module-1 Theory of computation

Kavya.S
Asst. professor
CSD, Dept
PESITM

July-19 Aug Define the following terms with an example (i) Alphabet (ii) power of an alphabet (iii) String (iv) string concatenation (v) Language.

(i) Alphabet (Σ):

It is defined as a set of finite symbol

ex: $\Sigma = \{0, 1\}$ Binary alphabet

$\Sigma = \{0, 1, 2, 3 \text{ ---- } 9\}$ set for decimal numbers.

$\Sigma = \{a, b, c, d \text{ ---}\}$ Alphabet set for english alphabetic

(ii) power of an alphabet (Σ^*)

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \text{----}$$

ex: 1) $\Sigma = \{0, 1\}$

Σ^0 = set of string of length 0

Σ^1 = set of string of length 1

$\Sigma^* = \{0, 1, 00, 11, 10, 01, 000, 001, 111, \text{---}\}$

It is defined as set of strings of all possible strings length derived from Σ.

(iii) String (s):

It is defined as a finite sequence of symbols derived from alphabet (Σ)

ex: consider $\Sigma = \{0, 1\}$

strings that can be derived are

0, 1, 01, 001, 101, 1001, 101010, ---

(iv) String concatenation:

concatenation of two strings (s || t) is defined as string appending t to s

ex: x = ab, y = c

xy = abc.

(v) Language (L):

set of strings, all are choosen from Σ^* where Σ is a particular alphabet is called a language.

$L \subseteq \Sigma^*$

ex: consider $\Sigma = \{a, b\}$

L_1 = set of string where length is odd

$L_1 = \{a, b, aaa, bbb, ababa, \text{---} bbbb\}$

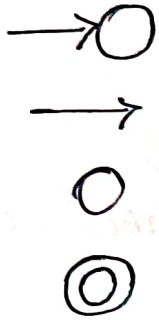
L_2 = set of string begin with a and end with b

$L_2 = \{ab, ababb, aaab, aabbabb \text{---}\}$

(vi) Symbol: It is a building block of automata theory

ex: letter, number (0-9) etc

Symbols to draw an Automata:



→ Initial state

→ Transition

○ → Intermediate state

◎ → End / final state

ϵ – epsilon: It is a special string. It is called null string or empty string.

String

- Functions
  - Length
  - concatenation
  - Replication
  - Reversal
- Relations
  - substring
  - proper substring
  - prefix, proper prefix
  - suffix, proper suffix

1) **Length:** Number of symbols in the given strings

ex: $|1011| = 3$ $|\epsilon| = 0$

2) **concatenation of String:** concatenation of two strings (s || t) is defined as string appending t to s

ex = $x = ab$, $y = c \Rightarrow xy = abc$

3) **Replication:** Replication of string w is defined as

$w^0 = \epsilon$

$w^{i+1} = w^i . w$ eg = $a^3 = aaa$

(Repeating the occurance of w 'i' no of times)

4) Replication: Replication of string 'w' is defined as $w^0 = \epsilon$

5) Reversal:
For each string w is defined as
(i) If $|w| = 0$ then $w = w^R = \epsilon$ eg: $(abc)^R = cba$

6) Relations on string

⇒ substring: A string 't' is a substring of s iff 't' continuously occurs in string

⇒ proper substring:
A string 't' is a proper substring of 's' iff it is a substring of 's' and t≠s
ex: ϵaabbaϵ

$\epsilon, a, aa, aab, aabba$

proper substring

⇒ prefix: A string s is a prefix of 't' iff there exists

$\exists x \in \Sigma^* (t = sx)$

⇒ proper prefix:
A string 's' is a proper prefix of s iff string is suffix of t and $s \neq t$

ex: consider "abba"

$\epsilon, b, bb, bba, abba$

proper suffix

Function on language

All set operation like union, intersection difference and complement can be applied.

(i) concatenation of Language:

ex: $L_1 = \{aa, ab\}$

$L_2 = \{xx, yy\}$

$L_1L_2 = \{aaxx, aayy, abxx, abyy\}$

set operation on language

$L_1 = \{\epsilon, a^2, a^4, a^6 \text{---}\}$ // even no of a's

$L_2 = \{a^1, a^3, a^5 \text{---}\}$ // odd no of a's

→ $L_1 \cup L_2 = \Sigma^*$ or $\{a\}^*$ // union operation

→ $L_1 \cap L_2 = \phi$ or $\{\}$ // intersection operation

→ $L_1 - L_2 =$ // difference operation

→ $\sim(L_1 - L_2)$ // complement operation.

with neat diagram, explain a hierarchy of language classes in Automata theory

⇒ Regular language (FSM)

⇒ context free language (PDA

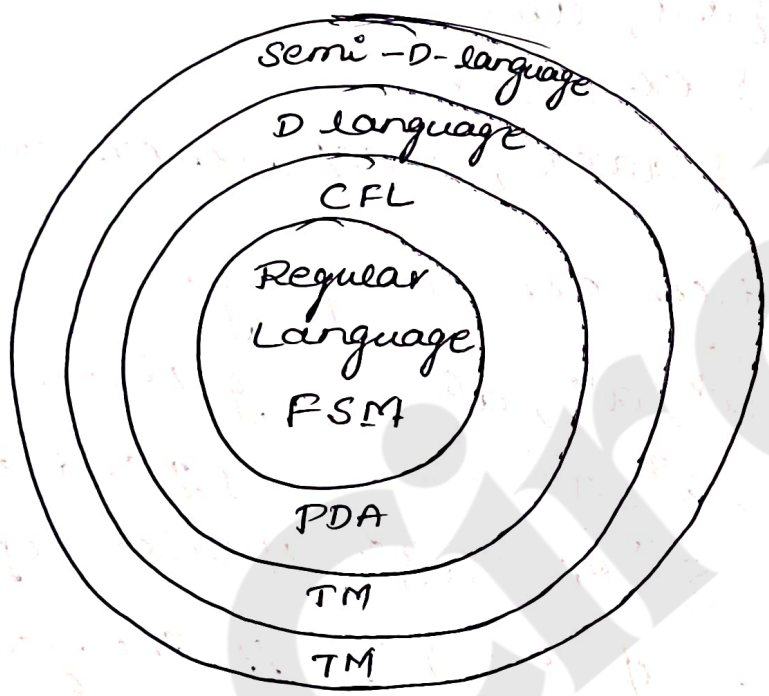
⇒ Decidable language (TM)

⇒ semidecidable language (some TM)

The tools can be selected based on

1. computational efficency : FSM groups

linearly to the length of input string PDA grows cube of length of input string.
TM grows exponentially with length of input string.



2. Decidability:
FSM are decidable for example does FSM accepts some particular string
Is FSM is minimal? Are 2 FSM are identical? PDA is also decidable. But question w.r.t TM can't be answered

3. Clarity:
There are tools to design FSM and every regular language can also be described using regular expression every CFL recognized by some PDA can be described by CF grammer. No corresponding tool exists for the broader classes of decidable & semidecidable languages.

July 1 Define DFSM / FSM.

* There is one and only one transition on an input
* DFSM / FSM is formally defined as,
* M is quintuple $(Q, \Sigma, \delta, q_0, F)$

where,

Q is finite set of state
Σ is the input alphabet
$q_0 \in Q$ is initial state
δ is the transition function
F is End / Final accepting state

$$\therefore \boxed{\delta : Q \times \Sigma \to Q}$$

How to write transition Diagram?

