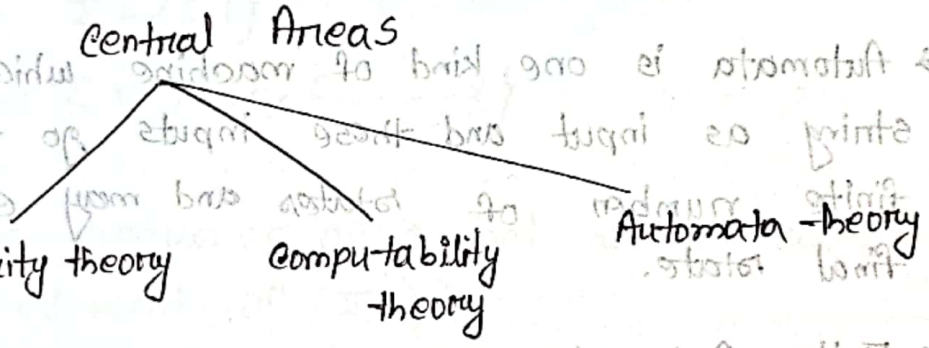
Introduction to TOC

Main objective of TOC: Develop formal mathematical models of computation reflect real world computers.

Our main purpose is to determine:

- What can and can't be computed?
- How quickly?
- With how much memory?
- On which type of computational model?



☐ Fundamentals capabilities and limitations of computers: Automata, computability and complexity

Complexity theory:

- What makes some problems computability hard and other problems easy?

⇒ Informally, a problem is called 'easy' if it is efficiently solvable. Ex: A sorting sequece of say 100000 number

⇒ On the other hand, a problem is called hard, if it can't be solved efficiently or if we don't know whether it can be solved efficiently.

Ex: factoring a 300 digit integer into its prime factors.

Computability theory:

- What problem can be solved by computers and which can not?

Ex: Determining a mathematical statements true or false.

Automata theory: Automata theory deals with the definitions and properties of different types of "mathematical models of computation.

⇒ Automata is one kind of machine which takes some string as input and these inputs go through a finite number of states and may enter in a final state.

⇒ Finite Automata: These are used in text processing, compiles and hardware design.

⇒ Context Free grammars: These are used to define programming languages and artificial intelligence.

⇒ Turning Machines: These from a simple abstract model of a "real" computer, such as your pc at home.

Symbol: Symbol is the basic building block of TOC

Ex: a, b, ---. z

0, 1, --- 9

Letters and digits

Alphabet: Finite set of symbol, nonempty set of symbol.

denoted by 'Σ'(sigma)

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

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

String: Finite sequence of symbol where each symbol is an element of Σ.

Denoted by "w"

Ex: $w = 0110$

$\Rightarrow \Sigma = \{a, b\} \Rightarrow w = abaa$

Length of string: If string is abaa then length $|w| = 4$

aba then length $|w| = 2$

Empty string: Denoted by "ε (epsilon)"

Ⓔ $\Sigma = \{0, 1\}$

Make strings of length 0, 1, 2, 3 over Σ

$\varepsilon \rightarrow$ 0 length

0/1 $\rightarrow$ 1 length

01, 10, 00, 11 $\rightarrow$ 2 length

000, 111, 010 $\rightarrow$ 3 length.

Language: Language is a collection of substring which can be finite or infinite.

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

L_1 = set of all strings over Σ of length 2

$= \{aa, ab, ba, bb\}$ (Finite language)

L_2 = set of all strings over Σ where it starts with 'a'.

$= \{a, aa, ab, aab, \ldots\}$ (Infinite language)

Powers of sigma (Σ):

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cdots$$

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

Σ^1 = set of all strings over Σ of length 1

$= \{a, b\}$

Σ^2 = set of all strings over Σ of length 2

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

Σ^3 = set of all strings over Σ of length 3

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

Σ^0 = set of all strings over Σ of length 0

$= \{\varepsilon\}$ (epsilon)

$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots$

$= \{\varepsilon\} \cup \{a, b\} \cup \{aa, ab, ba, bb\} \ldots$

$L_1 \subseteq \Sigma^*$
$L_2 \subseteq \Sigma^*$
$L_3 \subseteq \Sigma^*$

L_1, L_2, L_3 $\rightarrow \Sigma^*$

all languages are subset of Σ^*

- **Definition:** Describes the objects and notions that we use.
- **Mathematical statement:** Expresses that some object has a certain property.
- **Proof:** Convincing logical argument that a statement is true.
- **Theorem:** Mathematical statement proved true.
- **Lemma:** Proved mathematical statements that assist in the proof of another more significant statement.
- **Corollaries:** Easily concluded from a theorem on its proof.

Types of proofs:

Proof by construction: Many theorem state that a particular type of object exists. One way to prove such a theorem is by demonstrating how to construct the object. This technique is a proof by construction.

Proof by contradiction: First we will assume that the theorem is false and then show that this assumption leads to an obviously false consequence, called a contradiction.

Proof by induction:

- The basis proves that P(1) is true.
- The induction step proves that for each $i >= 1$, if P(i) is true then so is P(i+1).

Finite Automation: A finite Automation is a simple idealised machine used to recognize patterns within input taken from some character set (or alphabet) Σ.

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

$L(M)$ = set of all strings of length 2

$= \{aa, ab, ba, bb\}$ → Finite set / language

↓

Machine আছে যা এই language accept করবে.

If $\{abab\}$ or $\{aaaa\}$ → Machine accept করবে না কারণ string finite.

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

$L(M)$ = set of all strings starts with 'a'

$= \{a, ab, aa, aaa, \ldots\}$ → Infinite set / language

↓

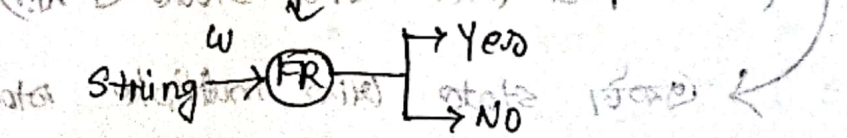
Machine accept করবে

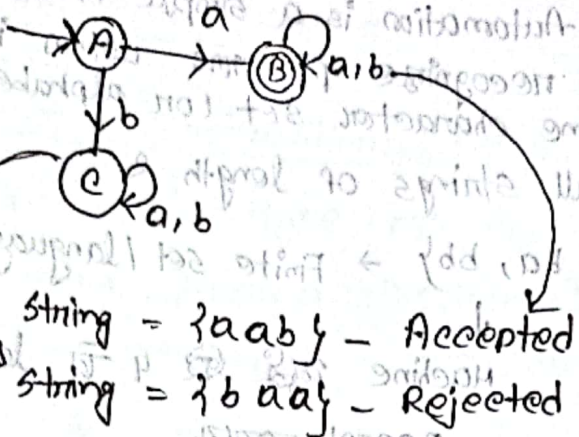
If $\{baaa\}$ → machine doesn't accept this string because it start with 'b'

যদি infinite set হয় তাহলে বুঝাতে পারি না আমাদের machine আমাদের দেয়া string accept করেছে কি করে নাই

তাই যদি আমরা আমাদের infinite language কে finite representation দিতে পারি তাহলে string দিয়ে check করতে পারি.

Infinite language





A ⇒ Starting state

B ⇒ Final state

String = {aab} – Accepted

String = {baa} – Rejected

Finite Automation.

Introduction to DFA

Finite Automata (FA)

- FA with output
  - Moore machine
  - Mealy machine
- FA without output
  - DFA
  - NFA
  - ε-NFA

DFA – Deterministic finite automata

NFA – Non deterministic finite Automata

(প্রত্যেক input এর জন্য প্রত্যেক state এ যায়)

প্রত্যেক state এর জন্য multiple state এ যাওয়া যায়,

DFA (Deterministic Finite Automata)

A finite automation that contains 5 tuples

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

where, (i) Q is a finite set of called 'states'

(ii) Σ is a finite set called 'alphabets'.

(iii) $\delta : Q \times \Sigma \to Q$ is a transition function

↓

Transition function

(iv) $q_0 \in Q$ is the start state

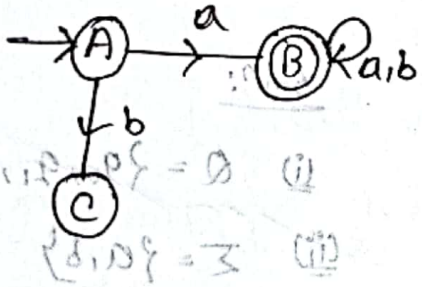
(v) $F \subseteq Q$ is the set of accept states.

↓

Final state

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

L(M) = set of all string that start with an 'a'



(i) Q = set of all states

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

(ii) Σ = input alphabet

$= \{a, b\}$

(iii) $\delta = Q \times \Sigma \to Q$

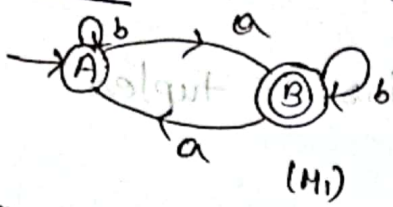
↓	a	b
→A	B	C
*B	B	B
C	C	C

(iv) q_0 = Start state = A

(v) F = set of final states

$= \{B\}$

Example-1



(M_1)

Soln:

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

(i) $Q = \{A, B\}$

(ii) $\Sigma = \{a, b\}$

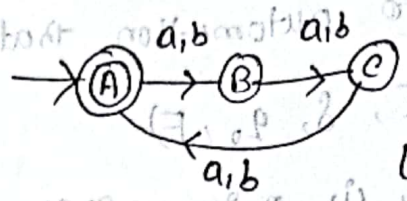
(iii)

States	a	b
→A	B	A
*B	A	B

(iv) $q_0 \in Q = A$

(v) $F \subseteq Q = \{B\}$

Example-2



(M_2)

Soln:

(i) $Q = \{A, B, C\}$

(ii) $\Sigma = \{a, b\}$

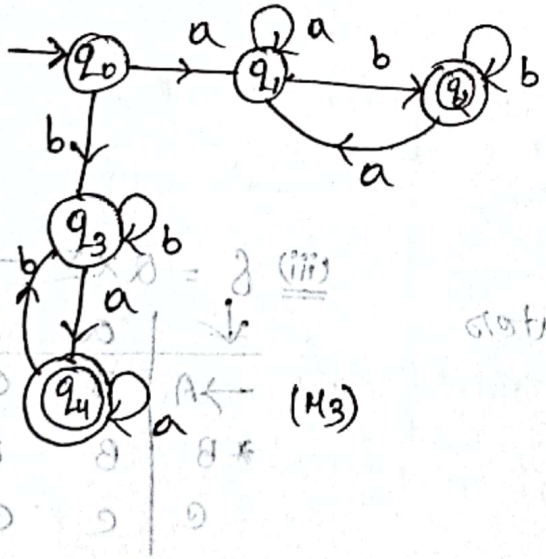
(iii)

States	a	b
→*A	B	B
B	C	C
C	A	A

(iv) $q_0 \in Q = A$

(v) $F \subseteq Q = \{A\}$

Example-3



(M_3)

Soln:

(i) $Q = \{q_0, q_1, q_2, q_3, q_4\}$

(ii) $\Sigma = \{a, b\}$

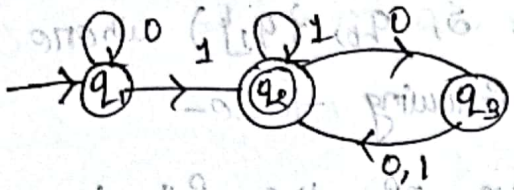
(iii)

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

(iv) $q_0 \in Q = q_0$

(v) $F \subseteq Q = \{q_2, q_4\}$

Example-4



(H4)

Soln:

(i) $Q = \{q_1, q_2, q_3\}$

(ii) $\Sigma = \{0, 1\}$

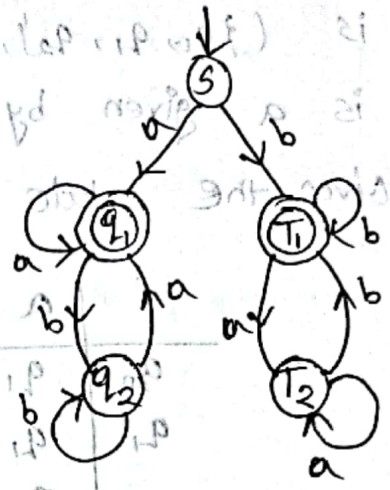
(iii)

States	0	1
$\rightarrow q_1$	q_1	q_2
$*q_2$	q_3	q_2
q_3	q_2	q_2

(iv) $q_0 \in Q = q_1$

(v) $F \subseteq Q = \{q_2\}$

Example-5



(H5)

