Unit - II

Regular Expression and Languages

Regular Expressions :-

* The language accepted by finite automata can easily described by simple Expression called the Regular Expressions.
* It is most effective way to represent any language.
* The language accepted by some regular expressions are referred to as Regular languages.
* A Regular Expression can also be described as a sequence of patterns that defines a string.

for instance,

→ In Regular Expression x^* means zero or more occurance of x

It can generate $\{\epsilon, x, xx, xxx, xxxx, \ldots\}$

→ In Regular Expression x^+ means one or more occurance of x

It can generate $\{x, xx, xxx, xxxx, \ldots\}$

Operations on Regular Language :-

i) Union :- If L & M are two Regular Language then their union $L \cup M$ is also a union

$$L \cup M = \{s \mid s \text{ is in } L \text{ or } s \text{ is in } M\}$$

ii) Intersection :- If L and M are two regular Language then their intersection is also an intersection

$$L \cap M = \{st \mid s \text{ is in } L \text{ and } t \text{ is in } M\}$$

iii) Kleen Closure :- If L is a regular language then its kleen closure L^* is also be a regular language.

L^* = Zero or more ocurance of language L

Example 1 : Write Regular Expression for the language accepting all strings containning any number of a's and b's

Solution :- $R.E = (a+b)^*$

This will given the set as $L = \{\varepsilon, a, aa, b, bb, ab, ba, aba \ldots\}$

⇒ The $(a+b)^*$ shows any combination with a & b even a null string

Example 2 : Write Regular Expression for the language accepting all the string which are starting with 1 and ending with 0, over $\Sigma = \{0, 1\}$

Solution :- First Symbol is 1

Last Symbol is 0

$\therefore R.E = 1(0+1)^*0$

⇒ $L = 10, 1000, 1100, 1010, \ldots$

Example 3 : Write the Regular Expression for the language accepting all combinations of a's, over the set $\Sigma = \{a\}$

Solution : All the combination of a's means, a may be zero, single, double and so on.

⇒ $L = \{\varepsilon, a, aa, aaa, aaaa, \ldots\}$

$\therefore R.E = a^*$

Example 4: Write the Regular Expression for the language accepting all combination of a's except the null string, over the ~~string~~ set $\Sigma = \{a\}$

Solution:- The regular expression has to be built for the language,

$$L = \{a, aa, aaa, aaaa, \ldots\}$$

Unit set indicates no null string,

$$\therefore R.E = a^+$$

Example 5: Write the Regular Expression for the language starting with a but not having consecutive b's.

Solution: $L = \{a, aba, aab, aaa, abab, \ldots\}$

$$\therefore R.E = \{a+ab\}^*$$

Example 6: Write the Regular Expression for the language over $\Sigma = \{0\}$ having even length of the string.

Solution: $L = \{\varepsilon, 00, 0000, 000000, \ldots\}$

$$\therefore R.E = (00)^*$$

Example 7: Write the Regular Expression for the language L over $\Sigma = \{0, 1\}$ such that all the string do not contain the substring 01.

Solution: $L = \{\varepsilon, 0, 1, 00, 11, 10, 100, \ldots\}$

$$\therefore R.E = (1^*0^*)$$

Note: Language associated with the Regular Expression r is denoted as $L(r)$

If r_1 and r_2 are regular expressions, then

$$L(r_1 + r_2) = L(r_1) + L(r_2)$$
$$L(r_1 . r_2) = L(r_1) . L(r_2)$$
$$L(r_1)^* = (L(r_1))^*$$

Example 8: $\Sigma = \{a, b, c, d\}$

Check whether $(a+b)^*(cd)$ is a regular expression?

Solution: Let $r = (a+b)^*(cd)$

r can be constructed from rules,

Let $r_1 = a$ and $r_2 = b$

$\Rightarrow r_3 = r_1 + r_2$

$= a+b$ is a regular expression

$(r_3)^*$ is also a regular expression

i.e, $(a+b)^*$ is regular expression

Let $r_4 = c$ and $r_5 = d$

$r_6 = r_4 . r_5$

$= cd$ is also a regular expression

Hence, $(a+b)^*(cd)$ is a regular expression

Example 9:

$$L(a^*.(a+b)) = L(a^*) \cdot L(a+b)$$
$$= L(a^*) . L(a) + L(b)$$
$$= (L(a))^* . L(a) + L(b)$$
$$= \{\varepsilon, a, aa, aaa, \ldots\} . \{a, b\}$$
$$= \{a, aa, aaa, aaaa, \ldots, b, ab, aab, \ldots\}$$

Equivalence of finite Automata and Regular Expressions :-

Given a regular expression, there is an associated regular language $L(r)$.

The equivalence of finite automata and regular expressions can be proved by two parts,

First part ---> Finite Automata from its Regular Expression.

Second part ---> Regular Expression from finite automata.

Case I : Kleen's theorem, Part I

For every regular expression, there exist a machine M which accepts the regular language.

Case II : Kleen's theorem, Part II

The language accepted by finite automata is regular.

⇒ Constructing Finite Automata from Regular Expression :

Kleen's theorem Part I :

For a Regular Expression, there exists a NFA - ϵ transitions that accepts $L(r)$.

Proof :

The proof is done by induction, based on the number of operators used in the regular expression.

Basis (zero operators) : The regular expression here is ϵ, ϕ (r) for some Symbol 'a' in Σ.

