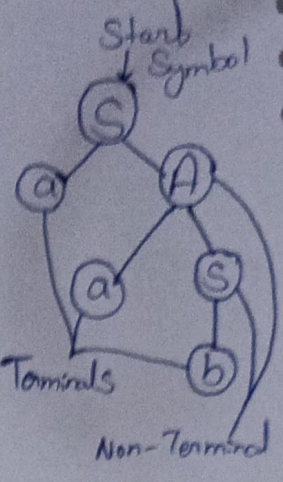
Unit - 3

Derivation Tree / Parse Tree

Eg. (for string aab) $S \to aA \mid b$
$A \to aS \mid b$



Ans:-

$S \to aA$ [S → aA]
$\to a\underline{aS}$ [A → aS]
$\to aab$ [S → b]

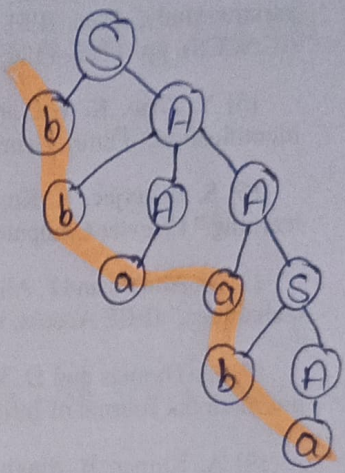
PYQ $S \to aB \mid bA$
$A \to a \mid aS \mid bAA$
$B \to b \mid bS \mid aBB$

Derive using Leftmost & Rightmost Derivation: ① bbaaba ② aaabbb
Draw parse tree of the same.

Ans:- ① bbaaba.

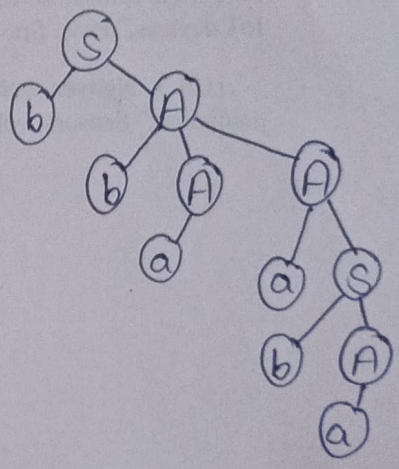
Left Most Derivation:

$S \to \cancel{aB}\ bA$ (S - bA)
$\to bb\underline{AA}$ (A - bAA)
$\to bba\underline{A}$ (A - a)
$\to bba\cancel{A}aS$ (A - aS)
$\to bbaabA$ (S - bA)
$\to bbaaba$ (A - a)



② RMD

$S \to bA$ (S - bA)
$\to bb\underline{AA}$ (A - bAA)
$\to bbAaS$ (A - aS)
$\to bbaaS$ (A - a)
$\to bbaabA$ (S - bA)
$\to bbaaba$ (A - a)

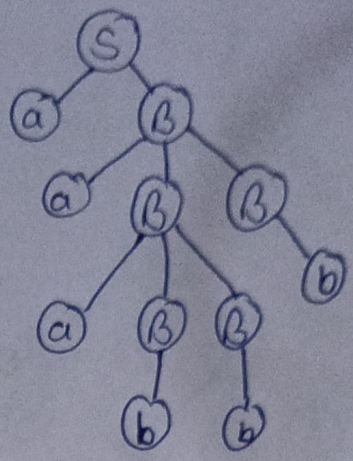


GNF ?

(2) aaabbb.

Leftmost Derivation (LMD)

$S \to aB$ (S - aB)
$\to aaBB$ (B - aBB)
$\to aaaBBB$ (B - aBB)
$\to aaabBB$ (B - b)
$\to aaabbB$ (B - b)
$\to aaabbb$ (B - b)



(RMD) Rightmost Derivation.

$S \to aB$ (S - aB)
$\to aaBB$ ~~(B - aBB)~~
$\to aaBb$ (B - b)
$\to aaaBBb$ (B - aBB)
$\to aaaBbb$ (B - b)
$\to aaabbb$ (B - b)

$S \to aB$ (S - aB)
$\to aaBB$ (B - aBB)
$\to aaBb$ (B - b)
$\to aaaBBb$ (B - aBB)
$\to aaaBbb$ (B - b)
$\to aaabbb$ (B - b)

* Ambiguous and Unambiguous Grammar.

- A grammar is ambiguous if there exists at least one string in the language that has :-
  - 2 or more diff. ~~Left~~ Leftmost Derivations, OR.
  - 2 or more diff. RMD, OR
  - 2 or more diff. parse Trees.

Context Free Language (CFL)

- A Language that can be generated by a Context Free Grammar. (CFG)
- CFL's are exactly the lang. accepted by a Pushdown Automata (PDA).
- CFL ⊃ Regular Languages. (all regular languages are CFLs, but not all CFLs are regular).

Q.a (a) Write a grammar G for generating the Language.

(i) L={w belongs to {a,b}* | w is on even length palindrome with |w|>0}

(ii) Set of odd length strings in {0,1}* with middle symbol ('1').

Ans:- (i)

Terminal = a,b , Palindrome = aba, bab, abba, baab

Even Length Palindrome = baab, abba, aa, bb

$S \to aa$ $S \to bb$ $S \to abba$ $S \to baab$

$S \to aSa$ $S \to bSb$

$$\boxed{S \to aa \mid bb \mid aSa \mid bSb}$$

(ii)

Terminals = 0,1

condition:- odd length strings with middle symbol '1'.

1
010
011
110
111

$S \to 1$

$S \to$ ~~010~~ $010 \to 0S0$

$S \to$ ~~011~~ $011 \to 0S1$

$S \to 110 \to 1S0$

$S \to 111 \to 1S1$

$$\boxed{S \to 1 \mid 0S0 \mid 0S1 \mid 1S0 \mid 1S1}$$

Simplification of CFG 1:20

- **Eliminating ε-productions**

Steps to eliminate ε-productions.

1. Find Nullable Non-Terminals.
2. Modify Productions.
3. Remove All ε-Productions
4. Handle Start Symbol.

- Consider ε from XYX one by one we got

XY	YX	XX

- Then consider ε from these XY | YX we got Y | X
- This is how we eliminates Epsilon.

