M.C.A (Third Semester) Examination,2013

Theory of Computation

Paper: Fourth

---

1. i)

   a) uvu= aab bbbab aab

   b) null followed by u = aab

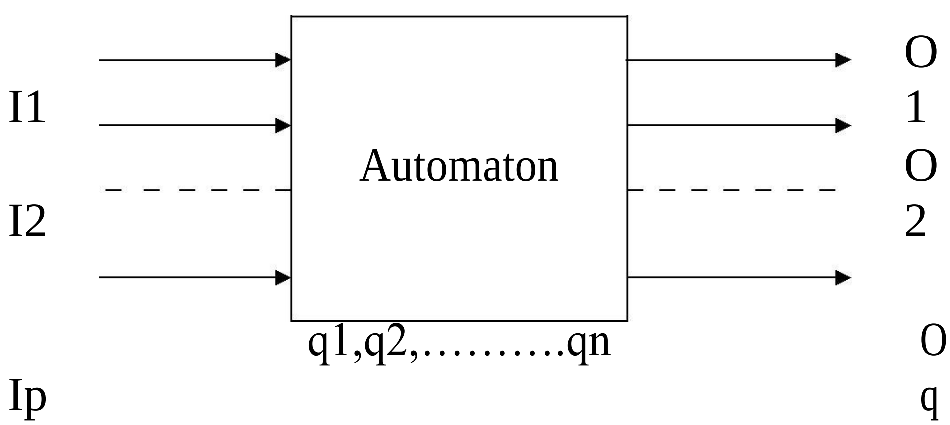
   c) u followed by null = aab

   d) u followed by null followed by v = aab bbbab

---

1. ii) Description of Automaton

   An automaton can be defined in an abstract way by the following figure.



Model of a discrete automaton

i) Input: - At each of the discrete instants of time t1,t2,…..input values I1,I2……… each of which can take a finite number of fixed values from the input alphabet ∑, are applied to the input side of the model.

ii) Output : - O1,O2….are the outputs of the model, each of which can take finite numbers of fixed values from an output O.

iii) States : - At any instant of time the automaton can be in one of the states q1,q2…..qn

iv) State relation : - The next state of an automaton at any instant of time is determined by the present state and the present input. ie, by the transition function.

v) Output relation : - Output is related to either state only or both the input and the state. It should be noted that at any instant of time the automaton is in some state. On 'reading' an input symbol, the automaton moves to a next state which is given by the state relation.

---

1. iii) Reduced grammar: A grammar which does not contain any useless symbols or productions that will never be used in any derivation process are reduced grammars.

   Ex- S->AB | a

B->b can be reduced to S->a as S->AB and B-> b will never be used in the derivation process.

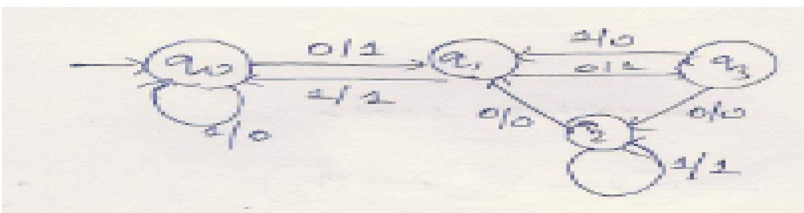
----

1. iv) $\delta(Q, \text{Input}, A) = \{ (Q,\alpha) \mid A \to \alpha \text{ is in } P\}$
   $\delta(Q,a,a) = \{(Q, \text{null})\}$ for every a in $\sum$

Where Q states, $A \in V_N$ $\alpha \in (V \cup \sum)^*$ and $a \in \sum$

----

1. V.



----

1. Vi, The pumping lemma is very useful to prove whether certain sets are regular or not. To perform this the steps used are

1) assume L is regular. Let n be the number of states in the corresponding FA
2) Choose the string W such that |W| >=n. use pumping lemma to write W = xyz
   With conditions |xy| <=n and |y| >0
3) Find a suitable integer such that xy^iZ does not belongs to L. this contradicts our assumption . hence L is not regular.