The NFA statisfying the conditions are,

(i) $r = \epsilon$ ⟶ 

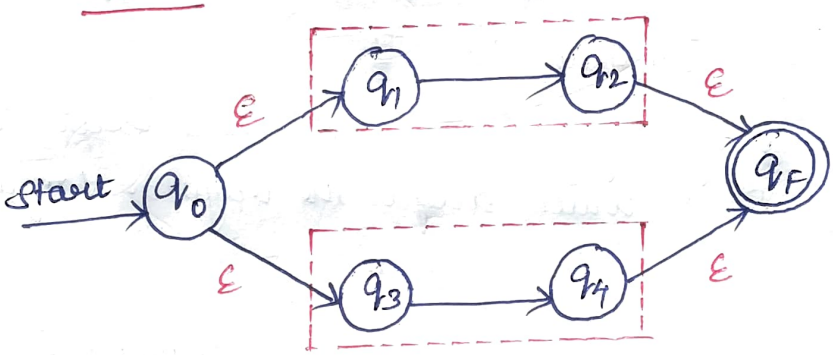
(ii) $r = a$

(iii) $r = \phi$

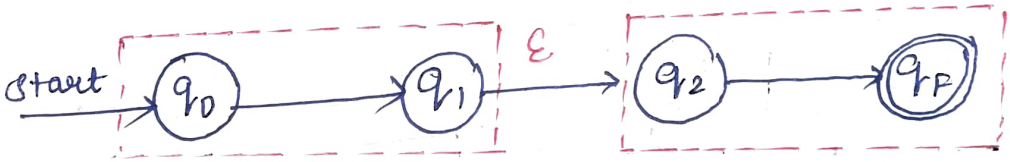
Induction (one (or) more operators)

Here, we are considering three cases.

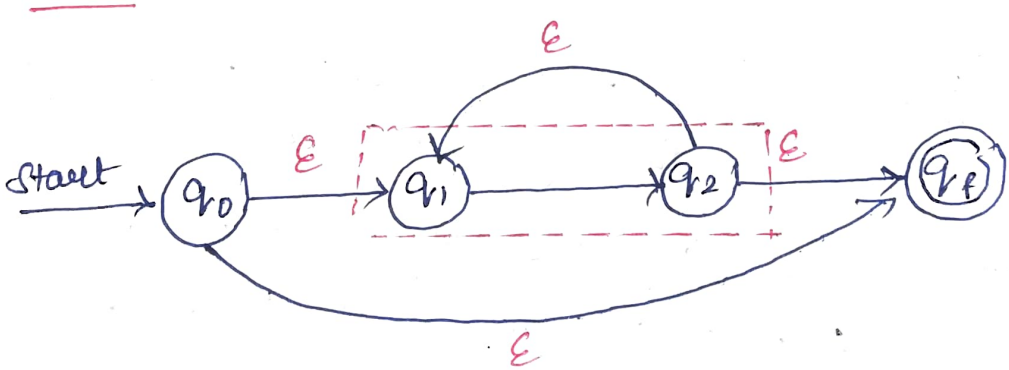
Case 1: $r = r_1 + r_2$



Case 2: $r = r_1 . r_2$



Case 3: $r = r_1^*$



Example 1 :-

Construct NFA for the regular expression $r = 1^*0+0$

Solution :

The regular expression r can be splitted into small components as,

$r = r_1 + r_2$, where

$r_1 = 1^*0$

$r_2 = 0$

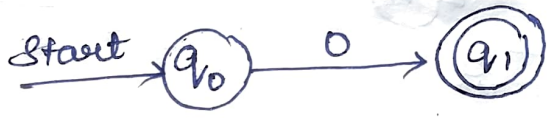
The regular expression r_1 can be splitted again as

$r_1 = r_3 r_4$, where

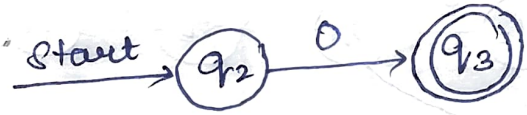
$r_3 = 1^*$

$r_4 = 0$

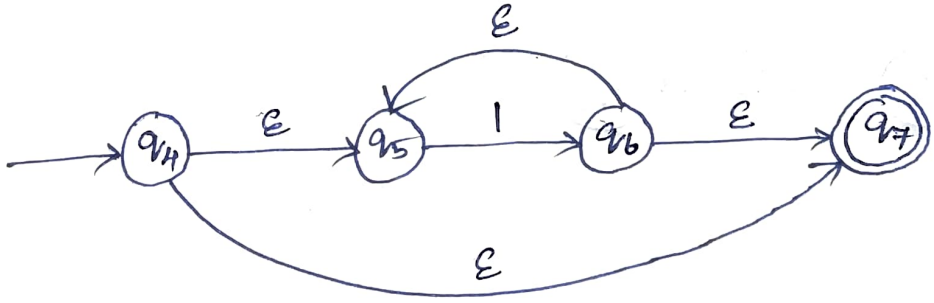
Automation for $r_2 = 0$:-



Automation for $r_4 = 0$:-

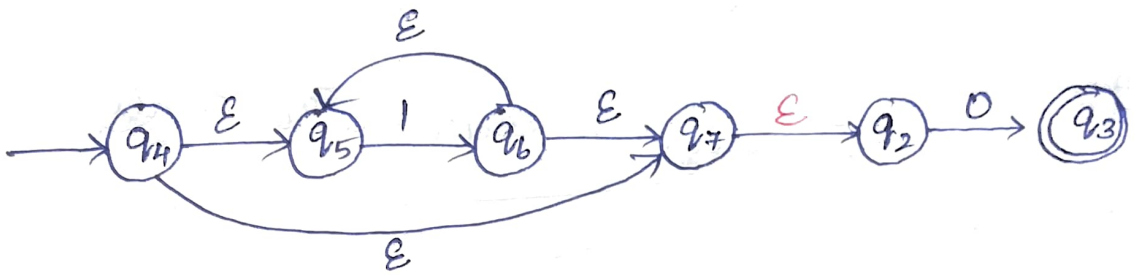


Automation for $r_3 = 1^*$:

