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 $2^{\wedge}$.

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 $\mathbf{2^{\wedge}}$.

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

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 $2^{\wedge}$.

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 $\bar{A}$ and is defined as a set containing everything that is not in A. Formally, complement of A is defined as follows:

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

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

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

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

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

$\Sigma = \{a, b, \ldots, z, A, B, \ldots Z, 0, 1, \ldots 9, \#, (,), \{, \}, <, >, !, [,], \ldots\ldots\}$

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

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

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: $\varepsilon w = w\varepsilon = w$

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"$\ldots\}$

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 the alphabets of a particular language. In other words, a language is subset of Σ^* which is denoted by $L \subseteq \Sigma^*$. 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 represented 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.

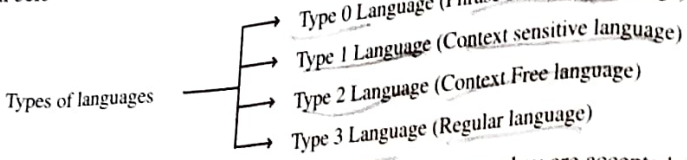
Now, let us see "What is a sentence? Give example".

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

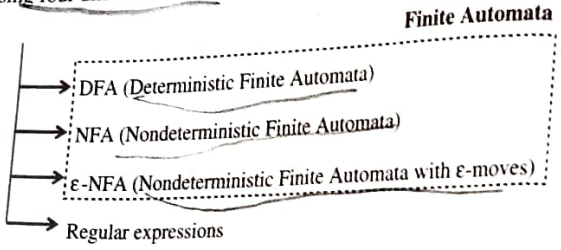
{ε, 01, 10, 0011,1010,0101,0011,.....}

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

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



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

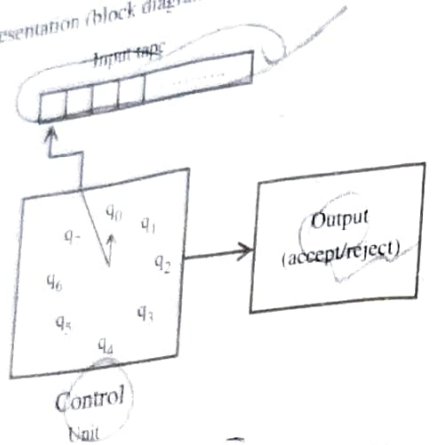
- Finite automata
- Linear bounded automata
- Push down automata
- Turing machine

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 a digital computer which has three components as shown below:

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

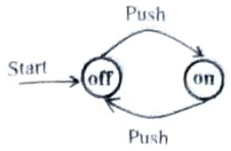


- *Input tape*: The input tape is divided into cells each of which can hold one symbol. The string to be processed is stored in these cells.
- *Control Unit*: The machine has some states one of which is the start state designated as q_0 and at least one final state. Apart from these, it has some finite 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.

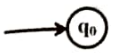
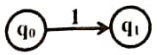
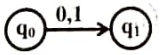
For example, consider an electric switch which has only two states "OFF" and "ON". To start with, the switch will be in OFF state. When we push the button, it goes to ON state. If we push once again it goes to OFF state. This can be represented as shown below:



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?"

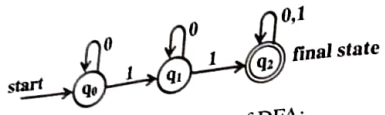
Symbol	Meaning
	A circle is used to represent a state. Here, q_0 is a state of the machine.
	A circle with an arrow which is not originating from any node represents the start state of machine.
	Two circles are used to represent a final state. Here, q_0 is the final state.
	An arrow with label 1 goes 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 w_i 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 of 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, q_1 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 DFA accepts empty language.

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

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

- **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

  $\delta(q_i, a) = q_j$

For example, in the above diagram, the various transitions shown are represented as shown 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)?".

4 Definition The Deterministic Finite Automaton is also [illegible] consisting five components

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

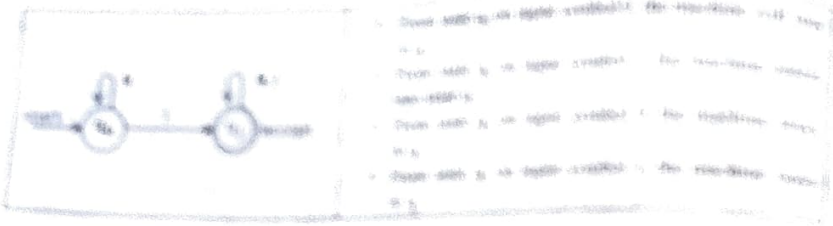
where

- Q = the finite set of all states in the machine
- Σ = [illegible]
- δ = the [illegible]
- q_0 = [illegible]
- $F \subseteq Q$ = [illegible]

