July 2022

## Part-A

1. Explain DFA and NFA with suitable examples.

Ans: DFA - Deterministic Finite Automata is a finite automata which can have only one transition from a state on an input symbol.

DFA is a five tuple $(Q, \Sigma, \delta, q_0, F)$

where $Q$ = non-empty finite set of states

$\Sigma$ = non-empty finite set of input symbols

$q_0$ : start state

$F$ : final state

$\delta : Q \times \Sigma \rightarrow Q$ - Transition function.

Example: Construct a DFA that ends with 01 where $\Sigma = \{0, 1\}$

Step 1: $L = \{01, 101, 001, 1001, 0001, \ldots\}$

Step-2: DFA:


Step-3: Transition table

| | | 0 | 1 |
|---|---|---|---|
| ε | $q_0$ | $q_1$ | $q_0$ |
| 0 | $q_1$ | $q_1$ | $q_2$ |
| 01 | $q_2$ | $q_1$ | $q_0$ |

~~ε0~~ ε1

~~00~~ 01

~~010~~ ~~011~~

Que-4: DFA is defined as: $D = \{Q, \Sigma, \delta, q_0, F\}$

$Q = \{q_0, q_1, q_2\}$

$\Sigma = \{0, 1\}$

$q_0 = q_0$

$F = q_2$

$\delta(q_0, 0) = q_1$

$\delta(q_0, 1) = q_0$

$\delta(q_1, 0) = q_1$

$\delta(q_1, 1) = q_2$

$\delta(q_2, 0) = q_1$

$\delta(q_2, 1) = q_0$

NFA - Non-Deterministic Finite Automata. If an automata makes more than one transition from a given state for a single input symbol it is called NFA.

Mathematical Representation of NFA $(Q, \Sigma, \delta, q_0, F)$

$Q$ = non-empty finite set of states

$\Sigma$ = non-empty finite set of input symbols

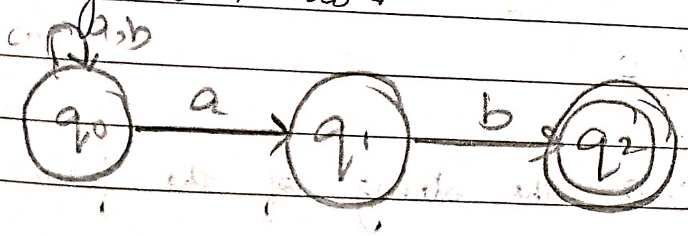
$q_0$ = start state

$F$ = final state

$\delta = Q \times \Sigma \rightarrow 2^Q$ : Transition function.

Example: Draw NFA to accept strings of a's and b's ending with ab.

Step-1:




Step-2: Transition table:

| | a | b |
|---|---|---|
| $q_0$ | $\{q_0, q_1\}$ | $q_0$ |
| $q_1$ | $\phi$ | $q_2$ |
| $q_2$ | $\phi$ | $\phi$ |

Step-3: ~~Transd~~ NFA is defined as: $N = (Q, \Sigma, \delta, q_0, F)$

$Q = \{q_0, q_1, q_2\}$

$\Sigma = \{a, b\}$

$q_0 = q_0$

$F = q_2$

$\delta$: $\delta(q_0, a) = q_0, q_1$

$\delta(q_0, b) = q_0$

$\delta(q_1, b) = q_2$

2. State and prove pumping lemma for regular languages and prove that the $L = \{ww \mid w \in \{a,b\}^*\}$

Ans: Pumping lemma is a method of pumping many input strings from a given string. It is used to show that certain languages are not regular.

Step-1: Consider a language as regular.

Step-2: Assume a constant 'c' and select the string 'w' for the language L such that $|w| \geq c$.

Step-3: Divide w as xyz where:

Case-1: $|y| > 0$

Case 2: $|xy| \leq c$

Step-4: For every $i \geq 0$ the string of the form $xy^iz$ belongs to L.

$L = \{ww \mid w \in (a,b)^*\}$

Step-1: Given language:

$w = \{a, b, aa, bb, ab, ba, aabb, bbaa, \ldots\}$

$(w\#w)$
$ww = L = \{aa, bb, aaaa, bbbb, abab, baba,$ ~~aaabbb~~ $aabbaabb\}$

Step-2: Let $w = abab$
$|w| \geq c$
$|abab| \geq 4$

$u = a$ $x = a$
$y = ba$
$z = b$

Step-3: case-1:
$|y| > 0$
$|ba| > 0$

case-2:
$|xy| \leq c$
$aba \leq 4$

Step-4: $xy^iz$ for every $i \geq 0$

$i = 0 \Rightarrow ab \notin L$
ab is not regular. Hence pumping lemma is proved.

3. Minimise the DFA:

| $\delta$ | a | b |
|---|---|---|
| $\rightarrow q_0$ | $q_1$ | $q_2$ |
| $q_1$ | $q_1$ | $q_3$ |
| $q_2$ | $q_1$ | $q_2$ |
| $q_3$ | $q_1$ | $q_4$ |
| $* q_4$ | $q_1$ | $q_2$ |

Step-1: Zero equivalence: Non-final states $= \{q_0, q_1, q_2, q_3\}$
Final states $= \{q_4\}$

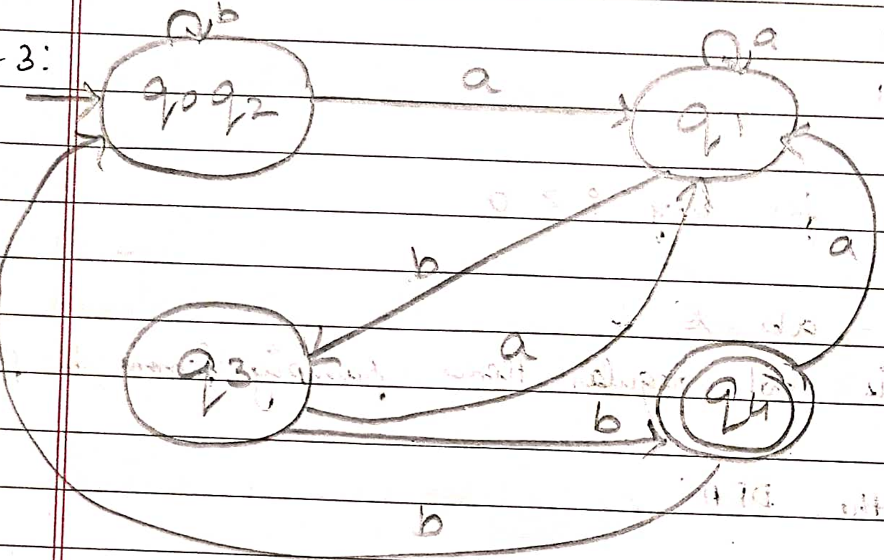
One-equivalence: $\{q_0, q_1, q_2\}$
$\{q_3\}$ $\{q_4\}$

Two-equivalence: $\{q_0, q_2\}$ $\{q_1\}$ $\{q_3\}$ $\{q_4\}$

Step-2: Minimised transition table:

| $\delta$ | a | b |
|---|---|---|
| $\rightarrow \{q_0, q_2\}$ | $q_1$ | $\{q_0, q_2\}$ |
| $\{q_1\}$ | $q_1$ | $q_3$ |
| $\{q_3\}$ | $q_1$ | $q_4$ |
| $*\{q_4\}$ | $q_1$ | $\{q_0, q_2\}$ |

Step-3:




4. Explain with example Chomsky hierarchy of generative grammar.

Ans: Chomsky hierarchy represents the class of languages that are accepted by the different machine. Noam Chomsky classifies the grammar and the generated languages into different types.

