Automata Theory and Introduction to compilers

Subject code : 21CST501
No. of Credits : 3 → L T P 2 : 1 : 0
No. of lecture hours/week : 4
Exam Duration : 3 hours.
Total no. of Contact hours : 52
CIE + Assignment + SEE = 45 + 5 + 50 = 100.

Text Book :

1. John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman : Introduction to Automata Theory, Languages and computation, Publisher : Pearson Education ; Third Edition (2011).
(Chapters : 1.1, 1.5, 2.2 to 2.5, 3.1 to 3.3, 4, 5, 6, 7, 8.1 to 8.4, 8.6).
ISBN-10 : 8131762688 & ISBN-13 : 978-8131762684.

2. Alfred W Aho, Monica S Lam, Ravi Sethi, and Jeffrey D ~~Publisher~~ Ullman, "Compilers - Principles, Technique and Tools" Publisher : Pearson Education ; Second Edition (1 January 2011)
ISBN-10 : 8131759024 ISBN-13 : 978-8131759028.

Reference Books :

1. K.L.P. Mishra : Theory of computer Science, Automata, Languages and Computation, 3rd Edition, PHI, 2007. ISBN - 978-81-203-2968-3.
2. Raymond Greenlaw, H. James Hoover : Fundamentals of the theory of computation, Principles and Practice, Morgan Kaufmann, 1998. ISBN 9781558605473, 9780080948355.
3. John C Martin : Introduction to Languages

PAGE NO :
DATE :

and Automata Theory, 3rd Edition, Tata McGraw-Hill, 2007. ISBN 10: 0070660484 / ISBN 13: 9780070660489.

4. Kavi Mahesh: Theory of Computation, A problem-solving approach, Wiley-India. ISBN: 9788126533114.

PAGE NO:
DATE:

Module - 1. (9 hours)

Introduction to Finite Automata

1.1.1 → * Introduction to Finite Automata.
1.5 → The central concepts of Automata theory.
2.2 → * Deterministic finite Automata
2.2.5 → * The language of a DFA.
2.2.4 → Extending the transition Function to strings
2.3, 2.3.7 → * Non deterministic finite automata and its conversion
2.3.3 → The extended Transition function
2.3.5 → Equivalence of Deterministic and Non deterministic finite automata.
2.5 → Finite automata with Epsilon-transitions
2.5.4 → Extended Transitions and Languages for NFAs
2.5.5 → Eliminating Epsilon-Transitions.

1.1 Why study Automata Theory?

Several reasons to study Automata and complexity as it is an important part of the core of Computer science.

1.1.1 Introduction to Finite Automata

Finite automata are a useful model for many important kinds of hardware and software.

- list of the most important kinds:

1. Software for designing and checking the behaviour of digital circuits.

2. The "lexical analyzer" of a typical compiler,

PAGE NO :
DATE :

i.e, the compiler component that breaks the input text into logical units, such as identifiers, keywords, and punctuation.

3. Software for scanning large bodies of text, such as collection of web pages, to find occurrences of words, phrases, or other patterns.

4. Software for verifying systems of all types that have a finite number of distinct states, such as communication protocols or protocols for secure exchange of information.

NOTE:- The advantage of having only a finite number of states is that we can implement the system with a fixed set of resources.

- For example, we could implement it in hardware as a circuit, or as simple form of program that can make decisions looking only at a limited amount of data or using the position in the code itself to make the decision.

Example 1.1 :-

The simplest nontrivial finite automaton is an on/off switch.

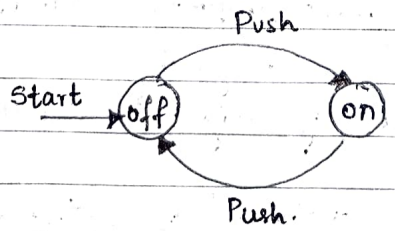


figure: A Finite automaton modeling an on/off switch

PAGE NO:
DATE:

NOTE:- Device reminds whether it is "on" state or the "off" state and allows the user to press a button, depending on the state of the switch.

For all Finite automata,

→ States are represented by circles. (i.e, states on and off)

→ Arcs between states are labeled by "inputs". (i.e Push), which represents a user pushing the button).

→ "Start state", the state in which the system is placed initially.
(i.e, In the above example, the start state is off, & by conventionally indicate the start state by the word "Start" and an arrow leading to that state.

→ "final" or "accepting" state (i.e, device being controlled by the switch will operate.)

→ It is conventional to designate accepting states by a double circle, although we have not made any such designation in above figure).

Example 1.2:-

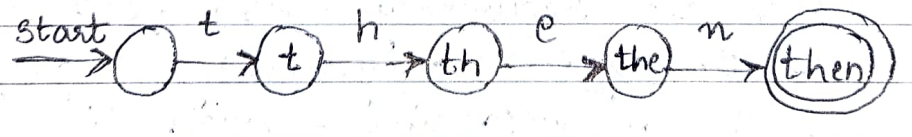


figure. A finite automaton modeling recognition of then.

NOTE:- figure shows another finite automaton that could be part of a lexical analyzer. job of this automaton is to recognize

PAGE NO:
DATE:

the keyword then.

- Five states are named by prefix of then.
- Inputs correspond to letters.
- Start state corresponds to empty string
- Each state has a transition on the next letter of then to the state that corresponds to the next-larger prefix.
- Job of this automaton to recognize when then has been seen, then consider that state the lone accepting state.

1.1.2 Structural Representations

- 2 important notations that are not automaton-like, but play important role in the study of automata and their applications.

1. Grammars - are useful models when designing software that processes data with a recursive structure.
   Ex:- parser.

- the component of a compiler that deals with the recursively nested features of the typical programming language, such as expressions - arithmetic, conditional, and so on.
- For instance, a grammatical rule like $E \Rightarrow E+E$ states that an expression can be formed by taking any two expressions and connecting them by

PAGE NO:
DATE:

a plus sign.

- This rule is typical of how expressions of real programming languages are formed.

2. Regular Expressions - also denote the structure of data, especially text strings.

- UNIX-style regular expression
'[A-Z][a-z]* [ ][A-Z][A-Z]'
represents capitalized words followed by a space and two capital letters.

- This expression ~~capitalized~~ represents patterns in text that could be Ex:- Ithaca NY.

- Example of complex expression '[A-Z][a-z]* ([ ][A-Z][a-z]*)* [ ][A-Z][A-Z]'

→ when interpreting such expressions, [A-Z] represents - a range of characters from capital "A" to capital "Z" (i.e any capital letters).,
and [ ] - is used to represents the blank character alone.
and symbol * - represents "any number of" the preceding expression.
and parentheses - are used to group components of the expression, they donot represent characters of the text described.

1.1.3. Automata and complexity.

Automata are essential for the study of the limits of computation.

PAGE NO:
DATE:

- There are 2 major important issues:

1. What can a computer do at all? This study is called "decidability", and the problems that can be solved by computer are called "decidable".

2. What can a computer do efficiently? This study is called "intractability", and the problems that can be sloved by a computer using no more time than some slowly growing function of the size of the input are called "tractable".

→ polynomial functions to be "slowly growing", while functions that grow faster than any polynomial are deemed to grow too fast.

1.5 The central Concepts of Automata Theory.

- The most important definitions of terms that pervade the theory of automata are explained below:

1. "alphabet" - a set of symbols.
2. "Strings" - a list of symbols from an alphabet.
3. "Language" - a set of strings from the same alphabet.

1.5.1 Alphabets

An alphabet is a finite, non-empty set of symbols.

PAGE NO:
DATE:

Conventionally, we use the symbol Σ for an alphabet.

Common alphabets include:

1. $\Sigma = \{0, 1\}$, the binary alphabet.

2. $\Sigma = \{a, b, \ldots, z\}$, the set of all lower-case letters.

3. The set of all ASCII characters, or the set of all printable ASCII characters.
(i.e, no. from 0-9, upper & lower case letters from A to Z & some special characters)
(i.e, 7-bit character set containing 128 characters)

1.5.2 Strings.

A string (or sometimes word) is a finite sequence of symbols chosen from some alphabet.

For ex:- 01101 is a string from the binary alphabet $\Sigma = \{0, 1\}$.

The string 111 is another string choosen from this alphabet.

→ The Empty string :- is the string with zero occurrences of symbols.

This string denoted ϵ, is a string that may be choosen from any alphabet whatsoever.

→ Length of a string:

The standard notation for the length of a string w is $|w|$.

For example, $|011| = 3$ and $|\epsilon| = 0$.

$|01101| = 5$.

- length of a string is the "number of symbols"

PAGE NO :
DATE :

in the string.
(i.e, there are only two symbols 0 & 1, in the string 01101, but there are five positions for symbols, and its length is 5).

→ Powers of an Alphabet:
If Σ is an alphabet, the set of all strings of a certain length from that alphabet by using an exponential notation,
define Σ^k to be the set of strings of length k, each of whose symbols is in Σ.

Example 1:- Note that $\Sigma^0 = \epsilon$, no matter what the alphabet Σ is.
In other words, ϵ is the only string of length 0.

→ if $\Sigma = \{a, b, c\}$,
then $\Sigma^1 = \{a, b, c\}$,
$\Sigma^2 = \{aa, ab, ac, ba, bb, bc, ca, cb, cc\}$,
$\Sigma^3 = \{aaa, aab, aac, aba, abb, abc, aca, acb, acc, baa, bab, bac, bba, bbb, bbc, bca, bcb, bcc, caa, cab, cac, cba, cbb, cbc, cca, ccb, ccc\}$.

NOTE:- Type Convention for symbols and strings.

lower-case letters at the beginning of the alphabet (or digits) to denote symbols, and lower-case letters

near the end of the alphabet, typically w, x, y, and z, to denote strings.

Ex:- The set of all strings over an alphabet Σ is conventionally denoted Σ^*.

i.e, $\{0,1\}^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}$

(or)

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

→ The set of non empty strings from alphabet Σ is denoted Σ^+.

- Two appropriate equivalences are:

1. $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots$
2. $\Sigma^* = \Sigma^+ \cup \{\epsilon\}$.

→ Concatenation of strings:

Let x and y be strings. Then xy denotes the concatenation of x and y.

i.e, if x is the string composed of i symbols
$x = a_1 a_2 \ldots a_i$ and
y is the string composed of j symbols
$y = b_1 b_2 \ldots b_j$,
then xy is the string of length $i+j$.
i.e, $xy = a_1 a_2 \ldots a_i b_1 b_2 \ldots b_j$.

Example 13:- Let $x = 1101$ and $y = 0011$.
Then $xy = 11010011$. and
$yx = 00111101$.

PAGE NO:
DATE:

NOTE:- For any string w, the equations $\epsilon w = w\epsilon = w$.
i.e ϵ is the identity for concatenation

1.5.3 Languages.

A set of strings all of which are chosen from some Σ^*, where Σ is a particular alphabet, is called a language.

If Σ is an alphabet, and $L \subseteq \Sigma^*$, then L is a language over Σ.

NOTE:- Alphabet is a subset of the ASCII characters. The exact alphabet may differ slightly among different programming languages, by generally includes the upper-case & lower-case letters, the digits, punctuation, & mathematical symbols.

Some are abstract examples, such as:

1. The language of all strings consisting of n 0's followed by n 1's, for some $n \geq 0$: $\{\epsilon, 01, 0011, 000111, \ldots\}$.

2. The set of strings of 0's and 1's with an equal number of each:
$\{\epsilon, 01, 10, 0011, 0101, 1001, \ldots\}$

3. The set of binary numbers whose value is a prime:

PAGE NO :
DATE :

$\{10, 11, 101, 111, 1011, \ldots\}$

4. Σ^* is a language for any alphabet Σ.

5. ϕ, the empty language, is a language over any alphabet.

6. $\{\epsilon\}$, the language consisting of only the empty string, is also a language over any alphabet.
$\phi \neq \{\epsilon\}$.
The former has no strings and the latter has one string.

1.5.4 Problems

In automata theory, a problem is the question of deciding whether a given string is a member of some particular language.

- if Σ is an alphabet, and L is a language over Σ, then the problem L is:

"Given a string w in Σ^*, decide whether or not w is in L."

2.2 Deterministic Finite Automata

- The term "deterministic" refers to the fact that on each input there is one and only one state to which the automaton can transition from its current state.
- The term "finite automaton" will refer to the deterministic variety, although we shall use

PAGE NO :
DATE :

"deterministic" or the abbreviation DFA normally.

2.2.1 Definition of a Deterministic Finite Automaton

A deterministic finite automaton consists of:

1. A finite set of states, denoted Q.
2. A finite set of input symbols, denoted Σ.
3. A transition function that takes as arguments a state and an input symbol and returns a state, denoted δ.
   - δ was represented by arcs between states and the labels on the arcs.
   - If q is a state, and a is an input symbol, then $\delta(q, a)$ is that state p such that there is an arc labeled a from q to p.
4. A start state, one of the states in Q.
5. A set of final or accepting states F. The set F is a subset of Q.

The most succinct representation of a DFA is a listing of the 5 components above.

- DFA in "five-tuple" notation:

$$A = (Q, \Sigma, \delta, q_0, F).$$

PAGE NO :
DATE :

where

A - is the name of the DFA.
Q - is its set of states.
Σ - its input symbols.
δ - its transition function.
q_0 - its Start state.
F - its set of accepting states.

2.2.2 How a DFA Processes Strings

The "language" of the DFA is the set of all strings that the DFA accepts.

Suppose $a_1 a_2 \ldots a_n$ is a sequence of input symbols.

q_0 Start state in DFA,
transition function δ, say $\delta(q_0, a_1) = q_1$
to find the state that the DFA A enters after processing the first input symbol a_1.

process the next input symbol, a_2 by evaluating $\delta(q_1, a_2)$, let suppose this state is q_2.
and III^ly finding states $q_3, q_4, \ldots q_n$ such that $\delta(q_{i-1}, a_i) = q_i$ for each i.

- If q_n is a member of F, then the input $a_1 a_2, \ldots a_n$ is "accepted", and if not then it is "rejected".

Example 1.4 :- [For Solution Refer (*)]

Let us formally specify a DFA that accepts all and only the strings of 0's and 1's that have the sequence 01 somewhere in the string.

PAGE NO:
DATE:

We can write this Language L as:

{ w | w is of the form x01y for some strings x and y consisting of 0's and 1's only }.

(or)

L as: { x01y | x and y are any strings of 0's and 1's }.

NOTE:- Example of strings in the language include 01, 11010, and 1000 1.1.

- Examples of strings not in the language include ϵ, 0, and 111000.
- Input alphabet is $\Sigma = \{0, 1\}$.
  Some set of states Q.
  q_0 - start state.
- To decide whether 01 is a substring of the input, A needs to remember:

1> Has it already seen 01? If so, then it accepts every sequence of further inputs (i.e, it will only be in accepting states from now on).

2> Has it never seen 01, but its most recent input was 0, so if it now sees a 1, it will have seen 01 and can accept everything it sees from here on?

3> Has it never seen 01, but its last input was either non existent (it just started) or it last saw a 1? In this case, A cannot accept

PAGE NO :
DATE :

until it first sees a 0 and then sees a 1 immediately after.

Finally, we must design the transitions for state q_1, In this state, we have already seen 01 sequence, so regardless of what happens, we shall still be in situation where we've seen 01.

That is, $\delta(q_1, 0) = \delta(q_1, 1) = q_1$.

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

q_0 is the start state, and the only accepting state is q_1; that is $F = \{q_1\}$.

The complete specification of the automaton A that accepts the Language L of strings that have a 01 substrings, is

$$A = (\{q_0, q_1, q_2\}, \{0,1\}, \delta, q_0, \{q_1\})$$

where δ is the transition function described above.

2.2.3 Simpler Notations for DFA's

Two preferred notations for describing automata:

1. A transition diagram, which is a graph
2. A transition table, which is a tabular listing of the δ function, which by implication tells us the set of states and the input alphabet.

Transition Diagrams.

A transition diagram for a DFA $A = (Q, \Sigma, \delta, q_0, F)$ is a graph defined as follows:

a) For each state in Q there is a node.