Example:- Eliminate ε-productions.

$S \to XYX$
$X \to 0X \mid \varepsilon$
$Y \to 1Y \mid \varepsilon$

- Nullable ~~variables~~ variables - S, X, Y

$S \to X4X \mid YX \mid XY \mid Y \mid XX \mid X$
$X \to 0X \mid 0$
$Y \to 1Y \mid 1$

- **Eliminating Unit Productions.**

Steps to eliminate unit productions.

1. Identify all unit pairs (A, B)
2. Replace unit productions.
3. Remove all unit productions.

$S \to 0X \mid 1Y \mid 01$
$X \to 0S \mid 00$
$Y \to 1 \mid 0S \mid 00$
$Z \to 01$

Eg. Remove Unit Production.

- $S \to 0X \mid 1Y \mid Z$
  $X \to 0S \mid 00$
  $Y \to 1 \mid X$
  $Z \to 01$

Ans:- Replacing Z in S & X in Y. from their initial values.

Eliminating Useless Symbols.

Two types of Useless Symbols.

① Non-generating symbols - Symbols that can never produce Terminals.

② Non-reachable symbols - Symbols that can never be reached from the start symbol.

• Eg :- $S \to aA \mid BC$
$A \to b \mid bB$
$B \to aB \mid \varepsilon$
$C \to aC \mid D$
$D \to bD$

Ans :- Remove C & D because these are recursive that is $D \to bD \to bbD \to bbbD$.

So,
$S \to aA$
$A \to b \mid bB$
$B \to aB \mid \varepsilon$

Didn't write BC is S because it is useless doesn't make any difference if I removed it.

① Simplify the flluing Grommar.

$S \to OAO \mid 1B1 \mid BB$
$A \to C$
$B \to S \mid A$
$C \to S \mid \varepsilon$

Ans :-

Step 1 - Removal of ε production.

Nullable set = {C, A, B, S}

$S \to OAO \mid 1B1 \mid BB \mid 00 \mid 11 \mid B$
$A \to C$
$B \to S \mid A$
$C \to S$

Step 2 - Removal of Unit Production

Here, $S \to B$
$A \to C$
$B \to S \quad B \to A$
$C \to S$ ... are unit production

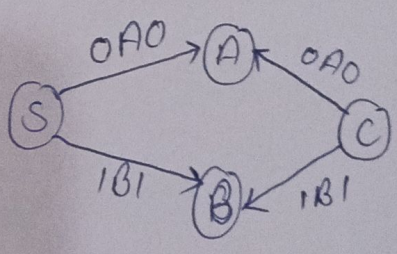
$S \to B \to A \to C \to S$ (A circuit of Unit Production is formed.)

$S \to OAO \mid 1B1 \mid 00 \mid 11 \mid BB$
$A \to OAO \mid 1B1 \mid 00 \mid 11 \mid BB$
$B \to OAO \mid 1B1 \mid 00 \mid 11 \mid BB$
$C \to OAO \mid 1B1 \mid 00 \mid 11 \mid BB.$

Step 3 - Removal of Useless Symbols. (a) Non - Reachable. (b) Non - Generating.

- All are generating.
∴ C is non reachable.

$S \to OAO \mid 1B1 \mid 00 \mid 11 \mid BB$
$A \to OAO \mid 1B1 \mid 00 \mid 11 \mid BB$
$B \to OAO \mid 1B1 \mid 00 \mid 11 \mid BB.$



∴ Simplified Grommar.

* Normal forms *

Both should be Non Terminals so we take new variable Z

(1) Chomsky Normal form (CNF) — eg. $A \to aB$, $Z \to a$ } $A \to ZB$, $Z \to a$

(2) Greibach Normal form (GNF)

• **CNF** - A standardized form of CFG where every production rule follows specific patterns to simplify the grammar & make parsing & analysis easier.

Q. Convert the following grammar to Chomsky Normal form.

$S \to a \mid aA \mid B$

$A \to aBB \mid \varepsilon$

$B \to Aa \mid b$

Ans:-

(1) Removal of ε productions.

$S \to a \mid aA \mid B$

$A \to aBB$

$B \to Aa \mid a \mid b$

(2) Removal of Unit Production.

$S \to a \mid aA \mid Aa \mid b$ — (Substituted B in S)

$A \to aBB$

$B \to Aa \mid a \mid b$

(3) Removal of Useless Symbol.

- No useless or No unreachable symbols.

(4) Chomsky Normal form

$S \to \breve{a} \mid \dot{a}\underset{\times}{A} \mid \underset{\times}{Aa} \mid \breve{b}$ — (Highlighted are wrong forms having so to correct them i.e. we will have to put both the Non-Terminals in it, so we will take $X \to a$)

$A \to aB\breve{B}$

$B \to \underset{\times}{Aa} \mid \breve{a} \mid \breve{b}$

$X \to a$

Substituting $X \to a$ ... (Jithe pan aahe fakt tithech changes karaych, direct 'a' la change karaych nahi fakt Non-Terminals aale pahije nit form)

$S \to a \mid XA \mid AX \mid b$

$A \to XBB$ ——— (R.H.S should be atmost 2, but ithe 3 ahe so we need to replace BB)

$B \to AX \mid a \mid b$

$X \to a$

$Y \to BB$

$S \to a \mid XA \mid AX \mid b$

$A \to XY$

$B \to AX \mid a \mid b$

$X \to a$

$Y \to BB$

Rules - ① If there is single Terminals like a, b then it is CNF.

② If there is Non-Terminals with Terminals like aA then we have take new Non-Terminal & replace it Bcoz There should be 2 non-Terminals

③ There should not be 3 non-Terminals if there is XBB we need to replace it.

