UNIT-4

NORMAL FORMS AND TURING MACHINES

UNIT - IV

NORMAL FORMS FOR CONTEXT - FREE GRAMMARS

Minimization of context free grammar (CFG):

The CFG may contain some extra non terminals which are useless. This causes the length of grammar to be increased. Hence we delete some useless symbols which is then called as simplification of grammar.

The properties of reduced grammar are given below:

1) Each terminal and non-terminal of grammar appears in the derivation of some word in the language.

2) There should not be any productions as $X \rightarrow Y$ where x and y are non-terminals.

3) If ϵ is not in the language 'L' then there need not be the production $x \rightarrow \epsilon$.

Elimination of useless symbols (or) unreachable symbols:

→ A symbol which is not involved in any string generation by the grammar is called as useless symbol.

(or)

→ A symbol which is not reachable from the starting symbol is called as unreachable symbols.

$S \xRightarrow{*} \alpha p \beta$ } Here p is said to be useful symbol.

$\alpha p \beta \xRightarrow{*} w$

Ex: ① $A \longrightarrow aB \mid b$

$B \longrightarrow bA \mid b$

$C \longrightarrow ec \mid f$

$D \longrightarrow d \mid dE$

$E \longrightarrow aE$

Sol^n: Here, the production 'c' is useless as it is unreachable from starting symbol 'A', and the production $E \longrightarrow aE$ cannot be terminated. So, it is also useless. Hence we can delete the useless symbols.

② $G_1 = (V, T, P, S)$ where $V = \{S, T, X\}$ $T = \{0, 1\}$

$S \longrightarrow 0T \mid 1T \mid X \mid 0 \mid 1$

$T \longrightarrow 00$

Sol: Here X is not having any further derivations Hence we delete X and associated if exists any

$\therefore S \longrightarrow 0T \mid 1T \mid X \mid 0 \mid 1 \quad S \longrightarrow 0T \mid 1T \mid 0 \mid 1$

$\Longrightarrow T \longrightarrow 00$

$T \longrightarrow 00$

③ Consider the CFG $G = (V, T, P, S)$ where $V = \{S, A, B\}$ $T = \{0, 1\}$, $P = \{$

$S \longrightarrow A11B \mid 11A$

$S \longrightarrow 1B \mid 11$

$A \longrightarrow 0$

$B \longrightarrow BB \}$

Soln Here B doesnot derive any terminal symbols so, we delete B and associated strings i.e., $B \to BB$ and $S \to A11B$

The resultant will be $P = \{ S \to 11A,\ S \to 11,\ A \to 0 \} \Rightarrow \{ S \to 11A/11,\ A \to 0 \}$

Elimination of ϵ productions from grammar:

If the grammar contains 'ϵ' which means it has no value, we remove it from the production rules.

→ 'ϵ' productions for a non-terminal is replaced by introducing a new production for the given N.T without ϵ Non Terminal.

[Note: only one 'ϵ' can be eliminated at a time]

Ex Remove 'ϵ' production from the following CFG, by preserving meaning of it.

$S \to aA \mid bBb \mid ab$

$A \to aA \mid \epsilon$

$B \to bBa \mid \epsilon$

Soln: $S \to aA \mid bBb \mid ab$

$A \to aA \mid \epsilon$

$B \to bBa \mid \epsilon$

$\Longrightarrow$ Removing 'ϵ' from A

$S \to aA \mid bBb \mid ab \mid a$

$A \to aA \mid a$

$B \to bBa \mid \epsilon$

⇓ Removing 'ϵ' from B

$S \to aA \mid bBb \mid ab \mid a \mid bb$

$A \to aA \mid a$

$B \to bBa \mid ba$

Ex (2) $A \longrightarrow 0B1 \mid 1B1$

$B \longrightarrow 0B \mid 1B \mid \epsilon$

Solution:

$A \longrightarrow 0B1 \mid 1B1$

$B \longrightarrow 0B \mid 1B \mid 0 \mid 1$

Ex.3 $S \longrightarrow aSa$

$S \longrightarrow bSb$

$S \longrightarrow \epsilon$

Sol: $S \longrightarrow aSa \mid bSb \mid aa \mid bb$

Elimination of unit productions:

A production is said to be unit if a non terminal derives another non-terminal alone. Ex: $X \longrightarrow Y$.

→ A unit production can be eliminated by replacing R.H.S non-terminal with all its productions.

Ex

$E \longrightarrow E+T \mid T$

$T \longrightarrow T*F \mid F$

$F \longrightarrow (E) \mid id$

Solution:

$E \longrightarrow E+T$

$E \longrightarrow T$

$T \longrightarrow T*F$

$T \longrightarrow F$

$F \longrightarrow (E)$

$F \longrightarrow id$

Here $E \longrightarrow T$ and $T \longrightarrow F$ are unit productions

$\Longrightarrow$

$E \longrightarrow E+T$

$E \longrightarrow T$

$T \longrightarrow T*F$

$T \longrightarrow (E) \mid id$

$F \longrightarrow (E)$

$F \longrightarrow id$

Here $T \longrightarrow F$ is eliminated by replacing F productions

$\Longrightarrow$

$E \longrightarrow E+T$

$E \longrightarrow T*F \mid (E) \mid id$

$T \longrightarrow T*F$

$T \longrightarrow (E) \mid id$

$F \longrightarrow (E)$

$F \longrightarrow id$

Here $E \longrightarrow T$ is eliminated by replacing T productions

Normal Forms:

At the right hand of production rules we have different combinations of terminal & non terminal. we need to arrange them in a specific format ie, fixed number of terminals. This process is known as "normalization"

→ These are 2 types of Normal Forms They are :-

(1) Chomsky Normal Form (CNF)

(2) Greibach Normal Form (GNF)

1) Chomsky Normal Form:

A Grammar is said to be in CNF if every production of the form is $A \longrightarrow XY$ or $A \longrightarrow a$ ie, In right hand side either two non-terminals or only one terminal is allowed.

Non-terminal ⟶ Nonterminal . Non terminal
Non-terminal ⟶ Terminal .

Conversion of Context Free grammar to chomsky's Normal Form:

Before converting the CFG to CNF we reduce the gramma by removing useless symbols and ϵ productions and removing unit productions.

Step 1: Remove ϵ productions.

Step 2: Remove unit productions.

Step 3 Remove useless symbols

Step4: Replace terminal symbol with new variable.

Step5: Break the symbols having length more than 2.

Ex: Convert the following CFG to CNF

$S \longrightarrow AaBb \mid abA$

$A \longrightarrow ab \mid Bb$

$B \longrightarrow bA \mid b$

Solution: Given grammar is

$S \longrightarrow AaBb$

$S \longrightarrow abA$

$A \longrightarrow ab$

$A \longrightarrow Bb$

$B \longrightarrow bA$

$B \longrightarrow b$

The given CFG does not contain any useless symbols, ϵ & unit productions. Now replace $X \to a$ & $Y \to b$

$S \longrightarrow AXBY$

$S \longrightarrow XYA$

$A \longrightarrow XY$

$A \longrightarrow BY$

$B \longrightarrow YA$

$B \longrightarrow b$

$\Longrightarrow$

Replace

$AX \longrightarrow P$

$BY \longrightarrow Q$

$XY \longrightarrow R$

$S \longrightarrow PQ$

$P \longrightarrow AX$

$Q \longrightarrow BY$

$S \longrightarrow RA$

$R \longrightarrow XY$

$A \longrightarrow XY$

$A \longrightarrow BY$

$B \longrightarrow YA$

$B \longrightarrow b$

$X \longrightarrow a$

$Y \longrightarrow b$

Ex:2 Convert the following CFG to CNF

$S \rightarrow ABA$

$A \rightarrow aA \mid \epsilon$

$B \rightarrow bB \mid \epsilon$

Solution The given CFG contains 'ϵ' productions.