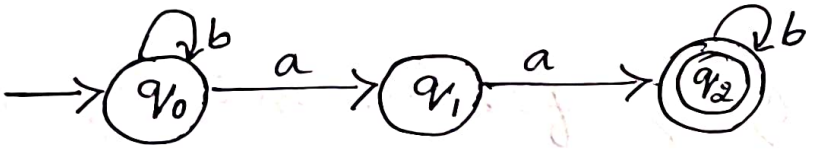
steps are,

1) Find the minimum string accepted, this decides the number of states in the FSM, in most of the cases.
2) Then take longer strings and make them accepted, while modifying the transition.
3) check for the minimum strings that are not to be accepted are really not accepted as per the transition diagram
4) see that each state has transitions & equal to the number of alphabet present (Σ)

5) Two transition on the same alphabet do not go to different states.

problems

1) from the given transition diagram of DFSM define all the taples.



Sol: $Q = \{q_0, q_1, q_2\}$ $M = \{Q, \Sigma, q_0, \delta, f\}$

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

q_0 = Initial state

$F = q_2$ (final state)

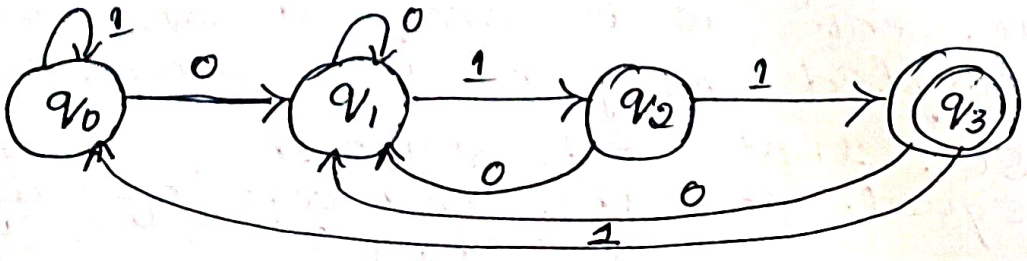
δ = is shown in transition table

δ	a	b
$\rightarrow q_0$	q_1	q_0
q_1	q_2	–
$*q_2$	–	q_2

2) Draw a DFSM to accept strings of 0's and 1's ending with string 011

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

$L = \{011, 0011, 1011, 10011 - - - -\}$



(9)

3) Draw a DFA to accept string of a's & b's having substring aa

Sol $L = \{ aa, baab, abaa, aabb - - - \}$

$M = \{Q, \Sigma, \delta, q_0, F\}$

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

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

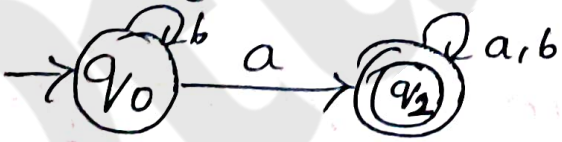
q_0 = Initial state

$F \rightarrow q_2$ = Final state

(4) Obtain a DFA to accept strings of a's and b's having atleast one a.

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

$L = \{ a, aa, aaa, abab - - - - \}$



$Q = \{q_0\ q_1\}$

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

q_0 = Initial state

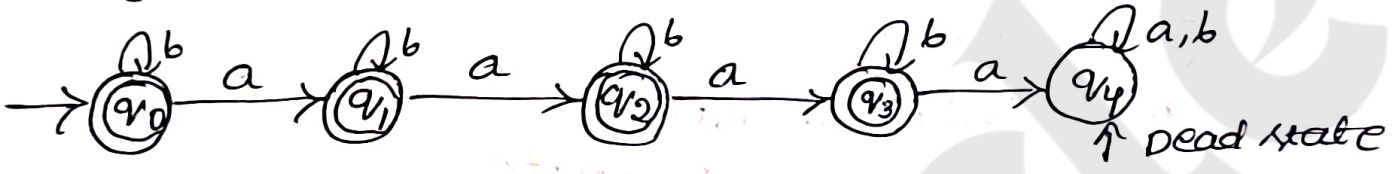
$F = q_1$ final state

(5) obtain a DFA to accept string a's & b's having not more than 3 a's

not more than 3 a's
can be either 0 or 1 or 2 or 3 a's

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

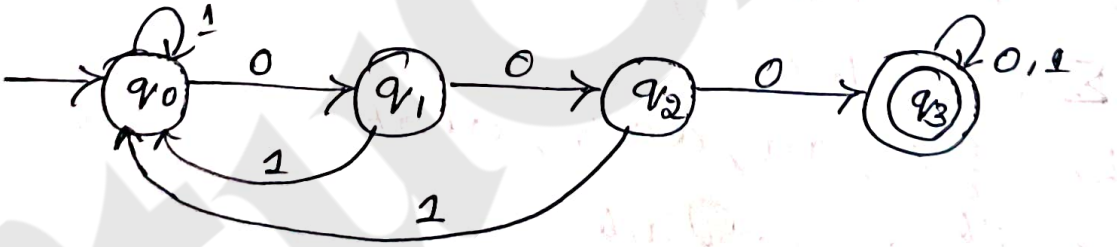
$L = \{\epsilon, a, aa, aaa, ba, aab \cdots\}$



(6) Draw a DFSM that accepts strings of 0's and 1's that contain 3 consecutive 0's as a substring

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

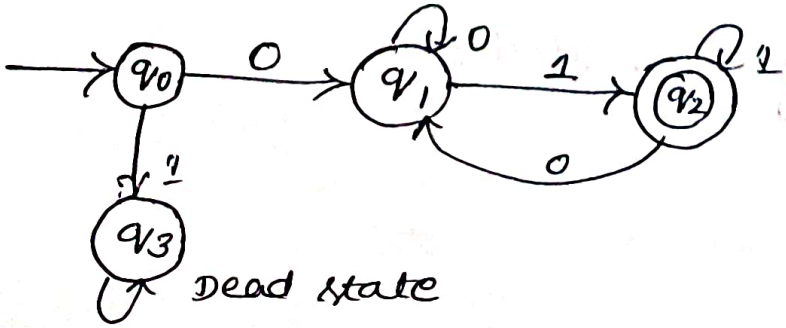
$L = \{000, 0001, 100011, 0100011100\}$



(7) Design a DFSM to accept strings of 0's and 1's where it begin with 0 & end with 1

