Lecture 32

Homework #32: 5.4.1, 5.4.2 – turn in 5.4.1 b, 5.4.2 a and d

We have a number of different ways to show that a language is decidable. We can:
- Directly present a Turing Machine that decides the language. Here we can select from any of the “complex” models that use multiple tapes, etc. We can also use simple Turing Machines to construct more complicated machines.
- Make use of the closure theorem from the previous lecture.

Today we will explore a mechanism by which we can prove that a language is not decidable. (and no, it’s not another Pumping Theorem…)

The primary technique we will explore is reduction. To show that $L_1$ is undecidable:
1. Identify $L_2$ that is undecidable.
2. Reduce $L_2$ to $L_1$. Show that if $L_1$ were decidable, $L_2$ would be as well. (Finding an algorithm for $L_2$ reduces to finding an algorithm for $L_1$.)

Thm. The following problems about Turing Machines (i.e., algorithms) are undecidable:
1) Given a Turing machine $M$ and an input string $w$, does $M$ halt on input $w$?
2) For a certain fixed machine, given an input string $w$, does $M$ halt on input $w$?
3) Given a Turing machine $M$, does $M$ halt on the empty tape?
4) Given a Turing machine $M$, is there any string at all on which $M$ halts?
5) Given a Turing machine $M$, does $M$ halt on every input string?
6) Given 2 machines $M_1$ and $M_2$, do they halt on the same input strings?
7) Given a Turing machine $M$, is the language $M$ accepts regular? context free? decidable?

Proof.
1) This is the Halting Problem. Recall proof from 1st class (which was essentially a diagonalization proof).
2) Let $L_h = \{<M,w> : M \text{ accepts - i.e., halts on - } w\}$

Now consider $M^1$, a machine that semidecides $L_h$. That is, it halts when given $<M,w>$ such that $M$ halts on $w$. It does not halt otherwise.

If we choose $M^1$ to be the fixed machine, then we can decide $L_h$.

3) Given $M$, does $M$ halt on the empty tape?

Let $L_h = \{<M,w> : M \text{ accepts - i.e., halts on - } w\}$

Suppose we could find $M_0$ to decide
\[
\{<M> : M \text{ accepts } e\}
\]

We will now show that $M_0$ could be used to decide $L_h$.

Claim: Given a Turing machine $M$ and input $w$, there is a systematic way to construct $M_w$ that operates as follows: $M_w$, when started with empty tape $(s, \#\#)$, writes $w$ and then simulates $M$ on $w$.

Now we can use $M_0$ to decide $L_h$.

Given $M$ and $w$, construct $M_w$.

Deliver $<M_w>$ as input to $M_0$.

- if $M_0$ halts with $Y$ on the tape, then $M$ halts on $w$
- if $M_0$ halts with $N$ on the tape, then $M$ does not halt on $w$. 
4) Given \( M \), is there any string at all on which \( M \) halts?

Let \( L_e = \{ <M> : M \text{ halts on } e \} \)

Suppose we could construct \( M_{\text{any}} \) to decide
\[ \{ <M> : M \text{ halts on some string} \} \]

We will now show that \( M_{\text{any}} \) could be used to decide \( L_e \).

Claim: Given a Turing machine \( M \), there is a systematic way to construct \( M_e \) that operates as follows: \( M_e \), when started with any input \( (s, \#w\#) \), erases \( w \) and then simulates \( M \) on \( e \).

Now we can use \( M_{\text{any}} \) to decide \( L_e \).

Given \( M \), construct \( M_e \).

Deliver \( <M_e> \) as input to \( M_{\text{any}} \).

if \( M_{\text{any}} \) halts with \( Y \) on the tape, then \( M \) halts on something, which means that \( M \) halts on \( e \).

if \( M_{\text{any}} \) halts with \( N \) on the tape, then \( M \) does not halt on anything, which means that \( M \) does not halt on \( e \).

5) Same as 4.

6) Given \( M_1 \) and \( M_2 \), do they halt on the same input strings?
Let \( L_{\text{all}} = \{<M> : M \text{ halts on all input}\} \)

Suppose we could find \( M_{\text{same}} \) to decide
\( \{<M_1, M_2> : M_1 \text{ and } M_2 \text{ halt on the same input strings}\} \)

We will now show that \( M_{\text{same}} \) could be used to decide \( L_{\text{all}} \).

First create a machine \( M_i \) that halts immediately on all input.

Now we can use \( M_{\text{same}} \) to decide \( L_{\text{all}} \).

Given \( M \), input \( <M, M_i> \) to \( M_{\text{same}} \).

But we know that \( L_{\text{all}} \) is undecidable. Therefore, this problem is undecidable as well.

7) See text.