Electronic Colloquium on Computational Complexity, Revision 3 of Report No. 40 (2016)

Affine Relativization: Unifying the Algebrization and Relativization Barriers

Barış Aydınlıoğlu [*] Eric Bach [†]

September 5, 2017

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 give a reformulation of algebrization without these shortcomings. First, we define what it means for any statement / proof to hold relative to any language, with no need to refer to devices like a Turing machine with an oracle tape. Our approach dispels the widespread misconception that the notion of oracle access is inherently tied to a computational model. We also connect relativizing statements to proofs, by showing that every proof that some statement relativizes is essentially a relativizing proof of that statement.

We then define a statement / proof as relativizing *affinely* if it holds relative to every *affine oracle* — here an affine oracle is the result of a particular error correcting code applied to the characteristic string of a language. We show that every statement that AW declare as algebrizing does relativize affinely, in fact has a *proof* that relativizes affinely, 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 $\mathrm{P/poly}$ question.

Using our definitions we obtain new streamlined proofs of several classic results in complexity, including $\mathrm{PSPACE} \subset \mathrm{IP}$ and $\mathrm{NEXP} \subset \mathrm{MIP}$. This may be of separate interest.

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

ISSN 1433-8092

Contents

1 Introduction **1**
- 1.1 Relativization and Affine Relativization . . . 3
  - 1.1.1 Relativization without oracle machines . . . 4
  - 1.1.2 ψ is true relative to $\mathcal{O}$ vs. ψ is provable relative to $\mathcal{O}$ . . . 5
  - 1.1.3 Affine relativization . . . 8
  - 1.1.4 Relativizing statements vs. relativizing proofs . . . 9
  - 1.1.5 Affinely relativizing statements vs. affinely relativizing proofs . . . 11
  - 1.1.6 Relativizing formulas . . . 12
- 1.2 Comparing With and Clarifying Prior Work . . . 13
- 1.3 Overview of Ideas and Techniques . . . 19

2 Definitions, Notation and Conventions **24**

3 Positive Relativization Results **30**
- 3.1 Checking and Compressing $\oplus$SAT . . . 31
- 3.2 The IP Theorem . . . 39
- 3.3 The MIP Theorem . . . 41
- 3.4 Lower Bounds against General Boolean Circuits . . . 51
- 3.5 The ZKIP Theorem . . . 53

4 Negative Relativization Results **56**
- 4.1 Interpolation Approach . . . 57
- 4.2 Communication Complexity Approach . . . 60
- 4.3 Reduction Approach . . . 63

5 Suggestions for Further Research **63**

Bibliography **64**

1 Introduction

Motivation. The algebrization notion — short for algebraic relativization — was put forth by Aaronson and Wigderson [2] (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 [2, 40, 30], 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,[1] 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 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 [9, 22].

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 1.1.6.)

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, hence classes definable[2] from P — 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 the same techniques, yet only one might be relativizing / algebrizing. For example, PSPACE $\subset$ IP, as AW show, algebrizes,

[1] We use $\subset$ for containment and $\subsetneq$ for proper containment throughout the paper.

[2] Most complexity classes can be viewed as the result of applying various operators to P; see Section 1.1.1.

yet its cousin, $\mathrm{NEXP} \subset \mathrm{MIP}$, does *not*, as observed by Impagliazzo, Kabanets, and Kolokolova [30] — 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 [35]) 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. (For example, does a proof that reads "Suppose $\mathrm{P} = \mathrm{NP}. \ldots$" not relativize, since $\mathrm{P} = \mathrm{NP}$ does not?)

Naturally thus the question arises, of whether precise definitions can be given for what constitutes a relativizing / algebrizing statement / proof, ideally in a way that agrees with the everyday intuitions for these notions.

Prior Work. An early draft by Arora, Impagliazzo, and Vazirani [5] (AIV) gives a precise definition of a relativizing proof, and building on the AIV approach to relativization, Impagliazzo, Kabanets, and Kolokolova [30] develop an analogous approach for algebrization. However, that approach falls short of recovering some key results of AW, most notably regarding the NEXP versus P/poly question. See Section 1.2.

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

First, we give a definition of what it means for a statement / proof to hold relative to a language, with no need to refer to devices like a Turing machine with an oracle tape. Our approach dispels the widespread misconception that the notion of oracle access is inherently tied to a computational model. We also show an interesting connection from relativizing statements to proofs, namely, that every proof that ψ relativizes is essentially a relativizing proof of ψ (see Theorem 8 for a precise statement).

Our main contribution is to the algebrization notion. We define a statement / proof as relativizing *affinely* if it holds relative to every *affine oracle* — here an affine oracle 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 algebrizing 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.[3] (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 [13].

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 $\mathcal{O}$ [25, 24, 16],

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

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 Section 1 — 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).

Our formulations also tell something about those "known proof techniques" that do not seem to algebrize, 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 [23, 5, 22, 30]. 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 [5], 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, namely that the algebrization barrier idea of AW can be rigorously supported:

Summary of Results. *Affinely relativizing proofs, as defined in Section 1.1, derive each statement classified as algebrizing by AW, and provably cannot derive any statement classified as non-algebrizing by AW in the classical model. In particular:*

- *Each of the following has an affinely relativizing proof*
  - $\mathrm{PSPACE} \subset \mathrm{IP}$, *(Corollary 34)*
  - $\mathrm{NEXP} \subset \mathrm{MIP}$, *viewed as gap amplification (Theorem 40)*
  - $\mathrm{MA_{EXP}} \not\subset \mathrm{SIZE}(2^{\log^d n}), \forall d$ *(Theorem 44)*
  - $\mathrm{prMA} \not\subset \mathrm{SIZE}(n^d), \forall d$ *(Theorem 44)*
  - $\mathrm{NP} \subset \mathrm{ZKIP}$ *if one-way-functions exist (Theorem 49)*
- *None of the following has an affinely relativizing proof*
  - $\mathrm{NP} \not\subset \mathrm{P}$, *in fact* $\mathrm{PSPACE} \not\subset \mathrm{P}$ *(Proposition 50)*
  - $\mathrm{NP} \subset \mathrm{P}$, *in fact* $\mathrm{RP} \subset \mathrm{SUBEXP}$ *(Corollary 57)*
  - $\mathrm{NP} \subset \mathrm{BPP}$, *in fact* $\mathrm{coNP} \subset \mathrm{MA}$ *(Corollary 56)*
  - $\mathrm{P}^{\mathrm{NP}} \subset \mathrm{PP}$ *(Corollary 56)*
  - $\mathrm{NEXP} \not\subset \mathrm{P/poly}$, *in fact* $\mathrm{NEXP} \not\subset \mathrm{SIZE}(n^d), \forall d$ *(Theorem 53)*

1.1 Relativization and Affine Relativization

We now explain our formulation of the relativization and affine relativization notion. We caution that our results do not depend on any peculiarity of the definitions we give here. The reader who is already at ease

with some notion of relativization (vague though it may be) can choose to skip this section and still follow the rest of the paper.

1.1.1 Relativization without oracle machines

One of the first things a student typically learns about relativization is that $\mathrm{C}^{\mathcal{O}}$, the class C relative to the language $\mathcal{O}$, is not obtained from C, but rather from the *definition* of C. (This is true in general, otherwise we cannot have both $\mathrm{IP} = \mathrm{PSPACE}$ and that $\mathrm{IP}^{\mathcal{O}} \neq \mathrm{PSPACE}^{\mathcal{O}}$ for some $\mathcal{O}$, for example.) Unfortunately, it is easy to misinterpret this fact, even for professionals, as though relativization is a notion inherently tied to computational devices like Turing machines. To relativize a class, say to get $\mathrm{P}^{\mathcal{O}}$ from P, a common belief is that we must take an enumeration of machines underlying P, and endow each with a mechanism to access the language $\mathcal{O}$. (See e.g, [33, 18, 22].) We cannot, as the belief goes, obtain $\mathrm{P}^{\mathcal{O}}$ by treating P as a set; we must modify the definition of that set.

As we show now, this is false — at least for the classical notion of relativization from Baker, Gill, and Solovay [13], which is the one we formalize in this paper.

Take any definition of FP equivalent to the typical definition that uses Turing machines, be it the typical definition itself, or a machineless one such as [19] or [15]. Consider the "oracle operator"

$$(V, \mathcal{O}, \ell) \mapsto V^{\mathcal{O}[\ell]}$$

which, given functions $V, \mathcal{O} : \{0,1\}^* \to \{0,1\}^*$, and given the function $\ell : \mathbb{N} \to \mathbb{N}$, outputs the function $V^{\mathcal{O}[\ell]}$ defined as

$$\begin{aligned} V^{\mathcal{O}[\ell]} : x \mapsto V(x, a), \quad & \text{where} \\ & a_1 = \mathcal{O}(V(x, \epsilon)), \\ & a_2 = \mathcal{O}(V(x, a_1)), \\ & \vdots \\ & a_i = \mathcal{O}(V(x, a_1..a_{i-1})), \\ & |a| = \ell(|x|), \end{aligned} \tag{1.1}$$

and where ϵ is the empty string. Now, the class FP can be relativized simply as

$$\mathrm{FP}^{\mathcal{O}} := \{\ V^{\mathcal{O}[n^c+c]} : V \in \mathrm{FP}, c \in \mathbb{N}\ \}. \tag{1.2}$$

Intuitively, $\mathrm{FP}^{\mathcal{O}}$ is obtained by taking every FP-predicate and conferring upon it the power to interact with a prover that can only compute $\mathcal{O}$ — the power to do a Cook-reduction to $\mathcal{O}$, essentially. (We formalize this intuition in §2.7.)

Proposition 1. *$\mathrm{FP}^{\mathcal{O}}$ is exactly the set of all functions computable by a polynomial-time Turing machine with oracle access to $\mathcal{O}$.*

Proof. Use $\mathrm{FP}^{(\mathcal{O})}$ to denote the set which we want to show $\mathrm{FP}^{\mathcal{O}}$ equals to.

That $\mathrm{FP}^{\mathcal{O}} \subset \mathrm{FP}^{(\mathcal{O})}$ is easy. To compute a function of the form (1.1) with $\ell = n^c + c$ and $V \in \mathrm{FP}$, a Turing machine with access to $\mathcal{O}$ can construct $a_1..a_\ell$ bit by bit, and then output $V(x, a_1..a_\ell)$.

The converse $\mathrm{FP}^{\mathcal{O}} \supset \mathrm{FP}^{(\mathcal{O})}$ is easy as well. If M is a Turing machine with access to $\mathcal{O}$, running in time at most $|x|^c + c$ on every input x, then there is an equivalent machine M' that makes exactly $|x|^c + c$ queries to $\mathcal{O}$, by repeating if necessary the last query made by M, and that takes no more than $|x|^d + d$ steps for some d. Also, there is a machine Q that on input $(x, a_1..a_i)$ simulates $M'(x)$, by interpreting $a_1..a_i$ as the answers to the first i queries of M', and that outputs the next query of M'. If M' halts during the simulation,

then Q outputs whatever M' outputs. In case M' takes too long during simulation, longer than $|x|^d + d$ steps, which could happen if $a_1..a_i$ incorrectly lists the oracle answers, then Q outputs something arbitrary, say 0. Notice Q does *not* need access to $\mathcal{O}$. Therefore, if V is the function computed by Q, then $V \in \mathrm{FP}$. It follows by (1.1) that $V^{\mathcal{O}[n^c+c]}$ is identical to the function computed by M; by (1.2), this function is in $\mathrm{FP}^{\mathcal{O}}$. □

So FP can be relativized *extensionally*, i.e., as a set, with no need to know the definition of that set.

What about other complexity classes? Most classes have straightforward definitions in terms of FP; it is for these classes that relativization is actually useful as a guide. For example, NP, BPP, P/poly, EXP are all definable, extensionally, from FP. Even PSPACE is definable in that manner, as $\Sigma_\infty \mathrm{P}$ (§2.3). In fact all these classes can be viewed as the result of applying various operators to the class P: NP as $\mathrm{N} \cdot \mathrm{P}$, BPP as $\mathrm{BP} \cdot \mathrm{P}$, so on (e.g., [34, p.187], [20], §2.3), and P in turn can be obtained extensionally from FP.

To relativize these classes, we simply relativize FP. In other words, we *define* $\mathrm{NP}^{\mathcal{O}}$ as $\mathrm{N} \cdot (\mathrm{P}^{\mathcal{O}})$, or more precisely (and bombastically) as $\mathrm{N} \cdot \frac{1}{\mathrm{F}} \cdot \mathrm{FP}^{\mathcal{O}}$, where $\frac{1}{\mathrm{F}}$ denotes the process of obtaining P from FP (say by taking each $V \in \mathrm{FP}$ and composing it with the projection function $\pi : x_1..x_n \mapsto x_1$). Similarly, we *define* $\mathrm{BPP}^{\mathcal{O}}$ as $\mathrm{BP} \cdot (\mathrm{P}^{\mathcal{O}})$, and so on.

By centering everything on FP, we thus take a disciplined approach to relativization, with no dilemmas about what oracle access means for different classes. Since we avoid such a dilemma for FP to begin with, we thus have an oracle-free treatment of relativization.

While this approach omits some complexity classes, e.g., Logspace, it is debatable whether that is a limitation of the approach, or whether such classes are beyond the scope of relativization as originally intended. Regardless, for all the results in this paper, the approach given here is sufficient.

Oracle machines without machines. By generalizing the above approach we can also dispense with the notion of "a polynomial-time oracle Turing machine". This notion is more general than "a polynomial-time Turing machine with oracle access to $\mathcal{O}$", because it allows us to say things like

$$\text{there exists an oracle machine } M^* \text{ such that for every } \mathcal{O},\ M^{\mathcal{O}} \text{ satisfies } \ldots \tag{♯}$$

and things like

$$\text{take an enumeration of oracle turing machines running in time } t \ldots \tag{♭}$$

(for an appropriate t), the point being in both cases that the quantification of the machine comes *before* the oracle.

For $V \in \mathrm{FP}$ and $\ell : \mathbb{N} \to \mathbb{N}$, let $V^{*[\ell]}$ be the function

$$V^{*[\ell]} : (\mathcal{O}, x) \mapsto V^{\mathcal{O}[\ell]}(x)$$

that maps a given language $\mathcal{O}$ and string x to $V^{\mathcal{O}[\ell]}(x)$ as in (1.1) above. Now let

$$\mathrm{FP}^* := \{V^{*[n^c+c]} : V \in \mathrm{FP}, c \in \mathbb{N}\}. \tag{1.3}$$

Then a member f^* of FP^* serves just like a polynomial-time oracle Turing machine; given $\mathcal{O}$, it yields some $f^{\mathcal{O}} \in \mathrm{FP}^{\mathcal{O}}$. (We make use of this construct in the sequel, to formalize statements similar to (♯) and (♭).)

1.1.2 ψ is true relative to $\mathcal{O}$ vs. ψ is provable relative to $\mathcal{O}$

Perhaps *the* first thing a student typically learns about relativization is to call a statement, say ψ, relativizing if it remains true when all Turing machines are given access to an arbitrary oracle $\mathcal{O}$. Two vague spots exist in this definition; we already started to address the second one, how to "give all Turing machines an oracle",

in the previous discussion. The first, and the more fundamental issue is, what is really meant for something to be "true".

Of course, the notion of truth was settled in the 1950s by Alfred Tarski and Robert Vaught who took a model-theoretic perspective (e.g., [29]). However, model-theoretic truth does not seem to capture the intent of "true" above in the casual definition of relativization.

Instead, we use *provables* and *refutables* to formalize relativization in this paper. If ψ has a proof, then it is provable; if it has a disproof, i.e., if its negation $\neg\psi$ has a proof, then it is refutable. Every statement is either true or false, but some statements are neither provable nor refutable, by the Incompleteness theorem — assuming, here and throughout, that everyday mathematics is consistent.

Let us be more precise. Take the axioms of everyday set theory, ZFC. The reader does not need to know much about ZFC, besides two things. First, one can express most of everyday mathematics, certainly all the math in this paper, as proofs from ZFC. The reader can take this as a thesis, akin to the Church-Turing thesis that every computer program can be implemented by a Turing machine.

The second thing to know about ZFC is that it is expressed in the language of first order logic, with two special symbols: the binary relation symbol $\in$, intended to represent membership, and the constant symbol $\emptyset$, intended to represent the empty set.

Of course, intentions are irrelevant; $\in$ and $\emptyset$ can be interpreted in any way that respects the axioms. In other words, the notions "set", "is a member of", "empty set" are left undefined; they are primitive notions. ZFC stipulates how these primitive notions behave with each other, and the rest of mathematics is built using these notions. (For example, the natural number 1 is defined as the set containing, as its only element, the empty set; the existence and uniqueness of 1 is then proved from ZFC.) In yet other words, the symbols $\in$ and $\emptyset$ are devoid of meaning; they constitute the special symbol set — the *signature*, in the terminology of mathematical logic — of the language of first order logic in which we express mathematics.

Note that symbols such as 0, 1, $\mathbb{N}$ are not a part of the signature like $\in$, $\emptyset$ are. They are introduced with definitions, and are placeholders for those definitions, like macros in a programming language. In other words they are not primitive notions.

Now, add a constant symbol, $\mathcal{O}$, to the signature. Add an axiom to ZFC that says

$$\mathcal{O} \text{ is a language.} \tag{$\dagger$}$$

We can formalize this axiom in a number of ways, e.g., as $\mathcal{O} \subset \{0,1\}^*$, or as $\mathcal{O} : \{0,1\}^* \to \{0,1\}$, depending on how we want to formalize languages. (Here "language" is used in the complexity-theoretic sense, not the linguistic sense as in "the language of first order logic".) Without loss of generality, we pick the latter, and thus view languages as functions from $\{0,1\}^*$ to $\{0,1\}$.

Now add another constant symbol, FP, to the signature. Also add another axiom to ZFC, as follows. Take any definition of the class of polynomial time computable functions, say, the standard one based on Turing machines; don't call this class FP yet — call it TheFP instead. Now add an axiom that says

$$\mathrm{FP} \text{ equals } (\mathrm{TheFP})^{\mathcal{O}}. \tag{$\ddagger$}$$

Here $(\mathrm{TheFP})^{\mathcal{O}}$ is obtained by taking TheFP and applying the oracle operator, as in (1.1) and (1.2).

Just as we can express everyday mathematics using ZFC, we can use

$$\mathrm{ZFC} + (\dagger) + (\ddagger)$$

to define relativized complexity classes P, NP, BPP, etc., from FP, as explained in Section 1.1.1, and then try to derive results about these classes.

If we want to study unrelativized versions of these classes, then we can set $\mathcal{O}$ to be a trivial language,

say the empty (i.e., constant-zero) language. That is, we can work under the axiom system

$$\mathrm{ZFC} + (\dagger) + (\ddagger) + \text{“}\mathcal{O} \text{ is empty”}$$

where “$\mathcal{O}$ is empty” can be formalized as $\forall x\, \mathcal{O}(x) = 0$, for example.

Similarly, if we want to study P, NP, BPP, etc., relative to a particular language O_0 (e.g., the language SAT) then we can work under

$$\mathrm{ZFC} + (\dagger) + (\ddagger) + \text{“}\mathcal{O} \text{ is } O_0\text{”}$$

where “$\mathcal{O}$ is O_0” might require possibly infinitely many axioms to formalize, but not more than countably many, since we can always give a list of input-output pairs describing $\mathcal{O}$.

To recap, we have:

Definition 2. Using the language of first order logic with signature $\mathcal{S} := (\in, \emptyset, \mathcal{O}, \mathrm{FP})$, let

- $\mathrm{CT}(*)$ denote ZFC + “$\mathcal{O}$ is a language” + “FP equals $(\mathrm{TheFP})^{\mathcal{O}}$”,
- $\mathrm{CT}(O_0)$ denote, for every language O_0, $\mathrm{CT}(*)$ + “$\mathcal{O}$ is O_0”.

Our framework of mathematics is $\mathrm{CT}(\mathbf{0})$ — *unrelativized complexity theory* — where $\mathbf{0}$ is the empty language mapping every input to 0. When we say ψ is provable, for example, we mean it is provable from $\mathrm{CT}(\mathbf{0})$ unless we say otherwise. We refer to $\mathrm{CT}(*)$ as *relativized complexity theory*, and $\mathrm{CT}(O_0)$ as *complexity theory relative to* O_0.

Definition 3. Let ψ be a statement in the language of first order logic with signature $\mathcal{S}$ as in Definition 2. Call ψ:

- a *fact* if it is a theorem of $\mathrm{CT}(\mathbf{0})$,
- a *fact relative to* O_0 if it is a theorem of $\mathrm{CT}(O_0)$.

Here “ψ is a theorem of T” means the same thing as “ψ is provable from T”, namely: if we take everything in T to be true, then ψ follows.

We caution that it is not required to abandon unrelativized complexity theory, just to be able to talk about relativized classes. For example, that $\mathrm{NP}^{\mathrm{SAT}} = \Sigma_2\mathrm{P}$ *is* a fact of unrelativized complexity theory. In fact we never have to leave unrelativized complexity: Switching to the relativized world $\mathrm{CT}(O_0)$ amounts to *reinterpreting* the symbol FP, thus giving every class (built from FP) an oracle for O_0, and the same effect can be achieved by syntactically replacing every occurrence of FP with FP^{O_0}. (This is in fact the approach taken from Chapter 2 (see §2.19) onwards; the reason we use relativized worlds here is that it makes the metamathematical exposition much easier.)

We are ready to define relativizing and nonrelativizing statements.

Definition 4. Let ψ be a statement in the language of first order logic with signature $\mathcal{S}$ as in Definition 2. Call ψ *relativizing* iff it is a fact relative to every language.

