TAFL - Theory of Automata and Formal Language

- **TAFL**: Automata theory is a subject which describes the behaviour of automatic machines mathematically.

Applications:-
- Digital circuit design
- Compiler design
- For designing the reduced mathematical model.

- Circuit design is possible if mathematical model exists.

- **Language**: Means of communication
- **Program**: Sequence of instructions
- **String**: Sequence of symbols
- **Sentence/instructions**: An ordered combination of word or string that makes complete sense

Why do we study theory?
Theory provides concepts and principles that help us understand the general nature of discipline

Alphabet set(Σ): Set of consisting of symbols or characters.

Eg: $\Sigma = \{0, 1\} = (0+1)$
0 or 1

Length of string ($|w|$): $w = abbcde$, $|w| = 6$

Null string (ϵ or λ): The string of length 0.

$|w| = 0$ $|\lambda| = 0$ $\epsilon = \{\}$

Cocatenation: $w_1 = 010$ $w_2 = 1011$

$w = w_2 w_1 = 1011010$

→ Cocatenation of string does not support commutative property but support associative property.

$L_1 \cdot L_2 = \{ w_1 \cdot w_2 \,/\, w_1 \in L_1 \ \& \ w_2 \in L_2 \}$

Kleen Closure (Σ^*): (all possible string)

Σ^* = Universal language over alphabet Σ

$$\boxed{|\Sigma^*| = \infty}$$

Positive Closure (Σ^+)

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

NOTE: $(01)^* = \{\epsilon, 01, 0101, 010101, \ldots\}$

$(01)^+ = \{01, 0101, 010101, \ldots\}$

$w = 010$

$(010)^0 = \epsilon$ length = 0

$(010)^1 = 010$ length = 3

$(010)^2 = 010010$ length = 6

* Symbol (* or +) with Σ is universal

$l = w$ given

$|w| = m$

Find nth length of w

$= m \times n$

Reverse

$w = a_1, a_2, a_3 \ldots a_n$

$w^R = a_n, a_{n-1}, \ldots a_2, a_1$

$w = XY$

$w^R = (XY)^R = Y^R X^R$

Palindrome

$ww^R \rightarrow$ even $\quad w = 1011$

10111101

$w\#w^R \rightarrow$ odd $\quad w = 101 \quad 10101$

Substring

Any sequence of consecutive symbols from w (anystring)

w = dog

Sub(w) = {ε, d, o, g, do, og, dog}

But dg is not a part of sub-string. It will be a part of power set.

Non-null sub-string:

Set of substring of any string w except null string i.e. ε.

Sub(w) − {ε} $\quad$ {d, o, g, do, og, dog}

Proper substring:

Set of all possible substring of w except string itself.

sub(w) − {w}

{ε, d, o, g, do, og}

Non-null Proper substring:

sub(w) − {ε, w}

{d, o, g, do, og}

Number of Substring →

1. Symbols are distinct

a = 1 + 1

ab = 3 + 1

abc = 6 + 1

$$\boxed{\frac{l(l+1)}{2} + 1}$$

2. Symbols are same

$a = 1+1$

$aa = 2+1$ $\boxed{l+1}$

$aaa = 3+1$

Prefix and Suffix

Prefix of any string is nothing but set of conjugate symbol from left to right in given string.

$\left.\begin{array}{l} \text{Prefix}(w) = \{\varepsilon, d, do, dog\} \\ \text{Suffix}(w) = \{\varepsilon, g, og, dog\} \end{array}\right\}$ $(l+1)$ no. of suffix & prefix

ε & ω are present in both

Power of string

$w^0 = \varepsilon$

$w^1 = w^1 \cdot w^0 = w^1 \cdot \varepsilon = w$

$w^2 = w^1 \cdot w^1 = w \cdot w$

$w^3 = w^2 \cdot w^1 = w w \cdot w$

Operations on languages:

$L_1 = \{0, 10, 11\}$

$L_2 = \{11, 101, 110\}$

$L_1 \cup L_2 = \{0, 10, 11, 101, 110\}$

$L_1 \cap L_2 = \{11\}$

$L_1^c = \Sigma^* - L_1$

$= (0+1)^* - L_1$ $\{\varepsilon, 1, 01, 00, \ldots \infty\}$

$L_1^R = \{W^R, w \in L\}$

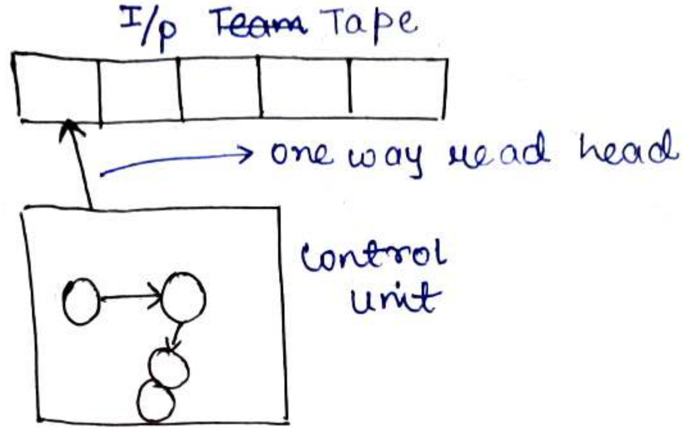
$L_2^R = \{11, 101, 011\}$

$L = L_1 L_2 = \{ \omega_1 \omega_2 \,/\, \omega_1 \in L_1 \;\&\; \omega_2 \in L_2 \}$

$$\boxed{|L_1 L_2| \leq |L_1| \, |L_2|}$$

Finite Automata

Architecture of Automata



Finite Automata (FA) is a quin(5) tuple machine.

