Context free grammar (CFG):

Defn. A type 2 production is a production of the form $A \to \alpha$, where $A \in V_N$ and $\alpha \in (V_N \cup \Sigma)^*$. In other words, the LHS. has no left context or right context, e.g. $S \to Aa$, $A \to a$, $B \to abC$, $A \to A$ are type 2 produc

Defn. A grammar is called a type 2 grammar if it contains only type 2 productions. It is also called a context-free grammar (as A can be reduced by α in any context).

A language generated by context-free grammar is ca context-free language.

Derivation Trees. The derivations in a CFG can be repre using trees. Such trees are called derivation trees.

Defn. A derivation tree for a CFG $G = (V_N, \Sigma, P, S)$ is tree satisfying the following conditions:

(i) every vertex has a label which is a variable or terminal o

(ii) The root has label S.

(iii) The label of an internal vertex is a variable.

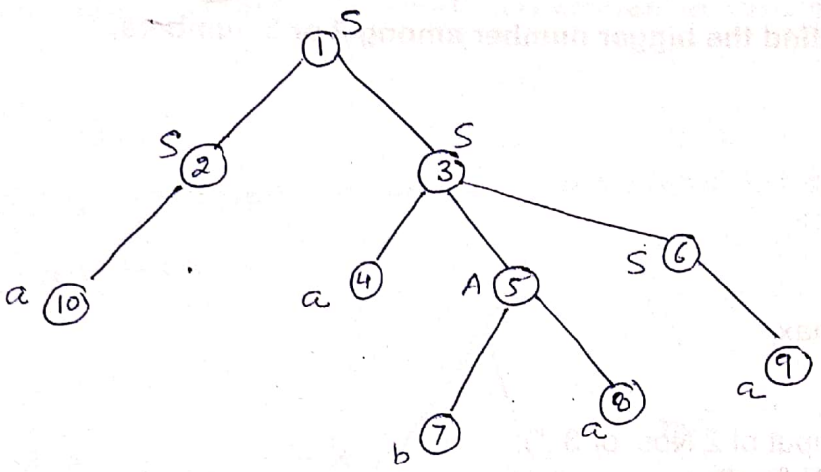
(iv) If the vertices $n_1, n_2, \cdots, n_k$ written with labels $X_1, X_2, \cdots$ are the sons of vertex n with label A, then $A \to X_1 X_2 \cdots$ is a production in P.

(v) A vertex n is a leaf if its label is $a \in \Sigma$ or Λ; n only son of its father if its label is Λ.

~~e.g. let $G = (\{S, A\}, \{a, b\}, P, S)$~~

~~S~~

Ex: Let $G = (\{S, A\}, \{a, b\}, P, S)$



$S \longrightarrow SS/a/aAS$, $A \longrightarrow ba$

Defn. The yield of a derivation tree is the concatanation of the labels of the leaves without repetition in the left-to-right-ordering.

e.g. the yield of the above derivation tree is aabaa.

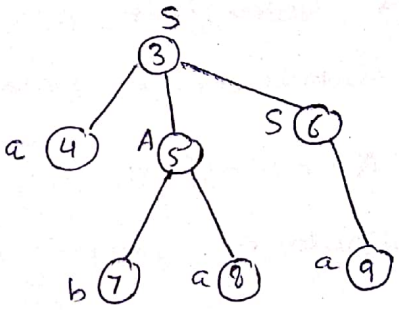
In the above tree,

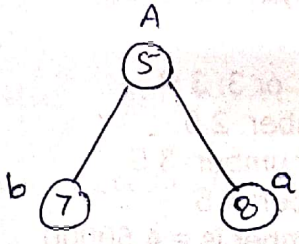
$$S \longrightarrow SS \longrightarrow aS \longrightarrow aaAS \longrightarrow aabaS \longrightarrow aabaa$$

Thus the yield of the derivation tree is a sentential form in G.

Defn. A subtree of a derivation tree T is a tree (i) whose root is some vertex v of T, (ii) whose vertices are the descendants of v together with their labels, and (iii) whose edges are those connecting the descendants of v.

e.g.





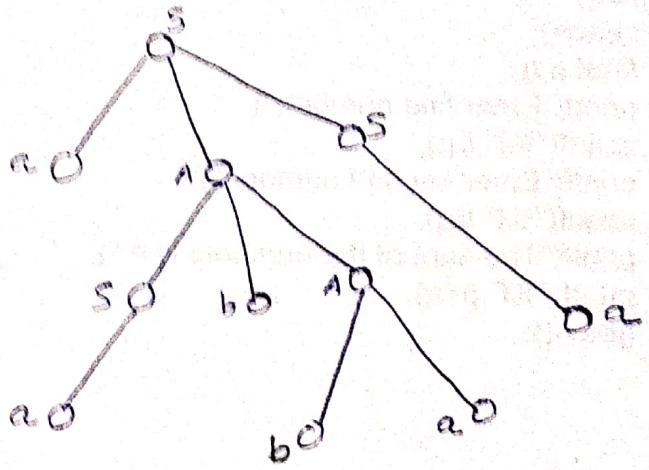
are the subtre
of the above tr

Note: A subtree look like a D-tree except that the label of the root may not be S. It is called an A-tree if the label of its root is A.

Ex: Consider a Grammar productions are $S \to aAS / a$, $A \to SbA / SS / ba$. Show that $S \Rightarrow^* aabbaa$ and construct a D-Tree whose yield is aabbaa.

Soln: $S \to aAS \to aSbAS \to aabAS \to aabbaS \to aabbaa$. (i)

(i) $S \Rightarrow^* aabbaa$.



Also we can write

$$S \to aAS \xrightarrow{S \to a} aAa \xrightarrow{A \to SbA} aSbAa \xrightarrow{A \to ba} aSbbaa \xrightarrow{S \to a} aabbaa \quad \text{(ii)}$$

Also, $S \to aAS \xrightarrow{A \to SbA} aSbAS \xrightarrow{S \to a} aSbAa \xrightarrow{S \to a} aabAa \xrightarrow{A \to ba} aabbaa$ (iii)

In derivation (i), whenever we replace a variable X using a production, there are no variables to the left of X, In (ii), there are no variables to the right of X. But in (iii), no such conditions are satisfied.

Defn. A derivation $A \Rightarrow^* w$ is called a 'leftmost' derivation if we apply a production only to the leftmost variable at every step.

Defn. A derivation $A \Rightarrow^* w$ is called a 'rightmost' derivation if we apply production to the rightmost variable at every step.

