Theory Of Automata/Computation

1. Why are switching circuits called as finite state systems?
A switching circuit consists of a finite number of gates, each of which can be in any one of the two conditions 0 or 1.Although the voltages assume infinite set of values, the electronic circuitry is designed so that the voltages corresponding to 0 or 1 are stable and all others adjust to these value. Thus control unit of a computer is a finite state system.

2.Give the examples/applications designed as finite state system.
Text editors and lexical analyzers are designed as finite state systems. A lexical analyzer scans the symbols of a program to locate strings corresponding to identifiers, constants etc, and it has to remember limited amount of information .

3.Define: (i) Finite Automaton(FA) (ii)Transition diagram
FA consists of a finite set of states and a set of transitions from state to state that occur on input symbols chosen from an alphabet . Finite Automaton is denoted by a 5- tuple(Q, , ,q_0,F), where Q is the finite set of states , is a finite input alphabet, q_0 in Q is the initial state, F is the set of final states and is the transition mapping function Q * to Q.
Transition diagram is a directed graph in which the vertices of the graph correspond to the states of FA. If there is a transition from state q to state p on input a, then there is an arc labeled ,, a ,, from q to p in the transition diagram.

4. What are the applications of automata theory?

- In compiler construction.
- In switching theory and design of digital circuits.
- To verify the correctness of a program.
- Design and analysis of complex software and hardware systems.
- To design finite state machines such as Moore and mealy machines.

5. What is Moore machine and Mealy machine?
A special case of FA is Moore machine in which the output depends on the state of the machine. An automaton in which the output depends on the transition and current input is called Mealy machine.

6.What are the components of Finite automaton model?
The components of FA model are Input tape, Read control and finite control :
(a)The input tape is divided into number of cells. Each cell can hold one i/p symbol.
(b)The read head reads one symbol at a time and moves ahead.
(c)Finite control acts like a CPU. Depending on the current state and input symbol read from the input tape it changes state.

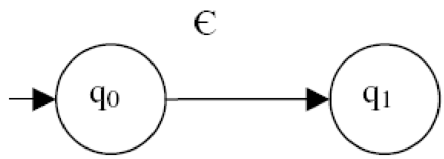
7.Differentiate NFA and DFA

NFA or Non Deterministic Finite Automaton is the one in which there exists many paths for a specific input from current state to next state. NFA can be used in theory of computation because they are more flexible and easier to use than DFA.

Deterministic Finite Automaton is a FA in which there is only one path for a specific input from current state to next state. There is a unique transition on each input symbol.

8.What is ϵ-closure of a state q_0?

ϵ -closure(q_0) denotes a set of all vertices p such that there is a path from q_0 to p labeled ϵ. Example :



ϵ-closure(q0)={q0,q1}

9.What is a : (a) String (b) Regular language

A string x is accepted by a Finite Automaton M=(Q,Σ, δ .q0,F) ifδ(q0,x)=p, for some p in F.FA accepts a string x if the sequence of transitions corresponding to the symbols of x leads from the start state to accepting state.

The language accepted by M is L(M) is the set {x | δ (q0,x) is in F}. A language is regular if it is accepted by some finite automaton.

10.What is a regular expression?

A regular expression is a string that describes the whole set of strings according to certain syntax rules. These expressions are used by many text editors and utilities to search bodies of text for certain patterns etc. Definition is: Let Σ be an alphabet. The regular expression over Σ and the sets they denote are:

i. Φ is a r.e and denotes empty set.

ii. ϵ is a r.e and denotes the set {ϵ}

iii. For each „a" in Σ , a^+ is a r.e and denotes the set {a}.

iv. If „r" and „s" are r.e denoting the languages R and S respectively then (r+s),(rs) and (r*) are r.e that denote the sets RUS, RS and R^* respectively.

11. Differentiate L^* and L^+

L^* denotes Kleene closure and is given by :

$$L^* = \bigcup_{i=0}^{\infty} L^i$$

Example : 0^* ={ϵ ,0,00,000,…………………………………}

Language includes empty words also.

L^+ denotes Positive closure and is given by :

$$L^+ = \bigcup_{i=1}^{\infty} L^i$$

12.What is Arden's Theorem?

Arden‟s theorem helps in checking the equivalence of two regular expressions. Let P and Q be the two regular expressions over the input alphabet Σ. The regular expression R is given as :R=Q+RP

Which has a unique solution as R=QP*.

13.Write a r.e to denote a language L which accepts all the strings which begin or end with either 00 or 11?

The r.e consists of two parts:

L1=(00+11) (any no of 0‟s and 1‟s) =(00+11)(0+1)*

L2=(any no of 0‟s and 1‟s)(00+11) =(0+1)*(00+11)

Hence r.e R=L1+L2 =[(00+11)(0+1)*] + [(0+1)* (00+11)]

14.Construct a r.e for the language which accepts all strings with atleast two c's over the set Σ ={c,b}

(b+c)* c (b+c)* c (b+c)*

15.Construct a r.e for the language over the set ={a,b} in which total number of a's are divisible by 3

(b* a b* a b* a b*)*

16.what is: (i) (0+1)* (ii)(01)* (iii)(0+1) (iv)(0+1)+

(0+1)*= { , 0 , 1 , 01 , 10 ,001 ,101 ,101001,…………………}

Any combinations of 0‟s and 1‟s.