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

where $Q \rightarrow$ Finite set of state

$\Sigma \rightarrow$ Finite set of character

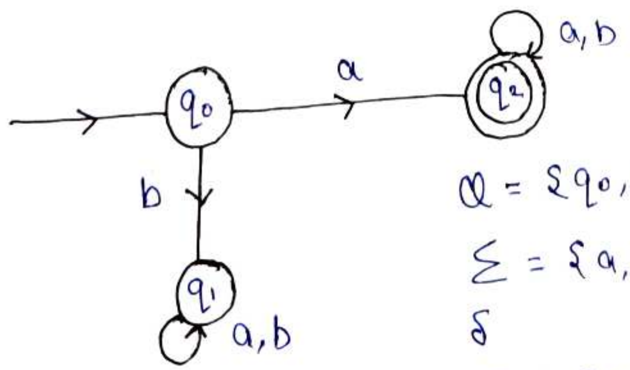
$\delta \rightarrow$ Transition function

$q_0 \rightarrow$ Single Initial State

$q_0 \in Q$

$F \rightarrow$ Finite set of Final State

$F \subseteq Q$



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

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

δ

$q_0 = \{q_0\}$

$F = \{q_2\}$

$\delta : Q \times \Sigma \rightarrow Q \rightarrow$ dfa

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

$\delta : (q_0, a) \rightarrow q_2$
$\delta : (q_0, b) \rightarrow q_1$
$\delta : (q_1, a) \rightarrow q_1$
$\delta : (q_1, b) \rightarrow q_1$
$\delta : (q_2, a) \rightarrow q_2$
$\delta : (q_2, b) \rightarrow q_2$

Difference b/w nfa and dfa

dfa	nfa
(i) No choices are allowed.	(i) Choices are allowed
(ii) ϵ-transition not allowed	(ii) ϵ-transition are allowed
(iii) Dead configuration is not allowed.	(iii) Dead configuration is allowed.

The Maximum & Minimum language over Σ.

$\Sigma = \{0, 1\}$
Maximum language $= \Sigma^* = \{0,1\}^* = (0,1)^* = (0+1)^*$
Minimum language $= \phi = (\text{Max } L)^c$

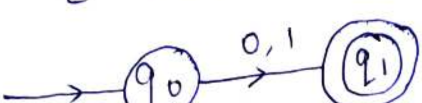
* Cascading of transition function

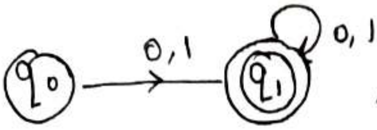
$\hat{\delta}(q_0, 0111) = \delta(\delta(\delta(\delta(q_0, 0), 1), 1), 1)$

Practices for dfa & nfa

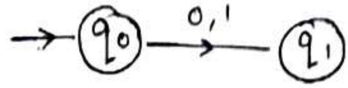
Ex 1: Construct a finite automata for the language over alphabet $\Sigma = \{0, 1\}$ which consist string of exactly length 1.

$\Sigma = \{0, 1\}$
$L = \{0, 1\} = (0+1)^1$

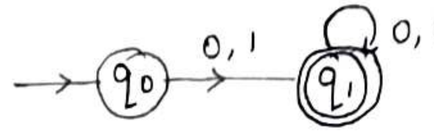




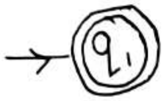
X



✓ (nfa)



✓ (dfa)

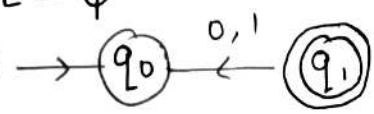


X

Invalid machine

Valid machine

→ $L = \phi$

(nfa) 

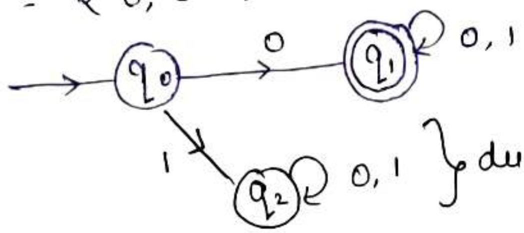
 (nfa/dfa)

Final state are not there or final state are not reachable from initial state.

Ex 2: Construct a finite automata for the language over alphabet $\Sigma = \{0,1\}$ in which every string must start from symbol 0.

