Argomenti trattati durante la lezione:

- Intro al corso, automi DFA e progettazione
- Equivalenza tra NFA e DFA. Chiusura rispetto alle operazioni regolari
- Esercizi DFA/NFA

Un Automa a Stati Finiti Deterministico (DFA) è una quintupla

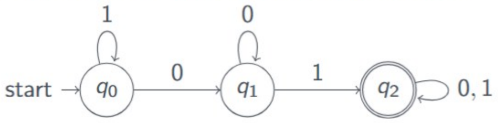
$$A = (Q, \Sigma, \delta, q_0, F)$$

- Q è un insieme finito di stati
- Σ è un alfabeto finito (= simboli in input)
- $\delta : Q \times \Sigma \mapsto Q$ è una funzione di transizione
- $q_0 \in Q$ è lo stato iniziale
- $F \subseteq Q$ è un insieme di stati finali

Possiamo rappresentare gli automi sia come diagramma di transizione che come tabella di transizione.

Esempio: costruiamo un automa A che accetta il linguaggio delle stringhe con 01 come sottostringa

- L'automa come diagramma di transizione:



- L'automa come tabella di transizione:

	0	1
$\rightarrow q_0$	q_1	q_0
q_1	q_1	q_2
$*q_2$	q_2	q_2

Esercizio 1 (1)

Costruire un DFA su $\Sigma = \{0, 1\}$ che accetti l'insieme di tutte le stringhe aventi tre 0 consecutivi.

Osservazioni:

- l'alfabeto è dato ($\Sigma = \{0, 1\}$);
- si possono individuare le seguenti possibili situazioni:
  - (a) sono già stati letti tre (o più) "0" consecutivi;
  - (b) gli ultimi due simboli letti sono "0";
  - (c) l'ultimo simbolo letto è "0";
  - (d) l'ultimo simbolo letto è "1".

Esercizio 1 (2)

Indicando con q_i lo stato in cui l'automa si troverà dopo aver letto i simboli "0" consecutivi:

- situazione (a) $\leftrightarrow q_3$;
- situazione (c) $\leftrightarrow q_1$;
- situazione (b) $\leftrightarrow q_2$;
- situazione (d) $\leftrightarrow q_0$;

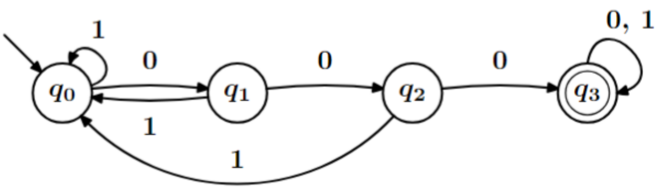
Note:

- dopo aver letto tre "0", l'automa accetterà la stringa qualunque siano i simboli ancora da leggere;
- q_3 assume quindi il ruolo di stato finale;
- q_0 assume il ruolo di stato iniziale;
- uno stato dal quale non si esce viene detto "pozzo".

La funzione di transizione può essere:

δ	0	1
q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_3	q_0
q_3	q_3	q_3

e il grafico:



Esercizio 2 (1)

Costruire un DFA su $\Sigma = \{0, 1\}$ che accetti l'insieme di tutte le stringhe che se interpretate in notazione binaria risultino divisibili per 2.

Osservazioni:

- l'alfabeto è dato ($\Sigma = \{0, 1\}$);
- i numeri pari in notazione binaria terminano per 0;
- un numero in notazione binaria che termina per 0 è pari;
- il problema diventa costruire l'automa che accetta solo le stringhe binarie che terminano per 0.

Esercizio 2 (2)

Se, banalmente, indichiamo con q_i lo stato in cui viene a trovarsi l'automa dopo aver letto il simbolo i:

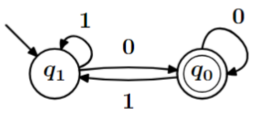
- ponendo q_0 come unico stato finale facciamo sì che l'automa accetti la stringa solo se l'ultimo simbolo che ha letto è stato 0;
- ponendo q_1 come stato iniziale evitiamo che venga accettata la stringa vuota.

