72 Chaitra

1(OU1)* 1a) Define Cartesian Product. Use Mathematical Induction ~~&~~ to show

$1.1!+2.2!+3.3!+\dots+n.n!=(n+1)!-1$ for $n\geq 1$.

Cartesian Products.

~~Orde~~

Cartesian Products from A to B is a set of of all the ordered pairs from A to B

i.e. $A\times B=\{(a,b): a\in A, b\in B\}$.

To prove

$1.1!+2.2!+3.3!+\dots+n.n!=(n+1)!-1$

Step 1 Basic step

For $n=1$,

$LHS=1.1!=1$

$RHS=(1+1)!-1=2-1=1$

Step 2: Induction hypothesis

Let us assume for $n=k$, $k<n$

$1.1!+2.2!+\dots+k.k!=(k+1)!-1$

Step 3 Induction method.

For $n=k+1$

To prove,

$1.1!+2.2!+\dots+k.k!+(k+1).(k+1)!=(k+2)!-1$

$$LHS = 1\cdot 1! + 2\cdot 2! + 3 \ldots + k\cdot k! + (k+1)\cdot(k+1)!$$
$$= (k+1)! - 1 + (k+1)(k+1)!$$
$$= (k+1)!\,[1+k+1] - 1$$
$$= (k+1)!\,(k+2) - 1$$
$$= (k+2)! - 1$$
$$= RHS$$

So the result is true for $n = k+1$

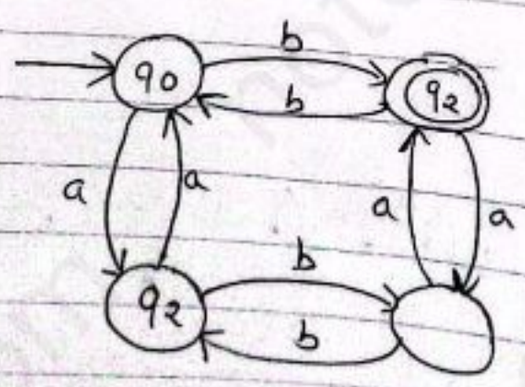
Hence ~~to~~ for $n = 1, 2, 3, \ldots, k, \ldots, n$ the result is true hence proved. //

(b) Find the regular expression for the language $L = \{ w \in \{0,1\}^*$, has 0101 as substring$\}$

$\Rightarrow$ RE is,

$$(0 \cup 1)^* \, 0101 \, (0 \cup 1)^*$$

(2) Construct a DFA over $\{a,b\}$ accepting string having even number of a & odd no. of 'b'



eg. $\vdash^{q_0}$ abb
$\vdash^{q_2}$ abb
$\vdash^{q_2}$ abb
$\vdash^{q_0}$ abb-
Not accepted

$\vdash^{q_0}$ aab
$\vdash^{q_2}$ aab
$\vdash^{q_0}$ aab.
$\vdash^{q_2}$ aab-
Accepted.

③ ~~Form~~ Define DFA formally. State & prove closure properties of regular language.

DFA
A deterministic finite automata is a quintuple
$M = (k, \Sigma, \delta, s, f)$
where,
k is the finite set of states
Σ is the alphabet
s is start symbol
f is final symbol
δ is the transition function, $k \times \Sigma \rightarrow k$.

Closure properties of regular language
① Union
② Intersection
③ Concatenation
④ Kleene star
⑤ Complementation.

(a) Union,

The language is closed under union

Let $M_1 = (k_1, \Sigma_1, \Delta_1, S_1, F_1)$

$M_2 = (k_2, \Sigma_2, \Delta_2, S_2, F_2)$

are two NFA

Let us construct a new NDFA, M such that

$L(M) = L(M_1) \cup L(M_2)$

Here,

$M = (k, \Sigma, \Delta, S, F)$

where,

$k = k_1 \cup k_2 \cup \{s\}$

$F = F_1 \cup F_2$

$\Delta = \Delta_1 \cup \Delta_2 \cup \{(s,e) \to s_1\} \cup \{(s,e) \to s_2\}$

So, regular language are closed under union.

(b) Concatenation,

Let M_1 & M_2 be two NFA,

We can construct a new NFA such that

$L(M) = L(M_1) \cdot L(M_2)$

Here,

$M = (k, \Sigma, \Delta, S, F)$

$k = k_1 \cup k_2$ $\qquad F = F_2$

$\Sigma = \Sigma_1 \cup \Sigma_2$

$\Delta = \Delta_1 \cup \Delta_2 \cup \{(F_1, e) \to S_2\}$

$S = S_1$

So, regular language is closed under concatenation

(c) Kleene star

Let M_1 ~~& M_2~~ be ~~two~~ a NFA.

We can construct a new NFA M such that
$L(M) = L(M_1)^*$

Here, $M = (k, \Sigma, \Delta, S, F)$
where,

$k = k_1$
$\Sigma = \Sigma_1$
$\Delta = \Delta_1 \cup \{(F_1, e) \longrightarrow s_1\}$
$s = s_1$
$F = F_1$

So, regular language is closed under kleene star.

(d) Complementation

Let $M = (k, \Sigma, \delta, S, F)$ be a DFA then
we can construct a new DFA that accept the complementary language of M

i.e. $\overline{M} = (k, \Sigma, \delta, S, k - F)$

(e) Intersection.

Let L_1 & L_2 be two regular languages accepted by FA M_1 & M_2 then

$$L_1 \cap L_2 = \Sigma^* - ((\Sigma^* - L_1) \cup (\Sigma^* - L_2))$$

Since language is closed under union & complementation.
So the language is closed under intersection also.

④ Define pumping lemma for regular language. Use pumping lemma for regular language to show $L = \{a^n b a^n \text{ for } n = 0, 1, 2, \ldots\}$ is not regular.

Pumping lemma for regular language

Let L be regular language. And w be any string $w \in L$, n be a pumping constant with $n \geq 1$, $|w| \geq n$ then w can be written as $w = xys$ such that

$$|xy| \leq n$$
$$|y| \geq 1$$
$$xy^iz \in L \text{ for all } i \geq 0.$$

$L = \{a^n b a^n\}$

For pumping lemma, we can write

$w = xyz = a^n b a^n$

which can be written as,

$$w = xyz = \underbrace{aaaa\ldots aa}_{x}\ \underbrace{aa\ldots aa}_{y}\ \underbrace{aa\ldots aabaa\ldots aa}_{z}$$

Here, $x z a\ y = a^k$, $k \geq 1$

Using pumping lemma,

For $i = 0$, we get

$$xz = a^{n-k} b a^n \notin L$$

Also for $i = 2$,

$$xy^2z = a^{n+k} b a^n \notin L$$

which contradicts the theorem.

Sol: $\{a^n b a^n, n = 0, 1, \ldots\}$ is not a regular language.

(5) Define the configuration of PDA. Design the PDA that accepts $L = \{a^{3n} b^n, n > 0\}$ & check the string aaaaaabb.

Configuration of PDA.

Configuration of PDA is defined as member of $k \times \Sigma^* \times \Gamma^*$ is triple (q, w, γ) where

q is state

w is remaining input

γ is the stack content.

$M = (k, \Sigma, \Gamma, \Delta, s, f)$

where,

$k = \{$

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

$\Gamma = \{a$

$s = \{P\}$

$f = \{f\}$

$\Delta = \{ (P, a, e) \longrightarrow (Q, a)$ — (1)

$(P, a, a) \longrightarrow (Q, a)$ — (2)

$(Q, a, a) \longrightarrow (R, e)$ — (3)

$(R, a, a) \longrightarrow (P, a)$ — (4)

$(R, a, e) \longrightarrow (P, a)$ — (5)

$(P, b, a) \longrightarrow (P, e)$ — (6)

$(P, e, e) \longrightarrow (f, e)$ — (7)

$\}$

For string aaaaaabb.

State	Input	Stack	Transition used
P	aaaaaabb	–	–
Q	aaaaaabb	a	1
R	aaaaaabb	–	3
P	aaaaaabb	a	5
Q	aaaaaabb	aa	2
R	aaaaaabb	a	3
P	aaaaaabb	aa	4
P	aaaaaabb	a	6
P	aaaaaabb_	–	6
f	aaaaaabb_	–	7

Accepted.

6) Define context free grammer. Convert the given Context Free Grammar (CFG) into equivalent CNF

$S \rightarrow AB$

$A \rightarrow aAA/e$

$B \rightarrow bBB/e$

- Here e is empty symbol

⇒ A context free Grammar is ~~defined as~~ quadrupole
$G = (V, \Sigma, R, S)$ where
V is an alphabet
Σ is set of terminals
R is the set of rules $(V-\Sigma) \times V^*$
S is the start symbol, element of $V-\Sigma$

Given CFG is,
$S \rightarrow AB$
$A \rightarrow aAA / e$
$B \rightarrow bBB / e$

Removing null production from A

$S \rightarrow AB / B$
$A \rightarrow aAA / aA / a$
$B \rightarrow bBB / e$

Removing null production from B

$S \rightarrow AB / B / A / e$
$A \rightarrow aAA / aA / a$
$B \rightarrow bBB / bB / b$

Now ~~removm~~ removing unit production of A

$S \to AB / B / aAA / aA / a$
$A \to aAA / aA / a$
$B \to bBB / bB / b$

Removing unit production of B

$S \to AB / bBB / bB / b / aAA / aA / a$
$A \to aAA / aA / a$
$B \to bBB / bB / b$

$\boxed{S \to AB}$

$S \to bBB \Rightarrow \boxed{S \to PB}$

$P \to bB \Rightarrow \boxed{P \to T_b B}$

$\boxed{T_b \to b}$

$S \to bB \Rightarrow \boxed{S \to T_b B}$

$\boxed{S \to b}$

$S \to aAA \Rightarrow \boxed{S \to QA}$

$Q \to aA \Rightarrow \boxed{Q \to T_a A}$

$\boxed{T_a \to a}$

$S \to aA \Rightarrow \boxed{S \to T_a A}$

$\boxed{S \to a}$

$A \to aAA \Rightarrow \boxed{A \to QA}$

$A \to aA \Rightarrow \boxed{A \to T_a A}$

$\boxed{A \to a}$

$B \to bBB \Rightarrow \boxed{B \to PB}$

$B \to bB \Rightarrow \boxed{B \to T_b B}$

$\boxed{B \to b}$

Now, CNF is,

$G = (V, \Sigma, R, S)$

$V = \{S, A, B, P, Q, T_a, T_b, a, b, c\}$

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

$R = \{ S \to AB / PB / T_b B / QA / T_a A / a / b$

$A \to QA / T_a A / a$

$B \to PB / T_b B / b$

$P \to T_b B$

$Q \to T_a A$

$T_a \to a$

$T_b \to b \}$

of an expandable gate? We c
bubble and arrow to arrow.
2.34a connected to the arrow and bubb
This means that the expander outputs ar
gate. In other words, Fig. 2.34b is equivale

9a) Write a CFG for regular expression,

$R = 0^* 1 (0 \cup 1)^*$

$G = (V, \Sigma, R, S)$
$V = \{S,$
$\Sigma = \{0, 1\}$
$R = \{ S \rightarrow 0S / 1A$
$A \rightarrow 0A / 1A / e$
$\}$

(b) Use concept of closure properties to prove that intersection of CFL is not Context Free.

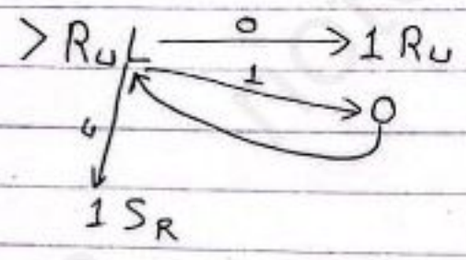
Let $L_1 = \{a^n b^n c^n, n \geq 1, m \geq 1\}$
~~$L_2 = \{ba^m b^m c^m$~~
$L_2 = \{a^n b^n c^m, n \geq 1, m \geq 1\}$

Grammar for L_1 is, $f = \{ S \rightarrow AB$
$A \rightarrow aAb / ab$
$B \rightarrow cB / c \}$

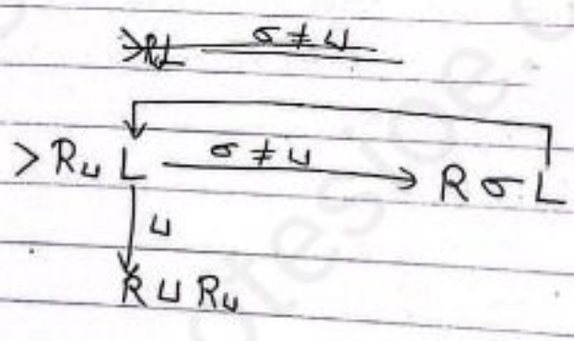
Grammar for L_2 is, $f = \{ S \rightarrow AB$
$A \rightarrow aA / a$
$B \rightarrow bBC / bc$

$L = \{a^n b^n c^n, n \geq 1\} = L_1 \cup L_2$
is not CFL.
So, CFL are not closed under intersection

(8) Design a TM to compute $f(n) = n + 1$,
n be a binary string
Show processing for 10111



where, S_R is

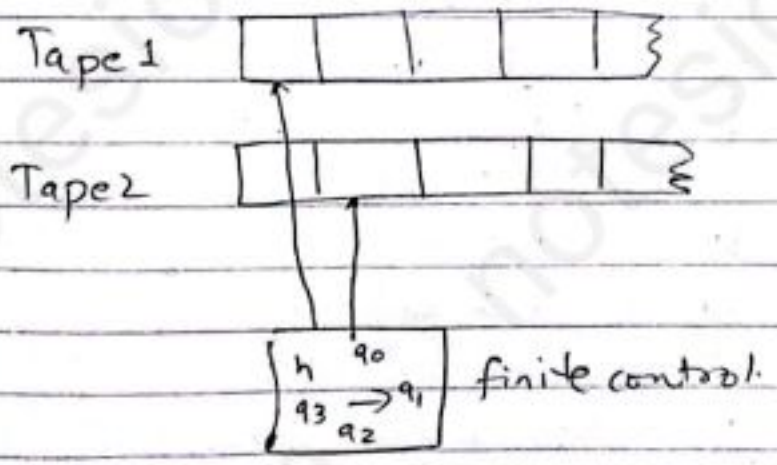


$\vdash$ ⊔10111⊔
$\vdash^{R_\sqcup}$ ⊔10111⊔
$\vdash^{L}$ ⊔10111⊔
$\vdash^{1}$ ⊔10110⊔
$\vdash^{L}$ ⊔10110⊔
$\vdash^{1}$ ⊔10100⊔
$\vdash^{L}$ ⊔10100⊔
$\vdash^{1}$ ⊔10000⊔
$\vdash^{L}$ ⊔10000⊔
$\vdash^{1}$ ⊔11000⊔ $\vdash^{R_\sqcup}$ ⊔11000⊔ #

(9) Define Multitape Turing Machine. With the help of suitable example, explain how Universal Turing machine works

Multitape Turing Machine.
It is a turing machine with several tapes, each tape is connected to finite control of head individually.

The machine can in one step read the symbol scanned by all its heads and depending on those symbols & current state can rewrite the scanned symbols move its head to left or right & change states



example,
We can define Turing machine not only as hardware but as software also. Such interpretation of Turing Machine with software is called Universal Turing machine.

example.

Let us consider a Turing machin
$M(k, \Sigma, \delta, s, h)$ where,
$k = \{s, q, h\}$
$\Sigma = \{\sqcup, \triangleright, a\}$
$\delta \Rightarrow$

state	symbol	δ
s	a	$(q, \sqcup)$
s	$\sqcup$	$(h, \sqcup)$
s	$\triangleright$	$(s, \rightarrow)$
q	a	(s, a)
q	$\triangleright$	$(q, \rightarrow)$
q	$\sqcup$	$(s, \sqcup)$

Since There are only 3 states & 3 symbols,
$\therefore i = 2 \quad j = 3 \quad [\because 2^i \geq |k| \ \& \ 2^j \geq |\Sigma| + 2]$

(10) State Church Turing Thesis. What is a recursive language?

Church Turing Thesis

It is believed that there are no functions, that can be defined by humans whose calculations can be described by any well-defined mathematical algorithm that can be performed, that cannot be computed by Turing machine.

In other words,
No computational procedure will be considered as an algorithm unless it is represented by a Turing machine.

Recursive Language

A language is recursive if there exists a Turing Machine that accepts every string of the language and rejects every string that are not in the language

Recursive language always halts the Turing machine.

Halting problem

Halting problem is to determine for an arbitary turing machine Tm & input w, whether Tm will eventually halt on input w.

We can write a program that takes an input X. The program can determine whether P halts by returning 'yes' or 'no' and run forever. The program can be named as halt(P,X)

Let us assume a Turing machine Tm that exists with input string w having initial configuration $q_0 w$. We cannot find ~~the answer by simulating~~ that the machine eventually haults or not by simulating the action of Tm on w because there is no limit ~~of~~ on the length of computation. So we need is an algorithm that can determine the correct answer for any Tm and w by performing some analysis on the machine's description & the input. But it is ~~eta~~ clear that there is no such algorithm that exists.

(12) Write short notes on:

(a) Computational Complexities.

Computational complexity theory is a branch of theory of computation in computer science & mathematics that focuses on classifying computational problems according to their inherent ~~def~~ difficulty.

In other words, computational complexities theory is a part of computation dealing with the resources required during computation to solve a given problem

The most common resources are.

- Time complexity
  It focuses on how many steps does it take to solve a given problem

- Space complexity
  It focuses on how much memory is acquired by with a given problem is solved.

(b) NP hard & NP Complete Problems.

NP ~~hard~~ complete Problem

~~Some problems are so hard that we can~~ prove condition

Let L be a problem in NP. We say L is NP-complete if the following statements are true.

- L is in NP
- For every language L' in NP, there is a polynomial time reduction of L' to L.

NP hard Problem

Some problem are so hard that we can prove condition ② of the definition of the NP completeness (i.e. every language in NP reduces to L in polynomial time) but we cannot prove condition ①, that L is in NP. If so, we call L as NP hard problem

71 Chaitra

① Define countably infinite & uncountable sets with example. Use principle of mathematical induction to prove $(5^n - 1)$ is divisible by 4 for all integers $n \geq 0$.

Countably infinite
A set ~~of~~ is said to be countably infinite if it is equinumerous with N

Let $N \times N$ is the union of $\{0\} \times N$, $\{1\} \times N$, $\{2\} \times N$ and so on then the union of this is, union of countably infinite collection of countably infinite set.

To prove: $(5^n - 1)$ is divisible by 4 for all integers.

Step 1: Basic step
For $n = 1$
$LHS = 5^1 - 1$
$= 4$ which is divisible by 4.

