The Power of Natural Properties as Oracles

Russell Impagliazzo
Department of Computer Science, University of California San Diego, La Jolla, CA, USA
russell@cs.ucsd.edu

Valentine Kabanets
School of Computing Science, Simon Fraser University, Burnaby, BC, Canada
kabanets@cs.sfu.ca

Ilya Volkovich
Department of EECS, CSE Division, University of Michigan, Ann Arbor, MI, USA
ilyavol@umich.edu

Abstract

We study the power of randomized complexity classes that are given oracle access to a natural property of Razborov and Rudich (*JCSS*, 1997) or its special case, the Minimal Circuit Size Problem (MCSP). We show that in a number of complexity-theoretic results that use the SAT oracle, one can use the MCSP oracle instead. For example, we show that $\mathsf{ZPEXP}^{\mathsf{MCSP}} \not\subseteq \mathsf{P/poly}$, which should be contrasted with the previously known circuit lower bound $\mathsf{ZPEXP}^{\mathsf{NP}} \not\subseteq \mathsf{P/poly}$. We also show that, assuming the existence of Indistinguishability Obfuscators (IO), SAT and MCSP are equivalent in the sense that one has a ZPP algorithm if and only the other one does. We interpret our results as providing some evidence that MCSP may be NP-hard under randomized polynomial-time reductions.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography

Keywords and phrases natural properties, Minimal Circuit Size Problem (MCSP), circuit lower bounds, hardness of MCSP, learning algorithms, obfuscation, Indistinguishability Obfuscators (IO)

Digital Object Identifier 10.4230/LIPIcs.CCC.2018.7

Acknowledgements We thank Eric Allender and Scott Aaronson for answering our questions and many useful conversations. We also thank the anonymous referees for their useful comments.

1 Introduction

Historically, the problem of minimizing a circuit representing a given Boolean function (MCSP) was one of the first where the prohibitive computational cost of searching through a huge space of candidate solutions was noted [30, 44]. This issue would later be formalized in the theory of NP-completeness. However, the complexity of circuit minimization itself remains largely mysterious. It is an NP problem, but neither known to be NP-complete nor in any sub-class of NP thought proper. This mystery remains despite a large body of work devoted to this problem [28, 2, 4, 3, 5, 24, 38, 23].

For negative hardness results, we do know that MCSP is *not* NP-hard (even P-hard) under very restrictive reductions [38]. We also know that MCSP is not NP-hard under certain "black-box" reductions [23]. For other kinds of restricted reductions, we know that proving the NP-hardness of MCSP under such reductions would be difficult as such a proof would also yield new circuit lower bounds [28, 38, 5].

On the other hand, for positive hardness result, we know that MCSP is SZK-hard under general randomized (BPP) reductions [3], and NC^1-hard under truth-table reductions computable by non-uniform TC^0 circuits [40].

Looking at the negative results about NP-hardness of MCSP, one has to wonder: Are these results actually about MCSP and its relationship to other problems, or about the weakness of certain types of reductions? Given the positive results about hardness of MCSP under more powerful reductions, it seems more likely that the aforementioned negative hardness results are in fact about the weakness of certain reductions, and that it may be the case that MCSP is NP-hard under, say, general randomized polynomial-time reductions.

We seem to be very far from being able to prove the NP-hardness of MCSP. If we cannot prove that MCSP is as hard as SAT, can we find other evidence that MCSP is indeed a hard problem, or at least that it will be difficult to design an efficient algorithm for it?

One possible kind of evidence that MCSP may be "almost as hard as" SAT would be to show that many known complexity-theoretic statements that use the SAT oracle will remain true when the SAT oracle is replaced with the MCSP oracle, i.e., that *the power of the* MCSP *oracle is often as good as that of* SAT. This is the research direction pursued in the present paper.

1.1 Our results

While, for simplicity, we state our results below for MCSP, in most of our results, MCSP could be replaced with any other natural property in the sense of Razborov and Rudich [41] (having largeness and usefulness, but with oracle access replacing constructivity). Roughly, our results are of three kinds:

- circuit lower bounds for randomized complexity classes with MCSP oracle,
- relations between Indistinguishability Obfuscation (IO) and MCSP, and
- hardness results for relativized versions of MCSP under randomized reductions.

We provide a more detailed description of our results next.

1.1.1 Conditional collapses

Below, the notation $\mathsf{SIZE}[s]$ denotes the class of Boolean functions computable by size s Boolean circuits.

▶ **Theorem 1.** *Let* $\Gamma \in \left\{\oplus\mathsf{P}, \mathsf{P}^{\#\mathsf{P}}, \mathsf{PSPACE}, \mathsf{EXP}, \mathsf{NEXP}, \mathsf{EXP}^{\mathsf{NP}}\right\}$. *If* $\Gamma \subseteq \mathsf{P/poly}$, *then* $\Gamma \subseteq \mathsf{ZPP}^{\mathsf{MCSP}}$.

1.1.1.1 Interpretation

The results of [36], [8], [25] and [15] (building upon [31]) imply collapse theorems for the classes $\mathsf{P}^{\#\mathsf{P}}$, PSPACE and EXP, NEXP, $\mathsf{EXP}^{\mathsf{NP}}$, respectively. More specifically, they show that if any of the above classes has polynomial size Boolean circuits, then the corresponding class collapses to MA, which is known to be contained in $\mathsf{ZPP}^{\mathsf{NP}}$ [7, 20]. Our Theorem 1 shows that the power of the MCSP oracle is sufficient for these conditional collapses.

As it is also known that $\mathsf{MA} \subseteq \mathsf{NP}^{\mathsf{MCSP}}$ (see, e.g., [2]), the conditional collapses to $\mathsf{NP}^{\mathsf{MCSP}}$ are immediate. Our Theorem 1 strengthens these collapses to the potentially smaller class $\mathsf{ZPP}^{\mathsf{MCSP}}$.

Finally, we also interpret Theorem 1 as follows: *A proof that* MCSP *is* not NP*-hard (or even* #P*-hard) under Turing* ZPP*-reductions would imply that* $\mathsf{P}^{\#\mathsf{P}} \not\subseteq \mathsf{P/poly}$.

1.1.2 Circuit lower bounds

Given the collapse theorems above, we get fixed-polynomial and super-polynomial lower bounds for randomized polynomial and exponential time, respectively. The extra bit of advice in the case of randomized polynomial time comes to accommodate the need to keep the promise of bounded error (the same problem arises in [10, 19, 37, 42, 49]). Alternatively, we can consider the corresponding class of promise problems (i.e., prZPP).

▶ **Theorem 2.** *We have the following:*
1. $\mathsf{ZPP}^{\mathsf{MCSP}}/1 \not\subseteq \mathsf{SIZE}[n^k]$ *and* $\mathsf{prZPP}^{\mathsf{MCSP}} \not\subseteq \mathsf{SIZE}[n^k]$*, for all* $k \in \mathbb{N}$*.*
2. $\mathsf{ZPEXP}^{\mathsf{MCSP}} \not\subseteq \mathsf{P/poly}$*.*

1.1.2.1 Interpretation

It is known that $\mathsf{MA\text{-}EXP} \not\subseteq \mathsf{P/poly}$ [14]. By padding, we get that $\mathsf{MA\text{-}EXP} \subseteq \mathsf{ZPEXP}^{\mathsf{NP}}$ (using $\mathsf{MA} \subseteq \mathsf{ZPP}^{\mathsf{NP}}$ [7, 20]), and hence $\mathsf{ZPEXP}^{\mathsf{NP}} \not\subseteq \mathsf{P/poly}$. Theorem 2 (item 2) shows that the MCSP oracle can replace the SAT oracle in that latter circuit lower bound.

1.1.2.2 Consequences for natural properties

The above result still holds if we relax the MCSP oracle to a natural property strongly useful against $\mathsf{P/poly}$ (see Theorem 41 for more details). Combining this result with Lemma 25, we obtain that PAC learning algorithms imply fixed-polynomial lower bounds against $\mathsf{BPP}/1$ and super polynomial lower bounds again BPEXP. These bounds match the results of [49] and [18, 32], respectively (see Corollary 26 for more details). In this sense, our *unconditional* lower bounds generalize the *conditional* lower bounds of [49] and [18, 32]. Indeed, our result is obtained by extending the techniques of [18, 32, 49].

The following theorem should be contrasted with a result from [25] saying that the existence of a P-natural property (even *without* the largeness condition) that is useful against $\mathsf{P/poly}$ would imply that $\mathsf{NEXP} \not\subseteq \mathsf{P/poly}$. *With* the largeness condition, the circuit lower bound can be shown to hold for the potentially smaller uniform complexity class ZPEXP. This theorem is an immediate consequence of Theorem 2, item (2).

▶ **Theorem 3.** *Suppose there is a strongly useful* ZPP*-natural property. Then* $\mathsf{ZPEXP} \not\subseteq \mathsf{P/poly}$*.*

▶ Remark. The conclusion of Theorem 3 still holds if we assume a natural property with only weakly-exponential usefulness, $2^{n^{\Omega(1)}}$.

▶ **Corollary 4.** *If there is a* ZPP*-natural property that is weakly-exponentially useful against* ACC^0 *circuits, then* $\mathsf{ZPEXP} \not\subseteq \mathsf{ACC}^0$*.* [1]

1.1.3 Obfuscation

We also relate the powers of MCSP and SAT to the existence of indistinguishability obfuscators (IO) [11]. Roughly speaking, an IO is an efficient randomized procedure that maps circuits to circuits, preserving the circuit input-output functionality but in an "unintelligible" manner. Indeed, applying the IO to any two functionally equivalent circuits of the same size yields two indistinguishable distributions on circuits (see Definition 27 for more details). We show the following.

[1] The result that P-natural properties against sub-exponential size circuits yield ZPEXP lower bounds was also obtained in independent work by Igor Oliveira and Rahul Santhanam [40].