Esercizio 2 (3)

Quindi, l'automa può essere formalizzato come:

- $Q = \{q_0, q_1\}$
- $\Sigma = \{0, 1\}$
- q_1 è lo stato iniziale
- $F = \{q_0\}$
-

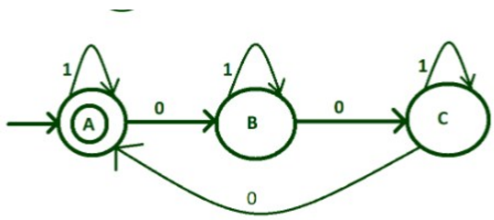
δ	0	1
q_0	q_0	q_1
q_1	q_0	q_1



Costruite un DFA che riconosce il linguaggio

$$L_1 = \{w \in \{0,1\}^* \mid w \text{ contiene un numero di 0 multiplo di 3}\}$$

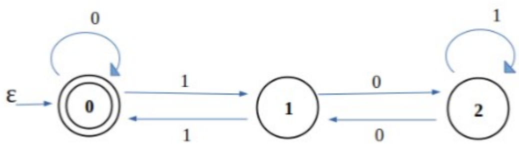
Per esempio, 000, 00110 e 010101010101 appartengono al linguaggio perché contengono rispettivamente 3, 3 e 6 zeri, mentre 00, 001010 e 0101010101, che contengono 2, 4 e 5 zeri, non appartengono al linguaggio.



2. Costruite un DFA che riconosce il linguaggio

$$L_2 = \{w \in \{0,1\}^* \mid w \text{ è la codifica binaria di un mumero multiplo di 3}\}$$

Per esempio, 11, 110 e 1001 appartengono al linguaggio perché sono le codifiche binarie di 3, 6 e 9, mentre 10, 111 e 1011 non appartengono al linguaggio perché sono le codifiche binarie di 2, 7 e 11. La stringa vuota non codifica nessun numero.



- <u>Operazioni regolari</u>

- **Intersezione:**
  $$L \cap M = \{w : w \in L \text{ e } w \in M\}$$
- **Unione:**
  $$L \cup M = \{w : w \in L \text{ oppure } w \in M\}$$
- **Complemento:**
  $$\overline{L} = \{w : w \notin L\}$$
- **Concatenazione:**
  $$L.M = \{uv : u \in L \text{ e } v \in M\}$$
- **Chiusura (o Star) di Kleene:**
  $$L^* = \{w_1 w_2 \dots w_k : k \geq 0 \text{ e ogni } w_i \in L\}$$

Theorem

Se L e M sono regolari, allora anche $L \cap M$ è un linguaggio regolare.

Dimostrazione. Sia L il linguaggio di

$$A_L = (Q_L, \Sigma, \delta_L, q_L, F_L)$$

e M il linguaggio di

$$A_M = (Q_M, \Sigma, \delta_M, q_M, F_M)$$

Possiamo assumere che entrambi gli automi siano deterministici
Costruiremo un automa che simula A_L e A_M in parallelo, e accetta se e solo se sia A_L che A_M accettano.

PROOF

Let M_1 recognize A_1, where $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, and
M_2 recognize A_2, where $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$.

Construct M to recognize $A_1 \cup A_2$, where $M = (Q, \Sigma, \delta, q_0, F)$.

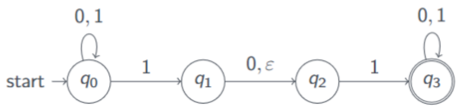
1. $Q = \{(r_1, r_2) | r_1 \in Q_1 \text{ and } r_2 \in Q_2\}$.
   This set is the ***Cartesian product*** of sets Q_1 and Q_2 and is written $Q_1 \times Q_2$. It is the set of all pairs of states, the first from Q_1 and the second from Q_2.
2. Σ, the alphabet, is the same as in M_1 and M_2. In this theorem and in all subsequent similar theorems, we assume for simplicity that both M_1 and M_2 have the same input alphabet Σ. The theorem remains true if they have different alphabets, Σ_1 and Σ_2. We would then modify the proof to let $\Sigma = \Sigma_1 \cup \Sigma_2$.
3. δ, the transition function, is defined as follows. For each $(r_1, r_2) \in Q$ and each $a \in \Sigma$, let
   $$\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a)).$$
   Hence δ gets a state of M (which actually is a pair of states from M_1 and M_2), together with an input symbol, and returns M's next state.
4. q_0 is the pair (q_1, q_2).
5. F is the set of pairs in which either member is an accept state of M_1 or M_2. We can write it as
   $$F = \{(r_1, r_2) | r_1 \in F_1 \text{ or } r_2 \in F_2\}.$$

Dati i DFA per L e M:

- possiamo costruire un DFA per $L \cup M$? Se si, come?
- possiamo costruire un DFA per $\overline{L}$? Se si, come?
- possiamo costruire un DFA per $L.M$? Se si, come?
- possiamo costruire un DFA per L^*? Se si, come?