Step 2: Induction hypothesis
Let us assume ~~to~~ that for $n = k$,
$(5^k - 1) = 4M$ which is divisible by 4

Step3: Induction method

For $n=k+1$

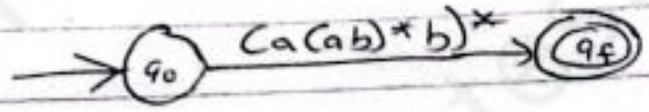
To poove, $(5^{k+1}-1)$ is also divisible by 5

$$
\begin{aligned}
LHS &= (5^{k+1}-1) \\
&= 5^k \cdot 5 - 1 \\
&= (4M+1)5 - 1 \\
&= 20M + 5 - 1 \\
&= 20M + 4 \\
&= 4(5M+1)
\end{aligned}
$$

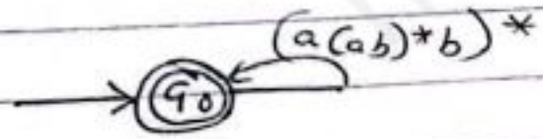
which also divisible by 4

So, for $n = 1, 2, 3, \ldots, k, k+1, \ldots n$, $(5^n - 1)$ is divisible by 4. //

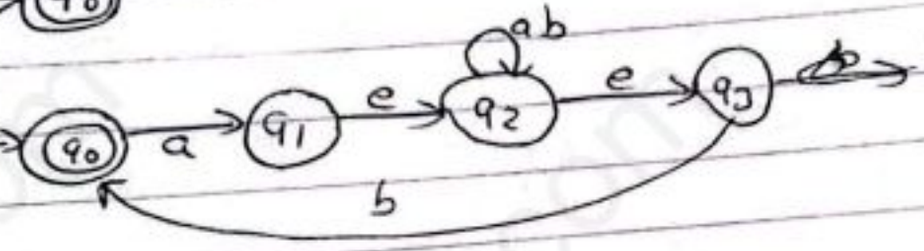
② Design a DFA for RE $(a(ab)^*b)^*$. Verify your design by taking one accepted & one rejected string



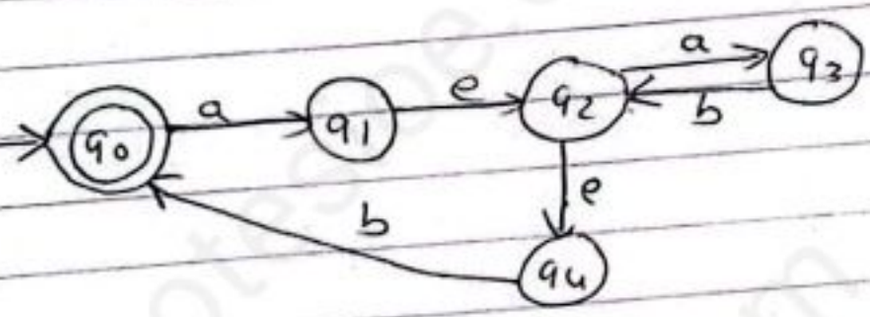
⇒



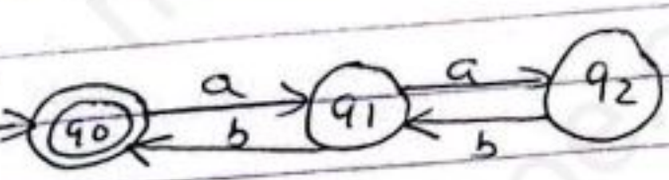
⇒



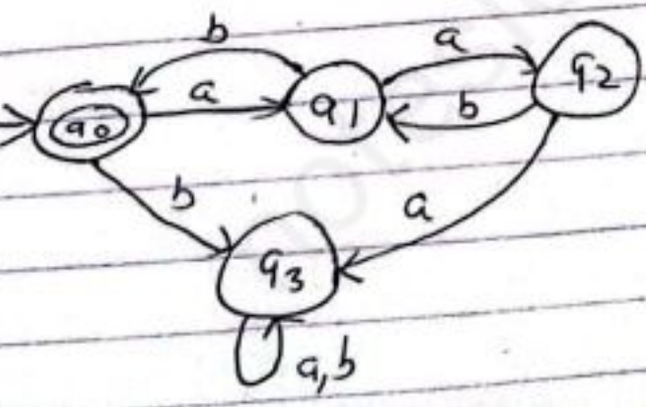
⇒



⇒



⇒



$\therefore M = (k, \Sigma, R, S, F)$

$k = \{q_0, q_1, q_2, q_3\}$

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

$S = q_0$

$f = q_0$

$R \Rightarrow$

$\delta \backslash \Sigma$	a	b
$\rightarrow^* q_0$	q_1	q_3
q_1	q_2	q_0
q_2	q_3	q_2
q_3	q_3	q_3

é.g. For, aababb

$\vdash q_0, \underline{a}ababb$

$\vdash q_1, a\underline{a}babb$

$\vdash q_2, aa\underline{b}abb$

$\vdash q_1, aab\underline{a}bb$

$\vdash q_2, aaba\underline{b}b$

$\vdash q_1, aabab\underline{b}$

$\vdash q_0, aababb_$

Accepted

For aaabb

$\vdash q_0, \underline{a}aabb$

$\vdash q_1, a\underline{a}abb$

$\vdash q_2, aa\underline{a}bb$

$\vdash q_3, aaa\underline{b}b$

$\vdash q_3, aaab\underline{b}$

$\vdash q_3, aaabb_$

Rejected.

③ State pumping lemma for regular language. Using this lemma to prove language $L = \{a^{n^2} : n > 0\}$ is not regular.

Pumping Lemma

Let L be a regular language

Let w be any string $w \in L$.

& n be any pumping constant with $n > 1$ & $|w| > n$

then w can be rewritten as,

$w = xyz$ & such that $|xy| \leq n$

$|y| \geq 1$

$xy^iz \in L$ for all $i \geq 0$

Here, $L = \{a^{n^2} : n > 0\}$

$w = a^{n^2}$

$$w = xyz = a^{n^2} = \underbrace{aaa.aa}_{x}\underbrace{aa\ldots aa}_{y}\underbrace{aa\ldots aa}_{z}$$

Here, $y = a^k$

Using pumping lemma,

For $i = 2$

$xy^2z = xyyz = a^{n^2+k}$

If it is regular then $a^{n^2+k} \in L$

i.e. n^2+k is perfect ~~fa~~ square.

We use,

$$n^2+k \le n^2+n+n+1$$
$$\le n^2+2n+1$$
$$\le (n+1)^2$$

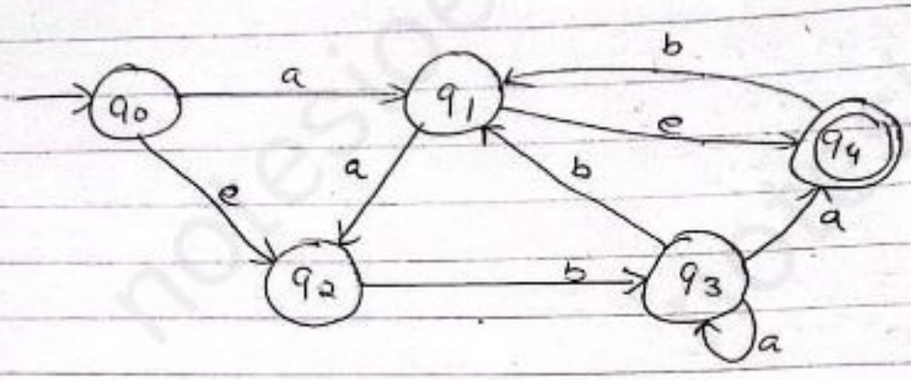
So, $n^2+k \ne (n+1)^2$

Hence $a^{n^2+k} \notin L$

$\therefore L=\{a^{n^2} : n>0\}$ is not a regular language.

④ Difference between DFA & NFA? Convert NFA to DFA

DFA	NFA
① DFA changes the state in such a way that is completely determined by the current state & input symbol	① NFA has ability to change states in a way that only partially determined by the current state & input symbol.
② There is only one fixed next state for a given combination of current symbol & next state.	② There may be several posibility of next state for given combination of current symbol & next input



$M = (k, \Sigma, \delta, f, s)$

$k = \{q_0, q_1, q_2, q_3, q_4\}$

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

$f = \{q_4\}$

$s = \{q_0\}$

δ is given by,

$\delta \mid \Sigma$	a	b	e-closure
→ q_0	q_1	-	q_0, q_2
q_1	q_2	-	q_1, q_4
q_2	-	q_3	q_2
q_3	q_3, q_4	q_1	q_3
* q_4	-	q_1	q_4

$f(\{q_0, q_2\}, a) \longrightarrow \{q_1\} \rightarrow \{q_1, q_4\}$ (NS)

$f(\{q_0, q_2\}, b) \longrightarrow \{q_3\}$ (NS)

$f(\{q_1, q_4\}, a) \longrightarrow \{q_2\}$ (NS)

$f(\{q_1, q_4\}, b) \longrightarrow \{q_1\} \rightarrow \{q_1, q_4\}$ (ES)

$f(\{q_3\}, a) \longrightarrow \{q_3, q_4\}$ (NS)

$f(\{q_3\}, b) \longrightarrow \{q_1\} \rightarrow \{q_1, q_4\}$ (ES)

$f(\{q_3, q_4\}, a) \longrightarrow \{q_3, q_4\}$ (ES)

$f(\{q_3, q_4\}, b) \longrightarrow \{q_1\} \rightarrow \{q_1, q_4\}$ (ES)

$f(\{q_2\}, a) \longrightarrow \phi$ (NS)

$f(\{q_2\}, b) \longrightarrow \{q_3\}$ (ES)

:M

What do we connect to the arrow and bubble inputs of an expandable gate? We connect the o
pander as in Fig. 2.34a. Connect bubble to bubble and arrow to arrow.

Visualize the outputs of Fig. 2.34a connected to the arrow and bubble inputs of Fig. 2.33. I
shows the logic circuit. This means that the expander outputs are being ORed with the signals o
OR-INVERT gate. In other words, Fig. 2.34b is equivalent to the AND-OR-INVERT circuit of Fig

Now New DFA is,

$M = (k, \Sigma, f, s, \delta)$

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

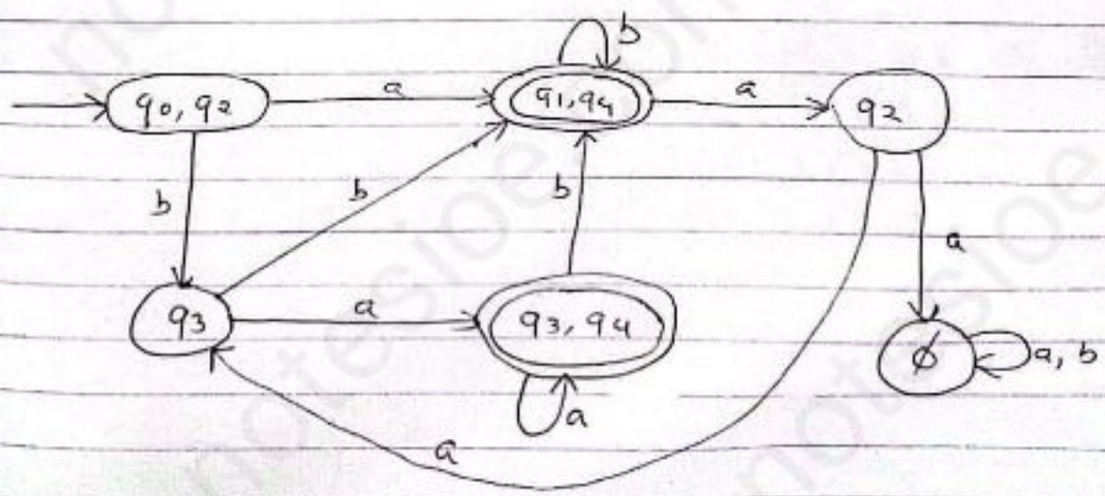
$\Sigma = \{a, b\}$

$s = \{q_0, q_2\}$

$f = \{\{q_1, q_4\}, \{q_3, q_4\}\}$

δ is,

$\delta \backslash \Sigma$	a	b
→ q_0, q_2	$\{q_1, q_4\}$	q_3
* q_1, q_4	q_2	q_1, q_4
q_3	q_3, q_4	q_1, q_4
q_2	ϕ	q_3
* q_3, q_4	q_3, q_4	q_1, q_4
ϕ	ϕ	ϕ

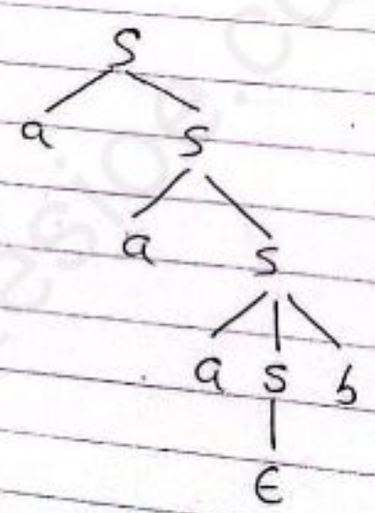


(5) Construct a CFG language $L(G) = \{a^m b^n ; m, n > 0, m \geq n\}$
Use grammar to generate string "aaab" Also draw parse tree.

$G = (V, \Sigma, k, S)$
$V = \{S\}$
$\Sigma = \{a, b\}$
~~k~~
~~$k = \{ S \rightarrow aSS / a / b$~~
$k = \{ S \rightarrow aS$
$S \rightarrow aSb$
$S \rightarrow \epsilon$
$\}$

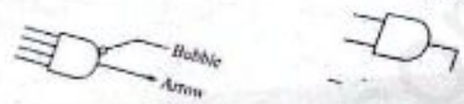
For aaab

$S \rightarrow aS$
$\rightarrow aaS$
$\rightarrow aaSb$
$\rightarrow aaab$



as in Fig. 2.34a. Connect bubble to bubble and arrow to arrow.

Visualize the outputs of Fig. 2.34a connected to the arrow and bubble in
shows the logic circuit. This means that the expander outputs are being ORe
OR-INVERT gate. In other words, Fig. 2.34b is equivalent to the AND-OR-IN



$= \{a^m b^n : m, n \geq 0, m \geq n\}$
string "aaab" Also

b

(i) Convert following CFG to CNF

$G = (V, \Sigma, R, S)$
$V = \{S, A, B, a, b\}$
$\Sigma = \{a, b\}$
$R = \{ S \to aAb / Ba / A$
$A \to SS / e$
$B \to e$
$\}$

Simplifying R,
removing unit production of A in S

$S \to aAb / Ba / SS / e$
$A \to SS / e$
$B \to e$

Removing null production of A

$S \to aAb / a\cancel{B} / \cancel{B}a / SS$
$\cancel{A \to SS / e}$
$\cancel{B}$
$S \to aAb / ab / Ba / SS / e$
$A \to SS$
$B \to e$

Removing null production of B

$S \to aAb \mid ab \mid a \mid SS \mid e$
$A \to SS$

~~Removing null production of S~~

Now, converting to CNF

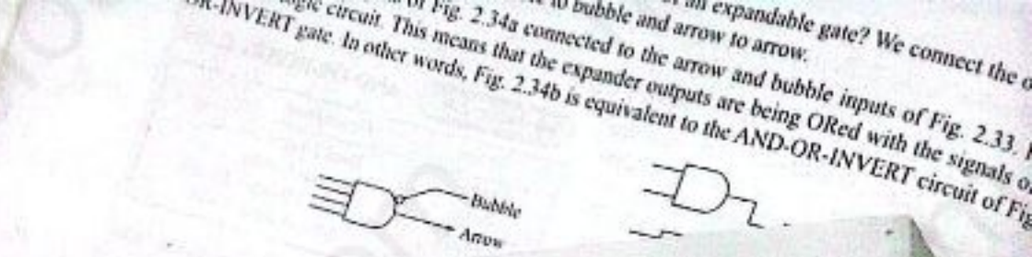
$S \to aAb \Rightarrow S \to Pb$ - $\quad S \to ab$
$\Rightarrow \boxed{S \to PT_b} \quad \Rightarrow \boxed{S \to T_aT_b}$

$P \to aA \Rightarrow \boxed{P \to T_aA}$

$\boxed{T_b \to b} \quad \boxed{T_a \to a}$

Now New CNF is,

$G = (V, \Sigma, R, S)$
$V = \{S, A, P, T_a, T_b\}$
$\Sigma = \{a, b\}$
$R = \{ S \to PT_b / T_aT_b / a / SS / e$
$A \to SS$
$P \to T_aA$
$T_a \to a$
$T_b \to b$
$\}$



② Define ambiguity & inherent ambiguity in parse tree. For a CFG ~~ge~~ defined by

$G = (V, \Sigma, R, S)$

$V = \{S\}$

$\Sigma = \{a\}$

$R = \{S \to SS$

$S \to a$

$\}$

Obtain Language $L(G)$

Ambiguity

For some CFG's, it is possible to find a terminal string with more than one parse tree or equivalently more than one leftmost or right most derivations. Such process is called ambiguity.

Inherent Ambiguity.

A ~~CF~~ context free language L is said to be inherent Ambiguity if all of its grammar are ambigous.

$G = (V,$

$G = (V, \Sigma, R, S)$

~~$V = \{S, a, \}$~~

$V = \{S, a\}$

$\Sigma = \{a\}$

$R = \{S \to SS$

$S \to a$

$\}$

$S \Rightarrow SS$

$\Rightarrow aS$

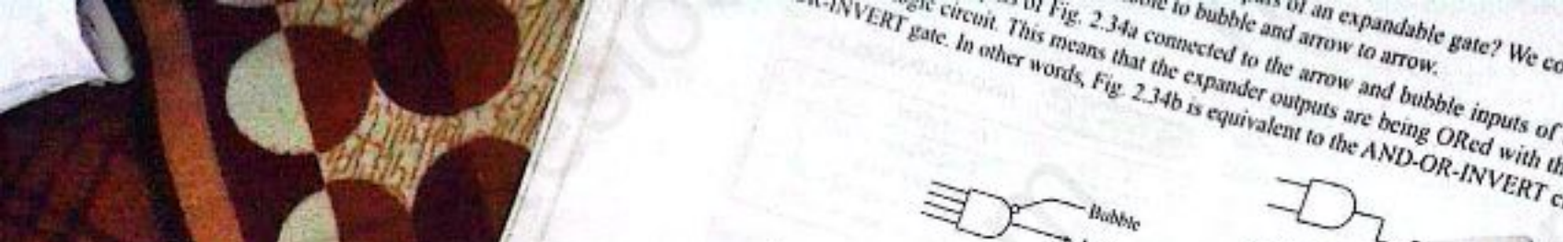
$\Rightarrow aSS$

$\Rightarrow aaS$

$\Rightarrow a^n$

$\therefore$ Language of given grammar is

$L(G) = \{a^n : n \geq 0\}$.



(3) Design a PDA that accepts language $L = \{a^n b^{3n} : n \geq 1\}$. Test your design for string aabbb.

$M = (V, \Sigma, \Gamma, \Delta, S, F)$

$V = \{\}$

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

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

$S = \{P\}$

$F = \{f\}$

$\Delta = \{$
$(P, a, \epsilon) \rightarrow (P, a)$ — (1)
$(P, a, a) \rightarrow (P, a)$ (2)
$(P, b, a) \rightarrow (Q, b)$ (3)
$(Q, b, b) \rightarrow (R, \epsilon)$ (4)
$(R, b, a) \rightarrow (P, \epsilon)$ (5)
$(P, \epsilon, \epsilon) \rightarrow (f, \epsilon)$ (6)

State	Input	Stack	Transitions used
P	a bbb	–	–
P	a bbb	a	(1)
Q	a b bb	ba	(3)
R	a b bb	a	(4)
P	a b b b –	–	(5)
f	a b b b –	–	(6)

Accepted. //

(9) What is the difference between CFG & unrestricted grammar with example. Design a Turing Machine that reads binary string & doubles the number represented by that string. A binary number is doubled if a 0 is added on right end ~~in~~ of the number.

CFG

A CFG is a ~~grammar~~ quadrupole which is defined as,

$G = (V, \Sigma, R, S)$, where,

V is an alphabet

$\Sigma \subseteq V$ set of terminal symbols,

$S \in V - \Sigma$ is start symbol

R is set of rules, subset of $((V-\Sigma) \times V^*)$

eg. For language $L = \{a^n, b^n, n \geq 0\}$

$G = (V, \Sigma, R, S)$

$V = \{S, a, b\}$

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

$R = \{S \rightarrow aSb / \epsilon\}$

Unrestricted Grammar

An unrestricted Grammar is quadrupole $G(V, \Sigma, R, S)$ where,

V is an alphabet

$\Sigma \subseteq V$ is set of terminal symbols

$S \in (V - \Sigma)$ is start symbol.

R is set of rules, finite subset of $(V^*(V-\Sigma)V^* \times V^*)$

eg. For language $L = \{a^n b^n c^n, n \geq 0\}$

$G = (V, \Sigma, R, S)$

$V = \{$

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

$R = \{ S \rightarrow aABCS$

$S \rightarrow T_c$

$CA \rightarrow AC$

$BA \rightarrow AB$

$CB \rightarrow BC$

$CT_c \rightarrow T_c c$

$BT_c \rightarrow T_b b$

$BT_b \rightarrow T_b b$

$AT_b \rightarrow T_a a$

$AT_a \rightarrow T_a a$

$T_a \rightarrow e$

$\}$

$M = (k, \Sigma, \delta, s, H)$

$\Rightarrow R_u$

$k = \{q_0, h\}$

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

$s = q_0$

$H = \{h\}$

q	σ	$\delta(q, \sigma)$
q_0	0	(q_0, L)
q_0	1	(q_0, L)
q_0	$\sqcup$	$(h, 0)$

For $\sqcup 101 \sqcup$

$> R_u \xrightarrow{\sqcup} 0$

$\vdash (q_0 \sqcup 101 \sqcup)$

$\vdash (q_0, \sqcup \underline{1} 01 \sqcup)$

$\vdash (q_0, \sqcup 1 \underline{0} 1 \sqcup)$

$\vdash (q_0, \sqcup 10 \underline{1} \sqcup)$

$\vdash (q_0, \sqcup 101 \underline{\sqcup})$

$\vdash (h q_0, \sqcup 101 \underline{0} \sqcup)$

(10) Define head shifting & symbol writing Turing Machines. Design TM for $f(w) = ww^R$, $w \in \{0,1\}^+$
- If your input is #01#, TM should give #0110#

Head shifting Turing Machine.