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

$L = \{0, 00, 01, 000, 001, 010, 011, \ldots\}$



} dummy state

$L = 0 \cdot (0+1)^*$

$= \{0\} \{\epsilon, 0, 1, 00..\}$

$= \{0, 00, 01, \ldots\}$

→ $0^* = \{\epsilon, 0, 00, 000, 0000, \ldots\}$

$0^+ = \{0, 00, 000, \ldots\}$

$\boxed{0 0^* = 0^* 0 = 0^+}$

If the base of the language is m & you are looking for universal language over that base Σ, then counting of any random l length string is given by m^n

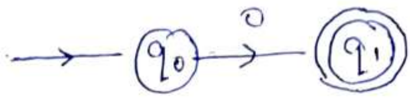
→

$\rightarrow \boxed{0^* 0^* = 0^*}$ $\rightarrow \boxed{0^+ + \{\epsilon\} = 0^* = \{\epsilon\} + 0^+}$

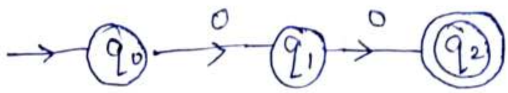
$0^* 0^* = \{\epsilon, 0, 00, 000, \dots\}\{\epsilon, 0, 00, 000, \dots\}$
$= \{\epsilon, 0, 00, 000, \dots\}$
$= 0^*$

$\rightarrow (0^+ 0^+ - 0^+)_n = (00 - 0)_{n-1}\, 0^+$

$L = \{0\}$



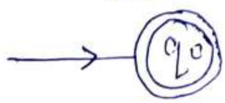
$L = \{00\}$



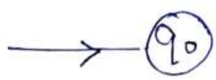
$L = \{0, 00\}$

$L = \{0, 1\}$

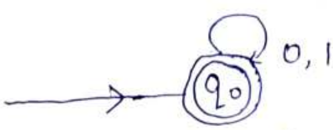
$L = \{\epsilon\}$



$L = \phi$

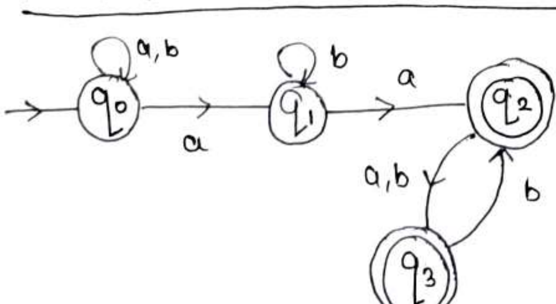


$L = \{0+1\}^*$
$= \{\epsilon, 0, 1, \dots\}$



$L = \{0+1\}^+$

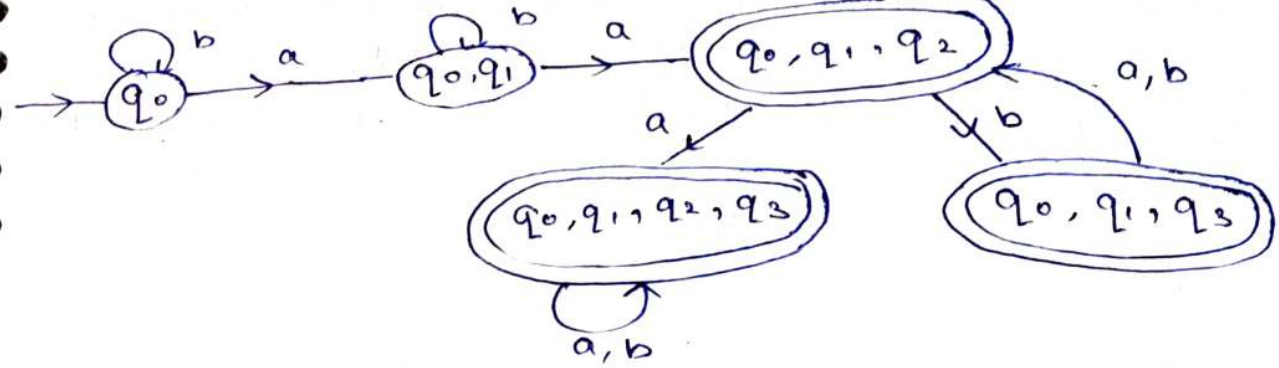
NFA (without ∈ transition) to DFA conversion



	a	b
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	q_2	q_1
q_2	q_3	q_3
q_3	ϕ	q_2

	a	b
q_0	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1, q_2\}$	$\{q_0, q_1\}$
$\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2, q_3\}$	$\{q_0, q_1, q_3\}$
$\{q_0, q_1, q_2, q_3\}$	$\{q_0, q_1, q_2, q_3\}$	$\{q_0, q_1, q_2, q_3\}$

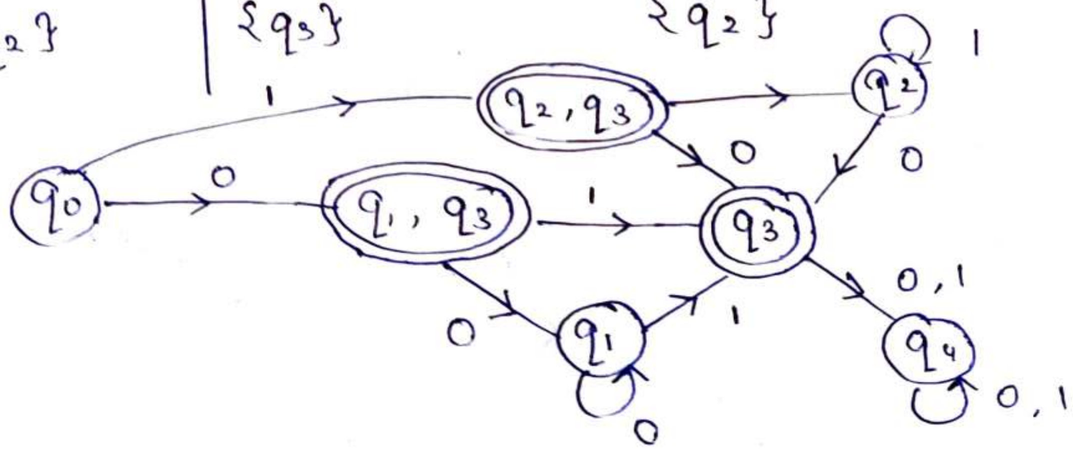
	a	b
$\{q_0, q_1, q_3\}$	$\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$



