CSE 531: Computability and Complexity — Autumn 2003

Problem Set #1 — Instructor: Venkatesan Guruswami

Due on **October 20, 2003** in class.

Reminder: If you haven't done so already, subscribe to CSE 531 email group ASAP by visiting `http://majordomo.cs.washington.edu/mailman/listinfo/cse531`.

Instructions: You are allowed to collaborate with fellow students taking the class in solving problem sets, but you must write up your solutions on your own. If you do collaborate in solving problems, you must acknowledge for each problem the people you worked with on that problem.

You are expected to refrain from referring to any source other than Sipser's text and your class notes in coming up with your solutions. The problems have been carefully chosen for their pedagogical value and hence might be similar or identical to those given out in past offerings of this course at UW, or similar courses at other schools. Using any pre-existing solutions from these sources, or using solution material from the Web is, needless to say, strictly prohibited.

Most of the problems only require one or two key ideas for their solution – spelling out these ideas should give you most of the credit for the problem even if you err in some finer details. So, make sure you clearly write down the main idea(s) behind your solution even if you could not figure out a complete solution.

---

1. Problem 3.9, Sipser's book (The power of k-PDA's for $k = 0, 1, 2, 3$)

2. Problem 3.16, Sipser's book (A language is decidable if and only if some enumerator enumerates it in standard order)

    - Note that Sipser's book uses the terminology lexicographic order to mean the familiar dictionary order except that shorter strings precede longer strings. Thus, the lexicographic ordering of all strings over $\{0, 1\}$ is $\{\varepsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}$. We will refer to this ordering as the standard order throughout the course.

3. Problem 4.17, Sipser's book. (A language C is Turing-recognizable iff there exists a decidable language D such that $C = \{x : \exists y(\langle x, y\rangle \in D)\}$)

4. Problem 4.18, Sipser's book. (Two disjoint co-Turing-recognizable languages are separated by some *decidable* language.)

5. Define the language

$$\begin{aligned} A \;=\; \{\, \langle M\rangle \mid M &\text{ is a nondeterministic finite automaton (NFA) that only accepts strings} \\ &\text{of the form } ww \text{ for some } w \in \{0,1\}^* \,\} \ . \end{aligned}$$

   (Note that for $\langle M\rangle$ to be in A, it need *not* accept all strings of the form ww, but any string it accepts must be of the form ww.) Prove that A is **decidable**.

6. An *unrestricted grammar* (or a *rewriting system*) is a 4-tuple $G = (V, \Sigma, R, S)$ where

    - V is an alphabet;
    - $\Sigma \subset V$ is the set of *terminal* symbols, and $V - \Sigma$ is called the set of *nonterminal* symbols;

- $S \in V - \Sigma$ is the *start* symbol; and
- R, the set of *rules*, is a finite subset of $(V^*(V - \Sigma)V^*) \times V^*$.

(Thus the "only" difference from context-free grammars is that the left-hand sides of rules need not consist of single nonterminals.) Let us write $\alpha \to \beta$ if $(\alpha, \beta) \in R$; and let's define $u \Rightarrow_G v$ iff, for some $w_1, w_2 \in V^*$ and some rule $\alpha \to \beta \in R$, $u = w_1 \alpha w_2$ and $v = w_1 \beta w_2$. Let $\overset{*}{\Rightarrow}_G$ denote the reflexive, transitive closure of $\Rightarrow_G$. We say that a string $w \in \Sigma^*$ is generated by G if and only if $S \overset{*}{\Rightarrow}_G w$. Finally, $L(G) \subseteq \Sigma^*$, the *language generated by* G, is defined to be the set of all strings in Σ^* generated by G.

Your exercise is now to prove that a language is generated by an unrestricted grammar if and only if it is Turing-recognizable.

(Suggestion: Using nondeterminism and/or multiple tapes might aid in constructing Turing machines to simulate a grammar. For the other direction, to simulate a Turing Machine M by a grammar, try to construct a grammar whose rules simulate backward moves of M, and whose derivations will consequently simulate backward computations of M.)

7. * **(Optional problem)** Show that single-tape TMs that cannot write on the portion of the tape containing the input can only recognize regular languages. (Problem 3.17, Sipser's book.)