M.Sc-IT (OLD) Examination,2013

Theory of Computation

Paper: Fourth

1.a. W = aaabbb

b. L={ $a^n b^n$, n>=0}

---

ii) The set which is expressed by regular expression is regular set. Ex: {null, ab} , { a, b} etc.

---

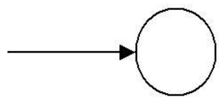
iii) Non distinguishable state: Two states are said to be non distinguishable states if upon the application of same input to the two states they yield same state as output.

---

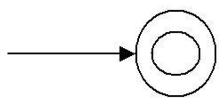
iv) ***Transition diagrams and Transition Systems***

A transition graph or a transition system is a finite directed labeled graph in which each vertex (node) represents a state and the directed edges indicate the transition of a state and the edges are labelled with input. A transition graph contains:

(i) A *finite set of states*, one of which are designated as start state and some of which are designated as final states.



Start state



Final state

(ii) An *alphabet* □ of possible input letters from which input strings are formed.

(iii) A *finite set of transitions* that show, how to go from some states to some other states.

So a transition system is a 5-tuple $(Q,\sum,\delta,q0,F)$

If $\delta(qi,a)=qj$, there is an edge labeled by 'a' from qi to qj . A transition system accepts a string 'w' in $\sum^*$ if

<u>Transition Table</u>

The description of the automation can be given in the form of transition table also, in which we tabulate the details of the transitions defined be the automaton from one state to another.

----

v) **<u>Mealy machine</u>** Output function maps $\sum$ X Q into delta

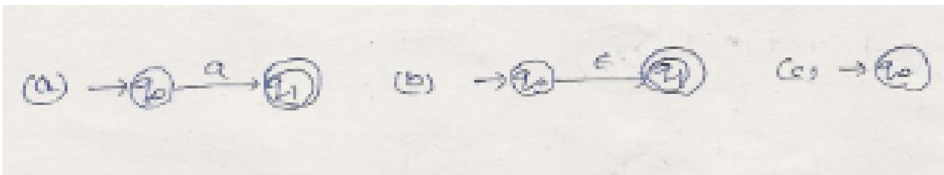
$Z(t) = \lambda\ (q(t),x(t))$

Moore machine Output function maps Q into delta

$Z(t) = \lambda\,(q(t))$

---

vi)



---

vii) A string x is accepted by a finite automaton

$M = (Q, \Sigma, \delta, q0, F)$

If $\delta(q0,x) = q$ for some $q \in F$

---

viii) viii) S->aAB

->aBbaB

->abbaB

->abbabB

->abbabc

---

ix) Context sensitive language are derived from context sensitive grammar containing production of the form

Type 1 production : A production of the form $\varphi A\psi \to \varphi\alpha\psi$ is called a type 1 production if α not equal to null Where φ is the left context ψ is the right context $A \in V_n$ and $\alpha\ (V_n \cup \Sigma)^*$

The production S-> null is also allowed in a type 1 grammar but in this case S does not appear on tht right hand side of any production.

Example:

2A->1B

B->0

X) A grammar in containing all derivation of the form A-> BC ,A->a Is said to be in CNF where A,B,C belongs to V_n and a belongs to ∑

SECTION B

2.a

S->XX (s->XX)

S->bXX (S->bX)

S->bbXX (X->bX)

S->bbaX (X->a)

S->bbaXXX (X->XXX)

S->bbaaXX (X->a)

S->bbaaaX (X->a)

S->bbaaaXb (X->Xb)

S->bbaaaab (X->a)

2.b Introduction to Turing Machine

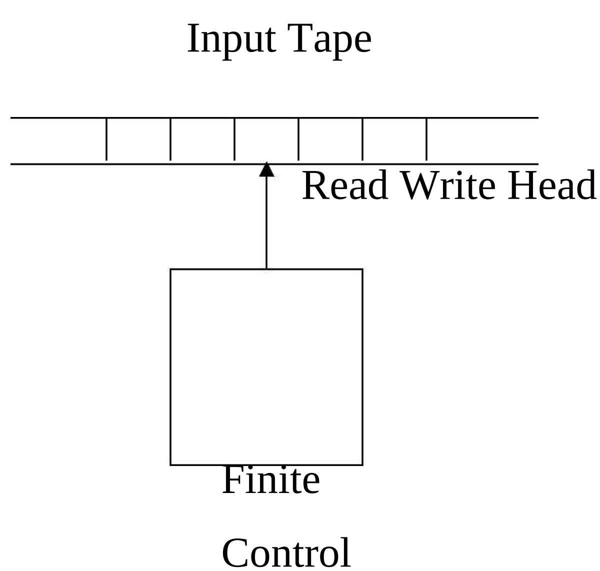
Turing Machine

Basic model of a Turing machine consists of

i) a two way infinite tape,

ii) a read/write head and

iii) a finite control.