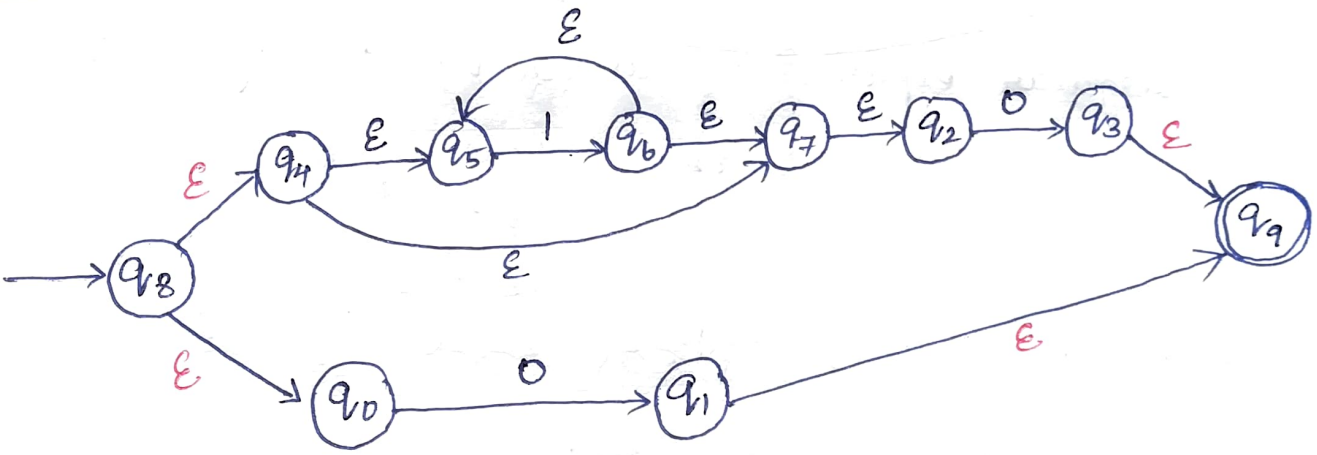


Now, let us construct a automation for $r_1 = r_3 r_4$, ~~Eq~~ such that $r_1 = 1^*0$.

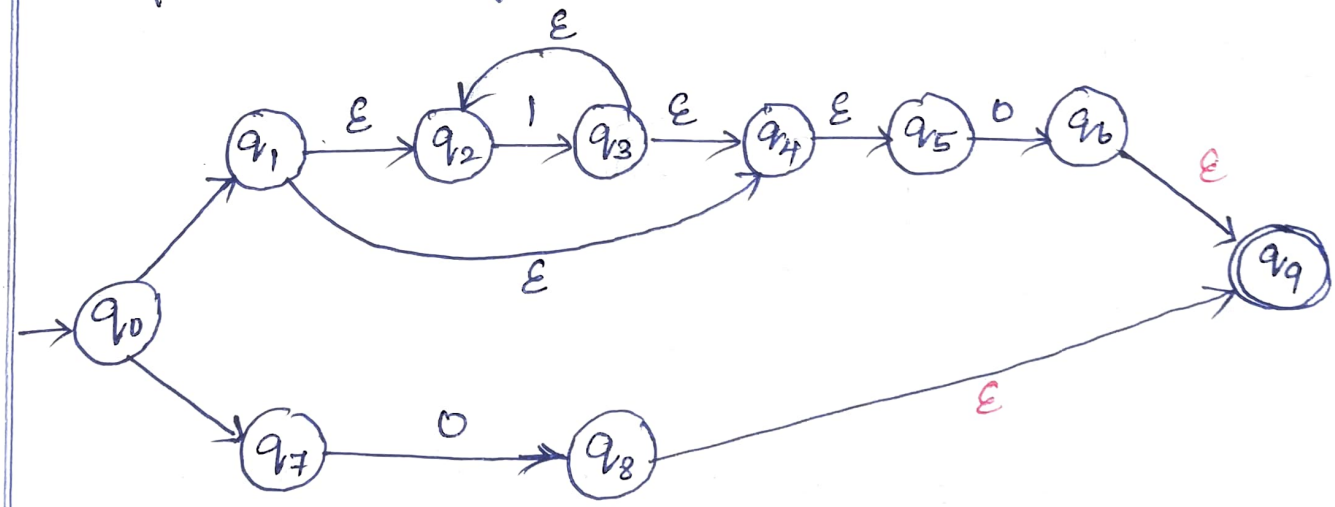
Automation for $r_1 = r_3 r_4 = 1^*0$, concatinating r_3 and r_4



Automation for $r_1 + r_2 = 1^*0 + 0$



The final state diagram is,



Example 2 :

Construct NFA to accept the language indicated by the regular expression $(01)^*(10)^* + 00^*$

