Homework #1: Regular Languages
Due: see Canvas for requirements and duedate

This handout first makes some definitions, and then provides a list of problems. See Canvas for the duedate and other requirements. In particular, you may not have to do all of these problems.

You should read the text of Chapter 1. Students in CS524 should also read the wikipedia article on the "Myhill-Nerode Theorem". For further review (beyond this assignment), look over the Chapter 1 exercises. I'm willing to discuss such exercises on Canvas.

In lecture we'll see the equivalence of DFA's, NFA's, and regular expressions. We'll delay the "pumping" topic until the second homework.

Definitions

Suppose we have a language $L \subseteq \Sigma^*$ and two strings $x, y \in \Sigma^*$. A *distinguishing* string (for x and y) is some $z \in \Sigma^*$ such that exactly one of the two strings xz and yz is in L. If there is no such z, then we say that strings x and y "*L-equivalent*", and we denote this relation by $x \, R_L \, y$.

Suppose $M = (Q, \Sigma, \delta, q_0, F)$ is a DFA (page 53). 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 read string x. Suppose we have two states $p, q \in Q$. A *distinguishing* string (for p and q) is some $z \in \Sigma^*$ such that exactly one of the two states $\delta^*(p, z)$ and $\delta^*(q, z)$ is in F (i.e. exactly is final). If there is no such z, then we say the states p and q are "*M-equivalent*", and we denote this relation by $p \, R_M \, q$.

It turns out R_L is an equivalence relation on Σ^*, and R_M is an equivalence relation on Q. (Seen in CS224?)

Define the function $q_M : \Sigma^* \to Q$ by the rule $q_M(x) = \delta^*(q_0, x)$. That is, $q_M(x)$ is the state M reaches if it starts in its initial state q_0, and reads the input string 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 Σ, and they accept the same language. A DFA is *minimal* if there is no equivalent DFA with fewer states.

Problems

You may not have to do all of these, see Canvas for details. For each Problem, you may assume that the previous problems are true, even if you did not solve them.

Problem 1 Exercise 1.4 (DFA intersections, page 84), parts (c) and (e). You should mimic Sipser's model solutions (pages 94-95). That is, first draw the two "small" machines, and then draw their grid-like "product" machine. You do not have to "minimize" the product machine.

Problem 2 Exercise 1.6 (DFA's, page 84), parts (e), (f), (g), (h), (i). Each DFA should be minimal.

Note: to verify that a DFA is minimal, check that each pair of states has a distinguishing string (you don't have to show this work).

Problem 3 Exercise 1.7 (NFA's, page 84), parts (b), (c), (d), (e), (g).

Problem 4 Suppose M is a DFA, p and q are states, and $a \in \Sigma$ is an input symbol. Argue that if $p \, R_M \, q$, then $\delta(p, a) \, R_M \, \delta(q, a)$. (Advice: it may be easier to argue the contrapositive direction, because then you have a specific distinguishing string z.)

Problem 5 Suppose M is a DFA, $L = L(M)$, and x and y are strings in Σ^*. Argue that if $q_M(x)\, R_M\, q_M(y)$, then $x\, R_L\, y$. (Advice: like the previous, try the contrapositive.)

Problem 6 Suppose $L, S \subseteq \Sigma^*$. Suppose that for every pair of distinct strings x and y in S, there is a distinguishing z (in other words, x and y are not L-equivalent). Argue that any DFA for L has at least $|S|$ states. (Hint: use Problem 5 here.)

Problem 7 Define a "DA" (deterministic automaton) just like Sipser's DFA, except we allow the state set Q (and its subset F) to be infinite. Argue that *every* language L equals $L(M)$ for some DA M. (Try $Q = \Sigma^*$. For simplicity, you may suppose $\Sigma = \{a, b\}$.)

Problem 8 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.

Problem 8(a). Draw a DFA for S_2, try to get it down to 5 states.

Problem 8(b). Argue that any DFA for S_k has at least 2^k states. (Hint: Use Problem 6 with $S = \{0,1\}^k$.)

Problem 9 Suppose M is a DFA such that every state is reachable. That is, for each state p, there is an input string x_p such that $q_M(x_p) = p$. Furthermore, suppose that every pair of states are distinguished. Argue that M is minimal. (Hint: apply Problem 6 with $S = \{x_p : p \in Q\}$.)

Remarks: Problems 5, 6, and 9 are parts of the "Myhill-Nerode theorem". You should not use that theorem in your solutions (we are proving it!). Based on similar ideas, there is also an efficient (polynomial-time) algorithm for minimizing a given DFA.