2021

Formal Language and Automata Theory

* **Formal language :-** (which has some rules)
Formal language consists of words whose letters are taken from an Alphabet and are well formed according to a specific set of rules.

* **Automata Theory :-**
Automata Theory is the study of abstract machines and automata, as well as the computational problems that can be solved using them.

Abstract :- theoretical model of real computer

Automata :- Self acting

* **Applications of FLAT :-**
AI, Embedded systems, Compilers, spam and malware, pattern matching.

* **Formal languages :-**
1) Regular language — FA (finite automata)
2) Context free language — PDA (push down automata)
3) Context sensitive language — LBA (linear bounded automata)
4) Recursively enumerable language — Turing machine.

* **Alphabet :-**
It is denoted by Σ and defined as non empty finite set of symbols.

$\Sigma = \{0,1\}$

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

$\Sigma = \{0,1,a,b\}$

$\Sigma = \{A, B, \ldots Z, a, b, c, \ldots z\}$

$\Sigma = \{a, b, c, \ldots z, A, B, C, \ldots Z, \{, \}, (,) \ldots\}$

Eg: Natural numbers is not an Alphabet as it is ∞ set.

* **String** :- Sequence of symbols from some alphabet is known as a string.

$\Sigma = \{0,1\}$

string = 101010

length = 6

$\Sigma = \{a, b, c, d, \ldots z\}$

string = naghma

length = 6.

* **Empty/Null string** :- (ϵ)

Empty string is denoted by 'ϵ' and is defined as a string with no occurance of symbols : length of Empty string is 0.

* **Kleene star** :- (Σ^*)

It is a unary operator on a set of symbols or string, Σ that gives the infinite set of all possible strings of all possible lengths over Σ including ϵ

$\Sigma^* = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2 \ldots$ where Σ_p is the set of all possible strings of length p.

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

$\Sigma^* = \{\epsilon, a, b, aa, bb, ab, ba, \ldots\}$

* **Kleene closure / Plus** :- (Σ^+)

The set Σ^+ is a infinite set of all possible strings of all possible lengths over sigma excluding ϵ.

$$\Sigma^+ = \Sigma^* - \epsilon$$

$$\Sigma^+ = \Sigma_1 \cup \Sigma_2 \cup \Sigma_3 \ldots\ldots$$

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

$\Sigma^+ = \{a, b, ab, ba, aa, bb, \ldots\}$

* **Language:-** A language is a subset of Σ^* for some alphabet Σ. it can be finite or infinite.

Eg: If the language take all possible strings of length 2 over $\Sigma = \{a, b\}$

$L = \{aa, ab, ba, bb\}$.

22/09/21

* **Operations on Languages:-**

1) **Complimentation:**

Let 'L' be a language over an alphabet Σ, the Compliment of L is denoted by $\bar{L}$.

$$\bar{L} = \Sigma^* - L$$

2) **Union :-**

Let L_1 and L_2 are languages over an alphabet Σ. The union of L_1 and L_2 is denoted by $L_1 \cup L_2$

3) **Concatination :-**

Let L_1 and L_2 be languages over an alphabet Σ. The Concatination of L_1 and L_2 is denoted by $L_1 \cdot L_2$ (or) $L_1 L_2$

4) **Reversal :-**

Let L be a language over an alphabet Σ. The reversal of L is denoted by L^R.

* **Regular Expressions :-(RE)**

The language excepted by finite automata can easily described by simple expressions called RE.

→ It is most effective way to represent any language.

→ The language accepted by some regular Expressions are reffred as Regular Language.

→ RE can also be described as a sequence of pattern

that define a string.

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

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

⇒ $L = \{001, 10, 111\}$ $M = \{\epsilon, 001\}$

$L \cup M = \{\epsilon, 10, 001, 111\}$

⇒ concatination $= L.M$

$L = \{001, 10, 111\}$ $M = \{\epsilon, 001\}$

$L.M = \{001, 10, 111, 001001, 10001, 111001\}$

* $L(\epsilon + F) = L(\epsilon) \cup L(F)$
* $L(\epsilon F) = L(\epsilon) L(F)$
* $L(\epsilon^*) = (L(\epsilon))^*$
* $L((\epsilon)) = L(\epsilon)$
* Precidence of regular Expression operator:-

1) *
2) • (contatination)
3) + (union)

Ex 1:

→ All strings of 0's and 1's

$\{\epsilon, 0, 1, 00, 11, 01, 10, 111, \ldots\}$

$RE = (0+1)^*$

Ex 2:

→ Set of all strings of 0's and 1's ending in 00

$\{00, 100, 000, 0100, \ldots\}$

$RE = (0+1)^* 0$

$\{\epsilon, 0, 1, 00, 11, 01, 10, 111, \ldots\} 00$

Ex 3:-

→ Set of all strings of 0's and 1's begining with 0 and ending with 1.

$\{01, 0011, 0101, 001, 0001, \ldots\}$

$RE = 0(0+1)^*1$

$0\{\epsilon, 0, 1, 00, 11, 01, 10, 111, \ldots\}1$

Ex 4:-

→ Set of all strings having even numbers of 1's

$\{\epsilon, 11, 1111, 111111, \ldots\}$

$RE = (11)^*$

Ex 5:

→ Set of all strings having odd number of 1's

$\{\epsilon, 1, 111, 11111, \ldots\}$

$RE = (11)^*1$ (or) $1(11)^*$

Ex 6:

→ Set of all strings of 0's and 1's with atleast two consecutive 0's.

$\{00, 001, 100, 0011, 1100, 0100, \ldots\}$

$RE = (0+1)^*00(0+1)^*$

Ex 7:

→ Set of all strings ending with 011.

$\{011, 0011, 1011, \ldots\}$

$RE = (0+1)^*011$

23/09/21

Ex 8:

→ Set of all strings with 0's followed by 1's, followed by 2's, such that it has atleast one 0 followed by atleast one 1, followed by atleast one 2,

$\{012, 0012, 00112, 0122, 001122, 012012 \ldots\}$

$RE = 0^+1^+2^+$

Ex 8:

→ Set of all strings of 0's and 1's whose last two symbols are the same.

$$\{00, 11, 0011, 011, 100, 0000, 1111, 1011, 0111, \ldots\}$$

$$RE = (0+1)^* (00+11)$$

* Empty set :- (ϕ)

A set having no element in it is called empty set

$$A = \{\}$$

$$|\phi| = 1$$

→ an empty string is present

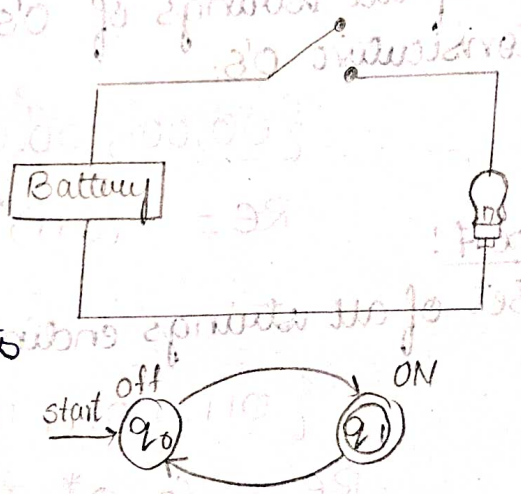
→ An empty string is denoted by ϵ.

→ ϵ indicates no input symbol i.e, 0 input symbol.

$$|\epsilon| = 0$$

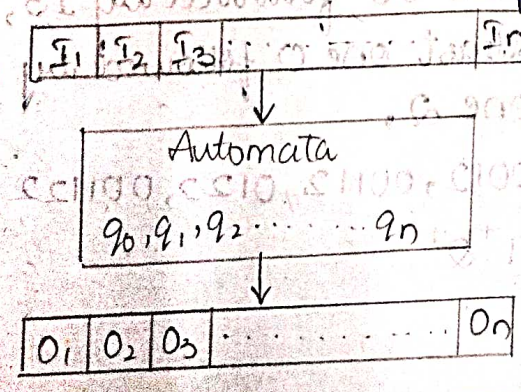
* Finite Automata :-

finite automata is the simplest machine to recognise patterns. The finite Automata or finite state machine is an abstract machine which have five elements or Tuple. It has a set of states and rules for moving one state to another but it depends upon the applied input symbol.



input : switch
output : bulb
status : ON, OF

Basically it is an abstract model of digital computer.



A finite Automata consists of the following

1) Q → finite set of states;
2) Σ → set of input symbols:
3) q → initial state
4) F → final state
5) δ → Transition function.

use moment of an automatan from one state to another by treating the current state and current input symbol as an order pair.

$$\{Q, \Sigma, q, F, \delta\}$$

You can have only one initial state but more final states.

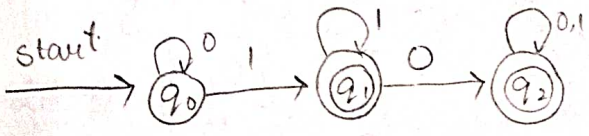
DFA
Deterministic finite automata

NFA
Non Deterministic finite automata

24/09

1) Deterministic finite automata:-

OFF



$Q = \{q_0, q_1, q_2\}$

$\Sigma = \{0, 1\}$

q = initial state = q_0

F = final states = q_1, q_2

$\delta(q_0, 0) = q_0$
$\delta(q_0, 1) = q_1$
$\delta(q_1, 0) = q_2$
$\delta(q_1, 1) = q_1$
$\delta(q_2, 0) = q_2$
$\delta(q_2, 1) = q_2$

Transition table:

	0	1
q_0	q_0	q_1
* q_1	q_2	q_1
* q_2	q_2	q_2

i) Deterministic finite automata:-

DFA consists of 5 tuples $\{Q, \Sigma, q, F, \delta\}$

Q: set of all states

Σ: set of input symbols

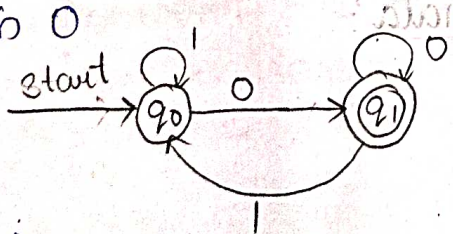
q: initial state

F: final states

δ: Transition function $\Rightarrow Q \times \Sigma \rightarrow Q$

In DFA, for a particular input character the machine goes to one state only. A Transition function is defined on every state for every input symbol. also in DFA null or ϵ move is not allowed. i.e DFA cannot change state without any input character.

Eg: DFA with $\Sigma = \{0,1\}$ accept all strings ending with 0



Q: $\{q_0, q_1\}$

Σ: $\{0,1\}$

q_0: $\{q_0\}$

F: $\{q_1\}$

$\delta(q_0, 0) = q_\phi$

$\delta(q_0, 1) = q_0$

$\delta(q_1, 0) = q_1$

$\delta(q_1, 1) = q_0$

$\{10, 0, 00, 010, 10010, \ldots\}$

Note:
There can be many possible DFA for a pattern. A DFA with minimum no. of states generally preferred.