PAGE NO :
DATE :

b) For each state q in Q and each input symbol a in Σ, Let $\delta(q, a) = p$. Then the tranition diagram has an arc from node q to node p, labeled a. If there are several input symbols that cause transitions from q to p, then the transition diagram can have one arc, labeled by the list of these symbols.

c) There is an arrow into the start state q_0, labeled Start. This arrow does not originate at any node.

d) Nodes corresponding to accepting states (those in F) are marked by a double circle. states not in F have a single circle.

Example 1.4 (Solution) (*).

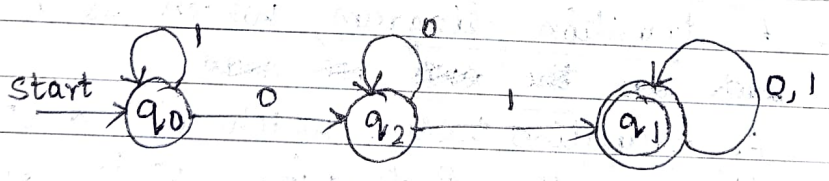


Figure. The transition diagram for the DFA accepting all strings with a substring 01.

We see in that diagram the three nodes that correspond to the three states. There is a start arrow entering the start state q_0, and the one accepting

PAGE NO :
DATE :

State q_1, is represented by a double circle. Out of each state is one arc labeled 0 and one arc labeled 1 (although the two arcs are combined into one with a double label in the case of q_1).

The arcs each correspond to one of the δ facts developed in example 1.

Transition Tables.

A Transition table is a conventional, tabular representation of a function like δ that takes two arguments and return a value.

→ Rows of the table correspond to the states,
→ Columns correspond to the inputs.
→ The entry for the row corresponding to state q and the column corresponding to input a is the state $\delta(q, a)$.

Example 1.4 (solution) *.

	0	1
→ q_0	q_2	q_0
* q_1	q_1	q_1
q_2	q_2	q_1

Figure. Transition table for the DFA of example 1.

The transition table corresponding to the function δ.

The start state is marked with an arrow, and the accepting states are marked with a star.

PAGE NO :
DATE :

2.2.4 Extending the Transition function to strings.

- Informally DFA defines a language:
  The set of all strings that result in a sequence of state transitions from the start state to an accepting state.

- In terms of transition diagram,
  The Language of a DFA is the set of labels along all the paths that lead from start state to any accepting state.

- Extended transition function that describes what happens when we start in any state and follows any sequences of inputs.

- If δ is our transition function, then the extended transition function constructed from δ will be called $\hat{\delta}$

- The extended transition function is a function that takes a state q and a string w and returns a state p - the state that the automaton reaches when starting in state q and processing the sequence of inputs w.

- Define $\hat{\delta}$ by induction on the length of the input string, as follows:

PAGE NO :
DATE :

BASIS : $\delta(q, \epsilon) = q$. i.e, if we are in state q and read no inputs, then we are still in state q.

INDUCTION : Suppose w is a string of the form xa ; i.e, a is the last symbol of w, and x is the string consisting of all but the last symbol.

For example, $w = 1101$ is broken into $x = 110$ and $a = 1$. Then.

$$\hat{\delta}(q, w) = \delta(\hat{\delta}(q, x), a) \rightarrow Eq^n (1)$$

Now, Eq^n (1) idea is simple.
To compute $\hat{\delta}(q, w)$,
First compute $\hat{\delta}(q, x)$, the state that the automaton is in after processing all but the last symbol of w.
Suppose this state is p; i.e,
$\hat{\delta}(q, x) = p$.
Then $\hat{\delta}(q, w)$ is what we get by making a transition from state p on input a, the last symbol of w.

i.e, $\hat{\delta}(q, w) = \delta(p, a)$.

Example 1.5 :-
Let us design a DFA to accept the language
$L = \{ w \mid w$ is of even length and begins with $01 \}$.

PAGE NO :
DATE :

Soln:- q_0 - The initial state.
q_1 - The state entered on reading 0 in state q_0.
q_2 - The state entered on reading 01 initially. The automaton subsequently returns to this state whenever the substring seen so far with 01 and is of even length.

q_3 - DFA enters this state whenever the substring seen so far starts with 01 and is of odd length.

q_4 - This state is entered whenever a 1 is encountered in state q_0 or a 0 is encountered in state q_1.

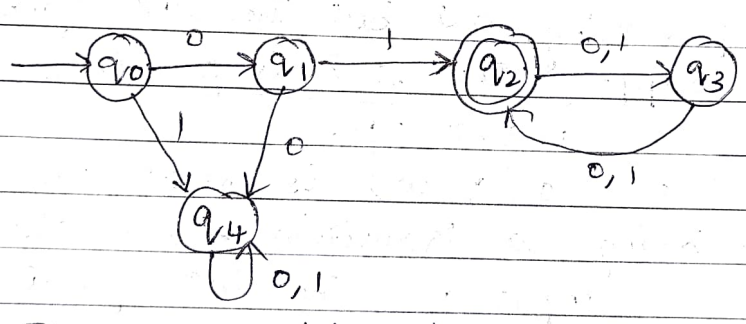


Figure. Transition diagram for the DFA.

State q_2 is the only accepting state.
∴ The DFA can thus be given as

$M = (\{q_0, q_1, q_2, q_3, q_4\}, \{0,1\}, \delta, q_0, \{q_2\}$

δ - transition function.

PAGE NO:
DATE:

Construction of $\hat{\delta}$ from its transition function δ.

Suppose the input is 011101.
Since this string starts with 01 and is of even length, expect it is in the language.
Thus, $\hat{\delta}(q_0, 011101) = q_2$,
Since q_2 is the only accepting state.

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

Figure. Transition table for the DFA.

→ The check involves computing $\hat{\delta}(q_0, w)$ for each prefix w of 011101, starting at ϵ and going in increasing size.

The summary of this calculation is:

$\hat{\delta}(q_0, \epsilon) = q_0$.

$\hat{\delta}(q_0, 0) = \delta(\hat{\delta}(q_0, \epsilon), 0) = \delta(q_0, 0) = q_1$

$\hat{\delta}(q_0, 1) = \delta(\hat{\delta}(q_0, 0), 1) = \delta(q_1, 1) = q_2$

$\hat{\delta}(q_0, 011) = \delta(\hat{\delta}(q_0, 01), 1) = \delta(q_2, 1) = q_3$

$\hat{\delta}(q_0, 0111) = \delta(\hat{\delta}(q_0, 011), 1) = \delta(q_3, 1) = q_2$.

$\hat{\delta}(q_0, 01110) = \delta(\hat{\delta}(q_0, 0111), 0) = \delta(q_2, 0) = q_3$.

$\hat{\delta}(q_0, 011101) = \delta(\hat{\delta}(q_0, 01110), 1) = \delta(q_3, 1) = q_2$

PAGE NO:
DATE:

2.2.5 The Language of a DFA

Define the language of a DFA
$A = (Q, \Sigma, \delta, q_0, F)$.
This language is denoted $L(A)$, and is defined by
$L(A) = \{ w \mid \delta(q_0, w) \text{ is in } F \}$.

i.e, the language of A is the set of strings w that take the start state q_0 to the accepting states.

If L is L(A) for some DFA A, then we say L is a regular language.

Example 1.6 :-
As mentioned earlier, if A is the DFA of example 1.4, then
L(A) is the set of all strings of 0's and 1's that contain a substring 01.
If A is instead the DFA of example 1.5, then
L(A) is the set of all strings of 0's and 1's whose numbers of 0's and 1's are both even.

2.2.6 Problems on DFA

Applications:-
1. Designing of compiler.
2. Pattern matching.
3. Spell check
4. String processing / String matching.
5. Software design.
6. Design of digital Circuit.
7. Other applications.

NOTE :-

Automata is an abstract machine or model.

Automata Theory is also known as theory of computation. (TOC).

TOC - deals with how efficiently a problem can be solved on a model of computation using an algorithm.

Computation - processing the input towards the output.

TOC - divided into 3 main branches

1. Automata Theory & language - Deals with the definitions and properties of various mathematical model of computers.

   Models - Finite automata, Context-free grammar, Turing machine.

2. Computability theory - It deals with what can and can't be computed by the model.

3. Complexity theory - It group the computable problem based on their complexity.

Definition of finite Automata :-

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

Q - is finite set of states, which is non empty.

Σ - is input alphabet, indicates input set.

δ - is transition function or mapping function. We can determine the next state using this function.

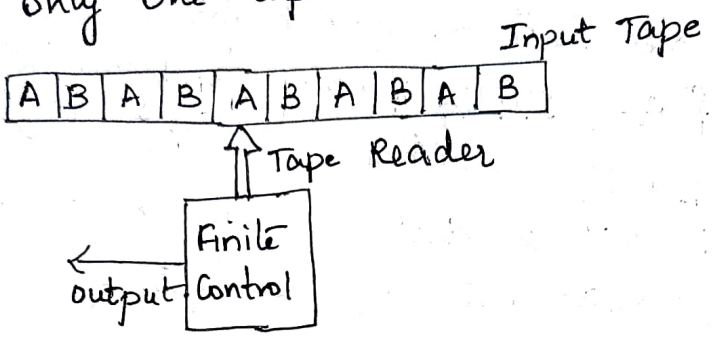
q_0 - is an initial state and $q_0 \in Q$.

F - is set of final states. $F \in Q$.

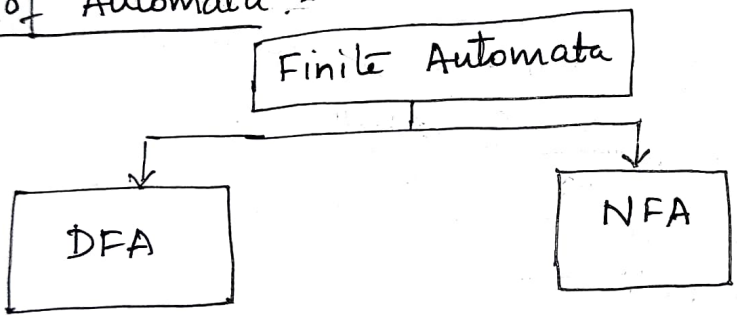
Finite Automata Model :-

It can be represented as

1. Input tape - It is a linear tape having some number of cells. Each input symbol is placed in each cell.
2. Finite Control - It decides the next state on receiving particular input from input tape.
3. Tape Reader - It reads the cell one by one, at a time only one input is read.



Types of Automata :-



NOTE :- In DFA, δ is transition function or mapping function. We can determine the next state using this function.

$$\boxed{\delta : Q \times \Sigma \longrightarrow Q}$$

In NFA, δ can determine the next state using this function.

$$\boxed{\delta : Q \times \Sigma \longrightarrow 2^{Q}}$$

Problems on DFA :-

1. Design DFA with $\Sigma = \{0, 1\}$ accepts those string which starts with 1 and ends with 0.

Soln:- $L = \{10, 100, 1010, \ldots\}$

Min. length = 2.

no. of states = |min. length| + 1
= 2 + 1 = 3.

DFA is defined as.

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

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

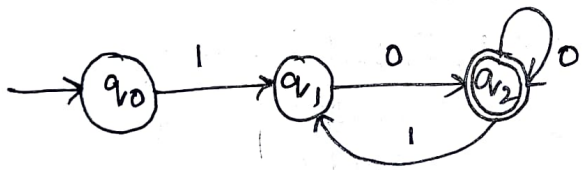
$\Sigma = \{0, 1\}$

q_0 is initial state

$F = \{q_2\}$.

δ is define as table.

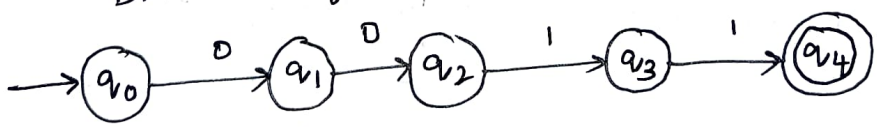
State/ Input	0	1
q_0	ϕ	q_1
q_1	q_2	ϕ
$*q_2$	q_2	q_1



transition diagram.

2. Construct DFA which accepts string 0011.
DFA is define as $M = (Q, \Sigma, \delta, q_0, F)$ where.

Soln:-



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

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

q_0 is initial state

$F = \{q_4\}$

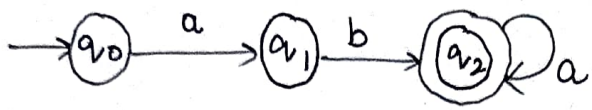
δ is define as table.

String = {0011}

State/ input	0	1
q_0	q_1	ϕ
q_1	q_2	ϕ
q_2	ϕ	q_3
q_3	ϕ	q_4
$*q_4$	ϕ	ϕ

③ Construct DFA for the string $aba^n \mid n \geq 0$.

Soln:- String = { ab, aba, abaa, abaaa, abaaaa, ... }



DFA is define as $M = (Q, \Sigma, \delta, q_0, F)$ where

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

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

q_0 is initial state

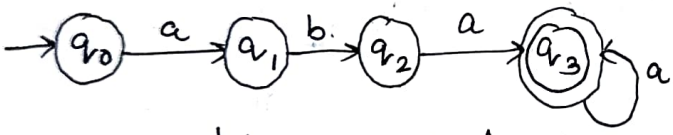
$F = \{q_2\}$

δ is define as table

State/ Input	a	b
→ q_0	q_1	ϕ
q_1	ϕ	q_2
* q_2	q_2	ϕ

④ Construct DFA for the string $aba^n \mid n \geq 1$.

Soln:- String = { aba, abaa, abaaa, abaaaa ..... }



DFA is define as $M = (Q, \Sigma, \delta, q_0, F)$ where

$Q = \{q_0, q_1, q_2, q_3\}$

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

q_0 is initial state

$F = \{q_3\}$

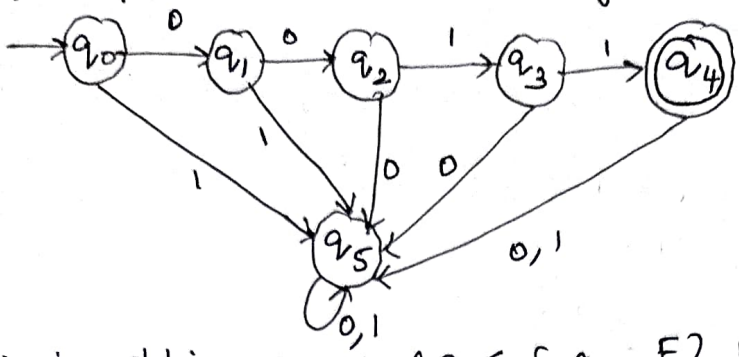
δ is define as

State/ Input	a	b
q_0	q_1	ϕ
q_1	ϕ	q_2
q_2	q_3	ϕ
q_3	q_3	ϕ

NOTE :- Dummy or Dead State :-

- Having no meaning in the generation of the string. It is only designed to full the requirements of the finite state machine that every state has one transition for each alphabet.

(5) Example of Machine with dummy state (dead state).



DFA is define as $M = (Q, \Sigma, \delta, q_0, F)$ where

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$.

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

q_0 is initial state

$F = \{q_4\}$

δ is define as table:

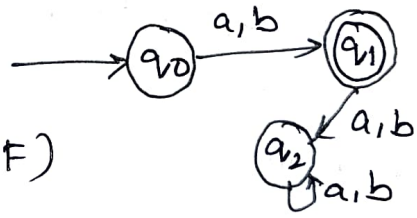
q_5 is dead state

State/ input	0	1
$\rightarrow q_0$	q_1	q_5
q_1	q_2	q_5
q_2	q_5	q_3
q_3	q_5	q_4
q_4	q_5	q_5
q_5	q_5	q_5

(6) Construct DFA which accept string either a or b.

Soln:- string $= \{a, b\}$

Language $L = \{a+b\}$.



DFA is define as $M = (Q, \Sigma, \delta, q_0, F)$

where

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

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

q_0 is initial state

$F = \{q_1\}$

δ is define as table

q_2 is dead state.

