PAGE No
DATE : / / 202

11/01/24

→ Alphabet : Set of symbols where every symbol cannot be broken / divided further. It is denoted by 'Σ'.

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

Curly brackets are used as alphabet is a set.

→ String : Finite sequence of symbols from alphabets is known as string. (w)

eg : ϵ ⇒ NULL STRING (length = 0)

a, b ⇒ String of length 1

aa, ab, ba, bb ⇒ String of length 2

aaa, aab, ... ⇒ " 3

• prefix : Set of all front part of the string

eg :-

$w = aab$

prefix (w) = $\{\epsilon, a, aa, aab\}$

• suffix : Set of all front part of string taken in reverse order.

eg :-

suffix (w) = $\{\epsilon, b, ab, aab\}$

• cardinality : No. of elements in a set is known as cardinality. (⇒ n + 1)

~~card. of~~ ↳ string size

• substring : Any sequence of consecutive symbols in the string W.

eg : W = abc

Set of substring = { ϵ, a, b, c, ab, bc, abc }

Substrings (W)

cardinality = 7

⇒ for W = abcd

cardinality = 11

cardinality for substrings = $\frac{n(n+1)}{2} + 1$

Only holds, if string contains all distinct symbols

⇒ W = abcd

Sub-strings of length 4 : {abcd} ⇒ 1

" " " 3 : {abc, bcd} ⇒ 2

" " " 2 : {ab, bc, cd} ⇒ 3

" " " 1 : {a, b, c, d} ⇒ 4

" " " 0 : {ϵ} ⇒ 1

1+2+3+4 ⇓ $\frac{n(n+1)}{2}$ + 1

⇒ When a string contains all different symbols, the number of substrings that the string have will be $\left\{\frac{n(n+1)}{2} + 1\right\}$

⇒ ϵ is also a substring of each string.

PAGE No
DATE : / / 202

• Concatenation : Joing two or more strings.

eg : $x = aab$

$y = ba$

$x.y$ or $xy = aabba$

⇒ Not commutative

⇒

• Reversal : Writing a string right to left one by one. Denoted by w^R.

eg : $x = aab$

$x^R = baa$

• Properties of NULL string :-

i) $\epsilon . x = x . \epsilon = x$

ii) $^6\epsilon^9$, i.e., epsilon is a representation of NULL string.

12/01/24

→ Language : Set or collection of strings over a given alphabet.

→ Grammar : Set of rules used to describe strings of a given language.

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

L_1 : Set of all strings over Σ of length exactly 2.

$L_1 = \{aa, ab, ba, bb\}$

→

L_2 : Set of all strings over Σ of length exactly 3.

$L_2 = \{aaa, aab, aba, baa, bba, bab, abb, bbb\}$

⇒ If language contain finite no. of strings, that language is known as finite language.

L_3 : Set of all strings over Σ, starts with a

$L_3 = \{a, aa, ab, aaa, aab, aba, abb, aaaa, aaab, \ldots\}$

↓

Infinite Language

→ Powers of Sigma

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

Σ^1 : Set of all strings of length exactly one.

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

Similarly,

$\Sigma^2 = \{aa, ab, ba, bb\}$

$\Sigma^3 = \{aaa, aab, aba, abb, baa, bab, bba, bbb\}$

$\Sigma^0 = \{\epsilon\}$

Σ^n = Set of all strings of length exactly n.

$\Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots \cup \Sigma^n \Rightarrow$

$\{\epsilon, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$

Universal Set for $\Sigma = \{a, b\}$

⇒ Represented by Σ^* (Also called ~~King~~ Closure Clean)

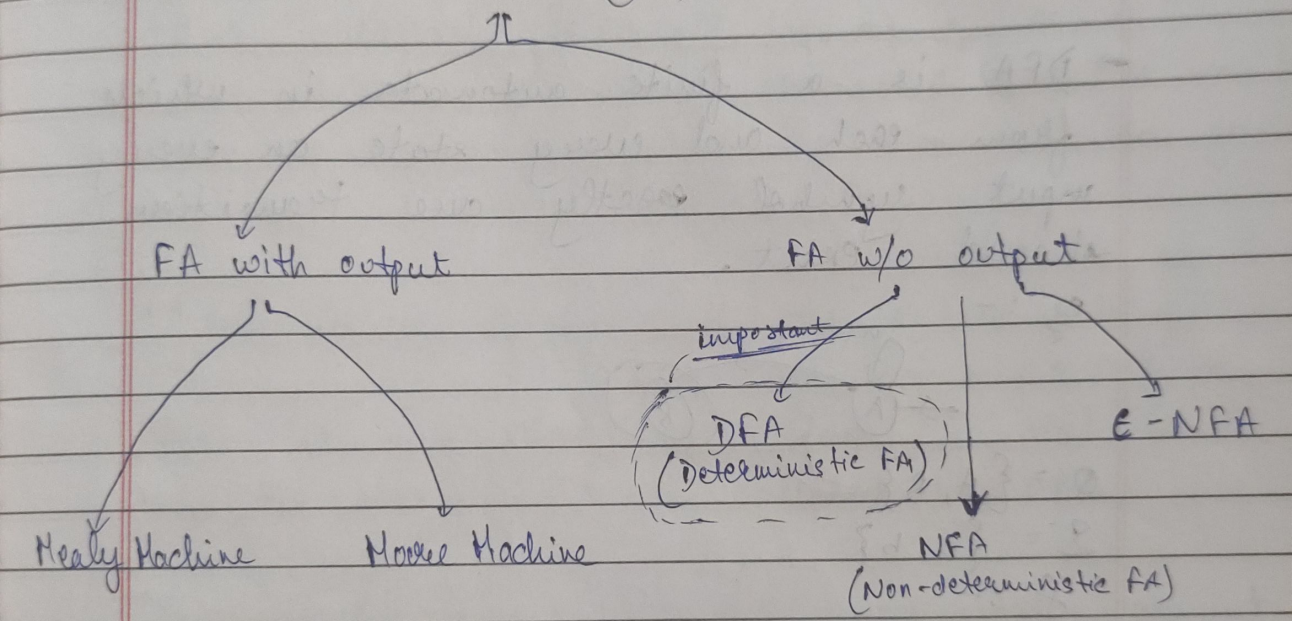
→ Finite Automata : It is a mathematical model, which contains finite no. of states and transitions defined b/w them.

states → Similar to what vertex is in a graph

transitions → Similar to what edge is in a graph

⇒ graph → transition diag.

→ FINITE AUTOMATA (FA)



• Notation of FA

Can be done using

i) Transit^n Diag.

ii) Transit^n Table

• DFA (Deterministic Finite Automata)

→ Defined as 5 tuples, i.e.,
$\{Q, \Sigma, \delta, q_0, F\}$

→ Q : Finite no. of states OR set of finite states

→ Σ : Set of symbols OR Alphabet OR input alphabet

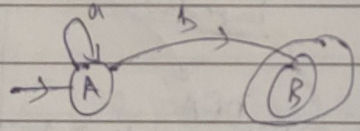
→ δ : Transition Func^n

→ q_0 : Initial state

→ F : Set of final states

→ DFA is a finite automata in which from each and every state on every input symbol exactly one transition should exist.

eg :-



$Q = \{A, B\}$

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

δ :-

i) $A, a \rightarrow A$

$A, b \rightarrow B$

$q_0 : A \Rightarrow$ { Representated by Arrow } "i" →

$F : \{B\} \Rightarrow$ { Representated by circle } "i" ◎

Initial state can be a final state.

eg : DFA of Σ^*

PAGE No
DATE : / 202

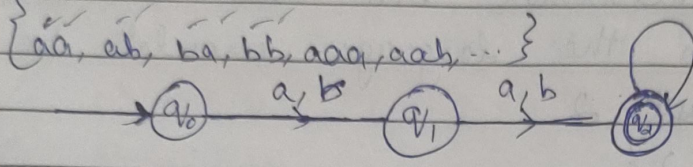
17/01/24

Q→ Construct a minimal DFA over alphabet {a, b}, where set of strings will be of length atleast 2, for every string in the language will be of length atleast 2.

→ $L = \{w \mid |w| \geq 2\}$

⇒ $L \cdot \Sigma = \{a, b\}$

⇒ $L = \{aa, ab, ba, bb, aaa, aab, \ldots\}$



** We would have done self loop on q_0 or q_1, but that would accept string ∈, a, b. As we have to reject these 3, hence we have to go to q_2.

** Acceptance of string : Scan the entire string from left to right, if we reach at any final state from initial state then the string will be accepted by DFA.

