# An Axiomatic Approach to Algebrization

Russell Impagliazzo * Valentine Kabanets † Antonina Kolokolova ‡

January 21, 2009

**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 concerning "non-controversially relativizable" complexity classes was proved, mainly using algebraic techniques. Two approaches have been proposed to understand the power and limitations of these algebraic techniques: (1) Fortnow [For94] 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 "$\mathsf{P}$ vs. $\mathsf{NP}$" question differs between different oracles in that class. (2) Aaronson and Wigderson [AW08a] 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 [AIV92], we propose an *axiomatic* approach to "algebrization", which complements and clarifies the approaches of [For94] and [AW08a]. We add to the axiomatic characterization of the class $\mathsf{P}$ from [AIV92] a new axiom, *Arithmetic Checkability*, and argue that, in a certain sense, the resulting theory provides a characterization of "algebrizing" techniques. In particular, we show the following: *(i)* Fortnow's algebraic oracles from [For94] all satisfy Arithmetic Checkability (so arithmetic checkability holds relative to arbitrarily powerful oracles). *(ii)* Most of the algebrizing collapses and separations from [AW08a], such as $\mathsf{IP} = \mathsf{PSPACE}$, $\mathsf{NP} \subset \mathsf{ZKIP}$ if one-way functions exist, $\mathsf{MA\text{-}EXP} \not\subset \mathsf{P/poly}$, etc., are *provable* from Arithmetic Checkability. *(iii)* Many of the open complexity questions (shown to require non-algebrizing techniques in [AW08a]), such as "$\mathsf{P}$ vs. $\mathsf{NP}$", "$\mathsf{NP}$ vs. $\mathsf{BPP}$", etc., *cannot be proved* from Arithmetic Checkability. *(iv)* Arithmetic Checkability is also insufficient to prove one known result, $\mathsf{NEXP} = \mathsf{MIP}$ (although relative to an oracle satisfying Arithmetic Checkability, $\mathsf{NEXP}^O$ restricted to poly-length queries is contained in $\mathsf{MIP}^O$, mirroring a similar result from [AW08a]).

*U. C. San Diego and IAS; russell@cs.ucsd.edu

†Simon Fraser University; kabanets@cs.sfu.ca

‡Memorial University of Newfoundland; kol@cs.mun.ca

# 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 [BGS75] 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 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 [BGS75], NP vs. EXP [Dek69, GH83, Lis86], BPP vs. NEXP [Hel86], IP vs. PSPACE [FS88], 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 [AIV92]. There the authors introduced a theory $\mathcal{RCT}$ (based on Cobham's axiomatization of polynomial-time computation [Cob64]) and argued that, for any complexity statement $S$ about P, this statement $S$ is provable relative to every oracle $A$ (in some "Relativized Math" proof system) iff $S$ is provable in the system $\mathcal{RCT}$; more precisely, [AIV92] argue that the (standard) models of the theory $\mathcal{RCT}$ are exactly all relativized classes $\mathsf{P}^O$, for all possible oracles $O$. It follows that a non-relativizing statement is precisely a statement independent of $\mathcal{RCT}$, and thus, e.g., the "P vs. NP" question is independent of $\mathcal{RCT}$. [AIV92] also show that extending $\mathcal{RCT}$ with another axiom, which captures the "locality" of a Turing machine computation in the style of the Cook-Levin theorem, almost exactly characterizes the class P in the following sense: the models for the resulting theory, denoted by $\mathcal{LCT}$ (for Local Checkability) in [AIV92], are necessarily of the form $\mathsf{P}^O$ 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 concerning "non-controversially relativizable" complexity classes 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 [For94] 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 [AW08a] 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 [AIV92]. We extend their theory $\mathcal{RCT}$ with an axiom capturing the notion of arithmetization: the *Arithmetic Checkability* axiom. Intuitively, Arithmetic Checkability postulates that all $\mathsf{NP}$ languages have verifiers that are polynomial-time computable families of *low-degree polynomials*; a verifier for an $\mathsf{NP}$ 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 $\mathsf{P}^O$ with oracles $O$ such that Arithmetic Checkability holds relative to this $O$, i.e., all $\mathsf{NP}^O$ languages have $\mathsf{P}^O$-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 $\mathsf{P}^O$ for arbitrarily powerful oracles $O$ (in particular, any oracle from Fortnow's [For94] recursive construction). Thus, $\mathcal{ACT}$ is a theory that lies between the theories $\mathcal{RCT}$ and $\mathcal{LCT}$.

