DHANALAKSHMI SRINIVASAN ENGINEERING COLLEGE
(AUTONOMOUS)
(Approved by AICTE & Affiliated to Anna University, Chennai)
Re-Accredited by NAAC with 'A' Grade
Accredited by NBA for AERO, BME, CSE, ECE, EEE, IT & MECH.
PERAMBALUR-621212, TAMILNADU, INDIA.
Website: www.dsengg.ac.in

DEPARTMENT OF INFORMATION TECHNOLOGY

U23CBT51- THEORY OF COMPUTATION

2 MARKS & 16 MARKS

UNIT I

AUTOMATA FUNDAMENTALS
Introduction to formal proof – Additional forms of Proof – Inductive Proofs –Finite Automata –Deterministic Finite Automata – Non-deterministic Finite Automata – Finite Automata with Epsilon Transitions

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. Define inductive proof.R Nov/Dec 2010 (Or) State the principle of induction Nov/Dec 2012

This is a very powerful and important technique for proving theorems.

For each positive integer n, let P(n) be a mathematical statement that depends on n.

Assume we wish to prove that P(n) is true for all positive integers n.

A proof by induction of such a statement is carried out as follows:

Basis: Prove that P(1) is true.

Induction step: Prove that for all $n \geq 1$, the following holds: If P(n) is true, then P(n + 1) is also true.

In the induction step, we choose an arbitrary integer $n \geq 1$ and assume that P(n) is true; this is called the induction hypothesis. Then we prove that P(n + 1) is also true.

3. What is structural induction? R May/June 2011

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.

4. What is proof by contradiction? R May/June 2012

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.

5. Define deductive proof. R Nov/Dec 2014

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.

6. Define Set, Infinite and Finite Set. R

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 }**

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

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

8. 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

9. 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.

10. 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.

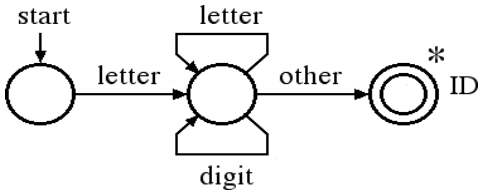
11. Define i. Finite automaton (or) What is a finite automaton? R
ii. Transition diagram May/June 2013, Nov/Dec 2012,2015 & 2017

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.

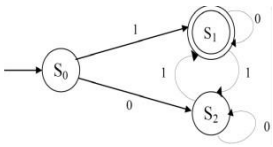
12. Draw transition diagram for an identifier. A Nov/Dec 2013



13. Define Deterministic Finite Automata. R May/June 2013, Nov-Dec 2016,2019

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

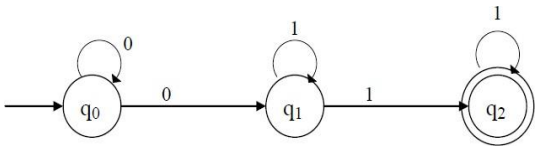
A finite automata is a collection of 5 tuples (Q, Σ. δ, q0, F) where Q is a finite set of states, which is non-empty.
Σ is a input alphabet, indicates input set.
δ is a transition function or a function defined for going to next state.
q0 is an initial state (q0 in Q) F is a set of final states.



14. Define Non-Deterministic Finite Automata. R Nov/Dec 2013

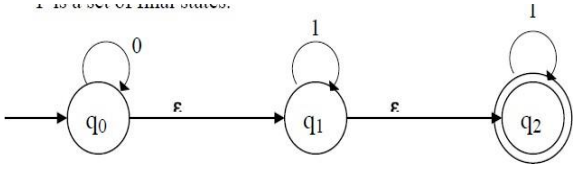
The finite automata are called NFA when there exists **many paths for a specific input from current state to next state.**

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.



15. Define NFA with Є transition. R

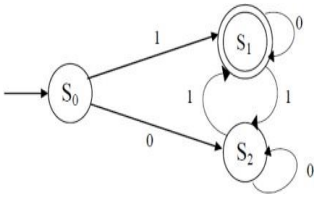
The Є **is a character** used to indicate null string. i.e the string which is used simply for transition from **one state to other state without any input.**



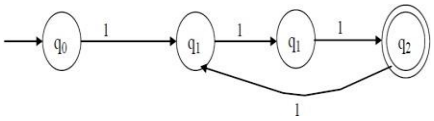
A Non Deterministic finite automata is a collection of 5 tuples (Q, Σ. δ, q0, F) where Q is a finite set of states, which is non-empty.
Σ is a input alphabet, indicates input set.
δ is a transition function or a function defined for

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

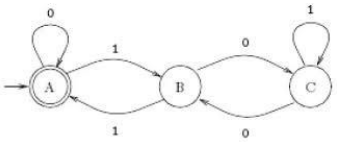
16. Design FA which accepts odd number of 1's and any number of 0's. C