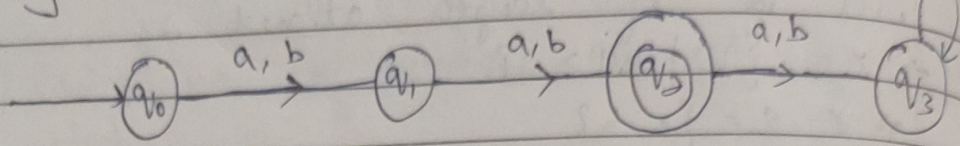
** Language Acceptance : Each and every string present in the language should be accepted by the DFA and all the strings that are not present in the language must be rejected.

** Only way to accept ∈ is to make the initial state as final state.

** Minimal DFA is DFA having minimum no. of states.

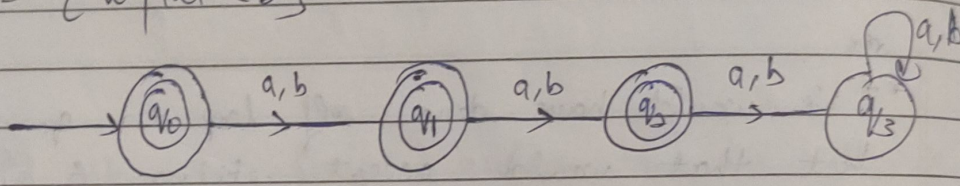
Q→ Construct a minimal DFA over alphabet {a,b} where every string in the language of length exactly 2.

→



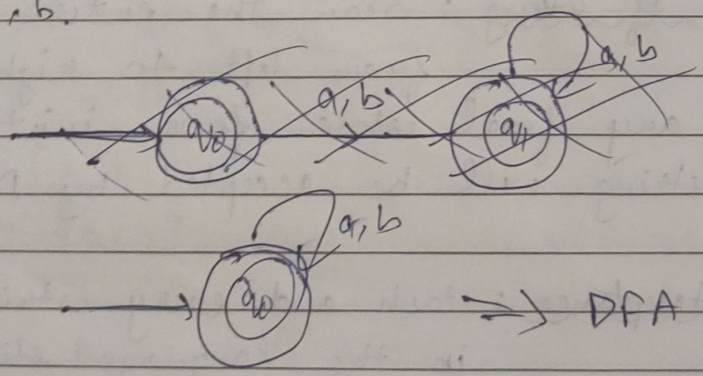
Q→ Construct a minimal DFA over $\Sigma = \{a, b\}$, where every string in the language should be of length at most 2.

→ $L = \{w \mid |w| \leq 2\}$



Q→ Construct a minimal DFA over $\Sigma = \{a, b\}$, that accepts all the strings of any length over a, b.

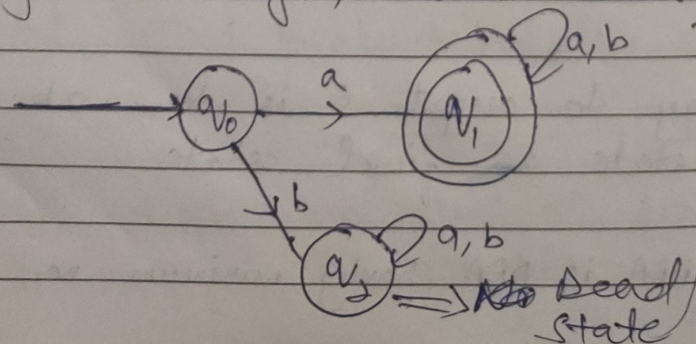
→



⇒ DFA of Σ^*.

Q→ Construct a minimal DFA over $\Sigma = \{a, b\}$, where every string in lang., should start with a.

→



18/01/24

δ : Transition Function

$$Q \times \Sigma \rightarrow Q$$

$$\{q_0, q_1, q_2\} \times \{a, b\} \longrightarrow \{q_0, q_1, q_2\}$$

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

$\Rightarrow \delta$ (TRANSIT^N FUNC^N)

Transition Table

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

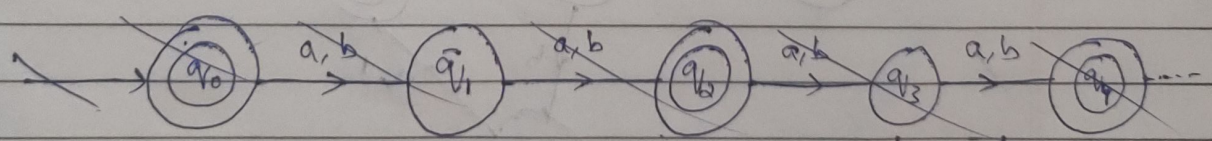
$\rightarrow \Rightarrow$ INITIAL STATE

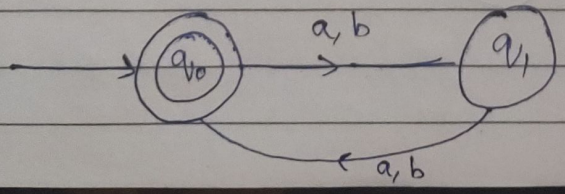
$*$ $\Rightarrow$ FINAL STATE

Q → Construct minimal DFA over $\Sigma = \{a, b\}$, where every string in the language is divisible by 2 or language containing even length string.

→ $L = \{w \mid |w| \text{ is even}\}$

$L = \{\epsilon, aa, ab, ba, bb, aaaa, aaab, \ldots\}$

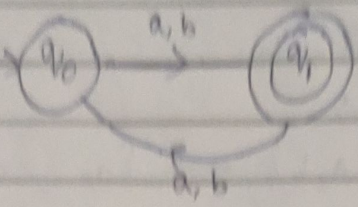




	a	b
$\rightarrow q_0^*$	q_1	q_1
q_1	q_0	q_0

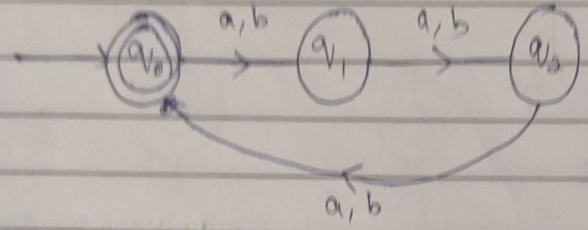
Q. Same as previous but for odd.

→



Q. Construct minimal DFA over $\Sigma = \{a, b\}$, where every string in the language is going to be divisible by 3.

→



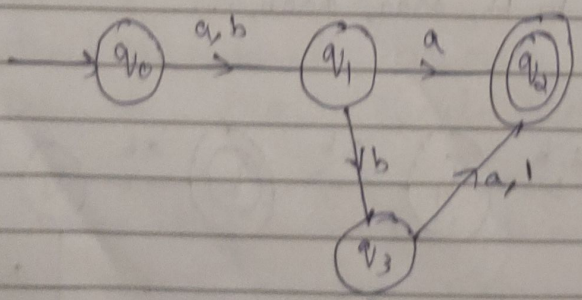
$L = \{w / |w| \bmod 3 = 0\}$

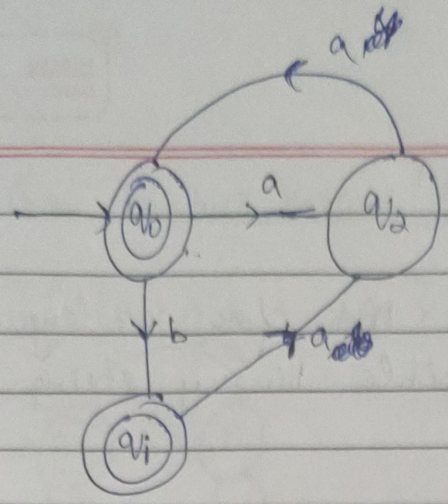
$\{w / |w| \cong 0 \bmod 3\}$

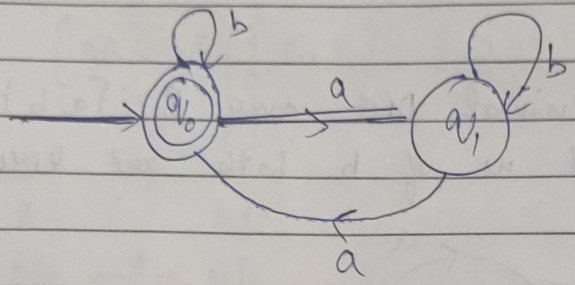
** A DFA will have n number of states if language have string of length divisible by n. (remainder wala node final node hoga).

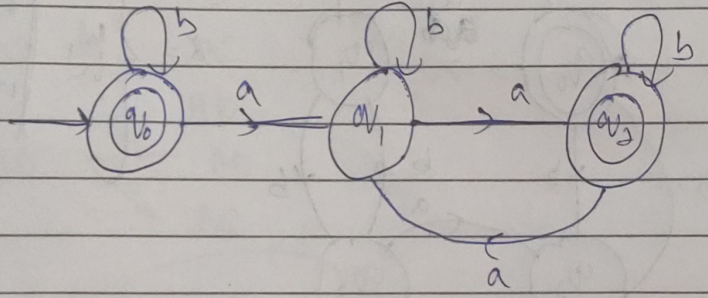
Q. Construct minimal DFA over $\Sigma = \{a, b\}$, if no. of a in each string is going to be divisible by 2.

