

Course Name : Theory Of Computation (TOC)
Course Code : CS301

Prepared by EBIN P.M , Assistant Professor , IESCE

1

MODULE 1

**INTRODUCTION TO AUTOMATA THEORY &
ITS SIGNIFICANCE**

Prepared by EBIN P.M , Assistant Professor , IESCE

2

Introduction of Theory of Computation (TOC)

- **Theory Of Computation (TOC)** is also known as **Automata** theory
- TOC is a theoretical branch of Computer Science and Mathematics, which mainly deals with the logic of computation with respect to simple machines, referred to as automata.
- **Automata** - An automaton (Automata in plural) is an abstract **self-propelled computing device** which follows a predetermined sequence of operations automatically.
- An automaton with a finite number of states is called a **Finite Automaton (FA)** or **Finite State Machine (FSM)**



Prepared by EBIN P.M , Assistant Professor , IESCE

3

- Automata is a machine that can accept the **Strings** of a **Language L** over an **input alphabet Σ** .

TOC BASICS

- **Symbols** – Symbols are the basic building blocks of TOC. It can be any letter, picture or number.

Eg : a, b, c, 0, 1, 2,  , 

- **Alphabets (Σ)** – Collection of symbols which are always finite.

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

$\Sigma = \{0, 1, 2, 3, \dots, 9\}$ Decimal alphabet

$\Sigma = \{a, b, c, \dots, z\}$ English alphabet

Prepared by EBIN P.M , Assistant Professor , IESCE

4

➤ **Strings** – Sequence of symbols taken from Σ .

Eg: a, b, aa, ab, abc,.....

- **Length of a string**: It is the number of symbols present in a string. It is denoted by $|S|$.

Eg: if $S=cabcad$. Then $|S|=6$

- **Empty string** : if $|S|=0$, it is called empty string (denoted by ϵ or λ)

Q: Find the number of finite 2 length string using the alphabet $\Sigma=\{a,b,c\}$?

Ans : aa, ab, ac, ba, bb, bc, ca, cb, cc

Prepared by EBIN P.M , Assistant Professor , IESCE

5

❖ Let alphabet $\Sigma=\{a,b\}$. How many strings of length 2 is possible from this alphabet?

Ans : aa, ab, ba, bb . ($2^2 = 4$)

How many strings of length n are possible from this alphabet

Ans = $\{a,b\} \{a,b\} \{a,b\} \dots \dots \dots n$

= $2 \times 2 \times 2 \times 2 \times \dots \dots \dots n$

= 2^n (Since we have only 2 symbols in this alphabet, we got 2^n)

- If number of symbols in Σ is $|\Sigma|$, then number of strings of length n possible over $|\Sigma|$ is $|\Sigma|^n$

Prepared by EBIN P.M , Assistant Professor , IESCE

6

➤ **Language** – Collection of strings.

Let $\Sigma = \{a,b\}$, then languages can be define like

$L_1 =$ set of all strings of length 2

$= \{aa, ab, ba, bb\}$ It is a finite set . So, L_1 is a Finite Language.

$L_2 =$ set of all strings of length 3

$= \{aaa, aab, aba, abb, baa, bab, bba, bbb\}$. It is a finite set. So L_2 is a Finite Language

$L_3 =$ set of all strings where each string starts with 'a'.

$= \{a, aa, ab, aaa, aab, aba, abb, \dots\}$. It is an infinite set. So L_3 is an Infinite Language.

Language can be Finite or Infinite

Prepared by EBIN P.M , Assistant Professor , IESCE

7

➤ **Power of Σ**

• Let $\Sigma = \{a, b\}$.

• $\Sigma^0 =$ set of all strings of length 0.

$= \{\epsilon\}$ [ϵ is a special symbol of length 0. ie, $|\epsilon| = 0$]

• $\Sigma^1 =$ set of all strings over Σ of length 1.

$= \{a, b\}$

• $\Sigma^2 =$ set of all strings over Σ of length 2

$= \Sigma.\Sigma = \{a, b\} . \{a, b\} = \{aa, ab, ba, bb\}$

• $\Sigma^3 = \Sigma.\Sigma.\Sigma =$ set of all strings over Σ of length 3

Cardinality = $|\Sigma^3| = 8$

• $\Sigma^n =$ n length strings

Prepared by EBIN P.M , Assistant Professor , IESCE

8

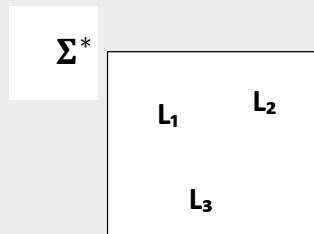
➤ Σ^*

- In place of $*$, we can substitute any number starting from 0.
- $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \dots\dots\dots$

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

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

Σ^* = a set of all possible strings over $\{a, b\}$ of all lengths. It can be considered as a universal set. So Σ^* contains every thing. It is infinite.



$L_1 \subseteq \Sigma^*$ (L_1 is a subset of Σ^*)

$L_2 \subseteq \Sigma^*$ (L_2 is a subset of Σ^*)

$L_3 \subseteq \Sigma^*$ (L_3 is a subset of Σ^*)

Prepared by EBIN P.M, Assistant Professor, IESCE

9

➤ Formal Language

- A formal language L over an alphabet Σ is a subset of Σ^* , that is, a set of strings over that alphabet.
- A formal language consists of strings whose letters are taken from an alphabet and are **well-formed according to a specific set of rules**.

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

$$a^n \mid n \geq 0$$

$$L = \{\epsilon, a, aa, aaa, aaaa, \dots\dots\dots\}$$

$$a^n b^n \mid n \geq 0$$

$$L = \{\epsilon, ab, aabb, aaabbb, \dots\dots\dots\}$$

Prepared by EBIN P.M, Assistant Professor, IESCE

10

➤ Where we use Languages, strings in practical applications?

- Consider C programming Language. The symbols used in C are {a, b,....z, A, B,....Z, 0, 1, 2,.....9, +, *, #,..... }. It is a finite set of symbols. This set is actually **Alphabet Σ** .
- $\Sigma = \{a, b, \dots, z, A, B, \dots, Z, 0, 1, 2, \dots, 9, +, *, \#, \dots\}$
- Using this alphabet we are forming strings , shown below

```
void main()
{
    Int a, b;      (In C, it is a program. But in TOC, it is a string)
    .....
    .....
}
```

Prepared by EBIN P.M , Assistant Professor , IESCE

11

- What is C programming language? . It is a set of all **valid** programs.
- What are the valid programs? How can we say a program is valid or not?
- We know that, there are infinite number of valid programs are possible in C language. Ie, $\{p_1, p_2, p_3, \dots\}$
- Suppose, given a C program P_n , how can we say that it is valid? One solution is that to save all possible programs and check it with our program P_n . Since computer has a finite memory and set of all valid programs are infinite, we can't do this technique.
- Ie, we can give a Language L and a string S and check whether this string is a part of this language or not. If the language is finite, we can find it but if the language is infinite it is not possible

Prepared by EBIN P.M , Assistant Professor , IESCE

12

• Let $\Sigma = \{a, b\}$

$L_1 = \{aa, ab, ba, bb\}$. Check whether the string "aaa" is a part of this language or not?