- Decision Properties of (CFL)

(A) Decidable problems (possible for CFLs)

1. Membership ~~Regt~~ Problem.
2. Emptiness Problem.
3. finiteness problem..

(B) Undecidable problems (Impossible for CFLs)

1. University Problem..
2. Equivalence problem.
3. Inclusion problem.
4. Regularity problem.

Unit 4 Push Down Automata

Stack → LIFO

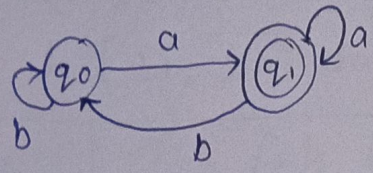
- Finite Automata + Stack = PDA.
- Finite Automata(FA) wasn't having any memory to store & retrieve symbols as PDA has memory as stack.

Key points -

- Stacks allows the PDA to store & retrieve symbols, enabling it to handle nested structures (like parentheses, recursion, etc.) which F.A. cannot.
- PDA is more powerful than a F.A. but less powerfull than a ~~Turn~~ Turing Machine.

Eg:- $L = \omega \in \{a,b\}^* \mid \omega$ ends with a.

aabbbaba , abbbak ×



- **Acceptance Methods :-**

① **By final State** - if it reaches an accepting state.

② **By Empty State** - if the stack becomes empty after reading the entire input.

PDA is formally defined as a 7-tuple.

$M = \{Q$, - set of states (q_0, q_1, q_2), f - final state $(q_f)\}$

Σ - Input alphabets $\{a,b\}$,

Γ - Stack alphabet $\{Z_0, a\}$

δ - Transition., $\rightarrow \delta : Q \times (\Sigma \cup (\varepsilon)) \times \Gamma \rightarrow P(Q \times \Gamma^*)$

q_0 - Initial state,

Z_0 - Initial Stack Top,

• Instantaneous Description of PDA.

- (a, w, α) Presents current status of the machine.

$a \rightarrow$ current state, $w \rightarrow$ Input (operations like push & pop)

$\alpha \rightarrow$ Current ~~Input~~ Stack Top.

Eg. Consider the PDA for Lang. $L = \{a^n b^n \mid n \geq 1\}$.

$w = aaabbb$

Ans -> for instantaneous of PDA,

$\vdash (q_0, aaabbb, Z_0)$

(state, I/P, Stack Top)

$\vdash (q_0, aabbb, aZ_0)$

$\vdash (q_0, abbb, aaZ_0)$

$\vdash (q_0, bbb, aaaZ_0)$

$\vdash (q_1, bb, aaZ_0)$

$\vdash (q_1, b, aZ_0)$

$\vdash (q_1, \varepsilon, Z_0)$

$\vdash (q_F, \varepsilon)$

Features	Finite Automata (FA)	Pushdown Automata (PDA)
Memory	Finite memory only (states).	Infinite memory using stack.
Storage Structure	No external storage.	Uses stack for storing symbols.
Acceptance	Accepts by final state only.	Accepts by final state or empty stack
Language Accepted.	Regular Languages (RL)	Context-free Language (CFL)
Components	5-tuples $(Q, \Sigma, \delta, q_0, F)$	7-tuples: $(Q, \Sigma, \delta, \Gamma, q_0, Z_0, F)$
Transition function.	$(\delta : Q \times \Sigma \rightarrow Q)$	$(\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow P(Q \times \Gamma^*))$
Power	Less powerful	More powerful
Eg. Language	$(L = a^m b)$	$(L = \{a^n b^n \mid n \geq 1\})$
Representation	Transition diagram with states & labelled arrows.	Transition diagram with stack operations (push, pop, no-change)

Q. Find transition rules of PDA for accepting a lang. $L = \{w \in \{a,b\}^* \mid w \text{ is of the } a^n b^n \text{ with } n \geq 1\}$ through both empty stack & final state & demonstrates the stack operation for the string aaabbb. (9 marks.)

Ans:-

$M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$

$Q = \{q_0, q_1, q_F\}$

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

$\Gamma = \{Z_0, a\}$

$F = \{q_F\}$

① Transition functions (δ)

$\delta(q_0, a, Z_0) = (q_0, aZ_0)$

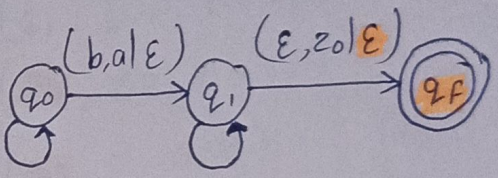
$\delta(q_0, a, a) = (q_0, aa)$

$\delta(q_0, b, a) = (q_1, \varepsilon)$

$\delta(q_1, b, a) = (q_1, \varepsilon)$

$\delta(q_1, \varepsilon, Z_0) = (q_F, \varepsilon)$

② Transition Diagram.



$(a, Z_0 | aZ_0)$ $(b, a | \varepsilon)$

$(a, a | aa)$

③ String aaabbb.

$\vdash (q_0, aaabbb, Z_0)$

$\vdash (q_0, aabbb, aZ_0)$

$\vdash (q_0, abbb, aaZ_0)$

$\vdash (q_0, bbb, aaaZ_0)$

$\vdash (q_1, bb, aaZ_0)$

$\vdash (q_1, b, aZ_0)$

$\vdash (q_1, \varepsilon, Z_0)$

$\vdash (q_F, \varepsilon)$

Q. Design PDA Lang. $\{0^n 1^m 0^n \mid m,n \geq 1\}$.
input string 00111100.

Ans:-

$$M = \{Q, \Sigma, \Gamma, q_0, \delta, z_0, F\}$$

$$Q = \{q_0, q_1, q_2, q_F\}$$
(q_0: 0^n, q_1: 1^m, q_2: 0^n)

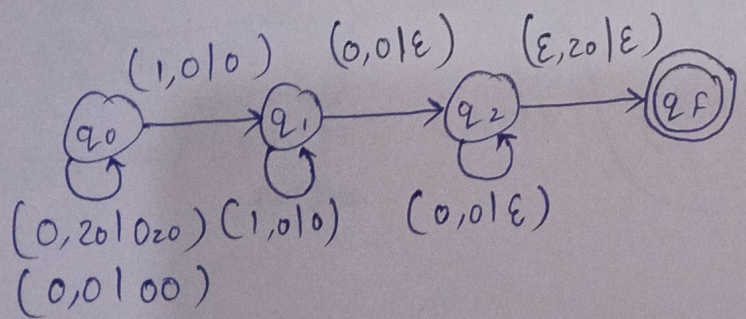
$$\Sigma = \{0,1\} \qquad F = \{q_F\}$$

$$\Gamma = \{z_0, 0\}$$

① Transition Fucs.

