# Assignment No.3

Q.1) Explain the algorithm to eliminate null productions.

→ The production of the form $A \rightarrow \epsilon$ are called as $\epsilon$-productions. such $\epsilon$-prod$^n$ confuse the parser and makes the process of derivation complecated therefore all such $\epsilon$-prod$^n$ are unwanted.
Hence eliminate all such production.
However if empty string is part of language generated by grammar G. i.e. $\Lambda \in L(G)$ then their must be the production $S \rightarrow \epsilon$ in that grammar. where 'S' is a start symbol of that grammar.
Algorithm to findout the nullable variable -

- If $A \rightarrow \epsilon$ is a production in the grammar then A is nullable variable.

- If $B \rightarrow \alpha$ is a production in the grammer & if all the symbol of $\alpha$ found nullable than automatically B also becomes nullable.
  we repeate this process until no more nullable variable found.

- Once nullable variable found sucessfully then ~~translati~~ transform each of the production such a that body of that production contains nullable variable once i.e. you have to write the body of production with nullable variable once, without nullable variable once.

Repeat this process for all the production of grammar as well as for all the nullable variable of the grammer.

• If S is a nullable variable it means that empty string $\epsilon$ is a part of language generated by that grammer. In these case, their must be a prod$^n$ $S \rightarrow \epsilon$ in the resultant grammer obtained by eliminating null productions means in these case the production $S \rightarrow \epsilon$ becomes exceptional & therefore mandatory in the grammar.

For example, ① Eliminate following $\epsilon$-prod$^n$ from grammer.

i) $S \longrightarrow aSb \mid aAb$

$A \longrightarrow \epsilon$

Sol$^n$- The nullable variable in thise ex is A. After that we will check the body of prod$^n$ in which that nullable variable appears;

After that, we write the body with nullable variable once, without nullable variable once.

So, based on this simplified grammer upon eliminating null production can be expressed as

i) $S \longrightarrow aSb$

ii) $S \longrightarrow aAb \mid ab$

---

Q.2) Explain the concept of useful and useless symbols in grammer.

→ <u>Concept of useful symbol</u>-

A symbol is called as usefull, iff its satisfy following 2 criterias -

i) It should be reachable from start symbol of grammer.

ii) It should derive certain part of I/p string.

If a symbol is not following any of above constraint, such symbol is called as useless symbol. Such useless symbol affects on performance of parser.

∴ always eliminate such useless symbols.

Examples on elimination of useless symbols

$S \longrightarrow ABC \mid BaB$

$A \longrightarrow aA \mid BaC \mid aaa$

$B \longrightarrow bBb \mid a$

$C \longrightarrow CA \mid AC$

soln :-

i) $S \longrightarrow ABC$

ii) $S \longrightarrow BaB$

iii) $A \longrightarrow aA$

iv) $A \longrightarrow BaC$

v) $A \longrightarrow aaa$

vi) $B \longrightarrow bBb$

vii) $B \longrightarrow a$

viii) $C \longrightarrow CA$

ix) $C \longrightarrow AC$

Let separate all production in the grammar are above.

Now, we shall try to findout the usefull symbol first. S is usefull always since, it is a start variable.

All A, B, C are reachable from S but unfortunatly 'C' doesn't derive any I/P string, therefore first ~~one~~ useless symbol we found 'C'.

Upon finding 'C' as a useless, we can straight away delete all the production involving 'C' (either in head or in body or both)

Now, Accordingly we can delete productions 1st, 4th, 8th, 9th to form the simplified CFG as

i) $S \longrightarrow BaB$

ii) $A \longrightarrow aA$

iii) $A \longrightarrow aaa$

iv) $B \longrightarrow bBb$

v) $B \longrightarrow a$

Now, in the simplified grammar it is again observed that variable A is unreachable, their itself you can announce A as ~~an~~ useless & we can delete all the prod$^n$ involving A (either in head

or in body or in both)
so, accordingly 2nd & 3rd production will be eliminated to form simplified CGFG as

$S \longrightarrow BaB$

$B \longrightarrow bBb$

$B \longrightarrow a$

Q.3) Explain the algorithm to eliminate unit production.

→ The production of the form $A \rightarrow B$ is called as a unit production, where $A, B \in V$.

i.e. if the single variable is deriving another single variable then such production are unit production. such unit productions are absolutely unwanted.

since they directly affects on the performance of derivation. such production increase the complexity of derivation.

Therefore, all such unit productions must be eliminated.

Algorithm to eliminate unit production from given ~~giv~~ given algorithm -

① select the unit production $A \rightarrow B$ such a that their exist a production $B \rightarrow \alpha$ where $\alpha$ is terminal.

② For every non-unit production $B \rightarrow \alpha$ repeat the following step
Add $A \rightarrow \alpha$ in the grammar.

③ After completion of previous step ② eliminate the prodⁿ $A \rightarrow B$ from grammar.