State / Input	a	b
q_0	q_1	q_1
* q_1	q_2	q_2
q_2	q_3	q_2

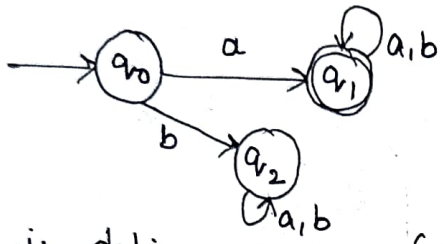
(7) Construct DFA which accept string start with a over $\{a,b\}$

Soln:- string = $\{a, aa, ab, aaa, aab, aba, abb \ldots\}$

Language $L = \{ax \mid x \in \{a,b\}^*\}$

(or) $\{a(a+b)^*\}$

(or) $\{ax \mid x \in \Sigma^*\}$



DFA is define as $M = (Q, \Sigma, \delta, q_0, F)$ where

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

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

q_0 is initial state

$F = \{q_1\}$

q_2 is dead state

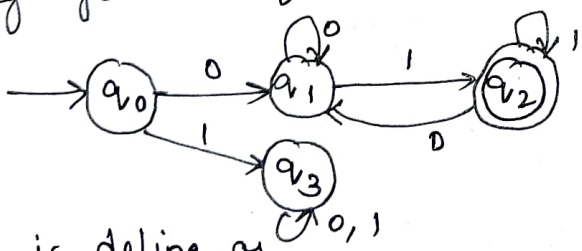
δ is define as table

State/ input	a	b
q_0	q_1	q_2
* q_1	q_1	q_1
q_2	q_2	q_2

(8) Construct DFA which accept string start with 0 and end with 1.

Soln:- string = $\{01, 001, 011, 0001, 0011, 0101, 0111 \ldots\}$

Language $L = \{0x1 \mid x \in \{0,1\}^*\}$



DFA is define as $M = (Q, \Sigma, \delta, q_0, F)$ where

$Q = \{q_0, q_1, q_2, q_3\}$

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

q_0 is initial state

$F = \{q_2\}$

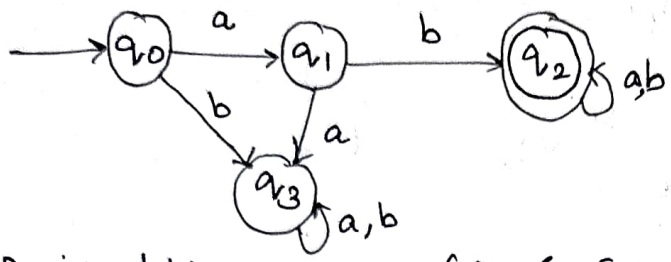
δ is define as table.

State/ Input	0	1
q_0	q_1	q_3
q_1	q_1	q_2
* q_2	q_1	q_2
q_3	q_3	q_3

(9) Construct DFA which accept string starting with ab.

Soln:- String = {ab, aba, abb, abaa, abab, abba, abbb...}

Language $L = \{ abx \mid x \in (a+b)^* \}$



DFA is define as $M = (Q, \Sigma, \delta, q_0, F)$ where

$Q = \{ q_0, q_1, q_2, q_3 \}$

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

q_0 is initial state

$F = \{ q_2 \}$

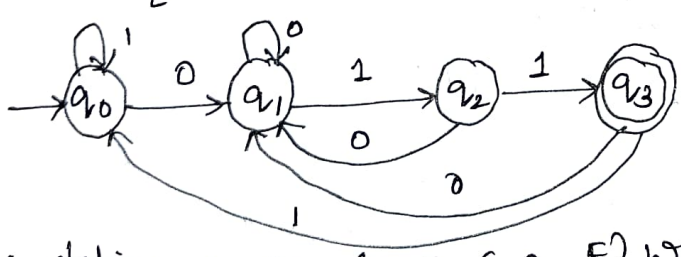
δ is define as table

State/ Input	a	b
q_0	q_1	q_3
q_1	q_3	q_2
* q_2	q_2	q_2
q_3	q_3	q_3

(10) Construct DFA which accept string ~~start~~ end with 011.

Soln:- String = {011, 0011, 1011, 00011, 01011, 10011, 11011, 011011, 0111011 ....}

Language $L = \{ x011 \mid x \in \{0,1\}^* \}$



DFA is define as $M = (Q, \Sigma, \delta, q_0, F)$ where

$Q = \{ q_0, q_1, q_2, q_3 \}$

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

q_0 is initial state

$F = \{ q_3 \}$

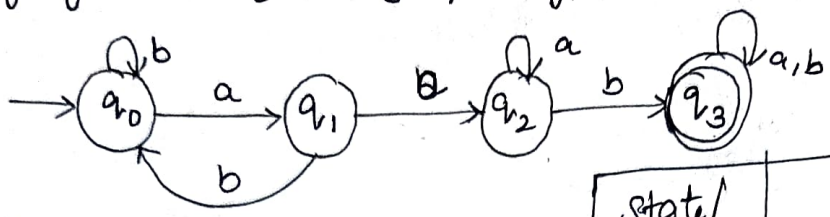
δ is defined as table

State/ Input	0	1
q_0	q_1	q_0
q_1	q_1	q_2
q_2	q_1	q_3
* q_3	q_1	q_0

(11) Construct DFA which accept string having substring aab.

Soln:- String = { aab, aaab, baab, aaba, aabb, aaabb, abaab, baabb, bbaab, abbaab ... }

Language $L = \{ xaaby \mid x, y \in \{a, b\}^* b \}$.



DFA,

$Q = \{ q_0, q_1, q_2, q_3 \}$

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

q_0 is initial state

$F = \{ q_3 \}$

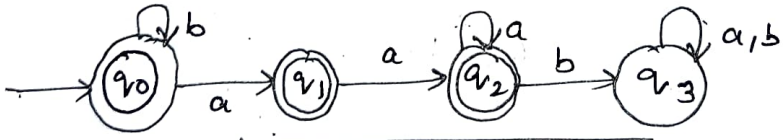
δ is define as table

State/ Input	a	b
q_0	q_1	q_0
q_1	q_1	q_0
q_2	q_2	q_3
q_3	q_3	q_3

(12) Construct DFA which accept string not having substring aab.

Soln:- TRICK :- First construct transition diagram for having substring aab., then reverse it by making final state as non-final and non-final as final. { Refer problem (11) }.

Now Reverse problem (11) DFA by making q_0, q_1, q_2 as final and q_3 as non final.

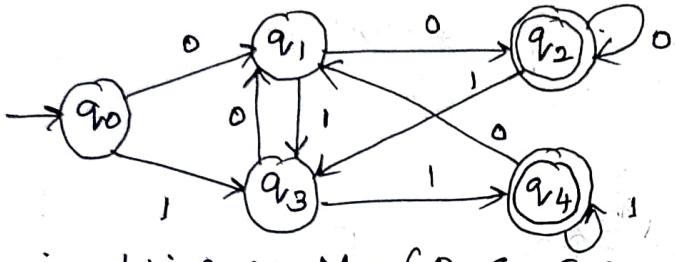


State/ Input	a	b
→ * q_0	q_1	q_0
* q_1	q_2	ϕ
* q_2	q_2	q_3
q_3	q_3	q_3

(13) Construct DFA which accept string end with either 00 or 11.

Soln:- String = {00, 11, 000, 011, 100, 111, 0000, 0100, 1000, 1100, 0011, 0111, 1111 ....}

Language $L = \{ X(00+11) \mid X \in \{0,1\}^* \}$



DFA is define as $M = (Q, \Sigma, \delta, q_0, F)$ where

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

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

q_0 - initial state.

$F = \{q_2, q_4\}$

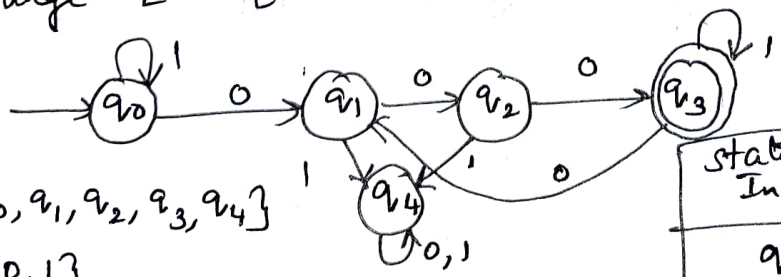
δ is define as table:

State/Input	0	1
q_0	q_1	q_3
q_1	q_2	q_3
* q_2	q_2	q_3
q_3	q_1	q_4
* q_4	q_1	q_4

(14) Construct DFA which accept string have three consecutive 0's. Over {0,1}:

Soln:- {000, 1000, 0001, 10001, 10001000, 0001000, 000000, ...}

Language $L = \{ (1+\epsilon)^* 000 (000+1)^* \}$.



DFA,

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

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

q_0 - initial state

$F = \{q_3\}$

δ is define as table.

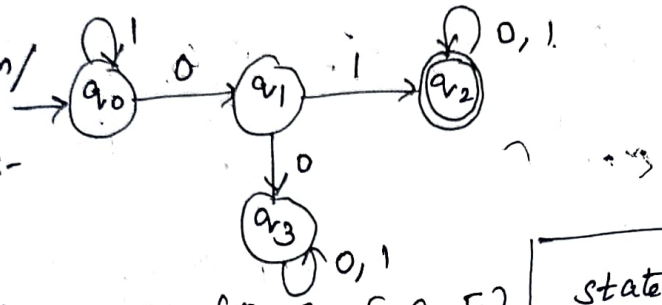
State/Input	0	1
q_0	q_1	q_0
q_1	q_2	q_4
q_2	q_3	q_4
* q_3	q_1	q_3
q_4	q_4	q_4

(15) Construct DFA which accept string of {0, 1} whose 2nd symbol from R.H.S is 1.

Soln:- String = { 10, 11, 010, 011, 110, 111, 0110, 0111, 1110, 1111.... }

Language $L = \{ (0+1)^* 1 (0+1) \}$

State diagram/ transition diagram:-



DFA is define as $M = (Q, \Sigma, \delta, q_0, F)$ where

$Q = \{ q_0, q_1, q_2, q_3 \}$

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

q_0 - initial state

$F = \{q_2\}$

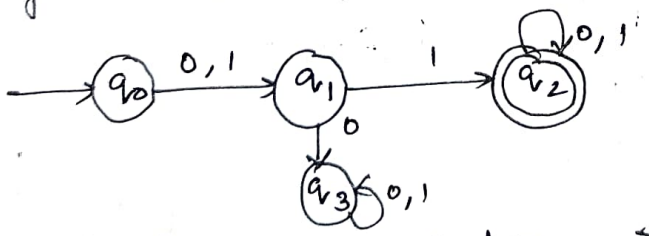
δ is define as table:

State/ Input	0	1
q_0	q_1	q_0
q_1	q_3	q_2
* q_2	q_2	q_2
q_3	q_3	q_3

(16) Construct DFA which accept string of {0, 1} whose 2nd symbol from Left is 1.

Soln:- String = { 01, 11, 010, 011, 110, 111, 0100, 0101, 0110, 0111, 1100, 1101 ... }.

Language $L = \{ (0+1) 1 (0+1)^* \}$.



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

$Q = \{ q_0, q_1, q_2, q_3 \}$

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

q_0 is initial state

$F = \{q_2\}$

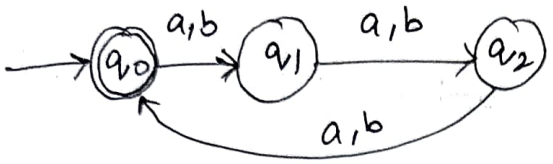
δ is define as table

State / Input	0	1
$\rightarrow q_0$	q_1	q_1
q_1	q_3	q_2
* q_2	q_2	q_2
q_3	q_3	q_3

(17) Construct DFA which accept strings of {a,b} such that length of the string is divisible by 3.

Soln:- String = { ε, aaa, aab, aba, abb, baa, bab, bba, bbb, aaaaaa, aaaaab. . . . }

Language $L = \{ ((a+b)(a+b)(a+b))^* \}$



State / Input	a	b
→ * q_0	q_1	q_1
q_1	q_2	q_2
q_2	q_0	q_0

DFA,

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

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

q_0 is initial state

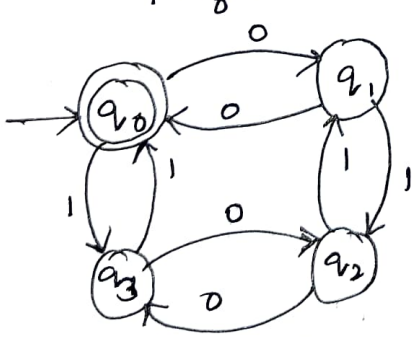
$F = \{ q_0 \}$

δ is define as table

(18) Construct DFA which accept string having even no. of 0's and even no. of 1's.

Soln:- String = { ε, 00, 11, 1010, 0101, 0110, 1001, 0000, 1111, 001100, 010100. . . }

Language $L = \{ x \mid x \in \{0,1\}^* \text{ and } |x|_0 \bmod 2 = 0 \text{ and } |x|_1 \bmod 2 = 0 \}$.



State / Input	0	1
→ * q_0	q_1	q_3
q_1	q_0	q_2
q_2	q_3	q_1
q_3	q_2	q_0

DFA,

$Q = \{ q_0, q_1, q_2, q_3 \}$

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

q_0 - initial state

$F = \{ q_0 \}$

δ is define as table:

NOTE :- q_0 final : state of even number of 0's and even number of 1's.

q_1 final : State of odd number of 0's and even number of 1's.

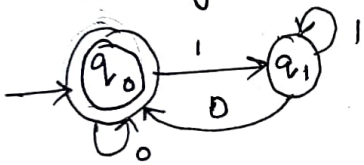
q_2 final : state of odd number of 0's and odd number of 1's.

q_3 final : state of even number of 0's and odd number of 1's.

(19) Construct DFA which accept strings binary divisible by 2.

Soln:- String in integer = {0, 2, 4, 6, 8, 10, 12, 14, 16 ...}

String in binary = {00, 10, 100, 110, 1000, 1100, 1110, 10000 ...}



State / Input	0	1
* q_0	q_0	q_1
q_1	q_0	q_1

DFA,

$Q = \{q_0, q_1\}$

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

q_0 is initial state

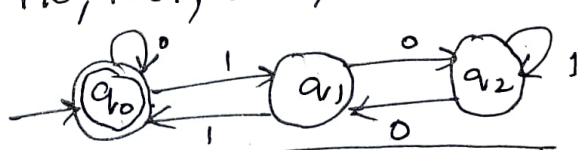
$F = \{q_0\}$

δ is define as table:

(20) Construct DFA which accept strings binary divisible by 3.

Soln:- String in integer = {0, 3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33 ...}

String in binary = {00, 11, 110, 1001, 1100, 1111, 10010, 10101, ...}



State / Input	0	1
→ * q_0	q_0	q_1
q_1	q_2	q_0
q_2	q_1	q_1

DFA,

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

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

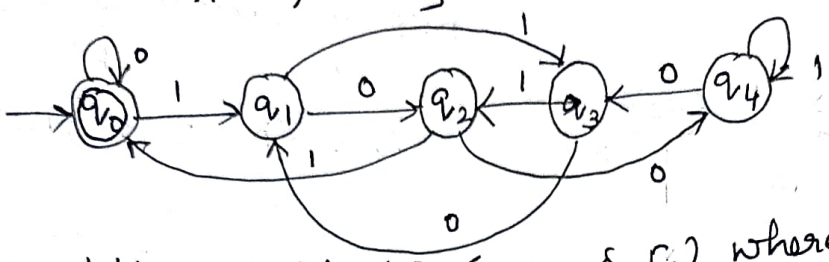
q_0 is initial state

$F = \{q_0\}$

(21) Construct DFA which accepts strings binary divisible by 5.

Soln:- String in integer = {0, 5, 10, 15, 20, 25, 30, 35, ...}

String in binary = {0, 101, 1010, 1111, 10100, 11001, 11110, ...}



DFA is define as $M = (Q, \Sigma, q_0, \delta, F)$ where

$Q = \{q_0, q_1, q_2, q_3, q_4\}$.

