Technical Reports on Mathematical and Computing Sciences: TR-C169
title: Relativized Collapsing Results under Stringent Oracle Access
author: Jin-Yi Cai[1] and Osamu Watanabe[2]
affiliation:
1. Computer Sciences Dept., University of Wisconsin, Madison, WI 53706, USA
   (jyc@cs.wisc.edu)
2. Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology
   Meguro-ku Ookayama, Tokyo 152-8552, Japan
   (watanabe@is.titech.ac.jp)
acknowledgments to financial supports:
1. Supported in part by NSF grants CCR-0208013 and CCR-0196197 and
   U.S.-Japan Collaborative Research NSF SBE-INT 9726724.
2. Supported in part by the JSPS/NSF Collaborative Research 1999 and
   by the Ministry for Education, Grant-in-Aid for Scientific Research (C), 2001.

Abstract. For relativized arguments, we propose to restrict oracle queries to "stringent" ones. For example, when comparing the power of two machine models relative to some oracle set X, we restrict that machines of both types ask queries from the same segment of the set X. In particular, for investigating polynomial-time (or polynomial-size) computability, we propose *polynomial stringency*, bounding query length to any fixed polynomial of input length. Under such stringent oracle access, we show, for example, an oracle G such that $\mathrm{P}^G = \Sigma_d^{\mathrm{p},G}$, for any constant $d \geq 1$.

1. Introduction

In computational complexity theory, many relationships between complexity classes have been open, and for many of them, it has been suggested they are not provable by any of our known proof techniques. This pessimistic belief is, in fact, supported by *relativized results*. Consider, for example, the most famous P vs. NP conjecture. For a relationship beteen P and NP, we can relativize both ways. That is, there exist two oracles A and B such that $\mathrm{P}^A = \mathrm{NP}^A$ (the *collapsing*) holds and $\mathrm{P}^B \neq \mathrm{NP}^B$ (the *separation*) holds. Intuitively, for each oracle set X, the relative computation model allowing oracle queries to X provides a "relativized complexity world" where all computation is the same as our real world except that one can use some special set of instructions, i.e., queries to the oracle set X. Then it is said that almost all known proof techniques are *relativizable*; that is, they would go through in such relativized worlds. Therefore, having the above oracles A and B means that almost all known proof techniques are hopeless for resolving the P vs. NP conjecture.

In this paper, however, we point out that some of those relativized results (in particular, the collapsing ones) are not stringent enough compared with our intuition explained above. More precisely, relativized collapsing results are often proved by allowing stronger usage of an oracle to a simulating machine than a simulated machine. Here we propose to restrict oracle access to more "stringent" one. In particular, for investigating polynomial-time (or polynomial-size) computability, we introduce "polynomial stringency", bounding query length to any fixed polynomial of input length. Under such stringent oracle access, we show, for example, an oracle G such that $\mathrm{P}^G = \Sigma_d^{\mathrm{p},G}$, for any constant $d \geq 1$.

Some Examples & Our Proposal

We use some examples to explain the motivation of proposing stringent oracle access. (Below we use standard complexity classes and standard results in computational complexity theory without explaining them. See, e.g., text books [BDG89, DK00] for the explanation.)

First consider the standard *collapsing* relativized argument for the P vs. NP conjecture; that is, showing an oracle A satisfying $\mathrm{P}^A = \mathrm{NP}^A$. For this set A, we may consider any complete set for PSPACE. For example, let us consider the problem QBF. From its PSPACE-completeness, we have $\mathrm{PSPACE} \subseteq \mathrm{P}^{\mathrm{QBF}}$. On the other hand, $\mathrm{NP}^{\mathrm{QBF}} \subseteq \mathrm{PSPACE}$. Thus, we have $\mathrm{NP}^{\mathrm{QBF}} \subseteq \mathrm{P}^{\mathrm{QBF}}$ (hence, $\mathrm{P}^{\mathrm{QBF}} = \mathrm{NP}^{\mathrm{QBF}}$).

While this is a valid argument, to see how queries are used for this collapsing argument, let us consider a bit more in detail the simulation of any NP query machine $\mathcal{Q}_0$ by some P query machine $\mathcal{Q}_1$ relative to QBF. Assume that $\mathcal{Q}_0$ is $\mathcal{O}(n^k)$-time bounded machine. Then one can easily design some PSPACE machine $\mathcal{M}_0$ simulating $\mathcal{Q}_1^{\mathrm{QBF}}$; more precisely, for any input x of length n, $\mathcal{M}_0(x)$ simulates $\mathcal{Q}_0^{\mathrm{QBF}}(x)$ using $\mathcal{O}(n^k)$ space. Then by using the standard reduction, we can express the computation $\mathcal{M}_0(x)$ in terms of some QBF formula F_x. That is, $\mathcal{M}_0$ accepts x if and only if $F_x \in \mathrm{QBF}$. Thus, by asking one query (namely, F_x) to the oracle QBF, one can simulate $\mathcal{M}_0(x)$ (which is $\mathcal{Q}_0^{\mathrm{QBF}}(x)$). Since F_x is polynomial-time computable from x, this proves that $\mathcal{Q}_0^{\mathrm{QBF}}$ is simulated by some deterministic polynomial-time machine $\mathcal{Q}_1$ asking one query to QBF.

