TOPICS (FINAL)

- CONTEXT FREE LANGUAGE
- CONTEXT FREE GRAMMAR
- FORMAL DEFINITION (CFG)
- DESIGNING (CFG)
- DERIVATION (CFG)
- PARSE TREE
- AMBIGUITY
- CHOMSKY NORMAL FORM
- FORMAL DEFINITION (PDA)
- DESIGNING PDA
- FORMAL DEFINITION (TM)
- TM TESTING MEMBERSHIP
  - → ALGORITHM
  - → IMPLEMENTATION
  - → STATE DIAGRAM
- CONFIGURATION SIMULATION

TOPICS (MID)

- FORMAL DEFINITION (DFA)
- DESIGNING MODEL (DFA)
- SIMULATION (DFA)
- FORMAL DEFINITION (NFA)
- DESIGNING MODEL (NFA)
- CONVERSION: (NFA → DFA)
- NFA TREE GENERATION
- CLOSURE
- REGULAR EXPRESSION CREATION
- RE → NFA (CONVERSION)
- RL → RE (CONVERSION)
- RE → RL (CONVERSION)
- DFA → RE (CONVERSION)
- RE → DFA (CONVERSION)

FINAL TERM

* CONTEXT-FREE LANGUAGES

* CONTEXT-FREE LANGUAGES includes all regular languages and many additional languages, such as $\{0^n \# 1^n \mid n \geq 0\}$

Rules :

$$A \rightarrow 0A1$$
$$A \rightarrow B$$
$$B \rightarrow \#$$

$$A \rightarrow 0A1 \mid B$$
$$B \rightarrow \#$$

' → ' can be replaced by

Variable ⟶ Terminal

| — (OR)

Variables :

- Uppercase letters
- Left side of arrow

Terminals :

- Lowercase letters
- Right side of arrow

* WRITE DOWN THE FORMAL DEFINITION OF CONTEXT FREE GRAMMAR :

A context free grammar is a 4-touple (V, Σ, R, S). where -

→ V is a finite set called variables

→ Σ is a finite set, disjoint from V, called alphabet

→ R is a finite set of rules, with each rule being a variable and a strings of variables and terminals

→ $S \in V$ is the start variable.

* DESIGNING CFG (RL → CFG)

* A = {w | w begins with a and ends with b} Σ = {a, b}

S → aTb
T → aT | bT | ∈

R.e = a(a∪b)*b

* A = {w | w begins with b or ends with a} Σ = {a, b}

S → bT | Ta
T → aT | bT | ∈

R.e = b(a∪b)* ∪ (a∪b)*a

* A = {w | w contains at least 3 'a'} Σ = {a, b}

S → TaTaTaP
T → bT | ∈
P → aP | bP | ∈

R.e = b*ab*ab*a(a∪b)*

* A = {w | w contains at most three 'a'} Σ = {a, b}

S → BABABAB
A → a | ∈
B → bB | ∈

R.e = (b*ab*ab*ab*) ∪ (b*ab*ab*) ∪ (b*ab*) ∪ (b*)

* A = {w | w contains substring 'aba'} Σ = {a, b}

S → TabaT
T → aT | bT | ∈

R.e = (a∪b)*aba(a∪b)*

* A = {w | w does not contain substring 'aba'} Σ = {a, b}

S → BTAB
T → AbbBT | ∈
A → aA | ∈
B → bB | ∈

R.e = b*(a*bbb*)*a*b*

*** $A = \{w \mid$ every odd position of w is 'a' $\}$ $\Sigma = \{a, b\}$

$S \rightarrow aT \mid \epsilon$

$T \rightarrow aS \mid bS \mid \epsilon$

r.e : $(a(a \cup b))^* (\epsilon \cup a)$

* $A = \{w \mid w = a^m b^n$ and $m = n\}$ $\Sigma = \{a, b\}$

$T \rightarrow aTb \mid \epsilon$

* $A = \{w \mid w$ has even length $\}$ $\Sigma = \{a, b\}$

$S \rightarrow aT \mid bT$

$T \rightarrow aP \mid bP$

$P \rightarrow aT \mid bT \mid \epsilon$

$S \rightarrow BS \mid \epsilon$

$B \rightarrow AA$

$A \rightarrow a \mid b$

r.e = $(a \cup b)(a \cup b))^*$

$a(a \cup b)^* \cup b(a \cup b)^*$

$a(a \cup (a \cup b)^*)$

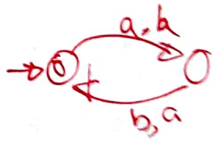
* $A = \{w \mid w = a^m b^n$ and $m > n\}$ $\Sigma = \{a, b\}$

$S \rightarrow AT$

$A \rightarrow aA \mid a$

$T \rightarrow aTb \mid \epsilon$

r.e = $\frac{a \cdot ab}{A \quad T}$



* $\{w \in \{0,1\}^* \mid w = w^R$ and $|w|$ is even $\}$

$S \rightarrow 0S0 \mid 1S1 \mid \epsilon$

* $A = \{a^i b^j c^k \mid$ where $i = j$ or $j = k$ and $i, j, k \geq 0\}$ and $\Sigma = \{a, b, c\}$

$S \rightarrow PR \mid QT$

$P \rightarrow aPb \mid \epsilon$

$R \rightarrow cR \mid \epsilon$

$T \rightarrow bTc \mid \epsilon$

$Q \rightarrow aQ \mid \epsilon$

* $L = \{a^n \mid n \geq 0\}$

$= \{\epsilon, a, aa, aaa \ldots\ldots\}$

$A \rightarrow aA \mid \epsilon$

Generate aa

$A \rightarrow aA$

$\rightarrow aaA$

$\rightarrow aa\epsilon$

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

$= \{a, aa, aaa \ldots\ldots\}$

$A \rightarrow aA \mid a$

* $L = \{$set of all strings over $a, b\}$

$= (a \cup b)^*$

$A \rightarrow aA \mid bA \mid \epsilon$

* $L = \{$set of all strings which length at least 2 $\}$ $\Sigma = \{a, b\}$

$L = \{aa, ab, ba, bb, aaa \ldots\}$

$(a \cup b)(a \cup b)(a \cup b)^*$

$S \rightarrow AAB$

$A \rightarrow a \mid b$

$B \rightarrow aB \mid bB \mid \epsilon$

* $L = \{$starts and ends with same symbol$\}$ $\Sigma = \{a, b\}$

$L = a(a \cup b)^* a \mid b(a \cup b)^* b \mid \epsilon \mid a \mid b$

$S \rightarrow aAa \mid bAb \mid \epsilon \mid a \mid b$

$A \rightarrow aA \mid bA \mid \epsilon$

* DERIVATION OF CFG

* $X \to YD$
  $Y \to Az$
  $A \to xAzz \mid C$
  $C \to Cy \mid \epsilon$
  $D \to Dd \mid \epsilon$

∴ Use left most derivation to xyyzzzd

$X \to YD$
$\to AzD$
$\to xAzzzD$
$\to xCzzzD$
$\to xCyzzzD$
$\to xCyyzzzD$
$\to xyyzzzD$
$\to xyyzzzDd$
$\to xyyzzzd$

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

∴ Use left most derivation to aaab

$S \to AB$
$\to aAAB$
$\to aaAB$
$\to aaaB$
$\to aaab$

* L = {even number of 'a' and each 'a' is followed by at least one 'b'} and prove bababababb

$= \underset{B}{b^*}(\underset{T}{abb^*.abb^*})^*$

$S \to BT$
$B \to bB \mid \epsilon$
$E \to abBabB$
$T \to ET \mid \epsilon$

S → BT
→ bBT
→ bT
→ bRT
→ babBabBT
→ bababBT
→ bababT
→ bababRT
→ babababBabBT
→ bababababBT
→ bababababbBT
→ bababababbbT
→ bababababbb

* Derive the string "aabbabba" on left most derivation given. S → aB|bA
A → a|aS|bAA
B → b|bS|aBB

S → aB
→ aaBB
→ aabB
→ aabbS
→ aabbaB
→ aabbabS
→ aabbabbA
→ aabbabba

* L = { has odd number of 'a' and ends with b } $\Sigma = \{a, b\}$

$= b^*abb^*(abb^*abb^*)^*$ and prove 'baabbab' — 'abbabaabab'

S → BaBT
B → bB | ∈
R → aBaB
T → RT | ∈

S → BaBT
→ bBaBT
→ baBT
→ baT
→ baRT
→ baaBaBT
→ baabBaBT
→ baabbBaBT
→ baabbaBT
→ baabbabBT
→ baabbabT
→ baabbab

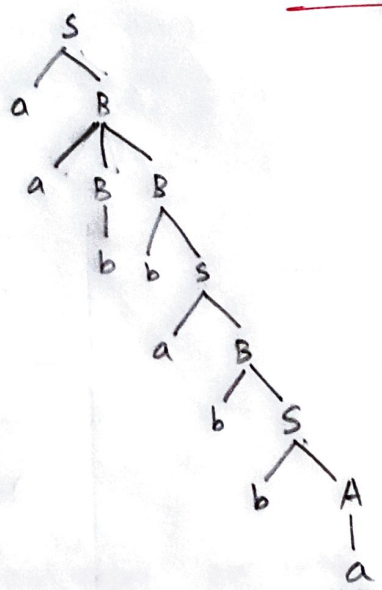
S → BaBT
→ aBT
→ abBT
→ abbBT
→ abbT
→ abbRT
→ abbaBaBT
→ abbabBaBT
→ abbabaBT
→ abbabaT
→ abbabaRT
→ abbabaaBaBT
→ abbabaabBaBT
→ abbabaabaBT
→ abbabaababBT
→ abbabaababT
→ abbabaabab

PARSE TREE

* DERIVE THE STRING : 'aabbabba' using Parse Tree :

S → aB | bA
A → a | aS | bAA
B → b | bS | aBB

(Left Most)



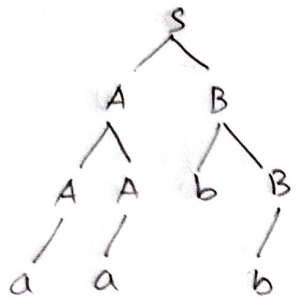
* Given, S → A | AB
A → ∈ | a | Ab | AA
B → b | bC | Bc | bB

Prove the string 'aabb'