(i) $Q = \{S, q_1, q_2, T_1, T_2\}$

(ii) $\Sigma = \{a, b\}$

(iii)

States	a	b
$\rightarrow S$	q_1	T_1
$*q_1$	q_1	q_2
q_2	q_1	q_2
$*T_1$	T_2	T_1
T_2	T_2	T_1

(iv) $q_0 \in Q = S$

(v) $F \subseteq Q = \{q_1, T_1\}$

The formal description of a DFA machine (M) is $(\{q_0, q_1, q_2\}, \{a,b\}, \delta, q_0, \{q_2\})$ where δ is a given by the following table-

Given the state diagram of the following:

	a	b
q_0	q_1	q_2
q_1	q_1	q_1
q_2	q_2	q_2

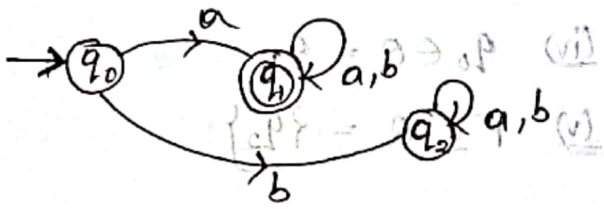
Soln:

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

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

$q_0 = q_0$

$F = q_1$



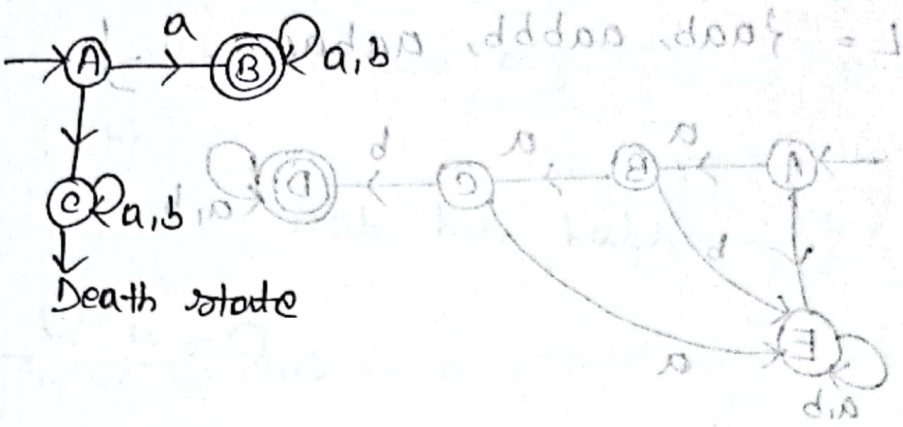
DFA Construction

Pb-1 Construct a DFA which accepts set of all strings over the $\Sigma = \{a,b\}$ where each string start with an 'a'.

$L(M) = \{w \mid w$ starts with an 'a'$\}$

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

$L = \{a, aa, ab, abba, abbaaa, \ldots\}$

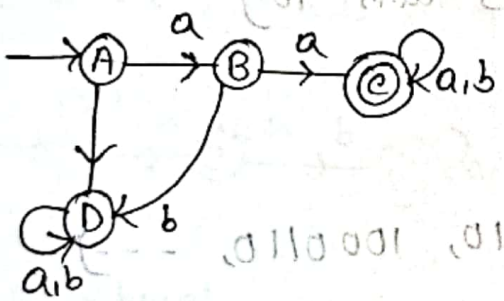


Pb-2

$L(M) = \{w | w$ starts with an 'aa'$\}$

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

$L = \{aa, aab, aaa, aabbaaa, \ldots\}$

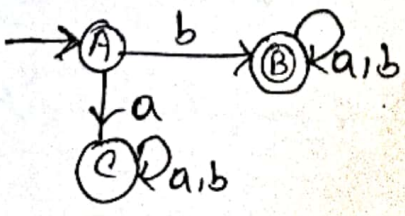


Pb-3

$L(M) = \{w | w$ starts with b$\}$

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

$L = \{b, ba, baa, \ldots\}$

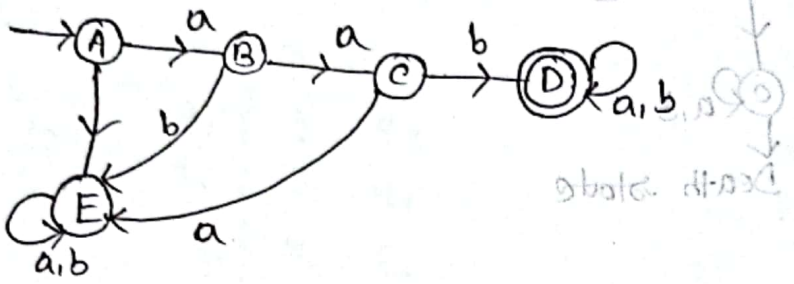


Pb-4

$L(M) = \{w \mid w \text{ starts with } aab\}$

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

$L = \{aab, aabbb, aabaa, \ldots\}$



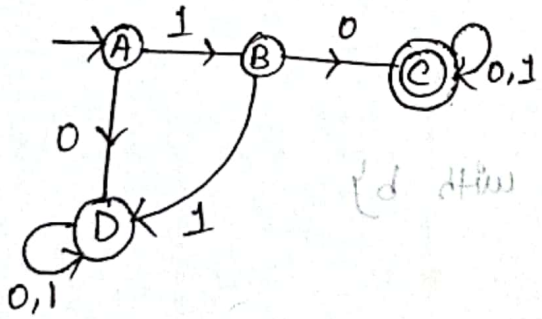
Pb-5

Construct DFA $\Sigma = \{0, 1\}$ which starts with '10'

$L(M) = \{w \mid w \text{ starts with } '10'\}$

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

$L = \{10, 1011, 1010, 1000110, \ldots\}$

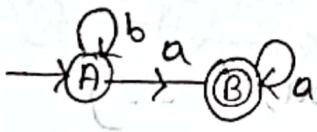


Pb-6 Construct a DFA which accepts set of all strings over the $\Sigma = \{a, b\}$ in where each string contains 'a'.

$L(M) = \{ w \mid w \text{ contains 'a'} \}$

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

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

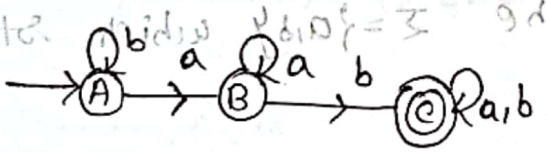


Pb-7

$L(M) = \{ w \mid w \text{ contains 'ab'} \}$

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

$L = \{ab, abaa, aabaa, \ldots\}$

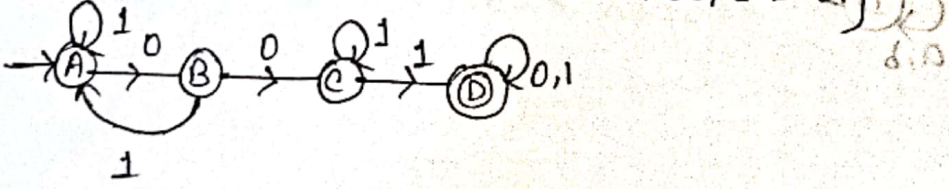


Pb-8 Construct a DFA which accepts set of all strings the $\Sigma = \{0, 1\}$ in where each string contains '001'.

$L(M) = \{ w \mid w \text{ contains '001'} \}$

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

$L = \{001, 00100, 01001, 00100 1 \ldots\}$



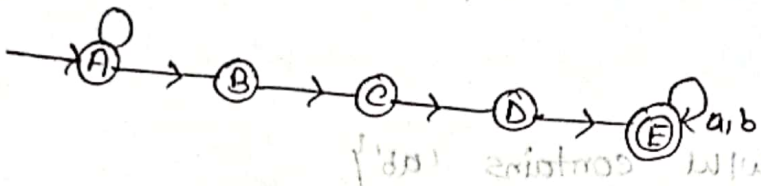
Pb-9

Construct a DFA which accepts set of all strings over the $\Sigma = \{a, b\}$ which contains the substring 'baba'.

$L(M) = \{ w \mid w$ contains substring 'baba' $\}$

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

$L = \{baba, ababa, aabbaba, \ldots\}$



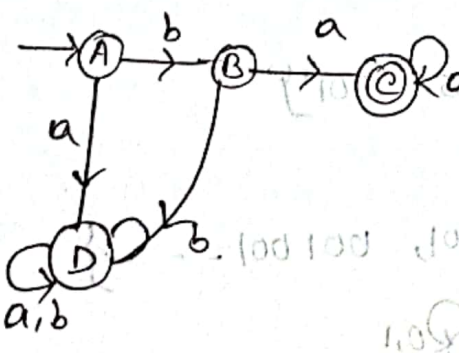
Pb-10

Construct a DFA which ~~contains~~ accepts of all strings over the $\Sigma = \{a, b\}$ which starts with 'ba'

$L(M) = \{ w \mid w$ starts with ba$\}$

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

$L = \{ba, baaab, babb, baba, \ldots\}$

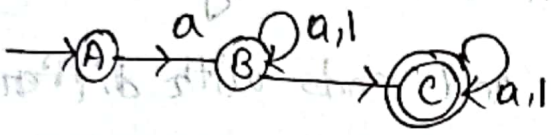


Pb-11 Construct a DFA which accepts set of all strings over the $\Sigma = \{0,1\}$ which contains '01'

Soln: $L(M) = \{w \mid w \text{ contains } 01\}$

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

$L = \{01, 001, 0001, 1101, \ldots\}$

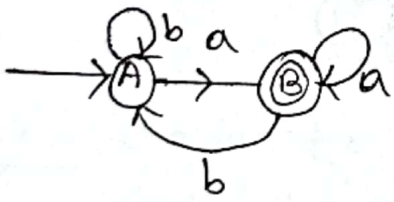


Pb-12 Construct a DFA which accepts set of all strings over $\{a,b\}$ where each string ends with 'a'.

Soln: $L(M) = \{w \mid w \text{ ends with } a\}$

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

$L = \{a, aba, ba, bba, baba, baaa\}$

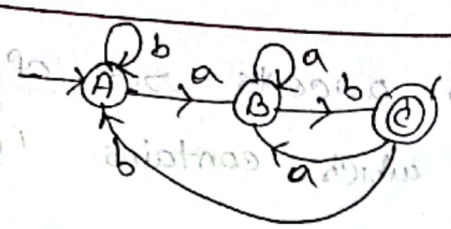


Pb-13 Construct a DFA which accepts set of all strings over $\{a,b\}$ where each strings ends with 'ab'.

Soln: $L(M) = \{w \mid w \text{ ends with 'ab'}\}$

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

$L = \{ab, bab, abab, babbab, bbab, \ldots\}$



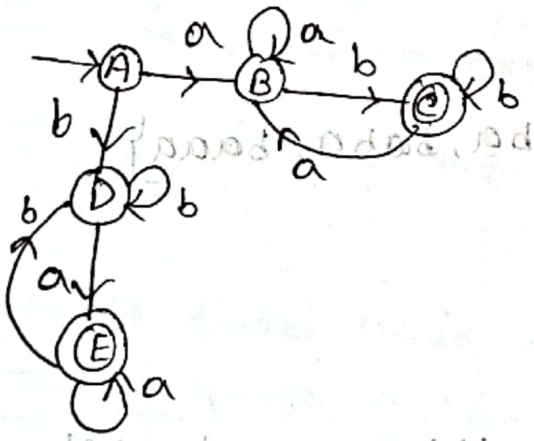
Pb: 14 Construct a DFA which accepts set of all strings over {a,b} where each string starts and ends with different symbol.

$L(M) = \{w | w$ starts and ends with different symbols$\}$

Soln:

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

$L = \{ab, ba, abab, bababa, bba, babb, \ldots\}$

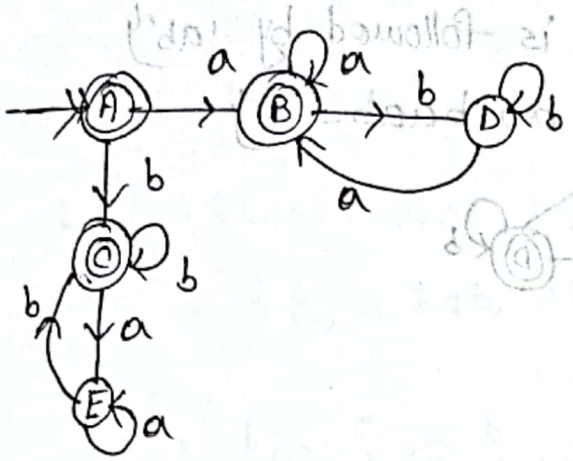


Pb: 15 Construct a DFA which accepts set of all strings over {a,b} where each string starts and ends with same symbols.