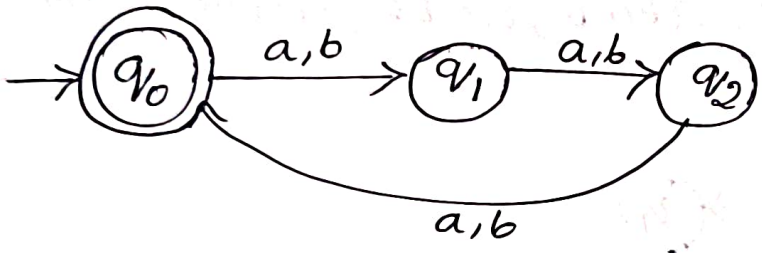
$\Sigma = \{0, 1\}$

$L = \{01, 011, 01001, 011001 \cdots\}$



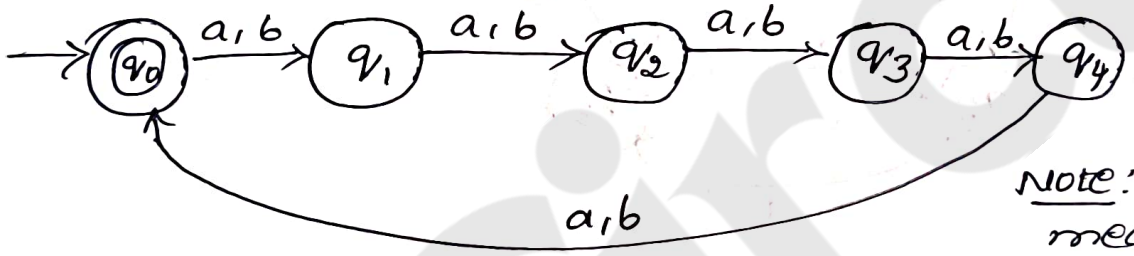
8) obtain a DFSM to accept language

(i) $L = \{ w \mid |w| \bmod 3 = 0 \}$ on $\Sigma = \{a, b\}$



Note: mod 3 means multiples of 3 (a,b)

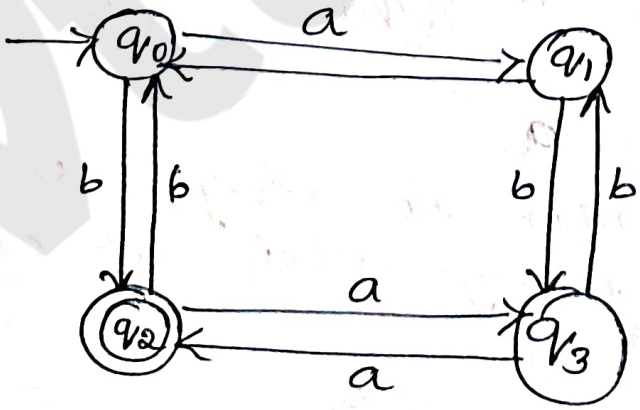
(ii) $L = \{ w \mid |w| \bmod 5 = 0 \}$ on $\Sigma = \{a, b\}$



Note: mod 5 means multiples of 5. (a,b)

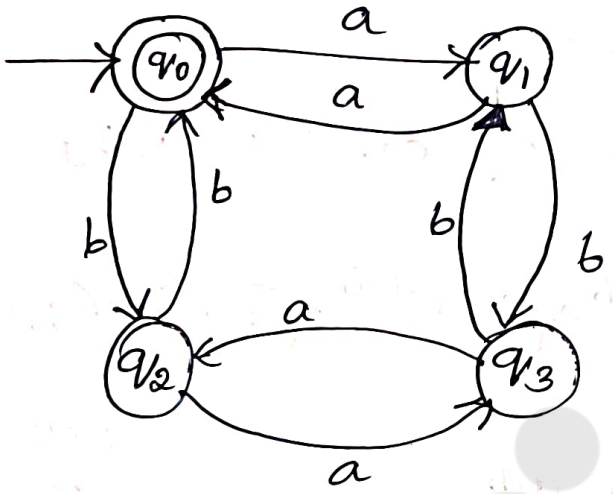
(9) $L = \{ w \mid w \in \{a, b\}^* \}$ is the string with even number of a's and odd no of b's

$L(m) = \{ b, aab, aba, aaaabb \ldots \}$



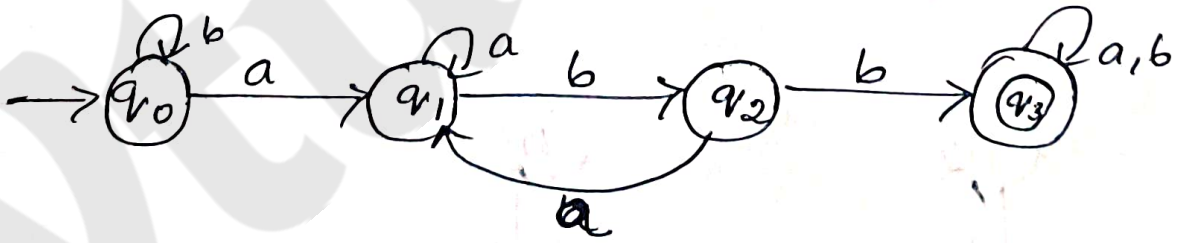
(10) obtain a DFA to accept string of a's and b's having even number of a's and b's

$L = \{ \epsilon, aa, bb, abab, aabb, aaaa, abababab \}$

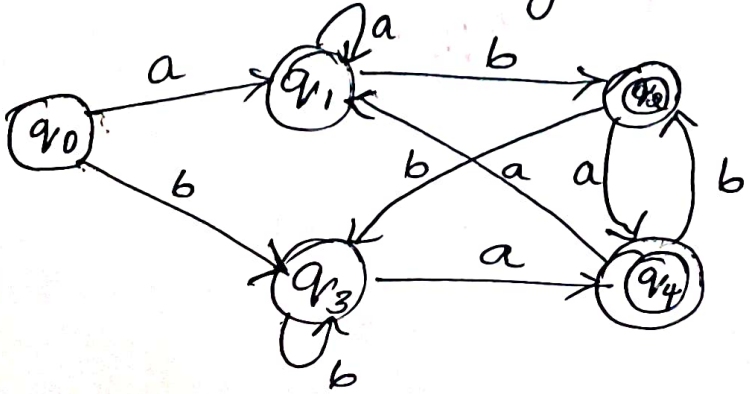


(11) $L = w \in \{a, b\}^*$ is the string of a's and b's and end with the sub string abb.

$L(M) = \{abb, aabb, baabb ---\}$

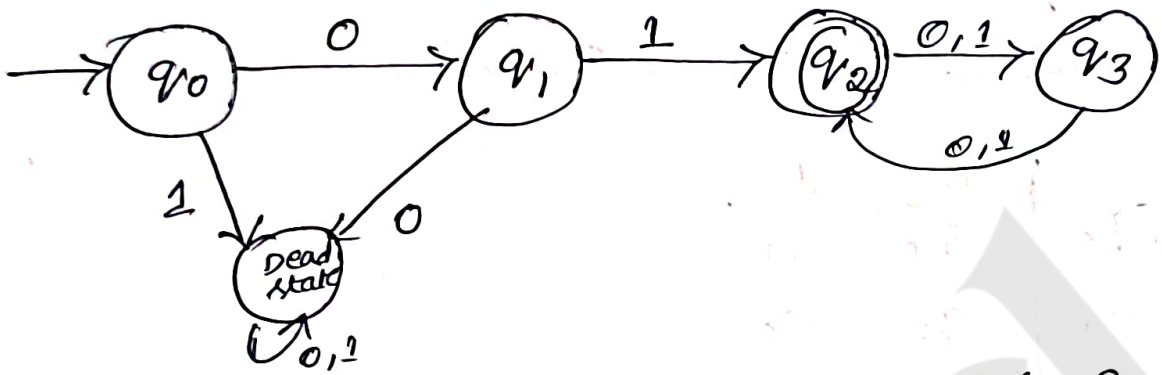


(12) construct a DFA to accept strings of a's and b's ending with ab or ba



(13)

(13) Design a DFA to accept that language $L = \{w \mid w$ is of even length & begin with 01$\}$

