Introduction to Complexity Theory*
Lecture 26: Relativization

Notes taken by Leia Passoni

July 28, 1999

Summary: In this lecture we deal with relativization of complexity classes. In particular, we discuss the role of relativization with respect to the $\mathcal{P} \stackrel{?}{=} NP$ question; that is, we shall see that for some oracle A, $\mathcal{P}^A = \mathcal{NP}^A$ whereas for another A $\mathcal{P}^A \neq \mathcal{NP}^A$. However, it also holds that $\mathcal{IP}^A \neq \mathcal{PSPACE}^A$ for a random A, whereas $\mathcal{IP} = \mathcal{PSPACE}$

Oded's Note: *The study of relativization is motivated by the belief that relativized results indicate limitations of proof techniques which may be applied in the real (unrelativized) world. In the conclusion section we explain why we do not share this belief. In a nutshell, whereas it is useful to refer to proof techniques when discussing and unifying known results, it is misleading to refer to a "proof technique" as if it were a domain with well-defined boundaries (and speculate on which results are beyond such boundaries).*

In contrast, we see benefit in an attempt to define frameworks for proving certain results, and discuss properties of proofs within such well-defined frameworks (e.g., proofs of certain well-defined properties cannot prove a particular result or results of certain well-defined form). In fact, our original intention was to present such a framework, called Natural Proofs, in this lecture. This intention was abandoned since we did not see circuit size lower bounds in the course, and such proofs are the context of Natural Proofs.

1 Relativization of Complexity Classes

We have already mentioned the use of polynomial time machines that have access to some oracle in order to define the Polynomial Hierarchy (PH).

Given any two compexity classes C_1 and C_2, were C_2 is the class of oracles, it is not always possible to have a natural notion of what is the class $C_1^{C_2}$ – it is not necessarily the case that for every complexity class C_1 we can define its *relativization* to C_2.

There are some conditions under which such a relativization can be done. One of those conditions seems to be that the complexity class C_1 has to be defined in terms of some type of machines – that is, for any language L in C_1 there exists a machine M_L such that $L = L(M_L)$. Furthermore, the definition of M_L has to be extendable to the definition of *oracle machine*.

* Lecture Notes for a course given by Oded Goldreich at the Weizmann Institute of Science, Israel.

Oracle Machines. We consider three types of oracle machines:

Definition 1 *A (deterministic/nondeterministic/probabilistic) polynomial time* oracle machine *is a machine with a distinguished work tape – the query tape, where the machine may make oracle queries. These queries are answered by a function called the* oracle.

Some notations: $M^f(x)$ denotes a computation of M on input x when given access to oracle f. When the machine writes on the oracle tape a query q (and invokes the oracle), then the tape's contents q is replaced by $f(q)$.

We also write $M^A(x)$, where the oracle A is a language. In this case M is given access to the characteristic function of the language, χ_A.

2 The $\mathcal{P} \stackrel{?}{=} \mathcal{NP}$ question Relativized

Trying to solve the $\mathcal{P} \neq \mathcal{NP}$ conjecture, the corresponding relativized conjecture has been investigated, that is, determining whether $\mathcal{P}^A \neq \mathcal{NP}^A$ or $\mathcal{P}^A = \mathcal{NP}^A$ for some oracle A. Thus, we ask which of the following two possibilities holds:

- There exists an oracle A such that $\mathcal{P}^A = \mathcal{NP}^A$
- There exists an oracle A such that $\mathcal{P}^A \neq \mathcal{NP}^A$

Recall that

$$\mathcal{P}^A \stackrel{\text{def}}{=} \{L : \exists \ \text{deterministic} \ \text{poly}-\text{time} \ \ M : L(M^A) = L\}$$

$\mathcal{P}^A$ is then a class of languages: from a countable set of machines M we get a countable set of oracle machines M^A that run in polynomial time, each having oracle access to A.

