguente percorso:




## • GNFA: Generalized NFA :

Per i GNFA, ogni transizione è marcata da una REX; dunque, per ogni transizione può essere consumata una porzione arbitraria di input di lunghezza da 0 a $k \in \mathbb{N}^+$.

Regole di transizioni:

(1) In $q_{START}$ non può esserci alcuna freccia entrante;

(2) Unico stato d'accettazione $q_{ACCEPT}$, che non può avere frecce uscenti;

(3) Per ogni coppia di nodi devono esserci tutte le transizioni definite, inclusi i self-loop, ad eccezione di $q_{START}$ e $q_{ACCEPT}$.

- GNFA:

GNFA $M = (Q, \Sigma, \delta, q_{START}, q_{ACCEPT})$.

* L'unica differenza è in $\delta$:

$$\delta : (Q \setminus \{q_{ACCEPT}\}) \times (Q \setminus \{q_{START}\}) \longmapsto \mathcal{R} = \{\text{REX over } \Sigma\}$$

- COMPUTAZIONE ACCETTANTE:

Sia $w = w_1 \dots w_k$, $w_i \in \Sigma^* \ \forall i = 1, \dots, k$.

$M$ accetta $w$ se $\exists\ r_0, r_1, \dots, r_m$ tale che:

(1) $r_0 = q_{START}$;

(2) $w_i \in \mathcal{L}(\delta(r_{i-1}, r_i))$, $\forall i = 1, \dots, k$;

$\delta(r_{i-1}, r_i)$ è la REX presente sulla transizione; $w_i$ deve appartenere al linguaggio generato dalla REX.

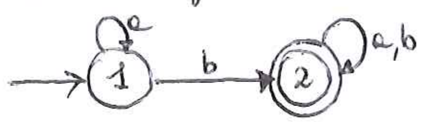
(3) $r_m = q_{ACCEPT}$.

* **NOTA**: c'è il NONDETERMINISMO! Inoltre $q_{START} \neq q_{ACCEPT}$

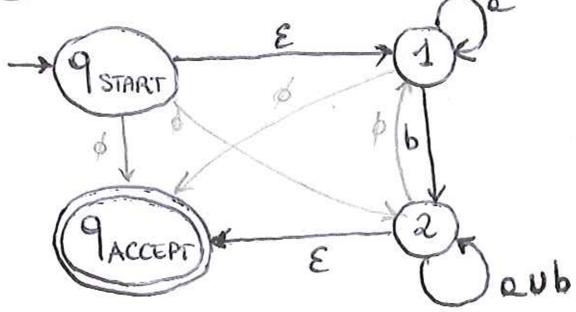
- Da GNFA a REX:

$\boxed{\text{Per ogni GNFA } M, \text{ esiste una REX } R \text{ tale che: } \mathcal{L}(M) = \mathcal{L}(R)}$

- Esercizio: Dato il seguente DFA, ricavarne la REX equivalente:




(1) Ottenere il GNFA:

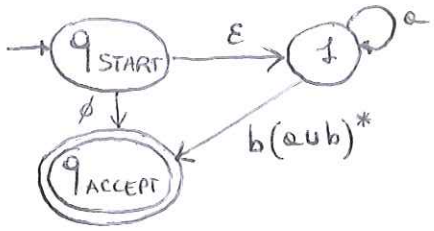



(2) Semplificare, partendo dal nodo 2:

$q_{START} \to q_{ACCEPT}$ : $\phi \cup \phi(a \cup b)^* \varepsilon = \phi \cup \phi = \phi$

$1 \to q_{ACCEPT}$ : $b(a \cup b)^* \varepsilon = b(a \cup b)^*$.

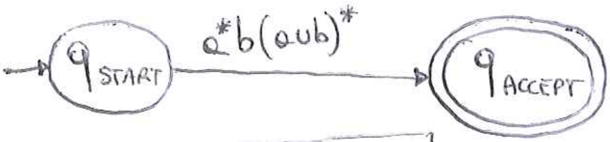
Si ottiene :




(3) Procedere, eliminando il nodo 1 :

$$q_{START} \rightarrow q_{ACCEPT} : \quad \emptyset \cup \varepsilon(a)^* b(a \cup b)^* = a^* b(a \cup b)^*$$

