ISSN 1342-2812

Research Reports on Mathematical and Computing Sciences

On Proving Circuit Lower Bounds Against the Polynomial-time Hierachy: Positive and Negative Results

Jin-Yi Cai and Osamu Watanabe

Aug. 2008, C–256

Department of
Mathematical and
Computing Sciences
Tokyo Institute of Technology

SERIES C: Computer Science

Technical Reports on Mathematical and Computing Sciences: TR-C-256
(Revised version of TR-C-161)
title: On Proving Circuit Lower Bounds Against the Polynomial-time Hierarchy: Positive and Negative Results
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-9634665.
2. Supported in part by the Minstry for Education, Grant-in-Aid for Scientific Research (C), 2001.

Abstract. We consider the problem of proving circuit lower bounds against the polynomial-time hierarchy. We give both positive and negative results. For the positive side, for any fixed integer $k > 0$, we give an explicit $\Sigma_2^{\rm p}$ language, acceptable by a $\Sigma_2^{\rm p}$-machine with running time $O(n^{k^2+k})$, that requires circuit size $> n^k$. This provides a constructive version of an existence theorem of Kannan [Kan82]. Our main theorem is on the negative side. We give evidence that it is infeasible to give relativizable proofs that any single language in the polynomial-time hierarchy requires super polynomial circuit size. Our proof techniques are based on the decision tree version of the Switching Lemma for constant depth circuits and Nisan-Wigderson pseudorandom generator.

1. Introduction

It is a most basic open problem in Theoretical Computer Science to give circuit lower bounds for various complexity classes. The class P has polynomial size circuits. It is also widely believed that NP does not share this property, i.e., that some specific set such as SAT in NP requires super polynomial circuit size. While this remains the most concrete approach to the NP vs. P problem, we can't even prove this for any fixed $k > 1$ that any set $L \in$ NP requires circuit size $> n^k$.

If we relax the restriction from NP to the second level of the Polynomial-time Hierarchy $\Sigma_2^{\rm p}$, R. Kannan [Kan82] did prove that for any fixed polynomial n^k, there is some set L in $\Sigma_2^{\rm p}$ which requires circuit size $> n^k$. Kannan in fact proved the existence theorem for some set in $\Sigma_2^{\rm p} \cap \Pi_2^{\rm p}$. This result has been improved by Köbler and Watanabe [KW98] who showed, based on the technique developed in [BCGKT], that such a set exists in $\mathrm{ZPP}^{\mathrm{NP}}$. More recently, the work in [Cai01] implies that a yet lower class $\mathrm{S}_2^{\rm p}$ contains such a set. (See [BFT98, MVW99] for related topics.)

However, Kannan's proof for $\Sigma_2^{\rm p}$, and all the subsequent improvements mentioned above, are not "constructive" in the sense that it does not identify a single $\Sigma_2^{\rm p}$ machine whose language requires circuit size $> n^k$. At the top level, all these proofs are of the following type: *Either* SAT does not have n^k size circuit, in which case we are done, *or* SAT has n^k size circuit, then we can define some other set, which by the existence of the hypothetical circuit for SAT can be shown in Σ_2^{p}, and it requires circuit size $> n^k$. Constructively, Kannan gave a set in $\Sigma_4^{\mathrm{p}} \cap \Pi_4^{\mathrm{p}}$. In [MVW99] a set in Δ_3^{p} was constructively given. We improve[1] this to Σ_2^{p}.

Theorem 1 *For any integer $k > 0$, we can construct a Σ_2^{p}-machine with $O(n^{k^2} \log^{k+1} n)$ running time that accepts a set with no n^k size circuits.*

A similar constructive proof is still open for the stronger statements, i.e., the existence of a set with n^k circuit size lower bound in $\Sigma_2^{\mathrm{p}} \cap \Pi_2^{\mathrm{p}}$ (resp., $\mathrm{ZPP}^{\mathrm{NP}}$, and $\mathrm{S}_2^{\mathrm{p}}$).

Our main result in this paper deals with the difficulty in proving super polynomial circuit size lower bound for any set in the Polynomial-time Hierarchy, PH. While it is possible to prove lower bound above any fixed polynomial, at least for some sets in Σ_2^{p}, the real challenge is to prove super polynomial circuit size lower bound for a single language. Not only have we not been able to do this for any set in NP, but also no super polynomial lower bound is known for any set in PH. In this paper we prove that it is infeasible to give relativizable super polynomial lower bound for any set in the Polynomial-time Hierarchy.

Our model of relativized circuit computation is standard, having AND, OR, NOT and oracle query gates with the important proviso of "a reasonable access to an oracle" X. Here an oracle query gate takes m input bits $z = b_1 b_2 \ldots b_m$, and has output $\chi_{[z \in X]}$, i.e., it outputs 1 or 0 depending on whether $z \in X$ or otherwise. The proviso of "a reasonable access to an oracle" is as follows: To show that, at length n, a circuit C_n computes the language of machine $\mathcal{M}$ with running time n^k, we allow the circuit only access to those strings that can be accessed by the simulated machine $\mathcal{M}$, namely all strings of length at most n^k. This is a natural requirement; otherwise, it would have been somewhat trivial to show some circuit capturing $L(\mathcal{M}^X)$ by coding its computation in the oracle X at a location with length longer than any that are accessible by $\mathcal{M}$.

In this model of reasonable access to an oracle, we prove that for any alternating oracle TM $\mathcal{M}$ with running time $O(n^k)$, there is an oracle X and a polynomial size circuit family accepting it.

Theorem 2 (Main Theorem) *For any integer $d > 0$ and any real $k > 1$, let $\mathcal{M}$ be an oracle Σ_d^{p}-machine with running time $O(n^k)$. Then we have an oracle X and a family of Boolean circuits $\{C_n\}_{n \geq 0}$ with a reasonable access to the oracle X that accepts $L(\mathcal{M}^X)$. For all sufficiently large n, the size of C_n is bounded by n^{cdk}, for some universal constant $c > 0$.*

This result shows that for proving n^k circuit size lower bound in Σ_d^{p} by a relativizable technique, we need at least $\Omega(n^{k/cd})$ running time. In particular, it is infeasible to give a relativizable proof of super polynomial circuit size lower bound for any set in PH.

Our proof technique for the main theorem is based on the decision tree version of the Switching Lemma for constant depth circuits and Nisan-Wigderson pseudorandom generator.

