Chapter 6

Simplification of Context-Free Grammars and Normal Forms

Before we can study context-free languages in greater depth, we must attend to some technical matters. The definition of a context-free grammar imposes no restriction whatsoever on the right side of a production. However, complete freedom is not necessary, and in fact, is a detriment in some arguments. In Theorem 5.2, we saw the convenience of certain restrictions on grammatical forms; eliminating rules of the form $A \to \lambda$ and $A \to B$ made the arguments easier. In many instances, it is desirable to place even more stringent restrictions on the grammar. Because of this, we need to look at methods for transforming an arbitrary context-free grammar into an equivalent one that satisfies certain restrictions on its form. In this chapter we study several transformations and substitutions that will be useful in subsequent discussions.

We also investigate **normal forms** for context-free grammars. A normal form is one that, although restricted, is broad enough so that any grammar has an equivalent normal-form version. We introduce two of the most useful of these, the **Chomsky normal form** and the **Greibach normal form.** Both have many practical and theoretical uses. An immediate application of the Chomsky normal form to parsing is given in Section 6.3.

The somewhat tedious nature of the material in this chapter lies in the fact that many of the arguments are manipulative and give little intuitive insight. For our purposes, this technical aspect is relatively unimportant and can be read casually. The various conclusions are significant; they will be used many times in later discussions.

6.1 Methods for Transforming Grammars

We first raise an issue that is somewhat of a nuisance with grammars and languages in general: the presence of the empty string. The empty string plays a rather singular role in many theorems and proofs, and it is often necessary to give it special attention. We prefer to remove it from consideration altogether, looking only at languages that do not contain λ. In doing so, we do not lose generality, as we see from the following considerations. Let L be any context-free language, and let $G = (V, T, S, P)$ be a context-free grammar for $L - \{\lambda\}$. Then the grammar we obtain by adding to V the new variable S_0, making S_0 the start variable, and adding to P the productions

$$S_0 \to S|\lambda,$$

generates L. Therefore any nontrivial conclusion we can make for $L - \{\lambda\}$ will almost certainly transfer to L. Also, given any context-free grammar G, there is a method for obtaining $\widehat{G}$ such that $L\left(\widehat{G}\right) = L(G) - \{\lambda\}$ (see Exercise 13 at the end of this section). Consequently, for all practical purposes, there is no difference between context-free languages that include λ and those that do not. For the rest of this chapter, unless otherwise stated, we will restrict our discussion to λ-free languages.

A Useful Substitution Rule

Many rules govern generating equivalent grammars by means of substitutions. Here we give one that is very useful for simplifying grammars in various way. We will not define the term *simplification* precisely, but we will use it nevertheless. What we mean by it is the removal of certain types of undesirable productions; the process does not necessarily result in an actual reduction of the number of rules.

Theorem 6.1 Let $G = (V, T, S, P)$ be a context-free grammar. Suppose that P contains a production of the form

$$A \to x_1 B x_2.$$

Assume that A and B are different variables and that

$$B \to y_1 |y_2| \cdots |y_n$$

is the set of all productions in P which have B as the left side. Let $\widehat{G} = \left(V, T, S, \widehat{P}\right)$ be the grammar in which $\widehat{P}$ is constructed by deleting

$$A \to x_1 B x_2 \tag{6.1}$$

from P, and adding to it

$$A \to x_1 y_1 x_2 \,|x_1 y_2 x_2| \cdots |x_1 y_n x_2.$$

Then

$$L\left(\widehat{G}\right) = L(G).$$

Proof: Suppose that $w \in L(G)$, so that

$$S \overset{*}{\Rightarrow}_G w.$$

The subscript on the derivation sign $\Rightarrow$ is used here to distinguish between derivations with different grammars. If this derivation does not involve the production (6.1), then obviously

$$S \overset{*}{\Rightarrow}_{\widehat{G}} w.$$

If it does, then look at the derivation the first time (6.1) is used. The B so introduced eventually has to be replaced; we lose nothing by assuming that this is done immediately (see Exercise 17 at the end of this section). Thus

$$S \overset{*}{\Rightarrow}_G u_1 A u_2 \Rightarrow_G u_1 x_1 B x_2 u_2 \Rightarrow_G u_1 x_1 y_j x_2 u_2.$$

But with grammar $\widehat{G}$ we can get

$$S \overset{*}{\Rightarrow}_{\widehat{G}} u_1 A u_2 \Rightarrow_{\widehat{G}} u_1 x_1 y_j x_2 u_2.$$

Thus we can reach the same sentential form with G and $\widehat{G}$. If (6.1) is used again later, we can repeat the argument. It follows then, by induction on the number of times the production is applied, that

$$S \overset{*}{\Rightarrow}_{\widehat{G}} w.$$

Therefore, if $w \in L(G)$, then $w \in L\left(\widehat{G}\right)$.

