

## B.E.

### Fourth Semester Examination, May-2006 THEORY OF AUTOMATA AND COMPUTATION

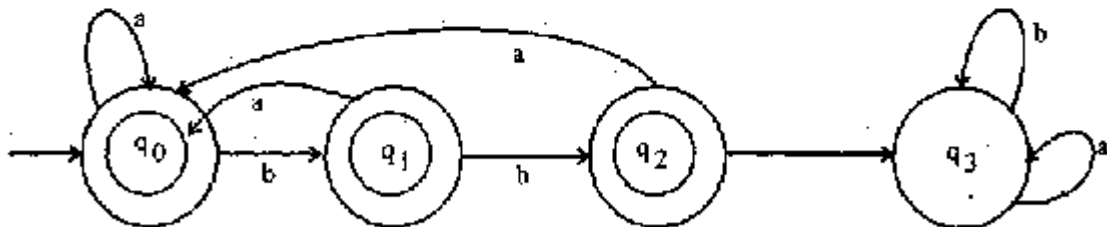
Note : Attempt any five questions. Part of the question must be attempted together.

Q. 1. (a) Construct FA equivalent to following regular expression :

$$(0+1)^*(00+11)(0+1)^*$$

Aus.

q	$\sigma$	$\delta(q, \sigma)$
$q_0$	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	$q_3$



The above diagram shows. State diagram :

Here a tends to zero

and b tends to one

The M is in state 0 or 1.

Here  $q_0$  is initial state & final state is  $q_3$  but not is transition diagram above. So state  $q_3$  is called dead

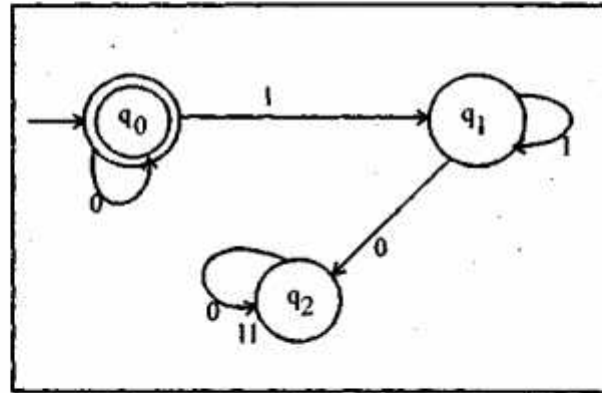
state.

**Q. 1. (b) If  $x$  and  $y$  are regular sets over  $\Sigma$ , then prove that  $x \cap y$  is also a regular set over  $\Sigma$ .**

**Ans.** If  $X$  and  $Y$  are regular set over  $\Sigma$ , then  $x \cap y$  is also regular.

So,  $T(M') = T(M)^T$ .

Here  $T(M)^T$  is regular.



The transition system  $M'$  is constructed as follows :

1. The initial state of  $M'$  are  $q_0$  and  $q_1$ .
2. The (only) final state of  $M'$  is  $q_0$ .
3. The direction of the directed edges is reversed.  $M'$  is given in above figure.

**Q. 2. What is Moore machine? How FA can be converted into Moore machine? Explain with the help of one example.**

**Ans. Moore Machine :** Moore M/C has six tuple  $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$ .

Where  $Q$  is finite set of states.

$\Sigma$  is i/p alphabet

$\Delta$  is o/p alphabet.

$\delta$  is the transition function

$\Sigma \times Q \text{ into } Q$

$\lambda$  is the o/p mapping  $Q$  into  $\Delta$   $q_0$  is initial state.

The m/c is which we remove the restriction of finite automata, and consider the model where the o/p's can

be chosen from same other alphabet. The value of o/p function  $Z(t)$  in the most general case is a function of the present state  $q(t)$  and the present i/p  $x(t)$

i.e.  $Z(t) = \lambda(q(t), X(t))$

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

This restricted model is called Moore Machine.

FA can be converted into Moore Machine, the FA, we consider here, would accept the string or do not accept the string. This acceptability was decided on the basis of reachability of the final state by the initial state.

Now one remove this restriction and consider the model which is called Moore Machine.