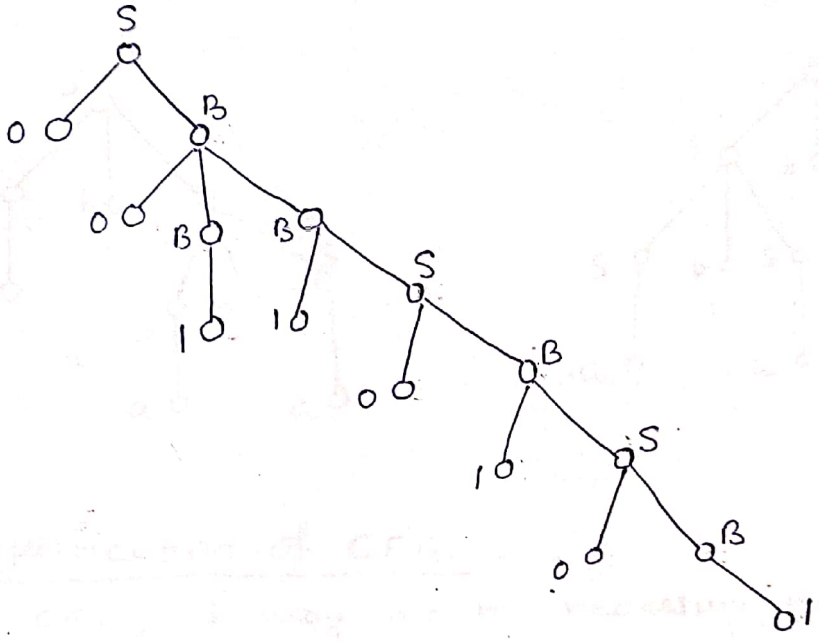
Ex. Let G be the grammar $S \to 0B / 1A$, $A \to 0 / 0S / 1AA$, $B \to 1 / 1S / 0BB$. For the string 00110101, find (a) the leftmost derivation, (b) the rightmost derivation, and (c) the D-tree.

Soln.

(a) $S \xrightarrow{B\to 0BB} 0B \to 00BB \xrightarrow{B\to 1} 001B \xrightarrow{B\to 1S} 0011S \xrightarrow{S\to 0B} 00110B \xrightarrow{B\to 1S} 00110S1 \xrightarrow{S\to 0B} 001101B1 \xrightarrow{B\to 1} 00110101$

(4)

(b) $S \xrightarrow{B\to 0BB} 0B \to 00BB \xrightarrow{B\to 1S} 00B1S \xrightarrow{S\to 0B} 00B10B \xrightarrow{B\to 1S} 00B101S \xrightarrow{S\to 0B} 00B1010B \xrightarrow{B\to 1} 00B10101 \xrightarrow{B\to 1} 00110101$



Ambiguity In CFG :

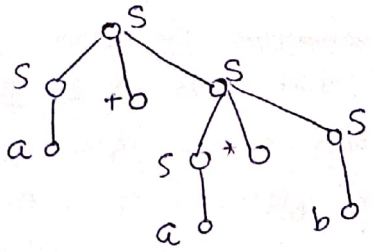
Defn. A terminal string $w \in L(G)$ is ambiguous if there exists two or more D-trees for w [or] two or more left most derivations of w).

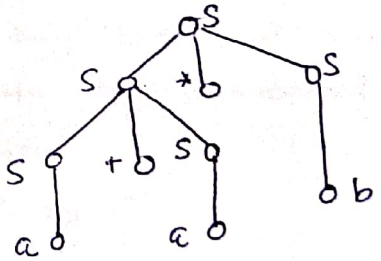
Ex. $G = (\{S\}, \{a, b, +, *\}, P, S)$, where P consists of-

$S \to S+S \mid S*S \mid a \mid b$. We have two D-trees for $a+a*b$

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

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





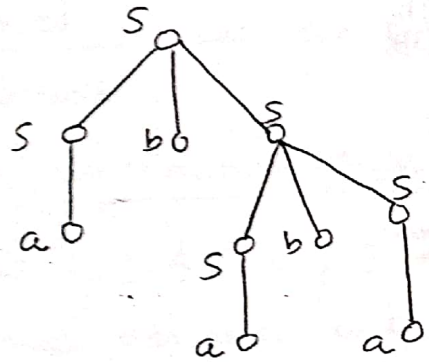
Defn. A CFG is ambiguous if $\exists$ some $w \in L(G)$ which is ambiguous.

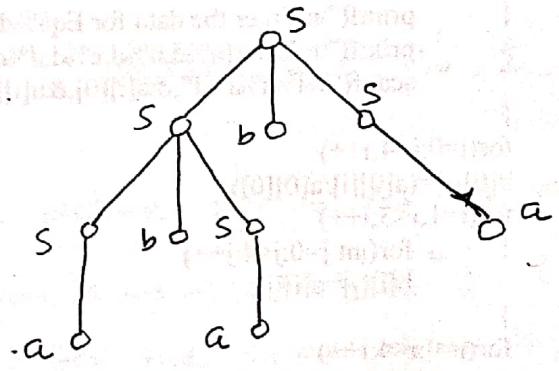
Ex If G is the grammar $S \to SbS / a$, show that G is ambiguous.

Soln: $S \xrightarrow{S\to SbS} SbS \xrightarrow{S\to a} abS \xrightarrow{S\to SbS} abSbS \xrightarrow{S\to a} ababS \xrightarrow{S\to a} ababa$

$S \xrightarrow{S\to SbS} SbSbS \xrightarrow{S\to a} abSbS \xrightarrow{S\to a} ababS \xrightarrow{S\to a} ababa$

$\therefore$ $w = ababa \in L(G)$ which is ambiguous. Thus G is ambiguous.





Simplification of CFG.

In a CFG, it may not be necessary to use all the symbols in $V_N \cup \Sigma$ or all the productions in P for deriving sentences. we may try to eliminate those symbols and productions in G, which are not useful for the derivation of sentences.

e.g. $G = (\{S, A, B, C, E\}, \{a, b, c\}, P, S)$, where

$P = \{S \to AB, A \to a, B \to b, B \to C, E \to c / \Lambda\}$.

It is easy to see that $L(G) = \{ab\}$. Let $G' = (\{S, A, B\}, \{a, b\}, P', S)$ where $P' = \{S \to AB, A \to a, B \to b\}$, $L(G) = L(G')$. we have eliminated the symbols C, E & C and the productions $B \to C$, $E \to c/\Lambda$. we note the following pts.

(i) C does not ~~appear~~ derive any terminal strings
(ii) E & C does not appear in any sentential form
(iii) $E \to \Lambda$ is a null production
(iv) $B \to C$ simply replaces B by C.

We give the construction to eliminate (i) variables not deriving terminal strings, (ii) symbols not appearing in any sentential form, (iii) null productions, and (iv) productions of the form $A \to B$.

