CSE 531: Computability and Complexity — Autumn 2003
Problem Set #**0** — Instructor: Venkatesan Guruswami
September 29, 2003

Does not have to be turned in.

Course webpage: `http://www.cs.washington.edu/education/courses/531/03au/`
Important: Subscribe to CSE 531 email group ASAP by visiting
`http://majordomo.cs.washington.edu/mailman/listinfo/cse531`.

Note: The goal of this problem set is to aid you in reviewing material on finite automata, regular languages, and context-free grammars. Though we will not cover this material in lecture, background on these topics is necessary for the some of the stuff we will cover in the class. This problem set will also serve as a good warm-up for the skills you will need in this class. So, while you do not have to turn it in, I strongly urge you to work through this problem set *at your own pace.*

---

1. Problem 1.24, Sipser's book (If A is regular, then so is $A^{\mathcal{R}}$.)
2. Problem 1.35, Sipser's book (A language is regular iff it has finite index.)
3. Problem 1.44, Sipser's book (Exponential blow-up is necessary when converting NFAs to DFAs.)
   - Here is a suggestion for such a language:
     $E_n = \{x \in \{0,1\}^* : x \text{ has a 1 in the } n\text{'th position from the right } \}$.
4. Give algorithms for answering the following questions about finite automata. (For a DFA M, $L(M)$ is the language of strings accepted by M.)
   (a) Given a DFA M, is $L(M) = \Sigma^*$ (i.e. does M accept *every* string)?
   (b) Given two DFAs M_1 and M_2, is $L(M_1) \subseteq L(M_2)$?
   (c) Given two DFAs M_1 and M_2, is $L(M_1) = L(M_2)$?
5. (a) Prove that the intersection of a context-free language with a regular language is context-free.
   (b) Give an example to show that context-free languages are **not** closed under intersection. Deduce that CFLs are not closed under complementation.
6. Given a context-free grammar G, give an algorithm to decide if $L(G) = \emptyset$.
7. Given a context-free grammar G, give an algorithm to decide if an input string x belongs to $L(G)$. Your algorithm should run in time which is bounded by a fixed polynomial in $|x|$, the length of x. (Hint: Use dynamic programming.)
8. Problem 2.9, Sipser's book. (Ambiguous grammars.)