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 [AW08a]) are *provable in* $\mathcal{ACT}$, and that many open complexity questions (like the ones shown to non-algebrize in [AW08a]) are *independent of* $\mathcal{ACT}$. Since all provable consequences of $\mathcal{ACT}$ are closed under deduction, we avoid the limitations of the approach in [AW08a]. 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 [For94] all satisfy Arithmetic Checkability (so arithmetic checkability holds relative to arbitrarily powerful oracles). *(ii)* Most of the algebrizing collapses and separations from [AW08a], such as $\mathsf{IP} = \mathsf{PSPACE}$, $\mathsf{NP} \subset \mathsf{ZKIP}$ if one-way functions exist, $\mathsf{MA\text{-}EXP} \not\subset \mathsf{P/poly}$, etc., are *provable* from Arithmetic Checkability. *(iii)* Many of the open complexity questions (shown to require non-algebrizing techniques in [AW08a]), such as $\mathsf{P}$ vs. $\mathsf{NP}$, $\mathsf{P}$ vs. $\mathsf{BPP}$, 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, $\mathsf{NEXP} = \mathsf{MIP}$ (although relative to an oracle satisfying Arithmetic Checkability, $\mathsf{NEXP}^O$ restricted to poly-length queries is contained in $\mathsf{MIP}^O$, mirroring a similar result from [AW08a]).

**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

In this section, 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 [AIV92], 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 instead 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.[1] In this abstract, we will only consider the model-theoretic interpretation, to avoid a long discussion of the [AIV92] 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$^*$.

**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 $\mathsf{FP}$.

The class $\mathsf{ALG\text{-}PF}$ (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 $\mathsf{NP} = \mathsf{ALG\text{-}PF}$. 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 $f_{n+m}(x, g(x)) = 1$, *(ii)* if $x = x_1 \dots x_n \notin L$ then $f_{n+m}(x, g(x)) = 0$, and *(iii)* for all $y \in \{0,1\}^m$, $y \neq g(x) \implies f(x, y) = 0$. 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, i.e., $A[x_1, \dots, x_n] = \tilde{A}_{n+1}[0, x_1, \dots, x_n]$ for Boolean $x$, and such that $\tilde{A} \in \mathsf{P}^A$.

[1] 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.

We will relate arithmetic checkability to the notion of *local checkability* from [AIV92]. 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. There were a number of different versions in [AIV92]. In order to make the way to properly relativize this statement less controversial (although probably still not non-controversial), we use an intermediate-strength version that allows the verifier to make polynomial length queries to the oracle (but not to query the input).

**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 $\exists \Pi \in \{0,1\}^{p(|x|)} \forall r \in [1, \ldots, p(n)]\ V^{x,\Pi}(1^n, r) = 1$.

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 $\mathsf{NP}^O = \mathsf{PF\text{-}CHK}[poly, log]^O$.

[AIV92] and many others have observed that $\mathsf{NP} = \mathsf{PF\text{-}CHK}[poly, log]$ follows from the standard proof of the Cook-Levin Theorem in terms of tableaux. [AIV92] 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 $\mathsf{NP}^O$ reduces to unrelativized $\mathsf{NP}$ (via $\mathsf{P}^O$ 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 [For94].

**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 $(x, \Pi)$ 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 $\Pi$ 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 $p(x_1, \ldots, x_n, \Pi_1, \ldots, \Pi_m) = \Pi_r p_r(x_1, \ldots, x_n, \Pi_1, \ldots, \Pi_m)$. 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 [For94]. While we cannot prove that "being a consequence of ACT" characterizes algebrizing techniques in the sense of [AW08a], 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.[2]

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.

