Algebrization: A New Barrier in Complexity Theory

Scott Aaronson[*] MIT

Avi Wigderson[†] Institute for Advanced Study

Abstract

Any proof of $\mathsf{P} \neq \mathsf{NP}$ will have to overcome two barriers: *relativization* and *natural proofs*. Yet over the last decade, we have seen circuit lower bounds (for example, that PP does not have linear-size circuits) that overcome both barriers simultaneously. So the question arises of whether there is a third barrier to progress on the central questions in complexity theory.

In this paper we present such a barrier, which we call *algebraic relativization* or *algebrization*. The idea is that, when we relativize some complexity class inclusion, we should give the simulating machine access not only to an oracle A, but also to a low-degree extension of A over a finite field or ring.

We systematically go through basic results and open problems in complexity theory to delineate the power of the new algebrization barrier. First, we show that all known non-relativizing results based on arithmetization—both inclusions such as $\mathsf{IP} = \mathsf{PSPACE}$ and $\mathsf{MIP} = \mathsf{NEXP}$, and separations such as $\mathsf{MA}_{\mathsf{EXP}} \not\subset \mathsf{P/poly}$ —do indeed algebrize. Second, we show that almost all of the major open problems—including P versus NP, P versus RP, and NEXP versus $\mathsf{P/poly}$—will require *non-algebrizing techniques*. In some cases algebrization seems to explain exactly why progress stopped where it did: for example, why we have superlinear circuit lower bounds for $\mathsf{PromiseMA}$ but not for NP.

Our second set of results follows from lower bounds in a new model of *algebraic query complexity*, which we introduce in this paper and which is interesting in its own right. Some of our lower bounds use direct combinatorial and algebraic arguments, while others stem from a surprising connection between our model and communication complexity. Using this connection, we are also able to give an MA-protocol for the Inner Product function with $O(\sqrt{n}\log n)$ communication (essentially matching a lower bound of Klauck), as well as a communication complexity conjecture whose truth would imply $\mathsf{NL} \neq \mathsf{NP}$.

1 Introduction

In the history of the P versus NP problem, there were two occasions when researchers stepped back, identified some property of almost all the techniques that had been tried up to that point, and then proved that *no technique with that property could possibly work*. These "meta-discoveries" constitute an important part of what we understand about the P versus NP problem beyond what was understood in 1971.

The first meta-discovery was *relativization*. In 1975, Baker, Gill, and Solovay [4] showed that techniques borrowed from logic and computability theory, such as diagonalization, cannot be powerful enough to resolve P versus NP. For these techniques would work equally well in a "relativized world," where both P and NP machines could compute some function f in a single time step. However, there are some relativized worlds where $\mathsf{P} = \mathsf{NP}$, and other relativized worlds

[*]Email: aaronson@csail.mit.edu.

[†]Email: avi@ias.edu.

where $\mathsf{P} \neq \mathsf{NP}$. Therefore any solution to the P versus NP problem will require *non-relativizing techniques*: techniques that exploit properties of computation that are specific to the real world.

The second meta-discovery was *natural proofs*. In 1993, Razborov and Rudich [31] analyzed the circuit lower bound techniques that had led to some striking successes in the 1980's, and showed that, if these techniques worked to prove separations like $\mathsf{P} \neq \mathsf{NP}$, then we could turn them around to obtain faster ways to distinguish random functions from pseudorandom functions. But in that case, we would be finding fast algorithms for some of the very same problems (like inverting one-way functions) that we wanted to prove were hard.

1.1 The Need for a New Barrier

Yet for both of these barriers—relativization and natural proofs—we do know ways to circumvent them.

In the early 1990's, researchers managed to prove $\mathsf{IP} = \mathsf{PSPACE}$ [23, 33] and other celebrated theorems about interactive protocols, even in the teeth of relativized worlds where these theorems were false. To do so, they created a new technique called *arithmetization*. The idea was that, instead of treating a Boolean formula φ as just a black box mapping inputs to outputs, one can take advantage of the structure of φ, by "promoting" its AND, OR, or NOT gates to arithmetic operations over some larger field $\mathbb{F}$. One can thereby extend φ to a low-degree polynomial $\widetilde{\varphi} : \mathbb{F}^n \to \mathbb{F}$, which has useful error-correcting properties that were unavailable in the Boolean case.

In the case of the natural proofs barrier, a way to circumvent it was actually known since the work of Hartmanis and Stearns [14] in the 1960's. Any complexity class separation proved via diagonalization—such as $\mathsf{P} \neq \mathsf{EXP}$ or $\Sigma_2^{\mathsf{EXP}} \not\subset \mathsf{P/poly}$ [19]—is inherently non-naturalizing. For diagonalization zeroes in on a specific property of the function f being lower-bounded—namely, the ability of f to simulate a whole class of machines—and thereby avoids the trap of arguing that "f is hard because it looks like a random function."

Until a decade ago, one could at least say that all known circuit lower bounds were subject either to the relativization barrier, *or* to the natural proofs barrier. But not even that is true any more. We now have circuit lower bounds that evade *both* barriers, by cleverly combining arithmetization (which is non-relativizing) with diagonalization (which is non-naturalizing).

The first such lower bound was due to Buhrman, Fortnow, and Thierauf [6], who showed that $\mathsf{MA}_{\mathsf{EXP}}$, the exponential-time analogue of MA, is not in $\mathsf{P/poly}$. To prove that their result was non-relativizing, Buhrman et al. also gave an oracle A such that $\mathsf{MA}_{\mathsf{EXP}}^{A} \subset \mathsf{P}^{A}/\mathsf{poly}$. Using similar ideas, Vinodchandran [37] showed that for every fixed k, the class PP does not have circuits of size n^k; and Aaronson [1] showed that Vinodchandran's result was non-relativizing, by giving an oracle A such that $\mathsf{PP}^{A} \subset \mathsf{SIZE}^{A}(n)$. Recently, Santhanam [32] gave a striking improvement of Vinodchandran's result, by showing that for every fixed k, the class $\mathsf{PromiseMA}$ does not have circuits of size n^k.

As Santhanam [32] stressed, these results raise an important question: given that current techniques can already overcome the two central barriers of complexity theory, *how much further can one push those techniques?* Could arithmetization and diagonalization already suffice to prove circuit lower bounds for NEXP, or even $\mathsf{P} \neq \mathsf{NP}$? Or is there a third barrier, beyond relativization and natural proofs, to which even the most recent results are subject?

1.2 Our Contribution

In this paper we show that there is, alas, a third barrier to solving P versus NP and the other central problems of complexity theory.

Recall that a key insight behind the non-relativizing interactive proof results was that, given a Boolean formula φ, one need not treat φ as merely a black box, but can instead reinterpret it as a low-degree polynomial $\widetilde{\varphi}$ over a larger field or ring. To model that insight, in this paper we consider *algebraic oracles*: oracles that can evaluate not only a Boolean function f, but also a low-degree extension $\widetilde{f}$ of f over a finite field or the integers. We then define *algebrization* (short for "algebraic relativization"), the main notion of this paper.

Roughly speaking, we say that a complexity class inclusion $\mathcal{C} \subseteq \mathcal{D}$ *algebrizes* if $\mathcal{C}^A \subseteq \mathcal{D}^{\widetilde{A}}$ for all oracles A and all low-degree extensions $\widetilde{A}$ of A. Likewise, a separation $\mathcal{C} \not\subset \mathcal{D}$ algebrizes if $\mathcal{C}^{\widetilde{A}} \not\subset \mathcal{D}^A$ for all $A, \widetilde{A}$. Notice that algebrization is defined differently for inclusions and separations; and that in both cases, only one complexity class gets the algebraic oracle $\widetilde{A}$, while the other gets the Boolean version A. These subtle asymmetries are essential for this new notion to capture what we want, and will be explained in Section 2.

We will demonstrate how algebrization captures a new barrier by proving two sets of results. The first set shows that, of the known results based on arithmetization that fail to relativize, *all* of them algebrize. This includes the interactive proof results, as well as their consequences for circuit lower bounds. More concretely, in Section 3 we show (among other things) that, for all oracles A and low-degree extensions $\widetilde{A}$ of A:

- $\mathsf{PSPACE}^A \subseteq \mathsf{IP}^{\widetilde{A}}$
- $\mathsf{NEXP}^A \subseteq \mathsf{MIP}^{\widetilde{A}}$
- $\mathsf{MA}_{\mathsf{EXP}}^{\widetilde{A}} \not\subset \mathsf{P}^A/\mathsf{poly}$
- $\mathsf{PromiseMA}^{\widetilde{A}} \not\subset \mathsf{SIZE}^A\left(n^k\right)$

The second set of results shows that, for many basic complexity questions, any solution will require non-algebrizing techniques. In Section 5 we show (among other things) that there exist oracles $A, \widetilde{A}$ relative to which:

- $\mathsf{NP}^{\widetilde{A}} \subseteq \mathsf{P}^A$, and indeed $\mathsf{PSPACE}^{\widetilde{A}} \subseteq \mathsf{P}^A$
- $\mathsf{NP}^A \not\subset \mathsf{P}^{\widetilde{A}}$, and indeed $\mathsf{RP}^A \not\subset \mathsf{P}^{\widetilde{A}}$
- $\mathsf{NP}^A \not\subset \mathsf{BPP}^{\widetilde{A}}$, and indeed $\mathsf{NP}^A \not\subset \mathsf{BQP}^{\widetilde{A}}$ and $\mathsf{NP}^A \not\subset \mathsf{coMA}^{\widetilde{A}}$
- $\mathsf{P}^{\mathsf{NP}^A} \not\subset \mathsf{PP}^{\widetilde{A}}$
- $\mathsf{NEXP}^{\widetilde{A}} \subset \mathsf{P}^A/\mathsf{poly}$
- $\mathsf{NP}^{\widetilde{A}} \subset \mathsf{SIZE}^A(n)$

These results imply, in particular, that any resolution of the P versus NP problem will need to use non-algebrizing techniques. But the take-home message for complexity theorists is stronger: non-algebrizing techniques will be needed even to derandomize RP, to separate NEXP from P/poly, or to prove superlinear circuit lower bounds for NP.[1]

By contrast, recall that the separations $\mathsf{MA}_{\mathsf{EXP}} \not\subset \mathsf{P}/\mathsf{poly}$ and $\mathsf{PromiseMA} \not\subset \mathsf{SIZE}\left(n^k\right)$ have already been proved with algebrizing techniques. Thus, we see that known techniques can prove

[1]In other words, find a language $L \in \mathsf{NP}$ that does not have circuits of size cn for any constant c.

superlinear circuit lower bounds for $\mathsf{PromiseMA}$, but *cannot* do the same for NP—even though $\mathsf{MA} = \mathsf{NP}$ under standard hardness assumptions [22]. Similarly, known techniques can prove superpolynomial circuit lower bounds for $\mathsf{MA}_{\mathsf{EXP}}$ but not for NEXP. To summarize:

> *Algebrization provides nearly the precise limit on the non-relativizing techniques of the last two decades.*

We speculate that going beyond this limit will require fundamentally new methods.[2]

1.3 Techniques

This section naturally divides into two, one for each of our main sets of results.

1.3.1 Proving That Existing Results Algebrize

Showing that the interactive proof results algebrize is conceptually simple (though a bit tedious in some cases), once one understands the specific way these results use arithmetization. In our view, it is the very naturalness of the algebrization concept that makes the proofs so simple.

To illustrate, consider the result of Lund, Fortnow, Karloff, and Nisan [23] that $\mathsf{coNP} \subseteq \mathsf{IP}$. In the LFKN protocol, the verifier (Arthur) starts with a Boolean formula φ, which he arithmetizes to produce a low-degree polynomial $\widetilde{\varphi} : \mathbb{F}^n \rightarrow \mathbb{F}$. The prover (Merlin) then wants to convince Arthur that

$$\sum_{x \in \{0,1\}^n} \widetilde{\varphi}(x) = 0.$$

To do so, Merlin engages Arthur in a conversation about the sums of $\widetilde{\varphi}$ over various subsets of points in $\mathbb{F}^n$. For almost all of this conversation, Merlin is "doing the real work." Indeed, the only time Arthur ever uses his description of $\widetilde{\varphi}$ is in the very last step, when he checks that $\widetilde{\varphi}(r_1, \ldots, r_n)$ is equal to the value claimed by Merlin, for some random field elements $r_1, \ldots, r_n$ chosen earlier in the protocol.

Now suppose we want to prove $\mathsf{coNP}^A \subseteq \mathsf{IP}^{\widetilde{A}}$. The only change is that now Arthur's formula φ will in general contain A gates, in addition to the usual AND, OR, and NOT gates. And therefore, when Arthur arithmetizes φ to produce a low-degree polynomial $\widetilde{\varphi}$, his description of $\widetilde{\varphi}$ will contain terms of the form $A(z_1, \ldots, z_k)$. Arthur then faces the problem of how to evaluate these terms when the inputs $z_1, \ldots, z_k$ are non-Boolean. At this point, though, the solution is clear: Arthur simply calls the oracle $\widetilde{A}$ to get $\widetilde{A}(z_1, \ldots, z_k)$!

While the details are slightly more complicated, the same idea can be used to show $\mathsf{PSPACE}^A \subseteq \mathsf{IP}^{\widetilde{A}}$ and $\mathsf{NEXP}^A \subseteq \mathsf{MIP}^{\widetilde{A}}$.

But what about the non-relativizing separation results, like $\mathsf{MA}_{\mathsf{EXP}} \not\subset \mathsf{P/poly}$? When we examine the proofs of these results, we find that each of them combines a single non-relativizing ingredient—namely, an interactive proof result—with a sequence of relativizing results. Therefore, having shown that the interactive proof results algebrize, we have already done most of the work of showing the separations algebrize as well.

[2]While we have shown that most non-relativizing results algebrize, we note that we have skipped some famous examples—involving small-depth circuits, time-space tradeoffs for *SAT*, and the like. We discuss some of these examples in Section 9.

1.3.2 Proving The Necessity of Non-Algebrizing Techniques

It is actually easy to show that any proof of $\mathsf{NP} \not\subset \mathsf{P}$ will need non-algebrizing techniques. One simply lets A be a PSPACE-complete language and $\widetilde{A}$ be a PSPACE-complete extension of A; then $\mathsf{NP}^{\widetilde{A}} = \mathsf{P}^{A} = \mathsf{PSPACE}$. What is harder is to show that any proof of $\mathsf{RP} \subseteq \mathsf{P}$, $\mathsf{NP} \subseteq \mathsf{BPP}$, and so on will need non-algebrizing techniques. For the latter problems, we are faced with the task of proving *algebraic oracle separations*. In other words, we need to show (for example) that there exist oracles $A, \widetilde{A}$ such that $\mathsf{RP}^{A} \not\subset \mathsf{P}^{\widetilde{A}}$ and $\mathsf{NP}^{A} \not\subset \mathsf{BPP}^{\widetilde{A}}$.

Just like with standard oracle separations, to prove an algebraic oracle separation one has to do two things:

(1) Prove a concrete lower bound on the query complexity of some function.

(2) Use the query complexity lower bound to diagonalize against a class of Turing machines.

Step (2) is almost the same for algebraic and standard oracle separations; it uses the bounds from (1) in a diagonalization argument. Step (1), on the other hand, is extremely interesting; it requires us to prove lower bounds in a new model of *algebraic query complexity*.

In this model, an algorithm is given oracle access to a Boolean function $A : \{0,1\}^n \rightarrow \{0,1\}$. It is trying to answer some question about A—for example, *"is there an $x \in \{0,1\}^n$ such that $A(x) = 1$?"*—by querying A on various points. The catch is that the algorithm can query not just A itself, but also an adversarially-chosen low-degree extension $\widetilde{A} : \mathbb{F}^n \rightarrow \mathbb{F}$ of A over some finite field $\mathbb{F}$.[3] In other words, the algorithm is no longer merely searching for a needle in a haystack: it can also search a low-degree extension of the haystack for "nonlocal clues" of the needle's presence!

This model is clearly at least as strong as the standard one, since an algorithm can always restrict itself to Boolean queries only (which are answered identically by A and $\widetilde{A}$). Furthermore, we know from interactive proof results that the new model is sometimes much stronger: sampling points outside the Boolean cube does, indeed, sometimes help a great deal in determining properties of A. This suggests that, to prove lower bounds in this model, we are going to need new techniques.

In this paper we develop *two* techniques for lower-bounding algebraic query complexity, which have complementary strengths and weaknesses.

The first technique is based on direct construction of adversarial polynomials. Suppose an algorithm has queried the points $y_1, \ldots, y_t \in \mathbb{F}^n$. Then by a simple linear algebra argument, it is possible to create a multilinear polynomial p that evaluates to 0 on all the y_i's, and that simultaneously has any values we specify on $2^n - t$ points of the Boolean cube. The trouble is that, on the remaining t Boolean points, p will not necessarily be Boolean: that is, it will not necessarily be an extension of a Boolean function. We solve this problem by multiplying p with a *second* multilinear polynomial, to produce a "multiquadratic" polynomial (a polynomial of degree at most 2 in each variable) that is Boolean on the Boolean cube and that also has the desired adversarial behavior.

The idea above becomes more complicated for randomized lower bounds, where we need to argue about the indistinguishability of distributions over low-degree polynomials conditioned on a small number of queries. And it becomes more complicated still when we switch from finite field extensions to extensions $\widehat{A} : \mathbb{Z}^n \rightarrow \mathbb{Z}$ over the integers. In the latter case, we can no longer use linear algebra to construct the multilinear polynomial p, and we need to compensate by bringing in some tools from elementary number theory, namely Chinese remaindering and Hensel lifting. Even then, a technical problem (that the number of bits needed to express $\widetilde{A}(x)$ grows with the running times

[3] Later, we will also consider extensions over the integers.

of the machines being diagonalized against) currently prevents us from turning query complexity lower bounds obtained by this technique into algebraic oracle separations over the integers.

Our second lower-bound technique comes as an "unexpected present" from communication complexity. Given a Boolean function $A : \{0,1\}^n \rightarrow \{0,1\}$, let A_0 and A_1 be the subfunctions obtained by fixing the first input bit to 0 or 1 respectively. Also, suppose Alice is given the truth table of A_0, while Bob is given the truth table of A_1. Then we observe the following connection between algebraic query complexity and communication complexity:

> If some property of A can be determined using T queries to a multilinear extension $\widetilde{A}$ of A over the finite field $\mathbb{F}$, then it can also be determined by Alice and Bob using $O(Tn \log |\mathbb{F}|)$ bits of communication.

This connection is extremely generic: it lets us convert randomized algorithms querying $\widetilde{A}$ into randomized communication protocols, quantum algorithms into quantum protocols, MA-algorithms into MA-protocols, and so on. Turning the connection around, we find that *any communication complexity lower bound automatically leads to an algebraic query complexity lower bound.* This means, for example, that we can use celebrated lower bounds for the Disjointness problem [29, 18, 21, 30] to show that there exist oracles $A, \widetilde{A}$ relative to which $\mathsf{NP}^A \not\subset \mathsf{BPP}^{\widetilde{A}}$, and even $\mathsf{NP}^A \not\subset \mathsf{BQP}^{\widetilde{A}}$ and $\mathsf{NP}^A \not\subset \mathsf{coMA}^{\widetilde{A}}$. For the latter two results, we do not know of any proof by direct construction of polynomials.

The communication complexity technique has two further advantages: it yields *multilinear* extensions instead of multiquadratic ones, and it works just as easily over the integers as over finite fields. On the other hand, the lower bounds one gets from communication complexity are more contrived. For example, one can show that solving the Disjointness problem requires exponentially many queries to $\widetilde{A}$, but not that finding a Boolean x such that $A(x) = 1$ does. Also, we do not know how to use communication complexity to construct $A, \widetilde{A}$ such that $\mathsf{NEXP}^{\widetilde{A}} \subset \mathsf{P}^A/\mathsf{poly}$ and $\mathsf{NP}^{\widetilde{A}} \subset \mathsf{SIZE}^A(n)$.

1.4 Related Work

In a survey article on "The Role of Relativization in Complexity Theory," Fortnow [10] defined a class of oracles $\mathcal{O}$ relative to which $\mathsf{IP} = \mathsf{PSPACE}$. His proof that $\mathsf{IP}^A = \mathsf{PSPACE}^A$ for all $A \in \mathcal{O}$ was similar to our proof, in Section 3.2, that $\mathsf{IP} = \mathsf{PSPACE}$ algebrizes. However, because he wanted both complexity classes to have access to the same oracle A, Fortnow had to define his oracles in a subtle recursive way, as follows: start with an arbitrary Boolean oracle B, then let $\widetilde{B}$ be the multilinear extension of B, then let f be the "Booleanization" of $\widetilde{B}$ (i.e., $f(x, i)$ is the i^{th} bit in the binary representation of $\widetilde{B}(x)$), then let $\widetilde{\widetilde{B}}$ be the multilinear extension of f, and so on ad infinitum. Finally let A be the concatenation of all these oracles.

As we discuss in Section 10.1, it seems extremely difficult to prove *separations* relative to these recursively-defined oracles. So if the goal is to show the limitations of current proof techniques for solving open problems in complexity theory, then a non-recursive definition like ours seems essential.

Recently (and independently of us), Juma, Kabanets, Rackoff and Shpilka [17] studied an algebraic query complexity model closely related to ours, and proved lower bounds in this model. In our terminology, they "almost" constructed an oracle A, and a multiquadratic extension $\widetilde{A}$ of

A, such that $\#\mathsf{P}^A \not\subset \mathsf{FP}^{\widetilde{A}}/\mathsf{poly}$.[4] Our results in Section 4 extend those of Juma et al. and solve some of their open problems.

Juma et al. also made the interesting observation that, if the extension $\widetilde{A}$ is multilinear rather than multiquadratic, then oracle access to $\widetilde{A}$ sometimes switches from being useless to being extraordinarily powerful. For example, let $A : \{0,1\}^n \to \{0,1\}$ be a Boolean function, and let $\widetilde{A} : \mathbb{F}^n \to \mathbb{F}$ be the multilinear extension of A, over any field $\mathbb{F}$ of characteristic other than 2. Then we can evaluate the sum $\sum_{x\in\{0,1\}^n} A(x)$ with just a *single* query to $\widetilde{A}$, by using the fact that

$$\sum_{x\in\{0,1\}^n} A(x) = 2^n \widetilde{A}\left(\frac{1}{2}, \ldots, \frac{1}{2}\right).$$

This observation helps to explain why, in Section 4, we will often need to resort to multiquadratic extensions instead of multilinear ones.

1.5 Table of Contents

The rest of the paper is organized as follows.

Section 2	Formal definition of algebraic oracles, and various subtleties of the model
Section 3	Why known results such as $\mathsf{IP} = \mathsf{PSPACE}$ and $\mathsf{MIP} = \mathsf{NEXP}$ algebrize
Section 4	Lower bounds on algebraic query complexity
Section 5	Why open problems will require non-algebrizing techniques to be solved
Section 6	Generalizing to low-degree extensions over the integers
Section 7	Two applications of algebrization to communication complexity
Section 8	The GMW zero-knowledge protocol for NP, and why it algebrizes
Section 9	Whether we have non-relativizing techniques besides arithmetization
Section 10	Two ideas for going beyond the algebrization barrier and their limitations
Section 11	Conclusions and open problems

Also, the following table lists our most important results and where to find them.

