DFA and NFA

1.1 DETERMINISTIC FINITE AUTOMATA (DFA)

1.1.1 Automata—What is it?

An automaton is an abstract model of a digital computer. An automaton has a mechanism to read input, which is a string over a given alphabet. This input is actually written on an “input file”, which can be read by the automaton but cannot change it.

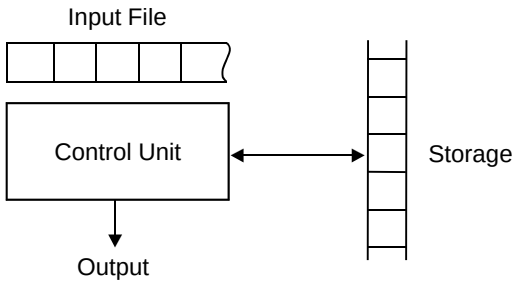


Fig. Automaton

Input file is divided into cells, each of which can hold one symbol. The automaton has a temporary “storage” device, which has unlimited number of cells, the contents of which can be altered by the automaton. Automaton has a control unit, which is said to be in one of a finite number of “internal states”. The automaton can change state in a defined way.

1.1.2 Types of Automaton

(a) Deterministic Automata
(b) Non-deterministic Automata

A deterministic automata is one in which each move (transition from one state to another) is unequally determined by the current configuration.

If the internal state, input and contents of the storage are known, it is possible to predict the future behaviour of the automaton. This is said to be deterministic automata otherwise it is nondeterminist automata.

An automaton whose output response is "yes" or "No" is called an "Acceptor".

1.1.3 Definition of Deterministic Finite Automaton

A Deterministic Finite Automator (DFA) is a 5-tuple

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

where

Q	=	Finite state of "internal states"
Σ	=	Finite set of symbols called "Input alphabet"
$\delta : Q \times \Sigma \to Q$	=	Transition Function
$q_0 \in Q$	=	Initial state
$F \subseteq Q$	=	Set of Final states

The input mechanism can move only from left to right and reads exactly one symbol on each step.

The transition from one internal state to another are governed by the transition function δ.

If $\delta(q_0, a) = q_1$, then if the DFA is in state q_0 and the current input symbol is a, the DFA will go into state q_1.

Example 1.1.1: Design a DFA, M which accepts the language $L(M) = \{w \in (a, b)^* : w$ does not contain three consecutive b's$\}$.

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

where

Q = $\{q_0, q_1, q_2, q_3\}$
Σ = $\{a, b\}$
q_0 is the initial state
F = $\{q_0, q_1, q_2,\}$ are initial states
and δ is defined as follows:

Initial state q	Symbol σ	Final state $\delta(q, \sigma)$
q_0	a	q_0
q_0	b	q_1
q_1	a	q_0
q_1	b	q_2
q_2	a	q_0
q_2	b	q_3
q_3	a	q_3
q_3	b	q_3

Solution

M does not accept specified language, as long as three consecutive *b*'s have not been read.

It should be noted that

(i) *M* is in state q_i (where $i = 0,1,$ or 2) immediately after reading a run of *i* consecutive *b*'s that either began the input string or was preceded by an '*a*'.

(ii) If an '*a*' is read and *M* is in state, q_0, q_1, or *M* returns to its initial state q_0.

q_0, q_1 and q_2 are "Final states" (as given in the problem). Therefore any input string not containing three consecutive *b*'s will be accepted.

In case we get three consecutive *b*'s then the q_3 state is reached (which is not final state), hence *M* will remain in this state, irrespective of any other symbol in the rest of the string. This state q_3 is said to be "dead state" or *M* is said to be "trapped" at q_3.

The DFA schematic is shown below based on the discussion above.

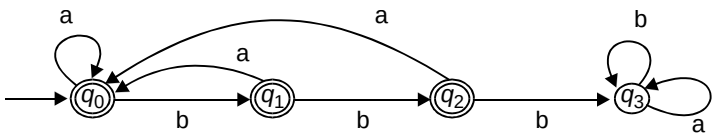


Fig. Finite Automaton with four states

Example 1.1.2: Determine the DFA schematic for $M = (Q, \Sigma, \delta, q, F)$ where $Q = \{q_1, q_2, q_3\}$, $\Sigma = \{0,1\}$, q_1 is the start state, $F = \{q_2\}$ and δ is given by the table below.

Initial state q	Symbol σ	Final state $\delta(q,\sigma)$
q_1	0	q_1
q_1	1	q_2
q_2	0	q_3
q_2	1	q_2
q_3	0	q_2
q_3	1	q_2

Also determine a Language *L* recognized by the DFA.

Solution

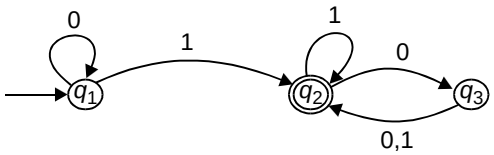


Fig. Finite Automaton having three states.

From the given table for δ, the DFA is drawn, where q_2 is the only final state.

(It is to be noted that a DFA can "accept" a string and it can "recognize" a language. Catch here is that "accept" is used for strings and "recognize" for that of a language).

It could be seen that the DFA accepts strings that has at least one 1 and an even number of 0s following the last 1.

Hence the language L is given by

$$L = \{w \mid w \text{ contains at least one 1 and an even number of 0s follow the last 1}\}$$

where $L = L(M)$ and M recognized the RHS of the equation above.

Example 1.1.3: Sketch the DFA given

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

and δ is given by

$$\delta(q_1, 0) = q_1 \text{ and } \delta(q_2, 0) = q_1$$
$$\delta(q_1, 1) = q_2 \qquad \delta(q_2, 1) = q_2$$

Determine a Language $L(M)$, that the DFA recognizes.

Solution

From the given data, it is easy to predict the schematic of DFA as follows.

Internal states = q_1, q_2.

Symbols = 0, 1.

Transition function = δ (as defined above in the given problem)

q_1 = Initial state

q_2 = Final state.

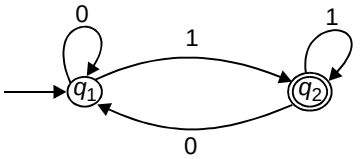


Fig. State diagram of DFA

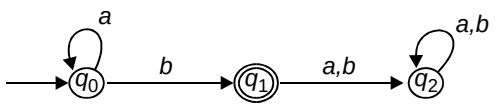
If a string ends in a 0, it is "rejected" and "accepted" only if the string ends in a 1. Therefore the language

$$L(M) = \{w \mid w \text{ ends in a } 1\}.$$

Example 1.1.4: Design a DFA, the language recognized by the Automaton being

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

Solution



For the given language $L = \{a^n b : n \geq 0\}$, the strings could be b, ab, a^2b, a^3b, ….

Therefore the DFA accepts all strings consisting of an arbitrary number of a's, followed by a single b. All other input strings are rejected.