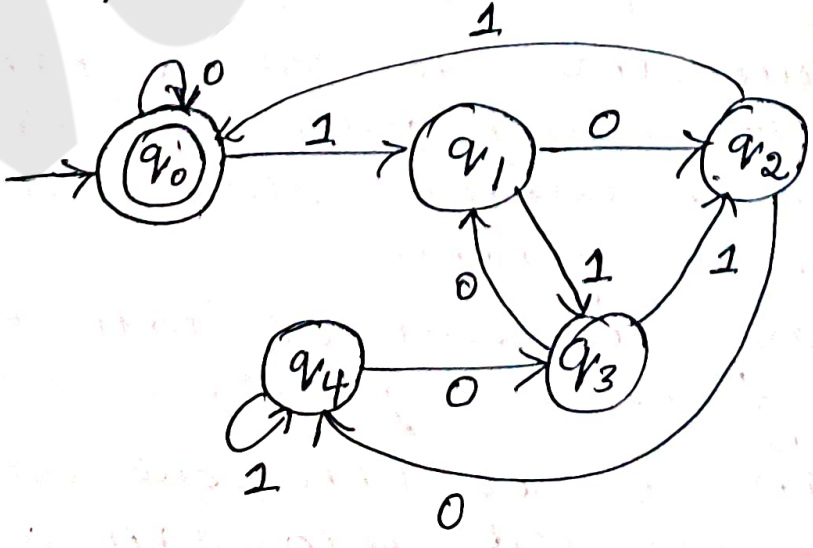


(14) Design a DFA which accept a binary number divisible by 5

Sol Binary number = $\Sigma = \{0, 1\}$

divisible by 5 $\{0, 1, 2, 3, 4\}$

δ	0	1
$\rightarrow *q_0$	q_0	q_1
q_1	q_2	q_3
q_2	q_4	q_0
q_3	q_1	q_2
q_4	q_3	q_4

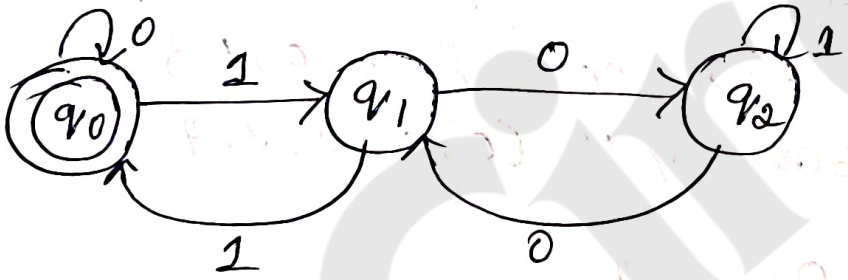


(15) Design DFSM to accept binary number devisible by 3

Binary number $\{0, 1\}$

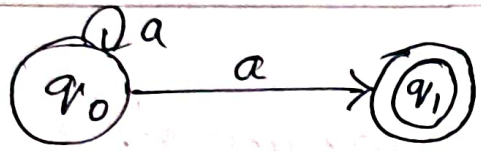
divisible by 3 $\{0, 1, 2\}$

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



Non-deterministic finite Automata (NDFA)

* In DFA there exist only one path from one state to another state on one symbol. (on specific input)
* Finite Automata are called NFA where there exist many path for specific input from current state to next state on a symbol.
* It is easy to construct NFA than DFA for a given language.
* Every NFA is not DFA, but every NFA can be translated into DFA.



possible for NFA but not for DFA

$(q_0, a) \rightarrow q_0$
$(q_0, a) \rightarrow q_1$

2 path exist. specific path is not fixed so it is called Non-Deterministic

→ Having Dead state is not required for NFA.

→ NFA contains ∈ (epsilon) transition i.e without any input, transition is possible from one state to another state.

→ DFA is the subset of NFA.

Formal Definition:

NFA having 5 states same as DFA but with different transition function.

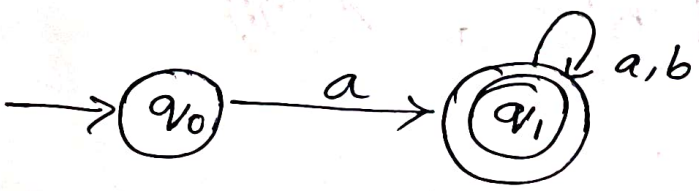
$$M = (Q, F, \delta, q_0, \Sigma)$$

Here $\delta : Q \times \Sigma \rightarrow 2^Q$

↑ Input state ↑ Symbol ↑ Multiple output state

Ex: construct NFA for language
$L = \{w \mid \text{where } w \text{ starts with } a\}$

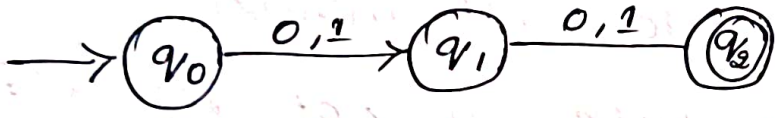
$\Sigma = \{a, b\}$



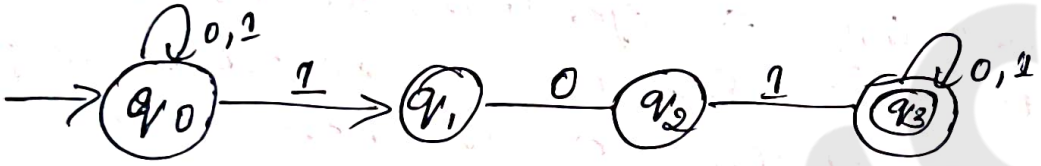
Problems on NFA

(1) Construct NFA for a language L where L = { set of all strings of length 2 }

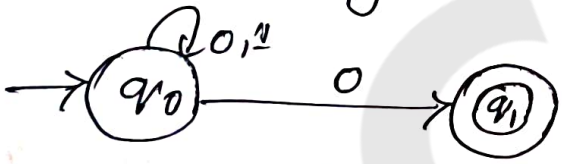
$\Sigma = \{0, 1\}$, $L = \{00, 01, 10, \cdots\}$



(2) Each string contains substring '101'