Solution : $r = (01)^*(10)^* + 00^*$

The R.E r can be splitted into,

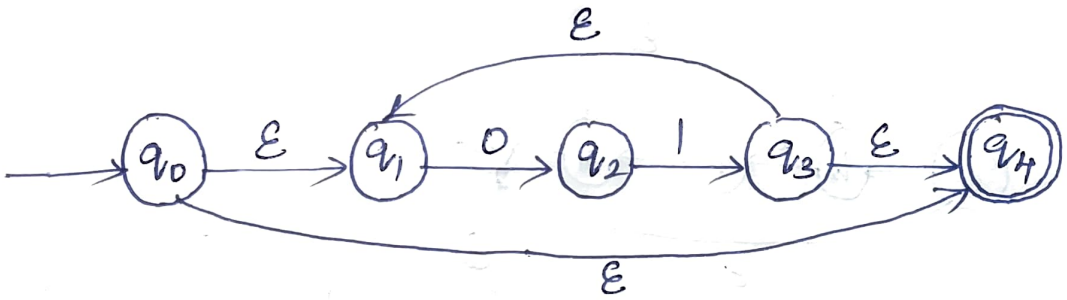
$r = r_1 r_2 + r_3$, where

$\Rightarrow r_1 = (01)^*$

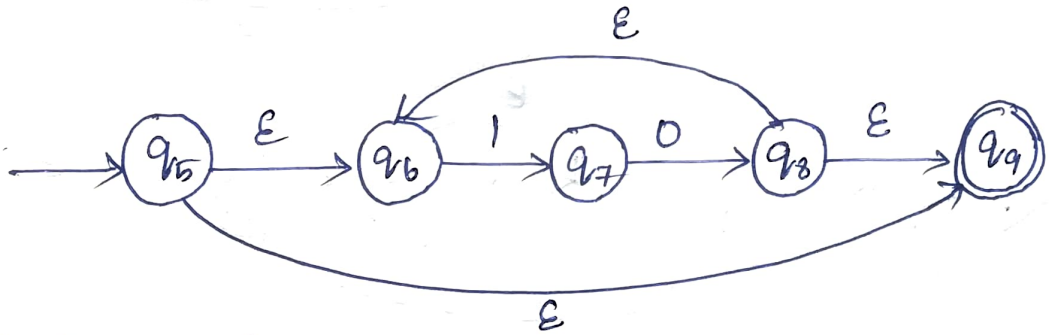
$r_2 = (10)^*$

$r_3 = 00^*$

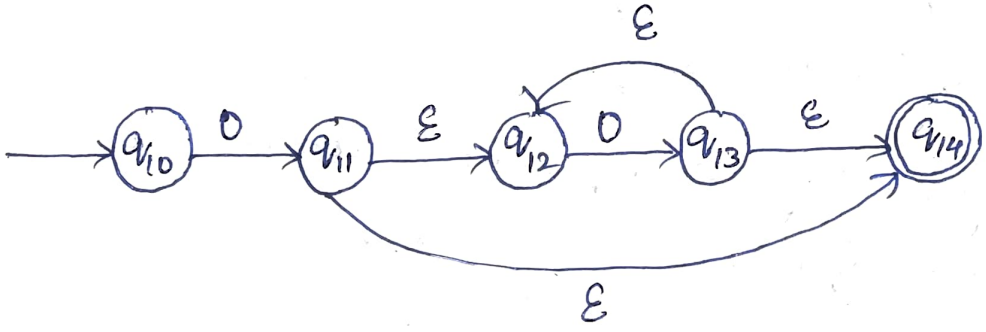
Automation for $r_1 = (01)^*$



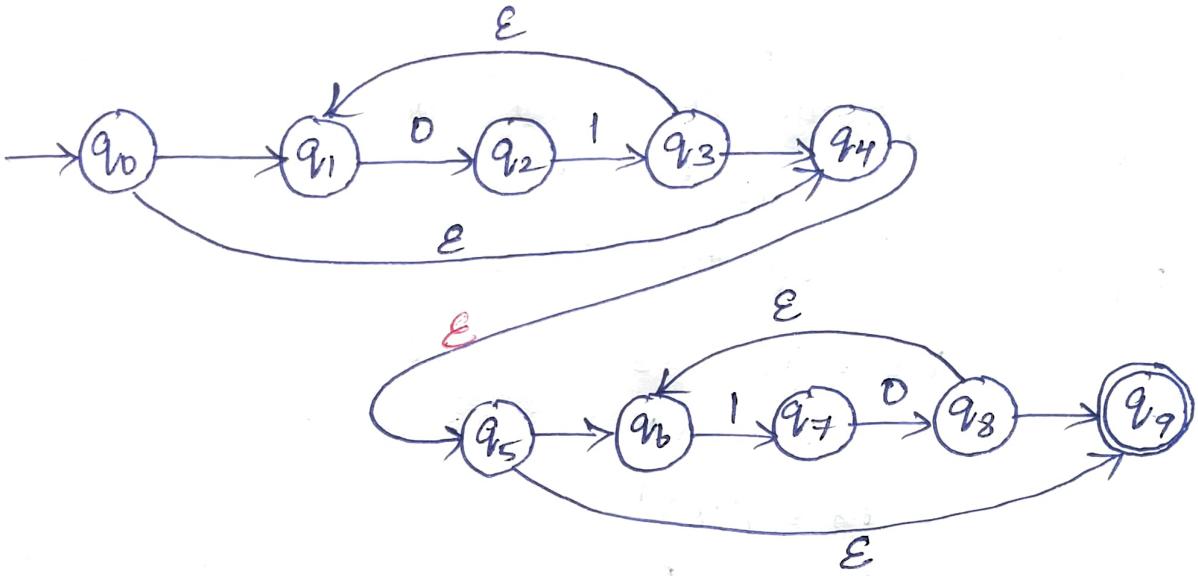
Automation for $r_2 = (10)^*$



Automation for $r_3 = 00^*$

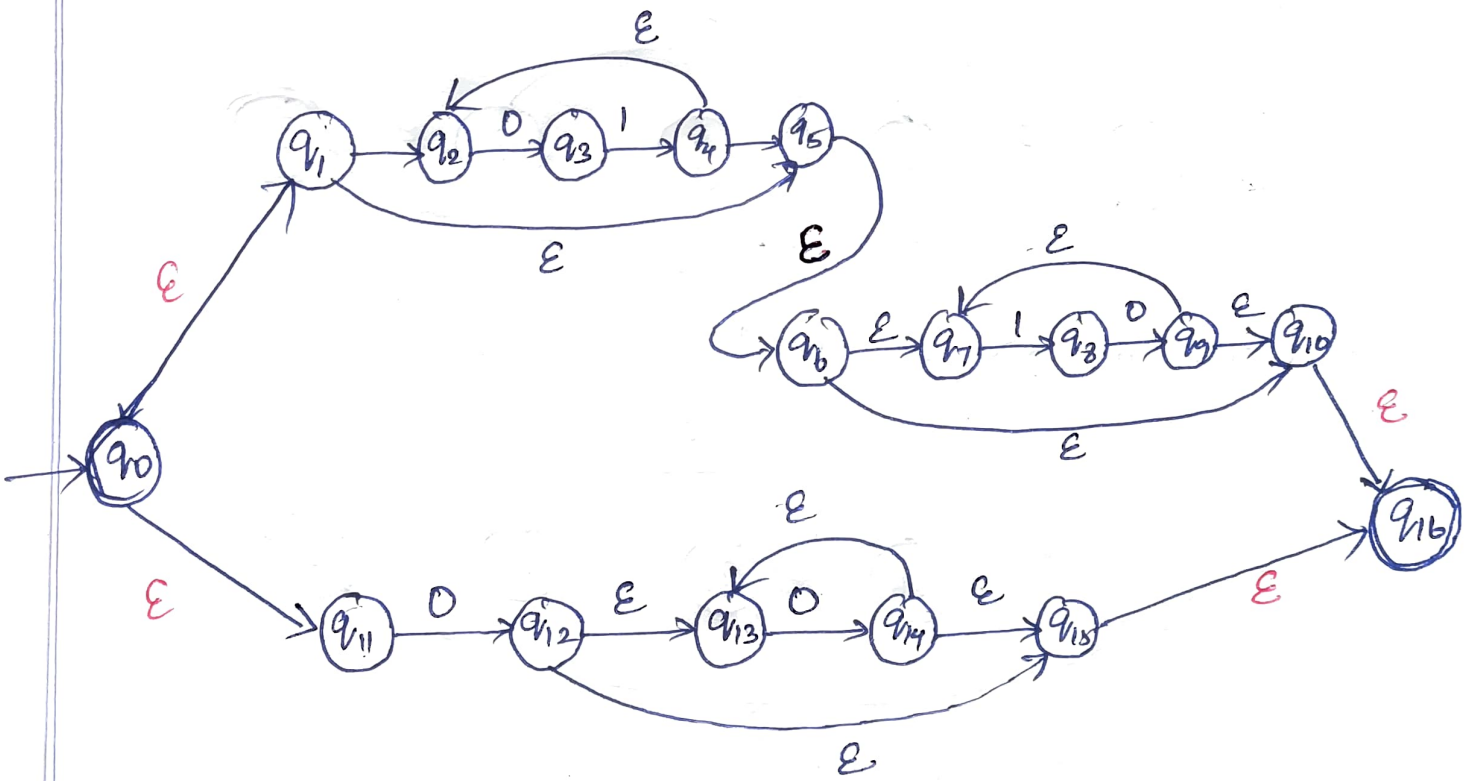


Automation for $r_1 r_2 = (01)^*(10)^*$



Automation for ~~$r = r_1 + r_2$~~ $r = r_1 r_2 + r_3$

i.e, $(01)^*(10)^* + 00^*$



Here, Starting state → q_0

Ending state → q_{11}

