RGPVNOTES.IN

Subject Name: **Theory of Computation**

Subject Code: **IT-5001**

Semester: **5th**

Syllabus:

Automata:

Basic machine, FSM , Transition graph, Transition matrix, Deterministic and non-deterministic FSM'S, Equivalence of DFA and NDFA, Mealy & Moore machines, minimization of finite automata, Two-way finite automata.

Regular Sets and Regular Grammars:

Alphabet, words, Operations, Regular sets, Finite automata and regular expression, Myhill-Nerode theorem Pumping lemma and regular sets, Application of pumping lemma, closure properties of regular sets.

1. Automata

Automata Theory deals with definitions and properties of different types of "computation models". Examples of such models are:

- **Finite Automata:** These are used in text processing, compilers, and hardware design.
- **Context-Free Grammars:** These are used to define programming languages and in Artificial Intelligence.
- **Turing Machines** : These form a simple abstract model of a "real" computer, such as your PC at home.

1.1 Finite State Machine (FSM) or Finite Automata :

A finite automaton is a 5-tuple M = (Q, Σ, δ, q, F), where

1. Q is a finite set, whose elements are called states,
2. Σ is a finite set, called the alphabet; the elements of Σ are called symbols,
3. δ : Q × Σ → Q is a function, called the transition function,
4. q is an element of Q; it is called the start state or initial state,
5. F is a subset of Q; the elements of F are called accept states or final state.

1.2 An example of finite automata

Let A = {w : w is a binary string containing an odd number of 1s}.

We claim that this language A is regular. In order to prove this, we have to construct a finite automaton M such that A = L(M).

How to construct M? Here is a first idea: The finite automaton reads the input string w from left to right and keeps track of the number of 1s it has seen. After having read the entire string w, it checks whether the number of 1s is odd (in which case w is accepted) or even (in which

case w is rejected). Using this approach, the finite automaton needs a state for every integer i ≥ 0; indicating that the number of 1s read so far is equal to i. Hence, to design a finite automaton that follows this approach, we need an infinite number of states. But, the definition of finite automaton requires the number of states to be finite. A better, and correct approach, is to keep track of whether the number of 1s read so far is even or odd. This leads to the following finite automaton:

- The set of states is Q = {qe , qo }. If the finite automaton is in state qe , then it has read an even number of 1s; if it is in state qo , then it has read an odd number of 1s.
- The alphabet is Σ = {0, 1}.
- The start state is qe , because at the start, the number of 1s read by the automaton is equal to 0, and 0 is even.
- The set F of accept states is F = {qo }.
- The **transition function δ** is given by the following table: by the following table:

	0	1
q_e	q_e	q_0
q_0	q_0	q_e

Fig 1.1 :Transition Table/Matrix

This finite automaton M = (Q, Σ, δ, qe , F) can also be described by its state diagram, which is given in the figure below. The arrow that comes “out of the blue” and enters the state qe , indicates that qe is the start state. The state depicted with double circles indicates the accept state. We have constructed a finite automaton M that accepts the language A. Therefore, A is a regular language.

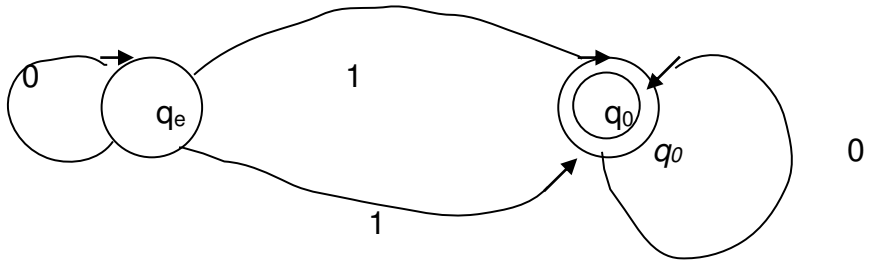

Fig1.2 : Transition Graph

1.3 Transition Graph : it is a finit directed labeled graph in which each vertex (or node)

represent a state and the directed edges indicate the transition of state. Edges are labeled with input symbol .

1.4 Transition Matrix : It is two dimension matrix between states of automata and Input symbol. Elements of matrix are state form mapping (Σ X Q) into Q.