By similar reasoning, we can show that if $w \in L\left(\widehat{G}\right)$, then $w \in L(G)$, completing the proof. ■

Theorem 6.1 is a simple and quite intuitive substitution rule: A production $A \to x_1 B x_2$ can be eliminated from a grammar if we put in its place

the set of productions in which B is replaced by all strings it derives in one step. In this result, it is necessary that A and B be different variables. The case when $A = B$ is partially addressed in Exercises 22 and 23 at the end of this section.

Example 6.1 Consider $G = (\{A, B\}, \{a, b, c\}, A, P)$ with productions

$$A \rightarrow a \,|aaA|\, abBc,$$
$$B \rightarrow abbA|b.$$

Using the suggested substitution for the variable B, we get the grammar $\widehat{G}$ with productions

$$A \rightarrow a \,|aaA|\, ababbAc|abbc,$$
$$B \rightarrow abbA|b.$$

The new grammar $\widehat{G}$ is equivalent to G. The string $aaabbc$ has the derivation

$$A \Rightarrow aaA \Rightarrow aaabBc \Rightarrow aaabbc$$

in G, and the corresponding derivation

$$A \Rightarrow aaA \Rightarrow aaabbc$$

in $\widehat{G}$.

Notice that, in this case, the variable B and its associated productions are still in the grammar even though they can no longer play a part in any derivation. We will see shortly how such unnecessary productions can be removed from a grammar. ■

Removing Useless Productions

One invariably wants to remove productions from a grammar that can never take part in any derivation. For example, in the grammar whose entire production set is

$$S \rightarrow aSb \,|\lambda|\, A,$$
$$A \rightarrow aA,$$

the production $S \rightarrow A$ clearly plays no role, as A cannot be transformed into a terminal string. While A can occur in a string derived from S, this can never lead to a sentence. Removing this production leaves the language unaffected and is a simplification by any definition.

Definition 6.1

Let $G = (V, T, S, P)$ be a context-free grammar. A variable $A \in V$ is said to be **useful** if and only if there is at least one $w \in L(G)$ such that

$$S \overset{*}{\Rightarrow} xAy \overset{*}{\Rightarrow} w, \tag{6.2}$$

with x, y in $(V \cup T)^*$. In words, a variable is useful if and only if it occurs in at least one derivation. A variable that is not useful is called **useless.** A production is useless if it involves any useless variable.

Example 6.2 A variable may be useless because there is no way of getting a terminal string from it. The case just mentioned is of this kind. Another reason a variable may be useless is shown in the next grammar. In a grammar with start symbol S and productions

$$\begin{aligned} S &\to A, \\ A &\to aA|\lambda, \\ B &\to bA, \end{aligned}$$

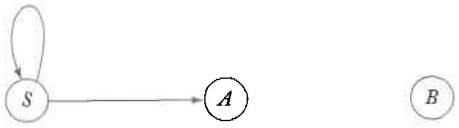
the variable B is useless and so is the production $B \to bA$. Although B can derive a terminal string, there is no way we can achieve $S \overset{*}{\Rightarrow} xBy$. ■

This example illustrates the two reasons why a variable is useless: either because it cannot be reached from the start symbol or because it cannot derive a terminal string. A procedure for removing useless variables and productions is based on recognizing these two situations. Before we present the general case and the corresponding theorem, let us look at another example.

Example 6.3 Eliminate useless symbols and productions from $G = (V, T, S, P)$, where $V = \{S, A, B, C\}$ and $T = \{a, b\}$, with P consisting of

$$\begin{aligned} S &\to aS|A|C, \\ A &\to a, \\ B &\to aa, \\ C &\to aCb. \end{aligned}$$

Figure 6.1



First, we identify the set of variables that can lead to a terminal string. Because $A \rightarrow a$ and $B \rightarrow aa$, the variables A and B belong to this set. So does S, because $S \Rightarrow A \Rightarrow a$. However, this argument cannot be made for C, thus identifying it as useless. Removing C and its corresponding productions, we are led to the grammar G_1 with variables $V_1 = \{S, A, B\}$, terminals $T = \{a\}$, and productions

$$S \rightarrow aS|A,$$
$$A \rightarrow a,$$
$$B \rightarrow aa.$$

Next we want to eliminate the variables that cannot be reached from the start variable. For this, we can draw a **dependency graph** for the variables. Dependency graphs are a way of visualizing complex relationships and are found in many applications. For context-free grammars, a dependency graph has its vertices labeled with variables, with an edge between vertices C and D if and only if there is a production form

$$C \rightarrow xDy.$$

A dependency graph for V_1 is shown in Figure 6.1. A variable is useful only if there is a path from the vertex labeled S to the vertex labeled with that variable. In our case, Figure 6.1 shows that B is useless. Removing it and the affected productions and terminals, we are led to the final answer $\widehat{G} = \left(\widehat{V}, \widehat{T}, S, \widehat{P}\right)$ with $\widehat{V} = \{S, A\}, \widehat{T} = \{a\}$, and productions

$$S \rightarrow aS|A,$$
$$A \rightarrow a.$$

The formalization of this process leads to a general construction and the corresponding theorem. ■

Theorem 6.2 Let $G = (V, T, S, P)$ be a context-free grammar. Then there exists an equivalent grammar $\widehat{G} = \left(\widehat{V}, \widehat{T}, S, \widehat{P}\right)$ that does not contain any useless variables or productions.