▶ **Theorem 5.** *Let $\mathcal{A}$ denote the class of randomized polynomial-time algorithms with* MCSP *oracle. If there exists an $\mathcal{A}$-indistinguishable obfuscator* IO *then* $\mathsf{NP} \subseteq \mathsf{ZPP}^{\mathsf{MCSP}}$.

▶ **Corollary 6.** *Suppose a computational obfuscator* IO *exists. Then* $\mathsf{MCSP} \in \mathsf{ZPP}$ *iff* $\mathsf{NP} = \mathsf{ZPP}$.

Proof. If $\mathsf{NP} = \mathsf{ZPP}$ then, clearly, $\mathsf{MCSP} \in \mathsf{ZPP}$. For the other direction, if $\mathsf{MCSP} \in \mathsf{ZPP}$ then IO is also an $\mathcal{A}$-indistinguishable obfuscator. Therefore, $\mathsf{NP} \subseteq \mathsf{ZPP}^{\mathsf{MCSP}} = \mathsf{ZPP}^{\mathsf{ZPP}} = \mathsf{ZPP}$. ◀

1.1.3.1 Interpretation

Corollary 6 says that, under a cryptographic assumption that computational IO exists, the computational powers of SAT and MCSP are the same in the sense that a ZPP algorithm for MCSP is as good as a ZPP algorithm for SAT.

1.1.4 Hardness of relativized versions of MCSP

We consider the relativized version of MCSP relative to an oracle A, denoted MCSP^A, which asks to determine the minimum circuit size for a given Boolean function (given by its truth table) where the circuit is allowed to use A-oracle gates. It is shown by [2] that every language in PSPACE is reducible to $\mathsf{MCSP}^{\mathsf{PSPACE}}$ via ZPP-reductions. We use different techniques to re-prove this result, as well as obtain a few new results along the same lines. (Below $C_k\mathsf{P}$ is the kth level of the counting hierarchy, CH, where $C_1\mathsf{P} = \mathsf{PP}$, and $C_{k+1}\mathsf{P} = \mathsf{PP}^{C_k\mathsf{P}}$, for all $k \geq 1$.)

▶ **Theorem 7.**
1. $\mathsf{PSPACE} \subseteq \mathsf{ZPP}^{\mathsf{MCSP}^{\mathsf{PSPACE}}}$ *[2]*
2. $\oplus\mathsf{P} \subseteq \mathsf{ZPP}^{\mathsf{MCSP}^{\oplus\mathsf{P}}}$
3. $\mathsf{P}^{\#\mathsf{P}} \subseteq \mathsf{BPP}^{\mathsf{MCSP}^{\#\mathsf{P}}}$
4. $\mathsf{PP} \subseteq \mathsf{BPP}^{\mathsf{MCSP}^{\mathsf{PP}}}$. *Moreover, for $k \geq 2$: $C_k\mathsf{P} \subseteq C_{k-1}\mathsf{P}^{\mathsf{MCSP}^{\mathsf{PP}}}$.*

1.1.4.1 Interpretation

All of the inclusions of Theorem 7 become trivial if one replaces the relativized MCSP problem with the relativized SAT problem (or even just some relativized P-complete problem), since we have trivially that, e.g., $\mathsf{PSPACE} \subseteq \mathsf{P}^{\mathsf{PSPACE}}$. Theorem 7 says that the circuit minimization problem for circuits with A-oracle gates (for certain kinds of oracles) is at least as hard as the evaluation problem for A, under sufficiently powerful (randomized) reductions.

In [23], Hirahara and Watanabe defined the notion of oracle-independent randomized reductions and initiated a study of the set of languages that are reducible in randomized polynomial time to MCSP^B for every B. As a part of their study, they showed that $\bigcap_B \mathsf{BPP}^{\mathsf{MCSP}^B[1]} \subseteq \mathsf{AM} \cap \mathsf{coAM}$; this implies that NP-hardness of MCSP cannot be established via oracle-independent reductions unless the polynomial hierarchy collapses. We show circuit lower bounds for the class $\bigcap_B \mathsf{BPP}^{\mathsf{MCSP}^B}$.

▶ **Theorem 8.** *We have that* $\bigcap\limits_B \mathsf{BPP}^{\mathsf{MCSP}^B}/1 \not\subseteq \mathsf{SIZE}[n^k]$ *and* $\bigcap\limits_B \mathsf{prBPP}^{\mathsf{MCSP}^B} \not\subseteq \mathsf{SIZE}(n^k)$, *for all $k \in \mathbb{N}$, and that* $\bigcap\limits_B \mathsf{BPEXP}^{\mathsf{MCSP}^B} \not\subseteq \mathsf{P/poly}$.

1.2 Our techniques

We rely on the result of [16] showing that natural properties useful against a (sufficiently powerful) circuit class $\mathcal{C}$ yield learning algorithms (under the uniform distribution, with membership queries) for the same circuit class. We note that this result relativizes in the following sense: if we have a natural property useful against circuits with L oracle gates (say, MCSP^L), for some language L, then we can approximately learn L, with the hypotheses being circuits with MCSP^L oracle gates. If, in addition, this language L is both downward and random self-reducible, then we can learn L exactly, with the same type of MCSP^L oracle circuits, using the ideas of [27].

This allows us to prove, for example, that $\mathsf{P}^{\#\mathsf{P}} \subseteq \mathsf{BPP}^{\mathsf{MCSP}^{\#\mathsf{P}}}$, as $\#\mathsf{P}$ has a complete problem (the permanent) that is well-known to be both downward and random self-reducible. We show that $\oplus\mathsf{P}$ also has such a complete problem (building upon [45]), getting the inclusion $\oplus\mathsf{P} \subseteq \mathsf{BPP}^{\mathsf{MCSP}^{\oplus\mathsf{P}}}$. To get the stronger result that $\oplus\mathsf{P} \subseteq \mathsf{ZPP}^{\mathsf{MCSP}^{\oplus\mathsf{P}}}$, we use Toda's Theorem [43] and hardness-randomness tradeoffs of [26] to get rid of the two-sided error of our BPP reduction (similarly to the work of [28]).

Our circuit lower bounds are proved using similar ideas. For example, $\mathsf{ZPEXP}^{\mathsf{MCSP}} \not\subseteq \mathsf{P/poly}$ is argued as follows. If $\mathsf{PSPACE} \not\subseteq \mathsf{P/poly}$, we are done (as $\mathsf{PSPACE} \subseteq \mathsf{EXP}$). Assuming $\mathsf{PSPACE} \subseteq \mathsf{P/poly}$, we get that $\mathsf{PSPACE} \subseteq \mathsf{ZPP}^{\mathsf{MCSP}}$, using the fact that PSPACE contains a complete problem that is both downward and random self-reducible [45], and that $\mathsf{MCSP}^{\mathsf{PSPACE}} \in \mathsf{PSPACE} \subseteq \mathsf{P/poly}$. The circuit lower bound then follows by a translation argument, as we get that $\mathsf{EXPSPACE} \subseteq \mathsf{ZPEXP}^{\mathsf{MCSP}}$ and $\mathsf{EXPSPACE}$ is known to contain languages of maximal circuit complexity (by a simple diagonalization argument).

As another consequence of the results in [16], we get the following.

▶ **Theorem 9.** *For any language B, $n \in \mathbb{N}$ and $\delta > 0$, there exists a MCSP^B-oracle circuit C of size $\mathsf{poly}(n, 1/\delta)$ that is $1-\delta$ close to $B|_n$. If, in addition, B is self-correctable then B has polynomial size MCSP^B-oracle circuits.*

▶ **Theorem 10.** *Let B be a language such that PSPACE^B has polynomial size B-oracle circuits. Then B has polynomial-size MCSP^B-oracle circuits.* [2]

For the indistinguishability related results, we combine ideas from [21, 35] with a result from [2]. Let $\perp_s$ denote a canonical circuit of size s that outputs $'0'$ on every input. Let $\mathcal{A}$ denote the class of randomized polynomial-time algorithms with MCSP oracle. Given an $\mathcal{A}$-indistinguishable obfuscator IO, we consider the function $f_s(r) = \mathsf{IO}(\perp_s, r)$, where r is a random string. Observe that for any s, the function $f_s(r)$ is computable in time polynomial in $|r|$. We then apply a result of [2] that allows us to find preimages of such functions with probability $1/\mathsf{poly}(n)$.

Given a circuit C of size s, we first compute an obfuscation of C, $\hat{C} = \mathsf{IO}(C, r)$, (for a random r). Next, we (attempt to) find a preimage r' of $\hat{C}$. That is, r' such that $\mathsf{IO}(\perp_s, r') = \hat{C}$. We accept if and only if r' is indeed a preimage. That is, if and only if $\mathsf{IO}(\perp_s, r') = \hat{C}$.

We observe the following:

- If $C = \perp_s$ then the algorithm will accept with probability $1/\mathsf{poly}(n)$.

[2] In [2], the same outcome was achieved under a stronger assumption that $\mathsf{PSPACE}^B \subseteq \mathsf{P}^B$. We note our result is not a mere syntactical improvement, as there are numerous languages B for which $\mathsf{PSPACE}^B \subseteq \mathsf{P^B/poly}$ yet $\mathsf{PSPACE}^B \neq \mathsf{P}^B$; see Appendix C for more details. While we suspect that the consequent of the theorem holds unconditionally, we note that the precondition statement of the theorem cannot be improved further since Lemma 19 implies that, for every language B, the class PSPACE^B does not have fixed-polynomial size B-oracle circuits.

- If C is satisfiable then by the correctness requirement of IO (Requirement 2) for all r, r': $\mathsf{IO}(C, r) \neq \mathsf{IO}(\perp_s, r')$. Therefore, the algorithm will always reject.
- Finally, if C is an *unsatisfiable* circuit of size s, then by the indistinguishability requirement (Requirement 3) the algorithm cannot distinguish between the obfuscation of $\perp_s$ and the obfuscation of C. Hence, the algorithm will accept with probability about $1/\mathsf{poly}(n)$.

  Overall, we obtain that $\overline{\mathsf{SAT}} \in \mathsf{RP}^{\mathsf{MCSP}}$.