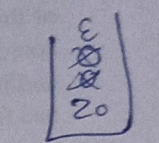
$\delta(q_0, 0, z_0) = (q_0, 0z_0)$
$\delta(q_0, 0, 0) = (q_0, 00)$
$\delta(q_0, 1, 0) = (q_1, 0)$
$\delta(q_1, 1, 0) = (q_1, 0)$
$\delta(q_1, 1, 0) = (q_1, 0)$
$\delta(q_1, 1, 0) = (q_1, 0)$
$\delta(q_1, 0, 0) = (q_2, \varepsilon)$
$\delta(q_2, 0, 0) = (q_2, \varepsilon)$
$\delta(q_2, \varepsilon, z_0) = (q_F, \varepsilon)$

② Diagram.



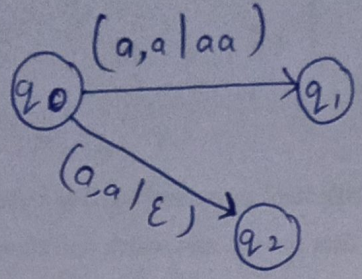
③ String 00111100.

$\vdash (q_0, 00111100, z_0)$
$\vdash (q_0, 0111100, 0z_0)$
$\vdash (q_0, 111100, 00z_0)$
$\vdash (q_1, 11100, 00z_0)$
$\vdash (q_1, 1100, 00z_0)$
$\vdash (q_1, 100, 00z_0)$
$\vdash (q_2, 00, 00z_0)$
$\vdash (q_2, 0, 0z_0)$
$\vdash (q_2, \varepsilon, z_0)$
$\vdash (q_F, \varepsilon)$

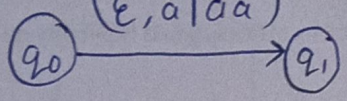


(Theory) **Non-Deterministic PDA (NPDA)**

- A type of PDA which has i/p symbol, current state, top of stack & automation have more than one moves.
- Eg.



- If I/p is abaa & I want to change the state, without reading any i/p symbol.
- So,



(ab will be read as ε & a is stack top. So when stack top becomes a the o/p will be aa to the next state)

- There are more Ques. in Theory about NPDA in PYQs

Q. What is NPDA? Construct a NPDA for the set of all strings over {a,b} with odd length palindrome:

Ans:-

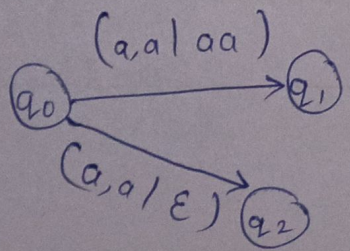
- NPDA is a type of PDA in which for a given input symbol, current state, and top of stack, the automaton may have more than one possible move.
- formal definition of NPDA :-

  $M = \{Q, \Sigma, \Gamma, \delta, q_0, z_0, F\}$

Characteristics

- NPDA can make choices both multiple transitions.
- NPDA can use ε-moves to change the state or modify the stack without reading any i/p symbol.
- NPDA is more powerful than DPDA's and some CFL can only be accepted by NPDA and not by DPDAs.

- Eg:-



Ans:- continue.

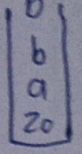
$M = \{Q, \Sigma, \delta, \Gamma, q_0, z_0, F\}$

$Q = \{q_0, q_1, q_F\}$

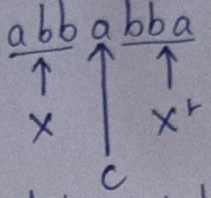
$F = \{q_F\}$

$\Gamma = \{a, b, z_0\}$

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



odd length palindrome :-



(If this is palindrome but xcx^r trick can be used to get the answer)

Transition Functions.

$\delta(q_0, a, z_0) = (q_0, az_0)$

~~$\delta(q_0, b, a) = (q_0, ba)$~~

~~$\delta(q_0, b, b) = (q_0, bb)$~~

$\delta(q_0, a, a) = (q_0, aa)$

$\delta(q_0, a, b) = (q_0, ab)$

$\delta(q_0, b, z_0) = (q_0, bz_0)$

$\delta(q_0, b, a) = (q_0, ba)$

$\delta(q_0, b, b) = (q_0, bb)$

} (Push abb in stack)

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

$\delta(q_0, a, b) = (q_1, b)$

$\delta(q_0, a, z_0) = (q_1, z_0)$

$\delta(q_0, b, b) = (q_1, b)$

$\delta(q_0, b, a) = (q_1, a)$

$\delta(q_0, b, z_0) = (q_1, z_0)$

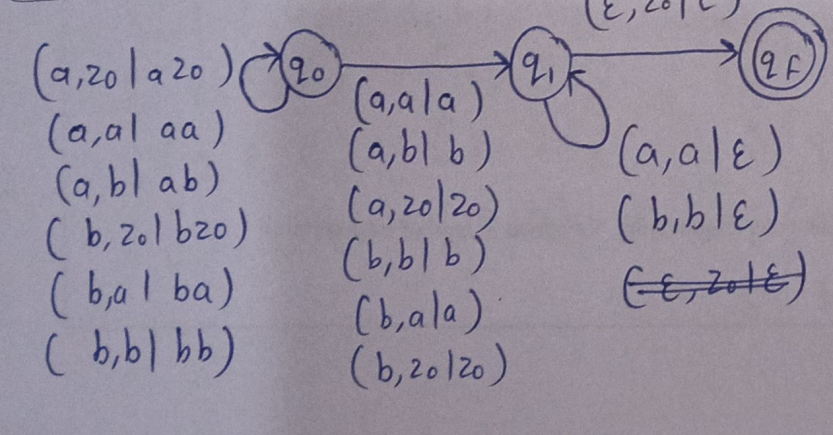
(Ignore a in the middle to make sure the ~~o~~ push doesn't change)

$\delta(q_1, a, a) = (q_1, \varepsilon)$

$\delta(q_1, b, b) = (q_1, \varepsilon)$

$\delta(q_1, \varepsilon, z_0) = (q_F, \varepsilon)$

} (Pop bba from stack)



$\vdash (q_0, abbabba, z_0)$

$\vdash (q_0, bbabba, az_0)$

$\vdash (q_0, babba, baz_0)$

$\vdash (q_0, abba, bbaz_0)$

$\vdash (q_0, ~~b~~ba, ~~b~~baz_0)$

$\vdash (q_1, ~~ba~~, ~~ba~~z_0)$

$\vdash (q_1, \alpha, \alpha z_0)$

$\vdash (q_{01}, \varepsilon, z_0)$

$\vdash (q_F, \varepsilon)$

* Conversion of CFG to PDA.