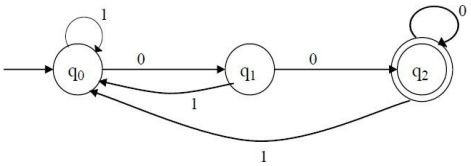
17. Design FA to check whether given unary number is divisible by three. C



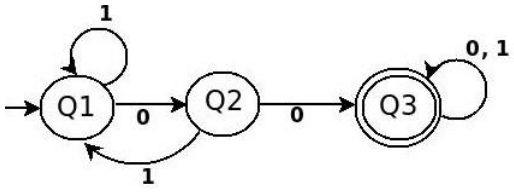
18. Design FA to check whether given binary number is divisible by three. C



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



20. Design DFA to accept the strings over {0,1} with two consecutive 0's. C Nov/Dec 2014



21. State the difference between NFA & DFA. AN May/June 2011& May/June 2014

Nov/Dec 2018

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

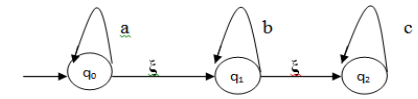
3	1. $\delta(q,\xi) = q$ 2. $\delta(q,wa) = \delta(\delta`(q,w),a)$	1. $\delta(q,\xi) = q$ 2. $\delta(q,wa) = \delta(\delta`(q,w),a)$ 3. $\delta(p,x) = \cup_{q \text{ in } p} \delta(q,w)$
4	Every DFA can simulate as NFA	NFA can't simulate as DFA
5	Transition function mapping from $Q \times \sum$ to Q.	Transition function mapping from $Q \times \sum$ to 2^Q.

22. Define the term Epsilon(Є) transition. R May/June 2013

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.**

23. Define ξ –Closure (q) **with an example. R May/June 2012, Nov/Dec 2022**

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

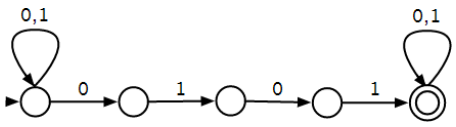


Solution:

ξ -**closure**(q_0) = {q_0,q_1,q_2}

ξ -**closure**(q_1) = {q_1,q_2}

24. Draw a NFA to accept strings containing the substring 0101. C May-June 2016



25. Define Extended Transition Function.

Extended Transition Function (δ):

The extended transition function, denoted as δ(q, w), defines the state reached by a finite automaton when starting from state **q** and processing the entire input string **w**.

It is defined recursively as:

- δ(q, ε) = q (for the empty string)
- δ(q, xa) = δ(δ(q, x), a), where **x** is a string and **a** is a symbol in the input alphabet.

PART B

1. Prove the following by induction for all $n \geq 0$
   May/June 2014

i. $1^2+2^2+3^2+4^2+\ldots\ldots+n^2=(n(n+1)(2n+1))/6$ May/June 2016

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 $n \geq 0$
   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. $1+2+\ldots\ldots+n=(n(n+1))/2$ **Nov/Dec 2022**
5. i. Design DFA to accept the Language.
   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 $L=\{0^n/n \bmod 3=2, n \geq 0\}$
   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'
   Nov/Dec 2013&Nov/Dec 2014
8. A Language _L' is accepted by some ε – NFA if and only if L is accepted by NFA without ε transition A May/June 2012 & Nov/Dec 2013
9. Convert to a DFA, the following NFA A May/June 2013

	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)	$\{q_0\}$	$\{\varphi\}$	$\{\varphi\}$	$\{q_1\}$
q_1	$\{\varphi\}$	$\{q_1\}$	$\{\varphi\}$	$\{q_2\}$

* 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 C Nov/Dec 2011

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

15. Construct a DFA accepting binary strings such that the third symbol from the right end is 1 C May/June 2012

DEPARTMENT OF INFORMATION TECHNOLOGY

U23CBT51- THEORY OF COMPUTATION

2 MARKS & 16 MARKS

UNIT II

REGULAR EXPRESSIONS AND LANGUAGES
Regular Expressions – FA and Regular Expressions – Proving Languages not to be regular – ClosureProperties of Regular Languages – Equivalence and Minimization of Automata.

PART A

1. Differentiate regular expression and regular language AN Nov/Dec 2012

(Or)

What is regular expression? May/June 2013

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 ∑ be an alphabet. The regular expression over ∑ 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 ∑ , 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. C Nov/Dec 2010

R.E= $(0+1)^*00$

3. State pumping lemma for regular language. R Nov/Dec 2022 Nov/Dec 2010, 2013 , 2014 & 2017 , May-June 2016, Nov/Dec 2018,2019

Let L be regular language then there exist a constant n (Number of states that accept the language L) such that if W is the word or set of input string in the language L then,

1. Z = UVW
2. |UV| ≤ n
3. |V| >= 1
4. UViW Є L For all i ≥ 0

4. Give the regular expression for the following C Nov/Dec 2012

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

5. Name any four CFG. U May/June 2013& May/June 2014 Nov-Dec 2016

- 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.