Vii) In bottom up parsing parsing takes place from the terminal nodes to the root node. In bottom-up parsing the derivation tree is traversed from the given input string to the start of the grammar symbolEx: S-> AB

A-> a
B-> b

For string ab

=aB (Using transition rule B->b)
=AB (Using transition rule A->a)
=S (Using transition rule S->AB

In Top down parsing takes place from the root nodes to the terminal nodes. In top-down parsing the derivation tree is traversed from the start symbol of the grammar to the terminal nodes. of the grammar symbol.

Ex: S-> AB
A-> a
B-> b

S-> AB (Using transition rule S->AB
S->aB (Using transition rule B->b)
S->ab (Using transition rule A->a)

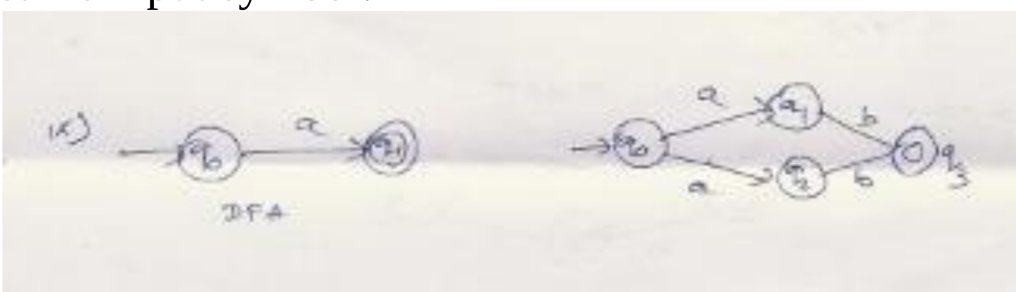
----

1. viii) abba + bbba + abab

----

1. ix) In deterministic finite automation no input symbol causes to move more than one state or state does not contain more than one transition from same input symbol.

In NFA one input sumbol causes to move more than one state or a state may contain more than one transition from same input symbol.



DFA NFA

----

1. x) Languages generated from regular grammar are regular languages. For example

S->aSa | b

So L = { $a^n b a^n$}={aba,aabaa,aaabaaa,aaaabaaaa,…………..}

SECTION-B

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

ii) $bb(bbb)^*$

b)

remainder set consists of { 0,1,2,3,4} -> { q0,q1,q2,q3,q4}

(q0,0) -> 2X0+0= 0->q0

(q0,1) -> 2X0+1= 1->q1

(q1,0) -> 2X1+0= 0->q2

(q1,1) -> 2X1+0= 1->q3

(q2,0) -> 2X2+0= 0->q4

(q2,1) -> 2X2+0= 1->q0

(q3,0) -> 2X3+0= 0->q1

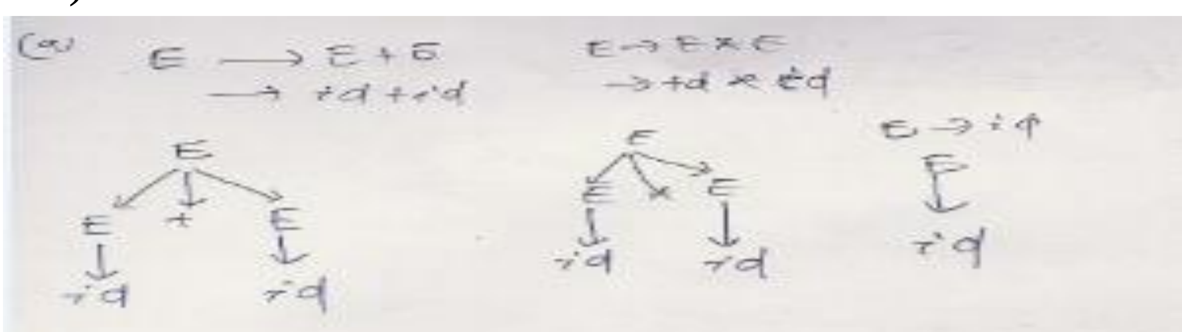
(q3,1) -> 2X3+0= 1->q2

(q4,0) -> 2X4+0= 0->q3

(q4,1) -> 2X4+0= 1->q4

---

3 .a)

