Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 40 (2016)

Affine Relativization: Unifying the Algebrization and Relativization Barriers

Barış Aydınlıoğlu [*] Eric Bach [†]

June 9, 2016

Abstract

We strengthen existing evidence for the so-called "algebrization barrier". Algebrization — short for algebraic relativization — was introduced by Aaronson and Wigderson (AW) (STOC 2008) in order to characterize proofs involving arithmetization, simulation, and other "current techniques". However, unlike relativization, eligible statements under this notion do not seem to have basic closure properties, making it conceivable to take two proofs, both with algebrizing conclusions, and combine them to get a proof without. Further, the notion is undefined for most types of statements, and does not seem to yield a general criterion by which we can tell, given a proof, whether it algebrizes. In fact the very notion of an algebrizing proof is never made explicit, and casual attempts to define it are problematic. All these issues raise the question of what evidence, if any, is obtained by knowing whether some statement does or does not algebrize.

We reformulate algebrization to handle these shortcomings. We first define a statement as *relativizing* if, intuitively, it is insensitive to the choice of a Boolean basis, and then as *relativizing affinely* if, roughly, it relativizes with respect to every affine extension — here an affine extension is the result of a particular error correcting code applied to the characteristic string of a language. We also define the notion of a *proof* to relativize (affinely), while ensuring closure under inference. We show that all statements that AW declare as algebrizing can be derived via an affinely relativizing proof, and that no such proof exists for any of the statements shown not-algebrizing by AW in the classical computation model.

Our work complements, and goes beyond, the subsequent work by Impagliazzo, Kabanets, and Kolokolova (STOC 2009), which also proposes a reformulation of algebrization, but falls short of recovering some key results of AW, most notably regarding the NEXP versus P/poly question.

One consequence of our definitions is a demystified perspective on the extent to which relativizing techniques view computation as a "black box" and current uses of arithmetization do not. As another consequence, we give new streamlined proofs of several classic results in complexity, including $\mathrm{PSPACE} \subset \mathrm{IP}$ and $\mathrm{NEXP} \subset \mathrm{MIP}$.

[*]University of Wisconsin-Madison; `baris@cs.wisc.edu`.
[†]University of Wisconsin-Madison; `bach@cs.wisc.edu`

ISSN 1433-8092

Contents

1 Introduction

Motivation. The algebrization notion — short for algebraic relativization — was put forth by Aaronson and Wigderson [1] (AW henceforth) to give evidence that certain complexity-theoretic conjectures are beyond the reach of "current proof techniques". Although the name suggests some type of relativization, algebrization lacks two essential properties of relativization:

Closure under inference. What exactly constitutes a "current technique" may be inherently unclear, but at a minimum it seems logical inference rules should be included. However, as pointed out in [1, 29, 22], statements that algebrize in the AW formulation are not known to be closed under inference.

For example, AW show that the statement $\psi := \mathrm{NEXP} \not\subset \mathrm{P/poly}$ does not algebrize, and interpret this to mean that a certain class of proof techniques, say "algebrizing techniques", cannot prove ψ. Yet, this does not rule out an approach where, say, one comes up with a class $\mathcal{C}$ and, shows $\mathcal{C} \subset \mathrm{NEXP}$ via algebrizing techniques, then shows $\mathcal{C} \not\subset \mathrm{P/poly}$ via algebrizing techniques, and thus derive the very same ψ.

Lack of closure under inference thus significantly thins any evidence imparted by a negative algebrization result — as AW obtained for NEXP versus $\mathrm{P/poly}$ and for other questions of structural complexity — since the class of proofs ruled out by such a result might be much smaller than intended.

This precludes algebrization from having one of the two key virtues of relativization, namely delineating those conjectures within possible reach of a robust family of techniques, from those that are not. Indeed, some major results in complexity are suggested to have been found using relativization as such a guide [6, 17].

Universality. A main appeal of relativization is being a universal notion, in the sense that it applies to every statement in one generic way. Intuitively, a statement relativizes if its truth is insensitive to broadening the definition of computer, from an ordinary Turing Machine, to one with oracle access to an arbitrary language $\mathcal{O}$. (We provide an alternate intuition later in Section 1.1.)

This intuition is so natural that it enables the second key virtue of relativization, namely being a "litmus test" for weeding out futile endeavours. The idea is that if ψ is already known to not relativize, then any strategy for proving ψ, in order to be viable, must somehow be unable to handle arbitrary extensions of the computer notion, or else it would be a strategy for proving not just ψ, but that ψ relativizes. Given the scarcity of such proof strategies in structural complexity — at least for those ψ involving P-based classes — this idea makes relativization a practical tool for guiding research. (Alas, we do not have a count on the number of fruitless research hours saved this way.)

For algebrization, however, we have no comparable intuition. This is mainly because algebrization is a selective notion, in the sense that it is defined only for containments $\mathcal{C} \subset \mathcal{D}$ and separations $\mathcal{C} \not\subset \mathcal{D}$, and moreover, it is applied differently to each side of the containment / separation. Supposing we have a strategy to prove ψ — and assuming, to begin with, ψ is of compatible syntax — there is no universal criterion we can apply, to check if our ideas can be extended to show that ψ algebrizes. This calls into question how relevant it is to know that ψ is non-algebrizing in the first place.

Besides the above problems, algebrization brings back some longstanding ones that are as old as the relativization notion itself:

Controversial relativizations. A pair of theorems might be derived using seemingly same techniques, yet only one might be relativizing / algebrizing. For example, $\mathrm{PSPACE} \subset \mathrm{IP}$, as AW show, algebrizes, yet its cousin, $\mathrm{NEXP} \subset \mathrm{MIP}$, does *not*, as observed by Impagliazzo, Kabanets, and Kolokolova [22] — except it *does*, as AW show, if we restrict oracle access for NEXP to be of polynomial-length.

It is not clear how to interpret such results without further work. Can we justify restricting oracle access, say by showing that it yields a natural subclass not tied to the Turing machine model? If so, then which "current technique" eliminates the difference between the two classes, the subclass and the original, thereby overcoming the limits of algebrizing techniques (whatever they are)?

Relativizing statements vs. proofs. A generally accepted (though not uncontested [25]) convention is to remark that some proof, say of ψ, relativizes or algebrizes, with no clear consensus on what that exactly means.

The typical intent behind such remarks seems to be that the said proof can be transformed into a proof *that ψ relativizes* (or algebrizes). However, as anything can be transformed into anything when there is no constraint, it is not clear which proofs do *not* relativize under such a definition. And even if some commonsense transformations are tacitly agreed upon — e.g., "give every Turing machine an oracle for $\mathcal{O}$," or "bring each statement to its relativized form" — it is unclear whether the transformed object would always be a valid proof, let alone a valid proof that ψ relativizes.

Naturally thus the question arises, of whether precise definitions can be given for what constitutes a relativizing / algebrizing statement / proof. Ideally, the definitions should capture the everyday intuition for these notions, as well as test the folkloric belief that relativizing techniques (whatever they are) view computation as a "black box", and relate that to algebrizing techniques.

Prior Work. Although an early draft by Arora, Impagliazzo, and Vazirani [3] (AIV) succeeds in giving a precise definition of a relativizing proof, it is not clear if their approach captures the everyday intuitions for the concept. For one thing, no distinction is made there between a statement ψ to relativize, versus ψ to have a relativizing *proof*, although such a distinction seems essential to capture the everyday uses; see Section 1.2 for more explanation. Further, the approach is recursion theoretic and makes no reference to computational devices such as circuits or Turing machines, and consequently it is difficult to tell, for example, that "Satisfiability is NP-complete" relativizes in their framework, or has such a proof, even though "oracle gates" for $\mathcal{O}$ can be easily incorporated into a circuit / formula. See Section 1.2 for more discussion.

Building on the AIV approach to relativization, Impagliazzo, Kabanets, and Kolokolova [22] develop an analogous approach for algebrization. However, that approach falls short of recovering some key results of AW, most notably regarding the NEXP versus $\mathrm{P/poly}$ question. Again, see Section 1.2 for details.

Our Results. In this paper, we reformulate relativization and algebrization, in a way that addresses all the problems raised in the first section.

We give a simple definition of what it means for a statement / proof to relativize, that yields the following intuition: a statement / proof relativizes iff it is insensitive to enlarging the standard Boolean basis. Our work delineates the notions of a statement ψ to relativize versus a *proof* of ψ to relativize (as well as other notions in between). As we argue in Section 1.2, this distinction is essential if we want to model the intuition behind the casual uses of these terms.

Our main contribution is to the algebrization notion. We define a statement / proof as relativizing *affinely* if, intuitively, it is insensitive to enlarging the standard Boolean basis *with any affine extension* — here affine extension is the result of a particular error correcting code applied to the characteristic string of a language. With this definition, we show that every statement that AW declare as relativizing algebraically does relativize affinely — in fact has a *proof* that relativizes affinely — and that the opposite holds for statements declared non-algebrizing by AW in the classical model.[1] (Both require new ideas.) Our formulation in this sense gives rigorous support to the "algebrization barrier" idea of AW, which can thus be viewed as a refinement of the classic "relativization barrier" of Baker, Gill, and Solovay [10].

[1] AW state some non-algebrization results for quantum-based complexity classes as well; we do not pursue these.

Affine relativization is a refinement of relativization so as to capture the known uses of *arithmetization*, a technique for interpolating Boolean formulas into polynomials. Famously used in early 90's for obtaining $\mathrm{PSPACE} \subset \mathrm{IP}$ and related results, which are false relative to some choices of an oracle $\mathcal{O}$ [20, 19, 12], arithmetization is widely regarded as a counterexample — maybe *the* counterexample — to the rule-of-thumb that "most known proof techniques relativize" in structural complexity theory. Affine relativization, to the extent that it captures the known uses of arithmetization — and it does so fairly well, as we argue in the rest of this section — can be viewed as a step towards reinstating that rule-of-thumb (albeit only a step, as the PCP theorem is out of scope of this and related work; see open question in Section 5).

As one conceptual consequence, our formulations yield a demystified perspective on the extent to which relativizing techniques are "black-box" and arithmetization-based techniques are not; see Section 5.

Our formulations also tell something about those "few known proof techniques" that do not seem to relativize affinely, in particular, about *locality of computation*. It is a longstanding debate whether locality — that the next step of a computation depends on only a "small" fragment of its current state — plays any role in current results of complexity, particularly in interactive proofs [18, 3, 17, 22]. On one hand, $\mathrm{NEXP} \subset \mathrm{MIP}$ can be explained away as relativizing algebraically with a convenient, but questionable, alteration of the oracle access mechanism as mentioned above; on the other hand, locality could provide an honest explanation of this theorem, as argued by Arora, Impagliazzo, and Vazirani [3], but an incongruent one to its algebraic nature, especially when its cousin, $\mathrm{PSPACE} \subset \mathrm{IP}$, needs no such explanation.

Our results shed some light onto this matter. As we explain in Section 1.3, it is fruitful to put a particular class between PSPACE and IP, and another one between NEXP and MIP, so that each theorem reads as two containments. The second containment, we argue, captures the real content in each theorem, namely "gap amplification"; affine relativization can derive every containment except the first one for NEXP versus MIP. We conclude that whether or not $\mathrm{NEXP} \subset \mathrm{MIP}$ algebrizes is just a matter of definition, because there is no application of this theorem (as far as we know) that is sensitive to how it is viewed, gap amplification versus the common view. Therefore affine relativization can be viewed as a robust proxy, or a candidate thereof, for the current state of the art.

This is mere interpretation, however, and is not to be confused with the main message of the paper:

Summary of Results. *Affinely relativizing proofs, as defined in Section 1.1, have the following properties.*

- *Each of the following has an affinely relativizing proof*
  - $\mathrm{PSPACE} \subset \mathrm{IP}$*, viewed as gap amplification* *(Corollary 21)*
  - $\mathrm{NEXP} \subset \mathrm{MIP}$*, viewed as gap amplification* *(Theorem 23)*
  - $\mathrm{MA}_{\mathrm{EXP}} \not\subset \mathrm{SIZE}(2^{\log^d n}), \forall d$ *(Theorem 30)*
  - $\mathrm{prMA} \not\subset \mathrm{SIZE}(n^d), \forall d$ *(Theorem 30)*
  - $\mathrm{NP} \subset \mathrm{ZKIP}$ *if one-way-functions exist* *(Theorem 35)*
- *None of the following has an affinely relativizing proof*
  - $\mathrm{NP} \not\subset \mathrm{P}$*, in fact* $\mathrm{PSPACE} \not\subset \mathrm{P}$ *(Proposition 36)*
  - $\mathrm{NP} \subset \mathrm{P}$*, in fact* $\mathrm{RP} \subset \mathrm{SUBEXP}$ *(Corollary 44)*
  - $\mathrm{NP} \subset \mathrm{BPP}$*, in fact* $\mathrm{coNP} \subset \mathrm{MA}$ *(Corollary 43)*
  - $\mathrm{P}^{\mathrm{NP}} \subset \mathrm{PP}$ *(Corollary 43)*
  - $\mathrm{NEXP} \not\subset \mathrm{P/poly}$*, in fact* $\mathrm{NEXP} \not\subset \mathrm{SIZE}(n^d), \forall d$ *(Theorem 39)*

Further, affinely relativizing proofs are closed under inference, and if a statement has an affinely relativizing proof, then it "affinely relativizes," i.e., it holds relative to each language that is an affine extension, as defined in Section 1.1.

Organization. We describe our formulation next in Section 1.1 and compare it to prior work in Section 1.2. We give an overview of the ideas and techniques in Section 1.3. The technical development starts from Section 2 and is self-contained from thereon.

1.1 Relativization and Affine Relativization

We now describe our formulation of the relativization and affine relativization notion.

Relativization without oracles. Let the standard Boolean basis refer to the set $\{0, 1, \wedge, \oplus\}$ comprising four languages (with 0 denoting the empty language, viewed as the function mapping all binary strings to zero, 1 its negation, and with $\wedge$,$\oplus$ denoting the AND,XOR function on binary strings respectively). We say that the statement ψ holds relative to the language $\mathcal{O}$ iff ψ is true when the standard Boolean basis is extended with $\mathcal{O}$. We say ψ relativizes to mean that ψ holds relative to every $\mathcal{O}$.

Some remarks are in order.

- Let us momentarily be more precise. We will work in the first order language with signature $\{\in, B_{std}\}$. The symbols $0, 1, \wedge, \oplus$ are introduced with definitions, and are not officially part of the signature. (Similar to the idea that first order number theory doesn't have any numerals besides 0; other numerals are abbreviations for $S(0)$, $SS(0)$, etc.)

  Take the axioms of everyday set theory, say ZFC, and add two new axioms: (i) that B_{std} includes $\{0, 1, \wedge, \oplus\}$ (i.e., $0 \in B_{std}$ and $1 \in B_{std}$ and so on), (ii) that B_{std} is included in $\{0, 1, \wedge, \oplus, \mathcal{O}\}$ for some language $\mathcal{O}$. Name the new collection of axioms $\mathcal{RCT}$, for relativized complexity theory.

  Now take $\mathcal{RCT}$ and add the axiom: (iii) B_{std} is included in $\{0, 1, \wedge, \oplus\}$. Call this set of axioms $\mathcal{CT}$, for real-world complexity theory. (Note (iii) implies (ii) as $\mathcal{O}$ need not be a distinct fifth element.)

  Given a statement ψ (in the language with signature $\{\in, B_{std}\}$), we call ψ relativizing iff it is true in every *standard* model of $\mathcal{RCT}$. Assuming, as we may here and throughout the paper, that everyday mathematics is consistent, a *standard model of set theory* is one where the symbol '$\in$' for set membership is interpreted as the actual "is an element of" relation.

- ψ being nonrelativizing per se does not make it interesting or hard to prove; e.g. let ψ be "the standard Boolean basis is $\{0, 1, \wedge, \oplus\}$".

  Conversely, relativizing statements can be nontrivial; in fact *every* true statement that does not mention the standard Boolean basis (i.e., every ψ over the signature $\{\in\}$ that holds in the standard models of ZFC) is by definition relativizing.[2]

- Relativizing statements are closed under inference, since if ϕ_{n+1} is a consequence of $\phi_1, .., \phi_n$, then ϕ_{n+1} is true whenever $\phi_1, .., \phi_n$ are, which makes ϕ_{n+1} relativizing assuming $\phi_1, .., \phi_n$ already are.

We agree to take the uniform-circuit-based definition of P, and P-based definitions of NP, NEXP, etc., so that extending the standard Boolean basis with $\mathcal{O}$ automatically gives us $\mathrm{P}^{\mathcal{O}}$, $\mathrm{NP}^{\mathcal{O}}$, etc., without having to mention oracle access to $\mathcal{O}$ at all — though we do mention it anyway, for emphasis. An algorithm thus means to us a uniform family of circuits. So if we "let V be a time-$t(n)$ algorithm with oracle access to a proof string π", for example, then unless otherwise stated, we mean to "let $V := \{V_n\}$ be a uniform circuit family of size $t(n)$ (or of size $t(n)\,\mathrm{polylog}\,t(n)$ — does not matter in this paper) over the basis $B_{std} \cup \pi$" for some language π defined appropriate to the context.

The uniformity in the above paragraph can be specified using any notion of classical computer running in polynomial-time, be it Turing machines or pointer machines or anything else; we pick Turing machines

[2]this agrees with intuition: if ψ does not mention a Boolean basis, then its truth is insensitive to the choice of such basis.

to get the following fact for free: for every $L \in \mathrm{P}$, there is $f_L \in \mathrm{FP}$ describing a circuit family $C := \{C_n\}$ for L, say via the map

$$(1^n, 1^i) \mapsto \text{the type of the } i^{\text{th}} \text{ gate in } C_n \text{ and the indices of all gates connected to the } i^{\text{th}} \text{ gate.}$$

We caution that we do not exploit any peculiarity in this way of defining P. The reader who prefers the Turing machine model can stick to it, provided the Turing machine is defined to have oracle access to every element of the standard Boolean basis, so that the above definitions make sense.

It is out of the scope of this paper whether our framework can handle classes "below" P, or those classes not definable from P. (And it is an observation of this paper that NEXP with poly-length oracle queries *can* be defined from P, as 0-gap-MIP. Similarly PSPACE can be defined as 0-gap-IP. See Section 1.3.)

ψ relativizes vs. ψ has a proof that relativizes. Given a proof Π of ψ — i.e., a sequence $\phi_1..\phi_n$ where $\phi_n = \psi$, and each ϕ_i is either taken for granted or is a consequence of $\phi_1, .., \phi_{i-1}$ — we call Π relativizing iff after extending the standard Boolean basis with an arbitrary language $\mathcal{O}$, it remains a proof of ψ.

Further remarks are in order.

- Let us again be more precise for a moment. Take any Hilbert-style system of proof, e.g., the one in [16, section 2.4]. Let ψ be a statement in the language with signature $\{\in, B_{std}\}$. We call a proof of ψ (from $\mathcal{CT}$) relativizing iff it is a proof from $\mathcal{RCT}$.
- ψ being relativizing per se does not mean ψ has a relativizing proof; e.g. let ψ be "ZFC is consistent".
- Relativizing proofs are closed under inference. (This is immediate from the precise definition, since they are exactly those proofs derivable from a certain set of axioms.) This is because if ϕ_{n+1} is a consequence of $\phi_1, .., \phi_n$, then it remains so no matter what, in particular no matter how the standard Boolean basis is extended, which makes $\phi_1..\phi_n\phi_{n+1}$ a relativizing proof assuming $\phi_1..\phi_n$ already is.[3]
- A relativizing proof of ψ yields a proof that ψ *and* that ψ relativizes. (Precisely speaking this is by the soundness theorem for first order logic). The converse is not clear; see open question in Section 5.

We emphasize that relativization of statements is a *semantic* concept, and of proofs is a *syntactic* one.

In Section 3, many theorems we derive are of the form "ψ, and ψ (affinely) relativizes", or something to that effect. As remarked just above, it is not clear if such a theorem implies, by itself, that ψ has a relativizing proof. Nonetheless, in the process of deriving each of these theorems we end up giving a relativizing proof of the corresponding ψ. (Of course we do not provide a formal proof of ψ in first order logic, just as we do not specify algorithms by implementing them as Boolean circuits.)

So those theorems in Section 3 should really be read as "ψ has an (affinely) relativizing proof" for some ψ. To be convinced of such claims, the salient point to be checked in their proofs is whether they account for the standard Boolean basis to be extended with some arbitrary (affine) language $\mathcal{O}$ — and they do.

Affine relativization. Take $\mathcal{RCT}$ and add the axiom: B_{std} is included in $\{0, 1, \wedge, \oplus, \mathcal{O}\}$ for some affine extension $\mathcal{O}$. That is, there is a language f, with f_n denoting its restriction to length-n inputs, and with $\widehat{f_n}$ denoting the unique n-variate polynomial of individual degree-≤ 1 extending f_n, such that $\mathcal{O}$ represents the evaluation of $\widehat{f_n}$ over $\mathrm{GF}(2^k)$, for all k and n. (See Section 2 for a precise definition, and Section 1.3 for a motivation.)

Call the resulting set $\mathcal{ACT}$, for affinely relativized complexity theory. Define the notion of a statement / proof affinely relativizing similarly to $\mathcal{RCT}$. (Precisely, a statement ψ is affinely relativizing iff it holds in every standard model of $\mathcal{ACT}$, and a proof of ψ is affinely relativizing iff it is a proof from $\mathcal{ACT}$.) It follows just as in the case for $\mathcal{RCT}$ above that affinely relativizing statements / proofs are closed under inference.

[3]A relativizing proof can in one step be turned into a proof that is non-relativizing — but not via an inference step.

The empty language is the affine extension of itself. Thus it does not make any difference to add the same axiom, that $\mathcal{O}$ is an affine extension, to $\mathcal{CT}$ as well. Now we have three theories $\mathcal{RCT} \subset \mathcal{ACT} \subset \mathcal{CT}$, each strictly more powerful than the one before, as the results in this paper imply.

Multiple oracles. If ψ is (affinely) relativizing, or has such a proof, then what happens if we want to extend the standard Boolean basis twice — say with $\mathcal{O}_0$ and $\mathcal{O}_1$? For plain relativization the answer is easy; just set $\mathcal{O}$ to be their disjoint union, $\mathcal{O}_0 \coprod \mathcal{O}_1 : bx \mapsto \mathcal{O}_b(x)$, and proceed as before.

For affine relativization, however, a bit more care is needed since we want $\mathcal{O}$ to be an affine extension. If $\mathcal{O}_0$ is the affine extension of L_0, and $\mathcal{O}_1$ of L_1, the key observation is that the disjoint union $\mathcal{O}_0 \coprod \mathcal{O}_1$ of the affine extensions is equivalent, under Cook reductions, to the affine extension of the disjoint union $L_0 \coprod L_1$. This is spelled out in Proposition 42, but intuitively is true because the disjoint union merely adds an extra dimension — the "b-axis" — and the affine extension acts on each dimension independently (see equation (†) on page 10). So we set $\mathcal{O}$ to the affine extension of $L_0 \coprod L_1$ and proceed as before, the upshot being that one affine oracle is just as good as k of them for $k > 1$.

1.2 Comparison with Prior Work

Four past works have a direct relation to ours. The main effort in all of them, and in ours, can be viewed as trying to: (i) formalize the relativization notion, and / or (ii) refine the notion so as to capture $\mathrm{PSPACE} \subset \mathrm{IP}$ and related results. We now do a comparison with past work, first with respect to (i) and then (ii).

1.2.1 Efforts to Formalize Relativization

In a widely-known manuscript, **Arora, Impagliazzo, Vazirani** [3] (AIV henceforth) build on Cobham's axioms for polynomial-time computation [14] to define $\mathcal{RCT}$, relativized complexity theory, and argue that derivations from $\mathcal{RCT}$ can be translated to and from relativizing proofs in our sense (which they refer to informally, as proofs in a "relativized normal math" system).

A common feature — or flaw, if the reader is logically inclined — of both our definition of $\mathcal{RCT}$ and AIV's is that the axioms for capturing relativization go on top of an existing collection of axioms governing everyday mathematics. On one hand, this is a feature because relativization is meant to be a guide for the everyday researcher, who has everyday mathematics at disposal. On the other hand, this is a flaw because statements such as "P versus NP is independent of $\mathcal{RCT}$" can be easily misunderstood, as the so-called independence concerns only *one* natural way of defining P, NP out of at least two — another one being to just ignore the extra axioms (they do not interfere with everyday math). This is inevitable unless one *removes* axioms from mathematics and not add to it, and the quest then becomes to find the "weakest" version of math that can prove statements such as $\mathrm{PSPACE} \subset \mathrm{IP}$, as opposed to finding, like we and AIV essentially set out to do, for the "strongest" version of these statements that can be proven by everyday math.

One difference of our version of $\mathcal{RCT}$ from AIV's is its accessibility. While AIV's approach à la Cobham [14] gives an elegant axiomatization of relativized P (with only a dozen or so axioms), it also avoids devices such as circuits, Turing machines, etc., making it difficult for a casual user to tell whether s/he is really working with their proposed definition. Our approach, in contrast, is "naive" in the sense that it does not attempt at a minimal set of axioms (nor does it spell out every axiom) but in return, it gives a formalism that is arguably closer to the everyday uses of relativization — e.g., unlike in AIV's approach, "Satisfiability is NP-complete" is easily seen to have a relativizing proof in our framework.[4]

[4]One can derive the same result in the AIV framework by first showing that their definition is equivalent to a device-based definition, but that would amount to not using their definition.

