

B. E.

Fourth Semester Examination, Dec.-2006

THEORY OF AUTOMATA AND COMPUTATION

Note : Attempt any five questions. Parts of a question must be attempted together.

Q. 1. (a) "Let L be a set accepted by a Non-deterministic finite Automaton-then there exists a deterministic finite Automaton that accepts L ." Prove it.

Ans. Let $M = (Q, \Sigma, \delta, q_0, F)$ be an NFA accepting L . Define a DFA $M' = (Q', \Sigma, \delta', q_0', F')$ as follows:

The states of M' are all the subsets of the set of states of M . That is $Q' = 2^Q$. M' will keep track in its states of all the states M could be in at any given time. F' is the set of all states in Q' containing a final state of M . An element of Q' will be denoted by $\{q_1, q_2, \dots, q_i\}$. Where q_1, q_2, \dots, q_i are in Q .

$$\delta'(\{q_1, q_2, \dots, q_i\}, a) = \{p_1, p_2, \dots, p_j\}$$

iff

$$\delta(\{q_1, q_2, \dots, q_i\}, a) = \{p_1, p_2, \dots, p_j\}$$

$$\delta'(q_0', x) = \{q_1, q_2, \dots, q_i\}$$

iff

$$\delta(q_0, x) = \{q_1, q_2, \dots, q_i\} \quad (\text{by induction})$$

Suppose the hypothesis is true for inputs of length m or less. Let x_a be a string of length $m+1$ with a in Σ . Then

$$\delta'(q_0', x_a) = \delta'(\delta'(q_0', x), a)$$

By inductive hypothesis

$$\delta'(q_0', x) = \{p_1, p_2, \dots, p_j\}$$

iff

$$\delta(q_0, x) = \{p_1, p_2, \dots, p_j\}$$

But by def δ'

$$\delta'(\{p_1, p_2, \dots, p_j\}, a) = \{r_1, r_2, \dots, r_k\}$$

iff

$$\delta(\{p_1, p_2, \dots, p_j\}, a) = \{r_1, r_2, \dots, r_k\}$$

Thus $\delta(q_0, x_n) = [r_1, r_2, \dots, r_k]$

iff $\delta(q_0, x_n) = \{r_1, r_2, \dots, r_k\}$

Which establish inductive relations.

Q. 1. (b) Find a regular expression corresponding to each of the following subsets of $\{0, 1\}^*$.

(i) The language of all strings containing exactly two 0's.

Ans. Any string λ can be obtained by $\{0, 1\}$. $\{0, 1\}$ is represented by $0 + 1$. Hence λ can be represented

as

$$(1)^*00.$$

Q. 1. (b) (ii) The language of all strings in which the number of 0's is even.

Ans. Any string of such type can be obtained by concatenating r and any string over $\{0, 1\}$

So the string can be

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

Q. 2. (a) What are Mealy and Moore machines? Let $M_1 = \{Q, \Sigma, \Delta, \delta, \alpha, q_0\}$ be a Mealy machine.

Show that there is a Moore machine M_2 equivalent to M_1 .

Ans. Moore machines : It is a six tuple $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$ where Q, Σ, δ and q_0 are as in the DFA. Δ is the output alphabet and λ is a mapping from Q to Δ giving the output associated with each state. The output of M in response to input $a_1 a_2 \dots a_n$ $n \geq 0$ is $\lambda(q_0)\lambda(q_1)\dots\lambda(q_n)$ where q_0, q_1, \dots, q_n is the sequence of states such that $\delta(q_{i-1}, a_i) = q_i$ for $1 \leq i \leq n$. Any Moore machine gives output $\lambda(q_0)$ in response to input ϵ .

Mealy Machine : It is also a six tuple. $M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$ where all is as in Moore machine except that λ maps $Q \times \Sigma$ to Δ . That is $\lambda(q, a)$ gives the output associated with the transition from state q on input a . The output of M in response to input $a_1 a_2 \dots a_n$ is $\lambda(q_0, a_1)\lambda(q_1, a_2)\dots\lambda(q_{n-1}, a_n)$ where q_0, q_1, \dots, q_n is the sequence of states such that $\delta(q_{i-1}, a_i) = q_i$ for $1 \leq i \leq n$.

