CSE 431 Theory of Computation **Spring 2014**

Lecture 6: April 17

Lecturer: James R. Lee *Scribe: Patrick Larson*

Disclaimer: *These notes have not been subjected to the usual scrutiny reserved for formal publications. They may be distributed outside this class only with the permission of the Instructor.*

6.1 Overview

- **Today:** Reduction and Recursion Theorem
- **Tuesday:** Proof Systems, Logic, Godel's Incompleteness Theorems
- **Thursday:** Complexity Theory.

6.2 Reductions

Last time we use the diagonalization argument to show that

$$A_{TM} = \{\langle M, w\rangle : M \text{ a Turing Machine accepting w}\}$$

is undecidable. Using reduction, we can use this fact to show other problems are undecidable. The idea is to take a problem that we think is undecidable, assume we have a decider for it, and use that decider to decide A_{TM}, a contradiction.

Example 1:

$$HALT_{TM} = \{\langle M, w\rangle : M \text{ a Turing Machine that halts on } w\}$$

is undecidable.

Proof: Assume for contradiction that N is a Turing Machine that decides $HALT_{TM}$. Define another Turing Machine

$B =$ "
1. On input $\langle M, w\rangle$, simulate N on $\langle M, w\rangle$.
2. If N rejects, REJECT.
3. Otherwise, simulate M on w.
4. If M accepts w, ACCEPT.
5. If M rejects w, REJECT. "

Claim 6.1 (Example 1) *B decides A_{TM}*

If M accepts w, then N halts, so B simulates M on w and will accept. If M rejects w, either it does not halt, and is rejected by N, or it does halt and is rejected by M. Then, $L(B) = A_{TM}$ and B halts on any input not in $L(B)$. Thus, B is a decider for A_{TM}.

This is a contradiction since A_{TM} is known to be undecidable. Thus, $HALT_{TM}$ is likewise undecidable. ■

Example 2:

$$E_{TM} = \{\langle M \rangle : M \text{ a Turing machine such that } L(M) = \emptyset\}$$

is undecidable.

Proof: We reduce to A_{TM}. Assume for contradiction that Q is a Turing Machine that decides E_{TM}. Define another Turing Machine

$B =$ " 1. On input $\langle M, w \rangle$, define R
$R =$ " 1.1 If input $x \neq w$, REJECT
1.2 If input $x = w$, simulate M on w and return the result. "
2. Simulate Q on $\langle R \rangle$
3. If Q accepts $\langle R \rangle$, REJECT.
4. If Q rejects $\langle R \rangle$, ACCEPT. "

Claim 6.2 (Example 2) *B decides A_{TM}*

Suppose M accepts w. Then R will accept w. When we simulate Q on $\langle R \rangle$, $L(R)$ is non-empty and so Q will reject, and thus B will accept.
If instead M rejects w, R will also reject w. R also rejects all other input, so $L(R)$ is empty, and Q will accept. Then B will reject.
Since B halts on all input and $L(B) = A_{TM}$, B is a decider for A_{TM}, a contradiction.

Thus, E_{TM} is undecidable. ■

Example 3:

$$EQ_{TM} = \{\langle M, N \rangle : M \text{ a Turing Machine, } N \text{ a Turing machine and } L(M) = L(N)\}$$

is undecidable.

Proof: We reduce to E_{TM}. Assume for contradiction that A is a Turing Machine that decides EQ_{TM}. Define another Turing Machine

$B =$ " 1. On input $\langle M \rangle$, define R
$R =$ " 1.1 On input x, REJECT "
2. Simulate A on $\langle M, R \rangle$
3. If A accepts, ACCEPT.
4. If A rejects, REJECT. "

Claim 6.3 (Example 3) *B decides E_{TM}*

First note that $L(R)$ is empty. Then, suppose $L(M)$ is empty. Then, $L(M) = L(R) = \emptyset$ and A accepts. Suppose $L(M)$ is not empty. Then, $L(M) \neq L(R) = \emptyset$ and A rejects. So $L(B) = E_{TM}$ and B halts on all input. Thus, B is a decider for E_{TM}, a contradiction.

Thus, EQ_{TM} is undecidable. ■

Aside Are there undecidable problems not involving Turing Machines?

Hilbert's Tenth Problem: Given an integer polynomial of more than one variable, there is no decider for the existence of an integer solution.

Post Correspondence Problem: Given a set of dominoes, there does not exist a decider for the existence of an ordering of arbitrarily many instances from that set in which the top line of the dominoes is equal to the bottom line.

6.3 The Recursion Theorem

Suppose that within a Turing Machine M, we want a step which obtains a copy of its own source code. This is non-trivial since obtaining the source code modifies the source code.

We can instead solve a simple problem. We design a Turing Machine $SELF$ which when executed prints $\langle SELF \rangle$. The existence of such a machine is also non-trivial:

```
printf("printf(\"...
```

We begin with a definition.

Definition 6.4 *A function $q : \Sigma^* \to \Sigma^*$ is computable if there is a Turing Machine M that on every input $w \in \Sigma^*$, M halts with $q(w)$ written on the tape.*

Lemma 6.5 *There exists a computable function q so that for every input $w \in \Sigma^*$, $q(w) = \langle P_w \rangle$, where P_w is a Turing machine that prints w.*

Proof: by construction.

$Q =$ "
1. On input w, design P_w as follows
   $P_w =$ "
   1.1 erase the input.
   1.2 write w on the tape.
   1.3 halt. "
2. erase the input.
3. write $\langle P_w \rangle$ on the tape "

■

Then, let $\langle SELF \rangle = \langle AB \rangle$. That is, $SELF$ is a machine that first executes A, then executes B, where A and B are defined as follows:

$A = P_{\langle B \rangle}$

$B =$ "
1. On input $\langle M \rangle$, compute $q(\langle M \rangle)$
2. Print $q(\langle M \rangle)\langle M \rangle$
3. HALT. "

Notice then that A prints the source code to B, which then prints the source code that prints B followed by the source code for B. So, $SELF$ prints $\langle SELF \rangle$.