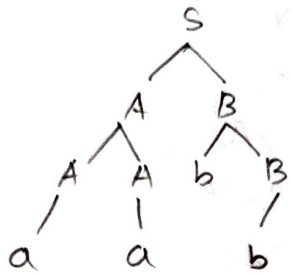
L.M.D
S → AB
→ AAB
→ aAB
→ aaB
→ aabB
→ aabb

R.M.D
S → AB
→ AbB
→ Abb
→ AAbb
→ Aabb
→ aabb

Parse Tree : (LMD)



Parse Tree : (RMD)

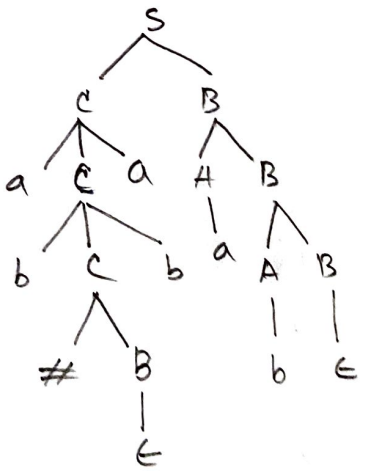


* Given, $S \rightarrow CB$
$C \rightarrow aCa \mid bCb \mid \#B$
$B \rightarrow AB \mid \in$
$A \rightarrow a \mid b$

Prove the string 'ab#baab'

$S \rightarrow CB$
$\rightarrow aCaB$
$\rightarrow abCbaB$
$\rightarrow ab\#BbaB$
$\rightarrow ab\#baB$
$\rightarrow ab\#baAB$
$\rightarrow ab\#baaB$
$\rightarrow ab\#baaAB$
$\rightarrow ab\#baabB$
$\rightarrow ab\#baab$

Parse Tree :-



* Use left most derivation. ⊢ xyyzzzd

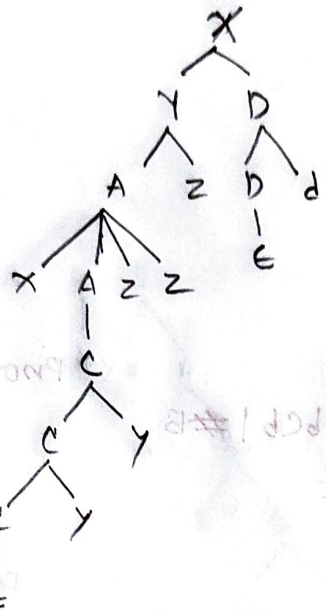
$X \rightarrow YD$

$Y \rightarrow Az$

$A \rightarrow xAzz \mid C$

$C \rightarrow Cy \mid \epsilon$

$D \rightarrow Dd \mid \epsilon$



* AMBIGUITY – PARSE TREE

A grammar is said to be Ambigious if there exists two or more derivation tree for a string w (that means two or more left derivation trees)

$G = (\{S\}, \{a+b, +, *\}, P, S)$ where P consists of

$S \rightarrow S+S \mid S*S \mid a \mid b$

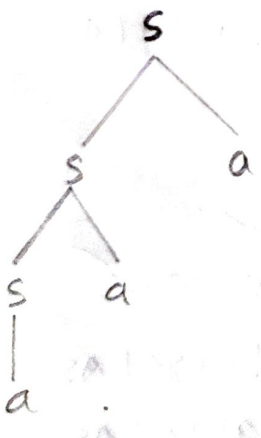
A string $a+a*b$ can be generated as:

$S \rightarrow S+S$
$\rightarrow a+S$
$\rightarrow a+S*S$
$\rightarrow a+a*S$
$\rightarrow a+a*b$

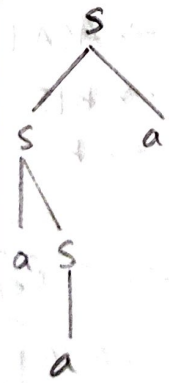
$S \rightarrow S*S$
$\rightarrow S+S*S$
$\rightarrow a+S*S$
$\rightarrow a+a*S$
$\rightarrow a+a*a$

---

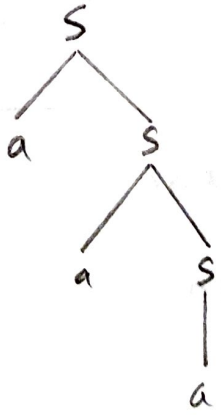
* $G: S \rightarrow Sa \mid aS \mid a$



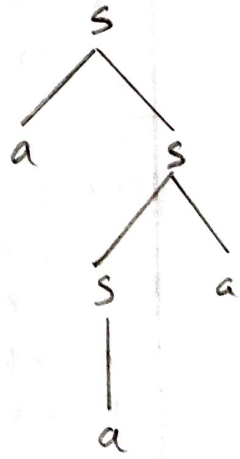
LMD



LMD



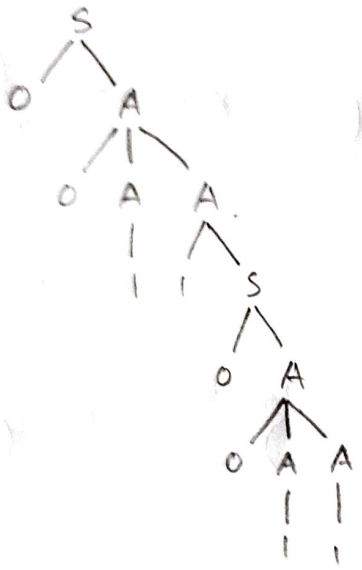
RMD



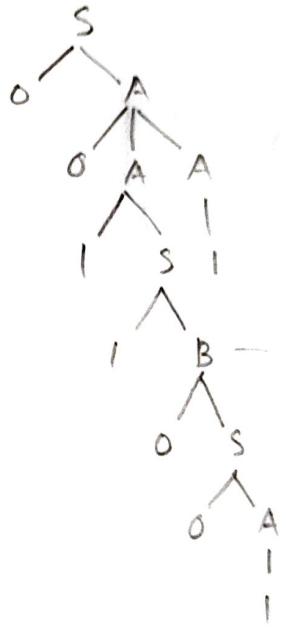
RMD

* What is ambigious Gnammar ? Prove that the following gnammar is ambigious. Given 00110011

S → 0A | 1B
A → 0AA | 1S | 1
B → 1BB | 0S | 0



Result : 00110011



Result : 00110011

Therefore, the Following Gnammar is ambiguous for the String 00110011 cause there exists more derivation.

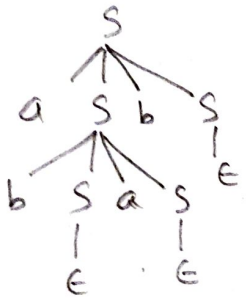
* $S \rightarrow aSbS \mid bSaS \mid \epsilon$ Prove the Gramman is ambiguous or not.

Let assume a string 'abab'
Using Left Most Derivation,

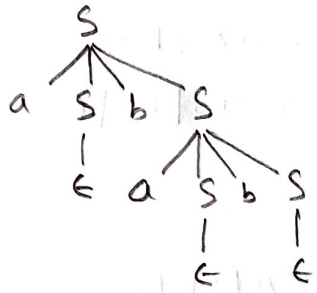
$S \rightarrow aSbS$
$\rightarrow abSaSbS$
$\rightarrow abaSbS$
$\rightarrow ababS$
$\rightarrow abab$

$S \rightarrow aSbS$
$\rightarrow abS$
$\rightarrow abaSbS$
$\rightarrow ababS$
$\rightarrow abab$

Parse Tree:



Parse Tree:



Two or more derivation exists of abab. Therefore The Gramman is ambiguous.

* Practice

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

$S \rightarrow aSbb$

$S \rightarrow abb$

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

$S \rightarrow aSa$

$S \rightarrow bSb$

$S \rightarrow c$

* $L = \{ a^i b^j c^k \mid i,j,k \geq 0 \text{ and } i=j \text{ or } i=k \}$

$S \rightarrow XY \mid Z$

$X \rightarrow aXb \mid \epsilon$

$Y \rightarrow cY \mid \epsilon$

$W \rightarrow aWc \mid Z$

$Z \rightarrow bZ \mid \epsilon$

* CFG Practice :

1. All strings containing at least two 0 and at least one 1

$S \rightarrow 00B1T \mid 01A0T \mid 1A0A0T$

$A \rightarrow 1A \mid \epsilon$

$B \rightarrow 0B \mid \epsilon$

$T \rightarrow 1T \mid 0T \mid \epsilon$

RE = $000^*1(0U1)^* U$
$011^*0(0U1)^* U$
$11^*01^*0(0U1)^*$

2. All strings not containing substring 000.

$S \rightarrow TR$

$T \rightarrow 1 \mid 01 \mid 001 \mid \epsilon$

$R \rightarrow \epsilon \mid 0 \mid 00$

RE = $\overline{(1U01U001)^*}^{T}$ $\underbrace{(\epsilon U0U00)}_{R}$

3. Strings of odd length. $\Sigma = \{a,b\}^*$

$S \rightarrow aX \mid bX$

$X \rightarrow aS \mid bS \mid \epsilon$

4. W|W has even number of '0' and even number of '1'

$S \rightarrow 0A \mid 1B \mid \epsilon$

$A \rightarrow 0S \mid 1C$

$B \rightarrow 1S \mid 0C$

$C \rightarrow 1A \mid 0B$

* Practice

* Construct CFG FOR FOLLOWING LANGUAGES : FOR $\Sigma = \{a, b\}$

(i) $A = \{w \mid w$ has even number of 'A' $\}$

$S \rightarrow BT$
$B \rightarrow bB \mid \epsilon$
$R \rightarrow aBaB$
$T \rightarrow RT \mid \epsilon$

RE = $\underbrace{b^*}_{B}\underbrace{(ab^*ab^*)^*}_{T}$

(ii) $A = \{w \mid$ each a in w is followed by at least one $b\}$

$S \rightarrow BT$
$B \rightarrow bB \mid \epsilon$
$R \rightarrow abB$
$T \rightarrow RT \mid \epsilon$

RE = $\underbrace{b^*}_{B}\underbrace{(abb^*)^*}_{T}$

(iii) $A = \{w \mid w$ contains exactly 2 'a' and at least two 'b' $\}$

$S \rightarrow bbaaB \mid babaB \mid aabbB \mid ababB$
$B \rightarrow bB \mid \epsilon$

$bbaa\underbrace{b^*}_{B} \cup baba\underbrace{b^*}_{B} \cup aabb\underbrace{b^*}_{B} \cup ababb^+$