**Theorem 2.4** ([For94])**.** *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. First we present the construction due to Fortnow [For94].

*Fortnow's construction:* 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 $i$th bit of the binary representation of the integer value $f_n(c_1, \ldots, c_n)$ is one.

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

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 $b_1 \dots b_n \in \{0,1\}^n$, the value $A_n(b_1, \dots, b_n)$ 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 $\mathsf{PSPACE}^L$-complete language (e.g., the $\mathrm{TQBF}^L$ 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 inclusion is because $A \in \mathsf{PSPACE}^L$, and the second one because $A$ is $\mathsf{PSPACE}^L$-hard. Finally, observe that $L \in \mathsf{P}^A$ since $L \in \mathsf{PSPACE}^L \subseteq \mathsf{P}^A$. □

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 Arithmetic Checkability

First we show that the famous $\mathsf{PSPACE} = \mathsf{IP}$ theorem [LFKN92, Sha92] 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.* For any oracle $O$, the relativized version of TQBF is a complete problem for $\mathsf{PSPACE}^O$: Given an input $x_1 \dots x_n \in \{0,1\}^n$, decide if $\exists y_1 \in \{0,1\}\ \forall z_1 \in \{0,1\}\ \dots\ \exists y_m \in \{0,1\}\ \forall z_m \in \{0,1\}\ \ P(x_1, \dots, x_n, y_1, \dots, y_m, z_1, \dots, z_m)$, 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 $P(x_1, \dots, x_n, y_1, \dots, y_m, z_1, \dots, z_m)$ as $\exists w_1 \dots w_{m'} \in \{0,1\}^{m'}\ \ [f_{n+2m+m'}(x_1, \dots, x_n, y_1, \dots, y_m, z_1, \dots, z_m, w_1, \dots, w_{m'}) \neq 0]$, where $f$ is a polynomial family computable in $\mathsf{P}^O$.

Fix input $x$, merge the $w$'s with $y$'s and $z$'s, and consider $f^2$. The problem becomes to decide if $\exists y_1 \in \{0,1\}\ \forall z_1 \in \{0,1\}\ \dots\ \exists y_m \in \{0,1\}\ \forall z_m \in \{0,1\}\ \ [p_{2m}(y_1, \dots, y_m, z_1, \dots, z_m) \neq 0]$, where $p_{2m}$ is an efficiently computable polynomial that is always non-negative. We will follow the same protocol as in the standard proof of the $\mathsf{PSPACE} = \mathsf{IP}$ theorem (see, e.g., [AB09]).

For a polynomial $p(a_1, a_2, \dots, a_t)$ and variable $a_i$, define the multilinearization of $p$ with respect to $a_i$ by $p'(a_1, \dots, a_t) = a_i(p(a_1, \dots, a_{i-1}, 1, a_{i+1}, \dots, a_t) + (1 - a_i)p(a_1, \dots, a_{i-1}, 0, a_{i+1}, \dots, a_t)$. Note that $p'$ agrees with $p$ whenever $a_i$ is Boolean, and $p'$ is linear in $a_i$.

Define a system of polynomials $p_{i,j}[a_1, \dots, a_i]$, with $0 \leq j \leq i \leq 2m$, inductively as follows: $p_{2m,0} = p_{2m}$. For $j \neq 0$, $p_{i,2j-1}$ is the multilinearization of $p_{i,j-1}$ with respect to $y_j$, and similarly $p_{i,2j}$ with respect to $z_j$. Observe that

$$p_{2i,0}[y_1, z_1, \dots, y_i, z_i] = p_{2i+1,2i+1}[y_1, \dots, z_i, 0] + p_{2i+1,2i+1}[y_1, \dots, z_i, 1],$$

and

$$p_{2i+1,0}[y_1, z_1, \dots, y_i, z_i] = p_{2i+2,2i+2}[y_1, \dots, z_i, y_{i+1}, 0] \cdot p_{2i+1,2i+1}[y_1, \dots, z_{i+1}, 1].$$

Note also that for any Boolean values $y_1, \dots, z_i$, we have $p_{2i,j} \geq 0$, with non-equality if and only if the quantified expression with those values evaluates to true. The expression $p_{0,0}$ is a constant, which is non-zero if and only if $x \in L$. Finally, note that $p_{i,j}$ is of at most polynomial total degree. (In fact, each $p_{i,i}$ is multilinear, so each $p_{i,j}$ has degree at most 2 in each variable if $i < 2m$, and at most the degree of $p_{2m}$ otherwise.)

