University of Virginia, Department of Computer Science

cs302 Theory of Computation Spring 2008

Problem Set 2

Due: Thursday, 7 February (2:02pm)

This problem set covers material in Sipser Chapter 1, focusing on Sections 1.2 and 1.3. Answer at least the first 7 questions, and optionally answer challenge question 8. For full credit, answers must be concise, clear, and convincing, not just correct.

If you turn in unattached sheets, we will only grade the first sheet (so, if you need more than one sheet for your answers, find a stapler before class!) You may handwrite your answers as long as your writing is legible and easily interpreted. **If you handwrite, please use a pen or sharp pencil, white paper with clean edges, and only write on the front side of the page.**

Honor Policy. For this assignment, we will use the "Tila Tequila" collaboration policy, which is similar to the "Gilligan's Island" policy from Problem Set 1, with the following clarification: you may keep notes you take during office hours or the problem-solving sessions for reference. You should, of course, make sure you understand everything in your notes before leaving office hours or the problem-solving session. If you go over problems from the assignment, or very similar problems, in these meetings, you should still follow the policy by writing up your own answers without using these notes, but you do not need to destroy your notes from these meetings, and can save them for reference. Otherwise, the policy is the same as from Problem Set 1. In particular, you should still destroy any written records from your group discussions before starting to write up your own answers.

Pledging. It is not necessary to scribble a pledge on your submission: we assume all students are honest and honorable regardless of whether or not you scribble a pledge on your submission. If you are dishonorable, however, please write a note on your submission indicating this.

Problem 1: Nondeterminstic Finite Automata. Answer these questions for the NFA N_1 in Sipser's Example 1.38. The extended transition function, δ^*, is as we defined it in class.

a. What is the set of possible states of N_1 after processing input `1`?

b. What is the set of possible states of N_1 after processing input `010`?

c. Is there a string w such that $\delta^*(q_1, w)$ does not include q_1? (If so, give the string w. If not, argue why no such w exists.)

d. Is there a string w such that $\delta^*(q_1, w) = \{q_1, q_2, q_3, q_4\}$? (If so, give the string w. If not, argue why no such w exists.)

Problem 2: Constructing NFAs. For each of the following languages, draw an NFA that recognizes the language using fewest possible number of states. For all languages, assume $\Sigma = \{0, 1\}$.

a. $\{w | w$ starts and ends with different symbols. $\}$

b. $\{w | w$ does not contain two consequtive 0s. $\}$

c. The language described by the regular expression $\{0, 1\}^* 01^*$.

Problem 3: Eliminating Epsilon. Prove that for every NFA N, there is an NFA N' that recognizes the same language as N but does not use any ϵ-transitions. (A very short, very convincing proof is possible, but you will receive some credit for a longer proof.)

Problem 4: Language Splitting. (Based on Sipser's 1.63a) Prove that any infinite regular language (that is, a regular language with an infinite number of strings) can be split into three infinite disjoint regular subsets.

Problem 5: Regularity. For each part, include a convincing proof supporting your answer.

a. Is $\{w | w$ describes a valid Sudoko puzzle $\}$ a regular language? (A Sudoko puzzle is a 9x9 grid of squares, some of which contain digits. A puzzle is *valid* if there is some way to fill in all the empty squares such that in the final grid every row, column, and 3x3 square contains exactly the digits 1-9.)

b. Define a new operation on languages, $\mathcal{D}$, as:

$$\mathcal{D}(L) = \left\{w | w \in \Sigma^* \text{ and } ww^R \in L\right\}$$

(where w^R denotes the reverse of w). Does $\mathcal{D}$ preserve regularity?

c. Define a new operation on languages, $\mathcal{X}$, as:

$$\mathcal{X}(L) = \{w | \exists z \in \Sigma^* \text{ such that } wz \in L\}$$

Does $\mathcal{X}$ preserve regularity?

d. Define an *deterministic infinite automaton* similarly to a deterministic finite automaton, except that Q is no longer required to be a finite set. Prove that DIAs can recognize non-regular languages.

Problem 6: Proving Irregularity. (Based on Sipser 1.46, but different) Prove that the following languages are not regular. You may use any technique you want, including the pumping lemma, and the closure properties we have established for regular languages in the book, class, and other problems.

a. $\{0^n 1 0^n | n \geq 0\}$

b. $\{0^n 1^m | m = 2n\}$

c. $\{w | w \in \{0,1\}^*$ is not a palindrome $\}$ (w is a palindrome iff $w = w^R$)

Problem 7: Dual Finite Automata. (The idea behind this question is from Pei-Chi Wu, Fend-Jian Wang, and Kai-Ru Young, *Scanning Regular Languages by Dual Finite Automata*, ACM SIGPLAN Notices, Vol 27, No 4, April 1992. It is not necessary to read this paper to answer this question, but you are welcome to read it if you are interested.)

Consider the problem of testing whether a long string is in some regular language: given A, a DFA recognizing some language, and w a string, determine if w is in $\mathcal{L}(A)$. The straightfoward solution is to process the string left-to-right through the DFA. This requires n steps where n is the length of the input string, and each step involves following one transition of the δ function for the DFA. Is there a way to speed up language recognition for an arbitrary regular language if we have multiple processors that can run in parallel?

The paper mentioned above proposes a method where language scanning is done using two processors: one processes the string from left-to-right using A, the other processes the string from right-to-left using A^R, a DFA that recognizes the reverse language of $\mathcal{L}(A)$. A^R is constructed using similar methods to what we saw in class: first, an NFA is constructed by reversing the edges in A and then the NFA is converts to a DFA using the subset construction. The resulting DFA, $A^R = (\mathcal{P}(Q), \Sigma, \delta', q_0', F')$ where $q_0' = F$, $F' = \{q' | q' \in \mathcal{P}(Q), q_0 \in q'\}$, and:

$$\delta'(q', a) = \left\{q | q \in Q \text{ such that } \delta(q, a) = q_x, q_x \in q'\right\}$$

When the scanners meet, we can divide w into x and y such that $w = xy$ and A has processed x and A^R has processed y^R. At this point, A is in some state in Q, $s_A = \delta^*(q_0, x)$, and A^R is in some state in $\mathcal{P}(Q)$, $s_R = \delta'^*(q_0', y^R)$.

What condition on s_A and s_R can be used to determine if w is in $\mathcal{L}(A)$?

Problem 8: Multi-Processor Scanning. *Challenge problem - you don't need to solve this problem to receive full credit on this assignment, but a good answer will be worth extra credit.*

Suppose you have thousands of available processors (as is the case, for example, on a modern graphics card) instead of just 2. Describe a method for significantly speeding up recognition for an arbitrary regular language using many processors in parallel.