6. Is regular set is closed under complement? Justify. U May/June 2012

If L is a regular language over alphabet Σ then $\overline{L} = \Sigma^* \setminus L$ is also regular.

Proof: Let L be recognized by a DFA

$$A = (Q, \Sigma, \delta, q_0, F).$$

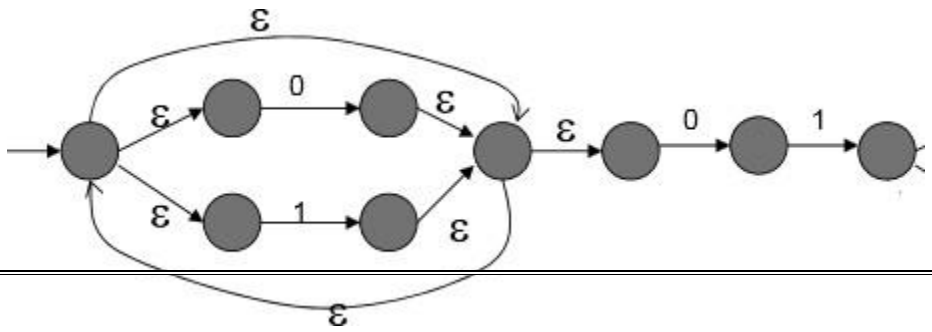
Then $\overline{L} = L(B)$ where B is the DFA

$$B = (Q, \Sigma, \delta, q_0, Q \setminus F).$$

That is, B is exactly like A, but the accepting states of A have become the nonaccepting states of B and vice-versa.

Then w is in $L(B)$ iff $\hat{\delta}(q_0, w)$ is in $Q \setminus F$, which occurs iff w is not in $L(A)$.

7. Construct NFA for the regular expression (0+1)01 C Nov/Dec 2013



8. Prove or disprove that (r+s)*=r*+s*. Nov/Dec 2014

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.

9. Give English description of the following language (0+10)*1*. April/May 2015

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

Write RE for the set of strings over {0,1} that have atleast one. Nov/Dec 2015

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

10. Show whether a language L=(0^n1^{2n}/n>0} is regular or not using pumping Lemma. May-June 2017

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 $uviw=a^{p+l}b^{2p}$ it shows that as $p+l\neq 2p$
therefore L is not regular.

11. What is a Finite Automaton (FA)?

A Finite Automaton is a mathematical model used to recognize regular languages. It consists of a finite number of states, an input alphabet, a transition function, a start state, and one or more accepting (final) states. As the input string is read symbol by symbol, the automaton transitions between states according to its transition function. If it ends in an accepting state after reading the entire input, the string is accepted.

12. Define ε-transition in NFA-ε.

An ε-transition is a transition that allows an NFA to move from one state to another without consuming any input symbol. It is useful in simplifying the construction of automata by enabling transitions that happen "for free." These transitions help in recognizing optional components within strings. During conversion to DFA, ε-transitions are removed using ε-closure.

13. What is the language denoted by (a+b)*?

The regular expression (a+b)* represents the set of all strings, including the empty string, that are made up of any combination of the symbols 'a' and 'b'. This includes strings like ε, a, b, ab, ba, aa, bb, and so on. It is essentially the Kleene star applied to the union of 'a' and 'b'. This language includes every possible string over the alphabet {a, b}.

14. State the identity: (r*)* = r*.

The identity (r*)* = r* means that applying the Kleene star multiple times has the same effect as applying it once. This is because r* already includes all possible concatenations of r, including the empty string. So repeating the star operation does not introduce any new strings. It is a simplification rule used in regular expression algebra.

15. Give regular expression for strings containing exactly one '1'.

The regular expression that describes all strings containing exactly one '1' is 0*10*. This means that any number of 0s can appear before and after a single 1. Strings such as "1", "01", "100", and "0010" match this pattern. No other 1 is allowed in the string apart from the one in the middle.

16. What is the significance of initial state in FA?

The initial state in a finite automaton is the state where processing of the input string begins. It serves as the starting point of the computation. From this state, the automaton follows transitions based on input symbols. Every automaton has exactly one initial state, and it is essential in determining the path of string processing.

17. What is meant by final state in FA?

A final state, also called an accepting state, is one where the finite automaton ends after processing an input string to accept it. If the automaton finishes reading the input and is in a final state, the string is said to be accepted. These states are usually denoted by double circles in state diagrams. An automaton may have more than one final state.