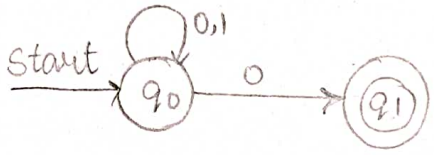
ii) Non deterministic finite automata:-
NFA is similar to DFA except following additional features.

1. Null or ϵ move is allowed i.e, it can move forward without reading symbols.
2. Ability to transmit to any number of states for a particular input.

* In terms of power both are equal.

$$\delta: Q \times (\Sigma \cup \epsilon) \rightarrow 2^Q.$$

Eg:



Note:
1) Both there can be multiple final states in both DFA & NFA.
2) NFA is more of a theoretical concept.

* Construction of DFA:-

→ Type 1:- for strings ending with a particular sub string.

step 1: decide the minimum no. of states required in the DFA and draw them.

~~step 2~~: null: all strings ending with 'n' length sub string will always require minimum of (n+1) states. in its DFA

step 2: decide the string for which you will construct the DFA

step 3: Construct DFA for the above decided strings.

Note: always prefer to go with the existing path.

Create a new path only one you can't find a path to go with.

Eg: draw the DFA for the language accepting strings ending with '01' over input alphabet.

$\Sigma = \{0, 1\}$.

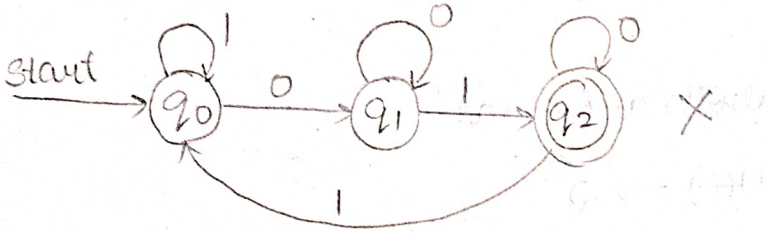
$RE = (0+1)^* 01$

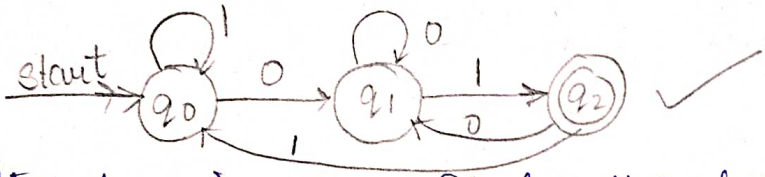
$n = 2$

minimum no. of states $= (n+1) = 3$

$\{01, 001, 0101 \ldots\ldots\}$

	0	1
$\epsilon\ q_0$	q_1	q_0
$0\ q_1$	q_1	q_2
$01^* q_2$	q_1	q_0





Step 4: after drawing the DFA for the above decided strings send the left possible combination to the starting state not over the dead configuration.

27/09/21

Dead state :-

A rejecting state that is essentially a dead state. Once the machine enters a dead state there is no way for it to reach the accepting state.

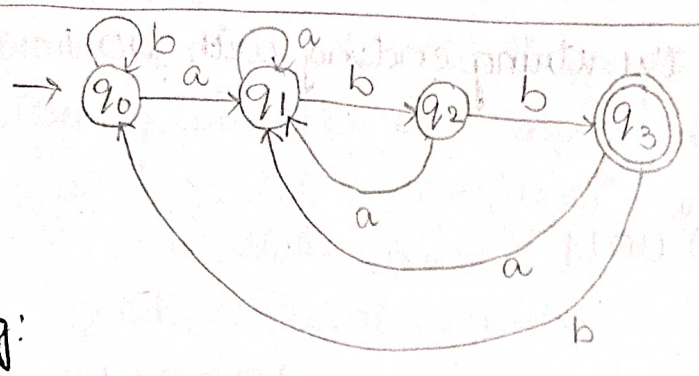
Eg: draw the DFA for the language accepting strings ending with 'abb' over a alphabet

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

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

states : $(n+1) = 4$.

$\{abb, aabb, ababb, abbabb, \ldots\ldots\}$



	a	b
ϵ q_0	q_1	q_0
a q_1	q_1	q_2
ab q_2	q_1	q_3
abb* q_3	q_1	q_0

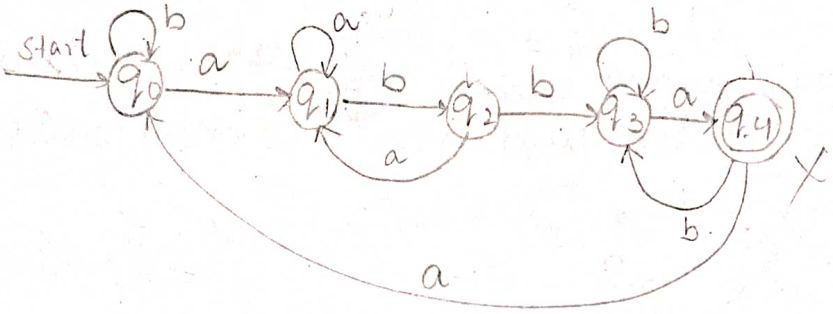
Eg:

Draw the DFA for the language accepting strings ending with 'abba' over $\Sigma = \{a, b\}$.

RE :- $(a+b)^*abba$

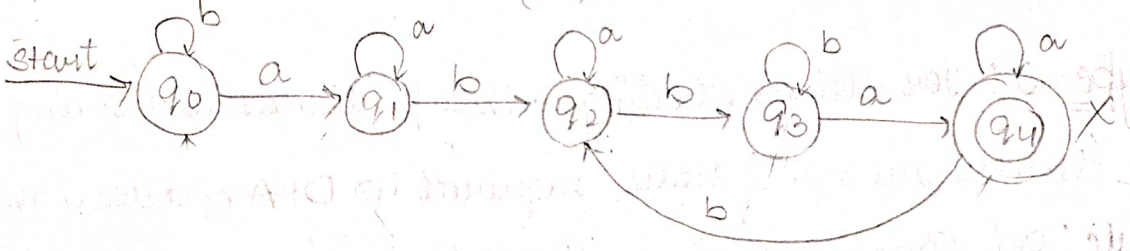
$(n+1) = 5$

{ aaabba, baabba, abba, abbabba, ... }

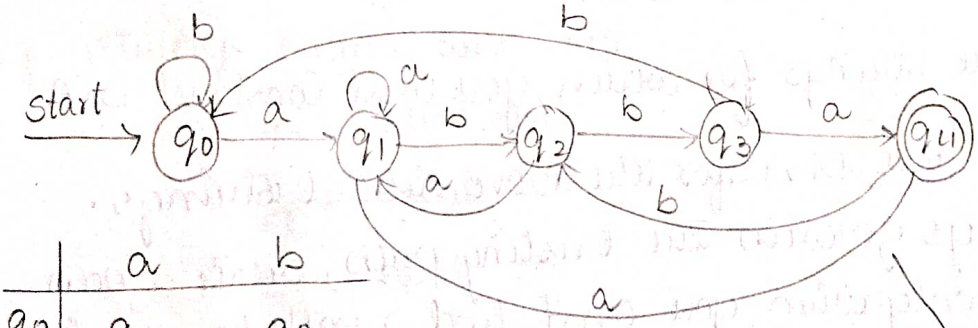


abbaabba

(or)



or



mam

	a	b
q_0	q_1	q_0
q_1	q_1	q_2
q_2	q_1	q_3
q_3	q_4	q_0
q_4	q_1	q_2

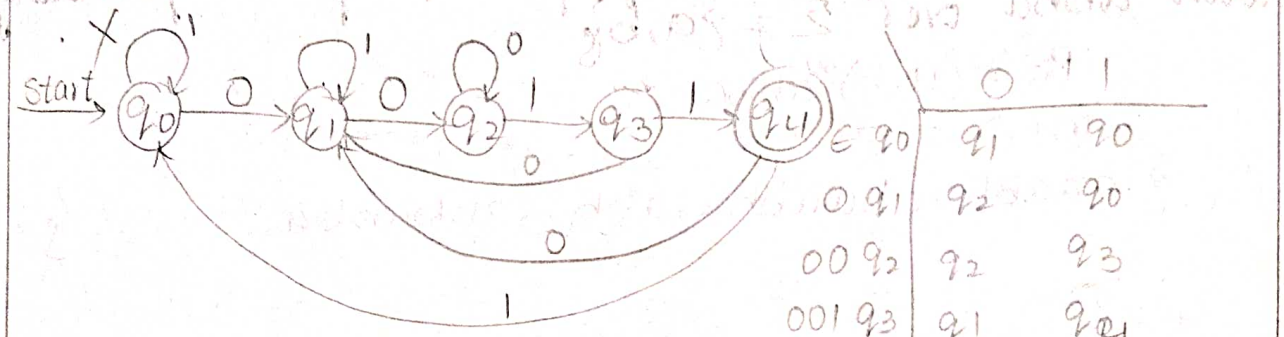
	a	b
ϵ q_0	q_1	q_0
a q_1	q_1	q_2
ab q_2	q_1	q_3
abb q_3	q_4	q_1
abba* q_4	q_1	q_2

Eg: draw the DFA with the string ending with 0011 over $\Sigma = \{0, 1\}$

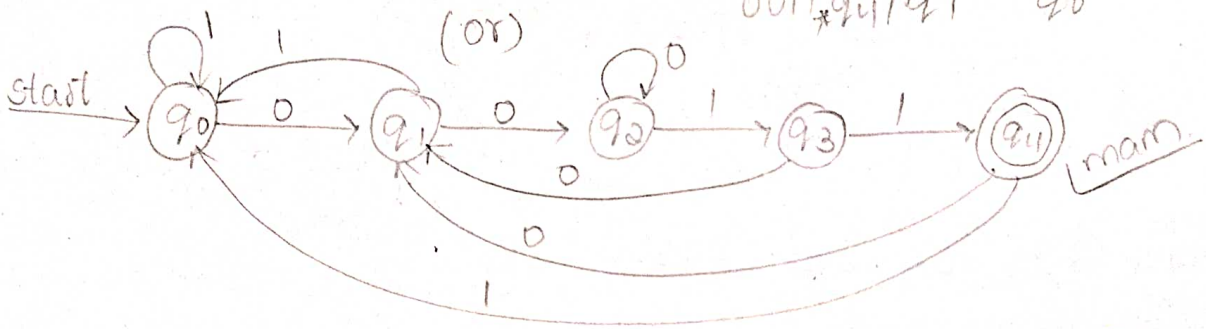
$RE = (0+1)^* 0011$

$(n+1) = 5$

$\{0011, 0100 11, 0000 11, 0011 0011, 10 0011, \ldots\}$



	0	1
$\in$ q0	q1	q0
0 q1	q2	q0
00 q2	q2	q3
001 q3	q1	q4
0011 *q4	q1	q0



* Type-2: For strings starting with a particular substring

Eg: 1 decides the no. of states required in DFA & draw them

Step rule: All strings starting with n length has minimum (n+2) states in the DFA.