Ans: no. Because L_1 is finite language.

$L_2 = \{a, aa, aaa, ab, \dots\}$. Check whether the string "baba" is a part of this language or not?

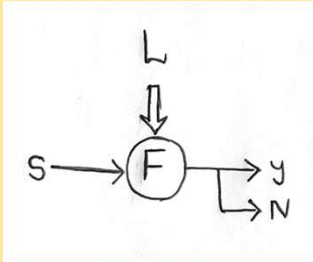
Ans: because L_2 is infinite, we should check with all strings in the language. Instead of this, we can use an easy method.

For a given infinite language, we can create a finite representation.

Prepared by EBIN P.M , Assistant Professor , IESCE

13

➤ What is Finite representation



➤ The finite representation is called **FINITE AUTOMATA (FA)**, which is a small machine.

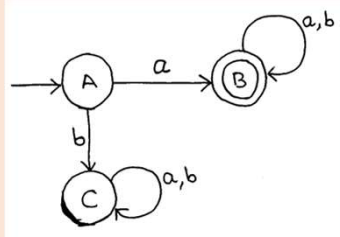
Prepared by EBIN P.M , Assistant Professor , IESCE

14

➤ Let alphabet $\Sigma = \{a, b\}$. Consider the language $L_1 = \text{set of all strings which starts with "a"}$.

$L_1 = \{a, aa, ab, aaa, \dots\}$ which is infinite.

- The finite representation of the above language is given as follows. It is called **Finite Automata (FA)**.

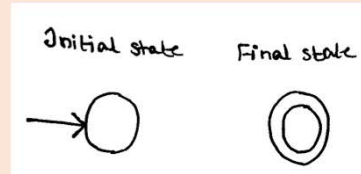


* The circles are called states.

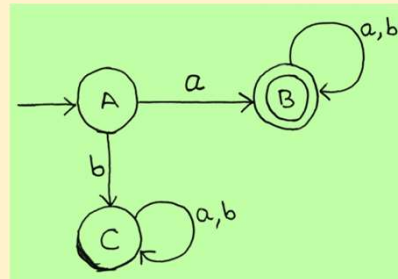
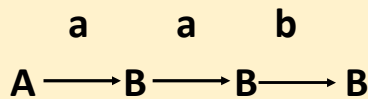
* A, B and C are states.

* A is called **initial state**

* B is called **final state**

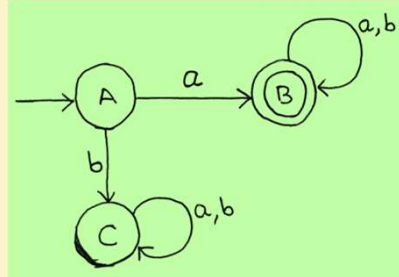
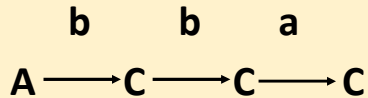


➤ Eg: Consider a string "aab". We can find whether this string is a part of the above language or not using the following FA



- Up on scanning the entire string, the state is in final state. We reach from initial state to final state. So, the string is accepted.

➤ Eg: Consider a string “bba” . We can find whether this string is a part of the above language or not using the following FA



- Up on scanning the entire string, the state is not in final state. So, the string is not accepted and is not present in the Language.

Prepared by EBIN P.M , Assistant Professor , IESCE

17

FINITE AUTOMATA

- An automaton with a finite number of states is called a **Finite Automaton (FA)** or **Finite State Machine (FSM)**
- Finite automata are used to recognize patterns
- It takes the string of symbol as input and changes its state accordingly. When the desired symbol is found, then the transition occurs.
- At the time of transition, the automata can either move to the next state or stay in the same state.
- Finite automata have two states, **Accept state** or **Reject state**. When the input string is processed successfully, and the automata reached its final state, then it will accept.

Prepared by EBIN P.M , Assistant Professor , IESCE

18

Formal Definition of FA

➤ A finite automaton is a collection of 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where:

Q : finite set of states

Σ : finite set of the input alphabet

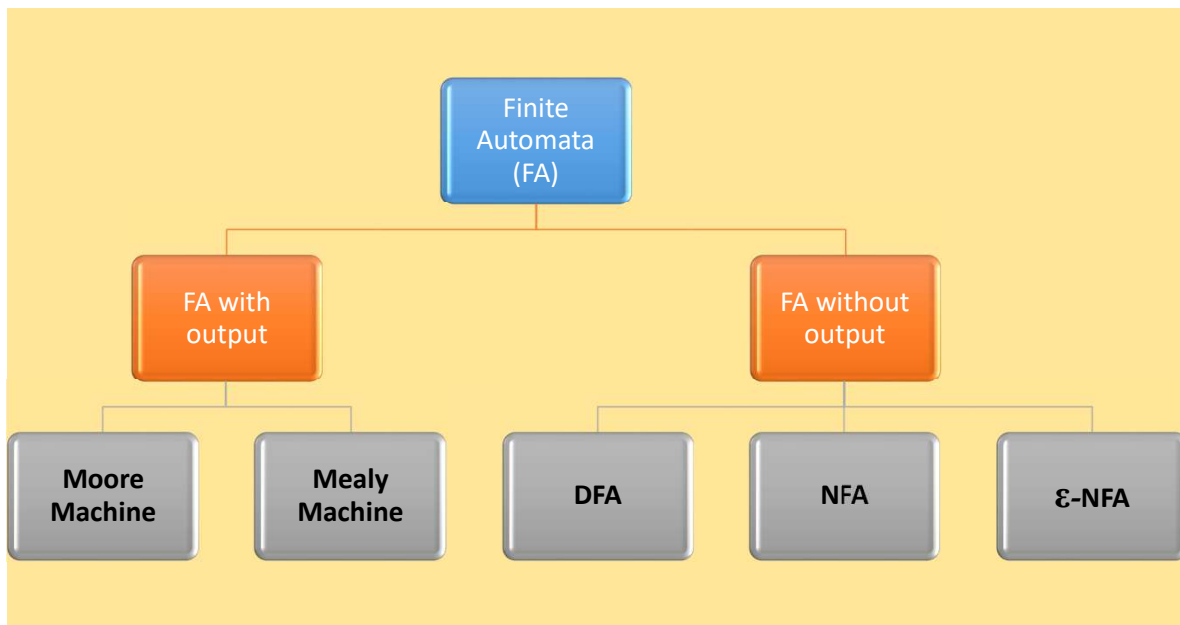
q_0 : initial state (start state)

