The Pumping Lemma for Context-Free Grammars

The grammar

$$
\begin{aligned}
S &\rightarrow AB \mid BC \\
A &\rightarrow BA \mid a \\
B &\rightarrow CC \mid b \\
C &\rightarrow AB \mid a
\end{aligned}
$$

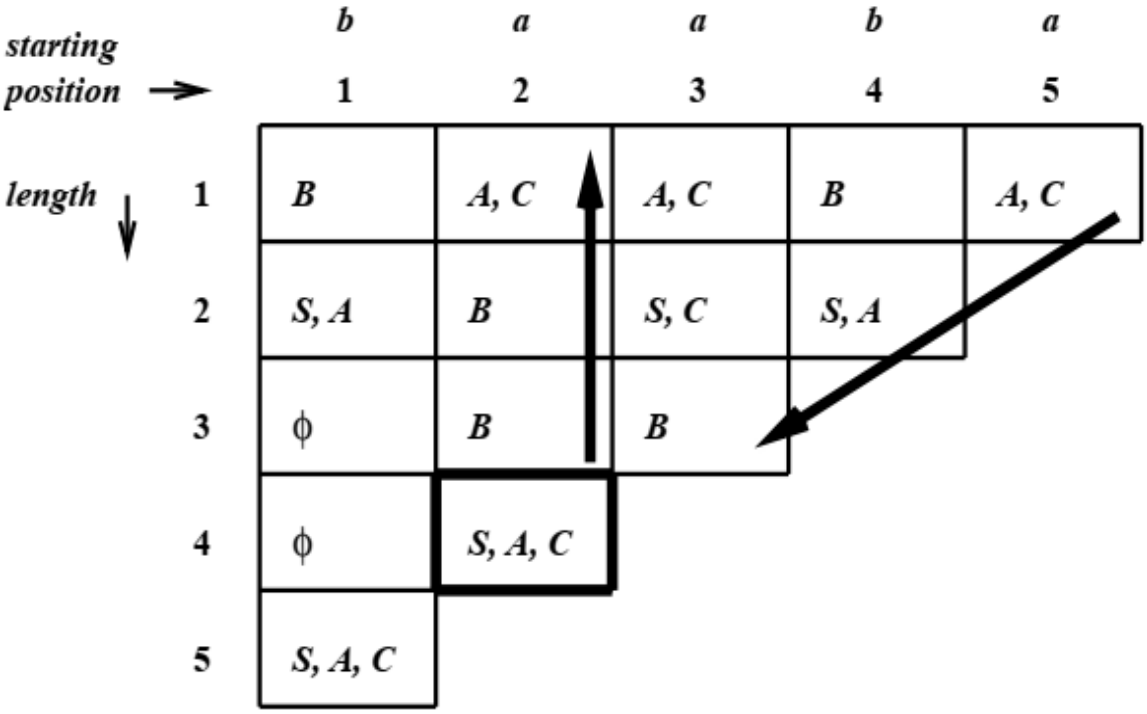

Figure 3.3: An example of "CYK" parsing

$$S \rightarrow 0\,S\,1 \mid 0\,1$$

generates the language

$$L_4 = \{0^n 1^n | n \geq 1\}.$$

(0^n means a string of n 0s.)

Similar grammars generate

$$L_5 = \{0^n 1^n 0^m 1^m | n, m \geq 1\},$$

$$L_6 = \{a^n b^m c^n | n, m \geq 1\},$$

$$L_7 = \{a^n b^m c^m d^n | n, m \geq 1\}.$$

If we think of 0, a, and b as opening parentheses and 1, c, and d as closing parentheses, then these language all are examples of balanced parentheses.[1] More complicated languages arise if we try to have equal numbers of not just two but three or more different characters, as for example in

$$L_8 = \{a^n b^n c^n : n \geq 1\},$$

and if we match numbers of characters beyond a nesting pattern, as for example in

$$L_9 = \{a^n b^m c^n d^m : n, m \geq 1\}.$$

If you try to write context-free grammars for L_8 and L_9, you sooner or later get frustrated trying to achieve the specified equalities between the exponents (i.e., the numbers of repetitions of characters) in the definitions of these languages. In fact,

Theorem 19 *L_8 and L_9 are not context-free.*

The languages L_8 and L_9 are of no practical interest. But after we develop a proof technique for showing that these languages are not context-free we will be able to use that technique to show that some aspects of programming languages are not context-free either.

