In CFG; RHS of a production can be any string of variables & terminals. When production in G satisfy certain restrictions, then G is said to be in a 'normal form'.

We study 2 normal forms among many normal forms :-

(i) Chomsky Normal Form (CNF)
(ii) Greibach Normal Form (GNF)

CHOMSKY NORMAL FORM (CNF)

(CYK algorithm for PARSING uses CNF form)

A CFG G is in CNF if every production is of the form $A \to a$, or $A \to BC$, and $S \to \Lambda$ is in G if $\Lambda \in L(G)$. When Λ is in $L(G)$, we assume that S does not appear on the RHS of any production.

Remark :- For a grammar in CNF, the derivation tree has the property : every node has atmost 2 descendants - either 2 internal vertices or a single leaf.

REDUCTION TO CNF

Theorem : For every CFG, there exists an equivalent grammar G_2 in Chomsky Normal form.

PROOF :- Construction of Grammar in CNF

STEP 1 :- Elimination of null productions & unit productions. to get $G=(V_N, \Sigma, P, S)$.

STEP 2 :- Elimination of terminals on RHS.

We define $G_1 = (V_N', \Sigma, P_1, S')$; where P_1 & V_N' are constructed as follows :-

(i) All the productions in P of the form $A \to a$ or $A \to BC$ are included in P_1. All the variables in V_N are included in V_N'.

(ii) Consider $A \to X_1 X_2 \dots X_n$ with some terminal on RHS. If X_i is a terminal say a_i, add a new variable C_{a_i} to V_N' and $C_{a_i} \to a_i$ to P_1. In production $A \to X_1 X_2 \dots X_n$, every terminal on RHS is replaced by corresponding new variable & the variables on the RHS are retained. The resulting production is added to P_1. Thus, we get $G_1 = (V_N', \Sigma, P_1, S)$.

STEP 3: Restricting the no. of variables on RHS :-

For any production in P_1, the RHS consists of either a single terminal (or Λ in $S \to \Lambda$) or two or more variables. We define $G_2 = (V_N'', \Sigma, P_2, S)$ as follows.

(i) All the productions in P_1 are added to P_2 if they are in the required form. All the variables in V_N' are added to V_N''.

(ii) Consider $A \to A_1 A_2 \dots A_m$, where $m \geq 3$. We introduce new productions $A \to A_1 C_1$, $C_1 \to A_2 C_2$, $\dots$ $C_{m-2} \to A_{m-1} A_m$ & new variables $C_1, C_2, \dots, C_{m-2}$. These are added to P'' and V_N'', respectively.

Thus, we get G_2 in CNF.

Ques :- Reduce the following grammar G to CNF. G is $S \to aAD$, $A \to aB \mid bAB$, $B \to b$, $D \to d$.

Soln :- As there are no null or unit productions, we can proceed to step 2.

STEP 2 :- (i) $B \to b$, $D \to d$ are included in P_1

(ii) $S \to aAD$ gives rise to $S \to C_a AD$; $C_a \to a$

$A \to aB$ " " " $A \to C_a B$

$A \to bAB$ " " " $A \to C_b AB$; $C_b \to b$

$V_N' = \{S, A, B, D, C_a, C_b\}$

STEP 3: P_1 consists of $S \to C_aAD$, $A \to C_aB \mid C_bAB$, $B \to b, D \to d$
$C_a \to a$, $C_b \to b$.

$A \to C_aB$, $B \to b$, $D \to d$, $C_a \to a$, $C_b \to b$ are added to P_2.

$S \to C_aAD$ is replaced by $S \to C_aC_1$; $C_1 \to AD$

$A \to C_bAB$ is replaced by $A \to C_bC_2$; $C_2 \to AB$

Let $G_2 = (\{S, A, B, D, C_a, C_b, C_1, C_2\}, \{a, b, d\}, P_2, S)$

where $P_2 = \{ S \to C_aC_1, C_1 \to AD, A \to C_bC_2 \mid C_aB, C_2 \to AB,$
$B \to b, D \to d, C_a \to a, C_b \to b\}$

G_2 is in CNF & is equivalent to G (i.e. $LCG_2) = LCG)$)

Ques:- Find a grammar in CNF equivalent to $S \to aAbB$,
$A \to aA \mid a$, $B \to bB \mid b$.