Then some DFA accepts how the following here

- **Observation** : The [illegible] the machine [illegible] from the state q_0 [illegible] a sequence exactly x which can be reached when we [illegible] the machine M the machine is [illegible]
- **Proof** : The [illegible] δ [illegible] q_0, x [illegible] and [illegible]
- **Automaton** : [illegible] which can accept the string x [illegible] the string x [illegible]

Note The DFA we have [illegible] the machine from [illegible] the following DFA and [illegible]



- From [illegible] the machine will [illegible]
- From [illegible] the machine [illegible]
- From [illegible] the machine [illegible]
- From [illegible] the machine [illegible]

Observe from the above diagram that [illegible] the machine [illegible] a string [illegible] transitions there [illegible]

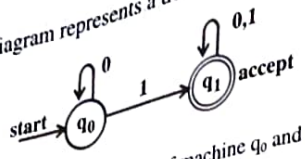
- From [illegible] the machine [illegible]
- For [illegible] the [illegible] of the input symbol [illegible] from [illegible]

Note In [illegible] there can be [illegible] from a state to an input symbol and it [illegible]

1.3 Lengthy Notations for DFA's

[illegible] the machine [illegible] [illegible] [illegible]

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 the arc with labels 0, 1 both from q_1 to q_1.

1.5.2. Transition Table

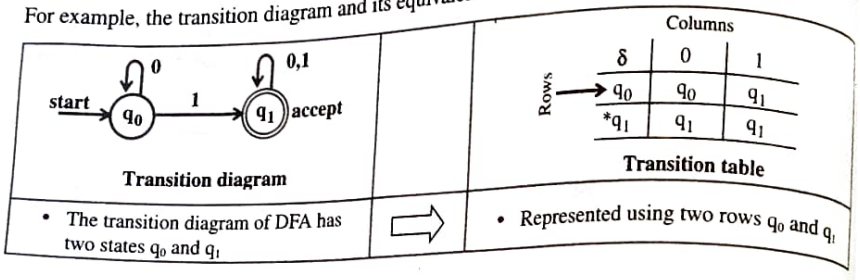
Definition: Now, let us see "What is a transition table?". The transition table for DFA

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

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.

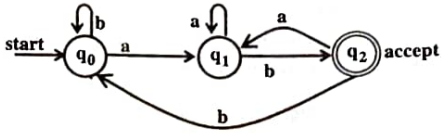
For example, the transition diagram and its equivalent transition table are shown below:



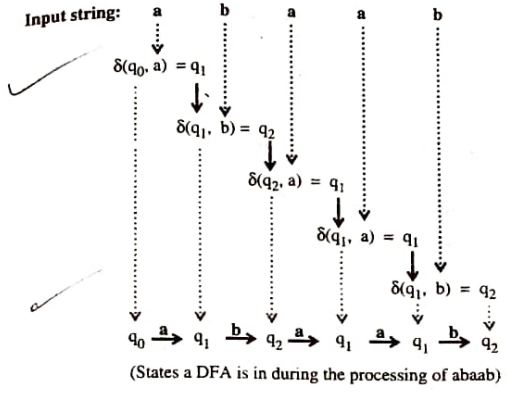
• There are two input symbols 0 and 1	⇨	• The two input symbols 0 and 1 correspond to two columns
• Start state is represented using an arrow mark not originating from any state and labeled start	⇨	• The start state is identified by putting an arrow with direction towards right
• The final states are represented by two circles.	⇨	• The final states are represented by putting stars (*'s) by the side of states
• 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 transition is represented by writing p in row q and column a

1.5.3. How a DFA Processes Strings

Now, let us see "What are the moves made by the following DFA while processing the string abaab and abb?".



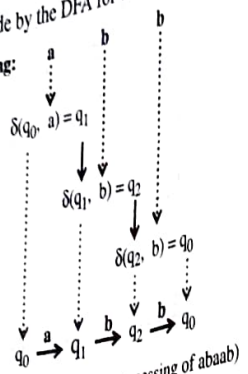
Solution: The moves made by the DFA for the input string abaab is shown below:



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

Note: At the end of the string abaab, the DFA will be in state q_2 which is the final state. So, the string abaab is accepted by 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 is 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 input 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 on 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, the 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 string *w* is accepted by DFA. After the end of the input, if the DFA is not in a final state, then the string *w* is rejected by the DFA.