(01)*={ , 01 ,0101 ,010101 ,…………………………………..}

All combinations with the pattern 01.

(0+1)= 0 or 1,No other possibilities.

(0+1)+= {0,1,01,10,1000,0101,………………………………….}

17.Reg exp denoting a language over ={1} having (i)even length of string (ii)odd length of a string?

(i) Even length of string R=(11)*

(ii) Odd length of the string R=1(11)*

18.Regular expression for:

(i)All strings over {0,1} with the substring '0101'

(ii)All strings beginning with '11 ' and ending with 'ab'

(iii)Set of all strings over {a,b}with 3 consecutive b's.

(iv)Set of all strings that end with '1'and has no substring '00'

(i)(0+1)* 0101(0+1)*

(ii)11(1+a+b)* ab

(iii)(a+b)* bbb (a+b)*

(iv)(1+01)* (10+11)* 1

19. What are the applications of Regular expressions and Finite automata?

Lexical analyzers and Text editors are two applications :

Lexical analyzers: The tokens of the programming language can be expressed using regular expressions. The lexical analyzer scans the input program and separates the tokens. For e.g. identifier can be expressed as a regular expression as:

(letter)(letter + digit)*

If anything in the source language matches with this regular expression then it is recognized as an identifier .The letter is{A,B,C,………..Z, a,b,c….z} and digit is {0,1,…9}.Thus regular expression identifies token in a language.

Text editors: These are programs used for processing the text. For example UNIX text editors uses the regular expression for substituting the strings such as: S/bbb*/b/

Gives the substitute a single blank for the first string of two or more blanks in a given line. In UNIX text editors any regular expression is converted to an NFA with ϵ –transitions, this NFA can be then simulated directly.

20.Reg exp for the language that accepts all strings in which 'a' appears tripled over the set Σ ={a}

reg exp=(aaa)*

21.What are the applications of pumping lemma?

Pumping lemma is used to check if a language is regular or not.

(i) Assume that the language(L) is regular.

(ii) Select a constant „n‟.

(iii) Select a string(z) in L, such that |z|>n.

(iv) Split the word z into u,v and w such that |uv|<=n and |v|>=1.

(v) You achieve a contradiction to pumping lemma that there exists an „i‟

Such that uviw is not in L.Then L is not a regular language.

22. What is the closure property of regular sets?

The regular sets are closed under union, concatenation and Kleene closure.

r1Ur2= r1 +r2

r1.r2= r1r2

(r)*=r*

The class of regular sets are closed under complementation, substitution, homomorphism and inverse homomorphism.

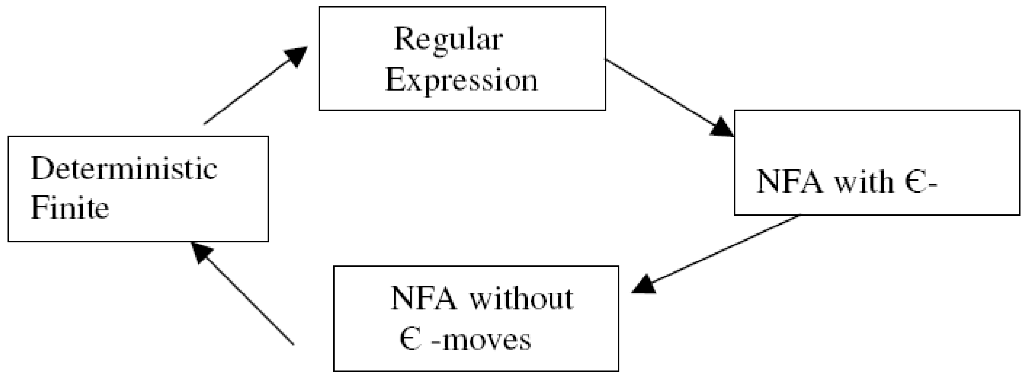
23.Reg exp for the language such that every string will have atleast one 'a' followed by atleast one 'b'?

$R=a^+b^+$

24.Write the exp for the language starting with and has no consecutive b's

reg exp=(a+ab)*

25.What is the relationship between FA and regular expression.



26. What are the applications of Context free languages?

Context free languages are used in :

- Defining programming languages.
- Formalizing the notion of parsing.
- Translation of programming languages.
- String processing applications.

27. What are the uses of Context free grammars?

- Construction of compilers.
- Simplified the definition of programming languages.
- Describes the arithmetic expressions with arbitrary nesting of balanced parenthesis { (,) }.
- Describes block structure in programming languages.
- Model neural nets.

28.Define a context free grammar

A context free grammar (CFG) is denoted as G=(V,T,P,S) where V and T are finite set of variables and terminals respectively. V and T are disjoint. P is a finite set of productions each is of the form A→a where A is a variable and „a" is a string of symbols from (V U T)*.

29.What is the language generated by CFG or G?

The language generated by G (L(G)) is {w | w is in T* and $S \overset{*}{\underset{G}{\Rightarrow}} w$ .That is a string is in L(G) if:

(1) The string consists solely of terminals.

(2) The string can be derived from S.

30.What is : (a) CFL (b) Sentential form

L is a context free language (CFL) if it is L(G) for some CFG G. A string of terminals and variables „a" is called a sentential form if:

$S \xrightarrow{*} a$,where S is the start symbol of the grammar.