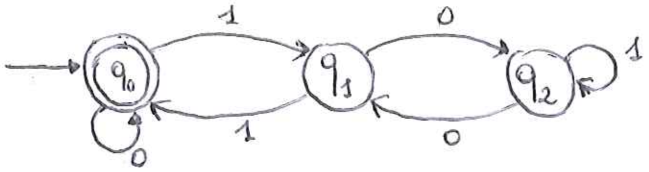
(4) Si ottiene la REX :




La REX cercata è $\boxed{R = a^* b(a \cup b)^*}$

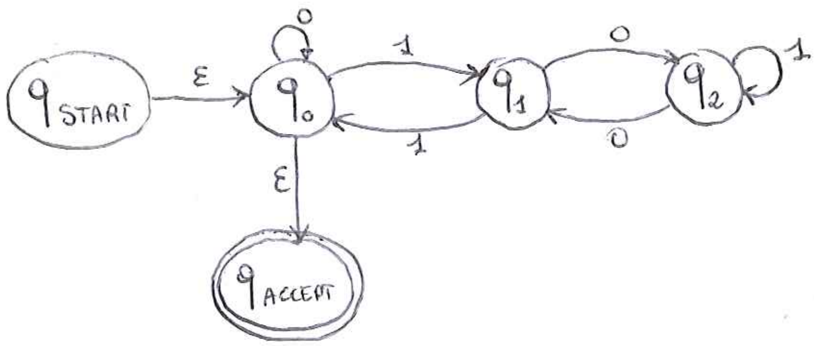
Esercizio :

Linguaggio $L = \{ w \in \{0,1\}^* \mid w = \varepsilon \lor w \text{ encodes a binary number divisible by } 3 \}$ riconosciuto del DFA :

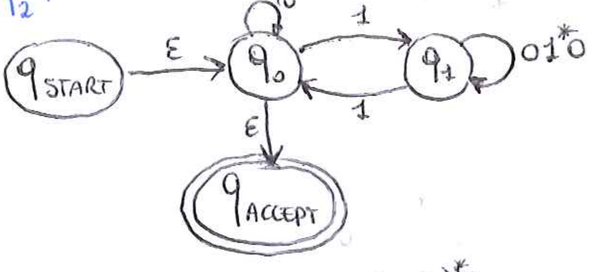



Costruire una REX equivalente.

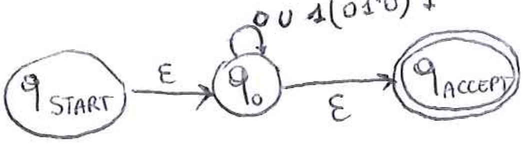
GNFA :




- Rimuovo $q_2$ :




- Rimuovo $q_1$ :



- Rimuovo $q_0$, ottenendo la REX:

$$q_{START} \xrightarrow{\varepsilon \cup (0 \cup 1(01^*0)^*1)^*} q_{ACCEPT}$$

$$\rightarrow \boxed{R = \varepsilon \cup (0 \cup 1(01^*0)^*1)^*}$$

**Riepilogo:** Come dimostrare che un linguaggio sia REGOLARE?

1) Esibire un DFA;
2) Esibire un NFA;
3) Esibire un GNFA;
4) Esibire una REX.

* SONO TUTTI EQUIVALENTI !!!

# PUMPING LEMMA

Se $A$ è un linguaggio regolare $\Rightarrow$ Esiste una "PUMPING LENGHT" $p > 0$, tale che:

se $s \in A$ e $|s| \geq p$, allora $s = xyz$, in modo che:

(1) $\forall i \geq 0$, $xy^iz \in A$;

(2) $|y| > 0$;

(3) $|xy| \leq p$.

* Serve per dimostrare che un linguaggio sia NON regolare!

Sfrutta il PRINCIPIO DELLA PICCIONAIA per individuare un loop tra gli stati del DFA $M$ che riconosce $A$; sostanzialmente, tale loop costituisce una parte di stringa che può essere pompata verso l'alto o verso il basso.

- Un linguaggio non è regolare se:

$\forall p > 0$, $\exists \bar{s} \in A$, $|\bar{s}| \geq p$ : $\forall$ suddivisione $\bar{s} = xyz$, con $|xy| \leq p$, $|y| > 0$,

$\exists \bar{i} \geq 0$ tale che : $xy^{\bar{i}}z \notin A$ $\Rightarrow$ $A$ non è REGOLARE

1) Per assurdo, si assume che il linguaggio $L$ sia regolare;
2) Applico il Pumping Lemma, fissando una pumping lenght $p$;
3) Trovo una stringa $\bar{s} \in L$ e faccio vedere che PER OGNI SUDDIVISIONE $\bar{s} = xyz$, con $|y| > 0$ e $|xy| \leq p$, esiste un $\bar{i} \geq 0$ tale per cui $xy^{\bar{i}}z \notin L$
4) Per assurdo, $L$ è NON REGOLARE.

