COMP-170: Homework #6

Ben Tanen - March 12, 2017

Problem 3

Prove that ACCEPTMORE is recognizable by a machine which has oracle access to $\mathrm{A_{TM}}$.

* * *

To prove ACCEPTMORE is recognizable by a machine with oracle access to A_{TM}, we will first construct a recognizer for the language $L_n = \{\langle M, n\rangle \mid M \text{ accepts } \geq n \text{ inputs }\}$. Using this recognizer and a properly phrased question for the A_{TM} oracle, we will then be able to construct a recognizer for ACCEPTMORE, proving that ACCEPTMORE is recognizable.

Consider the language $L_n = \{\langle M, n\rangle \mid M \text{ accepts} \geq n \text{ inputs }\}$. We claim L_n is recognizable. In order to prove this, we will build a machine R_L that can recognize L_n. Let R_L be defined as follows:

R_L **on input** $\langle M, n\rangle$**:**

For $i = 0 \to \infty$

Set $k = 0$

For $j = 0 \to i$

Run M on input s_j for i steps

If M accepts s_j in i steps, increment k by one.

If $k \geq n$, *ACCEPT*

We will now claim that R_L is a recognizer of L_n. To show this, consider the following cases:

1. Let $\langle M, n\rangle \in L_n$, where M accepts at least n inputs. Because M accepts at least n inputs, we know there exists some finite number of steps t for which M accepts n inputs within. Thus, we can see that after running M for t steps on these n different inputs, we will have set $k = n$. Thus, after running M on the n-th accepted input, we can see that $k = n$, causing R_L to correctly accept $\langle M, n\rangle$.
2. Let $\langle M, n\rangle \notin L_n$, where M accepts fewer than n inputs. Because M accepts fewer than n inputs, we can see that no matter what the value of i is, M will never accept n or more inputs. Thus, as i and j loop, we will always have $k < n$. Thus, since the only way to accept $\langle M, n\rangle$ is for $k \geq n$ at some point, we can see that R_L will never accept $\langle M, n\rangle$.

Based on our construction and these cases, we can see that R_L accepts all $\langle M, n\rangle \in L_n$ and that R_L doesn't accept all $\langle M, n\rangle \notin L_n$. Therefore, we can see that R_L is a recognizer of L_n.

Now, given our recognizer R_L for the language L_n, we will construct a new machine R_A^{ATM} (with oracle access) that can recognize ACCEPTMORE. Let R_A^{ATM} be defined as follows:

R_A^{ATM} **on input** $\langle M_1, M_2\rangle$**:**

For $i = 0 \to \infty$

Ask A_{TM} oracle $\langle R_L, \langle M_1, i\rangle\rangle$

If YES, continue

If NO, ask A_{TM} oracle $\langle R_L, \langle M_2, i\rangle\rangle$

If YES, for $j = i \to \infty$, ask A_{TM} oracle $\langle R_L, \langle M_2, j\rangle\rangle$

If YES, continue

If NO, *ACCEPT*

If NO, *REJECT*

We will now claim that R_A^{ATM} is a recognizer of ACCEPTMORE. To show this, consider the following cases:

1. Let $\langle M_1, M_2\rangle \in$ ACCEPTMORE, where $|L(M_1)|$ and $|L(M_2)|$ are both finite and $|L(M_1)| < |L(M_2)|$. Because $L(M_1)$ and $L(M_2)$ are finite sets, suppose $n = |L(M_1)|$ and $m = |L(M_2)|$ where $m > n$. We know, when we reach $i = n+1$, our oracle will say NO to $\langle R_L, \langle M_1, n+1\rangle\rangle$ because $|L(M_1)| < n+1$ (by definition of n), so we will then query the oracle on $\langle R_L, \langle M_2, n+1\rangle\rangle$. Because $n < m$, we know that our oracle will say YES to $\langle R_L, \langle M_2, n+1\rangle\rangle$. This will lead to continual querying of the oracle to test $\langle R_L, \langle M_2, j\rangle\rangle$ until $j = m+1$. When we query the oracle for $\langle R_L, \langle M_2, m+1\rangle\rangle$, we know we will get a NO, which will cause R_A^{ATM} to correctly accept $\langle M_1, M_2\rangle$.
2. Let $\langle M_1, M_2\rangle \notin$ ACCEPTMORE, where $|L(M_1)|$ and $|L(M_2)|$ are finite but $|L(M_1)| \geq |L(M_2)|$. Again, suppose $n = |L(M_1)|$ and $m = |L(M_2)$. We again know, when we reach $i = n+1$, the oracle will say NO to $\langle R_L, \langle M_1, n+1\rangle\rangle$, causing a query to the oracle of $\langle R_L, \langle M_2, n+1\rangle\rangle$. Because $n \geq m$, we know that the oracle will say NO to $\langle R_L, \langle M_2, n+1\rangle\rangle$ because $m < n+1$. Thus, a NO from the oracle on $\langle R_L, \langle M_2, n+1\rangle\rangle$ will cause R_A^{ATM} to correctly reject (not accept) $\langle M_1, M_2\rangle$.
3. Let $\langle M_1, M_2\rangle \notin$ ACCEPTMORE, where $|L(M_1)|$ is not finite. Since $|L(M_1)|$ is not finite, we can see that R_A will loop before finding an i such that the oracle will say NO to $\langle R_L, \langle M_1, i\rangle\rangle$. Thus, we can see that if $|L(M_1)|$ is not finite, R_A^{ATM} will correctly loop on (not accept) $\langle M_1, M_2\rangle$.
4. Let $\langle M_1, M_2\rangle \notin$ ACCEPTMORE, where $|L(M_1)|$ is finite but $|L(M_2)|$ is not finite. Similar to case #2, we can see that we will eventually loop on i until $i = |L(M_1)|+1$, where our oracle will tell us NO for $\langle R_L, \langle M_1, |L(M_1)|+1\rangle\rangle$. Next, since $|L(M_2)|$ is not finite, we can see that we will loop on j, infinitely querying the oracle with $\langle R_L, \langle M_1, j\rangle\rangle$, never reaching a point that the oracle answers NO. Thus, we can see that again R_A^{ATM} will correctly loop on (not accept) $\langle M_1, M_2\rangle$.

Based on our construction and these cases, we can see that R_A^{ATM} accepts all $\langle M_1, M_2\rangle \in$ ACCEPTMORE and that R_A^{ATM} doesn't accept all $\langle M_1, M_2\rangle \notin$ ACCEPTMORE. Therefore, we can see that R_A^{ATM} is a recognizer of ACCEPTMORE. ⊠