Problem Set 9
CS 373: Theory of Computation

Assigned: December 3, 2013 Due on: December 10, 2013

Instructions: This homework has 3 problems that can be solved in groups of size at most 3. Please follow the homework guidelines given on the class website; submitions not following these guidelines will not be graded.

Recommended Reading: Lecture 23 and 24.

Problem 1. [Category: Comprehension+Proof] The Post Correspondence Problem (PCP) is the following. Given a set of tiles with two strings, one on the top and the other at the bottom, you want to determine if there is a list of these tiles (repetitions allowed) so that the string obtained by reading the top symbols is the same as the string obtained by reading the bottom symbols. This list is called a "match". For example, consider the set of tiles

$$\left\{ \left[\frac{b}{ca}\right], \left[\frac{a}{ab}\right], \left[\frac{ca}{a}\right], \left[\frac{abc}{c}\right] \right\}$$

If we consider the sequence of tiles

$$\left[\frac{a}{ab}\right] \left[\frac{b}{ca}\right] \left[\frac{ca}{a}\right] \left[\frac{a}{ab}\right] \left[\frac{abc}{c}\right]$$

the top string is $a \cdot b \cdot ca \cdot a \cdot abc = abcaaabc$ while the bottom string is $ab \cdot ca \cdot a \cdot ab \cdot c = abcaaabc$, is the same. However, not all sets of tiles have a match. For example,

$$\left\{ \left[\frac{abc}{a}\right], \left[\frac{ca}{a}\right], \left[\frac{acc}{ba}\right] \right\}$$

does not have a match. More formally, given

$$P = \left\{ \left[\frac{t_1}{b_1}\right], \left[\frac{t_2}{b_2}\right], \ldots \left[\frac{t_k}{b_k}\right] \right\}$$

we need to determine if there is a sequence $i_1, i_2, \ldots i_n$, where every $i_j \in \{1, 2, \ldots k\}$, such that $t_{i_1} t_{i_2} \cdots t_{i_n} = b_{i_1} b_{i_2} \cdots b_{i_n}$. Thus,

$$\text{PCP} = \{\langle P \rangle \mid P \text{ is a set of tiles that has a match}\}$$

The PCP problem is known to be undecidable; interested students can read section 5.2 of Sipser's book.

Consider $\text{AMBIG}_{\text{CFG}} = \{\langle G \rangle \,|\, G \text{ is an ambiguous CFG}\}$. Prove that $\text{AMBIG}_{\text{CFG}}$ is undecidable by reducing PCP to $\text{AMBIG}_{\text{CFG}}$. *Hint:* Given an instance of PCP

$$P = \left\{ \left[\frac{t_1}{b_1}\right], \left[\frac{t_2}{b_2}\right], \ldots \left[\frac{t_k}{b_k}\right] \right\}$$

construct a CFG G with rules

$$\begin{aligned}
&S \to T \mid B \\
&T \to t_1 T a_1 \mid \cdots \mid t_k T a_k \mid t_1 a_1 \mid \cdots \mid t_k a_k \\
&B \to b_1 B a_1 \mid \cdots \mid b_k B a_k \mid b_1 a_1 \mid \cdots \mid b_k a_k
\end{aligned}$$

where $a_1, \ldots a_k$ are new terminal symbols. Prove that this reduction is correct. **[10 points]**

Problem 2. [Category: Proof] Let $A, B \subseteq \{0,1\}^*$ be r.e. languages such that $A \cup B = \{0,1\}^*$ and $A \cap B \neq \emptyset$. Prove that $A \leq_m (A \cap B)$. You may assume that x_0 is a string in $A \cap B$. **[10 points]**

Problem 3. [Category: Proof] [Matt's Problem] Infinite lists are data structures that are supported by many programming languages like Haskell, Python, and Scheme. How easy is it to sort such lists? We will investigate that in this problem.

Observe that any list is just a function that given a position (in the list) returns the element at that position. Thus, infinite lists in these programming languages can be mathematically defined as follows. A Turing machine (or program) M is an *infinite list* if M computes a function $f_M : \mathbb{N} \to \mathbb{N}$, i.e., given any $n \in \mathbb{N}$ as input, M halts with $f_M(n)$ on its tape. For any Turing machine M that is an infinite list, we will denote the "list" (or the function) it represents by f_M. Finally, note that because of the well-ordering property of natural numbers, for any $f : \mathbb{N} \to \mathbb{N}$, the value $\min_{n \in \mathbb{N}} f(n)$ exists.

1. Let
$$\text{MIN} = \{\langle M, m\rangle \mid M \text{ is an infinite list and } m = \min_{n \in \mathbb{N}} f_M(n)\}$$
Prove that $\overline{\text{HALT}} \leq_m \text{MIN}$, where $\overline{\text{HALT}} = \{\langle M, w\rangle \mid M \text{ does not halt on } w\}$. **[4 points]**

2. For any function $f : \mathbb{N} \to \mathbb{N}$, define the sorted form of f to be the function $\widehat{f} : \mathbb{N} \to \mathbb{N}$ such that $\widehat{f}(i)$ is the $(i+1)$th smallest item in the infinite sequence of items $f(0), f(1), \ldots$. Define the language SORT as
$$\text{SORT} = \{\langle M, \widehat{M}\rangle \mid M, \widehat{M} \text{ are infinite lists and } \widehat{f_M} = f_{\widehat{M}}\}$$
That is, SORT consists of pairs of Turing machines M and $\widehat{M}$ such that $\widehat{M}$ is the sorted list corresponding to M. Show that $\text{MIN} \leq_m \text{SORT}$. **[4 points]**

3. What can you conclude about MIN and SORT based on the above observations? For each language pick from decidable, undecidable but recursively enumerable, and not recursively enumerable. **[2 points]**