Result	**Theorem(s)**
$\mathsf{IP} = \mathsf{PSPACE}$ algebrizes	3.7
$\mathsf{MIP} = \mathsf{NEXP}$ algebrizes	3.8
Recent circuit lower bounds like $\mathsf{MA}_{\mathsf{EXP}} \not\subset \mathsf{P}/\mathsf{poly}$ algebrize	3.16-3.18
Lower bound on algebraic query complexity (deterministic, over fields)	4.4
Lower bound on algebraic query complexity (probabilistic, over fields)	4.9
Communication lower bounds imply algebraic query lower bounds	4.11
Proving $\mathsf{P} \neq \mathsf{NP}$ will require non-algebrizing techniques	5.1
Proving $\mathsf{P} = \mathsf{NP}$ (or $\mathsf{P} = \mathsf{RP}$) will require non-algebrizing techniques	5.3
Proving $\mathsf{NP} \subseteq \mathsf{BPP}$ (or $\mathsf{NP} \subset \mathsf{P}/\mathsf{poly}$) will require non-algebrizing techniques	5.4
Proving $\mathsf{NEXP} \not\subset \mathsf{P}/\mathsf{poly}$ will require non-algebrizing techniques	5.6
Proving $\mathsf{NP} \subseteq \mathsf{BQP}$, $\mathsf{BPP} = \mathsf{BQP}$, etc. will require non-algebrizing techniques	5.7
Lower bound on algebraic query complexity (deterministic, over integers)	6.10
Plausible communication complexity conjecture implying $\mathsf{NL} \neq \mathsf{NP}$	7.2
Inner Product admits an MA-protocol with $O(\sqrt{n}\log n)$ communication	7.4
The GMW Theorem algebrizes	8.4

[4] We say "almost" because they did not ensure $\widetilde{A}(x)$ was Boolean for all Boolean x; this is an open problem of theirs that we solve in Section 4.2.1. Also, their result is only for field extensions and not integer extensions.

2 Oracles and Algebrization

In this section we discuss some preliminaries, and then formally define the main notions of the paper: extension oracles and algebrization.

We use $[t]$ to denote the set $\{1,\ldots,t\}$. See the Complexity Zoo[5] for definitions of the complexity classes we use.

Given a multivariate polynomial $p(x_1,\ldots,x_n)$, we define the *multidegree* of p, or $\operatorname{mdeg}(p)$, to be the maximum degree of any x_i. We say p is *multilinear* if $\operatorname{mdeg}(p) \leq 1$, and *multiquadratic* if $\operatorname{mdeg}(p) \leq 2$. Also, we call p an *extension polynomial* if $p(x) \in \{0,1\}$ whenever $x \in \{0,1\}^n$. Intuitively, this means that p is the polynomial extension of some Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$.

The right way to relativize complexity classes such as PSPACE and EXP has long been a subject of dispute: should we allow exponentially-long queries to the oracle, or only polynomially-long queries? On the one hand, if we allow exponentially-long queries, then statements like "$\mathsf{IP} = \mathsf{PSPACE}$ is non-relativizing" are reduced to trivialities, since the PSPACE machine can simply query oracle bits that the IP machine cannot reach. On the other hand, if we allow only polynomially-long queries, then results based on padding—for example, $\mathsf{P} = \mathsf{NP} \Longrightarrow \mathsf{EXP} = \mathsf{NEXP}$—will generally fail to relativize.[6]

In this paper we adopt a pragmatic approach, writing $\mathcal{C}^A$ or $\mathcal{C}^{A[\mathrm{poly}]}$ to identify which convention we have in mind. More formally:

Definition 2.1 (Oracle) *An oracle A is a collection of Boolean functions $A_m : \{0,1\}^m \rightarrow \{0,1\}$, one for each $m \in \mathbb{N}$. Then given a complexity class $\mathcal{C}$, by $\mathcal{C}^A$ we mean the class of languages decidable by a $\mathcal{C}$ machine that can query A_m for any m of its choice. By $\mathcal{C}^{A[\mathrm{poly}]}$ we mean the class of languages decidable by a $\mathcal{C}$ machine that, on inputs of length n, can query A_m for any $m = O(\mathrm{poly}(n))$. For classes $\mathcal{C}$ such that all computation paths are polynomially bounded (for example, P, NP, BPP, $\#\mathsf{P}$...), it is obvious that $\mathcal{C}^{A[\mathrm{poly}]} = \mathcal{C}^A$.*

We now define the key notion of an *extension oracle*.

Definition 2.2 (Extension Oracle Over A Finite Field) *Let $A_m : \{0,1\}^m \rightarrow \{0,1\}$ be a Boolean function, and let $\mathbb{F}$ be a finite field. Then an extension of A_m over $\mathbb{F}$ is a polynomial $\widetilde{A}_{m,\mathbb{F}} : \mathbb{F}^m \rightarrow \mathbb{F}$ such that $\widetilde{A}_{m,\mathbb{F}}(x) = A_m(x)$ whenever $x \in \{0,1\}^m$. Also, given an oracle $A = (A_m)$, an extension $\widetilde{A}$ of A is a collection of polynomials $\widetilde{A}_{m,\mathbb{F}} : \mathbb{F}^m \rightarrow \mathbb{F}$, one for each positive integer m and finite field $\mathbb{F}$, such that*

(i) $\widetilde{A}_{m,\mathbb{F}}$ is an extension of A_m for all $m, \mathbb{F}$, and

(ii) there exists a constant c such that $\operatorname{mdeg}(\widetilde{A}_{m,\mathbb{F}}) \leq c$ for all $m, \mathbb{F}$.[7]

Then given a complexity class $\mathcal{C}$, by $\mathcal{C}^{\widetilde{A}}$ we mean the class of languages decidable by a $\mathcal{C}$ machine that can query $\widetilde{A}_{m,\mathbb{F}}$ for any integer m and finite field $\mathbb{F}$. By $\mathcal{C}^{\widetilde{A}[\mathrm{poly}]}$ we mean the class of languages decidable by a $\mathcal{C}$ machine that, on inputs of length n, can query $\widetilde{A}_{m,\mathbb{F}}$ for any integer $m = O(\mathrm{poly}(n))$ and finite field with $|\mathbb{F}| = 2^{O(m)}$.

[5] www.complexityzoo.com

[6] Indeed, let A be any PSPACE-complete language. Then $\mathsf{P}^A = \mathsf{NP}^A$, but $\mathsf{EXP}^{A[\mathrm{poly}]} = \mathsf{NEXP}^{A[\mathrm{poly}]}$ if and only if $\mathsf{EXP} = \mathsf{NEXP}$ in the unrelativized world.

[7] All of our results would work equally well if we instead chose to limit $\operatorname{mdeg}(\widetilde{A}_{m,\mathbb{F}})$ by a linear or polynomial function of m. On the other hand, nowhere in this paper will $\operatorname{mdeg}(\widetilde{A}_{m,\mathbb{F}})$ need to be greater than 2.

We use $\mathrm{mdeg}(\widetilde{A})$ to denote the maximum multidegree of any $\widetilde{A}_m$.

For most of this paper, we will restrict ourselves to extensions over *finite fields*, as they are easier to work with than integer extensions and let us draw almost the same conceptual conclusions. We note that many of our results—including all results showing that existing results algebrize, and all oracle separations proved via communication complexity—easily carry over to the integer setting. Furthermore, even our oracle separations proved via direct construction can be "partly" carried over to the integer setting. Section 6 studies integer extensions in more detail.

Definition 2.3 (Algebrization) *We say the complexity class inclusion $\mathcal{C} \subseteq \mathcal{D}$ algebrizes if $\mathcal{C}^A \subseteq \mathcal{D}^{\widetilde{A}}$ for all oracles A and all finite field extensions $\widetilde{A}$ of A. Likewise, we say that $\mathcal{C} \subseteq \mathcal{D}$ does not algebrize, or that proving $\mathcal{C} \subseteq \mathcal{D}$ would require non-algebrizing techniques, if there exist $A, \widetilde{A}$ such that $\mathcal{C}^A \not\subset \mathcal{D}^{\widetilde{A}}$.*

We say the separation $\mathcal{C} \not\subset \mathcal{D}$ algebrizes if $\mathcal{C}^{\widetilde{A}} \not\subset \mathcal{D}^A$ for all $A, \widetilde{A}$. Likewise, we say that $\mathcal{C} \not\subset \mathcal{D}$ does not algebrize, or that proving $\mathcal{C} \not\subset \mathcal{D}$ would require non-algebrizing techniques, if there exist $A, \widetilde{A}$ such that $\mathcal{C}^{\widetilde{A}} \subseteq \mathcal{D}^A$.

When we examine the above definition, two questions arise. First, why can one complexity class access the extension $\widetilde{A}$, while the other class can only access the Boolean part A? And second, why is it the "right-hand class" that can access $\widetilde{A}$ for inclusions, but the "left-hand class" that can access $\widetilde{A}$ for separations?

The basic answer is that, under a more stringent notion of algebrization, we would not know how to prove that existing interactive proof results algebrize. So for example, while we will prove that $\mathsf{PSPACE}^{A[\mathrm{poly}]} \subseteq \mathsf{IP}^{\widetilde{A}}$ for all oracles A and extensions $\widetilde{A}$ of A, we do not know how to prove that $\mathsf{PSPACE}^{\widetilde{A}[\mathrm{poly}]} = \mathsf{IP}^{\widetilde{A}}$ for all $\widetilde{A}$.

Note that for our *separation* results, this issue seems to make no difference. For example, in Section 5 we will construct oracles A, B and extensions $\widetilde{A}, \widetilde{B}$, such that not only $\mathsf{P}^{\widetilde{A}} = \mathsf{NP}^{\widetilde{A}}$ and $\mathsf{P}^{\widetilde{B}} \neq \mathsf{NP}^{\widetilde{B}}$, but also $\mathsf{NP}^{\widetilde{A}} \subseteq \mathsf{P}^{A}$ and $\mathsf{NP}^{B} \not\subset \mathsf{P}^{\widetilde{B}}$. This implies that, even under our "broader" notion of algebrization, any resolution of the P versus NP problem will require non-algebrizing techniques.

3 Why Existing Techniques Algebrize

In this section, we go through a large number of non-relativizing results in complexity theory, and explain why they algebrize. The first batch consists of conditional collapses such as $\mathsf{P}^{\#\mathsf{P}} \subset \mathsf{P/poly} \implies \mathsf{P}^{\#\mathsf{P}} = \mathsf{MA}$, as well as containments such as $\mathsf{PSPACE} \subseteq \mathsf{IP}$ and $\mathsf{NEXP} \subseteq \mathsf{MIP}$. The second batch consists of circuit lower bounds, such as $\mathsf{MA}_{\mathsf{EXP}} \not\subset \mathsf{P/poly}$.

Note that each of the circuit lower bounds actually has a conditional collapse as its only non-relativizing ingredient. Therefore, once we show that the conditional collapses algebrize, we have already done most of the work of showing that the circuit lower bounds algebrize as well.

The section is organized as follows. First, in Section 3.1, we show that the *self-correctibility of* $\#\mathsf{P}$, proved by Lund et al. [23], is an algebrizing fact. From this it will follow, for example, that for all oracles A and finite field extensions $\widetilde{A}$,

$$\mathsf{PP}^{\widetilde{A}} \subset \mathsf{P}^{\widetilde{A}}/\mathsf{poly} \implies \mathsf{P}^{\#\mathsf{P}^A} \subseteq \mathsf{MA}^{\widetilde{A}}.$$

Next, in Section 3.2, we reuse results from Section 3.1 to show that the interactive proof results of Lund et al. [23] and Shamir [33] algebrize: that is, for all $A, \widetilde{A}$, we have $\mathsf{P}^{\#\mathsf{P}^A} \subseteq \mathsf{IP}^{\widetilde{A}}$, and indeed $\mathsf{PSPACE}^{A[\mathrm{poly}]} \subseteq \mathsf{IP}^{\widetilde{A}}$.

Then, in Section 3.3, we sketch an extension to the Babai-Fortnow-Lund theorem [3], giving us $\mathsf{NEXP}^{A[\mathrm{poly}]} \subseteq \mathsf{MIP}^{\widetilde{A}}$ for all $A, \widetilde{A}$. The same ideas also yield $\mathsf{EXP}^{A[\mathrm{poly}]} \subseteq \mathsf{MIP}_{\mathsf{EXP}}^{\widetilde{A}}$ for all $A, \widetilde{A}$, where $\mathsf{MIP}_{\mathsf{EXP}}$ is the subclass of MIP with the provers restricted to lie in EXP. This will imply, in particular, that

$$\mathsf{EXP}^{\widetilde{A}[\mathrm{poly}]} \subset \mathsf{P}^{\widetilde{A}}/\mathsf{poly} \implies \mathsf{EXP}^{A[\mathrm{poly}]} \subseteq \mathsf{MA}^{\widetilde{A}}$$

for all $A, \widetilde{A}$.

Section 3.4 harvests the consequences for circuit lower bounds. We show there that the results of Vinodchandran [37], Buhrman-Fortnow-Thierauf [6], and Santhanam [32] algebrize: that is, for all $A, \widetilde{A}$,

- $\mathsf{PP}^{\widetilde{A}} \not\subset \mathsf{SIZE}^{A}\left(n^{k}\right)$ for all constants k
- $\mathsf{MA}_{\mathsf{EXP}}^{\widetilde{A}} \not\subset \mathsf{P}^{A}/\mathsf{poly}$
- $\mathsf{PromiseMA}^{\widetilde{A}} \not\subset \mathsf{SIZE}^{A}\left(n^{k}\right)$ for all constants k

Finally, Section 3.5 discusses some miscellaneous interactive proof results, including that of Impagliazzo, Kabanets, and Wigderson [16] that $\mathsf{NEXP} \subset \mathsf{P}/\mathsf{poly} \implies \mathsf{NEXP} = \mathsf{MA}$, and that of Feige and Kilian [9] that $\mathsf{RG} = \mathsf{EXP}$.

Throughout the section, we assume some familiarity with the proofs of the results we are algebrizing.

3.1 Self-Correction for #P: Algebrizing

In this subsection we examine some non-relativizing properties of the classes $\#\mathsf{P}$ and PP, and show that these properties algebrize. Our goal will be to prove *tight* results, since that is what we will need later to show that Santhanam's lower bound $\mathsf{PromiseMA} \not\subset \mathsf{SIZE}\left(n^{k}\right)$ [32] is algebrizing. The need for tightness will force us to do a little more work than would otherwise be necessary.

The first step is to define a convenient $\#\mathsf{P}$-complete problem.

Definition 3.1 (#FSAT) *An FSAT formula over the finite field $\mathbb{F}$, in the variables $x_1, \ldots, x_N$, is a circuit with unbounded fan-in and fan-out 1, where every gate is labeled with either $+$ or $\times$, and every leaf is labeled with either an x_i or a constant $c \in \mathbb{F}$. Such a formula represents a polynomial $p : \mathbb{F}^N \to \mathbb{F}$ in the obvious way. The size of the formula is the number of gates.*

Now let $\#FSAT_{L,\mathbb{F}}$ be the following problem: given a polynomial $p : \mathbb{F}^N \to \mathbb{F}$ specified by an $FSAT$ formula of size at most L, evaluate the sum

$$S(p) := \sum_{x_1,\ldots,x_N \in \{0,1\}} p(x_1, \ldots, x_N).$$

Also, let $\#FSAT$ be the same problem but where the input has the form $\langle L, \mathbb{F}, p\rangle$ (i.e., L and $\mathbb{F}$ are given as part of the input). For the purpose of measuring time complexity, the size of an $\#FSAT$ instance is defined to be $n := L \log|\mathbb{F}|$.

Observe that if p is represented by an $FSAT$ formula of size L, then $\deg(p) \leq L$.

It is clear that $\#FSAT$ is $\#\mathsf{P}$-complete. Furthermore, Lund, Fortnow, Karloff, and Nisan [23] showed that $\#FSAT$ is *self-correctible*, in the following sense:

Theorem 3.2 ([23]) *There exists a polynomial-time randomized algorithm that, given any $\#FSAT_{L,\mathbb{F}}$ instance p with* $\operatorname{char}(\mathbb{F}) \geq 3L^2$ *and any circuit C:*

(i) Outputs $S(p)$ with certainty if C computes $\#FSAT_{L,\mathbb{F}}$.

(ii) Outputs either $S(p)$ or "FAIL" with probability at least 2/3, regardless of C.

Now let A be a Boolean oracle, and let $\widetilde{A}$ be a low-degree extension of A over $\mathbb{F}$. Then an $FSAT^{\widetilde{A}}$ formula is the same as an $FSAT$ formula, except that in addition to $+$ and $\times$ gates we also allow $\widetilde{A}$-*gates*: that is, gates with an arbitrary fan-in h, which take $b_1, \ldots, b_h \in \mathbb{F}$ as input and produce $\widetilde{A}_{h,\mathbb{F}}(b_1, \ldots, b_h)$ as output. Observe that if p is represented by an $FSAT^{\widetilde{A}}$ formula of size L, then $\deg(p) \leq L^2 \operatorname{mdeg}(\widetilde{A})$.

Let $\#FSAT^{\widetilde{A}}$ be the same problem as $\#FSAT$, except that the polynomial p is given by an $FSAT^{\widetilde{A}}$ formula. Then clearly $\#FSAT^{\widetilde{A}} \in \#\mathsf{P}^{\widetilde{A}}$. Also:

Proposition 3.3 *$\#FSAT^{\widetilde{A}}$ is $\#\mathsf{P}^A$-hard under randomized reductions.*

Proof. Let C^A be a Boolean circuit over the variables $z_1, \ldots, z_N$, with oracle access to A. Then a canonical $\#\mathsf{P}^A$-hard problem is to compute

$$\sum_{z_1,\ldots,z_N \in \{0,1\}} C^A(z_1, \ldots, z_N),$$

the number of satisfying assignments of C^A.

We will reduce this problem to $\#FSAT^{\widetilde{A}}$. For each gate g of C, define a variable x_g, which encodes whether g outputs 1. Then the polynomial p will simply be a product of terms that enforce "correct propagation" through the circuit. For example, if g computes the AND of gates i and j, then we encode the constraint $x_g = x_i \wedge x_j$ by the term

$$x_g x_i x_j + (1 - x_g)(1 - x_i x_j).$$

Likewise, if g is an oracle gate, then we encode the constraint $x_g = \widetilde{A}_{h,\mathbb{F}}(x_{i_1}, \ldots, x_{i_h})$ by the term

$$x_g \widetilde{A}_{h,\mathbb{F}}(x_{i_1}, \ldots, x_{i_h}) + (1 - x_g)\left(1 - \widetilde{A}_{h,\mathbb{F}}(x_{i_1}, \ldots, x_{i_h})\right).$$

The last step is to find a sufficiently large prime $q > 2^N$, one that will not affect the sum, to take as the order of $\mathbb{F}$. This can be done in randomized polynomial time. ■

By contrast, we do *not* know how to show that $\#FSAT^{\widetilde{A}}$ is $\#\mathsf{P}^{\widetilde{A}}$-hard—intuitively because of a $\#\mathsf{P}^{\widetilde{A}}$ machine's ability to query the $\widetilde{A}_{h,\mathbb{F}}$'s in ways that do not respect their structure as polynomials.

We now prove an algebrizing version of Theorem 3.2.

Theorem 3.4 *There exists a $\mathsf{BPP}^{\widetilde{A}}$ algorithm that, given any $\#FSAT^{\widetilde{A}}_{L,\mathbb{F}}$ instance p with* $\operatorname{char}(\mathbb{F}) \geq 3L^3 \operatorname{mdeg}(\widetilde{A})$ *and any circuit $C^{\widetilde{A}}$:*

(i) Outputs $S(p)$ if $C^{\widetilde{A}}$ computes $\#FSAT^{\widetilde{A}}_{L,\mathbb{F}}$.

(ii) Outputs either $S(p)$ or "FAIL" with probability at least 2/3, regardless of $C^{\widetilde{A}}$.

Proof. The proof is basically identical to the usual proof of Lund et al. [23] that $\#FSAT$ is self-correctible: that is, we use the circuit C to simulate the prover in an interactive protocol, whose goal is to convince the verifier of the value of $S(p)$. The only difference is that at the final step, we get an $FSAT^{\widetilde{A}}$ formula instead of an $FSAT$ formula, so we evaluate that formula with the help of the oracle $\widetilde{A}$.

In more detail, we first call $C^{\widetilde{A}}$ to obtain S', the claimed value of the sum $S(p)$. We then define

$$p_1(x) := \sum_{x_2,\ldots,x_N \in \{0,1\}} p(x, x_2, \ldots, x_N).$$

Then by making $\deg(p)+1$ more calls to $C^{\widetilde{A}}$, we can obtain p_1', the claimed value of p_1. We then check that $S' = p'(0) + p'(1)$. If this test fails, we immediately output "FAIL." Otherwise we choose $r_1 \in \mathbb{F}_q$ uniformly at random and set

$$p_2(x) := \sum_{x_3,\ldots,x_N \in \{0,1\}} p(r_1, x, x_3, \ldots, x_N).$$

We then use $\deg(p)+1$ more calls to $C^{\widetilde{A}}$ to obtain p_2', the claimed value of p_2, and check that $p_1'(r_1) = p_2'(0) + p_2'(1)$. If this test fails we output "FAIL"; otherwise we choose $r_2 \in \mathbb{F}$ uniformly at random and set

$$p_3(x) := \sum_{x_4,\ldots,x_N \in \{0,1\}} p(r_1, r_2, x, x_4, \ldots, x_N),$$

and continue in this manner until we reach the polynomial

$$p_N(x) := p(r_1, \ldots, r_{N-1}, x).$$

At this point we can evaluate $p_N(0)$ and $p_N(1)$ directly, by using the $FSAT^{\widetilde{A}}$ formula for p together with the oracle $\widetilde{A}$. We then check that $p_{N-1}'(r_{N-1}) = p_N(0) + p_N(1)$. If this final test fails then we output "FAIL"; otherwise we output $S(p) = S'$.