31. What is the language generated by the grammar G=(V,T,P,S) where P={S→aSb, S→ab}?

S=> aSb=>aaSbb=>…………………………..=>$a^n b^n$

Thus the language L(G)={$a^n b^n$ | n>=1}.The language has strings with equal number of a‟s and b‟s.

32. What is :(a) derivation (b)derivation/parse tree (c) subtree

(a) Let G=(V,T,P,S) be the context free grammar. If A→β is a production of P and α and γ are any strings in (VUT)* then $\alpha A\gamma \xrightarrow[G]{} \alpha\beta\gamma$.

(b) A tree is a parse \ derivation tree for G if:

(i) Every vertex has a label which is a symbol of VU TU{_}.

(ii) The label of the root is S.

(iii) If a vertex is interior and has a label A, then A must be in V.

(iv) If n has a label A and vertices n1,n2,….. nk are the sons of the vertex n in order from left with labels X1,X2,………..X_k respectively then A X1X2…..X_k must be in P.

(v) If vertex n has label ,then n is a leaf and is the only son of its father.

(c) A subtree of a derivation tree is a particular vertex of the tree together with all its descendants ,the edges connecting them and their labels. The label of the root may not be the start symbol of the grammar.

33. If S→aSb | aAb , A→bAa , A→ba .Find out the CFL?

Soln. S→aAb→abab

S→aSb→aaAb b →a a ba b b(sub S→aAb)

S→aSb→a aSbb →aaaAbbb→aaababbb

Thus L={$a^n b^m a^m b^n$, where n,m>=1}

34. What is a ambiguous grammar?

A grammar is said to be ambiguous if it has more than one derivation trees for a sentence or in other words if it has more than one leftmost derivation or more than one rightmost derivation.

35.Consider the grammarP={S->aS | aSbS | _ } is ambiguous by constructing:

(a) two parse trees (b) two leftmost derivation (c) rightmost derivation

Consider a string aab :

(a)

(b) (i)S=>aS
=>aaSbS
=>aabS
=>aab

(ii) S=>aSbS
=>aaSbS
=>aabS
=>aab

(c)(i)S=>aS
=>aaSbS
=>aaSb
=>aab
=>aab

(ii) S=>aSbS
=>aSb
=>aaSbS
=>aaSb

36. Find CFG with no useless symbols equivalent to : S→AB | CA , B→BC | AB, A→a , CaB | b?

S→ AB
S→CA
B→BC
B→AB
A→a
C→aB
C→b are the given productions.

A symbol X is useful if $S \overset{*}{\Rightarrow} X \overset{*}{\Rightarrow} w$
The variable B cannot generate terminals as B→BC and B→AB.
Hence B is useless symbol and remove B from all productions.
Hence useful productions are: S→CA , A→a , C→b

37. Construct CFG without ϵ production from : S a | Ab | aBa , A b | ϵ , B b | A.

S→a
S→Ab
S→aBa
A→b A→ϵ**,** B→b, B→A are the given set of production.
A→ϵ is the only empty production. Remove the empty production

S→ Ab , Put A→ ∈ and hence S→ b.
If B→ A and A→∈ then B →∈
Hence S→aBa becomes S→aa .
Thus S→ a | Ab | b | aBa | aa
A→b
B→b
Finally the productions are: S→ a | Ab | b | aBa | aa
A→b
B→b

38. What are the three ways to simplify a context free grammar?

- By removing the useless symbols from the set of productions.
- By eliminating the empty productions.
- By eliminating the unit productions.

39. What are the properties of the CFL generated by a CFG?

- Each variable and each terminal of G appears in the derivation of some word in L
- There are no productions of the form A→B where A and B are variables.

40. Find the grammar for the language L={a^{2n} bc ,where n>1 }

let G=({S,A,B}, {a,b,c} ,P , {S}) where P:
S->Abc
A->aaA |

41.Find the language generated by :S→0S1 | 0A | 0 |1B | 1

A→0A | 0 , B→1B | 1
The minimum string is S→ 0 | 1
S→0S1=>001
S→0S1=>011
S→0S1=>00S11=>000S111=>0000A111=>00000111
Thus L={ 0^n 1^m | m not equal to n, and n,m >=1}

42.Construct the grammar for the language L={ an b an | n>=1}.

The grammar has the production P as:
S→aAa
A→aAa | b
The grammar is thus : G=({S,A} ,{a,b} ,P,S)

43. Construct a grammar for the language L which has all the strings which are all palindrome over Σ={a, b}.

G=({S}, {a,b} , P, S)
P:{ S → aSa ,S→ b S b,S→ a,S→b,S→∈ } which is in palindrome.

44. Differentiate sentences Vs sentential forms

A sentence is a string of terminal symbols.

A sentential form is a string containing a mix of variables and terminal symbols or all variables.This is an intermediate form in doing a derivation.

45. What is a formal language?

Language is a set of valid strings from some alphabet. The set may be empty,finite or infinite. L(M) is the language defined by machine M and L(G) is the language defined by Context free grammar. The two notations for specifying formal languages are:

- ✓ Grammar or regular expression(Generative approach)
- ✓ Automaton(Recognition approach)

46. (a)CFL are not closed under intersection and complementation –True.

