Context free grammar (CFG)

It is a formal grammar which is used to generate all possible patterns of strings in a given formal language. CFG G can be defined by four tuples as.

$G = (V, T, P, S)$

G : set of production rules, used to generate the string of a language

V : final set of (Variables) non-terminal symbol). (denoted by capital letter)

T : — // ——— of termal (symbol. (— // — lower case)

S : Start Variable.

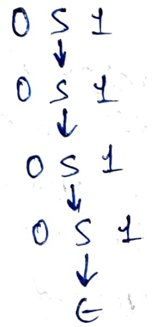
$(V \cap T = \phi)$

$S \rightarrow 0S1 \mid \epsilon$

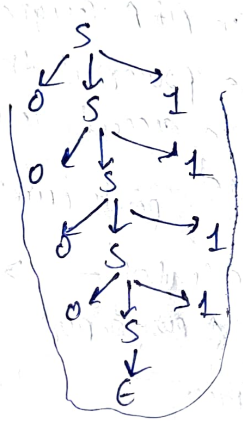
$V = \{S\}$

$Start = \{S\}$

$T = \{0, 1, \epsilon\}$



⇒



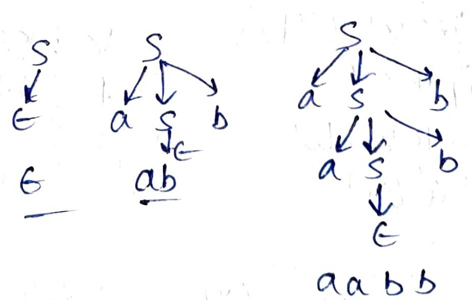
0000 1111 $\boxed{\text{String}}$

$\boxed{\text{language} = \{0^n 1^n \quad n \geq 0\}}$

→ CFL → CFG

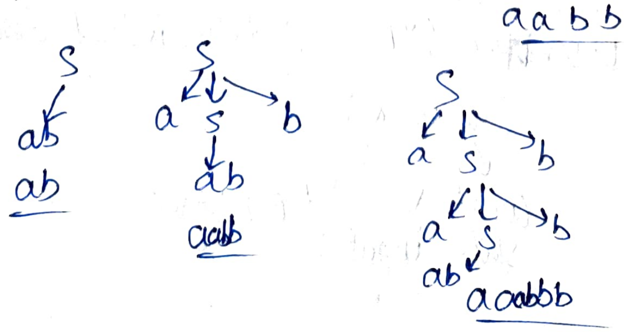
(i) $a^n b^n, n \geq 0$

$S \rightarrow aSb \mid \epsilon$



(ii) $a^n b^n, n \geq 1$

$S \rightarrow aSb \mid ab$



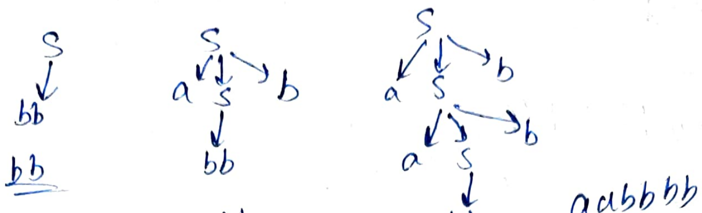
(iii) $a^n b^{n+2}, n \geq 0$

$n=0$

bb

$S \rightarrow aSb \mid bb$

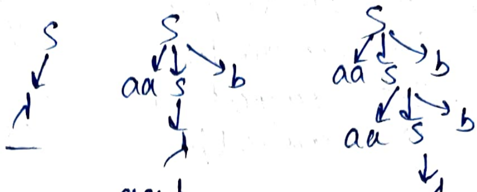
(ii)



