An Axiomatic Approach to Algebrization

Russell Impagliazzo*
UC San Diego, La Jolla, CA &
Institute for Advanced Study
Princeton, NJ, USA
russell@cs.ucsd.edu

Valentine Kabanets†
Simon Fraser University
Burnaby, BC, Canada
kabanets@cs.sfu.ca

Antonina Kolokolova‡
Memorial U. of Newfoundland
St. John's, NL, Canada
kol@cs.mun.edu

ABSTRACT

Non-relativization of complexity issues can be interpreted as giving some evidence that these issues cannot be resolved by "black-box" techniques. In the early 1990's, a sequence of important non-relativizing results was proved, mainly using algebraic techniques. Two approaches have been proposed to understand the power and limitations of these algebraic techniques: (1) Fortnow [12] gives a construction of a class of oracles which have a similar algebraic and logical structure, although they are arbitrarily powerful. He shows that many of the non-relativizing results proved using algebraic techniques hold for all such oracles, but he does not show, e.g., that the outcome of the "P vs. NP" question differs between different oracles in that class. (2) Aaronson and Wigderson [1] give definitions of algebrizing separations and collapses of complexity classes, by comparing classes relative to one oracle to classes relative to an algebraic extension of that oracle. Using these definitions, they show both that the standard collapses and separations "algebrize" and that many of the open questions in complexity fail to "algebrize", suggesting that the arithmetization technique is close to its limits. However, it is unclear how to formalize algebrization of more complicated complexity statements than collapses or separations, and whether the algebrizing statements are, e.g., closed under *modus ponens*; so it is conceivable that several algebrizing premises could imply (in a relativizing way) a non-algebrizing conclusion.

In this paper, building on the work of Arora, Impagliazzo, and Vazirani [4], we propose an *axiomatic* approach to "algebrization", which complements and clarifies the approaches of [12] and [1]. We present logical theories formalizing the notion of algebrizing techniques in the following sense: most known complexity results proved using arithmetization are *provable* within our theories, while many open questions are *independent* of the theories. So provability in the proposed theories can serve as a surrogate for provability using the arithmetization technique.

Our theories extend the [4] theory with a new axiom, *Arithmetic Checkability* which intuitively says that all NP languages have verifiers that are efficiently computable low-degree polynomials (over the integers). We show the following: *(i)* Arithmetic checkability holds relative to arbitrarily powerful oracles (since Fortnow's algebraic oracles from [12] all satisfy the Arithmetic Checkability axiom). *(ii)* Most of the algebrizing collapses and separations from [1], such as IP = PSPACE, NP $\subset$ ZKIP if one-way functions exist, MA-EXP $\not\subset$ P/poly, etc., are *provable* from Arithmetic Checkability. *(iii)* Many of the open complexity questions (including most of those shown to require non-algebrizing techniques in [1]), such as "P vs. NP", "NP vs. BPP", etc., *cannot be proved* from Arithmetic Checkability. *(iv)* Arithmetic Checkability is also insufficient to prove one known result, NEXP = MIP (although relative to an oracle satisfying Arithmetic Checkability, NEXP^O restricted to poly-length queries is contained in MIP^O, mirroring a similar result from [1]).

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems; F.4.1 [**Theory of Computation**]: Mathematical logic

General Terms

Theory

1. INTRODUCTION

Many basic questions in Complexity Theory (e.g., P vs. NP) have so far resisted all the attacks using currently known techniques. To understand better the limitations of the "currently known techniques", it is natural to try to identify some general property that these techniques share, and show what is provable and what is not provable by any techniques satisfying this property. The hope is that such classification of techniques will guide the search for new techniques that may potentially resolve some of the open questions.

There have been several such "meta-results" in Complexity Theory. In the mid-1970's, Baker, Gill and Solovay [7] used *relativization* as a tool to argue that techniques like simulation and diagonalization cannot, by themselves, resolve the "P vs. NP" question. Intuitively, a technique relativizes if it is insensitive to the presence of oracles (thus, a result about complexity classes holds for all oracle versions

*Work partially supported by NSF grants 0835373 and 0832797 and the Ellentuck Foundation
†Work partially supported by NSERC Discovery grant
‡Work partially supported by NSERC Discovery grant

STOC'09, May 31–June 2, 2009, Bethesda, Maryland, USA.

of these classes). If there are oracles giving a contradictory resolution of a complexity question (e.g., $\mathsf{P}^A = \mathsf{NP}^A$, but $\mathsf{P}^B \neq \mathsf{NP}^B$), then no relativizing technique can resolve this question. This method of relativization has been brought to bear upon many other open questions in Complexity Theory, for example, P vs. PSPACE [7], NP vs. EXP [11, 14, 23], BPP vs. NEXP [17], IP vs. PSPACE [13], and a long list of other classes.

In an informal sense, contrary relativizations of a complexity theory statement have been viewed as a *mini-independence result*, akin to the independence results shown in mathematical logic. But what *independence* is implied by contradictory relativizations, and what are the *proof techniques* from which this independence is implied? This was made precise in [4]. There the authors introduced a theory $\mathcal{RCT}$ (which stands for Relativized Complexity Theory) based on Cobham's axiomatization of polynomial-time computation [10]. Roughly speaking, $\mathcal{RCT}$ has standard axioms of arithmetic (Peano axioms), and an axiomatic definition of the class of functions that is supposed to correspond to the class P. This class is defined as the closure of a class of some basic functions under composition and limited recursion (as in Cobham's paper [10]), plus there is an extra axiom postulating the existence of a universal function for that class. $\mathcal{RCT}$'s view of the complexity class P is "black-box": the axioms are satisfied not only by the class P, but also by every relativized class P^O, for every oracle O. In fact, [4] shows that the (standard) models of $\mathcal{RCT}$ are exactly the classes P^O, over all oracles O.[1] It follows that, for any complexity statement S about P, this statement S is true relative to every oracle A (i.e., S relativizes) iff S is provable in $\mathcal{RCT}$. On the other hand, a non-relativizing statement is precisely a statement independent of $\mathcal{RCT}$. Thus, e.g., the "P vs. NP" question is independent of $\mathcal{RCT}$.

[4] also shows that extending $\mathcal{RCT}$ with another axiom, which captures the "local checkability" of a Turing machine computation in the style of the Cook-Levin theorem (the computation tableau can be checked by checking that all 2×3 "windows" are correct), almost exactly characterizes the class P in the following sense: the models for the resulting theory, denoted by $\mathcal{LCT}$ (for Local Checkability) in [4], are necessarily of the form P^O with $O \in \mathsf{NP}\cap\mathsf{co\text{-}NP}$. This makes the theory $\mathcal{LCT}$ "too strong", in the sense that resolving most complexity questions in $\mathcal{LCT}$ is essentially equivalent to resolving them in the non-relativized setting.

In the early 1990's, a sequence of important non-relativizing results was proved, mainly using algebraic techniques. Although the techniques used to obtain these results seem similar in flavor, it is not clear what common features they are exploiting. It is also not clear to what extent oracle results should be trusted as a guide to estimating the difficulty of proving complexity statements, in light of these algebraic techniques. Finally, it is unclear what the true power of these techniques is. Could they resolve the longstanding open problems in complexity, such as P vs. NP, or BPP vs. P? To answer this question requires a formalization of the "arithmetization technique" and its power.

Two approaches to this question have been formulated. Fortnow [12] gives a construction of a class of oracles which have a similar algebraic and logical structure, although they are arbitrarily powerful. He shows that many of the non-relativizing results proved using algebraic techniques hold for all such oracles. While this is revealing, it is only a partial characterization of the technique. For example, he does not show that the outcome of P vs. NP differs between different oracles in that class. The second approach is due to Aaronson and Wigderson [1] who give definitions of algebrizing separations and collapses of complexity classes, by comparing classes relative to one oracle to classes relative to an algebraic extension of that oracle. Using these definitions, they show that the standard collapses "algebrize" and that many of the open questions in complexity fail to "algebrize", suggesting that the arithmetization technique is close to its limits. However, it is unclear how to formalize algebrization of more complicated complexity statements than collapses or separations, and it is unclear whether the algebrizing statements are, e.g., closed under *modus ponens*. So, in particular, it is conceivable that several algebrizing premises could imply (in a relativizing way) a non-algebrizing conclusion.

Our results.

In this paper we provide an axiomatic framework for algebraic techniques in the style of [4]. We extend their theory $\mathcal{RCT}$ with an axiom capturing the notion of arithmetization: the *Arithmetic Checkability* axiom. Intuitively, Arithmetic Checkability postulates that all NP languages have verifiers that are polynomial-time computable families of *low-degree polynomials*; a verifier for an NP language is a polynomial f (say, over the integers) such that the verifier accepts a given witness y on an input x iff $f(x, y) \neq 0$. Standard techniques (the characterization of "real world" non-deterministic Turing Machines in terms of consistent tableaux, and algebraic interpolations of Boolean functions) show that this axiom holds for unrelativized computation.

