2(a)

Grammar:- Grammar is a set of rules that generates patterns of strings.

The formal definition of grammar consists of 4-tuples (V, Σ, R, S) where,

1. $V \rightarrow$ Set of variables
2. $\Sigma \rightarrow$ finite set of terminals
3. $R \rightarrow$ Set of rules
4. $S \rightarrow S \in V$ is the start variable.

Given that,

$L(G) = \{a^m b^n \mid m \geq 0 \text{ and } n > 0\}$

$= \{b, ab, aab, abb, \ldots\}$

Hence, the grammar that produces the given language:

$S \rightarrow AB$

$A \rightarrow aA \mid \varepsilon$

$B \rightarrow bB \mid b$

2) (b) According to Noam Chomsky, mention the types of Grammar with examples.

Ans: Noam Chomsky is known for his influential work on linguistics and grammar, which has also had an impact on the theory of computing. He proposed several types of grammers, including:

1) Type 0 Grammar (Recursively Enumerable Grammar):

These are the most general and include all possible formal languages.

Examples: Turing Machines, Post Systems.

2) Type 1 Grammer (Context-Sensitive Grammers):

Grammars where rules are sensitive to the

context in which they are applied.

Examples: Natural language grammers, certain programming languages.

3] Type 2 Grammar (Context-Free Grammar):
Rules are not context-sensitive, making them suitable for describing programming languages.
Examples: Backus–Naur Form (BNF) for programming languages.

4] Type 3 Grammer (Regular Grammar):
Simplest type with rules that can be expressed as regular expressions.

Examples: Regular expressions, Finite Automata.

These grammatical types are often used in the theory of formal languages and automata, which is fundamental in the field of computer scinence and the study of computability. Chomsky's hierarchy provides a framework for understanding the complexity of different language classes and their relation to computing machines.

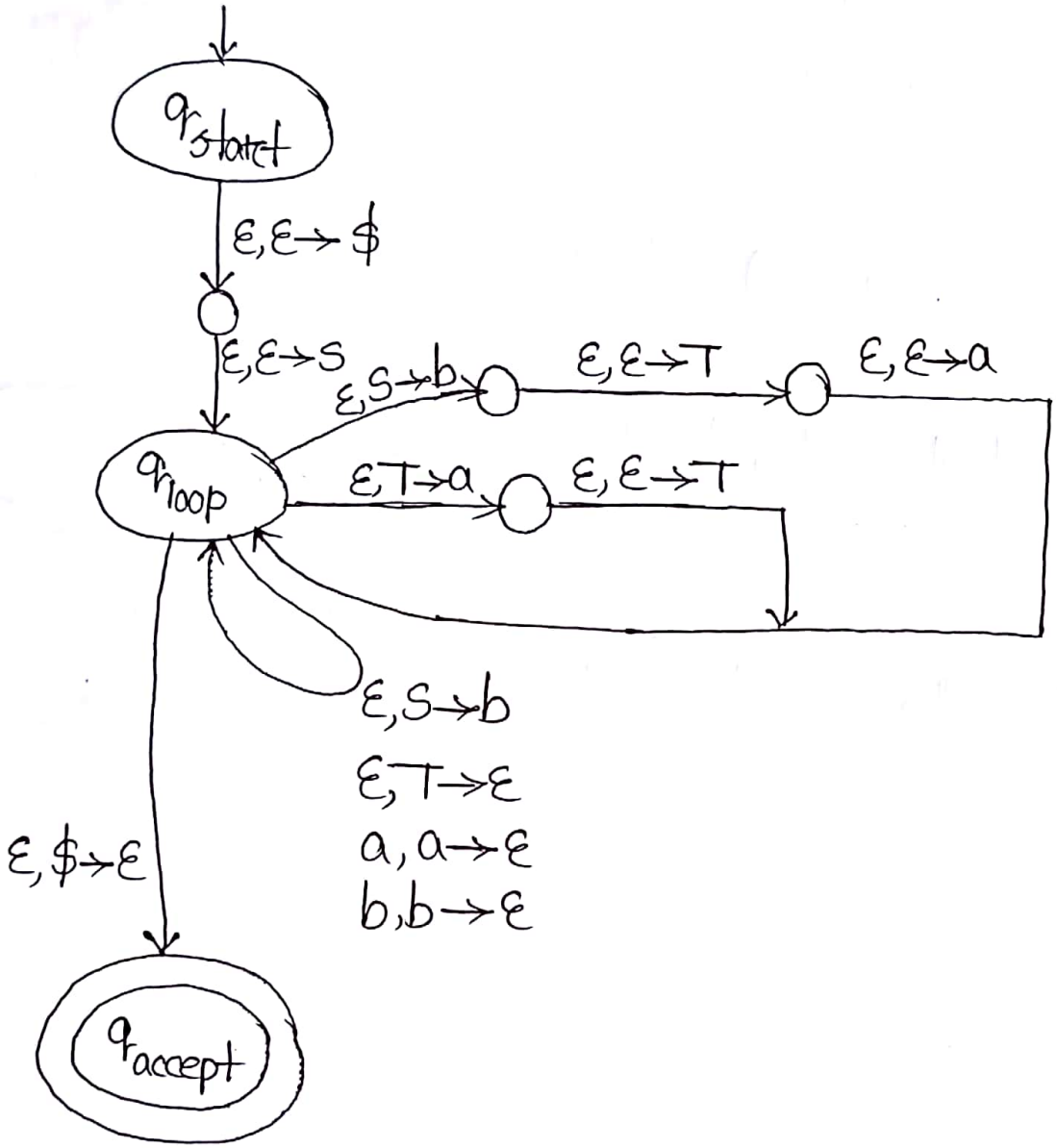
3(a)