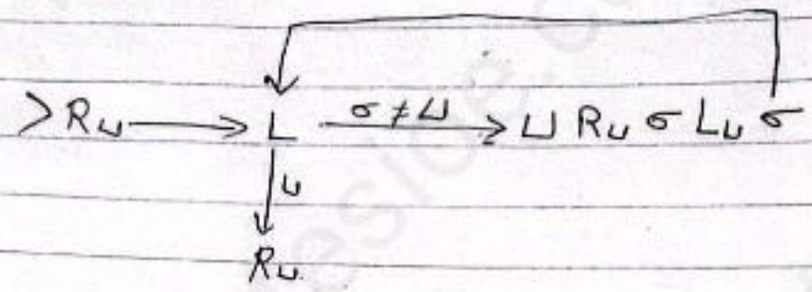
The Turing machine which ~~only~~ shift its head when it gets any input symbol. It moves to right or to left of its current head. It is denoted by $M_{\rightarrow}$ or $M_{\leftarrow}$

Symbol writing Turing Machine
The Turing machine which is used for writing the symbols in various positions is symbol writing Turing Machine.

It is denoted by M_a.

Now,
for $f(w) = ww^R$

$M = (\Sigma, k, \Sigma, \delta, S, H)$
$k = \{$
$\Sigma = \{0, 1, u\}$
$S = q_0$



For input #10#

$\vdash^{R_0}$ #10#
$\vdash^{R_\omega}$ #10#
$\vdash^{L}$ #10#
$\vdash^{\#}$ #1#
$\vdash^{R_\omega}$ #1##
$\vdash^{\sigma}$ #1#0#
$\vdash^{L_\omega}$ #1#0#
$\vdash^{\sigma}$ #100#

$\vdash$ #01#
$\vdash^{R_\omega}$ #01#
$\vdash^{L}$ #01#
$\vdash^{\#}$ #0##
$\vdash^{R_\omega}$ #0##
$\vdash^{\sigma}$ #0#1#
$\vdash^{L_\omega}$ #0#1#
$\vdash^{\sigma}$ #011#
$\vdash^{L}$ #011#
$\vdash^{\omega}$ ##11#
$\vdash^{R_\omega}$ ##11#
$\vdash^{\sigma}$ ##110#
$\vdash^{L_\omega}$ ##110#
$\vdash^{\sigma}$ #0110#
$\vdash^{L}$ #0110#
$\vdash^{R_\omega}$ #0110#

#

(1) Define class P and class NP problems with example. How do they relate to NP-complete problems?

Class P problem.
Class P is a class of problem that can be solved deterministically in polynomial time.

The class P is important because
- it is invarient over all models of computation
- practical problems in P class have efficient (low degree polynomial) algorithm.

Class NP problem
Class NP is a class of problem that can be solve non-deterministically in polynomial time.

The class NP is also invarient over all reasonal models of computation.

Clearly $P \subseteq NP$. ~~but not~~

(12) What is an "Algorithm" according to Church-Turing Thesis? Why is it called "thesis" & not a theorem? Prove that if a languge L & its complement $\bar{L}$ bothare recursively enumerable then L is recursive.

According to Church-Turing Thesis, an "Algorithm" is any computational procedure that can only be represented by a turing machine.

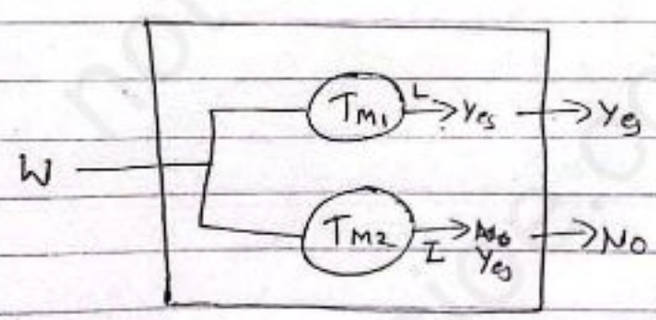
i.e. No computational procedure is considered as an algorithm unless it is represented by a Turing machine.

Church "thesis" can not be a theorem because ideas such as "can ever be defined by homans" & algorithm that people can taught to perform" are not part of any known mathematics".

If a language L & its complement $\bar{L}$ both are recursively enumerable the L is recursive.

Proof:
Let T_{m1} & T_{m2} accepts L & $\bar{L}$ (which are recursively enumerable) respectively.
Let us construct a Turing machine T_m which simulates T_{m1} & T_{m2} simultaneously.
T_m accepts w if T_{m1} accepts & reject w if T_{m2} will accept.
~~Thus~~
Thus, T_m will say 'Yes' or 'No' but not both.

Since T_m accepts L. So L is recursive.



If L & $\bar{L}$ are recursively enumerable
L is recursive.

73 chaitra.

1) Explain dovetailing technique. Write a regular expression for even number of 'a' followed by odd number of 'b' over an alphabet $\Sigma = \{a, b\}^*$.

Dovetailing technique.
It can be defined as technique of interweaving the enumeration of several sets.

We can show that union of finite number of countably infinite set is countably infinite.

Let A, B & C be 3 countably infinite set,
$A = \{a_0, a_1, a_2, \ldots\}$
$B = \{b_0, b_1, b_2, \ldots\}$
$C = \{c_0, c_1, c_2, \ldots\}$

$A \cup B \cup C = \{a_0, b_0, c_0, a_1, b_1, c_1, \ldots\}$.

i.e. we see first element of first set, ~~second~~ first element of second set, ~~third~~ first element of third set and so on.

Regular expression for even number of 'a' followed by odd number of 'b'

$RE = (aa)^* b (bb)^*$

② Design a DFA that accepts the language $L = \{x : x \in \{0,1\}^*\ 10110$ does not occur as a substring$\}$. Verify your design with supporting example

The DFA is defined as,

$M = (V, \Sigma, \Delta, f, s)$

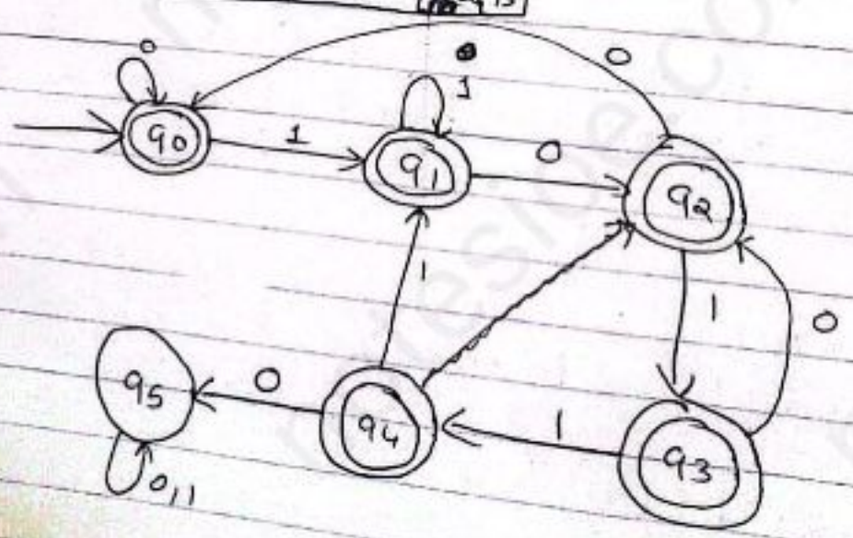
$V = \{$

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

$f = \{$

$s = \{q_0\}$

$\Delta \backslash \Sigma$	0	1
* q_0	q_0	q_1
q_1	q_2	q_1
q_2	q_0	q_3
q_3	q_2	q_4
q_4	q_5	q_1
* q_5	q_5	q_5



~~$\vdash q_0, 10$~~

For 101110110

$\vdash q_0, \underline{1}01110110$
$\vdash q_1, 1\underline{0}1110110$
$\vdash q_2, 10\underline{1}110110$
$\vdash q_3, 101\underline{1}10110$
$\vdash q_4, 1011\underline{1}0110$
$\vdash q_1, 10111\underline{0}110$
$\vdash q_2, 101110\underline{1}10$
$\vdash q_3, 1011101\underline{1}0$
$\vdash q_4, 10111011\underline{0}$
$\vdash q_5, 101110110 -$

Not accepted //

For 110

$\vdash q_0, \underline{1}10$
$\vdash q_1, 1\underline{1}0$
$\vdash q_1, 11\underline{0}$
$\vdash q_2, 110 -$

Accepted //

(5)

(B3) State Pumping Lemma for Regular Language. Using Pumping Lemma & prove that the language $L = \{ww \in \{0,1\}^* \text{ \& } w \text{ has an equal number of 0's \& 1's}\}$ is not regular.

Pumping Lemma.

Let L be a regular language
Let w be any string, $w \in L$
& n be any pumping constant $n > 1$ & $|w| > n$
then w can be rewritten as
$w = xyz$ such that $|xz| \leq n$
$|y| \geq 1$
$|xy^iz \in L$ for $i \geq 0$.

Here, $L = \{w : w \in \{0,1\}^* \text{ \& } w \text{ has equal no of 0's \& 1's}\}$
Let no. of 0's = no. of 1's = n
$w = xyz$
Such that $x = 0^{n-k} 1^{n}$
$y = 0^{k}$
$z = 1^{n-1}$

$xy^2z = xyyz = 0^n 1^n 0^k = 0^{n+k} 1^n$, $k \neq 0$

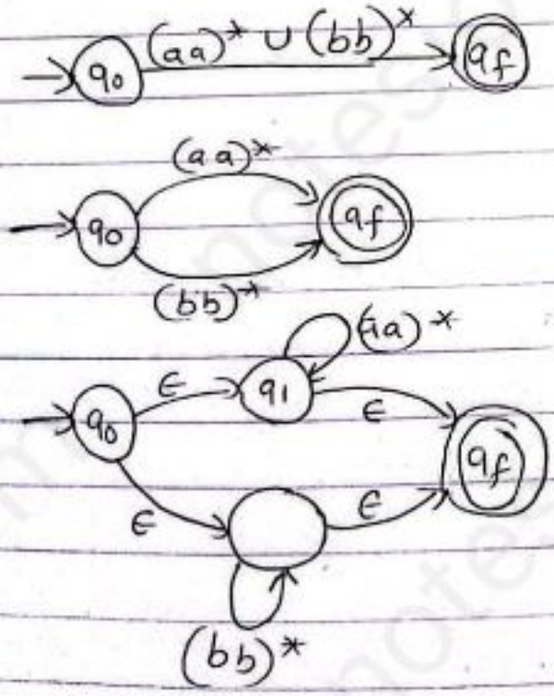
i.e. no. of 0's $\neq$ no. of 1's
So, $xy^2z \notin L$
So, ~~[illegible]~~
So, L is not regular.

(4) Why NDFA is important although it is equivalent to a DFA? Design NDFA which accepts $L = \{w : w \in \{a,b\}^*$ such that w contains either two consecutive a's or two consecutive b's?

NDFA is important although it is equivalent to a DFA because:-

→ NDFA has ability to change states in such a way that is only partially determined by the current state and input symbol. & can be much more convinienty designed.

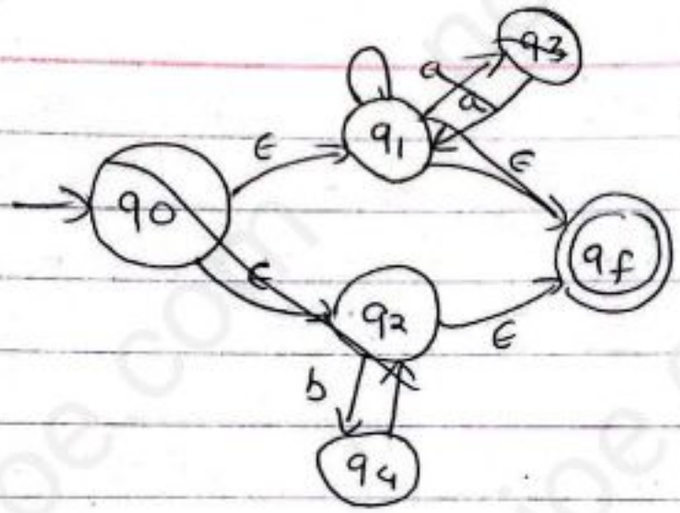
$L = (aa)^* \cup (bb)^*$

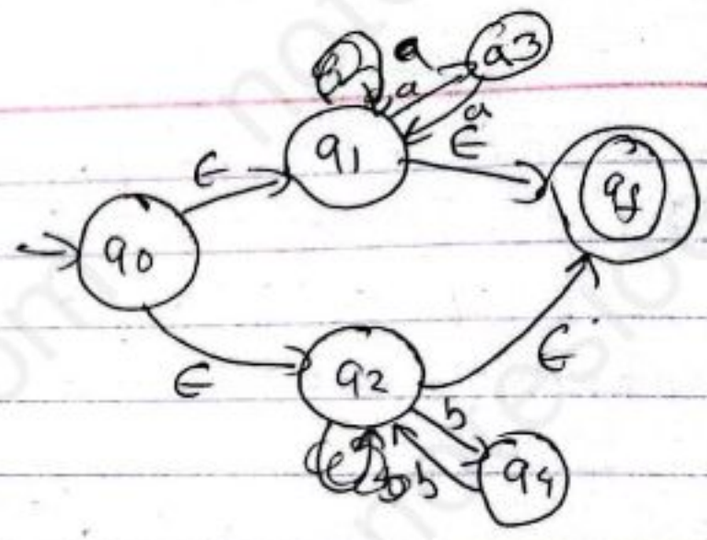


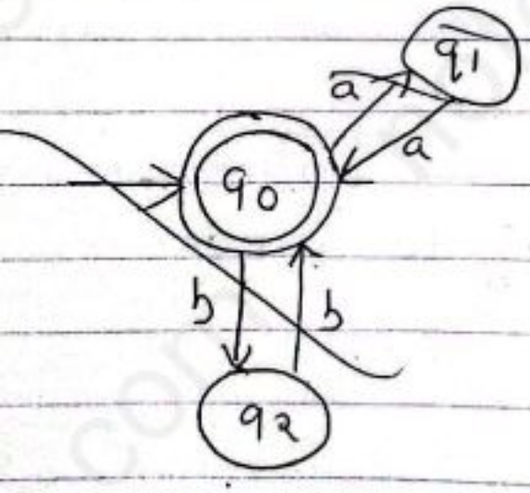
for t > 0 in the circuit of [8]

10Ω

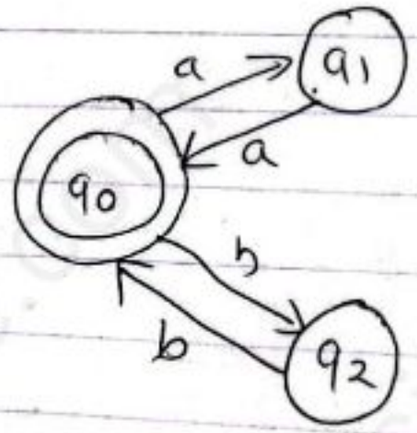
f current for t > 0, also find
m method.







⇒



//

) Define Configuration of DFA. Design DFA for
L = {w ∈ {0, 1} : w has both 01 and 10 as
substrings} Verif

ge.
language
ber of 0's & 1's}

|w|>n

i ≥ 0.

0's & 1's}

, k ≠ 0

(5) Write CFG for $L = \{w \in \{a,b\}^* : w \text{ is a palindrome}\}$ & also draw parse tree for the derivations of any two string of length even and odd.

$G = (V, \Sigma, R, S)$

$V = \{S, a, b\}$

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

$R = \{ S \to aSa$

$S \to bSb$

$S \to a$

$S \to b$

$S \to e$

$\}$

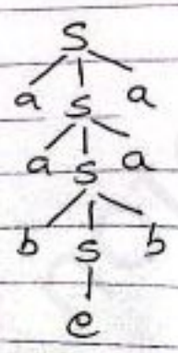
For, aabbaa

$S \to aSa$

$\to aaSaa$

$\to aabSbaa$

$\to aabbaa$

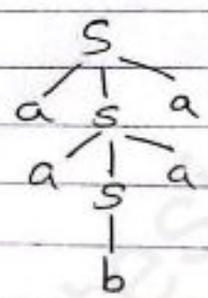


For aabaa

$S \to aSa$

$\to aaSaa$

$\to aabaa$



Removing null production in A

$S \to A/B/e$

$A \to aBa$

$B \to aAb/ab/e$

Removing null production in B

$S \to A/B/e$

$A \to aBa/aa$

$B \to aAb/ab$

Replacing unit production of A in S

$S \to aBa/aa/B/e$

$A \to aBa/aa$

$B \to aAb/ab$

Replacing unit production of B in S

$S \to aBa/aa/aAb/ab/e$

$A \to aBa/aa$

$B \to aAb/ab$

Now R is simplified CFG

To convert to CNF, it should be of form

$N \to NN$

$N \to T$

$S \to aBa \Rightarrow S \to Pa$

$\Rightarrow \boxed{S \to PT_a}$

$P \to aB \Rightarrow \boxed{P \to T_aB}$

$\boxed{T_a \to a}$

$S \to aa \Rightarrow \boxed{S \to T_aT_a}$

$S \to aAb \Rightarrow \boxed{S \to QT_b}$

$Q \to aA \Rightarrow \boxed{Q \to T_aA}$

$\boxed{T_b \to b}$

$S \to ab \Rightarrow \boxed{S \to T_aT_b}$

$\boxed{S \to e}$

$A \to aBa \Rightarrow \boxed{A \to PT_a}$

$A \to aa \Rightarrow \boxed{A \to T_aT_a}$

$B \to aAb \Rightarrow \boxed{B \to QT_b}$

$B \to ab \Rightarrow \boxed{B \to T_aT_b}$

New CNF is,

$G = (V, \Sigma, R, S)$

$V = \{S, A, B, P, Q, T_a, T_b, a, b\}$

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

$R = \{ S \to PT_a / T_aT_b / QT_b / T_aT_a$

$A \to PT_a / T_aT_a$

$B \to QT_b / T_aT_b$

$P \to T_a B$

$Q \to \cancel{T_b B} T_a A$

$T_a \to a$

$T_b \to b$

$\}$

② Design a PDA ~~which with a~~ that accepts those strings having total number of 'a' equal to the sum of no of 'b' & 'c' with sequence of a, b, c (ie $a^i b^j c^k$; $i = j+k$) Hence test for "aaaabbcc"

Let PDA be defined as,

$M = (Q, \Sigma, \Gamma, \Delta, s, f)$

$Q = \{P, f\}$

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

$\Gamma = \{a\}$

$s = \{P\}$

$f = \{f\}$

$\Delta = \{ (P, a, \epsilon) \longrightarrow (P, a)$ ①
$(P, a, a) \longrightarrow (P, a)$ ②
$(P, b, a) \longrightarrow (P, \epsilon)$ ③
$(P, c, a) \longrightarrow (P, \epsilon)$ ④
$(P, \epsilon, \epsilon) \longrightarrow (f, \epsilon)$ ⑤
$\}$

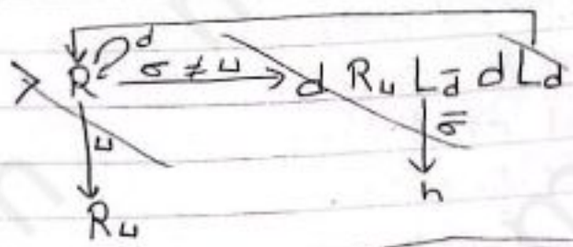
For string aaaabbcc

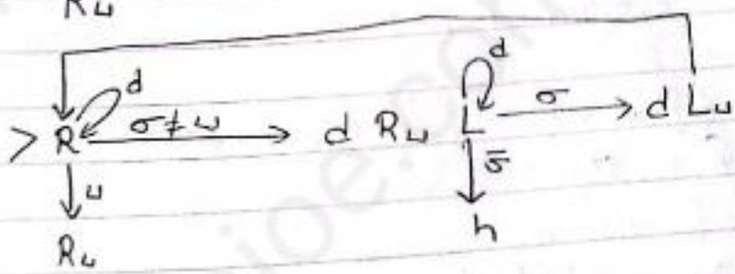
State	Input	Stack	Transition used
P	aaaabbcc	–	–
P	aaaabbcc	a	①
P	aaaabbcc	aa	②
P	aaaabbcc	aaa	②
P	aaaabbcc	aaaa	②
P	aaaabbcc	aaa	③
P	aaaabbcc	aa	③
P	aaaabbcc	a	④
f	aaaabbcc	–	④

Hence accepted. //

⑤

Design a Turing machine to accept language $L = \{ WW^R \mid W \in \{0,1\}^* \}$. Show processing for string 101101





$\vdash$ ⊔101101⊔
$\vdash^{R}$ ⊔101101⊔
$\vdash^{d}$ ⊔d01101⊔
$\vdash^{R⊔}$ ⊔d01101⊔
$\vdash^{d}$ ⊔d0110d⊔
$\vdash^{L⊔}$ ⊔d0110d⊔
$\vdash^{R}$ ⊔d0110d⊔
$\vdash^{d}$ ⊔dd110d⊔
$\vdash^{R⊔}$ ⊔dd110d⊔
$\vdash^{L}$ ⊔dd110d⊔
$\vdash^{d}$ ⊔dd11dd⊔
$\vdash^{L⊔}$ ⊔dd11dd⊔
for

$\vdash^{R}$ ⊔ddd11dd⊔
$\vdash^{d}$ ⊔dddd1dd⊔
$\vdash^{R⊔}$ ⊔dddd1dd⊔
$\vdash^{L}$ ⊔dddd1dd⊔
$\vdash^{d}$ ⊔ddddddd⊔
$\vdash^{R⊔}$ ⊔dddddddd⊔
$\vdash^{R}$ ⊔ddddddd⊔
for

//

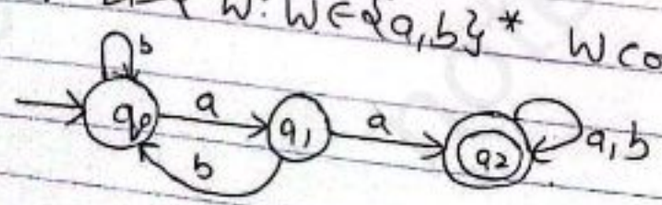
(8) Compare Turing Machine with Finite Automata (FA) and Pushdown automata.