At the start of the protocol, the prover picks a prime $q$, so that $p_{0,0} \neq 0 \mod q$. Since the polynomial $p_{2m}$ is polynomial-time computable, its values cannot be too huge, and so the same is true for $p_{0,0}$, which is at most a $2^m$'th power of these values. (The value is thus at most doubly exponential, so has at most an exponential number of prime factors. The number of primes of a suitable polynomial length will be much larger than its number of factors.) So such a $q$ will exist. The verifier checks that $q$ is prime.

The prover also gives a value $v_0 \neq 0 \mod q$, and claims that $p_{0,0} = v_0 \mod q$. The rest of the protocol goes in stages $0 \leq j \leq i \leq 2m$. At each stage, there will be values $r_1, \ldots, r_i$ assigned to the first $i$ variables, and a value $v_{i,j}$, with the implicit claim that $p_{i,j}(r_1, \ldots, r_i) = v_{i,j}$. The property is that if the claim for $i, j$ is false, so will the claim be for $i, j+1$ if $j \neq i$, or for $i+1, 0$ if $j = i$.

In all cases, there is a simple formula relating $p_{i,j}$ to its successor polynomial ($p_{i,j-1}$ if $j \neq 0$, or $p_{i+1,i+1}$ if $j = 0$), with the successor polynomial only evaluated when a variable $a$ is replaced by constants. First, the prover sends a polynomial $g(a)$ that is supposed to be the value of the successor polynomial at an arbitrary value of $a$, and giving the corresponding values in $r_1, \ldots, r_i$ to the other variables. This will be a low-degree polynomial. The verifier checks that the defining formulas hold between $v_{i,j}$, $g(0)$ and $g(1)$. So if $g$ were the correct restricted successor polynomial, then $v_{i,j}$ would be the correct value of the original polynomial on the inputs. Conversely, if $v_{i,j}$ is incorrect, $g$ is incorrect, and only agrees with the correct restriction in very few places. The verifier then sends a random value $r$ for $a$, which replaces the value for $a$ in $r_1, \ldots, r_i$ if it exists. The successor value of $v$ is set to $g(r)$.

When we reach $i = 2m$, $j = 0$, the prover is claiming that $p_{2m,0}(r_1, \ldots, r_{2m}) = v_{2m,0} \mod q$. Since $p_{2m,0} = p_{2m}$, which is a restriction of a polynomially computable function, the verifier can test this claim in polynomial time, by evaluating $p_{2m}(r_1, \ldots, r_{2m})$ and then reducing the resulting value modulo $q$. If $x \in L$ and the prover follows the protocol, the claim is always true. Inductively, if $x \notin L$ and the verifier has not previously rejected, with high probability this is false. □

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 [BFT98, Vin05, San07], respectively; the last item is from [BFL91].

**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 [AW08a].

*(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 $\mathsf{IP}^O$-protocol for $\mathsf{PSPACE}^O$ is computable in $\mathsf{PSPACE}^O$, and hence, Merlin can give to Arthur a small circuit for this prover, and Arthur can simulate the $\mathsf{IP}^O$-protocol by interacting with the circuit.

If $\mathsf{PSPACE}^O \not\subset \mathsf{P}^O/\mathsf{poly}$, then also $\mathsf{MA\text{-}EXP}^O \not\subset \mathsf{P}^O/\mathsf{poly}$, 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 $\mathsf{NP}$-machine on a given input $x$ is reducible to the polynomial summation problem $\sum_{z_1,\ldots,z_m \in \{0,1\}} p(z_1, \ldots, z_n)$, where $p$ is a polynomial family computable in $\mathsf{FP}^{O^*}$. 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 $\sum_{y,w\in\{0,1\}^*} f(x, y, w)$.[3]

We can now use the LFKN protocol [LFKN92] 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 [Tod91], we have $(\Sigma_2^p)^{O^*} \subseteq \mathsf{P}^{\#\mathsf{P}^{O^*}}$. Finally, by relativizing Kannan's theorem [Kan82], 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 [AW08b], using the $\mathsf{PP}^{O^*}$-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 [AW08b].

