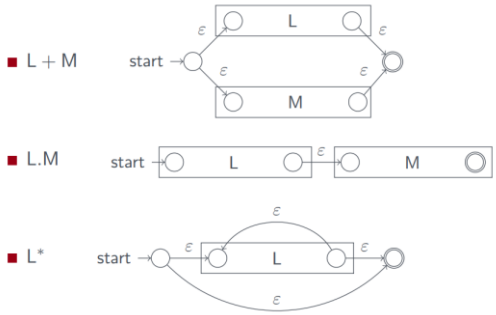
Linguaggio Regolare = Riconoscibile da un automa DFA

Se L e M sono linguaggi regolari, allora anche i seguenti linguaggi sono regolari:

- Unione: $L \cup M$
- Intersezione: $L \cap M$
- Concatenazione: $L.M$
- Complemento: $\overline{L}$
- Chiusura di Kleene: L^*

Grazie all'equivalenza DFA-NFA (con epsilon), è possibile realizzare questo.



Una espressione regolare è un modo dichiarativo per descrivere un linguaggio regolare.

Le Espressioni Regolari sono costruite utilizzando

- un insieme di costanti di base:
  - ε per la stringa vuota
  - $\emptyset$ per il linguaggio vuoto
  - a, b, ... per i simboli $a, b, \cdots \in \Sigma$
- collegati da operatori:
  - $+$ per l'unione
  - $\cdot$ per la concatenazione
  - * per la chiusura di Kleene
- raggruppati usando le parentesi:
  - ( )

- $L(\mathrm{E} + \mathrm{F}) = L(\mathrm{E}) \cup L(\mathrm{F})$
- $L(\mathrm{EF}) = L(\mathrm{E}).L(\mathrm{F})$
- $L(\mathrm{E}^*) = L(\mathrm{E})^*$
- $L((\mathrm{E})) = L(\mathrm{E})$

Alcuni esempi:

6) Tutte le stringhe w che contengono la sottostringa 101 (alfabeto 0,1)

Your regex: (0|1)*101(0|1)*

ER che mostra che tutte le stringhe contengano ciascun simbolo almeno una volta

$(a+b+c)^*a(a+b+c)^*b(a+b+c)^*c$

ER per stringhe binarie che contengono almeno tre 1:

(0+1)*1(0+1)*1(0+1)*1(0+1)*

ER per stringhe di testo che descriva le date in formato GG/MM/AAAA

0(1+2...+9)+(1+2)/(1+2+...9)+3(0+1)/(0+1)(1+2)/(0+1+...9)(0+1+...9)(0+1+...9)(0+1+...9)

7) Tutte le stringhe la cui lunghezza è multiplo di 3 (alfabeto a,b,c)

Your regex: ((a|b|c)(a|b|c)(a|b|c))*

Exercise: Give regular expressions that describe the following languages.

1. $L := \{w \in \{a,b\}^* \mid |w| \text{ divisible by } 3\}$
2. $L := \{w \in \{a,b,c\}^* \mid w \text{ does not contain } a, b, \text{ or } c\}$
3. $L := \{w \in \{a,b\}^* \mid \text{substring } ab \text{ occurs exactly twice in } w, \text{ but not at the end}\}$

Solution:

1. $((a+b)\cdot(a+b)\cdot(a+b))^*$
2. $(a+b)^* + (a+c)^* + (b+c)^*$
3. $b^*a^+b^+a^+b(b^+a^* + b^*a^+)$

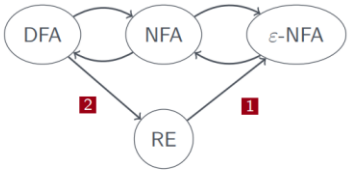
5) Tutte le stringhe che contengono 4k + 1 occorrenze di “b” per “k” >= 0
Soluzione:
((a|c)*b(a|c)*b(a|c)*b(a|c)*b(a|c)*)* (a|c)*b(a|c)*

3) Tutte le stringhe che NON contengono la sottostringa 101 (alfabeto {0,1})

Your regex: 0*(1*000*)*1*0*

Equivalenza tra FA e RE

Sappiamo già che DFA, NFA, e ε-NFA sono tutti equivalenti.