1.2.0.1 Remainder of the paper

We give basic definitions and notation in Section 2. In Section 3, we prove our main results (Theorems 1 - 8) which show new collapse results as well as new circuit lower bounds for uniform complexity classes with oracle access to (relativized) MCSP. In fact, we prove somewhat stronger results (Theorems 40 and 41) which apply to the more general type of oracles: strongly useful natural properties. We prove our IO-related result, Theorem 5, in Section 3.3. Next, in Section 3.4, we prove our results about reductions to the problem MCSP^B, for various languages B. Specifically, we give such reductions for several complexity classes (Theorem 7), and also show that every language B can be approximated by "small" Boolean circuits containing MCSP^B oracle gates (Theorem 9). Finally, we show that under certain conditions, a language B can be computed exactly by "small" Boolean circuits containing MCSP^B oracle gates (rather than just approximated) (Theorem 10). We conclude with some open questions in Section 4. Some of the proofs (e.g., our proof that $\oplus\mathsf{P}$ has a complete problem that is both downward and random self-reducible) are given in the appendix.

2 Preliminaries

2.1 Basics

A function $\mathrm{negl}(n)$ is *negligible* if for any $k \in \mathbb{N}$ there exists $n_k \in \mathbb{N}$ such that for all $n > n_k$, $\mathrm{negl}(n) < 1/n^k$.

For Boolean functions $f, g : \{0,1\}^n \to \{0,1\}$, we define the *relative distance* $\Delta(f, g)$ to be the fraction of inputs $x \in \{0,1\}^n$ where $f(x) \neq g(x)$. For $\varepsilon \geq 0$, we say that f is ε*-close* to g if $\Delta(f, g) \leq \varepsilon$, otherwise we say that f is ε*-far* from g.

Let $L \subseteq \{0,1\}^*$ be a language. We denote by $L|_n$ the set of the strings of length n in L. We will associate a language L with a corresponding Boolean function in the natural way: $L(x) = 1 \iff x \in L$. We say that L has circuits of size $a(n)$ and denote it by $L \in \mathsf{SIZE}(a(n))$ if for every $n \in \mathbb{N}$ the function $L|_n$ can be computed by a Boolean circuit of size $\mathcal{O}(a(n))$. The *circuit complexity* $\mathsf{s}_L(n)$ of L at length n is the smallest integer t such that there is a Boolean circuit of size t that computes $L|_n$. We similarly define $\mathsf{s}_L^B(n)$ to be the circuit complexity of L with respect to B-oracle circuits and $\mathsf{SIZE}^B(a(n))$. We have the following easy observation.

▶ **Observation 11.** *Let A, B be two languages. Suppose that $A \in \mathsf{SIZE}^B(n^k)$ for some $k \in \mathbb{N}$. Then for every language L: $\mathsf{s}_L^B(n) \leq \mathsf{s}_L^A(n)^{k+1}$.*

A promise problem is a relaxation of a language, defined as follows.

▶ **Definition 12** (Promise Problems). $\Pi = (\Pi_{YES}, \Pi_{NO})$ is a *promise problem* if $\Pi_{YES} \cap \Pi_{NO} = \emptyset$. We say that a language L is *consistent* with Π iff $x \in \Pi_{YES} \implies x \in L$ and $x \in \Pi_{NO} \implies x \notin L$. The containment of L outside of $\Pi_{YES} \dot\cup \Pi_{NO}$ can be arbitrary. We say that a set of languages Γ is consistent with a set of promise problems Λ iff for every $\Pi \in \Lambda$ there is $L \in \Gamma$ that is consistent with Π.

We refer the reader to [6] for the definitions of standard complexity classes such as P, ZPP, RP, BPP, NP, MA, PSPACE, etc. We say that a language $L \in \mathsf{BPP}/1$ if L can be decided by a BPP machine with an auxiliary *advice* bit b_n for each input of length n; note that given the complement advice bit $\bar{b}_n$, the machine is not guaranteed to be a BPP machine (i.e., may not have bounded away acceptance and rejection probabilities on all inputs of length n). We define $\mathsf{ZPP}/1$ in a similar fashion.

We define a family of natural problems complete for prBPP relative to any oracle.

▶ **Definition 13** (Circuit Approximation)**.** For a language B, define the following prBPP^B-complete problem: $\mathrm{CA}^B \triangleq (\mathrm{CA}^B_{YES}, \mathrm{CA}^B_{NO})$, where

$$\mathrm{CA}^B_{YES} = \left\{C \text{ is a } B\text{-oracle circuit } \mid \Delta(C, \bar{0}) \geq 3/4\right\},$$

and

$$\mathrm{CA}^B_{NO} = \left\{C \text{ is a } B\text{-oracle circuit } \mid \Delta(C, \bar{0}) \leq 1/4\right\}.$$

To prove lower bounds against randomized classes with one bit of advice, we shall rely on the following definitions (and their extensions) from [42, 49].

▶ **Definition 14** (Padded Languages)**.** Let L be a language. For $k \in \mathbb{N}$ we define the padded version of L, denoted L'_k, to consist of the strings $1^m x$ satisfying the following: (1) m is power of 2; (2) $0 < r \triangleq |x| \leq m$; (3) $x \in L$; and (4) $\mathsf{s}_L(r) \leq m^{2k}$.

The main property of the padded languages is that, for every L, sufficiently small circuits for L'_k can be used to construct small circuits for L.

▶ **Lemma 15** ([42, 49])**.** *Let $k \in \mathbb{N}$. Suppose $L'_k \in \mathsf{SIZE}[n^k]$. Then $\mathsf{s}_L(n) = \mathcal{O}(n^{2k})$.*

The next lemma is implicit in [49]. We provide the proof for completeness.

▶ **Lemma 16.** *Let $\mathcal{R}$ be a strongly useful natural property and let L be a downward self-reducible and self-correctable language. Then, for all $k \in \mathbb{N}$, we have $L'_k \in \mathsf{BPP}^{\mathcal{R}}/1$.*

Proof. Let $y = 1^m x$ be an input for L'_k. Conditions 1 and 2 of Definition 14 can be checked easily. As y has a unique interpretation, we use the advice bit to determine whether $\mathsf{s}_L(|x|) \leq m^{2k}$. If the advice bit is 0 (i.e "no") we reject. Otherwise, we apply Lemma 39 with $t = m^{2k}$ to decide if $x \in L$. ◀

We also need the following result that shows that a lower bound on $\mathsf{ZPP}/1$ carries over to prZPP.

▶ **Lemma 17** ([42])**.** *For every circuit function $u(n)$, if $\mathsf{ZPP}/1 \not\subseteq \mathsf{SIZE}[u(n)]$, then $\mathsf{prZPP} \not\subseteq \mathsf{SIZE}[u(n)]$.*

Finally, we need the following collapse results and a simple circuit lower bound against PSPACE.

▶ **Lemma 18** ([8, 25, 15])**.** *If $\Gamma \in \left\{\mathsf{EXP}, \mathsf{NEXP}, \mathsf{EXP}^{\mathsf{NP}}\right\}$ is in P/poly, then $\Gamma = \mathsf{MA}$.*

▶ **Lemma 19** ([29])**.** *For any language B and $k \in \mathbb{N}$, $\mathsf{PSPACE}^B \not\subseteq \mathsf{SIZE}^B[n^k]$. More generally, for every function $s(n) = \mathcal{O}(2^n)$, $\mathsf{DSPACE}^B(\mathsf{poly}(s(n))) \not\subseteq \mathsf{SIZE}^B[s(n)]$.*

▶ **Lemma 20** (Folklore)**.** *For any oracle A: $\mathsf{NP} \subseteq \mathsf{BPP}^A \implies \mathsf{NP} \subseteq \mathsf{RP}^A$.*

2.2 Derandomization from hardness

We recall the celebrated hardness-randomness tradeoff.

▶ **Lemma 21** ([39, 9, 26, 46, 33]). *There is a polynomial-time computable oracle predicate $M^B(x, y)$ and a constant $\ell \in \mathbb{N}$ such that the following holds for every language B and $s \in \mathbb{N}$. If $tt \in \{0,1\}^{2^m}$ is a string that represents the truth table of an m-variate Boolean function f which requires B-oracle circuits of size s^ℓ, then, for all s-size B-oracle circuits C, $M^B(C, tt)$ is consistent with* CA^B.

The non-relativized version of this result was used in [28] to show that $\mathsf{BPP} \subseteq \mathsf{ZPP}^{\mathsf{MCSP}}$. We use the relativized version to show that under certain assumptions $\mathsf{BPP}^A = \mathsf{ZPP}^A$.

▶ **Lemma 22.** *Let A, B be any languages such that: (1) $A \in \mathsf{P}^{\mathsf{B}}/\mathsf{poly}$, and (2) $\mathsf{MCSP}^B \in \mathsf{ZPP}^A$. Then $\mathsf{BPP}^A = \mathsf{ZPP}^A$.*

Proof. By definition, $\mathsf{ZPP}^A \subseteq \mathsf{BPP}^A$. For the second direction, let $L \in \mathsf{BPP}^A$. Then for each n there exists an A-oracle circuit $C(w, r)$ of size $\mathsf{poly}(n)$ such that $x \in L \iff C(x, \cdot) \in \mathrm{CA}^A$. We now describe a machine that decides L: "For $m = \mathcal{O}(\log n)$ pick a truth table $tt \in \{0,1\}^{2^m}$ at random. If tt has B-circuits of size less than $2^{m/4}$ return "?" (using an oracle to MCSP^B). Otherwise, run $M^A(C(x, \cdot), tt)$ and answer the same (using M^A from Lemma 21)."

By counting arguments, a random function requires exponential size circuits w.h.p. Therefore, the algorithm will output "?" extremely rarely. By Observation 11 tt requires A-oracle circuits of size $2^{\Omega(m)} = n^{\Omega(1)}$. Consequently, the correctness of the algorithm follows from Lemma 21. As described, the algorithm can be implemented in $\mathsf{ZPP}^{A,\mathsf{MCSP}^B}$. By the preconditions, $\mathsf{ZPP}^{A,\mathsf{MCSP}^B} \subseteq \mathsf{ZPP}^{A,\mathsf{ZPP}^A} = \mathsf{ZPP}^A$ due to the self-lowness of ZPP. ◀

2.3 Natural properties, PAC learning and MCSP

We first define natural properties.