Turing Machine is defined as the quadrupole
$M = (k, \Sigma, \delta, s, H)$
k = set of states
Σ = set of symbols
s = start ~~symbol~~ state
h = set of halting state
δ = Transition functions.

A Finite Automata is defined as the quadrupole
$M = (k, \Sigma, \delta, s, f)$
k = set of states
Σ = set of alphabets
s = start state.
f = set of final state
δ = Transition function

All the operations that can done with FA can also be done with TM.

For example
For $L = \{ w : w \in \{a,b\}^*$ w contains two consecutive's,



te Automata(FA)

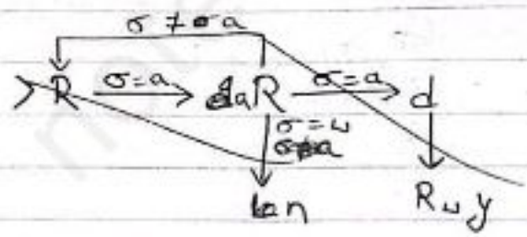
uadrupole

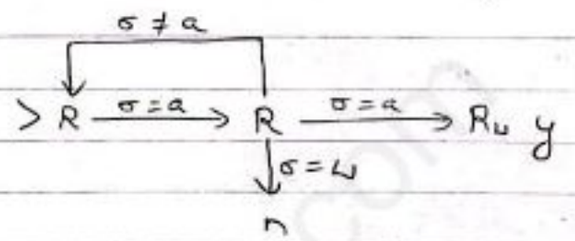
quadrupole

with ~~fin~~ FA

onsecutive's

From Turing machine,





So, Language L_1 can be computed both with Turing machine as well as FA.

For Pushdown Automata,

$M = (k, \Sigma, \Gamma, \delta, s, f)$

k = set of symbols

s = start state

Γ = stack

Σ = Alphabets

f = final state

δ = Transition functions.

for L_1, Can be expressed in PDA as,

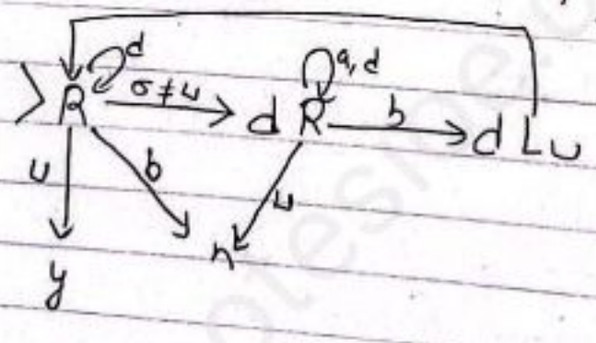
$$\delta = \{ (P, a, \epsilon) \rightarrow (QP, a)$$
$$(P, a, a) \rightarrow (Q, a)$$
$$(P, b, \epsilon) \rightarrow (P, \epsilon)$$
$$(QP, b, a) \rightarrow (P, \epsilon)$$

For $L_2 = \{ w = a^n b^n : w \in \{a, b\}^* \}$.

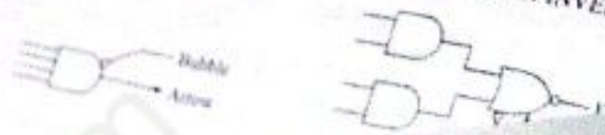
PDA is,

$$\delta = \{ (P, a, \epsilon) \rightarrow (P, a)$$
$$(P, a, a) \rightarrow (P, a)$$
$$(P, b, a) \rightarrow (P, \epsilon)$$
$$(P, \epsilon, \epsilon) \rightarrow (f, \epsilon)$$
$$\}$$

And with Turing Machine for L_2.



Hence L_2, which is solvable with PDA is also solvable with Turing machine.



(9) Explain unrestricted grammar with suitable ~~explai~~ example. Is unrestricted grammar is superset of Context Free Grammar? Justify.

Unrestricted grammar.

An unrestricted grammar is a quadruple $G(V, \Sigma, R, S)$ where

V is an alphabet

$\Sigma \subseteq V$ is set of terminal symbols

$V - \Sigma$ is set of non terminal symbols.

$S \in V - \Sigma$ is the start symbol.

R is set of rules, is finite ~~se~~ subset of

$$((V^*(V-\Sigma)V^*) \times V^*)$$

eg.

The following grammar generates the language $L = \{a^n b^n c^n : n \geq 1\}$.

$G = (V, \Sigma, R, S)$

$V = \{S, a, b, c, A, B, C, T_a, T_b, T_c\}$

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

$R = \{ S \to ABCS$

$S \to T_c$

$CA \to AC$

$BA \to AB$

$CB \to BC$

$CT_c \to T_c c$.

$BT_c \to T_b b$

$BT_b \to T_b b$.

$AT_{ab} \to T_a a$

$AT_a \to T_a a$

$T_a \to e$

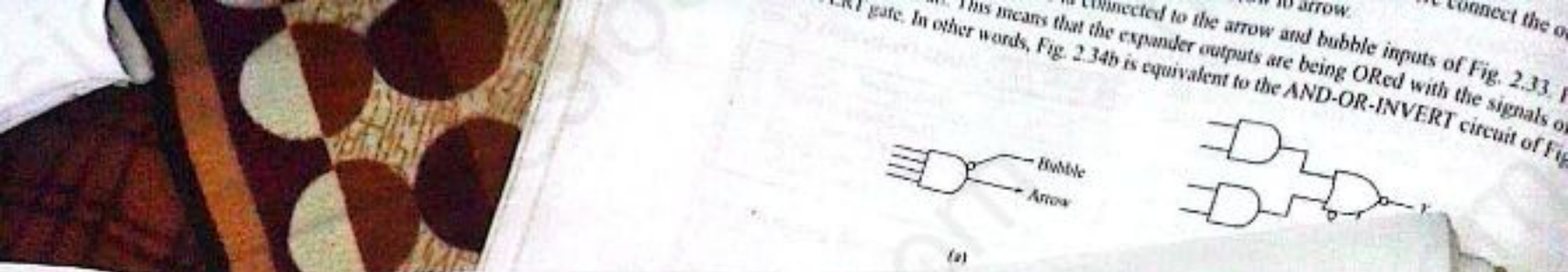
$\}$

For string abc

$S \to ABCS$
$\to ABCT_c$
$\to ABT_c c$
$\to AT_b bc$
$\to T_a abc$
$\to abc$ //

For context free Language is defined as, a quadrupole
$G = (V, \Sigma, R, S)$
V = set of alphabets
$\Sigma \subseteq V$ is the set of terminals
R is the set of rules, subset of $(V-\Sigma) \times V^*$
S is a start symbol, $S \in (V-\Sigma)$

From definitions of CFG & unrestricted grammar
We come to conclude that the set of rules of all the CFL may contain in unrestricted grammar
as, for CFL
$R = (V-\Sigma) \times V^*$
& for unrestricted grammar
$R = (V^*(V-\Sigma)V^*) \times V^*$



But all the transitions of unrestricted may not be the transitions of CFG

So unrestricted grammar may be called as superset of Context Free Grammar.

(11) What is recursive language? Mention its properties. Prove that "A language is ~~unrestricted if~~ recursive if and only if both it & its complement are recursively enumerable.

Recursive Language

A language is recursive if there exist a Turing Machine that accepts every string of the language and rejects every string that are not in the language

Properties

- Recursive Language always haults the Turing machine.
- Recursive language ~~is the~~ exactly says whether the string is to be accepted or rejected.

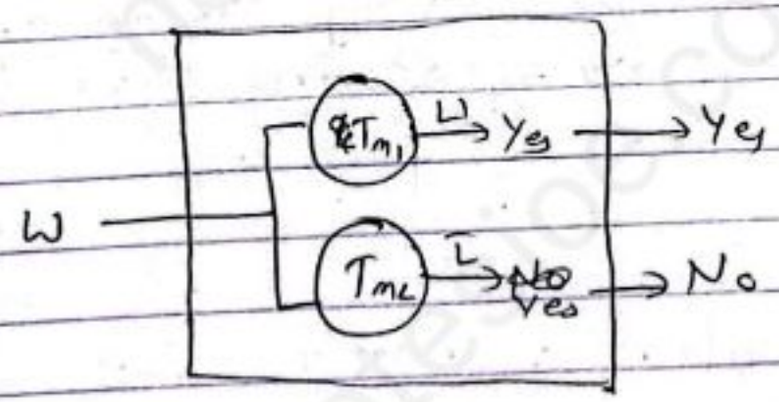
A language is recursive if and only if both it & its complement are recursively enumerable.

Let T_{m_1} & T_{m_2} accepts L & $\bar{L}$ which are recursively enumerable) respectively.

Let us construct a Turing machine T_m which simulates T_{m_1} & T_{m_2} simultaneously.

T_m accepts w if T_{m_1} accepts & rejects w if T_{m_2} accepts.

Thus T_m will always say either "yes" or "no" but never both. Since T_m is an algorithm that accepts L, it follows that L is recursive.



y if both
enumerable

ecursively

ich simulates

if Tm2

'no' but
that

(12) Define Computational Complexity & Polynomial Time Reduction. Also explain NP hard & NP Complete problems.

Computational Complexity

Computational Complexity theory is a part of theory of computation dealing with the resources required during computation to solve the given problem.

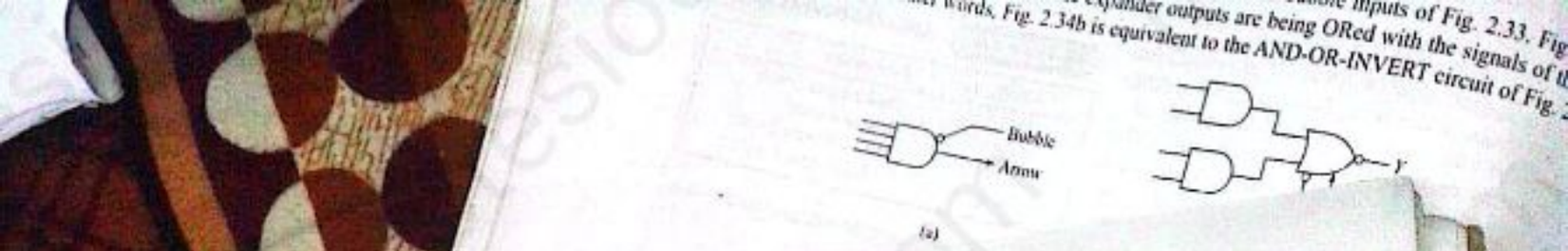
The most common resources are

(a) time complexity

(b) space complexity.

Polynomial Time Reduction

A problem P_1 is polynomially reducible to problem P_2 if there exists a polynomial-time algorithm that transforms every instance I_1 of P_1 to an instance I_2 of P_2 such that the answer to I_1 is "yes" ($I_1 \in P_1$) & if & only if answer to I_2 is "yes" ($I_2 \in P_2$).



Q0 Chaitra.

Q) Justify "The complement of diagonal set is different from each row ~~se~~ set." with the help of diagonalization principle. Show that if $3n+2$ is odd then n is odd by using proof by contradiction technique.

Let $A = \{1,2,3,4,5\}$

R be any arbitary relation defined on A

$R = \{(1,1), (1,4), (2,3), \cancel{(3,2)}, (2,4), (2,5), (3,1), (3,3), (4,2), (4,5), (5,5)\}$

ie. In Table form,

	1	2	3	4	5
1	X			X	
2			X	X	X
3	X		X		
4		X			X
5					X

Sequence along diagonal are

X		X		X

Its complement is

	X		X	

$\therefore \overline{D} = \{2,4\}$

Row sets are $R_1 = \{1,4\}$
$R_2 = \{3,4,5\}$
$R_3 = \{1,3\}$
$R_4 = \{2,5\}$
$R_5 = \{5\}$

This shows that ~~diagonal~~ complement of diagonal set is different from each row set by the diagonalization principle.

To prove,
$3n+2$ is odd the n is odd.

Let $3n+2$ is odd so for any integer k
$3n+2 = 2k+1$ — (1)

Now, Let us assume that $3n+2$ is odd & n is even

From ~~ef~~ definition we can write
$n = 2k$

Substituting $2k$ for n in (1)

$3n+2 = 3(2k)+2$
$= 2(3k+1)$

This shows that $3n+2$ is even, which contradicts our assumption.

So if $3n+2$ is odd then n is also odd.

③ Design DFA, $L = \{x \in \{0,1\}^* : x$ has an even no of 0's & even no of 1's$\}$.

Let $M = (V, \Sigma, \Delta, S, F)$

where,

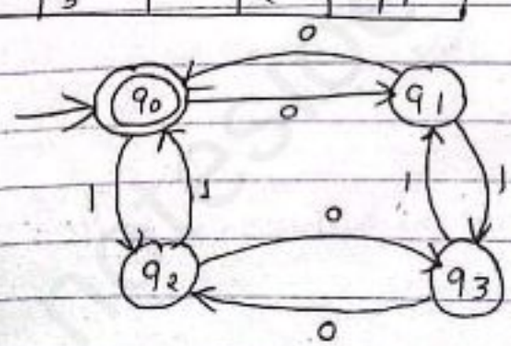
$V = \{q_0, q_1, q_2, q_3\}$

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

$s = \{q_0\}$

$f = \{q_0\}$

$\Delta \backslash \Sigma$	0	1
$\rightarrow^* q_0$	q_1	q_2
q_1	q_0	q_3
q_2	q_3	q_0
q_3	q_2	q_1



$\vdash q_0, \underline{0}011$
$\vdash q_1, 0\underline{0}11$
$\vdash q_0, 00\underline{1}1$
$\vdash q_2, 001\underline{1}$
$\vdash q_0, 0011_$
(Accepted)

$\vdash q_0, \underline{0}101$
$\vdash q_1, 0\underline{1}01$
$\vdash q_3, 01\underline{0}1$
$\vdash q_2, 010\underline{1}$
$\vdash q_0, 0101_$
(Accepted).

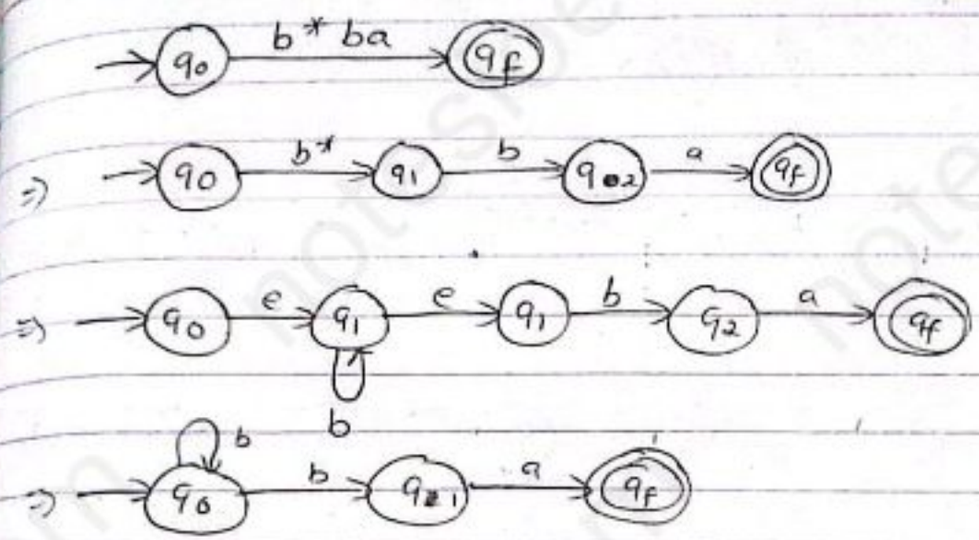
$\vdash q_0, \underline{0}0011$
$\vdash q_1, 0\underline{0}011$
$\vdash q_0, 00\underline{0}11$
$\vdash q_1, 000\underline{1}1$
$\vdash q_3, 0001\underline{1}$
$\vdash q_1, 00011_$
Not accepted

$\vdash q_0, \underline{1}101$
$\vdash q_2, 1\underline{1}01$
$\vdash q_0, 11\underline{0}1$
$\vdash q_1, 110\underline{1}$
$\vdash q_3, 1101_$
Not accepted

③ Show that for any Regular expression R, there is a NFA that accepts the same language represented by R. Construct a e-NFA for regular expression $bb(a \cup b)^* ab$.

Let us consider a RE
$RE = b^* aba$

We can construct a NFA as follows.

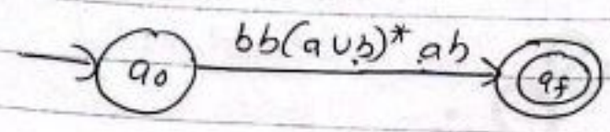


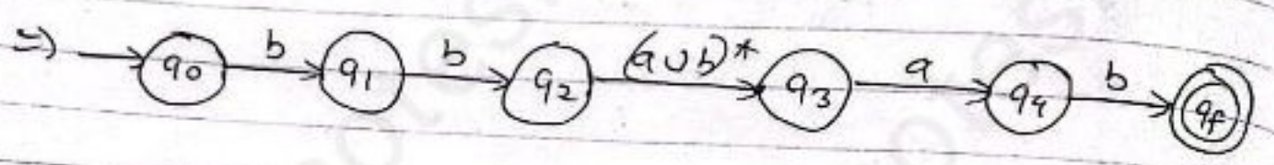
Hence is the NFA.

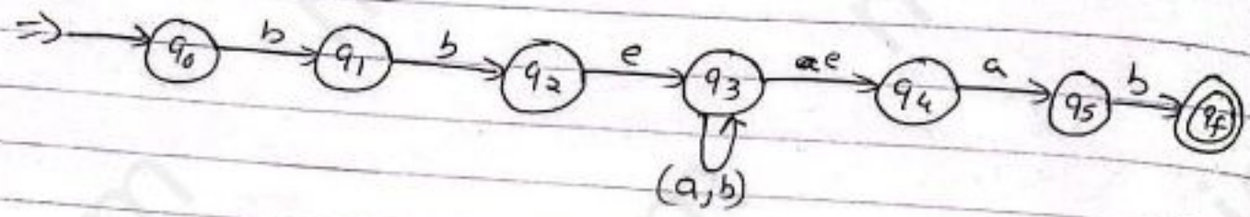
$\vdash q_0, \underline{b}bbba$
$\vdash q_0, \underline{b}\,\underline{b}bba$
$\vdash q_0, bb\underline{b}ba$
$\vdash q_0, bbb\underline{b}a$
$\vdash q_1, bbbb\underline{a}$
$\vdash q_f, bbbba$

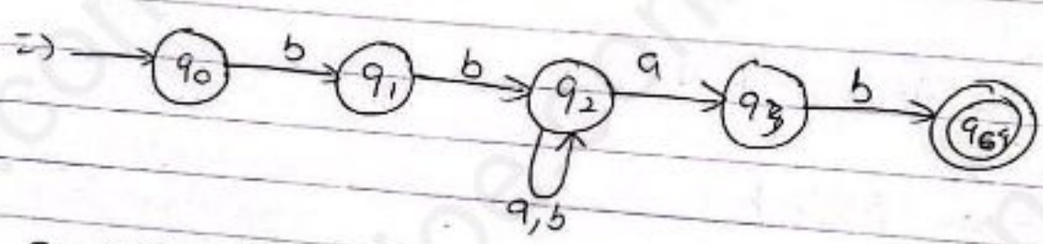
Hence also accepts the regular expression

So for all regular expressions there exists a NFA.









