$\mathsf{P} \neq \mathsf{NP}$ with probability 1

Introduction

Suppose we could show that $\mathsf{P} \neq \mathsf{NP}$, then by a similar proof we would expect that $\mathsf{P}^A \neq \mathsf{NP}^A$ for any oracle A. Similarly, if we could show that $\mathsf{P} = \mathsf{NP}$, then again by a similar proof we would expect that $\mathsf{P}^A = \mathsf{NP}^A$ for any oracle A. Baker, Gill and Solovay [1] showed that there are languages A and B such that

$$\mathsf{P}^A \neq \mathsf{NP}^A \quad \text{and} \quad \mathsf{P}^B = \mathsf{NP}^B.$$

This means that $\mathsf{P} \neq \mathsf{NP}$ and $\mathsf{P} = \mathsf{NP}$ are not relativizing results. On the contrary there are also results in complexity theory that can be relativized by giving additional oracle power. E.g. one consequence of the Deterministic Time Hierarchy Theorem, which itself is a relativizing result, is that $\mathsf{P} \neq \mathsf{EXP}$, and it is also true for all oracles A that

$$\mathsf{P}^A \neq \mathsf{EXP}^A.$$

Consider a random oracle A such that each string $x \in \{0,1\}^*$ is added in A say with probability $1/2$. The objective of this note is to prove a result by Bennett and Gill [2] which is

$$\Pr[\mathsf{P}^A \neq \mathsf{NP}^A] = 1 \quad \text{or} \quad \Pr[\mathsf{P}^A = \mathsf{NP}^A] = 0.$$

In other words, $\mathsf{P} \neq \mathsf{NP}$ relative to almost all oracles. Note that equality of two complexity classes is not implied by whether their relativized version is true for almost all oracles. E.g. $\mathsf{IP} = \mathsf{PSPACE}$, but $\mathsf{IP}^A \neq \mathsf{PSPACE}^A$ for almost all oracles A [3].

These notes are based on [4].

Defining $L(A)$

Let A be the random oracle as defined earlier. Corresponding to A, we'll define a language $L(A)$ such that $L(A) \in \mathsf{NP}^A$. Then, since $\mathsf{P}^A \subseteq \mathsf{NP}^A$, we have

$$\Pr[\mathsf{P}^A = \mathsf{NP}^A] = \Pr[\mathsf{NP}^A \subseteq P^A] \leq \Pr[L(A) \in \mathsf{P}^A]. \tag{1}$$

We define $L(A)$ as follows: Consider the lexicographic ordering of the strings in $\{0,1\}^*$. Let $x \in \{0,1\}^*$ such that $|x| = n$, and consider the 2^n blocks followed by x such that each

of them contain n strings. Now $x \in L(A)$ if and only if there exists at least one block (among the 2^n blocks) such that all its n strings are in A.

Claim 1: $L(A) \in \mathsf{NP}^A$.

Proof. Construct an NP machine N with oracle access to A. On input $x \in \{0,1\}^*$ of length n, the machine guesses the index $i \in \{1,2,\ldots,2^n\}$ of a block of size n that follows x. Then, it queries all the n strings of the i^{th} block to A. If all of them are in A, then the machine accepts x. Otherwise, if any of those strings are not in A, then the machine rejects x. The language accepted by the machine $L(N) = L(A)$. Since the number of queries is n, which is also the input size, $L(A) \in \mathsf{NP}^A$. □

The point of defining $L(A)$ in this way is that whether a string x is in $L(A)$ depends on 2^n conditions, and each of which have probability $1/2^n$ of being true.

Also, note that the certificate for $x \in L(A)$ is the set of n strings from the same block that are in A.

Claim 2: Let $x \in \{0,1\}^*$. Then, $\Pr[x \in L(A)] = 1 - \left(1 - \frac{1}{2^n}\right)^{2^n}$.

