CS500 Homework #2 Solutions

1. Consider the two languages

$$L_1 = \{a^i b^j c^k d^\ell \mid i = j \wedge k = \ell\}$$

$$L_2 = \{a^i b^j c^k d^\ell \mid i = k \wedge j = \ell\}$$

Show that L_1 is context-free but L_2 is not.

Answer. L_1 is the concatenation of $\{a^n b^n\}$ with $\{c^n d^n\}$, and each of these is easily seen to be context-free, e.g. with grammars $S \to aSb \mid \epsilon$ and similarly $S \to cSd \mid \epsilon$. Since the context-free languages are closed under concatenation we're done.

To prove that L_2 is not context-free, we prove by contradiction that the pumping lemma does not hold. Suppose it did with some value of p; then consider the word $w = a^p b^p c^p d^p$. Writing $w = uvxyz$, we see that for $uv^i xy^i z$ to be in L_2 for all i, neither v nor y can cross a boundary between the blocks of different symbols, since repeating them would then give a word outside $a^*b^*c^*d^* \supset L_2$. Moreover, if v is in the block of as or bs, then y must be in the block of cs or ds respectively to keep the corresponding block the same size. But this implies that x contains either b^p or c^p, and in either case since $|vy| > 0$ this violates the third condition of the pumping lemma, $|vxy| \leq p$. □

2. Show that both the context-free languages and the deterministic context-free languages are closed under intersection with regular languages. That is, suppose R is regular; show that if L is context-free, then so is $L \cap R$, and if L is deterministic context-free, then so is $L \cap R$.

Answer. The idea is to run the PDA for L and the DFA for R in parallel, just as we constructed a DFA for the intersection of two regular languages by running both their DFAs in parallel and accepting only if both machines accept.

Formally, suppose the PDA for L has finite states Q_1, stack alphabet Γ, transition function δ_1, and accepting states F_1, and the DFA for R has similarly Q_2, δ_2, and F_2. Assume for simplicity that the PDA for L reads a input symbol on every step. Then the PDA for $L \cap R$ will have states $Q = Q_1 \times Q_2$ and transition function $\delta : Q \times \Sigma \times \Gamma_\epsilon \to Q \times \Gamma_\epsilon$, where the new state is $(\delta_1(q_1, a), \delta_2(q_2, a))$ and we use δ_1 to determine what, if anything, we push on the stack. Our set of accepting states is $F = F_1 \times F_2 \subset Q$. The resulting PDA is deterministic if the original PDA for L is, so $L \cap R$ is a CFL or DCFL if L is. □

3. Inspired by the previous problem, let L be the parenthesis language $\{\epsilon, (), (()), ()(), \ldots\}$ and R the regular language where neither '(' nor ')' can occur more than 3 times in a row. Give a context-free grammar for $L \cap R$.

   Answer. The idea is for the variables to "know" how many left or right parentheses there already are next to them. Let there be 9 variables, $\{S_{\ell,r} \mid 0 \leq \ell, r < 3\}$ plus the start symbol S. Then $S_{\ell,r}$ means "a nonempty string with ℓ '('s to its left and r ')'s to its right." The simplest grammar I found is

$$S_{\ell,r} \to (S_{\ell+1,1})S_{0,r} \mid (S_{\ell+1,r+1}) \mid ()S_{0,r} \mid ()$$

   where we disallow any rules where $\ell \geq 3$ or $r \geq 3$ on the right-hand side. Note that we don't allow $S_{\ell,r}$ to disappear, since then we wouldn't know whether r' in $S_{\ell,r} \to (S_{\ell+1,r'})S_{0,r}$ should be 1 or $r+1$. Finally, we have

$$S \to S_{0,0} \mid \epsilon$$

   which also allows us to generate the empty word. □

4. Recall that a grammar in *Chomsky normal form* is one where all rules are of the form $A \to BC$ or $A \to a$, where capital and lower-case letters represent variables and terminals respectively. Let G be a grammar in Chomsky normal form with $|V| = k$ variables, and let L be the language generated by G. Show that if L contains a word of length greater than 2^{k-1}, then L is infinite.

   Answer. Derivations of grammars in Chomsky normal form are binary trees, where the internal nodes correspond to rules of the form $A \to BC$ and the leaves correspond to rules of the form $A \to a$. Since a binary tree of depth k has at most 2^{k-1} leaves (counting both the root and the leaves toward the depth) any word of length greater than 2^{k-1} must have a derivation tree of depth greater than k. That means that it contains some path of length greater than k, and by the Pigeonhole Principle some variable is repeated along that path. The argument of the pumping lemma then applies, and we can generate an infinite number of words of the form uv^ixy^iz. □

5. Show that the language

$$L = \{a^i b^j c^k \mid i \neq j \vee j \neq k\}$$

   is context-free but not deterministic context-free.

   Answer. To prove it is context-free, recall that context-free languages are closed under union, and write L as

$$L = \{a^i b^j c^k \mid i \neq j\} \cup \{a^i b^j c^k \mid j \neq k\}$$

   Both of these languages are easily seen to be context-free; for the first one, for instance, push when you read a and pop when you read b, and accept if the stack goes empty before we're done with the b's, or if it is still nonempty when we read the first c.