Construction of Reduced Grammars

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

Proof. Let $G = (V_N, \Sigma, P, S)$. Define $G' = (V_N', \Sigma, P', S)$ as follows:

(a) **construction of V_N'** – we define $W_i \subseteq V_N$ by recursion:

$W_1 = \{ A \in V_N \mid \exists$ a production $A \to w, w \in \Sigma^* \}$. (If $W_1 = \phi$, some variable will remain after the application of any production, and so $L(G) = \phi$).

$W_{i+1} = W_i \cup \{ A \in V_N \mid \exists$ some production $A \to \alpha, \alpha \in (\Sigma \cup W_i)^* \}$

By defn. of W_i, $W_i \subseteq W_{i+1}$ $\forall i$. As V_N is finite, $W_K = W_{K+1}$ for some $K \leq |V_N|$. Therefore, $W_K = W_{K+j}$ for $j \geq 1$. We define $V_N' = W_K$.

(b) **construction of P':** $P' = \{ A \to \alpha \mid A, \alpha \in (V_N' \cup \Sigma)^*$ where $\}$

we can define $G' = (V_N', \Sigma, P', S)$. $S \in V_N'$. (If $S \notin V_N'$, $L(G') = \phi$)

Now we are going to prove that every variable in V_N' derives some terminal string. So, we prove:

(i) If $A \in V_N'$, then $A \overset{*}{\underset{G'}{\Rightarrow}} w$ for some $w \in \Sigma^*$; conversely, if $A \overset{*}{\underset{G'}{\Rightarrow}} w$, then $A \in V_N'$.

(ii) $L(G') = L(G)$.

To prove (i) we note that $W_K = W_1 \cup W_2 \cup \cdots \cup W_K$. We prove by induction on i that for $i = 1, 2, \cdots, k$, $A \in W_i$ implies $A \overset{*}{\Rightarrow} w$ for some $w \in \Sigma^*$

If $A \in W_1$, then $A \underset{G}{\Rightarrow} w$. So $A \to w$ is in P'.
i.e. $A \underset{G'}{\overset{*}{\Rightarrow}} w$. Thus there is a basis for induction. Let us assume the result for i. Let $A \in W_{i+1}$. Then either $A \in W_i$, in which case, $A \underset{G'}{\overset{*}{\Rightarrow}} w$ for some $w \in \Sigma^*$ by induction hypothesis, or $\exists$ a production $A \to \alpha$, $\alpha \in (\Sigma \cup W_i)^*$. By defn. of P', $A \to \alpha$ is in P'. We can write $\alpha = X_1 X_2 \cdots X_m$ where $X_j \in \Sigma \cup W_i$. If $X_j \in W_i$ by induction hypothesis, $X_j \underset{G'}{\overset{*}{\Rightarrow}} w_j$ for some $w_j \in \Sigma^*$. So, $A \underset{G'}{\overset{*}{\Rightarrow}} w_1 w_2 \cdots w_m \in \Sigma^*$ (when X_j is terminal, $w_j = X_j$). By induction result is true for $i = 1, 2, \cdots, k$.

The converse part can be proved in a similar way by induction on the no. of steps in the derivation $A \underset{G}{\overset{*}{\Rightarrow}} w$.

We see immediately that $L(G') \subseteq L(G)$, as $V_N' \subseteq V_N$ and $P' \subseteq P$. To prove $L(G) \subseteq L(G')$, we need an auxillary result $A \underset{G'}{\overset{*}{\Rightarrow}} w$ if $A \underset{G}{\overset{*}{\Rightarrow}} w$ for some $w \in \Sigma^*$.

We prove this result by induction on the no. of steps in the derivation $A \underset{G}{\overset{*}{\Rightarrow}} w$.

If $A \underset{G}{\Rightarrow} w$, then $A \to w$ is in P and $A \in W_1 \subseteq V_N'$. As $A \in V_N'$ & $w \in \Sigma^*$, $A \to w$ is in P'. So $A \underset{G'}{\Rightarrow} w$, and there is a basis for induction. Assume the result for derivations in at most k steps. Let $A \underset{G}{\overset{k+1}{\Rightarrow}} w$. We can split this as $A \underset{G}{\Rightarrow} X_1 X_2 \cdots X_m \underset{G}{\overset{*}{\Rightarrow}} w_1 w_2 \cdots w_m$ s.t. $X_j \underset{G}{\overset{*}{\Rightarrow}} w_j$. If $X_j \in \Sigma$ then $w_j = X_j$.

If $X_j \in V_N$ then by (i), $X_j \in V_N'$. As $X_j \underset{G}{\overset{*}{\Rightarrow}} w_j$ in at most k steps, $X_j \underset{G'}{\overset{*}{\Rightarrow}} w_j$. Also $X_1, X_2, \cdots, X_m \in (\Sigma \cup V_N')^*$ implies that $A \to X_1 X_2 \cdots X_m$ is in P'. Thus $A \underset{G'}{\Rightarrow} X_1 X_2 \cdots X_m \underset{G'}{\overset{*}{\Rightarrow}} w_1 w_2 \cdots w_m$. Hence result is true for all derivations. In particular, $S \underset{G}{\overset{*}{\Rightarrow}} w$ implies $S \underset{G'}{\overset{*}{\Rightarrow}} w$.

$\therefore L(G) \subseteq L(G')$ and (ii) is completely proved.

Theorem For every CFG $G = (V_N, \Sigma, P, S)$, we can construct an equivalent grammar $G' = (V_N', \Sigma', P', S)$ s.t. every symbol in $V_N' \cup \Sigma'$ appears in some sentential form (i.e. $\forall X \in (V_N' \cup \Sigma')$, $\exists \alpha$ s.t. $S \overset{*}{\underset{G'}{\Rightarrow}} \alpha$ and X is a symbol in the string α).

We will not give complete proof but only construction of W_i, V_N', Σ' & P'.

construct $G' = (V_N', \Sigma', P', S)$ as follows:

(a) construction of W_i for $i \geq 1$;

(i) $W_1 = \{S\}$.

(ii) $W_{i+1} = W_i \cup \{ X \in V_N \cup \Sigma \mid \exists$ a production $A \to \alpha$ with $A \in W_i$ and α containing the symbol $X \}$.

$\therefore W_i \subseteq (V_N \cup \Sigma)$ & $W_i \subseteq W_{i+1}$. As $(V_N \cup \Sigma)$ is finite, $W_K = W_{K+1}$ for some K. $\therefore W_K = W_{K+j}$ $\forall j \geq 0$.