Hence, the given regular expression is converted into Non-deterministic finite automata (NFA)

Conversion of Regular Expression into FA :

(Another method)

" Subset Method "

Step 1 : Design a transition diagram for given Regular Expression, using NFA with ε-moves.

Step 2 : Convert NFA with ε to NFA without ε

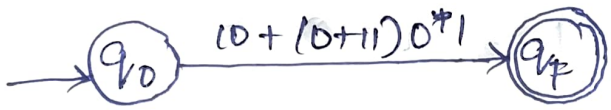
Step 3 : Convert the obtained NFA to equivalent DFA.

Example: Design a FA from given R.E $10 + (0+11)0^*1$

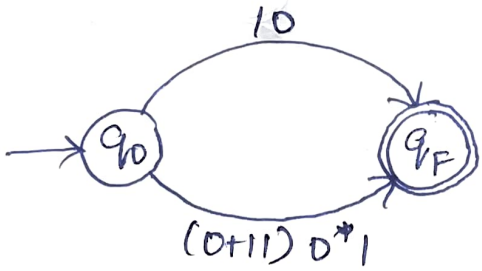
Solution:

The given regular expression is $10 + (0+11)0^*1$

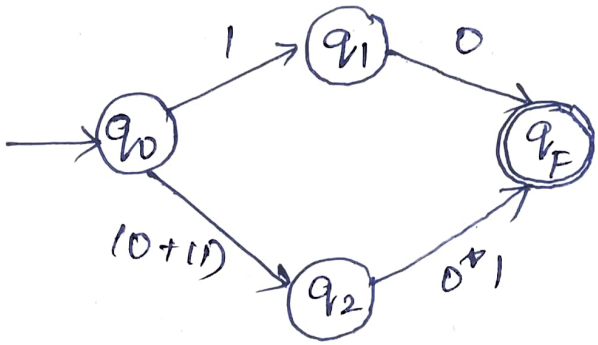
Step 1 :



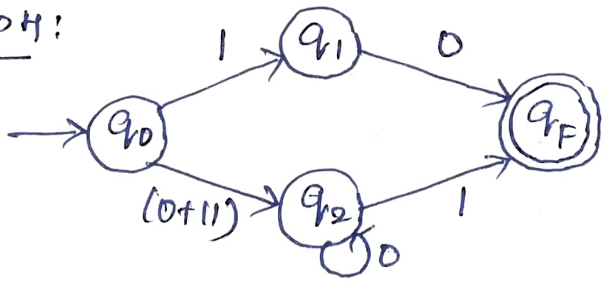
Step 2 :



Step 3 :

