DEEPAK SINGLA
(Asst. Prof)
CSE - III rd yr

(1) Nos UNIT - 3

Grammer :- It is a mechanism for representing a formal language.

→ A grammer is defined as a set of Rules to define Valid Sentence in any language.

→ Grammer is generally denoted by the symbol 'G' and can be defined formally with four tuppples,

$$G = (V_n, V_T, P, S)$$

Where V_n = finite nonempty set of Variables.

V_T = finite nonempty set of terminals

S = starting symbol

P = Set of production rules in the form of $\alpha \rightarrow \beta$ where α, β is string on $V_n \cup V_T$.

→ There are various kind of grammer

1) Regular Grammer
2) Context - free grammer.

1) CONTEXT FREE GRAMMAR :-

→ G is Context-free if every production is of the form $A \to \alpha$, where $A \in V_N$ and $\alpha \in (V_N \cup V_T)^*$.

→ formaly CFG is define with 4-tuples as!

$$CFG = (V_N, V_T, P, S)$$

where V_N = Set of Non Terminal (Varible)

V_T = Set of Terminals

P = Production rules in the form of $\alpha \to \beta$

S = Start symbol.

where

a) $|\alpha| \leq |\beta|$, length of L.H.S of rules must be equal or less than the size of R.H.S

b) $\wedge \alpha \in V_N$, every symbol on the L.H.S of the rule must be non-Terminal

c) $|\alpha| = 1$ means L.H.S of the rule Contain a single non-terminal symbol & Can never be empty.

d) β is string consisting terminal and non-terminals.

For example:- The grammer is given as:

$S \rightarrow aSa$
$S \rightarrow bSb$
$S \rightarrow c$

Where, $V_T = \{a, b, c\}$

$V_N = \{S\}$

$V_T = \{a, b, c\}$

P = Production rules are:

$S \rightarrow aSa$
$S \rightarrow bSb$
$S \rightarrow c$

$S = \{S\}$ = Start Symbol.

→ The language generated by a CFG is the set of all strings of terminals that can be produced from the start symbol S using the production.

→ A language generated by CFG is called

a Context-free language (CFL).

DERIVATIONS AND LANGUAGE GENERATED BY a GRAMMAR

Derivation :- Derive any string by using the production rules are called Derivation. f.e.

The grammar is given as:

$S \rightarrow 0S1$ —(i)

$S \rightarrow 01$ —(ii)

and derive a string 0011.

∴ Derive a string 0011 by using the above production rules as

$S \overset{1}{\Rightarrow} 0S1$

$\overset{2}{\Rightarrow} 0011$ {Replace S with S→0

Language Generation :- The language generated by a grammar 'G' is the set of all terminal strings derived from the start symbol S.

DERIVATION TREE OR PARSE TREE

→ We use a tree structure to derive a string from the given grammar.

→ These types of tree is called Derivation tree or parse tree or syntax tree.

→ The pictorial representation of Derivation of a sentance is called parse tree.

Types of Derivation Tree

→ It is of two types:→

① Left Most Derivation (LMD) :- In this the left most non terminal symbol in the derivation is always replaced first by a production rules then the type of derivation is called LMD.

for Example :- The LMD of the string w = abaaba with the given production rules as:

$S \rightarrow AAA \mid AA$

$A \rightarrow AA \mid aA \mid Ab \mid a \mid b$

LMD :-

$S \Rightarrow \underline{A}AA$

$\Rightarrow a\underline{A}AA$

$\Rightarrow ab\underline{A}A$

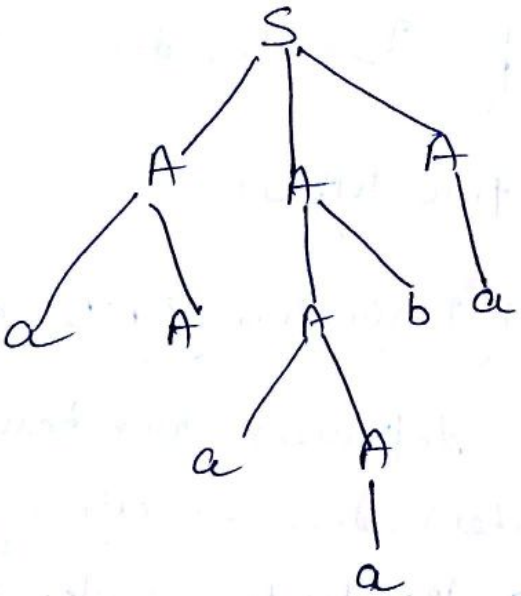
$\Rightarrow abA\underline{b}A$

$\Rightarrow aba\underline{A}bA$

$\Rightarrow abaa\underline{b}A$

$\Rightarrow abaaba$

Derivation Tree :-



2. Right most Derivation (RMD) :- In this type of derivation Right most non-Terminal symbol is replaced first by some production rule. then these type of Derivation is called RMD.

For Example:

$S \rightarrow bB \mid aA$

$A \rightarrow b \mid bS \mid aAA$

$B \rightarrow a \mid aS \mid bBB$

find a Right most Derivation for string "bbaababa"

Ans: RMD:-

$S \Rightarrow bB$

$\Rightarrow bbBB$

$\Rightarrow bbBaS$

$\Rightarrow bbBabB$

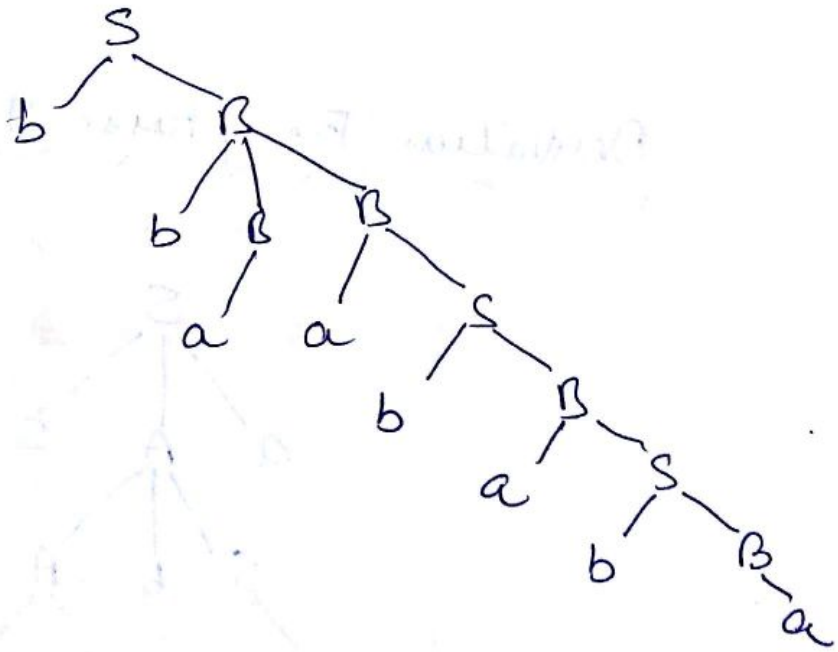
$\Rightarrow bbBabaS$

$\Rightarrow bbBababB$

$\Rightarrow bbBababa$

$\Rightarrow bbaababa$

Derivation Tree:-



Example: If CFG has productions

$$S \to aAS \mid a$$

$$A \to SbA \mid SS \mid ba$$

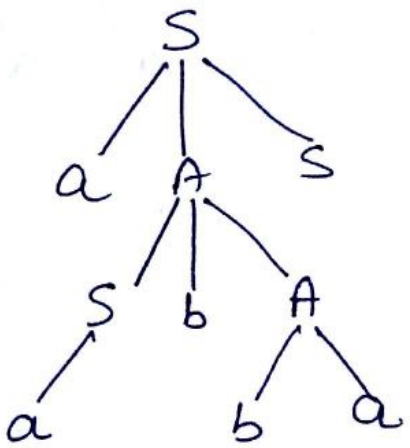
Show that $S \overset{*}{\Rightarrow} aabbaa$ & Construct parse tree whose yield is aabbaa.

Ans:

$$S \overset{lm}{\Rightarrow} aAS$$
$$\overset{lm}{\Rightarrow} aSbAS$$
$$\overset{lm}{\Rightarrow} aabAS$$
$$\overset{lm}{\Rightarrow} aabbaS$$
$$\overset{lm}{\Rightarrow} aabbaa$$

$\therefore S \overset{*}{\Rightarrow} aabbaa$

Derivation Tree / Parse Tree :-



Ans:

Sentential form :- If $S \overset{*}{\Rightarrow} \alpha$, then α is called a sentential form. i.e.

$S \to OS1$
$S \to 01$ } ← Sential form.

LANGUAGE GENERATION NUMERICAL

Example If $G = \{\{S\}, \{0,1\}, \{S \to OS1, S \to \wedge\}, S\}$, find $L(G)$.

Ans: As $S \to \wedge$ is a production. $\therefore \wedge \in L(G)$

$$S \overset{G}{\Rightarrow} OS1$$
$$\overset{G}{\Rightarrow} OOS11 = O^2S1^2$$
$$\overset{G}{\Rightarrow} OOOS111 = O^3S1^3$$
$$\vdots$$
$$\overset{G}{\Rightarrow} O^{n-n}S1\cdots n = O^nS1^n$$
$$\Rightarrow O^n1^n \quad \{\text{Replace } S \to \wedge\}$$

$\therefore O^n1^n \in L(G)$ for $n \geq 0$

$\therefore L(G) = \{O^n1^n \mid n \geq 0\}$ Ans.

Example: If $G = (\{S\}, \{a\}, \{S \to SS\}, S)$, find the language generated by G.

Ans: $L(G) = \phi$, since the only production $S \to SS$ in G has no terminal on right hand side.

Example: Let $G = (\{S, C\}, \{a, b\}, P, S)$, where P consist of

$S \to aCa$ — (1)

$C \to aCa \mid b$ — (2)

Find $L(G)$.

Ans: $S \Rightarrow aCa \Rightarrow aba$ so $aba \in L(G)$

$S \Rightarrow aCa$

$\underset{2}{\Rightarrow} aaCaa$

$\underset{2}{\Rightarrow} aaaCaaa$

$\vdots$

$\overset{*}{\Rightarrow} a^n b a^n$ { Replace C with $C \to b$ }

Hence $a^n b a^n \in L(G)$, where $n \geq 1$

$\therefore L(G) = \{a^n b a^n \mid n \geq 1\}$. Ans

Example: If G is $S \to \underset{①}{aS} \mid \underset{②}{bS} \mid \underset{③}{a} \mid \underset{④}{b}$, find L(G).

Ans:

As $S \to a$ ∴ $a \in L(G)$

As $S \to b$ ∴ $b \in L(G)$

As $S \Rightarrow aS$
$\Rightarrow aa$ [Replace S with $S \to a$]

∴ $aa \in L(G)$

As $S \Rightarrow aS$
$\underset{1}{\Rightarrow} abS$
$\underset{3}{\Rightarrow} aba$ ∴ $aba \in L(G)$

As. $S \Rightarrow bS$
$\underset{4}{\Rightarrow} bb$ ∴ $bb \in L(G)$

As $S \Rightarrow bS$
$\underset{1}{\Rightarrow} baS$
$\underset{2}{\Rightarrow} babS$
$\underset{3}{\Rightarrow} baba$ ∴ $baba \in L(G)$

∴ From above we see that, we generate all the combination of a & b except $\wedge$ only. ∴ we can write the above

Grammer as :

$$L(G) \subseteq \{a,b\}^* - \{\wedge\} = \{a,b\}^+$$

∴ we have $L(G) = \{a,b\}^+$

Ans.

Example. If G is $S \to aSa$ —(1)
$S \to bSb$ —(2)
$S \to c$ —(3)

then find L(G) ?

Ans: As $S \xRightarrow{3} c$ ∴ $c \in L(G)$

As. $S \xRightarrow{1} aSa$
$\xRightarrow{3} aca$ ∴ $aca \in L(G)$

As $S \xRightarrow{2} bSb$
$\xRightarrow{3} bcb$ ∴ $bcb \in L(G)$

As. $S \xRightarrow{1} aSa$
$\xRightarrow{2} abSba$
$\xRightarrow{3} abcba$ ∴ $abcba \in L(G)$

reverse

∴ $L(G) = \{wcw^R \mid w \in \{a,b\}^*\}$

Example If G is $S \to aSb$ — (1)
$S \to ab$ — (2) &
$S \to SS$ — (3)

$T = \{a, b\}$, $N = \{S\}$, Find L(G)?

Ans As $S \overset{3}{\Rightarrow} SS$
$\overset{2}{\Rightarrow} abS$
$\overset{2}{\Rightarrow} abab$ ∴ abab ∈ L(G)

As $S \overset{3}{\Rightarrow} SS$
$\overset{*}{\Rightarrow} asbab$ ← Two steps together
$\overset{3}{\Rightarrow} assbab$
$\overset{*}{\Rightarrow} aababbbab$ ← Single step
↑ equal no. of a's & b's

∴ aababbbab ∈ L(G)

∴ $L(G) = \{$ consist of equal no. of a's & b's $\}$
$= \{a^n b^n \mid n \geq 1\}$

OR

L(G) = { Consist of well form strings of parenthesis if we replace $a \Rightarrow ($ & $b \to)$ in above production rules.

Example: If G = $S \to aB$ ——— (1)
$B \to b$ ——— (2)
$B \to bS$ ——— (3)
$B \to aBB$ ——— (4)
$S \to bA$ ——— (5)
$A \to a$ ——— (6)
$A \to aS$ ——— (7)
$A \to bAA$ ——— (8)

Find L(G)?

Ans: $S \xRightarrow{1} aB \xRightarrow{2} ab$ ∴ $ab \in L(G)$

$S \xRightarrow{5} bA \xRightarrow{6} ba$ ∴ $ba \in L(G)$

$S \xRightarrow{1} aB \xRightarrow{3} abS$
$\xRightarrow{5} abbA$
$\xRightarrow{6} abba$ ∴ $abba \in L(G)$

In general it derive a string as:
{ab, ba, abba, baba, abab ---- }.

∴ L(G) = { Contain even length string with equal no. of a's & b's. }.

Answer.

AMBIGUITIE IN CFG

→ A context-free grammar 'G' is ambiguous if there is at least one string in L(G) having two or more different derivation Trees.

OR

A grammar which produce more than one Parse tree for the same sentence or string is called ambiguous grammar.

for Example :- The given grammar is ambiguous.

If we drive a string id+id*id.

$E \rightarrow E+E \mid E*E \mid id$

We can construct a two different Parse tree corresponding to the similar input string as id+id*id.

(1)

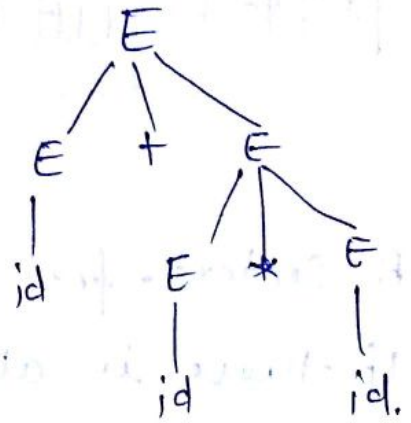
$E \overset{lm}{\Rightarrow} E+E$

$\Rightarrow id+E$

$\Rightarrow id+E*E$

$\Rightarrow id+id*E$

$\Rightarrow id+id*id$



(2)

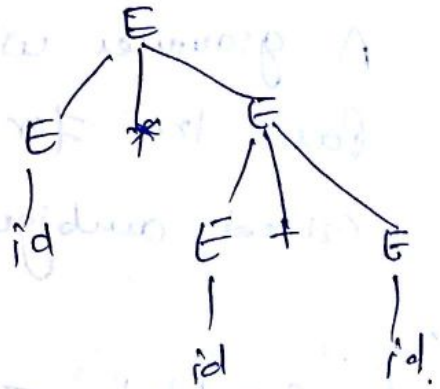
$E \overset{lm}{\Rightarrow} E*E$

$\Rightarrow E+E*E$

$\Rightarrow id+E*E$

$\Rightarrow id+id*E$

$\Rightarrow id+id*id$



So, from above we get same string is generated using two different leftmost deriv

∴ two different parse tree exist for string id+id*id . ∴ grammar is Ambiguous.

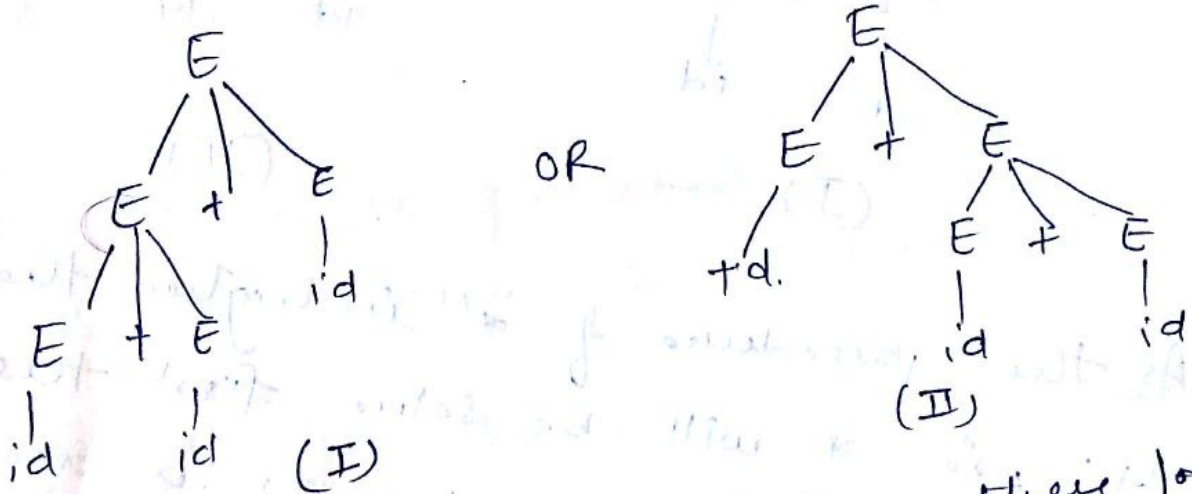
How To Remove Ambiguity

Causes for ambiguity:-

1) Rule of associativity is not respected.
2) Rule of precedence is not respected.