The given grammar :

$S \rightarrow aTb \mid b$

$T \rightarrow Ta \mid \varepsilon$

The pushdown automata for the given grammar :



Ans to Q. no. – 3(b)

i. $A = \{ a^n b^n c^n \mid n \geq 0 \}$.

Let us assume that A be a CFL.

Now, let us consider the pumping length, $p = 4$ & $s = a^4 b^4 c^4 \in A$.

Case 1: v & y each contain only one type of symbol.

$s = \underbrace{a}_{u} \underbrace{a\ a}_{v} \underbrace{a\ bbbb\ c}_{x} \underbrace{c}_{y} \underbrace{cc}_{z}$.

For $i = 2$,

$uv^2xy^2z = aaaaaa\ bbbb\ ccccc$

$= a^6 b^4 c^5 \notin A$.

Thus, condition 1 is violated.

Case 2: Either v or y has more than one kind of symbols.

$s = \underbrace{aa}_{u} \underbrace{aabb}_{v} \underbrace{bb}_{x} \underbrace{c}_{y} \underbrace{ccc}_{z}$.

For $i = 2$.

$uv^2xy^2z = aaaa\ bb\ aa\ bb\ bb\ cc\ ccc$

$= a^6 b^6 c^5 \notin A$.

because it violates the pattern $a^n b^n c^n$ to be followed.

Hence, the given language is not content free.

ii $B = \{ a^i b^j c^k \mid 0 \le i \le j \le k \}$.

Let us assume that B be a CFL.

Now let us consider the pumping length $P = 4$.

So, $S = a^4 b^4 c^4 \in B$.

case 1 : v & y each contain only one type of symbol.

$S = \underbrace{aa}_{u} \underbrace{aa}_{v} \underbrace{bbbb}_{x} \underbrace{cccc}_{z}$.

for $i = 2$, $uv^i xy^i z = aaaaaa\ bbbbb\ cccc$

$= a^6 b^5 c^4$.

Since number of a's is greater than that of b's & number of b's is greater than that of c's.

$a^6 b^5 c^4 \notin B$.

case 2 : Either v or y has more than one kind of symbols.

$S = \underbrace{aaa}_{u} \underbrace{ab}_{v} \underbrace{bb}_{x} \underbrace{b}_{y} \underbrace{cccc}_{z}$.

for, $i = 2$, $uv^i xy^i z = aaaaab\ ab\ bbbb\ cccc$

$= a^6 b^6 c^4$.

which doesn't contain the symbol in the correct order.

Thus, we have shown that S can't be pumped in violation of the pumping lemma & that B is not context free.

4(a) Describe the Church Turing thesis?

=> "The assumption that the intuitive notion of computable functions can be identified with partial recursive functions."

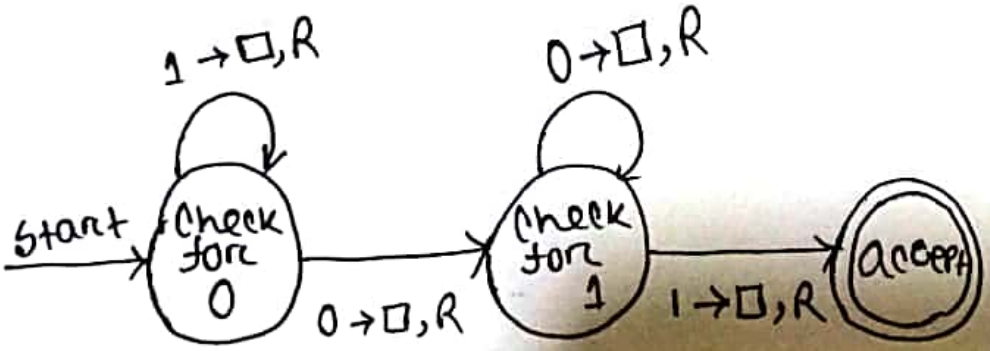
Or, we can say -

"Every computation that can be carried out in that real world can be effectively performed by a Turing machine".