So, remove 'ϵ' productions from the grammar

→ Eliminating 'ϵ' from A

$S \rightarrow ABA \mid BA \mid AB \mid B$

$A \rightarrow aA \mid a$

$B \rightarrow bB \mid \epsilon$

→ Eliminating 'ϵ' from B

$S \rightarrow ABA \mid BA \mid AB \mid B \mid AA \mid A$

$A \rightarrow aA \mid a$

$B \rightarrow bB \mid b$

Now, the grammar contain unit productions. So eliminate them by replacing R.H.S N.T with all the productions

$S \rightarrow ABA \mid BA \mid AB \mid aA \mid a \mid b \mid bB$

$A \rightarrow aA \mid a$

$B \rightarrow bB \mid b$

Now convert it into CNF

replace $X \rightarrow AB$

$A_1 \rightarrow a$

$B_1 \rightarrow b$

$S \longrightarrow XA$

$S \longrightarrow BA$

$S \longrightarrow AB$

$S \longrightarrow A_1A$

$S \longrightarrow a$

$S \longrightarrow B_1B$

$S \longrightarrow b$

$A \longrightarrow A_1A$

$A \longrightarrow a$

$B \longrightarrow B_1B$

$B \longrightarrow b$

$X \longrightarrow AB$

$A_1 \longrightarrow a$

$B_1 \longrightarrow b$

Greibech Normal Form

A Grammar is said to be in GNF if every production of the form $A \longrightarrow a\alpha$ ie, a terminal followed by any number of non-terminals.

steps to convert CFG to GNF:

step 1: check if the given CFG has any unit productions or Null productions and useless symbols, remove if there are any.

step 2: check whether the CFG is already in CNF and convert it to CNF if it is not.

step 3: change the names of the non-terminals symbols in to some A_j in ascending order of i

step 4: After the rules so that the non-terminals are in ascending order, such that if the production is of the form $A_i \longrightarrow A_jX$ then $i < j$ and should never be $i \geq j$ [left recursive grammar: A production of the form $A \rightarrow A\alpha$ is called left recursive as the left hand side variable appears as first symbol on right hand side]

step 5: Remove left recursion, introduce a new variable to remove the left recursion.

Example: Convert the following CFG to GNF

$S \longrightarrow CA \mid BB$

$B \longrightarrow b \mid SB$

$C \longrightarrow b$

$A \longrightarrow a$

Solution: Changing the non terminal symbols in Ascending order

Replace S with A_1
C with A_2
A with A_3
B with A_4

we get ⟹

$A_1 \longrightarrow A_2A_3 \mid A_4A_4$
$A_4 \longrightarrow b \mid A_1A_4$
$A_2 \longrightarrow b$
$A_3 \longrightarrow a$

Here $A_4 \longrightarrow b \mid A_1A_4$ is violating the rule-4 $[A_4 > A_1]$ i.e., $i > j$. so replace A_1 with its productions.

$A_1 \longrightarrow A_2A_3 \mid A_4A_4$
$A_4 \longrightarrow b \mid A_2A_3A_4 \mid A_4A_4A_4$
$A_2 \longrightarrow b$
$A_3 \longrightarrow a$

→ Again A_4 production is violating the rule. so replace it with A_2 productions.

$A_4 \longrightarrow b \mid bA_3A_4 \mid \underbrace{A_4A_4A_4}$ left recursion.

→ Introduce a new variable.

$Z \longrightarrow A_4A_4Z \mid A_4A_4$ then
$A_4 \longrightarrow b \mid bA_3A_4 \mid bZ \mid bA_3A_4Z$

The gramma is

$A_1 \longrightarrow A_2A_3 \mid A_4A_4$
$A_4 \longrightarrow b \mid bA_3A_4 \mid bZ \mid bA_3A_4Z$
$Z \longrightarrow A_4A_4 \mid A_4A_4Z$
$A_2 \longrightarrow b$
$A_3 \longrightarrow a$.

→ Here A_1 production is not in the GNF. So replace with their respective productions.

$A_1 \longrightarrow bA_3 \mid bA_4 \mid bA_3A_4A_4 \mid bZA_4 \mid bA_3A_4ZA_4$

$A_4 \longrightarrow b \mid bA_3A_4 \mid bZ \mid bA_3A_4Z$

$Z \longrightarrow bA_4 \mid bA_3A_4A_4 \mid bZA_4 \mid bA_3A_4ZA_4 \mid bA_4Z \mid bA_3A_4A_4Z \mid$

$bZA_4Z \mid bA_3A_4ZA_4Z$

$A_2 \longrightarrow b$

$A_3 \longrightarrow a$

Ex:② Conversion of CFG to GNF

$S \longrightarrow AA \mid a$

$A \longrightarrow SS \mid b$

Solution Given grammar is already in simplified form. There are no null productions, no unit productions & no useless symbols in the grammar.

step1: Now we proceed for renaming of variables. Variables S & A are renamed as A_1 & A_2.

So productions are.

$A_1 \longrightarrow A_2A_2 \mid a$

$A_2 \longrightarrow A_1A_1 \mid b$.

step2: Now every production should be of the form

$A_i \longrightarrow A_j\alpha \quad (i \not< j)$

'A_2' production does not satisfy the rule, hence substitute A_1 with its production.

Therefore

$A_2 \longrightarrow A_2A_2A_1 \mid aA_1 \mid b$

The resulting productions are

$A_1 \longrightarrow A_2A_2 \mid a$

$A_2 \longrightarrow A_2A_2A_1 \mid aA_1 \mid b$

step3: Removing left recursion

A_2 production of the form $A_2 \longrightarrow A_2A_2A_1 \mid aA_1 \mid b$ contains left recursion. So remove by introducing a new non terminal B

Therefore

$A_2 \longrightarrow A_2A_2A_1 \mid aA_1 \mid b$

can be written as

$A_2 \longrightarrow aA_1B \mid bB \mid aA_1 \mid b$

$B \longrightarrow A_2A_1B \mid A_2A_1$

[∵ If left recursive grammar is
$A \longrightarrow A\alpha \mid b$
can be written as
$A \longrightarrow b \mid bB$
$B \longrightarrow \alpha \mid \alpha B$]

Resulting productions are

$A_1 \longrightarrow A_2A_2 \mid a$

$A_2 \longrightarrow aA_1B \mid bB \mid aA_1 \mid b$

$B \longrightarrow A_2AB \mid A_2A_1$

Step 4: A_2 productions are in GNF. A_1 & B productions need to be converted to GNF

$A_1 \longrightarrow A_2A_2 \mid a$ replaced A_2 with its production

$A_1 \longrightarrow aA_1BA_2 \mid bBA_2 \mid aA_1A_2 \mid bA_2 \mid a$

Now $B \longrightarrow A_2A_1B \mid A_2A_1$ replace A_2 with its producing

$B \longrightarrow aA_1BA_1B \mid bBA_1B \mid aA_1A_1B \mid bA_1B \mid aA_1BA_1 \mid bBA_1 \mid aA_1A_1 \mid bA_1$

Final grammar in GNF is

$A_1 \longrightarrow aA_1BA_2 \mid bBA_2 \mid aA_1A_2 \mid bA_2 \mid a$

$A_2 \longrightarrow aA_1B \mid bB \mid aA_1 \mid b$

$B \longrightarrow aA_1BA_1B \mid bBA_1B \mid aA_1A_1B \mid bA_1B \mid aA_1BA_1 \mid bBA_1 \mid aA_1A_1 \mid bA_1$.

