CS 3452 - THEORY OF COMPUTATION

UNIT I AUTOMATA AND REGULAR EXPRESSIONS

Need for automata theory - Introduction to formal proof – Finite Automata (FA) – Deterministic Finite Automata (DFA) – Non-deterministic Finite Automata (NFA) – Equivalence between NFA and DFA – Finite Automata with Epsilon transitions – Equivalence of NFA and DFA- Equivalence of NFAs with and without ε-moves- Conversion of NFA into DFA – Minimization of DFAs.

PART-A

1. Define hypothesis. R

The formal proof can be using deductive proof and inductive proof. The deductive proof consists of sequence of statements given with logical reasoning in order to prove the first or initial statement. The initial statement is called hypothesis.

2. What is structural induction?

To prove a property of the elements of a recursively defined set, we use structural induction.

Basis Step: Show that the result holds for all elements specified in the basis step of the recursive definition.

Inductive Step: Assume that the property holds for the elements currently in the recursively defined set.

Show that it is true for each of the rules used to construct new elements in the recursive step of the definition.

3. What is proof by contradiction?

The method of proof by contradiction is to assume that a statement is not true and then to show that that assumption leads to a contradiction. A good example of this is by proving that $\sqrt{2}$ is irrational.

4. Define deductive proof.

Deductive proof consists of a sequence of statements whose truth leads from some initial statement, called the hypothesis to a conclusion statement. Each step in the proof must follow some accepted logical principle, from either the given facts or some previous in the deductive proof or a combinations of these.

5. Define Set, Infinite and Finite Set.

Set is Collection of various objects. These objects are called the elements of the set.
Eg : A = { a, e, i, o, u }
Infinite Set is a collection of all elements which are infinite in number.
Eg: A = {a | a is always even number}
Finite Set is a collection of finite number of elements. **Eg : A = { a, e, i, o, u }**

6. Give some examples for additional forms of proof. U

1. Proofs about sets
2. Proofs by contradiction
3. Proofs by counter examples.

7. Prove 1+2+3+… ..... +n= n(n+1)/2 using induction method. A

Consider the two step approach for a proof by method of induction
1. Basis of induction:
Let n = 1 then LHS = 1 and RHS = 1 + 1 / 2 = 1 Hence LHS = RHS.
2. Induction hypothesis:
To prove 1 + 2 + 3 …… + n = n (n + 1) / 2 + (n + 1)

Consider n = n + 1

then 1 + 2 + 3 ……+ n + (n + 1) = n (n + 1) / 2 + (n + 1)

= n2 + 3n + 2 / 2

= (n + 1) (n + 2) / 2

Thus it is proved that 1 + 2 + 3 …… + n = n (n + 1) / 2

8. Write down the operations on set. U

i) A U B is Union Operation

If A = { 1, 2, 3 } B = { 1, 2, 4 } then A U B = { 1, 2, 3, 4 }

i.e. combination of both the sets.

ii) A ∩ B is Intersection operation

If A = { 1, 2, 3 } B = { 1, 2, 4 } then A U B = { 2, 3 }

i.e. Collection of common elements from both the sets.

iii) A – B is the difference operation

If A = { 1, 2, 3 } B = { 1, 2, 4 } then A - B = { 3 }

i.e. elements which are there in set A but not in set B.

9. Write any three applications of Automata Theory. U

1. It is base for the formal languages and these formal languages are useful of the programming languages.
2. It plays an important role in complier design.
3. To prove the correctness of the program automata theory is used.
4. In switching theory and design and analysis of digital circuits automata theory is applied.
5. It deals with the design finite state machines.

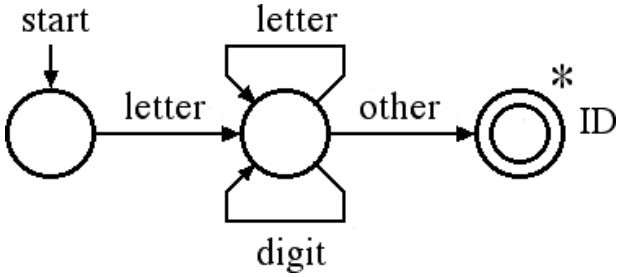
10. Define i. Finite automaton (or) What is a finite automaton?
ii. Transition diagram

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,∑,δ,q0,F), where Q is the finite set of states , ∑ is a finite input alphabet, q0 in Q is the initial state, F is the set of final states and δ is the transition mapping function Q * ∑ to Q.

Two types: **Deterministic Finite Automata (DFA)**
Non-Deterministic Finite Automata (NFA)

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.

11. Draw transition diagram for an identifier.

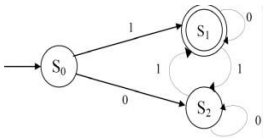


12. Define Deterministic Finite Automata.

The finite automata are called DFA if there is **only one path for a specific input from current state to next state.**

A finite automata is a collection of 5 tuples (Q, Σ. δ, q0, F)
where Q is a finite set of states, which is non-empty.
Σ is a input alphabet, indicates input set.
δ is a transition function or a function defined for going to next state.

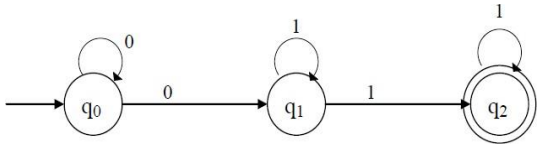
q0 is an initial state (q0 in Q)
F is a set of final states.



13. Define Non-Deterministic Finite Automata.

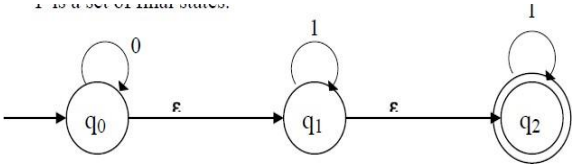
The finite automata are called NFA when there exists **many paths for a specific input from current state to next state.**

A finite automata is a collection of 5 tuples (Q, Σ. δ, q0, F)
where Q is a finite set of states, which is non-empty.
Σ is a input alphabet, indicates input set.
δ is a transition function or a function defined for going to next state.
q0 is an initial state (q0 in Q)
F is a set of final states.



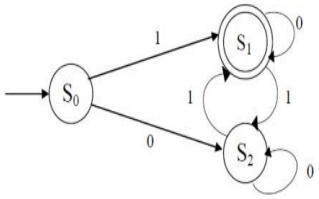
14. Define NFA with Є transition.

