Chapter 1

Introduction to Finite Automata

What are we studying in this chapter . . .

- *Introduction*
- *Central concepts of Automata Theory*
- *Deterministic Finite Automata*
- *Non-deterministic Finite Automata*

1.1. Introduction

Before understanding the various terminologies and notations, let us first know some simple but very important mathematical notations used in set theory. First, let us see "What is a set?" Give the example.

❖ **Definition:** A *set* is a collection of distinct elements. All the elements of the set should be enclosed between '{' and '}' separated by commas.

Example 1: The set of positive integers greater than 0 and less than or equal to 25 which are divisible by 5 can be represented as

$S = \{5, 10, 15, 20, 25\}$

The elements of a set can be in any order. The above set can also be written as

$S = \{10, 20, 5, 25, 15\}$

Note: Observe that no elements are repeated in this set. This representation is useful if the number of elements is less. The above set can also be represented by describing the properties of the elements as shown below:

Example 2: The set of integers greater than 0, less than or equal to 25 and which are divisible by 5 can be represented as

$S = \{5x \mid x \text{ is a positive integer where } 0 < x \leq 5\}$

Note: If x is an element in the set S, we write $x \in S$. If x is not an element in the set S, we write $x \notin S$. The sets are denoted by capital letters such as A, B, C,... etc. and the elements within the set are denoted by lower case letters such as a, b, c, etc.

Now, let us see "What is an empty set?".

◆ **Definition:** A set which has no elements is called an *empty set* or *null set* and is denoted by $\{\}$ or ϕ. For example, the set S that does not contain any element can be represented as shown below:

$S = \{\}$ or $S = \phi$

Now, let us see "What is a subset?".

◆ **Definition:** A *set* A is a subset of set B if every element of set A is an element of set B and is denoted by

$A \subseteq B$

If $A \subseteq B$ and B contain an element which is not in A, then A is a proper subset of B and is denoted by $A \subset B$.

Now, let us see "When we say that two sets are equal?".

◆ **Definition:** The two sets A and B are same (i.e., A = B) iff $A \subseteq B$ and $B \subseteq A$ i.e., every element of set A is an element of set B and every element of set B is also an element of set A. For example, if

$A = \{a, b, c\}$ and $B = \{a, b, c\}$ then set A = B.

Now, let us see "What is a power set?".

◆ **Definition:** Let A be the set. The set of all subsets of set A is called *power set* of A and is denoted by 2^A.

Example: Let $A = \{1, 2, 3\}$. The subsets of the set A are shown below:

$\{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{\}$

The set of these subsets is called ***power set*** and is denoted by 2^A.

i.e., $2^A = \{\{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{\}\}$

Note: $|A|$ denotes the number of elements in set A and $2^{|A|}$ denote the number of subsets of set A. In the above example, $|A| = 3$ i.e., the number of items in A and $2^{|A|} = 8$ i.e., the number of items in 2^A.

Now, let us see "What is Cartesian product (cross product) of two sets?".

◆ **Definition:** The Cartesian product of A and B is given by $A \times B = \{(a, b) \mid a \in A \text{ and } b \in B\}$. Here, (a, b) is an ordered pair such that 'a' is an element of set A and 'b' is an element of set B.

Example: Let $A = \{a, b, c\}$, $B = \{0, 1\}$. The cross product of A and B is given by

$A \times B = \{(a,0), (a,1), (b,0), (b,1), (c,0), (c,1)\}$

Now, let us see "What is union of two sets and intersection of two sets?".

◆ **Definition:** The union of two sets A and B is given by $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$.

Example: Let $A = \{a, b, c\}$ $B = \{0,1\}$. Union of A and B is given by

$A \cup B = \{a, b, c, 0, 1\}$

◆ **Definition:** The intersection of two sets A and B is given by

$A \cap B = \{x \mid x \in A \text{ and } x \in B\}$

which is the collection of common elements in both the sets A and B.

Example 1: Let $A = \{a, b, c\}$ $B = \{c, d, e\}$. The intersection of A and B is given by

$A \cap B = \{\{a, b, c\} \cap \{c, d, e\}\}$
$= \{c\}$

Note: If two sets A and B have no common elements then the two sets are called Disjoint Sets.

Example 2: Let A = {a, b, c} B ={0,1}. The intersection of A and B is given by

$A \cap B = \{ \{ a, b, c \} \cap \{ 0,1 \} \}$

$A \cap B = \phi$

Now, let us see "What is the difference of two sets? What is complement of a set?".

❖ **Definition:** The difference of two sets A and B is given by

$$A - B = \{x \mid x \in A \text{ and } x \notin B\}$$

Example: Let A = {a, b, c, d} B = {a, d}. Obtain A – B.

$A - B = \{ \{ a, b, c, d \} - \{ a, d \} \} = \{b, c\}$

❖ **Definition:** The complement of A is denoted by $\overline{A}$ and is defined as a set containing every thing that is not in A. Formally, complement of A is defined as follows:

$\overline{A} = U - A$ where U is the universal set. i.e.,

$\overline{A} = \{x \mid x \in U \text{ and } x \notin A\}$

Example: Let U = { 1,2,3,4,5,6 } A = { 1,2 }. Obtain $\overline{A}$

$\overline{A} = U - A$
$= \{1, 2, 3, 4, 5, 6\} - \{1, 2\}$
$= \{3, 4, 5, 6\}$

1.2. Basic Notations and Terminologies Used in FAFL

Before we see the definition of Finite Automata and Formal Languages (**FAFL**), let us have familiarity with basic notations and terminologies used in this book.

1.2.1. Alphabet

First, let us "Define an alphabet with example".

❖ **Definition:** A language consists of various symbols from which the words, statements etc can be obtained. These symbols are called alphabets. Formally, an alphabet is defined as a finite non-empty set of symbols. The symbol Σ denotes the set of alphabets of a language.

Example 1: The alphabets of C language has the letters from A to Z, a to z, digits from 0 to 9, symbols such as + , – , * , / , (,) , { , } , etc. and is denoted by

Σ = {a, b,….z, A, B,…Z, 0, 1,….9, #, (,), {, }, <, >, !, [,],……}

Example 2: The machine Language is made up of only 0's and 1's and so, the alphabets of machine language is denoted by

Σ = {0, 1}

1.2.2. String

Now, let us see "What is a string? Explain with example".

❖ **Definition**: The sequence of symbols obtained from the alphabets of a language is called a string. Formally, a *string* is defined as a finite sequence of symbols from the alphabet Σ. **Note:** An empty string is denoted by the symbol ε (pronounced as epsilon) or λ (pronounced as lambda) and note that $\varepsilon \notin \Sigma$ i.e., ε is not part of Σ.

Note: Let us use the symbol ε indicating an empty string instead of the symbol λ.

Example 1: Let Σ = {0, 1} is set of alphabets. The various strings that can be obtained from Σ are

{0, 1, 00, 01, 10, 11, 010101, 1010, …}

Note: Note that an infinite number of strings can be generated from Σ and once the string is generated, it has finite number of symbols in it and has a definite sequence.

Notations used: Normal notations used in this subject are shown below:

- The symbol ε is used to denote an empty string.
- The lowercase letters a, b, c, etc along with the symbols such as +, –, (,), {, and so on are used to denote the symbols in Σ.
- The lowercase letters such as u, v, w, x, y z are normally used to indicate the strings. For example, we can write

  w = 010101

where the symbols 0 and 1 are in Σ and the letter *w* denotes the string with a specific value.

Now, let us see "What is concatenation of two strings?".

❖ **Definition**: The concatenation of two strings *u* and *v* is the string obtained by writing the letters of string *u* followed by the letters of string *v* (i.e., appending the symbols of *v* to the right of *u*) i.e., if

$$u = a_1a_2a_3.....a_n$$

and

$$v = b_1b_2b_3.....b_m$$

then the concatenation of *u* and *v* is denoted by

$$uv = a_1a_2a_3.....a_nb_1b_2b_3.....b_m$$

▮ **Example 1:** Let the two strings *u* and *v* be

$$u = \text{Computer}$$

and

$$v = \text{Science}$$

The concatenation of *u* and *v* denoted by *uv* will be

$$uv = \text{ComputerScience}$$

Note: Concatenation is not commutative. For example, let

$$u = \text{“RAMA”}\ v = \text{“KRISHNA”}$$

Then uv = “RAMAKRISHNA” and vu = “KRISHNARAMA”. So,

$$uv \neq vu$$

Now, let us see “What is sub string? What is suffix? What is prefix?” Give example.

❖ **Definition:** Let *w* is a string obtained from the symbols in Σ. The string *w* if it can be decomposed into three strings *x*, *y* and *z* such that

$$w = xyz$$

then *x* is a sub string, y is a substring and z is a substring of string *w*.

❖ **Definition:** A *prefix* is string of any number of leading symbols.

▮ **Example 1:** Let w is the string and let w = xyz. The string *w* has prefix ε (empty string), *x*, *xy*, and *xyz*. In the string “Rama”, the various prefixes are ε, “R”, “Ra”, “Ram” and “Rama”.

❖ **Definition:** The *suffix* is a string of any number of trailing symbols.

▮ **Example 1:** If w = xyz, then the string w has suffix ε (empty string), z, yz, and *xyz*. In the string “Rama”, the strings ε, “a”, “ma”, “ama” and “Rama” are all the suffixes.

Now, let us see “What is reversal of a string?”.

Definition: The *reversal* of a string is obtained by writing the symbols in reverse order i.e., if

$$u = a_1a_2a_3.....a_n$$

then the reverse of *u* is denoted by u^R and is given by

$$u^R = a_na_{n-1}a_{n-2}....a_3a_2a_1$$

So, if *u* is an empty string denoted by ε and *u* has only one symbol then,

$$\varepsilon^R = \varepsilon$$
$$a^R = a$$

The reverse of a string can be defined recursively as follows:

❖ **Definition**: If *a* is the symbol and *w* is the string derived from the alphabet Σ, the reverse of a string can be defined as

$$w^R = \begin{cases} \varepsilon & \text{if } w = \varepsilon \\ a & \text{if } w = a \\ (xa)^R = ax^R & \text{Otherwise (i.e., if } w=xa) \end{cases}$$

Note: x^R in the definition indicates the reversed string of string x.

Now, let us see “What is the length of a string?”.

❖ **Definition**: The *length* of a string *u* is the number of symbols in *u* and is denoted by |u| i.e., if

$$u = a_1a_2a_3.....a_n$$

then the length *u* is given by

$$|u| = n$$

The length of an empty string ε is 0 and is denoted by

$$|\varepsilon| = 0$$

Note: εw = wε = w.

Now, let us see “What is the power of an alphabet?”.

❖ **Definition:** The power of an alphabet denoted by Σ^i is the set of words of length *i*. For example, if $\Sigma = \{0, 1\}$, then

$\Sigma^0 = \{\varepsilon\}$ denote the set of words of length 0. Here, ε represents an empty string;
$\Sigma^1 = \{0, 1\}$ denote the set of words of length 1;
$\Sigma^2 = \{00,01,10,11\}$ denote the set of words of length 2;
$\Sigma^3 = \{000,001,010,100,011,101,110,111\}$ denote the set of words of length 3

and so on.

Now, let us see "What are the two variations of power of an alphabet?" The variations of power of an alphabet are shown below:

Variations of power of an alphabet → Kleene Closure (Kleene star)
Variations of power of an alphabet → Kleene plus

Now, let us see "What is Kleene closure(or Kleene star/star operator)? Give example".

❖ **Definition:** The Kleene closure is defined as follows:

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

which is the set of words of any length (possibly ε i.e., the null string). Each string is made up of symbols only from Σ.

▮ **Example 1:** Let $\Sigma = \{0, 1\}$. Then Σ^* is obtained as shown below:

$\Sigma^0 = \{\varepsilon\}$ set of words of length 0
$\Sigma^1 = \{0, 1\}$ set of words of length 1
$\Sigma^2 = \{00,01,10,11\}$ set of words of length 2
$\Sigma^3 = \{000,001,010,100,011,101,110,111\}$ set of words of length 3
.....
.....
So, $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots$
$\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$
$\{0,1\}^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$

which is the set of strings of 0's and 1's of any length.

▮ **Example 2:** Kleene star can be applied to set of strings. {"a", "bc"}* is set of strings. The set of strings consisting of a's and bc's of any length can be obtained as shown below:

$\Sigma^0 = \{\varepsilon\}$
$\Sigma^1 = \{$"a", "bc"$\}$

$\Sigma^2 = \{$"aa", "abc", "bcbc", "bca"$\}$ i.e., string made up of a's and bc's.
.....
.....
So, $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots.$
$\Sigma^* = \{\varepsilon$, "a", "bc", "aa", "abc", "bcbc", "bca"...$\}$

which is the set of strings of a's and bc's of any length.

Now, let us see "What is Kleene plus? Give example".

❖ **Definition**: The Kleene plus is a variation of Kleene star operator. The Kleene plus denoted by Σ^+ is defined as follows:

$$\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots$$

which is the set of words of any length except the null string i.e., ε (Σ^0) is not part of Σ^+ and hence $\varepsilon \notin \Sigma^+$.

▮ **Example 1:** Let $\Sigma = \{0, 1\}$. Then Σ^+ is shown below:

$\Sigma^+ = \{0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$ is the set of strings of 0's and 1's of any length except the null string.

Note: $\Sigma^* = \Sigma^+ + \varepsilon$. This can be written as $\Sigma^+ = \Sigma^* - \varepsilon$ (See the above two examples for clarity).

Now, let us see "What is the length of the string? Give the recursive definition". The length of a string can be defined recursively as follows:

❖ **Definition**: If *a* is the symbol and *w* is the string derived from the alphabet Σ, the length of a string can be defined as

$$|w| = \begin{cases} 0 & \text{if } w = \varepsilon \\ 1 & \text{if } w = a \\ |ua| = |u|+1 & \text{if } w = ua \end{cases}$$

for each $a \in \Sigma$ (i.e., a is a symbol in Σ) and each $u, w \in \Sigma^*$ (i.e., u and w are strings).

1.2.3. Language

Now, let us see "What is a language? Give example".

❖ **Definition**: A *language* can be defined as a set of strings obtained from Σ^* where Σ is se alphabets of a particular language. In other words, a language is subset of Σ^* which is denote $L \subseteq \Sigma^*$. For example,

- A language of strings consisting of equal number of 0's and 1's can be represented as {ε, 01, 10, 0011,1010,0101,0011,....}
- The language of strings consisting of n number of 0's followed by n number of 1's can be represe using the set as shown below:
  {ε, 01, 0011, 000111,.....}
- A language containing empty string ε is denoted by {ε}
- An empty language is denoted by ϕ.

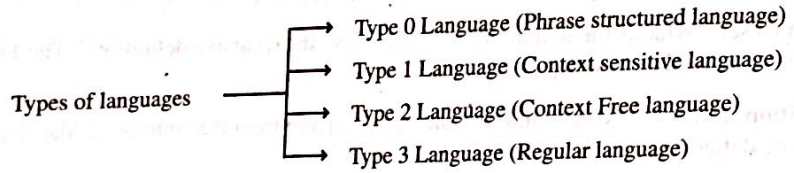
Now, let us see "What is a sentence? Give example".

❖ **Definition:** A string that belongs to a language is called word or sentence of that langua For example, a language of strings consisting of equal number of 0's and 1's can be represen as shown below:

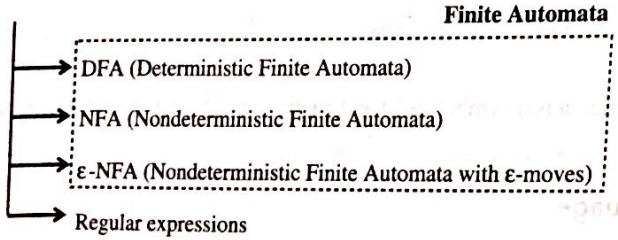
{ε, 01, 10, 0011,1010,0101,0011,....}

Each word separated by comma is a sentence (also called word). So, 01 is a sentence, 10 i sentence, 0011 is a sentence and so on.

Now, let us see "What are the different types of languages?" The languages are classified shown below:



In this chapter let us discuss about Regular languages and how they are accepted. Now, let see "What are the different ways of describing the regular languages?" The regular languages c be described using four different methods:



Now, the question is "What are regular languages?" The regular languages are defined as the languages accepted by DFA's, NFA's and ε-NFA's. They are also the languages defined by regular expressions.

In the subsequent sections, let us discuss how the regular languages are accepted by various types of finite automata and how to design various types of finite automata.

1.3. Finite Automata

The concept of an algorithm is one of the basic concepts in mathematics. We know that an algorithm is defined as unambiguous, step by step procedure (instructions) to solve a given problem in a finite number of steps by accepting a set of inputs and producing the desired output. The execution of an algorithm is carried out automatically by a computer (**Note:** Since a computer is a machine, we use the word machine in place of computer). In this subject, we use only abstract machines to execute our programs. So, first let us see "What is an abstract machine?".

Note: In English, an abstract is a brief summary of a research article, thesis or a document. In mathematics, abstract is nothing but a theoretical concept without thinking of a specific example.

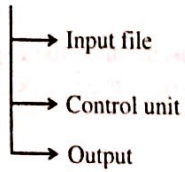
❖ **Definition:** An abstract machine (also called abstract computer) is a conceptual or theoretical model of a computer hardware or software system which really does not exist. These machines are not actual machines and hence, they are also called hypothetical computers. These machines have commonly encountered hardware features and concepts and avoids most of the details that are often found in real machines. The various types of abstract machines (or abstract computers) are:

- Finite automata
- Linear bounded automata
- Push down automata
- Turing machine

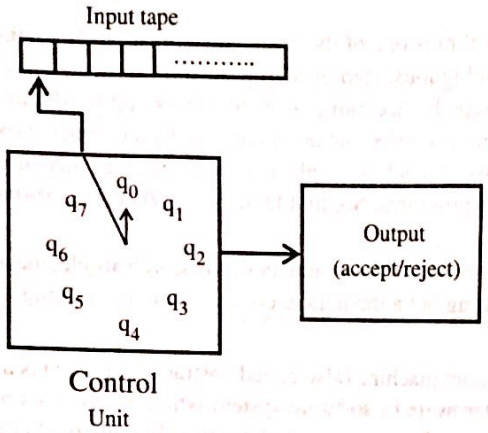
Now, let us "Define finite automata".

❖ **Definition:** A finite automaton is a mathematical model which is used to study the *abstract machines* or *abstract computing devices* with the inputs chosen from Σ. Here, Σ stands for set alphabets using which any string can be obtained. On reading the string, the machine may accept the string or reject the string. Using this abstract model, the behavior of the actual system can

be understood and build to perform various activities. Finite automaton is an abstract model of digital computer which has three components as shown below:



The pictorial representation (block diagram) of FA is shown below:

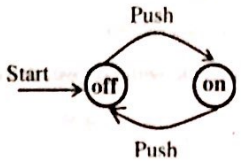


- Input tape: The input tape is divided into cells each of which can hold one symbol. The string to be processed is stored in these cells.
- Control Unit: The machine has some states one of which is the start state designated as q_0 and at least one final state. Apart from these, it has some finite states designated by q_1, q_2 and so on. Based on the current input symbol, the state of the machine can change.
- Output: Output may be *accept* or *reject*. When end of input is encountered, the control unit may be in *accept* or *reject* state.

Working: The finite automaton works as shown below:

- The machine is assumed to be in start state q_0.
- The input pointer points to the first cell of the tape pointing to the string to be processed.
- After scanning the current input symbol, the machine can enter into any of the states q_0, q_1, q_2 and so on and the input pointer automatically points to the next character by moving one cell towards right.
- When the end of the string is encountered, the string is accepted if and only if the automaton will be in one of the final states. Otherwise, the string is rejected.

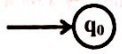
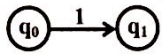
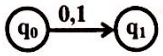
For example, consider an electric switch which has only two states "OFF" and "ON". To start with, the switch will be in OFF state. When we push the button, it goes to ON state. If we push once again it goes to OFF state. This can be represented as shown below:



Note: The circles represent the states and the labels associated with arcs represent the input given to go to another state. The arrow with the label Start (not originating from any state) is considered as the start state.

Symbols Used in FA

Now, let us see "What are the various notations used during designing of FA?".

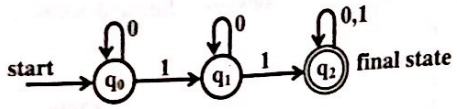
Symbol	Meaning
	A circle is used to represent a state. Here, q_0 is a state of the machine.
	A circle with an arrow which is not originating from any node represents the start state of machine.
	Two circles are used to represent a final state. Here, q_0 is the final state.
	An arrow with label 1 goes from state q_0 to state q_1. This indicates there is a transition (change of state) from state q_0 on input symbol 1 to state q_1. This is represented as: $\delta(q_0, 1) = q_1$
	An arrow with label 0 starts and ends in q_0. This indicates the machine in state q_0 on reading a 0, remains in same state q_0. This is represented as: $\delta(q_0, 0) = q_0$
	An arrow with label 0,1 goes from state q_0 to state q_1. This indicates that the machine in state q_0 on reading a 0 or a 1 enters into state q_1. This is represented as: $\delta(q_0, 0) = q_1$ $\delta(q_0, 1) = q_1$

Now, we can easily answer the question "How a FA processes the string?" The string is processed by the various types of FA as shown below:

Types of finite automata
- Deterministic finite automata (DFA)
- Non-deterministic finite automata (NFA)
- Non-deterministic finite automata with ε-moves (ε-NFA)

1.4. Deterministic Finite Automata (DFA)

Before worrying about the definition, let us consider the following pictorial representation DFA.



From the above figure, observe following components of DFA:

- **States:** The circles are called vertices or nodes or states. Each state is identified by a name i.e., q_0 and q_2. The states are classified and identified as shown below:

  Start state: q_0 with an arrow labeled start is considered as start state.

  Final state: q_2 with two concentric circles represent a final state.

  Intermediate state: q_1 is neither a start state nor a final state. It is called an intermediate state

  This DFA has three states q_0, q_1 and q_2 and can be represented as

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

  Note: A DFA can have one or more final states. If there is no final state in DFA, the DF accepts empty language.

- **Input alphabets:** Each edge is labeled with 0 or 1 and represent the input alphabets which can b denoted as:

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

- **Transitions:** Transition is nothing but change of state after consuming an input symbol. If there is change of state from q_i to q_j on an input symbol a, then we write

  $\delta(q_i, a) = q_j$

For example, in the above diagram, the various transitions shown are represented as show below:

$\delta(q_0,0) = q_0$ There is a transition from state q_0 on 0 to state q_0

$\delta(q_0,1) = q_1$ There is a transition from state q_0 on 1 to state q_1

$\delta(q_1,0) = q_1$ There is a transition from state q_1 on 0 to state q_1

$\delta(q_1,1) = q_2$ There is a transition from state q_1 on 1 to state q_2

$\delta(q_2,0) = q_2$ There is a transition from state q_2 on 0 to state q_2

$\delta(q_2,1) = q_2$ There is a transition from state q_2 on 2 to state q_2

δ:(Q x Σ) to Q

which is the cross product of $Q = \{q_0, q_1, q_2\}$ and $\Sigma = \{0,1\}$

Now, let us see "What is a transition function?" The transition function δ is defined as:

δ: Q X Σ to Q

which is read as "δ is a transition function which maps Q X Σ to Q". For example, the change of state from state q on input symbol a to state p is denoted by

$\delta(q, a) = p$

where

- δ: is a function called transition function.
- q : is the first parameter representing the current state of the machine.
- a : is the second parameter representing the current input symbol read.
- p : is the next state of machine which is returned by the transition function.

Note: In other words, the transition function δ accepts two parameters namely state q and input symbol a and returns a next state p:

- Start state (q_0): q_0 with the label start is treated as the start state.
- Final state (q_2): q_2 with two concentric circles is treated as the final state.

Note: From this discussion, it is observed that the DFA has five components:

(states, input alphabets, transitions, start state, final states)

Q Σ δ q0 F

With this concept, now let us see "What is a deterministic finite automaton (DFA)?".

❖ **Definition:** The Deterministic Finite Automaton in short DFA is 5-tuple or quintuple (indicating five components):

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

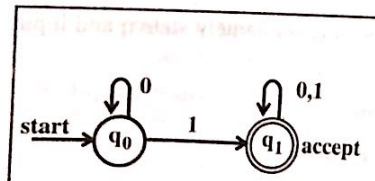
where

- M is the name of the machine. It can also be called by any name.
- Q is non-empty, finite set of states.
- Σ is non-empty, finite set of input alphabets.
- $\delta : Q \times \Sigma$ to Q i.e., δ is transition function which is a mapping from $Q \times \Sigma$ to Q. Based on the current state and input symbol, the machine enters into another state.
- $q_0 \in Q$ is the start state.
- $F \subseteq Q$ is set of accepting or final states.

Thus, name DFA emerges from the following facts:

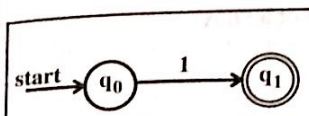
- D (Deterministic) – There is exactly one transition for every input symbol from the state. So, it is possible to determine exactly to which state the machine enters into after consuming the input symbol. So, the machine is deterministic.
- F (Finite) – Has finite number of states and arcs. So, it is deterministic and finite.
- A (Automaton) – Automaton is a machine which may accept the string or reject the string. So, it is deterministic finite automaton.

Note: The DFA can have only one transition from a state on an input symbol. Consider the following DFA and observe the various transitions:



- From state q_0 on input symbol 0, the machine will stay in q_0
- From state q_0 on input symbol 1, the machine enters into state q_1
- From state q_1 on input symbol 0, the machine stays in q_1
- From state q_1 on input symbol 1, the machine stays in q_1

Observe from the above diagram that, there is exactly one transition defined from a state on an input symbol. Sometimes, there can be no transitions from a state as shown below:



- From state q_0 on input symbol 1, the machine enters into state q_1
- But, from state q_1, the transition is not defined on any of the input symbol. We say there is a zero transition from state q_1

Note: In short, in a DFA there can be zero or one transition from a state on an input symbol and at any point of time, the DFA will be in one state.

1.5. Simple Notations for DFA's

When we design a DFA, specifying a DFA as a 5-tuple such as $M = (Q, \Sigma, \delta, q_0, F)$ along with the detailed description of each transition is very tedious and difficult to understand. So, a DFA can be specified using simpler and more effective notations. Now, let us see "What are the various simpler notations used to specify a DFA?" The two notations using which the DFA's can be easily represented are:

- Transition diagram (Transition graph)
- Transition table

1.5.1. Transition Diagram

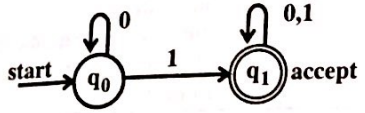
Definition: Now, let us see "What is a transition diagram?". The transition diagram for DFA

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

is defined as a graph with circles, arrows and arcs with labels, two circles etc. It is formally defined as shown below:

- Each state of Q corresponds to one node or vertex represented using a circle or two circles.
- Alphabets in Σ are represented as labels along with the edges.
- The transition from one state to another state is indicated by the directed edge. Let $\delta(q_i, a) = q_j$. This indicates that there is a directed edge from q_i to q_j and the edge is labeled a.
- The start state is a state which has an arrow not originating from any node and entering into the state. This is labeled with start.
- The final states or accepting states which are in F are represented by double circles. The states which are not in F are represented by a single circle.

For example, the following diagram represents a transition diagram of a DFA



- It has two nodes corresponding to the two states of machine q_0 and q_1.
- q_0 has a start arrow (incoming arrow not originating from any node) and hence it is the start state.
- q_1 is the accepting state and hence it is denoted by two circles.
- There are 3 arcs: the first one with label 0 from q_0 to q_0, the second with label 1 from q_0 to q_1 and third arc with labels 0, 1 both from q_1 to q_1.

1.5.2. Transition Table

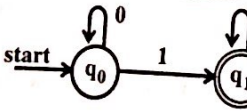
Definition: Now, let us see "What is a transition table?". The transition table for DFA

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

is defined as a conventional, tabular representation of a transition function such as δ which takes two arguments and returns a value with:

- The rows of the table correspond to the states of DFA obtained from Q.
- The columns correspond to the input symbols obtained from Σ. Note: There is one row for each state, and one column for each input.
- If q is the current state of DFA and a is the current input symbol, the value returned from $\delta(q, a)$ represent the next state of DFA and is entered in row q and column a.
- The state marked with an arrow is the start state.
- The final state is marked with a star.