Step 2. decide the strings for which you will Construct DFA.

Step 3: Construct the DFA for the above decided strings.

note: always go with the existing path, create a new path only when you can't find a path to go with.

Step 4: after drawing the DFA for the above decided strings send the left possible combinations to the dead state not over the starting state

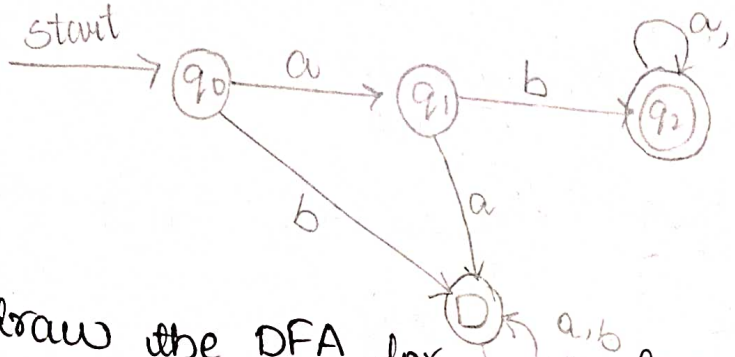
Eg: draw the DFA for the language accepting strings starting with ab. over input alphabet $\Sigma = \{a, b\}$.

$RE = ab(a+b)^*$

states = (n+2) = 4

{ab, abba, aba, abab, ...}

	a	b
$\in q_0$	q_1	q_ϕ
$a\ q_1$	q_ϕ	q_2
ab, q_2	q_2	q_2
reject q_ϕ	q_ϕ	q_ϕ



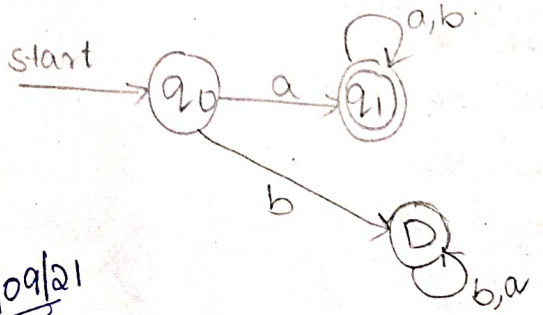
Eg: draw the DFA for the language accepting strings starting with a. over input alphabet $\Sigma = \{a, b\}$.

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

states = 3.

{a, aa, abb, abab, aabba,

	a	b
$\in q_0$	q_1	q_ϕ
a, q_1	q_1	q_1
reject q_ϕ	q_ϕ	q_ϕ



29/09/21

Eg: strings ending with '100'

q_0 = initial state

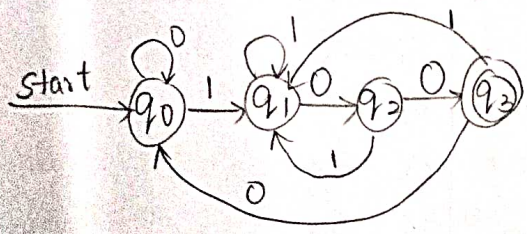
q_p = input 1

q_2 = input 10

* q_3 = input 100

	0	1
$\in q_0$	q_0	q_1
$1\ q_1$	q_2	q_1
$10\ q_2$	q_3	q_1
$100\ q_3$	q_0	q_1

10
X1
100
X01
1000
X001



* Type-2

Eg: strings starting with 'aba'

q_0 = initial state

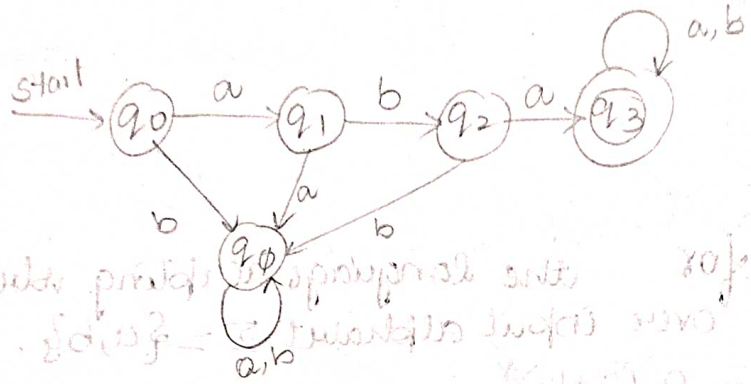
q_1 = a

q_2 = ab

q_3 = aba

reject q_ϕ =

		a	b
$\in$	q_0	q_1	q_ϕ
a	q_1	q_ϕ	q_2
ab	q_2	q_3	q_ϕ
aba*	q_3	q_3	q_3
reject	q_ϕ	q_ϕ	q_ϕ

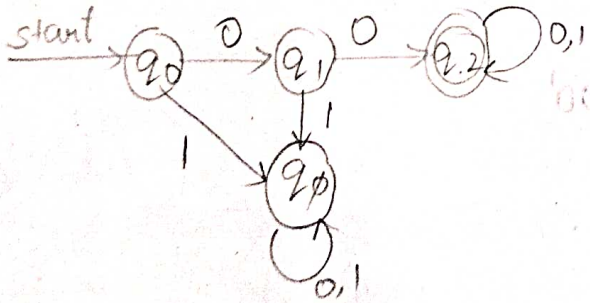


Eg: draw the DFA for the language accepting strings starting with '00'. $\Sigma = \{0, 1\}$

$RE = 00(0+1)^*$

States = (n+2) = 4

		0	1
$\in$	q_0	q_1	q_ϕ
0	q_1	q_2	q_ϕ
00*	q_2	q_2	q_2
reject	q_ϕ	q_ϕ	q_ϕ



30/09/21

* Type-3 substring

Eg: bab

q_0 – initial state

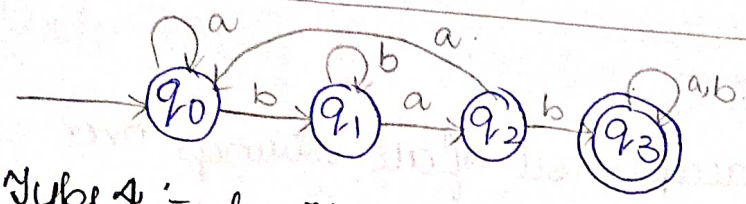
q_1 – b

q_2 – ba

$*q_3$ – bab

		a	b
$\in$	q_0	q_0	q_1
b	q_1	q_2	q_1
ba	q_2	q_0	q_3
bab	q_3	q_3	q_3

*



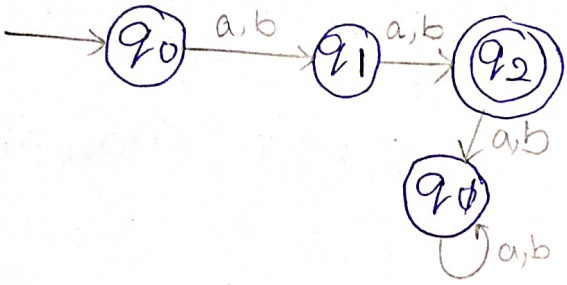
Type 4 :- length

construct a DFA which accepts all strings over $\Sigma = \{a, b\}$ of length 2.

length = 2

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

$L = \{aa, ab, bb, ba, ab\}$

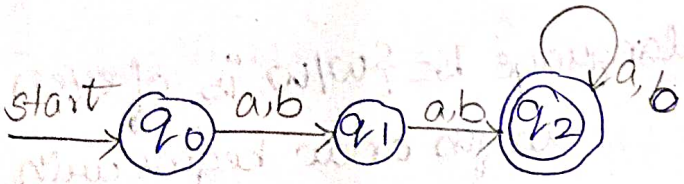


construct DFA accepts set of all strings over $\Sigma = \{a, b\}$ of length >= 2.

length >= 2

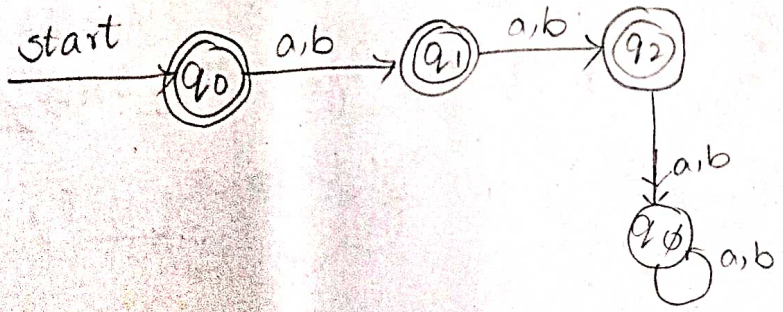
$\Sigma = \{a, b\}$

$L = \{aa, ab, aab, bab, baaba, \cdots\}$



let <= 2

$\Sigma = \{a, b\} \Rightarrow L = \{a, b, aa, bb, ab, ba, \in\}$



* Type 5 : divisibility

1. Construct a DFA which accepts set of all strings over $\Sigma = \{a, b\}$ where length of string is divisible by 3.

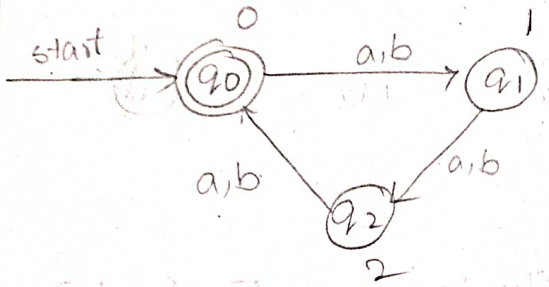
$\Sigma = \{a, b\}$

$L = \{\epsilon, a, aa, aab, bab, abaabb, \ldots\}$

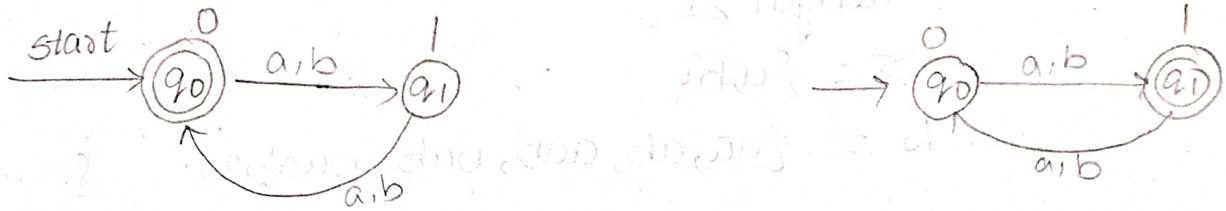
q_0 – remainder 0

q_1 – remainder 1

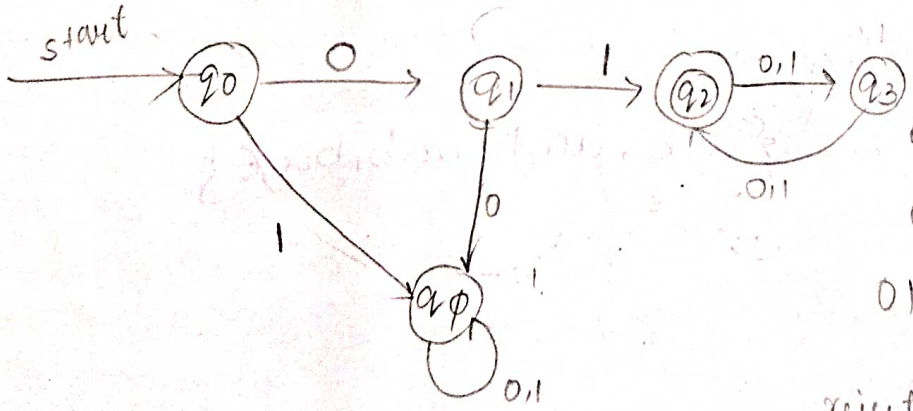
q_2 – remainder 2



2. length divisible by 2. (01) even lengths for odd lengths



3. design a DFA to accept the language $L = \{w / w$ is of even length and begins with 01$\}$