The Є **is a character** used to indicate null string. i.e the string which is used simply for transition from **one state to other state without any input.**

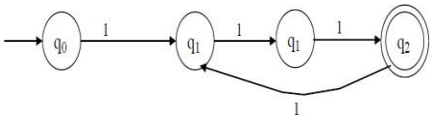


A Non Deterministic finite automata is a collection of 5 tuples (Q, Σ. δ, q0, F)
where Q is a finite set of states, which is non-empty.
Σ is a input alphabet, indicates input set.
δ is a transition function or a function defined for going to next state.
q0 is an initial state (q0 in Q)
F is a set of final states.

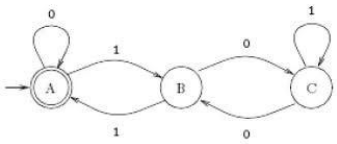
15. Design FA which accepts odd number of 1's and any number of 0's.



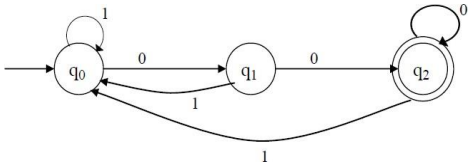
16. Design FA to check whether given unary number is divisible by three.



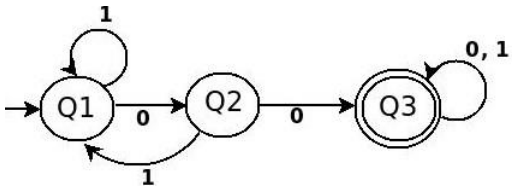
17. Design FA to check whether given binary number is divisible by three.



18. Design FA to accept the string that always ends with 00. C

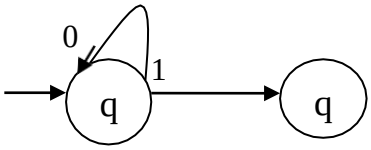
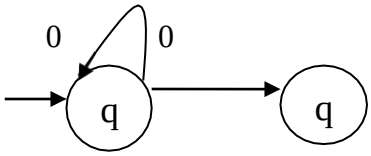


19. Design DFA to accept the strings over {0,1} with two consecutive 0's.



20. State the difference between NFA & DFA

S.NO	DFA	NFA
1	For each input symbol there is exactly one transition out of each state.	For each input symbol there is one or more transition from a state on the same input symbol.
2	It doesn't allow ξ moves	It allows ξ moves
3	1. δ(q,ξ) =q 2. δ(q,wa) = δ(δ`(q,w),a)	1. δ(q,ξ) =q 2. δ(q,wa) = δ(δ`(q,w),a) 3. δ(p,x) = U δ(q,w) q in p
4	Every DFA can simulate as NFA	NFA can't simulate as DFA
5	Transition function mapping from Q×∑ to Q.	Transition function mapping from Q×∑ to 2^Q.

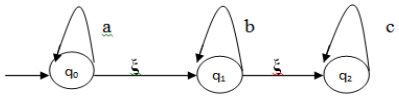
6 7	DFA stands for Deterministic finite automata. 	NFA stands for Non deterministic finite automata. 

21. Define the term Epsilon(€) transition.

The € **is a character** used to indicate null string. i.e the string which is used simply fr transition from **one state to other state without any input.**

22. Define ξ –Closure (q) **with an example.**

ξ –Closure (q) defines the set of all vertices p such that there is path from q to p labeled ξ.

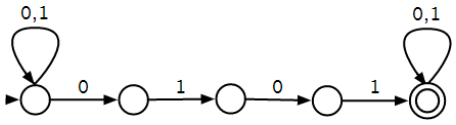


Solution:

ξ -**closure**(q_0) = {q_0,q_1,q_2}

ξ -**closure**(q_1) = {q_1,q_2}

23. Draw a NFA to accept strings containing the substring 0101.



PART B

1. Prove the following by induction for all n≥0
   $1^2+2^2+3^2+4^2+\ldots\ldots+n^2=(n(n+1)(2n+1))/6$ii. $1^3+2^3+\ldots.+n^3=(n^2(n+1)^2)/4$
2. Prove the following by mathematical induction method:
   i. $2^n>n$ for all n≥0
   ii. $X\geq 4, 2^x\geq x^2$
3. Prove by Induction that
   i. $n^3+(n+1)^3+(n+2)^3$ is divisible by 3 and n>0
   ii. $S(n)=a^n-b^n$ is divisible by a-b for all n>0
4. Prove
   i. $S(n)=5^{2n}-1$ is divisible by 24 for n>0
   ii. $1+2+\ldots\ldots+n=(n(n+1))/2$

5. i. Design DFA to accept the Languag L={w/w has both even number of 0‘s and even number of 1‘s}

   ii. Construct DFA that accepts input string of 0‘s and 1‘s that end with 11.

   Construct DFA for the set of all strings {0,1} with strings ending with 01.

   Construct DFA for the Language L={0^n/n mod 3=2,n≥0}

   Construct DFA for all the set of strings with {0,1} that has three consecutive 1‘s.

6. i. Construct an NFA for the set of strings with {0,1} ending with 01 draw the transition table for the same and check whether the input string 00101 is accepted by above NFA.

   ii. Construct NFA for set of all strings {0,1} that ends with three consecutive 1‘s at its end.

   iii. Construct NFA for set of all strings {a,b} with abb as substring.

7. If a Regular language _L‘ is accepted by a Non – deterministic Finite automata then there exist a Deterministic Finite Automata that accepts _L‘

8. A Language _L‘ is accepted by some ε – NFA if and only if L is accepted by NFA without ε transition

9. Convert to a DFA, the following NFA

	a	b
p(start)	{p}	{p,q}
q	{r}	{r}
r	{ Φ }	{ Φ }

10. Convert the following NFA-with ε, to a NFA- without ε

	0	1	2	ε
q_0 (start)	$\{q_0\}$	{φ}	{φ}	$\{q_1\}$
q_1	{φ}	$\{q_1\}$	{φ}	$\{q_2\}$
* q_2	{φ}	{φ}	{q2}	{φ}

11. Convert the following NFA-with ε, to a NFA- without ε

	a	b	ε
q_0 (start)	$\{q_0\}$	{φ}	$\{q_1\}$
* q_1	{φ}	$\{q_1\}$	{φ}

12. Convert the following NFA-with ε, to a NFA- without ε

	a	b	c	ε
p(start)	{q}	{p}	φ	φ
q	{r}	Φ	{q}	φ
*r	Φ	Φ	φ	{r}

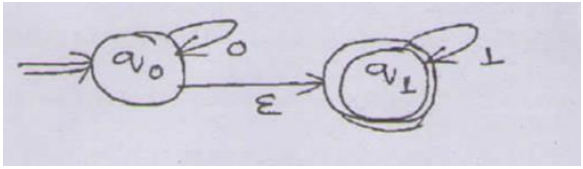
13. i. Prove that $\sqrt{2}$ is not rational.

ii. Construct a DFA accepting all strings w over {0,1} such that the number of 1‘s in w is 3 mod 4

14. Prove by induction on n that $\sum_{i=0}^{n} i = (n(n+1)) / 2$

15. Construct a DFA accepting binary strings such that the third symbol from the right end is 1

16. Construct NFA without ε transitions for the NFA given below



17. Construct an NFA accepting binary strings with two consecutive 0‘s.

18. Explain different forms of proof with examples.

19. i. Prove that if n is a positive integer such that n mod 4 is 2 or 3 then n is not a perfect square.

ii. Construct a DFA that accept the following language.