bb $abbb$ $aabbbb$

(iv) $a^{2n} b^n, n \geq 0$

$S \rightarrow aaSb \mid \lambda$



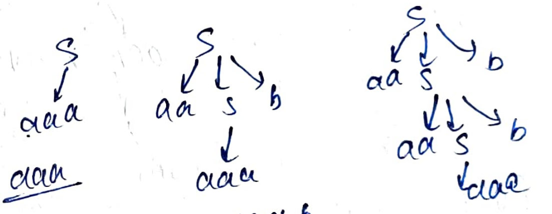
aab $aaaabb$

(v) $a^{2n+3} b^n, n \geq 0$

$n=0$

aaa

$S \rightarrow aaSb \mid aaa$



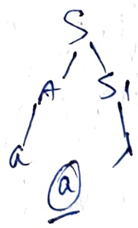
aaa $aaaaab$ $aaaaaaabb$

(vi) $a^m b^n, m > n$
$n \geq 0$

$S_1 \rightarrow aS_1b \mid \lambda$

$A \rightarrow aA \mid a$

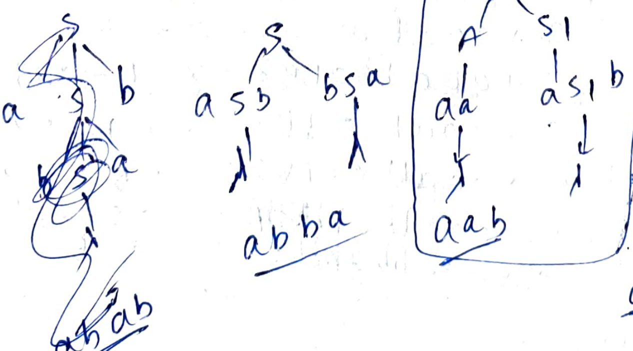
$S \rightarrow AS_1$



a

Q $\{w \mid n_a(w) = n_b(w)\}$

$S \rightarrow aSb \mid bSa \mid SS \mid \lambda$



$abab$ $abba$ aab

Q $ww^R \cup$
$w(a+b)w^R$

$S \rightarrow aSa \mid bSb \mid a \mid b$
$\mid \lambda$

$abba$

Q $a^m b^m c^n, m, n \geq 0$

$S \rightarrow aS_1b \mid \lambda$

$C \rightarrow cC \mid \lambda$

$S \rightarrow S_1C$

Derivation

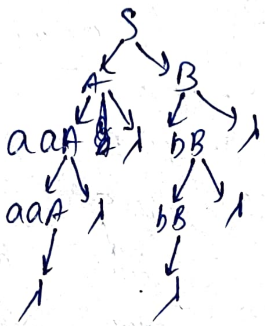
(1)

$S \to AB$
$A \to aaA \mid \lambda$
$B \to bB \mid \lambda$

check whether
$W = aaaabb \in L(G)$

$V = \{A, B\}$
$S = \{S\}$
$T = \{a, b\}$

$S \to AB$
$S \to aaAB$ (using $A \to aaA$)
$S \to aaaaAB$ (using $A \to aaA$)
$S \to aaaabB$ (using $A \to \lambda$)
$S \to aaaabbB$ (using $B \to bB$)
$S \to aaaabb$ (Hence string is accepted)



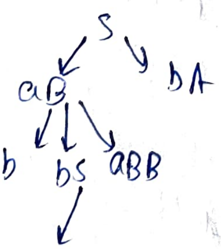
aaaabb

Ques

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

string $= aaabbabbba$

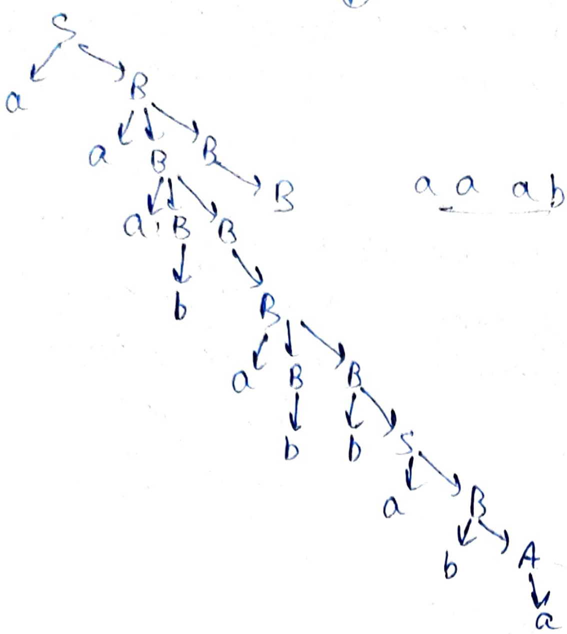
variable $V = \{A, B\}$
$T = \{a, b\}$
$S = \{S\}$