Eg: (1) $E \rightarrow E+E \mid E*E \mid id$, generate string.

id+id+id., we get two different parse tree.

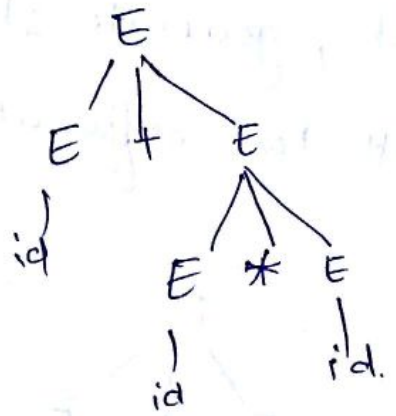


As the '+' is left associative, there fore we calculate left side of tree & then add it to Right one. ∴ the parse tree (I) is correct & (II) is not correct.

∴ to remove this ambiguity, we restrict our tree only to the left side not right side.

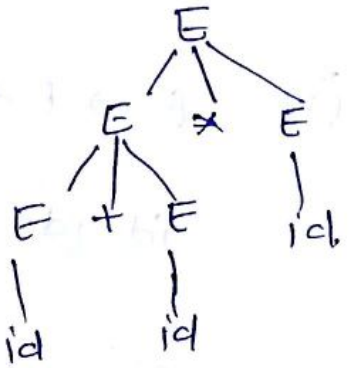
2 (Precedence rule) :-

for Example Construct a parse tree of id+ id * id with the same grammar E → E+E | E*E | id then we will get a two different parse tree as :



(I)

OR



(II)

As the precedence of '*' is higher than '+', so * will be solve first then '+', so the parse tree (I) is correct so to remove the ambiguity we introduce two New Non terminal symbol as T & form the Unambiguous grammar as :

E → E+T | T
T → T*F | F
F ⇒ id

Ans

Example: Show that the given grammar is ambiguous. $S \to SaS \mid b$

Ans: Suppose the string $w = bababab$ is derived we can make two LMD for this string as:

$S \overset{LMD}{\Rightarrow} \underline{S}aS$
$\Rightarrow ba\underline{S}$
$\Rightarrow baSaS$
$\Rightarrow babaS\underline{a}S$
$\Rightarrow bababa\underline{S}$
$\Rightarrow bababab$

$S \overset{LMD}{\Rightarrow} \underline{S}aS$
$\Rightarrow Sa\underline{S}aS$
$\Rightarrow ba\underline{S}aS$
$\Rightarrow baSa\underline{S}aS$
$\Rightarrow babaS\underline{a}S$
$\Rightarrow bababa\underline{S}$
$\Rightarrow bababab$

So, 'w' can be generated with two LMD
∴ the grammar 'G' is Ambiguous.

Example: Show that the grammar 'G' is ambiguous.

$S \to abSb \mid aAb \mid a$
$A \to aAAb \mid bS.$

Ans: Let the string $w = abab$ is derived.

$S \Rightarrow abSb$
$\Rightarrow abab$

or it can be obtained by other way as:

$S \Rightarrow aAb$

$\Rightarrow abSb$

$\Rightarrow abab$

Since the String 'w' have two LMD, therefore G is ambiguous.

(3)

CFG Generator Numericals

Example1: write cfg for the language contain a string of length 2.

Ans: $L = \{aa, ab, ba, bb\}$ or $\{(a+b)(a+b)\}$

$V_N = \{S, A\}$, $V_T = \{a, b\}$

$S \to AA$

$A \to a \mid b$

Answer

Example 2: write CFG for the language $L = \{a^n \mid n \geq 0\}$

Ans: $L = \{\wedge, a, aa, aaa, aaaa \cdots\}$

$V_N = \{S\}$, $V_T = \{a, \epsilon\}$

$S \to aS \mid \epsilon$

Example 3: write CFG for language $L = (a+b)^*$

Ans: $L = \{\wedge, a, b, ab, ba, aba \cdots\}$

$V_N = \{S\}$, $V_T = \{a, b, \epsilon\}$.

$S \to aS \mid bS \mid \epsilon$

Example 4: write CFG for the language contain atleast 2 length.

Ans: $L = \{(a+b)(a+b)(a+b)^*\}$

$S \rightarrow BBA$

$B \rightarrow a|b$

$A \rightarrow aA|bA|\epsilon$

Example: write a CFG for language have atmost length 2.

Ans $L = \{(a+b+\epsilon)(a+b+\epsilon)\}$

∴ grammer is :- $S \rightarrow AA$

$A \rightarrow a|b|\epsilon$

$V_N = \{S, A\}$, $V_T \{a, b, \epsilon\}$

Example : write CFG for $a(a+b)^*b$..

Ans $S \rightarrow aAb$

$A \rightarrow aA|bA|\epsilon$

$V_N = \{A, S\}$

$V_T = \{a, b, \epsilon\}$

Example: Write CFG, that start & end with different symbol.

Ans $L = a(a+b)^*b + b(a+b)^*a$

∴ grammer is :

$S \rightarrow aAb|bAa$

$A \rightarrow aA|bA|\epsilon$

Example: write cfg for the language that start & end with same symbol.

Ans: $L = a(a+b)^* a + b(a+b)^* b$.

∴ grammar is: $V_N = \{S, A\}$, $V_T = \{a, b, \in\}$

$S \to aAa \mid bAb \mid a \mid b \mid \in$

$A \to aA \mid bA \mid \in$.

Example: write cfg for the language $L = \{a^n b^n \mid n \geq 1\}$

Ans: As the above language is not regular But still we can generate a grammar for the same language.

$V_N = \{S\}$, $V_T = \{a, b\}$

$S \to aSb \mid ab$

Check whether the string aabb is derivable or not

$S \Rightarrow aSb$

$\Rightarrow aabb$

So, the string aabb can be derived from given cfg.

Q.P.

Example: write cfg for all the language that contain set of all palindrom.

Ans: The set of all palindrom contain $\{wwR \cup waw^R \cup wbw^R\}$

where wwR contain the even no. of Palin & waw^R & wbw^R contain the odd Palindrom. where $w \leftarrow (a,b)^*$

∴ The grammar is: $V_N = \{S\}$, $V_T = \{a,b\}$

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

Check whether the string aa accepted or not

$S \Rightarrow aSa$
$\Rightarrow aa$ {Replace $S \rightarrow \epsilon$}

Check whether string aaa is accepted or not

$S \Rightarrow aSa$
$\Rightarrow aaa$ {Replace S with $S \rightarrow a$}

∴ from above both odd length & even length palindrom accepted.

Example: write CFG for the language $L = \{a^n b^m / m, n \geq 1\}$

Ans: $L = \{ab, aab, abb, aaab, \ldots\}$

∴ grammar corresponding language is:

$V_N = \{S, A, B\}$

$V_T = \{a, b\}$

$S \rightarrow AB$ ——— (1)

$A \rightarrow aA \mid a$ ——— (2)

$B \rightarrow bB \mid b$ ——— (3)

Check the string aaab is derive from above grammar or not?

$S \xRightarrow{1} AB$

$\xRightarrow{2} aAB$

$\xRightarrow{2} aaAB$

$\xRightarrow{3} aaaB$

$\xRightarrow{3} aaab$

∴ the string 'aaab' is derived from the above grammar.

Example: write CFG for the Language $L = \{a^n b^n c^m / n, m \geq 1\}$

Ans: $L = \{abc, aabcc, abbcc \cdots\}$

∴ the grammer corresponding the Langu.

∴ $V_N = \{S, A, B\}$, $V_T = \{a, b, c\}$

$S \rightarrow AB$ —— (1)

$A \rightarrow aAb \mid ab$ —— (2)

$B \rightarrow CB \mid C$ —— (3)

check the string "aabbcc" is derive fr above grammer or not?

$S \xrightarrow{1} AB$

$\xrightarrow{2} aAbB$

$\xrightarrow{2} aabbB$

$\xrightarrow{3} aabbCB$

$\xrightarrow{3} aabbcc$

∴ the string "aabbcc" is derived from the above grammer.

Example: Write a CFG for the language $L = \{a^n c^m b^n / n, m \geq 1\}$

Ans: $L = \{aacbb, aaacbbb \ldots\}$

∴ the grammar corresponding to language is:

$V_N = \{S, A\}$

$V_T = \{a, b, c\}$

$S \rightarrow aSb \mid aAb$ —(1)

$A \rightarrow CA \mid C$ —(2)

Check the string 'aacbb' is derived from the above grammar?

$S \overset{1}{\Rightarrow} aSb$

$\overset{1}{\Rightarrow} aaAbb$

$\overset{2}{\Rightarrow} aacbb$

from above, the string 'aacbb' is accepted or derive from the above grammar.

Example: Write CFG for the langu $L = \{a^n b^n c^m d^m / n, m \geq 1\}$

Ans: $L = \{aabbcd, abccdd, aaabbbcddd\}$

∴ the grammar for above is:

$V_N = \{S, A, B\}$

$V_T = \{a, b, c\}$

$S \rightarrow AB$ —(1)

$A \rightarrow aAb \mid ab$ —(2)

$B \rightarrow cBd \mid cd$ —(3)

Check the string "aabbcd" is derived from the grammar.

$$S \xRightarrow{1} AB$$
$$\xRightarrow{2} aAbB$$
$$\xRightarrow{2} aabbB$$
$$\xRightarrow{3} aabbcd$$

∴ the string "aabbcd" is derived from the above grammar. Ans

Example write a CFG for $L = \{a^n b^{2n} \mid n \geq 1\}$

Ans $L = \{abb, aabbbb \cdots\}$

∴ the grammar correspond to above language is: $V_N = \{S\}$, $V_T = \{a, b\}$

$$S \to aSbb \mid abb$$

Check the string "aabbbb" is derived or not?

$$S \xRightarrow{1} aSbb$$
$$\xRightarrow{1} aabbbb.$$

Ans

Example: write a CFG for the $L = \{a^n b^m c^m d^n \mid n,m \geq 1\}$

Ans: $L = \{abcd, aabcdd, abbccdd \cdots\}$

So, grammar for above language is:

$V_N = \{S, A\}$

$V_T = \{a, b, c\}$

$S \rightarrow aSd \mid aAd \quad \text{---(1)}$

$A \rightarrow bAc \mid bc \quad \text{---(2)}$

Check the string "aabcdd" is derived or not?

$$S \xrightarrow{1} aSd$$
$$\xrightarrow{1} aaAdd$$
$$\xrightarrow{2} aabcdd$$

So the string "aabcdd" is derived from the above grammar.

Example: Write CFG for $L = \{a^{m+n} b^m c^n \mid m,n \geq 1\}$

Ans: $L = \{aabc, aaaabbcc \cdots\}$

So the grammar for above language is written as:

$S \to aSc \mid aAc$ $\quad V_N = \{S, A\}$

$A \to aAb \mid ab$ $\quad V_T = \{a, b, c\}$

Check the string "aabc" is derived from the grammar or not?

$S \Rightarrow aSc$ (1)

$\Rightarrow aabc$ (2)

from above the string 'aabc' is derived

Ans

Example: write cfg for $L = \{a^n b^{n+m} c^m \mid n, m \geq 1\}$ or $a^n b^n b^m c^m$

Ans $L = \{abbc, aabbbbcc \ldots\}$

∴ the grammar is written as:

$S \to AB$ —— (1)

$A \to aAb \mid ab$ —— (2)

$B \to bBc \mid bc$ —— (3)

Check the string "aabbbbcc" is derived from above grammar or not!

$S \xRightarrow{1} AB$

$S \xRightarrow{2} aAbB$

$\xRightarrow{2} aabbB$

$\xRightarrow{3} aabbbBc$

$\xRightarrow{3} aabbbbcc$

∴ the string "aabbbbcc" is derived from above grammar.

Example: $L = \{w \mid w \in (0,1)^*\}$, write CFG to generate L where w consist of equal no. of 0's & 1's.

Ans: $L = \{01, 10, 0011, 1100, 000111 \ldots\}$

∴ the grammar for above language is,

$S \rightarrow SS$ ——— (1)

$S \rightarrow 0S1 \mid 1S0$ ——— (2)

$S \rightarrow \wedge$ ——— (3)

Check the string "0011" is derived from the above grammar or not?

$S \xrightarrow{1} SS$

$S \xrightarrow{2} 0S1S$

$S \xrightarrow{2} 00S11S$

$S \xrightarrow{3} 0011S$

$S \xrightarrow{3} 0011$

∴ the above string is accepted & derive from the above grammer.

Example: write cfg of the language $L = \{0^i 1^j 0^k\}$

$L = \{0^i 1^j 0^k \mid j > i+k\}$.

Answer: The above language can be written as:

$x = 0^i 1^i 1^m 0^k 0^k \quad (m > 0)$

∴ the language support above is:

$L = L_1 L_2 L_3$, where.

$L_1 = \{0^i 1^i \mid i \geq 0\}$

$L_2 = \{1^m \mid m > 0\}$.

$L_3 = \{1^k 0^k \mid k \geq 0\}$

Language L_1 is written as

$$A \to 0A1 \mid \wedge$$

Langue L_2 is written as

$$B \to 1B \mid 1$$

Langug L_3 is written as

$$C \to 1C0 \mid \wedge$$

∴ final CFG $G = \{V, \Sigma, S, P\}$ can be written as!

$V = \{S, A, B, C\}$ $\quad \Sigma = \{0, 1\}$

$P = \{ S \to ABC$
$A \to 0A1 \mid \wedge$
$B \to 1B \mid 1$
$C \to 1C0 \mid \wedge \}$

Check the string $01^4 0^2 = (01)(1^2 0^2)$ is derive from above grammar or not?

$S \Rightarrow ABC$
$\Rightarrow 0A1BC$
$\Rightarrow 0\wedge 1BC$
$\Rightarrow 011C$
$\Rightarrow 0111C0 \Rightarrow 01111C00 \Rightarrow 01111\wedge 00$
$\Rightarrow 011100$ Ans.

CFG equivalent to Regular Expr

Imp. Ques:- write a CFG corresponds to the given R.E as $(011+1)^*(01)^*$

Ans:- $A \to 011 \mid 1$ generate the language $\{011, 1\}$.

$B \to AB \mid \wedge$
$A \to 011 \mid 1$

B as start symbol will generate the language $\{011, 1\}^*$.

$C \to DC \mid \wedge$
$D \to 01$

D generate the langue 01 & C generate $(01)^*$

The final cfg is

$S \to BC$
$B \to AB \mid \wedge$
$A \to 011 \mid 1$
$C \to DC \mid \wedge$
$D \to 01$

Answer

Example: Write the CFG for the language.

$$L = \{ 0^i 1^j 2^k \mid i=j \text{ or } j=k \}$$

Ans:

$$L = L_1 \cup L_2$$

$$L_1 = \{ 0^i 1^j 2^k \mid i=j \}$$

$$L_2 = \{ 0^i 1^j 2^k \mid j=k \}$$

$$L_1 \Rightarrow \begin{cases} S_1 \to AB \\ A \to 0A1 \mid \epsilon \\ B \to 2B \mid \epsilon \end{cases}$$

$$L_2 \Rightarrow \begin{cases} S_2 \to CD \\ C \to 0C \mid \epsilon \\ D \to 1D2 \mid \epsilon \end{cases}$$

$\therefore L_1 \cup L_2 \Rightarrow$

$S \to S_1 \mid S_2$
$S_1 \to AB$
$A \to 0A1 \mid \epsilon$
$B \to 2B \mid \epsilon$
$S_2 \to CD$
$C \to 0C \mid \epsilon$
$D \to 1D2 \mid \epsilon$

Ans:

Example 16: Design a CFG for the language.

$$L = \{a^n b^m : n \neq m\}$$

Sol^n:- There is two possibility occurs

Case I: $n > m$ $\quad V_T = \{a, b\}$

Case II: $n < m$ $\quad V_N = \{S, S_1, S_2\}$

Let Case1: $L_1 = \{a^n b^m : n > m\}$

$S_1 \rightarrow AB$

$B \rightarrow aBb \mid \epsilon$

$A \rightarrow aA \mid a$

Case 2: $L_2 = \{a^n b^m : n < m\}$

$S_2 \rightarrow CD$

$C \rightarrow aCb \mid \epsilon$

$D \rightarrow bD \mid b$

Now $L = L_1 \cup L_2$

Hence $\boxed{S \rightarrow S_1 \mid S_2}$ Ans.

Example: write a CFG for the language over $\Sigma = \{a, b\}$, defined by the following regular expression:

$$(a+b)^* (bbb + aaa) (a+b)^*$$

Ans: $L = L_1 L_2 L_3$

where $L_1 = (a+b)^*$

$L_2 = bbb + aaa$

$L_3 = (a+b)^*$

$V_T = \{a, b\}$

$V_N = \{S, A, B\}$

$P = \{ S \to ABA$
$A \to aA | bA | \wedge$
$B \to bbb | aaa$
$\}$

∴ The grammar corresponding to L_1 is:

$$A \to aA | bA | \wedge$$

The grammar corresponds to L_2 is

$$B \to bbb | aaa.$$

The total grammar is written as:

$$S \to ABA$$
$$A \to aA | bA | \wedge$$
$$B \to bbb | aaa$$

Now check the string 'abbbb' is

derive from the generated grammar or not?

$S \overset{LMD}{\Rightarrow} ABA$

$\Rightarrow aBBA$

$\Rightarrow a\wedge BA$

$\Rightarrow abbA$

$\Rightarrow abbbB$

$\Rightarrow abbb\wedge$

$\Rightarrow abbbb$

As, 'abbbb' string is derived from the above grammar. Answer

Deepak Singla CSE IInd

Mob Motor

SIMPLIFIED CONTEXT-FREE GRAMMAR

30

→ To restrict the formats of context free grammar without reducing the language generating power of CFG.

→ We can Simplified CFG by using following three properties!