→





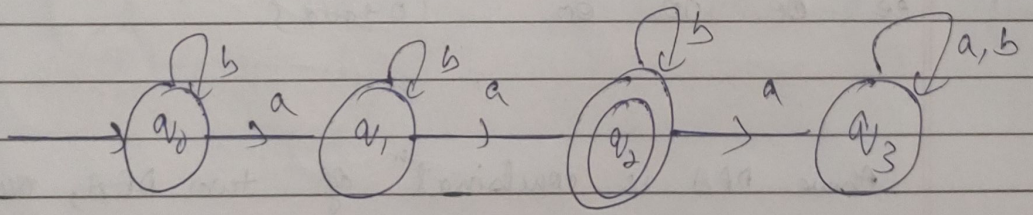




Also a DFA but not minimal.

Q → No. of a = 2, minimal DFA.

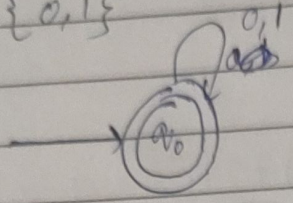
→



19/01/24

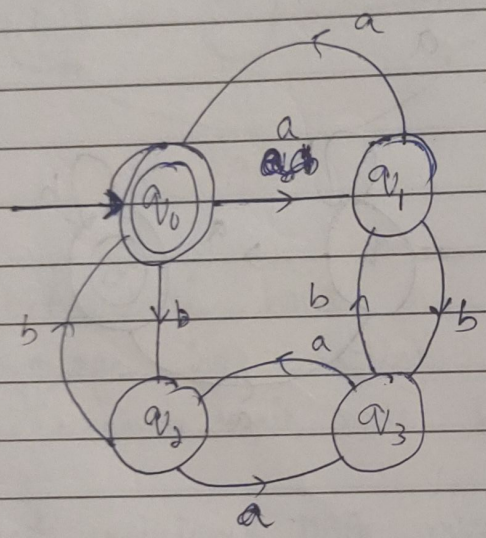
Q → Construct a minimal DFA for a language containing every possible binary string

→ $\Sigma = \{0, 1\}$



** Q → Construct a minimal DFA over $\Sigma = \{a, b\}$, where no. of a and no. of b both are even.

→

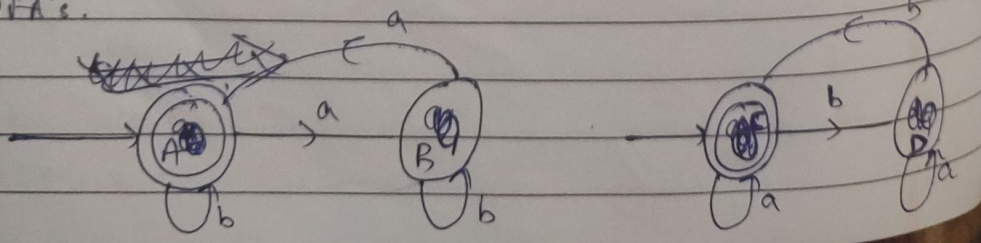


$L = \{ w \mid n_a(w) \stackrel{2}{=} 0 \text{ and } n_b(w) \stackrel{2}{=} 0 \}$

q_{00}	q_{01}	q_{10}	q_{11}	{e → even, o → odd}
ee	eo	oo	oe	

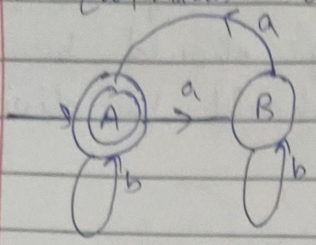
Above DFA is combinat^n of two DFA, one having even a's other having even b's.

⇒ We can make this DFA using 2 different DFA's.

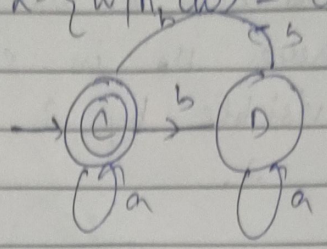


→ Cross Product Method (Compound DFA)

$L = \{w \mid n_a(w) \cong 0 \bmod 2\}$



$L = \{w \mid n_b(w) \cong 0 \bmod 2\}$



$\{A,B\} \times \{C,D\}$

AC —a→ BC, AC —b→ AD

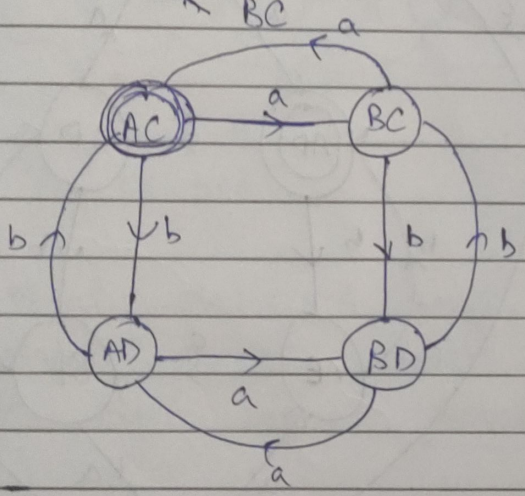
A —a→ B, A —b→ A

AD —a→ BD, AD —b→ AC

C —a→ C, C —b→ D

BC —a→ AC, BC —b→ BD

BD —a→ AD, BD —b→ BC

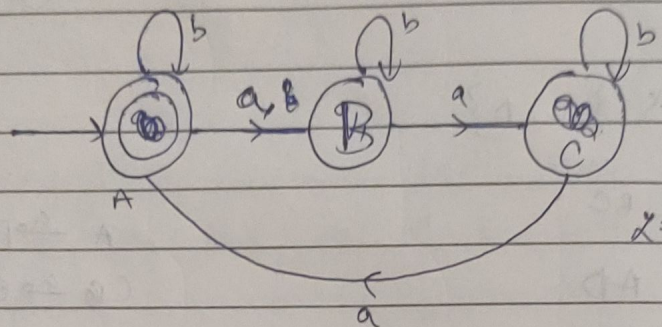


As A and C were final states, hence AC will be final

24/01/24

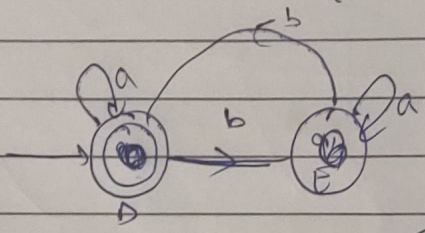
Q→ Construct a minimal DFA over alphabet {a,b} where n_A in every string should be divisible by 3 and n_B should be divisible by 2.