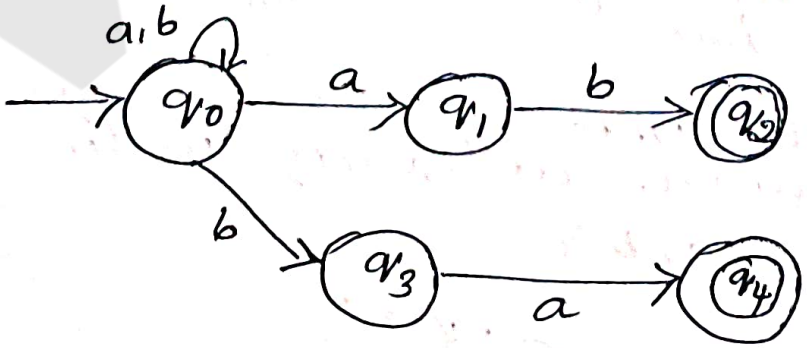


(3) Each string that ends with '0' $\Sigma = \{0, 1\}$

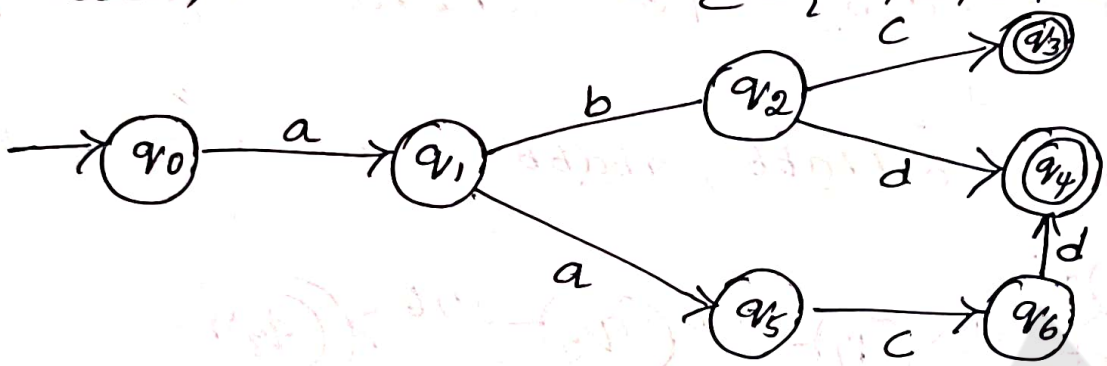


(4) Construct NFA that accepts 0's and 1's that end with 00

(5) Construct NFA to accept string ending with ab or ba over $\Sigma = \{a, b\}$

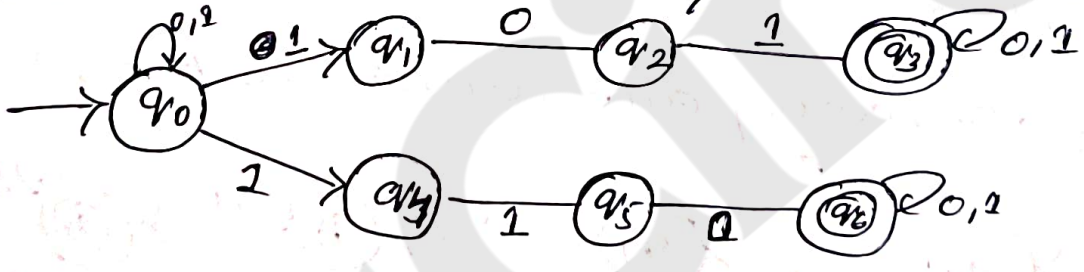


6. write NFA to recognize the string abc, abd, aacd where $\Sigma = \{a, b, c, d\}$

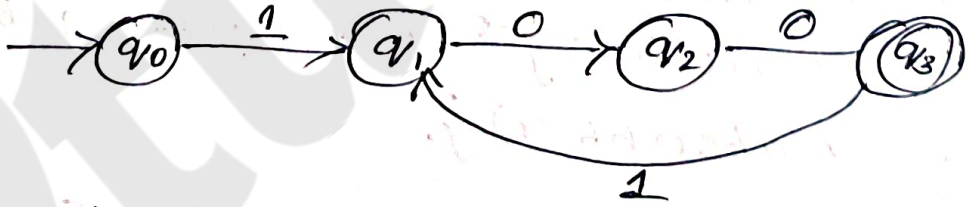


7. write NFA to accept the following language over $\{0, 1\}^*$

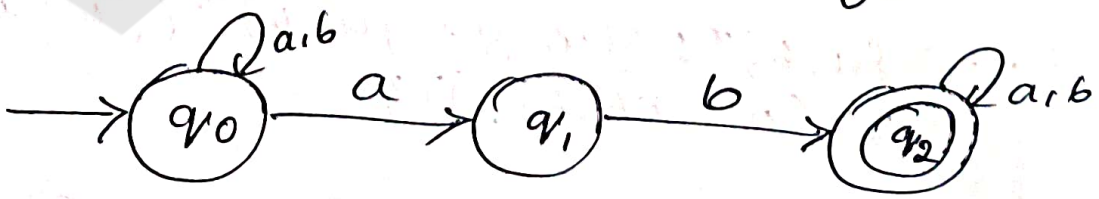
(i) Set of all strings such that containing either 101 or 110 as substring



(ii) Every 1 followed by 00

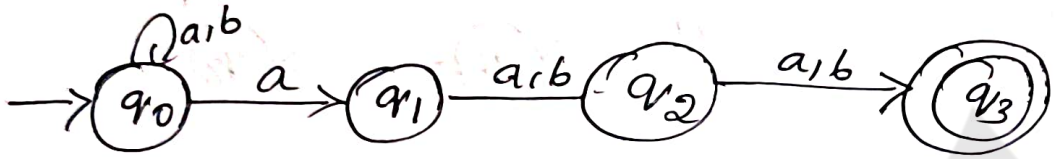


8. construct NFA to accept language over $\{a, b\}$, containing substring ab



9) write NFSM to accept language
$L = \{ w \in \{a,b\}^* \mid 3^{rd}$ character from right is a $\}$

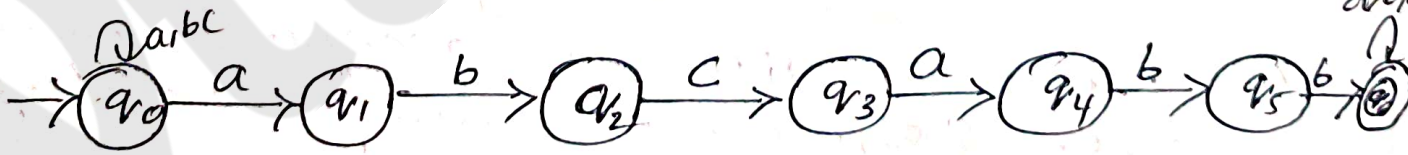
$L = \{abb, bbbbabb, ababb, aaba, aaaaa, ababa \cdots\}$



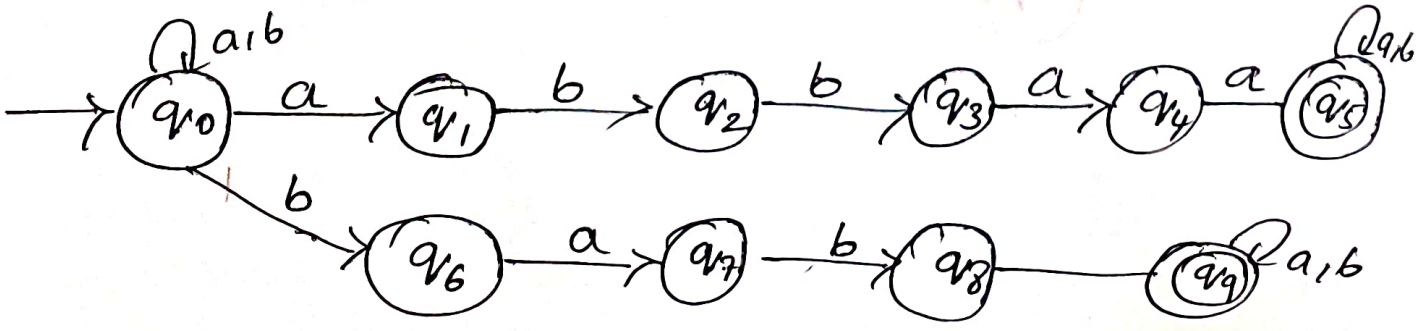
10) write NFSM to recognise the language
$L = \{ w \in \{a,b\}^*$ w is made up of an optional a followed by aa, zero or more b's $\}$

$L = \{aa, aaa, aab, aaab, aabbb, aaabbb \cdots\}$

11) construct NFSM where
$L = \{ w \in \{a,b,c\}^* : \exists\, x,y \in \{a,b,c\}^*$
$\{ w = x\,abcabb\,y) \}$.