Now compare the length of queries asked by these two machines $\mathcal{Q}_0$ and $\mathcal{Q}_1$. For any input x of length n, let the queries that $\mathcal{Q}_0$ makes on x be up to ℓ in length, which is bounded by $O(n^k)$. On the other hand, $\mathcal{Q}_1$ asks F_x, which is a QBF formula of length $> \ell$. Thus, so long as a standard reduction to QBF is used to define F_x, the simulating machine $\mathcal{Q}_1$ asks strictly longer queries than those asked by the simulated machine $\mathcal{Q}_0$.

We would find that most relativized *collapsing* arguments follow more or less the same style; that is, a simulating machine asks longer queries than those asked by a simulated machine, and a collapsing result is proved by using (sometimes implicitly) this superiority of the simulating machine to the simulated one. We think that this type of simulation does not fit to our intuitive motivation of relativization.

More technically, this type of relativized argument may cause some difference between relativized worlds and the real world. Consider the following example. Hopcroft, Paul, and Valiant [HPV77] proved that any machine with time complexity t can be simulated by some machine with space complexity $s = t/\log t$ that is slightly less than t. In this situation where we deal with specific bounds like t, to allow the simulating machine oracle access beyond what's allowed for the simulated machine will have manifestly unacceptable consequences. We can reason as follows: For any time complexity t, let $s = t/\log t$ be the space complexity in the simulation. Take an intermediate bound s' that is slightly more than s but slightly less than t (all asymptotically). Then we can construct an oracle X such that $\mathrm{DSPACE}^X(s') \subseteq \mathrm{DTIME}^X(t)$ holds, with the device of "unequal access". The idea is to encode all $\mathrm{DSPACE}^X(s')$ languages (up to a given length n) in X at length $t(n)$. Here, following the standard formulation for the space bounded relativized computation, we assume that any s'-space machine have access only up to length $s'(n) < t(n)$; then the above encoding part is beyond where the s'-space machine can look. On the other hand, if the Hopcroft-Paul-Valiant simulation were to be relativizable with this "unequal access" to the oracles, then we have $\mathrm{DTIME}^X(t) \subseteq \mathrm{DSPACE}^X(s)$. Thus, $\mathrm{DSPACE}^X(s') \subseteq \mathrm{DTIME}^X(t) \subseteq \mathrm{DSPACE}^X(s)$, violating the relativization of space hierarchy theorem.

For these considerations, we propose a new model of stringent relativization in this paper. We propose to restrict relativized computation so that both machines (i.e., simulated and simulating machines) can see the same segment of an oracle set. For example, we propose to limit query length to any *fixed* polynomial for both query machines, which we call "polynomially stringent" relativization. In the next section, we define this notion precisely.

In a paper currently in submission [CW02] we considered some positive and negative results regarding proving circuit lower bounds for specific polynomial bounds in the polynomial hierarchy. The concern for stringent oracle access was made manifestly necessary for specific polynomial bounds. In that paper, we proved that under stringent oracle access it is possible to collapse PH to P/poly. However, we believe this notion of stringent oracle access is more widely applicable, and we undertake to define this notion carefully in this paper. The main result of this paper is an improvement of the result from [CW02] from P/poly to P, i.e., we show that under stringent oracle access P = PH. We note that it is open how to exhibit a stringent relativization with which P = PSPACE.

2. Stringent Relativization

As discussed in Introduction, we propose to consider more stringent relativization, by restricting access to an oracle. In this paper, we consider the following restricted relativized computation. (Below by a "length bound", we mean a function from $\mathbf{N}$ to $\mathbf{N}$ that is "reasonable" such as time/space constructible in each context.)

Definition 1. Let Q be any query machine (or circuit family with query gates) of any type. For any length bound $\ell(n)$, we say Q is executed *under $\ell(n)$ length restricted relativization* (relative to an oracle set X) if any query of length $> \ell(n)$ gets 'no' answer (while any query of length $\leq \ell(n)$ gets an answer consistent with X) on the computation of Q of any input of length n.

Roughly speaking, for stringent relativizations, we propose to compare two machines under length restricted relativization. That is, compare query machines Q_1 and Q_2 executed under length restricted relativization, and discuss, for example, whether Q_1 simulates Q_2 relative to some oracle X under length restricted relativization.

While two machines can be compared in this way, some care is needed in order to obtain a reasonable definition for comparing two complexity classes. Here we propose the following definition for a basic "stringent relativization" notion.

Definition 2. Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be any complexity classes, and let $\ell(n)$ be any length bound.

(1) For any oracle set G, we say that containment $\mathcal{C}_1 \subseteq \mathcal{C}_2$ is shown *by $\ell(n)$ length stringent relativization relative to G* (or, in short, $\mathcal{C}_1^G \subseteq \mathcal{C}_2^G$ *under $\ell(n)$ length stringent oracle access*) if for any query machine $\mathcal{Q}_1$ for $\mathcal{C}_1$, there exists some query machine $\mathcal{Q}_2$ for $\mathcal{C}_2$ such that $L(\mathcal{Q}_2^G) = L(\mathcal{Q}_1^G)$ holds when both machines are executed under $\ell(n)$ length restricted relativization.