There are a couple of things to note about Definition 4, which are illustrated in Figure 1 as a sideways tree. First, if ψ is a theorem of relativized complexity theory, then it is relativizing (i.e., row 1 in the figure implies rows 1 & 2), because a single proof of ψ from $\mathrm{CT}(*)$ yields, for every O_0, a proof of ψ from $\mathrm{CT}(O_0)$. However, the converse is not clear. (An analogous situation is having a single algorithm that runs in time $n^{1+o(1)}$ versus having, for each ε, an algorithm running in time $n^{1+\varepsilon}$.) We may say that theorems of relativizing complexity theory are *uniformly* provable in every relativized world. (In the next subsection we will say such facts have relativizing *proofs*.) Interestingly, all facts designated as relativizing in the everyday

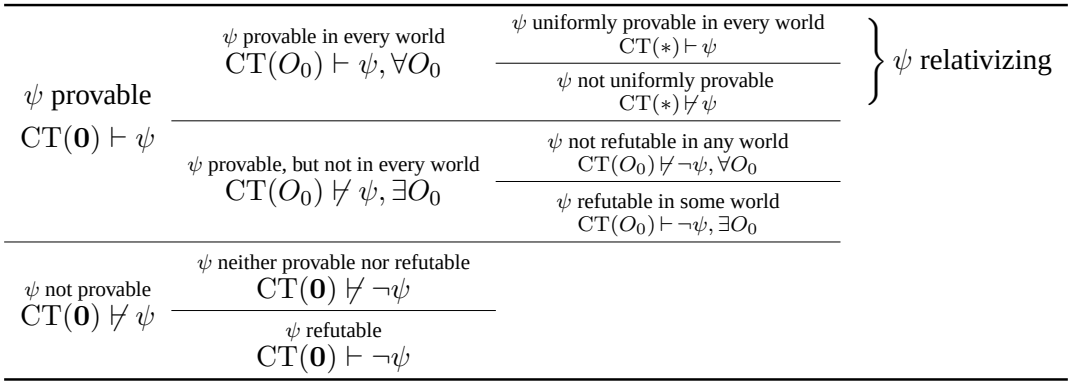

Figure 1: Illustration for Definition 4 (as a sideways tree)

sense, seem to be theorems of relativized complexity. (In the next subsection we will show that this is not a coincidence.)

Second, if ψ is a fact that is refuted relative to some O_0, then it is a nonrelativizing fact (i.e., row 4 in the figure implies rows 3 & 4). But the converse does not always hold. We do not know of a natural example for this, however; like in the first point above, all facts that are designated as nonrelativizing in the everyday sense, as far as we know, are facts that are refuted in some relativized world.[4]

1.1.3 Affine relativization

Affine oracles are motivated in Section 1.3 and precisely defined in Section 2. Roughly, the language A is an affine oracle if 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 A represents the evaluation of $\widehat{f}_n$ over $\mathrm{GF}(2^k)$, for all k and n.

Definition 5 (Definition 4 cont'd)**.** Let ψ be a statement in the language of first order logic with signature $\mathcal{S}$ as in Definition 2. Call ψ *affinely relativizing* iff it is a fact relative to every affine oracle.

The remarks made right after defining relativizing statements (Definition 4) have analogues for affine relativization. In particular, all facts we show in this paper as affinely relativizing are provable from the following theory:

Definition 6 (Definition 2 cont'd)**.** Using the language of first order logic with signature $\mathcal{S}$ as in Definition 2, let:

- $\mathrm{CT}(\widetilde{*})$ denote $\mathrm{CT}(*)$ + "$\mathcal{O}$ is an affine oracle".

We refer to $\mathrm{CT}(\widetilde{*})$ as *affinely relativized complexity theory*. Notice that $\mathrm{CT}(\widetilde{*})$ does not immediately appear to be a subtheory of $\mathrm{CT}(\mathbf{0})$, unrelativized complexity theory. It is: $\mathrm{CT}(\mathbf{0})$ by definition is $\mathrm{CT}(*)$ + "$\mathcal{O}$ is $\mathbf{0}$", and the empty language $\mathbf{0} : x \mapsto 0$ *is* an affine oracle; this follows from the precise definitions in Section 2 (and should be plausible given the rough definition in the beginning of this subsection).

[4]Here is an "unnatural" example: Let $\mathbf{1}$ be the language that outputs 1 on every input x. Then $\mathrm{CT}(\mathbf{1}) = \mathrm{CT}(*)$ + "$\mathcal{O}$ is $\mathbf{1}$", where we may use $\forall x\, \mathcal{O}(x) = 1$ to formalize that last axiom. Now, let ψ be "if $\mathcal{O} = \mathbf{1}$ then ZFC is consistent". Then for every language $O_0 \neq \mathbf{1}$, $\mathrm{CT}(O_0)$ proves "$\mathcal{O} \neq \mathbf{1}$", hence proves ψ. On the other hand, $\mathrm{CT}(\mathbf{1})$ proves "$\mathcal{O} = \mathbf{1}$", and hence, by the Incompleteness theorem, cannot prove or refute ψ — assuming, as we agreed to do, that ZFC is consistent.

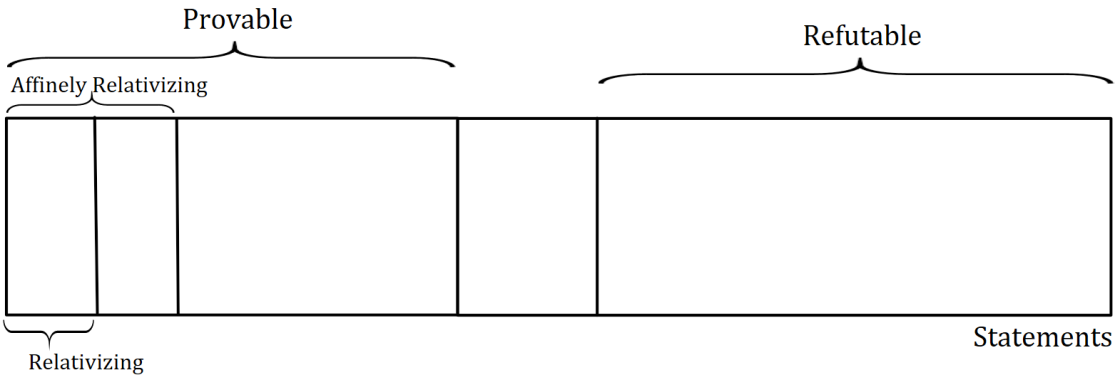

Figure 2: (Affinely) relativizing statements

We use Figure 2 to illustrate the relationship between relativization and affine relativization. The "relativization barrier" idea corresponds to the vertical line between the set of relativizing statements and its complement. Similarly, the "algebrization barrier" idea can be pictured as the line between the set of affinely relativizing statements and its complement; "ψ is affinely nonrelativizing" would then say that ψ is somewhere to the right of that line.

1.1.4 Relativizing statements vs. relativizing proofs

We call a proof relativizing if, intuitively, it goes through when we "give every polynomial-time Turing machine an arbitrary oracle $\mathcal{O}$".

Definition 7. Call a proof from $\mathrm{CT}(\mathbf{0})$ *relativizing* iff it is also a proof from $\mathrm{CT}(*)$.

Since $\mathrm{CT}(*)$ is a subtheory of $\mathrm{CT}(\mathbf{0})$, relativizing proofs are exactly those proofs from $\mathrm{CT}(*)$. Proofs that relativize make no assumptions about the primitive notion $\mathcal{O}$ other than that it is a language.

As remarked in Section 1.1.2, it is not obvious if a theorem of the form "ψ relativizes" implies, by itself, that ψ has a proof that relativizes. Nonetheless, in the process of deriving such a theorem, it seems invariably one ends up giving such a proof. (Of course everyday proofs are not spelled out in the language of first order logic, just as everyday algorithms algorithms are not specified as Turing machines or Boolean circuits.)

We now show this is not a coincidence.

Theorem 8. *Let ψ be a statement in the language of first order logic with signature $\mathcal{S}$. Every proof that ψ relativizes can be augmented to make it a relativizing proof of ψ. The augmentation does not depend on the contents of ψ.*

This theorem is consequential to the rest of the paper. It says that all statements that are *known* to relativize are theorems of relativized complexity theory, i.e., they have relativizing proofs. In fact, it says, to give a relativizing proof of ψ, it suffices to give a proof that ψ relativizes; a desired proof can then be obtained by adding a few additional lines, and those lines that do not even depend on ψ. So it says, essentially:

Every proof that ψ relativizes is a relativizing proof of ψ.

The upshot is that we never have to worry about showing results of the form "ψ has a relativizing proof"; it suffices to show "ψ relativizes" instead.

Proof of Theorem 8. Let ψ be a statement over the signature $(\in, \emptyset, \mathcal{O}, \mathrm{FP})$. Suppose we are given a proof that ψ relativizes. Since our framework of mathematics is unrelativized complexity theory $\mathrm{CT}(\mathbf{0})$, this means we are given a proof that takes all the axioms $\mathrm{CT}(\mathbf{0})$ as true and derives that: ψ relativizes, i.e., that: ψ is a fact relative to every language O_0, i.e., that:

$$\text{for every language } O_0, \mathrm{CT}(O_0) \text{ proves } \psi. \tag{♭}$$

First, notice there is nothing mysterious about mentioning one theory $\mathrm{CT}(O_0)$ while we are inside another theory $\mathrm{CT}(\mathbf{0})$; it is akin to one Turing machine simulating another.

Second, consider the following question. The proof of (♭), which we know is in unrelativized complexity theory — does it really use anything specific to *un*relativized complexity theory? The statement (♭) does seem to involve the symbol $\mathcal{O}$, but only superficially: we could talk about complexity theory by using another symbol, say $\mathcal{O}'$, instead of $\mathcal{O}$. That is, letting $\mathrm{CT}'(O_0)$ be the same theory as $\mathrm{CT}(O_0)$ except using $(\in', \emptyset', \mathcal{O}', \mathrm{FP}')$ as the signature instead of $(\in, \emptyset, \mathcal{O}, \mathrm{FP})$, we can restate (♭):

$$\text{for every language } O_0, \mathrm{CT}'(O_0) \text{ proves } \psi', \tag{♭′}$$

where ψ' is obtained by replacing all occurrences of $\in$ (and respectively, $\emptyset$, $\mathcal{O}$, and FP) with $\in'$ (and respectively, with $\emptyset'$, $\mathcal{O}'$, and FP'). Now (♭′) clearly does not require any assumption on $\mathcal{O}$ to be proven, because we can replace in the proof of (♭′) all occurrences of $\mathcal{O}$ with what $\mathcal{O}$ stands for in unrelativized complexity theory, namely the empty language $\mathbf{0}$. Therefore, (♭′) is a theorem of relativized complexity theory $\mathrm{CT}(*)$.

We now show that

$$\text{if (♭′) then } \psi \tag{♯}$$

is also a theorem of relativized complexity theory. This suffices to establish the claim, since we can put (♭′) and (♯) together and conclude ψ in relativized complexity theory.

Recall the thesis that everyday mathematics, in particular the mathematics in this paper, can be carried out in the language of first order logic using the ZFC axioms of set theory, which takes "set membership" and "empty set" as primitive notions. (The reader who is at ease with the Church-Turing thesis should feel the same about this one.) Under this thesis, whatever we can prove by starting out with "let $\mathcal{O}$ be a language" is a theorem of relativized complexity theory.

So let $\mathcal{O}$ be a language. Suppose (♭′). Then

$$\mathrm{CT}'(\mathcal{O}) \text{ proves } \psi'. \tag{⊢}$$

Applying the soundness theorem of first order logic (e.g., [21, Thm 6.2]) to (⊢) we get,

$$\psi' \text{ is a consequence of } \mathrm{CT}'(\mathcal{O}), \tag{⊨}$$

which means that under every interpretation of the symbols $(\in', \emptyset', \mathcal{O}', \mathrm{FP}')$ that satisfy the axioms $\mathrm{CT}'(\mathcal{O})$, the statement ψ' holds. In particular, if we interpret $\in'$ as set membership, $\emptyset'$ as the empty set, $\mathcal{O}'$ as $\mathcal{O}$, and FP' as FP, then ψ' holds.[5] But ψ' is exactly ψ under that interpretation.

So, ψ. We just proved (♯) as desired. □

For the reader who wants to avoid the model-theoretic perspective taken during the transition from (⊢) to (⊨) in the above proof, here is the sketch of an alternative argument. A proof can be viewed as a table, with each cell containing one statement. Each row is obtained by applying an *inference rule* to zero, or one,

[5] The application of the soundness theorem is subtle here. If a theory T proves φ, then φ is true in every model of T; this is the soundness theorem. But by definition a model is a *set*, whereas here the model is the universe of everyday set theory, the "set" of all sets, which is not a set. Nevertheless the soundness theorem still holds in this setting. Also see the remarks following the proof.

or two previous rows, depending on the arity of the rule. The last cell in the last row of the table contains the conclusion of the proof, and the rest of the cells in that row contain the assumptions of the proof.

So a proof of φ from the theory T is a table whose last row reads $\Gamma\ \varphi$, where Γ is a finite sequence of statements from T.

The following can be shown in relativized complexity theory:

$$\text{for every proof, if the assumptions of the proof are true, then so is its conclusion.} \qquad (\diamond)$$

This can be shown by induction on the structure of the proof, i.e., on how the proof was obtained from inference rules. The specifics of the induction depends on what rules of inference are allowed; for concreteness we take the exposition in [21, Ch IV] but in general any Hilbert-style inference system would do.

The base case of the induction corresponds to rules of arity zero. There are two such rules: (i) the row $\Gamma\ \varphi$ can be written anytime, for all sequences of statements Γ that contain the statement φ, and (ii) the row consisting of the single statement $t = t$ can be written anytime, for all terms t. The claim ($\diamond$) clearly holds for these two rules.

We illustrate the inductive step with one rule of arity two, namely the contradiction rule: for any two rows $\Gamma\ \neg\varphi\ \sigma$ and $\Gamma\ \neg\varphi\ \neg\sigma$, the row $\Gamma\ \varphi$ can be written anytime. Here Γ is a sequence of statements, φ and σ are statements. Suppose we have a proof ending with $\Gamma\ \varphi$, derived by applying the contradiction rule. Suppose the induction hypothesis holds for the subproofs ending with those two lines from which the last line is derived. We want to show that if everything in Γ is true, then so is φ. So suppose everything in Γ is true. Suppose towards a contradiction that φ is false. Then $\neg\varphi$ is true, and by the induction hypothesis, both σ and $\neg\sigma$ is true, a contradiction. Therefore, φ is true. The inductive step for the other rules proceed similarly.

Now let us go back to the proof of Theorem 8, to right after ($\vdash$), and give an alternate way of concluding ψ. Letting $\mathcal{O}$ be an arbitrary language, we know that $\mathrm{CT}'(\mathcal{O})$ proves ψ'. That is, there exists a proof (a table) that concludes with $\Gamma'\ \psi'$, where Γ' is a sequence formed from a finite subset of $\mathrm{CT}'(\mathcal{O})$. Take any such proof; replace all occurrences of $\in', \emptyset', \mathcal{O}', \mathrm{FP}'$ with $\in, \emptyset, \mathcal{O}, \mathrm{FP}$ respectively. The resulting object is a proof ending with $\Gamma\ \psi$, where every statement in Γ is either in $\mathrm{CT}(*)$, or is a trivial statement such as $\mathcal{O}(t) \equiv \mathcal{O}(t)$ for some term t (because $\mathrm{CT}'(\mathcal{O})$ by definition is $\mathrm{CT}'(*)$, plus axioms that implement "$\mathcal{O}'$ is $\mathcal{O}$", and replacing $\mathcal{O}'$ with $\mathcal{O}$ turns "$\mathcal{O}'$ is $\mathcal{O}$" into the trivial "$\mathcal{O}$ is $\mathcal{O}$"). Therefore, ψ, and we have Theorem 8 again.

1.1.5 Affinely relativizing statements vs. affinely relativizing proofs

Section 1.1.4 carries over to affine relativization in a straightforward fashion. We call a proof affinely relativizing if, intuitively, it goes through when we "give every polynomial-time Turing machine an arbitrary affine oracle $\mathcal{O}$".

Definition 9 (Definition 7 cont'd)**.** Call a proof from $\mathrm{CT}(\mathbf{0})$ *affinely relativizing* iff it is also a proof from $\mathrm{CT}(\widetilde{*})$.

Since $\mathrm{CT}(\widetilde{*})$ is a subtheory of $\mathrm{CT}(\mathbf{0})$ (see remarks after Definition 6), affinely relativizing proofs are exactly those proofs from $\mathrm{CT}(\widetilde{*})$. Intuitively, proofs that affinely relativize make no assumptions about the primitive notion $\mathcal{O}$ other than that it is an affine oracle.

Every proof that ψ affinely relativizes is essentially an affinely relativizing proof of ψ:

Theorem 10 (Theorem 8 cont'd)**.** *Let ψ be a statement in the language of first order logic with signature $\mathcal{S}$. Every proof that ψ affinely relativizes can be augmented to make it an affinely relativizing proof of ψ. The augmentation does not depend on the contents of ψ.*

The upshot is that we never have to worry about showing results of the form "ψ has an affinely relativizing proof"; it suffices to show "ψ affinely relativizes" instead.

(The proof is a straightforward adaptation of the proof of Theorem 8, so we omit it.)

1.1.6 Relativizing formulas

Our current definitions involve FP and classes definable from FP (§1.1.1). It is desirable to extend the definitions to formulas as well, so that statements such as "SAT is NP-hard" become relativizing. This is for a couple of reasons. First, it is natural: for example SAT naturally extends to the language $\mathrm{SAT}^{\mathcal{O}}$, of satisfiable formulas over the Boolean basis extended with $\mathcal{O}$, which is complete for $\mathrm{NP}^{\mathcal{O}}$. Second, complexity classes are typically used interchangeably with their complete languages, e.g., P^{NP} for $\mathrm{P}^{\mathrm{SAT}}$, but unless we find a way of relativizing formulas simultaneously with classes, such natural equivalences fail in relativized settings.

There are several ways to do this extension; we give two. The first approach is straightforward and corresponds to the everyday intuition, "'SAT is NP-complete' relativizes *if* we include oracle gates in SAT". The second approach is slightly more involved as it builds on the first, but it yields the rather clean intuition that a statement / proof relativizes (affinely) iff it is insensitive to extending the standard Boolean basis with any (affine) oracle.

First approach. Rewind to Section 1.1.2. In addition to $\mathcal{O}$, add another symbol, $\mathcal{B}$, to the signature of the language of first order logic in which we express mathematics ($\mathcal{B}$ for "basis", $\mathcal{O}$ for "oracle").

Let $\wedge : \{0,1\}^* \to \{0,1\} : x \mapsto \wedge_i x_i$ be the language implementing an AND-gate of arbitrary fan-in. Similarly, let $\oplus$, $\mathbf{0}$, $\mathbf{1}$ implement, respectively, an XOR, a constant 0, a constant 1 gate of arbitrary fan-in. Define B_{std} to be the ordered set $\{\mathbf{0}, \mathbf{1}, \wedge, \oplus\}$. (As is the case with symbols such as 0, 1, $\mathbb{N}$, the symbols B_{std}, $\mathbf{0}$, $\mathbf{1}$, etc. are not added to the signature; each is a placeholder for its definition.)

Now add the axiom "$\mathcal{B} = B_{\mathrm{std}} \cup \{\mathcal{O}\}$" to relativized complexity theory $\mathrm{CT}(*)$ — hence to other flavors of complexity theory as they are all built from $\mathrm{CT}(*)$. Just like we pledged to work with the relativized class $\mathrm{FP} = (\mathrm{TheFP})^{\mathcal{O}}$ instead of TheFP, we can pledge to define all Boolean circuits (hence formulas) over the basis $\mathcal{B}$. More precisely, define

$$CircEval(C, x)$$

as the function that, given the circuit C whose internal gates have labels of the form "ith basis element", for $i \in \{1, .., 5\}$, and given the input x to C, evaluates $C(x)$ by interpreting "1st basis element" as $\mathbf{0}$, "2nd basis element" as $\mathbf{1}$, ..., and "5th basis element" as $\mathcal{O}$.

This way, we can work with circuits / formulas without fear of making a nonrelativizing statement just because we mention circuits / formulas. In particular "SAT is NP-complete" does become relativizing, as does many others such as "$\oplus$SAT is $\oplus$P-complete", since SAT, $\oplus$SAT, etc., are all defined by building on $CircEval$.

Second approach. Implement the first approach first. Now define a new class, $\mathrm{FP}_{\mathrm{circ}}$, as the set of functions computable by polytime-uniform circuits of polynomial size, over the basis $\mathcal{B}$. More precisely, $F \in \mathrm{FP}_{\mathrm{circ}}$ iff there is a function $Desc_F \in \mathrm{TheFP}$ satisfying, for every $x \in \{0,1\}^*$,

$$F(x) = CircEval(Desc_F(1^{|x|}), x),$$

where $CircEval$ is as defined in the first approach.

By the Cook-Levin theorem, $\mathrm{FP}_{\mathrm{circ}}$ equals $(\mathrm{TheFP})^{\mathcal{O}} = \mathrm{FP}$. This suggests that we can drop the symbol $\mathcal{O}$ and use the signature $\mathcal{S} := (\in, \emptyset, \mathcal{B}, \mathrm{FP})$ when expressing mathematics. In particular, we can redefine relativized complexity theory and its variants by letting

- $\mathrm{CT}(*)$ denote ZFC + "FP equals $\mathrm{FP}_{\mathrm{circ}}$" + "$\mathcal{B} = B_{\mathrm{std}} \cup \{O_0\}$ for some language O_0"
- $\mathrm{CT}(\widetilde{*})$ denote $\mathrm{CT}(*)$ + "$\mathcal{B} = B_{\mathrm{std}} \cup \{O_0\}$ for some affine oracle O_0"
- $\mathrm{CT}(O_0)$ denote, for every language O_0, $\mathrm{CT}(*)$ + "$\mathcal{B} = B_{\mathrm{std}} \cup \{O_0\}$".

As was the case in the original definition (Definition 2), here $\mathrm{CT}(O_0)$ may require infinitely many axioms to formalize. (Think of "$\mathcal{B} = B_{\mathrm{std}} \cup \{O_0\}$" as "every language O that is a member of $\mathcal{B} \setminus B_{\mathrm{std}}$ is O_0", and formalize "... is O_0" as described right before Definition 2.)

Notice that everything expressed in relativized complexity theory in sense of the original definition (Definition 2) can be equivalently expressed in the sense here, by mentioning $\mathcal{B} \setminus B_{\mathrm{std}}$ instead of $\mathcal{O}$.

Furthermore, we can still work with circuits / formulas without fear of making a nonrelativizing statement just because we mention circuits / formulas. For example, let SAT denote the satisfiability problem for formulas over the basis $\mathcal{B}$. Then "SAT is NP-complete" is a theorem of $\mathrm{CT}(*)$, almost by definition, since the Cook-Levin theorem is essentially embedded in the definition of $\mathrm{FP}_{\mathrm{circ}}$, hence of FP.