▶ **Definition 23** (Natural Property [41]). Let $\mathcal{C}$ be a circuit class and Γ be complexity classes. We say that a property $\mathcal{R}$ is Γ-*natural with density* δ_n and *useful* against $\mathcal{C}$ if the following holds:

1. **Constructivity:** Given a binary string $tt \in \{0,1\}^{2^m}$, $tt \in \mathcal{R}$ can be decided in Γ.
2. **Largeness:** For all n, $\mathcal{R}$ contains at least a δ_n fraction of all 2^n binary strings, representing n-variate Boolean functions.
3. **Usefulness:** For every Boolean function family $\{f_n\}_{n\geq 0}$, where f_n is a function on n variables, such that $\{tt \mid tt \text{ is a truth table of some } f_n\} \subseteq \mathcal{R}$ for almost all n, we have that $\{f_n\} \notin \Lambda$ for almost all n.

We say that $\mathcal{R}$ is *strongly useful* if there exists $a \in \mathbb{N}$ such that $\mathcal{R}$ is useful against $\mathsf{SIZE}[2^{an}]$ and has density $\delta_n \geq 2^{-an}$.

Considering $\mathcal{R}$ as an oracle allows us to "ignore" its complexity. In addition, if $\mathcal{R}$ is a strongly useful property, then, as observed in [16, Lemma 2.7], there exists another strongly useful property $\mathcal{R}' \in \mathsf{P}^{\mathcal{R}}$ with density $\delta_n \geq 1/2$. Therefore, when considering a strongly useful property as an oracle we can assume w.l.o.g that it has density $\delta_n \geq 1/2$.

Observe that MCSP yields a strongly useful natural property. Often, the only requirement from an MCSP oracle is to "serve" as a strongly useful natural property. Consequently, the oracle can be relaxed. The following can be shown along the lines of the proof of Lemma 22.

▶ **Lemma 24.** *Let $\mathcal{R}$ be a strongly useful natural property. Then* $\mathsf{BPP} \subseteq \mathsf{ZPP}^{\mathcal{R}}$. *If, in addition,* $\mathcal{R} \in \mathsf{P/poly}$ *then* $\mathsf{BPP}^{\mathcal{R}} = \mathsf{ZPP}^{\mathcal{R}}$.

Recall Valiant's PAC learning model [48]. We have a (computationally bounded) learner that is given a set of samples of the form $(\bar{x}, f(\bar{x}))$ from some fixed function $f \in \mathcal{C}$, where $\bar{x}$ is chosen according to some unknown distribution D. Given $\varepsilon > 0$ and $\delta > 0$, the learner's goal is to output, with probability $1 - \varepsilon$ a hypothesis $\hat{f}$ such that $\hat{f}$ is a $1 - \delta$ close to f under D. We say that a function class $\mathcal{C}$ is PAC *learnable* if there exists a learner which given any $f \in \mathcal{C}$, $\varepsilon > 0$ and $\delta > 0$ in time polynomial in $n, 1/\varepsilon, 1/\delta, |f|$ outputs a hypothesis as required. In a more general model, the learner is allowed membership queries (as in the exact learning model). In this case, we say that $\mathcal{C}$ is PAC *learnable* with *membership queries.*

In [16] it was shown that natural properties yield efficient learning algorithms. Specifically, a BPP-natural property that is strongly useful against a circuit class $\mathcal{C}$ implies that $\mathcal{C}$ is PAC learnable under the uniform distribution, with membership queries (see Section 37 for more details). Here we show that the converse holds as well: if $\mathcal{C}$ is PAC learnable under the uniform distribution, with membership queries then there is a BPP-natural property that is strongly useful against $\mathcal{C}$.

▶ **Lemma 25.** *Let $\mathcal{C}$ be a circuit class. If $\mathcal{C}$ is* PAC *learnable under the uniform distribution, with membership queries, then there exists a* BPP*-natural property that is strongly useful against $\mathcal{C}$.*

The proof goes along the lines of Theorem 3 from [49], where it is shown how to turn an efficient randomized exact learner $\mathcal{A}$ for a circuit class $\mathcal{C}$ into a P/poly-natural property strongly useful against $\mathcal{C}$. Combined with Theorem 2, we obtain a somewhat different proof for the conditional lowers bounds of [49] and [18, 32].

▶ **Corollary 26.** *For every circuit class $\mathcal{C}$, if $\mathcal{C}$ is* PAC *learnable under the uniform distribution, with membership queries then:*

1. $\mathsf{BPP}/1 \not\subseteq \mathcal{C}\text{-}\mathsf{SIZE}[n^k]$ *and* $\mathsf{prBPP} \not\subseteq \mathcal{C}\text{-}\mathsf{SIZE}[n^k]$, *for all $k \in \mathbb{N}$ [49] .*
2. $\mathsf{BPEXP} \not\subseteq \mathcal{C}\text{-}\mathsf{SIZE}[\mathsf{poly}]$ *[18, 32].*

2.4 Obfuscation

In this section we define the notion of an Indistinguishability Obfuscator.

▶ **Definition 27** (Indistinguishability Obfuscator [11])**.** Let $\mathcal{A}$ be a class of algorithms. We say that a procedure IO is an $\mathcal{A}$-*Indistinguishability Obfuscator* for a circuit class $\mathcal{C}$ if the following holds:

1. **Correctness:** For every circuit $C \in \mathcal{C}$ and for all inputs x, $C(x) = \mathsf{IO}(C)(x)$.
2. **Polynomial slowdown:** There exists $k \in \mathbb{N}$ s.t. for every circuit $C \in \mathcal{C}$, $|\mathsf{IO}(C)| \leq |C|^k$.
3. **Indistinguishability:** For all pairs of circuits $C_1, C_2 \in \mathcal{C}$ that compute the same function, if $|C_1| = |C_2| = s$ then the distributions of $\mathsf{IO}(C_1)$ and $\mathsf{IO}(C_2)$ are indistinguishable by any algorithm $A \in \mathcal{A}$. More precisely, there is a negligible function $\mathrm{negl}(n)$ such that for any algorithm $A \in \mathcal{A}$:

$$|\Pr[A(\mathsf{IO}(C_1)) = 1] - \Pr[A(\mathsf{IO}(C_2)) = 1]| \leq \mathrm{negl}(s).$$

In particular, when $\mathcal{A}$ the class of **randomized polynomial-time algorithms**, we call such IO a *Computational Obfuscator.*

2.5 Downward self-reducible and self-correctable languages

▶ **Definition 28.** We say that a language L is *downward self-reducible* if there is a deterministic polynomial-time algorithm $\mathsf{COMPUTE}$ such that, for all $n \geq 1$, $\mathsf{COMPUTE}^{L|_{n-1}} = L|_n$. In other words, $\mathsf{COMPUTE}$ efficiently computes L on inputs of size n given oracle access to a procedure that computes L on inputs of size $n-1$.
We say that a language L is *self-correctable*[3] if there is a probabilistic polynomial-time algorithm $\mathsf{CORRECT}$ such that, for any $n \in \mathbb{N}$ and a Boolean function $f : \{0,1\}^n \to \{0,1\}$ it holds that if $\Delta(f, L|_n) \leq 1/n$ then for all $\bar{x} \in \{0,1\}^n$: $\Pr[\mathsf{CORRECT}^f(\bar{x}) \neq L|_n(\bar{x})] \leq 1/\mathsf{poly}(n)$.

Several complexity classes have complete problems that are both downward self-reducible and self-correctable.[4]

▶ **Lemma 29** ([47, 12, 36, 27]). *There exists a downward self-reducible and self-correctable $\#\mathsf{P}$-complete language L_{perm}.*

▶ **Lemma 30** ([45]). *There is a downward self-reducible and self-correctable PSPACE-complete language L_{PSPACE}.*

Using similar ideas as in [45], we also show the following; see Appendix A for the proof.

▶ **Lemma 31.** *There is a downward self-reducible and self-correctable $\oplus\mathsf{P}$-complete language $L_{\oplus\mathsf{P}}$.*

To handle a larger family of languages, we generalize the notion of self-correctability.

▶ **Definition 32.** A language L is $(\varepsilon(n), A)$*-correctable* if there are a polynomial $r(n)$ and a randomized polynomial-time algorithm $\mathsf{CORRECT}$ such that, for all $n \in \mathbb{N}$ and $f\colon \{0,1\}^{r(n)} \to \{0,1\}$, if $\Delta\left(f, A|_{r(n)}\right) \leq \varepsilon(n)$, then, for all $\bar{x} \in \{0,1\}^n$, $\Pr\left[\mathsf{CORRECT}^f(\bar{x}) \neq L|_n(\bar{x})\right] \leq 1/\mathsf{poly}(n)$.

In other words, there is a randomized polynomial-time algorithm that can decide $L|_n$ given an oracle to a function that approximates a $A|_{r(n)}$. Self-correctability is special case when $\varepsilon = 1/n$, $A = L$ and $r(n) = n$. The following is immediate using Adleman's result [1]:

▶ **Lemma 33.** *Let L be a $(\varepsilon(n), A)$-correctable language with $r(n)$, and let B be a language. Suppose C is an $r(n)$-variate B-oracle circuit of size s such that $\Delta(C, A|_{r(n)}) \leq \varepsilon(n)$, for some $n \in \mathbb{N}$. There exists a randomized polynomial-time algorithm that given C as input, outputs an n-variate B-oracle circuit C' of size $\mathsf{poly}(r(n), s)$ such that $C' \equiv L|_n$, w.h.p.*

In particular, this result implies that such C' exists for every C.
Klivans and van Melkebeek [33] show that any language L is $(\varepsilon(n), A)$-correctable for A computable in PSPACE with an oracle to L (by encoding the truth table of L with an appropriate error-correcting code).