$S \to aB$
$S \to aaBB$ {using $B \to aBB$}
$S \to aaaBBB$ {— " — " →}
$S \to aaabBB$ {— " — $B \to b$}
$S \to aaabbB$ {— " →}
$S \to aaabbaBB$ {— " — $B \to aBB$}
$S \to aaabbabB$ {— " — $B \to b$}
$S \to aaabbabbS$ {→ $B \to bS$}
$S \to aaabbabbbA$ {→ $S \to aB$}
$S \to aaabbabbba$ {→ $S \to bA$}

derivation tree

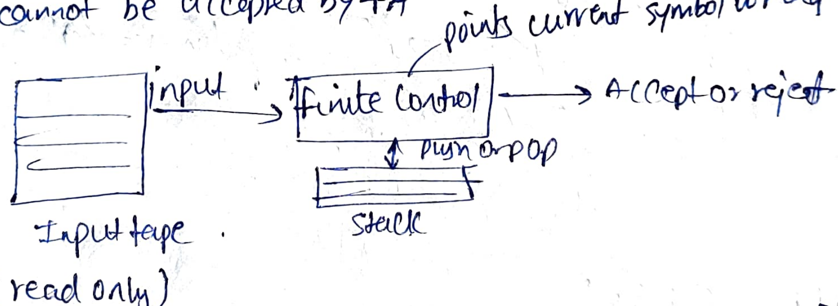
(P)



a a a b a b b a b a

→ Push Down Automata

PDA is a way to implement a CFG. PDA can remember infinite amount of info. It is simply an NFA augmented with an external stack memory. PDA is more powerful than FA. PDA is more superior than FA, it can accept a class of language which even cannot be accepted by FA



PDA is collection of 7 tuple. $(Q, \Sigma, \Gamma, q_0, Z, F, \delta)$

Q: finite set of states
Σ: input set
Γ: Stack symbol
q_0: initial state
Z: Start symbol
F: final set state
δ: transition function.

Instantaneous Description (ID): tells instant descript of PDA

(q, w, α)

q → current state
w → input
α → top of stack

Context free grammar (CFG)

It is a formal grammar which is used to generate all possible patterns of strings in a given formal Language. CFG G can be defined by four tuples as.

$G = (V, T, P, S)$

G: Set of production rules, used to generate the string of a language

V: final set of Variables (non-terminal symbol). (denoted by capital letter)

T: — " — of termal (symbol. (— " — lower case)

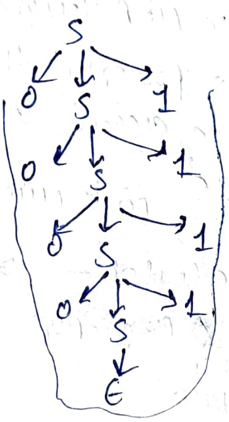
S: Start Variable. $(V \cap T = \emptyset)$

$S \to 0S1 \mid \epsilon$

0 S 1
↓
0 S 1
↓
0 S 1
↓
0 S 1
↓
ϵ

$V = \{S\}$
Start $= \{S\}$
$T = \{0, 1, \epsilon\}$

⇒



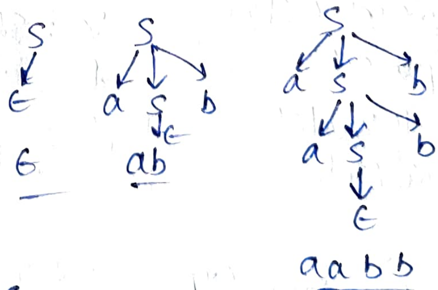
0000 1111 $\boxed{\text{String}}$

$\boxed{\text{langu}^n = \{0^n 1^n \quad n \geq 0\}}$

→ CFL → CFG

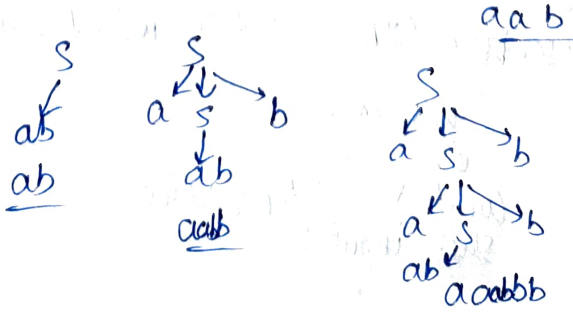
(i) $a^n b^n, n \geq 0$

$S \to aSb \mid \epsilon$



(ii) $a^n b^n, n \geq 1$

$S \to aSb \mid ab$



(A)

→ acceptance of String in PDA.