Note that if $A = \emptyset$ or $A = \{0,1\}^*$, then $\mathcal{P}^A = \mathcal{P}$. In fact, if the answers given to queries to the oracle are all "yes" or all "no", then any language in $\mathcal{P}^A$ can be accepted by a deterministic polynomial time machine, just committing the oracle questins and aswers.
Also, $A \in \mathcal{P} \Rightarrow \mathcal{P}^A = \mathcal{P}$: in this case a P machine can simulate the access to the oracle in polynomial time.
Therefore, it is worthwhile to use oracles machines when the oracle A defines a language which is "complex".

In the same way $\mathcal{NP}^A$ can be defined:

$$\mathcal{NP}^A \stackrel{\text{def}}{=} \{L : \exists \ \text{nondeterministic} \ \text{poly}-\text{time} \ \ M : L(M^A) = L\}$$

It could seem that if $\exists A : \mathcal{P}^A \neq \mathcal{NP}^A$ then $\mathcal{P} \neq \mathcal{NP}$, but this is not the case. Of course it would be correct to assert that if $\forall A \ \ \mathcal{P}^A \neq \mathcal{NP}^A$ then $\mathcal{P} \neq \mathcal{NP}$, because in this case it would be possible to consider a trivial oracle like $A = \emptyset$ or $A = \{0,1\}^*$.

Going back to the $\mathcal{P} \stackrel{?}{=} \mathcal{NP}$ question, a possible reason for investigating its relativized version is the hope that a result obtained for a class of oracles would shed light on the unrelativized question. The two following theorems show that this hope fails.

Theorem 2 *An oracle A exists such that $\mathcal{P}^A = \mathcal{NP}^A$*

Proof: It is obvious that $\forall A \ \ \mathcal{P}^A \subseteq \mathcal{NP}^A$. It is left to show that $\exists A : \mathcal{NP}^A \subseteq \mathcal{P}^A$.

The idea of the proof is to choose as oracle A a $\mathcal{PSPACE}$-complete language, that is, a language so "powerful" that no more advantage is left for a nondeterministic machine over a deterministic one if they both have access to oracle A.

We first show that $\mathcal{NP}^{\mathcal{PSPACE}} \subseteq \mathcal{PSPACE}$. Let A be a $\mathcal{PSPACE}$ language and let $L \in \mathcal{NP}^A$. Then there exists a nondeterministic poly-time machine M such that $L = L(M^A)$. Equivalently, there exists a deterministic poly-time machine $\bar{M}$ such that $\forall x \in L, \exists y$ and polynomial $p_1 : |y| \leq p_1(|x|)$ and such that $\bar{M}^A(x,y)$ accepts. It is possible then to derive a deterministic machine M' such that $M'(x)$ accepts using at most $p_1(|x|)$ cells of space. Machine M' cycles throughout all possible y's of length $\leq p_1(|x|)$, simulating $\bar{M}^A(x,y)$ for each such y. In this simulation, when $\bar{M}$ queries the oracle on a word w, machine M' simulates the oracle by determining whether $w \in A$. Note that $|w| \leq p_2(|x|)$, where $p_2(|x|)$ is the polynomial bound on the number of steps in which $\bar{M}$ accepts the input x. Since $A \in \mathcal{PSPACE}$ it follows that determining $w \in A$ can be done in space $poly(|w|) = poly(|x|)$. So, M' can decide $L \in \mathcal{NP}^A$ using in all $p(|x|)$ space cells for some polynomial p. We conclude that

$$\text{for any } A \in \mathcal{PSPACE}, \quad \mathcal{NP}^A \subseteq \mathcal{PSPACE}. \tag{1}$$

We now show that for some $A \in \mathcal{PSPACE}$ (specifically, any $\mathcal{PSPACE}$-complete A will do), $\mathcal{PSPACE} \subseteq \mathcal{P}^A$. Consider a $\mathcal{PSPACE}$-complete language, A. For any $\bar{L} \in \mathcal{PSPACE}$, consider the Cook-reduction of $\bar{L}$ to A; that is, a polynomial-time oracle machine R that given access to oracle A decides $\bar{L}$. But this means that $\bar{L} = L(R^A) \in \mathcal{P}^A$ (which also equals $\mathcal{P}^{\mathcal{PSPACE}}$). Thus,

$$\text{for some } A \in \mathcal{PSPACE}, \quad \mathcal{PSPACE} \subseteq \mathcal{P}^A. \tag{2}$$

Combining Equations (1) and (2), we conclude that there exists an oracle A such that

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

and the theorem follows. ■

In spite of the result of Theorem 2, the oracle A can be chosen in such a way that the two relativized classes $\mathcal{P}^A$ and $\mathcal{NP}^A$ are different.