1) We must eliminate useless symbol, those variables or terminals that do not appears in any derivation of a terminal string from the start symbol.

2) We must eliminate ε-production (NULL productions), are those of the form $X \to \varepsilon$ for some Non-Terminal X.

3) We must eliminate unit production, those of the form $X \to Y$ for vaible X & Y.

1) ELIMINATE USELESS SYMBOL :-

→ A symbol Y in a context-free grammar is usefull if and only if:

a) $Y \overset{*}{\Rightarrow} w$, where $w \in L(G)$ and w in V_t^*, that is Y leads to a string of terminals. Here Y is said to be generating.

b) If there is a derivation $S \overset{*}{\Rightarrow} \alpha Y \beta \Rightarrow w$, $w \in L(G)$, for some α & β then Y is said to be reachble.

→ A symbol that is usefull will be both generating and reachable.

→ If we eliminate the symbol that are not generating first and then eliminate from the remaining grammer those symbols that are not reachable. Then after this process, context free grammar 'G' will have only usefull symbols.

∴ In reduction of given grammer 'G', following steps

1) identified non-generating symbols in given CFG and eliminate those productions which contain non-generating symbols.

2) identified non-reachble symbols in grammer and eliminate those production which contain non-reachble symbols.

for Example I: Consider a CFG

$S \to AB/a$

$A \to b$

identify & eliminate useless symbols.

Ans: Given CFG is

$S \to AB/a$

$A \to b$

Step I: Generating Symbols are $\{S, A, a, b\}$

useless symbols = {total symbol} − {generating symbol}

$\{S, A, B, a, b\} - \{S, A, a, b\}$

$\Rightarrow \{B\}$

∴ useless symbol is 'B'.

Now delet check the production which contain B symbol, we will get the New grammar.

as. $S \to a$

$A \to b$

Step 2 Now check the reachability from the start symbol.

Non-reachble = Total symbol − reachable symbol

$\{S, A, a, b\} - \{S, a\}$.

we will get.

@ Non-reacble symbol = {A, b}

∴ elminate this production, we will get.

S → a Ans

Example: Remove the useless symbol from the given context free grammar.

S → aB | bX
A → BAd | bSX | a
B → aSB | bBX
X → SBD | aBx | ad.

Ans: Step I: from above we choose those non-terminal which are deriving to the string of terminals. we will get the generating symbols as.

Generating symbol = { A, a, X, d, S, b }

non-generating symbol :- {All symbol} - {generating symbol}

{S, A, B, X, a, b, d} - {A, a, X, d, S, b}

⇒ {B}

∴ 'B' is the useless symbol. So, eliminate

all the production which contain symbol 'B' we, well get the new grammar as!

$$S \to bX$$
$$A \to bSx \mid a$$
$$X \to ad$$

Step 2: Now, check the reachability from the start symbol S. as!

non-reachable symbol = { All symbol } - {reachable symbol}

$$\{S, X, A, a, b, d\} - \{S, b, X, a, d\}$$

$$\Rightarrow \{A, b\}$$

∴ Eliminate the non-reachable symbol from the above grammar, we get:

$$S \to bX$$
$$X \to ad$$

Ans

Example 3: Consider the following grammar & obtain an equivalent grammar contain no-use-less symbols.

$$A \to xyz \mid Xyzz$$
$$X \to Xz \mid xYz$$
$$Y \to yYy \mid Xz$$
$$Z \to Zy \mid z$$

Ans: Step 1:-

non-generating symbol = {TOTAL symbol} - {Generating symbol}

$\Rightarrow \{A, X, Y, Z, x, y, z\} - \{A, x, y, z, Z\}$

$\Rightarrow \{X, Y\}$

∴ X, Y are useless symbol. ∴ Eliminate X, Y symbol from the given grammar we will get the new grammar as:

$A \rightarrow xyz$

$Z \rightarrow Zy \mid z$

Step 2: Check the reachability from the start symbol 'A'. as:

non-reachable symbols = (Total symbol) - (reachable symbol)

$= \{A, Z, x, y, z\} - \{A, x, y, z\}$

$= \{Z\}$

∴ 'Z' is non-reachable symbol, ∴ delete the symbol 'Z' from the above grammar we get new grammar as: $\boxed{A \rightarrow xyz}$

Ans.

2. REMOVE NULL (ϵ) production :-

In a given CFG, we call a non-terminal N is nullable if there is a production of the form $\boxed{N \to \epsilon}$, where N is any non-terminal.

→ To eliminate ϵ-production from a grammar A' we use the following technique :

(1) If $A \to \epsilon$ is a production to be eliminated then we look for all production whose right side contain A, and replace each occurence of A in each of these production to obtain the non ϵ-production.

2) Now the resultant non-null production must be added to the grammar to keep the language generated the same.

Example :- Consider the grammar

$S \to aA$

$A \to b \mid \epsilon$

remove the nullable non terminal.

Ans: Given CFG is

$S \to aA$
$A \to b \mid \epsilon$

Here $A \to \epsilon$ is ϵ-production, so put ϵ in place of A, at the right side of production and add the resulted production to the grammar. we will get the new grammar as:

$S \to aA \mid a$
$A \to b$

Ans

Example: Consider the grammar G

$S \to ABAC$
$A \to aA \mid \epsilon$
$B \to bB \mid \epsilon$
$C \to c$

remove ϵ-production from the above grammar.

Ans: In this we have two ϵ-production to remove as: $A \to \epsilon$ & $B \to \epsilon$

∴ first eliminate $A \to \epsilon$, by putting ϵ in the right sides of given grammar as:

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

$A \rightarrow aA \mid a$

$B \rightarrow bB \mid \epsilon$

$C \rightarrow c$

Now, delet the $B \rightarrow \epsilon$, by putting 'ϵ' in the right side of gramm instead of B symbol we get.

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

$A \rightarrow aA \mid a$

$B \rightarrow bB \mid b$

$C \rightarrow c$

Ans

Example: Remove ϵ-production from the given gramm

$S \rightarrow aSa$

$S \rightarrow bSb \mid \epsilon$

Ans $S \rightarrow \epsilon$ is the only null production in given gramm. ∴ put ϵ instead of S' on the right side of the given grammer.

$S \rightarrow aSa \mid aa$

$S \rightarrow bSb \mid bb$

Ans

Example: Design a CFG for regular expression $r = (a+b)^* bb (a+b)^*$ which is free from $\in$-Production.

Ans: CFG required for above R.E is:

$V_T = \{a, b, \wedge\}$

$V_N = \{A, B\}$

$B \rightarrow AbbA$

$A \rightarrow aA \mid bA \mid \wedge$

As $A \rightarrow \wedge$ is the only $\in$-Production. So to eliminate it put $\wedge$ instead of A' on the right side of grammar, we get!

$B \rightarrow Abb A \mid Abb \mid bbA \mid bb$

$A \rightarrow aA \mid bA \mid a \mid b$ Ans

3 REMOVE UNIT PRODUCTION

→ The Unit production is define as if the Production is of the form:

Non Terminal → one non-Terminal.

OR

$A \rightarrow B$ is called Unit Production

→ Unit production increase the cost of derivation in a grammar. ∴ we eliminate the unit productions.

→ To Remove unit production we must follow two steps as:

Step I:- Collect all the unit production rules

Step 2:- Consider all the non terminals having unit production rule one by one.

Suppose $A \Rightarrow A_1 \Rightarrow A_2 \Rightarrow \cdots \Rightarrow A_n \Rightarrow \alpha$

where $A, A_1, A_2 \cdots A_n$ are non-terminal and α is string over $(V_n + S)^*$ then remove all A production by production rules:

$$A \rightarrow \alpha.$$

Example I:- Consider the grammar G.

$S \rightarrow AB$
$A \rightarrow a$
$B \rightarrow C \mid b$
$C \rightarrow D$
$D \rightarrow E$
$E \rightarrow a$

remove unit productions:

Ans: Given CFG is

$S \to AB$
$A \to a$
$B \to C \mid b$
$C \to D$
$D \to E$
$E \to a$

Contain three unit productions as:

$B \to C$
$C \to D$
$D \to E$

Now to remove unit production $B \to C$, we see if there exist a production whose left side has C and right side contains a terminal ($C \to a$), but there is no such production in G. Similar things holds for unit production $C \to D$.

∴ we try to remove the unit productions $D \to E$ because there is a production $E \to a$.

∴ After eliminate $D \to E$ & introduce $D \to a$, a grammar becomes

$S \rightarrow AB$
$A \rightarrow a$
$B \rightarrow C \mid b$
$C \rightarrow D$
$D \rightarrow a$
$E \rightarrow a$

Now we can remove $C \rightarrow D$ by using $D \rightarrow a$, we get:

$S \rightarrow AB$
$A \rightarrow a$
$B \rightarrow C \mid b$
$C \rightarrow a$
$D \rightarrow a$
$E \rightarrow a$

Similarly we can remove $B \rightarrow C$ by using $C \rightarrow a$ we get;

$S \rightarrow AB$
$A \rightarrow a$
$B \rightarrow a \mid b$
$C \rightarrow a$
$D \rightarrow a$
$E \rightarrow a$

As C, D, E are useless symbols, ∴ the reduce grammar is

$S \rightarrow AB$
$A \rightarrow a$
$B \rightarrow a \mid b$

Ans

Example 2: Consider the following grammar

$I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

$F \rightarrow I \mid (E)$

$T \rightarrow F \mid T * F$

$E \rightarrow T \mid E + T$

identified unit production and remove it.

Ans:- In the given grammar.

$F \rightarrow I$

$T \rightarrow F$

$E \rightarrow T$

are the unit production.

first delet $F \rightarrow I$ production by the help of

$I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

and grammar become.

$I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

$F \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \mid (E)$

$T \rightarrow F \mid T * F$

$E \rightarrow T \mid E + T$

Now we eliminate $T \rightarrow F$ by the help of

$F \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \mid (E)$

and now grammar become.

$I \rightarrow a|b|Ia|Ib|I0|I1$

$f \rightarrow a|b|Ia|Ib|I0|I1|(E)$

$T \rightarrow a|b|Ia|Ib|I0|I1|(E)|T*F$

$E \rightarrow T|E+T$

Now remove $E \rightarrow T$, we get the final grammar as:

$I \rightarrow a|b|Ia|Ib|I0|I1$

$f \rightarrow a|b|Ia|Ib|I0|I1|(E)$

$T \rightarrow a|b|Ia|Ib|I0|I1|(E)|T*F$

$E \rightarrow a|b|Ia|Ib|I0|I1|(E)|T*f|E+T.$

Ans

Example: Identified and remove the unit production from following grammar:

$S \rightarrow A|bb$

$A \rightarrow B|b$

$B \rightarrow S|a$

Ans: Here unit productions are:

$S \rightarrow A$

$A \rightarrow B$

$B \rightarrow S.$

Now after deleting all the unit products the grammar is generated as:

$S \to A$ generate $S \to b$

$S \to A \to B$ generate $S \to a$

$A \to B$ generate $A \to a$

$A \to B \to S$ generate $A \to bb$

$B \to S$ generate $B \to bb$

$B \to S \to A$ generate $B \to b$

The new cfg becomes.

$S \to bb \mid b \mid a$

$A \to b \mid a \mid bb$

$B \to a \mid bb \mid b$

which holds no unit productions.

Ans.

NORMAL FORMS FOR CFG

→ In Context-free grammer, the R.H.S of production can be any string of Variables and terminals. When the production in 'G' satisfy certain restrictions then 'G' is said to be in a "normal form".

→ CFG normal forms are of two types :-

1) Chomsky normal form (CNF)
2) Greibach normal form (GNF)

1) CHOMSKY NORMAL FORM (CNF)

Defination :- A context-free grammer 'G' is in chomsky normal form if every production is of the form

$A \rightarrow a$ or $A \rightarrow BC$

1.1 Conversion to Chomsky Normal form

Step I :- Elimination of null production & unit production.

Step 2:- Replace all the terminals on R.H.S with some new non-terminals & not replace any terminal symbol which are already in CNF form.

Step 3:- Restricting the number of variables on R.H.S:

a) for any production in P_1, the R.H.S Consist of either a single terminal ($A \to a$) or two or more varible ($A \to BC$).

b) If the obtained grammer from step 2 is not of this type then decompose the R.H.S grammer & replace last two symbol with new non-terminal symbol.

E.g. Consider $A \to A_1 A_2 \cdots A_m$ where $m \geq 3$

$\therefore A \to A_1 A_2 A_2$, then we introduce new production as $A \to A_1 C$

$C \to A_1 A_2$

which all of CNF type.

Example:-I. Reduce the grammer G in to CNF.

$S \to aAD$

$A \to aB \mid bAB$

$B \to b$, $D \to d$.

Step 1:- As there is no null production or unit production.

Step 2:- replace all the terminals with new non-Terminal symbol as:

i) $B \to b$, $D \to d$ are already in CNF form ∴ b, d Terminal are not changed with new non-Terminal

ii) $S \to aAD \Rightarrow C_a AD$ & $C_a \to a$

$A \to aB \Rightarrow A \to C_a B$

$A \to bAB \Rightarrow A \to C_b AB$ & $C_b \to b$

Step 3. As $S \to C_a AD$ is not CNF, ∴ AD is replace with new non-terminal as:

$S \to C_a C_1$

$C_1 \to AD$

Similarly $A \to C_b AB$ is not in CNF ∴ AB is replace with new non-terminal as

$A \to C_b C_2$

$C_2 \to AB$

& $C_a \to a$, $A \to C_a B$, $C_b \to b$ is already in CNF form, ∴ no need to replace any symbol.

∴ the final CNF is:

$S \to C_a C_1$
$C_1 \to AD$
$A \to C_b C_2$
$C_2 \to AB$
$C_a \to a$
$C_b \to b$

∴ the above grammar is in CNF.

Ans

Example2 find a grammar in CNF of

$S \to aAbB$
$A \to aA \mid a$
$B \to bB \mid b$

Ans:- Step 1:- As there is no null & unit production in given grammar.

Step 2:- Replace the terminal on R.H.S of grammar with new non terminal

or:

i) As $A \to a$, $B \to b$ are already in CNF, no need to replace terminal symbol

ii) $S \to C_a A C_b B$ & $C_a \to a$ & $C_b \to b$
$A \to C_a A$, $B \to C_b B$

Step 3 As $S \to C_a A C_b B$ is not in CNF form so replace last two non-terminal in R.H.S with new non-terminal. as.

$$S \to C_a A D$$
$$D \to C_b B$$

& $C_a \to a$, $C_b \to b$, $A \to C_a A$, $B \to C_b B$ are already in CNF. form.

∴ Complete CNF is.

$S \to C_a AD$
$D \to C_b B$
$C_a \to a$
$C_b \to b$
$A \to C_a A$
$B \to C_b B$

Ans.

Example:- Convert the following CFG to CNF.

$S \rightarrow ASB \mid \epsilon$

$A \rightarrow aAS \mid a$

$B \rightarrow SbS \mid A \mid bb$

Ans! Step I:- As there exist a null-production & UNIT productions in the given grammar ∴ first we remove null(ϵ)-production & Unit production. as!

a Remove Null production!-

On the given grammar $S \rightarrow \epsilon$ will show the ϵ-production. ∴ to remove this we put ϵ instead of S on the R.H.S of each production & we get the new grammar is of the form:

$S \rightarrow ASB \mid AB$

$A \rightarrow aAS \mid aA \mid a$

$B \rightarrow SbS \mid Sb \mid bS \mid b \mid A \mid bb$

b) Remove UNIT production :-

As there exist a UNIT production as $B \to A$ in the previous grammar. So to remove the unit production we replace 'A' symbol with all the 'A' production in the R.H.S of the grammar & we get the new grammar as:

$S \to ASB \mid AB$

$A \to aAS \mid aA \mid a$

$B \to SbS \mid Sb \mid bS \mid b \mid aAS \mid aA \mid a \mid bb$

Step 2:- Now replace the terminal symbol with non-Terminal new symbol as:

i) As $A \to a$, $S \to b$, $S \to a$, $S \to AB$ are already in CNF form, so no need to replace any symbol from them.

ii) After Replacing the terminal symbol on R.H.S we get the grammer as:

$S \rightarrow ASB \mid AB$

$A \rightarrow C_aAS \mid C_aA \mid a$

$B \rightarrow SC_bS \mid SC_b \mid C_bS \mid b \mid C_aAS \mid C_aA \mid C_bC_b \mid a$

$C_a \rightarrow a$

$C_b \rightarrow b$

Step 3:- Now to make the above grammer in CNF we replace the last two Non-terminal with new non-terminal symbol. i.e.

$S \rightarrow ASB$ CAs this grammer is not in CNF

So, replace last two symbol with new non-term) we get the new grammer as:

$S \rightarrow AD \mid AB$

$D \rightarrow SB$

$A \rightarrow C_aE \mid C_aA \mid a$

$E \rightarrow AS$

$B \rightarrow SB \mid SC_b \mid C_bS \mid b \mid C_aE \mid C_aA \mid C_bC_b \mid a$

$C_a \rightarrow a$

$C_b \rightarrow b$

(Answer)

Example: Convert the grammer with productions

$S \rightarrow abAB$

$A \rightarrow bAB \mid \epsilon$

$B \rightarrow BAa \mid A \mid \epsilon$

into CNF.

