ISSN 1342-2812

Research Reports on Mathematical and Computing Sciences

Random Access to Advice Strings and Collapsing Results

Jin-Yi Cai and Osamu Watanabe

Sept. 2004, C–199

Department of
Mathematical and
Computing Sciences
Tokyo Institute of Technology

SERIES C: **Computer Science**

Random Access to Advice Strings and Collapsing Results

Jin-Yi Cai*
Computer Sci. Dept.
Univ. of Wisconsin, Madison, WI 53706, USA
(jyc@cs.wisc.edu)
Osamu Watanabe†
Dept. of Math. and Computing Sciences
Tokyo Institute of Technology, W8-25, Tokyo 152-8552
(watanabe@is.titech.ac.jp)

Research Report C-199

Abstract

We propose a model of computation where a Turing machine is given random access to an *advice string*. With random access, an advice string of exponential length becomes meaningful for polynomially bounded complexity classes. We compare the power of complexity classes under this model. It gives a more stringent notion than the usual model of computation with relativization. Under this model of random access, we prove that there exist advice strings such that the Polynomial-time Hierarchy PH and Parity Polynomial-time $\oplus$P all collapse to P. Our main proof technique uses the decision tree lower bounds for constant depth circuits [Yao85, Cai86, Hås86], and the algebraic machinery of Razborov and Smolensky [Raz87, Smo87].

*Supported in part by NSF grants CCR-0208013 and U.S.-Japan Collaborative Research NSF SBE-INT 9726724.

†Supported in part by a Grant-in-Aid for Scientific Research on Priority Areas "Statical-Mechanical Approach to Probabilistic Information Processing" 2002-2005.

1 Introduction

In computational complexity theory, we cannot separate between many complexity classes, such as P, NP, PSPACE, etc, though such separations have been conjectured for a long time. It is generally believed that these separation results are very hard to prove. Among the supporting evidence for such a pessimistic belief, people frequently cite the collapsing results under *relativization*. Certainly, these collapsing results are sound; nevertheless, we raised some questions [CW04, CW03] on the stringency of the relativization model used for discussing the collapsing results. In this paper, we propose an alternative relativization model, which requires us to make more stringent relativized comparisons. Under this stringent relativization model, we show that a relativized P = NP relation holds as well as many other collapsing results.

Consider, for example, the most famous P $\neq$ NP conjecture. Baker, Gill and Solovay [BGS75] showed that we can relativize it in 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. It is said that most of known proofs can be *relativized*; that is, they are applicable in such relativized worlds. Therefore, having the above oracles A and B means that these proof techniques can not resolve the P vs. NP conjecture. For P vs. NP, perhaps the most straightforward proof of a relativized collapse is to use any standard PSPACE-complete set, for example, QBF as an oracle, because we then clearly have $\mathrm{P}^{\mathrm{QBF}} = \mathrm{NP}^{\mathrm{QBF}}$.

However, we feel that this argument is basd on a model of computation which is not stringent enough. This is especially true for most of the relativized collapsing results. More precisely, relativized collapsing results are often proved by allowing stronger usage of an oracle to a simulating machine than to a simulated machine.

Consider the set of polynomial-time nondeterministic query Turing machines, representing the (relativized) class NP, and let $\mathcal{Q}$ be any one such machine. A typical proof for the relativized P = NP result is to code the computation of $\mathcal{Q}$ for inputs of length n, in the oracle, in such a way that another machine $\mathcal{Q}'$ representing the (relativized) class P can recover the results. In order not to "interfere" with computations of $\mathcal{Q}$ at length n, these results are coded at locations *beyond* what $\mathcal{Q}$ can access at input of length n, and $\mathcal{Q}'$ is chosen so that its running time is large enough to access these locations. This encoding is sometimes explicitly carried out, sometimes implicitly done such as with the proof of $\mathrm{P}^{\mathrm{QBF}} = \mathrm{NP}^{\mathrm{QBF}}$. (See Section 3 for more detail and technical discussions.)

In order to rectify this problem we propose a model of computation that is more stringent than the usual relativization computation. This turns out to be equivalent to a generalization of the notion of advice strings proposed by Karp and Lipton [KL80]. Intuitively, any relativized result can be regarded as a comparison between complexity classes

under a certain nonuniform setting provided by an (infinite) advice, namely an oracle. Here we generalize the advice string formulation of Karp and Lipton by allowing random access to the advice string, so that advice strings longer than polynomial length become meaningful for polynomial-time bounded computations. Then we compare complexity classes, given such nonuniform advice strings. That is, we compare two machines $\mathcal{M}_1$ and $\mathcal{M}_2$ (representing complexity classes $\mathcal{C}_1$ and $\mathcal{C}_2$ respectively) that have random access to the same advice string s_n that is a priori given for their computation of any input of length n. Both machines will have complexity bounds that allow access to any bit of the advice string. This way we compare them on the same footing. Note that, since the advice string has a length accessible to both $\mathcal{M}_1$ and $\mathcal{M}_2$, we cannot in general "preserve" the computation of one and let it be read by another, as in the usual relativization model.

