CS 162 **Fall 2015**

Homework 5 Solutions

October 30, 2015 *Timothy Johnson*

1. Exercise 6.3.2 on page 251 of Hopcroft et al.
   Convert the grammar

$$\begin{aligned} S &\rightarrow 0S1 \mid A \\ A &\rightarrow 1A0 \mid S \mid \epsilon \end{aligned}$$

   to a PDA that accepts the same language by empty stack.

   We follow the construction given in the textbook on page 244. We let P have the single state $\{q\}$, input alphabet $\{0,1\}$, stack alphabet $\{0,1,S,A\}$, transition function δ, start state $\{q\}$, and start symbol S.

   We need to build all of the productions into our transition function, so we add the following transitions for our variables.

   - $\delta(q,\epsilon,S) = \{(q,0S1),(q,A)\}$
   - $\delta(q,\epsilon A) = \{(q,1A0),(q,S)\}$

   We also need to be able to match terminals against our input string, so we add a transition for each terminal.

   - $\delta(q,0,0) = \{(q,\epsilon)\}$
   - $\delta(q,1,1) = \{q,\epsilon)\}$

   These rules define a complete new PDA that accepts the same language as our grammar.

2. Exercise 6.3.4 on page 252 of Hopcroft et al.
   Convert the PDA of Exercise 6.1.1 to a context-free grammar.

   $P = (\{q,p\},\{0,1\},\{Z_0,X\},\delta,q,Z_0,\{p\})$, with the following transition function:

   - $\delta(q,0,Z_0) = \{(q,XZ_0)\}$
   - $\delta(q,0,X) = \{(q,XX)\}$
   - $\delta(q,1,X) = \{(q,X)\}$
   - $\delta(q,\epsilon,X) = \{(p,\epsilon)\}$
   - $\delta(p,\epsilon,X) = \{(p,\epsilon)\}$
   - $\delta(p,1,X) = \{(p,XX)\}$

- $\delta(p, 1, Z_0) = \{(p, \epsilon)\}$

To apply the construction given in the proof of Theorem 6.14 in the text, we would first have to convert this machine to accept with an empty stack, rather than accepting when it is in state p.

However, we note that this PDA will accept any string that starts with 0. We need to read a 0 when we see Z_0 on the stack, because there is no value given for $\delta(q, 1, Z_0)$. But after the first zero, we can just read 1's and 0's, staying it state q and pushing an extra X onto the stack for each 0. Then once we reach the end of the input, we can follow an ϵ transition to state p and accept. (This requires having an X on the stack, which we had already pushed by reading the first 0.)

So we write a grammar that produces any string starting with 0.

$$\begin{aligned} S &\rightarrow 0T \\ T &\rightarrow 0T \mid 1T \end{aligned}$$

3. Exercise 7.1.3 on page 277 of Hopcroft et al.
Begin with the following grammar:

$$\begin{aligned} S &\rightarrow 0A0 \mid 1B1 \mid BB \\ A &\rightarrow C \\ B &\rightarrow S \mid A \\ C &\rightarrow S \mid \epsilon \end{aligned}$$

(a) Eliminate ϵ-productions.

Since C can produce ϵ, A can also produce ϵ, so B can produce ϵ, so S can produce ϵ. therefore, we need to change the productions for every symbol.

$$\begin{aligned} S &\rightarrow 00 \mid 0A0 \mid 11 \mid 1B1 \mid B \mid BB \\ A &\rightarrow C \\ B &\rightarrow S \mid A \\ C &\rightarrow S \end{aligned}$$

(b) Eliminate any unit productions in the resulting grammar.

We note that A, B, and C will all just produce S again. So we remove them.

$$\begin{aligned}
S &\rightarrow 00 \mid 0A0 \mid 1B1 \mid BB \\
A &\rightarrow 00 \mid 0A0 \mid 1B1 \mid BB \\
B &\rightarrow 00 \mid 0A0 \mid 1B1 \mid BB \\
C &\rightarrow 00 \mid 0A0 \mid 1B1 \mid BB
\end{aligned}$$

(c) Eliminate any useless symbols in the resulting grammar.