1.5.4. Extended Transition Function of DFA to Strings

Note: The transition $\delta(q, a) = p$ accepts two parameters namely state *q* and input symbol *a* as the parameters and returns a state *p* which is the next state of the machine. But, if there is a change of state from state *q* to state *p* 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 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 $M = (Q, \Sigma, \delta, q_0, F)$ be a FA. The extended transition function δ^*: **$Q \times \Sigma^*$ to 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 w is the string to be processed and after consuming the string w, let the state of the machine is p. Then

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

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

- $\delta^*(q, \varepsilon) = q$
- $\delta^*(q, w) = \delta^*(q, xa) = \delta(\delta^*(q, x), a)$ where w = xa
- $\delta^*(q, w) = \delta^*(q, ax) = \delta^*(\delta(q, a), x)$ where w = ax (modification of above)

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

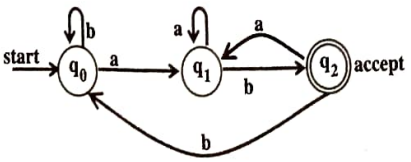
$$\delta^*(q, w) = p$$

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: $\delta^*(q_0, \varepsilon) = q_0$ (1)
- For prefix a: $\delta^*(q_0, a) = \delta(\delta^*(q_0, \varepsilon), a)$ Substituting (1)
  $= \delta(q_0, a)$
  $= q_1$ (2)
- For prefix ab: $\delta^*(q_0, ab) = \delta(\delta^*(q_0, a), b)$ Substituting (2)
  $= \delta(q_1, b)$
  $= q_2$ (3)
- For prefix aba: $\delta^*(q_0, aba) = \delta(\delta^*(q_0, ab), a)$ Substituting (3)
  $= \delta(q_2, a)$
  $= q_1$ (4)
- For prefix abaa: $\delta^*(q_0, abaa) = \delta(\delta^*(q_0, aba), a)$ Substituting (4)
  $= \delta(q_1, a)$
  $= q_1$ (5)
- 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 of 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 abaab is accepted by DFA. Now, let us see "When a language is accepted by DFA?".

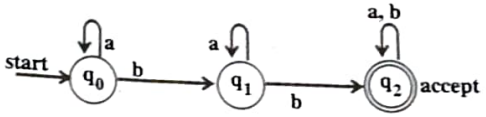
1.5.5. Language Accepted by a DFA

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

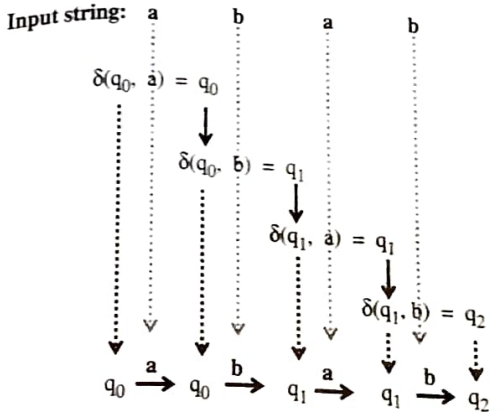
❖ **Definition**: Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA. A string w is accepted by the machine M, if it takes the machine from initial state q_0 to final state i.e., $\delta^*(q_0, w)$ is in F. Thus, the language accepted by DFA represented as L(M) can be formally written as:

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

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) abab

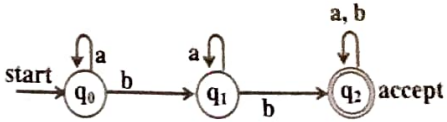
Now, let us see "What language is rejected by DFA?".

❖ **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.

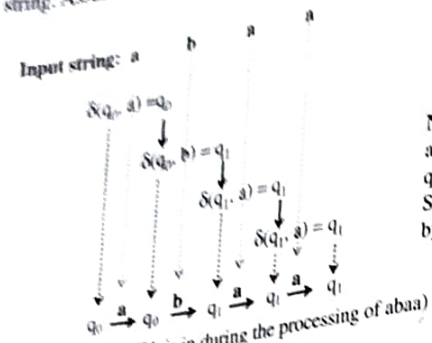
Thus, the non-acceptance of the string w by a FA or DFA can be defined as:

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

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:



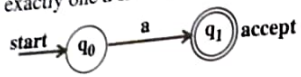
(States a DFA is in during the processing of abaa)

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.

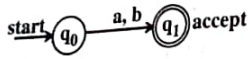
Example 1: DFA to accept empty language $L = \{ \phi \}$ is shown below:

Example 2: DFA to accept an empty string $L = \{ \varepsilon \}$ is shown below:

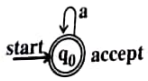
Example 3: DFA to accept exactly one *a* is shown below:



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.

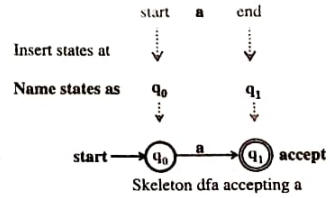
Example 1: Now, let us "Draw a DFA to accept string of a's having at least one *a*".

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 $\Sigma = \{a\}$.