- <u>Automi NFA</u>

- Cosa fa questo automa?



- È un esempio di automa a stati finiti non deterministico:
  - ci possono essere più transizioni con lo stesso simbolo
  - o simboli senza transizioni uscenti
  - ed ε-transizioni che non consumano simboli
- Data una parola, esistono più percorsi possibili
- Si accetta se esiste almeno un percorso accettante

Un Automa a Stati Finiti Non Deterministico (NFA) è una quintupla

$$A = (Q, \Sigma, \delta, q_0, F)$$

- Q è un insieme finito di stati
- Σ è un alfabeto finito che non contiene ε. Definiamo $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$.
- $\delta : Q \times \Sigma_\varepsilon \mapsto 2^Q$ è una funzione di transizione che prende in input (q, a) e restituisce un sottoinsieme di Q
- $q_0 \in Q$ è lo stato iniziale
- $F \subseteq Q$ è un insieme di stati finali

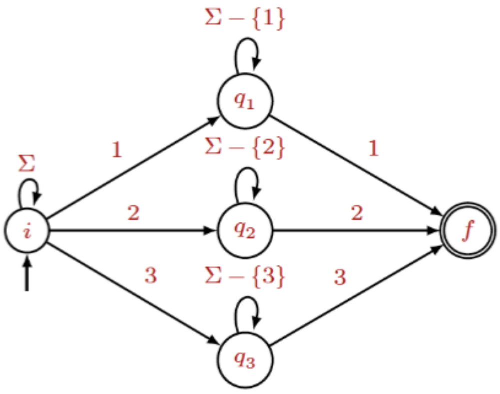
The formal description of N_1 is $(Q, \Sigma, \delta, q_1, F)$, where

1. $Q = \{q_1, q_2, q_3, q_4\}$,
2. $\Sigma = \{0,1\}$,
3. δ is given as

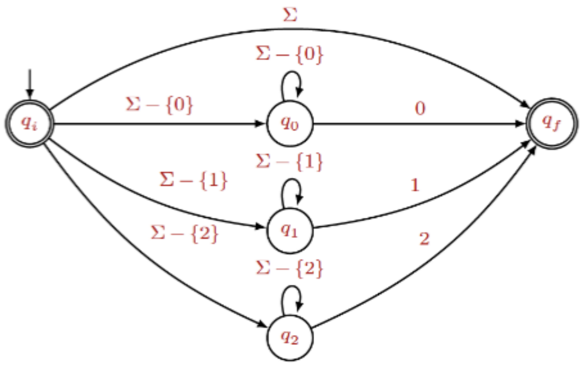
	0	1	ε
q_1	$\{q_1\}$	$\{q_1, q_2\}$	$\emptyset$
q_2	$\{q_3\}$	$\emptyset$	$\{q_3\}$
q_3	$\emptyset$	$\{q_4\}$	$\emptyset$
q_4	$\{q_4\}$	$\{q_4\}$	$\emptyset$,

4. q_1 is the start state, and
5. $F = \{q_4\}$.

- Definire gli NFA per i seguenti linguaggi sull'alfabeto $\{0, 1, 2\}$:
  - Insieme di tutte le stringhe tali che la cifra finale sia comparsa in precedenza.
  - Insieme di tutte le stringhe tali che la cifra finale *non* sia comparsa in precedenza.