Soln:- As there are no null or unit productions we can proceed to step 2 directly.

STEP 2 :- $A \to a$, $B \to b$ are included in P_1. (Elimination of Terminals)

$A \to aA$ give rise to $A \to C_aA$; $C_a \to a$

$B \to bB$ give rise to $B \to C_bB$; $C_b \to b$

$S \to aAbB$ give rise to $S \to C_aAC_bB$

$V_N' = \{ S, A, B, C_a, C_b \}$

$P_1 = \{ A \to a, B \to b, A \to C_aA, C_a \to a, B \to C_bB, C_b \to b,$
$S \to C_aAC_bB \}$

STEP 3 :- $A \to a$, $B \to b$, $A \to C_aA$, $B \to C_bB$, $C_a \to a$, $C_b \to b$ are included in P_2. (Restriction on # of variables on RHS).

$S \to C_aAC_bB$ is replaced by $S \to C_aC_1$; $C_1 \to AC_2$; $C_2 \to C_bB$

$\therefore V_N'' = \{ S, A, B, C_a, C_b, C_1, C_2 \}$

$P_2 = \{ S \to C_aC_1 ; C_1 \to AC_2, C_2 \to C_bB, A \to C_aA, B \to C_bB, C_a \to a, C_b \to b,$
$A \to a, B \to b \}$

GREIBACH NORMAL FORM (GNF)

A CFG generating the set accepted by a PDA is in GNF.

Defⁿ:- A CFG is in GNF if every production is of the form $A \to a\alpha$, where $\alpha \in V_N^*$ and $a \in \Sigma$ (α may be Λ), and $S \to \Lambda$ is in G if $\Lambda \in L(G)$. When $\Lambda \in L(G)$, we assume that S does not appear on RHS of any production.

NOTE: A grammar in GNF is a natural generalization of a regular grammar.

LEMMA 1 :- Let $G = (V_N, \Sigma, P, S)$ be a CFG. Let $A \to B\gamma$ be an A-production in P. Let the B-productions be $B \to \beta_1 | \beta_2 | \cdots | \beta_s$.

Define

$$P_1 = (P - \{A \to B\gamma\}) \cup \{A \to \beta_i\gamma \mid 1 \le i \le s\}$$

Then, $G_1 = (V_N, \Sigma, P_1, S)$ is a CFG equivalent to G.

(TO REMOVE AMBIGUITY WHILE BACKTRACKING) (Compiler Design)

PROOF :- If we apply $A \to B\gamma$ in some derivation for $\omega \in L(G)$ we have to apply $B \to \beta_i$ for some i at a later step. So, $A \overset{*}{\underset{G}{\Rightarrow}} \beta_i\gamma$. The effect of applying $A \to B\gamma$ & eliminating B in grammar G is the same as applying $A \to \beta_i\gamma$ for some i in grammar G_1. Hence, $\omega \in L(G_1)$ i.e. $L(G) \subseteq L(G_1)$. Similarly, instead of applying $A \to \beta_i\gamma$, we can apply $A \to B\gamma$ & $B \to \beta_i$ to get $A \overset{*}{\underset{G}{\Rightarrow}} \beta_i\gamma$. This proves $L(G_1) \subseteq L(G)$.

example $P = \{A \to Bab, \; B \to aA | bB | aa | AB\}$

$P_1 = \{A \to aAab | bBab | aaab | ABab\}$

Lemma 2: Let $G = (V_N, \Sigma, P, S)$ be a CFG. Let the set of A-productions be $A \to A\alpha_1 | \ldots A\alpha_r | \beta_1 | \beta_2 \cdots | \beta_s$ (β_i's do not start with A). Let Z be a new variable. Let $G_1 = (V_N \cup \{Z\}, \Sigma, P_1, S)$ where P_1 is defined as follows:

(i) The set of A-productions in P_1 are $A \to \beta_1 | \beta_2 | \ldots | \beta_s$

$$A \to \beta_1 Z | \beta_2 Z | \ldots | \beta_s Z$$

(ii) The set of Z-productions in P_1 are $Z \to \alpha_1 | \alpha_2 | \ldots | \alpha_r$

$$Z \to \alpha_1 Z | \alpha_2 Z | \ldots | \alpha_r Z$$

(iii) The production for other variables are as in P. Then G_1 is a CFG & equivalent to G.