(2) For any oracle set G, we say that separation $\mathcal{C}_1 \not\subseteq \mathcal{C}_2$ is shown *by $\ell(n)$ length stringent relativization relative to G* (or, in short, $\mathcal{C}_1^G \not\subseteq \mathcal{C}_2^G$ *under $\ell(n)$ length stringent oracle access*)

if for some query machine $\mathcal{Q}_1$ for $\mathcal{C}_1$, no query machine for $\mathcal{C}_2$ exists that simulates $\mathcal{Q}_1^G$ relative to G, when both machines are executed under $\ell(n)$ length restricted relativization.

Remark. We will use symbol G to denote an oracle for stringent relativization.

Clearly, bounding query length is vacuously true in the above definition, if the length bound $\ell(n)$ is larger than the bound allowed for one of the complexity classes $\mathcal{C}_1$ and $\mathcal{C}_2$. Thus, one should choose an appropriate query bound. In particular, for discussing relationships to polynomially resource bounded classes, we will consider the following stringency in this paper.

Definition 3. Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be any complexity classes. We say that their relationship (i.e., containment or separation) is shown *by polynomially stringent relativization* if for any length bound $\ell(n)$, $\Omega(n) \leq \ell(n) \leq n^{O(1)}$, there exists some oracle relative to which the relationship is shown by $\ell(n)$ length stringent relativization.

Remark. An oracle can be (in most cases "should be") defined for each length bound $\ell(n)$. But since each oracle for $\ell(n)$ is defined similarly (in most cases), we will simply refer them as if a single oracle and often state as, e.g., "$\mathcal{C}_1^G \subseteq \mathcal{C}_2^G$ by polynomially stringent relativization."

Notice that, in our definition, we compare complexity classes for each fixed query length bound $\ell(n)$. The stringency would become much weaker if we allow to use different length bound for each pair of machines. For example, consider the relationship NP $\subseteq$ P. If we can use *some* polynomial for a query length bound for simulating a given $p(n)$-time bounded nondeterministic machine, then the standard relativization that explained in Introduction works by using $q(n)$ $(> p(n))$ as a length bound.

An Example

For an example of polynomially stringent relativization, we show the following result.

Theorem 1. We can show NP $\subseteq$ P/poly by polynomially stringent relativization. We show, for any polynomial length bound $\ell(n) \geq (2+\epsilon)n$ for some ϵ, we can construct an oracle G such that $\mathrm{NP}^G \subseteq (\mathrm{P/poly})^G$ by $\ell(n)$ length stringent relativization.

Notice first that for the polynomially stringent relativization, a higher collapse is not immediate from a lower one. For example, $\mathrm{NP}^G \subseteq \mathrm{P}^G/\mathrm{poly}$ does not necessarily imply $\mathrm{P}^{\mathrm{NP}^G} \subseteq \mathrm{P}^G/\mathrm{poly}$, ..., $\mathrm{PH}^G \subseteq \mathrm{P}^G/\mathrm{poly}$, etc.

This theorem is provable by a well-known argument in complexity theory. For showing the idea, we explain briefly the way to define an oracle G for a given length bound, say, $\ell(n) = 3n$, and a given NP query machine $\mathcal{Q}$.

We define G by stage. Suppose that $G^{\leq 3n-1}$ has been already defined. Our task is to define $G^{=3n}$ so that, for every $x \in \{0,1\}^n$, $\mathcal{Q}^G(x)$ can be simulated by some simple circuit C_n relative to G by asking queries of length $\leq 3n$. Starting from 0^n, strings of $\{0,1\}^n$ are processed one by one lexicographically, and some part of $G^{=3n}$ is defined. Suppose that a string x is currently considered. We examine the execution of $\mathcal{Q}$ relative to the partially defined G, searching for some nondeterministic path of $\mathcal{Q}$ that accepts x *if* G is extended appropriately. If such an accepting path exists, then we extend G (i.e., fix the membership of some strings in $\{0,1\}^{3n}$ to G) consistent with this accepting path and proceed to the next input string. Otherwise, (i.e., $\mathcal{Q}$ rejects x no matter how G is extended), we simply proceed to the next input string.

Let $\widetilde{G}$ be the partially defined set obtained by examining all input strings of length n. A key observation, which is clear from the construction, is that the computation of $\mathcal{Q}^G$ is the same on $\{0,1\}^n$ no matter how G is extended from $\widetilde{G}$. Furthermore, to define $\widetilde{G}$, we have fixed their membership on at most $\mathrm{poly}(n) \cdot 2^n$ strings in $\{0,1\}^{3n}$, since $\mathcal{Q}$ asks at most polynomial number of strings on each nondeterministic path. Therefore, there is some string u of length $2n$ such that no string in $u\{0,1\}^n$ is touched when defining $\widetilde{G}$. We simply use this part to encode the information whether $\mathcal{Q}^G$ accepts x or not. That is, for each $x \in \{0,1\}^n$, the membership of a string ux is determined so that $ux \in G \Leftrightarrow \mathcal{Q}_i^{\widetilde{G}}$ accepts x. Then for each $x \in \{0,1\}^n$, one can easily compute $\mathcal{Q}^G(x)$ by checking whether ux is in G. For this computation, it is necessary to know this key string u, but this information can be hardwired in a polynomial-size circuit.

