CIS 511 Homework 2

Stephen Phillips, Dagaen Golomb

February 10, 2015

Problem 1

Part a

First we show that $A = \{\mathbf{a}^m\mathbf{b}^n\mathbf{c}^n | m, n \geq 0\}$ is context free. The grammar is as follows:

$$\begin{aligned} S &\to XY \\ X &\to \varepsilon \mid aX \\ Y &\to \varepsilon \mid bYc \end{aligned}$$

Clearly this generates only strings in the language, and every string in A can be mapped to this by the number of generations of X and Y corresponding to m and n respectively. The language $B = \{\mathbf{a}^n\mathbf{b}^n\mathbf{c}^m | m, n \geq 0\}$ is almost identical:

$$\begin{aligned} S &\to XY \\ X &\to \varepsilon \mid aXb \\ Y &\to \varepsilon \mid cY \end{aligned}$$

With the proof that for this being the same as A. So these two languages are context free, however their intersection $C = A \cap B = \{\mathbf{a}^n\mathbf{b}^n\mathbf{c}^n | n \geq 0\}$ is not context free, as we showed in class.

Part b

We show that the complement of a context free language is not necessarily context free. Suppose not, i.e. that CFLs are closed under complementation. Now, for any context free languages A and B, $A \cup B$ would be context free, since context free languages are closed under union. Then $A \bar{\cup} B = \bar{A} \cap \bar{B}$ would also be in the language. Since we assume that context free languages are closed under complement $\bar{A}$ and $\bar{B}$ would be context free. But since A and B are arbitrary that means the intersection of any two context free languages is context free, a contradiction (proved false in part A above). Therefore context free languages are not closed under complement.

Problem 2

The context free grammar G of $L = \{w\#x | w^{\mathcal{R}} \text{is a substring of } x, x \in \{0,1\}^*\}$ is:

$$\begin{aligned} S &\to aSXaX \mid bSXbX \mid \# \\ X &\to \varepsilon \mid aX \mid bX \end{aligned}$$

Now we need prove this is correct, or in other words $L(G) \subseteq L$ and $L \subseteq L(G)$.

- $L(G) \subseteq L$. This generates strings of the form $\sigma_1\sigma_2 \dots \sigma_n \# \{a,b\}^* \sigma_n \{a,b\}^* \sigma_{n-1} \dots \{a,b\}^* \sigma_1 \{a,b\}^* = w\#x$. Each generation of S creates some σ_i on the left and right, with the right having the reverse order of the left. The variable X can generate any string in $\{a,b\}^*$, which generates the strings between the characters on the left side. So by construction $w^{\mathcal{R}}$ is a substring of x as every σ_i in w is in x and in the same order. Therefore the language this grammar generates is a subset of L
- $L \subseteq L(G)$. Again the grammar generates strings of the form $\sigma_1\sigma_2 \dots \sigma_n \# \{a,b\}^* \sigma_n \{a,b\}^* \sigma_{n-1} \dots \{a,b\}^* \sigma_1 \{a,b\}^* = w\#x$. By definition if x is a substring of y then there exists $z_1, z_2, \dots, z_n, z_{n+1} \in \{a,b\}^*$ such that $y = z_1 x_1 \dots z_n x_n z_{n+1}$. So for every string $s = w\#x \in L$ map w_1 to σ_n, and in general w_i to σ_{n-i+1} and then the z_i map to the $\{a,b\}^*$.

Problem 3

Show that adding the rule $S \to SS$ fails to show that context free languages are closed under Kleene star.

The easiest way to show this is to show that this production does not exhibit the behavior of the Kleene star. This fails because it is possible that the original grammar did not have S able to go to a start symbol. For instance the language $L = \{a^n b \mid n > 0\}$ can be generated by the following grammar:

$$S \to \ aS \mid ab$$

Adding the rule SS makes this:

$$S \to \ aS \mid ab \mid SS$$

Which is the language $L' = \{x^m \mid x \in L, m > 0\}$ whereas $L^* = L' \cup \{\varepsilon\}$.

Another easy example is any CFL that does not include ε, adding this production will produce another grammar that does not contain ε. But the Kleene star can always generate ε by definition, so this production does not exhibit the behavior of the Kleene star. Therefore, it cannot be used to prove closure under such.