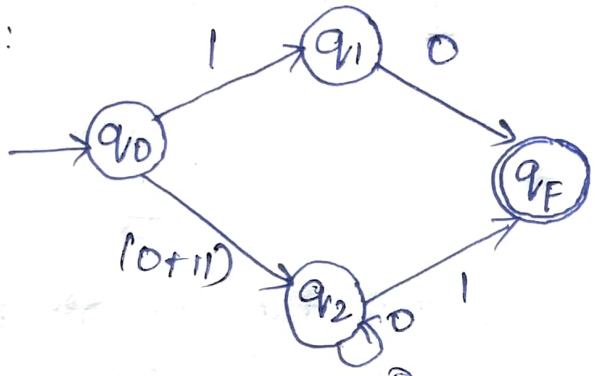


Step 4 :

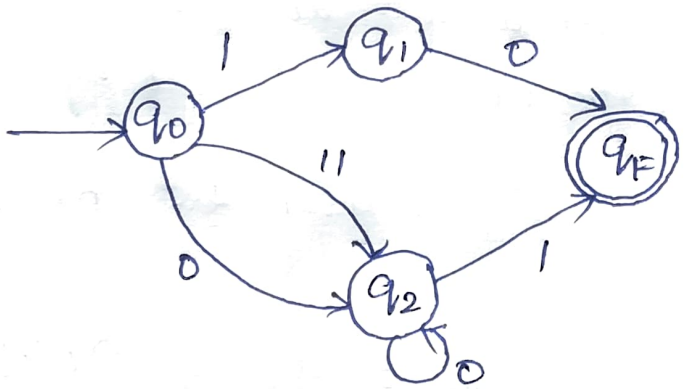


RE
↓
NFA - ε
↓
NFA
↓
DFA

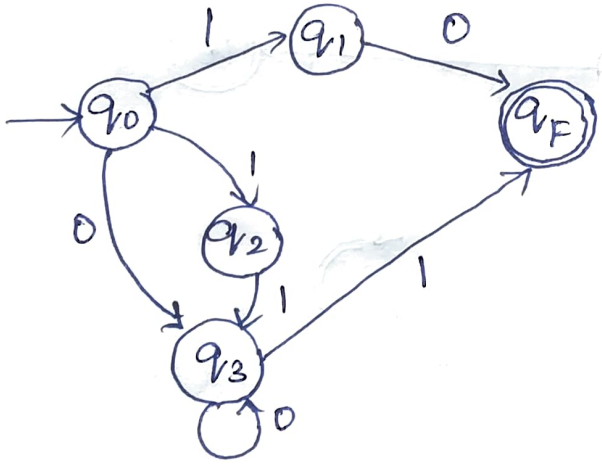
Step 5:



It can be converted into,



Step 6:



Now, we have to convert the NFA into equivalent DFA

State	0	1
$\rightarrow q_0$	q_3	$\{q_1, q_2\}$
q_1	q_F	ϕ
q_2	ϕ	q_3
q_3	q_3	q_F
$*q_F$	ϕ	ϕ

→ Transition table for NFA

Now, the equivalent DFA is,

State	0	1
$\rightarrow[q_0]$	$[q_3]$	$[q_1,q_2]$ ----> New Single state
$[q_1]$	$[q_F]$	ϕ
$[q_2]$	ϕ	$[q_3]$
$[q_3]$	$[q_3]$	$[q_F]$
$[q_1,q_2]$	$[q_F]$	$[q_F]$
$*[q_F]$	ϕ	ϕ

Pumping Lemma for Regular Languages :-

- The Language accepted by Finite Automata is known as Regular Language.
- Pumping Lemma is one of the property of Regular Language.
- It is a method to prove that certain Language are not regular.

Pumping | Lemma

Pumping: Generating many input strings by pushing a symbol in an input string repeatedly

Lemma: Intermediate theorem in a proof

- There are two pumping lemma, they are defined for
  i) Regular Language
  ii) Context free Language

Theorem:

Let L be a regular language. Then there exists a constant n such that for every string w in L,

$$|w| \geq p$$

we can break w into three strings, $w = xyz$ such that

(i) $|y| > 0$

(ii) $|xy| \leq p$

(iii) $\forall k \geq 0$, the string $xy^k z$ is also in L

[Note : $\forall$ means for all]

Applications of Pumping Lemma :-

Pumping lemma is to be applied to show that certain languages are not regular. It should never be used to show a language is regular.

- Assume that the language is Regular.
- If L is regular, it satisfies pumping lemma.
- If L does not satisfy pumping lemma, it is not regular.

To prove that a language is not regular using the pumping lemma, follow the steps :

→ At first, we have to assume that L is regular.

→ So, the pumping lemma should hold for L.

→ It has to have a pumping length (say p).

→ All strings longer than p can be pumped,

$$|w| \geq p$$

→ Now find a string 'w' in L such that $|w| \geq p$
→ Divide w into xyz
→ Show that $xy^iz \notin L$ for some i
→ Then consider all ways that w can be divided into xyz.
→ Show that none of them can satisfy all the three pumping conditions at the same time.
→ w cannot be pumped == contradiction.

Example 1: Prove that $L = \{a^i b^i \mid i \geq 0\}$ is not regular.

Solution:

1) At first, we assume that L is regular and n is the number of states.
2) Let $w = a^n b^n$. Thus $|w| = 2n \geq n$
3) By pumping lemma,
   Let $w = xyz$, where $|xy| \leq n$
4) Let $x = a^p$, $y = a^q$, $z = a^r b^n$
   where $p+q+r = n$ $(p \neq 0, q \neq 0, r \neq 0)$
   Then $|y| \neq 0$
5) Let $k = 2$
   Then $xy^2z = a^p a^{2q} a^r b^n$
6) Number of a's $= (p+2q+r)$
   $= (p+q+r)+q$
   $= n+q$
7) Hence, $xy^2z = a^{n+q} b^n$. Since $q \neq 0$
   $\Rightarrow xy^2z$ is not of the form $a^n b^n$
8) Thus xy^2z is not in L
   Hence, L is not regular //

Example 2: Show that $L = \{a^n b^n \mid n \geq 0\}$ is not regular.

