DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Year / Sem : **III / V**
Sub. Code & Subject : **CS1303 - THEORY OF COMPUTATION** (TOC)

TWO MARK QUESTION & ANSWERS

Unit - I

1. **Define hypothesis.**

   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.**

   It is a recursive kind of proof which consists of sequence of parameterized statements that use the statement itself with lower values of its parameter.

3. **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 }

4. **Give some examples for additional forms of proof.**
   1. Proofs about sets
   2. Proofs by contradiction
   3. Proofs by counter examples.

5. **Prove 1+2+3+……+n= n(n+1)/2 using induction method.**

   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 $1 + 2 + 3 \ldots\ldots + n = n (n + 1) / 2 + (n + 1)$

      Consider n = n + 1

      then $1 + 2 + 3 \ldots\ldots + n + (n + 1) = n (n + 1) / 2 + (n + 1)$

      $= n^2 + 3n + 2 / 2$

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

      Thus it is proved that $1 + 2 + 3 \ldots\ldots + n = n (n + 1) / 2$

6. Write down the operations on set.

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**

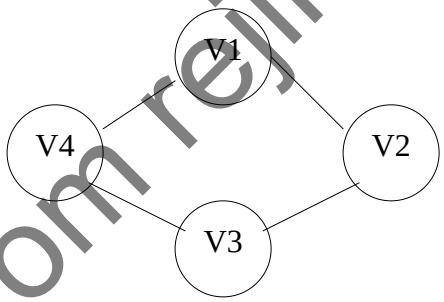
If A = { 1, 2, 3 } B = { 1, 2, 4 } then
A U B = { 1 }
i.e. elements which are there in set A but not in set B.

7. Define Graph, Directed graph and give example.

Graph is consists of finite set of Vertices (Node) V and set of Edges E, edges are nothing but pair of vertices.

It denoted G = (V, E)

Eg. :

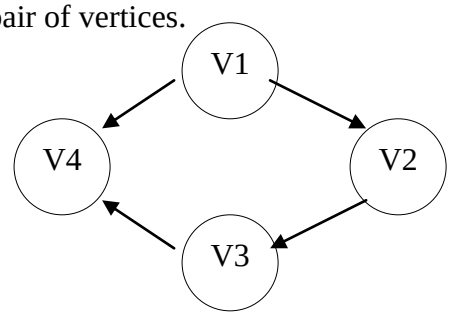


E1 is a edge connecting the vertices V1 and V2.

Directed Graph is consists of finite set of Vertices (Node) V and set of Edges E, edges are nothing but pair of vertices.

It denoted G = (V, E)

Eg.



The edge E1 shows the direction to V2 from V1.

8. Write any three applications of Automata Theory.

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.

9. Define Finite Automation.

A finite automata is a collection of 5 tuples (Q, Σ. δ, q_0, 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.

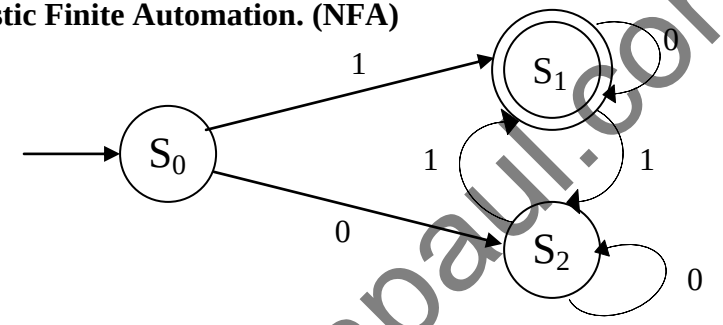
q_0 is an initial state (q_0 in Q)

F is a set of final states.

Two types :

Deterministic Finite Automation (DFA)

Non-Deterministic Finite Automation. (NFA)



10. Define Deterministic Finite Automation.

- The finite automata is 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, q_0, F)

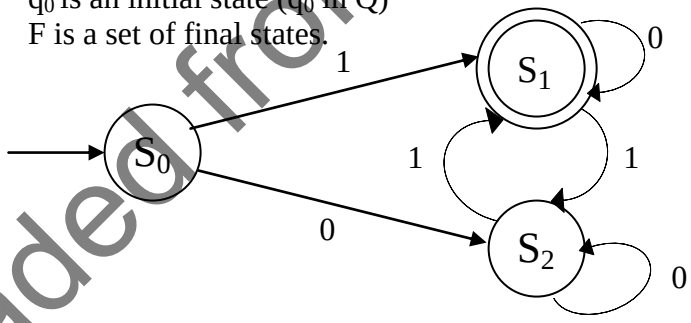
where Q is a finite set of states, which is non empty.

Σ is a input alphabet, indicates input set.

δ is a transition function or a function defined for going to next state.

q_0 is an initial state (q_0 in Q)

F is a set of final states.