Ans: Step 1 :- As there is null production & UNIT production in the above grammer.

i) Remove Null production:-

$A \rightarrow \epsilon$ & $B \rightarrow \epsilon$ are two null productions in the above grammer. So we remove the null production. first we remove $A \rightarrow \epsilon$ we get,

$S \rightarrow abAB \mid abB$

$A \rightarrow bAB \mid bB$

$B \rightarrow BAa \mid A \mid Ba \mid \epsilon$

then we remove $B \rightarrow \epsilon$ we get,

$S \rightarrow abAB \mid abB \mid abA \mid ab$

$A \rightarrow bAB \mid bB \mid bA \mid b$

$B \rightarrow BAa \mid A \mid Ba \mid Aa \mid a$

ii) Remove the UNIT productions :-

As $B \to A$ belongs to unit production only So we remove it by replace 'A' symbol with all the production of A on the R.H.S of grammer, we get the grammer as:

$S \to abAB \mid abB \mid abA \mid ab$

$A \to bAB \mid bB \mid bA \mid b$

$B \to BAa \mid bAB \mid bB \mid bA \mid b \mid Ba \mid Aa \mid a$

Step 2:- Replace the terminal symbol with new non-terminal symbols except those which are already in CNF like $A \to b$, $B \to b \mid a$ we get the New grammer as:

$S \to C_a C_b AB \mid C_a C_b B \mid C_a C_b A \mid C_a C_b$

$A \to C_b AB \mid C_b B \mid C_b A \mid b$

$B \to BAC_a \mid C_b AB \mid C_b B \mid C_b A \mid b \mid BC_a \mid AC_a \mid a$

Step 3:- Now introduce a new non-terminal symbol in above grammer to make them in to CNF form, we get the CNF grammer as:

$S \to C_a B \mid C_a A \mid C_a A \mid C_a C_b$

$A \to C_b D \mid C_b B \mid C_b A \mid b$

$D \to AB$

$B \to BB \mid C_b D \mid C_b B \mid C_b A \mid b \mid B C_a \mid A C_a \mid a$

Example: Let G be the grammar with production.

$S \to AACD$

$A \to aAb \mid \wedge$

$C \to aC \mid a$

$D \to aDa \mid bDb \mid \wedge$

Convert in to CNF.

Ans: Step I:-

a) Eliminate $\wedge$ production:- The nullable variable are A & D. Produce the grammar with production.

$S \to AACD \mid ACD \mid AAC \mid CD \mid AC \mid C$

$A \to aAb \mid ab$

$C \to aC \mid a$

$D \to aDa \mid bDb \mid aa \mid bb$

b). Eliminate unit production:. Here we may simply add the production

$S \to aC \mid a$

and delet $S \to C$

Step 2:- Replace the terminal symbol with non-terminal which are not in CNF.

$S \to AACD \mid ACD \mid AAC \mid CD \mid AC \mid X_aC \mid a$

$A \to X_aAX_b \mid X_aX_b$

$C \to X_aC \mid a$

$D \to X_aDX_a \mid X_bDX_b \mid X_aX_a \mid X_bX_b$

$X_a \to a$

$X_b \to b$

Step 3:- Reduce the grammer whose right side are too long, we ll get the grammer as!

$S \to AT_1$

$T_1 \to ACD$

$T_1 \to AT_2$

$T_2 \to CD$

$S \to AT_2$ ∴ $S \to CD \mid AC \mid X_aC$

$S \to AT_3$ $S \to a$

$T_3 \to AC$

$A \to X_aT_4 \mid X_aX_b$

$T_4 \to AX_b$

$C \to X_aC \mid a$

$D \to X_aT_5 \mid X_aT_6 \mid X_aX_a \mid X_bX_b$

$T_5 \to DX_a$

$T_6 \to DX_b$

$X_a \to a$

$X_b \to b$

Ans:

GNF (GREIBACH NORMAL FORM)

Definition :- A Context-free grammar is in Greibach normal form if every production is of the form $A \to a\alpha$, where $\alpha \in V_N^*$ and $a \in \Sigma$ and $S \to \Lambda$ is in G if $\Lambda \in L(G)$.

For Example :- A grammar G, given by

$S \to aAB \mid \Lambda$
$A \to bC$
$B \to b$
$C \to c$

is in GNf.

CONVERT CFG TO GNF.

Step I :- Rename all the non-Terminals with New non-Terminals. i.e $S \to A$ is replaced with $A_1 \to A_2$

Step2:- Get the productions in the form

$A_i \rightarrow A_j \alpha$ where $(i \leq j)$

Step3:- Remove left Recursion:-

Left Recursion:- If the first left-side Non-Terminal on R.H.S of the production is similar to the Non-terminal on L.H.S then it is called left Recursion. For Example:

$A \rightarrow A a$

Similar, therefore it is left Recursion.

How To Remove left Recursion:-

→ let the Vaible A has left recursive as follows:

$A \rightarrow A\alpha_1 | A\alpha_2 | A\alpha_3 \cdots | A\alpha_n | \beta_1 | \beta_2 | \beta_3 \cdots | \beta_m$,

where $\beta_1, \beta_2 \cdots \beta_m$ do not Begain with A, then we replace A-productions by:

$A \rightarrow \beta_1 A' | \beta_2 A' | \cdots | \beta_m A' | \beta_1 | \beta_2 | \beta_3 \cdots | \beta_m$ where

$A' \rightarrow \alpha_1 A' | \alpha_2 A' | \alpha_3 A' | \cdots | \alpha_n A' | \alpha_1 | \alpha_2 | \alpha_3 | \cdots | \alpha_n$

Step 4:- Get every production in GNF.

Advantages of GNF:-

1) Avoids left recursion
2) Always has terminal in leftmost position in RHS of each production rule.
3) Helps in selecting the suitable production in derivation of a string.
4) Guarantees derivation length no longer than string length.

NUMERICALS

Example:1. Convert the following grammar in to GNF.

$$S \rightarrow abaSa \mid aba$$

Ans:- If we introduce a new variable A & B and productions as $A \rightarrow a$, $B \rightarrow b$ and substitute into given grammar as:

$$S \rightarrow aBASA \mid aBA$$
$$A \rightarrow a$$
$$B \rightarrow b$$

Which is in GNF form. (Ans)

Example 2:- Convert the grammar into CNF.

$S \rightarrow AB$

$A \rightarrow aA \mid bB \mid b$

$B \rightarrow b$

Ans:- We know that in the CNF each production should be of type $A \rightarrow aB \mid a$

We can replace A in $S \rightarrow AB$ by aA or bB or b

we get $S \rightarrow aAB \mid bBB \mid bB$

$A \rightarrow aA \mid bB \mid b$

$B \rightarrow b$

Which is in CNF form. Ans

Example 3:- Convert the following grammar

$S \rightarrow abSb \mid aa$ into CNF.

Ans:- We introduce new variable A and B for a and b, we get the grammar as:

$S \rightarrow aBSB \mid aA$

$A \rightarrow a$

$B \rightarrow b$

(Ans)

Which is in CNF.

Example 4:- Convert the grammar

$S \rightarrow AB$

$A \rightarrow BS \mid a$

$B \rightarrow SA \mid b$ into GNF.

Answer:- The given grammar in CNF. So

Step 1:- Change all the non-Terminals with the new non-Terminals as use the non-Terminal A_1, A_2, A_3 for S, A, B. We get the new form as:

$A_1 \rightarrow A_2 A_3$

$A_2 \rightarrow A_3 A_1 \mid a$

$A_3 \rightarrow A_1 A_2 \mid b$

Step 2:- Check for the Condition $A_i \rightarrow A_j$, where $(i \leq j)$.

a) First production $A_{\underset{i}{1}} \rightarrow A_{\underset{j}{2}} A_3$, where $i=1$ & $j=2$ that satisfy the Condition as $i \leq j$ as $(1 \leq 2)$. So write it as it is.

b) Second production $A_2 \rightarrow A_3 A_1 \mid a$, where $i=2$, $j=3$ that also satisfy the

the condition $i \leq j$ as $(2 \leq 3)$. ∴ with it a

c) The third production $A_3 \rightarrow A_1 A_2 \mid b$, where

$i = 3$, $j = 1$ dont satisfy the condition

$(i \leq j)$ as $(3 \leq 1)$ false condition. ∴ To make

it correct form, put all the A_1 produc-

instead of A_1 in the third production

on the Right side, we get

$$A_3 \rightarrow A_2 A_3 A_2 \mid b$$

Still the condition is not satisfy as $(i \leq j)$

$(3 \leq 2)$ still it is false. ∴ To make

it true put all the production of A_2

in the RHS of the 3rd production, we

get the

$$A_3 \rightarrow A_3 A_1 A_3 A_2 \mid a A_3 A_2 \mid b$$

Now the condition is satisfy ∵ $i = 3$, $j = 3$

$\therefore (i \leq j) \Rightarrow (3 \leq 3)$ (True)

Step 3:- Eliminate Left Recursion :→

There is a Left Recursion in the new third production the left side symbol is similar to the left symbol on R.H.S

$$A_3 \rightarrow A_3 A_1 A_3 A_2 \mid a A_3 A_2 \mid b$$

Left recursion

Remove Left Recursion :-

$$A_3 \rightarrow a A_3 A_2 B_3 \mid b B_3 \mid a A_3 A_2 \mid b$$

$$B_3 \rightarrow A_1 A_3 A_2 B_3 \mid A_1 A_3 A_2$$

Where B_3 is new variable..

So we have a set of production as:

$$A_1 \rightarrow A_2 A_3$$

$$A_2 \rightarrow A_3 A_1 \mid a$$

$$A_3 \rightarrow a A_3 A_2 B_3 \mid b B_3 \mid a A_3 A_2 \mid b$$

$$B_3 \rightarrow A_1 A_3 A_2 B_3 \mid A_1 A_3 A_2$$

Step 4 :- Now, all the productions start with

terminals. with the help of Substitution rule we can replace A_3 in R.H.S of A_2 production. we get

$$A_2 \rightarrow aA_3A_2B_3A_1 \mid bB_3A_1 \mid aA_3A_2A_1 \mid bA_1 \mid a$$

Now all the production of A_2 start with terminal

Again we replace A_2 from $A_1 \rightarrow A_2A_3$, we have

$$A_1 \rightarrow aA_3A_2B_3A_1A_3 \mid bB_3A_1A_3 \mid aA_3A_2A_1A_3 \mid bA_1A_3 \mid aA_3$$

The resulting production are :-

$$A_1 \rightarrow aA_3A_2B_3A_1A_3 \mid bB_3A_1A_3 \mid aA_3A_2A_1A_3 \mid bA_1A_3 \mid aA_3$$

$$A_2 \rightarrow aA_3A_2B_3A_1 \mid bB_3A_1 \mid aA_3A_2A_1 \mid bA_1 \mid a$$

$$A_3 \rightarrow aA_3A_2B_3 \mid bB_3 \mid aA_3A_2 \mid b$$

$$B_3 \rightarrow A_1A_3A_2B_3 \mid A_1A_3A_2$$

In above only 'B' production are not in GNF

Apply substitution rules in production.

$$B_3 \rightarrow A_1A_3A_2B_3$$

we get,

$B_3 \rightarrow aA_3A_2B_3A_1A_3A_3A_2B_3 \mid bB_3A_1A_3A_3A_2B_3 \mid aA_3A_2A_1A_3A_3A_2B_3$
$\mid bA_1A_3A_3A_2B_3 \mid aA_3A_3A_2B_3$

and from $B_3 \rightarrow A_1A_2A_3$, we get.

$B_3 \rightarrow aA_3A_2B_3A_1A_3A_3A_2B_3 \mid bB_3A_1A_3A_3A_2B_3 \mid aA_3A_2A_1A_3A_3A_2A_3$
$\mid bA_1A_3A_3A_2B_3 \mid aA_3A_3A_2$

The final sets of production are:-

$A_1 \rightarrow aA_3A_2B_3A_1A_3 \mid bB_3A_1A_3 \mid aA_3A_2A_1A_3 \mid bA_1A_3 \mid aA_3$

$A_2 \rightarrow aA_3A_2B_3A_1 \mid aB_3A_1 \mid aA_3A_2A_1 \mid bA_1 \mid a$

$A_3 \rightarrow aA_3A_2B_3 \mid bB_3 \mid aA_3A_2 \mid b$

$B_3 \rightarrow aA_3A_2B_3A_1A_3A_2A_3B_3 \mid bB_3A_1A_3A_2A_3B_3 \mid aA_3A_2A_1A_3A_2$
$A_3B_3 \mid bA_1A_3A_2A_3B_3 \mid aA_3A_2A_3B_3 \mid aA_3A_2B_3A_1A_3A_2A_3$
$\mid bB_3A_1A_3A_2A_3 \mid aA_2A_3A_1A_3A_2A_3 \mid bA_1A_3A_2A_3 \mid aA_3A_2A_3$

all production are in CNF.

Example:- Find a grammar in CNF equivalent to the grammar

$E \to E+T \mid T$

$T \to T*F \mid F$

$F \to (E) \mid a$

Answer:- Step I:- We first eliminate unit production.

$E \to E+T \mid T*F \mid (E) \mid a$

$T \to T*F \mid (E) \mid a$

$F \to (E) \mid a$

Step II:- Reduce step I into CNF by introduce new variable A, B, C instead of +, *,)

The modified productions are:

$E \to EAT \mid TBF \mid CEC \mid a$

$T \to TBF \mid CEC \mid a$

$F \to (EC \mid a$

$A \to +$, $B \to *$, $C \to)$

Step 3:- The Variables (non-Terminals) A, B, C, F, T & E are renamed as $A_1, A_2, A_3, A_4, A_5, A_6$

Then the production become:

$A_1 \to +$ $A_2 \to *$, $A_3 \to)$, $A_4 \to (A_6 A_3 \mid a$

$A_5 \to A_5 A_2 A_4 \mid (A_6 A_3 \mid a$

$A_6 \to A_6 A_1 A_5 \mid A_5 A_2 A_4 \mid (A_6 A_3 \mid a$

Step 4: We have to modify only the A_5 & A_6 productions..

i) $A_5 \to A_5 A_2 A_4$ Contain the left Recursion.
(Same.)

∴ Remove left Recursion, we get:

$A_5 \to (A_6 A_3 \mid a$

$A_5 \to (A_6 A_3 Z_5 \mid a Z_5$

$Z_5 \to A_2 A_4 \mid A_2 A_4 Z_5$

ii) $A_6 \to A_5 A_2 A_4$ Cannot satisfy $i \leq j$, $6 \leq 5$ (false)

∴ put A_5 productions in this we get

$A_6 \to (A_6A_3A_2A_4 \mid aA_2A_4 \mid (A_6A_3Z_5A_2A_4 \mid aZ_5A_2A_4$

& $A_6 \to (A_6A_3 \mid a$ are already in GNF.

Now, we get all the production as:

$A_1 \to +$, $A_2 \to *$, $A_3 \to)$, $A_4 \to (A_6A_3 \mid a$

$A_5 \to (A_6A_3 \mid a$ $A_5 \to (A_6A_3Z_5 \mid aZ_5$

$Z_5 \to A_2A_4 \mid A_2A_4Z_5$

$A_6 \to A_6A_1A_5 \mid (A_6A_3A_2A_4 \mid aA_2A_4 \mid (A_6A_3Z_5A_2A_4 \mid aZ_5A_2A_4$

$A \to A \mid (A_6A_3 \mid a$

α ($A_6A_1A_5$) — F_1 ($(A_6A_3A_2A_4$) — F_2 (aA_2A_4) — F_3 ($(A_6A_3Z_5A_2A_4$) — F_4 ($aZ_5A_2A_4$)

β_5 ($(A_6A_3$) — β_6 (a)

On above the production $A_6 \to A_6A_1A_5$ contain left recursion. So remove it, we get.

$A_6 \to (A_6A_3A_2A_4A_6' \mid aA_2A_4A_6' \mid (A_6A_3Z_5A_2A_4Z_6 \mid aZ_5A_2A_4Z_6$

$\mid (A_6A_3Z_6 \mid aZ_6$

$Z_6 \to A_1A_5 \mid A_1A_5Z_6$

Step :- Z_5 production are not in GNF, so they can be modified as $Z_5 \to *A_4 \mid *A_4Z_5$

& The Z_6 Production are $Z_6 \to A_1A_5 \mid A_1A_5Z_6$, these Can be modified as

$$Z_6 \to +A_5 \mid +A_5Z_6$$

Now all the production are in GNF.

Answer)

Example:- Construct a grammar in Greibach Normal form equivalent to the grammar!

$$S \to AA \mid a$$
$$A \to SS \mid b$$

Answer: Step I The given grammar is in CNF.

S & A are renamed as A_1 & A_2 respectively.

So the productions are $A_1 \to A_2A_2 \mid a$

$A_2 \to A_1A_1 \mid b$

Step 2: The production $A_2 \to A_1A_1 \mid b$ does not satisfy the condition $A_i \to A_j$ where $(i \le j)$

So, put all the productions of A_1 in $A_2 \to A_1A_1 \mid b \mid aA_1$, we get the new production

$A_1 \rightarrow A_2 A_2 \mid a$

$A_2 \rightarrow A_2 A_2 A_1 \mid b \mid a A_1$

Step 3 The production $A_2 \rightarrow A_2 A_2 A_1 \mid b$ contains the left Recursion. So after removal of Left Recursion we get the grammar as:

$A_2 \rightarrow b Z_2 \mid a A_1 Z_2 \mid a A_1 \mid b$

$Z_2 \rightarrow A_2 A_1 \mid A_2 A_1 Z_2$

Step 4:- $A_1 \rightarrow A_2 A_2$ are not in GNF. So put A_2 production in above we get

$A_1 \rightarrow b Z_2 A_2 \mid a A_1 Z_2 A_2 \mid a A_1 A_2 \mid b A_2 \mid a$

Step 5:- $Z_2 \rightarrow A_2 A_1 \mid A_2 A_1 Z_2$ are not in GNF

So put A_2 in this we get:

$Z_2 \Rightarrow a A_1 A_1 \mid b A_1 \mid a A_1 Z_2 A_1 \mid b Z_2 A_1$

$Z_2 \rightarrow a A_1 A_1 Z_2 \mid b A_1 Z_2 \mid a A_1 Z_2 A_1 Z_2 \mid b Z_2 A_1 Z_2$

So The final productions are:

$A_1 \rightarrow a \mid aA_1A_2 \mid bA_2 \mid aA_1Z_2A_1 \mid bZ_2A_2$

$A_2 \rightarrow aA_1 \mid b \mid aA_1Z_2 \mid bZ_2$

$Z_2 \rightarrow aA_1A_1 \mid bA_1 \mid aA_1Z_2A_1 \mid bZ_2A_1$

$Z_2 \rightarrow aA_1A_1Z_2 \mid bA_1Z_2 \mid aA_1Z_2A_1Z_2 \mid bZ_2A_1Z_2$

Ans

Example: Convert the grammar

$S \rightarrow ABb \mid a$

$A \rightarrow aaA$

$B \rightarrow bAb$

Answer: Step D:- Replace $b \rightarrow C$, $a \rightarrow D$, we get

$S \rightarrow ABC$, $C \rightarrow b$

$A \rightarrow aDA$, $D \rightarrow a$

$B \rightarrow bAC$, $C \rightarrow b$

So the revised productions are:

$S \rightarrow ABC \mid a$

$A \rightarrow aDA$

$B \rightarrow bAC$

$C \rightarrow b$

$D \rightarrow a$

Step 2:- Rename S, A, B, C, D with A_1, A_2, A_3, A_4, A_5 we get:

$$A_1 \to A_2 A_3 A_4 \mid a$$
$$A_2 \to a A_5 A_2$$
$$A_3 \to b A_2 A_4$$
$$A_4 \to b$$
$$A_5 \to a$$

Step 3:- The only production $A_1 \to A_2 A_3 A_4$ can be modified by replace A_2 with all production of A_2, we get.

$$A_1 \to a A_5 A_2 A_3 A_4 \mid a$$
$$A_2 \to a A_5 A_2$$
$$A_3 \to b A_2 A_4$$
$$A_4 \to b$$
$$A_5 \to a$$

are in GNF form. (Answer)

PREPAIRED BY :→ DEEPAK SINGLA, A.P (CSE)

REGULAR GRAMMAR

Regular Grammar :- A grammar $G = (V, \Sigma, S, P)$ is regular if every production takes one of the two forms :

$$B \to aC$$
$$B \to a$$

where B & C are non-terminals & a is terminals.

Types of Regular Grammar :-

1) Right linear Grammar
2) Left linear Grammar.

1) **Right linear Grammar :-** If all production of a CFG are of the form

$A \to wB$ OR $A \to w$, where A & B are non-terminal and $w \in V_T^*$, then we say that grammar is a right linear.

f.e $A \rightarrow aB|a$
$B \rightarrow aB|bB|a|b$

In above, all the non-terminals on R.H.S are place on the right side of terminals. ∴ these types of grammar called Right linear grammar.

2 Left linear Grammar :- If all the productions of a CFG are of the form $A \rightarrow Bw$ or $A \rightarrow w$, we called it left linear grammar.

for Example :- $A \rightarrow Ba$
$B \rightarrow Ba|Bb|a|b$

In above all the non-terminals are at left side. ∴ these type of grammar are left linear grammar.

Note :- A regular language can be completely either left linear or Right linear. But combination of both left linear & Right linear cannot be regular.

CONVERT REGULAR GRAMMAR TO F.A

Given a regular grammar 'G', a finite Automata accepting L(G) can be obtained as follows:

1) The number of states in the automata will be equal to the number of non-terminals plus one. Each state in automata represents each non terminals in regular grammar. The additional state will be the final state of the automata. If start symbol in grammar derives 'ε' then start state is also final in Automata

2) The transitions for automata are obtained as follows:

a) for every production $A \to aB$ makes $\delta(A,a)=B$ that is make a label 'a' from A to B

b) for every production $A \to a$ make $\delta(A,a)=$ finalstate

c) For every Production $A \to \epsilon$, make $\delta(A,\epsilon)=A$ and A will be final state

Example:- Consider the following grammar

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

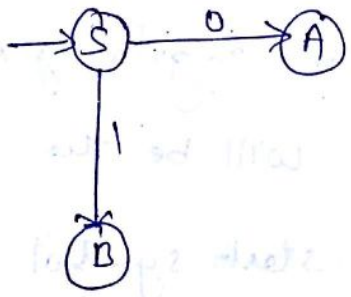
$A \rightarrow 0S \mid 1B \mid 1$

$B \rightarrow 0A \mid 1S$

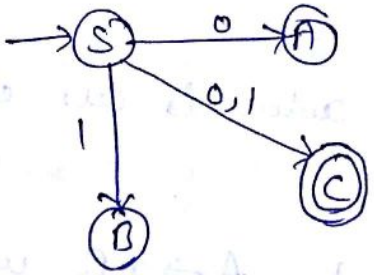
Find the automation for this grammar.

Answer:-

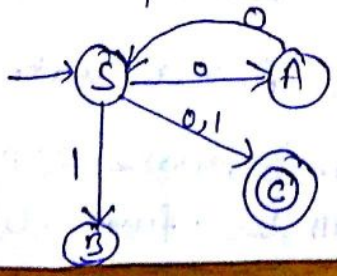
Step 1:- For production $S \rightarrow 0A$ & $S \rightarrow 1B$



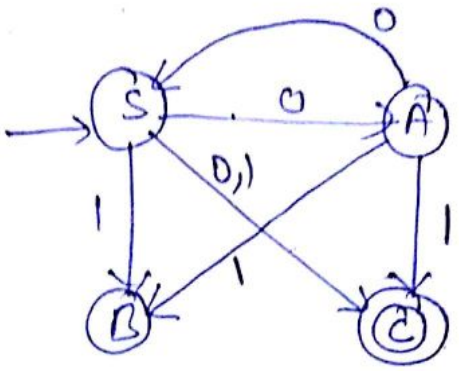
Step 2 :- For production $S \rightarrow 0 \mid 1$



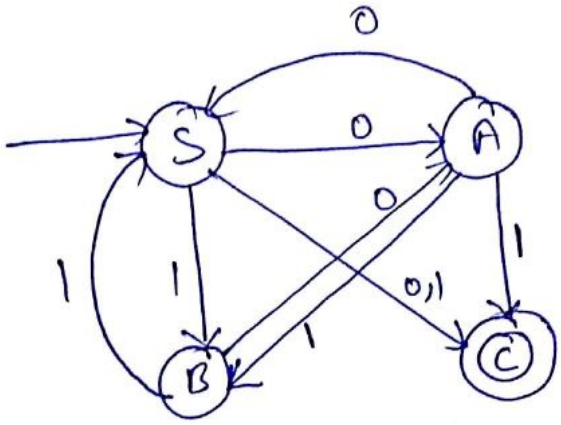
Step 3:- For production $A \rightarrow 0S$



Step 4:- For the production $A \rightarrow 1B$



Step 5:- For the productions $B \rightarrow 0A$ and $B \rightarrow 1S$



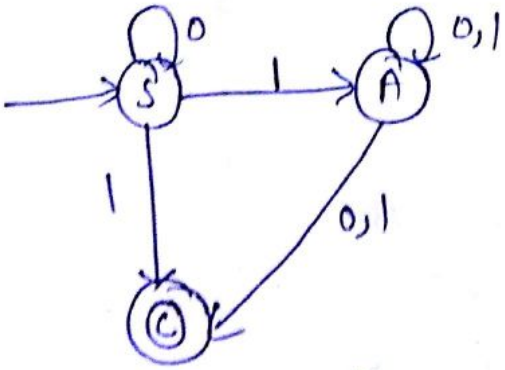
Answer

Example:- Give the automaton for the following grammar:

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

$A \rightarrow 0A \mid 1A \mid 0 \mid 1$

Ans:-



Example: Construct a FA for the following grammar.

$A \rightarrow 0B \mid 1E$

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

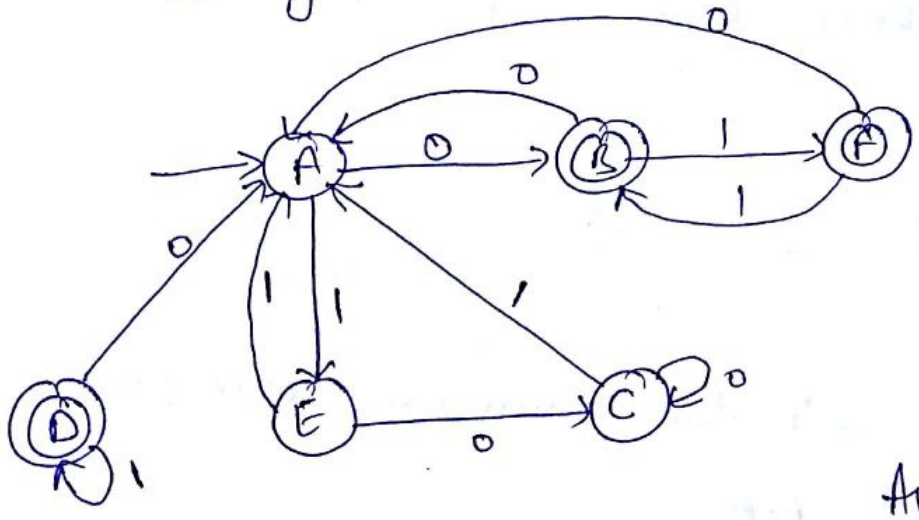
$C \rightarrow 0C \mid 1A$

$D \rightarrow 0A \mid 1D \mid \epsilon$

$E \rightarrow 0G \mid 1A$

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

Ans: The finite state Automata for grammar is given as:



Answer

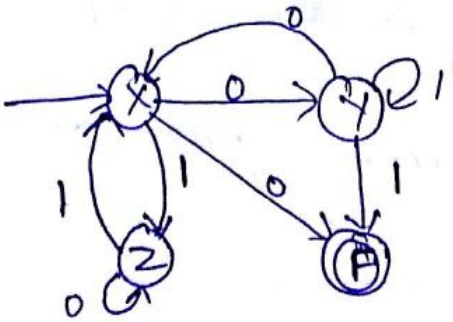
Example: Construct FA for the given grammar as:

$X \rightarrow 0Y \mid 0 \mid 1Z$

$Y \rightarrow 0X \mid 1Y \mid 1$

$Z \rightarrow 0Z \mid 1X$

Answer:



Answer

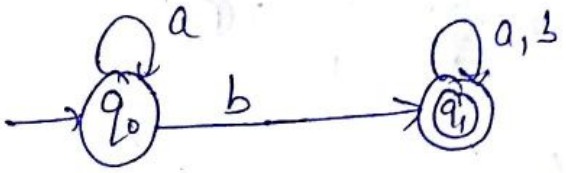
CONVERT F.A TO REGULAR GRAMMAR

Example I:- Construct a regular grammar G generating the regular set represented by:

$$P = a^*b(a+b)^*$$

Answer:-

Step I:- Design a F.A corresponding to the above Regular Expression as:



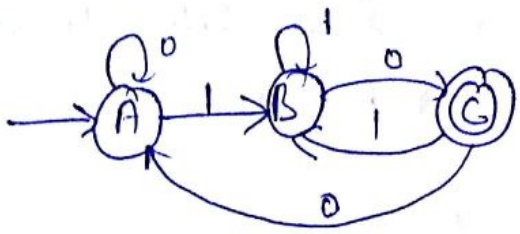
Step II:- write the grammar corresponding to the above F.A as:

Let $G = \{\{q_0, q_1\}, \{a, b\}, P, q_0\}$, where P

$$q_0 \rightarrow aq_0 \mid bq_1 \mid b$$
$$q_1 \rightarrow aq_1 \mid bq_1 \mid a \mid b$$

is required grammar.

Example 1: write a CFG corresponding to given F.A



Ans: let $G = (\{A, B, C\}, \{0, 1\}, P, A)$, where

P is:

$A \rightarrow 0A \mid 1B$

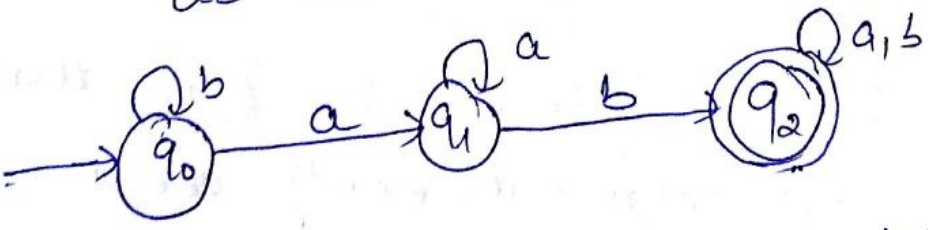
$B \rightarrow 1B \mid 0C \mid 0$

$C \rightarrow 1B \mid 0A$

is required grammar. (Ans)

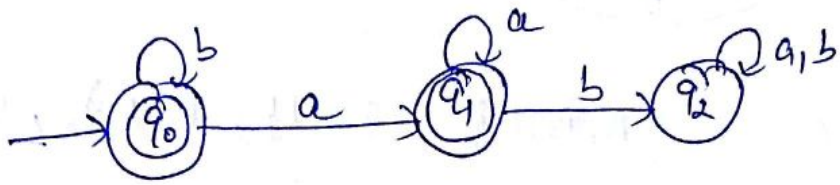
Example: Generate CFG corresponds to all words that don't have the substring "ab".

Ans: Step 1: first we construct the F.A corresponds to the language which have the substring 'ab' as:



Step 2: Now Take the Compliment of Step I which accept the language which don't have the substring 'ab'.

For the complement, we change the final state to non final state & non-final state to final state in the previous made F.A as:



Step 3: Now write the grammar for the above FA as:

Let $G = (\{q_0, q_1, q_2\}, \{a, b\}, P, q_0)$

where P is given as:

$$q_0 \rightarrow bq_0 \mid aq_1 \mid b \mid a$$
$$q_1 \rightarrow aq_1 \mid bq_2 \mid a$$
$$q_2 \rightarrow aq_2 \mid bq_2$$

is required grammar. (Answer)

Example: Design CFG for $\Sigma = \{a, b\}$ that generates
a) all strings with exactly one a.
b) all strings with at least one a.
c) all strings with at least 3 a's.

Ans: a) All strings with exactly one a.

→ We can write a Regular Expression corresponds to the given problem as:

$$R.E = b^* a b^*$$

We can write CFG for above R.E using two different Methodes either we directly write the CFG OR first we construct the FA Corresponds to the above R.E & then write the CFG by seeing the moves in F.A.

Method I.

$R.E = b^* a b^*$

Let G is CFG defined as:

$G = (V_n, V_T, P, S)$

Where $V_n = \{S, A\}$

$V_t = \{a, b\}$

P is defined as:

{

$S \rightarrow AaA$

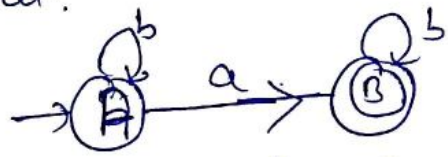
$A \rightarrow bA \mid \epsilon$

}.

Ans.

Method II

Construct FA for $R.E = b^* a b^*$ as:



Now write CFG for the F.A as.

$A \rightarrow bA \mid aB \mid a$

$B \rightarrow bB \mid b$

Where $V_N = \{A, B\}$

$V_T = \{a, b\}$

$P = \{A \rightarrow bA \mid aB \mid a$

$B \rightarrow bB \mid b$

}.

Ans.

b) all strings will at least one a'

Ans: We can write a Regular Expression (R.E) corresponding to the given problem as!

$$R.E = (a+b)^* a (a+b)^*$$

Now write the CFG corresponding to R.E as.

let G is CFG with $G = (V_N, V_T, P, S)$

where $V_N = \{S, A, B\}$

$V_T = \{a, b\}$

$P = \{ S \to AaA$

$A \to aA \mid bA \mid \wedge$

$\}$.

let derive a string baab

$S \Rightarrow AaA$
$\Rightarrow bAaA$
$\Rightarrow baAaA$
$\Rightarrow ba\wedge aA$
$\Rightarrow baabA$
$\Rightarrow baab\wedge$
$\Rightarrow baab$ Ans

(c) All strings with at least 3 a's.

Ans:- We can write a Regular Expression corresponding to the given problem as:

$$R.E = b^* a b^* a b^* a (a+b)^*$$

Now the CFG for above R.E is:

let G is CFG having:

$$G = (V_N, V_T, P, S)$$

where $V_N = \{S, A\}$

$V_T = \{a, b\}$

$P = \{$

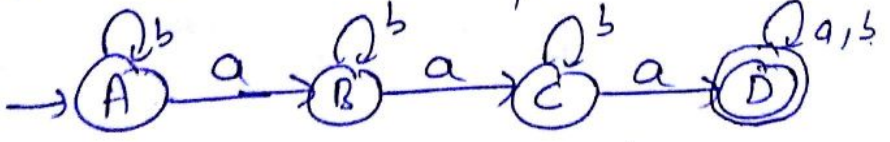
$S \rightarrow AaAaAaA$

$A \rightarrow aA \mid bA \mid \in$

$\}$

OR

Designe the FA corresponds to above R.E as.



Now we can write the CFG corresponds to the give F.A as:

Let G be CFG having:

$V_T = \{A, B, C, D\}$

$V_N = \{a, b\}$

$P = \{$

$A \rightarrow bA \mid aB$

$B \rightarrow bB \mid aC$

$C \rightarrow bC \mid aD \mid a$

$D \rightarrow aD \mid bD \mid a \mid b$

$\}$

Let derive a string "aaaaaba"

$A \Rightarrow aB$

$\Rightarrow aaC$

$\Rightarrow aaaD$

$\Rightarrow aaaaD$

$\Rightarrow aaaaaD$

$\Rightarrow aaaaabD$

$\Rightarrow aaaaaba$ (Ans)

CLOSURE PROPERTY OF CFL

→ Context-free language are closed under:

1) UNION
2) Concatenation
3) Kleen Star operation

① UNION :-

Let L_1 and L_2 be two Context free languages. Then $L_1 \cup L_2$ is also context free.

Ex: Example :- • Let $L_1 = \{a^n b^n, n>0\}$. corresponding grammar G_1 will have P: $S_1 \rightarrow aS_1b \mid ab$

• Let $L_2 = \{c^m d^m, m \geq 0\}$. Corresponding grammar G_2 will have P: $S_2 \rightarrow cS_2d \mid \wedge$

• The UNION of L_1 and $L_2 = L_1 \cup L_2 = \{a^n b^n\} \cup \{c^m d^m\}$ $m \geq 0$, $n > 0$

The corresponding grammar G will have Production as:

$$S \rightarrow S_1 | S_2.$$

2 Concatenation:-

If L_1 and L_2 are context free language, then $L_1 \cdot L_2$ is also context free.

Ans: Example:-

Concatenation of language L_1 & L_2 with the same grammar as in UNION will be given as $L = L_1 \cdot L_2$.

& the corresponding grammar G will have the production as: $\boxed{S \rightarrow S_1 S_2}$

Ans.

3. Kleen Star :-

If L is a context free language, then L^* is also context free.

Thm :- Example :- Let $L = \{a^n b^n, n \geq 0\}$, corresponding grammar G will have $P: S \to aSb \mid \in$

Kleene star $L_1 = \{a^n b^n\}^*$

The corresponding grammar G_1 will have production of the form :

$$S_1 \to SS_1 \mid \in$$

Hence

NOTE :- Context free language are not closed under →

(1) intersection :- If L_1 & L_2 are CFG then $L_1 \cap L_2$ not CFG.

(2) Complement :- If L_1 is CFG then $\overline{L_1}$ is not CFG.

* If L_1 is regular language and L_2 is CFG then $L_1 \cap L_2$

is a Context free language.

PREPAIRED BY → DEEPAK SINGLA
A.P (CSE), PIET

PDA

(PUSH DOWN AUTOMATA)

→ We have seen that the Regular language are accepted by finite Automata.

→ let us consider the $L = \{a^n b^n \mid n \geq 1\}$. This is a context free language but not a regular language (proved it by Pumping lemma).

→ So if the above grammar is not a regular grammar, so we cannot draw its FA.

→ We can not design the FA of language $\{a^n b^n\}$ as it has to remember the number of a's in a string and so it will require an infinite memory to store infinite number of states.

→ This difficulty can be avoided by adding an

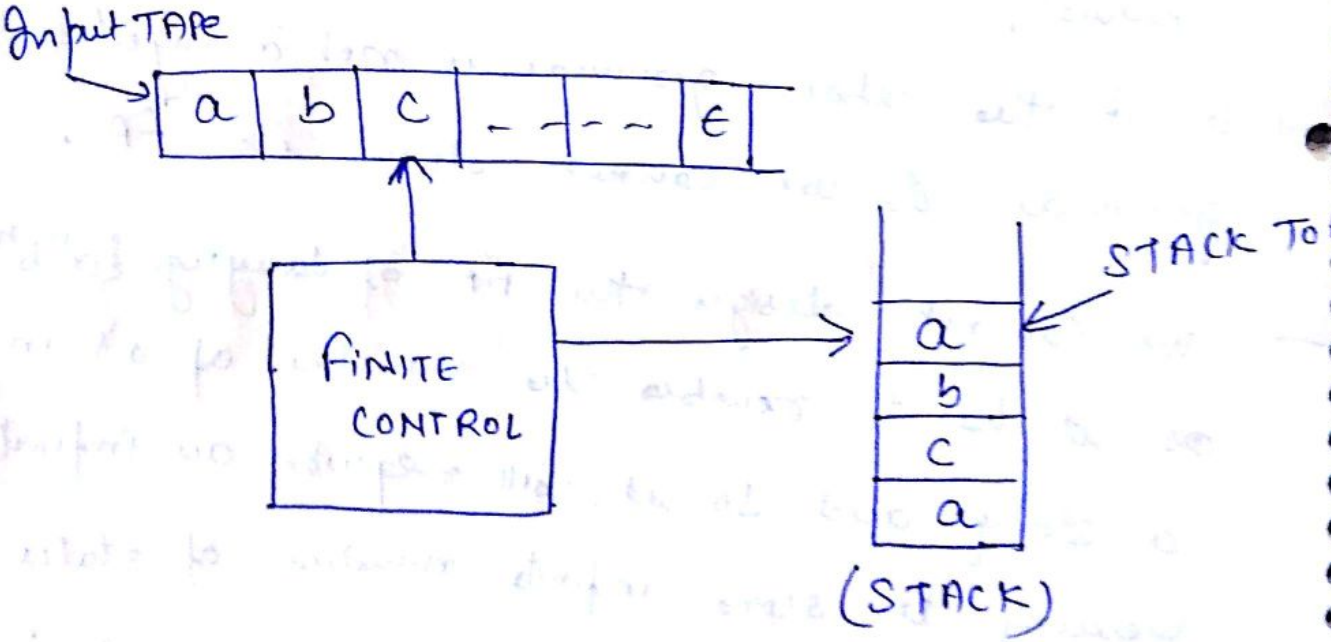
auxiliary memory in the form of STACK.

→ Now the a's in the given string are added to the stack & when the symbol b is encountered in input string, an 'a' is removed from the stack.

→ ∴ matching of number of a's & b's is done.

→ This type of arrangement where a finite Automata has a stack leads to the generation of PUSH DOWN AUTOMATA (PDA).

STRUCTURE OF PDA :-



→ In this diagram, there is a input tape,

Example: Construct the Push Down Automata for the following language:

$$L = \{a^n b^{n+1} \mid n = 1, 2, 3 \cdots\}$$

Ans: The language accepted by PDA is

$$L = \{abb, aabbb, aaabbbb \cdots\}$$

Let the PDA be

$$P = \{Q, \Sigma, \Gamma, \delta, S, F\}$$

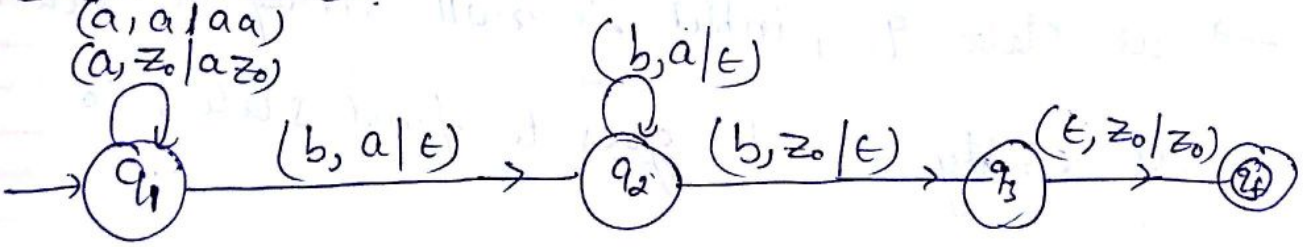
$$Q = \{q_0, q_1, q_2, q_3, q_4\}$$

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

$$S = \{q_1\}$$

$$F = \{q_4\}$$

STATE DIAGRAM:



TRANSITION function:-

1) $\delta(q_1, a, z_0) \Rightarrow (q_1, az_0)$

2) $\delta(q_1, a, a) \Rightarrow (q_1, aa)$ [2]

3) $\delta(q_1, b, a) \Rightarrow (q_2, \epsilon)$

4) $\delta(q_2, b, a) \Rightarrow (q_2, \epsilon)$

5) $\delta(q_2, b, Z_0) \Rightarrow (q_3, \epsilon)$

6) $\delta(q_3, \epsilon, Z_0) \Rightarrow (q_f, Z_0)$

→ Transition ① & ② we are pushing all the a's on the stack

→ Transition ③ & ④ pop all the a's when machine start reading b's from input string

→ When m/c reads last 'b', then it change to the state q_3

→ at state q_3, input is null string & stack is empty & m/c goes to final state, so the string is accepted.

from which finite control reads the input and same time it reads the symbol from stack ToP.

→ Now it depends on finite control, that what is the next state and what will happen with stack TOP.

FORMAL DEFINATION Of PDA

→ A push down automata is a system, which is mathematically defined as:

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

Where:

Q: Finite set of states

Σ: Input symbol

δ: Transition function

q_0: Initial state

Z_0: Bottom of stack

F: Set of final state

Γ: stack alphabet.

TYPES OF PDA

It is of 2 type!

1) DPDA (Deterministic PDA)

2) NDPDA (non-Deterministic PDA)

1) DPDA :- A pushdown automata $P = \{Q, \Sigma, \delta, S, F, \Gamma\}$ is said to be deterministic if it is defined in defination of PDA with the same no. of tuples, the only transition function is define in the form of:

$$\delta : Q \times \{\Sigma \cup \in\} \times \Gamma \rightarrow Q \times \Gamma^*$$

where $q \in Q$, $a \in (\Sigma \cup \wedge)$ and $b \in \Gamma$

2) NDPDA :- A pushdown automata $P = (Q, \Sigma, \delta, S, F, \Gamma)$ is said to be nondeterministic if it is defined in the

defination of PDA with same no. of tuples with transition fun" given as!

$$\delta : Q \times \{\Sigma \cup \epsilon\} \times \Gamma \rightarrow 2^{(Q \times \Gamma^*)}$$

* means if we are in one state(Q) & read any $\{\Sigma \cup \epsilon\}$ input symbol & read top of stack(Γ) then we move to many state & push many symbols in to the stack..

→ Deterministic PDA can recognize all deterministic CFL while Non Deterministic PDA can recognize all CFL

→ If there is only one computation exists for all accepted strings this is called (DPDA) & the languages of these strings is a Deterministic Context-free language.

* NOT ALL CFL ARE DETERMINSTIC

→ Because of the previous reason, DPDA is strictly weaker variant of PDA.

→ If there exist more than one choice of accepting all string then it is called Non-Deterministic PDA

→ The problem with this (NPDA) is due to execution of multiple instance it affect the automaton performance & lead to costly operations.

→ In the last we say that NPDA is more power full than DPDA & the language which is accepted by NPDA cannot be accepted by DPDA.

DPDA NUMERICAL 7

Example I:- Design a PDA for the language

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

Ans: The language Accepted by PDA is

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

Now let us assume that PDA be P.

$$P = (Q, \Sigma, \Gamma, \delta, S, f)$$

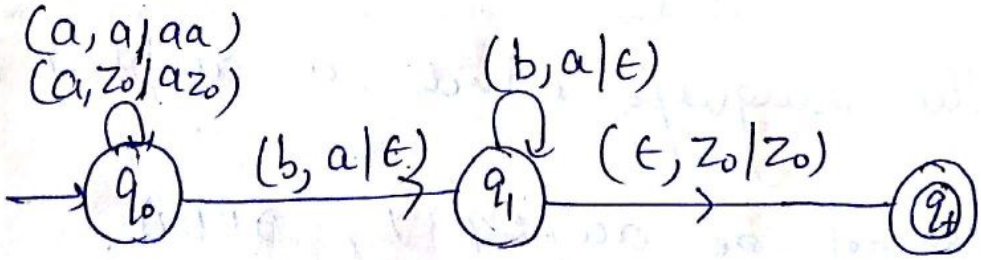
$$Q = \{q_0, q_1, q_f\}$$

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

$$\Gamma = \{Z_0\}$$

$$f = \{q_f\}.$$

PDA STATE DIAGRAM



Transistion function of PDA :-

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

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

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

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

5) $\delta(q_1, \epsilon, z_0) = (q_f, z_0)$

Example: Take a string 'aabb' & check the above transitions works or not?

S.No.	State	Unread input	Stack	Transition used
1.	q_0	aabbϵ	z_0	–
2.	q_0	abbϵ	az_0	1
3.	q_0	bbϵ	aaz_0	2
4	q_1	bϵ	az_0	3
5.	q_1	ϵ	z_0	4
6.	q_f	ϵ	z_0	5.

(Accepted)

Example:- Construct PDA that accepts the language

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

Ans:- The language accepted by PDA is:

$$L = \{\epsilon, 01, 0011, 000111 \text{ — } \}$$

Now let us assume that PDA be P defined as:

$$P = (Q, \Sigma, \Gamma, \delta, S, F)$$

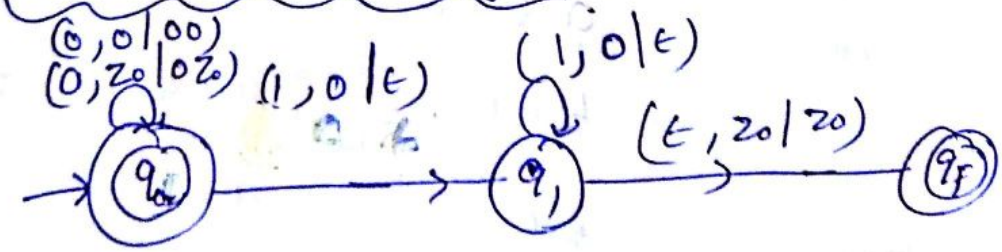
where $Q = \{q_0, q_1, q_f\}$

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

$\Gamma = \{Z_0\}$

$F = \{q_0, q_f\}$

PDA STATE DIAGRAM:-



(Ans)

Example! Construct a PDA which accepts the language $L = \{ w \in \{a,b\}^* \mid w$ has equal no. of a's and b's $\}$

Ans!: The language accepted by PDA is.

$L = \{ ab, aabb, baba, bbaa, abab \ldots \}$

Now let us assume the PDA be P defined as:

$$P = (Q, \Sigma, \Gamma, \delta, S, F)$$

where $Q = \{q_0, q_f\}$

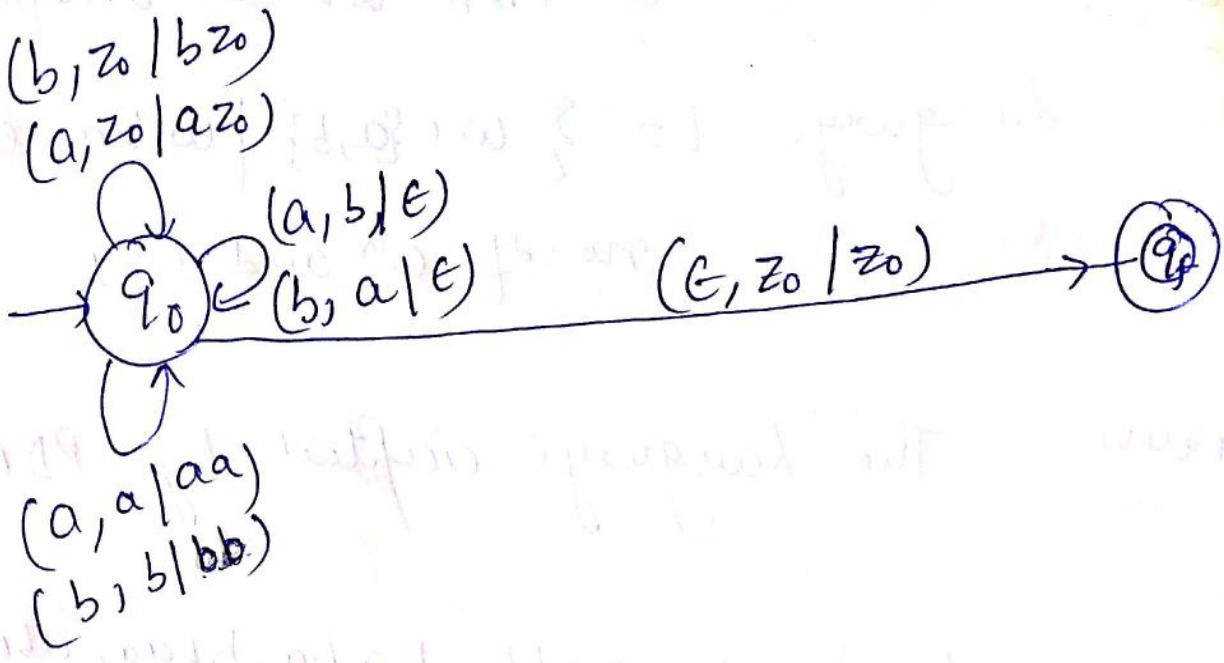
$\Sigma = \{a, b\}$

$\Gamma = \{z_0\}$

$F = \{q_f\}$.

PDA STATE DIAGRAME!-

It is shown as!



Transition function of PDA :-

1) $\delta(q_0, a, z_0) \Rightarrow (q_0, az_0)$

2) $\delta(q_0, b, z_0) \Rightarrow (q_0, bz_0)$

3) $\delta(q_0, a, a) \Rightarrow (q_0, aa)$

4) $\delta(q_0, b, b) \Rightarrow (q_0, bb)$

5) $\delta(q_0, a, b) \Rightarrow (q_0, \epsilon)$

6) $\delta(q_0, b, a) \Rightarrow (q_0, \epsilon)$

(7) $\delta(q_0, \epsilon, z_0) \Rightarrow (q_f, z_0)$

Ex. Example: Let us check the string 'baba' for the different transitions as:

S.No.	STATE	Unread I/P	STACK	Transition used
1.	q_0	babaϵ	Z_0	-
2.	q_0	abaϵ	bZ_0	2
3.	q_0	baϵ	Z_0	5
4.	q_0	aϵ	bZ_0	2
5.	q_0	ϵ	Z_0	5
6	q_f	ϵ	Z_0	7

So, the string is accepted

(Ans)

Example 3:- Construct a PDA for the language

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

Ans: The language accepted by the PDA is,

$$L = \{ abc, aabbc, aabbccc \cdots \}$$

Let us assume the PDA be P defined as

$$P = (Q, \Sigma, \Gamma, s, F, \delta)$$

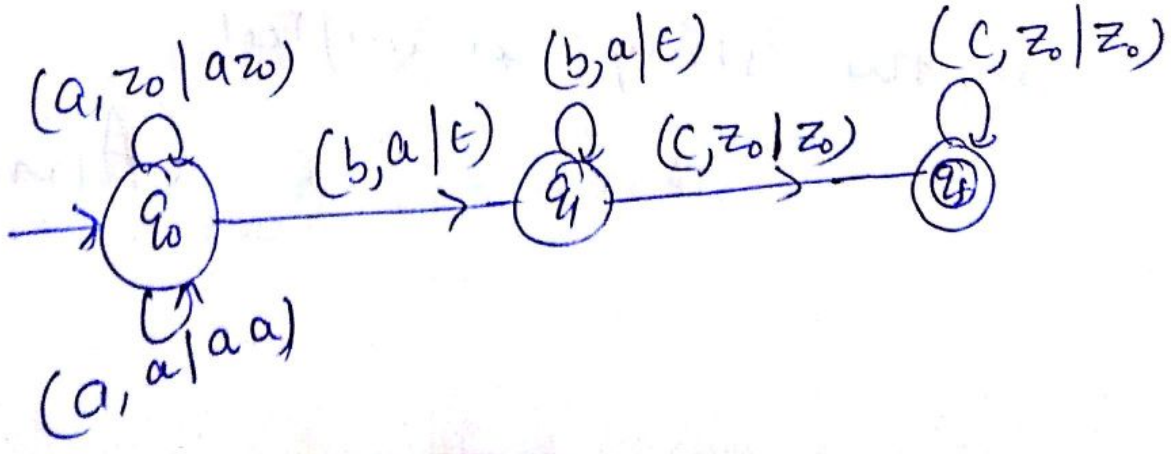
Where $Q = \{q_0, q_1, q_f\}$

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

$\Gamma = \{Z_0\}$

$F = \{q_f\}$

PDA STATE Diagram:-



Dead Configuration in PDA:-

→ let us accept a string 'bc' in the state diagram.