1. Type 0 language: It is known as unrestricted grammar. There is no restriction on grammar rules.

- Production rule: $\alpha \rightarrow \beta$ where $\alpha \neq \epsilon$
The LHS should contain only 1 non-terminal.

- It can be modeled by Turing machines.

Example : $bAa \to aa$
$S \to S$

2. Type 1 Grammar: It is known as Context sesitive Grammar (CSG). It is used to represent context sesitive language.

- Production rule: $\alpha \to \beta$ where it should not contain $\epsilon$. $\alpha \neq \epsilon$. The ~~mode~~ of $|\alpha| \leq |\beta|$.

- This language includes any string of terminal and non-terminal where the length of the string in the RHS must be greater than or equal to the length of the string in LHS.

Example; $S \to AT$
$T \to xy$
$A \to a$

3. Type 2 Grammar: It is known as Context Free Grammar. Context free languages are the languages which can be represented by CFG.

- Production rule: $A \to \alpha$ where $\alpha$ is any combination of terminal and non-terminals.

- It is recognised by PDA.

- Example - $A \to aBb$
$A \to b$
$B \to a$

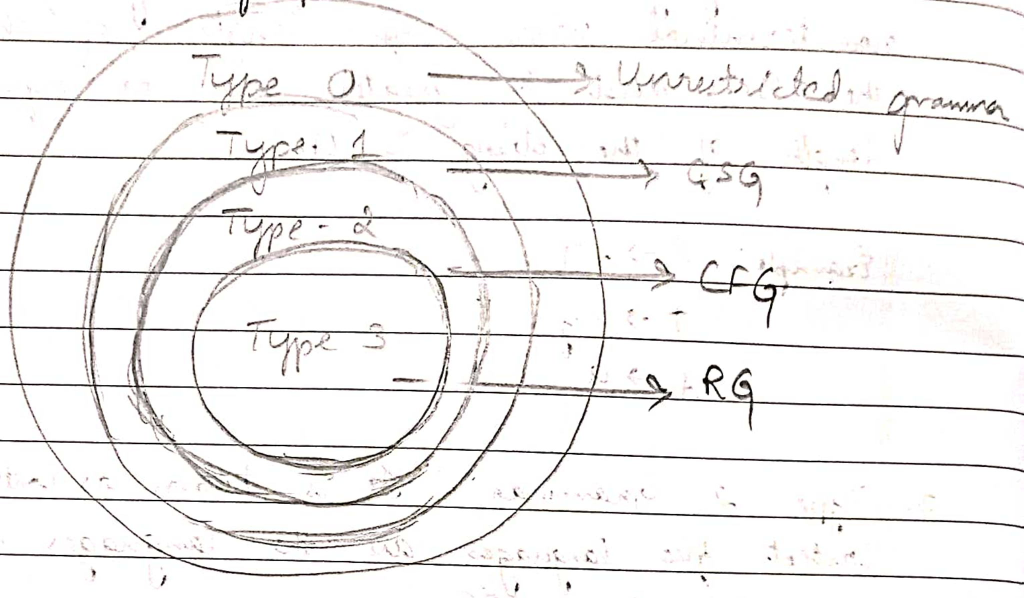
4. Type 3 Grammar: It is known as regular grammar. It is most restricted form of grammar. Regular languages are those languages which can be described using regular

expressions

- Production rule: $A \rightarrow a$
  $A \rightarrow aB$

- On the LHS it should contain only one terminal and on the RHS it should contain terminal or combination of terminal and non-terminal.

- It is used in finite automata.

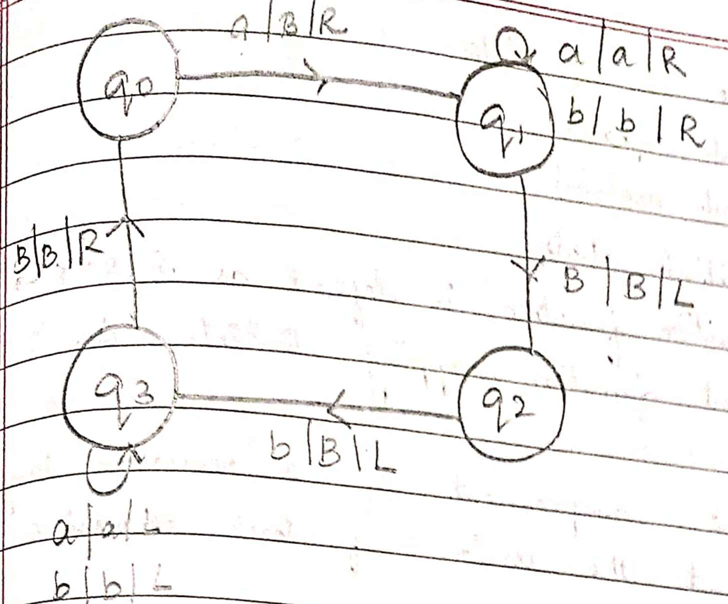



5. Construct a turing machine for the following language. $L = \{a^n b^n \mid n \geq 1\}$.

Ans: step-1 $L = \{ab, aabb, aaabbb, aaaabbbb \ldots\}$

Step-2: Consider input tape = BaaabbbB

BaaabbbB
→ BBaabbbB
→ BBaabbBB
→ BBBabbBB
→ BBBabBBB
→ BBBBbBBB
→ BBBBBBBB




6. Explain Moore and Mealy machine.

1. Moore machine: If the output $z(t)$ depends on current state $Q(t)$ and it is independent on the current input is defined as:

$$z(t) = \lambda(q(t))$$

where, $\lambda$ = output function.

Tuples of Moore machine: $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$.

where $Q$ = finite set of states
$\Sigma$ = input alphabets
$\Delta$ = output alphabet
$q_0 \in Q$ = initial state
$\delta$ = Transition function is defined as $\delta: Q \times \Sigma \to Q$
$\lambda$ = Output function mapping $Q$ into $\Delta$.

2. Mealy machine: The value of output function $z(t)$ is a function of present state $Q(t)$ and present input $x(t)$ is defined as:

$$z(t) = \lambda(q(t), x(t))$$ where,

$\lambda$ = output function.

Tuples of Mealy machine: $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$.

where, $Q$ = finite set of states
$\Sigma$ = input alphabets
$\Delta$ = output alphabet
$q_0 \in Q$ = Initial state
$\delta$ = Transition function is defined as $\delta : Q \times \Sigma \rightarrow Q$
$\lambda$ = Output function mapping $Q$ into $\Delta$.

7. Prove that the complement of a recursive language is recursive and the union of two recursive language is recursive.

Ans: 1. The complement of a recursive language is recursive: If $L(G)$ is a regular language, its complement $L'(G)$ will also be regular. Complement of a language can be found by subtracting strings which are in $L(G)$ from all possible strings.

Example: $L(G) = \{a^n \mid n > 3\} = \{4, 5, 6 \ldots\}$
$L'(G) = \{a^n \mid n \leq 3\} = \{1, 2, 3\}$

2. Union of two recursive language is recursive: If $L1$ and if $L2$ are two regular languages, their union $L1 \cup L2$ will also be regular.

