B.Tech (IT)

SOLUTION (Theory of Computation)

Subject code: IT-14503

Paper Id: 15405

Total Number of Pages: 19

Ajay K (Rupinder Kaur)

Q1.

Ans: Chomsky Hierarchy represents the class of languages that are accepted by the different machine. The category of language in Chomsky's Hierarchy is as given below:

1. Type 0 known as Unrestricted Grammar.
2. Type 1 known as Context Sensitive Grammar.
3. Type 2 known as Context Free Grammar.
4. Type 3 Regular Grammar.

Grammar Type	Grammar Accepted	Language Accepted	Automaton
Type 0	Unrestricted grammar	Recursively enumerable language	Turing Machine
Type 1	Context-sensitive grammar	Context-sensitive language	Linear-bounded automaton
Type 2	Context-free grammar	Context-free language	Pushdown automaton
Type 3	Regular grammar	Regular language	Finite state automaton

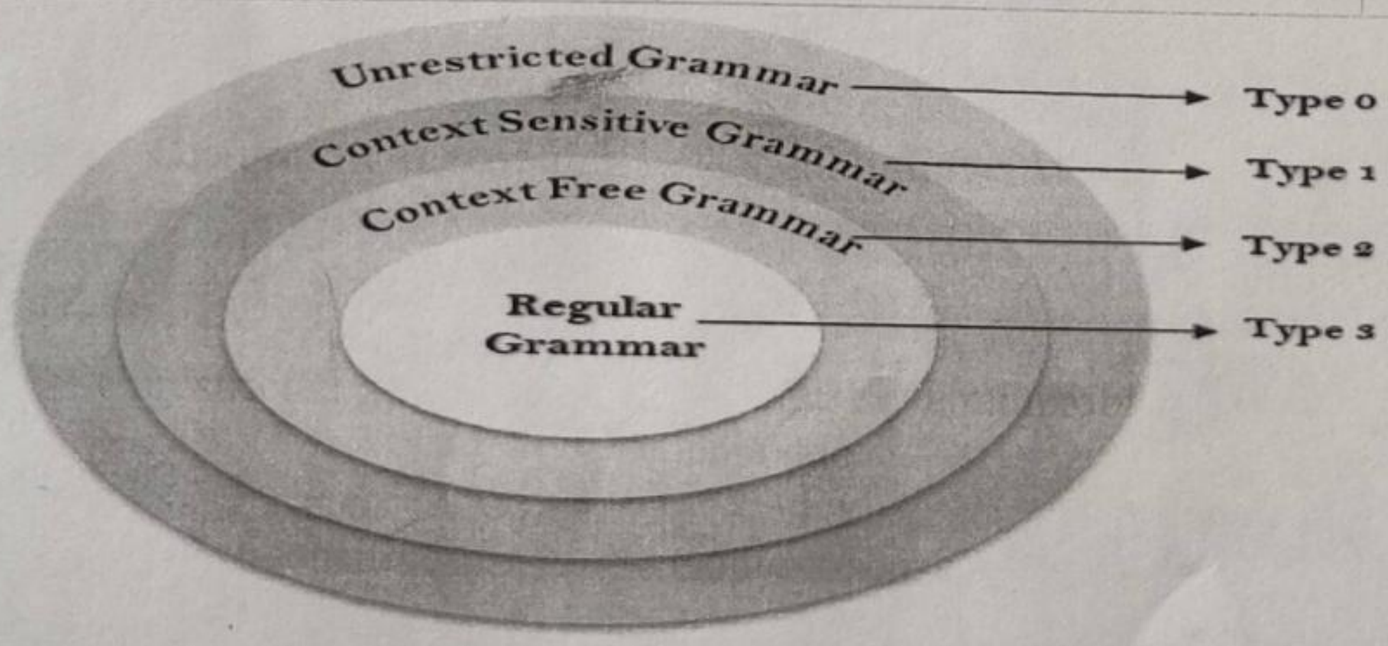


Fig: Chomsky Hierarchy

Scanned by TapScanner

Examples: S → X a
X → a
X → aX
X → abc
X → ε

Type 3:

Type-3 grammars generate regular languages. Type-3 grammars must have a single non-terminal on the left-hand side and a right-hand side consisting of a single terminal or single terminal followed by a single non-terminal.

The productions must be in the form **X → a or X → aY**

where **X, Y ∈ N** (Non terminal)

and **a ∈ T** (Terminal)

The rule **S → ε** is allowed if **S** does not appear on the right side of any rule.

Example
X → ε
X → a | aY
Y → b

140

K. L. P. Mashuq

Q2.

Theory of Computation.

String

Alphabet.

Σa^2

TOC.

Flat

Finite Automata

First Lec Topic →

String

Alphabet

Klein star Σ^*

Klein Closure Σ^+

Scanned by TapScanner

Type - 0 Grammar

Type-0 grammars generate recursively enumerable languages. The productions have no restrictions. They are any phase structure grammar including all formal grammars.They generate the languages that are recognized by a Turing machine.

The productions can be in the form of $\alpha \rightarrow \beta$ where α is a string of terminals and nonterminals with at least one non-terminal and α cannot be null. β is a string of terminals and non-terminals.

