CS 388T: Advanced Complexity Theory Spring 2020

Realtivization Barrier

Prof. Dana Moshkovitz *Scribe: Maruth Goyal*

1 Overview

If one recalls proofs about Turing Machines typically seen in an undergraduate course on complexity, say, the time hierarchy theorem [HS65, Sip06], they may observe that the proofs always treated the Turing Machines as black boxes. We simply assume some input goes in, some amount of time passes, and perhaps we receive an output which we then play around with, with potentially other black-box TMs. Other proofs that may come to mind are Savitch's theorem [Sav70, Sip06], or $\mathsf{NTIME}(n) \nsubseteq \mathsf{TISP}(n^{1.2}, n^{0.2})$. One might wonder, can this technique be used to prove $\mathsf{P} \overset{?}{=} \mathsf{NP}$. As it turns out, the answer is **no**. In the 1970s, researchers started analyzing proofs themselves, and Baker, Gill and Solovay [BGS75] introduced the "Relativization Barrier". They introduced this notion of "relativization", and proved that if any given proof relativizes, it cannot prove $\mathsf{P} \overset{?}{=} \mathsf{NP}$.

2 Oracles

Suppose one day you are taking a walk on campus, and happen across an Oracle with answers to all the questions in the universe. While you may have been intelligent earlier, you can certainly answer more questions if this Oracle is your friend, right? We consider a similar notion in complexity theory, although instead of considering all-knowing oracles we restrict ourselves to oracles who can answer any question in some complexity class $\mathcal{D}$. Then, if you are a Turing Machine who can generally solve any problem in a class $\mathcal{C}$, we represent this friendship as $\mathcal{C}^{\mathcal{D}}$. That is, $\mathcal{C}$ with *oracle access* to $\mathcal{D}$. This is the set of languages decidable in $\mathcal{C}$, given an oracle for $\mathcal{D}$ that can answer any query in one step.

For instance, P^{NP} is the set of languages decidable in polynomial time, given access to say a $\mathsf{3-SAT}$ oracle.

3 Relativization

Notice that if in our proof, we treat Turing Machines as absolute black boxes, then we can give them access to any oracle of our choice, and the result would still hold. This is the notion of relativization. If you prove $C \subseteq D$ treating TMs as black-boxes, then your result "relativizes" as then $C^O \subseteq \mathcal{D}^O$ must hold for all oracles O as well. [BGS75] showed that no relativizing proof can prove $\mathsf{P} \overset{?}{=} \mathsf{NP}$ by proving that there exist oracles $\mathcal{A}, \mathcal{B}$ such that $\mathsf{P}^{\mathcal{A}} = \mathsf{NP}^{\mathcal{A}}$, and $\mathsf{P}^{B} \neq \mathsf{NP}^{B}$.

Thus, if you prove $\mathsf{P} = \mathsf{NP}$ using a relativizing proof, you may not get your \$1,000,000 since your proof would imply $\mathsf{P}^O = \mathsf{NP}^O$ for all O, but BGS showed there is an oracle with respect to which they are unequal. With that said, let's prove BGS

Theorem 1. [BGS75] *There exist oracles $\mathcal{A}, \mathcal{B}$ such that*

1. $\mathsf{P}^{\mathcal{A}} = \mathsf{NP}^{\mathcal{A}}$
2. $\mathsf{NP}^{\mathcal{B}} \not\subseteq \mathsf{P}^{\mathcal{B}}$

Proof. Intuitively, if $\mathcal{A}$ is powerful enough to engulf both P and NP then both are equally powerful with respect to that oracle. For instance, $\mathcal{A} = \mathsf{EXP}$ would do the trick. $\mathcal{B}$ is a bit more involved to construct.

We define $L_{\mathcal{B}} = \{1^n \mid \exists x. \in B, |x| = n\}$. That is, the unary strings such that there is a string of that length in $\mathcal{B}$. Notice $L_{\mathcal{B}}$ can be decided in $\mathsf{NP}^{\mathcal{B}}$. The machine would just guess over all bitstrings of length n, and perform an oracle query. What we want to do is create $\mathcal{B}$ to play specifically to NP's strengths, and P's weaknesses.

Observe that there are 2^n bitstrings of length n. Any machine in $\mathsf{P}^{\mathcal{B}}$ by definition can query at most $n^{O(1)}$ of these. This means for any polynomial time oracle machine, there are strings which it simply cannot query. Hence, for each of these machines we will pick a string it cannot query and do the opposite of what the machine does.

More formally, consider all Turing Machines with oracle access to $\mathcal{B}$ that run in time $2^n/10$ (the 10 here is essentially arbitrary, we just want less than 2^n to make sure there are strings that are not queried). Let these be $M_1, M_2, \ldots$. Then, on the i^{th} step pick a string s much larger than anything currently in $\mathcal{B}$, which is not queried by M_i. Then:

1. If M_i accepts $1^{|s|}$, throw s away.
2. If M_i rejects $1^{|s|}$, put s in $\mathcal{B}$.

Notice with such a $\mathcal{B}$, M_i may query strings of the same length as s, but never s itself, and since we always chose the length of s to be sufficiently large, there isn't some other string of the same length that M_i can query that will be in the language. Similarly for the case when the string is not in the language.

Thus, we have constructed an oracle $\mathcal{B}$ such that $L_{\mathcal{B}} \in \mathsf{NP}^{\mathcal{B}}$, but $L_{\mathcal{B}} \notin \mathsf{P}^{\mathcal{B}}$, since for each machine in $\mathsf{P}^{\mathcal{B}}$ there is some string for which it is guaranteed to give the wrong answer. Hence, $\mathsf{NP}^{\mathcal{B}} \not\subseteq \mathsf{P}^{\mathcal{B}}$. □

References

[BGS75] Theodore Baker, John Gill, and Robert Solovay. Relativizations of the p=?np question. *SIAM Journal on computing*, 4(4):431–442, 1975.

[HS65] Juris Hartmanis and Richard E Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 117:285–306, 1965.

[Sav70] Walter J Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of computer and system sciences*, 4(2):177–192, 1970.

[Sip06] Michael Sipser. *Introduction to the Theory of Computation.* Course Technology, second edition, 2006.