Example: $L1 = \{a^n \mid n > 0\} = \{1, 2, 3, 4, 5\}$
$L2 = \{b^n \mid n > 0\} = \{5, 6, 7, 8, 9\}$
$L3 = L1 \cup L2 = \{a^n \cup b^n \mid n > 0\}$ is also regular. $= \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$

8. Explain primitive recursive functions and $\mu$- recursive functions.

Ans. 1. Primitive recursive function: A function $f$ is a primitive recursive function if

i. It is one of the basic function, or

ii. It is produced by performing operations such as composition and recursion to the basic functions.

Example: The addition function, add $(x, y)$ is a primitive recursive function because, add $(x, 0) = x$

$$= P_1(x)$$

$$\text{add}(x, y+1) = \text{add}(x, y) + 1$$
$$= S[\text{add}(x, y)]$$
$$= S[P_3(x, y, \text{add}(x, y))]$$

Thus add $(x, y)$ is a function produced by applying composition and recursion to basic functions, $P_k$ and $S$.

2. μ-recursive function / partial recursive function: These are the functions that can be computed by Turing machines. The μ-recursive functions are partial functions that take finite tuples of natural numbers and return a single natural number.

A function f is a partial recursive function if,

i. It is one of the basic function or

ii. It is produced by performing operations such as composition, recursion and minimisation to the basic functions.

Example: Show that $f(x) = \frac{x}{2}$ is partial recursive function over N.

$$f(x) = \frac{x}{2}$$

We may write, $y = \frac{x}{2}$

Then, $x = 2y$

$2y - x = 0$

Let, $g(x,y) = |2y - x|$

Let $g(x,y) = 0$, for some $x$ and $y$.

let $f_1(x) = \mu[|2y - x| = 0]$

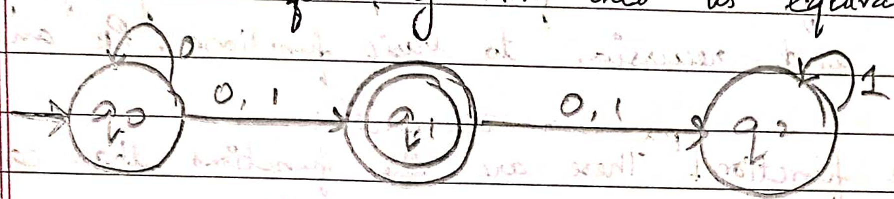
Only for even values of $x$, $f_1(x)$ is defined.

When $x$ is odd, $f_1(x)$ is not defined. so $f_1$ is partial recursive.

Since, $f(x) = \frac{x}{2} = f_1(x)$, $f$ is a partial recursive function

Part - B

9.a. Convert the following NFA into its equivalent DFA:




Ans:

Step-1: The given NFA is $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$

where $Q_N = \{q_0, q_1, q_2\}$

$\Sigma = \{0, 1\}$

$q_0 = q_0$

$F = q_1$

| $\delta_N$ | 0 | 1 |
|---|---|---|
| $\rightarrow q_0$ | $q_0 q_1$ | $q_1$ |
| $* q_1$ | $q_2$ | $q_2$ |
| $q_2$ | $\phi$ | $q_2$ |

Step-2: Since $q_0$ = Initial state of NFA

$\therefore$ $q_0$ = Initial state of DFA

$\therefore Q_0 = q_0$

Step-3: Construction of Transition table for DFA.

| | 0 | 1 |
|---|---|---|
| $\rightarrow q_0$ | $\{q_0 q_1\}$ | $q_1$ |
| $* q_0 q_1$ | $\{q_0 q_1 q_2\}$ | $\{q_1, q_2\}$ |
| $* q_1$ | $q_2$ | $q_2$ |
| $* q_0 q_1 q_2$ | $\{q_0 q_1 q_2\}$ | $\{q_1 q_2\}$ |
| $* q_1 q_2$ | $q_2$ | $q_2$ |
| $q_2$ | $\phi$ | $q_2$ |
| $q_\phi$ | $q_\phi$ | $q_\phi$ |

Transition function from $q_0$.

$\delta_D(q_0, 0) = q_0 q_1$

$\delta_D(q_0, 1) = q_1$

$\delta_D((q_0, q_1), 0) = \delta_N(q_0, 0) \cup \delta_N(q_1, 0)$

$= q_0 q_1 q_2$

$\delta_D((q_0, q_1), 1) = \delta_N(q_0, 1) \cup \delta_N(q_1, 1)$

$= q_1 q_2$

$\delta_D(q_1, 0) = q_2$

$\delta_D(q_1, 1) = q_2$

$\delta_D((q_0, q_1, q_2), 0) = \delta_N(q_0, 0) \cup \delta_N(q_1, 0) \cup \delta_N(q_2, 0)$

$= q_0 q_1 q_2$

$\delta_D((q_0, q_1, q_2), 1) = \delta_N(q_1, 0) \cup \delta_N(q_1, 1) \cup \delta_N(q_2, 1)$

$= q_1 q_2$

$\delta_D((q_1, q_2), 0) = \delta_N(q_1, 0) \cup \delta_N(q_2, 0)$

$= q_2$

$\delta_D((q_1, q_2), 1) = \delta_N(q_1, 0) \cup \delta_N(q_2, 1)$

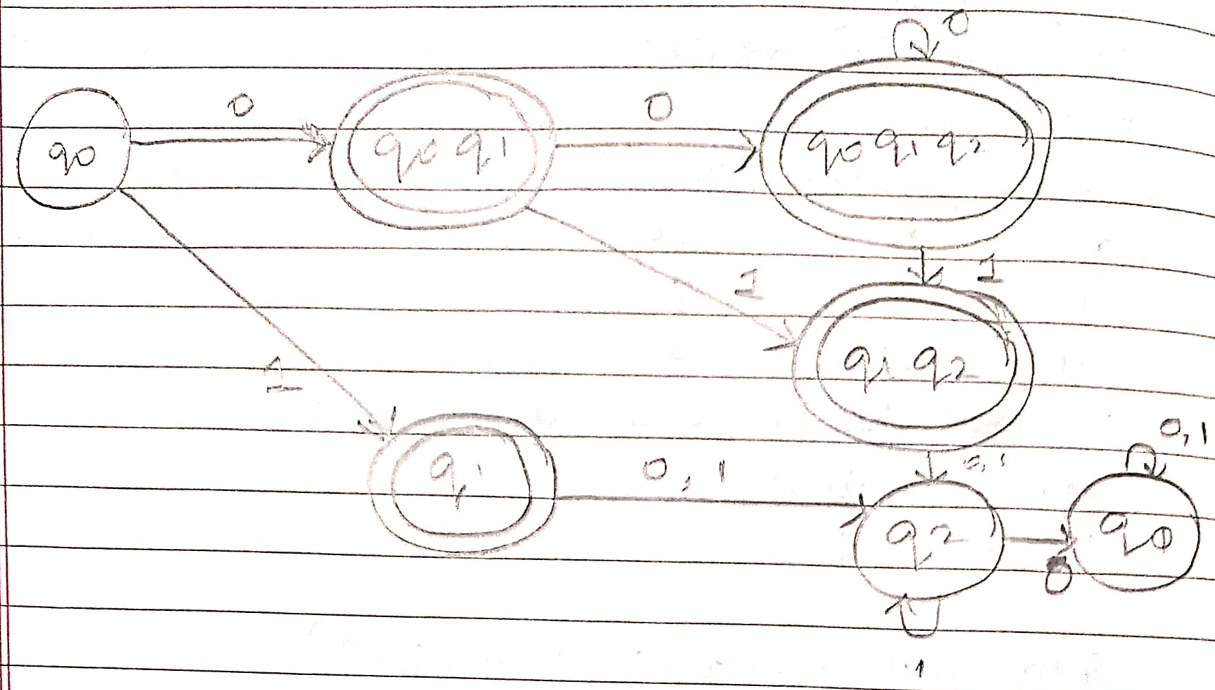
$= q_2$

$\delta_D(q_2, 1) = q_2$

No more new states. Therefore, we stop.