Proof. Let $|x| = n$. Then,

$$\begin{aligned}
\Pr[x \in L(A)] &= \Pr[\exists \text{ at least one block all of which is in } A] \\
&= 1 - \Pr[\text{none of the blocks are completely in } A] \\
&= 1 - \prod_{i=1}^{2^n} \Pr[\text{at least one string of the } i^{th} \text{ block is not in } A] \\
&= 1 - \left(1 - \Pr[\text{all the strings of a block are in } A]\right)^{2^n} \\
&= 1 - \left(1 - \frac{1}{2^n}\right)^{2^n}.
\end{aligned}$$

□

We'll see similar argument later.

Independence

Let $M_1, M_2, \ldots$ be a listing of all deterministic Turing machines that run in polynomial time. Then,

$$\mathsf{P}^A = \{L(M_i^A) \mid i \geq 1\}.$$

Now

$$\begin{aligned}
\Pr[L(A) \in \mathsf{P}^A] &= \Pr[\exists i \text{ such that } L(A) = L(M_i^A)] \\
&\leq \sum_i \Pr[L(A) = L(M_i^A)] \\
&= \sum_i \Pr[\forall x \; x \in L(A) \square L(M_i^A)],
\end{aligned}$$

where the inequality is due to the union bound, and

$$L(A)\square L(M_i^A) = \{x \mid x \in L(A) \Leftrightarrow x \in L(M_i^A)\}.$$

Our goal is to show that a fixed polynomial-time deterministic Turing machine M_i correctly accepts $L(A)$ with a very low probability, that is, it makes a lot of mistakes.

For a fixed a machine M_i^A, we want to construct a sequence $x_1 < x_2 < x_3 < \ldots$ of widely separated strings in lexicographic order such that the machine cannot correctly accept $L(A)$ in polynomial time. Now we can do the following approximation.

$$\begin{aligned}\Pr[L(A) \in \mathsf{P}^A] &\leq \sum_i \Pr[\forall x\, x \in L(A)\square L(M_i^A)] \\ &\leq \sum_i \Pr[\forall j\, x_j \in L(A)\square L(M_i^A)].\end{aligned}$$

Let $|x_i| = n_i$ for each i. We'd also like this sequence to satisfy the following two properties:

1. The events "$x_j \in L(A)$" for $j = 1, 2, \ldots$ are completely independent.
2. The event "$x_j \in L(M_i^A)$" is independent of the set of events {"$x_k \in L(M_i^A)$" $|k > j$} for each $j = 1, 2, \ldots$, in other words, we at least have sufficient independence for the events "$x_j \in L(M_i^A)$".

Ensuring Property 1

It is sufficient to choose x_j and x_{j+1} such that the regions following them that determine their membership in $L(A)$ are disjoint, that is, we'd like to have at least $n_j 2^{n_j}$ strings in between them. The possible lengths of strings between x_j and x_{j+1} are

$$n_j + 1, n_j + 2, \ldots, n_{j+1} - 1,$$

and there can be at least

$$\sum_{t=1}^{n_{j+1}-n_j-1} 2^{n_j+t}$$

many strings of such lengths. We would like this number to be at least $n_j 2^{n_j}$. Suppose **we choose** $n_{j+1} > 3n_j^2$. Then,

$$\begin{aligned}\sum_{t=1}^{n_{j+1}-n_j-1} 2^{n_j+t} &= (2^{n_{j+1}-n_j} - 2)2^{n_j} \\ &> (2^{3n_j^2-n_j} - 2)2^{n_j} \\ &> n_j 2^{n_j},\end{aligned}$$

since $2^{3n_j^2-n_j} - 2 > n_j$.

Ensuring Propery 2

What we want to show in this case is that the machine M_i on input x_j for large j cannot query strings of length n_{j+1} to the oracle A. For this **choose** $n_{j+1} > 2^{n_j+1}$ such that $n_1 > 2$. Then, in this case as well, we have

$$\begin{aligned}\sum_{t=1}^{n_{j+1}-n_j-1} 2^{n_j+t} &= (2^{n_{j+1}-n_j}-2)2^{n_j} \\ &> n_j 2^{n_j}.\end{aligned}$$

Since the running time of M_i^A is polynomially bounded, the space it can use is also polynomially bounded. Hence, on input x_j, the machine cannot query x_{j+1} or later strings because they have exponential length in the input size n_j.

