

CS385-Assignment 8

Due Monday, December 10

1 Decidable Problems on Finite Automata

Problems 4.15, 4.16 from the text.

Comments: In 4.15, you simply have to describe an algorithm for solving the problem. In 4.16, the author essentially tells you what the algorithm is, and you just have to figure out at what point you can stop and say that the two languages are equal. Unless I'm missing something, it is probably an error to try to work directly with the regular expression in 4.15—instead you should convert to a different representation.

2 Problems About Turing Machines

1. Problem 5.9 from the text.

2. Consider the problem of determining, given a single-tape Turing machine M and a tape symbol a , whether M , when started on a blank tape, every writes the symbol a . Show that this problem is undecidable.

Comment. In both these problems, the solution is a reduction from A_{TM} . In 5.9, it's already in the book, if you know where to look!

3. Problem 5.13 from the text.

3 Problems About Time Complexity and NP-Completeness

The textbook shows that the problem *COMPOSITES* is in *NP*, and in fact, it is now known to be in *P*. An outstanding open problem is whether it is possible to find the prime factors of an integer, given in binary, in polynomial time. It is believed that no polynomial time algorithm exists, and in fact, the security of many cryptographic systems in wide use depends on this belief. This problem (*FACTORIZATION*) does not have an obvious formulation as a language (this is what we will address below). It should be noted that the polynomial-time algorithm for *COMPOSITES* gives no hint about the factors of the number.

Consider the language L consisting of all pairs strings $\langle x, y \rangle$, where x and y are binary encodings of integers, such that y has a factor z with $1 < z < x$. For

example $\langle 7, 25 \rangle$ (with the integers encoded in binary) belongs to L , because $5 < 7$ and 5 is a factor of 25.

(a) Show that L is in NP .

(b) Show that if $P = NP$, then factoring can be done in polynomial time. (That is, if you are given an n -bit number, it can be factored in time polynomial in n .) HINT: If $P = NP$, then there is a polynomial-time algorithm for testing membership in L . Couple this with binary search to find a polynomial-time algorithm for factoring. Give an example to illustrate how your algorithm works.