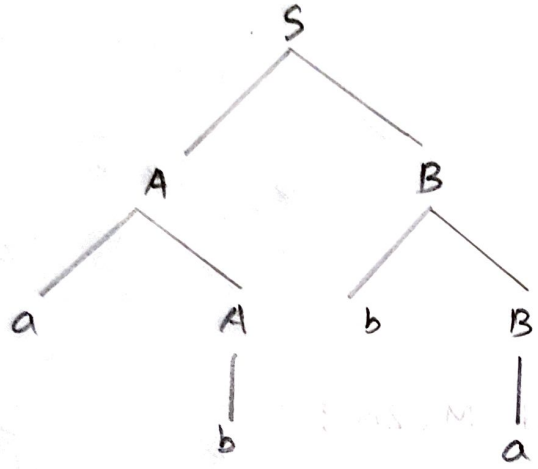
(iv) $A = \{w \mid w$ has even number of 'a' and each a in w is followed by at least one 'b' $\}$

$S \rightarrow BT$
$B \rightarrow bB \mid \epsilon$
$R \rightarrow abBabB$
$T \rightarrow RT \mid \epsilon$

RE = $\underbrace{b^*}_{B}\underbrace{(abb^*abb^*)^*}_{T}$

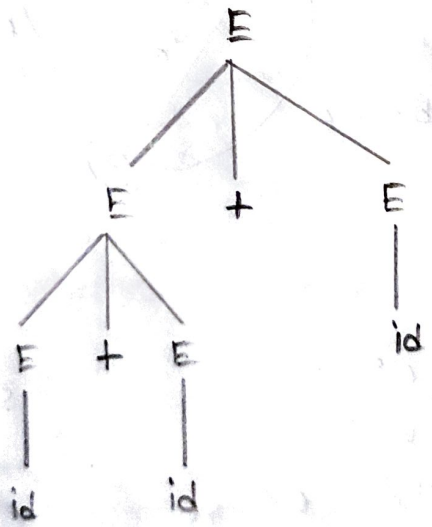
G_1: $S \rightarrow AB$
$A \rightarrow aA \mid b$
$B \rightarrow bB \mid a$

w: abba



G_2: $E \rightarrow E + E \mid id$

w: id + id + id



$\{ 0^n 1^n \mid n \geq 1 \} = L$ *

CHOMSKY NORMAL FORM (CNF)

CFG → CNF

Conversion Rules:

1. Add new start variable (substitute by old start variables)
2. Remove all epsilon rules
3. Remove all unit rules
4. Convert all rules to CNF rule format (Optional)

Given,

$S \rightarrow ASA \mid aB$

$A \rightarrow B \mid S$

$B \rightarrow b \mid \epsilon$

Step-1: Add start var

$S_0 \rightarrow S$

$S \rightarrow ASA \mid aB$

$A \rightarrow B \mid S$

$B \rightarrow b \mid \epsilon$

Step-2: Remove $B \rightarrow \epsilon$

$S_0 \rightarrow S$

$S \rightarrow ASA \mid aB \mid a$

$A \rightarrow B \mid S \mid \epsilon$

$B \rightarrow b$

Step-3: Remove $A \rightarrow \epsilon$

$S_0 \rightarrow S$

$S \rightarrow ASA \mid aB \mid a \mid SA \mid AS \mid S$

$A \rightarrow B \mid S$

$B \rightarrow b$

Step-3: Remove $S_0 \rightarrow S$

After step 3 $S \rightarrow S$

$S_0 \rightarrow ASA \mid aB \mid a \mid SA \mid AS$

$S \rightarrow ASA \mid aB \mid a \mid SA \mid AS$

$A \rightarrow B \mid S$

$B \rightarrow b$

Step-4: Remove $A \rightarrow B$

$S_0 \rightarrow ASA \mid aB \mid a \mid SA \mid AS$

$S \rightarrow ASA \mid aB \mid a \mid SA \mid AS$

$A \rightarrow b \mid S$

$B \rightarrow b$

Step-5: Remove $A \rightarrow S$

$S_0 \rightarrow ASA \mid aB \mid a \mid SA \mid AS$

$S \rightarrow ASA \mid aB \mid a \mid SA \mid AS$

$A \rightarrow b \mid ASA \mid aB \mid a \mid SA \mid AS$

$B \rightarrow b$

Step-6: Add $C \rightarrow SA$

$S_0 \rightarrow AC \mid aB \mid a \mid SA \mid AS$

$S \rightarrow AC \mid aB \mid a \mid SA \mid AS$

$A \rightarrow b \mid AC \mid aB \mid a \mid SA \mid AS$

$B \rightarrow b$

$C \rightarrow SA$

Step-7: Add $D \rightarrow a$

$S_0 \rightarrow AC \mid DB \mid a \mid SA \mid AS$

$S \rightarrow AC \mid DB \mid a \mid SA \mid AS$

$A \rightarrow b \mid AC \mid DB \mid a \mid SA \mid AS$

$B \rightarrow b$

$C \rightarrow SA$

$D \rightarrow a$

CNF : Rules Format

1. A variable substitute by two var
   $V_1 \rightarrow V_2V_3$
2. A variable substitute by one term
   $V_1 \rightarrow t$
3. The start variable may have ϵ
   $S_0 \rightarrow \epsilon$

* Given, $S \to bS|aT|\epsilon$
$T \to aT|bR|\epsilon$
$R \to bS|\epsilon$

Step-1 : Add Start Var P

$P \to S$
$S \to bS|aT|\epsilon$
$T \to aT|bR|\epsilon$
$R \to bS|\epsilon$

Step-2 : Remove $R \to \epsilon$

$P \to S$
$S \to bS|aT|\epsilon$
$T \to aT|bR|\epsilon|b$
$R \to bS$

Step-3 : Remove $T \to \epsilon$

$P \to S$
$S \to bS|aT|\epsilon|a$
$T \to aT|bR|b|a$
$R \to bS$

Step-4 : Remove $S \to \epsilon$

$P \to S|\epsilon$
$S \to bS|aT|a|b$
$T \to aT|bR|b|a$
$R \to bS|b$

Step-5 :
Remove $P \to S$

$P \to bS|aT|a|b|\epsilon$
$S \to bS|aT|a|b$
$T \to aT|bR|b|a$
$R \to bS|b$

Step-6 : Add rule $A \to a$, $B \to b$

$P \to BS|AT|a|b|\epsilon$
$S \to BS|AT|a|b$
$T \to AT|BR|b|a$
$A \to a$
$B \to b$

Given, CFG

$S \to aXbX$

$X \to aY \mid bY \mid \epsilon$

$Y \to X \mid c$

Add start van T

$T \to S$

$S \to aXbX$

$X \to aY \mid bY \mid \epsilon$

$Y \to X \mid c$

Remove $X \to \epsilon$

$T \to S$

$S \to aXbX \mid aXb \mid abX \mid ab$

$X \to aY \mid bY$

$Y \to X \mid c \mid \epsilon$

Remove $Y \to \epsilon$

$T \to S$

$S \to aXbX \mid aXb \mid abX \mid ab$

$X \to aY \mid bY \mid a \mid b$

$Y \to X \mid c$

Remove $T \to S$

$T \to aXbX \mid aXb \mid abX \mid ab$

$S \to aXbX \mid aXb \mid abX \mid ab$

$X \to aY \mid bY \mid a \mid b$

$Y \to X \mid c$

Remove $Y \to X$

$T \to aXbX \mid aXb \mid abX \mid ab$

$S \to aXbX \mid aXb \mid abX \mid ab$

$X \to aY \mid bY \mid a \mid b$

$Y \to aY \mid bY \mid a \mid b \mid c$

Add Rule $A \to a$, $B \to b$

$T \to AXBX \mid AXB \mid ABX \mid AB$

$S \to AXBX \mid AXB \mid ABX \mid AB$

$X \to AY \mid BY \mid a \mid b$

$Y \to AY \mid BY \mid a \mid b \mid c$

$A \to a$

$B \to b$

Add Rule $P \to AX$, $R \to BX$

$T \to PQ \mid PB \mid AR \mid AB$

$S \to PQ \mid PB \mid AR \mid AB$

$X \to AY \mid BY \mid a \mid b$

$Y \to AY \mid BY \mid a \mid b \mid c$

$A \to a$

$B \to b$

$P \to AX$

$R \to BX$

* Convert the following CFG to CNF

$S \to XaX \mid bX \mid Y$

$X \to XaX \mid XbX \mid \epsilon$

$Y \to ab$

Step-1: Add start variable

$S_0 \to S$

$S \to XaX \mid bX \mid Y$

$X \to XaX \mid XbX \mid \epsilon$

$Y \to ab$

Step-2: Remove $X \to \epsilon$

$S_0 \to S$

$S \to XaX \mid bX \mid Y \mid Xa \mid aX \mid a \mid b$

$X \to XaX \mid XbX \mid aX \mid Xa \mid Xb \mid bX \mid a \mid b$

$Y \to ab$

Step-3: Remove $S_0 \to S$

$S_0 \to XaX \mid bX \mid Y \mid Xa \mid aX \mid a \mid b$

$S \to XaX \mid bX \mid Y \mid Xa \mid aX \mid a \mid b$

$X \to XaX \mid XbX \mid aX \mid Xa \mid Xb \mid bX \mid a \mid b$

$Y \to ab$

Step-4: Add Rule : $A \to a$, $B \to b$

$S_0 \to XAX \mid BX \mid Y \mid XA \mid AX \mid a \mid b$

$S \to XAX \mid BX \mid Y \mid XA \mid AX \mid a \mid b$

$X \to XAX \mid XBX \mid AX \mid XA \mid XB \mid BX \mid a \mid b$

$Y \to AB$, $A \to a$, $B \to b$

Step-5: Add Rule : $P \to AX$, $R \to BX$

$S_0 \to XP \mid BX \mid Y \mid XA \mid AX \mid a \mid b$

$S \to XP \mid BX \mid Y \mid XA \mid AX \mid a \mid b$

$X \to XP \mid XR \mid AX \mid XA \mid XB \mid BX \mid a \mid b$

$Y \to AB$

$A \to a$

$B \to b$

Step-6: $Y \to AB$

$S_0 \to XP \mid BX \mid AB \mid XA \mid AX \mid a \mid b$

$S \to XP \mid BX \mid AB \mid XA \mid AX \mid a \mid b$

$X \to XP \mid XR \mid AX \mid XA \mid XB \mid BX \mid a \mid b$

$Y \to AB$

$A \to a$

$B \to b$

$P \to AX$

$R \to BX$

PUSH DOWN AUTOMATA

* WRITE THE FORMAL DEFINITION OF PUSH DOWN AUTOMATA :