Gli FA sono equivalenti alle espressioni regolari:

1. Per ogni espressione regolare R esiste un ε-NFA A, tale che $L(A) = L(R)$
2. Per ogni DFA A possiamo costruire un'espressione regolare R, tale che $L(R) = L(A)$

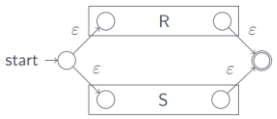
Caso Base:

- automa per ε 

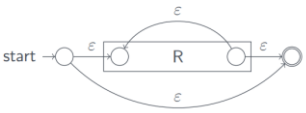
- automa per $\emptyset$ 

- automa per a 

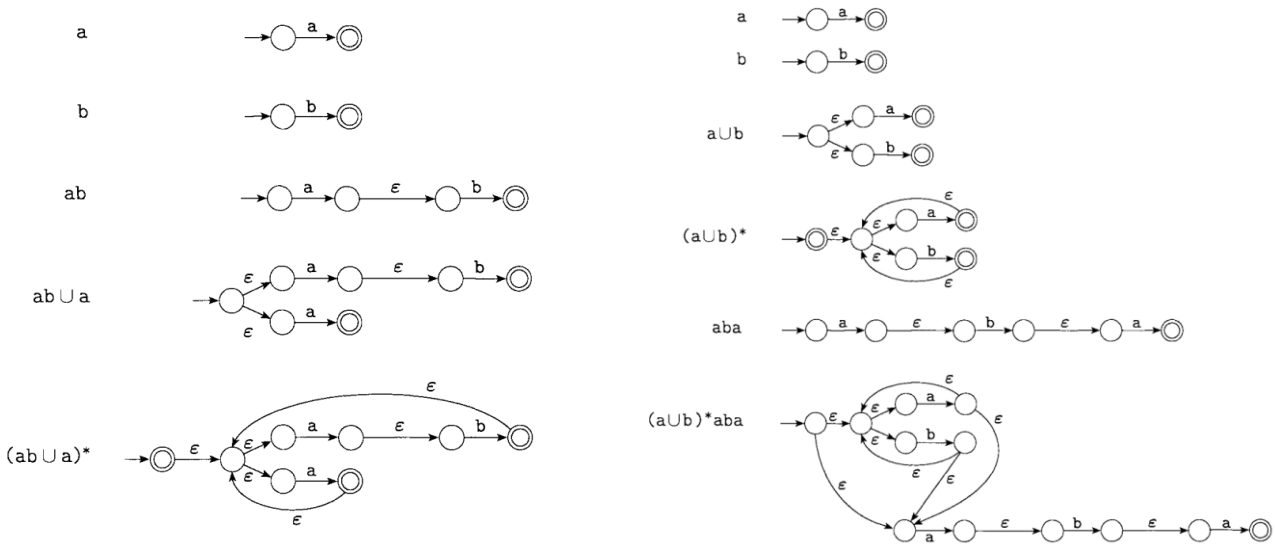
Caso Induttivo:

- automa per R + S 

- automa per RS 

- automa per R* 

Esempio completo di equivalenza:



Solve Regular expression to epsilon-NFA problem

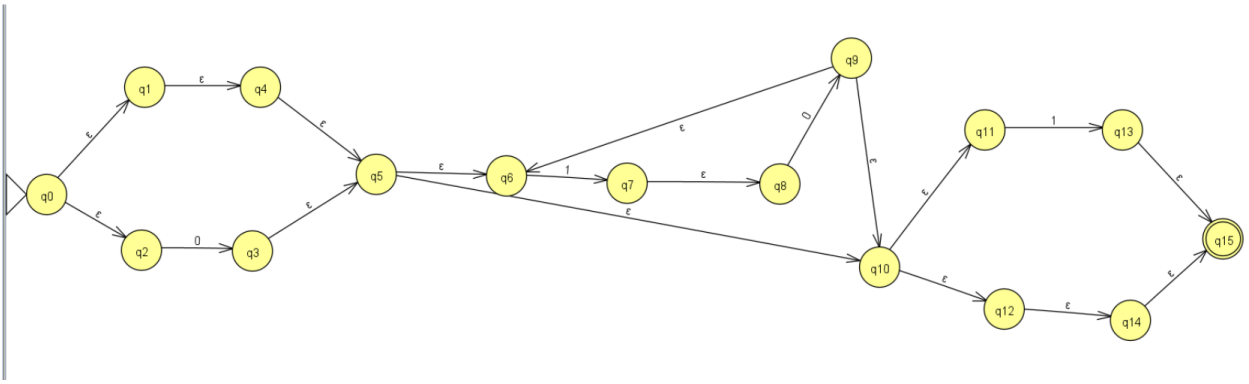
For the following regular expression:

$$(\epsilon|0)(10)^*(1|\epsilon)$$

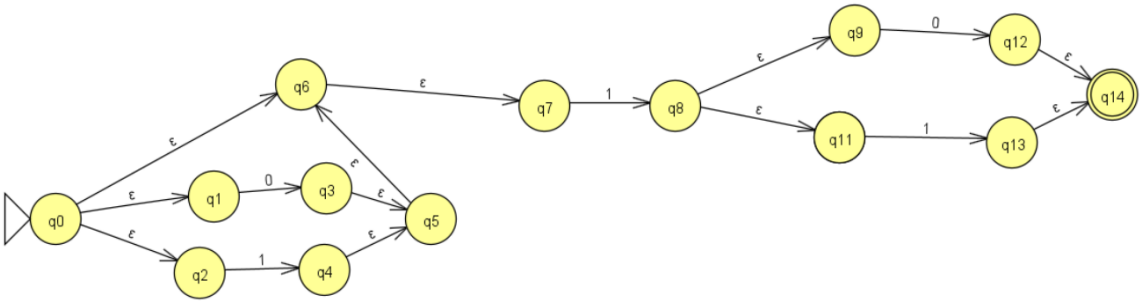
Over the alphabet:

$$\{0,1\}$$

Give an epsilon-NFA that recognizes the same language.



(0 + 1)∗1(0 + 1) in ε-NFA (da slide)

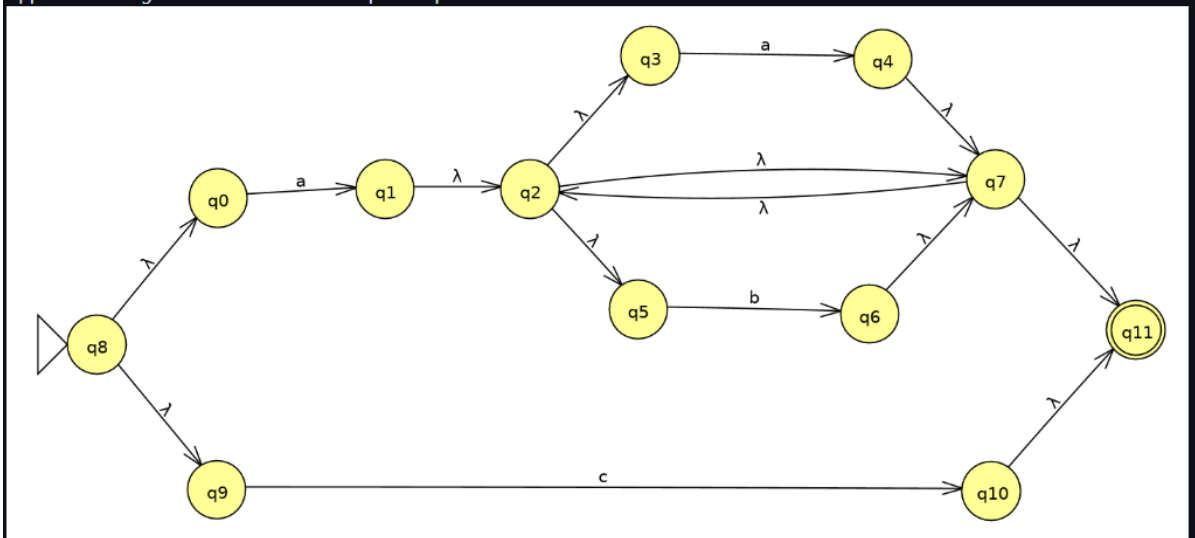


Soluzione: `a(a+b)*+c`

Esercizio 6.3

Convertire l'esercizio 6.2 in ε-NFA.

