As $|wx| > 0$, $yw^{i}ux^{i}$ cannot equal $yw^{i}ux^{j}$ if $i \neq j$. Thus the grammar generates an infinite number of strings.

Conversely, suppose the graph has no cycles. Define the rank of a variable $A$ to be the length of the longest path in the graph beginning at $A$. The absence of cycles implies that the rank of $A$ is finite. We also observe that if $A \rightarrow BC$ is a production, then the rank of $B$ and $C$ must be strictly less than the rank of $A$, because for every path from $B$ or $C$, there is a path of length one greater from $A$. We show by induction on $r$ that if $A$ has rank $r$, then no terminal string derived from $A$ has length greater than $2^r$.

**Basis** $r = 0$. If $A$ has rank 0, then its vertex has no edges out. Therefore all $A$-productions have terminals on the right, and $A$ derives only strings of length 1.

**Induction** $r > 0$. If we use a production of the form $A \rightarrow a$, we may derive only a string of length 1. If we begin with $A \rightarrow BC$, then as $B$ and $C$ are of rank $r - 1$ or less, by the inductive hypothesis, they derive only strings of length $2^{r-1}$ or less. Thus $BC$ cannot derive a string of length greater than $2^r$.

Since $S$ is of finite rank $r_S$, and in fact, is of rank no greater than the number of variables, $S$ derives strings of length no greater than $2^r$. Thus the language is finite.

**Example 6.6** Consider the grammar

$$
S \rightarrow AB \\
A \rightarrow BC | a \\
B \rightarrow CC | b \\
C \rightarrow a
$$

whose graph is shown in Fig. 6.7(a). This graph has no cycles. The ranks of $S, A, B,$ and $C$ are 3, 2, 1, and 0, respectively. For example, the longest path from $S$ is $S \rightarrow BCB \rightarrow CCCB \Rightarrow CCCCB \Rightarrow aaaaa.$

If we add production $C \rightarrow AB$, we get the graph of Fig. 6.7(b). This new graph has several cycles, such as $A, B, C, A$. Thus we can find a derivation $A \Rightarrow x_1ABx_2$, in particular $A \Rightarrow BC \Rightarrow CCC \Rightarrow CABC$, where $x_1 = C$ and $x_2 = BC$. Since $C \Rightarrow a$ and $BC \Rightarrow b$, we have $A \Rightarrow aAb$. Then as $S \Rightarrow Ab$ and $A \Rightarrow a$, we now have $S \Rightarrow a(a(bu)b)$ for every $i$. Thus the language is infinite.

**Membership**

Another question we may answer is: Given a CFG $G = (V, T, P, S)$ and string $x$ in $T^*$, is $x$ in $L(G)$? A simple but inefficient algorithm to do so is to convert $G$ to $G' = (V', T, P, S)$, a grammar in Greibach normal form generating $L(G)$ - [$c$]. Since the algorithm of Theorem 4.3 tests whether $S \Rightarrow x$, we need not concern ourselves with the case $x = c$. Thus assume $x \neq c$, so $x$ is in $L(G')$ if and only if $x$ is in $L(G)$. Now, as every production of a GNF grammar adds exactly one terminal to the string being generated, we know that if $x$ has a derivation in $G'$, it has one with exactly $|x|$ steps. If no variable of $G'$ has more than $k$ productions, then there are at most $k^n$ leftmost derivations of strings of length $|x|$. We may try them all systematically.

However, the above algorithm can take time which is exponential in $|x|$. There are several algorithms known that take time proportional to the cube of $|x|$ or even a little less. The bibliographic notes discuss some of these. We shall here present a cubic time algorithm known as the Cocke-Younger-Kasami or CYK algorithm. It is based on the dynamic programming technique discussed in the solution to Exercise 3.23. Given $x$ of length $n \geq 1$, and a grammar $G$, which we may assume is in Chomsky normal form, determine for each $i$ and $j$ and for each variable $A$, whether $A \Rightarrow x_{ij}$, where $x_{ij}$ is the substring of $x$ of length $j$ beginning at position $i$.

We proceed by induction on $j$. For $j = 1$, $A \Rightarrow x_{ij}$ if and only if $A \Rightarrow x_j$ is a production, since $x_{ij}$ is a string of length 1. Proceeding to higher values of $j$, if $j > 1$, then $A \Rightarrow x_{ij}$ if and only if there is some production $A \rightarrow BC$ and some $k$, $1 \leq k < j$, such that $B$ derives the first $k$ symbols of $x_j$ and $C$ derives the last $j - k$ symbols of $x_j$. That is, $B \Rightarrow x_{ik}$ and $C \Rightarrow x_{i+k,j-k}$. Since $k$ and $j - k$ are both less than $j$, we already know whether each of the last two derivations exists. We may thus determine whether $A \Rightarrow x_{ij}$. Finally, when we reach $j = n$, we may determine whether $S \Rightarrow x_{nn}$. But $x_{nn} = x$, so $x$ is in $L(G)$ if and only if $S \Rightarrow x_{nn}$.