Theorem 3 *An oracle A exists such that $\mathcal{P}^A \neq \mathcal{NP}^A$*

Proof: Since $\mathcal{P}^A \subseteq \mathcal{NP}^A$, we want to find a language L such that $L \in \mathcal{NP}^A \setminus \mathcal{P}^A$, that is, a language that separates the two classes.
In this case the idea-technique for getting the separation is to define the language using the oracle A. Thus, for every oracle A we define

$$L_A \stackrel{\text{def}}{=} \{1^n : \exists w \in \{0,1\}^n : w \in A\}$$

What is peculiar about language L_A is that it describes whether $A \cap \{0,1\}^n$ is a nonempty set. It can be seen that $L_A \in \mathcal{NP}^A$, by showing a nondeterministic machine with access to A that accepts L_A:

<u>On input $x \in \{0,1\}^n$:</u>

1. if $x \neq 1^n$, then reject

2. otherwise guess $w \in \{0,1\}^n$

3. accept iff $w \in A$

If $x = 1^n \in L_A$, there exists a computation of the machine that accepts – the computation that succeeds in guessing a proper w which is checked to be in A by an oracle query to A. Otherwise, if $1^n \notin L_A$, there is no $w \in \{0,1\}^n$ so that $w \in A$, and therefore the machine can never accept; so in this case, no matter what guess it makes, the machine always rejects.

Now we have to show the existence of an oracle A so that no deterministic poly-time machine can accept L_A. We first show the following

Claim 2.1 *For every deterministic polynomial time M, there exists A so that $L_A \neq L(M^A)$*

Proof: We are given machine M and $p(\cdot)$, the polynomial upper bound on the running time of M and on the number of possible queries M can ask to the oracle.

We then choose n_1 such that $n_1 = min\{m : 2^m > p(m)\}$, and we consider the computation of M^A on input 1^{n_1}. The purpose is to define A in such a way that M will err in its decision whether $1^{n_1} \in A$ or not.

Definition of A: we consider three cases regarding the length $n = |q|$ of a query q: $n < n_1$, $n = n_1$, $n > n_1$.

- If $n < n_1$, we can for instance assume we have already answered "0" to all 2^n possible queries during computations with inputs of length less than n_1. In this case, on input 1^{n_1} we answer consistently "0".

- If $n = n_1$, then we use the fact $2^n > p(n)$, which implies that not all n-bit strings are queried. To all (up to $p(n)$) queries on input 1^{n_1} answer "0". What about the queries which M didn't ask? We will define A in such a way that

  - If $M^A(1^{n_1})$ accepts then let $A \cup \{0,1\}^{n_1} = \emptyset$. So we don't put in A any word of length n_1, i.e. we answer "0" to all remaining queries;
  - If $M^A(1^{n_1})$ rejects then let $A \cup \{0,1\}^{n_1} = \{\bar{w}\}$, where $\bar{w}$ is one of the words of length n_1 that have not been queried by M.

  We stress that $M^A(1^{n_1})$ does not depend on whether $\bar{w} \in A$ (since $\bar{w}$ is not queried in this compuation), but $1^n \in L_A$ does depend on whether $\bar{w} \in A$.

- If $n > n_1$. Like before, answer "0". But in this case we are committing the oracle answers on longer strings.

  Anyway, this is not a problem because any $n > n_1$ has to be $n \leq p(n_1)$, otherwise M wouldn't have the time to write the query. Thus, given n_1 there is a finite number of possible lengths of queries $n : n > n_1$, so we can carry on at this stage the construction for these longer n's. (This extra discussion is only important for the extension below.)