11. Define Non-Deterministic Finite Automation.

The finite automata is 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, q_0, F)

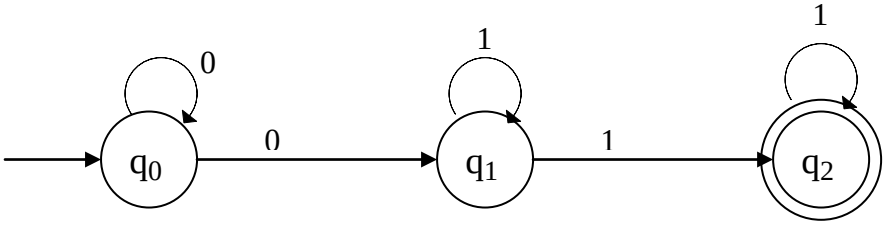
where Q is a finite set of states, which is non empty.

Σ is a input alphabet, indicates input set.

δ is a transition function or a function defined for going to next state.

q_0 is an initial state (q_0 in Q)

F is a set of final states.



12. Define NFA with ε transition.

The **ε 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.**

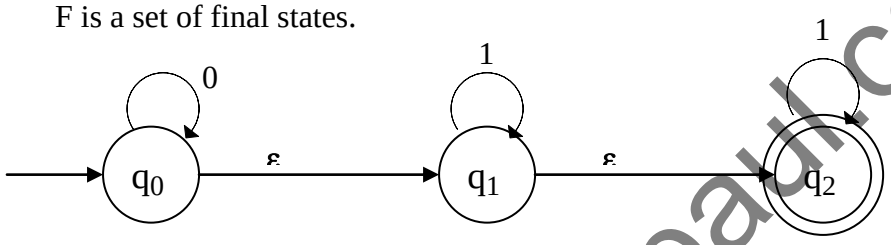
A Non Deterministic finite automata is a collection of 5 tuples (Q, Σ. δ, q_0, F) where Q is a finite set of states, which is non empty.

Σ is a input alphabet, indicates input set.

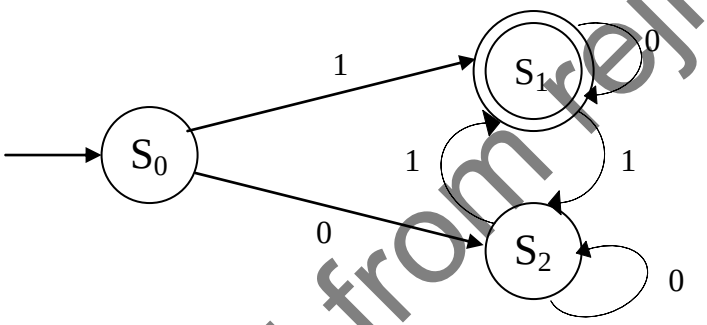
δ is a transition function or a function defined for going to next state.

q_0 is an initial state (q_0 in Q)

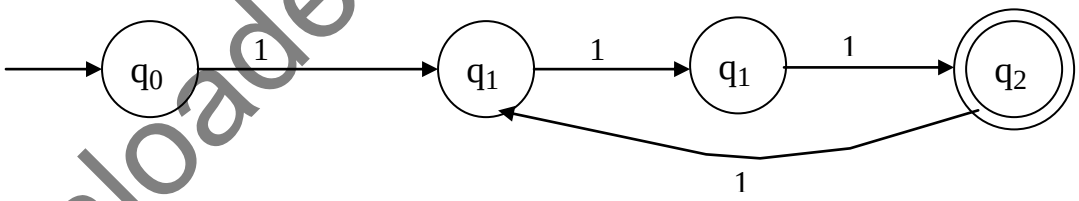
F is a set of final states.



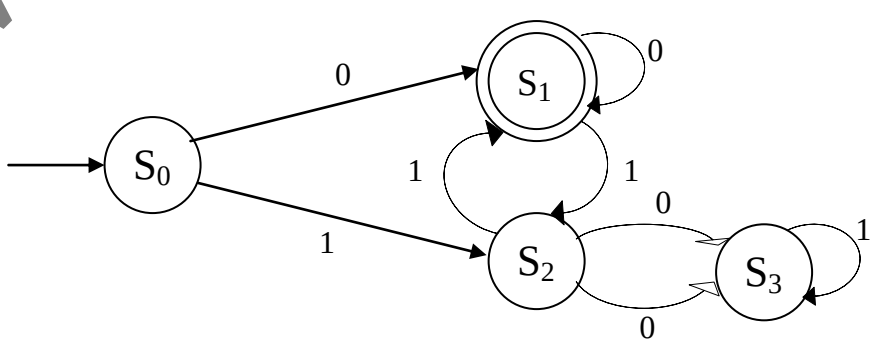
13. Design FA which accepts odd number of 1's and any number of 0's.