18. What do you mean by equivalence of two DFAs?

Two deterministic finite automata (DFAs) are said to be equivalent if they accept exactly the same set of strings, that is, the same language. For any string given as input, either both DFAs accept it or both reject it. This concept is used to verify whether two different automata define the same language. It also plays an important role in DFA minimization.

19. Mention one method to prove a language is not regular.

The most common method to prove a language is not regular is using the pumping lemma. This lemma provides a property that all regular languages must satisfy. If we can find a string in the language that violates this property, then the language cannot be regular. It is a proof by contradiction approach widely used in formal language theory.

20. Define union of two regular languages.

The union of two regular languages L_1 and L_2 is the set of strings that belong to either L_1 or L_2 or both. If both L_1 and L_2 are regular, their union is also regular, which shows closure under union. This can be expressed using a regular expression as $r_1 + r_2$, where r_1 and r_2 are expressions for L_1 and L_2. This operation is fundamental in building complex expressions.

21. What is difference between reachable and unreachable states?

Reachable states in an automaton are those that can be accessed from the initial state by following transitions for some input string. Unreachable states, on the other hand, can

never be reached from the initial state, regardless of the input. These states do not contribute to the language accepted by the automaton. Removing them is part of the minimization process.

22. What does the expression (a+b)*a(a+b)* represent?

The regular expression (a+b)*a(a+b)* describes the set of all strings that contain at least one occurrence of the symbol 'a'. It allows any combination of a's and b's before and after the required 'a'. For example, "a", "ba", "ab", and "bab" are all accepted. The key condition is that at least one 'a' must appear anywhere in the string.

23. State whether the language L = {$a^n b^m$ | n ≥ m} is regular.

The language L = {$a^n b^m$ | n ≥ m} is not regular because a finite automaton cannot compare and count the number of a's and b's. To determine if n is greater than or equal to m requires memory, which finite automata lack. Using the pumping lemma, we can prove that L cannot be accepted by any regular automaton. Hence, L is not a regular language.

24. What is minimization of DFA?

Minimization of a DFA involves reducing the number of states while preserving the language it accepts. This is done by identifying and merging equivalent states—states that behave the same for all inputs. The result is the smallest possible DFA for a given regular language. Minimization improves the efficiency of pattern matching and recognition tasks.

25. Define homomorphism of regular language.

A homomorphism is a function that maps each symbol of the input alphabet to a string over another alphabet. If L is a regular language and h is a homomorphism, then the language h(L), formed by applying h to every string in L, is also regular. This demonstrates the closure of regular languages under homomorphism. It is useful in language transformation problems.

PART-B

1. State and explain the conversion of DFA into R.E using Arden's theorem. Illustrate with an example. A Nov/Dec 2011

2. i. Define regular expression. R Nov/Dec 2011

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 May/June 2012s

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

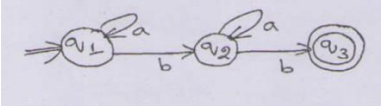
A May/June 2012

5. Which of the following language is regular? Justify. U May/June 2012

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

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

6. Obtain the regular expression for the finite automata. A May/June 2012



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. Nov/Dec 2012

8. Construct a minimized DFA from R.E 0*(01)(0/111)*. C Nov/Dec 2012

9. Discuss on the relation between DFA and minimal DFA U May/June 2013

10. i. Discuss on regular expression. U May/June 2013

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

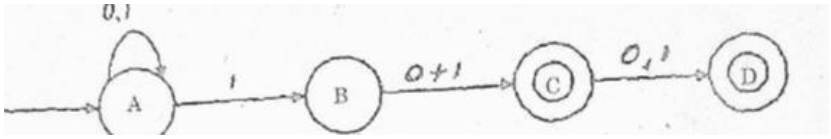
11. Prove that the following languages are not regular A May/June 2013

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

ii. $\{a^m b^n a^{m+n}/m>0 \text{and } n>0\}$ Nov/Dec 2013

12. Discuss on equivalence and minimization of automata. U May/June 2013

13. Convert the following NFA into a R.E C Nov/Dec 2013



14. i. Design a FA for the R.E (0+1)*(00+11)(0+1*) C May/June 2014

ii. Prove that $L=\{0^{i^2} / \text{ i is an integer; } i>0\}$ is not regular. An Nov/Dec 2014, 2015

15. Prove that the class of regular sets is closed under complementation. A May/June 2014

DEPARTMENT OF INFORMATION TECHNOLOGY

U23CBT51- THEORY OF COMPUTATION

2 MARKS & 16 MARKS

UNIT III

CONTEXT FREE GRAMMAR AND LANGUAGES
CFG – Parse Trees – Ambiguity in Grammars and Languages – Definition of the Pushdown Automata– Languages of Pushdown Automata – Equivalence of Pushdown Automata and CFG, Deterministic Pushdown Automata.