Let $X \in (V_N' \cup \Sigma') = W_K$. prove by induction on i. i.e. $X \in W_1 = \{S\}$ then it appears in some sentential form. Let result is true for i. Let $X \in W_{i+1}$ $\Rightarrow$ either $X \in W_i$ for which result is true or $\exists A \to \alpha$ with $A \in W_i$ $\therefore$ A appears in sent. form, hence result.

(b) construction of V_N', Σ' and P';

Define $V_N' = V_N \cap W_K$, $\Sigma' = \Sigma \cap W_K$

$P' = \{ A \to \alpha : A \in W_K \}$.

Defn. A grammar $G = (V_N, \Sigma, P, S)$ is said to be reduced or non-redundant if every symbol in $(V_N \cup \Sigma)$ appears in the course of the derivation of some terminal string, i.e. $\forall X \in (V_N \cup \Sigma)$, $\exists$ a derivation $S \overset{*}{\underset{G}{\Rightarrow}} \alpha X \beta \overset{*}{\Rightarrow} w \in L(G)$.

Theorem For every CFG G, $\exists$ a reduced grammar G' which is equivalent to G.

(For proof, follow the two previous theorems in order i.e. thm 1 and then thm 2).

Ex. Let $G = (V_N, \Sigma, P, S)$ be given by the productions $S \to AB$, $A \to a$, $B \to b$, $B \to C$, $E \to c$. Find G' s.t. every variable in G' derives some terminal string.

Soln. (a) construction of V_N':

$W_1 = \{A, B, E\}$.

$W_2 = W_1 \cup \{A_1 \in V_N : A_1 \to \alpha \text{ for some } \alpha \in (\Sigma \cup \{A, B, E\})^*\}$

$= W_1 \cup \{S\} = \{A, B, E, S\}$.

$W_3 = W_2 \cup \{A_1 \in V_N : A_1 \to \alpha \text{ for some } \alpha \in (\Sigma \cup \{S, A, B, E\})^*\}$

$= W_2 \cup \phi = W_2$.

$\therefore V_N' = \{S, A, B, E\}$.

(b) construction of P': $P' = \{A_1 \to \alpha : \alpha \in (V_N' \cup \Sigma)^*\}$

$= \{S \to AB, A \to a, B \to b, E \to c\}$

$\therefore G' = (\{S, A, B, E\}, \{a, b, c\}, P', S)$.

Ex. consider $G = (\{S, A, B, E\}, \{a, b, c\}, P, S)$, where P consists of $S \to AB$, $A \to a$, $B \to b$, $E \to c$. construct an equivalent grammar $G' = (V_N', \Sigma', P', S)$ s.t. every symbol in $V_N' \cup \Sigma'$ appears in some sentential form.

Soln. $W_1 = \{S\}$, $W_2 = W_1 \cup \{X \in (V_N \cup \Sigma) : \exists\ A_1 \to \alpha$ with $A_1 \in W_1$ & α containing $X\}$

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

$W_3 = \{S, A, B\} \cup \{a, b\}$, $W_4 = W_3$

$\therefore V_N' = \{S, A, B\}$, $\Sigma' = \{a, b\}$, $P' = \{S \to AB, A \to a, B \to b\}$.

Ex. Find a reduced grammar equivalent to the grammar G whose productions are $S \to AB | CA$, $B \to BC | AB$, $A \to a$, $C \to aB | b$.

Soln. **Step1.** $W_1 = \{A, C\}$ as $A \to a$ & $C \to b$ are productions with a terminal string on RHS.

$W_2 = \{A, C\} \cup \{ A_1 : A_1 \to \alpha \text{ with } \alpha \in (\Sigma \cup \{A, C\})^* \}$

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

$W_3 = \{S, A, C\} \cup \{ A_1 : A_1 \to \alpha \text{ with } \alpha \in (\Sigma \cup \{S, A, C\})^* \}$

$= \{S, A, C\} = W_2$

$V_N' = W_2 = \{S, A, C\}$

$P' = \{ A_1 \to \alpha : A_1, \alpha \in (V_N' \cup \Sigma)^* \}$

$= \{ S \to CA, A \to a, C \to b \}$

Thus $G_1 = (\{S, A, C\}, \{a, b\}, \{S \to CA, A \to a, C \to b\}, S)$

Step2- we have to apply 2nd thm. Thus,

$W_1 = \{S\}$

as we have $S \to CA$ & $S \in W_1$, $W_2 = \{S\} \cup \{A, C\} = \{S, A, C\}$

as $A \to a$, $C \to b$ are productions with $A, C \in W_2$, $W_3 = \{S, A, C, a, b\}$

as $W_3 = V_N' \cup \Sigma$, $P'' = \{A_1 \to a : A_1 \in W_3\} = P'$

$\therefore$ $G' = (\{S, A, C\}, \{a, b\}, \{S \to CA, A \to a, C \to b\}, S)$ is the reduced grammar.

Elimination of Null productions.

A CFG may have productions of the form $A \to \Lambda$. This is just used to erase A. So a production of the form $A \to \Lambda$, where A is a variable, is called a null production.

e.g. $S \to aS \mid aA \mid \Lambda$, $A \to \Lambda$ has two null productions $S \to \Lambda$ and $A \to \Lambda$. We can delete $A \to \Lambda$ provided we erase A whenever it occurs in a derivation of a terminal string. So, we can replace $S \to aA$ by $S \to a$. If G_1 denotes the grammar with $S \to aS \mid a \mid \Lambda$ then $L(G_1) = L(G) = \{a^n \mid n \geq 0\}$. Thus it is possible to eliminate the null production $A \to \Lambda$.

Defn. A variable in a CFG is nullable if $A \overset{*}{\Rightarrow} \Lambda$.

Theorem: If $G = (V_N, \Sigma, P, S)$ is a CFG, then we can find a CFG G_1 having no null productions s.t. $L(G_1) = L(G) - \{\Lambda\}$.

Proof: we construct $G_1 = (V_N, \Sigma, P', S)$ as follows:

Step1: construction of the set of nullable variables:

we find these variables recursively

(i) $W_1 = \{A \in V_N : A \to \Lambda \text{ in } P\}$.

(ii) $W_{i+1} = W_i \cup \{A' \in V_N : \exists\ A' \to \alpha \text{ with } \alpha \in W_i^*\}$.

$\therefore W_i \subseteq W_{i+1}\ \forall i$. As V_N is finite, $W_{K+1} = W_K$ for some $K \le |V_N|$. So $W_{K+j} = W_K\ \forall j$. Let $W = W_K$. W is the set of nullable variables.

Step2: (i) construction of P': Any production whose RHS does not have any nullable variable is included in P'.