For example, the transition diagram and its equivalent transition table are shown below:



Transition diagram

• The transition diagram of DFA has two states q_0 and q_1	⇨

Columns

Rows	δ	0	1
→	q_0	q_0	q_1
	$*q_1$	q_1	q_1

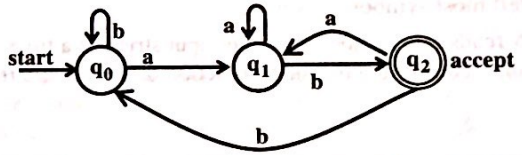
Transition table

⇨	• Represented using two rows q_0 and q_1

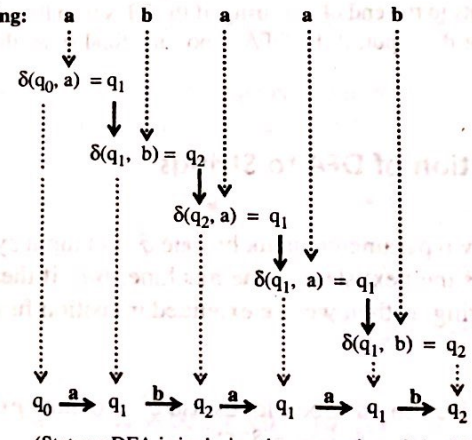
• There are two input symbols 0 and 1	⇨ • The two input symbo… 0 correspond to …wo c… m…
• Start state is represented using an arrow mark not originating from any state and labeled start	⇨ • The start stat… i… id… ifie… an arrow with d… ire… ion …
• The final states are represented by two circles.	⇨ • The final states a… rep… putting stars (*'s …oy…
• The transition from state q on input symbol a to state p is represented by a directed edge originating from state q and ending at state p with label a.	⇨ • The equivalent trans… … by writing p in row …

1.5.3. How a DFA Processes Strings

Now, let us see "What are the moves made by the following DFA whi… …oce… …ng abaab and abb?".



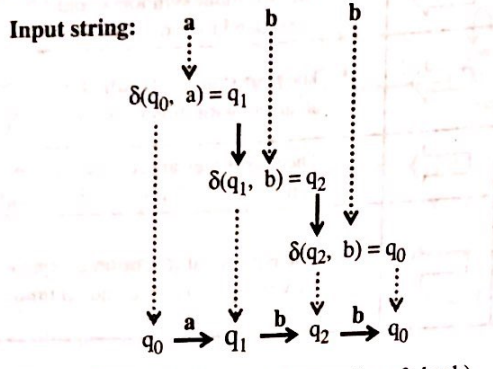
Solution: The moves made by the DFA for the input string abaab is shown …elo…



(States a DFA is in during the processing of abaab)

Note: At the e… d of t… …rin… abaab, the DFA … ll b… in sta… q_2 which is the f… al… tate… S… the string abaal… i… acc… ept… d… the machine.

The moves made by the DFA for the input string abb is shown below:



(States a DFA is in during the processing of abaab)

Note: At the end of the string abb, the DFA will be in state q_0 which is not the final state. So, the string abb is rejected by the machine.

Now, we can easily answer the question "How a DFA processes the string?" The string w processed by the DFA as shown below:

- **Initialization:** Let q_0 is the start of DFA and let *w* is the string to be processed. Initially, the inp pointer points to the left most symbol in string *w*.
- **Processing:** The DFA reads one symbol from the input string at a time. The machine in state q_0 reading the input symbol, consult the transition function δ. If there is a transition:

  $\delta(q_0,a_1) = q_1$

  then the machine changes its state to q_1 and the input pointer points to the next symbol. Now, t machine in state q_1, after reading the next input symbol may change its state to q_2 and so on.
- **Accept or reject:** When the input points to the end of the string, if the DFA is in final state, the stri *w* is accepted by DFA. After the end of the input, if the DFA is not in a final state, then the string is rejected by the DFA.

1.5.4. Extended Transition Function of DFA to Strings

Note: The transition **δ(q, a) = p** accepts two parameters namely state *q* and input symbol *a* as t parameters and returns a state *p* which is the next state of the machine. But, if there is a chang of state from state *q* to state *p* on input string *w*, then we use extended transition function denot by δ*.

Note: We can also use $\hat{\delta}$ in place of δ*. But, in our book let us use δ* to denote extended trans tion. Now, let us see "What is extended transition function δ*?".

❖ **Definition:** The extended transition function δ*describes what happens to a state of machine when the input is a string (sequence of symbols). Let M = (Q, Σ, δ, q_0, F) be a FA. The extended transition function **δ*: Q X Σ* to Q** is defined recursively as shown below:

- Basis: δ*(q, ε) = q. This indicates that if the machine is in state q and read no input, then the machine is still in state q.
- Induction: Let w = xa where a is the last symbol of w and x is the remaining string of w. Let q is the current state and w is the string to be processed and after consuming the string w, let the state of the machine is p. Then

  δ*(q, w) = δ*(q, xa) = δ(δ*(q, x), a) = p

Note: Thus, various properties of extended transition functions are:

- **δ*(q, ε) = q**
- **δ*(q, w) = δ*(q, xa) = δ(δ*(q, x), a)** where w = xa
- **δ*(q, w) = δ*(q, ax) = δ*(δ(q, a), x)** where w = ax (modification of above)

For example, change of state from state *q* on input string *w* to state *p* is denoted by:

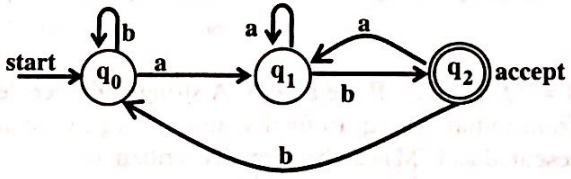
δ*(q, w) = p

where

- **δ* :** is a function called extended transition function
- **q :** is the first parameter representing the current state of the machine
- **w :** is the second parameter representing the current input string being read
- **p :** is the next state of machine which is returned by the transition function

Note: The transition function δ* accepts two parameters namely state *q* and input string *w* as the parameters and returns a state *p* which is the next state of the machine.

Now, let us see "What are the moves made by the following DFA while processing the string abaab using the extended transition function?".



Solution: The moves made by the DFA for the input string abaab using δ* is obtained starting from ε and taking prefix of abaab in increasing size as shown below:

- For prefix ε: $\delta^*(q_0, \varepsilon) = q_0$
- For prefix a: $\delta^*(q_0, a) = \delta(\delta^*(q_0, \varepsilon), a)$ Substituting (1)
  $= \delta(q_0, a)$
  $= q_1$
- For prefix ab: $\delta^*(q_0, ab) = \delta(\delta^*(q_0, a), b)$ Substituting (2)
  $= \delta(q_1, a)$
  $= q_1$
- For prefix aba: $\delta^*(q_0, aba) = \delta(\delta^*(q_0, ab), a)$ Substituting (3)
  $= \delta(q_1, a)$
  $= q_1$
- For prefix abaa: $\delta^*(q_0, abaa) = \delta(\delta^*(q_0, aba), a)$ Substituting (4)
  $= \delta(q_1, a)$
  $= q_1$
- For prefix abaab: $\delta^*(q_0, abaab) = \delta(\delta^*(q_0, abaa), b)$ Substituting (5)
  $= \delta(q_1, b)$
  $= q_2$

It is observed from (1) to (5) that the sequence of states the DFA is in during the processing string abaab is shown below:

$$q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_1 \xrightarrow{a} q_1 \xrightarrow{b} q_2$$

(States a DFA is in during the processing of abaab)

Note: After the string abaab, the machine is in state q_2 which is the final state. So, the string abaa is accepted by DFA. Now, let us see "When a language is accepted by DFA?".

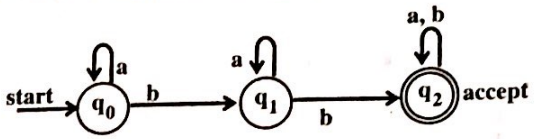
1.5.5. Language Accepted by a DFA

Now, let us "Define the language accepted by DFA" The language accepted by DFA is formally be defined as follows:

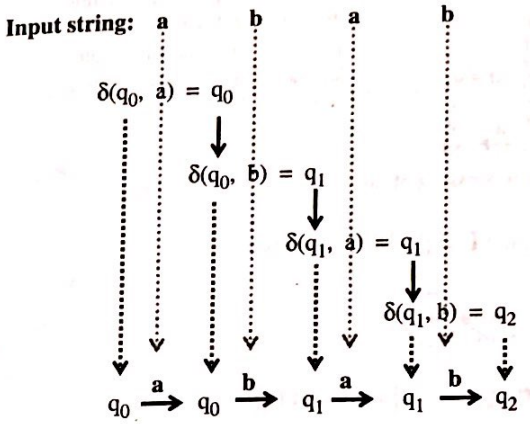
❖ **Definition**: Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA. A string w is accepted by the machine M, if it takes the machine from initial state q_0 to final state i.e., $\delta^*(q_0, w)$ is in F. Thus, the language accepted by DFA represented as L(M) can be formally written as:

$$L(M) = \{w \mid w \in \Sigma^* \text{ and } \delta^*(q_0, w) \text{ is in } F\}$$

For example, consider the following DFA with q_0 as the start state and q_2 as the final state.



Input string: Assume **abab** is the input string. The moves made by the DFA is shown below:



Note: At the end of the string abab, the DFA will be in state q_2 which is the final state. So, the string abab is accepted by the machine.

(States a DFA is in during the processing of abaab)

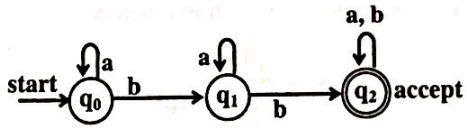
Now, let us see "What language is rejected by DFA?".

❖ **Definition:** Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA. The non-acceptance of a language indicates that after the string w is processed, the DFA will not be in the final state.

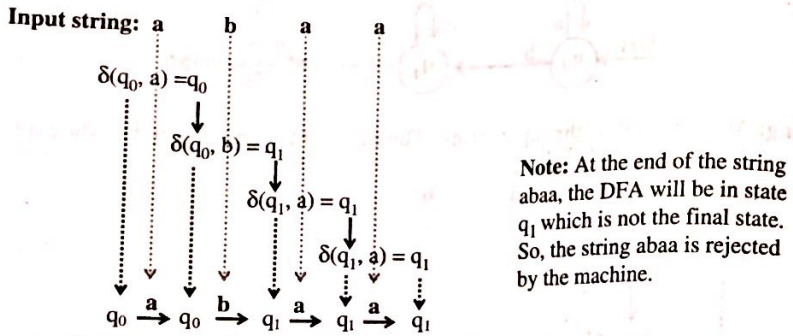
Thus, the non-acceptance of the string w by a FA or DFA can be defined as:

$$L(M) = \{w \mid w \in \Sigma^* \text{ and } \delta^*(q_0, w) \text{ is not in } F\}$$

For example, consider the following DFA with q_0 as the start state and q_2 as the final state:



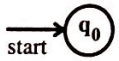
Input string: Assume **abaa** is the input string. The moves made by the DFA is shown below:



Note: At the end of the string abaa, the DFA will be in state q_1 which is not the final state. So, the string abaa is rejected by the machine.

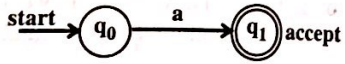
(States a DFA is in during the processing of abaa)

■ **Example 1:** DFA to accept empty language **L** = { ϕ } is shown below:

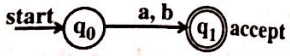


■ **Example 2:** DFA to accept an empty string **L** = { ε } is shown below:

■ **Example 3:** DFA to accept exactly one *a* is shown below:



■ **Example 4:** DFA to accept one *a* or one *b* is shown below:



■ **Example 5:** DFA to accept zero or more a's is shown below:

Now, let us see "How to construct the DFA for complex languages?" in the coming sections.

1.6. DFA Design Techniques

Now, let us see "What are the various problem types for which we can construct DFA?". There are three types of problems for which we can construct a DFA:

- Pattern recognition problems
- Divisible by k problems
- Modulo-k-counter problems

1.6.1. Pattern Recognition Problems

For the type of problems that involve pattern recognition, the DFA can be constructed very easily. Now, let us see "What are the general steps to be followed while designing the DFA for pattern recognition problems?". The various steps to be followed are:

Step 1: Identify the minimum string.

Step 2: Identify the alphabets.

Step 3: construct a skeleton DFA.

Step 4: Identify other transitions not defined in step 3 **Note:** This is difficult step. So, give more attention to this step while constructing DFA.

Step 5: Construct the DFA using transitions in step 3 and step 4.

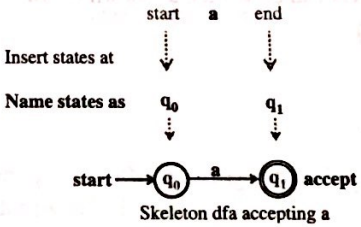
■ **Example 1:** Now, let us "Draw a DFA to accept string of a's having at least one *a*".

Solution: The DFA can be constructed as shown below:

Step 1: Identifying the minimum string: In this case, it is *a*.

Step 2: Identify the alphabets: In this case Σ = {a}.

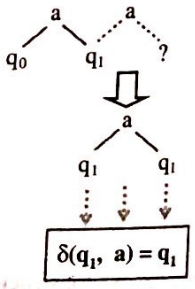
Step 3: Construct a skeleton DFA: The DFA for the string *a* identified in step 1 can be constructed as shown below:



Skeleton dfa accepting a

Step 4: Identify the transitions not defined in step 3:

step (i) : $\delta(q_1, a)$=? Move from q_0 to q_1 on a (see skeleton DFA) and then think of transition f
q_1 on a.



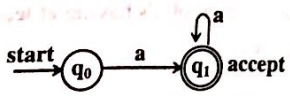
Note: The sequence *aa* should be accepted since it is having at least one *a*. (go to final state q_1)

Step 5: Construct the DFA. The DFA can be obtained by skeleton DFA and transitions obtai
from previous step. The DFA is defined as:

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

where

- $Q = \{q_0, q_1\}$
- $\Sigma = \{a\}$
- q_0 is the start state
- $F = \{q_1\}$
- δ is shown below using the transition diagram and table as shown below:



Transition diagram

δ	a
→ q_0	q_1
*q_1	q_1

Transition Table

The language to be accepted by DFA can be written as shown below:

$$L = \{ a, aa, aaa, aaaa, \ldots\}$$

or

$$L = \{ a^n | n \geq 1\}$$

or

$$L = \{w : n_a \geq 1, w \in \{a\}^*\}"$$

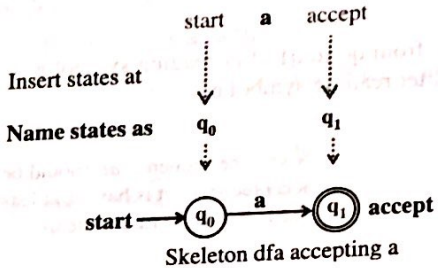
Example 2: Now, let us "Draw a DFA to accept strings of a's and b's having at least one *a*".

Solution: The DFA can be constructed as shown below:

Step 1: Identifying the minimum string: In this case it is *a*.

Step 2: Identify the alphabets: In this case $\Sigma = \{a, b\}$.

Step 3: Construct the skeleton DFA: The DFA for the string *a* identified in step 1 can be constructed as shown below:

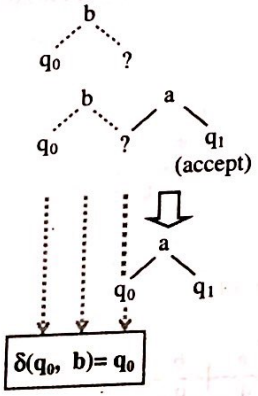


Skeleton dfa accepting a

Step 4: Identify the transitions not defined in step 3: We need to compute the following:

$\delta(q_0, b) = ?$ Step (i)
$\delta(q_1, a) = ?$ Step(ii)
$\delta(q_1, b) = ?$ Step(iii)

Step (i) : $\delta(q_0, b)$=?

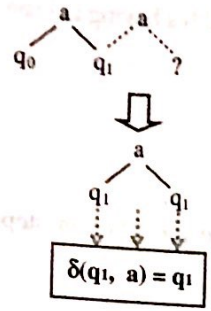


Note: The sequence *b* should not be accepted since it is not having at least one *a*.

Note: But, *b* followed by *a* resulting in string *ba* has to be accepted since it has at least one *a*. So, go to final state q1

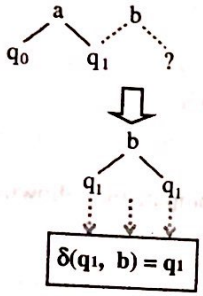
But, before the suffix *a*, the machine is in state q_0 (see skeleton DFA)

Step(ii): $\delta(q_1, a) = ?$ Move from q_0 to q_1 after reading symbol *a* (see skeleton DFA) and then think of transition from q_1 after reading symbol *a*.



Note: The sequence *aa* should be accepted since it is having at least one *a*. So, go to final state q_1.

Step(iii): δ(q1, b) = ? Move from q0 to q1 after reading symbol a (see skeleton DFA) and th think of transition from q1 after reading symbol b.



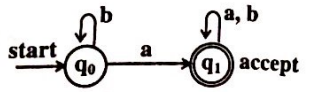
Note: The sequence *ab* should be accepted since it is having at least one *a*. So, go to final state q_1.

Step 5: Construction of DFA. The DFA can be obtained by skeleton DFA and transitions obtaine from previous step. The DFA is defined as:

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

where

- $Q = \{q_0, q_1\}$
- $\Sigma = \{a, b\}$
- q_0 is the start state
- $F = \{q_1\}$
- δ is shown below using the transition diagram and table as shown below:



$L = \{w : n_a(w)\ 1, w \in \geq \{a, b\}^*\}$

δ	a	b
→ q_0	q_1	q_0
*q_1	q_1	q_1

Note: Once the machine enters into state q_1, irrespective of any number of inputs of a's and b's, the machine remains in same state q_1 and can not come out of the state. So, the machine is trapped in state q_1 and hence it is called trap state. Since, q_1 is also a final state, the state q_1 is called trapped final state.

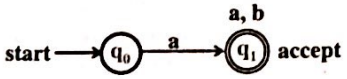
Example 3: Now, let us "Draw a DFA to accept strings of a's and b's having exactly one *a*".

Solution: The DFA can be constructed as shown below:

Step 1: Identifying the minimum string: In this case it is *a*.

Step 2: Identify the alphabets: In this case **Σ = {a, b}**.

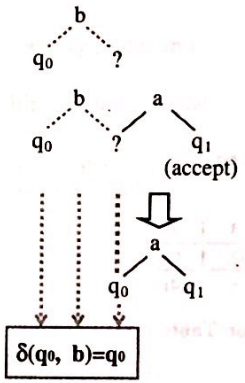
Step 3: Construct the skeleton DFA: The DFA for the string *a* identified in step 1 can constructed as shown below:



Step 4: Identify the transitions not defined in step 3: We need to compute the following:

$\delta(q_0, b) = ?$ See step (i)
$\delta(q_1, a) = ?$ See step(ii)
$\delta(q_1, b) = ?$ See step(iii)

Step (i) : $\delta(q_0, b)$=?

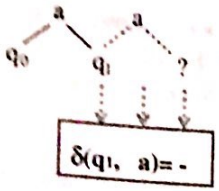


Note: The sequence *b* should not be accepted since it is not having one *a*.

Note: But, *b* followed by *a* resulting in string *ba* has to be accepted since it has one *a*. So, go to final state q1

But, before the suffix *a*, the machine is in state q_0 (see skeleton DFA)

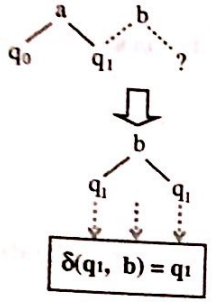
Step(ii): $\delta(q_1, a) = ?$ Move from q_0 to q_1 after reading symbol *a* (see skeleton DFA) and then think of transition from q_1 after reading symbol *a*.



Note: The sequence *aa* should not be accepted since it is not having exactly one *a*. So, the string *aa* should be rejected. For the reject state, the transition should not be defined.

Note: The symbol '-' indicates that the transition is not defined.

Step(iii): $\delta(q_1, b) = ?$ Move from q_0 to q_1 after reading symbol *a* (see skeleton DFA) and think of transition from q_1 after reading symbol *b*.

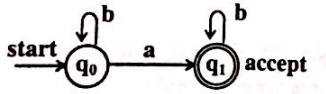


Note: The sequence *ab* should be accepted since it is having one *a*. So, go to final state q_1.

Step 5: Construction of DFA. The DFA can be obtained by skeleton DFA and transitions obtained from previous step. The DFA is defined as:

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

- $Q = \{q_0, q_1\}$
- $\Sigma = \{a, b\}$
- q_0 is the start state
- $F = \{q_1\}$
- δ is shown below using the transition diagram and table as shown below:

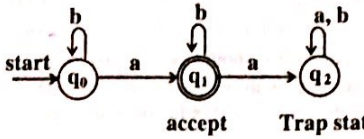


$\mathbf{L = \{w : n_a(w) = 1, w \in \{a, b\}^*\}}$

δ	a	b
→ q_0	q_1	q_0
*q_1	-	q_1

Transition Table

Note: In the above DFA, there is no transition from state q_1 for the input symbol *a*. But, we can include a transition from state q_1 for the input symbol *a* to a non final reject state as shown below:

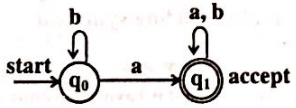


δ	a	b
→ q_0	q_1	q_0
*q_1	-	q_1
q_2	q_2	q_2

Here, state q_2 is called a trap state. Now, the question is "What is a trap state?".

❖ **Definition:** A state for which there exists transitions to itself for all the input symbols chosen from Σ is called a trap state.

For example, in the above DFA, once the machine changes its state to q_2, for any of the input symbols *a* or *b*, there is no escape from this state. The machine is trapped in this state and hence the name trap state. A trap state can be non final state (state q_2 in above figure). The non final trap state is also called dead state. A final state can also be a trap state (state q_1 shown in transition diagram below). But, it is not a dead state since the string is accepted at this state.

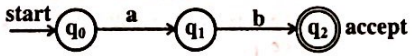


Example 4: Obtain a DFA to accept strings of a's and b's starting with the string ab.

Step 1: Identifying the minimum string: In this case it is *ab*.

Step 2: Identify the alphabets: In this case $\Sigma = \{a, b\}$.

Step 3: Construct the skeleton DFA: The DFA for the string *ab* identified in step 1 can constructed as shown below:

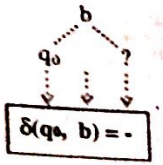


Skeleton dfa accepting a

Step 4: Identify the transitions not defined in step 3: We need to compute the following:

$\delta(q_0, b) = ?$	See step (i)	$\delta(q_2, a) = ?$	See step (iii)
$\delta(q_1, a) = ?$	See step (ii)	$\delta(q_2, b) = ?$	See step (iv)

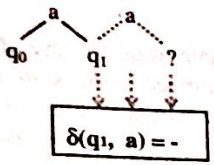
Step (i) : $\delta(q_0, b)$=?



Note: The sequence *b* should be rejected since it is not starting with substring *ab*. For the reject state, the transition should not be defined.

Note: Thesymbol '-' indicates that the transition is not defined.

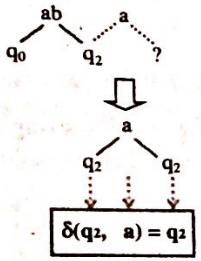
Step (ii): $\delta(q_1, a) = ?$ Move from q_0 to q_1 after reading symbol *a* (see skeleton DFA) and th think of transition from q_1 after reading symbol *a*.



Note: The sequence *aa* should not be accepted since it is not starting with*ab*. So, the string *aa* should be rejected. For the reject state, the transition should not be defined.

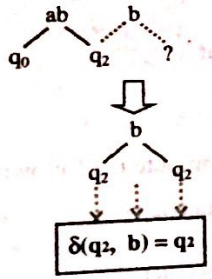
Note: Thesymbol '-' indicates that the transition is not defined.

Step (iii): $\delta(q_2, a) = ?$ Move from q_0 to q_2 after reading the sequence *ab* (see skeleton DF and then think of transition from q_2 after reading symbol *a*.



Note: The sequence *aba* should be accepted since it is having substring *ab*. So, go to final state q_2.

Step (iv): $\delta(q_2, b) = ?$ Move from q_0 to q_2 after reading the sequence *ab* (see skeleton DF and then think of transition from q_2 after reading symbol *b*.



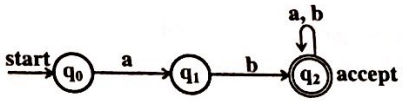
Note: The sequence *abb* should be accepted since it is having substring *ab*. So, go to final state q_2.

Step 5: Construction of DFA. The DFA can be obtained by skeleton DFA and transitions obtained from previous step. The DFA is defined as:

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

where

- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{a, b\}$
- q_0 is the start state
- $F = \{q_2\}$
- δ is shown below using the transition diagram and transition table as shown below:

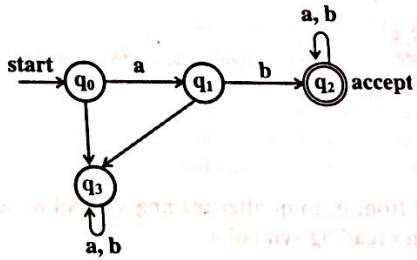


Transition diagram

δ	a	b
→ q_0	q_1	-
q_1	-	q_2
*q_2	q_2	q_2

Transition Table

Note: Since the transitions are not defined on q_0 and q_1 for the input symbol *b* and *a* respectively, we can have transitions to the trap state. So, the above DFA can also be written as shown below.

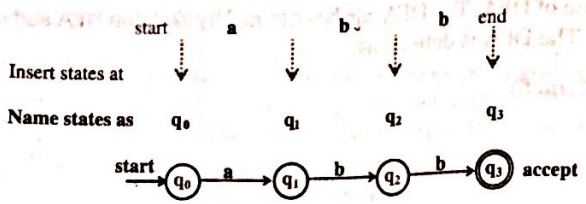


Example 5: Now, let us "Draw a DFA to accept string of a's and b's ending with the string abb".

Step 1: Identify the minimum string: In this case abb.

Step 2: Identify the alphabets. In this case $\Sigma = \{a, b\}$.

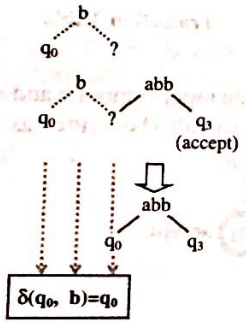
Step 3: construct a skeleton DFA: The DFA for the string *abb* identified in step 1 can constructed as shown below:



Step 4: Identify other transitions not defined in step 3: In this case, we need to compute following:

$\delta(q_0, b) = ?$ See step (i)
$\delta(q_1, a) = ?$ See step (ii)
$\delta(q_2, a) = ?$ See step (iii)
$\delta(q_3, a) = ?$ See step (iv)
$\delta(q_3, b) = ?$ See step (v)

Step (i) : $\delta(q_0, b)=?$

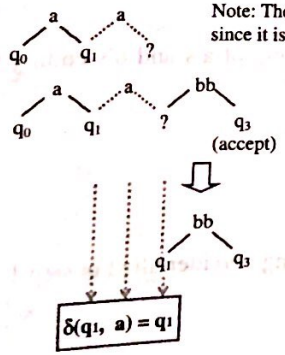


Note: The sequence *b* should not be accepted since it is not ending with *abb*.

Note: But, *b* followed by *abb* resulting in string *babb* has to be accepted since it ends with *abb*. So, go to final state q3

But, before the suffix *abb*, the machine is in state q_0 (see skeleton DFA)

Step (ii): $\delta(q_1, a) = ?$ Move from q_0 to q_1 after reading symbol *a* (see skeleton DFA) and the think of transition from q_1 after reading symbol *a*.

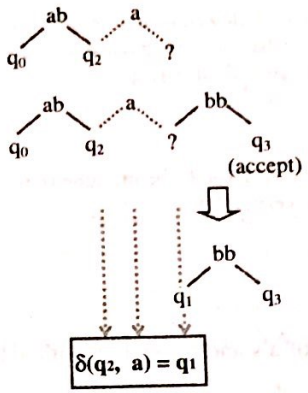


Note: The sequence *aa* should not be accepted since it is not ending with *abb*.