Ko has given a survey of proof techniques to construct oracles using circuit lower bounds, particularly the Switching Lemma [Ko89b]. In previous work, such as Ko's theorem giving an oracle X such that $\Sigma_d^{\mathrm{p,X}} = \Sigma_{d+1}^{\mathrm{p,X}}$, while separating $\Sigma_d^{\mathrm{p,X}}$ and $\Sigma_{d-1}^{\mathrm{p,X}}$ [Ko89a], one can define the

[1]We have learned that this was also proved independently by Daniels [Dan].

oracle X on specific lengths to separate certain language of $\Sigma_d^{\mathrm{p},\mathrm{X}}$ from lower classes, while code, at a higher length of X, a complete language of $\Sigma_{d+1}^{\mathrm{p},\mathrm{X}}$ so that a $\Sigma_d^{\mathrm{p},\mathrm{X}}$ machine can access it. This method of killing at one length but coding at a higher length is justifiable if one is considering the whole class. For our purpose of proving bounds for specific polynomials n^k, to code at a higher length would be "cheating". Thus we specifically prohibit this. It appears that the use of Nisan-Wigderson pseudorandom generator in this proof is crucial.

Preliminaries

In this paper, we consider computation on binary strings, and all sets are subsets of $\{0,1\}^*$. We assume the standard length-wise lexicographic order on $\{0,1\}^*$. By "the ith string in $\{0,1\}^n$" we mean the ith string in $\{0,1\}^n$ under this ordering.

By a *circuit* we consider a standard Boolean circuit consisting of AND, OR, and NOT gates and input gates. We consider a circuit as an acceptor of a fixed length binary string, and for a set L, we consider a family of circuits $\{C_n\}_{n\geq 0}$, where each C_n is used as an acceptor for $L^{=n}$. For relativized computation, we allow circuit to use an oracle query gate that is explained above. The size of a circuit is the total number of wires in the circuit. Note that an oracle query gate asking a query of m bits contributes m to circuit size. For any circuit C, we use size(C) to denote its circuit size.

2. Proof of Theorem 1

R. Kannan proved that for any fixed polynomial n^k, there is some set L in $\Sigma_2^{\mathrm{p}} \cap \Pi_2^{\mathrm{p}}$ with circuit size $> n^k$. However, in terms of explicit construction, he only gave a set in $\Sigma_4^{\mathrm{p}} \cap \Pi_4^{\mathrm{p}}$. An improvement to Δ_3^{p} was stated in [MVW99].

In this section we give a constructive proof of Kannan's theorem for Σ_2^{p}. (Though not essential, for simplifying our discussion, we will assume that $k \geq 2$ throughout this section.)

For any $n \geq 0$, a binary sequence χ of length $\ell \leq 2^n$ is called a *partial characteristic sequence*, which will specify the membership of lexicographically the first ℓ strings of $\{0,1\}^n$. We denote this subset of $\{0,1\}^n$ by $L(\chi)$. We say that χ is consistent with a circuit C with n input gates, iff $\forall i$, $1 \leq i \leq \ell$, $C(x_i)$ outputs the ith bit of χ, where x_i is the ith string of $\{0,1\}^n$.

Let $\mathrm{len}(s) = c_{\mathrm{circ}} \lfloor s \log s \rfloor$. We can encode every circuit C of size $\leq s$ as a string u of length $\mathrm{len}(s)$. (For avoiding the special case, we assume that $s > 1$.) We may consider every u with $|u| = \mathrm{len}(s)$ encodes a circuit of size $\leq s$; if it is improperly coded or the circuit has size $> s$, we assume that this u encodes the constant 0 circuit. Then the following lemma is immediate by counting.

Lemma 1 *For any $s > 1$, there exists a partial characteristic sequence of length $\ell = \mathrm{len}(s) + 1$ that is not consistent with any circuit of size $\leq s$.*

We may also assume that there is a deterministic machine $\mathcal{M}_{\mathrm{circ}}$ that simulates, on any string v of length $\mathrm{len}(s)$ and any string x of length n, the execution of the circuit encoded by v on x, and the time complexity of $\mathcal{M}_{\mathrm{circ}}(v,x)$ is $O(|v|^2)$. ($\mathcal{M}_{\mathrm{circ}}$ simply returns 0 if v does not encode properly some circuit of size $\leq s$ with n input gates.)

Denote $s = n^k$ and $\ell = \mathrm{len}(s) + 1$. By "$v \succ u$" we mean that u is a prefix of v. We define a set PreCIRC as follows. For any $n > 0$, and for any strings χ of length ℓ and u of length $\leq \mathrm{len}(s)$,

$1^n 0 \chi u 0 1^{\mathrm{len}(s)-|u|} \in \mathrm{PreCIRC}$
$\quad \Leftrightarrow \ (\exists v \succ u)\ [\ |v| = \mathrm{len}(s)$, and the circuit encoded by v is consistent with χ].

Strings of any other form are not contained in PreCIRC. For simplifying our notation, we will simply write (χ, u) for $1^n 0 \chi u 0 1^{\mathrm{len}(s)-|u|}$. Since n determines s and ℓ, and the length of χ is ℓ, χ and u are uniquely determined from $0^n 1 \chi u 1 0^{\mathrm{len}(s)-|u|}$. We will use $\widetilde{n}$ to denote the length of strings of the form (χ, u); note that $\widetilde{n} = n + \ell + \mathrm{len}(s) + 2$ and hence $\widetilde{n}$ is $O(n^k \log n)$.

We note here the following fact. (Its proof, which is straightforward, is left to the reader.)

Lemma 2 *There exists a Δ_2^{p} machine such that for any input (χ, u) of length $\widetilde{n}$ defined w.r.t. s, it determines whether (χ, u) is in* PreCIRC *in time $O(|\mathrm{len}(s)|^3) = O(|\widetilde{n}|)$.*

We now define our machine $\mathcal{M}$. Informally we want $\mathcal{M}$ to accept an input x if and only if either $x \in 1\{0,1\}^{n-1}$ and $x \in \mathrm{PreCIRC}$, or $x \in 0\{0,1\}^{n-1}$ and $x \in L$, where L is a set with no n^k size circuits *if* $\mathrm{PreCIRC}^{=n}$ has n^k size circuits for all sufficiently large n. Specifically, $\mathcal{M}$ is designed so that $L^{=n}$ would be $L(\chi_{\mathrm{non}})$ where χ_{non} is lexicographically the first χ of length ℓ with no n^k size circuit, *provided* $\mathrm{PreCIRC}^{=\widetilde{n}}$ has a $\widetilde{n}^k$ size circuit for length $\widetilde{n}$.

More formally, for any given input x, if x starts with 1, then $\mathcal{M}$ accepts it iff $x \in \mathrm{PreCIRC}$. Suppose otherwise; that is, x starts with 0. Then first $\mathcal{M}$ existentially guesses a partial characteristic sequence χ_{non} of length ℓ *and* a circuit C_{pre} of size $\widetilde{n}^k$, more precisely, a string v_{pre} of length $\mathrm{len}(\widetilde{n}^k)$ encoding a circuit for $\mathrm{PreCIRC}^{=\widetilde{n}}$ of size $\leq \widetilde{n}^k$. (Below we use C_{pre} to denote the circuit that is encoded by the guessed v_{pre}.) After that, $\mathcal{M}$ enters the universal stage, where it checks the following items.