Our work identifies a distinction between a *statement* ψ to relativize and a *proof* of ψ to relativize. By mapping the former notion to a semantic concept, and the latter to a syntactic one, our formulation yields, as a byproduct, that

(i) ψ is a statement that can be proven to relativize, and

(ii) ψ is a relativizing statement that can be proven,

are distinct notions, as is their combination (i)&(ii); none are known to be equivalent, some are not equivalent, and all are implied by

(iii) ψ has a relativizing proof.

In contrast, AIV use all of (i),(ii),(iii) synonymously in [3].[5]

We consider this separate treatment, of statements and proofs, essential in formalizing the casual uses of relativization. Indeed, in the casual sense, ψ relativizes if it is true no matter what oracle $\mathcal{O}$ we incorporate into the definition of a Turing machine — meaning if ψ is true and does not mention any Turing machines, then it should relativize automatically. So "$1+1=2$" should relativize, as should "ZFC is consistent". But this last statement dashes any hope of formalizing relativization solely with proofs.

It is tempting to get around this issue by saying that $1+1=2$ is an "uninteresting" statement. Indeed, in their sequel to the AIV paper, Impagliazzo, Kabanets, and Kolokolova [22] can be interpreted as taking this position when they claim "a complexity statement about P" relativizes iff it is provable from AIV's $\mathcal{RCT}$. Notice, however, that it is unclear what such a statement could be[6]. Besides, it disagrees with everyday usage to say, e.g., that whether $\mathcal{A} \subset \mathcal{B} \ \wedge\ \mathcal{B} \subset \mathcal{C} \implies \mathcal{A} \subset \mathcal{C}$ relativizes depends on what $\mathcal{A}, \mathcal{B}, \mathcal{C}$ are.

Our interpretation of the everyday usage of relativization for a statement ψ is this: – that ψ does relativize as a casual claim is really a claim that ψ has a relativizing proof, – that ψ does not relativize as a casual claim is really a claim that ψ does not relativize. Notice that the two cases sound complementary in casual use, but when made precise (under our interpretation) they are merely mutually exclusive: the former is syntactic and the latter semantic, justifying once again our separate treatment of statements and proofs.

1.2.2 Efforts to Refine Relativization

Although relativization succeeds at explaining the failures of structural complexity until the 90s, it fails at explaining the successes after, especially those regarding interactive proofs. We now discuss four past proposals to refine relativization. The overarching goal in them is (or so will be our view here) to provide some model for "known techniques", which involves meeting two competing objectives: (a) derive all relevant theorems in the model, and (b) provably fail to derive in the model all relevant conjectures that are evidently beyond current reach.

We will use Figure 1 to roughly illustrate how each proposal fares with respect to these two objectives (a) and (b). The take-away message from this micro-survey is that although parts of (a), (b) have been attained by prior work, ours is the first successful attempt that yields all the critical pieces under one framework.

Although the table is less precise than the discussion that follows, it does illustrate some key differences among prior work. The solid vertical line in the table is a caricature of the state of the art; to the left of the line are facts, and to the right are conjectures evidently out-of-reach. The dashed vertical line is where we would have drawn the solid vertical line had this been year 1985; it represents the relativization "barrier". (The reader should be able to visualize the title of this paper, if not now then by the end of this section.)

The first proposal is from the same paper discussed above, by **AIV** [3]. Besides $\mathcal{RCT}$, there the authors propose "local checkability" as the key non-relativizing ingredient underlying $\mathrm{PSPACE} \subset \mathrm{IP}$ as well as

[5]for usage in the sense of (i) see p.4 first par., for (ii) see p.7 last par. and p.8 second par., for (iii) see p.8 first par.

[6]for example, even defining NP using P would involve variables from the mathematical universe

Figure 1: Attempts at refining relativization

	$(\exists\mathcal{C}{:}\mathcal{C}\subset\mathrm{NEXP} \wedge \mathcal{C}\not\subset\mathrm{P/poly}) \implies \mathrm{NEXP}\not\subset\mathrm{P/poly}$	PSPACE⊂IP	PCP thm	NEXP⊄P/poly	NP⊄P, EXP⊄i.o.-P/poly,..
AIV	✓	✓	✓	?	?
For	✓	✓	?	?	✓̸
AW	?	✓	?	✓	✓
IKK	✓	✓	?	?	✓
this work	✓	✓	?	✓	✓

other results including the PCP theorem. The idea is that a polynomial-time computation should be verifiable by inspecting all bits of its transcript in parallel, where each bit depends on only a logarithmic number of bits elsewhere. For computations with oracle access, however, this property may not hold, although it will if the oracle itself is checkable. So their approach can be viewed very roughly in terms of ours, as taking our version of $\mathcal{RCT}$ and adding the constraint "B_{std} is included in $\{0, 1, \wedge, \oplus, \mathcal{O}\}$ for some $\mathcal{O}$ that is locally checkable", in other words, that any oracle $\mathcal{O}$ added to the standard basis must be locally checkable.

The authors call their refined theory $\mathcal{LCT}$, and point out that although $\mathcal{LCT}$ implies many known nonrelativizing results, whether it can settle questions such as P versus NP is very hard to know. In fact, they observe that if P versus NP were shown beyond reach of $\mathcal{LCT}$ in the manner of Baker, Gill, Solovay — by giving contradictory relativizations with oracles satisfying the theory — then P would actually be separated from NP. In this sense, $\mathcal{LCT}$ is an unsatisfactory candidate for "current techniques". (Notice that if all we want is a theory that can derive the current theorems then we can just let B_{std} be $\{0, 1, \wedge, \oplus\}$.)

In a counterview to the AIV proposal dated around the same time, **Fortnow** [17] argues that the nonrelativizing ingredient in the proof of PSPACE $\subset$ IP is of an algebraic nature. We can interpret his key insight as follows. Although PSPACE $\subset$ IP does not relativize, it does in a weaker sense: Let $\widehat{\mathcal{O}}$ denote the affine extension of $\mathcal{O}$, as defined on page 5. (Strictly speaking Fortnow works over $\mathbb{Z}$ instead of $\mathrm{GF}(2^k)$.) Then $\mathrm{PSPACE}^{\mathcal{O}} \subset \mathrm{IP}^{\widehat{\mathcal{O}}}$, and consequently, $\mathrm{PSPACE}^{\mathcal{O}} \subseteq \mathrm{IP}^{\mathcal{O}}$ whenever $\widehat{\mathcal{O}}$ Cook-reduces to $\mathcal{O}$. Effectively, then, he defines a theory $\mathcal{ACT}$ by taking our version of $\mathcal{RCT}$ and adding the constraint that any addition to the standard Boolean basis must be some oracle $\mathcal{O}$ for which $\widehat{\mathcal{O}} \in \mathrm{P}^{\mathcal{O}}$.

Although Fortnow does not prove any unprovability results for his theory, we can show that his version of $\mathcal{ACT}$ yields most of AW's classification of what algebrizes and what does not (hence the '✓̸' symbol) — but not all, as we explain later below.

A decade-and-half after the above two papers, **AW** [1] introduce algebrization. Their paper finesses the question of how relativization should be refined, by simply declaring that a statement $\mathrm{A} \subset \mathrm{B}$ relativizes algebraically if $\mathrm{A}^{\mathcal{O}} \subset \mathrm{B}^{\widehat{\mathcal{O}}}$ for every $\mathcal{O}$ (for a notion of $\widehat{\mathcal{O}}$ similar to our notion of affine extension), and that $\mathrm{A} \not\subset \mathrm{B}$ algebrizes if $\mathrm{A}^{\widehat{\mathcal{O}}} \not\subset \mathrm{B}^{\mathcal{O}}$. No definition is given for other types of statements, or for proofs.

Since we ultimately care about containments and their negations, the AW approach seems appealing. There are problems with it, however (page 1), chief among which is that not everything that relativizes can be said to algebrize. For example, the statement $(\exists\mathcal{C} : \mathcal{C} \subset \mathrm{NEXP} \wedge \mathcal{C} \not\subset \mathrm{P/poly}) \implies \mathrm{NEXP} \not\subset \mathrm{P/poly}$ is true no matter what NEXP or P/poly means — it is even true no matter what "is an element of" means — hence is relativizing, but it cannot be declared as algebrizing by building on the original definitions. Consequently, showing that NEXP $\not\subset$ P/poly is non-algebrizing, as AW did, does not rule out whether we can prove $\exists\mathcal{C} : \mathcal{C} \subset \mathrm{NEXP} \wedge \mathcal{C} \not\subset \mathrm{P/poly}$ by using solely "algebrizing techniques" [1, §10.1].

On the positive side, AW succeed in giving containments $\mathrm{A} \subset \mathrm{B}$ that do not algebrize, by showing that an oracle $\mathcal{O}$ exists for which $\mathrm{A}^{\mathcal{O}} \not\subset \mathrm{B}^{\widehat{\mathcal{O}}}$. (There are similar examples for negations of containments.) This is a critical idea upon which subsequent work expands, including ours; we say more about this below.

Soon after the AW paper, **Impagliazzo, Kabanets, Kolokolova** [22] (IKK henceforth) resume the approach of AIV, and propose an intermediate theory between $\mathcal{RCT}$ and $\mathcal{LCT}$ that they call $\mathcal{ACT}$, short for arithmetic checkability theory. (They also define a variant, $\mathcal{ACT}^*$, but we blur the distinction here.)

We can view IKK's approach as being along the same line of Fortnow's, by considering the following task. Given ϕ and α, evaluate $\Phi(\alpha)$; here ϕ is a Boolean formula, Φ is any fixed low-degree polynomial interpolating ϕ (such as its arithmetization), and α are inputs from $\mathrm{GF}(2^{O(n)})$. (Like Fortnow, IKK work over $\mathbb{Z}$, but both approaches can be adapted to $\mathrm{GF}(2^k)$.) Call the decision version of this task — given i return the i^{th} bit of the result, for example — the language AF, short for arithmetized formula evaluation.

Clearly $\mathrm{AF} \in \mathrm{P}$. Indeed, this seems to be an essential feature of arithmetization: it would seem pointless to interpolate Boolean formulas into polynomials that we cannot evaluate efficiently. But if ϕ is over an arbitrary basis $\{\wedge, \oplus, \mathcal{O}\}$, then it does not seem that $\mathrm{AF}^{\mathcal{O}} \in \mathrm{P}^{\mathcal{O}}$ since the $\mathcal{O}$-gates within ϕ need to be interpolated somehow as well.

Now, in both IKK's approach and Fortnow's, we can interpret the starting point as restricting the oracle $\mathcal{O}$ so that $\mathrm{AF}^{\mathcal{O}} \in \mathrm{P}^{\mathcal{O}}$ becomes a true statement — in Fortnow's case via the constraint $\widehat{\mathcal{O}} \in \mathrm{P}^{\mathcal{O}}$, and in IKK's case, directly via $\mathrm{AF}^{\mathcal{O}} \in \mathrm{P}^{\mathcal{O}}$. The IKK constraint (rather, our interpretation of it) is implied by Fortnow's; this will be clear in Section 1.3 once we generalize arithmetization. Hence anything provable from the IKK constraint is automatically provable from Fortnow's. Although the converse is not known to hold, it does hold in the following sense. Anything *IKK show to be* unprovable from the IKK constraint, we can show is unprovable from Fortnow's constraint as well; we can do this using our observation on page 6, that affine extensions respect disjoint unions. So the two approaches currently seem to have the same power.

The key advantage of our approach over IKK's and Fortnow's is its avoidance of computational notions in restricting the oracle $\mathcal{O}$. By giving a direct algebraic restriction, namely that $\mathcal{O}$ equals $\widehat{f}$ for some language f, our approach allows us to expand on one of AW's critical ideas: using interpolation to show that certain statements ψ do not relativize algebraically (in our case, affinely). In contrast, neither IKK's approach nor Fortnow's is known to allow interpolation. Consequently, although IKK pose it as an open question for their framework, we can capture a key result of AW, that $\mathrm{NEXP} \not\subset \mathrm{P/poly}$ does not relativize algebraically.[7]

Another place where IKK diverges from AW concerns the theorem $\mathrm{NEXP} \subset \mathrm{MIP}$. As mentioned, AW showed that this theorem algebrizes under a machine-specific restriction of the class NEXP. While IKK did show an analogous result for their framework in the model-theoretic sense, they did not show it in the proof-theoretic sense; in fact they use a machine-free characterization of NEXP and cannot directly express the query restriction of AW, so it is not even clear a priori if the result itself can be expressed in their approach. Instead, IKK observe that under the proper, unrestricted definition of NEXP, the theorem does not algebrize, and suggest that there are additional ingredients underlying this theorem besides arithmetization, and point this out as a point of divergence from the AW thesis that algebrization captures "current techniques" [22, p.15]. As mentioned in page 3, our formulation of this theorem substantially clarifies the discussion.

Whether our definitions implies IKK's or Fortnow's, or vice versa, is not clear; we do not know if algebrizing in one sense can be shown to imply the other. What we *can* say, however, is that every statement that IKK show as algebrizing has an affinely relativizing proof, and that the opposite holds for those shown non-algebrizing by IKK — just as the case for AW. In particular, IKK show various compound statements to be non-algebrizing; these follow as consequences of results on simpler statements and can be shown in our framework as well (via what we call the proof theoretic approach in Section 4.3).

[7]In fact IKK ask a weaker question: whether $\mathrm{EXP} \not\subset \mathrm{P/poly}$ can be shown to not algebrize in their framework [22, p.15]. The same question automatically applies to Fortnow's framework, since his constraint implies IKK's.

1.3 Overview of Ideas and Techniques

Defining affine relativization, and proving that it works, involve a number of observations as well as some technical ingredients. This section highlights the main ones.

Generalizing arithmetization using affine extensions. Our first observation concerns how the arithmetization method should be generalized to handle formulas over a generic Boolean basis, say $\{\wedge, \oplus, \mathcal{O}\}$ where $\mathcal{O}$ is an arbitrary language. In its typical description, the method states that the formula $\neg\phi$ arithmetizes as $1 - \Phi$ where Φ is the arithmetization of ϕ; similarly, $\phi \wedge \psi$ arithmetizes as $\Phi \cdot \Psi$. Other cases, such as $\vee$ and $\oplus$, are handled by reducing to these two.

We observe that $x \cdot y$ is the unique polynomial over $\mathbb{Z}$, of (individual) degree ≤ 1, that extends the Boolean function $(x, y) \mapsto x \wedge y$; in other words, it extends an $\wedge$-gate of fan-in 2. Similarly $1 - x$ extends a $\neg$-gate. We thus make the following generalization: Arithmetization replaces a Boolean gate $\mathcal{O}$, of fan-in m, with the gate $\widehat{\mathcal{O}}$ denoting the unique degree-≤ 1 polynomial

$$\widehat{\mathcal{O}}(x) := \sum_{b \in \{0,1\}^m} \mathcal{O}(b) \cdot \textstyle\prod_{i=1}^{m} (1 - x_i) \cdot (1 - b_i) + x_i \cdot b_i \qquad (\dagger)$$

that extends $\mathcal{O}$ from the Boolean domain to $\mathbb{Z}$. We call $\widehat{\mathcal{O}}$ *the (multi-)affine extension* of $\mathcal{O}$, and caution that the notation has nothing to do with Fourier analysis.

For our results we view (†) in fields of the form $\mathrm{GF}(2^k)$ only. There are several benefits to this, and we point them out as we explain our approach in this section. To begin with, we note that extension to $\mathrm{GF}(2^k)$ is conceptually cleaner, as it turns a function on n bits into a function on n vectors of k bits each. Also, in $\mathrm{GF}(2^k)$, the arithmetization of $\phi \oplus \psi$ becomes the natural $\Phi + \Psi$, whereas in other fields, neither $\oplus$, nor any other Boolean operator, gets arithmetized to $+$.

Affine Relativization — capturing known uses of arithmetization. Consider a functional view of an $\widehat{\mathcal{O}}$-gate, as returning k bits when each of its inputs come from $\mathrm{GF}(2^k)$. In this view, arithmetizing a formula ϕ creates a family of formulas $\{\Phi_k\}$, with each Φ_k redundantly describing the behavior of ϕ on the Boolean domain — the larger k, the higher the redundancy (with $k = 1$ corresponding to ϕ itself).

Now if ϕ is over an arbitrary basis that includes $\mathcal{O}$-gates, then unlike the case for the standard basis, its arithmetization Φ does not seem to allow efficient evaluation, over say $\mathrm{GF}(2^{O(n)})$. Interpreting this to be the non-relativizing ingredient in proofs of $\mathrm{PSPACE} \subset \mathrm{IP}$, etc., we take the following approach to refine relativization.

The formula Φ, which redundantly encodes ϕ, is obtained from ϕ via a "local" transformation acting on its gates, namely by adding redundancy at the gates. Based on this, our idea is to have the oracle gates of ϕ compute not some arbitrary $\mathcal{O}$, but something that contains redundancy already, namely $\widehat{\mathcal{O}}$ for an arbitrary $\mathcal{O}$. The plan being then to show that arithmetization — rather, current uses of it — need not introduce redundancy at those gates, or, at least do so in a feasible way.

We arrive at our formulation thus: whereas a statement relativizes if it holds relative to every language $\mathcal{O}$, a statement relativizes *affinely*, if it holds relative to every language $\mathcal{A}$ of the form $\widehat{\mathcal{O}}$ for some $\mathcal{O}$. More precisely, $\mathcal{A}$ encodes the family of polynomials $\{\widehat{\mathcal{O}_m}\}$ evaluated over $\mathrm{GF}(2^k)$ for all k, where $\mathcal{O}$ is an arbitrary language and $\mathcal{O}_m$ is its restriction to $\{0,1\}^m$. We also call $\mathcal{A}$ the *(multi-)affine extension* of $\mathcal{O}$.

Why was this notion not invented in 1994? Natural though it may seem, affine relativization poses the following difficulty: the very theorems that it is intended for, e.g. $\mathrm{PSPACE} \subset \mathrm{IP}$, do not appear to relativize affinely, at least not via a superficial examination of their proofs.

To see the issue, consider a property π of Boolean formulas — unsatisfiability, say. In proving $\pi \in \mathrm{IP}$ arithmetization is used as a *reduction*, from π to some property Π of arithmetic formulas — e.g., unsatisfiability of ϕ reduces, via arithmetization, to deciding if the product of $(1 + \Phi(\alpha))$, over all binary input vectors α, equals 1 in $\mathrm{GF}(2^k)$ for any k.

So each theorem of the form $\pi \in \mathrm{IP}$ is, in fact, a corollary of a more generic result of the form $\Pi \in \mathrm{IP}$, that gives an interactive protocol for an arithmetic property. It turns out those generic results can be further generalized, if we extend the arithmetic basis, from the standard $\times$-gates and $+$-gates — which are really $\widehat{\wedge}$- and $\widehat{\oplus}$-gates, respectively, per the first discussion above — by allowing $\widehat{\mathcal{O}}$-gates for an arbitrary $\mathcal{O}$. Then the same protocols that yield $\Pi \in \mathrm{IP}$ work just as well over this extended basis, given oracle access to the evaluation of $\widehat{\mathcal{O}}$. We may write $\Pi^{\widehat{\mathcal{O}}} \in \mathrm{IP}^{\widehat{\mathcal{O}}}$, where $\Pi^{\widehat{\mathcal{O}}}$ extends Π to formulas over the extended basis.

Now supposing we have a theorem $\pi \in \mathrm{IP}$, let us make a superficial attempt to extend its proof so that it yields $\pi^{\mathcal{A}} \in \mathrm{IP}^{\mathcal{A}}$ for some language $\mathcal{A}$; here π is a property of formulas, say over the basis $\{\wedge, \oplus\}$, and $\pi^{\mathcal{A}}$ is its extension to the basis $\{\wedge, \oplus, \mathcal{A}\}$. As just explained, the proof of $\pi \in \mathrm{IP}$ starts with a reduction, of the Boolean property π to an arithmetic property Π. Now here is the problem: what property do we reduce $\pi^{\mathcal{A}}$ to? By definition of arithmetization, it would be $\Pi^{\widehat{\mathcal{A}}}$, the extension of Π to formulas over the basis $\{\times, +, \widehat{\mathcal{A}}\}$. But then as just explained, we would be placing $\pi^{\mathcal{A}}$ in $\mathrm{IP}^{\widehat{\mathcal{A}}}$ — not in $\mathrm{IP}^{\mathcal{A}}$.

This seeming circularity — $\pi^{\mathcal{O}} \in \mathrm{IP}^{\widehat{\mathcal{O}}}$, $\pi^{\widehat{\mathcal{O}}} \in \mathrm{IP}^{\widehat{\widehat{\mathcal{O}}}}$, ... — can be interpreted as the main distraction from arriving at a natural notion such as ours. Indeed, all previous attempts to capture arithmetization [17, 1, 22], dating back to the 1994 article of Fortnow [17], can be interpreted as having to make compromises so as to break out of this circularity. For example, the AW notion of algebrization does this by declaring $\mathcal{C} \subset \mathcal{D}$ to algebrize if $\mathcal{C}^{\mathcal{O}} \subset \mathcal{D}^{\widehat{\mathcal{O}}}$ holds for every $\mathcal{O}$ (for a notion of $\widehat{\mathcal{O}}$ related to ours; there is a similar definition for $\mathcal{C} \not\subset \mathcal{D}$). We surveyed their approach and others in Section 1.2.

In contrast, our approach tackles circularity directly. The idea is to avoid the problematic reduction $\pi^{\mathcal{A}} \to \Pi^{\widehat{\mathcal{A}}}$, and to instead reduce $\pi^{\mathcal{A}}$ to $\pi^{\mathcal{O}}$ by somehow exploiting π whenever $\mathcal{A}$ is of the form $\widehat{\mathcal{O}}$ for some $\mathcal{O}$. Then the combined reduction $\pi^{\mathcal{A}} \to \pi^{\mathcal{O}} \to \Pi^{\mathcal{A}}$ breaks the circularity. This fulfils the plan of the previous discussion, namely to show that arithmetization, in its current uses, need not extend gates that are extensions of something already.

Relativizing $\oplus\mathrm{P} \subset \mathrm{IP}$. The idea of the previous discussion can be realized when π is the sum $\pi(\phi) := \oplus_x \phi(x)$, also known as the language $\oplus$SAT. This is because when ϕ is a formula over the $\mathcal{A}$-extended Boolean basis, each occurrence of $\mathcal{A}$ evaluates the sum (†) over $\mathrm{GF}(2^k)$ for some k, and then returns, say, the i^{th} bit of the result given i. Therefore, if we step from $\mathrm{GF}(2^k)$ to $\mathrm{GF}(2)^k$, we can rewrite each occurrence of $\mathcal{A}$ as $\oplus_y \gamma(y)$, for some formula γ over the $\mathcal{O}$-extended Boolean basis. This becomes the reduction we want, once we show how to convert formulas involving sums to prenex form, i.e. such that all sums appear up front. It follows that $\oplus\mathrm{SAT} \in \mathrm{IP}$ — or equivalently, $\oplus\mathrm{P} \subset \mathrm{IP}$ — relativizes affinely.

Scaling to $\mathrm{PSPACE} \subset \mathrm{IP}$ — a proof sans degree reduction. Our approach for $\oplus$P can be adapted to show that $\mathrm{PSPACE} \subset \mathrm{IP}$ affinely relativizes as well. However, we find a more natural approach which yields another proof of this theorem; this may be of separate interest because current proofs, as far as we know, employ syntactic tricks in order to control the degree of polynomials that arise from arithmetizing instances of a PSPACE-complete problem (e.g., [30, 7, 31, 2]).

In contrast we show, directly, that every downward-self-reducible language has an interactive protocol, by essentially bootstrapping the very fact that $\oplus\mathrm{P} \subset \mathrm{IP}$ relativizes affinely. In particular, we make no use of a specific PSPACE-complete problem; we do not even use any additional arithmetization beyond what is needed for $\oplus$SAT. (We emphasize that the new proof is sketched here because it might be of separate interest. The standard proofs of this theorem can also be adapted to our framework.)

The new proof goes as follows. If L is downward-self-reducible, then on inputs x of length n, it can be expressed as a $\mathrm{poly}(n)$-size circuit over the L-extended Boolean basis, of fan-in at most $n - 1$. This circuit in turn can be expressed as the sum $\oplus_y \phi(x, y)$, where ϕ is a formula verifying that y represents the computation of the circuit on input x. In notation we may summarize this reduction as

$$L_n \to \oplus\mathrm{SAT}^{L_{n-1}} \tag{$*$}$$

where $\oplus\mathrm{SAT}^{f_m}$ is the extension of $\oplus\mathrm{SAT}$ to formulas over the f-extended Boolean basis, of fan-in at most m. Repeating ($*$) for L_{n-1} instead of L_n, we get

$$\oplus\mathrm{SAT}^{L_{n-1}} \to \oplus\mathrm{SAT}^{\oplus\mathrm{SAT}^{L_{n-2}}} \to \oplus\mathrm{SAT}^{L_{n-2}} \tag{$**$}$$

where the first reduction is because extending the basis is functorial in the sense that $f \to g$ implies $\oplus\mathrm{SAT}^f \to \oplus\mathrm{SAT}^g$, and the second reduction follows by bringing sums to prenex form as mentioned in the previous discussion. Note that the reduced formula is now of size about n^{2d}, if the one in ($*$) is of size n^d.

