Banasthali Vidyapith

B.Tech VI Semester (CS)

Theory Of Computation (CS 315)

ASSIGNMENT -1

Submitted By →

Name :- Radhika Agrawal

Section :- 'B'

Roll no :- 1812830

ID :- BTBTC18037

Submitted To →

Dr. Neelam Sharma

(Department of Mathematics & Computing)

Que-1. (a) Why study Automaton Theory? Briefly describe any two applications of FA.

In order to understand why study Automata Theory, we need to know what exactly is Automaton and Automaton Theory.

Automaton is an abstract computing device and Automaton theory is the study of abstract computing devices or machines.

Automata Study is the study of abstract machine. It is important because it allows scientists to understand how machines solve problem.

An automata is any machine that uses a specific repetable process to convert information into different forms. If Scientists did not Studied Automata theory, they would have a much more difficult time designing Systems that could perform repeatable actions based on specific inputs and outputs and some other reason for studying automata are :-

(i) FINITE AUTOMATA → They are useful model for many kind of hardware and Software.

(ii) STRUCTURAL REPRESENTATION → Regular expression are denoted the structure of data, especially text string.

→ Applications of Finite Automata (FA) :-

- for the designing of lexical analysis of a computer.
- for recognizing the pattern using regular expression.
- for the designing of combination and Sequential -

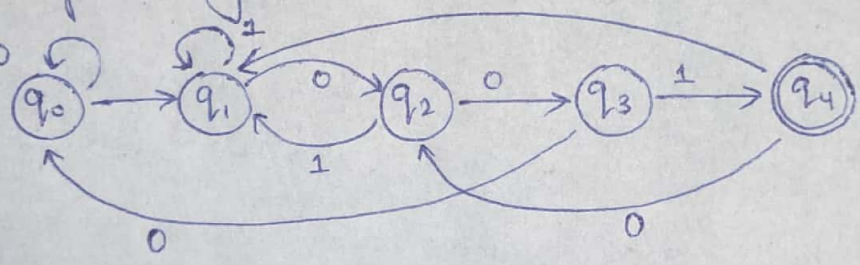
circuits using machines.
- Used in text editors
- for the implementation of spell checkers.

ii] String Processing :-

Consider finding all occurence of a short string (pattern string) within a long string (text string). This can be done by processing the text through a DFA. The DFA for all strings that end with pattern string. Each time the accept state is reached, the current position in the text is output.

Ex:- Finding 1001

To find all occurence of pattern 1001, construct DFA for all strings ending in 1001.



ii] Lexical Analysis :-

In compiling a program, the first step is lexical analysis. This isolates keywords, identifiers etc while eliminating irrelevant symbols.

A token is a category for ex; "identifier", "relation operator" OR specific keywords.

for ex:-

token RE

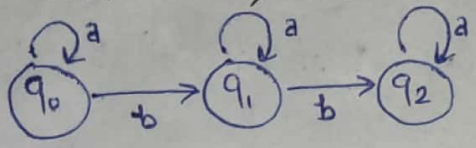
keyword then then

Variable name [a-z A-Z] [a-z A-Z 0-9]

where letter RE says it is any string of alphanumeric characters starting with a letter.

1(b) Define the regular expressions for the following.

(i) The set of all strings over $\Sigma\{a,b\}$ containing all strings with exactly 2b's.



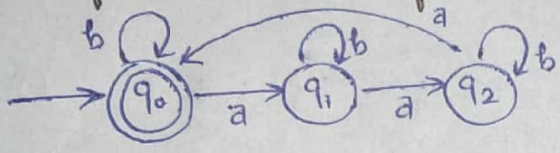
Alphabet ; $\Sigma = \{a, b\}$

The set contains all strings with exactly 2b's so language can be as follows :-

$L(R) = \{bb, abb, bba, abba, bab \text{ ---} \}$

So, Regular expression → $\boxed{R.E. = a^* ba^* ba^*}$

(ii) The set of all strings over $\Sigma\{a,b\}$ in which number of occurrences of a is divisible by 3.



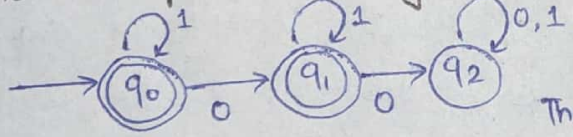
Alphabet; $\Sigma = \{a, b\}$

Here, language can be as follows:

$L(R) = \{aaa, baaa, aaab, ababa \text{ ---} \}$

So, Regular expression → $\boxed{R.E = (b^* ab^* ab^* ab^*)}$

(iii) The set of all strings over $\Sigma\{0,1\}$ containing at most one 0.