(1.) $\forall \chi$, $|\chi| = \ell$, and $\forall u$, $|u| \leq \mathrm{len}(s)$, check that C_{pre} is "locally consistent" on (χ, u) as follows:

$$\begin{aligned}
&C_{\mathrm{pre}}(\chi, u) = 1 \ \&\ |u| = \mathrm{len}(s) \\
&\quad \Longrightarrow \text{ the circuit that } u \text{ encodes is consistent with } \chi, \text{ and} \\
&C_{\mathrm{pre}}(\chi, u) = 1 \ \&\ |u| < \mathrm{len}(s) \\
&\quad \Longrightarrow \text{ either } C_{\mathrm{pre}}(\chi, u0) = 1 \text{ or } C_{\mathrm{pre}}(\chi, u1) = 1.
\end{aligned}$$

(2.) $\forall v$, $|v| = \mathrm{len}(s)$, compute the χ_v of length ℓ defined by (the circuit encoded by) v, and verify that C_{pre} works for χ_v and all prefix u of v, i.e., $C_{\mathrm{pre}}(\chi_v, u) = 1$.

(3.) The guessed χ_{non} is lexicographically the first string of length ℓ such that no circuit of size s $(= n^k)$ is consistent with it, *according to* C_{pre}. That is, check $C_{\mathrm{pre}}(\chi_{\mathrm{non}}, \epsilon) = 0$, where ϵ is the empty string, and $\forall \chi$ if $|\chi| = \ell$ and χ is lexicographically smaller than χ_{non} then $C_{\mathrm{pre}}(\chi, \epsilon) = 1$ holds.

Finally on each universal branch, if $\mathcal{M}$ passes the particular test of this branch, then $\mathcal{M}$ accepts the input $x \in 0\{0,1\}^{n-1}$ iff χ_{non} has bit 1 for the string x.

For all (χ, u), such that $|\chi| = \ell$ and $|u| \leq \mathrm{len}(s)$, if C_{pre} passes (1.) then $C_{\mathrm{pre}}(\chi, u) = 1 \Longrightarrow (\chi, u) \in \mathrm{PreCIRC}$, and if C_{pre} passes (2.) then $(\chi, u) \in \mathrm{PreCIRC} \Longrightarrow C_{\mathrm{pre}}(\chi, u) = 1$. Of course there is no guarantee that there exists a circuit C_{pre} (more precisely, v_{pre}) that will pass the tests in items (1.) and (2.) But if there is such a C_{pre}, then we can assume that C_{pre} correctly decides $\mathrm{PreCIRC}^{=\widetilde{n}}$ on strings of the form (χ, u), and the test in item (3.) is correctly performed. Thus, such a C_{pre} exists, then some existential path leads to C_{pre} together with the right χ_{non}. That is, $\mathcal{M}$ accepts $x \in 0\{0,1\}^{n-1}$ iff x is in $L(\chi_{\text{non}})$, where χ_{non} is lexicographically the first string of length ℓ with no consistent circuit of size $\leq n^k$.

On the other hand, if C_{pre} (satisfying (1.) and (2.)) does not exist, then $\mathcal{M}$ simply rejects all $x \in 0\{0,1\}^{n-1}$. But the nonexistence of C_{pre} means that there is no circuit of size $\widetilde{n}^k$ that is consistent with $\text{PreCIRC}^{=\widetilde{n}}$ on inputs of the form (χ, u) of length $\widetilde{n}$, in particular, strings in $1\{0,1\}^{\widetilde{n}-1}$. Thus, the desired hardness is guaranteed by the $1\{0,1\}^*$ part of $L(\mathcal{M})$. Therefore, we can conclude that $L(\mathcal{M})$ has no n^k size circuit. This Σ_2^{p} language proves Theorem 1.

It can be easily checked that the machine $\mathcal{M}$ runs in $O(n^{k^2} \log^{k+1} n)$ steps.

3. Proof of Theorem 2

We first give an outline of the proof.

It is well known from [FSS81] that a Σ_d^{p} alternating Turing machine $\mathcal{M}$ bounded in time n^k with oracle X, when given input x of length n, gives rise to a bounded depth Boolean circuit C_x of the following type: The inputs are Boolean variables representing membership of a string $z \in \{0,1\}^{\leq n^k}$ in the oracle X. The Boolean circuit C_x starts with an OR gate at the top, and alternate with AND's and OR's with depth $d+1$, where the bottom level gates have bounded fan-in at most n^k, and all other AND and OR gates are unbounded fan-in, except by the overall circuit size, which is bounded by $n^k 2^{n^k}$. Without loss of generality we may assume the Boolean circuit is tree like, except for the input level, where each Boolean variable corresponding to $\chi_{[z \in X]}$ is represented by a pair of complemented variables, which we will denote by z and $\overline{z}$.

Our first idea is to use random restrictions to "kill" the circuit. Here is what we mean. It is known that after a suitably chosen random restriction ρ, the circuit is sufficiently weakened so as to have either small min-terms or small max-terms. Results of this type are generally known as Switching Lemmas, and the strongest form known is due to Håstad [Hås86a] (see also [Ajt83, FSS81, Yao85, Cai86, Hås86b]). However it turns out that we need a different form, namely Switching Lemma of a decision tree type. We want to assign a suitably chosen random restriction ρ, after which the circuit admits a small depth decision tree. The reason we need this is as follows: We in fact will have to consider an aggregate of 2^n such Boolean circuits C_x simultaneously, one each for an input x of size n. We want to assign ρ, after which all these circuits have small depth decision trees. We then will proceed to set those variables to ensure that all these circuits are "killed", i.e., they all have a definite value now, either 0 or 1. We want to assign those variables consistently over all 2^n small depth decision trees. For decision trees, it is easy to achieve this by always setting "the next variable" asked by the decision tree to 0, say; it is not clear how to maintain this consistency in terms of min-terms and max-terms. If each decision tree has depth bounded by t, then we will have assigned at most $2^n t$ many variables corresponding to those strings of length n^k where ρ initially assigned a $*$ (i.e., they are left unassigned by ρ). We will argue that there are still plenty of unassigned variables left, where we may try to encode the now-determined computational values of these 2^n circuits. We will argue that t is sufficiently small, and yet with high probability all 2^n circuits admit decision trees of depth at most t.