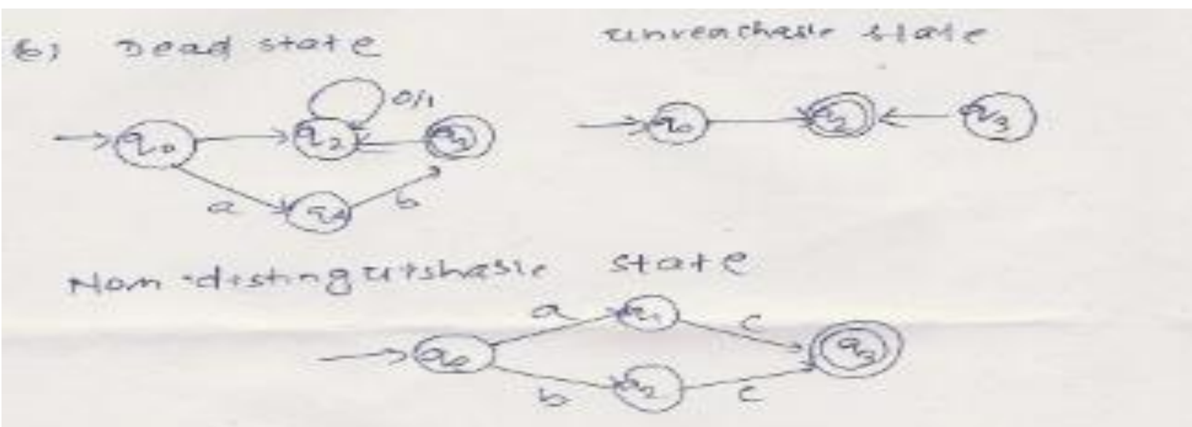


---

3. b) Dead state : The state which is not a final state and ends in itself upon the application of any input signals.

Unreachable state: Unreachable states are the states which are not reachable from the initial states upon the application of any input sequence.

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.



---

4. a) A context free grammar G such that some word has two parse trees is said to be ambiguous. A grammar which generates two or more parse tree for the same grammar.

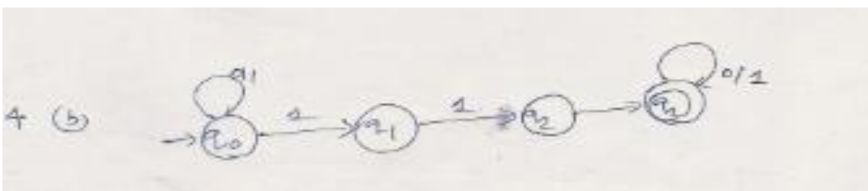
The given grammar is ambiguous because for the same string abaa it produces two derivation tree by using the derivations

S->SbS->abS->abSa->abaa   S->Sa->SbSa->Sbaa->abaa

The parse tree of the above derivations are different . Thus the language is ambiguous.

a) DFA accepting 111

1



	0	1
Q0	Q0	Q0,Q1
Q1	null	Q2
Q2	null	Q3
*Q3	Q3	Q3

	0	1
Q0	Q0	Q0,Q1
Q0,Q1	Q0	Q0,Q1,Q2
Q0,Q1,Q2	Q0	Q0,Q1,Q2,Q3
*Q0,Q1,Q2,Q3	Q0,Q3	Q0,Q1,Q2,Q3
*Q0,Q3	Q0,Q3	Q0,Q1,Q3

*Q0,Q1,Q3	Q0,Q3	Q0.Q1,Q2,Q3

	0	1
*Q0	Q0	Q0,Q1
*Q0,Q1	Q0	Q0,Q1,Q2
*Q0,Q1,Q2	Q0	Q0,Q1,Q2,Q3
Q0,Q1,Q2,Q3	Q0,Q3	Q0,Q1,Q2,Q3
Q0,Q3	Q0,Q3	Q0,Q1,Q3
Q0,Q1,Q3	Q0,Q3	Q0.Q1,Q2,Q3

---

5. a)Phase1: W0= { B, C} as B-> null , C-> null
W1= { A, B,C, S} as A-> BC and S-> ABaC

Phase II S-> ABaC | BaC | ABa | A aC | Aa | Ba | ac | a
As A-> BC and B and C has multiple option
So A-> B | C | BC
D->d is included

5.b

A PDA is defined as 7 tuple notation
M=(Q,∑, ┌, δ,q0,z0,F)