→ As on read input 'b' there is no transition which have the form $(b, Z_0 \mid bZ_0)$, therefore it leads to Dead configuration at state q_0 or we can say PDA will be Halt at state q_0.

Transition function for PDA:-

1) $\delta(q_0, a, Z_0) \Rightarrow (q_0, aZ_0)$
2) $\delta(q_0, a, a) \Rightarrow (q_0, aa)$
3) $\delta(q_0, b, a) \Rightarrow (q_1, \epsilon)$
4) $\delta(q_1, b, a) \Rightarrow (q_1, \epsilon)$
5) $\delta(q_1, c, Z_0) = (q_f, Z_0)$
6) $\delta(q_f, c, Z_0) = (q_f, Z_0)$

(Ans)

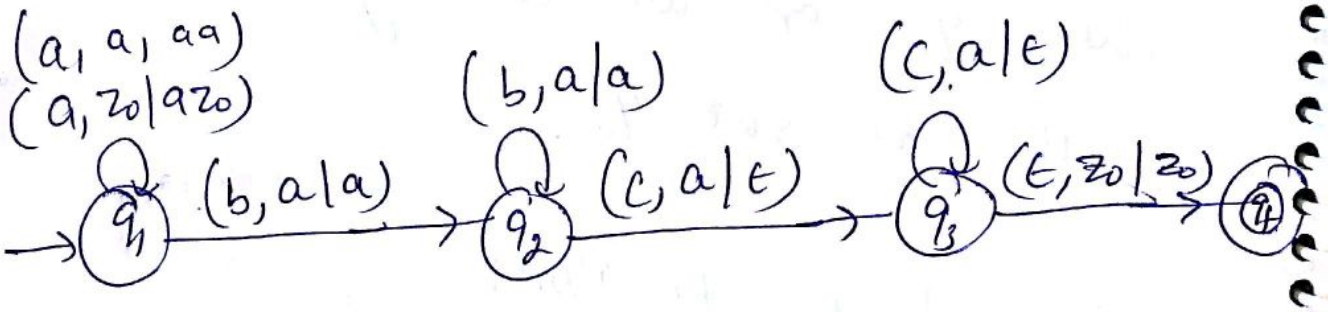
Example: 4 Construct a PDA for the language

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

Ans: = Language accepted by PDA is

$$L = \{ abc, aabcc --- \}$$

STATE DIAGRAM:-



Transition function:-

1) $\delta(q_1, a, z_0) \Rightarrow (q_1, az_0)$
2) $\delta(q_1, a, a) \Rightarrow (q_1, aa)$
3) $\delta(q_1, b, a) \Rightarrow (q_2, aa)$
4) $\delta(q_2, b, a) \Rightarrow (q_2, aa)$
5) $\delta(q_2, c, a) \Rightarrow (q_3, \epsilon)$
6) $\delta(q_3, c, a) \Rightarrow (q_3, \epsilon)$
7) $\delta(q_3, \epsilon, z_0) \Rightarrow (q_f, z_0)$ (Accepted)

(Ans)

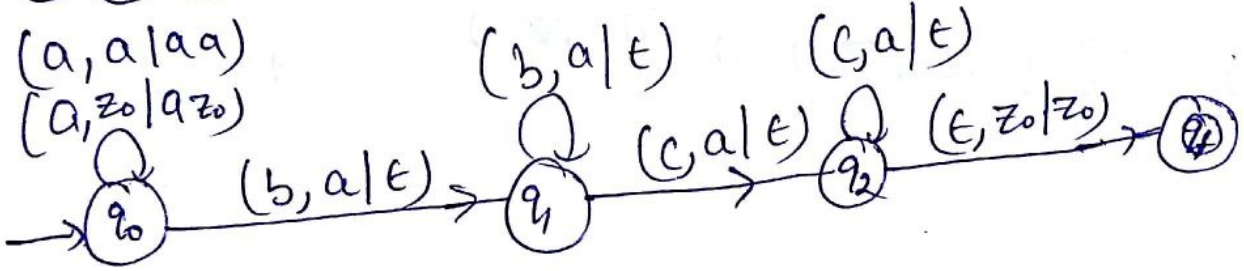
Example:4 Construct a PDA for the language.

$$L = \{ a^{m+n} b^m c^n \mid m, n \geq 1 \}$$

Ans: The language accepted by the PDA is

$$L = \{ aabc, aaaabbcc \cdots \}$$

STATE DIAGRAM:-



Let the PDA be defined as P which contains $P = \{ Q, \Sigma, \Gamma, \delta, S, F \}$

where $Q = \{ q_0, q_1, q_2, q_f \}$

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

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

$F = \{ q_f \}$

Transition function of PDA :-

1) $\delta(q_0, a, z_0) \Rightarrow (q_0, a z_0)$
2) $\delta(q_0, a, a) \Rightarrow (q_0, aa)$
3) $\delta(q_0, b, a) \Rightarrow (q_1, \epsilon)$
4) $\delta(q_1, b, a) \Rightarrow (q_1, \epsilon)$
5) $\delta(q_1, c, a) \Rightarrow (q_2, \epsilon)$
6) $\delta(q_2, c, a) \Rightarrow (q_2, \epsilon)$
7) $\delta(q_2, \epsilon, z_0) \Rightarrow (q_f, z_0)$

Ans

Example:- Construct a PDA for the language

$$L = \{ a^n b^{m+n} c^m \mid n, m \geq 1 \}$$

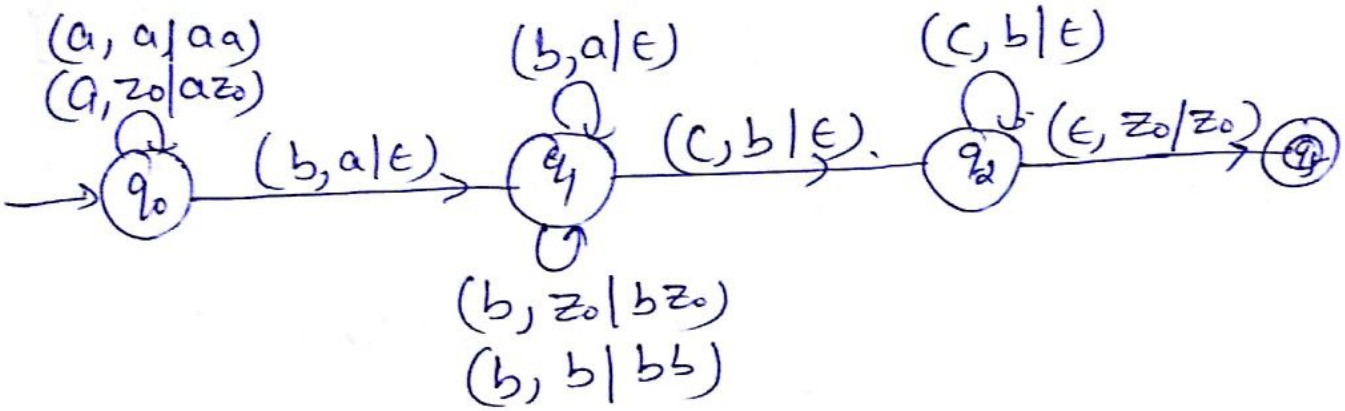
Ans:- We can rewrite the language as:

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

∴ The language accepted by PDA is

$$L = \{ abbc, aabbbcc, \ldots \}$$

STATE DIAGRAME of PDA :-



TRANSITION FUNCTION (δ) :-

1) $\delta(q_0, a, z_0) \Rightarrow (q_0, a z_0)$
2) $\delta(q_0, a, a) \Rightarrow (q_0, aa)$
3) $\delta(q_0, b, a) \Rightarrow (q_1, \epsilon)$
4) $\delta(q_1, b, a) \Rightarrow (q_1, \epsilon)$
5) $\delta(q_1, b, z_0) \Rightarrow (q_1, b z_0)$
6) $\delta(q_1, b, b) \Rightarrow (q_1, bb)$
7) $\delta(q_1, c, b) \Rightarrow (q_2, \epsilon)$
8) $\delta(q_2, c, b) \Rightarrow (q_2, \epsilon)$
9) $\delta(q_2, \epsilon, z_0) \Rightarrow (q_f, z_0)$

(Answer)

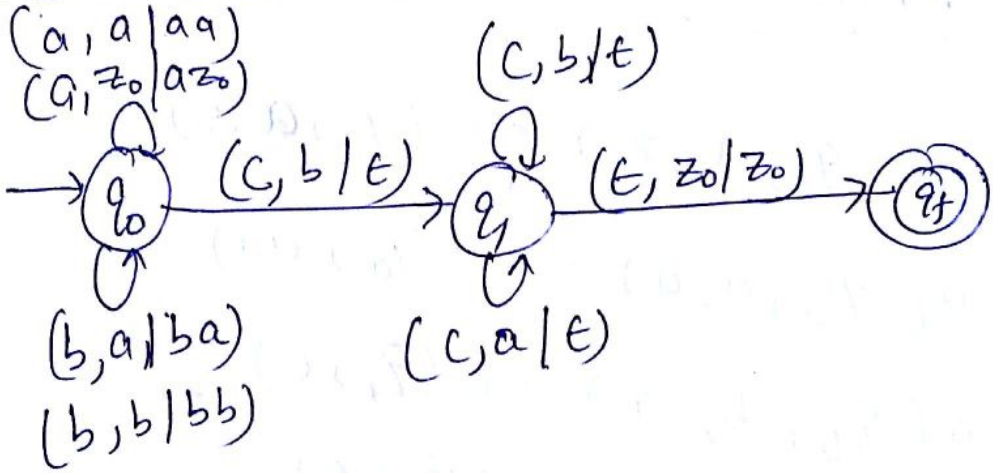
Example: Construct a PDA for the language

$$L = \{ a^n b^m c^{n+m} \mid n, m \geq 1 \}$$

Ans: The language accepted by the PDA:

$$L = \{ abcc, aabbcccc \cdots \}$$

STATE DIAGRAM:-



Transition function (δ):-

1) $\delta(q_0, a, z_0) \Rightarrow (q_0, az_0)$
2) $\delta(q_0, a, a) \Rightarrow (q_0, aa)$
3) $\delta(q_0, b, a) \Rightarrow (q_0, ba)$
4) $\delta(q_0, b, b) \Rightarrow (q_0, bb)$
5) $\delta(q_0, c, b) \Rightarrow (q_1, \epsilon)$
6) $\delta(q_1, c, b) \Rightarrow (q_1, \epsilon)$

7) $\delta(q_1, c, a)$
$\Downarrow$
(q_1, ϵ)
8) $\delta(q_1, \epsilon, z_0)$
$\Downarrow$
(q_f, z_0)

Ans

Example:- Construct a PDA for the regular expression.

$r = 0^* 1^+$

Answer:- We can write the above R.E in the form of language as:

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

So the language accepted by PDA is

$$L = \{1, 01, 001, 0011 \ldots\}$$

Let PDA be $P = \{Q, \Sigma, \Gamma, \delta, S, F\}$

$Q = \{q_0, q_f\}$

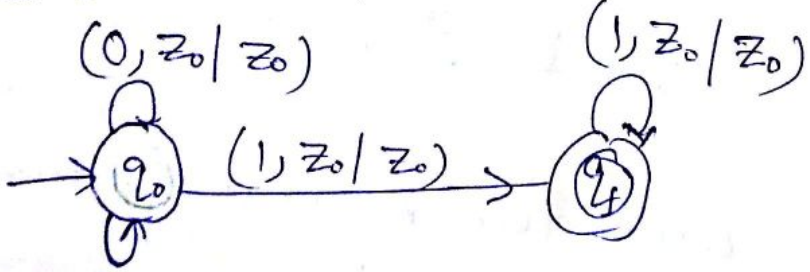
$S = \{q_0\}$

$F = \{q_f\}$

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

$\Gamma = \{Z_0\}$

STATE DIAGRAM:-



TRANSITION FUNCTION :-

1) $\delta(q_0, 0, z_0) \Rightarrow (q_0, z_0)$

2) $\delta(q_0, 1, z_0) \Rightarrow (q_f, z_0)$

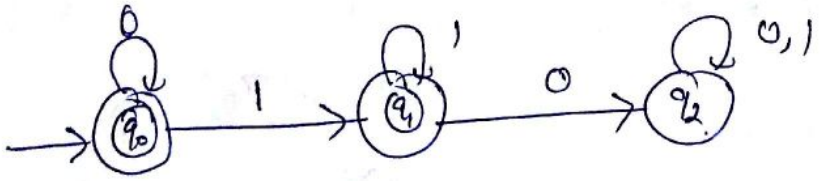
3) $\delta(q_f, 1, z_0) \Rightarrow (q_f, z_0)$

→ Transition (1) shows, when machine reads 0's then it does nothing

→ transition (2), shows when 1 is encountered it reach to final state & do nothing

→ Transition (3) shows any no. of 1's encountered it will remain in the final state q_f. & string is accepted.

Example: write the regular expression for the given DFA & design a push down Automata



Ans we can write the Regular Expression for above DFA as:

$$R = 0^* 1^*$$

Now let the push down automata be P

$$P = \{Q, \Sigma, \Gamma, \delta, S, f\}$$

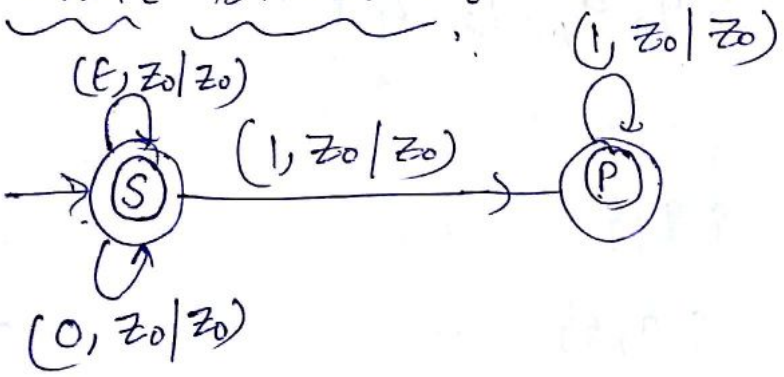
$$Q = \{S, P\}$$

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

$$\Gamma = \{Z_0\}$$

$$f = \{S, P\}$$

STATE DIAGRAM :-



TRANSITION FUNCTION :-

1) $\delta(S, \epsilon, Z_0) \Rightarrow (S, Z_0)$

2) $\delta(S, 0, Z_0) \Rightarrow (S, Z_0)$

3) $\delta(S, 1, Z_0) \Rightarrow (P, Z_0)$

4) $\delta(P, 1, Z_0) \Rightarrow (P, Z_0)$

(Accepted)

(Ans)

Example:- Design a PDA for the following language.

$$L = \{ a^n b^m \mid n > m \geq 0 \}$$

Ans:- The language accepted by PDA is

$$L = \{ a, aaabb, aab - \cdots \}$$

Let the PDA for the language is P:

$$P = \{ Q, \Sigma, \Gamma, s, \delta, f \}$$

$$Q = \{ q_0, q_1, q_2, q_f \}$$

$$s = \{ q_0 \}$$

$$f = \{ q_f \}$$

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

TRANSITION FUNCTION:-

1) $\delta(q_0, a, z_0) \Rightarrow (q_0, az_0)$

2) $\delta(q_0, a, a) \Rightarrow (q_0, aa)$

3) $\delta(q_0, \epsilon, a) \Rightarrow (q_0, \epsilon)$ 4) $(q_0, \epsilon, z_0) \Rightarrow (q_f, z_0)$

5) $\delta(q_0, b, a) \Rightarrow (q_1, \epsilon)$

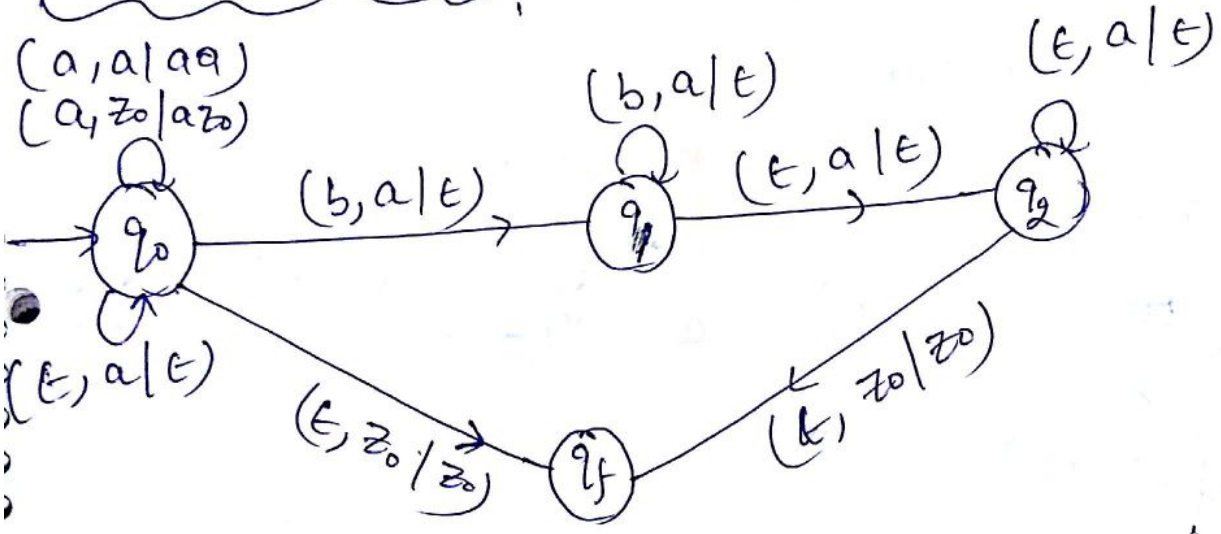
6) $\delta(q_1, b, a) \Rightarrow (q_1, \epsilon)$