(b)A regular grammar generates an empty string –True.

(c) A regular language is also context free but not reverse–True.

(d)A regular language can be generated by two or more different grammar –True.

(e) Finite State machine(FSM) can recognize only regular grammar-True.

47.What is Backus-Naur Form(BNF)?

Computer scientists describes the programming languages by a notation called Backus- Naur Form. This is a context free grammar notation with minor changes in format and some shorthand.

48. Let G= ({S,C} ,{a,b},P,S) where P consists of S→aCa , C→aCa |b. Find L(G).

S→ aCa => aba

S→aCa=> a aCa a=>aabaa

S→aCa=> a aCa a=> a a aCa a a =>aaabaaa

Thus L(G)= { $a^n b a^n$,where n>=1 }

49.Find L(G) where G= ({S} ,{0,1}, {S→0S1 ,S→ },S)

S→ , is in L(G)

S-> 0S1 =>0 1=>01

S->0S1=>0 0S11=>0011

Thus L(G)= { $0^n 1^n$ | n>=0}

50.What is a parser?

A parser for grammar G is a program that takes as input a string w and produces as output either a parse tree for w ,if w is a sentence of G or an error message indicating that w is not a sentence of G.

51.Define Pushdown Automata.

A pushdown Automata M is a system (Q, Σ, Γ ,δ ,q_0, Z_0,F) where

Q is a finite set of states.

Σ is an alphabet called the input alphabet.

Γ is an alphabet called stack alphabet.

q_0 in Q is called initial state.

Z_0 in Γ is start symbol in stack.
F is the set of final states.
δ is a mapping from Q X (Σ U {ϵ}) X Γ to finite subsets of Q X Γ *.

52.Compare NFA and PDA.

NFA	PDA
1.The language accepted by NFA is the regular language.	The language accepted by PDA is Context free language.
2.NFA has no memory.	PDA is essentially an NFA with a stack(memory).
3. It can store only limited amount of information.	It stores unbounded limit of information.
4.A language/string is accepted only by reaching the final state.	It accepts a language either by empty Stack or by reaching a final state.

53.Specify the two types of moves in PDA.
The move dependent on the input symbol(a) scanned is:
δ (q,a,Z) = { (p_1,Y_1), (p_2,Y_2),……..(p_m,Y_m) }
where q and p are states , a is in Σ ,Z is a stack symbol and i is in Γ*.
PDA is in state q , with input symbol a and Z the top symbol on state enter state pi
Replace symbol Z by string Y_i.
The move independent on input symbol is (ϵ-move):
δ (q, ϵ,Z)= { (p_1,Y_1), (p_2,Y_2),…………(p_m,Y_m) }.
Is that PDA is in state q , independent of input symbol being scanned and with
Z the top symbol on the stack enter a state p_i and replace Z by Y_i.

54.What are the different types of language acceptances by a PDA and define them.
For a PDA M=(Q, Σ ,Γ ,δ ,q_0 ,Z_0 ,F) we define :

- Language accepted by final state L(M) as:

  { w | (q_0 , w , Z_0) $\overset{*}{\vdash}$ (p, ϵ , γ) for some p in F and γ in Γ * }.
- Language accepted by empty / null stack N(M) is:

  { w | (q_0,w ,Z_0) $\overset{*}{\vdash}$(p, ϵ, ϵ) for some p in Q}.

55.Is it true that the language accepted by a PDA by empty stack and final states are different languages.
No, because the languages accepted by PDA „s by final state are exactly the languages accepted by PDA‟s by empty stack.

56. Define Deterministic PDA.

A PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ is deterministic if:

- For each q in Q and Z in Γ, whenever $\delta(q, \epsilon, Z)$ is nonempty, then $\delta(q, a, Z)$ is empty for all a in Σ.
- For no q in Q, Z in Γ, and a in $\Sigma \cup \{\epsilon\}$ does $\delta(q, a, Z)$ contains more than one element.

(Eg): The PDA accepting $\{wcw^R \mid w \text{ in } (0+1)^*\}$.

57.Define Instantaneous description(ID) in PDA.

ID describe the configuration of a PDA at a given instant.ID is a triple such as (q, w, γ), where q is a state, w is a string of input symbols and γ is a string of stack symbols. If $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ is a PDA we say that

$(q, aw, Z\alpha) \vdash_{M} (p, w, \beta\alpha)$ if $\delta(q, a, Z)$ contains (p, β).

'a' may be ϵ or an input symbol.

Example: (q_1, BG) is in $\delta(q_1, 0, G)$ tells that $(q_1, 011, GGR) \vdash (q_1, 11, BGGR)$.

58.What is the significance of PDA?

Finite Automata is used to model regular expression and cannot be used to represent non regular languages. Thus to model a context free language, a Pushdown Automata is used.

59.When is a string accepted by a PDA?

The input string is accepted by the PDA if:

- The final state is reached .
- The stack is empty.

60. Give examples of languages handled by PDA.

(1) $L = \{ a^n b^n \mid n >= 0 \}$,here n is unbounded , hence counting cannot be done by finite memory. So we require a PDA ,a machine that can count without limit.

(2) $L = \{ ww^R \mid w \in \{a,b\}^* \}$, to handle this language we need unlimited counting capability .

