Chapter-3 (Context free grammar)

A Context free grammar defined as follows

A grammar $G = (V_n, V_t, P, S)$ is said to be context free.

Where V_n = A finite set of non-terminal, generally represented by capital letters, A B C . . . Z.

V_t = A finite set of terminals, generally represented by small letters, like a, b, c . . . z.

S = Starting non terminal, called start symbol of the grammar. S belongs to V_n.

P = set of Rules or productions in CFG.

G is Context free and all productions in P have the form

$$\alpha \longrightarrow \beta$$

$$\alpha \in V_n \text{ and } \beta \in (V_t \cup V_n)^*$$

↓ Union

following observation are consider regarding the production rules.

① Reverse substition is not permitted:

If $S \to AB$ is a production then we can replace S by AB.

But we cannot replace AB by S.

② No inversion operation is permitted:

If $S \to AB$ is a production, it is not necessary that $AB \to S$ is a production.

Every regular grammar is context-free. so a regular language is also a context-free one.

Derivation & the language generate by a Grammar.

If $\alpha \to \beta$ is a production in Grammar 'G', then $\alpha \Rightarrow \beta$ is called one step derivation as G production

If $\alpha \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \alpha_3 \cdots \Rightarrow \alpha_n = \beta$

then $\alpha \overset{n}{\underset{G}{\Rightarrow}} \beta$ is called n step ~~deriv~~ derivation. The notation $\overset{}{\underset{G}{\Rightarrow}}$ represents relation for Grammar G.

eg → Following is a CFG for the language.

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

Solution → Let G be CFG for language $L = \{wcw^R / w \in (a,b)^*\}$

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

$$V_n = \{S\}$$

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

and P is given by

$S \to asa$
$S \to bsb$
$S \to c$

Let us check the abbacbba can be derived from the given CFG.

~~$S \Rightarrow asa$~~
~~$\downarrow s \to asa$~~

$S \to asa$ use $s \to asa$
$S \to abSba$ use $s \to bsb$
$S \to abbSbba$ use $s \to bsb$
$S \to abbcbba$ use $s \to c$

So string abbcbba can be derived from given CFG.

Bacos Naur Form (BNF)

A notation technique used in computer science to precisely describe the syntax of language.

$$S \rightarrow bA/aB$$
$$A \rightarrow bAA/aS/a$$
$$B \rightarrow aBB/bS/a$$

Hence BNF is a shorthand notation for context free grammars.

Q) Write a CFG, which generates palindrome for binary no.

Sol. Grammar will generate palindrome for binary numbers, that is 00, 010, 11, 101, 11100111 ...

Let CFG be $G_r = (V_n, V_t, P, S)$

V_n = set of non-terminal = {S}

V_t = set of teminal = {0,1}

and production rule P is defined as

$$S \rightarrow 0S0/1S1$$
$$S \rightarrow 0/1/\varepsilon$$

Obviously this grammar generates palindrome for binary no. if can be seen.

$$S \rightarrow 0S0$$
$$S \rightarrow 01S10$$
$$S \rightarrow 0105010$$
$$S \rightarrow 0101010$$

which is a palindrome.

Q) Write a CFG which generates strings having equal number of a's and b's. (4)

Sol → Let CFG be $G = (V_n, V_t, P, S)$.

$V_n = \{S\}$

$V_t = \{a, b\}$

where P is defined as $S \rightarrow aSbS/bSaS/\varepsilon$

Let us derive a string $w = bbabaabbaa$

$S \rightarrow bSaS$

$\downarrow S \rightarrow bSaS$

$S \rightarrow bbSaSaS$

$\rightarrow bbaSbSaSaS$

$\rightarrow bbaSbaSbSaSaS$

$\rightarrow bbaSbaaSbSbSaSaS$

$\rightarrow bbabaaSbSbSaSaS$

$\rightarrow bbabaabaSbSaSaS$

$\rightarrow b\,babaabbSaSaS$

$\rightarrow bbabaabbaSaS$

$\rightarrow bbabaabbaaS$

$\rightarrow bbabaabbaa$

Q) Design a CFG for the regular expression $r = (a+b)^*$.

Let G be CFG

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

$V_n = \{S\}$

$V_t = \{a, b\}$

P are desined as

$S \rightarrow aS$

$S \rightarrow bS$

$S \rightarrow a$

$S \rightarrow b$

$S \rightarrow \varepsilon$

The word ab can be generated by the derivation

$S \rightarrow aS$
$S \rightarrow abS$
$S \rightarrow ab\epsilon$
$S \rightarrow ab$

as by the derivation $S \rightarrow aS$
$S \rightarrow ab.$

Q) Write a CFG for the language $L(G) = \{ww^R : w \in \{0,1\}^*\}$.