At any time, action of a Turing machine depends on the current state and the input symbol and involves (i) change of state (ii) writing a symbol in the cell scanned (iii) head movement to the left or right and (iv) Turing machine halts or not halts. A Turing machine may utilize the tape cells beyond the input limits and ‘Blank’ cell plays a significant role in the working of a Turing machine. Turing machine halts in any situation for which a transition is not defined. Unlike the previously dealt automata, it is possible that a Turing machine may not halt. At any state a Turing machine can halt or not halt. ie, it ends in accepting state if it successfully halts(accept halt). Otherwise it halts in any non accepting state (reject halt).

A turing machine M is a 7-tuple namely (Q, ∑, ┌,δ,q0,b,F)

Where

Q is a finite nonempty set of states

┌ is a finite nonempty set of tape symbol

B ϵ ┌ is the blank

∑ is a nonempty set of input symbols and is a subset of ┌ and b does not belongs to ∑

Δ is the transition function mapping (q,x) onto (q’,y,D)

Q0 ϵ Q is the initial state

F is a subset or equal to Q

Left move :

Suppose δ (q,xi) =(p,y,L)

Id before processing

X1,x2…………………….xi-1 q xi ………………..xn

After processing

X1,……………….xi-2 p xi-1 y xi+1……………xn

Right move:

Suppose δ (q,xi) =(p,y,R)

Id before processing

X1,x2…………………….xi-1 q xi ………………..xn

After processing

X1,……………….xi-2 xi-1 y p xi+1……………xn

~~means that α is written in the current cell, β gives the movement of the head (L/R), and γ~~

denotes the new state into which Turing machine enters.

Eg:

Present state	Tape symbols		
	0	**1**	**b**
q1	0Rq1		1Lq2
q2	0Lq2	1Lq2	bRq3
q3	bRq4	bRq5	
q4	0Rq4	1Rq4	0Rq5
*q5			0Lq2

(iii) *transition diagram*

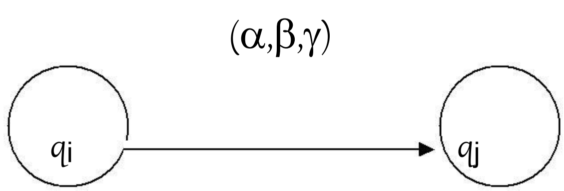
In the transition diagram the labels are triples of the form (α,β,γ) *where* α,β $\in \Gamma$ *and* γ

$\in$ {L,R}. When there is a directed edge from q_i to q_j with label (α,β,γ), it means that $\delta(q_i, \alpha)=(q_j, \beta, \gamma)$.

During the processing of an input string, suppose the Turing machine enters q_i and R/W head scans the present symbol α. As a result, the symbol β is written in the cell

under R/W head. The R/W head moves to the left or right, depending on γ, and the new state is q_j.

le,



eg:

Design a Turing machine to recognize all strings consisting of even number of 1's.

<u>Solution</u>: (i) q_1 is the initial state. M enters state q_2 on scanning 1 and writes b.

1) If M is in state q_2 and scans 1, it enters q_1 and writes b.

q_1 is the only accepting state.

So M accepts a string if it exhausts all input symbols and finally in state q_1. Symbolically,

$M=(\{q_1,q_2\},\{1\},\{1,b\}, \delta, q_1,b,\{q_1\})$ Where δ is

defined by

Present state	Input symbols	
	1	**B**
$*q_1$	XRq_2	BLq_1
q_2	XRq_1	

3.a

W0 = { A, X} as A-> a and X-> ad

W1= {A,X,S} as S->bX

W2= { A,X,S} as A-> bSX

Phase II

S->bX

X->ad

3b. S->aSb

S-> null

4a. Types of grammar:

A type 0 grammar is any phase structure grammar without any restriction

A->a

A grammar is called type 1 or context dependent if all its production are type 1 productions. The production S-> null is also allowed in a type 1 grammar but in this case S does not appear on tht right hand side of any production.