61.Is NPDA (Nondeterministic PDA) and DPDA (Deterministic PDA)equivalent?

The languages accepted by NPDA and DPDA are not equivalent.

For example: ww^R is accepted by NPDA and not by any DPDA.

62. State the equivalence of acceptance by final state and empty stack.

- If L = L(M2) for some PDA M2 , then L = N(M1) for some PDA M1.
- If L = N(M1) for some PDA M1 ,then L = L(M2) for some PDA M2.

where L(M) = language accepted by PDA by reaching a final state.

N(M) = language accepted by PDA by empty stack.

63. State the equivalence of PDA and CFL.

- If L is a context free language, then there exists a PDA M such that L=N(M).
- If L is N(M) for some PDA m, then L is a context free language.

64. What are the closure properties of CFL?

- CFL are closed under union, concatenation and Kleene closure.
- CFL are closed under substitution , homomorphism.
- CFL are not closed under intersection , complementation.
- Closure properties of CFL"s are used to prove that certain languages are not context free.

65. State the pumping lemma for CFLs.

Let L be any CFL. Then there is a constant n, depending only on L, such that if z is in L and |z| >=n, then z=uvwxy such that :
(i) |vx| >=1
(ii) |vwx| <=n and
(iii) for all i>=0 uv^iwx^iy is in L.

66. What is the main application of pumping lemma in CFLs?

The pumping lemma can be used to prove a variety of languages are not context free . Some examples are:
L1 ={ $a^ib^ic^i$ | i>=1} is not a CFL.
L2= { $a^ib^jc^id^j$ | i>=1 and J>=1 } is not a CFL.

67.What is Ogden's lemma?

Let L be a CFL. Then there is a constant n such that if z is any word in L, and we mark any n or more positions of z " distinguished" then we can write z=uvwxy such that:
(1) v and x together have atleast one distinguished position.
(2) vwx has at most n distinguished positions and
(3) for all i>=0 uv^iwx^iy is in L.

68. Give an example of Deterministic CFL.

The language L={a^nb^n : n>=0} is a deterministic CFL

69. What are the properties of CFL?

Let G=(V,T,P,S) be a CFG

- The fanout of G , _(G) is largest number of symbols on the RHS of any rule in R.
- The height of the parse tree is the length of the longest path from the root to some leaf.

70. Compare NPDA and DPDA.

71. What are the components of PDA ?

The PDA usually consists of four components:

- A control unit.
- A Read Unit.
- An input tape.
- A Memory unit.

72. What is the informal definition of PDA?

A PDA is a computational machine to recognize a Context free language. Computational power of PDA is between Finite automaton and Turing machines. The PDA has a finite control , and the memory is organized as a stack.

73. Give an example of NonDeterministic CFL

The language L={ ww^R : w $\in$ {a,b} + } is a nondeterministic CFL.

74. What is a Dyck language?

A Dyck language is a language with k-types of balanced parenthesis.
For ex: $[_1 [_2 [_1]_1 [_2]_2]_2]_1$ is in the Dyck language with two kinds of parenthesis.

75. What is a CYK algorithm?

Let Vij be the set of variables A such that $A \overset{*}{=>} x_{ij}$ iff there is some production A→BC and some k , 1<=k<=j such that B derives the first symbols of x_{ij} and C derives the last j-k symbols of x_{ij}.

76.What is a turing machine?

Turing machine is a simple mathematical model of a computer. TM has unlimited and unrestricted memory and is a much more accurate model of a general purpose computer. The turing machine is a FA with a R/W Head. It has an infinite tape divided into cells ,each cell holding one symbol.

77.What are the special features of TM?

In one move ,TM depending upon the symbol scanned by the tape head and state of the finite control:

- Changes state.
- Prints a symbol on the tape cell scanned, replacing what was written there.
- Moves the R/w head left or right one cell.

78. Define Turing machine.

A Turing machine is denoted as $M=(Q, \Sigma, \Gamma, \delta, q_0, B, F)$

Q is a finite set of states.

Σ is set of i/p symbols ,not including B.

Γ is the finite set of tape symbols.

q_0 in Q is called start state.

B in Γ is blank symbol.

F is the set of final states.

δ is a mapping from $Q \times \Gamma$ to $Q \times \Gamma \times \{L,R\}$.

79.Define Instantaneous description of TM.

The ID of a TM M is denoted as $\alpha_1 q \alpha_2$. Here q is the current state of M is in Q; $\alpha 1\ \alpha 2$ is the string in Γ^* that is the contents of the tape up to the rightmost nonblank symbol or the symbol to the left of the head, whichever is the rightmost.

80. What are the applications of TM?

TM can be used as:

- Recognizers of languages.
- Computers of functions on non negative integers.
- Generating devices.

81.What is the basic difference between 2-way FA and TM?

Turing machine can change symbols on its tape , whereas the FA cannot change symbols on tape. Also TM has a tape head that moves both left and right side ,whereas the FA doesn"t have such a tape head.

82. What is (a)total recursive function and (b)partial recursive function

If $f(i_1,i_2,\ldots\ldots\ldots i_k)$ is defined for all $i_1,\ldots..i_k$ then we say f is a total recursive function. They are similar to recursive languages as they are computed by TM that always halt. A function $f(i_1,\ldots i_k)$ computed by a Turing machine is called a partial recursive function. They are similar to r.e languages as they are computed by TM that may or may not halt on a given input.