To state the CYK algorithm precisely, let $V_i$ be the set of variables $A$ such that $A \Rightarrow x_{ij}$. Note that we may assume $1 \leq i \leq n - j + 1$, for there is no string of length greater than $n - i + 1$ beginning at position $i$. Then Fig. 6.8 gives the CYK algorithm formally.

Steps (1) and (2) handle the case $j = 1$. As the grammar $G$ is fixed, step (2) takes a constant amount of time. Thus steps (1) and (2) take $O(n)$ time. The nested for-loops of lines (3) and (4) cause steps (5) through (7) to be executed at most $n^2$ times, since $i$ and $j$ range in their respective for-loops between limits that are at
begin
1) for i := 1 to n do
2) \( V_i := \{ A | A \rightarrow a \text{ is a production and the } i\text{th symbol of } x \text{ is } a \} \);
3) for j := 2 to n do
4) for i := 1 to n - j + 1 do
5) \( V_{ij} := \emptyset \);
6) for k := 1 to j - 1 do
7) \( V_{ij} := V_{ij} \cup \{ A | A \rightarrow BC \text{ is a production, } B \text{ is in } V_k \text{ and } C \text{ is in } V_{i+k,j-k} \} \)
end

Fig. 6.8. The CYK algorithm.

most \( n \) apart. Step (5) takes constant time at each execution, so the aggregate time spent at step (5) is \( O(n^3) \). The for-loop of line (6) causes step (7) to be executed \( n \) or fewer times. Since step (7) takes constant time, steps (6) and (7) together take \( O(n) \) time. As they are executed \( O(n^2) \) times, the total time spent in step (7) is \( O(n^3) \). Thus the entire algorithm is \( O(n^3) \).

Example 6.7 Consider the CFG

\[
S \rightarrow AB | BC \\
A \rightarrow BA | a \\
B \rightarrow CC | b \\
C \rightarrow AB | a
\]

and the input string \( baaba \). The table of \( V_{ij} \)'s is shown in Fig. 6.9. The top row is filled in by steps (1) and (2) of the algorithm in Fig. 6.8. That is, for positions 1 and 4, which are \( b \), we set \( V_{11} = V_{41} = \{ B \} \), since \( B \) is the only variable which derives \( b \).

Similarly, \( V_{21} = V_{31} = V_{51} = \{ A, C \} \), since only \( A \) and \( C \) have productions with \( a \) on the right.

To compute \( V_j \) for \( j > 1 \), we must execute the for-loop of steps (6) and (7). We must match \( V_k \) against \( V_{i+k,j-k} \) for \( k = 1, 2, \ldots, j - 1 \), seeking variable \( D \) in \( V_k \) and \( E \) in \( V_{i+k,j-k} \) such that \( DE \) is the right side of one or more productions. The left sides of these productions are adjoined to \( V_j \). The pattern in the table which corresponds to visiting \( V_k \) and \( V_{i+k,j-k} \) for \( k = 1, 2, \ldots, j - 1 \) in turn is to simultaneously move down column \( i \) and up the diagonal extending from \( V_j \) to the right, as shown in Fig. 6.10.

Fig. 6.10 Traversal pattern for computation of \( V_{ij} \).

For example, let us compute \( V_{24} \), assuming that the top three rows of Fig. 6.9 are filled in. We begin by looking at \( V_{21} = \{ A, C \} \) and \( V_{31} = \{ B \} \). The possible right-hand sides in \( V_{21} \) are \( AB \) and \( CB \). Only the first of these is actually a right side, and it is a right side of two productions \( S \rightarrow AB \) and \( C \rightarrow AB \). Hence we add \( S \) and \( C \) to \( V_{24} \). Next we consider \( V_{22} V_{12} = \{ B \} \{ S, A \} = \{ BS, BA \} \). Only \( BA \) is a right side, so we add the corresponding left side \( A \) to \( V_{24} \). Finally, we consider \( V_{23} V_{13} = \{ B \} \{ A, C \} = \{ BA, BC \} \). \( BA \) and \( BC \) are each right sides, with left sides \( A \) and \( S \), respectively. These are already in \( V_{24} \) so we have \( V_{24} = \{ S, A, C \} \). Since \( S \) is a member of \( V_{13} \), the string \( baaba \) is in the language generated by the grammar.

EXERCISES

6.1 Show that the following are not context-free languages
   a) \( \{ a^i b^j | i < j < k \} \)
   b) \( \{ a^i b^j | j = i^2 \} \)
   c) \( \{ a^i \} | i \text{ is a prime} \)
   d) the set of strings of \( a \)'s, \( b \)'s, and \( c \)'s with an equal number of \( a \), \( b \), and \( c \)
   e) \( \{ a^i b^i c^m | n \leq m \leq 2n \} \)

6.2 Which of the following are CFL's?
   a) \( \{ a^i b^j | i \neq j \text{ and } i \neq 2j \} \)
   b) \( \{ (a + b)^* - \{ (a + b)^n | n \geq 1 \} \}
   c) \( \{ w^m x^n | w \text{ is in } (a + b)^* \}
   d) \( \{ b_i \neq b_{i+1} | b_i \text{ is } i \text{ in binary, } i \geq 1 \} \)