1.5 Deterministic FSM(automata):

A deterministic automaton is a automaton (DFA) which is a finite state machine where for each pair of state and input symbol there is one and only one transition to a next state. DFAs recognize the set of regular languages and no other languages. The fig 1.1 and Fig 1.2 are depict of DFA.

DFA stands for Deterministic Finite Automata.

1.6 Non-Deterministic FSM(automata):

A non-deterministic finite automaton (NFA) or non-deterministic finite state machine is a finite state machine where from each state and a given input symbol the automaton may jump into several possible next states. This distinguishes it from the deterministic finite automaton (DFA), where the next possible state is uniquely determined.

1.6.1 Definition of NDFA

A non-deterministic finite automaton (NFA) is a 5-tuple

M = (Q, Σ, δ, q, F), where

1. Q is a finite set, whose elements are called states,
2. Σ is a finite set, called the alphabet; the elements of Σ are called symbols,
3. δ : Q × Σϙ → P(Q) is a function, called the transition function,
4. q is an element of Q; it is called the start state,
5. F is a subset of Q; the elements of F are called accept states.

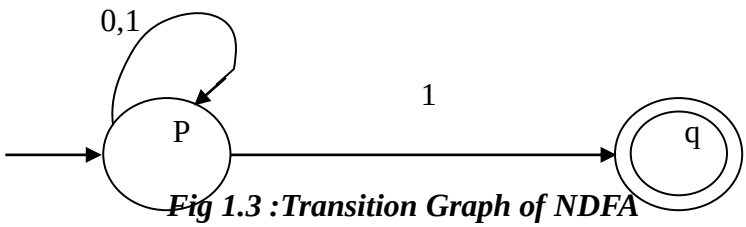

Fig 1.3 :Transition Graph of NDFA

1.7 Equivalence of DFA and NDFA :

Although the DFA and NFA have distinct definitions, a NFA can be translated to equivalent DFA using the subset construction algorithm . i.e., the constructed DFA and the NFA recognize the same formal language. Both types of automata recognize only regular

languages.

Therefor every language that can be described by some NDFA can also be described by some DFA..

Theorem
Let M = (Q, Σ, δ, q_0, F) be a non-deterministic finite automaton. There exists a deterministic finite automaton M', such that L(M') =
L(M).
i.e. for every NDFA there exists a DFA which simulates the behavior of NDFA . Hence if language L is accepted by NDFA , then there exist a DFA M' which also accept L. where M' = (Q', Σ, δ', q'_0, F')

1.8 Mealy and Moore Machine

These machines are modeled to show transition and output symbol. These machine do not define a language by accepting or rejecting input string , so there is no existence of final state.

1.8.1 Moore Machine

a Moore machine is a finite-state machine whose output values are determined solely by its current state.

$$z = \lambda (q)$$

Where z : out put
λ : output function
q : state

1.8.2 Formal definition of Moore Machine

A Moore machine can be defined as a 6-tuple (Q, Σ, Δ , δ, λ , q_0) consisting of the following:
Where

- Q : a finite set of states
- q_0 : a start state (also called initial state) which is an element of Q ($q_0 \in Q$)
- Σ : a finite set called the input alphabet
- Δ : a finite set called the output alphabet
- δ : a transition function ($Q \times \Sigma \rightarrow Q$) mapping a state and the input alphabet to the next state
- λ : an output function ($Q \rightarrow \Delta$) mapping each state to the output alphabet

A Moore machine can be regarded as a restricted type of finite state transducer.

1.8.3 Mealy Machine

a Mealy machine is a finite-state machine whose output values are determined both by its current state and the current inputs.

$$z = \lambda (q, x)$$

Where z : out put

λ : output function
q : state
x : input symbol

1.8.4 Formal definition of Mealy Machine

A Mealy machine can be defined as a 6-tuple (Q, Σ, Δ, δ, λ, q_0) consisting of the following:
Where

- Q : a finite set of states
- q_0 : a start state (also called initial state) which is an element of Q ($q_0 \in Q$)
- Σ : a finite set called the input alphabet
- Δ : a finite set called the output alphabet
- δ : a transition function ($Q \times \Sigma \rightarrow Q$) mapping a state and the input alphabet to the next state
- λ : an output function ($Q \times \Sigma \rightarrow \Delta$) mapping each state and input to the output alphabet.