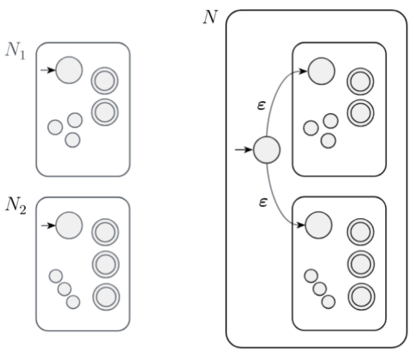
tutte le stringhe tali che la cifra finale sia comparsa in precedenza



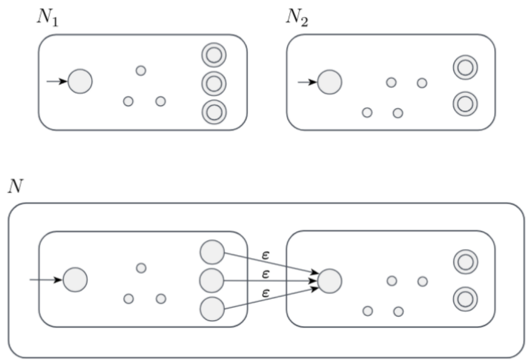
tutte le stringhe tali che la cifra finale **non** sia comparsa in precedenza

Gli automi generalmente sono *chiusi rispetto alle operazioni regolari*.

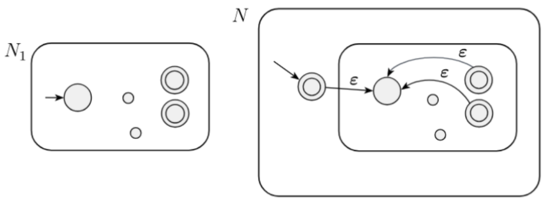
Unione



Concatenazione

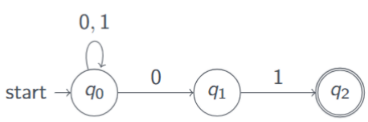


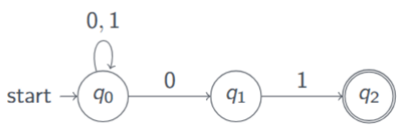
Star di Kleene



- Sorprendentemente, NFA e DFA sono in grado di riconoscere gli stessi linugaggi
- Per ogni NFA N c'è un DFA D tale che $L(D) = L(N)$, e viceversa

Esempio:

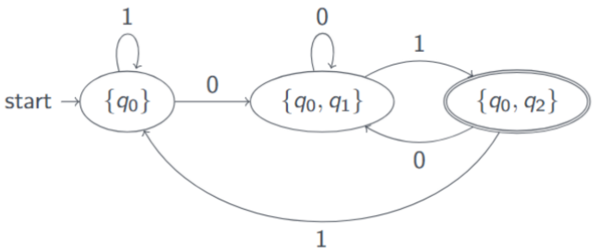




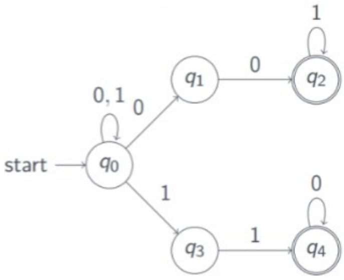
Costruiamo δ_D per l'NFA qui sopra:

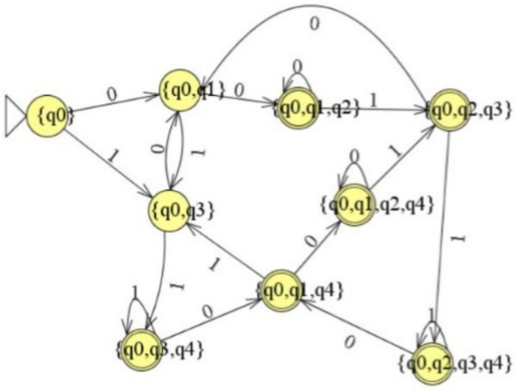
	0	1
$\emptyset$	$\emptyset$	$\emptyset$
$\rightarrow \{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_1\}$	$\emptyset$	$\{q_2\}$
$*\{q_2\}$	$\emptyset$	$\emptyset$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$*\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$
$*\{q_1, q_2\}$	$\emptyset$	$\{q_2\}$
$*\{q_0, q_1, q_2\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$

La tabella di transizione per D ci permette di ottenere il diagramma di transizione



Trasformare il seguente NFA in DFA

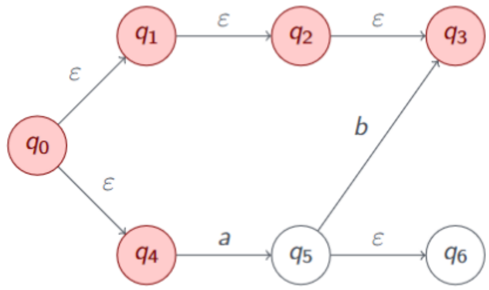




Epsilon chiusura: definizione

Per poter gestire le ε-transizioni introduciamo ε-chiusura degli stati:

- tutti gli stati raggiungibili da q con una sequenza $\varepsilon\varepsilon \dots \varepsilon$



$$\text{ECLOSE}(q_0) = \{q_0, q_1, q_4, q_2, q_3\}$$

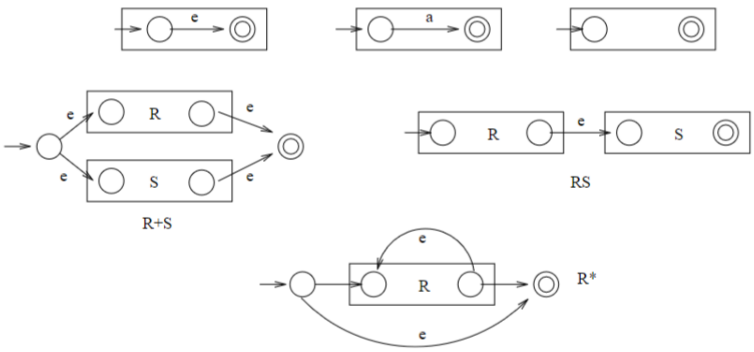
Esercizio 1

A partire dall'espressione regolare $(ab + a)^*$ costruire un ϵNFA equivalente.

Si ricorda che se $L = L(R)$ per una regexp R, allora esiste un $\epsilon NFAE$ tale che $L(E) = L(R)$ con:

1. esattamente uno stato accettante
2. nessun arco entrante in q_0
3. nessun arco uscente dalla stato finale

Le strutture di base sui simboli ($\epsilon = e$ in questo e negli esercizi successivi) e sugli operatori di unione, concatenzione e chiusura sono:



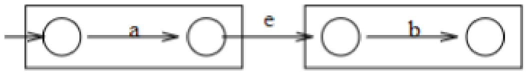


Figure 2: L'ϵ-NFA per ab.

Quest'ultimo viene utilizzato come blocco R per farne l'unione con l'espressione $S = a$:

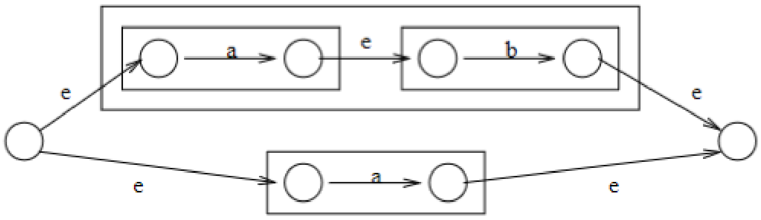


Figure 3: L'ϵ-NFA per $ab + a$.

L'automa risultante rappresenta il blocco R di una chiusura, il che permette di ottenere l'automa corrispondente all'espressione $(ab + a)^*$:

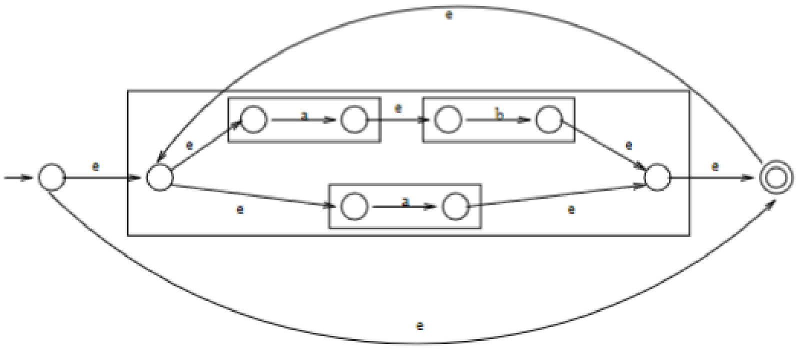
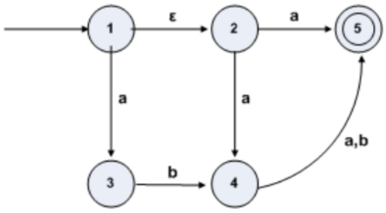


Figure 4: L'ϵ-NFA per $(ab + a)^*$.