Pumping lemma for context free languages:

→ To prove certain languages are not context free

- let L be a CFL.
- let n be a constant.
- Any string z in L, $|z| \geq n$
- split $z = uvwxy$ such that
  - (i) $|vwx| \leq n$
  - (ii) $vx \neq \epsilon$ or $|vx| \geq 1$
  - (iii) for all $i \geq 0$, $uv^iwx^iy \in L$

Ex: Show that $L = \{a^n b^n c^n \mid n \geq 1\}$ is not a CFL.

Proof: - let L be a CFL
- n be a constant
- let $z = a^n b^n c^n$ $|z| \geq n$
- split L, hence we get

$z = uvwxy$

$u = a^n \quad vwx = b^n \quad y = c^n$

(i) $|vwx| \leq n$

(ii) $vwx = b^n$

$|vx| = b^{n-m}$

for all $i \geq 0 \quad uv^iwx^iy$

$uv^iwx^iy = uv^{i-1}wxx^{i-1}y$

$= uvw(vx)^{i-1}xy$

$= uvw(b^{n-m})^{i-1}xy$

$= a^n b^n (b^{n-m})^{i-1} c^n$

$= a^n b^{ni-mi-n+m} c^n$

$= a^n b^m c^n$

not in CFL

∴ Hence language L is not CFL.

Eg: Show that $L = \{a^n b^n c^n \mid n \geq 0\}$ is not context free.

Solution

we divide string in to parts $uvwxy$

(or) (i) let $n = 4 \quad S = a^4 b^4 c^4$

v & x contain one type of symbol.

$\underbrace{aaaa}_{u}\underbrace{bbbb}_{v}\underbrace{cccc}_{\omega}\underbrace{}_{x}\underbrace{}_{y}$

let $k = 2 \quad uv^2wx^2y$

$\Rightarrow a(aa)^2 abbbbcc^2 cc$

$\Rightarrow aaaaaa\,bbbb\,cccccc$

$\Rightarrow a^6 b^4 c^5$

The resultant string is not satisfying the condition

$\therefore a^6 b^4 c^5 \notin L$

Given language is not CFL.

Ex 2: Prove that the following language is CFL or not. $L = \{a^i b^i c^i \mid i \geq 0\}$.

Solution: $L = \{\epsilon, abc, aabbcc, aaabbbccc, \ldots\}$

$n = 4$

$z = \underbrace{aa}_{u}\underbrace{bb}_{v}\underbrace{cc}_{\omega}\underbrace{}_{x}\underbrace{}_{y} \qquad |z| = 4 \geq n$

$|aabb| \leq n$

$$z = \underset{u}{a}\,\underset{v}{a}\;\underset{w}{bb}\;\underset{x}{c}\,\underset{y}{c}$$

For $i=1 \Rightarrow z = uv^1wx^1y = aabbcc \in L$

For $i=2 \Rightarrow z = uv^2wx^2y$

$= uvvwxxy$

$= aaabbccc \notin L$

∴ Hence given language is not CFL.

Closure properties of context free languages:

→ Regular languages are closed under every properties whereas CFL are not closed under CID (Complement Intersection, Difference).

→ CFL are closed under union, concatenation, Kleen closure, Reverse, Homomorphism, Inverse homomorphism, Substitution.

union: CFL are said to be closed under union if L_1 & L_2 are two C.F.L then $L_1 \cup L_2$ must be a CFL.

Ex: $L_1 = \{ a^n b^n c^m \mid m>=0 \ \& \ n>=0 \}$ and

$L_2 = \{ a^n b^m c^m \mid n>=0 \ \& \ m>=0 \}$

$L_3 = L_1 \cup L_2 = \{ a^n b^n c^m \cup a^n b^m c^m \mid n>=0, m>=0 \}$ is CFL.

∴ CFL is closed under union.

Concatenation: CFL are said to be closed under concatenation if L_1 & L_2 are two CFL's then $L_1 \cdot L_2$ must be a CFL.

Ex: $L_1 = \{ a^n b^n \mid n \geq 0 \}$ and $L_2 = \{ c^m d^m \mid m \geq 0 \}$

$L_3 = L_1 \cdot L_2 = \{ a^n b^n c^m d^m \mid m \geq 0 \text{ \& } n \geq 0 \}$ is also CFL.

∴ CFL is closed under concatenation.

Kleen closure: If L_1 is CFL, its kleen closure L_1^* is also CFL.

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

$L_1^* = \{ a^n b^n \mid n \geq 0 \}^*$ is also kleen closure.

Intersection: If L_1 & L_2 are CFL then $L_1 \cap L_2$ need not to be CFL.

$L_1 = \{ a^n b^n c^m \mid n \geq 0 \text{ \& } m \geq 0 \}$

$L_2 = \{ a^m b^n c^n \mid n \geq 0 \text{ \& } m \geq 0 \}$

$L_3 = L_1 \cap L_2 = \{ a^n b^n c^n \mid n >= 0 \}$ need not be CFL.

complement: complementation of CFL need not to be in CFL.

Ex: $\Sigma^* - L_1$

Hence, CFL is not closed under intersection and complementation.

Deterministic context free language: DCFL are closed under complementation & inverse Homomorphism.

Decision properties of CFL's:

Membership:

→ unlike finite automatas, we can't just run the string through the machine and see where it goes since PDA's are non-deterministic.

→ Must consider all possible paths.

→ Instead, start with your grammar in CNF.

→ The proof of pumping lemma states that the longest derivation path of a string of size n will be $2n-1$.

→ systematically, generate all the strings, if we derive the string is in the language.

Emptyness:

→ Remove useless symbols & productions.

→ If S is useless, then $L(G)$ is empty.

Finiteness:

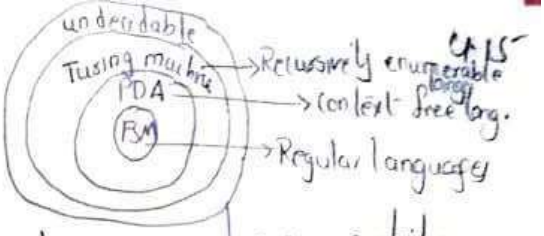
→ The language is infinite if there is a string x with length between n & 2n.

→ CFL $n = 2^{P+1}$ P is the non-terminal.

↓ no. of states.

→ If Language L is passed through membership algorithm is infinite else L is finite.

Turing machine



Introduction to Turing machine:

(1) It has an external memory which remembers arbitrary long sequence of input (where as F.A reads only current symbol).

(2) It has unlimited memory capability

(3) The model has facility by which the input at the left or right on the tape can be read easily.

(4) The machine can produce a certain output based on its input.

Formal definition of Turing machine.

The formal notation we use for Turing machine (TM) is similar to that used for finite automata. we represent TM by 7-tuple.

$$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$$

Q - Finite set of states of the finite control.

Σ - Finite set of input symbols.

Γ - Tape symbols.

δ - Transition function (or) mapping function.

$\delta(q, X)$ is defined as triple (P, Y, D) where

P - next state in Q.

Y - is symbol in Γ

D - is direction either L or R ie, left or right direction in which head move

q_0 - The start state.