Completeness and soundness follow by the same analysis as in Lund et al. [23]. First, if $C^{\widetilde{A}}$ computes $\#FSAT^{\widetilde{A}}_{L,\mathbb{F}}$, then the algorithm outputs $S(p) = S'$ with certainty. Second, if $S(p) \neq S'$, then by the union bound, the probability that the algorithm is tricked into outputting $S(p) = S'$ is at most

$$\frac{L \deg(p)}{\operatorname{char}(\mathbb{F})} \leq \frac{L^3 \operatorname{mdeg}(\widetilde{A})}{3L^3 \operatorname{mdeg}(\widetilde{A})} = \frac{1}{3}.$$

■

From the self-correcting property of #P-complete problems, Lund et al. [23] deduced the corollary that $\mathsf{PP} \subset \mathsf{P/poly}$ implies $\mathsf{P}^{\#\mathsf{P}} = \mathsf{PP} = \mathsf{MA}$. We now wish to obtain an algebrizing version of their result. Thus, let $MAJFSAT^{\widetilde{A}}$ be the following decision version of $\#FSAT^{\widetilde{A}}$: given a $\#FSAT^{\widetilde{A}}$ instance $\langle L, \mathbb{F}, p\rangle$, together with an integer $k \in [\operatorname{char}(\mathbb{F})]$, decide whether $S(p) \geq k$ interpreted as an integer. Then clearly $MAJFSAT^{\widetilde{A}}$ is in $\mathsf{PP}^{\widetilde{A}}$ and hard for PP^{A}. We will also refer to $MAJFSAT^{\widetilde{A}}_{L,\mathbb{F}}$ in the case where L and $\mathbb{F}$ are fixed.

Theorem 3.5 *For all $A, \widetilde{A}$ and time-constructible functions s,*

$$MAJFSAT^{\widetilde{A}} \in \mathsf{SIZE}^{\widetilde{A}}(s(n)) \implies MAJFSAT^{\widetilde{A}} \in \mathsf{MATIME}^{\widetilde{A}}(s(n)\operatorname{poly}(n)).$$

So in particular, if $\mathsf{PP}^{\widetilde{A}} \subset \mathsf{P}^{\widetilde{A}}/\mathsf{poly}$ *then* $\mathsf{P}^{\#\mathsf{P}^{A}} \subseteq \mathsf{MA}^{\widetilde{A}}$.[8]

Proof. Given a procedure to solve $MAJFSAT_{L,\mathbb{F}}^{\widetilde{A}}$, it is clear that we can also solve $\#FSAT_{L,\mathbb{F}}^{\widetilde{A}}$, by calling the procedure $O(\log q)$ times and using binary search. (This is analogous to the standard fact that $\mathsf{P}^{\mathsf{PP}} = \mathsf{P}^{\#\mathsf{P}}$.) So if $MAJFSAT^{\widetilde{A}} \in \mathsf{SIZE}^{\widetilde{A}}(s(n))$, then an MA machine can first guess a circuit for $MAJFSAT_{L,\mathbb{F}}^{\widetilde{A}}$ of size $s(n)$, and then use that circuit to simulate the prover in an interactive protocol for $MAJFSAT_{L,\mathbb{F}}^{\widetilde{A}}$, exactly as in Theorem 3.4. This incurs at most polynomial blowup, and therefore places $MAJFSAT^{\widetilde{A}}$ in $\mathsf{MATIME}^{\widetilde{A}}(s(n)\,\mathrm{poly}(n))$.

In particular, if $\mathsf{PP}^{\widetilde{A}} \subset \mathsf{P}^{\widetilde{A}}/\mathsf{poly}$, then $MAJFSAT^{\widetilde{A}}$ is in $\mathsf{P}^{\widetilde{A}}/\mathsf{poly}$, hence $MAJFSAT^{\widetilde{A}}$ is in $\mathsf{MA}^{\widetilde{A}}$, hence $\mathsf{PP}^{A} \subseteq \mathsf{MA}^{\widetilde{A}}$, hence $\mathsf{P}^{\mathsf{PP}^{A}} = \mathsf{P}^{\#\mathsf{P}^{A}} \subseteq \mathsf{MA}^{\widetilde{A}}$. ■

3.2 $\mathsf{IP} = \mathsf{PSPACE}$: Algebrizing

Examining the proof of Theorem 3.4, it is not hard to see that the $\mathsf{P}^{\#\mathsf{P}} \subseteq \mathsf{IP}$ theorem of Lund et al. [23] algebrizes as well.

Theorem 3.6 *For all* $A, \widetilde{A}$, $\mathsf{P}^{\#\mathsf{P}^{A}} \subseteq \mathsf{IP}^{\widetilde{A}}$.

Proof. It suffices to note that, in the proof of Theorem 3.4, we actually gave an interactive protocol for $\#FSAT^{\widetilde{A}}$ where the verifier was in $\mathsf{BPP}^{\widetilde{A}}$. Since $\#FSAT^{\widetilde{A}}$ is $\#\mathsf{P}^{A}$-hard by Proposition 3.3, this implies the containment $\mathsf{P}^{\#\mathsf{P}^{A}} \subseteq \mathsf{IP}^{\widetilde{A}}$. ■

Indeed we can go further, and show that the famous $\mathsf{IP} = \mathsf{PSPACE}$ theorem of Shamir [33] is algebrizing.

Theorem 3.7 *For all* $A, \widetilde{A}$, $\mathsf{PSPACE}^{A[\mathrm{poly}]} \subseteq \mathsf{IP}^{\widetilde{A}}$.

Proof Sketch. When we generalize the $\#\mathsf{P}$ protocol of Lund et al. [23] to the PSPACE protocol of Shamir [33], the conversation between the prover and verifier becomes somewhat more complicated, due to the arithmetization of quantifiers. The prover now needs to prevent the degrees of the relevant polynomials from doubling at each iteration, which requires additional steps of degree reduction (e.g. "multilinearization" operators). However, the only step of the protocol that is relevant for algebrization is the last one, when the verifier checks that $p(r_1, \ldots, r_N)$ is equal to the value claimed by the prover for some $r_1, \ldots, r_N \in \mathbb{F}$. And this step can be algebrized exactly as in the $\#\mathsf{P}$ case. ■

3.3 $\mathsf{MIP} = \mathsf{NEXP}$: Algebrizing

Babai, Fortnow, and Lund [3] showed that $\mathsf{MIP} = \mathsf{NEXP}$. In this subsection we will sketch a proof that this result algebrizes:

Theorem 3.8 *For all* $A, \widetilde{A}$, $\mathsf{NEXP}^{A[\mathrm{poly}]} \subseteq \mathsf{MIP}^{\widetilde{A}}$.

[8] We could have avoided talking about $MAJFSAT$ at all in this theorem, had we been content to show that $\mathsf{PP}^{\widetilde{A}} \subset \mathsf{SIZE}^{\widetilde{A}}(s(n))$ implies $\mathsf{PP}^{\widetilde{A}} \subseteq \mathsf{MATIME}^{\widetilde{A}}(s(\mathrm{poly}(n)))$. But in that case, when we tried to show that Santhanam's result $\mathsf{PromiseMA} \not\subset \mathsf{SIZE}(n^k)$ was algebrizing, we would only obtain the weaker result $\mathsf{PromiseMATIME}^{\widetilde{A}}(n^{\mathrm{polylog}\,n}) \not\subset \mathsf{SIZE}^{A}(n^k)$.

To prove Theorem 3.8, we will divide Babai et al.'s proof into three main steps, and show that each of them algebrizes.

The first step is to define a convenient NEXP-complete problem.

Definition 3.9 (hSAT) *Let an h-formula over the variables $x_1, \ldots, x_n \in \{0,1\}$ be a Boolean formula consisting of AND, OR, and NOT gates, as well as gates of fan-in n that compute a Boolean function $h : \{0,1\}^n \rightarrow \{0,1\}$.*

Then given an h-formula C^h, let $hSAT$ be the problem of deciding whether there exists a Boolean function $h : \{0,1\}^n \rightarrow \{0,1\}$ such that $C^h(x) = 0$ for all $x \in \{0,1\}^n$.

Babai et al. showed the following:

Lemma 3.10 ([3]) *$hSAT$ is NEXP-complete.*

The proof of this lemma is very simple: h encodes both the nondeterministic guess of the NEXP machine on the given input, as well as the entire tableau of the computation with that guess. And the extension to circuits with oracle access is equally simple. Let A be a Boolean oracle, and let $hSAT^A$ be the variant of $hSAT$ where the formula $C^{h,A}$ can contain gates for both h and A. Then the first observation we make is that Lemma 3.10 relativizes: $hSAT^A$ is $\mathsf{NEXP}^{A[\mathrm{poly}]}$-complete. Indeed, h is constructed in exactly the same way. We omit the details.

The second step in Babai et al.'s proof is to use the LFKN protocol [23] to verify that $C^h(x) = 0$ for all x, *assuming* that the prover and verifier both have oracle access to a low-degree extension $\widetilde{h} : \mathbb{F}^n \rightarrow \mathbb{F}$ of h.

Lemma 3.11 ([3]) *Let $\widetilde{h} : \mathbb{F}^n \rightarrow \mathbb{F}$ be any low-degree extension of a Boolean function h. Then it is possible to verify, in $\mathsf{IP}^{\widetilde{h}}$, that $C^h(x) = 0$ for all $x \in \{0,1\}^n$.*

Proof Sketch. Observe that if we arithmetize C^h, then we get a low-degree polynomial $\widetilde{C^h} : \mathbb{F}^n \rightarrow \mathbb{F}$ extending C^h. Furthermore, $\widetilde{C^h}$ can be efficiently evaluated given oracle access to $\widetilde{h}$. So by using the LFKN protocol, the verifier can check that

$$\sum_{x \in \{0,1\}^n} \widetilde{C^h}(x) = \sum_{x \in \{0,1\}^n} C^h(x) = 0.$$

■

Our second observation is that Lemma 3.11 algebrizes: if we allow the prover and verifier oracle access to any low-degree extension $\widetilde{A}$ of A, then the same protocol works to ensure that $C^{h,A}(x) = 0$ for all $x \in \{0,1\}^n$.

In reality, of course, the verifier is *not* given oracle access to a low-degree extension $\widetilde{h}$. So the third step in Babai et al.'s proof is a low-degree test and subsequent self-correction algorithm, which allow the verifier to *simulate* oracle access to $\widetilde{h}$ by exchanging messages with two untrustworthy provers.

Lemma 3.12 ([3]) *There exists a BPP^B algorithm that, given any oracle $B : \mathbb{F}^n \rightarrow \mathbb{F}$ and input $y \in \mathbb{F}^n$:*

(i) Outputs $B(y)$ if B is a low-degree polynomial.

(ii) Outputs "FAIL" with probability $\Omega(1/\operatorname{poly}(n))$ if B differs from all low-degree polynomials on a $\Omega(1/\operatorname{poly}(n))$ fraction of points.

Combining Lemmas 3.11 and 3.12, we see that the verifier in the LFKN protocol does not need the *guarantee* that the oracle gates in $\widetilde{C^h}$, which are supposed to compute $\widetilde{h}$, indeed do so. A cheating prover will either be caught, or else the execution will be indistinguishable from one with a real $\widetilde{h}$.

Our final observation is that Lemma 3.12 deals only with the gates of $\widetilde{C^h}$ computing $\widetilde{h}$, and is completely independent of what other gates C has. It therefore algebrizes automatically when we switch to circuits containing oracle gates A. This completes the proof sketch of Theorem 3.8.

We conclude this section by pointing out one additional result. In Babai et al.'s original proof, if the language L to be verified is in EXP, then the function h encodes only the tableau of the computation. It can therefore be computed by the provers in EXP. Furthermore, if h is in EXP, then the unique multilinear extension $\widetilde{h} : \mathbb{F}^n \to \mathbb{F}$ is also in EXP. So letting $\mathsf{MIP}_{\mathsf{EXP}}$ be the subclass of MIP where the provers are in EXP, we get the following consequence:

Theorem 3.13 ([3]) $\mathsf{MIP}_{\mathsf{EXP}} = \mathsf{EXP}$.

Now, it is clear that if $L \in \mathsf{EXP}^{A[\mathrm{poly}]}$ then h and $\widetilde{h}$ can be computed by the provers in $\mathsf{EXP}^{A[\mathrm{poly}]}$. We therefore find that Theorem 3.13 algebrizes as well:

Theorem 3.14 *For all $A, \widetilde{A}$, $\mathsf{EXP}^{A[\mathrm{poly}]} \subseteq \mathsf{MIP}_{\mathsf{EXP}}^{\widetilde{A}}$.*

Theorem 3.14 has the following immediate corollary:

Corollary 3.15 *For all $A, \widetilde{A}$, if $\mathsf{EXP}^{\widetilde{A}[\mathrm{poly}]} \subset \mathsf{P}^{\widetilde{A}}/\mathsf{poly}$ then $\mathsf{EXP}^{A[\mathrm{poly}]} \subseteq \mathsf{MA}^{\widetilde{A}}$.*

Proof. If $\mathsf{EXP}^{\widetilde{A}[\mathrm{poly}]} \subset \mathsf{P}^{\widetilde{A}}/\mathsf{poly}$, then an $\mathsf{MA}^{\widetilde{A}}$ verifier can guess two polynomial-size circuits, and use them to simulate the $\mathsf{EXP}^{\widetilde{A}[\mathrm{poly}]}$ provers in an $\mathsf{MIP}_{\mathsf{EXP}}^{\widetilde{A}}$ protocol for $\mathsf{EXP}^{A[\mathrm{poly}]}$. ■

3.4 Recent Circuit Lower Bounds: Algebrizing

As mentioned earlier, Vinodchandran [37] showed that $\mathsf{PP} \not\subset \mathsf{SIZE}\left(n^k\right)$ for all constants k, and Aaronson [1] showed that this result fails to relativize. However, by using Theorem 3.5, we can now show that Vinodchandran's result algebrizes.

Theorem 3.16 *For all $A, \widetilde{A}$ and constants k, we have $\mathsf{PP}^{\widetilde{A}} \not\subset \mathsf{SIZE}^A\left(n^k\right)$.*

Proof. If $\mathsf{PP}^{\widetilde{A}} \not\subset \mathsf{P}^A/\mathsf{poly}$ then we are done, so assume $\mathsf{PP}^{\widetilde{A}} \subset \mathsf{P}^A/\mathsf{poly}$. Then certainly $\mathsf{PP}^{\widetilde{A}} \subset \mathsf{P}^{\widetilde{A}}/\mathsf{poly}$, so Theorem 3.5 implies that $\mathsf{P}^{\#\mathsf{P}^A} \subseteq \mathsf{MA}^{\widetilde{A}}$. Therefore $\left(\Sigma_2^{\mathsf{P}}\right)^A \subseteq \mathsf{MA}^{\widetilde{A}}$ as well, since Toda's Theorem [35] (which relativizes) tells us that $\Sigma_2^{\mathsf{P}} \subseteq \mathsf{P}^{\#\mathsf{P}}$ and hence $\left(\Sigma_2^{\mathsf{P}}\right)^A \subseteq \mathsf{P}^{\#\mathsf{P}^A}$. But Kannan's Theorem [19] (which also relativizes) tells us that $\Sigma_2^{\mathsf{P}} \not\subset \mathsf{SIZE}\left(n^k\right)$ for fixed k, and hence $\left(\Sigma_2^{\mathsf{P}}\right)^A \not\subset \mathsf{SIZE}^A\left(n^k\right)$. Therefore $\mathsf{MA}^{\widetilde{A}} \not\subset \mathsf{SIZE}^A\left(n^k\right)$. So since $\mathsf{MA} \subseteq \mathsf{PP}$ and this inclusion relativizes, $\mathsf{PP}^{\widetilde{A}} \not\subset \mathsf{SIZE}^A\left(n^k\right)$ as well. ■

In a similar vein, Buhrman, Fortnow, and Thierauf [6] showed that $\mathsf{MA}_{\mathsf{EXP}} \not\subset \mathsf{P}/\mathsf{poly}$, and also that this circuit lower bound fails to relativize. We now show that it algebrizes.

Theorem 3.17 *For all $A, \widetilde{A}$, we have $\mathsf{MA}_{\mathsf{EXP}}^{\widetilde{A}} \not\subset \mathsf{P}^A/\mathsf{poly}$.*

Proof. Suppose $\mathsf{MA}_{\mathsf{EXP}}^{\widetilde{A}} \subset \mathsf{P}^{A}/\mathsf{poly} \subseteq \mathsf{P}^{\widetilde{A}}/\mathsf{poly}$. Then certainly $\mathsf{PP}^{\widetilde{A}} \subset \mathsf{P}^{\widetilde{A}}/\mathsf{poly}$ as well, so Theorem 3.5 implies that $\mathsf{P}^{\#\mathsf{P}^{A}} \subseteq \mathsf{MA}^{\widetilde{A}}$. Hence we also have $\left(\Sigma_2^{\mathsf{P}}\right)^{A} \subseteq \mathsf{MA}^{\widetilde{A}}$ by Toda's Theorem [35], and hence $\left(\Sigma_2^{\mathsf{EXP}}\right)^{A} \subseteq \mathsf{MA}_{\mathsf{EXP}}^{\widetilde{A}}$ by padding. But Kannan's Theorem [19] tells us that $\left(\Sigma_2^{\mathsf{EXP}}\right)^{A} \not\subset \mathsf{P}^{A}/\mathsf{poly}$, so $\mathsf{MA}_{\mathsf{EXP}}^{\widetilde{A}} \not\subset \mathsf{P}^{A}/\mathsf{poly}$ as well. ■

Finally, Santhanam [32] recently showed that $\mathsf{PromiseMA} \not\subset \mathsf{SIZE}\left(n^k\right)$ for all constants k. Let us show that Santhanam's result algebrizes as well.[9]

Theorem 3.18 *For all $A, \widetilde{A}$ and constants k, we have* $\mathsf{PromiseMA}^{\widetilde{A}} \not\subset \mathsf{SIZE}^{A}\left(n^k\right)$.

Proof. First suppose $\mathsf{PP}^{\widetilde{A}} \subset \mathsf{P}^{\widetilde{A}}/\mathsf{poly}$. Then $\mathsf{P}^{\#\mathsf{P}^{A}} \subseteq \mathsf{MA}^{\widetilde{A}}$ by Theorem 3.5. Hence $\left(\Sigma_2^{\mathsf{P}}\right)^{A} \subseteq \mathsf{MA}^{\widetilde{A}}$ by Toda's Theorem [35], so by Kannan's Theorem [19] we have $\mathsf{MA}^{\widetilde{A}} \not\subset \mathsf{SIZE}^{A}\left(n^k\right)$ and are done.

Next suppose $\mathsf{PP}^{\widetilde{A}} \not\subset \mathsf{P}^{\widetilde{A}}/\mathsf{poly}$. Then there is some superpolynomial function s (not necessarily time-constructible) such that

$$MAJFSAT^{\widetilde{A}} \in \mathsf{SIZE}^{\widetilde{A}}\left(s\left(n\right)\right) \setminus \mathsf{SIZE}^{\widetilde{A}}\left(s\left(n\right)-1\right).$$

We define a promise problem $\left(L'_{\mathrm{YES}}, L'_{\mathrm{NO}}\right)$ by padding $MAJFSAT_n^{\widetilde{A}}$ as follows:

$$L'_{\mathrm{YES}} := \left\{x1^{s(n)^{1/2k}} : x \in MAJFSAT_n^{\widetilde{A}}\right\},$$
$$L'_{\mathrm{NO}} := \left\{x1^{s(n)^{1/2k}} : x \notin MAJFSAT_n^{\widetilde{A}}\right\}.$$

Our first claim is that $\left(L'_{\mathrm{YES}}, L'_{\mathrm{NO}}\right) \notin \mathsf{SIZE}^{A}\left(n^k\right)$. For suppose otherwise; then by ignoring the padding, we would obtain circuits for $MAJFSAT_n^{\widetilde{A}}$ of size

$$\left(n + s\left(n\right)^{1/2k}\right)^k \ll s\left(n\right),$$

contrary to assumption.

Our second claim is that $\left(L'_{\mathrm{YES}}, L'_{\mathrm{NO}}\right) \in \mathsf{PromiseMA}^{\widetilde{A}}$. This is because, on input $x1^{s(n)^{1/2k}}$, a $\mathsf{PromiseMA}^{\widetilde{A}}$ machine can guess a circuit for $MAJFSAT_n^{\widetilde{A}}$ of size $s\left(n\right)$, and then use Theorem 3.5 to verify that it works. ■

3.5 Other Algebrizing Results

Impagliazzo, Kabanets, and Wigderson [16] proved that $\mathsf{NEXP} \subset \mathsf{P}/\mathsf{poly}$ implies $\mathsf{NEXP} = \mathsf{MA}$. In the proof of this theorem, the only non-relativizing ingredient is the standard result that $\mathsf{EXP} \subset \mathsf{P}/\mathsf{poly}$ implies $\mathsf{EXP} = \mathsf{MA}$, which is algebrizing by Corollary 3.15. One can thereby show that the IKW theorem is algebrizing as well. More precisely, for all $A, \widetilde{A}$ we have

$$\mathsf{NEXP}^{\widetilde{A}[\mathrm{poly}]} \subset \mathsf{P}^{\widetilde{A}}/\mathsf{poly} \implies \mathsf{NEXP}^{A[\mathrm{poly}]} \subseteq \mathsf{MA}^{\widetilde{A}}.$$

Feige and Kilian [9] showed that $\mathsf{RG} = \mathsf{EXP}$, where RG is Refereed Games: informally, the class of languages L decidable by a probabilistic polynomial-time verifier that can interact (and exchange

[9]Note that Santhanam originally proved his result using a "tight" variant of the $\mathsf{IP} = \mathsf{PSPACE}$ theorem, due to Trevisan and Vadhan [36]. We instead use a tight variant of the LFKN theorem. However, we certainly expect that the Trevisan-Vadhan theorem, and the proof of Santhanam based on it, would algebrize as well.

private messages) with two competing provers, one trying to convince the verifier that $x \in L$ and the other that $x \notin L$. By analogy to $\mathsf{IP} = \mathsf{PSPACE}$ and $\mathsf{MIP} = \mathsf{NEXP}$, one would expect this theorem to algebrize. And indeed it does, but it turns out to relativize as well! Intuitively, this is because the RG protocol of Feige and Kilian involves only multilinear extensions of Turing machine tableaus, and not arithmetization as used (for example) in the $\mathsf{IP} = \mathsf{PSPACE}$ theorem. We omit the details.

4 Lower Bounds on Algebraic Query Complexity

What underlies our algebraic oracle separations is a new model of *algebraic query complexity.* In the standard query complexity model, an algorithm is trying to compute some property of a Boolean function $A : \{0,1\}^n \to \{0,1\}$ by querying A on various points. In our model, the function $A : \{0,1\}^n \to \{0,1\}$ will still be Boolean, but the algorithm will be allowed to query not just A, but also a low-degree extension $\widetilde{A} : \mathbb{F}^n \to \mathbb{F}$ of A over some field $\mathbb{F}$. In this section we develop the algebraic query complexity model in its own right, and prove several lower bounds in this model. Then, in Section 5, we apply our lower bounds to prove algebraic oracle separations. Section 6 will consider the variant where the algorithm can query an extension of A over the ring of integers.

Throughout this section we let $N = 2^n$. Algorithms will compute Boolean functions (properties) $f : \{0,1\}^N \to \{0,1\}$. An input A to f will be viewed interchangeably as an N-bit string $A \in \{0,1\}^N$, or as a Boolean function $A : \{0,1\}^n \to \{0,1\}$ of which the string is the truth table.