Note: But, *aa* followed by *bb* resulting in string *aabb* has to be accepted since it ends with *abb*. So, go to final state q_3

But, before the suffix *bb*, the machine is in state q_1 (see skeleton DFA)

Step (iii): $\delta(q_2, a) = ?$ Move from q_0 to q_2 after reading sequence *ab* (see skeleton DFA) and then think of transition from q_2 after reading symbol *a*.

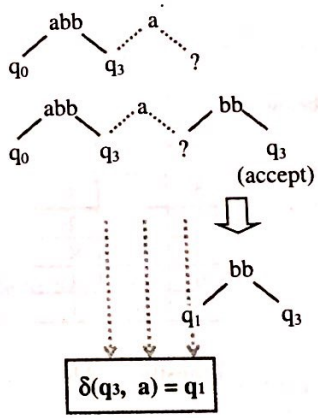


Note: The sequence *aba* should not be accepted since it is not ending with *abb*.

Note: But, *aba* followed by *bb* resulting in string *ababb* has to be accepted since itends with *abb*. So, go to final state q_3

But, before the suffix *bb*, the machine is in state q_1 (see skeleton DFA)

Step (iv): $\delta(q_3, a) = ?$ Move from q_0 to q_3 after reading sequence *abb* (see skeleton DFA) and then think of transition from q_3 after reading symbol *a*.

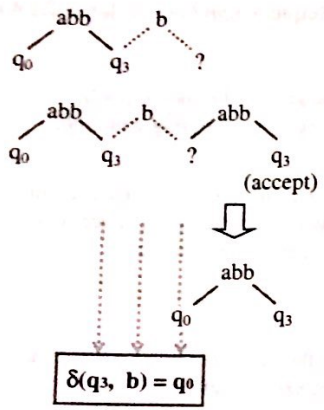


Note: The sequence *abba* should not be accepted since it is not ending with *abb*.

Note: But, *abba* followed by *bb* resulting in string *abbabb* has to be accepted since itends with *abb*. So, go to final state q_3

But, before the suffix *bb*, the machine is in state q_1 (see skeleton DFA)

Step (v): $\delta(q_3, b) = ?$ Move from q_0 to q_3 after reading sequence *abb* (see skeleton DFA) and then think of transition from q_3 after reading symbol *b*.



Note: The sequence *abbb* should not be accepted since it is not ending with *abb*.

Note: But, *abbb* followed by *abb* resulting in string *abbbabb* has to be accepted since it ends with *abb*. So, go to final state q_3

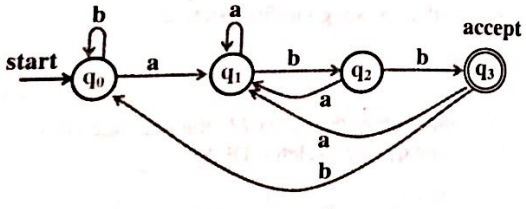
But, before the suffix *abb*, the machine is in state q_0 (see skeleton DFA)

Step 5: Construct the DFA. The DFA to accept strings of a's and b's ending with abb is given by:

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

where

- $Q = \{q_0, q_1, q_2, q_3\}$
- $\Sigma = \{a, b\}$
- q_0 = start state
- $F = \{q_3\}$
- δ is shown using the transition diagram/table as shown below:



Transition diagram

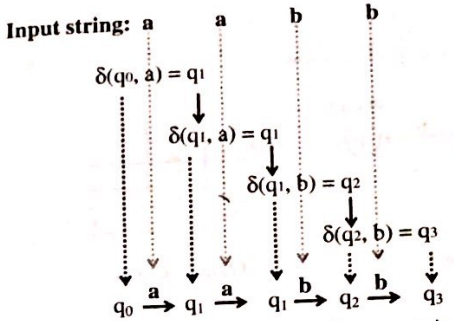
δ	a	b
→ q_0	q_1	q_0
q_1	q_1	q_2
q_2	q_1	q_3
*q_3	q_1	q_0

Transition table

Note: The language accepted by above DFA can be formally defined as:

$$L = \{ (a+b)^* abb : n \geq 0 \}$$

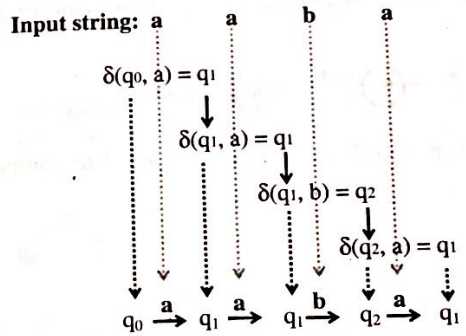
Now, let us "Show the sequence of moves made by the DFA for the string aabb". The moves made by the DFA for the string **aabb** is shown below:



Note: At the end of the string aabb, the DFA will be in state q_3 which is the final state. So, the string aabb is accepted by the machine.

(States a DFA is in during the processing of aabb)

Now, let us "Show the sequence of moves made by the DFA for the string aaba". The moves made by the DFA for the string **aaba** is shown below:



Note: At the end of the string aaba, the DFA will be in state q_1 which is not the final state. So, the string aaba is rejected by the machine.

(States a DFA is in during the processing of aaba)

Example 6: Now, let us "Draw a DFA to accept string of a's and b's which do not end with the string abb".

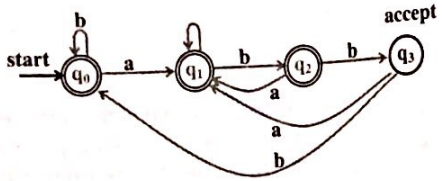
Solution: The design is exactly same as the previous problem. But, make final states as non final states and non final states as final states in the previous DFA. The resulting DFA is given by:

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

where

- $Q = \{q_0, q_1, q_2, q_3\}$
- $\Sigma = \{a, b\}$

- q_0 = start state
- $F = \{q_0, q_1, q_2\}$
- δ is shown using the transition diagram/table as shown below:



Transition diagram

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

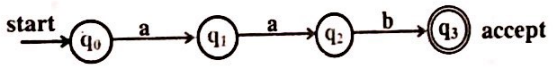
Transition table

Example 7: Now, let us "Draw a DFA to accept string of a's and b's having a substring aa

Step 1: Identify the minimum string: In this case aab.

Step 2: Identify the alphabets. In this case $\Sigma = \{a, b\}$.

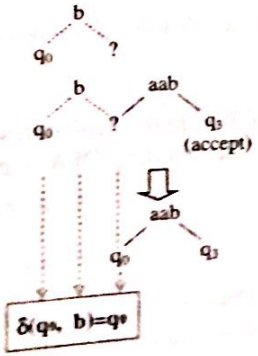
Step 3: Construct a skeleton DFA: The DFA for the string *aab* identified in step 1 can constru as shown below:



Step 4: Identify other transitions not defined in step 3. In this case, we need to compute following:

$\delta(q_0, b) = ?$ See step (i)
$\delta(q_1, b) = ?$ See step (ii)
$\delta(q_2, a) = ?$ See step (iii)
$\delta(q_3, a) = ?$ See step (iv)
$\delta(q_3, b) = ?$ See step (v)

Step (i) : $\delta(q_0, b)$=?

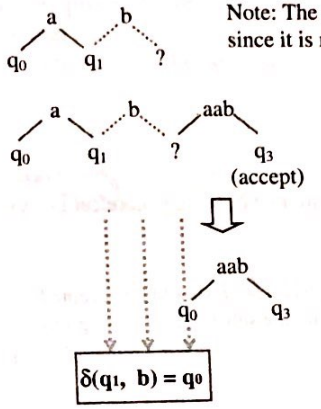


Note: The sequence *b* should not be accepted since it is not having the substring *aab*.

Note: But, *b* followed by *aab* resulting in string *baab* has to be accepted since it has the substring *aab*. So, go to final state q_3

But, before the suffix *aab*, the machine is in state q_0 (see skeleton DFA)

Step (ii): $\delta(q_1, b)$ = ? Move from q_0 to q_1 after reading symbol *a* (see skeleton DFA) and then think of transition from q_1 after reading symbol *b*.

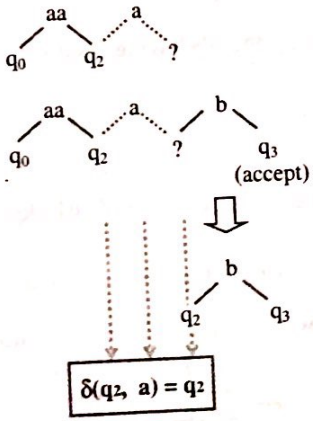


Note: The sequence *ab* should not be accepted since it is not having the substring *aab*.

Note: But, *ab* followed by *aab* resulting in string *abaab* has to be accepted since it has the substring *aab*. So, go to final state q_3

But, before the suffix *aab*, the machine is in state q_0 (see skeleton DFA)

Step (iii): $\delta(q_2, a)$ = ? Move from q_0 to q_2 after reading sequence *aa* (see skeleton DFA) and then think of transition from q_2 after reading symbol *a*.

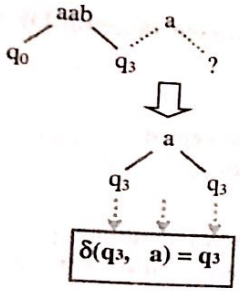


Note: The sequence *aaa* should not be accepted since it is not having the substring *aab*.

Note: But, *aaa* followed by *b* resulting in string *aaab* has to be accepted since it has the substring *aab*. So, go to final state q_3

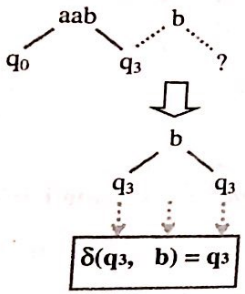
But, before the suffix *b*, the machine is in state q_2 (see skeleton DFA)

Step (iv): $\delta(q_3, a)$ = ? Move from q_0 to q_3 after reading sequence *aab* (see skeleton DFA) and then think of transition from q_3 after reading symbol *a*.



Note: The sequence *aaba* should be accepted since it is having the substring *aab*. So, go to final state q_3

Step (v): $\delta(q_3, b) = ?$ Move from q_0 to q_3 after reading sequence *aab* (see skeleton DFA) and think of transition from q_3 after reading symbol *b*.



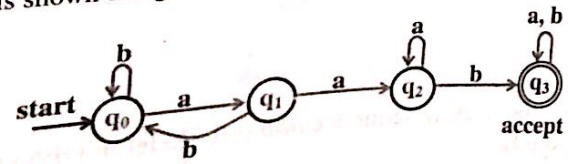
Note: The sequence *aabb* should be accepted since it is having the substring *aab*. So, go to final state q_3

Step 5: Construct the DFA. The DFA to accept strings of a's and b's having a substring *aa* given by:

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

where

- $Q = \{q_0, q_1, q_2, q_3\}$
- $\Sigma = \{a, b\}$
- $q0$ = start state
- $F = \{q_3\}$
- δ is shown using the transition diagram/table as shown below:



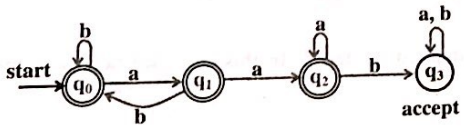
Transition diagram

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

Transition table

Example 8: Now, let us "Draw a DFA to accept string of a's and b's except those having the substring aab".

Solution: The procedure remains same as the previous problem. But, the DFA can be obtained by making non final states as final states and final state as non final state in the previous problem. The DFA obtained after the necessary changes is shown below:



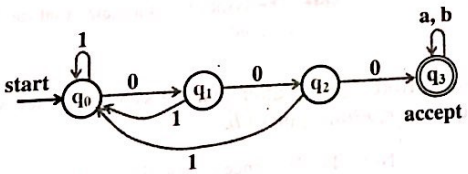
Transition diagram

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

Transition table

Example 9: Now, let us "Draw a DFA to accept string of 0's and 1's having three consecutive 0's".

Solution: The language can also be interpreted as strings of 0's and 1's having a substring 000. The procedure remains same as Example 7. But, start with the minimum string 000. The complete DFA is shown below:



Example 10: Now, let us "Draw a DFA to accept string of a's and b's such that

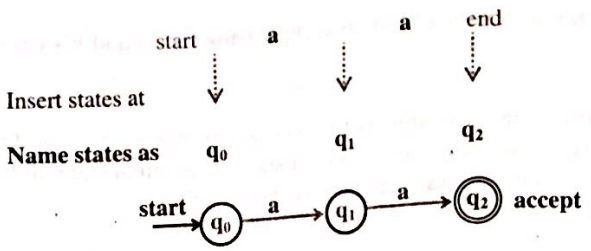
$$L = \{awa \mid w \in (a+b)^n \text{ where } n \geq 0\}$$"

Solution: The language can also be interpreted as "Strings of a's and b's starting with one *a* and ending with one *a* with minimum string *aa*".

Step 1: Identify the minimum string: In this case aa.

Step 2: Identify the alphabets. In this case $\Sigma = \{a, b\}$.

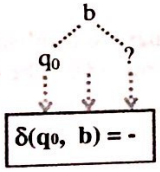
Step 3: Construct a skeleton DFA: The DFA for the string *aa* identified in step 1 can constructed as shown below:



Step 4: Identify other transitions not defined in step 3. In this case, we need to compute following:

$\delta(q_0, b) = ?$ See step (i)
$\delta(q_1, b) = ?$ See step (ii)
$\delta(q_2, a) = ?$ See step (iii)
$\delta(q_2, b) = ?$ See step (iv)

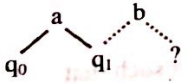
Step (i) : $\delta(q_0, b)$=?



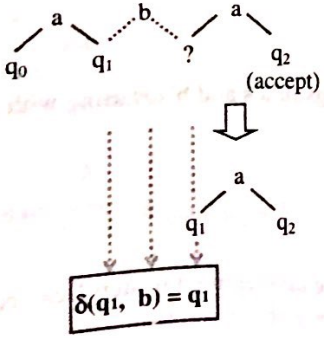
Note: The sequence *b* should be rejected since it is not starting with *a* and not ending with *a*. For the reject state, the transition should not be defined

Note: The symbol '-' indicates that the transition is not defined.

Step (ii): $\delta(q_1, b) = ?$ Move from q_0 to q_1 after reading symbol *a* (see skeleton DFA) and think of transition from q_1 after reading symbol *b*.



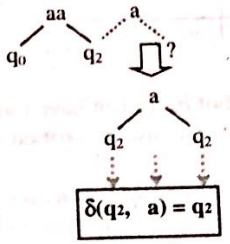
Note: The sequence *ab* should not be accepted. Because, even though it starts with *a*, it is not ending with *a*.



Note: But, *ab* followed by *a* resulting in string *aba* has to be accepted since it starts and ends with *a*. So, go to final state q_2

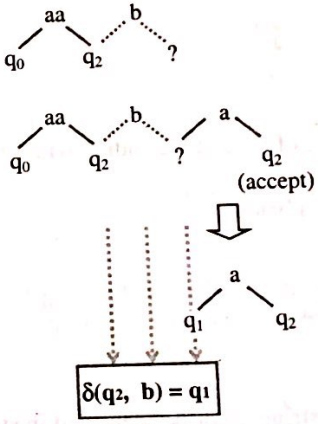
But, before the suffix *a*, the machine is in state q_1 (see skeleton DFA)

Step (iii): $\delta(q_2, a) = ?$ Move from q_0 to q_2 after reading sequence *aa* (see skeleton DFA) and then think of transition from q_2 after reading symbol *a*.



Note: The sequence *aaa* should be accepted since it starts and ends with *a*. So, go to final state q_1.

Step (iv): $\delta(q_2, b) = ?$ Move from q_0 to q_2 after reading sequence *aa* (see skeleton DFA) and then think of transition from q_2 after reading symbol *b*.



Note: The sequence *aab* should not be accepted since it is not ending with *a* (even though it starts with *a*)

Note: But, *aab* followed by *a* resulting in string *aaba* has to be accepted since it starts and ends with *a*. So, go to final state q_2

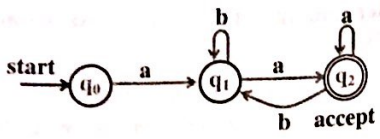
But, before the suffix *a*, the machine is in state q_1 (see skeleton DFA)

Step 5: Construct the DFA. The DFA to accept strings of a's and b's ending with abb is given by:

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

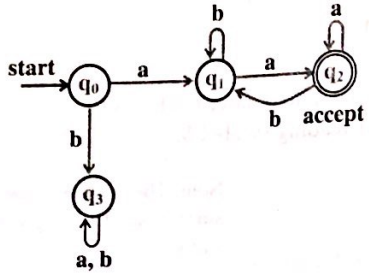
where

- $Q = \{q_0, q_1, q_2, q_3\}$
- $\Sigma = \{a, b\}$
- q_0 = start state
- $F = \{q_3\}$
- δ is shown using the transition diagram/table as shown below:



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

Note: Since the transition is not defined on q_0 for the input symbol *b*, we can have transition the trap state thus rejecting the string totally. So, the above DFA can also be written as sh below.



Example 11: Now, let us "Draw a DFA to accept string of a's and b's ending with *ab* or *ba*

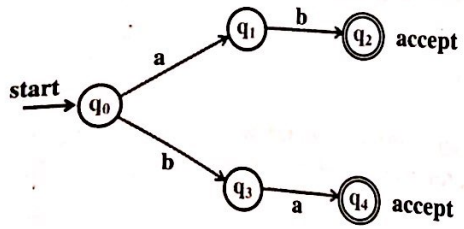
Solution: The given language can also be written as shown below:

$$L = \{w(ab + ba) \mid w \in \{a, b\}^*\}$$

Step 1: Identify the minimum string: In this case ab or ba.

Step 2: Identify the alphabets. In this case $\Sigma = \{a, b\}$.

Step 3: Construct a skeleton DFA: The DFA to accept the string *ab* or *ba* identified in step 1 constructed as shown below:

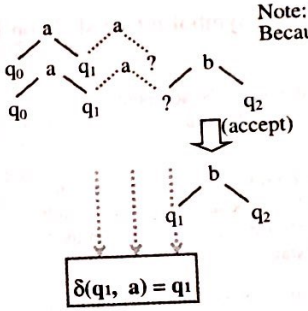


Step 4: Identify other transitions not defined in step 3: In this case, we need to compute the following:

$\delta(q_1, a) = ?$	See step (i)	$\delta(q_3, b) = ?$	See step (iv)
$\delta(q_2, a) = ?$	See step (ii)	$\delta(q_4, a) = ?$	See step (v)
$\delta(q_2, b) = ?$	See step (iii)	$\delta(q_4, b) = ?$	See step (vi)

Note: Whenever the *string ends with ab let us go to state* q_3 and whenever the string ends with *ba* let us go to state q_4.

Step (i): $\delta(q_1, a) = ?$ Move from q_0 to q_1 after reading symbol *a* (see skeleton DFA) and then think of transition from q_1 after reading symbol *a*.

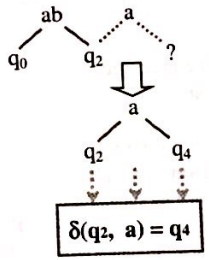


Note: The sequence *aa* should not be accepted. Because, it is not ending with *ab* or *ba*.

Note: But, *aa* followed by *b* resulting in string *aab* has to be accepted since it ends with *ab*. Because, it ends with *ab* let us go to final state q_2

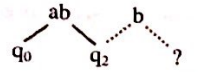
But, before the suffix *b*, the machine is in state q_1 (see skeleton DFA)

Step (ii): $\delta(q_2, a) = ?$ Move from q_0 to q_2 after reading string *ab* (see skeleton DFA) and then think of transition from q_2 after reading symbol *a*.

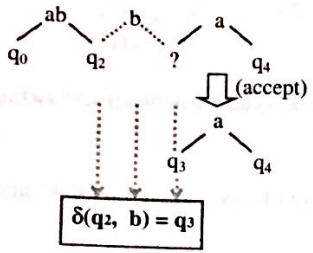


Note: The sequence *aba* should be accepted. Because, it is ending with *ba*. Since, the string ends with *ba*, the machine should go to q_4.

Step (iii): $\delta(q_2, b) = ?$ Move from q_0 to q_2 after reading string *ab* (see skeleton DFA) and then think of transition from q_2 after reading symbol *b*.



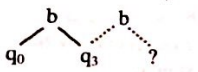
Note: The sequence *abb* should not be accepted. Because, it is not ending with *ab* or *ba*.



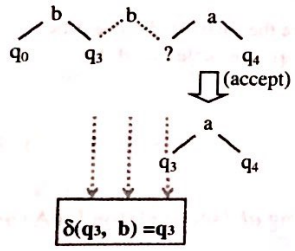
Note: But, *abb* followed by *a* resulting in string *abba* has to be accepted since it ends with *ba*. Because, it ends with *ba* let us go to final state q_4

But, before the suffix *a*, the machine is in state q_3 (see skeleton DFA)

Step (iv): $\delta(q_3, b) = ?$ Move from q_0 to q_3 after reading symbol *b* (see skeleton DFA) and think of transition from q_3 after reading symbol *b*.



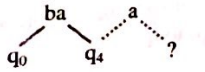
Note: The sequence *bb* should not be accepted. Because, it is not ending with *ab* or *ba*.



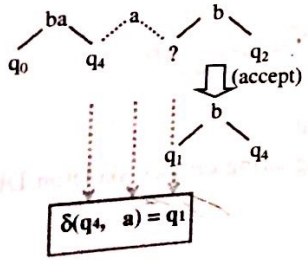
Note: But, *bb* followed by *a* resulting in string *bba* has to be accepted since it ends with *ba*. Because, it ends with *ba* let us go to final state q_4

But, before the suffix *a*, the machine is in state q_3 (see skeleton DFA)

Step (v): $\delta(q_4, a) = ?$ Move from q_0 to q_4 after reading string *ba* (see skeleton DFA) and then th of transition from q_4 after reading symbol *a*.



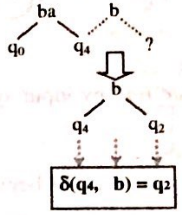
Note: The sequence *baa* should not be accepted. Because, it is not ending with *ab* or *ba*.



Note: But, *baa* followed by *b* resulting in string *baab* has to be accepted since it ends with *ab*. Because, it ends with *ab* let us go to final state q_2

But, before the suffix *b*, the machine is in state q_1 (see skeleton DFA)

Step (vi): $\delta(q_4, b) = ?$ Move from q_0 to q_4 after reading string *ba* (see skeleton DFA) and then think of transition from q_4 after reading symbol *b*.



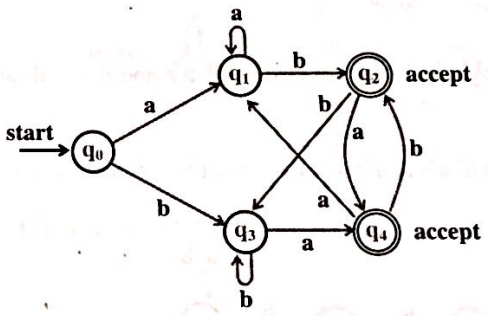
Note: The sequence *bab* should be accepted. Because, it is ending with *ab*. Since, the string ends with *ab*, the machine should go to q_2.

Step 5: Construct the DFA. The DFA to accept strings of a's and b's ending with *ab* or *ba* is given by:

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

where

- $Q = \{q_0, q_1, q_2, q_3, q_4\}$
- $\Sigma = \{a, b\}$
- q_0 = start state
- $F = \{q_2, q_4\}$
- δ is shown using the transition diagram/table as shown below:



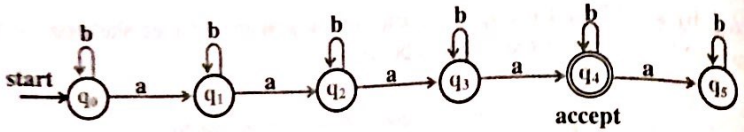
Transition diagram

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

Transition table

Example 12: Now, let us "Obtain a DFA to accept strings of a's and b's having four a's where $\Sigma = \{a, b\}$.

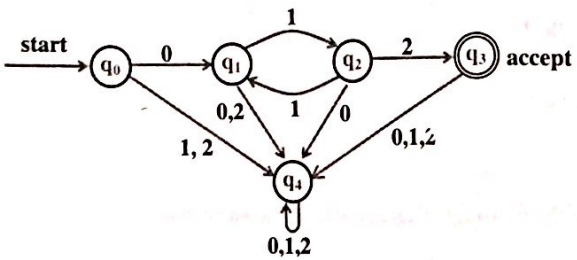
Solution: The DFA can be obtained using similar methods as explained earlier. The final DFA is shown below:



Note: A dead state is state where the FA remains in the same state for any input symbol. q_5 is a dead state.

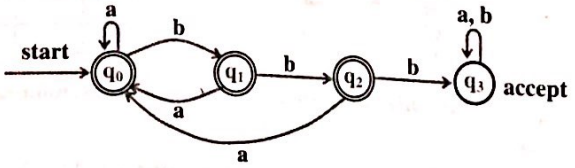
Example 13: Now, let us "Obtain a DFA to accept strings of 0's, 1's and 2's beginning with '0' followed by odd number of 1's and ending with a '2'".

Solution: The machine to accept the corresponding string is shown below:



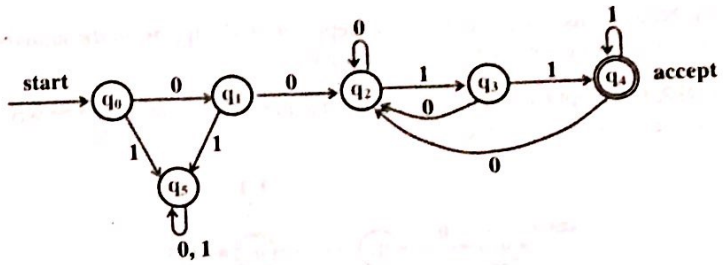
Example 14: Now, let us "Obtain a DFA to accept strings of a's and b's with at most two consecutive b's".

Solution: The machine to accept strings of a's and b's with at most two consecutive b's is shown below:



Example 15: Now, let us "Obtain a DFA to accept strings of 0's and 1's starting with at least two 0's and ending with at least two 1's".

Solution: The machine to accept the corresponding string is shown below:



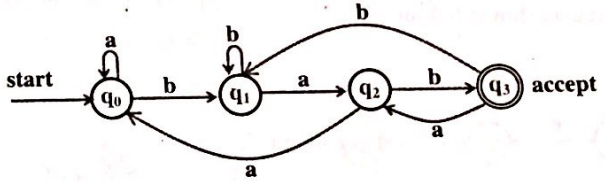
Note: In set notation, the language accepted DFA can be represented as

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

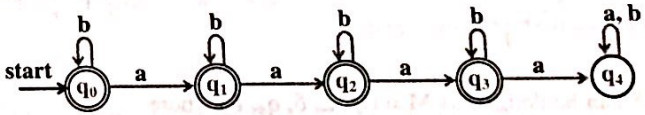
which is the language formed by words that begin with at least two 0's and ending with at least two 1's.

Example 16: Now, let us "Obtain a DFA to accept the language $L = \{wbab \mid w \in \{a, b\}^*\}$".

Solution: The DFA to accept the language $L = \{wbab \mid w \in \{a, b\}^*\}$ is shown below:



Example 17: Now, let us "Draw a DFA to accept string of a's and b's having not more than three a's".

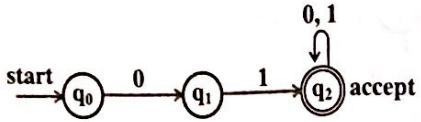


The language can also be represented as:

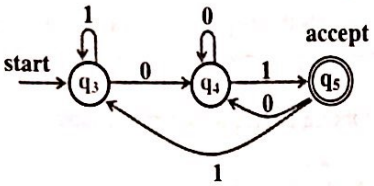
$$L = \{w : n_a(w) \leq 3, w \in \{a, b\}^*\}$$

Example 18: Now, let us "Draw a DFA to accept set of all strings on the alphabet Σ = {0, 1} that either begins or ends or both with the substring 01".