So ε-NFA is,

$M = (V, \Sigma, \Delta, S, F)$

$V = \{q_0, q_1, q_2, q_3, q_4\}$

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

$S = q_0$

$f = q_4$

$\Delta \Rightarrow$

$\Delta \backslash \Sigma$	a	b
→ q_0	–	q_1
q_1	–	q_1
q_2	q_2, q_3	q_2
q_3	–	q_4
* q_4	–	–

Use pumping to prove that
$L = \{ a^n b^{2n} : n \geq 1 \}$ is not regular}

Let L be regular language
w be any string $w \in L$
n be any pumping constant $n \geq 1$ & $|w| \geq n$
So w can be rewritten as,

$$|xy| \leq n$$
$$|y| \geq 0$$
$$\{xy^iz \in L \text{ for all } i \geq 0.$$

For pumping lemma we can write,

$$w = xyz = a^n b^{2n}$$
$$= \underbrace{aa\ldots aa}_{x}\underbrace{aa\ldots aa}_{y}\underbrace{bb\ldots bb}_{z}$$

$x = a^{n-k}$
$y = a^k$
$z = b^{2n}$ $\quad n \neq k,$

For $i=0$, $xyz = a^{n-k} b^{2n} \notin L$
For $i=2$, $xyyz = a^{n-k} a^k a^k b^{2n}$
$= a^{n+k} b^{2n} \notin L \because k \neq 0$

which contradicts the theorem.

So, $L = \{a^n b^{2n}$ is not regular. //

(5) Consider the regular Grammar
$G = (V, \Sigma, R, S)$ where,
$R = \{ S \to abA / B / baB / e$
$A \to bS / a$
$B \to aS$
$\}$

$S \Rightarrow abA$
$\Rightarrow abbS$
$\Rightarrow abbabA$
$\Rightarrow abbabbS$
$\Rightarrow (abb)^* S$
$\Rightarrow (abb)^* B$
$\Rightarrow (abb)^* aS$
$\Rightarrow (abb)^* aB$
$\Rightarrow (abb)^* aaS$
$\Rightarrow (abb)^* a^* S$
$\Rightarrow (abb)^* a^* baB$
$\Rightarrow (abb)^* a^* baaS$
$\Rightarrow (abb)^* a^* baabaaB$
$\Rightarrow (abb)^* a^* (baa)^* S$
$\Rightarrow (abb)^* a^* (baa)^* \cup abA$
$\Rightarrow (abb)^* a^* (baa)^* \cup aba$

Now Finite automata is,

$M = (K, \Sigma, \Delta, S, F)$

$K = \{ q_0$

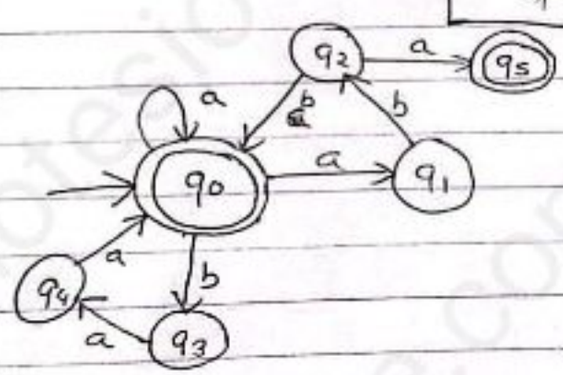
$\Sigma = \{a, b\}$

$S = q_0$

$F =$

$\Delta \Rightarrow$

$\Delta \backslash \Sigma$	a	b
$\rightarrow * q_0$	q_0, q_1	q_3
q_1	–	q_2
q_2	q_5	q_0
q_3	q_4	–
q_4	q_0	–
$* q_5$	–	–



⑥ Write context free grammars (CFG) for the language $L1 = \{a^m b^n c^n : m \geq 1, n \geq 1\}$ &
$L2 = \{a^n b^n c^m : m \geq 1, n \geq 1\}$ $L = L_1 \cup L_2$
Do Prove that the Language thus obtained is not context free by using Pumping lemma for context free.

$L_1 = \{a^n b^n c^m : n \geq 1, m \geq 1\}$
$L_2 = \{a^m b^n c^n : n \geq 1, m \geq 1\}$

Grammar for L_1 is,
$S \rightarrow AB$
$A \rightarrow aAb \mid ab$
$B \rightarrow cB \mid c$

Grammar for L_2 is,
$S \rightarrow AB$
$A \rightarrow aA \mid a$
$B \rightarrow bBC \mid bc$

$L = L_1 \cap L_2 = \{a^n b^n c^n, n \geq 1\}$

from pumping lemma,

Let L be context free language
& w be any string in L, i.e. $w \in L$
m be any pumping constant, $|w| \geq m$
Then w can be written as,

$$w = uvxyz$$

such that $|vxy| \leq m$

$|vy| \geq 0$

$uv^ixy^iz \in L$

$$w = uvxyz = \underbrace{aaa\ldots aaa}_{u}\underbrace{aaabbb\ldots bbb}_{vxy}\underbrace{ccc\ldots ccc}_{z}$$

$u = a^{n-p-q}$

$v = a^p$

$x = a^q$

$y = b^r$

$z = b^{n-r} c^n$

For $i=0$, $uxz = a^{n-p-q} \cdot a^q b^{n-r} c^n$

$= a^{n-p} b^{n-r} c^n \notin L$

For $i=2$, $uv^2xy^2z = a^{n-p-q} a^{2p} a^q b^{2r} b^{n-r} c^n$

$= \cancel{a^n b^n c^n} \cdot a^{n+p} b^{n+r} c^n \notin L$

So, $L = \{a^n b^n c^n, n \geq 1\}$ is not context free.

⑦ Convert CFG into CNF

$G = (V, \Sigma, R, S)$

$V = \{S, X, Y, Z, a, b, c\}$

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

$R = \{S \rightarrow XYZ / XY / aZ$

$X \rightarrow abX / e$

$Y \rightarrow bY / cZ / ab$

$Z \rightarrow aXZ$

$\}$

Now simplifying the rule of CFG.

Here Z is unyielding. So removing it.

$S \rightarrow XY$

$X \rightarrow ab / e$

$Y \rightarrow bY / ab$

Now removing null production of X

$S \rightarrow XY / Y$

$X \rightarrow ab$

$Y \rightarrow bY / ab$

Replacing unit production of Y in S

$S \rightarrow XY / bY / ab$

$X \rightarrow ab$

$Y \rightarrow bY / ab$

Now this is a simplified grammar.

Converting to CNF

CNF is of the form

$N \to NN$

$N \to T$

$\boxed{S \to XY}$

$S \to bY \Rightarrow \boxed{S \to T_bY}$

$\boxed{T_b \to b}$

$S \to ab \Rightarrow \boxed{S \to T_aT_b}$ $\boxed{T_a \to a}$

$X \to ab \Rightarrow \boxed{X \to T_aT_b}$

$Y \to bY \Rightarrow \boxed{Y \to T_bY}$

$Y \to ab \Rightarrow \boxed{Y \to T_aT_b}$

Now CNF is,

$G = (V, \Sigma, R, S)$

$V = \{S, X, Y, T_aT_b, a, b\}$

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

$R = \{S \to XY / T_bY / T_aT_b$

$X \to T_aT_b$

$Y \to T_bY / T_aT_b$

$T_a \to a \quad T_b \to b\}$

//

⑧ Design a PDA that accepts all palindrome defined over $\{a,b\}^*$

$M = (k, \Sigma, \Gamma, \Delta, S, F\}$

$k = \{$

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

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

$F = f$

$S = P$

$\Delta = \{$

$(P, a, \epsilon) \longrightarrow (P, a)$ ①

$(P, b, \epsilon) \longrightarrow (P, b)$ ②

$(P, a, a) \longrightarrow (P, a)$ ③

$(P, b, b) \longrightarrow (P, b)$ ④

$(P, a, b) \longrightarrow (P, a)$ ⑤

$(P, b, a) \longrightarrow (P, b)$ ⑥

$\cancel{(P, \epsilon, \epsilon) \longrightarrow (R, \epsilon)}$

$\cancel{(R, \epsilon, a) \longrightarrow (S, \epsilon)}$

$\cancel{(R, \epsilon, b) \longrightarrow (S, \epsilon)}$

$(P, a, \epsilon) \longrightarrow (R, \epsilon)$ ⑦

$(P, b, \epsilon) \longrightarrow (R, \epsilon)$ ⑧

$(R, a, a) \longrightarrow (R, \epsilon)$ ⑨

$(R, b, b) \longrightarrow (R, b)$ ⑩

$(R, \epsilon, \epsilon) \longrightarrow (f, \epsilon)$ ⑪

$\}$

For a

State	Input	Stack	Transition used
P	a	–	–
R	a –	–	7
F	a –	–	11

Accepted

For aba

State	Input	Stack	Transition used
P	aba	–	–
P	aba	a	1
R	aba	a	8
R	aba –	–	9
f	aba –	–	11

Accepted

For babab

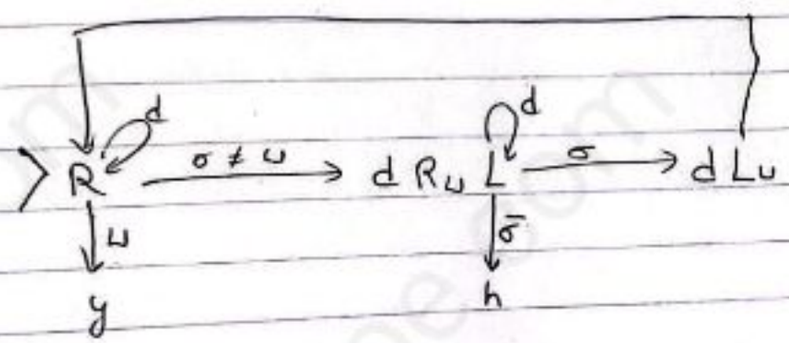
State	Input	Stack	Transition used
P	babab	–	–
P	babab	b	2
P	babab	ab	5
R	babab	ab	~~4~~ 8
R	babab	b	~~10~~ 9
R	babab –	–	10
f	babab –	–	11

(9) Define the term configuration of Turing Machine. Design a Turing machine which accepts the set of all palindromes over alphabets $\{0,1\}^*$

Configuration of Turing Machine:

A configuration of a Turing machine $M = (k, \Sigma, \delta, s, H)$

$$k \times \Delta \Sigma^* \times (\Sigma^* (\Sigma (\Sigma - \{\sqcup\} \cup \{e\})))$$



For 1001

$\vdash$ ⊔ 1001 ⊔

$\vdash^{R}$ ⊔ 1001 ⊔

$\vdash^{d}$ ⊔ d001 ⊔

$\vdash^{R\sqcup}$ ⊔ d001 ⊔

$\vdash^{L}$ ⊔ d001 ⊔

$\vdash^{d}$ ⊔ d00d ⊔

$\vdash^{L\sqcup}$ ⊔ d00d ⊔

$\vdash^{R}$ ⊔ d00d ⊔

$\vdash^{d}$ ⊔ dd0d ⊔

$\vdash^{R\sqcup}$ ⊔ dd0d ⊔

$\vdash^{L}$ ⊔ dd0d ⊔

$\vdash^{d}$ ⊔ ddd d ⊔

$\vdash^{L\sqcup}$ ⊔ dddd ⊔

$\vdash^{R\sqcup}$ ⊔ dddd ⊔

$\vdash^{y}$ ⊔ dddd y

//

74 Ashwin

(1) State diagonilization principle. Use principle of mathematical induction to prove $n^4 - 4n^2$ is divisible by 3 for $n \geq 0$

Diagonilization ~~pro~~ principle.

Let R be a binary relation on set A, & let D be the diagonal set for R be $\{a : a \in A \text{ and } (a,a) \notin R\}$
For each $a \in A$ let $R_a \{b : b \in A \text{ and } (a,b) \in R\}$
The D is ~~is~~ distinct ~~for e~~ from each R_a.

Let $R = \{(a,b), (a,d), (b,b), (b,e), (c,d), (c,e); (d,d), (e,a), (e,e)$

	a	b	c	d	e
a		x		x	
b		x			x
c				x	x
d				x	
e	x				x

D =

a	b	c	d	e
	x		x	x

$\overline{D}$ =

x		x		

Diagonal set
$D = \{a, c\}$ is distinct from each row.

strings a-z is
one term

To prove:
$n^4 - 4n^2$ is divisible by 3 for $n \geq 0$

Step1: Basis step
For $n = 1$

$$LHS = n^4 - 3n^2$$
$$= 1 - 3$$
$$= -3$$
$$= 3(-1)$$

which is divisible by 3

Step2: Induction hypothesis
Assumer for $n = k$, $k^4 - 4k^2$ is divisible by 3
i.e. $(k^4 - 4k^2) = 3N$

Step3: Induction method
For $n = k+1$
To prove $(k+1)^4 - 4(k+1)^2 = 3M$
is divisible by 3

$$LHS = (k+1)^4 - 4(k+1)^2$$
$$= (k^2+2k+1)(k^2+2k+1) - 4(k^2+2k+1)$$
$$= (k^2+2k+1)[k^2+2k-3]$$
$$= k^4 + 2k^3 - 3k^2 + 2k^3 + 4k^2 - 6k + k^2 + 2k - 3$$
$$= k^4 + 4k^3 + 2k^2 - 4k - 3$$
$$= k^4 - 4k^2 + 6k^2 + 4k^3 - 16k + 12k - 3$$
$$= (k^4 - 4k^2) + 6k^2 + \frac{4k}{k}(k^4 - 4k^2) + 3(4k-1)$$
$$= 3M + 6k^2 + 4/k \cdot 3M + 3(4k-1)$$
$$= 3(M + 2k^2 + 4M/k + (4k-1))$$

//

② What is significance of finite automata? Design a DFA that accepts the string over an alphabet $\Sigma\{0,1\}$ that either starts with 01 or end with 01. Hence test your design for any two strings

Significance of finite automata.

It is used ~~for~~ as

- Software for designing & checking behaviour of digital circuits.

- For scanning large bodies of texts such as collection of web pages

Let $M = \{k, \Sigma, R, S, F\}$

$k = \{q_0, q_1, q_2, q_3, q_4, q_5\}$

$\Sigma = \cancel{\{a,b\}}\ \{0, 1\}$

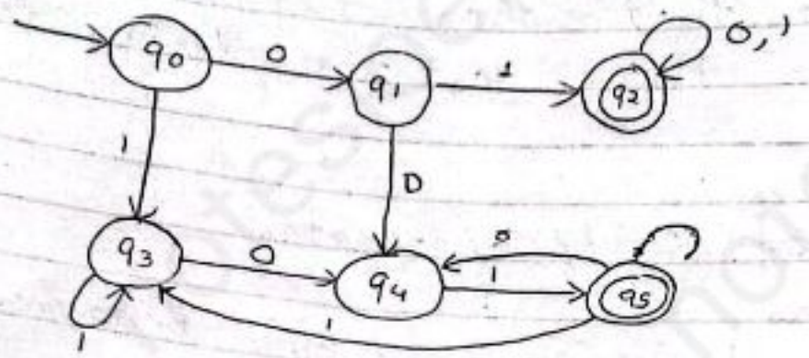
$S = \{q_0\}$

$F = \{q_2, q_5\}$

R =

$\Delta \backslash \Sigma$	0	1
$\rightarrow q_0$	q_1	q_3
q_1	q_4	q_2
$* q_2$	q_2	q_2
q_3	q_4	q_3
q_4	q_0	q_5
$* q_5$	q_4	q_3

State diagram



For 0111

⊣ q0, 0111
⊣ q1, 0111
⊣ q2, 0111
⊣ q2, 0111
⊣ q2, 0111_

Accepted.

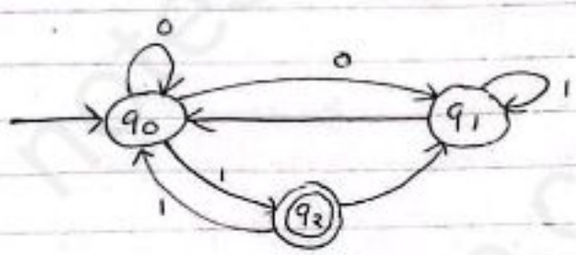
101101

⊣ q0, 101101
⊣ q3, 101101
⊣ q4, 101101
⊣ q5, 101101
⊣ q3, 101101
⊣ q4, 101101
⊣ q5, 101101_

Accepted.

③ Differentiate between DFA & NDFA. Convert the following NDFA to its DFA

DFA	NDFA
① DFA changes the state in such a way that is completely determined by the current state & input symbol.	① NDFA has ability to change states in such a way that ~~an~~ is only partially determined by the current state and input symbol.
② There is only one fixed next state for a given combination of current state & input symbol	② There may be several posibility of next state for given combination of current symbol & next input.



$M = (k, \Sigma, R, S, f)$

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

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

$S \rightarrow \{q_0\}$

$f \rightarrow \{q_2\}$

$R =$

$\delta \backslash \Sigma$	0	1
$\rightarrow q_0$	q_0, q_1	q_2
q_1	q_0	q_1
$*q_2$	-	q_0, q_1

$f(\{q_0, 0) \longrightarrow \{q_0, q_1\}$ NS

$f(q_0, 1) \longrightarrow \{q_2\}$ NS

$f(\{q_0, q_1\}, 0) \longrightarrow \{q_0, q_1\}$ ES

$f(\{q_0, q_1\}, 1) \longrightarrow \{q_2, q_1\}$ NS

$f(\{q_2\}, 0) \longrightarrow \phi$ NS

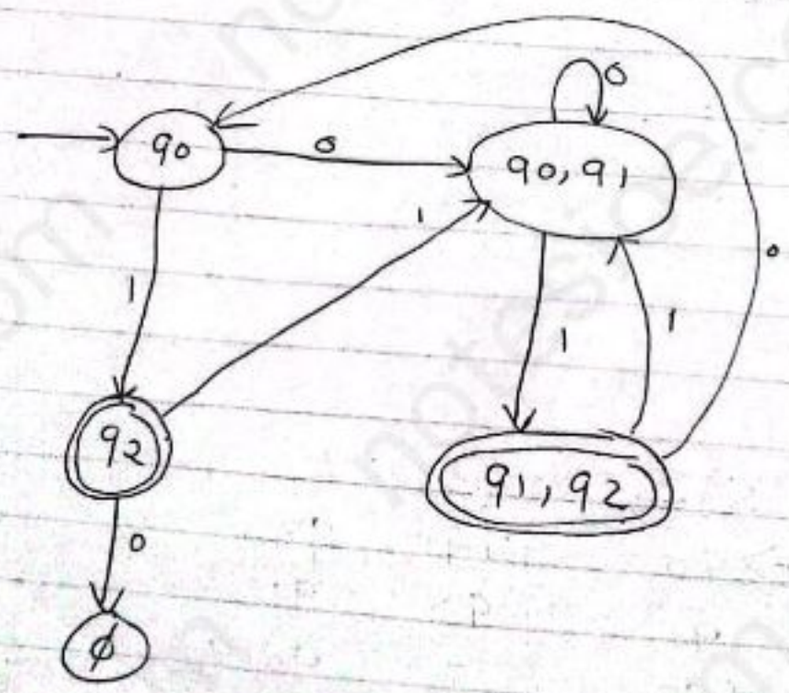
$f(\{q_2\}, 1) \longrightarrow \{q_0, q_1\}$ ES

$f(\{q_1, q_2\}, 1) \longrightarrow \{q_0, q_1\}$ ES

$f(\{q_1, q_2\}, 0) \longrightarrow \{q_0\}$ ES

$f(\phi, 0) \longrightarrow \phi$

$f(\phi, 1) \longrightarrow \phi$



Now new DFA is,

$M = (k, \Sigma, f, S, R)$

$k = \{\{q_0\}, \{q_2\}, \{q_0, q_1\}, \{q_1, q_2\}, \phi\}$

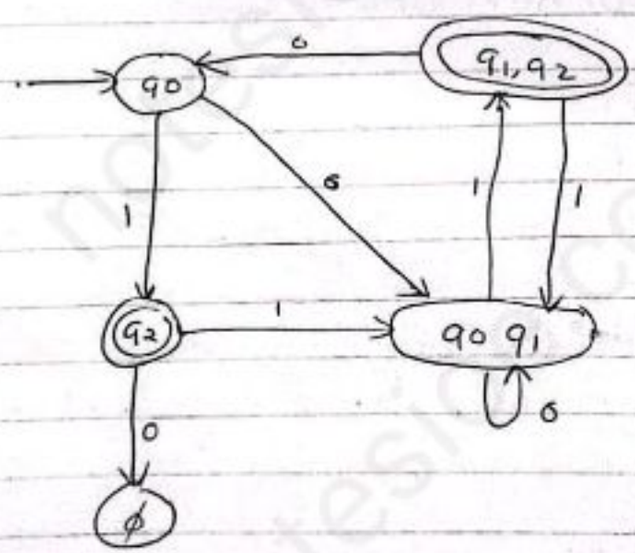
$\Sigma = \{0, 1\}$

$S = \{q_0\}$

$F = \{\{q_2\}, \{q_1, q_2\}\}$

$R =$

$R \backslash \Sigma$	0	1
q_0	q_0, q_1	q_2
q_0, q_1	q_0, q_1	$q_1 q_2$
q_1, q_2	q_0	$q_0 q_1$
q_2	ϕ	q_0, q_1
ϕ	ϕ	ϕ



④ Define Closure Properties of Regular Language. Prove that regular Language are closed under Union, Intersection and Complementation operation.