For example,

$S \rightarrow Aa \mid B$
$B \rightarrow A \mid bb$
$A \rightarrow a \mid bC \mid B$

Solⁿ- First separate all the productions

i) $S \rightarrow Aa$
ii) $S \rightarrow B$
iii) $B \rightarrow A$
iv) $B \rightarrow bb$
v) $A \rightarrow a$
vi) $A \rightarrow bC$
vii) $A \rightarrow B$

$S \rightarrow Aa$
$S \rightarrow bb$
$S \rightarrow a$
$S \rightarrow bC$
$B \rightarrow a$
$B \rightarrow bc$
$B \rightarrow bb$
$A \rightarrow a$
$A \rightarrow bc$
$A \rightarrow bb$.

classmate
Date
Page

Q.4) Explain the concept of union, concatation, kleen's closure of CFG.

→ CFL'S are closed w.r.to some of the ~~functio~~ fundamental operation such as union, concatatnation, kleen closure.

However CFL'S are not closed w.r.t intersection, complement operation.

CFL'S are supporting the operation like union, concatination, kleen closure as their underlying context free grammar are closed with these all operation.

That means, upon performing this are operation on two or more CFG'S the resultant grammer obtained is also a CFG.

Now, we will understand ~~are~~ supporting union, concatation and kleen closure operation.

i) union

let $G_1$ is CFG generating CFL $L_1$ defined as $G_2 = (V_1, \Sigma, S_1, P_1)$, lly, another CFG $G_2$ is generating CFL $L_2$ defined as $G_2 = (V_2, \Sigma, S_2, P_2)$.

upon performing union on grammer $G_1$, $G_2$, we will get resultant CFG $G_u$ generating CFL $= L_1 \cup L_2$.

The grammar $G_u$ can be defined as $G_u = (V_u, \Sigma, P_u, S_u)$

where $V_u = V_1 \cup V_2 \cup S_u$

$\Sigma =$ I/p alphabet

$S_u =$ start symbol of $G_u$.

$P_u = P_1 \cup P_2 \cup \{ S_u \longrightarrow S_1 | S_2 \}$

For ex, consider the grammer $G_1$ as follows,

$S_1 \longrightarrow AB$
$A \longrightarrow aAb | ab$
$B \longrightarrow cBd | cd$

This grammar generates CFL $L_1$ defined as. $L_1 = \{a^m b^m c^n d^n / m \geq 1, n \geq 1\}$

Now, consider another CFG $G_2$ as,

$S_2 \longrightarrow x S_2 y | xy$

It generates CFL $L_2$ as

$L_2 = \{x^k y^k | K \geq 1\}$

upon performing union on $G_1$ & $G_2$, we will get resultant grammar $G_u = G_1 \cup G_2$. Expressed as

$G_u = G_1 \cup G_2$:
$S_u \longrightarrow S_1 | S_2$
$S_1 \longrightarrow AB$
$A \longrightarrow aAb | ab$
$B \longrightarrow cBb | cd$
$S_2 \longrightarrow x S_2 y | xy$

With respect to $G_u$. $V_u = \{S_1, S_2, A, B, S_u\}$, $\Sigma = \{a, b, c, d, x, y\}$ $S_u$ = start symbol of $G_u$, $P_u$ = set of production of $G_1 \cup$ set of all production of $G_2 \cup S_u \rightarrow S_1 | S_2$

ii) Concatation -

like union CFG are supposed to perform concation operation as explained below.

Let $G_1$ is CFG defined as $(V_1, \Sigma, S_1, P_1)$ generating a CFL $L_1$. Another grammer $G_2$ defined as $(V_2, \Sigma, S_2, P_2)$ generating another CFL as $L_2$.

Upon performing concatation operation i.e. joining enter grammar $G_2$ and the of $G_1$. we shall get resultant CFG $G_c$ expressed as $G_c = (G_1 . G_2)$

$G_c = G_1 . G_2$

$G_c = (V_c, \Sigma, S_c, P_c)$

where, $V_c = V_1 \cup V_2 \cup S_c$

$\Sigma$ = I/p alphabet

$S_c$ = start symbol of $G_c$

$P_c = P_1 \cup P_2 \cup \{S_c \rightarrow S_1 . S_2\}$

Now, consider one example

Let context free grammar $G_1$ speciafied as $A \rightarrow aAb \mid ab$. This grammar generate CFL $L_1$ defined as

$L_1 = \{a^m b^m \mid m \geq 1\}$

Another CFG $G_2$ specified as

$B \rightarrow cBd \mid cd$

This grammar generate CFL $L_2$ equal to $\{c^n d^n \mid n \geq 1\}$

upon performing concatation of $G_2 . G_1$ the resultant grammer $G_c$ becomes

$G_c = G_1 . G_2$ :

$G_c$: $S_c \rightarrow A . B$

$A \rightarrow aAb \mid ab$

$B \longrightarrow cBd \mid cd$

where, $V_C = \{S_C, A, B\}$

$\Sigma = \{a, b, c, d\}$

$S_C = \{S_C\}$

$P_C = P_1 \cup P_2 \cup \{S_2 \longrightarrow A \cdot B\}$

3) Kleen closure operation.-

Like union, concatation it is quite possible to perform kleen closure operation of context free grammer.