Oracle A is thus defined in such a way that, on input 1^{n_1} machine M^A cannot decide whether 1^{n_1} is in L_A or not. ■

Anyway, Claim 2.1 is not enough to prove Theorem 3, since the oracle A in the claim is possibly different for any M. We need to state a stronger claim. That is

$$\exists A : \forall \text{ deterministic polynomial-time } M,\ \ L_A \neq L(M^A)$$

This claim is provable by extending the same argument used in Claim 2.1. The idea is as follows: first we enumerate all deterministic poly-time machines

$$\begin{array}{ccccc} M_1, & M_2, & \ldots, & M_i, & \ldots \\ \downarrow & \downarrow & & \downarrow & \\ n_1, & n_2, & \ldots, & n_i, & \ldots \end{array}$$

and to each machine M_i we associate a "large enough" integer n_i. The oracle A is built in stages in such a way that, at any step we deal with a portion $A(i-1) \subseteq \{0,1\}^{n_{i-1}}$ of the oracle. Then $A = \cup_{i>0} A(i)$ and $M_i^A(1^{n_i}) = M_i^{A(i-1)}(1^{n_i})$. The oracle A is such that, at any step i, M_i^A errs in decision on input 1^{n_i}.

Define $A(i) = \{\text{words that have been placed in } A \text{ after step } i\}$. We let $A(0) = \emptyset$ and $n_0 = 0$. Consider machine M_1 and its polynomial bound $p_1(\cdot)$. Choose n_1 such that

$$n_1 = min\{m : 2^m > p_1(m)\}$$

At this point $A = A(0) = \emptyset$. To build $A(1)$ we follow the same procedure used in Claim 2.1; in particular

- if $M_1^A(1^{n_1})$ accepts then $A(1) = A(0)$;
- if $M_1^A(1^{n_1})$ rejects then $A(1) = A(0) \cup \{w_{n_1}\}$, where w_{n_1} is one of the words of length n_1 that have not been queried by M_1.

At stage i, consider machine M_i and $p_i(\cdot)$. Choose n_i such that

$$n_i = min\{m : 2^m > p_i(m) \text{ and } (\forall j < i),\ m > p_j(n_j)\}$$

Currently A equals $A(i-1)$; so when on input 1^{n_i} machine M_i asks queries of length less than n_i, the anwers have to be consistent with what has previously defined to be in the oracle. Like before, we then build $A(i)$ such that

- if $M_i^A(1^{n_i})$ accepts then $A(i) = A(i-1)$;
- if $M_i^A(1^{n_i})$ rejects then $A(i) = A(i-1) \cup \{w_{n_i}\}$, where w_{n_i} is one of the words of length n_i that have not been queried by M_i.

The word w_{n_i} possibly added to A at this stage is one of those not queried by M_i on input 1^{n_i}; moreover, words possibly added in successive stages have length greater than $p_j(n_j)$ for every $j < i$. This ensures that $M_j^{A(i)}(1^{n_j}) = M_j^{A(j-1)}(1^{n_j})$ for every $j < i$.

The oracle A is then built in such a way that no polynomial time machine M with oracle A exists such that $L(M^A) = L_A$ – every M_i^A fails deciding on input 1^{n_i}.

It is important to note that this construction works only on a specific portion of the oracle A at every stage, keeping all other portions of A untouched. ■

Considering the results of Theorem 2 and Theorem 3, we can understand that the investigation of the relativized $\mathcal{P} \neq \mathcal{NP}$ conjecture cannot possibly help in proving or disproving the unrelativized one. What instead should be taken from these results is the fact that the $\mathcal{P} \neq \mathcal{NP}$ conjecture should be investigated with *proof techniques that do not relativize*, that is, proof thechniques that cannot be extended to oracles – so the simulation and diagonalization thechniques used in the above theorems do not seem to be adequate for investigating the conjecture.