Type 1 production : A production of the form $\varphi A\psi \to \varphi\alpha\psi$ is called a type 1 production if α not eqal to null

2A->1B

B->0

A grammar is called a type 2 grammar if it contains only type 2 productions .It is also called a context free grammar A language generated by a context free grammar is called a type 2 language or a context free language

A type 2 production is a production of the form A->α where A € V_n and α $(V_n \cup \sum)^*$

Example S-> Aa, A->a

A grammar is called a type 3 grammar if it contains only type 3 productions . The production S-> null is also allowed in a type 1 grammar but in this case S does not appear on tht right hand side of any production.

A type 3 production is a production of the form A->a or A-> aB where A , B € V_n and a ϵ∑

Example B-> aC, A->a

----

4 b. i) $(a+b)^*a$

ii) $bb(bbb)^*$

5.a E-> T X

X-> +T X | null

T-> T Y

Y-> *F Y | null

5.b Q0---1-----Q1----1----Q2----1----Q3----0----Q4----1-----Q5*

-

-------0----Q6----0-----Q7---1----Q8*

6.a)

Moore machine

A Moore machine M is a six-tuple namely (Q, ∑,delta,δ,λ,q0)

Where

Q is a finite nonempty set of states

∑ is a nonempty set of input symbols

Δ is the output alphabet

Δ is the transition function mapping ∑ X into Q

λ is the output function mapping Q into Q

Q0 ϵ Q is the initial state

A mealy machine is a six tuple (Q, ∑,delta,δ,λ,q0) where all the symbol except λ have the same meaning as in the Moore machine λ is the output function mapping ∑ X Q into delta

In practice mixed models are often used.

Mealy machine Output function maps $\sum$ X Q into delta

$Z(t) = \lambda (q(t),x(t))$

Moore machine Output function maps Q into delta

$Z(t) = \lambda (q(t))$

Moore Machine

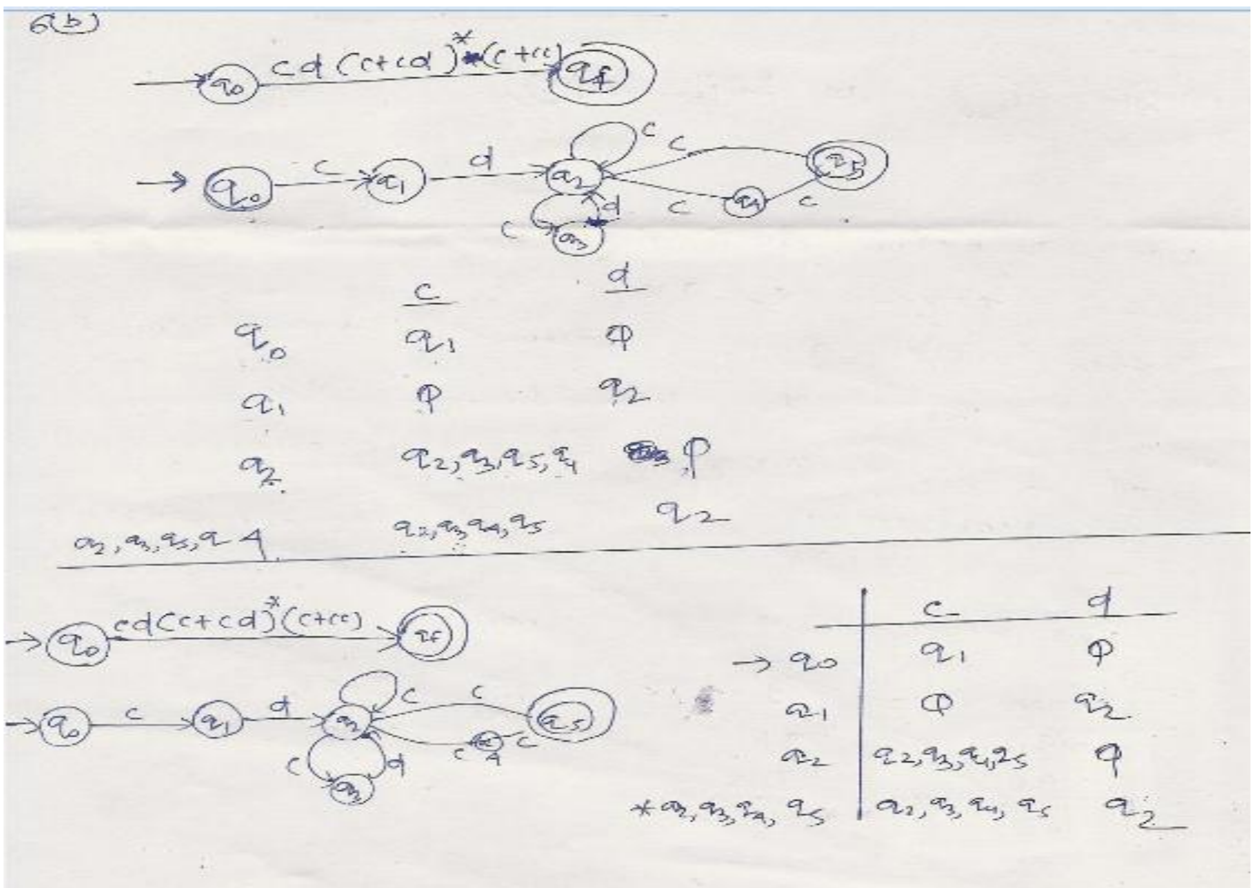
Present state	Next state a=0	Next state a=1	Output
->q1	q1	q2	0
Q2	q1	q3	0
Q3	q1	q3	1