F : Set of final states

δ : Transition function

Prepared by EBIN P.M , Assistant Professor , IESCE

19

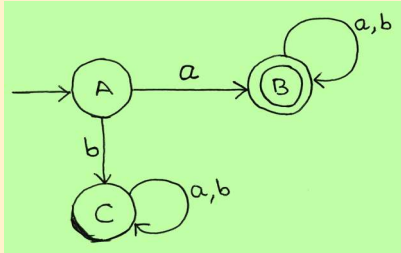


Prepared by EBIN P.M , Assistant Professor , IESCE

20

Deterministic Finite Automata (DFA)

- Consider the following FA



Here $Q =$ (Set of all states) A, B, C
 $\Sigma =$ (input alphabet) {a, b}
 $q_0 =$ (start state) A
 $F =$ (set of all final states) B
 More than one final state is possible.

- The relation between all the states and the final state is $Q \supseteq F$.
- Q, Σ, q_0 and F are common for DFA, NFA and ϵ -NFA. The only change is in δ .

Prepared by EBIN P.M, Assistant Professor, IESCE

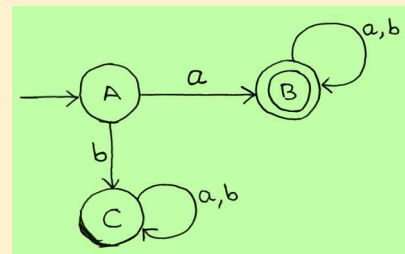
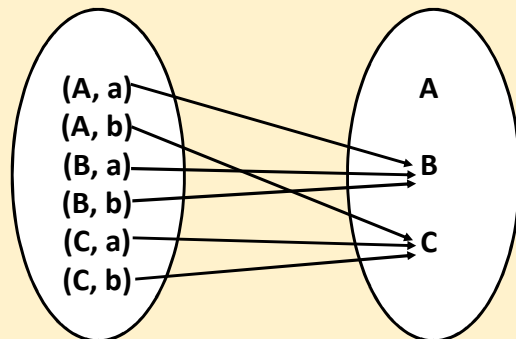
21

- δ is a transition function from $Q \times \Sigma \rightarrow Q$

That is $\{A, B, C\} \times \{a, b\}$
 $= (A, a) (A, b) (B, a) (B, b) (C, a) (C, b)$

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

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



Prepared by EBIN P.M, Assistant Professor, IESCE

22

➤ A Finite Automata (FA) is said to be deterministic, if corresponding to an input symbol, there is single resultant state i.e. there is only one transition.

➤ Eg: DFA is used in Compiler Design.

Construction of DFA

Q: Construct a DFA that accepts set of all strings over $\{a, b\}$ of length 2

• First, identify Σ

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

• Second, identify language

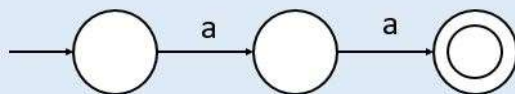
$$L = \{aa, ab, ba, bb\}$$

Prepared by EBIN P.M, Assistant Professor, IESCE

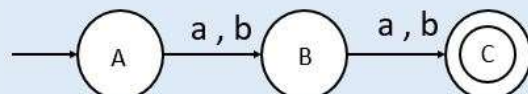
23

• Take the smallest string, "aa" from the language $L = \{aa, ab, ba, bb\}$

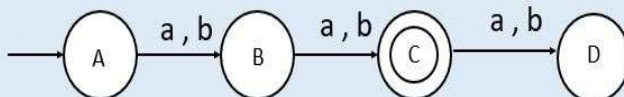
• Construct a skeleton



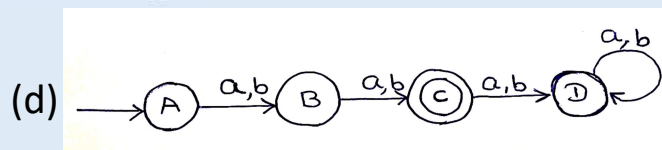
(a)



(b)

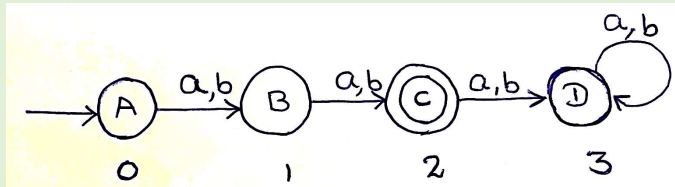


(c)



Prepared by EBIN P.M, Assistant Professor, IESCE

24



- If more than two length string comes, it will go to dead state D
- Once to reach D , it will never go to previous state. So state D is called **dead state**.
- A string is said to be accepted by a FA, if we are able to reach the final state starting from initial state up on scanning entire string, then the string is said to be **accepted** by the FA.
- If not reach to the final state, the string is said to be **Rejected**

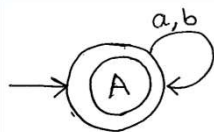
Prepared by EBIN P.M , Assistant Professor , IESCE

25

➤ **String Accept** : Scan the entire string and if we reach a **final** state from the initial state, we can say that the string is accepted. (more than one final state is possible)

➤ **Language Accept** : A finite automata is said to accept a language, if all the strings in the language are accepted **and all the strings not in the language are rejected**.

- The following FA accepts all the string in the language $L=\{aa, ab, ba, bb\}$. Here the same state act as initial and final states. But it does not satisfies the second condition, because it accept anything.



Prepared by EBIN P.M , Assistant Professor , IESCE

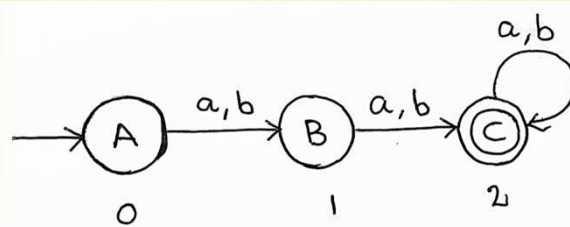
26

Q: Construct a DFA that accepts set of all strings w over $\{a, b\}$ where $|w| \geq 2$

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

$L = \{aa, ab, ba, bb, aaa, \dots, bbb, \dots\}$ this is infinite language.

