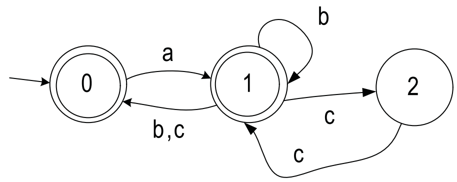
Problem Set 4 Solutions

Problem 1.

(a) *Using the procedure shown in class, convert NFA into a regular expression for the same language.*



(b) *Using the procedure shown in class, convert the regular expression $(ab^* \cup c)^*$ into an NFA for the same language.*

(c) *Suppose that a (fully parenthesized) regular expression α over the alphabet Σ has length n. Convert it to a DFA M for the same language using the procedures seen in class. Show that will M have at most 2^{2n} states. (A tighter bound is possible, but harder.)*

Parts (a) and (b) are straightforward; I'm not going to draw the pictures (but you should certainly be able to). For part (c), think about the method for converting a regular expression over Σ into an NFA. Each character $a \in \Sigma \cup \{\emptyset, \varepsilon\}$ of the NFA contributes at most 2 states to the NFA. Each three characters ($\cup$) contributes one state to the NFA; each three characters (*) contributes one state to the NFA; and each three characters ($\circ$) contributes zero states to the NFA. Summarizing, each character of the NFA contributes at most 2 states to the NFA. So our NFA will have at most $2n$ states and we will get at most 2^{2n} states when we convert it to a DFA. A tighter bound is possible, but requires more work.

Problem 2. *Use the pumping lemma to prove that the following languages are not regular.*

(a) $L = \{x \in \{a,b\}^* :\ x \textit{ is not a palindrome}\}$.

Suppose for contradiction that L is regular. Then its complement $\bar{L} = \{x \in \{a,b\}^* :\ x \text{ is a palindrome}\}$ is also regular. Let p be the pumping length for this language and consider the string $s = a^p b a^p$. By the strong form of the pumping lemma s can be partitioned into $s = xyz$ where y lives within the initial run of zeros, $|xy| \le p$, $|y| \ge 1$, and such that $xy^iz \in L$ for all $i \ge 0$. But xy^0z will then be a string of the form $a^{p-\delta}ba^p$ with $\delta \ge 1$, which is not a palindrome. This is a contradiction.

(b) $L = \{w{=}w\colon\ w \in \{0,1\}^*\}$. (The second $=$ is a character from the alphabet $\{0,1,=\}$ that L is over.)

Assume for contradiction that L were regular. Let p be the pumping length, as guaranteed by the pumping lemma. Let s be the string $1^p = 1^p$. By the strong form of the pumping lemma s can be partitioned into $s = xyz$ where $|xy| \le p$, $|y| \ge 1$, and $xy^iz \in L$ for all $i \ge 0$. With y living inside the initial run of 1, pumping up ($i > 1$) or down ($i = 0$) gives a string $xy^iz \notin L$.

(c) $L = \{a^{2^n} :\ n \ge 0\}$.

Assume for contradiction that L were regular. Let p be the pumping length, as guaranteed by the pumping lemma. Let $s = a^{2^p}$. Then $s \in L$ and $|s| \ge p$ so, by the pumping lemma, there exists x, y, z such that $s = xyz$ and $y \ne \varepsilon$ and $xy^iz \in L$ for all $i \ge 0$. In particular, for some $\alpha \ge 1$ (namely, $\alpha = |y|$), we have that $a^{2^p + i\alpha} \in L$ for all $i \ge 0$, which means that $2^p + i\alpha$ is always a power of two, for any $i \ge 0$. Thus (looking at $i = 1$) we have that $2^p + \alpha$ is a power of two, and (looking at $i = 2$) we have that $2^p + 2\alpha$ is a bigger power of two, so it must be at least twice $2^p + \alpha$; that is, $2^p + 2\alpha \ge 2(2^p + \alpha)$, which means that $2^p + 2\alpha \ge 2^p + 2^p + 2\alpha$, so $0 \ge 2^p$, which is impossible.

Problem 3. *Let $L = \{xx^R \colon\ x \in \{a,b\}^+\}$. Use the Myhill-Nerode theorem to prove that L is not regular.*

Let $\sim$ be the equivalence relation associated to L by the Myhill-Nerode theorem. We need to identify infinitely many inequivalent strings. Consider the set of strings $\{a^n b :\ n \geq 1\}$. We claim that no two of these can be $\sim$ equivalent. Fix $n \neq N$ and observe that $a^n b\ ba^n \in L$ while, on the other hand, $a^N b\ ba^n \notin L$. Note: I had previously listed the language $L = \{xx^R y \colon\ x, y \in \{a,b\}^+\}$. That one is considerably harder.

Problem 4. *Define $A = \{x \in \{a, b, \sharp\}^* : x$ contains an equal number of a's and b's or x contains consecutive $\sharp$s or consecutive letters$\}$.*

(a) Can you use the pumping lemma to prove that A is not regular? Explain.