Now using Theorem 1 as an example, let us discuss the meaning of "relativized results". Imagine three possible claims of proving NP $\not\subseteq$ P/poly:-

(i) a proof totally specific to a specific set, and uses specific properties of the set and its combinatorics,

(ii) a proof for a general machine or circuit, and the proof can be carried through if we allowed the machine and circuit to access any but the same segment of any oracle, and

(iii) a proof more general, and the proof can go through even if we allowed the machine and circuit to access different segments of the oracle.

Any relativization to the contrary says nothing about the first possibility. The above Theorem 1 rules out possibility (ii) and (iii). On the other hand, by the standard $\mathrm{NP}^A \subseteq \mathrm{P}^A/\mathrm{poly}$ result, which is immediate from $\mathrm{NP}^{\mathrm{QBF}} \subseteq \mathrm{P}^{\mathrm{QBF}}$, we can only rule out (iii), but not (ii).

3. Class P vs. Class $\oplus$P

A main result of this paper is to show P = PH by polynomially stringent relativization. But before discussing this relativized result, let us first consider the relation between P vs. $\oplus$P and show P = $\oplus$P by polynomially stringent relativization, which is a good introduction to the proof for our main result $\mathrm{P}^G = \mathrm{PH}^G$.

Theorem 2. For any polynomial length bound $\ell(n) \geq 3n$, we can construct an oracle G such that $\oplus\mathrm{P}^G \subseteq \mathrm{P}^G$ by $\ell(n)$ length stringent relativization.

Remark. The same result is provable for the relationship between P and Mod_p class, for any prime p.

Proof. Here we fix the query length bound to $\ell(n) = 3n$. It is easy to see the following argument works for any larger bound.

Let $\mathcal{Q}_1, \mathcal{Q}_2, \ldots$ be a standard enumeration of all $\oplus$P query machines. Our goal is to construct an oracle G with which it is possible to simulate any $\mathcal{Q}_i$ by some polynomial-time deterministic machine under $\ell(n)$ length restricuted relativization. Let us fix any $\oplus$P machine $\mathcal{Q}$ and any input length n, and discuss how to design G so that some P machine can simulate $\mathcal{Q}^G$ on $\{0,1\}^n$ with help of G. Let $m = n^{O(1)}$ be the maximum number of queries asked by $\mathcal{Q}$ on any nondeterministic path on any input of length n. We assume that n is large enough to satisfy $m \ll 2^n$.

Let $\ell = \ell(n)$ $(= 3n)$ and $L = 2^\ell$. By the stringent condition, we assume that $\mathcal{Q}$ as well as our simulating P machine asks only queries of length $\leq \ell$ to G (on any length n input). We may assume that $G^{<\ell}$ has been already defined and that our task is to assign strings for the oracle G at length ℓ appropriately.

We want to define $\{S_x\}_{x\in\{0,1\}^n}$ to be a family of pair-wise disjoint subsets of $\{0,1\}^\ell$, with $|S_x| \approx nm$. For example, for $s = \lceil \log_2 nm \rceil$, we can define

$$S_x \;=\; \{\, xu0^{\ell-(n+s)} \mid u \in \{0,1\}^s \,\},$$

and define $Z = \bigcup_{x\in\{0,1\}^n} S_x$. Let $M = |Z|$; note that $M \le 2nm2^n \ll 2^\ell$. Let us name the strings in Z as $z_1, z_2, \ldots, z_M$. We will also use z_i to denote a Boolean variable, which takes 1 (resp., 0) if and only if the string z_i is put into the oracle G.

We will define $G^{=\ell}$ to be a subset of Z, i.e., we fix all strings of length ℓ not in Z out of G. Then, since $G^{<\ell}$ has been already fixed (in the earlier stage), for any input $x \in \{0,1\}^n$, $\mathcal{Q}^G(x)$ is completely determined by the membership of each z_i to G. That is, $\mathcal{Q}^G(x)$ is a function on Boolean variables $z_1, \ldots, z_M$. Furthermore, since $\mathcal{Q}^G(x)$ is a parity computation asking at most m queries on each nondeterministic path, we may consider $\mathcal{Q}^G(x)$ as a parity (or its negation) of (at most) exponential number of conjunctions of at most m variables from $z_1, \ldots, z_M$. Thus, $\mathcal{Q}^G(x)$ is expressed by a polynomial $f_x(z_1, \ldots, z_M)$ of degree $\le m$ under mod 2. Note here that $f_x(z_1, \ldots, z_M)$ is multilinear, because we may assume that each string is not queried more than once on each nondeterministic path.

Now we would like to assign $z_1, \ldots, z_M$ so that the following system of equations $(*1)$ holds (under the mod 2 computation) for $\{0,1\}^n = \{x_1, \ldots, x_N\}$ (where $N = 2^n$).

$$(*1)\quad \left\{ \begin{array}{lcl} f_{x_1}(z_1, \ldots, z_M) & = & 1 - \prod_{z_j \in S_{x_1}} z_j, \\ & \vdots & \\ f_{x_N}(z_1, \ldots, z_M) & = & 1 - \prod_{z_j \in S_{x_N}} z_j. \end{array} \right.$$

If this assignment is feasible (i.e., the oracle G is constructed satisfying $(*1)$), then for any $x \in \{0,1\}^n$, one simply needs to check the membership of elements of S_x; $\mathcal{Q}^G(x)$ can be computed as $1 - \prod_{z_j \in S_x} z_j$.