(TO REMOVE LEFT-RECURSION : IT LEADS TO INFINITE LOOP WHILE PROGRAMMING

NEW: $Z \to \alpha_i Z$ ↓ cond^{n}s checked before

OLD: $A \to A\alpha$ ↓ Block of statements → cond^{n}s cond^{n}s checked after.

PROOF:- To prove $L(G) \subseteq L(G_1)$; consider a leftmost derivation of w in G. The only productions in $P - P_1$ are $A \to A\alpha_1 | A\alpha_2 \cdots | A\alpha_r$. If $A \to A\alpha_{i1}, A \to A\alpha_{i2}, \ldots, A \to A\alpha_{ik}$ are used, then $A \to \beta_j$ should be used at a later stage (to eliminate A). So, we have $A \overset{*}{\underset{G_1}{\Rightarrow}} \beta_j \alpha_{i1} \cdots \alpha_{ik}$ while deriving w in G. However,

$$A \underset{G_1}{\Rightarrow} \beta_j Z \underset{G_1}{\Rightarrow} \beta_j \alpha_{i1} Z \cdots \underset{G_1}{\Rightarrow} \beta_j \alpha_{i1} \alpha_{i2} \cdots \alpha_{ik}$$

i.e. $A \overset{*}{\underset{G}{\Rightarrow}} \beta_j \alpha_{i1} \alpha_{i2} \cdots \alpha_{ik}$

Thus, A can be eliminated by using productions in G_1.

$\therefore w \in L(G_1)$.

To prove $L(G_1) \subseteq L(G)$, consider a leftmost derivation of ω in G_1. The only productions in $P_1 - P$ are

$A \to \beta_1 Z \mid \beta_2 Z \mid \cdots \mid \beta_s Z$, $Z \to \alpha_1 \mid \cdots \mid \alpha_r$, $Z \to \alpha_1 Z \mid \alpha_2 Z \mid \cdots \mid \alpha_r Z$.

If the new variable Z appears in the course of the derivation of ω, it is because of the application of $A \to \beta_j Z$ in some earlier step. Also Z can be eliminated only by a production of the form $Z \to \alpha_j$ or $Z \to \alpha_j Z$ for some i and j in a later step. So, we get $A \overset{*}{\underset{G_1}{\Rightarrow}} \beta_j \alpha_{i_1} \alpha_{i_2} \cdots \alpha_{i_k}$ in the course of derivation of ω. But we know that

$A \overset{*}{\underset{G}{\Rightarrow}} \beta_j \alpha_{i_1} \alpha_{i_2} \cdots \alpha_{i_k}$, $\therefore \omega \in L(G)$.

Example :- Remove left-Recursion in the grammar below :-

$A \to aBD \mid bDB \mid C$, $A \to AB \mid AD$

New productions are :-

i) $A \to aBD \mid bDB \mid C$; $A \to aBDZ \mid bDBZ \mid CZ$

ii) $Z \to B$, $Z \to D$; $Z \to BZ \mid DZ$

Theorem : Reduction to GNF

Every CFL L can be generated by a CFG G in GNF.

PROOF :- Case : Construction of G (when $\Lambda \in L$)

STEP 1 : We eliminate null productions & then construct a grammar G in CNF generating L. We rename the variables as $A_1, A_2, \ldots, A_n$ with $S = A_1$. We write G as

$(\{A_1, A_2, \ldots, A_n\}, \Sigma, P, A_1)$

STEP 2 : To get productions in the form $A_i \to \alpha\gamma$ or $A_i \to A_j\gamma$, where $j > i$, convert the A_i - productions $(i = 1, 2, \ldots, n-1)$

to the form $A_i \rightarrow A_j\gamma$ s.t. $j > i$. Prove such modification is possible by induction on i.

Consider A_1-productions. If we have some A_1 productions of the form $A_1 \rightarrow A_1\gamma$, then we apply <u>lemma 2</u> to get rid of such productions. We get a new variable, say Z_1, and A_1-productions of the form $A_1 \rightarrow a$ or $A_1 \rightarrow A_j\gamma'$, where $j > 1$. Thus, there is a basis for induction.