*(4):* The proof of item (4) is also based on the proof of the corresponding non-relativized result [BFL91]. As pointed out in [AW08b], the tableau of a $\mathsf{NEXP}$-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 [BFL91]; see [AW08b] for details.

□

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 [AW08a]). 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 [GMW91] ($\mathsf{NP} \subseteq \mathsf{ZKIP}$ 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 [AW08a], but they have since independently obtained essentially the same result as we do below for their setting [AW08b]. (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 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 $\mathsf{P}^O$ secure against adversaries in $\mathsf{BPP}^O$. Then $\mathsf{NP}^O \subseteq \mathsf{ZKIP}^O$.*

*Proof.* If there are one-way functions, then there are semantically secure statistically binding commitment schemes ([Nao91, HILL99]; 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 $C(a)$.

[3] 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.

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 $\alpha_i$'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 $x = x_1 \ldots x_n \in L$ iff $\exists y_1, \ldots, y_m \in \{0,1\}^m \;\; f_{n+m}(x_1, \ldots, x_n, y_1, \ldots, y_m) \neq 0$. 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 $f(y_1, \ldots, y_m)$. We denote by $d$ the degree of $f$ (as a polynomial in the $y_i$'s).

First, the prover selects $\vec{y} = (y_1, \ldots, y_m)$ 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 $y_i$, $i = 1, \ldots, m$, and then to $(1 - y_i)$, $i = 1, \ldots, m$. 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 $\vec{s} = (s_1, \ldots, s_m) \in \mathbb{Z}_q^m$ and indirectly commits to the following values:

- each $s_j$, for $j = 1, \ldots, m$,
- each coordinate of $z(t) = \vec{y} + t\vec{s}$ for each $t = 1, 2, \ldots, d+1$,
- the value $v_t = f(z(t))$ for each $t = 0, 1, \ldots, d+1$, and
- the coefficients $c_0, \ldots c_d$ 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 $\mathsf{P}^O$. Using these values, the prover can also easily compute the coefficients $c_0, \ldots, c_d$ 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:

1. *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.

2. *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).

3. *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.

4. *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).

5. *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.

6. *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 $\mathsf{NEXP} \subseteq \mathsf{MIP}$ [BFL91]. 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) $\mathsf{EXP} \subseteq \mathsf{P/poly}$[4]; (3) $\mathsf{E} \subseteq \mathsf{MIP}$.*

*Proof. (1):* Let $L$ be a language complete for $\mathsf{PSPACE}$, and let $A$ be its self-algebrizing encoding (obtained using 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):* Take $L$ and $A$ as in the previous item. We get that $\mathsf{PSPACE}^A = \mathsf{P}^A$. By padding (which relativizes), we get $\mathsf{EXPSPACE}^A = \mathsf{EXP}^A$. By counting we know that $\mathsf{EXPSPACE}^A \not\subset \mathsf{P}^A/\mathsf{poly}$. Hence, $\mathsf{EXP}^A \not\subset \mathsf{P}^A/\mathsf{poly}$.

*(3):* Since both $\mathsf{PSPACE}^L$ and $\mathsf{MIP}^L$ 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 $\mathsf{E}^A \not\subseteq \mathsf{MIP}^A$. □

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

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* $\mathcal{ACT}^*$.

**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 [AW08a], 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., [KN97]). 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 [BPSW06] 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 [IW97, BFNW93] 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 $\mathsf{P}^A$-computable polynomial family for $A$, and let $p_0$ be a $\mathsf{P}^A$-computable polynomial family for $\bar{A}$ (the complement of $A$). Let

$g_0$ and $g_1$ be the corresponding $\mathsf{FP}^A$-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 $\mathsf{P}^A$-computable polynomial family showing that $L \in \mathsf{ALG\text{-}PF}^{*A}$, and a $\mathsf{FP}^A$-computable function $g$ mapping inputs to proofs.