Suppose, for a contradiction, that this is impossible to achieve. Then, since for every 0 or 1 value of $z_1, \ldots, z_M$, each f_x takes a 0 or 1 value, it follows that for every assignment to the $z_1, \ldots, z_M$, there exists some x such that

$$f_x(z_1, \ldots, z_M) \;=\; \prod_{z_j \in S_x} z_j.$$

Thus, for all 0,1-assignments to $z_1, \ldots, z_M$, we have

$$\prod_{1\le i\le N} \left[\prod_{z_j \in S_{x_i}} z_j - f_{x_i}(z_1, \ldots, z_M) \right] \;=\; 0.$$

Then it follows from a well-known claim (Claim 1 stated below) that modulo the ideal $I = (z_1^2 - z_1, \ldots, z_M^2 - z_M)$, the lefthand side expression is identical to 0. In other words, we have the identity

$$\prod_{1\le i\le N} \prod_{z_j \in S_{x_i}} z_j \;=\; L(z_1, \ldots, z_M),$$

in the ring $\mathbf{Z}_2[z_1,\ldots,z_M]/I$, where L is a polynomial of degree at most $(N-1)2^s+m$. On the other hand, the degree of the lefthand side of the above equality is $N2^s$, which is larger than $(N-1)2^s+m$ because $2^s>m$. A contradiction.

This is the way to define (a part of) an oracle for simulating one $\oplus$P query machine $\mathcal{Q}$. For completing the proof, it suffices to show how to simulate machines $\mathcal{Q}_1,\ldots,\mathcal{Q}_k$ up to some k. For this, we simply need to use different segments of $\{0,1\}^\ell$ for the family of sets $\{S_{i,x}\}_{x\in\{0,1\}^n}$ for each machine $\mathcal{Q}_i$, $1\le i\le k$. For defining the oracle, we need to consider the system of equations defined by all machines $\mathcal{Q}_1,\ldots,\mathcal{Q}_k$, but it is easy to see that the same argument also works for this situation. □

Claim 1. For any prime p, let $F(x_1,\ldots,x_n)$ be a polynomial evaluated to 0 modulo p on all 0,1-assingments to $x_1,\ldots,x_n$. Then modulo the ideal $I=(x_1^2-x_1,\ldots,x_n^2-x_n)$, i.e., in the ring $\mathbf{Z}_p[x_1,\ldots,x_n]/I$, $F(x_1,\ldots,x_n)$ is identical to 0.

4. Class P vs. Class PH

We now show P = PH by polynomially stringent relativization. Technically, we prove Theorem 3 below. The construction of a polynomially stringent oracle for P = PH can be done by a standard argument following Theorem 3, since PH is a countable union of classes $\Sigma_d^{\rm p}$.

Theorem 3. For any $d\ge 0$, we can show $\Sigma_d^{\rm p}\subseteq{\rm P}$ by polynomially stringent relativization. More specifically, for any polynomial length bound $\ell(n)\ge 3n$, we can construct an oracle G such that $\Sigma_d^{\rm p,G}\subseteq{\rm P}^G$ by $\ell(n)$ length stringent relativization.

Before stating our proof in detail, we explain its outline. We begin with recalling the decision tree version of the Switching Lemma.

Recall some notions and notations. For any Boolean function f over variables $x_1,\ldots,x_n$, a *random restriction* ρ is a random function that assigns each x_i either 0, 1, or $*$, with probability $\Pr[\rho(x_i)=*]=p$ (for some specified parameter p) and $\Pr[\rho(x_i)=0]=\Pr[\rho(x_i)=1]=(1-p)/2$, for each i independently. Assigning $*$ means to leave it as a variable. Let $f\mid_\rho$ denote a function obtained by this random restriction.

The decision tree complexity of a Boolean function f, denoted by ${\rm DC}(f)$, is the smallest depth of a Boolean decision tree computing the function. It can be shown easily that if ${\rm DC}(f)\le t$, then f can be expressed both as an AND of OR's as well as an OR of AND's, with bottom fan-in at most t. Moreover, what is crucial for our argument is the following property: If ${\rm DC}(f)\le t$, then the value of f is determined, when one assigns values to at most t variables, no matter how values are assigned to f's variables. (The choice of (at most) t variables may vary depending on each assignment to the variables.)

Adapting Håstad's proof to the decision tree model, one can prove the following.

Lemma 1. For any depth $d+1$ Boolean circuit C on $z_1,\ldots,z_L$, with bottom fan-in at most t,

$$\Pr[\ {\rm DC}(C\mid_\rho)\ge t\ ]\ \le\ \frac{{\rm size}(C)}{2^t},$$

where ρ is a random restriction with the parameter $p=\Pr[z_i=*]=1/(10t)^d$.

We now explain our construction. Here again fix the query length bound to $\ell(n) = 3n$. Also fix any Σ_d^{p} machine $\mathcal{Q}$ and any sufficiently large input length n, and explain the way to construct an oracle G with which some polynomial-time deterministic machine can simulate the computation of $\mathcal{Q}$ on inputs of length n by asking queries of length $\ell(n) = 3n$. Constructing the oracle for the simulation of *all* Σ_d^{p} machines can be done in the same way as we explained in the previous section.