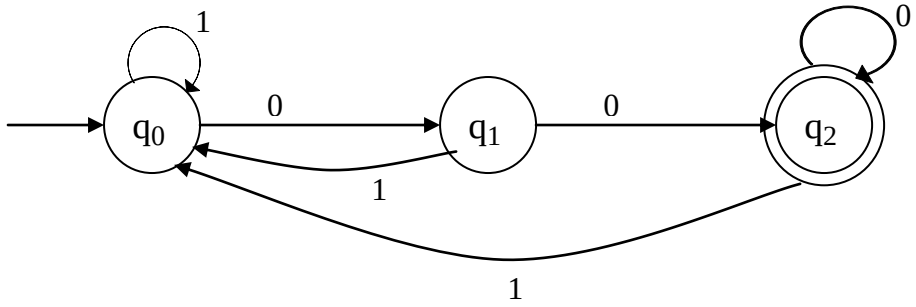
14. Design FA to check whether given unary number is divisible by three.



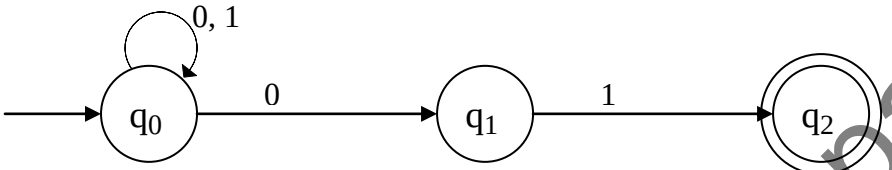
15. Design FA to check whether given binary number is divisible by three.



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



17. Obtain the DFA equivalent to the following NFA.



Solution :

The transition table for given **NFA** can be drawn as follows

States	Input	
	0	1
{q0}	{q0}{q1}	{q0}
{q1}	-	{q2}
{q2}	-	-

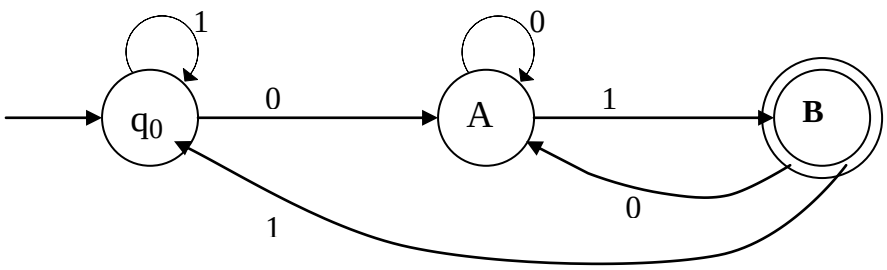
To construct equivalent DFA

δ(q0, 0) = {q0, q1 } a new state - **A**
δ{q0, 1} = {q0}
δ{q1, 0} = -
δ{q1, 1} = {q2}
δ{q2, 0} = -
δ{q2, 1} = -
δ{{qo,q1},0} = {q0,q1}
δ{{qo,q1},1} = {q0,q2} a new state - **B**
δ{{qo,q2},0} = {q0,q1}
δ{{qo,q2},0} = {q0}

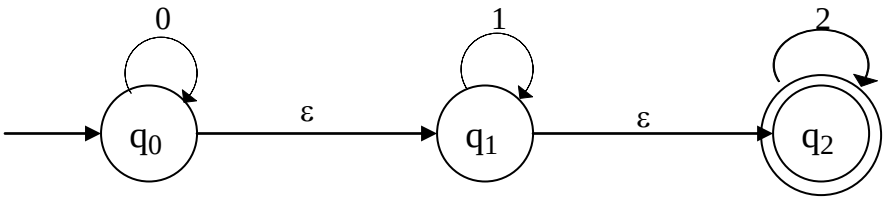
The transition table for DFA

States	Input	
	0	1
{q0}	{q0, q1}	{q0}
{q1}	-	{q2}
{q2}	-	-
{q0, q1}	{q0, q1}	{q0, q2}
{q0, q2}	{q0, q1}	{q0}

The transition diagram for DFA

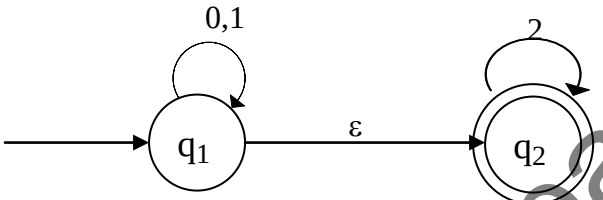


18. Obtain the NFA without ε transition to the following NFA with ε transition.

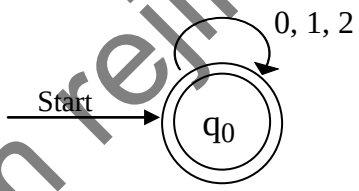


Solution :

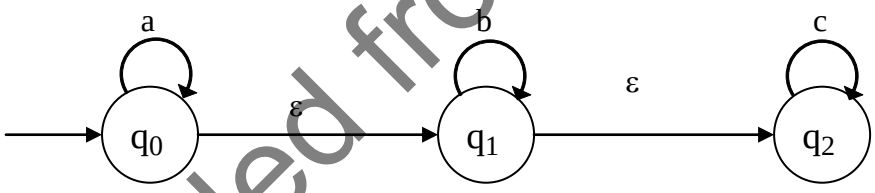
Remove ε transition from q0 to q1



Now remove transition from q0 to q2. As q0 to q2 is transition q0 will become start and final state both.