Proof:

Assume that A is Regular and has a pumping length = p

Let a string $s = a^p b^p$

Now divide the s into parts, xyz

To divide the s, let's take the value of $p = 7$

$\Rightarrow s = aaaaaaabbbbbbb$

↳ by putting $p = 7$ in $s = a^p b^p$

Case 1: y consits of a string having the letter only 'a'

$\underbrace{aa}_{x}\ \underbrace{aaaa}_{y}\ \underbrace{abbbbbbb}_{z}$

Case 2: y consists of a string having the letter only 'b'

$\underbrace{aaaaaaabb}_{x}\ \underbrace{bbbb}_{y}\ \underbrace{b}_{z}$

Case 3: y consits of a string with the leeters 'a' and 'b'

$\underbrace{aaaaa}_{x}\ \underbrace{aabb}_{y}\ \underbrace{bbbbb}_{z}$

For all the above cases, we need to show $xy^i z \notin A$ for some i.

Let the value of $i = 2$,

$xy^i z = xy^2 z$

In case1, $xy^2 z$ = aa aaaa aaaa a bbbbbbb

$\Rightarrow$ Number of 'a' = 11 and 'b' = 7

Thus, Number of 'a' != Number of 'b'

But the original language has an equal number of 'a' and 'b'. Therefore, this string is not like in our language

In case 2, xy^2z = aaaaaaa bb bbbb bbbb b

Number of 'a' = 7 and Number of 'b' = 11

Since, Number of 'a' != Number of 'b'

But the original language has an equal number of 'a' and 'b'. Therefore, this string will not like in our language.

In case 3, xy^2z = aaaa aabb aabb bbbbb

Number of 'a' = 8 and 'b' = 9

Since the No of 'a' ! = Number of 'b'

Therefore, this string will not like in our language.

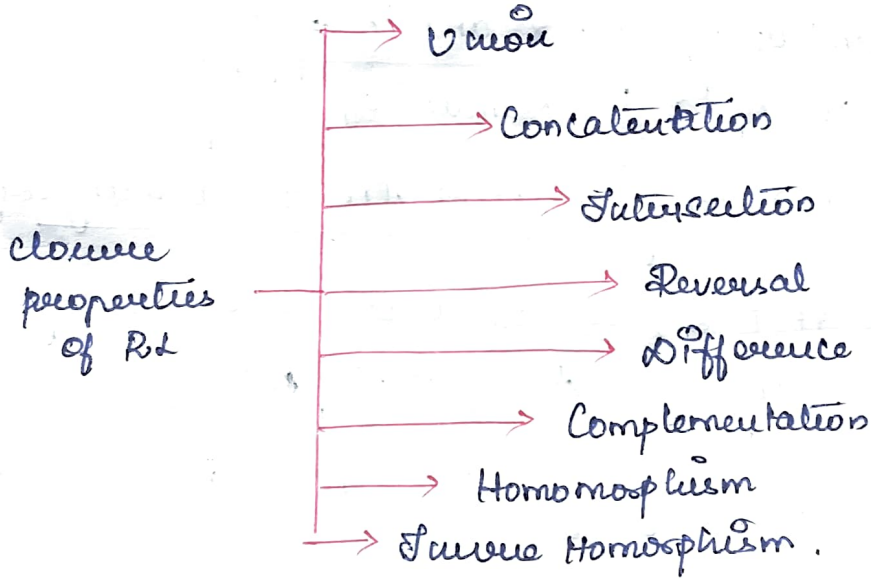
We can see at i=2, all the above three strings do not like in the language $A = \{a^n b^n \mid n \geq 0\}$

$\therefore$ The language $A = \{a^n b^n \mid n \geq 0\}$ is not regular

Closure Properties of Regular Languages :-

A Regular language may be expressed using Regular expressions, non-deterministic (or) deterministic finite automata (or) State Machines. A language is a collection of strings composed from a specific alphabet (or) set of symbols.

When we talk about groups of objects, we use "closure". If we have two regular languages, L_1 and L_2 and L is generated by performing specific operations on L_1 and L_2, the L is also regular.