Our main results in this paper show that both parity polynomial-time $\oplus$P and the polynomial-time hierarchy PH collapse to P for some exponential-size advice strings. The collapse between NP and P immediately follows from the latter one. More precisely, for P and $\oplus$P (respectively, P and PH), we show some set $\{s_n\}_{n\geq 0}$ of advice strings of length $2^{(1+\epsilon)n}$, i.e., each s_n of length $2^{(1+\epsilon)n}$, with which $\oplus$P (resp., PH) collapses to P. We use decision tree lower bounds for constant depth circuits [Yao85, Cai86, Hås86] and the algebraic machinery of Razborov and Smolensky [Raz87, Smo87]. It is open whether one can collapse PSPACE and P with some set of advice strings of some exponential size.

Results of this type are mainly of value in delineating the limit of our ability to settle some outstanding questions on complexity classes. Our model of random access to advice strings provides a more stringent model than the usual relativization model, and therefore it provides a more stringent perspective on the "provability" question. More specifically, the stringent collapsing results are stronger indications to the limit of our ability to separate these classes. It is interesting that a collapse of PSPACE to P under random access to advice is left open. We do not know whether this indicates the P vs. PSPACE question has a different nature from the other outstanding open questions. But we recall here the following result of Kozen [Ko78]: If PSPACE $\neq$ P, then there exists a proof of this fact by diagonalization.

2 Random Access to Advice Strings

Recall the definition of $\mathcal{C}$/poly by Karp and Lipton [KL80]. In order to define "polynomially bounded nonuniformity" for discussing complexity classes such as polynomial-size circuits, Karp and Lipton introduced a computation model where machines can make use of some auxiliary information called an "advice". More specifically, we may assume some "advice function" h mapping input size n to some binary "advice string" of length polynomially bounded in n. For a given problem instance of size n, we may assume that an advice string $h(n)$ is *somehow* given to a machine solving the problem. Here the computability of h or the way to compute h is ignored; that is, the computability or the

complexity of h are not accounted for in $\mathcal{C}$. On the other hand, the length of $h(n)$ must be polynomially bounded in n, and the same advice $h(n)$ must be used for all problem instances of size n.

We generalize their model by allowing the underlying machines to have random access to an advice string. Then we may consider advice functions whose outputs are not polynomially bounded. Let us fix any "length function" ℓ from $\mathbf{N}$ to $\mathbf{N}$. A function $s : n \mapsto \{0,1\}^{\ell(n)}$ is called an *advice function of size* $\ell(n)$. Given any advice function s of size $\ell(n)$, we say a language L is in the class $\mathcal{C}/s$ *via random access to advice* if there is some machine $\mathcal{M}$ representing the class $\mathcal{C}$, such that $x \in L$ iff $\mathcal{M}(x; s(|x|))$ accepts, where we denote the computation $\mathcal{M}$ on x with random access to $s(|x|)$ by $\mathcal{M}(x; s(|x|))$. (The notion of *random access* is the usual one: A machine $\mathcal{M}$ can write down an index to a bit of $s(|x|)$ on a special tape and then it gets that bit in unit time.) We denote this language as $L(\mathcal{M}; s)$. Clearly, if a time bound being considered is larger than the advice size, then the random accessibility is not necessary, and this notion is the same as the one by Karp and Lipton. (In the following, all complexity bounds and length functions are time and space constructible as appropriate. Furthermore, we assume that $\log(\ell(n))$ is polynomially bounded, which is reasonable for comparing with polynomial-time classes even if we allow random access to an advice string.)

Let s be any advice function, and let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two complexity classes represented by query TMs. We say *collapsing occurs w.r.t. s* (write as $\mathcal{C}_1/s \subseteq \mathcal{C}_2/s$) if for every machine $\mathcal{M}_1$ representing $\mathcal{C}_1$, there is a machine $\mathcal{M}_2$ representing $\mathcal{C}_2$, such that $L(\mathcal{M}_1; s) = L(\mathcal{M}_2; s)$. We say *two classes are equal w.r.t. s* (write as $\mathcal{C}_1/s = \mathcal{C}_2/s$) if both $\mathcal{C}_1/s \subseteq \mathcal{C}_2/s$ and $\mathcal{C}_2/s \subseteq \mathcal{C}_1/s$. On the other hand, we say *separation occurs w.r.t. s* (write as $\mathcal{C}_1/s \not\subseteq \mathcal{C}_2/s$) if there exists some machine $\mathcal{M}_1$ representing $\mathcal{C}_1$ such that $L(\mathcal{M}_1; s) \neq L(\mathcal{M}_2; s)$ for any machine $\mathcal{M}_2$ representing $\mathcal{C}_2$.

Then our main results can be stated as follows. (For both results, size bound $\ell(n)$ can be improved slightly. Our proofs in fact require that $\ell(n)/2^n$ is superpolynomial.)

Theorem 1 *For any length bound $\ell(n) \geq 2^{(1+\delta)n}$, where $\delta > 0$ is any positive constant, there exists an advice function s of advice size $\ell(n)$ such that $\oplus\mathrm{P}/s = \mathrm{P}/s$.*
Remark. *The same result is provable for the relationship between* P *and* Mod_p *class, for any prime p.*

Theorem 2 *For any length bound $\ell(n) \geq 2^{(1+\delta)n}$, where $\delta > 0$ is any positive constant, there exists an advice function s of advice size $\ell(n)$ such that* $\mathrm{PH}/s = \mathrm{P}/s$.