Examples: S → ACaB

Bc → acB

CB → DB

aD → Db

Type 1 Grammar:

Type 1 grammar is known as Context Sensitive Grammar. The context sensitive grammar is used to represent context sensitive language. The context sensitive grammar follows the following rules:

- The context sensitive grammar may have more than one symbol on the left hand side of their production rules.
- The number of symbols on the left-hand side must not exceed the number of symbols on the right-hand side.
- The rule of the form $A \rightarrow \varepsilon$ is not allowed unless A is a start symbol. It does not occur on the right-hand side of any rule.
- The Type 1 grammar should be Type 0. In type 1, Production is in the form of $V \rightarrow T$

Where the count of symbol in V is less than or equal to T.

Examples: S → AT

T → xy

A → a

Type 2 Grammar:

Type-2 grammars generate context-free languages.The productions must be in the form $A \rightarrow \gamma$

where $A \in N$ (Non terminal)

and $\gamma \in (T \cup N)^*$ (String of terminals and non-terminals).

These languages generated by these grammars are be recognized by a non-deterministic pushdown automaton.

Scanned by TapScanner

22 Pumping Lemma for Regular Languages.

Soln The term pumping is made up of two words one is "pumping" & second is "lemma". The word pumping means to generate many input strings by pushing a symbol in an input string again and again.

Pumping Lemma is used to prove that given language is not regular.

Conditions for Pumping Lemma

Step1. Assume that L is regular. Let n be the number of states in corresponding finite automaton

Step2 Choose a string w such that $|w| \geq n$ Use pumping lemma to write

$w = xyz$ with $|xy| \leq n$ & $|y| \geq 0$.

Step3 Find a suitable integer i such that $xy^iz \notin L$. This contradicts our assumption. Hence, L is not regular.

Prove that language

$L = \{ a^p$ where p is prime number is not regular $\}$

Scanned by TapScanner

Step 1. Suppose L is regular

Step 2 Let p be a prime number greater than n.

$\boxed{p = 2, 3, 5, 7, 11}$ → prime numbers.

Let $p = 3$

$a^3 = aaa$

No. of states $\Rightarrow n = 4$.

Choose w such that $|w| \geq n$

but $p = 5$

$a^5 = aaaaa$ $|w| \Rightarrow$ length of string $= 5$.

$\boxed{5 \geq 4}$

Step 3 ~~Choose xy^iz~~ Divide w into three parts

But $i = 0$

$w = xz$ ≠

$w = xyz$

$= aaaaa$

$x = a$

$y = aa$ $\boxed{|xy| \leq n}$

$z = aa$

Step 4 Choose $xy^iz \Rightarrow i = 0$

$w = xz = \boxed{aaa}$

Scanned by TapScanner

Put $y=1$

$w = xyz$

$w = aaaaa$

Put $y=2$

$w = xy^2z$

$= xyyz$

$= \boxed{aaaaaaaa} \Rightarrow a^7$

Put $y=3$, $w = xy^3z$

$= a\,aaaaaaaaa \Rightarrow a^9$.

9 is not a prime number.

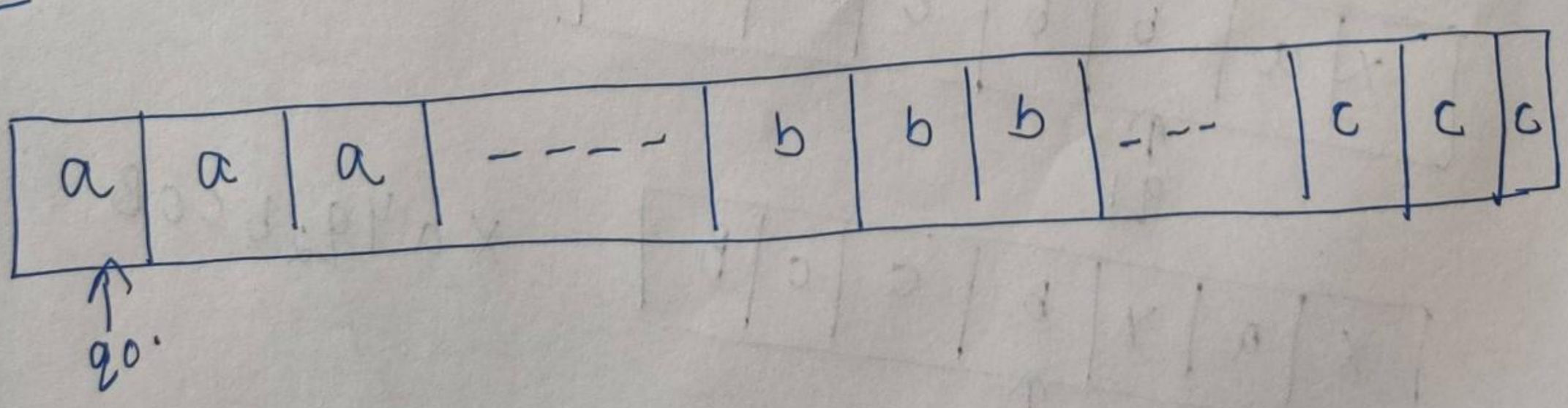
Hence, our assumption contradicts.

This shows $L = \{a^p \mid p \text{ is prime}\}$ is not regular.