If $M_1 = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$ is a Moore machine then there is a mealy machine M_2 equivalent to M_1 .

Let $M_2 = (Q, \Sigma, \Delta, \delta, \lambda', q_0)$ and define $\lambda'(q, a)$ to be $\lambda(\delta(q, a))$ for all states q and input symbols a .

Then M_1 and M_2 enter the same sequence of states on the same input and with each transition M_2 emits the output that M_1 associates with the state entered.

Q. 2. (b) Convert Moore machine to equivalent Mealy M/c.

Given :

Present state	Next state	Output
q_1	q_1, q_2	0
q_2	q_1, q_3	0
q_3	q_1, q_3	1

Ans. Transition table

Present State	Next state			
	$a = 0$		$a = 1$	
	State	O/P	State	O/P
q_1	q_1	0	q_2	0
q_2	q_1	0	q_3	1
q_3	q_1	0	q_3	1

Mealy machine conversion will be.

Present State	Next state			
	$a = 0$		$a = 1$	
	State	O/P	State	O/P
q_1	q_1	0	q_2	0
q_2	q_1	0	q_2	1

Q. 3. (a) State and prove Myhill-Nerode Theorem for minimisation of finite Automata.

Ans. The Myhill-Nerode theorem has, the implication that there is an essentially unique minimum state Δ F.A for every regular set. The minimum state automation accepting a set L is unique up to an isomorphism and is given by M' .

We say that any DFA $M = (Q, \Sigma, \delta, q_0, F)$ accepting L defines an equivalence relation that is a refinement of R_L . Thus the number of states of M is greater than or equal to the number of states of M' . If equalizing holds then each of the states of M can be identified with one of the states of M' . That is let q be a state of M . There must be some x in Σ^* , such that $\delta(q_0, x) = q$. Otherwise q could be removed from Q and a small automation found. Identify q with the state $\delta'(q_0, x)$ of M' . This identification be consistent. If $\delta(q_0, x) = \delta(q_0, y) = q$ and y are in same equivalence class of R_L .

Thus $\delta'(q_0, x) = \delta'(q_0, y)$.

Q. 3. (b) Find CSG generating the language :

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

Ans. We construct G as follows :

$$G = (\{S, S_1, S_2, S_3, A, B\}, \{a, b\}, P, S)$$

Where P consists of

$$P_1: S \rightarrow S_1 S_2 S_3$$

$$P_2, P_3: S_1 S_2 \rightarrow aS, A$$

$$S_1 S_2 \rightarrow bS_1 B$$

$$P_4, P_5: AS_3 \rightarrow S_2 AS_3$$

$$BS_3 \rightarrow S_2 bS_3$$

$$P_6, P_7, P_8, P_9: Aa \rightarrow aA$$

$$Ab \rightarrow bA, Ba \rightarrow aBBb \rightarrow bB$$

$$P_{10}, P_{11}: aS_2 \rightarrow S_2 a$$

$$bS_2 \rightarrow S_2 b$$

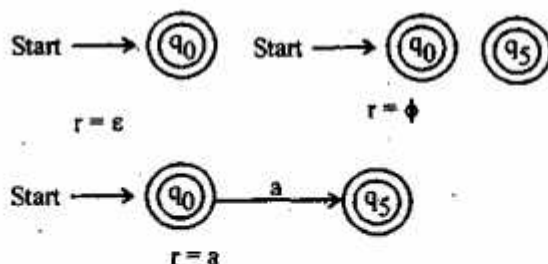
$$P_{12}, P_{13}: S_1 S_2 \rightarrow \Lambda$$

$$S_3 \rightarrow \Lambda$$

Q. 4. What do you mean by Regular expression? Let r be a regular expression, then prove that there exists a NFA with ϵ moves that accepts $L(r)$.

Ans. The languages accepted by finite automata are easily described by simple expression called regular expression. Let ϵ be an alphabet. The regular expressions over ϵ and the sets that they denote are defined recursively as follows :

1. ϕ is a regular expressions and denotes the empty set.
2. ϵ is a regular expression and denotes the set $\{\epsilon\}$.
3. For each $a \in \Sigma$; a^+ is a regular expressions and denotes the set $\{a\}$.
4. If r and s are regular expressions denoting the languages R and S respect. Then $(r + s)$, (rs) and (r^*) are regular expressions that denote the sets $R \cup S$, RS and R^* respect. We show by induction on the number of operators in the regular expression r that there is an NFA M with Σ transitions, having one final state and no transition out of this final state such that $L(M) = L(r)$. The expression r must be Σ , ϕ or a for source a in Σ . The NFA clearly satisfy following conditions in case of zero operator.