Convert given NFA into DFA

state	0	1
→ q_0	$\{q_1, q_3\}$	$\{q_2, q_3\}$
q_1	q_1	q_3
q_2	q_3	q_2
* q_3	—	—

state	0	1
→ q_0	$\{q_1, q_3\}$	$\{q_2, q_3\}$
* $\{q_1, q_3\}$	$\{q_1\}$	$\{q_3\}$
* $\{q_2, q_3\}$	$\{q_3\}$	$\{q_2\}$
$\{q_1\}$	$\{q_1\}$	$\{q_3\}$
* $\{q_3\}$	—	—
$\{q_2\}$	$\{q_3\}$	$\{q_2\}$

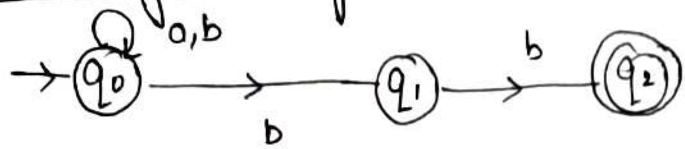


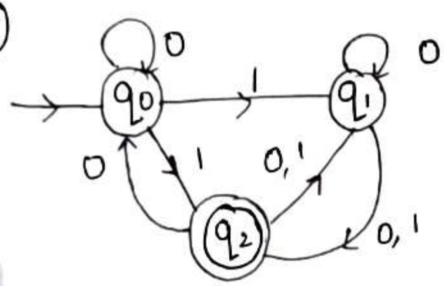
Dummy or trap state

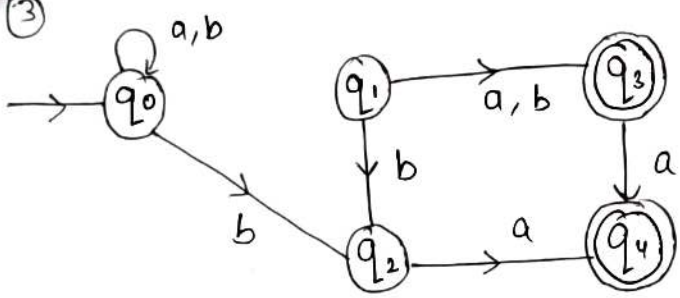
- Dummy state will always be in dfa. Dummy state will always be exactly one state in any machine. Dummy state will be non-initial & non-final state

• Dummy state must have one loop for all the input symbols for which machine is defined.

Convert following NFA machine into equivalent DFA.

①


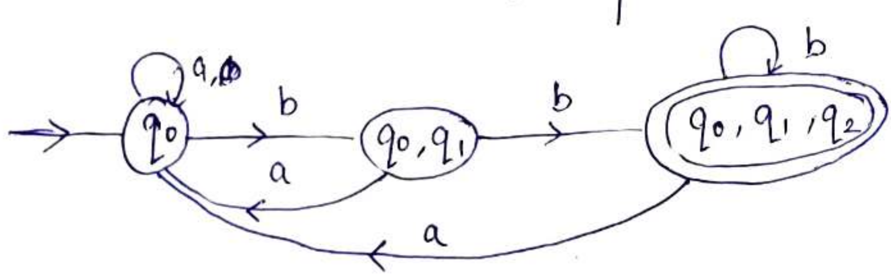
②


③


①

	a	b
→ q_0	{q_0	{q_0, q_1}
q_1	-	{q_2}
q_2	-	-

	a	b
→ q_0	{q_0}	{q_0, q_1}
{q_0, q_1}	{q_0}	{q_0, q_1, q_2}
* {q_0, q_1, q_2}	{q_0}	{q_0, q_1, q_2}



②

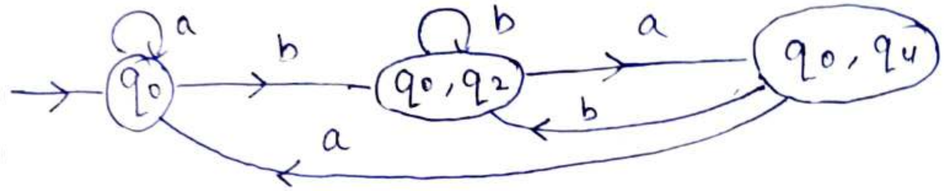
	0	1
→ q_0	q_0	{q_2, q_1}
q_1	{q_1, q_2}	q_2
q_2	{q_0, q_1}	q_1