83.Define a move in TM.

Let $X_1 X_2 \ldots X_{i-1}\, q\, X_i \ldots X_n$ be an ID.

The left move is: if $\delta(q, X_i) = (p, Y, L)$,if i>1 then

$$X_1 X_2 \ldots X_{i-1}\, q\, X_i \ldots X_n \underset{M}{\vdash\!\text{----}} X_1 X_2 \ldots X_{i-2}\, p\, X_{i-1}\, Y\, X_{i+1} \ldots X_n.$$

The right move is if $\delta(q, X_i) = (p, Y, R)$,if i>1 then

$$X_1 X_2 \ldots X_{i-1}\, q\, X_i \ldots X_n \underset{M}{\vdash\!\text{----}} X_1 X_2 \ldots X_{i-1}\, Y\, p\, X_{i+1} \ldots X_n.$$

84. What is the language accepted by TM?

The language accepted by M is L(M) , is the set of words in Σ^* that cause M to enter a final state when placed ,justified at the left on the tape of M, with M at qo and the tape head of M at the leftmost cell. The language accepted by M is:

$\{ w \mid w \text{ in } \Sigma^* \text{ and } q_0w \mid\!\!-\!\!-\!\!- \alpha1\ p\ \alpha2 \text{ for some } p \text{ in } F \text{ and } \alpha1, \alpha2 \text{ in } \Gamma^* \}$.

85.Give examples of total recursive functions.

All common arithmetic functions on integers such as multiplication , n!, [log2n] and 2^{2n} are total recursive functions.

86.What are(a) recursively enumerable languages (b) recursive sets?

The languages that is accepted by TM is said to be recursively enumerable (r. e) languages. Enumerable means that the strings in the language can be enumerated by the TM. The class of r. e languages include CFL"s. The recursive sets include languages accepted by at least one TM that halts on all inputs.

87. What are the various representation of TM?

We can describe TM using:

- Instantaneous description.
- Transition table.
- Transition diagram.

88. What are the possibilities of a TM when processing an input string?

- TM can accept the string by entering accepting state.
- It can reject the string by entering non-accepting state.
- It can enter an infinite loop so that it never halts.

89. What are the techniques for Turing machine construction?

• Storage in finite control.
• Multiple tracks.
• Checking off symbols.
• Shifting over
• Subroutines.

90. What is the storage in FC?

The finite control(FC) stores a limited amount of information. The state of the Finite control represents the state and the second element represent a symbol scanned.

91.When is checking off symbols used in TM?

Checking off symbols is useful method when a TM recognizes a language with repeated strings and also to compare the length of substrings.

(eg) : $\{ ww \mid w \in \Sigma^* \}$ or $\{a^ib^i \mid i>=1\}$.

This is implemented by using an extra track on the tape with symbols Blank or $\surd$.

92.When is shifting over Used ?

A Turing machine can make space on its tape by shifting all nonblank symbols a finite number of cells to the right. The tape head moves to the right , repeatedly storing the symbols in the FC and replacing the symbols read from the cells to the left. The TM can then return to the vacated cells and prints symbols.

93. What is a multihead TM?

A k-head TM has some k heads. The heads are numbered 1 through k, and move of the TM depends on the state and on the symbol scanned by each head. In one move, the heads may each move independently left or right or remain stationary.

94.What is a 2-way infinite tape TM?

In 2-way infinite tape TM, the tape is infinite in both directions. The leftmost square is not distinguished. Any computation that can be done by 2-way infinite tape can also be done by standard TM.

95.Differentiate PDA and TM.

PDA	TM
1. PDA uses a stack for storage.	1. TM uses a tape that is infinite .
2.The language accepted by PDA is CFL.	2. Tm recognizes recursively enumerable languages.

96.How can a TM used as a transducer?

A TM can be used as a transducer. The most obvious way to do this is to treat the entire nonblank portion of the initial tape as input , and to treat the entire blank portion of the tape when the machine halts as output. Or a TM defines a function y=f(x) for strings

$x, y \in \Sigma^*$ if: $q_0X \vdash\!\text{---}\ q_fY$, where q_f is the final state.

97. What is a multi-tape Turing machine?

A multi-tape Turing machine consists of a finite control with k-tape heads and k tapes ; each tape is infinite in both directions. On a single move depending on the state of finite control and symbol scanned by each of tape heads ,the machine can change state print a new symbol on each cells scanned by tape head, move each of its tape head independently one cell to the left or right or remain stationary.

98.What is a multidimensional TM?

The device has a finite control , but the tape consists of a k-dimensional array of cells infinite in all 2k directions, for some fixed k. Depending on the state and symbol scanned , the device changes state , prints a new symbol and moves its tape head in one of the 2k directions, either positively or negatively ,along one of the k-axes.

99. When a recursively enumerable language is said to be recursive ? Is it true that the language accepted by a non-deterministic Turing machine is different from recursively enumerable language?

A language L is recursively enumerable if there is a TM that accepts L and recursive if there is a TM that recognizes L. Thus r.e language is Turing acceptable and recursive language is Turing decidable languages.

No , the language accepted by non-deterministic Turing machine is same as recursively enumerable language.