A push down Automata is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where Q, Σ, Γ, and F are all finite sets and

→ Q is the set of states

→ Σ is the set of alphabet

→ $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$

→ Γ is the stack alphabet

→ $\Gamma_\epsilon = \Gamma \cup \{\epsilon\}$

→ $\delta : Q \times \Sigma_\epsilon \times \Gamma_\epsilon \longrightarrow P(Q \times \Gamma_\epsilon)$

→ $q_0 \in Q$ is the start state

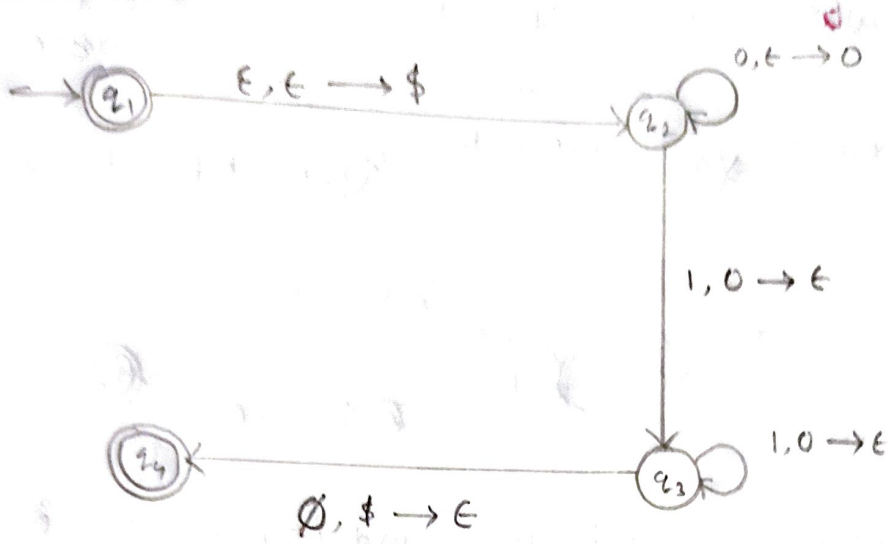
→ $F \subseteq Q$ is the set of accept states.

* Transition (4 conditions)

$$Q \times \Sigma_\epsilon \times \Gamma_\epsilon \longrightarrow P(Q \times \Gamma_\epsilon)$$

IGNORE	ϵ	ϵ
POP 'a'	a	ϵ
PUSH 'b'	ϵ	b
Conditional PUSH	a	b

* PDA For $0^n 1^n$



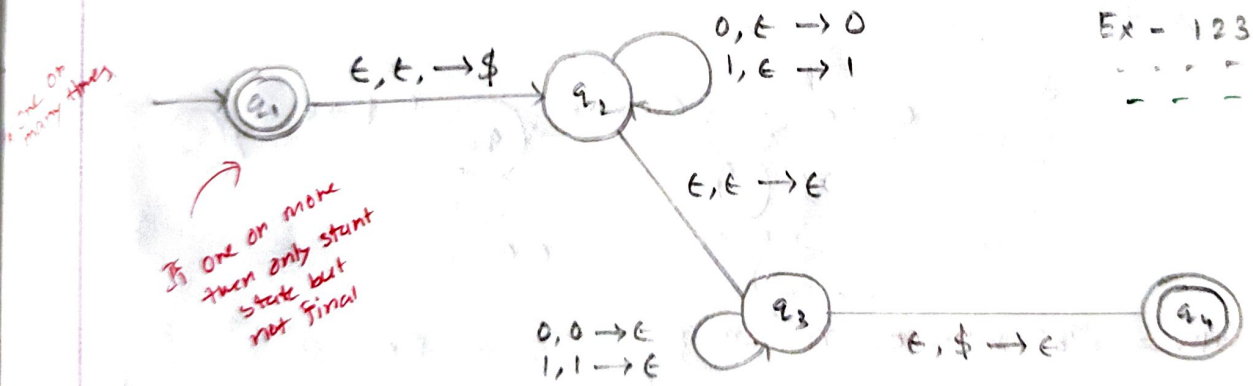
push Dollar first

Ex - 0011
Ex - 000111
. . . . . .

* PDA For $L = \{ww^R \mid w \in \{0,1\}^*\}$

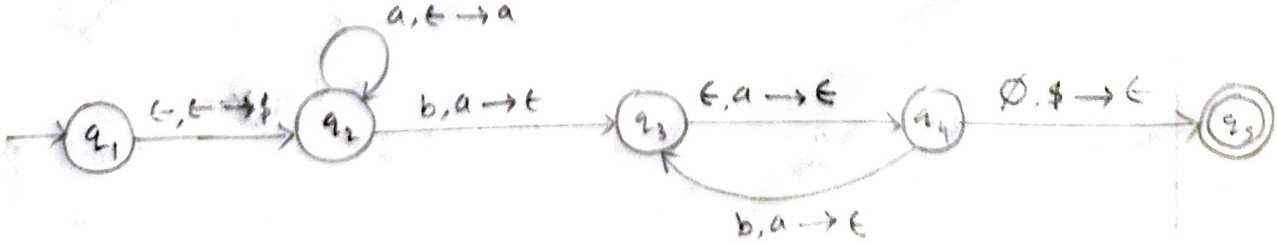
push dollar first

Ex - abba
Ex - 123321
. . . . . .

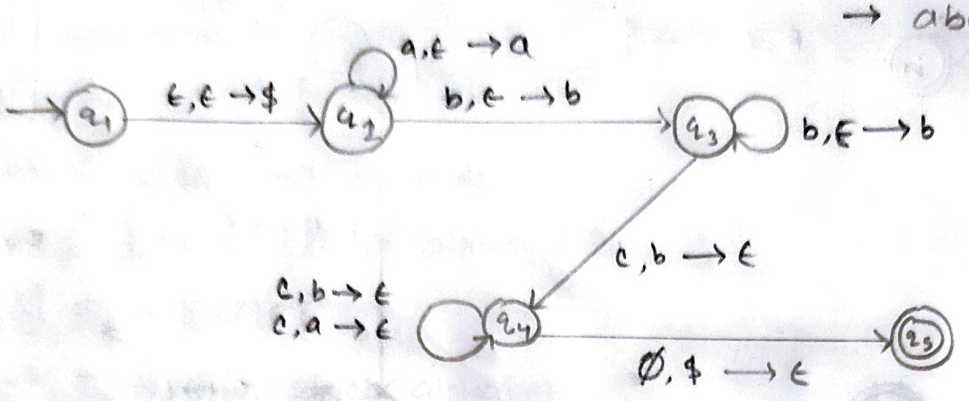


* $\{w \mid w = a^m b^n$ where $m = 2n\}$

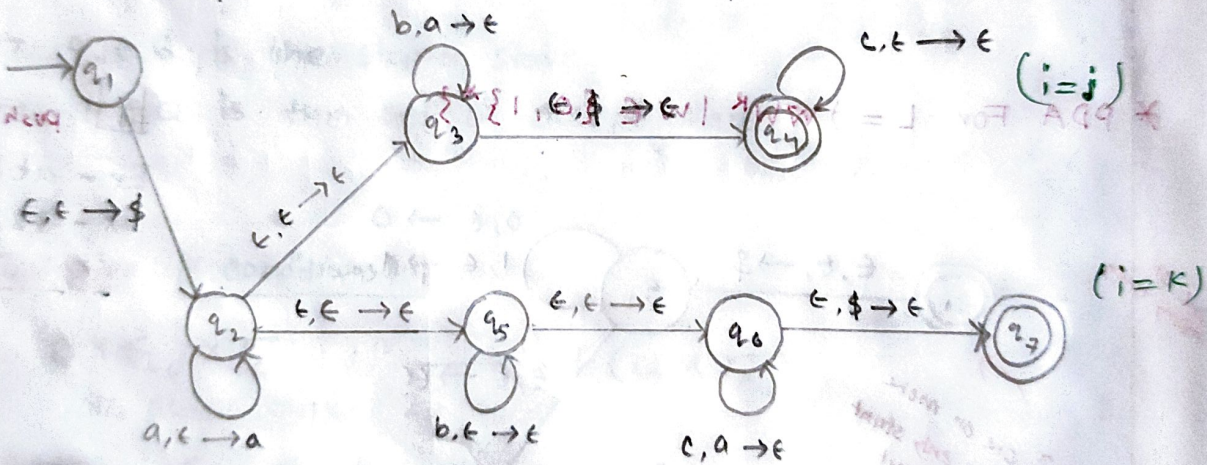
Ex - $a^{2n} b^n$
- $a^2 b^1$ - aab



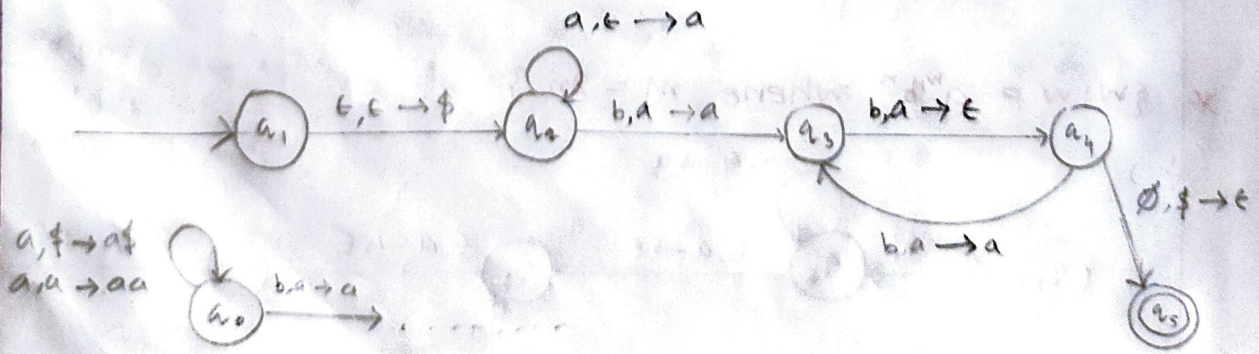
* $\{w | w = a^i b^j c^k$ where $i+j = k\}$ Ex → $a^1 b^1 c^2$ → abcc