	0	1
→ q_0	q_0	{q_1, q_2}
{q_1, q_2}	{q_0, q_1, q_2}	{q_1, q_2}
{q_0, q_1, q_2}	{q_0, q_1, q_2}	{q_1, q_2}

(3)

	a	b
$\rightarrow q_0$	q_0	$\{q_0, q_2\}$
q_1	q_3	$\{q_2, q_3\}$
q_2	q_4	-
q_3	q_4	-
q_4	-	-

	a	b
$\rightarrow q_0$	q_0	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_4\}$	$\{q_0, q_2\}$
$\{q_0, q_4\}$	q_0	$\{q_0, q_2\}$

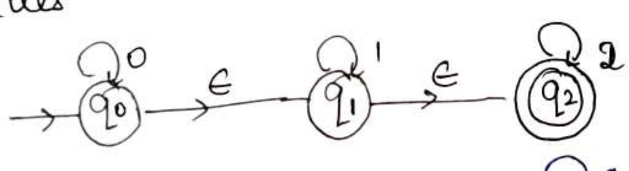


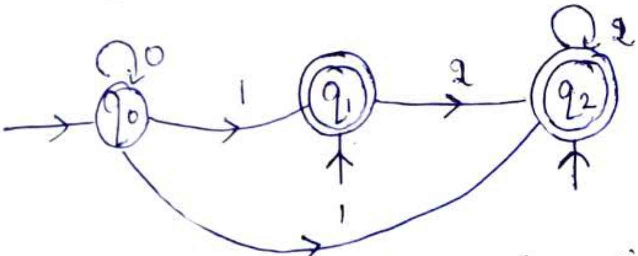
nfa with ϵ-transition to dfa conversion

Suppose we want to replace ϵ move from vertex v_1 to v_2 then processed as follow-

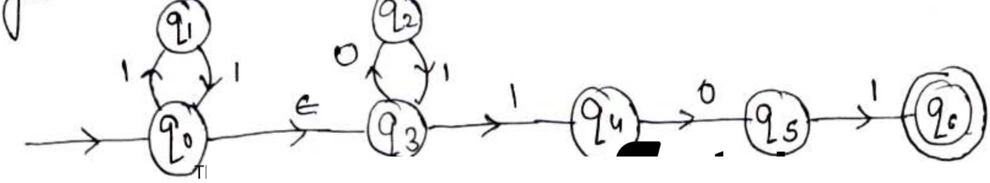
1. Find all the edges starting from vertex V_2.
2. Duplicate all these edges starting from v_1 without changing the edges levels.
3. If v_1 is an initial state then make v_2 also as initial. If v_2 is in final state, make v_1 as the final state.

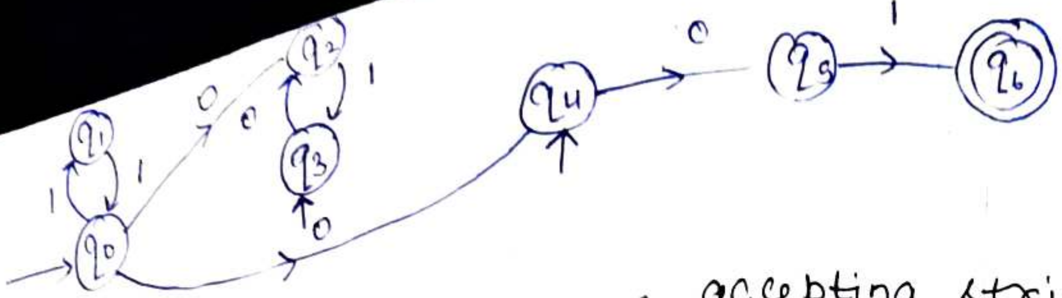
Ques-





Q. Convert the given nfa with ϵ-transition to nfa without ϵ-transition.

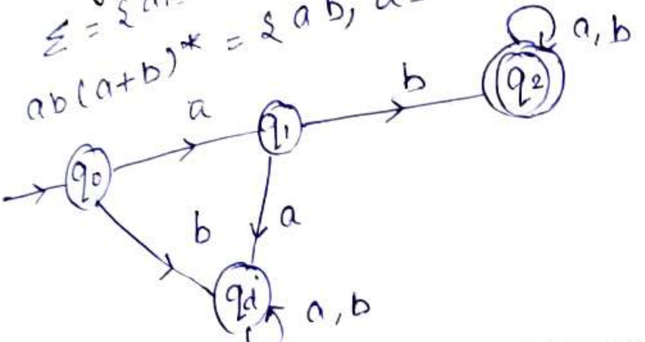




Draw a dfa for the language accepting string starting with ab over input alphabet $\Sigma = \{a, b\}$

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

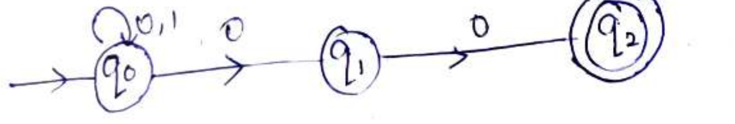
$ab(a+b)^* = \{ab, aba, abb, abaa, \ldots\}$



Draw a dfa for the language over alphabets $\Sigma = \{0, 1\}$ such that every string of l must have 00 as a substring.

