6.045J/18.400J: Automata, Computability and Complexity	Prof. Nancy Lynch

Homework 3

Due: March 3, 2004 — *Nati Srebro, Susan Hohenberger*

Problem 1: **Distinguishable strings and indices** (From Sipser Problems 1.34 and 1.35) Let x and y be strings and let L be *any* language (not necessarily regular). We say that x and y are *distinguishable by* L if some string z exists such that exactly one of the strings xz and yz is in L. In the opposite case, if for all strings z, xz is in L if and only if yz is in L, we say that x and y are *indistinguishable* by L. If x and y are indistinguishable by L, we write $x \equiv_L y$.

(a) Show that $\equiv_L$ is an equivalence relation.

Let L be a language and X a set of strings. We say that X is *pairwise distinguishable* by L if every two distinct strings in X are distinguishable by L. Define the *index* of L to be the maximum number of elements in any set that is pairwise distinguishable by L. In other words, the index of L is equal to the number of equivalence classes in L, which may be finite or infinite.

(b) Let L_1 be the regular language $(001)^*00$. What is the index of L_1? Describe the equivalence classes.

(c) Build a DFA for L_1 with states corresponding to the equivalence classes (i.e., the number is states is equal to the index of L_1).

(d) Let L_2 be the non-regular language $\{0^n1^n : n \geq 1\}$. What is the index of L_2? Describe the equivalence classes.

(e) Now consider an arbitrary language L. Prove that if L is recognized by a DFA with k states, then L has index at most k.

(f) Again consider an arbitrary language L. For L with index k, show how to construct a DFA with k states.

We can conclude from this problem that a language L is regular if and only if it has a finite index. Moreover, its index is the size of the smallest DFA recognizing it.

Problem 2: **Algorithms for finite automata** (From Sipser Problem 4.10) Design: (a) An algorithm that determines, for any DFA M with alphabet $\{0, 1\}$, whether or not M accepts any string containing an odd number of 1s. (b) An algorithm that determines for any DFA M with alphabet $\{0, 1\}$, whether or not M accepts all strings containing an odd number of 1s.

Problem 3: **Equivalent DFAs** (From Sipser Problem 4.16) Design an algorithm to determine whether two given DFAs are equivalent by testing the two DFAs on all strings up to a certain size. Calculate a size that works.