▶ **Theorem 34.** *For any language L and $\varepsilon(n)$ there exist a language $A \in \mathsf{DSPACE}^L(n+1/\varepsilon(n))$ such that L is $(\varepsilon(n), A)$-correctable with $r(n) = \mathsf{poly}(n, 1/\varepsilon(n))$.*

[3] More generally, such languages are referred to as "random self-reducible" languages.
[4] It is not hard to see that every downward self-reducible language is computable in PSPACE. On the other hand, the results of [17] suggest that there cannot be self-correctable languages which are complete for any level of the polynomial hierarchy, unless the hierarchy collapses.

2.6 Learning downward self-reducible and self-correctable languages

The following lemma essentially shows that the PAC learnability for downward self-reducible and self-correctable languages implies exact learnability; a similar lemma also appeared in [40]. See the Appendix (Section B) for the proof.

▶ **Lemma 35.** *Let B be a language. Suppose Boolean circuits are* PAC *learnable using membership and B queries with hypotheses being B-oracle circuits. Suppose L is a downward self-reducible and self-correctable language. Then there is a randomized algorithm making oracle queries to B, that, given x and t, computes $L(x)$ with probability at least $1-1/\mathsf{poly}(|x|)$ in time $\mathsf{poly}(|x|,t)$, provided that $t \geq \mathsf{s}_L(|x|)$.*

3 The proofs

Our proofs will use the following.

▶ **Lemma 36** (Extension of Theorem 5.1 from [16])**.** *Let $\mathcal{R}$ be a natural property with density at least $1/5$, that is useful against $\mathsf{SIZE}[u(n)]$, for some size function $u(n)\colon \mathbb{N} \to \mathbb{N}$. Then there is a randomized algorithm that makes oracle queries to $\mathcal{R}$ such that, given $s \in \mathbb{N}$, oracle access to a function $f : \{0,1\}^n \to \{0,1\}$ computable by a Boolean circuit of size s, and $\delta > 0$, it produces in time $\mathsf{poly}(n, 1/\delta, 2^{u^{-1}(\mathsf{poly}(n,1/\delta,s))})$ an $\mathcal{R}$-oracle circuit C where $\Delta(C,f) \leq \delta$.*

▶ **Corollary 37.** *Let $\mathcal{R}$ be a strongly useful natural property. Then Boolean circuits are* PAC *learnable under the uniform distribution, using membership and $\mathcal{R}$ queries with hypotheses being $\mathcal{R}$-oracle circuits.*

▶ **Theorem 38.** *Boolean circuits are* PAC *learnable under the uniform distribution, using membership and* MCSP *queries with hypotheses being* MCSP*-oracle circuits.*

Combining Lemma 35 and Corollary 37, we get the following.

▶ **Lemma 39.** *Let $\mathcal{R}$ be a strongly useful natural property and let L be a downward self-reducible and self-correctable language. Then there is a randomized algorithm that makes oracle queries to $\mathcal{R}$, that given x and t computes $L(x)$ with probability at least $1-1/\mathsf{poly}(|x|)$ in time $\mathsf{poly}(|x|,t)$ provided that $t \geq \mathsf{s}_L(|x|)$.*

3.1 Conditional collapses

Theorem 1 follows as a corollary from the next, somewhat stronger, theorem.

▶ **Theorem 40.** *Let $\mathcal{R}$ be a strongly useful natural property, and furthermore let $\Gamma \in \left\{\oplus\mathsf{P}, \mathsf{P}^{\#\mathsf{P}}, \mathsf{PSPACE}, \mathsf{EXP}, \mathsf{NEXP}, \mathsf{EXP}^{\mathsf{NP}}\right\}$. Then, if $\Gamma \subseteq \mathsf{P/poly}$, then $\Gamma \subseteq \mathsf{BPP}^{\mathcal{R}}$. If, in addition, $\mathcal{R} \in \mathsf{PH}$ then $\Gamma \subseteq \mathsf{ZPP}^{\mathcal{R}}$.*

Proof. First, consider the case of Γ such that $\mathsf{PSPACE} \subseteq \Gamma$. For L_{PSPACE} from Lemma 30, we have that $\mathsf{s}_{L_{\mathsf{PSPACE}}}(n) = \mathcal{O}(n^k)$ for some $k \in \mathbb{N}$. By Lemma 39, given x, we can compute $L_{\mathsf{PSPACE}}(x)$ in randomized polynomial time given oracle to $\mathcal{R}$. Consequently, $\mathsf{PSPACE} \subseteq \mathsf{BPP}^{\mathcal{R}}$. By Lemma 18, we get $\Gamma = \mathsf{MA}$. Hence, we have $\Gamma \subseteq \mathsf{MA} \subseteq \mathsf{PSPACE} \subseteq \mathsf{BPP}^{\mathcal{R}}$. If, in addition, $\mathcal{R} \in \mathsf{PH}$ then $\mathcal{R} \in \Gamma \subseteq \mathsf{P/poly}$. By Lemma 24, $\mathsf{BPP}^{\mathcal{R}} \subseteq \mathsf{ZPP}^{\mathcal{R}}$.

For $\Gamma = \oplus\mathsf{P}$, we argue as before, using Lemma 31 instead of Lemma 30, to obtain that $\oplus\mathsf{P} \subseteq \mathsf{BPP}^{\mathcal{R}}$. If, in addition, $\mathcal{R} \in \mathsf{PH}$, then, by Toda's Theorem [43], $\mathsf{PH} \subseteq \mathsf{BPP}^{\oplus\mathsf{P}}$, and hence $\mathcal{R} \in \mathsf{BPP}^{\oplus\mathsf{P}} \subseteq \mathsf{P/poly}$. The rest of the argument follows as above.

For $\Gamma = \mathsf{P}^{\#\mathsf{P}}$, we argue as before using Lemma 29 instead of 30, with the additional observation that in this case the permanent has small Boolean circuits. The only difference is that in this case the function has multiple inputs. For more details we refer the reader to [27]. ◀

3.2 Circuit lower bounds for MCSP-oracle classes

Theorems 2 and 3 follow from the next theorem.

▶ **Theorem 41.** *For any strongly useful natural property $\mathcal{R}$ we have*
1. $\mathsf{ZPP}^{\mathcal{R}}/1 \not\subseteq \mathsf{SIZE}[n^k]$ *and* $\mathsf{prZPP}^{\mathcal{R}} \not\subseteq \mathsf{SIZE}[n^k]$ *for all* $k \in \mathbb{N}$*, and*
2. $\mathsf{ZPEXP}^{\mathcal{R}} \not\subseteq \mathsf{P/poly}$.

Proof. Assume w.l.o.g that $\mathcal{R} \in \mathsf{P/poly}$ (otherwise there is nothing to prove). Consider $L = L_{\mathsf{PSPACE}}$ from Lemma 30. As $L \in \mathsf{PSPACE} \subseteq \mathsf{EXP}$, by a translation argument there exists $d \geq 1$ such that $L \in \mathsf{SIZE}(2^{n^d})$. Therefore, $\mathsf{s}_L(n)$ is well-defined and in particular $\mathsf{s}_L(n) = \mathcal{O}(2^{n^d})$.

We first prove part (1) of the theorem. We focus on the class $\mathsf{ZPP}^{\mathcal{R}}/1$; the claim about prZPP will follow by Lemma 17. We consider two cases:

Case 1: $\mathsf{PSPACE} \subseteq \mathsf{P/poly}$. By Theorem 1 and Lemma 24, $\mathsf{PSPACE} \subseteq \mathsf{BPP}^{\mathcal{R}} \subseteq \mathsf{ZPP}^{\mathcal{R}}$. Hence, by Lemma 19, for all $k \in \mathbb{N} : \mathsf{ZPP}^{\mathcal{R}} \not\subseteq \mathsf{SIZE}[n^k]$.

Case 2: $\mathsf{PSPACE} \not\subseteq \mathsf{P/poly}$. As L is PSPACE-complete, we have that $L \notin \mathsf{P/poly}$. Assume towards contradiction that $\mathsf{BPP}^{\mathcal{R}}/1 \subseteq \mathsf{SIZE}[n^k]$, for some $k \in \mathbb{N}$. By Lemma 16, $L'_k \in \mathsf{SIZE}[n^k]$. And thus, by Lemma 15, $\mathsf{s}_L(n) = \mathcal{O}(n^{2k})$. This contradicts the assumption that $L \notin \mathsf{P/poly}$. As in Lemma 24, we obtain that, for all $k \in \mathbb{N}$, $\mathsf{ZPP}^{\mathcal{R}}/1 \not\subseteq \mathsf{SIZE}[n^k]$.

Part (2) of the theorem is also shown by considering two cases:

Case 1: $\mathsf{PSPACE} \subseteq \mathsf{P/poly}$. As above, $\mathsf{PSPACE} \subseteq \mathsf{ZPP}^{\mathcal{R}}$. By a translation argument, $\mathsf{EXPSPACE} \subseteq \mathsf{ZPEXP}^{\mathcal{R}}$. By Lemma 19, $\mathsf{ZPEXP}^{\mathcal{R}} \not\subseteq \mathsf{P/poly}$.

Case 2: $\mathsf{PSPACE} \not\subseteq \mathsf{P/poly}$. Since $\mathsf{PSPACE} \subseteq \mathsf{EXP} \subseteq \mathsf{ZPEXP}^{\mathcal{R}}$, the theorem follows. ◀

3.3 IO related results

We prove more general statements for strongly useful natural properties.

▶ **Theorem 42.** *Let $\mathcal{R}$ be a strongly useful natural property and let $\mathcal{A}$ denote the class of randomized polynomial-time algorithms with $\mathcal{R}$ oracle. If there exists an $\mathcal{A}$-indistinguishable obfuscator* IO *then* $\mathsf{NP} \subseteq \mathsf{ZPP}^{\mathcal{R}}$.

Before proving the Theorem we require the following result of [2] that allows to find preimages of polynomial-time computable functions[5].

