CS 154 - Introduction to Automata and Complexity Theory

Spring Quarter, 2009

Assignment #1 - Due date: Tuesday, 4/14/09

Problem 1. [10 points] Provide DFAs for the following languages over the alphabet $\Sigma = \{0, 1\}$.
(a). All strings that contain three consecutive 1's.
(b). All strings that *do not end* with 00.

Problem 2. [30 points] Consider a DFA $M = (Q, \Sigma, \delta, q_0, F)$ with $F = \{q_f\}$ such that

$$\forall\, a \in \Sigma\,,\ \delta(q_f, a) = \delta(q_0, a)$$

(a). Show that for all *non-empty* strings $w \in \Sigma^*$, it must be the case that $\widehat{\delta}(q_f, w) = \widehat{\delta}(q_0, w)$.
(b). Let $w \in L(M)$ be any string in the language of M. Prove *by induction* that for all $k > 0$, the string $w^k \in L(M)$. (**Hint:** You may use the statement proved in Exercise 2.2.2 on page 53 of the text-book.)
(*Food for thought:* Can you characterize the class of languages accepted by DFAs with the property defined above?)

Problem 3. [25 points]
(a). [5 points] Let $L \subset \{0, 1\}^*$ be the language of all strings such that there are two 0's separated by a number of positions that is a *non-zero* multiple of 5. For example, 1001110 is not in L, but 10111110 is in L. Construct an NFA for this language.
(b). [20 points] You should be glad that I did not ask you to construct a DFA for this language. To truly appreciate this, prove that any DFA for this problem must have at least 2^5 states.
(*Hint:* Study Example 2.13 on page 65 of the text-book.)
(*Food for thought:* Why does your argument fail for NFAs? How does the lower bound on the DFA size compare with your NFA's size? What are the implications for the relative power of NFAs and DFAs?)

Problem 4. [15 points] Consider an NFA $N_1 = (Q, \Sigma, \delta, p, F_1)$ with language $L_1 = L(N_1)$. Define a new NFA $N_2 = (Q, \Sigma, \delta, p, F_2)$ with the set of final states $F_2 = Q - F_1$. (That is, in going from N_1 to N_2 the final states become non-final, and vice-versa.)

Prove or disprove the following statement: *The language $L_2 = L(N_2)$ is the* **complement** *of the language L_1, i.e., $L_2 = \Sigma^* - L_1$.*

Problem 5. [20 points] Recall that for an NFA $M = (Q, \Sigma, \delta, q_0, F)$, we say that it accepts a string $w \in \Sigma^*$ if *at least one* execution of M on input w leads to a final state when the end of input is reached.

Suppose now that we define a new class of NFAs called **Total-NFA** wherein a string w is considered to be accepted if *all possible* executions lead to a final state when the end of input is

reached. (We ignore the executions which get stuck before the end of the input is reached, and require that at least one execution should complete to the end of the input.)

Formally, for a Total-NFA $M = (Q, \Sigma, \delta, q_0, F)$ everything is exactly as in the case of NFAs, except that we define its language to be:

$$L(M) = \{w \in \Sigma^* \mid \widehat{\delta}(q_0, w) \subseteq F \textbf{ and } \widehat{\delta}(q_0, w) \neq \emptyset\}$$

In what follows, you will argue that the class of languages defined by Total-NFAs is exactly the same as the class of languages defined by a DFA, i.e., the *regular* languages.

a). [5 points] Explain why it must be that case that every regular language has a Total-NFA.

b). [15 points] Explain why it must be that case that the language of a Total-NFA is always a regular language.

Reading Assignment:

Each homework will specify a set of readings from the textbook — these are generally a required reading to follow the material presented in the lectures.

1. In Chapter 1, we have only covered Section 1.5 in class. But read Chapter 1 anyway; you really need to be be familiar with the background material in this chapter.

2. By now, we have covered Chapter 2.1, 2.2, and 2.3 in class. Read these and Chapter 2.4 for an interesting application.

3. We will be covering Section 2.5 and portions of Chapters 3 in the next couple of lectures.