(12) $L = \{ w \in \{a,b\}^* : \exists\, x,y \in \{a,b\}^* ((w = xabbaay) \vee (w = xbabay)) \}$



Finite Automata with ϵ transitions

ϵ-NFA Epsilon NFA

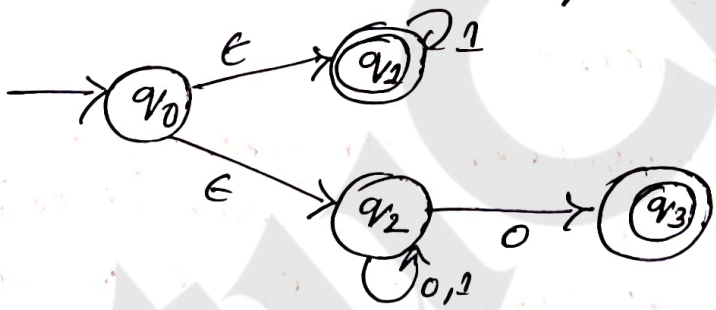
The NFA which includes transitions on the empty input ϵ is called ϵ NFA

* ϵ-NFA is a 5 tuple or quintuple

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

* ϵ-closure of a state, is a set of all vertices p such that there is a path from q to p labelled ϵ.

Ex: Draw ϵ-NFA that accept all binary string where the last symbol is 0 or that contain only 1's



ϵ-closure(q_0) = { q_0, q_1, q_2 }

ϵ-closure(q_1) = { q_1 }

ϵ-closure(q_2) = { q_2, q_3 }

ϵ-closure(q_3) = { q_3 }

Conversion from NFA to DFA (Subset construction)

* With an NFA, at any point in a scanning of the input string we may be faced with a choice of any number of paths to be followed to reach final state.
* With DFA there is never a choice of paths so when we construct a ~~DFA~~ DFA which accepts the same language as a particular NFA, the conversion process effectively involve merging all possible states which can be reached on a particular input character, from a particular state, into a single, composite, state which represent all those paths.

Steps in conversion:

Step1 : compute epsilon closure for all the state (handling ϵ transition)

Step2 : Find out ϵ-closure of old closure state (new starting state)

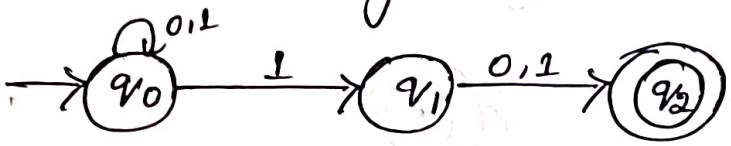
Step3 : Start finding new active state
Repeat step 3 until we dont get no new state.

Step4 : find final active states.

Step5 : write original DFSm for NDFSm using active states.

(21)

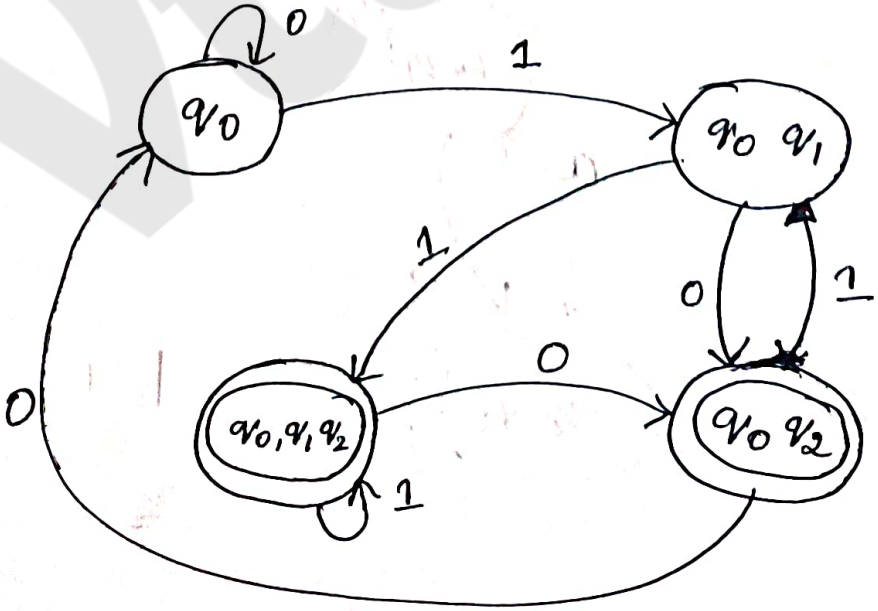
(1) convert the given NFA to DFA



Sol:

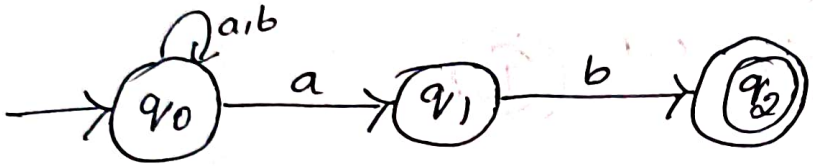
δ_{NFA}	0	1
→ q_0	$\{q_0$	$\{q_0, q_1\}$
q_1	q_2	q_2
* q_2	$-(\phi)$	$-(\phi)$

δ_{DFA}	0	1
→ q_0	q_0	$\{q_0, q_1\}$
$\{q_0 q_1\}$	$\{q_0 q_2\}$	$\{q_0, q_1, q_2\}$
* $\{q_0 q_2\}$	q_0	$\{q_0, q_1\}$
* $\{q_0, q_1, q_2\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$



22

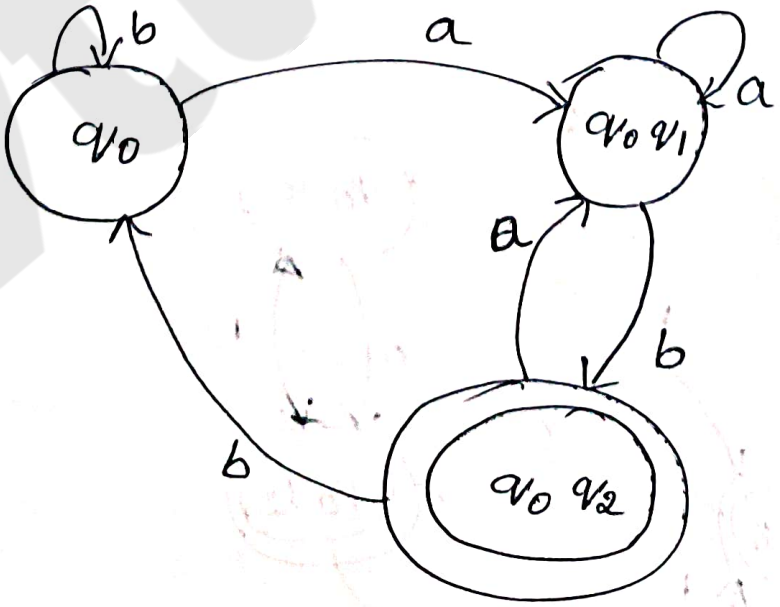
(2) convert the following NFA to DFA



Sol given

δ_{NFA}	a	b
$\rightarrow q_0$	$\{q_0\ q_1\}$	q_0
q_1	ϕ	q_2
$*q_2$	ϕ	ϕ

δ_{DFA}	a	b
q_0	$\{q_0\ q_1\}$	q_0
$\{q_0\ q_1\}$	$\{q_0\ q_1\}$	$\{q_0\ q_2\}$
$*\{q_0\ q_2\}$	$\{q_0\ q_1\}$	$\{q_0\}$