→



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

W | $n_A \stackrel{?}{=} 0 \pmod 3$; $n_B \stackrel{?}{=} 0 \pmod 2$



$\{A, B, C\} \times \{D, E\}$

AD —a→ BD, —b→ AE

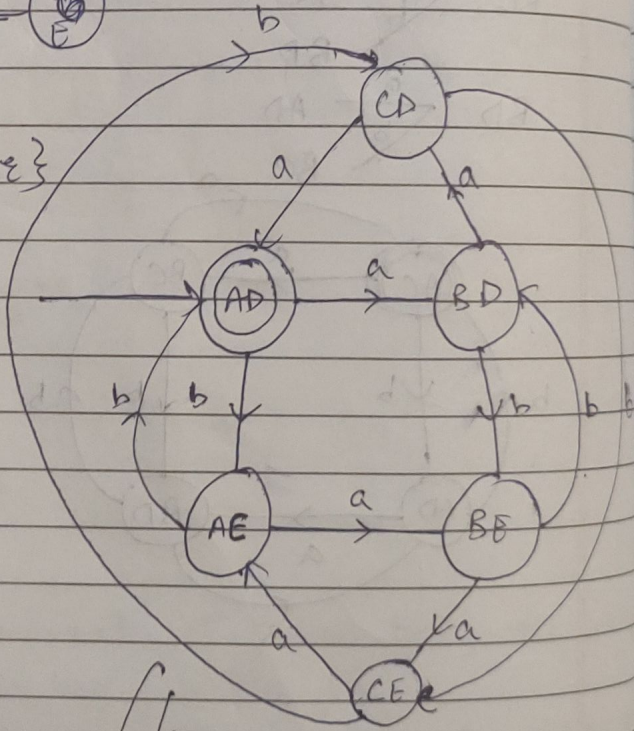
AE —a→ BE, —b→ AD

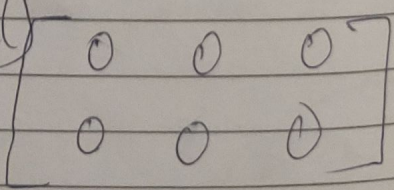
BD —a→ CD, —b→ BE

BE —a→ CE, —b→ BD

CD —a→ AD, —b→ CE

CE —a→ AE, —b→ CD





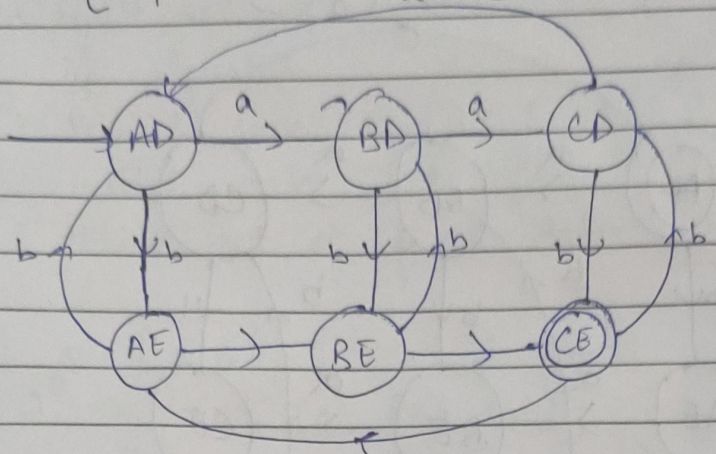
Aise banao toh acha lagega

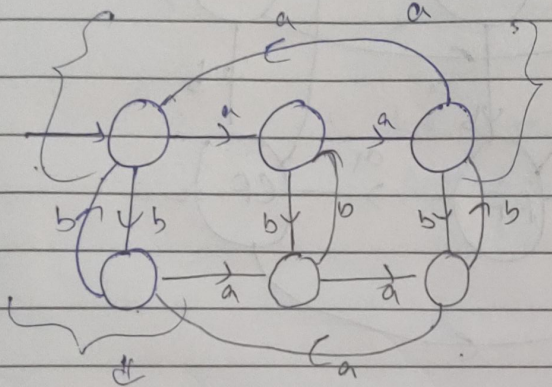
PAGE No
DATE : / 202

Q. Construct a minimal DFA over $\Sigma = \{a, b\}$, where

$$L = \left\{ w \;\middle|\; \begin{array}{l} n_A \overset{\text{a}}{\equiv} 2 \bmod 3 \\ \text{and} \\ n_B \overset{\text{a}}{\equiv} 1 \bmod 2 \end{array} \right\}$$

→





$\Rightarrow$ $n_A \bmod 3$ ka DFA

$n_B \bmod 2$ ka DFA

Total states 6 honge, toh bache hue 2 states purane states se parallel bana denge, or edges repeat karenge parallely

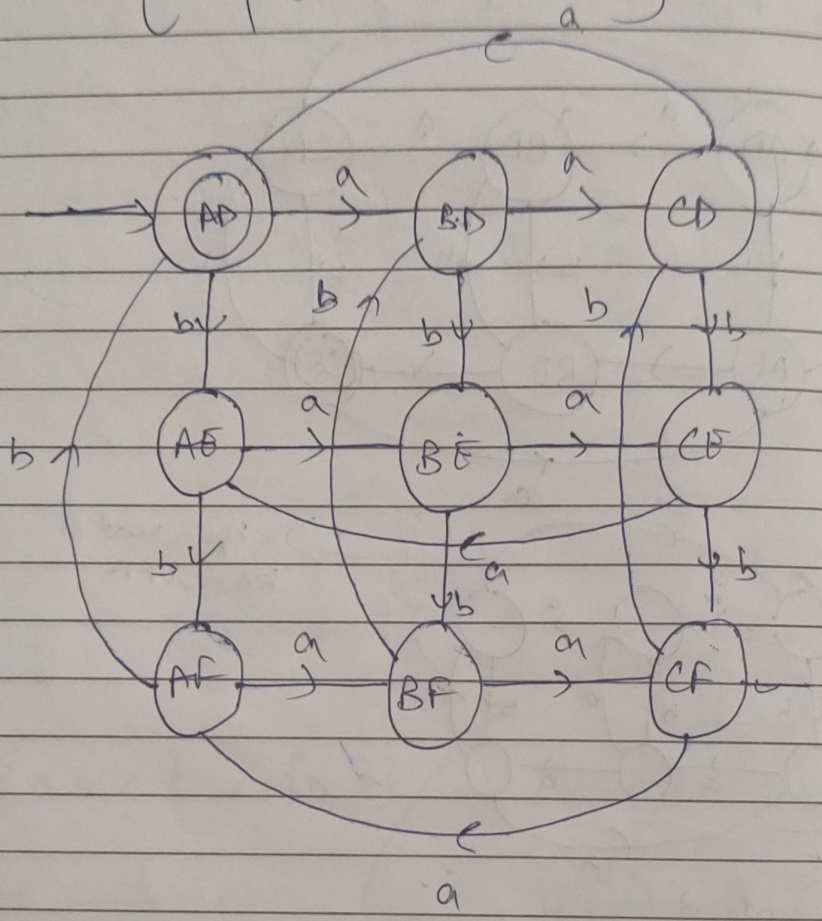
↓

done by black pen

Bhut Kaam Ki Short Trick

Q. Construct minimal DFA over $\Sigma = \{a, b\}$

$$L = \left\{ w \;\middle|\; \begin{array}{l} n_a \equiv 0 \bmod 3 \\ n_b \equiv 0 \bmod 3 \end{array} \right\}$$



PAGE No.
DATE: / / 202

25/01/24

• Non-deterministic finite Automata (NFA)

→ Defined by 5 tuple notation, i.e.,

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

Q : finite set of states

Σ : i/p alphabet

δ : transit" func" $\Rightarrow (Q \times \Sigma \rightarrow 2^Q)$ → power set

q_0 : initial state $(q_0 \in Q)$

F : finite set of final states $(F \subseteq Q)$

eg: $Q = \{A, B\}$

$2^Q \Rightarrow \{\{\epsilon\}, \{A\}, \{B\}, \{AB\}\}$

⇓

We can go anywhere (कहीं भी मतलब कहीं भी) causing (ambiguity)

⇓

confusion

→ Comparison b/w NFA and DFA

i) NFA & DFA accepts the type-3 languages (regular language)

ii) NFA & DFA are equivalent hence they have same power.

iii) Every DFA is a special case of NFA but every NFA need not be a DFA

iv) They differ in terms of transit" func".

$\delta_{DFA} : Q \times \Sigma \rightarrow Q$ whereas $\delta_{NFA} : Q \times \Sigma \rightarrow 2^Q$

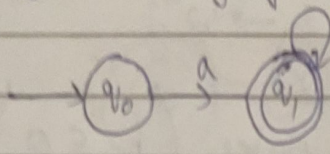
v) Construct" of NFA is easier than DFA for many problems.

vi) In NFA, zero or more no. of transit" from any

state for each input symbol should be there but in DFA there is exactly one transition from every state for each i/p symbol.

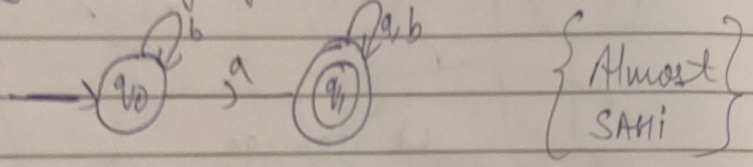
Q→ Construct a NFA over $\Sigma = \{a, b\}$ where each string in language should start with a.

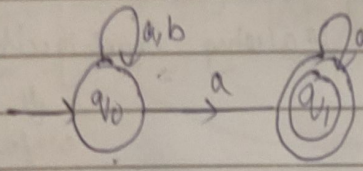
→



Q→ Construct a NFA over $\Sigma = \{a, b\}$ where each string in language contains a.

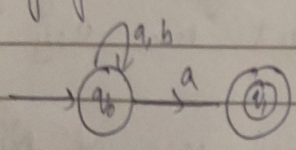
→





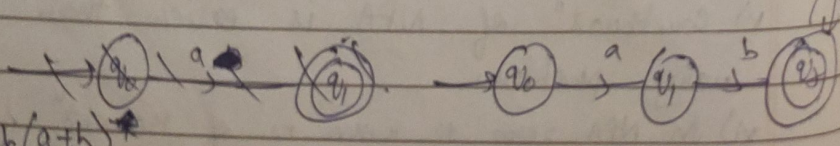
Q→ Construct NFA over $\Sigma = \{a, b\}$, where every string in language must end with a.