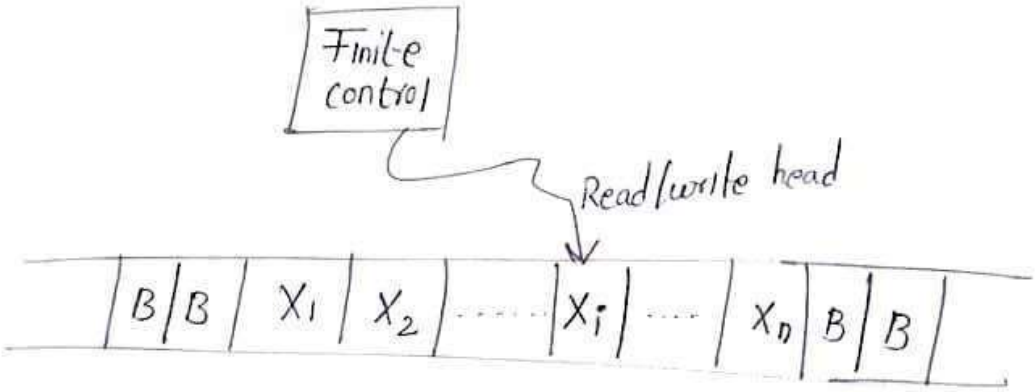
B - The blank symbol, this symbol is in T, but not in Σ.

F - set of final or accepting states.

Block diagram of TM:

→ Machine consists of a finite control, which can be in any of the finite set of states.

→ A tape divided in to squares (or) cells, each cell hold any one of the finite numbers of symbols.



Turing machine

→ The tape is infinite length on the left side as well as right side.

→ The tape is divided in to certain cells, it can hold input symbol and also blank.

→ The reading head always points on the left most cell of the tape.

→ Reading If the TM reaches the final state, the input string is accepted, otherwise rejected.

Instantaneous descriptions for Turing machines

→ An instantaneous description is used to show the acceptance of some string by automata.

→ An ID of a TM is a string of the form $\alpha_1 q \alpha_2$ where α_1 & $\alpha_2 \in \Gamma^*$ and $q \in K$.

→ We use the string $X_1 X_2 \ldots X_{i-1} q X_i X_{i+1} \ldots X_n$ to represent an ID in which

- q is the state of the Turing machine.
- The tape head is scanning the i^{th} symbol from the left
- $X_1 X_2 \ldots X_n$ is the portion of the tape between the leftmost & the rightmost nonblank.

→ we describe instantaneous description by using tanstyle notation ($\vdash$), $\vdash^*$ represents 0 or more moves.

case 1: leftmove

⇒ Suppose $\delta(q, X_i) = (p, Y, L)$ (current state, current tape symbol) ie, the next move is leftward then

$$X_1 X_2 \ldots X_{i-1} q X_i X_{i+1} \ldots X_n \vdash_M X_1 X_2 \ldots X_{i-2} p X_{i-1} Y X_{i+1} \ldots X_n$$

(∵ Here q is present state & X_i is the symbol currently scanned by tape head means we need to change q to p and X_i is replaced by Y & move is left mode

(i) if $i=1$ then M moves to the blank to the left of X_1.

so $X_1 X_2 \ldots X_n \vdash_M pBY X_2 \ldots X_n$

(i) if $i=n$ & $Y=B$ then the Symbol B written over X_n joins the infinite sequence of trailing blanks and does not appear in next ID.

then $X_1 X_2 \cdots X_{n-1} q X_n \vdash X_1 X_2 \cdots X_{n-2} P X_{n-1}$

case 2 : right move

$\delta(q, X_i) = (P, Y, R)$ the next move is rightward.

then

$X_1 X_2 \cdots X_{i-1} q X_i X_{i+1} \cdots X_n \vdash_M X_1 X_2 \cdots X_{i-1} Y P X_{i+1} \cdots X_n$

(∵ Now tape head is scanning X_i, here move is right move so it has to scan X_{i+1})

Again there are two important exceptions.

(i) If $i=n$ then $i+1$ st cell holds a blank, and that cell was not part of the previous ID, Thus we instead have $X_1 X_2 \cdots X_{n-1} q X_n \vdash_M X_1 X_2 \cdots X_{n-1} Y P B$

(ii) If $i=1$ & $Y=B$ then the symbol B written over X_1 joins the infinite sequence of leading blanks and does not appear in the next ID. Thus

$q X_1 X_2 \cdots X_n \vdash_M P X_2 \cdots X_n$

Ex: Design a turing machine for $L = \{a^n b^n / n \geq 1\}$

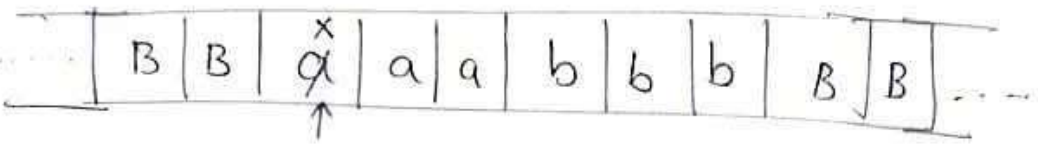
Soln: Given language $L = \{ab, aabb, aaabbb, \ldots\}$

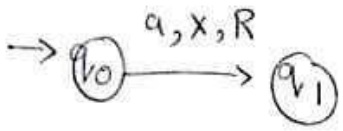
→ Here the concept is if we read 'a', then change a to 'X', then move head towards right till we find 'b' (ie. a a b b) then change b to Y.
X Y

→ Next move towards left till we get 'X', then move the head towards right applying the same concept.

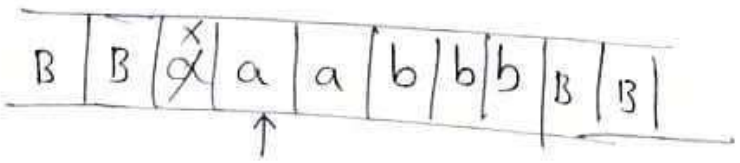
ie., a a b b
X X Y Y

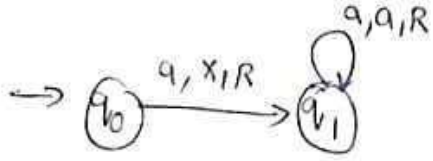
→ Let the string is



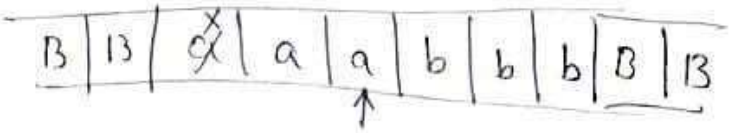


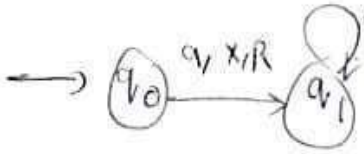
step2





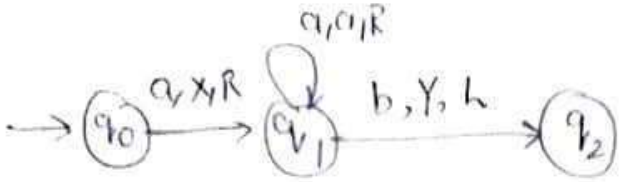
step3





Step4.

B	B	X	a	a	Y	b	b	B	B
				↑					

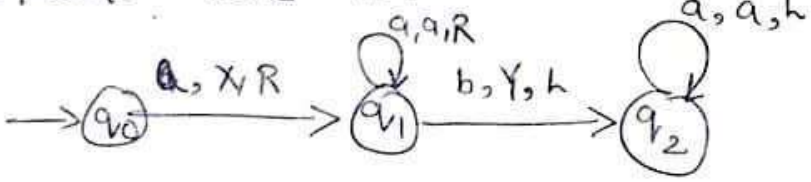


Here on reading b, then replace it with Y and move the read/write head towards left till we get X.