Example 1.1.5: Obtain the state table diagram and state transistion diagram (DFA Schematic) of the finite state Automaton $M = (Q, \Sigma, \delta, q_0, F)$, where $Q = \{q_0, q_1, q_2, q_3\}$, $\Sigma = \{a, b\}$, q_0 is the initial state, F is the final state with the transistion defined by

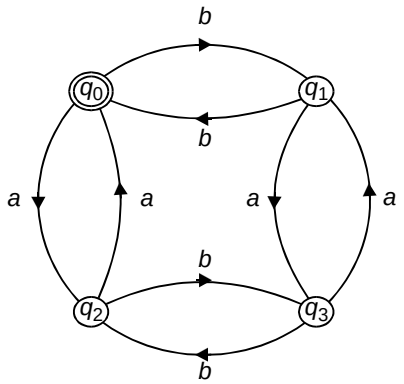
$$\delta(q_0, a) = q_2 \quad \delta(q_3, a) = q_1 \quad \delta(q_2, b) = q_3$$
$$\delta(q_1, a) = q_3 \quad \delta(q_0, b) = q_1 \quad \delta(q_3, b) = q_2$$
$$\delta(q_2, a) = q_0 \quad \delta(q_1, b) = q_0$$

Solution

The State Table diagram is as shown below

δ	a	b
q_0	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

With the given definitions, the State Transition diagram/DFA Schematic is shown on next page.



Example 1.1.6: Obtain the DFA that accepts/recognizes the language

$$L(M) = \{w \mid w \in \{a, b, c\}^* \text{ and } w \text{ contains the pattern } abac\}$$

(*Note:* This is an application of DFA's involving searching a text for a specified pattern)

Solution

Let us begin by "hard coding" the pattern into the machines states as shown in fig. (a) below.

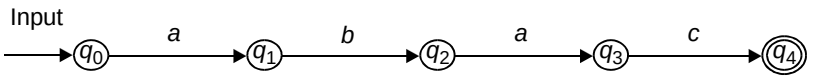


Fig. (a)

As the pattern '*abac*' has length four, there are four states required in addition to one intial state q_0, to remember the pattern. q_4 is the only accepting state required and this state q_4 can be reached only after reading '*abac*'.

The complete DFA is as shown below in Fig. (b).

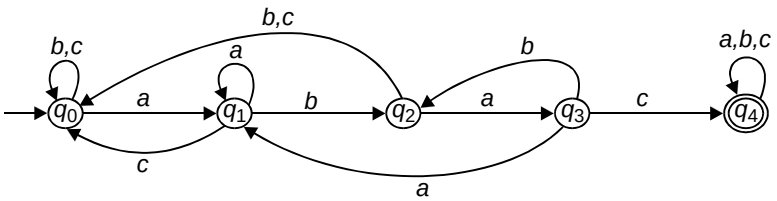
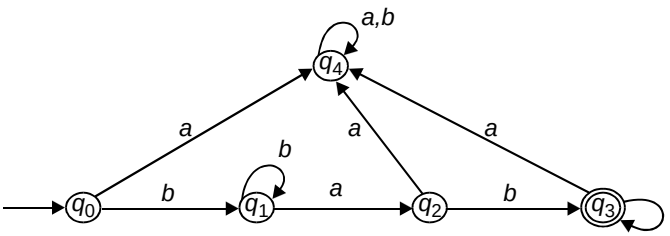


Fig. (b)

Example 1.1.7: Given $\Sigma = \{a, b\}$, construct a DFA that shall recognize the language $L = \{b^m ab^n : m, n > 0\}$.

Solution

The given language $L = \{b^m ab^n : m, n > 0\}$ has all words with exactly one '*a*' which is neither the first nor last letter of the word i.e., there is one or more *b*'s before or after '*a*'.



DFA is drawn above for the automaton *M*,

where $M = (Q, \Sigma, \delta, q_0, F)$ with
$Q = \{q_0, q_1, q_2, q_3, q_4\}$
$\Sigma = \{a, b\}$; q_0 = Initial state,
$F = \{q_3\}$ = Final state.
and δ is defined as per the language *L*. (q_4 is "dead" state)

Example 1.1.8: Given $\Sigma = \{a, b\}$, construct a DFA which recognize the language $L = \{a^m b^n : m, n > 0\}$.

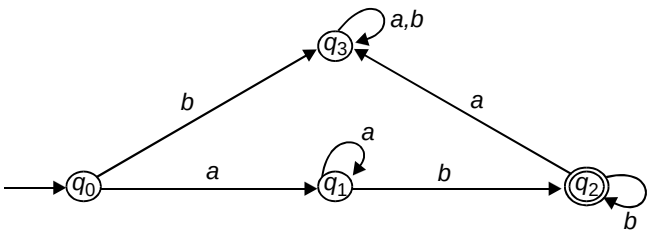
Solution

The given language $L = \{a^m b^n : m, n > 0\}$ has all words which begin with one or more *a*'s followed by one or more *b*'s.

The finite automaton $M(Q, \Sigma, \delta, q_0, F)$ is with

Q = $\{q_0, q_1, q_2, q_3\}$
Σ = $\{a, b\}$
q_0 = Initial state
$F = \{q_2\}$ = Final state
and δ as defined by language *L*.

The DFA is as shown below.



Here q_3 is a "dead" state.

Example 1.1.9: Construct a DFA which recognizes the set of all strings on $\Sigma = \{a, b\}$ starting with the prefix '*ab*'.

Solution

Only two states (q_1, q_2) are required to recognize *ab*, in addition to the input state. One additional state called the "trap" state is also required.

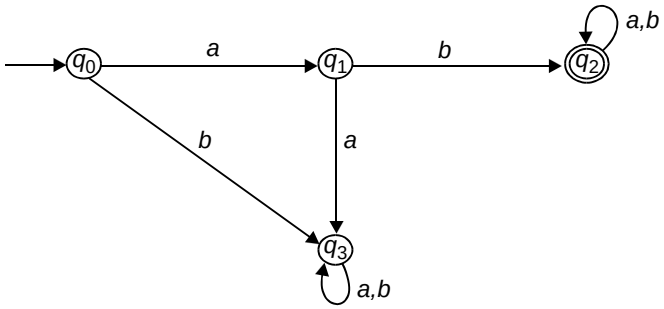


Fig. (a) DFA

Hence the DFA that recognizes the set of all strings on $\Sigma = \{a, b\}$ starting with the prefix '*ab*' is drawn above, where the automaton *M* is

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

with the state table diagram for δ as shown below.

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

Fig. (b) State table diagram

Example 1.1.10: Determine the DFA that will accept those words from $\Sigma = \{a, b\}$ where the number of *b*'s is divisible by three. Sketch the state table diagram of the finite Automaton *M* also.

Solution

The Finite Automaton *M* is $M(Q, \Sigma, \delta, q_0, F)$ with

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

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

q_0 = Initial state

F = Final state

We choose three states q_0, q_1, q_2. The states count the number of b's modulo 3, with q_0 as the input as well as accepting state where q_1 and q_2 are not accepting states. Run arrows from q_0 to q_1, q_1 to q_2 and q_2 to q_0 with label 'b'.