Let grammar be G.

$G = \{V_n, V_t, P, S\}$
$V_n = \{S\}$
$V_t = \{0,1\}$

Productions are defined as

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

let us derive a string $\underbrace{001}_{w}\underbrace{100}_{w^R}$ by above CFG.

$S \rightarrow 0S0$
$\rightarrow 00S00$
$\rightarrow 001S100$
$\rightarrow 001100$

Q) Design a CFG for the language

$L(G) = \{ab\,(bbaa)^n\,bba\,(ba)^n : n \geq 0\}$.

Sol Let CFG be G

$G = (V_n, V_t, P, S)$
$V_n = (S, X, Y)$
$V_t = \{a, b\}$

Productions are defined as $S \rightarrow abX$
$X \rightarrow bbYa$
$Y \rightarrow caXb$
$Y \rightarrow \epsilon$

or we can also defined production as

$S \rightarrow abX$
$X \rightarrow bbaaXba$
$X \rightarrow bba$

Let us derive a string $ab(bbaa)^2bba(ba)^2$

$S \rightarrow abX$
$S \rightarrow abbbYa$
$S \rightarrow abbbaaXba$
$S \rightarrow abbbaabbYaba$
$S \rightarrow abbbaaabbaaXbaba$
$S \rightarrow ab(bbaa)^2bbYa(ba)^2$
$S \rightarrow ab(bbaa)^2bb\epsilon a(ba)^2$
$S \rightarrow ab(bbaa)^2bba(ba)^2$

$S \rightarrow abX$
$\Rightarrow abbbaaXba$
$\Rightarrow abbbaabbaaXbaba$
$\Rightarrow ab(bbaa)^2bba(ba)^2$

Q) Design a CFG for the language

$$L = \{(0^n 1^n / n \geq 0) \cup (1^n 0^n / n \geq 0)\}$$

Sol We can assure L as

$$L = L_1 \cup L_2$$

$$L_1 = \{0^n 1^n / n \geq 0\}$$

Let say G_1 be CFG for the language L_1

$$G_1 = (V_n, V_t, P, S)$$

$$V_n = \{S_1\}$$

$$S = \{S_1\}$$

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

and P is defined as $S_1 \Longrightarrow 0S_1 1 / \varepsilon$

and $L_2 = \{\text{e } 1^n 0^1 / n \geq 0\}$

Let say G_2 be CFG for language L_2

$$G_2 = (V_n, V_t, P, S)$$

$$V_n = \{S_2\}$$

$$S = \{S_2\}$$

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

P is defined as $S_2 \rightarrow 1S_2 0 / \varepsilon$

Since $L = L_1 \cup L_2$, suppose G be CFG for language L, with starting non-terminal S.

Then productions in G will be

$$S \rightarrow S_1 / S_2$$

$$S_1 \rightarrow 0S_1 1 / \varepsilon$$

$$S_2 \rightarrow 1S_2 0 / \varepsilon.$$

Q) Write the CFG for the language

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

Sol Let us assume $L = L_1 \cup L_2$

where and

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

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

Let us consider L_1 first, let CFG for L_1 be G_1

$$G_1 = (V_n, V_t, P_1, S_1)$$

$$V_n = \{S_1, A, B\}$$

$$V_t = \{0, 1, 2\}$$

P_1 is defined as follows

$$S_1 \longrightarrow AB$$

$$A \longrightarrow 0A1 \mid \varepsilon$$

$$B \longrightarrow 2B \mid \varepsilon$$

Now let us assume that CFG for Language L_2 is G_2

$$G_2 = (V_n, V_t, P_2, S_2)$$

$$V_n = (S_2, C, D)$$

$$V_t = (0, 1, 2)$$

Production are defined as follows

$$S_2 \rightarrow CD$$

$$C \rightarrow 0C \mid \varepsilon$$

$$D \rightarrow 1D2 \mid \varepsilon$$

By the help G_1 and G_2 we can define CFG for the language L. Let it is G.

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

$$V_n = \{S, S_1, S_2, A, B, C, D\}$$

$$V_t = \{0, 1, 2\}$$

Productions are defined as follows

$$S \rightarrow S_1 \mid S_2$$

$$S_1 \rightarrow AB$$

$$A \rightarrow 0A1 \mid \varepsilon$$

$$B \rightarrow 2B \mid \varepsilon$$

$$S_2 \rightarrow CD$$

$$C \rightarrow 0C \mid \varepsilon$$

$$D \rightarrow 1D2 \mid \varepsilon$$

Derivations

The production rules are used derive certain string. If $G_1 = (V_n, V_t, P, S)$ be some context free grammar then generation of some language using specific rules are called derivations.

There are two types of derivations

Leftmost Derivations (LMD)

If $G_1 = (V_n, V_t, P, S)$ is a CFG and $w \in L(G_1)$, then a derivation $S \Rightarrow w$ is called a leftmost derivation if and only if all steps involved is derivation have leftmost variable replacement.

Rightmost Derivations (RMD)

If $G_1 = \{V_n, V_t, P, S\}$ is a CFG and $w \in L(G_1)$, then a derivation $S \overset{*}{\underset{RM}{\Rightarrow}} w$ is called a Rightmost derivation if and only if all steps involved is derivation have rightmost variable replacement.

Q) Consider the production rule of a Grammar 'G'

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

Find the leftmost and rightmost derivation for the string $w = a*a+b$

Sol Using leftmost derivation for $w = a*a+b$

$S \xrightarrow[LM]{} S*S$

$S \xrightarrow[LM]{} a*S$ (using $S \rightarrow a$)

$S \xrightarrow[LM]{} a*S+S$ (using $S \rightarrow S+S$)

$S \xrightarrow[LM]{} a*a+S$ (using $S \to a$)

$S \xrightarrow[LM]{} a*a+b$ (using $S \to b$)

using Rightmost derivation for $w = a*a+b$

$S \xrightarrow[RM]{} S+S$

$S \xrightarrow[RM]{} S+b$ (using $S \to b$)

$S \xrightarrow[RM]{} S*S+b$ (using $S \to S*S$)

$S \xrightarrow[RM]{} S*a+b$ (using $S \to a$)

$S \xrightarrow[RM]{} a*a+b$ (using $S \to a$)

Derivation Tree :-

Derivation Tree is a graphical representation for the derivation of the given production rules for a given CFG. The derivation tree is also called parse tree. Following are the properties of any derivation tree :-

1. The root node is always a node indicating start symbol.
2. The derivation is used ~~for~~ from left to Right.
3. The leaf nodes are always terminal nodes
4. The interior nodes are always the non-terminal nodes
5. The collection of leaves from left to right yields the string w.

Q) Consider production rules of a context free grammar

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

Construct derivations tree for string $w = a*a+b$.

Sol

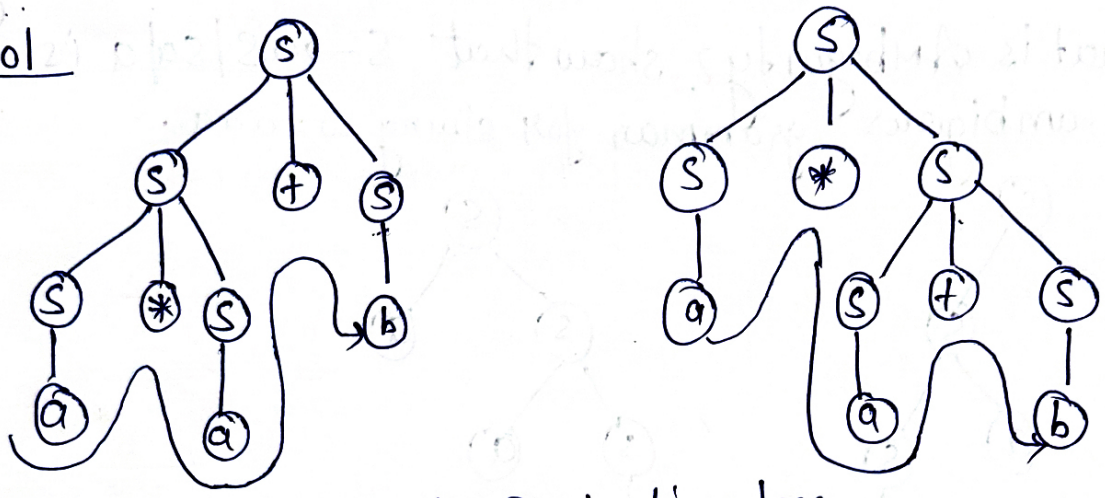


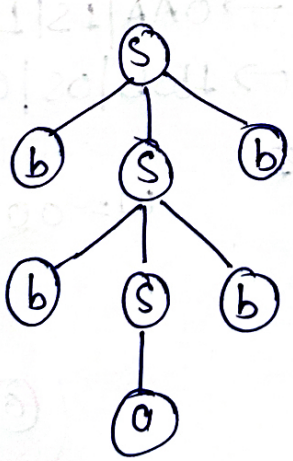
fig: Derivation tree
or
Parse tree.

Q) $S \rightarrow bSb \mid a \mid b$

S → starting symbol

Construction derivation tree from string

w = bbabb



Ambiguity in grammar :

A Grammar G is said to be ambigous if there exists some string $w \in L(G)$ for which there are two or more distinct D.T or there are two or more leftmost or rightmost derivation. Ambigous Grammar are undesirable bcz multible derivation trees provide multible info. about a certain string. Ambiguity is a negative property of a Grammar.

Q) What is Ambiguity? show that $S \to aS|Sa|a$ is an ambiguous grammar for string $w = aaa$.

Sol

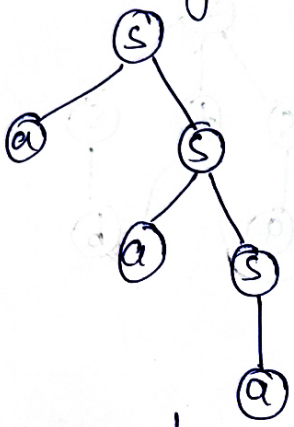


fig: parse tree 1

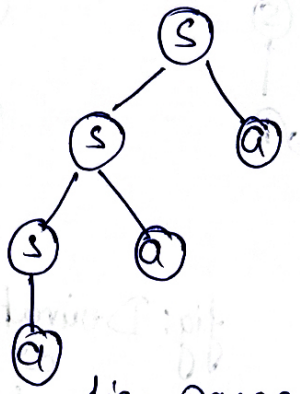


fig: parse tree 2

Q) Explain why the Grammar below that generate Strings with an equal no. zeroes and one's is ambiguous

$S \to 0A|1B$

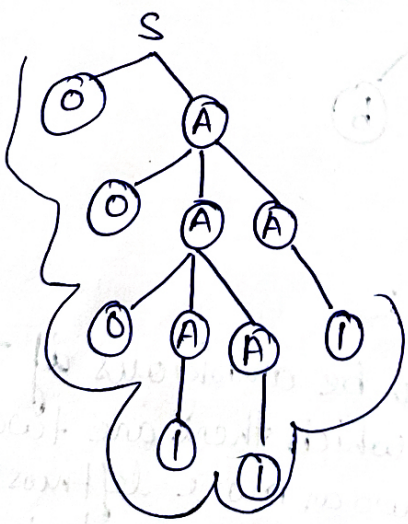
$A \to 0AA|1S|1$

$B \to 1BB|0S|0$

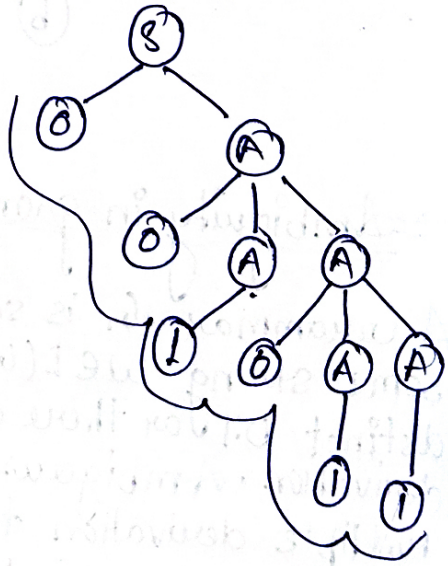
~~$w = 000111$~~ ~~001011~~

Sol

$w = 000111$



$w = 001011$



Simplification of CFG and its Normal Form. (13)

It can be generated by a context-free grammar G with the following properties.

(a) We must eleminate useless symbols, those variables or terminals that do not appears in any derivation of a terinal string from the start symbol.

(b) We must eliminate ϵ-productions, those of the form $X \longrightarrow \epsilon$ for some variable X.

(c) We must eliminate unit production, those of the form $X \longrightarrow Y$ for variables X and Y.

Eliminating Useless symbols

We are going to identified those symbols, which do not play any role in the derivation of any string w in L(G) and eliminate the identified production.

→ A symbols Y in a context-free grammar is useful if and only if:

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

(b) If there is a derivation $S \Rightarrow \alpha Y \beta \Rightarrow w$, $w \in L(G)$, for some α and β, then Y is said to be reachable.

There for reduction of a given grammar G, involves following steps:

(a) Identified non-generating symbols in given CFG and eliminate those productions which conteins non-generating symbols.

(b) Identified non-reachable symbols in grammar and eliminate those productions which contains non reachable symbols.

Q) Remove the useless symbol from the given context free grammar:

$$S \longrightarrow aB \mid bX$$
$$A \longrightarrow BAd \mid bSX \mid a$$
$$B \longrightarrow aSB \mid bBX$$
$$X \longrightarrow SBD \mid aBx \mid ad$$

Sol The non-terminal A and X directly deriving to the strings of terminal a and ad respectively. Hence they are useful symbols. Since $S \rightarrow bX$ and X is useful symbol, hence S is also useful symbol. But B does not derive any string w in V_t^*, so clearly B is a non-generating symbol, so eliminate those productions which contains B. Now grammar becomes.

$$S \longrightarrow bX$$
$$A \longrightarrow bSX \mid a$$
$$X \longrightarrow ad$$

Again here A is non-reachable, since A can not be reached from S (starting non-termial) it can be understand by following.

So eliminate $A \longrightarrow bSX \mid a$, then reduced CFG

$$S \longrightarrow bX$$
$$X \longrightarrow ad.$$

Q) Find CFG with no useless symbol equivalent to

$S \to AB \mid CA$
$B \to BC \mid AB$
$A \to a$
$C \to aB \mid b$

Sol

$S \to AB$ ✗ → aB → aBC — useless

$S \to CA$ ✓ → ba — useful

$B \to BC$ ✗ — useless

$B \to AB$ ✗ — useless

$B \to AB$ ✗ — useless

$A \to a$ — useful

$C \to b$ — useful

$P = \begin{bmatrix} S \to CA \\ C \to b \\ A \to a \end{bmatrix}$

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

Σ or $V_t = \{a, b\}$

S → starting symbol

Q) Consider the following grammar and obtain an equivalent grammar containing no useless grammar symbol.

$A \to xyz \mid Xyzz$
$X \to Xz \mid xY_2$
$Y \to yY_1 \mid X_2$
$Z \to Zy \mid z.$

Sol Since $A \to xyz$ and $Z \to z$ hence A and Z are directly deriving to the string of terminal. they are useful symbols.

Hence X and Y do not lead to a string of terminals that is X and Y are useless symbols. ∴ by eliminating these productions.

$A \to xyz$, $Z \to Zy \mid z$

Now since A is the starting non-terminal and right side of A does not contain Z it means Z is not reachable. Hence after eliminating $Z \rightarrow Zy/z$ grammar becomes.

$$A \rightarrow xyz$$

* Removal of Unit Production

A production of the form

Non-terminal $\rightarrow$ one non-terminal

that is a production of the form $A \rightarrow B$ (where A and B both are non-terminals) is called unit production. Unit production increase the cost of derivation in a grammar.

Algorithm:- Removal of unit production $\Rightarrow$

```
while (there exist a unit production, A → B)
{
Select a unit production A → B, such that there exist a
    production B → α, where α is a terminal.
For (every non-unit production, B → α)
Add production A → α to the grammar.
Eliminate A → B from the grammar.
}
```

Q) Consider the context free grammar G.

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

Remove the unit production.

Sol Given CFG - $S \to AB$
$A \to a$
$B \to C/b$
$C \to D$
$D \to E$
$E \to a$

Contain three unit productions.

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

Now to remove unit production $B \to C$, if there exists a production whose left side has C and right side contains terminal ($C \to a$), but there is no such productions in G. Similar things holds for productions $C \to D$. we try to remove unit production $D \to E$ because there is a production $E \to a$. Therefore eleminate $D \to E$ and introduce $D \to a$ grammar becomes.

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

Now we remove $C \to D$ by using $D \to a$, we get

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

Similarly we can remove $B \to C$ by using $C \to a$, we obtain

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

$C \to a$, $D \to a$, $E \to a$ are useless because if we start deriving from S, these productions will never be used. Hence eliminating them.

$$S \to AB$$
$$A \to a$$
$$B \to a/b$$

Completely reduced grammar.

Q) Consider the following unambiguous expression grammar.

$$I \to a/b/Ia/Ib/I0/I1$$
$$F \to I/(E)$$
$$T \to F/T*F$$
$$E \to T/E+T$$

Identified unit production and remove them.

Sol In the given grammar

$$F \to I$$
$$T \to F$$
$$E \to T$$

are three unit production

Unit production $F \to I$ can be removed by the help of

$$I \to a/b/Ia/Ib/I0/I1$$

and grammar becomes

$$I \to a/b/Ia/Ib/I0/I1$$
$$F \to (E)/a/b/Ia/Ib/I0/I1$$
$$T \to F/T*F$$
$$E \to T/E+T$$

Now we can eliminate $T \to F$ by the help of

$$F \to (E)/a/b/Ia/Ib/I0/I1$$

now grammar becomes

$I \rightarrow a|b|Ia|Ib|Io|I1$
$F \rightarrow (E)|a|b|Ia|Ib|Io|I1$
$T \rightarrow T*F|(E)|a|b|Ia|Ib|Io|I1$
$E \rightarrow T|E+T$

Now remove $E \rightarrow T$ by the help.

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

and finally, unit production free grammar

$I \rightarrow a|b|Ia|Ib|Io|I1$
$F \rightarrow (E)|a|b|Ia|Ib|Io|I1$
$T \rightarrow T*F|(E)|a|b|Ia|Ib|Io|I1$
$E \rightarrow E+T|T*F|(E)|a|b|Ia|Ib|Io|I1$

Removal of ε-productions

To eliminate ε-production from a grammar Gr we use the following technique.

If $A \rightarrow \varepsilon$ is a production to be eliminated then we look for all productions, whose contains A, and replace each occurrence of A in each of these productions to obtain the non ε-productions. Now these resultant non ε-production must be added to the grammar.

eg → Consider the following grammar Gr.

$S \rightarrow ABAC$
$A \rightarrow aA|\varepsilon$
$B \rightarrow bB|\varepsilon$
$C \rightarrow c$

remove the ε-production from the above grammar.

Sol → We have two ε-production to remove (20)
$A \to \varepsilon$ and $B \to \varepsilon$. To eliminate $A \to \varepsilon$ from the grammar the non ε-production to be added are obtained.

List of productions whose right side

$$S \to ABAC$$
$$A \to aA$$

so on replacing each occurrence of A by ε

$$S \to ABC/BAC/BC$$
$$A \to a$$

add these productions to the grammar and eliminate $A \to \varepsilon$, to obtain the following grammar

$$S \to ABAC/ABC/BAC/BC$$
$$A \to aA/a$$
$$B \to bB/\varepsilon$$
$$C \to c$$

Now to eliminate $B \to \varepsilon$ list all productions whose right side contain B,

$$S \to ABAC/BAC/ABC/BC$$
$$B \to bB$$

Replace occurrence of B in each of these productions to obtain non-ε-productions to be added to the grammar.

$$S \to AAC/AC/C$$
$$B \to b$$

Add these productions to the grammar and eliminate $B \to \varepsilon$ from the grammar

$$S \to ABAC/BAC/ABC/BC/AAC/AC/C$$
$$A \to aA/a$$
$$B \to bB/b$$
$$C \to c$$

The grammar without any ε-production.

Q) Design a CFG for regular expression $r = (a+b)^* bb (a+b)^*$, which is free from ε-productions

Sol Let CFG G be required Context free grammar with ε-production

$$S \longrightarrow XY$$
$$X \longrightarrow Zb$$
$$Y \longrightarrow bW$$
$$Z \longrightarrow AB$$
$$W \longrightarrow Z$$
$$A \longrightarrow aA|bA|\varepsilon$$
$$B \longrightarrow Ba|Bb|\varepsilon$$

Finally here $A \to \varepsilon$ and $B \to \varepsilon$ are ε-production A, B are nullable non-terminals.
The CFG without ε-production can be achieved.

$$S \longrightarrow XY$$
$$X \longrightarrow Zb|b$$
$$Y \longrightarrow bW|b$$
$$Z \longrightarrow AB|A|B$$
$$W \longrightarrow Z$$
$$A \longrightarrow aA|bA|a|b$$
$$B \longrightarrow Ba|Bb|a|b$$

Because ε was not generated by G.

CHOMSKY NORMAL FORM

If CFG has only production of the form.

* Non-terminal → string of exactly two non-terminals or of the form.
* Non-terminal → one terminal
* $S \longrightarrow \varepsilon$

is said to be chomsky normal form or CNF.

Conversion to ~~chost~~ chomsky Normal Form

Conversion of the grammar to CNF can be understand by.

$G_1 = [\{S, A, B\}, \{a, b\}, P, S]$: where S is ~~the~~ the start Symbol and P is given by

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

The right side in CNF either contains two non-terminal or one terminal. So it is clear that in the first production

We have to replace terminal 'b' by a non-terminal say C_b and 'a' by C_a, hence grammar

$$S \longrightarrow C_bA \mid C_aB$$
$$A \longrightarrow C_bAA \mid C_bS \mid a$$
$$B \longrightarrow C_aBB \mid C_bS \mid b$$

$$\{ C_a \longrightarrow a \}$$
$$\{ C_b \longrightarrow b \}$$

second rule AA can replace by a non-terminal D

$$AA \longrightarrow D \quad [\text{replaced}]$$
$$BB \longrightarrow E \quad [\text{replaced}]$$

$S \rightarrow C_bA \mid C_aB$
$A \rightarrow C_bD \mid C_aS \mid a$
$B \rightarrow C_aE \mid C_bS \mid b$

$C_a \rightarrow a$
$C_b \rightarrow b$
$D \rightarrow AA$
$E \rightarrow BB$

The above grammar is equivalent chomsky normal form.

Q) change the following grammar into to CNF

$S \rightarrow abSb \mid a \mid aAb$
$A \rightarrow bS \mid aAAb$

Sol Replace aA by B_a and A_b by B_b

$S \rightarrow abSb \mid a \mid aB_b$
$A \rightarrow bS \mid B_aB_b$

$B_a \rightarrow aA$
$B_b \rightarrow Ab$

Now replace ab by C and Sb by D

$S \rightarrow CD \mid a \mid aB_a$
$A \rightarrow bS \mid B_aB_b$

$B_a \rightarrow aA$
$B_b \rightarrow Ab$
~~$B_b \rightarrow$~~
$C \rightarrow ab$
$D \rightarrow Sb$

Now replace a by X_a, b by X_b

$S \rightarrow CD \mid a \mid X_aB_a$
$A \rightarrow X_aS \mid B_aB_b$
$B_a \rightarrow X_aA$
$B_b \rightarrow X_aB$

$C \rightarrow AX_b$
$D \rightarrow SX_b$
$X_a \rightarrow a$
$X_b \rightarrow b$

Greibach Normal form (GNF)

GNF stands for Greibach normal form. A CFG is in GNF, if all the production rules satisfy:

(1) A start symbol generating ε. For example, $S \rightarrow \varepsilon$

(2) A non-terminal generating a terminal

$$A \rightarrow a$$

(3) A non-terminal generating a terminal which is followed by any no. of non-terminals.

$$S \rightarrow aASB.$$

Q) Consider the following grammar and write its equivalent GNF

$$S \rightarrow AB$$
$$A \rightarrow aA \mid bB \mid b$$
$$B \rightarrow b.$$

Sol The GNF each production should be of the $A \rightarrow aV/a$ from where $V \in V_n^*$

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

$$S \rightarrow aAB \mid bBB \mid bB$$
$$A \rightarrow aA \mid bB \mid b$$
$$B \rightarrow b$$

which is in Greibach normal form.

Regular Grammars

A regular grammar may be left linear or right linear

"If all production of a CFG are of the form $A \to wB$ or $A \to w$, where A and B are variables and $w \in V_t^*$, then we say that grammar is right linear".

"If all the production of a CFG are of the form $A \to Bw$ or $A \to w$, we call it left linear."

A right or left linear grammar is called a regular grammar.

Q) Write the left linear and right linear grammar for the regular expression.

$$r = 0(10)^*$$

Sol Regular expression $r = 0(10)^*$, it means any string which starts with 0 followed by any number of 10's is in regular expression.

Left linear :- Let grammar be $G = \{V, V_t, P, S\}$

$$V_n = \{S\}$$
$$V_t = \{0, 1\}$$

and P is defined as

$$S \to S10 \mid 0$$

Right linear : Let grammar be $G' = (V_n, V_t, P, S)$

$$V_n = \{S, A\}$$
$$V_t = \{0, 1\}$$

and P is defined as

$$S \to 0A$$
$$A \to 10A \mid \varepsilon$$

Q) Write the right linear and left linear regular grammar for the regular expression $r = (ab)^* a$. (28)

Sol) It contains set of strings in which every string start with any number of ab's and end with single a.

Right linear :- Let grammar

$$G_1 = (\{S\}, \{a, b\}, S, P)$$

P_1 is defined as follows:

$$S \to abS \mid a$$

Left linear :- Let grammar be G_2

$$G_2 = (\{S, S_1, S_2\}, \{a, b\}, S, P_2)$$

with production

$$S \to S_1 a$$
$$S_1 \to S_1 ab \mid \varepsilon$$

is left linear.

Both G_1 and G_2 are regular grammars.

Conversion of Regular Grammar into Finite Automata.

(a) The number of states in the automata will be ~~equal~~ equal to the no. of non-terminal plus one.

Each state in automata represents each non-terminal in the regular grammar.

The additional state will be the final state of the automata.

(b)

(i) For every production $A \to aB$ make $\delta(A, a) = B$ that is make an are labelled 'a' from A to B.

(ii) For every production $A \to a$, make $\delta(A, a) =$ final state

(iii) For every production $A \to \varepsilon$, make $\delta(A, \varepsilon) = A$ and A will be final state.

