Solutions for Problem Set 10
CS 373: Theory of Computation

Assigned: April 18, 2013   Due on: April 25, 2013

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]**

Solution: In order to solve the problem, all we need to prove is that the mapping described in the hint for the problem is correct. Observe that if $T \overset{*}{\Rightarrow} w$ then w is of the form $t_{i_1} t_{i_2} \cdots t_{i_n} a_{i_n} \cdots a_{i_1}$. Moreover, we can show that (a) w has a unique parse tree with root labelled T, and (b) the top string when the dominoes $i_1, i_2, \ldots i_n$ are put (in order) together is $t_{i_1} \cdots t_{i_n}$. A similar property holds for strings w derived from B, namely, (a) w is of the form $b_{i_1} b_{i_2} \cdots b_{i_n} a_{i_n} \cdots a_{i_1}$, (b) w has a unique parse tree with root labelled B, and (c) the bottom string when the dominoes $i_1, i_2, \ldots i_n$ are put (in order) together is $b_{i_1} \cdots b_{i_n}$.

Suppose the PCP instance P has a match $i_1, i_2, \ldots i_n$. In other words, $t_{i_1} \cdots t_{i_n} = b_{i_1} \cdots b_{i_n}$. Then the string $w = t_{i_1} t_{i_2} \cdots t_{i_n} a_{i_n} \cdots a_{i_1} = b_{i_1} b_{i_2} \cdots b_{i_n} a_{i_n} \cdots a_{i_1}$ has two parse trees of the form

Figure 1: Two parse trees for a match

On the other hand suppose a string w has two parse trees in G, then because of the properties stated about the strings derivable from T and B, it must be the case that that the two trees for w look like in Figure 1. Thus, w is of the form $t_{i_1} t_{i_2} \cdots t_{i_n} a_{i_n} \cdots a_{i_1} = b_{i_1} b_{i_2} \cdots b_{i_n} a_{i_n} \cdots a_{i_1}$. From the properties of strings derivable from T and B, we can conclude that $i_1, i_2, \ldots i_n$ is a match for P. This completes the proof. ■

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 $A \leq_m (A \cap B)$. **[10 points]**

Solution: Let us assume that M_A is a TM that recognizes A, and M_B is a TM that recognizes B. Let $x_0 \in A \cap B$; you can relax this assumption that you know an element in the intersection (see later). The reduction f from A to $A \cap B$ is given by the following program.

On input w
  Run M_A and M_B in parallel on w and stop when one of them accepts
  If M_B is the first to accept then return w
  else return x_0

Observe that since $A \cup B = \{0,1\}^*$, either M_A or M_B will accept w, and so the simulation in step 1 will terminate. Thus, the above program describes a computable function. We need to show that f satisfies the properties of a reduction.

Suppose M_B (is the first to) accept w. Then $w \in B$. Therefore, we have $w \in A$ iff $w (= f(w)) \in A \cap B$. On the other hand, suppose M_A is the first to accept w, then $w \in A$. Then $f(w) = x_0 \in A \cap B$. Hence, for any w, $w \in A$ iff $f(w) \in A \cap B$.

In the above construction we assumed that we know $x_0 \in A \cap B$. However, this assumption can be relaxed because the reduction can compute an element in $A \cap B$. The way to do this is to dovetail the computations of M_A and M_B on all possible strings, until we find a string that both accept. Since $A \cap B \neq \emptyset$, we are guaranteed that this dovetailing step terminates. ■

Problem 3. [Category: Proof] Prove that a language A is decidable iff $A \leq_m \mathbf{L}(0^*1^*)$. **[10 points]**

Solution: Observe that since $\mathbf{L}(0^*1^*)$ is regular (and hence decidable), if $A \leq_m \mathbf{L}(0^*1^*)$ then A is decidable, from the properties of many-one reductions.

The main challenge is in showing that if A is decidable then $A \leq_m \mathbf{L}(0^*1^*)$. Let A be decided by TM M. Consider the function f computed by the following program

```
On input w
  Run M on w
  If M accepts w then return ϵ
  else (* M rejects w *)
    return 10
```

Observe that the above program terminates on all input strings w. Next, if M accepts w (i.e., $w \in A$) then $f(w) = \epsilon \in \mathbf{L}(0^*1^*)$. On the other hand, if M rejects w (i.e., $w \notin A$) then $f(w) = 10 \notin \mathbf{L}(0^*1^*)$. Thus, the function f computed by the above program is a reduction from A to $\mathbf{L}(0^*1^*)$. ■