CS 517: Baker-Gill-Solovay Theorem

Writeup by Mike Rosulek

Theorem 1 *There is a language B such that $P^B \neq NP^B$.*

The main idea is to consider the language $L_B = \{x \mid \exists y : |x| = |y| \text{ and } y \in B\}$. The definition of L_B depends on the choice of B, but $L_B \in \mathsf{NP}^B$ for any B. We want to construct a B so that $L_B \notin \mathsf{P}^B$.

Imagine an oracle B which answers 0 to everything (i.e., $B = \emptyset$). Hence $L_B = \emptyset$ as well. On an input of length n, the TM can only ask $p(n)$ queries of its oracle, for some polynomial p. For large enough n, $p(n) < 2^n$, so it is possible to find strings x and y of the same length where $M^B(x)$ never queries its oracle on y. If we change B to now include y, this might change the behavior of M on some inputs, but it cannot change M's behavior on input x since $M(x)$ never queries its oracle on y. On the other hand, adding y to B changes the "correct" answer of $x \in L_B$ (with respect to the modified B). Using this idea, we can always guarantee that $L(M^B)$ disagrees with L_B on input x: if $M^B(x)$ doesn't already give the wrong answer, we can redefine L_B by adding y to B.

So it is easy to construct a B that causes a *particular* oracle TM M to disagree with L_B. The idea of the proof is to do this trick *simultaneously to all* poly-time oracle TMs. To "sabotage" a particular M, we may need to add a string y to B. The main challenge is that adding y might change the behavior of some other M' that we've already considered. The construction of B is careful to avoid this kind of interference.

Proof Let $(M_1, p_1), (M_2, p_2), \ldots$ be an enumeration of all polynomial-time oracle TMs, where M_i halts on input x in at most $p_i(|x|)$ steps. We can safely assume without loss of generality that $p_i(n) > n$ for all i and n.

We define the oracle B using an enumerator. That is, B is the set of strings printed by the following process:[1]

<u>enumerator for B</u>:
initialize $S_1 := \emptyset$.
for $i = 1, 2, \ldots$:
  let n_i be the smallest integer such that:
    ▹ for all $j < i$, we have $n_i > p_j(n_j)$
    ▹ $2^{n_i} > p_i(n_i)$
  let $x_i := 0^{n_i}$ // *i.e., a string of n_i zeroes*
  let y_i be the lexicographically least string of length n_i
    . . . that is *not* made as an oracle query during the computation $M_i^{S_i}(x_i)$
  if $M_i^{S_i}(x_i) = 0$ then:
    print y_i
    set $S_{i+1} := S_i \cup \{y_i\}$
  else: set $S_{i+1} := S_i$

The enumerator's behavior is well-defined. In particular:

[1] It is not hard to see that the enumerator prints B in lexicographic order, implying that B is decidable.

- **Lines 3–5 are well-defined:** The previous iterations of the loop have already fixed specific values $p_j(n_j)$ for $j < i$. Furthermore, since p_i is a polynomial, it is true that $2^n > p_i(n)$ for sufficiently large n.
- **Line 7 is well-defined:** Since M_i runs for at most $p_i(|x_i|)$ steps on input x_i, it can make at most $p_i(|x_i|) = p_i(n_i)$ oracle queries. Since n_i is chosen so that $p_i(n_i) < 2^{n_i}$, there will always be some string of length n_i not queried by M_i.

Now define $L_B = \{x \mid \exists y : |x| = |y| \text{ and } y \in B\}$. Then $L_B \in \mathsf{NP}^B$, since we have defined L_B in terms of an existential quantifier followed by a condition that can be checked in polynomial time given a B-oracle.

Hence it suffices to show that $L_B \notin \mathsf{P}^B$. That is, no polynomial-time oracle TM decides L_B. More specifically, we will show that:

Claim *For all i, $x_i \in L_B \iff x_i \notin L(M_i^B)$. Hence for all i, $L(M_i^B) \neq L_B$. Since the M_i's range over all polynomial-time oracle TMs, it follows that $L_B \notin P^B$*

To prove the claim, take an arbitrary M_i. Consider the ith iteration of the main loop of the enumerator that defines B. In this loop, the enumerator explicitly simulates the computation $M_i^{S_i}(x_i)$. Note that S_i is equal to the set of strings printed so far by the enumerator. So $S_i \subseteq B$, but in general $S_i \neq B$. However,

- **The behavior of $M_i^{S_i}(x_i)$ is equal to that of $M_i^{S_{i+1}}(x_i)$:** From lines 11-12, we see that S_i and S_{i+1} are either the same (in which case the statement is trivially true), or else differ only in whether they include a specific y_i. But since M_i never queries its oracle on y_i, the difference in oracles cannot affect the computation of M_i.
- **The behavior of $M_i^{S_i}(x_i)$ is equal to that of $M_i^B(x_i)$.** Note B may include strings not in S_i (i.e., strings that the enumerator prints at a later time). However, line 4 of the enumerator ensures that any future string that is printed must be longer than $p_i(n_i)$ bits. Since M_i on input x_i runs for at most $p_i(n_i)$ steps, it cannot query its oracle on a string longer than $p_i(n_i)$ bits. So M_i (on input x_i) can never query its oracle on any string in $B \setminus S_i$.

Additionally, **if y_i is not printed, then B will contain no strings of length n_i.** This is because future values of n_i are ensured to be larger than this one, by line 3.

With this in mind, we see that:

$$
\begin{aligned}
x_i \in L_B &\iff \exists y \in B : |y| = |x| && \text{(definition of } L_B\text{)} \\
&\iff y_i \in B && \text{(previous observation)} \\
&\iff x_i \notin L(M_i^{S_i}) && \text{(by construction, lines 9–10)} \\
&\iff x_i \notin L(M_i^B)
\end{aligned}
$$

Summarizing, $L(M_i^B)$ always disagrees with L_B on input x_i. Hence $L(M_i^B) \neq L_B$. Since this is true for all poly-time oracle TMs M_i, we get $L_B \notin \mathsf{P}^B$. ■