Thus, from now on, we are concerned with the simulation of $\mathcal{Q}$ on 2^n inputs of length n. Let m be an integer bounding $\mathcal{Q}$'s running time on inputs of length n, where $m = O(n^k)$ for some $k \geq 0$. Let $\ell = \ell(n)$ $(= 3n)$ and $L = 2^\ell$. We assume that G has been already defined up to length $\ell - 1$; then our task is to assign strings in $\{0,1\}^\ell$ to G. Let us name the strings in $\{0,1\}^\ell$ as $z_1, z_2, \ldots, z_L$. As before, z_i will be also used to denote a Boolean variable indicating whether the string z_j is put into G. Let Z denote the set of all Boolean variables $z_1, \ldots, z_L$.

For any input string $x \in \{0,1\}^n$, consider the computation of $\mathcal{Q}(x)$ relative to G. Since G is defined up to length $\ell - 1$ and the length of queries made by $\mathcal{Q}(x)$ is at most ℓ, the computation $\mathcal{Q}^G(x)$ is regarded as a function from $z_1, \ldots, z_L$ to $\{0,1\}$. Furthermore, since $\mathcal{Q}$ is a Σ_d^{p} machine, by a standard interpretation of the Σ_d^{p} query computation, we may regard $\mathcal{Q}^G(x)$ as a depth $d+1$ circuit on input variables $z_1, \ldots, z_L$, of size at most $m2^m$ and bottom fan-in at most m. Our first step is to assign a random restriction ρ to $z_1, \ldots, z_L$ of an appropriate probability $p_0 = \Pr[\, z_i = * \,]$. By Lemma 1, with high probability the circuit is reduced to small depth (i.e., depth $t = 2m$) decision trees. In fact, by choosing p_0 appropriately, we can even show that with high probability, a random restriction convert *all* circuits for all 2^n input strings to depth t decision trees.

We then show, by a standard argument, that a small depth decision tree can be expressed by a low degree (i.e., degree $2m$) polynomial over the finite field $\mathbf{Z}_3$. That is, after the random restriction, each computation $\mathcal{Q}^G(x)$ is expressed as a degree $2m$ polynomial p_x. This is a situation similar to the parity computation; we can use essentially the same technique to encode the value of each $\mathcal{Q}^G(x)$ for some appropriately assigned $z_1, \ldots, z_L$.

More specifically, our encoding is implemented as follows. For each $x \in \{0,1\}^n$, we reserve a segment $S_x \subseteq \{0,1\}^\ell$ of enough size, precisely speaking, of size $20m/p_0$, which is polynomial in n. These segments are chosen so that the family $\{S_x\}_{x \in \{0,1\}^n}$ is a pair-wise disjoint. As in the proof of previous section, we would like to use the assignment of variables in S_x to encode the result of $\mathcal{Q}^G(x)$. Here notice that the random restriction ρ has already assigned values to some of the variables in S_x. But since (i) $|S_x| = 20m/p_0$, and (ii) variables remain unassigned with probability p_0, we can prove that with high probability, *all* segments S_x have at least $3m$ unassigned variables after the random restriction. We use these unassigned variables for encoding.

It follows from the above discussion that there exists a random restriction satisfying the following.

(a) Each computation $\mathcal{Q}^G(x)$ is reduced to a small depth decision tree T_x.

(b) Each segment S_x has at least $3m$ unassigned variables, i.e., assigned $*$ by the restriction.

Fix ρ_0 to be one such restriction. Denote by Z_0 the set of variables in $\bigcup_{x \in \{0,1\}^n} S_x$ that are assigned $*$ by ρ_0, and rename variables so that $Z_0 = \{z_1, \ldots, z_M\}$ and $Z - Z_0 = \{z_{M+1}, \ldots, z_L\}$.

The restriction ρ_0 may assign $*$ to some variables in $Z - Z_0$, but assign all such variables to

0. Then as explained above, the result of each computation of $\mathcal{Q}^G(x)$ is expressed as a degree $2m$ polynomial $p_x(z_1,\dots,z_M)$ over the integers $\mathbf{Z}$. For each x, we try to express $p_x(z_1,\dots,z_M)$ as the parity of S_x, i.e., $\oplus_{z_i\in S_x} z_i$. (Note that S_x contains variables not in $Z_0=\{z_1,\dots,z_M\}$ whose values are already fixed. By the term $\oplus_{z_i\in S_x} z_i$, we mean the parity of all variables in S_x including such determinates.) In other words, we choose an assignment to $z_1,\dots,z_M$ so that the following system of equations $(*2)$ holds for $\{0,1\}^n=\{x_1,\dots,x_N\}$, where $N=2^n$.

$$(*2)\ \left\{\begin{array}{rcl} p_{x_1}(z_1,\dots,z_M) & = & \oplus_{z_j\in S_{x_1}} z_j, \\ & \vdots & \\ p_{x_N}(z_1,\dots,z_M) & = & \oplus_{z_j\in S_{x_N}} z_j. \end{array}\right.$$

Following an argument similar to the one in the previous section, we can show that it is indeed possible to find such an assignment. Then the rest of the argument is similar to the one for the parity case.

