We can now summarize the various simplifications described so far. We want to convert any CFG G into an equivalent CFG that has no useless symbols, ϵ-productions, or unit productions. Some care must be taken in the order of application of the constructions. A safe order is:

1. Eliminate ϵ-productions.
2. Eliminate unit productions.
3. Eliminate useless symbols.

You should notice that, just as in Section 7.1.1, where we had to order the two steps properly or the result might have useless symbols, we must order the three steps above as shown, or the result might still have some of the features we thought we were eliminating.

Theorem 7.14: If G is a CFG generating a language that contains at least one string other than ϵ, then there is another CFG G_1 such that $L(G_1) = L(G) - \{\epsilon\}$, and G_1 has no ϵ-productions, unit productions, or useless symbols.

PROOF: Start by eliminating the ϵ-productions by the method of Section 7.1.3. If we then eliminate unit productions by the method of Section 7.1.4, we do not introduce any ϵ-productions, since the bodies of the new productions are each identical to some body of an old production. Finally, we eliminate useless symbols by the method of Section 7.1.1. As this transformation only eliminates productions and symbols, never introducing a new production, the resulting grammar will still be devoid of ϵ-productions and unit productions. □

7.1.5 Chomsky Normal Form

We complete our study of grammatical simplifications by showing that every nonempty CFL without ϵ has a grammar G in which all productions are in one of two simple forms, either:

1. $A \to BC$, where A, B, and C, are each variables, or
2. $A \to a$, where A is a variable and a is a terminal.

Further, G has no useless symbols. Such a grammar is said to be in *Chomsky Normal Form*, or CNF.[1]

To put a grammar in CNF, start with one that satisfies the restrictions of Theorem 7.14; that is, the grammar has no ϵ-productions, unit productions, or useless symbols. Every production of such a grammar is either of the form $A \to a$, which is already in a form allowed by CNF, or it has a body of length 2 or more. Our tasks are to:

[1] N. Chomsky is the linguist who first proposed context-free grammars as a way to describe natural languages, and who proved that every CFG could be converted to this form. Interestingly, CNF does not appear to have important uses in natural linguistics, although we shall see it has several other uses, such as an efficient test for membership of a string in a context-free language (Section 7.4.4).

a) Arrange that all bodies of length 2 or more consist only of variables.

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

The construction for (a) is as follows. For every terminal a that appears in a body of length 2 or more, create a new variable, say A. This variable has only one production, $A \to a$. Now, we use A in place of a everywhere a appears in a body of length 2 or more. At this point, every production has a body that is either a single terminal or at least two variables and no terminals.

For step (b), we must break those productions $A \to B_1 B_2 \cdots B_k$, for $k \geq 3$, into a group of productions with two variables in each body. We introduce $k-2$ new variables, $C_1, C_2, \ldots, C_{k-2}$. The original production is replaced by the $k-1$ productions

$$A \to B_1 C_1, \quad C_1 \to B_2 C_2, \ldots, C_{k-3} \to B_{k-2} C_{k-2}, \quad C_{k-2} \to B_{k-1} B_k$$

Example 7.15: Let us convert the grammar of Example 7.12 to CNF. For part (a), notice that there are eight terminals, a, b, 0, 1, $+$, $*$, (, and), each of which appears in a body that is not a single terminal. Thus, we must introduce eight new variables, corresponding to these terminals, and eight productions in which the new variable is replaced by its terminal. Using the obvious initials as the new variables, we introduce:

$$\begin{array}{llll} A \to a & B \to b & Z \to 0 & O \to 1 \\ P \to + & M \to * & L \to (& R \to) \end{array}$$

If we introduce these productions, and replace every terminal in a body that is other than a single terminal by the corresponding variable, we get the grammar shown in Fig. 7.2.

$$\begin{array}{lcl} E & \to & EPT \mid TMF \mid LER \mid a \mid b \mid IA \mid IB \mid IZ \mid IO \\ T & \to & TMF \mid LER \mid a \mid b \mid IA \mid IB \mid IZ \mid IO \\ F & \to & LER \mid a \mid b \mid IA \mid IB \mid IZ \mid IO \\ I & \to & a \mid b \mid IA \mid IB \mid IZ \mid IO \\ A & \to & a \\ B & \to & b \\ Z & \to & 0 \\ O & \to & 1 \\ P & \to & + \\ M & \to & * \\ L & \to & (\\ R & \to &) \end{array}$$

Figure 7.2: Making all bodies either a single terminal or several variables

Now, all productions are in Chomsky Normal Form except for those with the bodies of length 3: EPT, TMF, and LER. Some of these bodies appear in more than one production, but we can deal with each body once, introducing one extra variable for each. For EPT, we introduce new variable C_1, and replace the one production, $E \to EPT$, where it appears, by $E \to EC_1$ and $C_1 \to PT$.