$\{x_{\in \{a,b\}} : |x|_a$=odd and $|x|_b$= even.

20. i. Construct DFA to accept the language L= { w / w I of even length and begins with 11}

Give FA accepting the following language over the alphabet

C

i. Number of 1‘s is a multiples of 3

ii. Number of 1‘s is not a multiples of 3

UNIT II REGULAR EXPRESSIONS AND LANGUAGES

Regular expression – Regular Languages- Equivalence of Finite Automata and regular expressions – Proving languages to be not regular (Pumping Lemma) – Closure properties of regular languages.

PART A

1. **Differentiate regular expression and regular language** (Or)
   What is regular expression?

Regular Expression	**Regular Language**
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 $\sum$ be an alphabet. The regular expression over $\sum$ 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 $\sum$, 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.	A language is regular if it is accepted by some finite automaton.

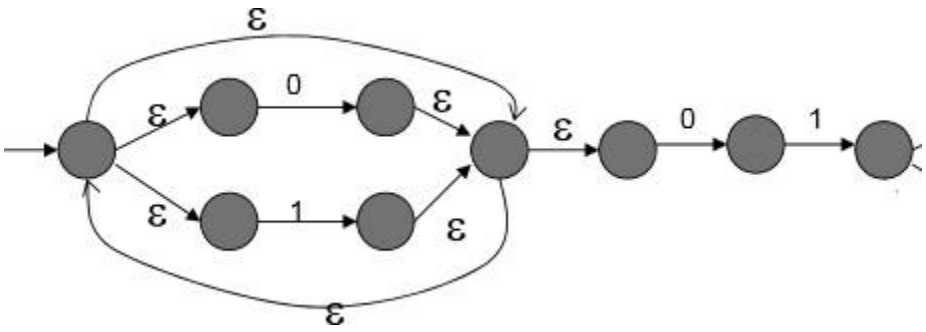
2. **Give the regular expression for set of all strings ending in 00.**
   R.E= $(0+1)^*00$

3. **Give the regular expression for the following L1= set of all strings of 0 and 1 ending in 00**
   L2= set of all string 0 and 1 beginning with 0 and ending with 1
   R1= (0+1)*00
   R2=0(0+1)*1

4. **Name any four CFG. U**
    - Union of two regular language is regular.
    - Concatenation of regular language is regular.
    - Closure of regular language is regular.
    - Complement of regular language is regular.
    - Intersection of regular language is regular.
    - Difference of regular language is regular.
    - Reversal of regular language is regular.
    - Homomorphism of regular language is regular.
    - Inverse Homomorphism of regular language is regular.

5. **Is regular set is closed under complement? Justify.**

6. Construct NFA for the regular expression (0+1)01



7. Prove or disprove that (r+s)*=r*+s*.

Replace r by *{a}* and s by *{b}*. The left side becomes all strings of *a*'s and *b*'s (mixed), while the right side consists only of strings of *a*'s (alone) and strings of *b*'s (alone). A string like *ab* is in the language of the left side but not the right.

8. Give English description of the following language (0+10)*1*.

Set of all strings of 0‘s and 1‘s including ξ

9. Write RE for the set of strings over {0,1} that have atleast one.

(0+1)*1(0+1)*

10. Show whether a language L=(0^n1^{2n}/n>0} is regular or not using pumping Lemma.

Suppose L is regular. We then have some $p>0$ and some $|m|>p$
a^pb^{2p}
$m=uvw$
and $|uv|\leq p$ and $uv^iw\in L$ for all $i>0$
As $|uv|\leq p$
then it follows that $v=a^l$. However, as $uviw\equiv a^{p+l}b^{2p}$ it shows that as $p+l\neq 2p$ therefore L is not regular.

PART-B

1. State and explain the conversion of DFA into R.E using Arden‘s theorem. Illustrate with an example.

2. i. Define regular expression. ii. Show that

(1+00*1)+(1+00*1)(0+10*1)*(0+10*1)=0*1(0+10*1)* A

3. Obtain minimized finite automata for the R.E (b/a)*baa. A

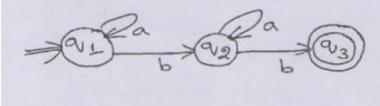
4. Prove that there exists an NFA with €- transition that accepts the regular expression r.

5. Which of the following language is regular? Justify.

i. L={ a^nb^m/n,m>0}

ii. L={ a^nb^n/n,>0}

6. Obtain the regular expression for the finite automata.



7. i. Using pumping lemma for the regular sets, prove that the language $L=\{a^m b^n / m>n\}$ is not regular.

ii. Prove any two closure properties of regular languages.

8. Construct a minimized DFA from R.E 0*(01)(0/111)*.

9. Discuss on the relation between DFA and minimal DFA

10. i. Discuss on regular expression.

ii. Discuss in detail about the closure properties of regular languages.

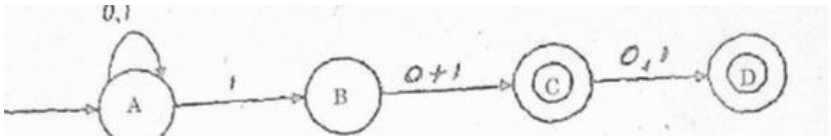
11. Prove that the following languages are not regular

i. $\{0^{2n}/n>0\}$

ii. $\{a^m b^n a^{m+n}/m>0 \text{and } n>0\}$

12. Discuss on equivalence and minimization of automata.

13. Convert the following NFA into a R.E



14. i. Design a FA for the R.E (0+1)*(00+11)(0+1*) ii. Prove that $L=\{0^{i^2} / \text{ i is an integer; } i>0\}$ is not regular.

Prove that the class of regular sets is closed under complementation.

15. Construct a minimized DFA for the RE 10+(0+11)0*1. C

16. Explain the DFA minimization algorithm with an example.

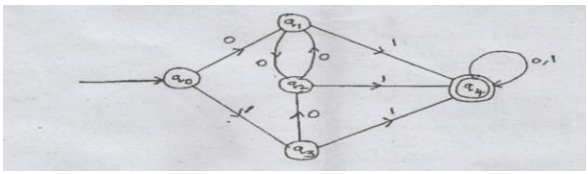
17. Find the min- state DFA for (0+1)*10.