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

input String $w =$ aaabbb

(A)

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

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

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

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

$\delta(q_1, \epsilon, z_0) = (q_2, z_0)$

Soln

$\delta(q_0, \text{aaabbb}, z_0) \vdash q_0, \text{aabbb}, az_0$ (push)

$\vdash q_0, \text{abbb}, aaz_0$ (push)

$\vdash q_0, \text{bbb}, aaaz_0$ (push)

$\vdash q_1, \text{bb}, aaz_0$ (POP)

$\vdash q_1, \text{b}, az_0$ (pop)

$\vdash q_1, \epsilon, z_0$ skip

$\vdash q_2, z_0$ String is accepted.

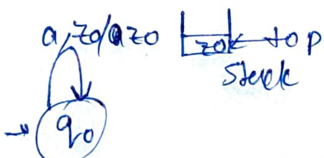
Q Construct PDA $L = \{a^n b^n \mid n \geq 1\}$

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

let String be aabb

Step-1

$w =$ aabb



Step-2

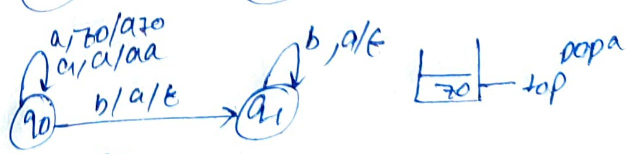
input symbol a

Step-3

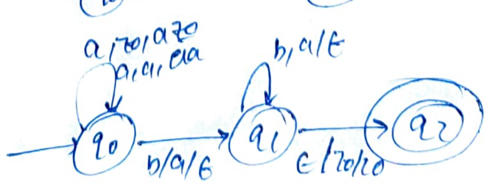
input symb a

Step 4 input symbo b

Step 5 input symbol b



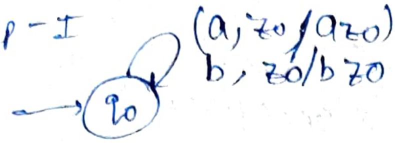
Step-6 input end String



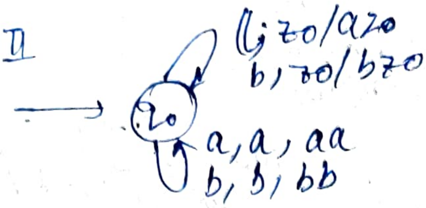
Q. construct PDA $L = \{ w \mid n_a(w) = n_b(w) \}$ (4)

$\{ ab, ba, aabb, bbaa, baba \ldots \}$

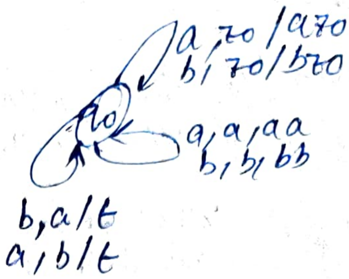
step - I



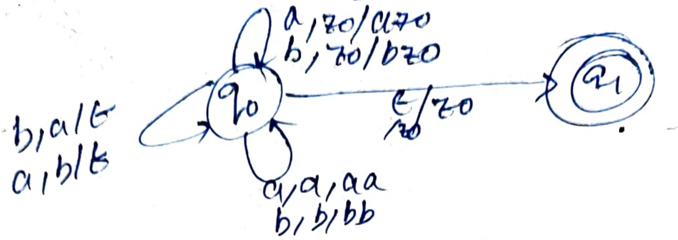
step - II



step - 3



step - 4



transition fcn.

$\delta(q_0, a, z_0) = (q_0, az_0)$
$\delta(q_0, b, z_0) = (q_0, bz_0)$
$\delta(q_0, a, a) = (q_0, aa)$
$\delta(q_0, b, b) = (q_0, bb)$
} push

$\delta(q_0, a, b) = (q_0, \epsilon)$
$\delta(q_0, b, a) = (q_0, \epsilon)$
} pop

$\delta(q_0, \epsilon, z_0) = (q_1, z_0)$ or stop accepted, q_1 final state.