Q3 Design a Turing Machine to recognize language

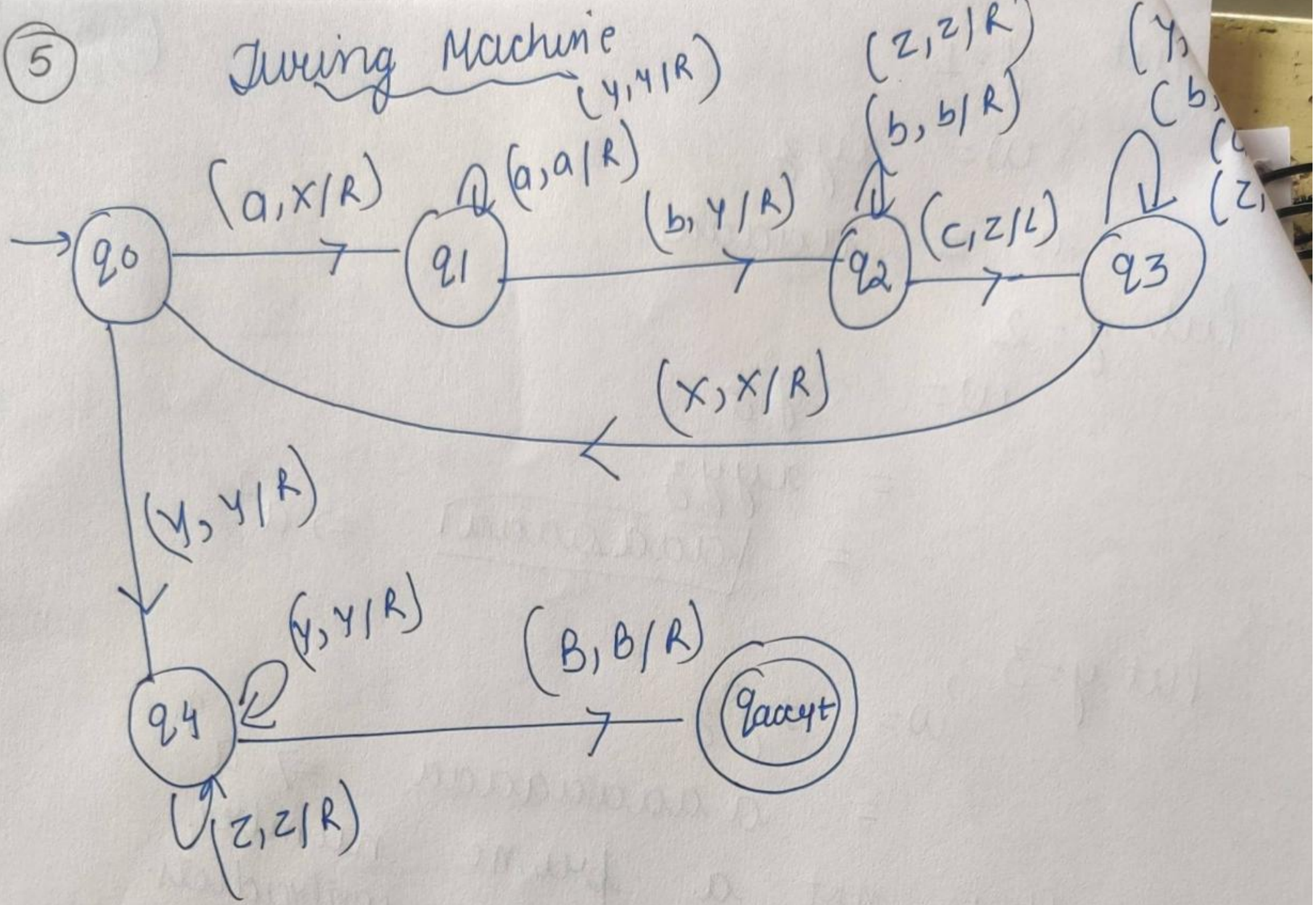
$$L = \{a^n b^n c^n \mid n \geq 1\}$$

Soln $w = \{abc, aabbcc, aaabbbccc \cdots\}$

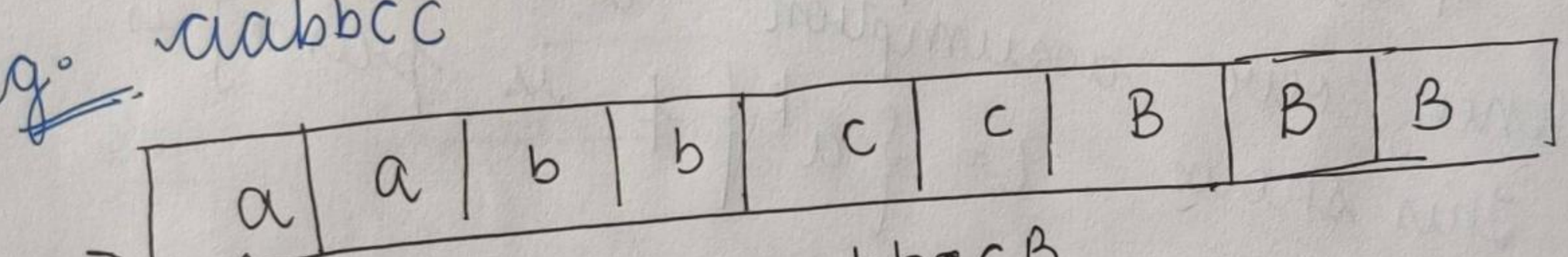


Scanned by TapScanner

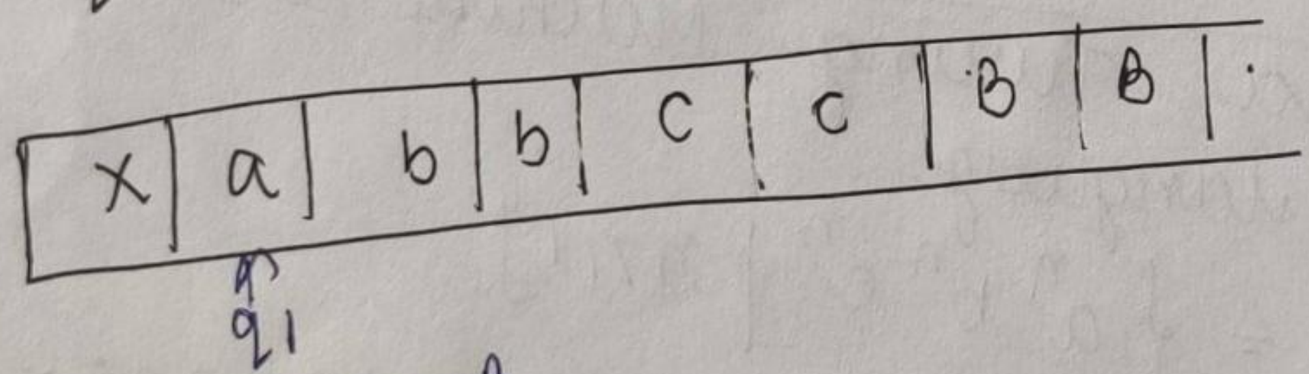
(5) Turing Machine



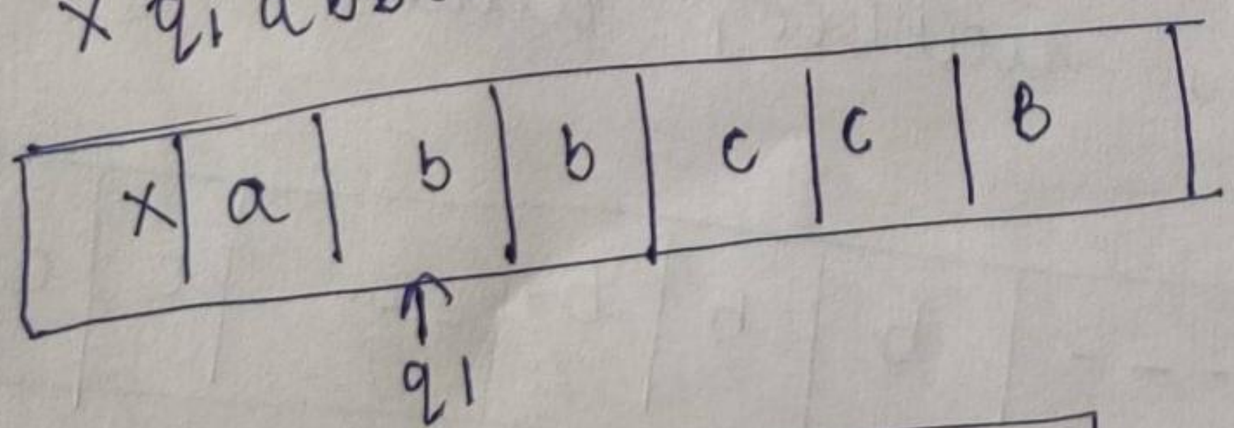
eg. aabbcc



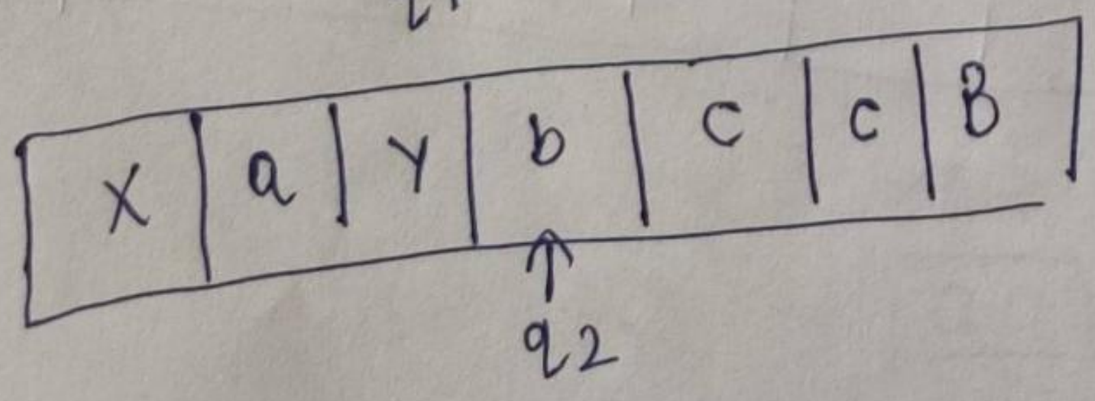
q_0 aabbccB



X q_1 abbccB



X a q_1 bbccB



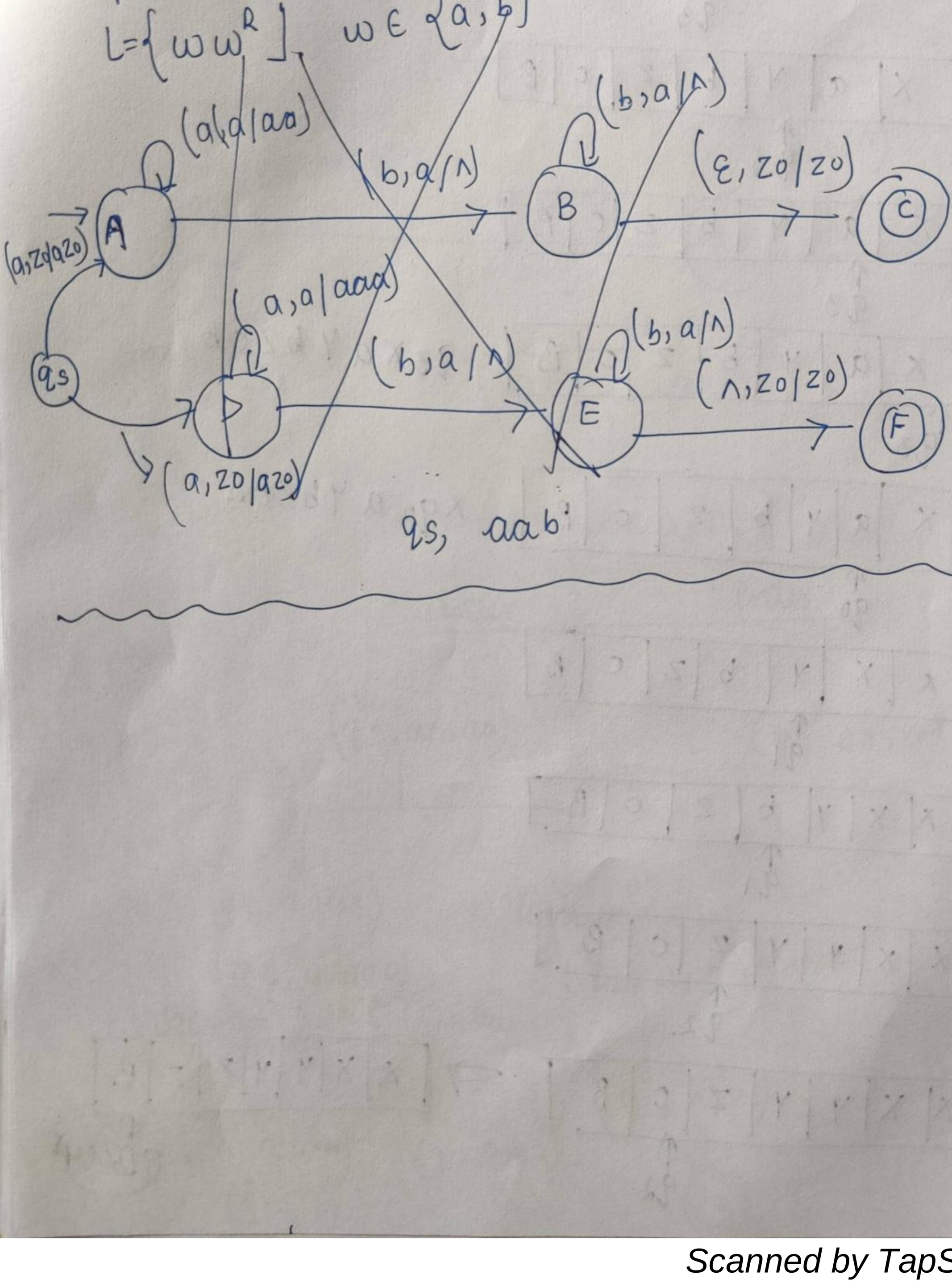
X a Y q_2 b c c B

Scanned by TapScanner

Design a Push Down Automata PDA for even palindromes Trace your with example.

Ans:- Push Down Automata for even palindrome

$L = \{ w w^R \mid w \in \{a, b\}^* \}$