7) $\delta(q_1, \epsilon, a) \Rightarrow (q_2, \epsilon)$

8) $\delta(q_2, \epsilon, a) \Rightarrow (q_2, \epsilon)$

9) $\delta(q_2, \epsilon, z_0) \Rightarrow (q_f, z_0)$

STATE DIAGRAM :-



For Example: Accept a string 'aaab'

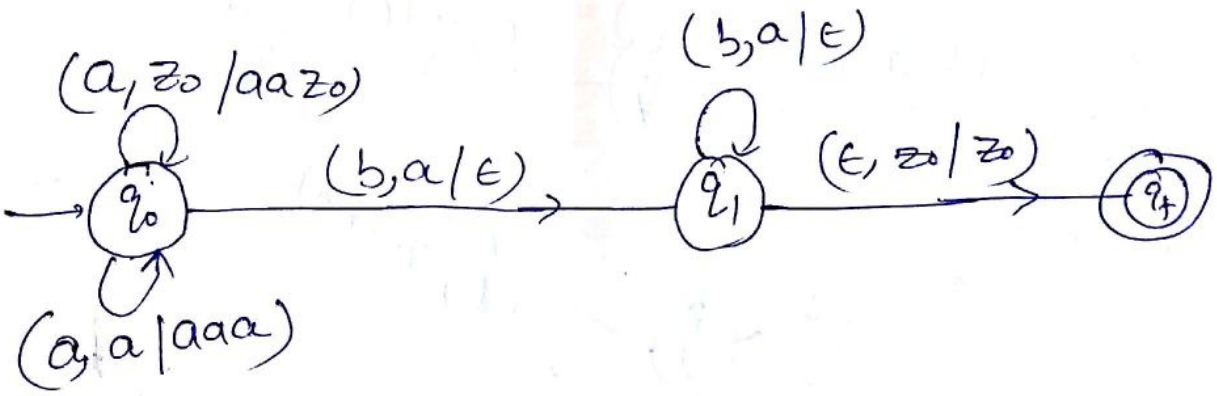
S.No.	Input string	State	stack	Transition used
1.	aaabε	q_0	z_0	–
2.	aabε	q_0	az_0	1
3.	abε	q_0	aaz_0	2
4.	bε	q_0	$aaaz_0$	2
5.	ε	q_1	aaz_0	5
6.	ε	q_2	az_0	7
7.	ε	q_2	z_0	8
8.	ε	q_f	z_0	9 (Accepted)

Ex: Construct a PDA for $L = \{a^n b^{2n} / n \geq 1\}$

Ans: The language accepted by PDA is

$L = \{abb, aabbbb, aaabbbbbb \ldots\}$

STATE DIAGRAM:-



TRANSITION FUNCTION:-

1) $\delta(q_0, a, z_0) \Rightarrow (q_0, aaz_0)$
2) $\delta(q_0, a, a) \Rightarrow (q_0, aaa)$
3) $\delta(q_0, b, a) \Rightarrow (q_1, \epsilon)$
4) $\delta(q_1, b, a) \Rightarrow (q_1, \epsilon)$
5) $\delta(q_1, \epsilon, z_0) \Rightarrow (q_f, z_0)$

→ From Transition (1) we push 2 a's corresponding to the 1 a read by the machine.

→ From transition 3 & 4, we pop single a corresponding to each b read by machine & move to final

Example:- Construct a PDA for the language.

$$L = \{ wcw^R \mid w \in (a+b)^+ \}$$

Ans:- The above language shows the odd no. of palindrom string.

∴ The language accepted by PDA is

$$L = \{ aabcbaa, abcba, aacaa \cdots \}$$

Let the pda be P is defin as:

$$P = \{ Q, \Sigma, \Gamma, \delta, S, F \}$$

where. $Q = \{q_0, q_1, q_f\}$

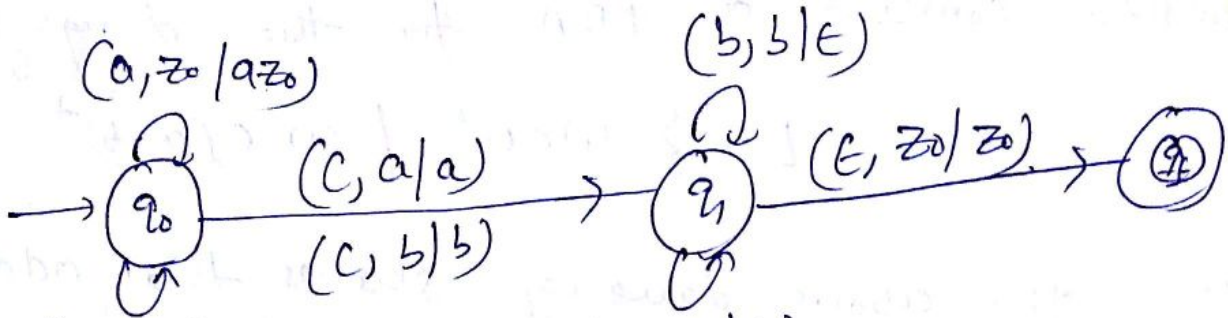
$\Sigma = \{a, b\}$

$\Gamma = \{z_0\}$

$S = \{q_0\}$

$F = \{q_f\}$

STATE DIAGRAM :-



TRANSITION FUNCTION :-

1) $\delta(q_0, a, z_0) \Rightarrow (q_0, az_0)$
2) $\delta(q_0, a, a) \Rightarrow (q_0, aa)$
3) $\delta(q_0, b, a) \Rightarrow (q_0, ba)$
4) $\delta(q_0, a, b) \Rightarrow (q_0, ab)$
5) $\delta(q_0, b, b) \Rightarrow (q_0, bb)$
6) $\delta(q_0, b, z_0) \Rightarrow (q_0, bz_0)$
7) $\delta(q_0, c, a) \Rightarrow (q_1, a)$
8) $\delta(q_0, c, b) = (q_1, b)$
9) $\delta(q_1, b, b) = (q_1, \epsilon)$
10) $\delta(q_1, a, a) \Rightarrow (q_1, \epsilon)$
11) $\delta(q_1, \epsilon, z_0) \Rightarrow (q_f, z_0)$

z_0

for Example:- Accept a String abacaba

S.No.	State	Output String	Stack	Transition used.
1.	q_0	abacabaε	z_0	–
2.	q_0	bacabaε	az_0	1
3.	q_0	acabaε	baz_0	3
4.	q_0	cabaε	$abaz_0$	4
5.	q_0	abaε	$abaz_0$	7
6.	q_1	baε	baz_0	10
7.	q_1	aε	az_0	9
8.	q_1	ε	z_0	10
9.	q_f	ε	z_0	11

(Accepted)

(Answer)

NPDA NUMERICALS

Example 1: Construct a PDA for the given Language

$$L = \{ ww^R \mid w \in (a,b)^+ \}$$

Ans: The above language shows the string of even palindrom. & this PDA cannot be constructed with DPDA

So, to construct it we use the NPDA.

The language accepted by NPDA is:

$$L = \{ aa, abba, aaaa ---- \}$$

Let the PDA be P defined as:

$$P = (Q, \Sigma, \delta, \Gamma, S, F)$$

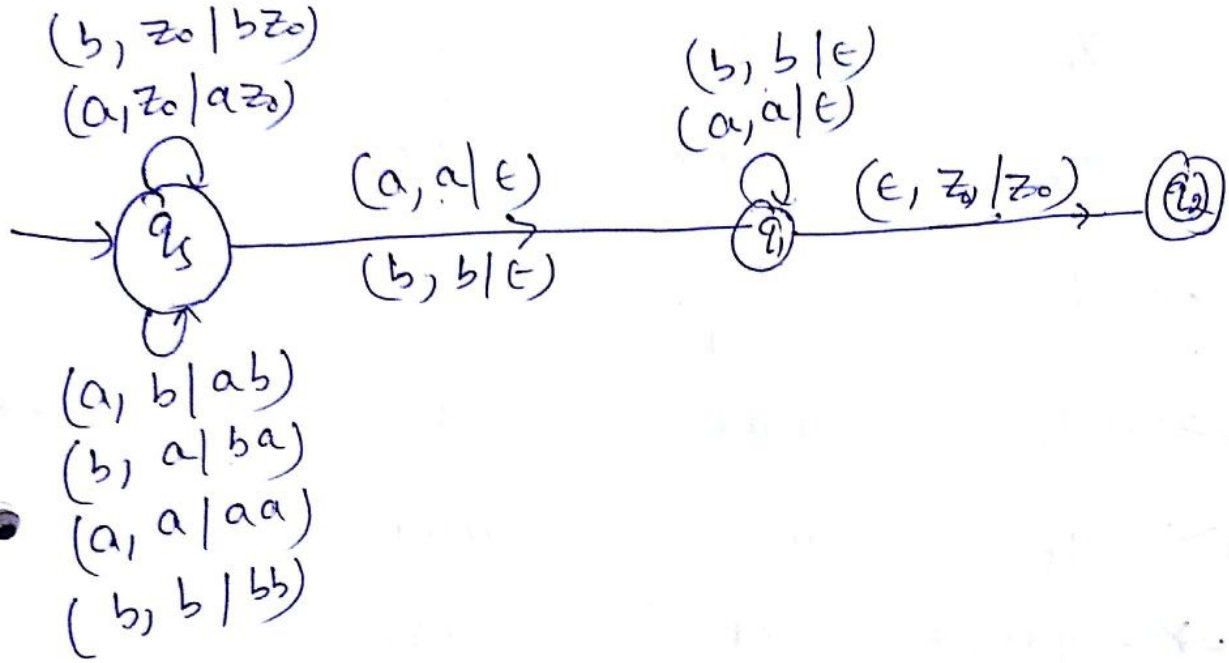
where $Q = \{q_s, q_1, q_2\}$ $F = \{q_2\}$

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

$\Gamma = \{Z_0\}$

$S = \{q_s\}$

STATE DIAGRAM:-



TRANSITION FUNCTION:-

1) $\delta(q_0, a, z_0) = (q_0, az_0)$
2) $\delta(q_0, b, z_0) = (q_0, bz_0)$
3) $\delta(q_0, a, b) = (q_0, ab)$
4) $\delta(q_0, b, a) = (q_0, ba)$
5) $\delta(q_0, a, a) = (q_0, aa)$
6) $\delta(q_0, a, a) = (q_1, \epsilon)$
7) $\delta(q_0, b, b) = (q_0, bb)$
8) $\delta(q_0, b, b) = (q_1, \epsilon)$
9) $\delta(q_1, a, a) = (q_1, \epsilon)$
10) $\delta(q_1, b, b) = (q_1, \epsilon)$
11) $\delta(q_1, \epsilon, z_0) = (q_2, z_0)$

Example: Accept a string 'aaaa'

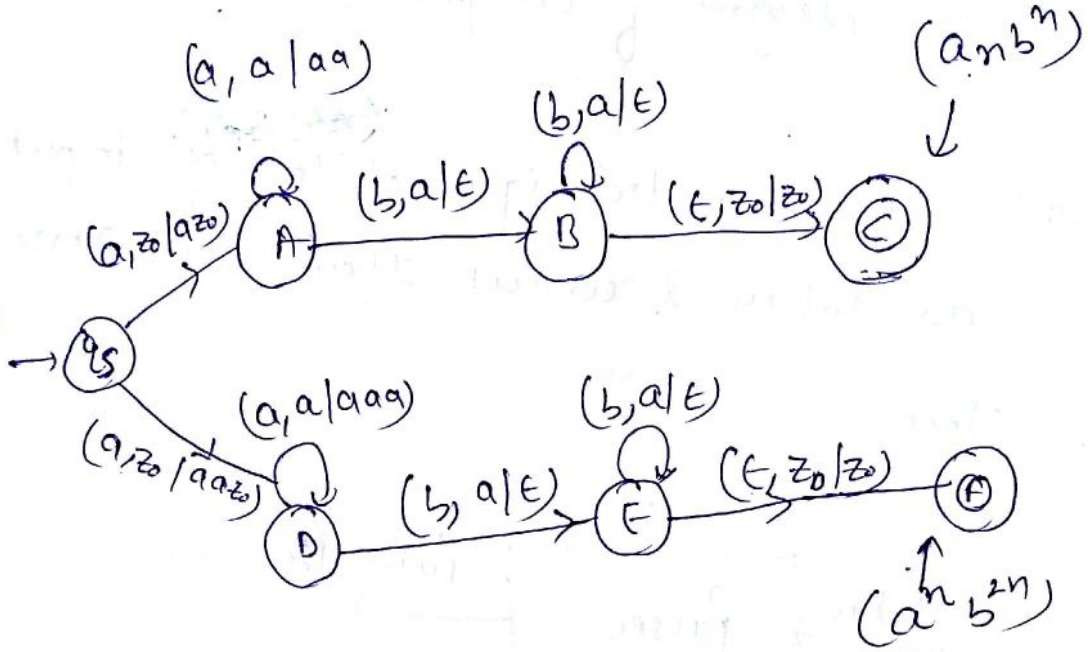
S.No.	State	Input string	Stack	Transition used
1.	q_s	aaaaϵ	Z_0	-
2.	q_s	aaaϵ	aZ_0	1
3.	3.1) q_s	aaϵ	aaZ_0	2
	OR			
	3.2) q_1	aaϵ	Z_0	6 (Halt)
4.	4.1) q_s	aϵ	aaaZ_0	2
	4.2) q_1	aϵ	aZ_0	6
5)	5.1) q_s	ϵ	aaaaZ_0	2 (Halt)
	5.2 q_1	ϵ	aaZ_0	6 (Halt)
	5.3) q_1	ϵ	Z_0	9
6.	q_2	ϵ	Z_0	11 (Accept)

∴ the above string is accepted by designing the NPDA of given language. (Ans)

Example:- Construct a NPDA for the language

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

Ans:- STATE DIAGRAM:-



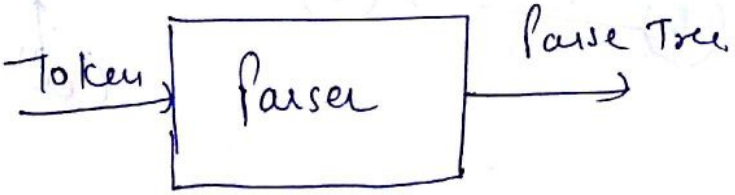
TRANSITION FUNCTION:-

1) $\delta(q_5, a, z_0) \Rightarrow (A, az_0)$
2) $\delta(q_5, a, z_0) \Rightarrow (D, aaz_0)$
3) $\delta(A, a, a) \Rightarrow (A, aa)$
4) $\delta(A, b, a) \Rightarrow (B, \epsilon)$
5) $\delta(B, b, a) \Rightarrow (B, \epsilon)$
6) $\delta(B, \epsilon, z_0) \Rightarrow (C, z_0)$
7) $\delta(D, a, a) \Rightarrow (D, aaa)$
8) $\delta(D, b, a) \Rightarrow (E, \epsilon)$
9) $\delta(E, b, a) \Rightarrow (E, \epsilon)$
10) $\delta(E, \epsilon, z_0) \Rightarrow (D, z_0)$ (Accepted)

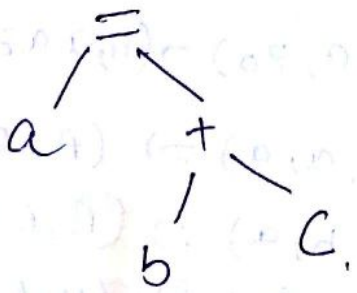
APPLICATION OF PDA:

→ The PDA is used in parsing Technique in the design of Compiler.

Parsing:- It is a technique of taking input as token & convert them into parse Tree.



f.e. $a = b + c$, the corresponding parse tree is



Types of Parser:-

1) Bottom-up Parser
2) Top-down Parser

1) Bottom-up Parser:- It generate the parse tree from leaves to root for a given input string.

Example:- Shift Reduce Parser

2) Top-Down Parser:- It generates the parse tree from root to leaves.

Example: predictive parser

SHIFT REDUCE PARSER:-

→ This is a type of Bottom-up parser that generates the parse tree from leaves to root.

→ In SRP, the input string will be reduced to the start symbol.

→ SHIFT-REDUCE Parser require 2 Data Structure
1) Input Buffer 2) STACK.

Example: perform shift reduce parser for the string $id_1 + id_2 * id_3$ for the given grammer as:

$E \rightarrow E+E$

$E \rightarrow E*E$

$E \rightarrow (E)$

$E \rightarrow id$

Ans:

STACK	Input String	Action
$	$id_1 + id_2 * id_3 \$$	SHIFT
$ id_1	$+ id_2 * id_3 \$$	Reduce $E \rightarrow id$
$ E	$+ id_2 * id_3 \$$	shift
$ $E+$	$id_2 * id_3 \$$	Shift
$ $E+id_2$	$* id_3 \$$	Reduce $E \rightarrow id$
$ $E+E$	$* id_3 \$$	Shift
$ $E+E*$	$id_3 \$$	Shift
$ $E+E*id_3$	$\$$	Reduce $E \rightarrow id$
$ $E+E*E$	$\$$	Reduce $E \rightarrow E*E$
$ $E+E$	$\$$	Reduce $E \rightarrow E+E$
$ E	$\$$	Accepted.