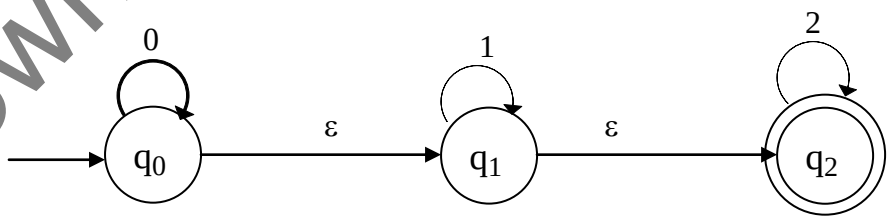
19. Obtain the ε closure of states q0 and q1 in the following NFA with ε transition.



Solution:

ε - CLOSURE {q0} = {q0, q1,q2}
ε - CLOSURE {q1} = {q1,q2}

20. Obtain ε closure of each state in the following NFA with ε move.



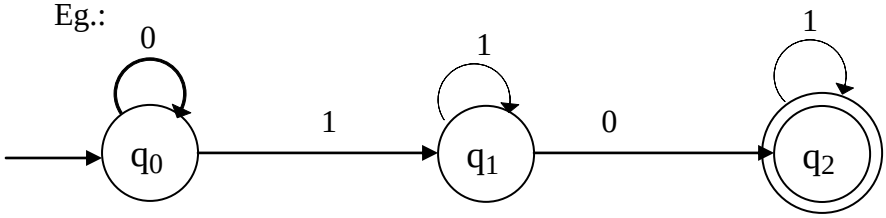
Solution:

ε - CLOSURE {q0} = {q0, q1,q2}
ε - CLOSURE {q1} = {q1,q2}
ε - CLOSURE {q2} = {q2}

21. Explain a transition diagram.

It is a 5-tuple graph used state and edges represent the transitions from one state to other state.

Eg.:



22. Explain a transition diagram.

It is the tabular representation of the DFA. For a transition table the transition function is used.

Eg.:

States	Input	
	0	1
{q0}	{q1}	{q0}
{q1}	-	{q2}
{q2}	-	-

23. Explain the transition function.

The mapping function or transition function denoted by δ. Two parameters are passed to this transition function : (i) current state and (ii) input symbol. The transition function returns a state which can be called as next state.

Eg.:

$$\delta (q0, a) = q1$$

24. Differentiate DFA and NFA?

Sl. No	DFA	NFA
1.	DFA is Deterministic Finite Automata	NFA is Non-Deterministic Finite Automata
2.	For given state, on a given input we reach to deterministic and unique state.	For given state, on a given input we reach to more than one state.
3.	DFA is a subset of NFA	Need to convert NFA to DFA in the design of complier.

25. Write short notes on Minimization of DFA?

- Reducing the number of states from given FA
- First find out which two states are equivalent we than replace those two states by one representative state.
- For finding the equivalent states we will apply the following rule
    - The two states S1 & S2 are equivalent if and only if both the states are final or non-final states.

Unit - II

1. **State regular expression.**

   Let Σ be an alphabet. The regular expressions over Σ and the sets that they denote are defined recursively as follows

   a. Ø is a regular expression and denotes the empty set.
   b. ε is a regular expression and denotes the set {ε}
   c. For each 'a' ∈ Σ, 'a' is a regular expression and denotes the set {a}.
   d. If 'r' and 's' are regular expressions denoting the languages L_1 and L_2 respectively then

      r + s is equivalent to $L_1 \cup L_2$ i.e. union
      rs is equivalent to L_1L_2 i.e. concatenation
      r^* is equivalent to L_1^* i.e. closure

2. **How the kleen's closure or closure of L can be denoted?**

   $$L^* = \bigcup_{i=0}^{n} L^i \qquad (\text{e.g. } a^* = \{\varepsilon, a, aa, aaa, \ldots\})$$

3. **How do you represent positive closure of L?**

   $$L^+ = \bigcup_{i=1}^{n} L^i \qquad (\text{e.g. } a^+ = \{a, aa, aaa, \ldots\})$$

4. **Write the regular expression for the language accepting all combinations of a's over the set ∑ = {a}.**

   L = { a,aa,aaa,……}

   R= a^* (i.e. kleen closure)

5. **Write regular expression for the language accepting the strings which are starting with 1 and ending with 0, over the set ∑ = {0,1}.**

   L = { 10,1100,1010,100010……}

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

6. **Show that (0*1*)* = (0+1)*.**

   LHS : (0*1*)* = { ε, 0,1,00,11,0011,011,0011110……}
   RHS : (0+1)* = { ε, 0,1,00,11,0011,011,0011110……}
   Hence
   LHS = RHS is proved

7. **Show that (r+s)* ≠ r* + s*.**

   LHS : (r+s)* = { ε, r,s,rs,rr,ss,rrrsssr,……}
   RHS : r* + s* = { ε, r,rr,rrr……}U { ε, s,ss,sss,……}
   = { ε, r,rr,rrr,s,ss,ssss……..}
   Hence
   LHS ≠ RHS is proved

8. What do you mean by homomorphism?

A string homomorphism is a function on strings that works by substring a particular sting for each symbol.