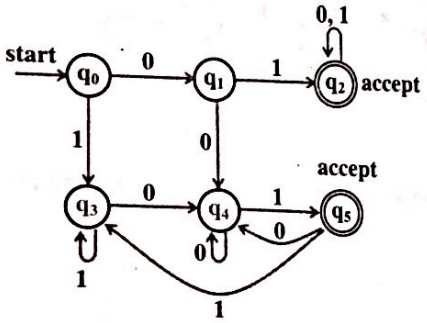
Solution: The DFA to accept strings of 0's and 1's starting with string 01 can be written as shown below (procedure remains same as Example 4):



The DFA to accept strings of 0's and 1's ending with string 01 can be written as shown below (procedure remains same as Example 5):



The two DFA can be joined to accept strings of 0's and 1's beginning with 01 and ending with 01 or both can be written as shown below:



δ	0	1
→ q_0	q_1	q_3
q_1	q_4	q_2
*q_2	q_2	q_2
q_3	q_4	q_3
q_4	q_4	q_5
*q_5	q_4	q_3

So, formally a DFA can be defined as **M = (Q, Σ, δ, q_0, F)** where

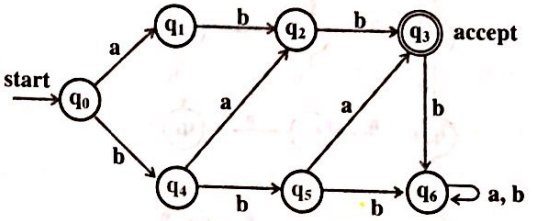
- Q = {q_0, q_1, q_2, q_3, q_4, q_5}
- Σ = {0, 1}
- δ is shown above using the transition diagram and transition table
- q_0 is the start state
- F = {q_2, q_5}

Example 19: Now, let us "Draw a DFA to accept the language L = {w: $n_a(w) \geq 1$, $n_b(w) = 2$}.

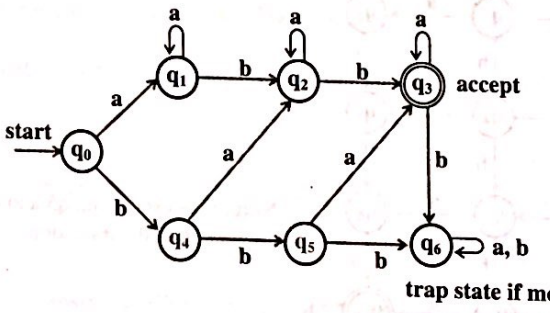
Solution: Note that the string should have exactly two *b*'s and at least one *a*. So, the minimum string that can be accepted by the DFA may be:

- abb
- bab
- bba

The skeleton DFA is shown below:



But, one or more a's can be present in the string. So, the above DFA can be written (using the same steps of designing technique) as shown below:

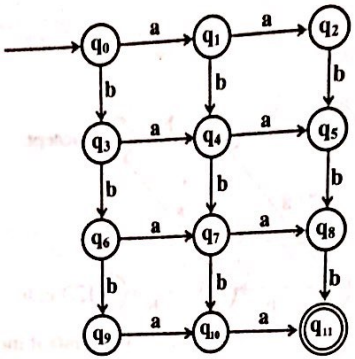


Example 20: Now, let us "Draw a DFA to accept the language L = {w: $n_a(w) = 2$, $n_b(w) >= 3$}.

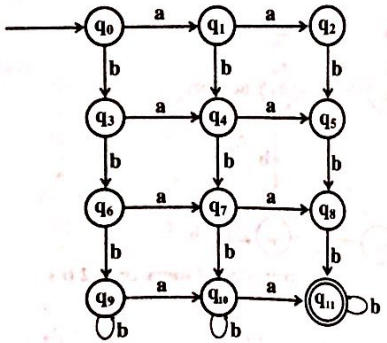
Solution: It is observed from the given language that the string that is accepted by DFA should have exactly two a's and at least three b's. So, the minimum strings that can be accepted are:

- aabbb
- ababb
- abbab
- abbba
- baabb
- babab
- babba
- bbaab
- bbaba
- bbbaa

The skeleton DFA that accepts all the above strings is shown below:



Since, minimum three b's should be there, after three b's we can consume any number of So, the above DFA can be written as shown below:



Note: From states q2, q5 and q8 do not define the transitions

1.6.2. Divisible by k Problems

In this section, let us discuss a method of constructing DFA that divide a number by *k*. For th types of problems, the transitions can be obtained using the following relation:

$\delta(q_i, a) = q_j$ where $j = (r*i + d) \bmod k$

- *r* is the radix of input. For binary *r* = 2
- *i* is the remainder obtained after dividing by k
- *d* represent digits. For binary *d* = { 0, 1 }
- *k* is divisor

he steps to be followed to find FA using these types of problems is shown below:

tep 1: Identify the radix, input alphabets and the divider *k*.

tep 2: Compute the possible remainders: These remainders represent the states of DFA.

tep 3: Find the transitions using $\delta(q_i, a) = q_j$ where $j = (r * i + d) \bmod k$.

tep 4: Construct the DFA using the transitions obtained in step 3.

Example 1: Now, let us "Construct a DFA which accepts strings of 0's and 1's where the value f each string is represented as a binary number. Only the strings representing zero modulo five hould be accepted. For example, 0000, 0101, 1010, 1111, etc. should be accepted".

olution: The DFA can be obtained as shown below:

tep 1: Identify the radix, input alphabets and the divisor k: In this case, r = 2:

$d = \{0, 1\}, k = 5$

tep 2: Compute the possible remainders: After dividing by k, the possible remainders are:

$i = 0, 1, 2, 3, 4$

tep 3: Compute transitions: The transitions can be computed using the following relation:

$$\delta(q_i, a) = q_j \text{ where } j = (r * i + d) \bmod k$$

with $r = 2$ and $k = 5$

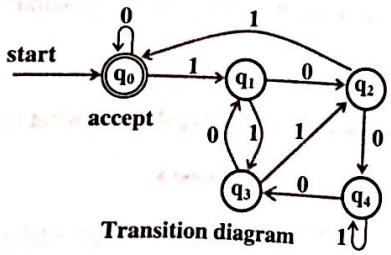
So, $j = (2i + d) \bmod 5$

remainder	d	(2* i +d) mod 5 = j	$\delta(q_i,d) = q_j$
i = 0	0	(2*0 + 0) mod 5 = 0	$\delta(q_0,0) = q_0$
	1	(2*0 + 1) mod 5 = 1	$\delta(q_0,1) = q_1$
i = 1	0	(2*1 + 0) mod 5 = 2	$\delta(q_1,0) = q_2$
	1	(2*1 + 1) mod 5 = 3	$\delta(q_1,1) = q_3$
i = 2	0	(2*2 + 0) mod 5 = 4	$\delta(q_2,0) = q_4$
	1	(2*2 + 1) mod 5 = 0	$\delta(q_2,1) = q_0$
i = 3	0	(2*3 + 0) mod 5 = 1	$\delta(q_3,0) = q_1$
	1	(2*3 + 1) mod 5 = 2	$\delta(q_3,1) = q_2$
i = 4	0	(2*4 + 0) mod 5 = 3	$\delta(q_4,0) = q_3$
	1	(2*4 + 1) mod 5 = 4	$\delta(q_4,1) = q_4$

Transitions of resulting DFA

Step 4: The DFA can be defined as $\mathbf{M = (Q, \Sigma, \delta, q_0, F)}$ where

- $\mathbf{Q} = \{q_0, q_1, q_2, q_3, q_4\}$
- $\Sigma = \{0, 1\}$
- $\mathbf{q_0}$ is the start state
- $\mathbf{F} = \{q_0\}$
- δ is shown below using the transition diagram and transition table as shown below:



δ	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

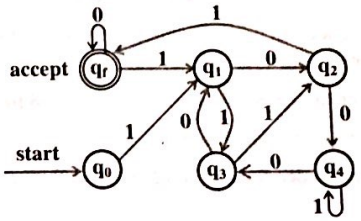
Transition Table

Note: Observe that for modulo 5, number of states of DFA will be 5.

In general, for modulo k, number of states of DFA will be k.

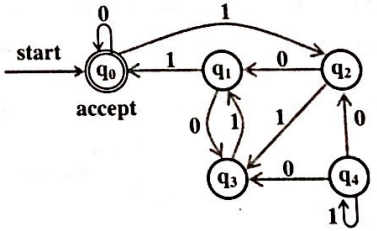
Example 2: Now, let us "obtain a DFA which accepts the set of all strings beginning with a 1 hat when interpreted as a binary integer, is a multiple of 5. For example, 101, 1010, 1111 etc are ıultiples of 5. Note that 0101 is not beginning with 1 and it should not be accepted".

olution: The solution to this problem is same as above problem. But, the number always should tart with a 1. If a binary number starts with a 0, the number should never be accepted. So, let us ename the final state as q_f and have the new start state q_0 and from this symbol on 1, enter into tate q_1. The resulting DFA is shown using the transition diagram as shown below:



Example 3: Now, let us "obtain a DFA that accepts set of all strings that, when interpreted n reverse as a binary integer, is divible by 5. Examples of strings in the language are 0, 10011, 001100 and 0101".

olution: The solution remains same as Example 1. But, reverse the direction of all arrow marks except the arrow labeled with start). The resulting DFA is shown below:



Example 4: Now, let us "Draw a DFA to accept decimal strings divisible by 3".

Solution: The DFA can be obtained as shown below:

Step 1: Identify the radix, input alphabets and the divisor k: In this case, r = 10:

$d = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, k = 3

Step 2: Compute the possible remainders: After dividing any decimal number by 3, re... following remainders:

i = 0, 1, 2 which implies q_0, q_1 and q_2 are the states of DFA

Step 3: Compute transitions: The transitions can be computed using the following r... $\delta(q_i, a) = q_j$ where $j = (r * i + d) \bmod k$

with r = 10 and k = 3

So, $j = (2i + d) \bmod 3$

Note: For the sake of convenience, let us group the digits from 0 to 9 based on the remaind... get after dividing by 3 as shown below:

- {0, 3, 6, 9} with 0 as the remainder. So, δ from {0, 3, 6, 9} ⇒ δ from {0}
- {1, 4, 7} with 1 as the remainder. So, δ from {1, 4, 7} ⇒ δ from {1}
- {2, 5, 8} with 2 as the remainder. So, δ from {2, 5, 8} ⇒ δ from {2}

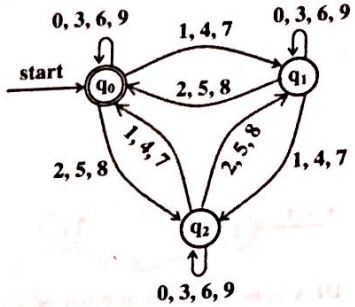
remainder	d	$(2*i+d) \bmod 3 = j$	$\delta(q_i,d) = q_j$		
i = 0	0	$(10*0 + 0) \bmod 3 = 0$	$\delta(q_0,0) = q_0$	⇒	$\delta(q_0,\{0,3,6,9\}) = q_0$
	1	$(10*0 + 1) \bmod 3 = 1$	$\delta(q_0,1) = q_1$	⇒	$\delta(q_0,\{1,4,7\}) = q_1$
	2	$(10*0 + 2) \bmod 3 = 2$	$\delta(q_0,2) = q_2$	⇒	$\delta(q_0,\{2,5,8\}) = q_2$
i = 1	0	$(10*1 + 0) \bmod 3 = 1$	$\delta(q_1,0) = q_1$	⇒	$\delta(q_1,\{0,3,6,9\}) = q_1$
	1	$(10*1 + 1) \bmod 3 = 2$	$\delta(q_1,1) = q_2$	⇒	$\delta(q_1,\{1,4,7\}) = q_2$
	2	$(10*1 + 2) \bmod 3 = 0$	$\delta(q_1,2) = q_0$	⇒	$\delta(q_1,\{2,5,8\}) = q_0$
i = 2	0	$(10*2 + 0) \bmod 3 = 2$	$\delta(q_2,0) = q_2$	⇒	$\delta(q_2,\{0,3,6,9\}) = q_2$
	1	$(10*2 + 1) \bmod 3 = 0$	$\delta(q_2,1) = q_0$	⇒	$\delta(q_2,\{1,4,7\}) = q_0$
	2	$(10*2 + 2) \bmod 3 = 1$	$\delta(q_2,2) = q_1$	⇒	$\delta(q_2,\{2,5,8\}) = q_1$

Transitions of DFA

Step 4: The DFA can be defined as $M = (Q, \Sigma, \delta, q_0, F)$ where

- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- q_0 is the start state

- $F = \{q_0\}$
- δ is shown below using the transition diagram and transition table as shown below:



1.6.3. Modulo k Counter Problems

Example 1: Let us "Obtain a DFA to accept strings of even number of a's". The DFA can be constructed by following the steps one by one as shown below:

Solution: It is given that L = {w : w has even number of a's}.

Identify the number of states: The string w may have even number of a's or odd number of a's and results in following two cases (two states):

- **Case 0:** Strings that accepts Even number of a's is denoted by: (E)
- **Case 1:** Strings that accepts Odd number of a's is denoted by: (O)

Identify the start state and final state: Before reading any of the input symbols, number of a's will be zero which represent Even a's. So, the state E with even number of a's is the start state.

Since, it is required to accept Even number of a's the state E which accepts even number of a's is the final state.

Design: Once the start state and final states are identified the transitions can be easily obtained as shown below:

- From a state E, on reading *a* results in odd number of a's and hence change to odd state O. The transition is:

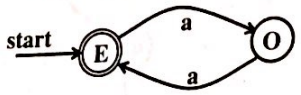
  $\delta(E, a) = O$

- From a state O, on reading *a* results in even number of a's and hence change to even. The transition is:

  $\delta(O, a) = E$

So, the DFA $M = (Q, \Sigma, \delta, q_0, F)$ is defined as:

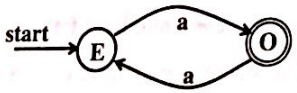
- $Q = \{E, O\}$
- $\Sigma = \{a\}$
- δ = shown in transition diagram



DFA to accept even number of a's

- $q_0 = E$ is the start state
- $F = \{E\}$

Note: The DFA to obtain odd number of a's can be obtained by making O state (odd state) as final state and E state as the start state as shown below:



DFA to accept odd number of a's

Example 2: Let us "Obtain a DFA to accept the language $L = \{ w : |w| \bmod 3 = 0\}$ where $\Sigma = \{a\}$".

Solution: It is given that $L = \{w : |w| \bmod 3 = 0\}$ where $\Sigma = \{a\}$ } which indicates that language consists of strings of multiples of 3 a's.

Identify the number of states: The language can be interpreted as strings of a's such that number of a's in string is divisible by 3. Note that $|w| \bmod 3$ results in three cases:

- **Case 0:** Results in remainder 0: The state is identified as q_0. 
- **Case 1:** Results in remainder 1. The state is identified as q_1. 
- **Case 2:** Results in remainder 2. The state is identified as q_2. 

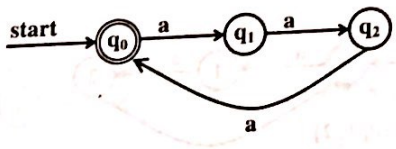
Identify the start state and final state: Before reading any inputs, number of a's will be zero. So, 0 mod 3 = 0 which results in case 0. Hence, state q0 is start state. Since, it is required to accept string of a's such that $|w| \bmod 3 = 0$, which results in case 0, q_0 is considered as final state.

Design: Once the start state and final states are identified the transitions can be easily obtained as shown below:

- From a state with remainder 0 (case 0) denoted by q_0, on reading *a* results in remainder 1 (case 1) denoted by q_1. So, $\delta(q_0, a) = q_1$
- From a state with remainder 1 (case 1) denoted by q_1, on reading *a* results in remainder 2 (case 2) denoted by q_2. So, $\delta(q_1, a) = q_2$
- From a state with remainder 2 (case 2) denoted by q_1, on reading *a* results in remainder 0 (case 0) denoted by q_0. So, $\delta(q_2, a) = q_0$

So, the DFA $M = (Q, \Sigma, \delta, q0, F)$ is defined as:

- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{a\}$
- δ : the transitions are shown using transition diagram as shown below:



- q_0 = start state
- $F = \{q_0\}$

The given language accepted by above DFA can also be represented as:

- $L = \{w : |w| \bmod 3 = 0\}$ where $\Sigma = \{a\}$ }

  or

- $L = \{w: n_a(w)$ are divisible by 3 where $\Sigma=\{a\}\}$

  or

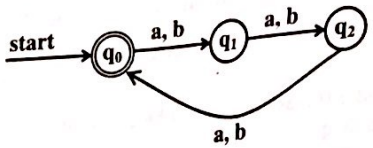
- $L = \{a^{3n} : n \geq 0\}$

Example 3: Let us "Obtain a DFA to accept the language $L = \{ w : |w| \bmod 3 = 0\}$ on $\Sigma = \{a, b\}$".

Solution: The language consists of strings of a's and b's whose length is a multiple of 3 and can be represented as:

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

The design is similar to that of the previous problem. But, add one extra label *b* for each as shown below:

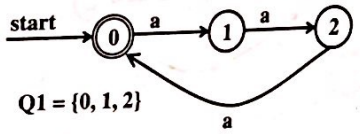


Example 4: Now, let us "Obtain a DFA to accept the following language L = {w such tha

a) |w| mod 3 ≥ |w| mod 2 where w ∈ Σ* and Σ ={a}

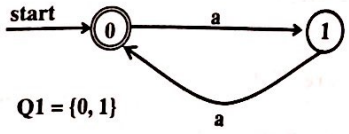
b) |w| mod 3 ≠ |w| mod 2 where w ∈ Σ* and Σ ={a}

Solution: The DFA to accept a string *w* such that |w| mod 3 = 0 can be written as shown b (see Example 2):



M1

On similar lines, the DFA to accept a string *w* such that |w| mod 2 = 0 can be written as sh below (see Example 1):

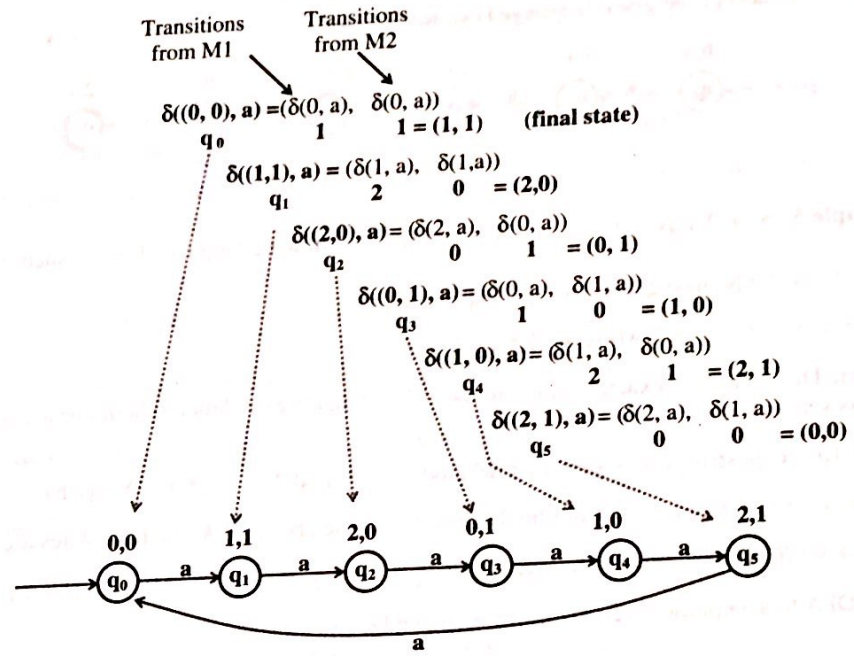


M2

Transitions: The transitions of DFA which has strings of *w* with |w| mod 3 and |w| mod 2 ca obtained by taking the cross product of Q1 and Q2 as shown below:

Q1 x Q2 = { (0,0), (0,1), (1,0), (1,1), (2,0), (2,1) }

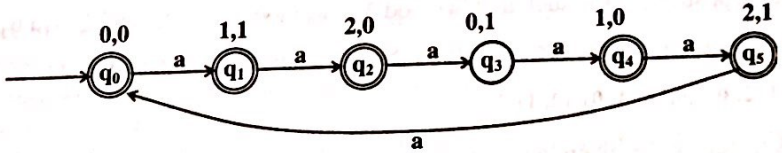
The transitions on each of the pair (x, y) can be obtained as shown below:



Case 1: To accept strings of *w* such that |w| mod 3 ≥ |w| mod 2, the pairs (x, y) such that x ≥ y are final states. So, in the above DFA, the final states are:

F = { (0, 0), (1, 1), (2, 0), (1, 0), (2, 1) }

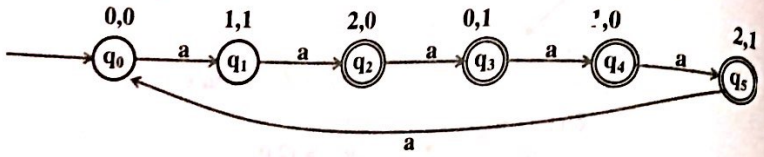
So, the DFA to accept the given language is shown below:



Case 2: To accept strings of *w* such that |w| mod 3 ≠ |w| mod 2, w ∈ Σ* and Σ ={a} the pairs (x, y) such that x ≠ y are final states. So, the final states in the DFA are:

F = { (2, 0), (0,1), (1, 0), (2, 1) }

So, the DFA to accept the given language is shown below:



Example 5: Now, let us "Obtain a DFA to accept the following language L = {w such th

a) $|w| \bmod 3 \geq |w| \bmod 2$ where $w \in \Sigma^*$ and $\Sigma = \{a, b\}$

b) $|w| \bmod 3 \neq |w| \bmod 2$ where $w \in \Sigma^*$ and $\Sigma = \{a,b\}$

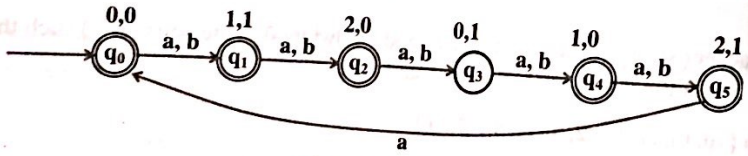
Solution: The solution is exactly similar to the above, but, the extra label *b* should be added with *a* as shown below:

Case 1: To accept strings of *w* such that $|w| \bmod 3 \geq |w| \bmod 2$, $w \in \Sigma^*$ and $\Sigma = \{a,b\}$.

The pairs (x, y) such that x >= y are final states. So, in the above DFA, the final states are:

F = { (0, 0), (1, 1), (2, 0), (1, 0), (2, 1) }

So, the DFA to accept the given language is shown below:

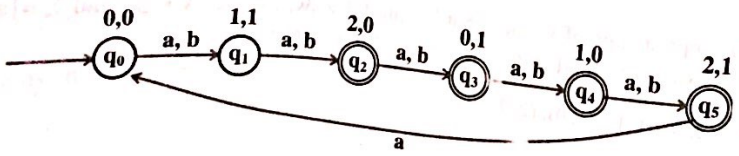


Case 2: To accept strings of *w* such that $|w| \bmod 3 \neq |w| \bmod 2$, $w \in \Sigma^*$ and $\Sigma = \{a,b\}$.

The pairs (x, y) such that x ≠ y are final states. So, the final states in the DFA are:

F = { (2, 0), (0,1), (1, 0), (2, 1) }

So, the DFA to accept the given language is shown below:



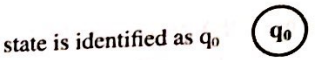
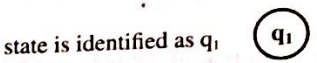
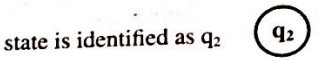
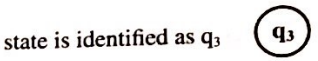
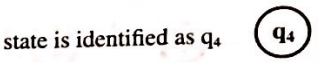
Example 6: Let us "Obtain a DFA to accept the language L = { w : $|w| \bmod 5 \neq 0$ } on $\Sigma = \{a\}$".

Solution: It is given that L = {w : $|w| \bmod 5 \neq 0$} where $\Sigma = \{a\}$ } which indicates that the language consists of strings of a's which are not multiples of 5 a's.

Identify the number of states: The given language can be interpreted as strings of only a's such that the number of a's in the string is not divisible by 5. Note that

$|w| \bmod 5$

results in remainder 0, 1, 2, 3 and 4 which results in five cases as shown below:

- **Case 0:** Results in remainder 0: The state is identified as q_0 
- **Case 1:** Results in remainder 1. The state is identified as q_1 
- **Case 2:** Results in remainder 2. The state is identified as q_2 
- **Case 3:** Results in remainder 3. The state is identified as q_3 
- **Case 4:** Results in remainder 4. The state is identified as q_4 

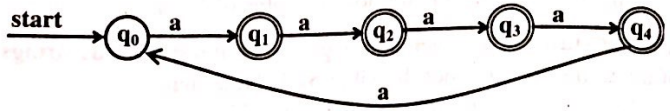
Identify the start state and final state: Before reading any of the input symbols, number of a's will be zero. So, 0 mod 5 = 0 which results in case 0 . Hence, state q0 is start state. Since, it is required to accept string of a's such that $|w| \bmod 5 \neq 0$ (i.e., remainder is not zero), other than q_0 all other states can be the final states. So, the states q1, q_2, q_3 and q_4 are final states.

Design: Once the start state and final states are identified, the transitions can be easily obtained as shown below:

- From a state with remainder 0 (case 0) denoted by q_0, on reading *a* results in remainder 1 (case 1) denoted by q_1. So, $\delta(q_0, a) = q_1$
- From a state with remainder 1 (case 1) denoted by q_1, on reading *a* results in remainder 2 (case 2) denoted by q_2. So, $\delta(q_1, a) = q_2$
- From a state with remainder 2 (case 2) denoted by q_1, on reading *a* results in remainder 3 (case 3) denoted by q_3. So, $\delta(q_2, a) = q_3$
- From a state with remainder 3 (case 3) denoted by q_3, on reading *a* results in remainder 4 (case 4) denoted by q_4. So, $\delta(q_3, a) = q_4$
- From a state with remainder 4 (case 4) denoted by q_4, on reading *a* results in remainder 0 (case 0) denoted by q_0. So, $\delta(q_2, a) = q_0$

So, the DFA $M = (Q, \Sigma, \delta, q_0, F)$ is defined as:

- $Q = \{q_0, q_1, q_2, q_3, q_4\}$
- $\Sigma = \{a\}$
- δ = shown in transition diagram as shown below:

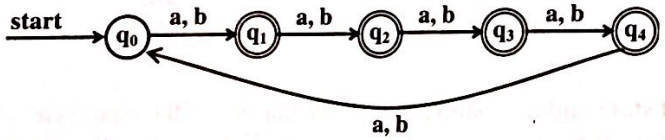


- q_0 = start state
- $F = \{q_1, q_2, q_3, q_4\}$

Example 7: Let us "Obtain a DFA to accept the language $L = \{ w : |w| \bmod 5 \neq 0 \}$ $\Sigma = \{a,b\}$".

Solution: The design is exactly similar to the previous problem but each arrow is labeled with *a, b*. So, the DFA $M = (Q, \Sigma, \delta, q_0, F)$ is defined as:

- $Q = \{q_0, q_1, q_2, q_3, q_4\}$
- $\Sigma = \{a, b\}$
- δ = shown in transition diagram as shown below:



- q_0 = start state
- $F = \{q_1, q_2, q_3, q_4\}$

Example 8: Let us "Obtain a DFA to accept strings of a's and b's having even number of a's and even number of b's".

Solution: It is given that L = {w : w has even number of a's and even number of b's.}

Identify the number of states: The string w made up of a's and b's may have:

- *Even number of a's* denoted by E_a
- *Even number of b's* denoted by E_b
- *Odd number of a's* denoted by O_a
- *Odd number of b's* denoted by O_b

So, even/odd a's along with even/odd b's results in following four cases (four states):

- **Case 0:** Strings having Even a's and Even b's is denoted by: q_0 — E_a, E_b (q_0)
- **Case 1:** Strings having Even a's, Odd b's is denoted by: q_1 — E_a, O_b (q_1)
- **Case 2:** Strings having Odd a's, Even b's is denoted by: q_2 — O_a, E_b (q_2)
- **Case 3:** Strings having Odd a's, Odd b's is denoted by: q_3 — O_a, E_b (q_3)

Identify the start state and final state: Before reading any of the input symbols, number of a's and number of b's will be zero which represent Even a's and Even b's (case 0). So, the state q_0 is the start state.

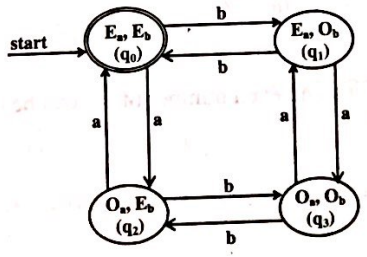
Since, it is required to accept Even a's and Even b's, the state q0 with even a's and even b's is the final state.

