Comments on Solutions to Problem Set 1

Problem 1. **A.** *A **Moore machine** is like a DFA except that each state is annotated by a character. Unlike a DFA, which computes a predicate on strings, a Moore machine computes a string-valued function on strings. Formally define Moore machines and the function computed by a Moore machine.* **B.** *A **Mealey machine** is like a DFA except that each transition out of a state is annotated by a character. As with a Moore machine, a Mealey machine computes a function on strings. Formally define Mealey machines and the function computed by a Mealey machine.* **C.** *Show that a function f is computable by a Moore machine iff f is computable by a Mealey machine.*

People got right the syntax of the machines and the constructions to convert one to the other, but most people didn't bother to define how the two kinds of machines behaved, nor to prove the correctness of each construction. While I'd view correctness as reasonably clear, definitely you should define the behavior of the machines, not only specify the syntax, because there are natural behaviors other than the "intended" one—eg., a Moore machine might spit out an output character each time it leaves a state, as opposed to each time it enters a state, which would not work for the problem.

Problem 2. *We have argued that, up to the naming of states, for any regular language L, there is a unique minimum-state DFA that accepts L. Resolve whether or not the same is true for NFAs: given a regular language L, is there a unique minimum-state NFA for it?*

People didn't have any problem with this. It was pointed out that I didn't say what *unique* means. I had in mind "isomorphic"—the machines are the same up to the naming of states. Under this notion, the answer is no, demonstrated by any old example.

Problem 3. *Use the Myhill-Nerode Theorem to give a different proof from the one given in class that L is regular iff L is decidable by a 2-Way Finite Automaton.*

Most people had some idea for this but a lot of the writeups were so difficult to understand that I didn't really get what you were saying. Writing up something here I come to believe that I misgraded a lot of your papers, too.

Adopting Papadimitriou's TM formalization for a 2WFA, let $M = (K, \Sigma, \delta, q_1)$ be a 2WFA (the δ function may not overwrite a character). Assume $K = \{1, \ldots, k\}$. For a string x from the input alphabet, $n = |x|$, and a state $s \in K$, run M with a start configuration $[sx]$ (the state written just to the left of the where the head is at) and let q be the state the machine is in when its head first reaches cell 0 or $n+1$ if the machine's head ever reaches one of these cells. In the first case, that the head first reaches cell 0, define $f_M^{\mathrm{L}}(s,x) = q$. In the second case, that the head first reaches cell $n+1$, define $f_M^{\mathrm{L}}(s,x) = k + q$. If the head never reaches cell 0 or $n+1$ when started from configuration $[sq]$ then let $f_M^{\mathrm{L}}(s,x) = 0$ in the case that M comes to accept and $f_M^{\mathrm{L}}(s,x) = -1$ otherwise (it loops or rejects). Mirror the definitions just given but with a start configuration of $[x_1 \ldots x_{n-1} s x_n]$ to define $f_M^{\mathrm{R}}(s,x)$. Define $f_M(s,x)\colon\ [1 \mathinner{..} 2k] \times \{0,1\}^* \to [-1 \mathinner{..} 2k]$ as $f_M^{\mathrm{L}}(s,x)$ for $s \in [1 \mathinner{..} k]$ and $f_M^{\mathrm{R}}(s-k,x)$ for $s \in [k+1 \mathinner{..} 2k]$. Define $x \equiv_M y$ to mean that $f_M(s,x) = f_M(s,y)$ for all all $s \in K$. It is easy to check that $\equiv_M$ is an equivalence relation and that it has at most $(2k+2)^{2k+2}$ equivalence classes. Furthermore, letting $L = L(M)$, this equivalence relation is a refinement of the equivalence relation $x \approx_L y$ where $x \approx_L y$ means that $xz \in L$ iff $yz \in L$: $x \equiv_M y \Rightarrow x \approx_{L(M)} y$. Thus the number of equivalence classes for $\approx_L$ is at most the number of equivalence classes for $\approx_M$, which is finite. By the Myhill-Nerode theorem, L is regular.

Problem 4. *Prove that if $L \subseteq 1^*$ then L^* is regular.*

This is a number-theoretic question in disguise, and there were a variety of solutions. The cleanest approach I saw was from Christian Bird and others, and works as follows.

Without loss of generality assume that $L \neq \emptyset$ (since the claim is trivial otherwise) and assume $\varepsilon \notin L$ (since it is irrelevant to L^* whether or not ε is in L). Equate each string 1^i with integer i and in this way regard L as a nonempty set of numbers, also denoted L. Let n be the smallest number in L and, for $i \in [0 \mathinner{..} n-1]$, let p_i be the smallest number in L that is congruent to i modulo n, if such a number exists, and 0 otherwise. We claim that $L^* = (1^{p_0} \cup \cdots \cup 1^{p_{n-1}})^*$ and so L^* is regular. First, we have that $L^* \supseteq \{1^{p_0}, \ldots, 1^{p_{n-1}}\}^*$ because L^* is closed under star, $\{1^{p_0}, \ldots, 1^{p_{n-1}}\}^*$ is the smallest set closed under star containing $1^{p_0}, \ldots, 1^{p_{n-1}}$, and L contains each nonempty string from $1^{p_0}, \ldots, 1^{p_{n-1}}$. More interestingly, $L^* \subseteq \{1^{p_0}, \ldots, 1^{p_{n-1}}\}^*$. Any nonzero $a \in L^*$ is a sum of elements in L and so it suffices to show that any $a \in L$ is a sum of p_i's. But for any $a \in L$ it is the case that $a = kn + p_{a \bmod n} = kp_0 + p_{a \bmod n}$ for some $k \geq 0$ because: $a = k'n + (p_{a \bmod n} \bmod n)$ for some $k' \geq 0$ and $p_{a \bmod n} = k''n + (p_{a \bmod n} \bmod n)$ for some $k'' \geq 0$ so $a = k'n + (p_{a \bmod n} \bmod n) = k'n + p_{a \bmod n} - k''n = (k' - k'')n + p_{a \bmod n}$ where $k = k' - k'' \geq 0$ because $p_{a \bmod n}$ was was defined as the *least* number in L congruent to $a \bmod n$ and a is *some* element of L congruent to $a \bmod n$.

For an example, let $L = \{3, 5, 7, 11, 13, \ldots\}$ be the set of odd primes. Then $n = 3$, $p_0 = 3$, $p_1 = 7$ $p_2 = 5$ and $L^* = (1^3 \cup 1^7 \cup 1^5)^*$.