Step 1 - Remove all ε productions.

step 2 - Remove unit Productions.

step 3 - ~~Rem~~ Convert CFG to GNF.

step 4 - Construct PDA from the Grammar.

step 4.1 - PDA Constructions.

step 4.2 - Define Transitions.

i) - Variable Replacement. — $\delta(q, \varepsilon, A) = (q, \alpha)$

ii) - Terminal Matching. — $\delta(q_0, a, a) = (q_0, \varepsilon)$

Eg. Q. Construct PDA for CFG, test whether 010^4 is acceptable by this PDA.

$S \to OBB$

$B \to OS \mid 1S \mid 0$

$S \to OBB$
$B \to OS$
$B \to 1S$
$B \to 0$

Ans :-

(1) No ε productions.
(2) No unit productions.
(3) Already in GNF form.

Transition functions (δ)

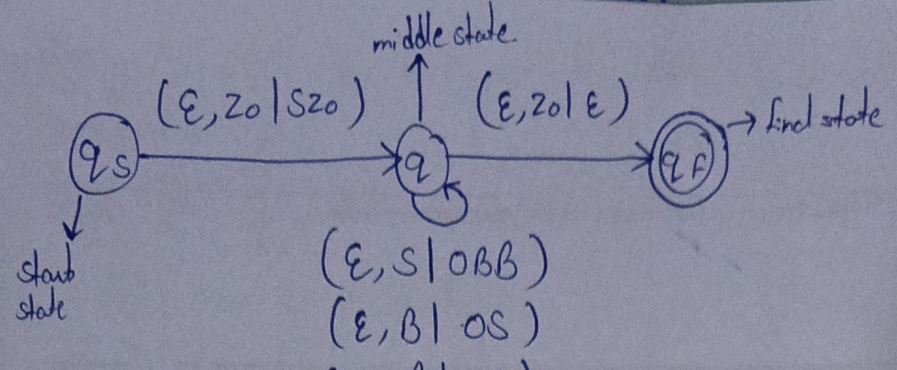
$\delta(q_s, \varepsilon, z_0) = (q, Sz_0)$

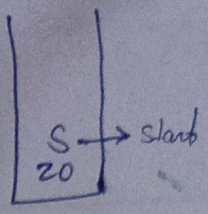
(i) Variable Replacement : $\delta(q, \varepsilon, A) = (q, \alpha)$

$\delta(q, \varepsilon, S) = (q, OBB)$
$\delta(q, \varepsilon, B) = (q, OS)$
$\delta(q, \varepsilon, B) = (q, 1S)$
$\delta(q, \varepsilon, B) = (q, 0)$

(ii) Terminal Matching : $\delta(q, a, a) = (q, \varepsilon)$

$\delta(q, 0, 0) = (q, \varepsilon)$
$\delta(q, 1, 1) = (q, \varepsilon)$
$\delta(q, \varepsilon, z_0) = (q_f, \varepsilon)$





Test : $01^0 = 010000$

$\vdash (q_s, 010000, z_0)$ — (Here input which is 010000 doesn't change like PDA. Here when the i/p matches the string it is popped.)

$\vdash (q, 010000, \underline{S} z_0)$

$\vdash (q, \otimes 10000, \underline{\otimes BB}\, z_0)$ – (Substitute $S \to 0BB$)

$\vdash (q, 10000, \underline{B} B z_0)$

$\vdash (q, *0000, \underline{*S} B z_0)$ – $(B \to 1S)$

$\vdash (q, 0000, \underline{S} B z_0)$

$\vdash (q, \otimes 000, \underline{\otimes BB} B z_0)$ – $(S \to 0BB)$

$\vdash (q, 000, \underline{B} B B z_0)$

$\vdash (q, \otimes 00, \otimes B B z_0)$ – $(B \to 0)$

$\vdash (q, \otimes 0, \otimes B z_0)$ – $(B \to 0)$

$\vdash (q, 0, B z_0)$

$\vdash (q, 0, 0 z_0)$ – $(B \to 0)$

$\vdash (q, \varepsilon, z_0)$

$\vdash (q_F, \varepsilon)$

* Conversion of PDA to CFG

step1 - Find the Start Symbol.

step2 - Find the Terminals.

step3 - Find the Non-Terminals.

step4 - Find the Production Rules.

(a) Push operation.

(b) Pop operation.

(c) Read / No stack Change.

step5 - Acceptance.

Non-Terminals

(S, V, T, P)

S: Start, T: Terminals, P: Production Rule.

Q. Construct a context free Grammar which accepts N(A), where $A = (\{q0, q1\}, \{0, 1\}, \{Z0, Z\}, \delta, q0, Z0, F)$ where δ is given by: Q, Σ, Γ

$\delta(q0, 1, Z0) = (q0, ZZ0)$ — Push

$\delta(q0, \varepsilon, Z0) = (q0, \varepsilon)$ — Pop

$\delta(q0, 1, Z) = (q0, ZZ)$ — Push

$\delta(q0, 0, Z) = (q1, Z)$ — Read

$\delta(q01, 1, Z) = (q1, \varepsilon)$ — pop

$\delta(q01, 0, Z0) = (q0, Z0)$ — Read

Ans:-

① Start Symbol

$S \to (q0, Z0, q0)$

$S \to (q0, Z0, q1)$

② Terminals

$T = \varepsilon$

$T = \{0, 1\}$

③ Non-Terminds (V)

$A \to (q0, Z0, q0)$	$E \to (q0, Z, q0)$
$B \to (q0, Z0, q1)$	$F \to (q0, Z, q1)$
$C \to (q01, Z0, q0)$	$G \to (q1, Z, q0)$
$D \to (q1, Z0, q1)$	$H \to (q1, Z, q1)$

(4) Production Rule.

