We start by finishing the sample construction started in class on 11/21/2017, converting a PDA into an equivalent grammar. Recall that $P = (\{p, q\}, \{a, b\}, \{Z\}, \delta, p, Z, \emptyset)$ is a two-state PDA where the only allowed (i.e., nonempty) transitions are

$\delta(p, a, Z) = \{(p, ZZ), (q, \epsilon)\}$,

$\delta(q, b, Z) = \{(q, \epsilon)\}$,

$\delta(q, \epsilon, Z) = \{(q, \epsilon)\}$.

(Just for simplicity, I’ll use $Z$ as the stack symbol instead of $Z_0$.) You might also recall that $N(P) = \{a^m b^n \mid m > n \geq 0\}$. Using the construction in the book or in class, we started constructing a CFG $G$ such that $L(G) = N(P)$. The construction given in class (which is simpler than the textbook’s) dictates that $G$ have two productions involving the start symbol $S$:

$$S \rightarrow [pZp] \mid [pZq].$$

The rest of the productions arise from $P$’s transitions:

<table>
<thead>
<tr>
<th>transition</th>
<th>production(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(p, ZZ) \in \delta(p, a, Z)$</td>
<td>$[pZp] \rightarrow a[pZp][pZp] \mid a[pZq][qZp]$</td>
</tr>
<tr>
<td></td>
<td>$[pZq] \rightarrow a[pZp][pZq] \mid a[pZq][qZq]$</td>
</tr>
<tr>
<td>$(q, \epsilon) \in \delta(p, a, Z)$</td>
<td>$[pZq] \rightarrow a$</td>
</tr>
<tr>
<td>$(q, \epsilon) \in \delta(q, b, Z)$</td>
<td>$[qZq] \rightarrow b$</td>
</tr>
<tr>
<td>$(q, \epsilon) \in \delta(q, \epsilon, Z)$</td>
<td>$[qZq] \rightarrow \epsilon$</td>
</tr>
</tbody>
</table>

Let’s rename the variables: $A := [pZp], B := [pZq], C := [qZp]$, and $D := [qZq]$. Then after rearranging, the productions are

$$S \rightarrow A \mid B$$

$$A \rightarrow aAA \mid aBC$$

$$B \rightarrow aAB \mid aBD \mid a$$

$$D \rightarrow b \mid \epsilon$$

Now to simplify $G$. Since there are no $C$-productions (because you can never get from state $q$ back to state $p$), the second $A$-production is useless, and we can just remove it. But then the only $A$-production is $A \rightarrow aAA$, so that once an $A$ shows up in a derivation, you can never get rid of
it, and that means that all productions involving $A$ cannot derive any strings of terminals, and so they can all be removed. Here is the resulting equivalent grammar:

$$
S \rightarrow B \\
B \rightarrow aBD \mid a \\
D \rightarrow b \mid \varepsilon
$$

Two more simplifications are possible: (1) since $S$ only occurs in the production $S \rightarrow B$, we can just remove that production and make $B$ the start symbol; (2) $D$ can only derive $b$ or $\varepsilon$, so we can remove those two $D$-productions while substituting in $b$ or $\varepsilon$ in turn for $D$ in the body of the first $B$-production. The result is a grammar with one variable and three productions:

$$
B \rightarrow aBb \mid aB \mid a,
$$

and this is an ambiguous grammar for $N(P)$.

**Exercises**

The last two exercises, marked “optional,” mostly build on one another and on previous exercises. The last four exercises should be considered in order.

1. Show that the grammar $B \rightarrow BB \mid (B) \mid \varepsilon$ of the last problem is ambiguous by giving two different parse trees for the same string. What is the shortest string you can find with two or more parse trees in this case? [Hint: This may be easier than you think.]

2. The grammar

$$
S \rightarrow (S)S \mid \varepsilon
$$

generates all strings of balanced parentheses and is unambiguous.

   (a) Give parse trees for the strings $()()$, $(()())$, and $()((()))$.

   (b) Show that every string derivable from this grammar is a string of balanced parentheses. 

   *Hint:* Do an induction on the length of a derivation, or else do an induction on the size of a parse tree.