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

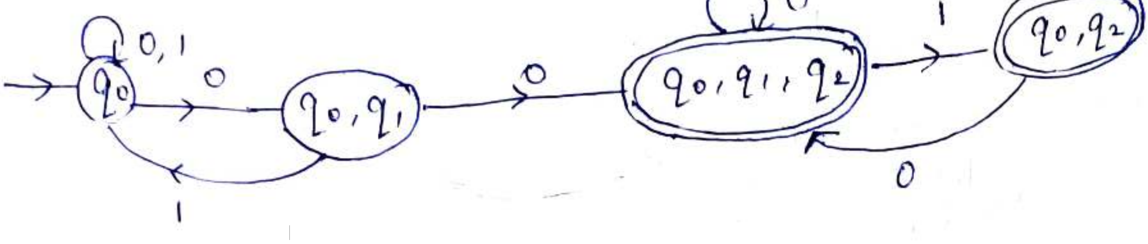
$(0+1)^* 00 (0+1)^* = \{00, 000, 100, 001, \ldots\}$

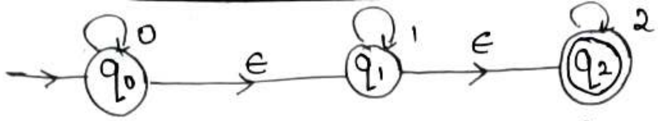


	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	q_0
q_1	q_2	–
q_2	q_2	q_2

$\Rightarrow$

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



Q. 

$$\begin{cases} 0\text{-closure}(q_0) = \{q_0, q_1, q_2\} \\ 1\text{-closure}(q_0) = \{q_1, q_2\} \\ 2\text{-closure}(q_0) = \{q_2\} \end{cases}$$

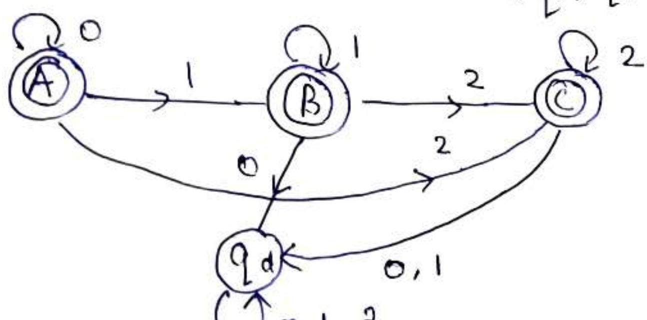
$$\begin{cases} 0\text{-closure}(q_1) = \phi \\ 1\text{-closure}(q_1) = \{q_1, q_2\} \\ 2\text{-closure}(q_1) = \{q_2\} \end{cases}$$

$$\begin{cases} 0\text{-closure}(q_2) = \phi \\ 1\text{-closure}(q_2) = \phi \\ 2\text{-closure}(q_2) = \{q_2\} \end{cases}$$

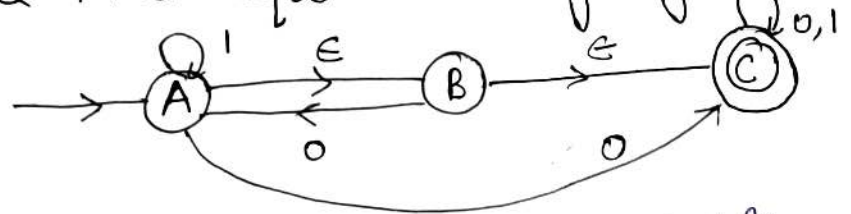
		0	1	2
$A \rightarrow$	$\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
B	$\{q_1, q_2\}$	–	$\{q_1, q_2\}$	$\{q_2\}$
C	$\{q_2\}$	–	–	$\{q_2\}$

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

0-closure $(q_0, q_1, q_2) =$ 0-closure $(q_0) \cup$
0-closure $(q_1) \cup$
0-closure (q_2)
$= \{q_0, q_1, q_2) \cup$
$\phi \cup \phi$
$= \{q_0, q_1, q_2\}$



Q. Find equivalent dfa for the given nfa -



$$\begin{cases} 0\text{-closure of } A = \{A, B, C\} \\ 1\text{-closure of } A = \{A, B, C\} \end{cases}$$

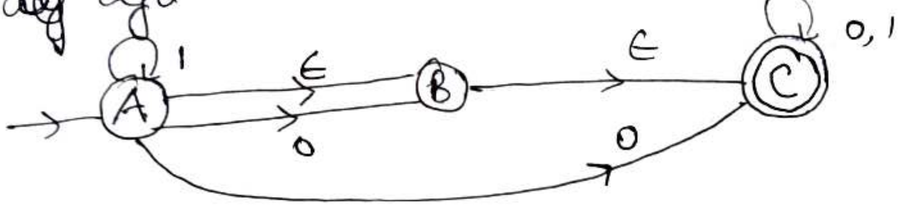
0-closure of $B = \{A, B, C\}$

0-closure of C = {C}
1-~~closure~~ closure of C = {C}
ε-closure (A) = {A, B, C}

	0	1
$q_0' \rightarrow$ {A, B, C}	{A, B, C}	{A, B, C}

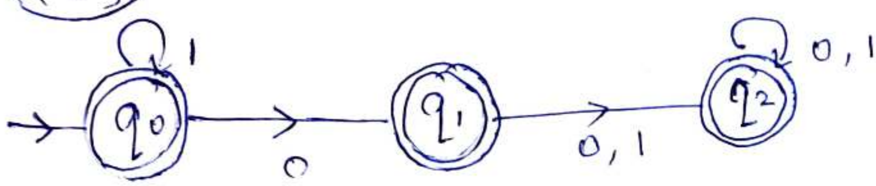
$= (0+1)^*$

Convert the given nfa machine into equivalent ~~of~~ dfa machine.