The motivation of this model as a relativization model and the relations to the conventional relativization model will be explained in the next section. Here we remark on the position of our model and results in the study of nonuniform complexity classes.

First note that collapsing results with exponential-size nonuniform advice strings would be interesting only if the same advice is given to both classes. For example, we trivially have some advice function of advice size 2^n such that $\mathrm{NP} \subseteq \mathrm{P}/s$ holds, because

the size 2^n is large enough to record all answers on inputs of length n on one advice string of size 2^n.

Second, since our nonuniform notion is a generalization of the standard nonuniform model of Karp and Lipton, there are immediate implications for our nonuniform comparison from some of the results for the standard nonuniform model. For example, it has been known [Kan82] that PH $\not\subseteq$ P/$p(n)$ for any fixed polynomial $p(n)$. Then the following fact is immediate from this result, because, for any advice function s, we clearly have PH $\subseteq$ PH/s. This fact justifies the consideration of at least super-polynomial advice size for obtaining a nonuniform collapsing result for P and PH.

Proposition 3 *For any polynomially bounded advice $\ell(n)$, there is no advice function s of advice size $\ell(n)$ for which* PH/s $\subseteq$ P/s.

3 Stringent Relativization: Model and Implications

Let us discuss the meaning of our relativization model. We begin with what we feel to be unsatisfactory with collapsing results in the conventional relativizations.

Again consider the P vs. NP relation, and the well-known proof of the relativized P = NP result. Consider any complete set C for PSPACE. Then one can argue as follows to show $\mathrm{P}^C = \mathrm{NP}^C$: From the PSPACE-completeness of C, it follows that PSPACE $\subseteq \mathrm{P}^C$. On the other hand, $\mathrm{P}^C \subseteq \mathrm{NP}^C \subseteq \mathrm{PSPACE}^C = \mathrm{PSPACE}$. Hence, we conclude that $\mathrm{P}^C = \mathrm{NP}^C$.

This proof is valid, but in order to see how queries are used in this collapsing argument, we fix some specific PSPACE-complete set and any NP query machine $\mathcal{Q}_0$, and examine the simulation of $\mathcal{Q}_0$ by some P query machine $\mathcal{Q}_1$ relativized to this complete set. For our complete set, consider the following canonical complete set K.

$$K \;=\; \{\, (\mathcal{M}, x, 0^s) \,:\, \mathcal{M} \text{ accepts } x \text{ by using } s \text{ space} \,\}.$$

Where $\mathcal{M}$, x, and 0^s are respectively the description of a deterministic Turing machine (with no oracle access), a string in $\{0,1\}^*$, and a sequence of s 0's, where s is any positive integer. We assume that $(\mathcal{M}, x, 0^s)$ is encoded as a string in $\{0,1\}^*$ in some reasonable way.

Note that the computation of $\mathcal{Q}_0^K$ can be simulated by some PSPACE machine $\mathcal{M}_0$. Let $s_0(\cdot)$ be a polynomial space bound for $\mathcal{M}_0$. Then the construction of a P query machine $\mathcal{Q}_1$ simulating $\mathcal{Q}_0^K$ is easy. On a given input x of length n, $\mathcal{Q}_1$ simply asks the query $(\mathcal{M}_0, x, 0^{s_0(n)})$ to K, and then outputs its answer. It is easy to see that $\mathcal{Q}_1^K$ simulates $\mathcal{Q}_0^K$ correctly. In this way, for a given NP query machine, we can define a P query machine simulating the machine relative to K, which proves that $\mathrm{P}^K = \mathrm{NP}^K$.

Notice here that a query made by $\mathcal{Q}_1$ is longer than queries asked in the simulated $\mathcal{Q}_0^K$ computation. If on an input x of length n, $\mathcal{Q}_0^K(x)$ makes a query of length $\ell_0(n)$

to K, the only way we know how to design a simulating PSPACE machine $\mathcal{M}_0$ is to have space bound $s_0(n) \geq \ell_0(n)$. Then the entry $(\mathcal{M}_0, x, 0^{s_0(n)})$ has length greater than $\ell_0(n)$. Therefore the query made by $\mathcal{Q}_1$ is longer than those asked by $\mathcal{Q}_0$. This situation remains the same if we used any other PSPACE-complete set, such as QBF. In the case of QBF this increase in the length of queried strings occurs when we translate a PSPACE computation, which is an NP computation at a certain length n relativized to QBF, to an instance of QBF at a longer length. Thus the standard proof of relativized collapse of P and NP all have this property that a simulating machine needs to ask queries longer than those asked by a simulated machine.

It raises the question whether it could still be the case that $\mathrm{P}^K = \mathrm{NP}^K$ can be proved by some more sophisticated argument where a simulating machine does not make queries longer than those queried by a simulated machine.

Similar to the case of relativized collapse of P and NP, most of the known relativized collapsing results are proved by using such asymmetric access to an oracle. One can argue that this asymmetry is within a polynomial factor; but it nonetheless denies access to certain segments of the oracle to the simulated machine while affords such access to the simulating machine. We think that a better comparison can be made for the underlying computational powers if we can introduce a relativization model where such asymmetry is avoided naturally. If a relativized collapse result is supposed to provide evidence to the difficulty of proving unrelativized separation result, then it is natural that we examine whether or not this asymmetry of oracle access actually occurs in the few separation results that are known. However, if one actually relativizes the proofs of the few separation results such as the hierarchy theorems, one observes that this asymmetry is *not* present in the relativized proof.