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

q_0 - initial state $\Rightarrow F = \{q_0\}$

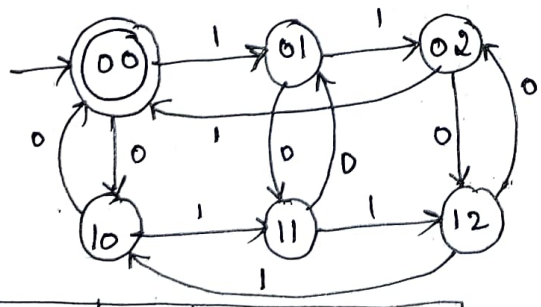
δ is define as table:

State/Input	0	1
* q_0	q_0	q_1
q_1	q_2	q_3
q_2	q_4	q_0
q_3	q_1	q_2
q_4	q_3	q_4

(22) Construct DFA which accept strings such that number of 0's divisible by 2 and number of 1's divisible by 3.

Soln:- String = {10101, 11001, 00111, 11100, 0000111, 1001001111 ....}

Language $L = \{x \mid x \in \{0,1\}^* \text{ and } |x|_0 \bmod 2 = 0 \text{ \& } |x|_1 \bmod 3 = 0\}$



State/input	0	1
00	10	01
01	11	02
02	12	00
10	00	11
11	12	01
12	02	10

DFA is define as $M = (Q, \Sigma, \delta, q_0, F)$ where

$Q = \{00, 01, 02, 10, 11, 12\}$

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

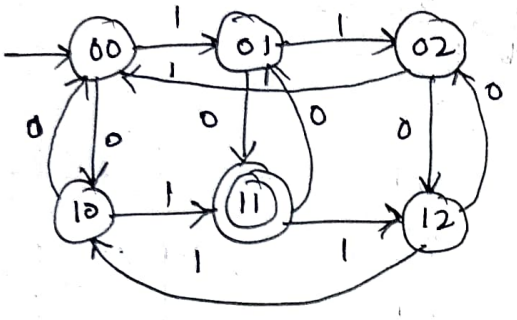
00 - initial state

$F = \{00\}$

δ is define as table.

(23) Construct DFA which accept strings x such that $|x|_0 \bmod 2 = 1$ and $|x|_1 \bmod 3 = 1$.

Soln:- String = {10101, 11001, 00111, 11100, 0000111, 1001001111 ....}



DFA is define as $M = (Q, \Sigma, \delta, q_0, F)$ where

$Q = \{00, 01, 02, 10, 11, 12\}$

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

00 - initial state

$F = \{11\}$

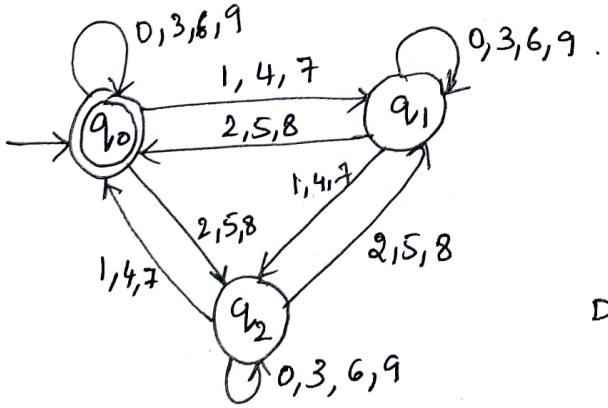
δ is define as table:

State/input	0	1
00	10	01
01	11	02
02	12	00
10	00	11
11	01	12
12	02	10

(24) Construct DFA which accept strings of integer divisible by 3.

Soln:- String = {0, 3, 6, 9, 12, 15, 18, 21, ...}

Language $L = \{x \mid x \in \{0-9\}^* \text{ and } x \bmod 3 = 0\}$



State/Input	0, 3, 6, 9	2, 5, 8	1, 4, 7
→ q_0*	q_0	q_2	q_1
q_1	q_1	q_0	q_2
q_2	q_2	q_1	q_0

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

$Q = \{q_0, q_1, q_2\}$ q_0 - initial state

$\Sigma = \{0-9\}$

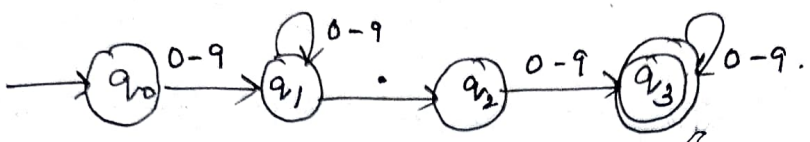
$F = \{q_0\}$

δ is define as table.

(25) Construct DFA which accept strings of decimal number.

Soln:- string = {0.23, 1.33, 4.555333, ...}

Language $L = \{ x.y \mid x, y \in \{0-9\}^+ \}$



DFA is define as $M = (Q, \Sigma, \delta, q_0, F)$ where

$Q = \{q_0, q_1, q_2, q_3\}$

$\Sigma = \{0-9\}$

q_0 - initial state

$F = \{q_3\}$

δ is define as table:

state / input	0 - 9	.
→ q_0	q_1	
q_1	q_1	q_2
q_2	q_3	
q_3	q_3	

2.3 Nondeterministic Finite Automata. (NFA (or) NDFA)

The Finite Automata is called Non deterministic Finite Automata if there are more than one path for a specific input from current state to next state.

→ Like DFA, NFA has 5 tuples.

A Machine $M = (Q, \Sigma, \delta, q_0, F)$ where, ← Formal Definition of NFA

Q - is finite set of states, which is non empty.

Σ - is input alphabet, indicates input set.

δ - is transition function or mapping function. We can determine the next state using this function.

$$\boxed{\delta : Q \times \Sigma \rightarrow 2^Q}$$

q_0 - is an initial state and is in Q.

F - is set of final states.

2.3.2 Definition of Nondeterministic Finite Automata.

2.3.1 An Informal View of Nondeterministic Finite Automata.

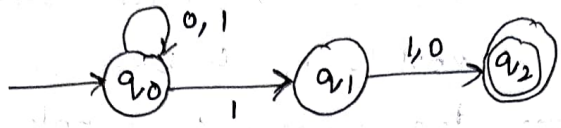


Figure. An NFA accepting the set of all strings whose second last symbol is 1.

- Figure shows a finite automaton which accepts exactly those string that have the symbol 1 in the second last position.
- State q_0 is the initial state from where it moves when it sees a 1 and guesses that there is only one more symbol to follow.
- Since it is possible that there is more than one symbol to be examined, there are transitions from q_0 to itself on reading either a 0 or 1.
- Note that now there are two possible transitions labeled 1 out of q_0 and hence NFA has 2 options - it can move to either q_0 or q_1
- Each such point, it starts a new thread corresponding to each of the new transitions. ~~there~~ Also note that there are no transitions out of q_2.

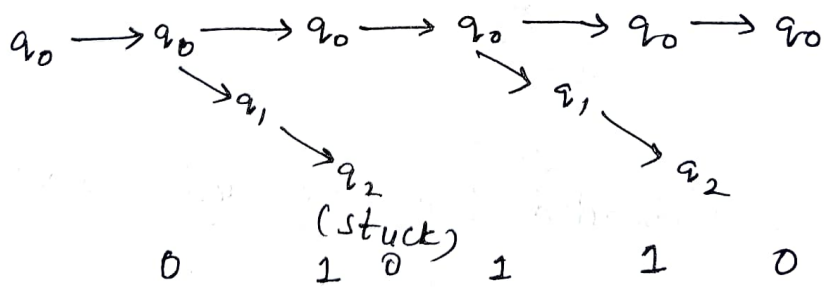


Figure. The states of the NFA ~~while~~ processing the string 01010.

- Consider the input string 01010. The NFA starts from the initial state q_0 and remains there on reading 0.

On Reading a 1 next it may go either to q_0 or to q_1 as seen in the third column.

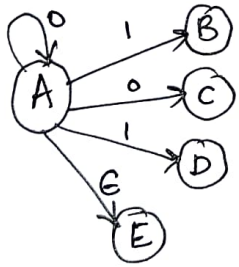
Now the next symbol 0 is read.

- We need to consider transitions out of both q_0 and q_1. q_0 goes to itself while q_1 moves to q_2.
- When the next symbol 1 is read we need to consider two transitions once again – one out of q_0 and the other out of q_2.
- q_0 can go either to q_0 or to q_1 whereas q_2 has no transition on 0 or 1 and hence it "dies".
- With the last input 0 q_0 goes to q_0 while q_1 goes to q_2. Since q_2 is an accepting state the NFA accepts 0101.

NOTE :- In NFA, given the current state there could be multiple next states.

- The next state may be chosen at random.
- All the next states may be chosen in parallel.

Ex:-



2.3.3. The Extended Transition Function ($\hat{\delta}$)

- An extended transition function takes two arguments. The first argument is a state q and the second argument is a string w.
- It describe what happens when we start with

state and process any sequence of string.

Transition function is denoted by δ whereas extended transition function denoted by $\hat{\delta}$.

It takes a state q and a string w and returns a state p (the state that automata reaches when starting from q and processing string w).

Formally, we define $\hat{\delta}$ for an NFA's transition function δ by:

BASIS: $\hat{\delta}(q, \epsilon) = \{q\}$. That is, without reading any input symbols, we are only in the state we began in.

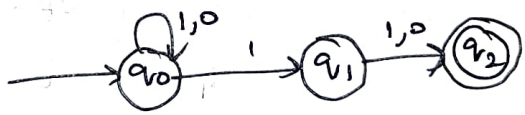
INDUCTION: Suppose w is of the form $w = xa$, where a is the final symbol of w and x is the rest of w. Also suppose that $\hat{\delta}(q, x) = \{p_1, p_2, \ldots, p_k\}$.

Let $\bigcup_{i=1}^{k} \delta(p_i, a) = \{r_1, r_2, \ldots, r_m\}$.

Then $\hat{\delta}(q, w) = \{r_1, r_2, \ldots, r_m\}$.

We compute $\hat{\delta}(q, w)$ by first computing $\hat{\delta}(q, x)$, and by then following any transition from any of these states that is labelled a.

Ex:- Let us use $\hat{\delta}$ to describe how the string 01010 is processed by the NFA.



Soln:-

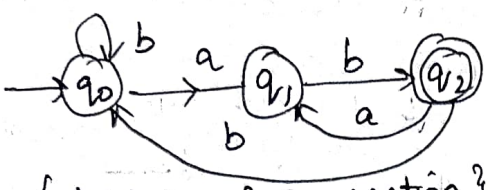
1. $\hat{\delta}(q_0, \epsilon) = \{q_0\}$
2. $\hat{\delta}(q_0, 0) = \delta(q_0, 0) = \{q_0\}$.
3. $\hat{\delta}(q_0, 01) = \delta(q_0, 1) = \{q_0, q_1\}$.
4. $\hat{\delta}(q_0, 010) = \delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}$
5. $\hat{\delta}(q_0, 0101) = \delta(q_0, 1) \cup \delta(q_2, 1) = \{q_0, q_1\} \cup \phi = \{q_0, q_1\}$
6. $\hat{\delta}(q_0, 01010) = \delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}$

Line (1) is the basis rule. We obtain line (2) by applying δ with input 0 to the state q_0 from the previous set to get $\{q_0\}$ once again. Now input 1 applied to this result in $\{q_0, q_1\}$ in line (3). Line (4) is obtained by applying δ with input 0 to the two states resulting from (3) and taking their union. Lines (5) and (6) are similar to (4).

(26) Construct DFA accepting all strings over {a, b} ending with 'ab'?

Soln:- $L = \{ab, aab, bab, \ldots\}$

min. length = 2. no. of states = 2+1 = 3.

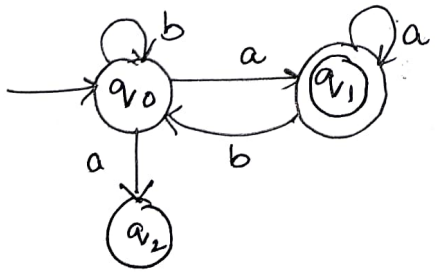


NOTE:- {Add to 2.3 section}.

NFA :-

1. The FA are called NFA, when there exist many paths for specific i/p from the current state to the next state.
2. It is easy to construct NFA than DFA for a given regular language.
3. Every NFA is not DFA, but each NFA can be translated into DFA.
4. DFA → only one path for specific input.
   NFA → many paths for specific input.
5. NFA is defined in the same way as DFA, but with two exceptions.
   → It contains multiple next state and
   → It contains ε transitions.

Ex:-



$\delta(q_0, a) = q_1$ and

$\delta(q_0, a) = q_2$

↓

multiple next state.

∴ specific path is not fixed.

if the path doesn't give result. then we can select another path to reach the final state.

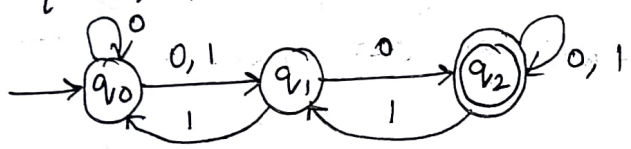
Graphical Representation of a NFA :-

An NFA can be represented by digraphs called state diagrams. In which:

1. The state is represented by vertices.
2. The arc labeled with an i/p character show the transitions.
3. The initial state is marked with an arrow.
4. The final state is denoted by the double circle.

Problems on NFA :-

(1.) $Q = \{q_0, q_1, q_2\}$ $\Sigma = \{0, 1\}$, $q_0 = \{q_0\}$, $F = \{q_2\}$.



Soln:-

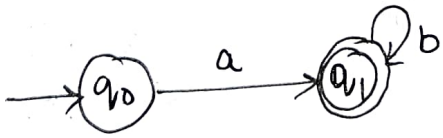
States	0	1
→ q_0	$\{q_0, q_1\}$	$\{q_1\}$
q_1	q_2	q_0
* q_2	q_2	$\{q_1, q_2\}$

transition table.

(2.) Construct NFA where language given as $L = \{$start with a$\}$. $\Sigma = \{a, b\}$.

Soln:- $L = \{a, ab, abb, abbb, abab, \ldots\ldots\}$.

min. length = 1 no. of states = 2.

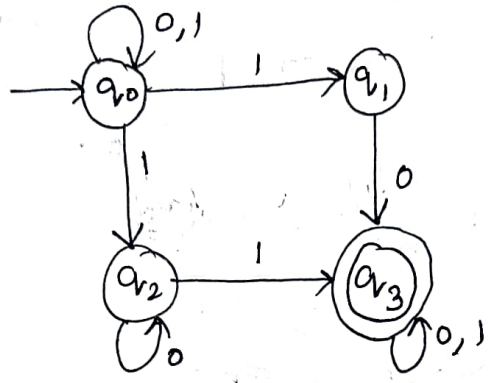


③. Design NFA for transition table.

State	0	1
→ q_0	q_0, q_1	q_0, q_2
q_1	q_3	ϵ
q_2	$q_2\ q_3$	q_3
* q_3	q_3	q_3

Soln:-

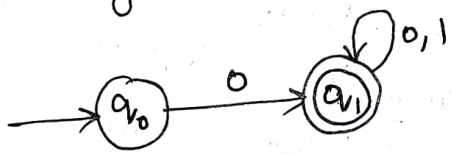
transition diagram



④. Construct NFA where $L = \{$ set of all strings that starts with $0\}$. $\Sigma = \{0, 1\}$.
(i.e binary alphabet).

Soln:- $L = \{0, 00, 01, 001, 010, 000, \ldots\}$

min. length = 1, no. of state = 2.



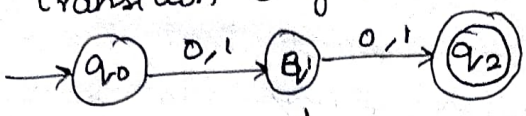
NOTE :-
$\delta(q_0, 1) \rightarrow \phi$ {i.e dead configuration}
because q_0 on 1, there is no state here, so we need to consider as dead configuration. This state diagram can be represented in DFA by taking dead state also, but in NFA, no need of taking dead state, only take the state that are accepting, strings based on given condition.

⑤. Construct NFA that accepts set of all strings over $\{0, 1\}$ of length 2.

soln:- $\Sigma = \{0, 1\}$.

$L = \{00, 01, 10, 11\}$

state transition diagram.

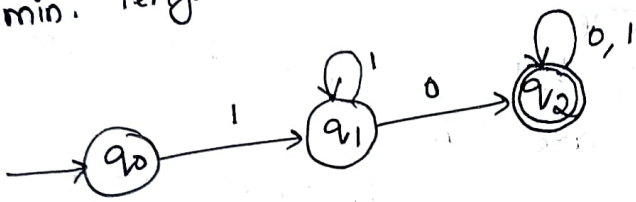


(6) Construct NFA accept for the set of all strings over $\{0,1\}$ that starts with 10.