Step-3: The final states are: $\{q_0, q_1\}$ $\{q_1\}$ $\{q_0 q_1 q_2\}$ $\{q_1 q_2\}$

DFA:




b. Write short notes on the applications of Finite automata.

Ans: A Finite automata is a mathematical model which is used to study the abstract machines with the inputs chosen from Σ.

The applications of Finite Automata are:

1. Language Processing: The applications of finite automata are found in language processing such as string processing, parsing, information retrieval, image compressions etc.

2. Compiler Construction: FA is used in the design of first phase of compiler i.e. Lexical Analysis, which breaks the input text into tokens.

~~3.~~

3. Computer Networks: FA is used in the design of communication protocols. Example- Communication link.

4. Video games: Games take advantage of complex FA to

control artificial intelligence. Chat dialogues where the user is prompted with choices can be run using F.A.

5. Design of Digital circuits: FA is used during designing and checking the behaviour of digital circuits using software. It is useful in design of automatic traffic signals and circuit verification.

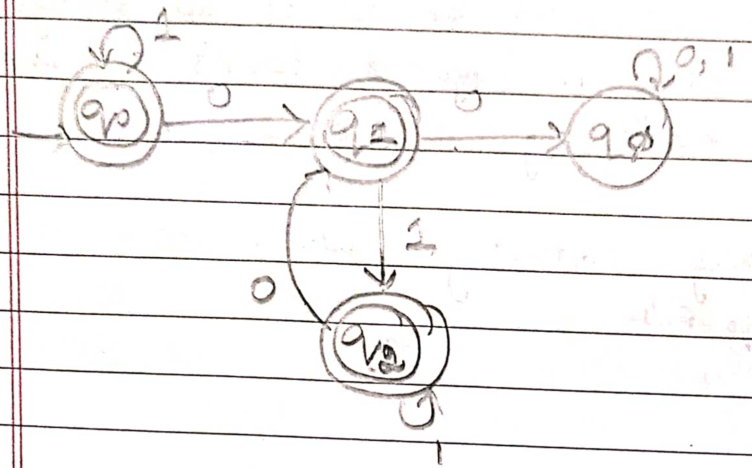
10Q. a Design a DFA to accept the set of all strings not containing the substring 00 for $\Sigma = \{0, 1\}$ and show the acceptability of the string 101011.

b ~~Explain ε-closure and write the steps to find out ε-closure.~~

Ans. Step-1: Minimum string : 0, 1

Step-2: $\Sigma = \{0, 1\}$

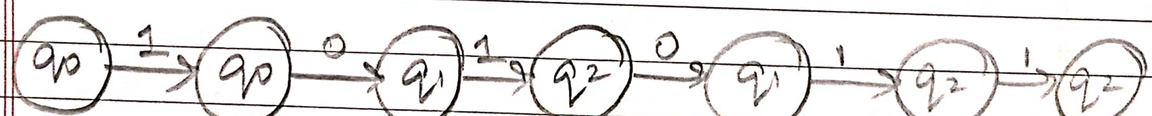
Step-3: Transition diagram




| | 0 | 1 |
|---|---|---|
| $*q_0$ | $q_1$ | $q_0$ |
| $*q_1$ | $q_\phi$ | $q_2$ |
| $*q_2$ | $q_1$ | $q_2$ |
| $q_\phi$ | $q_\phi$ | $q_\phi$ |

DFA can be defined as $Q = \{Q, \Sigma, \delta, q_0, F\}$
where $Q = \{q_0, q_1, q_2, q_\phi\}$
$\Sigma = \{0, 1\}$
$q_0 = q_0$
$F = \{q_0, q_1, q_2\}$

Checking the string 101011.




since $q_2$ is final state, string is accepted.

Date
Page

b. Explain E-closure and write the steps to find out E-closure.

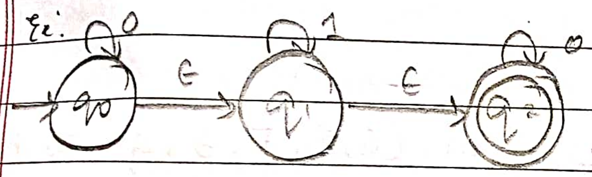
Ans. E-closure is the set of all states which are reachable from state q without processing any input symbol. i.e. it is just the set of states that can be reached from q on E-transition only.

steps to find E-closure:

step-1: q is added to E-closure (q)

step-2: If $q_1$ is in E-closure(q) and there is an edge labelled E from $q_1$ to $q_2$, then $q_2$ is added to E-closure (q), if $q_2$ is not already there. This is repeated until no more states can be added to E-closure (q).

Ex:

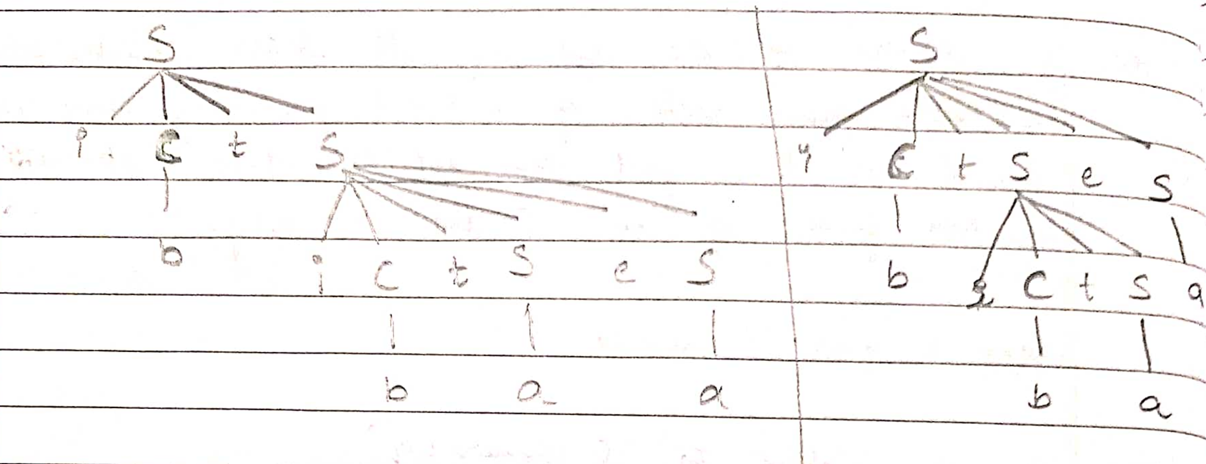



E-closure $(q_0) = \{q_0, q_1, q_2\}$
E-closure $(q_1) : \{q_1, q_2\}$
E-closure $(q_2) = \{q_2\}$

11Q. a. Check whether the following grammar is ambiguous.
$S \rightarrow iCtS \mid iCtSeS \mid a$        ibtibtaea
$C \rightarrow b$

| | |
|---|---|
| $S \rightarrow iCtS$ | $S \rightarrow iCtSeS$ |
| $\rightarrow ibtS$ | $\rightarrow ibtSeS$ |
| $\rightarrow ibtiCtSeS$ | $\rightarrow ibtiCtSeS$ |
| $\rightarrow ibtibtSeS$ | $\rightarrow ibtibtSeS$ |
| $\rightarrow ibtibtaeS$ | $\rightarrow ibtibtaeS$ |
| $\rightarrow ibtibtaea$ | $\rightarrow ibtibtaea$ |

The string ibtibtaea can be obtained by applying LMD in 2 different ways. Hence grammar is ambiguous.



b. Define Push Down Automata and language accepted by a PDA.