Cast in this light, then, what would be a reasonable relativization model? To discuss this point in more detail, consider the relationship between two complexity classes NTIME[n^2] and DTIME[n^3], classes of decision problems solvable, respectively, by nondeterministic $\mathcal{O}(n^2)$-time machines and by deterministic $\mathcal{O}(n^3)$-time machines. We conjecture that NTIME[n^2] $\not\subseteq$ DTIME[n^3]; yet, it seems difficult to prove. To justify the difficulty of proving this conjecture, we need a collapsing "relativized" result showing NTIME[n^2] $\subseteq$ DTIME[n^3]. That is, we would like to prove the collapse in some "parallel world" that is defined by some "non-standard model" of computation.

A non-standard computation model can be defined by extending primitives for computation, and one would naturally think of using some black box functions as new primitives. That is, we assume that such functions are computable at unit computational cost. Without loss of generality, we assume that each of the black box functions is a Boolean predicate defined on $\{0,1\}^k$ for some k, i.e., a function mapping $\{0,1\}^k$ to $\{0,1\}$. Here based on the way to add black box functions we may consider the following three types of non-standard computation model.

(Relativization Type 1)

First consider a model obtained by adding some finite set of black box functions. But this does not make any essential difference, because all these functions are finitely representable, and they can be embedded as a finite table in machines. Thus, this relativization model is equivalent to the standard one.

Then to obtain a different model, we need to allow an infinite set of black box functions.

(Relativization Type 2)
We assume that machines can make use of a family of Boolean functions $\{Q_\ell\}_{\ell\geq 0}$, where each Q_ℓ computes some (a priori fixed) predicate on $\{0,1\}^\ell$. This is essentially the same as the standard relativization model. But note the subtle but important point here: there are respective limitations of how to use these Boolean functions due to machine's resource bound such as time bound. Even though we assume that the computation of Q_ℓ takes unit cost, certain time and space are at least necessary for preparing an input to the function. For example, a machine whose running time is at most, say, $3n^2+4n+21$, cannot use any Q_ℓ with $\ell > 3n^2+4n+21$. Thus, the set of primitives is not the same for $O(n^2)$-time and for $O(n^3)$-time machines. In this sense, we may claim that relativized classes NTIME$[n^2]$ and DTIME$[n^3]$ are compared on different non-standard computation models.

(Relativization Type 3)
We consider an intermediate model between (Type 1) and (Type 2). We will have infinitely many Boolean functions, but we will avoid comparisons on an unequal footing. Our requirement is as follows: (i) Specify a family of infinite number of black box Boolean functions $\{Q_\ell\}_{\ell\geq 0}$, and (ii) use the same finite subset of primitives for simulating and simulated machines at any given input length n. Then one natural approach is to bound ℓ for predicates Q_ℓ by some fixed function $\ell(n)$ on input size n. This is the model for our $\ell(n)$-stringent relativization. Note that the query length bound $\ell(n)$ must be smaller than the size bounds of both simulating and simulated machines. In the above example, $\ell(n)$ should be at most n^2. In fact, for "polynomially bounded" complexity classes, we propose to use $\ell(n) = cn$ for the query length bound.

Looking back over existing relativized results in the literature (e.g., [BDG89, DK00]), we notice that almost all collapsing results (except the one mentioned below, i.e., Proposition 5) are proved by an argument similar to the above P vs. NP case, where simulating machines make oracle accesses at a length beyond those queried by the simulated machines. We propose to reconsider these collapsing results under polynomially stringent relativization.

Book, Long, and Selman [BLS84] introduced the notion of "positive relativization". Their motivation has similarities to ours. However in their approach they restrict the total number of oracle queries and therefore their results have a different flavor from our results. For instance, they denote by $\mathrm{NP}_{\mathrm{b}}^X$ the class of languages accepted by NP query machines which make at most a polynomial number of queries to the oracle X. Note that this restriction applies to the total number of queries over the entire computational tree. By

contrast, in our model of stringent relativization, NP query machines are allowed to make exponentially many queries over the computational tree. We believe that the ability to invoke "allowed primitives" an exponential number of times over the entire computational tree is essential to the nature of NP computation. It is the type of "allowed primitives" we provide at length n that we keep equal to both sides.

3.1 Relation with Existing Relativized Results

Let us consider our results and proofs as related to existing relativized results.

First it should be noted that most relativized *separation* results are proved in a stringent way; that is, if not already so, the proofs of such results can be easily modified to give the same separation with respect to some common advice function of some exponential (or super-polynomial) advice size. For example, we can prove the following relation.

Proposition 4 *For any super-polynomial length bound $\ell(n)$, there exists an advice function s of advice size $\ell(n)$ such that* $\mathrm{NP}/s \not\subseteq \mathrm{P}/s$.

While most relativized collapsing results are proved in a non-stringent way, there are some relativized collapsing proofs (essentially one, and the others are its variations) in the literature that also yield non-uniform collapsing results in our context. A typical example of such results is as follows.

Proposition 5 *For any positive constant $\delta > 0$, and for any length bound $\ell(n) \geq 2^{(2+\delta)n}$, there exists an advice function s of advice size $\ell(n)$ such that* $\mathrm{NP}/s \subseteq (\mathrm{P}/\mathrm{poly})/s$.

