

CSE 350 – Theory of Computation: Honors, Spring 2007

Problem Set #1

Due Monday, February 19, 2007

Please don't wait until the last minute to look at the problems. Please cite any collaborators and any sources. The homework is to be submitted at the beginning of the class. The material in this problem set is covered in Chapter 1 (Section 1.1,1.2) of Sipser.

Problem 1

Construct a Deterministic Finite Automata for each of the following languages on $\{0,1\}$.

- (A) Strings ending with 010 or 100.
- (B) The integer equivalent of the string is divisible by 5.
- (C) Contains the substring 1010.

Problem 2

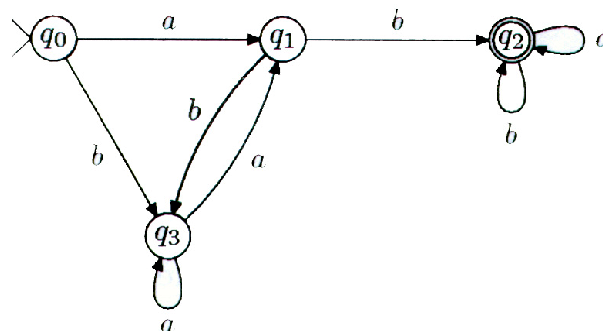
For any string $w = w_1w_2 \dots w_n$, the *reverse* of w , written $w^{\mathcal{R}}$, is the string w in reverse order, $w_n \dots w_2w_1$. For any language A , let $A^{\mathcal{R}} = \{w^{\mathcal{R}} \mid w \in A\}$.

Show that if A is regular (i.e. accepted by a Finite Automata), so is $A^{\mathcal{R}}$.

Problem 3

Minimize the DFA's constructed by you in Problem 1(A) & (B) into the minimum possible number of states.

Problem 4



Using the subset construction, convert the NFA pictured here into a DFA. There is no need to include unreachable states in your construction.

Problem 5

Prove that if a language L over $\{0,1\}$ is a regular language (i.e. accepted by a Finite Automata), then the complement of L , i.e. $\{0,1\}^* - L$, is also a regular language.

Problem 6

Suppose we define

$$\text{Prefix}(L) = \{\omega : \omega x \in L \text{ for some } x \in \Sigma^*\}$$

Prove that if L is a regular language, then $\text{Prefix}(L)$ is also regular. [*Hint*: Given a DFA M accepting L , describe how to construct a DFA M' accepting $\text{Prefix}(L)$. Then prove your construction correct i.e. prove that M' does in fact accept $\text{Prefix}(L)$.]

Problem 7 (Extra Credit)

A proper linked list consists of a set of nodes two of which, namely the root and the end are distinguished. Each node except the end has one pointer to some other node and each node except the root has pointer from some other node. For each node only the pointer to the next one is given. One can visit the nodes of the list by starting at the root and each time visiting the next node indicated by the pointer. Given a large linked list, we want to determine whether it is a proper list or not; i.e, whether it contains a loop and thus has no end.

- (1) Give a linear-time algorithm to do this, using a constant number of 2 registers. Your algorithm is not allowed to mark the list.
- (2) Give a linear-time ($O(N)$) algorithm that finds, in case the list is improper, the place where the loop starts.