qs, aab

Scanned by TapScanner

x	a	Y	b	c	c	B

q_2

$x\ a\ Y\ b\ q_2\ c\ c\ B$

x	a	Y	b	z	c	B

q_3

$x\ a\ Y\ q_3\ b\ z\ c\ B$

x	a	Y	b	z	c	B

q_3

x	a	Y	b	z	c	B

q_3

$q_3\ x\ a\ Y\ b\ z\ c\ B$

x	a	Y	b	z	c	B

q_3

x	a	Y	b	z	c	B

q_0

$x\ q_0\ a\ Y\ b\ z\ c\ B$

x	x	Y	b	z	c	B

q_1

x	x	Y	b	z	c	B

q_1

x	x	Y	Y	z	c	B

q_2.

x	x	Y	Y	z	c	B

q_2.

$\Rightarrow$

x	x	Y	Y	z	z	B

q_B accept

Scanned by TapScanner

Design a PDA for even number of palindromes.

Soln Let $P = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ be PDA which recognize the given language

where $Q = \{q_0, q_1 -- q_f\}$

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

Scanned by TapScanner

Q5 State and Prove Pumping Lemma for Context Free Languages with suitable ex

Soln Pumping Lemma is a theorem used to prove that certain languages are not context free.

Let L be a context free language then there is a constant 'n' depending only on L such that if $w \in L$ and $|w| \geq n$ then we may be divided into five pieces

$w = uvxyz$

(i) for all $i \geq 0$, $uv^i xy^i z \in L$

(ii) $|vy| \geq 1$

(iii) $|vxy| \leq n$.

eg. Prove that language $L = \{a^i b^i c^i \mid i \geq 0\}$ is not CFG.

Step 1 → Assume that the language is context free language

Find w such that $|w| \geq n$

for $i=1$, $w = abc$ $n=4$

for $i=2$ $w = aabbcc$ $|w| \geq n$.

$6 \geq 4$

Step 2 Divide w into five parts

$w = uvxyz$.