Assume that we have modified A_1-productions, A_2-productions, ..., A_i-productions.

<u>Ques</u>:- Construct a grammar in GNF equivalent to the grammar $S \rightarrow AA|a$; $A \rightarrow SS|b$.

<u>Soln</u> <u>step 1</u>:- The given grammar is in CNF. Rename S & A as A_1 & A_2 respectively, we get (CNF & Rename) ①

$$A_1 \rightarrow A_2A_2|a \qquad A_2 \rightarrow A_1A_1|b$$

As given grammar is in CNF & has no null production, Proceed to step 2.

<u>Step 2</u>:- i) A_1 productions are in required form (As $j > i$, $A_i \rightarrow A_j\gamma$)

They are $A_1 \rightarrow A_2A_2|a$

ii) $A_2 \rightarrow b$ is in required form. Apply lemma 1 to $A_2 \rightarrow A_1A_1$; we get ② ($A_i \rightarrow A_j\gamma$, $j > i$)

$$A_2 \rightarrow A_2A_2A_1|aA_1$$

Thus, A_2-productions are:-

$$A_2 \rightarrow A_2A_2A_1 \quad , \quad A_2 \rightarrow aA_1 \quad , \quad A_2 \rightarrow b$$

③ (if $j = i$)

<u>Step 3</u>:- In $A_2 \rightarrow A_2A_2A_1$, we need to apply Lemma 2

due to left recursion (as $j \neq i$). Let Z_2 be the new variable. ∴ The resulting productions are :-

$A_2 \to aA_1 \; ; \; A_2 \to b$

$A_2 \to aA_1Z_2 \; ; \; A_2 \to bZ_2$

$Z_2 \to A_2A_1 \; ; \; Z_2 \to A_2A_1Z_2$

④ (A_i - productions to GNF)

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

Step 4 :- i) The A_2 - productions are $A_2 \to aA_1 \mid b \mid aA_1Z_2 \mid bZ_2$

ii) Among the A_1 - productions, we retain $A_1 \to a$ & eliminate $A_1 \to A_2A_2$ using Lemma 1.

The set of all (modified) A_1 - productions are :-

$A_1 \to a \mid aA_1A_2 \mid bA_2 \mid aA_1Z_2A_2 \mid bZ_2A_2$

(∵ in GNF $A \to a\alpha$ TYPE)

Step 5 : The Z_2 - productions to be modified are :-

$Z_2 \to A_2A_1 \; ; \; Z_2 \to A_2A_1Z_2$. Apply lemma 1 & get

⑤ (Z_i - productions to GNF)

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

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

Hence, the equivalent grammar is :-

$G' = (\{A_1, A_2, Z_2\}, \Sigma, P_1, A_1)$

where P_1 consists of

$A_1 \to a \mid aA_1A_2 \mid bA_2 \mid aA_1Z_2A_2 \mid bZ_2A_2$

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

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

$Z_2 \to aA_1A_1Z_2 \mid bA_1Z_2 \mid aA_1Z_2A_1Z_2 \mid bZ_2A_1Z_2$.

Ques :- Convert the grammar $S \to AB$, $A \to BS \mid b$, $B \to SA \mid a$ into GNF.

Sol^n :- Step 1 :- As given grammar is in CNF; proceed to Step 2. after renaming S, A & B by A_1, A_2 & A_3 respectively.

$A_1 \to A_2A_3$; $A_2 \to A_3A_1 \mid b$; $A_3 \to A_1A_2 \mid a$

Step 2 :- (i) The A_1- productions & A_2 productions are in required form. $A_3 \to a$ is in required form.