Design: Once the start state and final states are identified the transitions can be easily obtained as shown below:

- From a state E_a with even number of a's, on reading input symbol *a*, results in odd number of a's denoted by O_a. So, $\delta(E_a, a) = O_a$
- From a state O_a with odd number of a's, on reading input symbol *a*, results in even number of a's denoted by E_a. So, $\delta(O_a, a) = E_a$
- From a state E_b with even number of b's, on reading input symbol *b*, results in odd number of b's denoted by O_b. So, $\delta(E_b, b) = O_b$
- From a state O_b with odd number of b's, on reading input symbol *b*, results in even number of b's denoted by E_b. So, $\delta(O_b, b) = E_b$

So, the DFA $M = (Q, \Sigma, \delta, q0, F)$ is defined as:

- $Q = \{q_0, q_1, q_2, q_3\}$
- $\Sigma = \{a, b\}$
- δ = shown in transition diagram



DFA to accept even number of a's and even number of b's

- q_0 = start state
- $F = \{q_0\}$

Note: The language accepted by above DFA can be written as:

$$L = \{w : w \text{ has even number of a's and even number of b's.}\}$$

or

$$L = \{w : \text{Both } N_a(w) \text{ and } N_b(w) \text{ are divisible by } 2\}$$

or

$$L = \{w : \text{Both } N_a(w) \text{ and } N_b(w) \text{ are multiples of } 2\}$$

or

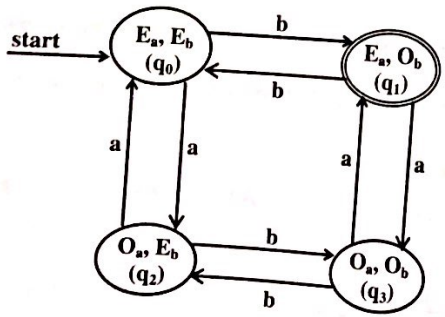
$$L = \{w \mid N_a(w) \bmod 2 = 0 \text{ and } N_b(w) \bmod 2 = 0\}$$

Note: $N_a(w)$ is the total number of a's in the string w and $N_b(w)$ is the total number of b's in string w.

Note: The DFA to accept even number of a's and odd number of b's can be obtained by making:

E_a, O_b
(q_1)

as the only final state as shown below:

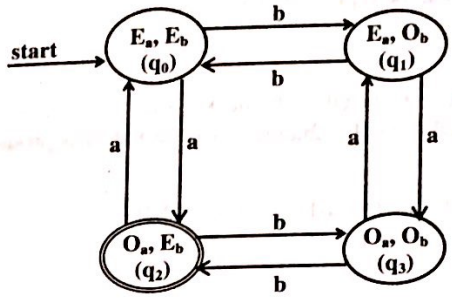


DFA to accept even number ofa's and odd number of b's

Note: The DFA to accept odd number of a's and even number of b's can be obtained by making:

O_a, E_b
(q_2)

as the only final state as shown below:

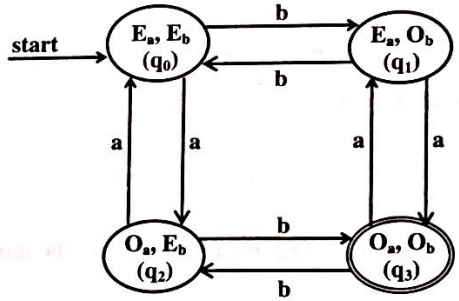


DFA to accept odd number of a's and even number of b's

Note: The DFA to accept odd number of a's and odd number of b's can be obtained by making:

O_a, O_b
(q_3)

as the only final state as shown below:



DFA to accept odd number of a's and odd number of b's

■ **Example 9:** Obtain a DFA to accept strings of a's and b's such that

$$L = \{w \mid w \in (a+b)^* \text{ such that } N_a(w) \bmod 3 = 0 \text{ and } N_b(w) \bmod 2 = 0\}\ \}$$

Solution: The $N_a(w) \bmod 3$ gives the remainder after dividing number of a's by 3.

The possible remainders are $\{0, 1, 2\}$ and can be represented as:

$$Q_1 = \{A_0, A_1, A_2\} \quad (1)$$

The $N_b(w) \bmod 2$ gives the remainder after dividing number of b's by 2.

The possible remainders are {0, 1} and can be represented as:

$$Q_2 = \{B_0, B_1\}$$

Identify the states of DFA: Since each state of DFA should keep track of Na(w) mod 3 and Nb(w) mod 2, the possible states of the DFA can be obtained by Q1xQ2 (cross product) and can be represented as shown below:

$$Q1 \times Q2 = \{(A0, B0), (A0, B1), (A1, B0), (A1, B1), (A2, B0), (A2, B1)\}$$

where

- A0 indicates that $N_a(w) \bmod 3 = 0$
- A1 indicates that $N_a(w) \bmod 3 = 1$
- A2 indicates that $N_a(w) \bmod 3 = 2$
- B0 indicates that $N_b(w) \bmod 2 = 0$
- B1 indicates that $N_b(w) \bmod 2 = 1$

Identify the start state: Before reading any of the input symbols, number of a's and number of b's will be zero. So, Na(w) mod 3 = 0 and Nb(w) mod 2 = 0 which can be denoted by the state (A0, B0). So, (A0, B0) is the start state.

Identify the final state: Since, it is required to accept the language

$$L = \{w : N_a(w) \bmod 3 = 0 \text{ and } N_b(w) \bmod 2 = 0\}$$

(A0 under the first condition, B0 under the second)

the state (A0, B0) is the final state.

Design: Once the start state and final states are identified, the transitions for the input symbol *a* can be obtained as shown below:

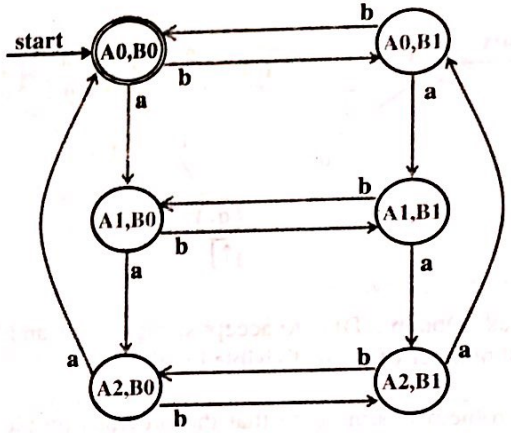
- From state A0, on reading input symbol *a*, the machine should change the state to A1 $\delta(A_0, a) = A_1$
- From state A1, on reading input symbol *a*, the machine should change the state to A2 $\delta(A_1, a) = A_2$
- From state A2, on reading input symbol *a*, the machine should change the state to A0 $\delta(A_2, a) = A_0$

Similarly, the transitions for the input symbol *b* can be obtained as shown below:

- From state B0, on reading input symbol *b*, the machine should change the state to B1 $\delta(B_0, b) = B_1$
- From state B1, on reading input symbol *b*, the machine should change the state to B0 $\delta(B_1, b) = B_0$

So, the DFA M = (Q, Σ, δ, q_0, F) is defined as:

- Q = {(A0,B0), (A0, B1), (A1, B0), (A1, B1), (A2,B0), (A2, B1)}
- Σ = {a, b}
- q_0 = {(A0, B0) is the start state
- F = {(A0, B0)}
- δ = shown in transition diagram



Note: By changing the final states of DFA various languages can be accepted by DFAas shown below:

- To accept the language $L = \{w \mid w \in (a+b)^* N_a(w) \bmod 3 = 1 \text{ and } N_b(w) \bmod 2 = 0\}$ (A1 under the first condition, B0 under the second)

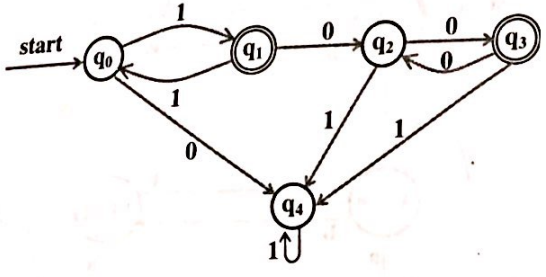
  make the state (A1, B0) as the final state.
- To accept the language $L = \{w \mid w \in (a+b)^* N_a(w) \bmod 3 = 2 \text{ and } N_b(w) \bmod 2 = 0\}$ (A2 under the first condition, B0 under the second)

  make the state (A2, B0) as the final state.
- To accept the language $L = \{w \mid w \in (a+b)^* N_a(w) \bmod 3 = 2 \text{ and } N_b(w) \bmod 2 = 1\}$ (A2 under the first condition, B1 under the second)

  make the state (A2, B1) as the final state and so on.

Example 10: Now, let us "Draw a DFA to accept the language: L = {w: w has odd number of 1's and followed by even number of 0's) Completely define DFA and transition function:

Solution: The DFA to accept strings of 0's and 1's such that the string has odd number of 1's and followed by even number of 0's is shown below:



Example 11: Now, let us "Obtain a DFA to accept strings of a's and b's such that the number of a's is divisible by 5 and number of b's is divisible by 3".

Note: Observe that the problem is similar to that the previous problem with more number of states.

Solution: The given language can be interpreted as shown below:

$$L = \{w \mid w \in (a+b)^* \; N_a(w) \bmod 5 = 0 \text{ and } N_b(w) \bmod 3 = 0\}$$

The $N_a(w)$ mod 5 gives the remainder after dividing number of a's by 5.

The possible remainders are { 0, 1, 2, 3, 4} and can be represented as:

$$\downarrow \; \downarrow \; \downarrow \; \downarrow \; \downarrow$$

$$Q_1 = \{A0, A1, A2, A3, A4\} \quad (1)$$

The $N_b(w)$ mod 3 gives the remainder after dividing number of b's by 3.

The possible remainders are {0, 1, 2} and can be represented as:

$$\downarrow \; \downarrow \; \downarrow$$

$$Q_2 = \{ B0, B1, B2 \} \quad (2)$$

Identify the states of DFA: Since each state of DFA should keep track of Na(w) mod 3 and Nb(w) mod 2, the possible states of the DFA can be obtained by Q1xQ2 (cross product) and can be represented as shown below:

$$Q1 \times Q2 = \{(A0, B0), (A0, B1), (A0, B2),$$
$$(A1, B0), (A1, B1), (A1, B2),$$
$$(A2, B0), (A2, B1), (A2, B2),$$
$$(A3, B0), (A3, B1), (A3, B2),$$
$$(A4, B0), (A4, B1), (A4, B2)$$
$$\}$$

where

- A0 indicates that $N_a(w) \bmod 5 = 0$
- A1 indicates that $N_a(w) \bmod 5 = 1$
- A2 indicates that $N_a(w) \bmod 5 = 2$
- A3 indicates that $N_a(w) \bmod 5 = 3$
- A4 indicates that $N_a(w) \bmod 5 = 4$
- B0 indicates that $N_b(w) \bmod 3 = 0$
- B1 indicates that $N_b(w) \bmod 3 = 1$
- B2 indicates that $N_b(w) \bmod 3 = 2$

Identify the start state: Before reading any of the input symbols, number of a's and number of b's will be zero. So, $N_a(w) \bmod 5 = 0$ and $N_b(w) \bmod 3 = 0$ which can be denoted by the state (A0, B0). So, (A0, B0) is the start state.

Identify the final state: Since, it is required to accept the language

$$L = \{w : N_a(w) \bmod 5 = 0 \text{ and } N_b(w) \bmod 3 = 0\}$$
$$\downarrow \qquad\qquad \downarrow$$
$$\mathbf{A0} \qquad\qquad \mathbf{B0}$$

the state (A0, B0) is the final state.

Design: Once the start state and final states are identified, the transitions for the input symbol *a* can be obtained as shown below:

- From state A0, on reading input symbol *a*, the machine should change the state to A1. So, $\delta(A_0, a) = A_1$
- From state A1, on reading input symbol *a*, the machine should change the state to A2. So, $\delta(A_1, a) = A_2$
- From state A2, on reading input symbol *a*, the machine should change the state to A3. So, $\delta(A_2, a) = A_3$
- From state A3, on reading input symbol *a*, the machine should change the state to A4. So, $\delta(A_3, a) = A_4$
- From state A4, on reading input symbol *a*, the machine should change the state to A0. So, $\delta(A_4, a) = A_0$

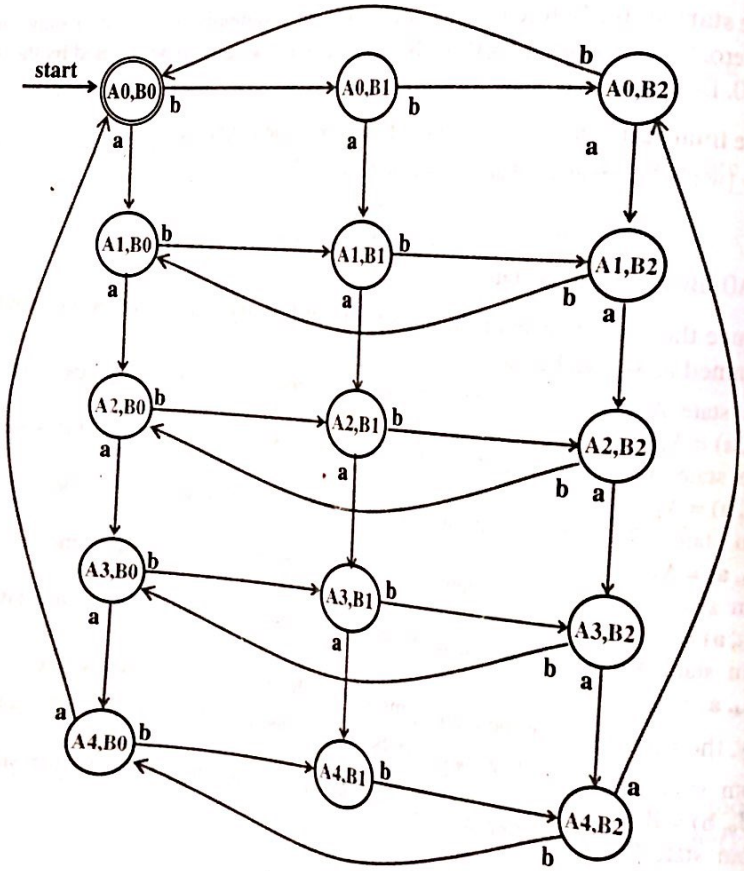
Similarly, the transitions for the input symbol *b* can be obtained as shown below:

- From state B0, on reading input symbol *b*, the machine should change the state to B1. So, $\delta(B_0, b) = B_1$
- From state B1, on reading input symbol *b*, the machine should change the state to B2. So, $\delta(B_1, b) = B_2$

- From state B2, on reading input symbol *b*, the machine should change the state to $\delta(B_2, b) = B_0$

So, the DFA M = (Q, Σ, δ, q0, F) is defined as:

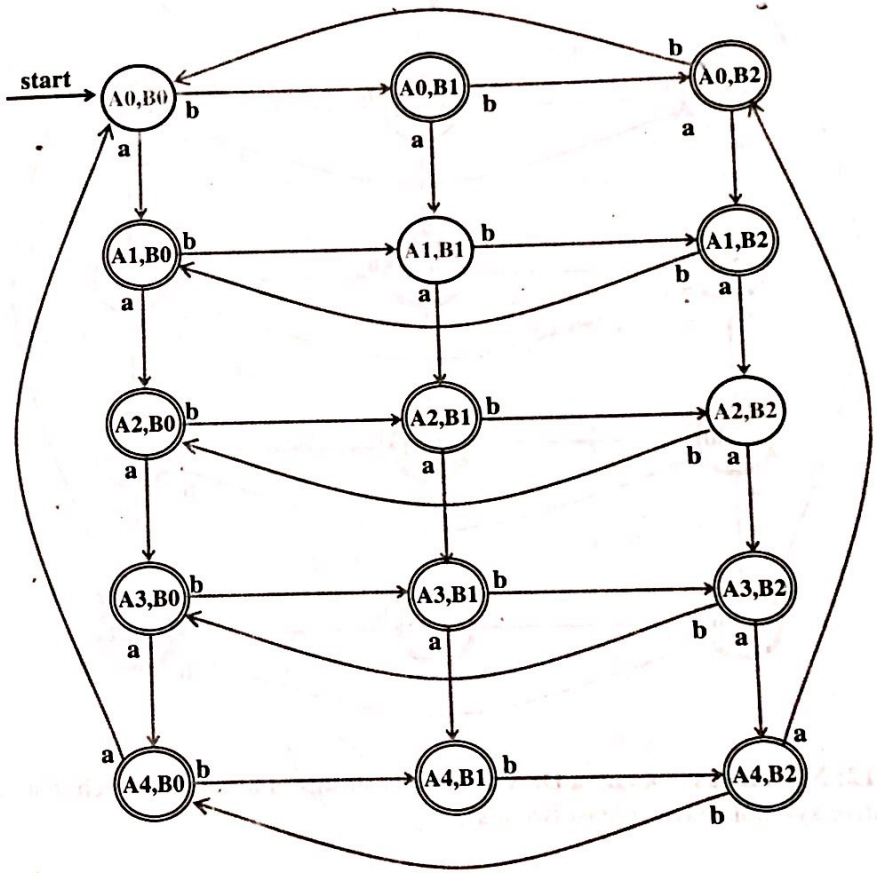
- Q = {(A0, B0), (A0, B1), (A0, B2),
  (A1, B0), (A1, B1), (A1, B2),
  (A2, B0), (A2, B1), (A2, B2),
  (A3, B0), (A3, B1), (A3, B2),
  (A4, B0), (A4, B1), (A4, B2) }
- Σ = {a, b}
- q_0 = (A0, B0) is the start state
- F = {(A0, B0)}
- δ = shown below using the transition diagram



Now, let us "Construct a DFA to accept the following language":

$L = \{w : n_a(w) \bmod 5 \neq n_b(w) \bmod 3\}$

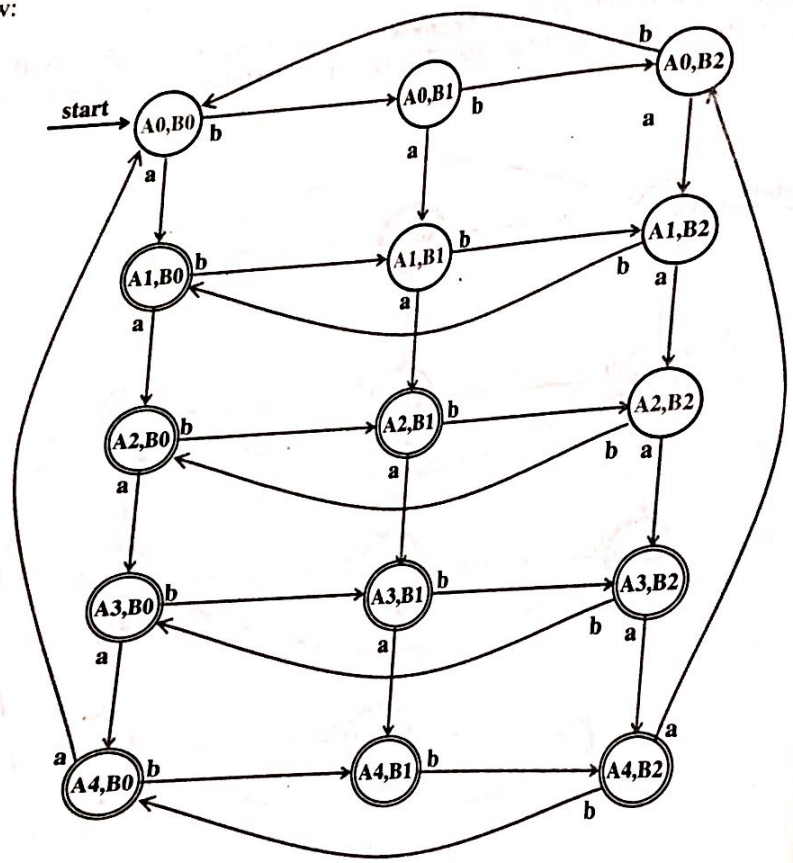
Note: The design is exactly same as the above problem. Except the states (A0, B0), (A1, B1) and (A2, B2), the rest of the states are final states. The DFA to accept the given language is shown below:



Now, let us "Construct a DFA to accept the following language":

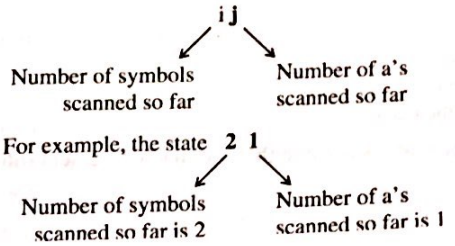
$L = \{w : n_a(w) \bmod 5 > n_b(w) \bmod 3\}$

Note: The design is exactly similar to the above problem. But, in all possible states (A_i, B_j) index i should be greater than index j. So, the final DFA to accept the above language is s below:



Example 12: *Now, let us "Obtain a DFA to accepts strings of a's and b's such that each blo of 5 consecutive symbols have at least two a's".*

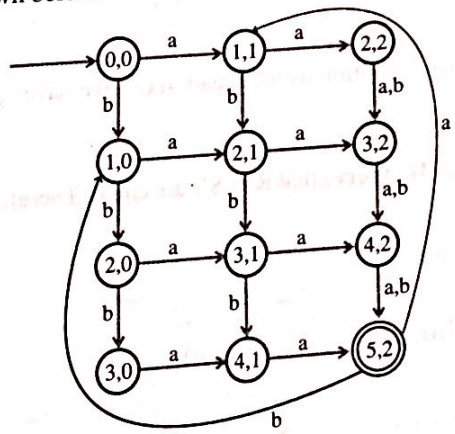
Solution: *This DFA should keep track of number of symbols scanned for and number of a Hence, each state of DFA is represented as shown below:*



The transition can be obtained using the following relationships:

- The state 00 is the start state of DFA.
- If $i \leq 4$ and $j \leq 1$ then $\delta(ij, a) = \delta(i+1\ \ j+1)$.
- If $i \leq 4$ and $j = 2$ then $\delta(ij, x) = \delta(i+1\ \ j)$ where x can be either a or b.
- 52 indicates that current block has 5 symbols and has at least 2 a's. So, the string has to be accepted and 52 is the final state.
- 50 and 51 states indicate that the block has 5 symbols and at least 2 a's are not present. So, the string has to be rejected and represent the trap state.
- $\delta(52, a) = 11$ indicate that the beginning of the next block has scanned one symbol and has one a
- $\delta(52, b) = 10$ indicate that the beginning of the next block has scanned one symbol and has no a

The complete DFA is shown below:



Example 13: Now, let us "Prove that $\delta^*(q, xa) = \delta(\delta^*(q, x), a)$" where x is a string and a is the current input symbol.

Proof: It is required to prove that $\delta^*(q, xa) = \delta(\delta^*(q, x), a)$. From this statement it is observed that:

$w = xa$

where

- a is the last symbol of w
- x is the remaining string of w
- q is the current state of the machine

From state q on input string w let us assume that the machine enters into state p. This is given by the transition:

$$\delta^*(q, w) = p \quad (1)$$

Since w = xa, the above transition can be written as shown below:

$$\delta^*(q, xa) = p \quad (2)$$

Since w = xa, first, we should find the transition on string x and then on symbol a.

On string x: From state q on input string x, let the machine enters into state r. This is denoted by:

$$\delta^*(q, x) = r \quad (3)$$

On symbol a: From state r on input symbol a, the machine should enter into state p. This is denoted by:

$$\delta(r, a) = p$$

Replacing the state r in above transition using equation (3), we have:

$$\delta(\delta^*(q, x), a) = p \quad (4)$$

Comparing equation (2) and (4) observe that R.H.S's are equal. Therefore, L.H.S.'s are also equal. So,

$$\delta^*(q, xa) = \delta(\delta^*(q, x), a)$$

$$\boxed{\text{i.e., } \delta^*(q, w) = \delta^*(q, xa) = \delta(\delta^*(q, x), a)}$$

Example 14: Show that $\delta^*(q, xy) = \delta^*(\delta^*(q, x), y)$ for any state w and strings x and y.

Note: The symbol δ^* can also be denoted by the symbol $\hat{\delta}$.

Solution: The statement to be proved is:

$$\delta^*(q, xy) = \delta^*(\delta^*(q, x), y) \quad (1)$$

Let us proceed by induction on the length of y.

Basis: Let $\delta(q, x) = p$

If $y = \varepsilon$, L.H.S. of Eq. (1) $= \delta^*(q, xy)$ (2)

$= \delta^*(q, x)$ [By replacing y by ε]

$= p$ [from Eq. (2)]

Now, consider R.H.S. of Eq. (1) $= \delta^*(\delta^*(q, x), y)$

$= \delta^*(p, y)$ [By relation (2)]

$= \delta^*(p, \varepsilon)$ [Replacing y by ε]

$= p$ [According to the property: $\delta(q, \varepsilon) = q$]

Since, L.H.S. is same as R.H.S the relation (1) holds good.

Induction: Let us split the string y into za i.e., $y = za$ (3)

a is last symbol of y

Consider R.H.S. of relation (1) $= \delta^*(\delta^*(q, x), y)$

$= \delta^*(\delta^*(q, x), za)$ Substituting relation (3)

$= \delta(\delta^*(\delta^*(q, x), z), a)$ Treat $\delta^*(q,x)$ as a state

$= \delta(\delta^*(q, xz), a)$ Inductive hypothesis

$= \delta(\delta^*(q, xz), a)$ Inductive hypothesis

$= \delta^*(q, xza)$ By definition

$= \delta^*(q, xy)$ Using relation (3)

= L.H.S. of relation (1)

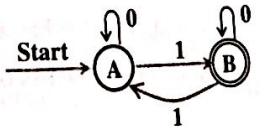
Hence, the proof.

Example 15: Consider the DFA with following transition table:

δ	0	1
→ A	A	B
*B	B	A

Informally describe the language accepted by this DFA and prove by induction on the length of an input string, that your description is correct.

Solution: The transition diagram corresponding to the transition table is shown below:



State A: The machine in state A can consume any number of 0's. But, the number of 1 consumed at state A is 0 which is EVEN. So, from EVEN state on consuming a 1, the machine enters into ODD state called B.

State B: In ODD state B, the machine can consume any number of 0's. But, if the input symbol is 1, the machine goes to state A which consumes odd number of 1's.

Informal description of language: So, the language accepted by the machine is "Strings of 0 and 1's with odd number of 1's".

Proof: It is required to show by induction that $\delta^*(A, w) = A$ if and only if w has even number of 1's.

Basis: $|w| = 0$. Since w is an empty string, surely has an even number of 1's, namely zero 1's, and so $\delta^*(A, w) = A$.

Induction: Let us consider a string shorter than w. So, let $w = za$ (1)

where a is either 0 or 1.

Case 1: Let a = 0. So, if w has an odd number of 1's, z also has odd number of 1's.

By the inductive hypothesis, $\delta^*(A, z) = B$. The transitions of the DFA tell us

$$\delta^*(A, w) = B$$

If w has an even number of 1's, then z also has even number of 1's. By the inductive hypothesis, $\delta^*(A, z) = A$. The transitions of the DFA tell us

$$\delta^*(A, w) = A$$

Thus, in this case, $\delta^*(A, w) = A$ if and only if w has an even number of 1's.

$\delta^*(A, w) = B$ if and only if w has an odd number of 1's.

Case 2: Let a = 1. So, if w has an even number of 1's, then z has an odd number of 1's. By the inductive hypothesis, $\delta^*(A, z) = B$. The transitions of the DFA tell us

$$\delta^*(A, w) = A$$

If w has an odd number of 1's, then z has an even number of 1's. By the inductive hypothesis, $\delta^*(A, z) = A$. The transitions of the DFA tell us

$$\delta^*(A, w) = B$$

Thus, in this case as well, $\delta^*(A, w) = A$ if and only if w has an even number of 1's

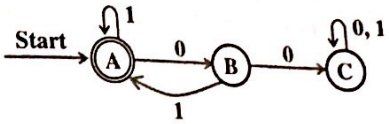
$\delta^*(A, w) = B$ if and only if w has odd number of 1's.

Example 16: Consider the DFA with following transition table:

δ	0	1
→*A	B	A
*B	C	A
C	C	C

Informally describe the language accepted by this DFA and prove by induction on the length of an input string, that your description is correct.