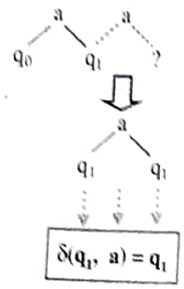
Step 3: Construct a skeleton DFA: The DFA for the string *a* identified in step 1 can be constructed as shown below:



Skeleton dfa accepting a

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

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



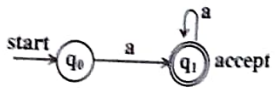
Note: The sequence *aa* should be accepted since it is having at least one *a*. (go to final state q_1)

Step 5: Construct the DFA. The DFA can be obtained by skeleton DFA and transitions obtained 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:



δ	a
$\rightarrow q_0$	q_1
$*q_1$	q_1

Transition diagram **Transition Table**

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

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

or

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

or

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

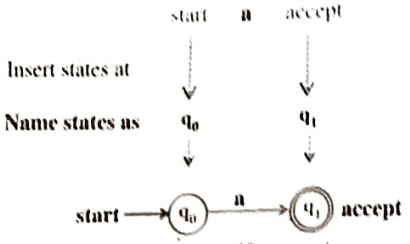
Example 2: Now, let us "Draw a DFA to accept strings of a's and b's having at least one *a*".

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:

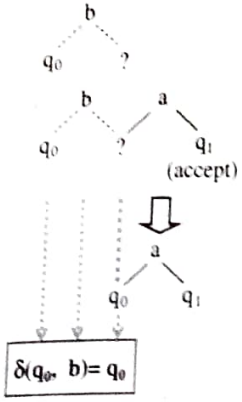


Skeleton dfa accepting a

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

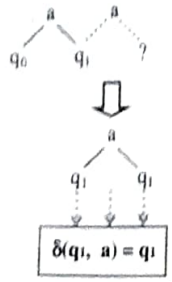


Note: The sequence *b* should not be accepted since it is not having at least one *a*.

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

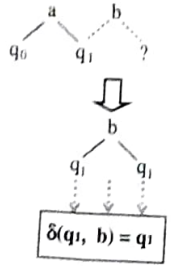
But, before the suffix *a*, the machine is in state q_0 (see skeleton DFA)

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): $\delta(q1, b) = ?$ Move from q0 to q1 after reading symbol a (see skeleton DFA) and then think of transition from q1 after reading symbol b.



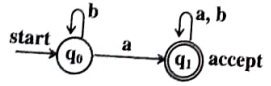
Note: The sequence *ab* should be accepted since it is having at least one *a*. So, go to final state q_1.

Step 5: Construction of DFA. The DFA can be obtained by skeleton DFA and transitions obtained 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.

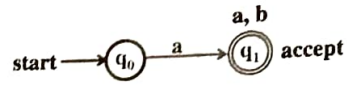
Example 3: Now, let us "Draw a DFA to accept strings of a's and b's having exactly one *a*".

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

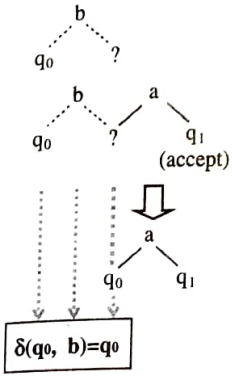
Step 3: Construct the skeleton DFA: The DFA for the string *a* identified in step 1 can constructed as shown below:



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

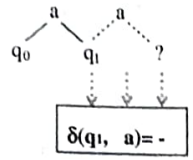


Note: The sequence *b* should not be accepted since it is not having one *a*.

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)

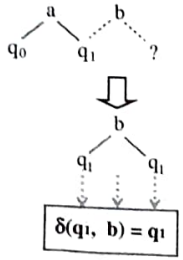
Step(ii): $\delta(q_1, a) = ?$ Move from q_0 to q_1 after reading symbol *a* (see skeleton DFA) and then think of transition from q_1 after reading symbol *a*.



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 then think of transition from q_1 after reading symbol *b*.

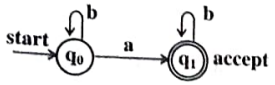


Note: The sequence *ab* should be accepted since it is having one *a*. So, go to final state q_1.

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:

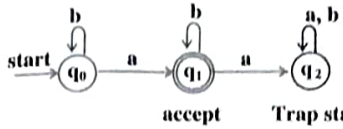


$\mathbf{L = \{w : n_a(w) = 1, w \in \{a, b\}^*\}}$

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:

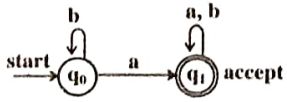


δ	a	b
→ q_0	q_1	q_0
*q_1	-	q_1

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

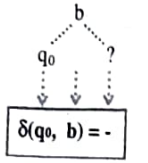
Step 3: Construct the skeleton DFA: The DFA for the string *ab* identified in step 1 can constructed as shown below:

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)

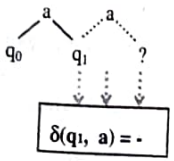
Step (i) : $\delta(q_0, b)$=?



