CS331: Homework #7

Due on April 1, 2013 at 11:59pm

Professor Zhang 9:00am

Josh Davis

Problem 1

Let $K = \{\langle M \rangle : \langle M \rangle \notin L(M)\}$. Prove that K is not Turing recognizable.

Proof. To prove that K is not Turing recognizable, we will prove this by coming up with a contradiction.

Assume that K is Turing recognizable and N is a Turing machine that recognizes it. K is the set of all encoded Turing machines that recognize the language that doesn't include their own encoding.

Using this assumption, we can see that if we take the TM that recognizes K, N, we can see there are two cases in which this property of K still holds.

Case One

If $N \in K$ then $N \notin K$. This is because if it contains itself, then it must be removed because K cannot contain the encoding of itself.

Case Two

If $N \notin K$ then $N \in K$. This is because the encoding of N must be represented in K because it is the set of all TM that don't recognize themselves.

Thus $N \in K \iff N \notin K$. This is clearly a contradiction and thus we have shown that K is not Turing recognizable. $\square$

Problem 2

Prove the following statements:

1. $L_1 \leq_m L_2$ and $L_2 \leq_m L_3$ imply $L_1 \leq_m L_3$.
2. $L_1 \leq_T L_2$ implies that $\overline{L_1} \leq_m \overline{L_2}$.

Part One

Proof. Let there be three TMs that decide the three languages. M_1 recognizes L_1, M_2 recognizes L_2, and M_3 recognizes L_3.

According to many-one reduction, if one language reduces to another language, that means there is a computable function such that the function reduces the first language to the second.

Therefore with $L_1 \leq_m L_2$, there is a computable function, f_{12} where for every w, $w \in L_1 \iff f_{12}(w) \in L_2$.

The same can be said for $L_2 \leq_m L_3$, there is a computable function, f_{23} where for every w, $w \in L_2 \iff f_{23}(w) \in L_3$.

Now to prove that $L_1 \leq_m L_2$ and $L_2 \leq_m L_3$ imply $L_1 \leq_m L_3$, we will construct a new TM, N, that recognizes $L_1 \leq_m L_3$. We will construct N using the following computable function, $f(w) = f_{23}(f_{12}(w))$:

$N =$" On input string w:

1. Run M_1 on w, if it accepts, move on, else reject
2. Compute $f(w)$ using our new computable function
3. Run M_3 on the previously computed value, output whatever M_3 outputs"

Thus we have shown that when L_1 reduces to L_2 and L_2 reduces to L_3, we can construct a new many-reduction that reduces L_1 to L_3. Thus we have proved what we sought to prove and our proof is complete. □

Part Two

Proof. Since L_1 is Turing reducible to L_2, that means L_1 is deciable relative to L_2 and there is an oracle TM, M^1, that can report whether or not any string w is a member of L_1. Likewise there is another oracle TM, M^2, that can report whether or not any string w is a member of L_2.

To show that this then implies that $\overline{L_1} \leq_m \overline{L_2}$, we can construct a new oracle TM using these existing oracle TMs. Let our new oracle TM be N and defined as follows:

$N =$" On input string w:

1. Query M^1 with w, if it answers **no**, continue, else reject
2. Next query M^2 with w, if it answers **no** accept, else reject"

This is valid because when we query the oracles, since we want the complement of the languages, we just answer the opposite of what the oracle tells us. Thus we can create a new oracle from the oracles for L_1 and L_2 that can show that $L_1 \leq_T L_2$ implies that $\overline{L_1} \leq_m \overline{L_2}$.

Thus we have proven what we wanted to and our proof is complete.

□

Problem 3

Prove that $A_{TM} \not\leq_m \overline{A_{TM}}$, where $A_{TM} = \{\langle M, w\rangle : M \text{ is a TM and } M \text{ accepts } w\}$

Proof. To prove this, we will do a proof by contradiction.

Suppose that $A_{TM} \leq_m \overline{A_{TM}}$, or A_{TM} reduces to $\overline{A_{TM}}$. This means that there is a computable function, f that on every input w, $w \in A_{TM} \iff f(w) \in \overline{A_{TM}}$. Let M be the TM that recognizes A_{TM}.

According to the above, we can construct a new TM, N that recognizes $\overline{A_{TM}}$. Let N be constructed as follows:
$N =$" On input string w:

1. Compute $f(w)$
2. Run M with w, if M rejects, then accepts and if M accepts, then reject"

As we can see, the only way to that A_{TM} reduces to $\overline{A_{TM}}$ is if A_{TM} is decidable. Since we know that A_{TM} is not decidable, we have a contradiction.

Since we have a contradiction $A_{TM} \leq_m \overline{A_{TM}}$ cannot be true and we have concluded our proof. □

Problem 4

Which of of the following PCP problems has a solution? Justify.

1. $\left\{\left[\frac{ab}{a}\right], \left[\frac{bb}{ab}\right], \left[\frac{aa}{ba}\right], \left[\frac{cc}{bc}\right], \left[\frac{aa}{ca}\right], \left[\frac{d}{cd}\right]\right\}$
2. $\left\{\left[\frac{ab}{a}\right], \left[\frac{bb}{ab}\right], \left[\frac{aa}{ba}\right], \left[\frac{c}{bc}\right], \left[\frac{aa}{ca}\right], \left[\frac{d}{cd}\right]\right\}$

Part One

One possible solution is below:

$$\left[\frac{ab}{a}\right]\left[\frac{cc}{bc}\right]\left[\frac{d}{cd}\right]$$

This is a solution because if we read across the top, we get *abccd* and if we read across the bottom we get *abccd*. Thus we have solved this given PCB problem.

Part Two

One possible solution is below:

$$\left[\frac{ab}{a}\right]\left[\frac{aa}{ba}\right]\left[\frac{bb}{ab}\right]\left[\frac{c}{bc}\right]$$

This is a solution because if we read across the top, we get *abaabbc* and if we read across the bottom we get *abaabcc*. Thus we have solved this given PCB problem.

Problem 5

Does the following PCP problem P have a solution?

$$\{\left[\frac{a}{ab}\right], \left[\frac{b}{ccc}\right], \left[\frac{c}{b}\right], \left[\frac{c}{d}\right], \left[\frac{dddd}{}\right], \left[\frac{ddde}{e}\right], \}$$

Yes, it has a solution. One possible solution is below:

$$\left[\frac{a}{ab}\right]\left[\frac{b}{ccc}\right]\left[\frac{c}{d}\right]\left[\frac{c}{d}\right]\left[\frac{c}{d}\right]\left[\frac{ddde}{e}\right]$$

This is a solution because if we read across the top, we get *abcccddde* and if we read across the bottom we get *abcccddde*. Thus we have solved this given PCB problem.