The idea is to tame the growth in the size of the reduced formulas, by using interaction. Building on the ideas of the previous discussion, it is easy to show a protocol yielding the *interactive* reduction

$$(\oplus\mathrm{SAT}^{f_m})_{n^d} \to (\oplus\mathrm{SAT}^{f_m})_{n^c}$$

that compresses instances to $\oplus\mathrm{SAT}^{f_m}$ of size n^d down to size n^c, for an arbitrarily large d and a *fixed* c, for every language f, in particular for $f = L$, whenever $m \in O(n)$. We sketch this protocol later on page 14.

Thus we can keep repeating ($**$) to get

$$L_n \to \oplus\mathrm{SAT}^{L_{n-1}} \to \oplus\mathrm{SAT}^{L_{n-2}} \to \cdots \to \oplus\mathrm{SAT}^{L_{O(1)}}$$

provided we interleave a compression phase whenever the formula size exceeds n^c. Since an L-gate of constant fan-in can be expressed as a constant-size formula, $\oplus\mathrm{SAT}^{L_{O(1)}}$ reduces to $\oplus\mathrm{SAT}$. So $L \in \mathrm{IP}$ as desired.

That this proof affinely relativizes becomes obvious, once we carry over the results on $\oplus\mathrm{SAT}$ to the $\mathcal{A}$-extended Boolean basis, for an arbitrary affine extension $\mathcal{A}$.

(Interestingly, just as this proof builds on the relativization of $\oplus\mathrm{P} \subset \mathrm{IP}$, we use the relativization of $\mathrm{PSPACE} \subset \mathrm{IP}$ in turn to give a streamlined proof of the $\mathrm{NEXP} \subset \mathrm{MIP}$ theorem, that uses no specific NEXP-complete problem nor any additional arithmetization; see Section 3.3.)

NEXP vs. MIP — the role of locality. As mentioned in the introduction, AW show that $\mathrm{NEXP} \subset \mathrm{MIP}$ algebrizes only under a restriction, and a questionable one at that, of the oracle access mechanism for NEXP.[8] Since we define complexity classes using P, it would be even more artificial to try to express this restriction in our framework. Instead, we find a natural approach that also sheds some light into the issues surrounding oracle access.

Consider generalizing the class IP, by replacing in its definition the popular constant $2/3$ with γ, so that if the input x is supposed to be rejected, then the verifier erroneously accepts x with probability $< 1 - \gamma$. (If x should be accepted, then, as before, it is.) Call this class γ-gap-IP.

It is easy to see, by the classical PSPACE-completeness result of Stockmeyer and Meyer [33], that 0-gap-IP is identical to PSPACE. Therefore $\mathrm{PSPACE} \subset \mathrm{IP}$ can be broken into the containments

$$\mathrm{PSPACE} \subset 0\text{-gap-IP} \subset \Omega(1)\text{-gap-IP}$$

with the second containment, "gap amplification", being the actual content of the theorem.

[8]We caution that neither AW, nor we, advocate or assume that NEXP be *always* relativized in this restricted way. It is only for the purpose of deriving this theorem that this restriction seems inevitable — and this discussion investigates why.

The corresponding case for $\mathrm{NEXP} \subset \mathrm{MIP}$ becomes revealing. Of the containments

$$\mathrm{NEXP} \subset 0\text{-gap-MIP} \subset \Omega(1)\text{-gap-MIP}$$

only the second one, gap amplification, affinely relativizes as we show in Section 3.3. So what "current technique" is it that yields the first containment, that affine relativization cannot capture?

It is locality, more specifically, *polylog*-locality, which yields the following variant of the Cook-Levin theorem: A language is in P iff it has circuits that are polylog-time uniform, i.e., iff it is computable by a family $\{C_n\}$ of circuits, such that given (n,i), the task to produce the type of the ith gate of C_n, as well as the indices of all gates connected to it, can be performed in $\mathrm{poly}\log n$ time. Intuitively, this theorem does not relativize, even affinely, simply because it restricts the circuits to have polylogarithmic fan-in.

In our framework, we define P so that it satisfies *poly*-locality instead of polylog, and use P itself to express polylog-locality (by saying that the above function on pairs (n,i) is in FP) and call the class thus obtained $\mathrm{P}_{\mathrm{local}}$, the subclass of P satisfying the above locality theorem. We then use $\mathrm{P}_{\mathrm{local}}$ to define $\mathrm{NP}_{\mathrm{local}}$, $\mathrm{NEXP}_{\mathrm{local}}$, etc. Immediately two things fall out of these definitions. First, that 0-gap-MIP is identical to $\mathrm{NEXP}_{\mathrm{local}}$, so locality does capture the first containment above. Second, that $\mathrm{NEXP}_{\mathrm{local}}$ is equivalent to the dubious version of NEXP with polynomial-length oracle queries, making it not so dubious after all.

We do not know of any result using $\mathrm{NEXP} \subset \mathrm{MIP}$, that would break down if $\mathrm{NEXP}_{\mathrm{local}} \subset \mathrm{MIP}$ is used instead — in fact we do not know of any result using $\mathrm{NEXP} \subset \mathrm{MIP}$, period. We conclude that locality arises in $\mathrm{NEXP} \subset \mathrm{MIP}$ only definitionally; it is an ingredient that has not been exploited beyond making definitions. (It would be interesting to know if the same reasoning could apply to the PCP theorem; see open problem in Section 5.)

NEXP vs. P/poly — a coding-theoretic interpolation lemma. One of the technical contributions of this paper is in showing that certain statements ψ do not relativize affinely. As usual (though not always), this entails constructing an eligible language — an affine extension $\mathcal{A}$ in our case — relative to which ψ is false.

For some ψ, this task turns out to be relatively easy given prior work. Such ψ are of the form $\mathcal{C} \subset \mathcal{D}$, for which AW invented an approach based on communication complexity. Our observation that affine extensions respect disjoint unions (see Section 1.1, multiple oracles) enables us to import their approach.

For other ψ, however, in particular for $\mathrm{NEXP} \not\subset \mathrm{P}/\mathrm{poly}$, we need more substantial ideas. While AW [1] did construct a $\mathcal{Q}$ such that $\mathrm{NEXP}^{\mathcal{Q}} \subset \mathrm{P}^{\mathcal{Q}}/\mathrm{poly}$, they did this only for a multi-*quadratic* extension, i.e., for $\mathcal{Q}$ encoding a family of polynomials where each member has (individual) degree-≤ 2, instead of degree-≤ 1. It seemed "crucial" [1], in fact, to increase the degree for this purpose. While quadratic extensions suffice for the AW notion of algebrization, they do not for our notion of affine relativization.

As the key technical step for this purpose, we derive a coding-theoretic ingredient (Lemma 37 and Theorem 38), stating that knowing t bits of a codeword exposes at most t bits of its information word, and this holds for every binary code, including the affine extension (over $\mathrm{GF}(2^k)$).

AW implicitly proved a weaker form of this fact, involving quadratic polynomials. One of the ideas that enables us to do better, is to consider a different formulation for what it means to "expose" a bit of the information word. Whereas the AW approach (implicitly) considers each exposed bit as being completely revealed, our approach gives a finer treatment: an exposed bit is one whose *location* is revealed, but whose contents may vary as a function of the unexposed bits.

The advantage of this refinement is that it allows us to show, given t bits of a codeword, that the set of all codewords agreeing on these t bits form an affine space, of dimension at most t less than the maximum possible. In contrast, the AW approach resorts to using indicator polynomials to surgically alter, bit-by-bit, the codeword whose t bits are revealed; this inevitably raises the degree to quadratic because each indicator polynomial must also vanish on the t points that are revealed, in addition to all-but-one point of the Boolean cube.

Compressing $\oplus\mathrm{SAT}$. For the sake of completing the sketch of the alternate proof of $\mathrm{PSPACE} \subset \mathrm{IP}$ explained earlier, we now outline the compression protocol mentioned.

The protocol is based on the fact alluded to earlier, that $\oplus\mathrm{SAT}^f \in \mathrm{IP}^{\widehat{f}}$ for any language f. This fact follows from standard considerations: Given ϕ over the f-extended basis, in order to compute $\oplus_z \phi(z)$, the verifier: (i) arithmetizes ϕ to get Φ, a formula over the $\widehat{f}$-extended arithmetic basis, (ii) engages in a sumcheck protocol, thus reduces the original task to that of evaluating Φ over $\mathrm{GF}(2^k)$, with $k \in O(\log n)$ being sufficient for ϕ of size n, and (iii) evaluates Φ, by using the $\widehat{f}$-oracle for the $\widehat{f}$-gates.

The compression protocol also starts out as above. The difference begins in step (iii): instead of calling the $\widehat{f}$-oracle, the verifier engages the prover. By using standard interpolation techniques, the verifier reduces the task of computing the values of $\widehat{f}$ on up to n points, to doing the same on just m points or fewer, where m is the largest fan-in of any f-gate in the formula ϕ.

Thus the output of step (iii) is a list of at most m claims of the form "$\widehat{f}_{m'}(x) = v$" with $m' \leq m$ and $v, x_i \in \mathrm{GF}(2^k)$. Now because $\widehat{f}_{m'}$ is merely the sum (†) on page 10, which can be viewed in $\mathrm{GF}(2)^k$ rather than in $\mathrm{GF}(2^k)$, it follows that these claims can be expressed as a conjunction of $\oplus\mathrm{SAT}^{f_m}$-instances, of combined size $\mathrm{poly}(mk)$. This yields the compressed instance, since $\oplus\mathrm{SAT}$ is closed under conjunction.

2 Definitions, Notation and Conventions

$\mathcal{O}$ and $\mathcal{A}$. Unless stated otherwise, $\mathcal{O}$ stands for an arbitrary language, and $\mathcal{A}$ for its affine extension as defined later in this section.

Well-behaved resource bound. We call a function $s : \mathbb{N} \to \mathbb{N}$ a *well-behaved resource bound* if it is increasing, satisfies $O(s(n)) \subset s(O(n)) \subset s(n)^{O(1)} \subset s(n^{O(1)})$ and $n \leq s(n)$, and if the function $n \mapsto s(n)$ is in FP. Functions of the form $n^d, (n^d \log n)^{d'}, 2^{(\log n)^d}, 2^{dn}$ are well-behaved resource bounds.

The above generalizes to $s : \mathbb{N}^2 \to \mathbb{N}$ if fixing either of the inputs yields a well-behaved resource bound.

Languages as families. We view languages $L : \{0,1\}^* \to \{0,1\}$ as families of Boolean functions $\{L_n : \{0,1\}^n \to \{0,1\}\}_{n\in\mathbb{N}}$, though we sometimes specify them as $\left\{f_m : \{0,1\}^{s(m)} \to \{0,1\}\right\}_{n\in\mathbb{N}}$, or as $\left\{f_{m,k} : \{0,1\}^{s(m,k)} \to \{0,1\}\right\}_{m,k\in\mathbb{N}}$ for some well-behaved resource bound s that is bounded by a polynomial (respectively, in m or in mk).

It is an elementary fact that a family of the form $\{f_m\}$ or $\{f_{m,k}\}$ as above can be efficiently viewed as a language of the form $\{L_n\}$ as above, and vice versa. For concreteness, let $m \diamond k$ denote the Cantor pairing of m and k. Then given $\{f_{m,k}\}$, define $\{L_n\}$ as $L_n(x) := f_{m,k}(x_{1..s(m,k)})$ for the largest $m \diamond k$ such that $s(m \diamond k, m \diamond k) \leq n$. Conversely, given $\{L_n\}$, define $\{f_{m,k}\}$ as $f_{m,k}(x) := L_n(x0^p)$, where p is set so that the input to L is of length exactly $n = s(m \diamond k, m \diamond k)$.

Representing $\mathbb{F}_{2^k}$. We represent each element of $\mathbb{F}_{2^k}$ by a k-bit Boolean string, forming the coefficients of a polynomial in the ring $\mathbb{F}_2[x]$ mod some irreducible $p_k(x)$ of degree k. We fix a uniform collection $\{p_k\}_{k\in\mathbb{N}}$ so that a deterministic algorithm can produce p_k in time polynomial in k [32].

The **Boolean version** of a function $q : \mathbb{F}_{2^k}^m \to \mathbb{F}_{2^k}$ is, for concreteness, the function $\mathrm{bool}(q)$ mapping (x, y) to the y^{th} bit of $q(x)$. (Our results do not depend on this definition; any other equivalent function under Cook reductions would work.)

Affine extensions. (This definition uses all the definitions above.)

Given $f_m : \{0,1\}^m \to \{0,1\}$, we define its *affine extension polynomial* as the unique m-variate polynomial over $\mathbb{F}_2$, with individual degree ≤ 1, that agrees with f_m over $\mathbb{F}_{2^k}$ for all k, i.e., as

$$\widehat{f}_m(x) := \textstyle\sum_{b\in\{0,1\}^m} f_m(b) \cdot \prod_{i=1}^m (1 + x_i + b_i)$$

By the *affine extension* of $f_m : \{0,1\}^m \to \{0,1\}$, we mean the family

$$\widetilde{f}_m := \left\{ \widetilde{f}_m^{\,k} \right\}_{k\in\mathbb{N}}$$

where $\widehat{f}_m^{\,k}$ denotes the function that evaluates $\widehat{f}_m$ over $\mathbb{F}_{2^k}$, and $\widetilde{f}_m^{\,k}$ denotes the Boolean version of $\widehat{f}_m^{\,k}$.

Given a family $f := \{f_m\}$ we define its affine extension $\widetilde{f}$ (or its affine extension polynomial $\widehat{f}$) as the family obtained by applying the above definitions to each member. In particular, for the language

$$\mathcal{O} = \{\mathcal{O}_m : \{0,1\}^m \to \{0,1\}\}_{m\in\mathbb{N}}$$

its affine extension $\widetilde{O}$, which we denote by $\mathcal{A}$, is

$$\mathcal{A} := \left\{ \mathcal{A}_{m,k} : \{0,1\}^{mk+\lceil \log k\rceil} \to \{0,1\} \right\}_{k,m\in\mathbb{N}}$$

$$\mathcal{A}_{m,k} : (y_1..y_m z) \mapsto z^{\text{th}} \text{ bit of } \widehat{\mathcal{O}}_m(y_1,..,y_m)$$

where each y_i is interpreted as a member of $\mathbb{F}_{2^k}$. (By the previous definitions, $\mathcal{A}$ can be efficiently viewed as a family of the form $\{\mathcal{A}_n : \{0,1\}^n \to \{0,1\}\}_{n\in\mathbb{N}}$, and vice versa.)

By an *affine extension language*, we mean the affine extension of a language.

(Affine) Relativization. For succinctness we summarize the discussion in Section 1.1 here. Working in the first order language with signature $\{\in, B_{std}\}$, define

1. $\mathcal{RCT}$:= ZFC + "$\{\mathbf{0},\mathbf{1},\wedge,\oplus\} \subset B_{std}$" + "$B_{std} \subset \{\mathbf{0},\mathbf{1},\wedge,\oplus,\mathcal{O}\}$ for some language $\mathcal{O}$"
2. $\mathcal{ACT}$:= $\mathcal{RCT}$ + "$B_{std} \subset \{\mathbf{0},\mathbf{1},\wedge,\oplus,\mathcal{A}\}$ for some affine extension language $\mathcal{A}$"
3. $\mathcal{CT}$:= $\mathcal{ACT}$ + "$B_{std} \subset \{\mathbf{0},\mathbf{1},\wedge,\oplus\}$"

where ZFC is the axioms of everyday set theory, and $\wedge$,$\oplus$,$\mathbf{0}$,$\mathbf{1}$ is shorthand for the AND/XOR/constant 0/constant 1 function on binary strings. The statements in quotes are to be formalized in the obvious way.

We say ψ has a proof iff $\mathcal{CT} \vdash \psi$, has a relativizing proof iff $\mathcal{RCT} \vdash \psi$, and has an affinely relativizing proof iff $\mathcal{ACT} \vdash \psi$. We say ψ relativizes iff $\mathcal{RCT} \models_{\overline{std}} \psi$, shorthand for ψ being true in all standard models of $\mathcal{RCT}$. We say ψ affinely relativizes iff $\mathcal{ACT} \models_{\overline{std}} \psi$.

We do not know if $\mathcal{CT} \vdash \psi$ and $\mathcal{RCT} \models_{\overline{std}} \psi$ together imply $\mathcal{RCT} \vdash \psi$ (the converse is immediate), but in the sequel, all theorems of the form "ψ, and this also holds if the standard Boolean basis is extended with $\mathcal{O}$" are derived by proving that $\mathcal{RCT} \vdash \psi$. Similarly, "$\psi$, and this also holds if the standard Boolean basis is extended with $\mathcal{A}$" is derived by proving that $\mathcal{ACT} \vdash \psi$.

P **and the Cook-Levin Theorem.** For a family of circuits $C := \{C_n\}_n$ over B_{std}, define

(a) $\text{Desc}_C : (1^n, 1^i) \mapsto$ the type of the i^{th} gate in C_n and the indices of all gates connected to the i^{th} gate.

(b) $\text{StrongDesc}_C : (n, i) \mapsto \text{Desc}_C(1^n, 1^i)$

We define P as the set of languages L computable by a polynomial-size family C_L for which Desc_{C_L} is computable by a polynomial-time Turing machine. Of the two statements

(i) every $L \in$ P is computable by a polynomial-size circuit family C_L for which Desc_{C_L} is in FP.

(ii) every $L \in$ P is computable by a polynomial-size circuit family C_L for which StrongDesc_{C_L} is in FP.

we call (i) the Cook-Levin theorem, and (ii) the strong Cook-Levin theorem. We take for granted that $\mathcal{RCT}$ proves (i), and $\mathcal{CT}$ proves (ii) [15]. (It is a consequence of Proposition 26 that $\mathcal{ACT}$ does not prove (ii).)

In fact, $\mathcal{RCT}$ proves that Desc_{C_L} is in "unrelativized" FP, i.e., that $C_{\mathrm{Desc}_{C_L}}$ is over the basis $\{0, 1, \oplus, \wedge\}$. (Similarly for $\mathcal{CT}$ and $\mathrm{StrongDesc}_{C_L}$, but we do not use this.)

By a polynomial-time algorithm we mean the circuit family C_L for the decision version L of a function $f \in \mathrm{FP}$.

Partial languages. We call f a partial language if it is a language or can be extended to a language.

In particular, for a language $L := \{L_n\}_{n\in\mathbb{N}}$ we use $L_{\leq n}$ to denote the partial language $\{L_i\}_{i\leq n}$.

Boolean bases. We define a Boolean basis inductively, as either the set $\{0, 1, \oplus, \wedge\}$, or the set $B \cup \{f\}$ where B is a Boolean basis and f is a partial language for which $\widetilde{f}$ is defined (in particular, f is defined on either all length-n strings or none, for every n). Here $\wedge$,$\oplus$,0,1 is shorthand for the AND / XOR / constant 0 / constant 1 function on binary strings respectively.

By the *standard Boolean basis* we mean the set $\{0, 1, \oplus, \wedge\}$, and by the basis B *extended with* f we mean $B \cup \{f\}$. (Caution: extending the standard basis changes only what we *mean* by the standard basis.)

The (partial) language $\oplus\mathrm{SAT}^f$. For every Boolean basis B and eligible partial language f, we define $\oplus\mathrm{SAT}^f$ as the partial language mapping $\phi(\vec{x})$ to the evaluation of the mod-2 sum $\oplus_{\vec{\alpha}}\phi(\vec{\alpha})$, where ϕ denotes a formula over the basis B extended with f. By default B is the standard basis and f is the trivial map $x \mapsto 0$.

$\oplus\mathrm{SAT}^f$ is undefined on those ϕ, and only on those, that are undefined on some input $\vec{\alpha}$ (due to some gate of ϕ receiving an input out of its domain while evaluating $\phi(\vec{\alpha})$). So over the standard basis, if f is a language then so is $\oplus\mathrm{SAT}^f$, and the same holds if the standard basis is extended with any language.

We index $\oplus\mathrm{SAT}$ by n, any upper bound on the number of gates of the formula ϕ. That is, we view $\oplus\mathrm{SAT}$ as $\{\oplus\mathrm{SAT}_n\}_{n\in\mathbb{N}}$, where $\oplus\mathrm{SAT}_n$ is defined on length-$s(n)$ strings for some fixed $s(n) \in \mathrm{poly}(n)$, with each such string representing a formula ϕ of at most n gates.

Since $(\oplus\mathrm{SAT}^f)^g$ is equivalent to $(\oplus\mathrm{SAT}^g)^f$ under Karp reductions, we write $\oplus\mathrm{SAT}^{f,g}$ to mean either.

(Interactive) Reductions. For partial languages f and g, we write

$$f \to g$$

if there is an interactive protocol satisfying the following. Let $[E]$ denote the maximum probability that the event E occurs, over all possible prover strategies in the protocol. Then given $x \in \{0, 1\}^n$, the protocol runs in time $\mathrm{poly}(n)$ and outputs y such that:

(i) if $x \in \mathrm{dom}\, f$ then $[y \in \mathrm{dom}\, g$ and $f(x) = g(y)] \geq 1 - \varepsilon$,

(ii) if $x \in \mathrm{dom}\, f$ then $[y \notin \mathrm{dom}\, g \cup \{\text{'fail'}\}] \leq \varepsilon$

for a negligible function $\varepsilon \in (1/n)^{w(1)}$.

Since the verifier in an interactive protocol may ignore the prover, and/or ignore its coin flips, we use the same notation $f \to g$ for randomized reductions, as well as for Karp reductions.

By default, $f \to g$ denotes a Karp reduction.

Cook reductions. For partial languages f and g, we say f Cook-reduces to g, if there is a function $V \in \mathrm{FP}$ that behaves as follows. On input (x, a), intuitively, V consumes a bit-by-bit, interpreting each bit as the response to its next query for g; if a is of length exactly $\ell(|x|)$, say, then V computes $f(x)$, otherwise V computes the next query for g — here ℓ is a well-behaved size bound in $\mathrm{poly}(n)$. More precisely,

- $V(x, a) \in \mathrm{dom}\, g$ if $x \in \mathrm{dom}\, f$ and $|a| < \ell(|x|)$ and $a_i = g(V(x, a_{1..i-1}))$ for $i = 1..|a|$,
- $V(x, a) = f(x)$ if $x \in \mathrm{dom}\, f$ and $|a| = \ell(|x|)$ and $a_i = g(V(x, a_{1..i-1}))$ for $i = 1..|a|$.

3 Positive Relativization Results

This section shows that the famous results on interactive proofs admit affinely relativizing proofs, as do the circuit lower bounds that build on them. These are the IP theorem of Shamir ($\mathrm{PSPACE} \subset \mathrm{IP}$, Section 3.2), the MIP theorem of Babai, Fortnow, and Lund ($\mathrm{NEXP} \subset \mathrm{MIP}$, Section 3.3), the ZKIP theorem of Goldreich, Micali, and Wigderson ($\mathrm{NP} \subset \mathrm{ZKIP}$ if one-way functions exist, Section 3.5), and the strongest lower bounds known-to-date against general Boolean circuits, by Buhrman, Fortnow, Thireauf, and by Santhanam (Section 3.4). All of these build on several properties of $\oplus\mathrm{SAT}$ developed in Section 3.1.

(As explained in Section 1.1, we do not state these results in proof-theoretic terms; e.g., Theorem 2 asserts, among other things, that $\oplus\mathrm{SAT} \subset \mathrm{IP}$ affinely relativizes, rather than that it has such a proof, even though the latter also holds.)

3.1 Checking and Compressing $\oplus\mathrm{SAT}$

This section develops three results on $\oplus\mathrm{SAT}$ that enable most of the positive relativization results in the paper.

We first recall some notions from program checking [11].

Definition 1 (Same-length checkable)**.** We say a language $L := \{L_n\}_{n\in\mathbb{N}}$ is *same-length checkable* if there is an interactive protocol for computing L, i.e. for deciding the language $(x, b) \mapsto L(x) \equiv b$, wherein the prover acts as a purported oracle for $L_{|x|}$. We say L is *checkable* if the prover is allowed to answer queries also for $L_{\neq|x|}$. We refer to the verifier algorithm in these protocols as a *checker* for L.

The first main result in this section shows (via an affinely relativizing proof) the existence of a $\oplus\mathrm{P}$-complete language that is same-length checkable.

Theorem 2 (Checking $\oplus\mathrm{SAT}$)**.** *$\oplus\mathrm{SAT}$ is checkable. In fact, $\oplus\mathrm{SAT}$ is equivalent, under Karp reductions, to some language that is* same-length *checkable.*

This also holds if the standard Boolean basis is extended with $\mathcal{A}$, via a checker that has access to $\mathcal{A}$. Neither the checker nor any of the reductions depend on the choice of $\mathcal{A}$.

Theorem 2 is used, from different aspects, in deriving Shamir's IP theorem (Section 3.2) and the circuit lower bounds of Buhrman et al. and of Santhanam (Section 3.4).

The second result gives an interactive compression scheme for $\oplus\mathrm{SAT}^L$, which cuts the size of a formula from n^d to n^c, for an arbitrary large d and a fixed c, as long as the L-gates have fan-in $O(n)$ in the original formula. (The runtime of the interaction depends on d.) The verifier in the interaction need not have oracle access to L; in fact L may even be undecidable as far as the verifier is concerned.