If any a is encountered, it does not alter the state. The suitable DFA is as shown in the figure (a).

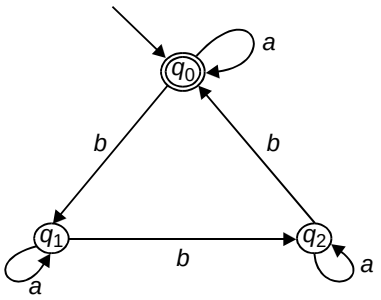


Fig. (a) DFA

The state table diagram is shown in Fig. (b).

δ	a	b
q_0	q_0	q_1
q_1	q_1	q_2
q_2	q_2	q_0

Fig. (b) State table diagram

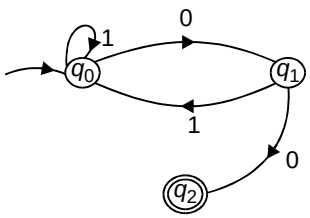
Example 1.1.11: Construct an FA accepting all strings in $\{0,1\}^*$ having even number of 0's.

Solution

The Finite Automaton M is given by

$$M(\{q_0, q_1, q_2\}, \{0,1\}, \delta, q_0, \{q_2\}).$$

The Finite Automaton is as shown.



Example 1.1.12: Construct a finite automaton accpting all strings over {0, 1}

(a) having odd number of 0's
(b) having even number of 1's.

Solution

(a) $M(\{q_0, q_1, q_2\}, \{0,1\}, \delta, q_0, \{q_1\})$. (*See* Fig. (a))
(b) $M(\{q_0, q_1, q_2, q_3\}, \{0,1\}, \delta, q_0, \{q_0\})$. (*See* Fig. (b))

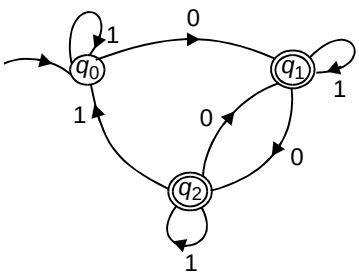


Fig. (a)

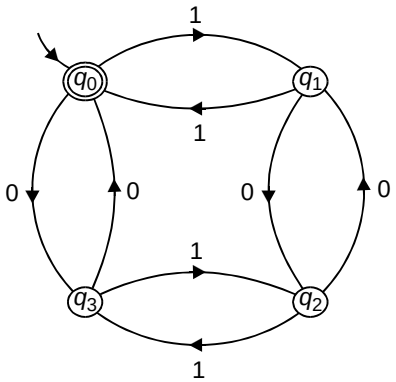
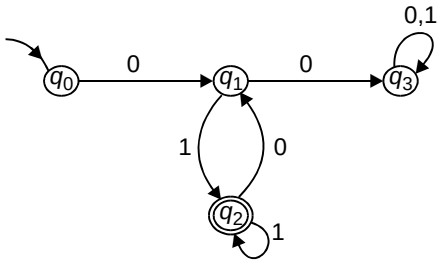


Fig. (b)

Example 1.1.13: Determine an FA, M accepting L,
where $L = \{w \in \{0,1\}^* : \text{Every 0 in } w \text{ has a 1 immediately to its right}\}$.

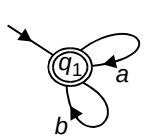
Solution



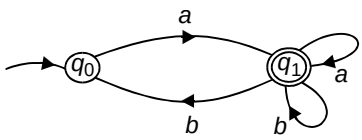
The finite automaton is given by

$$M(\{q_0, q_1, q_2, q_3\}, \{0,1\}, \delta, q_0, \{q_2\}).$$

Example 1.1.14: Determine the languages produced by the FA shown in Figs. (a) and (b).



(a)



(b)

Solution

(a) For $\Sigma = \{a, b\}$, language generated = $\{a,b\}^*$
($\in$ will be accepted when initial state equal final state).

(b) For $\Sigma = \{a, b\}$, language generated = $\{a, b\}^+$
{$\in$ is not accepted).

Example 1.1.15: Determine the FA if $\Sigma = \{a, b\}$ for

(a) Language generated $L_A = (ab)^* = \{(ab)^n \mid n \geq 0\}$
($\in$-not accepted)

(b) Language generated $L_B = \{(ab)^n \mid n \geq 1\}$
($\in$-not accepted)

Solution

(a) Given $L_A = \{(ab)^n \mid n \geq 0\}$.

The FA is shown below in Fig. (a).

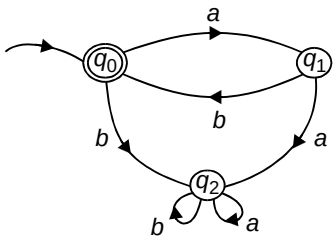


Fig. (a)

The FA is given by

$$M(\{q_0, q_1, q_2\}, \{a, b\}, \delta, q_0, q_0)$$

where q_2 is a "dead state".

(b) Given $L_B = \{(ab)^n \mid n \geq 1\}$ ($\in$-not accepted) i.e., initial state ≠ final state). The FA for this language L_B is shown in Fig. (b).

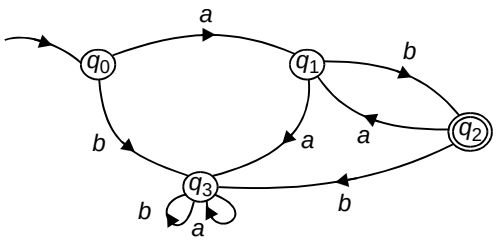


Fig.(b)

The FA is given by

$$M(\{q_0, q_1, q_2, q_3\}, \{a, b\}, \delta, q_0, q_2)$$

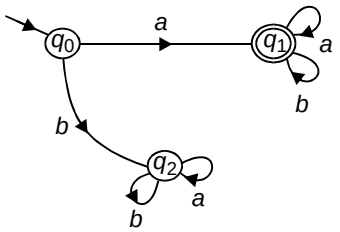
where q_3 is a "dead state".

Example 1.1.16: Determine the FA with the

(a) Set of strings beginning with an '*a*'.
(b) Set of strings beginning with '*a*' and ending with '*b*'.
(c) Set of strings having '*aaa*' as a subword.
(d) Set of integers
(e) Set of signed integers.

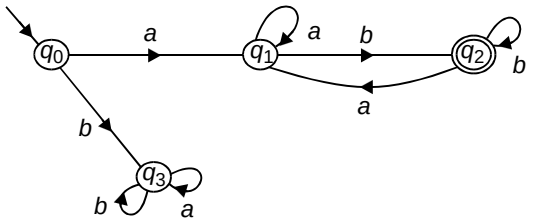
Solution

(a) Set of strings beginning with an '*a*'.