The variable C has now become unreachable. We also remove A and B, because they are exactly equal to S. Our grammar becomes:

$$S \rightarrow 00 \mid 0S0 \mid 1S1 \mid SS$$

(d) Put the resulting grammar into Chomsky Normal Form.

To make this a CNF grammar, we first create variables $A \rightarrow 0$ and $B \rightarrow 1$. We then divide the two productions of length 3 using variables C and D.

Our final CNF grammar is:

$$\begin{aligned}
S &\rightarrow AA \mid AC \mid BD \mid SS \\
A &\rightarrow 0 \\
B &\rightarrow 1 \\
C &\rightarrow SA \\
D &\rightarrow SB
\end{aligned}$$

4. Exercise 7.1.6 on page 278 of Hopcroft et al.
Design a CNF grammar for the set of strings of balanced parentheses. You need not start from any particular non-CNF grammar.

We begin with the following grammar for this language given in class:

$$S \rightarrow SS \mid (S) \mid ()$$

To convert this to a CNF grammar, we first let $L \rightarrow ($ and $R \rightarrow)$. Now we can write our grammar as:

$$\begin{aligned}
S &\rightarrow SS \mid LSR \mid LR \\
L &\rightarrow (\\
R &\rightarrow)
\end{aligned}$$

Lastly, we need to change our productions so that we can only produce two non-terminals in each step. So we introduce a new symbol, A, to break up the production of LSR. This gives us:

$$\begin{aligned} S &\rightarrow SS \mid LA \mid LR \\ A &\rightarrow SR \\ L &\rightarrow (\\ R &\rightarrow) \end{aligned}$$

5. Exercise 7.2.1(b,c) on page 286 of Hopcroft et al.
Use the CFL pumping lemma to show that each of these languages is not context-free.

   (b) $L = \{a^n b^n c^i | i \leq n\}$.

   Assume this language is context free. We will apply the pumping lemma to reach a contradiction.

   Let our string be $z = a^p b^p c^p$, where p is the pumping length. We can break z into $uvwxy$, where

   - $|vwx| \leq p$
   - $|vx| > 0$
   - $uv^i wx^i y \in L$ for all $i \geq 0$

   Now we have five different cases:

   - v and x both contain only a's
   - v contains only a's and x contains only b's
   - v and x both contain only b's
   - v contains only b's and x contains only c's
   - v and x both contain only c's

   Note that v and x cannot bridge the gap between the a's and the c's, because it has length p, and the length of vwx is at most p. Also, v and x each must contain a single type of character, or else pumping them up will break the format of our language (a's, and then b's, and then c's).

   In the first case, we choose $i = 0$. Note that v and x cannot both be empty, so in case 1, we have removed some a's, but no b's or c's. A string with more b's and c's than a's cannot be in our language.

   In case 2, again having choosing $i = 0$, we remove either some a's, some b's, or both. So either the number of c's is larger than the number of a's, or it is larger than the number of b's, or both. This string cannot be in our language.

Case 3 follows similarly. In case 4, we choose $i = 2$. If we add b's, the b's will no longer match the a's. If we add c's, there will be more c's than a's. Since $|vx| > 0$, there is no way for the resulting string to be in our language.

In case 5, we again choose $i = 2$. We add c's, so there is no way for the string to be in our language.

(c) $L = \{0^p | p \text{ is a prime}\}$. (Hint: Adapt the same ideas in Example 4.3, which showed this language not to be regular.)

Assume L is context free, and let $z = 0^k$, where k is a prime that is at least $n+2$, where n is the pumping length.

Now $z = uvwxy$, and no matter how we pump, v and x, we will get another string in our language. Let $|vx| = m$, and let $z' = uv^{k-m}wx^{k-m}y$. Then $|z'| = m(k-m) + m = (m+1)(k-m)$.

$m+1$ must be at least 2, because $|vx| > 0$. Also, $k - m$ must be at least 2, because we chose k to be at least $n+2$, and $|vx| \leq |vwx| \leq n$. Therefore, $|z'|$ cannot be prime, so L cannot be context free.