However, if L were deterministic context-free then $\overline{L}$ would be also. But if $\overline{L}$ were context-free, then

$$\{a^n b^n c^n\} = \overline{L} \cap a^* b^* c^*$$

would be context-free since the CFLs are closed under intersection with regular languages. Since we know that $\{a^n b^n c^n\}$ is not context-free, this is a contradiction. Thus L is a CFL but not a DCFL. □

6. Show that the *complement* of the set of palindromes, $L = \{w \in \{a, b\}^* \mid w \neq w^R\}$, is context-free. Since both L and $\overline{L}$ are context-free, does this mean that they are deterministic context-free? Give a DPDA that recognizes L, or some intuition for why one does not exist.

   Answer. Here is a context-free grammar for L:

$$\begin{aligned} S &\rightarrow aSa \mid bSb \mid aSb \mid bSa \mid aTb \mid bTa \\ T &\rightarrow aTa \mid aTb \mid bTa \mid bTb \mid a \mid b \mid \epsilon \end{aligned}$$

   We require S to emit at least one mismatched pair before it becomes a T, which then emits more pairs until it leaves behind a single symbol or disappears.

   However, just because L and $\overline{L}$ are both context-free doesn't mean that they are deterministic context-free (although the converse is true); it could be that they both require non-deterministic PDAs. In fact, imagine a DPDA trying to recognize a palindrome. At the center of the word, it needs to stop pushing and start popping; but without some kind of marker, it has no way of knowing when it is halfway through. An NPDA, on the other hand, can guess when to turn around. □

7. We saw in class that while $\{a^n b^n c^n\}$ is not context-free, it is the intersection of two context-free languages. Consider a k-symbol alphabet $\{a_1, \ldots, a_k\}$, and the language $L_k = \{a_1^n a_2^n \cdots a_k^n\}$. What is the smallest number of context-free languages such that L_k is equal to their intersection?

   Answer. Two.

$$L_k = \{a_1^{n_1} a_2^{n_2} \cdots a_k^{n_k} \mid n_i = n_{i+1} \text{ for even } i\} \cap \{a_1^{n_1} a_2^{n_2} \cdots a_k^{n_k} \mid n_i = n_{i+1} \text{ for odd } i\}$$

   Both of these are CFLs since they are concatenations of $k/2$ or $k/2 - 1$ CFLs of the form $\{a^n b^n\}$ with perhaps the regular languages a_1^* and a_k^*. □

8. Show that the language

$$L = \{ucv \mid u, v \in \{a, b\}^*, u \neq v\}$$

   is context-free. (Note that u and v are not required to have the same length, and c is used as a marker.) Hint: first, design a grammar for

$$\{ucv \mid u, v \in \{a, b\}^*, |u| = |v|, \text{ and } u \text{ and } v \text{ differ in their last symbol}\}$$

and then figure out how to extend u and v with arbitrary strings.

Answer. We have $\Sigma = \{a, b\}$, so Σ^i means any word of length i. Then rewrite L as

$$(\Sigma^i a \Sigma^* \, c \, \Sigma^i b \Sigma^*) \cup (\Sigma^i b \Sigma^* \, c \, \Sigma^i a \Sigma^*)$$

This formula requires that $u_{i+1} \neq v_{i+1}$ for some i, and therefore $u \neq v$. We start by generating

$$(\Sigma^i a \, c \, \Sigma^i b) \cup (\Sigma^i b \, c \, \Sigma^i a) \, ,$$

in which u and v differ on their last symbol, with the grammar

$$\begin{array}{rcl} S & \rightarrow & Tb \mid Ua \\ T & \rightarrow & aTa \mid aTb \mid bTa \mid bTb \mid ac \\ U & \rightarrow & aUa \mid aUb \mid bUa \mid bUb \mid bc \end{array}$$

Then we get L by allowing arbitrary words Σ^* to the left of c and at the end of the word. To do this, add the rules

$$\begin{array}{rcl} c & \rightarrow & ac \mid bc \\ S & \rightarrow & Sa \mid Sb \end{array}$$

which completes the grammar for L. □

9. Let L be the copy language,

$$L = \{ww \mid w \in \{a, b\}^*\} \quad .$$

Show that its complement $\overline{L}$ is context-free. This is tricky, but the idea is similar to the previous problem.

Answer. $\overline{L}$ is the union of the set of words of odd length (which is regular and therefore context-free) with the language

$$L' = \{uv \mid |u| = |v| \text{ and } u \neq v\}$$

We can rewrite this as

$$L' = (\Sigma^i a \Sigma^j \cdot \Sigma^i b \Sigma^j) \cup (\Sigma^i b \Sigma^j \cdot \Sigma^i a \Sigma^j)$$

As in the previous problem, this ensures that $u_{i+1} \neq v_{i+1}$ and so $u \neq v$.