Theorem 3 (Compressing $\oplus\mathrm{SAT}$)**.** *For every language $L := \{L_m\}_{m\in\mathbb{N}}$, there is an interactive protocol that reduces instances of $\oplus\mathrm{SAT}^{L_{\leq m}}$ of size n, to instances of $\oplus\mathrm{SAT}^{L_{\leq m}}$ of size $\mathrm{poly}(m \log n)$ for every m, n.*

This also holds if the standard Boolean basis is extended with $\mathcal{A}$, via a protocol that has access to $\mathcal{A}$. The protocol does not depend on the choice of $\mathcal{A}$.

Theorem 3 is used in deriving Shamir's IP theorem (Section 3.2) with a new streamlined proof of that result.

We also derive an auxiliary fact that comes in handy when proving Theorems 2-3 and the IP theorem:

Proposition 4. *$\oplus\mathrm{SAT}^f \to \oplus\mathrm{SAT}^g$ whenever f Cook-reduces to g. The reduction works over any basis for formulas, and depends only on the Cook-reduction from f to g.*

Remark. Throughout this section, "reduction" disambiguates to the *algorithm* that yields the reduction.

3.1.1 Proofs of Theorems 2-3 and Proposition 4

We now prove the claims made in the previous section. We do this in three steps: In the first step we generalize $\oplus\mathrm{SAT}$, from formulas to expressions involving sums. In the second step, we define arithmetic analogues of $\oplus\mathrm{SAT}$ and of its generalization. We derive some key facts about these functions, right after defining them. In the last step we put everything together and prove Theorems 2-3 and Proposition 4.

— Step 1: Generalizing $\oplus\mathrm{SAT}$ to $\oplus^*\mathrm{SAT}$ —

Definition 5 (bbs)**.** For every Boolean basis B, consider the set of expressions obtained inductively, by letting in: (i) every variable; (ii) $f(\psi_1..\psi_m)$ for every $\psi_1,..,\psi_m$ already let in, for every element f in the basis B of arity m, for every $m \in \mathbb{N}$; (iii) $\oplus_y\psi$ for every ψ already let in, for every free variable y of ψ. Call this the set of *Boolean expressions involving binary sums* (bbs) over the basis B.

Definition 6 ($\oplus^*\mathrm{SAT}$)**.** Define $\oplus^*\mathrm{SAT}^f$ over the basis B as the map $\psi(\vec{x}) \mapsto \oplus_{\vec{\alpha}}\psi(\vec{\alpha})$ where ψ is a bbs over $B \cup \{f\}$, with input variables $\vec{x}$. By default B is the standard basis and f is the trivial map $x \mapsto 0$.

In this section we derive two facts relating $\oplus\mathrm{SAT}$ to $\oplus^*\mathrm{SAT}$:

Lemma 7. $\oplus\mathrm{SAT}^f \to \oplus^*\mathrm{SAT}^g$ *whenever f Cook-reduces to g. The reduction works over any basis for formulas, and depends only on the Cook-reduction from f to g.*

Lemma 8. $\oplus\mathrm{SAT}^{\oplus\mathrm{SAT}} \to \oplus^*\mathrm{SAT}$. *The reduction works over any basis for formulas.*

Lemma 7 is a weaker version of Proposition 4. Lemma 8 is what one would expect. We prove these next.

Proof of Lemma 7. Let $V \in \mathrm{FP}$ be the function realizing the Cook-reduction from f to g, as defined in Section 2. By applying the Cook-Levin theorem to V, we get a well-behaved size bound $\ell \in \mathrm{poly}(n)$, and a function in FP that on input 1^n, produces a sequence of $\ell(n)+1$ circuits $C_1,..,C_{\ell(n)},C_*$, such that

- $C_i : \{0,1\}^n \times \{0,1\}^{i-1} \to \{0,1\}^{\ell(n)}$
  $C_* : \{0,1\}^n \times \{0,1\}^{\ell(n)} \to \{0,1\}$
- $C_i(x,a) \in \mathrm{dom}\, g,$ if $x \in \mathrm{dom}\, f$ and $a_i = g(C'_i(x,a_{1..i-1}))$ for $i \in 1..|a|$,
- $C_*(x,a) = f(x),$ if $x \in \mathrm{dom}\, f$ and $a_i = g(C'_i(x,a_{1..i-1}))$ for $i \in 1..|a|$,

where C'_i indicates that the ℓ-bit output of C_i is to be trimmed off excess bits before being fed into g (here ℓ upper bounds the "running time" of V hence the length of each "query" made by V). More precisely, $C'_i(z) := (C_i(z))_{1..D_i(z)}$ where D_i is a circuit that computes the number of valid output bits of $C_i(z)$.

Therefore $f(x)$, which can be written as

$$V(x,a), \quad \text{where } a_i = g(V(x,a_{1..i-1})) \text{ and } |a| = \ell$$

can also be written as

$$\bigoplus_{a\in\{0,1\}^\ell} C_*(x,a) \wedge \textstyle\bigwedge_{i=1..\ell}(a_i \equiv g(C'_i(x,a_{1..i-1}))$$

for every $x \in \mathrm{dom}\, f \cap \{0,1\}^n$, by some function in FP. The same function can view the expression inside the sum as a circuit over the g-extended basis, say as $E(x,a)$, and rewrite it as the formula $\xi(x,a,v)$ checking that v describes the computation of E on its input.

It follows that there is a function in FP that given input 1^n, outputs a formula ξ such that

$$f(x) = \textstyle\bigoplus_{a,v}\xi(x,a,v)$$

for every $x \in \operatorname{dom} f \cap \{0,1\}^n$. Consequently, there is a function in FP that given a formula ϕ, takes each occurrence of a subformula of the form $f(\phi_1..\phi_n)$, and performs the replacement

$$f(\phi_1..\phi_n) \mapsto \bigoplus_{a,v} \xi(\phi_1..\phi_n, a, v).$$

proving that $\oplus\mathrm{SAT}^f \to \oplus^*\mathrm{SAT}^g$. The reduction depends only on the Cook-reduction from f to g, and works over any choice of a basis for formulas. $\square$

Proof of Lemma 8. Let B be a Boolean basis for formulas. Given a formula $\phi(x)$ over B extended with $\oplus\mathrm{SAT}$, we want a reduction from the task of computing $\oplus_x \phi(x)$ to that of computing $\oplus_z \psi(z)$, for some bbs $\psi(z)$ over B. We want the reduction to work for every choice of B.

The reduction proceeds by simply replacing each occurrence of $\oplus\mathrm{SAT}$ in $\phi(x)$ with the actual sum to be computed. More precisely, let $\mathrm{FormulaEval}$ be the partial language that, on input (t, u), interprets t as a formula τ over the basis B, and outputs $\tau(u)$, the evaluation of τ on u. (In case τ has fewer inputs than $|u|$, let $\mathrm{FormulaEval}$ output $\tau(u)$ only if the extra bits in u are set to zero, else let it output zero.)

Each subformula in ϕ of form

$$\oplus\mathrm{SAT}(\phi_1..\phi_m) \tag{1}$$

can be viewed as the sum

$$\oplus_{u\in\{0,1\}^m} \mathrm{FormulaEval}(\phi_1..\phi_m, u)$$

for each setting of x, since the subformulas $\phi_1(x), .., \phi_m(x)$ describe a Boolean formula τ_x with $\le m$ input variables. And since $\mathrm{FormulaEval}$ Cook-reduces to the basis B — more precisely, to the function $(i, x) \mapsto B_i(x)$ where B_i is the ith element of the basis B — the same reasoning in the proof of Lemma 7 gives a formula ξ over the basis B such that

$$\mathrm{FormulaEval}(z) = \bigoplus_{a,v} \xi(z, a, v)$$

for every input $z \in \{0,1\}^m$. Here ξ can be produced by some function in FP from the input 1^m. Hence there is a function in FP that takes each occurrence of a subformula of the form (1), and performs the replacement

$$\oplus\mathrm{SAT}(\phi_1..\phi_m) \mapsto \bigoplus_{u,a,v} \xi(\phi_1..\phi_m, u, a, v)$$

proving that $\oplus\mathrm{SAT}^{\oplus\mathrm{SAT}} \to \oplus^*\mathrm{SAT}$. The reduction works over any choice of a basis for formulas. $\square$

— Step 2: Arithmetic analogues of $\oplus\mathrm{SAT}$ and $\oplus^*\mathrm{SAT}$ —

In order to proceed towards Proposition 4 and Theorems 2-3, we need to define the arithmetic analogues of $\oplus\mathrm{SAT}$ and $\oplus^*\mathrm{SAT}$. We begin by introducing arithmetic bases.

Definition 9 (Arithmetic basis)**.** For every Boolean basis B, define the *arithmetic basis* $\widehat{B}$ as the set comprising all constants in $\mathbb{F}_{2^k}$ for each k, and $\widehat{f}$ for each $f \in B$. By the standard arithmetic basis we mean $\widehat{B}$ where B is the standard Boolean basis.

The following two definitions are very analogous to Definitions 5-6 (bbs - $\oplus^*\mathrm{SAT}$).

Definition 10 (abs)**.** For every arithmetic basis A, consider the set of expressions obtained inductively, by letting in: (i) every variable; (ii) $f(\Psi_1..\Psi_m)$ for every $\Psi_1, .., \Psi_m$ already let in, for every element in A defined on m variables, for every $m \in \mathbb{N}$; (iii) $\sum_{y\in\{0,1\}} \Psi$ for every Ψ already let in, for every free variable y of Ψ. Call this the set of *arithmetic expressions involving binary sums* (abs) over the basis A.

Definition 11 ($+^*\mathrm{ASAT}$, $+\mathrm{ASAT}$)**.** Define $+^*\mathrm{ASAT}^f$ over the basis A as the Boolean version (as defined in Section 2) of the map $\Psi(\vec{x}) \mapsto \Sigma_{\vec{\alpha}}\Psi(\vec{\alpha})$; here Ψ is an abs over $A \cup \{\widehat{f}\}$ with input variables $\vec{x}$, and each α_i ranges over $\{0,1\}$. By default, A is the standard arithmetic basis and f is the trivial map $x \mapsto 0$..

Define $+\mathrm{ASAT}^f$ as the special case of $+^*\mathrm{ASAT}^f$ where $\Psi(\vec{x})$ is in fact a formula over A.

We index $+\mathrm{ASAT}$ by n and k, and write the corresponding member as $+_k\mathrm{ASAT}_n$; here n upper bounds the number of nodes in formula Φ, and k denotes the field $\mathbb{F}_{2^k}$ where the constants of Φ reside.

For our purposes (to be made clear in Step 3) we require that each instance of $+_k\mathrm{ASAT}_n$, say involving the formula Φ, is represented such that each input node of Φ takes up $\geq k$ bits. We also require $k \geq \log^2 n$.

In this section we derive four facts relating $+^*\mathrm{ASAT}$, $+\mathrm{ASAT}$, $\oplus^*\mathrm{SAT}$, $\oplus\mathrm{SAT}$:

Lemma 12 (Arithmetization)**.** $\oplus^*\mathrm{SAT} \to +^*\mathrm{ASAT}$. *The reduction works over any Boolean basis for formulas and its corresponding arithmetic basis.*

In addition, the same reduction yields $\oplus\mathrm{SAT}_n \to +_k\mathrm{ASAT}$ *for some* $k \in \mathrm{poly}\log n$.

Lemma 13 (Prenex)**.** $+^*\mathrm{ASAT} \to +\mathrm{ASAT}$. *The reduction works over any basis for formulas.*

Lemma 14 (Booleanization)**.** $+\mathrm{ASAT} \to \oplus\mathrm{SAT}$. *The reduction works over any Boolean basis for formulas and its corresponding arithmetic basis.*

Corollary 15. $\oplus^*\mathrm{SAT} \to \oplus\mathrm{SAT}$. *The reduction works over any basis for formulas.*

Proof of Lemma 12. Given a bbs ϕ over any Boolean basis B, let Φ be its arithmetization as defined in Section 1.3, i.e., Φ is obtained by replacing each non-input gate f in ϕ with its affine extension polynomial $\widehat{f}$, and by replacing each mod-2 sum with a generic sum so that a subexpression of ϕ of the form $\oplus_{y\in\{0,1\}}\phi'$ becomes $\Sigma_{y\in\{0,1\}}\Phi'$.

Because $\widehat{f}$ agrees with f on Boolean settings of its inputs by definition (Section 2), it follows that ϕ agrees with Φ on every Boolean input. And because we represent $\mathbb{F}_{2^k}$ as k-bit vectors (Section 2), computing $\oplus_{\vec{\alpha}}\phi(\vec{\alpha})$ reduces to computing the least significant bit of $\Sigma_{\vec{\alpha}}\Phi(\vec{\alpha})$ over $\mathbb{F}_{2^k}$ for any k, where each α_i ranges over $\{0,1\}$ in both sums. The reduction works over any choice of a basis for formulas. $\square$

Proof of Lemma 13. Given an abs Ψ over any arithmetic basis A, we give a reduction that produces a (summation-free) formula Φ over A satisfying, for every setting of inputs x of Ψ over $\mathbb{F}_{2^k}$ for every k,

$$\Psi(x) = \Sigma_y\Phi(x,y).$$

There is nothing to do if Ψ is just a variable or constant, so suppose not.

If $\Psi(x)$ is of the form $\Psi_1 \cdot \Psi_2$, and if by recursion Ψ_1 is already brought to the desired form $\Sigma_y\Phi_1(x,y)$, and Ψ_2 to $\Sigma_z\Phi_2(x,z)$, then the rest is easy: just make sure y and z refer to disjoint sets of variables by renaming as needed, and write $\Psi(x) = \Sigma_{y,z}\Phi_1(x,y)\cdot\Phi_2(x,z)$.

In case $\Psi = \Psi_1 + \Psi_2$, after recursing and renaming as before, write

$$\Psi(x) = \textstyle\sum_{b,y,z}\big(\Phi_1(x,y)\cdot b\cdot\Pi_i z_i \;+\; \Phi_2(x,z)\cdot(1-b)\cdot\Pi_i y_i\big),$$

where b is a single variable.

In case Ψ is of the form $\widehat{f}(\Psi_1,..,\Psi_m)$, where f is a nonstandard basis element, use the definition of $\widehat{f}_m$ (Section 2) to rewrite Ψ as

$$\Psi(x) = \textstyle\sum_{b_1..b_m}\widehat{f}(b_1,..,b_m)\cdot\prod_{i=1..m}(1+\Psi_i(x)+b_i), \tag{2}$$

then recurse into the product on the right side, and then finish by going to the first case, $\Psi = \Psi_1\cdot\Psi_2$.

The reduction works over any choice of a Boolean basis for formulas and its corresponding arithmetic basis. $\square$

Proof of Lemma 14. Given an arithmetic formula $\Phi(x)$ and given ℓ, we give a reduction from finding the ℓ^{th} bit of $\Sigma_x \Phi(x)$, to evaluating the mod-2 sum $\oplus_z \phi(z)$ for some Boolean formula ϕ.

To begin with, let us assume that there are no nonstandard $\widehat{f}$-gates in Φ, in other words, that Φ is a $\mathbb{F}_{2^k}$-polynomial for some k. By the way we represent $\mathbb{F}_{2^k}$ (Section 2), there is a Boolean circuit $C(X)$ that takes as input a k-bit vector X_j corresponding to each input x_j of $\Phi(x)$, and outputs k bits representing the value $\Phi(x)$. C is constructible in polynomial-time given Φ, independent of the choice of a basis.

Because the original task is to find the ℓ^{th} bit of the sum $\Sigma_x \Phi(x)$, and because addition in $\mathbb{F}_{2^k}$ corresponds to componentwise addition in $\mathbb{F}_2^k$, we can ignore all output bits of C except the ℓ^{th} one. Further, because the summation variables x_i range over binary values, we can fix in each X_i all the bits to 0 except the least significant bit, which we can call x_i. So we now have a circuit $C(x)$ returning the ℓ^{th} bit of $\Phi(x)$ for every x from the Boolean domain.

It follows that the ℓ^{th} bit $\Sigma_x \Phi(x)$ equals $\oplus_{x,y} \phi(x, y)$, where ϕ is the formula verifying that y describes the computation of the circuit C on input x. This proves the lemma when $\Phi(x)$ is a polynomial.

Now suppose that Φ contains $\widehat{f}$-gates for an arbitrary f. Mimicking the above reasoning for the standard basis, we want to express the evaluation of Φ as a Boolean circuit C over the f-extended Boolean basis. Once this is done, the rest follows as in the earlier case with no $\widehat{f}$-gates.

Perform the process, explained in the proof of Lemma 13 just above, of bringing Φ to prenex form — a seemingly useless thing to do as Φ does not involve sums. But notice that as a side effect, the process transforms the summation-free $\Phi(x)$ into the sum $\Sigma_B \Phi'(x, B)$, where each $\widehat{f}$-gate in Φ', say the i^{th} one, is "isolated" in the sense that its inputs now come from some $B_{i1}, .., B_{im_i}$ among the variables B, which all range over Boolean values. Since $\widehat{f}$ agrees with f on Boolean inputs, now the $\widehat{f}$-gates can be replaced with f-gates.

It thus follows, with the same reasoning as earlier, that the ℓ^{th} bit of $\Sigma_x \Phi(x)$ — which is the same as the ℓ^{th} bit of $\Sigma_{x,B} \Phi'(x, B)$ — equals $\oplus_{x,B,y} \phi'(x, B, y)$, where ϕ' is a formula over the Boolean basis corresponding to the basis of Φ. The reduction works over any choice of a basis for Φ. $\square$

— Step 3: Putting things together —

We now proceed to prove Theorems 2-3 and Proposition 4, beginning with the proposition.

Proof of Proposition 4. Immediate from Lemma 7 and Corollary 15. $\square$

Both Theorem 2 and Theorem 3 involve interactive proofs. For shortening the exposition, we now give a generic lemma that provides this ingredient to both results. It says that claims of the form '$\Sigma_x \Phi(x) = v$' are same-length checkable, provided that the checker has oracle access to each gate in Φ. If there is no access for one type of gate, say the $\widehat{f}$-gates, then instead of a yes/no answer, the checker can output a small conjunction of claims of the form '$\widehat{f}(z) = w$' where "small" means no more than m conjuncts if the fan-in of the $\widehat{f}$-gates in Φ is at most m.

Lemma 16 (Same-length Checking)**.** *For every language $L := \{L_m\}_{m \in \mathbb{N}}$, there is a same-length checker V that reduces $+\mathrm{ASAT}^L$ to the task of verifying, in parallel, multiple claims regarding $\widehat{L}$.*

In particular, $+_k\mathrm{ASAT}^{L_{\leq m}}$ gets reduced to at most m claims of the form '$\widehat{L}_i^k(y) = v$' where $i \leq m$.

This also holds if we extend the standard Boolean basis with $\mathcal{O}$, and give the checker access to $\mathcal{A}$. The checker does not depend on the choice of $\mathcal{A}$.

Proof. The checker V is to verify that the ℓ^{th} bit of $\Sigma_x \Phi(x)$ equals b, given (Φ, ℓ, b); here Φ has all its constants in $\mathbb{F}_{2^k}$ and hence the sum is over $\mathbb{F}_{2^k}$. V works as follows:

First, it obtains the claimed values for the rest of the k bits for $\Sigma_x\Phi(x)$, so that the claim becomes '$\Sigma_x\Phi(x) = u$' for some $u \in \mathbb{F}_{2^k}$.

Second, it performs the sumcheck protocol [8, Section 3.2] over $\mathbb{F}_{2^k}$ to get rid of the sum and update the claim to '$\Phi(y) = v$' for some y, v over the same range as that of x, u. (Note: field remains the same.)

At this point, V obtains the value of each gate in the evaluation of $\Phi(y)$ — i.e., the value of each subformula of Φ, when evaluating Φ on y — and checks all of them except those for $\widehat{L}$.

Finally, V uses the interpolation technique from the LFKN protocol [26], and combines multiple claims of the form '$\widehat{L}_i(z) = w$' into a single one, for each distinct i. More precisely, to merge two claims "$\widehat{L}_i(\vec{\alpha}) = v$" and "$\widehat{L}_i(\vec{\beta}) = w$", consider the line t passing through $\vec{\alpha}$ and $\vec{\beta}$ (i.e., $t(x) = \vec{\alpha} + (\vec{\beta} - \vec{\alpha})x \in \mathbb{F}^i_{2^k}[x]$), and rewrite the claims to be merged as "$\widehat{L} \circ t(0) = v$" and "$\widehat{L} \circ t(1) = w$". The checker obtains the univariate polynomial $\widehat{L} \circ t$ — or rather the polynomial $\underline{\widehat{L} \circ t}$ purported to be $\widehat{L} \circ t$ — and sets the merged claim as "$\widehat{L}_i(\gamma) = y$" where $\vec{\gamma} = t(\rho)$ and $y = \underline{\widehat{L} \circ t}(\rho)$ for a randomly chosen $\rho \in \mathbb{F}_{2^k}$.

The analysis of the protocol is standard: if the original claim, that the ℓ^{th} bit of $\Sigma_x\Phi(x)$ equals b, is false, where Φ has $\leq n$ nodes, then the sumcheck stage erroneously yields a true claim with probability at most

$$\#\text{ of rounds} \cdot \deg \Phi \;/\; \text{size of the field}$$

which grows slower than $1/n^d$ for any d, due to the requirement $k \geq \log^2 n$ in the definition of $+\text{ASAT}$ (Definition 11). Similarly, given $\leq n$ claims of the form '$\widehat{L}_i(z) = w$', if one of them is false, then the merging stage erroneously yields a correct conjunction of claims with probability at most

$$\#\text{ of merges} \cdot \deg R \;/\; \text{size of the field}$$

which again grows slower than $1/n^d$ for any d.

Since each input node of Φ is represented with a string of length $\geq k$ (Definition 11), the checker can maintain its queries to be of the same length.

Finally, the checker does not depend on the choice of $\mathcal{A}$. This finishes the proof. $\square$

Before we finally prove Theorems 2-3, let us note one consequence of what is done in Steps 1-2:

Corollary 17 (Extension Closure)**.** $\oplus\text{SAT}^{\widetilde{f}} \to \oplus\text{SAT}^f$. *The reduction works over any basis for formulas.*

Proof. Being the affine extension of f, by the definitions in Section 2, on input x, $\widetilde{f}$ gives the z^{th} bit of the value $\widehat{f}$ takes at y, where y and z are computable in polynomial-time out of x. In other words, $\widetilde{f}$ gives the $+\text{ASAT}^f$ instance (Φ, z) where Φ is the formula '$\widehat{f}(y)$'. Thus $\widetilde{f} \to +\text{ASAT}^f$. Combining with Lemma 14 gives $\widetilde{f} \to \oplus\text{SAT}^f$. Therefore,

$$\oplus\text{SAT}^{\widetilde{f}} \to \oplus\text{SAT}^{\oplus\text{SAT}^f} \to \oplus^*\text{SAT}^f \to \oplus\text{SAT}^f$$

by Proposition 4, Lemma 8, and Corollary 15, respectively. $\square$

Proof of Theorem 2. Extend the standard Boolean basis with $\mathcal{A}$. We are to show that there is a language K that is equivalent to $\oplus\text{SAT}$ under Karp reductions. Further, K must be same-length checkable given access to $\mathcal{A}$, and neither the checker nor the reductions to and from $\oplus\text{SAT}$ can depend on the choice of $\mathcal{A}$.

Recall that $\oplus\text{SAT}$ with respect to the (now-extended) standard basis is the same as $\oplus\text{SAT}^{\mathcal{A}}$ with respect to the basis $\{0, 1, \wedge, \oplus\}$. In the rest of the proof we will work in the latter basis for $\oplus\text{SAT}$, and its corresponding arithmetic basis for $+\text{ASAT}$.

We claim that $K := +\text{ASAT}^{\mathcal{O}}$ is a language as desired.

We begin by showing that K and $\oplus\mathrm{SAT}^{\mathcal{A}}$ reduce to each other. In one direction we have

$$K \to \oplus\mathrm{SAT}^{\mathcal{O}} \to \oplus\mathrm{SAT}^{\mathcal{A}},$$

where the first reduction is by Lemma 14 (with the $\mathcal{O}$-extended basis for $\oplus$SAT), and the second by using Proposition 4 and the fact that $\mathcal{O}$ Karp-reduces to its affine extension $\mathcal{A}$. For the other direction, do the same sequence of reductions in reverse, by using first Corollary 17 and then Lemma 12 (again with the $\mathcal{O}$-extended basis for Boolean formulas). Since none of the reductions used depend on the choice of a Boolean basis, neither does their composition in either direction.

Next, the same-length checkability of K given access to $\mathcal{A}$, is immediate from Lemma 16 by setting L to the empty language $L : x \mapsto 0$ and the basis for $+\mathrm{ASAT}$ to be the standard arithmetic basis. By the same lemma, the checker does not depend on the choice of $\mathcal{A}$.

Finally, $\oplus\mathrm{SAT}^{\mathcal{A}}$ is checkable: On input x, reduce it to an input x' for K, then simulate the checking protocol for $K(x')$, by reducing each query for K to one for $\oplus\mathrm{SAT}^{\mathcal{A}}$. Because the reductions in either direction do not depend on the choice of a Boolean basis, and because the checker for K does not depend on the choice of $\mathcal{A}$, the same holds for the checker for $\oplus\mathrm{SAT}^{\mathcal{A}}$. $\square$

Proof of Theorem 3. Extend the standard Boolean basis with $\mathcal{A}$. Let L be a language. We are to show that there is an interactive protocol yielding the reduction

$$\oplus\mathrm{SAT}_n^{L_{\le m}} \to \oplus\mathrm{SAT}_{\mathrm{poly}(m \log n)}^{L_{\le m}},$$

for every n, m. The protocol will have access to $\mathcal{A}$ but will not depend on the choice of $\mathcal{A}$.