Ans. PDA is a finite automata with the addition of a stack. It has input tape, control unit and a stack with infinite size.

A PDA has 7 tuples: $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$

where, $Q$ = Set of Finite states

$\Sigma$ = Set of Input alphabets

$\Gamma$ = Set of stack alphabets

$\delta$ = Transitions from $Q \times (\Sigma \cup \epsilon) \times \Gamma$ to finite subset of $Q \times \Gamma^*$

$q_0$ = start state

$Z_0$ = Initial symbol of stack

$F$ = Set of ~~final states~~ final states.

The language accepted by PDA are:

1. Final state acceptance - In final state acceptance, the language accepted is the set of all inputs for which some choice of moves causes the PDA to enter a final state.

2. Empty stack acceptance:- In empty stack acceptance, the language accepted is the set of all inputs for which some sequence of moves causes the PDA to empty its stack.

12(a). Convert the following grammar into CNF. ($A \to BC$ or $A \to a$)

$S \to bB \mid aA$

$A \to aBB \mid aS \mid bA \mid a$

$B \to aBB \mid bS \mid b$

Ans Step-1: There is no start symbol on RHS.

Step-2: There are no unit, null and useless symbols.

Step-3: Replace the terminals with non-terminals.

$S \to bB \mid aA \Rightarrow S \to YB \mid XA$ [$\because Y \to b, X \to a$]

$A \to aBB \mid aS \mid bA \mid a \Rightarrow A \to aBB$ [$\because X \to a$

$\to \underbrace{XB}B$ [$\because Z \to XB$]

$\to ZB$

$A \to aS$ [$\because X \to a$]

$\to XS$

$A \to bA$ [$\because Y \to b$]

$\to YA$

$A \to a$

$B \to aBB \mid bS \mid b \Rightarrow B \to aBB$ [$\because X \to a$]

$\to \underbrace{XB}B$ [$\because Z \to XB$]

$\to ZB$

$B \to bS$ [$Y \to b$]

$\to YS$

$B \to b$

The grammar is CNF $G' = (V', T', P', S)$

where $V' = \{S, A, B, X, Y, Z\}$

$T' = \{a, b\}$

$S = S$

$P' = \{ S \to YB \mid XA$

$A \to ZB \mid XS \mid YA \mid a$

$B \to ZB \mid YS \mid b$

$Y \to b$

$X \to a$

$Z \to XB \}$

b. Explain GNF with example.

Ans: GNF stands for Greibach Normal Form. In GNF there is no restriction on the length of RHS of a production but restrictions occur on the position in which terminals and variables appear.

Let $G = (V, T, P, S)$ be a CFG. The CFG is said to be in GNF if all the productions are of the form:

$A \to a\alpha$ where $a \in T$ and $\alpha \in V^*$

i.e. The first symbol on the right hand side of the production should be a terminal followed by zero or more number of variables.

Example: The grammar

$S \to aA \mid bBB \mid bB$

$A \to aA \mid bB \mid b$

$B \to b$

is in GNF since the RHS of all the productions is a terminal followed by zero or more variables.

13Q. a. Explain Turing machines that accepts empty language.

Ans: A Turing machine is a finite state machine with an infinite tape and a tape head that can read or write.

The empty language, denoted by $\phi$ or $\{\}$, is a special language that contains no strings. An empty language can be accepted by a TM that never halts or rejects any input string regardless of what symbols are on the tape.

- To construct a TM that accepts the empty language, we can design a machine with a single state and a transition function that loops indefinitely on any input symbol.

- The machine has a single state $q_0$.

- The tape alphabet contains a symbol set consisting of all possible input symbols, including blanks.

- The initial tape is blank, and the read/write head starts at the leftmost cell.

- Transition function is : For any input symbol encountered on the tape, the machine stays in $q_0$, moves the read/write head one cell to the right, and writes the same symbol back to the tape.

- The machine remains in state $q_0$ indefintely, regardless of the input symbols encountered.

- There is no accepting state defined for this machine.

With this design, TM will continuously move back and forth on the tape, never halting or accepting any input string. Therefore, it accepts the empty language because it never reaches an accepting state for any input.

b. Define Post Correspondence Problem with an example.

Ans: The Post Correspondence Problem is defined as:
Given two sequences of $n$ strings on some alphabet $\Sigma$ say,

$A = w_1, w_2, \ldots w_n$

and $B = v_1, v_2 \ldots v_n$.

we say that there exists a PCP for pair $(A, B)$ if there is a non empty sequence of integers $i, j \ldots k$ such that $w_i, w_j \ldots w_k = v_i, v_j \ldots v_k$.

The PCP problem is to device an algorithm that will tell us, for any $(A, B)$ whether or not there exists a solution. If there exists a solution to PCP, there exist infinitely many solutions.

Example :- Let $\Sigma = \{0, 1\}$

$A = \{w_1, w_2, w_3\}$

where $w_1 = 11$

$w_2 = 100$

$w_3 = 111$

and $B = \{v_1, v_2, v_3\}$

where $v_1 = 111$

$v_2 = 001$

$v_3 = 11$

For the pair $(A, B)$ there is PCP solution as :

$$\overbrace{1\ 1}^{w_1}\ \overbrace{1\ 0\ 0}^{w_2}\ \overbrace{1\ 1\ 1}^{w_3}$$

$$\underbrace{1\ 1\ 1}_{v_1}\ \underbrace{0\ 0\ 1}_{v_2}\ \underbrace{1\ 1}_{v_3}$$

If we take,

$w_1 = 00$,

$w_2 = 001$

$w_3 = 1000$

$v_1 = 0$

$v_2 = 11$

$v_3 = 011$

there cannot be any post correspondence solutions because any string composed of elements of A will be longer then the corresponding string from B.

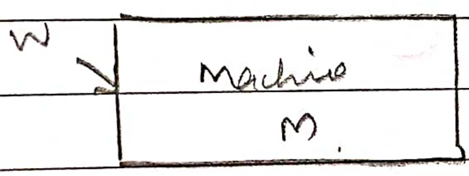
14Q. Write short notes on:

a. Halting problem.

Ans: ~~Alan Turing proved in 1936, that~~ a
In computability theory HM is a problem of determining form and description for a computer program and an input, whether the program will finish running or continue forever. It is an important problem which is not recursively enumerable that is unsolvable/undecidable decision problem.

It can be stated as "Given a TM M and an input string w with the initial configuration $q_0$, after some computations do the machine M halts?"

