*CS172 Computability & Complexity, Spring 2009*

# Homework 1

Out: 29 Jan., 2009
Due: **5** Feb., 2009

*Note: Not all questions are of equal difficulty. Questions marked with an asterisk (*) are to be handed in. The others are for practice and will not be graded. Put your solutions to the (*) problems in the homework box on Soda level 2 by 4pm on the due date. Take time to write* clear *and* concise *answers; confused and long-winded solutions will be penalized. You are allowed to work in small groups (two or three people) to discuss the homework and gain an understanding of what's involved. But your submitted solutions must be completely your own work and your own write-up, done completely by yourself. Depending on grading resources, we might only grade a random subset of the required problems and simply check off the rest; so you are advised to attempt all questions.*

1. Construct deterministic finite automata (DFAs) that accept each of the following languages. In each case, by drawing a fully labeled transition diagram, specify the five components, $Q$, $\Sigma$, $\delta$, $q_0$ and $F$, of your DFA.

   (a) (*) The set of all 0-1 strings that begin with 0 and end with 1.

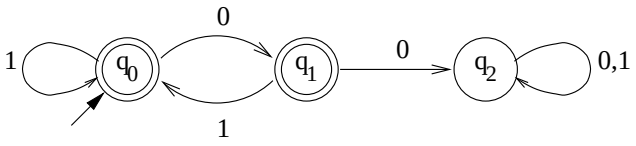
   (b) $\Sigma = \{a, b\}$. $L$ is any string *not* in $a^*b^*$.

   (c) (*) The set of all words over the English alphabet whose third-last letter is 'b'.

   (d) $\Sigma = \{a, b\}$. $L$ is the set of strings with an odd number of $a$'s and an even number of $b$'s, and doesn't contain "$ba$" as a substring.

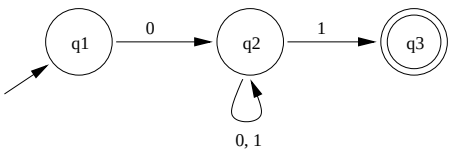
   (e) (*) The set of all 0-1 strings that contain at least two 0's and at most one 1.

2. (*) Prove carefully that the language accepted by the following DFA is the set of all 0-1 strings that do not contain a pair of consecutive 0's.




   HINT: First, for each state $q$, write down a *precise* definition of the set of input strings $w$ that cause the DFA to end in state $q$. Then prove by induction on the *length* of $w$ that these definitions are correct.

3. (*) Consider the following NFA, which accepts the same language as your DFA from problem 1(a). Convert this NFA to an equivalent DFA using the "subset construction" discussed in class. *Do not* simplify your construction.




**(OVER)**

4. (*) For any positive integer $k$, let $L_k$ denote the language of 0-1 strings in which the $k$th letter from the end is 0, i.e.,

$$L_k = \{x = x_1 x_2 \ldots x_n \in \{0,1\}^* : n \geq k \wedge x_{n-k+1} = 0\}.$$

   (a) Design an NFA with $k+1$ states that accepts $L_k$.

   (b) Design a DFA with $2^k$ states that accepts $L_k$. [HINT: Generalize your construction of problem 1(c).]

   (c) Show that the number of states in part (b) is optimal by proving that *any* DFA accepting $L_k$ must have at least $2^k$ states. [HINT: Let $x$ and $y$ be any pair of distinct strings of length $k$. Argue *carefully* that $M_k$ must end up in different states after processing $x$ and $y$.]

   [NOTE: This problem shows that, for some languages, we cannot avoid an exponential blow-up in the number of states when we convert an NFA into an equivalent DFA.]

5. Draw the state diagram of an NFA $M$ that recognizes $L$, the set of strings $\underline{w} = a^*b^*c^*$ over $\Sigma = \{a, b, c\}$. Carefully argue that $M$ really does recognize $L$.