- If a language is finite, we can create finite automata.
- If a language is infinite, we can't predict whether it able to give FA or not. We shall try to give a finite automata for this.
- Take the smallest string "aa"



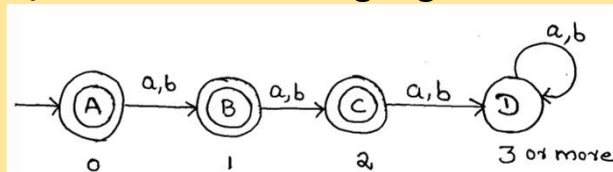
Prepared by EBIN P.M, Assistant Professor, IESCE

27

Q: Construct a DFA that accepts set of all strings w over $\{a, b\}$ where $|w| \leq 2$ (at most 2)

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

$L = \{\epsilon, a, b, aa, ab, ba, bb\}$ this is a finite language. So we can create FA definitely.

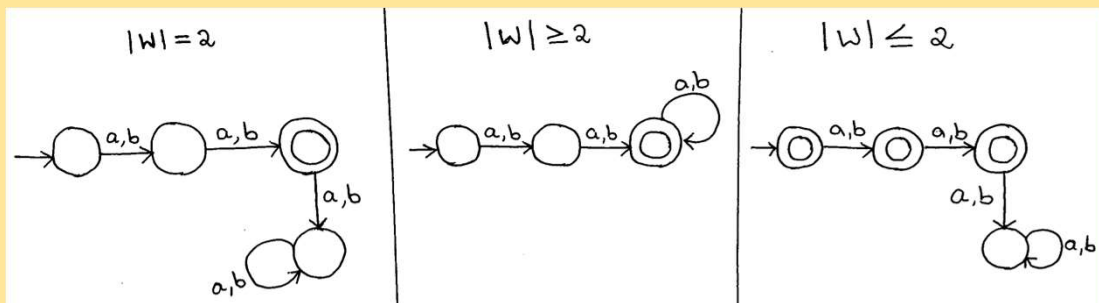


- In the language ϵ is there and $|\epsilon| = 0$. So to accept ϵ we should make the initial state as final state.
- State B will accept one length strings. D is dead state.
- One language can have many DFA. But **only one minimal DFA**

Prepared by EBIN P.M, Assistant Professor, IESCE

28

❖ Comparison



- ✓ If $|w| = n$, then states required in minimal DFA is $n+2$
- ✓ If $|w| \geq n$, then states required in minimal DFA is $n+1$
- ✓ If $|w| \leq n$, then states required in minimal DFA is $n+2$
- ✓ **There can be many possible DFAs for a pattern.** A DFA with minimum number of states is generally preferred.

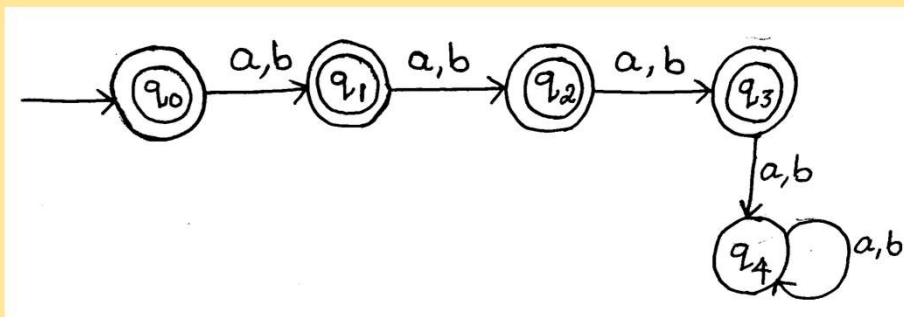
Prepared by EBIN P.M , Assistant Professor , IESCE

29

Q: Construct a DFA that accepts set of all strings w over $\{a, b\}$ where $|w| \leq 3$ (at most 3)

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

$L = \{\epsilon, a, b, aa, ab, ba, bb, aaa, abb, aab, \dots\}$

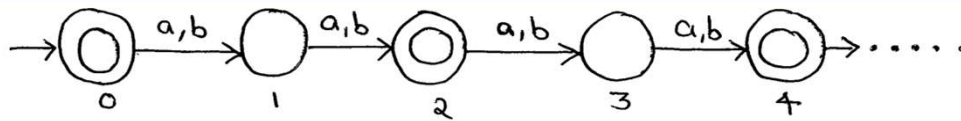


Prepared by EBIN P.M , Assistant Professor , IESCE

30

Q: Construct a minimal DFA which accepts set of all strings w over $\{a,b\}$ such that $|w| \bmod 2 = 0$. (ie, even length strings)

$L = \{\epsilon, aa, ab, ba, bb, aaaa, \dots, bbbb, \dots\}$. The Language is infinite.



(Fig:1)

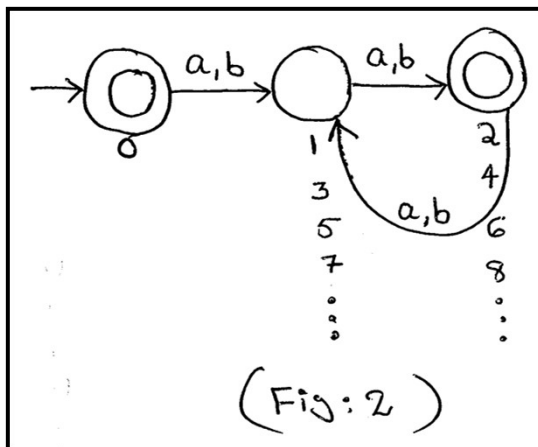
- If we are using the above method (Fig : 1), The FA will never end. But we can create a finite automata by decreasing the number of states. For that, we can distinguish with two classes.