Eg. h(0) = ab

h(1) = ε is a homomorphism, where replace all 0's by ab and replace all 1's by ε.

Let w = 0011

h(w) = abab

9. Explain the application of the pumping lemma.

Pumping Lemma is used to prove the language is not regular.

10. Describe the following by regular expression

a. L1 = the set of all strings of 0's and 1's ending in 00.

b. L2 = the set of all strings of 0's and 1's beginning with 0 and ending with 1.

r1 = (0+1)*00

r2 = 0(0+1)*1

11. Show that (r*)* = r* for a regular expression r.

LHS = r* = { ε, r,rr,rrr, …….)

(r*)* = { ε, r,rr,rrr, …….)*

(r*)* = { ε, r,rr,rrr, …….) = r*

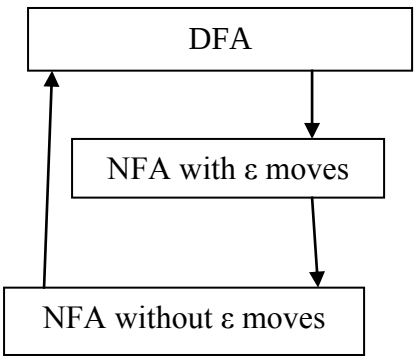
LHS = RHS

12. Write down the closure properties of regular language.

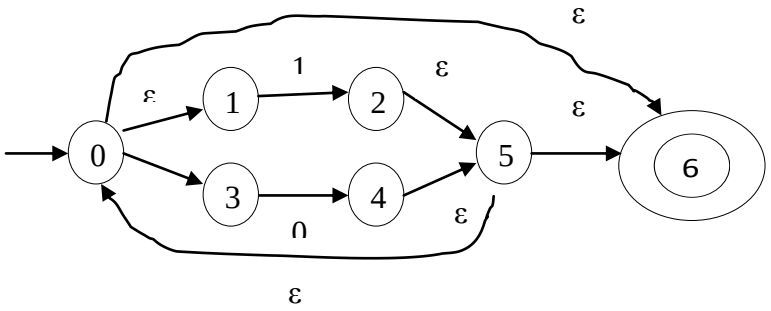
The regular languages are closed under

1. Union
2. Intersection
3. Complement
4. Difference
5. Reversal
6. Closure
7. Concatenation
8. Homomorphism
9. Inverse Homomorphism

13. Write down the relationship between FA and regular expression.



14. Construct NFA with ε moves for the regular expression (0+1)*.



15. What is pumping lemma?

Let L be a regular language. Then there exists a constant n such that for every string w in L such that $|w| \geq n$, w = xyz such that

i) $y \neq \varepsilon$
ii) $|xy| \leq n$
iii) For all $i \geq 0$, $xy^iz \sum L$

16. If L = { The language starting and ending with 'a' and having any combinations of b's in between, that what is r?

r1 = a b*a

17. Give regular expression for

L= L1 ∩ L2 over alphabet {a,b}
where L1 = all strings of even length
L2 = all strings starting with 'b'.

Solution :

r = r1 + r2
$r = a^nb^n + b (a+b)^*$

18. State Arden's theorem.

Let P and Q be two regular expression over alphabet ∑. If P does not contain null string ε, then

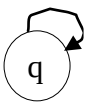
R = Q + RP

It has the solution

R = QP*

19. What is dead state?

All the non final states which transmit to itselt for all input symbols in ∑, are called Dead state.



q – non final state, also the dead state.

20. Let ∑ = {0,1} and $\sum^1$ = {0,1,2} with h(0) = 01 and h(1) = 112. Find h(010) and homomorphic image of L = { 00, 010 }.

Solution :

$\sum = \{0,1\}$ and $\sum^1 = \{0,1,2\}$

h is defined as :

h(0) = 01 and h(1) = 112
h(010) = 0111201

The homomorphic image of L = { 00, 010 }is h(L) = {0101, 0111201}.

Unit - III

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

Solution:

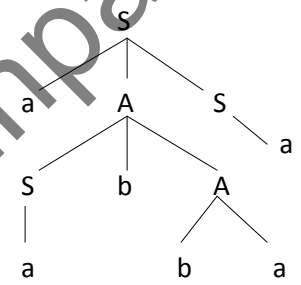
S → aCa
→ aaCaa C→ aCa
.
.
.
→ a^nCa^n
→ a^nba^n C → b

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

2. **Consider G whose productions are S→ aAS/ a ,A→SbA / SS / ba, show that → aabbaa and construct aderivation tree.**

Solution:

S → aAs
→ aSbAs A→SbA
→ aabAS S→a
→ aabbaS A→ba
→ aabbaa S→a



3. **Find L(G) where G = ({S},{0,1}, {S→0S1, s→ε },S)**

Solution:

S → 0S1
→ 00S11
.
.
.
→ 0^nS1^n

$L(G) = \{ 0^n1^n ; n > 0\}$

4. **Define a derivation tree for CFG.**

A derivation tree for a CFG G=(V,T,P,S) is a tree satisfying the following

- Every vertex has a label, which is a symbol of V U T U ε
- The label of the root is S.
- If a vertex is interior and has a label A, then A must be in V.
- If n has a label A and vertices n1, n2,……nk are sons of the vertex n, in x1,x2,……xk must be a production in P.
- If vertex n has label ε, then n is a leaf and is the only son of its father.