For example :

Present state	Next state $\delta$		o/p $\lambda$
	a = 0	a = 1	
$\rightarrow q_0$	$q_3$	$q_1$	0
$q_1$	$q_1$	$q_2$	0
$q_2$	$q_2$	$q_3$	0
$q_3$	$q_3$	$q_0$	0

For i/p string  $\overline{0111}$ , the transition of states is given by,

$$q_0 \rightarrow q_3 \rightarrow q_0 \rightarrow q_1 \rightarrow q_2$$

$$0 \text{ o/p is } 00010 \text{ so } \boxed{\text{o/p is } \lambda(q_0) = 0}$$

Q. 3. (a) Find context sensitive grammar (C.S.G.) generating the language.

$$\{SS | S \in \{a, b\}^+\}$$

Ans. Find C.S.G generating the language

$$\boxed{\{SS / S \in \{a, b\}^+\}}$$

The language generated by a type 1 grammars is called type -1 or C.S.L—Context sensitive language.

The production of form  $\phi A \psi \rightarrow \phi \alpha \psi$  is called type 1 production if  $\alpha \neq \lambda$ .

In type 1 productions erasing of A is not permitted.

If  $G = (\{S\}, \{a\}^+, \{b\}^+, \{S \rightarrow SS\}, S)$

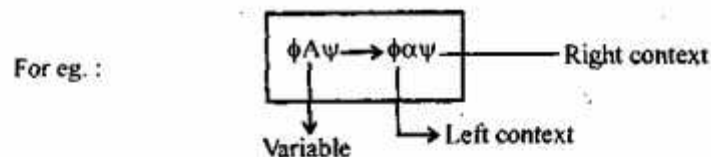
$L(G) = \phi$ , since the only production  $S \rightarrow SS$  in  $G$  has no terminal on right hand side.

**Q. 3. (b) Explain Chomsky hierarchy of Grammar. What is difference between languages of classes?**

**Ans.** Chomsky Classification of Language :

1. Type-0 grammar
2. Type-1 grammar
3. Type-2 grammar
4. Type-3 grammar

**Type 0 Grammar :** A type 0 grammar is any phase structure without any restrictions.



$\phi \alpha \psi$  replacing string.

**Type-1 Grammar :** A production of the form  $\phi A \psi \rightarrow \phi \alpha \psi$  is called a type-1 production if  $\boxed{\alpha \neq \lambda}$ .

In type 1 productions erasing of  $A$  is not permitted.

**Type-2 Grammar :** If it contains only type 2 production. It is also called a C.F.G. - Context free grammar.

**Type-3 Grammar :** A grammar is called a type 3 or regular grammar if all its productions are type 3 productions. A production  $S \rightarrow \lambda$  is allowed in type 3 grammar, but in this case  $S$  does not appear on the right hand side of any production.

**Q. 4. (a) Suppose  $M_1$  and  $M_2$  are PDA's accepting  $L_1$  and  $L_2$  respectively. Describe a procedure for constructing a PDA, accepting each of the following languages :**

- (i)  $LIUL^2$
- (ii)  $LIL^2$
- (iii)  $LI^*$ .

**Ans.** Let  $G = (V, E, R, S)$  be a context free grammar, we must construct a PDAM such that  $L(M) = L(G)$ .

The m/c the construct has only two states.

1.  $p$  and 2.  $q$  and remains permanently in state  $q$  after its first move. Also,  $M$  uses  $V$ , the set of terminals and nonterminals, as its stack alphabets.

We set

$$M = \{((p, q), \Sigma, \nu, \Delta, \rho, \{a\})\}$$

Where  $\Delta$  contain following transitions.

1.  $((P, e, e), (q, s))$
2.  $((q, e, A), (q, x))$  for each rule  $A \rightarrow x$  in  $R$ .
3.  $((q, a, a), (q, e))$  for each  $a \in \Sigma$ .

Are of above languages  $L1/L2$ ,  $L1L2$ ,  $L1^*$  are accepted by above produce used to design PDA.