Given an arbitrary TM $M = (Q, \Sigma, \Gamma, \delta, q_0, B, A)$ and input $w \in \Sigma^*$, does M halt on input w?



Aug / Sept 2021

Part - A

1. Define $\epsilon$-NFA .

Ans: $\epsilon$-NFA is 5 tuple : $\epsilon$ : {Q, $\Sigma$, $\delta$, q0, F }

where Q: set of finite set of states

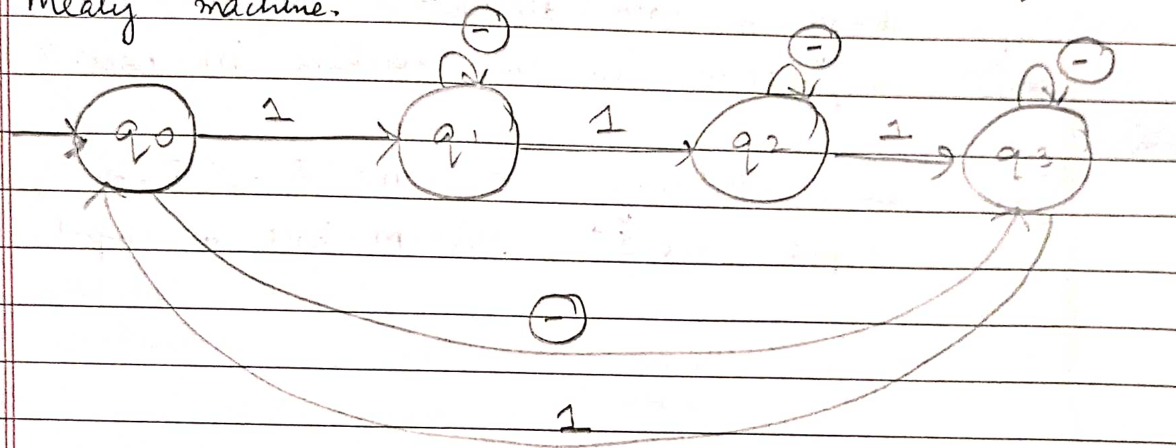
$\Sigma$: ~~finite set of~~ Input finite set of input

q0: Initial state

F: final state

$\delta : Q \times (\Sigma \cup \epsilon) \rightarrow 2^Q$

6Q. Convert the following Moore Machine to an equivalent Mealy machine.




Given $\lambda(q_0) = a$,

$\lambda(q_1) = b$

$\lambda(q_2) = c$

$\lambda(q_3) = d$.

Ans

18. Explain Turing Machine and Instantaneous description of Turing machine.

Ans. A Turing Machine (TM) is a seven tuple $M: (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ where, $Q$ = set of finite states

$\Sigma$ = set of input symbols

$\Gamma$ = set of tape symbols

$\delta$ - Transitions from $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$

$q_0 \in Q$. is the start state of $M$

$B$ - Special symbol indicating blank character

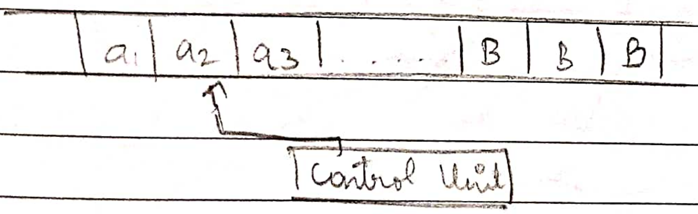
$F$ ~~$\subseteq Q$~~ - Set of final states.

TM is a finite automaton with :

1. Tape : It is used to store information and is divided into cells. Each cell can store only one symbol.

2. Read write head : It can read/write a symbol from where it points i.e. it scans one cell of the tape at a time.

3. Control unit : It is a finite set of states. It carries out the tasks based on transition table.




Instantaneous Description of a TM : An ID of a TM is a string $x\,q\,y$ where $q$ is the current state, $xy$ is the string made from tape symbols. The head points to the first character of the substring.

Example: $q\,x\,y \vdash^{*} x\,q\,y$

If there is a transition $\delta(q, x) = (q, x, R)$

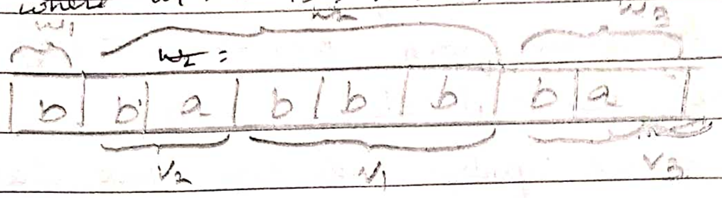
80. Check whether the list $A = \{b, bab^3, ba\}$ and $B = \{b^3, ba, ba\}$ have a PCP solution.

Ans. For the pair $(A, B)$ there is a solution :

Let $\Sigma = \{a, b\}$

$A = \{$~~$w_1, w_2, w_3$~~$\}$ $\{b, babbb, ba\}$

~~where $w_1 = b$~~ $B = \{bbb, ba, ba\}$



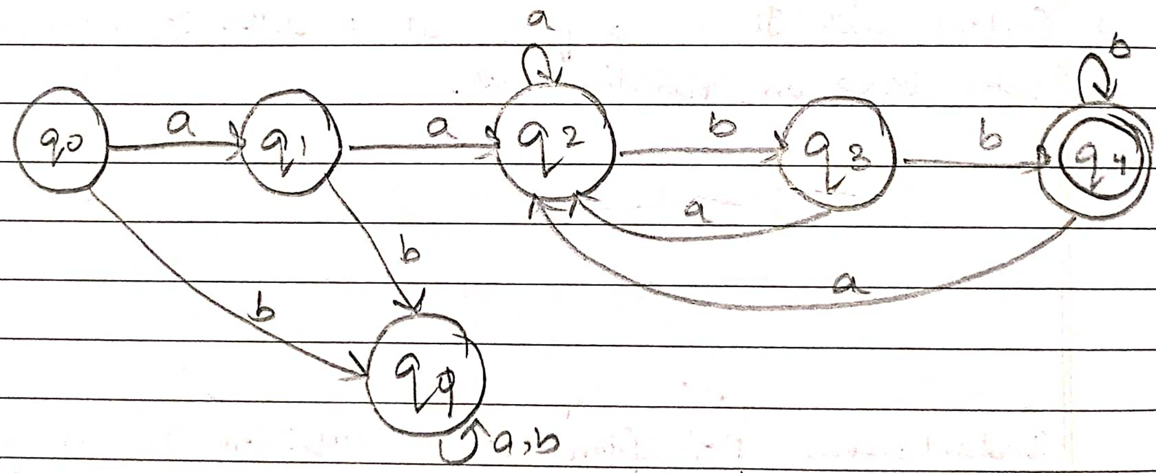

Part- B

9R. a. Design a DFA to accept strings of a's and b's starting with atleast two a's and ending with atleast two b's.