▶ **Lemma 43** ([2]). *Let $\mathcal{R}$ be a strongly useful natural property. Let $f_y(x) = f(y,x)$ be a function computable uniformly in time polynomial in $|x|$. There exists a polynomial-time probabilistic oracle Turing machine M and $k \in \mathbb{N}$ such that for any n and y:*

$$\Pr_{|x|=n,t}\left[f_y\left(M^{\mathcal{R}}(y, f_y(x), t)\right) = f_y(x)\right] \geq 1/n^k$$

where x is chosen uniformly at random and t denotes the randomness of M.

[5] The original result in formulated in a slightly different terminology.

We now present the proof of Theorem 42.

Proof. Consider the function $f_C(r) = \mathsf{IO}(C, r)$, where C is a circuit and r is a random string. Observe that $f_C(r)$ is computable uniformly in time polynomial in $|r|$. By the Lemma above there exists a polynomial-time probabilistic oracle Turing machine M and $k \in \mathbb{N}$ such that for any circuit C:

$$\Pr_{r,t}\left[f_C\left(M^{\mathcal{R}}(C, \mathsf{IO}(C, r), t)\right) = f_C(r)\right] \geq 1/\left|r\right|^k$$

where r is chosen uniformly at random and t denotes the randomness of M.

We now describe a polynomial-time randomized Turing machine that decides SAT. For $s \in \mathbb{N}$, we denote by $\bot_s$ a canonical unsatisfiable circuit. Note that given s, $\bot_s$ can be computed uniformly in time polynomial in s. Given a circuit C as input:

1. Let $s = |C|$.
2. $\hat{C} = \mathsf{IO}(C, r)$ for r chosen uniformly at random.
3. Run $M^{\mathcal{R}}(\bot_s, \hat{C}, t)$ to obtain r'.
4. Accept if and only if $\mathsf{IO}(\bot_s, r') = \hat{C}$.

We observe the following:

- If $C = \bot_s$ then the algorithm will accept with probability $\geq 1/\left|r\right|^k$.
- If $C \in \mathsf{SAT}$ then by the correctness requirement of IO (Requirement 2) for all r, r': $\mathsf{IO}(\bot_s, r') \neq \mathsf{IO}(C, r)$. Therefore, the algorithm will always reject.
- Finally, if $C \in \overline{\mathsf{SAT}}$, then by the indistinguishability requirement (Requirement 3), the oracle machine $M^{\mathcal{R}}$ could not distinguish between the obfuscation of $\bot_s$ and the obfuscation of C. Hence, the algorithm will accept with probability $1/\left|r\right|^k - \mathrm{negl}(|r|)$. By repeating the algorithm, we obtain that $\overline{\mathsf{SAT}} \in \mathsf{RP}^{\mathcal{R}}$

By Lemma 20, $\mathsf{SAT} \in \mathsf{RP}^{\mathcal{R}}$ and hence $\mathsf{SAT} \in \mathsf{ZPP}^{\mathcal{R}}$. ◀

3.4 Hardness of relativized versions of MCSP

First we observe that, for every oracle B, there is a $\mathsf{P}^{\mathsf{MCSP}^B}$-natural property for B-oracle circuits. Combined with Lemma 36, this yields the following theorem along the lines of Theorem 38.

▶ **Theorem 44.** *For every oracle B the class of B-oracle circuits is* PAC *learnable under the uniform distribution, using membership and MCSP^B queries with hypotheses being MCSP^B-oracle circuits.*

▶ **Lemma 45.** *Let A, B be two oracles (languages) such that $A \in \mathsf{P}^{\mathsf{B}}/\mathsf{poly}$. Then:*

1. *For every $n \in \mathbb{N}$ and $\delta > 0$, there exists a MCSP^B-oracle circuit C of size $\mathsf{poly}(n, 1/\delta)$ such that $\Delta(C, A|_n) \leq \delta$.*
2. *If, in addition, A is self-correctable then $A \in \mathsf{P}^{\mathsf{MCSP}^{\mathsf{B}}}/\mathsf{poly}$.*
3. *If, in addition to the above, A is downward self-reducible, then $A \in \mathsf{BPP}^{\mathsf{MCSP}^B}$.*

Proof.

1. By the assumption, for every $n \in \mathbb{N}$ the function $A|_n(x)$ has a B-oracle circuit of size $\mathsf{poly}(n)$. Therefore, by Theorem 44, given oracle access to $A|_n$, the learning algorithm produces an MCSP^B-oracle circuit C of size $\mathsf{poly}(n, 1/\delta)$ such that $\Delta(C, B|_n) \leq \delta$.
2. Follows from Lemma 33.
3. Follows by combining Theorem 44 with Lemma 35. ◀

▶ Remark. In the lemma above, although the learning algorithm actually needs oracle access to $A|_n$ to produce C, in parts 1 and 2 we are only interested in mere existence. In part 3, on the other hand, we actually benefit from the learning algorithm.

We are now ready to give the proofs of Theorems 7–10. For convenience, we re-state them below.

▶ **Theorem 46** (Theorem 7 re-stated).
1. $\mathsf{PSPACE} \subseteq \mathsf{ZPP}^{\mathsf{MCSP}^{\mathsf{PSPACE}}}$ *[2]*
2. $\oplus\mathsf{P} \subseteq \mathsf{ZPP}^{\mathsf{MCSP}^{\oplus\mathsf{P}}}$
3. $\mathsf{P}^{\#\mathsf{P}} \subseteq \mathsf{BPP}^{\mathsf{MCSP}^{\#\mathsf{P}}}$
4. $\mathsf{PP} \subseteq \mathsf{BPP}^{\mathsf{MCSP}^{\mathsf{PP}}}$. *Moreover, for $k \geq 2$: $C_k\mathsf{P} \subseteq C_{k-1}\mathsf{P}^{\mathsf{MCSP}^{\mathsf{PP}}}$.*

Proof.
1. Consider any PSPACE-complete language B. Let L_{PSPACE} be the language from Lemma 30. By Lemma 45 (3), $L_{\mathsf{PSPACE}} \in \mathsf{BPP}^{\mathsf{MCSP}^B}$, and hence $\mathsf{PSPACE} \subseteq \mathsf{BPP}^{\mathsf{MCSP}^B}$. Observe that
$$\mathsf{MCSP}^B \in \mathsf{NP}^B \subseteq \mathsf{PSPACE}^B = \mathsf{PSPACE},$$
and so $\mathsf{MCSP}^B \in \mathsf{P}^{\mathsf{B}}/\mathsf{poly}$. By Lemma 22, we conclude that $\mathsf{BPP}^{\mathsf{MCSP}^B} = \mathsf{ZPP}^{\mathsf{MCSP}^B}$.
2. Arguing as above, using Lemma 31 instead of Lemma 30, we obtain $\oplus\mathsf{P} \subseteq \mathsf{BPP}^{\mathsf{MCSP}^{\oplus\mathsf{P}}}$. By Toda's Theorem [43], $\mathsf{NP}^{\oplus\mathsf{P}} \subseteq \mathsf{BPP}^{\oplus\mathsf{P}}$. Therefore, we get
$$\mathsf{MCSP}^{\oplus\mathsf{P}} \in \mathsf{NP}^{\oplus\mathsf{P}} \subseteq \mathsf{BPP}^{\oplus\mathsf{P}} \subseteq \mathsf{P}^{\oplus\mathsf{P}}/\mathsf{poly}.$$
The rest of the argument is the same as above.
3. Similar to the first proof, using Lemma 29 instead of Lemma 30.
4. Since $\mathsf{P}^{\#\mathsf{P}} = \mathsf{P}^{\mathsf{PP}}$ [43], we get that $\mathsf{PP} \subseteq \mathsf{P}^{\#\mathsf{P}} \subseteq \mathsf{BPP}^{\mathsf{MCSP}^{\mathsf{PP}}}$.
5. $\mathsf{PP} \subseteq \mathsf{BPP}^{\mathsf{MCSP}^{\mathsf{PP}}}$. The second part of the claim follows by induction since $C_k\mathsf{P} = \mathsf{PP}^{C_{k-1}\mathsf{P}}$. ◀

▶ **Theorem 47** (Theorem 8 re-stated). $\bigcap_B \mathsf{BPP}^{\mathsf{MCSP}^B}/1 \not\subseteq \mathsf{SIZE}[n^k]$ *and* $\bigcap_B \mathsf{prBPP}^{\mathsf{MCSP}^B} \not\subseteq \mathsf{SIZE}(n^k)$ *for all $k \in \mathbb{N}$, and* $\bigcap_B \mathsf{BPEXP}^{\mathsf{MCSP}^B} \not\subseteq \mathsf{P}/\mathsf{poly}$.

Proof. As in the proof of Theorem 41, we consider two cases:

Case 1: $\mathsf{PSPACE} \subseteq \mathsf{P}/\mathsf{poly}$. Let L_{PSPACE} be the language from Lemma 30. Observe that for every language B, we have $L_{\mathsf{PSPACE}} \in \mathsf{P}^{\mathsf{B}}/\mathsf{poly}$. By Lemma 45 (3), $\mathsf{PSPACE} \subseteq \mathsf{BPP}^{\mathsf{MCSP}^B}$ for all B. By Lemma 19, the required circuit lower bound follows.