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

$L = \{10, 100, 1011, \ldots\}$.

min. length = 2, no. of states = 3.



2.3.4. The Language of an NFA.

An NFA accepts a string w if it is possible to make any sequence of choices of next state, while reading the characters of w, and go from the start state to any accepting state.

Formally, if $A = (Q, \Sigma, \delta, q_0, F)$ is an NFA, then, $L(A) = \{w \mid \hat{\delta}(q_0, w) \cap F \neq \phi\}$.

i.e, $L(A)$ is the set of strings w in Σ^* such that $\hat{\delta}(q_0, w)$ contains at least one accepting state.

Ex 1:- P.T NFA accepts the language $L = \{w \mid w$ has the symbol 1 in the second last position$\}$.

The proof is by mutual induction involving the following 3 statements:

1. $\hat{\delta}(q_0, w)$ contains q_0 for every w.
2. $\hat{\delta}(q_0, w)$ contains q_1 if and only if w ends in 1.
3. $\hat{\delta}(q_0, w)$ contains q_2 if and only if w has the symbol 1 in the second last position.

Since the language of the given automaton is the set of strings w such that $\hat{\delta}(q_0, w)$ contains q_2, the proof of (3) ensures that the language of the NFA is the set of strings that have the symbol 1 in the second last position.

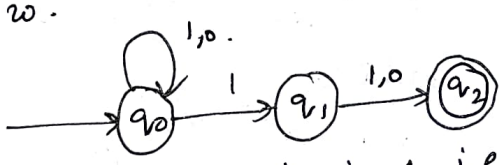
The proof is by induction on the length of w.

<u>BASIS</u>: If $|w| = 0$ then $w = \epsilon$.

Statement (1) says that $\hat{\delta}(q_0, \epsilon)$ contains q_0 which is obviously true. Since ϵ does not satisfy the conditions of statement (2) or (3), it is clear that both the statements are true.

<u>INDUCTION</u>: Let us assume that $w = xa$ where a may be either 0 or 1 and that all the three statements are true for x.

1. We know $\hat{\delta}(q_0, x)$ contains q_0. Since there are transitions from q_0 to itself on both 0 and 1, it follows that $\hat{\delta}(q_0, w)$ contains q_0 and hence statement (1) is true for w.



2. (If) Suppose w ends in 1, i.e. $a = 1$. By statement (1) we know $\hat{\delta}(q_0, x)$ contains q_0. Since there is a transition from q_0 to q_1 on reading 1 it follows that $\hat{\delta}(q_0, w)$ contains q_1.

(only if) Assume $\hat{\delta}(q_0, w)$ contains q_1. However this is possible only if w is of the form $x1$. Hence the result (2).

3. (If) Suppose w has the symbol 1 in the second last position. By statement (2) we know $\hat{\delta}(q_0, x)$ contains q_1. Since there are transitions from q_1 to q_2 on reading

0 or 1 it follows that $\hat{\delta}(q_0, w)$ contains q_2.

(only if) Assume $\hat{\delta}(q_0, w)$ contains q_2. On examining the transition function we find that this is possible only if w has the symbol 1 in the second last position. Hence statement (3).

Non deterministic finite Automata and its Conversion:-

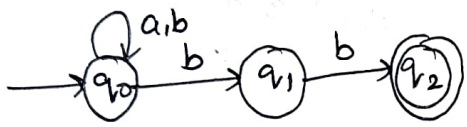
2.3.7 Conversion of NFA to DFA problems :- (or) Subset construction method

DFA :- $\delta : Q \times \Sigma \rightarrow Q$ (i.e one path).

NFA :- $\delta : Q \times \Sigma \rightarrow 2^Q$. (Multiple path)

NOTE :- Every DFA is an NFA, but not vice versa. But there is an equivalent DFA for every NFA.

① Convert to a DFA the following NFA.



Soln:- Step 1 :- Construct NFA transition table by using the state transition diagram.

State/input	a	b
→ q_0	q_0	$\{q_0, q_1\}$
q_1	ϕ	q_2
* q_2	ϕ	ϕ

NOTE :- NFA to DFA Conversion is possible, but DFA to NFA conversion is not possible.

Step 2 :- Convert NFA table to DFA table.

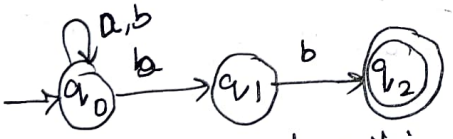


figure. NFA transition diagram.

State/input	a	b
→ q_0	q_0	$[q_0 q_1]$
$[q_0 q_1]$	$[q_0]$	$[q_0 q_1 q_2]$
* $[q_0 q_1 q_2]$	$[q_0]$	$[q_0 q_1 q_2]$

Consider as single state

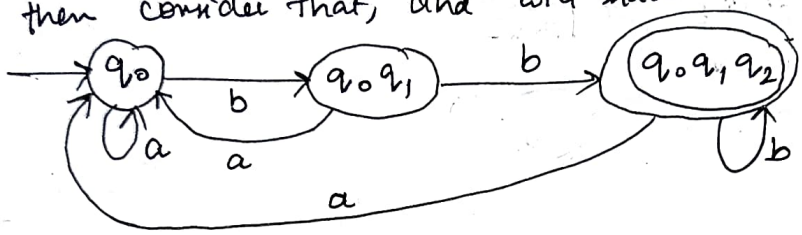
NOTE :- ① To Construct DFA table, Firstly look at state diagram of NFA and table, initial state is q_0, $\delta(q_0, a) \rightarrow q_0$. $|||^{ly}$ $\delta(q_0, b) \rightarrow [q_0 q_1]$ [∴ Consider it as single state and represent it in square bracket]

② Then take $[q_0 q_1]$ as new state and find the transition on input a, b

[i.e. $[q_0 q_1]$ will become now,
$\delta(q_0, a) \rightarrow q_0$, $\delta(q_1, a) \rightarrow \phi$. and III^{ly},
$\delta(q_0, b) \rightarrow \{q_0 q_1\}$, $\delta(q_1, b) \rightarrow q_2$.
$\therefore$ new state $[q_0 q_1 q_2]$ is found.] and so on.

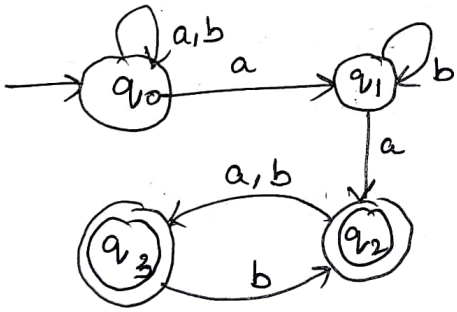
Therefore whatever the new state is there, that we have to write in the transition function and that new state becomes the current state. and these input symbols yields to the next state.

<u>Step 3</u>:- Find the final state., check in which state q_2 state is present i.e, $[q_0 q_1 q_2]$ it becomes the final state. [suppose if we have q_2 in more than one then consider that, and will have more than 1 final state]



State transition of DFA.

(2). Convert to a DFA the following NFA.



NFA.

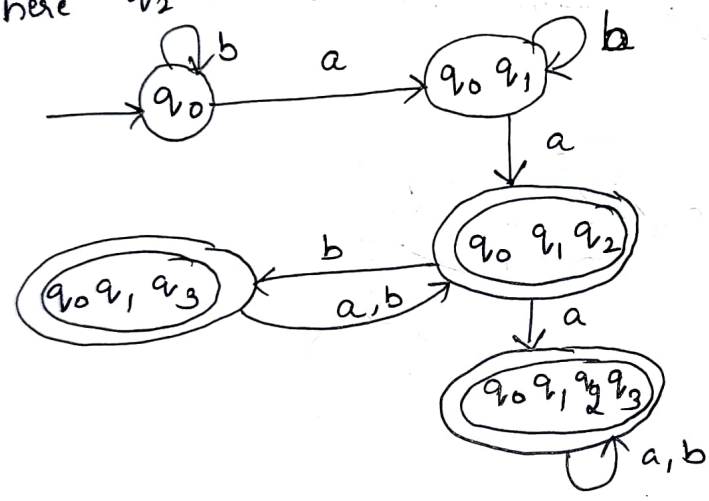
Soln:- <u>Step 1</u> :- Construct transition table of given NFA. with the help of above NFA transition state diagram.

δ	a	b
$\rightarrow q_0$	$\{q_0 q_1\}$	q_0
q_1	q_2	q_1
* q_2	q_3	q_3
* q_3	–	q_2

Step 2 :- Now construct DFA table by using NFA table.

δ	a	b
$\rightarrow q_0$	$[q_0 q_1]$	q_0
$[q_0 q_1]$	$[q_0 q_1 q_2]$	$[q_0 q_1]$
$[q_0 q_1 q_2]$	$[q_0 q_1 q_2 q_3]$	$[q_0 q_1 q_3]$
$[q_0 q_1 q_3]$	$[q_0 q_1 q_2]$	$[q_0 q_1 q_2]$
$[q_0 q_1 q_2 q_3]$	$[q_0 q_1 q_2 q_3]$	$[q_0 q_1 q_2 q_3]$

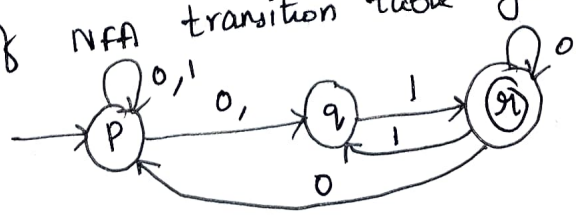
Step 3 :- Construct DFA State diagram by using DFA table.
here q_2 and q_3 are final state.



(3) · Convert to a DFA the following NFA.

δ	0	1
$\rightarrow p$	$\{p, q\}$	$\{p\}$
q	ϕ	$\{r\}$
$*r$	$\{p, r\}$	$\{q\}$.

soln:- **Step 1** :- NFA transition diagram is constructed with the help of NFA transition table given above

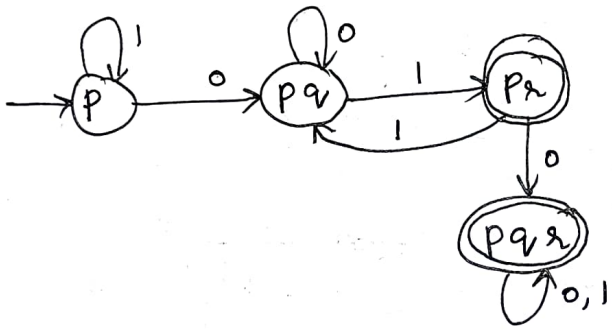


Step 2 :- Now construct DFA table by using NFA table.

δ	0	1
→ p	[p, q]	p
[p q]	[p q]	[p r]
* [p r]	[p q r]	[p q]
[p q]	[p q]	[p r]
* [p q r]	[p q r]	[p q r]

∴ from ∫ above transition diagram, "r" is the final state. Construct state diagram of DFA in step 3.

Step 3 :-



(4.) Convert to a DFA the following NFA :-

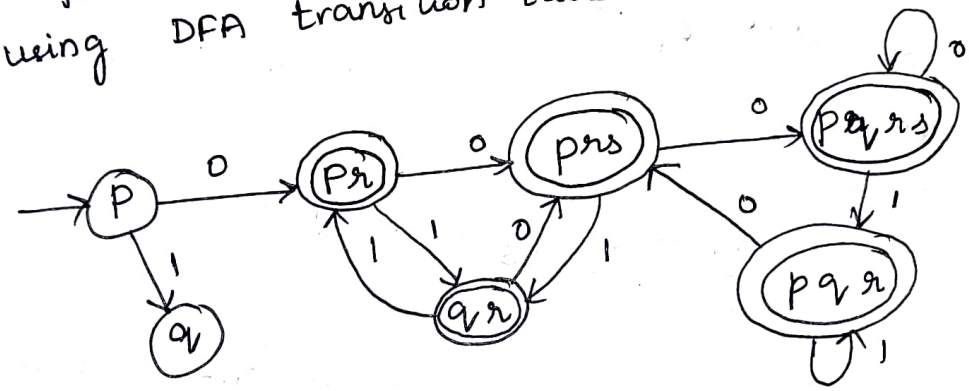
δ	0	1
→ p	{p, r}	{q}
q	{r, s}	{p}
* r	{p, s}	{r}
* s	{q, r}	ϕ

Soln:- Step 1 :- from above NFA table, construct DFA table.

δ	0	1
→ p	[p r]	[q]
* [p r]	[p r s]	[q r]
* [q r]	[p r s]	[p r]
* [p r s]	[p r s q]	[q r]
* [p q r s]	[p q r s]	[p q r]
* [p q r]	[p r s]	[p q r]

← DFA transition table.

Step 2:- from step 2, construct DFA state diagram by using DFA transition table.



(5). Convert the following NFA to DFA.

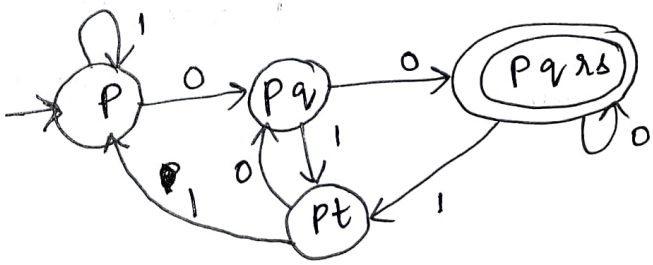
	0	1
→ p	{p, q}	{p}
q	{r, s}	{t}
r	{p, r}	{t}
* s	ϕ	ϕ
* t	ϕ	ϕ

↑ NFA table.

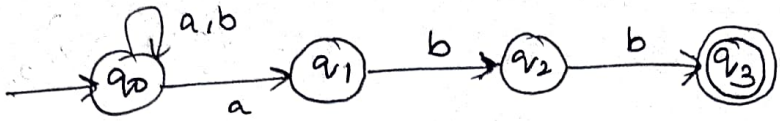
Soln:- DFA table:-

	0	1
→ p	[p q]	[p]
[p q]	[p q r s]	[p t]
* [p t]	[p q]	[p]
* [p q r s]	[p q r s]	[p t]

DFA state diagram:-



(6). Convert the following NFA to DFA.



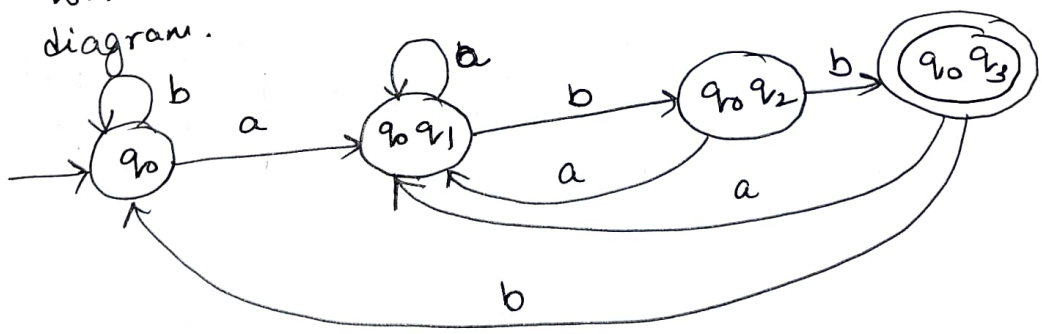
Soln:- Step 1 :- NFA transition table from the given above NFA state diagram.

δ_N	a	b
→ q_0	$\{q_0\ q_1\}$	q_0
q_1	–	q_2
q_2	–	q_3
* q_3	–	–

Step 2 :- Now construct DFA table by using NFA table.

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

Step 3 :- with the help of DFA table, now draw the state diagram.



$A_D = (Q_D, \Sigma, \delta_D, q_0, F_D)$.

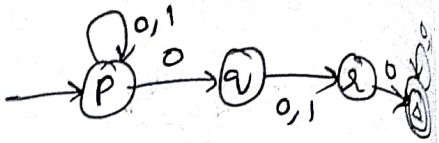
$Q_D = \{\{q_0\}, \{q_0 q_1\}, \{q_0 q_2\}, \{q_0 q_3\}\}$

$\Sigma = \{a, b\}$ δ_D is shown above.