The models for the resulting theory, which we call $\mathcal{ACT}$ (for Arithmetic Checkability Theory), are, essentially, all relativized classes P^O with oracles O such that Arithmetic Checkability holds relative to this O, i.e., all NP^O languages have P^O-computable families of low-degree polynomials as verifiers. Arithmetic Checkability is implied by, yet is strictly weaker than Local Checkability, since $\mathcal{ACT}$ has models P^O for arbitrarily powerful oracles O (in particular, any oracle from Fortnow's [12] recursive construction). Thus, $\mathcal{ACT}$ is a theory that lies between the [4] theories $\mathcal{RCT}$ and $\mathcal{LCT}$. Moreover, both inclusions are proper: $\mathcal{RCT} \subsetneq \mathcal{ACT} \subsetneq \mathcal{LCT}$. That is, there are statements provable in $\mathcal{ACT}$ that can't be proved in $\mathcal{RCT}$, and there are statements provable in $\mathcal{LCT}$ that can't be proved in $\mathcal{ACT}$.

We use the Arithmetic Checkability axiom (and the theory $\mathcal{ACT}$ based on it) as an axiomatic framework to capture the "arithmetization technique". We show that many complexity theorems (like the ones shown to algebrize in [1]) are *provable in* $\mathcal{ACT}$, and that many open complexity questions (like the ones shown not to algebrize in [1]) are *independent of* $\mathcal{ACT}$. Since all provable consequences of $\mathcal{ACT}$ are closed under deduction, we avoid the limitations of the approach in [1]. However, there are some known complexity statements (proved using algebraic techniques) that are also independent from $\mathcal{ACT}$.

The following is a summary of our main results: *(i)* Fortnow's algebraic oracles from [12] all satisfy Arithmetic Checkability (so arithmetic checkability holds relative to arbitrarily powerful oracles). *(ii)* Most of the algebrizing collapses and separations from [1], such as IP = PSPACE, NP $\subset$ ZKIP

[1] Cobham [10] gets the exact characterization of P by considering the *minimal* model for his theory.

if one-way functions exist, MA-EXP $\not\subset$ P/poly, etc., are *provable* from Arithmetic Checkability. *(iii)* Many of the open complexity questions (including most of those shown to require non-algebrizing techniques in [1]), such as P vs. NP, P vs. BPP, the existence of explicit functions without small circuits, etc., *cannot be proved* from Arithmetic Checkability. *(iv)* Arithmetic Checkability is also insufficient to prove one known result, NEXP = MIP (although relative to an oracle satisfying Arithmetic Checkability, NEXP^O restricted to poly-length queries is contained in MIP^O, mirroring a similar result from [1]).

Finally, comparing the three notions of "algebrizing techniques" of Fortnow [12], Aaronson and Wigderson [1], and the present paper, we observe that all relativizing techniques are contained in the [1] algebrizing techniques, which usually coincide with our notion of algebrizing techniques, which in turn are contained in the [12] algebrizing techniques.

Remainder of the paper. In Section 2, we define the axiom of Arithmetic Checkability and study its properties. Section 3 contains a number of provable consequences of $\mathcal{ACT}$, and Section 4 a number of complexity statements independent from $\mathcal{ACT}$.

2. ARITHMETIC CHECKABILITY

Here we define the axiom of arithmetic checkability, and prove some of its basic properties. We could view this axiom in one of two ways. Proof-theoretically, we could add the axiom to the theory $\mathcal{RCT}$ of [4], and view the results provable in this theory as those provable with relativizing and algebrizing techniques. Somewhat simpler conceptually, for the purposes of this paper, we take a model-theoretic viewpoint, where we look at the *class of oracles* that satisfy (are consistent with) the new axiom. In the model-theoretic viewpoint, a complexity statement is algebrizing if it holds for all oracles that satisfy the new Arithmetic Checkability axiom.[2] In this abstract, we will only consider the model-theoretic interpretation, to avoid a long discussion of the [4] axioms.

The Arithmetic Checkability axiom intuitively says that every easily computable function can be interpolated into an easily computable, low-degree polynomial by adding extra variables. While extensions of Boolean functions to polynomials makes sense over many fields and rings, for simplicity, we limit ourselves to polynomials over the integers.

Below we define two versions of the Arithmetic Checkability axiom: one for checkability of *nondeterministic* computation (where the verifier polynomial accepts a proof if its output is any non-zero integer), and one for *deterministic* computation (where the verifier polynomial accepts a proof if its output is 1, and, moreover, the proof is unique and efficiently computable). We call the first version weak ACT (or simply ACT), and the second version strong ACT (or ACT*). Most of our positive results (in Section 3) are provable from the weak version of ACT, while all our independence results (in Section 4) are with respect to the stronger theory based on ACT*.

[2]It is somewhat stronger to say that a statement *does algebrize* in the proof-theoretic sense than in the model-theoretic sense (because the statement may be true for all such oracles without being provable). Contrapositively, an independence result is stronger in the model-theoretic sense than in the proof-theoretic sense (because we only consider standard models). All of our positive implications hold in the proof-theoretic sense, and all of our independence results hold in the model-theoretic sense.

DEFINITION 2.1. *A* polynomial family *is a family of polynomials $f_n : Z^n \to Z$, where, for each $n \in \mathbb{N}$, f_n is an n-variate polynomial over $\mathbb{Z}$ of total degree $n^{O(1)}$. It is* polynomial-time computable *if the function $F(n, y_1, ..y_n) = f_n(y_1, ..y_n)$ is in* FP.

The class ALG-PF *(algebraically checkable proof systems) is the class of languages L such that there is a polynomial-time computable polynomial family $\{f_n\}$ and a polynomially bounded polynomial-time computable function $m = m(n)$ so that $x = x_1 \dots x_n \in L$ iff $\exists y_1 \dots y_m \in \{0,1\}^m$ such that $f_{n+m}(x_1, \dots, x_n, y_1, \dots, y_m) \neq 0$.*

The (weak) Arithmetic Checkability axiom *is the statement* NP = ALG-PF. *We will denote this axiom by* ACT *(for Arithmetic Checkability Theorem). The* theory $\mathcal{ACT}$ *is defined to be $\mathcal{RCT}$ + ACT (i.e., the theory $\mathcal{RCT}$ together with the axiom ACT). An oracle A is* consistent with ACT *if* $\mathsf{NP}^A = \mathsf{ALG\text{-}PF}^A$.

The class $\mathsf{ALG\text{-}PF}^*$ *is the class of languages L such that there are a polynomially bounded polynomial-time computable function $m = m(n)$, a polynomial-time computable function family $\{g_n : \{0,1\}^n \to \{0,1\}^m\}$, and a polynomial-time computable polynomial family $\{f_n\}$ so that* (i) *if $x = x_1 \dots x_n \in L$ then $f_{n+m}(x, g_n(x)) = 1$,* (ii) *if $x = x_1 \dots x_n \notin L$ then $f_{n+m}(x, g_n(x)) = 0$, and* (iii) *for all $y \in \{0,1\}^m$, $y \neq g_n(x) \implies f(x, y) = 0$. The* strong ACT*, denoted by ACT*, is the statement* $\mathsf{P} = \mathsf{ALG\text{-}PF}^*$*; the corresponding strong version of $\mathcal{ACT}$ is denoted $\mathcal{ACT}^*$. An oracle A is* consistent with ACT* *if* $\mathsf{P}^A = \mathsf{ALG\text{-}PF}^{*A}$.

A set A of integers is self-algebrizing *if there is a polynomial family $\tilde{A}$ extending A under projection, e.g., $A[x_1, \dots, x_n] = \tilde{A}_{n+1}[0, x_1, \dots, x_n]$ for Boolean x, and such that $\tilde{A} \in \mathsf{P}^A$.*

We will relate arithmetic checkability to a notion of *local checkability* from [4]. The latter essentially says that (non-deterministic) computation can be verified in terms of a small number of conditions that each involve a small part of an input and proof.

DEFINITION 2.2. *Let* $\mathsf{PF\text{-}CHK}[poly, log]$ *be the class of languages L so that there is a polynomial $p(n)$ and a polynomial-time computable verifier $V^{x,\pi}(1^n, r)$ with random access to the input x and a proof $\pi \in \{0,1\}^{p(n)}$ so that V makes at most $O(\log n)$ queries to the input and proof, and so that $x \in L$ if and only if $\exists \pi \in \{0,1\}^{p(|x|)} \forall r \in [1, \dots, p(n)]\ V^{x,\pi}(1^n, r) = 1$.*

The Local Checkability axiom *is the statement* $\mathsf{NP} = \mathsf{PF\text{-}CHK}[poly, log]$*, which we also denote by* LCT *(for Local Checkability Theorem). The theory $\mathcal{LCT}$ is $\mathcal{RCT}$+LCT. An oracle O is* consistent with LCT *if* $\mathsf{NP}^O = \mathsf{PF\text{-}CHK}[poly, log]^O$.

[4] and many others have observed that NP = $\mathsf{PF\text{-}CHK}[poly, log]$ follows from the standard proof of the Cook-Levin Theorem in terms of tableaux. [4] also observed that all oracles O consistent with the version of LCT defined above are in $\mathsf{NP/poly} \cap \mathsf{coNP/poly}$, and that NP^O reduces to unrelativized NP (via P^O reductions). This severely limits the power of such oracles, and the number of provable independence results from $\mathcal{LCT}$.

Here, we show that most (but not all) of the known complexity consequences of local checkability actually follow from the weaker statement, ACT, but that ACT (even ACT*) does not suffice to resolve many of the open problems in

complexity. Thus, provability in $\mathcal{ACT}$ is a good surrogate for "provable with relativizing and algebrizing techniques". Independence from $\mathcal{ACT}$ suggests that not only do we need non-black-box techniques, but also we need to go beyond algebraic interpolation as the only non-black-box technique.

The following theorem relates $\mathcal{ACT}$ to $\mathcal{LCT}$ and the notion of algebrizing suggested by [12].