Step5:

B	B	X	a	a	Y	b	b	B	B
				↑					

Now on reading a, no need to change state and move read/write head left ie.



Step6:

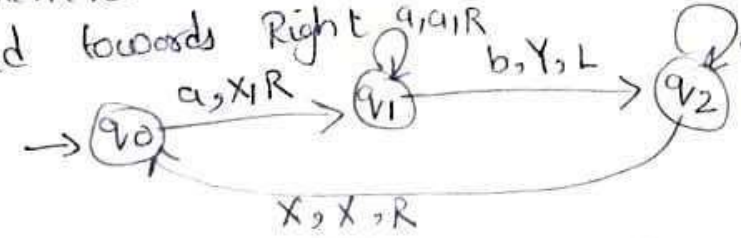
B	B	X	a	a	Y	b	b	B	B
			↑						

Now also on reading a, no need to change state, just move towards left:

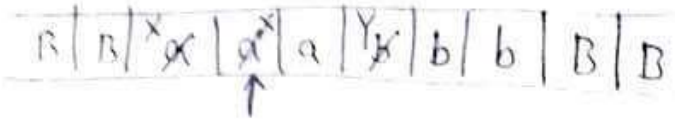
Step7.

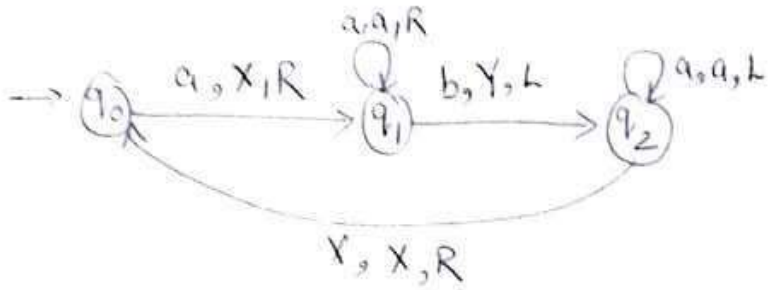
B	B	X	a	a	Y	b	b	B	B
		↑							

whenever we read X, no need to change X, but move read/write head towards Right



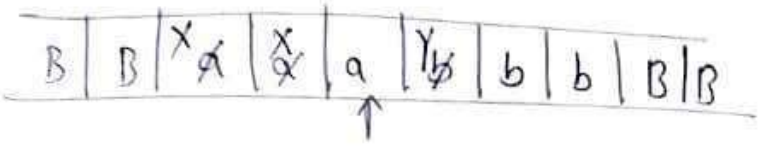
steps





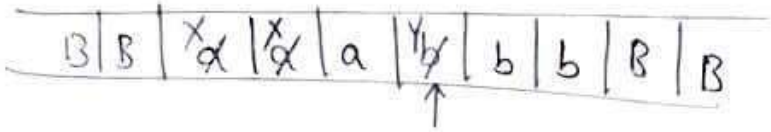
Here a is applied on q_0, so 'a' is replaced with X and move towards Right

Step 9:

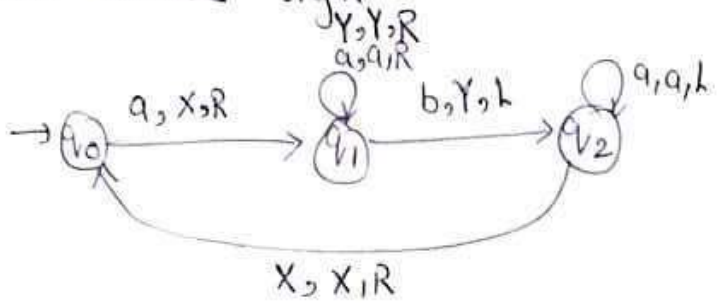


Now in state q_1 on reading a, no need to change state just move towards right.

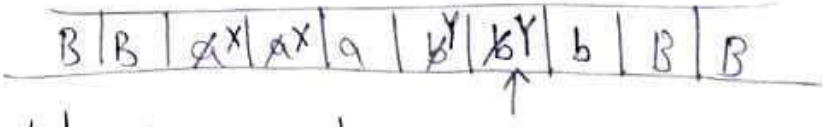
Step 10:



Now on reading $Y \overset{a}{\underset{A}{}}$ no need to change Y, just move read/write head towards right.



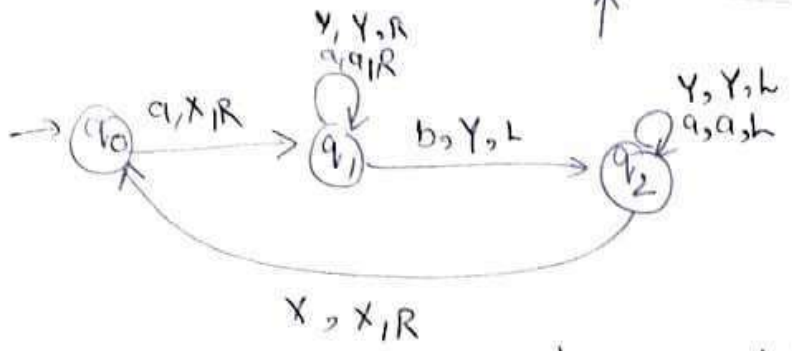
Step 11:



Now at state q_1 on reading b, change state to Y and move read/write head towards L.

step 12:

B	B	X	X	a	Y	Y	b	B	B



Here at state q_2 on reading Y, don't change state, just move towards left.

Step 13:

B	B	X	X	a	Y	Y	b	B	B

Here at state q_2 on read a, don't change change state just move towards left.

step 14:

B	B	X	X	a	Y	Y	b	B	B

If we read X on q_2, do'nt change X, but move read/write head towards R, leading to q_0.

step 15.

B	B	X	X	X	Y	Y	b	B	B

At q_0 on reading a, change context to X and move read/write head to words right

step 16:

B	B	X	X	X	Y	Y	b	B	B

Now at q_1 on reading Y, dont change the content, just move read/write head towards right.

step 17

B	B	a^X	a^X	X^X	Y^Y	B^Y	b	B	B
						↑			

Same as before step.

step 18

B	B	a^X	a^X	a^X	Y^Y	Y^Y	B^Y	B	B
							↑		

Now at q_1 on reading b, change the content to Y and state is changed to q_2 and move towards L.

step 19

B	B	a^Y	a^Y	a^X	Y^Y	Y^Y	B^Y	B	B
						↑			

Same as before step.

