Problem Set 4 – Due Friday, April 24, 2015

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 \(\alpha\) over the alphabet \(\Sigma\) 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.)

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

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

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

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

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

Problem 4. Define \(A = \{x \in \{a, b, \#\}^* : x\) contains an equal number of a’s and b’s or \(x\) contains consecutive \(\#\)s or consecutive letters\}.

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

(b) Prove that \(A\) is not regular.

Problem 5. Are the following statements true or false? Either prove the statement or give a counterexample.

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

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

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

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

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

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

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