→



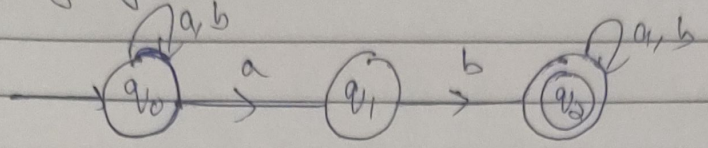
Q→ Construct NFA over $\Sigma = \{a, b\}$, where every string in language should start with ab.

→



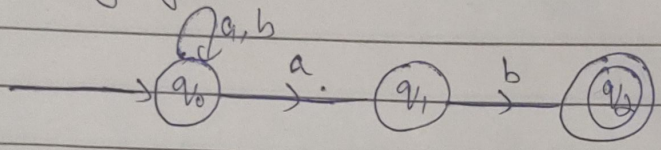
Q→ Construct a NFA over $\Sigma = \{a, b\}$, where every string in language should contain a, b.

→



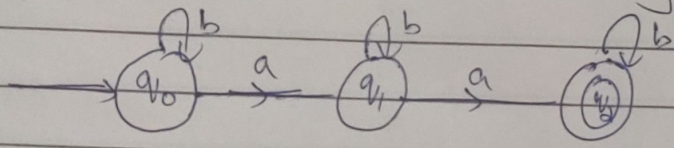
Q→ Construct NFA over $\Sigma = \{a, b\}$, where every string in language must end with a, b.

→



Q→ Construct NFA over $\Sigma = \{a, b\}$, where every string in language should contain exactly two a.

→



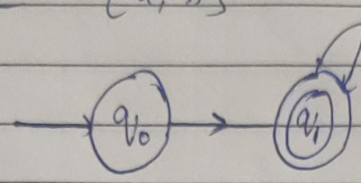
07/02/24

→ NFA to DFA Conversion

Let

L = {All strings starting with 'a'}

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



} NFA for above

transit^n table for this NFA :-

	a	b
→ q_0	q_1	
q_1*	q_1	q_1

} Not completely correct (Something Missing)

⇓

	a	b
→ q_0	$\{q_1\}$	ϕ or $\{\}$
q_1*	$\{q_1\}$	$\{q_1\}$

} As this is for NFA, here states are written in form of subset, not as single set, as NFA has mapping ($Q \times \Sigma \to 2^Q$)

NFA to DFA

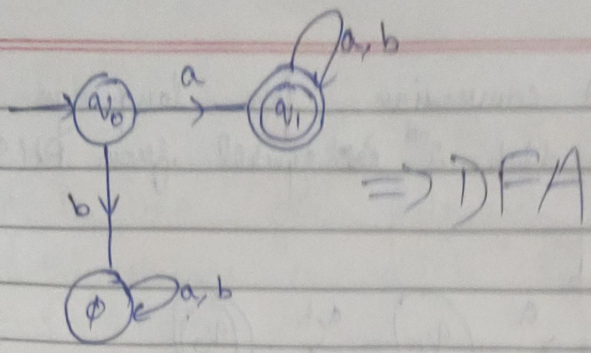
	a	b
→ q_0	$[q_1]$	ϕ
$[q_1]$	$[q_1]$	$[q_1]$
ϕ	ϕ	ϕ

} iss table mai hum left side mai wo saari states lenge jo-jo table mai aa rhi hai, ∴ ϕ ka transition liya hai

⇓

Transit^n table for the equivalent DFA. Now draw DFA for this table

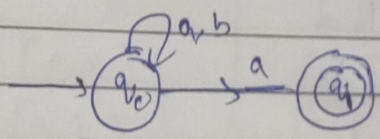
→ i.e., we will treat ϕ as a state



** [ ] ~~को~~ use kara, ye aage questions mai samajh aayega ~~कैसे~~ ki NFA mai subset use hote hai but DFA mai use subset ko ~~हम~~ As a single state treat karenge.

Q. ⇒ NFA to DFA for language
$\Sigma = \{a, b\}$
L = {All string ends with 'a'}

⇒



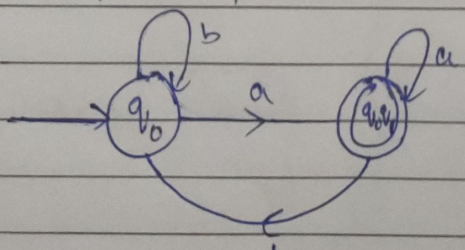
	a	b	a
$\rightarrow q_0$		$\{q_0\}$	$\{q_0, q_1\}$
q_1^*	ϕ	ϕ	ϕ

NFA to DFA

	a	b
$\rightarrow q_0$	$[q_0 q_1]$	$[q_0]$
$[q_0 q_1]$	$[q_0 q_1]$	$[q_0]$

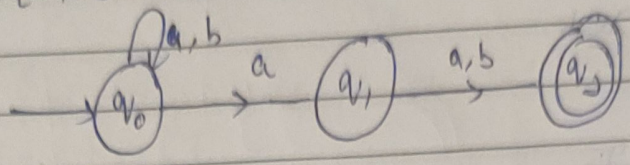
$q_0 q_1 \xrightarrow{a} (q_0, a) \cup (q_1, a)$
$\{q_0, q_1\} \cup \{\ \}$
$\{q_0\ q_1\}$

$q_0 q_1 \xrightarrow{b} (q_0, b) \cup (q_1, b)$
$\{q_0\}$



Q→ NFA to DFA conversion for language
$L = \{$All strings in which 2nd ~~last~~ symbol from RHS is 'a'$\}$
$\Sigma = \{a, b\}$

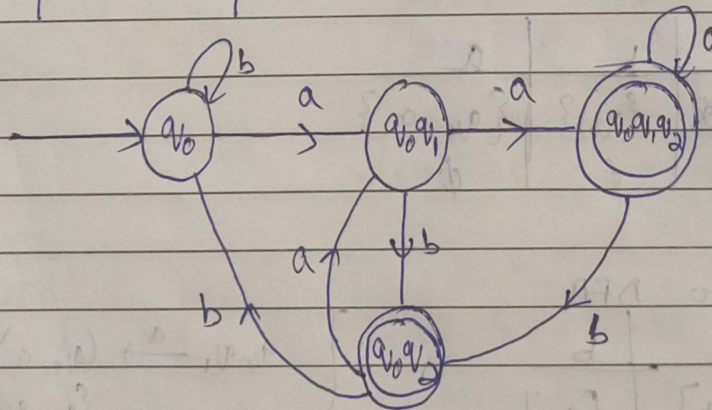
→



	a	b
→q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\{q_2\}$	$\{q_2\}$
*q_2	$\emptyset$	$\emptyset$

NFA to DFA

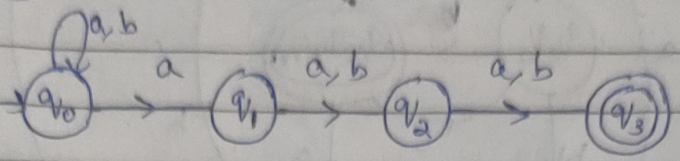
	a	b
→$[q_0]$	$[q_0 q_1]$	$[q_0]$
$[q_0 q_1]$	$[q_0 q_1 q_2]$	$[q_0 q_2]$
*$[q_0 q_2]$	$[q_0 q_1]$	$[q_0]$
*$[q_0 q_1 q_2]$	$[q_0 q_1 q_2]$	$[q_0 q_2]$



08/02/24

Q→ NFA to DFA for language
$L = \{$All strings in which 3rd symbol from RHS 'a'$\}$
$\Sigma = \{a, b\}$