THEOREM 2.3. *(1.) Any language A that is consistent with LCT is also consistent with ACT.*
(2.) For any language A, and any polynomial extension $\tilde{A}$, $\mathsf{P}^A \subseteq \mathsf{ALG\text{-}PF}^{*\tilde{A}}$ and $\mathsf{NP}^A \subseteq \mathsf{ALG\text{-}PF}^{\tilde{A}}$.
(3.) Any language A that is self-algebrizing is consistent with ACT^* and ACT.

PROOF. *(1.)* We claim for any A, $\mathsf{PF\text{-}CHK}[poly, log]^A \subseteq \mathsf{ALG\text{-}PF}^A$. If $L \in \mathsf{PF\text{-}CHK}[poly, log]^A$, let V^A be a $O(\log n)$-query proof checker that accepts L, i.e., $x \in L$ if and only if $\exists \pi \in \{0,1\}^{n^c} \forall r \in [1, \ldots, n^c]$ $M^{A,x,\pi}(1^n, r) = 1$. For each r computation, it is possible to compute (using oracle A) a decision tree of queries to bits of its inputs (x, π) of depth $c \log n$ (and hence polynomial size) that expresses acceptance along this path. (Note that we bound the number of queries of M to x or π to $O(\log n)$, but M may make any number of queries to A.) We can then represent this decision tree as a degree $c \log n$ polynomial by taking the sum over all accepting paths of the product of the corresponding literals, where the negation of a variable z is represented by $1 - z$. Let the resulting polynomial be called $p_r(x_1, \ldots, x_n, \pi_1, \ldots, \pi_m)$. On Boolean inputs, p_r will have value 1 if $M^{A,x,\pi}(1^n, r)$ accepts and 0 otherwise. We can then let $p(x_1, \ldots, x_n, \pi_1, \ldots, \pi_m) = \prod_r p_r(x_1, \ldots, x_n, \pi_1, \ldots, \pi_m)$. Each p_r and hence p can be computed in polynomial time, and p has degree at most $O(n^c \log n)$. So p is a polynomial-time computable polynomial family. For Boolean inputs, p is 1 if all p_r are 1 and 0 otherwise, so p is 1 if and only if $M^{A,x,\pi}(1^n, r) = 1$ for all r.

(2.) Let A be an oracle with algebraic extension $\tilde{A}$. Let $L \in \mathsf{P}^A$ be accepted by a machine M^A. We define the proof to be the tableau of the computation on x of the deterministic machine M^A, together with the oracle answers given as bits $b_1, \ldots, b_T$. Since M is deterministic, such a proof is unique. Let g be the function mapping inputs x into proofs (i.e., tableaux of M^A and oracle answers). Clearly, $g(x)$ is computable in $\mathsf{FP}^A \subseteq \mathsf{FP}^{\tilde{A}}$.

Without loss of generality, we can assume the time step and length of the i'th oracle query is known in advance (say, by clocking the maximum number of steps between queries and by having the machine make dummy queries of all possible lengths in order). So the i'th query will be at a fixed l_i consecutive positions in the tableau. An accepting tableau is valid if each square of six is possible for the machine, and if each $b_i = A[q_{i_1}, \ldots, q_{i_{l_i}}]$. The first set of conditions can each be written as a polynomial using the decision tree method above, and the second by the polynomials $1 - (b_i - \tilde{A}_{l_i+1}[0, q_{i_1}, \ldots, q_{i_{l_i}}])^2$. (Note that on Boolean inputs starting with a 0, $\tilde{A}$ is either 0 or 1, as is b_i. Therefore the above polynomial is either 0 or 1.) Then the total correctness is the product of these polynomials, which is clearly of polynomial degree, is computable in $\mathsf{P}^{\tilde{A}}$, is Boolean-valued on Boolean inputs, and is 1 on a given input x and a proof y iff y is the unique correct proof that $x \in L$. Hence, $L \in \mathsf{ALG\text{-}PF}^{*\tilde{A}}$.

For the case of $L \in \mathsf{NP}^A$, we define $L' \in \mathsf{P}^A$ to be the set of those pairs (x, y) such that $x \in L$ and y is a witness for $x \in L$. By the above, we get that $L' \in \mathsf{ALG\text{-}PF}^{*\tilde{A}}$, with a polynomial family f. It follows that $x \in L$ iff there exist y and z such that $f(x, y, z) = 1$, where z represents the proof that $(x, y) \in L'$ (moreover, z is unique and efficiently computable from (x, y), but we do not need this extra property here). Thus, we get $L \in \mathsf{ALG\text{-}PF}^{\tilde{A}}$.

(3.) Follows from (2), since if A is self-algebrizing, then $\mathsf{ALG\text{-}PF}^{*A} = \mathsf{ALG\text{-}PF}^{*\tilde{A}}$ and $\mathsf{ALG\text{-}PF}^A = \mathsf{ALG\text{-}PF}^{\tilde{A}}$. □

Part (3) essentially says that any technique that "algebrizes" in our sense also algebrizes in the sense of Fortnow [12]. While we cannot prove that "being a consequence of ACT" characterizes algebrizing techniques in the sense of [1], part (2) is an explanation why results that algebrize in the two senses frequently coincide. Namely, to show $\mathsf{NP} \subseteq C_2$ algebrizes in both senses, it suffices to show that $\mathsf{ALG\text{-}PF} \subseteq C_2$ relativizes.[3]

The following theorem summarizes a construction due to Fortnow, which shows that the self-algebrizing languages come in arbitrarily strong complexities. By Theorem 2.3, it follows that so do oracles consistent with ACT^* and ACT, a property which LCT-consistent oracles do not have.

THEOREM 2.4 ([12]). *For each language L there is a self-algebrizing language A such that $A \in \mathsf{PSPACE}^L$ and $L \in \mathsf{P}^A$.*

PROOF. We will give two constructions.

Fortnow's construction [12]: Let $\langle y_1, \ldots, y_k \rangle$ be a standard pairing function such that $|\langle y_1, \ldots, y_k \rangle| > |y_1| + \cdots + |y_k|$. For a language A, denote by A_n the restriction of A to $\{0,1\}^n$. We define $A = \cup_{n \geq 1} A_n$ inductively over n as follows. Set $A_1 = \emptyset$. Suppose A_n is already defined for some $n \geq 1$. Let $f_n(x_1, \ldots, x_n)$ be the unique multilinear polynomial extension of $A_n(x_1, \ldots, x_n)$. We extend the definition of A according to the following three cases.

1. For each $b_1 \ldots b_n \in \{0,1\}^n$, we put $\langle 0, b_1, \ldots, b_n \rangle$ into A iff $b_1 \ldots b_n \in L$.

2. For each n-tuple of integers $(c_1, \ldots, c_n)$, we put $\langle 1, c_1, \ldots, c_n \rangle$ into A iff $f_n(c_1, \ldots, c_n) > 0$.

3. For each n-tuple of integers $(c_1, \ldots, c_n)$ and an integer $i \geq 0$, we put $\langle i + 2, c_1, \ldots, c_n \rangle$ into A iff the ith bit of the binary representation of the integer value $f_n(c_1, \ldots, c_n)$ is one.

It is easy to see that the constructed language A is self-algebrizing, and hence, in particular, $L \in \mathsf{P}^A$. To see that $A \in \mathsf{PSPACE}^L$, observe that for each $n \geq 1$ and $b_1 \ldots b_n \in \{0,1\}^n$, the value $A_n(b_1, \ldots, b_n)$ is computable either in $\mathsf{P}^{L_{n'}}$ for some $n' < n$ (in case 1 above), or in $\mathsf{PSPACE}^{A_{n''}}$ for some $n'' < n$ (in cases 2 or 3, since the multilinear extension $f_{n''}$ of $A_{n''}$ can be evaluated at any given point, using a polynomial-space algorithm with oracle access to $A_{n''}$). In other words, A is downward self-reducible, with a polynomial-space reduction. The latter easily implies that $A \in \mathsf{PSPACE}^L$.

Alternative construction: For a given language L, let A be any PSPACE^L-complete language (e.g., the TQBF^L language). Then the unique multilinear extension $\tilde{A}$ is computable in $\mathsf{PSPACE}^A \subseteq \mathsf{PSPACE}^L \subseteq \mathsf{P}^A$, where the first

[3]More generally, for $C_1 \subseteq C_2$, give a containment for C_1 from some construction over NP, and then give a relativizing inclusion of the same construction over ALG-PF inside C_2.

inclusion is because $A \in \mathsf{PSPACE}^L$, and the second one because A is PSPACE^L-hard. Finally, observe that $L \in \mathsf{P}^A$ since $L \in \mathsf{PSPACE}^L \subseteq \mathsf{P}^A$. $\square$

We will call the self-algebrizing language A obtained from a given language L using Theorem 2.4 the *self-algebrizing encoding* of L.

3. CONSEQUENCES OF ACT

First we show that the famous $\mathsf{PSPACE} = \mathsf{IP}$ theorem [24, 27] can be proved from Arithmetic Checkability.