Let us recall some standard query complexity measures. Given a Boolean function $f : \{0,1\}^N \to \{0,1\}$, the *deterministic query complexity* of f, or $\mathrm{D}(f)$, is defined to be the minimum number of queries made by any deterministic algorithm that evaluates f on every input. Likewise, the (bounded-error) *randomized query complexity* $\mathrm{R}(f)$ is defined to be the minimum expected[10] number of queries made by any randomized algorithm that evaluates f with probability at least $2/3$ on every input. The bounded-error *quantum query complexity* $\mathrm{Q}(f)$ is defined analogously, with quantum algorithms in place of randomized ones. See Buhrman and de Wolf [8] for a survey of these measures.

We now define similar measures for algebraic query complexity. In our definition, an important parameter will be the multidegree of the allowed extension (recall that $\mathrm{mdeg}(p)$ is the largest degree of any of the variables of p). In all of our results, this parameter will be either 1 or 2.

Definition 4.1 (Algebraic Query Complexity Over Fields) *Let $f : \{0,1\}^N \to \{0,1\}$ be a Boolean function, let $\mathbb{F}$ be any field, and let c be a positive integer. Also, let $\mathcal{M}$ be the set of deterministic algorithms M such that $M^{\widetilde{A}}$ outputs $f(A)$ for every oracle $A : \{0,1\}^n \to \{0,1\}$ and every finite field extension $\widetilde{A} : \mathbb{F}^n \to \mathbb{F}$ of A with $\mathrm{mdeg}(\widetilde{A}) \leq c$. Then the deterministic algebraic query complexity of f over $\mathbb{F}$ is defined as*

$$\widetilde{\mathrm{D}}_{\mathbb{F},c}(f) := \min_{M \in \mathcal{M}} \max_{A,\widetilde{A} \,:\, \mathrm{mdeg}(\widetilde{A}) \leq c} T_M(\widetilde{A}),$$

where $T_M(\widetilde{A})$ is the number of queries to $\widetilde{A}$ made by $M^{\widetilde{A}}$. The randomized and quantum algebraic query complexities $\widetilde{\mathrm{R}}_{\mathbb{F},c}(f)$ and $\widetilde{\mathrm{Q}}_{\mathbb{F},c}(f)$ are defined similarly, except with (bounded-error) randomized and quantum algorithms in place of deterministic ones.

[10] Or the worst-case number of queries: up to the exact constant in the success probability, one can always ensure that this is about the same as the expected number.

4.1 Multilinear Polynomials

The construction of "adversary polynomials" in our lower bound proofs will require some useful facts about multilinear polynomials. In particular, the basis of delta functions for these polynomials will come in handy.

In what follows $\mathbb{F}$ is an arbitrary field (finite or infinite). Given a Boolean point z, define

$$\delta_z(x) := \prod_{i:z_i=1} x_i \prod_{i:z_i=0} (1 - x_i)$$

to be the unique multilinear polynomial that is 1 at z and 0 elsewhere on the Boolean cube. Then for an arbitrary multilinear polynomial $m : \mathbb{F}^n \to \mathbb{F}$, we can write m uniquely in the basis of δ_z's as follows:

$$m(x) = \sum_{z \in \{0,1\}^n} m_z \delta_z(x)$$

We will often identify a multilinear polynomial m with its coefficients m_z in this basis. Note that for any Boolean point z, the value $m(z)$ is simply the coefficient m_z in the above representation.

4.2 Lower Bounds by Direct Construction

We now prove lower bounds on algebraic query complexity over fields. The goal will be to show that querying points outside the Boolean cube is useless if one wants to gain information about values on the Boolean cube. In full generality, this is of course false (as witnessed by interactive proofs and PCPs on the one hand, and by the result of Juma et al. [17] on the other). To make our adversary arguments work, it will be crucial to give ourselves sufficient freedom, by using polynomials of multidegree 2 rather than multilinear polynomials.

We first prove deterministic lower bounds, which are quite simple, and then extend them to probabilistic lower bounds. Both work for the natural NP predicate of finding a Boolean point z such that $A(z) = 1$.

4.2.1 Deterministic Lower Bounds

Lemma 4.2 *Let $\mathbb{F}$ be a field and let $y_1, \ldots, y_t$ be points in $\mathbb{F}^n$. Then there exists a multilinear polynomial $m : \mathbb{F}^n \to \mathbb{F}$ such that*

(i) $m(y_i) = 0$ for all $i \in [t]$, and

(ii) $m(z) = 1$ for at least $2^n - t$ Boolean points z.

Proof. If we represent m as

$$m(x) = \sum_{z \in \{0,1\}^n} m_z \delta_z(x),$$

then the constraint $m(y_i) = 0$ for all $i \in [t]$ corresponds to t linear equations over $\mathbb{F}$ relating the 2^n coefficients m_z. By basic linear algebra, it follows that there must be a solution in which at least $2^n - t$ of the m_z's are set to 1, and hence $m(z) = 1$ for at least $2^n - t$ Boolean points z. ■

Lemma 4.3 *Let $\mathbb{F}$ be a field and let $y_1, \ldots, y_t$ be points in $\mathbb{F}^n$. Then for at least $2^n - t$ Boolean points $w \in \{0,1\}^n$, there exists a multiquadratic extension polynomial $p : \mathbb{F}^n \to \mathbb{F}$ such that* *(i)* $p(y_i) = 0$ *for all* $i \in [t]$,

(ii) $p(w) = 1$, *and*

(iii) $p(z) = 0$ *for all Boolean* $z \neq w$.

Proof. Let $m : \mathbb{F}^n \to \mathbb{F}$ be the multilinear polynomial from Lemma 4.2, and pick any Boolean w such that $m(w) = 1$. Then a multiquadratic extension polynomial p satisfying properties (i)-(iii) can be obtained from m as follows:

$$p(x) := m(x)\,\delta_w(x)\,.$$

■

Given a Boolean function $A : \{0,1\}^n \to \{0,1\}$, let the OR problem be that of deciding whether there exists an $x \in \{0,1\}^n$ such that $A(x) = 1$. Then Lemma 4.3 easily yields an exponential lower bound on the algebraic query complexity of the OR problem.

Theorem 4.4 $\widetilde{\mathrm{D}}_{\mathbb{F},2}(\mathrm{OR}) = 2^n$ *for every field* $\mathbb{F}$.

Proof. Let $\mathcal{Y}$ be the set of points queried by a deterministic algorithm, and suppose $|\mathcal{Y}| < 2^n$. Then Lemma 4.3 implies that there exists a multiquadratic extension polynomial $\widetilde{A} : \mathbb{F}^n \to \mathbb{F}$ such that $\widetilde{A}(y) = 0$ for all $y \in \mathcal{Y}$, but $\widetilde{A}(w) = 1$ for some Boolean w. So even if the algorithm is adaptive, we can let $\mathcal{Y}$ be the set of points it queries *assuming each query is answered with* 0, and then find $\widetilde{A}, \widetilde{B}$ such that $\widetilde{A}(y) = \widetilde{B}(y) = 0$ for all $y \in \mathcal{Y}$, but nevertheless $\widetilde{A}$ and $\widetilde{B}$ lead to different values of the OR function. ■

Again, the results of Juma et al. [17] imply that multidegree 2 is essential here, since for multilinear polynomials it is possible to solve the OR problem with only *one* query (over fields of characteristic greater than 2).

Though Lemma 4.3 sufficed for the basic query complexity lower bound, our oracle separations will require a more general result. The following lemma generalizes Lemma 4.3 in three ways: it handles extensions over many fields simultaneously instead of just one field; it lets us fix the queried points to any desired values instead of just zero; and it lets us toggle the values on many Boolean points instead of just the single Boolean point w.

Lemma 4.5 *Let* $\mathcal{F}$ *be a collection of fields (possibly with multiplicity). Let* $f : \{0,1\}^n \to \{0,1\}$ *be a Boolean function, and for every* $\mathbb{F} \in \mathcal{F}$, *let* $p_{\mathbb{F}} : \mathbb{F}^n \to \mathbb{F}$ *be a multiquadratic polynomial over* $\mathbb{F}$ *extending* f. *Also let* $\mathcal{Y}_{\mathbb{F}} \subseteq \mathbb{F}^n$ *for each* $\mathbb{F} \in \mathcal{C}$, *and* $t := \sum_{\mathbb{F} \in \mathcal{C}} |\mathcal{Y}_{\mathbb{F}}|$. *Then there exists a subset* $B \subseteq \{0,1\}^n$, *with* $|B| \leq t$, *such that for all Boolean functions* $f' : \{0,1\}^n \to \{0,1\}$ *that agree with* f *on* B, *there exist multiquadratic polynomials* $p'_{\mathbb{F}} : \mathbb{F}^n \to \mathbb{F}$ *(one for each* $\mathbb{F} \in \mathcal{F}$*) such that*

(i) $p'_{\mathbb{F}}$ *extends* f', *and*

(ii) $p'_{\mathbb{F}}(y) = p_{\mathbb{F}}(y)$ *for all* $y \in \mathcal{Y}_{\mathbb{F}}$.

Proof. Call a Boolean point z *good* if for every $\mathbb{F} \in \mathcal{F}$, there exists a multiquadratic polynomial $u_{\mathbb{F},z} : \mathbb{F}^n \to \mathbb{F}$ such that

(i') $u_{\mathbb{F},z}(y) = 0$ for all $y \in \mathcal{Y}_{\mathbb{F}}$,

(ii') $u_{\mathbb{F},z}(z) = 1$, and

(iii') $u_{\mathbb{F},z}(w) = 0$ for all Boolean $w \neq z$.

Then by Lemma 4.3, each $\mathbb{F} \in \mathcal{F}$ can prevent at most $|\mathcal{Y}_{\mathbb{F}}|$ points from being good. Hence there are at least $2^n - t$ good points.

Now let G be the set of all good points, and $B = \{0,1\}^n \setminus G$ be the set of all "bad" points. Then for all $\mathbb{F} \in \mathcal{F}$, we can obtain a polynomial $p'_{\mathbb{F}}$ satisfying (i) and (ii) as follows:

$$p'_{\mathbb{F}}(x) := p_{\mathbb{F}}(x) + \sum_{z \in G} \left(f'(z) - f(z)\right) u_{\mathbb{F},z}(x).$$

■

4.2.2 Probabilistic Lower Bounds

We now prove a lower bound for randomized algorithms. As usual, this will be done via the Yao minimax principle, namely by constructing a distribution over oracles which is hard for every deterministic algorithm that queries few points. Results in this subsection are only for *finite* fields, the reason being that they allow a *uniform* distribution over sets of all polynomials with given restrictions.

Lemma 4.6 *Let $\mathbb{F}$ be a finite field. Also, for all $w \in \{0,1\}^n$, let D_w be the uniform distribution over multiquadratic polynomials $p : \mathbb{F}^n \to \mathbb{F}$ such that $p(w) = 1$ and $p(z) = 0$ for all Boolean $z \neq w$. Suppose an adversary chooses a "marked point" $w \in \{0,1\}^n$ uniformly at random, and then chooses p according to D_w. Then any deterministic algorithm, after making t queries to p, will have queried w with probability at most $t/2^n$.*

Proof. Let $y_i \in \mathbb{F}^n$ be the i^{th} point queried, so that $y_1, \ldots, y_t$ is the list of points queried by step t. Then as in Lemma 4.5, call a Boolean point z *good* if there exists a multiquadratic polynomial $u : \mathbb{F}^n \to \mathbb{F}$ such that

(i) $u(y_i) = 0$ for all $i \in [t]$,

(ii) $u(z) = 1$, and

(iii) $u(z') = 0$ for all Boolean $z' \neq z$.

Otherwise call z *bad*. Let G_t be the set of good points immediately after the t^{th} step, and let $B_t = \{0,1\}^n \setminus G_t$ be the set of bad points. Then it follows from Lemma 4.3 that $|G_t| \geq 2^n - t$, and correspondingly $|B_t| \leq t$. Also notice that $B_t \subseteq B_{t+1}$ for all t.

For every good point $z \in \{0,1\}^n$, fix a "canonical" multiquadratic polynomial u_z that satisfies properties (i)-(iii) above. Also, for every Boolean point z, let V_z be the set of multiquadratic polynomials $v : \mathbb{F}^n \to \mathbb{F}$ such that

(i') $v(y_i) = p(y_i)$ for all $i \in [t]$,

(ii') $v(z) = 1$, and

(iii') $v(z') = 0$ for all Boolean $z' \neq z$.

Now let $x, x' \in G_t$ be any two good points.

Claim 4.7 *Even conditioned on the values of $p(y_1), \ldots, p(y_t)$, the probability that $p(x) = 1$ is equal to the probability that $p(x') = 1$.*

To prove Claim 4.7, it suffices to show that $|V_x| = |V_{x'}|$. We will do so by exhibiting a one-to-one correspondence between V_x and $V_{x'}$. Our correspondence is simply the following:

$$v \in V_x \Longleftrightarrow v + u_{x'} - u_x \in V_{x'}.$$

Now imagine that at every step i, all points in B_i are automatically queried "free of charge." This assumption can only help the algorithm, and hence make our lower bound stronger.

Claim 4.8 *Suppose that by step t, the marked point w still has not been queried. Then the probability that w is queried in step $t+1$ is at most*

$$\frac{|B_{t+1}| - |B_t|}{2^n - |B_t|}.$$

To prove Claim 4.8, notice that after t steps, there are $2^n - |B_t|$ points still in G_t—and by Claim 4.7, any of those points is as likely to be w as any other. Furthermore, there are at most $|B_{t+1}| - |B_t|$ points queried in step $t+1$ query that were not queried previously. For there are $|B_{t+1}| - |B_t|$ points in $B_{t+1} \setminus B_t$ that are queried "free of charge," plus one point y_{t+1} that is queried explicitly by the algorithm. Naïvely this would give $|B_{t+1}| - |B_t| + 1$, but notice further that if y_{t+1} is Boolean, then $y_{t+1} \in B_{t+1}$.

Now, the probability that the marked point was *not* queried in steps 1 through t is just $1 - |B_t|/2^n$. Therefore, the total probability of having queried w after t steps is

$$\sum_{i=0}^{t-1} \left(1 - \frac{|B_i|}{2^n}\right)\left(\frac{|B_{i+1}| - |B_i|}{2^n - |B_i|}\right) = \sum_{i=0}^{t-1} \frac{|B_{i+1}| - |B_i|}{2^n} \leq \frac{t}{2^n}.$$

■

An immediate corollary of Lemma 4.6 is that, over a finite field, randomized algebraic query algorithms do no better than deterministic ones at evaluating the OR function.

Theorem 4.9 $\widetilde{\mathrm{R}}_{\mathbb{F},2}(\mathrm{OR}) = \Omega(2^n)$ *for every finite field $\mathbb{F}$.*

To give an algebraic oracle separation between NP and BPP, we will actually need a slight extension of Lemma 4.6, which can be proven similarly to Lemma 4.5 (we omit the details).

Lemma 4.10 *Given a finite field $\mathbb{F}$ and string $w \in \{0,1\}^n$, let $D_{w,\mathbb{F}}$ be the uniform distribution over multiquadratic polynomials $p : \mathbb{F}^n \to \mathbb{F}$ such that $p(w) = 1$ and $p(z) = 0$ for all Boolean $z \neq w$. Suppose an adversary chooses $w \in \{0,1\}^n$ uniformly at random, and then for every finite field $\mathbb{F}$, chooses $p_{\mathbb{F}}$ according to $D_{w,\mathbb{F}}$. Then any algorithm, after making t queries to any combination of $p_{\mathbb{F}}$'s, will have queried w with probability at most $t/2^n$.*

4.3 Lower Bounds by Communication Complexity

In this section we point out a simple connection between algebraic query complexity and communication complexity. Specifically, we show that *algebraic* query algorithms can be efficiently simulated by *Boolean* communication protocols. This connection will allow us to derive many lower bounds on algebraic query complexity that we do not know how to prove with the direct techniques of the previous section. Furthermore, it will give lower bounds even for *multilinear* extensions, and even for extensions over the integers. The drawbacks are that (1) the functions for which we obtain the lower bounds are somewhat more complicated (for example, Disjointness instead of OR), and (2) this technique does not seem useful for proving algebraic oracle *collapses* (such as $\mathsf{NP}^{\widetilde{A}} \subset \mathsf{SIZE}^{A}(n)$).

For concreteness, we first state our "transfer principle" for deterministic query and communication complexities—but as we will see, the principle is much broader.

Theorem 4.11 *Let $A : \{0,1\}^n \to \{0,1\}$ be a Boolean function, and let $\widetilde{A} : \mathbb{F}_q^n \to \mathbb{F}_q$ be the unique multilinear extension of A over a finite field $\mathbb{F}$. Suppose one can evaluate some Boolean predicate f of A using T deterministic adaptive queries to $\widetilde{A}$. Also, let A_0 and A_1 be the subfunctions of A obtained by restricting the first bit to 0 or 1 respectively. Then if Alice is given the truth table of A_0 and Bob is given the truth table of A_1, they can jointly evaluate $f(A)$ using $O(Tn \log |\mathbb{F}|)$ bits of communication.*

Proof. Given any point $y \in \mathbb{F}^n$, we can write $\widetilde{A}(y)$ as a linear combination of the values taken by A on the Boolean cube, like so:

$$\widetilde{A}(y) = \sum_{z \in \{0,1\}^n} \delta_z(y) A(z).$$

Now let M be an algorithm that evaluates f using T queries to $\widetilde{A}$. Our communication protocol will simply perform a step-by-step simulation of M, as follows.

Let $y_1 \in \mathbb{F}^n$ be the first point queried by M. Then Alice computes the partial sum

$$\widetilde{A}_0(y_1) := \sum_{z \in \{0,1\}^{n-1}} \delta_{0z}(y_1) A(0z)$$

and sends $(y_1, \widetilde{A}_0(y_1))$ to Bob. Next Bob computes

$$\widetilde{A}_1(y_1) := \sum_{z \in \{0,1\}^{n-1}} \delta_{1z}(y_1) A(1z),$$

from which he learns $\widetilde{A}(y_1) = \widetilde{A}_0(y_1) + \widetilde{A}_1(y_1)$. Bob can then determine y_2, the second point queried by M given that the first query had outcome $\widetilde{A}(y_1)$. So next Bob computes $\widetilde{A}_1(y_2)$ and sends $(y_2, \widetilde{A}_1(y_2))$ to Alice. Next Alice computes $\widetilde{A}(y_2) = \widetilde{A}_0(y_2) + \widetilde{A}_1(y_2)$, determines y_3, and sends $(y_3, \widetilde{A}_0(y_3))$ to Bob, and so on for T rounds.

Each message uses $O(n \log |\mathbb{F}|)$ bits, from which it follows that the total communication cost is $O(Tn \log |\mathbb{F}|)$. ■

In proving Theorem 4.11, notice that we never needed the assumption that M was deterministic. Had M been randomized, our simulation would have produced a randomized protocol; had M been quantum, it would have produced a quantum protocol; had M been an MA machine, it would have produced an MA protocol, and so on.

To illustrate the power of Theorem 4.11, let us now prove a lower bound on algebraic query complexity without using anything about polynomials.

Given two Boolean strings $x = x_1 \ldots x_N$ and $y = y_1 \ldots y_N$, recall that the Disjointness problem is to decide whether there exists an index $i \in [N]$ such that $x_i = y_i = 1$. Supposing that Alice holds x and Bob holds y, Kalyasundaram and Schnitger [18] showed that any randomized protocol to solve this problem requires Alice and Bob to exchange $\Omega(N)$ bits (see also the simpler proof by Razborov [29]).

In our setting, the problem becomes the following: given a Boolean function $A : \{0,1\}^n \to \{0,1\}$, decide whether there exists an $x \in \{0,1\}^{n-1}$ such that $A(0x) = A(1x) = 1$. Call this problem DISJ, and suppose we want to solve DISJ using a randomized algorithm that queries the multilinear extension $\widetilde{A} : \mathbb{F}^n \to \mathbb{F}$ of A. Then Theorem 4.11 immediately yields a lower bound on the number of queries to $\widetilde{A}$ that we need:

Theorem 4.12 $\widetilde{\mathrm{R}}_{\mathbb{F},1}(\mathrm{DISJ}) = \Omega\left(\frac{2^n}{n \log|\mathbb{F}|}\right)$ *for all finite fields* $\mathbb{F}$.

Proof. Suppose $\widetilde{\mathrm{R}}_{\mathbb{F},1}(\mathrm{DISJ}) = o\left(\frac{2^n}{n \log|\mathbb{F}|}\right)$. Then by Theorem 4.11, we get a randomized protocol for the Disjointness problem with communication cost $o(N)$, where $N = 2^{n-1}$. But this contradicts the lower bound of Razborov [29] and Kalyasundaram and Schnitger [18] mentioned above. ■

In Section 5, we will use the transfer principle to convert many known communication complexity results into algebraic oracle separations.

5 The Need for Non-Algebrizing Techniques

In this section we show formally that solving many of the open problems in complexity theory will require non-algebrizing techniques. We have already done much of the work in Section 4, by proving lower bounds on algebraic query complexity. What remains is to combine these query complexity results with diagonalization or forcing arguments, in order to achieve the oracle separations and collapses we want.

5.1 Non-Algebrizing Techniques Needed for P vs. NP

We start with an easy but fundamental result: that *any proof of* $\mathsf{P} \neq \mathsf{NP}$ *will require non-algebrizing techniques.*

Theorem 5.1 *There exist* $A, \widetilde{A}$ *such that* $\mathsf{NP}^{\widetilde{A}} \subseteq \mathsf{P}^A$.

Proof. Let A be any PSPACE-complete language, and let $\widetilde{A}$ be the unique multilinear extension of A. As observed by Babai, Fortnow, and Lund [3], the multilinear extension of any PSPACE language is also in PSPACE. So as in the usual argument of Baker, Gill, and Solovay [4], we have $\mathsf{NP}^{\widetilde{A}} = \mathsf{NP}^{\mathsf{PSPACE}} = \mathsf{PSPACE} = \mathsf{P}^A$. ■

The same argument immediately implies that any proof of $\mathsf{P} \neq \mathsf{PSPACE}$ will require non-algebrizing techniques:

Theorem 5.2 *There exist* $A, \widetilde{A}$ *such that* $\mathsf{PSPACE}^{\widetilde{A}[\mathrm{poly}]} = \mathsf{P}^A$.

Next we show that any proof of $\mathsf{P} = \mathsf{NP}$ would require non-algebrizing techniques, by giving an algebraic oracle *separation* between P and NP. As in the original work of Baker, Gill, and Solovay [4], this direction is the harder of the two.

Theorem 5.3 *There exist* $A, \widetilde{A}$ *such that* $\mathsf{NP}^A \not\subset \mathsf{P}^{\widetilde{A}}$. *Furthermore, the language* L *that achieves the separation simply corresponds to deciding, on inputs of length* n*, whether there exists a* $w \in \{0,1\}^n$ *with* $A_n(w) = 1$.

Proof. Our proof closely follows the usual diagonalization argument of Baker, Gill, and Solovay [4], except that we have to use Lemma 4.5 to handle the fact that P can query a low-degree extension.