Let $L \in \mathsf{P}^A$ be decided by a $\mathsf{P}^A$ 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 $1 \le i \le t$), 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 $\mathsf{FP}^A$. We let $g$ be the $\mathsf{FP}^A$-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 $\ell$ ones). Define $\tilde{p}_0(x_1, \dots, x_n, y_1, \dots, y_{m_1}) = p_0(x_1, \dots, x_n, y_1, \dots, y_{m_0}) \cdot \prod_{i=m_0+1}^{m_1} y_i$. 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 $2 \times 3$ "windows" of the tableau, of the polynomials expressing the correctness of the window). The second condition can also be expressed as the product, over $1 \le i \le t$, 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 $\mathsf{FP}^A$, showing that $L \in \mathsf{ALG\text{-}PF}^{*A}$. □

*Proof of Lemma 4.3.* By 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 $\mathsf{FP}^{A_0}$, and those for $A_1$ in $\mathsf{FP}^{A_1}$. 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 $x = x_1, \dots, x_n$ and $y = y_1, \dots, y_m$. 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 NP $\not\subseteq$ io-SIZE$(2^{n/4})$

We construct an oracle $L_1 + L_2$ so that, relative to the oracle $A_1 + A_2$ (for the corresponding self-algebrizing encodings of $L_1$ and $L_2$), we have a language in $\mathsf{NP} \cap \mathsf{E}$ with exponential circuit size complexity. This will allow us to prove item (1) of Theorem 4.2.

We'll use the following result from [BPSW06, Corollary 4.12, page 27].

**Theorem 4.5** ([BPSW06])**.** *There is a constant $c$ so that, if $N$ and $k$ are integers with $k \leq 2^{c\sqrt{N}}$, 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 $1 \leq i \leq k$, 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 $\alpha$, 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 BPP = P and P ≠ 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 $\mathsf{E}$. By relativizing [IW97], 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 $\mathsf{RP} \not\subseteq \mathsf{DTIME}(2^{o(n)})$

Next we prove the existence of an ACT-consistent oracle which separates RP from subexponential deterministic time.

**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 $\Delta$ 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 $n$th 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 the sake of 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 lower bound for inequality on $N = 2^n$-bit strings is at least $N$. □

#### 4.1.4 $\mathsf{EXP} \subset \mathsf{io\text{-}P/poly}$

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

*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 [BFNW93], this implies that $\mathsf{EXP} \subset \mathsf{io\text{-}P/poly}$, relative to the same oracle $A_1 + A_2$. □

#### 4.1.5 Items (5)–(8) of Theorem 4.2

As in [AW08a], 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 [AW08a] 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 $\mathsf{NEXP}^L \subseteq \mathsf{P}^L/\mathsf{poly}$ [Wil85]. 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

$\mathcal{ACT}$ seems like a useful intermediate theory for capturing some of what algebraic techniques add to relativizing complexity theory. Unlike the Aaronson-Wigderson [AW08a] 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 $\mathsf{NEXP} = \mathsf{MIP}$. One way to interpret this is that, although the proof of $\mathsf{NEXP} = \mathsf{MIP}$ 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 [AW08b] (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 $\phi$ is that it can be completely *factored into 3-variate polynomials*. In contrast, polynomials we get for $\mathsf{NP}^O$, with ACT-consistent oracles $O$, do not necessarily have this feature.

Can a $\mathsf{BPP}$ 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?[5] We observe that the answer is yes. The idea is that a $\mathsf{BPP}$ algorithm with oracle access to the polynomial $p$ can learn $p$ (and also $\phi$) by *factoring* $p$. Namely, one can use the $\mathsf{BPP}$ algorithm of [KT90] 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 [6], 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 $\mathsf{BPP}$ 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?

**Acknowledgments** We want to thank Scott Aaronson and Lance Fortnow for their comments on an early draft of this paper.