Proof: The grammar $\widehat{G}$ can be generated from G by an algorithm consisting of two parts. In the first part we construct an intermediate grammar $G_1 = (V_1, T_2, S, P_1)$ such that V_1 contains only variables A for which

$$A \stackrel{*}{\Rightarrow} w \in T^*$$

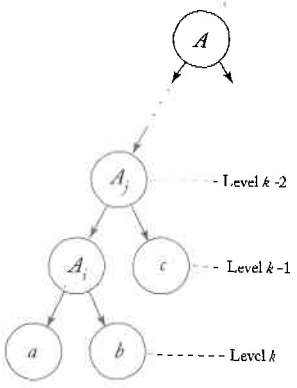
is possible. The steps in the algorithm are:

1. Set V_1 to $\varnothing$.
2. Repeat the following step until no more variables are added to V_1.
   For every $A \in V$ for which P has a production of the form
   $$A \to x_1 x_2 \cdots x_n, \text{ with all } x_i \text{ in } V_1 \cup T,$$
   add A to V_1.
3. Take P_1 as all the productions in P whose symbols are all in $(V_1 \cup T)$.

Clearly this procedure terminates. It is equally clear that if $A \in V_1$, then $A \stackrel{*}{\Rightarrow} w \in T^*$ is a possible derivation with G_1. The remaining issue is whether every A for which $A \stackrel{*}{\Rightarrow} w = ab\cdots$ is added to V_1 before the procedure terminates. To see this, consider any such A and look at the partial derivation tree corresponding to that derivation (Figure 6.2). At level k, there are only terminals, so every variable A_i at level $k-1$ will be added to V_1 on the first pass through Step 2 of the algorithm. Any variable at level $k-2$ will then be added to V_1 on the second pass through Step 2. The third time through Step 2, all variables at level $k-3$ will be added, and so on. The algorithm cannot terminate while there are variables in the tree that are not yet in V_1. Hence A will eventually be added to V_1.

In the second part of the construction, we get the final answer $\widehat{G}$ from G_1. We draw the variable dependency graph for G_1 and from it find all

Figure 6.2



variables that cannot be reached from S. These are removed from the variable set, as are the productions involving them. We can also eliminate any terminal that does not occur in some useful production. The result is the grammar $\widehat{G} = \left(\widehat{V}, \widehat{T}, S, \widehat{P}\right)$.

Because of the construction, $\widehat{G}$ does not contain any useless symbols or productions. Also, for each $w \in L(G)$ we have a derivation

$$S \stackrel{*}{\Rightarrow} xAy \stackrel{*}{\Rightarrow} w.$$

Since the construction of $\widehat{G}$ retains A and all associated productions, we have everything needed to make the derivation

$$S \stackrel{*}{\Rightarrow}_{\widehat{G}} xAy \stackrel{*}{\Rightarrow}_{\widehat{G}} w.$$

The grammar $\widehat{G}$ is constructed from G by the removal of productions, so that $\widehat{P} \subseteq P$. Consequently $L\left(\widehat{G}\right) \subseteq L(G)$. Putting the two results together, we see that G and $\widehat{G}$ are equivalent. ■

Removing λ-Productions

One kind of production that is sometimes undesirable is one in which the right side is the empty string.

Definition 6.2

Any production of a context-free grammar of the form

$$A \rightarrow \lambda$$

is called a **λ-production.** Any variable A for which the derivation

$$A \stackrel{*}{\Rightarrow} \lambda \tag{6.3}$$

is possible is called **nullable.**

A grammar may generate a language not containing λ, yet have some λ-productions or nullable variables. In such cases, the λ-productions can be removed.

Example 6.4 Consider the grammar

$$S \to aS_1b,$$
$$S_1 \to aS_1b|\lambda.$$

This grammar generates the λ-free language $\{a^nb^n : n \geq 1\}$. The λ-production $S_1 \to \lambda$ can be removed after adding new productions obtained by substituting λ for S_1 where it occurs on the right. Doing this we get the grammar

$$S \to aS_1b|ab,$$
$$S_1 \to aS_1b|ab.$$

We can easily show that this new grammar generates the same language as the original one.

In more general situations, substitutions for λ-productions can be made in a similar, although more complicated, manner. ■

Theorem 6.3 Let G be any context-free grammar with λ not in $L(G)$. Then there exists an equivalent grammar $\widehat{G}$ having no λ-productions.

Proof: We first find the set V_N of all nullable variables of G, using the following steps.

1. For all productions $A \to \lambda$, put A into V_N.
2. Repeat the following step until no further variables are added to V_N.
   For all productions
   $$B \to A_1A_2 \cdots A_n,$$
   where $A_1, A_2, ..., A_n$ are in V_N, put B into V_N.