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.

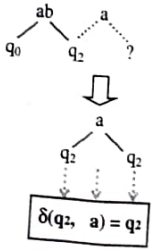
Step (ii): $\delta(q_1, a) = ?$ Move from q_0 to q_1 after reading symbol *a* (see skeleton DFA) and then think of transition from q_1 after reading symbol *a*.



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.

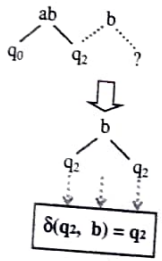
Note: Thesymbol '-' indicates that the transition is not defined.

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



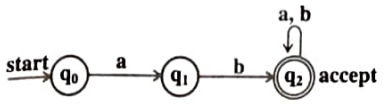
Note: The sequence *abb* should be accepted since it is having substring *ab*. So, go to final state q_2.

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:

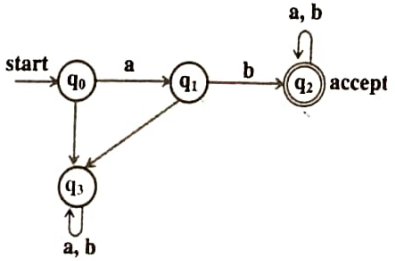


Transition diagram

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.

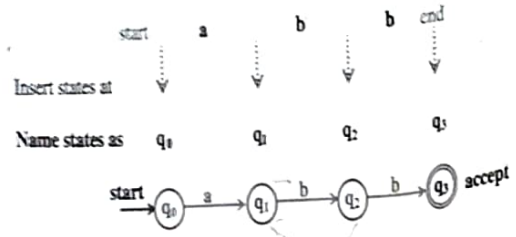


Example 5: Now, let us "Draw a DFA to accept string of a's and b's ending with the string abb".

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

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

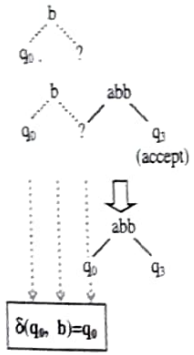
Step 3: construct a skeleton DFA: The DFA for the string *abb* identified in step 1 can constructed as shown below:



Step 4: Identify other transitions not defined in step 3: In this case, we need to compute the 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) : $\delta(q_0, b)$=?

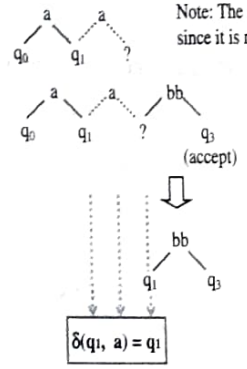


Note: The sequence *b* should not be accepted since it is not ending with *abb*.

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

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 then think of transition from q_1 after reading symbol *a*.

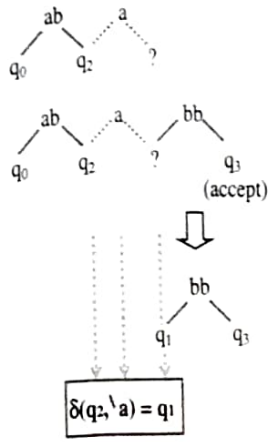


Note: The sequence *aa* should not be accepted since it is not ending with *abb*.

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*.

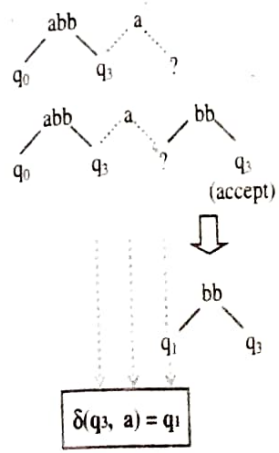


Note: The sequence *aba* should not be accepted since it is not ending with *abb*.

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*.

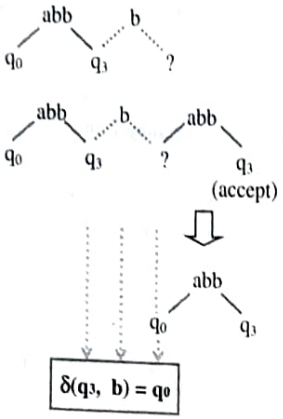


Note: The sequence *abba* should not be accepted since it is not ending with *abb*.

