6.3 A Membership Algorithm for Context-Free Grammars*

In Section 5.2, we claim, 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 $V_{11}, V_{22}, ..., V_{nn}$,

2. Compute $V_{12}, V_{23}, ..., V_{n-1,n}$,

3. Compute $V_{13}, V_{24}, ..., V_{n-2,n}$,

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 $O\left(n^3\right)$ steps, notice that exactly $n\left(n+1\right)/2$ 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.