3 Relativization with a Random Oracle

Theorem 3 can be somehow generalized; in fact it is possible to prove that

$$Prob_A[\mathcal{P}^A \neq \mathcal{NP}^A] = 1 \tag{3}$$

The probabilily is taken over all oracles A; Eq. (3) means that for almost all oracles A, $\mathcal{P} \neq \mathcal{NP}$ holds. Since $\mathcal{P}^A \subseteq \mathcal{BPP}^A$ for all A, Eq. (3) is a consequence of the following

Theorem 4 *For a random oracle A, $\mathcal{NP}^A \nsubseteq \mathcal{BPP}^A$, that is, $Prob_A[\mathcal{NP}^A \nsubseteq \mathcal{BPP}^A] = 1$*

Proof: We have to find a language L such that $L \in \mathcal{NP}^A$ but $L \notin BPP^A$ with probability 1. Again, the idea is to define L_A depending on the oracle A:

$$L_A \stackrel{\text{def}}{=} \{1^n : \exists w \in \{0,1\}^n : \forall u \in C_n, wu \in A\}$$

Here C_n is defined to be any canonical set of n strings of length n. For instance, we can take C_n to be the set of all strings of Hamming weight 1: $C_n \stackrel{\text{def}}{=} \{10\ldots0, 010\ldots0, \ldots, 0\ldots01\}$.

Oded's Note: *The idea behind the use of C_n is to associate with each $w \in \{0,1\}^n$ a coin with odds 2^{-n} of being 1 (and being zero otherwise). If our oracle were to be chosen so that each string $w \in \{0,1\}^n$ is in it with probability 2^{-n} then the extra complication of using C_n would not have been needed. However, the oracle is chosen so that each string is equally likely to be or not be in it, and so we need to implement a coin with odds 2^{-n} of being 1 by using an unbiased coin. That's exactly what the above construction does. The aim of all this is to get to a situation that $1^n \notin L_A$ and "$1^n \in L_A$ via a single witness w" are about as likely, and are also quite likely.*

Again, it can be seen that $L_A \in \mathcal{NP}^A$ by showing a nondeterministic machine with access to A that generates L_A:

<u>On input $x \in \{0,1\}^n$:</u>

1. if $x \neq 1^n$, then reject
2. otherwise guess $w \in \{0,1\}^n$
3. accept iff $\forall u \in C_n, wu \in A$

Here the machine makes n queries to A and accepts iff all of them are anwered "yes" by the oracle.

Now for a randomly chosen A it can be shown that

$$Prob_A[1^n \notin L_A] = Prob_A[\nexists w : \forall u \in C_n, wu \in A] = 1/\mathrm{e} \quad = \text{ constant} \tag{4}$$

In fact, if w and u are fixed, then $Prob_A[wu \in A] = 1/2$ because A is chosen at random, thus every query has probability 1/2 to be in A or not.

If only w is fixed then $Prob_A[\forall u \in C_n, wu \in A] = 2^{-n}$ since all n events $[wu \in A]$ are independent.

So for this fixed w, $Prob_A[\neg\forall u \in C_n, wu \in A] = 1 - Prob_A[\forall u \in C_n, wu \notin A] = 1 - 2^{-n}$.

Again, there are 2^n possible w and all events are independent, thus $Prob_A[1^n \notin L_A] = Prob_A[\forall w, \neg\forall u \in C_n, wu \in A] = (1-2^{-n})^{2^n} = 1/\mathrm{e} = \text{constant}$.

In a similar way it can also be shown that

$$Prob_A[\exists! w \in \{0,1\}^n : \forall u \in C_n, wu \in A] \approx 1/\mathrm{e} \quad = \text{ constant} \tag{5}$$

Language L_A was then defined in such a way that

- with constant probability, $1^n \notin A$
- with constant probability, there exists a unique witness for $1^n \in A$ i.e. $\exists! w : \forall u \in C_n, wu \in A$

Intuitively, there is therefore a very small probability for any probabilistic polynomial time machine to come across the unique w so that $\forall u \in C_n, wu \in A$. In fact the following can be proven:

Claim 3.1 *Let M be a probabilistic polynomial time oracle machine. Then, for sufficiently large n*

$$Prob_A[M^A(1^n) = \chi_{L_A}(1^n)] < 0.8$$

Remarks: What we actually want to achieve is $Prob_A[L(M^A) = L_A] = 0$; by Claim 3.1 we can make the probability that M^A decides correctly on all inputs 1^n (i.e. for all n) vanish exponentially with the number of n's we consider.
Moreover, we want Claim 3.1 to hold for any possible M, that is $Prob_A[\mathcal{BPP}^A \ni L_A] = 0$. This extension can be carried out similarly to what was done in Theorem 3.

