Homework #1: Regular Languages

Due Date: 5pm Friday 1/29

Read Chapter 1. In lecture I’ll show you the general equivalence of DFA’s, NFA’s, and regular expressions. We’ll delay the “pumping” topic until the second homework. In this homework I’ll ask you some more specialized questions, mostly about DFA’s.

Definitions

Suppose $M = (Q, \Sigma, \delta, q_0, F)$ is a DFA, as defined by Sipser. Given state $p \in Q$ and input string $x \in \Sigma^*$, let $\delta^*(p, x)$ denote the state that we reach, if we start in state $p$ and then read string $x$. We may define $\delta^*$ recursively (on the length of $x$) by this rule:

$$
\delta^*(p, x) = \begin{cases} 
p & \text{if } x = \varepsilon, \text{ or else} \\
\delta^*(\delta(p, a), y) & \text{where } x = ay \text{ for some } a \in \Sigma \text{ and } y \in \Sigma^*.
\end{cases}
$$

Define the function $q_M : \Sigma^* \to Q$ by the rule $q_M(x) = \delta^*(q_0, x)$. That is, $q_M$ maps string $x$ to the state that $M$ would reach after reading the input $x$. In particular, the language accepted by $M$ is $L(M) = \{x \in \Sigma^* : q_M(x) \in F\}$.

Two DFA’s are equivalent if they have the same input alphabet and they accept the same language. A DFA is minimal if there is no equivalent DFA with fewer states.

Suppose we have a language $L \subseteq \Sigma^*$ and two strings $x, y \in \Sigma^*$ (not necessarily in $L$). Define “$x$ and $y$ are $L$-equivalent” to mean: for all $z \in \Sigma^*$, $xz \in L \iff yz \in L$. We denote this by $x \equiv_L y$. If $x$ and $y$ are not $L$-equivalent, then there should be a distinguishing suffix: that is a string $z$ such that exactly one of the two strings $xz, yz$ is in $L$.

Review Exercises

For general review, I recommend that you try the early exercises at the end of each chapter. You might also check whether you can answer any questions yet on the old final (see share/0112/). Here are some specific questions for review. They are not graded, and you do not need to write them up, but you should be able to answer such questions. We may discuss some of these in class, and I’ll ask for a few volunteers, but we will not have time for all of them. It is OK to discuss these questions and their solutions with your classmates.

1. Exercise 1.6 (DFA’s, page 84), all parts. Try to make each DFA minimal.
2. Exercise 1.7 (NFA’s, page 84), all parts. Try to make each NFA minimal.
3. Exercise 1.16 (NFA to DFA), both parts.
4. Exercise 1.18 (converting DFA languages of 1.6 to regexps).
5. Check that $\equiv_L$ (as defined above) is an equivalence relation. For the definition of “equivalence relation”, see Chapter 0.
6. Suppose $L = L(M)$ and $x, y \in \Sigma^*$. Argue that $q_M(x) = q_M(y)$ implies $x \equiv_L y$. Also, state the contrapositive. (It is useful in some of our written problems below.)
7. Define a “DA” like Sipser’s DFA, except we allow the state set $Q$ (and its subset $F$) to be infinite. Argue that every language equals $L(M)$ for some DA $M$. (Hint: try $Q = \Sigma^*$.)
Written Problems

Write up solutions to four of the following problems on paper, and get them to me (or under my office door, W426) by the due date. Most of these require you to write some sort of convincing argument; a good guide is to mimic the textbook style. You should write up what you can, even for problems that you do not completely solve, since this will earn partial credit.

In order to make this easier to grade, I will allow you, at your option, to partner with one other classmate on this problem set. If you do this, you simply need to give both names clearly on the written work; you both get the same grade. There is no penalty for working with a partner, except you are not allowed to partner with the same person again.

You may discuss the general meaning of the questions with any of your classmates, but with nobody else. You may also ask me (MG) for more specific help, and I might send out some general replies to the class by email. Do not otherwise seek or discuss solutions. Your work on these problems is governed by the Emory Honor Code.

**Problem 1.** Suppose \( L, S \subseteq \Sigma^* \). Suppose \( x \not\equiv_L y \) for every distinct pair of strings \( x, y \in S \). Argue that any DFA for \( L \) has at least \( |S| \) states. (Hint: use exercise 6.)

**Problem 2.** A binary string (in \( \{0, 1\}^* \)) is odd if it contains an odd number of 1’s. For an integer \( k \geq 1 \), let \( S_k \) be the language of binary strings whose suffix of length \( k \) is odd. (In other words: the string must have length at least \( k \), and there are an odd number of 1’s among its last \( k \) bits.) For example, \( S_2 \) contains 01 and 110, but not 1 or 00 or 111.

2(a). Draw a DFA for \( S_2 \), try to get it down to 5 states.

2(b). Argue that any DFA for \( S_k \) has at least \( 2^k \) states. (Note: I will email out a proof for a similar claim, and I want you to mimic that proof as closely as possible!)

**Bonus:** Guess a formula for the exact minimum number of states needed for \( S_k \).

**Problem 3.** Suppose \( M \) is a DFA with two distinct states \( p_1 \) and \( p_2 \). Also suppose that for all strings \( x \in \Sigma^* \), \( \delta^*(p_1, x) \in F \) iff \( \delta^*(p_2, x) \in F \). (We say that two such states are “indistinguishable”.) Show how to construct a smaller equivalent DFA \( M' \). (You do not have to prove it is equivalent, just give the construction.)

**Problem 4.** Suppose \( M \) is a DFA such that every state is reachable. That is, for each state \( p \), there is a string \( x_p \) such that \( \delta^*(q_0, x_p) = p \). Furthermore, suppose that no two states are indistinguishable (as defined in the previous problem). Argue that \( M \) is minimal. (Hint: show you can apply Problem 1 with \( S = \{x_p : p \in Q\} \).)

**Remark:** there is an efficient (poly-time) algorithm for minimizing a given DFA, based on the above ideas.

**Problem 5.** Suppose language \( L_1 \) is decided by an NFA with \( q_1 \) states, and \( L_2 \) is decided by an NFA with \( q_2 \) states. State how to construct an NFA with \( q_1 \cdot q_2 \) states, deciding \( L_1 \cap L_2 \). (This should resemble the book product construction for DFA’s, except you handle \( \epsilon \) transitions differently. You do not have to prove your construction is correct.) Draw a diagram of your construction, applied to the two NFA’s in Exercise 1.16 (page 86).