Now we specify the parameters and the conditions explained above, and describe our proof precisely.

Here as in the above we focus on the simulation of some Σ_d^{p} query machine $\mathcal{Q}^G$ on N $(=2^n)$ inputs of length n for sufficiently large n. Let $m=O(n^k)$ be an integer bounding $\mathcal{Q}$'s running time on length n inputs, and let $\ell=\ell(n)$ $(=3n)$ and $L=2^\ell$. We assume that G has been already defined up to length $\ell-1$, and regard the computation of $\mathcal{Q}(x)$ as a function over Boolean variables $z_1,\dots,z_L$, where each z_i indicates whether the string $z_i\in\{0,1\}^\ell$ is put into the oracle or not. Furthermore, we may consider $\mathcal{Q}^G(x)$ as a circuit C_x of depth $\le d+1$, size $\le m2^m$, and bottom fan-in $\le m$.

As explained above, we consider a random restriction to the variables $z_1,\dots,z_L$, with $p_0=1/(20m)^d$ being the probability $\Pr[\,z_i=*\,]$. For each $x\in\{0,1\}^n$, the segment S_x is defined by $S_x=\{\,xu0^{\ell-n-n_0}\ :\ u\in\{0,1\}^{n_0}\,\}$, where $n_0=\lceil\log_2 20m/p_0\rceil=\lceil(d+1)\log_2 20m\rceil$. Clearly, any S_x and $S_{x'}$, for $x\neq x'$, are disjoint, and $|S_x|$ is of size larger than $20m/p_0$ but still polynomial in n.

For our some random restriction ρ, we require the following two conditions.

(a) For every $x\in\{0,1\}^n$, the circuit C_x is reduced to a depth $t=2m$ decision tree.

(b) For every $x\in\{0,1\}^n$, the segment S_x has at least $3m$ unassigned variables.

By using Lemma 1 and Chernoff's bound (see, e.g., Corollary A.1.14 of [AS00]), it is easy to show the following claim; hence, there exists some random restriction satisfying both (a) and (b).

Claim 2. Under our choice of parameters, the probability that a random restriction ρ satisfies both (a) and (b) is not zero.

Consider one of the restrictions ρ_0 satisfying both (a) and (b). We define G based on this ρ_0; that is, we will complete the partial assignment ρ_0 by determining the value of variables that are assigned $*$ by ρ_0, and then define $G^{=\ell}$ to be the set of strings $z_i\in\{0,1\}^\ell$ such that the corresponding variable z_i is assigned 1 by this extension of ρ_0. Let Z_* be the set of variables assigned $*$ by ρ_0. From the condition (b) it follows that each S_x has at least $3m$ variables in Z_*. For each S_x, we pick lexicographically the first $3m$ such variables, and define Z_0 to be the set of those variables, for all x. Note that Z_0 has exactly $3mN$ variables because all S_x's are disjoint. By renaming variables, we assume that $Z_0 = \{z_1, \ldots, z_M\}$, where $M = 3mN$. We assign 0 to all variables in $Z_* - Z_0$; thus, Z_0 is the set of remaining unassigned variables.

From the condition (a), the computation $\mathcal{Q}(x)$ for each $x \in \{0,1\}^n$ (relative to $G^{\leq \ell}$ defined so far) is represented as a depth $2m$ decision tree T_x on $z_1, \ldots, z_M$. Then we can express T_x as a low degree polynomial p_x in the following way. For the trivial decision tree of depth 0 (where no variable is accessed at all), the value is a constant 0 or 1. Inductively, suppose in the decision tree T, the first branch is on the variable z_i, and depending on its value, its left subtree T_0 if $z_i = 0$, or the right subtree C_1 if $z_i = 1$ is selected. Then we see immediately that the polynomial $p = (1-z_i)p_0 + z_i p_1$ evaluates to the truth value of T, where p_0 and p_1 are the polynomials that correspond to the subtrees T_0 and T_1 respectively. In this way, we can define the polynomial p_x computing the value of T_x. Note here that the degree of p is at most $1 + \max\{\deg p_0, \deg p_1\}$. In particular, we have $\deg p_x \leq 2m$ for each decision tree T_x.

For these polynomials p_x, $x \in \{0,1\}^n$, we show below that there exists an 0,1-assignment to variables in Z_0 satisfying $(*2)$ above. We complete ρ_0 by using one of such assignments, and define $G^{=\ell}$ using it. Then one can compute the value of $\mathcal{Q}^G(x)$, for each $x \in \{0,1\}^n$, by asking queries on all strings in S_x and taking the parity of the oracle answers. Since the size of S_x is polynomially bounded in n, this is a P computation relative to G.

Now the remaining task is to prove that $(*2)$ has a solution. Let us first transform $(*2)$ to a system of equations in $\mathbf{Z}_3$. Note here that the polynomials p_x, though defined over the integers $\mathbf{Z}$, only evaluate to the values 0 or 1 when each z_i takes either 0 or 1. This fact is verified inductively as follows. In the above decomposition $p = (1 - z_i)p_0 + z_i p_1$, on any 0,1-assignment of variables $z_1, \ldots, z_M$, we can see that exactly one of these two terms evaluates to 1, and the other term evaluates to 0 (if we inductively assume that both the values of p_0 and p_1 are either 0 or 1). Furthermore, this property is invariant even if the polynomials are evaluated under modulo to q, for any prime q. Thus, we may argue these polynomials under the modulo q computation, for any prime q. In particular, we consider the polynomials under the modulo 3 computation, i.e., over the finite field $\mathbf{Z}_3$.

