Solution to 5.1.1

October 15, 2003

Part (a)

\[ S \rightarrow 0S1|01 \]

It is the simplest one in this problem. The only thing that needs attention is that we have \( n \geq 1 \) in the description, so the string is unable to be \( \epsilon \).

Part (b)

\[
\begin{align*}
S & \rightarrow AB|CD \\
A & \rightarrow aA|\epsilon \\
B & \rightarrow bBc|E|cD \\
C & \rightarrow aCb|E|aA \\
D & \rightarrow cD|\epsilon \\
E & \rightarrow bE|b 
\end{align*}
\]

To understand how this grammar works, observe the following:

- \( A \) generates zero or more \( a \)'s.
- \( D \) generates zero or more \( c \)'s.
- \( E \) generates one or more \( b \)'s.
- \( B \) first generates an equal number of \( b \)'s and \( c \)'s, then produces either one or more \( b \)'s (via \( E \)) or one or more \( c \)'s (via \( cD \)). That is, \( B \) generates strings in \( b^*c^* \) with an unequal number of \( b \)'s and \( c \)'s.
- Similarly, \( C \) generates unequal numbers of \( a \)'s then \( b \)'s.
- Thus, \( AB \) generates strings in \( a^*b^*c^* \) with an unequal numbers of \( b \)'s and \( c \)'s, while \( CD \) generates strings in \( a^*b^*c^* \) with an unequal number of \( a \)'s and \( b \)'s.

This problem is not quite hard. What we need to do is just consider different cases, say

- More \( a \)'s than \( b \)'s.
- More \( b \)'s than \( a \)'s.
- More \( b \)'s than \( c \)'s.
- More \( c \)'s than \( b \)'s.

and give CFG for each of them.

A rare number of people forgot to let the strings with unequal numbers of \( a \)'s and \( b \)'s have whatever number of \( c \)'s.
Part (c)

\[
\begin{align*}
S & \rightarrow AB|BA|T \\
T & \rightarrow aTb|bTa|aTb|bTb|a|b \\
A & \rightarrow aAb|bAa|aAa|bAb|a \\
B & \rightarrow aBb|bBa|aBa|bBb|b
\end{align*}
\]

There are two cases for a string not of the form \(ww\):

- \(S\) has odd number of symbols. It is what \(T\) in the CFG stands for.
- \(S\) consists of two parts, each of which has odd number of symbols. The “center” symbol in the first part is different than that in the second part.

The second case is for the strings with even number of symbols. Suppose a string \(S\) is of length \(2l\). There has to be a number \(k\), s.t. \(1 \leq k \leq l\) and \(S(k) \neq S(l+k)\). (Here I use \(S(i)\) to denote the \(i\)-th symbol in the string \(S\)). Then we can divide this string \(S\) into two parts,

- \(S_1\), which is the sub-string of \(S\) centered at \(k\)-th symbol and has length of \(2k-1\)
- \(S_2\), which is the remaining part of \(S\), centered at \((l+k)\)-th symbol and has length \(2(l-k)+1\)

So \(S = S_1S_2\), and \(S_1, S_2\) has different symbols in the center. \(A\) and \(B\) in the CFG stand for \(S_1\) and \(S_2\).

Common errors are:

- Only considering one case (odd length or even length).
- Only including symmetric strings while constructing \(A\) and \(B\), that is, \(A \rightarrow aAa|bAb|a\).

Part (d)

There are several kinds of solutions that work very well. The thoughts behind the solutions generally fall into two categories:

- It is incorrect to assume that the string can be described by simply giving the sub-strings at the beginning and the end, such as, \(S \rightarrow 00S1|0S01|01S0\ldots\). However we can show that the whole string can be decomposed into several (we don’t know how many, when giving the CFG) sub-strings, each of which has simple prefix and postfix. A correct solution of this kind might be

  \[
  S \rightarrow \varepsilon|SS|00S1|1S00|01S0
  \]

Notice that \(S \rightarrow SS\) has the function that decompose the original string into several simpler sub-strings.

- Another way is to identify some “separating” symbols, and every sub-strings in between are those with twice as many 0’s as 1’s. We can show that the “separating” symbols always exist. The corresponding answer can be

  \[
  S \rightarrow \varepsilon|0S0S1S|0S1S0S|1S0S0S
  \]

We have left some unproved claims. Here is a technique that you can make use of it to argue those claims:
Consider a string \(S\) of length \(n\) as a sequence of numbers \(\{a_i\}_{i=1}^{n}\), in which the symbol “1” is treated as a number \(+1\), and the symbol “0” is treated as a number \(-\frac{1}{2}\). Then define partial summation

\[
p_k = \sum_{i=1}^{k} a_i
\]

2
It is easy to find out that $p_0 = 0$ and $p_n = 0$. We also know that for any consecutive partial summation $p_i$ and $p_{i+1}$, there are only two possibilities: $p_{i+1} = p_i + 1$ or $p_{i+1} = p_i - \frac{1}{2}$, which means that there is no sudden change in the sequence of partial summations $\{p_i\}_{i=0,\ldots,n}$. By observing how this sequence $p_i$ goes up and down and when it can get to 0, we are able to figure out the proof of our solutions.

Common errors are:

- Only giving several possible prefix and postfix. Such as $S = 0S01|0S10|1S00|\ldots$. A good counterexample to these answers is “000111000”.

- Not including $\epsilon$ into the CFG.