$(q_0, 1, z_0) = (q_0, 2z_0)$ — Push

$(q_0, z_0, q_0) \rightarrow 1\,[q_0, 2, q_0]\,[q_0, z_0, q_0]$ (E, A)

$(q_0, z_0, q_0) \rightarrow 1\,[q_0, 2, q_1]\,[q_1, z_0, q_0]$ (F, C)

$(q_0, z_0, q_1) \rightarrow 1\,[q_0, 2, q_0]\,[q_0, z_0, q_1]$ (E, B)

$(q_0, z_0, q_1) \rightarrow 1\,[q_0, 2, q_1]\,[q_1, z_0, q_1]$ (F, 0)

$A \rightarrow 1EA \mid 1FC$

$B \rightarrow 1EB \mid 1F0$

$\delta(q_0, \varepsilon, z_0) = (q_0, \varepsilon)$ — Pop

$[q_0, z_0, q_0] \rightarrow \varepsilon$

$A \rightarrow \varepsilon$

$\delta(q_0, 1, z) = (q_0, zz)$ — Push

$[q_0, z, q_0] \rightarrow 1\,[q_0, z, q_0]\,[q_0, z, q_0]$ | $E \rightarrow 1EE$

$[q_0, z, q_0] \rightarrow 1\,[q_0, z, q_1]\,[q_1, z, q_0]$ | $E \rightarrow 1FG$

$[q_0, z, q_1] \rightarrow 1\,[q_0, z, q_0]\,[q_0, z, q_1]$ | $F \rightarrow 1EF$

$[q_0, z, q_1] \rightarrow 1\,[q_0, z, q_1]\,[q_1, z, q_1]$ | $F \rightarrow 1FH$

$\delta(q_0, 0, z) = (q_1, z)$ — Read.

$[q_0, z, q_0] \rightarrow 0\,[q_1, z, q_0]$ | $E \rightarrow 0G$

$[q_0, z, q_1] \rightarrow 0\,[q_1, z, q_1]$ | $F \rightarrow 0H$.

$\delta(q_1, 1, z) = (q_1, \varepsilon)$ — Pop

$[q_1, z, q_1] \rightarrow 1$

$H \rightarrow 1$

$\delta(q_1, 0, z_0) = (q_0, z_0)$

$(q_1, z_0, q_0] \rightarrow 0\,[q_0, z_0, q_0]$ | $C \rightarrow 0A$

$[q_1, z_0, q_1] \rightarrow 0\,[q_0, z_0, q_1]$ | $0 \rightarrow 0B$

$S \rightarrow A|B$

$A \rightarrow 1EA|1FC|\varepsilon$

$B \rightarrow 1EA|1FD$

$C \rightarrow 0A$

$D \rightarrow 0B$

$E \rightarrow 1EE|1FG|0G$

$F \rightarrow 1EEF|1FH|0H$

$H \rightarrow 1$

Feature	Pushdown Automata (PDA)	Context free language (CFL)
Definition	A computational machine that uses a stack to recognize languages.	A set of languages generated by CFGs.
Basis	Machine-Oriented model.	Grammar-oriented model.
Acceptance Method.	Accepts by final state or empty stack.	Generated using production rules of a CFG.
Represent-ation.	Defined by 7 tuples: $(Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$	Defined by 4 tuples: (S, V, T, R) or (V, Σ, R, S)
Power	Used to recognize CFLs.	Defines languages that PDAs can recognize.
Relationship.	Every PDA → CFL.	Every CFL → PDA recognizes it.

DCFL's →

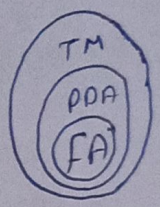
Unit 5 - Turing Machine.

Turing Machine Model -

- What is Turing Machine?
- A Turing Machine (TM) is the most powerful computing model. It can solve any problem that any computer can solve.

• Components of a Turing Machine -

① Infinite Tape - Divided into cells, can store symbols.
② Read/Write Head - can read & write symbols on tape.
③ State Register - stores current state.
④ Finite Control - Contains program / instructions.



• Features of a TM -

A Turing Machine is defined as a 7 Tuple -

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

where, Q - Finite set of states.
Σ - I/P alphabet.
Γ - Tape alphabet
δ - Transition function
$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$
q_0 - Initial State
B - Blank symbol.
F - Final states.

- Most powerful model of computation

• Possible outcomes -

① Accept - TM halts in accepting state.
② Reject - TM halts in non-accepting state.
③ Loop - TM never halts (runs forever).

• Design of Turing Machine - $L = \{a^n b^n \mid n \geq 1\}$

B	B	a	a	a	b	b	b	B	B
		X	X	X	Y	Y	Y		

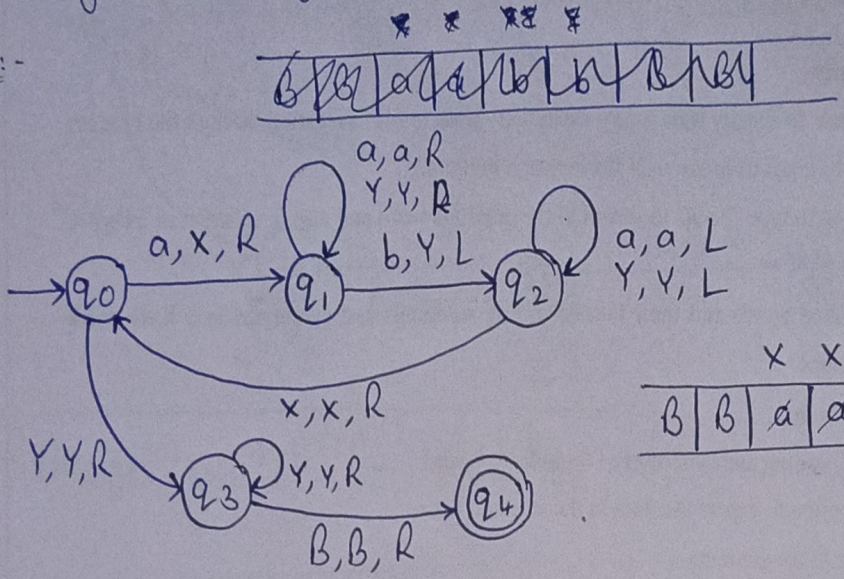
- Types of Languages -

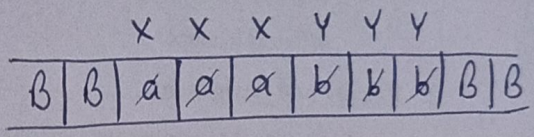
① Recursively Enumerable (RE) - Language accepted by TM (may Loop).