	0	1
$\in$ q_0	q_1	q_ϕ
0 q_1	q_ϕ	q_2
01 q_2	q_2	q_2
reject q_ϕ	q_ϕ	q_ϕ

01/10/21

* Extending the transition function to strings:

$\omega = xa$

$\omega = 1101 \quad x = 110 \quad a = 1$

$\hat{\delta}(q, \omega) = \delta(\hat{\delta}(q, x), a)$

$\omega = 011101$

δ	0	1
q_0	q_1	q_4
q_1	q_4	q_2
q_2	q_3	q_3
q_3	q_2	q_2
q_4	q_4	q_4

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

$\hat{\delta}(q_0, 0) = \delta(\hat{\delta}(q_0, \epsilon), 0)$

$= \delta(q_0, 0) = q_1$

$\hat{\delta}(q_0, 01) = \delta(\hat{\delta}(q_0, 0), 1)$

$= \delta(q_1, 1) = q_2$

$\hat{\delta}(q_0, 011) = \delta(\hat{\delta}(q_0, 01), 1)$

$= \delta(q_2, 1) = q_3$

$\hat{\delta}(q_0, 0111) = \delta(\hat{\delta}(q_0, 011), 1)$

$= \delta(q_3, 1) = q_2$

$\hat{\delta}(q_0, 01110) = \delta(\hat{\delta}(q_0, 0111), 0)$

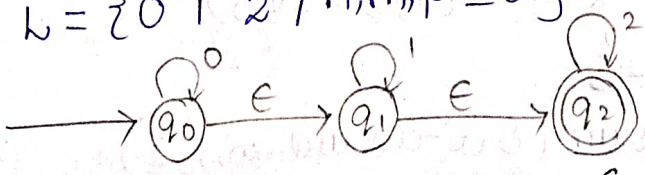
$= \delta(q_2, 0) = q_3$

$\hat{\delta}(q_0, 011101) = \delta(\hat{\delta}(q_0, 01110), 1)$

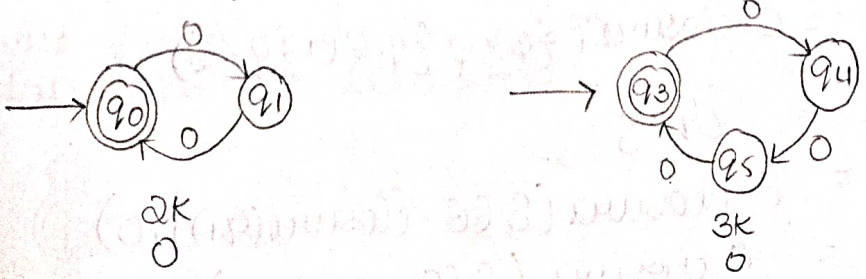
$= \delta(q_3, 1) = q_2$

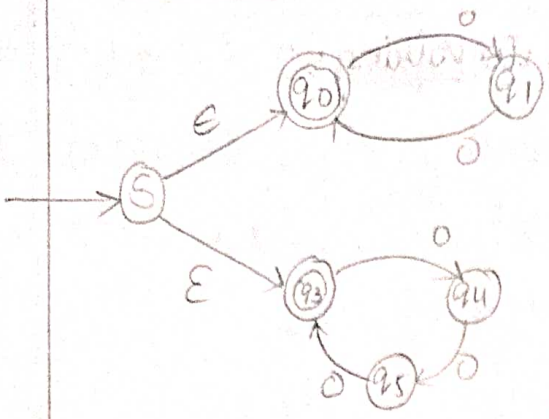
* NFA:

$L = \{0^n 1^m 2^p / n, m, p \geq 0\}$



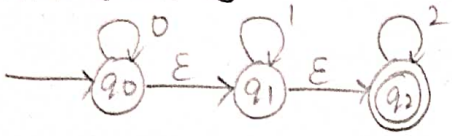
Eg: construct NFA for language $L = \{0^k / k$ is multiple of 2 or 3$\}$





$0^{2K}/0^{3K}$

* Converting NFA with ε Transitions to NFA without ε transitions.



	0	1	2	ε
q_0	q_0	ϕ	ϕ	q_1
q_1	ϕ	q_1	ϕ	q_2
q_2	ϕ	ϕ	q_2	ϕ

ε-closures:-

ε-closure$(q_0) = \{q_0, q_1, q_2\}$

ε-closure$(q_1) = \{q_1, q_2\}$

ε-closure$(q_2) = \{q_2\}$

$\hat{\delta}(q_0,0) = \varepsilon$-closure$(\delta(\varepsilon\text{-closure}(q_0)),0)$

$= \varepsilon$-closure$(\delta(q_0,q_1,q_2),0)$

$= \varepsilon$-closure$(\{q_0\} \cup \{\phi\} \cup \{\phi\})$

$= \{q_0, q_1, q_2\}$

$\hat{\delta}(q_0,1) = \varepsilon$-closure$(\delta(\epsilon\text{-closure}(q_0)),1)$

$= \varepsilon$-closure$(\delta(q_0,q_1,q_2),1)$

$= \epsilon$-closure$(\{\phi\} \cup \{q_1\} \cup \{\phi\})$

$= \{q_1, q_2\}$

$\hat{\delta}(q_0,2) = \varepsilon$-closure$(\delta(\epsilon\text{-closure}(q_0)),2)$

$= \varepsilon$-closure$(\delta(q_0,q_1,q_2),2)$

$= \epsilon$-closure$(\{\phi\} \cup \{\phi\} \cup \{q_2\})$

$= \{q_2\}$

$\hat{\delta}(q_1,0) = \varepsilon$-closure$(\delta(\epsilon\text{-closure}(q_1)),0)$

$= \epsilon$-closure$(\delta(q_1,q_2),0)$

$= \epsilon\text{-closure}(\{\phi\} \cup \{\phi\})$

$= \{\phi\}$.

$\hat{\delta}(q_1, 1) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_1)), 1)$

$= \epsilon\text{-closure}(\delta(q_1, q_2), 1)$

$= \epsilon\text{-closure}(\{q_1\} \cup \{\phi\})$

$= \{q_1, q_2\}$.

$\hat{\delta}(q_1, 2) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_1)), 2)$

$= \epsilon\text{-closure}(\delta(q_1, q_2), 2)$

$= \epsilon\text{-closure}(\{\phi\} \cup \{q_2\})$

$= \{q_2\}$

$\hat{\delta}(q_2, 0) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_2)), 0)$

$= \epsilon\text{-closure}(\delta(q_2), 0)$

$= \epsilon\text{-closure}(\{\phi\})$

$= \{\phi\}$

$\hat{\delta}(q_2, 1) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_2)), 1)$

$= \epsilon\text{-closure}(\delta(q_2), 1)$

$= \epsilon\text{-closure}(\{\phi\})$

$= \{\phi\}$

$\hat{\delta}(q_2, 2) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_2)), 2)$

$= \epsilon\text{-closure}(\delta(q_2), 2)$

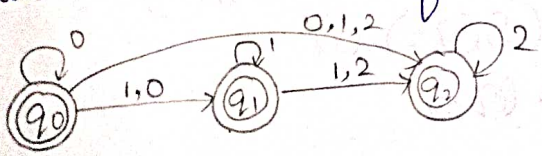
$= \epsilon\text{-closure}(\{q_2\})$

$= \{q_2\}$

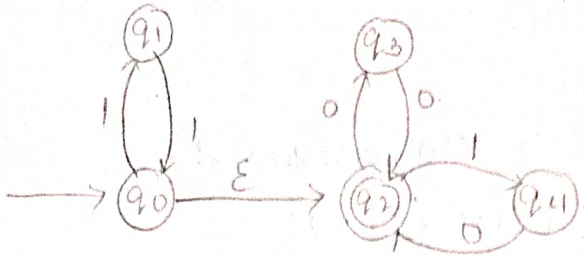
	0	1	2
$*q_0$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
$*q_1$	$\{\phi\}$	$\{q_1, q_2\}$	$\{q_2\}$
$*q_2$	$\{\phi\}$	$\{\phi\}$	$\{q_2\}$

Note:

for each state compute ϵ-closures of q on each input symbol if the ϵ-closure of a state contains a final state then make the state final.