* $L = \{a^i b^j c^k \mid i, j, k \geq 0$ and $i = j$ or $i = k\}$

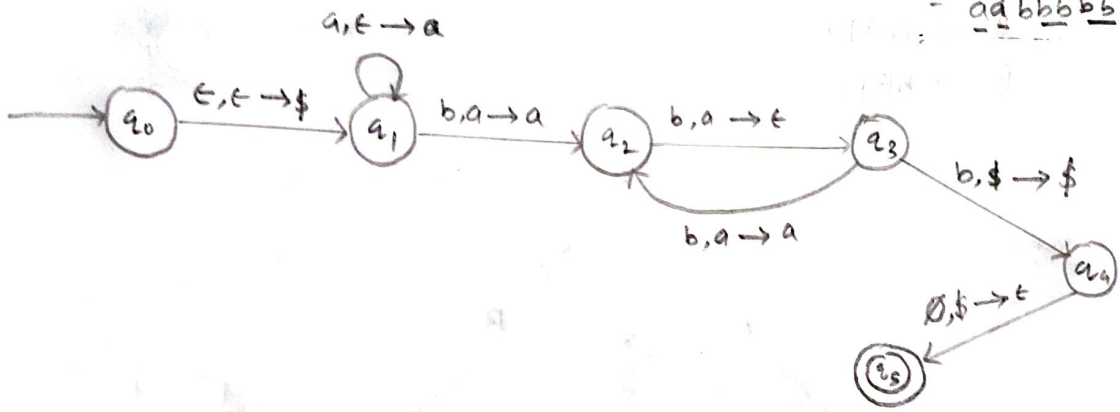


* $\{w | w = a^m b^n$ where $n = 2m\}$ Ex - $a^m b^{2m}$ - $a^1 b^2$ - abb

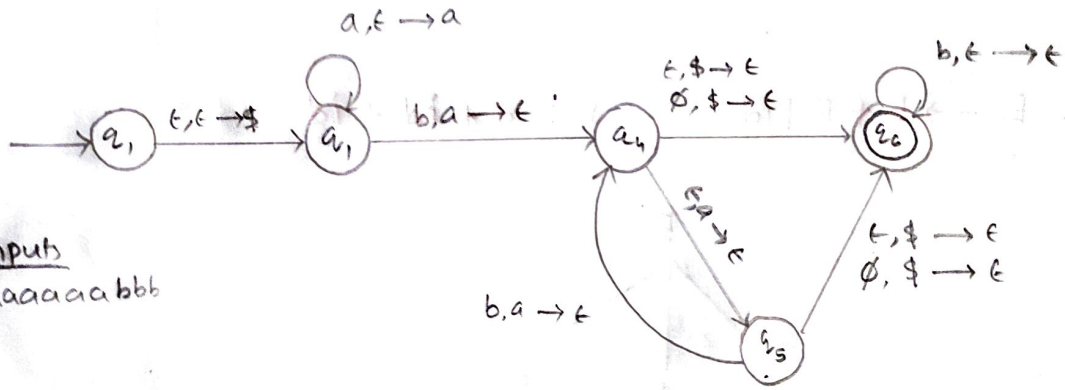


* $\{ w | w = a^m b^{2m+1} \}$.

Ex- abbb
aabbbbb



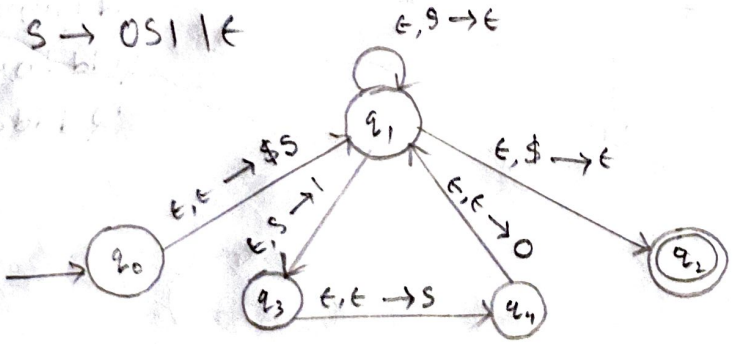
* $L = \{ w | w = a^m b^n \text{ and } M \leq 2N \}$



Inputs
aaaaaabbb

* $L = \{ w | w = 0^n 1^n \mid n \geq 0 \}$

CFG : $S \rightarrow 0S1 \mid \epsilon$



$S \rightarrow \epsilon$

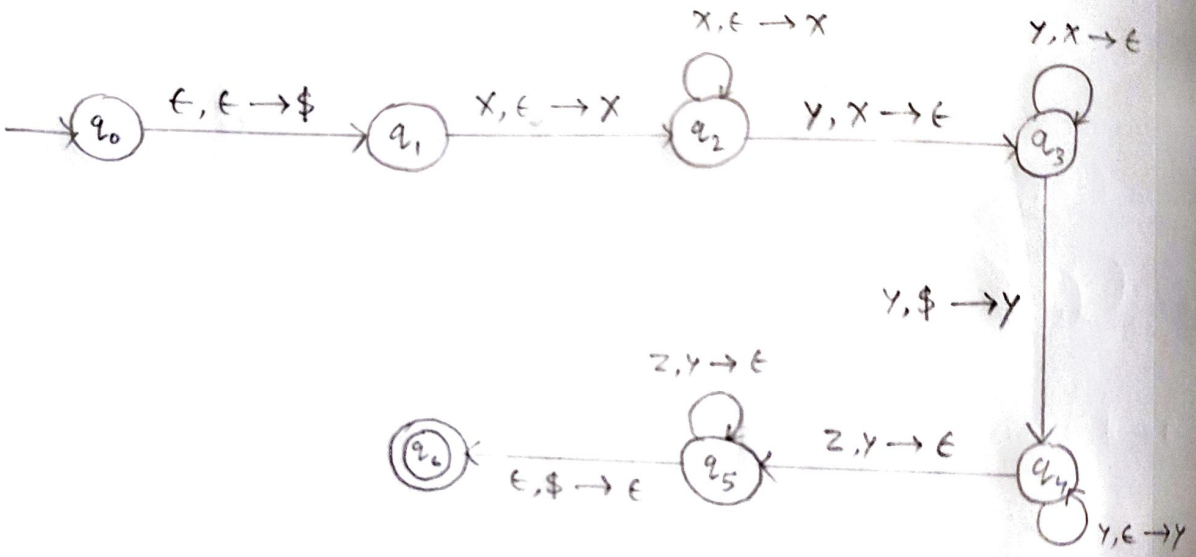
$S \rightarrow 0S1$

* Draw a pushdown automata for the following languages where $\Sigma = \{x, y, z\}$.

$A = \{x^a y^b z^c \mid \text{where } a = b - c \text{ and } a, b, c > 0\}$

$a = 2$, $b = 3$, $c = 1$ $\quad x^2y^3z^1 = xxyyyz$

$x^3y^4z^1 = xxxyyyyz$ $\quad x^1y^3z^2 = xyyyzz$

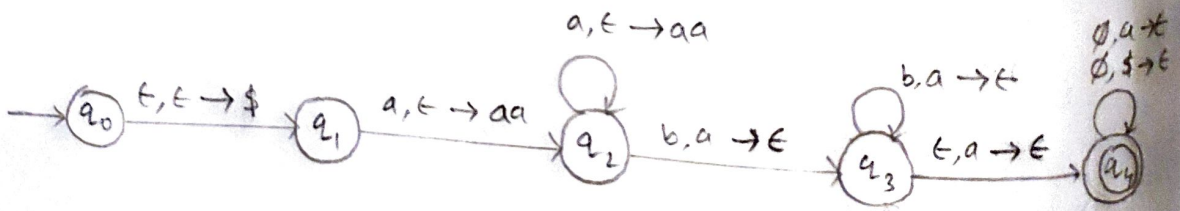


* $A = \{a^n b^m \mid \text{where } n < m < 2n\}$

$m = 3$, $n = 2$, $2n = 4$ $\quad 2 < 3 < 4$ $\quad a^2b^3$

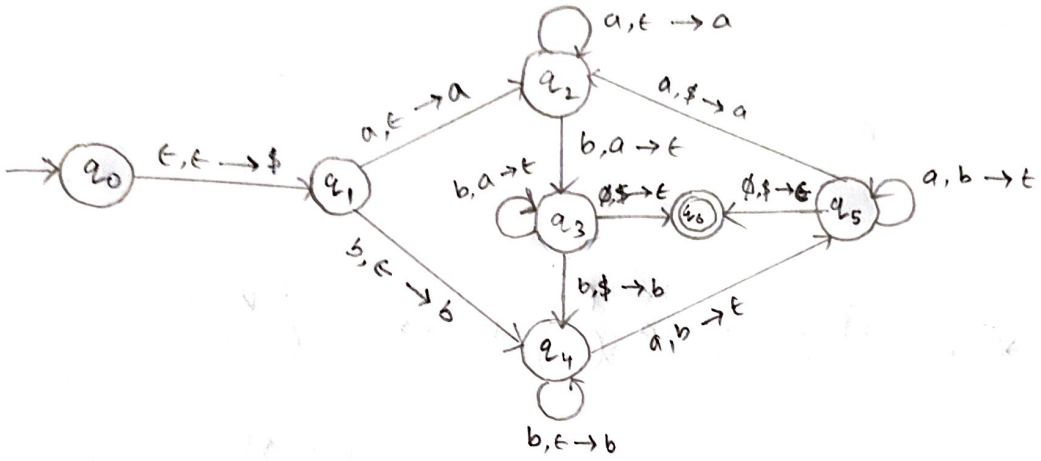
aa bbb

aaaa aa bbbbb



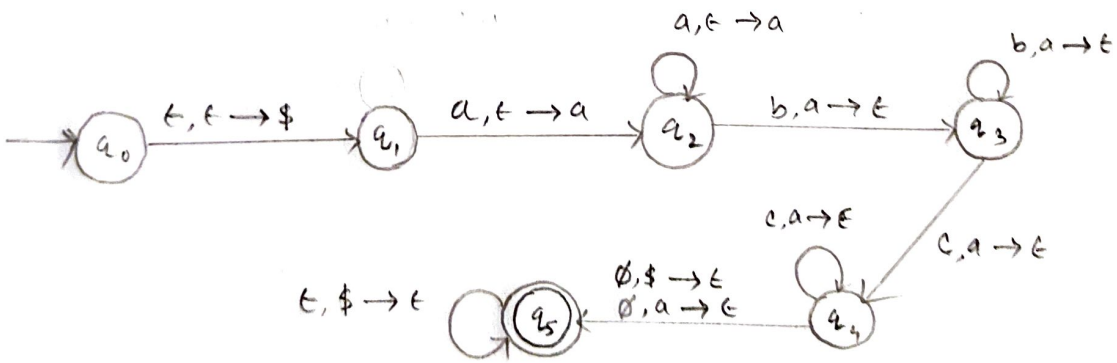
* $A = \{w | w = \{a,b\}^*$ and the number of a's and b's are equal $\}$

Ex- ab, aabb, abab, bbaa, ...