Once the set V_N has been found, we are ready to construct $\widehat{P}$. To do so, we look at all productions in P of the form

$$A \to x_1x_2 \cdots x_m, m \geq 1,$$

where each $x_i \in V \cup T$. For each such production of P, we put into $\widehat{P}$ that production as well as all those generated by replacing nullable variables with λ in all possible combinations. For example, if x_i and x_j are both nullable, there will be one production in $\widehat{P}$ with x_i replaced with λ, one in which x_j is replaced with λ, and one in which both x_i and x_j are replaced with λ.

There is one exception: if all x_i are nullable, the production $A \to \lambda$ is not put into $\widehat{P}$.

The argument that this grammar $\widehat{G}$ is equivalent to G is straightforward and will be left to the reader. ■

Example 6.5 Find a context-free grammar without λ-productions equivalent to the grammar defined by

$$S \to ABaC,$$
$$A \to BC,$$
$$B \to b|\lambda,$$
$$C \to D|\lambda,$$
$$D \to d.$$

From the first step of the construction in Theorem 6.3, we find that the nullable variables are A, B, C. Then, following the second step of the construction, we get

$$S \to ABaC\,|BaC|\,AaC\,|ABa|\,aC\,|Aa|\,Ba|a,$$
$$A \to B\,|C|\,BC,$$
$$B \to b,$$
$$C \to D,$$
$$D \to d.$$

■

Removing Unit-Productions

As we see from Theorem 6.2, productions in which both sides are a single variable are at times undesirable.

Definition 6.3

Any production of a context-free grammar of the form

$$A \to B,$$

where $A, B \in V$ is called a **unit-production.**

To remove unit-productions, we use the substitution rule discussed in Theorem 6.1. As the construction in the next theorem shows, this can be done if we proceed with some care.

Theorem 6.4 Let $G = (V, T, S, P)$ be any context-free grammar without λ-productions. Then there exists a context-free grammar $\widehat{G} = \left(\widehat{V}, \widehat{T}, S, \widehat{P}\right)$ that does not have any unit-productions and that is equivalent to G.

Proof: Obviously, any unit-production of the form $A \to A$ can be removed from the grammar without effect, and we need only consider $A \to B$, where A and B are different variables. At first sight, it may seem that we can use Theorem 6.1 directly with $x_1 = x_2 = \lambda$ to replace

$$A \to B$$

with

$$A \to y_1 \,|y_2| \cdots |y_n.$$

But this will not always work; in the special case

$$A \to B,$$
$$B \to A,$$

the unit-productions are not removed. To get around this, we first find, for each A, all variables B such that

$$A \overset{*}{\Rightarrow} B. \tag{6.4}$$

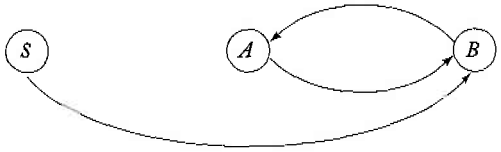
We can do this by drawing a dependency graph with an edge (C, D) whenever the grammar has a unit-production $C \to D$; then (6.4) holds whenever there is a walk between A and B. The new grammar $\widehat{G}$ is generated by first putting into $\widehat{P}$ all non-unit productions of P. Next, for all A and B satisfying (6.4), we add to $\widehat{P}$

$$A \to y_1 \,|y_2| \cdots |y_n,$$

where $B \to y_1 \,|y_2| \cdots y_n$ is the set of all rules in $\widehat{P}$ with B on the left. Note that since $B \to y_1 \,|y_2| \cdots |y_n$ is taken from $\widehat{P}$, none of the y_i can be a single variable, so that no unit-productions are created by the last step.

To show that the resulting grammar is equivalent to the original one we can follow the same line of reasoning as in Theorem 6.1. ■

Figure 6.3



Example 6.6 Remove all unit-productions from

$$S \to Aa|B,$$
$$B \to A|bb,$$
$$A \to a\,|bc|\,B.$$

The dependency graph for the unit-productions is given in Figure 6.3; we see from it that $S \overset{*}{\Rightarrow} A, S \overset{*}{\Rightarrow} B, B \overset{*}{\Rightarrow} A$, and $A \overset{*}{\Rightarrow} B$. Hence, we add to the original non-unit productions

$$S \to Aa,$$
$$A \to a|bc,$$
$$B \to bb,$$

the new rules

$$S \to a\,|bc|\,bb,$$
$$A \to bb,$$
$$B \to a|bc,$$

to obtain the equivalent grammar

$$S \to a\,|bc|\,bb|Aa,$$
$$A \to a\,|bb|\,bc,$$
$$B \to a\,|bb|\,bc.$$

Note that the removal of the unit-productions has made B and the associated productions useless. ■

We can put all these results together to show that grammars for context-free languages can be made free of useless productions, λ-productions, and unit-productions.

Theorem 6.5 Let L be a context-free language that does not contain λ. Then there exists a context-free grammar that generates L and that does not have any useless productions, λ-productions, or unit-productions.