0-closure of A = {B, C}
1-closure of A = {A, B, C}
0-closure of B = {C}
1-closure of B = {C}
0-closure of C = {C}
1-closure of C = {C}
ε closure (A) = {A, B, C}

	0	1
$q_0 \rightarrow$ {A, B, C}	{B, C}	{A, B, C}
q_1 {B, C}	{C}.	{C}
q_2 {C}	{C}	{C}



Minimalization of ~~any~~ dfa

There is two algorithms to minimise any given dfa-

(i) Partitioning method

(ii) My hill ~~marodo~~ narode method

Partitioning Method

→ **Distinguishable & Indistinguishable state**

Two or more than two states (in any machine) are indistinguishable if for the same input, for those state we are getting output - all final or all non-final otherwise states are distinguishable.

Eg: If q_0 & q_1 are indistinguishable for input w then-

$$\begin{rcases} \delta(q_0, w) = F \\ \delta(q_1, w) = F \end{rcases} / \begin{matrix} NF \\ NF \end{matrix}$$

But if they are distinguishable for same input, we will get -

$$\begin{rcases} \delta(q_0, w) = NF \\ \delta(q_1, w) = F \end{rcases} / \begin{matrix} F \\ NF \end{matrix}$$

K-equivalence

Two or more than two states will be k-equivalent if and only if up to the length K for all possible string, states must be indistinguishable & it is denoted by π_K...

Q. Construct a min. state automata equivalent to dfa whose transition table is defined by -

State	a	b
→ q_0	q_1	q_2
q_1	q_4	q_3

$*$ q_4 — q_7 — q_6

q_5 — q_3 — q_6

q_6 — q_6 — q_6

q_7 — q_4 — q_6

Irreachable or dead state or inaccessible state

In any finite automata, if any state is not reachable from starting state then it is known as irreachable/dead/inaccessible state.

$\Pi_0 = \{\{q_0, q_1, q_2, q_5, q_6, q_7\}, \{q_3, q_4\}\}$

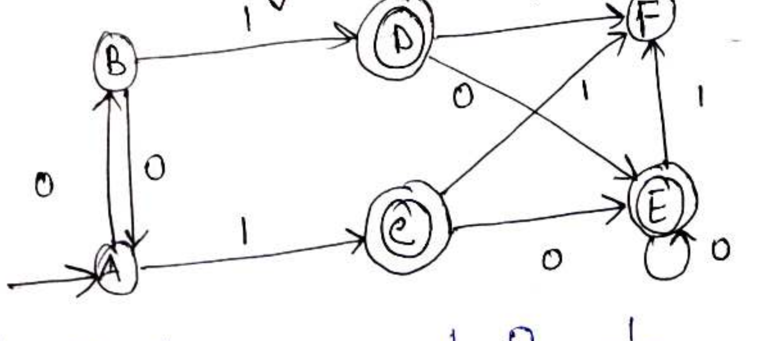
$\Pi_1 = \{\{q_0, q_6\}, \{q_1, q_2\}, \{q_3, q_4\}, \{q_5, q_7\}\}$

$\Pi_2 = \{\{q_0\}, \{q_6\}, \{q_1, q_2\}, \{q_3, q_4\}, \{q_5, q_7\}\}$

$\Pi_3 = \{\{q_0\}, \{q_6\}, \{q_1, q_2\}, \{q_3, q_4\}, \{q_5, q_7\}\}$

	a	b
$\rightarrow q_0$	q_1	q_2
q_6	q_6	q_6
$\{q_1, q_2\}$	q_4	q_3
$\{q_3, q_4\}$	$\{q_5, q_7\}$	q_6
$\{q_5, q_7\}$	$\{q_3, q_4\}$	q_6

Q. Minimise the given dfa



	0	1
$\rightarrow$ A	B	C
B	A	D
C	E	[illegible]

	0	1
D	E	F
E	E	F
F	F	F

$\Pi_0 = \{\{A, B, F\}, \{C, D, E\}\}$

$\Pi_1 = \{\{A, B\}, \{F\}, \{C, D, E\}\}$

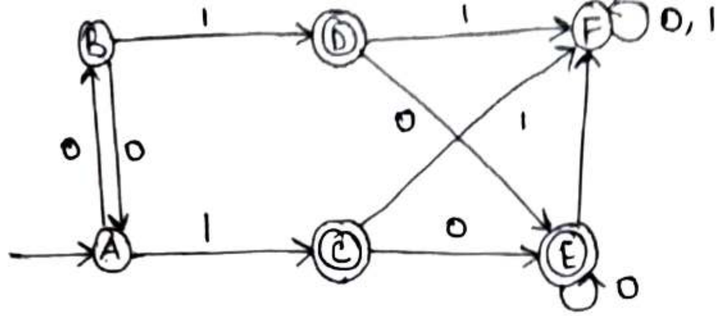
$\Pi_2 = \{\{A, B\}, \{F\}, \{C, D, E\}\}$