Converting an NFA to a DFA - Example

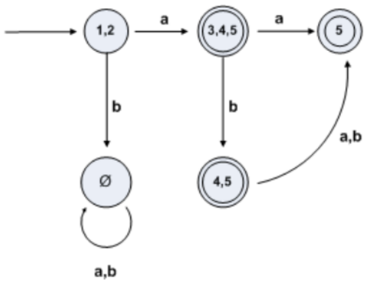
Consider the following NFA



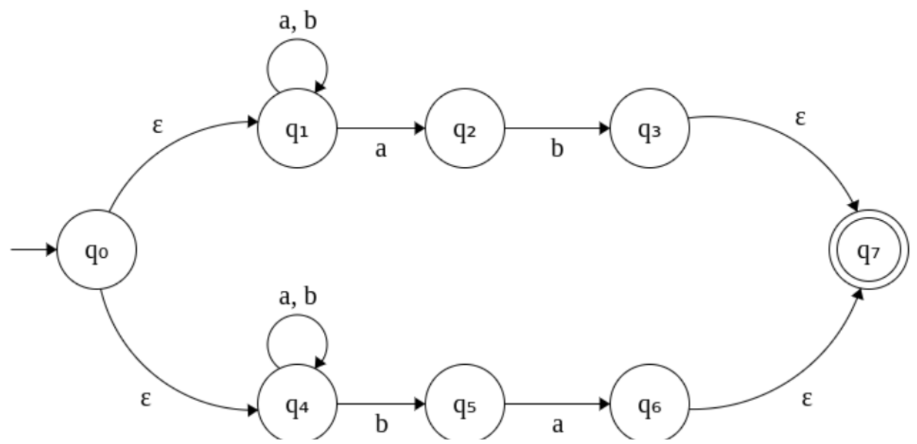

```
Q = states = {1,2,3,4,5}
Start state: { 1 }
Accepting state(s): { 5 }
```

The final table and corresponding DFA state diagram are:

	a	b
{1,2}	{3,4,5}	∅
{3,4,5}	{5}	{4,5}
{5}	∅	∅
{4,5}	{5}	{5}
∅	∅	∅



4A. ε-NFA that accepts strings that end with "ab" or "ba". Σ = {a, b}



Exercise 3 (2.5.3 from textbook) E2.5

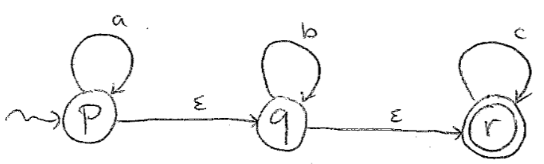
Design ε-NFA's for the following languages.

a) The set of strings [redacted] consisting of zero or more a's followed by zero or more b's, followed by zero or more c's.

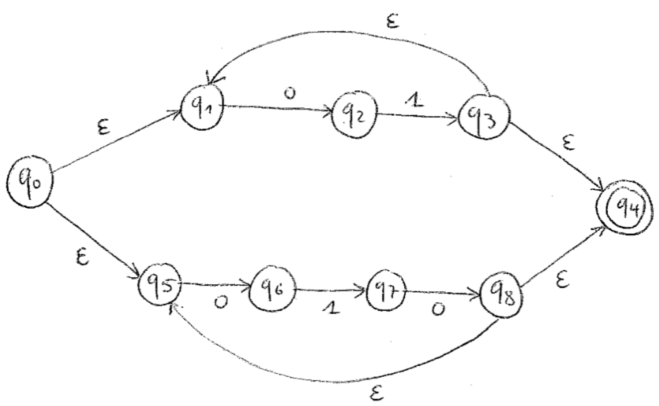
b) The set of all strings that consist [redacted] of either 01 repeated one or more times or 010 repeated one or more times.

Solution:

a)



b)

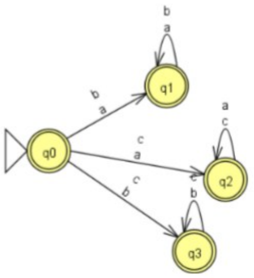


NFA e ε-NFA

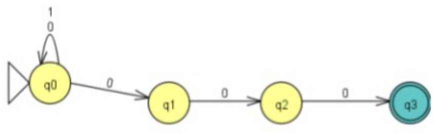
Per ognuno dei seguenti linguaggi, costruisci un ε-NFA che accetti il linguaggio.