1. odd length strings
2. Even length strings

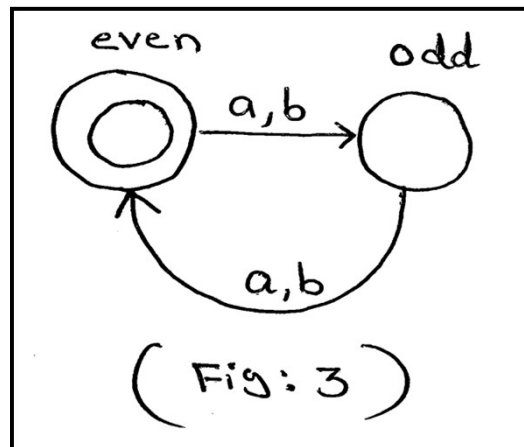
Prepared by EBIN P.M , Assistant Professor , IESCE

31

In the left side DFA, we reduce the number of states. But it is not minimal DFA. The DFA shown on right side is the minimal DFA. Here one state accept even length strings and other for odd length string



(Fig:2)

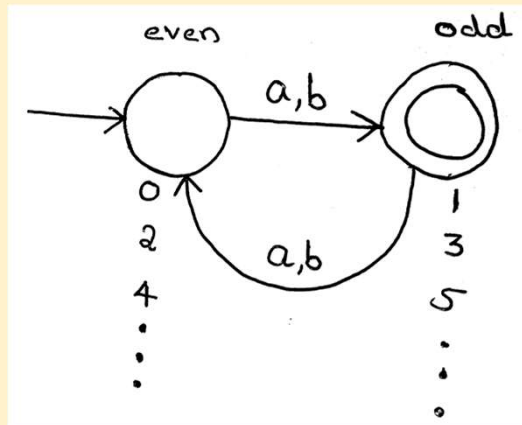


(Fig:3)

Prepared by EBIN P.M , Assistant Professor , IESCE

32

➤ Based on the previous question, we can create a DFA which accept odd length strings. ($|w| \bmod 2 = 1$)

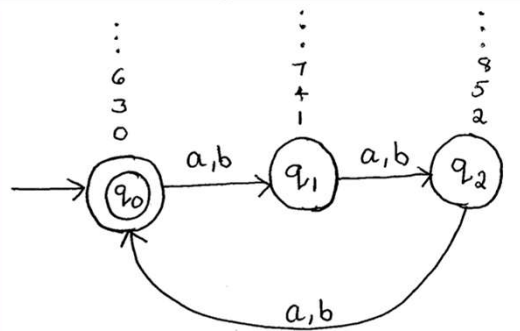


Prepared by EBIN P.M , Assistant Professor , IESCE

33

Q: Construct a minimal DFA which accepts set of all strings w over $\{a,b\}$ such that $|w| \bmod 3 = 0$. (ie, length of string is divisible by 3)
 $L = \{\epsilon, aaa, bab, \dots bbb, \dots aaaaaa, \dots bbbbbb, \dots\}$ infinite.

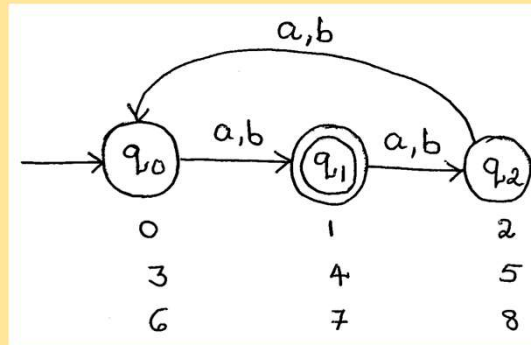
- If we divide a number using 3, the possible remainders are 0, 1 or 2. So, to distinguish the remainders, we need 3 states.



Prepared by EBIN P.M , Assistant Professor , IESCE

34

Q: Construct a minimal DFA which accepts set of all strings w over $\{a,b\}$ such that $|w| \bmod 3 = 1$. ($|w| \equiv 1 \pmod 3$)



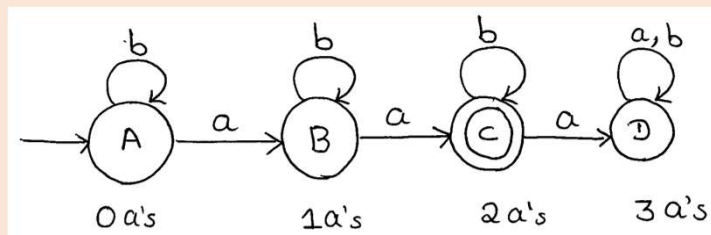
- Let, the length of string $|w| \bmod n = 0$, then minimal DFA contains n number of state.

Prepared by EBIN P.M , Assistant Professor , IESCE

35

Q: Construct a minimal DFA which accepts set of all strings w over $\{a,b\}$ such that $n_a(w) = 2$. (number of 'a' in a string should be two)

$L = \{aa, baa, aba, aab, bbaa, \dots\}$. It is infinite

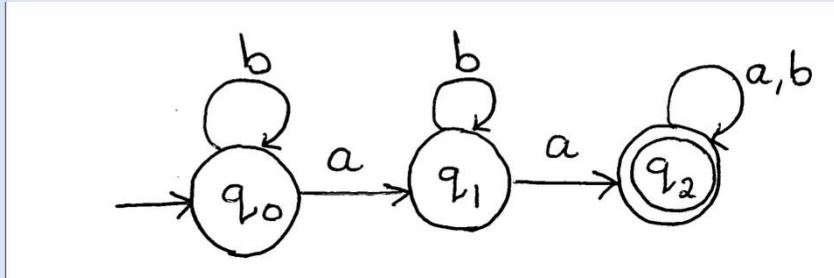


- Here, 'b' should never contribute the counting because we only consider the number of a's.
- There is no transition while seeing 'b' and we kept out of counting.

Prepared by EBIN P.M , Assistant Professor , IESCE

36

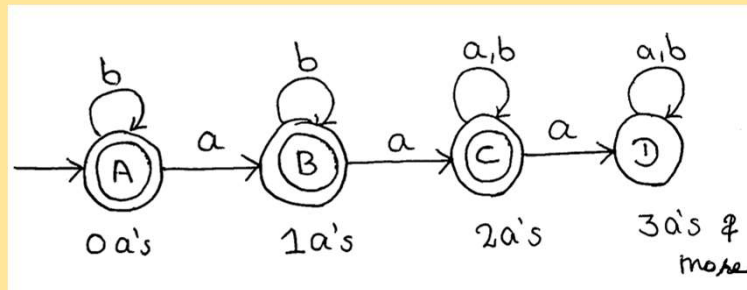
Q: Construct a minimal DFA which accepts set of all strings w over $\{a,b\}$ such that $n_a(w) \geq 2$. (number of 'a' in a string should be ≥ 2)
 $L = \{aa, baaa, aaaab, abab, \dots\}$. It is infinite



Prepared by EBIN P.M , Assistant Professor , IESCE

37

Q: Construct a minimal DFA which accepts set of all strings w over $\{a,b\}$ such that $n_a(w) \leq 2$. (number of 'a' in a string should be ≤ 2)
 $L = \{\epsilon, a, ab, ba, aa, baa, aba, \dots\}$. It is infinite