With this approach, a statement (and respectively, a proof) is relativizing iff it remains a fact (respectively, a proof) when the standard Boolean basis is extended with an arbitrary language. Similarly for affinely relativizing statements and proofs. Notice that using this approach we dispense with the intuition of giving "every Turing machine oracle access", because extending the standard basis automatically does that.

In the sequel, either of the approaches above can be taken as the foundation for our results. For the sake of readability, we pick the first approach.

1.2 Comparing With and Clarifying Prior Work

Four past works have a direct relation to ours — besides the paper that started it all, of course, by Baker-Gill-Solovay. The main effort in these four works, 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 do a quick survey of these works, first with respect to (i) and then (ii). Along the way we dispel some common misconceptions as needed.

1.2.1 Efforts to formalize relativization

In mathematics, we can introduce a new object in two ways: constructive or axiomatic. The constructive way is to introduce the object by defining it in terms of other objects already available. For example, if we have sets available, then we can define natural numbers as

$$\begin{aligned} 0 &= \emptyset \\ 1 &= \{0\} = \{\emptyset\} \\ 2 &= \{0, 1\} = \{\emptyset, \{\emptyset\}\} \\ 3 &= \{0, 1, 2\} = \{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}\} \end{aligned}$$

and so on. (Of course, "and so on" is to be replaced with rigorous mathematics: inductive sets, the successor operation — and so on.)

In the axiomatic approach, on the other hand, we introduce the object as a primitive notion, essentially leaving it undefined; instead we add axioms that describe how instances of this new object behave in a

universe consisting entirely of these objects. For example, we can treat "natural number" as a primitive notion, along with a relation called $\mathrm{Successor}$, and add axioms saying that "for every natural number x, there is a unique natural number y such that (y, x) are in the $\mathrm{Successor}$ relationship" and so on.

In other words: we can introduce an object either by saying what it *is*, or by saying what it *does*. The former would be constructive, the latter axiomatic.

Since we cannot create something out of nothing, we have to begin by declaring some notions as primitive. So the foundations of mathematics are axiomatic. Beyond that, however, the two approaches often complement each other. For example, real numbers can be introduced as a complete ordered field, or as infinite sequences of decimals, or as Dedekind cuts, or as equivalence classes of Cauchy sequences. The first approach would be axiomatic, the rest constructive, each one informative in its own right.

Our treatment of relativization in Section 1.1, in particular, our definition of relativized classes, at first glance might appear to be axiomatic, since the very act of putting FP in the signature is technically declaring FP as a primitive notion. That is superficial, however; FP can be taken out of the signature and treated instead as an ordinary symbol with no loss of generality — after all it is simply declared to equal the class $(\mathrm{TheFP})^{\mathcal{O}}$, an object introduced constructively. (Recall TheFP stands for unrelativized polynomial time computable functions.)

So our approach is constructive, and it is an interesting question whether relativized classes can be defined axiomatically. This is what **Arora, Impagliazzo, Vazirani** [5] (AIV henceforth) partially answer. Following Cobham [19], AIV introduce FP by an inductive definition: they first declare certain basic functions to be in FP — the length function $x \mapsto |x|$, all constant functions, etc. Then they add new functions to FP based on existing ones — e.g., if f and g are in FP then so is their composition. The resulting object exactly characterizes relativized computation: $(\mathrm{TheFP})^{\mathcal{O}}$ satisfies the definition of FP for every language $\mathcal{O}$, and anything that satisfies the definition of FP equals $(\mathrm{TheFP})^{\mathcal{O}}$ for some $\mathcal{O}$.

The AIV result is particularly interesting because it strengthens the message of Section 1.1.1 considerably: not only relativization can be defined independently of machines, as we showed in Section 1.1.1, but also the very idea of oracle access can arise out of a natural definition that has nothing to do with oracles — there is no mention of an oracle $\mathcal{O}$ in the AIV approach.

(On the flip side, lack of any reference to an oracle $\mathcal{O}$ makes it unclear whether a statement like "SAT is NP-complete" is relativizing under the AIV definition.)

There is a big caveat that must be noted regarding the AIV approach as well as ours. In both works, FP is defined *after* introducing an entire collection of axioms formalizing everyday mathematics.[6] (This is why we consider AIV's work as only partially answering the question of whether relativized classes can be defined axiomatically.)

It is the lack of appreciation of this caveat that we believe underlies a widespread misconception, namely, that these results prove some sort of logical independence for statements that do not relativize (e.g., [3], [38], [1, Section 6.1.2]). To illustrate, consider the fact that $\mathrm{NEXP} \subset \mathrm{P/poly}$ is provable relative to some $\mathcal{O}$ and refutable relative to some others. We could summarize this fact in the terminology of Section 1.1 as

$$\mathrm{NEXP} \subset \mathrm{P/poly} \text{ is independent of } \mathrm{CT}(*),$$

but we would be saying nothing about the axiomatic complexity of the $\mathrm{NEXP} \stackrel{?}{\subset} \mathrm{P/poly}$ question: we certainly believe that ZFC, a *sub*-theory of $\mathrm{CT}(*)$, proves either $\mathrm{NEXP} \subset \mathrm{P/poly}$ or its negation!

The confusion arises because there are two ways of expressing the same question in $\mathrm{CT}(*)$: the first one, $\mathrm{NEXP} \stackrel{?}{\subset} \mathrm{P/poly}$, follows the convention we took in Section 1.1, of defining complexity classes based on FP, which in turn is defined as the relativized class $(\mathrm{TheFP})^{\mathcal{O}}$. The second one ignores that convention and

[6] AIV take Peano arithmetic as a formalization for everyday mathematics, but their approach can be framed using ZFC as well.

directly works with the class TheFP, thereby asking $\mathrm{TheNEXP} \stackrel{?}{\subset} \mathrm{TheP/poly}$ (where $\mathrm{The}\mathcal{C}$ is obtained from TheFP in the same way as $\mathcal{C}$ is obtained from FP). This is inevitable when we work with a theory that is an *extension* of everyday mathematics instead of a *restriction*, as one can always ignore any additional axioms and stick to everyday mathematics. For a genuine independence result, we must work with a *subset* of the axioms that govern the mathematical universe. But such a set of axioms for relativization is not known to exist.

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, in particular those related to 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 3 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 works. The vertical gap in the table is a caricature of the current state of the art; to the left of the chasm are facts, and to the right are conjectures apparently out-of-reach. That gap would have been one column to the left had this been the 80s; while the 80s result $\Sigma_2\mathrm{EXP} \not\subset \mathrm{P/poly}$ [31] relativizes, the 90s result $\mathrm{MAEXP} \not\subset \mathrm{P/poly}$ [16], proven using $\mathrm{PSPACE} \subset \mathrm{IP}$, does not, thereby bridging that chasm or breaking that barrier — pick your metaphor. (The reader should be able to visualize the title of this paper, if not now then by the end of this section.)

We now survey each of the four proposals in turn.

1.2.2.1 AIV [5]: The first proposal is from the same paper discussed in Section 1.2.1, by Arora, Impagliazzo, and Vazirani (AIV) [5]. There the authors propose what they call "local checkability" as the key principle underlying $\mathrm{PSPACE} \subset \mathrm{IP}$ and related results such as $\mathrm{MAEXP} \not\subset \mathrm{P/poly}$.

The starting point of AIV is the classical idea, used by Cook to prove the NP-completeness of SAT, that a computation running in time t can be represented as a transcript of t rows, with each row corresponding to the state of the computation at one time step. Cook observed that given a table of t rows, we can verify that it is a valid transcript by inspecting all bits of the table in parallel, where each bit depends on only

	examples for goal (a)			examples for goal (b)	
	$(\exists\mathcal{C}{:}\mathcal{C}\subset\mathrm{NEXP} \wedge \mathcal{C}\not\subset\mathrm{P/poly}) \implies \mathrm{NEXP}\not\subset\mathrm{P/poly}$	$\Sigma_2\mathrm{EXP}\not\subset\mathrm{P/poly}$	$\mathrm{MAEXP}\not\subset\mathrm{P/poly}$	$\mathrm{NEXP}\not\subset\mathrm{P/poly}$	$\mathrm{NP}\not\subset\mathrm{P}$, $\mathrm{EXP}\not\subset$i.o.-$\mathrm{P/poly}$,..
AIV	✓	✓	✓	?	?
For	✓	✓	✓	?	✓
AW	?	✓	✓	✓	✓
IKK	✓	✓	✓	?	✓
this work	✓	✓	✓	✓	✓

Figure 3: Attempts at refining relativization

$O(\log t)$ bits elsewhere. As AIV observe, however, this property will not hold for computations with access to an arbitrary oracle $\mathcal{O}$: just consider the program that takes its input $x_1..x_n$ and outputs $\mathcal{O}(x_1..x_n)$ — the transcript of any execution of this program will have a bit that depends on n bits. This property is called *local checkability* by AIV.

We can interpret AIV's proposal as follows. Local checkability does not hold for an arbitary oracle $\mathcal{O}$, but it does if $\mathcal{O}$ itself can be computed by a locally checkable process. So the AIV framework can be viewed as this: take the Baker-Gill-Solovay framework of relativization, and then restrict the oracle $\mathcal{O}$, from an arbitrary function, to an arbitrary locally checkable function. (AIV present their approach using an oracle-free framework as explained in §1.2.1, with no reference to $\mathcal{O}$, but that presentation is not an integral part of their approach and can be switched out like we do here.)

This framework derives many known nonrelativizing results — including those outside the scope of this paper, such as the PCP theorem — but as AIV point out, whether it can settle questions such as P versus NP or NEXP versus P/poly may be very hard to know. In fact, they observe that if P versus NP were shown beyond reach of their framework in the manner of Baker, Gill, Solovay — by giving contradictory relativizations, $\mathrm{NP}^{\mathcal{O}} \subset \mathrm{P}^{\mathcal{O}}$ and $\mathrm{NP}^{\mathcal{O}} \not\subset \mathrm{P}^{\mathcal{O}}$, using oracles satisfying local checkability — then P would actually be separated from NP. In this sense, the AIV framework is an unsatisfactory candidate for "known techniques". (Note that if all we want is a theory that can derive the current theorems, then we can just let the oracle $\mathcal{O}$ be empty.)

1.2.2.2 Fortnow [22]: In a response to the AIV proposal dated around the same time, Fortnow [22] argues that the nonrelativizing ingredient in $\mathrm{PSPACE} \subset \mathrm{IP}$ and related results is not local checkability; rather, it is something of an algebraic nature.

We can interpret Fortnow's key insight as follows. $\mathrm{PSPACE} \subset \mathrm{IP}$ does not relativize, but it does, if every oracle $\mathcal{O}$ is constrained to have two properties:

(i). Algebraic redundancy. This means, roughly, that if we look at the truth table of $\mathcal{O}$ on inputs of length N, for any N, then we must see a table whose information content is significantly less than 2^N, in much the same way that if we look at the values of a function $f(x) = ax + b$ over an interval in $\mathbb{R}$, say, then we would see a list that can be condensed to merely two entries.

More specifically, $\mathcal{O}$ must encode a family of polynomials $G = \{G_n(x_1, .., x_n)\}_n$ that interpolate a family of Boolean functions $g = \{g_n(z_1, .., z_n)\}_n$ such that

$$G_n(x) = \textstyle\sum_{z \in \{0,1\}^n} g_n(z)\Delta_z(x) \tag{1.4}$$

where $\Delta_z(x)$ denotes the monomial that is 1 if $x = z$, and 0 if $x \neq z$, for all Boolean x.

(ii). Closure. This roughly means that $\mathcal{O}$ is closed under adding redundancy. Just as $\mathcal{O}$ is an algebraically redundant version of a family g by property (i) above, there is an algebraically redundant version of $\mathcal{O}$ itself (after all $\mathcal{O}$ is a family just like g); the closure property dictates that the redundant version of $\mathcal{O}$ must essentially be $\mathcal{O}$ itself — more precisely, it must be efficiently computable given access to $\mathcal{O}$.

We discuss the motivation behind these two properties later below, in conjunction with a related paper (IKK).

The upshot is that Fortnow takes, like AIV essentially do, the Baker-Gill-Solovay framework of relativization, and then restricts the oracle $\mathcal{O}$ to satisfy some constraint; for lack of a better name we refer to this constraint as *closed algebraic redundancy.*

Like AIV, Fortnow does not show any formal limits of his framework. However, we can use the tools we develop in this paper to show that several major conjectures of complexity can provably not be settled within it (hence the ✓ symbol in the table) — alas, we do not know how to show this for NEXP vs. P/poly.

In this sense, Fortnow's framework is (in hindsight given by this thesis) a superior candidate for "known techniques" compared to AIV's, but still an unsatisfactory one in the context of NEXP vs. P/poly.[7]

1.2.2.3 AW [2]: A decade-and-half after the above two papers, Aaronson and Wigderson (AW) [2] introduce algebrization. Their paper is the first one that, after the nonrelativizing results of the 90s, sheds some light on whether "known techniques" — a notion that evidently has expanded during the 90s — can settle questions such as NEXP vs. P/poly.

We can interpret the key insight of AW as follows. In Fortnow's refinement of relativization described just above, recall that any oracle $\mathcal{O}$ must satisfy two properties that we collectively referred to as "closed algebraic redundancy". What AW found is that if we drop the closure requirement from this, then the resulting framework fails to settle many questions of complexity. (Even the NEXP vs. P/poly question cannot be settled, AW found, if we go a step further and broaden the definition of algebraic redundancy, by admitting any low-degree polynomial that extends a Boolean function, and not just those of degree 1; cf. (1.4)).

This is a significant development because neither of the previous works, AIV & Fortnow, show any such limitation of the framework they propose. However, this progress by itself is not enough to yield a satisfactory framework for "known techniques", because such a framework must, as explained in the beginning of this survey, meet two objectives: (a) derive known theorems and (b) fail to settle conjectures. But all that is shown by this insight is that Fortnow's framework, which achieves goal (a), can be weakened to achieve goal (b) — albeit losing goal (a) in the process. (Notice that if all we want is a theory that cannot settle conjectures, then we can just take the empty theory.)

So what remains for AW is to figure out a way of doing what Fortnow did (attain goal (a) using two properties) by using only one of his properties, namely algebraic redundancy.

However, AW fall short of this. As a compromise they finesse the question of how relativization should be refined, by simply declaring that a statement $\mathrm{A} \subset \mathrm{B}$ relativizes algebraically (algebrizes) if $\mathrm{A}^{\mathcal{O}} \subset \mathrm{B}^{\widehat{\mathcal{O}}}$ for every $\mathcal{O}$, where $\widehat{\mathcal{O}}$ is an algebraically redundant version of $\mathcal{O}$. Also they declare 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 (nor for proofs).

Since we ultimately care about containments and their negations, the AW approach might seem appealing. But it only partially meets goal (a) of deriving known theorems, as it takes a quite limited view of "known theorems". For example, the statement

$$(\exists \mathcal{C} : \mathcal{C} \subset \mathrm{NEXP} \ \wedge \ \mathcal{C} \not\subset \mathrm{P}/\mathrm{poly}) \implies \mathrm{NEXP} \not\subset \mathrm{P}/\mathrm{poly} \tag{1.5}$$

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 algebraically relativizing in AW's framework. Consequently, showing that $\mathrm{NEXP} \not\subset \mathrm{P}/\mathrm{poly}$ is non-algebrizing, as AW did, does not rule out whether we can prove $\exists \mathcal{C} : \mathcal{C} \subset \mathrm{NEXP} \ \wedge \ \mathcal{C} \not\subset \mathrm{P}/\mathrm{poly}$ by using solely "algebrizing techniques". (This is alluded to by AW themselves in their paper [2, §10.1].)

On the positive side, the key ideas of AW — how to meet goal (b) of showing unprovability results, using oracles with an algebraic property — influence all subsequent work, including ours.

1.2.2.4 IKK [30]: Motivated by the lack of basic closure properties in the AW framework — of which the above pathology (1.5) is just an example — Impagliazzo, Kabanets, and Kolokolova (IKK) [30] propose an alternative formulation soon after the AW paper.

[7] The NEXP vs. P/poly problem is representative of a host of other open problems whose provability is unknown in Fortnow's framework; see "This work" later in this section.

We can view the approach of IKK as being along the same line as Fortnow's (hence also of AW's) by considering the following fact. Any Boolean formula φ can be extended to non-Boolean values, by viewing each conjunction as multiplication and each negation as subtraction from 1; the resulting expression — called the *arithmetization* of φ — is a low-degree polynomial that agrees with φ on the Boolean values, and that can be efficiently evaluated on all small values (here "low" and "small" means, as usual, polynomial in the size of φ).

Arithmetization of Boolean formulas appears to be the key technique in deriving $\mathrm{PSPACE} \subset \mathrm{IP}$ and related results; it is the one ingredient that clearly stands out in all proofs of nonrelativizing results from the 90s. Invariably, at some point in these proofs, some Boolean formula, used to model some efficient computation, gets arithmetized; notice this step does not seem to go through in the Baker-Gill-Solovay framework of relativization because such a formula φ would involve non-standard gates — oracle gates — yielding subformulas of the form $\mathcal{O}(\varphi_1, .., \varphi_m)$ for which there is no obvious way to proceed.

Now, in both Fortnow's framework and in IKK's, we can interpret the approach as being aimed at making this arithmetization step go through, for as large a class of oracles $\mathcal{O}$ as possible. In Fortnow's case this is achieved by constraining all oracles $\mathcal{O}$ to have "closed algebraic redundancy"; to see how this constraint helps, notice that in arithmetization, the act of replacing a conjunction $x \wedge y$ with a multiplication $x \cdot y$ is nothing other than the act of extending the Boolean function $(x, y) \mapsto x \wedge y$ to non-Boolean values via polynomial interpolation, in other words by adding algebraic redundancy (similarly for $\neg x$ versus $1 - x$). Stated this way, arithmetization easily generalizes to Fortnow's oracles: simply replace each occurrence $\mathcal{O}$ in the formula with its algebraically redundant version, which does no harm because the class of $\mathcal{O}$'s under consideration is closed under adding algebraic redundancy. (Without closure, however, the resulting polynomial is not guaranteed to be efficiently computable, and this is where the AW framework runs into trouble.)

In the framework of IKK, on the other hand, the strategy to enable arithmetization is more direct: they allow $\mathcal{O}$ to be any oracle for which arithmetization, broadly construed, is possible. That is, $\mathcal{O}$ can be any family such that every Boolean formula, possibly with $\mathcal{O}$-gates besides the standard ones ($\wedge$, $\neg$, etc.), has a corresponding low-degree polynomial that extends it to non-Boolean values, and that can be efficiently evaluated given access to $\mathcal{O}$. (IKK present their approach using the oracle-free framework of AIV explained in §1.2.1, with no reference to $\mathcal{O}$, but that presentation is not an integral part of their approach and can be switched out like we do here.)

With this definition, IKK obtain a framework that, for the first time, meets both goal (a) of deriving known theorems, and (b) of failing to resolve conjectures — albeit not for NEXP versus $\mathrm{P/poly}$.[8] In fact, the extent to which IKK show their framework meets goal (b) is identical what we said we can show for Fortnow's framework using the tools we develop in this paper (the $\not\checkmark$ symbol in Figure 3). Thus the IKK framework is not satisfactory for our purposes either.

1.2.2.5 This work. Affine relativization can be roughly viewed as achieving what AW aimed at but fell short of: take Fortnow's framework — relativization with oracles having "closed algebraic redundancy" — and relax it somehow, so that it still meets goal (a) of deriving known theorems, yet it also meets goal (b) of failing to resolve conjectures.

Recall from earlier in this survey that AW did find a relaxation of Fortnow's framework that achieved goal (b), but lost goal (a) in the process — trading off one good thing with another, where both is needed. (As a compromise they came up with an ad-hoc notion whose limitations are discussed earlier in the survey

[8] The NEXP vs. $\mathrm{P/poly}$ problem is representative of a host of other open problems whose provability is unknown in IKK's framework; see "This work" below.

and in the introduction.) In order to fix this situation, the natural thing to try is to aim at a model between Fortnow's and AW's, in the hope of obtaining the best of both worlds.

This is what we essentially manage to do. Our model is simple to state given previous work: relativization with respect to oracles satisfying algebraic redundancy (closed or not). With this basic definition we succeed in (a) deriving all nonrelativizing theorems classified as algebrizing by AW, such as $\mathrm{PSPACE} \subset \mathrm{IP}$, as well as all relativizing theorems, and (b) showing that it is impossible to derive any of the statements classified as non-algebrizing by AW in the classical model of computation, in particular NEXP versus $\mathrm{P/poly}$.

Remark. From a cursory inspection of Figure 3, it might seem as though $\mathrm{NEXP} \not\subset \mathrm{P/poly}$ is the only place where our framework has an edge over Fortnow's and IKK's — a nitpick of sorts. That is only the tip of the iceberg, however; $\mathrm{NEXP} \not\subset \mathrm{P/poly}$ is a representative of a host of other statements whose unprovability can be shown in our framework but is not known for Fortnow's or IKK's — and in some cases even for AW's. To illustrate this, the first author in his PhD thesis [8] took a fairly well-known construction, due to Beigel, Buhrman, and Fortnow [14], of an unrestricted oracle relative to which $\mathrm{P} = \oplus\mathrm{P} \subsetneq \mathrm{NP} = \mathrm{EXP}$, and used it almost verbatim to give an affine oracle for the same statement. This construction does not seem to carry through in any other framework surveyed here. See [8, Chapter 8] for details.

Remark. Whether our definitions imply 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, relativizes affinely, and that the opposite holds for those shown non-algebrizing by IKK — just as is 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 reduction approach in Section 4.3).

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.

1.3.1 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

$$\widehat{\mathcal{O}}(x) := \sum_{b \in \{0,1\}^m} \mathcal{O}(b) \cdot \left(\textstyle\prod_{i=1}^{m} (1 - x_i) \cdot (1 - b_i) + x_i \cdot b_i \right) \tag{1.6}$$

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 (1.6) 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 $+$.

1.3.2 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 a basis that includes $\mathcal{O}$-gates for an arbitrary $\mathcal{O}$, then unlike the case for the standard basis, its arithmetization Φ does not seem to allow efficient evaluation; for example if ϕ is of size n, then it seems Φ_k is not efficiently computable for $k \in \Theta(n)$ — even if we have oracle access to $\mathcal{O}$. Interpreting this to be the nonrelativizing ingredient in proofs of $\mathrm{PSPACE} \subset \mathrm{IP}$ and related results, we take the following approach to refine relativization.