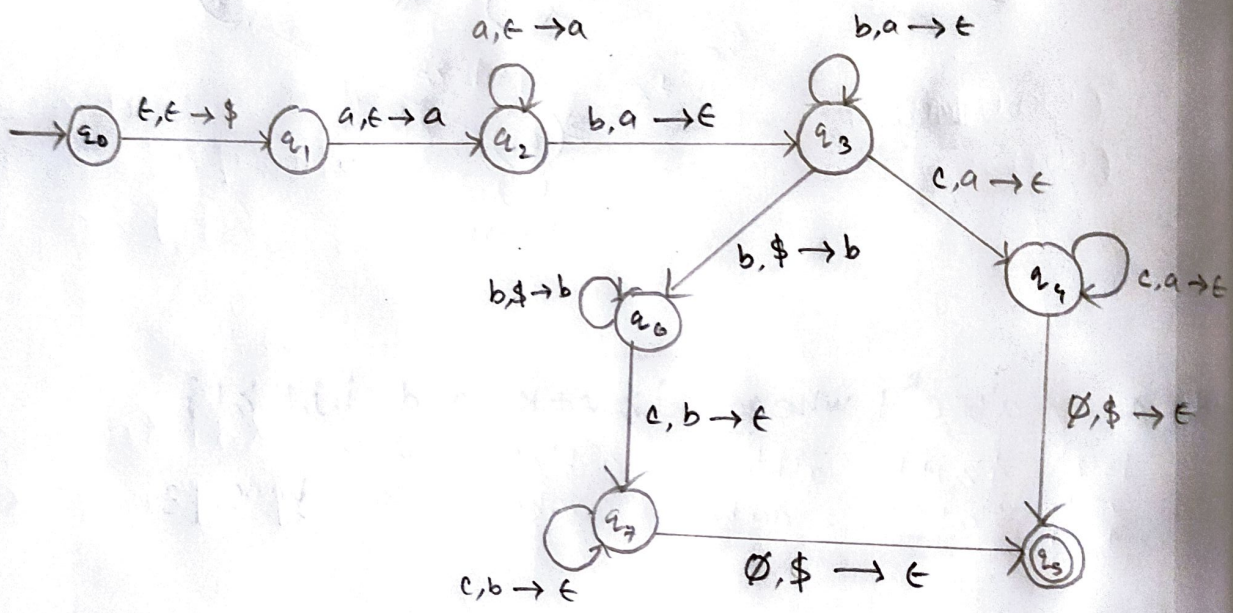
* $A = \{a^i b^j c^k \mid$ where $i \geq j+k$ and $i,j,k \geq 1\}$

$i=2$, $j=1$, $k=1$: $i \geq j+k$: $2 \geq 1+1$: $a^2b^1c^1 = aabc$ / $a^3b^1c^1 = aaabc$ / $a^5b^1c^3 = aaaaabccc$



* $A = \{ a^i b^j c^k \mid \text{where } i-k=j \text{ or } i+k=j \text{ and } i,j,k > 1 \}$ *

$i = k+j$	$i+k=j$	$i = k+j$	$1+k=j$
$3 = 1+2$	$1+2=3$	$2 = 1+1$	$1+1=2$
$a^3 b^2 c^1$	$a^1 b^3 = c^2$	aabc	abbc
=aaabbc	abbbcc		



Turing Machine

TURING MACHINE ⟶ Must reach Accept / Reject state
⟶ Halts once when reaches ...
⟶ Might loop forever

* WRITE FORMAL DEFINITION OF TURING MACHINE

→ A turing machine M is defined by 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ where,

→ Q finite set of states

→ Σ finite input alphabet (without blank symbol) "␣"

→ Γ finite tape alphabet with $\{␣\} \cup \Sigma \subseteq \Gamma$

⟶ δ the transition function

$$\delta : Q \setminus \{q_{accept}, q_{reject}\} \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}$$

→ q_0 start state & $\in Q$

→ q_{accept} state $\in Q$

→ q_{reject} state $\in Q$

→ $q_{accept} \neq q_{reject}$

* TESTING MEMBERSHIP IN $B = \{ w\#w \mid w \in \{0,1\}^* \}$

→ Computing Steps :

Ex- 011#011

1. Read (store) 1st alphabet, write X
2. (Find #) Move right until Read #, if ␣ found Reject
3. (skip X) Move right while Read X
4. If current alphabet = stored alphabet, ~~Move~~ Write

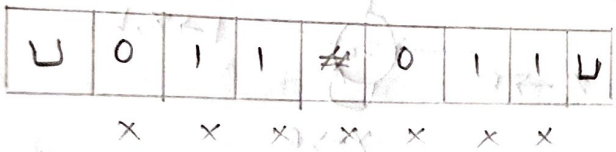
X, Move left to ⊔, else Reject

5. (skip X) Move right while Read X

6. Read (store) current alphabet. If alphabet is not # then continue to step 2.

7. If alphabet is #, (skip X) Move right while Read X.

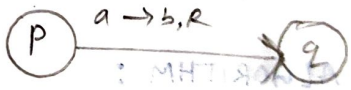
8. If current alphabet ⊔ then Accept otherwise Reject.



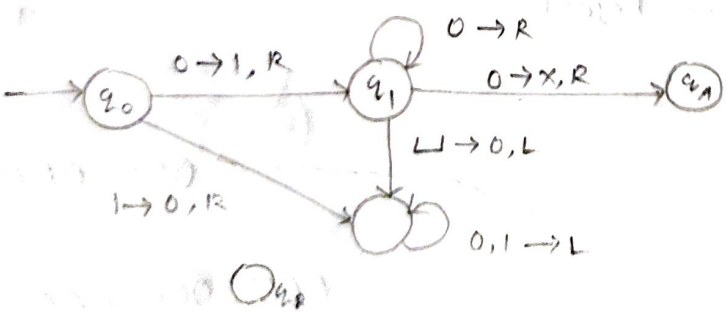
* TM TRANSITION FUNCTION

$\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}$

Current, Next, Move Left, Move Right



$\delta(P, a) = (Q, b, R)$



$\Sigma = \{\alpha, \beta, \gamma\}$

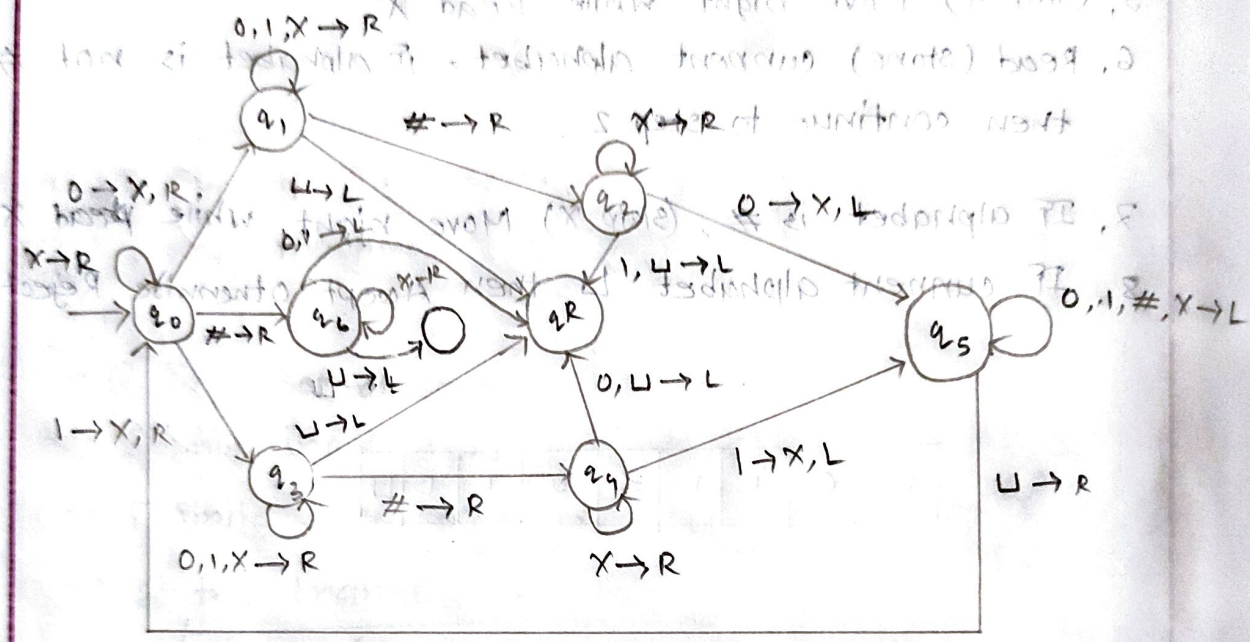
$D = \{L, R\}$

q_A = accept

q_R = Reject

* $\alpha \longrightarrow \alpha, D$ written as $\alpha \longrightarrow D$
* $\alpha \longrightarrow D$, $\beta \longrightarrow D$ written as $\alpha, \beta \longrightarrow D$

* STATE DIAGRAM OF TURING MACHINE : $B = \{w\#w \mid w \in \{0,1\}^*\}$



* TESTING MEMBERSHIP : $A = \{0^J \mid J = 2^N\}$

ALGORITHM :

If j = 0 then Reject
while j > 1 do
  if j mod 2 = 1 then Reject
  Otherwise j ← j/2
End do
if j = 1 then Accept

Ex→

2 | 16
2 | 8 — 0
2 | 4 — 0
2 | 2 — 0
2 | 1 — 0
0 — 1

2 | 12
2 | 6 — 0
2 | 3 — 0
1 — 1
0 — 1
X

00000000

00000000000

IMPLEMENTATION: (ACCEPT)

1. Replace the leftmost zero by ␣
2. Cross (x) every second 0 from left to right. Skip ␣ from left, cross next 0, skip next 0, cross next zero. Every skip must be followed by a cross.
3. If no 0 are found to skip and cross in step 3 (means 2^N of 0s), Accept
4. If there is no 0 to cross after skipping a 0 (means odd number of 0s) Reject.
5. Otherwise Move to the left ␣, continue step 2