1.8.5 Difference between Mealy and Moore Machine.

I. Mealy machine determines output by its current state and the current inputs, while Moore machine determines output by its current state only.
II. Mealy machine produces same numbers of characters in output string as no. of characters in input in string, while output of Moore machine has one more characters as that of in input string

1.9 Two way finite automata

The finite automata discussed so far has a δ transition which indicate where to next from current state on receiving particular input. But 2-way FA is a model in which linear direction is mentioned on receiving particular and being in some current state. There are two direction that are allowed in 2-FA and these are left and right direction as shown in fig. 1.4

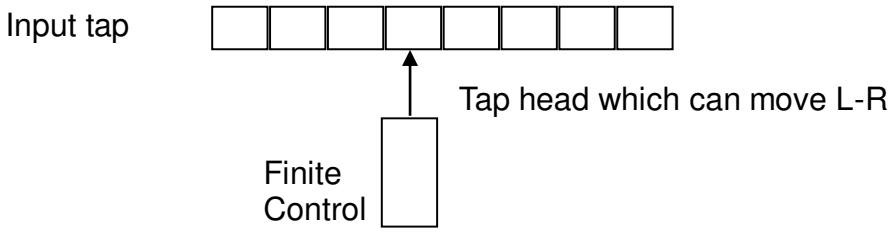

Fig 1.4 :Two way finite automata

1.9.1 Formal Definition of Two way finite automata

It is a collection of 5-tuple M = (Q, Σ, δ, q_0, F), where

Q : a finite set of states
Σ : a finite set called the input alphabet
δ : a transition function which maps Q x {L,R}
q_0 : a start state (also called initial state) which is an element of Q ($q_0 \in Q$)

F : Set of final states.

1.10 The centre concept o automata theory.

a. Alphabets : It is finite non-empty set of sysmbol represeted by Σ.

$(Q, \Sigma, \delta, q_0, F)$ $\Sigma = \{0,1\}$

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

b. String or word: It a finit sequence of sysmbols chosen from some alphabet. In other words , A string over alphabet Σ is a init sequence of sysmbols from Σ.

Example : If { 0,1} be alphabet

Then string is 00011, 1101 wtc.

Set of String Σ*

Σ* represent set of all string over Σ including empty string ∧

$\Sigma^* = \{0,1\}^* = \{ \wedge, 0,1,00,11,010,101, ...\}$

Set of String Σ^+ without {∧}

Also $\Sigma^+ = \Sigma^* - \{\wedge\}$

C. Language : Language is a collection of appropriate strings. It is a subset of Σ*.

$L \subseteq \Sigma^*$ indicates L is language over alphabet Σ.

d. Operations on string

I. Concatenation : Two strings are combined together to form a single string.

II. Transpose : This operation reverse the string

If S = ababbaa

Then transpose of S, S^T = aabbaba .

1.11 Acceptability of string by FA

A string ω is accepted by a finite automaton M= $(Q, \Sigma, \delta, q_0, F)$ if $\delta(q_0, \omega) = q$ for some $q \in F$. This is basically the acceptability of a string by the final state.

1.12 Regular Expression(RE) :

DFA and NDFA are machine like description of langiage whereas Regular expression is algebraic description of language. RE offer a way to expresss the string we want to express/accept.

Example :

Consider a set of string in which every string has *aab* as substring followed and preceded by any number of a's and b's like {a*aab*, aabb*aab*bb, ----} there will be infinite number of such strings. This set of string can be represented in very convient way by regular expression as follows :

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

Therefor RE is very conveniet way to represent a pattern of string.

1.13 Regular operations

In this section, we define three operations on languages. Later, we will answer the question whether the set of all regular languages is closed under these operations. Let X and Y be two languages over the same alphabet.

1. Union : The union of X and Y is defined as

$X \cup Y = \{w : w \in X \text{ or } w \in Y\}$.

2. Concatenation : Concatenation of X and Y is defined as

$XY = \{ww' : w \in X \text{ and } w' \in Y\}$.

In words, XY is the set of all strings obtained by taking an arbitrary string w in X and an arbitrary string w′ in Y , and gluing them together (such that w is to the left of w′).