5. **Construct CFG L= { a^nb^n ; n≥1}.**

Solution:

The Production are

S → aSb / ε, where G= ({S}, {a,b,ε}, P, S)

6. Find a LM derivation for aaabbabbba with the productions.

P : S → aB / bA, A → a /S / bAA, B → b / bS / aBB

Solution:

S → aB → aaBB → aaaBBB→aaabBB→ aaaabbB→aaabbaBB→aaabbabB
→ aaabbabbS → aaabbabbbA
S → aaabbabbba

7. Find L(G), S → aSb, S → ab.

Solution:

S → aSb
→ aaSbb C→ aSb
.
.
➔ a^nSb^n
➔ a^nb^n C → ab

$$L(G) = \{ a^n b^n ; n \geq 1 \}$$

8. Show that id* id can be generated by two distinct leftmost derivation in the grammar

E→ E+E / E*E / (E) / id

Solution:

(i) E → E +E
→id+E
→id+E*E
→id+id*E
→id+id*id

(ii) E → E *E
→ E +E*E
→E+E*id
→E+id*id
→id+id*id

We showed that id+id*id can be generated by two distinct LMD.

9. Define pushdown automaton.

A Pushdown Automata is a finite automation with extra resource called stack.
It consists of 7 tuples.
P = (Q, ∑, ⌈, δ, q_0, Z_0, F)
Where
Q – Finite set of states
∑ - Finite set of input symbols
⌈ - Finite set of stack symbols
δ - Transition function
q_0 – Start State
Z_0 – Start symbol of the stack
F – Final State.

10. What are the different ways of language acceptances by a PDA and define them.

i) Acceptance by final state
L (M) = { w | (q_0, w, z_0) |--*(p, ε, γ) for some P in F and γ in |--*}

ii) Acceptance by empty stack
N (M) = { w | (q_0, w, z_0) |--*(p, ε, ε) for some P in Q}

11. Write a CFG for the set of strings which does not produce any palindromes.

Here the grammar should be designed in such a way that $w \neq w^R$

S → aSa / bSb / C

C → aAb / bAa

A→ aA / bA / ε

12. Find the language for the CFG

S →> aSa / aAb

A → bAa / b a

Solution:

$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow \ldots\ldots \Rightarrow a^nSb^n \Rightarrow a^naAbb^n \Rightarrow a^nabAabb^n$

$\Rightarrow a^nabnaabb^n \Rightarrow a^nb^ma^mb^n$

$L = \{ a^nb^ma^mb^n ; m,n \geq 1\}$

13. Find the derivation tree for the grammar

G = ({S, A, B}, {a,b}, P, S}

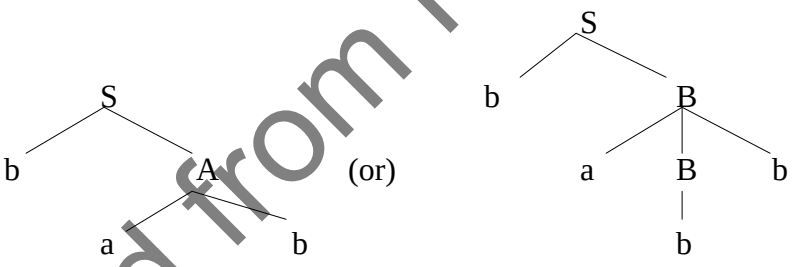
Where P is given by

S → Aa / bB

A→ab

B→ aBb / a

Solution:



14. Construct a PDA that accepts the language generated by the grammar

S→ aSbb / aab

Solution:

The PDA A = ({q}, {a,b}, {S,a,b},δ, q,S}

Where δ :

i) $\delta (q, z_0, S) = \{(q, aSbb), (q, abb)\}$

ii) $\delta(q, a, a) = \{(q, \varepsilon)\}$

iii) $\delta(q, b, b) = \{(q, \varepsilon)\}$

15. Construct a PDA that accepts the language generated by the grammar

S → aABB , A → aB / a , B → bA / b

Solution:

The PDA A = ({q}, {a,b}, {S,A,B,Z,a,b},δ, q,S}

Where δ :

i) $\delta (q, z, S) = \{(q, aABB)\}$

ii) $\delta(q, z, A) = \{(q, aB, (q,a)\}$

iii) $\delta(q, z, B) = \{(q, bA, (q,b)\}$

iv) $\delta(q, a, a) = \{(q, \varepsilon)\}$

v) $\delta(q, b, b) = \{(q, \varepsilon)\}$

16. Define parse tree.

A data structure to represent the source program in a complier is called parse tree. Parse tree can have nodes and edges.