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

Draw the state transition diagram for a Turing machine whose, $L = \{w \in \Sigma^* \mid w \text{ contains } 01 \text{ as substring}\}$

=> This Turing machine mimics the DFA for the same language, moving the tape head one step to the right at each step. In the following, assume that missing transitions implicitly cause TM to reject.



(4b)

Given that,

$S \to ABAC$

$A \to aA \mid \varepsilon$

$B \to bB \mid \varepsilon$

$C \to c$

Removing $A \to \varepsilon$,

$S \to ABAC \mid BAC \mid ABC \mid BC$

$A \to aA \mid a$

$B \to bB \mid \varepsilon$

$C \to C$

Removing $B \to \varepsilon$,

$S \to ABAC \mid BAC \mid ABC \mid BC \mid AAC \mid AAC \mid AC \mid C$

$A \to aA \mid a$

$B \to bB \mid b$

$C \to C$

(6a)

Differentiate among finite state machine, pushdown automata and turing machine.

Finite State Machine	Pushdown Automata	Turing Machine
(1) State Machine have a limited memory	(1) Pushdown automata have a stack	(1) Turing Machine have an infinite tape as their memory.
(2) They are less powerful than push automata and Turing machine.	(2) More powerful than state machine but less powerful than Turing Machine.	(2) Most powerful than state machine and pushdown automata
(3) Recognize regular language	(3) Recognize context-free language	(3) Recognize both context-free and regular language
(4) Commonly used in scenarios where the system can be in a finite number of well defined states	(4) used in the parsing of context-free grammers.	(4) Theoretical Concept used in the study of computability and complexity

The rules for a Turing machine typically defined as a set of quintuples. there,

$$(q_i, s_j) \rightarrow (q_k, s_l, D_m)$$

① Current state (q_i) : The state that the machine is currently in

② Current symbol being read (s_j) : The symbol that the machine reads from the current tape position.

③ Write symbol (s_l) : The symbol that the machine writes to the current tape position.

④ Move Direction (D_m) : The direction in which the tape head moves after writting the symbol.

⑤ Next step (q_k) : The state that the machine transition to after the current step

Definition of Ambiguous Grammar :-

An ambiguous grammer is a type of context-free grammar in computer science that can produce more than one leftmost derivation or parse tree for a given string.

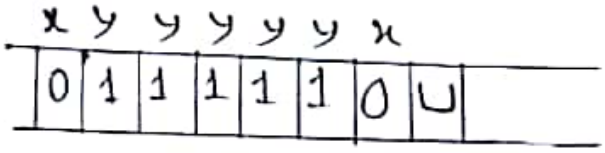
A turing machine that recognizes the language,