The ~~Cl~~ Closure Properties of ~~Reo~~ Regular language are :-

① Union
② Intersection
③ Complementation
④ Kleene Star
⑤ Concatenation.

Union

The language is closed under union.

Let $M_1 = (k_1, \Sigma_1, \Delta_1, S_1, F_1)$
$M_2 = (k_2, \Sigma_2, \Delta_2, S_2, F_2)$
are two NFAs

Let us consider a new NFA, M such that
$L(M) = L(M_1) \cup L(M_2)$

$M = (k, \Sigma, \Delta, S, F)$
$k = k_1 \cup k_2 \cup \{s\}$
$\Sigma = \Sigma_1 \cup \Sigma_2$
$F = F_1 \cup F_2$
$\Delta = \Delta_1 \cup \Delta_2 \cup \{\delta(s,e) \to s_1\} \cup \{\delta(s,e) \to s_2\}$

So regular language is closed under union.

Complementation

Let $M = (k, \Sigma, \delta, S, F)$ be a NFA then we can construct a new ~~DFA that~~ NFA that accepts complementary language of M

i.e. $\overline{M} = (k, \Sigma, \delta, S, k-F)$

So regular language is closed under complementation

Intersection

Let L_1 & L_2 be two regular languages accepted by NFA M_1 & M_2 then

$$L_1 \cap L_2 = \Sigma^* - ((\Sigma_1^* - L_1) \cup (\Sigma_2^* - L_2))$$

Since a language is closed under complementation & union

So, the language is also closed under intersection.

increments any binary
1/0,2 0/1,2 → (L)

(5) Define pumping lemma for context free language. Prove that $L = \{ WW | W \in \{0,1\}^* \}$ is not context free.

Pumping lemma for context free

Let L be a context free Language & w be the string in L, $w \in L$
Let n be any pumping constant with $|w| \geq n$
Then w can be written as $w = uvxyz$
such that
i) $|vxy| \leq n$
ii) $|vy| \geq 1$
iii) $uv^ixy^iz \in L$

$L = WW$, $W \in \{0,1\}^*$
Let $W = 0^m 1^n$, $m \geq 0$ $n \geq 0$

$L = WW = 0^m 1^m 0^m 1^m$

Let $u = 0^{m-p-r}$
$v = 0^p$
$x = 0^r 1^m$
$y = 0^n 1^s$
$z = 1^{m-s}$ $\quad p \neq r \neq s \neq 0$

$|vy| \geq 1$

For $i=0$

$$uxz = \cancel{0^{2n-p-r}\, 0^{n}\, 1^{m-s}}\; 0^{n-p-r}\, 0^{r}\, 1^{m}\, 0^{n}$$
$$= 0^{2n-p-r}\, 1^{m-s} \notin L$$

Since $2n-p-r \neq 2n$
$m-s \neq$

For $i=0$

$$uxz = 0^{n-p-r}\, 0^{r}\, 1^{m}\, 1^{m-s}$$
$$= 0^{n-r}\, 1^{m}\, 1^{m-s} \notin L$$

$n-r \neq n$
$m-s \neq m$
$n \neq 0$

For $i=2$

$$uv^2xy^2z = 0^{n-p-r}\, 0^{2p}\, 0^{r}\, 1^{m}\, 0^{2n}\, 1^{2s}\, 1^{m-s}$$
$$= 0^{n+p}\, 1^{m}\, 0^{2n}\, 1^{m+s} \notin L$$

$n \neq p$
$\Rightarrow n+p \neq 2n$
$s \neq 0$
$\Rightarrow m+s \neq m$

Hence $L = \{ww \mid w \in \{0,1\}^*\}$
is not context free.

(6) Convert the following CFG into CNF

$G = (V, \Sigma, R, S)$

$V = \{S, A, B, a, b\}$

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

$R = \{ S \to ASB / \epsilon$

$A \to aAS / a$

$B \to AB / b / \epsilon$

$\}$

Removing null production in S

$R \to \{ S \to ASB / AB$

$A \to aAS / a / aA$

$B \to AB / b / \epsilon$

$\}$

Removing null production in B

$R \to \{ S \to ASB / AB / AS / A$

$A \to aAS / a / aA$

$B \to AB / b / A$

Replacing unit production of A in S

$R \to \{ S \to ASB / AB / AS / aAS / a / aA$

$A \to aAS / a / aA$

$B \to AB / b / A$

$\}$

Removing null production of A in B

$R \to \{ S \to ASB / AB / AS / aAS / a / aA$

$A \to aAS / a / aA$

$B \to AB / b / aAS / a / aA$

$\}$

Now R is a simplified CFG

To convert to CNF, it should be of the form

$N \to NN$

$N \to T$

$S \to ASB \Rightarrow \boxed{S \to PB}$

$\boxed{P \to AS}$

$\boxed{S \to AB}$ $\boxed{S \to AS}$ $\boxed{S \to a}$

$S \to aAS \Rightarrow \boxed{S \to QS}$

$Q \to aA \Rightarrow \boxed{Q \to T_aA}$

$\boxed{T_a \to a}$

$S \to aA \Rightarrow \boxed{S \to T_aA}$

$A \to aAS \Rightarrow \boxed{A \to QS}$

$\boxed{A \to a}$

$A \to aA \Rightarrow \boxed{A \to T_aA}$

...ments any binary strings
...1a.L

$\boxed{B \to AB}$ $\boxed{B \to b}$

$B \to aAS \Rightarrow \boxed{B \to QS}$

$\boxed{B \to a}$

$B \to aA \Rightarrow \boxed{B \to T_a A}$

Now converted CNF is

$G = (V, \Sigma, R, S)$

$V = \{S, A, B, P, Q, T_a, T_b, a, b\}$

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

$R = \{S \to PB / AB / AS / QS / a / T_a A$

$A \to QS / a / T_a A$

$B \to AB / b / QS / a / T_a A$

$P \to AS$

$Q \to T_a A$

$T_a \to a$

$\}$

⑦ Mention the role of parse tree in context free grammar. Design a PDA that accepts $L = \{a^n b^{2n+1}, n>0\}$, check it for aabbbbb

Role of parse tree in CFG.

The parse tree shows how a string is derived from a given Context free Grammar i.e. either by Left most derivation or right most derivation.

eg.

For Garammar $G = (V, \Sigma, R, S)$

$V = \{S, (,)\}$

$\Sigma = \{(,)\}$

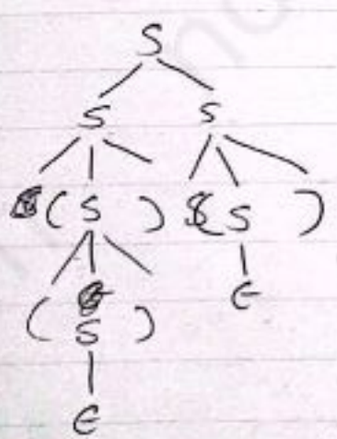
$R = \{S \to SS$

$S \to (S)$

$S \to \epsilon$

$\}$

To derive (()()



LMD	RMD
$S \to SS$	$S \to SS$
$\to (S)S$	$\to S(S)$
$\to ((S))S$	$\to S()$
$\to (())S$	$\to (S)()$
$\to (())(S)$	$\to ((S))()$
$\to (())()$	$\to (())()$

$L = a^n b^{2n+1}$

$M = (K, \Sigma, \Gamma, s, F, \Delta)$

$K = \{P, Q, f\}$

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

$\Gamma = \{a\}$

$s = \{P\}$

$F = \{f\}$

$\Delta = \{$

$(P, a, e) \rightarrow (P, a)$ ①

$(P, a, a) \rightarrow (P, a)$ ②

$(P, b, a) \rightarrow (Q, \epsilon)$ ③

$(Q, b, \epsilon) \rightarrow (P, \epsilon)$ ④

$(P, b, \epsilon) \rightarrow (f, \epsilon)$ ⑤

$\}$

For aabbbbb

State	Input	Stack	Ts
P	aabbbbb	–	–
P	aabbbbb	a	①
P	aabbbbb	aa	②
Q	aabbbbb	a	③
P	aabbbbb	a	④
Q	aabbbbb	–	③
P	aabbbbb	–	④
f	aabbbbb-	–	⑤

Accepted.

Name : Anish Maharjan
Roll No

OR

The CFG for $L = a^n b^{2n+1}$ is,

$G = (V, \Sigma, R, S)$
$\Sigma = \{a, b\}$
~~Re~~ $V =$
$R = \{ S \to aSbb$
$S \to b$
$\}$

For aabbbbb
$S \to aSbb$
$\to aaSbbbb$
$\to aabbbbb$

~~Ma~~ ~~CF~~ PDA is,

$M = (K, \Sigma, \Gamma, \Delta, S, F\}$
$K = \{$
$\Sigma = \{a, b\}$
$\Gamma = \{a, b\}$
$\Delta = \{ (P, \epsilon, \epsilon) \longrightarrow (P, S)$ ①
$(P, \epsilon, S) \longrightarrow (P, aSbb)$ ②
$(P, \epsilon, S) \longrightarrow (P, b)$ ③

$(P, a, a) \longrightarrow (P, \epsilon)$ ④
$(P, b, b) \longrightarrow (P, \epsilon)$ ⑤
$(P, \epsilon, \epsilon) \longrightarrow (f, \epsilon)$ ⑥
$\}$

$L = a^n b^{2n+1}$

$M = (k, \Sigma, \Gamma, s, F, \Delta)$

$k = \{P, Q, R, f\}$

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

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

$s = \{P\}$

$F = \{f\}$

$\Delta = \{$

$(P, a, \epsilon) \longrightarrow (P, a)$ ①

$(P, a, a) \longrightarrow (P, a)$ ②

$(P, b, a) \longrightarrow (Q, b)$ ③

$(Q, b, b) \longrightarrow (R, \epsilon)$ ④

$(R, \epsilon, a) \longrightarrow (P, \epsilon)$ ⑤

$(P, b, \epsilon) \longrightarrow (f, \epsilon)$ ⑥

$\}$

For aabbbbb

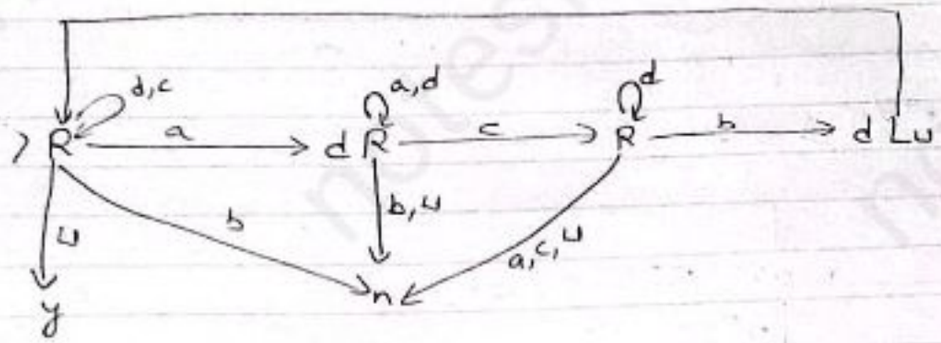
State	Input	Stack	TS
P	aabbbbb	–	–
P	aabbbbb	a	①
P	aabbbbb	aa	②
Q	aabbbbb	baa	③
R	aabbbbb	aa	④
P	aabbbbb	a	⑤
Q	aabbbbb	ba	③
R	aabbbbb	a	④
P	aabbbbb	–	⑤
f	aabbbbb–	–	⑥

increments any binary strings
1|0,L
0|1,L → (L)

State	Input	Stack	Transition used
P	aabbbbb	–	–
P	aabbbbb	S	1
P	aabbbbb	aSbb	~~3~~ ~~4~~ 2
P	aabbbbb	Sbb	4
P	aabbbbb	aSbbbb	2
P	aabbbbb	Sbbbb	4
P	aabbbbb	bbbbb	3
P	aabbbbb	bbbb	5
P	aabbbbb	bbb	5
P	aabbbbb	bb	5
P	aabbbbb	b	5
P	aabbbbb–	–	5

Hence accepted.

(8) Design a single Tape deterministic Turing machine which accepts all strings ~~fef~~ defined for the language $\{a^n c b^n, n \geq 0\}$ over alphabet $\{a, b, c\}$.



For aacbb

$\vdash \underline{u}aacbbu$
$\vdash^{R} u\underline{a}acbbu$
$\vdash^{d} u\underline{d}acbbu$
$\vdash^{R} udac\underline{b}bu$
$\vdash^{R} udacb\underline{b}u$
$\vdash^{d} udac\underline{d}bu$
$\vdash^{Lu} \underline{u}dacdbu$
$\vdash^{R} ud\underline{a}cdbu$
$\vdash^{d} udd\underline{c}dbu$
$\vdash^{R} udd\underline{c}dbu$
$\vdash^{R} uddcd\underline{b}u$
$\vdash^{d} uddcd\underline{d}u$
$\vdash^{Lu} \underline{u}ddcddu$
$\vdash^{d} uddcdd\underline{u}$
$\vdash^{y} uddcddy$

Accepted. //

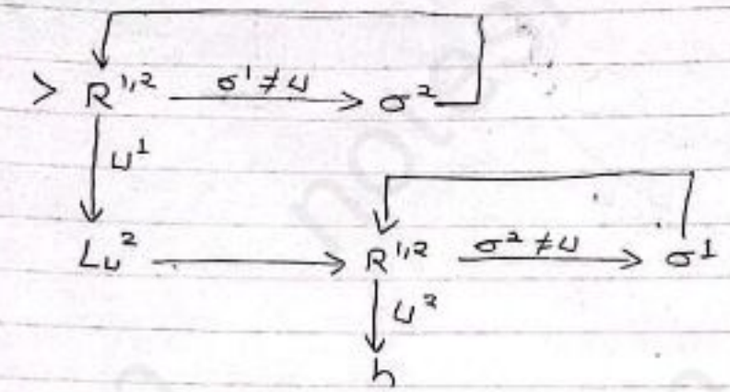
for acbb

$\vdash \underline{u}acbbu$
$\vdash^{R} u\underline{a}cbbu$
$\vdash^{d} ud\underline{c}bbu$
$\vdash^{R} udc\underline{b}bu$
$\vdash^{R} udcb\underline{b}u$
$\vdash^{d} udcd\underline{b}u$
$\vdash^{Lu} \underline{u}dcdbu$
$\vdash^{R} udcd\underline{b}u$
$\vdash^{n} udcdb\underline{u}$

Not accepted. //

increments any 1,0,2

(9) Design a multi tape Turing machine ~~wit~~ which acts as a copying machine over an alphabets $\Sigma = \{0,1\}^*$ that transforms the string #10# into #10#10#



	1st Tape	2nd Tape
$\vdash$	$\underline{\sqcup}\,1\,0\,\sqcup$	$\underline{\sqcup}\,\sqcup$
$\vdash^{R^{1,2}}$	$\sqcup\,\underline{1}\,0\,\sqcup$	$\underline{\sqcup}\,\underline{\sqcup}\,\sqcup$
$\vdash^{\sigma^2}$	$\sqcup\,\underline{1}\,0\,\sqcup$	$\sqcup\,\underline{1}\,\sqcup$
$\vdash^{R^{1,2}}$	$\sqcup\,1\,\underline{0}\,\sqcup$	$\sqcup\,1\,\underline{\sqcup}\,\sqcup$
$\vdash^{\sigma^2}$	$\sqcup\,1\,\underline{0}\,\sqcup$	$\sqcup\,1\,\underline{0}\,\sqcup$
$\vdash^{R^{1,2}}$	$\sqcup\,1\,0\,\underline{\sqcup}$	$\sqcup\,1\,\underline{0}\,\underline{\sqcup}$
$\vdash^{L_\sqcup^2}$	$\sqcup\,1\,0\,\underline{\sqcup}\,\sqcup$	$\underline{\sqcup}\,1\,0\,\sqcup$
$\vdash^{R^{1,2}}$	$\sqcup\,1\,0\,\sqcup\,\underline{\sqcup}$	$\sqcup\,\underline{1}\,0\,\sqcup$
$\vdash^{\sigma^1}$	$\sqcup\,1\,0\,\sqcup\,\underline{1}\,\sqcup$	$\sqcup\,\underline{1}\,0\,\sqcup$
$\vdash^{R^{1,2}}$	$\sqcup\,1\,0\,\sqcup\,1\,\underline{\sqcup}\,\sqcup$	$\sqcup\,1\,\underline{0}\,\sqcup$
$\vdash^{\sigma^1}$	$\sqcup\,1\,0\,\sqcup\,\underline{1}\,\underline{0}\,\sqcup$	$\sqcup\,1\,\underline{0}\,\sqcup$
$\vdash^{R^{1,2}}$	$\sqcup\,1\,0\,\sqcup\,1\,0\,\underline{\sqcup}$	$\sqcup\,1\,0\,\underline{\sqcup}$

halt //

73 Shrawan

① State pigeonhole principle. Prove the following statement by using mathematical induction

$$1 \times 1! + 2 \times 2! + 3 \times 3! + \dots + n \times n! = (n+1)! - 1 \quad \text{where } (n \geq 1)$$

Pigeonhole principle

If A & B are finite sets and $|A| > |B|$ then there is no one-to-one function from A to B

In other words, if we attempt to pair off the elements of A ("the pigeons") with the element of B ("the pigeon hole") sooner or later we will have to put more than one pigeons in a pigeonhole.

To prove:

$$1 \times 1! + 2 \times 2! + 3 \times 3! + \dots + n \times n! = (n+1)! - 1$$

Step1: Basis step

For $n = 1$

$LHS = 1 \times 1! = 1$

$RHS = (1+1)! - 1 = 2 - 1 = 1$

Step2: Induction hypothesis

Let us assume that for $n = k$

$$1 \times 1! + 2 \times 2! + 3 \times 3! + \dots + k \times k! = (k+1)! - 1$$

Step 3: Induction method

For $n = k+1$

We need to prove:

$$1\times1! + 2\times2! + 3\times3! + \dots + k\times k! + (k+1)\times(k+1)! = (k+1+1)! - 1 = (k+2)! - 1$$

$$\begin{aligned} LHS &= 1\times1! + 2\times2! + 3\times3! + \dots + k\times k! + (k+1)\times(k+1)! \\ &= (k+1)! - 1 + (k+1)(k+1)! \\ &= (k+1)![1+k+1] - 1 \\ &= [k+2](k+1)! - 1 \\ &= (k+2)! - 1 \\ &= RHS \end{aligned}$$

② Define a Configuration of ~~PDA~~ DFA. Design a DFA for language $L = \{w \in \{0,1\} : w$ has both 01 & 10 as a substrings$\}$. Verify your design by taking one accepted & one rejected string.

Configuration of DFA.