THEOREM 3.1. *Let O be any oracle consistent with ACT. Then $\mathsf{PSPACE}^O = \mathsf{IP}^O$.*

PROOF SKETCH. For any O, the relativized version of TQBF is a complete problem for PSPACE^O: Given an input $x_1 \dots x_n \in \{0,1\}^n$, decide if $\exists y_1 \in \{0,1\}\ \forall z_1 \in \{0,1\}\ \dots\ \exists y_m \in \{0,1\}\ \forall z_m \in \{0,1\}\ \ P(\vec{x}, \vec{y}, \vec{z})$, where $P \in \mathsf{P}^O$ and m is a polynomially bounded function of n. Since $P \in \mathsf{P}^O \subseteq \mathsf{NP}^O = \mathsf{ALG\text{-}PF}^O$, we can write $P(\vec{x}, \vec{y}, \vec{z})$ as $\exists w_1 \dots w_{m'} \in \{0,1\}^{m'}\ [f_{n+2m+m'}(\vec{x}, \vec{y}, \vec{z}, \vec{w}) \neq 0]$, where f is a polynomial family computable in P^O.

Fix input x, merge the w's with y's and z's, and consider f^2; for simplicity of notation, we use $2m$ (rather than $2(m+m')$) for the number of variables in the resulting polynomial. The problem is to decide if $\exists y_1 \in \{0,1\}\ \forall z_1 \in \{0,1\}\ \dots\ \exists y_m \in \{0,1\}\ \forall z_m \in \{0,1\}\ [p_{2m}(\vec{y}, \vec{z}) \neq 0]$, where p_{2m} is an efficiently computable polynomial that is always non-negative. We can follow the same protocol as in the standard proof of the $\mathsf{PSPACE} = \mathsf{IP}$ theorem (see, e.g., [3]), using the fact that p_{2m} is computable in P^O. $\square$

We also get that many known circuit lower bounds (based on the collapses like $\mathsf{PSPACE} = \mathsf{IP}$) are provable from ACT or ACT*. The corresponding non-relativized versions of the lower bounds in the next theorem (items 1–3) are from [9, 29, 26], respectively; the last item is from [5].

THEOREM 3.2. *Let O and O^* be any oracles consistent with ACT and ACT*, respectively. Then all of the following statements hold: (1) $\mathsf{MA\text{-}EXP}^O \not\subset \mathsf{P}^O/\mathsf{poly}$; (2) For each constant k, $\mathsf{PP}^{O^*} \not\subset \mathsf{SIZE}^{O^*}(n^k)$; (3) For each constant k, $\mathsf{promise\text{-}MA}^{O^*} \not\subset \mathsf{SIZE}^{O^*}(n^k)$; (4) $\mathsf{NEXP}^{O^*[\mathsf{poly}]} \subseteq \mathsf{MIP}^{O^*}$.*

PROOF SKETCH. The proofs are similar to the corresponding proofs in [1].

(1): Observe that for any ACT-consistent language O, if $\mathsf{PSPACE}^O \subset \mathsf{P}^O/\mathsf{poly}$, then $\mathsf{PSPACE}^O = \mathsf{MA}^O$ following the same argument as in the unrelativized case: the prover in the IP^O-protocol for PSPACE^O is computable in PSPACE^O, and hence, Merlin can give to Arthur a small circuit for this prover, and Arthur can simulate the IP^O-protocol by interacting with the circuit.

If $\mathsf{PSPACE}^O \not\subset \mathsf{P}^O/\mathsf{poly}$, then $\mathsf{MA\text{-}EXP}^O \not\subset \mathsf{P}^O/\mathsf{poly}$, and we are done. Otherwise, we get by the argument above that $\mathsf{PSPACE}^O = \mathsf{MA}^O$, which by padding yields $\mathsf{EXPSPACE}^O = \mathsf{MA\text{-}EXP}^O$. Finally, by diagonalization, we conclude that $\mathsf{EXPSPACE}^O \not\subset \mathsf{P}^O/\mathsf{poly}$.

(2): Let O^* be any ACT*-consistent oracle. Relative to O^*, counting the number of accepting paths of a given NP-machine on a given input x is reducible to the polynomial summation problem $\sum_{z_1,\dots,z_m \in \{0,1\}} p(z_1, \dots, z_n)$, where p is a polynomial family computable in FP^{O^*}. Indeed, let $L \in \mathsf{NP}^{O^*}$ be any language decided by a nondeterministic machine N^{O^*}. Let $L' \in \mathsf{P}^{O^*}$ be the language consisting of those pairs (x, y) such that $x \in L$ and y describes an accepting computation of N on x. By the definition of ACT*-consistency, there is a polynomial-time computable polynomial family f such that, for any (x, y), we have $(x, y) \in L'$ iff there is some Boolean w such that $f(x, y, w) = 1$, and moreover, such Boolean w is unique (if exists). It follows that the number of accepting computations y on a given input x is exactly $\sum_{y,w \in \{0,1\}^*} f(x, y, w)$.[4]

We can now use the LFKN protocol [24] to argue that $\#\mathsf{P}^{O^*}$ has proof checkers. In particular, we get that if $\mathsf{PP}^{O^*} \subset \mathsf{P}^{O^*}/\mathsf{poly}$, then $\mathsf{P}^{\#\mathsf{P}^{O^*}} = \mathsf{PP}^{O^*} = \mathsf{MA}^{O^*}$.

This is sufficient to prove item (2) of Theorem 3.2, arguing as in the non-relativized case. Indeed, if $\mathsf{PP}^{O^*} \not\subset \mathsf{P}^{O^*}/\mathsf{poly}$, then we are done. Otherwise, we have by the above that $\mathsf{P}^{\#\mathsf{P}^{O^*}} = \mathsf{PP}^{O^*} = \mathsf{MA}^{O^*}$, and by relativizing Toda's theorem [28], we have $(\Sigma_2^p)^{O^*} \subseteq \mathsf{P}^{\#\mathsf{P}^{O^*}}$. Finally, by relativizing Kannan's theorem [20], we get that for every fixed constant k, $(\Sigma_2^p)^{O^*} \not\subset \mathsf{SIZE}^{O^*}(n^k)$, and therefore, also $\mathsf{MA}^{O^*} \not\subset \mathsf{SIZE}^{O^*}(n^k)$. Since $\mathsf{MA} \subseteq \mathsf{PP}$ (and this inclusion relativizes), we get item (2) of Theorem 3.2.

(3): We follow the corresponding proof in [2], using the PP^{O^*}-complete problem of deciding, for a given efficiently computable low-degree polynomial, whether its sum over the Boolean domain is at least a given integer k. The argument is the same as that in [2].

(4): The proof of item (4) is also based on the proof of the corresponding non-relativized result [5]. As pointed out in [2], the tableau of a NEXP-machine remains locally checkable even if we allow oracle access to any oracle A, provided that the machine is only allowed to ask *polynomial-length* oracle queries. Since O^* is ACT*-consistent, we have access to the polynomial extensions of O^* and it complement, and therefore can arithmetize the local-check algorithm for a given $\mathsf{NEXP}^{O^*[\mathsf{poly}]}$-machine. The result of arithmetization is a polynomial that is 1 iff the check is satisfied. By subtracting 1 and squaring the result, we get a new polynomial that is 0 iff the check is satisfied, and greater than 0 otherwise. Then the problem is to verify that the sum of all these local-check polynomials is 0. This can be done as in the original proof of [5]; see [2] for details. $\square$

Note that Theorem 3.2 above shows that $\mathsf{NEXP}^{O^*} \subseteq \mathsf{MIP}^{O^*}$ is provable from ACT* only for the case of *polynomial-length* oracle queries (the restriction assumed also in [1]). This is unavoidable. As we show below (Theorem 4.1, item 3), it is *impossible* to prove $\mathsf{NEXP} \subseteq \mathsf{MIP}$ from ACT*.

We also show that the famous GMW theorem [15] ($\mathsf{NP} \subseteq \mathsf{ZKIP}$ if one-way functions exist) can be proved from ACT. We prove that the theorem holds relative to every oracle O consistent with ACT. A similar result for a restricted case was also shown in [1], but they have since independently obtained essentially the same result as we do below for their setting [2]. (More precisely, their new result shows that if $\tilde{A}$ is an algebraic extension of oracle A, and there is a one-way function with respect to $\tilde{A}$, then $\mathsf{NP}^A \subseteq \mathsf{ZKIP}^{\tilde{A}}$. While the

[4]This is where we use the assumption that O^* is consistent with the *strong* version of ACT. We do not know if the weak version of ACT suffices to prove the items (2)–(4) of Theorem 3.2, and leave it as an interesting open question.

phrasing is different, their protocol is very close to ours.)

THEOREM 3.3. *Let O be any oracle consistent with ACT and such that there is a one-way function in P^O secure against adversaries in BPP^O. Then $\mathsf{NP}^O \subseteq \mathsf{ZKIP}^O$.*

PROOF. If there are one-way functions, then there are semantically secure statistically binding commitment schemes ([25, 16]; all arguments there relativize). Let C be such a scheme. For simplicity of notation, we drop the randomness used by C and refer to any commitment to a number a as $C(a)$.

The prover in our protocol will use *indirect* commitments. An indirect commitment to a given value a is a pair of commitments $C(r)$ and $C(a+r)$, where r is a randomly chosen residue mod q, and addition is modulo q (for some q chosen at random within the protocol). A general subroutine is to prove that a certain set of indirectly committed values satisfies a linear relationship $\sum \alpha_i a_i = 0$, where the α_i's are publicly known and the a_i's are indirectly committed to. We refer to this as a *linear relationship test*. In such a test, the prover will have sent (in addition to the indirect commitment $(C(r_i), C(a_i + r_i))$) a commitment $C(\sum \alpha_i r_i) = C(s)$. The verifier then flips a coin; if heads, the prover decommits to all r_i's and s, and the verifier checks that $s = \sum \alpha_i r_i$. Since the r_i's are random, and s really is this sum, this reveals no information. Alternatively, if the coin is tails, the prover decommits to s and all $(a_i + r_i)$'s and the verifier does the same check that $s = \sum(\alpha_i(a_i + r_i))$. Again, these are random numbers and a predefined linear combination of them, so this reveals no information about the a_i's. Note that both checks work for the same value of s if and only if $\sum \alpha_i a_i = 0$.