$u = a$ $y = b$

$v = a$ $z = cc$

$x = b$

Scanned by TapScanner

Substituting q_2 in q_1

$$q_1 = q_1 a + q_1 a(b+aa)^* b + \wedge$$

$$q_1 = q_1 [a + a(b+aa)^* b] + \wedge$$

$$q_1 = (a + a(b+aa)^* b)^*$$

$$q_2 = (a + a(b+aa)^* b)^* a(b+aa)^*$$

$$\boxed{q_3 = (a + a(b+aa)^* b)^* a(b+aa)^* a}$$

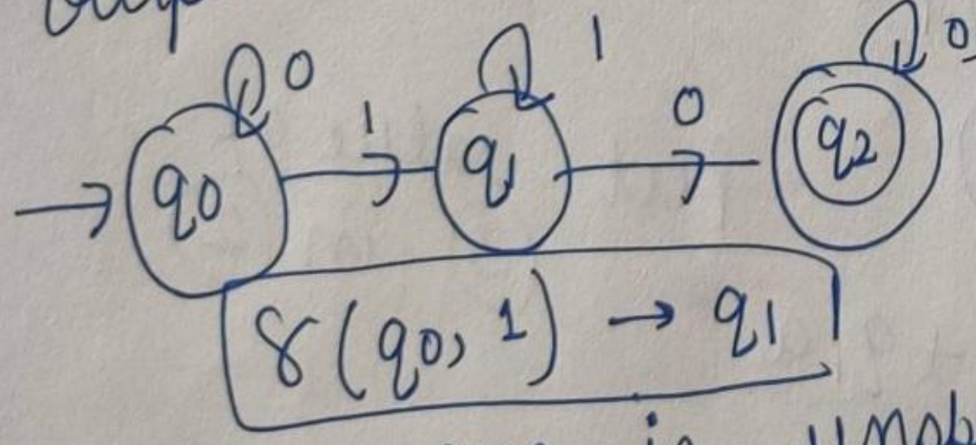
Any

Q7. (I) Differentiate DFA and NDFA
(II) Compare Moore and Mealy Machines

Ans (I)

DFA

1) The transition function δ takes a state & input symbol as argument & returns exactly one state as output

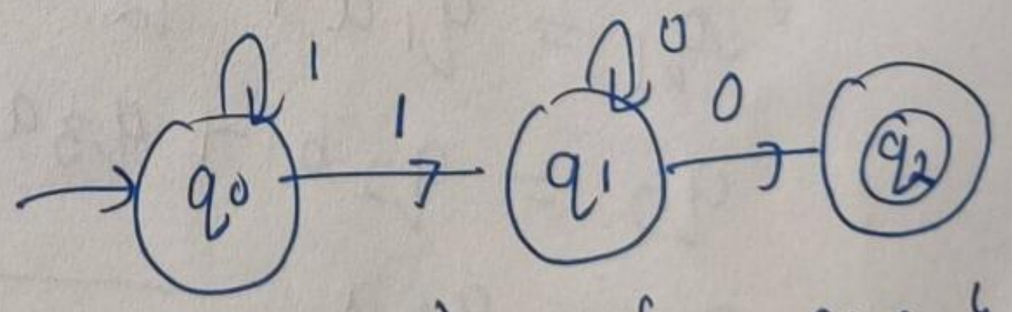


2) No move is unspecified in DFA

3) In DFA, we cant move from one state to another without consuming a symbol.

NDFA

1) The transition function δ takes a state and input symbol as argument & returns no. of choices for next state



2) Some moves can be unspecified in NDFA
eg - $\delta(q_2, 1) = \phi$.

3. In NDFA, we can make a transition without

Scanned by TapScanner

Put $i=0$

$w = u v^i x y^i z$

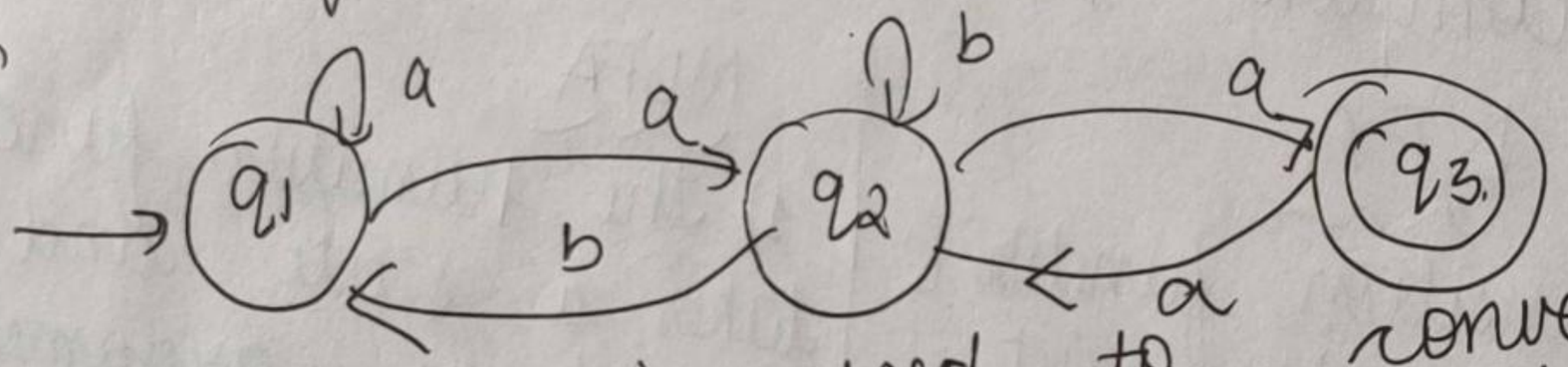
$w = uxz$

$= abcc \Rightarrow \boxed{abc^2}$

This language does not contain equal number of a's and b's Hence, our assumption contradicts. So, this language is not context free language.