The problem with this idea is that after we have coded the values of all the 2^n circuits in X, there does not seem to be any easy way to recover this information. Since X had already been "ravaged" by the random restriction ρ, it is not clear how to distinguish those "code bits" from those “random bits”. Further complicating the matter are those bits assigned during the decision tree settlement. All of this must be sorted out, supposedly, by a polynomial size oracle circuit which is to accept $L(\mathcal{M}^X)^{=n}$. Note that, after a random restriction ρ, it is probabilistically almost impossible to have an easily identifiable segment of the set X all assigned $*$ by ρ, (e.g., all strings in $\{0,1\}^{=n^k}$ with a certain leading bit pattern), not to mention the subsequent all 0 assignment to fix the decision trees. On the other hand, we have 2^n computations to code. It is infeasible for the final polynomial size oracle circuit to “remember” more than a polynomial number of bits as the address of the coding region. So it appears that we must have an easily identifiable region to code, identified with at most a polynomial number of bits for its address, and, to accommodate 2^n computations, this region must be large.

To overcome this difficulty, our idea is to use not true random restrictions, but pseudorandom restrictions via the Nisan-Wigderson generator [Nis91a, Nis91b, NW88]. Nisan and Wigderson designed a pseudorandom generator provably indistinguishable from true random bits by polynomial size constant depth circuits. While our circuits are not of polynomial size, this can be scaled up easily. Our idea is then to use the output of the pseudorandom NW generator to perform the “random” restriction, and to argue that all 2^n circuits are “killed” with high probability, just as before with true random restrictions. The basic argument is that no constant depth circuits of an appropriate size can tell the difference under either a true random assignment or a pseudorandom assignment coming from the NW generator. However, for our purpose in this paper, we wish to say that a certain behavior of these 2^n constant depth circuits—namely they are likely to possess small depth decision trees after a “random” restriction with 0, 1 and $*$'s—is preserved when “pseudorandom restrictions” are substituted for “random restrictions”. It is vitally important that whatever property we wish to claim to have been maintained by the substitution of random bits by pseudorandom bits, the property must be expressible as a constant depth circuit with an appropriate size upper bound. It is not clear the property of “having a small depth decision tree” can be expressed in this way.

We overcome this difficulty by using a weaker property which is a consequence of “having a small depth decision tree”, which nonetheless is sufficient for our purpose. Namely, we take directly the property that, after a restriction with 0, 1 and $*$'s, all 2^n circuits can be determined after an additional small number of 0's are assigned for each circuit. This property is expressible in a constant depth way. Then we will mimic the probability distribution of the 0, 1 and $*$'s under the random restrictions by uniform random bits 0's and 1's, so that we can come up with a constant depth circuit D with the following property: It takes only boolean inputs y of 0's and 1's, and D evaluates to 1 iff when a restriction $\rho = \rho_y$ with 0, 1 and $*$'s defined by y is applied to all 2^n circuits C_x, every C_x can be set to either 0 or 1 after a small number of additional variables are set to 0. We will design D in such a way that under a uniform bit sequence y, D will almost certainly evaluate to 1.

In fact we need more than that. We also need to have the property that a certain segment of the oracle is untouched by the additional setting of 0's in all 2^n decision tree settlements. We will argue by the pigeonhole principle, that our bounds guarantee a suitable region unspoiled by all these decision tree settlement variables. It is not reasonable to expect that any such region is entirely assigned with $*$'s, but at least there should be many $*$'s. (We can also include some mechanism in D to guarantee the existence of such region. For example, we first identify, in the construction of D, a certain specific set of additional variables for each C_x, e.g., the lexicographically the first set, such that they are of an appropriate size and were all set to $*$ initially, and when all set to 0, settles C_x. Then we explicitly require in the construction of D that they do not touch a certain segment of the oracle. This can be done in D, i.e., expressible in a constant depth without much bigger size. However, for the bound we will present in this paper, the simpler approach will circumvent this difficulty.)

Assume now we have designed such a D satisfying all these requirements. For this D we apply the NW generator, substituting pseudorandom bits for true random bits as its input y. We conclude that D still evaluates to 1 with high probability. In particular, there must be some setting of the source bits for the generator, such that D is evaluated to 1. This implies that we can assign the oracle set X first according to the pseudorandom restriction described by the pseudorandom bits, then according to the 2^n small depth decision trees, which are guaranteed by the evaluation of D, and set these additional variables all to 0. This settles all the decision trees and thus the values of all 2^n circuits C_x are determined. Furthermore there is a significant segment T_{z_0} of X free from any variables used in any decision tree settlement, where we will code these 2^n results of C_x.

Even though this segment T_{z_0} is free from any variables used in any decision tree settlement, in order to code the computation results of C_x there must be plenty of $*$ left, and they must be recoverable by polynomial size circuits. We will show that with high probability over the pseudorandom bits y, the pseudorandom restriction defined by y will leave plenty of $*$ in each segment such as T_{z_0}. We then in fact choose a sequence of bits y that satisfy both the requirement $D = 1$ and this additional requirement.

Finally, we will show that with a suitable choice of parameters in the *combinatorial design* used in the NW generator, we will be able to recover in polynomial time the location where we assigned $*$'s in X, in particular from within the coding segment of X given the address of this segment. The polynomial size circuit will remember (hardwired with) z_0, the address of T_{z_0}, and remember the polynomially many source bits for the NW generator. On any input x, it will perform the polynomial time computation over a finite field to extract the coded result of C_x from the appropriate location in X.

3.1. Proof of Theorem 2

Fix any Σ_d^{p} polynomial time bounded oracle alternating Turing machine $\mathcal{M}$, with time bound n^k. For notational convenience we will assume $k > 2$ and $d \geq 7$. We assume that n is sufficiently large. On input of length n, $\mathcal{M}$ can only query strings of length at most n^k. We will denote by $m = n^k$ and use M to denote 2^m throughout this proof.

Assume all memberships in the oracle set "$z \in X$?" up to length $< m$ have been decided already. Our task is to fix the membership for "$z \in X$?" of length exactly m in X, so that for all input x of length n, membership "$x \in L(\mathcal{M}^X)$?" can be decided by a polynomial size circuit $C_{\mathcal{M}}$ with oracle gates that can access $X^{=m}$. Note that since $X^{<m}$ had already been fixed, membership "$x \in L(\mathcal{M}^X)$?" is determined by the set $X^{=m}$. Here we specifically require that the circuit $C_{\mathcal{M}}$ can access only those strings that can be possibly accessed by the simulated machine $\mathcal{M}$ on input of length n.