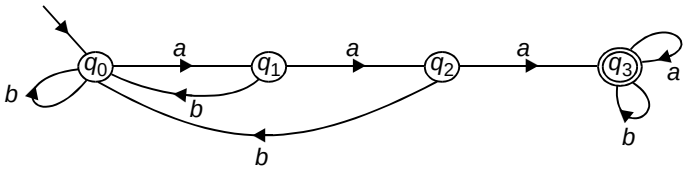


[It is not necessary always to have a dead state]

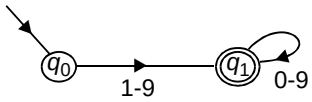
(b) Set of strings beginning with '*a*' and ending with '*b*'.



(c) Set of strings having '*aaa*' as a subword.

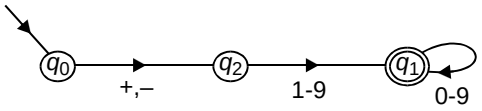


(d) Set of integers.



Alphabet $\Sigma = \{0,1, \ldots, 9\}$

(e) Set of signed Integers.



1.2 NON-DETERMINISTIC FINITE AUTOMATA (NFA)

Definition

A Nondeterministic Finite Automata (NFA) is defined by a 5-tuple

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

where $Q, \Sigma, \delta, q_0, F$ are defined as follows:

Q = Finite set of internal states
Σ = Finite set of symbols called "Input alphabet"
$\delta = Q \times (\Sigma \cup \{\lambda\}) \rightarrow 2^Q$
$q_0 \in Q$ is the Initial states
$F \subseteq Q$ is a set of Final states

NFA differs from DFA in that, the range of δ in NFA is in the powerset 2^Q.

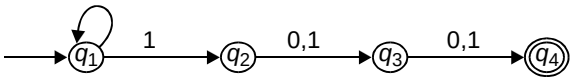
A string is accepted by an NFA if there is some sequence of possible moves that will put the machine in the final state at the end of the string.

Example 1.2.1: Obtain an NFA for a language consisting of all strings over {0,1} containing a 1 in the third position from the end.

Solution

q_1, q_2, q_3 are initial states

q_4 is the final state.



Please note that this is an NFA as $\delta(q_2,0) = q_3$ and $\delta(q_2,1) = q_3$.

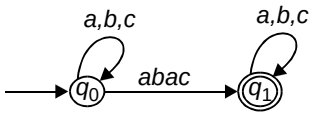
Example 1.2.2: Determine an NFA accepting the language

(a) $L_1 = \{x \mid x \in \{a, b, c\}^*$ and x contains the pattern $abac\}$

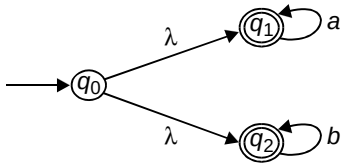
(b) $L_2 = \{a^* \cup b^*\}$

Solution

(a)



(b)



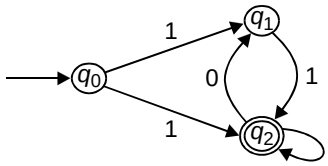
Example 1.2.3: Determine an NFA accepting all strings over {0,1} which end in 1 but does not contain the substring 00.

Solution

The conditions to be satisfied are:

(a) String should end in a 1

(b) String should not contain 00.



The NFA is shown in figure.

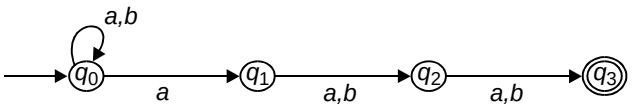
Example 1.2.4: Obtain an NFA which should accept a language L_A, given by $L_A = \{x \in \{a, b\}^* : |x| \geq 3$ and third symbol of x from the right is $\{'a'\}$.

Solution

The conditions are

(a) the last two symbols can be '*a*' or '*b*'.
(b) third symbol from the right is '*a*'
(c) symbol in any position but for the last three position can be '*a*' or '*b*'.

The NFA is shown in fig. below.



Example 1.2.5: Sketch the NFA state diagram for

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

with the state table as given below.

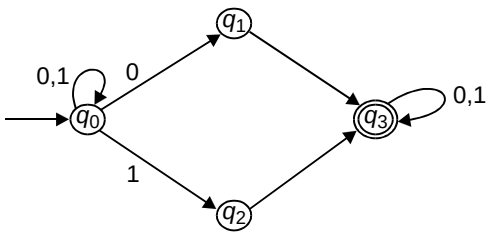
δ	0	1
q_0	q_0, q_1	q_0, q_2
q_1	q_3	$\varnothing$
q_2	$\varnothing$	q_3
q_3	q_3	q_3

Solution

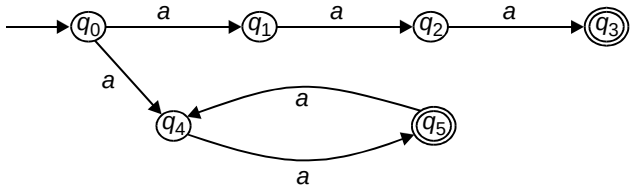
The NFA states are q_0, q_1, q_2 and q_3.

$$\delta(q_0,0) = \{q_0, q_1\} \qquad \delta(q_0,1) = \{q_0, q_2\}$$
$$\delta(q_1,0) = \{q_3\} \qquad \delta(q_2,1) = \{q_3\}$$
$$\delta(q_3,0) = \{q_3\} \qquad \delta(q_3,1) = \{q_3\} \ .$$

The NFA is as shown below.



Example 1.2.6: Given L is the language accepted by NFA in Fig. Determine an NFA that accepts $L \cup \{a^5\}$.



Solution

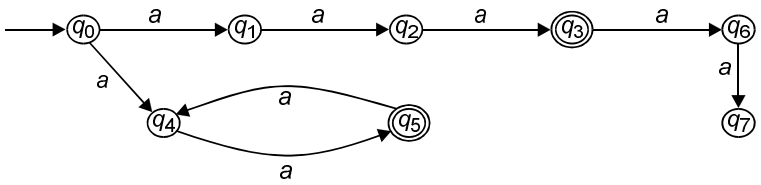
The language accepted by the given NFA is

$$L = \{a^3\} \cup \{a^n : n \text{ is odd}\}.$$

Now to make an NFA accepting the language:

$$L = \{a^3\} \cup \{a^n : n \text{ is odd}\} \cup \{a^5\}.$$

This is accomplished by adding two states after state q_3 viz., q_6 and q_7 as shown in fig.



The NFA is given by

$$M = (\{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7\}, \{a\}, \delta, q_0, \{q_3, q_5, q_7\})$$

Example 1.2.7: Find an NFA with four states for

$$L = \{a^n : n \geq 0\} \cup \{b^n a : n \geq 1\}$$

Solution

NFA for the language:

$$L = \{a^n : n \geq 0\} \cup \{b^n a : n \geq 1\}$$

For such a language two cases are to be considered.