100. What is Church's Hypothesis?

The notion of computable function can be identified with the class of partial recursive functions is known as Church-hypothesis or Church-Turing thesis. The Turing machine is equivalent in computing power to the digital computer.

101.When we say a problem is decidable? Give an example of undecidable problem?

A problem whose language is recursive is said to be decidable. Otherwise the problem is said to be undecidable. Decidable problems have an algorithm that takes as input an instance of the problem and determines whether the answer to that instance is "yes" or "no". (eg) of undecidable problems are (1)Halting problem of the TM.

102.Give examples of decidable problems.

1. Given a DFSM M and string w, does M accept w?
2. Given a DFSM M is L(M) = Φ ?
3. Given two DFSMs M1 and M2 is L(M1)= L(M2) ?
4. Given a regular expression α and a string w ,does α generate w?
5. Given a NFSM M and string w ,does M accept w?

103. Give examples of recursive languages?

i. The language L defined as L= { "M" ,"w" : M is a DFSM that accepts w} is recursive.
ii. L defined as { "M1" U "M2" : DFSMs M1 and M2 and L(M1) =L(M2) } is recursive.

104. Differentiate recursive and recursively enumerable languages.

Recursive languages	Recursively enumerable languages
1. A language is said to be recursive if and only if there exists a membership algorithm for it.	1. A language is said to be r.e if there exists a TM that accepts it.
2. A language L is recursive iff there is a TM that decides L. (Turing decidable languages). TMs that decide languages are algorithms.	2. L is recursively enumerable iff there is a TM that semi-decides L. (Turing acceptable languages). TMs that semi-decides languages are not algorithms.

105. What are UTMs or Universal Turing machines?

Universal TMs are TMs that can be programmed to solve any problem, that can be solved by any Turing machine. A specific Universal Turing machine U is : Input to U: The encoding "M " of a Tm M and encoding "w" of a string w. Behavior : U halts on input "M" "w" if and only if M halts on input w.

106. What is the crucial assumptions for encoding a TM?

There are no transitions from any of the halt states of any given TM . Apart from the halt state , a given TM is total.

107. What properties of recursive enumerable seta are not decidable?

- Emptiness
- Finiteness
- Regularity
- Context-freedom.

108.Define L_l .When is a trivial property?

L_l is defined as the set { <M> | L_l(M) is in . } is a trivial property if is empty or it consists of all r.e languages.

109. What is a universal language L_u?

The universal language consists of a set of binary strings in the form of pairs (M,w) where M is TM encoded in binary and w is the binary input string.

L_u = { < M,w> | M accepts w }.

110.What is a Diagonalization language Ld?

The diagonalization language consists of all strings w such that the TM M whose code is w does not accept when w is given as input.

111. What properties of r.e sets are recursively enumerable?

112. What properties of r.e sets are not r.e?

113.What are the conditions for L_l to be r.e?
L is recursively enumerable iff satisfies the following properties:
i. If L_l is in and L_l is a subset of L` ,then L` is in l (containment property)
ii. If L is an infinite language in l ,then there is a finite subset of L in l.
iii. The set of finite languages in l is enumaerable.

114. What is canonical ordering?
Let Σ^* be an input set. The canonical order for Σ^* as follows . List words in order of size, with words of the same size in numerical order. That is let $\Sigma = \{ x_0, x_1, \ldots x_{t-1} \}$ and x_i is the digit i in base t.
(e.g) If $\Sigma = \{ a,b \}$ the canonical order is ϵ , a ,b , aa, ab ,……..

115. How can a TM acts as a generating device?
In a multi-tape TM ,one tape acts as an output tape, on which a symbol, once written can never be changed and whose tape head never moves left. On that output tape , M writes strings over some alphabet Σ , separated by a marker symbol # , G(M) (where G(M) is the set w in Σ^* such that w is finally printed between a pair of #'s on the output device).

116. What are the different types of grammars/languages?
• Unrestricted or Phase structure grammar.(Type 0 grammar).(for TMs)
• Context sensitive grammar or context dependent grammar (Type1)(for Linear Bounded Automata)
• Context free grammar (Type 2) (for PDA)
• Regular grammar (Type 3) (for Finite Automata).
This hierarchy is called as Chomsky Hierarchy.

117. What is a PS or Unrestricted grammar?
A grammar without restrictions is a PS grammar. Defined as G=(V,T,P,S)
With P as :

$\Phi A \psi \rightarrow \Phi \alpha \psi$ where A is variable and $\Phi \alpha \psi$ is replacement string.

The languages generated by unrestricted grammars are precisely those accepted by Turing machines.

118. State a single tape TM started on blank tape scans any cell four or more times is decidable?
If the TM never scans any cell four or more times , then every crossing sequence is of length at most three. There is a finite number of distinct crossing sequence of length 3 or less. Thus either TM stays within a fixed bounded number of tape cells or some crossing sequence repeats.

119.Does the problem of " Given a TM M ,does M make more than 50 moves on input B "?
Given a TM M means given enough information to trace the processing of a fixed string for a certain fixed number of moves. So the given problem is decidable.

120. Show that AMBIGUITY problem is un-decidable.
Consider the ambiguity problem for CFGs. Use the "yes-no" version of AMB. An algorithm for FIND is used to solve AMB. FIND requires producing a word with two or more parses if one exists and answers "no" otherwise. By the reduction of AMB to FIND we conclude there is no algorithm for FIND and hence no algorithm for AMB.