The arithmetic formula Φ is a redundant encoding of the Boolean formula ϕ; it is obtained via a transformation that acts "locally" on ϕ, by taking each of its gates and adding redundancy to it — $\wedge$-gates become $\times$-gates, $\neg$-gates become $(1-.)$-gates, and in general, f-gates become $\widehat{f}$-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.

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 call $\mathcal{A}$ an *affine oracle*.

1.3.3 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 enlarge 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 [22, 2, 30], dating back to the 1994 article of Fortnow [22], 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 the circularity problem without compromising the natural notion we arrived at the previous discussion. Recall our plan from the previous discussion: show that arithmetization, in its current uses, need not introduce redundancy at gates that contain redundancy already. In the notation from above, we want to show that the Boolean property $\pi^{\mathcal{A}}$, when $\mathcal{A}$ contains redundancy already — i.e. when $\mathcal{A}$ is of the form $\widehat{\mathcal{O}}$ for some $\mathcal{O}$ — can be reduced to the arithmetic property $\Pi^{\mathcal{A}}$, rather than to the property $\Pi^{\widehat{\mathcal{A}}}$ as all past works seem to have passively accepted.

1.3.4 Relativizing $\oplus\mathrm{P} \subset \mathrm{IP}$. The plan of the previous discussion can be realized when π is the sum $\pi(\phi) := \oplus_x \phi(x)$, also known as the language $\oplus\mathrm{SAT}$. This is because when ϕ is a formula over the Boolean basis containing $\mathcal{A}$-gates (where $\mathcal{A}$ is of the form $\widehat{\mathcal{O}}$ for some $\mathcal{O}$) each occurrence of $\mathcal{A}$ evaluates the sum (1.6) 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 Boolean basis containing $\mathcal{O}$-gates — not $\mathcal{A}$-gates. After some basic manipulation, we can bring the resulting expression into prenex form, i.e. such that all the sums appear up front.

So given a formula $\phi(x)$ with $\mathcal{A}$ gates, we can write it as a sum $\oplus_z \psi(x, z)$ where ψ is a formula with $\mathcal{O}$-gates. Thus, in the notation of the previous discussion, $\pi^{\mathcal{A}}$ reduces to $\pi^{\mathcal{O}}$ when π is $\oplus\mathrm{SAT}$. Now, we know from the previous discussion that $\pi^{\mathcal{O}} \in \mathrm{IP}^{\mathcal{A}}$. Therefore, $\pi^{\mathcal{A}} \in \mathrm{IP}^{\mathcal{A}}$, and we fulfil the plan of the previous discussion. In other words, $\oplus\mathrm{SAT} \in \mathrm{IP}$ — or equivalently, $\oplus\mathrm{P} \subset \mathrm{IP}$ — relativizes affinely.

This approach can be adapted to show $\mathrm{PSPACE} \subset \mathrm{IP}$ and $\mathrm{NEXP} \subset \mathrm{MIP}$ can be affinely relativizes as well. We describe an alternative approach later in this section.

1.3.5 Showing certain statements are affinely nonrelativizing. 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 oracle $\mathcal{A}$ in our case — relative to which ψ is false.

For some ψ, this task turns out to be not too hard given prior work. Such ψ are of the form $\mathcal{C} \subset \mathcal{D}$, for which AW invented an approach based on communication complexity. A technical observation we make regarding how affine extensions respect disjoint unions (Proposition 11) enables us to import their approach.

For other ψ, however, in particular for $\mathrm{NEXP} \not\subset \mathrm{P/poly}$, we need more significant ideas. If we wanted to show $\mathrm{NEXP} \subset \mathrm{P/poly}$ relative to *some* oracle $\mathcal{O}$, affine or not, or more generally, to show $\mathcal{C}^{\mathcal{O}} \subset \mathcal{D}^{\mathcal{O}}$ for classes $\mathcal{C}$ and $\mathcal{D}$, then there is a simple approach to this, due to Heller [28]: at iteration $n \in \mathbb{N}$, take the first n algorithms underlying $\mathcal{C}^{\mathcal{O}}$, and partially fix $\mathcal{O}$ so as to *force* the behavior of these algorithms on $\{0,1\}^n$. Assuming $\mathcal{C}$ is not too powerful, this forcing can be done without having to fix $\mathcal{O}$ on all of $\{0,1\}^{kn}$, for some constant k, even considering prior iterations. The free inputs of $\{0,1\}^{kn}$ on which $\mathcal{O}$ is yet undefined can then be used to store information on how the forced algorithms behave, in such a way that some algorithm in $\mathcal{D}^{\mathcal{O}}$ can retrieve that information.

When it comes the *affine* oracles, however, we face a difficulty in making this strategy work. An arbitrary affine oracle $\widetilde{\mathcal{O}}$, being the algebraically redundant version (§1.3.1, line (1.6)) of an arbitrary oracle $\mathcal{O}$, is less "dense" in its information content then an arbitrary oracle. So how do we guarantee that partially fixing $\widetilde{\mathcal{O}}$, as done in the previous paragraph, still leaves sufficiently many free inputs on which we can do encoding?

We derive a coding-theoretic ingredient (Lemma 51 and Theorem 52) to provide this guarantee. Roughly, our result says that knowing t bits of a binary codeword exposes at most t bits of its information word in the following sense: there are still $n - t$ bits of the information word that can be set completely independently of each other, if the information word had n bits originally.

Our result can be viewed as improving an earlier, implicit attempt by AW [2] at the same question, who did construct a $\mathcal{Q}$ such that $\mathrm{NEXP}^{\mathcal{Q}} \subset \mathrm{P}^{\mathcal{Q}}/\mathrm{poly}$, albeit 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" [2], 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.

Given a codeword whose t bits are revealed, the AW approach is to show the existence of $n - t$ other codewords such that: (i) each encodes some information word of Hamming weight 1, and (ii) each vanishes on the t positions revealed. Taken together, (i) and (ii) form too many points to interpolate with a degree-≤ 1 polynomial, thus forcing the AW approach to go quadratic.

In contrast, we take a linear algebraic approach and show that 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. See Section 4.1 for details.

1.3.6 Scaling $\oplus\mathrm{P} \subset \mathrm{IP}$ to $\mathrm{PSPACE} \subset \mathrm{IP}$ — a proof sans degree reduction. Our approach for $\oplus\mathrm{P}$ 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., [41, 10, 42, 4]).

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 $\oplus\mathrm{SAT}$. (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 on $\oplus\mathrm{P} \subset \mathrm{IP}$ (§1.3.4). 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 on $\oplus\mathrm{SAT}$, 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 at the end of this section (§1.3.8).

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 $\oplus$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.)

1.3.7 NEXP **vs.** MIP **— the gap amplification perspective.** 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.[9] 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 $1/3$ with $1-\gamma$, 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 [44], that 0-gap-IP is identical to PSPACE. Therefore $\mathrm{PSPACE} \subset \mathrm{IP}$ can be broken into the containments

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

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

The corresponding case for $\mathrm{NEXP} \subset \mathrm{MIP}$ becomes

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

and as our proofs in Section 3.3 suggest, the second containment is the actual content of the theorem.

To put the first containment, $\mathrm{NEXP} \subset$ 0-gap-MIP, into perspective, consider the following variant of the Cook-Levin theorem: every language L in P has circuits that are polylog-time uniform, i.e., L 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 other words, it restricts computations to have polylogarithmic *locality*.

Let $\mathcal{O}$ be an arbitrary language. Relative to $\mathcal{O}$, let $\mathrm{P}_{\mathrm{local}}$ be the subclass of P satisfying this variant of the Cook-Levin theorem. Now, using $\mathrm{P}_{\mathrm{local}}$, define $\mathrm{NP}_{\mathrm{local}}$ and $\mathrm{NEXP}_{\mathrm{local}}$, just like NP and NEXP are defined from P (e.g., §2.3 and §2.6). It is not hard to see that 0-gap-MIP is identical to $\mathrm{NEXP}_{\mathrm{local}}$ (if it is, then see proof of Proposition 39). It is also not hard to see that $\mathrm{NEXP}_{\mathrm{local}}$ is equivalent to the dubious version of NEXP with polynomial-length oracle queries $\mathcal{O}$, 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.)

1.3.8 Compressing $\oplus$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 $\oplus_z \phi(z)$,

[9] 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 verifier: (i) arithmetizes ϕ to get Φ, a formula over the $\widehat{f}$-extended arithmetic basis, (ii) engages in a sumcheck protocol, thus reducing 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 (1.6) in §1.3.1, 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

2.1 We use $A \subset B$ to mean A is a subset of B; we never use '$\subseteq$'. By $\mathrm{poly}(n)$ we mean the set of polynomials $\{n^d + d : d \in \mathbb{N}\}$. By $\mathrm{dom}\, f$ we mean the domain of f.

2.2 Languages and partial languages. A *language* is a function from $\{0,1\}^*$ to $\{0,1\}$. A *partial language* is a function that is or can be extended to a language. We confuse $\{0,1\}$ with $\{\mathrm{False}, \mathrm{True}\}$.

Given a language L and an integer m, we use $L_{\leq m}$ to denote the partial language obtained by restricting L to $\{0,1\}^{\leq m}$.

2.3 Basic complexity classes from FP**.** FP is the set of all $f : \{0,1\}^* \to \{0,1\}^*$ that are efficiently computable. We do not rely on a particular implementation of efficient computability; for concreteness the reader can take the standard definition based on random access Turing machines. We rely on FP being enumerable.

P is obtained by taking each function in FP and projecting its output to its first coordinate.

NP is the set of all languages in $\exists \cdot \mathrm{P}$, where $\exists \cdot \mathcal{C}$ denotes, for a set $\mathcal{C}$ of partial languages, the set of all partial languages L such that

$$L(x) = 1 \implies \exists y \in \{0,1\}^{\ell(|x|)} : (x,y) \in \mathrm{dom}\, V \text{ and } V(x,y)$$
$$L(x) = 0 \implies \forall y \in \{0,1\}^{\ell(|x|)} : (x,y) \in \mathrm{dom}\, V \text{ and } \neg V(x,y)$$

for some $\ell \in \mathrm{poly}(n)$ and $V \in \mathcal{C}$.

$\mathrm{co}\cdot\mathcal{C}$ denotes, for a set $\mathcal{C}$ of partial languages, the set of all partial languages of the form $L(x) = \neg M(x)$ for some $M \in \mathcal{C}$. It is customary to write $\mathrm{co}\mathcal{C}$ for $\mathrm{co} \cdot \mathcal{C}$.

$\forall \cdot \mathcal{C}$ denotes $\mathrm{co} \cdot \exists \cdot \mathcal{C}$. In particular, $\mathrm{coNP} = \forall \cdot \mathrm{P}$.

Define $\Sigma_0\mathrm{P} = \Pi_0\mathrm{P} = \mathrm{P}$, and inductively define $\Sigma_k\mathrm{P}$ as the set of all languages in $\exists \cdot \Pi_{k-1}\mathrm{P}$, and $\Pi_k\mathrm{P}$ as the set of all languages in $\forall \cdot \Sigma_{k-1}\mathrm{P}$. The set $\bigcup_{k\in\mathbb{N}} \Sigma_k\mathrm{P}$ is called the *polynomial-time hierarchy*. Note that $\mathrm{NP} = \Sigma_1\mathrm{P}$ and $\mathrm{coNP} = \Pi_1\mathrm{P}$.

PSPACE, or $\Sigma_\infty\mathrm{P}$, is the set of languages of the form

$$L(x) = \forall y_1 \exists z_1 \cdots \forall y_{t(|x|)} \exists z_{t(|x|)} V(x,y,z)$$

for some $V \in \mathrm{P}$ and $t(n) \in \mathrm{poly}(n)$, where y_i, z_i are quantified over $\{0,1\}$. (We could quantify y_i, z_i over $\{0,1\}^{\ell(|x|)}$ for some $\ell \in \mathrm{poly}(n)$; the definition would be equivalent to the one given.) The justification for this definition of PSPACE comes from the well-known result of Stockmeyer and Meyer [44, Theorem

4.3] that functions computable by a polynomial-space Turing machine are contained in $\Sigma_\infty \mathrm{P}$ (the reverse containment is clear).

BPP is the set of all languages in $\mathsf{Я} \cdot \mathrm{P}$, where $\mathsf{Я} \cdot \mathcal{C}$ denotes, for a set $\mathcal{C}$ of partial languages, the set of all partial languages L such that

$$\begin{aligned} L(x) = 1 &\implies \Pr_{y \in \{0,1\}^{\ell(|x|)}}[(x,y) \in \operatorname{dom} V \text{ and } V(x,y)] > 2/3 \\ L(x) = 0 &\implies \Pr_{y \in \{0,1\}^{\ell(|x|)}}[(x,y) \in \operatorname{dom} V \text{ and } \neg V(x,y)] > 2/3 \end{aligned}$$

for some $\ell \in \mathrm{poly}(n)$ and $V \in \mathcal{C}$.

2.4 Interactive proofs from FP**.** Let $\mathsf{A}x\varphi(x)$ denote $\mathbf{E}_x[\varphi(x)]$, and let $\mathsf{M}x\varphi(x)$ denote $\max_x \varphi(x)$.

AM is the set of all languages L such that

$$\begin{aligned} L(x) = 1 &\implies \mathsf{A}y\, \mathsf{M}z\; V(x,y,z) > 2/3 \\ L(x) = 0 &\implies \mathsf{A}y\, \mathsf{M}z\; V(x,y,z) < 1/3 \end{aligned}$$

and MA is the set of all languages L such that

$$\begin{aligned} L(x) = 1 &\implies \mathsf{M}z\, \mathsf{A}y\; V(x,y,z) > 2/3 \\ L(x) = 0 &\implies \mathsf{M}z\, \mathsf{A}y\; V(x,y,z) < 1/3 \end{aligned}$$

for some $V \in \mathrm{P}$ and $\ell \in \mathrm{poly}(n)$, where y, z are quantified over $\{0,1\}^{\ell(|x|)}$.

Notice that AM is the set of languages in $\mathsf{Я} \cdot \exists \cdot \mathrm{P}$, and MA is the set of languages in $\exists \cdot \mathsf{Я} \cdot \mathrm{P}$. The class $\mathsf{Я} \cdot \exists \cdot \mathrm{P}$ is called prAM, and the class $\exists \cdot \mathsf{Я} \cdot \mathrm{P}$ is called prMA.

IP is the set of languages L such that

$$\begin{aligned} L(x) = 1 &\implies \mathsf{A}y_1\, \mathsf{M}z_1 \cdots \mathsf{A}y_{t(|x|)}\, \mathsf{M}z_{t(|x|)} V(x,y,z) > 2/3 \\ L(x) = 0 &\implies \mathsf{A}y_1\, \mathsf{M}z_1 \cdots \mathsf{A}y_{t(|x|)}\, \mathsf{M}z_{t(|x|)} V(x,y,z) < 1/3 \end{aligned}$$

for some $V \in \mathrm{P}$ and $t \in \mathrm{poly}(n)$, where y_i, z_i are quantified over $\{0,1\}$. (We could quantify y_i, z_i over $\{0,1\}^{\ell(|x|)}$ for some $\ell \in \mathrm{poly}(n)$; the definition would be equivalent to the one given.)

The A-quantifier in these definitions can be thought of as providing the coin tosses of a probabilistic verifier *Arthur*, who interacts with an all-powerful prover *Merlin* corresponding to the M-quantifier. Merlin's goal is to make Arthur accept, which Arthur does iff the "verdict" predicate V, given the input x and transcript (y, z) of the interaction, returns 1. The criteria by which V returns 0 or 1 is typically described as a *protocol* between Merlin and Arthur. The quantity $t(|x|)$ is referred to as the *number of rounds* taken by — or the *round complexity* of — the protocol in computing inputs of length $|x|$.

Power of the Honest Prover. Consider the following subclass of IP. It contains languages L such that whenever $L(x) = 1$, Merlin can just compute a language $\Pi \in \mathcal{C}$ instead of using the M-quantifier. That is, there is a language $\Pi \in \mathcal{C}$ such that for all x, if $L(x) = 1$, then

$$\mathbf{Pr}_z[V(x,y,z)] = \mathsf{A}y_1\, \mathsf{A}y_2 \cdots \mathsf{A}y_{t(|x|)} V(x,y,z) > 2/3, \quad \text{where} \qquad (\P)$$

$$\begin{aligned} z_1 &= \Pi(x, y_1) \\ z_2 &= \Pi(x, y_1 y_2) \\ &\vdots \\ z_t &= \Pi(x, y_1 .. y_{t(|x|)}), \end{aligned}$$

and the case for $L(x) = 0$ remains as before. Any L in this class is said to have interactive proofs where *the power of the honest prover* is in $\mathcal{C}$.

Checkable. We call a language L *checkable* if it has an interactive protocol where the power of the honest prover reduces to L itself. I.e., in (¶), Π reduces to L via a Karp reduction (as defined below in §2.5).

Same-length checkable. We call L *same-length checkable* if it is checkable, and if the reduction from Π to L satisfies the following property: each input of the form (x, y) gets mapped to some string of length $|x|$.

Perfect completeness. Replacing the condition "$> 2/3$" in the above definitions with the condition "$= 1$" yields equivalent definitions [26].

2.5 Defining reductions from FP. As in the previous section (§2.4), let $\mathsf{A}x\varphi(x)$ denote $\mathbf{E}_x[\varphi(x)]$, and let $\mathsf{M}x\varphi(x)$ denote $\max_x \varphi(x)$.

Let F and G be functions into $\{0,1\}^*$ such that $\operatorname{dom} F, \operatorname{dom} G \subset \{0,1\}^*$. We write

$$F \to G$$

and say that F *reduces to* G *via an interactive protocol*, iff there exists $R \in \mathrm{FP}$, $t \in \mathrm{poly}(n)$, and $\varepsilon \in 1/n^{\omega(1)}$, such that for every $x \in \operatorname{dom} F$:

$$\mathsf{A}y_1\, \mathsf{M}z_1 \cdots \mathsf{A}y_{t(n)}\, \mathsf{M}z_{t(n)}[\ F(x) = G(R(x,y,z))\ \vee\ F(x) = R(x,y,z)\ ] \geq 1 - \varepsilon(n)$$
$$\mathsf{A}y_1\, \mathsf{M}z_1 \cdots \mathsf{A}y_{t(n)}\, \mathsf{M}z_{t(n)}[\ F(x) \neq G(R(x,y,z))\ \wedge\ R(x,y,z) \neq \text{'fail'}\ ] \leq \varepsilon(n)$$

where $n = |x|$, and y_i, z_i are quantified over $\{0,1\}$. (Notice that $v = G(u)$ implies $u \in \operatorname{dom} G$.)

We call R an *interactive reduction* from F to G with *round complexity* $t(n)$. We caution that the word "reduction" refers to a function in FP, not to the notion that some F reduces to some G.

Intuitively, as in §2.4, the A-quantifiers in this definition can thought of as Arthur and the M-quantifiers as Merlin. Given x, after sending random coin tosses y_i to Merlin and receiving responses z_i, Arthur uses the predicate R to obtain a string r. Arthur wants either r or $G(r)$ to equal $F(x)$. Merlin can, with high probability over Arthur's coin tosses, ensure that Arthur obtains a desired r. If Merlin is devious, then he has negligible chance in making Arthur obtain a string $r \neq$ 'fail' that is not desired.

We believe this definition to be new. There are three special cases of R being an interactive reduction that capture some classical definitions:

- in a *randomized reduction*, we have $R(x,y,z) = R(x,y)$. Intuitively, Arthur (§2.4) does not need to interact with Merlin (§2.4) to do the reduction.
- in a *Karp reduction*, we have $R(x,y,z) = R(x)$. Notice that $\varepsilon(n) = 0$ in this case. Intuitively, Arthur does not need Merlin's help to do the reduction, nor does he need to flip any coins.
- in a *Cook reduction*, we have $R(x,y,z) = R(x,z)$. Further, for every extension of G to a function G' on $\{0,1\}^*$, for every $x \in \operatorname{dom} F$, and for z satisfying

  $$z_i = G'(R(x, z_1..z_{i-1}))$$

  we have $F(x) = R(x,z)$.
  Notice that $\varepsilon(n) = 0$ in this case. Intuitively, Arthur does not need to flip any coins to do the reduction, and the power of the honest prover is G itself.

We call R a *strong Cook reduction* from F to G, if R is a Cook reduction from F to G, and if $R(x,z) \in \operatorname{dom} G$ for every $x \in \operatorname{dom} F$ and every z satisfying $z_i = G(R(x, z_1..z_{i-1}))$. (This would be the case, for example, when G is a language.) Intuitively, while interacting with Arthur to convince him of the value of $F(x)$, the honest prover never gets asked a question outside $\operatorname{dom} G$.

By default, all Cook reductions are strong. By default, all reductions are Karp.

The "reduces to via an interactive reduction" relation is transitive: $F \to G$ together with $G \to H$ imply $F \to H$. Further, "reduces to via a Karp reduction", "reduces to via a randomized reduction", "reduces to via a *strong* Cook reduction", are all transitive relations.

2.6 General time classes from FP. Let $T \subset n^{\omega(1)}$ be a class of functions each of which is computable in FP. Suppose that T is closed under taking polynomials in the following sense: for every $t \in T$ and $d \in \mathbb{N}$, there is some $t' \in T$ such that $t^d(n) < t'(n)$ for every n.

Define $\mathrm{DTIME}(T)$ as the set of languages L for which there exists $K \in \mathrm{P}$ and $t \in T$ such that $L(x) = K(x, 1^{t(|x|)})$ for every x.

Define $\mathrm{NTIME}(T)$, $\Sigma_2\mathrm{TIME}(T)$, $\mathrm{MATIME}(T)$, etc., in the same way, except by picking K respectively from NP, Σ_2P, MA, etc.

Use E, NE, Σ_2E, MAE, etc., to denote respectively $\mathrm{DTIME}(\mathrm{linexp}(n))$, $\mathrm{NTIME}(\mathrm{linexp}(n))$, $\Sigma_2\mathrm{TIME}(\mathrm{linexp}(n))$, $\mathrm{MATIME}(\mathrm{linexp}(n))$, etc., where $\mathrm{linexp}(n)$ is the set $\{2^{cn} : c \in \mathbb{N}\}$.