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 string *abbbabb* has to be accepted since it ends with *abb*. So, go to final state q_3

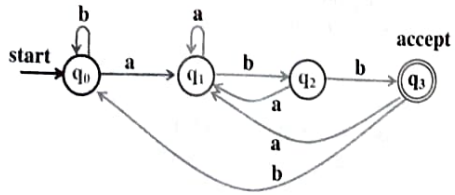
But, before the suffix *abb*, the machine is in state q_0 (see skeleton DFA)

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

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

where

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



Transition diagram

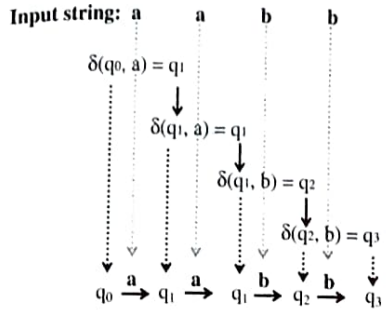
δ	a	b
→ q_0	q_1	q_0
q_1	q_1	q_2
q_2	q_1	q_3
*q_3	q_1	q_0

Transition table

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

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

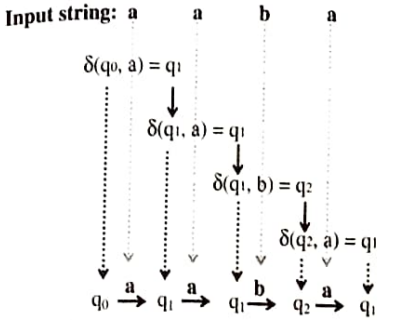
Now, let us "Show the sequence of moves made by the DFA for the string aabb". The moves made by the DFA for the string **aabb** is shown below:



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:

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

where

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

- q_0 = start state
- $F = \{q_3, q_4, q_5\}$
- δ is shown using the transition diagram/table as shown below:

Transition diagram

δ	a	b
$\rightarrow q_0$	q_1	q_6
q_1	q_1	q_2
q_2	q_3	q_6
$*q_3$	q_3	q_3
$*q_4$	q_4	q_4
$*q_5$	q_5	q_5
q_6	q_6	q_6

Transition table

Example 2: Now, let us "Draw a DFA to accept string of a's and b's having a substring *aab*"

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

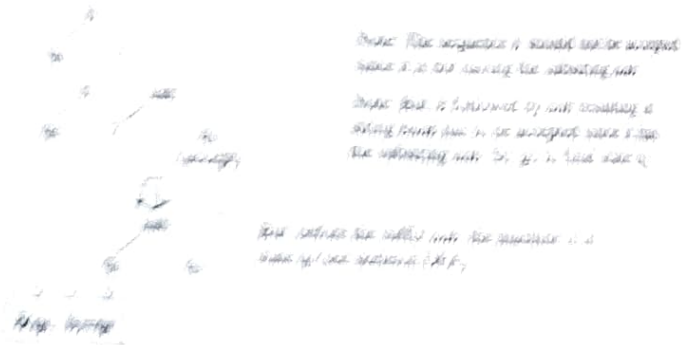
Step 2: Identify the alphabets: In this case $\Sigma = \{a, b\}$.

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

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

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



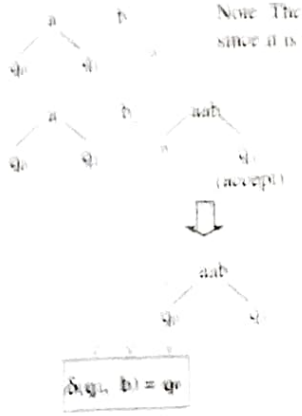
Note: The sequence *b* should not be accepted since it is not having the substring *aab*

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)

$\delta(q_0, b) = q_0$

Step (ii): $\delta(q_1, b) = ?$ Move from q_1 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_1 (see skeleton DFA)

$\delta(q_1, b) = q_2$

Step (iii): $\delta(q_2, a) = ?$ Move from q_2 to q_3 after reading sequence *aa* (see skeleton DFA) and then think of transition from q_3 after reading symbol *a*

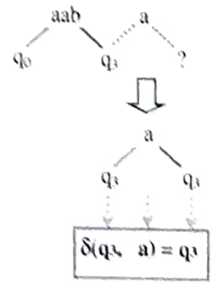
Note: The sequence *aaa* should not be accepted since it is not having the substring *aab*

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 *a*, the machine is in state q_2 (see skeleton DFA)

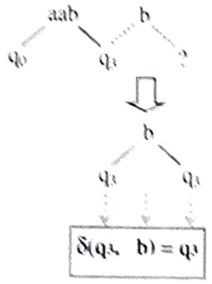
$\delta(q_2, a) = q_3$

Step (iv): $\delta(q_3, a) = ?$ Move from q_3 to q_3 after reading sequence *aaa* (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): $\delta(q_1, b) = ?$ Move from q_0 to q_1 after reading sequence *aab* (see skeleton DFA) and then think of transition from q_3 after reading symbol *b*.



