CSE200 Lecture Notes – Diagonalization

Lecture by Russell Impagliazzo
Notes by Jiawei Gao
Adapted from notes by William Matthews

January 19, 2016

1 Real numbers are uncountable

Claim (Cantor)**.** There are more real numbers than positive integers.

Proof. Both of these sets are infinite, how can we argue that one is bigger than the other? Two sets A and B are the same size if and only if there exists a bijection (a 1-to-1 function) between them.

Assume, for the sake of contradiction, that there exists a bijection from real numbers in $[0,1)$ to positive integers. We represent the real numbers by their binary expressions. We will construct the following matrix $M[i,j]$ where $M[i,j]$ is the j^{th} bit of the i^{th} real number.

$$M = \begin{array}{c|cccc} & 1 & 2 & 3 & \dots \\ \hline 1 & 1 & 0 & 1 & \dots \\ 2 & 0 & 0 & 0 & \dots \\ 3 & 1 & 1 & 1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{array}$$

Consider the real number D whose i^{th} bit is $1 - M[i,i]$. There must exist a j such that $f(j) = S$. By the definition of D, the j^{th} bit of D does not equal $M[j,j]$, which gives a contradiction. $\square$

2 From integers to Turing machines

A *decision problem* is to decide whether a string is in a language. On input $w \in \{0,1\}^*$, we output 1 if accept, output 0 if reject.

Theorem 2.1. There exists an uncomputable language.

Since each Turing machine has a finite description, we can map Turing machines to positive integers. Similarly since strings have finite lengths, we can also map strings to integers. A language is just a set of strings, which we may also view as a set of integers. By a similar argument, there are more languages than Turing machines. Thus, there exist languages for which no Turing machine decides them (languages which are not recursive).

Similarly to what we did before, construct the following matrix $M[i, j]$ where $M[i, j] = 1$ iff $j \in L(M_i)$ i.e. M_i halts and accepts on input j, where M_i is the Turing machine corresponding to integer i, and j is viewed as the string corresponding to integer j.

Define the language $D = \{i \mid M_i \text{ does not accept input } i\}$, namely, $i \in D$ iff $M[i, i] = 0$.

Suppose for the sake of contradiction that there existed an integer i such that M_i decided D. Then $i \in D$ iff $M[i, i] = 0$ iff M_i never accepts i iff $i \notin D$, which contradicts our assumption. Therefore D is an uncomputable language.

3 Time Hierarchy

Definition 3.1. Function $T(n)$ is *time-constructible* if given input 1^n, we can compute the value of $T(n)$ in time $T(n)$.

Almost all functions we use are time-constructible.

For a time-constructible $T(n)$, we can extend our construction of D to construct language $D_{T(n)}$, which cannot be computed by any Turing machines in time $T(n)$. Let $D_{T(n)} = \{i \mid M_i \text{ does not accept input } i \text{ in } T(|i|) \text{ steps}\}$.

Suppose for the sake of contradiction that there existed an integer i such that M_i decided $D_{T(n)}$ in $T(n)$ steps. Then $i \in D_{T(n)}$ iff M_i does not accept i in $T(|i|)$ steps iff $i \notin D_{T(n)}$ – a contradiction.

Note that the language $D_{T(n)}$ is decidable in time $\mathsf{poly}(T(n))$, by a universal Turing machine. Let UTB be the language that accepts $\langle i, x, 1^T \rangle$ iff machine i accepts x in at most T steps. To decide if $i \in D$, we just need to query whether $\langle i, i, 1^{T(|i|)} \rangle \in UTB$, and negate the answer. Because a universal TM can simulate any TM that runs in time $T(n)$ using time $T^3(n)$, we have $D_{T(n)} \in \mathsf{TIME}(T^3(n))$.

From the argument above, we have seen that there exists a language computable in $O(T^3(n))$ steps, but cannot be computed by any TM in $T(n)$ steps. This gives a weak version of the Time Hierarchy Theorem.

Theorem 3.1 (weak version of Time Hierarchy Theorem)**.** Let $T(n)$ by any time-constructable function satisfying $T(n) \geq n$. Then

$$\mathsf{TIME}(T(n)) \subsetneq \mathsf{TIME}((T(n))^3).$$

4 Halting Problem

Since D is somewhat contrived, can we come up with a more natural language that is not computable?

Define the language $HALT = \{(i,x) \mid M_i \text{ eventually halts on input } x\}$.

Theorem 4.1. $HALT$ is uncomputable.

Recall from Section 2 that there exists uncomputable language $D = \{i \mid M_i \text{ never accepts input } i\}$. We will use it to prove $HALT$ is uncomputable.

Suppose, for the sake of contradiction that $HALT$ is computable. Thus, there exists an algorithm A that solves the halting problem. Then we can use A to decide D.

On input i:

1. Run A on (i,i).
2. If A says "No", Accept.
3. If A says "Yes", Run M_i on input i.
   (a) If M_i accepts, then reject.
   (b) Otherwise accept.

Since by assumption A decides $HALT$, we know that it always halts. We only run M_i on input i when $(i,i) \in HALT$, so we know that M_i will halt. Second, we argue that our TM decides D. Each input i is accepted if and only if either M_i doesn't halt on i or M_i halts and rejects i. This corresponds exactly to $i \notin L(M_i)$, and therefore $i \in D$.

5 Binary Universal Turing Machines

Let language $BUTB = \{\langle i,x,T\rangle \mid M_i \text{ accepts } x \text{ in } T \text{ steps}\}$, but T is written in binary. So the input length n becomes logarithmic to T, the running time becomes $\mathsf{poly}(|i|,|x|,T) = 2^{O(n)}$.

Let $L \in \mathsf{TIME}(2^{3n}) - \mathsf{TIME}(2^n)$. Let M be a TM solving L in 2^{3n} steps.

If I could solve $BUTB$ in $2^{O(n)}$ time, I could solve L in $2^{O(n)}$ time: on input x, query if $\langle M,x,2^{3n}\rangle$ is in $BUTB$, and return that answer.