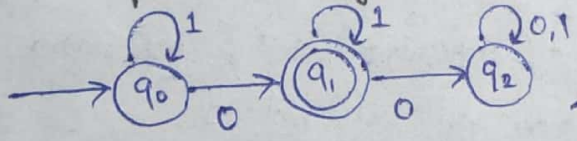
Alphabet ; $\Sigma = \{0, 1\}$

The set contains of all strings containing atmost one 0. So language can be as follows :-

$L(R) = \{\varepsilon, 1, 01, 0, 10, 101 \text{ --} \}$

So, $\boxed{R.E = 1^* + 1^* 01^*}$

(iv) The set of all strings over $\{0,1\}$ containing exactly one 0.



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

The set consists of all strings containing exactly one 0, so language can be as follows: $L(R) = \{0, 10, 01, 101, 1101, \text{ ---} \}$

So, $\boxed{R.E = 1^* 01^*}$

Que-2. Prove or disprove the following :-

i) $(0+(01)*0+0*(10)*)^* = (0+10)^*$

L.H.S = $(0+(01)*0+0*(10)*0)$

$= (0+0(10)^* + 0^*(10)^*)^*$ [$\because P(Q)^*P = P(QP)^*$]

$= (0^*(\varepsilon + (10)^* + (10)^*))^*$ [Since, 0 was coming in starting after each union]

$= (0^*(\varepsilon + (10)^*))^*$ [$\because R+R=R \rightarrow R^*+R^*=R^*$]

$= (0^* + 0^*(10)^*)^*$ [$\because R(P+Q) = RP+PQ$]

$= (0^*((0^*)(10)^*)^*)^*$

$= (0^*(0+10)^*)^*$

$= (0+0+10)^*$ [$\because (P^*+Q^*)^* = (P+Q)^*$]

$= (0+10)^* = R.H.S$

Hence, L.H.S = R.H.S Proved

ii) $(r+s)^* = r^* + s^*$

Here, the language set of the L.H.S regular expression can be $L(R) = \{\varepsilon, rsr, ssss, rssrs, \text{---}\}$

whereas, the language set of R.H.S regular expression can be $L(R') = \{\varepsilon, rrrr, \text{---} ssss \text{---}\}$

So, here we can see that L(R') contains ε and repetitive r's or s's and not the combination of r and s both.

However, L(R) contains ε, repetitive r's, repeatative s's and even all the combination of r and s.

for ex:- If we have $(0+1)^*$ by this we can generate any combination of 0 and 1 but for $(r+s)^* = r^*+s^*$ we can not generate 01 string.

So, Both Regular expressions can never be equal. L.H.S ≠ R.H.S

Que-3. Design a DFA which accepts set of all strings which are divisible by 7 for binary alphabets.

We have to design a DFA which will accept set of all strings which are divisible by 7 for binary alphabet.

So,

Numbers :	7	8	9	10	11	12	13	14
Rem :	0	1	2	3	4	5	6	0

Here, for the above following numbers, the remainders are given on dividing them by 7.

So, we can see that we have to make total 7 States $(q_0, q_1, q_2, q_3, q_4, q_5, q_6)$ according to remainder.

Now, let's deduce the transition on input 0 & 1, to make this DFA. [m= Some random english alphabet]