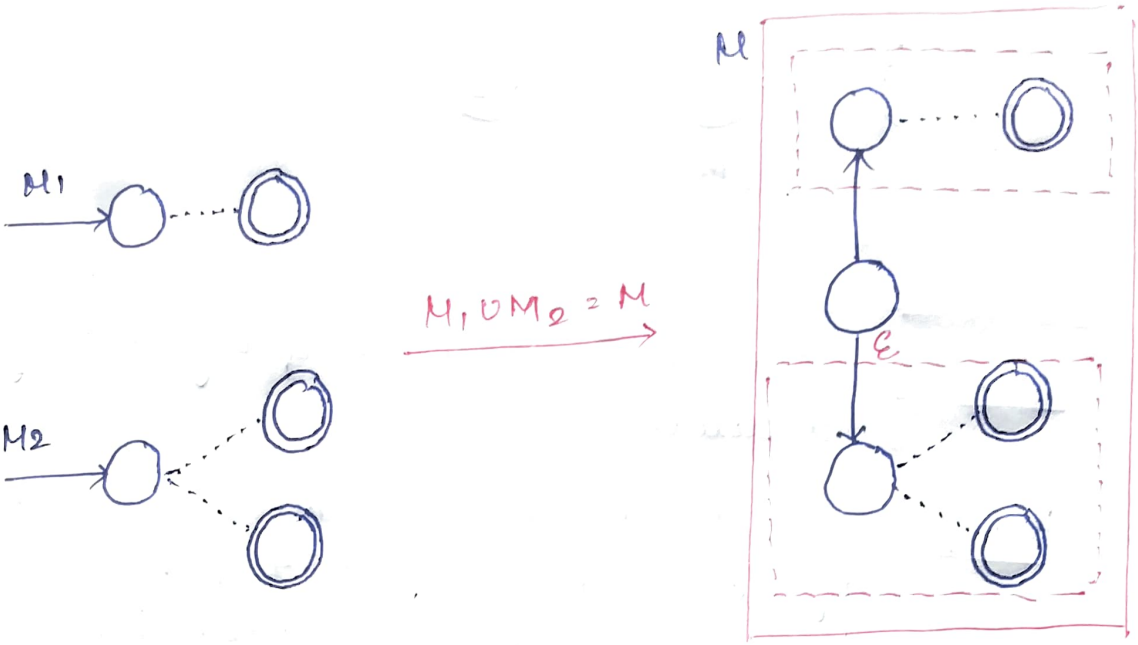


1) Union :-

Statement: If L_1 and L_2 are regular languages, then their union $L_1 \cup L_2$ is also a regular language.

Proof: Let M_1 and M_2 be two finite automata that accept the L_1 and L_2 regular languages respectively. If we wish to demonstrate that the union of L_1 and L_2 is likewise a regular language, we may do the following :

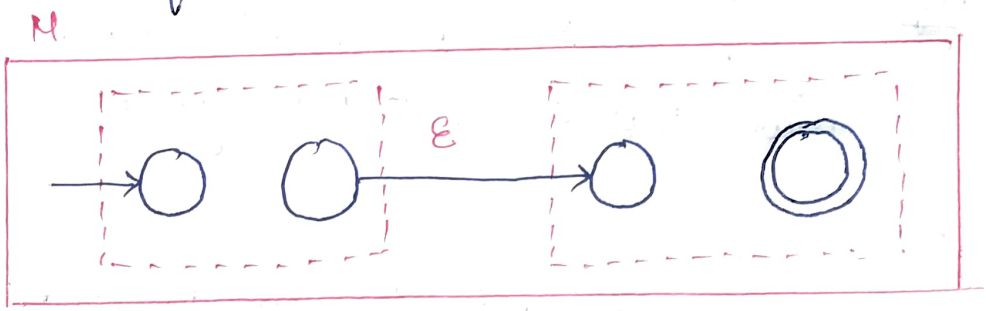
- Make a new starting point
- Make λ-transitions from the new state to each of M_1 and M_2 original s.



ii) Concatenation :

Statement : Concatenation of two regular languages is also regular.

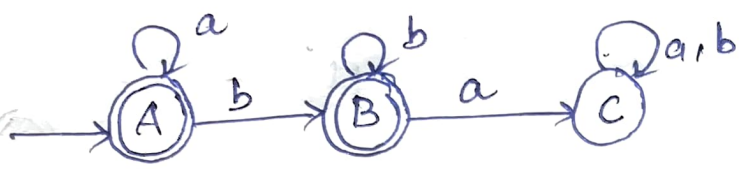
Proof : Let M_1 and M_2 be two finite automata, and L_1 and L_2 be the languages that M_1 and M_2 accept, respectively. We aim to demonstrate that $L_1 L_2 = L$. i.e, their concatenation yields regular language. Let M be a finite automation composed of M_1 and M_2



iii) Intersection :

Statement : If L and M are regular languages, then $L \cap M$ is also

Proof : Assume L_1 and L_2 are regular languages, and we wish to demonstrate that the intersection of L_1 and L_2 is likewise a regular language. Demorgen's law may be used to calculate the intersection of languages L_1 and L_2.



(M)

iv) Reversal :-

Statement : Under Reversal, the set of regular languages is closed.

Proof : Let M be a deterministic finite automata that accepts L; we will create M' from M so that M' ~~and the final~~ state are same. Make the final state of M the accepting state of M' and the final state of M the beginning state of M'.

v) Difference :

Statement : If L1 and L2 are regular languages, then L1 - L2 is as well, which implies all the strings in L1 but not in L2.

Proof : Since L_2 is regular, its complement L_2' is also regular, and $L_1 \cap L_2'$ too is regular, this proves that L1 - L2 is also regular.

vi) Complementation :

Statement : According to the theorem, the complement of two regular languages is also regular.

Proof : If M is a DFA that accepts L, we may write $L = L(M)$, then $\bar{L} = L(M_1)$

The DFA M_1 is similar to M, except that M's accepting states are now M_1's non-accepting state and vice versa.

Theorem 1:

The regular sets are closed under union, concatenation and Kleen closure.

$$r_1 \cup r_2 = r_1 + r_2$$
$$r_1 . r_2 = r_1 r_2$$
$$(r)^* = r'$$

Theorem 2:

The class of regular sets are closed under complementation.

Theorem 3:

If L_1 and L_2 are regular sets, then $L_1 \cap L_2$ is also regular.

Proof:

$$L_1 \cap L_2 = \overline{(\overline{L_1} \cup \overline{L_2})} \text{ by Demorgan's Law.}$$

If L_1 and L_2 are regular, then

→ $\overline{L_1}$ and $\overline{L_2}$ are also regular (By theorem 2)

→ $\overline{L_1} \cup \overline{L_2}$ are regular (By theorem 1)

$\therefore \overline{(\overline{L_1} \cup \overline{L_2})}$ is also regular // (By theorem 2)

Example:

If $L_1 = (a+b)^* a$
$L_2 = b(a+b)^* b$
} L_1 and L_2 are languages.

then, $\boxed{L_1 \cap L_2 = b(a+b)^* a}$

Hence, it is proved by thm 3.

Theorem 4:

The regular set is closed under substitution

Theorem 5 :

If L is regular then L^T is also regular.

Demorgan's law :-

Theorem 1 $\rightarrow$ $\overline{A \cdot B} = \overline{A} + \overline{B}$

Theorem 2 $\rightarrow$ $\overline{A + B} = \overline{A} \cdot \overline{B}$