PART A

1. Define CFG .Give an example. **Nov-Dec 2016**

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

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

2. What is CFL? **May/June 2013**

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 $\alpha \underset{G}{\overset{*}{\Rightarrow}} \beta$ 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?

Left most derivation:

If at each step in derivation, a production is applied to the left most variable (or) left most non-terminal then the derivation method is called left most derivation.

Eg:

E w = id+id*id

E => E*E

E => id

E => E+E

E => id+E ∴ E=>id

E => id+E*E ∴ E=>E*E

E => id+id*E ∴ E=>id

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

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

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

E => E+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:

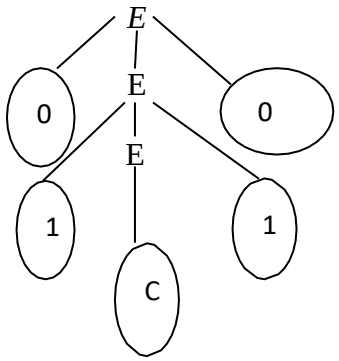
E => 0E0

=>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? R Nov/Dec 2012,2019, May/June 2013 , May/June 2014 & Nov/Dec 2022

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. A**

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**

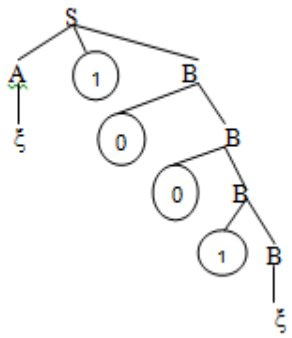
Nov/Dec 2007

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** **May/June 2007**

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

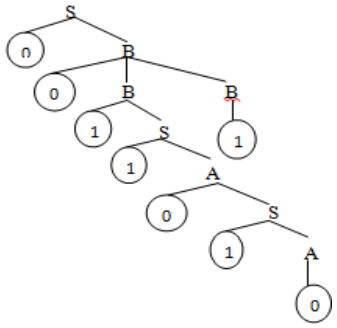
=> 00110S1 ∵ A => 0S

=> 001101A1 ∵ S => 1A

=> 00110101 ∵ A => 0

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

PARSE TREE:



L.M.D:

S => 0B

=> 0BB ∵ B => 0BB

=> 001B ∵ B =>1

=> 0011S ∵ B => 1S

=> 00110B ∵ S => 0B

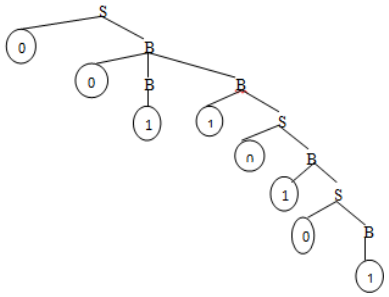
=> 001101S ∵ B => 1S

=> 0011010B ∵ S => 0B

=> 00110101 ∵ B => 1

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

PARSE TREE:



10. Define sentential form R

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

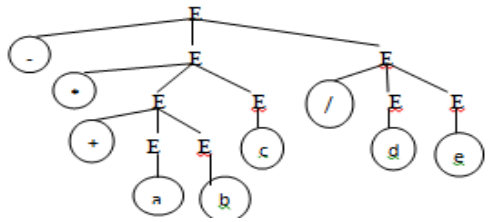
=> -*+abc/EE ∵ E => /EE

=> -*+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}

Nov/Dec 2006

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}?

S=> 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=>abab
S->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}. C May/June 2014 , Nov/Dec 2015

G=({S}, {a,b} , P, S)
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.

25. Give the general forms of CNF. (Or) State CNF. Nov/Dec 2014, Nov-Dec 2016

Every CFL is generated by a CFG in which all productions are of the form

A->BC
(or) A->a
Where A,B,C – variables
a – terminals

This form of CFG is called as Chomsky Normal Form

In order to find CNF,we need to perform the following operations.

1. Eliminate useless symbols i.e.,symbols or terminals which do not appear in any derivation of a terminal string from start symbol.
2. Eliminate €-productions which are of the form A->€ form some variable A.
3. Eliminate unit production which are of the form A->B for variables A and B.

PART-B

1. What is deterministic PDA? Explain with an example. U Nov/Dec 2010
2. Is NPDA and DPDA equivalent? Illustrate with an example. U Nov/Dec 2011
3. (i) Construct the PDA for the Language L= {WCWR | W is in (0+1)*. C Nov/Dec 2010, May/June 2012, Nov/Dec 2013
   (ii) Let L is a context free language. Prove that there exists a PDA that accepts L. A
4. Construct the PDA accepting the language { $(ab)^n/n>0$} by empty stack. C Nov/Dec 2012
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. C Nov/Dec 2012
   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)*}. C May/June 2013 & 2016
7. Construct the PDA accepting the language L= { $a^nb^n/n>0$} by empty stack and final state. C May/June 2014
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). A May/June 2014
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. A Nov/Dec 2014,2019
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? U Nov/Dec 2011
11. Convert the grammar S→aSb/A, A→bSa/S/ε to PDA that accepts the same language by empty stack. A Nov/Dec 2011
12. Discuss the equivalence between PDA and CFG. May/June 2012, May/June 2013, Nov/Dec 2013, May/June 2014
13. Construct a PDA for the language L={x€{a,b}*/ $n_a(x)>n_b(x)$} C April/May 2015
14. Convert the following CFG to a PDA. A Nov/Dec 2015