Case (i): $a^n, n \geq 0$

q_0 goes to a state q_3 where all a's are absorbed. Hence a^n is accepted.

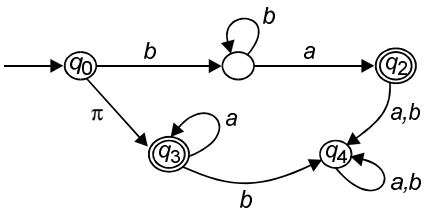
Case (ii): $b^n a : n \geq 1$

q_0 goes to a state q_1 where all b's are accepted and when an 'a' is encountered it goes to final state q_2. An additional state q_4 is added as a rejection state for the cases when 'b' is encountered after a's of case (i) or when 'a' or 'b' is encountered after $b^n a$ of case (ii).

The NFA is given by

$$M = (\{q_0, q_1, q_2, q_3, q_4\}, \{a, b\}, \delta, q_0, \{q_2, q_3\})$$

which is shown in the fig. below.



Example 1.2.8: Design an NFA with no more than five states for the set

$$\{abab^n : n \geq 0\} \cup \{aba^n : n \geq 0\}.$$

Solution

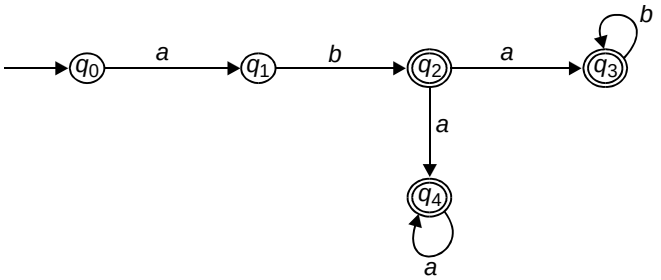
NFA for the language

$$L = \{abab^n : n \geq 0\} \cup \{aba^n : n \geq 0\}$$

is

$$M = (\{q_0, q_1, q_2, q_3, q_4\}, \{a, b\}, \delta, q_0, \{q_2, q_3, q_4\}).$$

Here the NFA is such that it accepts all strings of the type aba^n and $abab^n$ where $n \geq 0$.



q_2 is for the case when string is ab, i.e. ab^n with $n = 0$.
q_3 is for the case when string is $abab^n$ with $n \geq 0$.
q_4 is for the case when string is aba^n with $n \geq 0$

This NFA is shown in the fig. above.

Example 1.2.9: Determine an NFA with three states that accepts the language $\{ab, abc\}^*$.

Solution:

NFA for the language

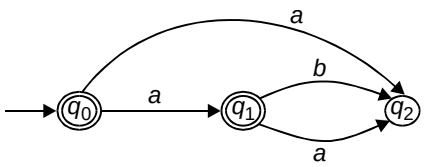
$$L = \{ab, abc\}^*$$

should be such that it accepts "*ab*" or "*abc*" in the first step and then this is looped with initial state so that any combination of "*ab*" and "*abc*" can be accepted.

Hence we have the NFA as

$$M = (\{q_0, q_1, q_2\}, \{a, b, c\}, \delta, q_0, \{q_1\})$$

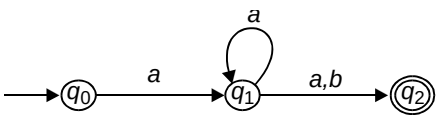
which is shown below:



Example 1.2.10: Determine an NFA that accepts the language

$$L(aa^*(a + b)).$$

Solution:



NFA is given by

$$M = (\{q_0, q_1, q_2\}, \{a, b\}, \delta, q_0, \{q_2\})$$

1.3 EQUIVALENCE OF NFA AND DFA

Definition

Two finite accepters M_1 and M_2 are equivalent iff

$$L(M_1) = L(M_2)$$

i.e., if both accept the same language.

Both DFA and NFA recognize the same class of languages. It is important to note that every NFA has an equivalent DFA.

Let us illustrate the conversion of NDA to DFA through an example.

Example 1.3.1: Determine a deterministic Finite State Automaton from the given Nondeterministic FSA.

$$M = (\{q_0, q_1\}, \{a, b\}, \delta, q_0, \{q_1\})$$

with the state table diagram for δ given below.

δ	a	b
q_0	$\{q_0, q_1\}$	$\{q_1\}$
q_1	$\varnothing$	$\{q_0, q_1\}$

Solution

Let $M' = (Q', \Sigma, \delta', q_0', F')$ be a determine. Finite state automaton (DFA), where

$Q' = \{[q_0], [q_1], [q_0, q_1], [\varnothing]\}$,
$q_0' = [q_0]$
and $F' = \{[q_1], [q_0, q_1]\}$

Please remember that [ ] denotes a single state. Let us now proceed to determine δ' to be defined for the DFA.

δ'	a	b
$[q_0]$	$[q_0, q_1]$	$[q_1]$
$[q_1]$	$\varnothing$	$[q_0, q_1]$
$[q_0,q_1]$	$[q_0,q_1]$	$[q_0,q_1]$
$\varnothing$	$\varnothing$	$\varnothing$

It is to be noted that

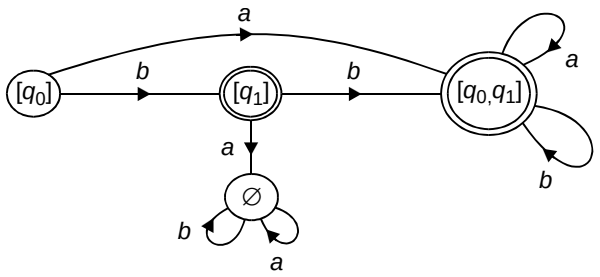
$$\delta'([q_0, q_1], a) = [q_0, q_1]$$

since
$$\begin{aligned}\delta'([q_0, q_1], a) &= \delta(q_0, a) \cup \delta(q_1, a)\\ &= \{q_0, q_1\} \cup \varnothing\\ &= \{q_1, q_1\}\end{aligned}$$

and
$$\delta'([q_0, q_1], b) = [q_0, q_1]$$

since
$$\begin{aligned}\delta([q_0, q_1], b) &= \delta(q_0, b) \cup \delta(q_1, b)\\ &= \{q_1\} \cup \{q_0, q_1\}\\ &= \{q_0, q_1\}\end{aligned}$$

Here any subset containing q_1 is the final state in DFA. This is shown as below.



Example 1.3.2: Given the NDA as shown in Fig. (a), with δ as shown in Fig. (b).

