

MODULE-3

Context free grammar (CFG):-

A CFG 'G' is having four tuples $G = (V, T, P, S)$ where V represents set of variables that is also called as non-terminals and it is represented by capital letters. T represents set of terminal symbols and it is represented by lower case letters. S is a start symbol. P is a set of productions or rules that represent the recursive definition of a language. Each production consists of

i) A variable that is being defined by production, this variable is often called the head of the production.

ii) The production symbol is single arrow ($\rightarrow$)

iii) A string of 0 or more terminals followed by variables. This string is called the body of the production.

Example:- Valid productions:-

$$A \rightarrow aB$$

$$A \rightarrow a\bar{1}babbs$$

Invalid productions

$$aB \rightarrow A$$

$$AB \rightarrow a$$

Production:-

$$V = \{B, A\} \quad B \rightarrow A$$

$$T = \{a\} \quad A \rightarrow a$$

$$S = B$$

$$\boxed{\begin{array}{l} B \Rightarrow A \\ \Rightarrow a \end{array}}$$

1. Obtain a grammar to generate string consisting of any number of a's.

$$L = \{\epsilon, a, aa, aaa, \dots\}$$

$$S \rightarrow \epsilon \mid aS$$

$$1) S \Rightarrow aS$$

$$\Rightarrow a.\epsilon$$

$$2) S \Rightarrow aS$$

$$\Rightarrow aaS$$

$$\Rightarrow aa\epsilon$$

$$3) S \Rightarrow aS$$

$$\Rightarrow aaS$$

$$\Rightarrow aaaS$$

$$\Rightarrow aaa\epsilon$$

2. Obtain a grammar to generate string consisting of atleast one a.

$$S \rightarrow a | aS$$

3. Obtain a grammar to generate string consisting of any a's or b's.

$$S \rightarrow \epsilon | aS | bS$$

$$\begin{aligned} S &\Rightarrow aS \\ &\Rightarrow abS \\ &\Rightarrow abaaS \\ &\Rightarrow abaaS \end{aligned}$$

4. Obtain a grammar to generate string consisting of atleast two a's

$$S \rightarrow aa | aS$$

$$\begin{aligned} S &\Rightarrow aS \\ &\Rightarrow aaa \end{aligned}$$

5. Obtain a grammar to generate string consisting of even number of a's

$$A \rightarrow \epsilon | aaA$$

$$\begin{aligned} A &\Rightarrow aaA \\ &\Rightarrow aa\epsilon \end{aligned}$$

6. Obtain a grammar to generate string consisting of multiple of 3 a's

$$B \rightarrow \epsilon | aaaB$$

$$\begin{aligned} B &\Rightarrow aaaB \\ &\Rightarrow aaa\epsilon \end{aligned}$$

$$\begin{aligned} B &\Rightarrow aaaB \\ &\Rightarrow aaaaaaaB \\ &\Rightarrow aaaaaaa\epsilon \end{aligned}$$

7. Obtain a grammar to generate string consisting of a's and b's such that string length is multiple of 3.

$$S \rightarrow \epsilon | AAAS$$

$$A \rightarrow a | b$$

$$abbaab'$$

$$S \Rightarrow AAAS$$

$$\Rightarrow aAAS$$

$$\Rightarrow abAS$$

$$\Rightarrow abbs$$

$$\Rightarrow abbAAAS$$

$$\Rightarrow abbaAAS$$

$$\Rightarrow abbaaAS$$

$$\Rightarrow abbaabS$$

$$\Rightarrow abbaab\epsilon$$

8. Obtain a string consisting of a's and b's with atleast one a or one b.

$$A \rightarrow a | b | aS | bS$$

9. Obtain a grammar to accept the following language $L = \{w \in a^* : |w| \bmod 3 = 0\}$

$$A \rightarrow \epsilon / aaaaA$$

10. Obtain a grammar to accept the following language $L = \{w \in a^* : |w| \bmod 3 = 1\}$

$$A \rightarrow a / aa / aaaaA$$

11. Obtain a grammar to generate strings of a's and b's having a substring ab.

$$S \rightarrow AabA$$

$$A \rightarrow \epsilon / aA / bA$$

12. Obtain a grammar to generate strings of a's and b's beginning with aba

$$S \rightarrow abaA$$

$$A \rightarrow \epsilon / aA / bA$$

13. Obtain a grammar to generate strings of a's and b's ending with bab

$$S \rightarrow Abab$$

$$A \rightarrow \epsilon / aA / bA$$

14. Obtain a grammar to accept the following language $L = \{w \in \{a,b\}^* : \#_a(w) \bmod 2 = 0\}$

$$S \rightarrow BaBaBS / B$$

$$B \rightarrow \epsilon / bB$$

15. Obtain a grammar to generate the following language $L = \{a^n b^n / n \geq 0\}$

$$S \rightarrow \epsilon / aSb$$

16. Obtain a grammar to generate the following language $L = \{a^n b^n / n \geq 1\}$

$$S \rightarrow ab / aSb$$

17. Obtain a grammar to generate the following language $L = \{a^{n+1} b^n / n \geq 0\}$

$$S \rightarrow a / aSb$$

18. Obtain a grammar to generate the following language

$$L = \{a^n b^{n+2} : n \geq 0\}$$

$$S \rightarrow bb / aSb$$

19. Obtain a grammar to generate the following language

$$L = \{a^n b^{2n} : n \geq 0\}$$

$$S \rightarrow \epsilon | aSbb$$

20. Obtain a grammar to generate the following language.

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

$$S \rightarrow aab | aSb$$

21. Obtain a grammar to generate the language $L = \{0^m 1^n : m \geq 1, n \geq 0\}$

$$S \rightarrow AB$$

$$A \rightarrow 01 | 0A1$$

$$B \rightarrow \epsilon | 2B$$

22. Obtain a grammar to generate the language $L = \{W \in \{a, b\}^* : WW^R\}$

$$S \rightarrow \epsilon | aSa | bSb$$

babbab

$$S \Rightarrow bSb$$

$$\Rightarrow baSab$$

$$\Rightarrow babsbab$$

$$\Rightarrow bababab$$

23. Obtain a grammar to generate $L = \{W \in \{a, b\}^* : \text{number of } a\text{'s in } W = \text{number of } b\text{'s in } W\}$.

$$1) S \rightarrow \epsilon | aSb | bSa | SS$$

$$2) S \rightarrow \epsilon | aSb | bSa | aS | bS$$

$$3) S \rightarrow \epsilon | aSbS | bSaS$$

24. Obtain a grammar to generate $L = \{W \in \{a\}^* : |W| \bmod 3 \neq |W| \bmod 2\}$

$$L = \{2, 3, 4, 5, \cancel{6}, \cancel{7},$$

$$8, 9, 10, 11, \cancel{12}, \cancel{13},$$

$$14, 15, 16, 17, \cancel{18}, \cancel{19}\}$$

$$S \rightarrow aa | aaa | aaaa | aaaaa | aaaaaa | S$$

25. Obtain a grammar to generate $L = \{W \in \{a\}^* : |W| \bmod 3 \geq |W| \bmod 2\}$

$$L = \{0, 1, 2, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 16, 17, 18, 19, 20, 22\}$$

$$S \rightarrow \epsilon | a | aa | aaaa | aaaaa | aaaaaa | S$$

26. Obtain a grammar to generate $L = \{W \in \{a\}^* : |W| \bmod 3 \leq |W| \bmod 2\}$

$$L = \{0, 1, 3, \cancel{4}, 6, 7, 9, 12, 13, 15, 18\}$$

$$S \rightarrow \epsilon | a | aaa | aaaaaa S$$

27. Obtain a grammar to generate $L = \{W \in \{a, b\}^* : |W| \bmod 3 \leq |W| \bmod 2\}$

$$S \rightarrow \epsilon | A | AAA | AAAAAA S$$

$$A \rightarrow a | b$$

28. Obtain a grammar to generate $L = \{a^{n+2} b^m \mid n \geq 0 \text{ and } m > n\}$

$$n=0 \text{ then } m \geq 1$$

$$n=0 \quad \underline{aabb}^*$$

$$n=1 \quad \underline{aaa bbb}^*$$

$$n=2 \quad \underline{aaaa bbbb}^*$$

$$n=3 \quad \underline{aaaaa bbbbbb}^*$$

$$S \rightarrow aAB$$

$$A \rightarrow ab | aAb$$

$$B \rightarrow \epsilon | bB$$

29. $L = \{a^n b^m \mid n \geq 0 \text{ and } m > n\}$

$$n=0 \text{ then } m \geq 1$$

$$n=0, \quad \underline{bb}^*$$

$$n=1 \quad \underline{abb b}^*$$

$$n=2 \quad \underline{aabb bbb}^*$$

$$n=3 \quad \underline{aaabbb bbbb}^*$$

$$S \rightarrow AB$$

$$A \rightarrow \epsilon | aAb$$

$$B \rightarrow b | bB$$

30. $L = \{a^n b^m c^k \mid n + 2m = k, n \geq 0, m \geq n\}$

$$S \rightarrow aSc | A$$

$$A \rightarrow \epsilon | bAcc$$

$$a^n b^m c^{n+2m}$$

$$a^n b^m c^n c^{2m}$$

$$a^n b^m c^{2m} c^n$$

A

31. obtain a grammar to generate $L = \{0^i 1^j \mid i \neq j, i \geq 0, j \geq 0\}$

$i=0$ then $j \geq 1$

$i=0$ 11^*

$i=1$ 0111^*

$i=2$ 001111^*

$i=3$ 00011111^*

$j=0$ then $i \geq 1$

$i=1, j=0$ 0

$i=2, j=1$ $00\underline{1}1^*$

$i=3, j=2$ $000\underline{A}111^*$

$S \rightarrow BA \mid AC$

$A \rightarrow \epsilon \mid 0A1$

$B \rightarrow 0 \mid 0B$

$C \rightarrow 1 \mid 1C$

Derivations and Parse trees:-

There are two types of derivations

i) left most derivation (LMD) from left to right

ii) Right most derivation (RMD) from right to left

1. Obtain LMD, RMD and Parse tree the string $(a101+b1)^*(a1+b)$ from the following grammar

$E \rightarrow E * E \mid E + E \mid (E) \mid I$

$I \rightarrow a \mid b \mid I_a \mid I_b \mid I_0 \mid I_1$

$E \xRightarrow{\text{LMD}} E * E$

$\Rightarrow (E) * E$

$\Rightarrow (E + E) * E$

$\Rightarrow (I + E) * E$

$\Rightarrow (I_1 + E) * E$

$\Rightarrow (I_0 I_1 + E) * E$

$\Rightarrow (I_0 I_1 + E) * E$

$\Rightarrow (a I_0 I_1 + E) * E$

$\Rightarrow (a I_0 I_1 + I) * E$

$\Rightarrow (a I_0 I_1 + I_1) * E$

$\Rightarrow (a I_0 I_1 + b I) * E$

$\Rightarrow (a I_0 I_1 + b I)^*(E)$

$\Rightarrow (a I_0 I_1 + b I)^*(E + E)$

$\Rightarrow (a I_0 I_1 + b I)^*(I + E)$

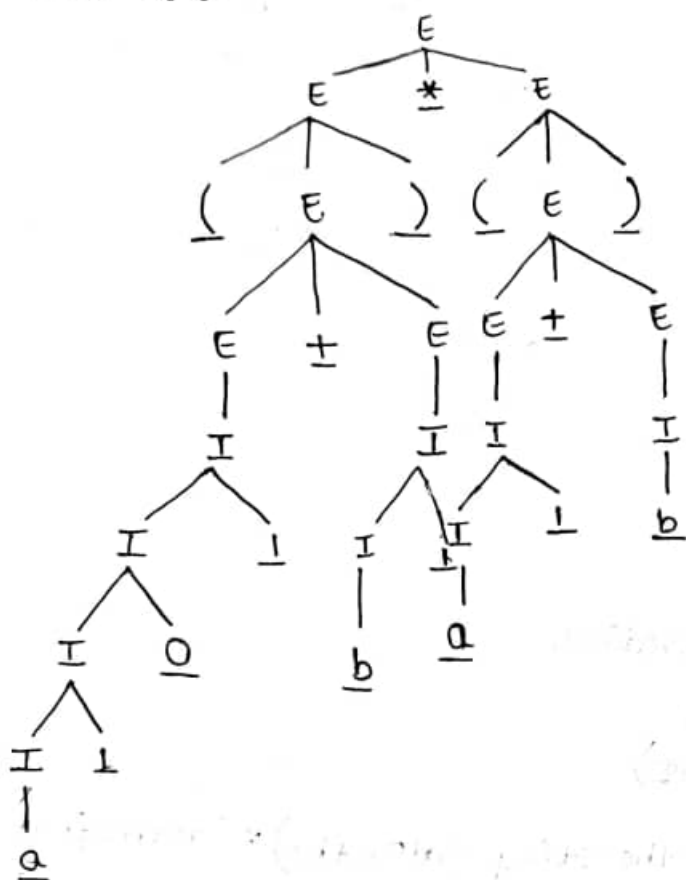
$\Rightarrow (a I_0 I_1 + b I)^*(I_1 + E)$

$\Rightarrow (a I_0 I_1 + b I)^*(a I + E)$

$\Rightarrow (a I_0 I_1 + b I)^*(a I + I)$

$\Rightarrow (a I_0 I_1 + b I)^*(a I + b)$

Parse tree:-



$$E \Rightarrow E * E$$

RMD

$$\Rightarrow E * (E)$$

$$\Rightarrow E * (E + E)$$

$$\Rightarrow E * (E + I)$$

$$\Rightarrow E * (E + b)$$

$$\Rightarrow E * (I + b)$$

$$\Rightarrow E * (II + b)$$

$$\Rightarrow E * (a + b)$$

$$\Rightarrow (E) * (a + b)$$

$$\Rightarrow (E + E) * (a + b)$$

$$\Rightarrow (E + I) * (a + b)$$

$$\Rightarrow (E + II) * (a + b)$$

$$\Rightarrow (E + bI) * (a + b)$$

$$\Rightarrow (I + bI) * (a + b)$$

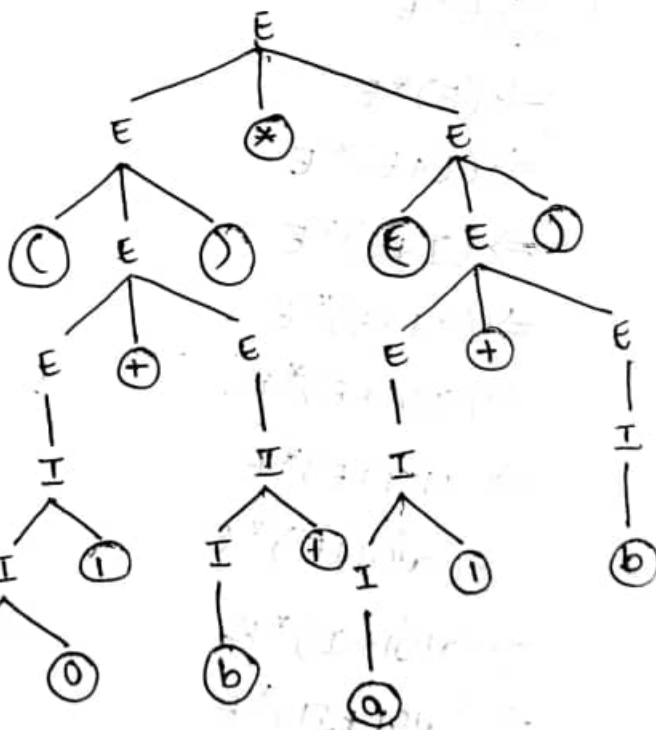
$$\Rightarrow (II + bI) * (a + b)$$

$$\Rightarrow (Ib + bI) * (a + b)$$

$$\Rightarrow (IOI + bI) * (a + b)$$

$$\Rightarrow (IIOI + bI) * (a + b)$$

$$\Rightarrow (aIOI + bI) * (a + b)$$



2. Obtain LMD, RMD, Parse tree for the string $+-*xyxy$ from the grammar

$E \rightarrow +EE \mid -EE \mid *EE \mid I$

$I \rightarrow x \mid y$

$E \xRightarrow{\text{LMD}} +EE$

$\Rightarrow + - EEE$

$\Rightarrow + - * EEEE$

$\Rightarrow + - * I EEE$

$\Rightarrow + - * x EEE$

$\Rightarrow + - * x I EE$

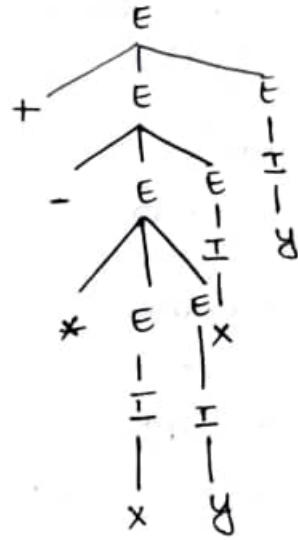
$\Rightarrow + - * x y EE$

$\Rightarrow + - * x y IE$

$\Rightarrow + - * x y x E$

$\Rightarrow + - * x y x I$

$\Rightarrow + - * x y x y$



$E \xRightarrow{\text{RMD}} +EE$

$\Rightarrow +EI$

$\Rightarrow +Ey$

$\Rightarrow + - EEy$

$\Rightarrow + - EIy$

$\Rightarrow + - Exy$

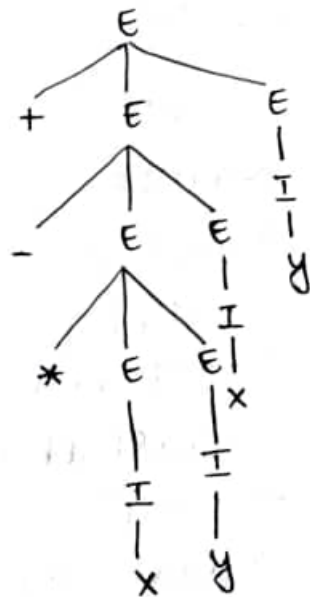
$\Rightarrow + - * EExy$

$\Rightarrow + - * EIxy$

$\Rightarrow + - * Eyxy$

$\Rightarrow + - * Iyxy$

$\Rightarrow + - * xyxy$



3. Consider the grammar

$$S \rightarrow aB/bA$$

$$A \rightarrow aS/bAA/a$$

$$B \rightarrow bS/aBB/b$$

Obtain LMD, RMD, Parse trees for the string aaabbabbba

$$S \Rightarrow aB$$

LMD

$$\Rightarrow aaBB$$

$$\Rightarrow aaaBBB$$

$$\Rightarrow aaabSBB$$

$$\Rightarrow aaabbABB$$

$$\Rightarrow aaabbbaBB$$

$$\Rightarrow aaabbabB$$

$$\Rightarrow aaabbabbs$$

$$\Rightarrow aaabbabbbaA$$

$$\Rightarrow aaabbabbba$$

$$S \Rightarrow aB$$

RMD

$$\Rightarrow aaBB$$

$$\Rightarrow aaBbs$$

$$\Rightarrow aaBbbA$$

$$\Rightarrow aaBbba$$

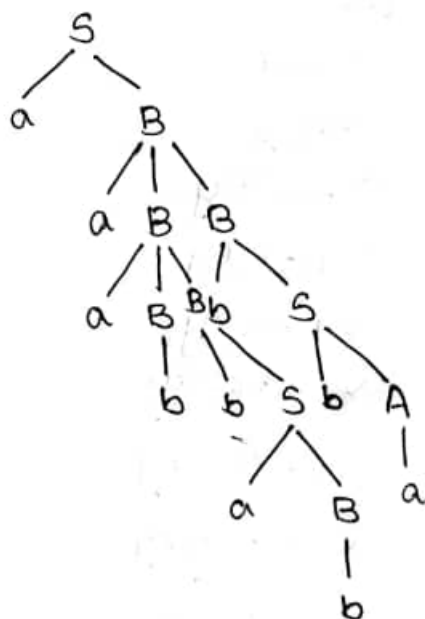
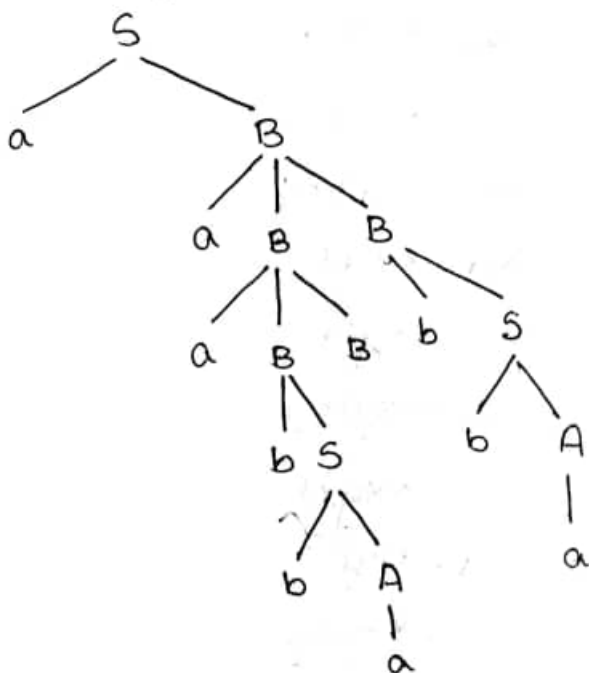
$$\Rightarrow aaabBBbba$$

$$\Rightarrow aaabBsbbba$$

$$\Rightarrow aaabBbaBbba$$

$$\Rightarrow aaabBbabbbba$$

$$\Rightarrow aaabbabbba$$



Pumping Lemma for Context free Language:-

Theorem:- Let L be a CFL, then there exists a constant n such that if z in any string in L such that $|z|$ is atleast n then we can write $z = uvwxy$ subject to the following condition.

1) $|vwx| \leq n$ (The middle portion is not too long)

2) $vx \neq \epsilon$ i.e., $|vx| \geq 1$ (Atleast one of the strings we pump must not be empty)

3) for all $i \geq 0$, uv^iwx^iy is in L (that is two strings v and x may be pumped any number of times).

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

Closure properties of CFL's:-

1) CFLs are closed under union, concatenation and star

i) Union of CFLs is CFL:-

Let L_1 and L_2 are two CFLs generated by CFGs.

$$G_1 = (V_1, T_1, P_1, S_1)$$

$$G_2 = (V_2, T_2, P_2, S_2)$$

Let us consider the language L_3 generated by the grammar

$$G_3 = (V_1 \cup V_2 \cup S_3, T_1 \cup T_2, P_3, S_3)$$

where S_3 is the start symbol from G_3 .

$$S_3 \notin (V_1 \cup V_2)$$

$$P_3 = P_1 \cup P_2 \cup \{S_3 \rightarrow S_1 \mid S_2\}$$

$$L_3 = L_1 \cup L_2$$

G_3 is context free and the language generated by this grammar is context free. If we assume $W \in L$, then the possible derivation from S_3 is

$$S_3 \Rightarrow S_1$$

$$\Rightarrow W$$

If we assume $W \in L_2$

$$S_3 \Rightarrow S_2$$

$$\Rightarrow W$$

So if $W \in L_3$ one of the derivation

$$S_3 \Rightarrow S_1 \Rightarrow W$$

or $S_3 \Rightarrow S_2 \Rightarrow W$ is possible

Thus it is proved that context free languages are closed under union.

ii) Concatenation of two CFLs is CFL:-

Let us consider the language L_1 generated by the grammar

$$G_1 = (V_1, U_1, S_1, P_1)$$

$$S_1 \in V_1$$

$$P_1 = \{S_1 \rightarrow S_1 S_1\}$$

It is clear from this that the grammar G_1 is context free and language generated by this grammar is context free and so

$$L_3 = L_1 L_2$$

Thus, it is proved that CFL are closed under concatenation.

iii) CFLs are closed under star closure:-

Now, let us consider the language L_5 generated by grammar

$$G_5 = (V_5, U_5, S_5, P_5)$$

S_5 is start symbol of G_5

$$P_5 = \{S_5 \rightarrow S_5 S_5^* | \epsilon\}$$

$$L_5 = L_5^*$$

It is proved that CFL are closed under star-closure

iv) CFLs are not closed under intersection:-

CFLs are not closed under intersection. If L_1 and L_2 are CFL, it is not always true that $L_1 \cap L_2$ is CFL.

Consider two languages

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

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

The two languages are context free, as we can easily obtain the corresponding CFG.

$$S \rightarrow S_1 S_2$$

$$S_1 \rightarrow a S_1 b \mid \epsilon$$

$$S_2 \rightarrow c S_2 \mid \epsilon$$

$$\text{and } S \rightarrow a S \mid \epsilon$$

$$S_1 \rightarrow b S_1 c \mid \epsilon$$

$$L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 0\}$$

We have already proved earlier that this language is not CFL.
So, it is not closed under intersection.

1. Show that $L = \{a^n b^n c^n \mid n \geq 0\}$ is context free language

$$n=3$$

$$W = \underline{aaa} \underline{bbb} \underline{ccc}$$

u
 v
 w
 x
 y

$$|W| \geq n$$

$$9 \geq 3$$

$$i) |vwx| \leq n$$

$$3 = 3$$

$$ii) vx \neq \epsilon$$

$$bb = vx$$

$$iii) uv^i wx^i y \in L$$

$$i=0 \quad uv^0 wx^0 y = aaabccc \notin L$$

$$i=1 \quad uv^1 wx^1 y = aaabbbccc \in L$$

$$i \geq 2 \quad uv^2 wx^2 y = aaabbbbbccc \notin L$$

For $i=0$ and $i \geq 2$ the assumption is false

2. Show that $L = \{a^p b^q \mid p = q^2\}$ is not context free language

$$\text{Let } p=9, q=3$$

$$W = \underline{aaaaaaaa} \underline{abbb}$$

u
 v
 w
 x
 y

$$i) |vwx| \leq n$$

$$3 = 3$$

$$ii) vx \neq \epsilon$$

$$vx = aa$$

$$iii) \forall i \geq 0 \quad uv^i wx^i y \in L$$

$$i=0 \quad uv^0 wx^0 y = aaaaaa \notin L$$

$$i=1 \quad uv^1 wx^1 y = aaaaaaaaaabbb \in L$$

$$i \geq 2 \quad uv^2 wx^2 y = aaaaaaaaaaaaaabbb \notin L$$

For $i=0$ and $i \geq 2$ the assumption is false.

Ambiguity of a Grammar:-

A CFG $\{G = \{V, T, P, S\}$ is ambiguous if there is atleast one string W for which we can find two different parse trees each with root label S produce W .

→ If each string has atmost one parse tree in the grammar, then the grammar is unambiguous.

1. Consider the grammar.

$E \rightarrow E + E \mid E * E \mid I$

$I \rightarrow a$

Check whether it is ambiguous

$E \Rightarrow E + E$

LMD

$\Rightarrow E * E + E$

$\Rightarrow I * E + E$

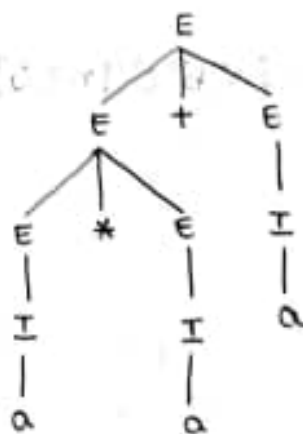
$\Rightarrow a * E + E$

$\Rightarrow a * I + E$

$\Rightarrow a * a + E$

$\Rightarrow a * a + I$

$\Rightarrow a * a + a$



$E \Rightarrow E * E$

LMD

$\Rightarrow I * E$

$\Rightarrow a * E$

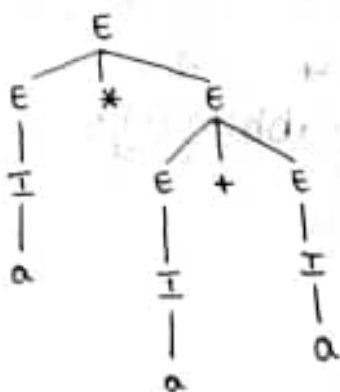
$\Rightarrow a * E + E$

$\Rightarrow a * I + E$

$\Rightarrow a * a + E$

$\Rightarrow a * a + I$

$\Rightarrow a * a + a$



For the same string $a*a+a$, we get two different parse trees, so the grammar is ambiguous.

2. Check whether the given grammar is ambiguous or not.

$$S \rightarrow aS \mid x$$

$$x \rightarrow ax \mid a$$

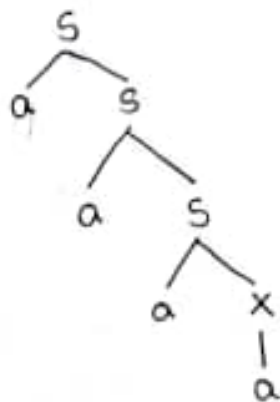
$$S \xRightarrow{\text{LMD}} aS$$

$$\Rightarrow aaS$$

$$\Rightarrow aaX$$

$$\Rightarrow aaaS$$

$$\Rightarrow aaaa$$



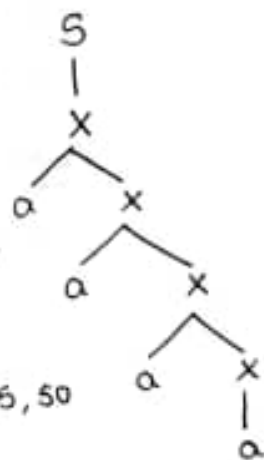
$$S \xRightarrow{\text{LMD}} x$$

$$\Rightarrow ax$$

$$\Rightarrow aaX$$

$$\Rightarrow aaaS$$

$$\Rightarrow aaaa$$



For the same string, we get two different parse trees, so the grammar is ambiguous.

3. Check whether the given grammar is ambiguous or not

$$S \rightarrow aSbS \mid bSaS \mid \epsilon$$

$$S \xRightarrow{\text{LMD}} aSbS$$

$$\Rightarrow abSaSbS$$

$$\Rightarrow ab\epsilon aSbS$$

$$\Rightarrow ab\epsilon a\epsilon bS$$

$$\Rightarrow ab\epsilon a\epsilon b\epsilon$$

$$\Rightarrow abab$$

$$S \xRightarrow{\text{LMD}} aSbS$$

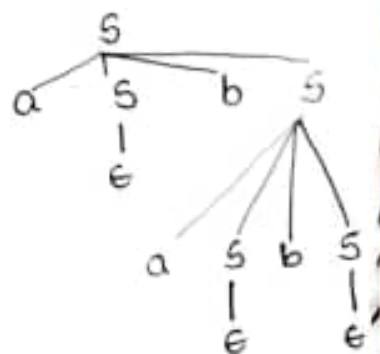
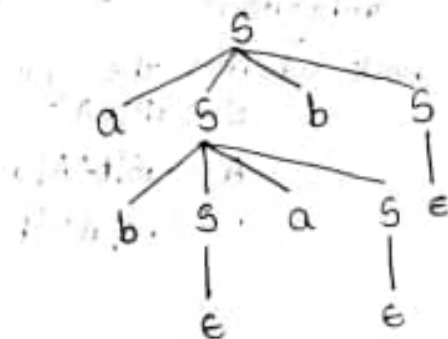
$$\Rightarrow a\epsilon bS$$

$$\Rightarrow a\epsilon baSbS$$

$$\Rightarrow a\epsilon ba\epsilon bS$$

$$\Rightarrow a\epsilon ba\epsilon b\epsilon$$

$$\Rightarrow abab$$



For the same string, we get two different parse trees, so the grammar is ambiguous.

4. Check whether the given grammar is ambiguous or not

$$S \rightarrow iCtS \mid ictSeS \mid a$$

$$C \rightarrow b$$

$$S \Rightarrow iCtS$$

LDM

$$\Rightarrow ibtS$$

$$\Rightarrow ibtiCtSeS$$

$$\Rightarrow ibtibtSeS$$

$$\Rightarrow ibtibtaeS$$

$$\Rightarrow ibtibtaea$$

$$S \Rightarrow ictSeS$$

LDM

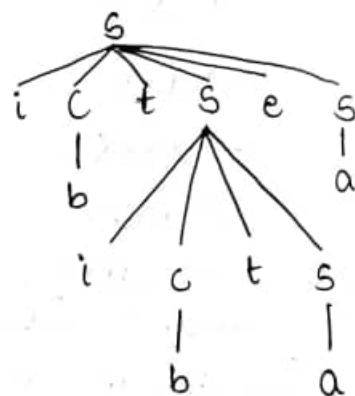
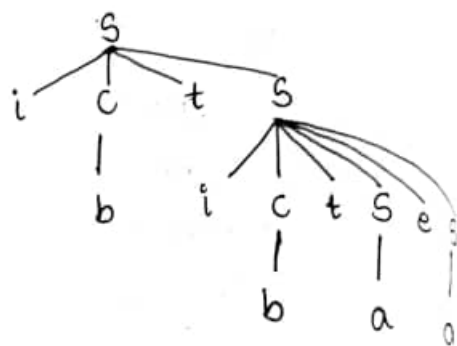
$$\Rightarrow ibtSeS$$

$$\Rightarrow ibtiCtSeS$$

$$\Rightarrow ibtibtSeS$$

$$\Rightarrow ibtibtaeS$$

$$\Rightarrow ibtibtaea$$



For the same thing, we get two different parse trees, so the grammar is ambiguous.

5. Check whether the given grammar is ambiguous or not

$$S \rightarrow aB \mid bA$$

$$A \rightarrow aS \mid bAA \mid a$$

$$B \rightarrow bS \mid aBB \mid b$$

$$S \Rightarrow aB$$

LMD

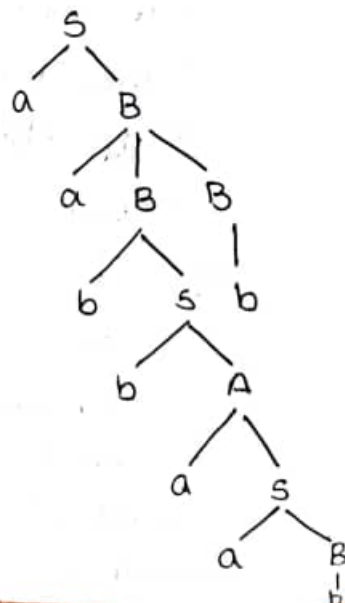
$$\Rightarrow aaBB$$

$$\Rightarrow aabBB$$

$$\Rightarrow aabbAB$$

$$\Rightarrow aabbbaaBB$$

$$\Rightarrow aabbbaabbb$$



$S \Rightarrow aB$
 LMD

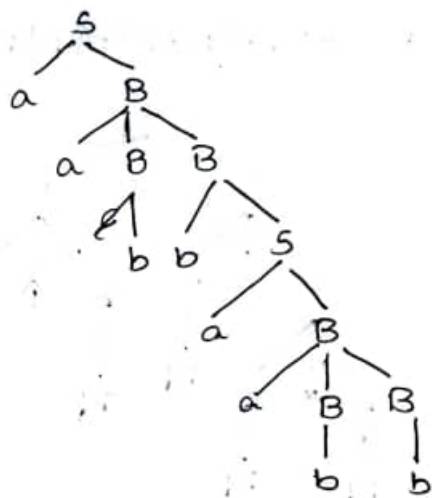
$\Rightarrow aaBB$

$\Rightarrow aabbS$

$\Rightarrow aabbaB$

$\Rightarrow aabbbaBB$

$\Rightarrow aabbaabb$



Elimination of Ambiguity

There are mainly two reasons to get ambiguous grammar

1) Associativity

2) Precedence

Associativity can be removed by using recursion.

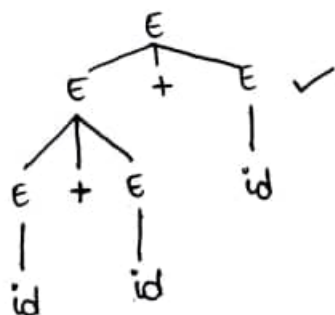
$+, -, *, / \Rightarrow$ Left associate

$\uparrow \Rightarrow$ Right associate

Precedence can be removed by levels i.e., highest precedence operation should be in bottom level.

Consider the grammar

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

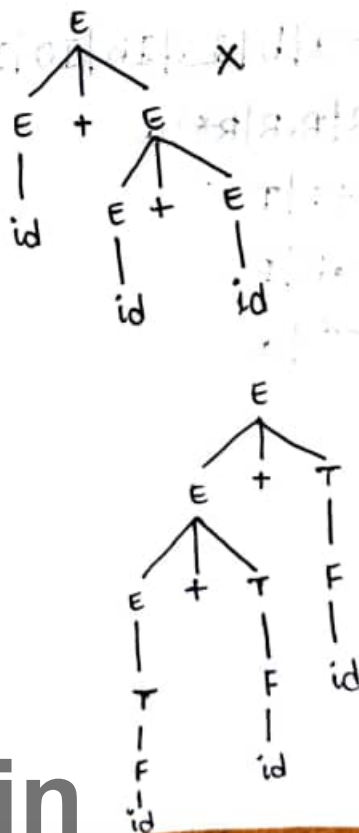


Unambiguous grammar

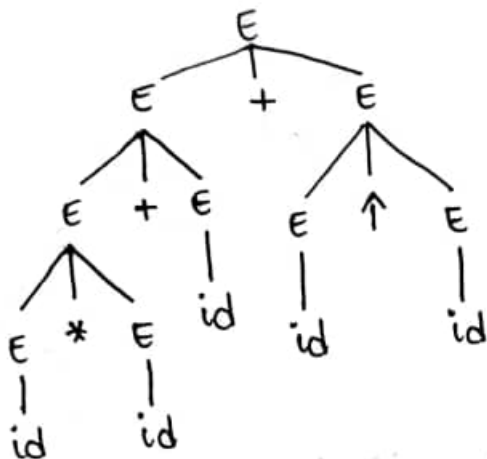
$E \rightarrow E + T \mid T$

$T \rightarrow T * F \mid F$

$F \rightarrow id$



$$ii) E \rightarrow E + E \mid E * E \mid E \uparrow E \mid id$$



$id * id + id + id \uparrow id$

$$E \rightarrow E + T \mid T$$

$$T \rightarrow T * F \mid F$$

$$F \rightarrow G * F \mid G$$

$$G \rightarrow id$$

$$iii) E \rightarrow E * E \mid E + E \mid (E) \mid I$$

$$I \rightarrow a \mid b \mid Ia \mid Ib \mid Io \mid Ii$$

$$E \rightarrow E + T \mid T$$

$$T \rightarrow T * F \mid F$$

$$F \rightarrow (E) \mid I$$

$$I \rightarrow a \mid b \mid Ia \mid Ib \mid Io \mid Ii$$

$$iv) R \rightarrow R + R \mid R.R \mid R * \mid a$$

$$R \rightarrow R + T \mid T$$

$$T \rightarrow T.F \mid F$$

$$F \rightarrow F * \mid a$$

PDA [Push Down Automata] :-

Definition:-

PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ where Q, Σ, q_0, F are same as DFA (or) NFA.

Z_0 is the stack start symbol

Γ (big gamma) is a finite stack alphabet

δ is a transition function that takes three arguments a input and returns two arguments.

$$\delta(q, a, x) = (p, \gamma)$$

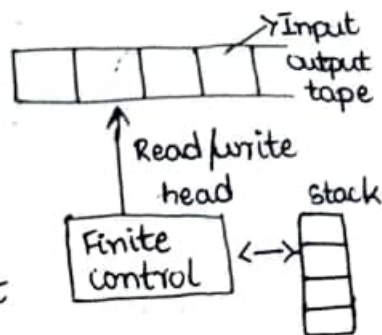
↓
current state

↓
Input symbol

↓
stack symbol

↓
next state

↓
string of stack symbol that replaces x at the top of the stack.



- Consider the language $L = \{a^n b^n \mid n \geq 1\}$. Obtain a PDA to accept the language by final state and show the Instantaneous Description (ID) moves for the string aaabbbb from the same PDA as above and draw the transition diagram for the same.

$$1) \delta(q_0, a, Z_0) = (q_0, aZ_0) \quad \left. \begin{array}{l} \\ \\ \end{array} \right\} \text{Push operation}$$

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

$$3) \delta(q_0, b, a) = (q_1, \epsilon) \quad \left. \begin{array}{l} \\ \\ \end{array} \right\} \text{Pop operation}$$

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

$$5) \delta(q_1, \epsilon, Z_0) = (q_f, Z_0)$$

ID moves:-

$$(q_0, aaabbbb, Z_0) \vdash (q_0, aabbbb, aZ_0)$$

$$\vdash (q_0, abbbb, aaZ_0)$$

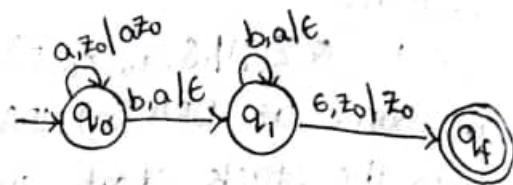
$$\vdash (q_0, bbbb, aaaZ_0)$$

$$\vdash (q_1, bbb, aaZ_0)$$

$$\vdash (q_1, bb, aZ_0)$$

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

$\vdash (q_f, \epsilon, z_0)$



ii) aabbb

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

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

ID moves:-

$(q_0, aabbb, z_0) \vdash (q_0, abbb, a z_0)$

$\vdash (q_0, bbb, aa z_0)$

$\vdash (q_1, bb, a z_0)$

$\vdash (q_1, b, z_0)$

String is not reaching to final state. So it is not accepted.

Languages of PDA:-

1) Acceptance by final state:-

Let $P = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ be a PDA then $L(P) = \{w \mid (q_0, w, z_0) \vdash^* (q, \epsilon, \alpha)\}$

For some state q in F and any stack string α

2) Acceptance by empty stack:-

Let $P = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ be a PDA then $N(P) = \{w \mid (q_0, w, z_0) \vdash^* (q, \epsilon, \epsilon)\}$

For any state q and stack content should be ϵ and the input string also ϵ .

2. Obtain a PDA for the string $L = \{WCW^R \mid W \in \{a,b\}^*\}$ by final state.

abbCbba
baaCaab
babCbab
abaCaba

$$\left. \begin{aligned} \delta(q_0, a, z_0) &= (q_0, az_0) \\ \delta(q_0, b, z_0) &= (q_0, bz_0) \end{aligned} \right\} \text{Initialize}$$

$$\left. \begin{aligned} \delta(q_0, b, a) &= (q_0, ba) \\ \delta(q_0, a, b) &= (q_0, ab) \\ \delta(q_0, a, a) &= (q_0, aa) \\ \delta(q_0, b, b) &= (q_0, bb) \end{aligned} \right\} \text{Push}$$

Deterministic PDA

$$\left. \begin{aligned} \delta(q_0, C, b) &= (q_1, b) \\ \delta(q_0, C, a) &= (q_1, a) \\ \delta(q_0, C, z_0) &= (q_f, z_0) \end{aligned} \right\} \text{Shift}$$

$$\left. \begin{aligned} \delta(q_1, b, b) &= (q_1, \epsilon) \\ \delta(q_1, a, a) &= (q_1, \epsilon) \end{aligned} \right\} \text{Pop}$$

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

IO moves :-

$$(q_0, abbCbba, z_0) \vdash (q_0, bbCbba, az_0)$$

$$\vdash (q_0, bCbba, ba z_0)$$

$$\vdash (q_0, Cbba, bba z_0)$$

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

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

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

$$\vdash (q_f, \epsilon, z_0)$$

(q_0)

(q_1)

(q_f)

abCbb

$$(q_0, abCbb, z_0) \vdash (q_0, bCbb, az_0)$$

$$\vdash (q_0, Cbb, baz_0)$$

$$\vdash (q_0, bb, baz_0)$$

$$\vdash (q_1, b, az_0)$$

The stack is not empty. So the string is not accepted.

3. Obtain a PDA for a language $L = \{w \in \{a, b\}^* : N_a(w) = N_b(w)\}$ by final state and draw the transition diagram and show the ID moves for string ababab, babab.

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

Non-deterministic PDA

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

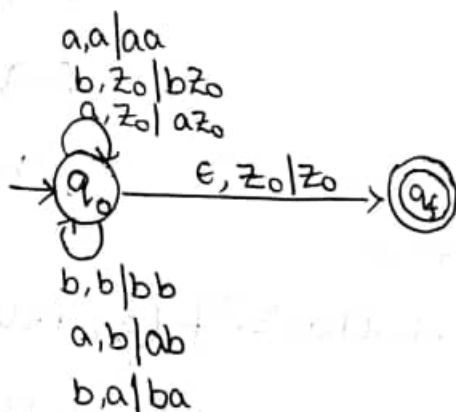
$$\delta(q_0, a, a) = (q_0, aa)$$

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

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

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

$$\delta(q_0, \epsilon, z_0) = (q_f, z_0)$$



ababab

$$(q_0, ababab, z_0) \vdash (q_0, bababa, az_0)$$

$$\vdash (q_0, abab, z_0)$$

$$\vdash (q_0, bab, az_0)$$

$$\vdash (q_0, ab, z_0)$$

$$\vdash (q_0, b, az_0)$$

$$\vdash (q_f, \epsilon, z_0)$$

The stack is empty. Therefore, it is accepted.

babaa

$$(q_0, babaa, z_0) \vdash (q_0, abaa, bz_0)$$

$$\vdash (q_0, baa, z_0)$$

$$\vdash (q_0, aa, bz_0)$$

$$\vdash (q_0, a, z_0)$$

The stack is not empty. Therefore, it is not accepted.

4. Obtain a PDA to accept the language $L = \{a^n b^{2n} \mid n \geq 1\}$. Draw the transition diagram for the same and show the ID moves aabbbb and aabbb.