• Esercizi:

1) Provare che $B=\{0^n1^n \mid n\geq 0\}$ è NON regolare.

Sia per assurdo regolare $\Rightarrow \exists$ pumping lenght $p>0$; Scegliamo $\bar{s}=0^p1^p$, $|\bar{s}|\geq p$.

Abbiamo 3 possibili casi di suddivisione:

- $y \in L(0^*)$
- $y \in L(1^*)$
- $y \in L(0^*)\circ L(1^*)$

CASO 1: $\bar{i}=2 \rightarrow xyyz$: $\overbrace{\underbrace{00\ldots0}_{xy}\underbrace{0\ldots0}_{y}}^{>p}\overbrace{\underbrace{11\ldots1}_{z}}^{p} \Rightarrow xyyz \notin B$

CASO 2: Analogo; $\bar{i}=2$: $xyyz$: $\overbrace{\underbrace{00\ldots0}_{x}}^{p}\overbrace{\underbrace{1\ldots1}_{y}\underbrace{11\ldots1}_{yz}}^{>p} \Rightarrow xyyz \notin B$

CASO 3: $i=2$ $\quad y=0^k1^k$, $k<p$

$xyyz$: $\underbrace{00\ldots0}_{x}\underbrace{0\ldots01\ldots1}_{y}\underbrace{0\ldots01\ldots1}_{y}\underbrace{11\ldots1}_{z} \rightarrow xyyz \notin B$

Il Pumping Lemma non è verificato $\rightarrow$ ↯ ASSURDO! $\rightarrow$ B è NON REGOLARE!

• **NOTA**: sfruttando la condizione $|xy|\leq p$, si sarebbe potuto esaminare solo il caso 1.

2) $C=\{w\in\{0,1\}^* \mid w \text{ ha lo stesso numero di } 0 \text{ e di } 1\}$ è NON REGOLARE.

Sia per assurdo regolare $\Rightarrow \exists\, p>0$; $\bar{s}=0^p1^p$; $|\bar{s}|\geq p$

Poiché $|xy|\leq p$ e $|y|>0 \Rightarrow y\in L(0^*)$

$\Rightarrow \forall i\geq 2$: $xy^iz = 0^q1^p$, con $q>p \Rightarrow$ C è NON REGOLARE! ok!!!

* SENZA PUMPING LEMMA:

$B\subseteq C$ e B è non regolare. Linguaggi regolari sono chiusi rispetto all'intersezione; $L(0^*1^*)$ è regolare poiché indotto da una REX.

$B = C\cap L(0^*1^*)$; Se C fosse regolare, anche B lo sarebbe.

Ma B è NON regolare $\Rightarrow$ C è NON regolare.

3) $F = \{ww \mid w \in \{0,1\}^*\}$ è NON regolare.

$\bar{s} = 0^p 1 0^p 1$ ; $|\bar{s}| \geq p$

Poiché $|xy| \leq p \Rightarrow y \in \mathcal{L}(0^*)$

$\Rightarrow \forall i \geq 2 : xy^i z = 0^q 1 0^p 1$, con $q > p$ $\Rightarrow$ ↯ ASSURDO! $\Rightarrow$ F è NON REGOLARE!

4) $E = \{0^i 1^j \mid i > j\}$ è NON regolare.

$\exists p > 0$ tale che vale il PL.

$\bar{s} = 0^{p+1} 1^p$ ; $|\bar{s}| \geq p$

$|xy| \leq p$ ed $|y| > 0 \Rightarrow y \in \mathcal{L}(0^*)$

POMPAGGIO VERSO IL BASSO ; $\bar{i} = 0 \Rightarrow xy^i z = xz = 0^q 1^p$, con $q \leq p$

$\Rightarrow 0^q 1^p = xz \notin E \longrightarrow$ ↯ ASSURDO! $\longrightarrow$ E è NON regolare!