(ii) $A_3 \to A_1A_2$ is modified (as $j<i$) using lemma 1

$A_3 \to A_2A_3A_2$

Again applying lemma 1 (as $j<i$)

Step 3 :

$\Rightarrow A_3 \to A_3A_1A_3A_2 \mid bA_3A_2$

$A_3 \to bA_3A_2$ is in required form. Use lemma 2 for $A_3 \to A_3A_1A_3A_2$ to remove left-recursion. let new variable is Z_3

$A_3 \to a \mid bA_3A_2$; $A_3 \to aZ_3$; $A_3 \to bA_3A_2Z_3$

$Z_3 \to A_1A_3A_2$ $Z_3 \to A_1A_3A_2Z_3$

Step 4 : (i) The A_3- productions are :-

$A_3 \to a \mid bA_3A_2 \mid aZ_3 \mid bA_3A_2Z_3$

(ii) Among A_2-production, retain $A_2 \to b$ & modify $A_2 \to A_3A_1$ to required GNF form. The new A_2-productions are :-

$A_2 \to b \mid aA_1 \mid bA_3A_2A_1 \mid aZ_3A_1 \mid bA_3A_2Z_3A_1$

(iii) Apply lemma 1 to A_1-production to get

$A_1 \to bA_3 \mid aA_1A_3 \mid bA_3A_2A_1A_3 \mid aZ_3A_1A_3 \mid bA_3A_2Z_3A_1A_3$

Step 5 :- The Z_3-productions to be modified are

$$Z_3 \to A_1A_3A_2 \; ; \; Z_3 \to A_1A_3A_2Z_3$$

Apply Lemma 1 & get :-

$$Z_3 \to bA_3A_3A_2 \mid aA_1A_3A_3A_2 \mid bA_3A_2A_1A_3A_3A_3 \mid aZ_3A_1A_3A_3A_2 \mid bA_3A_2Z_3A_1A_3A_3A_2$$

$$Z_3 \to bA_3A_3A_2Z_3 \mid aA_1A_3A_3A_2Z_3 \mid bA_3A_2A_1A_3A_3A_3Z_3 \mid aZ_3A_1A_3A_3A_2Z_3 \mid bA_3A_2Z_3A_1A_3A_3A_2Z_3 .$$

The required grammar is in GNF.

PUMPING LEMMA FOR CFL

The Pumping Lemma for CFL gives a method of generating an infinite # of strings from a given sufficiently long string in a CFL L. It is used to prove that certain languages are not context-free.

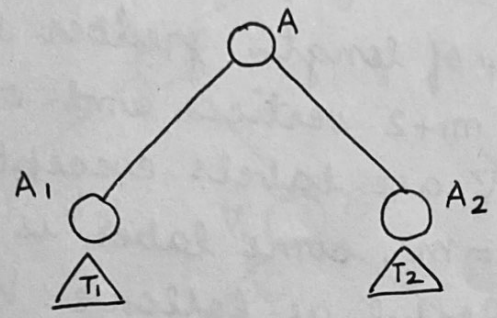
Lemma : Let G be a CFG in CNF & T be a derivation tree in G. If the length of the longest path in T is less than or equal to k, then the yield of T is of length less than or equal to 2^{k-1}.

Proof :- We prove the result by induction on k, the length of the longest path for all A-trees (Recall A-tree is a derivation tree whose root has label A).

When the longest path in an A-tree is of length 1, the root has only one son whose label is a terminal (when the root has 2 sons, the labels are variables). (∵ in CNF, productions form $A \to AB$ or $A \to a$)

So, the yield is of length 1. Thus, there is a basis for induction.

Assume the result for k-1 (k>1). let T be an A-tree with a longest path of length less than or equal to k. As k > 1, the root of T has exactly two sons with labels A_1 & A_2. The two subtrees with these sons as roots have the longest paths of length less than or equal to (k-1).



Tree T with subtrees T_1 and T_2.

If w_1 and w_2 are their yields, then by induction hypothesis, $|w_1| \leq 2^{k-2}$, $|w_2| \leq 2^{k-2}$. So, the yield of $T = w_1 w_2$

$$|w_1 w_2| = |w_1| + |w_2| \leq 2^{k-2} + 2^{k-2}$$
$$= 2^{k-2} \cdot 2 = 2^{k-1}$$

By principle of induction, the result is true ∀ A-trees, & hence for all derivation trees.

THEOREM: (PUMPING LEMMA FOR CONTEXT-FREE LANGUAGES)

Let L be a CFL. Then, we can find a natural number n such that:

(i) Every $z \in L$ with $|z| \geq n$ can be written as $uvwxy$ for some strings u, v, w, x and y.