    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‘. C Nov/Dec 2015

DHANALAKSHMI SRINIVASAN ENGINEERING COLLEGE
(AUTONOMOUS)
(Approved by AICTE & Affiliated to Anna University, Chennai)
Re-Accredited by NAAC with 'A' Grade
Accredited by NBA for AERO, BME, CSE, ECE, EEE, IT & MECH.
PERAMBALUR-621212, TAMILNADU, INDIA.
Website: www.dsengg.ac.in

DEPARTMENT OF INFORMATION TECHNOLOGY

U23CBT51- THEORY OF COMPUTATION

2 MARKS & 16 MARKS

UNIT IV

PROPERTIES OF CONTEXT FREE LANGUAGES
Normal Forms for CFG – Pumping Lemma for CFL – Closure Properties of CFL – Turing Machines – Programming Techniques for TM.

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? **Nov/Dec 2017**

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
May/June 2012, Nov/Dec 2012, May/June 2014, April/May2015

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. **Nov/Dec 2017**

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 **Nov/Dec 2014**

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).

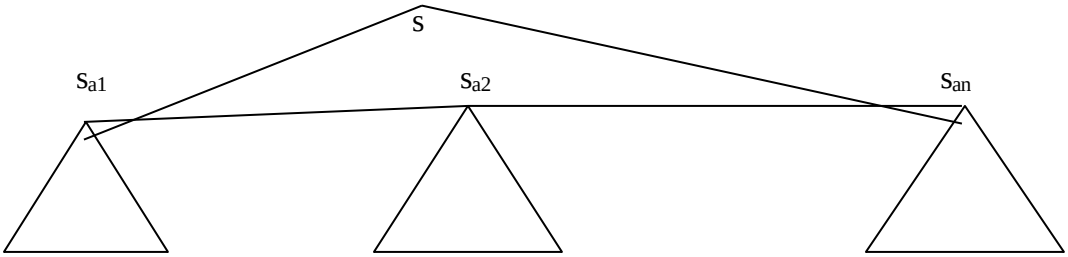
Let G =(V,£,P,S) be a grammer for L.

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).



6. Show that L={a^p/ P is prime } is not context free. **Nov/Dec 2017**

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 $n \geq m$. Hence, w can be decomposed as w=uvwxy such that $|vwx| \leq n$ and $|vx| > 0$.

Since $w \in L$ 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 $w = uvwxy \in L$, 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. R May/June 2013, Nov/Dec 2013, Nov/Dec 2022

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

8.Define Turing Machine. Nov/Dec 2010,2015& 2017 , May/June 2014 & 2016

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

8. What are the required fields of an instantaneous description or configuration of a TM? Nov-Dec2016

It requires

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

9. 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.

10. 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.

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

A Turing Machine defines a function y=f(x) for strings $x,y \in \Sigma^*$, 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.

12. 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.

13. List out the different techniques for TM construction. Nov/Dec 2013

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

16. What is Universal Turing machine? Nov/Dec 2013, 2016

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. Nov/Dec 2014, Nov/Dec 2015

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.

Φ	1	0	1	1	$	B	B		
B	B	B	B	1	0	B	B	B	B……
	B	B	1	0	1	1	B	B	

Finite Control

(A Three Track Turing Machine)

14. **List the primary objectives of TM. R** **Nov-Dec2016**

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? A May-June2103**

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** **May-June2107**

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.**
May-June2107 Nov/Dec 2018

- 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.** **Nov/Dec 2017**

A **configuration** for 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? U**

Nov/Dec 2019, Nov/Dec 2022 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.

21. **Define the language recognized by the TM. R Nov/Dec 2019**

A TM accepts a language if it enters into a final state for any input string w. A language is recursively enumerable (generated by Type-0 grammar) if it is accepted by a Turing machine. A TM decides a language if it accepts it and enters into a rejecting state for any input not in the language.

22. **When do you say a TM is an algorithm? U Nov/Dec 2019**

If an algorithm exists, then a turing machine can run it!.‖In other words, what all can be done by an algorithm can also be done by the Turing Machine.

23. **What is Chomsky Normal Form (CNF) in context-free grammar?**

Chomsky Normal Form is a type of CFG where all production rules are either of the form $A \rightarrow BC$ or $A \rightarrow a$, where A, B, C are variables and a is a terminal. CNF is useful for simplifying parsing algorithms and proving properties like the Pumping Lemma.