**Q. 4. (b) Give transition tables for PDAs, recognizing each of the following languages :**

(i) Language of All Palindromes over  $\{a, b\}$

(ii)  $\{a^i b^j c^k \mid i, j, k \geq 0 \text{ and } j = 0 \text{ or } j = k\}$ .

**Ans. (i)**

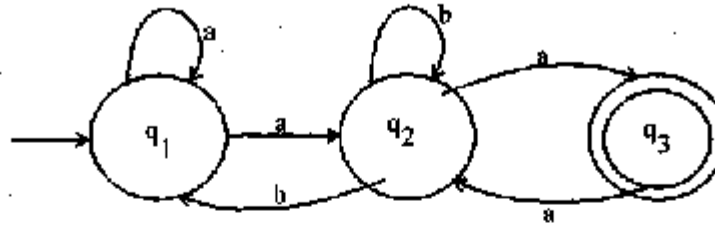
State	UnRead I/P	Stack	Transition	Comments
s	abbbabaa	c	-	Initial
q	abbbabaa	c	1	Bottom Marker
q	bbbabaa	ac	2	Starting of stack for a's
q	bbabaa	ac	7	Remove one a
q	babaa	c	5	Start stack
q	abaa	bbc	6	of or the b's.
q	baa	bc	4	
q	aa	bc	6	
q	a	c	4	
q	e	e	4	
f	e	c	8	Accepts

(ii)

State	UnRead I/P	Stack	Transition Used
s	abbcbbba	e	-
s	bcbba	a	1
s	bcbba	ba	2
s	cbba	bba	2
f	bba	ba	3
f	ba	a	5
f	e	e	4

Q. 5. (a) State and prove Myhill-Nerode theorem. How it is useful in minimization of finite Automata?

Ans. Myhill-Nerode Theorem :



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

$$\begin{aligned} q_1 &= q_{1a} + q_{2b} + \lambda \\ q_2 &= q_{1a} + a_{2b} + a_{2a} \\ q_3 &= q_{2a} \end{aligned}$$

It is necessary to reduce the No. of unknown by repeated substitution. By substituting  $q_3$  is  $q_2$  - equation one get.

$$\begin{aligned} q_2 &= q_{1a} + q_{2b} + q_{2aa} \\ &= q_{1a} + a_{2b}(b+aa)^* \\ &= a_{1a}(b+aa)^* \end{aligned}$$

Now apply Myhill-Nerode theorem we get substitution  $q_2$  is  $q_1$ , we get.

$$\begin{aligned} q_1 &= q_{1a} + q_{1a}(b+aa)^*b + \lambda \\ &= q_1(a + a(b+aa)^*b) + \lambda \end{aligned}$$

Hence by theorem, we get :

$$q_1 = \lambda(a + a(b+aa)^*b)^*$$

$$q_2 = (a + a(b+aa)^*b)^*a(b+aa)^*$$

$$q_3 = (a + a(b + aa)^* b) a(b + aa)^* a$$

Since  $q_3$  is final state, the set of strings recognized by the graph :

$$(a + a(baa)^* b)^* a(b + aa)^* a$$

**Q. 5. (b) Prove Pumping Lemma for regular sets. What are applications of pumping lemma?**

**Ans.** Let  $M = (Q, \Sigma, g, q_0, F)$  be F.A with  $n$  states. Let  $L$  be regular set accepted by  $M$ . Let  $\omega \in L$  and  $|\omega| \geq m$ . If  $m \geq n$  then there exists  $x, y, z$  such that  $\omega = xyz$   $\omega = x y^i z$   $y \neq \lambda$  and  $x y^i z \in L$  for each  $i \geq 0$ .

**Application of Pumping Lemma :** \* This theorem can be used to prove that certain sets are not regular we need some steps to prove that given set is not regular.

**Step 1 :** Assume  $L$  be Regular and  $n$  is No. of states in F.A.

**Step 2 :** Choose a string  $w$  such that  $|w| \geq n$ . Use pumping lemma to write  $\omega = xyz$  with  $|xy| \leq n$  and  $|y| > 0$ .