For every n, the oracle A will contain a Boolean function $A_n : \{0,1\}^n \to \{0,1\}$, while $\widetilde{A}$ will contain a multiquadratic extension $\widetilde{A}_{n,\mathbb{F}} : \mathbb{F}^n \to \mathbb{F}$ of A_n for every n and finite field $\mathbb{F}$. Let L be the unary language consisting of all strings 1^n for which there exists a $w \in \{0,1\}^n$ such that $A_n(w) = 1$. Then clearly $L \in \mathsf{NP}^A$ for all A. Our goal is to choose $A, \widetilde{A}$ so that $L \notin \mathsf{P}^{\widetilde{A}}$.

Let $M_1, M_2, \ldots$ be an enumeration of $\mathsf{DTIME}(n^{\log n})$ oracle machines. Also, let $M_i(n) = 1$ if M_i accepts on input 1^n and $M_i(n) = 0$ otherwise, and let $L(n) = 1$ if $1^n \in L$ and $L(n) = 0$ otherwise. Then it suffices to ensure that for every i, there exists an n such that $M_i(n) \neq L(n)$.

The construction of $\widetilde{A}$ proceeds in stages. At stage i, we assume that $L(1), \ldots, L(i-1)$ are already fixed, and that for each $j < i$, we have already found an n_j such that $M_j(n_j) \neq L(n_j)$. Let S_j be the set of all indices n such that some $\widetilde{A}_{n,\mathbb{F}}$ is queried by M_j on input 1^{n_j}. Let $T_i := \bigcup_{j<i} S_j$. Then for all $n \in T_i$, we consider every $\widetilde{A}_{n,\mathbb{F}}$ to be "fixed": that is, it will not change in stage i or any later stage.

Let n_i be the least n such that $n \notin T_i$ and $2^n > n^{\log n}$. Then simulate the machine M_i on input 1^{n_i}, with the oracle behaving as follows:

(i) If M_i queries some $\widetilde{A}_{n,\mathbb{F}}(y)$ with $n \in T_i$, return the value that was fixed in a previous stage.

(ii) If M_i queries some $\widetilde{A}_{n,\mathbb{F}}(y)$ with $n \notin T_i$, return 0.

Once M_i halts, let S_i be the set of all n such that M_i queried some $\widetilde{A}_{n,\mathbb{F}}$. Then for all $n \in S_i \setminus T_i$ other than n_i, and all $\mathbb{F}$, fix $\widetilde{A}_{n,\mathbb{F}} := 0$ to be the identically-zero polynomial. As for n_i itself, there are two cases. If M_i accepted on input 1^{n_i}, then fix $\widetilde{A}_{n_i,\mathbb{F}} := 0$ for all $\mathbb{F}$, so that $L(n_i) = 0$. On the other hand, if M_j rejected, then for all $\mathbb{F}$, let $\mathcal{Y}_{\mathbb{F}}$ be the set of all $y \in \mathbb{F}^{n_i}$ that M_i queried. We have $\sum_{\mathbb{F}} |\mathcal{Y}_{\mathbb{F}}| \leq n^{\log n}$. So by Lemma 4.5, there exists a Boolean point $w \in \{0,1\}^{n_i}$ such that for all $\mathbb{F}$, we can fix $\widetilde{A}_{n_i,\mathbb{F}} : \mathbb{F}^{n_i} \to \mathbb{F}$ to be a multiquadratic polynomial such that

(i') $\widetilde{A}_{n_i,\mathbb{F}}(y) = 0$ for all $y \in \mathcal{Y}_{\mathbb{F}}$,

(ii') $\widetilde{A}_{n_i,\mathbb{F}}(w) = 1$, and

(iii') $\widetilde{A}_{n_i,\mathbb{F}}(w) = 0$ for all Boolean $z \neq w$.

We then have $L(n_i) = 1$, as desired. ■

In the proof of Theorem 5.3, if we simply replace $2^n > n^{\log n}$ by the stronger condition $2^{n-1} > n^{\log n}$, then an RP algorithm can replace the NP one. Thus, we immediately get the stronger result that there exist $A, \widetilde{A}$ such that $\mathsf{RP}^A \not\subset \mathsf{P}^{\widetilde{A}}$.

5.2 Non-Algebrizing Techniques Needed for NP vs. BPP

We now show an algebraic oracle separation between NP and BPP. This result implies that any proof of $\mathsf{NP} \subseteq \mathsf{BPP}$ would require non-algebrizing techniques—or to put it more concretely, *there is no way to solve 3SAT in probabilistic polynomial time, by first arithmetizing a 3SAT formula and then treating the result as an arbitrary low-degree black-box polynomial.*

Theorem 5.4 *There exist $A, \widetilde{A}$ such that $\mathsf{NP}^A \not\subset \mathsf{BPP}^{\widetilde{A}}$. Furthermore, the language L that achieves the separation simply corresponds to finding a $w \in \{0,1\}^n$ with $A_n(w) = 1$.*

Proof. Our proof closely follows the proof of Bennett and Gill [5] that $\mathsf{P}^A \neq \mathsf{NP}^A$ with probability 1 over A.

Similarly to Lemma 4.10, given a Boolean point w and a finite field $\mathbb{F}$, let $D_{n,w,\mathbb{F}}$ be the uniform distribution over all multiquadratic polynomials $p : \mathbb{F}^n \to \mathbb{F}$ such that $p(w) = 1$ and $p(z) = 0$ for all Boolean $z \neq w$. Then we generate the oracle $\widetilde{A}$ according to following distribution. For each $n \in \mathbb{N}$, first draw $w_n \in \{0,1\}^n$ uniformly at random, and set $A_n(w_n) = 1$ and $A_n(z) = 0$ for all n-bit Boolean strings $z \neq w_n$. Next, for every finite field $\mathbb{F}$, draw the extension $\widetilde{A}_{n,\mathbb{F}}$ of A_n from $D_{n,w_n,\mathbb{F}}$.

We define the language L as follows: $0^i 1^{n-i} \in L$ if and only if the i^{th} bit of w_n is a 1, and $x \notin L$ for all x not of the form $0^i 1^{n-i}$. Clearly $L \in \mathsf{NP}^A$. Our goal is to show that $L \notin \mathsf{BPP}^{\widetilde{A}}$ with probability 1 over the choice of $\widetilde{A}$.

Fix a BPP oracle machine M. Then let $E_{M,n,i}$ be the event that M correctly decides whether $0^i 1^{n-i} \in L$, with probability at least 2/3 over M's internal randomness, and let

$$E_{M,n} := E_{M,n,1} \wedge \cdots \wedge E_{M,n,n}.$$

Supposing $E_{M,n}$ holds, with high probability we can recover w_n in polynomial time, by simply running M several times on each input $0^i 1^{n-i}$ and then outputting the majority answer as the i^{th} bit of w_n. But Lemma 4.10 implies that after making t queries, we can guess w_n with probability at most

$$\frac{t}{2^n} + \frac{1}{2^n - t},$$

just as if we had oracle access only to A_n and not to the extensions $\widetilde{A}_{n,\mathbb{F}}$.

So given n, choose another input size $n' \gg n$ which is so large that on inputs of size n or less, M cannot have queried $\widetilde{A}_{n',\mathbb{F}}$ for any $\mathbb{F}$ (for example, $n' = 2^{2^n}$ will work for sufficiently large n). Then for all sufficiently large n, we must have

$$\Pr_{\widetilde{A}}\left[E_{M,n'} \mid E_{M,1} \wedge \cdots \wedge E_{M,n}\right] \leq \frac{1}{3}.$$

This implies that

$$\Pr_{\widetilde{A}}\left[E_{M,1} \wedge E_{M,2} \wedge \cdots\right] = 0.$$

But since there is only a countable infinity of BPP machines, by the union bound we get

$$\Pr_{\widetilde{A}}\left[\exists M : E_{M,1} \wedge E_{M,2} \wedge \cdots\right] = 0$$

which is what we wanted to show. ■

Theorem 5.4 readily extends to show any proof of $\mathsf{NP} \subset \mathsf{P/poly}$ would require non-algebrizing techniques:

Theorem 5.5 *There exist* $A, \widetilde{A}$ *such that* $\mathsf{NP}^A \not\subset \mathsf{P}^{\widetilde{A}}/\mathsf{poly}$.

Proof Sketch. Suppose we have a $\mathsf{P}^{\widetilde{A}}/\mathsf{poly}$ machine that decides a language $L \in \mathsf{NP}^A$ using an advice string of size n^k. Then by *guessing* the advice string, we get a $\mathsf{BPP}^{\widetilde{A}}$ machine that decides L on all inputs with probability $\Omega(2^{-n^k})$. We can then run the $\mathsf{BPP}^{\widetilde{A}}$ machine sequentially on (say) n^{2k} inputs $x_1, \ldots, x_{n^{2k}}$, and decide all of them with a greater probability than is allowed by the proof of Theorem 5.4.[11] ■

[11] Because of the requirement that the $\mathsf{BPP}^{\widetilde{A}}$ machine operates *sequentially*—i.e., that it outputs the answer for each input x_t before seeing the next input x_{t+1}—there is no need here for a direct product theorem. On the other hand, proving direct product theorems for algebraic query complexity is an interesting open problem.

5.3 Non-Algebrizing Techniques Needed for Circuit Lower Bounds

We now give an oracle A and extension $\widetilde{A}$ such that $\mathsf{NEXP}^{\widetilde{A}} \subset \mathsf{P}^{A}/\mathsf{poly}$. This implies that any proof of $\mathsf{NEXP} \not\subset \mathsf{P}/\mathsf{poly}$ will require non-algebrizing techniques.

Theorem 5.6 *There exist $A, \widetilde{A}$ such that $\mathsf{NTIME}^{\widetilde{A}}(2^n) \subset \mathsf{SIZE}^{A}(n)$.*

Proof. Let $M_1, M_2, \ldots$ be an enumeration of $\mathsf{NTIME}(2^n)$ oracle machines. Then on inputs of size n, it suffices to simulate $M_1, \ldots, M_n$, since then every M_i will be simulated on all but finitely many input lengths.

For simplicity, we will assume that on inputs of size n, the M_i's can query only a single polynomial, $p : \mathbb{F}^{4n} \to \mathbb{F}$. Later we will generalize to the case where the M_i's can query $\widetilde{A}_{n,\mathbb{F}}$ for every n and $\mathbb{F}$ simultaneously.

We construct p by an iterative process. We are dealing with $n2^n$ pairs of the form $\langle i, x\rangle$, where $x \in \{0,1\}^n$ is an input and $i \in [n]$ is the label of a machine. At every iteration, each $\langle i, x\rangle$ will be either *satisfied* or *unsatisfied*, and each point in $\mathbb{F}^{4n}$ will be either *active* or *inactive*. Initially all $\langle i, x\rangle$'s are unsatisfied and all points are active.

To *fix* an active point y will mean we fix the value of $p(y)$ to some constant c_y, and switch y from active to inactive. Once y is inactive, it never again becomes active, and $p(y)$ never again changes.

We say that y is fixed *consistently*, if after it is fixed there still exists a multiquadratic extension polynomial $p : \mathbb{F}^{4n} \to \mathbb{F}$ such that $p(y) = c_y$ for all inactive points y. Then the iterative process consists of repeatedly asking the following question:

Does there exist an unsatisfied $\langle i, x\rangle$, such that by consistently fixing at most 2^n active points, we can force M_i to accept on input x?

If the answer is yes, then we fix those points, switch $\langle i, x\rangle$ from unsatisfied to satisfied, and repeat. We stop only when we can no longer find another $\langle i, x\rangle$ to satisfy.

Let D be the set of inactive points when this process halts. Then $|D| \le n2^{2n}$. So by Lemma 4.5, there exists a subset $G \subseteq \{0,1\}^{4n}$, with $|G| \ge 2^{4n} - n2^{2n}$, such that for any Boolean function $f : \{0,1\}^{4n} \to \{0,1\}$, there exists a multiquadratic polynomial $p : \mathbb{F}^{4n} \to \mathbb{F}$ satisfying

(i) $p(y) = c_y$ for all $y \in D$,

(ii) $p(z) = f(z)$ for all $z \in G$, and

(iii) $p(z) \in \{0,1\}$ for all Boolean z.

To every machine-input pair $\langle i, x\rangle$, associate a unique string $w_{i,x} \in \{0,1\}^{4n}$ in some arbitrary way. Then for all $\langle i, x\rangle$ we have

$$\Pr_{z \in \{0,1\}^{4n}} [z \oplus w_{i,x} \in G] \ge 1 - \frac{n2^{2n}}{2^{4n}}.$$

So by the union bound, there exists a fixed string $z' \in \{0,1\}^{4n}$ such that $z' \oplus w_{i,x} \in G$ for all $\langle i, x\rangle$. We will choose the Boolean function f so that for every $\langle i, x\rangle$ pair, $f(z' \oplus w_{i,x})$ encodes whether or not M_i accepts on input x. Note that doing so cannot cause any additional $\langle i, x\rangle$ pairs to accept, for if it could, then we would have already forced those pairs to accept during the iterative process.

Our linear-size circuit for simulating the M_i's will now just hardwire the string z'.

Finally, let us generalize to the case where the M_i's can query $\widetilde{A}_{n,\mathbb{F}}$ for any input length n and finite field $\mathbb{F}$ of their choice. This requires only a small change to the original proof. We construct $\widetilde{A}$ in stages. At stage n, assume that $\widetilde{A}_{1,\mathbb{F}}, \ldots, \widetilde{A}_{n-1,\mathbb{F}}$ have already been fixed for every $\mathbb{F}$. Then our goal is to fix $\widetilde{A}_{n,\mathbb{F}}$ for every $\mathbb{F}$. Let $\mathcal{Y}_{\mathbb{F}}$ be the set of points in $\mathbb{F}^n$ for which the value of $\widetilde{A}_{n,\mathbb{F}}$ was fixed in one of the previous $n-1$ stages. Then

$$\sum_{\mathbb{F}} |\mathcal{Y}_{\mathbb{F}}| \leq \sum_{m=1}^{n-1} m2^{2m} \leq n2^{2n}.$$

So by Lemma 4.5, for all $\mathbb{F}$ we can find multiquadratic polynomials $\widetilde{A}_{n,\mathbb{F}} : \mathbb{F}^{4n} \to \mathbb{F}$ that satisfy all the forcing conditions, and that also encode in some secret location whether M_i accepts on input x for all $i \in [n]$ and $x \in \{0,1\}^n$. ■

By a standard padding argument, Theorem 5.6 immediately gives $A, \widetilde{A}$ such that $\mathsf{NEXP}^{\widetilde{A}} \subset \mathsf{P}^A/\mathsf{poly}$. This collapse is almost the best possible, since Theorem 3.17 implies that there do *not* exist $A, \widetilde{A}$ such that $\mathsf{MA}_{\mathsf{EXP}}^{\widetilde{A}} \subset \mathsf{P}^A/\mathsf{poly}$.

5.4 Non-Algebrizing Techniques Needed for Other Problems

We can use the communication complexity transfer principle from Section 4.3 to achieve many other separations.

Theorem 5.7 *There exist $A, \widetilde{A}$ such that*

(i) $\mathsf{NP}^A \not\subset \mathsf{BPP}^{\widetilde{A}}$,

(ii) $\mathsf{coNP}^A \not\subset \mathsf{MA}^{\widetilde{A}}$,

(iii) $\mathsf{P}^{\mathsf{NP}^A} \not\subset \mathsf{PP}^{\widetilde{A}}$,

(iv) $\mathsf{NP}^A \not\subset \mathsf{BQP}^{\widetilde{A}}$,

(v) $\mathsf{BQP}^A \not\subset \mathsf{BPP}^{\widetilde{A}}$, *and*

(vi) $\mathsf{QMA}^A \not\subset \mathsf{MA}^{\widetilde{A}}$.

Furthermore, for all of these separations $\widetilde{A}$ is simply the multilinear extension of A.

Proof Sketch. Let us first explain the general idea, before applying it to prove these separations. Given a complexity class $\mathcal{C}$, let $\mathcal{C}_{\mathsf{cc}}$ be the communication complexity analogue of $\mathcal{C}$: that is, the class of communication predicates $f : \{0,1\}^N \times \{0,1\}^N \to \{0,1\}$ that are decidable by a $\mathcal{C}$ machine using $O(\text{polylog } N)$ communication. Also suppose $\mathcal{C}^A \subseteq \mathcal{D}^{\widetilde{A}}$ for all oracles A and multilinear extensions $\widetilde{A}$ of A. Then the transfer principle (Theorem 4.11) would imply that $\mathcal{C}_{\mathsf{cc}} \subseteq \mathcal{D}_{\mathsf{cc}}$. Thus, if we know already that $\mathcal{C}_{\mathsf{cc}} \not\subset \mathcal{D}_{\mathsf{cc}}$, we can use that to conclude that there exist $A, \widetilde{A}$ such that $\mathcal{C}^A \not\subset \mathcal{D}^{\widetilde{A}}$.

We now apply this idea to prove the six separations listed above.

(i) Recall that Kalyasundaram and Schnitger [18] (see also [29]) proved an $\Omega(N)$ lower bound on the randomized communication complexity of the Disjointness predicate. From this, together with a standard diagonalization argument, one easily gets that $\mathsf{NP}_{\mathsf{cc}} \not\subset \mathsf{BPP}_{\mathsf{cc}}$. Hence there exist $A, \widetilde{A}$ such that $\mathsf{NP}^A \not\subset \mathsf{BPP}^{\widetilde{A}}$.

(ii) Klauck [21] has generalized the lower bound of [29, 18] to show that Disjointness has MA communication complexity $\Omega(\sqrt{N})$. From this it follows that $\mathsf{coNP}_{\mathsf{cc}} \not\subset \mathsf{MA}_{\mathsf{cc}}$, and hence $\mathsf{coNP}^{A} \not\subset \mathsf{MA}^{\widetilde{A}}$.

(iii) Buhrman, Vereshchagin, and de Wolf [7] showed that $\mathsf{P}_{\mathsf{cc}}^{\mathsf{NP}} \not\subset \mathsf{PP}_{\mathsf{cc}}$, which implies $\mathsf{P}^{\mathsf{NP}^{A}} \not\subset \mathsf{PP}^{\widetilde{A}}$.

(iv) Razborov [30] showed that Disjointness has quantum communication complexity $\Omega(\sqrt{N})$. This implies that $\mathsf{NP}_{\mathsf{cc}} \not\subset \mathsf{BQP}_{\mathsf{cc}}$, and hence $\mathsf{NP}^{A} \not\subset \mathsf{BQP}^{\widetilde{A}}$.[12]

(v) Raz [26] gave an exponential separation between randomized and quantum communication complexities for a promise problem. This implies that $\mathsf{PromiseBQP}_{\mathsf{cc}} \not\subset \mathsf{PromiseBPP}_{\mathsf{cc}}$, and hence $\mathsf{BQP}^{A} \not\subset \mathsf{BPP}^{\widetilde{A}}$ (note that we can remove the promise by simply choosing oracles $A, \widetilde{A}$ that satisfy it).

(vi) Raz and Shpilka [27] showed that $\mathsf{PromiseQMA}_{\mathsf{cc}} \not\subset \mathsf{PromiseMA}_{\mathsf{cc}}$. As in (iv), this implies that $\mathsf{QMA}^{A} \not\subset \mathsf{MA}^{\widetilde{A}}$.

■

6 The Integers Case

For simplicity, thus far in the paper we restricted ourselves to low-degree extensions over *fields* (typically, finite fields). We now consider the case of low-degree extensions over the integers. When we do this, one complication is that we can no longer use Gaussian elimination to construct "adversary polynomials" with desired properties. A second complication is that we now need to worry about the *size* of an extension oracle's inputs and outputs (i.e., the number of bits needed to specify them). For both of these reasons, proving algebraic oracle separations is sometimes much harder in the integers case than in the finite field case.

Formally, given a vector of integers $v = (v_1, \ldots, v_n)$, we define the *size* of v,

$$\mathrm{size}(v) := \sum_{i=1}^{n} \lceil \log_2 (|v_i| + 2) \rceil,$$

to be a rough measure of the number of bits needed to specify v. Notice that $\mathrm{size}(v) \geq n$ for all v.

We can now give the counterpart of Definition 2.2 for integer extensions:

Definition 6.1 (Extension Oracle Over The Integers) *Let $A_m : \{0,1\}^m \rightarrow \{0,1\}$ be a Boolean function. Then an extension of A_m over the integers $\mathbb{Z}$ is a polynomial $\widehat{A}_m : \mathbb{Z}^m \rightarrow \mathbb{Z}$ such that $\widehat{A}_m(x) = A_m(x)$ whenever $x \in \{0,1\}^m$. Also, given an oracle $A = (A_m)$, an extension $\widehat{A}$ of A is a collection of polynomials $\widehat{A}_m : \mathbb{Z}^m \rightarrow \mathbb{Z}$, one for each $m \in \mathbb{N}$, such that*

(i) $\widehat{A}_m$ is an extension of A_m for all m,

(ii) there exists a constant c such that $\mathrm{mdeg}(\widehat{A}_m) \leq c$ for all m, and

(iii) there exists a polynomial p such that $\mathrm{size}(\widehat{A}_m(x)) \leq p(m + \mathrm{size}(x))$ for all $x \in \mathbb{Z}^m$.

[12] Let us remark that, to our knowledge, this reduction constitutes the first use of quantum communication complexity to obtain a new lower bound on quantum query complexity. The general technique might be applicable to other problems in quantum lower bounds.

Then given a complexity class $\mathcal{C}$, by $\mathcal{C}^{\widehat{A}}$ or $\mathcal{C}^{\widehat{A}[\text{poly}]}$ we mean the class of languages decidable by a $\mathcal{C}$ machine that, on inputs of length n, can query $\widehat{A}_m$ for any m or any $m = O(\text{poly}(n))$ respectively.

Notice that integer extensions can always be used to *simulate* finite field extensions—since given an integer $\widehat{A}_m(x)$, together with a field $\mathbb{F}$ of order q^k where q is prime, an algorithm can just compute $\widetilde{A}_{m,\mathbb{F}}(x) := \widehat{A}_m(x) \bmod q$ for itself. In other words, for every integer extension $\widehat{A}$, there exists a finite field extension $\widetilde{A}$ such that $\mathcal{D}^{\widetilde{A}} \subseteq \mathcal{D}^{\widehat{A}}$ for all complexity classes $\mathcal{D}$ capable of modular arithmetic. Hence any result of the form $\mathcal{C}^A \subseteq \mathcal{D}^{\widetilde{A}}$ for all $A, \widetilde{A}$ automatically implies $\mathcal{C}^A \subseteq \mathcal{D}^{\widehat{A}}$ for all $A, \widehat{A}$. Likewise, any construction of oracles $A, \widehat{A}$ such that $\mathcal{C}^A \not\subset \mathcal{D}^{\widehat{A}}$ automatically implies the existence of $A, \widetilde{A}$ such that $\mathcal{C}^A \not\subset \mathcal{D}^{\widetilde{A}}$.

We now define the model of algebraic query complexity over the integers.