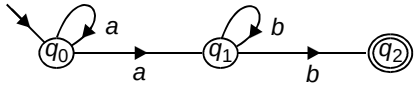


Fig. (a)

	a	*b*
q_0	$\{q_0, q_1\}$	$\varnothing$
q_1	$\varnothing$	$\{q_1, q_2\}$
q_2	$\varnothing$	$\varnothing$

Fig. (b)

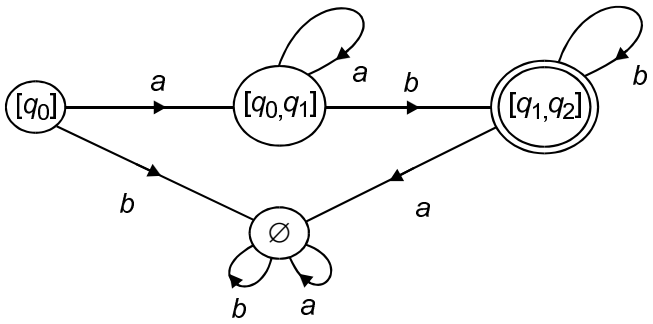
Determine the equivalent DFA for the above given NDA.

Solution

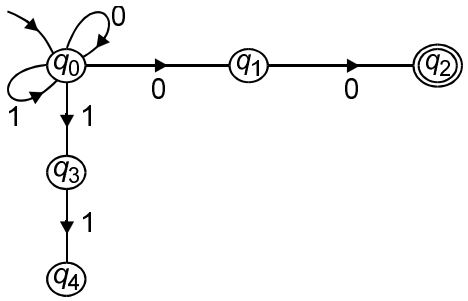
Conversion of NDA to DFA is done through subset construction as shown in the State table diagram below.

	a	*b*
$[q_0]$	$[q_0, q_1]$	$\varnothing$
$[q_0, q_1]$	$[q_0, q_1]$	$[q_1, q_2]$
$[q_1 , q_2]$	$\varnothing$	$[q_1, q_2]$
$\varnothing$	$\varnothing$	$\varnothing$

The corresponding DFA is shown below. Please note that here any subset containing q_2 is the final state.



Example 1.3.3: Given the NDA as shown in fig. below, determine the equivalent DFA.



Solution

The given NDA has q_2 and q_4 as final states. It accepts strings ending in 00 or 11. The state table is shown below.

	0	1
q_0	$\{q_0, q_1\}$	$\{q_0, q_3\}$
q_1	$\{q_2\}$	$\varnothing$
q_2	$\varnothing$	$\varnothing$
q_3	$\varnothing$	$\{q_4\}$
q_4	$\varnothing$	$\varnothing$

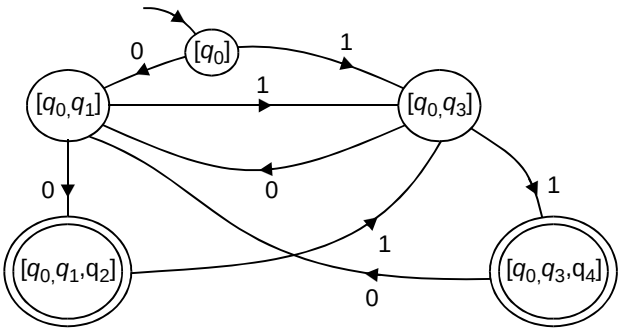
The conversion of NDA to DFA is done through the subset construction.

δ' is given by the following state table.

	0	1
$\rightarrow [q_0]$	$[q_0, q_1]$	$[q_0, q_3]$
$[q_0, q_1]$	$[q_0, q_1, q_2]$	$[q_0, q_3]$
$[q_0, q_3]$	$[q_0, q_1]$	$[q_0, q_3, q_4]$
$[q_0, q_1, q_2]$	$[q_0, q_1, q_2]$	$[q_0, q_3]$
$[q_0, q_3, q_4]$	$[q_0, q_1]$	$[q_0, q_3, q_4]$

Any state containing q_2 or q_4 will be a final state.

The DFA is shown below.



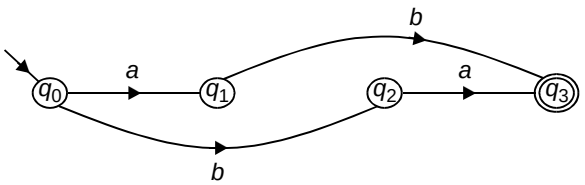
Example 1.3.4: Determine a NFA accepting $\{ab, ba\}$ and use it to find a DFA accepting it.

Solution

The state table is as shown below.

	a	*b*
q_0	q_1	q_2
q_1	$\varnothing$	q_3
q_2	q_3	$\varnothing$
q_3	$\varnothing$	$\varnothing$

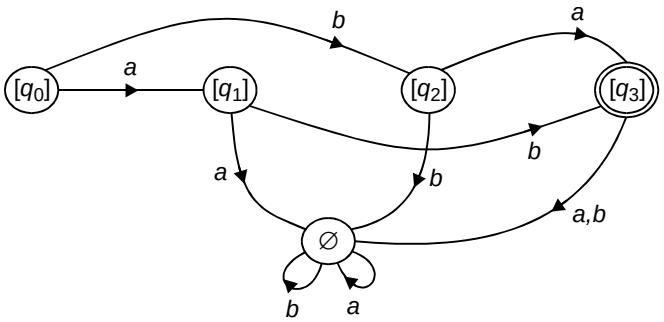
The NFA is shown below.



q_0 is the input state, q_3 is the final state.

The state table corresponding to the DFA is derived by using subset construction. State table for DFA is as shown below.

	a	b
$[q_0]$	$[q_1]$	$[q_2]$
$[q_1]$	$\varnothing$	$[q_3]$
$[q_2]$	$[q_3]$	$\varnothing$
$[q_3]$	$\varnothing$	$\varnothing$
$\varnothing$	$\varnothing$	$\varnothing$



The DFA is as shown above.

1.4 REGULAR EXPRESSION

1.4.1 Regular Languages

The regular languages are those languages that can be constructed from the "big three" set operations viz., (a) Union (b) Concatenation (c) Kleene star.

A regular language is defined as follows.

Definition: Let Σ be an alphabet. The class of "regular languages" over Σ is defined inductively as follows:

(a) $\varnothing$ is a regular language
(b) For each $\sigma \in \Sigma$, $\{\sigma\}$ is a regular language
(c) For any natural number $n \geq 2$ if $L_1, L_2, \ldots\ldots L_n$ are regular languages, then so is $L_1 \cup L_2 \cup \ldots \ldots \cup L_n$.
(d) For any natural number $n \geq 2$, if $L_1, L_2, \ldots\ldots L_n$ are regular languages, then so is $L_1 \circ L_2 \circ \ldots \ldots \circ L_n$.
(e) If L is a regular language, then so is L^*.
(f) Nothing else is a regular language unless its construction follows from rules (a) to (e).

Examples:

(i) $\varnothing$ is a regular language (by rule (a))

(ii) $L = \{a, ab\}$ is a language over $\Sigma = \{a, b\}$ because, both $\{a\}$ and $\{b\}$ are regular languages by rule (b). By rule (d) it follows that $\{a\} \circ \{b\} = \{ab\}$ is a regular language. Using rule (c), we see that $\{a\} \cup \{ab\} = L$ is a regular language.