There are 2^n inputs x of length n, each computation of $\mathcal{M}$ on x gives rise to a depth $d+1$ Boolean circuit C_x with size at most mM, and bottom fan-in at most m. The input to each circuit C_x is the $2M$ literals z_i and $\overline{z_i}$, for $1 \leq i \leq M$, where z_i corresponds to the truth value of $\chi_{[z_i \in X]}$, for the lexicographically the i th string z_i in $\{0,1\}^m$. (While there is no confusion we will denote by z_i both the i th string in $\{0,1\}^m$ as well as the Boolean variable corresponding to $\chi_{[z_i \in X]}$.) As stated earlier, we may assume in C_x the circuit starts with an OR gate at the top, and alternate with AND's and OR's in a tree like fashion, until inputs z_i's and $\overline{z_i}$'s. Each circuit C_x has size at most mM. (Note that in this circuit formulation we simply replace each oracle query by the corresponding input. This is different from the standard circuit computation model (for discussing the cirucit complexity in relativized worlds) that assumes a query gate for oracle queries.)

For such constant depth circuits the following Switching Lemma due to Håstad [Hås86a] is well-known.

Lemma 3 (Håstad) *Let $G = G(z_1, \ldots, z_M)$ be of the form $G_1 \wedge G_2 \wedge \cdots \wedge G_M$, where each G_i is an OR of at most t literals. Let ρ be a random restriction which assigns $\Pr[\rho(x_i) = *] = p$ and $\Pr[\rho(x_i) = 0] = \Pr[\rho(x_i) = 1] = (1-p)/2$, for each i independently, then*

$$\Pr[G \mid_\rho \text{ has a minterm of size } \geq s] \leq (5pt)^s.$$

A dual statement also holds for G as an OR of AND's.

The purpose of this lemma is to effect a successive conversion of all the bottom 2-level circuits from AND's of OR's to OR's of AND's (or vice versa). As indicated before, for our purpose in this paper, we will require something more.

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 $\mathrm{DC}(f) \leq t$, 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, clearly, there is a subset of no more than t variables, if one assigns all of them to 0, the function f will be determined. This is an important advantage as we will have to assign many non-disjoint subsets of variables for multiple Boolean functions, and all these assignments need to be consistent.

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

Lemma 4 *For any depth $d+1$ Boolean circuit C on $z_1, \ldots, z_M$ of size s and bottom fan-in at most t,*

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

where the random restriction ρ has $p = \frac{1}{(10t)^d}$.

Then the following lemma can be proved by the Decision Tree Lemma 4.

Lemma 5 *For any depth d Boolean circuit C on $z_1, \ldots, z_M$ of size at most $2^{cM^{\frac{1}{d}}}$, the discrepancy*

$$|\Pr[C = \oplus] - \Pr[C \neq \oplus]| \leq \frac{1}{2^{cM^{\frac{1}{d}}}},$$

where the probability is taken uniformly over all 2^M assignments to $z_1, \ldots, z_M$, where $\oplus = \oplus(z_1, \ldots, z_M)$ is the parity function, and c is a universal constant.

The above bound is optimal up to the constant c. We will also use the slightly weaker form $2^{M^{\frac{1}{d+1}}}$ in place of $2^{cM^{\frac{1}{d}}}$; this is valid for all sufficiently large M. In fact qualitatively what we prove in this paper can also be done with weaker forms of the bound (see Remark 4 later); however, the decision tree version of the Switching Lemma is crucial.

We will carry out a sequence of transformations on the circuits C_x, for each $x \in \{0,1\}^n$, with the ultimate goal of constructing the circuit D which, in some sense, is a test for the success of a "random restriction".

Step 1 (C^1_x): C^1_x takes $2M$ Boolean inputs (a_i, b_i), for $i = 1, \ldots, M$. The pair (a_i, b_i) will represent the status of the Boolean variable z_i to C_x as follows: $a_i = 1$ iff z_i is set (to either 0 or 1, i.e., not set to $*$), and $a_i = 0$ otherwise. If $a_i = 1$ then $z_i = b_i$, i.e., the 0-1 value of z_i is represented by b_i. If the pair (a_i, b_i) represents z_i, then the pair $(a_i, \overline{b_i})$ represents $\overline{z_i}$. Clearly, if z_i is set 0 (resp., 1), then $\overline{z_i}$ must be set 1 (resp., 0).

C^1_x is constructed from C_x as follows. Each gate g in C_x will be represented by a pair of gates (g_s, g_v). $g_s = 1$ iff g is set to either 0 or 1, i.e., it is determined; $g_s = 0$ otherwise. If $g_s = 1$ then $g = g_v$. Thus, $(g_s, g_v) = (0,0)$ or $(0,1)$ represent the situation where g has not been determined, and $(g_s, g_v) = (1,0)$, or $(1,1)$ respectively, represent the case where g is set to 0, or 1 respectively.

Suppose $g = \bigvee_{i=1}^s g^{(i)}$, where $g^{(i)}$ is an input literal or an internal gate. Suppose $g^{(i)}$ is represented by the pair $(g^{(i)}_s, g^{(i)}_v)$, then we let

$$g_s = \bigvee_{i=1}^{s} \left((g^{(i)}_s \wedge g^{(i)}_v) \right) \vee \left(\bigwedge_{i=1}^{s} (g^{(i)}_s \wedge \overline{g^{(i)}_v}) \right).$$

That is, g is set iff either some g_i is set to 1, or else all g_i are set to 0. Note that the formula given for g_s is a depth 2 circuit of size $O(s)$. Also

$$g_v = \bigvee_{i=1}^{s} g^{(i)}_v.$$

Note that g_v is only a "valid" value for g when $g_s = 1$. Also g_v is depth 1 and has size s.

The case $g = \bigwedge_{i=1}^s g^{(i)}$ is dual. In this case, g is set iff either some g_i is set to 0, or else all g_i are set to 1. Thus

$$g_s = \bigvee_{i=1}^{s} \left((g^{(i)}_s \wedge \overline{g^{(i)}_v}) \right) \vee \left(\bigwedge_{i=1}^{s} (g^{(i)}_s \wedge g^{(i)}_v) \right), \text{ and } g_v = \bigwedge_{i=1}^{s} g^{(i)}_v.$$

Again they are depth 2, size $O(s)$, and depth 1, size s, respectively.

In order to maintain alternating form of AND's and OR's in the circuit C^1_x, with all negations pushed to the input level, we can represent each gate g by both g and its negated value $\overline{g}$. This can introduce at most a factor of 2 in the size. C^1_x has two output gates g_s and g_v for the output gate g of C_x. It follows that

$$\text{size}(C^1_x) = O(\text{size}(C_x)), \text{ and } \text{depth}(C^1_x) = 2 \text{ depth}(C_x).$$

Step 2 (C^2_x): Let $p = \frac{1}{(2m)^d}$. Let $L = \lceil \log_2 \frac{1}{p} \rceil \approx dk \log_2(20n)$. C^2_x takes Boolean inputs $(a_{i,1}, \ldots, a_{i,L}, b_i)$, for $i = 1, \ldots, M$. The circuit C^2_x is identical to C^1_x, except instead of taking inputs a_i, it has $a_i = \bigvee_{j=1}^L a_{i,j}$.