18. Find the regular expression of a language that consists of set of strings with 11 as well as ends with 00 using Rij formula.

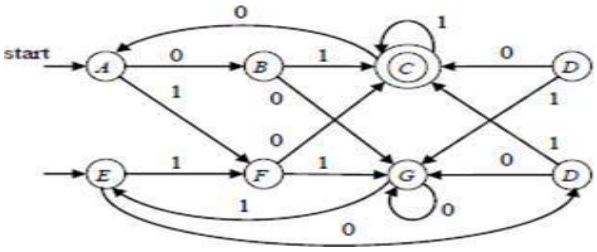
19. Construct FA equivalent to the regular expression(ab+a)*

20. What is Regular Expression? Write a regular expression for set of strings that consists of alternating 0's and 1's.

21. Minimize the FA shown in fig below and show both the given and the reduced one are equivalent



22. Write and explain the algorithm for minimization of a DFA. Using the above algorithm minimize the following DFA.

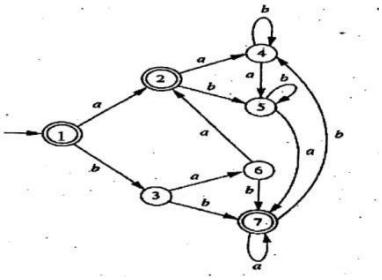


23. Construct NFA with epsilon for the R.E=(a/b)*ab and convert into DFA and further find the minimized DFA.

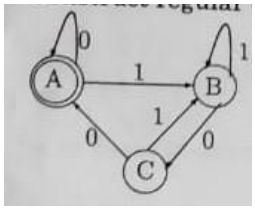
24. Show that the regular language are closed under :

- Union
- Intersection
- Kleen closure
- Complement
- Difference

25. Minimize the following automaton



26. Construct RE for



27. Prove that any language accepted by a DFA can be represented by regular expression. E Nov-Dec 2019

28. Construct a finite automata for the RE 10+(0+11)0*1

UNIT III

CONTEXT FREE GRAMMAR AND PUSH DOWN AUTOMATA

Types of Grammar - Chomsky_s hierarchy of languages -Context-Free Grammar (CFG) and Languages – Derivations and Parse trees – Ambiguity in grammars and languages – Push Down Automata (PDA): Definition – Moves - Instantaneous descriptions -Languages of pushdown automata – Equivalence of pushdown automata and CFG-CFG to PDA-PDA to CFG – Deterministic Pushdown Automata

1. Define CFG .Give an example.

This is the way of describing language by recursive rules called production. It consists of set of variables, set of terminal symbols, and a starting variable as well as the production. G = (V,T,P,S) Where V = variables, T = Terminals, P = productions, S = starting variable.

Eg 1: $G = (V,T,P,S)$ $V = \{E\}$ $T = \{+,*,\mathrm{id}\}$ $S = \{E\}$

E => E+E E => E*E E => id

2. What is CFL?

If grammar G = (V,T,P,S) be a context free grammar then the language L(G) is a set of terminal strings that have derivation from the starting symbol.

$$L(G) = \{ w \text{ in } T / S \overset{*}{\underset{G}{\Rightarrow}} w \}$$

- The language generated by the CFG is called Context Free Language.

Ex: Find L(G) for the following grammar.

a) S => aSb / ab

S=> aSb

=> aaSbb ∵ S=>aSb

=> aaaSbbb

=> aaaaSbbbb ∵ S=>ab

3. What is derivation?

It is defined as $\alpha \overset{*}{\underset{G}{\Rightarrow}} \beta$ where β is derived from the symbol ‗α' with the grammar ‗G'.

Here, we use the production from head to body (i.e.) from starting root node expanding until it reaches the given input string.

(i.e.) R.N => w

Eg: w=01C10

4. What are the 2 types of derivation? Right most derivation:

A derivation in which the right most variable is replaced at each step then, the derivation method is called right most derivation.

Eg:

E => E+E

E => E+E*E ∴ E=>E*E

E => E+E*id ∴ E=>id

E => E+id*id

∴ $E \overset{*}{\underset{G}{\Rightarrow}} id+id*id$

5. What is parse tree (or) derivation tree?

Parse tree is a pictorial representation of derivation, where the interior nodes are labeled by variables (or) non-terminals and leaf nodes are labeled by terminals symbols.

Eg:

w = 01C10

E => 0E0

E => 1E1

E => C

Derivation:

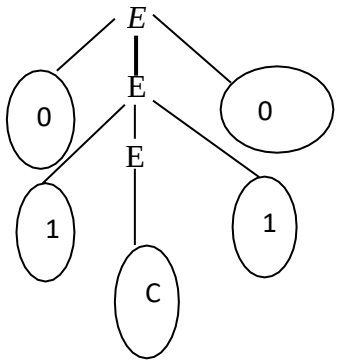
E => 0E0

=>01E10 ∵ E => 1E1

=> 01C10 ∵ E => C

$\therefore E \overset{*}{\Rightarrow} 01C10$

Parse tree (or) derivation tree:



6. What is ambiguous grammar? Or When do you say grammar is ambiguous?

A grammar that produces more than one parse tree (or) derivation tree for some sentence, then the grammar is said to be an ambiguous grammar. An ambiguous grammar produces more than one left most derivation (or) more than 1 RMD then, the given grammar is set to be an ambiguous grammar.

7. For the grammar defined by the productions recognize the string 1001 **and also construct the parse tree.**

S => A,B

A =>0A/ξ

B => 0B/1B/ξ

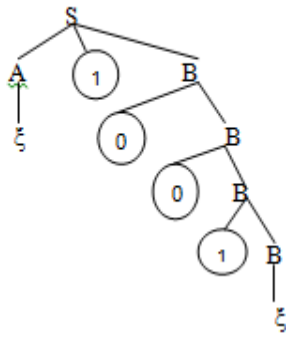
8. **Consider the alphabet Σ = { a,b,(,),+,*,-, . ,ξ }. Construct a CFG that generate all the strings in Σ* that are regular expression on the alphabet, Σ. C**

E => E+E
E => E*E
E => (E)
E => E.E
E => E-E
E => a / b / ξ

w => 1001

S => A1
=>A10B ∵ B => 0B
=> A100B
=> A1001B ∵ B => 1B
=> ξ1001B ∵ A => ξ
=> 1001ξ ∵ B => ξ

PARSE TREE:



9. **Find LMD & RMD, parse tree for the following grammar. A**

w = 00110101

S => 0B / 1A

A => 0/0S/1AA

B => 1/1S/0BB

R.M.D:

S => 0B

=>0BB ∵ B => 0BB

=> 00B1 ∵ B => 1

=> 001S1 ∵ B => 1S

=> 0011A1 ∵ B => 1A

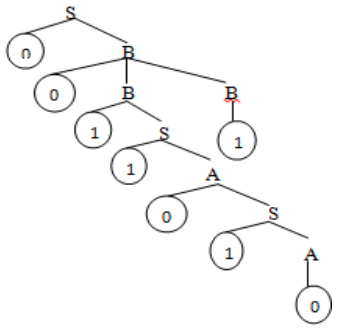
=> 00110S1 ∵ A => 0S

=> 001101A1 ∵ S => 1A

=> 00110101 ∵ A => 0

∴ $S \underset{rmd}{\overset{*}{\Rightarrow}} 00110101$

PARSE TREE:



L.M.D:

S => 0B

=> 0BB ∵ B => 0BB

=> 001B ∵ B =>1

=> 0011S ∵ B => 1S

=> 00110B ∵ S => 0B

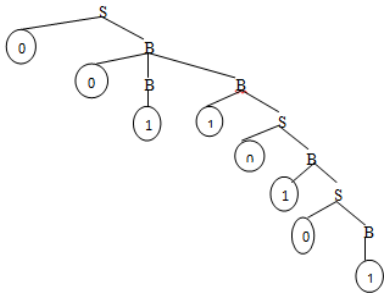
=> 001101S ∵ B => 1S

=> 0011010B ∵ S => 0B

=> 00110101 ∵ B => 1

∴ $S \underset{lmd}{\overset{*}{\Rightarrow}} 00110101$

PARSE TREE:



10. Define sentential form The string's are derived from the starting non-terminal is called sentential form.

If grammar G=(V,T,P,S) is a context free grammar, then α in (VUT)* such that non-terminal δ derives α is a sentential form.

$S \underset{rmd}{\overset{*}{\Rightarrow}} \alpha$ then α is left sentential form.
$S \underset{lmd}{\overset{*}{\Rightarrow}} \alpha$ then α is right sentential form.

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

Solution: S → aCa

→ aaCaa C→ aCa . . .

→ a^nCa^n

→ a^nba^n C → b

L (G) = { a^nba^n ; n >0}

12. Write a grammar to recognize all prefix expressions involving all binary arithmetic operators. Construct the parse tree for the sentence "-*+abc/de" from your grammar.

A **Nov/Dec 2006**

E => -EE
E => *EE
E => +EE
E => /EE
E => a / b / c / d / e

E => -EE

=>-*EEE ∵ E => *EE

=> -*+EEEE ∵ E => +EE

=> -*+aEEE ∵ E => a

=> -*+abEE ∵ E => b

=> -*+abcE ∵ E => c

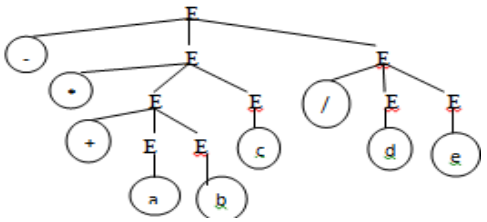
=> -*+abc/EE ∵ E => /EE

=> -*+abc/dE ∵ E => d

=> -*+abc/de ∵ E => e

∴ E => -*+abc/de

PARSE TREE:



13. Write the CFG for the following CFL L(G) = {$a^m b^n c^p$/ m+n=p, m&n>1}

C

E => aEc / bTc / a /bc

T => bTc / bc

E => aEc

=>aaEcc ∵ E => aEc

=> aabTccc ∵ E => bTc

=> aabbcccc ∵ T => bc ∴

$E \underset{LMD}{\overset{*}{\Rightarrow}} aabbcccc$

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

Solution: S → aCa
→ aaCaa C→ aCa . . .
→ a^nCa^n
→ a^nba^n C → b

L (G) = { a^nba^n ; n >0}

15. What is the language generated by the grammar G=(V,T,P,S) where P={S->aSb, S->ab}?..... AS=> aSb=>aaSbb=> ..... =>anbn

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

16. If S->aSb | aAb , A->bAa , A->ba .Find out the CFL

soln. S->aAb=>ababS->aSb=>a aAb b =>a a ba b b(sub S->aAb)
S->aSb =>a aSb b =>a a aAb b b=>a a a ba b bb
Thus L={anbmambn, where n,m>=1}

17. 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.

18. 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 | €

19. 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={ 0n 1 m | m not equal to n, and n,m >=1}

20. Construct the grammar for the language L={ a^n b a^n | 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)

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

P:{ S -> aSa ,
S-> b S b,
S-> a,
S->b,
S->€ } which is in palindrome.

22. 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.

23. 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)

24. 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.

PART-B

1. What is deterministic PDA? Explain with an example.
2. Is NPDA and DPDA equivalent? Illustrate with an example.
3. (i) Construct the PDA for the Language L= {WCWR | W is in (0+1)*.
4. Let L is a context free language. Prove that there exists a PDA that accepts L. Construct the PDA accepting the language { $(ab)^n/n>0$} by empty stack.
5. a. Construct a transition table for PDA which accepts the language L={ $(a^{2n}b^n/n>0$} Trace your PDA for the input with n=3.
   b. Find the PDA equivalent to the give CFG with the following productions.
   S→A A→BC B→ba C→ac
6. Construct PDA for the Language L= {WW^R | W is in (a+b)*}.
7. Construct the PDA accepting the language L= { $a^nb^n/n>0$} by empty stack and final state.
8. Convert the grammar S→0S1/A: A→1A0/S/ ε into PDA that accepts the same language by empty stack. Check whether 0101 belongs to N(M).
9. Prove that If L is $N(M_1)$ (the language accepted by empty stack) for some PDA M_1, then L is $N(M_2)$ (the language accepted by final state) for some PDA M_2.
10. What are the different types of language acceptances by a PDA and define them. Is it true that the language accepted by PDA by these different types provides different languages?
11. Convert the grammar S→aSb/A, A→bSa/S/ε to PDA that accepts the same language by empty stack.
12. Discuss the equivalence between PDA and CFG.
13. Construct a PDA for the language L={x€{a,b}*/ $n_a(x)>n_b(x)$}
14. Convert the following CFG to a PDA. S→aAA, A→aS|bS|a
15. Design a PDA to accept {0n1n|n>1}. Draw the transition diagram for the PDA. Show by instantaneous description that the PDA accepts the strings‘0011‘.
16. Convert PDA to CFG.PDA is given by P=({p,q},{0,1},{X,Z}, δ,q,Z), δ is defined by δ(p,1,Z)={(p,XZ)}, δ(p, ε,Z)={(p, ε)}, δ(p,1,X)={(p,XX)}, δ(q,1,X)={(q, ε,)}, δ(p,0,X)={(q,X)}, δ(q,0,Z)={(p,Z)}.
17. What are deterministic PDA‘s? Give example for non-deterministic and deterministic PDA.
18. What is an instantaneous description that the PDA? How will you represent it? Also give three important principles of ID and their transactions.
19. Explain acceptance by final state and acceptance by empty stack of a Push down Automata.
20. Outline an ID of a PDA.