Bounding the probability

Now consider

$$\begin{aligned}\Pr[\forall x\, x \in L(A)\square L(M_i^A)] &\leq \Pr[\forall j\, x_j \in L(A)\square L(M_i^A)] \\ &= \prod_j \Pr[x_j \in L(A)\square L(M_i^A) | x_k \in L(A)\square L(M_i^A) \text{ for } k < j].\end{aligned}$$

For a fixed machine M_i^A and a string x_j for some i and j, let us refer to the following condition as condition C:

$$x_k \in L(A)\square L(M_i^A) \text{ for } k < j,$$

and we'll always assume this condition. Assuming this condition means that the oracle A is "fixed" on the strings $x_1, x_2, \ldots, x_{j-1}$, and this has the effect of shrinking the sample space of the random oracles that we are considering.

Putting everything together, we have

$$\Pr[\mathsf{P}^A = \mathsf{NP}^A] \leq \sum_i \prod_j \Pr[x_j \in L(A)\square L(M_i^A) | C].$$

To show $\Pr[\mathsf{P}^A = \mathsf{NP}^A] = 0$ it is sufficient to show that $\Pr[x_j \in L(A)\square L(M_i^A) | C] < 1$. Let

- $p_1 = \Pr[x_j \in L(M_i^A) \text{ and } x_j \in L(A) \mid C]$,
- $p_2 = \Pr[x_j \notin L(M_i^A) \text{ and } x_j \in L(A) \mid C]$,
- $p_3 = \Pr[x_j \in L(M_i^A) \text{ and } x_j \notin L(A) \mid C]$,
- $p_4 = \Pr[x_j \notin L(M_i^A) \text{ and } x_j \notin L(A) \mid C]$.

Then,

$$\begin{aligned}\Pr[x_j \in L(A)\square L(M_i^A) | C] < 1 &\Leftrightarrow p_1 + p_4 < 1 \\ &\Leftrightarrow p_2 + p_3 > 0\end{aligned}$$

Hence, we only need to show the following that $p_2 + p_3 > 0$. To do this we need to prove two more claims.

Claim 3: $p_1 + p_2 > 0.6$, and $p_3 + p_4 > 0.3$.

Proof.

$$p_1 + p_2 = \Pr[x_j \in L(A) \mid x_k \in L(A) \square L(M_i^A) \text{ for } k < j]$$

Whether $x_j \in L(A)$ depends on the 2^n block following x_j. By Property 1, those blocks are disjoint from the blocks corresponding to x_k for any $k < j$. Also, on input x_k for $k < j$, the machine M_i cannot query the oracle strings of length n_j or more by Property 2. Thus we can drop condition C. By Claim 2, we have

$$p_1 + p_2 = \Pr[x_j \in L(A)] = 1 - \left(1 - \frac{1}{2^{n_j}}\right)^{2^{n_j}} \geq 1 - \frac{1}{e} > 0.6.$$

Similarly,

$$p_3 + p_4 = \Pr[x_j \notin L(A) | C] = \Pr[x_j \notin L(A)] = \left(1 - \frac{1}{2^{n_j}}\right)^{2^{n_j}} > 0.3.$$

Here, the bound holds, in particularly, due to $n_1 > 2$ as chosen earlier. □

Claim 4: $\frac{p_2}{p_2+p_4} \geq \frac{1}{3}$

Proof.

$$\begin{aligned}
\frac{p_2}{p_2 + p_4} &= \frac{\Pr[x_j \notin L(M_i^A) \text{ and } x_j \in L(A) \mid C]}{\Pr[x_j \notin L(M_i^A) \mid C]} \\
&= \Pr[x_j \in L(A) | x_j \notin L(M_i^A), C] \\
&= \Pr[x_j \in L(A) | x_j \notin L(M_i^A), x_k \in L(A) \square L(M_i^A) \text{ for } k < j]
\end{aligned}$$

In this case as well, the previous argument holds. But we cannot drop the condition easily. The only difference is that we are given the condition $x_j \notin L(M_i^A)$. The machine hence cannot query strings of length n_{j+1}, but it might have queried strings of length n_j. Since the running time of the machine M_i polynomially bounded. Hence, on input x_j, it can only query polynomially (in n_j) many strings to the oracle of length n_j that could affect the blocks following x_j on which the event "$x_j \in L(A)$" depends.