Now for the trick: since $\Sigma^j \Sigma^i = \Sigma^i \Sigma^j$, this is the same as

$$L' = (\Sigma^i a \Sigma^i \cdot \Sigma^j b \Sigma^j) \cup (\Sigma^i b \Sigma^i \cdot \Sigma^j a \Sigma^j)$$

Now the Σ^is and Σ^js no longer cross, and each of the languages in this union is the concatenation of two context-free languages. Our grammar is then

$$\begin{aligned} S &\to TU \mid UT \\ T &\to aTa \mid aTb \mid bTa \mid bTb \mid a \\ U &\to aUa \mid aUb \mid bUa \mid bUb \mid b \end{aligned}$$

Of course, this provides another example of a language which is a CFL but not a DCFL. □

10. Consider the language

$$L = \{w \in \{a, b\}^* \mid \#_a(w) = \#_b(w)\}$$

i.e., words with an equal number of as and bs.

(a) How many words $N(\ell)$ are there of each length ℓ?

Answer. If ℓ is odd, $N(\ell) = 0$; if ℓ is even, we just need to choose which $\ell/2$ of the symbols are a's, giving $N(\ell) = \binom{\ell}{\ell/2}$.

(b) What is its generating function $g(z) = \sum_\ell N(\ell)\, z^\ell$? (Feel free to ask Mathematica or Maple to sum the series.)

Answer. We get

$$\sum_{\ell=0,2,4,\ldots}^{\infty} \binom{\ell}{\ell/2} z^\ell = \sum_{n=0}^{\infty} \binom{2n}{n} z^{2n} = \frac{1}{\sqrt{1-4z^2}}$$

(c) Recall the unambiguous grammar for the bracket language, $S \to (S)S \mid \epsilon$. Inspired by this, we might hope that

$$S \to aSbS \mid bSaS \mid \epsilon$$

is an unambiguous grammar for L, but unfortunately it's not. Explain why.

Answer. Because we can derive the word $abab$ in two distinct ways: $S \to aSbS \to aSb[aSbS] \to abab$, and $S \to aSbS \to a[bSaS]bS \to abab$ (here $[\cdot]$ marks where the rule is applied in the second step).

(d) Now construct an unambiguous grammar for L. Hint: what kind of paths — tracking the depth of the stack, or tracking the imbalance between as and bs seen so far — do words in L correspond to, and how can we unambiguously define them as being made up of smaller paths?

Answer. If we define the height as the number of as so far minus the number of bs — or, equivalently, the depth of the stack if there have been more as and minus the depth if there have been more bs — the words in L correspond to paths which begin and end at height zero.

One way to decompose these unambiguously is shown in the figure. Look at the first place where the height crosses zero — i.e., the shortest initial subword which is also in L. Consider this part of the path. It either consists of a path which is nowhere negative, bracketed by a and b (the example shown in the figure) or a path which is nowhere positive, bracketed by b and a. Let's define variables A and B for the nowhere-negative and nowhere-positive paths respectively. In both cases the remainder of the path is another path which begins and ends at zero, and thus which corresponds to another word in L; this is generated by another copy of the start symbol S.

Paths which are nowhere negative are isomorphic to words in the bracket language, where '(' and ')' correspond to a and b respectively. We can generate this unambiguously with the grammar $A \to aAbA \mid \epsilon$. We generate nowhere-positive paths by switching a and b.

All this gives the following set of rules:

$$\begin{aligned} S &\to aAbS \mid bBaS \mid \epsilon \\ A &\to aAbA \mid \epsilon \\ B &\to bBaB \mid \epsilon \end{aligned}$$

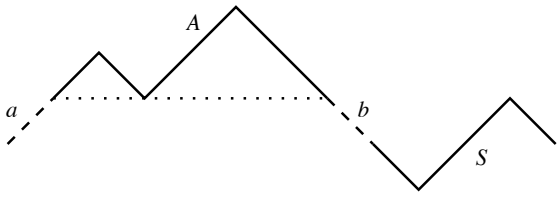

Figure 1: An example of the decomposition. Here $aabaabbbbaab = a(abaabb)b \cdot baab$.

(e) Use your unambiguous grammar from (d) to derive the generating function $g(z)$ and check that it gives the same answer as you gave in (b) above.

Answer. The rules of the grammar above imply that

$$\begin{aligned} g(z) &= z^2 g_A(z) g(z) + z^2 g_B(z) g(z) + 1 \\ g_A(z) &= z^2 g_A(z)^2 + 1 \\ g_B(z) &= z^2 g_B(z)^2 + 1 \end{aligned}$$

where g_A and g_B are the generating functions for the languages generated by A and B. Solving the quadratic equations for g_A and g_B, we get

$$g_A(z) = g_B(z) = \frac{1 - \sqrt{1 - 4z^2}}{2z^2}$$

which is the generating function for the bracket language. Substituting this into the above equation for $g(z)$ we get

$$g(z) = z^2 \left(g_A(z) + g_B(z)\right) g(z) + 1 = \left(1 - \sqrt{1 - 4z^2}\right) g(z) + 1$$

and solving this linear equation for g gives

$$g(z) = \frac{1}{\sqrt{1 - 4z^2}} \ .$$