(ii) If $A \to X_1 X_2 \cdots X_K$ is in P, the productions of the form $A \to \alpha_1 \alpha_2 \cdots \alpha_K$ are included in P', where $\alpha_i = X_i$ if $X_i \notin W$, $\alpha_i = X_i$ or Λ if $X_i \in W$ and $\alpha_1 \alpha_2 \cdots \alpha_K \neq \Lambda$. Actually (ii) gives several productions in P'. The productions are obtained either by not erasing any nullable variable on the RHS of $A \to X_1 X_2 \cdots X_K$ or by erasing some or all nullable variables provided some symbol appears on the RHS after erasing.

Now we have to prove that $L(G_1) = L(G) - \{\Lambda\}$. we prove an auxillary result given by the following relation:

for all $A \in V_N$ and $\omega \in \Sigma^*$,

$$A \overset{*}{\underset{G_1}{\Rightarrow}} \omega \text{ iff } A \overset{*}{\underset{G}{\Rightarrow}} \omega \text{ and } \omega \neq \Lambda. \qquad \text{(iii)}$$

we prove the 'if' part first. Let $A \overset{*}{\underset{G}{\Rightarrow}} \omega$ and $\omega \neq \Lambda$. we prove $A \overset{*}{\underset{G_1}{\Rightarrow}} \omega$ by induction on the no. of steps in the derivation $A \overset{*}{\underset{G}{\Rightarrow}} \omega$. If $A \underset{G}{\Rightarrow} \omega$ & $\omega \neq \Lambda$, $A \to \omega$ is in P', so $A \underset{G_1}{\Rightarrow} \omega$. Assume the result is true for derivations in at most i steps. Let $A \overset{i+1}{\underset{G}{\Rightarrow}} \omega$ & $\omega \neq \Lambda$.

we can split the derivation as

$A \underset{G}{\Rightarrow} X_1 X_2 \cdots X_k \overset{i}{\underset{G}{\Rightarrow}} w_1 w_2 \cdots w_k$, where $w = w_1 w_2 \cdots w_k$.

and $A_j \overset{*}{\underset{G}{\Rightarrow}} w_j$. As $w \neq \Lambda$, not all w_j's are ~~empty~~ Λ.

If $w_j \neq \Lambda$, then by induction hypothesis $X_j \overset{i}{\underset{G}{\Rightarrow}} w_j$. If $w_j = \Lambda$, then $X_j \in w$. So using the production $A \to A_1 A_2 \cdots A_k$ in P, we construct $A \to \alpha_1 \alpha_2 \cdots \alpha_k$ in P', where $\alpha_j = X_j$ if $w_j \neq \Lambda$ and $\alpha_j = \Lambda$ if $w_j = \Lambda$ (i.e. $X_j \in w$). Therefore,

$$A \underset{G_1}{\Rightarrow} \alpha_1 \alpha_2 \cdots \alpha_k \overset{*}{\underset{G_1}{\Rightarrow}} w_1 \alpha_2 \cdots \alpha_k \overset{*}{\underset{G_1}{\Rightarrow}} w_1 w_2 \cdots w_k = w$$

By PMI, 'if' part is proved.

we prove the 'only if' part by PMI on the no. of steps in the derivation of $A \overset{*}{\underset{G_1}{\Rightarrow}} w$. If $A \overset{}{\underset{G_1}{\Rightarrow}} w$ then $A \to w$ is in P'.

$\therefore A \to w$ is obtained from some production $A \underset{G}{\Rightarrow} X_1 X_2 \cdots X_k \overset{*}{\underset{G}{\Rightarrow}} w$.

So, there is a basis for PMI. Assume the result for derivation in at most j steps. Let $A \overset{j+1}{\underset{G}{\Rightarrow}} w$. This can be split as $A \overset{}{\underset{G_1}{\Rightarrow}} X_1 X_2 \cdots X_k \overset{j}{\underset{G_1}{\Rightarrow}} w_1 w_2 \cdots w_k$, where $X_i \overset{*}{\underset{G}{\Rightarrow}} w_i$, $A \Rightarrow X_1 X_2 \cdots X_k$ in P' is obtained from some production $A \to \alpha$ in P by erasing some (or none of the) nullable variables in α. So $A \underset{G}{\Rightarrow} \alpha \overset{*}{\underset{G}{\Rightarrow}} X_1 X_2 \cdots X_k$. If $X_i \in \Sigma$ then $X_i \overset{0}{\underset{G}{\Rightarrow}} X_i = w_i$. If $X_i \in V_N$ then by induction hypothesis, $X_i \overset{*}{\underset{G}{\Rightarrow}} w_i$, So $A \overset{*}{\underset{G}{\Rightarrow}} X_1 X_2 \cdots X_k \overset{*}{\underset{G}{\Rightarrow}} w_1 w_2 \cdots w_k$.

Hence by PMI, result follows.

By applying this result (i.e. (iii)), we have $w \in L(G_1)$ iff $w \in L(G)$ and $w \neq \Lambda$. $\Rightarrow$ Hence, $L(G_1) = L(G) - \{\Lambda\}$.

Ex. Consider the grammar G whose productions are $S \to aS \mid AB$, $A \to \Lambda$, $B \to \Lambda$, $D \to b$. Construct a grammar G_1 without null productions generating $L(G) - \Lambda$.

Soln. Step1 :

$$W_1 = \{ A_i \in V_N : A_i \to \Lambda \text{ is in } G \}$$
$$= \{ A, B \}$$

$$W_2 = W_1 \cup \{ A_i \in V_N : A_i \to \alpha \text{ with } \alpha \in W_1^* \}$$
$$= \{A, B\} \cup \{S\} = \{ S, A, B \}$$

$$W_3 = W_2 \cup \phi = W_2.$$

$$\therefore W = \{ S, A, B \}.$$

Step2 : ~~$D \to b$~~, ~~$S \to aS$~~

(i) $D \to b$ is included in P'.

(ii) $S \to aS$ gives rise to $S \to aS$ and $S \to a$

(iii) $S \to AB$ gives rise to $S \to AB$, $S \to A$ and $S \to B$.

(we can not erase both A, B in $S \to AB$ as we will get $S \to \Lambda$).

Hence $G_1 = (\{S, A, B\}, \{a, b\}, P', S)$

Elimination of Unit Productions.

consider, e.g. G as $S \to A$, $A \to B$, $B \to C$, $C \to a$. Then $L(G) = \{a\}$.

The productions $S \to A$, $A \to B$, $B \to C$ are useful just to replace S by C.

To get terminal string, we need $C \to a$. If G_1 is $S \to a$, then $L(G_1) = L(G)$.

Defn. A unit production in CFG G is a production of the form $A \to B$, where $A, B \in V_N$.

Theorem. If G is a CFG, we can find a CFG G_1 which has no null productions or unit productions s.t. $L(G_1) = L(G)$.