step 20

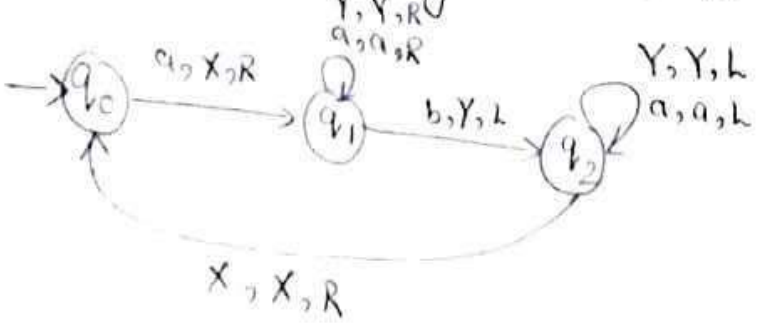
B	B	a^X	a^X	a^X	Y^Y	Y^Y	Y^Y	B	B
					↑				

Same as before step

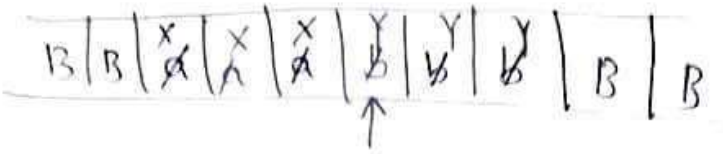
step 21

B	B	X^X	X^X	X^X	Y^Y	Y^Y	Y^Y	B	B
		↑							

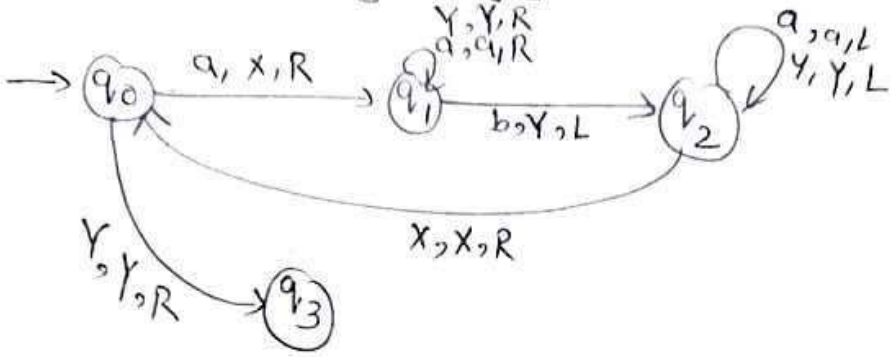
Now at q_2 on reading X move to q_0, and move towards right.



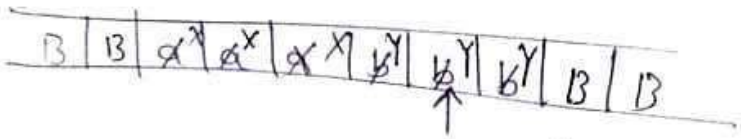
Step22



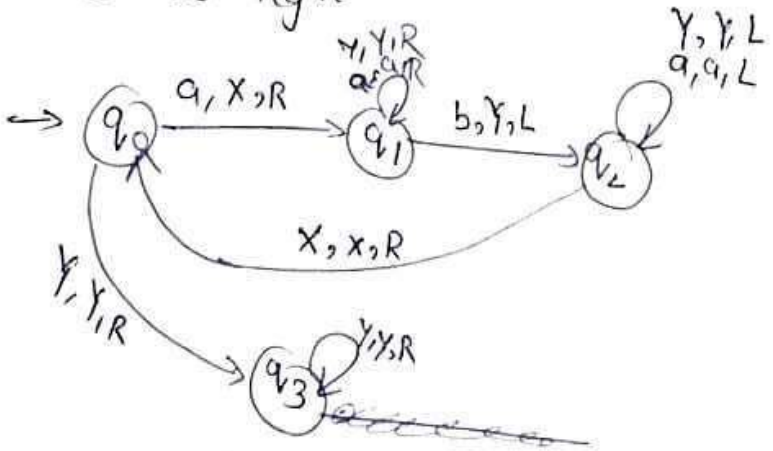
Now at q_0, on seeing Y, don't change Y, just move towards right by changing state to q_3.



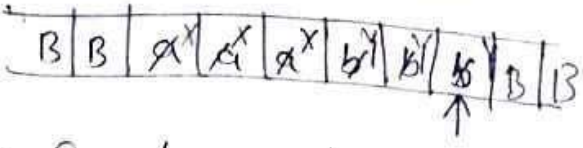
step23



Now at q_3, on seeing Y, don't change Y, just move read/write head towards right.



step24

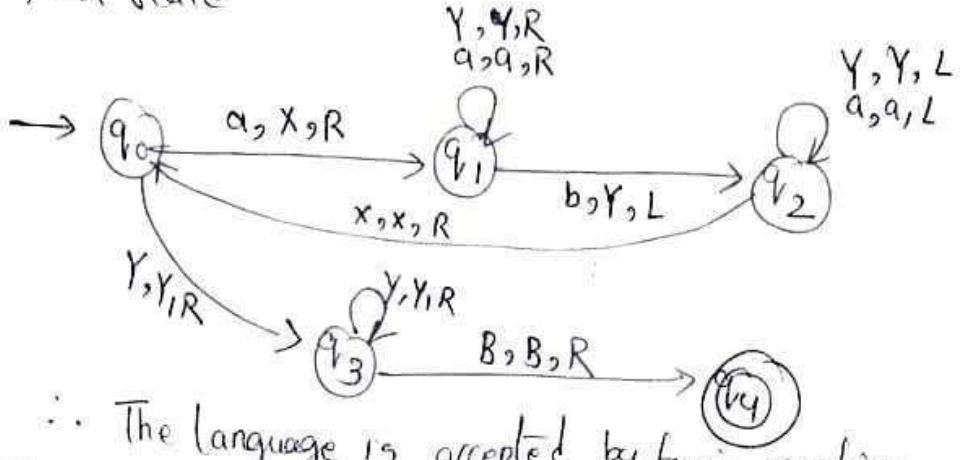


Repeat before step and move read/write head towards right.

step25

B	B	X	X	X	Y	Y	Y	B	B

Now at q_3, when we see blank symbol then it should be replaced by blank symbol only and leads to final state



∴ The language is accepted by turing machine.

Transition table is as follows, rows correspond to states and columns correspond to tape symbols.

states	a	b	X	Y	B
q_0	(q_1, X, R)	ϕ	ϕ	(q_3, Y, R)	ϕ
q_1	(q_1, a, R)	(q_2, Y, L)	ϕ	(q_1, Y, R)	ϕ
q_2	(q_2, a, L)	ϕ	(q_0, X, R)	(q_2, Y, L)	ϕ
q_3	ϕ	ϕ	ϕ	(q_3, Y, R)	(q_4, B, R)
q_4	ϕ	ϕ	ϕ	ϕ	ϕ

Formal representation of turing machine is

$M = (\{q_0, q_1, q_2, q_3, q_4\}, \{a, b\}, \{a, b, X, Y, B\}, \delta, q_0, B, \{q_4\})$

Transition functions are

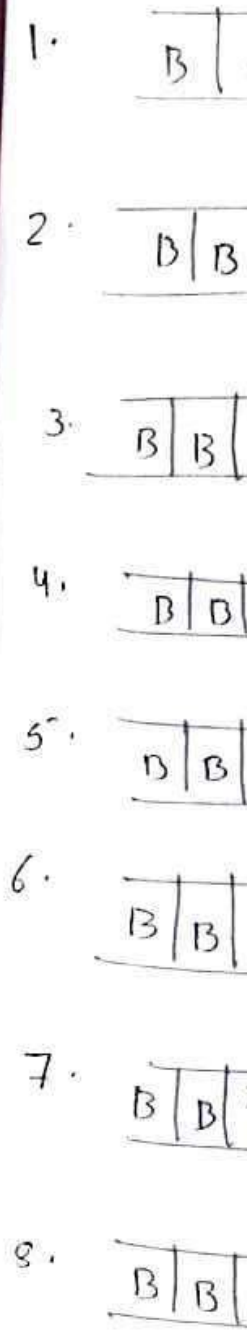
1. $\delta(q_0, a) = (q_1, X, R)$
2. $\delta(q_1, a) = (q_1, a, R)$
3. $\delta(q_1, b) = (q_2, Y, L)$
4. $\delta(q_2, a) = (q_2, a, L)$
5. $\delta(q_2, X) = (q_0, X, R)$
6. $\delta(q_1, Y) = (q_1, Y, R)$
7. $\delta(q_2, Y) = (q_2, Y, L)$
8. $\delta(q_0, Y) = (q_3, Y, R)$
9. $\delta(q_3, Y) = (q_3, Y, R)$
10. $\delta(q_3, B) = (q_4, H, -)$ (H: halts)

Ex.2: Design turing m/c for $L = \{a^{2n} b^n \mid n \geq 1\}$

Soln: Language $L = \{aab, aaaabb, aaaaaabbb, \ldots\}$

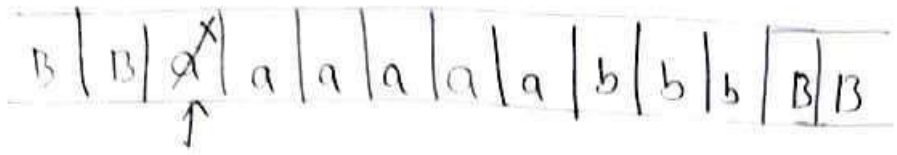
→ Here the logic is, if we read two conserative a's then make them as two x's and move right till we read b and replace b with Y ie., aaaabb → XXaaYb.

