Harvard University
Computer Science 121
Problem Set 7

Due Friday, November 12, 2009 at 1:20 PM.
Submit a single PDF (lastname+ps7.pdf) of your solutions to cs121+ps7@seas.harvard.edu
Late problem sets may be turned in until Monday, November 15, 2009 at 1:20 PM with a 20% penalty.
See syllabus for collaboration policy.

PROBLEM 1 (10 points)

Let $L_1 = \{\langle M\rangle : M \text{ accepts } \langle M\rangle\}$ and $L_2 = \{\langle M\rangle : M \text{ rejects } \langle M\rangle\}$. Prove that there is no recursive language R such that $L_1 \subseteq R$ and $L_2 \subseteq \overline{R}$. (Hint: Suppose there were, and think about the TM that supposedly decides $\overline{R}$.)

PROBLEM 2 (10 points)

Let L_1 be a language. Prove that L_1 is r.e. if and only if there exists some recursive language L_2 such that $L_1 = \{x : \text{there exists } y \text{ such that } \langle x, y\rangle \in L_2\}$. (Hint: Imagine y gives you some information about an accepting computation on x, if one exists.)

PROBLEM 3 (10 points)

A function $f : \Sigma^* \to \Sigma^*$ is *computable* if there exists a Turing Machine M that, when given $s \in \Sigma^*$ as input, halts with $f(s)$ written on the tape. The *range* of f is $\{f(x) : x \in \Sigma^*\}$. Prove that a nonempty language is r.e. if and only if it is the range of a computable function.

PROBLEM 4 (5+10+10 points)

In this problem you will define a function that grows faster than any computable function. Thus you will prove the existence of an uncomputable function directly, without relying on Turing's diagonalization argument. The **busy-beaver function** $\beta(n)$ is the the largest number of a's that can be printed by any n-state, two-symbol Turing machine that eventually halts when started from the empty tape.

(A) Show that adding more states increases the number of a's that can be written. Specifically, show that there is a constant t such that, for all natural numbers n and m,

$$\text{if } n \geq m + t, \text{ then } \beta(n) > \beta(m).$$

(B) Show that if $f : \mathbb{N} \to \mathbb{N}$ is computable, then there exists some constant k_f such that $\beta(n+k_f) \geq f(n)$ for all n. In other words, there is an $(n + k_f)$-state Turing machine M_n that writes at least $f(n)$ a's on a blank tape before halting. (*Hint:* First recall that any TM can be simulated by one with a two-symbol alphabet. Show that, for each n, there is an $(n + c)$-state TM N_n that write n a's, where c is a small positive constant independent of n. Combine such a TM with the fixed two-tape-symbol machine F which computes f. The overall constant k_f will represent the number of "extra" states c required to construct N_n plus the number of states of F.)

(C) Show that β is not computable.

PROBLEM 5 (5 + 5 points)

In the near future you're working as an engineer at Google/Microsoft/Facebook when your manager asks you to write the following two programs. Is this a problem? Why or why not?

(A) Take another program's code as input and decide if that program is implemented in the fewest possible lines of code.

(B) Take another program's code and remove all inaccessible (dead) code from it.

PROBLEM 6 (**Challenge** + 1 points)

Given a particular method of encoding a Turing machine M into a string $\langle M \rangle$, define T_w to be the Turing machine encoded by the string w, or if w is not a proper encoding of any Turing machine, then define T_w to be an arbitrary fixed Turing machine. Prove that if f is any computable function, then there exists some string x such that $L(T_x) = L(T_{f(x)})$.