Pf. For null productions, we have already done. i.e. we find G' s.t. $L(G') = L(G)$

Let $A \in V_N$.

Step1. Construction of the set of variables from A :

Define $W_i(A)$ as follows :

$W_0(A) = \{A\}$

$W_{i+1}(A) = W_i(A) \cup \{B \in V_N : C \to B \text{ is in } P \text{ with } C \in W_i(A)\}$

By defn. $W_i(A) \subseteq W_{i+1}(A)$. As V_N is finite, $W_{K+1}(A) = W_K(A)$ for some $K \leq |V_N|$. So $W_{K+j}(A) = W_i(A) \; \forall \; j \geq 0$.

Let $W(A) = W_K(A)$. Then $W(A)$ is the set of variables derivable from A.

Step2. construction of A- productions in G_1:

The A- productions in G_1 are either (i) the nonunit production in G' or (ii) $A \to \alpha$ whenever $B \to \alpha$ is in G with $B \in W(A)$ and $\alpha \notin V_N$.

Define $G_1 = (V_N, \Sigma, P_1, S)$, where P_1 is constructed using step 2 $\forall \; A \in V_N$.

(we have given only construction). Proof of $L(G_1) = L(G)$ can be seen from the book.

Corollary. If G is a CFG, we can construct an equivalent grammar G' which is reduced and has no null productions or unit productions.

pf. eliminate null productions to get G_1. Then eliminate unit productions in G_1 to get G_2. Construct a reduced grammar G' equivalent to G_2. G' is the required grammar.

(do not change the order of construction).

Normal Forms For CFG. In CFG, the RHS of a production can be any string of variables and terminals. When the productions in G satisfy certain restrictions, then G is said to be in a 'normal form'. Among several normal forms, main normal forms are 'chomsky normal form' (CNF) and 'Greibach normal form' (GNF).

Defn. (CNF): A CFG 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 $\Lambda \in L(G)$, where $\Lambda \in L(G)$, we assume that S does not appear on the RHS. of any production.

Ex. Let G be $S \to AB$, $A \to a$, $B \to C/b$, $C \to D$, $D \to E$ and $E \to a$. Eliminate unit productions.

Soln. **Step1.** $W_0(S) = \{S\}$, $W_1(S) = W_0(S) \cup \phi$

$\therefore W(S) = (S)$. Similarly $W(A) = A$, $W(E) = E$.

$W_0(B) = \{B\}$, $W_1(B) = \{B\} \cup \{C\} = \{B, C\}$

$W_2(B) = \{B, C\} \cup \{D\}$, $W_3(B) = \{B, C, D, E\} = W_4(B)$

$\therefore W(B) = \{B, C, D, E\}$.

$W_0(C) = \{C\}$, $W_1(C) = \{C, D\}$, $W_2(E) = \{C, D, E\} = W_3(C)$

$\therefore W(C) = \{C, D, E\}$

$W_0(D) = \{D\}$, $W_1(D) = \{D, E\} = W_2(D)$

$\therefore W(D) = \{D, E\}$

Step2. The productions in G_1 are

$S \to AB$, $A \to a$, $E \to a$

$B \to b/a$, $C \to a$, $D \to a$.

e.g. Consider G whose productions are $S \to AB \mid \Lambda$, $A \to a$, $B \to b$. Then G is in CNF.

Note: For a grammar in CNF, the derivation tree has the following property: every node has almost two descendants — either two internal nodes or a single leaf.

Reduction to CNF - We develop a method of constructing a grammar in CNF equivalent to a given CFG.

Step1. Elimination of null and unit productions.
First we eliminate null productions and then for resulting grammar, eliminate unit productions. Let the grammar thus obtained be $G = (V_N, \Sigma, P, S)$.

Step2. (Elimination of terminals on R.H.S.)
Define $G_1 = (V_N', \Sigma, P_1, S)$, where P_1 and 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 $A \to X_1 X_2 \cdots X_n$ with some terminals on R.H.S. 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 $A \to X_1 X_2 \cdots X_n$, every terminal on R.H.S is replaced by the corresponding new variable and the variables on the R.H.S. are retained. The resulting production is added to P_1. Thus, we get $G_1 = (V_N', \Sigma, P_1, S)$.

Step3 (Restricting the no. of variables on RHS)
For any production in P_1, the R.H.S. 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 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 $A \longrightarrow A_1 A_2 \cdots A_m$, where $m \geq 3$. we introduce new productions $A \longrightarrow A_1 C_1$, $C_1 \longrightarrow A_2 C_2 \cdots C_{m-2} \longrightarrow A_{m-1} A_m$ and new variables $C_1, C_2, \cdots, C_{m-2}$. These are added to P'' and V_N'' respectively.

Thus, we get G_2 in CNF.

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

Soln. Step1. There are no null productions or unit productions.

Step2. Let $G_1 = (V_N', \{a, b, d\}, P_1, S)$, where P_1 and V_N' are constructed as follows:

(i) $B \to b$, $D \to d$ are included in P_1.

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

$A \to aB$ gives rise to $A \to C_a B$

$A \to bAB$ gives rise to $A \to C_b AB$ and $C_b \to b$.

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

Step3. P_1 consists of $S \to C_a AD$, $A \to C_a B \mid C_b AB$, $B \to b$, $D \to d$, $C_a \to a$, $C_b \to b$.

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

$S \to C_a AD$ is replaced by $S \to C_a C_1$ and $C_1 \to AD$.

$A \to C_b AB$ is replaced by $A \to C_b C_2$ and $C_2 \to AB$.

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

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

Then G_2 is in CNF.

Ex. Find a grammar in CNF equivalent to the grammar $S \to nS \mid [S \supset S] \mid p \mid q$ (S being the only variable).

Soln. Step 1 : There are no null or unit productions.

Step 2. Let $G_1 = (V_N', \Sigma, P_1, S)$.

(i) $S \to p \mid q$ are added to P_1.

(ii) $S \to nS$ gives $S \to AS$ and $A \to n$,

(iii) $S \to [S \supset S]$ gives $S \to BSCSD$, $B \to [$, $C \to \supset$, $D \to]$

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

Step 3. P_1 consists of $S \to p \mid q$, $S \to AS$, $A \to n$, $B \to [$, $C \to \supset$, $D \to]$, $S \to BSCSD$.

$S \to BSCSD$ is replaced by $S \to BC_1$, $C_1 \to SC_2$, $C_2 \to CC_3$, $C_3 \to SD$.