Q6 Construct a Regular expression corresponding to given automata

Ans



Arden's Theorem is used to convert a given Automata into regular expression.

$q_1 = q_1 a + q_2 b + \wedge$ — (1)

$q_2 = q_2 b + q_3 a + q_1 a$ — (2)

$q_3 = q_2 a$ — (3) Put value of (3) in (2)

Using (2) $q_2 = q_2 b + q_2 aa + q_1 a$

$q_2 = q_2 (b + aa) + q_1 a$

$\boxed{q_2 = q_1 a (b + aa)^*}$

$\boxed{R = Q + RP \quad R = QP^*}$

Scanned by TapScanner

when we convert more to Mealy, we have same no. of states & same no of edges

4) When we convert mealy to moore, states & edges increase

Q8 Write note on
(I) Leftmost & Rightmost Derivation Trees

Ans Derivation in CFG is represented using Trees.
Such Trees are known as Derivation Trees.

Left Most Derivation Tree

$A \overset{*}{\Rightarrow} w$ is called leftmost derivation if we apply a production only to the left most variable at every step.

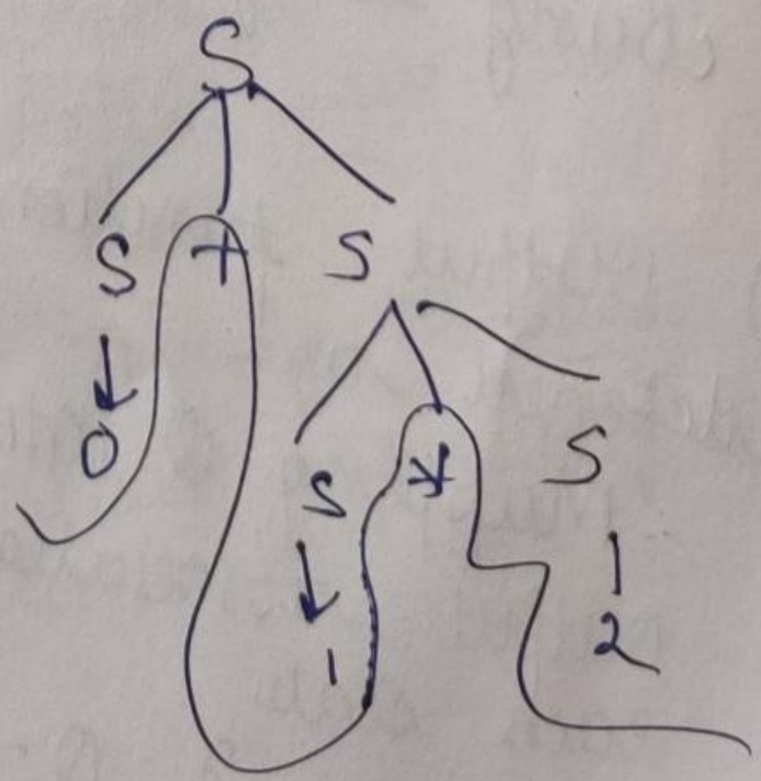
Right Most Derivation Tree

$A \overset{*}{\Rightarrow} w$ is called rightmost derivation if we apply production only to the right most variable at every step

Eg $S \to S * S$
$S \to S + S$
$S \to 0$
$S \to 1$
$S \to 2$

Left Most Derivation

$S \to S + S$
↓
$0 + S * S$
$0 + 1 * S$
$0 + 1 * 2$



Scanned by TapScanner

DFA

(4) Designing DFA is difficult as compared to NDFA

(5) Digital computers are completely deterministic ie their state at any time is uniquely predictable from input and initial state

(II) Compare Moore and Mealy Machines

Moore

(1) Output depends only on present state and is independent of current input

(2) If input string is of length n, the output string is of length n+1

(3) Output function λ is defined as:- Mapping Q into Δ giving output associated with each state

$\lambda : Q \rightarrow \Delta$

NDFA

without consuming input symbol

(4) NFA is easier to design as compared to DFA

(5) Not associated with real computers

Mealy

(1) In mealy machine, output depends on the present state & present input

(2) If input string is of length n, output string is of same length n.

(3) $\lambda :$

$Q \times \Sigma \rightarrow \Delta$

The O/P function depends on both state & input

Scanned by TapScanner

Right Most Derivation

(9)

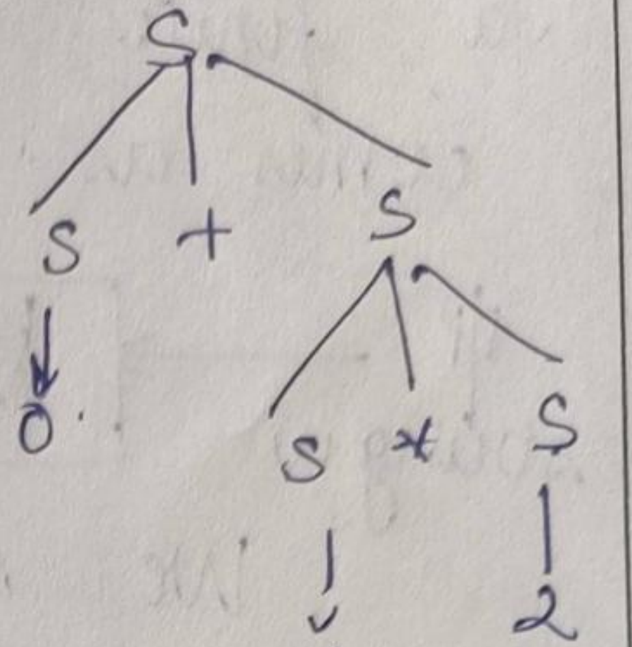
$S \rightarrow S+S$

$S \rightarrow S+S*S$

$S \rightarrow S+S*2$

$S \rightarrow S+1*2$

$S \rightarrow 0+1*2$



11) Halting Problem of Turing Machine

Ans Halting Problem is undecidability.