$q_0 \rightarrow 2(7m)$
$0 \Rightarrow 14m$
$1 \Rightarrow 14m+1$

$q_1 \rightarrow 2(7m+1)$
$0 \Rightarrow 14m+2$
$1 \Rightarrow 14m+3$

$q_2 \rightarrow 2(7m+2)$
$0 \Rightarrow 14m+4$
$1 \Rightarrow 14m+5$

$q_3 \rightarrow 2(7m+3)$
$0 \Rightarrow 14m+6$
$1 \Rightarrow 14m+6+1$

$q_4 \rightarrow 2(7m+4)$
$0 \Rightarrow 14m+6+2$
$1 \Rightarrow 14m+6+3$

$q_5 \rightarrow 2(7m+5)$
$0 \Rightarrow 14m+6+4$
$1 \Rightarrow 14m+6+5$

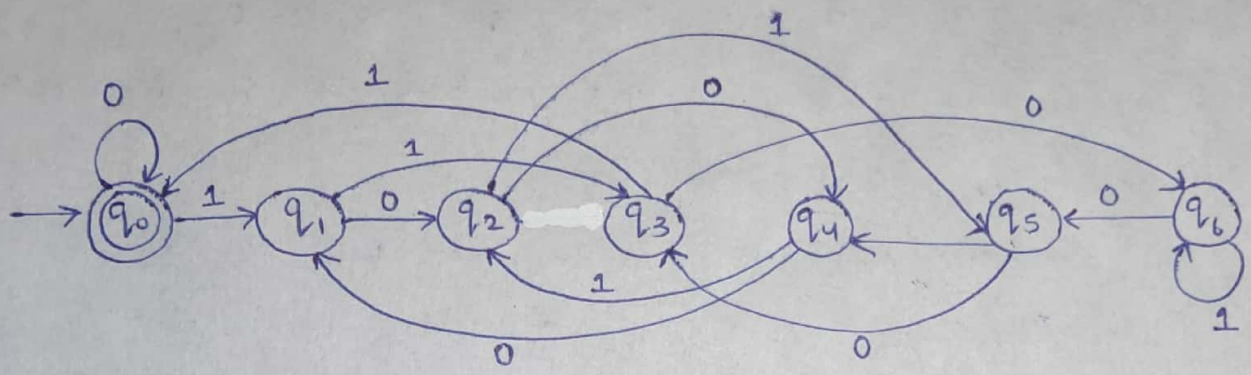
$q_6 \rightarrow 2(7m+6)$
$0 \Rightarrow 14m+6+6$
$1 \Rightarrow 14m+6+6+1$

So, here we can see that on a given state, there are given inputs 0 and 1 with their related transition.

for ex:- q_0 is the state where on 0 input, the transition will be on q_0 as (14m+0) has 0 after it.

Similarly we will see for input 1 the transition will be on q_1 as (14m + 1) has 1 after it.

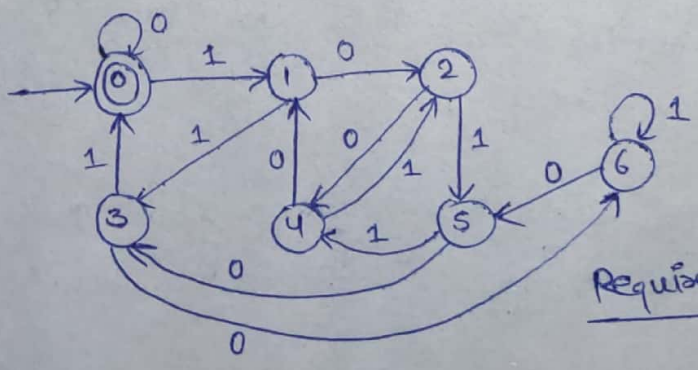
That is how we are going to draw our DFA.



Hence, this is required DFA.

OR, we can write like this also :-

Q x Σ	Input 0	Input 1
q_0	q_0	q_1
q_1	q_2	q_3
q_2	q_4	q_5
q_3	q_6	q_0
q_4	q_1	q_2
q_5	q_3	q_4
q_6	q_5	q_6



Required DFA

Que-4. Explain the following algebric laws for regular expressions :-

i] Identities and Annihilators →

An IDENTITY for an operator is a value such that when the operator is applied to the identity and some other value, the result is the other value.

There are two laws for Regular expressions involving these concepts, those are :-

- $\phi + L = L + \phi = L$, This law asserts that ϕ is the identity of union.
- $\varepsilon L = L\varepsilon = L$, This law asserts that ε is the identity for Concatenation.

An ANHILATOR for an operation is a value such that when the operator is applied to the anhilator and some other value, the result is the anhilator.

- $\phi L = L\phi = \phi$ This law asserts that ϕ is anhilator for concatenation.

ii] The Idempotent law →

An operator is said to be idempotent if the result of applying it to two of the same values as arguments is that value. The common arithmetic operators are not idempotent; $x + x \neq x$ in general and $x \times x \neq x$ in general (although there are some values of x which, equality holds, such as $0+0=0$). However, union and Intersection are common examples of idempotent operators. Thus, for regular expressions, we may assert following law:

- $L + L = L$, this law; the idempotent law for union, states that if we take the union of two identical expression, we can replace them by one copy of the expression