3. **Closure or star:** Closure or star of X is defined as

$X^* = \{u1\ u2 \ldots uk : k \geq 0 \text{ and } ui \in X \text{ for all } i = 1, 2, \ldots, k\}$.

In words, X^* is obtained by taking any finite number of strings in X, and gluing them together. Observe that k = 0 is allowed; this

corresponds to the empty string ε . Thus, $\varepsilon \in X^*$.

Example, let X = {001, 10, 111} and Y = {ε , 001,101}. Then

$X \cup Y = \{001, 10, 111, \varepsilon, 101\}$,

and

XY = {001, 001001, 001101, 10, 10001,10101, 111,111001,111101 ,

$X^* = \{\varepsilon, 001, 10, 111, 00110, 001111, 00110111, 111001.....\}$.

1.14 Operators for RE:

1. Union (+) : If E and F are RE than E + F is RE denoting the union of L(E) and L(F)

$L(E+F) = L(E) \cup L(F)$

2. Concatenation : If E and F are RE than EF is RE denoting the concatenation of L(E) and L(F).

$L(EF) = L(E)\ L(F)$

3. Closure or star Star : If E is RE than E* is RE denoting the closure of L(E) .

$L(E^*) = (L(E))^*$

1.15 Regular Set : Set represented by a RE is called regular Set.

{a,b} represented by RE **a + b** then L(a+b) is regular set.

1.16 Pumping lemma:

As we know that the class of regular languages is closed under various operations, and that these languages can be described by (deterministic or nondeterministic) finite automata and regular expressions. These properties helped in developing techniques for showing that a language is regular. Pumping lemma is tool that can be used to prove that certain languages

are not regular. Observe that for a regular language,

1. the amount of memory that is needed to determine whether or not a given string is the language is finite and independent of the length of the string, and

2. if the language consists of an infinite number of strings, then this language should contain infinite subsets having a fairly repetitive struc-ture.

Intuitively, languages that do not follow 1. or 2. should be nonregular. For example, consider the language

$$\{0^n 1^n : n \geq 0\}.$$

This language should be nonregular, because it seems unlikely that a DFA can remember how many 0s it has seen when it has reached the border between the 0s and the 1s. Similarly the language

$$\{0^n : n \text{ is a prime number}\}$$

should be nonregular, because the prime numbers do not seem to have any repetitive structure that can be used by a DFA. To be more rigorous about this, we will establish a property that all regular languages must possess.

This property is called the **pumping lemma**. If a language does not have this property, then it must be nonregular. The pumping lemma states that any sufficiently long string in a regular language can be pumped, i.e., there is a section in that string that can be repeated any number of times, so that the resulting strings are all in the language.

Theorem :**(Pumping Lemma for Regular Language):**

If L is A RL which can be described by FA of n states, then for every string ω in L , such that $|\omega| \geq n$, we can break ω into three strings x, y and z such that

1. $y \neq \varepsilon$
2. $|xy| \leq n$
3. and $xy^i z \in L$ for all $i \geq 0$.

1.17 Closure properties of Regular Language(RL)

If certain languages are regular then language formed by certain operations is also regular. These is called Closure properties of Regular Language(RL).

1. The set of regular languages is closed under the union operation, i.e., if A1 and A2 are regular languages over the same alphabet Σ, then $A1 \cup A2$ is also a regular language.

2. The set of regular languages is closed under the concatenation operation, i.e., if A1 and A2 are regular languages over the same alphabet Σ, then A1 A2 is also a regular language.

3. The set of regular languages is closed under the star operation, i.e., if A is a regular language, then A* is also a regular language.

4. The set of regular languages is closed under the complement operation. i.e.Complement of RL is Regular.

5. The set of regular languages is closed under the difference operation.i.e Deiffernece of two RL is regular.

6. The set of regular languages is closed under the reversal operation i.e. Reversal of a RL is r e g u l a

RGPVNOTES.IN

We hope you find these notes useful.

You can get previous year question papers at https://qp.rgpvnotes.in .

If you have any queries or you want to submit your study notes please write us at rgpvnotes.in@gmail.com

LIKE & FOLLOW US ON FACEBOOK
facebook.com/rgpvnotes.in