The proof of Theorem 19 uses a "cutting-and-pasting" argument on the parse trees of context-free grammars. It is the most common argument for proving languages not to be context-free. The heart of the argument is that every context-free language has a certain "closure" property: The fact that certain strings are in the language forces other strings to be in the language as well. A

[1] L_7 is an example of nesting of different types of parentheses. Replace a and d by opening and closing parentheses and b and c by opening and closing square brackets. Then L_7 becomes $\{(^n[^m]^m)^n | n, m \geq 1\}$.

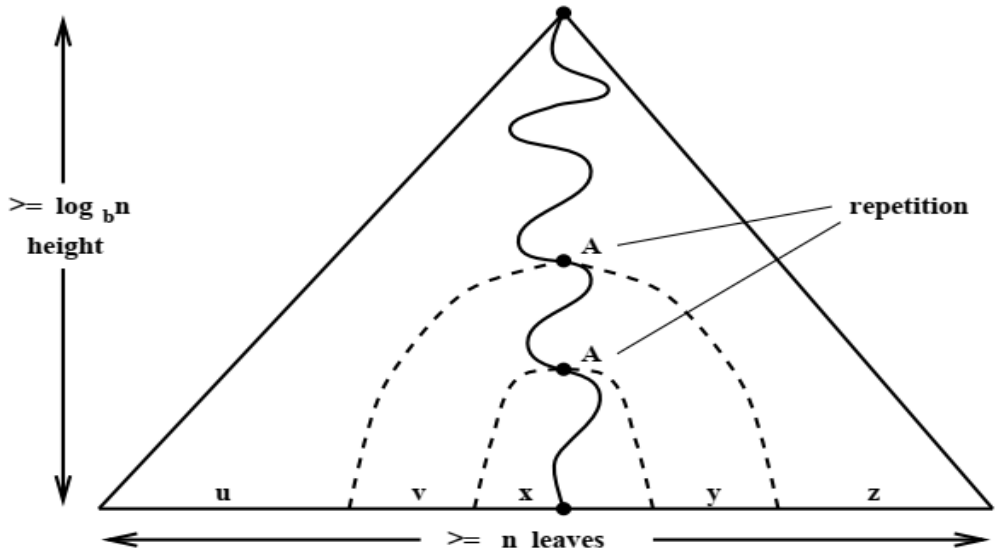

Figure 3.4: Illustrating the proof of Theorem 19

language that does not have the closure property cannot be context-free. The details of that closure property are developed next.

If a language L is context-free there is, by definition, a context-free grammar G that generates L, $L = L(G)$. Let b be the maximum length of right-hand sides of productions of G. In other words, b is a bound on the maximum number of children for nodes in parse trees of G. Then a parse tree with n leaves must be of height at least $\log_b n$, where height is defined to be 0 for a tree consisting of a single node. This means that a tree with n leaves must have a path from the root to a leaf with at least $\log_b n - 1$ interior nodes. What if n is so large that $\log_b n - 1$ exceeds the number of different left-hand sides of productions in G? (These are also called "syntactic variables.") Then some syntactic variable must show up at least twice along that path. Figure 3.4 illustrates this.

Pick such a variable, call it A, and pick two occurrences of A along one path. These two occurrences of A split the string w that the tree derives into five parts, labeled u, v, x, y, z in Figure 3.4. (The substring vxy is derived from the first of the two occurrences of the variable A, the substring x from the second occurrence.)

Knowing that we can derive x from A, the tree shown in the top half of Figure 3.5 is also a parse tree in this grammar. It derives the string uxz. There are other trees we can construct. For example, as illustrated in the bottom half of Figure 3.5, the second occurrence of A can be made to derive vxy, resulting in a tree that derives the string $uvvxyyz$. No need to stop there. We can build trees that derive $uvvvxyyyz$, $uvvvvxyyyyz$, in fact any string uv^ixy^iz for any $i \geq 2$.