$q_0 = \{q_0\}$, $F_D = \{q_0 q_3\}$.

(7) Convert the following NFA to DFA

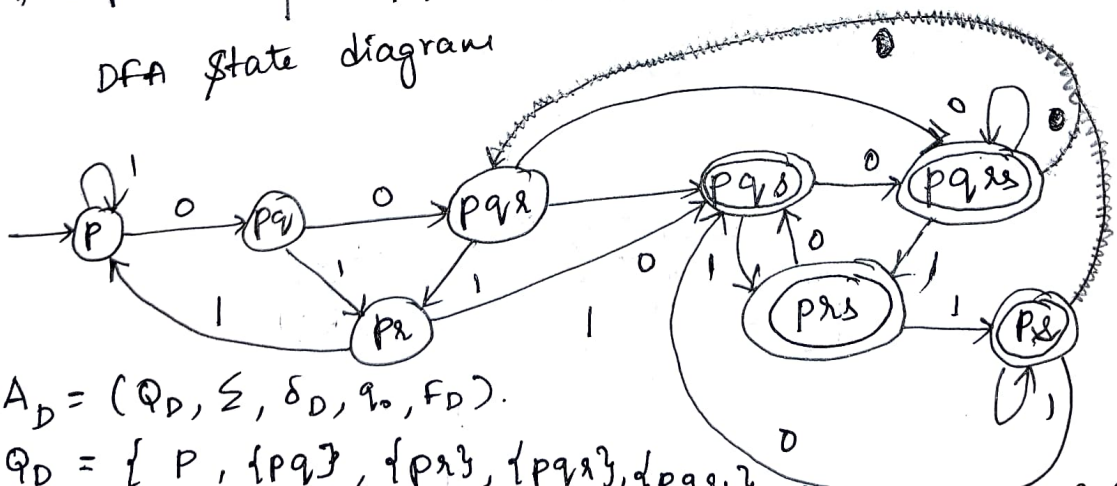
δ_N	0	1
→ p	{pq}	{p}
q	{r}	{r}
r	{s}	–
* s	{s}	{s}



Soln:- DFA transition diagram :-

δ_D	0	1
→ p	[pq]	[p]
[pq]	[pqr]	[pr]
[pr]	[pqs]	[p]
[pqr]	[pqrs]	[pr]
* [pqrs]	[pqrs]	[prs]
* [prs]	[pqs]	[ps]
* [pqs]	[pqrs]	[prs]
* [ps]	[pqs]	[ps]

DFA state diagram



$A_D = (Q_D, \Sigma, \delta_D, q_0, F_D)$.

$Q_D = \{ p, \{pq\}, \{pr\}, \{pqr\}, \{pqrs\}, \{prs\}, \{pqs\}, \{ps\} \}$

$\Sigma = \{0, 1\}$, $q_0 = \{p\}$, $F_D = \{ \{prs\}, \{ps\}, \{pqs\}, \{pqrs\} \}$

δ_D is define above in the state diagram

2.3.5 Equivalence of Deterministic and Nondeterministic Finite Automata.

Procedure to convert NFA to DFA (Subset construction method)

- Let NFA $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ and its equivalent DFA $D = (Q_D, \Sigma, \delta_D, q_0, F_D)$ such that $L(N) = L(D)$.
- Input alphabets and start state of both the automata is always same.
- Remaining tuples of DFA are calculated as:

1. Q_D is the set of subset of Q_N i.e, Q_D is the power set of Q_N. If Q_N has n states then Q_D has 2^n states.
2. F_D is the all set of D's states that includes at least one final state of N.
3. For each set $S \subseteq Q_N$ and for each input symbol a in Σ

$$\delta_D(S, a) = \bigcup_{p \text{ in } S} \delta_N(p, a).$$

Steps to identify equivalence of two finite Automata:-

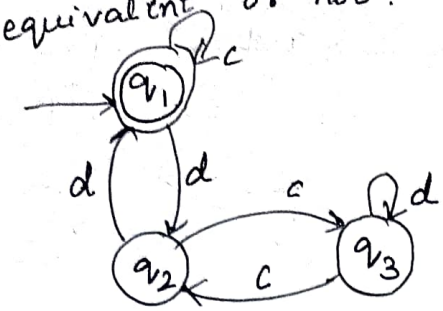
1. For any pair of states (q_i, q_j) the transition for input $a \in \Sigma$ is defined by $\{q_a, q_b\}$ where $\delta\{q_i, a\} = q_a$ and $\delta(q_j, a) = q_b$.
   the two automata are not equivalent if for a pair $\{q_a, q_b\}$ one is intermediate state and other is final state.
2. If initial state is final state of one automata, then, in second automata also initial state must be final state for them to be equivalent.

Refer (1)

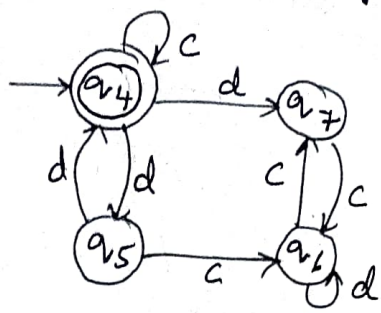
Problems :-

1. Two automatas namely A and B. Check whether they are equivalent or not.

(A)



(B)



Soln :- NOTE :- If ~~both~~ pair of states, both holding final state or intermediate state or both, then we can say they are equivalent. Otherwise it is not equivalent.

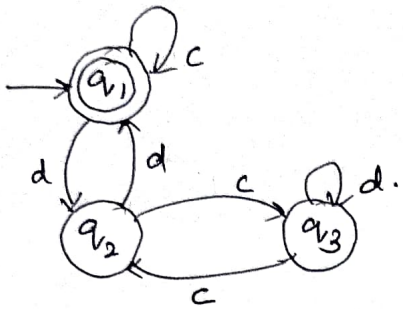
Step 1 :- Initially, take q_1 and q_4 as they are final state and initial state from the automata A and B.

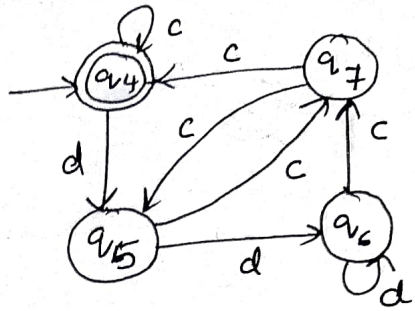
States.	c	d
(q_1, q_4)	(q_1, q_4) FS FS	(q_2, q_5) Is Is.
(q_2, q_5)	(q_3, q_6) Is Is	(q_1, q_4) FS FS
(q_3, q_6)	(q_2, q_7) Is Is	(q_3, q_6) Is Is
(q_2, q_7)	(q_3, q_6) Is Is	(q_1, q_4) FS FS.

from transition table, All the pairs are forming Final state (and or) also intermediate state. Then we can say that both A and B are equivalent.

NOTE :- Fs → Final state.
Is → Intermediate state.

(2.) Find out whether the following automata are equivalent or not.





Soln:-

States	c	d
(q_1, q_4)	(q_1, q_4) → FS, FS	(q_2, q_5) → IS, IS
(q_2, q_5)	(q_3, q_7) → IS, IS	(q_1, q_6) → FS, IS.

If one of the pair is not in same state. Here (q_1, q_6) is (Final, Intermediate) so we can state that the given automata is not equivalent.

∴ A & B are not equivalent.

Minimization of DFA :-

The minimization of DFA means reducing the number of states from a given DFA.

Step1 :- Remove all the states that are unreachable from initial state via any set of transition of DFA.

Step2 :- Draw the transition table for all pair of states.

Step3 :- Now split the transition table into two ~~states~~ tables T_1 and T_2. where,

T_1 contains all final states.

T_2 contains non-final states and initial state.

Step4 :- Find similar rows from T_1 such that

$\delta(a, a) = p$

$\delta(r, a) = p$

i.e, find the two states which have same value of a and b and remove one of them.

Step5 :- Repeat step3 until we find no similar rows available in T_1.

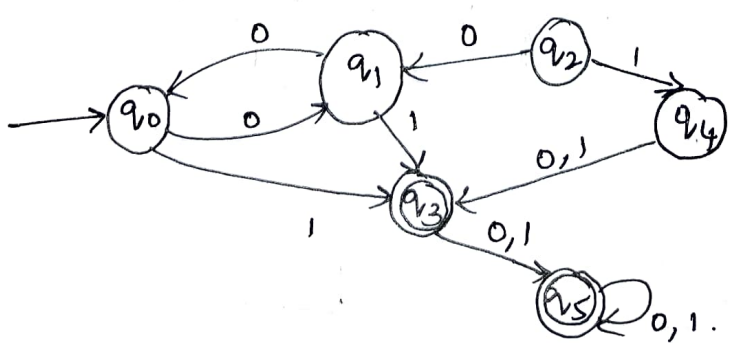
Step6 :- Repeat step 3 and step 4 for table T_2 also.

Step7 :- Now combine the reduced T_1 and T_2 tables.

i.e, the final transition table of minimized DFA.

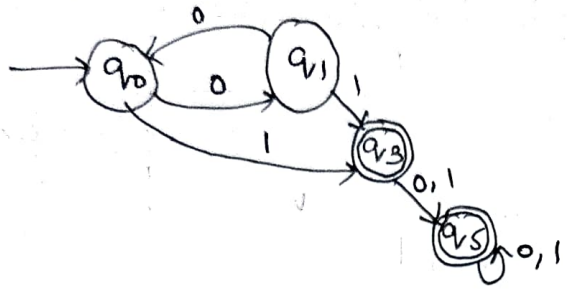
Problems :-

(1.)



Soln:- Step 1 :- Remove q_2 and q_4 states in the finite automata as they are unreachable states from q_0.

Step 2 :- Now draw the transition table for the rest of the states.



States	0	1
→ q_0	q_1	q_3
q_1	q_0	q_3
* q_3	q_5	q_5
* q_5	q_5	q_5

Step 3 :- (i) Table which starts from non-final states. (T_1)

States	0	1
→ q_0	q_1	q_3
q_1	q_0	q_3

(ii) Table which starts from final states. (T_2).

states	0	1
* q_3	q_5	q_5
* q_5	q_5	q_5

Step 4 :- Set 1 has no similar rows, they will be same in table 1.

But, whereas in the table T_2, set 2 has similar rows, ~~so~~ skip q_5 and then replace q_5 by q_3

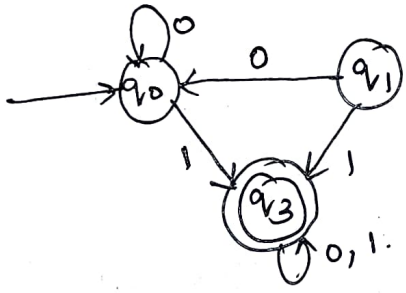
States	0	1
* q_3	q_3	q_3

Step 5 :- Combine set 1 and set 2.

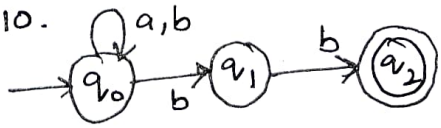
State	0	1
→ q_0	q_0	q_3
q_1	q_0	q_3
* q_3	q_3	q_3

← final transition table

final minimization DFA state diagram is shown below.



NOTE :- Difference between DFA and NFA

DFA	NFA
1. Each transition leads to exactly one state called as deterministic.	1. A transition leads to a subset of states i.e, some transitions can be non-deterministic.
2. Accepts input if the last state is in Final.	2. Accepts input if one of the last states is in Final.
3. Requires more space.	3. Requires less space.
4. Empty string transitions are not seen in DFA.	4. Permits empty string transition.
5. For a given state, on a given input we reach a deterministic and unique state.	5. For a given state, on a given input we reach more than one state.
6. DFA is a subset of NFA.	6. Need to convert NFA to DFA in the design of a Compiler
7. $\delta : Q \times \Sigma \rightarrow Q$ for ex:- $\delta(q_0, a) = \{q_1\}$	7. $\delta : Q \times \Sigma \rightarrow 2^Q$. for ex: $\delta(q_0, a) = \{q_1, q_2\}$
8. DFA is more difficult to construct.	8. NFA is easier to construct.
9. DFA is understood as one machine	9. NFA is understood as multiple small machines computing at the same time.
10.	10. 

2.5 Finite Automata With Epsilon-Transitions

- NFA includes transition without taking any input as Epsilon transition.
- If NFA contains ϵ transition then NFA is called as ϵ-NFA.
- Every state on ϵ goes to itself that allowed change of state without reading any input.

2.5.1 Uses of ϵ-Transitions

ϵ-NFA that accepts decimal numbers consisting of:

1. An optional + or − sign,
2. A string of digits,
3. A decimal point, and
4. Another string of digits.

Either this string of digits, or the string can be empty, but at least one of the two strings of digits must be nonempty.

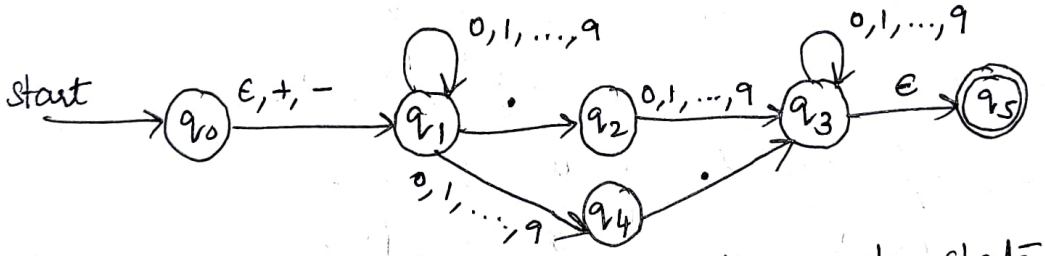


Figure. An NFA that searches for the words state and regular.

where, ϵ-NFA $\Rightarrow A = (Q_A, \Sigma, \delta_A, q_0, F_A)$

$Q_A = \{q_0, q_1, q_2, q_3, q_4, q_5\}$.

q_0 – initial state.

$\Sigma = \{\epsilon, +, -, \cdot, 0\text{-}9\}$

$F_A = \{q_5\}$.

δ_A is shown in above diagram and below table.

NOTE :- $\boxed{\delta : Q \times \Sigma \cup \epsilon \longrightarrow 2^Q}$

2.5.2 The formal notation for an ϵ-NFA.

Represent an ϵ-NFA exactly as we do an NFA, with one exception : the transition function must include information about transition on ϵ.

Formally, we represent an ϵ-NFA A by

$A = (Q, \Sigma, \delta, q_0, F)$, where all components have their same interpretation as for an NFA, except that δ is now a function that takes as arguments :

1. A state in Q, and
2. A member of $\Sigma \cup \{\epsilon\}$, that is, either an input symbol, or the symbol ϵ, We require that ϵ, the symbol for the empty string, cannot be a member of the alphabet Σ, so no confusion results.

transition table :-

States/input	ϵ	+, −	.	0, 1, ..., 9
q_0	$\{q_1\}$	$\{q_1\}$	ϕ	ϕ
q_1	ϕ	ϕ	$\{q_2\}$	$\{q_1, q_4\}$
q_2	ϕ	ϕ	ϕ	$\{q_3\}$
q_3	$\{q_5\}$	ϕ	ϕ	$\{q_3\}$
q_4	ϕ	ϕ	$\{q_3\}$	ϕ
q_5	ϕ	ϕ	ϕ	ϕ

2.5.3. Epsilon-Closures.

ϵ-closure (q) is defined as the set of all states which are reachable from q without giving any i/p.

NOTE :- Here, include both implicit and explicit transition.

$\delta(q, \epsilon) = q$ in case of DFA or an NFA.