0 0 0 0 0 0 0 0 ␣

␣ 0 0 0 0 0 0 0 ␣

␣ X 0 X 0 X 0 X 0 ␣

␣ X X X 0 X X X ␣

␣ X X X X X X X ␣

Accept

IMPLEMENTATION : (REJECT)

0 0 0 0 0 0 0 0 0 0 0 0 ␣

␣ 0 0 0 0 0 0 0 0 0 0 0 ␣

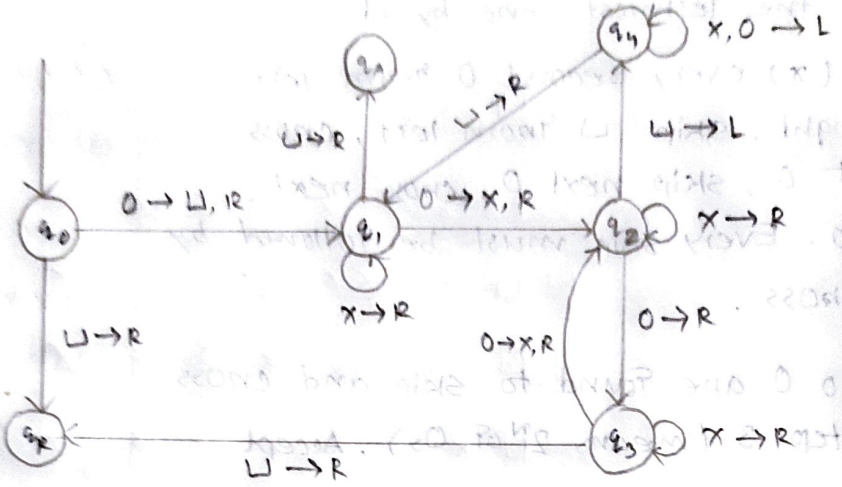
␣ X 0 X 0 X 0 X 0 X 0 X ␣

␣ X X X 0 X X X 0 X X X ␣

␣ X X X X X X X 0 X X X ␣

Reject

STATE DIAGRAM :



CONFIGURATION :

Let $u, v \in \Gamma^*$

$a, b, c \in \Gamma$

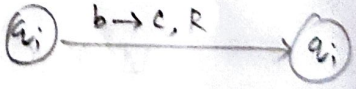
$q_i, q_j \in Q$ and

M be a TM with transition function δ

$ua q_i bv$ yields the configuration $uac q_i v$

if and only if

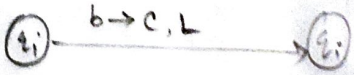
$\delta(q_i, b) = (q_j, c, R)$



$uaq_i bv$ yields the configuration $uq_j acv$

if and only if

$\delta(q_i, b) = (q_j, c, L)$



starting configuration on input = a..w

accepting

rejecting

Same soultion for the question
State names might differ.

* SIMULATE

1. Consider the following Turing machine. In the transition Function, q0 is the Start state, q_{accept} is the Accept State and q_{reject} is the Reject State. In the transition function '#' represents Blank Space. Now simulate this machine with '00000000' string and find whether it is accepted or rejected by the machine.

State	Input	Transition
q0	0	q1, # ,R
q0	x	q_{reject}, x, R
q0	#	q_r, #, R
q1	0	q3, x, R
q1	x	q1, x, R
q1	#	q_{accept}, #, R
q2	0	q2, 0, L
q2	x	q2, x, L
q2	#	q1, #, R
q3	0	q4, 0, R
q3	x	q3, x, R
q3	#	q2, #, L
q4	0	q3, x, R
q4	x	q4, x, R
q4	#	q_{reject}, #, R

starting configuration on input = $q_0 w$

accepting configuration : $u q_A v$

rejecting configuration : $u q_R v$

* SIMULATE THE STRING 00000000 (0^{2^3}) (58 Lines)

q_0 ⊔ 0 0 0 0 0 0 0 ⊔

⊔ q_1 0 0 0 0 0 0 0 ⊔

⊔ X q_2 0 0 0 0 0 0 ⊔

⊔ X 0 q_3 0 0 0 0 0 ⊔

⊔ X 0 X q_2 0 0 0 0 ⊔

⊔ X 0 X 0 q_3 0 0 0 ⊔

⊔ X 0 X 0 X q_2 0 0 ⊔

⊔ X 0 X 0 X 0 q_3 0 ⊔

⊔ X 0 X 0 X 0 X q_2 ⊔

⊔ X 0 X 0 X 0 q_4 X ⊔

⊔ X 0 X 0 X q_4 0 X ⊔

⊔ X 0 X 0 q_4 X 0 X ⊔

⊔ X 0 X q_4 0 X 0 X ⊔

⊔ X 0 q_4 X 0 X 0 X ⊔

⊔ X q_4 0 X 0 X 0 X ⊔

⊔ q_4 X 0 X 0 X 0 X ⊔

q_4 ⊔ X 0 X 0 X 0 X ⊔

⊔ q_1 x 0 x 0 x 0 x ⊔

⊔ x q_2 0 x 0 x 0 x ⊔

⊔ x x q_2 x 0 x 0 x ⊔

⊔ x x x q_2 0 x 0 x ⊔

⊔ x x x 0 q_3 x 0 x ⊔

⊔ x x x 0 x q_3 0 x ⊔

⊔ x x x 0 x x q_2 x ⊔

⊔ x x x 0 x x x q_2 ⊔

⊔ x x x 0 x x q_4 x ⊔

⊔ x x x 0 x q_4 x x ⊔

⊔ x x x 0 q_4 x x x ⊔

⊔ x x x q_4 0 x x x ⊔

⊔ x x q_4 x 0 x x x ⊔

⊔ x q_4 x x 0 x x x ⊔

⊔ q_4 x x x 0 x x x ⊔

q_4 ⊔ x x x 0 x x x ⊔

⊔ q_1 x x x 0 x x x ⊔

⊔ x q_1 x x 0 x x x ⊔

⊔ x x q_1 x 0 x x x ⊔

␣ x x x x x x x q_1 ␣

␣ x x x x x x x ␣ q_A

* Question-2 Simulate the string : aabbaa# (24 Lines)

q_1 a a b b a a #

q_1 a b b a a

a q_1 b b a a

a x q_2 b a a

a x b q_2 a a

a x b x q_3 a

a x b q_4 x a

a x q_4 b x a

a q_4 x b x a

q_4 a x b x a

q_4 # a x b x a #

q_5 a x b x a

x q_1 x b x a

x x q_1 b x a

x x x q_2 x a

x x x x q_2 a

x x x x x q_3

2. Consider a Turing machine with the following transitions:

State	Input	δ(State, Symbol, Move)
Q_0	a	Q_1,#,R
Q_0	#	Q_{accept}, #,R
Q_1	a	Q_1, a, R
Q_1	b	Q_2,x,R
Q1	x	Q_1, x, R
Q_2	a	Q_3,x,R
Q_2	b	Q_2, b, R
Q_2	x	Q_2, x, R
Q_3	a	Q_4,a,L
Q_3	#	Q_6,#,L
Q_4	a	Q_4,a,L
Q_4	b	Q_4,b,L
Q_4	x	Q_4,x,L
Q_4	#	Q_5,#,R
Q_5	a	Q_1,x,R
Q_5	x	Q_5,x,R
Q_6	x	Q_6,x,L
Q_6	#	Q_{accept}, #,R

Here 'Q_0' is the start state, 'Q_{Accept}' is the accept state. Trace the execution of this Turing machine with the string **aabbaa#** as input. Note that '#' represents the blank symbol.

x x x x Q_6 x

x x x Q_6 x x

x x Q_6 x x x

x Q_6 x x x x

Q_6 x x x x x

Q_6 # x x x x x #

Q_{accept} x x x x x

* Trace the execution of this Turing Machine below:
Given string baaab## (21 lines)

0 b a a a b #
4 a a a b
a 4 a a b
a a 4 a b
a a a 4 b
a a a b 4
a a a 5 b
a a 3 a #
a 3 a a #
3 a a a #
a a a #
1 a a #
a 1 a #
a a 1 #
a 2 a #
3 a # #
3 # a # #
0 a # #
1 # #

Transition Table

0 — a — (1, #, R)
0 — b — (4, #, R)
0 — # — (0, #, Y)
1 — a — (1, a, R)
1 — b — (1, b, R)
1 — # — (2, #, L)
2 — a — (3, #, L)
2 — # — (2, #, Y)
3 — a — (3, a, L)
3 — b — (3, b, L)
3 — # — (0, #, R)
4 — a — (4, a, R)
4 — b — (4, b, R)
4 — # — (5, #, L)
5 — b — (3, #, L)
5 — # — (5, #, Y)

2 # # #

(Accept)

Practice : Turing Machine

$T = \{a^i b^j c^k \mid i \times j = k \text{ and } i, j, k \geq 1\}$

ALGORITHM:

```
for each a in sequence:
    replace a with 'X'

    for each b in sequence:
        replace b with 'Y'

        for i from current index to length(sequence)
            if sequence[i] == 'c':
                replace sequence[i] with 'Z'
                break

    for each 'Y' in sequence
        replace 'Y' with b
```

IMPLEMENTATION LEVEL : DESCRIPTION

1. First replace an 'a' with X and move 1 step right. Then skip all 'a' and move right. If ␣ found reject

2. When reached first 'b' stop. Replace one 'b' with Y then move right skipping all intermediate 'b' and corresponding to the replaced b now replace one 'c' with 'Z' and move left.