(iii) The language over the alphabet {0,1} where strings contain an even number of 0's can be constructed by

$$(1^*((01^*)(01^*))^*)$$

or simply $1^*(01^*\ 01^*)^*$.

1.4.2 Regular Expressions

Regular expressions were designed to represent regular languages with a mathematical tool, a tool built from a set of primitives and operations.

This representation involves a combination of strings of symbols from some alphabet Σ, parantheses and the operators +, $\cdot$, and *.

A regular expression is obtained from the symbol $\{a, b, c\}$, empty string $\in$, and empty-set $\varnothing$ perform the operations +, $\cdot$ and * (union, concatenation and Kleene star).

Examples

$0 + 1$ represents the set $\{0, 1\}$

1 represents the set $\{1\}$

0 represents the set $\{0\}$

$(0 + 1)\ 1$ represents the set $\{01, 11\}$

$(a + b) \cdot (b + c)$ represents the set $\{ab, bb, ac, bc\}$

$$(0 + 1)^* = \in + (0 + 1) + (0 + 1)\ (0 + 1) + \cdots\cdots = \Sigma^*$$

$$(0+1)^+ = (0+1)\,(0+1)^* = \Sigma^+ = \Sigma^* - \{\varepsilon\}$$

1.4.3 Building Regular Expressions

Assume that $\Sigma = \{a, b, c\}$

Zero or more: a^* means "zero or more *a*'s",

To say "zero or more *ab*'s," i.e., $\{\lambda, ab, abab, \ldots\ldots\}$ you need to say $(ab)^*$.

One or more: Since a^* means "zero or more *a's*", you can use aa^* (or equivalently a^*a) to mean "one or more *a*'s". Similarly to describe 'one or more *ab*'s", that is $\{ab, abab, ababab, \ldots\ldots\}$, you can use $ab\ (ab)^*$.

Zero or one: It can be described as an optional '*a*' with $(a + \lambda)$.

Any string at all: To describe any string at all (with $\Sigma = \{a, b, c\}$ you can use $(a + b + c)^*$.

Any nonempty string: This is written any character from $\Sigma = \{a, b, c\}$ followed by any string at all: $(a + b + c)(a + b + c)^*$

Any string not containing .....: To describe any string at all that does not contain an '*a*' (with $\Sigma = \{a, b, c\}$), you can use $(b + c)^*$.

Any string containing exactly one .....: To describe any string that contains exactly one '*a*' put "any string not containing an *a*", on either side of the '*a*' like: $(b + c)^* a (b + c)^*$.

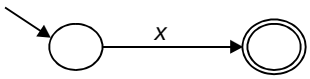
1.4.4 Languages defined by Regular Expressions

There is a very simple correspondence between regular expressions and the languages they denote:

Regular expression	L (Regular Expression)
x, for each $x \in \Sigma$	$\{x\}$
λ	$\{\lambda\}$
$\varnothing$	$\{\ \}$
(r_1)	$L(r_1)$
r_1^*	$(L(r_1))^*$
$r_1 r_2$	$L(r_1)L(r_2)$
$r_1 + r_2$	$L(r_1) \cup L(r_2)$

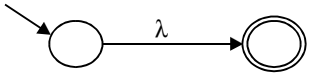
1.4.5 Regular Expressions to NFA

(i) For any x in Σ, the regular expression denotes the language $\{x\}$. The NFA (with a single start state and a single final state) as shown below, represents exactly that language.



NFA for x

(ii) The regular expression λ denotes the language $\{\lambda\}$that is the language containing only the empty string.

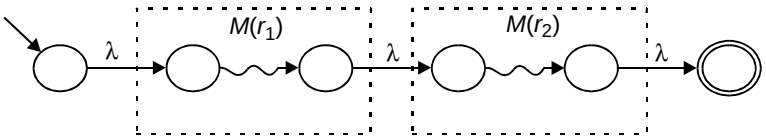


NFA for λ

(iii) The regular expression ∅ denotes the language ∅; no strings belong to this language, not even the empty string.

NFA for ∅

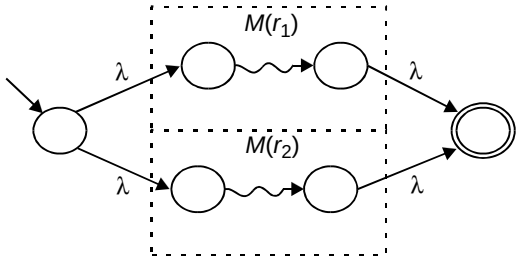
(iv) For juxtaposition, strings in $L(r_1)$ followed by strings in $L(r_2)$, we



NFA for $r_1 r_2$

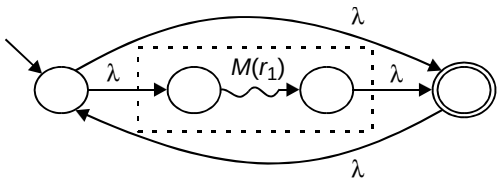
chain the NFAs together as shown.

(v) The "+" denotes "or" in a regular expression, we would use an NFA with a choice of paths.



NFA for $r_1 + r_2$

(vi) The star (*) denotes zero or more applications of the regular expression, hence a loop has to be set up in the NFA.



1.4.6 NFAs to Regular Expression

The basic approach to convert NFA, to Regular Expressions is as follows:

(i) If an NFA has more than one final state, convert it to an NFA with only one final state. Make the original final states nonfinal, and add a λ-transition from each to the new (single) final state.

(ii) Consider the NFA to be a generalised transition graph, which is just like an NFA except that the edges may be labeled with arbitrary regular expressions. Since the labels on the edges of an NFA may be either λ or members of each of these can be considered to be a regular expression.

(iii) Removes states one by one from the NFA, relabeling edge as you go, until only the initial and the final state remain.

(iv) Read the final regular expression from the two state automaton that results.

The regular expression derived in the final step accepts the same language as the original NFA.

Example 1.4.1: Represent the following sets by regular expression

(a) $\{\wedge, ab\}$
(b) $\{1, 11, 111, \ldots\ldots\}$
(c) $\{ab, a, b, bb\}$

Solution

(a) The set $\{\wedge, ab\}$ is represented by the regular expression $\wedge + ab$
(b) The set $\{1, 11, 111, \ldots\ldots\}$ is got by concatenating 1 and any element of $\{1\}^*$. Therefore $1(1)^*$ represent the given set.
(c) The set $\{ab, a, b, bb\}$ represents the regular expression $ab + a + b + bb$.

