Problem Set 6 Solutions

Problem 1. *Let $L = \{w \in \{a,b,c\}^* : w$ has an equal number of a's, b's, and c's$\}$. Describe a PDA for $\overline{L}$ in English, and then formally specify it.*

A solution and pretty picture can be found at `http://tinyurl.com/ecs120ps6b`

Problem 2. *Let $h\colon \Sigma_\varepsilon \to \Gamma^*$ be an arbitrary function. Extend it to strings and then languages by way of $h(\varepsilon) = \varepsilon$, $h(a_1 a_2 \cdots a_n) = h(a_1)h(a_2)\cdots h(a_n)$ and then $h(L) = \{h(x)\colon x \in L\}$. Here $a_1, \ldots, a_n \in \Sigma$ and $L \subseteq \Sigma^*$. The function h, and its extension to strings and languages, is called a homomorphism. Are the CFLs closed under homomorphism? Prove your answer.*

Yes, CFLs are closed under homomorphism. Let L by a CFL and let $G = (V, \Sigma, R, S)$ be a grammar for it. To make a grammar for $h(L)$, just replace the right-hand side of every rule $A \to \alpha$ by the same right-hand side but with each $a \in \Sigma$ appearing in α replaced by $h(\alpha)$. For example, if $A \to B\,C\,b\,c\,A \in R$ then replace this rule by $A \to B\,C\;h(b)\,h(c)\;A$. It is easy to see that the new grammar, G', satisfies $L(G') = h(L)$.

Problem 3.

a. *Prove that $L_a = \{a^i b^j c^k : j = \max\{i, k\}\}$ is not context free.*

Suppose for contradiction that L_a were context free. Let p be the "p" of the pumping lemma for context free languages. Consider the string $w = a^p b^p c^p$. Suppose $w = uvxyz$, where $|vxy| \le p$ and $|vy| \ge 1$. If vy contains only a's or vy contains only c's, then pump up: the string $uv^2xy^2z \notin L_a$. Suppose vy contains only b's. Then we can pump either way to get a string not in L_a. Suppose v contains two different letters or y contains two different letters. Then uv^2xy^2z is not even of the form $a^*b^*c^*$, so certainly it is not in L_a. Finally, suppose ($v \in a^+$ and) $y \in b^+$, or $v \in b^+$ (and $y \in c^+$). Then we can pump down and there will be too few b's. By $|vwy| \le p$, these are all the possible cases. So in all cases there is some i for which $uv^ixy^iz \notin L$, a contradiction.

b. *Prove that $L_b = \{b_i \# b_{i+1} : b_i$ is i in binary, $i \ge 1\}$ is not context free.*

Suppose for contradiction that L_b were context free. Let p be the "p" of the pumping lemma for context free languages. Consider the string $w = 1^p 0^p \# 1^p 0^{p-1} 1 \in L_b$. Suppose $w = uvxyz$, where $|vxy| \le p$ and $|vy| \ge 1$. If vxy does not contain a #, then pumping either way will cause a contradiction (increasing or decreasing one of the numbers without touching the other). If the # is contained in v or y, then pumping either way leads to a string not even in $(a \cup b)^* \# (a \cup b)^*$, i.e., a string definitely outside of L_b. Because of the $|vxy| \le p$ condition, the only remaining possibility is for $v = 0^i$ and $y = 1^j$, $i, j \ge 1$, to fall on opposite sides of the "#." But pumping up in this case means multiplying the left hand number by some power of two, while it never means multiply the right hand number by some power of two. Thus the pumped-up strings will not remain with the right number one more than the left.

Problem 4. *Are the following languages context free? Prove your answers either way.*

a. $L = \{ww^R : w \in \{a,b\}^*\}$

Context free. A PDA for L would push a marker onto the bottom of the stack, then push the characters of the input string onto the stack. It would non-deterministically guess the middle of the input string and then, having made that guess, it would start popping symbols off the stack, comparing them with the remaining characters. If all of these characters match, exposing the bottom-of-stack marker, the machine accepts.

b. $L = \{ww^Rw : w \in \{a,b\}^*\}$

No, the language is not context free. Suppose for contradiction that L were context free. Let p be the "p" of the pumping lemma for context free languages. Consider the string $s = a^pb^{2p}a^{2p}b^p \in L$. Suppose $s = vwxyz$, where $|wxy| \le p$ and $|wy| \ge 1$. Note that any string $t \in L \cap a^+b^+a^+b^+$ has its partitioning into $t = rr^Rr$ uniquely determined by the length of the longest prefix of t which is in a^+ and of the longest suffix of t which is in b^+. Thus if wxy falls entirely outside of the longest prefix of initial a's and wxy falls entirely outside the longest suffix of final b's, then $vxz \notin L$. So wxy must begin in the initial block of a's or wxy must ends within the final block of b's. In the first case, wxy ends before the second block of b's ends; in the second case, wxy begins after the second block of a's begins. For the former, note that any string $t \in L \cap a^+b^+a^+b^+$ has its partitioning into $t = rr^Rr$ uniquely determined by the longest suffix of t in a^+b^+, and so $vxz \notin L$. In the latter case, note that any string $t \in L \cap a^+b^+a^+b^+$ has its partitioning into $t = rr^Rr$ uniquely determined by the longest prefix of s in a^+b^+, so $vxz \notin L$.

Problem 5. *Answer true or false, proving each answer.*

(a) *Every regular language is generated by an unambiguous CFG.*

True. The grammar obtained by the procedure from PS4 that converts a DFA M to a CFG is always unambiguous. The unique leftmost derivation of $x \in L(M)$ is read off of the DFA.

(b) *Every regular language is generated by an ambiguous CFG.*

False—but almost true. The empty set is not generated by an ambiguous CFG. Every other language is. Given $G = (V, \Sigma, R, S)$, form a new grammar by adding the rule $S \to S \in R$. The grammar will now be ambiguous—assuming $L(G) \neq \emptyset$

(c) *The CFLs are closed under intersection.*

False—described in class.

(d) *Every subset of a regular language is context free.*

False—every non-context-free language $L \subseteq \Sigma^*$ is a subset of the regular language Σ^*.

(e) *If L is context free then $L \cap a^*b^*a^*$ is context free.*

True—the intersection of a context-free language and a regular language is regular. This was mentioned in class and described in the book.

Problem 6 *A queue is similar to a stack, except that pushing and popping happen at opposite ends: symbols are pushed onto the top of the queue and popped, and read, from the bottom of the queue. A queue automaton (QA) is like a PDA but has a queue instead of a stack. Describe a language that is recognized by a QA but not by any PDA.*

The language $L = \{w\#w : w \in a, b^*\}$ is easily seen not to be context-free, by the pumping lemma, and hence not recognized by any PDA. But L is recognized by a simple QA M: as M reads its input x, it pushes each symbol of x onto the top of the queue, until it encounters the #. It pushes the # onto the queue, too. As M reads the remainder of x, it pops off the symbols from the bottom of the queue, checking that they match what is on the queue. If any mismatch is found, the machine rejects. Machine M accepts if the stack contains # after all of the input has been read.

It turns out that QAs are equivalent in power to Turing Machines—a stack beats a queue!