Proof: We choose n large enough so that the running time of $M^A(1^n)$ which equals $poly(n) \ll 2^n$. Now consider the case in which on input 1^n machine M makes queries of length $2n$ (other cases can be carried as it has been done in Theorem 3). We evaluate

$$\begin{aligned}
Prob_A[M^A(1^n) = \chi_{L_A}(1^n)] &= P_{\text{no}} + P_{\text{uni}} + P_{\text{rest}} \text{ , where} \\
P_{\text{no}} &\stackrel{\text{def}}{=} Prob_A[\chi_{L_A}(1^n) = 0] \cdot Prob_A[M^A(1^n) = 0 | \chi_{L_A}(1^n) = 0] \\
P_{\text{uni}} &\stackrel{\text{def}}{=} Prob_A[\exists! w : \forall u \in C_n, wu \in A] \\
&\qquad\qquad \cdot Prob_A[M^A(1^n) = 1 | \exists! w : \forall u \in C_n, wu \in A] \\
P_{\text{rest}} &\stackrel{\text{def}}{=} Prob_A[\chi_{L_A}(1^n) = 1 \text{ and } \exists w' \neq w : \forall u \in C_n,\ wu \in A \,\&\, w'u \in A]
\end{aligned}$$

(That is, P_{no} represents the probability that M is correct on an input not in L_A, P_{uni} the probability M is correct on an input with a unique witness, and P_{rest} the that M is correct on an input with several witnesses.) Looking at the various probabilities, we have:

1. By Eq. (4), $\quad Prob_A[\chi_{L_A}(1^n) = 0] = 1/\mathrm{e}$,
2. By Eq. (5), $\quad Prob_A[\exists! w : \forall u \in C_n, wu \in A] \approx 1/\mathrm{e}$.

So

$$P_{\text{rest}} \le Prob_A[\exists w' \neq w : \forall u \in C_n,\ wu \in A \,\&\, w'u \in A] \approx 1 - 2/\mathrm{e} \tag{6}$$

It is possible to see that the difference between the two probabilities

a) $p_1 \stackrel{\text{def}}{=} Prob_A[M^A(1^n) = 1 | \chi_{L_A}(1^n) = 0]$

b) $p_2 \stackrel{\text{def}}{=} Prob_A[M^A(1^n) = 1 | \exists! w : \forall u \in C_n, wu \in A]$

is very small. In fact, we can define the following oracle:

$A^{(w_0)} \stackrel{\text{def}}{=}$
- randomly choose A so that $1^n \notin L_A$ meaning $\forall w \in \{0,1\}^n \;\; \exists u \in C_n : wu \notin A$
- only on w_0 modify A so that $\forall u \in C_n, w_0 u \in A$

Then it can be seen that the probability – taken over a random A and a random selceted w of length n

b') $Prob_{A, w \in \{0,1\}^n}[M^{A^{(w)}}(1^n) = 1] = p_2$

In fact, consider the process of selecting A at random such that there exists no w for which $\forall u \in C_n, wu \in A$. Then select w_0 at random and modify A so that only in w_0 $\forall u \in C_n, w_0 u \in A$. This process is the same as selecting a random A conditioned to the existence of a unique w such that $\forall u \in C_n, wu \in A$. So the set of oracles $A^{(w)}$ considered in **b')** has the same distribution as the set of random oracles conditioned to the existence of a unique w for which $\forall u \in C_n, wu \in A$, that is considered in **b)**.