(ii) $|vx| \geq 1$.

(iii) $|vwx| \leq n$

(iv) $uv^k wx^k y \in L$ for all $k \geq 0$.

PROOF:- We decide whether or not $\Lambda \in L$. When $\Lambda \in L$, we consider $L - \{\Lambda\}$ & construct a grammar $G = (V_N, \Sigma, P, S)$

in CNF generating $L - \{\wedge\}$ (when $\wedge \notin L$, we constru... G in CNF generating L).

Let $|V_N| = m$ and $n = 2^m$. To prove that n is the required number we start with $z \in L$, $|z| \geq 2^m$, and construct a derivation tree T (parse tree) of z. If the length of the longest path in T is atmost m, by previous lemma, $|z| \leq 2^{m-1}$ ($\because$ z is the yield of T). But $|z| \geq 2^m > 2^{m-1}$. So, T has a path, say Γ, of length greater than or equal to (m+1). Γ has at least m+2 vertices and only the last vertex is a leaf. Thus in Γ all labels except the last one are variables. As $|V_N| = m$, some label is repeated.

We choose a repeated label as follows: We start with the leaf of Γ and travel along Γ upwards. We stop when some label, say B, is repeated. (Among several repeated labels, B is first). Let v_1 and v_2 be the vertices with label B, v_1 being nearer the root. In Γ, the portion of path from v_1 to leaf has only one label, namely B, which is repeated, and so its length is atmost (m+1).

Let T_1 and T_2 be the subtrees with v_1 and v_2 as roots and z_1, w as yields, respectively. As Γ is the longest path in T, the portion of Γ from v_1 to the leaf is a longest path in T_1 & of length at most (m+1). By previous lemma, $|z_1| \leq 2^m$ ($\because$ z_1 is the yield of T_1).

As z and z_1 are the yields of T and a proper subtree T_1 of T, we can write $z = uz_1y$. As z_1 and w are the yields of T_1 & a proper subtree T_2 of T_1, we can write $z_1 = vwx$. Also $|vwx| > |w|$. So $|vx| \geq 1$. Thus, we have $z = uvwxy$.

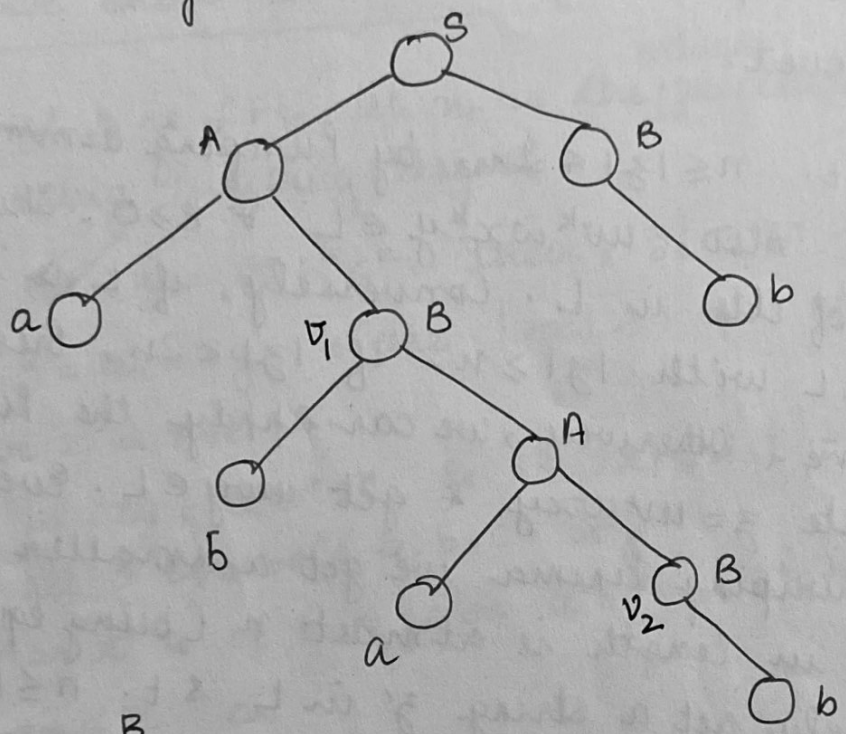
As T is an S-tree and T_1, T_2 are B-trees, we get $S \overset{*}{\Rightarrow} uBy$, $B \overset{*}{\Rightarrow} vBx$ and $B \overset{*}{\Rightarrow} w$. As $S \overset{*}{\Rightarrow} uBy \Rightarrow uwy$, $uv^0wx^0y \in L$. For $k \geq 1$, $S \overset{*}{\Rightarrow} uBy \overset{*}{\Rightarrow} uv^kBx^ky \overset{*}{\Rightarrow} uv^kwx^ky \in L$. This proves (iv):