Then by a linear transformation, we can change the representation of 0 and 1 by $+1$ and -1 respectively; that is, 0 is represented by $+1$ and 1 by -1. More specifically, for each polynomial p_x, we replace z_i by $z'_i = 1 + z_i$, and express $p'_x = 1 + p_x$ as polynomials in z'_i's. Note that when $z_i = 0$ and 1 respectively, $z'_i = 1$ and -1 respectively, and similarly for p_x and p'_x. On the other hand, the parity is now expressed by simply a product. (In the following we will rewrite z_i for z'_i and p_x for p'_x.) Thus, the system of equations $(*2)$ is transformed into the following system of equations in $\mathbf{Z}_3$.

$$(*3)\quad \left\{ \begin{array}{lllll} p_{x_1}(z_1, \ldots, z_M) & = & \prod_{z_j \in S_{x_1}} z_j & = & \alpha_{x_1} \cdot \prod_{z_j \in Z_0 \cap S_{x_1}} z_j \\ & & \vdots & & \\ p_{x_N}(z_1, \ldots, z_M) & = & \prod_{z_j \in S_{x_N}} z_j & = & \alpha_{x_N} \cdot \prod_{z_j \in Z_0 \cap S_{x_N}} z_j. \end{array} \right.$$

Where each $\alpha_x \in \{-1, +1\}$ denotes the product of all determinate variables $z_i \in S_x - Z_0$.

We claim that there is at least one assignment to ± 1 for all $z_i \in Z_0$ satisfying $(*3)$. Suppose, for a contradiction, that there is no such assignment. Then, since for every ± 1 values of $z_1, \ldots, z_M$, each p_x takes a ± 1 value, it follows that for every $+1, -1$-assignment $(a_1, \ldots, a_M)$ to the z_i's, there must be at least one x such that

$$p_x(a_1, \ldots, a_M) \;=\; \alpha_x \cdot \prod_{z_i \in Z_0 \cap S_x} a_i.$$

Thus, we have

$$\prod_{1 \leq i \leq N} \left[\alpha_{x_i} \prod_{z_j \in Z_0 \cap S_{x_i}} z_j - p_{x_i}(z_1, \ldots, z_M) \right] \;=\; 0,$$

for all $+1, -1$-assignments to $z_1, \ldots, z_M$. Then it follows from Claim 1 that the lefthand side expression is identical to 0 modulo the ideal $I = (z^2 - 1 : z \in Z_0)$. In other words, we have the identity

$$\prod_{1 \leq i \leq N} \alpha_{x_i} \prod_{z_j \in Z_0 \cap S_{x_i}} z_j \;=\; L(z_1, \ldots, z_M)$$

in the ring $\mathbf{Z}_3[z_1, \ldots, z_M]/I$, where L is a multilinear polynomial of degree at most $3m(N-1)+2m$. On the other hand, the lefthand side is multilinear and its degree is $3mN$, which is larger than $3m(N-1)+2m$. A contradiction.

References

[Ajt83] M. Ajtai, Σ_1^1-formulae on finite structures, *Ann. Pure Applied Logic*, 24, 1–48, 1983.

[AS00] N. Alon and J. Spencer, *The Probabilistic Method*, John Wiley & Sons, Inc., 2000.

[BDG89] J. Balcázar, J. Díaz, and J. Gabarró, *Structural Complexity I & II*, Springer, 1989 and 1990.

[Cai86] J-Y. Cai, With probability one, a random oracle separates PSPACE from the polynomial-time hierarchy, in *Proc. 18th ACM Sympos. on Theory of Comput.*, 21–29, 1986. (The final version appeared in *J. Comp. Syst. Sci.*, 38(1), 68–85, 1989.)

[CW02] J-Y. Cai and O. Watanabe, On proving circuit lower bounds against the polynomial-time hierarchy, subimitted.

[DK00] D. Du and K. Ko, *Theory of Computational Complexity*, John Wiley & Sons, Inc., 2000.

[FSS81] M. Furst, J. Saxe, and M. Sipser, Parity, circuits, and the polynomial time hierarchy, in *Proc. 22nd IEEE Symposium on Foundations of Computer Science* (FOCS'81), IEEE, 260–270, 1981.

[Hås86a] J. Håstad, Almost optimal lower bounds for small depth circuits, in *Proc. 18th ACM Symposium on Theory of Computing* (STOC'86), ACM, 6–20, 1986.

[HPV77] J.E. Hopcroft, W.J. Paul, and L.G. Valiant, On time versus space, *J. ACM* 24(2), 332–337, 1977.

[Raz87] A. Razborov, Lower bounds on the size of bounded depth networks over a complete basis with logical addition, *Mathematical Notes of the Academy of Sciences of the USSR* 41, 333–338, 1987.

[Smo87] R. Smolensky, Algebraic methods in the theory of lower bounds for Boolean circuit complexity, in *Proc. 19th ACM Symposium on Theory of Computing* (STOC'87), ACM, 77-82, 1987.