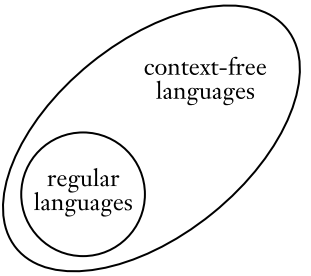


FIGURE **2.33**
Relationship of the regular and context-free languages

2.3

NON-CONTEXT-FREE LANGUAGES

In this section we present a technique for proving that certain languages are not context free. Recall that in Section 1.4 we introduced the pumping lemma for showing that certain languages are not regular. Here we present a similar pumping lemma for context-free languages. It states that every context-free language has a special value called the ***pumping length*** such that all longer strings in the language can be "pumped." This time the meaning of *pumped* is a bit more complex. It means that the string can be divided into five parts so that the second and the fourth parts may be repeated together any number of times and the resulting string still remains in the language.

THE PUMPING LEMMA FOR CONTEXT-FREE LANGUAGES

THEOREM **2.34**

Pumping lemma for context-free languages If A is a context-free language, then there is a number p (the pumping length) where, if s is any string in A of length at least p, then s may be divided into five pieces $s = uvxyz$ satisfying the conditions

1. for each $i \geq 0$, $uv^ixy^iz \in A$,
2. $|vy| > 0$, and
3. $|vxy| \leq p$.

When s is being divided into $uvxyz$, condition 2 says that either v or y is not the empty string. Otherwise the theorem would be trivially true. Condition 3

states that the pieces v, x, and y together have length at most p. This technical condition sometimes is useful in proving that certain languages are not context free.

PROOF IDEA Let A be a CFL and let G be a CFG that generates it. We must show that any sufficiently long string s in A can be pumped and remain in A. The idea behind this approach is simple.

Let s be a very long string in A. (We make clear later what we mean by "very long.") Because s is in A, it is derivable from G and so has a parse tree. The parse tree for s must be very tall because s is very long. That is, the parse tree must contain some long path from the start variable at the root of the tree to one of the terminal symbols at a leaf. On this long path, some variable symbol R must repeat because of the pigeonhole principle. As the following figure shows, this repetition allows us to replace the subtree under the second occurrence of R with the subtree under the first occurrence of R and still get a legal parse tree. Therefore, we may cut s into five pieces $uvxyz$ as the figure indicates, and we may repeat the second and fourth pieces and obtain a string still in the language. In other words, uv^ixy^iz is in A for any $i \geq 0$.

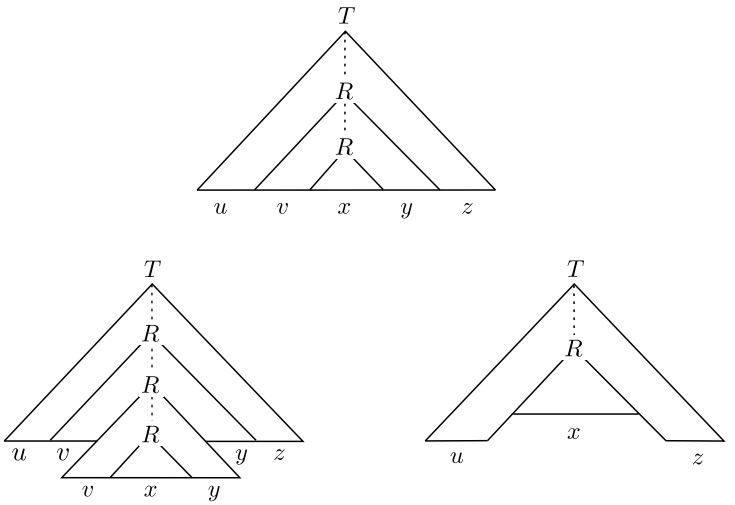


FIGURE 2.35
Surgery on parse trees

Let's now turn to the details to obtain all three conditions of the pumping lemma. We also show how to calculate the pumping length p.