3. Do Exercise 5.3.2.

4. Do Exercise 5.4.7. In part (a), they say “derivation tree” when they meant to say “parse tree.” Part (b) is optional.

5. Do Problem 5.4 on page 221.

6. Do Problem 5.7 on page 221. Ignore the last sentence.

7. Consider the following grammar for statements in a programming language:

$$
S \rightarrow a \mid iSeS \mid \{L\} \mid wS \\
L \rightarrow S \mid L;S
$$

Here, $S$ stands for “statement,” $L$ for “list of statements,” $a$ for “assignment,” $i$ for “if,” $e$ for “else,” and $w$ for “while.” Give a parse tree for the string, $\{a; i\{a; a\}e; w; a\}$. 
8. (Optional) Find an unambiguous grammar for the language of all binary strings with the same number of 0’s as 1’s.

9. Do Exercise 6.1.1(a,b,c). (Part (a) has a solution on the book’s website.)

10. Do Exercise 6.2.1(a,b,c). (Part (a) has a solution on the book’s website.)

11. Do Exercise 6.2.5(a,b,c,d).

12. Do Exercise 6.3.2. (Notice that Exercise 6.3.1 is similar and has its solution on the textbook’s website.)

13. Convert the PDA $P = (\{q\}, \{0, 1\}, \{X, Z\}, \delta, q, Z_0)$ to an equivalent CFG, if $\delta$ is given by

   (a) $\delta(q, 0, Z_0) = \{(q, XZ_0)\}$.
   (b) $\delta(q, 0, X) = \{(q, XX)\}$.
   (c) $\delta(q, 1, X) = \{(q, \epsilon)\}$.
   (d) $\delta(q, \epsilon, Z_0) = \{(q, \epsilon)\}$.

   This is similar to Exercise 6.3.3, which has a solution on the textbook’s website. You may use either the method of the book or the method I described in class on 11/21/2017 (see the sample construction at the top of this handout. (Note that this PDA satisfies the strict definition of a restricted PDA I gave on 11/21.)

14. Do Exercise 8.2.1(a,b,c).

15. Do Exercise 8.2.2(b).

16. Do Exercise 8.2.3(a,b).

17. (Optional) Do Exercise 8.2.5(c).

18. Do Exercise 8.3.2.

19. Do Exercise 8.1.1(a). This has a solution on the book’s website.

20. Do Exercise 8.2.4(a,b). This exercise has a complete solution on the book’s website.

21. Let $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ be a Turing machine. As usual, we assume that $Q \cap \Gamma = \emptyset$. Recall that an instantaneous description (ID) of $M$ is any string over the alphabet $Q \cup \Gamma$ containing exactly one symbol from $Q$, and this symbol is not the last symbol in the string. Give a DFA that recognizes the language of all IDs of $M$.

22. Let $M$ be as in the last problem. Recall that if ID$_1$ and ID$_2$ are IDs of $M$, then ID$_1 \vdash$ ID$_2$ means that ID$_2$ results from ID$_1$ by a single step of $M$. Let $\$ be some symbol not in $Q \cup \Gamma$. The languages

   $$L_1 := \{w\$x^R \mid w \text{ and } x \text{ are IDs of } M \text{ and } w \vdash x\}$$
   $$L_2 := \{w^R\$x \mid w \text{ and } x \text{ are IDs of } M \text{ and } w \vdash x\}$$

   are both context-free. (Recall that $x^R$ and $w^R$ are the reversals of strings $x$ and $w$, respectively.) Describe how to build CFGs for $L_1$ and $L_2$, given a complete description of $M$. 

3
23. (Optional) Let $M$ and $\$$ be as in the last problem. Describe a PDA $P$ that, given as input some string of the form $w\$$x, where $w$ and $x$ are IDs of $M$, accepts if and only if $w \not\vdash x$. In other words, $P$ accepts iff it finds a “mistake” in $M$’s transition from $w$ to $x$. If you want, you may assume that the IDs $w$ and $x$ cover the exact same portion of $M$’s tape, but this assumption is not necessary. You can stick to a high-level description of $P$ instead of a formal one.

24. (Optional) Prove that there is no algorithm that decides, given a PDA $P$, whether $P$ accepts all strings over its input alphabet. Hint: Given a TM $M$ as in the last problem and an input string $x$, design a PDA $P$ that accepts a string $w$ if and only if $w$ is not of the form $ID_0\$$ID_1\$$\cdots\$$ID_n$, where $ID_0 \vdash ID_1 \vdash \cdots \vdash ID_n$ is a complete trace of a halting computation of $M$ on input $x$. ($P$ can be constructed using ideas from the last problem.) Then $P$ accepts all strings if and only if $M$ does not halt on input $x$. Conclude that any decision procedure for the former statement can be used to decide the latter, which we know is undecidable.