Problem 4

Changing the grammar to new add new start symbol S' and add to the production rules:

$$S' \to \ SS' \mid \varepsilon$$

This makes an arbitrary number of strings in the language concatenated, including 0.

Problem 5

Find a context free grammar for the language $L = \{x | \#\text{a's is 2 times \#b's}\}$

$$S \to aSbSa \mid bSaSa \mid aSaSb \mid SS \mid \varepsilon$$

Proof by induction: Note strings of length 0 (ε) hold, since 0 = 2*0. The next valid strings are of length 3 and include only aab, aba, and baa. So this holds for the smallest 4 possible strings. Now assume productions up to n have generated strings in L. For production n+1, we either add 0 a's and 0'b, hence keeping twice as many a's as b's, or we add 2 a's and 1 b. In the latter case, we still maintain that we have twice as many a's as b's. Therefore, this production only produces strings with twice as many a's as b's. Note that since we include all permutations of the a's and b's and allow insertion between any two (including possibly with the empty string), this generates all such strings.

Problem 6

Show that $L = \{xy \mid x, y \in \{0,1\}^* | x| = |y|, x \neq y\}$ is context free

The following grammar generates the language

$$\begin{aligned} S &\to S_0S_1|S_1S_0|\varepsilon \\ S_0 &\to BS_0B|0 \\ S_1 &\to BS_1B|1 \\ B &\to 0|1 \end{aligned}$$

Proof:

- $L(G) \subseteq L$. Strings generated by this language are of the form

  $$\{0,1\}^n1\{0,1\}^n\{0,1\}^m0\{0,1\}^m = \{0,1\}^n1\{0,1\}^m\{0,1\}^n0\{0,1\}^m$$

  (or switch the 1 and the 0, it is equivalent). We can see that $x = \{0,1\}^n1\{0,1\}^m$ and $y = \{0,1\}^n0\{0,1\}^m$. as the $(n+1)^{th}$ bit differs, these strings cannot be the same. However they are both of length $m+n+1$. Therefore this string xy is in the language.

- $L \subseteq L(G)$. Suppose we are given two strings x, y such that $|x| = |y|, x \neq y$. Then by definition there must exist one bit where they differ, say the $(i+1)^{th}$ bit, with the strings being of length $i+j+1$. Then the string xy is of the form $\{0,1\}^ib\{0,1\}^j\{0,1\}^i\bar{b}\{0,1\}^j$, where $\bar{b}$ is the complement of b. Then we can generate with the above grammar using i generations of S_b and j generations of $S_{\bar{b}}$ with the appropriate ordering, since $\{0,1\}^ib\{0,1\}^j\{0,1\}^i\bar{b}\{0,1\}^j = \{0,1\}^ib\{0,1\}^i\{0,1\}^j\bar{b}\{0,1\}^j$. Therefore we can generate any string in this language.

Problem 7

To implement the above grammar we use the machine $M = (Q, \Sigma, \Gamma, q_0, \delta, F)$ implemented in the same fasion of converting a grammar to a machine shown in class. The machine is as follows:

$$\begin{aligned} \Sigma &= \{0,1\} \\ \Gamma &= \{0,1,\$,S_0,S_1,B\} \\ Q &= \{q_{start}, q_{end}, q_c, q_{S_0}, q_{S_1}, q_{S_0B}, q_{S_1B}\} \\ q_0 &= q_{start} \\ F &= \{q_{end}\} \end{aligned}$$

Each of the states has a meaning. The states q_{start} and q_{end} have obvious meanings. The state q_c means the computation state, q_{S_0} means the states where we add the rules for S_0. Similarly for q_{S_1}. The states q_{S_0B}, q_{S_1B} are intermediate states to push more symbols on the stack.