PROOF Let G be a CFG for CFL A. Let b be the maximum number of symbols in the right-hand side of a rule (assume at least 2). In any parse tree using this grammar, we know that a node can have no more than b children. In other words, at most b leaves are 1 step from the start variable; at most b^2 leaves are within 2 steps of the start variable; and at most b^h leaves are within h steps of the start variable. So, if the height of the parse tree is at most h, the length of the string generated is at most b^h. Conversely, if a generated string is at least $b^h + 1$ long, each of its parse trees must be at least $h + 1$ high.

Say $|V|$ is the number of variables in G. We set p, the pumping length, to be $b^{|V|+1}$. Now if s is a string in A and its length is p or more, its parse tree must be at least $|V| + 1$ high, because $b^{|V|+1} \geq b^{|V|} + 1$.

To see how to pump any such string s, let τ be one of its parse trees. If s has several parse trees, choose τ to be a parse tree that has the smallest number of nodes. We know that τ must be at least $|V| + 1$ high, so its longest path from the root to a leaf has length at least $|V| + 1$. That path has at least $|V| + 2$ nodes; one at a terminal, the others at variables. Hence that path has at least $|V| + 1$ variables. With G having only $|V|$ variables, some variable R appears more than once on that path. For convenience later, we select R to be a variable that repeats among the lowest $|V| + 1$ variables on this path.

We divide s into $uvxyz$ according to Figure 2.35. Each occurrence of R has a subtree under it, generating a part of the string s. The upper occurrence of R has a larger subtree and generates vxy, whereas the lower occurrence generates just x with a smaller subtree. Both of these subtrees are generated by the same variable, so we may substitute one for the other and still obtain a valid parse tree. Replacing the smaller by the larger repeatedly gives parse trees for the strings uv^ixy^iz at each $i > 1$. Replacing the larger by the smaller generates the string uxz. That establishes condition 1 of the lemma. We now turn to conditions 2 and 3.

To get condition 2, we must be sure that v and y are not both ε. If they were, the parse tree obtained by substituting the smaller subtree for the larger would have fewer nodes than τ does and would still generate s. This result isn't possible because we had already chosen τ to be a parse tree for s with the smallest number of nodes. That is the reason for selecting τ in this way.

In order to get condition 3, we need to be sure that vxy has length at most p. In the parse tree for s the upper occurrence of R generates vxy. We chose R so that both occurrences fall within the bottom $|V| + 1$ variables on the path, and we chose the longest path in the parse tree, so the subtree where R generates vxy is at most $|V| + 1$ high. A tree of this height can generate a string of length at most $b^{|V|+1} = p$.

For some tips on using the pumping lemma to prove that languages are not context free, review the text preceding Example 1.73 (page 80) where we discuss the related problem of proving nonregularity with the pumping lemma for regular languages.

EXAMPLE 2.36

Use the pumping lemma to show that the language $B = \{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n | \ n \geq 0\}$ is not context free.

We assume that B is a CFL and obtain a contradiction. Let p be the pumping length for B that is guaranteed to exist by the pumping lemma. Select the string $s = \mathtt{a}^p\mathtt{b}^p\mathtt{c}^p$. Clearly s is a member of B and of length at least p. The pumping lemma states that s can be pumped, but we show that it cannot. In other words, we show that no matter how we divide s into $uvxyz$, one of the three conditions of the lemma is violated.

First, condition 2 stipulates that either v or y is nonempty. Then we consider one of two cases, depending on whether substrings v and y contain more than one type of alphabet symbol.

1. When both v and y contain only one type of alphabet symbol, v does not contain both a's and b's or both b's and c's, and the same holds for y. In this case, the string uv^2xy^2z cannot contain equal numbers of a's, b's, and c's. Therefore, it cannot be a member of B. That violates condition 1 of the lemma and is thus a contradiction.
2. When either v or y contains more than one type of symbol, uv^2xy^2z may contain equal numbers of the three alphabet symbols but not in the correct order. Hence it cannot be a member of B and a contradiction occurs.

