

Homework 3

Due: March 2, 2005

Vinod Vaikuntanathan

This problem set contains some harder-than-usual problems. In solving them, you can call upon everything you have learned so far about finite-state automata and regular languages.

Problem 1: Distinguishable strings and indices (From Sipser Problems 1.51 and 1.52)

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 . On the other hand, 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.

Now 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.

Solution 1.a: For \equiv_L to be an equivalence relation, it must be reflexive, symmetric, and transitive. First, it is reflexive, since a string is indistinguishable from itself. Secondly, it is symmetric, since x being indistinguishable from y implies y is indistinguishable from x . Finally, indistinguishability is transitive. Suppose it isn't. This means that there exist strings a, b, c such that $a \equiv_L b$, $b \equiv_L c$, and yet $a \not\equiv_L c$. Since a and c are distinguishable, then there must be an extension string z such that one and only one of az and cz are in L . However, the indistinguishability of $a \equiv_L b$ and $b \equiv_L c$, implies that bz must be simultaneously in and not in L (i.e., indistinguishable from both a and c). Obviously, this is not possible.

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.

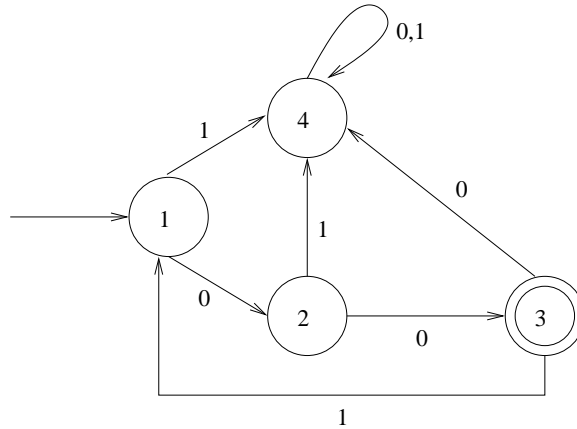
(b) Let L_1 be the regular language $(001)^*00$. What is the index of L_1 ? Describe the equivalence classes.

Solution 1.b: The index is 4. The equivalence classes are:

1. $(001)^* = \{\varepsilon, 001, 001001, \dots\}$ - strings two zeros from being in L_1
2. $(001)^*0 = \{0, 0010, 0010010, \dots\}$ - strings one zero from being in L_1
3. $(001)^*00 = \{00, 00100, 00100100, \dots\}$ - strings in L_1
4. $\{1, 10, 11, 100, 110, \dots\}$ - strings not in L_1

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

Solution 1.c:



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

Solution 1.d: The index is countably infinite. There is one unique equivalence class for each string 0^n and $n \in \mathbb{N}$, since any two strings $0^i, 0^j$ where $i \neq j$ can be distinguished by L_2 via the string 1^i . (There are also unique equivalence classes for strings not in L_2 and ε .)

(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 .

Solution 1.e: Suppose not; assume that L has index $i > k$, then by the pigeon-hole principle two distinguishable strings (i.e., from different equivalence classes) must reach the same state in the DFA. Since it's a DFA, the computation on any string z that extends x and y must end in the same state. (Recall that the transition function δ^* takes as input only the current state and a string to be read *from that point*.) Since xz and yz must end in the same state for all extensions z , then either both strings are accepted or both rejected. This contradicts the assumption that they are distinguishable.

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

Solution 1.f: We will create a DFA $M = (Q, \Sigma, \delta, q_0, F)$ as follows.

1. Q : Create k states, one for each equivalence class in L .
2. Σ : let this be the alphabet for L .
3. δ : For each state in Q , and each symbol in Σ , draw a transition arrow to the state corresponding to the new equivalence class once the symbol is added *to any string in the equivalence class for Q* (possibly a self-loop).
4. q_0 : the state in Q corresponding to the equivalence class containing ε .
5. F : the set of states in Q corresponding to the equivalence classes that contain strings in L .

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: Inequivalent DFAs Suppose that two DFAs $M_1 = (Q_1, \{0, 1\}, \delta_1, q_01, F_1)$ and $M_2 = (Q_2, \{0, 1\}, \delta_2, q_02, F_2)$ over alphabet $\{0, 1\}$ recognize different languages.

- (a) In terms of the sizes of the state sets Q_1 and Q_2 , determine an upper bound u on the length of the smallest string on which machines M_1 and M_2 must give different answers (accept vs. reject). That is, determine some u such that M_1 and M_2 must actually give different results for some string of length $\leq u$.
- (b) Prove your answer to (a).

Solution 2: We provide two possible solutions. First, we analyze the algorithm for deciding EQ_{DFA} given in class (Theorem 4.5 in Sipser). Given two DFAs A and B , we build a third DFA C that contains their symmetric difference (i.e., all strings that are accepted by one and only one of A and B). Now, if A and B are equivalent their symmetric difference must be empty. A simple construction for C is:

$$L(C) = (L(A) \cap \overline{L(B)}) \cup (\overline{L(A)} \cap L(B)).$$

How many states are in C ? Let $|C|$ denote the number of states in C . DFA intersection and union can be done using the product construction. DFA complement does not change the number of states. Thus, the two intersections in C are of size $|A||B|$ and $|A||B|$, while their union construction is $|C| = |A|^2|B|^2$.

Now, we know that if C accepts any string in Σ^* , then it must accept a string of length $|C|$ or less. Thus, we can test C on all strings of length $|A|^2|B|^2$ or less, and accept only if *none* of the strings are accepted; otherwise, reject.

Can we think of a more efficient solution than above? Given DFAs A and B , let's use the product construction to make a new DFA C where we will mark all "double reject" (i.e., all states in C containing rejecting states in *both* A and B) and "double accept" (i.e., all states in C containing accepting state in *both* A and B) states as rejecting states. We set all other states in C to be accepting (i.e., all states in C where the output of A and B would disagree). Again, we wish to test that this new DFA C recognizes only the empty language. Since this time we only used one product construction, $|C| = |A||B|$ and thus, we can test for non-emptiness by simulating the DFA on all strings of length $|A||B|$ or less. We accept only if *none* of the strings are accepted; otherwise, reject.