1. $\{w \in \{a, b, c\}^* \mid$ non compaiono tutti i simboli$\}$
2. $\{w \in \{0, 1\}^* \mid$ contiene almeno tre 000 consecutivi$\}$
3. $\{w \in \{0, 1\}^* \mid$ contiene al suo interno la stringa 11 oppure 101$\}$

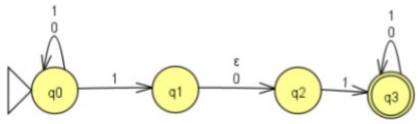
1)



2)

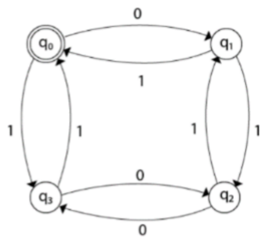


3)

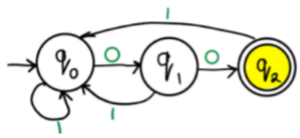


DFA per i seguenti linguaggi sull'alfabeto {0, 1}:

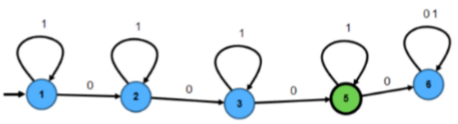
- Insieme di tutte e sole le stringhe con un numero pari di zeri e un numero pari di uni



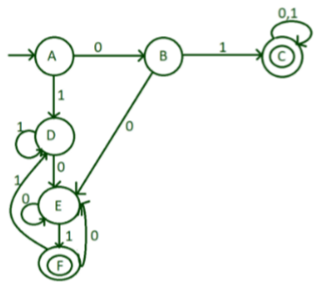
Insieme di tutte le stringhe che finiscono con 00



- Insieme di tutte le stringhe che contengono esattamente tre zeri (anche non consecutivi)



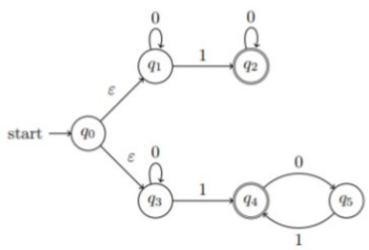
- Insieme delle stringhe che cominciano o finiscono (o entrambe le cose) con 01



Conversione NFA $\to$ DFA

Trasforma ciascuno dei seguenti ε-NFA in DFA usando la costruzione per sottoinsiemi.

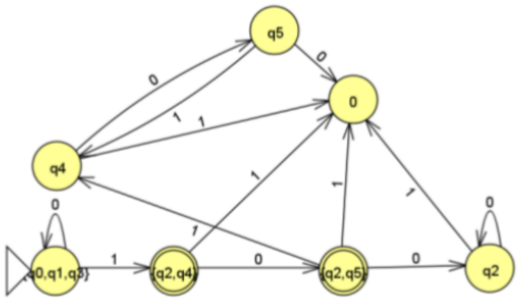
1.



1) Primo step: calcolare le ε-chiusure
In questo caso avremmo (q0,q1) e (q0,q3). Lo stato iniziale sarà quindi (q0,q1,q3).
ENCLOSE (q0) = {q0,q1,q3}

2) Si calcola poi come sempre la tabella di transizione; sempre per unione e osservazione dello stato attuale e stati precedenti, in maniera complementare a quanto visto sopra.
Consiglio: mettere subito lo stato vuoto, perché come si vede servirà nel caso ci siano altri stati vuoti che lo raggiungono.

	0	1
∅	∅	∅
-> {q0,q1,q3}	{q0,q1,q3}	{q2,q4}
{q1,q3}	{q1,q3}	{q2,q4}
*{q2,q4}	{q2,q5}	∅
*{q2,q5}	{q2}	{q4}
{q2}	{q2}	∅
{q4}	{q5}	∅
{q5}	∅	{q4}



1. (**20 Pts**) Design an NFA (non-deterministic finite automata) to accept the set of strings of 0's and 1's that either
   (a) end in 010 and have 011 somewhere preceding, or
   (b) end in 101 and have 100 somewhere preceding

   Solution: