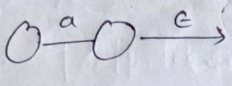
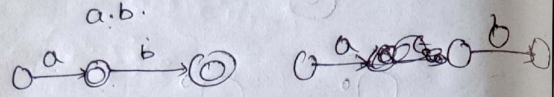
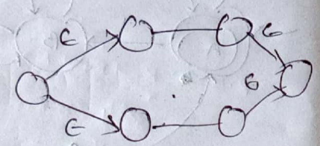
$r_1 . r_2$



a.b.



a+b



Unit-3

Context free language

<sentence>

<predicate> ← → <Nounphrase>

<verb>   <Noun> <Article>

< > – BNF notation.

Ex:- $x = (a+b)*c$.

$A \Rightarrow id = E$

$\Rightarrow id = E * E$

$\Rightarrow id = (E) * E$

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

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

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

$\Rightarrow id = (id+id) * id$.

* Grammar is a 4 tuple (V, T, S, P).

  V - set of variables
  T - set of Terminals
  S - set of start symbol.
  P - set of productions.

Conditions: for context free language:

$A \rightarrow x$ — A is head of production & x is body of

$A \in V$

$x \in (V \cup T)^*$

* Terminals

  small letters, operators, Digits, punctuation marks, keywords written in thick form.

* Non Terminals.

  Upper case letters, Letter S, lowercase letters with italic font, (Upper case letters (X, Y, Z) can be either terminal or nonterminal), (lower case (u, v, w, x, y, z) string of terminal symbols), (Greek letters (string of grammar symbols can be either of them).

* More than one body for a head then separate it by a vertical line.

  $A \rightarrow x_1 | x_2 | \ldots x_n$.

Scanned with CamScanner

Ex:

$S \rightarrow AS \mid \epsilon$

$A \rightarrow aa \mid ab \mid ba \mid bb.$

Give LMD, RMD and Parse tree for a sentence.

(i) aabbaa

(ii) baabab.

Sol^n

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

$V = \{S, A\}.$

$T = \{a, b\} \cup \epsilon$

$S = S$

$P = AS \mid \epsilon \mid aa \mid ab \mid ba \mid bb.$

$S \xRightarrow{LMD} \underline{A}S$

$\Rightarrow aa\underline{S}$

$\Rightarrow aa\underline{A}S$

$\Rightarrow aabb\underline{S}.$

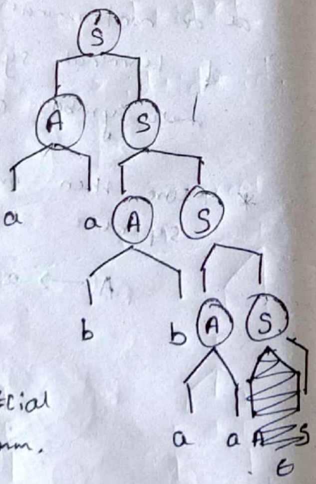
$\Rightarrow aabb\underline{A}S$

$= aabbaa\underline{S}$

$= aabbaa\epsilon$

$= aabbaa$

Left sentential form.



$S \xRightarrow{RMD} A\underline{S}$

$\Rightarrow AA\underline{S}$

$\Rightarrow AAAS.$

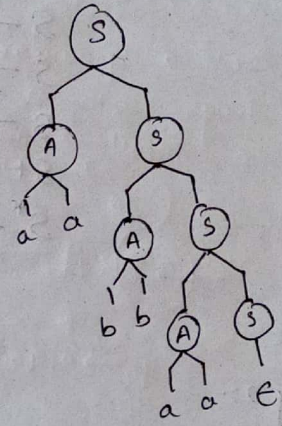
$\Rightarrow AA\underline{A}$

$\Rightarrow A\underline{A}aa$

$\Rightarrow \underline{A}bbaa$

$\Rightarrow aabbaa.$

Right sentential form.



~~Bo~~ sequence of production application (parsing).

$S \rightarrow AS$

$A \rightarrow aa$

$S \rightarrow AS$

$A \rightarrow bb$

$S \rightarrow AS$

$A \rightarrow aa$

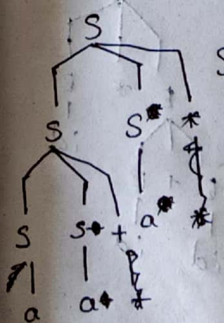
$S \rightarrow \epsilon$

Ex: write LMD and RMD and parsing of the sentence. Find whether its valid.

$S \rightarrow SS+ \mid SS* \mid a$

i) ~~aaAa*~~ aa+a*

LMD.



$S \Rightarrow \underline{S}S*$

$\Rightarrow \underline{S}S+S*$

$\Rightarrow a\underline{S}+S*$

$\Rightarrow aa+\underline{S}*$

$\Rightarrow aa+a*$

RMD.

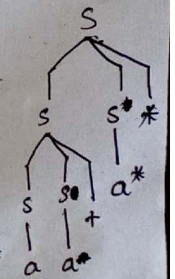
$S \Rightarrow S\underline{S}*$

$\Rightarrow \underline{S}a*$

$\Rightarrow S\underline{S}+a*$

$\Rightarrow \underline{S}a+a*$

$\Rightarrow aa+a*$



$S \rightarrow SS^*$

$S \rightarrow SS^*+$

$S \rightarrow a$

$S+ \rightarrow a+$

$S^* \rightarrow a^*$

Eg 2:-

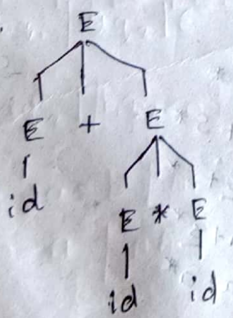
$E \rightarrow E+E \mid E*E \mid (E) \mid id.$

$w =$ id+id * id.

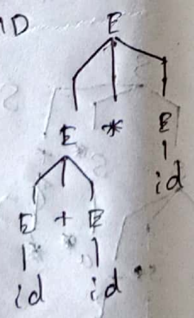
$E \overset{\cancel{RMD}}{\Longrightarrow} E+\underline{E}$	$E \overset{L\ RMD}{\Longrightarrow} \underline{E} * E$
$\Rightarrow E+E*\underline{E}$	$\Rightarrow \underline{E}+E*E$
$\Rightarrow E+\underline{E}*id$	$\Rightarrow id+\underline{E}*E$
$\Rightarrow \underline{E}+id*id$	$\Rightarrow id+id*\underline{E}$
$\Rightarrow id+id*id$	$\Rightarrow id+id*id.$

Correct ~~E → E+E~~ incorrect

RMD.



LMD



This happens when Language is ambiguous.

* priority of operators is considered.

Ex:-

① $S \rightarrow AaAb \mid BbBa.$

$A \rightarrow aAb \mid bAB \mid d.$

$B \rightarrow aB \mid bB \mid a.$

(i) aabbaa (ii) badbaba aHb.

② $E \rightarrow ET+ \mid T$

$T \rightarrow TF* \mid F$

$F \rightarrow FP\uparrow \mid P.$

$P \rightarrow E \mid id$

(i) id id id * + id +

(ii) id id id ↑ id * id

(i) $S \overset{LMD.}{\Longrightarrow} \underline{B}bBa.$

$\Longrightarrow a\underline{B}bBa$

$\Longrightarrow aab\underline{B}a$

$\Longrightarrow aabb\underline{B}a.$

$\Longrightarrow aabbaa //.$

$S \overset{RMD.}{\Rightarrow} Bb\underline{B}a$

$\Rightarrow Bbb\underline{B}a.$

$\Rightarrow \underline{B}bbaa.$

$\Rightarrow a\underline{B}bbaa.$

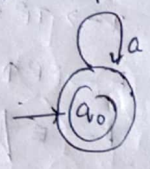
$\Rightarrow aabbaa //.$

(ii) $S \overset{LMD.}{\Longrightarrow}$

Ex:-

Write a grammar to accept a set of strings with '0' or more combination of an input symbol 'a'.

i)



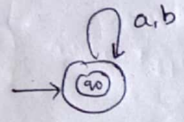
$V = \{q_0\}$

$T = a.$

$S \rightarrow aS \mid \epsilon$

$q_0 \rightarrow a q_0 \mid \epsilon$

ii)

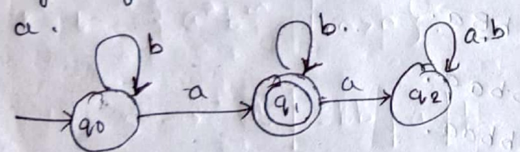


$V = \{q_0\}$

$T = a, b$

$q_0 \rightarrow a q_0 \mid b q_0 \mid \epsilon$

Write a DFA for the language set of strings of a's and b's having exactly one a.



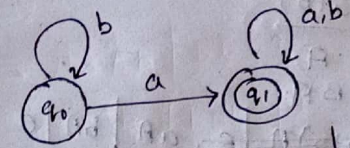
$q_0 \rightarrow \cancel{bq_0}\ a q_1 \mid b q_0$

$q_1 \rightarrow b q_1 \mid a q_2 \mid \epsilon$

$q_2 \rightarrow a q_2 \mid b q_2$

ii) atleast one 'a'

$b^* \cancel{(a+b)^*}\ a(a+b)^*$

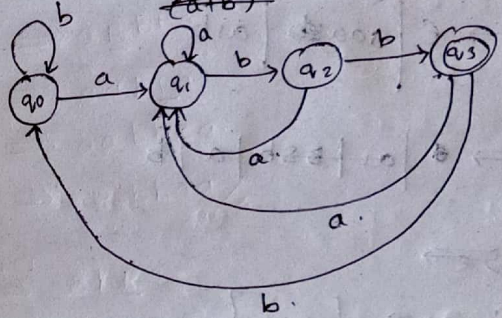


$q_0 \rightarrow b q_0 \mid a q_1$

$q_1 \rightarrow a q_1 \mid b q_1 \mid \epsilon$

iii). ~~set~~ of s Ending with abb.

$\cancel{(a+b)}^* \cancel{abb}.$



$q_0 \rightarrow b q_0 \mid a q_1$

$q_1 \rightarrow a q_1 \mid b q_2.$

$q_2 \rightarrow a q_1 \mid b q_3.$

$q_3 \rightarrow a q_1 \mid b q_0 \mid \epsilon.$

Write a regular Expression in grammar.

having substring aa.

$$\underbrace{(a+b)^*}_{A}\ \underbrace{aa}_{B}\ \underbrace{(a+b)^*}_{A}$$

$S \rightarrow ABA$

$A \rightarrow \cancel{(a+b)A \mid \epsilon}\ aA \mid bA \mid \epsilon$

$B \rightarrow aa$

* $L = \{a^n b^{n-3} \mid n \geq 3\}$.

$S \rightarrow \epsilon \mid aaa \mid asb$

$\cancel{S \rightarrow} \mid \cancel{a} \mid \cancel{bsb} \mid a \mid b$

* $L = \{a^n b^m \mid m > n\}$.

$S \rightarrow \cancel{as \mid sbb \mid a \mid b}$

$S \rightarrow \cancel{asb \mid \epsilon}\ asb \mid B$.

$B \rightarrow b \mid bB$.

* $L = \{a^n b^m \mid m < n\}$.

$S \rightarrow asb \mid A$

$A \rightarrow a \mid aA$

* $L = \{a^n b^m \mid m <> n\}$.

$S \rightarrow asb \mid A \mid B$.

$A \rightarrow a \mid aA$

$B \rightarrow b \mid bB$.

$V = \{S, A, B\}$.

$T = \{a, b\}$

$S = \{S$.

$P =$

* $L = \{ wwR \mid w \in (a+b)^* \}$.

$\Sigma = (a, b)$

$w = a^n b^n$.

$w^R = b^n a^n$

$ww^R = a^n b^n b^n a^n$.

$S \rightarrow asa \mid bsb \mid a \mid b \mid \epsilon$ – odd aba.

$S \rightarrow asa \mid bsb \mid \epsilon$ – even abba.

* signed or unsigned digit.

$I \rightarrow N \mid S_1 N$

$N \rightarrow D \mid ND$.

$D \rightarrow 0 \mid 1 \mid \ldots \mid 9$

$S_1 \rightarrow + \mid -$

$I \xRightarrow{LMD} S_1 N$

$\Rightarrow +N$

$\Rightarrow +ND$

$\Rightarrow +NDD$.

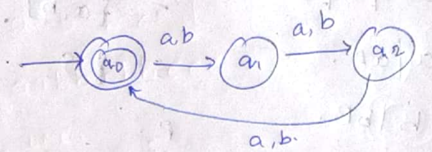
$\Rightarrow +DDD$

$\Rightarrow +1DD$

$\Rightarrow +12D$

$\Rightarrow +123$ //

$* \left((a+b)(a+b)(a+b) \right)^*$



$q_0 - S$

$q_1 - S_1$

$q_2 - S_2$.

$S \rightarrow aS_1 \mid bS_1 \mid \epsilon$

$S_1 \rightarrow aS_2 \mid bS_2$.

$S_2 \rightarrow aS \mid bS$.

$W = abaabb$.

LMD

$S \Rightarrow a\underline{S_1}$

$\Rightarrow ab\underline{S_2}$

$\Rightarrow aba\underline{S}$

$\Rightarrow abaaS_1$

$\Rightarrow abaabS_2$

$\Rightarrow abaabbS$

$\Rightarrow abaabb$ ✓.

$\omega = abaa$.

$S \Rightarrow a\underline{S_1}$

$\Rightarrow ab\underline{S_2}$

$\Rightarrow abaS$

$\Rightarrow abaa\underline{S_1}$

$= \cancel{abaa}$ ✗.

$* \ L = \{ a^i b^j c^k, \ i = j+k, \text{ where } j, k \geq 0.$

$L = a^{j+k} b^j c^k$.

$= \underbrace{a^k \underbrace{a^j b^j}_{S_1} c^k}_{S}$.

$S \longrightarrow aSc \mid S_1 \mid \epsilon$

$S_1 \longrightarrow aS_1b. \mid \epsilon$

$*$ $j=2$ & $k=3$.

$\omega = a^5 b^2 c^3$

$\omega = a^3 a^2 b^2 c^3$.

$S \overset{LMD}{\Longrightarrow} aSc$

$\Rightarrow aaScc$

$\Rightarrow aaaSccc$

$\Rightarrow aaaS_1ccc$.

$\Rightarrow aaaaS_1bccc$.

$\Rightarrow aaaaaS_1bbccc$.

$\Rightarrow aaaaa\,bbccc$.

$* \ L = \{ a^i b^j c^k, k = i+j.$ where $i, j \geq 0 \}$

$L = \underbrace{a^i \underbrace{b^j c^j}_{S_1} c^i}_{S}$

$S \rightarrow aSc \mid S_1 \mid \epsilon$

$S_1 \rightarrow bS_1c \mid \epsilon$

$*$ $j = i+k$. $i, k \geq 0$.

$L = \underbrace{a^i b^i}_{S} \underbrace{b^k c^k}_{S_1}$.

$\cancel{S \rightarrow aSbS \mid S_1 \mid \epsilon}$

$\cancel{S_1 \rightarrow bS_1c \mid S \mid \epsilon}$

$S \rightarrow S_1 S_2$

$S_1 \rightarrow aS_1b \mid \epsilon$

$S_2 \rightarrow bS_2c \mid \epsilon$

* $i \geq 2 \quad k \geq 3$.

$a^2 b^2 b^3 c^3$.

aabbbbbccc.

$S \xRightarrow{LMD} S_1 S_2$

$\Rightarrow a S_1 b S_2$.

$\Rightarrow aa S_1 bb S_2$.

$\Rightarrow aabb S_2$

$\Rightarrow aabbb S_2 c$

$\Rightarrow aabbbb S_2 cc$

$\Rightarrow aabbbbb S_2 ccc$

$= aabbbbbccc$ //

* $L = \{ a^n w w^R b^n \mid w \in (a+b)^*, n \geq 0 \}$.

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

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

$n = 3$.

w - aaa abaaba bbb.

$S \xRightarrow{LMD} a S b$

$\Rightarrow aa S bb$

$\Rightarrow aaa S bbb$.

$\Rightarrow aaa S_1 bbb$.

$\Rightarrow aaaa S_1 abbb$.

$\Rightarrow aaaab S_1 babbb$.

$\Rightarrow aaaaba S_1 ababbb$.

$\Rightarrow aaaabaababbb$ //

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

Formal definition for language of a Grammar.

If G is a $= \{V, T, S, P\}$ be a context free grammar the language of G denoted as L(G) defined to be the set of terminal strings that have a derivations from the start symbol.

set notation.

$L(G) = \{ w \text{ in } T^* \mid S \overset{*}{\underset{G}{\Rightarrow}} w \}$

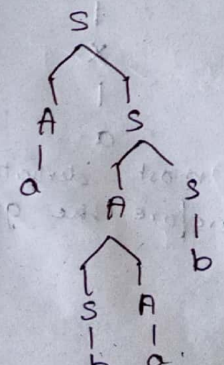
* $S \rightarrow AS \mid b$ (1) (2)

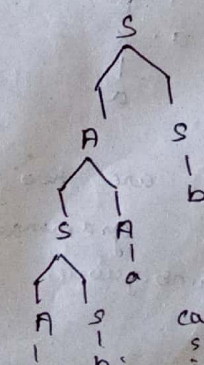
$A \rightarrow SA \mid a$ (3) (4)

Gramar is ambigous on w = abab.

shift
accept
reduce
error.

1, 4, 1, 3, 2, 4, 2	1, 2, 1, 4, 2, 4, 2
$S \xRightarrow{LMD} AS$.	$S \xRightarrow{LMD} AS$
$\Rightarrow aS$	$\Rightarrow SAS$
$\Rightarrow aAS$.	$\Rightarrow ASAS$
$\Rightarrow aSAS$.	$\Rightarrow aSAS$.
$\Rightarrow abAS$	$\Rightarrow abAS$
$\Rightarrow abaS$	$\Rightarrow abaS$
$\Rightarrow abab$.	$\Rightarrow abab$.





Since there are two left most derivations for a grammar with correspon ding parse tree for each we can reach to a conclu sion that the grammar is ambiguous.

* P_2 $S \to aS \mid X$ (1) (2)

$X \to aX \mid a$ (3) (4)

$V = \{S, X\}$.

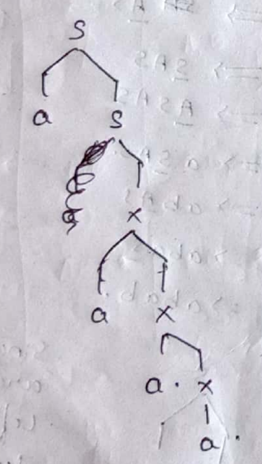
$T = \{a\}$

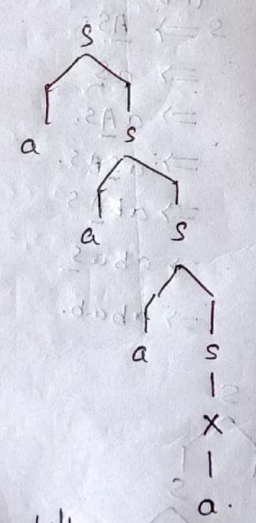
$S = \{S\}$.

P =

$w = aaaa$.

LMD 1,2,3,3,4	LMD. 1,1,1,2,4
$S \Rightarrow a\underline{S}$	$S \Rightarrow a\underline{S}$
$\Rightarrow a\underline{X}$	$\Rightarrow aa\underline{S}$
$\Rightarrow aa\underline{X}$	$\Rightarrow aaa\underline{S}$
$\Rightarrow aaa\underline{X}$	$\Rightarrow aaa\underline{X}$
$\Rightarrow aaaa$.	$\Rightarrow aaaa$.





As there are two left most derivations for a grammar Therefore the grammar is ambiguous.

* $S \to ABaC$

$A \to BC$

$B \to b \mid \epsilon$

$C \to D \mid \epsilon$

$D \to d$

(1) finding nullable variables V_N

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

(2) Exclude ϵ production & rewrite.

$S \to ABaC \mid BaC \mid AaC \mid ABa \mid aC \mid Ba \mid Aa \mid a$.

$A \to BC \mid C \mid B$

$B \to b$.

$C \to D$

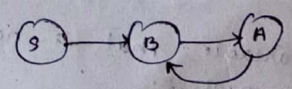
$D \to d$.

unit production removing technique.

(1) Listing all Non unit production.

$S \to Aa \mid B$.	$S \to Aa \mid bb. \mid a \mid bc$
$B \to A \mid bb$.	$B \to bb. \mid a \mid bc$
$A \to a \mid bc \mid B$.	$A \to a \mid bc. \mid bb$.

(2) dependency graph for unit production.



$S \overset{*}{\Rightarrow} B$

$S \overset{*}{\Rightarrow} A$

$B \overset{*}{\Rightarrow} A$

$A \overset{*}{\Rightarrow} B$.

Removal of useless production.

Useful production: Let $G = \{V, T, S, P\}$ be a context free grammar any variable $A \in V$ is said to be useful if and only if there is ~~one~~ ~~to~~ atleast one $w \in L(G)$ such that $S \overset{*}{\Rightarrow} xAy \overset{*}{\Rightarrow} w$

Ex:-

$S \longrightarrow aS \mid A \mid C$

$A \longrightarrow a$

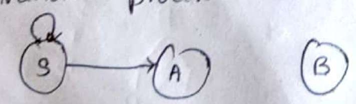
$B \longrightarrow aa$

$C \longrightarrow aCb.$

(1) Remove the variable that doesn't derive any terminal String

OV	NV	Prod^n
ϕ	A, B	$A \rightarrow a$ $B \rightarrow aa$
A, B	A, B, S	$S \rightarrow A$
A, B, S	A, B, S	$S \rightarrow aS$

(2) Remove the variable that is not taking part in the derivation process.



B is not reachable & hence it is useless

$S \longrightarrow A \mid aS$

$A \longrightarrow a$

* Ex simplify the grammar.

$S \longrightarrow a \mid aA \mid B \mid C$

$A \longrightarrow aB \mid \epsilon$

$B \longrightarrow Aa.$

$C \longrightarrow cCD$

$D \longrightarrow ddd.$

(1) Removal of ϵ production.

$V_N = \{A, , \}$

$S \longrightarrow a \mid aA \mid B \mid C$

$A \longrightarrow aB.$

$B \longrightarrow Aa \mid a$

$C \rightarrow cCD.$

$D \longrightarrow ddd$

(2) Removal of unit productions.

List all non-unit productions.

$S \longrightarrow a \mid aA \mid cCD \mid Aa$

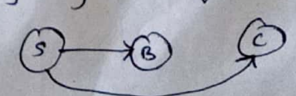
$A \longrightarrow aB.$

$B \longrightarrow Aa \mid a.$

$C \longrightarrow cCD$

$D \longrightarrow ddd.$

dependency graph for unit productions



$S \overset{*}{\Rightarrow} B$

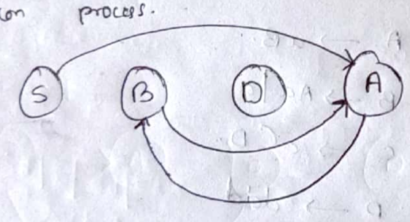
$S \overset{*}{\Rightarrow} C.$

(3) Removal of useless production.

Remove the variable that does not derive any terminal string.

OV	NV	Prod^n.
ϕ	S, B, D.	$S \to a$ $B \to a$. $D \to ddd$
S, B, D	S, B, D, A	$A \to aB$. ~~$B \to Aa$.~~
S, B, D, A	S, B, D, A	$S \to aA \mid Aa$ $B \to Aa$.

Remove the variable that is not taking part in derivation process.



D is not reachable & hence it is useless.

$S \to a \mid aA \mid Aa$

$A \to aB$.

$B \to Aa \mid a$.

* Ex :-

$S \to aAa \mid bBb \mid \epsilon$

$A \to \epsilon \mid a$.

$B \to \epsilon \mid b$.

$C \to CDE \mid \epsilon$

$D \to A \mid B \mid ab$.

(1) Removal of ϵ production :

$V_N = \{ S, C, A, B, D \}$.

$S \to aAa \mid bBb \mid aa \mid bb$.

$A \to C \mid a$.

$B \to \epsilon \mid b$.

$C \to CDE \mid DE \mid CE \mid E$

$D \to A \mid B \mid ab$.

(2) Removal of unit productions.

List of all non unit production.

$S \to aAa \mid bBb \mid aa \mid bb.$ ~~CDE | DE | CE~~

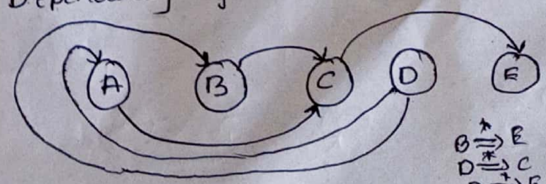
$A \to a \mid CDE \mid DE \mid CE$

$B \to b. \mid CDE \mid DE \mid CE$

$C \to CDE \mid DE \mid CE$

$D \to ab \mid a \mid CDE \mid DE \mid CE \mid b$

Dependency graph of unit productions



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

$A \overset{*}{\Rightarrow} C$

$B \overset{*}{\Rightarrow} C$

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

$D \overset{*}{\Rightarrow} A$

$D \overset{*}{\Rightarrow} B$

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

$D \overset{*}{\Rightarrow} C$

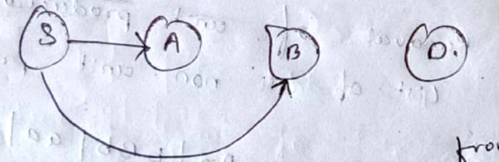
$D \overset{*}{\Rightarrow} E$

(3) Removal of useless production.

Remove the variable that doesnot derive any terminal string.

OV	NV	Prod^n
ϕ	S, A, B, D	$S \to aa$ $S \to bb$ $A \to a$ $B \to b$ $D \to ab \mid a \mid b$
S, A, B, D	S, A, B, D	$S \to aAa$ $S \to bBb$
S, A, B, D	S, A, B, D	ϕ —

Remove the variable that is not taking part in derivation process.



D is not reachable from S & hence it is useless, S, A & B are useful.

$S \to aAa \mid bBb \mid aa \mid bb$.

$A \to a$ ~~...~~

$B \to b$

Chomsky Normal Form.

Definition: If G is a grammar representing non empty context free language without ϵ and unit productions in which all productions are in one of two simple forms either.

① $A \to BC$ where A, B, C are all variables.

② $A \to a$ where A is variable and a is terminal

Such a grammar we say that it is in Chomsky Normal Form.