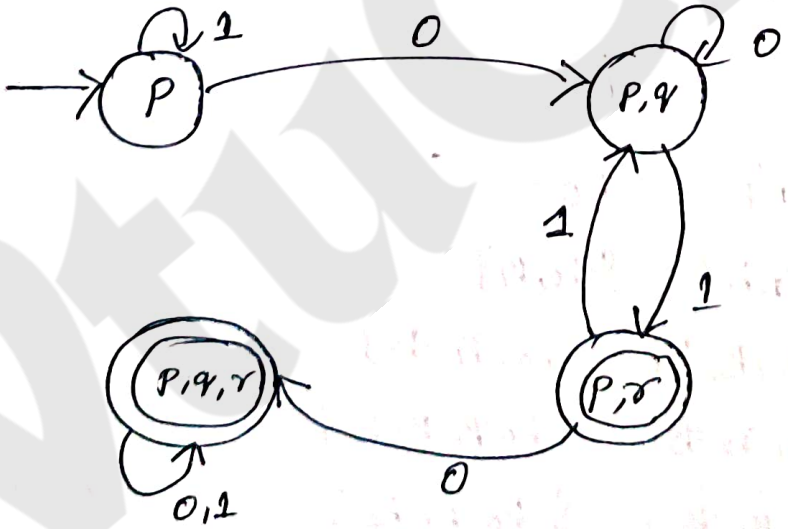
(23)

3) convert the following NFA to DFA

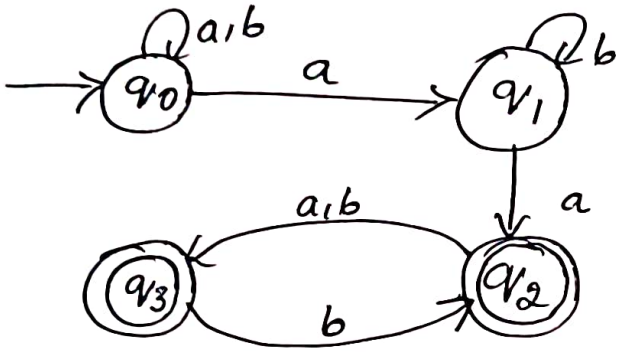
δ_{NFA}	0	1
→ p	{p, q}	p
q	ϕ	r
* r	{p r}	q

Sol

δ_{DFA}	0	1
→ p	{p, q}	p
{p, q}	{p, q}	{p, r}
* {p, r}	{p, q, r}	{p, q}
* {p, q, r}	{p, q, r}	{p, q, r}



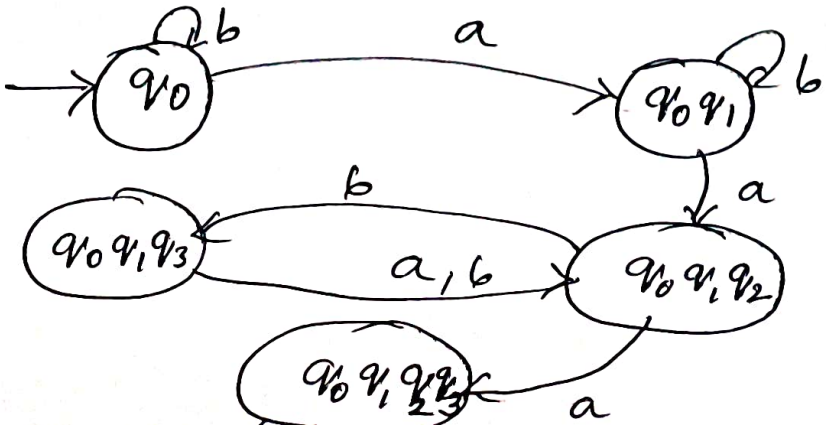
④ convert the following ε-NFA to DFA



Sol

δ_{NFA}	a	b
→ q_0	$\{q_0 q_1\}$	q_0
q_1	q_2	q_1
* q_2	q_3	q_3
* q_3	—	q_2

δ_{DFA}	a	b
→ q_0	$\{q_0 q_1\}$	q_0
$\{q_0,q_1\}$	$\{q_0,q_1,q_2\}$	$\{q_0,q_1\}$
$\{q_0,q_1,q_2\}$	$\{q_0,q_1,q_2,q_3\}$	$\{q_0,q_1,q_3\}$
$\{q_0q_1q_2q_3\}$	$\{q_0q_1q_2q_3\}$	$\{q_0q_1q_3q_2\}$
$\{q_0q_1q_3\}$	$\{q_0,q_1,q_2\}$	$\{q_0q_1q_2\}$



(5) convert the given ∈ - NFA to DFA

∈-NFA	a	b	c	∈
→ q_0	q_0	ϕ	ϕ	q_1
q_1	ϕ	q_1	ϕ	q_2
* q_2	ϕ	ϕ	q_2	ϕ

∈-closure(q_0) = $\{q_0, q_1, q_2\}$

∈-closure(q_1) = $\{q_1, q_2\}$

∈-closure(q_2) = $\{q_2\}$

DFA	a	b	c
→ * $\{q_0 q_1 q_2\}$	$\{q_0 q_1 q_2\}$	$\{q_1 q_2\}$	q_2
* $\{q_1 q_2\}$	ϕ	$\{q_1, q_2\}$	$\{q_2\}$
* $\{q_2\}$	ϕ	ϕ	q_2

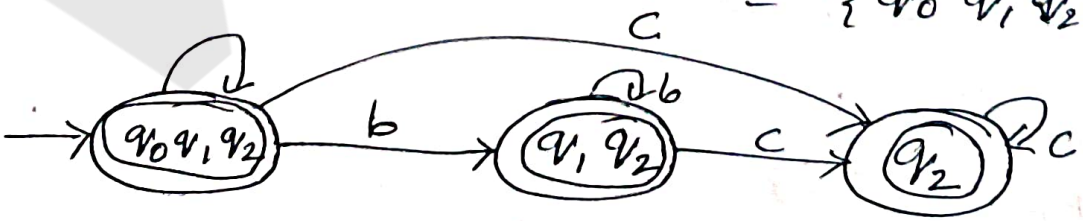
$\delta_D(\{q_0 q_1 q_2\}, a) = \in\text{-closure}(\delta(q_0, q_1, q_2\}, a)$

$= \in\text{-closure}(\delta_E(q_0, a) \cup \delta_E(q_1, a) \cup \delta_E(q_2, a)$