Let $L \in \mathsf{NP}^O$. Then there is a polynomial-time (with respect to O) computable polynomial family $\{f_k\}$ such that $x = x_1 \dots x_n \in L$ iff $\exists y_1, \dots, y_m \in \{0,1\}^m \quad f_{n+m}(\vec{x}, \vec{y}) \neq 0$. For the rest of the proof, we fix x, and think of f as a polynomial in the variables y_i only. To simplify the notation, we will denote this new polynomial by $f(y_1, \dots, y_m)$. We denote by d the degree of f (as a polynomial in the y_i's).

First, the prover selects $\vec{y} = (y_1, \dots, y_m)$ such that $f(\vec{y}) \neq 0$, and the verifier selects a moderately sized random prime q. With high probability $f(\vec{y}) \neq 0 \mod q$. (The prime q should be sufficiently larger than d, and in fact could be larger than the possible values of f on m-bit inputs, in which case $f(\vec{y}) \neq 0 \mod q$ is certainly true.)

Next, the prover picks a random bit b, and directly commits to b. If $b = 0$, the prover indirectly commits to y_i, $i = 1, \dots, m$, and then to $(1 - y_i)$, $i = 1, \dots, m$. If $b = 1$, the prover does these indirect commitments in reverse, i.e., indirectly commits to the $(1 - y_i)$'s and then to the y_i's.

The prover also picks a random non-zero vector $\vec{s} = (s_1, \dots, s_m) \in \mathbb{Z}_q^m$ and indirectly commits to the following values: (a) each s_j, for $j = 1, \dots, m$, (b) each coordinate of $z(t) = \vec{y} + t\vec{s}$ for each $t = 1, 2, \dots, d+1$, (c) the value $v_t = f(z(t))$ for each $t = 0, 1, \dots, d+1$, and (d) the coefficients $c_0, \dots c_d$ of the univariate polynomial $f(\vec{y} + t\vec{s})$ (in the variable t).

Note that the values v_t can be computed easily by the prover since f is in P^O. Using these values, the prover can also compute the coefficients $c_0, \dots, c_d$ by interpolation.

Finally, let r_0 be the random number used in the indirect commitment $(C(r_0), C(v_0 + r_0))$ to v_0. The prover picks values a, b at random (with $a \neq 0$), and directly commits to a, b, $ar_0 + b$, and $a(r_0 + v_0) + b$. The prover does similarly for the random numbers used in the indirect commitments to the s_j's, for $1 \leq j \leq m$.

The verifier chooses one of the following tests at random:
Test for Booleanness. The verifier picks a random i and the prover reveals the two bits corresponding to y_i and $1 - y_i$ (but does not reveal b.) If the prover follows protocol, these bits are just 0 and 1 in a random order. If the verifier does not choose this test, b is revealed.
Non-zeroness test. The verifier views one of the three possibilities: $a, b, r_0, ar_0 + b$ (checking that the last really is computed correctly from the first three); $a, b, (r_0 + v_0), a(r_0 + v_0) + b$, (similarly checking); or, $ar_0 + b$ and $a(r_0 + v_0) + b$, checking that these two are distinct (hence $v_0 \neq 0$, if they are correctly computed).
Non-degeneracy test. The verifier picks a random $1 \leq j \leq m$, and performs the non-zeroness test on s_j, as described above.
Test of polynomial values. The verifier picks a random t and tests that $v_t = \sum_{i=0}^{d} c_i t^i$, i.e., that the committed polynomial really has value v_t at this point. This is a linear relationship test, handled as described above (without actually revealing v_t or any of the coefficients).
Test of linearity of the $z(t)$'s. The verifier picks a random $t \neq 0$ and a coordinate $1 \leq j \leq m$, and verifies that $y_j + ts_j = z(t)_j$. Again, this is a linear relationship test.
Test of consistency with f. The verifier picks a random $t \neq 0$ and tests that $v_t = f(z(t))$. For this, the prover completely reveals $z(t)$ and v_t. If the prover follows the protocol, $z(t)$ is a random vector independent of y, and v_t is the (easily computable) value above, so the revealed information can be simulated by the verifier.

It is easy to see that any of the tests performed by the verifier reveals no information that the verifier could not simulate by himself; so the described protocol is zero-knowledge. Completeness of the protocol is obvious. For soundness, suppose that the committed values pass all of the above tests, then it follows that: There is an indirectly committed vector $\vec{y}$ that is Boolean, and a non-zero value v_0, and a polynomial $c(t)$ of degree d with $c(t) = v_t$ for each t. There are points $z(t) = \vec{y} + t\vec{s}$ for some non-zero vector $\vec{s}$. And $f(\vec{y} + t\vec{s}) = v_t$ for $d+1$ non-zero values of t. Since c and the restriction of f to this line both have degree d and agree on $d+1$ points, they must be equal polynomials. Therefore they have the same value on $t = 0$, so $f(\vec{y}) = c(0) = v_0 \neq 0$. Therefore, $x \in L$. □

4. INDEPENDENCE RESULTS

Using Fortnow's construction of Theorem 2.4, we get a rich family of oracles consistent with ACT*, which we can use to prove a number of complexity statements independent from $\mathcal{ACT}^*$, including the known true statement that $\mathsf{NEXP} \subseteq \mathsf{MIP}$ [5]. The following theorem (the last item) shows that even the weaker statement $\mathsf{E} \subseteq \mathsf{MIP}$ is *not* provable in $\mathcal{ACT}^*$.

THEOREM 4.1. *$\mathcal{ACT}^*$ does* not *imply any of the following: (1) $\mathsf{P} \neq \mathsf{PSPACE}$ (and hence, also $\mathsf{P} \neq \mathsf{NP}$); (2) $\mathsf{E} \subseteq \mathsf{io\text{-}SIZE}(2^{n/4})$[5]; (3) $\mathsf{E} \subseteq \mathsf{MIP}$.*

PROOF. *(1):* Let L be a language complete for PSPACE, and let A be its self-algebrizing encoding (obtained using

[5]See also Theorem 4.2, item (1), for a stronger result.

Theorem 2.4). We get that A is consistent with ACT*, and $L \in \mathsf{P}^A$ and $A \in \mathsf{PSPACE}^L$. Then $\mathsf{PSPACE}^A \subseteq \mathsf{PSPACE}^L = \mathsf{P}^L \subseteq \mathsf{P}^A \subseteq \mathsf{PSPACE}^A$. In particular, $\mathsf{P}^A = \mathsf{PSPACE}^A$.

(2): For L and A as in the previous item, we have $\mathsf{PSPACE}^A = \mathsf{P}^A$. By padding (which relativizes), we get $\mathsf{SPACE}^A(2^{O(n)}) = \mathsf{E}^A$. By counting, $\mathsf{SPACE}^A(2^{O(n)}) \not\subset \mathsf{io\text{-}SIZE}^A(2^{n/4})$. Hence, $\mathsf{E}^A \not\subset \mathsf{io\text{-}SIZE}^A(2^{n/4})$.

(3): Since both PSPACE^L and MIP^L only depend on strings in L of polynomial-size, it is easy to diagonalize to construct an oracle L so that $\mathsf{E}^L \not\subset \mathsf{MIP}^{\mathsf{PSPACE}^L}$. Now let A be the self-algebrizing encoding of this L, and so A is consistent with ACT*. We get $\mathsf{MIP}^A \subseteq \mathsf{MIP}^{\mathsf{PSPACE}^L}$ and $\mathsf{E}^L \subseteq \mathsf{E}^A$, so $\mathsf{E}^A \not\subseteq \mathsf{MIP}^A$. □

Using Fortnow's construction of Theorem 2.4 together with communication complexity lower bounds, we get the following independence results, which show that many of the complexity frontier questions (non-determinism, derandomization, quantum computing, circuit lower bounds) are *not resolvable within* $\mathcal{ACT}^*$. Note that item (2) of the theorem below is an example of a more complicated statement than just a single inclusion, or a single separation statement.

THEOREM 4.2. *$\mathcal{ACT}^*$ does* not *imply any of the following: (1)* $\mathsf{NP} \subseteq \mathsf{io\text{-}SIZE}(2^{n/4})$ *(which, with the previous theorem, implies independence for* $\mathsf{NP} = \mathsf{P}$ *and* $\mathsf{NP} \subseteq \mathsf{BPP}$*); (2)* $\mathsf{BPP} \neq \mathsf{P}$ *or* $\mathsf{P} = \mathsf{NP}$*; (3)* $\mathsf{BPP} \subseteq \mathsf{DTIME}(2^{o(n)})$*; (4)* $\mathsf{EXP} \not\subseteq \mathsf{io\text{-}P/poly}$*; (5)* $\mathsf{coNP} \subseteq \mathsf{MA}$*; (6)* $\mathsf{NP} \subseteq \mathsf{BQP}$*; (7)* $\mathsf{BQP} \subseteq \mathsf{BPP}$*; (8)* $\mathsf{QMA} \subseteq \mathsf{MA}$.

