Introduction to the Theory of Computation
Homework #3 Solutions

1. (Problems 2.15 and 2.16) Prove that regular languages are also CFLs. There are several ways to do this, using grammars, push-down automata, and closure under the regular expression operations (union, concatenation, and star). Give all three proofs if you can.

Proof 1: Grammars. The idea is for the variables to represent states of a DFA and move right, leaving terminals behind them, and finally disappear if they correspond to an accepting state. If $L = L(M)$ where M is the DFA (Q, q_0, δ, F), construct a grammar with variables $V = Q$, terminals Σ, start symbol $S = q_0$, and the following rules:

$$\begin{aligned} q &\rightarrow aq' \text{ if } \delta(q, a) = q' \\ q &\rightarrow \epsilon \text{ if } q \in F \end{aligned}$$

Then a derivation of a word in Σ^* corresponds to an accepting computation of the DFA.

Proof 2: PDAs. Any DFA is a PDA (in fact, a DPDA) which never looks at the stack, never pushes or pops the stack, and accepts by being in an accepting state.

Proof 3: Regular expressions. Remember a language is regular if and only if it can be derived from the languages $\{a\}$ for each $a \in \Sigma$, and $\emptyset$, using the operations union, concatenation, and star. We showed in class that the CFLs are closed under these three operations: if two languages L_1, L_2 have start symbols S_1, S_2, we can make their union with the start symbol S and the rules $S \rightarrow S_1$, $S \rightarrow S_2$, their concatenation with the rule $S \rightarrow S_1 S_2$, and L_1^* with $S \rightarrow S_1 S, \epsilon$.

Finally, the language $\{a\}$ has the grammar $S \rightarrow a$, and $\emptyset$ has the grammar with no rules (and so no way to make a string of terminals from S).

2. (Problem 2.17a) Suppose L is a CFL and R is regular. Show that $L \cap R$ is a CFL.

Proof. 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. 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. Then we construct a PDA for $L \cap R$ with states $Q = Q_1 \times Q_2$ and transition function $\delta : Q \times \Sigma \times (\Gamma \cup \{\epsilon\}) \rightarrow Q \times (\Gamma \cup \epsilon)$, where the new state is $(\delta_1(q_1, a), \delta_2(q_2, a))$ (we assume for simplicity that the PDA for L reads a symbol of the input on every step) and we use δ_1 to determine what, if anything, we push on the stack. Our set of accepting states is $F = F_1 \times F_2$.

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 $L \cap R$.

Solution: The variables will be $V = \{S_0, S_1, S_2\}$ with start symbol $S = S_0$ and rules

$$\begin{aligned} S_0 &\rightarrow (S_1)S_0,\ \epsilon \\ S_1 &\rightarrow (S_2)S_1,\ \epsilon \\ S_2 &\rightarrow ()S_2,\ \epsilon \end{aligned}$$

4. Recall that a grammar in *Chomsky normal form* is one where all rules are of the form $A \rightarrow BC$ or $A \rightarrow a$, where capital and lower-case letters represent variables and terminals respectively. Give a context-free grammar for the language of palindromes over $\{a, b\}^*$, in Chomsky normal form. Remember to generate palindromes of both odd and even length.

Solution: Let $V = \{S, T, U, A, B\}$ with the rules

$$\begin{aligned} S &\rightarrow AU,\ BT,\ AA,\ BB,\ a,\ b \\ U &\rightarrow SA \\ T &\rightarrow SB \\ A &\rightarrow a \\ B &\rightarrow b \end{aligned}$$

We're just using U and T to simulate the rules $S \rightarrow ASA, BSB$.

5. Give a context-free grammar for the *complement* of the set of palindromes over $\{a, b\}^*$.

Solution: We need to make sure that we generate at least one pair of symbols which doesn't match. To do this, we'll demand that S, which generates matching pairs, has to make a mismatched pair and turn into another variable T before it can disappear. Then T can make any pair it wants, matching or not. So the rules are

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

6. 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. Is $\{a^n b^n c^n d^n\}$? Is $\{a_1^n a_2^n \cdots a_k^n\}$ with k different symbols?

Solution: $\{a^n b^n c^n d^n\}$ is the intersection of two CFLs, namely $\{a^i b^i c^j d^j\}$ and $\{a^i b^j c^j d^k\}$. More generally, $\{a_1^n a_2^n \cdots a_k^n\}$ can be written

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

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/or a_k^*).