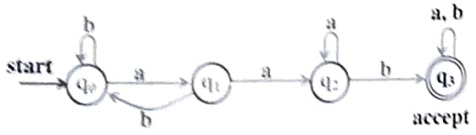
Note: The sequence *aabb* should be accepted since it is having the substring *aab*. So, go to final state q_3

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



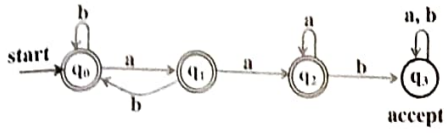
Transition diagram

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:



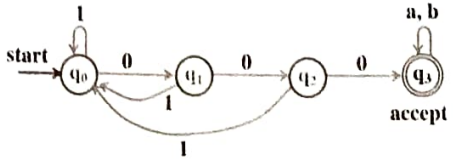
Transition diagram

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

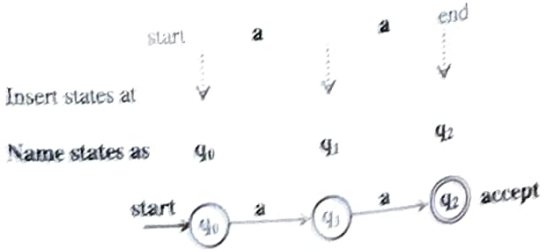
$$L = \{awa \mid w \in (a+b)^n \text{ where } n \geq 0\}"$$

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

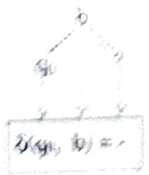
Step 3: Construct a skeleton DFA: The DFA for the string *aa* identified in step 1 can constructed as shown below:



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

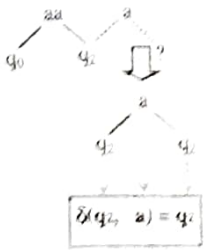
Step (ii): $\delta(q_1, b) = ?$ Move from q_1 to q_1 after reading symbol *a* (see skeleton DFA) and then think of transition from q_1 after reading symbol *b*.

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

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

Bbaa, before the shift *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_f.

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: Just *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_f.

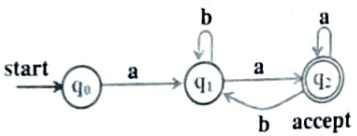
Bba, before the shift *a* the machine is in state q_f (see skeleton DFA).

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

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

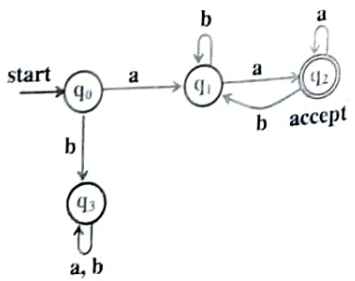
where

- $Q = \{q_0, q_1, q_2, q_f\}$
- $\Sigma = \{a, b\}$
- q_0 – start state
- $F = \{q_f\}$
- δ is shown using the transition diagram 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 to the trap state thus rejecting the string totally. So, the above DFA can also be written as shown below.



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

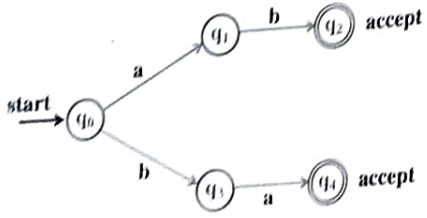
Solution: The given language can also be written as shown below:

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

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 can constructed as shown below:

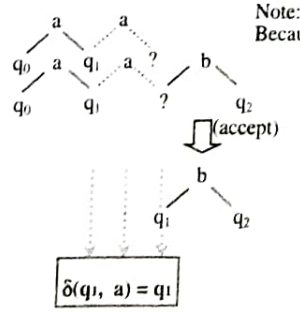


Step 4: Identify other transitions not defined in step 3: In this case, we need to compute the following:

$\delta(q_1, a) = ?$	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*.

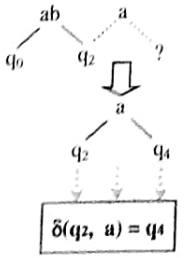


Note: The sequence *aa* should not be accepted. Because, it is not ending with *ab* or *ba*.

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

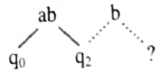
But, before the suffix *b*, the machine is in state q_1 (see skeleton DFA)

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*.

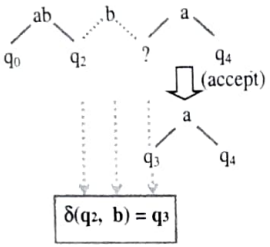


Note: The sequence *aba* should be accepted. Because, it is ending with *ba*. Since, the string ends with *ba*, the machine should go to q_4.

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*.



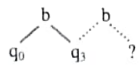
Note: The sequence *abb* should not be accepted. Because, it is not ending with *ab* or *ba*.