3. Now move towards left skipping all 'z's and b in the way

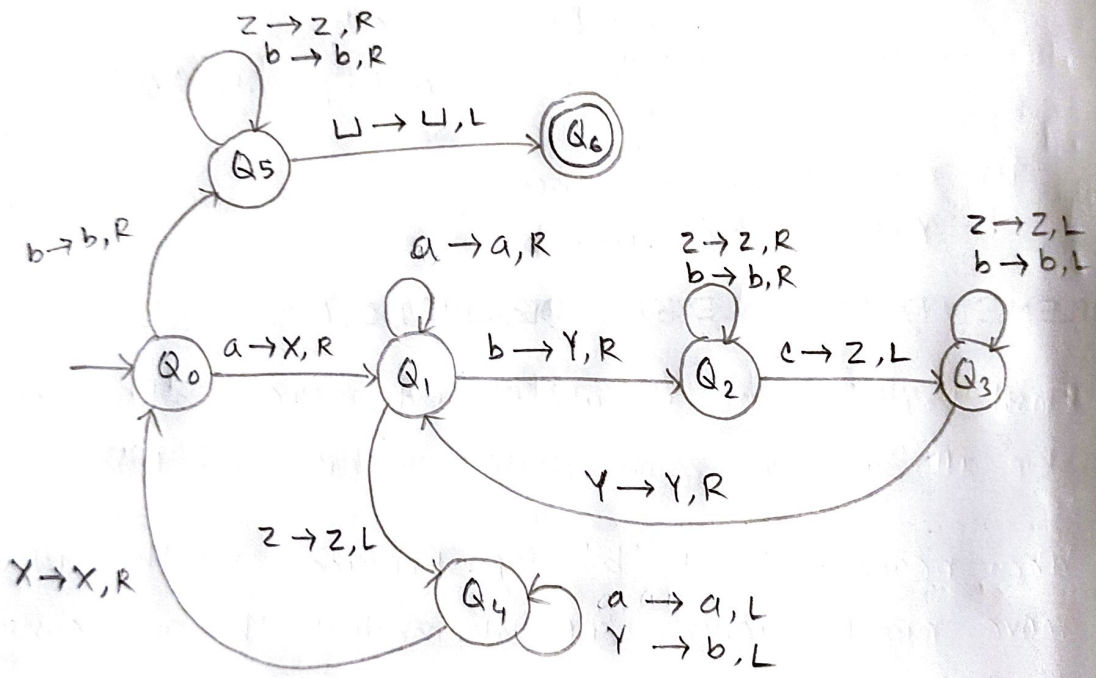
4. When pointer reaches most recent 'Y' move right

5. If the pointer is pointing at 'b' repeat all steps from 2 to 4, else if the pointer is pointing at 'z' then move towards left while replacing all 'Y' to 'b' and skipping all 'a's.

6. When the pointer comes to most recent 'X' move one step right.

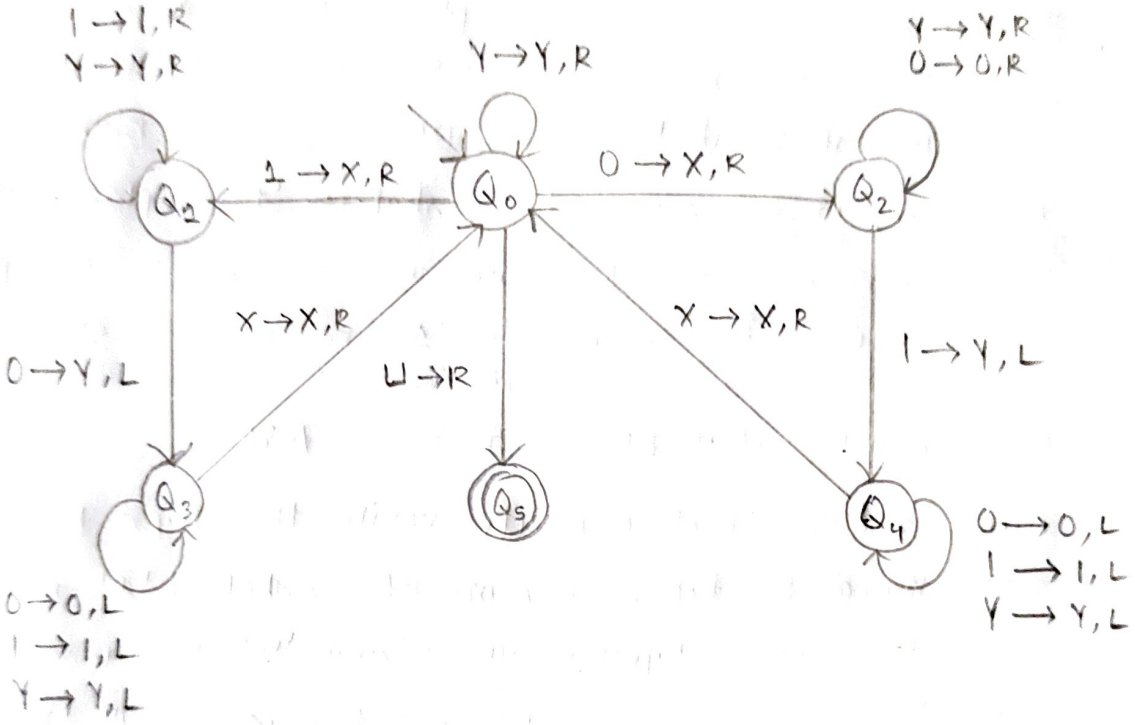
7. If the pointer is still pointing at 'a' then repeat all the above steps else if the pointer is at 'b' then move towards right skipping all 'b' and 'z'

8. When ⊔ is reached then move left. The string is Accepted.



aabbbccccc

* T = {WIW is a binary string and number of 0s and 1s are equal} 01101001



ALGORITHM:

for every 0 :

replace next 0 from left by X

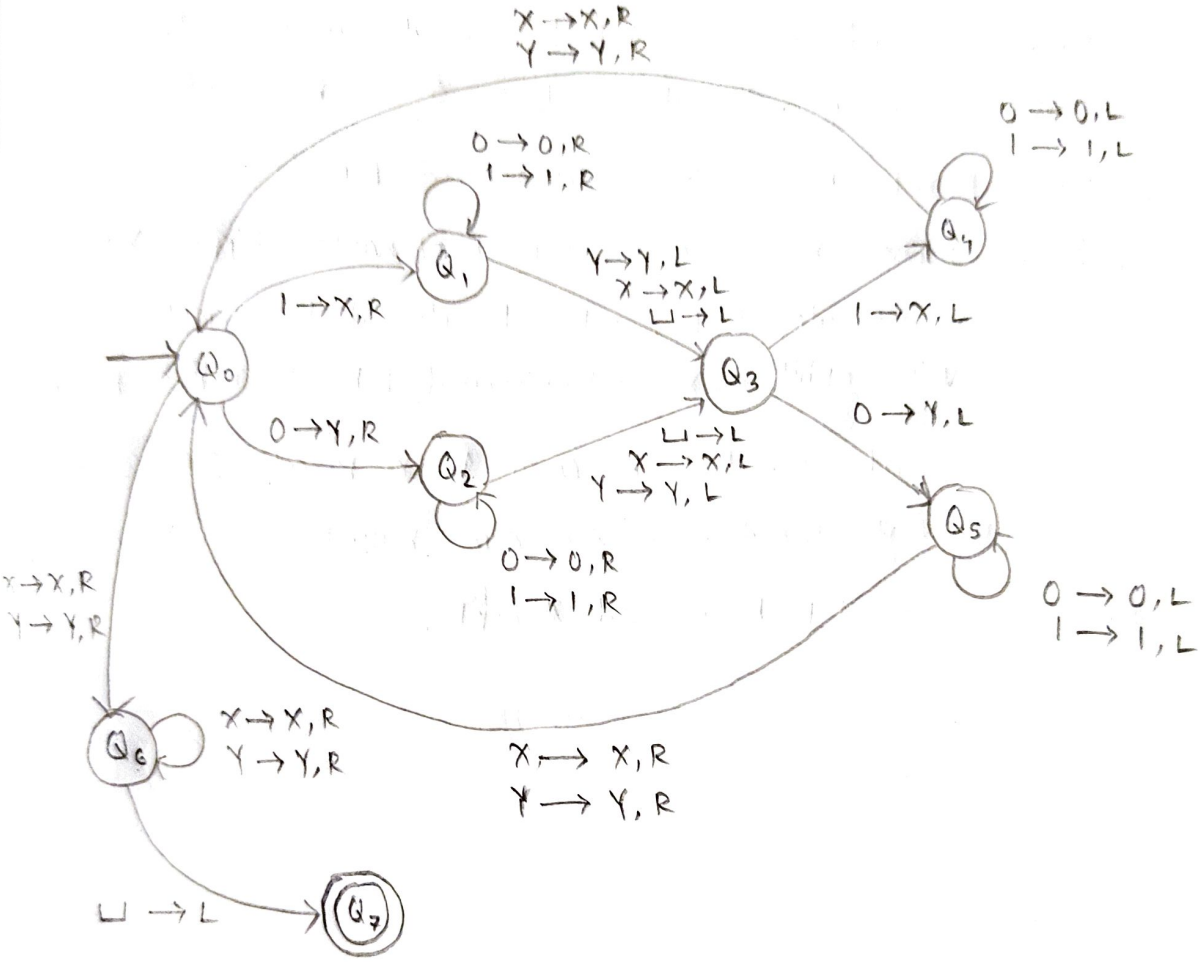
replace next 1 from left by Y

end for

Implementation Level : DESCRIPTION

1. Take the pointer to the beginning of tape head.
2. Enter a loop that repeats while tape head is not at the end of the tape.
3. If current symbol is '0' replace with 'X' and move one step right and enter a subloop. While the current symbol is '0' move one step right. However, If the current symbol is '1' replace with 'Y' and move one step right and enter a subloop. while current symbol is '1' move one step right.
4. Move right and if ⊔ found move left
5. Enter a loop that repeats while the tape head is not end of the tape. If current symbol is 'X' move one step right skipping all 'X' and 'Y'. If current symbol is 'Y' move one step right skipping all 'X' and 'Y'
6. If current symbol is '0' or '1' repeat steps 3 to 5
7. If ⊔ found current symbol move one step left and accept.

* $T = \{ ww^R \mid w \text{ is a binary string} \}$ 010010



ALGORITHM :

For each leftmost alphabet (means not X)

if alphabet is 0:

replace 0 with 'X'

if the rightmost alphabet is 0:

replace with 'X'

else alphabet is 1:

replace 1 with 'X'

if the rightmost alphabet is 1:

replace with 'X'

End for.

IMPLEMENTATION LEVEL : DESCRIPTION

1. (skip X) read first alphabet. If alphabet is 0 (stone) and move one step right. If alphabel is 1 (stone) and move right
2. Move Right (skipping X) until found ⊔.
3. If ⊔ found move one step left while (skipping X)
4. If current alphabet = stored alphabet mark 'X' and move left skipping X until found ⊔. Otherwise reject.
5. If current alphabet is 0 or 1 repeat 1,2,3,4 steps.
6. If current alphabet is 'X' move right until found ⊔
7. Move one step left and Accept.