Recall that $\oplus$SAT with respect to the (now-extended) standard basis is the same as $\oplus\mathrm{SAT}^{\mathcal{A}}$ with respect to the basis $\{0, 1, \wedge, \oplus\}$. In the rest of the proof we will work in the latter basis for $\oplus$SAT, and its corresponding arithmetic basis for $+\mathrm{ASAT}$.

Also recall that we use $\oplus\mathrm{SAT}^{f,g}$ to refer to either of $(\oplus\mathrm{SAT}^{f})^{g}$ and $(\oplus\mathrm{SAT}^{g})^{f}$ depending on context, as they are equivalent under Karp reductions.

We proceed with the proof. We have

$$\oplus\mathrm{SAT}^{\mathcal{A},L_{\le m}} \to \oplus\mathrm{SAT}^{\mathcal{O},L_{\le m}} \to +\mathrm{ASAT}^{\mathcal{O},L_{\le m}},$$

where the first reduction is by Corollary 17 (with the $L_{\le m}$-extended basis for $\oplus$SAT) and the second by Lemma 12 (with the $\mathcal{O}$- and $L_{\le m}$-extended basis for $\oplus$SAT)). In fact, the same sequence yields

$$\oplus\mathrm{SAT}_n^{\mathcal{A},L_{\le m}} \to +_k\mathrm{ASAT}^{\mathcal{O},L_{\le m}},$$

for some $k \in \mathrm{poly}\log n$.

Now, Lemma 16 says (with the $\mathcal{O}$-extended basis) that $+_k\mathrm{ASAT}^{\mathcal{O},L_{\le m}}$ is reducible, via an interactive protocol that has access to $\mathcal{A}$, to the conjunction of at most m claims regarding $\widehat{L}_i^k$, $i \le m$, and therefore to the conjunction of at most mk claims regarding $\widetilde{L}_i^k$, the Boolean version of $\widehat{L}_i^k$. Altogether these claims can be expressed as one Boolean formula, hence as one $\oplus$SAT instance, of size $\mathrm{poly}(mk)$, over the basis extended with the affine extension of $L_{\le m}$. In notation,

$$+_k\mathrm{ASAT}^{\mathcal{O},L_{\le m}} \to \oplus\mathrm{SAT}_{\mathrm{poly}(mk)}^{\widetilde{L}_{\le m}}.$$

Finally, Corollary 17 says, with the setting $\mathcal{O} = L_{\le m}$, that

$$\oplus\mathrm{SAT}^{\widetilde{L}_{\le m}} \to \oplus\mathrm{SAT}^{L_{\le m}}$$

completing the desired reduction.

All the Karp reductions above work over any choice of basis, in particular, of $\mathcal{A}$ and m. The interactive reduction of Lemma 16 also does not depend on the choice of $\mathcal{A}$ and m. This completes the proof. $\square$

3.2 The IP Theorem

In this section we show that Shamir's IP theorem, $\mathrm{PSPACE} \subset \mathrm{IP}$, admits an affinely relativizing proof. As a byproduct we obtain a new streamlined proof of this result; see Section 1.3 for an overview and comparison with previous proofs.

The proof is a straightforward consequence of the results in Section 3.1 on $\oplus\mathrm{SAT}$. We show:

Theorem 18. *Every downward-self-reducible language is computable by an interactive protocol.*

This also holds if the standard Boolean basis is extended with $\mathcal{A}$, via a protocol that has access to $\mathcal{A}$. The protocol does not depend on the choice of $\mathcal{A}$ if the self-reduction does not.

Proof. Extend the standard Boolean basis with $\mathcal{A}$. Let $L := \{L_n\}_{n\in\mathbb{N}}$ be downward-self-reducible, so that there is a function in FP that induces the Cook-reduction from L_n to $L_{\leq n-1}$ for every $n > 0$. By Proposition 4, there is a function in FP that induces the reduction

$$\oplus\mathrm{SAT}^{L_n} \to \oplus\mathrm{SAT}^{L_{n-1}}$$

for every $n \in \mathbb{N}$; here and throughout the rest of the proof, $\oplus\mathrm{SAT}^{L_{-1}}$ denotes $\oplus\mathrm{SAT}$, and L_i denotes $L_{\leq i}$.

Iterating this and combining it with Theorem 3, we get a function in FP and an interactive protocol that together induce the two-step reduction

$$(\oplus\mathrm{SAT}^{L_{n-1}})_{n^d} \to (\oplus\mathrm{SAT}^{L_{n-2}})_{n^{d'}} \to (\oplus\mathrm{SAT}^{L_{n-2}})_{n^d}, \tag{3}$$

for every n, for some large enough constants d, d' where n^i denotes $n^i + i$ (in particular d must exceed the exponent hidden in the $\mathrm{poly}(\cdot)$ notation of Theorem 3). We also have the trivial reduction

$$L_n \to \oplus\mathrm{SAT}^{L_n} \tag{4}$$

since the task of computing $L(\alpha)$ reduces to the task of evaluating the formula '$L(\alpha)$'.

Now consider the reduction that on input x to L_n, first applies the reduction in (4), and then for n iterations, applies the reduction sequence in (3). This compound reduction yields

$$L_n \to \oplus\mathrm{SAT}$$

for every $n \in \mathbb{N}$, in other words, it yields $L \to \oplus\mathrm{SAT}$.

By Theorem 2 on checking $\oplus\mathrm{SAT}$, it follows that L is computable by a protocol that has access to $\mathcal{A}$ but does not depend on the choice of $\mathcal{A}$. □

We now state Shamir's theorem as a gap-amplification result, as explained on page 12.

Definition 19 (γ-gap-IP)**.** Say that a language L is in γ-gap-IP iff there is an interactive protocol for L with completeness 1 and soundness $< 1 - \gamma$. Here γ can be any function from $\mathbb{N}$ to $[0, 1) \subset \mathbb{R}$.

By its very definition, the class 0-gap-IP has TQBF as a complete language, because the interaction between the verifier and prover in a 0-gap-IP protocol can be expressed as a TQBF instance. Hence:

Proposition 20. *There is a downward-self-reducible language that is complete for* 0-gap-IP. *This also holds if the standard Boolean basis is extended with $\mathcal{O}$. The reductions do not depend on the choice of $\mathcal{O}$.*

Remark. Let $\mathcal{PSPACE}$ be the class polynomial-space defined in the classical sense using Turing machines, and let $\mathcal{PSPACE}^{\mathcal{O}}$ be its relativized variant in the sense of Baker-Gill-Solovay. The well-known result of Stockmeyer and Meyer [33], showing that the language TQBF is complete for $\mathcal{PSPACE}$, in fact can be restated as saying that $\mathcal{PSPACE} \subset 0$-gap-IP. More generally, $\mathcal{PSPACE}^{\mathcal{O}} \subset 0$-gap-IP with respect to the $\mathcal{O}$-extended Boolean basis.

Corollary 21 (IP theorem)**.** 0-gap-IP $\subset \Omega(1)$-gap-IP. *This also holds if the standard Boolean basis is extended with $\mathcal{A}$. The correspondence between the* 0-gap-IP *and the* $\Omega(1)$-gap-IP *protocols does not depend on the choice of $\mathcal{A}$.*

So for every 0-gap-IP-protocol Π_0 under the $\mathcal{A}$-extended basis, there is a $\Omega(1)$-gap-IP-protocol $\Pi_{\Omega(1)}$ that not only computes the same language as Π_0, but continues to do so even if $\mathcal{A}$ is changed to be some other affine extension language.

3.3 The MIP Theorem

In this section we show that the $\mathrm{NEXP} \subset \mathrm{MIP}$ theorem of Babai, Fortnow, and Lund [8] admits an affinely relativizing proof, if it is viewed as a gap amplification result as explained on page 12. We show:

Definition 22 (γ-gap-MIP)**.** Say that a language L is in γ-gap-MIP iff there is a multiple-prover interactive protocol for L with completeness 1 and soundness $< 1-\gamma$. Here γ can be any function from $\mathbb{N}$ to $[0,1) \subset \mathbb{R}$.

Theorem 23 (MIP theorem)**.** 0-gap-MIP $\subset \Omega(1)$-gap-MIP. *This also holds if the standard Boolean basis is extended with $\mathcal{A}$, via protocols with access to $\mathcal{A}$.*

The proof becomes a straightforward consequence of Section 3.2, that the IP theorem affinely relativizes, once two ingredients are introduced. First is a very useful characterization of MIP, and more generally of γ-gap-MIP, due to Fortnow, Rompel, and Sipser [14]. We paraphrase their result:

Fact 24. *$L \in \mathrm{MIP}$ iff there is a language π such that $L \in \mathrm{IP}^{\pi}$, with a protocol that is robust to its oracle π in the following sense: if some other oracle π^* is used instead of π then no prover strategy can exploit this, i.e., the verifier cannot be convinced to accept x with probability $\geq 1/3$ whenever $L(x) = 0$, even if some other π^* is used as oracle instead of π.*

This equivalence more generally holds if "MIP" *is replaced with* "γ-gap-MIP", "IP" *with* "γ-gap-IP", *and* "$1/3$" *with* "$1-\gamma$".

The above also holds when the standard Boolean basis is extended with $\mathcal{O}$, and all the protocols involved are given (additional) access to $\mathcal{O}$.

The second ingredient in proving Theorem 23, and the key one, is the seminal "multi-linearity test" of Babai, Fortnow, Lund [8, Thm 5.13]. We combine it with a "Booleanness test" from the same paper [8, §7.1] and a standard decoding procedure for low-degree polynomials (e.g., [4, §7.2.2]):

Proposition 25. *There is an interactive protocol, $Decode$, for which the following holds. Let $[\mathcal{E}]$ denote the maximum probability that event $\mathcal{E}$ occurs, over all possible prover strategies in the protocol.*

For every language F and every number $N \in \mathbb{N}$, there is some affine extension language G such that: given inputs (x, ε, N) where $|x| \leq N$ and $\varepsilon \leq 1/2$, and given oracle access to F, $Decode$ runs in time $\mathrm{poly}(N/\varepsilon)$, and outputs a value $output$ such that:

(i) *if F is an affine extension, then $[output = F(x)] = 1$,*

(ii) *if F is an affine extension, then $[output \notin \{F(x), \text{'fail'}\}] = 0$,*

(iii) *otherwise, $[output \notin \{G(x), \text{'fail'}\}] \leq \varepsilon$.*

We defer the proof to the end of this section (§3.3.2) and proceed to derive Theorem 23.

Proof of Theorem 23. Let us adopt the notation of Proposition 25, and use $[V(x)]$ to denote the maximum probability that a verifier V accepts its input x, over all possible prover strategies in a protocol.

Extend the standard Boolean basis with $\mathcal{A}$. Let $L \in 0\text{-gap-MIP}$. By Fact 24, $L \in 0\text{-gap-IP}^\pi$ for some language π; we may assume π is an affine extension since every language reduces to its affine extension. Pick any protocol realizing $L \in 0\text{-gap-IP}^\pi$, and let its verifier be V_0. We know that this protocol is robust to its oracle π in the sense of Fact 24.

By Corollary 21, $L \in \text{IP}^\pi$.[9] Therefore, by Fact 24, all that remains to show is that among the protocols realizing $L \in \text{IP}^\pi$, one is robust to π. So pick any protocol realizing $L \in \text{IP}^\pi$, with verifier V say. We may assume that the soundness error of this protocol is $< 1/6$ by amplification.

Consider modifying V, so that it performs each of its oracle queries, say to $\pi(X)$, via the protocol of Proposition 25, as $Decode^\pi(X, \varepsilon, t)$, and rejects upon failure; here $t := t(|x|)$ is the total running time of V on an input of length $|x|$, and ε will be worked out later. By Proposition 25-(i)-(ii), and by the assumption that π is an affine extension, this modification does not affect the outcome of the protocol when π is used as oracle, so we still have a IP^π-protocol for L.

We claim that this new protocol has the desired robustness; in the notation introduced up front,

$$[V \circ Dec^{\pi^*}(x)] < 1/3 \qquad (\dagger)$$

for every language π^* and every x such that $L(x) = 0$, where $V \circ Dec$ denotes the modified verifier.

To see this, let $L(x) = 0$ and let π^* be any language. Depending on whether or not π^* is an affine extension, respectively, let H denote either π^*, or the language G obtained from Proposition 25 by putting $F := \pi^*$.

Starting from the very first protocol we mentioned for L, i.e. the one with the verifier V_0, and going toward the last one with the verifier $V \circ Dec$, we will now argue the validity of four implications

$$L(x) = 0 \implies [V_0^H(x)] < 1 \implies [V^H(x)] < 1/6 \implies [V \circ Dec^H(x)] < 1/6 \implies (\dagger)$$

which will prove the theorem.

The first implication holds no matter what H is, because V_0 is the verifier of a 0-gap-IP^π-protocol for L that is robust to π in the sense of Fact 24.

For the second implication, we know that V^π agrees with V_0^π on x by Corollary 21. Moreover, by the second part of by Corollary 21, V^H agrees with V_0^H on x, and hence accepts x with probability $< 1/6$.

The third implication is by H being an affine extension, and hence by $Decode^H$ working the same as H, by Proposition 25-(i)-(ii).

The last implication is trivial if $H = \pi^*$, so suppose not. Then by Proposition 25, except with probability $\le \varepsilon$, $Decode^{\pi^*}$ returns either H or 'fail'. This means, recalling that t bounds the running time of V on x, except with probability $\le t\varepsilon$, $V \circ Dec^{\pi^*}$ either works the same as $V \circ Dec^H$, or it rejects because $Decode^{\pi^*}$ returns "fail" at some point. Therefore, $[V \circ Dec^{\pi^*}(x)] \le [V \circ Dec^H(x)] + t\varepsilon$. To finish, set $\varepsilon \le 1/(6t)$. $\square$

3.3.1 Comparison with the standard view

We now take up the discussion on page 12, that relates the gap amplification view of the MIP theorem, Theorem 23, to the standard view, $\text{NEXP} \subset \text{MIP}$:

Proposition 26. $\text{NEXP} \subset 0\text{-gap-MIP}$. *This does* not always *hold if the standard Boolean basis is extended with some $\mathcal{A}$.*

[9]When the Boolean basis is extended with $\mathcal{A}$, notice that 0-gap-IP^π really involves two affine oracles not just one. We are still justified in invoking Corollary 21, however; see the discussion in page 6 titled "multiple oracles".

In order to make transparent what "current technique" yields it, we prove Proposition 26 in two steps. First, we characterize 0-gap-MIP as a subclass of NEXP, namely as those languages in NEXP with "strong locality". Then we show that the strong Cook-Levin theorem collapses NEXP to that subclass.

Definition 27 (strong uniformity)**.** Call a family of circuits $\{C_n\}_{n\in\mathbb{N}}$ strongly uniform iff the function that outputs, given input (n, i), the type of the i^{th} gate in C_n, as well as the indices of all gates connected to it, is in FP.

Definition 28 ($\text{NEXP}_{\text{local}}$)**.** Let P_{local} be the class of all languages with strongly uniform polynomial-size circuits. Using this class define NP_{local}, and then by padding define $\text{NEXP}_{\text{local}}$; in other words let $\text{NEXP}_{\text{local}}$ be the class of all languages with strongly uniform exponential-size nondeterministic circuits.

Proposition 29. $0\text{-gap-MIP} = \text{NEXP}_{\text{local}}$. *This also holds if the standard Boolean basis is extended with $\mathcal{O}$, and the protocols are given access to $\mathcal{O}$.*

Remark. Over an arbitrary basis extension $\mathcal{O}$, the class $\text{NEXP}_{\text{local}}$ corresponds exactly to what we may denote as $\mathcal{NEXP}^{\mathcal{O}[\text{poly}]}$, the Turing-machine-based definition of relativized NEXP where the oracle queries are restricted to be of polynomial length. In logical terms (Section 1.1), in every standard model of $\mathcal{ACT}$, the variable $\text{NEXP}_{\text{local}}$ gets interpreted as the class $\mathcal{NEXP}^{\mathcal{O}[\text{poly}]}$ for some $\mathcal{O}$, and conversely for every $\mathcal{O}$, there is some standard model of $\mathcal{ACT}$ in which $\text{NEXP}_{\text{local}}$ is interpreted as $\mathcal{NEXP}^{\mathcal{O}[\text{poly}]}$. Proposition 29 thus vindicates the use of poly-length query restriction in previous work; see the discussion on page 12.

We now proceed to prove Propositions 29 & 26.

Proof of Proposition 29. Part ($\subset$): Extend the standard Boolean basis with $\mathcal{O}$. Let $L \in 0$-gap-MIP. By Fact 24, $L \in 0\text{-gap-IP}^{\pi}$ for some language π. In other words, there is a family of size-$\text{poly}\, n$ circuits $V^{\pi} := \{V_n^{\pi}(x, r)\}_n$ over the π-extended basis (this extension is in addition to $\mathcal{O}$) such that $L(x) = 1$ iff

$$\bigwedge_{r_1} \bigvee_{r_2} \cdots \bigvee_{r_{|r|}} V_n^{\pi}(x, r) \tag{5}$$

evaluates to 1, where each r_i ranges over $\{0, 1\}$, $|r| \in \text{poly}\, n$ (wlog $|r|$ is even), and $n = |x|$.

View (5) as a circuit $D_n^{\pi}(x)$, of size $O(2^{|r|})$ times the size of V_n^{π}. Now view D_n^{π} as a circuit $C_n(x, y_{\pi})$ over the *standard* basis (we still have $\mathcal{O}$ in the basis), by replacing each π-gate in D with a y_{π}-gate.

V^{π} is a uniform family, i.e., there is a function in FP that outputs, given input $(1^n, 1^i)$, information about the i^{th} gate of V_n^{π} in the sense of Definition 27. It follows that $C := \{C_n(x, y_{\pi})\}_n$ is a strongly-uniform family. This proves the first part.

Part ($\supset$): Extend the standard Boolean basis with $\mathcal{O}$. Let $L \in \text{NEXP}_{\text{local}}$ with a corresponding strongly uniform circuit family $\{C_n(x, y)\}_n$ of size $s(n) \in \exp\text{poly}\, n$. For every $x \in \{0, 1\}^n$ and $i \in \{0, 1\}^{\log s(n)}$, let $\pi(x, i)$ be the value of the i^{th} gate in $C_n(x, y^*)$ for some fixed y^* maximizing the output of $C_n(x, y)$ over all eligible y. Then in the protocol for $L(x)$, the verifier V simply: (i) picks at random $i \in \{0, 1\}^{\log s(n)}$; (ii) using the strong uniformity of $\{C_n\}$, finds out that the i^{th} gate is, say, of type f and is connected, say, to gates $i_1..i_m$ in that order; and (iii) checks that the transcript is consistent with the i^{th} gate, i.e., that $z = f(z_1..z_m)$, where z stands for $\pi(x, i)$ in general, with the special case being when gate i is the output gate (then $z = 1$), and where z_k stands for $\pi(x, i_k)$ in general, with the special case being when gate i_k is an input gate (then $z_k = x_j$ for an appropriate j). It is easy to see the robustness of this protocol to π. The claim follows. $\square$

Proof of Proposition 26. $\text{P} \subset \text{P}_{\text{local}}$ by the strong Cook-Levin theorem (Section 2), implying $\text{NEXP} \subset \text{NEXP}_{\text{local}}$ and proving the first claim via Proposition 29.

For the second claim, we want to show a language $\mathcal{O}$ with its affine extension $\mathcal{A}$, such that $\text{NEXP} \not\subset \text{NEXP}_{\text{local}}$ with respect to the $\mathcal{A}$-extended basis. It actually suffices to show this non-containment with respect to the $\mathcal{O}$-extended basis, because $\mathcal{A}$ reduces to $\oplus\text{SAT}^{\mathcal{O}}$ by Corollary 17, and because $\oplus\text{SAT}$ can be computed brute-force in $\text{NEXP}_{\text{local}}$, the latter holding also under the $\mathcal{O}$-extended basis. (To see this, let $\text{FmlaEval}(\phi, x)$ be the language that interprets ϕ as a formula and evaluates ϕ at x. Then $\oplus\text{SAT}(\phi(x)) = \oplus_x C(\phi, x)$, where C is the circuit for FmlaEval at the appropriate input length. View C as a circuit D of size $2^{|x|}$ times the size of C. Because C is uniform, D is strongly uniform, so $\oplus\text{SAT} \in \text{EXP}_{\text{local}}$ in fact.)

Thus $\text{NEXP}_{\text{local}}$ does not become smaller if $\mathcal{O}$ is used as the basis extension instead of $\mathcal{A}$ — it does not matter whether NEXP does — and what remains is to construct $\mathcal{O}$. But this is an easy matter: let $M_1, M_2, M_3..$ be a list (repetitions allowed) of all nondeterministic algorithms with access to an arbitrary language $\mathcal{O}$, such that M_i runs in time $\leq 2^{n^{\log n}}$ on all inputs of length $n \geq i$, and such that all queries to $\mathcal{O}$ are of length $\leq n^{\log n}$. Now for $i = 1..\infty$, update $\mathcal{O}$ at a large enough input, say on 1^{2^i}, to the output of M_i on 1^i. Every language in $\text{NEXP}_{\text{local}}$ with respect to the $\mathcal{O}$-extended basis is computed by some M_i in the list, yet the language mapping $1^i \mapsto \neg\mathcal{O}(1^{2^i})$ is clearly not, proving $\text{E} \not\subset \text{NEXP}_{\text{local}}$ with respect to that $\mathcal{O}$. $\square$

3.3.2 Proof of Proposition 25

We complete Section 3.3 by providing the deferred proof of Proposition 25.

— Proof of Proposition 25 : The protocol Decode —

Let L be a language. Being the affine extension of L, by the definitions in Section 2, on input $x \in \{0,1\}^n$, $\widetilde{L}$ gives the z^{th} bit of the value $\widehat{L}$ takes at $y \in \mathbb{F}_{2^k}^m$, i.e.,

$$\left(\widehat{L}_m^k(y)\right)_z \tag{6}$$

where y, z, m, k are all computable in polynomial-time out of x, and are all $\leq n$. Conversely, given (y, z, m, k), an input x for which this holds is also computable in polynomial-time.

The protocol Decode interprets its input x as though it were for some $\widetilde{L}$, and extracts y, z, m, k. Since k denotes the field size, or the logarithm thereof, and since $\mathbb{F}_{2^k}$ can be efficiently identified in $\mathbb{F}_{2^{\geq k}}$, and moreover, since $k \leq n \leq N$, (6) can be viewed as

$$\left(\widehat{L}_m^N(Y)\right)_Z \tag{7}$$

where Y denotes y identified in $\mathbb{F}_{2^N}^m$, and Z denotes the accordingly updated z.

Owing to this, Decode overrides (y, z, m, k) with (Y, Z, m, N), and x with some X corresponding to the latter tuple. For notational convenience, we will not capitalize Y and Z because they represent the same information as y and z. Also, we will assume that $N \geq 10 \log 2m$; this is without loss of generality as we can always have Decode increase N before overriding (y, z, m, k).

After this initial adjustment phase, Decode proceeds into the main phase consisting of three steps:

Step 1. Test if $F_{m,N}$ is (multi-)affine:

- Pick at random an axis-parallel line, and three points on this line.
- Check if the values $F_{m,N}$ takes on the three points are collinear.

Step 2. Test if $F_{m,N}$ is an affine *extension*, i.e., if $F_{m,N}$ is Boolean on Boolean inputs:

- Pick at random up to m Boolean vectors $v_1, .., v_i \in \mathbb{F}_{2^N}^m$.

- Do a sumcheck protocol on the claim $0 = \sum_{b\in\{0,1\}^m} Q(b)$ where
$$Q(y) := F_{m,N}(y)(1 + F_{m,N}(y)) \prod_{j=1..i}(1 + \langle y, v_j\rangle) \tag{8}$$
with $\langle y, w\rangle$ denoting the inner product $\sum_\ell y_\ell w_\ell$.
- Save the point y' at which Q is evaluated at the last round of sumcheck.

Step 3. Test for consistency:

- Pick at random a line ℓ originating at y, i.e., let $\ell(t) := y + th$ for a random $h \in \mathbb{F}_{2^N}^m \setminus \{0\}$.
- Letting $(1), .., (m+1)$ be a canonical choice of $m+1$ distinct nonzero elements of $\mathbb{F}_{2^N}$, interpolate into a polynomial $q(t)$ the values of $F_{m,N}$ at $\ell((1)), .., \ell((m + 1))$
- Check if $F_{m,N}(y) = q(0)$. Also check $F_{m,N}(y_*) = q(t_*)$ for a random t_* and $y_* := \ell(t_*)$.
- Repeat Step 3 also for the point y' saved in Step 2, in place of y.

We refer to [8, §3.2 and §5] for explanations of the terms 'axis parallel line', 'sumcheck protocol', etc.

If any of the checks fails, then $Decode$ outputs 'fail'; otherwise, it repeats Step 1 - 3. This goes on for T times, after which point $Decode$ outputs $F(X)$, i.e., the z^{th} bit of $F_{m,N}(y)$. With foresight, we set $T := cm^9 \ln \frac{1}{\varepsilon}$, where $c = 8100$. This completes the description of the protocol.