If there are one or more operator. Let regular expression r have i operator then there can be three cases.

1. $\rightarrow r = r_1 + r_2$
2. $r = r_1 * r_2$
3. $r = r_1^*$

Q. 5. (a) Design a Turing machine to accept the Language

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

Ans. Let $Q = \{q_0, q_1, q_2, q_3, q_4\}$, $\Sigma = \{0, 1\}$, $\Gamma = \{0, 1, X, Y, B\}$ and $F = \{q_4\}$. Each state represents a statement or a group of statements in a program. State q_0 is entered initially and also immediately prior to each replacement of a leftmost 0 by an X.

State	Symbol				
	0	1	X	Y	B
q_0	(q_1, X, R)	—	—	(q_3, Y, R)	—
q_1	(q_1, O, R)	(q_2, Y, L)	—	(q_1, Y, R)	—
q_2	—	(q_0, X, R)	(q_2, Y, L)	—	—
q_3	—	—	—	(q_3, Y, R)	(q_4, B, R)
q_4	—	—	—	—	—

Computation of M

$q_0 0011 \vdash Xq_1 011 \vdash XOq_1 11 \vdash Xq_2 OY11 \vdash$
 $q_2 XOY11 \vdash Xq_0 OY11 \vdash XXq_1 Y11 \vdash XXYq_1 11 \vdash$

$XXq_2YY \vdash Xq_2XYY \vdash XXq_0YY \vdash XXYq_3Y \vdash$

$XXYYq_3 \vdash XXYYBq_4$.

Q. 5. (b) Explain Chomsky hierarchy of grammar. What is relation between languages of classes?

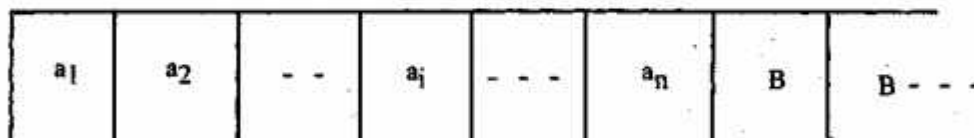
Ans. If all productions of a CFG are of the form $A \rightarrow WB$ or $A \rightarrow W$ where A and B are variables and W is a string of terminals, then we say the grammar is right linear. If all productions are of form $A \rightarrow BW$ or $A \rightarrow W$ we call it left linear. A right or left linear grammar is called a regular grammar. The four classes of languages—i.e., sets, CSL's, CFL's and regulator sets are often referred to as languages of types 0, 1, 2 and 3 respect.

CSL's and recursive sets.

- Every CSL is recursive.
- There is a recursive language that is not context sensitive.
- The regular sets are properly contained in the context-free language.
- The CFL's not containing the empty string are properly contained the context sensitive languages.
- The CSL's are properly contained in the re c. sets.

Q. 6. What is a Turing machine? What are the advantages of T.M. over Finite state machine. What is halting problem of T.M.?

Ans. Turing machine, has a finite control, an input tape that is divided into cells and a tape a tap at a time. The tape has a leftmost cell but is infinite to the right. Each cell of the tape may hold exactly one of a finite number of tape symbols. Initially, the n leftmost cells, for some finite $n \geq 0$, hold the input, which is a string of symbols chosen from a subset of the tape symbols called the input symbols. The remaining infinity of cells each hold the blank which is a special tape symbol that is not an input symbol.



The finite automation is a mathematical model of a system with discrete inputs and outputs. The system can be in any one of a finite number of internal configurations or "states". The state of the system summarises the information concerning past inputs that is needed to determine the behavior of the system on subsequent inputs.