Formally, we define the ϵ-closure ECLOSE(q) recursively, as follows:

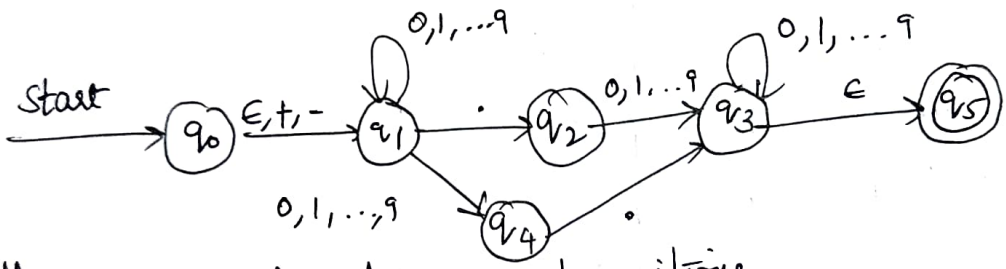
BASIS: state q is in ECLOSE(q).

INDUCTON: If state p is in ECLOSE(q), and there is a transition from state p to state r labeled ϵ, then r is in ECLOSE(q).

More precisely, if δ is the transition function of the ϵ-NFA involved, and p is in ECLOSE(q), then ECLOSE(q) also contains all the states in $\delta(p, \epsilon)$.

Example 1:- for the automaton of figure below, each state is its own ϵ-closure, with two exceptions:

ECLOSE $(q_0) = \{q_0, q_1\}$ and ECLOSE $(q_3) = \{q_3, q_5\}$.



there are only two ϵ-transitions,

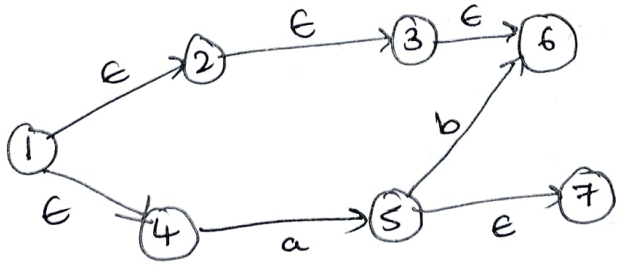
q_1 to ECLOSE (q_0) and

q_5 to ELCOSE (q_3).

Example 2:- A more complex example is explained below. For this collection of states, which may be part of some ϵ-NFA, we can conclude that

ECLOSE(1) = {1, 2, 3, 4, 6}. as there is an path $1 \rightarrow 2 \rightarrow 3 \rightarrow 6$. which is reachable.

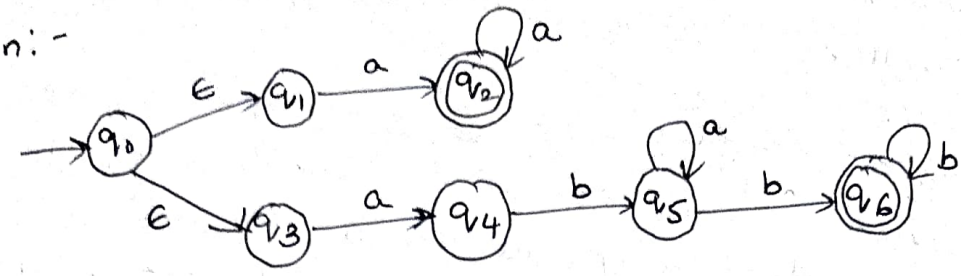
But, state 7 is not in ECLOSE(1) from state 1.



Problems :-

(1) write ϵ-NFA for the following. $L(aa^* + aba^* b^*)$.

Soln:-

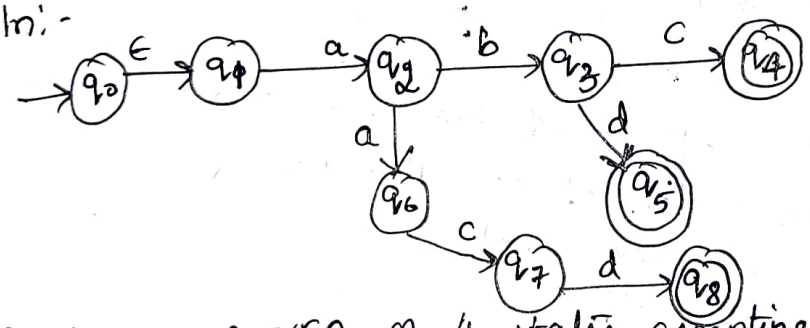


$\delta_{\epsilon\text{-NFA}}$.

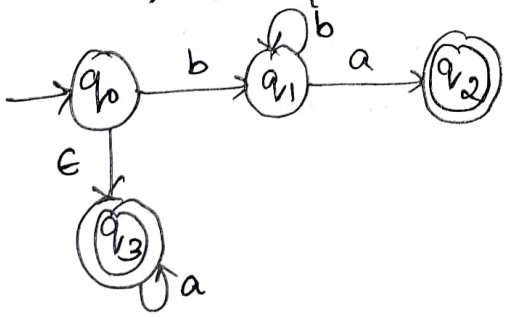
states	ϵ	a	b
$\rightarrow q_0$	q_1	-	-
q_1	-	q_2	-
* q_2	-	q_2	-
q_3		q_4	-
q_4		-	q_5
q_5		q_5	q_6
* q_6		-	q_6

(2). L (abc, abd, aacd).

Soln:-



(3) Design an ϵ-NFA of 4 states accepting the language $L = \{a^n : n \geq 0\} \cup \{b^n a : n \geq 1\}$.



2.5.4 Extended Transitions and Languages for ϵ-NFA's.

The ϵ-closure allows us to explain easily what the transitions of an ϵ-NFA look like when given a sequence of (non-ϵ) inputs.

Suppose that $E = (Q, \Sigma, \delta, q_0, F)$ is an ϵ-NFA then define $\hat{\delta}$, the extended transition function, to reflect what happens on a sequence of inputs.

The intent is that $\hat{\delta}(q, w)$ is the set of states that can be reached along a path whose labels, when concatenated, form the string w.

As always, ϵ's along this path do not contribute to w.

The appropriate recursive definition of $\hat{\delta}$ is:

BASIS: $\hat{\delta}(q, \epsilon) = \text{ECLOSE}(q)$. i.e, if the label of the path is ϵ, then we can follow only ϵ-labeled arcs extending from state q; i.e exactly what ECLOSE does.

INDUCTION: Suppose w is of the form xa, where a is the last symbol of w. Note that a is a member of Σ; it cannot be ϵ, which is not in Σ.

We compute $\hat{\delta}(q, w)$ as follows:

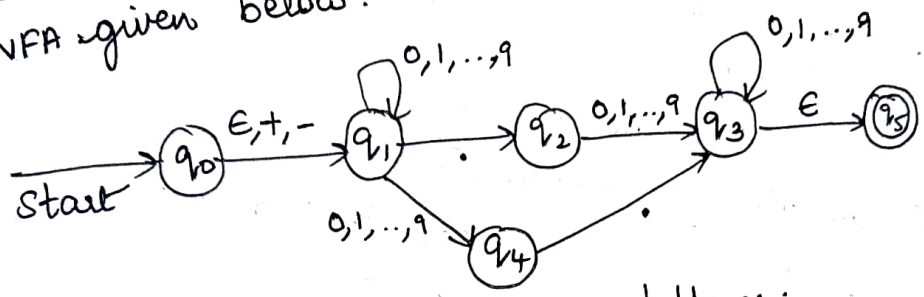
1. Let $\{p_1, p_2, \ldots, p_k\}$ be $\hat{\delta}(q, x)$. i.e the p_i's are all and only the states that we can reach from q following a path labeled x. This path may end with one or more transitions labeled ϵ, and may have other ϵ-transitions, as well.

2. Let $\bigcup_{i=1}^{k} \delta(p_i, a)$ be the set $\{r_1, r_2, \ldots, r_m\}$. i.e follow all transitions labeled a from states we can reach from q along paths labeled x.

The r_j's are some of the states we can reach from q along paths labeled w. The additional states we can reach are found from the r_j's by following ϵ-labeled arcs in step(3), below.

3. Then $\hat{\delta}(q, w) = \bigcup_{j=1}^{m} \text{ECLOSE}(r_j)$. This additional closure step includes all the paths from q labeled w, by considering the possibility that there are additional ϵ-labeled arcs that we can follow after making a transition on the final "real" symbol, a.

<u>Example 1:-</u> Let us compute $\hat{\delta}(q_0, 5.6)$ for the ϵ-NFA given below.



<u>Soln :-</u> Steps to compute are as follows:

1. $\hat{\delta}(q_0, \epsilon) = \text{ECLOSE}(q_0) = \{q_0, q_1\}$.

2. Compute $\hat{\delta}(q_0, 5)$ as follows:

a) First compute the transitions on i/p 5 from the states q_0 and q_1 that we obtained in the calculation of $\hat{\delta}(q_0, \epsilon)$, above.

i.e, we compute $\delta(q_0, 5) \cup \delta(q_1, 5) = \{q_1, q_4\}$.

b) Next, ϵ-close the members of the set computed in step (a). we get,

$\text{ECLOSE}(q_1) \cup \text{ECLOSE}(q_4) = \{q_1\} \cup \{q_4\} = \{q_1, q_4\}$.

That set is $\hat{\delta}(q_0, 5)$.

This two-step (a) & (b) pattern repeats for the next two symbols.

3. Compute $\hat{\delta}(q_0, 5.)$ as follows:

a) first compute $\delta(q_1, .) \cup \delta(q_4, .) = \{q_2\} \cup \{q_3\}$

$= \{q_2, q_3\}$.

b) Then compute,

$\hat{\delta}(q_0, 5.) = \text{ECLOSE}(q_2) \cup \text{ECLOSE}(q_3) = \{q_2\} \cup \{q_3, q_5\}$

$= \{q_2, q_3, q_5\}$

4. III^{ly} compute $\hat{\delta}(q_0, 5.6)$ as follows:

a) First compute $\delta(q_2, 6) \cup \delta(q_3, 6) \cup \delta(q_5, 6)$

$$= \{q_3\} \cup \{q_3\} \cup \phi.$$

$$= \{q_3\}.$$

b) Then compute $\hat{\delta}(q_0, 5.6) = ECLOSE(q_3) = \{q_3, q_5\}$.

$\therefore$ Language of an ϵ-NFA $E = (Q, \Sigma, \delta, q_0, F)$.

$$L(E) = \{ w \mid \hat{\delta}(q_0, w) \cap F \neq \phi \}.$$

i.e, the language of E is the set of strings w that take the start state to at least one accepting state.

2.5.5 Eliminating ϵ-Transitions

Given any ϵ-NFA E, we can find a DFA D that accepts the same language as E.

Construction we use is very close to the subset construction, as the states of D are subsets of the states of E.

The only difference is ϵ-transitions of E, which we do through the mechanism of the ϵ-closure.

Let $E = (Q_E, \Sigma, \delta_E, q_0, F_E)$. Then equivalent DFA

$$D = (Q_D, \Sigma, \delta_D, q_0, F_D).$$

is defined as follows:

1. Q_D is the set of subsets of Q_E. We shall find that the only accessible states of D are the ϵ-closed subsets of Q_E, i.e, those sets $S \subseteq Q_E$ such that

$S = ECLOSE(S)$.

The ϵ-closed sets of states S are those such that any ϵ-transitions out of one of the states in S lead to a state that is also in S.

Note that, ϕ is an ϵ-closed set.

2. $q_D = ECLOSE(q_0)$; i.e, we get the start state of D by closing the set consisting of only the start state of E. (Note that, this rule differs from the original Subset construction, where the start state of the constructed automaton was just the set containing the start state of the given NFA.

3. F_D is those sets of states that contain at least one accepting state of E. i.e, $F_D = \{S \mid S \text{ is in } Q_D \text{ and } S \cap F_E \neq \phi\}$.

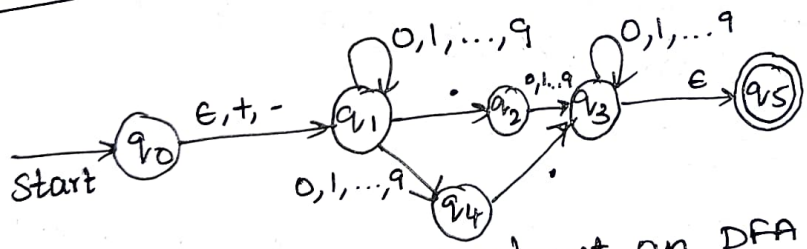
4. $\delta_D(S, a)$ is computed, for all a in Σ and sets S in Q_D by:

(a) Let $S = \{p_1, p_2, \ldots, p_k\}$.

(b) Compute $\bigcup_{i=1}^{k} \delta(p_i, a)$; let this set be $\{r_1, r_2, \ldots, r_m\}$.

(c) Then $\delta_D(S, a) = \bigcup_{j=1}^{m} ECLOSE(r_j)$.

Example 1 :- Eliminate ϵ-transitions from the ϵ-NFA given



Soln:- From E, we construct an DFA D, shown as below. the state ϕ has transitions to itself on all input symbols.

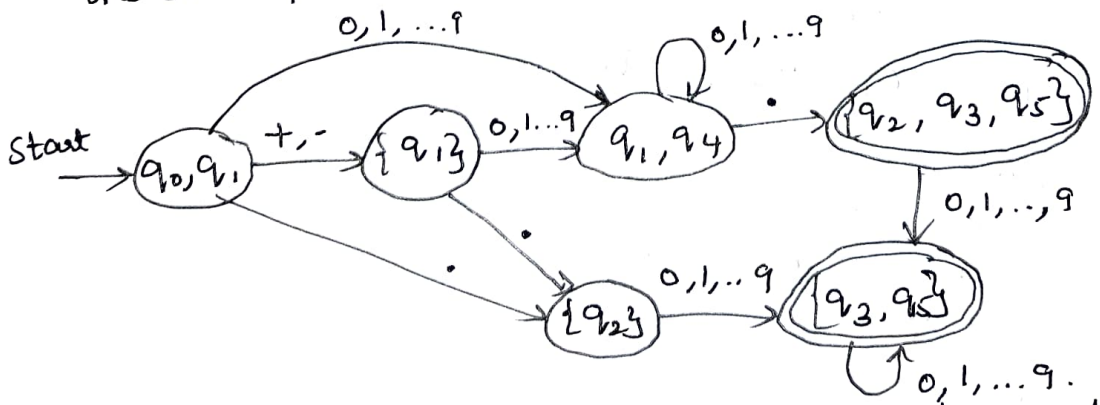


Figure: The DFA D that eliminates ϵ-transitions.

~~Statt~~ state of E is q_0.

$\therefore$ start state of D is $ECLOSE(q_0) = \{q_0, q_1\}$.

① Compute $\delta_D(\{q_0, q_1\}, +) = \{q_1\}$.

III^{ly} $\delta_D(\{q_0, q_1\}, -) = \{q_1\}$.

② Compute $\delta_D(\{q_0, q_1\}, \cdot) = \{q_2\}$.

$\delta_D(\{q_0, q_1\}, 0) = \{q_1, q_4\}$.

Since q_5 is the only accepting state

$\therefore$ two sets $\{q_3, q_5\}$ and $\{q_2, q_3, q_5\}$.

indicated by double circles.

<u>Theorem</u> :- A Language L is accepted by some ϵ-NFA if and only if L is accepted by some DFA.

<u>Proof</u> :- (If) Suppose $L = L(D)$ for some DFA.

Turn D into an ϵ-DFA E by adding transitions $\delta(q, \epsilon) = \phi$ for all states q of D.

Eg:- $\delta_E(q, a) = p$ into an NFA-transition to the set containing only p, i.e, $\delta_E(q, a) = \{p\}$.

Thus, the transitions of E and D are same, but E explicitly states that there <u>are no transitions</u> out of any <u>state on ϵ</u>.

(only-if) Let $E = (Q_E, \Sigma, \delta_E, q_0, F_E)$ be an ϵNFA.

Apply the modified subset construction described above to produce the DFA.

$D = (Q_D, \Sigma, \delta_D, q_D, F_D)$

Show that, $L(D) = L(E)$.

Formally, we show $\hat{\delta}_E(q_0, w) = \hat{\delta}_D(q_D, w)$ by induction on the length of w.

<u>BASIS</u> :- If $|w| = 0$, then $w = \epsilon$. W.K.T

$\hat{\delta}_E(q_0, \epsilon) = ECLOSE(q_0)$.

W.K.T $q_D = ECLOSE(q_0)$

Finally, for a DFA, W.K.T $\hat{\delta}(p, \epsilon) = p$ for any state p, $\hat{\delta}_D(q_0, \epsilon) = ECLOSE(q_0)$.