Solution: The transition diagram corresponding to the transition table is shown below:



Observe the following facts from the above transition diagram:

- A string w having either ε or ends in 1 but does not having the substring 00 is consumed at state A
- A string w ends in 0 and does not have the substring 00 is consumed at state B
- A string w that contains the substring 00 is consumed at state C

Informal description of the language: Since A is the final state, the language accepted by DFA is "Strings of 0's and 1's having ε or ending with 1 but does not have a substring 00".

Proof: To start with machine will be in start state A.

Basis: If w = ε, the machine will be in state A which is the final state. Thus, empty string is accepted by DFA.

Induction: If w has only 1's the machine will be in state A. So, w1 does not contain the substring 00 and hence the string is accepted by DFA.

If w ends with 0, the machine goes to state B and hence string ends with one 0. But, on 1 the machine enters into state A and thus the substring 00 is not accepted.

1.7. Applications of Finite Automata

Now, let us see "Why to study finite automata? or What are applications of finite automata?"

Some of the applications where automata play an important role are shown below:

- Design of digital circuits: The FA is used during designing and checking the behavior of the digi circuits using software. The FA is very useful in hardware design such as circuit verification design of automatic traffic signals etc.
- Compiler construction: Used in the design of *lexical analyzer* (the first phase of compiler desig which breaks the input text into various units such as identifiers, keywords, punctuation etc.
- String matching: In designing a software for identifying the words, phrases and other patterns in larg bodies of text (such as collection of web pages).
- String processing: To write software for processing the natural language (Ex: Speech processing Large natural vocabularies can be described which includes the applications such as spelling check ers and advisers, multi-language dictionaries, indenting the documents etc.
- Software design: In building the software to verify the systems having finite number of state (for example, communication protocols in computer networks).
- Other applications: The FA are used in variety of applications in Artificial intelligence and knowledg engineering, in game theory and games, computer graphics, linguistics, etc.

Exercises

1. What is DFA? Explain with an example.
2. When we say that a language is accepted by the machine? Explain with example.
3. When a given language is not accepted by DFA? Explain with example.
4. How DFA's can be represented? Explain with example.
5. What is a transition diagram/graph?
6. Obtain a DFA to accept strings of a's and b's starting with the string ab.
7. Draw a DFA to accept string of 0's and 1's ending with the string 011.
8. Obtain a DFA to accept strings of a's and b's having a sub string aa.
9. Obtain a DFA to accept strings of a's and b's except those containing the substring aab.
10. Obtain DFAs to accept strings of a's and b's having exactly one a, atleast one a, not more than three a's.
11. Obtain a DFA to accept the language L = {awa | w ∈ (a+b)* }.
12. Obtain a DFA to accept even number of a's, odd number of a's.
13. Obtain a DFA to accept strings of a's and b's having even number of a's and b's.
14. Obtain a DFA to accept odd number of a's and even number of b's.
15. Obtain a DFA to accept even number of a's and odd number of b's.

16. Obtain a DFA to accept strings of a's and b's having odd number of a's and b's.
17. Obtain a DFA to accept strings of 0's, 1's and 2's beginning with a '0' followed by odd number o 1's and ending with a '2'.
18. Obtain a DFA to accept binary odd numbers.
19. Obtain a DFA to accept strings of a's and b's starting with at least two a's and ending with at lea two b's.
20. Obtain a DFA to accept strings of a's and b's with at most two consecutive b's.
21. Obtain a DFA to accept the language L = { w : |w| mod 3 = 0 on Σ = {a, b}.
22. Obtain a DFA to accept the language L = { w : |w| mod 5 ≠ 0 on Σ = {a, b}.
23. What is a regular language?
24. What are the applications of finite automaton?

1.8. Disadvantages of DFA

Now, let us see "What are the various disadvantages of DFA?" The various disadvantages of DFA are listed below:

- Constructing a DFA is difficult
- The DFA cannot guess about its input
- The DFA is not very powerful
- At any point of time, the DFA is in only state. So, a DFA does not have the power to be in several states at once

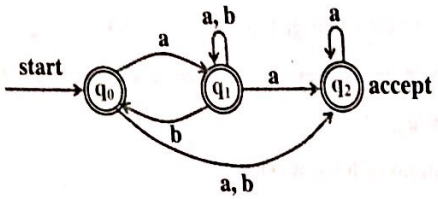
1.9. Why NFA?

Computers are completely deterministic machines (DFA). The state of the computer can be predicted from the input and initial state. We cannot find a computer which is non-deterministic. In such case, the question is "Why non-deterministic Finite Automata?" All the disadvantages of DFA mentioned earlier can be overcome using Non-Deterministic Finite Automata (in short NFA or NDFA). The various advantages of NFA are:

- Very easy to construct

- A "non-deterministic" finite automaton has the ability to guess something about its input
- A "non-deterministic" finite automaton is more powerful than DFA
- It has the power to be in several states at once
- An NFA is an efficient mechanism to describe some complicated languages concisely

Before defining NFA, let us consider the following transition diagram:



Let us see "What is the specialty of the above finite automaton?" Observe the following facts:

- From a given state there is possibility of zero, one or more edges leaving that state after consuming the input symbol.

Example 1: From state q_2, there are zero edges (indicates no edges) for the input symbol *b*.

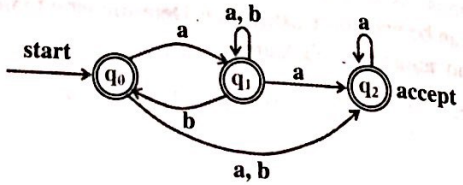
Example 2: From state q_2, there is one edge (q_2, q_2) labeled *a*.

Example 3: From state q_0, there are two edges (q_0, q_1) and (q_0, q_2) labeled *a*.

Note: In the FA shown above, there can be zero, one or more transitions on an input symbol. Such machines are called non-deterministic finite state machines or non-deterministic finite automata.

1.10. Non-Deterministic Finite Automaton

Before worrying about the definition, let us consider the following pictorial representation of NFA.



From the above figure, observe following components of NFA:

- **States:** The NFA has three states q_0, q_1 and q_2 and can be represented as

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

Note: The power set of Q is denoted by 2^Q which is set of subsets of set Q. This is denoted as shown below:

$$2^Q = \{ \{\}, \{q_0\}, \{q_1\}, \{q_0, q_1\}, \{q_0, q_2\}, \{q_1, q_2\}, \{q_0, q_1, q_2\} \}$$

- **Input alphabets:** Each edge is labeled with *a* or *b* and represent the input alphabets which can be denoted as:

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

- **Transitions:** Transition is nothing but change of state after consuming an input symbol. If there is a change of state from q_i to q_j on an input symbol *a*, then we write

$$\delta(v_i, a) = q_j$$

If there is a change of state from q_i to q_j and q_i to q_k on an input symbol *a*, then we write

$$\delta(v_i, a) = \{q_j, q_j\}$$

The transition diagram of above NFA is shown below:

Current state	Input	Next state	Representations
q_0	a	q_1, q_2	$\delta(q_0, a) = \{q_1, q_2\}$
q_0	b	q_2	$\delta(q_0, b) = q_2$
q_1	a	q_1, q_2	$\delta(q_1, a) = \{q_1, q_2\}$
q_1	b	q_0, q_2	$\delta(q_1, b) = \{q_0, q_1\}$
q_2	a	q_2	$\delta(q_2, a) = q_2$
q_2	b	q_1	$\delta(q_2, b) = q_1$

δ: Q x Σ to 2^Q [power set]

Now, let us see "What is a transition function?" The transition function δ is defined as:

$$\delta: Q \times \Sigma \text{ to } 2^Q$$

which is read as "δ is a transition function which maps Q X Σ to 2^Q". For example, the change state from state q on input symbol a to states $p_1, p_2, \ldots, p_n$ is denoted by

$$\delta(q, a) = \{ p_1, p_2, \ldots, p_n \}$$

where

- **δ:** is a function called transition function
- **q**: is the first parameter representing the current state of the machine
- **a**: is the second parameter representing the current input symbol read
- $p_1, p_2, \ldots, p_n$ represent possible states of machine

Note: In other words, the transition function δ accepts two parameters namely state q and input symbol a as the parameters and returns set of states the machine enters into.

- **Start state (q_0):** q_0 with the label start is treated as the start state.
- **Final state (q_2):** q_2 with two circles is treated as the final or accept state.

Note: From this discussion, it is observed that the NFA has five components:

(states, input alphabets, transitions, start state, final states)

↓ ↓ ↓ ↓ ↓

Q Σ δ q_0 F

With this concept, now let us see "What is a non-deterministic finite automaton (NFA)?"

❖ **Definition:** The non-deterministic finite automaton in short NFA is 5-tuple or quintuple indicating five components:

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

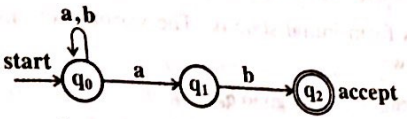
where

- **M** is the name of the machine. It can also be called by any name.
- **Q** is non-empty, finite set of states.
- **Σ** is non-empty, finite set of input alphabets.
- **δ:** Q X Σ to 2^Q i.e. δ is transition function which is a mapping from Q X Σ to 2^Q. Based on the current state and input symbol, the machine enters into one or more states.

- $q_0 \in Q$ – is the start state.
- $F \subseteq Q$ – is set of accepting or final states.

Suppose δ(q, a) is in P where P is in 2^Q and a is in Σ. This indicates that there is a transition from state q on an input symbol a to set of states P. Note: In an NFA there can be zero, one or more transitions on an input symbol.

For example, an NFA to accept strings of a's and b's ending with *ab* is shown below:



The above NFA can be specified formally as M = (Q, Σ, δ, q_0, F) where

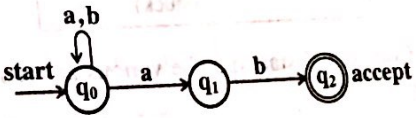
- Q = {q_0, q_1, q_2}
- Σ = {a, b}
- q_0 is the start state
- F = {q_2}
- δ is shown below using the transition table:

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

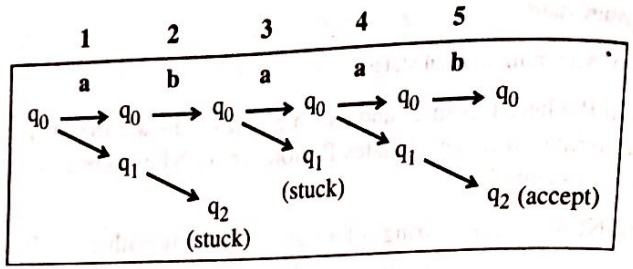
Note: In the transition table of NFA, each entry in the table should be denoted by a set. Also, when there is no transition at all from a given state on an input symbol, make an entry ϕ indicating an empty set.

1.10.1. Moves made by NFA

Now, let us see "What are the states an NFA is in during the processing of input sequence abaab and abb?".



Solution: The states an NFA is in during the processing of input sequence *abaab* is shown below:



The machine always starts from initial state q_0. The various actions performed by NFA for string *abaab* is shown below:

1) From q_0 on reading *a*, the NFA may go to q_0 **or** q_1
2) q_0 on *b* NFA goes to state q_0

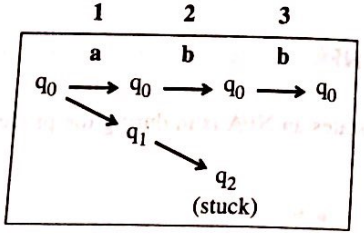
   q_1 on *b* NFA goes to state q_2
3) q_0 on *a* NFA goes to q_0 **or** q_1

   Note: q_2 on *a* there is no transition and hence NFA is stuck.
4) q_0 on *a* NFA goes to q_0 **or** q_1

   Note: q_1 on *a* there is no transition and hence NFA is stuck.
5) q_0 on *b* NFA goes to q_0. At the end, the NFA is in q_0 (non-final state). So, this path is not chosen.

   But, q_1 on *b* NFA goes to q_2. So, at the end, NFA is in q_2 which is a final state and hence the input string **abaab** is accepted by the machine.

The states an NFA is in during the processing of input sequence *abb* is shown below:



The machine always starts from initial state q_0. The various actions performed by NFA for the string abaab is shown below:

1) From q_0 on reading *a*, the NFA may go to q_0 or q_1
2) q_0 on *b* NFA goes to state q_0

   q_1 on *b* NFA goes to state q_2
3) q_0 on *b* NFA goes to state q_0

   Note: q_2 on *b* there is no transition and hence NFA is stuck.

After the string *abb*, the machine is in state q_0 which is non-final state. So the string *abb* is rejected by the machine.

1.10.2. Extended Transition Function of NFA to Strings

Note: The transition $\delta(q, a) = P$ accepts two parameters namely state *q* and input symbol *a* as parameters and returns a set of states P where $P = \{p_1, p_2, p_3, \ldots, p_n\}$ which represent the possible states of the NFA at a given instance. But, if there is a change of state from *q* to set of states P where $P\{p_1, p_2, p_3, \ldots, p_n\}$ on input string *w*, then we use extended transition function denoted by δ^*.

Note: We can also use $\hat{\delta}$ in place of δ^*. But, in our book let us use δ^* to denote extended transition. Now, let see "What is extended transition function δ^* for NFA?".

◆ **Definition:** The extended transition function δ^* describes what happens to a state of machine when the input is a string (sequence of symbols). Let $M = (Q, \Sigma, \delta, q_0, F)$ be an NFA. The extended transition function δ^*: **$Q \times \Sigma^*$ to 2^Q** is defined recursively as shown below:

- **Basis:** $\delta^*(q, \varepsilon) = \{q\}$. This indicates that if the machine is in state q and read no input, then the machine is still in state q.
- **Induction:** Let $w = xa$ where *a* is the last symbol of *w* and *x* is the remaining string of *w*. Let *q* is the current state and *x* is the string to be processed and after consuming the string *x*, let the state of the machine is $\{p_1, p_2, p_3, \ldots, p_m\}$.

  i.e. $\delta^*(q, x) = \{p_1, p_2, p_3, \ldots, p_m\}$

  Let the transition from $\{p_1, p_2, p_3, \ldots, p_m\}$ on input symbol *a* is:

  $\delta(\{p_1, p_2, p_3, \ldots, p_m\}, a) = \{r_1, r_2, r_3, \ldots, r_k\}$

  $$\text{i.e. } \bigcup_{i=1}^{m} \delta(p, a) = \{r_1, r_2, r_3, \ldots, r_k\}$$

  Then $\delta^*(q, w) = \delta^*(q, xa) = \{r_1, r_2, r_3, \ldots, r_k\}$.

Note: Thus, various properties of extended transition functions for an NFA can be:

- $\delta^*(q, \varepsilon) = \{q\}$
- $\delta^*(q, w) = \delta^*(q, xa) = \{\delta(\{\delta^*(q, x)\}, a\}$ where $w = xa$
- $\delta^*(q, w) = \delta^*(q, ax) = \{\delta^*(\{\delta(q, x)\}, x\}$ where $w = ax$

For example, change of state from state q on input string w to set of states $\{r_1, r_2, r_3, \ldots, r_k\}$ denoted by

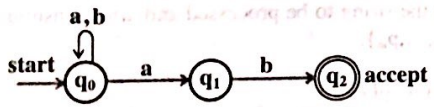
$$\delta^*(q, w) = \{r_1, r_2, r_3, \ldots, r_k\}$$

where

- δ^*: is a function called as extended transition function
- q: is the first parameter representing the current state of the machine
- w: is the second parameter representing the current input string being read
- $\{r_1, r_2, r_3, \ldots, r_k\}$: represent the possible states of the machine which is returned by the transition function

Note: The transition function δ^* accepts two parameters namely state q and input string w as the parameters and returns set of states of the machine.

Now, let us see "What are the moves made by the following NFA while processing the string *abaab* using the extended transition function?".



Solution: The moves made by the DFA for the input string *abaab* using δ^* is obtained starting from ε and taking prefix of *abaab* in increasing size as shown below:

- For prefix ε: $\delta^*(q_0, \varepsilon) = \{q_0\}$ .....(1)
- For prefix a: $\delta^*(q_0, a) = \delta(\delta^*(q_0, \varepsilon), a\}$ Substituting (3)
  $= \delta(q_0, a)$
  $= \{q_0, q_1\}$ .....(2)

- For prefix ab: $\delta^*(q_0, ab) = \delta(\delta^*(q_0, a), b\}$
  $= \delta(\{q_0, q_1\}, b)$ Substituting (2)
  $= \delta(q_0, b) \cup \delta(q_1, b)$
  $= \{q_0\} \cup \{q_1\}\ \phi$
  $= \{q_0, q_2\}$ .....(3)
- For prefix aba: $\delta^*(q_0, aba) = \delta(\delta^*(q_0, ab), a\}$
  $= \delta(\{q_0, q_2\}, a)$ Substituting (3)
  $= \delta(q_0, a) \cup \delta(q_2, a)$
  $= \{q_0, q_1\} \cup \phi$
  $= \{q_0, q_1\}$ .....(4)
- For prefix abaa: $\delta^*(q_0, abaa) = \delta(\delta^*(q_0, aba), a\}$
  $= \delta(\{q_0, q_1\}, a)$ Substituting (3)
  $= \delta(q_0, a) \cup \delta(q_1, a)$
  $= \{q_0, q_1\} \cup \phi$
  $= \{q_0, q_1\}$ .....(5)
- For prefix abaab: $\delta^*(q_0, abaab) = \delta(\delta^*(q_0, abaa), b\}$ Substituting (4)
  $= \delta(\{q_0, q_1\}, b)$
  $= \delta(q_0, b) \cup \delta(q_1, b)$
  $= \{q_0\} \cup \{q_2\}$
  $= \{q_0, q_1\}$

Note: After the string abaab, the states of NFA = $\{q_0, q_2\}$. Since the possible states of NFA after consuming abaab has one final state, the string abaab is accepting by the machine.

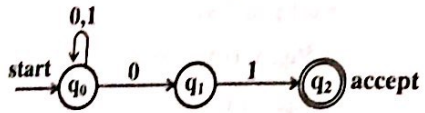
1.10.3. Language Accepted by a NFA

Now, let us "Define the language accepted by NFA". The language accepted by NFA is formally be defined as follows:

❖ **Definition:** Let $M = (Q, \Sigma, \delta, q_0, F)$ be an NFA. A string w is accepted by the machine M if it takes the initial state q_0 to final state, i.e., $\delta^*(q_0, w)$ is in F. Thus, the languages accepted by M represented as $L(M)$ can be formally written as:

$$L(M) = \{w \mid w \in \Sigma^* \text{ and } \delta^*(q_0, w) \text{ is in } F\}$$

Now, let us "Prove formally that the following NFA accepts the language $L = \{w: w$ e... $01\}$".



Proof: It is given that the above NFA accepts the language:

$$L = \{w: w \text{ ends in } 01\}$$

The statement can be proved by mutual induction on the length of string w accepted by the ... states shown in three cases as shown below:

- Case 1: $\delta^*(q_0, w) = q_0$ for every w
- Case 2: $\delta^*(q_0, w) = q_1$ for every w ending in 0
- Case 3: $\delta^*(q_0, w) = q_2$ for every w ending in 01

Basis: Consider a string $w = \varepsilon$ where $|w| = 0$.

Case 1: $\delta(q_0, \varepsilon) = q_0$ by definition. Hence case 1 is proved.

Case 2: $\delta(q_0, \varepsilon) = q_0$ by definition. Here, ε do not end with 0 and hence $\delta(q_0, \varepsilon)$ does not cont... q_1. Hence case 2 is proved.

Case 3: $\delta(q_0, \varepsilon) = q_0$ by definition. Here, ε do not end with 01 and hence $\delta(q_0, \varepsilon)$ does not cont... q_2. Hence case 3 is proved.

Induction hypotheses: Let $w = xa$ where a is either 0 or 1 and assume the three statements ca... 1 through case 2 holds good for x.

Let $|x| = n$. So, $|w| = n+1$

We have to prove that the three cases mentioned above are true for $n + 1$.

Proof: The proof for all the three cases can be obtained as shown below:

Case 1: $\delta^*(q_0, w) = q_0$. This statement is true. Because, on any of the input symbols 0 and 1 the NFA may stay in q_0. Hence, case 1 is proved.

Case 2: Assume w ends in 0, i.e., a = 0. We proved that $\delta^*(q_0, w) = q_0$. So, for the string x can write $\delta^*(q_0, x) = q_0$. Since there is a transition from q_0 to q_1 on 0, we conclude that $\delta^*(q_0, x) = q_1$. Hence, case 2 is proved.

Case 3: Assume w ends in 01. Let w = xa where $a = 1$ and x is a string such that $\delta^*(q_0, x) = q_1$ where the string ends with 0. Since there is a transition from q_1 to q_2 on 1, we conclude that $\delta^*(q_0, w) = q_2$. Hence, case 3 is proved.

Note: Thus, the different properties of the transition function with respect to NFA are:

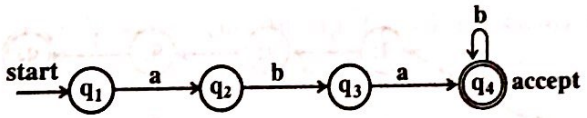
$$\delta(q, \varepsilon) = \delta^*(q, \varepsilon) = q$$
$$\delta(q, wa) = \delta(\delta^*(q, w), a) = P_j$$
$$\delta(q, aw) = \delta^*(\delta(q, a), w) = P_q$$

where q is in Q, a is im Σ, w is in Σ^* and P_j and P_q are the set of states which are reachable from q.

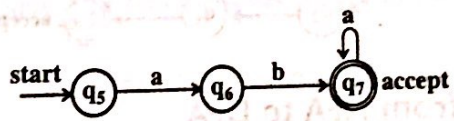
Example 1: Obtain an NFA to accept the following language:

$$L = \{w \mid w \in abab^n \text{ or } aba^n \text{ where } n \geq 0\}$$

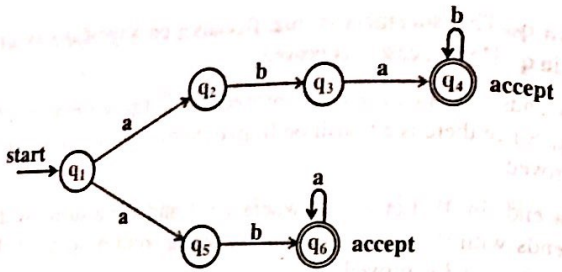
Solution: The machine to accept $abab^n$ where $n \geq 0$ is shown below:



The machine to accept aba^n where $n \geq 0$ is shown below:

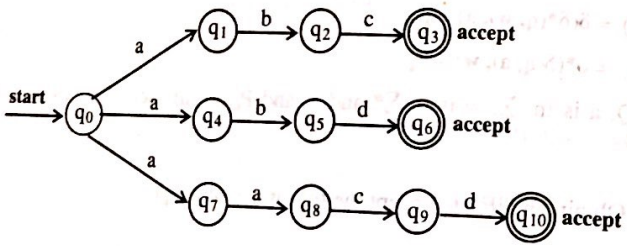


Since both the machines accept a as the first input symbol, the states q_1 and q_5 can be merged into a single state and the machine to accept either $abab^n$ or aba^n where $n \geq 0$ is shown below:



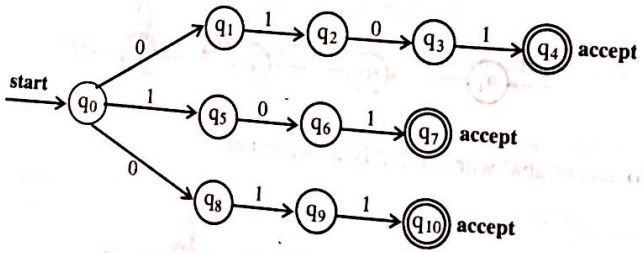
Example 1: Design an NFA to recognize the following set of strings abc, abd and aacd.

Solution: An NFA to recognize the following set of strings abc, abd and aacd is shown below.



Example 2: Obtain an NFA to recognize the following set of strings 0101, 101 and 011.

Solution: An NFA to recognize the following set of strings 0101, 101 and 011 is shown below.



1.11. Conversion from NFA to DFA

Digital computers are deterministic machines. Given the input, the state of the machine is predictable. Sometimes, constructing DFA is difficult compared to NFA. So, given any problem we construct

a NFA. This is an efficient mechanism to describe some complicated languages concisely. Practically, non-deterministic machines will not exist. So, we convert an NFA in to a DFA. Now, let us see "What are the methods using which an NFA can be converted into a DFA?". The NFA can be converted into DFA using two methods:

- Subset construction
- Lazy evaluation

1.11.1. Conversion from NFA to DFA (Subset Construct Method)

Now, let us "Describe the subset construction procedure to convert an NFA into a DFA"

Procedure NFA_DFA: Given an NFA $M_N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ which accepts the language $L(M_N)$, we can find an equivalent DFA $M_D = (Q_D, \Sigma, \delta_D, q_0, F_D)$ such that $L(M_D) = L(M_N)$.

Step 1: Identify the start state of DFA: Since q_0 is the start state of NFA, $\{q_0\}$ is the start state of DFA.

Step 2: Identify the alphabets of DFA: The input alphabets of DFA are the input alphabets of NFA. So, $\Sigma = \{a, b\}$.

Step 3: Identify Q_D which are the states of DFA: The set of subsets of Q_N will be the states of DFA Q_D. So, if Q_N has n states then Q_D will have 2^n states. For example,

Let $Q_N = \{q_0, q_1, q_2\}$ then $|Q_N| = 3$

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

1 2 3 4 5 6 7 8

So, $|Q_D| = 8$

Note: In this example, the number of states of NFA = 3 and hence number of states of DFA = 8. If n is number of states of NFA the number of states of DFA will be 2^n.

In general, $\{q_i, q_j, \ldots, q_k\}$ is considered as a state in Q_D.

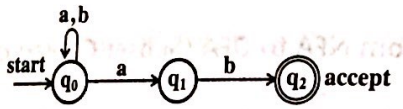
Step 4: Identify the final states of DFA: If $\{q_i, q_j, \ldots, q_k\}$ is a state in Q_D, then $\{q_i, q_j, \ldots, q_k\}$ will be final state of DFA provided one of $q_i, q_j, \ldots, q_k$ is the final state of NFA.

Step 5: Identify the transitions (i.e., δ_D) of DFA: For each state $\{q_i, q_j, \ldots, q_k\}$ in Q_D and for input symbol a in Σ, the transition can be obtained as shown below:

$$\delta_D(\{q_i, q_j, \ldots, q_k\}, a) = \delta_N(q_i, a) \cup \delta_N(q_j, a) \cup \ldots \delta_N(q_k, a)$$

Thus, DFA can be obtained using subset construction method.

Example: Now, let us "Obtain the DFA for the following NFA using subset construction method".



Solution: The transition table for the above DFA can be written as shown below:

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

Step 1: Identify the start state of DFA: Since q_0 is the start state of NFA, $\{q_0\}$ is the start state of DFA.

Step 2: Identify the alphabets of DFA: The input alphabets of NFA are the input alphabets of DFA. So, $\Sigma = \{a, b\}$.