It is not a problem. It just ask question" is it possible to tell whether a given machine will halt for some given input.

eg Input A Turing Machine and input string w.

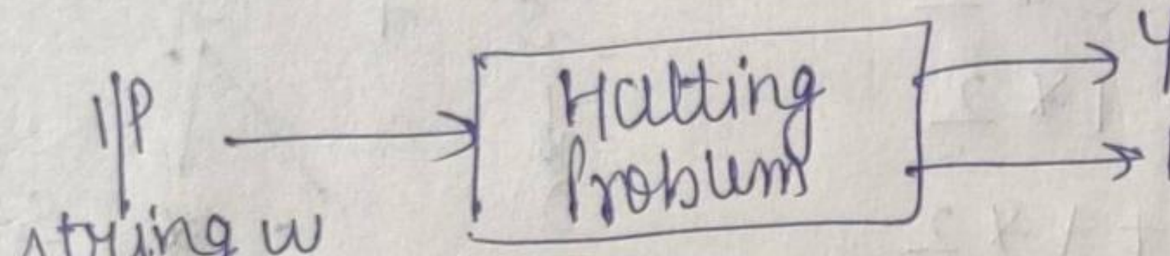
Problem computing Does the Turing Machine finish steps of string w in finite no of (Answer must be Y or N)

Proof Assume Turing Machine exist to solve the problem and then we will show it is contradicting itself.

as a Halting Machine that produce 'Yes' or 'No' in finite amount of time we will call this Turing

Scanned by TapScanner

If the halting machine finishes a finite amount of time, the o/p comes as 'Yes' otherwise 'No'



We will design an inverted halting machine

If H returns Yes, then loop forever.
If H returns No, then Halt

Q.9 Chomsky Normal Form

A context free grammar G is in Chomsky Normal form if every production is of form $A \to BC$ or $A \to a$ & $S \to \Lambda$ is in G

ie restrictions on length of right hand side and type of symbols in R.H.S. of productions.

eg $S \to AB \mid \Lambda$
$A \to a$
$B \to b$
→ (CNF)

Greibach Normal Form

In GNF there is restriction on position in which terminals variables can appear on right hand side of production rule

Scanned by TapScanner

Every production in GNF must start with single terminal followed by variables

$$A \to a\alpha$$

where $\alpha \in V_N^*$

$a \in \Sigma$.

Eg $S \to aAB / \wedge$

$A \to bC$

$B \to b$

$C \to c$

is in GNF

Convert this into GNF

$S \to AB$

$A \to BS / b$

$B \to SA / a$

Soln Rename variables as $A_1, A_2 \& A_3$

$A_1 \to S$

$A_2 \to A$

$A_3 \to B$

such that

$A_1 \to A_2 A_3$ —(i)

$A_2 \to A_3 A_1 / b$ —(ii)

$A_3 \to A_1 A_2 / a$ —(iii)

Step2 (i) The production $A_1 \to A_2 A_3$ is in required form.

(ii) $A_2 \to A_3 A_1 / b$ are in required form.

Scanned by TapScanner

(iii) $A_3 \to a$ is in required form.

$A_3 \to A_1 A_2$ $\boxed{i < j}$

~~$A_3 \to A_2 A_3 A_2$~~, Applying Lemma-

$A_3 \to A_3 A_1 A_3 A_2 \mid bA_3A_2$ — (iv)

Step 3. Left Recursion Remove

$A_3 \to A_3 A_1 A_3 A_2 \mid bA_3A_2 \mid a$

$\boxed{A_3 \to a \mid bA_3A_2 \mid aZ \mid bA_3A_2Z}$ GNF

$\boxed{Z \to A_1 A_3 A_2 \mid A_1 A_3 A_2 Z \mid Z}$

Step 4 (i) $A3 \to a \mid bA_3A_2 \mid aZ \mid bA_3A_2Z$

(ii) $A_2 \to b$

$A_2 \to b \mid aA_1 \mid bA_3A_2A_1 \mid aZA_1 \mid bA_3A_2ZA_1$

(iii) $A_1 \to A_2A_3$

$A_1 \to bA3 \mid aA_1A_3 \mid bA_3A_2A_1A_3 \mid aZA_1A_3 \mid$
$bA_3A_2ZA_1A_3$

$Z \to A_1A_3A_2 \mid A_1A_3A_2Z$

$Z \to bA_3A_3A_2 \mid bA_3A_3A_2Z$

$Z \to aA_1A_3A_3A_2 \mid aA_1A_3A_3A_2Z$

$Z \to bA_3A_2A_1A_3A_2 \mid bA_3A_2A_1A_3A_2Z$

$Z \to aZA_1A_3A_3A_2 \mid aZA_1A_3A_3A_2Z$

$Z \to bA_3A_2ZA_1A_3A_3A_2 \mid bA_3A_2ZA_1A_3A_3A_2Z$

Scanned by TapScanner