Ex:



	0	1	ε
q0	ϕ	q1	q2
q1	ϕ	q0	ϕ
q2	q3	q4	ϕ
q3	q2	ϕ	ϕ
q4	q2	ϕ	ϕ

ϵ-closure$(q_0) = \{q_0, q_2\}$
ϵ-closure$(q_1) = \{q_1\}$
ϵ-closure$(q_2) = \{q_2\}$
ϵ-closure$(q_3) = \{q_3\}$
ϵ-closure$(q_4) = \{q_4\}$

$\hat{\delta}(q_0, 0) = \epsilon$-closure$(\delta(\epsilon$-closure$(q_0), 0)$
$= \epsilon$-closure$(\delta(q_0, q_2), 0)$
$= \epsilon$-closure$(\{\phi\} \cup \{q_3\})$
$= \{q_3\}$

$\hat{\delta}(q_0, 1) = \epsilon$-closure$(\delta(\epsilon$-closure$(q_0), 1)$
$= \epsilon$-closure$(\delta(q_0, q_2), 1)$
$= \epsilon$-closure$(\{q_1\} \cup \{q_4\})$
$= \{q_1, q_4\}$

$\hat{\delta}(q_\phi, 0) = \epsilon$-closure$(\delta(\epsilon$-closure$(q_1)), 0)$
$= \epsilon$-closure$(\delta(q_1), 0)$
$= \epsilon$-closure$(\{\phi\})$
$= \{\phi\}$

$\hat{\delta}(q_1, 1) = \epsilon$-closure$(\delta(\epsilon$-closure$(q_1)), 1)$
$= \epsilon$-closure$(\delta(q_1), 1)$
$= \epsilon$-closure$(\{q_0\})$
$= \{q_0, q_2\}$

$\hat{\delta}(q_2, 0) = \epsilon$-closure$(\delta(\epsilon$-closure$(q_2)), 0)$
$= \epsilon$-closure$(\delta(q_2), 0)$
$= \epsilon$-closure$(\{q_3\})$
$= \{q_3\}$

$\hat{\delta}(q_2,1) = \varepsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_2)),1)$

$= \varepsilon\text{-closure}(\delta(q_2),1)$

$= \epsilon\text{-closure}(\{q_4\})$

$= \{q_4\}$

$\hat{\delta}(q_3,0) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_3)),0)$

$= \epsilon\text{-closure}(\delta(q_3),0)$

$= \epsilon\text{-closure}(\{q_2\})$

$= \{q_2\}$

$\hat{\delta}(q_3,1) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_3)),1)$

$= \epsilon\text{-closure}(\delta(q_3),1)$

$= \epsilon\text{-closure}(\{\phi\})$

$= \{\phi\}$

$\hat{\delta}(q_4,0) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_4)),0)$

$= \epsilon\text{-closure}(\delta(q_4),0)$

$= \epsilon\text{-closure}(\{q_2\})$

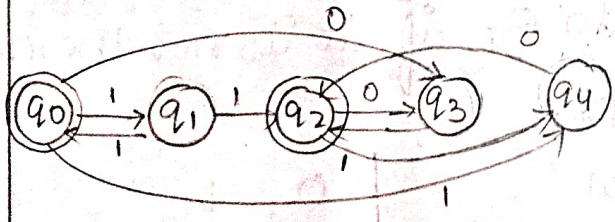
$= \{q_2\}$

$\hat{\delta}(q_4,1) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_4)),1)$

$= \epsilon\text{-closure}(\delta(q_4),1)$

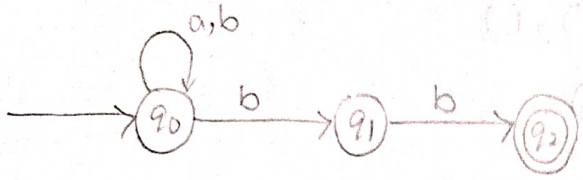
$= \epsilon\text{-closure}(\{\phi\})$

$= \{\phi\}$



	0	1
*q_0	$\{q_3\}$	$\{q_1, q_4\}$
q_1	$\{\phi\}$	$\{q_0, q_2\}$
*q_2	$\{q_3\}$	$\{q_4\}$
q_3	$\{q_2\}$	$\{\phi\}$
q_4	$\{q_2\}$	$\{\phi\}$

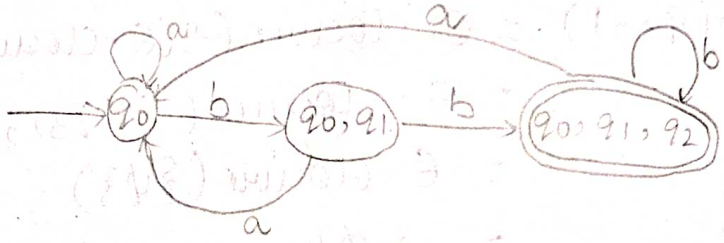
* NFA to DFA :-



	a	b
q0	q0	{q0,q1}
q1	ϕ	{q2}
* q2	ϕ	ϕ

	a	b
q0	q0	{q0,q1}
{q0,q1}	q0	{q0,q1,q2}

	a	b
q0	q0	{q0,q1}
{q0,q1}	q0	{q0,q1,q2}
{q0,q1,q2}	q0	{q0,q1,q2}



04/10/21

1) $Q_D = 2^{Q_N}$. if NFA has 'n' states DFA has atmost 2^n states.

Q = set of states

2) $\Sigma_n = \Sigma_D$

3) $[q_0] = \{q_n\}$. q = initial state

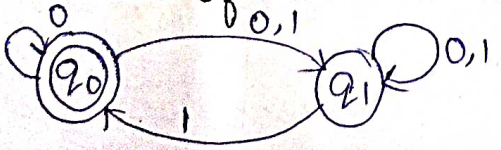
4) F_D = set of all states of Q_D that contain atleast one of the final states of F_n.

5) $\delta_D = ((q_1,q_2,q_3),a) = \delta_n(q_1,a) \cup \delta_n(q_2,a) \cup \delta(q_3,a)$

$= \{P_1,P_2,P_3\}$

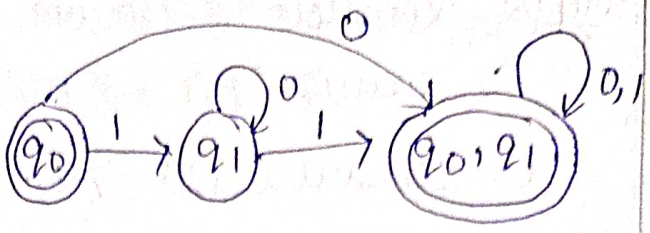
Add the state $[P_1\ P_2\ P_3]$ to Q_D if it is not there.

Ex: 1. conversion of NFA to DFA



	0	1
* q0	(q0,q1)	(q1)
q1	q1	(q0,q1)

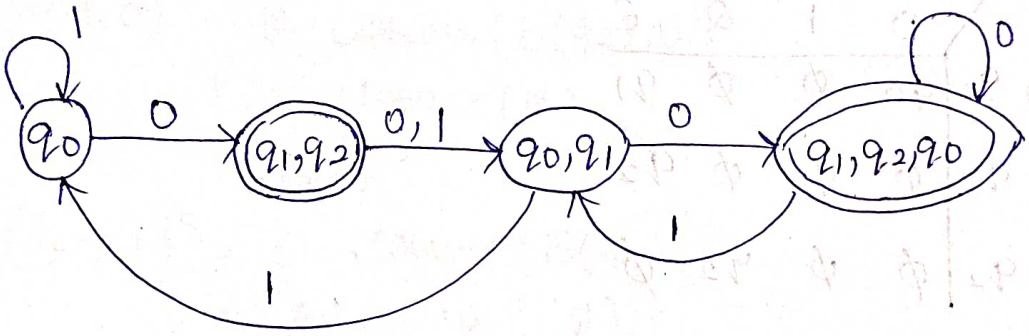
	0	1
q_0	(q_0, q_1)	(q_1)
q_1	(q_1)	(q_0, q_1)
(q_0, q_1)	(q_0, q_1)	(q_0, q_1)



Ex:2 δ

	0	1
q_0	$\{q_1, q_2\}$	$\{q_0\}$
q_1	$\{q_0, q_1\}$	ϕ
* q_2	$\{q_1\}$	$\{q_0, q_1\}$

	0	1
q_0	$\{q_1, q_2\}$	$\{q_0\}$
* (q_1, q_2)	$\{q_0, q_1\}$	$\{q_0, q_1\}$
(q_0, q_1)	$\{q_1, q_2, q_0\}$	$\{q_0\}$
* (q_0, q_1, q_2)	$\{q_1, q_2, q_0\}$	$\{q_0, q_1\}$



* conversion/converting NFA with 'ϵ' transition ito DFA:

step 1:- Q has $2^3 = 8$ states and it is the set of all subsets q_0, q_1, q_2: $\{\epsilon, q_0, q_1, q_2, \{q_0, q_1\}, \{q_0, q_2\}, \{q_1, q_2\}, \{q_0, q_1, q_2\}\}$

$\Sigma = 0, 1, 2$

$q_0 = \epsilon$-closure (q_0)

	0	1	2	ϵ
q_0	q_0	ϕ	ϕ	q_1
q_1	ϕ	q_1	ϕ	q_2
* q_2	ϕ	ϕ	q_2	ϕ

$F = \{q_2, \{q_0, q_2\}, \{q_1, q_2\}, \{q_0, q_1, q_2\}\}$

Step 2: Complete ϵ-closure of each state

ϵ-closure $(q_0) = \{q_0, q_1, q_2\}$

ϵ-closure $(q_1) = \{q_1, q_2\}$

ϵ-closure $(q_2) = \{q_2\}$

Step 3: find the initial state by finding the ϵ-closure of initial state.

Step 4: explore the states that are valid states in DFA starting in the new initial states explore each new states by finding the transitions on every input element (0,1,2).

7/10/21

* Converting NFA with ϵ-transition to DFA

	0	1	2	ϵ
$\rightarrow q_0$	q_0	ϕ	ϕ	q_1
q_1	ϕ	q_1	ϕ	q_2
$*q_2$	ϕ	ϕ	q_2	ϕ

$Q = 2^3 = 8 = \{q_0, \epsilon, q_1, q_2, q_0q_1, q_0q_2, q_1q_2, q_0q_1q_2\}$

$F = \{q_2, q_0q_2, q_1q_2, q_0q_1q_2\}$

ϵ-closure $(q_0) = \{q_0, q_1, q_2\}$

ϵ-closure $(q_1) = \{q_1, q_2\}$.

ϵ-closure $(q_2) = \{q_2\}$.

$\hat{\delta}((q_0q_1q_2), 0) = \epsilon$-closure $(\delta(q_0, q_1, q_2), 0)$

$= \epsilon$-closure (q_0)

$= (q_0, q_1, q_2)$

$\hat{\delta}((q_0q_1q_2), 1) = \epsilon$-closure $(\delta(q_0, q_1, q_2), 1)$

$= \epsilon$-closure (q_1)

$= \{q_1, q_2\}$

$\hat{\delta}((q_0,q_1,q_2),2) = \epsilon\text{-closure}(\delta(q_0,q_1,q_2),2)$

$= \epsilon\text{-closure}(q_2)$

$= \{q_2\}$

$\hat{\delta}((q_1,q_2),0) = \epsilon\text{-closure}(\delta(q_1,q_2),0)$

$= \epsilon\text{-closure}(\phi)$

$= \{\phi\}$

$\hat{\delta}((q_1,q_2),1) = \epsilon\text{-closure}(\delta(q_1,q_2),1)$

$= \epsilon\text{-closure}(q_1)$

$= \{q_1,q_2\}$.

$\hat{\delta}((q_1,q_2),2) = \epsilon\text{-closure}(\delta(q_1,q_2),2)$

$= \epsilon\text{-closure}(q_2)$

$= \{q_2\}$.

$\hat{\delta}((q_2),0) = \epsilon\text{-closure}(\delta(q_2),0)$

$= \epsilon\text{-closure}(\phi)$

$= \{\phi\}$

$\hat{\delta}((q_2),1) = \epsilon\text{-closure}(\delta(q_2),1)$

$= \epsilon\text{-closure}(\phi)$

$= \{\phi\}$.