21. With an example, explain the procedure to obtain a PDA from the given CFG.
22. Construct a DPDA for even length palindrome.
23. Prove –If PDA P is constructed from CFG G by the above construction, then N(P)=L(G)‖.
24. Convert the CFG to PDA and Verify for (a+b) and a++E→I/E+E/E*E/(E)
25. Find PDA that accept the given CFGS→XaaX
    X→aX/bX/ ε
26. Construct PDA for the language $a^n b^m a^{n+m}$.

UNIT IV

NORMAL FORMS AND TURING MACHINES

Normal forms for CFG – Simplification of CFG- Chomsky Normal Form (CNF) and Greibach Normal Form (GNF) – Pumping lemma for CFL – Closure properties of Context Free Languages –Turing Machine : Basic model – definition and representation – Instantaneous Description – Language acceptance by TM – TM as Computer of Integer functions – Programming techniques for Turing machines (subroutines).

PART A

1. **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.

2. **What are the closure properties of CFG?**

   Union : If L1 and If L2 are two context free languages, their union L1 ∪ L2 will also be context free.

   Concatenation : If L1 and If L2 are two context free languages, their concatenation L1.L2 will also be context free.

   Kleene Closure : If L1 is context free, its Kleene closure L1* will also be context free.

   Intersection and complementation : If L1 and If L2 are two context free languages, their intersection L1 ∩ L2 need not be context free.

3. **State the pumping lemma for CFL.R**

   Let L be a CFL then there exist a constant M such that if Z is any word in language L and $|Z| \geq n$ then we may write the above statements.

   By pumping lemma,

   $Z = UVWXY \quad |Z| >= n$

   $|VWX| \leq n$

   $|VX| >= 1$

   $UV^iWX^iY \in L$ For all $i \geq 0$

4. Give the steps to eliminate useless symbols.

1. Find the non-generating variables and delete them, along with all productions involving non-generating variables.
2. Find the non-reachable variables in the resulting grammar and delete them, along with all productions involving non-reachable variables.

5. Show that CFLs are closed under substitutions

If L is a Context – free language over alphabet £ ,and S is a substitution on £ such that S(a) is a CFL for each a in £ ,then S(L) is a CFL.

Proof:

The idea here is that for a CFG,replce each terminal a by the start symbol for language S(a).The result is a single CFG that generates S(L).

Let G =(V,£,P,S) be a grammer for L.

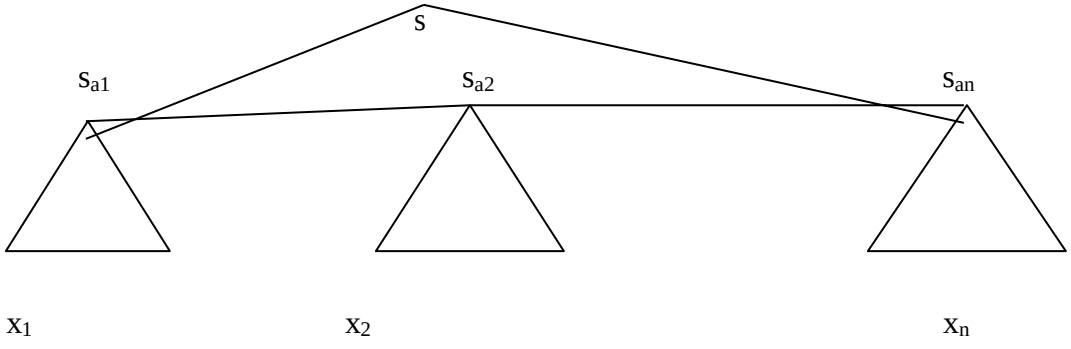
and $G_a = (V_a , T_a , P_a , S_a)$ be a grammer for each a in £.

Construct a new grammer $G^1 = (V^1,T^1,P^1,S)$ for S(L).

Where

- V^1 is the union of V and V_a .[for all a in £]
- T^1 is the union of all T_a.
- P^1 is given by
  - P_a for a in £.
  - P where each terminal a is replaced by S_a.

Thus all parse trees in grammer G_1 sart out with parse trees in G but all nodes have labels that are S_a for some a in £.Then the generation of each such node produces a parsertree of G_a whose yield belongs to S(a).



(Parse tree of G^1 due to substitution)

6. Show that L={a^p/ P is prime } is not context free.

Suppose that L be a context-free language and M be its corresponding Finite Automata with m number of states. Let w be a string which belongs to context-free language L with |w|=n where n is a prime number such that n ≥ m. Hence, w can be decomposed as w=uvwxy such that |vwx| ≤ n and |vx| > 0.

Since w ∈ L with |w|=n and w= uvwxy therefore we can get |uvwxy| = n (prime number). Means we may say that if length of a^n is prime no then it is a regular language **.** Now Since w = uvwxy ∈ L, then for L to be context-free uv^iwx^iy should also belong to L for every value of i. Then we can say that the length of uv^iwx^iy = $|uv^iwx^iy|$ should be a prime number.

$|uv^iwx^iy| = |uvwxy| + (i-1)*|vx|$

let |vx| = k where k > 0 and i = n+1 then
$|uv^iwx^iy| = |uv^iwx^iy| + (i-1)*|vx| = n + (n+1-1)*k = n+n*k = n*(1+k)$ => which is composite for i=p+1. Hence, uv^iwx^iy not belongs to L for all values of i. Therefore, L is not a context-free language.

7. List the closure properties of CFL.

- Substitutions
- Union
- Concatenation
- Closure and Positive Closure
- Homomorphism
- Reversal
- Intersection
- Inverse Homomorphism

8. Define Turing Machine.

The Turing machine is denoted by
M=(Q,Σ, ├,δ,q_0,B,F) Where Q –finite set of states
Σ -finite set of allowable tape symbols
a symbol of ├, a blank
Σ- set of input symbols
q_0∈Q- start state
F- set of final state
δ-Transition function mapping
Q x ├ →Q x ├ x{L,R}
Where L,R –Directions

9. What are the required fields of an instantaneous description or configuration of a TM?

It requires

- The state of the TM
- The contents of the tape
- The position of the tape head on the tape.