121.State the halting problem of TMs.
The halting problem for TMs is:
Given any TM M and an input string w, does M halt on w?
This problem is undecidable as there is no algorithm to solve this problem.

122.Define PCP or Post Correspondence Problem.
An instance of PCP consists of two lists , A = $w_1, w_2, \ldots. w_k$ and B = $x_1, \ldots.. x_k$ of strings over some alphabet Σ .This instance of PCP has a solution if there is any sequence of integers $i_1, i_2, .. i_m$ with m >=1 such that
$w_{i1}, w_{i2}, \ldots w_{im} = x_{i1}, x_{i2}, \ldots x_{im}$
The sequence $i_1, i_2, \ldots i_m$ is a solution to this instance of PCP.

123.Define MPCP or Modified PCP.
The MPCP is : Given lists A and B of K strings from * ,say
A = $w_1, w_2, \ldots w_k$ and B= $x_1, x_2, \ldots.. x_k$ does there exists a sequence of integers i1,i2,…ir such that $w_1 w_{i1} w_{i2} \ldots.. w_{ir} = x_1 x_{i1} x_{i2} \ldots x_{ir}$?

124 . What is the difference between PCP and MPCP?
The difference between MPCP and PCP is that in the MPCP ,a solution is required to start with the first string on each list.

125. What are the concepts used in UTMs?
- Stored program computers.
- Interpretive Implementation of Programming languages.
- Computability.

126.Leftmost and rightmost derivations.
If we apply a production only to the leftmost variable at every step to derive the required string then it is called as leftmost derivation.
If we apply a production only to the rightmost variable at every step to derive the required string then it is called as rightmost derivation.
Example:
Consider G whose productions are S→aAS|a , A→SbA|SS|ba.For the string w=aabbaa find the leftmost and rightmost derivation.
LMD: S=>aAS
=>aSbAS
=>aabAS
=>aabbaS
=>aabbaa
RMD: S=>aAS

=>aAa
=>aSbAa
=>aSbbaa
=>aabbaa

127.Construction of reduced grammar.

• <u>Elimination of null productions</u>

- In a CFG,productions of the form A->_ can be eliminated, where A is a variable.

• <u>Elimination of unit productions.</u>

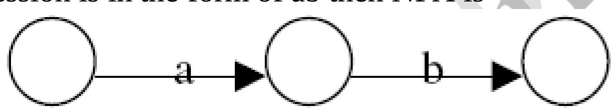
- In a CFG,productions of the form A->B can be eliminated, where A and B are variables.

• <u>Elimination of Useless symbols.</u>

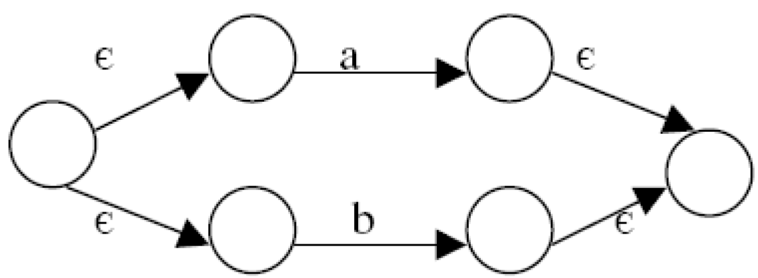
- These are the variables in CFG which does not derive any terminal or not reachable from the start symbols. These can also eliminated.

128.Construct the NFA with ϵ-transitions from the given regular expression.

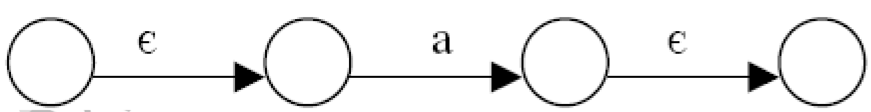
- If the regular expression is in the form of ab then NFA is



- If the regular expression is in a+b form then NFA is



- If the regular expression is in a* form then NFA is



129. Chomsky normal form(CNF)

If the CFG is in CNF if it satisfies the following conditions

- All the production must contain only one terminal or only two variables in the right hand side.

Example: Consider G with the production of S→aAB , A→ bC , B→b, C→c.

G in CNF is S→EB , E→DA , D→ a , A→FC , F→ b , B→b , C→ c.

130.Explain the various techniques for Turing machine construction.

- Storage in finite control
- Multiple tracks
- Checking off symbols
- Shifting over
- Subroutines.

131.Briefly explain the different types of Turing machines.

- Two way finite tape TM
- Multi tape TM
- Nondeterministic TM
- Multi dimensional TM
- Multihead TM

132. Explain how a TM can be used to determine the given number is prime or not?

It takes a binary input greater than 2,written on the first track, and determines whether it is a prime. The input is surrounded by the symbol $ on the first track. To test if the input is a prime, the TM first writes the number 2 in binary on the second track and copies the first track on to the third. Then the second track is subtracted as many times as possible, from the third track effectively dividing the third track by the second and leaving the remainder. If the remainder is zero, the number on the first track is not a prime.If the remainder is non zero,the number on the second track is increased by one.If the second track equals the first,the number on the first track is the prime.If the second is less than first,the whole operation is repeated for the new number on the second track.