Case 2: $\mathsf{PSPACE} \not\subseteq \mathsf{P}/\mathsf{poly}$. For each $k \in \mathbb{N}$ there exists $L'_k \notin \mathsf{SIZE}(n^k)$ such that $L'_k \in \mathsf{BPP}^{\mathsf{MCSP}^B}/1$ for all B. ◀

▶ **Theorem 48** (Theorem 9 re-stated). *For any language B, $n \in \mathbb{N}$ and $\delta > 0$, there exists a MCSP^B-oracle circuit C of size $\mathsf{poly}(n, 1/\delta)$ that is $1-\delta$ close to $B|_n$. If, in addition, B is self-correctable then B has polynomial size MCSP^B-oracle circuits.*

Proof. The proof follows from Lemma 45 for $A = B$, since $B \in \mathsf{P}^{\mathsf{B}}/\mathsf{poly}$. ◀

▶ **Theorem 49** (Theorem 10 re-stated). *Let B be a language such that PSPACE^B has polynomial size B-oracle circuits. Then B has polynomial-size MCSP^B-oracle circuits.*

Proof. Let $A \in \mathsf{PSPACE}^B$ be a language such that B is $(1/\mathsf{poly}(n), A)$-correctable with $r(n) = \mathsf{poly}(n)$, as guaranteed by Theorem 34. By assumption, $A \in \mathsf{PSPACE}^B \subseteq \mathsf{P}^{\mathsf{B}}/\mathsf{poly}$. By Lemma 45 (1), for every $n \in \mathbb{N}$, there exists an MCSP^B-oracle circuit C_n of size $\mathsf{poly}(n)$ such that $\Delta(C_n, A|_{r(n)}) \leq 1/\mathsf{poly}(n)$. Lemma 33 completes the proof. ◀

4 Open questions

The main open question is, of course, to determine the complexity of MCSP. The results in this paper may be interpreted as giving some hope that hardness of MCSP is possible to prove under randomized Turing reductions, as we see a growing list of non-trivial computational tasks that can be solved with the help of the MCSP oracle rather than the SAT oracle. It would be interesting to see more examples of complexity results proved with the SAT oracle that remain true when SAT is replaced with MCSP. For example, is it true that if $\mathsf{SAT} \in \mathsf{P}/\mathsf{poly}$, then SAT circuits can be found by a $\mathsf{ZPP}^{\mathsf{MCSP}}$ algorithm (strengthening the $\mathsf{ZPP}^{\mathsf{NP}}$ result by [13, 34])? Probably a simpler question along these lines is: Does $\mathsf{SAT} \in \mathsf{P}/\mathsf{poly}$ imply that $\mathsf{NP} \subseteq \mathsf{ZPP}^{\mathsf{MCSP}}$?

Some of our hardness results for the relativized MCSP (Theorem 7) are for ZPP reductions, while others for BPP reductions. Is it possible to replace the BPP reductions with ZPP reductions? We have shown it for PSPACE and $\oplus\mathsf{P}$, but not for $\#\mathsf{P}$.

Finally, we proved that, under some assumptions, every language L is computable by a polynomial-size circuit with MCSP^L oracle gates (Theorem 10). Is it true without any assumptions?

References

1 L. M. Adleman. Two theorems on random polynomial time. In *Proceedings of the 19th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 75–83, 1978.

2 Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. Power from random strings. *SIAM J. Comput.*, 35(6):1467–1493, 2006. `doi:10.1137/050628994`.

3 Eric Allender and Bireswar Das. Zero knowledge and circuit minimization. In Erzsébet Csuhaj-Varjú, Martin Dietzfelbinger, and Zoltán Ésik, editors, *Mathematical Foundations of Computer Science 2014 - 39th International Symposium, MFCS 2014, Budapest, Hungary, August 25-29, 2014. Proceedings, Part II*, volume 8635 of *Lecture Notes in Computer Science*, pages 25–32. Springer, 2014. `doi:10.1007/978-3-662-44465-8_3`.

4 Eric Allender, Lisa Hellerstein, Paul McCabe, Toniann Pitassi, and Michael E. Saks. Minimizing disjunctive normal form formulas and ac^0 circuits given a truth table. *SIAM J. Comput.*, 38(1):63–84, 2008. `doi:10.1137/060664537`.

5 Eric Allender, Dhiraj Holden, and Valentine Kabanets. The minimum oracle circuit size problem. In Ernst W. Mayr and Nicolas Ollinger, editors, *32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015, March 4-7, 2015, Garching, Germany*, volume 30 of *LIPIcs*, pages 21–33. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015. `doi:10.4230/LIPIcs.STACS.2015.21`.

6 S. Arora and B. Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

7 Vikraman Arvind and Johannes Köbler. On pseudorandomness and resource-bounded measure. *Theor. Comput. Sci.*, 255(1-2):205–221, 2001. `doi:10.1016/S0304-3975(99)00164-4`.

8 L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991.

9 L. Babai, L. Fortnow, N. Nisan, and A. Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3:307–318, 1993.

10 B. Barak. A probabilistic-time hierarchy theorem for "slightly non-uniform" algorithms. In *RANDOM*, pages 194–208, 2002.

11 B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. In *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference*, pages 1–18, 2001.

12 Donald Beaver and Joan Feigenbaum. Hiding instances in multioracle queries. In Christian Choffrut and Thomas Lengauer, editors, *STACS 90, 7th Annual Symposium on Theoretical Aspects of Computer Science, Rouen, France, February 22-24, 1990, Proceedings*, volume 415 of *Lecture Notes in Computer Science*, pages 37–48. Springer, 1990. `doi:10.1007/3-540-52282-4_30`.

13 Nader H. Bshouty, Richard Cleve, Ricard Gavaldà, Sampath Kannan, and Christino Tamon. Oracles and queries that are sufficient for exact learning. *J. Comput. Syst. Sci.*, 52(3):421–433, 1996. `doi:10.1006/jcss.1996.0032`.

14 H. Buhrman, L. Fortnow, and T. Thierauf. Nonrelativizing separations. In *Proceedings of the 13th Annual IEEE Conference on Computational Complexity (CCC)*, pages 8–12, 1998.

15 Harry Buhrman and Steven Homer. Superpolynomial circuits, almost sparse oracles and the exponential hierarchy. In R. K. Shyamasundar, editor, *Foundations of Software Technology and Theoretical Computer Science, 12th Conference, New Delhi, India, December 18-20, 1992, Proceedings*, volume 652 of *Lecture Notes in Computer Science*, pages 116–127. Springer, 1992. `doi:10.1007/3-540-56287-7_99`.

16 Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning algorithms from natural proofs. In Ran Raz, editor, *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, volume 50 of *LIPIcs*, pages 10:1–10:24. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016. `doi:10.4230/LIPIcs.CCC.2016.10`.

17 J. Feigenbaum and L. Fortnow. Random-self-reducibility of complete sets. *SIAM J. on Computing*, 22(5):994–1005, 1993.

18 L. Fortnow and A. R. Klivans. Efficient learning algorithms yield circuit lower bounds. *J. Comput. Syst. Sci.*, 75(1):27–36, 2009.

19 L. Fortnow and R. Santhanam. Hierarchy theorems for probabilistic polynomial time. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 316–324, 2004.

20 O. Goldreich and D. Zuckerman. Another proof that BPP $\subseteq$ PH (and more). *Studies in Complexity and Cryptography*, pages 40–53, 2011.

21 S. Goldwasser and G. N. Rothblum. On best-possible obfuscation. In *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC*, pages 194–213, 2007.

22 Hans Heller. On relativized exponential and probabilistic complexity classes. *Information and Control*, 71(3):231–243, 1986. `doi:10.1016/S0019-9958(86)80012-2`.

23 Shuichi Hirahara and Osamu Watanabe. Limits of minimum circuit size problem as oracle. In Ran Raz, editor, *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, volume 50 of *LIPIcs*, pages 18:1–18:20. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016. `doi:10.4230/LIPIcs.CCC.2016.18`.

24 John M. Hitchcock and Aduri Pavan. On the np-completeness of the minimum circuit size problem. In Prahladh Harsha and G. Ramalingam, editors, *35th IARCS Annual Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2015, December 16-18, 2015, Bangalore, India*, volume 45 of *LIPIcs*, pages 236–245. Schloss

Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015. doi:10.4230/LIPIcs.FSTTCS.2015.236.

25 R. Impagliazzo, V. Kabanets, and A. Wigderson. In search of an easy witness: Exponential time vs. probabilistic polynomial time. *J. of Computer and System Sciences*, 65(4):672–694, 2002.

26 R. Impagliazzo and A. Wigderson. P=BPP unless E has subexponential circuits: Derandomizing the XOR lemma. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing (STOC)*, pages 220–229, 1997.

27 R. Impagliazzo and A. Wigderson. Randomness vs. time: De-randomization under a uniform assumption. In *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 734–743, 1998.

28 V. Kabanets and J.-Y. Cai. Circuit minimization problem. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 73–79, 2000.

29 Ravi Kannan. Circuit-size lower bounds and non-reducibility to sparse sets. *Information and Control*, 55(1-3):40–56, 1982. doi:10.1016/S0019-9958(82)90382-5.

30 Richard M. Karp. Turing award lecture. In Bill Healy and Judith D. Schlesinger, editors, *Proceedings of the 1985 ACM annual conference on The range of computing: mid-80's perspective: mid-80's perspective, Denver, Colorado, USA, October 14-16, 1985*, page 193. ACM, 1985. doi:10.1145/320435.320497.

31 Richard M. Karp and Richard J. Lipton. Some connections between nonuniform and uniform complexity classes. In Raymond E. Miller, Seymour Ginsburg, Walter A. Burkhard, and Richard J. Lipton, editors, *Proceedings of the 12th Annual ACM Symposium on Theory of Computing, April 28-30, 1980, Los Angeles, California, USA*, pages 302–309. ACM, 1980. doi:10.1145/800141.804678.

32 A. Klivans, P. Kothari, and I. Oliveira. Constructing hard functions from learning algorithms. In *Proceedings of the 28th Annual IEEE Conference on Computational Complexity (CCC)*, pages 86–97, 2013.

33 Adam R. Klivans and Dieter van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM J. Comput.*, 31(5):1501–1526, 2002. doi:10.1137/S0097539700389652.

34 Johannes Köbler and Osamu Watanabe. New collapse consequences of NP having small circuits. *SIAM J. Comput.*, 28(1):311–324, 1998. doi:10.1137/S0097539795296206.

35 Ilan Komargodski, Tal Moran, Moni Naor, Rafael Pass, Alon Rosen, and Eylon Yogev. One-way functions and (im)perfect obfuscation. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 374–383. IEEE Computer Society, 2014. doi:10.1109/FOCS.2014.47.

36 C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. *JACM*, 39(4):859–868, 1992.

37 D. van Melkebeek and K. Pervyshev. A generic time hierarchy with one bit of advice. *Computational Complexity*, 16(2):139–179, 2007.

