CSE322: Formal Models in Computer Science Spring 2006

PROBLEM SET 2

Due Friday, April 14, 2006, in class

Reading Assignment: Sipser's book, Sections 1.2 and 1.3.

Instructions: The basic instructions are the same as in Problem Set 1.

There are **SIX** questions in this assignment. Two of them are bonus questions (one of them is due in *two* weeks). The bonus question scores will be maintained (and marked) separately from the total homework score. This problem set is longer (and possibly harder) than the first one– so start working on the problems early!

---

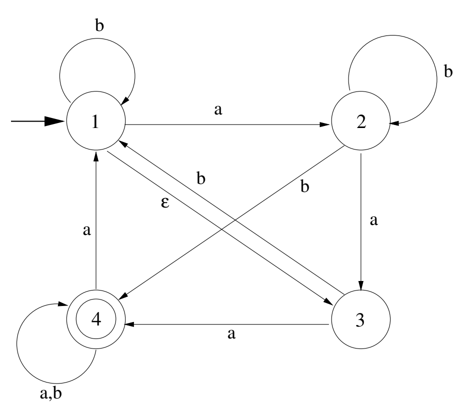
1. (*) (10 points) Given two strings x and y of exactly the same length, we can create a new string called $shuffle(x, y)$ that consists of characters of x and y alternating one after another starting with the first character of x. That is, if $x = x_1 \dots x_k$ and $y = y_1 \dots y_k$, then $shuffle(x, y) = x_1y_1x_2y_2 \dots x_ky_k$. For languages A and B, define

$$SHUFFLE(A, B) = \{shuffle(x, y) \mid x \in A, y \in B \text{ and } |x| = |y|\}.$$

Given DFAs that accept A and B, give an intuitive description and then a formal description of how to build a DFA that accepts $SHUFFLE(A, B)$.
(Note that in the above we did not specify that A and B have the same alphabet. Also note that the DFA for $SHUFFLE(A, B)$ only gets one symbol at a time, that is, on input $x_1y_1x_2y_2 \dots x_ky_k$; it reads it as $x_1, y_1, x_2, y_2 \dots, x_k, y_k$ and not (for example) in pairs like $x_1y_1, x_2y_2, \dots, x_ky_k$).

2. ($2 \times 5 = 10$ points) Sipser Exercise 1.14, Page 85 (Ex 1.10, Pg. 85 in 1st edition).

3. (*) (16 points) Convert the following NFA into a DFA using the subset construction covered in class (only the states reachable from the start state need to be shown).



4. (10 points) (**Bonus**) An odd-NFA M is a 5-tuple $\langle Q, \Sigma, \delta, s, F\rangle$ that accepts a string $x \in \Sigma^*$ if the number of *possible* states that M could be in after reading input x, which are also in F, is an *odd* number. In other words, the set of all possible states has an odd number of states from F. Note, in contrast, a "regular" NFA accepts a string if *some* state among these possible states is a final state.

   Prove that odd-NFAs accept the set of regular languages.

5. ($3 \times 8 = 24$ points) Draw NFAs with at most 8 states that accept the following languages. Explain briefly why each of your NFAs are correct.

   (a) $L_1 = \{w \mid w \in \{0,1\}^*, w \text{ is any string except } 110 \text{ and } 101\}$.

   (b) $L_2 = \{w \mid w \in \{0,1\}^*, w \text{ contains a } 111 \text{ to the left of its last (right most) 3 symbols}\}$. For example, $111011, 0111000 \in L_2$ but $111, 100000 \notin L_2$.

   (c) $L_3 = \{w \mid w \in \{e, a, t, \#\}^*, w \text{ contains either } eat \text{ or } ate\}$.

6. ($2 \times 10 + 10 = 30$ points) In this problem you will prove that regular languages are closed under certain *unary* operations. For all the three parts, assume L is a regular language.

   (a) Prove that $L^R = \{x^R \mid x \in L\}$ is also regular.

   (b) Prove that $\mathsf{min}(L) = \{w \in L \mid \text{no proper prefix of } w \text{ is in } L\}$ is also regular.[1] Here is an example of the operation. If $L = \{a, ab, abb\}$, then $\mathsf{min}(L) = \{a\}$.

   (c) (**Bonus**, due April 21) Prove that $\mathsf{half}(L)$ is regular, where the operation is defined as follows:

   $$\mathsf{half}(L) = \{x \mid \text{ for some } y \text{ such that } |x| = |y|, xy \text{ belongs to } L\}.$$

   (For all of the above parts, presenting a *correct* construction of an NFA/DFA (which should include a formal description) for the languages in question, along with an informal yet convincing explanation of why the constructions work will be sufficient. Fully formal proofs of correctness using induction is *not* required.)

[1] A string x is a prefix of string y if a string z exists such that $y = xz$. Further, x is a *proper prefix* of y if $x \neq y$.