- **Il COMPLEMENTO:**

Il complemento di un linguaggio $\mathcal{L} \subseteq \Sigma^*$ è

$$\mathcal{L}^c := \Sigma^* \setminus \mathcal{L} = \{w \in \Sigma^* \mid w \notin \mathcal{L}\}$$

* $\boxed{\text{Se } \mathcal{L} \text{ è REGOLARE} \Rightarrow \mathcal{L}^c \text{ è REGOLARE}}$

Basta invertire gli stati d'accettazione dell'automa in stati normali e viceversa

- **Utilizzo:**

Dimostrare che $A = \{w \in \{0,1\}^* \mid w$ ha un differente numero di 0 e di 1$\}$ è NON regolare.

$A^c = \{w \in \{0,1\}^* \mid \#0 = \#1\}$. E' il C degli esempi precedenti, ed è non regolare.

Se A fosse regolare, anche $A^c$ lo sarebbe; ma non lo è.

$\Rightarrow$ A è NON REGOLARE!

- **Il REVERSAL:**

Sia $\sigma \in \Sigma^*$, $\sigma = \sigma_1 \dots \sigma_m \Rightarrow \sigma^R := \sigma_m \dots \sigma_1$

Sia $A \subseteq \Sigma^* \Rightarrow A^R := \{w \in \Sigma^* \mid w^R \in A\}$

* $\boxed{\text{Se } A \text{ è regolare} \Rightarrow A^R \text{ è regolare}}$

M' è un NFA con un nuovo start state, con ε-arrows verso i vecchi stati di accettazione di M; le frecce sono invertite e l'unico stato di accettazione è lo START STATE di M.

## L'OMOMORFISMO:

Un omomorfismo su un alfabeto $\Sigma$ è una funzione:

$$h: \Sigma^* \longmapsto \Gamma^*$$

Con $\Gamma$ un alfabeto, che può essere anche $\Sigma$ stesso.

### SU STRINGHE:

Sia $w = w_1 \dots w_n \in \Sigma^* \Rightarrow h(w) = h(w_1) \dots h(w_n)$

È la concatenazione degli omomorfismi dei singoli caratteri della stringa.

### SU LINGUAGGI:

Sia $A \subseteq \Sigma^* \Rightarrow h(A) = \{ u \in \Gamma^* \mid \exists w \in \Sigma^* \text{ tale che } h(w) = u \}$

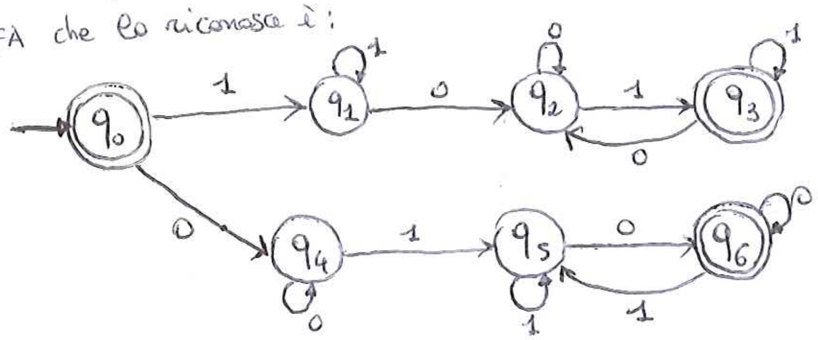
$*$ $\boxed{\text{Linguaggi Regolari sono chiusi rispetto all' OMOMORFISMO.}}$

## Esercizio:

(1) $D = \{ w \in \{0,1\}^* \mid w \text{ ha lo stesso numero di sottostringhe "01" e "10"} \}$ è REGOLARE.

Una REX equivalente è: $R = (010)^* \cup (101)^*$

Un DFA che lo riconosce è:




(2) $D' = \{ w \in \{0,1\}^* \mid w \text{ ha lo stesso numero di sottostringhe "00" e "11"} \}$ è NON REGOLARE.

Per assurdo, sia regolare, vale il Pumping Lemma: $\exists p > 0$;

$\overline{S} = (00)^p (11)^p$ ; $|\overline{S}| \geq p$

$|xy| \leq p$ e $|y| > 0 \Rightarrow i = 2 \rightarrow xyyz = 0^q 1^{2p}$, con $q > 2p \Rightarrow xyyz \notin D'$

$\rightarrow$ ↯ ASSURDO $\Rightarrow D'$ è NON REGOLARE.