Ans Step 1: minimum string = aabb

Step 2: $\Sigma = \{a, b\}$

Step 3:




DFA is defined as :

$D = (Q, \Sigma, \delta, q_0, F)$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_\phi\}$

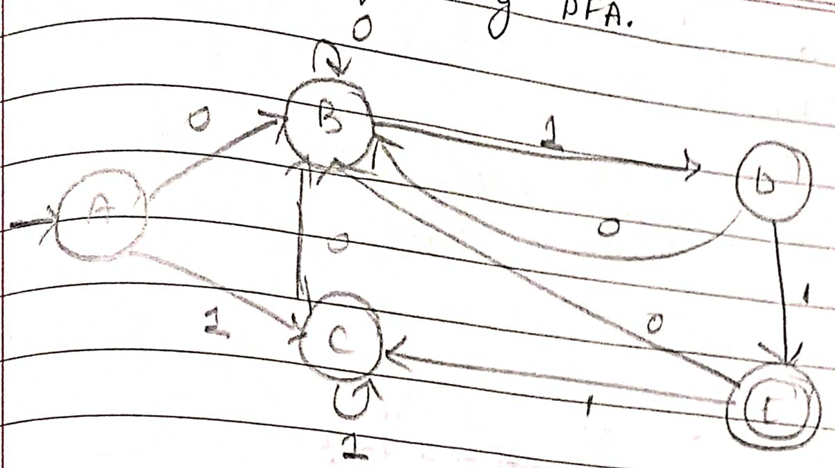
$\Sigma = \{a, b\}$

$q_0 : q_0$

$F = q_4$

| $\delta$ | a | b |
|---|---|---|
| $q_0$ | $q_1$ | $q_\phi$ |
| $q_1$ | $q_2$ | $q_\phi$ |
| $q_2$ | $q_2$ | $q_3$ |
| $q_3$ | $q_2$ | $q_4$ |
| $q_4$ | $q_4$ | $q_2$ |
| $q_\phi$ | $q_\phi$ | $q_\phi$ |

b) Minimise the following DFA.



Ans: Step 1: Construct Transition Table

| δ | 0 | 1 |
|---|---|---|
| A | B | C |
| B | B | D |
| C | B | C |
| D | B | E |
| * E | B | C |

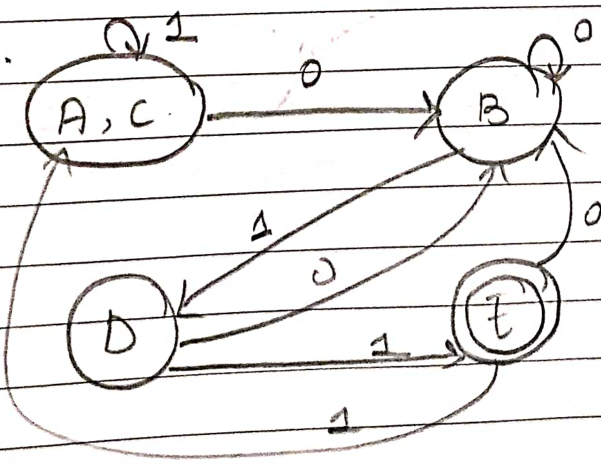
Step 2: Zero - equivalence : {A, B, C, D} {E}

One - equivalence: {A, B, C} {D} {E}

Two - equivalence: {A, C} {B} {D} {E}

Three - equivalence: {A, C} {B} {D} {E}

Hence we stop.



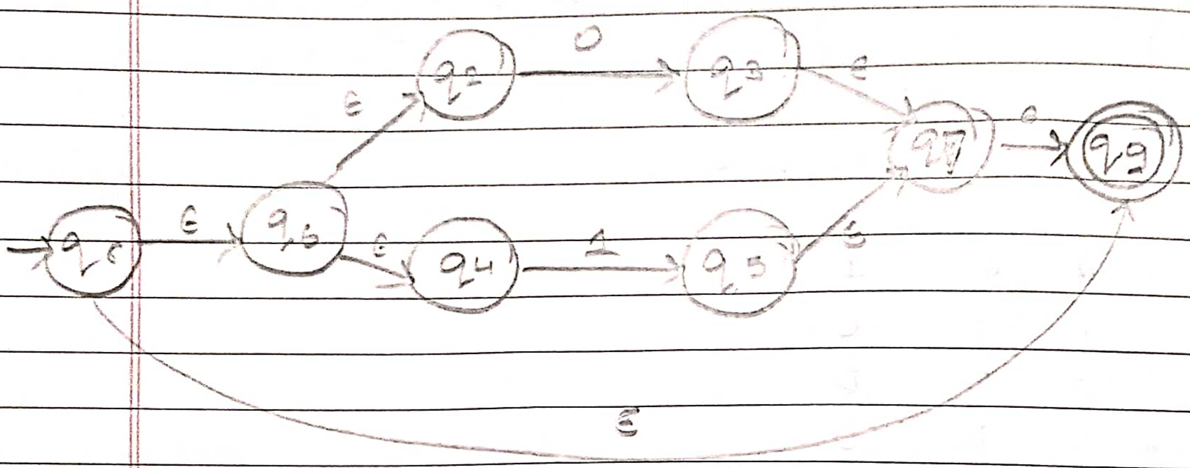
Transition table

| | 0 | 1 |
|---|---|---|
| AC | B | AC |
| B | B | D |
| D | B | E |
| E | B | AC |

12. Construct an ε-NFA for the regular expression $0(0+1)^*\ 01$.

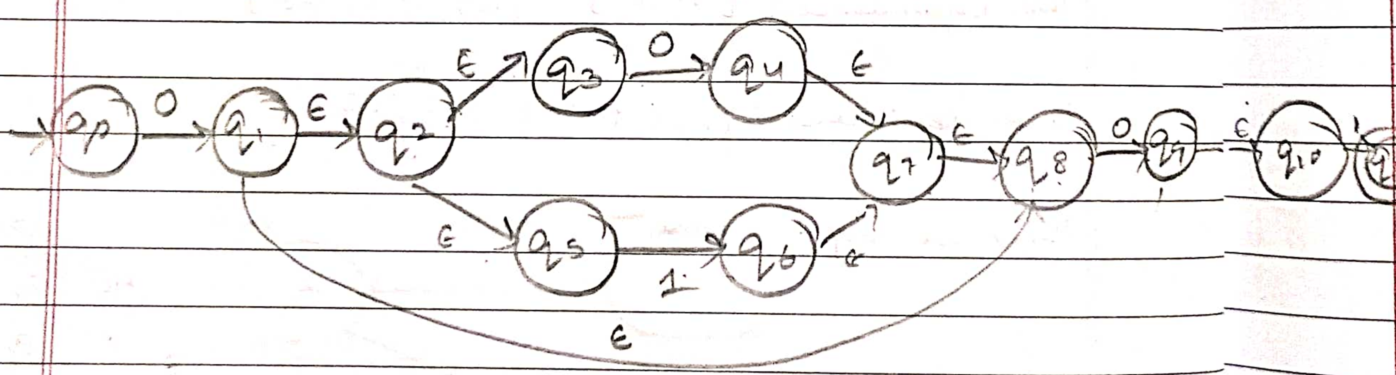
To accept 0:

To accept $(0+1)^*$



To accept 01

ε-NFA for $0(0+1)^*\ 01$.



b Prove ~~that~~ context free ~~languagg~~ languages are closed under union, concatenation and star.

Ans. CFL are closed under Union, Concatenation and Kleen Closure (star).

1. Union

- Consider 2 languages $L_1, L_2$

$L_1 = \{a^n \mid n \geq 0\}$
$L_2 = \{b^n \mid n \geq 0\}$

$L_1 = \{\epsilon, a, aa, aaa \dots\}$
$L_2 = \{\epsilon, b, bb, bbb \dots\}$

If $L_1$ and $L_2$ are CFL then $L_1 \cup L_2$ is also CFL.
$S_1 \Rightarrow aS_1 \mid \epsilon$
$S_2 \Rightarrow bS_2 \mid \epsilon$
$S \Rightarrow S_1 \cup S_2$.