Use EXP, NEXP, Σ_2EXP, MAEXP, etc., to denote respectively $\mathrm{DTIME}(\exp(n))$, $\mathrm{NTIME}(\exp(n))$, $\Sigma_2\mathrm{TIME}(\exp(n))$, $\mathrm{MATIME}(\exp(n))$, etc., where $\exp(n)$ is the set $\{2^{cn^d} : c, d \in \mathbb{N}\}$.

2.7 Relativized classes. For every language $\mathcal{O}$, we define the class $\mathrm{FP}^{\mathcal{O}}$ — "FP *relative to* $\mathcal{O}$," or "FP *with oracle access to* $\mathcal{O}$" — as the set of all functions from $\{0,1\}^*$ to $\{0,1\}^*$ that Cook-reduce to $\mathcal{O}$.

All definitions built on FP (§2.3-§2.6) naturally generalize to their *relativized* versions: NP to $\mathrm{NP}^{\mathcal{O}}$, IP to $\mathrm{IP}^{\mathcal{O}}$, MAEXP to $\mathrm{MAEXP}^{\mathcal{O}}$, etc. When we say "$L$ is checkable with oracle access to $\mathcal{O}$", for example, we mean to replace FP with $\mathrm{FP}^{\mathcal{O}}$ in the definition for a language to be checkable (§2.4), and then declare L as checkable.

2.8 Oracle access capability. Given $r \in \mathrm{poly}(n)$ and $f \in \mathrm{FP}$, consider the function

$$f^* : (\mathcal{O}, x) \mapsto f^{\mathcal{O}}(x)$$

that takes as input any language $\mathcal{O}$ and string x, and outputs $g(x)$, where f is a Cook reduction of round complexity r (§2.5) from the language g to $\mathcal{O}$. We call the set of all such f^*, over all $f \in \mathrm{FP}$ and all $r \in \mathrm{poly}(n)$, the class FP^* — "FP *with oracle access capability*".

For readability, we always use the notation in the previous paragraph: a starred symbol such as f^* denotes a member of FP^*, and its un-starred version f denotes the member of FP on which f^* is based.

For each $f^* \in \mathrm{FP}^*$, we say f^* *has oracle access capability*. We use $f^{\mathcal{O}}$ to denote $f^*(\mathcal{O}, \cdot)$, the restriction of f^* obtained by setting its first argument to $\mathcal{O}$, and refer to $f^{\mathcal{O}}$ as f^* *when given access to* $\mathcal{O}$.

All definitions built on FP (§2.3-§2.6) naturally generalize to their *oracle-access-capable* versions. For example,

$$\begin{aligned}\mathrm{NP} := \{L : \exists V \in \mathrm{P},\ \exists \ell \in \mathrm{poly}(n),\ \forall x \text{ string} \\ L(x) = 1 \iff \exists y \in \{0,1\}^{\ell(|x|)}\ V(x,y) \quad \}\end{aligned}$$

generalizes to

$$\begin{aligned}\mathrm{NP}^* := \{L^* : \exists V^* \in \mathrm{P}^*,\ \exists \ell \in \mathrm{poly}(n),\ \forall x \text{ string},\ \forall \mathcal{O} \text{ language}, \\ L^*(\mathcal{O}, x) = 1 \iff \exists y \in \{0,1\}^{\ell(|x|)}\ V^*(\mathcal{O}, x, y) \quad \}.\end{aligned} \tag{2.1}$$

Similarly IP generalizes to IP^*, etc. When we say "there is an interactive protocol where the verifier has oracle access capability", for example, we are merely referring to a function in IP^*.

2.9 Enumeration. We take it as a fact that FP is enumerable. It follows that every class defined above (§2.3-§2.8) is enumerable. For example, to find an enumeration of NP^*, by (2.1) it suffices to find an enumeration of P^* and cross with $\mathbb{N}$. To find an enumeration of P^*, it suffices to find an enumeration of FP^* since P^* is obtained by taking every function in FP^* and projecting its output to the first coordinate (§2.8, §2.3). Finally, to find an enumeration of FP^*, it suffices to take an enumeration for FP and cross it with $\mathbb{N}$, because by definition (§2.8), underlying every $f^* \in \mathrm{FP}$ are some $f \in \mathrm{FP}$ and some $r \in \mathrm{poly}(n)$.

2.10 Query complexity. Let $f^* \in \mathrm{FP}^*$. By definition (§2.8), underlying f^* are some $r \in \mathrm{poly}(n)$ and $f \in \mathrm{FP}$, where f is a Cook reduction of round complexity r (§2.5). We refer to r as the *query complexity* of f^*.

Let $g^* \in \mathrm{P}^*$. By definition (§2.3), underlying g^* is some $f^* \in \mathrm{FP}^*$. By the query complexity of g^* we mean that of f^*.

2.11 Boolean bases. By default, all circuits (hence all formulas) are over the *standard Boolean basis* $B_{\mathrm{std}} := \{\mathbf{0}, \mathbf{1}, \wedge, \oplus\}$, where $\mathbf{0}$ is the all-zeroes language and $\mathbf{1}$ is the all-ones, $\wedge$ maps x to $\wedge_i x_i$ and $\oplus$ maps x to $\oplus_i x_i$.

More generally, by a Boolean basis we mean any finite set B containing B_{std}, comprising languages and partial languages of the form $L_{\leq m}$ (§2.2) for some $m \in \mathbb{N}$ and language L. We refer to $B \cup \{f\}$ as the basis B *extended with* f, and when $B = B_{\mathrm{std}}$, as the *f-extended basis*. We call f *eligible* if the f-extended basis is defined (i.e., if f is a language, or a partial language of the form $L_{\leq m}$).

When representing circuits (hence formulas) over extended bases $B \supsetneq B_{\mathrm{std}}$, we assume a generic labeling of gates — using labels such as 'the i^{th} nonstandard element' — so that a given circuit can be interpreted over different bases.

2.12 Well-behaved resource bound. 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 that maps the binary encoding of n to the binary encoding of $s(n)$ 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.

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

2.13 Languages as families. We sometimes specify a language $L : \{0,1\}^* \to \{0,1\}$ as a family of Boolean functions $\{L_n : \{0,1\}^n \to \{0,1\}\}_{n \in \mathbb{N}}$, and sometimes 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, here is one way to do this: 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 $n = s(m \diamond k, m \diamond k)$.)

2.14 The (partial) language $\oplus\mathrm{SAT}^f$. For every Boolean basis B and eligible (§2.11) f, define $\oplus\mathrm{SAT}^f$ as the map

$$\phi(x) \mapsto \oplus_{\alpha \in \{0,1\}^n} \phi(\alpha)$$

where ϕ is a formula over the basis $B \cup \{f\}$, with n inputs $x_1..x_n$. By default B is the standard basis and f is the all-zeroes language.

$\oplus\mathrm{SAT}^f$ is undefined on those $\phi(x)$ that are undefined for some setting α of its inputs x (due to some gate of ϕ receiving inputs out of its domain). So $\oplus\mathrm{SAT}^f$ is a language when the basis B comprises entirely of languages and f is also a language, which is the case by default.

We index $\oplus\mathrm{SAT}$ by n, any upper bound on the number of nodes 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 $s \in \mathrm{poly}(n)$, with each such string representing a formula ϕ of at most n nodes.

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.

2.15 The (partial) language $\Sigma_k\mathrm{SAT}^f$. For every Boolean basis B and eligible (§2.11) f, and for every $k \in \mathbb{N} \cup \{\infty\}$, define $\Sigma_k\mathrm{SAT}^f$ as the map

$$\phi(X_1, .., X_k) \mapsto \exists \alpha_1 \in \{0,1\}^{n_1} \forall \alpha_2 \in \{0,1\}^{n_2} \ldots \mathsf{Q}\alpha_k \in \{0,1\}^{n_k}\, \phi(\alpha_1..\alpha_k)$$

where Q is $\exists$ or $\forall$ depending respectively on k being odd or even, and where ϕ is a formula over the basis $B \cup \{f\}$ with k *sets* of inputs: the X_1 inputs $X_{1,1}..X_{1,n_1}$, the X_2 inputs $X_{2,1}..X_{2,n_2}$, and so on.

When $k = \infty$, there is no bound on k — other than the size of the formula ϕ, that is — and without loss of generality, $n_1 = \cdots = n_k = 1$. By default B is the standard basis and f is the all-zeroes language.

$\Sigma_k\mathrm{SAT}^f$ is undefined on those ϕ that are undefined for some setting of its inputs (due to some gate of ϕ receiving inputs out of its domain). So $\Sigma_k\mathrm{SAT}^f$ is a language when the basis B comprises entirely of languages and f is also a langauge, which is the case by default.

2.16 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 $\{p_k\}_{k\in\mathbb{N}}$ of such irreducibles, i.e., we use a function in FP that outputs p_k given k in unary [43].

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.)

2.17 Affine extensions and affine oracles. Given $f_m : \{0,1\}^m \to \{0,1\}$, we define its *affine extension polynomial* $\widehat{f_m}$ 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 (§2.16) 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 here 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 §2.13, $\mathcal{A}$ 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 oracle*, we mean the affine extension of a language.

2.18 Affine extensions respect disjoint unions. The disjoint union of languages $\mathcal{O}_0$ and $\mathcal{O}_1$ is the language $\mathcal{O}_0 \amalg \mathcal{O}_1 : (b, x) \mapsto \mathcal{O}_b(x)$. Affine extensions are compatible with disjoint unions in the following sense:

Proposition 11. *For every pair of languages $\mathcal{O}_0, \mathcal{O}_1$.*

$$\mathrm{FP}^{\widetilde{\mathcal{O}_0} \amalg \widetilde{\mathcal{O}_1}} = \mathrm{FP}^{\widetilde{\mathcal{O}_0 \amalg \mathcal{O}_1}}.$$

In words, the disjoint union of the affine extension of $\mathcal{O}_0$ and the affine extension of $\mathcal{O}_1$ is equivalent, under Cook reductions, to the affine extension of the disjoint union of $\mathcal{O}_0$ and $\mathcal{O}_1$.

Proof. Let $\mathcal{O} := \mathcal{O}_0 \coprod \mathcal{O}_1$. By definition (§2.17), $\widetilde{\mathcal{O}}$ is the Boolean version (§2.16) of the function that evaluates, given $B, X_1, .., X_n \in \mathbb{F}_{2^k}$ for any k, the polynomial

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

It follows that $\widetilde{\mathcal{O}} \in \mathrm{P}^{\widetilde{\mathcal{O}_0} \amalg \widetilde{\mathcal{O}_1}}$ and $\widetilde{\mathcal{O}_0} \in \mathrm{P}^{\widetilde{\mathcal{O}}}$ and $\widetilde{\mathcal{O}_1} \in \mathrm{P}^{\widetilde{\mathcal{O}}}$, implying the claim. □

2.19 Relativization. We say that a statement *holds relative to* the language $\mathcal{O}$, if the statement can be proven when FP is redefined to be $\mathrm{FP}^{\mathcal{O}}$ and the standard Boolean basis is redefined to be the $\mathcal{O}$-extended basis. (See Section 1.1 for a metamathematical exposition.)

2.20 P and the Cook-Levin theorem. For a family of circuits $C := \{C_n\}_n$, define

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

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

Of the two statements

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

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

we call (i) the Cook-Levin theorem, and (ii) the strong Cook-Levin theorem. We take for granted that (i) holds relative to every language. (It is a consequence of Proposition 43 that (ii) does not, even relative to every affine oracle.)

3 Positive Relativization Results

This section shows that the famous results on interactive proofs affinely relativize, as do the circuit lower bounds that build on them. (By §1.1.4, it follows that these results do not have *proofs* that affinely relativize.) 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, Thierauf, and by Santhanam (Section 3.4). All of these build on several properties of $\oplus\mathrm{SAT}$ developed in Section 3.1.

3.1 Checking and Compressing $\oplus$SAT

This section develops three results on $\oplus$SAT that enable most of the positive relativization results in the paper. The reader is referred to Section 2 for all undefined terms and notation used in this section.

The first main result in this section shows the existence of a $\oplus$P-complete language that is same-length checkable (§2.4), and that this affinely relativizes.

Theorem 12 (Checking $\oplus$SAT)**.** *$\oplus$SAT is checkable (§2.4). In fact, there a language K that is* same-length *checkable such that $\oplus\mathrm{SAT} \to K$ and $K \to \oplus\mathrm{SAT}$.*

This holds relative to every affine oracle.

Theorem 12 is used, from different aspects, in deriving Shamir's IP theorem (§3.2) and the circuit lower bounds of Buhrman et al. and of Santhanam (§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 arbitrarily large d and a fixed c, as long as the L-gates have fan-in $O(n)$ in the original formula. (The round complexity 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 13 (Compressing $\oplus$SAT)**.** *There is a function $s(m,n) \in \mathrm{poly}(m \log n)$ such that the following holds. There is an interactive reduction (§2.5) that for every language L, maps instances of $\oplus\mathrm{SAT}^{L\leq m}$ of size n, to instances of $\oplus\mathrm{SAT}^{L\leq m}$ of size $s(m,n)$, for every m, n.*

This holds relative to every affine oracle.

Theorem 13 is used in deriving Shamir's IP theorem (Section 3.2) with a new streamlined proof of that result.

We also derive two auxiliary facts that will be useful in the sequel:

Proposition 14. *$\mathcal{A} \to \oplus\mathrm{SAT}^{\mathcal{O}}$ for every language $\mathcal{O}$ and its affine extension $\mathcal{A}$.*

Proposition 15. *For every function $R \in \mathrm{FP}$ there is a function $R' \in \mathrm{FP}$ such that the following holds. If R is a Cook reduction (§2.5) from some eligible f to some eligible g (§2.11), then R' is a reduction (§2.5) from $\oplus\mathrm{SAT}^f$ to $\oplus\mathrm{SAT}^g$ that works over any Boolean basis (§2.11).*

This holds relative to every language.

In the rest of Section 3.1 we prove the four claims above.

3.1.1 Organization of the Proofs

We prove the above four claims, Theorems 12-13 and Propositions 14-15, in four steps:

- In §3.1.2 we define an arithmetic analogue of $\oplus$SAT called $+$ASAT, and state several lemmas relating the two (Lemmas 18-21).
- In §3.1.3 we derive Theorems 12-13 assuming Propositions 14-15 and the lemmas of the first step.
- In §3.1.4 we extend $\oplus$SAT and $+$ASAT to expressions involving summations *within* the formula, not just in front. We call these extensions $\oplus^*$SAT and $+^*$ASAT, respectively, and derive several facts relating $\oplus^*$SAT, $\oplus$SAT, $+^*$ASAT, and $+$ASAT.
- In §3.1.5 we use the facts derived in step 3 to give the remaining proofs.

3.1.2 $+\mathrm{ASAT}$, an Arithmetic Analogue of $\oplus\mathrm{SAT}$

We now define an arithmetic analogue of $\oplus\mathrm{SAT}$.

Definition 16 (Arithmetic basis)**.** For every Boolean basis (§2.11) 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 $\widehat{B}_{\mathrm{std}}$ where B_{std} is the standard Boolean basis.

As is the case with Boolean bases (§2.11), when representing circuits (hence formulas) over extended bases $\widehat{B} \supsetneq \widehat{B}_{\mathrm{std}}$, we assume a generic labeling of gates — using labels such as 'the i^{th} nonstandard element' — so that a given circuit can be interpreted over different bases.

Definition 17 ($+\mathrm{ASAT}^f$)**.** For every Boolean basis B and eligible f (§2.11), define $+\mathrm{ASAT}^f$ as the Boolean version (§2.16) of the map $\Phi(\vec{x}) \mapsto \Sigma_{\vec{\alpha}}\Phi(\vec{\alpha})$, where Φ denotes a formula over the arithmetic basis corresponding to $B \cup \{f\}$, that has all its constants in $\mathbb{F}_{2^k}$ for some k. By default B is the standard basis and f is the all-zeroes language.

We index (§2.13) $+\mathrm{ASAT}$ 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 become clear in the proof of Lemma 21) we require $k \geq \log^2 n$, i.e., if $k < \log^2 n$ then $+_k\mathrm{ASAT}_n$ behaves trivially, say by returning 0. We also 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.

Four lemmas regarding $\oplus\mathrm{SAT}$ and $+\mathrm{ASAT}$ are used in proving Theorems 12-13:

Lemma 18. $\oplus\mathrm{SAT}^{\widetilde{f}} \to \oplus\mathrm{SAT}^f$. *The reduction works over any Boolean basis and any eligible f (§2.11).*

Lemma 19. $\oplus\mathrm{SAT} \to +\mathrm{ASAT}$. *The reduction works over any Boolean basis and its corresponding arithmetic basis. In addition, the same reduction yields $\oplus\mathrm{SAT}_n \to +_{k(n)}\mathrm{ASAT}$ for some $k \in \mathrm{poly}\log n$.*

Lemma 20. $+\mathrm{ASAT} \to \oplus\mathrm{SAT}$. *The reduction works over any Boolean basis and its corresponding arithmetic basis.*

Lemma 21. *There is a function $p \in \mathrm{poly}(n)$ and an interactive protocol that yields the following:*

i. $+\mathrm{ASAT}$ *is same-length checkable.*

ii. $+_k\mathrm{ASAT}^{L^{\leq m}}$ *reduces to* $\oplus\mathrm{SAT}^{L^{\leq m}}_{p(km)}$ *for every language L and for every $k, m \in \mathbb{N}$.*

More generally, there is an interactive protocol where the verifier has oracle access capability (§2.8), such that for every language $\mathcal{O}$, when the verifier is given access (§2.8) to $\widetilde{\mathcal{O}}$, the protocol yields (i) and (ii) for $+\mathrm{ASAT}^{\mathcal{O}}$ instead of $+\mathrm{ASAT}$.

We defer the proof of these four lemmas to §3.1.5.

3.1.3 Proofs of Theorems 12-13

We now derive Theorems 12-13, assuming Propositions 14-15 and Lemmas 18-21.

Proof of Theorem 12. Let $\mathcal{O}$ be a language and $\mathcal{A}$ its affine extension. We show that $\oplus\mathrm{SAT}^{\mathcal{A}}$ reduces, to and from, some language K that is same-length checkable (§2.4) with oracle access (§2.7) to $\mathcal{A}$. We also show how this implies that $\oplus\mathrm{SAT}^{\mathcal{A}}$ is itself checkable with oracle access to $\mathcal{A}$.

Put $K := +\mathrm{ASAT}^{\mathcal{O}}$. We claim that

$$\oplus\mathrm{SAT}^{\mathcal{A}} \to +\mathrm{ASAT}^{\mathcal{O}} \tag{3.1}$$

and that

$$+\mathrm{ASAT}^{\mathcal{O}} \to \oplus\mathrm{SAT}^{\mathcal{A}}. \tag{3.2}$$

To see the first claim, use Lemma 18 to get $\oplus\mathrm{SAT}^{\mathcal{A}} \to \oplus\mathrm{SAT}^{\mathcal{O}}$, and then use Lemma 19 with the $\mathcal{O}$-extended Boolean basis (§2.11) to get $\oplus\mathrm{SAT}^{\mathcal{O}} \to +\mathrm{ASAT}^{\mathcal{O}}$.

For the second claim, use Lemma 20 with the $\mathcal{O}$-extended Boolean basis to get $+\mathrm{ASAT}^{\mathcal{O}} \to \oplus\mathrm{SAT}^{\mathcal{O}}$, and then use Proposition 15, together with the fact that $\mathcal{O}$ Karp-reduces, hence Cook-reduces, to $\mathcal{A}$, to get $\oplus\mathrm{SAT}^{\mathcal{O}} \to \oplus\mathrm{SAT}^{\mathcal{A}}$.

Now, by Lemma 21, we know that $+\mathrm{ASAT}^{\mathcal{O}}$ is same-length checkable with oracle access to $\mathcal{A}$. But then $\oplus\mathrm{SAT}^{\mathcal{A}}$ is also checkable: On input x, use the reduction (3.1) to get an input x' for $+\mathrm{ASAT}^{\mathcal{O}}$, and then simulate the checking protocol for $+\mathrm{ASAT}^{\mathcal{O}}$, by using the reduction (3.2) to translate each query for $+\mathrm{ASAT}^{\mathcal{O}}$ to one for $\oplus\mathrm{SAT}^{\mathcal{A}}$.

(Theorem 12, mod Proposition 15 and Lemmas 18-21)
□

Proof of Theorem 13. Let $\mathcal{O}, L$ be two languages. Let $\mathcal{A}$ be the affine extension of $\mathcal{O}$. We are to show that there is some $s(m,n) \in \mathrm{poly}(m \log n)$, and an interactive protocol with oracle access (§2.7) to $\mathcal{A}$, that yield the reduction

$$\oplus\mathrm{SAT}_n^{L_{\leq m}} \to \oplus\mathrm{SAT}_{s(m,n)}^{L_{\leq m}},$$

over the $\mathcal{A}$-extended Boolean basis (§2.11) for every n, m. Equivalently, recalling from §2.14 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, we are to show

$$\oplus\mathrm{SAT}_n^{\mathcal{A},L_{\leq m}} \to \oplus\mathrm{SAT}_{s(m,n)}^{\mathcal{A},L_{\leq m}}$$

over the standard basis, which is what we do now.

We have, over the standard basis,

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

where the first reduction is by Lemma 18 (with the $L_{\leq m}$-extended basis) and the second by Lemma 19 (with the basis extended by $\mathcal{O}$ and $L_{\leq m}$). In fact, the same sequence yields

$$\oplus\mathrm{SAT}_n^{\mathcal{A},L_{\leq m}} \to +_{k(n)}\mathrm{ASAT}^{\mathcal{O},L_{\leq m}},$$

for some $k \in \mathrm{poly}(\log n)$. Now by Lemma 21 (with the $\mathcal{O}$-extended Boolean basis), there is a polynomially bounded function p, and an interactive protocol with oracle access to $\mathcal{A}$, that yield

$$+_{k(n)}\mathrm{ASAT}^{\mathcal{O},L_{\leq m}} \to \oplus\mathrm{SAT}_{p(mk(n))}^{L_{\leq m}},$$

completing the proof when we put $s(m,n) := p(mk(n))$.