No, the pumping lemma won't work to show that A is not regular. It won't work because, whatever string $s \in L$ you choose, the string *will* pump. In particular, the portion we usually denote "y" might be a single $\sharp$ symbol or a single letter, and repeating that character, or excising it, will continue to give strings in A.

(b) Prove that A is not regular.

Proof 1. Assume for contradiction that A were regular. Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA that accepts A. Consider the infinitely many strings $x_n = (a\sharp)^n$, for all $n \geq 0$. Because Q is finite, the pigeonhole principle tells us there will be distinct values i, j for which $\delta^*(q_0, x_i) = \delta^*(q_0, x_j)$. But then $\delta^*(q_0, x_i(b\sharp)^i) = \delta^*(q_0, x_j(b\sharp)^i))$. But the LHS must be a state in F while the RHS must be a state outside of F, a contradiction.

Proof 2. Same as above, but cast in the language of the Myhill-Nerode theorem. Let $x_n = (a\sharp)^n$. Let $\sim$ be the equivalence relation defined by $x \sim y$ iff $(\forall z)\ (xz \in A \Leftrightarrow yz \in A)$. Let $[x]$ be the equivalence class of string x with respect to this equivalence relation. I claim that $[x_i] \neq [x_j]$ for all $i \neq j$. The reason is simple: $x_i(b\sharp)^i \in A$, but $x_j(b\sharp)^i \notin A$. So there are infinitely many blocks, so Myhill-Nerode says that A is not regular.

Proof 3. Finally, a proof based on closure properties. Let $L = \{x \in \{a, b, \sharp\}^* : x$ contains an equal number of a's and b's and every other character of x is a $\sharp\}$. Let $R = (a \cup b \cup \sharp)^*\ ((a \cup b)(a \cup b) \cup \sharp\sharp)\ (a \cup b \cup \sharp)^*$. Then R is regular, L and R are disjoint, and $L \cup R = A$, so, by problem 4(e) of this problem set, to show that A is not regular it is enough to show that L is not regular.

Recall that the regular languages are closed under homomorphisms: let $h : \Sigma \to \Gamma^*$ and extend h character-wise to strings (ie, $h(a_1 \cdots a_n) = h(a_1) \cdots h(a_n)$) and string-wise to languages (ie, $h(L) = \{h(x) : x \in L\}$). We claim that if a language L is regular then so is $h(L)$. For if we are given a regular expression α for L then (the properly parenthesized version of) $h(\alpha)$ is a regular expression for $h(L(\alpha))$.

Now consider the specific map h where $h(a) = a$, $h(b) = b$, and $h(\sharp) = \varepsilon$. Then $h(L)$, for the L we specified above, is the set L' of all strings over $\{a, b\}$ with an equal number of a's and b's. We know this language to be not regular (we showed it in class, or you can show it with the pumping lemma, or you can show it with closure properties). So L is not regular, and so A is not regular.

Problem 5. *Are the following statements true or false? Either prove the statement or give a counter-example.*

(a) *If $L \cup L'$ is regular then L and L' are regular.*

False. $L = \{a^n b^n : n \geq 0\}$ and $L = \{a, b\}^*$.

(b) *If L^* is regular then L is regular.*

False. $L = \{1^{2^i} : i \geq 0\}$.

(c) *If LL' is regular then L and L' are regular.*

False. $L = \{a^n b^n : n \geq 0\}$ and $L' = \emptyset$.

(d) *If L and L' agree on all but a finite number of strings, then one is regular iff the other is regular.*

True. $L \oplus R = L'$ and for some finite, and therefore regular, R. But the regular languages are closed under symmetric difference.

(e) *If R is regular, L is not regular, and L and R are disjoint, then $L \cup R$ is not regular.*

True. Suppose instead that $L \cup R$ were regular. Then $(L \cup R) \setminus R = L$ by disjointedness, and the regular languages are closed under difference, so L would be regular.

(f) *If L differs from a non-regular language A by a finite number of strings F, then L itself is not regular.*

True. If L were regular then $A = L \oplus F$ would be regular, too, by closure under $\oplus$.

Problem 6. Specify an algorithm to answer the following question: given a regular expression α, is $L(\alpha) = (L(\alpha))^R$? Upperbound the running time of your algorithm.

The NFA-acceptable languages are closed under reversal: the proof is to take an NFA M and convert it to an NFA M^R, where $L(M^R) = (L(M))^R$, by adding a new start state, connecting it to all the old final states, definalizing those final states, and finalizing the start state. Thus an algorithm to answer this question is as follows: convert α into an NFA M; construct the NFA M^R as above; and apply the procedure we did (convert to a DFA and use the product construction for symmetric difference, then DFS to decide emptiness)

How long will this take? If the regular expression α has length $|\alpha| = n$ then its NFA will have at most $2n$ states by our solution to 1(c); the NFA for α^R will thus have at most $2n + 1$ states; the corresponding DFAs will thus have at most 2^{2n} and 2^{2n+1} states; the size of the DFA constructed by the product construction will then have at most 2^{4n+1} states; and DFS on this will take $O(2^{4n})$ time. A tighter bound is possible,