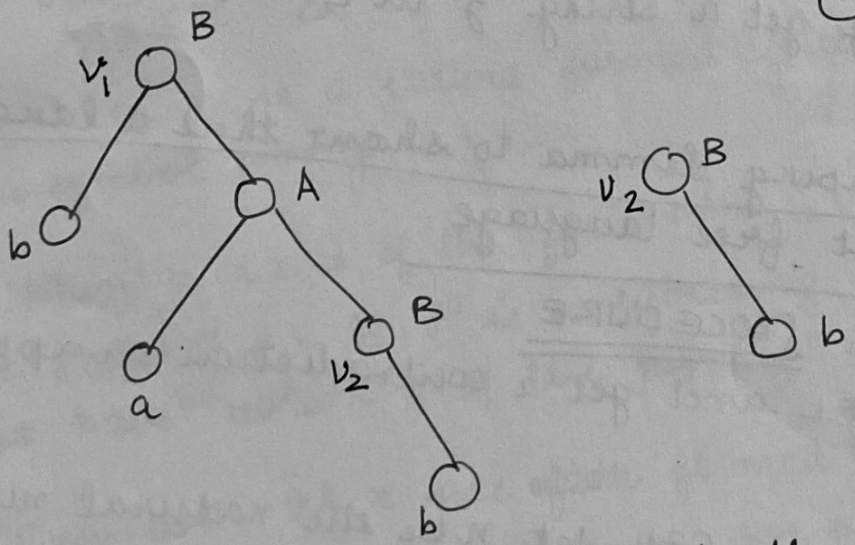
EXAMPLE :- $S \to AB,\ A \to aB \mid a,\ B \to bA \mid b$

Here, $\nabla = S \to A \to B \to A \to B \to b$

$z = ababbb \quad z_1 = bab \quad w = b \quad v = ba \quad x = \Lambda$

$u = a \quad y = b.$





Corollary: Let L be a CFL & n be the natural number obtained by using Pumping Lemma. Then, (i) $L \neq \phi$ iff $\exists\ w \in L$ with $|w| < n$, and (ii) L is infinite iff $\exists\ z \in L$ s.t. $n \leq |z| < 2n$.

Proof :-

(i) We have to prove the 'only if' part. If $z \in L$ with $|z| \geq n$, we apply Pumping Lemma to write $z = uvwxy$, where $1 \leq |vx| \leq n$. Also, $uwy \in L$ & $|uwy| < |z|$. Applying the Pumping Lemma repeatedly, we can get $z' \in L$ s.t. $|z'| < n$. Thus, (i) is proved.

(ii) If $z \in L$ s.t. $n \leq |z| < 2n$, by Pumping Lemma we can write $z = uvwxy$. Also, $uv^k wx^k y \in L \ \forall \ k \geq 0$. Thus, we get an infinite # of elts. in L. Conversely, if L is infinite, we can find $z \in L$ with $|z| \geq n$. If $|z| < 2n$, there is nothing to prove. Otherwise, we can apply the Pumping Lemma to write $z = uvwxy$ & get $uwy \in L$. Every time we apply the Pumping Lemma we get a smaller string & the decrease in length is atmost n (being equal to $|vx|$). So, we ultimately get a string z' in L s.t. $n \leq |z'| < 2n$. This proves (ii).

NOTE :- We use Pumping Lemma to show that a language L is not context free language.

PROCEDURE

Assume L is CFL and get a contradiction on applying Pumping Lemma.

STEP 1 : Assume L is CFL. Let n be the natural number obtained by using Pumping Lemma.