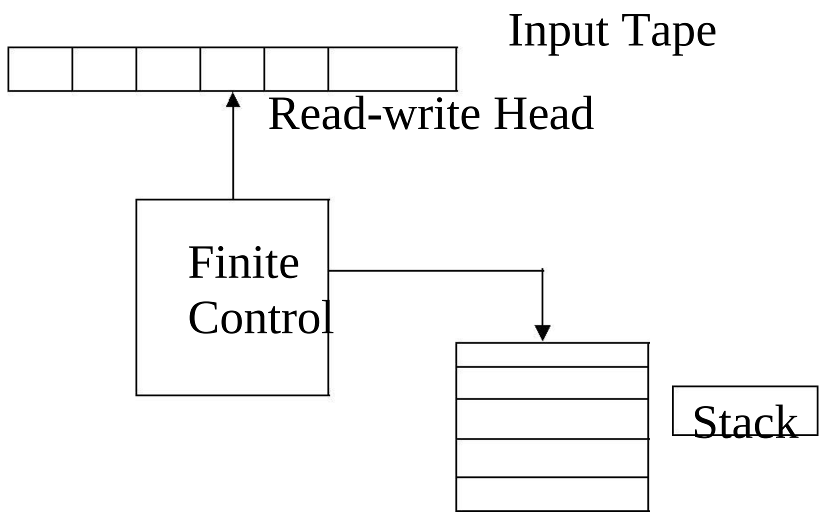
Where Q= finite set of states
∑ = Input alphabet
┌ = is an alphabet called the stack
Q0= is the initial state q0 € Q

F is set of final states F subset / equal to Q
δ is a transition mapping $\delta = QX (\sum U \{ €\}) X \Gamma \to Q X \Gamma^*$

Basic model of PDA consists of 3 components:

i) an infinite tape
ii) a finite control
iii) a stack

Now let us consider the 'concept of PDA' and the way it 'operates'.



PDA has a read only input tape, an input alphabet, a finite state control , a set if initial states, and an initial state . in addition it has a stack called the pushdown stack. It is a read-write pushdown store as we add elements to PDS or remove element from PDS. A finite automation is on some state and on reading, an input symbol moves to a new state. The push down automaton is also in some state and on reading an input symbol , the topmost symbol ,it moves to a new state and writes a string of symbols in PDS.

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. $\delta(q_3, \in, z) = \{(q_4, z)\}$

$(q_1, aabb, z) \vdash (q_2, abb, az)$ (using transition 1)

$\vdash (q_2, bb, aaz)$ (using transition 2)

$\vdash (q_3, b, az)$ (using transition 3)

$\vdash (q_3, \in, z)$ (using transition 4)

$\vdash (q_4, \in, z)$ (using transition 5)

q_4 is final state. Hence ,accept. So the string *aabb* is rightly accepted by *M*.

6.a) Phase 1: W0= { A, S}
As A->b and S->a

Phase II: S->a

---

b) d(q0,B)= (q0,B,R)
d(q0,a)= (q1,B,R)
d(q1,a)= (q1,a,R)
d(q1,b)= (q1,b,R)
d(q1,B)= (q1,B,L)
d(q1,b)= (q2,B,L)
d(q2,b)= (q3,B,L)
d(q3,b)= (q4,B,L)
d(q4,b)= (q4,b,L)
d(q4,a)= (q4,b,L)
d(q4,B)= (q0,B,R)
d(q0,null)= qf

---

7 a) S->a AS

S->abSS (as A->bS)

S -> abaS (as S->a)

S->abaaAS (as S->aAS)

S-> abaabSS (as A->bS)

S-> abaabaS (as S->a)

S->abaabaa (as S->a)

7.b

of the length of the input string. The models can be described formally by the following set format:

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

All the symbols have the same meaning as in the basic model of Turing machines with the difference that the input alphabet Σ contains two special symbols ¢ and \$. ¢ is called the left-end marker which is entered in the left-most cell of the input tape and prevents the R/W head from getting off the left end of the tape. \$ is called the right-end marker which is entered in the right-most cell of the input tape and prevents the R/W head from getting off the right end of the tape. Both the endmarkers should not appear on any other cell within the input tape, and the R/W head should not print any other symbol over both the endmarkers.

