

P is different from NP

Arthur MILCHIOR

February 23 2010

Abstract

In this article, I intend to prove that $P \neq NP$ by showing an EXP-complete language B such that $P^B \neq NP^B$, and that $P=NP$ imply that $P^E=NP^E$ when E is an EXP-complete language, thus ensuring a contradiction.

1 EXP-complete oracle

This section is almost a copy of the proof of theorem 3.7 in the Arora-Barak book, and the answer of question 3.3.

Let E be an Exp-Time-complete language.

For any language B , let U_b be the unary language $U_b = \{1^n \mid \text{some string of length } n \text{ is in } B \text{ and begin by } 1\}$

For every oracle B , the language U_b is clearly in NP^B , since a nondeterministic TM can make a nondeterministic guess for the string $x \in \{0, 1\}^n$ such that $x \in B$.

We now construct an oracle B such that $U_B \notin P^B$, implying that $P^B \neq NP^B$.

1.1 Construction of B:

For every i , we let M_i be the oracle TM represented by the binary expansion of i . We construct B in stages, where stage i ensure that M_i^B does not decide U_B in $2^n/10$ step. Initially we let B be $\{0w \mid w \in E\}$ and gradually add strings beginning by 1 to it. Each stage determines the status of a finite number of strings (i.e., whether or not these strings will ultimately be in B)

Stage i : So far, we have declared whether or not a finite number of strings beginning by 1 are in B . Choose n large enough so that it exceeds the length of any such string, and run M_i on input 1^n for $2^n/10$ steps. Whenever M_i queries the oracle about strings whose status has been determined we answer consistently. When M_i queries strings whose status is undetermined, we declare that the string is not in B . After letting M_i finish computing on 1^n , we now wish to ensure that the answer of M_i on 1^n (whatever it was) is incorrect.

If M_i accepts 1^n , we declare that all strings of length n not in B , there is no contradiction since we only accepted strings smaller than n , thus ensuring $1^n \notin U_b$. Conversely, if M_i rejects 1^n , we pick the smallest¹ string x of length n beginning by 1 that was not yet rejected

(such a string exists because we have only decided the fate of at most $\sum_{i=0}^{i \leq n} 2^i/10 < 2^{n+1}/10 <$

$2^{n-1}/2$ strings in $1\{0, 1\}^{n-1}$ and that there is 2^{n-1} such strings), declare that x is in B , thus ensuring $1^n \in U_B$, and reject all other strings of the same length. In either case, the answer

¹smallest mean by the usual order on binary number.

of M_i is incorrect. Since every polynomial $p(n)$ is smaller than $2^n/10$ for large enough n , and every TM M is represented by infinitely many strings, our construction will ensure that M does not decide U_b .

Thus we have shown U_B is not in P^B , and thus $P^B \neq NP^B$.

1.2 B is EXP-complete

Let's prove that B is EXP-complete, first it's easy to see that B is EXP-Hard, a word w is in E iff $0w$ is in B , and it is a log-space reduction, so let's prove $B \in \text{EXP}$.

A word either begins by 0 or 1, to decide if a word w is in B , if it begins by 0 and continue with x , we calculate if $x \in E$, which take an exponential time since $E \in \text{EXP}$, else if it begins by 1 and continue with x we must iterate the stage of construction until $n \geq |x|$, since at each stage n augment, then at most n steps are executed.

We will create two lists, one of the word beginning by 1 accepted by B , and one of the word beginning by 1 refused, and when a query is made, if it begins by 0 we will execute E on the rest of the query, else we will check in both list to see if the word is known, else we will add it to the refused list.

During each stage we simulate at most $2^n/10$ step of a TM with oracle B , and we decide at most $2^n/10$ words by querying, and the $2^n/10$ words' size are at most $2^n/10$, so the list size is bigger of $2^{2n}/100$ letters. So at stage i , the size of the list is at most $i * 2^{2n}/100 < n * 2^{2n}/100$ which is of exponential size, which imply that the answer of a request only take exponential time. There is 2^n words of size n , if we need to find the smaller word not yet in a list (when we decide to accept a word), we can try them all, until we find one not in the list, this also take exponential time $O(n2^{2n} * 2^n) = O(n2^{3n})$.

So simulating a stage take $O(n2^n/10 * 2^{2n} + n2^{3n}) = O(n2^{3n})$, and since there is at most n stage, the entire execution is in time $O(n^2 2^{3n})$ which is clearly exponential. So $B \in \text{EXP}$.

So B is EXP-complete.

2 $P=NP \Rightarrow P^B=NP^B$

Let suppose $P=NP$, let prove that $P^B=NP^B$.

We will only prove $NP^B \subseteq P^B$. Let L be a language in NP^B and M a NDTM with oracle B in time n^c who accept L . There exist an equivalent NDTM M' without oracle, that calculate L by calculating B each time a query is made. At most n^c query, which take time exponential, are made, so $M' \in \text{NEXP}$.

Since $P=NP$ we have $\text{NEXP}=\text{EXP}$, so $L \in \text{EXP}$, and then since B is EXP-complete, we can reduce L to B . Since it's poly-time reduction, let M'' be a DTM with oracle B that reduce L to B , M'' is clearly in P^B and thus $L \in P^B$.

So we have $NP^B \subseteq P^B$ and thus $NP^B=P^B$.

Since we know that $NP^B \neq P^B$, there is a contradiction, hence $P \neq NP$.