Note that, the random restriction ρ with p on C_x is simulated by uniformly and independently assigning all the bits $(a_{i,1}, \dots, a_{i,L}, b_i)$ to 0 or 1, in C_x^2, for $i = 1, \dots, M$. The behavior of C_x is represented in C_x^2 exactly.

$$\text{size}(C_x^2) = \text{size}(C_x^1) + O(ML), \text{ and } \text{depth}(C_x^2) = \text{depth}(C_x^1) + 1.$$

Step 3 (C_x^3)**:** In C_x^3 we will check for the existence of a subset $S \subset [M]$ of cardinality $|S| = 2m$ such that, first they are assigned $*$ by the ρ, and second if we further set them all to 0, it would determine the circuit C_x. We know from Lemma 4 that this is almost certainly true for a random restriction ρ with $p = \frac{1}{(20m)^d}$. The failure probability for this to happen is bounded above by $\text{size}(C_x)/2^{2m}$.

Thus, we let

$$C_x^3 = \bigvee_S \left[\bigwedge_{i \in S} \overline{a_i} \wedge [(C_x^2)_s]_S \right],$$

where $\bigvee_S$ ranges over all subsets $S \subset [M]$ of cardinality $|S| = 2m$, and $(C_x^2)_s$ is the "set bit output" for C_x^2, and $[(C_x^2)_s]_S$ is obtained from $(C_x^2)_s$ by setting all $b_i = 0$ for $i \in S$. Recall that $a_i = \bigvee_{j=1}^L a_{i,j}$.

$$\text{size}(C_x^3) \le \binom{M}{2m}(\text{size}(C_x^2) + O(m)), \text{ and } \text{depth}(C_x^3) = \text{depth}(C_x^2) + 2.$$

Step 4 (D)**:** Finally, define D by $D = \bigwedge_{x \in \{0,1\}^n} C_x^3$. Then we have

$$\text{size}(D) = 2^n(\text{size}(C_x^3)), \text{ and } \text{depth}(D) = \text{depth}(C_x^3) + 1.$$

This completes the construction of D, with

$$\text{size}(D) < 2^{3m^2}, \text{ and } \text{depth}(D) \le 2d + 6 < 3d.$$

Applying Lemma 4 to all C_x simultaneously, with $p = \frac{1}{(20m)^d}$, we get

$$\Pr[(\exists x \in \{0,1\}^n)[\text{DC}(C_x \restriction_\rho) \ge 2m]] \le 2^n \cdot \frac{mM}{2^{2m}} = \frac{m}{2^{m-n}}.$$

When each $C_x \restriction_\rho$ has decision tree depth at most $2m$, D is true. As the uniform independent distribution on the input bits of D simulates the random restriction ρ, we conclude that

$$\Pr[D = 1] \ge 1 - \frac{m}{2^{m-n}}, \tag{1}$$

where the probability is over uniform input bits of D, namely $a_{i,1}, \dots, a_{i,L}, b_i$, for $i = 1, \dots, M$, and $L = \lceil \log_2 \frac{1}{p} \rceil$.

Now we will apply the NW generator to this circuit D. The fact that the output of this generator is indistinguishable from true random bits for D follows from the properties of the NW generator, which ultimately follows from the Inapproximability Lemma 5 and the fact that D is a constant depth circuit. The proofs by Nisan and Wigderson in [Nis91b, NW88] for polynomial size circuits can be scaled up and we omit the details.

We set the parameters in the combinatorial design used by Nisan and Wigderson [Nis91b, NW88]. We will take a specific finite field $\mathbf{F} = \mathbf{Z}_2[X]/(X^{2\cdot 3^u} + X^{3^u} + 1)$ [vL91], where each element $\alpha \in \mathbf{F}$ takes $K = 2 \cdot 3^u$ bits, and we denote $q = |\mathbf{F}| = 2^K$. We will choose u so that $q \geq (3m^2)^{3d}$. Clearly $q \leq n^{ckd}$ will do, for some universal constant c, for example $c = 7$. Then $K = O(dk \log n)$. Thus, this field has polynomial size and each element is represented by $O(\log n)$ bits. All arithmetic in the field $\mathbf{F}$ is easy.

We will consider precisely 2^m polynomials $f_z(\xi) \in \mathbf{F}[\xi]$, each of degree at most m, where each f_z is indexed by its coefficients, concatenated as a bit sequence of length exactly m. The precise manner in which this is done is not very important, but for definiteness, we can take the following. We take polynomials of degree $\delta = \lfloor m/K \rfloor = \Omega(n^k/(dk \log n)) \gg n^2$, with exactly $\delta + 1$ coefficients,

$$f_z(\xi) = c_\delta \xi^\delta + \ldots + c_1 \xi + c_0,$$

where all c_j varies over $\mathbf{F}$, except c_δ is restricted to exactly $2^{m-K\cdot\delta}$ many values. Note that $0 \leq m - K \cdot \delta < K$. The concatenation $z = \langle c_\delta \ldots c_0 \rangle$ has exactly m bits, and the polynomial f_z will be used to determine the pseudorandom restriction on $z \in \{0,1\}^m$.

The source bits for the generator is a sequence of independently and uniformly distributed bits $\{b^{(0)}_{\alpha,\beta}, b^{(1)}_{\alpha,\beta}, \ldots, b^{(L)}_{\alpha,\beta}\}$, for each $\alpha, \beta \in \mathbf{F}$, where $L = \lceil \log_2 \frac{1}{p} \rceil$ was the number of fan-in to form $a_z = \bigvee_{j=1}^{L} a_{z,j}$, Let $\boldsymbol{b}_{\alpha,\beta}$ be a column vector of 0-1 uniform bits $(b^{(1)}_{\alpha,\beta}, b^{(2)}_{\alpha,\beta}, \ldots, b^{(L)}_{\alpha,\beta})^{\mathrm{T}}$.

Each f_z defines a subset of $\mathbf{F} \times \mathbf{F}$ of cardinality q, $\{(\alpha, f_z(\alpha)) \mid \alpha \in \mathbf{F}\}$, which we will denote by F_z. If $\deg(f_z) < m$ and $\deg(f_{z'}) < m$, then $|F_z \cap F_{z'}| < m$, for all $z \neq z'$.

We let $a_{z,j}$, part of the input bits to D, be the parity sum of $b^{(j)}_{\alpha,\beta}$, over F_z, i.e., we let $a_{z,j} = \oplus_{\alpha \in \mathbf{F}} b^{(j)}_{\alpha, f_z(\alpha)}$. Thus, z is assigned a $*$ by this pseudorandom restriction iff $a_z = 0$ iff $\sum_{\alpha \in \mathbf{F}} \boldsymbol{b}_{\alpha, f_z(\alpha)} = \mathbf{0}$ in $\mathbf{Z}_2^L$. Also we compute b_z, input to D as well, to be the parity sum of $b^{(0)}_{\alpha,\beta}$, over F_z, i.e., $b_z = \oplus_{\alpha \in \mathbf{F}} b^{(0)}_{\alpha, f_z(\alpha)}$.