Let $m = n_j$, and let $p_i(m)$ be the bound on the number of queries made by the machine M_i to the oracle. These strings can lie in at most $p_i(m)$ many different blocks following x_j.

Let us now bound the probability as done in the proof of Claim 2,

$$\begin{aligned}
1 - \Pr[x_j \in L(A) \mid x_j \notin L(M_i^A) \mid C] &= 1 - \Pr[\exists \text{ at least one block all of which is in } A \mid x_j \notin L(M_i^A), C] \\
&= \Pr[\text{none of the blocks are completely in } A \mid x_j \notin L(M_i^A), C] \\
&= \prod_{i=1}^{2^m} \Pr[i^{th} \text{ block is not completely in } A \mid x_j \notin L(M_i^A), C]
\end{aligned}$$

For the sake of approximation, let us fix the blocks in which the strings the machine queried lie. There are at most $p_i(m)$ many of those, and so we have rest of the $2^m - p_i(m)$ blocks free for our random experiment. Then,

$$\begin{aligned}
1 - \Pr[x_j \in L(A) \mid x_j \notin L(M_i^A) \mid C] &\leq \prod_{i=1}^{2^m - p_i(m)} \Pr[i^{th} \text{ block is not completely in } A] \\
&= \left(1 - \Pr[\text{all strings of a block are in } A]\right)^{2^m - p_i(m)} \\
&= \left(1 - \frac{1}{2^m}\right)^{2^m - p_i(m)} \\
&\leq \left(1 - \frac{1}{2^m}\right)^{2^m/2} \\
&\leq 2/3.
\end{aligned}$$

The second last bound is obtained by considering large m for which $p_i(m) \leq 2^m/2$. Hence,

$$\frac{p_2}{p_2 + p_4} = \Pr[x_j \in L(A) \mid x_j \notin L(M_i^A)] \geq 1/3. \tag{2}$$

□

Now we can prove $p_2 + p_3 > 0$ be considering two cases:

Case 1: $p_3 \geq 0.1$.

Then, clearly $p_2 + p_3 \geq 0.1$.

Case 2: $p_3 < 0.1$.

Since $p_3 + p_4 > 0.3$ by Claim 3, we have $p_4 > 0.2$. Also since $\frac{p_2}{p_2+p_4} \geq \frac{1}{3}$ by Claim 4, we get $p_2 > 0.1$, and therefore, $p_2 + p_3 > 0.1$.

In any case, we get $p_2 + p_3 > 0$, and this completes a proof of $\Pr[\mathsf{P}^A = \mathsf{NP}^A] = 0$.

References

[1] Theodore Baker, John Gill, and Robert Solovay. Relativizations of the $\mathscr{P} =? \mathscr{NP}$ question. *SIAM Journal on Computing*, 4(4):431–442, 1975.

[2] Charles H. Bennett and John Gill. Relative to a random oracle a, $P^A \neq NP^A \neq \text{co-}NP^A$ with probability 1. *SIAM Journal on Computing*, 10(1):96–113, 1981.

[3] Richard Chang, Benny Chor, Oded Goldreich, Juris Hartmanis, Johan Håstad, Desh Ranjan, and Pankaj Rohatgi. The random oracle hypothesis is false. *Journal of Computer and System Sciences*, 49(1):24–39, August 1994.

[4] Uwe Schöning and Randall Pruim. $\mathsf{P} \neq \mathsf{NP}$ *with probability 1, Gems of Theoretical Computer Science*, pages 191–195. Springer Berlin Heidelberg, 1998.