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

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

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

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

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

ID moves:-

aabbbb

$$(q_0, aabbbb, z_0) \vdash (q_0, abbbb, aa, z_0)$$

$$\vdash (q_0, bbbb, aaaa, z_0)$$

$$\vdash (q_1, bbb, aaaa, z_0)$$

$$\vdash (q_1, bb, aaaa, z_0)$$

$$\vdash (q_1, b, aaaa, z_0)$$

$$\vdash (q_f, \epsilon, z_0)$$

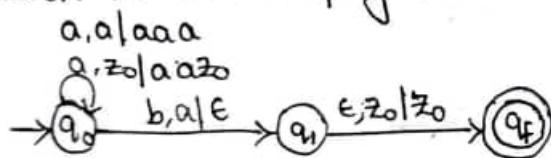
$$(q_0, aabbb, z_0) \vdash (q_0, abbb, az_0)$$

$$\vdash (q_0, bbb, aa z_0)$$

$$\vdash (q_0, bb, a z_0)$$

$$\vdash (q_0, b, z_0)$$

The stack is not empty. So the string is not accepted



5. Obtain a PDA to accept the language $L = \{WW^R \mid W \in \{a,b\}^*\}$ by final state. Draw the transition diagram for the same and show the ID moves for the string aabbbaa, ababa

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

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

$$\delta(q_0, \epsilon, z_0) = (q_0, z_0)$$

Initialization

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

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

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

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

Push

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

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

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

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

Pop

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

ID moves:-

aabbaa

$$(q_0, aabbaa, z_0) \vdash (q_0, abbaa, az_0)$$

$$\vdash (q_0, bbaa, aa z_0)$$

$$\vdash (q_0, baa, baaz_0)$$

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

$$\vdash (q_1, a, az_0)$$

$$\vdash (q_f, \epsilon, z_0)$$

The stack is empty. So the string is accepted.

ababa

$$(q_0, ababa, z_0) \vdash (q_0, ababa, az_0)$$

$$\vdash (q_0, aba, ba z_0)$$

$$\vdash (q_0, ba, abaz_0)$$

$$\vdash (q_0, a, babaz_0)$$

The stack is not empty. So, the string is not accepted.

6. Obtain a PDA to accept a string of balanced parenthesis. The parenthesis to be considered are (,), [,].

$$\delta(q_0, (, z_0) = (q_0, (z_0) \quad \left. \begin{array}{l} \delta(q_0, [, z_0) = (q_0, [z_0) \end{array} \right\} \text{Initialization}$$

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

$$\delta(q_0, [, [) = (q_0, [[)$$

$$\delta(q_0, (, [) = (q_0, ([)$$

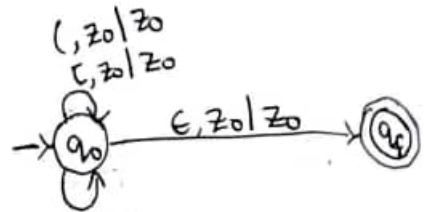
$$\delta(q_0, [, () = (q_0, [[()$$

$$\delta(q_0,), () = (q_0, \epsilon) \quad \left. \begin{array}{l} \delta(q_0,], [) = (q_0, \epsilon) \end{array} \right\} \text{Pop}$$

$$\delta(q_0, \epsilon, z_0) = (q_f, z_0)$$

ID moves - $([])()$

$(q_0, ([])(), z_0) \vdash (q_0, [])(), (z_0)$
 $\vdash (q_0,])(), ([z_0)$
 $\vdash (q_0,)(), (z_0)$
 $\vdash (q_0, (), z_0)$
 $\vdash (q_0,), (z_0)$
 $\vdash (q_f, \epsilon, z_0)$



7. Obtain a PDA for the language $L = \{a^i b^j c^k \mid i+j=k, i \& j \geq 0\}$

$\delta(q_0, a, z_0) = (q_0, a z_0)$
 $\delta(q_0, \epsilon, z_0) = (q_f, z_0)$

} Initialize

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

} push

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

} Pop

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

ID moves :- aabbccccc

$(q_0, aabbccccc, z_0) \vdash (q_0, abbccccc, a z_0)$
 $\vdash (q_0, bbccccc, aa z_0)$
 $\vdash (q_0, bccccc, baa z_0)$
 $\vdash (q_0, cccc, bb aa z_0)$
 $\vdash (q_1, ccc, baa z_0)$
 $\vdash (q_1, cc, aa z_0)$
 $\vdash (q_1, c, a z_0)$
 $\vdash (q_f, \epsilon, z_0)$

Conversion from grammar to PDA:- [CFG to PDA]

For each variable A

$$\delta(q, \epsilon, A) = \{(q, \beta) \mid A \rightarrow \beta \text{ is a production of } P\}$$

Terminal ' a '

$$\delta(q, a, a) = \{(q, \epsilon)\}$$

1. Convert the given CFG into PDA

$$i) E \rightarrow E+E \mid E * E \mid (E) \mid I$$

$$I \rightarrow a \mid b \mid Ia \mid Ib \mid Io \mid Ii$$

$$\delta(q, \epsilon, E) = \{(q, E+E), (q, E * E), (q, (E)), (q, I)\}$$

$$\delta(q, \epsilon, I) = \{(q, a), (q, b), (q, Ia), (q, Ib), (q, Io), (q, Ii)\}$$

$$\delta(q, +, +) = (q, \epsilon)$$

$$\delta(q, *, *) = (q, \epsilon)$$

$$\delta(q, (, () = (q, \epsilon)$$

$$\delta(q,),)) = (q, \epsilon)$$

$$\delta(q, a, a) = (q, \epsilon)$$

$$\delta(q, b, b) = (q, \epsilon)$$

$$\delta(q, o, o) = (q, \epsilon)$$

$$\delta(q, i, i) = (q, \epsilon)$$

$$ii) S \rightarrow aABC$$

$$A \rightarrow aB \mid a$$

$$B \rightarrow bA \mid b$$

$$C \rightarrow a$$

$$\delta(q, \epsilon, S) = \{(q, aABC)\}$$

$$\delta(q, \epsilon, A) = \{(q, aB), (q, a)\}$$

$$\delta(q, \epsilon, B) = \{(q, bA), (q, b)\}$$

$$\delta(q, \epsilon, c) = (q, a)$$

$$\delta(q, a, a) = (q, \epsilon)$$

$$\delta(q, b, b) = (q, \epsilon)$$

$$\text{iii)} \quad S \longrightarrow aABB \mid aAA$$

$$A \longrightarrow aBB \mid a$$

$$B \longrightarrow bBB \mid A$$

$$C \longrightarrow a$$

$$\delta(q, \epsilon, S) = \{(q, aABB), (q, aAA)\}$$

$$\delta(q, \epsilon, A) = \{(q, aBB), (q, a)\}$$

$$\delta(q, \epsilon, B) = \{(q, bBB), (q, A)\}$$

$$\delta(q, \epsilon, C) = (q, a)$$

$$\delta(q, a, a) = (q, \epsilon)$$

$$\delta(q, b, b) = (q, \epsilon)$$