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 that means N recognizes the language of all encoded TMs that don't recognize themselves. This also means that N must now be removed from the language because it cannot contain itself. Thus N cannot be in K.

Case Two

If $N \notin K$ then that means N recognizes the language of all encoded TMS that don't recognize themselves. This also means that N, by the definition, doesn't recognize itself. Thus it must be added to K.

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.

Proof. To show that the PCP problem P has a solution, we wil show that P has a solution if and only if there exists an n such that $(3^n \mod 4) = 3$.

If $(3^n \mod 4) = 3$, then there must be n of tile 2. Since there are 3^n c's, they must be matched on the top with c's as well. This will either give zero d's on the bottom or 3^n d's as well. Thus the next tile that needs to be used is the 5th tile. The 5th tile needs to be used until $3^n \mod 4 = 3$. Then we can add the last tile, making a solution.

Likewise the opposite is similar, if there is a solution, then the number of d's must match on the top and the bottom. Since we can only add more d's to the bottom by matching c's, we must add 3 at a time. Then we need to use the 4th tile until there are only three d's left. Then we add the last tile and the solution is complete.

Since we have proven both ways, we have shown that a solution can exist iff there exists an n such that $(3^n \mod 4) = 3$.

□

Problem 6

Show that $BB(k)$ is not a computable function.

Proof. We will show that $BB(k)$ is not a computable function by proof by contradiction.

Assume that $BB(k)$ is a computable function. Since $BB(k)$ is a computable function, we can represent this function as a Turing machine. Let's name this TM, M.

According to the definition of $BB(k)$, it is able to give us the maximum number of steps for a k state TM. Using this, we can determine if a machine halts. Thus we will use $BB(k)$ to decide $HALT_{TM}$.

We will now construct a TM N that decides $HALT_{TM}$.

$N =$" On input string $\langle M, w \rangle$:

1. Decode M and count the states, let it be k
2. Compute k with the Busy Beaver TM, M, let this be n
3. Now execute M with w, counting each step along the way, let this be m
4. If M ever rejects or accepts, then *accept*.
5. If m ever exceeds n, then $reject$.

This is a contradiction because if we can solve $BB(k)$, then we can solve $HALT_{TM}$. We know that $HALT_{TM}$ is undecidable which means that $BB(k)$ must also be undecidable. Thus we have shown that $BB(k)$ is not a computable function and our proof is complete. □

This works because since $BB(k)$ gives us the max number of steps for a k state TM, there is no way for a TM to execute more steps than $BB(k)$ unless it is looping. Thus if $BB(k)$ is solvable, then $HALT_{TM}$ is decidable.