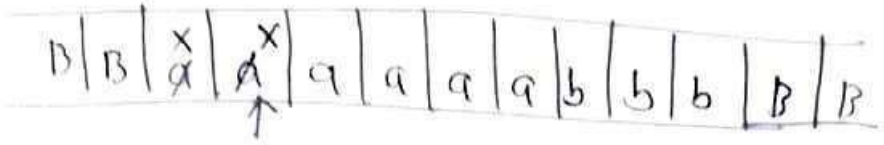
→ Then move head towards left, till we get X, then make two a's to two x's, then move towards right and replace b with Y. ie., aaaabb → XXXXYY

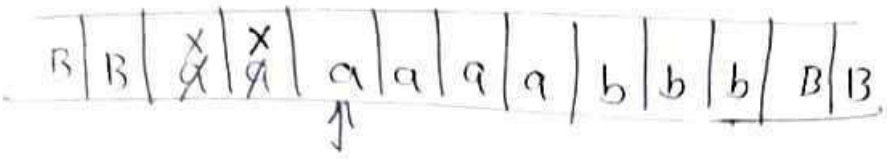


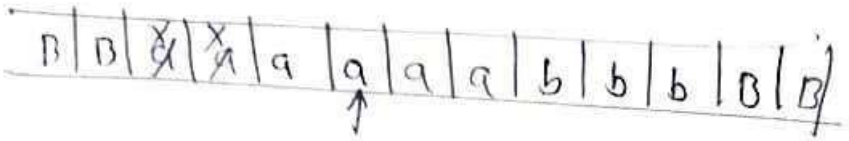
→ Then same right and

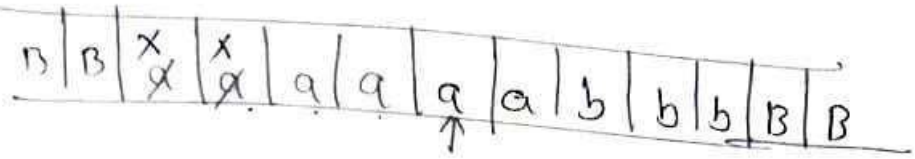
→ Transition

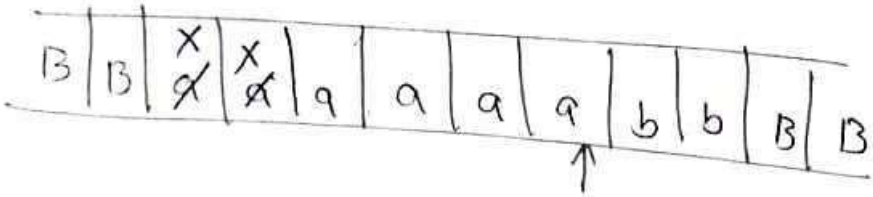
1. 

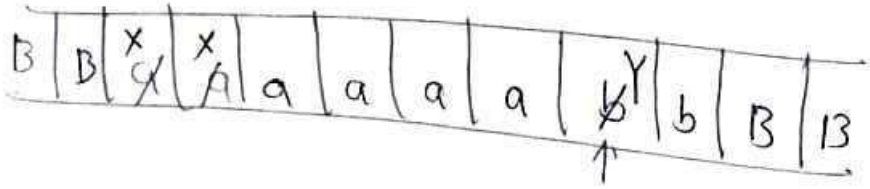
2. 

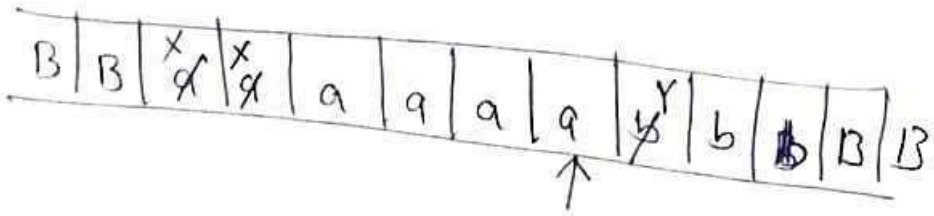
3. 

4. 

5. 

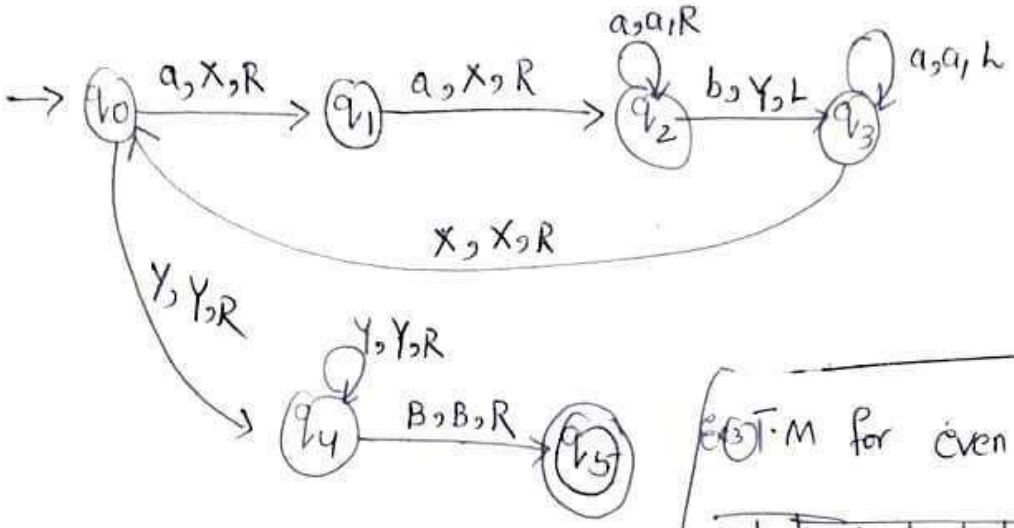
6. 

7. 

8. 

→ Then same process continues till we get X, then move right and repeat the same process.

→ Transition diagram is as follows.