38 Cody D. Murray and Richard Ryan Williams. On the (non) np-hardness of computing circuit complexity. In David Zuckerman, editor, *30th Conference on Computational Complexity, CCC 2015, June 17-19, 2015, Portland, Oregon, USA*, volume 33 of *LIPIcs*, pages 365–380. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015. doi:10.4230/LIPIcs.CCC.2015.365.

39 N. Nisan and A. Wigderson. Hardness vs. randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994.

40 I. C. Oliveira and R. Santhanam. Conspiracies between learning algorithms, circuit lower bounds and pseudorandomness. *CoRR*, abs/1611.01190, 2016. URL: http://arxiv.org/abs/1611.01190.

41 A. A. Razborov and S. Rudich. Natural proofs. *J. of Computer and System Sciences*, 55(1):24–35, 1997.
42 R. Santhanam. Circuit lower bounds for Merlin–Arthur classes. *SIAM J. Comput.*, 39(3):1038–1061, 2009.
43 S. Toda. PP is as hard as the polynomial time hierarchy. *SIAM J. on Computing*, 20(5):865–877, 1991.
44 Boris A. Trakhtenbrot. A survey of russian approaches to perebor (brute-force searches) algorithms. *IEEE Annals of the History of Computing*, 6(4):384–400, 1984. `doi:10.1109/MAHC.1984.10036`.
45 L. Trevisan and S. P. Vadhan. Pseudorandomness and average-case complexity via uniform reductions. *Computational Complexity*, 16(4):331–364, 2007.
46 C. Umans. Pseudo-random generators for all hardnesses. *J. of Computer and System Sciences*, 67(2):419–440, 2003.
47 L. G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201, 1979.
48 L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.
49 I. Volkovich. On learning, lower bounds and (un)keeping promises. In *Proceedings of the 41st ICALP*, pages 1027–1038, 2014.

A Self-correctable and downward-reducible $\oplus$P-complete problem

In this section we prove Lemma 31

▶ **Lemma 50.** *There is a downward self-reducible and self-correctable $\oplus$P-complete language $L_{\oplus \mathsf{P}}$.*

Proof sketch. The proof is very similar to the one in [45] for the case of PSPACE. We define a formula $\Phi_n(\bar{x}, \bar{y})$ that is universal for n-variate 3-cnf formulas on the variables $\bar{x} = (x_1, \ldots, x_n)$, where $\bar{y} = (y_1, \ldots, y_{8n^3})$ describes a particular 3-cnf formula ϕ by specifying, for each possible clause on 3 variables, whether this clause is present in ϕ. For example, if $c_1, \ldots, c_m$, for $m = 8n^3$, is a sequence of all possible 3-clauses on n variables $x_1, \ldots, x_n$, we can define Φ as follows:

$$\Phi(\bar{x}, \bar{y}) = \wedge_{i=1}^{m}((y_i \wedge c_i) \vee \neg y_i).$$

We now "arithmetize" the formula Φ, getting a polynomial that agrees with Φ over all Boolean inputs. We will work over the finite field $\mathbb{F}_{2^k}$ of characteristic 2, for $k = 5 \log n$. Arithmetizing all clauses c_i's (by replacing each c_i with a degree 3 multilinear polynomial c_i', in the same 3 variables, that agrees with c_i over Boolean all assignments), we get the following arithmetization Φ' of Φ:

$$\Phi'(\bar{x}, \bar{y}) = \prod_{i=1}^{m}(y_i \cdot c_i' + 1 + y_i).$$

For each $0 \le i \le n$, define a polynomial

$$f_{n,i}(x_1, \ldots, x_i, \bar{y}) = \sum_{x_{i+1} \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} \Phi'(\bar{x}, \bar{y}),$$

where the summation is over our field $\mathbb{F}_{2^k}$ of characteristic 2. Note that $f_{n,0}(\bar{y})$, for a Boolean $\bar{y}$, is exactly $\oplus$SAT on the 3-cnf instance described by $\bar{y}$. So $f_{n,0}$ is $\oplus$P-hard to compute.

We have that $f_{n,i}$ can be expressed in terms of $f_{n,i+1}$, for $i < n$, by the formula:

$$f_{n,i}(x_1, \ldots, x_i, \bar{y}) = f_{n,i+1}(x_1, \ldots, x_i, 0) + f_{n,i+1}(x_1, \ldots, x_i, 1).$$

So $f_{n,i}$ can be computed in polynomial time with oracle access to $f_{n,i+1}$. It is also clear that $f_{n,n}$ can be evaluated in polynomial time (directly).

Next, in the same way as in [45], we define a Boolean function family $F = \{F_t\}_{t\geq 1}$ so that each $f_{n,i}$ is "embedded" into some $F_{h(n,i)}$, for some function $h\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$. Namely, h can be chosen so that

- $h(n, i) > h(n, i + 1)$ (and so we have downward-reducibility for $f_{n,i}$'s), and
- the length $h(n, i)$ is large enough to accommodate both an input to $f_{n,i}$ and an index $j \in [k]$.

We then define

$$F_{h(n,i)}(x_1, \ldots, x_i, \bar{y}, j) = f_{n,i}(x_1, \ldots, x_i, \bar{y})_j,$$

i.e., the jth bit of the value of $f_{n,i}$ in the field $\mathbb{F}_{2^k}$, whose elements are viewed as k-bit vectors.

The downward-reducibility of F follows from the properties of $f_{n,i}$ (and the way we arranged the lengths $h(n, i)$). The self-correctability of F follows from the fact each $f_{n,i}$ is a $O(n^3)$-degree polynomial over the field of size $2^k \geq n^5$ (see [45] for more details). The $\oplus\mathsf{P}$-hardness of F follows from $\oplus\mathsf{P}$-hardness of $f_{n,0}$'s.

It remains to show that $F \in \oplus\mathsf{P}$. Note that every bit j of the value of $f_{n,n}(\bar{y})$, for every input $\bar{y}$, is computable in P, and hence also in $\oplus\mathsf{P}$. For any $0 \leq i < n$, the jth bit of $f_{n,i}(x_1, \ldots, x_i, \bar{y})$ can be computed in $\oplus\mathsf{P}$ using the following nondeterministic algorithm:

"Nondeterministically guess Boolean values $b_{i+1}, \ldots, b_n$. Compute the value

$$v = \Phi'(x_1, \ldots, x_i, b_{i+1}, \ldots, b_n, \bar{y}).$$

Accept if the jth bit of the computed field element v is 1, and reject otherwise."

The parity of the number of accepting paths of the algorithm above is exactly the sum modulo 2 of the bits v_j, over all Boolean assignments to $x_{i+1}, \ldots, x_n$. The latter is exactly the jth bit of $f_{n,i}$ because addition in the field $\mathbb{F}_{2^k}$ is the bit-wise XOR of the corresponding k-bit vectors. ◀

B Proof of Lemma 35

Let $\mathcal{A}$ be a PAC learner for $\mathcal{C}$ and let $n = |x|$. First, we describe an algorithm that produces B-oracle circuits for $L|_1$, $L|_2$, ..., $L|_n$ w.h.p. We then use the circuit for $L|_n$ to decide x.

- Begin with a look-up table $\tilde{C}_1 = C_1$ for $L|_1$.
- For $i \geq 2$, invoke $\mathcal{A}$ with $\varepsilon = 1/i^3$ and $\delta = 1/i$ to learn a circuit $\tilde{C}_i$ of size t for $L|_i$.
- Answer the queries to B using the provided oracle.
- Given a query to $L|_i$, invoke $\mathsf{COMPUTE}$ with $C_{i-1}(x)$ as an oracle.
- Set $C_i \triangleq \mathsf{CORRECT}^{\tilde{C}_i}$ (convert the algorithm into a circuit using Lemma 33).

We claim that w.h.p it holds for all $1 \leq i \leq n$ that C_i is a B-oracle circuit of size $\mathsf{poly}(i, t)$ computing $L|_i$. The proof is by induction on i. Basis $i = 1$ is clear. Now assume that hypothesis holds for $i - 1$. Observe that since $C_{i-1}(x)$ is B-oracle circuit, it can be evaluated in polynomial time given and an oracle to B. Hence, by downward self-reducibility of L

invoking COMPUTE with $C_{i-1}(x)$ can be used to obtain oracle access to $L|_i$. As $t \geq \mathsf{s}_L(i)$, $\mathcal{A}$ will output a circuit $\tilde{C}_i$ of size $\mathsf{poly}(i,t)$. which is $1/i$ close to $L|_i$. Finally, using Lemma 33 the algorithm will produce a circuit C_i of size $\mathsf{poly}(i,t)$ that computes $L|_i$.

The above analysis is correct assuming that no errors have occurred. Note that the total number of steps is $\mathsf{poly}(i)$ while each steps has at most $1/\mathsf{poly}(i)$ probability error. As the latter polynomial can be made arbitrary small, we obtain that w.h.p. for all i, $C_i \equiv L|_i$.

Finally, all the listed procedures are in time $\mathsf{poly}(n,t)$, given oracle access to B.

C Oracles B where $\mathsf{PSPACE}^B \subseteq \mathsf{P}^\mathsf{B}/\mathsf{poly}$ but $\mathsf{PSPACE}^B \neq \mathsf{P}^B$

▶ **Lemma 51.** *Let B be a language such that* $\mathsf{EXP}^B \subseteq \mathsf{P}^\mathsf{B}/\mathsf{poly}$. *Then* $\mathsf{PSPACE}^B \neq \mathsf{P}^B$.

Proof. Assume the contrary. By Meyer's Theorem [31]: $\mathsf{EXP}^B \subseteq \Sigma_2^B \subseteq \mathsf{PSPACE}^B$. By the assumption, $\mathsf{EXP}^B \subseteq \mathsf{PSPACE}^B \subseteq \mathsf{P}^B$ which contradicts Time Hierarchy Theorem. ◀

There are numerous examples of languages satisfying the preconditions of the Lemma; see, e.g., [22, 14].