A configuration of deterministic finite automaton $M = (k, \Sigma, \delta, s, F)$ is any element of $K \times \Sigma^*$

Let $M = (k, \Sigma, \delta, S, F)$

$k = \{q_0, q_1, q_2, q_3, q_4, q_5\}$

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

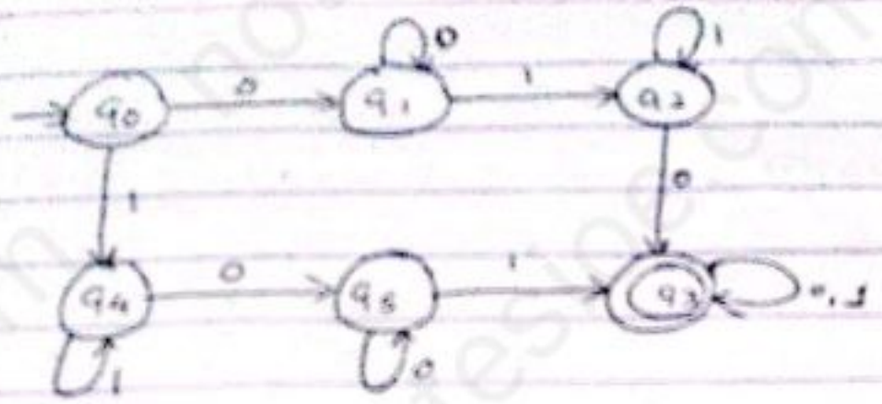
$S = \{q_0\}$

$F = \{q_3\}$

δ

$\Rightarrow$

$\delta \backslash \Sigma$	0	1
$\rightarrow q_0$	q_1	q_4
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_3	q_3
q_4	q_4	q_5
$*q_5$	q_5	q_3



w = 000111

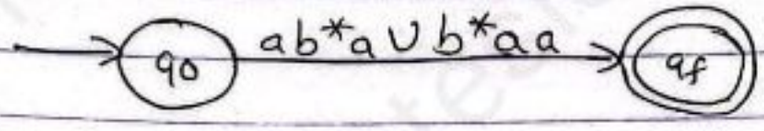
$\vdash q_0, \underline{0}00111$
$\vdash q_1, 0\underline{0}0111$
$\vdash q_1, 00\underline{0}111$
$\vdash q_1, 000\underline{1}11$
$\vdash q_2, 0001\underline{1}1$
$\vdash q_2, 00011\underline{1}$
$\vdash q_2, 000111_$

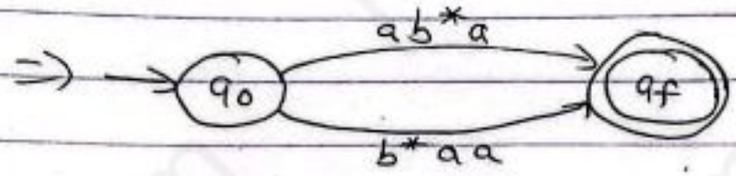
Not Accepted

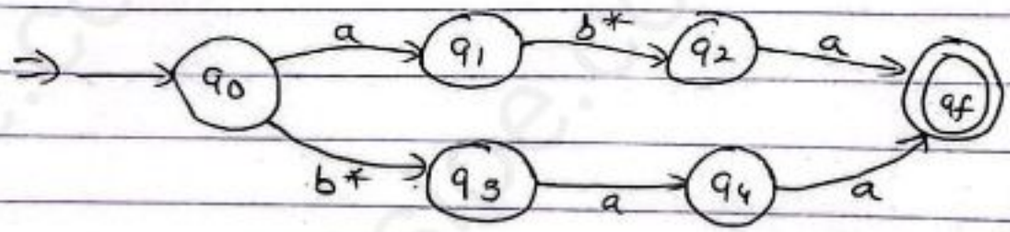
w = 011000

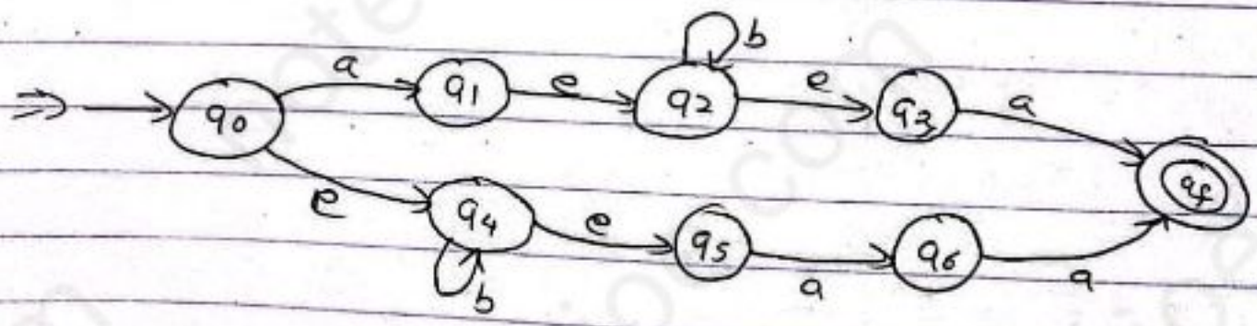
$\vdash q_0, \underline{0}11000$
$\vdash q_1, 0\underline{1}1000$
$011\underline{1}000$
011000

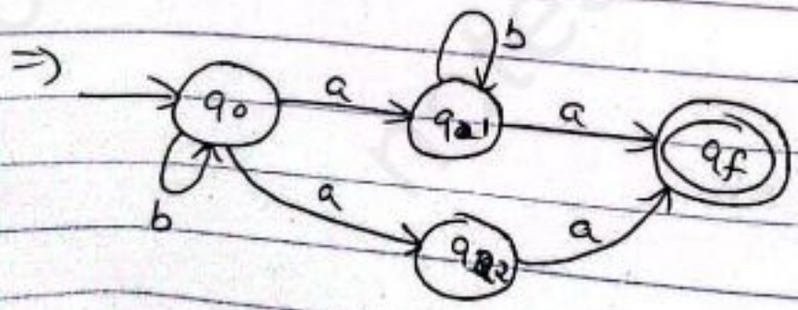
③ Construct a NFA for $(ab^*a \cup b^*aa)$











$\therefore$ NFA is,

$M = (k, \Sigma, \delta, s, F)$

$k = \{q_0, q_1, q_2, q_f\}$

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

$s = \{q_0\}$

$F = \{q_f\}$

$\delta =$

$\delta \backslash \Sigma$	a	b
$\rightarrow q_0$	q_1, q_2	q_0
q_1	q_f	q_1
q_2	q_f	–
q_f	–	–

Let $w = abba$

$\vdash q_0, \underline{a}bba$
$\vdash q_1, a\underline{b}ba$
$\vdash q_1, ab\underline{b}a$
$\vdash q_1, abb\underline{a}$
$\vdash q_f, abba_$
Accepted

$w = bbbaa$

$\vdash q_0, \underline{b}bbaa$
$\vdash q_0, b\underline{b}baa$
$\vdash q_0, bb\underline{b}aa$
$\vdash q_0, bbb\underline{a}a$
$\vdash q_2, bbba\underline{a}$
$\vdash q_f, bbbaa_$
Accepted

$w = abb$

$\vdash q_0, \underline{a}bb$
$\vdash q_1, a\underline{b}b$
$\vdash q_1, ab\underline{b}$
$\vdash q_1, abb_$
rejected

$w = abba$

$\vdash q_0, \underline{b}bba$
$\vdash q_0, b\underline{b}a$
$\vdash q_0, bba$
$\vdash q_1, bba_$
rejected.

increments any binary string
1/0,1 1,1

(4) State pumping lemma for regular language & use this theorem to prove that $L = \{a^n b^{2n} : n \geq 1\}$ is not regular

Pumping Lemma for regular language
Let L be any regular language
and w be any string $w \in L$
& n be a pumping constant $|w| \geq n$
then w can be rewritten as $w = xyz$
such that $|xy| \leq n$
$|y| \geq 1$
$xy^iz \in L$ for all $i \geq 0$

$L = \{a^n b^{2n} : n \geq 1\}$

$L = a^n b^{2n}$
$= \underbrace{aa\ldots aa}_{x}\underbrace{aa\ldots aa}_{y}\underbrace{bb\ldots bb}_{z}$

Let $y = a^k$ $\quad x = a^{n-k}$ $\quad z = b^{2n}$

For $i=0$, $xz = a^{n-k} b^{2n}$ $\quad , k \neq 0 \quad n-k \neq n$
$\notin L$

For $i=2$, $xy^2z = a^{n-k} a^{2k} b^{2n}$
$= a^{n+k} b^{2n} \notin L \quad \because k \neq 0 \quad n+k \neq n.$

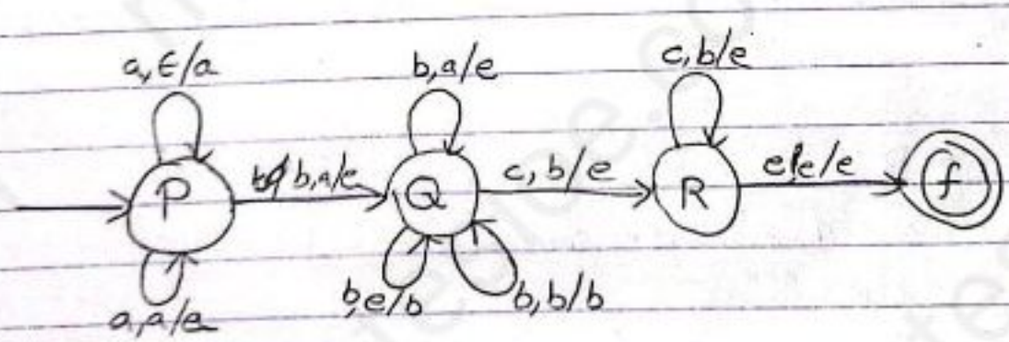
Hence L is not regular.

(5) Construct a PDA which accepts the language $L = \{a^n b^{n+m} c^m,\ n, m \geq 1\}$ verify your design for ~~aa~~ abbbcc

$M = (k, \Sigma, \Gamma, \Delta, s, F)$

$k = \{P, Q, R\}$
$\Sigma = \{a, b, c\}$
$\Gamma = \{a, b\}$
$s = \{P\}$
$f = \{f\}$

$\Delta = \{$
$(P, a, e) \rightarrow (P, a)$ (1)
$(P, a, a) \rightarrow (P, a)$ (2)
$(P, b, a) \rightarrow (Q, \epsilon)$ (3)
$(Q, b, a) \rightarrow (Q, \epsilon)$ (4)
$(Q, b, \epsilon) \rightarrow (Q, b)$ (5)
$(Q, b, b) \rightarrow (Q, b)$ (6)
$(Q, c, b) \rightarrow (R, \epsilon)$ (7)
$(R, c, b) \rightarrow (R, \epsilon)$ (8)
$(R, \epsilon, \epsilon) \rightarrow (f, \epsilon)$ (9)



increments any binary 1/0,2 -1,2 → (L)

w = abbbcc

State	Input	Stack	Transitions used
P	abbbcc	–	–
P	abbbcc	a	(1)
Q	abbbcc	b-	(3)
Q	abbbcc	b	(5)
Q	abbbcc	bb	(6)
R	abbbcc	b	(7)
R	abbbcc-	–	(8)
f	abbbcc-	–	(9)

Hence accepted.

(6) What is Chomsky Normal Form (CNF)?

$G = (S, L, M, N, a, b, c)$

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

$R = \{ S \to MaN \mid bL \mid bM$

$L \to ab \mid cN \mid M \mid e$

$M \to a \mid cM$

$N \to abN$

$\}$

=> ~~Chomsk~~

Simplying R,

N is unyielding production.

So removing N

$S \to bL / bM$
$L \to ab / ML / \epsilon$
$M \to a / cM$

Removing null production of L

$S \to bL / bM / b$
$L \to ab / ML / M$
$M \to a / cM$

Replacing unit production of M

$S \to bL / bM / b$
$L \to ab / ML / a / cM$
$M \to a / cM$

This is now simplified grammar

Now converting to CNF

$S \to bL \Rightarrow \boxed{S \to T_b L}$

$\boxed{T_b \to b}$

$S \to bM \Rightarrow \boxed{S \to T_b M}$ $\boxed{S \to b}$

$L \to ab \Rightarrow \boxed{L \to T_a T_b}$

$\boxed{T_a \to a}$

$L \to cM \Rightarrow \boxed{L \to T_c M}$

$\boxed{L \to a}$ $\boxed{\cancel{L \to cM}}$ $\boxed{T_c \to c}$

$M \to cM \Rightarrow \boxed{M \to T_c M}$

$\boxed{M \to a}$

Now CNF is,

$V = \{S, L, M, T_a, T_b, T_c, a, b, c\}$

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

$R = \{ S \to a \mid T_b L \mid T_a M$

$L \to T_a T_b \mid a \mid ML \mid T_c M$

$M \to T_c M \mid a$

$T_a \to a$

$T_b \to b$

$T_c \to c$

$\}$

(7) Construct a CFG for a language $L = a^n b^{2n}, n > 0$ & use this grammar to generate the string aabbbb. Also construct parse Tree.

The CFG is,

$G = (V, \Sigma, R, S)$

$V = \{S, a, b\}$

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

$R = \{ S \rightarrow aSbb$ (1)

$S \rightarrow \epsilon$ (2)

$\}$

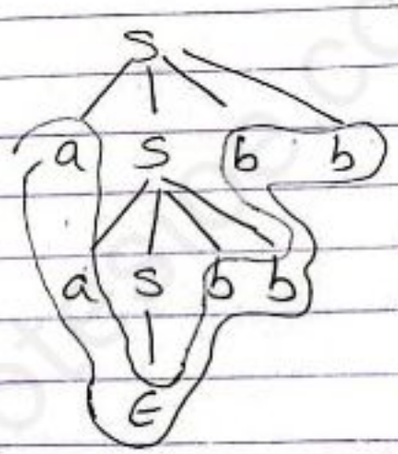
Derivation for aabbbb

$S \rightarrow aSbb$ (1)

$\rightarrow aaSbbbb$ (1)

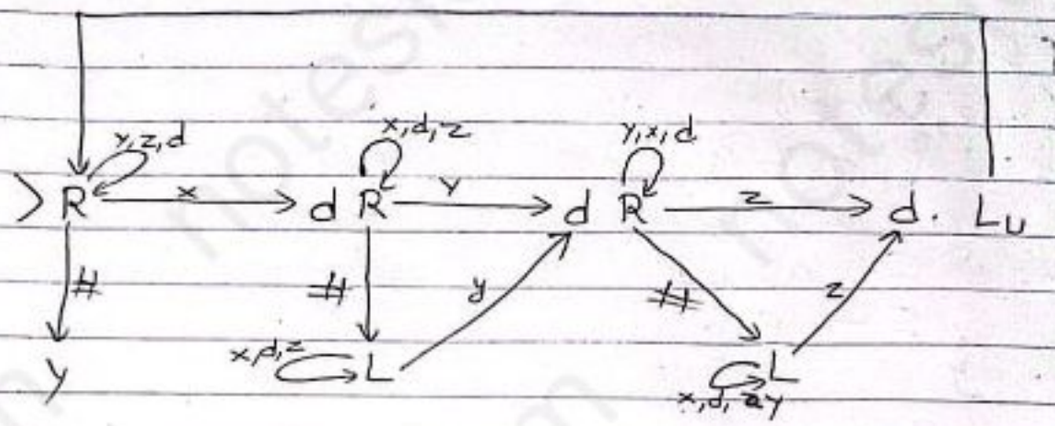
$\rightarrow aabbbb$ (2)

Parse tree



$\Rightarrow$ aabbbb

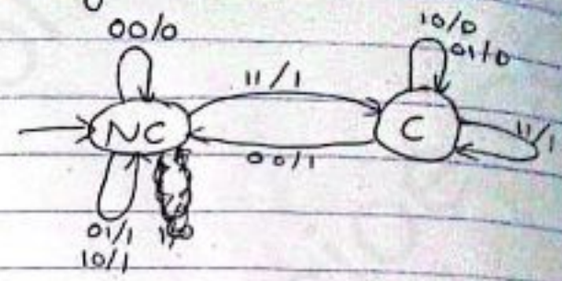
(8) Design a TM for $L = \{w \in \{x,y,z\}^* : w$ has equal no. of x's, y's & z's$\}$
Verify for #xyxyzz#

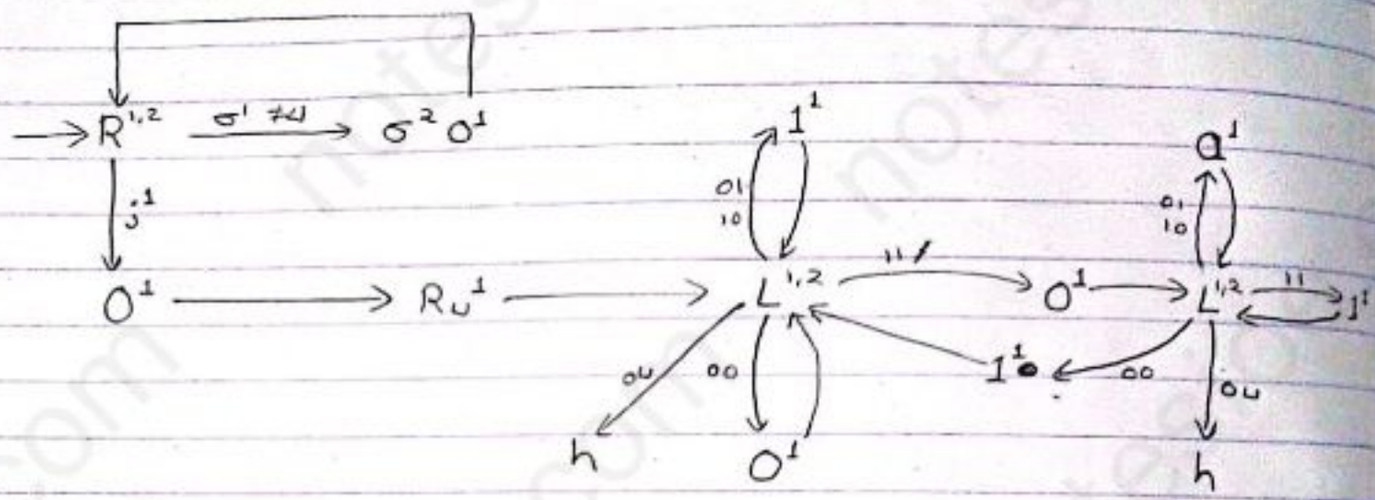


$\vdash$ #xyxyzz#
$\vdash^{R}$ #xyxyzz#
$\vdash^{d}$ #dyxyzz#
$\vdash^{R}$ #dyxyzz#
$\vdash^{d}$ #ddxyzz#
$\vdash^{R}$ #ddxyzz#
$\vdash^{d}$ #ddxydz#
$\vdash^{L_u}$ #ddxydz#
$\vdash^{R}$ #ddxydz#
$\vdash^{d}$ #dddydz#
$\vdash^{R}$ #dddydz#
$\vdash^{d}$ #ddddd z#
$\vdash^{R}$ #ddddd dz#
$\vdash^{d}$ #dddddd#
$\vdash^{L_u}$ #dddddd#
$\vdash^{R}$ #ddddd d#

Accepted

(9) Design a two tape turing maching that acts as binary adder.





For 10;01

	Tape 1	Tape 2
$\vdash$	⊔ 10; 01 ⊔	⊔ ⊔
$\vdash_{R^{1,2}}$	⊔ 10; 01 ⊔	⊔ ⊔ ⊔
$\vdash_{\sigma^2}$	⊔ 100; 01 ⊔	⊔ 1 ⊔
$\vdash_{0^1}$	⊔ 00; 01 ⊔	⊔ 1 ⊔
$\vdash_{R^{1,2}}$	⊔ 00; 01 ⊔	⊔ 1 ⊔ ⊔
$\vdash_{\sigma^2}$	⊔ 00; 01 ⊔	⊔ 10 ⊔
$\vdash_{0^1}$	⊔ 00; 01 ⊔	⊔ 10 ⊔
$\vdash_{R^{1,2}}$	⊔ 00: 01 ⊔	⊔ 10 ⊔
$\vdash_{0^1}$	⊔ 000 01 ⊔	⊔ 10 ⊔
$\vdash_{R_⊔^1}$	⊔ 00001 ⊔	⊔ 10 ⊔
$\vdash_{L^{1,2}}$	⊔ 00001 ⊔	⊔ 10 ⊔

q_0	⊔ 00001⊔	⊔ 1 0 ⊔
q_1	⊔ 00001⊔	⊔ 1 0 ⊔
q_0	⊔ 00011⊔	⊔ 1 0 ⊔
q_1	⊔ 00011⊔	⊔ 1 0 ⊔
h	⊔ 00011⊔	⊔ 1 0 ⊔

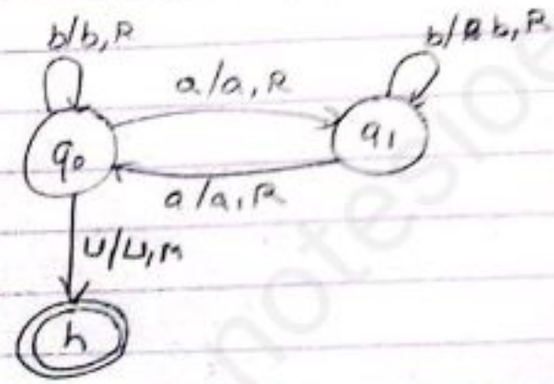
G_0 00011 is the ans //

(10) Explain Recursive & Recursively Enumerable Languages with suitable examples of each language

Recursive Language

~~The set of~~
A language is recursive if there exists a Turing machine that accepts every string of the language and rejects every string ~~of~~ not in the language

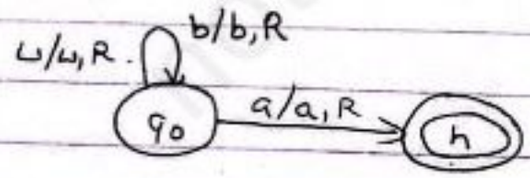
eg. $L = \{w \in \{a,b\}^*$, ~~aaaa~~ a substr has even no. of a's$\}$



Recursively Enumerable.