Definition 6.2 (Algebraic Query Complexity Over $\mathbb{Z}$) *Let $f : \{0,1\}^N \to \{0,1\}$ be a Boolean function, and let s and c be positive integers. Also, let $\mathcal{M}$ be the set of deterministic algorithms M such that for every oracle $A : \{0,1\}^n \to \{0,1\}$, and every integer extension $\widehat{A} : \mathbb{Z}^n \to \mathbb{Z}$ of A with $\text{mdeg}(\widehat{A}) \le c$,*

(i) $M^{\widehat{A}}$ outputs $f(A)$, and

(ii) every query x made by $M^{\widehat{A}}$ satisfies $\text{size}(x) \le s$.

Then the deterministic algebraic query complexity of f over $\mathbb{Z}$ is defined as

$$\widehat{\mathrm{D}}_{s,c}(f) := \min_{M \in \mathcal{M}} \max_{A, \widehat{A} : \text{mdeg}(\widehat{A}) \le c} T_M(\widehat{A}),$$

where $T_M(\widehat{A})$ is the number of queries to $\widehat{A}$ made by $M^{\widehat{A}}$. (For the purposes of this definition, we do not impose any upper bound on $\text{size}(\widehat{A}(x))$.) The randomized and quantum algebraic query complexities $\widehat{\mathrm{R}}_{s,c}(f)$ and $\widehat{\mathrm{Q}}_{s,c}(f)$ are defined similarly, except with (bounded-error) randomized and quantum algorithms in place of deterministic ones.

Notice that proving lower bounds on $\widehat{\mathrm{D}}_{s,c}$, $\widehat{\mathrm{R}}_{s,c}$, and $\widehat{\mathrm{Q}}_{s,c}$ becomes harder as s increases, and easier as c increases.

Our goal is twofold: (1) to prove lower bounds on the above-defined query complexity measures, and (2) to use those lower bounds to prove algebraic oracle separations over the integers (for example, that there exist $A, \widehat{A}$ such that $\mathsf{NP}^A \not\subset \mathsf{P}^{\widehat{A}}$).

6.1 Lower Bounds by Communication Complexity

A first happy observation is that *every lower bound or oracle separation proved using Theorem 4.11 (the communication complexity transfer principle) automatically carries over to the integers case.* This is so because of the following direct analogue of Theorem 4.11 for integer extensions:

Theorem 6.3 *Let $A : \{0,1\}^n \to \{0,1\}$ be a Boolean function, and let $\widehat{A} : \mathbb{Z}^n \to \mathbb{Z}$ be the unique multilinear extension of A over $\mathbb{Z}$. Suppose one can evaluate some Boolean predicate f of A using T deterministic adaptive queries to $\widehat{A}$, where each query $x \in \mathbb{Z}^n$ satisfies $\text{size}(x) \le s$. Also, let A_0 and A_1 be the subfunctions of A obtained by restricting the first bit to 0 or 1 respectively. Then if Alice is given the truth table of A_0 and Bob is given the truth table of A_1, they can jointly evaluate $f(A)$ using $O(Ts)$ bits of communication.*

The proof of Theorem 6.3 is essentially the same as the proof of Theorem 4.11, and is therefore omitted.

By analogy to Theorem 4.12, Theorem 6.3 has the following immediate consequence for the randomized query complexity of Disjointness over the integers:

Theorem 6.4 $\widehat{\mathrm{R}}_{s,1}(\mathrm{DISJ}) = \Omega(2^n/s)$ *for all* s.

Proof. Suppose $\widehat{\mathrm{R}}_{s,1}(\mathrm{DISJ}) = o(2^n/s)$. Then by Theorem 6.3, we get a randomized protocol for Disjointness with communication cost $o(2^n)$, thereby violating the lower bound of Razborov [29] and Kalyasundaram and Schnitger [18]. ■

One can also use Theorem 6.3 to construct oracles A and integer extensions $\widehat{A}$ such that

- $\mathsf{NP}^A \not\subset \mathsf{P}^{\widehat{A}}$,
- $\mathsf{RP}^A \not\subset \mathsf{P}^{\widehat{A}}$,
- $\mathsf{NP}^A \not\subset \mathsf{BQP}^{\widehat{A}}$,

and so on for all the other oracle separations obtained in Section 5.4 in the finite field case. The proofs are similar to those in Section 5.4 and are therefore omitted.

6.2 Lower Bounds by Direct Construction

Unlike with the communication complexity arguments, when we try to port the direct construction arguments of Section 4.2 to the integers case we encounter serious new difficulties. The basic source of the difficulties is that the integers are not a field but a ring, and thus we can no longer construct multilinear polynomials by simply solving linear equations.

In this section, we partly overcome this problem by using some tools from elementary number theory, such as Chinese remaindering and Hensel lifting. The end result will be an exponential lower bound on $\widehat{\mathrm{D}}_{s,2}(\mathrm{OR})$: the number of queries to a multiquadratic integer extension $\widehat{A} : \mathbb{Z}^n \to \mathbb{Z}$ needed to decide whether there exists an $x \in \{0,1\}^n$ with $A(x) = 1$, assuming the queries are deterministic and have size at most $s \ll 2^n$.

Unfortunately, even after we achieve this result, we will still not be able to use it to prove oracle separations like $\mathsf{NP}^A \not\subset \mathsf{P}^{\widehat{A}}$. The reason is technical, and has to do with size($\widehat{A}(x)$): the number of bits needed to specify an output of $\widehat{A}$. In our adversary construction, size($\widehat{A}(x)$) will grow like $O(\mathrm{size}(x) + ts)$, where t is the number of queries made by the algorithm we are fighting against and s is the maximum size of those queries. The dependence on size(x) is fine, but the dependence on t and s is a problem for two reasons. First, the number of bits needed to store $\widehat{A}$'s output might exceed the running time of the algorithm that calls $\widehat{A}$! Second, we ultimately want to diagonalize against *all* polynomial-time Turing machines, and this will imply that size($\widehat{A}(x)$) must grow faster than polynomial.

Nevertheless, both because we hope it will lead to stronger results, and because the proof is mathematically interesting, we now present a lower bound on $\widehat{\mathrm{D}}_{s,2}(\mathrm{OR})$.

Our goal is to arrive at a lemma similar to Lemma 4.3 in the field case; its analogue will be Lemma 6.9 below.

Lemma 6.5 *Let* $y_1, \ldots, y_t$ *be points in* $\mathbb{Z}^n$ *and let* q *be a prime. Then there exists a multilinear polynomial* $h_q : \mathbb{Z}^n \to \mathbb{Z}$ *such that* *(i)* $h_q(y_i) \equiv 0 \pmod{q}$ *for all* $i \in [t]$*, and*

(ii) $h_q(z) = 1$ *for at least* $2^n - t$ *Boolean points* z*.*

(Note that h_q *could be non-Boolean on the remaining Boolean points.)*

Proof. Let $N = 2^n$; then we can label the N Boolean points $z_1, \ldots, z_N$. For all $i \in [N]$, let δ_i be the unique multilinear polynomial satisfying $\delta_i(z_i) = 1$ and $\delta_i(z_j) = 0$ for all $j \neq i$.

Now let Λ be a $(t+N) \times N$ integer matrix whose top t rows are labeled by $y_1, \ldots, y_t$, whose bottom N rows are labeled by $z_1, \ldots, z_N$, and whose columns are labeled by $\delta_1, \ldots, \delta_N$. The (x, δ_i) entry is equal to $\delta_i(x)$. We assume without loss of generality that the top $t \times N$ submatrix of Λ has full rank mod q, for if it does not, then we simply remove rows until it does. Notice that the bottom $N \times N$ submatrix of Λ is just the identity matrix I.

Now remove t of the bottom N rows, in such a way that the resulting $N \times N$ submatrix B of Λ is nonsingular mod q. Then for every vector $v \in \mathbb{F}_q^N$, the system $B\alpha \equiv v \pmod{q}$ is solvable for $\alpha \in \mathbb{F}_q^N$. So choose v to contain 0's in the first t coordinates and 1's in the remaining $N - t$ coordinates; then solve to obtain a vector $\alpha = (\alpha_1, \ldots, \alpha_N)$. Finally, reinterpret the α_i's as integers from 0 to $q-1$ rather than elements of $\mathbb{F}_q$, and set the polynomial h_q to be

$$h_q(x) := \sum_{i=1}^{N} \alpha_i \delta_i(x).$$

It is clear that h_q so defined satisfies property (i). To see that it satisfies (ii), notice that the last $N - t$ rows of B are unit vectors. Hence, even over $\mathbb{F}_q$, any solution to the system $B\alpha \equiv v \pmod{q}$ must set $\alpha_{t+1} = \cdots = \alpha_N = 1$. ■

We wish to generalize Lemma 6.5 to the case where the modulus q is not necessarily prime. To do so, we will need two standard number theory facts, which we prove for completeness.

Proposition 6.6 (Hensel Lifting) *Let* B *be an* $N \times N$ *integer matrix, and suppose* B *is invertible mod* q *for some prime* q*. Then the system* $B\alpha \equiv v \pmod{q^e}$ *has a solution in* $\alpha \in \mathbb{Z}^N$ *for every* $v \in \mathbb{Z}^N$ *and* $e \in \mathbb{N}$*.*

Proof. By induction on e. When $e = 1$ the proposition obviously holds, so assume it holds for e. Then there exists a solution α to $B\alpha \equiv v \pmod{q^e}$, meaning that $B\alpha - v = q^e c$ for some $c \in \mathbb{Z}^N$. From this we want to construct a solution α' to $B\alpha' \equiv v \left(\operatorname{mod} q^{e+1}\right)$. Our solution will have the form $\alpha' = \alpha + q^e \beta$ for some $\beta \in \mathbb{Z}^N$. To find β, notice that

$$\begin{aligned} B\alpha' &= B(\alpha + q^e \beta) \\ &= B\alpha + q^e B\beta \\ &= v + q^e c + q^e B\beta \\ &= v + q^e (c + B\beta). \end{aligned}$$

Thus, it suffices to find a β such that $B\beta \equiv -c \pmod{q}$. Since B is invertible mod q, such a β exists. ■

Proposition 6.7 (Chinese Remaindering) *Let* K *and* L *be relatively prime. Then there exist integers* $a, b \in [KL]$ *such that:*

(i) For all x, y, z, the congruence $z \equiv ax + by \,(\mathrm{mod}\, KL)$ holds if and only if $z \equiv x \,(\mathrm{mod}\, K)$ and $z \equiv y \,(\mathrm{mod}\, L)$.

(ii) If $x = y = 1$, then $ax + by = KL + 1$ as an integer.

Proof. Let K' and L' be integers in $[KL]$ such that $K' \equiv K^{-1} \,(\mathrm{mod}\, L)$ and $L' \equiv L^{-1} \,(\mathrm{mod}\, K)$; note that these exist since K and L are relatively prime. Then we simply need to set $a := LL'$ and $b := KK'$. ■

We can now prove the promised generalization of Lemma 6.5.

Lemma 6.8 *Let $y_1, \ldots, y_t$ be points in $\mathbb{Z}^n$, let Q be an integer, and let $Q = q_1^{e_1} \cdots q_m^{e_m}$ be its prime factorization. Then there exists a multilinear polynomial $h_Q : \mathbb{Z}^n \to \mathbb{Z}$ such that*

(i) $h_Q(y_i) \equiv 0 \,(\mathrm{mod}\, Q)$ for all $i \in [t]$, and

(ii) $h_Q(z) = 1$ for at least $2^n - mt$ Boolean points z.

Proof. Say that a multilinear polynomial $h : \mathbb{Z}^n \to \mathbb{Z}$ is (K, r)-*satisfactory* if

(i') $h(y_i) \equiv 0 \,(\mathrm{mod}\, K)$ for all $i \in [t]$, and

(ii') $h(z) = 1$ for at least $2^n - r$ Boolean points z.

Recall that if q is prime, then Lemma 6.5 yields a (q, t)-satisfactory polynomial h_q. Furthermore, the coefficients $(\alpha_1, \ldots, \alpha_N)$ of h_q were obtained by solving a linear system $B\alpha \equiv b \,(\mathrm{mod}\, q)$ where B was invertible mod q.

First, suppose $K = q^e$ is a prime power. Then by Proposition 6.6, we can "lift" the solution $\alpha \in \mathbb{Z}^n$ of $B\alpha \equiv b \,(\mathrm{mod}\, q)$ to a solution $\alpha' \in \mathbb{Z}^n$ of $B\alpha' \equiv v \,(\mathrm{mod}\, K)$. Furthermore, after we perform this lifting, we still have $\alpha'_{t+1} = \cdots = \alpha'_N = 1$, since the matrix B has not changed (and in particular contains the identity submatrix). So if we set

$$h_K(x) := \sum_{i=1}^{N} \alpha'_i \delta_i(x)$$

then h_K is (K, t)-satisfactory.

Now let K and L be relatively prime, and suppose we found a (K, r)-satisfactory polynomial h_K as well as an (L, r')-satisfactory polynomial h_L. We want to combine h_K and h_L into a $(KL, r + r')$-satisfactory polynomial h_{KL}. To do so, we use Chinese remaindering (as in Proposition 6.7) to find an affine linear combination

$$h_{KL}(x) := ah_K(x) + bh_L(x) - KL$$

such that

(i") $h_{KL}(x) \equiv 0 \,(\mathrm{mod}\, KL)$ if and only if $h_K(x) \equiv 0 \,(\mathrm{mod}\, K)$ and $h_L(x) \equiv 0 \,(\mathrm{mod}\, L)$, and

(ii") if $h_K(x) = 1$ and $h_L(x) = 1$ then $h_{KL}(x) = 1$.

Since there are at least $2^n - (r + r')$ Boolean points z such that $h_K(z) = h_L(z) = 1$, this yields a $(KL, r + r')$-satisfactory polynomial as desired.

Thus, given any composite integer $Q = q_1^{e_1} \cdots q_m^{e_m}$, we can first use Hensel lifting to find a $(q_i^{e_i}, t)$-satisfactory polynomial h_{q_i} for every $i \in [m]$, and then use Chinese remaindering to combine the h_{q_i}'s into a (Q, mt)-satisfactory polynomial h_Q. ■

We are finally ready to prove the integer analogue of Lemma 4.3.

Lemma 6.9 *Let $y_1, \ldots, y_t$ be points in $\mathbb{Z}^n$, such that* $\operatorname{size}(y_i) \leq s$ *for all $i \in [t]$. Then for at least $2^n - 2t^2 s$ Boolean points $w \in \{0,1\}^n$, there exists a multiquadratic polynomial $p : \mathbb{Z}^n \to \mathbb{Z}$ such that*

(i) $p(y_i) = 0$ for all $i \in [t]$,

(ii) $p(w) = 1$, and

(iii) $p(z) = 0$ for all Boolean $z \neq w$.

Proof. Assume $t \leq 2^n$, since otherwise the lemma is trivial.

Let $h_Q : \mathbb{Z}^n \to \mathbb{Z}$ be the multilinear polynomial from Lemma 6.8, for some integer Q to be specified later. Then our first claim is that there exists a multilinear polynomial $g : \mathbb{Q}^n \to \mathbb{Q}$, with rational coefficients, such that

(i') $g(y_i) = h_Q(y_i)$ for all $i \in [t]$, and

(ii') $g(z) = 0$ for at least $2^n - t$ Boolean points z.

This claim follows from linear algebra: we know the requirements $g(y_i) = h_Q(y_i)$ for $i \in [t]$ are mutually consistent, since there exists a multilinear polynomial, namely h_Q, that satisfies them. So if we write g in the basis of δ_z's, as follows:

$$g(x) = \sum_{z \in \{0,1\}^n} g(z)\, \delta_z(x)$$

then condition (i') gives us t' independent affine constraints on the 2^n coefficients $g(z)$, for some $t' \leq t$. This means there must exist a solution g such that $g(z) = 0$ for at least $2^n - t'$ Boolean points z. Let $z_1, \ldots, z_{t'}$ be the remaining t' Boolean points.

Notice that $z_1, \ldots, z_{t'}$ can be chosen independently of h_Q. This is because we simply need to find t' Boolean points $z_1, \ldots, z_{t'}$, such that any "allowed" vector $(h_Q(y_1), \ldots, h_Q(y_t))$ can be written as a rational linear combination of vectors of the form $\left(\delta_{z_j}(y_1), \ldots, \delta_{z_j}(y_t)\right)$ with $j \in [t']$.

We now explain how Q is chosen. Let Γ be a $t \times t'$ matrix whose rows are labeled by $y_1, \ldots, y_t$, whose columns are labeled by $z_1, \ldots, z_{t'}$, and whose (i, j) entry equals $\delta_{z_j}(y_i)$. Then since we had t' independent affine constraints, there must be a $t' \times t'$ submatrix Γ' of Γ with full rank. We set $Q := |\det(\Gamma')|$.

With this choice of Q, we claim that g is actually an *integer* polynomial. It suffices to show that $g(z_j)$ is an integer for all $j \in [t']$, since the value of g at any $x \in \mathbb{Z}^n$ can be written as an integer linear combination of its values on the Boolean points. Note that the vector $(g(z_1), \ldots, g(z_{t'}))$ is obtained by applying the matrix $(\Gamma')^{-1}$ to some vector $(v_1, \ldots, v_{t'})$ whose entries are $h_Q(y_i)$'s. Now, every entry of $(\Gamma')^{-1}$ has the form k/Q, where k is an integer; and since $h_Q(y_i) \equiv 0 \pmod{Q}$ for all $i \in [t]$, every v_i is an integer multiple of Q. This completes the claim.

Also, since $\operatorname{size}(y_i) \leq s$ for all $i \in [t]$, we have the upper bound

$$
\begin{aligned}
Q &= \left|\det\left(\Gamma'\right)\right| \\
&\leq t'! \left(\prod_{i=1}^{n} \left(|y_i| + 1\right)\right)^{t'} \\
&\leq t^t \left(\prod_{i=1}^{n} \left(|y_i| + 1\right)\right)^{t} \\
&\leq t^t 2^{ts} \\
&= 2^{ts + t\log_2 t} \\
&\leq 2^{2ts}.
\end{aligned}
$$

Here the last line uses the assumption that $\log_2 t \leq n$, together with the fact that $n \leq s$.

Therefore Q can have at most $2ts$ distinct prime factors. So by Lemma 6.8, we have $h_Q(z) = 1$ for at least $2^n - 2t^2 s$ Boolean points z.

Putting everything together, if we define $m(x) := h_Q(x) - g(x)$, then we get a multilinear polynomial $m : \mathbb{Z}^n \to \mathbb{Z}$ such that

(i") $m(y_i) = 0$ for all $i \in [t]$, and

(ii") $m(z) = 1$ for at least $2^n - 2t^2 s$ Boolean points z.

Then for any $w \in \{0,1\}^n$ with $m(w) = 1$, we can get a multiquadratic polynomial $p : \mathbb{Z}^n \to \mathbb{Z}$ satisfying conditions (i)-(iii) of the lemma by taking $p(x) := m(x)\,\delta_w(x)$. ■

Lemma 6.9 easily implies a lower bound on the deterministic query complexity of the OR function.

Theorem 6.10 $\widehat{\mathrm{D}}_{s,2}(\mathrm{OR}) = \Omega\left(\sqrt{2^n/s}\right)$ *for all* s.

Proof. Let $\mathcal{Y}$ be the set of points queried by a deterministic algorithm, and assume $\operatorname{size}(y) \leq s$ for all $y \in \mathcal{Y}$. Then provided $2^n - 2|\mathcal{Y}|^2 s > 0$ (or equivalently $|\mathcal{Y}| < \sqrt{2^{n-1}/s}$), Lemma 6.9 implies that there exists a multiquadratic extension polynomial $\widehat{A} : \mathbb{Z}^n \to \mathbb{Z}$ such that $\widehat{A}(y) = 0$ for all $y \in \mathcal{Y}$, but nevertheless $\widehat{A}(w) = 1$ for some Boolean point w. So even if the algorithm is adaptive, we can let $\mathcal{Y}$ be the set of points it queries assuming each query is answered with 0, and then find $\widehat{A}, \widehat{B}$ such that $\widehat{A}(y) = \widehat{B}(y) = 0$ for all $y \in \mathcal{Y}$, but nevertheless $\widehat{A}$ and $\widehat{B}$ lead to different values of the OR function. ■

As mentioned before, one can calculate that the polynomial p from Lemma 6.9 satisfies $\operatorname{size}(p(x)) = O(\operatorname{size}(x) + ts)$. For algebrization purposes, the key question is whether the dependence on t and s can be eliminated, and replaced by some fixed polynomial dependence on $\operatorname{size}(x)$ and n. Another interesting question is whether one can generalize Lemma 6.9 to queries of unbounded size—that is, whether the assumption $\operatorname{size}(y_i) \leq s$ can simply be eliminated.

7 Applications to Communication Complexity

In this section, we give two applications of our algebrization framework to communication complexity:

(1) A new connection between communication complexity and computational complexity, which implies that certain plausible communication complexity conjectures would imply $\mathsf{NL} \neq \mathsf{NP}$.

(2) MA-protocols for the Disjointness and Inner Product problems with total communication cost $O\left(\sqrt{n}\log n\right)$, essentially matching a lower bound of Klauck [21].

Both of these results can be stated without any reference to algebrization. On the other hand, they arose directly from the "transfer principle" relating algebrization to communication complexity in Section 4.3.

7.1 Karchmer-Wigderson Revisited

Two decades ago, Karchmer and Wigderson [20, 38] noticed that certain communication complexity lower bounds imply circuit lower bounds—or in other words, that one can try to separate complexity classes by thinking only about communication complexity. In this section we use algebrization to give further results in the same spirit.

Let $f : \{0,1\}^N \times \{0,1\}^N \to \{0,1\}$ be a Boolean function, and let x and y be inputs to f held by Alice and Bob respectively. By an IP*-protocol* for f, we mean a randomized communication protocol where Alice and Bob exchange messages with each other, as well as with an omniscient prover Merlin who knows x and y. The communication cost is defined as the total number of bits exchanged among Alice, Bob, and Merlin. If $f(x,y) = 1$, then there should exist a strategy of Merlin that causes Alice and Bob to accept with probability at least $2/3$, while if $f(x,y) = 0$ no strategy should cause them to accept with probability more than $1/3$.

Lemma 7.1 *Suppose* $f : \{0,1\}^N \times \{0,1\}^N \to \{0,1\}$ *is in* NL. *Then* f *has an* IP*-protocol with communication cost* $O(\operatorname{polylog} N)$.

Proof. Let $N = 2^n$. Then we can define a Boolean function $A : \{0,1\}^{n+1} \to \{0,1\}$, such that the truth table of $A(0x)$ corresponds to Alice's input, while the truth table of $A(1x)$ corresponds to Bob's input. Taking n as the input length, we then have $f \in \mathsf{PSPACE}^{A[\mathrm{poly}]}$. By Theorem 3.7 we have $\mathsf{PSPACE}^{A[\mathrm{poly}]} \subseteq \mathsf{IP}^{\widetilde{A}}$, where $\widetilde{A}$ is the multilinear extension of A. Hence $f \in \mathsf{IP}^{\widetilde{A}}$. But by Theorem 4.11, this means that f admits an IP-protocol with communication cost $O(\operatorname{poly} n) = O(\operatorname{polylog} N)$. ■

An immediate consequence of Lemma 7.1 is that, to prove a problem is outside NL, it suffices to lower-bound its IP communication complexity:

Theorem 7.2 *Let Alice and Bob hold* $3SAT$ *instances* φ_A, φ_B *respectively of size* N. *Suppose there is no* IP*-protocol with communication cost* $O(\operatorname{polylog} N)$*, by which Merlin can convince Alice and Bob that* φ_A *and* φ_B *have a common satisfying assignment. Then* $\mathsf{NL} \neq \mathsf{NP}$.