② Recursive - Language where TM always halts (accepts or rejects).

Q. Design a Turing Machine for Language, $L = \{a^n b^n \mid n \geq 1\}$

Ans:-





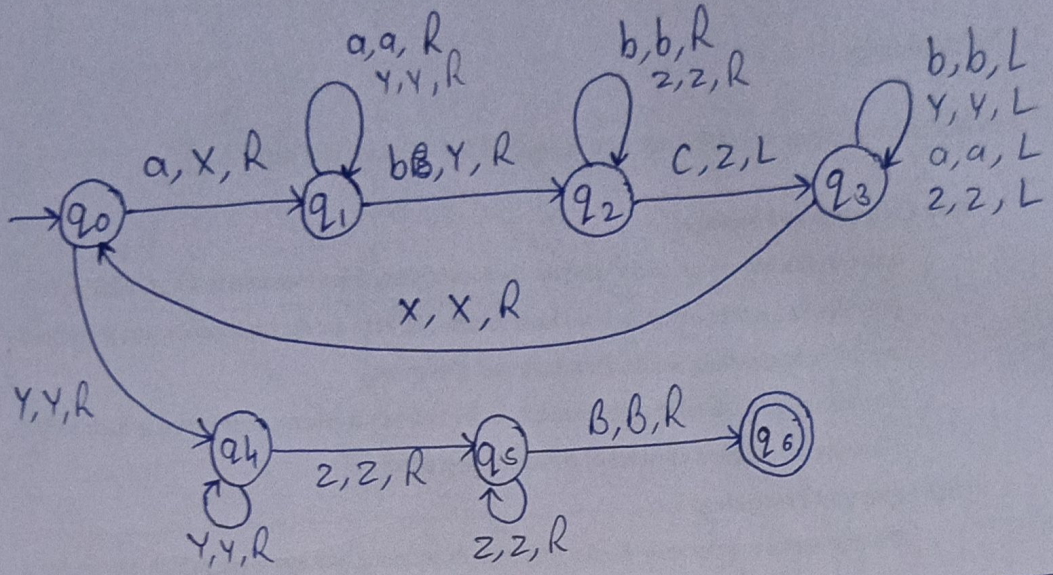
- Transition Table

	a	b	X	Y	B
q_0	q_1, X, R	Halts	Halts	q_3, Y, R	Halts
q_1	q_1, a, R	q_2, Y, L	Halts	q_1, Y, R	Halts
q_2	q_2, a, L	-	q_0, X, R	q_2, Y, L	-
q_3	-	-	-	q_3, Y, R	q_4, B, R
q_4					

Q. Design a TM for language :- $L = \{a^n b^n c^n \mid n \geq 1\}$

Ans-

B	B	a	a	a	b	b	b	c	c	c	B	B
		X	X	X	Y	Y	Y	Z	Z	Z		

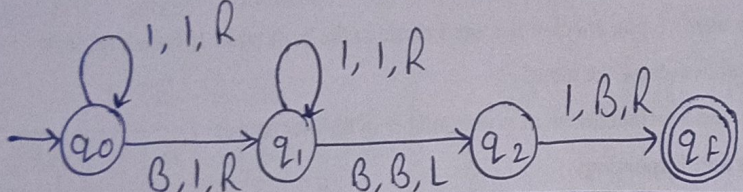


Q. Design a TM for Addition of 2 numbers in Unary.

Ans:-

B	B	1	1	1	Ø	1	1	1	X	B
					1					

3 + 4 = 7

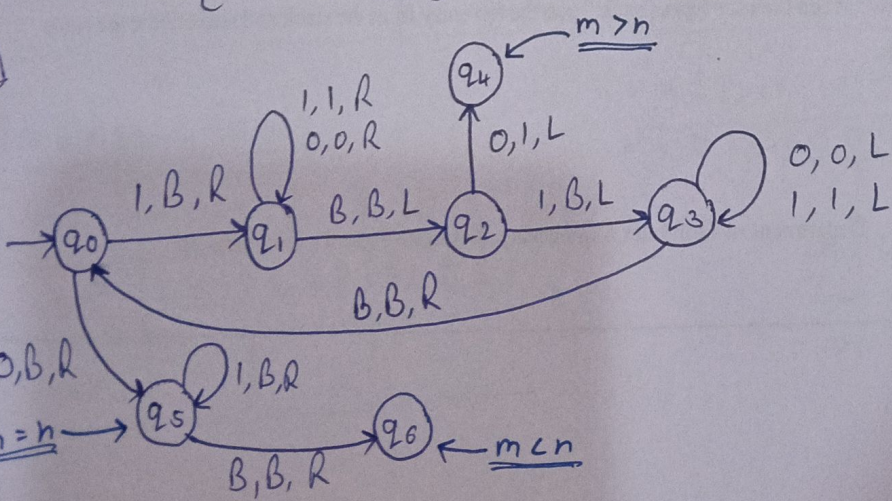


Q. Design TM ~~of~~ for substraction of 2 unary no. $f(a-b) = c$ where a is always greater than b.

Ans:- $f(a,b) = \begin{cases} a-b & \text{if } a > b \\ 0 & \text{if } a \leq b \end{cases}$