The language is said to be recursively enumerable if there exist a Turing machine that halts if ~~the~~ $w \in L$ and if $w \notin L$, the machine may reject w or go into an infinite loop.

eg. $L = \{w \in \{a,b\}^*, w$ contains at least one 'a'$\}$



(11) Define unrestricted grammar

2072 Kartik

① Explain what is an equivalent relation is.
Show by induction that for $n \geq 0$, $1+2+3+\ldots+n = (n^2+n)/2$

Equivalence relation
An relation that is reflexive, symmetric & transitive is called equivalence relation.

We write $[a]$ for the equivalence class containing an element a, provided the equivalence relation R i.e. $[a] = \{b : (a,b) \in R\}$ or since R is symmetric $[a] = \{b : (b,a) \in R\}$

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

Step 1: Basis step
For $n=1$, LHS $= 1$ RHS $= (1^2+1)/2 = 1$

Step 2: Induction hypothesis
For $n=k$, where $k \leq n$, $1+2+3+\ldots+k = (k^2+k)/2$

Step 3: Induction method
For $n=k+1$
To prove: $1+2+3+\ldots+k+k+1 = ((k^2+1)^2+(k+1))/2$

$$LHS = 1+2+3+\ldots+k+k+1$$
$$= \frac{k^2+k}{2} + k+1$$
$$= \frac{k^2+k+2k+2}{2}$$

INSTITUTE OF ENGINE
Examination Control Di
2071 Chaitra
Subject: - Theory of
...ts are required to give their answers in the
...estions.
...argin indicate Full Marks
...cessary.
...le sets with example. Use princip
...d integers $n \geq 0$.
...regular expression $(a(ab)^*b)^*$
... to prove language. [2+5]
...a to its [2+5]

$$= \frac{k^2+2k+1+k+1}{2}$$

$$= \frac{(k+1)^2+(k+1)}{2}$$

$$= RHS$$

② Let $\Sigma = \{a, b\}$. Write a regular expression for the language ~~in~~ with all string in Σ^* with no more 3 a's.

Regular expression is,
$(b^* a b^* a b^* a b^*)$

eg. - bbaabab

- babbabba

- baaabbb

③ Formally define DFA
DFA is a quadrupole $M = (k, \Sigma, \delta, S, F)$ where
k = set of states
Σ = alphabets
S = start symbol $S \in k$
F = Set of final states $F \subseteq k$
δ = Transition functions $k \times \Sigma \rightarrow k$

(1) Design a DFA accepting strings over the alphabet $\{0,1\}$ defined by $\{00\}^*\{11\}^*$

$M = (k, \Sigma, S, F, \delta)$

$k = \{q_0, q_1, q_2\}, q_3\}$

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

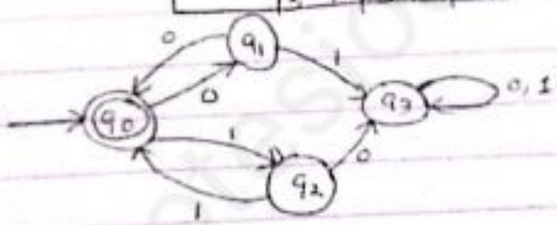
$S = \{q_0\}$

$F = \{q_0$

$\delta =$

$S \backslash \Sigma$	0	1
$\rightarrow q_0$	q_1	q_2
q_1	q_0	q_3
q_2	q_3	q_0
q_3	q_3	q_3

State diagram



For $w = 0011$

$\vdash q_0, \underline{0}011$
$\vdash q_1, 0\underline{0}11$
$\vdash q_0, 00\underline{1}1$
$\vdash q_2, 001\underline{1}$
$\vdash q_0, 0011 -$

Accepted

$w = 00111$

$\vdash q_0, \underline{0}0111$
$\vdash q_1, 0\underline{0}111$
$\vdash q_0, 00\underline{1}11$
$\vdash q_2, 001\underline{1}1$
$\vdash q_0, 0011\underline{1}$
$\vdash q_2, 00111 -$

Rejected.

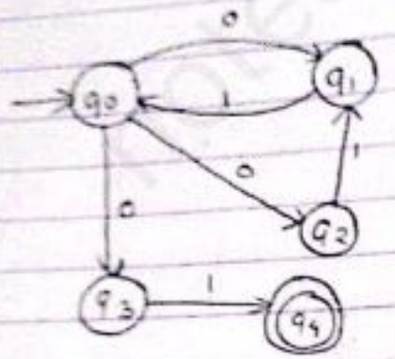
INSTITUTE OF ENGINEERING
Examination Control Division
2071 Chaitra

Subject - Theory of Co...

...are required to give their answers in their own ...
...margin indicate Full Marks
...necessary

...ate sets with example. Use principle of ...
...of integers $n \geq 0$. [3+...]
...regular expression $(a(ab)^*b)^*$. Ve... to prove language. [2+5]
...in to its [2+5]

(5) Convert the following NFA to DFA



$M = (K, \Sigma, S, F, \delta)$
$K = \{q_0, q_1, q_2, q_3, q_4\}$
$\Sigma = \{0, 1\}$
$S = \{q_0\}$
$F = \{q_4\}$
$\delta =$

$\delta \backslash \Sigma$	0	1	$\wedge$
q_0	q_1, q_2, q_3		–
q_1	–		q_0
q_2	–		q_1
q_3	–		q_4
q_4	–		–

~~f(q, 1)~~

$f(q_0, 0) \longrightarrow \{q_1, q_2, q_3\}$ (NS)
$f(q_0, 1) \longrightarrow \{\phi\}$ (NS)

$f(\{q_1, q_2, q_3\}, 0) \longrightarrow \{\phi\}$ (ES)
$f(\{q_1, q_2, q_3\}, 1) \longrightarrow \{q_0, q_1, q_4\}$ (NS)

$f(\{q_0, q_1, q_4\}, 0) \longrightarrow \{q_1, q_2, q_4\}$ (ES)
$f(\{q_0, q_1, q_4\}, 1) \longrightarrow \{q_0\}$ (ES)

The DFA is,

$M = (k, \Sigma, S, F, \delta)$

$k = \{\{q_0\}, \{q_0, q_1, q_4\}, \{q_1, q_2, q_4\}, \{\phi\}\}$

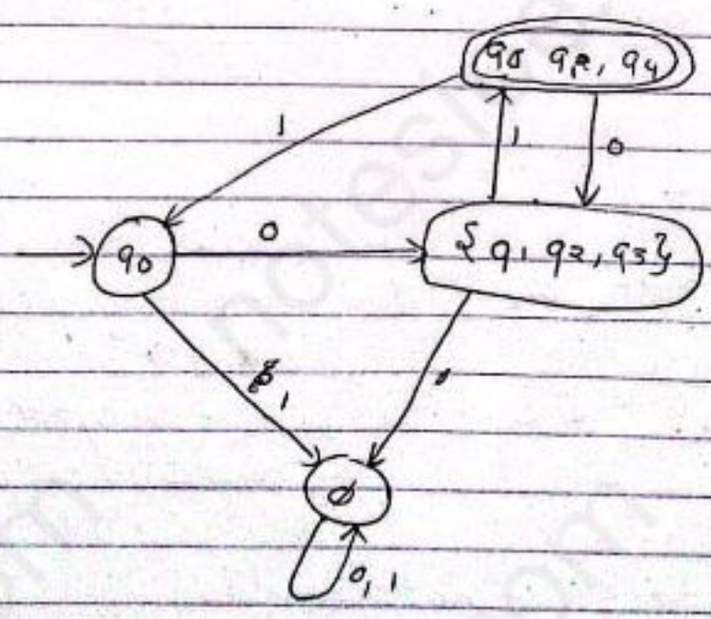
$\Sigma = \{0, 1\}$

$S = \{q_0\}$

$f = \{\{q_0, q_1, q_2, q_4\}$

$\delta =$

$\delta \backslash \Sigma$	0	1
q_0	$\{q_1, q_2, q_3\}$	$\{\phi\}$
$\{q_0, q_1, q_4\}$	$\{q_1, q_2, q_3\}$	$\{q_0\}$
$\{q_1, q_2, q_3\}$	$\{\phi\}$	$\{\phi\}$
ϕ	$\{\phi\}$	$\{\phi\}$



INSTITUTE OF ENGINEERING
Examination Control Division
2071 Chaitra

Subject : Theory of Computation

margin indicate Full Marks

necessary

(6) State pumping lemma for regular language. Show that the language $L = \{a^n : n \text{ is prime}\}$ is not regular.

Pumping lemma for regular Language

Let L be a regular language
w be an string $w \in L$
& n be a pumping constant $n \geq 1$ & $|w| > n$
then w can be rewritten as

$w = xyz$

such that $|xy| \leq n$

$|y| \geq 1$

$xy^iz \in L$ for all $i \geq 0$

$w = xyz = a^n$

$w = xyz = \underbrace{aa\ldots aa}_{x}\underbrace{aa\ldots aa}_{y}\underbrace{aa\ldots aa}_{z}$

$y = a^k$, $k \geq 1$

For $i=0$, $xz = a^{n-k} \notin L$ Since $n-k$ may not be prime as $k \neq 0$.

For $i=2$, $xy^2z = a^{n+k} \notin L$ Since $n+k$ may not be prime as $k \neq 0$

Hence $\{a^n : n \text{ is prime no}\}$ is not regular.

⑦ Define Context Free Grammar (CFG) along with an example.

A context Free Grammar (CFG) is a quadruple $G = (V, \Sigma, R, S)$ where

V is a alphabet

Σ is the set of terminals (i.e. $\Sigma \subseteq V$)

R is the set of rules or productions, subset of $(V - \Sigma) \times V^*$

S is start symbol $S \in V - \Sigma$

eg.

For the the Language $L = \{a^n, b^n, n \geq 0\}$

$G = (V, \Sigma, R, S)$

$V = \{S, a, b\}$

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

$S = S$

$R = \{ S \rightarrow aSb$

$S \rightarrow e$

$\}$

(5) Design PDA to accept $L = \{ww^R,\ w \in \{a,b\}^*\}$
Show how it accepts "abbbba"

$M = (K, \Sigma, \Delta, \Gamma, F, S)$

$K = \{$

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

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

$F = \{f\}$

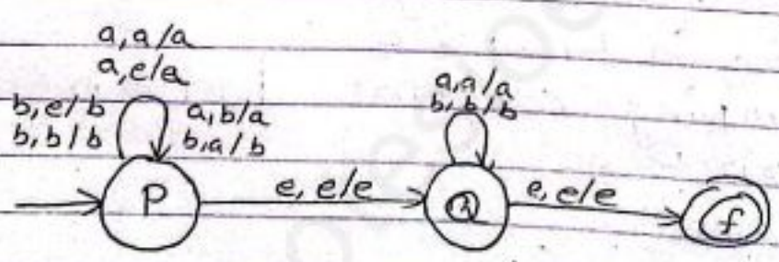
$S = \{p\}$

$\Delta = \{$

$(P, a, e) \longrightarrow (P, a)$	(1)
$(P, b, e) \longrightarrow (P, b)$	(2)
$(P, a, a) \longrightarrow (P, a)$	(3)
$(P, b, b) \longrightarrow (P, b)$	(4)
$(P, a, b) \longrightarrow (P, a)$	(5)
$(P, b, a) \longrightarrow (P, b)$	(6)
$(P, \epsilon, \epsilon) \longrightarrow (Q, \epsilon)$	(7)
$(Q, a, a) \longrightarrow (Q, \epsilon)$	(8)
$(Q, b, b) \longrightarrow (Q, \epsilon)$	(9)
$(Q, \epsilon, \epsilon) \longrightarrow (f, \epsilon)$	(10)

$\}$

State Diagram,



For abbbba

State	Input	Stack	Transitions used
P	abbbba	a-	- 1
P	abbbba	a	1 4
P	abbbba	ba	6 4
P	abbbba	bba	4 8 7
P	abbbba	bba	7 9
Q	abbbba	ba	9 9
Q	abbbba-	a	9
Q	abbbba-	-	8
f	abbbba-	-	10

Hence Accepted //

(10) Using pumping theorem for context free language show that $L = \{a^n b^n c^n ; n \geq 0\}$ is not context free.

Let L be a context free Language
w be any string in L, $w \in L$
n be any pumping constant $|w| \geq n$
Then w can written as:

$w = uvxyz$

such that $|vxy| \leq n$

$|vy| \geq 0$

$uv^ixy^iz \in L$

$$w = uvxyz$$
$$= \underbrace{aa\ldots aaa}_{u}\underbrace{aabb\ldots bb}_{vxy}\underbrace{bbcc\ldots cc}_{z}$$

$u = a^{n-p-q}$
$v = a^{p}$
$x = a^{q} b^{r}$
$y = b^{s}$
$z = b^{n-s-r} c^{n}$

For $i=0$, $uxz = a^{n-p-q} a^{q} b^{r} b^{n-s-r} c^{n}$
$= a^{n-p} b^{n-s} c^{n} \notin L$

Since $p \neq 0$, ~~$n-p \neq 0$~~ $\Rightarrow n-p \neq n$
$s \neq 0$, $\Rightarrow n-s \neq n$

For $i=2$ $uv^{2}xy^{2}z = a^{n-p-q} a^{2p} a^{q} b^{r} b^{2s} b^{n-s-r} c^{n}$
$= a^{n+p} b^{n+s} c^{n} \notin L$

Since, $p \neq 0$, $n+p \neq n$
$s \neq 0$, $n+s \neq n$

So $L = \{a^{n} b^{n} c^{n}, n \geq 1\}$ is not context free.

(11) What is a Turing machine? Describe its operation

A Turing machine is a ~~quadruple~~ quintuple
$M = (K, \Sigma, \delta, s, H)$
K = a finite set of states
Σ = an alphabet containing the blank symbol $\sqcup$ or # and left end symbol $\triangleright$ but not containing the symbol $\rightarrow$ or $\leftarrow$
$s \in K$ is the initial state
$H \subseteq K$ is halting state.
δ is the transition functon
$(K-H) \times \Sigma$ to $K \times (\Sigma \cup \{\leftarrow, \rightarrow\})$

The turing machine consists of a finite state control unit & a tape a follows

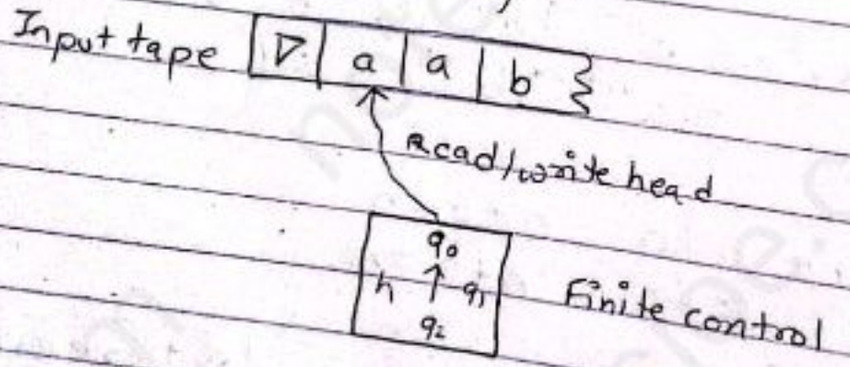


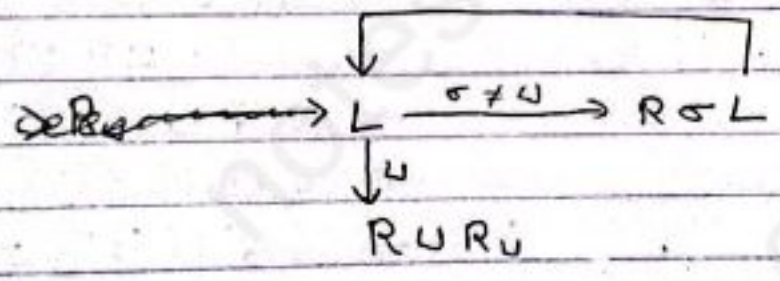
Fig. A Turing machine

The communication of tape & finite control is done by read/write head.

Now the control unit either changes state, reads the input, writes on the tape or moves the head of the tape to left or right.

The tape has a left end marke '$\triangleright$' & is extended infinitely to the right. '$\rightarrow$' & '$\leftarrow$' symbol is used for head movement.

(12) Construct a Turing machinge to Transform $\sqcup w \sqcup$ to $\sqcup \sqcup w \sqcup$



$\sqcup a b \underline{\sqcup}$

$\vdash^{L} \sqcup a \underline{b} \sqcup$

$\vdash^{R} \sqcup a b \underline{\sqcup}$

$\vdash^{\sigma} \sqcup a b \underline{b} \sqcup$

$\vdash^{L} \sqcup a \underline{b} b \sqcup$

$\vdash^{L} \sqcup a b b \sqcup$

$\vdash^{R} \sqcup a \underline{b} b \sqcup$

$\vdash^{\sigma} \sqcup a \underline{a} b \sqcup$

$\vdash^{L} \sqcup \underline{a} a b \sqcup$

$\vdash^{L} \underline{\sqcup} a a b \sqcup$

$\vdash^{R} \sqcup \underline{a} a b \sqcup$

$\vdash^{\sqcup} \sqcup \underline{a} \sqcup a b \sqcup$

$\vdash^{R\sqcup} \sqcup \sqcup a b \underline{\sqcup}$

(13) Construct a Grammar to accept the language
$L = \{a^n b^n c^n : n \geq 1\}$

$G = (V, \Sigma, R, S)$
$V = \{$
$\Sigma = \{a, b, c\}$
$S = \{S\}$
$R = \{$

$S \rightarrow ABCS$	(1)
$S \rightarrow T_c$	(2)
$CA \rightarrow AC$	(3)
$BA \rightarrow AB$	(4)
$CB \rightarrow BC$	(5)
$CT_c \rightarrow T_c c$	(6)
$BT_c \rightarrow T_b b$	(7)
$BT_b \rightarrow T_b b$	(8)
$AT_b \rightarrow T_a a$	(9)
$AT_a \rightarrow T_a a$	(10)
$T_a \rightarrow e$	(11)

$\}$

For aabbcc

$S \rightarrow ABCS$	(1)
$\rightarrow ABCABCS$	(1)
$\rightarrow ABCABCT_c$	(2)
$\rightarrow ABCABCT_c$	(3)
$\rightarrow AABCBCT_c$	(4)
$\rightarrow AABBCCT_c$	(5)
$\rightarrow AABBCT_c c$	(6)
$\rightarrow AABBT_c cc$	(6)
$\rightarrow AABT_b bcc$	(7)
$\rightarrow AAT_b bbcc$	(8)
$\rightarrow AT_a abbcc$	(9)
$\rightarrow T_a aabbcc$	(10)
$\rightarrow aabbcc$	(11)

(14) Explain Church Turing Thesis. Show that that Halting problem is undecidable.

Church Turing Thesis

It is believed that there is no functions that can be defined by humans whose calculations can be described by any well defined mathematical algorithm that can be performed, that cannot be computed by a turing machine.

In other words,
No computational procedure will be considered as a algorithm unless it is represented by a Turing machine.

Church turing thesis can not be called as a theorem because ideas such as "can ever be defined by humans" and "algorithm that people can taught to perform" are not any part of any known mathematics.