Similarly to [1], we prove these non-algebrization results by reduction to communication complexity. However, our reduction is a bit more indirect than theirs. We now sketch the reduction to communication complexity. The proof of Theorem 4.2 is given in Section 4.1 below.

For any two languages A_0 and A_1, let $A_0 + A_1 = \{(b, x) \mid x \in A_b\}$ be their disjoint union. We show that for any two languages A_0 and A_1 consistent with ACT*, $A_0 + A_1$ is also ACT*-consistent (see Lemma 4.3 below).

Let L_1 and L_2 be arbitrary oracles, which we think of as two inputs to a communication protocol, such as for set disjointness (e.g., we are trying to see if there is an x of length n so that $x \in L_1 \cap L_2$). Using Theorem 2.4, we can construct from each oracle L_i its self-algebrizing encoding A_i, which is consistent with ACT*. Then L_i is reducible to A_i, and $A_i \in \mathsf{PSPACE}^{L_i}$. (For the communication-complexity setting we will consider, we actually won't even need any upper bound on the complexity of A_i.)

Consider the communication complexity problem relative to the oracle $A_1 + A_2$. Set disjointness is easily solved in $\mathsf{coNP}^{L_1+L_2}$, and hence also in $\mathsf{coNP}^{A_1+A_2}$. We will argue that it can't always be solved in $\mathsf{P}^{A_1+A_2}$, since otherwise we would get a deterministic communication protocol for set disjointness on $N = 2^n$-bit input strings of communication complexity only polynomial in n, which is impossible by the well-known $\Omega(N)$ lower bounds for set disjointness (see, e.g., [22]). The idea is that queries to $A_1 + A_2$ are either to A_1, which depends only on L_1, or to A_2, which depends only on A_2. Thus, any algorithm with such an oracle can be simulated by two players, Alice and Bob, where Alice knows L_1 and Bob knows L_2. The overall communication complexity of the resulting protocol is exactly the number of oracle queries. The same reasoning holds for almost any other model of communication complexity, e.g., probabilistic, quantum, and non-deterministic communication complexities. We'll use stronger distributional lower bounds for the direct product of many set disjointness problems of [8] to extend this to a strong circuit lower bound.

This strategy can be used to prove all parts except (2) and (4). However, (2) and (4) follow directly from (1) and (3), and the fact that the hardness-randomness tradeoffs from [18, 6] relativize. We provide more details in the next subsection.

4.1 Proof of theorem 4.2

First we prove that the class of ACT*-consistent oracles is closed under disjoint union.

LEMMA 4.3. *If A_0 and A_1 are consistent with ACT*, then $A_0 + A_1$ is also consistent with ACT*.*

The proof will depend on the following.

LEMMA 4.4. $\mathsf{P}^A = \mathsf{ALG\text{-}PF}^{*A}$ *if and only if* $A \in \mathsf{ALG\text{-}PF}^{*A} \cap \mathsf{coALG\text{-}PF}^{*A}$.

PROOF. It is obviously necessary. For the other direction, let p_1 be a P^A-computable polynomial family for A, and let p_0 be a P^A-computable polynomial family for $\bar{A}$ (the complement of A). Let g_0 and g_1 be the corresponding FP^A-computable functions that compute proofs for membership in A and $\bar{A}$, respectively. We will show that for every language $L \in \mathsf{P}^A$, there is a P^A-computable polynomial family showing that $L \in \mathsf{ALG\text{-}PF}^{*A}$, and a FP^A-computable function g mapping inputs to proofs.

Let $L \in \mathsf{P}^A$ be decided by a P^A machine M^A. This machine accepts input $x = x_1 \dots x_n \in \{0,1\}^n$ iff there is an accepting tableau $w = w_1 \dots w_T$ of the machine on x, with bits $b_1 \dots b_t$ representing answers to oracle queries $q_1, \dots, q_t$, and $z_1, \dots, z_t$ being witnesses corresponding to the queries (where z_i's are provided by the function g_0 or g_1, depending on b_i being 0 or 1), so that *(1)* w is a correct accepting tableau, assuming that all oracle answers are correct (i.e., assuming that $b_i = A(q_i)$ for all $1 \leq i \leq t$), and *(2)* all oracle answers are correct.

Observe that this tableau, oracle queries q_i, oracle answers b_i, and oracle witnesses z_i (i.e., a proof that $x \in L$) are all computable in FP^A. We let g be the FP^A-computable function mapping inputs x to such proofs. Also, without loss of generality, we may assume that for each query q_i the dimension of the witness z_i is the same for both p_0 and p_1. Indeed, suppose that the dimension of witness for p_0 is m_0, which is less than the dimension m_1 of the witness for p_1. Let $\ell = m_1 - m_0$. Define $\tilde{g}_0 : \{0,1\}^n \to \{0,1\}^{m_1}$ by $\tilde{g}_0(x) = g_0(x)1^\ell$ (i.e., $g_0(x)$ followed by ℓ ones). Define $\tilde{p}_0(\vec{x}, y_1, \dots, y_{m_1}) = p_0(\vec{x}, y_1, \dots, y_{m_0}) \cdot \prod_{i=m_0+1}^{m_1} y_i$. Clearly, the defined $\tilde{p}_0$ and $\tilde{g}_0$ also show that $\bar{A} \in \mathsf{ALG\text{-}PF}^{*A}$. (The case of $m_1 < m_0$ is similar.)

The first condition above can be expressed by a low-degree polynomial (in the variables x, w, b, q) in a standard way (as the product, over all 2×3 "windows" of the tableau, of the polynomials expressing the correctness of the window). The second condition can also be expressed as the product, over $1 \leq i \leq t$, of the following low-degree polynomial p on the variables q_i, b_i, z_i: $p(q_i, b_i, z_i) = b_i \cdot p_1(q_i, z_i) + (1 - b_i) \cdot p_0(q_i, z_i)$, which is 1 (for Boolean-valued b_i) iff [$b_i = 1$ and z_i is a witness that $q_i \in A$] or [$b_i = 0$ and z_i is a witness

that $q_i \notin A$]. Finally, the product of the polynomials for these two conditions yields a polynomial family computable in FP^A, showing that $L \in \mathsf{ALG\text{-}PF}^{*A}$. □

Proof of Lemma 4.3. By the easy direction of Lemma 4.4, we get that A_0 and its complement are in $\mathsf{ALG\text{-}PF}^{*A_0}$, and similarly, A_1 and its complement are in $\mathsf{ALG\text{-}PF}^{*A_1}$. Let R_0 and R_1 be polynomial families for A_0 and A_1. Let g_0 and g_1 be the corresponding witness-computing functions for R_0 and R_1. Suppose that $g_0 : \{0,1\}^n \to \{0,1\}^{m_0}$ and $g_1 : \{0,1\}^n \to \{0,1\}^{m_1}$. As before, we may assume without loss of generality that $m_0 = m_1 = m$.

The polynomial families for A_0 are computable in FP^{A_0}, and those for A_1 in FP^{A_1}. Define the polynomial family R for $A_0 + A_1$ by $R((b,x),y) = b \cdot R_1(x,y) + (1-b) \cdot R_0(x,y)$, where $x = x_1,\ldots,x_n$ and $y = y_1,\ldots,y_m$. Define the witness-computing function g for R on Boolean inputs (b,x) equal to $g_b(x)$. These R and g are computable in $\mathsf{FP}^{A_0+A_1}$. It follows that $A_0 + A_1 \in \mathsf{ALG\text{-}PF}^{*A_0+A_1}$.

Similarly, we can argue that the complement of $A_0 + A_1$ is also in $\mathsf{ALG\text{-}PF}^{*A_0+A_1}$. So by Lemma 4.4, $A_0 + A_1$ is consistent with ACT*. □

Next we prove the items of Theorem 4.2. We show the existence of ACT*-consistent oracles for which the negations of the corresponding statements in Theorem 4.2 hold. Each of these oracles will have the form of a disjoint union of two ACT*-consistent oracles, and hence, by Lemma 4.3, these oracles are also ACT*-consistent, as required. We give the details next.

4.1.1 $\mathsf{NP} \not\subseteq \mathsf{io\text{-}SIZE}(2^{n/4})$

We'll use the following result from [8, Cor. 4.12, page 27].

Theorem 4.5 ([8]). *There is a constant c so that, if N and k are integers with $k \le 2^{c\sqrt{N}}$, then the following holds: Consider the distribution on sets $S \subseteq N$ where each element is independently added to S with probability $1/\sqrt{N}$. Let $S_1,\ldots,S_k$ independently chosen sets from this distribution be the input to player 1, and similarly independently chosen such sets $T_1,\ldots,T_k$ be the input to player 2. Then any communication protocol to determine, for each $1 \le i \le k$, whether $S_i \cap T_i = \emptyset$, with $o(k\sqrt{N})$ bits of communication, has at most $2^{-\Omega(k)}$ probability of success.*

Lemma 4.6. *There exist languages L_1 and L_2 with the corresponding self-algebrizing encodings A_1 and A_2 such that $\mathsf{NP}^{A_1+A_2} \not\subseteq \mathsf{io\text{-}SIZE}^{A_1+A_2}(2^{n/4})$.*

Proof. Let L_1 and L_2 be chosen at random so that queries of the form (x,y), where $|x| = |y| = 2n$, are in L_b independently with probability 2^{-n}, and no other queries are in L_b, for $b = 1, 2$. Let $M(L_1,L_2) = \{x | \exists y, (x,y) \in L_1 \cap L_2\}$. For every L_1, L_2, $M(L_1,L_2)$ is in $\mathsf{NP}^{L_1+L_2} \subseteq \mathsf{NP}^{A_1+A_2}$. We claim that the probability, for each even length $2n$, that there is a circuit with oracle $A_1 + A_2$ of sub-exponential size for $M(L_1,L_2)$ is doubly exponentially small in n. It follows that there is a non-zero probability that the circuit complexity is exponentially large for all but finitely many n.