This follows from the proof [Wil83] showing a (standard) oracle A such that $\mathrm{NP}^A \subseteq \mathrm{P}^A/\mathrm{poly}$. (We note that this result is superceded by our Theorem 2.) Here we recall its idea. Suppose that the oracle set A has been partially defined up to length $< (2+\delta)n$, and that we are now in the stage for simulating some NP query machine $\mathcal{Q}_0$ on $\{0,1\}^n$, i.e., all inputs of length n. For any input x of length n, consider the execution of $\mathcal{Q}_0$ on x relative to (so far constructed) A. We check whether there is any extension of A so that $\mathcal{Q}_0^A$ accepts x. If such an extension exists, then choose some accepting path of $\mathcal{Q}_0^A(x)$, and fix the membership of queries asked on the path, thereby "freezing" the result of $\mathcal{Q}_0^A$ on x. (On the other hand, if no such extension exists, we do nothing on x; x is anyway rejected no matter how A is extended.) Note that the number of elements fixed for each input x is at most $p(n)$ for some polynomial p; thus, for freezing the results of $\mathcal{Q}_0^A$ on $\{0,1\}^n$, we only need to fix the membership of at most $p(n)2^n < 2^{(1+\delta)n}$ elements of $\{0,1\}^{(2+\delta)n}$. Thus there must be some block of size 2^n in $\{0,1\}^{(2+\delta)n}$ that is not touched by this process; we use this block to encode the results of $\mathcal{Q}_0^A$ on $\{0,1\}^n$. Then for every input $x \in \{0,1\}^n$, one can get the answer of $\mathcal{Q}_0^A(x)$ by asking a string corresponding to x in the block. The block can be specified by a string of length $(1+\delta)n$, and it can be given as a polynomial-size advice. This is the idea of simulating NP query machines by some P query machine with polynomial-size advice strings.

Note that to simulate computations at length n inputs, the oracle A defined on $\{0,1\}^{(2+\delta)n}$ is used. This means, in our framework, that the simulation can be done with some $2^{(2+\delta)n}$-size advice function.

By a similar proof technique, we can in fact prove $\mathrm{NEXP}^B \subseteq \mathrm{P}^B/\mathrm{poly}$ in the standard relativization model [He86]. This is because for any NEXP query machine $\mathcal{Q}_0$ with running time $2^{p(n)}$, we can freeze its results on $\{0,1\}^n$ by fixing at most $2^{n+p(n)}$ strings. Then in a segment, say, $\{0,1\}^{3p(n)}$, we can again find a block of 2^n elements that are not touched for the purpose of freezing the results of $\mathcal{Q}_0$ on $\{0,1\}^n$. Thus, giving the location of this block as an advice, a query machine can simulate $\mathcal{Q}_0$ on $x \in \{0,1\}^n$ by asking one query of length $3p(n)$ to the oracle. On the other hand, this argument does not work in our context because the advice size $2^{3p(n)}$ cannot be bounded by any single exponential function. On length n inputs, the simulating machine needs to use the segment of the oracle (e.g., $\{0,1\}^{3p(n)}$) that is determined by, and greater than, the time bound for the simulated machine. Our stringent relativization requires all machines to use the same segment of an oracle on length n inputs.

It should be also remarked that, in stringent relativization, a higher collapse does not immediately follow from a lower collapse. For example, the relatively simple proof of $\mathrm{NP}/s \subseteq (\mathrm{P}/\mathrm{poly})/s$ for some advice s of some exponential advice size bound does not give a proof of $\mathrm{PH}/s' \subseteq (\mathrm{P}/\mathrm{poly})/s'$ for some s' with some exponential advice size bound. To appreciate this difficulty, consider Σ_2^{p}/s. Note that for a Σ_2^{p} computation on an input of length n, we may very well have to deal with queries y by the base level NP machine, where y itself encodes an NP computation relativized to s. However the length of the query y may be more than n. (Nevertheless, it is true that $\mathrm{PH}/s' \subseteq (\mathrm{P}/\mathrm{poly})/s'$, for some s' with some exponential advice size bound, and it follows from our Main theorem 2.)

4 Class P vs. Class $\oplus$P

In this section we consider the relation between P and $\oplus$P and prove Theorem 1. The proof techniques will be extended in the next section to prove Theorem 2.

To simplify the presentation we will consider only $\log(\ell(n)) = (1+\delta)n$. It is easy to extend the following proof to any $\ell(n)$ with $\log(\ell(n)) \geq (1+\delta)n$.

Proof of Theorem 1. Let $\mathcal{M}_1, \mathcal{M}_2, \ldots$ be a standard enumeration of all $\oplus$P machines. Our goal is to construct an advice function s with $s(n) \in \{0,1\}^{\ell(n)}$, with which the computation of every $\mathcal{M}_i(x; s(|x|))$ can be simulated by some P computation with the common advice $s(|x|)$. Let us fix any $\oplus$P machine $\mathcal{M}$ and any input length n, and discuss how to design $s(n)$ so that some P machine can simulate $\mathcal{M}$ on $\{0,1\}^n$ with advice $s(n)$. It would be easy later to "paste" together a single $s(n)$ for all machines to be considered at length n. (Only finitely many need to be dealt with at any finite length n. This uses standard relativization techniques, and therefore we will omit this detail.)