Greibach Normal Form (GNF)

Defn. A CFG is in GNF if every production is of the form $A \to a\alpha$, where $\alpha \in V_N^*$ & $a \in \Sigma$ (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 the R.H.S. of any production. e.g. G given by $S \to aAB \mid \Lambda$, $A \to bC$, $B \to b$, $C \to c$ is in GNF.

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 $B \to \beta_1 \mid \beta_2 \mid \cdots \mid \beta_s$, Define $P_1 = (P - \{A \to B\gamma\}) \cup \{A \to \beta_i \gamma : 1 \le i \le s\}$. Then $G_1 = (V_N, \Sigma, P_1, S)$ is a CFG equivalent to G.

Lemma 2. Let $G = (V_N, \Sigma, P, S)$ be a CFG. Let the set of A-productions be $A \to A\alpha_1 | \cdots | A\alpha_r | \beta_1 | \cdots | \beta_s$ (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 $A \to \beta_1 | \beta_2 | \cdots | \beta_s$

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

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

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

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

Ex. Apply Lemma 1 to the following A-production $A \to Bab$ where B-productions are $B \to aA | bB | aa | AB$.

Soln.

$B \to aA$ gives $A \to aAab$

$B \to bB$ gives $A \to bBab$

$B \to aa$ gives $A \to aaab$

$B \to AB$ gives $A \to ABab$

Ex. Apply lemma 2 to the following A-productions in a CFG G.

$A \to aBD | bDB | c$, $A \to AB | AD$

Soln. $\alpha_1 = B$, $\alpha_2 = D$, ~~$\alpha_3 = c$~~ , $\beta_1 = aBD$, $\beta_2 = bDB$, $\beta_3 = c$. So the new productions are:

(i) $A \to aBD | bDB | c$, $A \to aBDZ | bDBZ | cZ$

(ii) $Z \to B$, $Z \to D$, $Z \to BZ | DZ$

Reduction to GNF

Case 1 : construction of G (when $\Lambda \notin L$).

Step 1. Eliminate null productions and then construct a grammar G in CNF generating L. We rename the variables as $A_1, A_2, \ldots, A_n$ with $S = A_1$. $G = (\{A_1, A_2, \ldots, A_n\}, \Sigma, P, A_1)$.

Step2. To get productions in the form $A_i \to a\gamma$ or $A_i \to A_j\gamma$, where $j > i$, convert the A_i - productions ($i = 1, 2, \cdots, n-1$) to the form $A_i \to A_i\gamma$ s.t. $j > i$.

Step3. convert A_n - productions to the form $A_n \to a\gamma$. Here the productions of the form $A_n \to A_n\gamma$ are eliminated using lemma 2. The resulting A_n - productions are of the form $A_n \to a\gamma$.

Step4. Modify the A_i - productions to the form $A_i \to a\gamma$, $i = 1, 2, \cdots, n-1$. At the end of step 3, the A_n - productions are of the form $A_n \to a\gamma$. The A_{n-1} - productions are of the form $A_{n-1} \to a\gamma'$ or $A_{n-1} \to A_n\gamma$. By applying lemma 1, we eliminate productions of the form $A_{n-1} \to A_n\gamma$. The resulting A_{n-1} - productions are in the required form. Repeat the construction by considering $A_{n-2}, A_{n-3}, \cdots, A_1$.

Step5. Modify Z_i - productions. Every time apply lemma 2, we get a new variable. (take it as Z_i when apply the lemma for A_i - productions). The Z_i - productions are of the form $Z_i \to \alpha Z_i$ or $Z_i \to \alpha$ (where α is obtained from $A_i \to A_i\alpha$) and hence of the form $Z_i \to a\gamma$ or $Z_i \to A_k\gamma$ for some k. At the end of step 4, the R.H.S. of any A_k - production starts with a terminal. So, we can apply lemma 1 to eliminate $Z_i \to A_k\gamma$. Thus at the end of step 5, we get an equivalent grammar G_1 in GNF.

It is easy to see that G_1 is in GNF. We start with G in CNF. In G, any A - production is of the form $A \to a$ or $A \to AB$ or $A \to CD$. When we apply lemma 1 or 2 in step 2, we get new productions of the form $A \to a\alpha$ or $A \to \beta$ where $\alpha \in V_N^*$, $\beta \in V_N^+$ and $a \in \Sigma$. In steps 3 - 5, the productions are modified to the form $A \to a\alpha$ or $Z \to a'\alpha'$, where $a, a' \in \Sigma$, $\alpha, \alpha' \in V_N^*$.

Case 2. construction of G when $\Lambda \in L$.

By the previous construction we get $G' = (V_N', \Sigma, P_1, S)$ in GNF s.t. $L(G') = L - \{\Lambda\}$. Define a new grammar G_1 as

$G_1 = (V_N' \cup \{S'\}, \Sigma, P_1 \cup \{S' \to S, S' \to \Lambda\}, S')$

$S' \to S$ can be eliminated. (elimination of unit production). As S-productions are in the required form, S'-productions are also in the required form. So G_1 is in GNF.

Ex. construct a grammar in GNF equivalent to the grammar $S \to AA \mid a$, $A \to SS \mid b$.

Soln. The given grammar is in CNF, S and A are renamed as A_1 and A_2 respectively. So the productions are

$A_1 \to A_2A_2 \mid a$ and $A_2 \to A_1A_1 \mid b$.

As the given grammar has no null productions and is in CNF, leave step1.

Step2. (i) A_1-productions are in the required form,

(ii) $A_2 \to b$ is in the required form, Apply lemma1 to $A_2 \to A_1A_1$, the resulting productions are $A_2 \to A_2A_2A_1$, $A_2 \to aA_1$. Thus the A_2-productions are

$A_2 \to A_2A_2A_1$, $A_2 \to aA_1$, $A_2 \to b$.

Step3. we have to apply lemma 2 to A_2-productions as we have $A_2 \to A_2A_2A_1$. Let Z_2 be the new variable. The resulting productions are

$A_2 \to aA_1$, $A_2 \to b$

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

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

Step4. (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 $A_1 \to a$ and eliminate $A_1 \to A_2A_2$ using lemma 1. The resulting productions are $A_1 \to aA_1A_2 \mid bA_2$, $A_1 \to aA_1Z_2A_2 \mid bZ_2A_2$. The set of all modified productions is $A_1 \to a \mid aA_1A_2 \mid bA_2 \mid aA_1Z_2A_2 \mid bZ_2A_2$.

Step 5. The Z_2-productions to be modified are $Z_2 \to A_2A_1$, $Z_2 \to A_2A_1Z_2$. We apply lemma 1 and get

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

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

Hence the equivalent grammar is

$G' = (\{A_1, A_2, Z_2\}, \{a, b\}, P_1, A_1)$, where P_1 consists of

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

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

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

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

Pumping Lemma For Context-Free Languages.