Let G is CFG defined as

$G = (V, \Sigma, S, P)$

This CFG 'G' generates a CFL 'L'. Now, it is quite possible to perform kleen closure operation on given grammer G. upon performing ~~kk~~ kleen closure on G i.e. $G^*$. The proposed grammar is also ~~pivour~~ purely context in nature.

Let's all is grammar as $G_K$

$G_K$ can be defined by $V_K = V \cup S_K$

where $V_K$ = set of all variable of $G_K$ obtained by $V_K = V \cup S_K$.

$\Sigma_K = \Sigma$

$S_K$ = Start symbol of proposed grammar $G_K$.

$P_K$ = set of all the prod^n of grammar $G_K$ obtained as

$P_K = P \cup \{S_K \longrightarrow S S_K \mid \epsilon\}$

For example, consider following grammar

$$A \longrightarrow a|b$$

This grammer is generating a language containing only two string $\{a,b\}$

If we perform Kleen closure on above grammar, the proposed graman or can be ~~rewirrtten~~ as rewritten

$$S \longrightarrow AS|\epsilon$$

Now, this grammar ~~generate~~ generate a universal language over $\{a,b\}$

This language denoted as $L^*$

Q.5) Write the context free Grammar for Following language.

$$L = \{a^i b^j c^k \mid j \neq i+k\}$$

$\rightarrow$

$j \neq i+k$

case1 — $j < i+k$

case 2 — $j > i+k$

ii) case 2

$$L_2 = \{a^i b^j c^k \mid j > i+k\}$$

$$j > i+k$$

$$j = i+k+m$$

$$\therefore L_2 = \{a^i b^{(i+k+m)} c^k \mid i \geq 0,\ m \geq 1,\ k \geq 0\}$$

Rewrite

$$L_2 = \{\underbrace{a^i b^i}_{Si}\ \underbrace{b^m}_{Sm}\ \underbrace{b^k c^k}_{Sk} \mid i \geq 0,\ m \geq 1,\ k \geq 0\}$$

$S_2 \longrightarrow S_i S_m S_k$

$S_i \longrightarrow a S_i b \mid \epsilon$

$S_m \longrightarrow b S_m \mid b$

$S_k \longrightarrow b S_k c \mid \epsilon$

i) case 1

$L_1 = \{a^i b^j c^k \mid j < i + k\}$

$j < i + k$

p: $j < i$      q: $i \leq j < i + k$

p: $j + n = i$

$L_p = \{a^{j+n} b^j c^k \mid n \geq 1, j \geq 0, k \geq 0\}$

$L_p = \{\underbrace{a^n}_{S_n} \underbrace{a^j b^j}_{S_j} \underbrace{c^k}_{S_c} \mid n \geq 1, j \geq 0, k \geq 0\}$

$S_p \longrightarrow S_n S_j S_c$

$S_n \longrightarrow a S_n \mid a$

$S_j \longrightarrow a S_j b \mid \epsilon$

$S_c \longrightarrow c S_c \mid \epsilon$

q: $i \leq j < i + k$

$b^j$

$\underbrace{(a^i b^i)}_{S_1} \quad \underbrace{(b^{j-i} c^{j-i})}_{S_2} \quad \underbrace{c^{k-j+i}}_{S_3}$

$S_q \longrightarrow S_1 \cdot S_2 \cdot S_3$

$S_1 \longrightarrow aS_1b \mid ab$
$S_2 \longrightarrow bS_2c \mid \epsilon$
$S_3 \longrightarrow cS_3 \mid c$

The grammer for language $L_1$ can be designed performing union on respective grammers for the language $L_p$ or $L_q$.
i.e. proposed grammar should have capability to unite the grammar for $L_p$ as well as $L_q$.
∴ In the resultant grammar for $L_1$ the 1st production should be

$S_1 \longrightarrow S_p \mid S_q$
$S_p \longrightarrow S_n S_j S_c$
$S_n \longrightarrow aS_n \mid a$
$S_j \longrightarrow aS_jb \mid \epsilon$
$S_c \longrightarrow cS_c \mid \epsilon$
$S_q \longrightarrow S_1 \cdot S_2 \cdot S_3$
$S_1 \longrightarrow aS_1b \mid ab$
$S_2 \longrightarrow bS_2c \mid \epsilon$
$S_3 \longrightarrow cS_3 \mid c$
$S \longrightarrow S_1 \mid S_2$
$S_1 \longrightarrow S_p \mid S_q$
$S_p \longrightarrow S_n S_j S_c$
$S_n \longrightarrow aS_n \mid a$
$S_j \longrightarrow aS_jb \mid \epsilon$
$S_c \longrightarrow cS_c \mid \epsilon$
$S_q \longrightarrow S_1 \cdot S_2 \cdot S_3$
$S_1 \longrightarrow aS_1b \mid ab$
$S_2 \longrightarrow bS_2c \mid \epsilon$
$S_3 \longrightarrow cS_3 \mid c$

$S_2 \rightarrow S_i S_m S_k$

$S_i \rightarrow a S_i b \mid \epsilon$

$S_m \rightarrow b S_m \mid b$

$S_k \rightarrow b S_k c \mid \epsilon$

17/10.