$L(M) = \{w | w$ starts and ends with same symbols$\}$

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

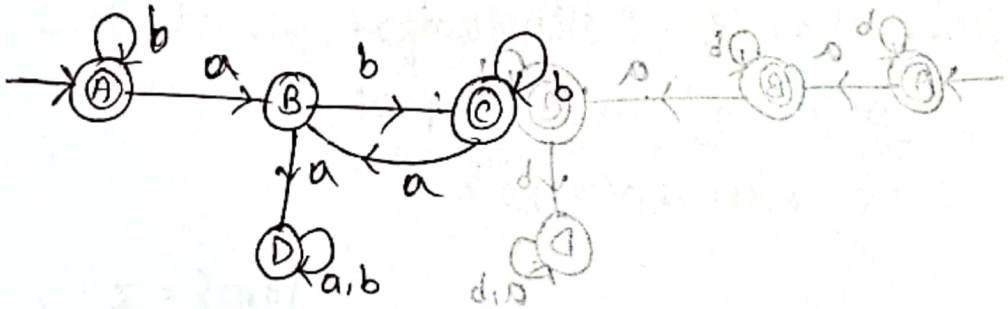
$L = \{ \varepsilon, a, b, aba, aa, bb, aaabbaa, bbbaabb, \ldots \}$



Pb:16 Construct a DFA, $\Sigma = \{a, b\}$

$L(M) = \{ w \mid w$ every 'a' in w is followed by one 'b' $\}$

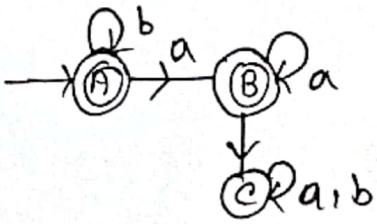
$= \{ \varepsilon, b, ab, abab, abbb, abab, bbb, \ldots \}$



Pb:17 – Construct a DFA which accepts set of all strings over $\{a, b\}$ whose every 'a' in string never followed by 'b'.

$L(M) = \{ w \mid w$ every 'a' in w is never followed by one 'b' $\}$

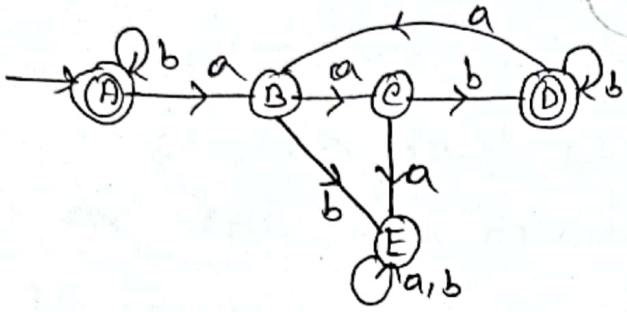
$= \{ \varepsilon, a, b, ba, aa, bbaa, \ldots \}$



Pb: 18 Construct a DFA $\Sigma = \{a, b\}$

$L(M) = \{w \mid w$ every a is followed by 'ab'$\}$

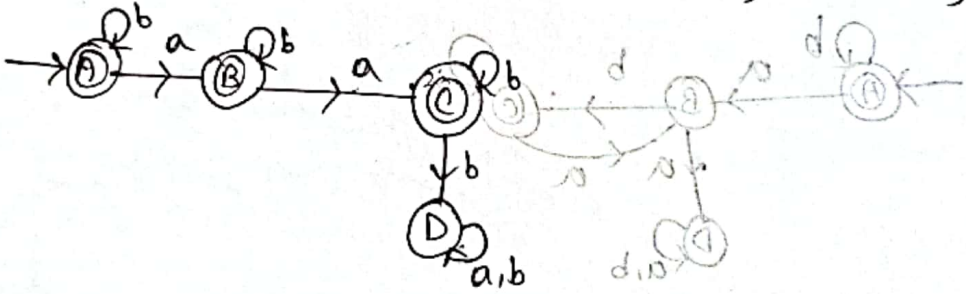
$= \{\varepsilon, b, aa, b, aabbaab - \}$



Pb: 19 Construct a DFA, $\Sigma = \{a, b\}$

$L(M) = \{w \mid w$ every a is never followed by 'ab'$\}$

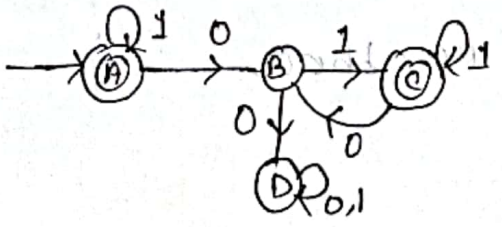
$= \{\varepsilon, b, aa, bb, aaa, baba, - \}$



Pb: 20 Construct a DFA $\Sigma = \{0, 1\}$

$L = \{w \mid w$ every '0' in w is followed by at least one '1'$\}$

$= \{\varepsilon, 1, 01, 11011101, 1101, - \}$

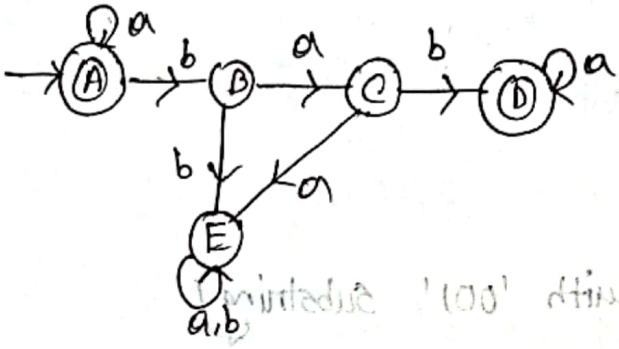


Pb:21 Construct DFA. $\Sigma = \{a, b\}$

Set of all strings where 'b' in the string is followed by 'ab'

$L(M) = \{w | w$ every b in w is followed by 'ab'$\}$

$= \{\varepsilon, a, bab, abab, --\}$

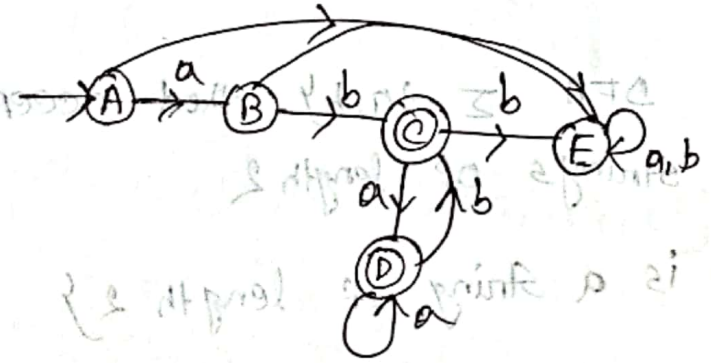


Pb:22 $L(M) = \{w | w$ begins with 'ab' and every other 'b' is preceeded by an 'a'$\}$

b এর আগে a থাকবে

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

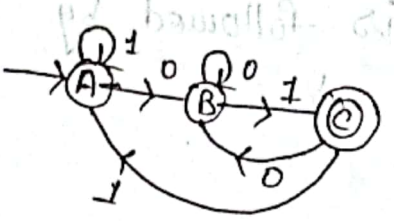
$L = \{ab, abaaa, abaab, --\}$



Pb:23 $L(M) = \{w | w \text{ ends with '01'}\}$

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

$L = \{01, 0001, 11011101, \ldots\}$

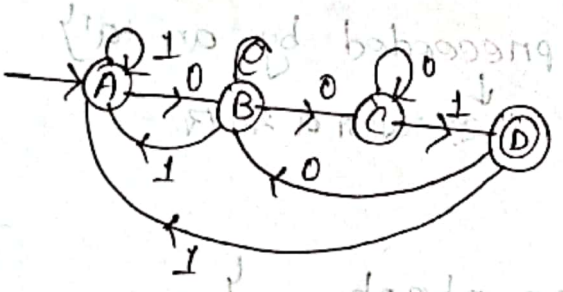


Pb:24

$L(M) = \{w | w \text{ ends with '001' substring}\}$

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

$L = \{001, 11001, 01001, 0001, \ldots\}$

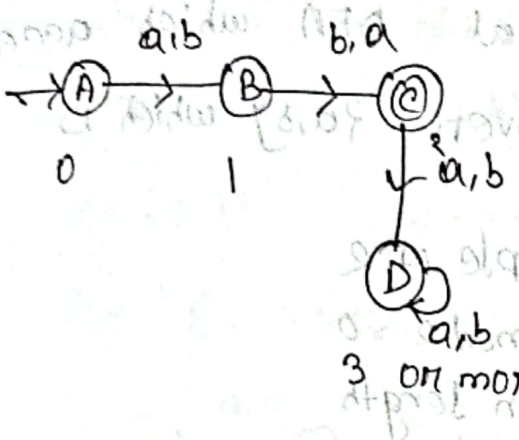


Pb:25 construct a DFA $\Sigma = \{a, b\}$ that accepts set of all strings of length 2

$L(M) = \{w | w \text{ is a string of length 2}\}$

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

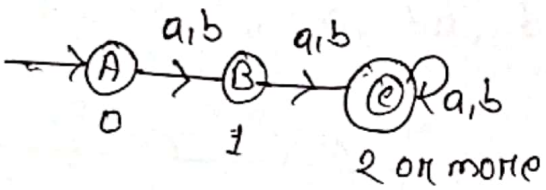
$L = \{aa, ab, ba, bb\}$



Pb:26 $L(M) = \{w | w$ is a string of length at least 2$\}$

$\Sigma = \{a, b\}$ $|w| \geq 2$

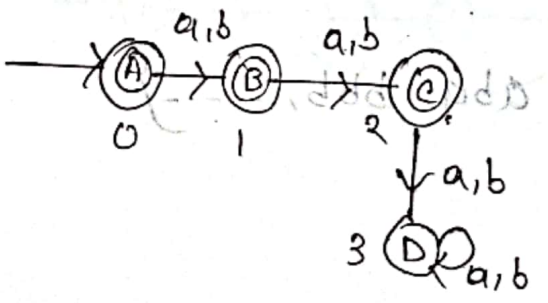
$L = \{aa, ab, ba, bb, aaa, aab, aba, bab, \ldots\}$



Pb:27 $L(M) = \{w | w$ is a string of length at most 2$\}$

$\Sigma = \{a, b\}$ $|w| = 2$, $|w| \geq 2$, $|w| \leq 2$

$L = \{\varepsilon, a, b, ab, ba, aa, bb\}$



Pb:28 Construct a minimal DFA which accepts set of all strings over {a,b} which is divisible by 2}

multiple of 2
|w| mod 2 = 0
even length

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

$L = \{\varepsilon, aa, ab, ba, bb, aaaa, aaab, bbaa, bbbb \ldots\}$

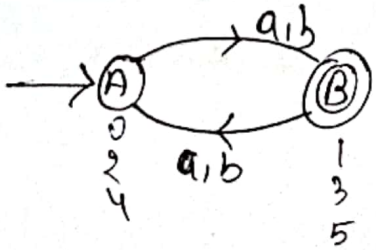
0 1
2 3
4 5
6 7

Pb:29 Construct a mDFA which accepts set of all strings over {a,b} which is not divisible by 2 (not multiple of 2} | not |w| mod2 = 1

L(M) = {w|w has odd length of string}

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

L = {a, b aaa, aba, bbb, ...}



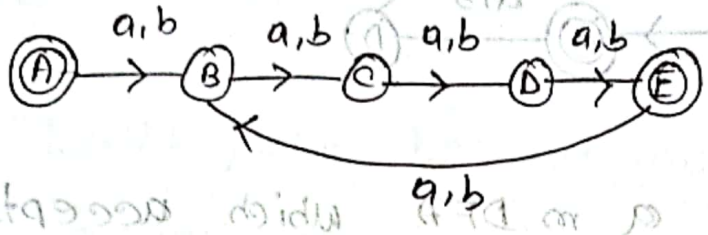
0
2
4

1
3
5

Pb:30 Construct a DFA which accepts set of all strings over {a,b} which is divisible by 4

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

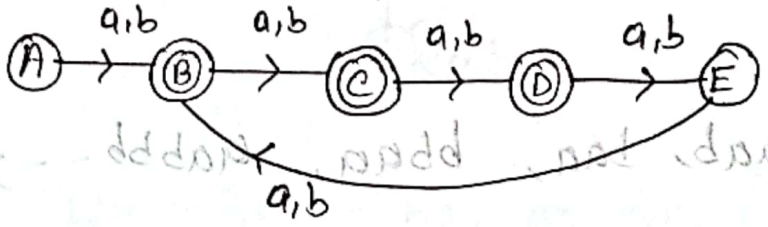
$\Sigma = \{\varepsilon, aaaa, bbbb, abbaa, \ldots\}$