24. **State the Pumping Lemma for context-free languages.**

The Pumping Lemma for CFL states that for any CFL, there exists a constant p such that any string s of length $\geq p$ can be split as $s = uvwxy$, satisfying:

1. $|vwx| \leq p$, 2) $vx \neq \varepsilon$, and 3) $u(v^n)w(x^n)y \in L$ for all $n \geq 0$.

25. **Mention any two closure properties of CFLs.**

Context-Free Languages are closed under **union**, **concatenation**, and **Kleene star**, but **not closed** under **intersection** and **complement**. These closure properties help in analyzing and constructing complex languages from simpler ones.

PART-B

1. Is the language L = {$a^n b^n c^n$ | n>=1} is context free? Justify. AN Nov/Dec 2010
   (Or) Show that the Language L={ $a^i b^i c^i$/ i≥ 1} is not context free. A May/June 2014

2. Discuss the closure properties of CFL. U Nov/Dec 2010,May/June 2012, Nov/Dec 2012, May/June 2013
3. Show that language {$0^n1^n2^n$/n>=1} is not CFL. A Nov/Dec 2014& Nov/Dec 2015
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. A May-June2016
5. State and explain the pumping Lemma for CFG. U Nov-Dec2016
6. Explain pumping Lemma for CFL. U May- June 2017
7. Convert the following grammar into GNF A Nov/Dec 2013
   S→XY1/0, X→00X/Y,Y→1X1

8. Construct the following grammar in CNF **C**

$A \rightarrow BCD \mid b$

$B \rightarrow Yc \mid d$

$C \rightarrow gA \mid c$

$D \rightarrow dB \mid a$

$Y \rightarrow f.$

9. Construct the following grammar in CNF C Nov/Dec 2012
   S → cBA, S→ A, A → cB, A → AbbS, B→ aaa.
10. 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). C Nov/Dec 2015

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

    S→AACD

    A→aAb/^

    C→aC/a

    D→aDa/bDb/^ A April/May 2015

12. 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}. C May-June2016
13. What is the purpose of normalization? Construct the CNF and GNF for the following grammar and explain the steps. A May-June2016

S → aAa | bBb | ε
A→ C | a
B→ C | b
C→ CDE | ε
D→ A | B | ab

14. Given the CFG G, find CFG G‘ in CNF generaring the language L(G)- { ξ}

S→AACD C May- June 2017

A→aAb\ ξ

C→aC\a

D→aDa\bDb\ ξ

15. Convert the following grammar G into Greibach

Normal Form S→XA\BB C May- June 2017 B→b\SB

X→b, A→a

DHANALAKSHMI SRINIVASAN ENGINEERING COLLEGE
(AUTONOMOUS)
(Approved by AICTE & Affiliated to Anna University, Chennai)
Re-Accredited by NAAC with 'A' Grade
Accredited by NBA for AERO, BME, CSE, ECE, EEE, IT & MECH.
PERAMBALUR-621212, TAMILNADU, INDIA.
Website: www.dsengg.ac.in

DEPARTMENT OF INFORMATION TECHNOLOGY

U23CBT51- THEORY OF COMPUTATION

2 MARKS & 16 MARKS

UNIT V

UNDECIDABILITY
Non Recursive Enumerable (RE) Language – Undecidable Problem with RE – Undecidable Problemsabout TM – Post's Correspondence Problem, The Class P and NP.

PART A

1. **Define Non Recursive language.** **R** **Nov/Dec. 2022**
   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.

2. **When a language is said to be recursive?** **U**
   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.

3. **Define decidable problems.** **R**
   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.

4.

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

6. **Define universal language.** **R**
   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.

7. **Define problem solvable in polynomial time.** **R**
   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.

8. Define the class P and NP. R May/June 2013, 2014 & Nov-Dec 2019

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.

9. Define NP – Complete Problem. Nov-Dec2016

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.

10. Write the Significance of NP-Complete Problem. Nov-Dec2022

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.

11. What are tractable problems? Nov-Dec2017

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.

12. What are the properties of recursive and recursively enumerable language? Nov-Dec2017

(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.

13. Mention the difference between decidable and undecidable problems. Nov/Dec 2010

Decidable Problem	Undecidable Problem
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.	If a problem is not a recursive language, then it is called undecidable problem.

14. Mention the difference between P and NP problems. AN May/June 2012

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.