One of these cases must occur. Because both cases result in a contradiction, a contradiction is unavoidable. So the assumption that B is a CFL must be false. Thus we have proved that B is not a CFL. ■

EXAMPLE 2.37

Let $C = \{\mathtt{a}^i\mathtt{b}^j\mathtt{c}^k | \ 0 \leq i \leq j \leq k\}$. We use the pumping lemma to show that C is not a CFL. This language is similar to language B in Example 2.36, but proving that it is not context free is a bit more complicated.

Assume that C is a CFL and obtain a contradiction. Let p be the pumping length given by the pumping lemma. We use the string $s = \mathtt{a}^p\mathtt{b}^p\mathtt{c}^p$ that we used earlier, but this time we must "pump down" as well as "pump up." Let $s = uvxyz$ and again consider the two cases that occurred in Example 2.36.

1. When both v and y contain only one type of alphabet symbol, v does not contain both a's and b's or both b's and c's, and the same holds for y. Note that the reasoning used previously in case 1 no longer applies. The reason is that C contains strings with unequal numbers of a's, b's, and c's as long as the numbers are not decreasing. We must analyze the situation more carefully to show that s cannot be pumped. Observe that because v and y contain only one type of alphabet symbol, one of the symbols a, b, or c doesn't appear in v or y. We further subdivide this case into three subcases according to which symbol does not appear.

a. *The* a*'s do not appear.* Then we try pumping down to obtain the string $uv^0xy^0z = uxz$. That contains the same number of a's as s does, but it contains fewer b's or fewer c's. Therefore, it is not a member of C, and a contradiction occurs.

b. *The* b*'s do not appear.* Then either a's or c's must appear in v or y because both can't be the empty string. If a's appear, the string uv^2xy^2z contains more a's than b's, so it is not in C. If c's appear, the string uv^0xy^0z contains more b's than c's, so it is not in C. Either way, a contradiction occurs.

c. *The* c*'s do not appear.* Then the string uv^2xy^2z contains more a's or more b's than c's, so it is not in C, and a contradiction occurs.

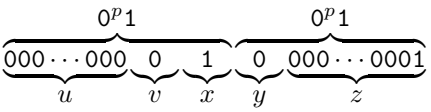
2. When either v or y contains more than one type of symbol, uv^2xy^2z will not contain the symbols in the correct order. Hence it cannot be a member of C, and a contradiction occurs.

Thus we have shown that s cannot be pumped in violation of the pumping lemma and that C is not context free. ∎

EXAMPLE 2.38

Let $D = \{ww \mid w \in \{0,1\}^*\}$. Use the pumping lemma to show that D is not a CFL. Assume that D is a CFL and obtain a contradiction. Let p be the pumping length given by the pumping lemma.

This time choosing string s is less obvious. One possibility is the string 0^p10^p1. It is a member of D and has length greater than p, so it appears to be a good candidate. But this string *can* be pumped by dividing it as follows, so it is not adequate for our purposes.



Let's try another candidate for s. Intuitively, the string $0^p1^p0^p1^p$ seems to capture more of the "essence" of the language D than the previous candidate did. In fact, we can show that this string does work, as follows.

We show that the string $s = 0^p1^p0^p1^p$ cannot be pumped. This time we use condition 3 of the pumping lemma to restrict the way that s can be divided. It says that we can pump s by dividing $s = uvxyz$, where $|vxy| \leq p$.

First, we show that the substring vxy must straddle the midpoint of s. Otherwise, if the substring occurs only in the first half of s, pumping s up to uv^2xy^2z moves a 1 into the first position of the second half, and so it cannot be of the form ww. Similarly, if vxy occurs in the second half of s, pumping s up to uv^2xy^2z moves a 0 into the last position of the first half, and so it cannot be of the form ww.

But if the substring vxy straddles the midpoint of s, when we try to pump s down to uxz it has the form $0^p1^i0^j1^p$, where i and j cannot both be p. This string is not of the form ww. Thus s cannot be pumped, and D is not a CFL. ∎