For $i = 0, 1$, the string uv^ixy^iz becomes uxz and $uvxyz$, respectively. This lets us summarize the effect of all this cutting-and-pasting of trees as the fol-

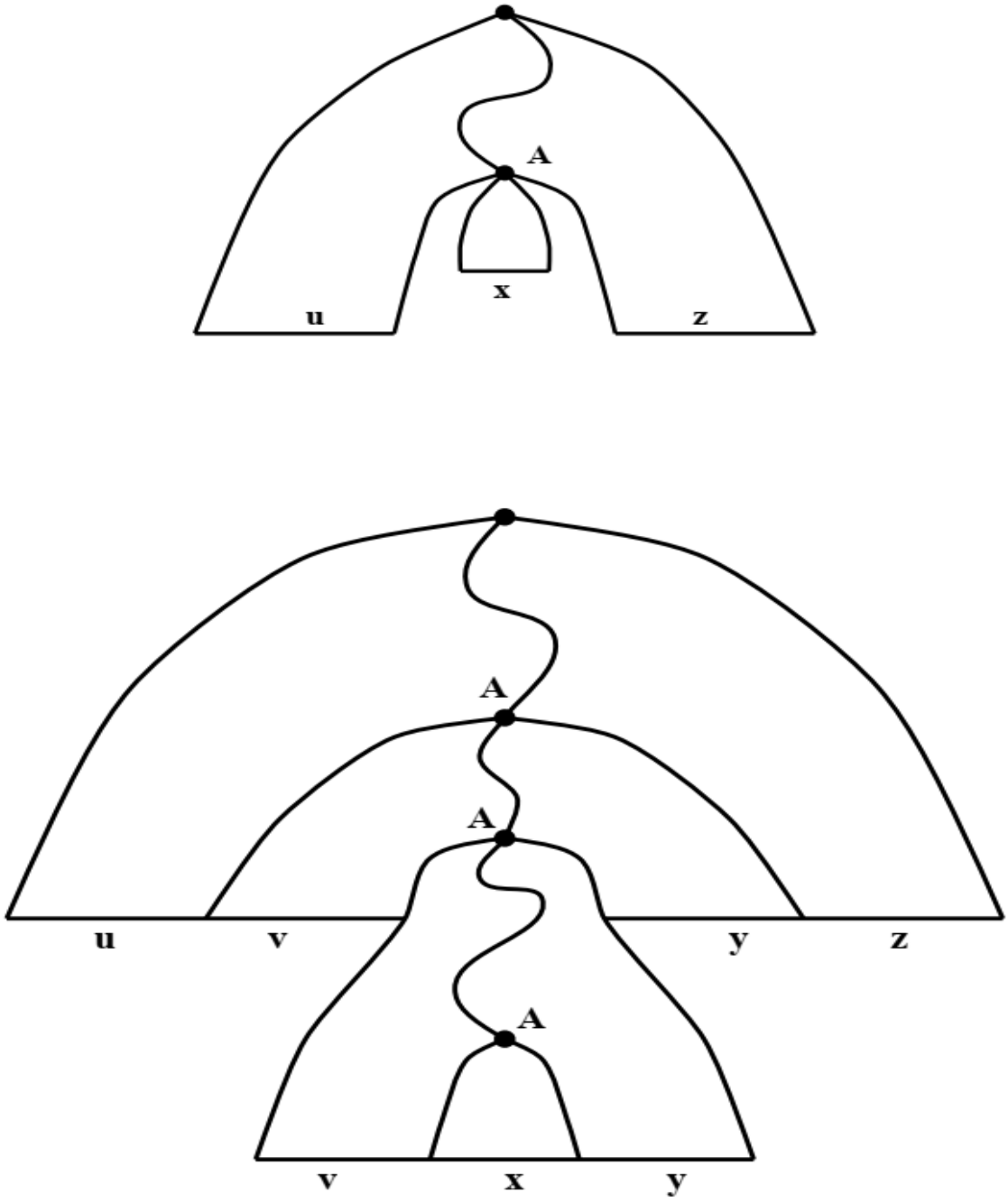

Figure 3.5: Deriving uxz and $uvvxyyz$

lowing

Lemma 4 (Pumping Lemma for Context-Free Languages, Version 1) *If L is an context-free language then there is a constant k such that for every string w in L of length k or more the following is true.*

There are five strings, u, v, x, y, z, such that

1. *$w = uvxyz$, and*
2. *for all $i \geq 0$, $uv^ixy^iz \in L$, and*
3. *$|vy| > 0$.*[2]

This lemma is strong enough to let us prove that the language $L_8 = \{a^n b^n c^n : n \geq 1\}$ from Theorem 19 is not context-free. (We will need to add one more refinement before we can handle L_9.)