To see this, fix n, let $K = 2^{2n}$, and let $N = 2^{2n}$. Condition on all elements of L_1 and L_2 not of the form (x,y) where $|x| = |y| = 2n$ (up to size, say 2^{2n}, so that conditioning is finite).

For each oracle circuit C of size $2^{n/4}$, describable using $2^{\alpha n}$ bits of advice for some suitable constant α, we can define a communication protocol for the direct product of K random set intersection problems as follows: Let $S_1,\ldots,S_K$ be the inputs to player 1, $T_1,\ldots,T_K$ to player 2. Player 1 adds to L_1 all queries (i,y) where $y \in S_i$ and computes A_1 (note that A_1 only depends on the part of L_1 of strictly smaller length, so the part of A_1 up to length 2^n is defined by the part of L_1 up to length 2^n). Similarly, player 2 computes L_2 and A_2 from $T_1,\ldots,T_K$. The players then simulate $C^{A_1+A_2}$ on all inputs x of length $2n$ (in an arbitrary order). Whenever a query is made to A_1, player 1 gives the value, and similarly for A_2. They output the tuple of outputs for C on all such x's. Since for each x, the number of queries C makes is at most $2^{n/4}$, the total number of bits communicated is at most $2^{2n} \cdot 2^{n/4} = K \cdot 2^{n/4} = o(K\sqrt{N})$. Therefore, by Theorem 4.5, the probability of success is at most $2^{-\Omega(K)} = 2^{-\Omega(2^{2n})}$. Note that if the protocol fails, then $C^{A_1+A_2}$ fails to compute $M(L_1,L_2)$ for length $2n$ inputs. Taking a union bound over all $2^{2^{\alpha n}}$ such circuits, the probability (over random L_1 and L_2) that there is an oracle circuit of size $2^{n/4}$ that correctly decides the language $M(L_1,L_2)$ on $2n$-bit inputs is doubly exponentially small for any $\alpha < 2$.

Since the sum of these probabilities over all n converges, there is an n_0 so that there is a non-zero chance (over the choice of L_1 and L_2) of an exponential circuit-size lower bound for all $n > n_0$. □

4.1.2 $\mathsf{BPP} = \mathsf{P}$ *and* $\mathsf{P} \neq \mathsf{NP}$

Corollary 4.7. *There exist languages L_1 and L_2 with the corresponding self-algebrizing encodings A_1 and A_2 such that $\mathsf{BPP}^{A_1+A_2} = \mathsf{P}^{A_1+A_2}$ and $\mathsf{NP}^{A_1+A_2} \neq \mathsf{P}^{A_1+A_2}$.*

Proof. The language $M(L_1,L_2)$ defined in the proof of Lemma 4.6 is always in E. By relativizing [18], it follows that $\mathsf{BPP}^{A_1+A_2} = \mathsf{P}^{A_1+A_2}$ for the same choice of A_1 and A_2. At the same time, that language $M(L_1,L_2)$ is in $\mathsf{NP}^{A_1+A_2} \setminus \mathsf{P}^{A_1+A_2}$. □

4.1.3 $\mathsf{RP} \not\subseteq \mathsf{DTIME}(2^{o(n)})$

Lemma 4.8. *There exist languages L_1 and L_2 with the corresponding self-algebrizing encodings A_1 and A_2 such that $\mathsf{RP}^{A_1+A_2} \not\subseteq \mathsf{DTIME}^{A_1+A_2}(2^{o(n)})$.*

Proof. We use the separation for deterministic and probabilistic communication complexities. Let n_i be a sequence of integers with $n_{i+1} > 2^{n_i}$.

Think of L_1 and L_2 as inputs to the inequality problem. More precisely, let $B(L_1,L_2) = \{1^n | \exists x, |x| = n, x \in L_1 \Delta L_2\}$, where Δ denotes symmetric difference. Relative to any $A_1 + A_2$, $B(L_1,L_2)$ is always in $\mathsf{RP}^{A_1+A_2}$. This is because A_1 and A_2 are self-algebrizing and extend L_1 and L_2, and hence can be used to compute polynomial extensions $\tilde{L}_1$ and $\tilde{L}_2$ of L_1 and L_2, respectively. Observe that $L_1 = L_2$ on length n inputs if and only if the same is true for $\tilde{L}_1$ and $\tilde{L}_2$ on dimension n inputs. If $L_1 \neq L_2$ on inputs of length n, then $\tilde{L}_1$ and $\tilde{L}_2$ will differ with high probability on a random input of dimension n. This gives the *RP* algorithm.

We will construct L_1 and L_2 so that there is no subexponential-time machine M deciding the inequality problem for all large enough inputs. We pick L_1 and L_2 to only contain strings of length n_i (and be empty elsewhere). Consider an enumeration of clocked $2^{n/2}$-time oracle machines $M_1, M_2, \ldots$. Note

that the oracle machine $M_i(1^{n_i})$ cannot ask oracle queries of length n_{i+1}.

We will construct the languages L_1 and L_2 so that they are empty everywhere outside the input lengths n_i's. Suppose that after stage i, both L_1 and L_2 are defined for inputs of length up to n_i (and empty elsewhere). Also suppose that each of the first i oracle machines $M_1, \ldots, M_i$ incorrectly solves the inequality problem for L_1 and L_2 for some length less than n_i, when given oracle access to A_1 and A_2 which are the self-algebrizing encodings of the languages L_1 and L_2 (defined after stage i; as before, for each $b = 1, 2$, the set A_b up to length n_{i+1} is determined by L_b up to length n_i, since there are no elements of L_b of lengths between n_i and n_{i+1}).

Observe that for any extensions L_1' and L_2' of L_1 and L_2 obtained in later stages, and for the corresponding self-algebrizing encodings A_1' and A_2', each of the first i machines will make the same mistakes solving inequality, even though the oracles have been modified. The reason is simple: the modifications of L_1' and L_2' are at the lengths that are beyond the reach of any such oracle machine, and the portions of the self-algebrizing encodings before the length n_{i+1} depend only on the portions of L_1' and L_2' before the length n_i, which stay the same.

Thus we are free to diagonalize against the oracle machine $M = M_{i+1}$ at some length $n = n_j$ for $j > i$. We will argue that there exist some strings X_1 and X_2 of lengths $N = 2^n$, so that if we extend L_1 and L_2 by setting their nth slices equal to X_1 and X_2, respectively, $M^{A_1'+A_2'}(1^n)$ is wrong on the inequality problem, where A_i' is the self-algebrizing encoding of the updated language L_b, for $b = 1, 2$. Clearly, this will conclude the proof of the lemma.

For contradiction, suppose no such strings X_1 and X_2 exist. Then M can be used to deterministically solve the inequality problem on all $N = 2^n$-bit strings X_1 and X_2, with only $2^{n/2}$-bit communication complexity. Indeed, let X_1 be given to Alice, and X_2 to Bob. They will simulate the machine $M^{A_1'+A_2'}$ on input 1^n (for A_b' the self-algebrizing encoding of $L_b \cup X_b$, for $b = 1, 2$). This machine queries either A_1' or A_2'. Alice can answer queries to A_1', since she knows X_1 (and the earlier part of L_1 , which is fixed for all inputs to Alice). Similarly, Bob can answer queries to A_2'. So the two players can simulate $M^{A_1'+A_2'}$ on input 1^n by communicating to each other the one-bit answers to the oracle queries made by M. Hence, the total communication complexity of this protocol is exactly the number of oracle queries made by M, which is at most $2^{n/2}$. This is a contradiction since the deterministic communication complexity for inequality on $N = 2^n$-bit strings is at least N. □

4.1.4 EXP ⊂ io-P/poly

COROLLARY 4.9. *There exist languages L_1 and L_2 with the corresponding self-algebrizing encodings A_1 and A_2 such that* $\mathsf{EXP}^{A_1+A_2} \subset \text{io-}\mathsf{P}^{A_1+A_2}/\mathsf{poly}$.

PROOF. Let L_1, L_2, A_1, A_2 be as in Lemma 4.8. Relative to $A_1 + A_2$, we have that $\mathsf{RP} \not\subseteq \mathsf{DTIME}(2^{o(n)})$. By relativizing the hardness-randomness tradeoff of [6], this implies that $\mathsf{EXP} \subset \text{io-}\mathsf{P}/\mathsf{poly}$, relative to the same oracle $A_1 + A_2$. □

4.1.5 Items (5)–(8) of Theorem 4.2

As in [1], the other items are proved using analogs of Lemma 4.6 for other complexity classes where we know communication-complexity separations; we skip the details.

We conclude this section by pointing out that although we are able to re-prove most of the results from [1] for our notion of algebrization, we do not know how to construct an ACT-consistent oracle O so that $\mathsf{NEXP}^O \subseteq \mathsf{P}^O/\mathsf{poly}$. We state the following much weaker result for the case where oracle access is restricted to polynomial-length queries only; however, such a restriction is very unsatisfactory, and it would be interesting to remove it.