Mealy Machine

Present state	Next state			
	a=0	out1	a=1	out2

->q1	q1	0	q2	0
Q2	q1	0	q3	0
Q3	q1	0	q3	1

6.b



7a. S-> 0B

->OOBB

->OO1B

->0011SS

->00110B

->001101S

->0011010B

->00110101

S-> 0B

->00BB

->00B1S

->00B10B

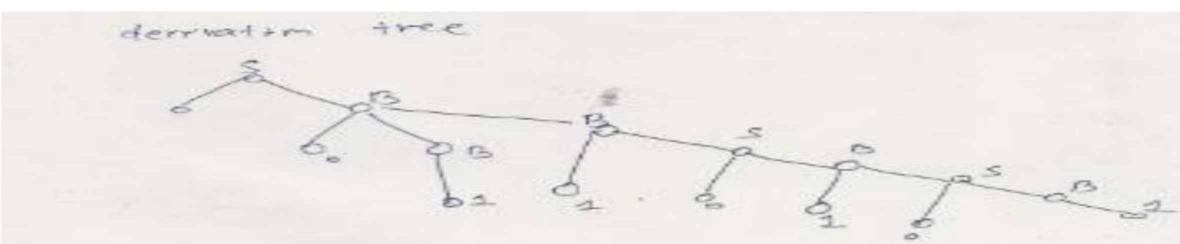
->00B101S

->00B1010B

->00B10101

->00110101

Derivation tree:



Ambiguous grammar:

A context free grammar G isambigious if there exists some w ϵ L(G) which is ambiguous

Example G= ({ S},{a,b,+,*},P,S) Where P consists of

S-> S+S | S*S | a | b

We have two derivation trees for a+a*b

S-> S+S->a+S->a+S*S->a+a*S->a+a*b

S->S*S->S+S*S->a+S*S->a+a*S->a+a*b

8.a Example 1: Construct a PDA that accepts the language

$\{a^n b^n \mid n \geq 0\}$.

$M = (Q, \Sigma, \Gamma, \delta, q_1, Z, F)$

$Q = \{q_1, q_2, q_3, q_4\}$

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

$\Gamma = \{a, b, z\}$

$F = \{q_1, q_4\}$, and δ consists of the following transitions

1. $\delta(q_1, a, z) = \{(q_2, az)\}$
2. $\delta(q_2, a, a) = \{(q_2, aa)\}$
3. $\delta(q_2, b, a) = \{(q_3, \in)\}$

4. $\delta(q_3, b, a) = \{(q_3, \in)\}$
5. $\delta(q_3, \in, z) = \{(q_4, z)\}$

8.b Step-1 : Find all the edges starting from v2

Step-2: Duplicate all these edges starting from v1 without changing the edge label

Step-3 If v1 is an initial state, make v2 also as initial state

Step-Iv If v2 is a final state make v1 as the final state.