For TMF we introduce new variable C_2. The two productions that use this body, $E \to TMF$ and $T \to TMF$, are replaced by $E \to TC_2$, $T \to TC_2$, and $C_2 \to MF$. Then, for LER we introduce new variable C_3 and replace the three productions that use it, $E \to LER$, $T \to LER$, and $F \to LER$, by $E \to LC_3$, $T \to LC_3$, $F \to LC_3$, and $C_3 \to ER$. The final grammar, which is in CNF, is shown in Fig. 7.3. □

$$
\begin{array}{lcl}
E & \to & EC_1 \mid TC_2 \mid LC_3 \mid a \mid b \mid IA \mid IB \mid IZ \mid IO \\
T & \to & TC_2 \mid LC_3 \mid a \mid b \mid IA \mid IB \mid IZ \mid IO \\
F & \to & LC_3 \mid a \mid b \mid IA \mid IB \mid IZ \mid IO \\
I & \to & a \mid b \mid IA \mid IB \mid IZ \mid IO \\
A & \to & a \\
B & \to & b \\
Z & \to & 0 \\
O & \to & 1 \\
P & \to & + \\
M & \to & * \\
L & \to & (\\
R & \to &) \\
C_1 & \to & PT \\
C_2 & \to & MF \\
C_3 & \to & ER
\end{array}
$$

Figure 7.3: Making all bodies either a single terminal or two variables

Theorem 7.16: If G is a CFG whose language contains at least one string other than ϵ, then there is a grammar G_1 in Chomsky Normal Form, such that $L(G_1) = L(G) - \{\epsilon\}$.

PROOF: By Theorem 7.14, we can find CFG G_2 such that $L(G_2) = L(G) - \{\epsilon\}$, and such that G_2 has no useless symbols, ϵ-productions, or unit productions. The construction that converts G_2 to CNF grammar G_1 changes the productions in such a way that each production of G_2 can be simulated by one or more productions of G_1. Conversely, the introduced variables of G_1 each have only one production, so they can only be used in the manner intended. More formally, we prove that w is in $L(G_2)$ if and only if w is in $L(G_1)$.

(Only-if) If w has a derivation in G_2, it is easy to replace each production used, say $A \to X_1X_2\cdots X_k$, by a sequence of productions of G_1. That is,

one step in the derivation in G_2 becomes one or more steps in the derivation of w using the productions of G_1. First, if any X_i is a terminal, we know G_1 has a corresponding variable B_i and a production $B_i \to X_i$. Then, if $k > 2$, G_1 has productions $A \to B_1C_1$, $C_1 \to B_2C_2$, and so on, where B_i is either the introduced variable for terminal X_i or X_i itself, if X_i is a variable. These productions simulate in G_1 one step of a derivation of G_2 that uses $A \to X_1X_2\cdots X_k$. We conclude that there is a derivation of w in G_1, so w is in $L(G_1)$.

(If) Suppose w is in $L(G_1)$. Then there is a parse tree in G_1, with S at the root and yield w. We convert this tree to a parse tree of G_2 that also has root S and yield w.

First, we "undo" part (b) of the CNF construction. That is, suppose there is a node labeled A, with two children labeled B_1 and C_1, where C_1 is one of the variables introduced in part (b). Then this portion of the parse tree must look like Fig. 7.4(a). That is, because these introduced variables each have only one production, there is only one way that they can appear, and all the variables introduced to handle the production $A \to B_1B_2\cdots B_k$ must appear together, as shown.

Any such cluster of nodes in the parse tree may be replaced by the production that they represent. The parse-tree transformation is suggested by Fig. 7.4(b).

The resulting parse tree is still not necessarily a parse tree of G_2. The reason is that step (a) in the CNF construction introduced other variables that derive single terminals. However, we can identify these in the current parse tree and replace a node labeled by such a variable A and its one child labeled a, by a single node labeled a. Now, every interior node of the parse tree forms a production of G_2. Since w is the yield of a parse tree in G_2, we conclude that w is in $L(G_2)$. □

7.1.6 Exercises for Section 7.1

* **Exercise 7.1.1:** Find a grammar equivalent to

$$
\begin{array}{lcl}
S & \to & AB \mid CA \\
A & \to & a \\
B & \to & BC \mid AB \\
C & \to & aB \mid b
\end{array}
$$

with no useless symbols.

* **Exercise 7.1.2:** Begin with the grammar:

$$
\begin{array}{lcl}
S & \to & ASB \mid \epsilon \\
A & \to & aAS \mid a \\
B & \to & SbS \mid A \mid bb
\end{array}
$$