10. What is multiple tracks Turing machine?

A Turing machine in which the input tape is divided into multiple tracks where each track having different inputs is called multiple track Turing machine.

11. What is multidimensional Turing machine?

The Turing machine which has the usual 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 new symbol and moves its tape head in one of 2K directions along one of K axes.

12. When is a function f said to be Turing computable?

A Turing Machine defines a function y=f(x) for strings x,yЄΣ*, if

$q_0 x \vdash^* q_f y$ where q_0 – initial state ,q_f final state

A function f is _Turing computable' if there exist a Turing machine that perform a specific function.

13. What is off line Turing machine?

An off-line Turing machine is a multitape Tm whose input tape is read only. The Turing machine is not allowed to move the input tape head off the region between left and right end markers.

14. List out the different techniques for TM construction.

1. Storage in the finite control (or) State.
2. Multiple tracks.
3. Subroutines.
4. Checking off symbols

16. What is Universal Turing machine?

A universal Turing machine is a Turing machine Tu that works as follows.

It is assumed to receive an input string of the form e(T)e(z), where T is an arbitrary TM, z is a string over the input alphabet of T , and e is an encoding function whose values are

strings in {0, 1}*. The computation performed by Tu on this input string satisfies these two properties:

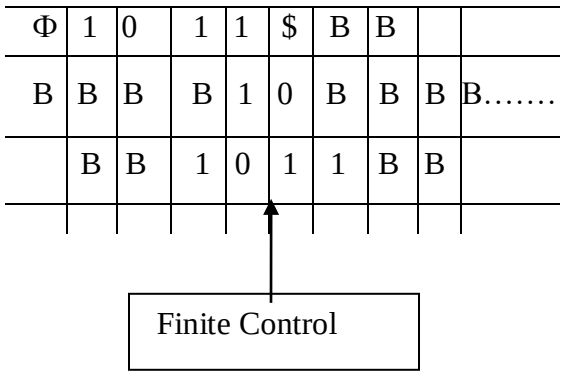
1. Tu accepts the string e(T)e(z) if and only if T accepts z.
2. If T accepts z and produces output y, then Tu produces output e(y).

17. Define multitape TM.

A **Multi-tape Turing machine** is like an ordinary Turing machine with several tapes. Each tape has its own head for reading and writing. Initially the input appears on tape 1, and the others start out blank.

A k-tape Turing machine can be described as a 6-tuple $M = \langle Q, \Gamma, s, b, F, \delta \rangle$ where:

- Q is a finite set of states
- Γ is a finite set of the tape alphabet
- $s \in Q$ is the initial state
- $b \in \Gamma$ is the blank symbol
- $F \subseteq Q$ is the set of final or accepting states
- $\delta : Q \times \Gamma^k \to Q \times (\Gamma \times \{L, R, S\})^k$ is a partial function called the transition function, where k is the number of tapes, L is left shift, R is right shift and S is no shift.



(A Three Track Turing Machine)

14. List the primary objectives of TM.

A Turing machine is an abstract machine that manipulates symbols on a strip of tape according to a table of rules; to be more exact, it is a mathematical model of computation that defines such a device. Despite the model's simplicity, given any computer algorithm, a Turing machine can be constructed that is capable of simulating that algorithm's logic.

15. What are the differences between a Finite automata and a Turing machine?

Finite Automata	Turing Machine
Finite Automation is a 5-tuple (Q, $\sum$,δ,q_0,F) where Q be a finite set of states $\sum$ be a finite set of symbols δ be a transition function mapping from Q X $\sum$ to Q q_0 the initial state and F the set of final state	A Turing Machine M is a 7-Tuple M = (Q,$\sum$,$\top$,δ,q_0,B,F) Where Q – finite set of states $\sum$ - finite set of input symbols $\top$ - finite set of tape symbols. δ – Transition function mapping the states of finite automaton and tape symbols to states,tape symbols and movement of the head. i.e., Q x $\top$ -> Q x $\top$ x {L,R} q_0 $\sum$ Q is the intial state F $\leq$ Q is the set of final states. B $\sum$ $\top$ is the blank symbol.

16. What is halting problem. R

In computability theory, the halting problem is the problem of determining, from a description of an arbitrary computer program and an input, whether the program will finish running or continue to run forever.

17. **Write short note on chomskian hierarchy of languages.**

- Chomsky Hierarchy is a broad classification of the various types of grammar available
- These include Unrestricted grammar, context-free grammar, context-sensitive grammar and restricted grammar
- Grammars are classified by the form of their productions.
- Each category represents a class of languages that can be recognized by a different automaton

18. Give the configuration of Turing Machine.

A Turing machine is an ordered pair of the current state and the tape contents with the symbol currently under the head marked with underscore. For example (q , aab**a**bb) shows that the Turing machine is currently in state q, the taper contents are the string aababb and the head is reading the last a of the string.

We write (p , x**a**y) |- (q , z**b**w) if the Turing machine goes from the first configuration to the second in one move, and (p , x**a**y) |-* (q , z**b**w) if the Turing machine goes from the first configuration to the second in zero or more moves.

19. Differentiate multihead and multi tape Turing machine. U Nov/Dec 2018

Multihead TM	Multi tape TM
A multi-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.	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.

20. What are the advantages of having a normal form for a grammar?

While PDAs can be used to parse words with any **grammar**, this is often inconvenient. **Normal forms** can give us more structure to work with, resulting in easier parsing algorithms.

PART B

1. Is the language L = $\{a^n b^n c^n \mid n>=1\}$ is context free? Justify. (Or) Show that the Language L=$\{ a^i b^i c^i / i \geq 1\}$ is not context free. A

2. Discuss the closure properties of CFL.
3. Show that language $\{0^n1^n2^n/n>=1\}$ is not CFL.
4. State the pumping lemma for CFL. Use pumping lemma to show that the language L= $\{a^ib^jc^k/i<j<k\}$ is not a CFL.
5. State and explain the pumping Lemma for CFG.
6. Explain pumping Lemma for CFL.
7. Convert the following grammar into GNF S→XY1/0, X→00X/Y,Y→1X1

8. Construct the following grammar in CNF

$A \rightarrow BCD \mid b$

$B \rightarrow Yc \mid d$

$C \rightarrow gA \mid c$

$D \rightarrow dB \mid a$

$Y \rightarrow f.$

9. Construct the following grammar in CNF

$S \rightarrow cBA,\ S \rightarrow A,\ A \rightarrow cB,\ A \rightarrow AbbS,\ B \rightarrow aaa.$

10. Find GNF for the grammar

$S \rightarrow AA \mid \perp$

$A \rightarrow SS \mid \theta$

11. Construct a equivalent grammar G in CNF for the grammar G1 where G1=({S,A,B}, {a,b}, { S→ASB/£, A→aAS/a, B→SbS/A/bb},S).