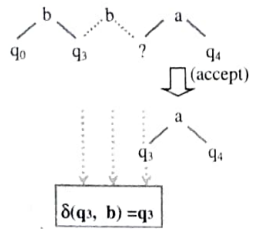
Note: But, *abb* followed by *a* resulting in string *abba* has to be accepted since it ends with *ba*. Because, it ends with *ba* let us go to final state q_4

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 then think of transition from q_3 after reading symbol *b*.



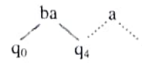
Note: The sequence *bb* should not be accepted. Because, it is not ending with *ab* or *ba*.



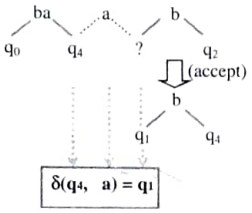
Note: But, *bb* followed by *a* resulting in string *bba* has to be accepted since it ends with *ba*. Because, it ends with *ba* let us go to final state q_4

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 think of transition from q_4 after reading symbol *a*.



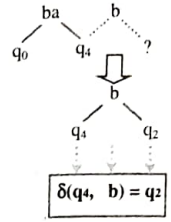
Note: The sequence *baa* should not be accepted. Because, it is not ending with *ab* or *ba*.



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*.



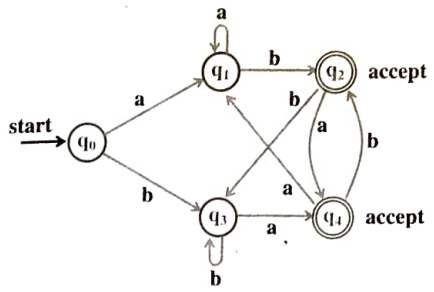
Note: The sequence *bab* should be accepted. Because, it is ending with *ab*. Since, the string ends with *ab*, the machine should go to q_2.

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

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

where

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



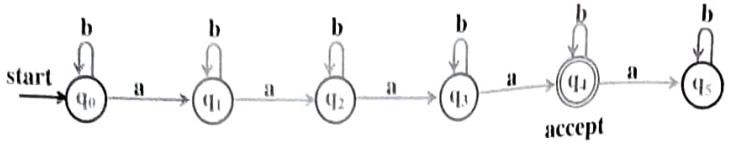
Transition diagram

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

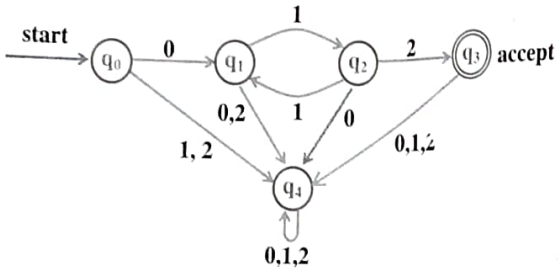
Solution: The DFA can be obtained using similar methods as explained earlier. The final DFA is shown below:



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

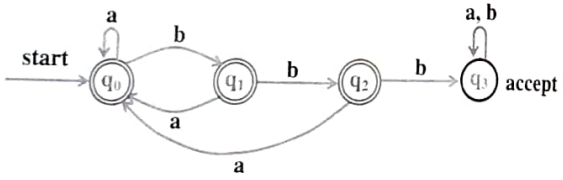
Example 13: Now, let us "Obtain a DFA to accept strings of 0's, 1's and 2's beginning with a '0' followed by odd number of 1's and ending with a '2'".

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



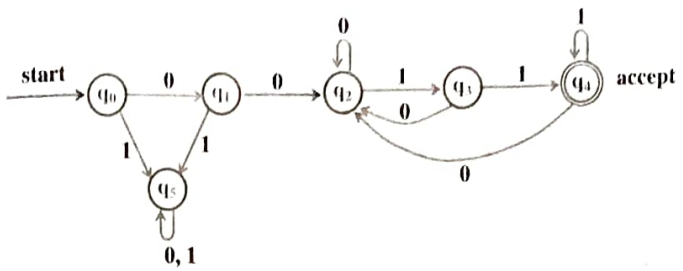
Example 14: Now, let us "Obtain a DFA to accept strings of a's and b's with at most two consecutive b's".

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:



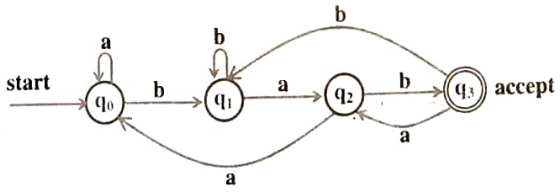
Note: In set notation, the language accepted DFA can be represented as

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

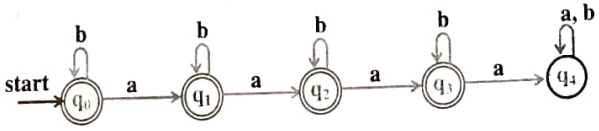
which is the language formed by words that begin with at least two 0's and ending with at least two 1's.

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:

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