Applicando le regole si ottiene una cosa di questo tipo:

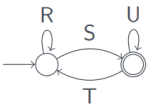


- La procedura che vedremo è in grado di convertire un qualsiasi automa (DFA, NFA, ε-NFA) in una espressione regolare equivalente
- Si procede per eliminazione di stati
- Quando uno stato q viene eliminato, i cammini che passano per q scompaiono
- si aggiungono nuove transizioni etichettate con espressioni regolari che rappresentano i cammini eliminati
- alla fine otteniamo un'espressione regolare che rappresenta tutti i cammini dallo stato iniziale ad uno stato finale
  - $\Rightarrow$ cioè il linguaggio riconosciuto dall'automa

- Lo stato da eliminare può avere un ciclo
- $q_1, \ldots, q_k$ sono i predecessori
- $p_1, \ldots, p_m$ sono i successori
- ci possono essere transizioni dirette tra i predecessori ed i successori

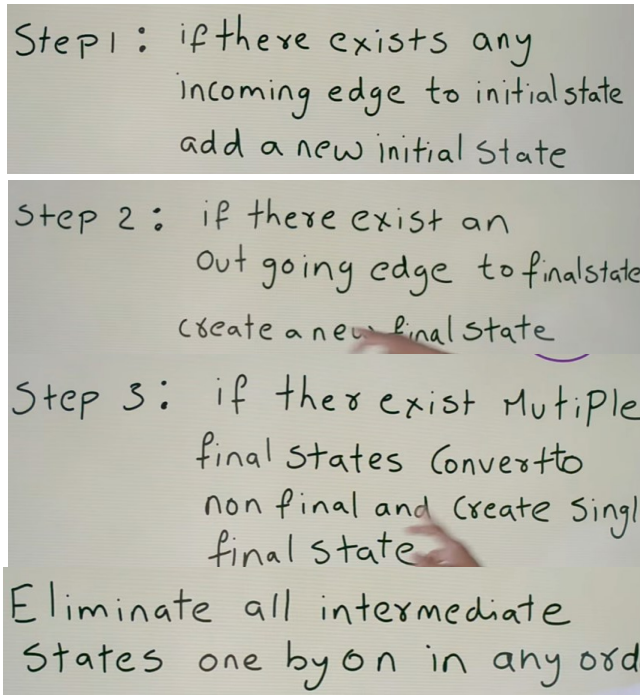
- Dobbiamo ricreare la transizione per ogni coppia predecessore-successore q_i, p_j
- Se non c'è la transizione diretta, l'etichetta è $Q_i S^* P_j$
- Se c'è la transizione diretta, l'etichetta è $R_{ij} + Q_i S^* P_j$

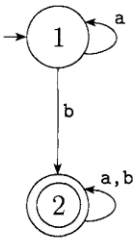
La strategia completa

1. l'automa deve avere un unico stato finale
   - se c'è più di uno stato finale, crea un nuovo stato finale q_f con ε-transizioni provenienti dai vecchi stati finali
2. collassa le transizioni tra la stessa coppia di stati
3. elimina tutti gli stati tranne lo stato iniziale e lo stato finale
4. se $q_f \neq q_0$ l'automa finale è 