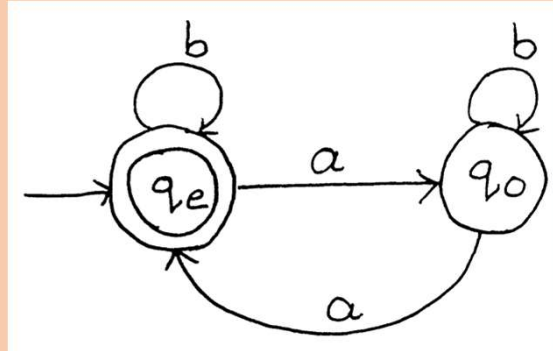


Prepared by EBIN P.M , Assistant Professor , IESCE

38

Q: Construct a minimal DFA which accepts set of all strings w over $\{a,b\}$ such that $n_a(w) \bmod 2 = 0$

$L = \{\epsilon, aa, baa, aba, aaaa, abab, \dots\}$. It is infinite



Prepared by EBIN P.M , Assistant Professor , IESCE

39

Q: Construct a minimal DFA which accepts set of all strings w over $\{a,b\}$ such that $n_a(w) \bmod 2 = 0$ and $n_b(w) \bmod 2 = 0$ (That is, number of a's and number of b's should be even)

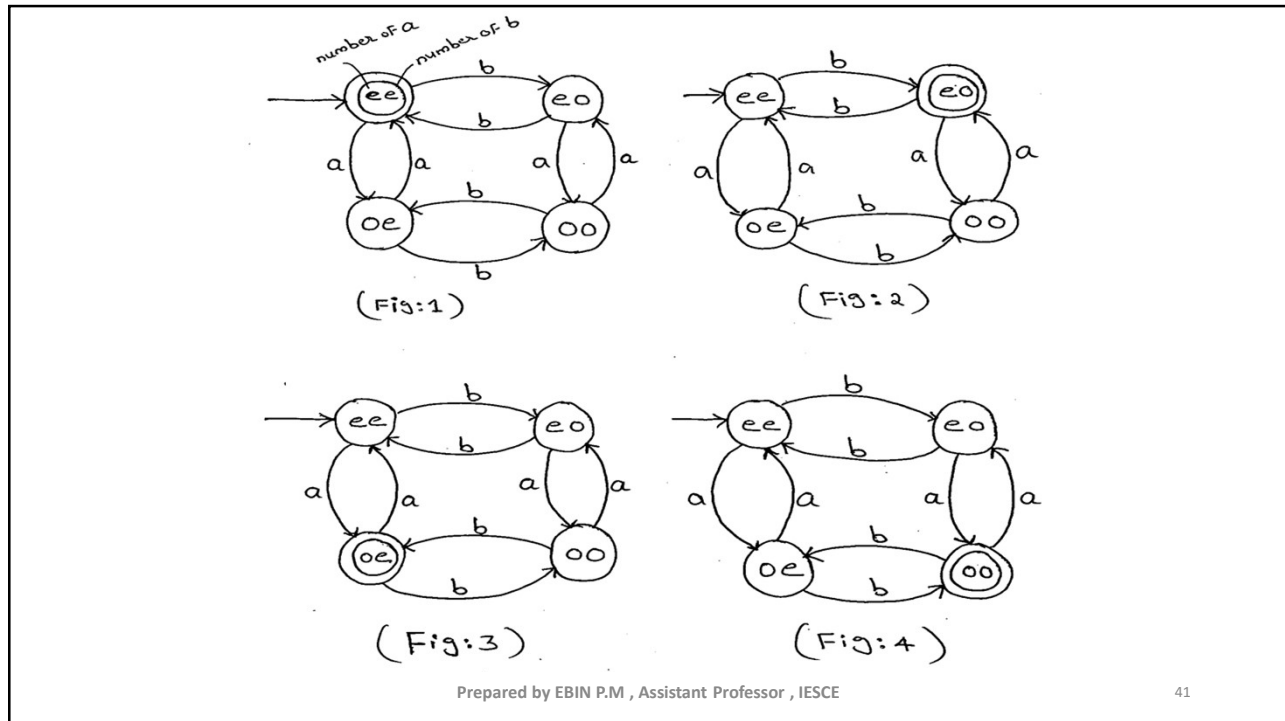
- All the even number of strings need not contains, even number of a's and even number of b's.

$L = \{\epsilon, aa, bb, aabb, abab, bbbb, \dots\}$

$n_a(w)$	$n_b(w)$	Example
Even	Even	ϵ, aa, bb
Even	Odd	aab
Odd	Even	$aaabb$
Odd	Odd	ab

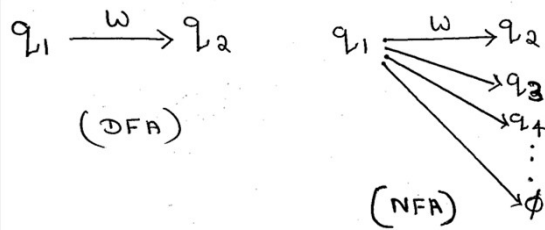
Prepared by EBIN P.M , Assistant Professor , IESCE

40



Nondeterministic Finite Automata (NFA)

- A Finite Automata (FA) is said to be non deterministic, if there is more than one possible transition from one state on the same input symbol.
- If we are starting from some state (say q_1), on seeing the input w , it will go to many state include nothing state (ϕ).



- Nondeterministic machines are not real. (no real machines exist)
- If we want to do exhaustive search and backtracking, Non-determinism can be used.
- If we don't want to do any backtracking, determinism is used.
- NFA has a **different transition function**, rest is same as DFA.
- NFA is a collection of 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where:

Q : finite set of states

Σ : finite set of the input alphabet

q_0 : initial state (start state)

F : Set of final states

δ : Transition function

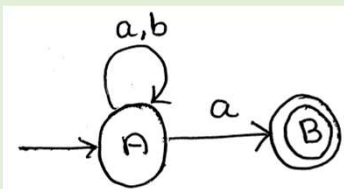
Prepared by EBIN P.M , Assistant Professor , IESCE

43

- Consider the example

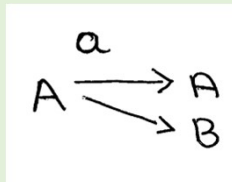
Construct NFA which accept set of all strings over $\Sigma=\{a,b\}$ in which every string ends with 'a'.

