The University of North Carolina at Greensboro
CSC 653: Advanced Theory of Computation
Prof. Stephen R. Tate

Handout 9
October 30, 2008

Practice Problems 2

These problems are "practice problems" to prepare for the second exam. There are more problems here than will be on the exam, but they are approximately the same difficulty and depth of the problems that will be on the exam — as a result they are easier than many of the homework problems (where you have time to think about them), but they are also not just "memorize some facts to repeat" questions!

Students will work out and present solutions to as many of these problems as we can get to in class on Tuesday, November 4, and we will discuss the solutions. Be prepared for that!

1. (Textbook Exercise 5.1) Show that EQ_{CFG} is undecidable.

2. (Similar to Textbook Exercise 5.4) If $A \leq_m B$ and B is a regular language, does that imply that A is a regular language? Give sound reasoning for your answer, and if the answer is "no" give a language A that serves as an example of why this isn't true.

3. Consider the problem of determining if a Turing machine ever writes a "1" on its tape. Formulate this problem as a language and show that it is undecidable.

4. Show that there is an undecidable subset of $(11)^*$ (strings with an even number of 1's).

5. Consider the problem that takes a description of a two-tape Turing machine, a string w, and an integer k and asks whether this machine ever has k or more non-blank symbols on the second (non-input) tape when processing w. Formulate this problem as a language and show that it is undecidable.

6. Consider the problem of determining if two Turing machines both halt on the some input. Formulate this problem as a language and show that it is Turing-recognizable, but not Turing-decidable.

7. In programming, "dead code" refers to parts of a program that are never executed (for example, parts of the program that were accidentally left in after revisions made them obsolete). Show that it is impossible to write a program that can identify all "dead code" by defining an appropriate language and showing that it is undecidable. (In addition to the standard undecidable problems used repeatedly in the book, you may use the result from Problem 5.13 that you did in your homework related to finding useless states in a TM.)

8. (This problem has a very simple answer, but it's probably too tricky to put on an exam — think about it anyway and see what you can come up with!) Give an example of a language L such that neither L nor $\overline{L}$ are Turing-recognizable.

9. Consider the language *3COLOR*$= \{\langle G \rangle \mid$ the nodes of G can be colored with three colors such that no two nodes joined by an edge have the same color $\}$. Show that *3COLOR* $\in NP$.

10. (Textbook 7.11) Call graphs G and H **isomorphic** if the nodes of G may be reordered so that it is identical to H. Let *ISO* $= \{\langle G, H \rangle \mid G$ and H are isomorphic graphs$\}$. Show that *ISO* $\in NP$.

11. Joe the Programmer (no relation to Joe the Plumber) writes a piece of software that reads in a boolean expression and efficiently (meaning polynomial time) determines if there is a way of assigning truth values to the variables so that the expression evaluates to true. What are the consequences of this? Explain your answer with solid reasoning and using appropriate terminology.

12. Consider the language *MIN-FACTOR* $= \{\langle n, k \rangle \mid n$ has a non-trivial factor $\leq k\}$ (a "non-trivial factor" is a factor other than 1 or n). Show that *MAX-FACTOR* $\in NP$. Show that if *MAX-FACTOR* $\in P$ then you can calculate all the prime factors of a number n in polynomial time.

13. (Textbook Problem 7.17) Show that if $P = NP$, then every language $A \in P$, except $A = \emptyset$ and $A = \Sigma^*$, is NP-complete.

14. Define the language *SET-PARTITION* $= \{\langle S \rangle \mid S = \{x_1, \ldots, x_k\}$ and for some $T \subseteq S$, we have $\sum_{x \in T} x = \sum_{x \notin T} x\}$. Show that *SET-PARTITION* is NP-complete. (You may use the fact that *SUBSET-SUM* is NP-complete, since this was shown in Theorem 7.56.)

15. Consider the language *3CLIQUE* $= \{\langle G \rangle \mid G$ is a graph that contains a clique of size $3\}$. Recall that *CLIQUE* is NP-complete — what can you say about the complexity of *3CLIQUE*? Justify (prove) your answer.

16. (Textbook Problem 7.22) Let *HALF-CLIQUE* $= \{\langle G \rangle \mid G$ is an undirected graph with m nodes having a complete subgraph with at least $m/2$ nodes$\}$. Show that *HALF-CLIQUE* is NP-complete. (You may use the fact that *CLIQUE* is NP-complete.)