Pb:31 $L(M) = \{w | w$ which is not divisible by 4$\}$

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

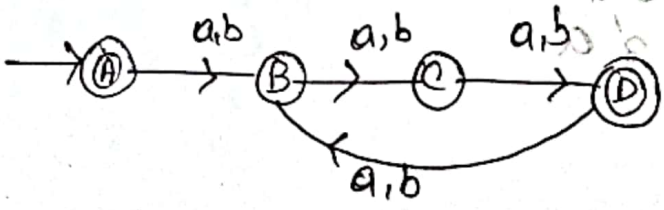
$L = \{a, b, aa, ba, bba, \ldots\}$



Pb:32 $L(M) = \{w | w$ which is divisible by 3$\}$

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

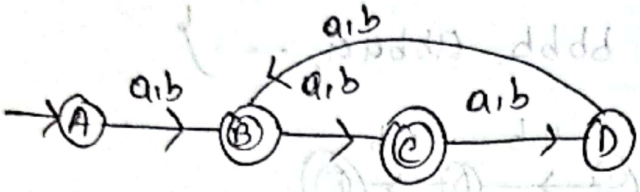
$L = \{\varepsilon, aaa, ababbb, \ldots\}$



Pb 33 $L(M) = \{w | w$ is which is not divisble by 3$\}$

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

$L = \{a, b, ab, bb, aaaa, --\}$

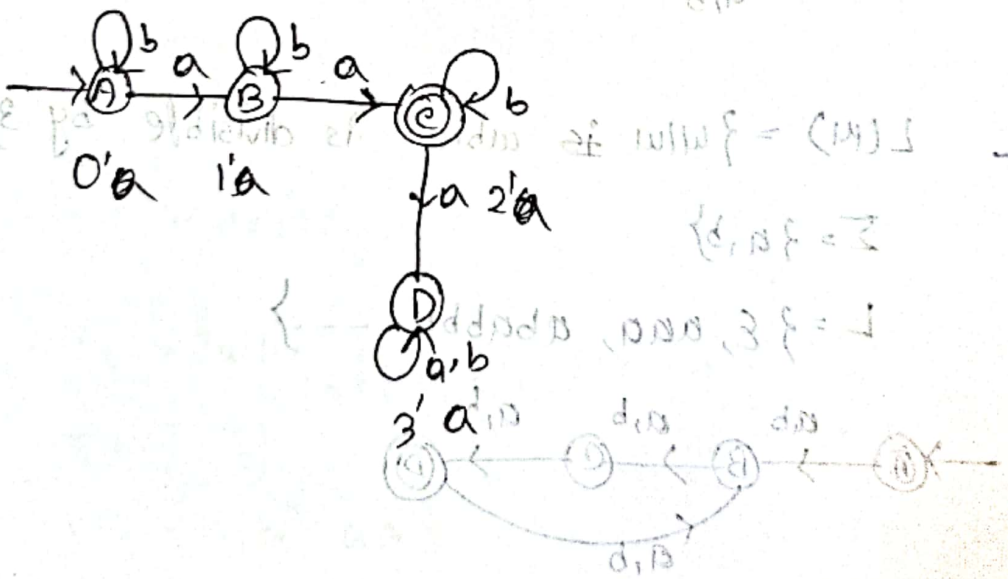


Pb: 34 Construct a m DFA which accepts Sal of all strings over $\{a, b\}$ where each string has exactly two a's.

$L(M) = \{w | w$ has exactly two a's$\}$

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

$L = \{aa, aab, baa, bbaa, aabbb - -\}$



Pb 35 $L(M) = \{w \mid w$ has at least two a's$\}$

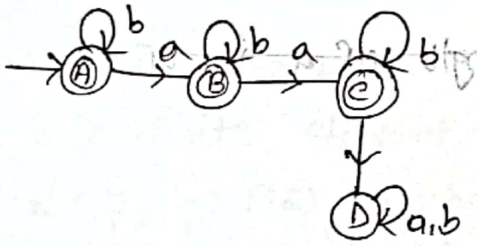
$\Sigma = \{a, b\}$

$L = \{aa, aab, abaa, baa, bbaaa, \ldots\}$

Pb 36 $L(M) = \{w \mid w$ has at most two a's$\}$

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

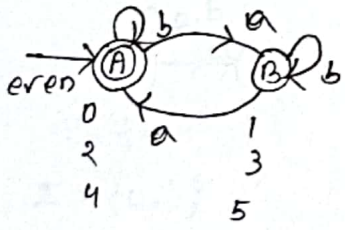
$L = \{\varepsilon, a, aa, b, ab, ba, bba, bbaa \ldots\}$



Pb 37 $L(M) = \{w \mid w$ has an even number of a's$\}$

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

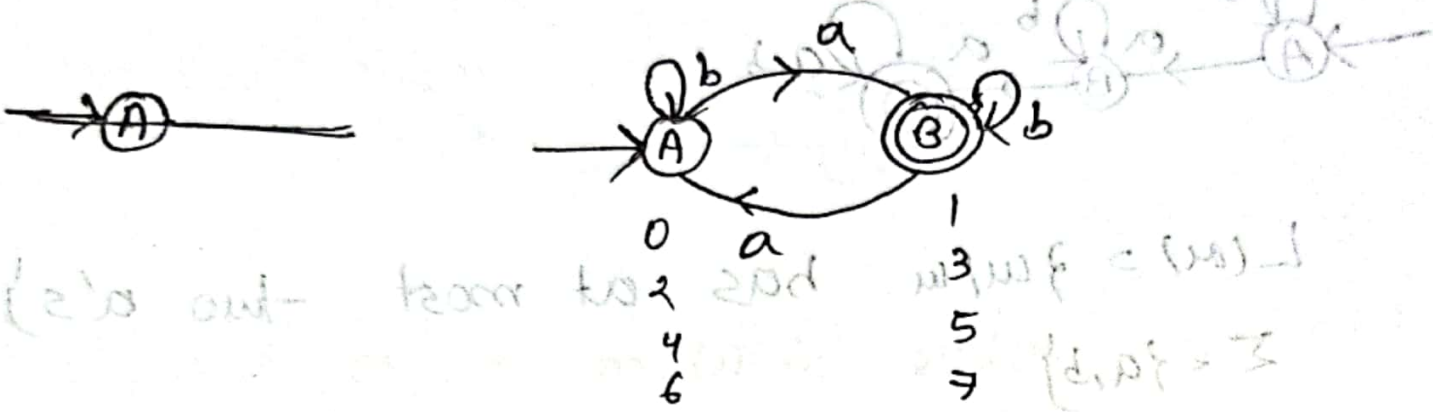
$L = \{\varepsilon, b, aa, bbaa, aaaa, aaaab, \ldots\}$



Pb 38 $L(M) = \{w | w$ has an odd number of a's$\}$

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

$L = \{b, a, ba, bba, aaa, bbaaa, \ldots\}$



Introduction to NFA

Formal definition of NFA

NFA → $(Q, \Sigma, \delta, q_0, F)$

NFA → Death configuration

DFA → Trape state

Q = set of all states

Σ = Input alphabet

q_0 = start state

F = set of final states

① Q is a finite set of states

② Σ is a finite alphabet set

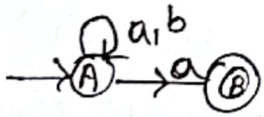
③ $\delta : Q \times \Sigma_E \rightarrow P(Q)$ is the transition function

④ $q_0 \in Q$ is the start state

⑤ $F \subseteq Q$ is the set of accept states

Example: L = {ends with a}

= {a, aa, ba, aaa, ...}



① $Q = \{A, B\}$

② $\Sigma = \{a, b\}$

③ $Q \times \Sigma \rightarrow 2^Q$

$\{A \times B\} \times \{a \times b\}$

→ (A, a), (A, b), (B, a), (B, b)

(A, a) ∅

(A, b) {A}

(B, a) {B}

Subset

∅

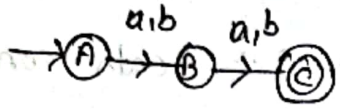
{A}

{B}

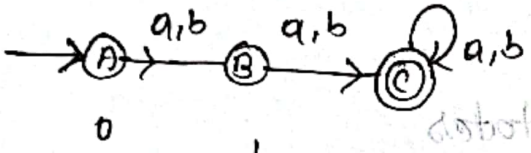
{A, B}

Construction of NFA

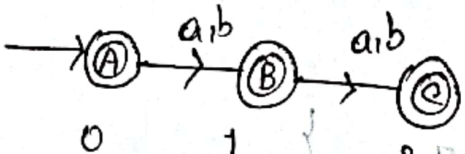
1. L = {w | w length of string exactly 2}
   = {aa, bb, ab, ba}



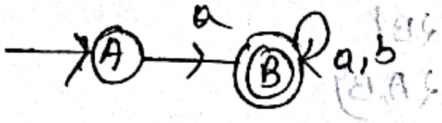
2. L = {w | w length of string at least 2}
   = {aa, bb, ab, ba, aaa, abaabb ...}



3. L = {w | w length of string at most 2}
   = {epsilon, a, b, aa, bb, ab, ba}



4. L = {w | w starts with a}
   = {a, abbaa, aaaab. ..}



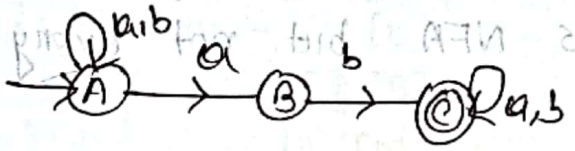
a থাকলেও আরও থাকতে পারে

5. $L = \{w | w$ containing 'a'$\}$

$= \{a, abbbaa, aaab \ldots\}$

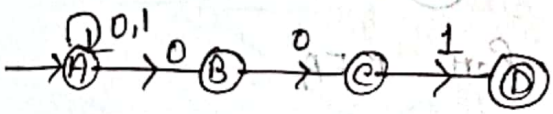
6. $L = \{w | w$ containing 'ab'$\}$

$= \{ab, abbbaa, aaaaab, abab, \ldots\}$



7. $L = \{w | w$ ends with '001'$\}$

$= \{001, 10001, 1111001, \ldots\}$

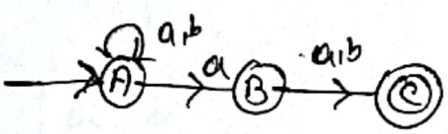


8. $L = \{w | w$ from L.H.S second symbol is 'a'$\}$

$= \{ba, aa, baaa, \ldots\}$

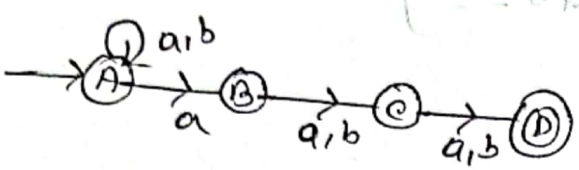
9. $L = \{w | w$ from R.H.S second symbol is 'a'$\}$

$= \{aaa, baa, \ldots\}$



10. $L = \{w | w$ from RHS third symbol is 'a'$\}$

$= \{baaab, bbaaab \ldots\}$



Steps for converting NFA to DFA:

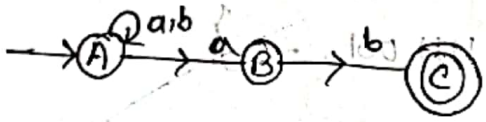
1. Convert the given NFA to its equivalent transition table.
2. Create the DFA's start state
3. Create the DFA's transition tab
4. Create the DFA's final state
5. Simplify the DFA

NFA to DFA conversion

Every ~~DFA~~ DFA is NFA but not every NFA is DFA.