Here language $L=\{a, aa, ba, \dots\}$



Lets take the string 'a'

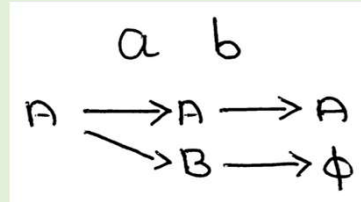
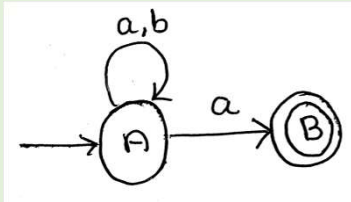
On seeing 'a', the state A go to state A or state B. This is non-determinism. If at least one state will be final, then we can say that the string is accepted.



Prepared by EBIN P.M , Assistant Professor , IESCE

44

- Lets take another string 'ab', which is not in our language.



- Here , A and ϕ are not final states. So the string 'ab' is rejected.
- Now we can define the transition function of NFA

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

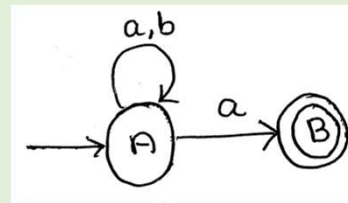
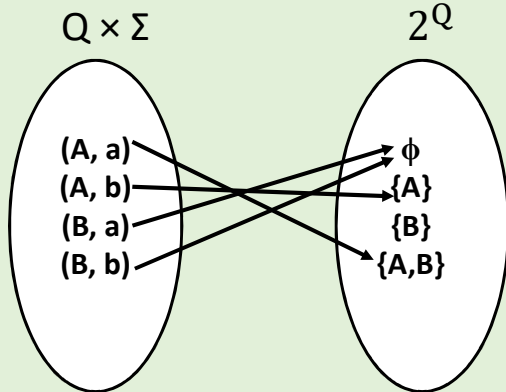
- 2^Q is the power set of Q . That is, all possible subset of Q. Lets check how it works.

Prepared by EBIN P.M , Assistant Professor , IESCE

45

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

$Q = \{A, B\}$ $\Sigma = \{a, b\}$



Prepared by EBIN P.M , Assistant Professor , IESCE

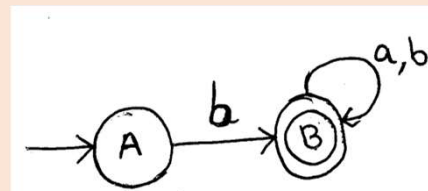
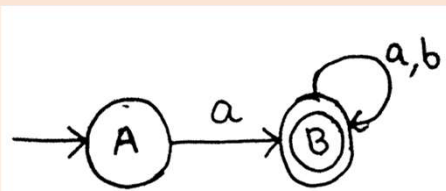
46

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

DFA : $\delta : Q \times \Sigma \rightarrow Q$, here Q is a part of 2^Q . So we can say that

Every DFA is an NFA, but the reverse is not true.

Q: Construct NFA which accepts set of all strings over $\Sigma=\{a,b\}$ which (1) starts with 'a' (2) starts with 'b'

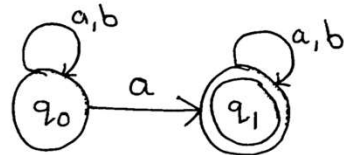


Prepared by EBIN P.M, Assistant Professor, IESCE

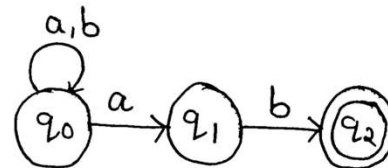
47

Q: construct NFA which accepts set of all strings over $\Sigma=\{a,b\}$

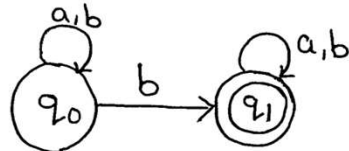
Which contains 'a'



ends with ab



Which contains b



Prepared by EBIN P.M, Assistant Professor, IESCE

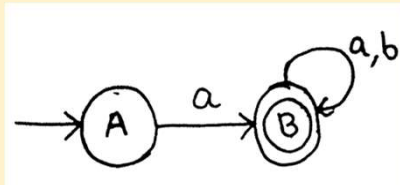
48

DFA	NFA
DFA stands for Deterministic Finite Automata	NFA stands for Nondeterministic Finite Automata.
For each symbolic representation of the alphabet, there is only one state transition in DFA	No need to specify how does the NFA react according to some symbol.
DFA cannot use Empty String transition.	NFA can use Empty String transition.
DFA can be understood as one machine.	NFA can be understood as multiple little machines computing at the same time.
In DFA, the next possible state is distinctly set.	In NFA, each pair of state and input symbol can have many possible next states.
DFA is more difficult to construct.	NFA is easier to construct.
DFA rejects the string in case it terminates in a state that is different from the accepting state.	NFA rejects the string in the event of all branches dying or refusing the string.
Time needed for executing an input string is less.	Time needed for executing an input string is more.
All DFA are NFA.	Not all NFA are DFA.
DFA requires more space.	NFA requires less space than DFA.

Prepared by EBIN P.M , Assistant Professor , IESCE 49

CONVERSION OF NFA TO DFA

Q : Construct NFA which accepts set of all strings over $\Sigma=\{a,b\}$ which starts with 'a' and convert to DFA

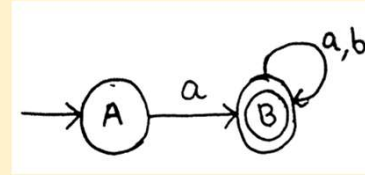


- The above diagram is called state transition diagram.
- To convert NFA to DFA many methods are possible. Here we use **Subset Construction Method**.

STEP1: Draw state transition table for the NFA

	a	b
A	B	ϕ
B	B	B

NFA

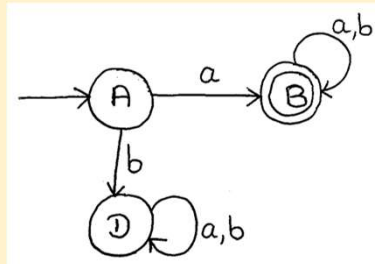

STEP2: Draw state transition table for DFA from the above table

	a	b
A	B	D
B	B	B
D	D	D

Here D indicates dead state

Prepared by EBIN P.M , Assistant Professor , IESCE

51

STEP3 : Draw state transition diagram for DFA


➤ Here, for every state we shows what happens on a or b.