15. When we say a problem is decidable? Give example of undecidable problem. U Nov/DEC 2012, Nov/Dec 2015

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.
E.g Halting problem

16. Give examples for NP – Complete Problem. U Nov/Dec 2014

1. Complete sub graph problem is NP-complete.
2. The *k*-colorability problem is *NP*-complete.

17. Differentiate Recursive and Non-recursive language. AN April/May2015

A **recursive language** (also called decidable) is a language for which there exists a Turing machine that halts on all inputs and accepts strings that belong to the language and rejects others. A **non-recursive language**, on the other hand, is one for which no Turing machine can decide membership for all inputs—it may loop forever for some strings. Recursive languages are closed under union, intersection, and complementation, while non-recursive languages are not necessarily closed under these operations. Recursive languages are algorithmically solvable, whereas non-recursive ones are not always solvable by any algorithm.

18. When is a Recursively Enumerable language said to be Recursive? May-June2016

A language is Recursively Enumerable (RE) if some Turing machine accepts it. A TM M with alphabet _ accepts L if L = {w ∈ _*|M halts with input w }
Let L be a RE language and M the Turing Machine that accepts it., for w ∈ L, M halts in final state. For w /∈ L, M halts in non-final state or loops forever.
A language is Recursive (R) if some Turing machine M recognizes it and halts on every input string, w ∈ _*. Recognizable = Decidable. Or A language is recursive if there is a membership algorithm for it. Let L be a recursive language and M the Turing Machine that accepts (i.e. recognizes) it. For string w, if w ∈ L, then M halts in final state. If w /∈ L, then M halts in non-final state.

19. Identify whether 'Tower of Hanoi' problem is tractable or intractable. Justify your answer. May-June2016

'Tower of Hanoi' problem is intractable.
Intractable Problem: a problem that cannot be solved by a polynomial-time algorithm. The lower bound is exponential.
Towers of Hanoi: we can prove that any algorithm that solves this problem must have a

worst-case running time that is at least 2^n-1.

20. What is primitive recursive function? R May-June2107
Define the primitive recursion operation. R Nov-Dec 2018
Function is considered primitive recursive if it can be obtained from initial functions and through finite number of composition and recursion steps.

21. Define NP completeness. R May-June2107
A problem is NP-complete if answers can be verified quickly, and a quick algorithm to solve this problem can be used to solve all other NP problems quickly.

22. **What is a Non-Recursively Enumerable (Non-RE) language?**
A Non-RE language is a language for which no Turing machine can even semi-decide the membership of a string. That is, there exists no Turing machine that accepts all strings in the language and may not halt for others. These languages lie beyond the power of Turing machines to recognize. They are neither decidable nor semi-decidable.

23. **What is an Undecidable Problem with RE languages?**
An undecidable problem with RE languages is one for which the language is recursively enumerable, but no algorithm exists to always give a correct yes/no answer. An example is the Halting Problem, where the Turing machine may accept valid inputs but loop indefinitely on others. These problems are semi-decidable, not fully decidable.

24. **What are Undecidable Problems about Turing Machines?**
Undecidable problems about Turing machines involve questions that cannot be answered by any algorithm. Examples include determining whether a Turing machine halts for all inputs or whether two Turing machines accept the same language. These problems highlight the limits of what can be computed, even with powerful theoretical models.

25. **What is Post's Correspondence Problem (PCP)?**
Post's Correspondence Problem is an undecidable problem involving two lists of strings. The task is to find a sequence of indices such that the concatenation of strings from both lists (using those indices) results in the same string. There is no general algorithm to determine if a solution exists for arbitrary inputs, making it undecidable.

PART-B

1. Prove that _If _L' is a recursive language, then L' is also a Recursive Language'. E
2. Prove that _If a language L and L' are recursively enumerable (RE) , then L is Recursive'. E
3. Prove that (i) Lu is recursively enumerable but not recursive. E
   (ii) Non empty language Lne is recursively enumerable.
4. Find the languages obtained from the following operations: A
(i) Union of two recursive languages. (6) Nov/Dec 2014
(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)

6. b) Discuss the properties of Recursive and Recursive enumerable languages. U (8)
7. Prove that the universal language Lu is recursively enumerable. E
8. May/June 2014 , Nov/Dec 2014, Nov/ Dec 2015
9. Define the universal language and show that it is recursively enumerable but not recursive. U
10. Whether the problem of determining given recursively enumerable language is empty or not? Is decidable? Justify your answer. AN
11. Define Universal language Lu. Show that Lu is recursively enumerable but not recursive. U
12. Explain the Halting problem. Is it decidable or undecidable problem? U Nov/DEC 2011 , Nov/Dec 2012
13. Explain the difference between tractable and intractable problems with examples. U Nov/Dec 2010
14. Write short notes on: i. Recursive and recursively enumerable language
15. ii. NP hard and NP complete Problems U Nov/Dec 2011
16. Discuss the properties of recursive languages. U May/June 2012
17. Explain any two undecidable problems with respect to TM. U
18. May/June 2012 , May/June 2013
19. Discuss the difference between NP-complete and NP-hard problems. May/June 2012