Proof: The procedures given in Theorems 6.2, 6.3, and 6.4 remove these kinds of productions in turn. The only point that needs consideration is that the removal of one type of production may introduce productions of another type; for example, the procedure for removing λ-productions can create new unit-productions. Also, Theorem 6.4 requires that the grammar have no λ-productions. But note that the removal of unit-productions does not create λ-productions (Exercise 15 at the end of this section), and the removal of useless productions does not create λ-productions or unit-productions (Exercise 16 at the end of this section). Therefore, we can remove all undesirable productions using the following sequence of steps:

1. Remove λ-productions
2. Remove unit-productions
3. Remove useless productions

The result will then have none of these productions, and the theorem is proved. ■

EXERCISES

1. Complete the proof of Theorem 6.1 by showing that

$$S \stackrel{*}{\Rightarrow}_{\hat{G}} w$$

implies

$$S \stackrel{*}{\Rightarrow}_{G} w.$$

2. In Example 6.1, show a derivation tree for the string $ababbbac$, using both the original and the modified grammar.

3. Show that the two grammars

$$S \rightarrow abAB|ba,$$
$$A \rightarrow aaa,$$
$$B \rightarrow aA|bb$$

and

$$S \rightarrow abAaA\,|abAbb|\,ba,$$
$$A \rightarrow aaa$$

are equivalent. ●

4. In Theorem 6.1, why is it necessary to assume that A and B are different variables?

5. Eliminate all useless productions from the grammar

$$S \to aS|AB,$$
$$A \to bA,$$
$$B \to AA.$$

What language does this grammar generate?

6. Eliminate useless productions from

$$S \to a|aA|B|C,$$
$$A \to aB|\lambda,$$
$$B \to Aa,$$
$$C \to cCD,$$
$$D \to ddd.$$

7. Eliminate all λ-productions from

$$S \to AaB|aaB,$$
$$A \to \lambda,$$
$$B \to bbA|\lambda.$$

8. Remove all unit-productions, all useless productions, and all λ-productions from the grammar

$$S \to aA|aBB,$$
$$A \to aaA|\lambda,$$
$$B \to bB|bbC,$$
$$C \to B.$$

What language does this grammar generate? ●

9. Eliminate all unit productions from the grammar in Exercise 7.

10. Complete the proof of Theorem 6.3.

11. Complete the proof of Theorem 6.4.

12. Use the construction in Theorem 6.3 to remove λ-productions from the grammar in Example 5.4. What language does the resulting grammar generate?

13. Suppose that G is a context-free grammar for which $\lambda \in L(G)$. Show that if we apply the construction in Theorem 6.3, we obtain a new grammar $\widehat{G}$ such that $L\left(\widehat{G}\right) = L(G) - \{\lambda\}$.

14. Give an example of a situation in which the removal of λ-productions introduces previously nonexistent unit-productions.

15. Let G be a grammar without λ-productions, but possibly with some unit-productions. Show that the construction of Theorem 6.4 does not then introduce any λ-productions.

16. Show that if a grammar has no λ-productions and no unit-productions, then the removal of useless productions by the construction of Theorem 6.2 does not introduce any such productions.

17. Justify the claim made in the proof of Theorem 6.1 that the variable B can be replaced as soon as it appears.

18. Suppose that a context-free grammar $G = (V, T, S, P)$ has a production of the form

$$A \to xy,$$

where $x, y \in (V \cup T)^+$. Prove that if this rule is replaced by

$$A \to By,$$
$$B \to x,$$

where $B \notin V$, then the resulting grammar is equivalent to the original one.

19. Consider the procedure suggested in Theorem 6.2 for the removal of useless productions. Reverse the order of the two parts, first eliminating variables that cannot be reached from S, then removing those that do not yield a terminal string. Does the new procedure still work correctly? If so, prove it. If not, give a counterexample.

20. It is possible to define the term simplification precisely by introducing the concept of **complexity** of a grammar. This can be done in many ways; one of them is through the length of all the strings giving the production rules. For example, we might use

$$\textit{complexity}(G) = \sum_{A \to v \in P} \{1 + |v|\}.$$

Show that the removal of useless productions always reduces the complexity in this sense. What can you say about the removal of λ-productions and unit-productions?

21. A context-free grammar G is said to be minimal for a given language L if $complexity\,(G) \leq complexity\left(\widehat{G}\right)$ for any $\widehat{G}$ generating L. Show by example that the removal of useless productions does not necessarily produce a minimal grammar.

★22. Prove the following result. Let $G = (V, T, S, P)$ be a context-free grammar. Divide the set of productions whose left sides are some given variable (say, A), into two disjoint subsets

$$A \to Ax_1 \,|Ax_2| \cdots |Ax_n,$$

and

$$A \to y_1 \,|y_2| \cdots |y_m,$$

where x_i, y_i are in $(V \cup T)^*$, but A is not a prefix of any y_i. Consider the grammar $\widehat{G} = \left(V \cup \{Z\}, T, S, \widehat{P}\right)$, where $Z \notin V$ and $\widehat{P}$ is obtained by replacing all productions that have A on the left by

$$\begin{aligned} A &\to y_i | y_i Z, \quad i = 1, 2, ..., m, \\ Z &\to x_i | x_i Z, \quad i = 1, 2, ..., n. \end{aligned}$$

Then $L\,(G) = L\left(\widehat{G}\right)$.