Probabilities **a)** and **b')** are then very close; in fact in **b')**, when the machine asks the first query, there is a chance only of 2^{-n} for the machine to come across the unique w such that $\forall u \in C_n, wu \in A$. (On all other w's the two oracles behave the same.) Asking the second query, there is a chance only of $1/(2^n - 1)$ for the machine to get different answers to the query, and so on. Since the total number of possible queries in n is $p(n)$ for some polynomial p, the difference $p_1 - p_2$ is approximately $\frac{p(n)}{2^n} \ll 1$ for the appropriate n that has been chosen.
The fact that these two probabilities are so close means that there is a very small chance that the machine can notice the modification that has been introduced.

Thus, since $Prob_A[M^A(1^n) = 0 | \chi_{L_A}(1^n) = 0] = 1 - Prob_A[M^A(1^n) = 1 | \chi_{L_A}(1^n) = 0] = 1 - p_1$, we see that

$$
\begin{aligned}
Prob_A[M^A(1^n) = \chi_{L_A}(1^n)] &= P_{\text{no}} + P_{\text{uni}} + P_{\text{rest}} \\
&\leq Prob_A[\chi_{L_A}(1^n) = 0] \cdot (1 - p_1) \\
&\quad + Prob_A[\exists! w : \forall u \in C_n, wu \in A] \cdot p_2 \; + \; (1 - 2\mathrm{e}^{-1}) \\
&\approx \mathrm{e}^{-1} \cdot (1 - p_1) + \mathrm{e}^{-1} \cdot p_2 + 1 - 2\mathrm{e}^{-1} \\
&= \mathrm{e}^{-1} \cdot (p_2 - p_1) + 1 - \mathrm{e}^{-1}
\end{aligned}
$$

Since the term $\mathrm{e}^{-1}(p_2 - p_1) < p(n)/2^n$ is negligible, we get

$$Prob_A[M^A(1^n) = \chi_{L_A}(1^n)] \leq 1 - \mathrm{e}^{-1} < 0.8$$

and the claim follows. ■

This also completes the proof of Theorem 4. ■

The reason for investigating how $\mathcal{P}$ and $\mathcal{NP}$ behave if relativized to a random oracle is the following. The statements in Theorems 2 and 3 only mean that both $\mathcal{P} = \mathcal{NP}$ and $\mathcal{P} \neq \mathcal{NP}$ are supported by *at least* one oracle. This cannot possibly give evidence towards, say, $\mathcal{P} \neq \mathcal{NP}$. But then, what Theorem 4 says is that, considering all possible oracles, only a negligible fraction of them supports $\mathcal{P} = \mathcal{NP}$. So it seem plausible that such an assertion supports the $\mathcal{P} \neq \mathcal{NP}$ conjecture. The underlying reasoning, if valid, should extend to any pair of complexity classes. It is known as the *Random Oracle Hypothesis* and asserts that two complexity classes are different if and only if they differ under relativization to a random oracle. Unfortunately, the Random Oracle Hypothesis turns out to be false. For example, the following can be proven:

Theorem 5 *For a random oracle A, $\text{co}\mathcal{NP}^A \not\subseteq \mathcal{IP}^A$, that is, $Prob_A[\text{co}\mathcal{NP}^A \not\subseteq \mathcal{IP}^A] = 1$*

Since it is known that $\text{co}\mathcal{NP} \subseteq \mathcal{IP}$, Theorem 5 indicates that the Random Oracle Hypothesis fails. Thus, the random-relativization separation result of Theorem 4 cannot say anything about separation of the unrelativized classes.

4 Conclusions

> Oded's Note: *This section was written by me. Although my main motivation for giving this lecture was to present these views, and although I did express them in class, little of them found their way to the above notes. (Unfortunately, in general, students tend to focus on technical material and attach less importance to conceptual discussions. At times even high-level discussions of proof ideas are gives less attention than some very low-level details.)*

The study of relativized complexity classes was initiated with the hope that it may shed some light on the unrelativized classes. Our own opinion is that this hope is highly unjustified.