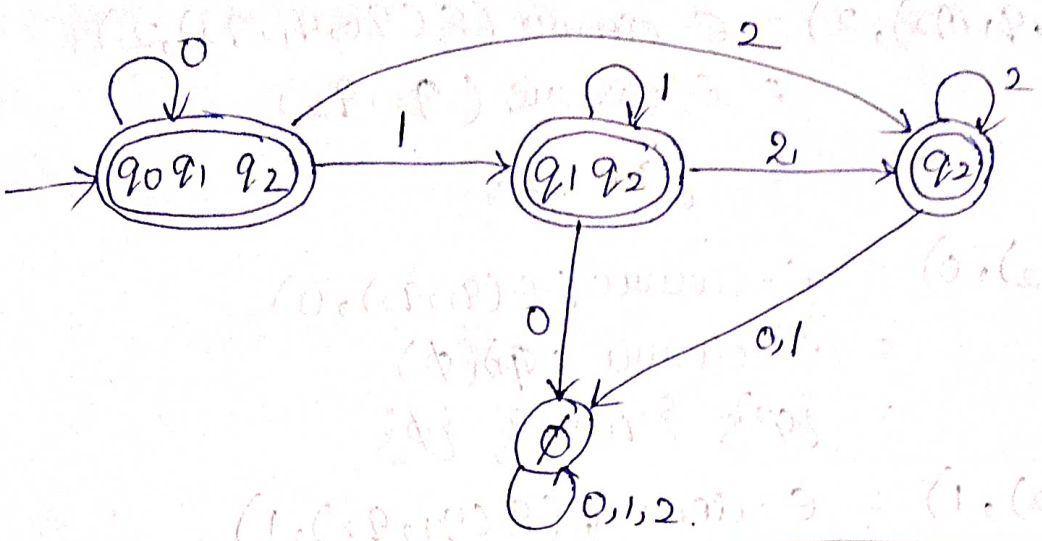
$\hat{\delta}((q_2),2) = \epsilon\text{-closure}(\delta(q_2),2)$

$= \epsilon\text{-closure}(\phi_2)$

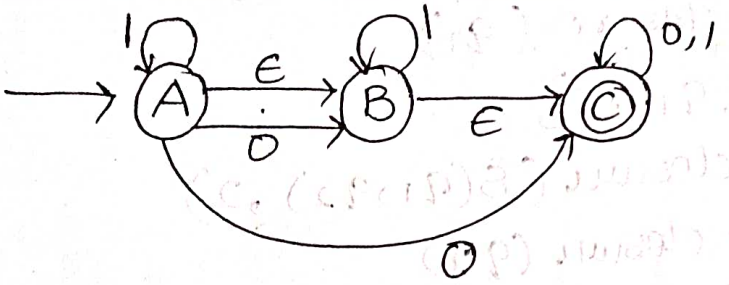
$= \{q_2\}$.

$\hat{\delta}(\phi,0) = \epsilon\text{-closure}(\delta(\phi),0)$

	0	1	2
*$(q_0\,q_1\,q_2)$	$(q_0q_1q_2)$	(q_1q_2)	(q_2)
*(q_1q_2)	ϕ	(q_1,q_2)	(q_2)
*(q_2)	ϕ	ϕ	(q_2)
ϕ	ϕ	ϕ	ϕ



*



	0	1	ϵ
A	(B,C)	(A)	(B)
B	∅	(B)	(C)
*C	C	C	∅

$Q = 2^3 = 8$

$= \{A, \epsilon, B, C, (A,B), (A,C), (B,C), (A,B,C)\}$.

$F = \{C, (A,C), (B,C), (A,B,C)\}$

ϵ-closure $(A) = \{A, B, C\}$

ϵ-closure $(B) = \{B, C\}$

ϵ-closure $(C) = \{C\}$.

$\hat{\delta}((A,B,C), 0) = \epsilon$-closure $(\delta(A,B,C), 0)$

$= \epsilon$-closure $((B,C) \cup (\emptyset) \cup (C))$

$= \{B, C\}$

$\hat{\delta}((A,B,C), 1) = \epsilon$-closure $(\delta(A,B,C), 1)$

$= \epsilon$-closure (A, B, C)

$= \{A, B, C\}$

$\hat{\delta}((A,B,C),0) = \epsilon\text{-closure}(\delta(A,B,C),0)$

$= \epsilon\text{-closure}(\phi \cup C)$

$= \{C\}.$

$\hat{\delta}((B,C),1) = \epsilon\text{-closure}(\delta(B,C),1)$

$= \epsilon\text{-closure}(B \cup C)$

$= \{B,C\}.$

$\hat{\delta}((C),0) = \epsilon\text{-closure}(\delta(C,0))$

$= \epsilon\text{-closure}(C)$

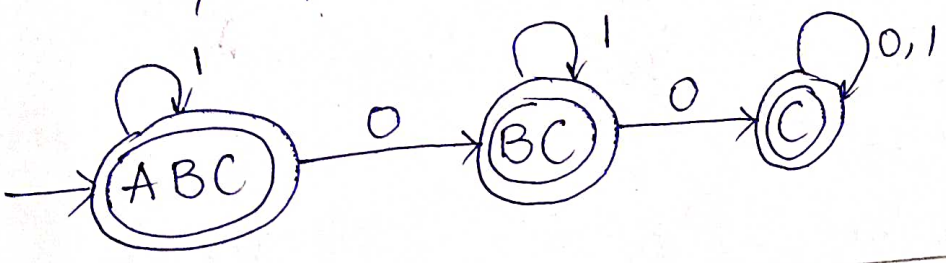
$= \{C\}.$

$\hat{\delta}((C),1) = \epsilon\text{-closure}(\delta(C),1)$

$= \epsilon\text{-closure}(C)$

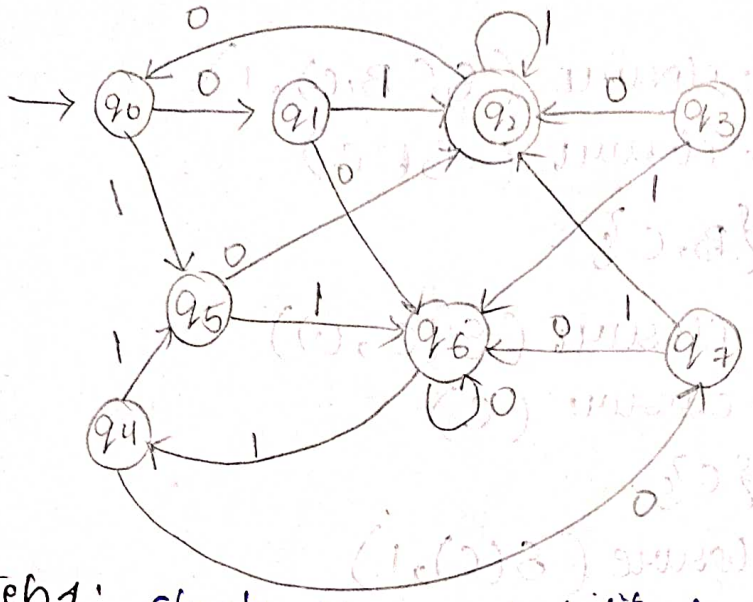
$= \{C\}.$

	0	1
* (A B C)	(B,C)	(A,B,C)
* (BC)	(C)	(B,C)
* (C)	(C)	(C)
ϕ	ϕ	ϕ



8/10/21

* Minimization of DFA :- (reduce the no. of states).



q_0 is not reaching q_3.

step 1: check the reachability from the initial state to all the remaining states.

step 2: construct the transition table.

	0	1
q_0	q_1	q_5
q_1	q_6	q_2
* q_2	q_0	q_2
q_3	q_2	q_6
q_4	q_7	q_5
q_5	q_2	q_6
q_6	q_6	q_4
q_7	q_6	q_2

→ remove q_3

step 3: make the set of states into two parts.

i) set of non-final states

ii) set of final states.

0 – equivalence.

non-final states → $\{q_0, q_1, q_4, q_5, q_6, q_7\}$

final states → $\{q_2\}$.

(q_0, q_1)

1 – equivalence.

$\delta(q_0,0) = q_1 \quad \delta(q_1,0) = q_6 \quad \delta(q_0,1) = q_5$

$\delta(q_1,1) = q_2$

$\{q_0 \quad \{q_1$

(q_0, q_4)

$\delta(q_0,0) = q_1 \quad \delta(q_4,0) = q_7$

$\delta(q_0,1) = q_5 \quad \delta(q_4,1) = q_5$

$\{q_0, q_4 \quad \{q_1$

(q_0, q_5)

$\delta(q_0,0) = q_1 \quad \delta(q_5,0) = q_2.$

$\delta(q_0,1) = q_5 \quad \delta(q_5,1) = q_6$

(q_1, q_5)

$\delta(q_1,0) = q_6 \quad \delta(q_5,0) = q_2$

$\delta(q_1,1) = q_2 \quad \delta(q_5,1) = q_6$

$\{q_0, q_4 \quad \{q_1 \quad \{q_5$

(q_0, q_6)

$\delta(q_0,0) = q_1 \quad \delta(q_6,0) = q_6$

$\delta(q_0,1) = q_5 \quad \delta(q_6,1) = q_4.$

$\{q_0, q_4, q_6 \quad \{q_1 \quad \{q_5$

(q_0, q_7)

$\delta(q_0,0) = q_1 \quad \delta(q_7,0) = q_6$

$\delta(q_0,1) = q_5 \quad \delta(q_7,1) = q_2$

(q_0, q_7) $\delta(q_0,0) = q_6 \quad \delta(q_7,0) = q_6$

$\delta(q_1,1) = q_2 \quad \delta(q_7,1) = q_2$

$\{q_0, q_4, q_6\} \quad \{q_1, q_7\} \quad \{q_5\} \quad \{q_2\}$

$\{q_0, q_4, q_6\}$ $\{q_1, q_7\}$ $\{q_5\}$ $\{q_2\}$

2- Equivalence

(q_0, q_4)

$\delta(q_0, 0) = q_1$ $\delta(q_4, 0) = q_7$

$\delta(q_0, 1) = q_5$ $\delta(q_4, 1) = q_5$

$\{q_0, q_4$

(q_0, q_6)

$\delta(q_0, 0) = q_1$ $\delta(q_6, 0) = q_6$

$\delta(q_0, 1) = q_5$ $\delta(q_6, 1) = q_4$

$\{q_0, q_4$ $\{q_6$

(q_1, q_7)

$\delta(q_1, 0) = q_6$ $\delta(q_7, 0) = q_6$

$\delta(q_1, 1) = q_2$ $\delta(q_7, 1) = q_2$

$\{q_0, q_4$ $\{q_6$ $\{q_1, q_7\}$

(q_1, q_6)

$\delta(q_1, 0) = q_6$ $\delta(q_6, 0) = q_6$

$\delta(q_1, 1) = q_2$ $\delta(q_6, 1) = q_4$

(q_7, q_6)

$\delta(q_7, 0) = q_6$ $\delta(q_6, 0) = q_6$

$\delta(q_7, 1) = q_2$ $\delta(q_6, 1) = q_4$

$\{q_0, q_4$ $\{q_6$ $\{q_1, q_7$

(q_6, q_5)