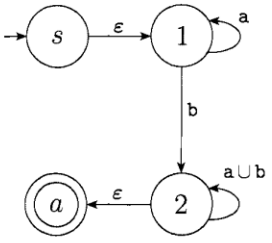
   che è equivalente a $(R + SU^*T)^*SU^*$
5. se $q_f = q_0$ l'automa finale è 

   che è equivalente a R^*

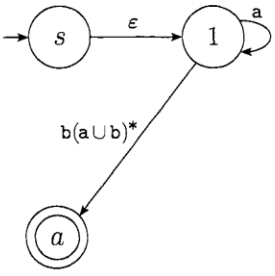




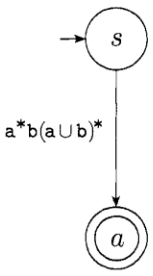
(a)



(b)



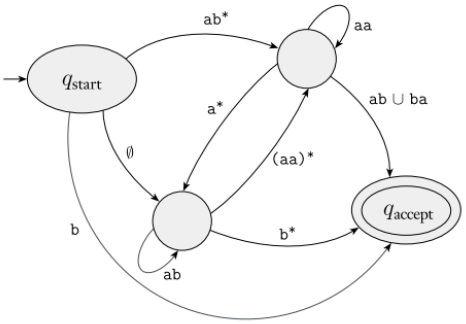
(c)



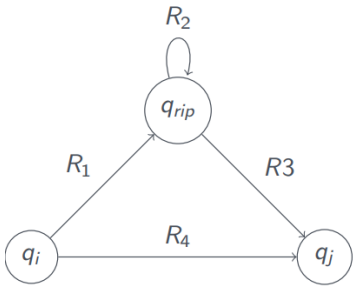
(d)

Sono NFA dove le transizioni sono etichettate con espressioni regolari

Ogni transizione consuma un blocco di simboli dall'input che appartiene al linguaggio dell'espressione regolare

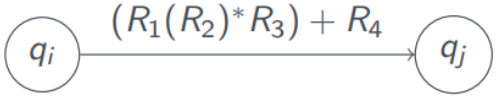


1. Nuovo stato iniziale q_{start} con transizione ε verso il vecchio q_0
2. Nuovo stato finale q_{accept} con transizione ε da tutti i vecchi stati finali $q \in F$
3. Rimpiazzo transizioni multiple tra due stati con l'unione delle etichette
4. Aggiungo transizioni etichettate con $\emptyset$ tra stati non collegati da transizioni



Se, nel GNFA:

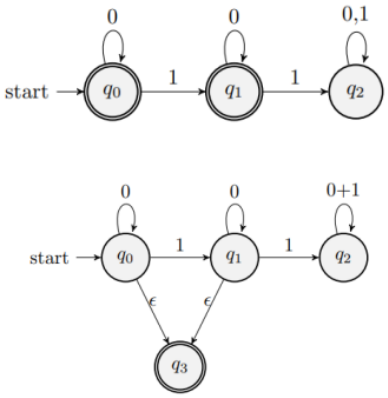
1. q_i va in q_{rip} con etichetta R_1
2. q_{rip} ha un self loop con etichetta R_2
3. q_{rip} va in q_j con etichetta R_3
4. q_i va in q_j con etichetta R_4



dopo l'eliminazione di q_{rip}, q_i va in q_j con etichetta

$$(R_1(R_2)^*R_3) + R_4$$

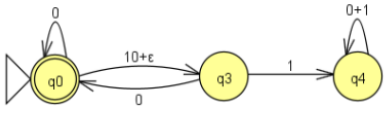
2. Stringhe binarie che non comprendono la stringa 101



