## Convert a CFG into its Proper Form

**Definition**: A CFG G is <u>proper</u> if it is
(1) ε-free, and
(2) cycle free (i.e., no single production), and
(3) has no "useless" symbols.

**Definition**: A CFG is <u>ε-free</u> if either
(1) it does not have a ε-production, or
(2) it has exactly one ε-production $S \rightarrow \varepsilon$ and the initial symbol S does not appear on the right side of any production.

**Definition**: A production of the form $A \rightarrow B$ is called <u>single production</u>.

**Definition**: A symbol is <u>useless</u> if it is nonproductive or inaccessible.

**Definition**: A non-terminal symbol A is <u>nonproductive</u> if $A \xrightarrow{*} x$ is not true $\forall x \in \Sigma^*$.

**Definition**: A symbol $x$ is <u>accessible</u> if for some $\alpha, \beta \in (N \cup \Sigma)^*$, $S \xrightarrow{*} \alpha\, x\, \beta$.

A. Algorithm to find an equivalent ε-free grammar for any CFG:

Step 1 - identify the set of all nonterminals that can generate the empty string

Iterative algorithm:

Let $E_0 = \{\}$

$E_{n+1} = E_n \cup \{ A \mid A \rightarrow ð \text{ is in P and } ð \text{ in } E_n^* \}$

The process terminates when $E_i = E_{i+1}$ and let the set of all nonterminals that generate the empty string be denoted by E.

Step 2 - replace every production of the form

$A \rightarrow r_1B_1r_2B_2 \,.....\, r_kB_kr_{k+1}$ where $B_i$ in E and $r_i$ in $[\,(N\text{-}E) \cup \Sigma\,]^*$

by the set of all productions of the form

$A \rightarrow r_1x_1r_2x_2 \,.....\, r_kx_kr_{k+1}$ where $x_i$ in $\{ B_i, \varepsilon \}$

without adding $A \rightarrow \varepsilon$ ( this could occur if all $r_i = \varepsilon$ )

Step 3 - if the initial symbol S is in E, then add a new initial symbol S' and the rules $S' \rightarrow S \mid \varepsilon$

B. Algorithm to eliminate single productions:

Step 1 - For each nonterminal A, compute the set $N_A$

Iterative algorithm:

Let $N_0 = \{ A \}$

$N_{i+1} = N_i \cup \{ C \mid B \rightarrow C \text{ is in P and B in } N_i \}$

The process terminates when $N_i = N_{i+1}$.

Step 2 - If $B \rightarrow \alpha$ is in P and not a single production, place $A \rightarrow \alpha$ to the grammar for all A such that B in $N_A$, and delete $A \rightarrow B$.

C. Iterative algorithm to identify all productive symbols in a grammar G:

Let $N_0 = \{\}$

$N_{i+1} = N_i \cup \{ A \mid A \rightarrow ð \text{ and } ð \text{ in } (N_i \cup \Sigma)^* \}$

The process terminates when $N_i = N_{i+1}$. At that point, $N_i$ contains all of the productive symbols. The nonproductive symbols are those not in $N_i$ at this point.

D. Iterative algorithm to compute the set of all accessible symbols:

Let $V_0 = \{ S \}$

$V_{n+1} = V_n \cup \{ x \mid \text{there exist } ð, ß \text{ in } (N \cup \Sigma)^* \text{ and a symbol A in } V_n$ such that $A \rightarrow ð\ x\ ß$ is a production $\}$

The process terminates when $V_i = V_{i+1}$. Note that we have to throw away nonproductive symbols before we throw away inaccessible symbols.