23. Use the result of the preceding exercise to rewrite the grammar

$$\begin{aligned} A &\to Aa \,|aBc| \,\lambda \\ B &\to Bb|bc \end{aligned}$$

so that it no longer has productions of the form $A \to Ax$ or $B \to Bx$.

★24. Prove the following counterpart of Exercise 22. Let the set of productions involving the variable A on the left be divided into two disjoint subsets

$$A \to x_1 A \,|x_2 A| \cdots |x_n A$$

and

$$A \to y_1 \,|y_2| \cdots |y_m$$

where A is not a suffix of any y_i. Show that the grammar obtained by replacing these productions with

$$\begin{aligned} A &\to y_i | Z y_i, \quad i = 1, 2, ..., m \\ Z &\to x_i | Z x_i, \quad i = 1, 2, ..., n \end{aligned}$$

is equivalent to the original grammar.

6.2 Two Important Normal Forms

There are many kinds of normal forms we can establish for context-free grammars. Some of these, because of their wide usefulness, have been studied extensively. We consider two of them briefly.

Chomsky Normal Form

One kind of normal form we can look for is one in which the number of symbols on the right of a production are strictly limited. In particular, we can ask that the string on the right of a production consist of no more than two symbols. One instance of this is the **Chomsky normal form.**

Definition 6.4

A context-free grammar is in Chomsky normal form if all productions are of the form

$$A \to BC,$$

or

$$A \to a,$$

where A, B, C are in V, and a is in T.

Example 6.7 The grammar

$$S \to AS|a,$$
$$A \to SA|b$$

is in Chomsky normal form. The grammar

$$S \to AS|AAS,$$
$$A \to SA|aa$$

is not; both productions $S \to AAS$ and $A \to aa$ violate the conditions of Definition 6.4. ■

Theorem 6.6 Any context-free grammar $G = (V, T, S, P)$ with $\lambda \notin L(G)$ has an equivalent grammar $\widehat{G} = \left(\widehat{V}, \widehat{T}, S, \widehat{P}\right)$ in Chomsky normal form.

Proof: Because of Theorem 6.5, we can assume without loss of generality that G has no λ-productions and no unit-productions. The construction of $\widehat{G}$ will be done in two steps.

Step 1: Construct a grammar $G_1 = (V_1, T, S, P_1)$ from G by considering all productions in P in the form

$$A \to x_1 x_2 \cdots x_n, \tag{6.5}$$

where each x_i is a symbol either in V or T. If $n = 1$ then x_1 must be a terminal since we have no unit-productions. In this case, put the production into P_1. If $n \geq 2$, introduce new variables B_a for each $a \in T$. For each production of P in the form (6.5) we put into P_1 the production

$$A \to C_1 C_2 \cdots C_n,$$

where

$$C_i = x_i \text{ if } x_i \text{ is in } V,$$

and

$$C_i = B_a \text{ if } x_i = a.$$

For every B_a we also put into P_1 the production

$$B_a \to a.$$

This part of the algorithm removes all terminals from productions whose right side has length greater than one, replacing them with newly introduced variables. At the end of this step we have a grammar G_1 all of whose productions have the form

$$A \to a, \tag{6.6}$$

or

$$A \to C_1 C_2 \cdots C_n, \tag{6.7}$$

where $C_i \in V_1$.

It is an easy consequence of Theorem 6.1 that

$$L(G_1) = L(G).$$

Step 2: In the second step, we introduce additional variables to reduce the length of the right sides of the productions where necessary. First we put all productions of the form (6.6) as well as all the productions of the form (6.7) with $n = 2$ into $\widehat{P}$. For $n > 2$, we introduce new variables $D_1, D_2, \ldots$ and put into $\widehat{P}$ the productions

$$\begin{aligned} A &\to C_1 D_1, \\ D_1 &\to C_2 D_2, \\ &\vdots \\ D_{n-2} &\to C_{n-1} C_n. \end{aligned}$$

Obviously, the resulting grammar $\widehat{G}$ is in Chomsky normal form. Repeated applications of Theorem 6.1 will show that $L(G_1) = L\left(\widehat{G}\right)$, so that

$$L\left(\widehat{G}\right) = L(G).$$

This somewhat informal argument can easily be made more precise. We will leave this to the reader. ■

Example 6.8 Convert the grammar with productions

$$\begin{aligned} S &\to ABa, \\ A &\to aab, \\ B &\to Ac, \end{aligned}$$

to Chomsky normal form.

As required by the construction of Theorem 6.6, the grammar does not have any λ-productions or any unit-productions.