Example 1.4.2: Obtain the regular expressions for the following sets:

(a) The set of all strings over $\{a, b\}$ beginning and ending with 'a'.
(b) $\{b^2, b^5, b^8, \ldots\ldots\}$
(c) $\{a^{2n+1} \mid n > 0\}$

Solution

(a) The regular expression for 'the set of all strings over $\{a, b\}$ beginning and ending with 'a' is given by:

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

(b) The regular expression for $\{b^2, b^5, b^8, \ldots\ldots\}$ is given by:

$$bb\,(bbb)^*$$

(c) The regular expression for $\{a^{2n+1} \mid n > 0\}$ is given by:

$$a\,(aa)^*$$

Example 1.4.3: Obtain the regular expressions for the languages given by

(a) $L_1 = \{a^{2n} b^{2m+1} \mid n \geq 0, m \geq 0\}$
(b) $L_2 = \{a,\ bb,\ aa,\ abb,\ ba,\ bbb,\ \ldots\ldots\}$
(c) $L_3 = \{w \in \{0,1\}^* \mid w$ has no pair of consecutive zeros$\}$
(d) $L_4 = \{$strings of 0's and 1's ending in 00$\}$

Solution

(a) $L_1 = \{a^{2n} b^{2m+1} \mid n \geq 0, m \geq 0\}$ denotes the regular expression

$$(aa)^*(bb)^* b$$

(b) The regular expression for the language $L_2 = \{a,\ bb,\ aa,\ abb,\ ba,\ bbb,\ \ldots\ldots\}$

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

(c) The regular expression for the language $L_3 = \{w \in \{0,1\}^* \mid w$ has no pair of consecutive zeros$\}$ is given by

$$(1^*011^*)^* (0 + \lambda) + 1^* (0 + \lambda)$$

(d) The regular expression for the language $L_4 = \{$strings of 0's and 1's beginning with 0 and ending with 1$\}$ is given by

$$0\ (0 + 1)^* 1$$

Example 1.4.4: Describe the set represented by the regular expression $(aa + b)^* (bb + a)^*$

Solution

The given regular expression is

$$(aa + b)^* (bb + a)^*.$$

The English language description is as follows: "The set of all the strings of the form uv where a's are in pairs in u and b's are in pairs in v".

Example 1.4.5: Give Regular expressions for the following on $\Sigma = \{a, b, c\}$

(a) all strings containing exactly one a
(b) all strings containing no more than three a's
(c) all strings which contain at least one occurrence of each symbol in Σ.

(d) all strings which contain no runs of *a*'s of length greater than two.
(e) all strings in which all runs of *a*'s have lengths that are multiples of three.

Solution

(a) R.E = $(b + c)^* a (b + c)^*$ [for all strings containing exact one *a*]

(b) All strings containing no more than three *a*'s: We can describe the string containing zero, one, two or three *a*'s (and nothing else) as

$$(\lambda + a)(\lambda + a)(\lambda + a)$$

Now we want to allow arbitrary strings not containing *a*'s at the places marked by *X*'s:

$$X(\lambda + a)X(\lambda + a)X(\lambda + a)X$$

Therefore we put $(b + c)^*$ for each *X*.

$$(b+c)^*(\lambda + a)(b+c)^*(\lambda + a)(b+c)^*(\lambda + a)(b+c)^*$$

(c) *All strings which contain at least one occurrence of each symbol in* Σ*:*
Here we cannot assume the symbols are in any particular order. We have no way of saying "in any order', so we have to list the possible orders:

$$abc + acb + bac + bca + cab + cba$$

Let us put *X* in every place where we want to allow an arbitrary string:

$$XaXbXcX + XaXcXbX + XbXaXcX + XbXcXaX + XcXaXbX + XcXbXaX$$

Finally, we replace all X's with $(a + b + c)^*$ to get the final regular expression:

$$\begin{aligned}
&(a+b+c)^* a(a+b+c)^* b(a+b+c)^* c(a+b+c)^* + \\
&(a+b+c)^* a(a+b+c)^* c(a+b+c)^* b(a+b+c)^* + \\
&(a+b+c)^* b(a+b+c)^* a(a+b+c)^* c(a+b+c)^* + \\
&(a+b+c)^* b(a+b+c)^* c(a+b+c)^* a(a+b+c)^* + \\
&(a+b+c)^* c(a+b+c)^* a(a+b+c)^* b(a+b+c)^* + \\
&(a+b+c)^* c(a+b+c)^* b(a+b+c)^* a(a+b+c)^*
\end{aligned}$$

(d) *All strings which contain no runs of a's of length greater than two:* An expression containing no *a*, one *a*, or one *aa*:

$$(b + c)^* (\lambda + a + aa)(b + c)^*$$

But if we want to repeat this, we have to ensure to have least one non-*a* between repetitions:

$$(b+c)^*(\lambda + a + aa)(b+c)^*((b+c)(b+c)^*(\lambda + a + aa)(b+c)^*)^*$$

(e) *All strings in which all runs of* a*'s have lengths that are multiples of three:*

$$(aaa + b + c)^*$$

Example 1.4.6: Find regular expressions over $\Sigma = \{a, b\}$ for the language defined as follows:

(a) $L_1 = \{a^m b^m : m > 0\}$

(b) $L_2 = \{b^m a b^n : m > 0, n > 0\}$

(c) $L_3 = \{a^m b^m, m > 0, n > 0\}$

Solution

(a) Given $L_1 = \{a^m b^m : m > 0\}$,

L_1 has those words beginning with one or more *a*'s followed by one or more *b*'s.

Therefore the regular expression is

$$aa^*bb^* \text{ (or) } a^*ab^*b$$

(b) Given $L_2 = \{b^m a b^n : m > 0, n > 0\}$. This language has those words *w* whose letters are all *b* except for one '*a*' that is not the first or last letter of *w*.

Therefore the regular expression is

$$bb^*abb^*$$

(c) Given $L_3 = \{a^m b^m, m > 0\}$.

There is no regular expression for this beginning as L_3 is not regular.

Example 1.4.7: Determine all strings in $L((a+b)^*b(a+ab)^*)$ of length less than four.

Solution

b, ab, bb, ba, aab, abb, bab, bbb, baa, bba, aba

Example 1.4.8: Find the regular expressions for the languages defined by

(i) $L_1 = \{a^n b^m : n \geq 1,\ m \geq 1,\ nm \geq 3\}$

(ii) $L_2 = \{ab^n w : n \geq 3, w \in \{a, b\}^+\}$

(iii) $L_3 = \{vwv : v, w \in \{a, b\}^*, |v| = 2\}$

(iv) $L_4 = \{w : |w| \bmod 3 = 0\}$

Solution:

(i) Regular Expression for $L_1 = \{a^n b^m : n \geq 1,\ m \geq 1,\ nm \geq 3\}$ is given by

$$aa\,(a^*)\,b(b^*) + a(a^*)\,bb\,(b^*)$$

(ii) Regular Expression for $L_2 = \{ab^n w : n \geq 3, w \in \{a, b\}^+\}$ is given by

$$abbb\,(b^*)\,(a + b)\,(a + b)^*$$

(iii) Regular Expression for $L_3 = \{vwv : v, w \in \{a, b\}^*, |v| = 2\}$ is given by

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

(iv) The regular expression for $L_4 = \{w : |w| \bmod 3 = 0\}$ is given by

$$(aaa + bbb + ccc + aab + aba + abb + bab + bba + cab + cba + cbb + caa)^*$$