(Theorem 13, mod Lemmas 18-21)
□

3.1.4 Extending $\oplus\mathrm{SAT}$ to $\oplus^{*}\mathrm{SAT}$ and $+\mathrm{ASAT}$ to $+^{*}\mathrm{ASAT}$

We now extend $\oplus\mathrm{SAT}$ and $+\mathrm{ASAT}$ to expressions involving summations *within* the formula, not just in front. We give four definitions, two for extending $\oplus\mathrm{SAT}$ and two for $+\mathrm{ASAT}$.

Definition 22 (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 defined on m inputs, 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 23 ($\oplus^*\mathrm{SAT}^f$)**.** For every Boolean basis B and eligible f (§2.11), 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 all-zeroes language.

Definition 24 (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 inputs, 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 25 ($+^*\mathrm{ASAT}^f$)**.** For every arithmetic basis A and eligible f (§2.11), Define $+^*\mathrm{ASAT}^f$ over the basis A as the Boolean version (§2.16) of the map $\Psi(\vec{x}) \mapsto \Sigma_{\vec{\alpha}} \Psi(\vec{\alpha})$, where 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 all-zeroes language.

We derive six facts relating $\oplus\mathrm{SAT}$, $\oplus^*\mathrm{SAT}$, $+\mathrm{ASAT}$ and $+^*\mathrm{ASAT}$:

Lemma 26. *For every function $R \in \mathrm{FP}$ there is a function $R' \in \mathrm{FP}$ such that the following holds. If R is a Cook reduction (§2.5) from some eligible f to some eligible g (§2.11), then R' is a reduction from $\oplus\mathrm{SAT}^f$ to $\oplus^*\mathrm{SAT}^g$ that works over any Boolean basis.*

This holds relative to every language.

Lemma 27. $\oplus\mathrm{SAT}^{\oplus\mathrm{SAT}} \to \oplus^*\mathrm{SAT}$. *The reduction works over any Boolean basis.*

Lemma 28. $\oplus^*\mathrm{SAT} \to +^*\mathrm{ASAT}$. *The reduction works over any Boolean basis and its corresponding arithmetic basis.*

Lemma 29. $+^*\mathrm{ASAT} \to +\mathrm{ASAT}$. *The reduction works over any arithmetic basis.*

Lemma 30. $+\mathrm{ASAT} \to \oplus\mathrm{SAT}$. *The reduction works over any Boolean basis and its corresponding arithmetic basis.*

Corollary 31. $\oplus^*\mathrm{SAT} \to \oplus\mathrm{SAT}$. *The reduction works over any Boolean basis.*

We now proceed to prove each of these facts in turn. Before we begin, we derive an auxiliary fact that will be useful in proving the first two facts, Lemmas 26 and 27.

Lemma 32. *For every function $R \in \mathrm{FP}$ there is a function $R' \in \mathrm{FP}$ such that the following holds. If R is a Cook reduction (§2.5) from some eligible f to some eligible g (§2.11), then for every n, $R'(1^n)$ gives a formula $\xi(x,y)$ over the g-extended Boolean basis such that*

$$f(x) = \oplus_y \xi(x,y)$$

for every $x \in \{0,1\}^n \cap \mathrm{dom}\, f$.

This holds relative to every language.

Proof. Let $\mathcal{O}$ be an arbitrary language, and let $R \in \mathrm{FP}^{\mathcal{O}}$ be a Cook reduction from the partial language f to the partial language g. By definition, this means there is some $\ell \in \mathrm{poly}(n)$ such that for every $x \in \mathrm{dom}\, f$,

$$f(x) = R(x,z), \quad \text{where } z_i = g(R(x, z_1..z_{i-1})) \text{ and } |z| = \ell(|x|). \tag{3.3}$$

(For a general Cook reduction, we would use in (3.3) an arbitrary language g' extending g, instead of g, but recall from §2.5 that Cook reductions are *strong* by default.)

By the Cook-Levin theorem (§2.20) applied to R, it follows that there is $\ell \in \mathrm{poly}(n)$ such that for every $n \in \mathbb{N}$, there are circuits $C_0, .., C_{\ell(n)}$, each over the $\mathcal{O}$-extended Boolean basis, such that for every $x \in \{0,1\}^n \cap \mathrm{dom}\, f$,

$$f(x) = C_{\ell(n)}(x, z), \quad \text{where } z_i = g(C'_{i-1}(x, z_1..z_{i-1})) \text{ and } |z| = \ell(|x|),$$

and where

$$C'_i(y) := (C_i(y))_{1..|R(y)|},$$

the idea being to calculate the output length of $R(y)$ and trim the excess from the output of $C_i(y)$ before making any use of it. Notice that the function that calculates the output length of R is in $\mathrm{FP}^{\mathcal{O}}$, because R is. Hence by the Cook-Levin theorem again, each C'_i can be implemented by a circuit; moreover, each such circuit can be produced by a function in $\mathrm{FP}^{\mathcal{O}}$ given 1^n.

It follows that for every $x \in \{0,1\}^n \cap \mathrm{dom}\, f$,

$$f(x) = \bigoplus_{z \in \{0,1\}^\ell} C_\ell(x, z) \wedge \textstyle\bigwedge_{i=1..\ell}(z_i \equiv g(C'_{i-1}(x, z_{1..i-1}))$$

where ℓ denotes $\ell(n)$; further, the righthand side is produced by some function in $\mathrm{FP}^{\mathcal{O}}$ given input 1^n.

The expression inside the sum is a circuit $E(x, z)$ over the g- and $\mathcal{O}$-extended basis, and can be equivalently written as the sum $\oplus_v \xi(x, z, v)$ where $\xi(a, v)$ checks that v describes the computation of $E(a)$, and v ranges over $\{0,1\}^s$ for some $s \in \mathrm{poly}(\text{size of } E) \subset \mathrm{poly}(n)$.

It follows that there is a function in $R' \in \mathrm{FP}^{\mathcal{O}}$ that, given input 1^n, outputs a formula ξ over the g- and $\mathcal{O}$-extended basis satisfying

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

for every $x \in \{0,1\}^n \cap \mathrm{dom}\, f$. This was to be shown. □

Proof of Lemma 26. Suppose f Cook-reduces to g. By Lemma 32, there is a function in FP that, given a formula ϕ over the basis $B \cup \{f\}$, where B is any Boolean basis, takes each subformula of the form $f(\phi_1..\phi_n)$ and performs the replacement

$$f(\phi_1..\phi_n) \mapsto \textstyle\bigoplus_y \xi(\phi_1..\phi_n, y)$$

where ξ is a formula over the basis $B \cup \{g\}$. This shows $\oplus\mathrm{SAT}^f \to \oplus^*\mathrm{SAT}^g$ over B, for every basis B. □

Proof of Lemma 27. Let B be a Boolean basis for formulas. Given a formula $\phi(x)$ over the basis $B \cup \{\oplus\mathrm{SAT}^B\}$, 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.

Intuitively, replacing each occurrence of $\oplus\mathrm{SAT}^B$ in $\phi(x)$ with the actual sum to be computed, would constitute a reduction as desired. More precisely, let $\mathrm{FormulaEval}^B$ 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 $\mathrm{FormulaEval}^B$ output $\tau(u)$ only if the extra bits in u are set to zero, else let it output zero. Also, in case $\tau(u)$ is undefined — which may happen if some nonstandard gate in τ receives inputs out of its domain — then let $\mathrm{FormulaEval}^B$ be undefined on (t, u).)

Each subformula in $\phi(x)$ of the form

$$\oplus\mathrm{SAT}^B(\phi_1..\phi_m) \tag{3.4}$$

can be viewed as the sum

$$\oplus_{u\in\{0,1\}^m}\mathrm{FormulaEval}^B(\phi_1..\phi_m,u) \tag{3.5}$$

for each setting of x, since the subformulas $\phi_1(x),..,\phi_m(x)$ collectively describe a Boolean formula τ_x with $\leq m$ input variables.

Now, $\mathrm{FormulaEval}^B$ Cook-reduces to the basis B, more precisely, to the partial language

$$\amalg B : (i,x)\mapsto B_i(x)$$

where B_i is the i^{th} element of the basis B. Notice that this reduction does not depend on what the basis B is, provided we have a reasonable representation of formulas that uses generic labels for gates — 'the i^{th} nonstandard element' etc. — which is the case by the way we set things up in §2.11.

It follows by Lemma 32 that there is a function in FP that, given input 1^m, outputs a formula $\xi^{\amalg B}$ over the basis $\amalg B$ satisfying

$$\mathrm{FormulaEval}^B(a)=\bigoplus_y \xi^{\amalg B}(a,y) \tag{3.6}$$

for every input $a\in\{0,1\}^m$ on which the left hand side is defined. In case the left hand side is undefined, then so is the right hand side.

If the basis B contains d elements, then $\amalg B$ can be written as

$$\amalg B(i,x)=((i\equiv 1)\wedge B_1(x))\oplus\cdots\oplus((i\equiv d)\wedge B_d(x)) \tag{3.7}$$

where '$i\equiv j$' is shorthand for the formula checks that i is the binary encoding of the number j. The righthand side of (3.7) is a formula over the basis B. Combining with (3.6), we get a function in FP that, given input 1^m, outputs a formula ξ^B over the basis B satisfying

$$\mathrm{FormulaEval}^B(a)=\bigoplus_y \xi^B(a,y) \tag{3.8}$$

for every input $a\in\{0,1\}^m$.

It follows, from (3.5) and (3.8), that there is a function in FP that takes each subformula of the form (3.4), and performs the replacement

$$\oplus\mathrm{SAT}^B(\phi_1..\phi_m)\mapsto\bigoplus_{u,y}\xi^B(\phi_1..\phi_m,u,y)$$

proving $\oplus\mathrm{SAT}^{\oplus\mathrm{SAT}}\to\oplus^*\mathrm{SAT}$. The reduction works over any choice of the basis B. □

Proof of Lemma 28. Given a bbs ϕ over any Boolean basis B, let Φ be its "arithmetization", 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 (§2.17), it follows that ϕ agrees with Φ on every Boolean input. And because we represent $\mathbb{F}_{2^k}$ as k-bit vectors (§2.16), 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 transformation $\phi\mapsto\Phi$ works over any choice of a basis, provided we have a reasonable representation of formulas, which is the case by §2.11 and Definition 16. □

Proof of Lemma 29. Given an abs Ψ over any arithmetic basis A, we give a reduction that produces a (summation-free) formula Φ over A satisfying

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

for every setting of inputs x of Ψ over $\mathbb{F}_{2^k}$, for every k. Here y will be over $\{0,1\}^m$ for some m that depends on Ψ, and that is bounded by a polynomial on the size of Ψ.

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 $\sum_y \Phi_1(x,y)$, and Ψ_2 to $\sum_z \Phi_2(x,z)$, then the rest is easy: just make sure y and z refer to disjoint sets of variables by renaming as needed, and write $\Psi(x) = \sum_{y,z} \Phi_1(x,y) \cdot \Phi_2(x,z)$.

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

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

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 $\widehat{f}_m$ (§2.17) to rewrite Ψ as

$$\Psi(x) = \sum_{b_1..b_m} \widehat{f}(b_1,..,b_m) \cdot \prod_{i=1..m} (1 + \Psi_i(x) + b_i), \tag{3.9}$$

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 the basis A, provided we have a reasonable representation of formulas, which is the case by §2.11 and Definition 16. □

Proof of Lemma 30. Given a formula $\Phi(x)$ over any arithmetic basis A, and given ℓ, we show a reduction from finding the ℓ^{th} bit of $\sum_x \Phi(x)$, to evaluating the mod-2 sum $\oplus_z \phi(z)$ for some formula ϕ over the Boolean basis corresponding to A.

To begin with, let us assume that A is the standard arithmetic basis so 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 $\mathbb{F}_{2^k}$ (§2.16), 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 can be produced by some function in FP given Φ.

Because the original task is to find the ℓ^{th} bit of the sum $\sum_x \Phi(x)$, 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 $\sum_x \Phi(x)$ 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 A is the standard arithmetic basis.

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 29 just above, of bringing Φ to prenex form — a seemingly useless thing to do as Φ does not involve sums. But notice from (3.9) that as a side effect, the process transforms the summation-free $\Phi(x)$ into the sum $\sum_B \Phi'(x,B)$, 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 $\sum_x \Phi(x)$ — which is the same as the ℓ^{th} bit of $\sum_{x,B} \Phi'(x,B)$ — 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 the basis A, provided we have a reasonable representation of formulas, which is the case by §2.11 and Definition 16. □

Proof of Corollary 31. Immediate by chaining together Lemmas 28, 29, and 30. □

3.1.5 Finishing up — Proofs of Propositions 14-15 and Lemmas 18-21

We finish up Section 3.1 by proving Propositions 14-15 and Lemmas 18-21.

Proof of Proposition 15. Immediate from Lemma 26 and Corollary 31. □

Proof of Lemma 18. Being the affine extension of f, by definition (§2.17), 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 FP given 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 20 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 15, Lemma 27, and Corollary 31, respectively. The composite reduction $\oplus\text{SAT}^{\widetilde{f}} \to \oplus\text{SAT}^f$ works over any choice of a Boolean basis since each constituent reduction does. □

Proof of Proposition 14. Let $\mathcal{O}$ be a language. The evaluation of $\widetilde{\mathcal{O}}$ on a given input x can be expressed as the formula '$\widetilde{\mathcal{O}}(x)$', which is a $\oplus\text{SAT}^{\widetilde{\mathcal{O}}}$ instance (with no free variables). By Lemma 18, there is a reduction from $\oplus\text{SAT}^{\widetilde{\mathcal{O}}}$ to $\oplus\text{SAT}^{\mathcal{O}}$. Putting together, $\widetilde{\mathcal{O}} \to \oplus\text{SAT}^{\mathcal{O}}$. □

Proof of Lemma 19. Immediate from Lemma 28 and Lemma 29. □

Proof of Lemma 20. Lemma 20 is identical to Lemma 30. □

Proof of Lemma 21. Let $\mathcal{O}$ be a language, and let B be the $\mathcal{O}$-extended Boolean basis. Given (Φ, ℓ), consider the task of computing $+\text{ASAT}^{\mathcal{O}}(\Phi, \ell)$, i.e., computing the ℓ^{th} bit of $\Sigma_x \Phi(x)$, where each $x_i \in \{0, 1\}$, and Φ is a formula over the arithmetic basis corresponding to B, that has all its constants in $\mathbb{F}_{2^k}$ for some k.

We show an interactive protocol (§2.4) where the verifier has oracle access capability (§2.8), such that when given access (§2.8) to $\widetilde{\mathcal{O}}$, the protocol performs this task. Further, the protocol we give will show that $+\text{ASAT}^{\mathcal{O}}$ is same-length checkable (§2.4), proving part (i) of the Lemma. Later we will amend the protocol to derive part (ii) as well.

Here is the protocol (to be later amended for part (ii)):

1. The verifier asks from the prover all k bits of the sum $\Sigma_x \Phi(x)$. By responding to this request, the prover implicitly makes the claim '$\Sigma_x \Phi(x) = u$' for some $u \in \mathbb{F}_{2^k}$. If the ℓ^{th} bit of u is 0, then there is nothing to be done; the verifier outputs 0 at this point.
2. The verifier and the prover perform the sumcheck protocol [11, §3.2] over $\mathbb{F}_{2^k}$, and replace the claim with a new one, of the form '$\Phi(y) = v$' for some y, v over the *same* field as that of x, u.
3. The verifier asks from the prover 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 to see if they are consistent and if indeed $\Phi(y) = v$. If all checks pass then the verifier outputs 1.

Notice that all of the responses of an honest prover can be obtained by using $+\text{ASAT}^{\mathcal{O}}$ on formulas of size exactly the size of Φ. (Formulas smaller than Φ can always be appropriately padded; the point is that no larger formula needs to be used.)

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 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 17). This proves part (i) of the lemma.

For part (ii), let L be a language, and let $m \in \mathbb{N}$. Consider extending the basis B from part (i) with $L_{\leq m}$ (§2.2). (So B is now the standard basis extended with $\mathcal{O}$ and $L_{\leq m}$.) Given (Φ, ℓ), consider the same task as in part (i), of computing the ℓ^{th} bit of $\Sigma_x \Phi(x)$ where Φ is over the arithmetic basis corresponding to B.

Modify and amend the protocol from part (i) as follows. For convenience, letting M denote $L_{\leq m}$:

3′. The verifier asks from the prover the value of each gate in the evaluation of $\Phi(y)$, and checks all of them *except* those for $\widehat{M}$. If any of the checks fail, then the verifier outputs 0. If there is no $\widehat{M}$-gate, then there is nothing left to do; the verifier outputs 1.

4. The verifier and the prover perform the interpolation technique from the LFKN protocol [36], to combine multiple claims of the form '$\widehat{M}(z) = w$' into a single one, for each distinct input length $|z|$, as follows.

   For every pair of claims '$\widehat{M}(\vec{\alpha}) = v$' and '$\widehat{M}(\vec{\beta}) = w$' where $\vec{\alpha}, \vec{\beta} \in \mathbb{F}_{2^k}^i$ for some $i \leq m$:

   a. Let $t \in \mathbb{F}_{2^k}^i[x]$ be the line that passes through $\vec{\alpha}$ and $\vec{\beta}$, i.e., let
   $$t(x) := \vec{\alpha} + (\vec{\beta} - \vec{\alpha})x.$$
   Then the pair of claims can be rewritten as '$\widehat{M} \circ t(0) = v$' and '$\widehat{M} \circ t(1) = w$'.

   b. The verifier asks from the prover the univariate polynomial $\widehat{M} \circ t$, and receives some polynomial $\underline{\widehat{M} \circ t}$ as response. The verifier checks that $\underline{\widehat{M} \circ t}(0) = v$ and $\underline{\widehat{M} \circ t}(1) = w$, and outputs 0 if either check fails.

   c. The verifier picks a random $\rho \in \mathbb{F}_{2^k}$ and replaces the pair of claims with the single claim '$\widehat{M}(t(\rho)) = \underline{\widehat{M} \circ t}(\rho)$'.

5. Reverting back to using $L_{\leq m}$ instead of M, at this point the verifier has at most m claims of the form '$\widehat{L_i}^k(\vec{\gamma}) = y$', where $i \leq m$, regarding the value of $\widehat{L}$ over $\mathbb{F}_{2^k}$. These claims can be expressed as the conjunction of at most mk claims regarding the value of $\widetilde{L}$, of the form '$\widetilde{L_i}^k(\vec{\gamma}, j) = y_j$', where $i \leq m$ and $j \leq k$. This conjunction is a formula over the standard Boolean basis extended by $\widetilde{L}_{\leq m}$, hence is a $\oplus\text{SAT}^{\widetilde{L}_{\leq m}}$ instance, of size polynomial in mk. By Lemma 18, it can be transformed to a $\oplus\text{SAT}^{L_{\leq m}}$ instance of size polynomial in mk. The verifier performs this transformation and outputs the result.

Notice that as in part (i), all of the responses of an honest prover can be obtained by using $+\text{ASAT}^{\mathcal{O}}$ on formulas of size exactly the size of Φ. (Formulas smaller than Φ can always be appropriately padded; the point is that no larger formula needs to be used.)

The analysis of the protocol is again standard: if Φ has $\leq n$ nodes, then Step 4 takes $\leq n$ claims of the form '$\widehat{L}_i(z) = w$', where $i \leq m$, and merges them into fewer claims; the probability that there is an error in this merging process, i.e., the probability that all of the merged claims are true, assuming some original claim is false, is at most

$$\#\text{ of merges} \cdot \deg(\widehat{L}_m \circ t) \;/\; \text{size of the field}$$

which again grows slower than $1/n^d$ for any d, because $m \leq n$ and because $k \geq \log^2 n$ by the definition of $+\text{ASAT}$ (Definition 17). This finishes the proof. □

3.2 The IP Theorem

In this section we show that Shamir's IP theorem, PSPACE $\subset$ IP, affinely relativizes. (By §1.1.5, it follows that the IP theorem does not have a *proof* that affinely relativizes.) 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 $\oplus$SAT. We show:

Theorem 33. *Every downward-self-reducible language is in* IP. *This holds relative to every affine oracle.*

Proof. Let $\mathcal{A}$ be an affine oracle. Suppose that the language $L := \{L_n\}_{n\in\mathbb{N}}$ is downward-self-reducible with oracle access to $\mathcal{A}$ (§2.7), i.e., that there is a Cook-reduction (§2.5) $R \in \mathrm{FP}^{\mathcal{A}}$ that yields

$$L_{\leq n} \to L_{\leq n-1}$$

for every $n > 0$. By Proposition 15, there is a reduction $R' \in \mathrm{FP}^{\mathcal{A}}$ that yields

$$\oplus\mathrm{SAT}^{L_{\leq n}} \to \oplus\mathrm{SAT}^{L_{\leq n-1}} \tag{3.10}$$

for every $n \in \mathbb{N}$. Here and throughout the rest of the proof, we take the $\mathcal{A}$-extended Boolean basis for $\oplus$SAT, and use $\oplus\mathrm{SAT}^{L_{\leq -1}}$ to denote $\oplus$SAT.

Repeating (3.10) and combining with Theorem 13, we get a two-step reduction

$$(\oplus\mathrm{SAT}^{L_{\leq n-1}})_{n^d} \to (\oplus\mathrm{SAT}^{L_{\leq n-2}})_{n^{d'}} \to (\oplus\mathrm{SAT}^{L_{\leq n-2}})_{n^d}, \tag{3.11}$$

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 13). We also have the trivial reduction

$$L_{\leq n} \to \oplus\mathrm{SAT}^{L_{\leq n}} \tag{3.12}$$

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 (3.12), and then for n iterations, applies the reduction sequence in (3.11). This composite reduction yields

$$L_{\leq n} \to \oplus\mathrm{SAT}$$

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

By Theorem 12 on checking $\oplus$SAT, it follows that L is computable by an interactive protocol with oracle access to $\mathcal{A}$. □

By its very definition (§2.3), PSPACE has a complete language that is downward-self-reducible — Σ_∞SAT (§2.15), a.k.a. TQBF — and this holds relative to every language. Hence we have:

Corollary 34. $\mathrm{PSPACE} \subset \mathrm{IP}$. *This holds relative to every affine oracle.*

3.2.1 Aside: strong relativization

Corollary 34 allows, for example, to have a different interactive protocol for $\Sigma_\infty\mathrm{SAT}^{\mathcal{A}}$ for each affine oracle $\mathcal{A}$. But if we unwind the proof of that result, we can see that there is essentially one interactive protocol, more precisely, there is one interactive protocol where the verifier has oracle access capability (§2.8) such that when given access (§2.8) to $\mathcal{A}$, the protocol computes $\Sigma_\infty\mathrm{SAT}^{\mathcal{A}}$.