STEP 2 : Choose $z \in L$ so that $|z| \geq n$. Write $z = uvwxy$ using the pumping lemma.

STEP 3 : Find a suitable k so that $uv^k wx^k y \notin L$. This is a contradiction & so L is not context free.

Ex 3/Imp :- Show that $L = \{a^n b^n c^n \mid n \geq 1\}$ is not context-free but context-sensitive.

Soln :- To show it is context-sensitive construct a grammar G which is context-sensitive and $L(G) = L$.

Next, we show L is not CFL

STEP 1 :- Let L is CFL. Let n be the natural number obtained by using Pumping lemma.

STEP 2 : Let $z = a^n b^n c^n$. Then, $|z| = 3n > n$. Write $z = uvwxy$, where $|vx| \geq 1$ i.e. at least one of v or x is not Λ.

STEP 3 : $uvwxy = a^n b^n c^n$. As $1 \leq |vx| \leq n$; v or x contains all 3 symbols a, b, c. So,

(i) v or x is of the form $a^i b^j$ (or $b^i c^j$) for some i & j s.t. $i+j \leq n$ or

(ii) v or x is a string formed by the repitition of only one symbol among a, b, c.

When v or x is of the form $a^i b^j$, $v^2 = a^i b^j a^i b^j$ (or $x^2 = a^i b^j a^i b^j$). As v^2 is a substring of uv^2wx^2y, we cannot have uv^2wx^2y of the form $a^m b^m c^m$. So, $uv^2wx^2y \notin L$.

When both v & x are formed by repitition of a single symbol. (e.g. $u = a^i$ & $v = b^j$ for some i & j $\exists$, $i \leq n$, $j \leq n$), the string uwy will contain the remaining symbol, say a_1. Also, a_1^n will be a substring of uwy as a_1 does not occur in v or x. The # of occurrences of one of the other two symbols in uwy is less than n (recall, $uvwxy = a^n b^n c^n$) & n is the # of occurrences of a_1. So, $uv^0wx^0y = uwy \notin L$

Thus, for any choice of v or x, we get a contradiction. $\therefore$ L is not CFL.

Ques Show that $L = \{a^p \mid p \text{ is prime}\}$ is not a CFL.

Sol^n: We use the property of L : If $w \in L$, then $|w|$ is prime.

STEP 1 :- Suppose $L = L(G)$ is CFL. Let n be the natural no. obtained by using pumping lemma.

STEP 2: Let p be a prime no. greater than n. Then, $z = a^p \in L$

We write $z = uvwxy$

STEP 3: By pumping lemma, $uv^0wx^0y = uwy \in L$. So, $|uwy|$ is a prime no. say q. Let $|vx| = r$. Then, $|uv^qwx^qy|$

$= q + qr$. As $q + qr$ is not prime, $uv^qwx^qy \notin L$. This is a contradiction. $\therefore$ L is not CFL.

DECISION ALGORITHMS FOR CFL

(i) Algorithm for deciding whether a CFL L is EMPTY

We can apply the construction in theorem :

If G is a CFG s.t. $L(G) \neq \phi$, we can find an equivalent grammar G' s.t. each variable in G' derives a terminal string.

for getting $V_N' = W_K$.

L is non-empty iff $S \in W_K$.

(ii) Algorithm for deciding whether a CFL L is finite

Construct a non-redundant CFG G in CNF generating $L - \{\wedge\}$. We draw a directed graph whose vertices are variables in G. If $A \to BC$ is a production, there are directed edges from A to B & A to C. L is finite iff the directed graph has no cycles.

(iii) Algorithm for deciding whether a regular language L is empty

Construct a DFA M accepting L. We construct the set of all states reachable from the initial state q_0. We find the states reachable from q_0 by applying a single input symbol. These states are arranged in a row - under columns corresponding to every input symbol. The construction is repeated for every state appearing in earlier row. The construction terminates in a finite # of steps. If a final state appears in this tabular - column, then $L \neq \phi$. (We can actually terminate the construction as soon as some final state is obtained in tabular column). Otherwise, $L = \phi$.

(iv) Algorithm for deciding whether a regular language L is infinite

Construct a DFA M accepting L. L is infinite iff M has a cycle.