6.045J/18.400J: Automata, Computability and Complexity Nancy Lynch, Nati Srebro

Homework 5.5 (FAKE)

Due: Never *Susan Hohenberger*

This "fake" homework is intended as a study guide covering the material on lectures 12 (Reducibility, Rice's Theorem), 13 (Recursion Theorem) and 14 (Counter Machines).

Readings: Sections 5.3, 6.1; Handouts 6, 10.

Problem 1: (Mapping Reducibility)
Define *Total* to be the language $\{< M > \mid M$ is a Turing machine and $L(M) = \{0,1\}^*\}$, that is, the set of representations of (basic) Turing machines that accept all strings of 0s and 1s. Use mapping reducibility to prove the following statements. Cite explicitly any theorems about mapping reducibility that you use.

1. The complement of *Total* is not Turing-recognizable.
2. *Total* is not Turing-recognizable.

Problem 2: (Rice's Theorem Applications)
Which of the following are decidable and which are undecidable? Give proofs. Use Rice's Theorem wherever possible, in showing undecidability.

1. $\{< M > \mid M$ is a Turing machine and M has more than 50 states$\}$.
2. $\{< M > \mid M$ is a Turing machine and $L(M)$ is recognized by some Turing machine M' with more than 50 states$\}$.
3. $\{< M > \mid M$ is a Turing machine that accepts every palindrome (and possibly other words)$\}$.
4. $\{< M_1, M_2 > \mid M_1$ and M_2 are Turing machines and $L(M_1) \subseteq L(M_2)\}$.

Problem 3: (Generalization of Rice' Theorem)
Prove the following generalization of Rice's Theorem: If β is a non-trivial property of pairs of recognizable languages, then

$$A_\beta = \{< M, N > \mid M \text{ and } N \text{ are Turing Machines and } \beta(M,N) = TRUE\}$$

is undecidable. (Note: A property, β, is *non-trivial* if there exist Turing Machines M_1, M_2, N_1 and N_2 such that $\beta(M_1, N_1) = FALSE$ and $\beta(M_2, N_2) = TRUE$).

Problem 4: (Recursion Theorem Applications)
We saw in lecture the program (i.e., TM) $SELF$ that prints a *single* description of itself (see page 201). The goal of a computer virus is often to make *multiple* copies of itself. Let's (theoretically!) explore how to write such a program – i.e., a Turing machine that prints many copies of itself.

1. For our first attempt at "virus" writing, suppose we write a program V_1 that runs $SELF$ many times (i.e., a TM that simulates $SELF$ many times). What does the output of V_1 look like? Why does V_1 not fulfill our requirement of a program that prints many copies of itself?
2. Describe how to use the recursion theorem to get a machine R that prints its own description k times, where k is the input to R. Describe what T,t,R and r would be relative to the formal definition on page 200.

Problem 5: (More Recursion Theorem)
Let $A_\epsilon = \{< M > \mid M$ is a Turing machine and M accepts the empty string $\epsilon\}$ be the language of descriptions of machines accepting the empty string. In homework 5 problem 1.1, we demonstrated that A_ϵ was undecidable by reducing from A_{TM} (or some other undecidable language). Alternatively, prove, **using the recursion theorem**, that A_ϵ is undecideable.

Problem 6: (Counter Machine Examples)
Informally but clearly describe counter machines that recognize the following languages. Use as few counters as you can. (You may assume that a special endmarker appears at the end of the input string.)

1. $\{a^i b^j | i \leq j\}$.
2. $\{a^i b^j c^k | i \leq j \leq k\}$.
3. $\{a^i b^j c^k | k = ij\}$.