Capturing this sort of phenomenon can be useful. In fact in Section 3.3 below, we give a streamlined proof of $\mathrm{NEXP} \subset \mathrm{MIP}$ that uses exactly this.

So let us make a definition. Recall that we use FP^* to capture the notion of a polynomial-time oracle Turing machine (§1.1.1, §2.8). Here we want to capture a variant notion: we still have a polynomial-time oracle Turing machine, but now the oracle access mechanism is altered so that instead of what is on the oracle tape, say the language $\mathcal{O}$, it gives $\tilde{\mathcal{O}}$, the affine extension of what is on the tape.

Definition 35. ($\mathrm{FP}^{\widetilde{*}}$) Define the class $\mathrm{FP}^{\widetilde{*}}$ — FP with *affine* oracle access capability — as the set of all functions of the form

$$f^{\widetilde{*}} : (\mathcal{O}, x) \mapsto f^*(\widetilde{\mathcal{O}}, x)$$

where $f^* \in \mathrm{FP}^*$.

All definitions built on FP (§2.3-§2.6) naturally generalize to their affine-oracle-access-capable versions: NP to $\mathrm{NP}^{\widetilde{*}}$, IP to $\mathrm{IP}^{\widetilde{*}}$, and so on. Corollary 34 can be strengthened thus:

Theorem 36. $\mathrm{PSPACE}^{\widetilde{*}} \subset \mathrm{IP}^{\widetilde{*}}$.

We do not state Theorem 36 as a corollary because it requires an examination of the proofs leading to it. To rephrase this result in terms of $\Sigma_\infty\mathrm{SAT}$, define the function

$$\Sigma_k\mathrm{SAT}^* : (\mathcal{O}, \varphi) \mapsto \Sigma_k\mathrm{SAT}^{\mathcal{O}}(\varphi)$$

as the natural generalization of $\Sigma_k\mathrm{SAT}$ (§2.15) that takes as input a language $\mathcal{O}$ and a formula φ, and computes $\Sigma_k\mathrm{SAT}(\varphi)$ by interpreting φ over the $\mathcal{O}$-extended basis. Notice that $\Sigma_\infty\mathrm{SAT}^* \in \mathrm{PSPACE}^*$. In fact, for every $L^* \in \mathrm{PSPACE}^*$, there is some $R \in \mathrm{FP}$ such that

$$L^*(\mathcal{O}, x) = \Sigma_\infty\mathrm{SAT}^*(\mathcal{O}, R(x))$$

for every $\mathcal{O}, x$. Thus $\Sigma_\infty\mathrm{SAT}^*$ is PSPACE^*-complete.

Now define

$$\Sigma_k\mathrm{SAT}^{\widetilde{*}} : (\mathcal{O}, \varphi) \mapsto \Sigma_k\mathrm{SAT}^*(\widetilde{\mathcal{O}}, \varphi)$$

as the variant of $\Sigma_k\mathrm{SAT}^*$ that, given the formula φ and the language $\mathcal{O}$, interprets φ over the $\widetilde{\mathcal{O}}$-extended basis, and computes $\Sigma_k\mathrm{SAT}(\varphi)$. Then Theorem 36 says

$$\Sigma_\infty\mathrm{SAT}^{\widetilde{*}} \in \mathrm{IP}^{\widetilde{*}}.$$

3.3 The MIP Theorem

In this section we show that the NEXP $\subset$ MIP theorem of Babai, Fortnow, and Lund [11] affinely relativizes, if it is viewed as a gap amplification result as mentioned in §1.3 (and explained below in §3.3.2).

To begin with, how do we even define MIP? Typically, this would be done via Turing machines equipped with communication tapes. Since our approach (§2.3-2.6) builds exclusively on FP, we must find another, more robust definition for MIP.

Fortunately, such a definition already exists; we recall it in the next section (§3.3.1). After that, we introduce the MIP theorem from the gap amplification perspective (§3.3.2), and then prove it (§3.3.3). Then we prove why the MIP theorem affinely relativizes when viewed as a gap amplification result (§3.3.4), and then contrast the ordinary view of the MIP theorem to the gap amplification view (§3.3.5). We finish by giving some deferred proofs (§3.3.6).

3.3.1 Defining MIP

Recall that IP is defined (§2.4 and the remark on perfect completeness therein) as the set of all languages L such that

$$\begin{aligned} L(x) = 1 &\implies \mathsf{A}y_1\,\mathsf{M}z_1 \cdots \mathsf{A}y_{t(|x|)}\,\mathsf{M}z_{t(|x|)} V(x, y, z) = 1 \\ L(x) = 0 &\implies \mathsf{A}y_1\,\mathsf{M}z_1 \cdots \mathsf{A}y_{t(|x|)}\,\mathsf{M}z_{t(|x|)} V(x, y, z) < 1/3 \end{aligned}$$

for some $V \in \mathrm{P}$ and $t \in \mathrm{poly}(n)$, where $\mathsf{A}x\varphi(x)$ denotes $\mathbf{E}_x[\varphi(x)]$ and $\mathsf{M}x\varphi(x)$ denotes $\max_x \varphi(x)$.

Fortnow, Rompel, and Sipser gave [24] a similar definition for MIP. We paraphrase:

Definition 37 (MIP)**.** MIP is the class of all languages L for which there exists $V^* \in \mathrm{P}^*$, $t \in \mathrm{poly}(n)$, a language π, such that for every string x and language π'

$$L(x) = 1 \implies \mathsf{A}y_1 \, \mathsf{M}z_1 \cdots \mathsf{A}y_{t(|x|)} \, \mathsf{M}z_{t(|x|)} V^{\pi}(x, y, z) = 1$$
$$L(x) = 0 \implies \mathsf{A}y_1 \, \mathsf{M}z_1 \cdots \mathsf{A}y_{t(|x|)} \, \mathsf{M}z_{t(|x|)} V^{\pi'}(x, y, z) < 1/3$$

where y_i, z_i are quantified over $\{0, 1\}$.

In words, L is computable by an interactive protocol where the verifier V^* has oracle access capability (§2.8) such that:

i. when V^* is given access (§2.8) to π, the protocol computes L, or in short, $L \in \gamma\text{-gap-IP}^{\pi}$,
ii. V^* is robust to its oracle, in the sense that even if some other oracle π' is used instead of π, V^* accepts with probability less than $1/3$ whenever $L(x) = 0$.

Intuitively, $L \in \mathrm{MIP}$ iff L has an "interactively checkable proof system": if $L(x) = 1$, then there is an exponentially long proof string π_x that a verifier can check by interacting with a prover. (In the definition above we assemble all such proofs into one language $\pi : (x, i) \mapsto \pi_x(i)$.)

3.3.2 The gap amplification perspective

Consider generalizing IP and MIP as follows.

Definition 38 (γ-gap-IP, γ-gap-MIP)**.** Let $\gamma(n) = 1/\kappa(n)$ for some $\kappa : \mathbb{N} \to \mathbb{N}$. Define:

- γ-gap-IP as in IP (§3.3.1), except by replacing the constant $1/3$ with $1 - \gamma(|x|)$
- γ-gap-MIP as in MIP (§3.3.1), except by replacing the constant $1/3$ with $1 - \gamma(|x|)$.

By definition, $2/3\text{-gap-IP} = \mathrm{IP}$, and $0\text{-gap-IP} = \mathrm{PSPACE}$ (§2.6). So Shamir's IP theorem (Corollary 34) can be restated as $0\text{-gap-IP} \subset 2/3\text{-gap-IP}$.

Similarly, $2/3\text{-gap-MIP} = \mathrm{MIP}$, and:

Proposition 39. $0\text{-gap-MIP} = \mathrm{NEXP}$.

So, similar to Shamir's theorem, the Babai-Fortnow-Lund result $\mathrm{NEXP} \subset \mathrm{MIP}$ can be restated as:

Theorem 40 (MIP theorem)**.** $0\text{-gap-MIP} \subset 2/3\text{-gap-MIP}$.

We defer the proof of Proposition 39 to §3.3.6, and proceed to prove Theorem 40 in the next section.

3.3.3 Proof of the MIP theorem using the IP theorem

The proof becomes a straightforward consequence of Section 3.2, that the IP theorem affinely relativizes, once a key ingredient is introduced. Namely, it is the seminal "multi-linearity test" of Babai, Fortnow, Lund [11, Thm 5.13]. Here we combine it with a "Booleanness test" from the same paper [11, §7.1] and a standard decoding procedure for low-degree polynomials (e.g., [6, §7.2.2]):

Proposition 41. *There is $W^* \in \mathrm{P}^*$, $r \in \mathrm{poly}(n)$, such that the following holds.*

(In words, there is an interactive protocol with round complexity r (§2.4) and with a verifier W^* that has oracle access capability (§2.8) such that the following holds.)

For every language F, and for every $N \in \mathbb{N}$, there is some affine oracle G such that for every $x \in \{0,1\}^{n \leq N}$:

$$\begin{array}{lll} \textit{i.} & \mathsf{A}y_1\,\mathsf{M}z_1 \cdots \mathsf{A}y_t\,\mathsf{M}z_t\ [W^F\left((x,1^N),y,z\right) = F(x)] = 1, & \textit{if } F \textit{ is an affine oracle} \\ \textit{ii.} & \mathsf{A}y_1\,\mathsf{M}z_1 \cdots \mathsf{A}y_t\,\mathsf{M}z_t\ [W^F\left((x,1^N),y,z\right) \notin \{F(x), \textit{`fail'}\}] = 0, & \textit{if } F \textit{ is an affine oracle} \\ \textit{iii.} & \mathsf{A}y_1\,\mathsf{M}z_1 \cdots \mathsf{A}y_t\,\mathsf{M}z_t\ [W^F\left((x,1^N),y,z\right) \notin \{G(x), \textit{`fail'}\}] \leq 1/N, & \textit{otherwise} \end{array}$$

where $t := r(n+N)$.

(In words, on input $(x,1^N)$, when the verifier is given access (§2.8) to F: the protocol computes $F(x)$ if F is an affine oracle, otherwise it computes $G(x)$ with probability $1 - 1/N$.)

We defer the proof to the end of this section (§3.3.6) and proceed to derive Theorem 40.

Proof of Theorem 40. Let $L \in 0\text{-gap-MIP}$. By definition (§3.3.2), there is a language π and $V_0^* \in \mathrm{P}^*$ such that for every language π',

$$\begin{aligned} L(x) = 1 &\implies [V_0^{\pi}(x)] = 1 \\ L(x) = 0 &\implies [V_0^{\pi'}(x)] < 1, \end{aligned} \tag{3.13}$$

where we use $[U(x)]$ as a shorthand for $\mathsf{A}y_1\,\mathsf{M}z_1 \cdots \mathsf{A}y_{t(|x|)}\,\mathsf{M}z_{t(|x|)}U(x,y,z)$ and suppress $t(x)$, the number of rounds taken by the protocol.

(In words, there is a 0-gap interactive protocol where the verifier, say V_0^*, has oracle access capability (§2.8) such that when V_0^* is given access (§2.8) to π, the protocol computes L; in short, $L \in 0\text{-gap-IP}^{\pi}$. Moreover, V_0^* is robust to its oracle in the sense of Definition 37.)

Take any π satisfying (3.13). We may assume that π is an affine oracle since every language Karp-reduces to its affine extension.

By Theorem 36, there is $V_{2/3}^* \in \mathrm{P}^*$ such that

$$\begin{aligned} L(x) = 1 &\implies [V_{2/3}^{\pi}(x)] = 1 \\ L(x) = 0 &\implies [V_{2/3}^{\pi}(x)] < 1/3. \end{aligned} \tag{3.14}$$

(In words, there is a $2/3$-gap interactive protocol where the verifier, say $V_{2/3}^*$, has oracle access capability such that when given access to π, the protocol computes L; in short, $L \in \mathrm{IP}^{\pi}$.)

So all that remains to show, by Definition 37, is that out of all $V_{2/3}^*$ satisfying (3.14), there is a robust one. I.e.,

$$L(x) = 0 \implies [V_{2/3}^{\pi'}(x)] < 1/3 \tag{3.15}$$

for every language π', for some $V_{2/3}^*$ satisfying (3.14).

So take any $V_{2/3}^*$ satisfying (3.14). By amplification, there is $V_{5/6}^* \in \mathrm{P}^*$ such that

$$\begin{aligned} L(x) = 1 &\implies [V_{5/6}^{\pi}(x)] = 1 \\ L(x) = 0 &\implies [V_{5/6}^{\pi}(x)] < 1/6. \end{aligned}$$

Now consider the protocol where the verifier behaves just like $V_{5/6}^*$ except when issuing oracle queries; at those times, instead of querying π directly, say to retrieve $\pi(X)$, it engages the prover to execute the protocol of Proposition 41 on the input $(X,1^m)$, and rejects (outputs 0) immediately if that protocol results in 'fail'. The value of m will be worked out later. Since the verifier of the protocol in Proposition 41 is denoted as W^*, let us use $(V_{5/6} \circ W)^*$ to denote this new verifier.

By Proposition 41-(i), and because π is an affine extension, we have

$$[W^{\pi}(X, 1^m) = \pi(X)] = 1$$

hence

$$L(x) = 1 \implies [(V_{5/6} \circ W)^{\pi}(x)] = 1.$$

Further, by Proposition 41-(ii), and again because π is an affine extension, we have

$$[W^{\pi}(X, 1^m) \neq \pi(X)/\text{`fail'}] = 0$$

hence

$$L(x) = 0 \implies [(V_{5/6} \circ W)^{\pi}(x)] < 1/6.$$

(In words, if we modify $V^*_{5/6}$ so that it uses the protocol of Proposition 41 to perform its oracle queries, then this modification does not change anything when the oracle is π.)

All that is left to show is that

$$L(x) = 0 \implies [(V_{5/6} \circ W)^{\pi'}(x)] < 1/3 \tag{3.16}$$

for every language π'.

So let $L(x) = 0$ and let π' be any language. Depending on whether or not π' is an affine oracle, respectively, let H denote either π', or the language G obtained from Proposition 41 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 $(V_{5/6} \circ W)^*$, we will now argue the validity of four implications

$$L(x) = 0 \implies [V_0^H(x)] < 1 \implies [V_{5/6}^H(x)] < 1/6 \implies [(V_{5/6} \circ W)^H(x)] < 1/6 \implies (3.16)$$

which will prove the theorem.

The first implication is immediate from (3.13).

The second implication follows by Theorem 36 and by H being an affine oracle.

The third implication follows by Proposition 41-(i)-(ii) and by H being an affine oracle.

The last implication is trivial if $H = \pi'$, so suppose $H \neq \pi'$. Then, by Proposition 41-(iii),

$$[W^{\pi'}(X, 1^m) \neq H(X)/\text{`fail'}] < 1/m.$$

Hence, if $q := q(|x|)$ upper bounds the length and the number of oracle queries issued by the verifier $V^*_{5/6}$ during the execution of its protocol on any input $x' \in \{0,1\}^{|x|}$, then

$$[(V_{5/6} \circ W)^{\pi'}(x)] \leq [(V_{5/6} \circ W)^H(x)] + q/m,$$

because except with probability $\leq q/m$, $(V_{5/6} \circ W)^{\pi'}$ either works the same as $(V_{5/6} \circ W)^H$, or rejects (outputs 0) because $W^{\pi'}$ outputs 'fail' at some point. Therefore, putting $m := 6q$ yields (3.16) as desired.

(If $r \in \text{poly}(n)$ denotes the query complexity (§2.10) of $V^*_{5/6}$, and $t \in \text{poly}(n)$ the round complexity (§2.4) of the protocol of $V^*_{5/6}$, then $q := r(|x| + 2t(|x|))$ works.) □

3.3.4 Relativizing the MIP theorem

With the MIP theorem (Theorem 40) just proven, we now turn to showing that it affinely relativizes.

Theorem 42. *Theorem 40 holds relative to every affine oracle.*

It is tempting to try to prove Theorem 42 by merely taking the proof of Theorem 40, and then putting an affine oracle $\mathcal{A}$ in the superscript everywhere we see a complexity class or a function in a complexity class. This does in fact work, but we need to argue that the transformed proof goes through at every step.

In particular, there are two points in the proof of Theorem 40 where Theorem 36 is invoked. If we replace FP with $\mathrm{FP}^{\mathcal{A}}$ throughout the proof, then at those points we would be no longer be invoking Theorem 36, which is a statement about classes built from $\mathrm{FP}^{\widetilde{*}}$. Rather, we would be asserting the truth of a statement about classes built from $(\mathrm{FP}^{\mathcal{A}})^{\widetilde{*}}$, namely

$$(\mathrm{PSPACE}^{\mathcal{A}})^{\widetilde{*}} \subset (\mathrm{IP}^{\mathcal{A}})^{\widetilde{*}}, \tag{3.17}$$

which does not seem a trivial consequence of Theorem 36. The rest of the proof of Theorem 40, however, does clearly go through after the transformation. So all that is left to prove Theorem 42 is to show (3.17).

We now show how (3.17) follows Theorem 36. The lefthand and righthand sides of (3.17), respectively, are the class of 0-gap and $2/3$-gap interactive protocols (Definition 38) where the verifier/verdict predicate (§2.4) is a function in $(\mathrm{FP}^{\mathcal{A}})^{\widetilde{*}}$. To parse what the latter class means, first notice that $(\mathrm{FP}^{p})^{q} = \mathrm{FP}^{p \amalg q}$ for every pair of languages p, q and their disjoint union $p \amalg q$ (§2.18). Second, recall Proposition 11 (§2.18), that $\mathrm{FP}^{\widetilde{p} \amalg \widetilde{q}} = \mathrm{FP}^{\widetilde{p \amalg q}}$. Putting together, it follows that $(\mathrm{FP}^{\mathcal{A}})^{\widetilde{*}}$ comprises functions of the form

$$f^{\widetilde{\mathcal{O} \amalg^*}} : (\mathcal{R}, x) \mapsto f^{\widetilde{\mathcal{O} \amalg \mathcal{R}}}(x) \tag{3.18}$$

where $\mathcal{O}$ is the language that $\mathcal{A}$ is the affine extension of, and the righthand side is the evaluation of the function that Cook-reduces to $\widetilde{\mathcal{O} \amalg \mathcal{R}}$ via f in ℓ rounds (§2.5), for some $f \in \mathrm{FP}$ and $\ell \in \mathrm{poly}(n)$.

So what (3.17) says is, for every function $f^{\widetilde{\mathcal{O} \amalg^*}}$ of the form (3.18), there is another function $g^{\widetilde{\mathcal{O} \amalg^*}}$ of the form (3.18), such that for every language $\mathcal{R}$ and string x,

$$\begin{aligned} [f^{\widetilde{\mathcal{O} \amalg \mathcal{R}}}(x)] = 1 &\implies [g^{\widetilde{\mathcal{O} \amalg \mathcal{R}}}(x)] = 1 \\ [f^{\widetilde{\mathcal{O} \amalg \mathcal{R}}}(x)] < 1 &\implies [g^{\widetilde{\mathcal{O} \amalg \mathcal{R}}}(x)] < 1/3, \end{aligned} \tag{3.19}$$

where we use $[U(x)]$ as a shorthand for $\mathsf{A}y_1\, \mathsf{M}z_1 \cdots \mathsf{A}y_{t(|x|)}\, \mathsf{M}z_{t(|x|)} U(x, y, z)$ for an appropriate $t(x)$ representing the number of rounds taken by the protocol.

So, to show (3.17), take any function $f^{\widetilde{\mathcal{O} \amalg^*}}$ of the form (3.18). This function is based on a function $f \in \mathrm{FP}$ and some round complexity $\ell \in \mathrm{poly}(n)$. Corresponding to f and ℓ, there is also a function $f^{\widetilde{*}} \in \mathrm{FP}^{\widetilde{*}}$. Corresponding to $f^{\widetilde{*}}$, there is, by Theorem 36, some $g^{\widetilde{*}} \in \mathrm{FP}^{\widetilde{*}}$ such that

$$\begin{aligned} [f^{\widetilde{\mathcal{R}}}(x)] = 1 &\implies [g^{\widetilde{\mathcal{R}}}(x)] = 1 \\ [f^{\widetilde{\mathcal{R}}}(x)] < 1 &\implies [g^{\widetilde{\mathcal{R}}}(x)] < 1/3. \end{aligned} \tag{3.20}$$

for every language $\mathcal{R}$ and string x.

Now, if we take $g^{\widetilde{*}}$, and go in the opposite direction of the path we took from $f^{\widetilde{\mathcal{O} \amalg^*}}$ to $f^{\widetilde{*}}$, then we get a function $g^{\widetilde{\mathcal{O} \amalg^*}}$ of the form (3.18). We claim that this function satisfies (3.19). To see this, rewrite (3.18) as

$$f^{\widetilde{\mathcal{O} \amalg^*}} : (\mathcal{R}, x) \mapsto (\mathcal{O} \amalg \mathcal{R}, x) \mapsto f^{\widetilde{\mathcal{O} \amalg \mathcal{R}}}(x)$$

and notice that this is the composition of a simple injection with $f^{\widetilde{*}}$. Similarly

$$g^{\widetilde{\mathcal{O} \amalg^*}} : (\mathcal{R}, x) \mapsto (\mathcal{O} \amalg \mathcal{R}, x) \mapsto g^{\widetilde{\mathcal{O} \amalg \mathcal{R}}}(x)$$

is the composition of the same injection with $g^{\widetilde{*}}$. The claim now follows by (3.20).

3.3.5 Gap amplification versus the standard view

The gap amplification perspective introduced in §3.3.2 broke the MIP theorem into two containments, NEXP $\subset$ 0-gap-MIP and 0-gap-MIP $\subset$ MIP. The second containment was shown in §3.3.4 to hold relative to every affine oracle. We now show that the first one does not.