1. $L = \{w | w$ ends with $ab\}$

NFA,



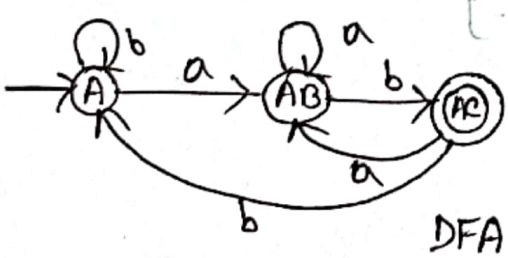
State transition table for NFA

	a	b
→A	{A,B}	{A}
B	∅	{C}
*C	∅	∅

	a	b
→[A]	[AB]	[A]
[AB]	[AB]	[AC]
*[AC]	[AB]	[A]

→ আলাদাভাবে NFA থেকে নিয়ে,

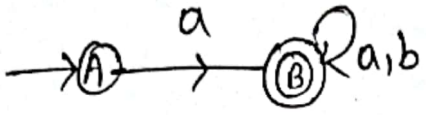
S(NFA) S(DFA)



DFA

2. L = {w/w starting with 'a'}

= {a, ab, aa. ---}



NFA

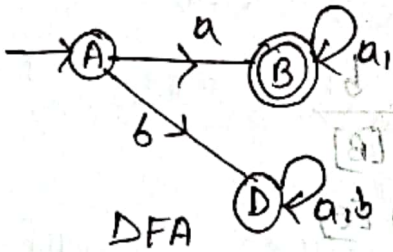
State transition

	a	b
→A	{B}	{ }
*B	{B}	{B}

S(NFA)

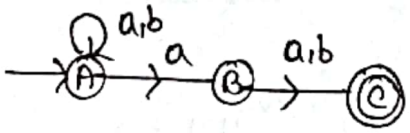
	a	b
→[A]	[B]	[D]
→*[B]	[B]	[B]
[D]	[D]	[D]

S(DFA)



DFA

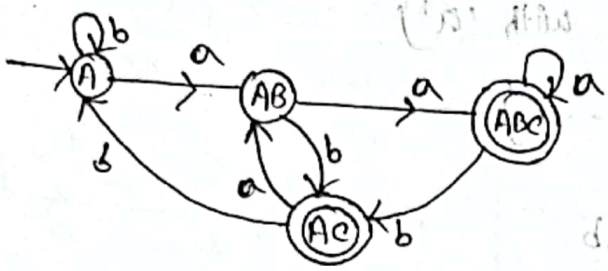
3. L = {w/w second symbol from R.H.S is 'a'}



NFA

	a	b
→A	{A,B}	{A}
B	{C}	{C}
*C	∅	∅

	a	b
→[A]	[AB]	[A]
[AB]	[ABC]	[AC]
*[ABC]	[ABC]	[AC]
*[AC]	[AB]	[A]



DFA

(4) $L = \{w \mid |w| \text{ length of string exactly } 2\}$

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

(NFA)

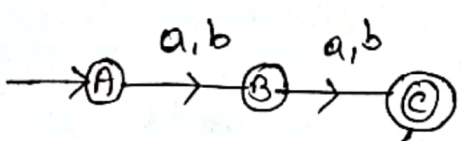
State transition table for NFA

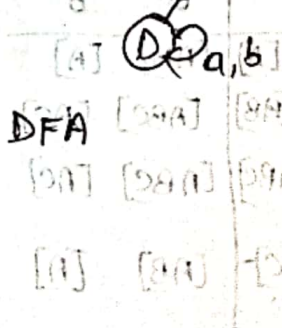
	a	b
→A	{B}	{B}
B	{C}	{C}
*C	∅	∅

S(NFA)

	a	b
→[A]	[B]	[B]
[B]	[C]	[C]
*[C]	[D]	[D]
[D]	[D]	[D]

S(DFA)

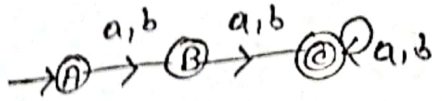




DFA

(a) L = {w/w length of string at least 2} =

= {aa, bb, ab, ba, aaa, abaab---}

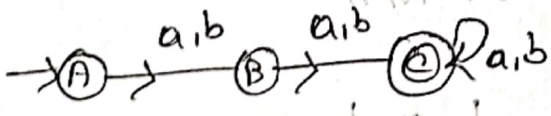


NFA

State transition table for NFA

	a	b
→A	{B}	{B}
B	{C}	{C}
*C	{C}	{C}

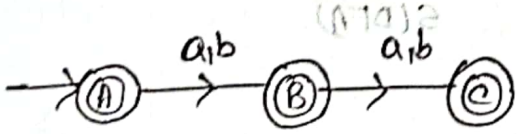
	a	b
→[A]	[B]	[B]
[B]	[C]	[C]
*[C]	[C]	[C]



DFA

(b) L = {w/w length of, starting at most 2}

= {epsilon, a, b, aa, bb, ab, ba}



(NFA)

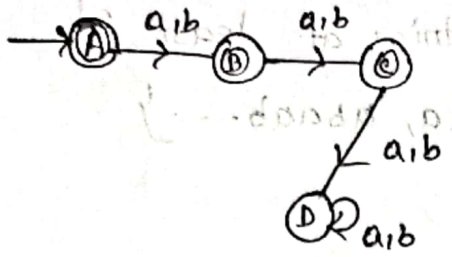
State transition table for NFA

	a	b
→A	{B}	{B}
*B	{C}	{C}
*C	∅	∅

S(NFA)

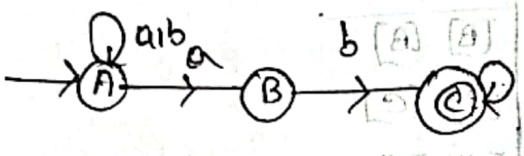
	a	b
[A]	[B]	[B]
[B]	[C]	[C]
[C]	[D]	[D]
[D]	[D]	[D]

S(DFA)



④ L = {w/w containing 'ab'}

= {ab, abbbaa, aaaab, abab, ...}



(NFA)

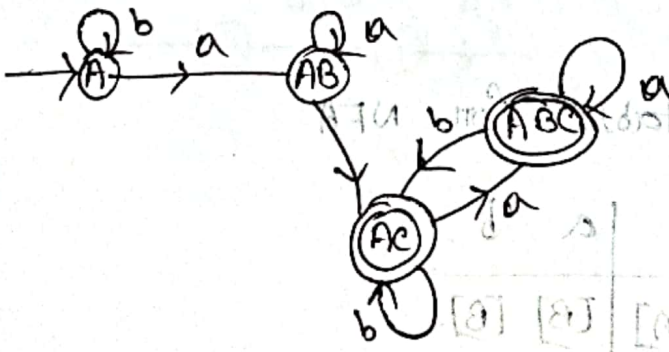
State transition table for NFA

	a	b
→A	{AB}	{A}
B	∅	{C}
*C	{C}	{C}

	a	b
→[A]	[AB]	[A]
[AB]	[AB]	[AC]
*[AC]	[ABC]	[AC]
*[ABC]	[ABC]	[AC]

S(NFA)

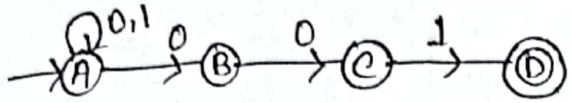
S(DFA)



DFA

(4) $L = \{w / w$ ends with $1001\}$

$= \{1001, 01001, 111001, \ldots\}$



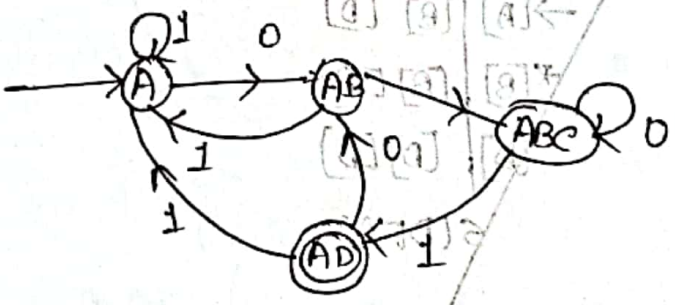
State transition table for NFA

	0	1
→A	{AB}	{A}
B	{C}	∅
C	∅	{D}
*D	∅	∅

S(NFA)

	0	1
→[A]	[AB]	[A]
[AB]	[ABC]	[A]
[ABC]	[ABC]	[AD]
*[AD]	[AB]	[A]

S(DFA)



DFA

Differences between DFA and NFA

DFA	NFA
1. DFA stands for Deterministic Finite Automata.	1. NFA stands for Nondeterministic Finite Automata.
2. DFA cannot use empty string transition	2. NFA can use empty string transition.
3. DFA can be understood as one machine.	3. NFA can be understood as multiple machines computing at the same time
4. DFA is more difficult to construct	4. NFA is easier to construct.
5. All DFA are NFA	5. Not all NFA are DFA.
6. DFA requires more space.	6. NFA requires less space. than DFA.
7. Epsilon is move is not allowed in DFA	7. Epsilon move is allowed in NFA
8. Backtracting is allowed in DFA	8. Backtracting is not always possible in NFA
9. It follows only one path.	9. It chooses between many path.

Regular language: A language is said to be a regular language if and only if some finite state machine recognized it.

Regular expression: The language accepted by finite automata can be described by simple expressions called regular expression

⇒ It is a method to represent regular languages.

⇒ A regular expression can also be described as a sequence of a pattern that defines a string.

Operations on Regular languages:

The various operations on regular languages are:

① Union (A∪B or A+B)

② Concatenation (A°B or AB)

③ Star (A^*) ; * means 0 to many occurences

④ plus (A^+) ; + means 1 to many occurences

Example: Let, A = {xy, P}, B = {Pq, π}

union, A∪B = {xy, P, Pq, π}

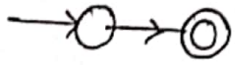
Concatenation, A°B = AB = {xyPq, xyπ, Ppq, Pπ}

Star : A^* = {∈, P, xy, xy, P}

$A^+ = \{ P, xy, xy, P, P, xy \}$

Formal definition of Regular expression:

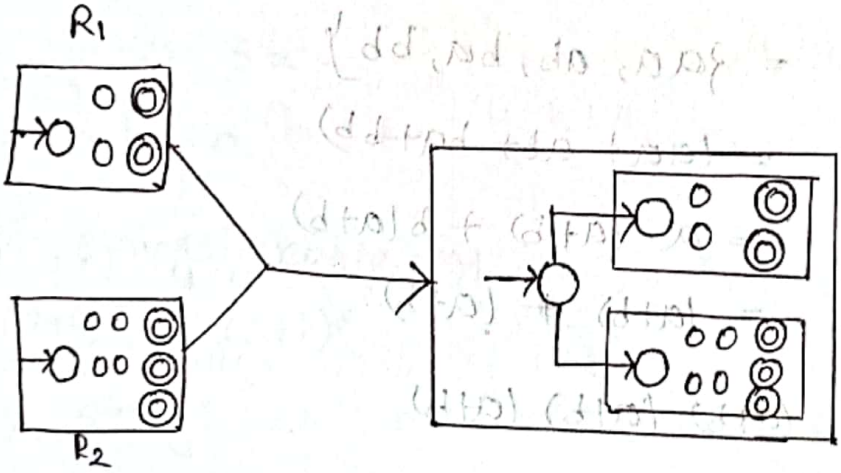
There are 6 cases.

case-1: $R = a$; where $a \in \Sigma$ 

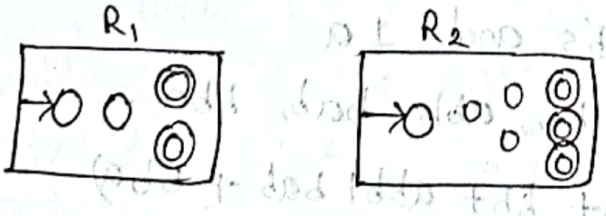
case-2: $R = \epsilon$. 

case 3: $R = \emptyset$ – empty 

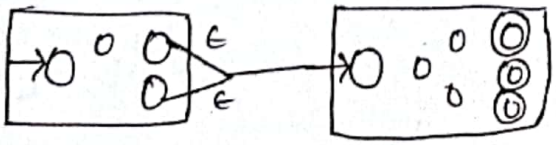
case 4: $R = R_1 \cup R_2$ or R_1 / R_2 where R_1 and R_2 are regular interpression



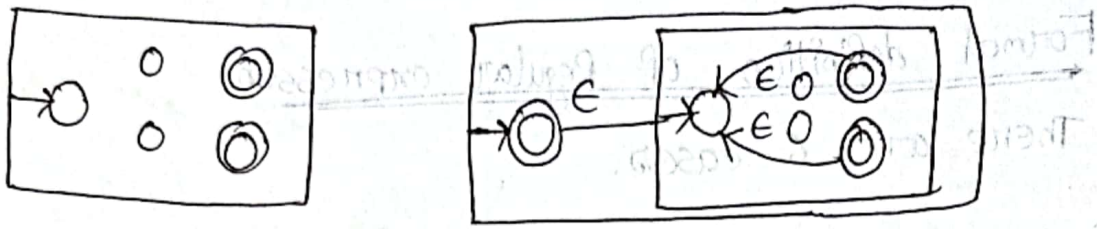
Case-5: $R_1 \cdot R_2$ or $R_1 R_2$ where R_1 and R_2 are regular expression