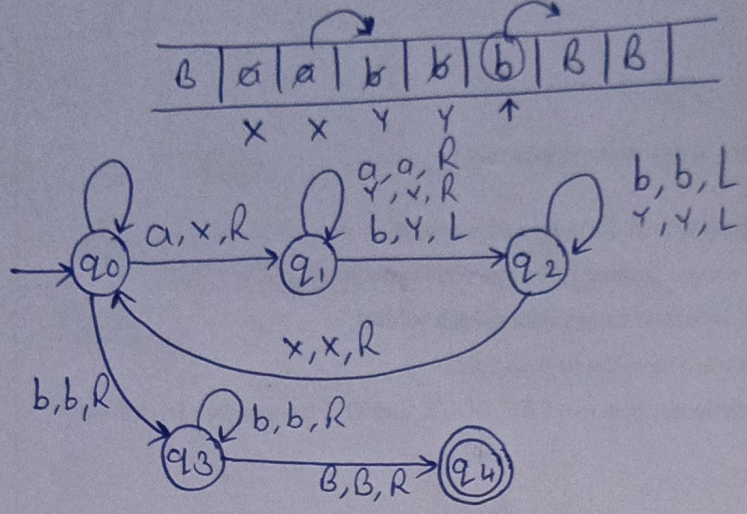
B	B	X	X	X	1	Ø	X	X	B	B
					1					
		B	B	B	-		B	B		

2024 PYQ



Q. Design TM for language $L: \{a^i b^j \mid i<j\}$

Ans:- ~~$a^i b^j$~~ $a^i b^j \mid i<j$ i.e $a<b$ $2<3$



Description of TM:-

ways to describe a TM:-

① Instantaneous Description (ID) :- Shows current configuration of TM format: $\alpha q \beta$

TM then the ID of a TM can be given by a triplet or a program which is as given below:

$$(q_0, a) \rightarrow (q_1, A, L)$$

② Transition Notation :-

$ID_1 \vdash ID_2$ (one move)

$ID_1 \vdash^* ID_2$ (zero or more moves)

Techniques for TM Construction.

① Storage in State :- The state name itself can store small bits of information.

Ex :- q_0_saw_0 → means we are in state q_0 & we have already seen a 0.

Idea : Instead of writing Data on tape, we remember it in the name of the state.

② Multiple Tracks - Here, tape has multiple lanes or tracks. i.e. Each track can hold different types of information.-

Track 1 : Original input.

Track 2 : Extra notes or markings.

Idea : Helps store extra data side by side without overwriting the original input

③ Marking - We can mark symbols that we have already visited or processed.

Ex :- Replace 0 with X,
Replace 1 with Y.

④ Shifting - Sometimes, we need to move the contents of the tape Left or Right, this is called Shifting.

Idea - Useful when we want to insert or delete symbols or rearrange data.

⑤ Subroutines - Just like in programming, we can design small parts (modules) of a TM separately - each doing a specific task. Later we combine them to build the complete TM.

Q. How Turing machine can compute functions?

Ans :- A TM is not only used to ~~rec~~ recognize languages, but it can also be used to compute functions just like a computer performs calculations.

eg. Addition, Sub, Mult, pattern checking.

- How TM computes a function :

① Input - The functions arguement (s) written on tape.

② Process - TM performs required computation.

③ Output - when TM halts, tape shows the results.

Q. How ~~Turing~~ Explain Halting problem of Turing Machine.

Ans:- The halting problem states -

Given any functional TM 'T' & an input data tape 't' of initial configuration, determine whether the process will run forever or will eventually stop (halt).

- The Halting problem is unsolvable - ~~These~~ is there is no Turing Machine that can correctly decide this for all possible i/p.

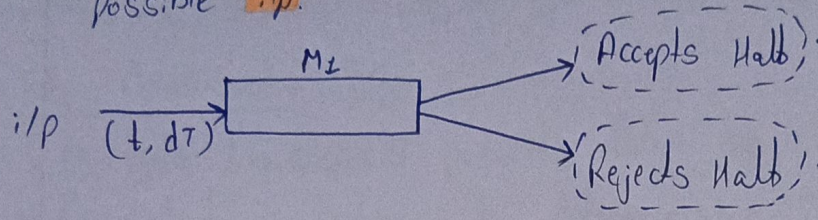


Fig. Hypothetical Halting Decider.

- When T does not halt for t.

Machine M2 - The modified machine.

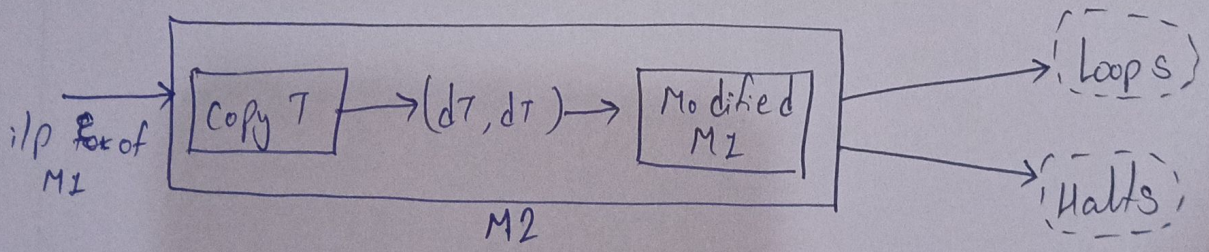
Input to M2 - description of TM T.

- Steps of M2 - ① Copies T onto its own tape.

② Feeds (T, T) (i.e. machine T with its own description as i/p) to M1.

③ Based on result from M1 :-

- if M1 says T halts on T, then M2 loops forever.
- if M1 says T does not halt on T, then M2 halts.



So, no Turing Machine can decide whether any other TM halts or Loops forever.

Q. Extensions of TM.

Ans:- TM have several extensions, all of which are computationally equivalent to the basic TM.

① Multi-Tape Turing Machine.
- Has multiple tapes & heads.

② Multi-Track TM.
- One tape divided into multiple tracks.
- Allowing storing multiple symbols on the same tape cell.

③ Non-Deterministic TM.
- Multiple transitions possible for a single configuration.
- Equivalent in power to deterministic TM.

④ Multi-Dimensional TM.
- Tape is 2-D instead of 1-D.

⑤ Universal TM. (UTM)
- A TM that can simulate Any other TM when given its encoding.

- All these extensions do NOT increase computational power - only convenience.