2. Concatenation

$L_1 = \{a^n \mid n \geq 0\}$
$L_2 = \{b^n \mid n \geq 0\}$

9,10 1 $L_1 = \{\epsilon, a, aa, aaa\}$
$L_2 = \{\epsilon, b, bb, bbb \dots\}$

If $L_1$ and $L_2$ are CFL then $L_1 \cap L_2$ is also CFL.

$S_1 \Rightarrow aS_1 \mid \epsilon$
$S_2 \Rightarrow bS_2 \mid \epsilon$
~~$S \Rightarrow S_1 \cdot S_2$~~. $S \Rightarrow S_1 \cdot S_2$

3. Kleen Closure | star

$L_1 = \{a^n \mid n \geq 0\}$
$L_2 = \{b^n \mid n \geq 0\}$

$L_1 = \{\epsilon, a, aa, aaa \ldots\}$
$L_2 = \{\epsilon, b, bb, bbb \ldots\}$

$L_1^* = L^0 \cup L^1 \cup L^2 \cup L^3$
$= \epsilon \cup a \cup aa \cup aaa \ldots$
$L_2^* = L^0 \cup L^1 \cup L^2 \cup L^3 \ldots$
$= \epsilon \cup b \cup bb \cup bbb \ldots$

11Q.a. Design a PDA to accept the language $L = \{0^n, 2^n \mid n \geq 1\}$ and check whether it accepts the string 0011 and 0111.

Ans.

120. a. Convert the following grammar into equivalent CNF.

$S \to 0A \mid 1B$

$A \to 0AA \mid 1S \mid 1$

$B \to 1BB \mid 0S \mid 0$.

Ans. Step 1: No useless symbols, $\epsilon$-productions and unit production.

Step 2: Replace terminals by variables

$S \to 0A \mid 1B \Rightarrow XA \mid YB$ [$\because X \to 0,\ Y \to 1$]

$A \to 0AA \mid 1S \mid 1 \Rightarrow \underbrace{XA}A \mid YS \mid 1$

$= ZA$ [$\because Z = XA$]

$B \to 1BB \mid 0S \mid 0 \Rightarrow \underbrace{YB}B \mid XS \mid 0$

$= WB$ [$\because W = YB$]

The equivalent grammar in CNF $G' = (V', T', P', S)$

where $V' = \{S, A, B, X, Y, Z, W\}$

$T' = \{0, 1\}$

$S = S$

$P' = \{ S \to XA \mid YB$

$A \to ZA \mid YS \mid 1$

$B \to WB \mid XS \mid 0$

$X \to 0$

$Y \to 1$

$Z \to XA$

$W \to YB \}$

b. ~~Define GNF and~~ Write the steps to convert CFG to GNF.

Ans. Let $G = (V, T, P, S)$ be a CFG, generating the language L without $\epsilon$. An equivalent grammar G1 generating the same language exists for which each production is of the form $A \to a\alpha$

Step-1: Obtain the grammar in CNF

Step-2: Rename the non-terminals to $A_1, A_2, A_3 \ldots$

Step-3: Using substitution method, obtain the production to be the form
$A_i \rightarrow A_j \alpha$ for $i < j$
where $\alpha \in V^*$.

Step-4: After substitution, if the grammar has left recursion, eliminate left recursion.

Step-5: Repeat step 3 and step-4 to get the grammar in GNF.

13Q Q. Write short notes on Rice theorem.

Ans: Rice theorem states that any non-trivial ~~somatic~~ semantic property of a language which is recognised by a TM is undecidable. A property, P, is the language of all TM that satisfy that property.

Definition: If P is a non-trivial property, and the language holding the property, $L_p$, is recognised by TM M, then $L_p = \{ \langle M \rangle \mid L(M) \in P \}$ is undecidable.

Proof: Suppose, a property P is non-trivial and $\varphi \in P$
Since, P is non-trivial, at least one language satisfies P, i.e. $L(M_0) \in P$, $\exists$ Turing Machine $M_0$.

Let, w ~~be~~ be an input in a particular instant and N is a TM which follows:
On input x:

- Run M on w

- If M does not accept ~~for does~~, then do not accept x.

- If M accepts w then run $M_0$ on x. If $M_0$ accepts x, then accept x.

A function that maps an instance ~~ATM = {<M, w> | M~~ ATM = {<M, w> | M accepts input w} to a N such that

- If M accepts w and N accepts the same language as $M_0$, then $L(M) = L(M_0) \in P$

- If M does not accept w and N accepts $\varphi$, then $L(N) = \varphi \notin P$

Since $A_{TM}$ is undecidable and it can be reduced to $L_P$, $L_P$ is also undecidable.

b. Obtain a PDA for the CFG given below:

$S \rightarrow aABB \mid aAA$

$A \rightarrow aBB \mid a$

$B \rightarrow bBB \mid A$

$C \rightarrow a$.

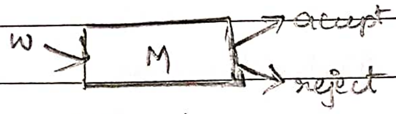
Ans:

11. Q- Define Recursive and Recursively Enumerable languages and prove that the union of two recursive languages is recursive.

Ans Recursive and Recursively Enumerable languages are associate with Turing machines.

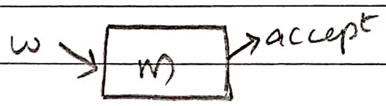
A language, L is recursive if $L = L(M)$ for some Turing Machine, M such that

- If w is in L, then M accepts
- If w is not in L, then M eventually halts and never enters an accepting state.




TM for a recursive language

A Language L is said to be recursive enumerable if there exists a turing machine that accepts it.

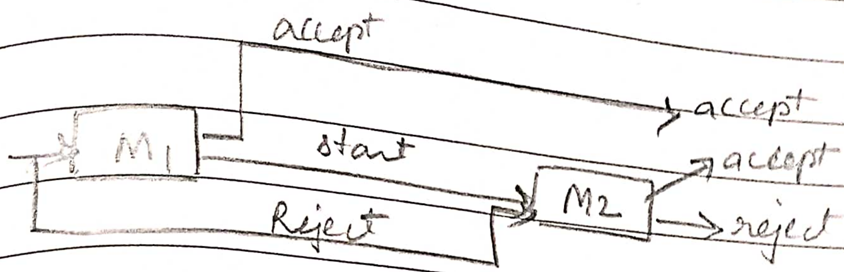



TM for a Recursively enumerable language

i.e. L is recursively enumerable if $L = L(M)$ for some turing machine, M. It doesn't say anything about w that is not in L.

The Union of two Recursive language is recursive.

Proof: Let $L_1$ and $L_2$ be recursive languages accepted by algorithms $M_1$ and $M_2$ we construct M, which first simulates $M_1$. If $M_1$ accepts, then M accepts. If $M_1$ rejects then M simulates $M_2$ and accepts if and only if $M_2$ accepts. Since both $M_1$ and $M_2$ are algorithms, M is guranteed to halt. ~~Clearly~~ M accepts $L_1 \cup L_2$.




Construction for Union of Recursive language.