$= \in\text{-closure}\{\{q_0\} \cup \phi \cup \phi)$

$= \in\text{-closure}(q_0)$

$= \{q_0 q_1 q_2\}$

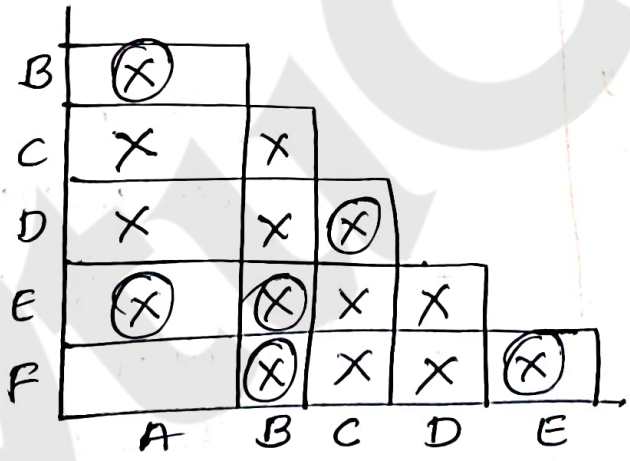


Minimization of DFA (Table filling method)

δ	0	1
→ A	B	C
B	D	B
* C	E	F
* D	D	B
E	A	B
F	B	C

$\{A, B, C, E, F\} \notin F$

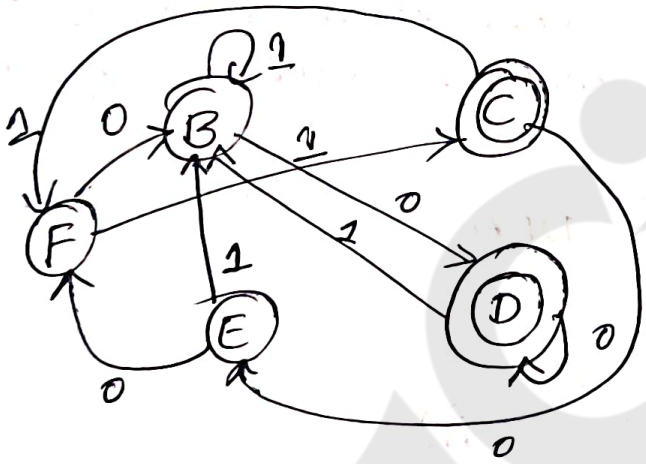
$\{C, D\} \in F$



	0	1	
{AF}	{B, B}	{C, C}	= E
{AE}	{B, A}	{C_F, B_N}	= NE
{AB}	{B D}	{C, B}	= NE
{BF}	{D B}	{B, C}	= NE
{BE}	{D, A}	{B, B}	= NE
{C D}	{E, D}	{F, B}	= NE
{E, F}	{A, B}	{B, C}	= NE

A = F

δ	0	1
→ ~~F~~	B	C
B	D	B
* C	E	F
* D	D	B
E	~~F~~	B
F	B	C



Minimization of DFA means reducing the number of states from given FA

Step1: Remove all the state that one unreachable from initial state via any set of transition of DFA

Step2: Draw transition table for all pair of states

Step3: Now split the transition table into two table T_1, T_2

T_1 contains all final states

T_2 contains non final states

Step4 : Find similar row from T such that

$\delta(q_1, a) = p$

$\delta(r, a) = p$

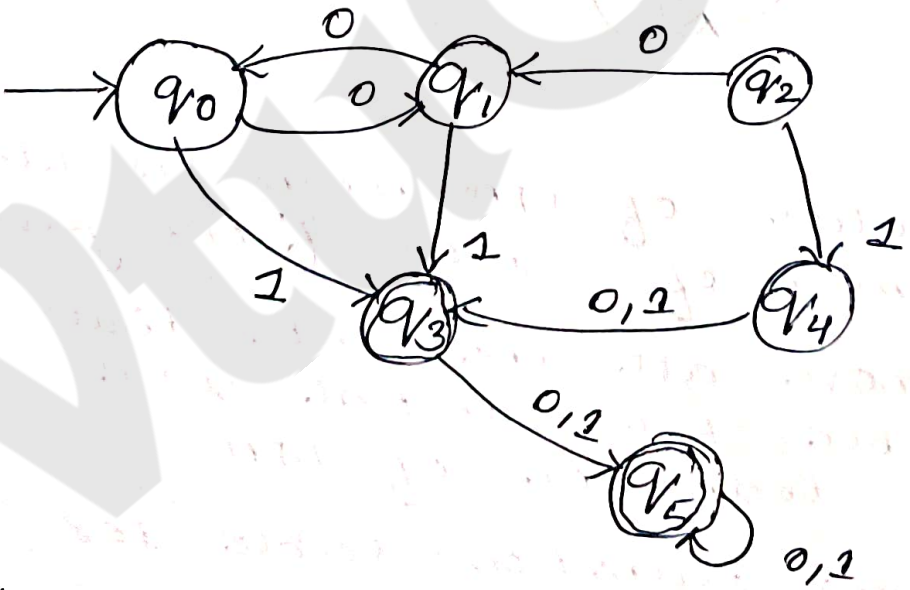
so find two states which have same value of a & b & remove one of them.

Step5 : Repeat step 3 until we find no similar rows available in T

Step6 : Repeat step3 & step4 for table T_2 also

Step7 : Now combine the reduced T_1 & T_2 table is the final transition table of minimize DFA

② minimize given DFA



Step1 : Remove q_2 and q_4 in finite automata

unreachable state

Step2 Draw transition table for the rest of the state

State	0	1
→ q_0	q_2	q_3
q_1	q_0	q_3
* q_3	q_5	q_5
* q_5	q_5	q_5

(iii) ~~the~~

Step3 :(i) Table which starts from non final state

	0	1
q_0	q_1	q_3
q_1	q_0	q_3

(ii) table which starts from final state

	0	1
* q_3	q_5	q_5
* q_5	q_5	q_5

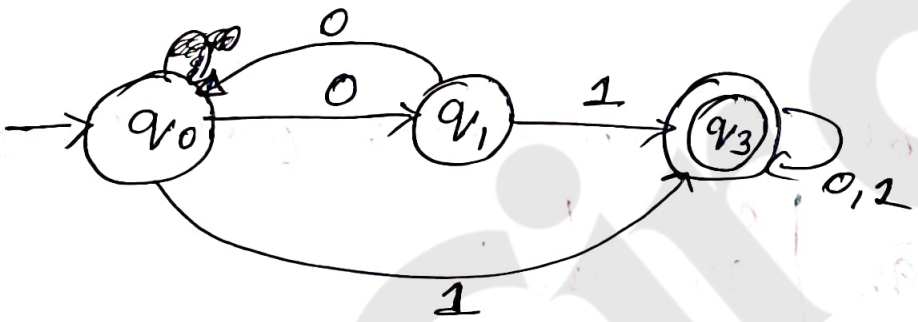
Step4 :(i) set1 has no similar rows, they will be same

(ii) set2 has similar rows, so skip q_5 & replace q_5 by q_3

	0	1
q_3	q_3	q_3

Step6: combine set1 & set2

State	0	1
q_0	q_1	q_3
q_1	q_0	q_3
$*q_3$	q_3	q_3



Distinguishable and Indistinguishable

$\delta(p, w)$ & $\delta(q, w) \in F$

$\delta(p, w)$ & $\delta(q, w) \notin F$

one state move to final other state move to non final state called as distinguishable state.