Let $m = n^{O(1)}$ be the maximum number of accesses to the advice string made by $\mathcal{M}$

on any nondeterministic path on any input of length n. We assume that n is sufficiently large.

Let $L = 2^{(1+\delta)n}$. We will consider the advice string $s(n)$ of length L as being indexed by a binary string of length $I = (1+\delta)n$.

For any $x \in \{0,1\}^n$, we will define S_x to be a subset of $\{0,1\}^I$ of size $\approx nm$. Furthermore we want $\{S_x\}_{x\in\{0,1\}^n}$ to be a family of pair-wise disjoint subsets of $\{0,1\}^I$. We intend to use the bits in S_x to code the result of a $\oplus$P computation $\mathcal{M}(x; s(n))$, where $s(n)$ is the advice string of which S_x is a part. However, unlike in some standard relativization proofs where a single bit in S_x will be used to code this computation result, we will use all the bits collectively to code this result. In fact we will use the NAND function over all the bits in S_x to do so.

Now we define $s = \lceil \log nm \rceil$, and let

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

Each string in $\bigcup_{x\in\{0,1\}^n} S_x$ is the index of a bit in $s(n)$. We assign Boolean variables for these bits, and denote the set of these Boolean variables as Z. Let $M = |Z|$; note that $M \le 2nm2^n \ll 2^I$. Let us name the Boolean variables in Z as $z_1, z_2, \ldots, z_M$.

Assign arbitrarily the bit values for all bits in $s(n)$ other than those in Z. Then, for any input $x \in \{0,1\}^n$, $\mathcal{M}(x; s(n))$ is completely determined by the values of z_i. That is, $\mathcal{M}(x; s(n))$ is a function on Boolean variables $z_1, \ldots, z_M$. Furthermore, since $\mathcal{M}(x; s(n))$ is a parity computation asking at most m queries on each nondeterministic path, we may consider $\mathcal{M}(x; s(n))$ as a parity (or its negation) of at most $\sum_{i=0}^{m} 2^i \binom{M}{i}$ many conjunctions, each of which has at most m literals from $z_1, \ldots, z_M$. Thus, $\mathcal{M}(x; s(n))$ is expressed by a polynomial $f_x(z_1, \ldots, z_M)$ of degree $\le m$ with integer coefficients mod 2. As a Boolean function we may assume that $f_x(z_1, \ldots, z_M)$ is multilinear.

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 $N = 2^n$).

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

We recognize that each right hand side represents the NAND function over the subset S_{x_i}. If this assignment is feasible (i.e., the advice string $s(n)$ is constructed satisfying $(*1)$), then for any $x \in \{0,1\}^n$, one simply needs to check the membership of elements of S_x; $\mathcal{M}(x; s(n))$ can then be computed as $1 - \prod_{z_j \in S_x} z_j$ in polynomial time.

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 $x \in \{0,1\}^n$ 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 Fact 1 stated below that modulo the ideal $J = (z_1^2 - z_1, \ldots, z_M^2 - z_M)$, the left hand 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 $\mathbf{Z}_2[z_1, \ldots, z_M]/J$, 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 $(N-1)2^s + m$. A contradiction. □

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

5 Class P vs. Class PH

We now show that there exists an advice function of length $2^{(1+\delta)n}$, such that the class PH collapses to P with random access to the advice strings given by the advice function. It can be easily extended to any length $\ell(n) \ge 2^{(1+\delta)n}$. However for simplicity of presentation we will assume $\ell(n) = 2^{(1+\delta)n}$ in what follows. We prove the following result for a fixed level Σ_d^{p}; the construction for the advice string for PH follows since PH is a countable union of classes Σ_d^{p}, $d \ge 0$.

Theorem 6 *For any constant $d \ge 0$, and constant $\delta > 0$, let $\ell(n) = 2^{(1+\delta)n}$; then there exists an advice function s of advice size $\ell(n)$ such that $\Sigma_d^{\mathrm{p}}/s \;=\; \mathrm{P}/s$.*

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

Some notions and notations first. For any Boolean function f over variables $x_1, \ldots, x_n$, by a *random restriction* ρ, we mean a 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. We can also think of ρ as a partial function from $\{x_1, \ldots, x_n\}$ to $\{0, 1\}$. Let $f \mid_\rho$ denote the function on the unassigned variables obtained from f by this random restriction.

The decision tree complexity of a Boolean function f, denoted by $\mathrm{DC}(f)$, is the smallest depth of a Boolean decision tree computing the function. It can be shown easily that if $\mathrm{DC}(f) \le t$, 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 $\mathrm{DC}(f) \leq t$, then f can be expressed as a polynomial on the variables, with integer coefficients and with degree at most t. In fact this polynomial always evaluates to 0 or 1, for any 0-1 assignments to its variables.

Superpolynomial lower bounds for constant depth circuits were first proved by Furst, Saxe and Sipser [FSS81], and by Ajtai [Ajt83]. Exponential lower bounds of the form $2^{n^{\Omega(1/d)}}$ for depth d circuits were first proved by Yao [Yao85] in a breakthrough result. Yao's bound was further improved by Håstad [Hås86] to $2^{\frac{1}{10}n^{\frac{1}{d-1}}}$, and his proof has become the standard proof. Independently, Yao's work was improved upon in another direction. Cai [Cai86] investigated whether constant depth circuits of size $2^{n^{\Omega(1/d)}}$ must err on an asymptotically 50 % of inputs against parity. To attack this problem, the decision tree point of view was first introduced in [Cai86]. This approach in terms of inapproximability has been found most fruitful in the beautiful work of Nisan and Wigderson [Nis91, NW94] on pseudorandom generators.