— Proof of Proposition 25 : Analysis of $Decode$ —

To begin with, note that the protocol never outputs something besides $F(X)$ or 'fail'. Thus
$$\text{for every prover, } \Pr[output \notin \{F(X), \text{'fail'}\}] = 0. \tag{9}$$

Parts (i) and (ii) of the claim are fairly immediate. Indeed, suppose $F = \widetilde{L}$ for some language L. Then all steps succeed with certainty, provided the prover acts honestly in Step 2. Thus
$$\text{for some prover, } \Pr[output = F(X)] = 1.$$
Also, $F(x) = F(X)$ in this case, because
$$F(x) = \widetilde{L}(x) = \widetilde{L}_m^{\,k}(yz) = \widetilde{L}_m^{\,N}(yz) = \widetilde{L}(X) = F(X).$$
Putting together with (9), we get claims (i)-(ii).

To proceed with part (iii), let us introduce a piece of notation. For functions f, g with the same finite domain, say f is *nearby* g, and write $f \approx g$, to mean
$$\Pr_y[f(y) \neq g(y)] \leq \gamma$$
over the uniform choice of y from $\operatorname{dom} f = \operatorname{dom} g$. With foresight, we set $\gamma := \frac{1}{100m^2}$.

Also, let us call a function $P : \mathbb{F}_{2^N}^m \to \mathbb{F}_{2^N}$ affine, or m-affine, if it is the evaluation in $\mathbb{F}_{2^N}$ of an m-variate polynomial over $\mathbb{F}_{2^N}$ with individual degree ≤ 1.

For all functions $f : \mathbb{F}_{2^N}^m \to \mathbb{F}_{2^N}$ and all m, there can be at most one affine function nearby f, due to the Schwartz-Zippel Lemma (see, e.g., [4, Lemma 4.2]) and the fact that $m \leq N$.

We now define the language G claimed to satisfy part (iii). For every $m \in \mathbb{N}$, consider whether there exists a Boolean function $L_m : \{0,1\}^m \to \{0,1\}$ such that $\widehat{L}_m^{\,N}$ is nearby $F_{m,N}$. If yes, then L_m is unique by the previous paragraph; if not, then let L_m be arbitrary, say the map $x \in \{0,1\}^m \mapsto 0$. Then set $G := \widetilde{L}$.

Now consider three cases:

case i. $F_{m,N}$ is not nearby any affine function $P : \mathbb{F}_{2^N}^m \to \mathbb{F}_{2^N}$. In this case, by [8, Thm 5.13 and §7.1],[10]

$$\Pr[\text{Step 1 passes}] \le 1 - 1/(8100m^9). \tag{i}$$

case ii. $F_{m,N}$ is nearby $\widehat{L}_m^N$. Suppose $F_{m,N}$ disagrees with $\widehat{L}_m^N$ on y. Then by [4, Proposition 7.2.2.1],[11]

$$\begin{aligned}&\Pr[\text{Step 3 passes}]\\ &\le \Pr[\text{the interpolated value of } F_{m,N} \text{ agrees with } F_{m,N} \text{ at } y_*\text{, but disagrees with } \widehat{L}_m^N \text{ at } y]\\ &\le 2\sqrt{\gamma} + m/(2^N-1) \le 2/(9m).\end{aligned} \tag{ii}$$

Now suppose $F_{m,N}$ does agree with $\widehat{L}_m^N$ on y. Then $F(X) = G(X)$ by the way we defined G. Further, $G(X) = G(x)$ since

$$G(x) = \widetilde{L}(x) = \widetilde{L}_m^k(yz) = \widetilde{L}_m^N(yz) = \widetilde{L}(X) = G(X).$$

Putting together with (9) we get, for every prover,

$$\Pr[\mathit{output} \notin \{G(x), \text{‘fail’}\}] = 0. \tag{ii'}$$

case iii. $F_{m,N}$ is not nearby $\widehat{L}_m^N$, but is nearby some affine function $P : \mathbb{F}_{2^k}^m \to \mathbb{F}_{2^k}$. In this case, by the way we defined L_m, we know that P is not Boolean on all Boolean inputs, i.e., $P(\mathbb{F}_2^m) \not\subseteq \mathbb{F}_2$.

Consider Step 2, in particular, the randomly picked point $y' \in \mathbb{F}_{2^N}^m$ on which the expression Q in (8) is evaluated at the end of the sumcheck protocol. Let $\mathcal{E}$ be the event that $F_{m,N}$ agrees with P on this point y'. Then just as in case ii., by [4, Proposition 7.2.2.1],

$$\begin{aligned}&\Pr[\text{Step 3 passes} \mid \neg\mathcal{E}]\\ &\le \Pr[\text{the interpolated value of } F_{m,N} \text{ agrees with } F_{m,N} \text{ at } y'_*\text{, but disagrees with } P \text{ at } y']\\ &\le 2/(9m).\end{aligned}$$

On the other hand, suppose $\mathcal{E}$. Then at the end of sumcheck, the final claim, of the form

$$\text{‘}v = Q(y')\text{’} \quad \text{where} \quad Q(y) := F_{m,N}(y)(1 + F_{m,N}(y)) \textstyle\prod_{j=1..i}(1 + \langle y, v_j\rangle)$$

for some $v \in \mathbb{F}_{2^N}$, can be alternately written as

$$\text{‘}v = Q_P(y')\text{’} \quad \text{where} \quad Q_P(y) := P(y)(1 + P(y)) \textstyle\prod_{j=1..i}(1 + \langle y, v_j\rangle).$$

Therefore, for every prover,

$$\begin{aligned}\Pr[\text{Step 2 passes} \mid \mathcal{E}] &\le \Pr[\text{‘}v = Q(y')\text{’ is a correct claim} \mid \mathcal{E}]\\ &= \Pr[\text{‘}v = Q_P(y')\text{’ is a correct claim} \mid \mathcal{E}] = (*)\end{aligned}$$

where we just switched from a sumcheck involving the initial claim ‘$0 = \sum_b Q(b)$’ to one involving the initial claim ‘$0 = \sum_b Q_P(b)$’. We are justified in this transition because v is a function purely of y' and the prover, with “the prover” being just a function from $\mathbb{F}^{\le m}$ to the univariate polynomials over $\mathbb{F}$ of degree $\deg Q$, that has nothing to do with the particulars of Q. Therefore

$$(*) \le \Pr[\text{penultimate claim in the sumcheck for ‘}0 = \textstyle\sum_b Q_P(b)\text{’ is correct}]$$

[10]To invoke [8, Theorem 5.13] we set $\epsilon = 1/(900m^4)$, $\delta = 1/(900m^4)$, and use $|\mathbb{F}_{2^N}| \ge 900m^4$. In return we get an axis $i \in \{1..m\}$ along which $\ge \epsilon$-fraction of lines would fail the test in Step 1 with $\ge \delta$-chance, provided that $F_{m,N}$ differs from every affine function on $\ge \epsilon'$-fraction of inputs, where $\epsilon' \le 1/(100m^2)$.

[11]To invoke [4, Proposition 7.2.2.1], we let $A := F_{m,N}$, and B be the function that given (y, h), outputs the polynomial $q(t)$ as described in the protocol. In return, we get that if there is a polynomial P of degree d such that $\Pr_y[f(y) \ne P(y)] = \varepsilon$, then $\Pr_{h,t_*}[A(y) \ne q(0) \text{ yet } A(\ell(t_*)) = q(t_*)\,] \le 2\sqrt{\varepsilon} + \frac{d}{|\mathbb{F}|-1}$.

$$+\ \Pr[\text{last sumcheck round errs }]$$
$$\leq \Pr[\text{the first claim `}0 = \textstyle\sum_b Q_P(b)\text{' is correct}] + m\rho$$

where we dropped $\mathcal{E}$ by independence, and where an erroneous round is one that takes an incorrect claim and produces a correct one, with ρ denoting the probability of such a round taking place and m being the number of rounds. Because each round involves evaluating a given univariate polynomial of degree $\leq \deg Q$ at a random point in $\mathbb{F}_{2^N}$,

$$\rho \leq \deg Q/2^N \leq m(m+2)/2^{10\log 2m} \leq 1/(500m^9).$$

Now, applying Rabin's Isolation Lemma [35, Theorem 2.4] to the set $B \subset \mathbb{F}_2^m : P(B) \not\subset \mathbb{F}_2$, we get

$$\Pr[\text{the first claim `}0 = \textstyle\sum_b Q_P(b)\text{' is correct}] \leq 1 - 1/(4m),$$

and putting together we get, for every prover,

$$\Pr[\text{all steps pass}] \leq \min(\,\Pr[\text{Step 2 passes} \mid \mathcal{E}]\,,\ \Pr[\text{Step 3 passes} \mid \neg\mathcal{E}]\,) \leq 1 - 1/10m. \qquad \text{(iii)}$$

This concludes the case analysis. Step 1 through Step 3 are repeated T times, which for our choice of T makes all of (i), (ii), and (iii) at most ε, thus

$$\Pr[\mathit{output} \notin \{G(x), \text{`fail'}\}] \leq (\max\{\text{(i),(ii),(iii)}\})^T + \text{(ii')} \leq \varepsilon$$

as claimed. □

3.4 Lower Bounds against General Boolean Circuits

Using the IP theorem and its variants, Buhrman, Fortnow, Thireauf [12] and Santhanam [28] succeeded in obtaining the strongest lower bounds known-to-date against general Boolean circuits. In both results, the lower bound is shown for the class of Merlin-Arthur protocols; in the case of Buhrman et al. it is for the class $\mathrm{MA}(\exp n)$, of protocols running in exponential time, and in Santhanam it is for $\mathrm{MA}(\mathrm{poly}\, n)$. For notational convenience, we use $\mathrm{MA}(t(n))$ to denote classes of partial languages, and make it explicit when we talk about the subclass of (total) languages.

In this section we give an affinely relativizing proof that unifies both results. We prove:

Theorem 30. *For every constant d,*

(i) $\mathrm{MA}(\exp n)$ contains a language that does not have circuits of size $O(2^{\log^d n})$.

(ii) $\mathrm{MA}(\mathrm{poly}\, n)$ contains a partial language that does not have circuits of size $O(n^d)$.

This also holds if the Boolean basis is extended with $\mathcal{A}$.

The proof consists of three main ingredients. The first one shows that if the lower bound fails to hold, then this failure scales to $\oplus\mathrm{SAT}$.

Lemma 31 (Scaling)**.**

(i) If part (i) of Theorem 30 is false, then $\oplus\mathrm{SAT}$ has circuits of size $O(2^{\log^d n})$ for some d.

(ii) If part (ii) of Theorem 30 is false, then $\oplus\mathrm{SAT}$ has circuits of size $O(n^d)$ for some d.

This also holds if the Boolean basis is extended with $\mathcal{A}$.

We defer the proof of Lemma 31 to the end of this section.

To proceed with the rest of the proof it will be convenient to introduce some notation.

Definition 32 ($\Sigma_3\mathrm{SAT}$)**.** Let $\Sigma_3\mathrm{SAT}$ denote the language mapping $\phi(x, y, z) \mapsto \exists x \forall y \exists z\, \phi(x, y, z)$ where ϕ is over the standard Boolean basis by default.

For t a well-behaved resource bound, let $\Sigma_3\mathrm{SAT}(t)$ denote the set of all languages Karp-reducible in time t to $\Sigma_3\mathrm{SAT}$. In other words, $\Sigma_3\mathrm{SAT}(t)$ is the set of all languages L for which the language $(x, 1^{t(|x|)}) \mapsto L(x)$ Karp-reduces to $\Sigma_3\mathrm{SAT}$.

The second ingredient in proving Theorem 30 is a collapse result: if the conclusion of the Scaling lemma holds, then the polynomial-time hierarchy collapses. We defer its proof to the end of this section.

Lemma 33 (Collapse)**.** *Let s be a well-behaved resource bound.*
If $\oplus\mathrm{SAT}$ has circuits of size $O(s(n))$, then $\Sigma_3\mathrm{SAT}$ is computable by a protocol in $\mathrm{MA}(s(\mathrm{poly}\, n))$.
This also holds if the Boolean basis is extended with $\mathcal{A}$.

The last ingredient of the proof is a classical result of Kannan [23], showing circuit lower bounds for $\Sigma_3\mathrm{SAT}$, and more generally for $\Sigma_3\mathrm{SAT}(t)$. His proof relativizes.

Fact 34 (Kannan's bound)**.** *Let s be a well-behaved resource bound.*
$\Sigma_3\mathrm{SAT}(\mathrm{poly}\, s(n))$ contains a language that does not have circuits of size $O(s(n))$.
This also holds if the Boolean basis is extended with $\mathcal{O}$.

With the three ingredients in hand — Scaling and Collapse lemmas, and Kannan's bound — we can prove Theorem 30. For part (i), we let $\mathcal{C}$ be the set of languages in $\mathrm{MA}(\exp n)$ and put $s(n) = 2^{\log^d n}$; for part (ii), we let $\mathcal{C}$ be the set of partial languages in $\mathrm{MA}(\mathrm{poly}\, n)$ and put $s(n) = n^d$.

The proof goes by contradiction. We give the argument using notation.

$$\begin{aligned}
&\mathcal{C} \subset \mathrm{SIZE}(O(s(n))) && \\
&\implies \oplus\mathrm{SAT} \in \mathrm{SIZE}(O(s(n))) && \text{(by Scaling lemma)}\\
&\implies \Sigma_3\mathrm{SAT} \in \mathrm{MA}(s(\mathrm{poly}\, n)) && \text{(by Collapse lemma)}\\
&\implies \Sigma_3\mathrm{SAT}(\mathrm{poly}\, s(n)) \in \mathcal{C} && \text{(*)}\\
&\implies \quad \text{contradiction} && \text{(by Kannan's bound)}
\end{aligned}$$

where step (*) follows from Definition 32 and the fact that $s(\mathrm{poly}\, s(n)) \subset \mathrm{poly}\, s(n)$ for the particular choices of $s(n)$.

What remains is the proof of Scaling and Collapse lemmas.

Proof of Scaling Lemma . There is nothing to prove in part (i), because a $\oplus\mathrm{SAT}$ instance of size n is computable by brute force in deterministic time $\exp n \cdot \mathrm{poly}\, n$, which by definition is a protocol in $\mathrm{MA}(\exp n)$.

For part (ii), suppose that every partial language in $\mathrm{MA}(\mathrm{poly}\, n)$ has circuits of size $O(n^d)$ for some fixed d. We want to show that $\oplus\mathrm{SAT}$ has circuits of size $\mathrm{poly}\, n$. By Theorem 2, $\oplus\mathrm{SAT}$ reduces to some same-length checkable language K, so it suffices to show this for K instead of $\oplus\mathrm{SAT}$.

So let K be *any* same-length checkable language, and suppose towards a contradiction that K does not have polynomial-size circuits. Let $s : \mathbb{N} \to \mathbb{N}$ be such that $s(n)$ is the size of the smallest circuit deciding K on inputs of length n, for every n. By assumption, $s(n)$ is super-polynomial, i.e., $s(n) >_{\text{i.o.}} n^k$ for every constant k. Note that $s(n)$ might not be well-behaved.

Consider the partial language $K'(xy) := K(x)$ that is defined only on inputs of the form xy where $y \in 01^*$ serves as a pad of length $|y| = \lfloor s(|x|)^\epsilon \rfloor$, for some constant $\epsilon > 0$ to be later determined.

Now consider the following MA-protocol for K': given xy, the prover sends the smallest circuit for K on inputs of length $|x|$, i.e. a circuit of size $s(|x|)$, and the verifier uses the same-length checkability of K to compute $K(x)$, hence $K'(xy)$. This takes, on an input of length $|x|+|y|$, time $\operatorname{poly} s(|x|) \subset \operatorname{poly} s(|x|)^{\epsilon} \subset \operatorname{poly}(|x|+|y|)$. So K' is in $\mathrm{MA}(\operatorname{poly} n)$, and hence has circuits of size $O(n^d)$ by assumption. But then K has circuits of size $O(n+s(n)^{\varepsilon})^d$, which is less than $s(n)$ for infinitely many n whenever $\varepsilon < 1/d$ because $s(n)$ is superpolynomial. But this contradicts $s(n)$ being the smallest circuit size for K. $\square$

Proof of Collapse Lemma. Toda famously showed [34]

$$\Sigma_3\mathrm{SAT} \to \oplus\mathrm{SAT}$$

via a randomized reduction that works over every Boolean basis for formulas. (The same holds in general for $\Sigma_k\mathrm{SAT}$ for all constant k.) So if $\oplus\mathrm{SAT}$ has circuits of size $O(s(n))$ for formulas of size n, then the MA-protocol for computing $\Sigma_3\mathrm{SAT}$, on a formula of size n, proceeds by the verifier doing the above reduction to obtain a formula of size $m \in \operatorname{poly} n$, then the prover sending a circuit for $\oplus\mathrm{SAT}$ at a large enough input length $\operatorname{poly} m$, hence a circuit of size $O(s(\operatorname{poly} n))$, and finally, the verifier running the checker for $\oplus\mathrm{SAT}_m$ (Theorem 2) on the circuit, in time $\operatorname{poly}(s(\operatorname{poly} n))$, i.e. in time $s(\operatorname{poly} n)$ since s is well-behaved. $\square$

3.5 The ZKIP Theorem

AW made the surprising observation that the famous theorem of Goldreich, Micali, and Wigderson, $\mathrm{NP} \subset \mathrm{ZKIP}$ if one-way-functions exist [21], can be proven via the same techniques underlying the IP theorem [1]. This is in contrast to the standard proof of this result involving a graph-based construction, which seems incompatible with the oracle concept.

IKK turned this idea into a complete proof by devising an indirect commitment scheme for this purpose [22]. In this section we adapt this AW-IKK proof to our framework, to get an affinely relativizing proof of the ZKIP theorem.

Theorem 35 (ZKIP theorem)**.** $\mathrm{NP} \subset \mathrm{ZKIP}$ *if there is a one-way function in* P *secure against* BPP.
This also holds if the standard Boolean basis is extended with $\mathcal{A}$.

Similarly to previous work (AW, IKK), we take for granted that there is a relativizing proof showing that under the assumption of Theorem 35, there are bit commitment schemes as in [27]. Also as in previous work, we take an informal approach to zero knowledge, declaring some protocol as leaking no information if, assuming a physical implementation of a perfectly secure bit commitment scheme (such as locked boxes containing the commitments), the verifier's view of each decommitted bit, when dealing with an honest prover, is either uniformly distributed, or deterministically computable by the verifier itself, .

Idea. We can interpret the combined AW-IKK insight as follows. Fix a vector space over any field $\mathbb{F}$. We want a protocol where given a publicly known vector u, the prover can commit to any vector v that is orthogonal to u, and the verifier checks that $v \perp u$, but learns nothing additional about v.

This can be realized by the honest prover committing three things: (i) a random vector r, (ii) the vector $r+v$, and (iii) the inner product $\langle r,u\rangle$. Since a cheating prover may deviate, let us use r, $\underline{r+v}$, and $\underline{\langle r,u\rangle}$ to denote what is actually committed for (i),(ii), and (iii) respectively.

Since $v \perp u$ iff $\langle v+r,u\rangle = \langle r,u\rangle$, the verifier picks at random one of the following two tests.

Test a. prover decommits to r and $\underline{\langle r,u\rangle}$, and verifier checks that $\langle r,u\rangle = \underline{\langle r,u\rangle}$.

Test b. prover decommits to $r+v$ and $\underline{\langle r,u\rangle}$, and verifier checks that $\langle r+v,u\rangle = \underline{\langle r,u\rangle}$.

Any prover not committing to a vector v orthogonal to u is caught by a 1/2-chance in this protocol, because then at least one equality in $\langle r+v,u\rangle = \underline{\langle r,u\rangle} = \langle r,u\rangle$ fails. On the other hand, an honest prover reveals no information about v.

Following IKK, let us refer to the prover's commitment to (i) and (ii) above, as *an indirect commitment to* v, and refer to the rest of the protocol from commitment to (iii) and onwards, as an *orthogonality test for* v with respect to u.

This protocol suggests that given a circuit C, and given a satisfying assignment x of inputs to C, in order to show that C is satisfiable without leaking x, all that an efficient prover needs to do is to commit, indirectly, to the transcript of the computation $C(x)$, to which the verifier then applies various orthogonality tests.

Protocol. Initially let us not extend the standard basis; we will visit the case of an extended basis later.

The prover is given a circuit C and a satisfying assignment x to C. Say the gates in C are indexed from $1..s$, with s being for the output gate and $1..N$ being for input gates.

Let $\mathbb{F}$ denote $\mathbb{F}_{2^k}$ for a large enough k, say $k = s$. Let $(1), .., (n{+}1)$ denote the first $n + 1$ nonzero elements under some canonical ordering of $\mathbb{F}$.

The protocol proceeds in two phases: In the first phase, for each fragment in C, of the form

$$i = f(g_1..g_n), \tag{10}$$

meaning the gate indexed $i > N$ is of type f and receives its inputs from gates indexed $g_1..g_n$ in that order (where $i > N$ because there is nothing to check for input gates), the honest prover commits, *directly*, to:

- a randomly picked nonzero vector $\vec{h} \in \mathbb{F}^n$,
- letting $z_1..z_n$ be the values of gates $g_1..g_n$ in computing $C(x)$, and ℓ be the line $\ell(t) := \vec{z} + t\vec{h}$ in $\mathbb{F}^n$, the vectors $\ell((1)), .., \ell((n{+}1))$,

and *indirectly*, to:

- the coefficients $c_1, .., c_n$ of the polynomial $\widehat{f}{\circ}\ell(t) = c_n t^n + .. + c_0$,
- the evaluations $\widehat{f}{\circ}\ell((1)), .., \widehat{f}{\circ}\ell((n{+}1))$ of the polynomial $\widehat{f}{\circ}\ell(t)$.

Also in the first phase, the honest prover commits, *indirectly*, to:

- the value v_i of each gate i in the computation $C(x)$.

This ends the first phase. Notice that there is no need to commit to the coefficient c_0 in the polynomial $\widehat{f}{\circ}\ell(t)$ for any fragment, because c_0 is supposed to equal the value v_i of gate i for the fragment (10).

Because a cheating prover may commit to other values than what he is supposed to, let us use

$$\vec{h}, \underline{\ell((1))}, .., \underline{\ell((n{+}1))}, \underline{c_1}, .., \underline{c_n}, \underline{\widehat{f}{\circ}\ell((1))}, .., \underline{\widehat{f}{\circ}\ell((n{+}1))} \tag{11}$$

to denote the commitments for each fragment, and let us use

$$y_1, .., y_s \tag{12}$$

to denote the commitments for the purported values $v_1, .., v_s$ of the gates in the computation $C(x)$.

In the second phase, the verifier V picks at random a fragment in C, say the fragment where gate i on the left hand side in (10) — call it the i^{th} fragment — and then picks at random one of the following tests:

1. letting $\ell(t)$ be the line $\ell(t) := \vec{z} + t\vec{h}$ where $\vec{z} = y_{g_1}..y_{g_n}$,
   check $\ell(j) = \underline{\ell(j)}$ for a randomly picked $j \in \{(1), .., (n{+}1)\}$
2. check $\underline{\widehat{f}{\circ}\ell(j)} = \widehat{f}(\underline{\ell(j)})$ for a randomly picked $j \in \{(1), .., (n{+}1)\}$
3. letting $\underline{\widehat{f}{\circ}\ell}(t)$ be the polynomial $\underline{c_n} t^n + .. + \underline{c_1} t + \underline{c_0}$, where $\underline{c_0} = y_i$,
   check $\underline{\widehat{f}{\circ}\ell}(j) = \underline{\widehat{f}{\circ}\ell(j)}$ for a randomly picked $j \in \{(1), .., (n{+}1)\}$
4. check $\vec{h}$ is nonzero

In case gate i is the output gate, then V in addition does:

5. check $y_i = 1$.

In tests 1, 2, 4, and 5, the prover completely reveals the relevant information; notice this means decommitting to two bits for each bit committed indirectly. Test 3 is an orthogonality test for w with respect to u, where

$$w = \underline{\widehat{f\circ\ell}(j)}\ \ \underline{c_n} \ldots \underline{c_0} \quad \text{and} \quad u = 1\ \ j^n \ldots j^0$$

so if this test is selected, then the honest prover in addition commits to $\langle r, u\rangle$, with r being the vector formed by putting together the random values sent during indirect commitments to $\widehat{f\circ\ell}(j), c_n, \ldots, c_0$ respectively.

Analysis. The completeness of the test is clear. As for soundness, suppose that C is not a satisfiable circuit. Then the committed values in (12) satisfy either of the following:

(a) There is a fragment of the form (10), for which the equality

$$y_i = f(y_{g_1}..y_{g_n})$$

fails, or

(b) the reported value of the output gate is wrong, i.e., $y_s \neq 1$.

Since there are at most s fragments, with probability $\geq 1/s$, the verifier picks an erroneous fragment for which either (a) or (b) holds. Once picked, case (b) is detected with certainty in Test 5. As for case (a), consider the values among those in (11) committed for this fragment. Adopting the notation of the second phase of the protocol, either of the following subcases must hold:

(i) the vector $\vec{h}$ is zero, or

(ii) there is some $j \in \{(1)..(n+1)\}$ for which one of the equalities

$$\widehat{f\circ\ell}(j) = \widehat{f}(\underline{\ell(j)}) = \underline{\widehat{f\circ\ell}(j)} = \underline{\widehat{f\circ\ell}}(j) \tag{13}$$

fails,

because otherwise $\underline{\widehat{f\circ\ell}}(t)$ would be the polynomial $\widehat{f\circ\ell}(t)$ and we would have

$$y_i = \underline{\widehat{f\circ\ell}}(0) = \widehat{f\circ\ell}(0) = \widehat{f}(y_{g_1}..y_{g_n}) = f(y_{g_1}..y_{g_n}).$$

contradicting that we are in case (a).