Transition functions are

$\delta(q_0, a) = (q_1, X, R)$

$\delta(q_1, a) = (q_2, X, R)$

$\delta(q_2, a) = (q_2, a, R)$

$\delta(q_2, b) = (q_3, a, L)$

$\delta(q_3, a) = (q_3, a, L)$

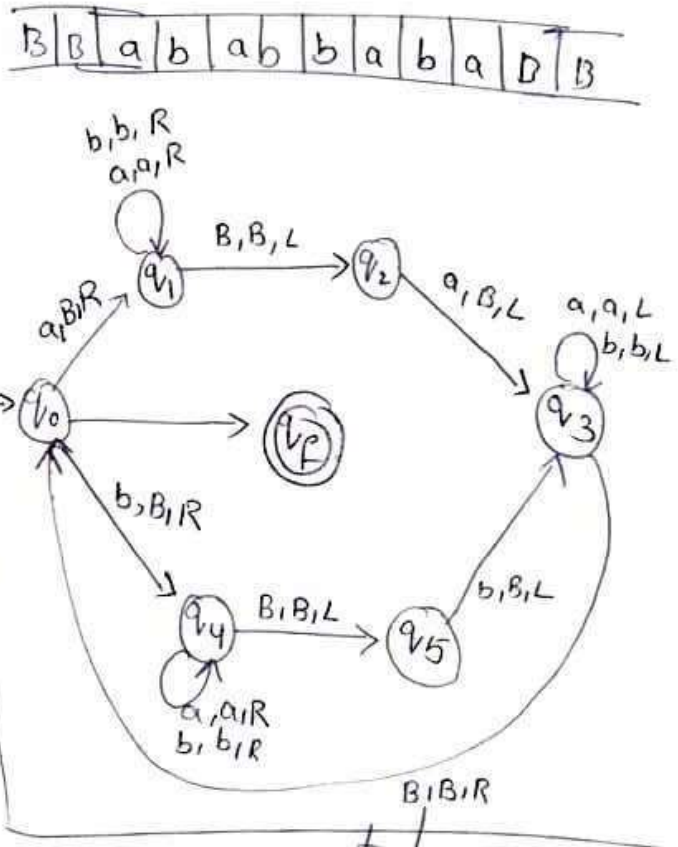
$\delta(q_3, X) = (q_0, X, R)$

$\delta(q_0, Y) = (q_4, Y, R)$

$\delta(q_4, Y) = (q_4, Y, R)$

$\delta(q_4, B) = (q_5, B, -)$.

Ex3 T.M for even palindrome



String is accepted.

Ex3 Design turing machine for ~~even~~ palindrome.

Soln: B B a b a a b a B B

↓ even palind

B B a a a B B

↓ odd palind

Ex: Construct a T.M which accepts the language ~~abaa~~ over ~~$\Sigma = \{a, b\}$~~ of subtraction in unary notation.

Soln:

$$f(m,n) = \begin{cases} m-n & \text{if } m > n \\ 0 & \text{if } m \leq n \end{cases}$$

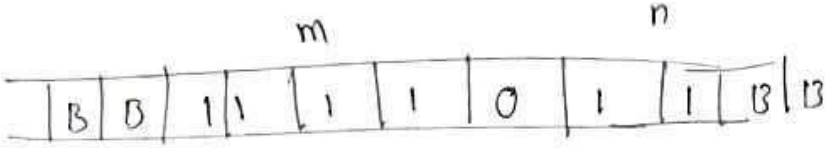
Let $m = 4$ $n = 2$

Represented by $(111) - (11) = 11$

(m = 111, n = 11)

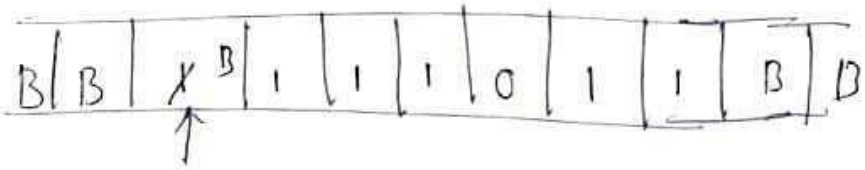
If $m = 2$ $n = 4$

$m - n = 0$



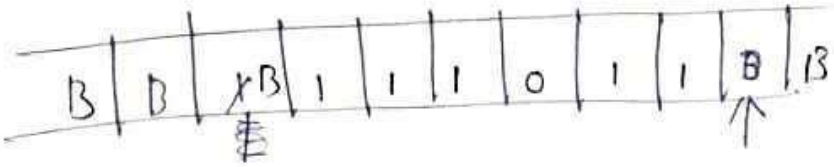
Here m & n are seperated by using seperator '0'

Step1:



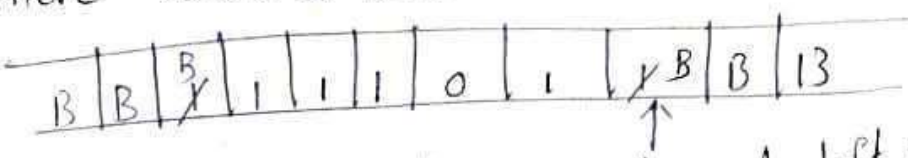
Replace 1 with B and move towards right without changing any symbol; till Blank symbol

Step2:



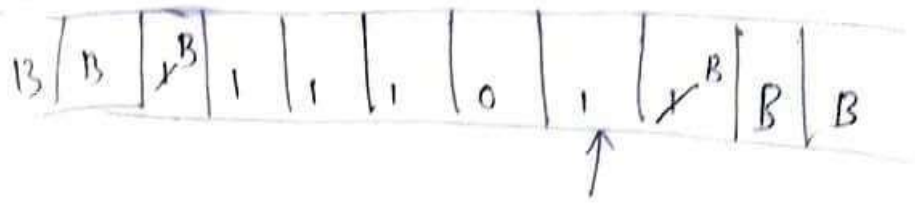
Now move towards left

Step3:

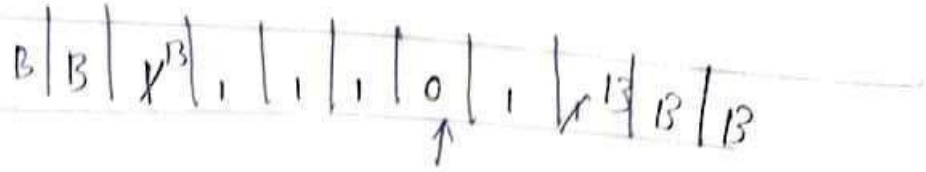


Now replace '1' with B and move towards left until 'B' is read.

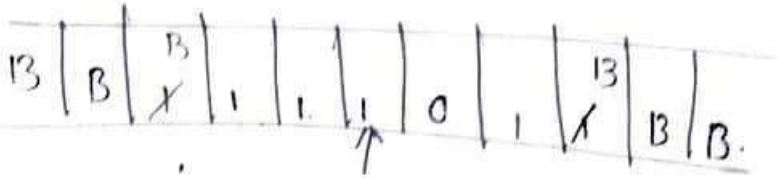
step4



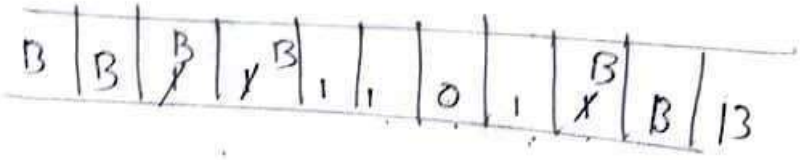
step5



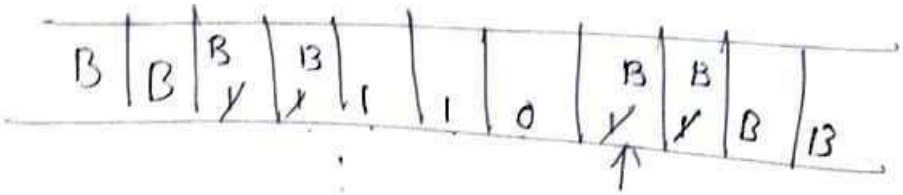
step6



step10



step16



step21