CIS 511: Theory of Computation **Feb 2, 2017**

Lecture 7: Undecidability, Continued

Professor Sampath Kannan *Zach Schutzman*

NB: *These notes are from CIS511 at Penn. The course followed Michael Sipser's Introduction to the Theory of Computation (3ed) text.*

An Undecidable Language

Recall, we showed last time that, by a cardinality argument, that there must exist some language that is not Turing decidable, or even recognizable.

Example: Let the language $A_{TM} = \{\langle M, w\rangle | M(w) \textit{ accepts}\}$, the language of machines and words such that M reaches its q_a on input w.

Claim 7.1 *A_{TM} is not decidable.*

Proof: *Assume for the sake of contradiction that A_{TM} is decidable, and let H be the TM that decides it. We will show that H cannot exist.*

How does H work? H takes a pair $\langle M, w\rangle$ and accepts if M accepts w and rejects if M does not accept (reject or loop forever). Note that because of this 'loop forever' case, we can't just simulate M on w.

Create a new TM D that takes as input the description of a Turing machine $\langle M\rangle$. D calls H on $\langle M, \langle M\rangle\rangle$. Since H decides a language, it always terminates. Define the behavior of D to return the complement of the answer that H provides. That is, if H accepts on $\langle M, \langle M\rangle\rangle$, then D rejects. I.e., if H says that Machine #17 accepts on the input '17', then D rejects.

Now, how should D behave when given input $\langle D\rangle$? First, D calls H on $\langle D, \langle D\rangle\rangle$. Next, H decides if D accepts D. In the former case, it accepts, the latter, rejects. So, if H says that D accepts $\langle D\rangle$, then D rejects. Conversely, if H says that D does not accept $\langle D\rangle$, then D accepts. Both of these cases are paradoxical, hence no such decider H for A_{TM} exists, so A_{TM} is not a decidable language. ■

This proof is equivalent to Cantor's diagonal argument for uncountablility. Consider the rows indexed by Turing machines and the columns by string descriptions of Turing machines. The table values are either 'accept' or 'reject' when H receives input $\langle r, \langle c\rangle\rangle$. D then works by going down the diagonal and taking the complement of the entry at that point. If we think about 'accept' and 'reject' as being '1' and '0', we have exactly the same case as Cantor's diagonal argument that $\mathbb{R}$ is uncountable.

What about an unrecognizable language?

An Unrecognizable Language

Claim 7.2 *A language L is decidable if and only if L and L^C are both recognizable.*

Proof:

If L is decidable, then clearly it is recognizable. Additionally, L^C must be decidable, as it is decided by a machine that simulates a decider for L and outputs the complement of the result.

Corollary 7.3 *The class of decidable languages is closed under complement. Hence, L^C is recognizable.*

Now, assume that L and L^C are both recognizable. We will construct a decider for L. Let M and M_C be recognizers for L and L^C, respectively. Let M_D be a TM that works by alternating simulating M and M_C one step at a time. Because L and L^C are both recognizable, if a string x is in L, then M halts and accepts on x. If x is not in L, then M_C halts and accepts on x. Since M_D alternates steps of M and M_C, it halts in a finite amount of time, and outputs an answer corresponding to the submachine that accepted. M_D accepts if M accepts or M_C rejects. M_D rejects when M rejects or M_C accepts.

■

Is A_{TM} recognizable? Yes! The simple approach of simulating M on w will halt if M accepts. The issue came from us wanting a machine to halt in the case that w is not in $L(M)$.

Let's define $\overline{A_{TM}}$ as the language where M does not accept w. Is this language Turing recognizable?

Claim 7.4 *$\overline{A_{TM}}$ is not recognizable.*

Proof:

If $\overline{A_{TM}}$ is recognizable, then together with the proof that A_{TM} is recognizable, we know that A_{TM} would have to be decidable. We know this is false, so $\overline{A_{TM}}$ is not recognizable. ■

We found a language that is not recognizable!

Reductions

The idea of reductions is ubiquitous in mathematics and computer science. Fundamentally, it is the idea of solving a new problem by converting it to one you already know how to solve.

Suppose we have two languages A and B.

Definition 7.5 *A reduction from A to B is a method for using a procedure for solving/deciding/recognizing B to solve/decide/recognize A.*

We can think about a subroutine that can solve B being used to solve A.

The existence of a reduction from A to B in a sense means that A is no harder than B (B is at least as hard as A).

If we can reduce an undecidable language A to some language B, then B must be undecidable. We will denote "A reduces to B" as $A \preceq B$.

A reduction from A to B: given a TM S for solving B, construct a TM R that uses S as a subroutine to solve A.

Example: *Define $HALT_{TM} = \{\langle M, w\rangle \mid M \text{ halts on } w\}$. This is the set of pairs $\langle M, w\rangle$ such that M halts on w.*

Claim 7.6 *H_{TM} is undecidable*

Proof: *We prove this by reduction. We will show that, if we assume $HALT_{TM}$ is decidable, we can use it to decide $HALT_{TM}$. This is a reduction from A_{TM} to $HALT_{TM}$.*

Suppose S is a TM that decides $HALT_{TM}$. S takes as input $\langle M, w\rangle$ and accepts if M halts on w, rejects if M loops forever. Then a TM for A_{TM} works by taking input $\langle M, w\rangle$, asks S whether M halts on w, if S rejects, then reject, otherwise, simulate M on w and output the result.

This decides A_{TM}, which is a contradiction. Hence such an S cannot exist and $HALT_{TM}$ is not decidable.

■

Example: *Define $E_{TM} = \{\langle M\rangle \mid M \text{ recognizes } \emptyset\}$ be the language of TMs that accept no strings.*

Claim 7.7 *E_{TM} is undecidable*

Proof: *We proceed again by reduction from A_{TM} to E_{TM}.*

Let S be a decider for E_{TM}. S takes as input $\langle M\langle$ and accepts if M accepts no strings and rejects if M accepts at least one string. A machine for A_{TM} takes as input $\langle M, w\rangle$, a machine and a string. We will show that S can be used to build a decider for A_{TM}

Define a new machine M' such that M' on input $x \neq w$ rejects and on input $x = w$ simulates M on w. M' recognizes a language that is either $\emptyset$ or $\{w\}$. But M' accepts w only when M accepts w. We can then feed $\langle M'\rangle$ to S. If S accepts, then M does not accept w. Conversely, if S rejects, then M accepts w.

We can now decide A_{TM} as follows: on input $\langle M, w\rangle$, construct M' and pass it to S. Since S always halts, A_{TM} halts on all inputs, and is therefore decidable.

Because A_{TM} is not decidable, no such S can exist and E_{TM} must be undecidable.

■