Let us consider the input string w with $|w| = n - 2$. The input string w can be recognized by an LBA if it can also be recognized by a Turing machine using no more than kn cells of input tape, where k is a constant specified in the description of LBA. The value of k does not depend on the input string but is purely a property of the machine. Whenever we process any string in LBA, we shall assume that the input string is enclosed within the endmarkers ¢ and \$. The above model of LBA can be represented by the block diagram of Fig. 9.11. There are two tapes: one is called the input tape, and the other, working tape. On the input tape the head never prints and never moves to the left. On the working tape the head can modify the contents in any way, without any restriction.

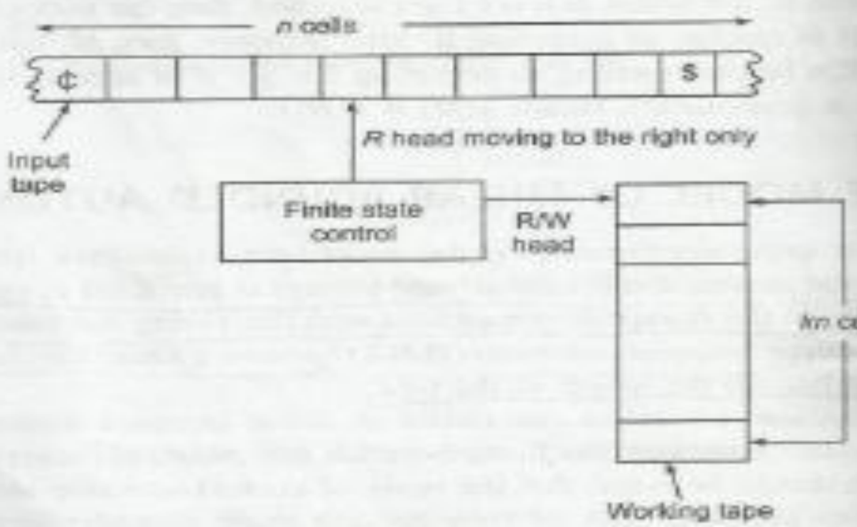


Fig. 9.11 Model of linear bounded automaton.

In the case of LBA, an ID is denoted by (q, w, k), where $q \in Q$, $w \in \Gamma$ and k is some integer between 1 and n. The transition of IDs is similar except

----

8.a.

i) a or b followed by b or c over the alphabet a,b,c

ii) all binary number precced by 1

iii) String accepting only zero

iv) Set of string over 0 and 1 where 0 or 1 is followed by zero or any numbers of 11

8.a

i) a or b followed by b or c over the alphabet a,b,c

ii) all binary number precced by 1

iii) String accepting only zero

iv) Set of string over 0 and 1 where 0 or 1 is followed by zero or any numbers of

----

8.b Context free Language: Languages formed from contex free grammar is contex fre e language

Definition(Contex free grammar)

A CFG can be defined as G=(V,T,P,S) where V is the set of non terminals, T is the set of terminals, S is the start symbol and P is the set of productions of the form A->α where A belongs to V_N, α∈(VUT)*.

Derivation tree (Parse tree)

The derivation in a CFG can be represented by using trees called 'derivation tree' or 'parse tree'. A derivation tree for a CFG is a tree satisfying the following :

i) every vertex has a label which is a variable (non terminal) or terminal.

ii) The root has label which is non terminal

iii) The label of an internal vertex is a variable.

ie, a derivation tree is a labeled tree in which each internal node is labeled by a non terminal and leaves are labeled by terminals. Strings formed by labels of the leaves traversed from left to right is called the 'yield of the parse tree'. Ie, the yield of a derivation tree is the concatenation of the labels of the leaves without repetition in the left-to-right ordering.

Eg: Let G=({S,A},{a,b},P,S)
where P is defined as S-> aAS/a,
A->b

S->aAS->aaASS->aabaa

A language is called type 1 or context dependent if its grammar contains all its production as 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 $\varphi A\psi \rightarrow \varphi\alpha\psi$ is called a type 1 production if α not eqal to null

2A->1B

B->0

A Language is called a context free Language if its grammar contains all type 2 production language

A type 2 production is a production of the form $A \rightarrow \alpha$ where $A \in V_n$ and α $(V_n \cup \sum)^*$

Example S-> Aa, A->a