(1) Arrange in such a way that all the length of 2 or more consists only of variables.

(2) Break bodies of length 3 or more into a cascade of productions each with a body consisting of two variables.

Ex:-

$E \to E+T \mid T$.

$T \to T*F \mid F$

$F \to (E) \mid id$.

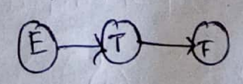
Removal of unit productions.

$E \to E+T \mid T*F \mid (E) \mid id$ $\quad F \to (E)$

$T \to T*F \mid (E) \mid id$ $\quad F \to id$

~~$F \to id \mid (E)$~~

dependency graph.



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

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

$T \overset{*}{\Rightarrow} F$

~~Removal~~ of Converting to variables

$E \to E B_+ T \mid T B_* F \mid B_c E B_) \mid id$

$B_+ \to +$

$B_* \to *$

$B_c \to ($

$B_) \to)$

$T \to T B_* F \mid B_c E B_) \mid id.$

$F \to B_c E B_)$

$F \to id.$

Length 2

$E \to E D_1 \mid T D_2 \mid B_c D_3 \mid id.$

$D_1 \to B_+ T$

$D_2 \to B_* F$

$D_3 \to E B_)$

$B_+ \to +$

$B_* \to *$

$B_c \to ($

$B_) \to)$

$T \to T D_2 \mid B_c D_3 \mid id.$

$F \to B_c D_3.$

$F \to id.$

Greg's back NF.

A context free grammar is said to be in GBNF if all productions are of the form.

$A \to ax$ where $a \in T$ & $x \in V^*$

Ex :- $S \to AB$ $\Rightarrow$ $S \to$ ~~aA | bB | b~~

$A \to aA \mid bB \mid b.$

$B \to b.$

Conversion to GNF

$S \to aAB \mid bBB \mid bB.$

$A \to aA \mid bB \mid b.$

$B \to b.$

* yacc specification is used as input to the yacc tool.

yacc programming

Using yacc tool write a parser to validate the syntax of airthmetic expression.

step 1 :- Grammar.

$S \to e$

$e \to e+e \mid e-e \mid e*e \mid e/e \mid e^{\wedge}e \mid ID \mid$

D

yacc program: to evaluate the airthmetic expression and for it's validity.

```
%{
    #include <stdio.h>
%}
%token ID D
%left '+' '-'
%left '*' '/' %right '^'
%%              printf("%d", $1); exit(0);
    S : e '\n' {printf("valid\n");};
    ;
~~e : e+e | e-e | e*e | e/e | e^e | ID | D {printf("valid~~
    ;
    e : e'+'e { ~~print~~ ~~valid~~ } {$$ = $1+$3;}
      | e'-'e   {$$= $1 - $3}
      | e'*'e   {$$ = $1 * $3;}
      | e'/'e   {$$ = $1 / $3;}
      | e'^'e   {$$ = $1 ^ $3;}
      | ID      {$$ = $1;} X
      | D       {$$ = $1;}.
    ;
%%
main()
{
    printf("Enter A.E :");
    yyparse();
    return 0;
}
yyerror()
{   printf("... Invalid ... \n");
}
```

evaluation part

lexprogram associated with the previous ~~lex~~ yacc program:

```
%{
    # include "y.tab.h"
    # include <stdio.h>
    extern int yylval;
%}
%%                  yyval = atoi(yytext);
[0-9]+              {return D;}
~~[a-zA-Z]*~~       ~~{return ID;}~~ X
[\+\-\*\/\(\)\^]    {return yytext[0];}.
.|
\n                  {return yytext[0];}.
%%
```

✓ when program used for validity.

→ yacc -d file.yacc

→ lex files.l

→ cc y.tab.c lex.yy.c -ll

→ ./a.out

* stored in $1

* For division action - { if ($3 == 0)
  {printf("Divide of zero")
  exit(0);}
  else
  $$ = $1/$3;
  }.