Relativized results as predictors of non-relativized results. The most naive hope regarding relativized complexity classes is that the relations which hold in *some* relativized world are exactly those holding in the real (unrelativized) world. As shown above, this cannot possibly be true since different oracles may lead to different relativized results. That is, there may be a relativized world (i.e., an oracle) in which some relation holds and another in which the relation does not hold; whereas the same situation can not possibly hold in the (single) real world.

Conflicting relativizations cannot occur when one considers relativization results which hold for almost all orcales (i.e., relative to a random oracle). A new hope postulated by the "Random Oracle Conjecture" was that relations which hold in *almost all* relativized world are exactly those holding in the real (unrelativized) world. However, this conjecture has been refuted; we have mentioned above one such refutation (i.e., $\text{co}\mathcal{NP} \subseteq \mathcal{IP}$ and yet $\text{co}\mathcal{NP}^A \not\subseteq \mathcal{IP}^A$ for almost all oracles A).

We mention that not only that we have results which contradict the above naive hopes, but also the way these results are proven seems to indicate that the structure of computation of an oracle machine has little to do with the computation of an analogous regular machine. Add to this our initial comment by which not any class can be relativized (i.e., relativization requires that the class be defined in terms of some type of machines and that these machine extend naturally – at least in a syntaxtic sense – to orcale machines). We stress again that such extensions, even when they seem syntaxtically natural, may totally disrupt the "semantics" associated with the bare machine.

Our conclusion is that relativized results are poor predictors of non-relativized results. The advocates of relativization are certainly aware of the above, and their current advocacy is based on the belief that relativized results indicate limitations of proof techniques which may be applied in the real (unrelativized) world. That is –

Relativized results as indicating limitations of proof techniques. The thesis (of the above advocates) is that a relativized result asserting some relation indicates that the contrary relation *cannot be proven* in the real (unrelativized) world *via* "a proof which relativizes". This thesis is based on the hidden assumption that there is a natural (generic) way of extending a real (unrelativized) proof into a version which refers to the corresponding oracle classes. The thesis suggests that one should consider the question of whether such a natural extention is indeed valid in the current case. Suppose that some result fails relative to some oracle, then the thesis asserts that any proof for which the natural relativization remains valid will fail to prove the real (unrelativized) result (since it would have proven the same result also relative to any oracle, whereas the relativized result asserts the existence of an oracle for which the result does not hold).

Our objection to the above thesis is based on the objection to the assumption that such a natural (generic) way of extending a real (unrelativized) proof into a version which refers to the "corresponding oracle classes" exists. Indeed certain known proofs (e.g., the time hierarchy) extend naturally to the relativized world, but we don't see a generic procedure of transforming real (unrelativized) proofs into relativized ones. (Recall our warning that not all complexity classes have a relativized version.) Without such generic procedure of transforming unrelativized proofs into relativized ones, it is not clear what is suggested by the above thesis. Furthermore, one can transform any proof which is known to "relativize" into a proof (of the same statement) that does not "relativize" (via the process witnessing the former "relativization"). Of course this transformation can be undone by a new transformation, but the latter can not be considered generic in any natural sense.

What is left is a suggestion to use relativized results as a tool for testing the viability of certain approaches for proving real (unrelativized) results. That is –

Relativized results as a debugging tool. Suppose one has a vague idea on how to prove some result regarding the real world. Further suppose that one is not sure whether this idea can be carried out. Then the suggestion is to ask whether the same idea seems to apply also to the relativized world and whether a contrary relativization result is known. The thesis is that if the answer to both questions is possitive then one should give-up hope that the idea can work (in the real world).

We object even to this minimalistic suggestion. If one has a vague idea which may or may not work in the real work, possibly depending on unspecified details, then why should one believe in the validity of statements inferred from applying a vague procedure (i.e., "relativization") to this vague idea? Indeed, in some cases the (relativized) indication obtained (as suggested above) may save time which could have been wasted pursuing an unfruitful idea, but in some cases it may cause to abandon a fruitful idea. We see no reason to believe that the probability of the first event is greater than that of the second, and point out that the pay-off (benefit/damage) seems to be much greater in the second.