## References

[AB09] S. Arora and B. Barak. *Complexity theory: a modern approach*. Cambridge University Press, New York, 2009.

[AIV92] S. Arora, R. Impagliazzo, and U. Vazirani. Relativizing versus nonrelativizing techniques: The role of local checkability. Manuscript, 1992.

[AW08a] S. Aaronson and A. Wigderson. Algebrization: A new barrier in complexity theory. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, pages 731–740, 2008.

[AW08b] S. Aaronson and A. Wigderson. Algebrization: A new barrier in complexity theory. *ACM Transactions on Computation Theory*, 2008. (to appear).

[5] This was an open question in [AW08a], but has been resolved in [AW08b], independently of our work.

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

[BFL91] L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991.

[BFNW93] L. Babai, L. Fortnow, N. Nisan, and A. Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3:307–318, 1993.

[BFT98] H. Buhrman, L. Fortnow, and L. Thierauf. Nonrelativizing separations. In *Proceedings of the Thirteenth Annual IEEE Conference on Computational Complexity*, pages 8–12, 1998.

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

[BPSW06] P. Beame, T. Pitassi, N. Segerlind, and A. Wigderson. A strong direct product theorem for corruption and the multiparty communication complexity of disjointness. *Computational Complexity*, 15(4):391–432, 2006.

[Cob64] A. Cobham. The intrinsic computational difficulty of functions. In Y. Bar-Hillel, editor, *Proceedings of the 1964 International Congress for Logic, Methodology, and Philosophy of Science*, pages 24–30. North-Holland, Amsterdam, 1964.

[Dek69] M. Dekhtiar. On the impossibility of eliminating exhaustive search in computing a function relative to its graph. *DAN SSSR = Soviet Math. Dokl.*, 14:1146–1148, 1969.

[For94] L. Fortnow. The role of relativization in complexity theory. *Bulletin of the European Association for Theoretical Computer Science*, 52:229–244, February 1994. Columns: Structural Complexity.

[FS88] L. Fortnow and M. Sipser. Are there interactive protocols for co-NP Languages? *Information Processing Letters*, 28:249 –251, 1988.

[GH83] I. Gasarch and S. Homer. Relativizations comparing NP and EXP. *Information and Control*, 58:88–100, 1983.

[GMW91] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the Association for Computing Machinery*, 38:691–729, 1991.

[Hel86] H. Heller. On relativized exponential and probabilistic complexity classes. *Information and Computation*, 71(3):231 –243, 1986.

[HILL99] J. Håstad, R. Impagliazzo, L. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28:1364–1396, 1999.

[IW97] R. Impagliazzo and A. Wigderson. P=BPP if E requires exponential circuits: Derandomizing the XOR Lemma. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 220–229, 1997.

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

[KN97] E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, New York, 1997.

[KT90] E. Kaltofen and B. Trager. Computing with polynomials given by black boxes for their evaluations: Greatest common divisors, factorization, separation of numerators and denominators. *Journal of Symbolic Computation*, 9(3):301–320, 1990.

[LFKN92] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. *Journal of the Association for Computing Machinery*, 39(4):859–868, 1992.

[Lis86] G. Lischke. Relationships between relativizations of $\mathsf{P}, \mathsf{NP}, \mathrm{EL}$, NEL, EP and NEP. *Zeitschrift fur Mathematische Logik und Grundlagen der Mathematik*, 2:257 –270, 1986.

[Nao91] M. Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4:151–158, 1991.

[San07] R. Santhanam. Circuit lower bounds for Merlin-Arthur classes. In *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*, pages 275–283, 2007.

[Sha92] A. Shamir. IP=PSPACE. *Journal of the Association for Computing Machinery*, 39(4):869–877, 1992.

[Tod91] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991.

[Vin05] N.V. Vinodchandran. A note on the circuit complexity of PP. *Theoretical Computer Science*, 347(1-2):415–418, 2005.

[Wil85] C.B. Wilson. Relativized circuit complexity. *Journal of Computer and System Sciences*, 31:169–181, 1985.