$\delta(q_6, 0) = q_6$ $\delta(q_5, 0) = q_2$

$\delta(q_6, 1) = q_4$ $\delta(q_5, 1) = q_6$

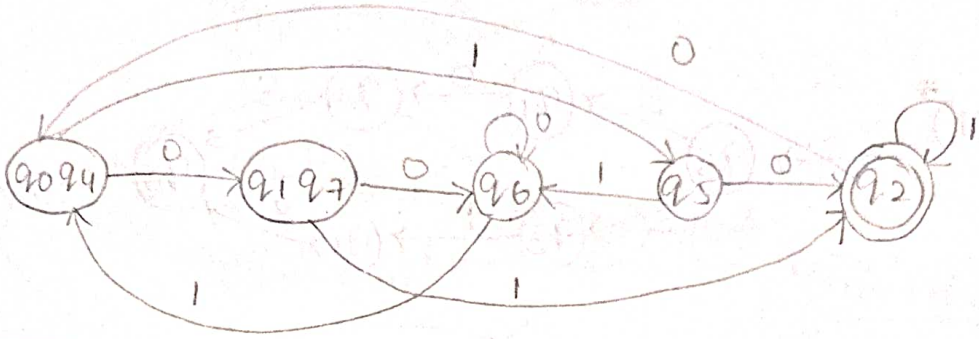
$\{q_0, q_4$ $\{q_6$ $\{q_1, q_7$ $\{q_5$

(q_6, q_2)

$\delta(q_6, 0) = q_6$ $\quad$ $\delta(q_2, 0) = q_0$

$\delta(q_6, 1) = q_4$ $\quad$ $\delta(q_2, 1) = q_2$

$\{q_0, q_4\}$ $\{q_1, q_7\}$ $\{q_6\}$ $\{q_5\}$ $\{q_2\}$.



11/1/21

* Regular expression to NFA with ε-transition:

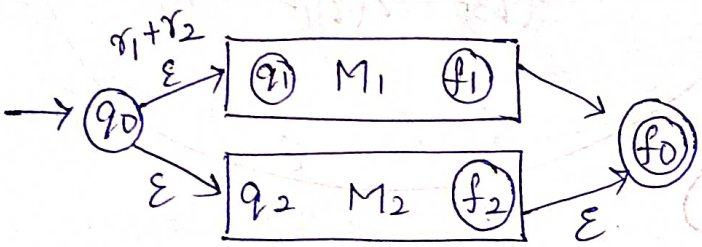
→ Let 'r' be a regular expression then there exists a NFA with ε-transitions that accepts $L(r)$ language.

Case 1: union case

$r = r_1 + r_2$

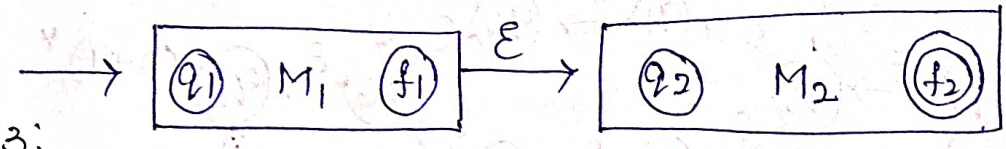
$r_1 =$ (q1 M_1 f1)

$r_2 =$ (q2 M_2 f2)



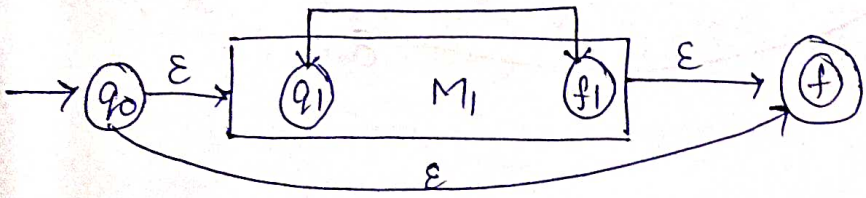
Case 2:

$r = r_1 r_2$



Case 3:

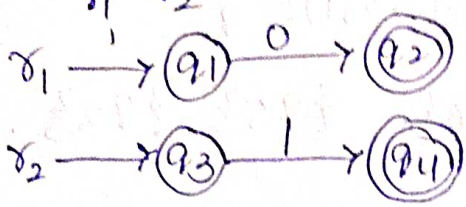
$r = r_1^*$



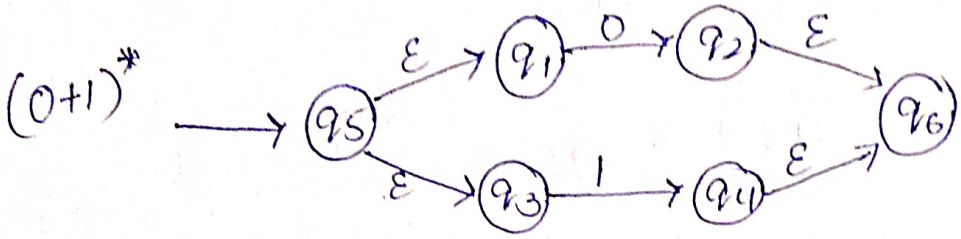
Eg:

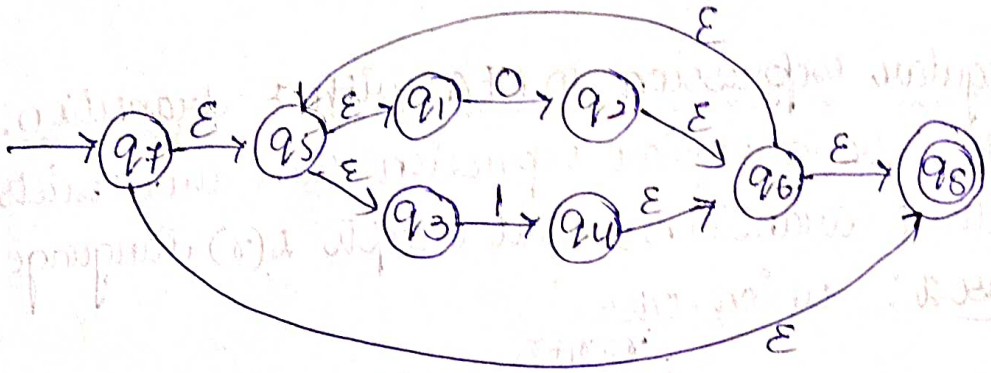
1) $(0+1)^*$

r_1 r_2

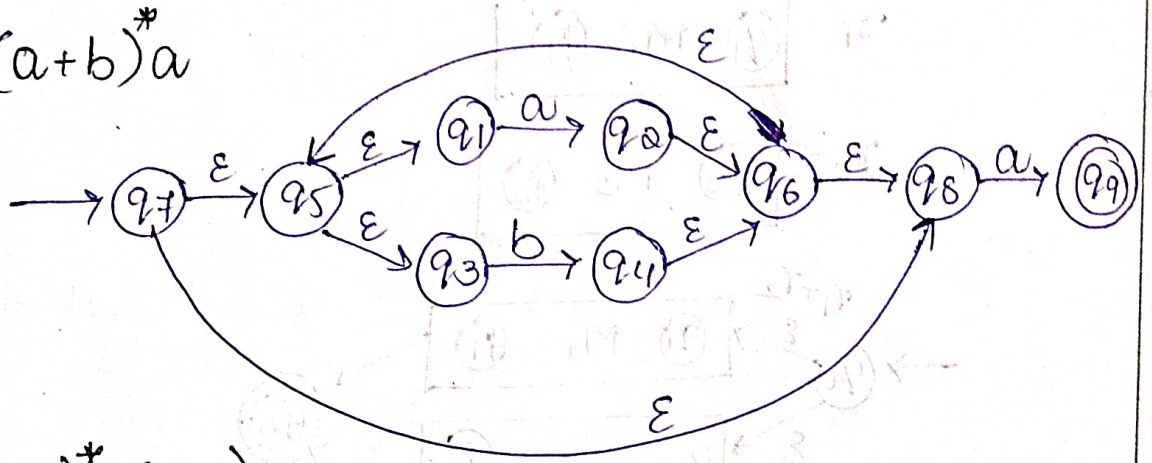


Double

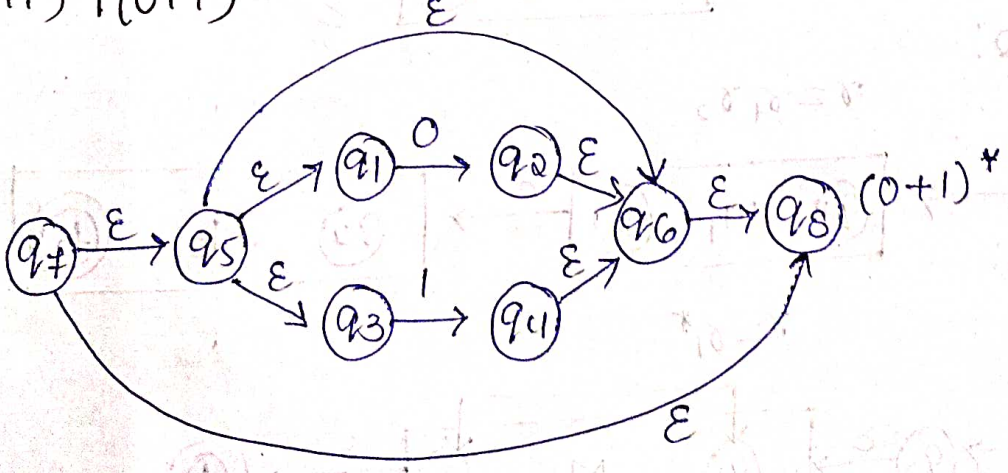


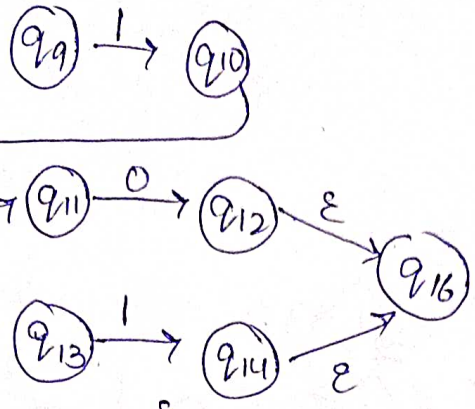


2) $(a+b)^*a$

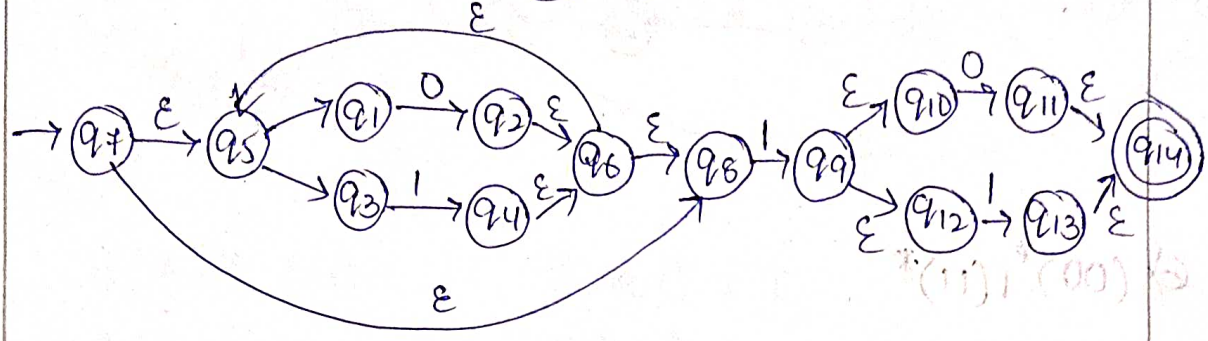


3) $(0+1)^*1(0+1)$

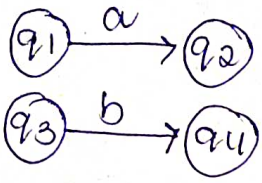




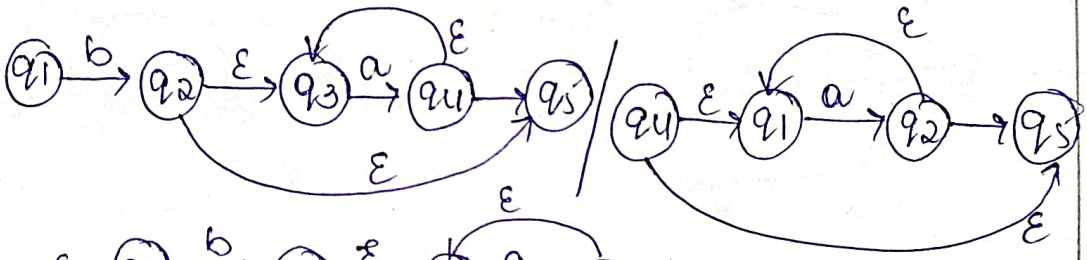
(0+1)

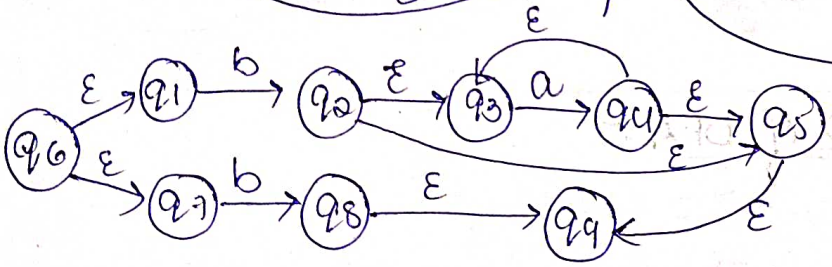


4) $a+ba^*+ab^*$

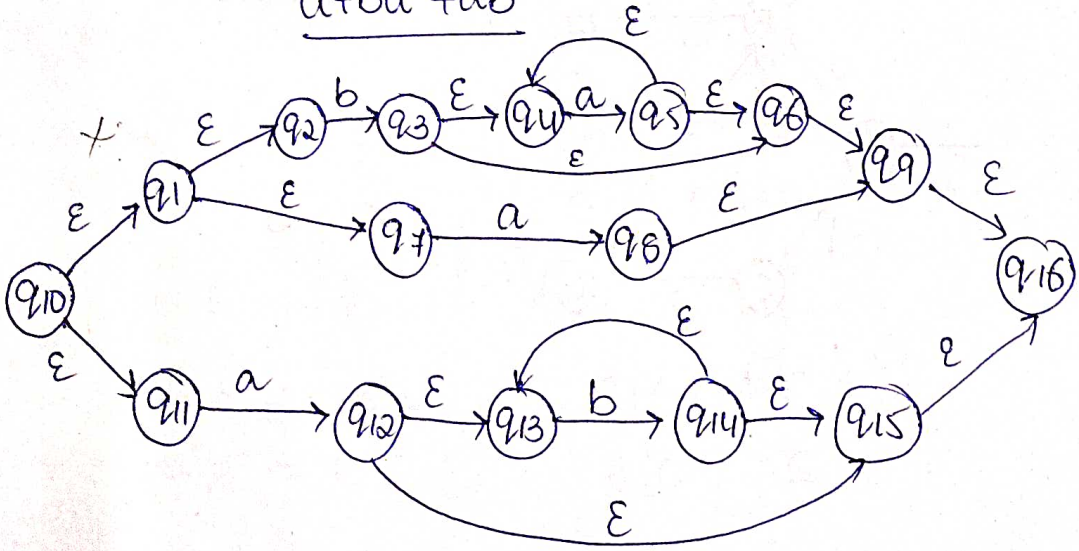


ba^*



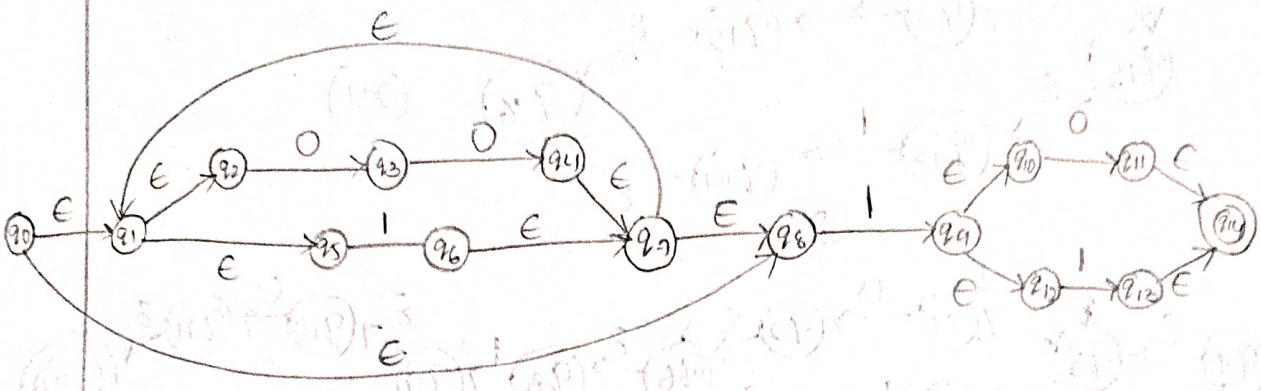


$a+ba^*+ab^*$

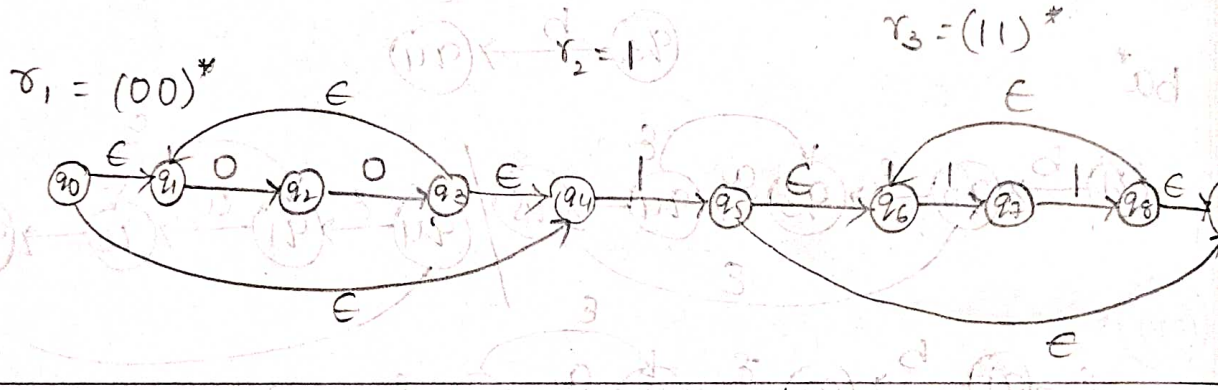


20/10/21

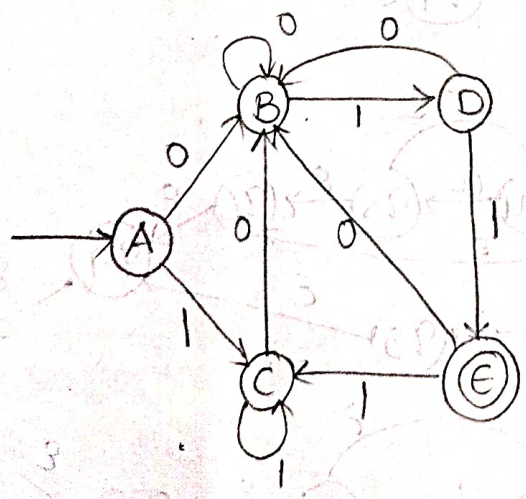
5) $(00+1)^*1(0+1)$



6) $(00)^*1(11)^*$



* Minimisation of DFA :-



	0	1
A	B	C
B	B	D
C	B	C
D	B	E
* E	B	C

0- equivalence.

non-final states → {A, B, C, D}

final states → {E}

1 - equivalence.

(A, B)

$\delta(A,0) = B$ $\quad$ $\delta(A,1) = C$

$\delta(B,0) = B$ $\quad$ $\delta(B,1) = D$

{A, B

(A, C)

$\delta(A,0) = B$ $\quad$ $\delta(A,1) = C$

$\delta(C,0) = B$ $\quad$ $\delta(C,1) = C$.

{A, B, C

(A, D)

$\delta(A,0) = B$ $\quad$ $\delta(A,1) = C$

$\delta(D,0) = B$ $\quad$ $\delta(D,1) = E$.

{A, B, C, D} $\quad$ {D} $\quad$ {E}.

2- equivalence

(A, B)

$\delta(A,0) = B$ $\quad$ $\delta(A,1) = C$

$\delta(B,0) = B$ $\quad$ $\delta(B,1) = D$

{A $\quad$ {B

(A, C)

$\delta(A,0) = B$ $\quad$ $\delta(A,1) = C$

$\delta(C,0) = B$ $\quad$ $\delta(C,1) = C$

{A, C} $\quad$ {B}

(A, D)

$\delta(A,0) = B$ $\quad$ $\delta(A,1) = C$

$\delta(D,0) = B$ $\quad$ $\delta(D,1) = E$

{A, C} $\quad$ {B}

(B, D)

$\delta(B,0) = B$ $\quad$ $\delta(B,1) = D$

$\delta(D,0) = B$ $\quad$ $\delta(B,1) = \epsilon$

$\{A, C\}$ $\quad$ $\{B\}$ $\quad$ $\{D\}$ $\quad$ $\{\epsilon\}$.

(A, ~~ϵ~~)

$\delta(A,0) = B$ $\quad$ $\delta(A,1) = C$

$\delta(\epsilon,0) = B$ $\quad$ $\delta(\epsilon,1) = C$.