**Step 3 :** Find a suitable integer  $i$  such that  $x y^i z \in L$ , so  $L$  is NOT regular.

**Q. 6. (a) Design a Turing Machine to accept the language**

$$L = \{1^n 0^n / n \geq 1\}.$$

**Ans.** The following transition table used here :

Present State	Tape symbol				
	0	1	x	y	b
$\rightarrow q_1$	$xRq_2$			$bRq_5$	
$q_2$	$0Rq_2$	$yLq_3$		$yRq_2$	
$q_3$	$0Lq_4$		$xRq_5$		$yLq_3$
$q_4$	$0Lq_4$		$xRq_1$		
$q_5$				$yRq_5$	$bRq_6$
$q_6$					

Here the processing used for strings are :

(a) 011

(b) 0011

(c) 001

Now the strings accepted by M are :

(a)  $q_1 001 : X R_{q_2} 11 : q_3 x y 1 : X_{q_5} y_1 \times y_{q_5} 1$

As  $\delta(q_{5,1})$  is not defined,

So M halts, so i/p string 011 is not accepted.

Where as 0011 is accepted by M as :

$q_1 0011 : X q_2 011 : X o q_2 11 : X_{q_3} 0 y_1$

$: q_4 X 0 y_1 : x_{q_1} 0 y_1 : X X q_2 y_1 : X X y q_2 1$

are states are defined.

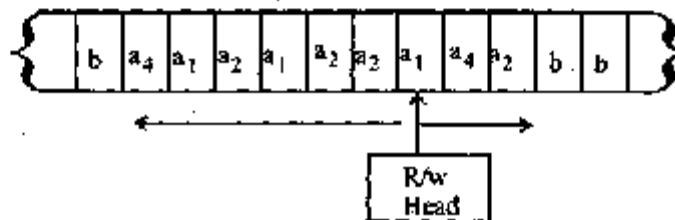
**Q. 6. (b) Design a Turing Machine to perform the addition  $p + q$ .**

**Ans.** The Turing m/c can be designed by using.

→ Instantaneous description using more relation.

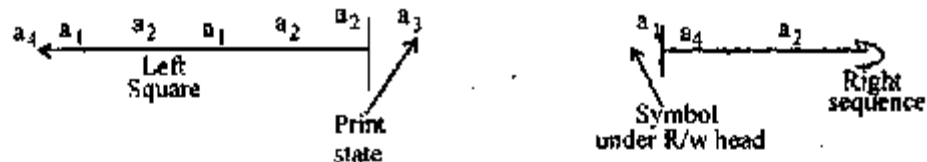
→ Transition table.

→ Transition diagram.





Present state is  $q_1$



Now transition table :

P.S.	Take symbol		
	b	0	1
$q_1$	$1Lq_2$	$0Rq_1$	
$q_2$	$bRq_3$	$0Lq_2$	$1Lq_2$
$q_3$		$0Rq_4$	$0Rq_5$
$q_4$	$0Rq_5$	$0Rq_4$	$1Rq_4$
$q_5$			

Q. 7. (a) Describe two Lemmas required to convert a grammar to Chomsky Normal form.

Ans. We used pumping lemma for regular sets.

Let  $M = (Q, \Sigma, \delta, q_0, F)$  be a F.A with  $n$  states. Let  $L$  be the regular set accepted by  $M$ .

Let  $w \in L$  and  $|w| \geq m$ .

If  $m \geq n$  then there exists  $x, y, z$  such that  $w = xyz$

$y \neq \lambda$  and  $xy^jz \in L$  for each  $j \geq 0$ .

The following steps are required for conversion :

Step 1 : Assume  $L$  is regular. Let  $n$  be the No. of states in FA.

**Step 2 :** Choose a string  $w$  such that  $|w| \leq n$ . Use pumping lemma to unite  $w = xyz$  with  $|xy| \leq n$  and  $|y| > 0$ .

**Step 3 :** Find a suitable integer  $i$  such that  $xy^iz \in L$ , Here  $L$  is not regular.

**Q. 7. (b) Describe difference between context-free and context-sensitive grammar. Also describe basic defects of context-free grammar.**

**Ans. Context Free grammar (C.F.G.) :**

A finite set of variables each of which reference a language, and they reference language as recursively is terms of each other & the primitive symbols are called terminals. The rules are called production.