R_1 এর Final State normal state হয়ে, R_2 same থাকবে,



Case-6: $R_1^* = R$; where R_1 is a regular expression



Regular language to Regular expression:

① No string → $\emptyset$ / $\{\}$

② Length 0 → $\{\epsilon\}$ [Empty string]

③ Length 2 = $\{aa, ab, ba, bb\}$

$= (aa + ab + ba + bb)$

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

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

Length 3 = $(a+b)(a+b)(a+b)$

④ At most 1 → $\{\epsilon, a, b\} = \{\epsilon + a + b\}$

⑤ At most 2 → $\{\epsilon + a + b\}\{\epsilon + a + b\}$

⑥ Not more than 2 b's and 1 a

$= \{\epsilon, b, a, ab, ba, bb, abb, bab, bba \ldots\}$

$= (\epsilon + a + b + ab + ba + bb + abb + bab + bba)$

Example:

① All strings having a single 'b' over $\Sigma = \{a, b\}$

→ $a^* b a^*$

② All strings having at least one 'b'

→ $(a+b)^* b (a+b)^*$

③ All strings having 'bbbb' as substring

→ $(a+b)^* bbbb (a+b)^*$

④ All strings ends with 'ab'

→ $(a+b)^* ab$ [Here '+' use as OR]

⑤ All strings start and ends with 'a'

→ $a (a+b)^* a$

⑥ All strings containing 'a'

→ $(a+b)^* a (a+b)^*$

⑦ All strings start and ends with different symbol

→ case:1 ⇒ $a (a+b)^* b$

case:2 ⇒ $b (a+b)^* a$

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

⑧ All strings having 2 b's exact

→ $a^* b a^* b a^*$

⑨ $\{w | w$ every 0 is followed by at least one 1$\}$

→ $1^* (01^+)^*$

10. {w/w every strings length is odd}

→ $(a+b)((a+b)(a+b))^*$

11. {w/w length is multiple of 3}

→ $((a+b)(a+b)(a+b))^*$

12. Start and ends with same symbol.

case 1: $a(a+b)^*a$

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

∴ $(a(a+b)^*a) + (b+(a+b)^*b)$

13. {w/w does not ends with '11'}

⇒ $((\in+0+1)+(0+1))^*(00+10+01)$

14. {w/w has at least 3 character and 3rd character is '0'}

→ $(0+1)(0+1)0(0+1)^*$

Regular expression to Regular language:

Example:

① 0^*10^*

→ {w/w contains only one '1'}

② $(0+1)^*1(0+1)^*$ / $\Sigma^*1\Sigma^*$

→ {w/w contains at least one '1'}

③ $((a+b)(a+b))^*$

→ {w/w that has even length}

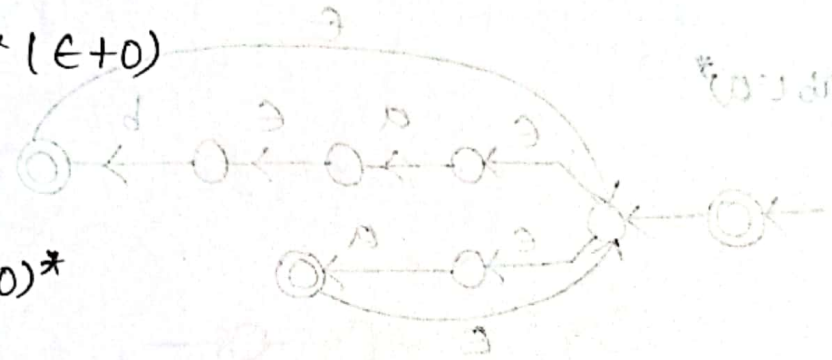
④ which two of the following four are equivalent?

a. $(00)^*(\epsilon+0)$

b. $(00)^*$

c. 0^*

d. $0(00)^*$



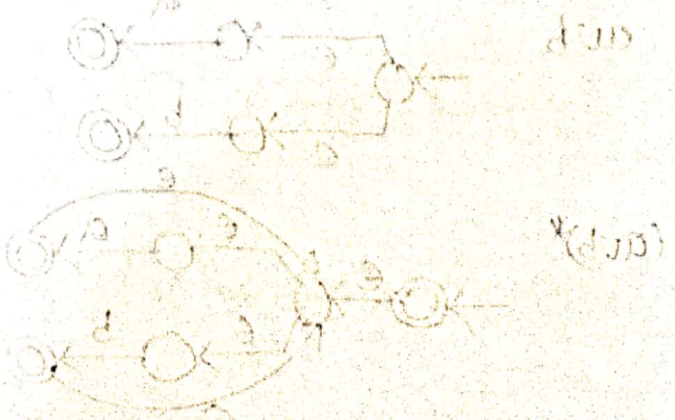
Ans: a. $(00)^*(\epsilon+0) \rightarrow \{\epsilon, 00, 000, 0000, \ldots\}$

b. $(00)^* \rightarrow \{\epsilon, 00, 0000, \ldots\}$

c. $0^* \rightarrow \{\epsilon, 0, 00, 000, \ldots\}$

d. $0(00)^* \rightarrow 0, 000, 00000, \ldots$

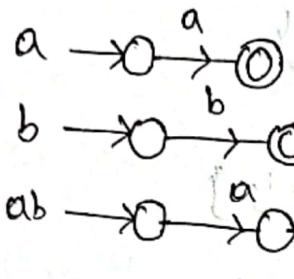
a and c are same.



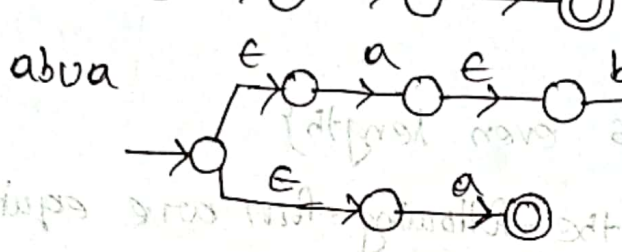
Convert Regular expression to NFA:

① $(ab \cup a)^*$

Soln:

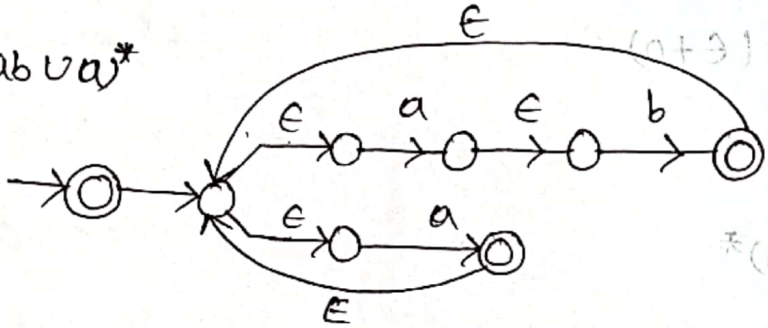


ab



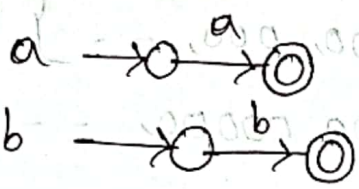
abua

$(ab \cup a)^*$

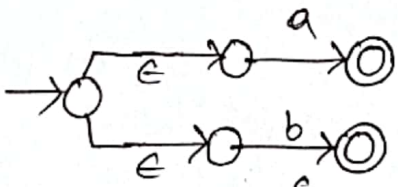


② $(a \cup b)^* aba$

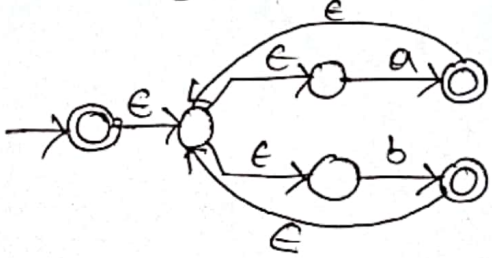
Soln:



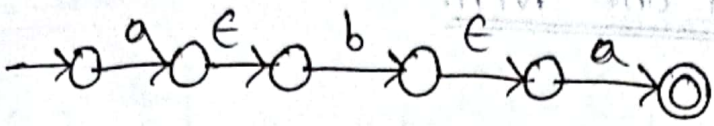
$a \cup b$



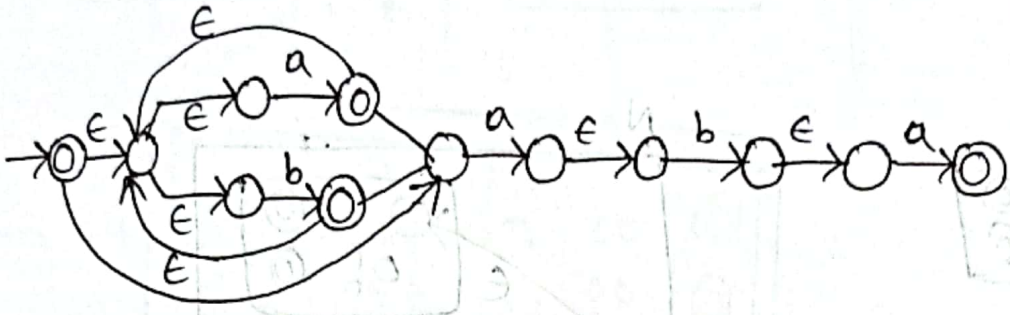
$(a \cup b)^*$



aba



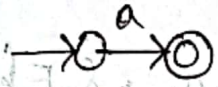
$(a \cup b)^* aba$



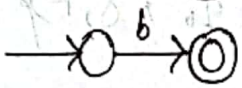
③ $(a \cup b^+)\, a^+ b^+$

Soln:

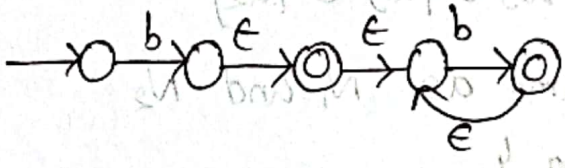
a



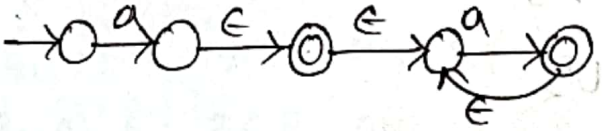
b



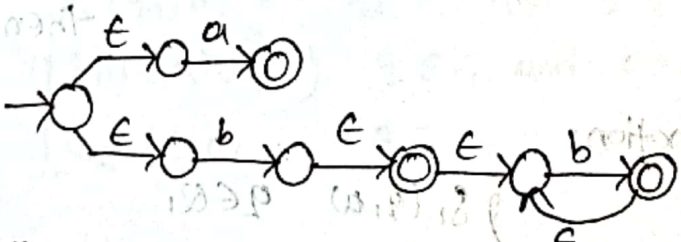
$b^+ = bb^*$



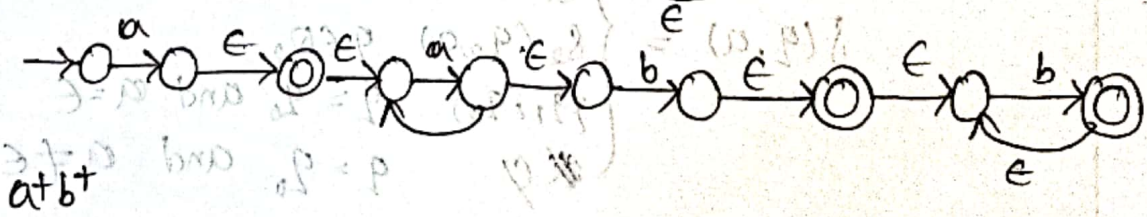
$a^+ = aa^*$



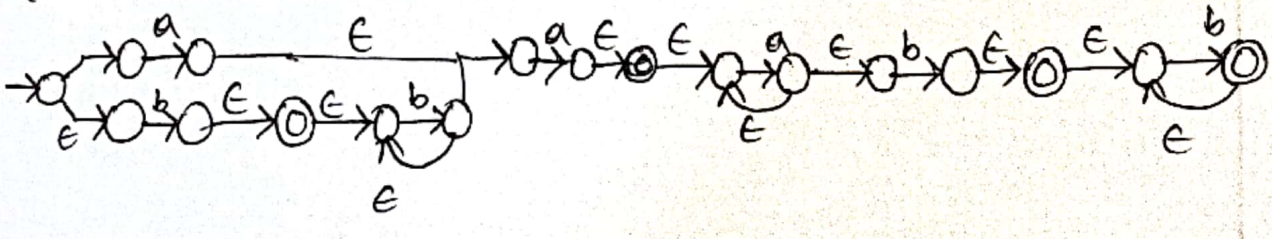
$a \cup b^+$



$a^+ b^+$



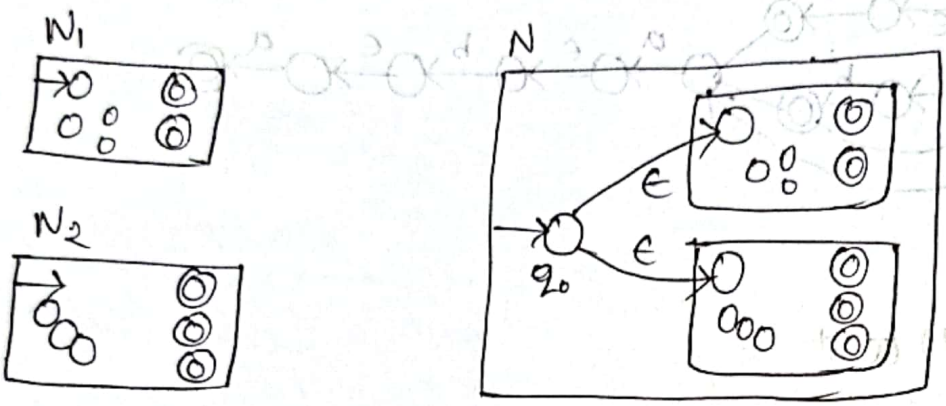
$(a \cup b^+)\, a^+ b^+$



Regular expression one NFA:

Union:

Example-1



Formal definition of N: $\{Q, \Sigma, q_0, \delta, F\}$

$Q = \{q_0\} \cup \{Q_1\} \cup \{Q_2\}$

Σ = same as N_1 and N_2

$q_0 = \{q_0\}$

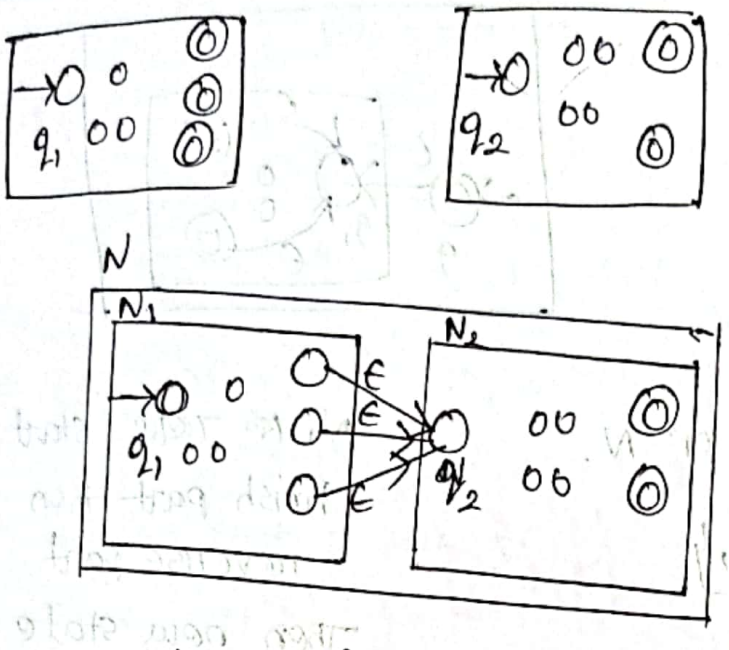
$F = F_1 \cup F_2$

Union:
প্রথমে N_1 আর N_2 এর টা লিখব তারপর start state then $\emptyset$ state.

Transition function:

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 \\ \delta_2(q_2,a) & q \in Q_2 \\ \{q_1, q_2\} & q = q_0 \text{ and } a = \epsilon \\ \emptyset & q = q_0 \text{ and } a \neq \epsilon \end{cases}$$

Concatenation:



N_1 এর শেষ থেকে
N_2 এর start state
-then Final state
→ Concatenation part
→ N_2 এর টা লিখব

Formal definition of N: $\{Q, \Sigma, q_0, \delta, F\}$

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

Σ = same as N_1 and N_2

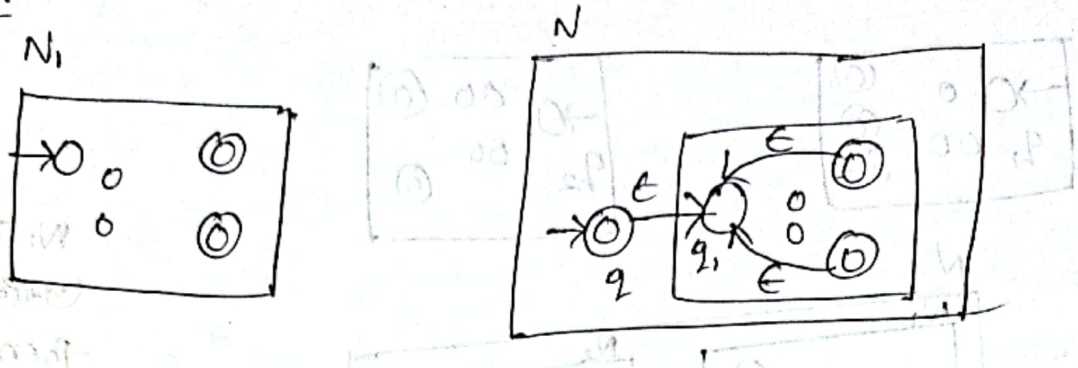
$q_0 = \{q_1\}$

$F = F_2$

Transition function:

$$\delta(q,a) = \begin{cases} \delta_1(q_1,a) & q \in Q \text{ and } q \notin F_1 \\ \delta_1(q_1,a) & q \in F_1 \text{ and } q \neq \epsilon \\ \delta_1(q_1,a) \cup \{q_2\} & q \in F_1 \text{ and } a = \epsilon \\ \delta_2(q,a) & q \in Q_2 \end{cases}$$

Star:



Formal definition of N

$Q = \{q_0\} \cup \{Q_1\}$

Σ = same as N_1

$q_0 = \{q_0\}$

$F = \{q_0\} \cup \{F_1\}$

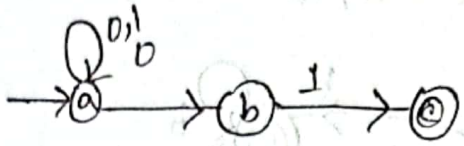
N_1 to make start and Finish part then reverse part then new state then null port

Transition function:

$$\delta(q,a) = \begin{cases} \delta(q,a) & q \in Q_1 \text{ and } q \notin F_1 \\ \delta(q,a) & q \in F_1 \text{ and } a \neq \epsilon \\ \delta_1(q,a) \cup \{q_1\} & q \in F_1 \text{ and } a = \epsilon \\ \{q_1\} & q = q_0 \text{ and } a = \epsilon \\ \emptyset & q = q_0 \text{ and } a \neq \epsilon \end{cases}$$

Computation of NFA:

Example: ①



Compute NFA for 0010

⇒ Computations for input word w = 0010

possible states after 0: {a, b}

after another 0: {a, b}

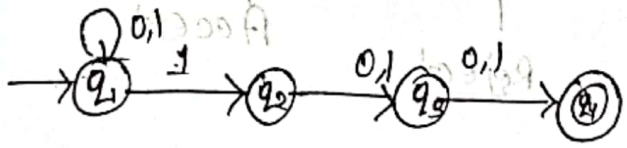
After 1: {a, c}

After final 0: {a, b}

Since neither b or a is accepting, M does not accept 0010

$\{a\} \xrightarrow{0} \{a,b\} \xrightarrow{0} \{a,b\} \xrightarrow{1} \{a,c\} \rightarrow \{a,b\}$

②



compute NFA for 010100

⇒ Computations for input word w = 010100

possible states after 0: {q1}

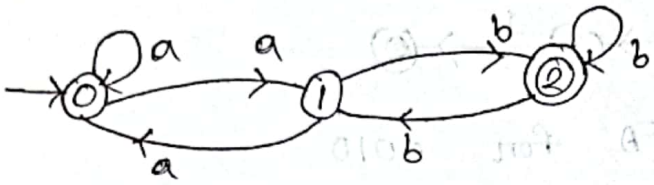
After 1: {q1, q2}

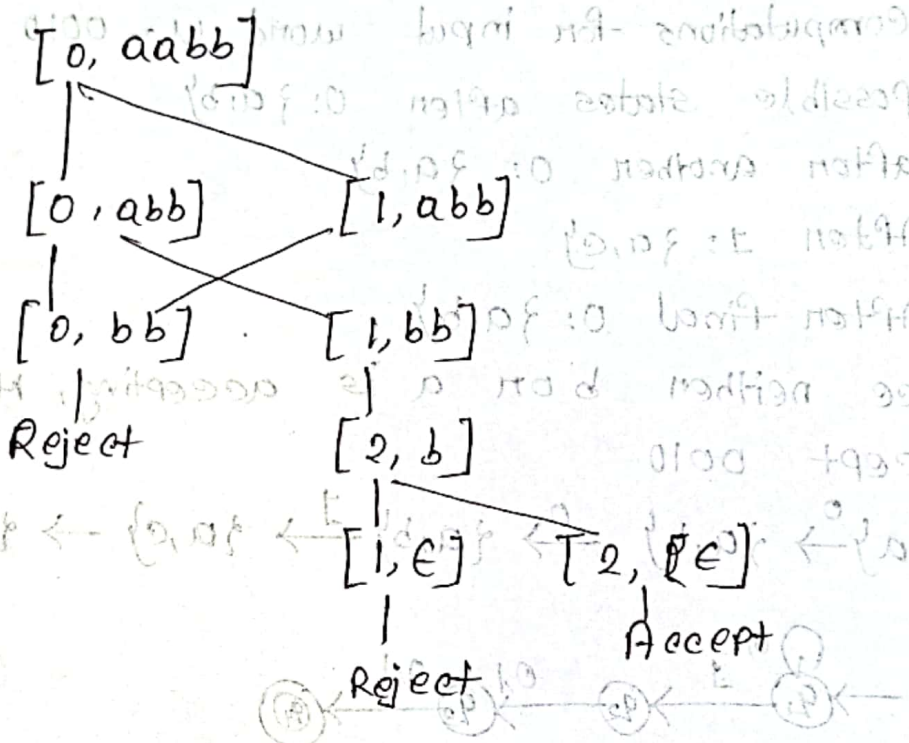
After 0: {q1, q3}

After 1: {q1, q2, q3, q4}

Since q_4 is accepting, M accept 010100.

Draw the computational parse tree of the following NFA on the input aabb





GNFA: A GNFA is a nondeterministic finite automation (NFA) in which each transition is labeled with a regular expression over the alphabet set instead of only members of the alphabet or ϵ.

Formal definition of GNFA:

A GNFA is a 5 tuple $(Q, \Sigma, \delta, q_{start}, q_{accept})$ where,

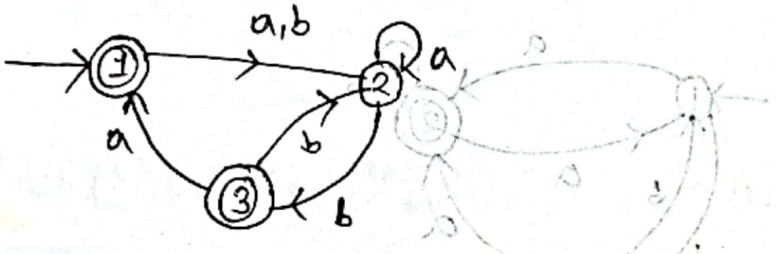
- Q is the finite set of states
- Σ is the input alphabet
- $\delta : (Q - \{q_{accept}\}) \times (Q - \{q_{start}\}) \to R$ is the transition function
- q_{start} is the start ~~date~~ state and
- q_{accept} is the accept date

DFA to Regular expression:

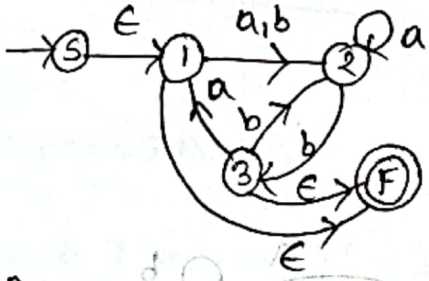
Steps: 1. convert the DFA to GNFA by adding a start state with an ϵ arrow with old start and a new accept state with an ϵ arrow with old accept and make previous final states to normal state.