The pumping lemma for CFL gives a method of generating an infinite no. of strings from a given sufficiently long string in CFL L.

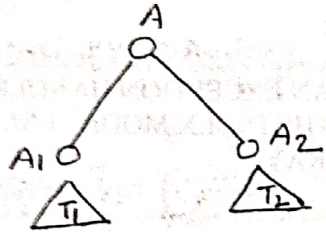
Lemma 3. Let G be a CFG in CNF and T be a derivation tree in G. If the lengths of the longest paths in T is less than or equal to k, then the yield of T is of length less than or equal to 2^{k-1}.

Proof. We prove the result by induction on k, the length of the longest path.

When the longest paths in an A-tree is of length 1, the root has only one son whose label is terminal (when the root has two sons, the labels are variables). So the yield is of length $2^{1-1} = 2^0 = 1$. 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 and A_2. The two subtrees with the two sons

as roots have the longest paths of length less than or equal to k-1.



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 $T = w_1 w_2$, $|w_1 w_2| \leq 2^{k-2} + 2^{k-2} = 2^{k-1}$. By PMI, the result is true for all A-trees and hence all derivation trees.

Pumping Lemma. Let L be a CFL. Then we can find a natural no. n s.t.

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

(ii) $|vx| \geq 1$

(iii) $|vwx| \leq n$

(iv) $uv^k wx^k y \in L \quad \forall k \geq 0$.

Proof When $\wedge \in L$, consider $L - \{\wedge\}$ and construct a grammar $G = (V_N, \Sigma, P, S)$ in CNF generating $L - \{\wedge\}$. (When $\wedge \notin L$, construct G in CNF generating L).

Let $|V_N| = m$ and $n = 2^m$. To prove that n is the required no., we start with $z \in L$, $|z| \geq 2^m$ and construct a derivation tree T of z. If the length of the longest path in T is at most m, by lemma 3, $|z| \leq 2^{m-1}$ (∵ 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 the labels except the last one are variables. As $|V_N| = m$, some labels is repeated.

Choose a repeated label as follows: 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 the first.) Let v_1 & v_2 be the vertices with label B, v_1 being nearer the root, the portion of the path from v_1 to the leaf has only one label, namely B, which is repeated, and so its length is at most $(n+1)$.

Let T_1 and T_2 be the subtrees with v_1, 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 the longest path in T_1 and of length at most $(n+1)$. By the previous lemma, $|z_1| \leq 2^n$. (since z_1 is the yield of T_1).

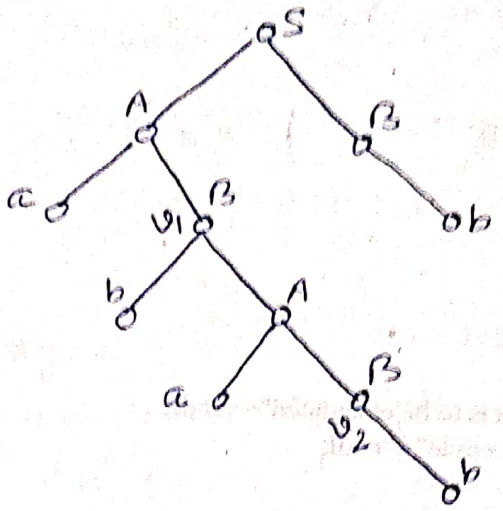
As z and z_1 are the yields of T and a proper subtree T_1 of T, we can write $z = uzy$. As z_1 and w are the yields of T_1 and a proper subtree T_2 of T_1, we can write $z_1 = vwx$. Also $|vwx| > |w|$. So $|vx| \geq 1$. Thus, we have $z = uvwxy$ with $|vwx| \leq n$ & $|vx| \geq 1$.

This proves the pts. (i) — (iii) of the theorem.

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 the pt. (iv) of the theorem.

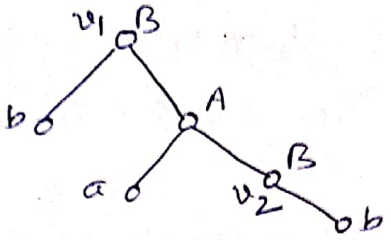
For better understanding, we illustrate the construction whose productions are $S \to AB$, $A \to aB/a$, $B \to bA/b$.



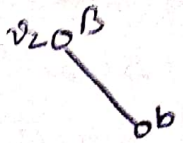
(T)

$\Gamma = S \to A \to B \to A \to B \to b$

$z = ababb$, $z_1 = bab$, $w = b$, $v = ba$, $x = \wedge$, $u = a$, $y = b$



(T_1)



(T_2)

Note! We use pumping lemma to show that L is not CFL. The procedure can be carried out by using the following steps!

Step 1. Assume L is a CFL. Let $n \in \mathbb{N}$ obtained by P. lemma.

Step 2. Choose $z \in L$ so that $|z| \geq n$. Write $z = uvwxy$ using P. lemma.

Step 3. Find a suitable k so that $uv^k wx^k y \notin L$. This is a contradiction and so L is not CFL.

Ex. Show that $L = \{ a^n b^n c^n : n \geq 1 \}$ is not CFL.

Soln. Step1: Assume L is CFL. Let n be the natural no. obtained by P. lemma.

Step2: 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.

Step3: $uvwxy = a^n b^n c^n$. As $1 \leq |vx| \leq n$, v & x cannot contain all the three 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 repetition 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 the repetition of a single symbol (e.g. $x = a^i$, $v = b^j$ for some i, j, $i \leq n, j \leq n$), 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 no. of occurrences of one of the other two symbols in uwy is less than n, and n is the no. 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.

Ex: Show that $L = \{a^p / p \text{ is a prime}\}$ is not a CFL.

Soln: **Step1:** Suppose L is CFL. Let n be the natural no. obtained by the P. lemma.

Step2: Let p be a prime greater than n. Then $z = a^p \in L$, we write $z = uvwxy$

Step 3: By P. lemma, $uv^0wx^0y = uwy \in L$. So $|uwy|$ is a prime no., say q. Let $|vx| = r$. Then $|uv^qwx^qy| = q + qr$. But $q + qr$ is not prime, $\therefore uv^qwx^qy \notin L$. Hence L is not CFL.

Decision Algorithms For CFL

1. Algorithm for deciding whether a CFL is empty.

We can apply the construction of the theorem in which we find an equivalent grammar s.t. every variable in it derives some terminal string for getting $V_N = W_K$. L is non empty iff $S \in W_K$.

2. Algorithm for deciding whether a CFL is finite.

Construct a reduced CFG G in CNF generating $L - \{\Lambda\}$, 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 and A to C. L is finite iff the directed graph has no cycles.