Step 3: Identify Q_D which are the states of DFA: Here, $QN = \{q_0, q_1, q_2\}$. Its subsets are the states of DFA. So, states of DFA are:

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

Step 4: Identify the final states of DFA: Since q_2 is the final state of NFA in the above set, wherever q_2 is present as an element, the corresponding set is the final state of DFA. So

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

Step 5: Identify the transitions (i.e., δ_D) of DFA: Obtain the transitions for each of the states of Q_D obtained in step 3 as shown below:

For state ϕ:

Input symbol = a

$\delta_D(\phi, a) = \phi$

Input symbol = b

$\delta_D(\phi, b) = \phi$

For state $\{q_0\}$:

Input symbol = a

$\delta_D(\{q_0\}, a) = \{q_0, q_1\}$

Input symbol = b

$\delta_D(\{q_0\}, b) = \{q_0\}$

For state $\{q_1\}$:

Input symbol = a

$\delta_D(\{q_1\}, a) = \phi$

Input symbol = b

$\delta_D(\{q_1\}, b) = q_2$

For state $\{q_2\}$:

Input symbol = a

$\delta_D(\{q_2\}, a) = \phi$

Input symbol = b

$\delta_D(\{q_2\}, b) = \phi$

For state $\{q_0, q_1\}$:

Input symbol = a

$$\delta_D(\{q_0, q_1\}, a) = \delta_N(\{q_0, q_1\}, a)$$
$$= \delta_N(q_0, a) \cup \delta_N(q_1, a)$$
$$= \{q_0, q_1\} \cup \phi$$
$$= \{q_0, q_1\}$$

Input symbol = b

$$\delta_D(\{q_0, q_1\}, b) = \delta_N(\{q_0, q_1\}, b)$$
$$= \delta_N(q_0, b) \cup \delta_N(q_1, b)$$
$$= \{q_0\} \cup \{q_2\}$$
$$= \{q_0, q_2\}$$

For state $\{q_0, q_2\}$:

Input symbol = a

$$\delta_D(\{q_0, q_2\}, a) = \delta_N(\{q_0, q_2\}, a)$$
$$= \delta_N(q_0, a) \cup \delta_N(q_2, a)$$
$$= \{q_0, q_1\} \cup \phi$$
$$= \{q_0, q_1\}$$

Input symbol = b

$$\delta_D(\{q_0, q_2\}, b) = \delta_N(\{q_0, q_2\}, b)$$
$$= \delta_N(q_0, b) \cup \delta_N(q_2, b)$$
$$= \{q_0\} \cup \phi$$
$$= \{q_0\}$$

For state $\{q_1, q_2\}$:

Input symbol = a

$$\delta_D(\{q_1, q_2\}, a) = \delta_N(\{q_1, q_2\}, a)$$
$$= \delta_N(q_1, a) \cup \delta_N(q_2, a)$$
$$= \phi \cup \phi$$
$$= \phi$$

Input symbol = b

$$\delta_D(\{q_1, q_2\}, b) = \delta_N(\{q_1, q_2\}, b)$$
$$= \delta_N(q_1, b) \cup \delta_N(q_2, b)$$
$$= \{q_2\} \cup \phi$$
$$= \{q_2\}$$

For state $\{q_0, q_1, q_2\}$:

Input symbol = a

$$\delta_D(\{q_0, q_1, q_2\}, a) = \delta_N(\{q_0, q_1, q_2\}, a)$$
$$= \delta_N(q_0, a) \cup \delta_N(q_1, a) \cup \delta_N(q_2, a)$$
$$= \{q_0, q_1\} \cup \phi \cup \phi$$
$$= \{q_0, q_1\}$$

Input symbol = b

$$\delta_D(\{q_0, q_1, q_2\}, b) = \delta_N(\{q_0, q_1, q_2\}, b)$$
$$= \delta_N(q_0, b) \cup \delta_N(q_1, b) \cup \delta_N(q_2, b)$$
$$= \{q_0\} \cup \{q_2\} \cup \phi$$
$$= \{q_0, q_2\}$$

Now, all the above transitions can be represented using transition table as shown below:

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

By renaming the states of DFA as A, B, C, D, E, F, G, H

δ	a	b
A	A	A
→ B	E	B
C	A	D
*D	A	A
E	E	F
*F	E	B
*G	A	D
*H	E	F

Note: Even though we have 8 states, observe from the table that B is the start state. The states reachable from B are B, E, F. The rest of the states are not reachable and hence can be eliminated.

1.11.2. Disadvantage of Subset Construction Method

Now, let us see "What are the disadvantages of subset construction method?". Observe that from the above table that there are 2^n states and from each state we have the transition from input symbol in Σ and hence the time complexity to convert an NFA to DFA is $\Sigma * 2^n$. Since the time complexity is exponential, the procedure takes very long time to construct the table. This exponential time complexity can be avoided using another technique called "Lazy evaluation" on the subsets.

1.11.3. Conversion from NFA to DFA (Lazy Evaluation Method)

Now, let us "Describe the lazy evaluation method to convert NFA into a DFA". The lazy evaluation method of obtaining a DFA from NFA is shown below:

Procedure NFA_DFA: Given an NFA $M_N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ which accepts the language L(MN), we can find an equivalent DFA $M_D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$ such that $L(M_D) = L(M_N)$.

Step 1: Identify the start state of DFA: Since q_0 is the start state of NFA, $\{q_0\}$ is the start state of DFA.

Step 2: Identify the alphabets of DFA: The input alphabets of DFA are the input alphabets of NFA. So, $\Sigma = \{a, b\}$.

Step 3: Identify the transitions (i.e., δ_D) of DFA: For each state $\{q_i, q_j, \ldots, q_k\}$ in Q_D and for each input symbol a in Σ, the transition can be obtained as shown below:

$$\delta_D(\{q_i, q_j, \ldots, q_k\}, a) = \delta_N(q_i, a) \cup \delta_N(q_j, a) \cup \ldots \delta_N(q_k, a)$$

$$= [q_l, q_m, \ldots, q_n] \text{ say}$$

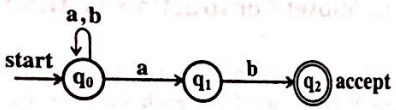
- Add the state $[q_l, q_m, \ldots, q_n]$ to Q_D, if it is not already in Q_D.
- Add the transitions from $[q_i, q_j, \ldots, q_k]$ to $[q_l, q_m, \ldots, q_n]$ on the input symbol *a*

Note: The step 3 has to be repeated for each state that is added to Q_D.

Step 4: Identify the final states of DFA: If $\{q_i, q_j, \ldots, q_k\}$ is a state in Q_D and if one of $q_i, q_j, \ldots$ is the final state of NFA, then $\{q_i, q_j, \ldots, q_k\}$ will be the final state of DFA.

Thus, DFA can be obtained using lazy evaluation method.

Example: Now, let us "Obtain the DFA for the following NFA using lazy evaluation method"



Solution: The transition table for the above DFA can be written as shown below:

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

Step 1: Identify the start state of DFA: Since q_0 is the start state of NFA, $\{q_0\}$ is the start state of DFA.

Step 2: Identify the alphabets of DFA: The input alphabets of NFA are the input alphabets of DFA. So, $\Sigma = \{a, b\}$.

Step 3: Identify the transitions (i.e., δ_D) of DFA: Start from the start state q_0 and find the transitions as shown below:

For state $\{q_0\}$:

Input symbol = *a*

$\delta_D(\{q_0\}, a) = \{q_0, q_1\}$

Input symbol = *b*

$\delta_D(\{q_0\}, b) = \{q_0\}$

For state $\{q_0, q_1\}$:

Input symbol = *a*

$$\delta_D(\{q_0, q_1\}, a) = \delta_N(\{q_0, q_1\}, a)$$
$$= \delta_N(q_0, a) \cup \delta_N(q_1, a)$$
$$= \{q_0, q_1\} \cup \phi$$
$$= \{q_0, q_1\}$$

Input symbol = *b*

$$\delta_D(\{q_0, q_1\}, b) = \delta_N(\{q_0, q_1\}, b)$$
$$= \delta_N(q_0, b) \cup \delta_N(q_1, b)$$
$$= \{q_0\} \cup \{q_2\}$$
$$= \{q_0, q_2\}$$

For state $\{q_0, q_2\}$:

Input symbol = *a*

$$\delta_D(\{q_0, q_2\}, a) = \delta_N(\{q_0, q_2\}, a)$$
$$= \delta_N(q_0, a) \cup \delta_N(q_2, a)$$
$$= \{q_0, q_1\} \cup \phi$$
$$= \{q_0, q_1\}$$

Input symbol = *b*

$$\delta_D(\{q_0, q_2\}, b) = \delta_N(\{q_0, q_2\}, b)$$
$$= \delta_N(q_0, b) \cup \delta_N(q_2, b)$$
$$= \{q_0\} \cup \phi$$
$$= \{q_0\}$$

Since, no new state is generated this step is terminated.

Step 4: Identify the final states of DFA: Since q_2 is the final state of NFA in the above set, wherever q_2 is present as an element, the corresponding set is the final state of DFA. So, the final state is $\{q_0, q_2\}$.

Now, all the above transitions can be represented using transition table as shown below.

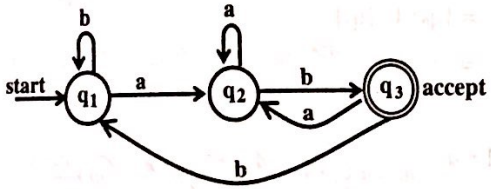
δ	a	b
→ {q_0}	{q_0,q_1}	{q_0}
{q_0,q_1}	{q_0,q_1}	{q_0,q_2}
*{q_0,q_2}	{q_0,q_1}	{q_0}

By renaming the states of DFA as A, B, C

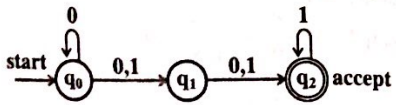
δ	a	b
→ A	B	A
B	B	C
*C	B	A

So, the final DFA is given by M = (Q, Σ, δ, q_0, F) where

- Q = {A, B, C}
- Σ = {a, b}
- q_0 = A
- F = {C}
- δ is shown below using the transition table:



Example: Now, let us "convert the following NFA to its equivalent DFA".



Solution: The transition table for the above DFA can be written as shown below:

δ	0	1
→ q_0	{q_0,q_1}	{q_1}
*q_1	{q_2}	{q_2}
q_2	ϕ	{q_2}

Step 1: Identify the start state of DFA: Since q_0 is the start state of NFA, {q_0} is the start state of DFA.

Step 2: Identify the alphabets of DFA: The input alphabets of NFA are the input alphabets of DFA. So, Σ = {0, 1}.

Step 3: Identify the transitions (i.e., δ_D) of DFA: Start from the start state q_0 and find the transitions as shown below:

For state {q_0}:

Input symbol = *0*

$\delta_D(\{q_0\}, a) = \delta_N(\{q_0\}, 0\})$

$= \{q_0, q_1\}$

Input symbol = *1*

$\delta_D(\{q_0\}, 1) = \delta_N(\{q_0\}, 1)$

$= \{q_1\}$

For state {q_0, q_1}:

Input symbol = *0*

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

$= \delta_N(q_0, 0) \cup \delta_N(q_1, 0)$

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

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

Input symbol = *1*

$\delta_D(\{q_0, q_1\}, 1) = \delta_N(\{q_0, q_1\}, 1)$

$= \delta_N(q_0, 1) \cup \delta_N(q_1, 1)$

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

$= \{q_1, q_2\}$

The above two states are added to Q_D shown in previous step. The resulting states are shown below:

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

For state {q_1}:

Input symbol = *0*

$\delta_D(\{q_1\}, 0) = \delta_N(\{q_1\}, 0)$

$= \{q_2\}$

Input symbol = *1*

$\delta_D(\{q_1\}, 1) = \delta_N(\{q_1\}, 1)$

$= \{q_2\}$

The above two states are added to Q_D obtained in previous step so that

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

For state $\{q_0, q_1, q_2\}$:

Input symbol = *0*

$\delta_D(\{q_0, q_1, q_2\}, 0) = \delta_N(\{q_0, q_1, q_2\}, 0)$

$= \delta_N(q_0, 0) \cup \delta_N(q_1, 0) \cup \delta_N(q_2, 0)$

$= \{q_0, q_1\} \cup \{q_2\} \cup \{\phi\}$

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

Input symbol = *1*

$\delta_D(\{q_0, q_1, q_2\}, 1) = \delta_N(\{q_0, q_1, q_2\}, 1)$

$= \delta_N(q_0, 1) \cup \delta_N(q_1, 1) \cup \delta_N(q_2, 1)$

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

$= \{q_1, q_2\}$

The above two states are added to Q_D obtained in previous step so that

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

For state $\{q_1, q_2\}$:

Input symbol = *0*

$\delta_D(\{q_1, q_2\}, 0) = \delta_N(\{q_1, q_2\}, 0)$

$= \delta_N(q_1, 0) \cup \delta_N(q_2, 0)$

$= \{q_2\} \cup \phi$

$= \{q_2\}$

Input symbol = *1*

$\delta_D(\{q_1, q_2\}, 1) = \delta_N(\{q_1, q_2\}, 1)$

$= \delta_N(q_1, 1) \cup \delta_N(q_2, 1)$

$= \{q_2\} \cup \{q_2\}$

$= \{q_2\}$

The above two states are added to Q_D obtained in previous step so that

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

For state $\{q_2\}$:

Input symbol = *0*

$\delta_D(\{q_2\}, 0) = \delta_N(\{q_2\}, 0)$

$= \phi$

Input symbol = *1*

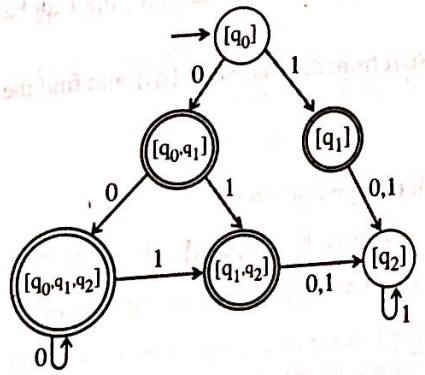
$\delta_D(\{q_2\}, 1) = \delta_N(\{q_2\}, 1)$

$= \{q_2\}$

The above two states are added to Q_D obtained in previous step so that

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

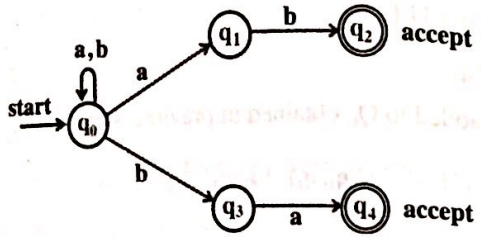
The final transition table along with transition diagram is shown below:



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

Example: now, let us "Obtain an NFA to accept strings of a's and b's ending with ab or ba. From this obtain an equivalent DFA".

Solution: The NFA to accept strings of a's and b's ending *ab* or *ba* is shown below:



The transition table for the above transition diagram is shown below:

	δ	a	b
→	q_0	$\{q_0, q_1\}$	$\{q_0, q_3\}$
	q_1	ϕ	$\{q_2\}$
	$*q_2$	ϕ	ϕ
	q_3	$\{q_4\}$	ϕ
	$*q_4$	ϕ	ϕ

Step 1: Identify the start state of DFA: Since q_0 is the start state of NFA, $\{q_0\}$ is the start s DFA. So, $Q_D = \{ \{q_0\} \}$.

Step 2: Identify the alphabets of DFA: The input alphabets of NFA are the input alphab DFA. So, $\Sigma = \{a, b\}$.

Step 3: Identify the transitions (i.e., δ_D) of DFA: Start from the start state $\{q_0\}$ and find the t tions as shown below:

For state $\{q_0\}$:

Input symbol = a

$\delta_D(\{q_0\}, a) = \{q_0, q_1\}$

Input symbol = b

$\delta_D(\{q_0\}, b) = \{q_0, q_3\}$

The above two states are added to Q_D obtained in step 1 so that

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

For state $\{q_0, q_1\}$:

Input symbol = a

$\delta_D(\{q_0, q_1\}, a) = \delta_N(\{q_0, q_1\}, a)$

$= \delta_N(q_0, a) \cup \delta_N(q_1, a)$

$= \{q_0, q_1\} \cup \phi$

$= \{q_0, q_1\}$

Input symbol = b

$\delta_D(\{q_0, q_1\}, b) = \delta_N(\{q_0, q_1\}, b)$

$= \delta_N(q_0, b) \cup \delta_N(q_1, b)$

$= \{q_0, q_3\} \cup \{q_2\}$

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

The above two states are added to Q_D so that

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

For state $\{q_0, q_3\}$:

Input symbol = a

$\delta_D(\{q_0, q_3\}, a) = \delta_N(\{q_0, q_3\}, a)$

$= \delta_N(q_0, a) \cup \delta_N(q_3, a)$

$= \{q_0, q_3\} \cup \{q_2\}$

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

Input symbol = b

$\delta_D(\{q_0, q_3\}, b) = \delta_N(\{q_0, q_3\}, b)$

$= \delta_N(q_0, b) \cup \delta_N(q_3, b)$

$= \{q_0, q_3\} \cup \phi$

$= \{q_0, q_3\}$

The above two states are added to Q_D so that

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

On similar lines, the reader is supposed to find the transitions for other states specified in Q_D. The reader is advised to verify the following answers:

$\delta_D(\{q_0, q_2, q_3\}, a) = \{q_0, q_1, q_4\}$

$\delta_D(\{q_0, q_2, q_3\}, b) = \{q_0, q_3\}$

$\delta_D(\{q_0, q_1, q_3\}, a) = \{q_0, q_1\}$

$\delta_D(\{q_0, q_1, q_4\}, a) = \{q_0, q_2, q_3\}$

The final DFA obtained along with trabsition diagram and transition table is shown be

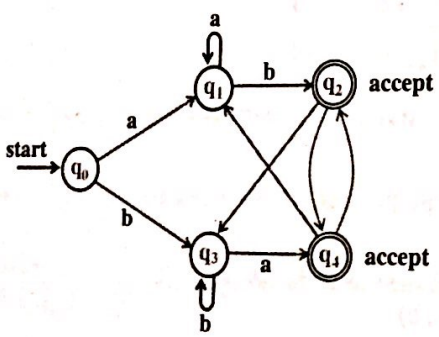
δ	a	b
→ {q0}	{q0, q1}	{q0, q3}
{q0, q1}	{q0, q1}	{q0, q2,q3}
{q0, q3}	{q0, q1,q4}	{q0, q3}
{q0, q2,q3}	{q0, q1,q4}	{q0, q3}
{q0, q2,q4}	{q0, q1}	{q0, q2,q3}

The states of the above DFA are:

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

By renaming q_0, q_1, q_2, q_3, q_4

The final transition diagram and transition table are shown below:



Transition diagram

δ	a	b
→ q0	q1	q3
q1	q1	q2
*q2	q4	q3
q3	q4	q3
*q4	q1	q2

Transition table

Now, it is observed that for every NFA there exists some DFA that accepts the same languag accepted by NFA. Now, let us "Formally prove that every NFA N can be converted into a DF such that L(D) = L(M)".

Theorem: If there exists NFA $M_N = (Q_N, \Sigma, \delta_D, q_0, F_N)$ which accepts the language $L(M_N)$, th exists an equivalent DFA $M_D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$ such that $L(M_D) = L(M_N)$.

Proof: It is required to prove that

$$\delta^*_D(q_0, w) = \delta^*_N(q_0, w)$$

We know that if Q_N represents states of NFA then power set of Q_N which contains the set of subsets of set Q_N are the states of DFA denoted by Q_D. The DFA interprets each set as a single state in DFA.

Basis: Consider a string $w = \varepsilon$ where $|w| = 0$

$\delta^*_D(\{q_0\}, \varepsilon) = \{q_0\}$ by definition of extended transition function of DFA

$\delta^*_N(\{q_0\}, \varepsilon) = \{q_0\}$ by definition of extended transition function of NFA

Hence $\delta^*_D(q_0, w) = \delta^*_N(q_0, w)$ is proved when $w = \varepsilon$.

Induction hypotheses: Now, let us assume that

$\delta^*_D(q_0, w) = \delta^*_N(q_0, w)$ for some w where $|w| = n$

Now, it is required to prove that the statement:

$\delta^*_D(q_0, w) = \delta^*_N(q_0, w)$ is true for some w where $|w| = n + 1$

Inductive proof: Let $w = xa$ where a is the last symbol of w and x is the remaining string of w:

So, $|x| = n$ and $|xa| = |w| = n + 1$

By extended definition δ* of NFA, we know that:

$$\delta^*_N(q_0, w) = \delta^*_N(q_0, xa) \quad (1)$$
$$= \delta_N(\underbrace{\delta^*_N(q_0, x)}, a)$$

Now, x is the string to be processed and after consuming the string x, let the states of the machine be $\{p_i, p_j, \ldots, p_k\}$

i.e., $\delta^*_N(q_0, x) = \{p_i, p_j, \ldots, p_k\}$

Substituting this in Eq. (1), we have

$$\delta^*_N(q_0, w) = \delta_N(\{p_i, p_j, \ldots, p_k\}, a) \quad (2)$$
$$= \delta_N(p_i, a) \cup \delta_N(p_j, a) \cup \ldots \delta_N(p_k, a)$$

By extended definition δ* of DFA, we know that (3)

$$\delta^*_N(q_0, w) = \delta^*_N(q_0, xa)$$
$$= \delta_N(\underbrace{\delta^*_N(q_0, x)}, a)$$

Now, x is the string to be processed and after consuming the string x, let the states of the ma
be $\{p_i, p_j, \ldots, p_k\}$.

i.e., $\delta^*_D(q_0, x) = \{p_i, p_j, \ldots, p_k\}$

Substituting this in Eq. (3), we have

$$\delta^*_D(q_0, w) = \delta_D(\{p_i, p_j, \ldots, p_k\}, a)$$

$$= \delta_N(p_i, a) \cup \delta_N(p_j, a) \cup \ldots \delta_N(p_k, a)$$

By comparing Eqs. (2) and (4), we have

$$\delta^*_D(\{q_0\}, w) = \delta^*_N(\{q_0\}, w)$$

So, if $\delta^*_D(\{q_0\}, w)$ is in F_N and $\delta^*_N(\{q_0\}, w)$ is in F_N, then both enters into final state accepting same language. Thus, $L(M_N) = L(M_D)$. Hence, the proof.

Theorem: Now, let us "Prove that a language L is accepted by some DFA if and only if L accepted some NFA".

Proof: The above statement can be proved as shown below:

Note: Write the subset construction method of converting an NFA to DFA + proof of previo theorem.

Since in the subset construction, all the transitions defined for NFA are also defined for DFA, t language accepted by NFA is same as the language accepted by DFA. So, *w* is accepted by DF M_D if and only if *w* is accepted by M_N, i.e., $L(M_N) = L(M_D)$. Hence the proof.

Exercises

1. What is an NFA? Explain with example.
2. What is the need for an NFA?
3. What is the difference between DFA and NFA?
4. Give a general procedure to convert as NFA to DFA.

5. Convert the following NFA into an equivalent DFA.

6. Draw an NFA to accept the string of a's and b's such that it can accept either the string consisting of one a followed by any number of a's or one b followed by any number of b's (i.e. aa* | bb*) and obtain the corresponding DFA.

Chapter 2

Finite Automata and Regular Expressions

What are we studying in this chapter . . .

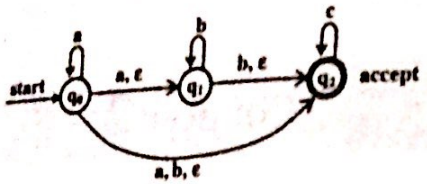
- *An application of finite automata*
- *Finite automata with ε-transitions*
- *Regular expressions*
- *Finite automata and regular expressions*
- *Applications of regular expressions*

2.1. ε-NFA (Finite Automata with Epsilon Transitions)

In this section, let us see the extended model of NFA called ε-NFA. An NFA with zero or more ε-transitions is called an ε-NFA. Now, let us see "What is an ε-transition?"

❖ **Definition:** A transition with an empty input string is called an ε-transition (read as epsilon transition). That is, if there is a transition from one state to another state without any input (i.e.,

no input implies empty string denoted by ε) is called ε-transition. For example, consider the automaton shown below:



From above FA, observe the following points:

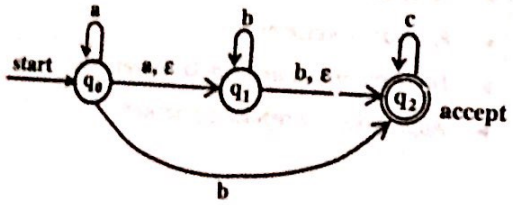
- In the above FA, we have more than one transition from state q_0 with input symbol *a*. So, it is a NFA.
- There is a transition from state q_0 to q_1 with ε as the input. There is another transition from state q_1 to q_2 with ε as the input. So, the above FA is an NFA with ε-transitions and hence it is an ε-NFA.

Note: If zero or more ε-transitions are present in a FA then it is an ε-NFA.

Note: If there is an ε-transition, then NFA makes transition without receiving any input symbol.

ε-NFA

Before worrying about the definition, let us consider the following pictorial representation of ε-NFA.



From the above figure, observe following components of ε-NFA:

- **States:** The ε-NFA has three states q_0, q_1 and q_2 and can be represented as

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

Note: The power set of Q is denoted by 2^Q which is set of subsets of set Q. This is denoted as shown below:

$$2^Q = \{ \{\}, \{q_0\}, \{q_1\}, \{q_0, q_1\}, \{q_0, q_2\}, \{q_1, q_2\}, \{q_0, q_1, q_2\} \}$$

- **Input alphabets:** Each edge is labeled with *a* or *b* and represent the input alphabets which can be denoted as:

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

- **Transitions:** Transition is nothing but change of state after consuming an input symbol. If there is a change of state from q_i to q_j on an input symbol *a*, then we write

  $\delta(q_i, a) = q_j$

If there is a change of state from q_i to q_j and q_i to q_k on an input symbol *a*, then we write

$\delta(q_i, a) = \{q_j, q_k\}$

If there is a change of state from q_i to q_j on ε (no input is read), then we write

$\delta(q_i, \varepsilon) = \{q_j\}$

The transitions for above ε-NFA is shown below:

Current State	Input	Next state	Representation
q_0	a	$\{q_0, q_1\}$	$\delta(q_0, a) = \{q_0, q_1\}$
q_0	b	$\{q_2\}$	$\delta(q_0, b) = q_2$
q_0	c	ϕ	$\delta(q_0, c) = \phi$
q_0	ε	$\{q_1\}$	$\delta(q_0, \varepsilon) = \{q_1\}$
q_1	a	ϕ	$\delta(q_1, a) = \phi$
q_1	b	$\{q_1, q_2\}$	$\delta(q_1, b) = \{q_1, q_2\}$
q_1	c	ϕ	$\delta(q_1, c) = \phi$
q_1	ε	$\{q_2\}$	$\delta(q_1, \varepsilon) = \{q_2\}$
q_2	a	ϕ	$\delta(q_2, a) = \phi$
q_2	b	ϕ	$\delta(q_2, b) = \phi$
q_2	c	$\{q_2\}$	$\delta(q_2, c) = \{q_2\}$
q_2	ε	ϕ	$\delta(q_2, \varepsilon) = \phi$

$\delta: Q \times (\Sigma \cup \varepsilon)$ to 2^Q [power set]

Now, let us see "What is a transition function for ε-NFA?". The transition function δ is defined as:

$\delta: Q \times (\Sigma \cup \varepsilon)$ to 2^Q

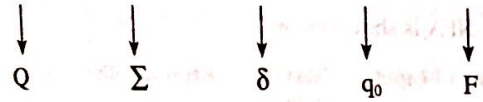
which is read as "δ is a transition function which maps Q x (Σ U ε) to 2^Q", i.e., the fu
accepts:

- a state q as the first parameter
- input symbol in Σ or ε as the second parameter and
- returns set of states the machine enters into.

- **Start state (q_0):** q_0 with the label start is treated as the start state.
- **Final state (q_2):** q_2 with two circles is treated as the final or accept state.

Note: From this discussion, it is observed that the ε-NFA has five components:

(states, input alphabets, transitions, start state, final states)



With this concept, now let us see "What is an ε-NFA?".

❖ **Definition:** The ε-NFA is 5-tuple or quintuple indicating five components:

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

where

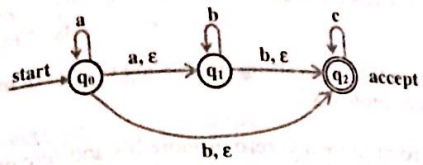
- **Q** is non-empty, finite set of states.
- **Σ** is non-empty, finite set of input alphabets.
- **δ** : Q x (Σ U ε) to 2^Q i.e., δ is transition function which is a mapping from Q x (Σ U ε) to 2^Q. Based on the current state there can be a transition to other states with or without any input symbols.
- $q_0 \in Q$ - is the start state.
- $F \subseteq Q$ - is set of accepting or final states.

Note: In an ε-NFA there can be zero, one or more transitions with or without any input symbol. Before proceeding further, let us see "What is ε-CLOSURE?".

❖ **Definition:** The ε-CLOSURE of q denoted by ECLOSE(q) is the set of all states which are reachable from q on ε-transitions only. It is recursively defined as shown below:

- State q is in ECLOSE(q), i.e., ECLOSE(q) = q
- If ECLOSE(q) contains p and if there is a transition from state *p* to state *r* labeled ε, then state *r* is also in ECLOSE(q).

For example, the ε-CLOSURE(q) for each $q \in Q$ is shown below:



The states q_0, q_1 and q_2 are reachable from q_0 without giving any input. So, $ECLOSE(q_0) = \{q_0, q_1, q_2\}$

- The states q_1 and q_2 are reachable from q_1 without giving any input. So,
  $ECLOSE(q_1) = \{q_1, q_2\}$
- The state q_2 is reachable from q_2 without giving any input. So,
  $ECLOSE(q_2) = \{q_2\}$

2.2. Extended Transition Function of ε-NFA to Strings

Now, let us see "What is extended transition function δ* for ε-NFA?".