7. (Problem 2.19) If a grammar is in Chomsky normal form, prove that a derivation of a word of length n involves exactly $2n-1$ steps (where each step applies one rule in one place).

Proof: Each terminal is made by a rule of the form $A \to a$, so we apply rules of this form n times. We have to make n variables first; since we start with one variable S, and each rule of the form $A \to BC$ increases the number of variables by 1, we apply rules of this type $n-1$ times.

8. Let G be a grammar in Chomsky normal form with $|V| = k$ variables including the start symbol. Show that if $L(G)$ contains a word of length greater than 2^{k-1}, then $L(G)$ is infinite.

Solution: Since derivations in grammars in Chomsky normal form are binary trees, any such tree with depth k has at most 2^{k-1} leaves (where the depth is the length of the longest path from the root to a leaf, including the start symbol). Conversely, any derivation tree with more than 2^{k-1} leaves has a path of length greater than k. Then by the Pigeonhole Principle, some variable A occurs twice along this path.

It's convenient to use the notation $A \to^* w$ if a variable A can lead to a word w (of variables, terminals, or both) through a series of rules. Then (as in the pumping lemma) there is a derivation $A \to^* vAy$ where v and y cannot both be the empty word. Then if $S \to^* uAz$ and $A \to^* x$, we have $S \to^* uv^ixy^iz$ for all $i \geq 0$. Thus $L(G)$ is infinite.

9. (Problems 2.26 and 2.27 – Tricky!) We saw that the "copy language" $\{ww : w \in \{0,1\}^*\}$ is not context-free. However, surprisingly, its complement is context-free! (Thus it gives an example of a CFL whose complement is not a CFL.) To work towards this, construct context-free grammars for the following languages:

$$\{x\#y : x, y \in \{0,1\}^* \text{ and } x \neq y\}$$
$$\{xy : x, y \in \{0,1\}^*, |x| = |y|, \text{ and } x \neq y\}$$

Solution: For the first language, we just need to make sure that x and y differ in at least one place. In other words, we can rewrite this language as

$$\Sigma^i\, 0\, \Sigma^*\, \#\, \Sigma^i\, 1\, \Sigma^* \quad \cup \quad \Sigma^i\, 1\, \Sigma^*\, \#\, \Sigma^i\, 0\, \Sigma^*$$

so that $x_{i+1} \neq y_{i+1}$ (here $\Sigma = \{0,1\}$). We start by generating

$$\Sigma^i\, 0\, \#\, \Sigma^i\, 1 \quad \cup \quad \Sigma^i\, 1\, \#\, \Sigma^i\, 0$$

with the grammar

$$\begin{aligned} S &\to T1,\, U0 \\ T &\to 0T0,\, 0T1,\, 1T0,\, 1T1, 0\# \\ U &\to 0U0,\, 0U1,\, 1U0,\, 1U1, 1\# \end{aligned}$$

Then to allow for Σ^* to the left of # and at the end of the word, we add the rules

$$\begin{aligned} \# &\to 0\#,\, 1\# \\ S &\to S0,\, S1 \end{aligned}$$

Similarly, we can rewrite the second language as

$$\Sigma^i\, 0\, \Sigma^j\, \Sigma^i\, 1\, \Sigma^j \quad \cup \quad \Sigma^i\, 1\, \Sigma^j\, \Sigma^i\, 0\, \Sigma^j$$

so that $x_{i+1} \neq y_{i+1}$. But since $\Sigma^j \Sigma^i = \Sigma^i \Sigma^j$, this is the same as

$$\Sigma^i\, 0\, \Sigma^i \cdot \Sigma^j\, 1\, \Sigma^j \quad \cup \quad \Sigma^i\, 1\, \Sigma^i \cdot \Sigma^j\, 0\, \Sigma^j$$

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

$$\begin{aligned} S &\rightarrow TU,\, UT \\ T &\rightarrow 0T0,\, 0T1,\, 1T0,\, 1T1,\, 0 \\ U &\rightarrow 0U0,\, 0U1,\, 1U0,\, 1U1,\, 1 \end{aligned}$$

Finally, taking the union of this language with the regular language of all words of odd length gives the complement of the copy language $\{ww\}$.