2. Remove the middle order states by ascending order.

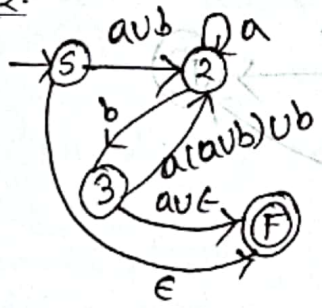
Example: ①

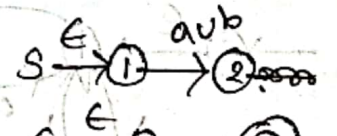


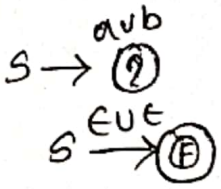
Soln: Step-1:

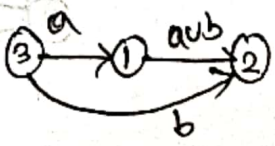


Step-2:

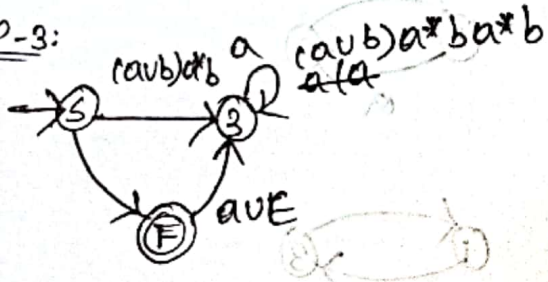


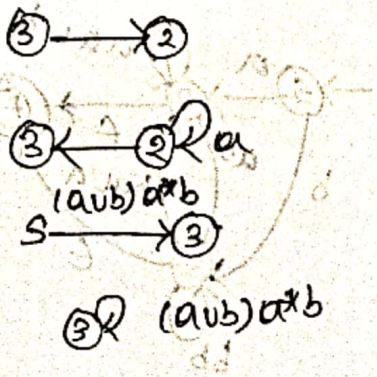






Step-3:

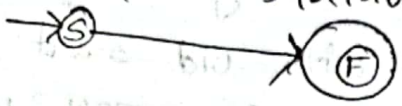




(aub)a*b

Step-4

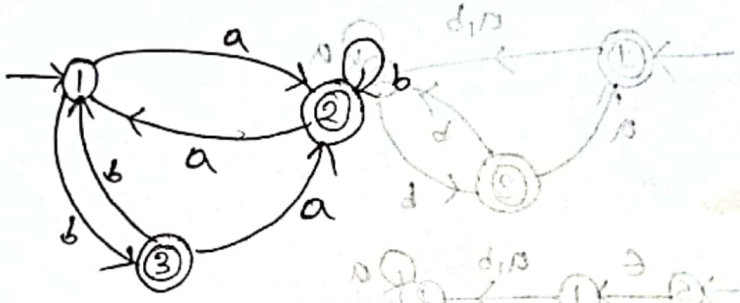
$(1a \cup b)a^*b\,(a\,|\,a \cup b) \cup ba^*b)^*.\ a \cup \epsilon) \cup \epsilon$



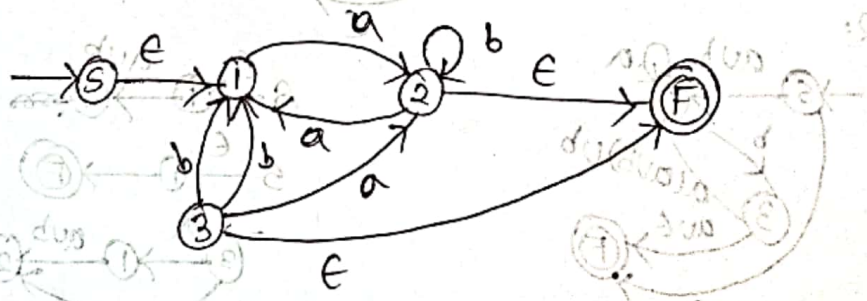
The regular expression is,

$$(1a \cup b)\ a^*b.\ (a\ (a \cup b) \cup ba^*b)^*.\ a \cup \epsilon) \cup \epsilon$$

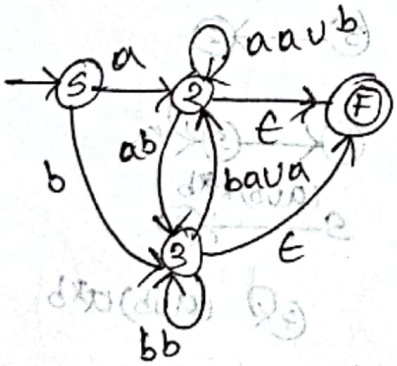
Example-2

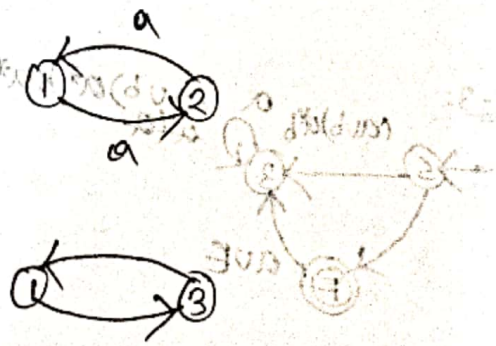


Soln: Step:1

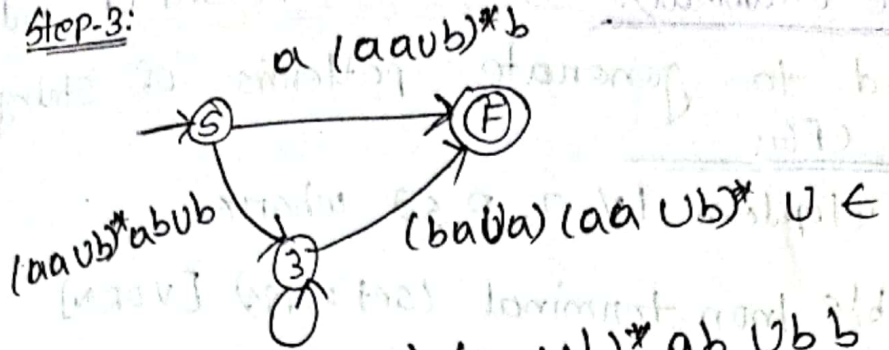


Step-2:

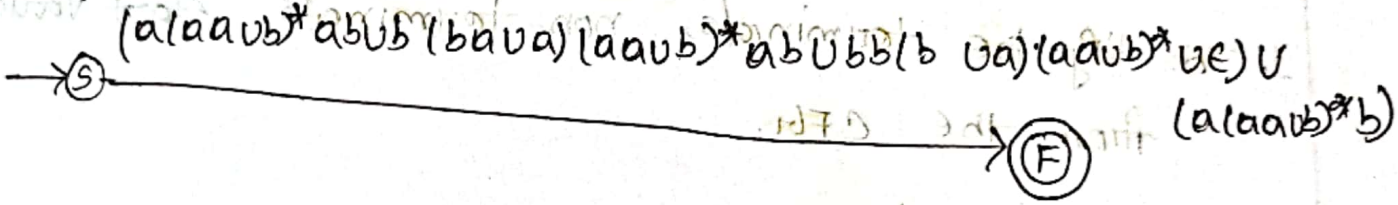




Step-3:



Step -4:



The regular expression is,

(1a(aaUb)* abUb (baUa) (aaUb)*ab Ubb (baUa) (aaUb)*U∈)
U (a (aaUb)*b)

CFG (Context Free Grammar): A CFG is a set of rules/ Productions used to generate patterns of strings.

Formal defn of CFG:

A CFG is a 4tuples (V, T, P, S) where,

V = Variable /non terminal (set আকারে) [V OR N]

T = Set of terminals [T or Σ]

P = Production /Rules [P OR R]

S = Start Variable

1. Identify the terminals, non-terminals, start Variable from the CFG.

$E \Rightarrow E+T \mid T$

$T \Rightarrow T*F \mid F$

$F \Rightarrow (E) \mid id$

Soln: $V = \{E, T, F\}$ [বাম side এর সব Variable]

$T = \{+, *, (,), id\}$ [ডানদিকে যা আছে Variable ছাড়া বাকিগুলো terminal]

$S = E$

2. $S \Rightarrow (L) \mid a$

$L \Rightarrow L, S \mid S$

Soln: $V = \{S, L\}$

$T = \{(,), a, ,\}$

$S = S$.

Proof:

① The class of regular languages is closed under the union operation.

$N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognise A1,

$N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognise A2,

Let construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognise $A1 \cup A2$

① $Q = \{q_0\} \cup Q1 \cup Q2$

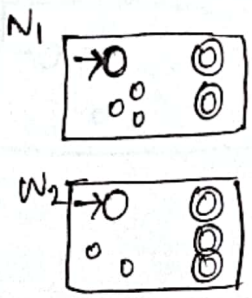
The states of N are all the states of N1 and N2 with the addition of a new state q_0.

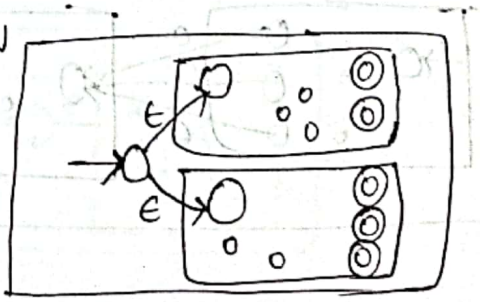
② The state q_0 is the start of N.

③ The set of accept states $F = F1 \cup F2$. The accept states of N are all the accept states of N1 and N2 That way, N accepts if either N1 accepts or N2 accepts.

④ Define δ so that for any $q \in Q$ and any $\in \Sigma_\epsilon$,

$$\delta(q,a) \begin{cases} \delta_1(q,a) & q \in Q_1 \\ \delta_2(q,a) & q \in Q_2 \\ \{q_1, q_2\} & q = q_0 \text{ and } a = \epsilon \\ \emptyset & q = q_0 \text{ and } a \neq \epsilon \end{cases}$$





② The class of Regular languages is closed under the concatenation operation.

Let, $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1

$N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize A_2

Construct $N = (Q, \Sigma, \delta, q_1, F_2)$ to recognize $A_1 \circ A_2$

① $Q = Q_1 \cup Q_2$

The states of N are all the states of N_1 and N_2

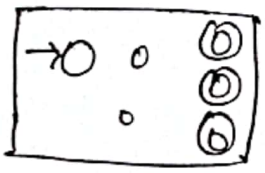
② The state q_1 is the same as the ~~output~~ start state of N_1.

③ The accept states F_2 are the same as the accept states of N_2

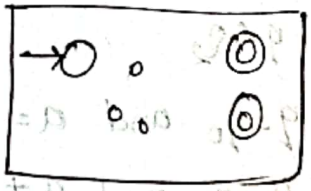
④ Define δ so that for any $q \in Q$ and any $a \in \Sigma_\epsilon$

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q,a) & q \in F_1 \text{ and } a \neq \epsilon \\ \delta_1(q,a) \cup \{q_2\} & q \in F_1 \text{ and } a = \epsilon \\ \delta_2(q,a) & q \in Q_2 \end{cases}$$

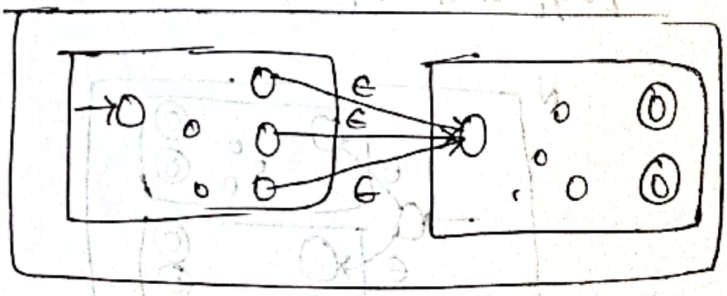
N_1



N_2



N



③ The class of Regular languages closed under star operation.

⇒ Let, $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1

Construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize A_1^*

① $Q = \{q_0\} \cup Q_1$

The states of N are the states of N_1 plus a new start state.

② The state q_0 is the new start state.

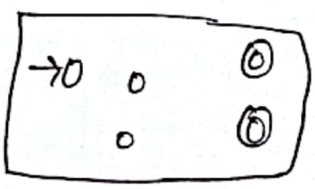
③ $F = \{q_0\} \cup F_1$

④ The accept states are the old accept states plus the new start state.

⑤ Define δ so that for any $q \in Q$ and any $a \in \Sigma_\epsilon$

$$\delta(q,a) = \begin{cases} \delta_1(q_0,a) & q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q,a) & q \in F_1 \text{ and } a \neq \epsilon \\ \delta_1(q,a) \cup \{q_1\} & q \in F_1 \text{ and } a = \epsilon \\ \{q_1\} & q = q_0 \text{ and } a = \epsilon \\ \emptyset & q = q_0 \text{ and } a \neq \epsilon \end{cases}$$

N_1



N