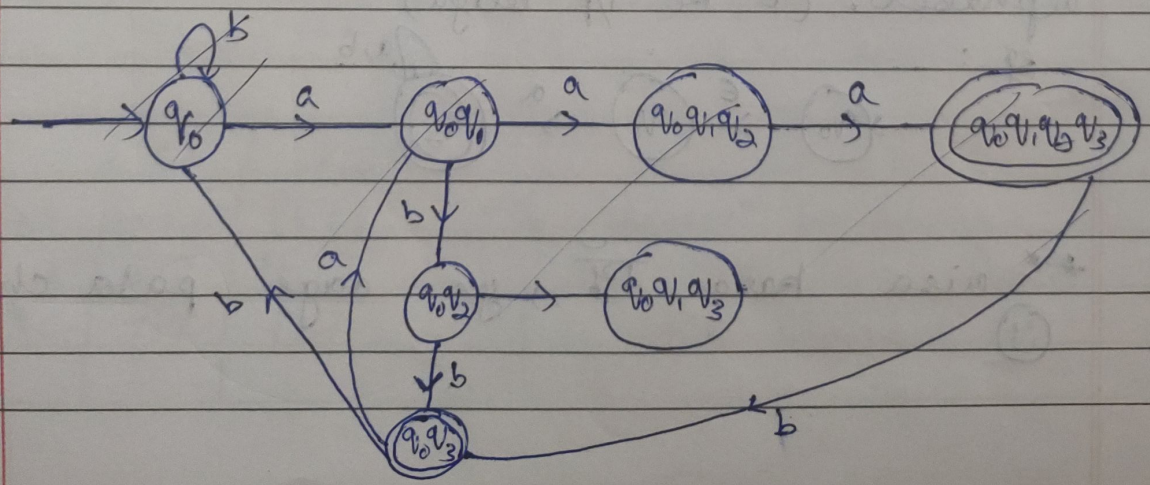
→

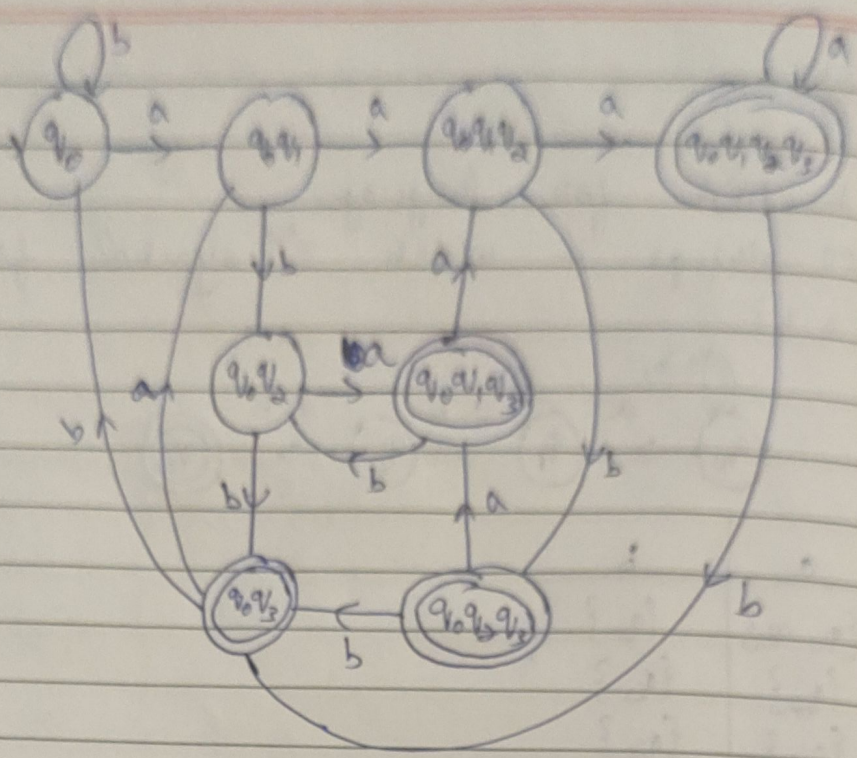


	a	b
→ q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\{q_2\}$	$\{q_2\}$
q_2	$\{q_3\}$	$\{q_3\}$
* q_3	ϕ	ϕ

NFA to DFA

	a	b
→ $[q_0]$	$[q_0 q_1]$	$[q_0]$
$[q_0 q_1]$	$[q_0 q_1 q_2]$	$[q_0 q_2]$
$[q_0 q_2]$	$[q_0 q_1 q_3]$	$[q_0 q_3]$
$[q_0 q_1 q_2]$	$[q_0 q_1 q_2 q_3]$	$[q_0 q_2 q_3]$
* $[q_0 q_3]$	$[q_0 q_1]$	$[q_0]$
* $[q_0 q_1 q_3]$	$[q_0 q_1 q_2]$	$[q_0 q_2]$
* $[q_0 q_1 q_2 q_3]$	$[q_0 q_1 q_2 q_3]$	$[q_0 q_2 q_3]$
* $[q_0 q_2 q_3]$	$[q_0 q_1 q_3]$	$[q_0 q_3]$





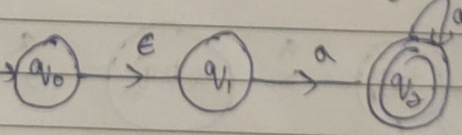
- ~~ϵ-NFA~~

→ Defined by 5 tuples, i.e.,
$Q, \Sigma, \delta, q_0, F$

Q: Finite set of states
Σ: i/p alphabet
δ: transit^n funct^n $\Rightarrow (Q \times \Sigma \cup \{\epsilon\} \rightarrow 2^Q)$
q_0: initial state
F: Finite set of final states

** basically, isme hum kisi ek state se dusre par ja sakte ho w/o taking any alphabet. (ϵ ko i/p leliya)

eg:

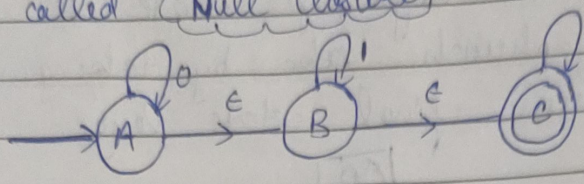


** aisa kara $\overline{\text{FJ}}$, ye aage pata chalega ☺

09/02/24

→ Epsilon Closure (ϵ^*) : All the states that can be reached from a particular state by only seeing ϵ-symbol.
Also called Null Closure

eg :



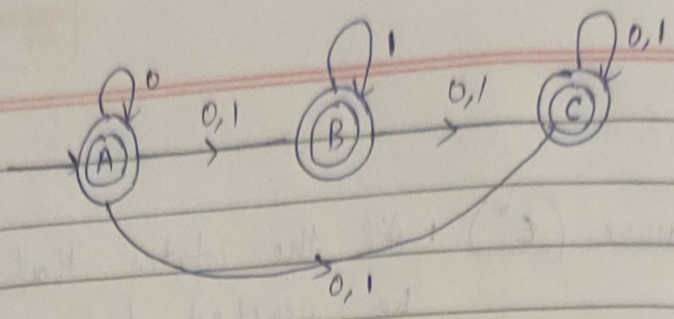
ϵ^* of A : {A, B, C}
ϵ^* of B : {B, C}
ϵ^* of C : {C}

Doing this for converting ϵ-NFA to NFA

State		input	ϵ^*	ϵ^*
A	A	0 → A	{A, B, C}	∪ ⇒ {A, B, C}
	B	0 → ϕ	{}	
	C	0 → C	{C}	
	A	1 → ϕ	{}	
	B	1 → B	{B, C}	⇒ {B, C}
	C	1 → C	{C}	
B	B	0 → ϕ	{}	⇒ {C}
	C	0 → C	{C}	
	B	1 → B	{B, C}	⇒ {B, C}
	C	1 → C	{C}	

	0	1
→A	{A, B, C}	{B, C}
B	{C}	{B, C}
*C	{C}	{C}

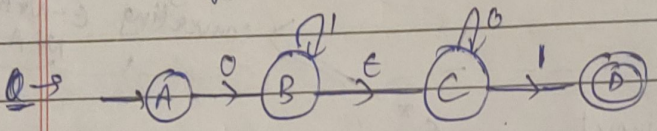
TRANSIT^N TABLE FOR NFA



** Final state wo hoga, jis state se hum kisi state tak pahunch sake by taking only 0 wo state hum final state lenge

OR

jiske ϵ^* mai final state aayegi, wo-wo states final states hongi.



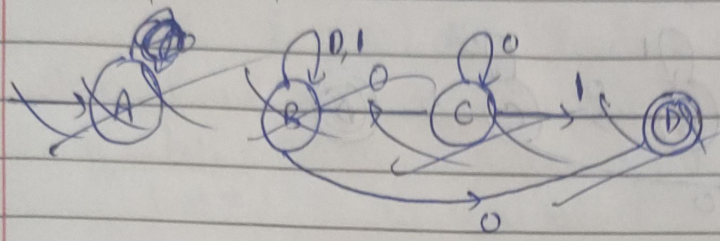
→ ϵ^* of A : {A}
ϵ^* of B : {B, C}
ϵ^* of C : {C}
ϵ^* of D : {D}

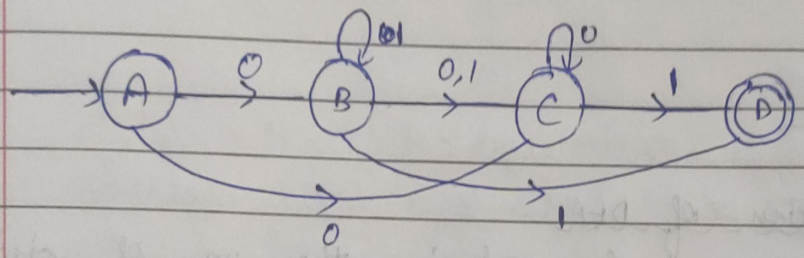
A → A 0 → B {B, C} ⇒ {A} {B, C}
A → A 1 → ϕ {}