The verifier detects case (i) with probability $\geq 1/4$ (conditioned on having picked an erroneous fragment in the first place). As for case (ii), with probability $\geq 1/(n+1)$, the verifier picks an offending j, and depending on which of the first/second/third equality in (13) is violated for j, Test 1/2/3 fails respectively, with (conditional) probability $\geq \frac{1}{2}$ for Test 3 and probability 1 for Tests 1 and 2.

It follows that if the circuit C is not satisfiable, then the verifier rejects with probability $\geq 1/s^2$, with s being the number of nodes of C. Repeating the protocol from scratch $2s^2$ times brings down the soundness error to $1/3$.

Finally, the protocol is zero-knowledge, because each test that passes reveals a value that is either uniformly distributed, or is deterministically computable by the verifier itself.

Extended basis. We now generalize the protocol to handle an $\mathcal{A}$-extended Boolean basis. The idea is that the above protocol, over the unextended Boolean basis, generalizes to a protocol over the unextended *arithmetic* basis, where each gate in the given circuit is a function of the form $\mathbb{F}^m \to \mathbb{F}$ rather than $\{0,1\}^m \to \{0,1\}$. This is because all the values committed by the prover are already in $\mathbb{F}$ except for those in (12), which are over $\{0,1\}$, but which can be taken over $\mathbb{F}$ with no change to the protocol.

Therefore, given a circuit C over the $\mathcal{A}$-extended Boolean basis, all we need to do is to transform C to an appropriate arithmetic circuit D. We now explain how to do this transformation.

Let $\mathcal{O}$ be a language, and $\mathcal{A}$ its affine extension. By the defininitions in Section 2, on input $x \in \{0,1\}^n$, $\mathcal{A}$ gives the z^{th} bit of the value $\widehat{\mathcal{O}}$ takes at $y \in \mathbb{F}_{2^k}^m$, i.e.,

$$\left(\widehat{\mathcal{O}}_m^k(y)\right)_z \tag{14}$$

where y, z, m, k are all computable in polynomial-time out of x. Conversely, given (y, z, m, k), an input x for which this holds is also computable in polynomial-time.

Since k denotes the field size, or the logarithm thereof, and since $\mathbb{F}_{2^k}$ can be efficiently identified in $\mathbb{F}_{2^{\geq k}}$, (14) can be viewed as

$$\left(\widehat{\mathcal{O}}_m^K(Y)\right)_Z \tag{15}$$

for any $K \geq k$, where Y denotes y identified in $\mathbb{F}_{2^K}^m$, and Z denotes the accordingly updated z.

It follows that given a circuit C over the $\mathcal{A}$-extended Boolean basis, a polynomial-time algorithm can take each $\mathcal{A}$-gate in C, say

$$\mathcal{A}(g_1..g_n), \tag{16}$$

where g_i denotes the index of the gate that is connected to the i^{th} input of $\mathcal{A}$, and replace it with

$$\left(\widehat{\mathcal{O}}_m^s(Y(g_1..g_n))\right)_{Z(g_1..g_n)} \tag{17}$$

where Y and Z are now overloaded to denote the circuit that parses its input x as (y, z, m, k) of (14), and then outputs the values Y and Z of (15) respectively, for any $K \geq k$, in particular for $K = s$, the size of C. Note that writing Z as a subscript in (17) actually denotes another circuit, namely the circuit $\pi(a, b)$ that gives the b^{th} bit of a.

So the transformation of C is as follows.

- Perform (16) $\mapsto$ (17).
- For every m and every standard gate f_m with m inputs, replace that gate with $\widehat{f}_m^s$.

The point of the second step here is to unify the treatment of the standard gates with nonstandard ones. In the modified circuit, each gate becomes a function $\mathbb{F}^m \to \mathbb{F}$ for some m, where $\mathbb{F} = \mathbb{F}_{2^s}$.

After the transformation, the original protocol carries through, provided the inputs and the output of each gate are treated as over $\mathbb{F}$ instead of $\{0, 1\}$. This completes the proof of Theorem 35.

4 Negative Relativization Results

This section shows that several major conjectures in structural complexity are impossible to settle via an affinely relativizing proof, mirroring corresponding results of AW.

There are two main approaches to deriving such results: an interpolation approach, used for separations of the form $\mathcal{C} \not\subset \mathcal{D}$, and an approach based on communication complexity, used for containments $\mathcal{C} \subset \mathcal{D}$. Both of these approaches are model theoretic, in the sense that they construct an eligible language relative to which the statement in question is false.

The main novelty in this section, as explained in Section 1.3, is in the development of the interpolation approach, which is then used to show that $\text{NEXP} \not\subset \text{P/poly}$ is affinely non-relativizing. This is carried out in Section 4.1. The communication complexity approach is taken in Section 4.2.

Besides these two approaches there is a third, proof theoretically flavored approach, that is quite convenient to use when the situation allows. To show that ψ admits no proof that affinely relativizes, we find a statement ψ' for which this is already known, and then derive the implication $\psi \implies \psi'$ via an affinely relativizing proof. We thus show that ψ' is "no harder" to prove than ψ, in similar spirit to the use of reductions in structural complexity. It should be noted that in general, this approach cannot be used for the AW notion of algebrizing proofs, as it critically relies on the closure of such proofs under inference. Section 4.3 employs this approach.

4.1 Interpolation Approach

The classical way to show that $\mathcal{C} \not\subset \mathcal{D}$ does not admit a relativizing proof is to construct a language $\mathcal{O}$ relative to which $\mathcal{C} \subset \mathcal{D}$ holds. Such a construction amounts to a balancing act of sorts; the goal, vaguely, is to have $\mathcal{O}$ give more power to $\mathcal{D}$ than it does to $\mathcal{C}$, so as to make $\mathcal{D}$ contain $\mathcal{C}$ in the $\mathcal{O}$-extended basis. This can be done nonetheless, and sometimes easily so, as can be seen by taking $\mathcal{C} = \mathrm{PSPACE}$, $\mathcal{D} = \mathrm{P}$, and $\mathcal{O}$ to be any PSPACE-complete language (we spell this out in Proposition 36). Typically, however, the construction is more involved, and it was one of the main contributions of AW to develop an approach — the interpolation approach — that enables such constructions in the algebrization framework. Their techniques do not work for our setting, however.

In this section we develop the interpolation approach within our framework, using quite different techniques from AW's (see Section 1.3 for a comparison). Our key result here is Theorem 38, that affine extensions enable interpolation. With that result in hand, we are able to import the ideas of AW to our setting, and apply it to the NEXP versus $\mathrm{P/poly}$ question; this we do in Section 4.1.1.

Before we proceed let us note, like AW did, that the easy fact regarding PSPACE and P mentioned above carries over to our setting easily: (Recall we use 0-gap-IP for PSPACE; see Definition 19.)

Proposition 36. $0\text{-gap-IP} \not\subset \mathrm{P}$ *does not hold for all extensions of the standard Boolean basis with some* $\mathcal{A}$.

Proof. Every downward self-reducible language is in 0-gap-IP. This is because if $L := \{L_n\}$ is d-s-r, then all that a prover needs to give as proof that $L_n(x)$ equals $b \in \{0,1\}$ is the transcript of a computation involving queries for $L_{\leq n-1}$; the verifier then picks one of the claimed queries, say $L_{n-1}(y)$ and thus reduces the task to one involving $L_{\leq n-1}$, and so on.

So both $\oplus\mathrm{SAT}$ and TQBF are in 0-gap-IP, as are their negation $\neg\oplus\mathrm{SAT}$ and $\neg\mathrm{TQBF}$. Moreover, TQBF is complete for 0-gap-IP, because the interaction in any 0-gap-IP protocol can be readily expressed as a TQBF instance. All these hold also when the standard Boolean basis is extended with $\mathcal{O}$.

It follows that $0\text{-gap-IP} \subset \mathrm{P}$ with respect to the $\mathcal{O}$-extended standard basis, where $\mathcal{O} = \mathrm{TQBF}$. It turns out this containment also holds if the basis is extended, instead, with the affine extension $\mathcal{A}$ of $\mathcal{O}$. This is because regardless of what $\mathcal{O}$ is, $\mathcal{A}$ reduces to $\oplus\mathrm{SAT}^{\mathcal{O}}$ (by Corollary 17), and $\oplus\mathrm{SAT}, \neg\oplus\mathrm{SAT} \in 0\text{-gap-IP}$ under the $\mathcal{O}$-extended basis (by previous paragraph). Therefore when $\mathcal{O} = \mathrm{TQBF}$, 0-gap-IP does not become larger if $\mathcal{A}$ is used as the basis extension instead of $\mathcal{O}$. $\square$

We now move to the interpolation approach. The crux of our development is two coding-theoretic ingredients. The first one states that knowing t bits of a binary codeword exposes at most t bits of its information word, and the second scales this result to affine extensions.

Lemma 37 (Interpolation)**.** *Let* $\mathcal{E} : \mathbb{F}_2^K \to \mathbb{F}_2^N$ *be linear and injective. Given a "dataword"* $u \in \mathbb{F}_2^K$ *and a set of indices* $A \subseteq [N]$*, consider the collection* U *of all datawords* $u' \in \mathbb{F}_2^K$ *such that* $\mathcal{E}(u)$ *and* $\mathcal{E}(u')$ *agree on* A.

There is a set of indices $B \subseteq [K]$*, no larger than* A*, such that projecting* U *onto* $G := [K] \setminus B$ *gives all of* $\mathbb{F}_2^G$.

Proof. The claim of the lemma on U is true iff it is true on $U^+ := U + u$. So it suffices to show that U^+ is a subspace of $\mathbb{F}_2^K$ with dimension at least $K - |A|$.

Now, $y \in U^+$ iff $y + u \in U$, which is iff $\mathcal{E}(y + u)$ and $\mathcal{E}(u)$ agree on A, which is iff $\mathcal{E}(y)$ vanishes on A. Therefore U^+ is identical to the space of all datawords whose encodings vanish on A.

All that is left is to bound $\dim U^+$, or equivalently, to bound $\dim \mathcal{E}(U^+)$ since $\mathcal{E}$ is injective. The latter quantity is the dimension of the space $\mathcal{C} \cap \mathcal{Z}$, where $\mathcal{C}$ is the image of $\mathcal{E}$, and $\mathcal{Z}$ is the space of all N-bit vectors that vanish on A. But then by the theorem on the dimension of a sum of subspaces (e.g. [5, Thm 1.4])

$$\begin{aligned}\dim(U^+) &= \dim(\mathcal{Z}) \quad + \dim(\mathcal{C}) - \dim(\mathcal{Z} + \mathcal{C}) \\ &= (N - |A|) + \quad K \quad - \dim(\mathcal{Z} + \mathcal{C})\end{aligned}$$

which is at least $K - |A|$ because $\mathcal{Z} + \mathcal{C} \subseteq \mathbb{F}_2^N$. This finishes the proof. ☐

Theorem 38 (Interpolation)**.** *Given a language f and a finite set A of inputs, consider the collection $\mathcal{F}$ of all languages g such that $\widetilde{f}$ and $\widetilde{g}$ agree on A.*

There is a set B of inputs, no larger than A, such that every partial Boolean function g' defined outside B can be extended to some $g \in \mathcal{F}$.

Further, in extending g' to g, the values of g at length-n inputs depend only on those of g' at length n.

Proof. To begin with, consider the special case where $A \subseteq \operatorname{dom}(\widetilde{f}_m^k)$ for some fixed k and m. For the purpose of invoking Lemma 37, let $\mathcal{E}$ be the map that takes as input the truth table of a Boolean function g_m on m bits, and outputs the truth table of $\widetilde{g}_m^k$. So $\mathcal{E} : \mathbb{F}_2^K \to \mathbb{F}_2^N$, where $K = 2^m$ and $N = k2^{km}$ (to see the value of N, recall that $\widetilde{g}_m^k(y, z)$ gives the z^{th} bit of $\widehat{g}_m^k(y)$, where $\widehat{g}_m^k$ is the extension of g_m to $\mathbb{F}_{2^k}^m$).

Clearly $\mathcal{E}$ is injective; it is also linear because $\widehat{g}_m^k$ is additive, and because we represent $\mathbb{F}_{2^k}$ with $\mathbb{F}_2^k$ where addition is componentwise (Section 2). So $\mathcal{E}$ fulfils the conditions of Lemma 37, which yields a set $B \subseteq \{0,1\}^m$ that is no larger than A, such that every partial Boolean function on $\{0,1\}^m \setminus B$ can be extended to a language in $\mathcal{F}$. This proves the theorem in the special case.

To handle the general case, partition A into $A_{m,k} := A \cap \operatorname{dom}(\widetilde{f}_m^k)$, and use the above special case as a building block to create a bigger code. In detail, for every m involved in the partition, define $\mathcal{E}_m$ as the map sending the truth table of g_m to the list comprising the truth tables of $\widetilde{g}_m^{k_1}, \widetilde{g}_m^{k_2}, \ldots$ for every A_{m,k_j} in the partition. Now, take each $\mathcal{E}_m$ thus obtained, and let $\mathcal{E}$ be their product. In other words, let $\mathcal{E}$ take as input a list $T_{m_1}, T_{m_2}, ..$ where T_{m_i} is the truth table of some Boolean function g_{m_i} on m_i bits, and outputs $\mathcal{E}_{m_1}(T_{m_1}), \mathcal{E}_{m_2}(T_{m_2}), ..$. The theorem now follows from Lemma 37. ☐

4.1.1 Application — NEXP vs. $\mathrm{P/poly}$

The Interpolation theorem enables us to adapt some of the classical constructions from relativization to affine relativization. The general idea is to construct a language $\mathcal{O}$ such that $\mathcal{C} \subset \mathcal{D}$ holds relative to $\mathcal{O}$, i.e., such that $\mathcal{C}^{\mathcal{O}} \subset \mathcal{D}^{\mathcal{O}}$, by taking each algorithm underlying $\mathcal{C}^{\mathcal{O}}$, say the first n algorithms, and by fixing $\mathcal{O}$ up to a certain length, say $m(n)$, so as to force the behavior of these algorithms on inputs of length n. While doing so, the goal is for $\mathcal{O}$ to encode those forced behaviors, in a way that can be easily queried by some algorithm in $\mathcal{D}^{\mathcal{O}}$.

This classical idea can be extended to our setting via the Interpolation theorem. Even though the goal here is to have $\mathcal{C} \subset \mathcal{D}$ hold not relative to $\mathcal{O}$, but relative to its affine extension $\mathcal{A}$, we can proceed almost as before. This is because thanks to the Interpolation theorem, forcing the behavior of a $\mathcal{C}^{\mathcal{A}}$ algorithm by fixing $\mathcal{A}$ bears little extra burden on $\mathcal{O}$ than does fixing $\mathcal{O}$ to force a $\mathcal{C}^{\mathcal{O}}$ algorithm. For details we refer to the proof of the next theorem, which is the main result of this section:

Theorem 39. $\mathrm{NEXP} \not\subset \mathrm{P/poly}$ *cannot be derived via an affinely relativizing proof.*

Proof. It is a basic fact that NEXP has polynomial-size circuits iff NE (the linear-exponential version of NEXP) has circuits of size a *fixed* polynomial, and that this relativizes. In notation,

$$\mathrm{NEXP}^{\mathcal{O}} \subset \mathrm{SIZE}^{\mathcal{O}}(\mathrm{poly}\, n) \iff \mathrm{NE}^{\mathcal{O}} \subset \mathrm{SIZE}^{\mathcal{O}}(n^d) \text{ for some } d \in \mathbb{N}.$$

Therefore, to prove Theorem 39, it suffices to show a language f satisfying

$$\mathrm{NE}^{\widetilde{f}} \subset \mathrm{SIZE}^{f}(n^d), \tag{18}$$

for some constant d because f reduces to $\widetilde{f}$.

So let $M_0, M_1,..$ be a list (repetitions allowed) of all nondeterministic algorithms with access to an arbitrary language $\mathcal{O}$, such that M_i runs in time $\leq 2^{n \log n}$ on all inputs of length $n > i$. We construct the language f in such a way that when $\mathcal{O} = \widetilde{f}$, the information regarding how each M_i behaves on each large enough input x, is stored by f in a format retrievable by a small circuit. More precisely, we ensure that for every $n > 1$, a size-n^d circuit with access to f, say C_n^f, can compute the function $L_n : \{0,1\}^{\lfloor \log n \rfloor} \times \{0,1\}^n \to \{0,1\}$ defined as

$$L_n(i,x) := M_i^{\widetilde{f}}(x). \tag{19}$$

This yields (18), hence the theorem, because each language $K \in \mathrm{NE}^{\widetilde{f}}$ corresponds to some $M_i^{\widetilde{f}}$, and in order to compute $K(x)$ on all but finitely many inputs x (in particular for $x \in \{0,1\}^{>2i}$) we can just provide (i,x) to the circuit $C_{|x|}^f$, implying $K \in \mathrm{SIZE}^f(n^d)$.

We construct f inductively, as the limit of a sequence $f_1, f_2,..$ of Boolean functions where f_n extends f_{n-1}. The domain of f_n will include all of $\{0,1\}^{\leq n^d}$, plus some additional $2^{4n \log n}$ strings at most. Set $f_1 : \{0,1\} \to \{0\}$.

At iteration $n > 1$, proceed to set f_n as follows. Consider all possible ways of extending f_{n-1} to a language f. Out of all such f, pick one that maximizes (19), i.e., one for which the collection

$$\mathcal{S}_f := \{(i,x) : L_n(i,x) = 1\} \tag{20}$$

of accepting algorithm-input pairs is maximal.

Now we want to "open up space" in f by un-defining it at some inputs, the idea being then to encode the function in (19) in the freed space so that a small circuit can look it up. In doing so, of course, we do not want to disturb (19), which, by the way we picked f, is equivalent to wanting that $\mathcal{S}_f$ does not shrink — i.e., as we restrict f to some f', no matter how we extend f' back to some language g, we want $\mathcal{S}_g = \mathcal{S}_f$.

Consider an accepting algorithm-input pair (i,x) in $\mathcal{S}_f$. Because M_i runs in nondeterministic $2^{n \log n}$-time on input $x \in \{0,1\}^n$, it could issue a great many oracle queries to $\widetilde{f}$, however, as far as the membership of (i,x) in $\mathcal{S}_f$ is concerned, it suffices for $\widetilde{f}$ to honor only those queries of M_i along *one accepting* computation path. So each such pair (i,x) actually forces $\widetilde{f}$ to be fixed at only $2^{n \log n}$ inputs or less. There are at most $n2^n$ pairs in $\mathcal{S}_f$. Thus if we want $\mathcal{S}_f$ not to shrink, it suffices to fix $\widetilde{f}$ at $2^{3n \log n}$ inputs. By the Interpolation theorem, this means we only need to reserve a small set of "bad" inputs B, of size $\leq 2^{3n \log n}$, beyond those already reserved in previous iterations, i.e., beyond $\mathrm{dom}\, f_{n-1}$, such that on B we have no control as to how f behaves, but on the "good" inputs $\{0,1\}^* \setminus (B \cup \mathrm{dom}\, f_{n-1})$, we can change f arbitrarily. So let f_n be the restriction of f to $B \cup \mathrm{dom}\, f_{n-1}$.

Now that we opened up space in f, we are ready to store the information in (19) so that a small circuit can look it up. That information is the truth table of a function on $n + \log n$ bits, so it suffices to have $2^{2n \log n}$ bits available in $\mathrm{dom}\, f_n$ for this purpose. Since there are at most $2^{3n \log n}$ bad inputs in f_n by the previous paragraph, and since there are at most $2^{4(n-1)\log(n-1)}$ inputs in $\mathrm{dom}\, f_{n-1}$ that are outside $\{0,1\}^{\leq (n-1)^d}$

by induction, we know there are at most $2^{4n\log n}$ inputs currently in $\operatorname{dom} f_n$ that are outside $\{0,1\}^{\leq (n-1)^d}$. So there is sufficient space in $\{0,1\}^{n^d}$ for storage when d is large enough. As for how to actually store the information, initially consider each input (i,x) to L_n as prepended with zeroes until it becomes a string $Y_{(i,x)}$ of length n^d, and then set $f_n(Y_{(i,x)}) := L_n(i,x)$. Of course this may not work as some bad inputs may coincide with some $Y_{(i,x)}$, but this can be handled simply by changing the encoding of (i,x) to $Y_{(i,x)} \oplus Z$ for a suitably picked $Z \in \{0,1\}^{n^d}$; such Z exists because it can be picked at random with non-zero probability (by a union bound on the event that some bad input coincides with $Y_{(i,x)} \oplus Z$ for some (i,x)). This Z can then be hardwired to a circuit of size n^d, as we wanted to do.

To finish, let f_n behave arbitrarily on the rest of the good inputs in $\{0,1\}^{\leq n^d}$, and then accordingly adjust f_n on the bad inputs in $\{0,1\}^{\leq n^d}$ — recall from the Interpolation theorem that on a bad input, f_n is a function of how it behaves on non-bad inputs of same length. We have thus constructed f_n as desired. $\square$

4.2 Communication Complexity Approach

AW show that one can take a lower bound from communication complexity, and use it to construct an eligible language — an algebraic oracle in their case — relative to which $\mathcal{C} \not\subset \mathcal{D}$ holds, for an appropriate $\mathcal{C}$ and $\mathcal{D}$ depending on the lower bound picked. Therefore, AW conclude, $\mathcal{C} \subset \mathcal{D}$ cannot have an algebrizing proof.

In this section we develop this approach of AW for our framework. Our key observation here is in Proposition 42, that the affine extension respects disjoint unions. With this in hand, we are able to import the ideas of AW and of IKK to our setting, which we do in Sections 4.2.1 and 4.2.2.

We start by making a notational convention involving the classical communication complexity classes $\mathrm{P_{cc}}, \mathrm{NP_{cc}}, \mathrm{BPP_{cc}}$, etc.

Definition 40 ($\mathrm{P_{cc}}$ vs. $\mathrm{P_{ticc}}$)**.** Define $\mathrm{P_{ticc}}$ as the class of families $f := \{f_n\}$ satisfying the following. (i) Each f_n is a Boolean function on pairs of 2^n-bit strings, (ii) There is a protocol involving two algorithms M_0, M_1 such that for all n and all $(X,Y) \in \operatorname{dom}(f_n)$, the two parties $M_0^X(1^n), M_1^Y(1^n)$ compute $f_n(X,Y)$ in time $\operatorname{poly} n$.

Let $\mathrm{P_{cc}}$ denote the relaxation of $\mathrm{P_{ticc}}$ where M_0, M_1 are allowed to be non-uniform, and where only the communication between M_0, M_1 is counted towards time elapsed.

Use $\mathrm{P_{ticc}}$ to define $\mathrm{NP_{ticc}}, \mathrm{BPP_{ticc}}$, etc., similar to how we define $\mathrm{NP}, \mathrm{BPP}$, etc., from P.[12] Similarly for $\mathrm{NP_{cc}}, \mathrm{BPP_{cc}}$, etc., versus $\mathrm{P_{cc}}$.

The notation $\mathcal{C}_{\mathrm{ticc}}$ is meant to indicate that time is measured on equal grounds with communication. A function in $\mathcal{D}_{\mathrm{cc}}$ according to the classical definition [9] is defined on strings of every even length, while Definition 40 requires length a power of two; our convention causes nothing but convenience in this section.

We formalize the high-level idea of AW with the following generic theorem in our framework:

Theorem 41. *If $\mathcal{C}_{\mathrm{ticc}} \not\subset \mathcal{D}_{\mathrm{cc}}$, then $\mathcal{C} \subset \mathcal{D}$ does not hold for every extension of the standard Boolean basis with some $\mathcal{A}$. Here $\mathcal{C}, \mathcal{D}$ can be any class in the polynomial-time hierarchy containing* P.

The key ingredient in arguing Theorem 41 is the observation, mentioned on page 6, that affine extensions are compatible with disjoint unions in the following sense.

[12]Recall that definitions of $\mathrm{BPP}, \mathrm{NP}$, etc. involve some counting of the witnesses w of a P-predicate $L(x,w)$. Here, that predicate would be of the form $f(\,(X,w)\,,\,(Y,w)\,)$ where $|w|$ is polynomially bounded in n for f_n, i.e., polylogarithmic in $|X|$.