By Eqn (1), the NW generator guarantees that

$$\Pr[D = 1] = 1 - o(1), \tag{2}$$

where now the probability is over independently and uniformly distributed bits $\{b^{(0)}_{\alpha,\beta}, b^{(1)}_{\alpha,\beta}, \ldots, b^{(L)}_{\alpha,\beta}\}$, for $\alpha, \beta \in \mathbf{F}$.

By the pigeonhole principle, there is a prefix y_0, consisting of the concatenation $\langle c_\delta \ldots c_{\delta - n^2} \rangle$, such that the segment $T_{y_0} = \{z \in \{0,1\}^m \mid y_0 \text{ is a prefix of } z\}$ of X is free from any variables used in any of the 2^n decision tree settlements. This is simply because $2^n \cdot 2m \ll 2^{n^2}$.

Our plan is to code the results of C_x in a $*$-place of the form $\langle y_0 x x' \rangle$, for some x'. This is within the segment T_{y_0}. In the polynomial f_z, for $z = \langle y_0 x x' \rangle$, x has n bits, which takes up to n additional coefficients. We will round it off, by padding v many 0's, say, and then $z = \langle y_0 x 0^v x'' \rangle$, where x'' consists of an integral number of coefficients, $c_\gamma, \ldots, c_1, c_0$ of f_z. We want

$$\forall c_\delta, \ldots, c_{\gamma+1},\ \exists c_\gamma, \ldots, c_1, c_0 \ \left[\sum_{\alpha \in \mathbf{F}} \boldsymbol{b}_{\alpha, f_z(\alpha)} = \mathbf{0} \right], \tag{3}$$

where $z = \langle c_\delta, \ldots, c_0 \rangle$.

We will show that with high probability a random choice of all the bits $\{b^{(1)}_{\alpha,\beta}, \ldots, b^{(L)}_{\alpha,\beta}\}$ satisfies this requirement.

In fact we will be more drastic. We show that we can set $c_\gamma = \ldots = c_1 = 0$, and still satisfy the requirement (3). That is, the following holds.

$$\forall c_\delta, \ldots, c_{\gamma+1},\ \exists c_0 \ \left[\sum_{\alpha \in \mathbf{F}} \boldsymbol{b}_{\alpha, f_{z^*}(\alpha)} = \mathbf{0} \right], \tag{4}$$

where z^* denotes $\langle c_\delta, \ldots, c_{\gamma+1}, 0, \ldots, 0, c_0 \rangle$.

For any fixed $c_\delta, \ldots, c_{\gamma+1}$, let w denote $\langle c_\delta, \ldots, c_{\gamma+1} \rangle$, and define $g_w(\xi) = c_\delta \xi^\delta + \cdots + c_{\gamma+1}\xi^{\gamma+1}$. Then for any c, let $z^*(c)$ denote $\langle c_\delta, \ldots, c_{\gamma+1}, 0, \ldots, 0, c \rangle$. Clearly $f_{z^*(c)}(\xi) = g_w(\xi) + c$.

Now define $\boldsymbol{b}^*_{\alpha,c} = \boldsymbol{b}_{\alpha, g_w(\alpha)+c}$. Then $\boldsymbol{b}^*_{\alpha,c} = \boldsymbol{b}_{\alpha, f_{z^*(c)}(\alpha)}$. Thus, the above (4) can be stated as

$$\forall c_\delta, \ldots, c_{\gamma+1},\ \exists c_0 \ \left[\sum_{\alpha \in \mathbf{F}} \boldsymbol{b}^*_{\alpha, c_0}(\alpha) = \mathbf{0} \right]. \tag{5}$$

Notice that for any α and any $c \neq c'$, the vectors $\boldsymbol{b}^*_{\alpha,c}$ and $\boldsymbol{b}^*_{\alpha,c'}$ consist of disjoint sets of bits. Hence, they are independent, from which the following bound follows.

$$\Pr\left[\forall c_0 \ [\sum_{\alpha \in \mathbf{F}} \boldsymbol{b}^*_{\alpha,c_0} \neq \mathbf{0}\,] \right] \ = \ \prod_{c \in \mathbf{F}} \Pr\left[\sum_{\alpha \in \mathbf{F}} \boldsymbol{b}^*_{\alpha,c} \neq \mathbf{0} \right] \ = \ \left(1 - \frac{1}{2^L}\right)^q \ < \ e^{-\Omega(q/(20m)^d)},$$

where the probability is taken uniformly over all the bits $\{b^{(1)}_{\alpha,\beta}, \ldots, b^{(L)}_{\alpha,\beta}\}$, for all $\alpha, \beta \in \mathbf{F}$.

It follows that

$$\Pr\left[\forall c_\delta, \ldots, c_{\gamma+1}, \exists c_0 \ [\sum_{\alpha \in \mathbf{F}} \boldsymbol{b}^*_{\alpha,c_0}(\alpha) = \mathbf{0}\,] \right] \ \geq \ 1 - 2^{n^2K+n} e^{-\Omega(q/(20m)^d)} \ = \ 1 - o(1).$$

Armed with this estimate, we will choose a source sequence $\{b^{(0)}_{\alpha,\beta}, b^{(1)}_{\alpha,\beta}, \ldots, b^{(L)}_{\alpha,\beta}\}$ that satisfies both requirements (2) and (4). Since both event happens with probability approaching 1, this can be done.

Finally, note that for this choice of the source sequence, we can fix one y_0 with the following property: for every $x \in \{0,1\}^n$, there exists some x' such that $z = \langle y_0 x 0^v x' \rangle$ is assigned a $*$ by the random restriction and unassigned by any of the 2^n decision tree settlements. Note further that such x' can be found by checking all q $(= |\mathbf{F}| \leq n^{ckd})$ possibilities for c_0. That is, given polynomially many source bits and y_0, it is computable in polynomial time to find $z \in T_{y_0}$ that is unassigned by the random restriction, (and since it is within T_{y_0} it is unassigned by all the 2^n decision trees), and we can set any value to z without changing the output of $\mathcal{M}^X$ on x. We use all such z of this form in this segment to encode the result of $\mathcal{M}^X$ on x. This is our oracle construction.

Remark 1: For convenience we assumed in the proof that $k > 2$ and $d \geq 7$. Clearly $d \geq 7$ is unnecessary. We only need to forgo the estimate of $2d+6 < 3d$, and use $2d+6$. Similarly, $k > 2$ is not necessary. If one traces through the proof, any real number $k > 1$ is sufficient.