Proof that L_8 is not context-free. Assuming it was. Then Lemma 4 applies. Let k be the constant said to exist in Lemma 4. Consider the string $w = a^k b^k c^k$. The length of w is k or more (in fact, $3k$), hence statements (1) — (3) from the Lemma apply, implying that it is possible to split w into five parts, $w = uvxyz$, such that $|vy| > 0$ and $uv^2xy^2z \in L_8$. But this is not true. No matter what five-way split of w one tries, uv^2xy^2z is never of the form of equal numbers of a's, b's, and c's following each other.[3] This contradiction shows that L_8 is not context-free. □

The Pumping Lemma can be stated as follows.

Why is this so messy?

$$L \text{ context-free}$$

$$\Longrightarrow$$

There is a constant k such that
for all strings $w \in L$ with $|w| \geq k$
there are strings u, v, x, y, z with $w = uvxyz$ and $|vy| > 0$ such that
for all $i \geq 0$, $uv^ixy^iz \in L$.

[2] This last part of the statement we did not prove. The proof is left as an exercise. The statement is saying that not both of v and y are empty.

[3] One can come up with a more detailed argument for this. The challenge is to cover all ways in which the strings could get divided up into five pieces. An added challenge is to come up with a *concise* argument. Here is an attempt:

The substring v cannot contain more than one type of character because otherwise v^2 contains characters in an order not permitted in L_8. The same is true for y. Hence at most two types of characters take part in the "pumping" from $uvxyz$ to uv^2xy^2z, which therefore does not contain equal numbers of all three characters anymore.

What makes such a statement hard to digest are the alternations of "*for all*" with "*there is/are*," akin to the nesting of loops in a program.[4]

The above statement is of the form "L *context-free* $\Rightarrow B$," where B is that complicated statement with four alternating quantifiers. Since we are trying to prove that some languages are not context-free, a more useful version of the statement is "L *not context-free* $\Leftarrow \neg B$." Spelled out in detail, this version reads

$$L \text{ not context-free}$$

$$\Longleftarrow$$

$$\begin{array}{c} \textit{For all constants } k \\ \textit{there is a string } w \in L \textit{ with } |w| \geq k \textit{ such that} \\ \textit{for all strings } u,\ v,\ x,\ y,\ z \textit{ with } w = uvxyz \textit{ and } |vy| > 0 \\ \textit{there is an } i \geq 0 \textit{ with } uv^i xy^i z \notin L. \end{array}$$

Thus, we can prove that some language L is not context-free by proving the above four-line statement $\neg B$: *"For all* $\ldots \notin L$*"*. This is what we did in the previous example. Here is another one.

Another Example Let's take the language that consists of strings like

```
1234+111=1345
10000+10000=20000
1020304+505=1020809
11111+22222=33333
```

The language consists of strings of the form $\alpha + \beta = \gamma$ that represent a correct equation between decimal integers.

This language L is not context-free, a fact we prove by proving the statement "$\neg B$" above.

How do we prove something *"for all constants* k*"*? By starting out the proof by saying "Let k be an arbitrary constant" and carrying the k along as a parameter of the argument.

What's next? We need to show that *there exists* a string w with certain properties. To show this, all we need to do is exhibit *one* such string. In this example, let's pick w to be the string

$$\underbrace{1\cdots1}_{k} + \underbrace{2\cdots2}_{k} = \underbrace{3\cdots3}_{k}$$

[4]This this called an "alternation of quantifiers." There are formal results that say that adding a level of alternation makes statements harder in the sense that even if we had a program that could determine the truth of statements with q alternations of quantifiers, this would not enable us to build a program that determines truth of statements with $q + 1$ quantifiers. (The concept of transformations is rearing its ugly head.) Look for the subject of "the arithmetic hierarchy" in textbooks on recursive function theory if you want to read about this.