Prepared by EBIN P.M , Assistant Professor , IESCE

52

Convert to DFA

State transition table for the above NFA

	a	b
A	{A,B}	{A}
B	{B}	{C}
C	{C}	{C}

Corresponding state transition table for DFA

	a	b
→ A	{A,B}	{A}
AB	{A,B}	{A,C}
AC	{A,B,C}	{A,C}
ABC	{A,B,C}	{A,C}

Prepared by EBIN P.M , Assistant Professor , IESCE 53

Q: Find the equivalent DFA for the following NFA?

Prepared by EBIN P.M , Assistant Professor , IESCE 54

State transition table for the above NFA Corresponding state transition table for DFA

	a	b
A	{A,B}	{A}
B	{c}	{c}
C	{ }	{ }

	a	b
A	{A,B}	{A}
AB	{A,B,c}	{A,c}
ABC	{A,B,c}	{A,c}
AC	{A,B}	{A}

Prepared by EBIN P.M , Assistant Professor , IESCE 55

DFA

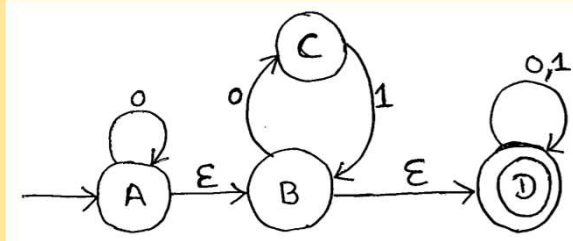
```

    graph LR
      Start(( )) --> A((A))
      A -- b --> A
      A -- a --> AB((AB))
      AB -- a --> ABC(((ABC)))
      ABC -- b --> AC(((AC)))
      AC -- b --> A
      AC -- a --> AB
      AC -- b --> AC
      ABC -- a --> ABC
  
```

Prepared by EBIN P.M , Assistant Professor , IESCE 56

ϵ - NFA

- ϵ is an empty string.



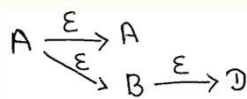
- It can be defined as $(Q, \Sigma, \delta, q_0, F)$
- $\delta : Q \times \Sigma \cup \{\epsilon\} \rightarrow 2^Q$
- From a particular state, on seeing an input symbol or without seeing something (ϵ), a transition occurs. It is also called ϵ -closure

Prepared by EBIN P.M , Assistant Professor , IESCE

57

❖ ϵ -closure (ϵ^*)

- ϵ -closure(A)

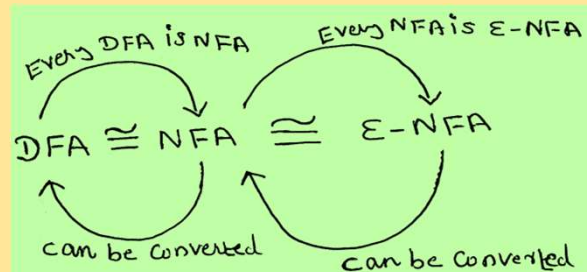
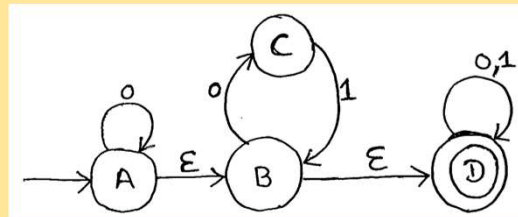


$$\epsilon\text{-closure}(A) = \{A, B, D\}$$

$$\epsilon\text{-closure}(B) = \{B, D\}$$

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

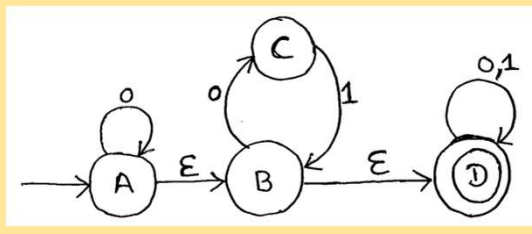
$$\epsilon\text{-closure}(D) = \{D\}$$



Prepared by EBIN P.M , Assistant Professor , IESCE

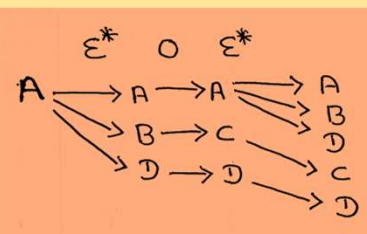
58

❖ Conversion of ϵ -NFA to NFA

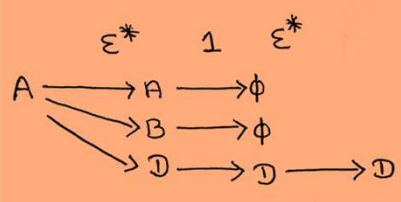


	0	1
A	?	?
B	?	?
C	?	?
D	?	?

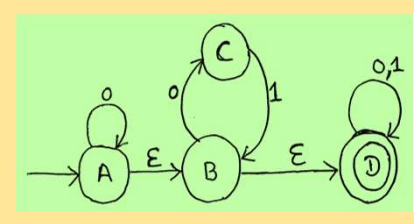
ϵ -closure($\delta(\epsilon$ -closure(A), 0))



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



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



$\epsilon^* \quad 0 \quad \epsilon^*$

$B \xrightarrow{\epsilon^*} B \xrightarrow{0} C \xrightarrow{\epsilon^*} C$
 $\quad \quad \quad \searrow \quad \quad \quad \rightarrow D \xrightarrow{\epsilon^*} D \xrightarrow{0} D \xrightarrow{\epsilon^*} D$

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

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

Prepared by EBIN P.M , Assistant Professor , IESCE 61

$\epsilon^* \quad 0 \quad \epsilon^*$

$C \xrightarrow{\epsilon^*} C \xrightarrow{0} \phi \xrightarrow{\epsilon^*} \phi$

$\epsilon^* \quad 1 \quad \epsilon^*$

$C \xrightarrow{\epsilon^*} C \xrightarrow{1} B \xrightarrow{\epsilon^*} B$
 $\quad \quad \quad \searrow \quad \quad \quad \rightarrow D$

$\epsilon^* \quad 0 \quad \epsilon^*$

$D \xrightarrow{\epsilon^*} D \xrightarrow{0} D \xrightarrow{\epsilon^*} D$

$\epsilon^* \quad 1 \quad \epsilon^*$

$D \xrightarrow{\epsilon^*} D \xrightarrow{1} D \xrightarrow{\epsilon^*} D$

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

Prepared by EBIN P.M , Assistant Professor , IESCE 62

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

Some state can reach to final state only on seeing ϵ , that state should be a final state.

Prepared by EBIN P.M , Assistant Professor , IESCE

63