CSE322: Formal Models in Computer Science Spring 2006

Problem Set 1

Due Friday, April 7, 2006, in class

Reminder: If you have not subscribed yourself to the CSE322 mailing list, please do so ASAP by following the link on the course webpage (which is `http://www.cs.washington.edu/322`).

Reading Assignment: Sipser's book, Sections 1.1 and 1.2; you should have read Chapter 0 by now.

Instructions: Information on the collaboration policy and honor code for solving problem sets can be found on the course webpage — **please read it carefully!**. In a nutshell, you are allowed to collaborate (other than problems marked with a (∗)) with fellow students taking the class to the extent of discussing solution ideas, *provided you think about each problem on your own for at least 30 minutes.* You must write down solutions on your own, and must clearly acknowledge each person with whom you discussed the solutions. You are expected to refrain from looking up solutions from web-sites, pre-existing sources from prior offerings of this course at UW or similar courses at other schools, or other literature.

Again, the questions marked with a (∗) are meant to be review questions and you are supposed to work on them *on your own.* Of course, for any problem you can talk with the instructor and/or the TAs.

There are **FIVE** questions in this assignment. Please be as clear as possible in your arguments and answers. Poorly written solutions, even if more or less correct, will be penalized.

---

1. (∗) (10 points) The reversal of a string w denoted by w^R, is the string when you "look at it backwards": for example, $homer^R = remoh$. Here is the formal inductive definition (where the alphabet is Σ):

   Base case. If $w = \epsilon$, then $w^R = \epsilon$.

   Inductive step. If $w = va$ for $v \in \Sigma^*$ and $a \in \Sigma$, then $w^R = av^R$.

   Prove by induction (on the length of y) that for all strings $x, y \in \Sigma^*$, $(xy)^R = y^R x^R$.

2. (∗) (10 points) Sipser's book, Exercise 1.3, Page 83 (Pg. 84 in 1st edition).

3. (∗) (6 points) For the machine M_2 in Exercise 1.1 (Sipser's book, Pg. 83), write out the sequence of states the machine M_2 goes through on input *abaabba* and input *bbaaba*. Which of these strings is accepted by M_2 ?

4. ($2 \times 5 = 10$ points) I mentioned in the first lecture that finite automata are used in the lexical analyzer part of a compiler. In this problem we will look at two (simple) sub-problems in the lexical analyzer that use deterministic finite automata (DFAs).

   (a) The rule for valid names for variables in C programs is the following. Variables must begin with a character (that is, a letter in the English alphabet) or underscore and maybe followed by any combination of characters, underscores, or the digits $0 - 9$. Design a

DFA that accepts strings that are valid variable names. (For simplicity assume that $\Sigma = \{\langle c \rangle, \langle d \rangle, \langle u \rangle, \#\}$, where $\langle c \rangle$ denotes a character, $\langle d \rangle$ denotes a digit, $\langle u \rangle$ denotes an underscore and # denotes any other possible ASCII character.)

(b) There are certain strings that are reserve words and not allowed as variable names. For example, **for** and **do** are reserve words in C. Build a DFA that given a string accepts it if it is same as either the reserve word **for** or **do** and rejects it otherwise. (For simplicity assume that the alphabet is given by $\Sigma = \{\mathbf{f}, \mathbf{o}, \mathbf{r}, \mathbf{d}, \#\}$, where # denotes any ASCII character which is not in the set $\{\mathbf{f}, \mathbf{o}, \mathbf{r}, \mathbf{d}\}$).

5. ($4 \times 6 = 24$ points) Give state diagrams of DFAs that accept the following languages. As documentation for your DFAs, for each state write a very brief (informal) description of the set of strings that reach that state. For all the languages the alphabet is $\{0, 1\}$.

   (a) $L_1 = \{w \mid w$ contains at least three 1s$\}$.

   (b) $L_2 = \{w \mid w$ has even number of 0's and odd number of 1's$\}$.

   (c) $L_3 = \{w \mid w$ begins with a 1, and which, interpreted as the binary representation of a positive integer, is divisible by 4$\}$.
   For this problem assume that the DFA starts reading the string from its most significant bit. For example if $w = 1000$, then w is the binary representation of the (decimal) number 8 (and thus, is in L_3).

   (d) $L_4 = \{w \mid w$ does **not** contain 111 as a substring $\}$.