Lecture on Relativization

Jonathan Katz

1 Relativizing the $\mathcal{P}$ vs. $\mathcal{NP}$ Question

The main result of this lecture is to show the existence of oracles[1] A, B such that $\mathcal{P}^A = \mathcal{NP}^A$ while $\mathcal{P}^B \neq \mathcal{NP}^B$. A fancy way of expressing this is to say that *the $\mathcal{P}$ vs. $\mathcal{NP}$ question has contradictory relativizations*. This shows that the $\mathcal{P}$ vs. $\mathcal{NP}$ question cannot be solved by any proof techniques that "relativize" (since a "relativizing" proof of $\mathcal{P} = \mathcal{NP}$, say, would *by definition* hold relative to any oracle). As such, when this result was first demonstrated [2] it was taken as an indication of the difficulty of resolving the $\mathcal{P}$ vs. $\mathcal{NP}$ question using "standard techniques". It is important to note, however, that various non-relativizing proof techniques are known; as one example, the proof that $\mathsf{PSPACE} \subseteq \mathsf{IP}$ does not relativize (it is known that there exists an oracle A such that $\mathsf{PSPACE}^A \neq \mathsf{IP}^A$). See [4, Lect. 26] and [1, 3, 5] for further discussion.

An oracle A for which $\mathcal{P}^A = \mathcal{NP}^A$. Let A be a PSPACE-complete language. It is obvious that $\mathcal{P}^A \subseteq \mathcal{NP}^A$ for any A, so it remains to show that $\mathcal{NP}^A \subseteq \mathcal{P}^A$. We do this by showing that

$$\mathcal{NP}^A \subseteq \mathsf{PSPACE} \subseteq \mathcal{P}^A.$$

The second inclusion is immediate (just use a Cook reduction from any language $L \in \mathsf{PSPACE}$ to the PSPACE-complete problem A), and so we have only to prove the first inclusion. This, too, is easy: Let $L \in \mathcal{NP}^A$ and let M be a poly-time non-deterministic machine such that $L(M^A) = L$. Then using a deterministic PSPACE machine M' we simply try all possible non-deterministic choices for M, and whenever M makes a query to A we have M' answer the query by itself.

An oracle B for which $\mathcal{P}^B \neq \mathcal{NP}^B$. This is a bit more interesting. We want to find an oracle B such that $\mathcal{NP}^B \setminus \mathcal{P}^B$ is not empty. For any oracle B, define the language L_B as follows:

$$L_B \stackrel{\text{def}}{=} \{1^n \mid B \cap \{0,1\}^n \neq \emptyset\}.$$

It is immediate that $L_B \in \mathcal{NP}^B$ for any B. (On input 1^n, guess $x \in \{0,1\}^n$ and submit it to the oracle; output 1 iff the oracle returns 1.) As a "warm-up" to the desired result, we show:

Claim 1 *For any deterministic, polynomial-time oracle machine M, there exists a language B such that $L_B \neq L(M^B)$.*

Proof Given M with polynomial running time $p(\cdot)$, we construct B as follows: let n be the smallest integer such that $2^n > p(n)$. Note that on input 1^n, machine M cannot query its oracle on all strings of length n. We exploit this by defining B in the following way:

[1] We associate oracles with languages; i.e., if A is a language then we also let A denote the oracle that computes the characteristic function of A.

Run $M(1^n)$ and answer "0" to all queries of M. Let b be the output of M, and let $Q = \{q_1, \ldots\}$ denote all the queries of length exactly n that were made by M. Take arbitrary $x \in \{0,1\}^n \setminus Q$ (we know such an x exists, as discussed above). If $b = 0$, then put x in B; if $b = 1$, then take B to just be the empty set.

Now $M^B(1^n) = b$ (since B returns 0 for every query made by $M(1^n)$), but this answer is incorrect by construction of B. ■

This claim is not enough to prove the desired result, since we need to reverse the order of quantifiers and show that there exists a language B such that for *all* deterministic, poly-time M we have $L_B \neq L(M^B)$. We do this by extending the above argument. Consider an enumeration $M_1, \ldots$ of all deterministic, poly-time machines with running times $p_1, \ldots$. We will build B inductively. Let $B_0 = \emptyset$ and $n_0 = 1$. Then in the i^{th} iteration do the following:

- Let n_i be the smallest integer such that $2^{n_i} > p_i(n_i)$ and also $n_i > p_j(n_j)$ for all $1 \leq j < i$.
- Run $M_i(1^{n_i})$ and respond to its queries according to B_{i-1}. Let $Q = \{q_1, \ldots\}$ be the queries of length exactly n_i that were made by M_i, and let $x \in \{0,1\}^{n_i} \setminus Q$ (again, we know such an x exists). If $b = 0$ then set $B_i = B_{i-1} \cup \{x\}$; if $b = 1$ then set $B_i = B_{i-1}$ (and so B_i does not contain any strings of length n_i).

Let $B = \cup_i B_i$. We claim that B has the desired properties. Indeed, when we run $M_i(1^{n_i})$ with oracle access to B_i, we can see (following the reasoning in the previous proof) that M_i will output the wrong answer (and thus $M_i^{B_i}$ does not decide L_{B_i}). But the output of $M_i(1^{n_i})$ with oracle access to B is the same as the output of $M_i(1^{n_i})$ with oracle access to B_i, since all strings in $B \setminus B_i$ have length greater than $p_i(n_i)$ and so none of M_i's queries (on input 1^{n_i}) will be affected by using B instead of B_i. It follows that M_i^B does not decide L_B.

Bibliographic Notes

This is adapted from [4, Lecture 26]. The result presented here is due to [2].

References

[1] E. Allender. Oracles versus Proof Techniques that Do Not Relativize. *SIGAL Intl. Symposium on Algorithms*, pp. 39–52, 1990.

[2] T. Baker, J. Gill, and R. Solovay. Relativizations of the $\mathcal{P} \stackrel{?}{=} \mathcal{NP}$ Question. *SIAM J. Computing* 4(4): 431–442, 1975.

[3] L. Fortnow. The Role of Relativization in Complexity Theory. *Bulletin of the European Association for Theoretical Computer Science*, 52: 229–243, 1994.

[4] O. Goldreich. Introduction to Complexity Theory (July 31, 1999).

[5] J. Hartmanis, R. Chang, S. Chari, D. Ranjan, and P. Rohatgi. Relativization: A Revisionist Retrospective. *Current Trends in Theoretical Computer Science*, 1993. Available from http://www.cs.umbc.edu/ chang/papers/revisionist/.