In Step 1, we introduce new variables B_a, B_b, B_c and use the algorithm to get

$$\begin{aligned} S &\to ABB_a, \\ A &\to B_a B_a B_b, \\ B &\to AB_c, \\ B_a &\to a, \\ B_b &\to b, \\ B_c &\to c. \end{aligned}$$

In the second step, we introduce additional variables to get the first two productions into normal form and we get the final result

$$\begin{aligned} S &\to AD_1, \\ D_1 &\to BB_a, \\ A &\to B_aD_2, \\ D_2 &\to B_aB_b, \\ B &\to AB_c, \\ B_a &\to a, \\ B_b &\to b, \\ B_c &\to c. \end{aligned}$$

■

Greibach Normal Form

Another useful grammatical form is the **Greibach normal form.** Here we put restrictions not on the length of the right sides of a production, but on the positions in which terminals and variables can appear. Arguments justifying Greibach normal form are a little complicated and not very transparent. Similarly, constructing a grammar in Greibach normal form equivalent to a given context-free grammar is tedious. We therefore deal with this matter very briefly. Nevertheless, Greibach normal form has many theoretical and practical consequences.

Definition 6.5

A context-free grammar is said to be in Greibach normal form if all productions have the form

$$A \to ax,$$

where $a \in T$ and $x \in V^*$.

If we compare this with Definition 5.4, we see that the form $A \to ax$ is common to both Greibach normal form and s-grammars, but Greibach normal form does not carry the restriction that the pair (A, a) occur at most once. This additional freedom gives Greibach normal form a generality not possessed by s-grammars.

If a grammar is not in Greibach normal form, we may be able to rewrite it in this form with some of the techniques encountered above. Here are two simple examples.

Example 6.9 The grammar

$$S \to AB,$$
$$A \to aA\,|bB|\,b,$$
$$B \to b$$

is not in Greibach normal form. However, using the substitution given by Theorem 6.1, we immediately get the equivalent grammar

$$S \to aAB\,|bBB|\,bB,$$
$$A \to aA\,|bB|\,b,$$
$$B \to b,$$

which is in Greibach normal form. ■

Example 6.10 Convert the grammar

$$S \to abSb|aa,$$

into Greibach normal form.

Here we can use a device similar to the one introduced in the construction of Chomsky normal form. We introduce new variables A and B that are essentially synonyms for a and b, respectively. Substituting for the terminals with their associated variables leads to the equivalent grammar

$$S \to aBSB|aA,$$
$$A \to a,$$
$$B \to b,$$

which is in Greibach normal form. ■

In general, though, neither the conversion of a given grammar to Greibach normal form nor the proof that this can always be done are simple matters. We introduce Greibach normal form here because it will simplify the technical discussion of an important result in the next chapter. However, from

a conceptual viewpoint, Greibach normal form plays no further role in our discussion, so we only quote the following general result without proof.

Theorem 6.7 For every context-free grammar G with $\lambda \notin L(G)$, there exists an equivalent grammar $\widehat{G}$ in Greibach normal form.

EXERCISES

1. Provide the details of the proof of Theorem 6.6.
2. Convert the grammar $S \to aSb|ab$ into Chomsky normal form.
3. Transform the grammar $S \to aSaA|A, A \to abA|b$ into Chomsky normal form.
4. Transform the grammar with productions

$$S \to abAB,$$
$$A \to bAB|\lambda,$$
$$B \to BAa\,|A|\,\lambda,$$

into Chomsky normal form.

5. Convert the grammar

$$S \to AB|aB$$
$$A \to aab|\lambda$$
$$B \to bbA$$

into Chomsky normal form. ●

6. Let $G = (V, T, S, P)$ be any context-free grammar without any λ-productions or unit-productions. Let k be the maximum number of symbols on the right of any production in P. Show that there is an equivalent grammar in Chomsky normal form with no more than $(k-1)\,|P| + |T|$ production rules.
7. Draw the dependency graph for the grammar in Exercise 4.
8. A linear language is one for which there exists a linear grammar (for a definition, see Example 3.13). Let L be any linear language not containing λ. Show that there exists a grammar $G = (V, T, S, P)$ all of whose productions have one of the forms

$$A \to aB,$$
$$A \to Ba,$$
$$A \to a,$$

where $a \in T, A, B \in V$, such that $L = L(G)$. ●

9. Show that for every context-free grammar $G = (V, T, S, P)$ there is an equivalent one in which all productions have the form

$$A \to aBC,$$

or

$$A \to \lambda,$$

where $a \in \Sigma \cup \{\lambda\}, A, B, C \in V$.

10. Convert the grammar

$$S \to aSb|bSa|a|b$$

into Greibach normal form.

11. Convert the following grammar into Greibach normal form.

$$S \to aSb|ab.$$

12. Convert the grammar

$$S \to ab\,|aS|\,aaS$$

into Greibach normal form.

13. Convert the grammar

$$S \to ABb|a,$$
$$A \to aaA|B,$$
$$B \to bAb$$

into Greibach normal form.

14. Can every linear grammar be converted to a form in which all productions look like $A \to ax$, where $a \in T$ and $x \in V \cup \{\lambda\}$?