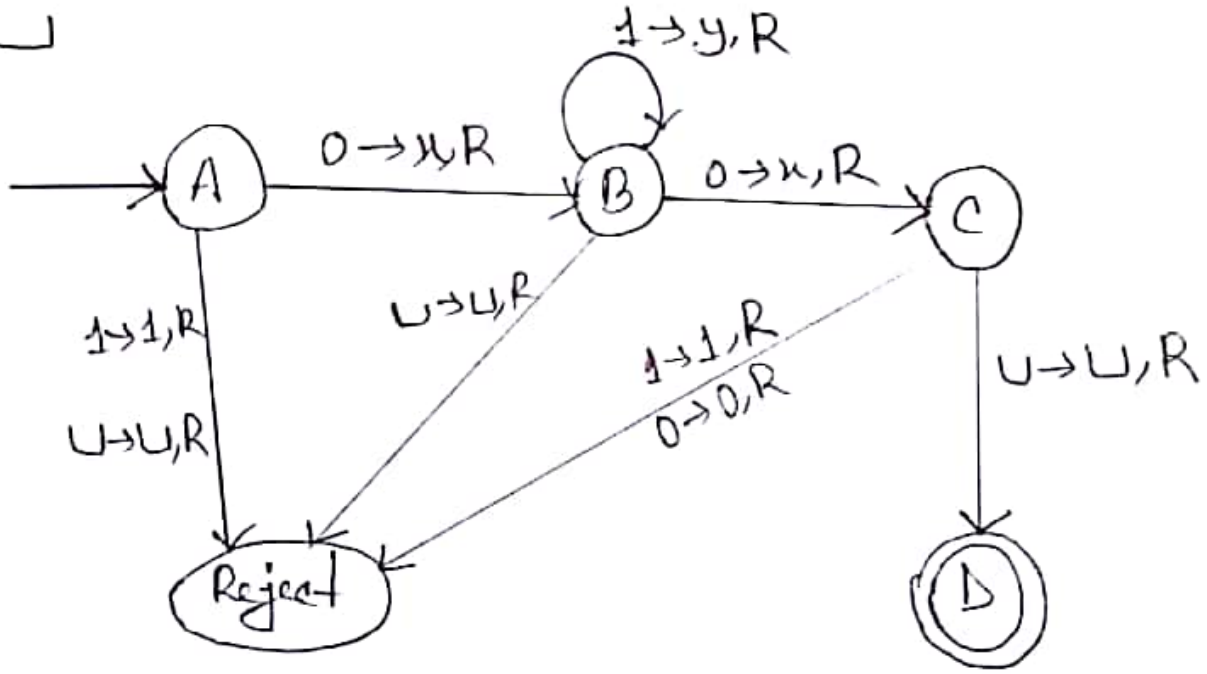
$L = 01^*0$ —

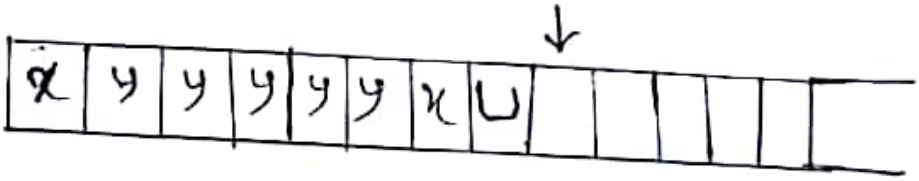
$\Sigma = \{0, 1\}$

$L = 01^*0$

$b = \sqcup$







Ans. to the Q. no - 02 (a)

Every non-deterministic Turing Machine has an equivalent deterministic Turing Machine.

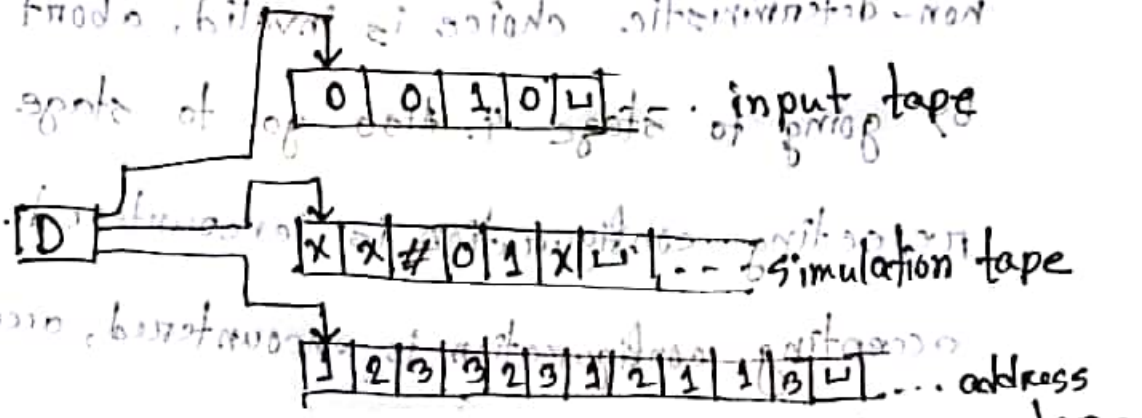


fig: Deterministic TM "D" simulating non-deterministic TM "N"

1. Initially tape 1 contains the input w, and tapes 2 & 3 are empty.

2. Copy tape 1 to tape 2 and initialize the string on tape 3 to be ε.

3. Use tape 2 to simulate N with input w on one branch of its non-deterministic computation. Before each step of N, consult the next symbol on tape

3 to determine which choice to make among those allowed by N's transition function. If no more symbols remain on tape 3, or if this non-deterministic choice is invalid, abort this branch by going to stage 4. Also go to stage 4 if a rejecting configuration is encountered. If an accepting configuration is encountered, accept the input.

4. Replace the string on tape 3 with the next string in the string ordering. Simulate the next branch of N's computation by going to stage 2.