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

Lemma 7 *For any depth $d+1$ Boolean circuit C on $z_1, \ldots, z_L$, with bottom fan-in at most t,*

$$\Pr[\ \mathrm{DC}(C\mid_\rho) \geq t\ ] \ \leq\ \frac{\mathrm{size}(C)}{2^t},$$

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

We now describe our construction. Fix any Σ_d^{p} machine $\mathcal{M}$ and any sufficiently large input length n. We want to construct $s(n)$, such that the computation $\mathcal{M}(x; s(n))$ can be simulated by a polynomial-time deterministic machine, for all x of length n. Constructing the advice function s for the simulation of *all* Σ_d^{p} machines can be done as before for $\oplus\mathrm{P}$ and is omitted here.

Thus, from now on, we are concerned with the simulation of $\mathcal{M}$ on 2^n inputs of length n. Let m be an integer bounding $\mathcal{M}$'s running time on inputs of length n, where $m = O(n^k)$ for some $k \geq 0$. Let $I = (1+\delta)n$ and $L = 2^I$. Let $z_1, z_2, \ldots, z_L$ be Boolean variables denoting the bits in $s(n)$. Let Z denote the set of all Boolean variables $z_1, \ldots, z_L$. With a slight abuse of notation we will also let Z denote a set of corresponding indeterminants.

For any input string $x \in \{0,1\}^n$, consider the computation of $\mathcal{M}(x; s(n))$. The computation $\mathcal{M}(x; s(n))$ is a function from the Boolean variables $z_1, \ldots, z_L$ to $\{0,1\}$. Furthermore, since $\mathcal{M}$ is a Σ_d^{p} machine, by a standard interpretation (see [FSS81]) of the Σ_d^{p} query computation, we may regard $\mathcal{M}(x; s(n))$ 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 7, with high probability the circuit is reduced to

small depth decision trees with depth $t = 2m$. In fact, by choosing p_0 appropriately, we can even show that with high probability, a random restriction converts *all* circuits for all 2^n input strings to depth t decision trees.

Then these small depth decision trees can be expressed by low degree (i.e., degree $2m$) polynomials with integer coefficients. That is, after the random restriction, each computation $\mathcal{M}(x; s(n))$ is expressed as a degree $2m$ polynomial p_x. We have arrived at a similar situation to the parity computation. We will use a similar technique to attack this. However the exact approach in the $\oplus$P case does not work.

In the $\oplus$P case the function encoded is essentially the AND function $\bigwedge z_j$. This will not survive the random restriction. Instead we will try to encode the *parity* on a suitable subset, one for each x. Our encoding is implemented as follows. For each $x \in \{0,1\}^n$, we define a segment $S_x \subset \{0,1\}^I$ of enough size, roughly speaking, $20m/p_0$, which is polynomial in n. These segments are chosen so that the family $\{S_x\}_{x\in\{0,1\}^n}$ is pairwise 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 $\mathcal{M}(x; s(n))$. 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.

Thus, there exists a partial assignment satisfying the following.

(a) Each computation $\mathcal{M}(x; s(n))$ is reduced to a decision tree T_x of depth at most $2m$.
(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 $Z - Z_0$, we now assign all such variables to 0. Then as explained above, the result of each computation of $\mathcal{M}(x; s(n))$ is expressed as a degree $2m$ polynomial $p_x(z_1, \ldots, z_M)$ over the integers $\mathbf{Z}$. For each x, we try to equate $p_x(z_1, \ldots, z_M)$ to the parity of S_x, i.e., $\oplus_{z_i\in S_x} z_i$. (Note that S_x contains variables not in $Z_0 = \{z_1, \ldots, z_M\}$ whose values are already fixed. By the expression $\oplus_{z_i\in S_x} z_i$ we mean the parity of all variables in S_x including such variables.) In other words, we wish to choose an assignment to $z_1, \ldots, z_M$ so that the following system of equations ($*2$) holds for $\{0,1\}^n = \{x_1, \ldots, x_N\}$, where $N = 2^n$.

$$(*2)\quad \begin{cases} p_{x_1}(z_1, \ldots, z_M) &= \oplus_{z_j\in S_{x_1}} z_j, \\ & \vdots \\ p_{x_N}(z_1, \ldots, z_M) &= \oplus_{z_j\in S_{x_N}} z_j. \end{cases}$$

The proof will be completed by considering the dimension of a certain finite dimensional algebra over the finite field $\mathbf{Z}_3$, and show that it is indeed possible to find such an assignment.

Now we give the formal proof.

We focus on the simulation of some Σ_d^{p} machine $\mathcal{M}(x; s(n))$ on N $(= 2^n)$ inputs of length n for sufficiently large n. Let $m = O(n^k)$ be an integer bounding $\mathcal{M}$'s running time on length n inputs, and let $I = (1+\delta)n$ and $L = 2^I$. We regard the computation of $\mathcal{M}(x; s(n))$ as a function over Boolean variables $z_1, \ldots, z_L$, where each z_i is the Boolean variable for a bit in $s(n)$. Furthermore, we may consider $\mathcal{M}(x; s(n))$ as a circuit C_x of depth $\leq d+1$, size $\leq m2^m$, and bottom fan-in $\leq m$.