1

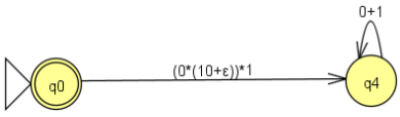
2

Passaggi:
Elimino q2 q0-q3 10+ε



3

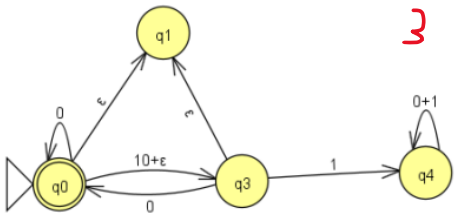
Elimino q3: q0-q4 (0*(10+ε))*1



Elimino q4: (0*(10+ε))*1+(0+1)*

ER: 0*(ε+(10+ε))1(0+1)*

6

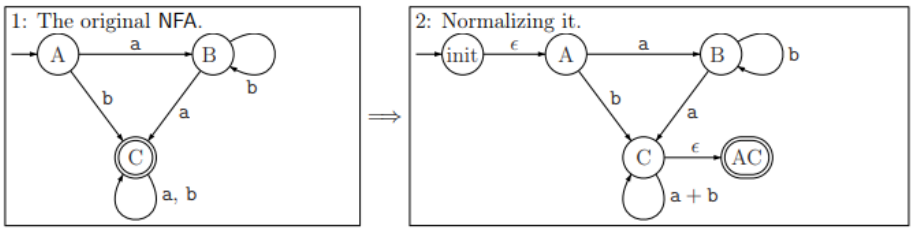


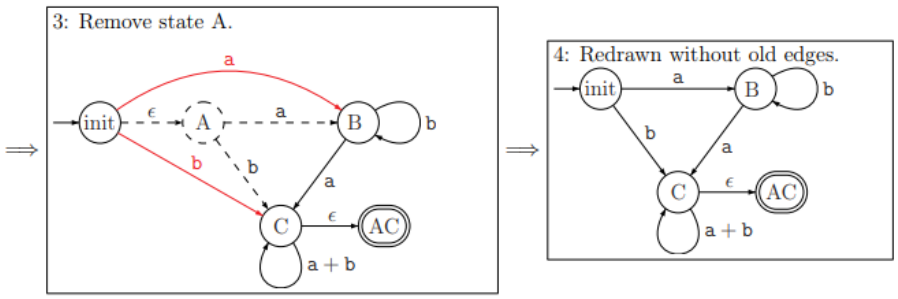
3

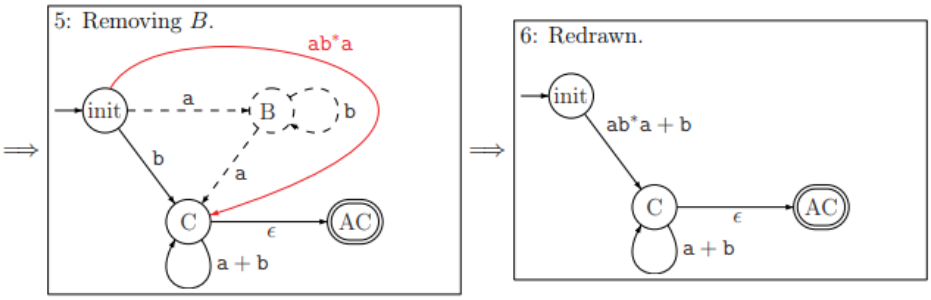
4

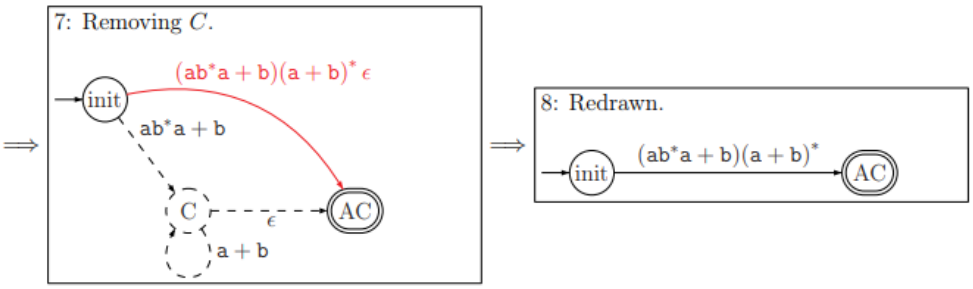
Elimino q1: q0-q3-q4 (10+ε)

2.1 Example: From GNFA to regex in 8 easy figures









Thus, this automata is equivalent to the regular expression $(ab^*a + b)(a + b)^*$.

Costruiamo l'espressione regolare equivalente al seguente *NFA*: