Lecture 9: The problem with using Diagonalization to prove $\mathbf{P} \neq \mathbf{NP}$.

Anup Rao

October 25, 2018

The only way we know how to prove lower bounds on the running time of Turing Machines is via diagonalization. Can we hope to show that $\mathbf{P} \neq \mathbf{NP}$ by some kind of diagonalization argument? In this lecture, we discuss an issue that is an obstacle to finding such a proof.

Definition 1 (Oracle Machines)**.** *Given a function* $O : \{0,1\}^* \to \{0,1\}$*, an* oracle-machine *is a Turing Machine that is allowed to use a special oracle tape to make queries to* O*. Each query to* O *takes unit time.*

We can define $\mathbf{P}^O$, $\mathbf{NP}^O$ as functions computable in poly time (resp nondeterministic poly time) with oracle access to O.

Then we have the following theorem:

Theorem 2. *There exists an oracle* A *such that* $\mathbf{P}^A = \mathbf{NP}^A$*, and an oracle* B *such that* $\mathbf{P}^B \neq \mathbf{NP}^B$*.*

The theorem gives a hint about one of the ways in which it will be hard to determine whether or not $\mathbf{P} = \mathbf{NP}$. Any such proof must not work in the *relativized* worlds where access to A, B is permitted. On the other hand, the kinds of proofs that we have seen using diagonalization *do relativize*—the same argument would work even if the machines have oracle access to some oracle O.

Proof Let A be the function that on input α, x outputs 1 if and only if $M_\alpha(x)$ outputs 1 in $2^{|x|}$ steps. Then $\mathbf{P}^A = \mathbf{EXP}$, since every exponential time computation can be simulated with access to A, and every query to A can be simulated in exponential time. Also $\mathbf{NP}^A = \mathbf{EXP}$, since in exponential time we can simulate all queries to A and simulate all nondeterministic choices.

The second part is more interesting. We shall define an oracle $B : \{0,1\}^* \to \{0,1\}$ and a function $f \in \mathbf{NP}^B$ such that $f \notin \mathbf{P}^B$. f is defined in terms of B as follows:

$$f(x) = \begin{cases} 1 & \text{if there exists } y \text{ such that } |y| = |x| \text{ and } B(y) = 1, \\ 0 & \text{else.} \end{cases}$$

We first show that $f \in \mathbf{NP}^B$: a non-deterministic machine can guess y of the same length as x, and make a single query to verify that $B(y) = 1$.

To define B, we shall use diagonalization. Let $M_1, M_2, \ldots, M_i, \ldots,$ be an enumeration of all machines that query B. Our goal is to make sure that the $i'th$ machine fails to compute the correct value of $f(x)$ in time $2^{n/10}$, for some n where $n = |x|$. To do this we define the value of B gradually. We define the value of B in phases. After each phase, we shall have defined the value of B on a finite set of strings.

In Phase i, let t be so large that the value of B is not yet defined on each string of length t. Then run the i'th machine $M_i(1^t)$ for $2^{t/10}$ steps. Each time M_i queries a string of B whose value has not yet been defined, return 0 and define the value of B on that string to be 0. If M_i halts with value 1, then set B to be 0 on all strings of length t. If M_i halts with value 0, then pick a string y of length t that $M_i(1^t)$ did not query (note that such a string always exists since there are 2^t binary stings of length t and M_i did not take more than $2^{t/10}$ steps), and set $B(y) = 1$.

Set the value of B on strings that are not defined by the above process to be 0.

Suppose for the sake of contradiction that $f \in \mathbf{P}^B$. Then consider the machine M that computes f. Let i be the index such that the i'th machine in the enumeration is M and t be such that $M_i(1^t)$ was used to define B on strings of length t during the i'th phase. Clearly, $f(1^t) \neq M(1^t)$ and hence M does not compute f. ■