15. A context-free grammar is said to be in two-standard form if all production rules satisfy the following pattern

$$A \to aBC,$$
$$A \to aB,$$
$$A \to a,$$

where $A, B, C \in V$ and $a \in T$.

Convert the grammar $G = (\{S, A, B, C\}, \{a, b\}, S, P)$ with P given as

$$S \to aSA,$$
$$A \to bABC,$$
$$B \to b,$$
$$C \to aBC,$$

into two-standard form.

★**16.** Two-standard form is general; for any context-free grammar G with $\lambda \notin L(G)$, there exists an equivalent grammar in two-standard form. Prove this.

6.3 A Membership Algorithm for Context-Free Grammars*

In Section 5.2, we claimed, without any elaboration, that membership and parsing algorithms for context-free grammars exist that require approximately $|w|^3$ steps to parse a string w. We are now in a position to justify this claim. The algorithm we will describe here is called the CYK algorithm, after its originators J. Cocke, D. H. Younger, and T. Kasami. The algorithm works only if the grammar is in Chomsky normal form and succeeds by breaking one problem into a sequence of smaller ones in the following way. Assume that we have a grammar $G = (V, T, S, P)$ in Chomsky normal form and a string

$$w = a_1 a_2 \cdots a_n.$$

We define substrings

$$w_{ij} = a_i \cdots a_j,$$

and subsets of V

$$V_{ij} = \left\{ A \in V : A \stackrel{*}{\Rightarrow} w_{ij} \right\}.$$

Clearly, $w \in L(G)$ if and only if $S \in V_{1n}$.

To compute V_{ij}, observe that $A \in V_{ii}$ if and only if G contains a production $A \to a_i$. Therefore, V_{ii} can be computed for all $1 \leq i \leq n$ by inspection of w and the productions of the grammar. To continue, notice that for $j > i$, A derives w_{ij} if and only if there is a production $A \to BC$, with $B \stackrel{*}{\Rightarrow} w_{ik}$ and $C \stackrel{*}{\Rightarrow} w_{k+1j}$ for some k with $i \leq k, k < j$. In other words,

$$V_{ij} = \bigcup_{k \in \{i, i+1, \ldots, j-1\}} \{A : A \to BC, \text{ with } B \in V_{ik}, C \in V_{k+1,j}\}. \quad (6.8)$$

An inspection of the indices in (6.8) shows that it can be used to compute all the V_{ij} if we proceed in the sequence

1. Compute $V_{11}, V_{22}, \ldots, V_{nn}$
2. Compute $V_{12}, V_{23}, \ldots, V_{n-1,n}$
3. Compute $V_{13}, V_{24}, \ldots, V_{n-2,n}$

and so on.

Example 6.11 Determine whether the string $w = aabbb$ is in the language generated by the grammar

$$\begin{aligned} S &\to AB, \\ A &\to BB|a, \\ B &\to AB|b, \end{aligned}$$

First note that $w_{11} = a$, so V_{11} is the set of all variables that immediately derive a, that is, $V_{11} = \{A\}$. Since $w_{22} = a$, we also have $V_{22} = \{A\}$ and, similarly,

$$V_{11} = \{A\}, V_{22} = \{A\}, V_{33} = \{B\}, V_{44} = \{B\}, V_{55} = \{B\}.$$

Now we use (6.8) to get

$$V_{12} = \{A : A \to BC, B \in V_{11}, C \in V_{22}\}.$$

Since $V_{11} = \{A\}$ and $V_{22} = \{A\}$, the set consists of all variables that occur on the left side of a production whose right side is AA. Since there are none, V_{12} is empty. Next,

$$V_{23} = \{A : A \to BC, B \in V_{22}, C \in V_{33}\},$$

so the required right side is AB, and we have $V_{23} = \{S, B\}$. A straightforward argument along these lines then gives

$$\begin{aligned} &V_{12} = \varnothing, V_{23} = \{S, B\}, V_{34} = \{A\}, V_{45} = \{A\}, \\ &V_{13} = \{S, B\}, V_{24} = \{A\}, V_{35} = \{S, B\}, \\ &V_{14} = \{A\}, V_{25} = \{S, B\}, \\ &V_{15} = \{S, B\}, \end{aligned}$$

so that $w \in L(G)$. ■

The CYK algorithm, as described here, determines membership for any language generated by a grammar in Chomsky normal form. With some additions to keep track of how the elements of V_{ij} are derived, it can be converted into a parsing method. To see that the CYK membership algorithm requires On^3 steps, notice that exactly $n(n+1)/2$ sets of V_{ij} have to be computed. Each involves the evaluation of at most n terms in (6.8), so the claimed result follows.

EXERCISES

1. Use the CYK algorithm to determine whether the strings *aabb*, *aabba*, and *abbbb* are in the language generated by the grammar in Example 6.11.
2. Use the CYK algorithm to find a parsing of the string *aab*, using the grammar of Example 6.11.
3. Use the approach employed in Exercise 2 to show how the CYK membership algorithm can be made into a parsing method.

★★ 4. Use the result in Exercise 3 to write a computer program for parsing with any context-free grammar in Chomsky normal form.