Remark 2: The final computation by the polynomial size circuit can be done in NC^1. We only need to evaluate some arithmetic operations in the finite field $\mathbf{F}$. It turns out that since elements in $\mathbf{F}$ are represented by $O(\log n)$ bits, the only step that really requires NC^1 is the parity sum of $n^{O(1)}$ terms, when we evaluate the polynomial f_z.
Remark 3: It is also possible to construct a single oracle X such that for every d and k, and every Σ_d^{p}-machine $\mathcal{M}$ running in time $O(n^k)$ can be accepted by a polynomial size circuit family with reasonable access to oracle X with respect to each $\mathcal{M}$.
Remark 4: The original motivation for Furst-Saxe-Sipser [FSS81] where super polynomial lower bounds were proved for parity against constant depth circuits, was to provide an oracle separation of PH and PSPACE. This was achieved in a breakthrough result by Yao [Yao85] who proved a lower bound of the form $2^{N^{\Omega(1/d)}}$ for parity on N bits for depth d circuits. Cai [Cai86] was the first to investigate whether constant depth circuits of size $2^{N^{\Omega(1/d)}}$ must err on an asymptotically 50 % of inputs against parity. This was motivated by another long standing open problem, that of random oracle separation of PH and PSPACE (see also [Bab87]). To attack this problem, the decision tree point of view was first adopted in [Cai86], although a different but completely synonymous terminology (Master-Player Game and t-monochromaticity) was used. It was proved in [Cai86] that after a suitable random restriction ρ, with high probability, the constant depth circuit $C\mid_\rho$ has decision tree depth smaller than the number of unassigned Boolean variables. In such cases, of course $\Pr[C = \oplus]$ is exactly $\frac{1}{2}$. Thus the discrepancy

$$|\Pr[C = \oplus] - \Pr[C \neq \oplus]| \tag{6}$$

was shown to be $o(1)$ for circuits of depth d and size $2^{N^{\Omega(1/d)}}$. Implicitly a bound of the form $2^{-N^{\Omega(1/d)}}$ for the discrepancy (6) was proved there as well [Cai86]. The $o(1)$ upper bound for the discrepancy was sufficient for the random oracle separation result which was the purpose of [Cai86], but one needs Håstad's technique to improve the bound from $2^{-N^{\Omega(1/d)}}$ to $2^{-cN^{\frac{1}{d}}}$ as in Lemma 4. However, the weaker bound $2^{-N^{\Omega(1/d)}}$ would have sufficed for our Theorem 2.

It was a marvelous application by Nisan and Wigderson [Nis91a, Nis91b, NW88] who turned this inapproximability type of lower bounds based on decision trees on its head, and produced an explicit construction—usually considered an upper bound—of a pseudorandom generator provably indistinguishable from true random bits by polynomial size constant depth circuits. A central ingredient in [Nis91a, Nis91b, NW88] is a suitable combinatorial design. Seen in this way, our proof of Theorem 2 can be viewed as using a *lower bound* (Switching Lemma), to get an *upper bound* (the NW pseudorandom generator), to prove a *lower bound* (to kill all 2^n circuits C_x simultaneously with the pseudorandom assignments), to finally prove an *upper bound* (to be able to code all the computations). And all this, is to show that relativizable proof with reasonable access to an oracle of circuit super polynomial *lower bound* for any fixed language in Polynomial-time Hierarchy is impossible.

Acknowledgments

We thank Professor Eric Allender for letting us know the result [Dan].

References

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

[Bab87] L. Babai, Random oracles separate PSPACE from the polynomial-time hierarchy, *Information Processing Letters*, 26(1), 51–53, 1987.

[BCGKT] N. Bshouty, R. Cleve, R. Gavaldà, S. Kannan, and C. Tamon, Oracles and queries that are sufficient for exact learning, *J. Comput. and System Sci.* 52(3), 421–433, 1996.

[BFT98] H. Buhrman, L. Fortnow, and T. Thierauf, Nonrelativizing separations, in *Proc. the 13th IEEE Conference on Computational Complexity* (CCC'98), IEEE, 8–12, 1998.

[Cai86] J-Y. Cai, With probability one, a random oracle separates PSPACE from the polynomial-time hierarchy, in *Proc. 18th ACM Symposium on Theory of Computing* (STOC'86), ACM, 21–29, 1986. JCSS 38(1): 68-85 (1989).

[Cai01] J-Y. Cai, $\mathrm{S}_2^{\mathrm{P}} \subseteq \mathrm{ZPP}^{\mathrm{NP}}$, in *Proc. 42th IEEE Symposium on Foundations of Computer Science* (FOCS'01), IEEE, 620–628, 2001.

[Dan] S. Daniels, A Constructive proof presenting sets in Σ_2 that require circuits of size n^k, unpublished manuscript.

[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.

[Hås86b] J. Håstad, *Computational Limitations for Small-Dept Circuits*, MIT Press, 1986.

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

[KL80] R.M. Karp and R.J. Lipton, Some connections between nonuniform and uniform complexity classes, in *Proc. 12th ACM Symposium on Theory of Computing* (STOC'80), ACM, 302–309, 1980.

[Ko89a] K. Ko, Relativized polynomial time hierarchies having exactly k levels, *SIAM J. Comput.*, 18, 392–408, 1989.

[Ko89b] Ker-I Ko, Constructing oracles by lower bound techniques for circuits, in *Combinatorics, Computing and Complexity* (D. Du and G. Hu eds.), Kluwer, 30–76, 1989.

[KW98] J. Köbler and O. Watanabe, New collapse consequences of NP having small circuits, *SIAM J. Comput.*, 28, 311–324, 1998.

[MVW99] P.B. Miltersen, N.V. Vinodchandran, and O. Watanabe, Super-Polynomial versus half-exponential circuit size in the exponential hierarchy, in *Proc. 5th Annual International Conference on Computing and Combinatorics* (COCOON'99), Lecture Notes in Computer Science 1627, 210–220, 1999.

[Nis91a] N. Nisan, Pseudorandom bits for constant depth circuits, *Combinatorica* 11(1), 63–70, 1991.

[Nis91b] N. Nisan, *Using Hard Problems to Create Pseudorandom Generators*, MIT Press, 1991.

[NW88] N. Nisan and A. Wigderson, Hardness vs. randomness, in *Proc. 29th IEEE Symposium on Foundations of Computer Science* (FOCS'88), IEEE, 2–12, 1988.

[vL91] J. van Lint. *Introduction to Coding Theory.* Springer-Verlag, 1991.

[Yao85] Andrew Chi-Chih Yao, Separating the Polynomial-Time Hierarchy by Oracles, FOCS 1985: 1-10.