Solution to 7.4.1

November 11, 2003

Part (a)

If there is any string at all that can be “pumped,” then the language is infinite. Thus, let \( n \) be the pumping-lemma constant. If there are no strings as long as \( n \), then surely the language is finite. However, how do we tell if there is some string of length \( n \) or more? If we had to consider all such strings, we’d never get done, and that would not give us a decision algorithm.

The trick is to realize that if there is any string of length \( n \) or more, then there will be one whose length is in the range \( n \) through \( 2n-1 \), inclusive. For suppose not. Let \( z \) be a string that is as short as possible, subject to the constraint that \( |z| \geq n \). If \( |z| < 2n \), we are done; we have found a string in the desired length range. If \( |z| \geq 2n \), use the pumping lemma to write \( z = uvwxy \). We know \( uwy \) is also in the language, but because \( |uvx| \leq n \), we know \( |z| > |uwy| \geq n \). That contradicts our assumption that \( z \) was as short as possible among strings of length \( n \) or more in the language.

We conclude that \( |z| < 2n \). Thus, our algorithm to test finiteness is to test membership of all strings of length between \( n \) and \( 2n-1 \). If we find one, the language is infinite, and if not, then the language is finite.

Part (b)

As in (a), if there is a string of length more than or equal to \( n \), where \( n \) is the pumping-lemma constant, then the language is infinite, and there is clearly more than 100 strings. So we first test whether the language is infinite by the method given in (a). If it is not infinite, we simply enumerate all the strings that have length less than \( n \), do membership test on each of them, and count the number of strings that belong to the language. Finally check whether it is more than 100.

Some Comment

Another common idea is to do cycle detection on some graph. However, one need to be cautious:

- It is not correct to restrict the grammar to be linear, since CFL does not necessarily have linear grammar. It is Regular Language that has left linear or right linear grammar.

- Since it is not necessarily linear grammar, it is a wrong assumption that a path in the constructed graph is corresponding to a string.