As explained above, we consider a random restriction to the variables $z_1, \ldots, z_L$, 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.

We want some restriction ρ, such that it satisfies 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 7 and Chernoff's bound (see, e.g., Corollary A.1.14 of [AS00]), it is easy to show the following claim:

Claim 1 *Under our choice of parameters, the probability that a random restriction ρ satisfies both (a) and (b) is not zero.*

Hence, there exists some restriction satisfying both (a) and (b).

Consider one such restriction ρ_0 satisfying both (a) and (b). We define $s(n)$ based on this ρ_0; that is, we will assign a bit in $s(n)$ to 0 or 1 according to ρ_0. We will assign those variable assigned $*$ by ρ_0 later. Let Z_* be the set of variables assigned $*$ by ρ_0. From 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, over 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 condition (a), the computation $\mathcal{M}(x; s(n))$ for each $x \in \{0,1\}^n$ 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 is T_0 for $z_i = 0$, and its right subtree is T_1 for $z_i = 1$. 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 a 0,1-assignment to variables in Z_0 satisfying $(*2)$ above. We complete the partial assignment on Z to a full assignment by this 0,1-assignment on Z_0, and define $s(n)$ accordingly. Then one can compute the value of $\mathcal{M}(x; s(n))$, for each $x \in \{0,1\}^n$, by asking queries on all the bits indexed in S_x and taking the parity of the answers. Since the size of S_x is polynomially bounded in n, this is a P computation with random access to $s(n)$.

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 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 by looking at the above decomposition $p = (1 - z_i)p_0 + z_i p_1$. Furthermore, this property is invariant even if the polynomials are evaluated modulo q, for any prime q. Thus, we may consider these polynomials under 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 $\mathbf{Z}_3$.

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

Where each $\alpha_x \in \{-1, +1\}$ denotes the product of all the variables $z_i \in S_x - Z_0$ which had already been set to ± 1.

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

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

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

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

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.

This completes the proof of Theorem 6, and hence Theorem 2. With some more work one can show

Theorem 8 *For any prime p, and for any length bound $\ell(n) \ge 2^{(1+\delta)n}$, where $\delta > 0$ is any positive constant, there exists an advice function s of advice size $\ell(n)$ such that* $\mathrm{Mod}_p^{\mathrm{PH}}/s \;=\; \mathrm{P}/s$.

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.

[BGS75] T. Baker, J. Gill, and R. Solovay, Relativizatons of the P =? NP question, *SIAM J. Comput.* 4(4), 431–442, 1975.

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

[BLS84] R. Book, T. Long, and A. Selman, Quantitative relativization of complexity classes, *SIAM J. Comput.* 13, 461–487, 1984.

[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.* (STOC'89), 21–29, 1986. (The final version appeared in *J. Comp. Syst. Sci.* 38(1), 68–85, 1989.)

[CW03] J-Y. Cai and O. Watanabe, On proving circuit lower bounds against the polynomial-time hierarchy: positive and negative results, *Proc. 9th International Computing and Combinatorics Conference* (COCOON'03), LNCS 2697, 202–211, 2003. (The journal version appeared in *SIAM J. Comput.* 33(4), 989–1009, 2004.)

[CW04] J-Y. Cai and O. Watanabe, Relativized collapsing between BPP and PH under stringent oracle access, *Inform. Process. Lett.* 90, 147–154, 2004.

[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.

[Hås86] 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.

[He86] H. Heller, On relativized exponential and probabilistic complexity classes, *Information and Control* 71(3), 231–243, 1986.

[Kan82] R. Kannan, Circuit-size lower bounds and non-reducibility to sparse sets, *Information and Control* 55, 40–56, 1982.

[KL80] R. Karp and R. Lipton, Some connections between nonuniform and uniform complexity classes, in *Proc. 12th ACM Symposium on Theory of Computing* (STOC'80), ACM, 302–309, 1980. (An extended version appeared as: Turing machines that take advice, in *L'Enseignement Mathématique* (2nd series) 28, 191–209, 1982.)

[Ko78] D. Kozen, Indexing of subrecursive classes, in *Proc. 10th ACM Symposium on Theory of Computing* (STOC'78), ACM, 287–295, 1978. (The final version appeared in *Theoretical Computer Science* 11, 277–301, 1980.)

[Nis91] N. Nisan, Pseudorandom bits for constant depth circuits, *Combinatorica* 11(1), 63–70, 1991.

[NW94] N. Nisan and A. Wigderson, Hardness vs randomness, *J. Comput. Syst. Sci.* 49, 149–167, 1994.

[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.

[Wil83] C.B. Wilson, Relativized circuit complexity, in *Proc. 24th IEEE Symposium on Foundations of Computer Science* (FOCS'83), IEEE, 329–334, 1983.

[Yao85] A.C. Yao, Separating the polynomial-time hierarchy by oracles, in *Proc. 26th IEEE Symposium on Foundations of Computer Science* (FOCS'85), IEEE, 1-10, 1985.