12. Given the CFG G, find CFG G‘ in CNF generating the language L(G)-{^}

S→AACD

A→aAb/^

C→aC/a

D→aDa/bDb/^

13. Construct a reduced grammar equivalent to the grammar G = (N, T, P, S) where, N = {S. A, C, D, E} T= {a, b} C

P = { S —> aAa, A —> Sb, A —> bCC, A —> DaA, C —> abb. C —> DD, E→ aC, D —> aDA}. What is the purpose of normalization? Construct the CNF and

$S \rightarrow aAa \mid bBb \mid \varepsilon$

$A \rightarrow C \mid a$

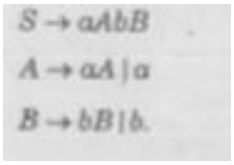
$B \rightarrow C \mid b$

$C \rightarrow CDE \mid \varepsilon$

$D \rightarrow A \mid B \mid ab$

GNF for the followinggrammar and explain the steps.

14. Obtain a Grammar in CNF



UNIT V UNDECIDABILITY

Unsolvable Problems and Computable Functions –PCP-MPCP- Recursive and recursively enumerable languages – Properties - Universal Turing machine -Tractable and Intractable problems - P and NP completeness – Kruskal‘s algorithm – Travelling Salesman Problem- 3-CNF SAT problems.

PART A

1. **When a language is said to be recursively enumerable?**
   A language is recursively enumerable if there exists a Turing machine that accepts every string of the language and does not accept strings that are not in the language.

2. **Define Non Recursive language.**
   If the languageL is not recursively enumerable, then there is no algorithm for listing the members of L. It might be possible to define L by specifying some property that all its members satisfy, but that property can't be computable.

3. **When a language is said to be recursive?**
   A language L is said to be recursive if there exists a Turing machine M that accepts L, and goes to halt state or else M rejects L.

4. **Define decidable problems.**
   A problem is said to be decidable if there exists a Turing machine which gives one _yes‘ or _no‘ answer for every input in the language.

5. **Define undecidable problems.**
6. **If a problem is not a recursive language, then it is called undecidable problem.**
7. **Define universal language.**

   A universal Turing machine M_u is an automaton, that given as input the description of any Turing machine M and a string w, can simulate the computation of M on w.

8. **Define problem solvable in polynomial time.**

   A Turing machine M is said to be of time complexity T(n) if whenever m is given an input w of length n, m halts after making at most T(n) moves, regardless of whether or not m accepts.

9. **Define the class P and NP.**

   P consists of all those languages or problems accepted by some Turing machine that runs in some polynomial amount of time, as function of its input length.

   NP is the class of languages or problems that are accepted by nondeterministic TM‘s with a polynomial bound on the time taken along any sequence of non – deterministic choices.

10. **Define NP – Complete Problem.**

    A language L is NP – complete if the following statements are true.

    (i)L is in NP.

    (ii)For every language L^1 in NP there is a polynomial – time reduction of L^1 to L.

11. **Write the Significance of NP-Complete Problem.**

    NP-complete languages are significant because all NP-complete languages are thought of having similar hardness, in that process solving one implies that others are solved as well. If some NP-complete languages are proven to be in P, then all of NPs are proven to be in P.

12. **What are tractable problems?**

    The problems which are solvable by polynomial – time algorithm are called tractable problems. For Eg. The complexity of the Kruskal‘s algorithm is 0(e(e+m)where e ,the number of edges and m,the number of nodes.

13. **What are the properties of recursively enumerable sets which are undecidable?**

    1.Emptiness

    2.Finiteness

    3.Regularity

    4.Context – freedom.

14. What are the properties of recursive and recursively enumerable language?

(i) The complement of a recursive language is recursive.

(ii) The union of two recursive languages are recursive the union of two recursively enumerable languages are recursively enumerable.

(iii)If a language L and L‘ are both recursively enumerable, Then L is recursive.

15. Show that any PSPACE-hard language is also NP-hard.

16. Mention the difference between P and NP problems.

P problems	**NP problems**
P consists of all those languages or problems accepted by some Turing machine that runs in some polynomial amount of time, as function of its input length.	NP is the class of languages or problems that are accepted by nondeterministic TM‘s with a polynomial bound on the time taken along any sequence of non – deterministic choices.

PART B

1. Prove that _If _L‘ is a recursive language, then L‘ is also a Recursive Language‘.
2. Prove that _If a language L and L‘ are recursively enumerable (RE) , then L is Recursive‘.
3. Prove that (i) Lu is recursively enumerable but not recursive.
   (ii) Non empty language Lne is recursively enumerable.
4. Find the languages obtained from the following operations:
   (i) Union of two recursive languages. (6)
   (ii) Union of two recursively enumerable languages (6)
   (iii) L if L and complement of L are recursively enumerable (4)
5. a) Show that the following language is not decidable. E
   L={<M>| M is a TM that accepts the string aaab}. (8)
   b) Discuss the properties of Recursive and Recursive enumerable languages. U (8)
6. Prove that the universal language Lu is recursively enumerable. E
7. Define the universal language and show that it is recursively enumerable but not recursive. U
8. Whether the problem of determining given recursively enumerable language is empty or not? Is decidable? Justify your answer. AN

9. Define Universal language Lu. Show that Lu is recursively enumerable but not recursive. U
10. Explain the Halting problem. Is it decidable or undecidable problem? U Nov/DEC 2011 , Nov/Dec 2012
11. Explain the difference between tractable and intractable problems with example

12. Write short notes on: i. Recursive and recursively enumerable language ii. NP hard and NP complete Problems
13. Discuss the properties of recursive languages.
14. Explain any two undecidable problems with respect to TM.
15. Discuss the difference between NP-complete and NP-hard problems.

16. Write note on NP problems.
17. Explain about ―A language that is not Recursively Enumerable‖.
18. Prove that for two recursive language L1 and L2 their union and intersection is recursive.
19. Explain post correspondence problems and decidable and undecidable problems with examples.
20. Explain the class P and NP problems with suitable example.
21. Prove that ―MPCP reduce to PCP‖.
22. Discuss about the tractable and intractable problems.
23. State and explain RICE theorem.
24. Describe about Recursive and Recursively Enumerable languages with examples.
25. What is Universal Turing Machine? Bring out its significance. Also construct a TM to add two numbers and encode it.
26. What is post correspondence problem (PCP) Explain with the help of an example.
27. Elaborate on primitive recursive functions with an example.
28. Compare recursive language with recursively enumerable languages.
29. What are tractable problems? Compare with intractable problems.