And now for the transition function:

$$\begin{aligned} \delta(q_{start},\varepsilon,\varepsilon) &= \{(q_c,\$)\} \text{Pushing the end of stack symbol} \\ \delta(q_c,0,B) &= \{(q_c,\varepsilon)\} \text{Implementing } B\text{'s rules} \\ \delta(q_c,1,B) &= \{(q_c,\varepsilon)\} \\ \delta(q_c,\varepsilon,S_0) &= \{(q_{S_0},B)\} \text{Implementing } S_0\text{'s rules} \\ \delta(q_{S_0},\varepsilon,\varepsilon) &= \{(q_{S_0B},S_0)\} \\ \delta(q_{S_0B},\varepsilon,\varepsilon) &= \{(q_c,B)\} \\ \delta(q_c,\varepsilon,S_1) &= \{(q_{S_1},B)\} \text{Implementing } S_1\text{'s rules} \\ \delta(q_{S_1},\varepsilon,\varepsilon) &= \{(q_{S_1B},S_1)\} \\ \delta(q_{S_1B},\varepsilon,\varepsilon) &= \{(q_c,B)\} \\ \delta(q_c,\varepsilon,\$) &= \{(q_{end},\varepsilon)\} \end{aligned}$$

As this implements the above grammar, and we showed that this works in class, this implements the language desired. I realized after writing this that I didn't have to go into all this detail, but since I finished it already here it all is.

Problem 8

Show that the language $L = \{a^i b^j \mid i \neq j, 2i \neq j\}$ is context free.

Consider 3 cases (not including the cases where $i = 0$ or $j = 0$)

1. $j < i$

$$S \to aSb|aS|a$$

2. $i < j < 2i$

$$S \to S$$

3. $2i < j$

$$S \to aSbb|Sb|b$$

Problem 9

Part a

Prove $L = \{0^n 1^n 0^n 1^n \mid n \geq 0\}$ is not context free.

Proof: Suppose not. Let p be the pumping lemma constant. Consider the string $s = 0^p 1^p 0^p 1^p$. By the pumping lemma we have $s = uvxyz$ with $|vxy| \leq p$ and $|vy| > 0$, and that $\forall i \geq 0,\ uv^i xy^i z \in L$. By the first property we know vxy cannot contain more than two symbols. If you let $i = 2$ you have for any possible $s' = uv^2xy^2z$ that we have an uneven number of 0s and 1s in the section v^2xy^2 than in the sections it is not in. This means $s' \notin L$. But by the pumping lemma $s' \in L$, a contradiction. Therefore L is not context free.

Part d

Prove $L = \{t_1 \# t_2 \# \ldots \# t_k \mid k \geq 2, t_i \in \{0,1\}^* \text{ and } \exists i, j : i \neq j \wedge t_i = t_j\}$ is not context free.

Proof: Suppose not. Let p be the pumping lemma constant. Consider the string $s = 0^p 1^p \# 0^p 1^p$. By the pumping lemma we have $s = uvxyz$ with $|vxy| \leq p$ and $|vy| > 0$, and that $\forall i \geq 0,\ uv^i xy^i z \in L$. We consider two cases. First, if vxy does not contain the #, then it lies entirely on one side. We can pump once to change the string on this side, therefore making it not equivalent to the other side. In this case the new string $uv^2xy^2z \notin L$. In the other case, assume vxy does contain the #. There are two subcases. First, if the # is in x, then u consists of only 1's and y consists of only 0's. So we can pump once and the two sides are unequal, so $uv^2xy^2z \notin L$. If either u or y contain the #, we can pump down to 0 and remove it, thereby getting a string with no # that cannot be in the language. We have now covered all cases and subcases, and arrived at strings not in L using the pumping lemma. Hence, L is not context free.

Problem 10

If a CFG has b symbols show that if there exists a string in the language that has a derivation of size greater than 2^b that the language is infinite. Assume that the CFG is in Chomsky normal form.

Proof: Each step in a derivation represents a split in the parse tree. Therefore the derivation with at least $2^b + 1$ steps represents a parse tree with $2^b + 1$ steps. Therefore the height of this parse tree is $b + 1$, since we are in Chomsky normal form, where each node has at most branching factor 2. And by the pigeonhole principle we have a repeated symbol on the path from the root start symbol to the furthest leaf node or terminal. Therefore like we did in the pumping lemma, we can repeat that symbol as many times as we wish to generate a new string in the language. Therefore we have a infinite language.