The notation used to describe PLs is called Backus-naur form.

**For eg :**

1.  $\langle \text{exp} \rangle \rightarrow \langle \text{Mexp} \rangle + \langle \text{exp} \rangle$
2.  $\langle \text{exp} \rangle \rightarrow \langle \text{exp} \rangle * \langle \text{exp} \rangle$
3.  $\langle \text{exp} \rangle \rightarrow (\langle \text{exp} \rangle)$
4.  $\langle \text{exp} \rangle \rightarrow \text{id}$

**Context Sensitive Grammar (C.S.G.) :**

The grammar generated by C.S.L called context sensitive grammar.

A production of form

$$\phi A \psi \rightarrow \phi \alpha \psi$$

called context sensitive grammar if  $\alpha \neq \lambda$ .

Every Monotonic grammar  $G$  is equivalent to type I grammar or C.S.G. for eg. :

If every production is  $P$  is of form

$$\alpha \rightarrow B \text{ with}$$

$$|\alpha| \leq |B| \text{ or}$$

$$S \rightarrow \lambda.$$

**Q. 8. Write short notes on the following :**

**(a) Primitive recursive functions**

(b) Greibach Normal Form (GNF)

(c) Application of PDA.

(d) PCP Problem.

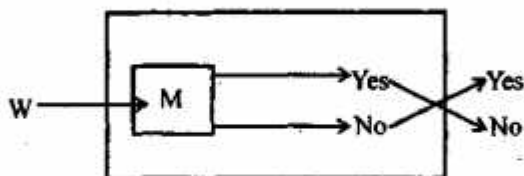
Ans. (a) Primitive recursive functions :

A No. of theorem are proved by reducing one problem to another. These recursive functions integrate many terming unless into one composite machine.

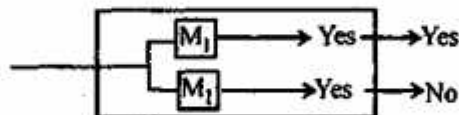
The state of the composite TM has a component for each individual m/c. Similarly the composite m/c are not tedious.

One algorithm which is given can be recursive for another, but sometime we can't used if for another problem, it is only used for one algorithm.

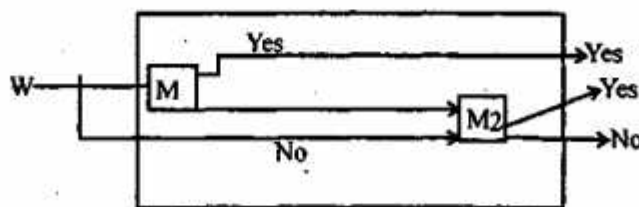
1. The Recursive function complement is also recursive, this is the properties of Recursive function.



2. The UNION of two Recursive functions is Recursive.



3. If a language L & its component I are both Recursively enumerable then L is also recursive & I also.



(b) Greibach Normal Form (GNF) :

A context for grammar  $G = (V, \Sigma, R, S)$  is called GNF, if any rule is of the form of  $A \rightarrow w$  for some  $w \in \Sigma (V - \Sigma)^*$ .

**(c) Application of PDA :**

A finite automatives can of accept the language  $L$ , where the strings are of the form  $a^n b^n$  as it has to remember the No's of a's & b's. This difficulty is reduced by adding auxiliary memory in form of stack. The a's & b's can be added into stack. First remove a's and add b's into stack.

So this type of arrangement where F.A. has a stack leads to generation of a pushdown automata (PDA).

**(d) PCP Problem :**

**Post's correspondence problem.** An instance of PCP consists of two lists  $A = w_1 \dots w_n$  and  $B = x_1 \dots x_k$  of strings over some alphabet  $\Sigma$ . This instance of PCP has a solution if there is any sequence of integers  $i_1, i_2, \dots, i_m$  with  $m \geq 1$  such that  $w_{i_1} w_{i_2} \dots w_{i_m} = x_{i_1} x_{i_2} x_{i_3}$ , so this is solution of PCP.