The language accepted by M denoted $L(M)$ is the set of those words in Σ^* that cause M to enter a final state when placed, justified at the left, on the tape of M with M in state q_0 and the tape head of M at the leftmost cells formally the language accepted by $M = (Q, \Sigma, r, \delta, q_0, B, F)$ is

$$\{\omega / \omega \in \Sigma^* \text{ and } q_0\omega \vdash \alpha_1 p \alpha_2 \text{ for } p \in F \text{ and } \alpha_1 \text{ and } \alpha_2 \in \Gamma^*\}.$$

Given a TM recognizing a language L we assume without loss of generality that the TM halts i.e. has no next move, whenever the input is accepted. However for words not accepted it is possible that the TM will

never halt.

Q. 7. Explain the following :

(i) Chomsky Normal Form (CNF)

(ii) Difference between context free and context sensitive grammar

(iii) Greibach normal form.

Ans. (i) Chomsky normal form (CNF) : Any context-free language without ϵ is generated by a grammar in which all productions are of the form $A \rightarrow BC$ or $A \rightarrow a$. Here A, B, C are variables and a is a terminal.

Let G be a context-free grammar generating a language not containing by CFL we can find an equivalent grammar $G_1 = (V, T, P, S)$ such that P contain no unit production or ϵ -production. Thus, if a production has a single symbol on the right that symbol is a terminal and the production is already in an acceptable form.

(ii) Difference between context free and context sensitive grammar.

Context-free grammar is a finite set of variables (also called non-terminals or syntactic categories) each of which represents a language. The language represented by the variables are described recursively in terms of each other and primitive symbols called terminals. The rules relating the variables are called productions. A typical production states that the language associated with a given variable contains strings that are formed by concatenating strings from the languages of certain to her variables possibly along with some terminals.

Suppose we place restriction on productions $\alpha \rightarrow \beta$ of phase structure grammar that β be atleast as long as α . Then we can say the resulting grammar is context sensitive and its languages a context-sensitive language.

(iii) Greibach normal form : GNF (Greibach Normal Form) : Every context-free language L without ϵ can be generated by a grammar for which every production is of the form $A \rightarrow a\alpha$, where A is a variable, 'a' is a terminal and α is (possibly empty) string of variables.

Q. 8. Write short notes on the following :

(i) Universal Turing Machines

(ii) Non-deterministic Turing Machines

(iii) Linear Bounded Automata

(iv) Context Sensitive Language.

Ans. (i) Universal turing machines : We encode turing machine with restricted alphabets as strings over $\{0, 1\}$. Let

$$M = (Q, \{0, 1\}, \{0, 1, \beta\}, \delta, q, \beta, \{q_2\})$$

Be a turing machine with input alphabets $\{0, 1\}$ and the blank as the only additional tape symbol. Assume that $Q = \{q_1, q_2, \dots, q_n\}$ is the set of states and that q_2 is the final state. It is convenient to call symbols 0, 1 and β by synonyms X_1, X_2, X_3 respect. We also give directions L and R the synonyms D_1 and D_2 respect. Then a generic move $\delta(q_i, X_j) = (q_k, X_L, D_m)$ is encoded by the binary string.

$$0^i 1 0^j 1 0^k 1 0^l 1 0^m$$

The binary code for turing machine M is

$$||| \text{Code}_1 || \text{code}_2 || \dots || \text{code}_r |||.$$

(ii) **Non-deterministic turing machines** : A non-deterministic turing machine is a device with a finite control and a single one way infinite tape. For a given state and tape symbol scanned by the tape head, the machine has a finite number of choices for the next move. Each choice consists of a new state, a tape symbol to print and a direction of head motion. The non-deterministic TM is not permitted to make a move in which the next state is selected from one choice and the symbol printed and/or direction of head motion are selected from other choices.

(iii) **Linear bounded automata** : A linear bounded automation (LBA) is a non-deterministic turing machine satisfying the following two conditions :

1. Its input alphabet includes two special symbols ϕ and $\$$ the left and right end markers respect.
2. The LBA has no moves left from ϕ or right from $\$$ nor may it print another symbol over ϕ or $\$$.

(iv) **Context sensitive language** : Suppose we place the restriction on production $\alpha \rightarrow \beta$ of phrase structure grammar that β be at least as long as α . The resulting grammar context-sensitive and its language a context-sensitive language. The term "context-sensitive" comes from a normal form for these grammars, where each production is of the form look almost like context-free productions but they permit replacement of variable A by string B only in the "context" $\alpha_1 - \alpha_2$.