Proposition 42. *Let $\mathcal{A}_0, \mathcal{A}_1$ be the affine extension of the languages $\mathcal{O}_0, \mathcal{O}_1$ respectively. Then the disjoint union $\mathcal{A}_0 \coprod \mathcal{A}_1 : bx \mapsto \mathcal{A}_b(x)$ is equivalent, under Cook reductions, to the affine extension of the disjoint union $\mathcal{O}_0 \coprod \mathcal{O}_1 : bx \mapsto \mathcal{O}_b(x)$.*

Proof. Let $\mathcal{O} := \mathcal{O}_0 \coprod \mathcal{O}_1$. By definition, the affine extension of $\mathcal{O}$ is the Boolean version of the function that evaluates, given $B, X_1, .., X_n \in \mathbb{F}_{2^k}$ for any k, the polynomial

$$\begin{aligned}\widehat{O}(BX) &= \sum_{b,x_1,..,x_n \in \{0,1\}} \mathcal{O}(bx) \cdot \textstyle\prod_i (1 + (BX)_i + (bx)_i) \\ &= (\mathcal{O}_0(x) \cdot (1+B) + \mathcal{O}_1(x) \cdot B) \cdot \textstyle\prod_i (1 + X_i + x_i) \\ &= (1+B) \cdot \widehat{\mathcal{O}_0}(X) + B \cdot \widehat{\mathcal{O}_1}(X)\end{aligned}$$

which clearly can be evaluated given access to $\mathcal{A}_0$ and $\mathcal{A}_1$, i.e. to $\mathcal{A}_0 \coprod \mathcal{A}_1$, and vice versa. □

We now give a generic argument for Theorem 41. Supposing there is some $f := \{f_n\}$ in $\mathcal{C}_{\text{ticc}} \setminus \mathcal{D}_{\text{cc}}$, we construct a language $\mathcal{O}$ such that relative to its affine extension $\mathcal{A}$, the statement $\mathcal{C} \subset \mathcal{D}$ fails. For concreteness, the reader may take $\mathcal{C}$ to be NP, say, and $\mathcal{D}$ to be BPP.

Let $M_1^{\mathcal{O}}, M_2^{\mathcal{O}}$,... be a list of all polynomial-time decision algorithms with access to an arbitrarily picked language $\mathcal{O}$. Since $\mathcal{D}$ is definable from P, we can use this list to define a list $N_1^{\mathcal{O}}, N_2^{\mathcal{O}}, ...$ of algorithms that includes every algorithm for $\mathcal{D}^{\mathcal{O}}$, i.e., the class $\mathcal{D}$ with respect to the $\mathcal{O}$-extended basis. (Note that the list of N_i's may include more than every algorithm for $\mathcal{D}$: it corresponds to the class $\text{pr}\mathcal{D}$, the extension of $\mathcal{D}$ to partial languages.)

For every $n \in \mathbb{N}$, pick an arbitrary $(X_n, Y_n) \in \operatorname{dom} f_n$. Initialize $\mathcal{O}$ to be the disjoint union $\mathcal{O}_0 \coprod \mathcal{O}_1$, where $\mathcal{O}_0$ is the language that has the same truth table as X_n for every n, and similarly for $\mathcal{O}_1$ versus Y_n. Because $f \in \mathcal{C}_{\text{ticc}}$, the language $L := \{L_n\}$ defined as

$$L_n : 1^n \mapsto f(X_n, Y_n)$$

and defined trivially on the rest of the inputs (say 0) is in $\mathcal{C}^{\mathcal{O}}$; to see this just consider using $\mathcal{O}$ to simulate a $\mathcal{C}_{\text{ticc}}$-protocol for f. Our objective is to modify $\mathcal{O}_0$,$\mathcal{O}_1$ so that L remains in $\mathcal{C}^{\mathcal{O}}$, and becomes out of $\mathcal{D}^{\mathcal{A}}$. Then we will be done since $\mathcal{O}$ reduces to $\mathcal{A}$.

We realize this objective as follows. For $i = 1..\infty$, we pick some $(X_{n_i}, Y_{n_i}) \in \operatorname{dom} f_{n_i}$ for a large enough n_i, and update $\mathcal{O}_0, \mathcal{O}_1$ at length n_i to have the same truth table as X_{n_i}, Y_{n_i} respectively. Let us denote this update operation with $\mathcal{O}_0 \leftarrow X_{n_i}$ and $\mathcal{O}_1 \leftarrow Y_{n_i}$. Notice that updating $\mathcal{O}$ in this way readily maintains $L \in \mathcal{C}^{\mathcal{O}}$. As for ensuring $L \notin \mathcal{D}^{\mathcal{A}}$, we pick X_{n_i}, Y_{n_i} so that if $\mathcal{O}'$ denotes the updated $\mathcal{O}$ using X_{n_i}, Y_{n_i}, and $\mathcal{A}'$ denotes its affine extension, then

$$N_i^{\mathcal{A}'}(1^{n_i}) \neq f(X_{n_i}, Y_{n_i}), \tag{21}$$

which makes the i^{th} algorithm in the list fail to compute L. (Note that N_i's output might not be well-defined on the input 1^{n_i} due to N_i computing some partial language which 1^{n_i} is outside the domain of; (21) still holds in that case.)

All that remains to argue is that there are infinitely many n_i satisfying (21), because then we can pick n_i arbitrarily large, so as to not disturb the previous phases of the construction — e.g., $n_i > 2^{2^{n_{i-1}}}$ suffices since $\mathcal{D} \subset \text{EXP}$.

To argue this, let g_n be the function with the same domain as f_n, behaving as

$$g_n : (X, Y) \mapsto N_i^{\mathcal{A}'}(1^n), \tag{22}$$

where $\mathcal{A}'$ is, as before, the affine extension of $\mathcal{O}'$, and $\mathcal{O}'$ is $\mathcal{O}$ updated using X, Y. (In case $N_i^{\mathcal{A}'}(1^n)$ is not well-defined, then let g_n have value '$\perp$' on that X, Y. We may assume that g_n takes on finitely-many '$\perp$' values, for otherwise there is nothing to argue.)

Now what remains is to argue that $g := \{g_n\}$ is in $\mathcal{D}_{\text{cc}}$, for then there are infinitely many n for which $f_n \neq g_n$, implying (21) can be enforced infinitely often, as desired.

By Proposition 42, we know that the language $\mathcal{A}'$ of (22) Cook reduces to $\mathcal{A}'_0 \coprod \mathcal{A}'_1$, where $\mathcal{A}'_0$ is the affine extension of $\mathcal{O}'_0$ obtained by updating $\mathcal{O}_0$ with X, and similarly for $\mathcal{A}'_1$. Letting R denote this reduction, we can modify N_i so that it issues its oracle queries to $\mathcal{A}'_0 \coprod \mathcal{A}'_1$ instead of $\mathcal{A}'$; in notation,

$$N_i^{\mathcal{A}'}(1^n) = (N_i \circ R)^{\mathcal{A}'_0 \coprod \mathcal{A}'_1}(1^n).$$

But this shows already that g is in $\mathcal{D}_{\text{cc}}$, because there is a protocol where one party is given access to X and knows $\mathcal{O}_0$, the other party is given Y and knows $\mathcal{O}_1$, and the two parties simulate $N_i^{\mathcal{A}'_0 \coprod \mathcal{A}'_1}(1^n)$ by using each other as an oracle for $\mathcal{A}'_0$ and $\mathcal{A}'_1$ respectively. This finishes the generic argument for Theorem 41.

4.2.1 Applications

Theorem 41 allows us to replicate two negative algebrization results of AW:

Corollary 43. *Neither of the following statements can be derived via an affinely relativizing proof: (i)* $\text{coNP} \subset \text{MA}$, *(ii)* $\text{P}^{\text{NP}} \subset \text{PP}$.

Proof. Let $\text{Disj}(X, Y) := \forall i\ \neg(X(i) \wedge Y(i))$ be the disjointness predicate. It is clear that $\text{Disj} \in \text{coNP}_{\text{ticc}}$. On the other hand, it takes at least $\Omega(2^{n/2})$ bits of communication to compute Disj by a Merlin-Arthur protocol [24, Corollary 1], implying $\text{Disj} \notin \text{MA}_{\text{cc}}$. Part (i) now follows from Theorem 41.

For part (ii), let $\text{LYB}(X, Y) := \max_{i \in \{0,1\}^n}(X(i) \wedge Y(i))$, and let $f(X, Y) := \text{LYB(X, Y)} \mod 2$ be the predicate for whether the Last-Yes-Bit position of X and Y is odd. It is easy to see that $f \in (\text{P}^{\text{NP}})_{\text{ticc}}$. On the other hand, it takes at least $\Omega(2^{n/3})$ bits of communication to compute f by a probabilistic protocol [13, Section 3.2],[13] implying $f \notin \text{PP}_{\text{cc}}$. Part (ii) now follows from Theorem 41. ☐

We can use Theorem 41 to replicate a result of IKK as well:

Claim 44. $\text{RP} \subset \text{SUBEXP}$ *cannot be derived via an affinely relativizing proof.*[14] *Here* SUBEXP *denotes* $\cap_\varepsilon \text{DTIME}(2^{n^\varepsilon})$.

Proof sketch. Consider the set of all pairs of strings (X, Y) such that X, Y respectively equal the truth table of $\widetilde{f}_m^k, \widetilde{g}_m^k$ for some m and some $f, g : \{0,1\}^m \to \{0,1\}$, where $k = 2 + \log m$. Consider restricting the equality predicate $\text{Equal}(X, Y) := \forall i(X(i) \equiv Y(i))$ to this set, and call the resulting function $E(X, Y)$.

Yao's classical result on Equal implies that there is no $F \in \text{SUBEXP}_{\text{cc}}$ that E can be extended to. In short, and with a slight abuse of notation, $E \notin \text{SUBEXP}_{\text{cc}}$.

On the other hand, if we take the class RP_{ticc}, and relax its definition to include families $\{f_n\}$ where $\text{dom}\, f_n$ is no longer required to be the set of every pair X, Y of length 2^n strings, but rather some of them, then we may call the resulting class $\text{prRP}_{\text{ticc}}$, and then see that $E \in \text{prRP}_{\text{ticc}}$ by the Schwartz-Zippel Lemma.

[13]The authors of [13] show a stronger result where the protocol allows both parties to use private randomness as well, with a suitable generalization of the acceptance condition for the protocol.

[14]IKK show the stronger result where $\text{SUBEXP} = \cap_\varepsilon \text{DTIME}(2^{n^\varepsilon})$ is replaced by $\cap_\varepsilon \text{DTIME}(2^{\varepsilon n})$. Using a less modular argument we can derive this result as well.

Now the proof of Theorem 41 in Section 4.2 is written exactly with this more general situation in mind. Namely, if $\mathrm{pr}\mathcal{C}_{\mathrm{ticc}} \not\subset \mathcal{D}_{\mathrm{cc}}$, then $\mathcal{C} \subset \mathcal{D}$ does not hold for every extension of the standard Boolean basis with some $\mathcal{A}$; here $\mathcal{C}$, $\mathcal{D}$ can be any class definable from P and contained in EXP. The claim follows. $\square$

4.2.2 Extensions

Refining the AW approach, IKK considerably strengthened a result of AW: they showed that no algebrizing proof (for their notion of algebrization) exists for NP having sub-linear-exponential circuits, even at infinitely many input lengths. We can extend Theorem 41 to replicate this result in our framework as well:

Claim 45. $\mathrm{NP} \subset \text{i.o.-SIZE}(2^{\varepsilon n})$ *cannot be derived via an affinely relativizing proof for some* $\varepsilon > 0$.

We give a rough outline of a proof. Similar to the argument for Theorem 41, we consider a list of all size-$2^{\varepsilon n}$ Boolean circuits on n-bits, for each $n \in \mathbb{N}$. Similarly again, we define a language L that encodes, at each input length n, a single instance of a communication problem $f(X, Y)$, with X and Y being encoded in the oracle. The difference here, following IKK, is that f is the direct product of a Boolean problem instead of a merely Boolean one, for which we know a much stronger variant of $f \notin \mathcal{D}_{\mathrm{cc}}$, namely the strengthening of this to average-case hardness on all input lengths (as opposed to worst-case hardness at infinitely many input lengths). This allows us to use a randomized process to define the oracle "at once", thereby obtaining a hardness for L that holds at every input length. We refer to [22, Lemma 4.2] for details.

4.3 Proof Theoretic Approach

As mentioned in the beginning of Section 4, sometimes we can get away without constructing oracles, and still show that ψ admits no proof that relativizes affinely. To do so, we find some ψ' which we already know has that status, and then derive the implication $\psi \implies \psi'$ via an affinely relativizing proof. We thus reduce the task of creating an oracle relative to which ψ is false, to doing the same for ψ', with the proof of $\psi \implies \psi'$ serving as the reduction. More generally, we reduce the task of showing that ψ admits no affinely relativizing proof, to doing the same for ψ'.

Using the results of Section 4.1-4.2 we can readily show:

Theorem 46. *None of the following statements can be derived via an affinely relativizing proof:*
(i) $\mathrm{NP} \subset \mathrm{P}$, *(ii)* $\mathrm{NP} \not\subset \mathrm{P}$, *and (iii)* $\mathrm{NP} \subset \mathrm{BPP}$.

Proof. Part (i): Theorem 39 showed that $\mathrm{NEXP} \not\subset \mathrm{P/poly}$ cannot have an affinely relativizing proof, and Theorem 30 showed that $\mathrm{MA(exp)} \not\subset \mathrm{P/poly}$ via an affinely relativizing proof. The claim follows because $\mathrm{NP} \subset \mathrm{P}$ implies $\mathrm{MA} \subset \mathrm{P}$, which in turn implies $\mathrm{MA(exp)} \subset \mathrm{NEXP}$, both implications being derivable via a relativizing (hence affinely relativizing) proof.

Part (ii): Proposition 36 showed that $0\text{-gap-IP} \not\subset \mathrm{P}$ cannot have an affinely relativizing proof. The claim follows since $\mathrm{NP} \subset 0\text{-gap-IP}$ via a relativizing proof.

Part (iii): Corollary 43 states that $\mathrm{coNP} \subset \mathrm{MA}$ does not have an affinely relativizing proof. The claim follows since $\mathrm{NP} \subset \mathrm{BPP}$ implies $\mathrm{coNP} \subset \mathrm{MA}$ via a relativizing proof. $\square$

5 Conclusions and Open Problems

Our results counter the folkloric belief that relativizing techniques treat computation only as a "black box" mapping inputs to outputs (e.g., [1, p. 2]), and that arithmetization, or more generally a circuit-based view of computation, seems to let us "peer into the guts of it" [2, p. 115], and hence circumvents the limits of relativizing techniques.

In contrast, according to our definitions, a Boolean formula in the relativizing view, say over the basis $\{\wedge, \oplus, \mathcal{O}\}$, gives complete freedom regarding how the $\mathcal{O}$-gates behave, and in this sense each $\mathcal{O}$-gate is a black box, of "volume" the size of its truth-table. In the affinely relativizing view, however, each $\mathcal{O}$-gate redundantly encodes a Boolean function, by extending its domain from $\mathrm{GF}(2)^n$, say, to $\mathrm{GF}(2^k)^n$; this means that the behavior of the gate is determined by 2^n entries of a truth table of size roughly 2^{kn}. So each $\mathcal{O}$-gate has a black-box "core", carrying on with the metaphor, of volume roughly k^{th} root of its overall volume; here k must be $\Omega(\log n)$ for all the results catalogued in this paper, and can be taken as $\Theta(\log n)$ for a formula of size $O(n)$.

So it seems that: (i) circuit-based techniques *are* relativizing, if they are insensitive to enlarging the basis arbitrarily, (ii) arithmetization-based techniques *are* also relativizing, only "slightly less" so. To make this a bit more precise, consider the following question: what can be the circuit complexity, over the standard basis $\{0, 1, \wedge, \oplus\}$, of a size-n circuit over the extended basis $\{0, 1, \wedge, \oplus, \mathcal{O}\}$? In the relativizing view, i.e., in $\mathcal{RCT}$, the answer is $2^{O(n)}$ — just consider a single $\mathcal{O}$-gate with $n-1$ inputs. To see this in the affine-relativizing view $\mathcal{ACT}$, let us first clean up the definition of affine extension a bit, so that if f is a Boolean function on n inputs, then its affine extension involves $\mathrm{GF}(2^k)$ for $k \geq \log n$ only, instead of $k \geq 1$. By the above discussion, this makes no difference for the results catalogued in this paper, but now the answer is easily seen to be $2^{O(n/\log n)}$, again via an $\mathcal{O}$-gate with $n-1$ inputs. Dividing by n and taking logarithms, we get what might be called the "opacity" of each theory, a quantity that ranges from $O(1)$ at the real-world end of complexity theory, to $O(n)$ at the fully relativized end, with affine relativization being above $O(n^{1-\varepsilon})$ for every $\varepsilon > 0$, just "slightly less" than relativization.

We finish by listing some suggestions for further research.

A quantitative theory of relativization. Both relativization and affine/algebraic relativization are rigid notions, in the sense that something either relativizes or does not. However, the discussion just above, on the various degrees of being opaque, calls for a theory of relativization that is gradual, based on the information content — or density, so to speak — in an oracle.

Can we associate to each statement a "relativization rank", so that the algebrization barrier arises as a quantitative gap, between a lower bound on one hand for the rank of algebrizing statements, and an upper bound on the other, for the rank of non-algebrizing statements? If so, then we could view the reciprocal of the rank as a useful complexity measure on theorems and conjectures, just as we have complexity measures on algorithmic tasks: the larger the reciprocal of the rank, the higher the "relativization sensitivity" of the statement in hand, indicating more resources — stronger axioms — required to prove it.

New oracles from old. Section 4.3 showed that sometimes we can evade the task of constructing an oracle, by reducing the task to another one already done. For example, there is no need to construct an (affine) oracle refuting $\mathrm{NP} \subset \mathrm{P}$ when we already have one refuting $\mathrm{NEXP} \not\subset \mathrm{P/poly}$, because $\mathrm{NP} \subset \mathrm{P} \implies \mathrm{NEXP} \not\subset \mathrm{P/poly}$ via an affinely relativizing proof — meaning $\mathrm{NP} \subset \mathrm{P}$ is harder to prove than $\mathrm{NEXP} \not\subset \mathrm{P/poly}$ in some sense.

Can we use this idea to simplify the landscape of oracle constructions? For example, many of the statements shown not affinely relativizing in Section 4.2 are containments of the form $\mathcal{C} \subset \mathcal{D}$ for which various circuit lower bound consequences are known. This suggests that a handful of oracles, each refuting some circuit lower bound, may yield a rich collection of statements getting indirectly refuted via reductions of the form given in the above example.

Weaker theories for arithmetization. As asserted in Section 1.2, we can replicate all the classification given by IKK (as well as by AW) for what algebrizes and what does not, however, we do not know if algebrization in the IKK sense implies affine relativization, or vice versa.[15] This suggests that there should be a weaker characterization of arithmetization-based techniques that subsumes both notions.

Is there a constraint that we can place on the basis extension $\mathcal{O}$, besides that it is a language, so that the resulting theory is a consequence of both versions of $\mathcal{ACT}$, ours and IKKs, and still derives all the theorems shown algebrizing by AW? (Notice that such a theory would automatically be unable to prove anything unprovable by $\mathcal{ACT}$, hence all the non-algebrizing statements of AW.)

Of course, the weakest axiom deriving a theorem is the theorem itself, so there is a trivial answer to the question the way stated above: just take the conjunction of all the algebrizing statements, IP theorem, MIP theorem, etc., and add it as an axiom. This kind of "overfitting" clearly lacks the succinctness desired in a theory, so we need to amend the question a bit. Say that a proof is nontrivial if the proof remains valid when viewed in $\mathcal{CT}$. Then we want a theory that is a consequence of both versions of $\mathcal{ACT}$, and that *nontrivially* derives all theorems shown algebrizing by AW.

The PCP **theorem.** Section 1.3 explained that both the IP theorem and the MIP theorem can be naturally viewed as a gap-amplification result, and from that point of view both theorems have affinely relativizing proofs. Can we extend this reasoning to the PCP theorem? If so, this would bolster the candidacy of affine relativization as a proxy for arithmetization-based techniques.

A completeness theorem for oracles. If we can prove ψ, and that ψ relativizes, then is there a relativizing proof of ψ? It is consistent with experience that such a "completeness" phenomenon holds. Confirming this would allow us to focus solely on proving facts about statements, and not on how we prove those facts.

Along the same lines, what if each statement in a proof relativizes — then does the proof itself relativize? If so, then we could say that a proof relativizes if and only if each of its intermediate statements does. (The "only if" direction is already true by the way we defined things in Section 1.1; the non-trivial part is to make the jump from the semantic fact that each step relativizes, to the syntactic one that the proof relativizes.)

A genuine independence result. Be it in our version of $\mathcal{RCT}$ and $\mathcal{ACT}$, or in AIVs and IKKs, Section 1.2 pointed out that the axioms go on top of an existing collection of axioms governing everyday mathematics. Another approach to formalizing these barriers, would be to propose a *subset* of axioms governing everyday math, the idea being to find the "weakest" version of everyday math that can derive each algebrizing statement, and then to show that no non-algebrizing statement can be derived by that much of mathematics.

Acknowledgements

We thank Scott Aaronson and Eric Allender for their helpful comments. This research was supported by NSF Grant CCF-1420750, and by the Graduate School and the Office of the Vice Chancellor for Research and Graduate Education at the University of Wisconsin-Madison with funding from the Wisconsin Alumni Research Foundation.

Bibliography

[1] Scott Aaronson and Avi Wigderson. Algebrization: A new barrier in complexity theory. *ACM Transactions on Computation Theory*, 1(1), 2009.

[15] The IKK approach is over $\mathbb{Z}$ but can be adapted to $\mathrm{GF}(2^k)$, so this is not the issue. Also, the IKK approach builds on the AIV formulation of $\mathcal{RCT}$, but it can also use our version of $\mathcal{RCT}$, so again this is not the issue.

[2] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[3] Sanjeev Arora, Russell Impagliazzo, and Umesh Vazirani. Relativizing versus nonrelativizing techniques: the role of local checkability. Manuscript retrieved from http://cseweb.ucsd.edu/ russell/ias.ps, 1992.

[4] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998.

[5] Emil Artin. *Geometric Algebra*. John Wiley & Sons, 1957.

[6] László Babai. E-mail and the unexpected power of interaction. In *Proceedings of the Structure in Complexity Theory Conference*, pages 30–44, 1990.

[7] László Babai and Lance Fortnow. Arithmetization: A new method in structural complexity theory. *Computational Complexity*, 1:41–66, 1991.

[8] László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991.

[9] László Babai, Peter Frankl, and Janos Simon. Complexity classes in communication complexity theory (preliminary version). In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 337–347, 1986.

[10] Theodore P. Baker, John Gill, and Robert Solovay. Relativizatons of the P =? NP question. *SIAM Journal on Computing*, 4(4):431–442, 1975.

[11] Manuel Blum and Sampath Kannan. Designing programs that check their work. *Journal of the ACM*, 42(1):269–291, 1995.

[12] Harry Buhrman, Lance Fortnow, and Thomas Thierauf. Nonrelativizing separations. In *Proceedings of the IEEE Conference on Computational Complexity*, pages 8–12, 1998.

[13] Harry Buhrman, Nikolai K. Vereshchagin, and Ronald de Wolf. On computation and communication with small bias. In *Computational Complexity*, pages 24–32, 2007.

[14] Alan Cobham. The intrinsic computational difficulty of functions. In *Proceedings of the International Conggress for Logic, Methodology, and Philosophy of Science II*, pages 24–30, 1964.

[15] Stephen A. Cook. Short propositional formulas represent nondeterministic computations. *Information Processing Letters*, 26(5):269–270, 1988.

[16] Herbert B. Enderton. *A mathematical introduction to logic*. Academic Press, 1972.

[17] Lance Fortnow. The role of relativization in complexity theory. *Bulletin of the EATCS*, 52:229–243, 1994.

[18] Lance Fortnow. [Blog post: The great oracle debate of 1993]. Retrieved from http://blog.computationalcomplexity.org/2009/06/great-oracle-debate-of-1993.html, 2016.

[19] Lance Fortnow, John Rompel, and Michael Sipser. On the power of multi-prover interactive protocols. *Theoretical Computer Science*, 134(2):545–557, 1994.

[20] Lance Fortnow and Michael Sipser. Are there interactive protocols for co-NP languages? *Information Processing Letters*, 28(5):249–251, 1988.

[21] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity for all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(3):691–729, 1991.

[22] Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. An axiomatic approach to algebrization. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 695–704, 2009.

[23] Ravi Kannan. Circuit-size lower bounds and nonreducibility to sparse sets. *Information and Control*, 55(1):40–56, 1982.

[24] Hartmut Klauck. Rectangle size bounds and threshold covers in communication complexity. In *Computational Complexity*, pages 118–134, 2003.

[25] Richard Lipton. [Blog post: I hate oracle results]. Retrieved from http://rjlipton.wordpress.com/2009/05/21/i-hate-oracle-results, 2016.

[26] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *Journal of the ACM*, 39(4):859–868, 1992.

[27] Moni Naor. Bit commitment using pseudorandomness. *J. Cryptology*, 4(2):151–158, 1991.

[28] Rahul Santhanam. Circuit lower bounds for Merlin-Arthur classes. *SIAM Journal on Computing*, 39(3):1038–1061, 2009.

[29] Rahul Santhanam. [Comment to blog post: Barriers to proving P!=NP]. Retrieved from http://www.scottaaronson.com/blog/?p=272#comment-7634, 2016.

[30] Adi Shamir. IP = PSPACE. *Journal of the ACM*, 39(4):869–877, 1992.

[31] Alexander Shen. IP = PSPACE: simplified proof. *Journal of the ACM*, 39(4):878–880, 1992.

[32] Victor Shoup. New algorithms for finding irreducible polynomials over finite fields. *Mathematics of Computation*, 54(189):435–447, 1990.

[33] Larry J. Stockmeyer and Albert R. Meyer. Word problems requiring exponential time: Preliminary report. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 1–9, 1973.

[34] Seinosuke Toda. On the computational power of PP and +P. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 514–519, 1989.

[35] Leslie G. Valiant and Vijay V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47(3):85–93, 1986.