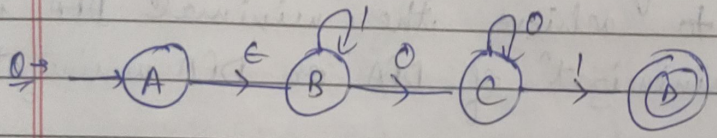
B → B 0 → ϕ {}
C 0 → C {C} ⇒ {C}
B 1 → B {B, C} ⇒ {B, C, D}
C 1 → D {D}

C → C 0 → C {C}
C 1 → D {D}
D → D 0 → ϕ {} ⇒ {}
D 1 → ϕ {}

	0	1
→A	{A, B, C}	ϕ
B	{C}	{B, C, D}
C	{C}	{D}
D	ϕ	ϕ







→ $\epsilon^*_A : \{A, B\}$

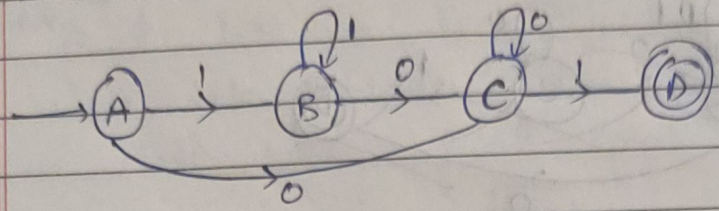
$\epsilon^*_B : \{B\}$

$\epsilon^*_C : \{C\}$

$\epsilon^*_D : \{D\}$

A
- A —0→ ϕ {} → {C}
- B —0→ C {C}
- A —1→ ϕ {} → {B}
- B —1→ B {B}
- B —0→ C {C}
- B —1→ B {B}
- C —0→ C {C}
- C —1→ D {D}
- D —0→ ϕ {}
- D —1→ ϕ {}

	0	1
A	ϕ {C}	ϕ {B}
B	{C}	{B}
C	{C}	{D}
D	ϕ	ϕ



14/02/24

- ~~Minimization of DFA~~
The process of reducing the no. of states of a DFA to achieve the minimal DFA is known as minimizat^n of DFA or DFA minimizat^n

→ There are 2 commonly used technique for MOD (Minimizat^n of DFA)
i) Equivalence Theorem
OR
Partitioning Method
ii) Myhill Nerode Theorem
OR
Table filling Method

→ Equivalence Theorem / Partitioning Method
(i) If DFA contains any dead state or inaccessible (unreachable) state, then remove & eliminate it.
** ~~Unreachable state~~: state that can't be reached from initial state.

(ii) Let us say, there are 2 states P and Q and a string w:

if,

$\delta(P, w) \in F \Rightarrow \delta(Q, w) \in F$

OR

$\delta(P, w) \notin F \Rightarrow \delta(Q, w) \notin F$

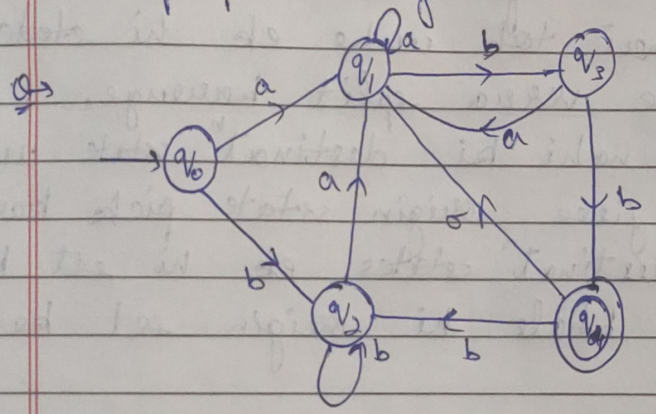
then, P and Q are equivalent.

** if $|w| = 0 \Rightarrow$ 0 - Equivalents

$|w| = 1 \Rightarrow$ 1 - Equivalents

$|w| = k \Rightarrow$ k - Equivalent

Whenever (k+1) equivalent = k equivalent, then stop partitioning.



→ Transit^n Table:

	a	b
→q_0	q_1	q_2
q_1	q_1	q_3
q_2	q_1	q_2
q_3	q_1	q_4
q_4*	q_1	q_2

0-Equivalent

$[q_0, q_1, q_2, q_3]$ $[q_4]$ } Non-final or final states ko set ko alag-alag likh do.

1- Equivalent
$[q_0, q_1, q_2]$ $[q_3]$ $[q_4]$

** Hamne kya kara hai ki, q_4 wala toh singleton set hai, usse as it is likh denge, bade set ko break karenge.
Bada set lenge, usko koi 2 state lenge, or table ki madad se equivalency dekhenge.
** Equivalency kaise dekhni hai?

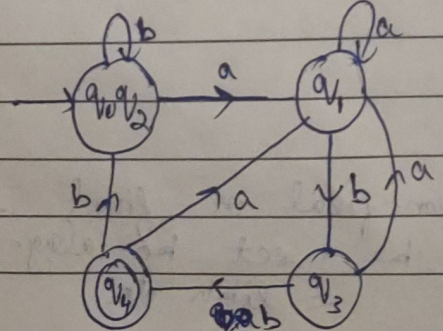
⇓

Ek set ke 2 state lenge, fir table se dekhenge. alphabet se transit kis state par ho raha hai.
Destinat^n states agar same state se aa rahi hai toh unhe ek hi state mein likhenge varna split karenge.
** Aisa zaruri nahi ki destinat^n state usi set ka ho jisse origin state pick kare hai, but destinat^n states ek hi set ke hone chahiye bhale hi origin set ke na ho.

2- Equivalent
$[q_0, q_2]$ $[q_1]$ $[q_3]$ $[q_4]$

3- Equivalent
$[q_0, q_2]$ $[q_1]$ $[q_3]$ $[q_4]$

Same aaya unko jayenge



PAGE No
DATE : / 202

21/02/24

δ	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

Koi bhi state q_3 par nahi jaa rahi
$\Rightarrow$ q_3 is ded, toh q_3 ko hatakar bhi question solve ho sakta hai.
Do As You Wish

→ 0-equivalent

$[q_0, q_1, q_3, q_4, q_5, q_6, q_7]$ $[q_2]$

1-equivalent

$[q_0, q_4, q_6]$ ~~$[q_1], [q_7], [q_2] [q_3] [q_5]$~~

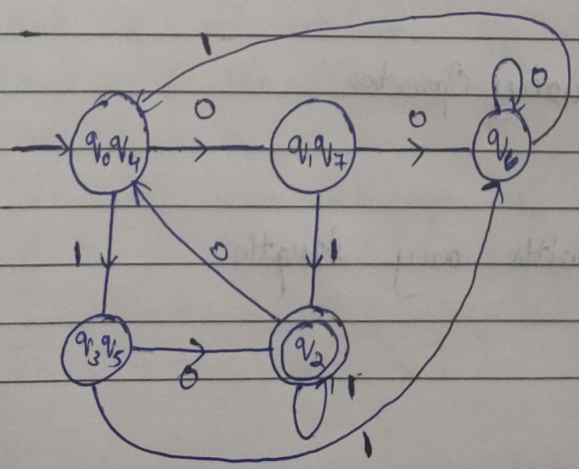
2-equivalent

~~$[q_0] [q_1] [q_3] [q_3] [q_4] [q_5] [q_6] [q_7]$~~

$[q_0, q_4, q_6], [q_1, q_7], [q_3, q_5], [q_2]$

2-equivalent

$[q_0, q_4], [q_6], [q_1, q_7], [q_3, q_5], [q_2]$



- Regular Expressions (R.E.)

It is the expression for the set of strings of any language which is accepted by any Finite Automata.

Σ^* => Clean Closure = $\Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots$

Σ^+ => +ve Closure = $\Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots$

OR

+ve Clean Closure

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

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

Eg :-

* ϕ => Empty Language

R.E. = ϕ

* $\Sigma = \{\epsilon\}$ => set of all strings with length 0

R.E. = ϵ

** Operations

R^*	+	=> Union
R^+	.	=> Concatenat^n
Unary Operator		Binary Operator

* Set of strings with any length

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

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

Q→ Write R.E. for language over Σ with strings of length exactly 2.

→ $L = \{aa, ab, ba, bb\}$

R.E. = $aa + ab + ba + bb$

$= a(a+b) + b(a+b)$

$= (a+b)(a+b)$

Q→ Set of all strings of length at least 2.

→ $L = \{aa, ab, ba, bb, aaa, aab, \ldots\}$

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

$(a+b)(a+b)$ ⟹ for exactly 2

$(a+b)^*$ ⟹ string of any length

Q→ Set of all strings of length at most 2

→ R.E. = $\epsilon + a + b + aa + ab + ba + bb$

$= (\epsilon + a + b)(a+b)(a+b)$

$\Rightarrow (a + b + \epsilon)(a + b + \epsilon)$

22/02/24

Q→ Set of all strings of length exactly 3.