❖ **Definition:** The extended transition function δ* describes what happens to a state of machine when the input is a string (sequence of symbols). Let $M = (Q, \Sigma, \delta, q_0, F)$ be an ε-NFA. The extended transition function δ^*: Q x $(\Sigma \cup \varepsilon)^*$ to 2^Q is defined recursively as shown below:

- Basis: $\delta^*(q, \varepsilon) = ECLOSE(q)$. This indicates that if the machine is in state q and read no input, then the machine is still in state q.
- Induction: Let w = xa where *a* is the last symbol of *w* and *x* is the remaining string of w. Let *q* is the current state and *x* is the string to be processed and after consuming the string *x*, let the states of the machine is $\{p_1, p_2, p_3, \ldots p_m\}$:

  i.e., $\delta^*(q, x) = \{p_1, p_2, p_3, \ldots p_m\}$

Let the transition from $\{p_1, p_2, p_3, \ldots p_m\}$ on input symbol *a* is

$$\delta(\{p_1, p_2, p_3, \ldots, p_m\}, a) = \delta(p_1, a) \cup \delta(p_2, a) \cup \delta(p_3, a) \ldots \cup \delta(p_m, a)$$
$$= \{r_1, r_2, r_3, \ldots, r_m\}$$

Now, $\delta^*(q, w) = ECLOSE(r_1, r_2, r_3, \ldots, r_m)$

$$= ELCOSE(r_1) \cup ECLOSE(r_2) \cup \ldots ECLOSE(r_m)$$

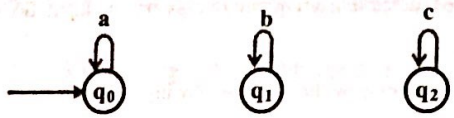
Note: Thus, various properties of extended transition functions for an ε-NFA can be:

- $\delta^*(q, \varepsilon) = ECLOSE(q)$

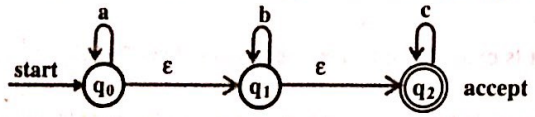
- $\delta^*(q, w) = \delta^*(q, xa) = ECLOSE(\delta(\delta^*(q, x), a))$ where $w = xa$
- $\delta^*(q, w) = \delta^*(q, ax) = ECLOSE(\delta^*(\delta(q, a), x))$ where $w = ax$

Example 1: Now, let us "Obtain an ε-NFA which accepts strings consisting of zero or more a's followed by zero or more b's followed by zero or more c's".

Solution: The ε-NFA that accepts strings of zero or more a's, zero or more b's and zero or more c's can be represented as shown below:



But, it is given that zero or more a's should be followed by zero or more b's followed by zero or more c's. So, the corresponding ε-NFA is shown below:



Thus, an ε-NFA is given by $M = (Q, \Sigma, \delta, q_0, F)$ where

- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{a, b, c\}$
- q_0 is start state
- $F = \{q_2\}$
- δ is shown below using the transition table:

δ	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

2.3. Conversion from ε-NFA to DFA (Algorithm to Convert ε-NFA to DFA)

We have seen in the previous section that an NFA can be converted into DFA using subset construction. On similar lines, it is possible to convert an ε-NFA to DFA. Now, let us see "What is the procedure (or algorithm) to obtain a DFA from an ε-NFA?". The procedure (or algorithm) is shown below:

Procedure (Algorithm): Let $M_E = (Q_E, \Sigma, \delta_E, q_{0E}, F_E)$ be an ε-NFA where Q_E is set of finite states, Σ is set of input alphabets, δ_E is transition from $Q_E \times \{\Sigma \cup \varepsilon\}$ to 2^{Q_E}, q_{0E} is the start state and F_E is the set of final states. The equivalent DFA

$$M_D = (Q_D, \Sigma, \delta_D, q_{0D}, F_D)$$

can be obtained as shown below:

Step 1: If q_0 is the start state of NFA, then $ECLOSE(q_0)$ is the start state of DFA

i.e., $q_{0D} = ECLOSE(q_0)$

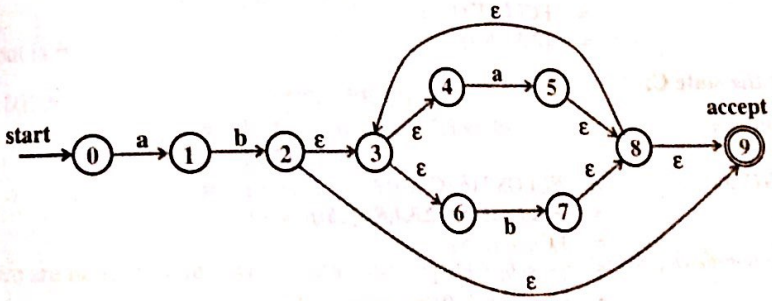
Step 2: Compute the transitions for DFA. Let $\{q_i, q_j, \ldots, q_k\}$ is a state in DFA. Then, the transitions from this state on a i.e., $\delta_D(\{q_i, q_j, \ldots, q_k\}, a)$ is computed as shown below:

- Let $\delta_E(\{q_i, q_j, \ldots, q_k\}, a) = \{p_1, p_2, \ldots, p_m\}$
- Then take $ECLOSE(\{p_1, p_2, \ldots, p_m\})$

Thus, $\delta_D(\{q_i, q_j, \ldots, q_k\}, a) = ECLOSE(\{p_1, p_2, \ldots, p_m\})$.

Step 3: If $\{q_i, q_j, \ldots, q_k\}$ is a state in DFA and if this set contains at least one final state of ε-NFA, then $\{q_i, q_j, \ldots, q_k\}$ is a final state of DFA.

Example: Convert the following NFA to its equivalent DFA.



Note: δ_E represent the transition for ε-NFA.

Step 1: Identify the start state of DFA: Since 0 is the start state of ε-NFA, ECLOSE(0) is the start state of DFA i.e., ECLOSE (0) = {0}.....(A)

Consider the state A:

When input is *a*:

$\delta(A, a) = ECLOSE(\delta_E(A, a)) = ECLOSE(\delta_E(0, a)) = \{1\}$

When input is *b*:

$\delta(A, b) = ECLOSE(\delta_E(A, b)) = ECLOSE(\delta_E(0, b)) = \{\phi\}$

Consider the state B:

When input is *a*:

$\delta(B, a) = ECLOSE(\delta_E(B, a)) = ECLOSE(\delta_E(1, a)) = \phi$

When input is *b*:

$\delta(B, b) = ECLOSE(\delta_E(B, b)) = ECLOSE(\delta_E(1, b)) = ECLOSE(\{2\}) = \{2,3,4,6,9\}$

Consider the state C:

When input is *a*:

$\delta(C, a) = ECLOSE(\delta_E(C, a)) = ECLOSE(\delta_E(\{2,3,4,6,9\}, a)) = ECLOSE\{5\} = \{5, 8, 9, 3, 4, 6\} = \{3, 4, 5, 6, 8, 9\}$(ascending order)

When input is *b*:

$\delta(C, b) = ECLOSE(\delta_E(C, b)) = ECLOSE(\delta_E(\{2, 3, 4, 6, 9\}, b) = ECLOSE(\{7\}) = \{7, 8, 9, 3, 4, 6\} = \{3, 4, 6, 7, 8, 9\}$(ascending order)

Consider the state D:

When input is *a*:

$\delta(D, a) = ECLOSE(\delta(D, a)) = ECLOSE(\delta_E(\{3,4,5,6,8,9\}, a)) = ECLOSE(\{5\}) = \{5, 8, 9, 3, 4, 6\} = \{3, 4, 5, 6, 8, 9\}$(ascending order) (D)

When input is *b*:

$\delta(D, b) = ECLOSE(\delta(D, b)) = ECLOSE(\delta_E(\{3,4,5,6,8,9\}, b)) = ECLOSE(\{7\}) = \{7, 8, 9, 3, 4, 6\} = \{3, 4, 6, 7, 8, 9\}$(ascending order) (E)

Consider the state E:

When input is *a*:

$\delta(E, a) = ECLOSE(\delta(E, a)) = ECLOSE(\delta_E(\{3,4,6,7,8,9\}, a)) = ECLOSE(\{5\}) = \{5, 8, 9, 3, 4, 6\} = \{3, 4, 5, 6, 8, 9\}$(ascending order) (D)

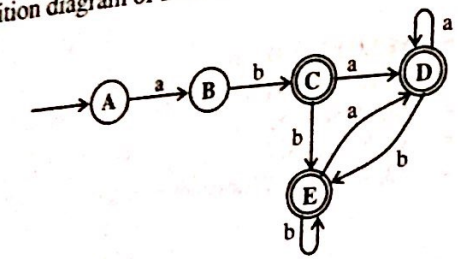
When input is *b*:

$\delta(E, b) = ECLOSE(\delta(E, b)) = ECLOSE(\delta_E(\{3,4,6,7,8,9\}, b)) = ECLOSE(\{7\}) = \{7, 8, 9, 3, 4, 6\} = \{3, 4, 6, 7, 8, 9\}$(ascending order) (E)

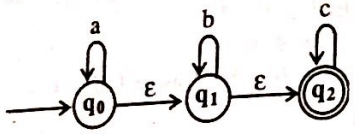
Since there are no new states, we can draw the transition table for the DFA as shown below:

δ	a	b
→ A	B	φ
B	φ	C
*C	D	E
*D	D	E
*E	D	E

Note: The states C,D and E are final states, since 9 which is final state of ε-NFA is present in D and E. The final transition diagram of DFA is shown below:



Example: Convert the following NFA to its equivalent DFA.



Note: Identify the start state of DFA: Since q_0 is the start state of ε-NFA, ECLOSE(q_0) is the start state of DFA i.e., ECLOSE (q_0) = {q_0, q_1, q_2}..... (A)

Consider the state {q_0, q_1, q_2}:

On input *a*:

$$\delta(\{q_0, q_1, q_2\}, a) = \text{ECLOSE}(\{q_0\}) = \{q_0, q_1, q_2\}$$ (A)

On input *b*:

$$\delta(\{q_0, q_1, q_2\}, b) = \text{ECLOSE}(\{q_1\}) = \{q_1, q_2\}$$ (B)

On input *c*:

$$\delta(\{q_0, q_1, q_2\}, c) = \text{ECLOSE}(\{q_2\}) = q_2$$ (C)

Consider the state {q_1, q_2}:

On input *a*:

$$\delta(\{q_1, q_2\}, a) = \phi$$

On input *b*:

$$\delta(\{q_1, q_2\}, b) = \text{ECLOSE}(\{q_1\}) = \{q_1, q_2\}$$ (B)

On input *c*:

$$\delta(\{q_1, q_2\}, c) = \text{ECLOSE}(\{q_2\}) = q_2$$ (C)

Consider the state{q_2}:

On input *a*:

$$\delta(q_0, a) = \phi$$

On input *b*:

$$\delta(q_1, b) = \phi$$

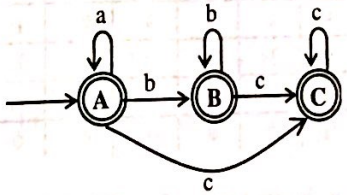
On input *c*:

$$\delta(q_2, c) = \text{ECLOSE}(q_2) = q_2$$ (C)

The transition table along with transition diagram is shown below:

δ	a	b	c
→*A	A	B	C
*B	ϕ	B	C
*C	ϕ	ϕ	C

The DFA can be written as shown below:



Example: Now, let us "Obtain an NFA with ε-transitions (ε-NFA) to accept decimal numbers and obtain $\delta^*(q_0, 4.7)$".

Question to be changed for integers and floating point numbers.

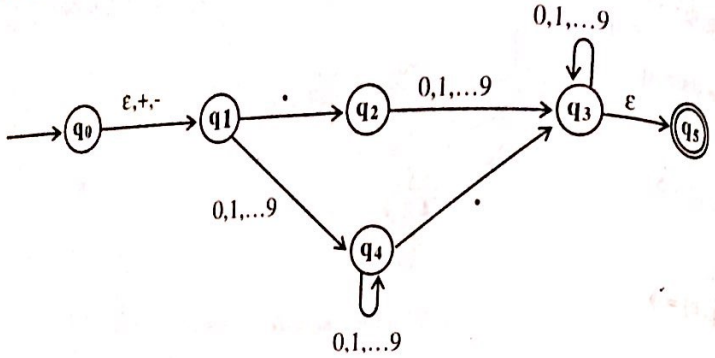
Note: A decimal number has

1. an optional + or – sign followed by '.' followed by string of digits

or

2. an optional + or – sign followed by a string of digits followed by a '.' and in turn followed by string of digits

The ε-NFA is shown in figure below:



Here, the machine M = (Q, Σ, δ, q_0, F) where

Q = {q_0, q_1, q_2, q_3, q_4, q_5}
Σ = { +, –, ., 0, 1, 2, ..., 9}
q_0 is the start state
F = { q_5 } is the final state
δ is shown below using the transition table

δ	ε	+	–	.	0 to 9
→ q_0	q_1	q_1	q_1	ϕ	ϕ
q_1	ϕ	ϕ	ϕ	q_2	q_4
q_2	ϕ	ϕ	ϕ	ϕ	q_3
q_3	q_5	ϕ	ϕ	ϕ	q_3
q_4	ϕ	ϕ	ϕ	q_3	q_4
q_5	ϕ	ϕ	ϕ	ϕ	ϕ

To obtain $\delta^*(q_0, 4.7)$: Since q_0 is the start state of ε-NFA, ECLOSE(q_0) is the start state of DF
i.e., ECLOSE (q_0) = {q_0, q_1}

$\delta^*(q_0, 4) = \text{ECLOSE}(\delta_E(\{q_0, q_1\}, 4))$
$= \text{ECLOSE}(\delta_E(q_0, 4) \cup \delta_E(q_1, 4))$
$= \text{ECLOSE}(\phi \cup q_4)$
$= \{q_4\}$

$\delta^*(\{q_4\}, .) = \text{ECLOSE}(\delta_E(\{q_4\}, .))$
$= \text{ECLOSE}(q_3)$
$= \{q_3, q_5\}$

$\delta^*(\{q_3, q_5\}, 7) = \text{ECLOSE}(\delta_E(q_3, 7) \cup \delta_E(q_5, 7))$
$= \text{ECLOSE}(q_3 \cup \phi)$

$= \text{ECLOSE}(q_3)$
$= \{q_3, q_5\}$

Thus, $\delta^*(q_0, 4.7) = \{q_3, q_5\}$

Example: Now, let us "Obtain the equivalent DFA for the ε-NFA shown above".

Note: Identify the start state of DFA: Since q_0 is the start state of ε-NFA, ECLOSE(q_0) is the start state of DFA i.e., ECLOSE (q_0) = {q_0, q_1}..... (A)

Consider the state [A]:

When input is ± :

$\delta(A, \pm) = \text{ECLOSE}(\delta_E(A, \pm))$
$= \text{ECLOSE}(\delta_E(\{q_0, q_1\}, \pm))$
$= \text{ECLOSE}(q_1) = \{q_1\}$ (B)

When input is .

$\delta(A, .) = \text{ECLOSE}(\delta_E(A, .))$
$= \text{ECLOSE}(\delta_E(\{q_0, q_1\}, .))$
$= \text{ECLOSE}(q_2) = \{q_2\}$ (C)

When input is 0,1,2,...9

$\delta(A, \{0,1,...9\}) = \text{ECLOSE}(\delta_E(A, \{0,1,...9\}))$
$= \text{ECLOSE}(\delta_E(\{q_0, q_1\}, \{0,1,...9\}))$
$= \text{ECLOSE}(q_4) = \{q_4\}$ (D)

Consider the state [B]:

When input is ± :

$\delta(B, \pm) = \text{ECLOSE}(\delta_E(B, \pm))$
$= \text{ECLOSE}(\delta_E(q_1, \pm))$
$= \text{ECLOSE}(\phi) = \phi$

When input is .

$\delta(B, .) = \text{ECLOSE}(\delta_E(B, .))$
$= \text{ECLOSE}(\delta_E(q_1, .))$
$= \text{ECLOSE}(q_2) = \{q_2\}$ (C)

When input is 0,1,2,...9

$\delta(B, \{0,1,...9\}) = ECLOSE(\delta_E(B, \{0,1,...9\}))$
$= ECLOSE(\delta_E(q_1, \{0,1,...9\}))$
$= ECLOSE(q_4) = \{q_4\}$

Consider the state [C]:

When input is ± :

$\delta(C, \pm) = ECLOSE(\delta_E(C, \pm))$
$= ECLOSE(\delta_E(q_2, \pm))$
$= ECLOSE(\phi) = \phi$

When input is .

$\delta(C, .) = ECLOSE(\delta_E(C, .))$
$= ECLOSE(\delta_E(q_2, .))$
$= ECLOSE(\phi) = \phi$

When input is 0,1,2,...9

$\delta(C, \{0,1,...9\}) = ECLOSE(\delta_E(C, \{0,1,...9\}))$
$= ECLOSE(\delta_E(q_2, \{0,1,...9\}))$
$= ECLOSE(q_3)$
$= \{q_3, q_5\}$

Consider the state [D]:

When input is ± :

$\delta(D, \pm) = ECLOSE(\delta_E(D, \pm))$
$= ECLOSE(\delta_E(q_4, \pm))$
$= ECLOSE(\phi) = \phi$

When input is .

$\delta(D, .) = ECLOSE(\delta_E(D, .))$
$= ECLOSE(\delta_E(q_4, .))$
$= ECLOSE(q_3)$
$= \{q_3, q_5\}$

When input is 0,1,2,...9

$\delta(D, \{0,1,...9\}) = ECLOSE(\delta_E(D, \{0,1,...9\}))$
$= ECLOSE(\delta_E(q_4, \{0,1,...9\}))$
$= ECLOSE(q_4)$
$= \{q_4\}(D)$

(E)

Consider the state [E]:

When input is ±:

$\delta(E, \pm) = ECLOSE(\delta_E(E, \pm))$
$= ECLOSE(\delta_E(\{q_3, q_5\}, \pm))$
$= ECLOSE(\phi) = \phi$

When input is .

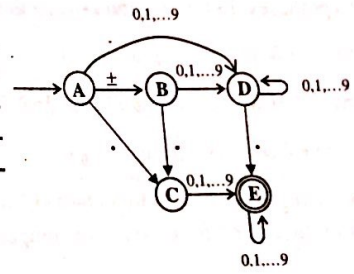
$\delta(E, .) = ECLOSE(\delta_E(E, .))$
$= ECLOSE(\delta_E(\{q_3, q_5\}, .))$
$= ECLOSE(\phi) = \phi$

When input is 0,1,2,...9

$\delta(E, \{0,1,...9\}) = ECLOSE(\delta_E(E, \{0,1,...9\}))$
$= ECLOSE(\delta_E(\{q_3, q_5\}, \{0,1,...9\}))$
$= ECLOSE(q_3)$
$= \{q_3, q_5\}$ (E)

The transition table along with transition diagram is shown below:

δ	+	−	.	0 to 9
→A	B	B	C	D
B	ϕ	ϕ	C	D
C	ϕ	ϕ	ϕ	E
D	ϕ	ϕ	E	D
*E	ϕ	ϕ	ϕ	E



Note: Since E = $\{q_3, q_5\}$ has a final state of NFA, E is the final state in DFA.

Now, let us "Prove that the language L is accepted by DFA M iff L is accepted by an (ε-NFA) N".

Theorem: A language L is accepted by DFA M iff L is accepted by an (ε-NFA) N.

Proof: Let $N = (Q_E, \Sigma, \delta_E, q_0, F_E)$ be an ε-NFA. Applying the modified subset construction (by incorporating ε-CLOSURE), let the DFA obtained is

$M = (Q_D, \Sigma, \delta_D, q_{0D}, F_D)$

Since all the transitions of DFA are obtained from the transitions of ε-NFA, the language accepted by ε-NFA is accepted by DFA. This can be proved using mathematical induction as below:

Basis: if w = ε, we know that

$$\delta_E^*(q_0,\varepsilon) = ECLOSE(q_0)$$

We also know that

$$q_{0D} = ECLOSE(q_0)$$

is correct since that is how the start state of DFA is obtained. So, if w = ε, then

$$\delta_D^*(q_{0D},\varepsilon) = ECLOSE(q_0)$$

By comparing the two relations (1) and (2) we have

$$\delta_E^*(q_0,\varepsilon) = \delta_D^*(q_{0D},\varepsilon)$$

Thus, we have proved that language accepted by ε-NFA is same as language accepted by DFA when w = ε.

Induction hypotheses: Let w = x and assume that the result is true for the string *x* i.e.,

$$\delta_E^*(q_0,x) = \delta_D^*(q_{0D},x) = \{q_i, q_j, \ldots, q_k\}$$

Inductive proof: Let w = xa where *a* is the final symbol of w and we know that:

$$\delta_E^*(q_0,x) = \delta_D^*(q_{0D},x) = \{q_i, q_j, \ldots, q_k\}$$

is true for the string *x* (induction hypotheses). Now, we have to prove that the result is true for w = xa. By definition of δ* for ε-NFA, we compute $\delta_E^*(q_0, w)$ as shown below:

- Let $\delta_E(\{q_i, q_j, \ldots..q_k\}, a) = \{p_1, p_2, \ldots, p_m\}$
- Then $\delta_E^*(q_0, w) = ECLOSE(\{p_1, p_2, \ldots, p_m\})$

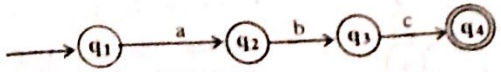
Thus,

$$\delta_E^*(\{q_0, w) = ECLOSE(\{p_1, p_2, \ldots, p_m\})$$

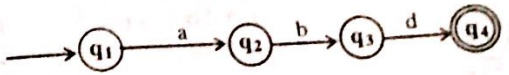
If we examine the construction of DFA from ε-NFA, the above two steps [shown in relations (3) and (4)] are used. Thus, $\delta_D^*(q_0, w) = \delta_E^*(q_0, w)$.

Example: Now, let us "Obtain a NFA's to recognize the strings *abc*, *abd* and aacd assuming Σ = {a, b, c, d}".

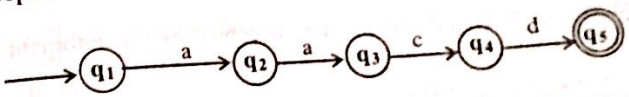
Solution: The machine to accept the string abc is shown below:



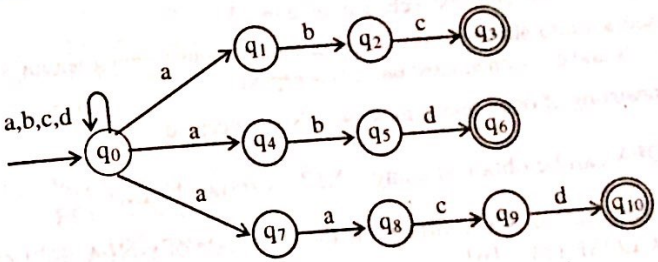
The machine to accept the string abd is shown below:



The machine to accept the string aacd is shown below:



But, all these strings can be preceded by strings of a's, b's, c's and d's and observe that the symbol in each of these machines is *a*. So, the complete NFA to accept the strings abc, abd, aacd can be written as shown below:



Example: Now, let us "Consider the following ε-NFA":

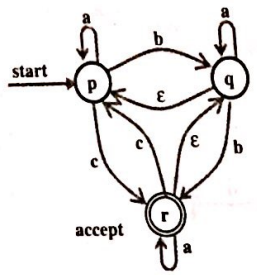
δ	ε	a	b	c
p	ϕ	{p}	{q}	{r}
q	{p}	{q}	{r}	ϕ
*r	{q}	{r}	ϕ	{p}

(i) Compute ε-CLOSURE of each state.

(ii) Give all the strings of length three or less accepted by the automata.

(iii) Convert the automata to DFA.

Solution: The transition diagram for the given ε-NFA is shown below:



1. **Solution:** The ε-CLOSURE of each state can be computed as shown below:

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

2. Solution: All the strings of length three or less accepted by the automata can be obtained as shown below:

- We can move from state to *p* to state *r* using the strings: *c*, bb and *bc*.
- After reaching *r*, on either *b* or *c* we can return to *r* using ε-transitions. So, observe that strings of b's and c's of length 3 or less except *b* are accepted i.e., the following strings are accepted:

  c, bb, bc, cb and cc (length 2 or less)

  bbb, bbc, bcb, bcc, cbb, cbc, ccb, ccc (of length 3)
- When we insert a's into above strings at any of the positions keeping length 3 or less, we find the strings of a's, b's and c's with at most one *a* is accepted.
- The strings consisting of one *c* and at most two *a*'s are accepted.

3. Solution: The DFA can be obtained using ε-NFA as shown below:

Note: Identify the start state of DFA: Since p is the start state of ε-NFA, ECLOSE(p) is the start state of DFA i.e., ECLOSE (p) = {p}..... (A)

Consider the state {p} i.e., A:

On input *a*:

$\delta(\{p\}, a)$ = ECLOSE ({p})
= {p}

On input *b*:

$\delta(\{p\}, b)$ = ECLOSE ({q}).
= {p, q}(B)

On input *c*:

$\delta(\{p\}, c)$ = ECLOSE ({r})
= {p, q, r}

Consider the state {p,q} i.e., B:

(A)

(C)

On input *a*:

$\delta(\{p,q\}, a)$ = ECLOSE (δ(p,a) U δ(q, a))
= ECLOSE (p, q)
= ECLOSE (p,q)
= {p,q}(B)

On input *b*:

$\delta(\{p,q\}, b)$ = ECLOSE (δ(p,b) U δ(q, b))
= ECLOSE (q, r)
= ECLOSE (p) U ECLOSE (r)
= {p,q,r}(C)

On input *c*:

$\delta(\{p,q\}, c)$ = ECLOSE (δ(p,c) U δ(q, c))
= ECLOSE (r, ϕ)
= ECLOSE (r)
= {p,q,r}(C)

Consider the state {p,q,r} i.e., C:

On input *a*:

$\delta(\{p,q,r\}, a)$ = ECLOSE (δ(p,a) U δ(q, a) U δ(r,a))
= ECLOSE (p, q, r)
= ECLOSE (p,q,r)
= {p,q,r}(C)

On input *b*:

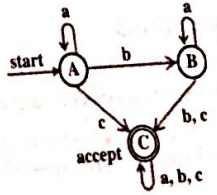
$\delta(\{p,q,r\}, b)$ = ECLOSE (δ(p,b) U δ(q, b) U δ(r,b))
= ECLOSE (q, r, ϕ)
= ECLOSE (p,q,r)
= {p, q, r}(C)

On input *c*:

$\delta(\{p,q,r\}, c)$ = ECLOSE (δ(p,c) U δ(q, c) U δ(r,c))
= ECLOSE (r, ϕ , ϕ)
= ECLOSE (r)
= {p, q, r}(C)

The transition table along with transition diagram is shown below:

δ	a	b	c
→ A	A	B	C
B	B	C	C
*C	C	C	C



2.4. Difference Between DFA, NFA and ε-NFA

Now, let us see "What are the difference between DFA, NFA and ε-NFA?" Strictly speaking, difference between DFA and NFA lies only in the definition of δ. Using this difference some more points can be derived and can be written as shown:

DFA	NFA	ε-NFA
The DFA is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$ where • Q is set of finite states • Σ is set of input alphabets • $\delta : Q \times \Sigma$ to Q • q_0 is the start state • $F \subseteq Q$ is set of final states	An NFA is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$ where • Q is set of finite states • Σ is set of input alphabets • $\delta : Q \times \Sigma$ to 2^Q • q_0 is the start state • $F \subseteq Q$ is set of final states	An ε-NFA is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$ where • Q is set of finite states • Σ is set of input alphabets • $\delta : Q \times (\Sigma \cup \varepsilon)$ to 2^Q • q_0 is the start state • $F \subseteq Q$ is set of final states
• There can be zero or one transition from a state on an input symbol	• There can be zero, one or more transitions from a state on an input symbol	• There can be zero, one or more transitions from a state with or without giving any input
• More number of transitions	• Less number of transitions	• Relatively more transitions when compared with NFA
• Difficult to construct	• Easy to construct	• Easy to construct using regular expressions
• Less powerful since at any point of time it will be in only one state	• More powerful than DFA since at any point of time it will be in more than one state	• More powerful than NFA since at any point of time it will be in more than one state with or without giving any input