17. How do you convert CFG to a PDA.

Let G = (V, T, P, S) be a CFG. Then construct a PDA P that accepts L(G) by empty stack as follows :

P = ({q}, T, V U T, δ, q, S)

Where δ is given by

1) For each variable A,

$\delta(q,\varepsilon,A) = \{(q,\beta) \mid A \rightarrow \beta$ is a production of P}

2) For each terminal a,

$\delta(q,a,a) = \{(q,\varepsilon)\}$

18. Define Deterministic PDA.

A PDA P = (Q, $\sum$, $\lceil$, δ, q_0, Z_0, F) to be deterministic iff

1) $\delta(q,a,X)$ has at most one member for any q in Q, a in $\sum$ or a = ε and X in |--.
2) If $\delta(q,a,X)$ is not empty, for some a in $\sum$, then $\delta(q,\varepsilon,X)$ must be empty.

19. Is it true that NDPA is more powerful than that od DPDA? Justify your answer.

No, NPDA is not powerful than DPDA. Because NPDA may produce ambiguous grammar by reaching its final state or by emptying its stack. But DPDA produces only unambiguous grammar.

20. What is the additional feature PDA has when compared with NFA? Is PDA superior over NFA in the sense L acceptance? Justify your answer.

PDA is superior NFA by having the following additional features.

- Stack which is used to store the necessary tape symbols and use the state to remember the conditions.
- Two ways of L acceptances, one by reaching its final state and another by emptying its stack.

UNIT – IV

1. **What are the two major normal forms for context-free grammar?**

   The two Normal forms are
   i. Chomsky Normal Form (CNF)
   ii. Greibach Normal Form (GNF)

2. **What is a useless symbol?**

   A symbol x is useful if there is a derivation

   $S \Rightarrow^* \alpha x \beta \Rightarrow^* w$ for some $\alpha, \beta, w \sum T^*$ or else , it is useful.

3. **How do you simplify the context-free grammar?**
   - First eliminate useless symbols, where the variable or terminals that do not appear in any derivation of a terminal string from the start symbol.
   - Next eliminate ε- productions which is of the form A→ ε for some variable A.
   - Eliminate unit productions, which are of the form A → B for variables A, B.
   - Finally use any of the normal forms to get the simplified CFG.

4. **Define Nullable Variable?**

   Nullable variable in a CFG G =(V,T,P,S) can be defined as follows.
   - Any variable A for which P contains the production A→ A, is nullable.
   - If P contains the production A→ B1,B2, ……. Bn and B1, B2, ….Bn are nullable variables, then A is nullable.
   - No other variables in V are nullable.

5. **Define generating symbol?**

   Let G = (V,T,P,S) is generating, if $X \Rightarrow^* w$ for some terminal string w.

   e.g. A→ aAB / ε
   B→ b
   Then A is a generating symbol since $A \Rightarrow *ab$

6. **Let G = (V, T, P,S) with the productions given by**

   S → aSbS/ B / ε
   B→ abB Eliminate the useless production.

   Solution:

   Remove B is useless production because of Variable is not reachable.
   S → aSbS / ε

7. **What is substitution Rule?**

   A production $A \rightarrow x_1 B x_2$ can be eliminated from a grammar if B is replaced by all strings derived by B in one step, provided A and B are variables.

8. **What is a useful production?**

   Let G = (V,T,P,S) be a CFG. A variable A ∑ V is said to be useful iff there is atleast one w∑L(G) such that
   $S \Rightarrow^* xAy \Rightarrow^* w$ with x,y in (VUT)*.

9. **Determine whether the grammar G has a useless production?**

   S→A
   A→aA / ε
   B → bA

The variable B is useless, since it is used by the start variable or by the variable in the start production.

B→ bA is a useless production.

10. Write a procedure to eliminate ε production.

i) For all productions A→ε, put A into V

ii) Repeat the following steps until no new variable are added.

a. For all productions

B→ $A_1A_2A_3$……A_n, where $A_1A_2A_3$……A_n are in V

b. Put B into V

11. Write the procedure to eliminate the unit productions.

i) Find all variables B, for each A such that

A ⇒* B

ii) The new grammar G is obtained by letting into P all non-unit productions of P.

iii) For all A and B satisfying A ⇒* B, add to p

A→ y_1 / y_2 / y_3 /……/ y_n , where B → y_1 / y_2 / y_3 /……/ y_n is the set of productions in P.

12. Define CNF.

A CFG without any ε–production is generated by a grammar in which the productions are of the form.

A → BC or A→ a, where A,B ∑ V and a ∑ T.

13. What is GNF?

Every CFL L without ε cab be generated by a grammar for which every production is of the form A→ aα, where A∑V, a∑T, is a string of variables.

14. What is a Turning Machine?

A finite state machine with storage is called a Turing Machine.