Likewise, to prove a problem is outside P, it suffices to lower-bound its RG communication complexity, where RG is the Refereed Games model of Feige and Kilian [9] (with a competing yes-prover and no-prover). In this case, though, the $\mathsf{EXP} = \mathsf{RG}$ theorem is not only algebrizing but also relativizing, and this lets us prove a stronger result:

Theorem 7.3 *Let* φ *be a* $3SAT$ *instance of size* N. *Suppose there is no bounded-error randomized verifier that decides whether* φ *is satisfiable by*

(i) making $O(\operatorname{polylog} N)$ *queries to a binary encoding of* φ*, and*

(ii) exchanging $O(\text{polylog } N)$ bits with a competing yes-prover and no-prover, both of whom know φ and can exchange private messages not seen by the other prover.

Then $\mathsf{P} \neq \mathsf{NP}$.

Proof. Suppose $\mathsf{P} = \mathsf{NP}$. Then by padding, $\mathsf{EXP}^{A[\text{poly}]} = \mathsf{NEXP}^{A[\text{poly}]}$ for all oracles A. As discussed in Section 3.5, the work of Feige and Kilian [9] implies that $\mathsf{EXP}^{A[\text{poly}]} = \mathsf{RG}^{A}$ for all oracles A. Hence $\mathsf{NEXP}^{A[\text{poly}]} = \mathsf{RG}^{A}$ as well. In other words, given oracle access to an exponentially large $3SAT$ instance φ, one can decide in RG whether φ is satisfiable. Scaling down by an exponential now yields the desired result. ■

7.2 Disjointness and Inner Product

In this subsection we consider two communication problems. The first is Disjointness, which was defined in Section 4.3. The second is Inner Product, which we define as follows. Alice and Bob are given n-bit strings $x_1 \ldots x_n$ and $y_1 \ldots y_n$ respectively; then their goal is to compute

$$IP(x,y) := \sum_{i=1}^{n} x_i y_i$$

as an integer. Clearly Disjointness is equivalent to deciding whether $IP(x,y) = 0$, and hence is reducible to Inner Product.

Klauck [21] showed that any MA-protocol for Disjointness has communication cost $\Omega(\sqrt{n})$. The "natural" conjecture would be that the $\sqrt{n}$ was merely an artifact of his proof, and that a more refined argument would yield the optimal lower bound of $\Omega(n)$. However, using a protocol inspired by our algebrization framework, we are able to show that this conjecture is false.

Theorem 7.4 *There exist MA-protocols for the Disjointness and Inner Product problems, in which Alice receives an $O(\sqrt{n}\log n)$-bit witness from Merlin and an $O(\sqrt{n}\log n)$-bit message from Bob.*

Proof. As observed before, it suffices to give a protocol for Inner Product; a protocol for Disjointness then follows immediately.

Assume n is a perfect square. Then Alice and Bob can be thought of as holding functions $a : [\sqrt{n}] \times [\sqrt{n}] \to \{0,1\}$ and $b : [\sqrt{n}] \times [\sqrt{n}] \to \{0,1\}$ respectively. Their goal is to compute the inner product

$$IP := \sum_{x,y \in [\sqrt{n}]} a(x,y)\, b(x,y).$$

Choose a prime $q \in [n, 2n]$. Then a and b have unique extensions $\widetilde{a} : \mathbb{F}_q^2 \to \mathbb{F}_q$ and $\widetilde{b} : \mathbb{F}_q^2 \to \mathbb{F}_q$ respectively as degree-$(\sqrt{n}-1)$ polynomials. Also, define the polynomial $s : \mathbb{F}_q \to \mathbb{F}_q$ by

$$s(x) := \sum_{y=1}^{\sqrt{n}} \widetilde{a}(x,y)\, \widetilde{b}(x,y) \pmod{q}.$$

Notice that $\deg(s) \leq 2(\sqrt{n}-1)$.

Merlin's message to Alice consists of a polynomial $s' : \mathbb{F}_q \to \mathbb{F}_q$, which also has degree at most $2(\sqrt{n}-1)$, and which is specified by its coefficients. Merlin claims that $s = s'$. If Merlin is honest, then Alice can easily compute the inner product as

$$IP = \sum_{x=1}^{\sqrt{n}} s(x).$$

So the problem reduces to checking that $s = s'$. This is done as follows: first Bob chooses $r \in \mathbb{F}_q$ uniformly at random and sends it to Alice, along with the value of $\widetilde{b}(r, y)$ for every $y \in [\sqrt{n}]$. Then Alice checks that

$$s'(r) = \sum_{y=1}^{\sqrt{n}} \widetilde{a}(r, y)\, \widetilde{b}(r, y) \pmod{q}.$$

If $s = s'$, then the above test succeeds with certainty. On the other hand, if $s \neq s'$, then

$$\Pr_{r \in \mathbb{F}_q}\left[s(r) = s'(r)\right] \leq \frac{\deg(s)}{q} \leq \frac{1}{3},$$

and hence the test fails with probability at least $\frac{2}{3}$. ■

Let us make two remarks about Theorem 7.4.

First, we leave as an open problem whether one could do even better than $\widetilde{O}(\sqrt{n})$ by using an AM-protocol: that is, a protocol in which Alice (say) can send a single random challenge to Merlin and receive a response. (As before, the communication cost is defined as the sum of the lengths of *all* messages between Alice, Bob, and Merlin.) On the other hand, it is easy to generalize Theorem 7.4 to give an MAM-protocol (one where first Merlin sends a message, then Alice, then Merlin) with complexity $O\left(n^{1/3} \log n\right)$. Similarly, one can give an MAMAM-protocol with complexity $O\left(n^{1/4} \log n\right)$, an MAMAMAM-protocol with complexity $O\left(n^{1/5} \log n\right)$, and so on. In the limit of arbitrarily many rounds, one gets an IP-protocol with complexity $O(\log n \log \log n)$.

Second, one might wonder how general Theorem 7.4 is. In particular, can it be extended to give an MA-protocol for *every* predicate $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ with total communication $\widetilde{O}(\sqrt{n})$? The answer is no, by a simple counting argument.

We can assume without loss of generality that every MA-protocol has the following form: first Alice and Bob receive an m-bit message from Merlin; then they exchange T messages between themselves consisting of a single bit each. Let p_t be the probability that the t^{th} message is a '1', as a function of the $n + m + t - 1$ bits (one player's input plus Merlin's message plus $t - 1$ previous messages) that are relevant at the t^{th} step. It is not hard to see that each p_t can be assumed to have the form i/n^2, where i is an integer, with only negligible change to the acceptance probability. In that case there are $\left(n^2\right)^{2^{n+m+t-1}}$ choices for each function $p_t : \{0,1\}^{n+m+t-1} \to [0,1]$, whence

$$\left(n^2\right)^{2^{n+m}} \left(n^2\right)^{2^{n+m+1}} \cdots \left(n^2\right)^{2^{n+m+T-1}}$$

possible protocols. But if $m + T = o(n)$, this product is still dwarfed by $2^{2^{2n}}$, the number of distinct Boolean functions $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$.

Thus, Theorem 7.4 has the amusing consequence that the Inner Product function, which is often considered the "hardest" function in communication complexity, is actually unusually *easy* for MA-protocols. (The special property of Inner Product we used is that it can be written as a degree-2 polynomial in Alice's and Bob's inputs.)

8 Zero-Knowledge Protocols

In searching complexity theory for potentially non-algebrizing results, it seems the main source is cryptography—and more specifically, cryptographic results that exploit the locality of computation. These include the zero-knowledge protocol for NP due to Goldreich, Micali, and Wigderson [12] (henceforth the GMW Theorem), the two-party oblivious circuit evaluation of Yao [40], and potentially many others. Here we focus on the GMW Theorem.

As discussed in Section 1, the GMW Theorem is inherently non-black-box, since it uses the structure of an NP-complete problem (namely 3-Coloring). On the other hand, the way the theorem exploits that structure seems inherently non-algebraic: it does not involve finite fields or low-degree polynomials. Nevertheless, in this section we will show that even the GMW Theorem is algebrizing.

Let us start by defining the class CZK, or Computational Zero Knowledge, as well as CZK^A (that is, CZK with oracle access to A).

Definition 8.1 *A language L is in CZK if there exists a protocol in which a probabilistic polynomial-time verifier V interacts with a computationally-unbounded prover P, such that for all inputs x the following holds.*

- ***Completeness.*** *If $x \in L$ then P causes V to accept with probability 1.*
- ***Soundness.*** *If $x \notin L$ then no prover P^* can cause V to accept with probability more than 1/2.*
- ***Zero-Knowledge.*** *If $x \in L$ then there exists an expected polynomial-time simulator that, given black-box access[13] to a polynomial-time verifier V^*, produces a message transcript that cannot be efficiently distinguished from a transcript of an actual conversation between V^* and P. (In other words, the two probability distributions over message transcripts are computationally indistinguishable.)*

We define CZK^A to mean the version of CZK where all three machines—the prover, verifier, and simulator—have access to the oracle A.

Then Goldreich et al. [12] proved the following:

Theorem 8.2 (GMW Theorem) *If one-way functions exist then $\mathsf{NP} \subseteq \mathsf{CZK}$.*

It is not hard to show that Theorem 8.2 is non-relativizing. Intuitively, given a black-box function $f : \{0,1\}^n \to \{0,1\}$, suppose a prover P wants to convince a polynomial-time verifier V that there exists a z such that $f(z) = 1$. Then there are two possibilities: either P can cause V to query $f(z)$, in which case the protocol will necessarily violate the zero-knowledge condition (by revealing z); or else P can *not* cause V to query $f(z)$, in which case the protocol will violate either completeness or soundness. By formalizing this intuition one can show the following:

Theorem 8.3 *There exists an oracle A relative to which*

(i) one-way functions exist (i.e. there exist functions computable in P^A that cannot be inverted in BPP^A on a non-negligible fraction of inputs), but

(ii) $\mathsf{NP}^A \not\subset \mathsf{CZK}^A$.

By contrast, we now show that the GMW Theorem is algebrizing. In proving this theorem, we will exploit the availability of a low-degree extension $\widetilde{A}$ to make the oracle queries zero-knowledge.

[13] CZK can also be defined in a way that allows non-black-access (i.e., access to the verifier's source code). However, our protocol, like the original GMW one, will only require black-box access.

Theorem 8.4 *Let A be an oracle and let $\widetilde{A}$ be any extension of A. Suppose there exists a one-way function, computable in $\mathsf{P}^{\widetilde{A}}$, which cannot be inverted with non-negligible probability by $\mathsf{BPP}^{\widetilde{A}}$ adversaries. Then $\mathsf{NP}^{A} \subseteq \mathsf{CZK}^{\widetilde{A}}$.*

Proof. We will assume for simplicity that the extension $\widetilde{A}$ is just a polynomial $\widetilde{A} : \mathbb{F}^n \rightarrow \mathbb{F}$ over a fixed finite field $\mathbb{F}$, which extends a Boolean function $A : \{0,1\}^n \rightarrow \{0,1\}$. Also, let $d = \deg(\widetilde{A})$, and assume $d \ll \operatorname{char}(\mathbb{F})$. (The proof easily generalizes to the case where A and $\widetilde{A}$ are as defined in Section 2.)

Deciding whether an NP^{A} machine accepts is equivalent to deciding the satisfiability of a Boolean formula $\varphi(w_1, \ldots, w_m)$, which consists of a conjunction of two types of clauses:

(i) Standard $3SAT$ clauses over the variables $w_1, \ldots, w_m$.

(ii) "Oracle clauses," each of which has the form $y_i = A(Y_i)$, where $Y_i \in \{0,1\}^n$ is a query to A (composed of n variables $w_{j_1}, \ldots, w_{j_n}$) and y_i is its expected answer (composed of another variable $w_{j_{n+1}}$).

Given such a formula φ, our goal is to convince a $\mathsf{BPP}^{\widetilde{A}}$ verifier that φ is satisfiable, without revealing anything about the satisfying assignment $w_1, \ldots, w_m$ (or anything else). To achieve this, we will describe a constant-round zero-knowledge protocol in which the verifier accepts with probability 1 given an honest prover, and rejects with probability $\Omega(1/\operatorname{poly}(n))$ given a cheating prover. Given any such protocol, it is clear that we can increase the soundness gap to $\Omega(1)$, by repeating the protocol $\operatorname{poly}(n)$ times.

Let us describe our protocol in the case that the prover and verifier are both honest. In the first round, the prover uses the one-way function to send the verifier commitments to the following objects:

- A satisfying assignment $w_1, \ldots, w_m$ for φ.
- A random nonzero field element $r \in \mathbb{F}$.
- For each oracle clause $y_i = A(Y_i)$,
  - A random affine function $L_i : \mathbb{F} \rightarrow \mathbb{F}^n$ (in other words, a line) such that $L_i(0) = Y_i$ and $L_i(1) \neq Y_i$.
  - A polynomial $p_i : \mathbb{F} \rightarrow \mathbb{F}$, of degree at most d, such that $p_i(t) = \widetilde{A}(L_i(t))$ for all $t \in \mathbb{F}$.

Given these objects, the verifier can choose randomly to perform one of the following four tests:

(1) Ask the prover for a zero-knowledge proof that the standard $3SAT$ clauses are satisfied.

(2) Choose a random oracle clause $y_i = A(Y_i)$, and ask for a zero-knowledge proof that $L_i(0) = Y_i$.

(3) Choose a random oracle clause $y_i = A(Y_i)$, and ask for a zero-knowledge proof that $p_i(0) = y_i$.

(4) Choose a random oracle clause $y_i = A(Y_i)$ as well as a random nonzero field element $s \in \mathbb{F}$. Ask for the value u of $L_i(rs)$, as well as a zero-knowledge proof that $u = L_i(rs)$. Query $\widetilde{A}(u)$. Ask for a zero-knowledge proof that $p_i(rs) = \widetilde{A}(u)$.

To prove the correctness of the above protocol, we need to show three things: completeness, zero-knowledge, and soundness.

Completeness: This is immediate. If the prover is honest, then tests (1)-(4) will all pass with probability 1.

Zero-Knowledge: Let V^* be any verifier. We will construct a simulator to create a transcript which is computationally indistinguishable from its communication with the honest prover P. The simulator first chooses random values for the w_i's (which might not be satisfying at all) and commits to them. It also commits to a random $r \in F^*$. For tests (1)-(3), the simulator acts as in the proof of the GMW Theorem [12]. So the interesting test is (4).

First note that rs is a random nonzero element, regardless of how V^* selected s. Now the key observation is that $L_i(rs)$, the point at which the verifier queries $\widetilde{A}$, is just a uniform random point in $\mathbb{F}^n \diagdown \{Y_i\}$. Thus, we can construct a simulator as follows: if the verifier is going to ask the prover about an oracle clause $y_i = A(Y_i)$, then first choose a point $X_i \in \mathbb{F}^n$ uniformly at random and query $\widetilde{A}(X_i)$. (The probability that X_i will equal Y_i is negligible.) Next choose nonzero field elements $r, s \in \mathbb{F}$ uniformly at random. Let L_i be the unique line such that $L_i(0) = Y_i$ and $L_i(rs) = X_i$, and let p_i be the unique degree-d polynomial such that $p_i(t) = \widetilde{A}(L_i(t))$ for all $t \in \mathbb{F}$ (which can be found by interpolation). Construct commitments to all of these objects. Assuming the underlying commitment scheme is secure against $\mathsf{BPP}^{\widetilde{A}}$ machines, the resulting probability distribution over messages will be computationally indistinguishable from the actual distribution.

Soundness: Suppose the NP^A machine rejects. Then when the prover sends the verifier a commitment to the "satisfying assignment" $w_1, \ldots, w_m$, some clause C of φ will necessarily be unsatisfied. If C is one of the standard $3SAT$ clauses, then by the standard GMW Theorem, the prover will be caught with $\Omega(1/\operatorname{poly}(n))$ probability when the verifier performs test (1). So the interesting case is that C is an oracle clause $y_i = A(Y_i)$.

In this case, since the truth is that $y_i \neq A(Y_i)$, at least one of the following must hold:

(i) $y_i \neq p_i(0)$,

(ii) $p_i(0) \neq \widetilde{A}(L_i(0))$, or

(iii) $\widetilde{A}(L_i(0)) \neq A(Y_i)$.

If (i) holds, then the prover will be caught with $\Omega(1/\operatorname{poly}(n))$ probability when the verifier performs test (3).

If (ii) holds, then the two degree-d polynomials $p_i(t)$ and $\widetilde{A}(L_i(t))$ must differ on at least a $1 - d/\operatorname{char}(\mathbb{F})$ fraction of points $t \in \mathbb{F}$. Hence, since rs is a random nonzero element of $\mathbb{F}$ conditioned only on s being random, the prover will be caught with $\Omega(1/\operatorname{poly}(n))$ probability when the verifier performs test (4).

If (iii) holds, then $L_i(0) \neq Y_i$. Hence the prover will be caught with $\Omega(1/\operatorname{poly}(n))$ probability when the verifier performs test (2). ■

Notice that in our zero-knowledge protocol, the prover's strategy can actually be implemented in $\mathsf{BPP}^{\widetilde{A}}$, given a satisfying assignment $w_1, \ldots, w_m$ for the formula φ.

9 The Limits of Our Limit

Some would argue with this paper's basic message, on the grounds that we already have various non-relativizing results that are not based on arithmetization. Besides the GMW protocol (which we dealt with in Section 8), the following examples have been proposed:

(1) Small-depth circuit lower bounds, such as $\mathsf{AC}^0 \neq \mathsf{TC}^0$ [28], can be shown to fail relative to suitable oracle gates.

(2) Arora, Impagliazzo, and Vazirani [2] argue that even the Cook-Levin Theorem (and by extension, the PCP Theorem) should be considered non-relativizing.

(3) Hartmanis et al. [13] cite, as examples of non-relativizing results predating the "interactive proofs revolution," the 1977 result of Hopcroft, Paul, and Valiant [15] that $\mathsf{TIME}(f(n)) \neq \mathsf{SPACE}(f(n))$ for any space-constructible f, as well as the 1983 result of Paul et al. [25] that $\mathsf{TIME}(n) \neq \mathsf{NTIME}(n)$. Recent time-space tradeoffs for SAT (see van Melkebeek [24] for a survey) have a similar flavor.

There are two points we can make regarding these examples. Firstly, the small-depth circuit lower bounds are already "well covered" by the natural proofs barrier. Secondly, because of subtleties in defining the oracle access mechanism, there is legitimate debate about whether the results listed in (2) and (3) should "truly" be considered non-relativizing; see Fortnow [10] for a contrary perspective.[14]

Having said this, we do not wish to be dogmatic. Our results tell us a great deal about the future prospects for arithmetization, but about other non-relativizing techniques they are comparatively silent.

10 Beyond Algebrizing Techniques?

In this section, we discuss two ideas one might have for going beyond the algebrization barrier, and show that some of our limitation theorems apply even to these ideas.

10.1 k-Algebrization

One of the most basic properties of relativization is *transitivity*: if two complexity class inclusions $\mathcal{C} \subseteq \mathcal{D}$ and $\mathcal{D} \subseteq \mathcal{E}$ both relativize, then the inclusion $\mathcal{C} \subseteq \mathcal{E}$ also relativizes. Thus, it is natural to ask whether algebrization is transitive in the same sense. We do not know the answer to this question, and suspect that it is negative. However, there is a *kind* of transitivity that holds. Given an oracle A, let a *double-extension* $\widetilde{\widetilde{A}}$ of A be an oracle produced by

(1) taking a low-degree extension $\widetilde{A}$ of A,

(2) letting f be a Boolean oracle such that $f(x,i)$ is the i^{th} bit in the binary representation of $\widetilde{A}(x)$, and then

(3) taking a low-degree extension $\widetilde{\widetilde{A}}$ of f.

(One can similarly define a triple-extension $\widetilde{\widetilde{\widetilde{A}}}$, and so on.) Then the following is immediate:

Proposition 10.1 *For all complexity classes $\mathcal{C}, \mathcal{D}, \mathcal{E}$, if $\mathcal{C}^A \subseteq \mathcal{D}^{\widetilde{A}}$ and $\mathcal{D}^A \subseteq \mathcal{E}^{\widetilde{A}}$ for all $A, \widetilde{A}$, then $\mathcal{C}^A \subseteq \mathcal{E}^{\widetilde{\widetilde{A}}}$ for all $A, \widetilde{\widetilde{A}}$.*

[14] Eric Allender has suggested the delightful term "irrelativizing," for results that neither relativize nor fail to relativize.

Now, the above suggests one possible approach to defeating the algebrization barrier. Call a complexity class inclusion $\mathcal{C} \subseteq \mathcal{D}$ *double-algebrizing* if $\mathcal{C}^{A} \subseteq \mathcal{D}^{\widetilde{\widetilde{A}}}$ for all $A, \widetilde{\widetilde{A}}$, *triple-algebrizing* if $\mathcal{C}^{A} \subseteq \mathcal{D}^{\widetilde{\widetilde{\widetilde{A}}}}$ for all $A, \widetilde{\widetilde{\widetilde{A}}}$, and so on. Then any k-algebrizing result is also $(k+1)$-algebrizing, but the converse need not hold. We thus get a whole infinite hierarchy of proof techniques, of which this paper studied only the first level.

Alas, we now show that *any proof of* $\mathsf{P} \neq \mathsf{NP}$ *will need to go outside the entire hierarchy.*

Theorem 10.2 *Any proof of* $\mathsf{P} \neq \mathsf{NP}$ *will require techniques that are not merely non-algebrizing, but non-k-algebrizing for every constant k.*

Proof. Recall that in Theorem 5.1, we showed that any proof of $\mathsf{P} \neq \mathsf{NP}$ will require non-algebrizing techniques, by giving oracles $A, \widetilde{A}$ such that $\mathsf{NP}^{\widetilde{A}} = \mathsf{P}^{A} = \mathsf{PSPACE}$. In that case, A was any PSPACE-complete language, while $\widetilde{A}$ was the unique multilinear extension of A, which is also PSPACE-complete by Babai, Fortnow, and Lund [3]. Now let $\widetilde{\widetilde{A}}$ be the multilinear extension of the binary representation of $\widetilde{A}$. Then $\widetilde{\widetilde{A}}$ is also PSPACE-complete by Babai et al. Hence $\mathsf{NP}^{\widetilde{\widetilde{A}}} = \mathsf{P}^{A} = \mathsf{PSPACE}$. The same is true inductively for $\widetilde{\widetilde{\widetilde{A}}}$ and so on. ■

Similarly, any proof $\mathsf{P} \neq \mathsf{PSPACE}$ will require techniques that are non-k-algebrizing for every k.