→ R.E. = $(a+b)(a+b)(a+b)$

Q→ Set of all strings of length at least 3.

→ R.E. = $(a+b)(a+b)(a+b)(a+b)^*$

Q→ Set of all strings of length at most 3.

→ R.E. = $(a+b+\epsilon)(a+b+\epsilon)(a+b+\epsilon)$

Q→ Set of all strings of even length.
→ R.E. = $[(a+b)(a+b)]^*$

Q→ Set of all strings of odd length.
→ R.E. = $[(a+b)(a+b)]^*(a+b)$

Q→ Set of all strings of length divisible by 3 a multiple of 3.
→ R.E. = $[(a+b)(a+b)(a+b)]^*$

Q→ Set of all strings of length congruents to 2 mod 3.
→ R.E. = $[(a+b)(a+b)(a+b)]^*(a+b)(a+b)$

** x mod y ko liye banana hua toh bade bracket mai y times eh term, baki power *, then x times waki term uske aage

Q→ Set of all strings starting with a.
→ R.E. = $a(a+b)^*$

Q→ Set of all strings ending with b.
→ R.E. = $(a+b)^*b$.

Q→ Set of all strings where every string should start and end with different symbol.
→ R.E. = $a(a+b)^*b + b(a+b)^*a$

Q→ Set of all strings where every string start and ends with same symbol.
→ R.E. = $a(a+b)^*a + b(a+b)^*b + (\epsilon + a + b)$

Q→ Set of all strings where each string contains 'aa' as substring.

→ R.E. = $(a+b)^* aa (a+b)^*$

Q→ Set of all strings where every string contains exactly two a's.

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

Q→ Set of all strings where every string contains atleast two a's.

→ R.E. = $b^* a b^* a (a+b)^*$

OR

$(a+b)^* a (a+b)^* a (a+b)^*$ } Yeh bhi theek hai

Q→ Set of all strings where every string contains atmost two a's.

→ R.E. = $b^* a b^* a b^* + b^* a b^* + b^*$

$\Downarrow$

$b^* (a+\epsilon) b^* (a+\epsilon) b^*$

23/02/24

→ Myhill Nerode Theorem / Table Filling Method

4 Step theorem :-

(i) Draw a table for all the possible pairs of state P, Q.

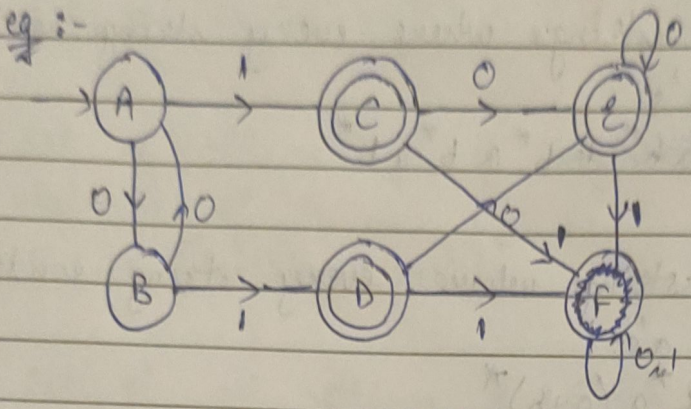
(ii) Mark all pairs where $P \in F$ and $Q \notin F$

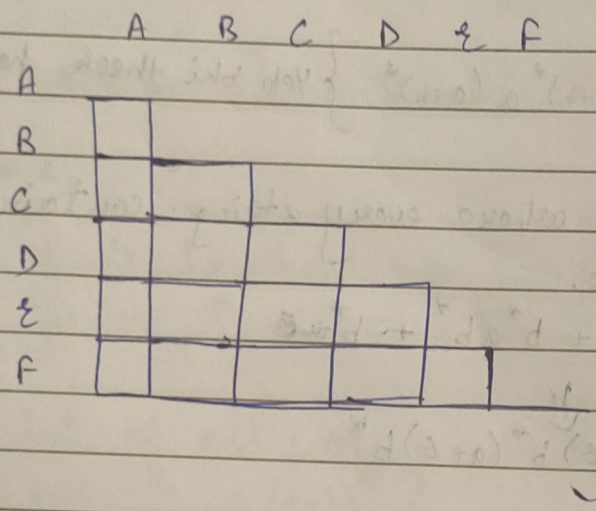
(iii) If there are any unmarked pairs (P, Q) such that $\delta(P, x)$ and $\delta(Q, x)$ are marked then mark (P, Q) where x is an i/p symbol.

Repeat step 3 until no more marking can be done.

(iv) Combine all the unmarked pairs and make them a single state in the minimized DFA.

eg :-

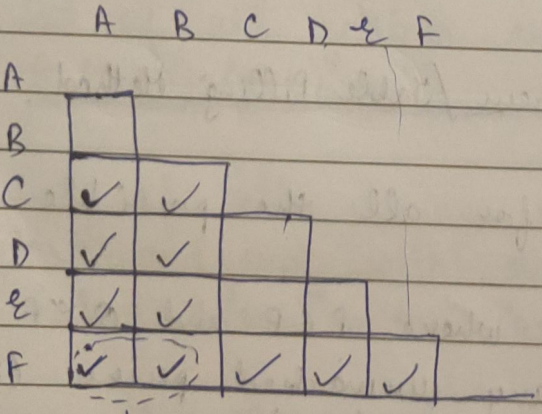




Step (i)
2 different states का pair लेना चाहिए or PQ pair लेलिया तो QP चाहिए लेना. ||

triangular matrix select करना है।

(ii) ||



||

Marked with step (iii)

(iii)

BA :	$\delta(B,0) \to A$	$\delta(B,1) \to D$
	$\delta(A,0) \to B$	$\delta(A,1) \to C$
	AB not marked	DC not marked

⇒ BA will not mark

** Agar kisi ek i/p par bhi marked nhi jaata toh mark kar dete.

DC :	$\delta(D,0) \to \xi$	$\delta(D,1) \to F$
	$\delta(C,0) \to \xi$	$\delta(C,1) \to F$

⇒ Not marked

ξC :	$\delta(\xi,0) \to \xi$	$\delta(\xi,1) \to F$
X	$\delta(C,1) \to \xi$	$\delta(C,1) \to F$

ξD :	$\delta(\xi,0) \to \xi$	$\delta(\xi,1) \to F$
X	$\delta(D,0) \to \xi$	$\delta(D,0) \to F$
FA :	$\delta(F,0) \to F$	$\delta(F,1) \to F$
	$\delta(A,0) \to B$	$\delta(A,1) \to C$
		FC is marked

⇒ FA will also be marked

FB :	$\delta(F,0) \to F$	$\delta(F,1)$
	$\delta(B,0) \to A$	δ

Marked

Ab hume isse fir se iterate karna hai, unmarked state se. Agar kuch bhi change nhi toh ek or iterat^n karna nahi hoga.

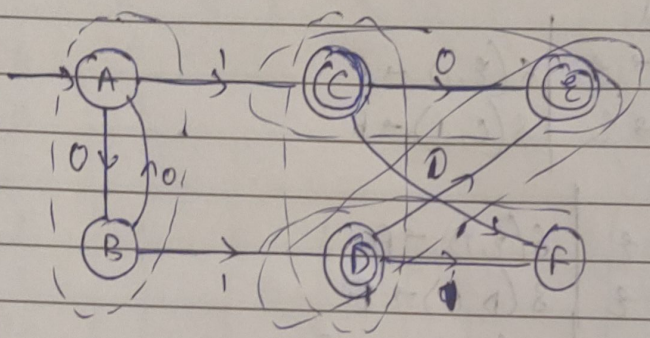
iterat^n 2:

BA:	$\delta(B,0) \to A$	$\delta(B,1) \to D$
	$\delta(A,0) \to B$	$\delta(A,1) \to DC$
DC:	$\delta(D,0) \to \varepsilon$	$\delta(D,1) \to F$
	$\delta(C,0) \to \varepsilon$	$\delta(C,1) \to F$

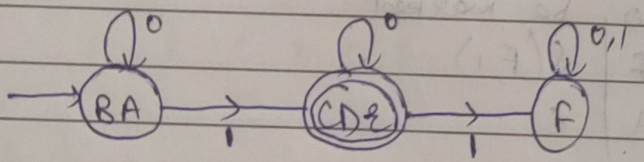
εC or εD will also not marked

** iterat^n 2 mai changes nahi aay isiliy (iii) complete.

(iv) ⇓



⇓



*** Paper mai iss topic par quest^n pkka aayega or ho sakta hai ki yahi quest^n aajay ya jo quest^n partioning se hard, they wo iss method se aajayenge.

⇓

PRACTICE KARO!!