THEOREM 4.10. *There is an oracle A consistent with ACT^* such that* $\mathsf{NEXP}^{A[\mathsf{poly}]} \subseteq \mathsf{P}^A/\mathsf{poly}$.

PROOF. Let L be an oracle such that $\mathsf{NEXP}^L \subseteq \mathsf{P}^L/\mathsf{poly}$ [30]. Let A be the self-algebrizing encoding of L. Then we have $\mathsf{NEXP}^{A[\mathsf{poly}]} \subseteq \mathsf{NEXP}^{\mathsf{PSPACE}^L[\mathsf{poly}]} \subseteq \mathsf{NEXP}^{L[\mathsf{poly}]} \subseteq \mathsf{P}^L/\mathsf{poly} \subseteq \mathsf{P}^A/\mathsf{poly}$. □

5. CONCLUSIONS

The theory $\mathcal{RCT}$ of [4] explains why relativizing techniques are insufficient for resolving the big open questions about the class P: these techniques have a very limited, "black-box" view of the class P. On the other hand, $\mathcal{LCT}$ knows everything there is to know about P (by having the Local Checkability axiom), and so everything that can be proved about P is provable in $\mathcal{LCT}$. However, it is one thing to have all the properties of P that one can use, and a completely different thing to know *how* to use them for proving theorems about P.

The arithmetization technique suggests one possible way to use local checkability, which has been quite fruitful in complexity theory. The theory $\mathcal{ACT}$ defined in the present paper seems like a useful intermediate theory for capturing some of what algebraic techniques add to relativizing complexity theory. Unlike the Aaronson-Wigderson [1] approach, it is clear that the provable consequences of $\mathcal{ACT}$ are closed under deduction. However, we do not have some of the results they could prove for their notion of algebrization; e.g., we do not know how to get an oracle O consistent with ACT so that $\mathsf{EXP}^O \subseteq \mathsf{P}^O/\mathsf{poly}$ (although we do have this inclusion infinitely often.) Also there are known results, proved using algebraic techniques, which do not follow from $\mathcal{ACT}$; e.g., $\mathcal{ACT}$ cannot prove NEXP = MIP. One way to interpret this is that, although the proof of NEXP = MIP certainly uses algebraic interpolation, it also uses other non-black-box arguments. Thus, while a statement failing to algebrize shows a broad range of techniques that will fail to resolve it, it certainly does not mean that it is beyond the scope of all current techniques in complexity. We should use algebrization as a tool for homing in on the correct proof techniques to solve open problems, not as an alibi for failing to solve them.

One way to make further progress is to use algebraic techniques in a non-algebrizing way. $\mathcal{ACT}$ treats the way we interpolate relations as polynomials as a black box. However, as observed in [2] (footnote, p. 46), the particular interpolant we choose often has other nice properties besides being low degree. In a standard application of arithmetization, one takes a small 3-cnf formula $\phi(x_1, \ldots, x_n)$ on m clauses $c_1, \ldots, c_m$, and produces its arithmetized version as the product of the polynomials $p_1, \ldots, p_m$, where each polynomial p_i is a multilinear polynomial that depends only on 3 variables (occurring in clause c_i), is Boolean-valued on Boolean inputs, and is 1 on a Boolean input iff that input

satisfies clause c_i. The distinguishing feature of this polynomial obtained from ϕ is that it can be completely *factored into 3-variate polynomials*. In contrast, polynomials we get for NP^O, with ACT-consistent oracles O, do not necessarily have this feature.

Can a BPP algorithm distinguish between a polynomial $p(x_1, \ldots, x_n)$ obtained by arithmetizing some 3cnf $\phi(x_1, \ldots, x_n)$ (as described above) and a random low-degree polynomial $q(x_1, \ldots, x_n)$, when given oracle access to the polynomials?[6] We observe that the answer is yes. The idea is that a BPP algorithm with oracle access to the polynomial p can learn p (and also ϕ) by *factoring* p. Namely, one can use the BPP algorithm of [19] to get a list of algorithms (each with oracle access to p) that compute all factors of p. Since each factor of p depends on at most 3 variables [7], we can learn a small arithmetic formula for each such factor (doing Polynomial Identity Tests to figure out which one is the right formula). Thus we can recover a small arithmetic formula for the entire polynomial p. In particular, this means that we can verify that p has small arithmetic complexity (and actually learn a small arithmetic formula for p). On the other hand, a random low-degree polynomial q is most likely of very high arithmetic circuit complexity, and so, when given oracle access to q, our algorithm will not be able to find any small arithmetic formula that computes q. Thus our BPP algorithm will distinguish between polynomials p and q.

Interestingly, the property of the polynomial p we have exploited in the above algorithm is very similar to the "locality of computation" property (Local Checkability) which was used as the basis for the theory $\mathcal{LCT}$: the reason p has a factorization into 3-variate polynomials is that we use a 3-cnf formula to describe a computation of a nondeterministic polynomial-time machine. So perhaps, the arithmetization technique can be pushed further, if we learn how to exploit this additional "locality" property of the polynomials obtained by arithmetization.

Another avenue to explore in future work are variants of $\mathcal{ACT}$ and their consequences. As mentioned before, interpolation of easily computable functions is possible over a large variety of algebraic structures, not just the integers. How do variants of $\mathcal{ACT}$ for different algebraic domains compare?

There are many questions one can ask about the power of $\mathcal{ACT}$. Here are just a few. Is there an ACT-consistent oracle O relative to which there exist one-way functions in P^O secure against BPP^O? Is there an ACT-consistent (nonempty) oracle relative to which $\mathsf{NEXP} = \mathsf{MIP}$? Can $\mathcal{ACT}$ prove the implication $\mathsf{EXP} \subset \mathsf{P/poly} \rightarrow \mathsf{EXP} = \mathsf{MA}$ [5], or the simpler implication $\mathsf{EXP} \subset \mathsf{P/poly} \rightarrow \mathsf{EXP} = \Sigma_2^p$ [21]?

Acknowledgments. We want to thank Scott Aaronson, Lance Fortnow, and the anonymous referees for their comments.

6. REFERENCES

[1] S. Aaronson and A. Wigderson. Algebrization: A new barrier in complexity theory. In *STOC*, pages 731–740, 2008.

[2] S. Aaronson and A. Wigderson. Algebrization: A new barrier in complexity theory. *ACM Trans. Comp. Theory*, 2008.

[6] This was an open question in [1], but has been resolved in [2], independently of our work.

[7] It is also easy to handle the case of k-cnf formulas on n variables for any $k \in O(\log n)$.

[3] S. Arora and B. Barak. *Complexity theory: a modern approach*. 2009.

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

[5] L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. *Comp. Complex.*, 1:3–40, 1991.

[6] L. Babai, L. Fortnow, N. Nisan, and A. Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Comp. Complex.*, 3:307–318, 1993.

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

[8] P. Beame, T. Pitassi, N. Segerlind, and A. Wigderson. A strong direct product theorem for corruption and the multiparty communication complexity of disjointness. *Comp. Complex.*, 15(4):391–432, 2006.

[9] H. Buhrman, L. Fortnow, and L. Thierauf. Nonrelativizing separations. In *CCC*, pages 8–12, 1998.

[10] A. Cobham. The intrinsic computational difficulty of functions. In Y. Bar-Hillel, editor, *Proc 1964 Int'l Congress for Logic, Methodology, and Philosophy of Science*, pages 24–30. 1964.

[11] M. Dekhtiar. On the impossibility of eliminating exhaustive search in computing a function relative to its graph. *Soviet Math. Dokl.*, 14:1146–1148, 1969.

[12] L. Fortnow. The role of relativization in complexity theory. *BEATCS*, 52:229–244, February 1994.

[13] L. Fortnow and M. Sipser. Are there interactive protocols for co-NP Languages? *IPL*, 28:249 –251, 1988.

[14] I. Gasarch and S. Homer. Relativizations comparing NP and EXP. *Inf. and Contr.*, 58:88–100, 1983.

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

[16] J. Håstad, R. Impagliazzo, L. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SICOMP*, 28:1364–1396, 1999.

[17] H. Heller. On relativized exponential and probabilistic complexity classes. *Inf. and Comp.*, 71(3):231 –243, 1986.

[18] R. Impagliazzo and A. Wigderson. P=BPP if E requires exponential circuits: Derandomizing the XOR Lemma. In *STOC*, pages 220–229, 1997.

[19] E. Kaltofen and B. Trager. Computing with polynomials given by black boxes for their evaluations: Greatest common divisors, factorization, separation of numerators and denominators. *JSC*, 9(3):301–320, 1990.

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

[21] R.M. Karp and R.J. Lipton. Turing machines that take advice. *L'Ens. Math.*, 28(3-4):191–209, 1982.

[22] E. Kushilevitz and N. Nisan. *Communication Complexity*. 1997.

[23] G. Lischke. Relationships between relativizations of P, NP, EL, NEL, EP and NEP. *Z. fur Math. Logik und Grundlagen der Math.*, 2:257 –270, 1986.

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

[25] M. Naor. Bit commitment using pseudorandomness. *J. Cryptology*, 4:151–158, 1991.

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

[27] A. Shamir. IP=PSPACE. *JACM*, 39(4):869–877, 1992.

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

[29] N.V. Vinodchandran. A note on the circuit complexity of PP. *TCS*, 347(1-2):415–418, 2005.

[30] C.B. Wilson. Relativized circuit complexity. *JCSS*, 31:169–181, 1985.