On the other hand, for most of the other open problems mentioned in this paper—P versus RP, NEXP versus $\mathsf{P/poly}$, and so on—we do not know whether double-algebrizing techniques already suffice. That is, we do not know whether there exist $A, \widetilde{\widetilde{A}}$ such that $\mathsf{RP}^{A} \not\subset \mathsf{P}^{\widetilde{\widetilde{A}}}$, $\mathsf{NEXP}^{\widetilde{\widetilde{A}}} \subset \mathsf{P}^{A}/\mathsf{poly}$, and so on. Thus, of the many open problems that are beyond the reach of arithmetization, at least some could conceivably be solved by "k-arithmetization."

10.2 Non-Commutative Algebras

We have shown that arithmetization—"lifting" Boolean logic operations to arithmetic operations over the integers or a field—will not suffice to solve many of the open problems in complexity theory. A natural question is whether one could evade our results by lifting to other algebras, particularly non-commutative ones. Unfortunately, we now explain why our limitation theorems extend with little change to *associative algebras with identity over a field*. This is a very broad class that includes matrix algebras, quaternions, Clifford algebras, and more. The one constraint is that the dimension of the algebra (or equivalently, the representation size of the elements) should be less than exponential in n.[15]

Formally, an *algebra* over the field $\mathbb{F}$ is a vector space V over $\mathbb{F}$, which is equipped with a multiplication operation $V \cdot V \rightarrow V$ such that $u(v+w) = uv + uw$ for all $u, v, w \in V$. The algebra is *associative* if its multiplication is associative, and *has identity* if one of its elements is a multiplicative identity. The *dimension* of the algebra is the dimension of V as a vector space.

A crucial observation is that every k-dimensional associative algebra over $\mathbb{F}$ is isomorphic to a subalgebra of $M_k(\mathbb{F})$, the algebra of $k \times k$ matrices with entries in the field $\mathbb{F}$. The embedding is the natural one: every element $v \in V$ defines a linear transformation M_v via $M_v x = v \cdot x$.

We will now explain why, for associative algebras with identity, our main results go through almost without change. For notational simplicity, we will state our results in terms of the full

[15]This is similar to the requirement that the integers should not be too large in Section 6.

matrix algebra $M_k(\mathbb{F})$, though the results would work just as well for any subalgebra of $M_k(\mathbb{F})$ containing the zero and identity elements.

Given a polynomial $p : M_k(\mathbb{F})^n \to M_k(\mathbb{F})$, call p *sorted-multilinear* if it has the form

$$p(X_1, \ldots, X_n) = \sum_{S \subseteq [n]} a_S \prod_{i \in S} X_i,$$

where the coefficients a_S belong to $\mathbb{F}$, and all products are taken in order from X_1 to X_n.

Now let I_k be the $k \times k$ identity matrix and 0_k be the all-zeroes matrix. Also, call a point $Z \in M_k(\mathbb{F})^n$ Boolean if every coordinate is either I_k or 0_k, and let

$$\delta_Z(X) := \prod_{i=1}^{n} [Z_i X_i + (I_k - Z_i)(I_k - X_i)]$$

be the unique sorted-multilinear polynomial such that $\delta_Z(Z) = I_k$ and $\delta_Z(W) = 0_k$ for all Boolean $W \neq Z$.

Then just as in the commutative case, every sorted-multilinear polynomial m has a unique representation in the form

$$m(X) = \sum_{Z \in \{0_k, I_k\}^n} m_Z \delta_Z(X)$$

where m_Z is a coefficient in $\mathbb{F}$ such that $m(Z) = m_Z I_K$. Also, every Boolean function $f : \{0_k, I_k\}^n \to \{0_k, I_k\}$ has a unique extension

$$\widetilde{f}(X) = \sum_{Z \in \{0_k, I_k\}^n} f(Z) \delta_Z(X)$$

as a sorted-multilinear polynomial.

Provided $k = O(\text{poly}(n))$, it is easy to show that any proof of $\mathsf{P} \neq \mathsf{NP}$ will require "non-commutatively non-algebrizing techniques." Once again, we can let A be any PSPACE-complete language, and let $\widetilde{A}$ be the unique sorted-multilinear extension of A over $M_k(\mathbb{F})$. Then the observations of Babai, Fortnow, and Lund [3] imply that $\widetilde{A}$ is also computable in PSPACE, and hence $\mathsf{NP}^{\widetilde{A}} = \mathsf{P}^A = \mathsf{PSPACE}$.

We can also repeat the separation results of Sections 4 and 5 in the non-commutative setting. Rather than going through every result again, we will just give one illustrative example. We will show that, given a non-commutative extension $\widetilde{A} : M_k(\mathbb{F})^n \to M_k(\mathbb{F})$ of a Boolean function $A : \{0_k, I_k\}^n \to \{0_k, I_k\}$, any deterministic algorithm needs $\Omega(2^n/k^2)$ queries to $\widetilde{A}$ to find a Boolean point $W \in \{0_k, I_k\}^n$ such that $A(W) = I_k$. (Note that switching from fields to $k \times k$ matrix algebras causes us to lose a factor of k^2 in the bound.)

The first step is to prove a non-commutative version of Lemma 4.2.

Lemma 10.3 *Let $Y_1, \ldots, Y_t$ be any points in $M_k(\mathbb{F})^n$. Then there exists a sorted-multilinear polynomial $m : M_k(\mathbb{F})^n \to M_k(\mathbb{F})$ such that*

(i) $m(Y_i) = 0_k$ for all $i \in [t]$, and

(ii) $m(W) = I_k$ for at least $2^n - k^2 t$ Boolean points $W \in \{0_k, I_k\}^n$.

Proof. If we represent m as

$$m(X) = \sum_{Z \in \{0_k, I_k\}^n} m_Z \delta_Z(X),$$

then the constraint $m(Y_i) = 0_k$ for all $i \in [t]$ corresponds to $k^2 t$ linear equations relating the 2^n coefficients m_Z. By basic linear algebra, it follows that there must be a solution in which at least $2^n - k^2 t$ of the coefficients are equal to I_k, and hence $m(W) = I_k$ for at least $2^n - k^2 t$ Boolean points W. ■

Using Lemma 10.3, we can also prove a non-commutative version of Lemma 4.3.

Lemma 10.4 *Let $Y_1, \ldots, Y_t$ be any points in $M_k(\mathbb{F})^n$. Then for at least $2^n - k^2 t$ Boolean points $W \in \{0_k, I_k\}^n$, there exists a multiquadratic polynomial $p : M_k(\mathbb{F})^n \to M_k(\mathbb{F})$ such that*

(i) $p(Y_i) = 0_k$ for all $i \in [t]$,

(ii) $p(W) = I_k$, and

(iii) $p(Z) = 0_k$ for all Boolean $Z \neq W$.

Proof. Let $m : M_k(\mathbb{F})^n \to M_k(\mathbb{F})$ be the sorted-multilinear polynomial from Lemma 10.3, and pick any Boolean W such that $m(W) = I_k$. Then a multiquadratic polynomial p satisfying properties (i)-(iii) can be obtained from m as follows:

$$p(X) := m(X)\, \delta_W(X).$$

■

Lemma 10.4 immediately gives us a non-commutative version of Theorem 4.4, the lower bound on deterministic query complexity of the OR function.

Theorem 10.5 $\widetilde{\mathrm{D}}_{M_k(\mathbb{F}),2}(\mathrm{OR}) = \Omega\left(2^n/k^2\right)$ *for every matrix algebra $M_k(\mathbb{F})$.*

By using Theorem 10.5, for every $k = O(\mathrm{poly}(n))$ one can construct an oracle A, and a $k \times k$ matrix extension $\widetilde{A}$ of A, such that $\mathsf{NP}^{\widetilde{A}} \not\subset \mathsf{P}^A$. This then implies that any resolution of the P versus NP problem will require "non-commutatively non-algebrizing techniques."

11 Conclusions and Open Problems

Arithmetization is one of the most powerful ideas in complexity theory. It led to the $\mathsf{IP} = \mathsf{PSPACE}$ Theorem, the PCP Theorem, non-relativizing circuit lower bounds, and many other achievements of the last two decades. Yet we showed that arithmetization is fundamentally unable to resolve many of the barrier problems in the field, such as P versus NP, derandomization of RP, and circuit lower bounds for NEXP.

Can we pinpoint what it is about arithmetization that makes it incapable of solving these problems? In our view, arithmetization simply fails to "open the black box wide enough." In a typical arithmetization proof, one starts with a polynomial-size Boolean formula φ, and uses φ to produce a low-degree polynomial p. But having done so, one then treats p as an arbitrary black-box function, subject only to the constraint that $\deg(p)$ is small. Nowhere does one exploit the small size of φ, except insofar as it lets one evaluate p in the first place. The message of this paper has been that, to make further progress, one will have to probe φ in some "deeper" way.

To reach this conclusion, we introduced a new model of *algebraic query complexity*, which has already found independent applications in communication complexity, and which has numerous facets to explore in its own right.

We now propose five directions for future work, and list some of the main open problems in each direction.

(1) Find non-algebrizing techniques. This, of course, is the central challenge we leave.

If arithmetization—which embeds the Boolean field $\mathbb{F}_2$ into a larger field or the integers—is not enough, then a natural idea is to embed $\mathbb{F}_2$ into a "richer" algebra. But in Section 10.2 we showed that for every subexponential k, the algebra of $k \times k$ matrices is still not "sufficiently rich." So the question arises: what other useful algebraic structures can mathematics offer complexity theory?

Another possible way around the algebrization barrier is "recursive arithmetization": first arithmetizing a Boolean formula, then reinterpreting the result as a Boolean function, then arithmetizing *that* function, and so on ad infinitum. In Section 10.1, we showed that k-arithmetization is not powerful enough to prove $\mathsf{P} \neq \mathsf{NP}$ for any constant k. But we have no idea whether double-arithmetization is already powerful enough to prove $\mathsf{P} = \mathsf{RP}$ or $\mathsf{NEXP} \not\subset \mathsf{P/poly}$.

(2) Find ways to exploit the *structure* of polynomials produced by arithmetization. This is also a possible way around the algebrization barrier, but seems important enough to deserve its own heading. The question is: given that a polynomial $\widetilde{A} : \mathbb{F}^n \to \mathbb{F}$ was produced by arithmetizing a small Boolean formula, *does $\widetilde{A}$ have any properties besides low degree that a polynomial-time algorithm querying it could exploit*? Or alternatively, do there exist "pseudorandom extensions" $\widetilde{A} : \mathbb{F}^n \to \mathbb{F}$—that is, low-degree extensions that are indistinguishable from "random" low-degree extensions by any $\mathsf{BPP}^{\widetilde{A}}$ machine, but that were actually produced by arithmetizing small Boolean formulas? As a hint of how the structure of $\widetilde{A}$ might be exploited, let us point out that, if $\widetilde{A}$ was produced by arithmetizing a $3SAT$ formula φ, then one can actually recover φ in polynomial time by making oracle queries to $\widetilde{A}$.[16]

(3) Find open problems that can still be solved with algebrizing techniques. In the short term, this is perhaps the most "practical" response to the algebrization barrier. Here is a problem that, for all we know, might still be solvable with tried-and-true arithmetization methods: improve the result of Santhanam [32] that $\mathsf{PromiseMA} \not\subset \mathsf{SIZE}\left(n^k\right)$ to $\mathsf{MA} \not\subset \mathsf{SIZE}\left(n^k\right)$.

(4) Prove algebraic oracle separations. Can we show that the interactive protocol of Lund, Fortnow, Karloff, and Nisan [23] cannot be made constant-round by any algebrizing technique? In other words, can we give an oracle A and extension $\widetilde{A}$ such that $\mathsf{coNP}^{A} \not\subset \mathsf{AM}^{\widetilde{A}}$? In the communication complexity setting, Klauck [21] mentions coNP versus AM as a difficult open problem; perhaps the algebraic query version is easier.

The larger challenge is to give algebraic oracles that separate all the levels of the polynomial hierarchy—or at least separate the polynomial hierarchy from larger classes such as $\mathsf{P}^{\#\mathsf{P}}$ and

[16] The algorithm is as follows. Assume for simplicity that every clause of φ contains exactly 3 literals. Then for each triple of variables (x_i, x_j, x_k):

(1) Set the remaining $n-3$ variables to constants.

(2) Query $\widetilde{A}$ on $O(\deg(\widetilde{A})^3)$ values of x_i, x_j, x_k, then use interpolation to recover a trivariate polynomial $p\,(x_i, x_j, x_k)$.

(3) Try to divide p by each of the eight multilinear polynomials ($1 - x_i x_j x_k$, $1 - x_i x_j\,(1 - x_k)$, etc.) corresponding to the possible $3SAT$ clauses involving x_i, x_j, x_k. Whenever such a polynomial divides p, we have learned another clause of φ.

The only thing to be careful about is that, in setting the remaining $n-3$ variables to constants, we do not inadvertently set $p\,(x_i, x_j, x_k) = 0$. But this is easily arranged.

PSPACE.[17] In the standard oracle setting, these separations were achieved by Furst-Saxe-Sipser [11] and Yao [39] in the 1980's, whereas in the communication setting they remain notorious open problems. Again, algebraic query complexity provides a natural intermediate case between query complexity and communication complexity.

Can we show that non-algebrizing techniques would be needed to give a Karp-Lipton collapse to MA? Or give an interactive protocol for coNP where the prover has the power of NP?

Can we show that a $\mathsf{BQP}^{\widetilde{A}}$ or $\mathsf{MA}^{\widetilde{A}}$ machine needs exponentially many queries to the extension oracle $\widetilde{A}$, not only to solve the Disjointness problem, but also just to find a Boolean point x such that $\widetilde{A}(x) = 1$? Also, in the integers case, can we show that a $\mathsf{P}^{\widehat{A}}$ machine needs exponentially many queries to $\widehat{A}$ to find an x such that $\widehat{A}(x) = 1$? (That is, can we remove the technical limitations of Theorem 6.10 regarding the size of the inputs and outputs?)

(5) Understand algebrization better. In defining what it meant for inclusions and separations to algebrize, was it essential to give only one machine access to the extension oracle $\widetilde{A}$, and the other access to A? Or could we show (for example) not only that $\mathsf{coNP}^{A} \subseteq \mathsf{IP}^{\widetilde{A}}$, but also that $\mathsf{coNP}^{\widetilde{A}} \subseteq \mathsf{IP}^{\widetilde{A}}$? What about improving the separation $\mathsf{PP}^{\widetilde{A}} \not\subset \mathsf{SIZE}^{A}(n^k)$ to $\mathsf{PP}^{\widetilde{A}} \not\subset \mathsf{SIZE}^{\widetilde{A}}(n^k)$? Likewise, can we improve the separation $\mathsf{MA}_{\mathsf{EXP}}^{\widetilde{A}} \not\subset \mathsf{P}^{A}/\mathsf{poly}$ to $\mathsf{MA}_{\mathsf{EXP}}^{\widetilde{A}[\mathrm{poly}]} \not\subset \mathsf{P}^{A}/\mathsf{poly}$?

Are there complexity classes $\mathcal{C}$ and $\mathcal{D}$ that can be separated by a finite field extension $\widetilde{A}$, but *not* by an integer extension $\widehat{A}$? Are there complexity classes that can be separated in the algebraic oracle setting, but not the communication setting?

Low-degree extensions can be seen as just one example of an error-correcting code. To what extent do our results carry over to arbitrary error-correcting codes?

Arora, Impagliazzo, and Vazirani [2] showed that contrary relativizations of the same statement (for example, $\mathsf{P}^{A} = \mathsf{NP}^{A}$ and $\mathsf{P}^{B} \neq \mathsf{NP}^{B}$) can be interpreted as proving independence from a certain formal system. Can one interpret contrary algebrizations the same way?

Acknowledgments

We thank Benny Applebaum, Sanjeev Arora, Boaz Barak, Andy Drucker, Lance Fortnow, Russell Impagliazzo, Hartmut Klauck, Adam Klivans, Ryan O'Donnell, Rahul Santhanam, Sasha Sherstov, Amir Shpilka, Madhu Sudan, Luca Trevisan, and Ryan Williams for helpful discussions.

References

[1] S. Aaronson. Oracles are subtle but not malicious. In *Proc. IEEE Conference on Computational Complexity*, pages 340–354, 2006. ECCC TR05-040.

[17] If the oracle $\widetilde{A}$ only involves a low-degree extension over $\mathbb{F}_q$, for some fixed prime $q = o(n/\log n)$, then we can give $A, \widetilde{A}$ such that $\mathsf{PP}^{A} \not\subset \mathsf{PH}^{\widetilde{A}}$. The idea is the following: let $\widetilde{A}$ be the unique multilinear extension of A over $\mathbb{F}_q$. Clearly a PP^{A} machine can decide whether $\sum_{x\in\{0,1\}^n} A(x) \geq 2^{n-1}$. On the other hand, supposing a $\mathsf{PH}^{\widetilde{A}}$ machine solved the same problem, we could interpret the universal quantifiers as AND gates, the existential quantifiers as OR gates, and the queries to $\widetilde{A}$ as summation gates modulo q. We could thereby obtain an $\mathsf{AC}^0[q]$ circuit of size $2^{\mathrm{poly}(n)}$, which computed the Boolean MAJORITY given an input of size 2^n (namely, the truth table of A). But when $q = o(n/\log n)$, such a constant-depth circuit violates the celebrated lower bound of Smolensky [34].

Unfortunately, the above argument breaks down when the field size is large compared to n—as it needs to be for most algorithms that would actually exploit oracle access to $\widetilde{A}$. Therefore, it could be argued that this result is not "really" about algebrization.

[2] S. Arora, R. Impagliazzo, and U. Vazirani. Relativizing versus nonrelativizing techniques: the role of local checkability. Manuscript, 1992.

[3] L. Babai, L. Fortnow, and C. Lund. Nondeterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1(1):3–40, 1991.

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

[5] C. H. Bennett and J. Gill. Relative to a random oracle A, $P^A \neq NP^A \neq coNP^A$ with probability 1. *SIAM J. Comput.*, 10(1):96–113, 1981.

[6] H. Buhrman, L. Fortnow, and T. Thierauf. Nonrelativizing separations. In *Proc. IEEE Conference on Computational Complexity*, pages 8–12, 1998.

[7] H. Buhrman, N. Vereshchagin, and R. de Wolf. On computation and communication with small bias. In *Proc. IEEE Conference on Computational Complexity*, pages 24–32, 2007.

[8] H. Buhrman and R. de Wolf. Complexity measures and decision tree complexity: a survey. *Theoretical Comput. Sci.*, 288:21–43, 2002.

[9] U. Feige and J. Kilian. Making games short. In *Proc. ACM STOC*, pages 506–516, 1997.

[10] L. Fortnow. The role of relativization in complexity theory. *Bulletin of the EATCS*, 52:229–244, February 1994.

[11] M. Furst, J. B. Saxe, and M. Sipser. Parity, circuits, and the polynomial time hierarchy. *Math. Systems Theory*, 17:13–27, 1984.

[12] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *J. ACM*, 38(1):691–729, 1991.

[13] J. Hartmanis, R. Chang, S. Chari, D. Ranjan, and P. Rohatgi. Relativization: a revisionistic perspective. *Bulletin of the EATCS*, 47:144–153, 1992.

[14] J. Hartmanis and R. E. Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 117:285–306, 1965.

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

[16] R. Impagliazzo, V. Kabanets, and A. Wigderson. In search of an easy witness: exponential time vs. probabilistic polynomial time. *J. Comput. Sys. Sci.*, 65(4):672–694, 2002.

[17] A. Juma, V. Kabanets, C. Rackoff, and A. Shpilka. The black-box query complexity of polynomial summation. ECCC TR07-125, 2007.

[18] B. Kalyanasundaram and G. Schnitger. The probabilistic communication complexity of set intersection. *SIAM J. Discrete Math*, 5(4):545–557, 1992.

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

[20] M. Karchmer and A. Wigderson. Monotone circuits for connectivity require super-logarithmic depth. *SIAM J. Comput.*, 3:255–265, 1990.

[21] H. Klauck. Rectangle size bounds and threshold covers in communication complexity. In *Proc. IEEE Conference on Computational Complexity*, pages 118–134, 2003. cs.CC/0208006.

[22] A. Klivans and D. van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM J. Comput.*, 31:1501–1526, 2002. Earlier version in ACM STOC 1999.

[23] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39:859–868, 1992.

[24] D. van Melkebeek. A survey of lower bounds for satisfiability and related problems. *Foundations and Trends in Theoretical Computer Science*, 2:197–303, 2007. ECCC TR07-099.

[25] W. J. Paul, N. Pippenger, E. Szemerédi, and W. T. Trotter. On determinism versus non-determinism and related problems. In *Proc. IEEE FOCS*, pages 429–438, 1983.

[26] R. Raz. Exponential separation of quantum and classical communication complexity. In *Proc. ACM STOC*, pages 358–367, 1999.

[27] R. Raz and A. Shpilka. On the power of quantum proofs. In *Proc. IEEE Conference on Computational Complexity*, pages 260–274, 2004.

[28] A. A. Razborov. Lower bounds for the size of circuits of bounded depth with basis $\{\&, \oplus\}$. *Mathematicheskie Zametki*, 41(4):598–607, 1987. English translation in *Math. Notes. Acad. Sci. USSR* 41(4):333–338, 1987.

[29] A. A. Razborov. On the distributional complexity of disjointness. *Theoretical Comput. Sci.*, 106:385–390, 1992.

[30] A. A. Razborov. Quantum communication complexity of symmetric predicates. *Izvestiya Math. (English version)*, 67(1):145–159, 2003. quant-ph/0204025.

[31] A. A. Razborov and S. Rudich. Natural proofs. *J. Comput. Sys. Sci.*, 55(1):24–35, 1997.

[32] R. Santhanam. Circuit lower bounds for Merlin-Arthur classes. In *Proc. ACM STOC*, pages 275–283, 2007.

[33] A. Shamir. IP=PSPACE. *J. ACM*, 39(4):869–877, 1992.

[34] R. Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *Proc. ACM STOC*, pages 77–82, 1987.

[35] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 20(5):865–877, 1991.

[36] L. Trevisan and S. Vadhan. Pseudorandomness and average-case complexity via uniform reductions. In *Proc. IEEE Conference on Computational Complexity*, pages 129–138, 2002.

[37] N. V. Vinodchandran. A note on the circuit complexity of PP. ECCC TR04-056, 2004.

[38] A. Wigderson. Information theoretic reasons for computational difficulty. In *Proceedings of the International Congress of Mathematicians*, pages 1537–1548, 1990.

[39] A. C-C. Yao. Separating the polynomial-time hierarchy by oracles (preliminary version). In *Proc. IEEE FOCS*, pages 1–10, 1985.

[40] A. C-C. Yao. How to generate and exchange secrets (extended abstract). In *Proc. IEEE FOCS*, pages 162–167, 1986.