	0	1
$\{A, B\}$	$\{A, B\}$	$\{C, D\}$
$\{F\}$	$\{F\}$	$\{F\}$
$\{C, D, E\}$	$\{E\}$	$\{F\}$

dfa minimisation by Ma Hill Narode Theorem

- Draw a table for all pairs of states p, q.
- Mark all pairs where $P \in F$ & $Q \notin F$
- If there are any unmarked pair P, Q such that $\delta(P, x)$, $\delta(Q, x)$ is already marked then mark pair (P, Q) also.
- Repeat this process until no marking can be made.
- Combine all unmark pair & make them as a single state.

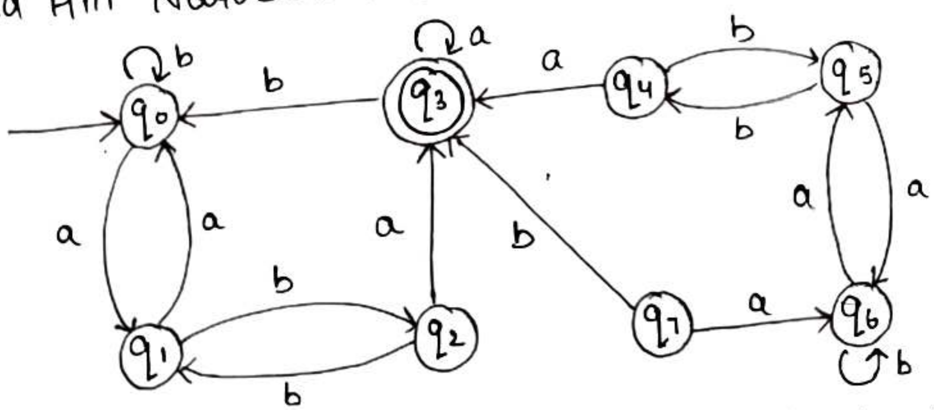
Q. Minimise the given dfa by using Ma Hill Narode Theorem.



	A	B	C	D	E	F
A						
B						
C	✓	✓				
D	✓	✓				
E	✓	✓				
F	✓	✓	✓	✓	✓	

$\{A, B\}$ $\underbrace{(C, D), (C, E), (D, E)}_{(C, D, E)}$

Q. Minimise the given dfa by using partioning & Mc Hill Narode method.



$|F| = \{1\}$, $|Q| = 8$, $\Sigma = \{a, b\}$, $|\Sigma| = 2$, $|q_0| = 1$

	a	b
$\rightarrow q_0$	q_1	q_0
q_1	q_0	q_2
q_2	q_3	q_1
q_3	q_3	

	a	b
q_4	q_3	q_5
q_5	q_6	q_4
q_6	q_5	q_6
q_7	q_6	q_3 ✗

q_7 is not a reachable state from initial state

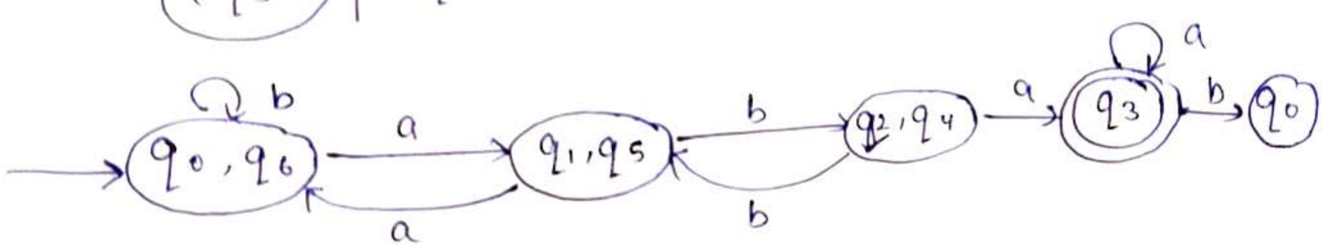
$\Pi_0 = \{\{q_0, q_1, q_2, q_4, q_5, q_6\}, \{q_3\}\}$

$\Pi_1 = \{\{q_0, q_1, q_5, q_6\}, \{q_2, q_4\}, \{q_3\}\}$

$\Pi_2 = \{\{q_0, q_6\}, \{q_1, q_5\}, \{q_2, q_4\}, \{q_3\}\}$

$\Pi_3 = \{\{q_0, q_6\}, \{q_1, q_5\}, \{q_2, q_4\}, \{q_3\}\}$

	a	b
$\rightarrow\{q_0, q_6\}$	$\{q_1, q_5\}$	$\{q_0, q_6\}$
$\{q_1, q_5\}$	$\{q_0, q_6\}$	$\{q_2, q_4\}$
$\{q_2, q_4\}$	$\{q_3\}$	$\{q_1, q_5\}$
$\{q_3\}$	$\{q_3\}$	$\{q_0\}$



Q. Minimise the given dfa by partioning & Ma Hill Narode method.