Q) Consider the following grammar

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

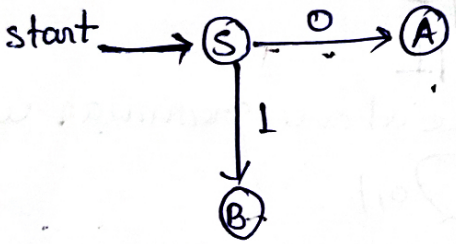
$A \rightarrow 0S \mid 1B \mid 1$

$B \rightarrow 0A \mid 1S$

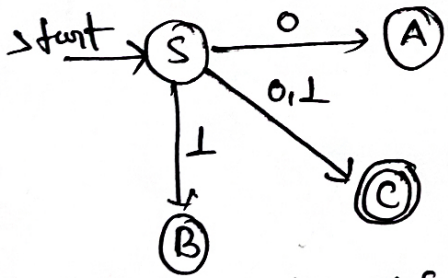
Find the automation for this grammar.

Sol The automation for this grammar. four states S, A, B and one additional say C, which will be the final state.

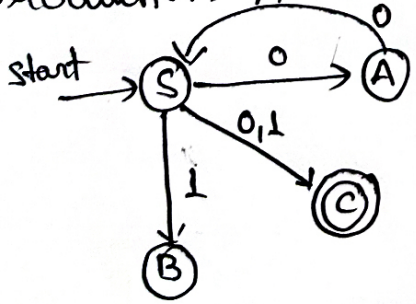
Step-1. For production $S \rightarrow 0A$ and $S \rightarrow 1B$



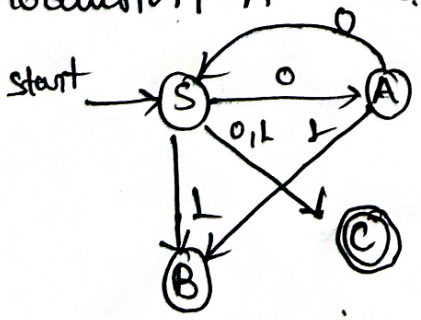
Step-2. For production $S \rightarrow 0 \mid 1$



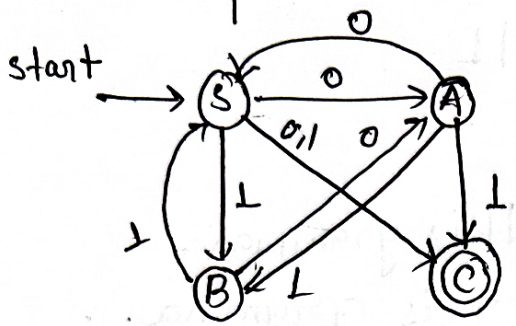
Step-3 For production $A \rightarrow 0S$



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



step-5 For the production $B \rightarrow 0A$ and $B \rightarrow 1S$

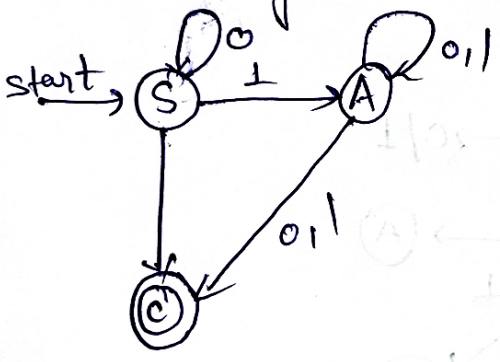


There is no need to show all the steps every time and FA can be generated directly.

Q) Give the automation for the following grammars.

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

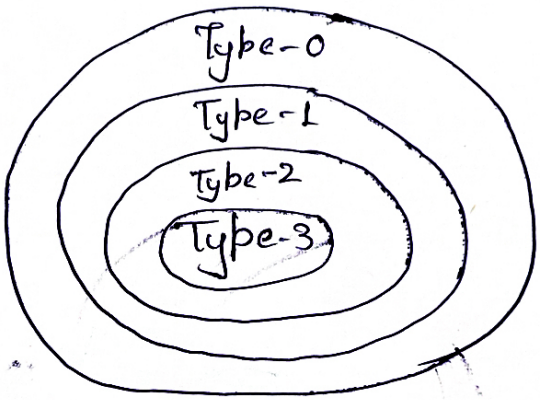
Sol The automation for the above grammar will be



The Chomsky Hierarchy

Noum chomsky, a founder of formal language theory provided an initial classification into four lang types.

Type-0 (Unrestricted grammar)
Type-1 (Context sensitive grammar)
Type-2 (Context free grammar)
Type-3 (Regular grammar)



The original chomsky hierarchy

Unrestricted (Type-0) grammars

The ~~unrestriste~~ unrestricted grammar is defined as

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

V_n = a finite set of non-terminals
V_t = a finite set of terminals
S = starting non-terminal, $S \in V_n$

and P is set of productions of the following form

$$\alpha \longrightarrow \beta$$

where α and β are arbitrary string of grammar symbols with $\alpha \neq \varepsilon$. These grammars are known as type-0, phase-structure or unrestricted grammars.