15. Define a Turing Machine.

The Turing Machine is denoted by

M = (Q, ∑, ⌈, δ, q_0, B, F)

Where

Q – Finite set of states

∑ - Finite set of input symbols

⌈ - Finite set of stack symbols

δ - Transition function - Q x |-- → Q x |-- x {L,R}, Where L,R – Directions.

q_0 – Start State

B – a Start symbol of the |--, a blank

F – Final State.

16. 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.

17. What is multiple tracks Turing machine?

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

18. What is multidimensional Turing Machine?

The Turing Machine which has the useful finite control, but the tape consist of a k-dimensional array of cells infinite in all 2K directions for some fixed K. Depending on the state and symbol scanned, the device changes state, prints a new symbol and moves its tape head in one of 2K directions, along one of the K axes.

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

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

q_0 x |--* q_fy

A function f is 'Turing Computable' if there exists a Turing Machine that performs a specific function.

20. 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.

UNIT – V

1. **What is the weak-form of Turing thesis?**

   A Turing Machine can compute anything that can be computed by a general purpose digital computer.

2. **What is the strong-form of Turing thesis?**

   A Turing Machine can compute anything that can be computed. This is the strong form of Turing thesis.

3. **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.

4. **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.

5. **What is diagonalization language?**

   The language L_d. Which consists of all those strings w such that the Turing machine represented by w does not accept the input w.

   $L_d = \{ w_i \mid w_i \notin L(M_i)\}$

6. **Define decidability (or) 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.

   (or)

   A problem is said to be decidable if it is a recursive language.

7. **Define Undecidable problem?**

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

8. **Define Universal language?**

   A Universal Turing Machine Mu is an automation, that given as input the description of any Turing Machine M and a string w, can simulate the computation of M on w.

9. **What are the reasons for a TM not accepting its input?**

   i) The TM may halt in a non final state.
   ii) The TM may enter into an indefinite loop.

10. **Define trivial property?**

    A property is trivial if it is either empty of is all RE languages.

11. **Define rice Theorem?**

    Every non-trivial property of the RE languages is undecidable.

12. Define post's correspondence problem?

An instance of PCP consists of two lists,

$A = w_1, w_2, w_3, \ldots\ldots\ldots w_k$

$B = x_1, x_2, x_3, \ldots\ldots\ldots\ldots x_k$ of strings over some ∑. This instance of PCP has a solution if there is any sequence of integers $i_1, i_2, \ldots\ldots\ldots\ldots i_m$ with m≥1.

Such that

$w_{i1}, w_{i2}, w_{i3} \ldots\ldots\ldots\ldots\ldots\ldots\ldots w_{im} = x_{i1}, x_{i2}, x_{i3}, \ldots\ldots\ldots\ldots x_{im}$

The sequence of $i_1, i_2, \ldots\ldots\ldots\ldots i_m$ is a solution to this instance of PCP.

13. Let A and B be lists of three strings each, as defined in the following table?

	A	B
	w	**x**
1	**1**	**111**
2	**10111**	**10**
3	**10**	**0**

Find the instance of post correspondence Problem.

Solution :

Apply $w_{i1}, w_{i2}, w_{i3} \ldots\ldots\ldots\ldots\ldots\ldots\ldots w_{im} = x_{i1}, x_{i2}, x_{i3}, \ldots\ldots\ldots\ldots x_{im}$ to this problem.

Take M = 4

$w_2 w_1 w_1 w_{i3} = x_2 x_1 x_1 x_{i3}$

10 111 111 0 =10 111 111 0

Instance = 2,1,1,3.

14. Define modified post's correspondence problem?

Given lists A and B, of K strings each from ∑*, say

$A = w_1, w_2, w_3, \ldots\ldots\ldots w_k$

$B = x_1, x_2, x_3, \ldots\ldots\ldots\ldots x_k$

Does there exist a sequence of integers $i_1, i_2, \ldots\ldots\ldots\ldots i_r$ such that

$w_{i1}, w_{i2}, w_{i3} \ldots\ldots\ldots\ldots\ldots\ldots\ldots w_{im} = x_{i1}, x_{i2}, x_{i3}, \ldots\ldots\ldots\ldots x_{ir}$

The sequence of $i_1, i_2, \ldots\ldots\ldots\ldots i_m$ is a solution to this instance of PCP.

15. Define problem solvable in polynomial Time?

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

16. Define the classes 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 a 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.

17. Define NP – Complete Problem?

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

a. L is in NP
b. For every language L^1 in NP there is a polynomial time reduction of L^1 to L

18. What are tractable problems?

The problems which are solvable by polynomial time algorithms are called tractable problems.

19. What are the properties of recursive enumerable sets Which are undecidable?

i) Emptiness
ii) Finiteness
iii) Regularity
iv) Context – freedom

20. What are the properties of recursive and Recursively Enumerable Language?

1. The complement of a Recursive language is Recursive.
2. The union of two recursive languages are recursive. The union of two RE languages are RE.
3. If a language L and complement L are both RE, then L is recursive.