(Note how the k plays its role.) What's the next step? We have to show that *for all* strings u, v, x, y, z which represent a five-way breakup of w ("$w = uvxyz$") and which don't have the second and fourth piece both empty ("$|vy| > 0$"), there is a constant i with $uv^ixy^iz \notin L$. This is the hard part of the proof because we somehow have to provide an argument that covers all possible such five-way breakups of w. It is tedious, but it can be done, by dealing with an exhaustive set of cases. Here is such an exhaustive set of cases for this example.

Case 1. Either v or y or both contain the + and/or the =. Then $uv^0xy^0z \notin L$ if for no other reason then because it does not have a + and an = in it.

Case 2. Neither v nor y contain the + or the =.

Case 2.1. Both v and y lie on the same side of the =. Then pumping (with any $i \neq 1$) changes that side of the equation but not the other, thus creating a string outside of L.

Case 2.2. v lies entirely on the left side of the equation and y lies entirely on the right side.

Case 2.2.1. One of v and y is empty. Then pumping changes one side of the equation but not the other, creating a string outside L.

Case 2.2.2. Neither v nor y is empty.

Case 2.2.2.1. v lies within the first number of the equation. Then uv^2xy^2z is of the form

$$\underbrace{1\cdots1}_{>k}+\underbrace{2\cdots2}_{k}=\underbrace{3\cdots3}_{>k}$$

which is not a correct equation.

Case 2.2.2.2. v lies within the second number of the equation. This case is analog to Case 2.2.2.1. uv^2xy^2z is of the form

$$\underbrace{1\cdots1}_{k}+\underbrace{2\cdots2}_{>k}=\underbrace{3\cdots3}_{>k}$$

which is not a correct equation.

An example that does not work

Going back to the language $L_9 = \{a^nb^mc^nd^m : n, m > 0\}$, if we tried to construct a proof that it is not context-free by constructing an exhaustive set of cases just like in the previous example, we would sooner or later realize that there is a case (or two) that does not work.

This is the case where v consists of a number of a's and y consists of the same number of c's. (The other case is the analogous situation with b's and d's.)

The proof attempt fails.[5] It takes a stronger version of the Pumping Lemma to prove that this language is not context-free.

[5]This is nothing but bad news. Failing to prove that L is not context-free does not prove that it is context-free either. It proves nothing.

A Stronger Version of the Pumping Lemma

In the proof of the Pumping Lemma, instead of merely arguing that there is a repetition of some syntactic variable along a path from the root to a leaf, we could instead argue that such a repetition can always be found "close to" a leaf. This then implies a limit on how long the string vxy can be. Here are the details.

If a grammar has s different syntactic variables and no right-side of a production is longer than b, then a repetition of a syntactic variable occurs within $s+1$ steps along the path from a leaf to the root and the subtree under the second occurrence of the variable (second meaning the one further away from the leaf) has at most $k' = b^{s+1}$ leaves. This number is a constant that depends only on the grammar. (Draw a picture.) This improvement in the proof of the Pumping Lemma makes it easier to prove that a language L is context-free. We only have to show that

$$\begin{gathered}\textit{For all constants } k \\ \textit{there is a string } w \in L \textit{ with } |w| \geq k \textit{ such that} \\ \textit{for all strings } u,\ v,\ x,\ y,\ z \textit{ with } w = uvxyz \textit{ and } |vy| > 0 \textit{ and } |vxy| \leq k \\ \textit{there is an } i \geq 0 \textit{ with } uv^i xy^i z \notin L.\end{gathered}$$

(There is a little subtlety in this statement in that the old constant k and the new constant k' were combined into a single constant, which is the maximum of the two and called again k.)

Now it works

The only difference to the old version is the additional condition that "$|vxy| \leq k$." This is very helpful, though, because it means that cases in which "$|vxy| > k$" need not be considered. This takes care of the failed cases in the unsuccessful proof attempt above, because for v to consist of a's only and y of c's only (or b's and d's, respectively) the string vxy would have had to stretch over all the b's (or the c's) and hence be longer than k.