We have thus proved that $\hat{\delta}_E(q_0, \epsilon) = \hat{\delta}_D(q_D, \epsilon)$

INDUCTION :- Suppose $w = xa$, where a is final symbol of w, and Assume that statement holds for x.

i.e, $\hat{\delta}_E(q_0, x) = \hat{\delta}_D(q_D, x)$.

Let both these sets of states be $\{P_1, P_2, \cdots P_k\}$.

By the definition of $\hat{\delta}$ for ϵ-NFA's, we compute $\hat{\delta}_E(q_0, w)$ by :

1). Let $\{r_1, r_2, \ldots, r_m\}$ be $\bigcup_{i=1}^{k} \delta_E(P_i, a)$,

2) Then $\hat{\delta}_E(q_0, w) = \bigcup_{j=1}^{m} ECLOSE(r_j)$.

hence proved that $\hat{\delta}_E(q_0, w) = \hat{\delta}_D(q_D, w)$.

NOTE :-

Procedure to Convert the ϵ-NFA to DFA :-

Let $A_E = (Q_E, \Sigma, \delta_E, q_0, F_E)$ be an ϵ-NFA which accepts the language $L(A_E)$. There is an equivalent DFA $A_D = (Q_D, \Sigma, \delta_D, q_D, F_D)$ such that $L(A_D) = L(A_E)$.

Step 1 :- If q_0 is the start state of ϵ-NFA then ϵ-CLOSE(q_0) is the start state of DFA, so add ϵ-CLOSE(q_0) to Q_D.

Step 2 :- For a state $[q_1 q_2 \ldots q_i] \in Q_D$, the transitions for each i/p symbol in Σ can be obtained as shown below.

(1) $\delta_D([q_1 q_2 \ldots q_i], a) = E\text{-}CLOSE[\delta_E(q_1, a) \cup \delta_E(q_2, a) \cup \ldots \cup \delta_E(q_i, a)]$

$= [q_1 q_2 q_3 \ldots q_3]$ is an single state of DFA.

(2) add the state $[q_1, q_2 \ldots q_n]$ to Q_D if not present in Q_D.

repeat step 2 for each new states added to Q_D.

<u>Step 3</u> :- If $[q_1 q_2 q_3 \ldots q_n]$ is a state in a DFA then it will be a final state iff atleast one of $q_1, q_2 \ldots q_n$ belongs to the final states of ϵ-NFA.

<u>Problems</u> :-

(1). Convert the following ϵ-NFA to DFA.

Soln:- $\epsilon CLOSE(q_0) = \{q_0, q_1\} \longrightarrow$ (A)

$\delta_D(A,0) = \epsilon CLOSE(\delta_E(A,0))$
$= \epsilon CLOSE(q_1, q_0, q_2)$
$= \{q_1, q_0, q_2\} \longrightarrow$ (B)

$\delta_D(A,1) = \epsilon CLOSE(\delta_E(A,1))$
$= \epsilon CLOSE(q_1, q_2)$
$= \{q_1, q_2\} \rightarrow$ (C)

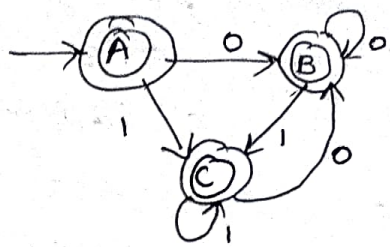
$\delta_D(B,0) = \epsilon CLOSE(\delta_E(B,0))$
$= \epsilon CLOSE(q_0, q_2, q_1)$
$= \{q_0, q_1, q_2\} \longrightarrow$ (B)

$\delta_D(B,1) = \epsilon CLOSE(\delta_E(B,1))$
$= \epsilon CLOSE(q_1, q_2)$
$= \{q_1, q_2\} \rightarrow$ (C).

$\delta_D(C,0) = \epsilon CLOSE(\delta_E(C,0))$
$= \epsilon CLOSE(q_0, q_1, q_2)$
$= \{q_0, q_1, q_2\} \longrightarrow$ (B)

$\delta_D(C,1) = \epsilon CLOSE(\delta_E(C,1))$
$= \epsilon CLOSE(q_1, q_2)$
$= \{q_1, q_2\} \rightarrow$ (C).

δ_D	0	1
→ * A	B	C
* B	B	C
* C	B	C



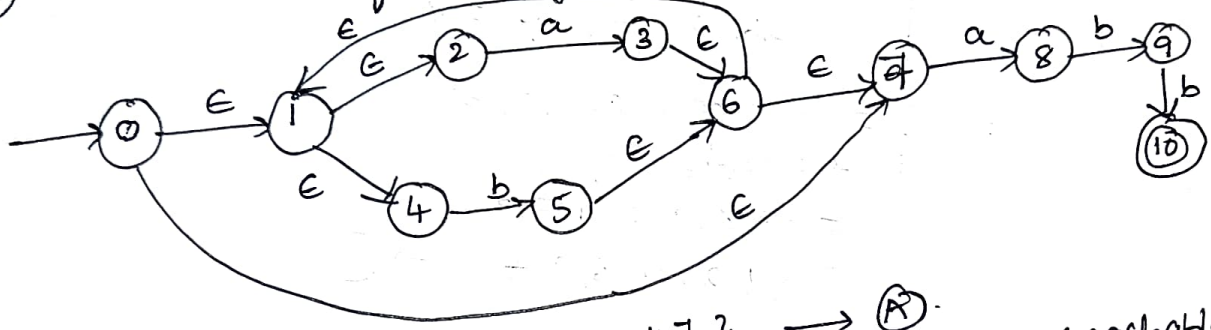
$A_D = (Q_D, \Sigma, \delta_D, q_D, F_D)$.

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

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

$\Sigma = \{0, 1\}$ $\quad q_0 = \{A\}$ $\quad F_D = \{A, B, C\}$.

(2) Convert the following ε-NFA to DFA.



Soln:- $\text{ECLOSE}(0) = \{0, 1, 2, 4, 7\} \rightarrow$ (A).
(select all the states that are directly reachable from state '0').

$\delta_D(A, a) = \text{ECLOSE}(\delta_E(A, a))$
$= \text{ECLOSE}(3, 8)$
$= \{3, 6, 7, 1, 2, 4, 8\} = \{1, 2, 3, 4, 6, 7, 8\} \rightarrow$ (B)

$\delta_D(A, b) = \text{ECLOSE}(\delta_E(A, b))$
$= \text{ECLOSE}(5)$
$= \{5, 6, 7, 1, 4, 2\} = \{1, 2, 4, 5, 6, 7\} \rightarrow$ (C).

$\delta_D(B, a) = \text{ECLOSE}(\delta_E(B, a))$
$= \text{ECLOSE}(3, 8)$
$= \{3, 6, 7, 1, 2, 4, 8\} = \{1, 2, 3, 4, 6, 7, 8\} \rightarrow$ (B)

$\delta_D(B, b) = \text{ECLOSE}(\delta_E(B, b))$
$= \text{ECLOSE}(5, 9)$
$= \{5, 6, 7, 1, 2, 4, 9\} = \{1, 2, 4, 5, 6, 7, 9\} \rightarrow$ (D)

$\delta_D(C, a) = ECLOSE(\delta_E(C, a))$
$= ECLOSE(3, 8) = \{3, 6, 1, 7, 2, 4, 8\}$
$= \{1, 2, 3, 4, 6, 7, 8\} \rightarrow B$

$\delta_D(C, b) = ECLOSE(\delta_E(C, b))$
$= ECLOSE(5) = \{5, 6, 7, 1, 2, 4\}$
$= \{1, 2, 4, 5, 6, 7\} \rightarrow C$

$\delta_D(D, a) = ECLOSE(\delta_E(D, a))$
$= ECLOSE(3, 8)$
$= \{3, 6, 7, 1, 2, 4, 8\}$
$= \{1, 2, 3, 4, 6, 7, 8\} \rightarrow B$

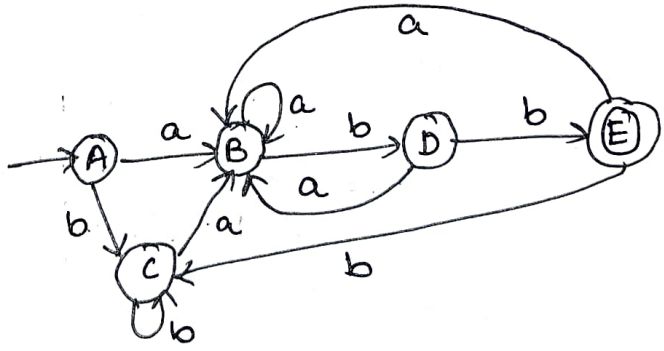
$\delta_D(D, b) = ECLOSE(\delta_E(D, b))$
$= ECLOSE(5, 10)$
$= \{5, 6, 1, 2, 4, 7, 10\} = \{1, 2, 4, 5, 6, 7, 10\} \rightarrow E$

$\delta_D(E, a) = ECLOSE(\delta_E(E, a))$
$= ECLOSE(3, 8)$
$= \{3, 6, 1, 2, 4, 7, 8\}$
$= \{1, 2, 4, 6, 7, 8\} \rightarrow B$

$\delta_D(E, b) = ECLOSE(\delta_E(E, b))$
$= ECLOSE(5)$
$= \{5, 6, 1, 2, 4, 7\} = \{1, 2, 4, 5, 6, 7\} \rightarrow C$

transition diagram and table is shown below:-

δ_D	a	b
→ A	B	C
B	B	D
C	B	C
D	B	E
* E	B	C



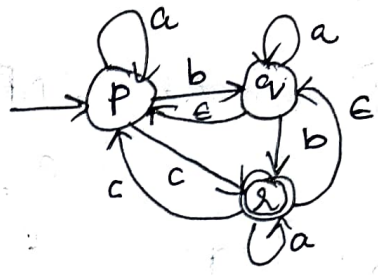
③ Consider the following transition table. Compute ECLOSE for all states..

	ϵ	a	b	c
→ p	ϕ	{p}	{q}	{r}
q	{p}	{q}	{r}	ϕ
* r	{q}	{r}	ϕ	{p}

Soln:- ECLOSE (p) = {p} → (A)

ECLOSE (q) = {q, p}.

ECLOSE (r) = {r, q, p}.



ε-NFA diagram

ECLOSE (p) = {p} → (A).

$\delta_D(A,a)$ = ECLOSE ($\delta_E(A,a)$)
= ECLOSE (p) = {p} → (A).

$\delta_D(A,b)$ = ECLOSE ($\delta_E(A,b)$)
= ECLOSE (q) = {p, q} → (B)

$\delta_D(A,c)$ = ECLOSE ($\delta_E(A,c)$)
= ECLOSE (r) = {r, q, p} → (C).

$\delta_D(B,a)$ = ECLOSE ($\delta_E(B,a)$)
= ECLOSE (p, q) = {p, q} → (B)

$\delta_D(B,b)$ = ECLOSE ($\delta_E(B,b)$)
= ECLOSE (q, r) = {p, q, r} → (C).

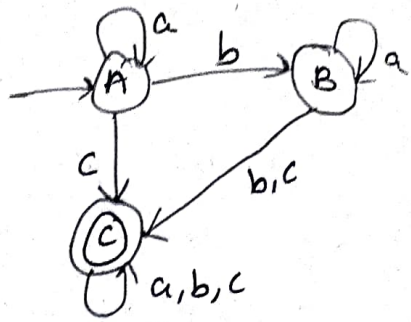
$\delta_D(B,c)$ = ECLOSE ($\delta_E(B,c)$)
= ECLOSE (r) = {r, q, p} → (C)

$\delta_D(C,a)$ = ECLOSE ($\delta_E(C,a)$)
= ECLOSE (p q r) = {p, q, r} → (C)

$\delta_D(C,b)$ = ECLOSE ($\delta_E(C,b)$)
= ECLOSE (q, r) = {q, p, r} → (C)

$\delta_D(C,c)$ = ECLOSE ($\delta_E(C,c)$)
= ECLOSE (r, p) = {r, q, p} → (C).

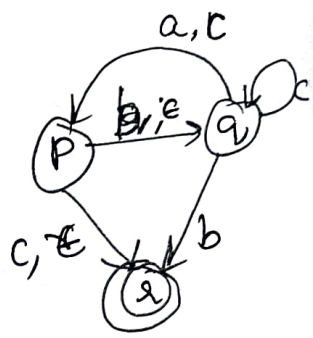
δ_D	a	b	c
→ A	A	B	C
B	B	C	C
* C	C	C	C



{c, ca, cb, cc, caa, cab ..... abb, abc, bad, bac, bba, bbb ...} ← set of strings can be obtained by above DFA.

④ Convert ENFA to DFA.

	ε	a	b	c
→ p	{q,r}	∅	{q}	{r}
q	∅	{p}	{r}	{pq}
* r	∅	∅	∅	∅



Soln:- ECLOSE (p) = {p,q,r}
ECLOSE (q) = {q}
ECLOSE (r) = {r}.

ECLOSE (p) = {pqr} → (A)

$\delta_D(A,a)$ = ECLOSE ($\delta_E(A,a)$)
= ECLOSE (p) = {p,q,r} → (A)

$\delta_D(A,b)$ = ECLOSE ($\delta_E(A,b)$)
= ECLOSE (q,r) = {q,r} → (B)

$\delta_D(A,c)$ = ECLOSE ($\delta_E(A,c)$)
= ECLOSE(r,p,q) = {r,p,q} → (A)

$\delta_D(B,a)$ = ECLOSE ($\delta_E(B,a)$)
= ECLOSE(p) = {p,q,r} → (A).

$\delta_D(B,b)$ = ECLOSE($\delta_E(B,b)$)
= ECLOSE(r) = {r} → (C)

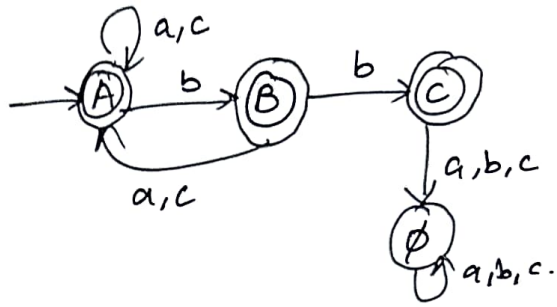
$\delta_D(B,c)$ = ECLOSE ($\delta_E(B,c)$)
= ECLOSE(p,q) = {p,q,r} → (A)

$\delta_D(C, a) = ECLOSE(\delta_E(C, a))$
$= ECLOSE(\phi)$
$= \phi$.

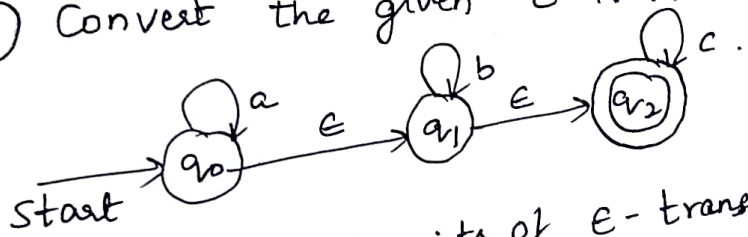
$\delta_D(C, b) = ECLOSE(\delta_E(C, b))$
$= ECLOSE(\phi)$
$= \phi$

$\delta_D(C, c) = ECLOSE(\delta_E(C, c))$
$= ECLOSE(\phi)$.
$= \phi$.

δ_D	a	b	c
→ * A	A	B	A
* B	A	C	A
* C	ϕ	ϕ	ϕ



(5) Convert the given ϵ-NFA to DFA.



Soln:- ϵ-NFA consists of ϵ-transitions (i.e, without reading an i/p symbol, it can change the state from one state to another state. i.e, from q_0 state we can reach state q_1 without reading an input symbol.)

Step 1:- Construct ϵ-NFA transition table from the above state diagram given above.

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

Step 2:- Then find the ϵ-closure of all the states.

ϵ-closure $(q_0) = \{q_0$

ϵ-closure of a state represents, the set of all states that are reachable from the particular state by following only ϵ-transitions. (i.e, if we want to find ϵ-closure (q_0), firstly include that state only