Proposition 43. NEXP $\subset$ 0-gap-MIP *does* not *hold relative to every affine oracle.*

Proof of Proposition 43. First, let us show an unrestricted language $\mathcal{O}$ (affine or not) such that

$$\text{NEXP}^{\mathcal{O}} \not\subset \text{0-gap-MIP}^{\mathcal{O}}. \tag{3.21}$$

Initialize $\mathcal{O}$ to the all-zeroes language. Let $V_1^*, V_2^*, V_3^*..$ be an enumeration of P* (§2.9). For $(i,j) := (1,1)\ldots(\infty,\infty)$, update $\mathcal{O}$ as follows.

Consider the 0-gap-MIP protocol with verifier $V_i^{\mathcal{O}}$ and round complexity $n^j + j$. Let $L_{i,j}$ be the language computed by this protocol. For all large enough n, say for all $n > n_{i,j}$, on every input $x \in \{0,1\}^n$, the value of $L_{i,j}(x)$ would be unchanged if we modify $\mathcal{O}$ at length 2^n. Hence we may set

$$\mathcal{O}(1^{2^N}) := L_{i,j}(1^N)$$

where $N = 1+\max(n_{i,j}, N')$ and N' is the value of N in the previous iteration (in case there is no previous iteration let N' be -1).

It follows that under this construction for $\mathcal{O}$, the language

$$L(x) := \mathcal{O}(1^{2^{|x|}}),$$

which clearly is in $\text{NEXP}^{\mathcal{O}}$, is not in $\text{0-gap-MIP}^{\mathcal{O}}$, satisfying (3.21) as desired.

Now let $\mathcal{A}$ be the affine extension of $\mathcal{O}$ just constructed. We claim

$$\text{NEXP}^{\mathcal{O}} \not\subset \text{0-gap-MIP}^{\mathcal{A}} \tag{3.22}$$

which, if true, would finish the proof since $\mathcal{O}$ reduces to $\mathcal{A}$ by definition (§2.17), implying $\text{NEXP}^{\mathcal{O}} \subset \text{NEXP}^{\mathcal{A}}$.

By Proposition 14, $\mathcal{A}$ reduces to $\oplus\text{SAT}^{\mathcal{O}}$. Further, being downward self-reducible, $\oplus\text{SAT}^{\mathcal{O}} \in \text{PSPACE}^{\mathcal{O}}$ (see Proposition 50 for a proof), and by Definition 38, $\text{PSPACE}^{\mathcal{O}} \subset \text{0-gap-MIP}^{\mathcal{O}}$. Putting together, we get $\text{0-gap-MIP}^{\mathcal{A}} \subset \text{0-gap-MIP}^{\mathcal{O}}$, proving (3.22) as desired. □

3.3.6 Deferred Proofs

We complete Section 3.3 by proving Propositions 39 and 41.

Proof of Proposition 39. Part (⊂). Let $L \in$ 0-gap-MIP. By Definition 37, there is $V^* \in \text{P}^*$, $t \in \text{poly}(n)$, a language π, such that for every language π'

$$L(x) = 1 \iff \forall y_1 \exists z_1 \cdots \forall y_{t(|x|)} \exists z_{t(|x|)} V^{\pi}(x,y,z) \tag{3.23}$$

$$L(x) = 0 \implies \neg\left(\forall y_1 \exists z_1 \cdots \forall y_{t(|x|)} \exists z_{t(|x|)} V^{\pi'}(x,y,z)\right) \tag{3.24}$$

where y_i, z_i are quantified over $\{0,1\}$.

On input $x \in \{0,1\}^n$, view the righthand side of (3.23) as a tree. The root level of the tree corresponds to the variable y_1, the next level to z_1, then to y_2, z_2, and so on for $2t(n)$ levels until the leaf level, where each leaf node has a binary value $V^{\pi}(x,y,z)$ obtained by using y and z to represent the root-to-leaf path for that node. The value at a non-leaf node is obtained by recursively evaluating the children and then aggregating, via either: (a) the maximum of the values, in case the node corresponds to some z_i variable, or

(b) the minimum of the values, in case it corresponds to some y_i variable. The value of the tree is the value of the root node, which equals 1 if $L(x) = 1$; otherwise, it is 0.

This tree has $S(n)$ nodes, where $S = 2^{2t(n)}$. The value at each leaf node depends on the values of π at inputs of length $p(n + 2t(n))$ or less, where $p \in \mathrm{poly}(n)$ is the query complexity (§2.10) of V^*. Therefore, the entire tree depends only on the first $T(n)$ values of π, where $T = 2^{p(n+2t(n))}$. Combining with (3.23) and (3.24), we get

$$L(x) = 1 \iff \exists\gamma : \forall y_1 \exists z_1 \cdots \forall y_{t(|x|)} \exists z_{t(|x|)} V^{\Gamma}(x, y, z) \tag{3.25}$$

where $\gamma \in \{0,1\}^{T(n)}$ and Γ is the language whose characteristic string is formed by extending γ to an infinite string in some trivial fashion, say with all zeroes.

Notice that the function

$$(\gamma, x, y, z) \mapsto V^{\Gamma}(x, y, z)$$

is in P. (Here γ, x, y, z are, as before, of length $T(n), n, t(n), t(n)$ respectively.)

In fact, even the function

$$(\gamma, x) \mapsto \forall y_1 \exists z_1 \cdots \forall y_{t(n)} \exists z_{t(n)} V^{\Gamma}(x, y, z)$$

is in P because the righthand side corresponds to a tree with $S(n) < T(n)$ nodes as described earlier.

It follows that (3.25) can be rewritten as

$$L(x) = 1 \iff \exists\gamma \in \{0,1\}^{T(|x|)} W(x, \gamma)$$

for some $W \in \mathrm{P}$. This implies that $L \in \mathrm{NEXP}$.

Part (⊃). Let $L \in \mathrm{NEXP}$. By the strong Cook-Levin theorem (§2.20), there is a circuit family $\{C_n(x, y)\}_n$ such that C_n is of size $s(n) \in 2^{\mathrm{poly}(n)}$ and can be described by a function of the form

$$(n, i) \mapsto \text{the type of the } i\text{th gate in } C_n \text{ and the indices of all gates connected to the } i^{\text{th}} \text{ gate} \tag{3.26}$$

that is in FP.

Given $x \in \{0,1\}^n$ 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.

Consider the following zero-gap interactive protocol to compute $L(x)$. The verifier has oracle access capability (§2.8), and when given access (§2.8) to π, behaves as follows:

(i) pick at random $i \in \{0,1\}^{\log s(n)}$,

(ii) using the descriptor function (3.26) for C_n, find out that the i^{th} gate is, say, of type f and is connected, say, to gates $i_1..i_m$ in that order,

(iii) check 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 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).

This protocol is guaranteed to accept if $L(x) = 1$. It will reject with nonzero probability if $L(x) = 0$, because the check at step (iii) has a nonzero chance of being for the output gate of C_n. Moreover, when $L(x) = 0$, even if some other language $\pi' \neq \pi$ is accessed by the verifier, the protocol has a nonzero chance of rejecting, because there is no y for which $C(x, y) = 1$, hence there is no π' that can describe an accepting computation for $C_n(x, \cdot)$.

By Definition 37, it follows that $L \in 0\text{-gap-MIP}$. This finishes the proof. □

Proof of Proposition 41. Description of the protocol. Let L be a language. The affine oracle $\widetilde{L}$, by definition (§2.17), on input $x \in \{0,1\}^n$ gives the z^{th} bit of the value $\widehat{L}$ takes at $y \in \mathbb{F}_{2^k}^m$, i.e.,

$$\widetilde{L}(x) = \left(\widehat{L}_m^{\,k}(y)\right)_z \tag{3.27}$$

where y, z, m, k are all computable in FP 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 FP.

The verifier W^F interprets the input x as in (3.27), and extracts y, z, m, k. Since k denotes the field size, i.e. the logarithm thereof, and since $\mathbb{F}_{2^k}$ can be efficiently identified in $\mathbb{F}_{2^{\geq k}}$, and moreover, since $k \leq n \leq N$, the righthand side of (3.27) can be viewed as

$$\left(\widehat{L}_m^{\,N}(Y)\right)_Z \tag{3.28}$$

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

Owing to this, W^F 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 W^F increase N before overriding (y, z, m, k).

After this initial adjustment phase, W^F proceeds into the main phase where the objective is to check if $F = \widetilde{L}$ for some L. Let $F_{m,N}$ denote the interpretation of F on appropriate input lengths as a function of the form $\widehat{L}_m^{\,N}$. The main phase consists 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 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

$$Q(y) := F_{m,N}(y)(1 + F_{m,N}(y)) \prod_{j=1..i}(1 + \langle y, v_j \rangle) \tag{3.29}$$

  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 [11, §3.2 and §5] for explanations of the terms 'axis parallel line', 'sumcheck protocol', etc.

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

Analysis of the protocol. To begin with, note that W^F never outputs something besides $F(X)$ or 'fail'. Thus, letting $output$ denote the output of W^F,

$$\text{for every prover, } \Pr[output \notin \{F(X), \text{'fail'}\}] = 0. \tag{3.30}$$

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 (3.30), 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., [6, 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 no, then let L_m be arbitrary, say the all-zeroes map $x \in \{0,1\}^m \mapsto 0$. Then set $G := \widetilde{L}$.

Now consider three cases:

(A). $F_{m,N}$ is not nearby any affine function $P : \mathbb{F}_{2^N}^m \to \mathbb{F}_{2^N}$.

In this case, by [11, Thm 5.13 and §7.1],[10]

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

(B). $F_{m,N}$ is nearby $\widehat{L}_m^{\,N}$.

Suppose $F_{m,N}$ disagrees with $\widehat{L}_m^{\,N}$ on y. Then by [6, Proposition 7.2.2.1],[11]

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

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

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

[10] To invoke [11, Theorem 5.13] we set $\epsilon = 1/(900m^4)$, $\delta = 1/(900m^4)$, and use $|\mathbb{F}_{2^N}| \geq 900m^4$. In return we get an axis $i \in \{1..m\}$ along which $\geq \epsilon$-fraction of lines would fail the test in Step 1 with $\geq \delta$-chance, provided that $F_{m,N}$ differs from every affine function on $\geq \epsilon'$-fraction of inputs, where $\epsilon' \leq 1/(100m^2)$.

[11] To invoke [6, 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 $\Pr_y[f(y) \neq P(y)] = \epsilon$, then $\Pr_{h,t_*}[A(y) \neq q(0)$ yet $A(\ell(t_*)) = q(t_*)\,] \leq 2\sqrt{\epsilon} + \frac{d}{|\mathbb{F}|-1}$.

Putting together with (3.30) we get, for every prover,

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

(C). $F_{m,N}$ 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., $P(\mathbb{F}_2^m) \not\subset \mathbb{F}_2$.

Consider Step 2, in particular, the randomly picked point $y' \in \mathbb{F}_{2^N}^m$ on which the expression Q in (3.29) 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 (B), by [6, Proposition 7.2.2.1],

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

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

$$\begin{aligned}
(*) &\le \Pr[\text{penultimate claim in the sumcheck for '}0 = \textstyle\sum_b Q_P(b)\text{' is correct}] \\
&\quad + \Pr[\text{last sumcheck round errs }] \\
&\le \Pr[\text{the first claim '}0 = \textstyle\sum_b Q_P(b)\text{' is correct}] + m\rho
\end{aligned}$$

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 $\le \deg Q$ at a random point in $\mathbb{F}_{2^N}$,

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

Now, applying Rabin's Isolation Lemma [46, Theorem 2.4] to the set $B \subset \mathbb{F}_2^m : P(B) \not\subset \mathbb{F}_2$, we get

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

and putting together we get, for every prover,

$$\begin{aligned}
\Pr[\text{all steps pass}] &= \Pr[\mathcal{E}] \Pr[\text{all steps pass} \mid \mathcal{E}] + (1 - \Pr[\mathcal{E}]) \Pr[\text{all steps pass} \mid \neg\mathcal{E}] \\
&\le \max\left(\Pr[\text{Step 2 passes} \mid \mathcal{E}] \,,\, \Pr[\text{Step 3 passes} \mid \neg\mathcal{E}] \right) \\
&\le \max\left(1 - 1/(4m) + 1/(500m^8) \,,\, 2/(9m) \right)
\end{aligned}$$

$$\leq 1 - 1/(10m). \qquad \text{(c)}$$

This concludes the case analysis. Step 1 through Step 3 are repeated T times, which for our choice of T makes all of (a), (b), and (c) at most $1/N$, thus

$$\Pr[output \notin \{G(x), \text{'fail'}\}] \leq (\max\{\text{(a), (b), (b'), (c)}\})^T \leq 1/N$$

as claimed. □

3.4 Lower Bounds against General Boolean Circuits

Using the IP theorem and its variants, Buhrman, Fortnow, Thierauf [16] and Santhanam [39] succeeded in obtaining the strongest lower bounds known-to-date against general Boolean circuits. In both results, the lower bound is shown for a class of functions computable by Merlin-Arthur protocols; in the case of Buhrman et al. it is for the class MAEXP (§2.6), and in Santhanam it is for prMA, the extension of the class MA (§2.4) to partial languages.

In this section we give a proof that unifies both results. We prove:

Theorem 44. *For every constant d,*

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

(ii) prMA *contains a partial language that does not have circuits of size $O(n^d)$.*

This relative to every affine oracle.

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 45 (Scaling)**.**

(i) If part (i) of Theorem 44 is false, then $\oplus\mathrm{SAT}$ has circuits of size $O(2^{\log^d n})$ for some d.

(ii) If part (ii) of Theorem 44 is false, then $\oplus\mathrm{SAT}$ has circuits of size $O(n^d)$ for some d.

This holds relative to every affine oracle.

We defer the proof of Lemma 45 to the end of this section.

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

Definition 46 ($\Sigma_3\mathrm{SAT}^f(t)$)**.** Let t be a well-behaved resource bound (§2.12), and let f be a language. Define $\Sigma_3\mathrm{SAT}^f(t)$ as the set of all languages L for which there is a Karp reduction to $\Sigma_3\mathrm{SAT}^f$ (§2.15) from the language mapping $(x, 1^{t(|x|)}) \mapsto L(x)$ and $(x, \neq 1^{t(|x|)}) \mapsto 0$.

The second ingredient in proving Theorem 44 is a collapse result: if the conclusion of the Scaling lemma (Lemma 45) holds, then the polynomial-time hierarchy collapses. We defer its proof to the end of this section.

Lemma 47 (Collapse)**.** *Let s be a well-behaved resource bound (§2.12).*

If $\oplus\mathrm{SAT}$ has circuits of size $O(s(n))$, then $\Sigma_3\mathrm{SAT}$ is in $\mathrm{MA}(s(\mathrm{poly}\, n))$.

This holds relative to every affine oracle.

The last ingredient of the proof is a classical result of Kannan [31], showing circuit lower bounds for $\Sigma_3\mathrm{SAT}$, and more generally for $\Sigma_3\mathrm{SAT}(t)$.

Fact 48 (Kannan's bound)**.** *Let s be a well-behaved resource bound (§2.12).*
$\Sigma_3\mathrm{SAT}(\mathrm{poly}\, s(n))$ *contains a language that does not have circuits of size* $O(s(n))$.
This holds relative to every oracle.

With the three ingredients in hand — Scaling and Collapse lemmas, and Kannan's bound — we can prove Theorem 44. Let $\mathcal{A}$ be an affine oracle. For notational convenience, so as to avoid putting $\mathcal{A}$ in the superscripts throughout, extend the standard Boolean basis with $\mathcal{A}$, and let FP stand for $\mathrm{FP}^{\mathcal{A}}$ — hence let P denote $\mathrm{P}^{\mathcal{A}}$, NP denote $\mathrm{NP}^{\mathcal{A}}$, and so on for all classes built on FP (§2.3-§2.6).

For part (i), let $\mathcal{C}$ be the set MAEXP and put $s(n) := 2^{\log^d n}$; for part (ii), let $\mathcal{C}$ be the set prMA 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 46 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 the Scaling and Collapse lemmas.

Proof of Scaling Lemma. Let $\mathcal{A}$ be an affine oracle. For notational convenience, so as to avoid putting $\mathcal{A}$ in the superscripts throughout, extend the standard Boolean basis with $\mathcal{A}$, and let FP stand for $\mathrm{FP}^{\mathcal{A}}$ — hence let P denote $\mathrm{P}^{\mathcal{A}}$, NP denote $\mathrm{NP}^{\mathcal{A}}$, and so on for all classes built on FP (§2.3-§2.6).

There is nothing to prove in part (i), because a $\oplus$SAT instance of size n is computable by brute force in deterministic time $\exp n \cdot \mathrm{poly}\, n$, implying $\oplus\mathrm{SAT} \in \mathrm{EXP} \subset \mathrm{MAEXP}$.

For part (ii), suppose that every partial language in prMA has circuits of size $O(n^d)$ for some fixed d. We want to show that $\oplus$SAT has circuits of size $\mathrm{poly}\, n$. By Theorem 12, $\oplus$SAT reduces to some same-length checkable language K, so it suffices to show this for K instead of $\oplus$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 (§2.12).

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 protocol for computing 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 $\mathrm{poly}\, s(|x|) \subset \mathrm{poly}\, s(|x|)^\epsilon \subset \mathrm{poly}(|x| + |y|)$. So K' is in prMA, 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. □

Proof of Collapse Lemma. Toda famously showed [45] that

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

via a randomized reduction (§2.5). His result holds relative to every oracle. (The same holds in general for $\Sigma_k\mathrm{SAT}$ for all constant k.)

For notational convenience, so as to avoid putting $\mathcal{A}$ in the superscripts throughout, extend the standard Boolean basis with $\mathcal{A}$, and let FP stand for $\mathrm{FP}^{\mathcal{A}}$ — hence let P denote $\mathrm{P}^{\mathcal{A}}$, NP denote $\mathrm{NP}^{\mathcal{A}}$, and so on for all classes built on FP (§2.3-§2.6).

If $\oplus$SAT has circuits of size $O(s(n))$ for formulas of size n, then the protocol for computing Σ_3SAT, on a formula of size n, proceeds by the verifier doing Toda's reduction to obtain a formula of size $m \in \mathrm{poly}\, n$, then the prover sending a circuit for $\oplus$SAT at a large enough input length $\mathrm{poly}\, m$, hence a circuit of size $O(s(\mathrm{poly}\, n))$, and finally, the verifier running the checker for $\oplus\mathrm{SAT}_m$ (Theorem 12) on the circuit, in time $\mathrm{poly}(s(\mathrm{poly}\, n))$, i.e. in time $s(\mathrm{poly}\, n)$ since s is well-behaved. □

3.5 The ZKIP Theorem

AW made the surprising observation that the famous theorem of Goldreich, Micali, and Wigderson, NP $\subset$ ZKIP if one-way-functions exist [27], can be proven via the same techniques underlying the IP theorem [2]. 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 [30]. In this section we adapt this AW-IKK proof to our framework, to show that the ZKIP theorem affinely relativizes.

Theorem 49 (ZKIP theorem)**.** NP $\subset$ ZKIP *if there is a one-way function in* P *secure against* BPP.
This holds relative to every affine oracle.

Similarly to previous work (AW, IKK), we take for granted that under the assumption of Theorem 49, there are bit commitment schemes as in [37], and that this holds relative to every oracle. 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 to 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 r and $\underline{\langle r, u\rangle}$, and verifier checks that $\langle r, u\rangle = \underline{\langle r, u\rangle}$.

Test b. prover decommits $r + v$ 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 since the verifier's view of each decommited bit is uniformly distributed.

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

$$i = f(g_1..g_n), \tag{3.31}$$

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 of 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 (3.31).

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

$$\vec{h}, \underline{\ell((1))}, .., \underline{\ell((n+1))}, \underline{c_1}, .., \underline{c_n}, \underline{\widehat{f} \circ \ell((1))}, .., \underline{\widehat{f} \circ \ell((n+1))} \tag{3.32}$$

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

$$y_1, .., y_s \tag{3.33}$$

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 (3.31) — 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 $\langle r, u\rangle$, where r is 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 (3.33) satisfy either of the following:

(a) There is a fragment of the form (3.31), 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 (3.32) 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

$$\widehat{f}\circ\ell(j) = \widehat{f}(\underline{\ell(j)}) = \underline{\widehat{f}\circ\ell(j)} = \underline{\widehat{f}\circ\ell}(j) \tag{3.34}$$

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 (3.34) 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 $\geq 1/s^2$, where s is 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 for an arbitrary affine oracle $\mathcal{A}$. The idea is that the above protocol, over the standard Boolean basis, generalizes to a protocol over the standard *arithmetic* basis (Definition 16), 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 $\mathbb{F}$, or over $\{0,1\}$ which can be taken as a subset of $\mathbb{F}$ 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 the language that $\mathcal{A}$ is the affine extension of. By definition (§2.17), 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.,

$$\mathcal{A}(x) = \left(\widehat{\mathcal{O}}_m^{\,k}(y)\right)_z \tag{3.35}$$

where y, z, m, k are all computable in FP given x. Conversely, given (y, z, m, k), an input x for which this equality holds is also computable in FP.

Since k denotes the field size, or the logarithm thereof, and since $\mathbb{F}_{2^k}$ can be efficiently identified in $\mathbb{F}_{2^{\geq k}}$, (3.35) can be viewed as

$$\mathcal{A}(x) = \left(\widehat{\mathcal{O}}_m^{\,K}(Y)\right)_Z \tag{3.36}$$

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 function in FP can take each $\mathcal{A}$-gate in C, say

$$\mathcal{A}(g_1..g_n), \tag{3.37}$$

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

$$\left(\widehat{\mathcal{O}}_m^{\,s}(Y(g_1..g_n))\right)_{Z(g_1..g_n)} \tag{3.38}$$

where Y and Z are now overloaded to denote the circuit that parses its input x as (y, z, m, k) of (3.35), and then outputs the values Y and Z of (3.36) respectively, for any $K \geq k$, in particular for $K = s$, the size of C. Notice that writing Z as a subscript in (3.38) is really a shorthand for the idea that a_b is implemented as $\pi(a, b)$ where $\pi(i, j)$ is the circuit that gives the j^{th} bit of i.

So the transformation of C is as follows.

- Perform (3.37) $\mapsto$ (3.38).
- 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 $\{0,1\}$, and the prover is given oracle access (§2.7) to $\mathcal{A}$. (The verifier does not need oracle access to $\mathcal{A}$.) This completes the proof of Theorem 49.

4 Negative Relativization Results

This section shows that several major conjectures in structural complexity do not relativize affinely, mirroring corresponding results of AW. (By Section 1.1, it follows that these results are impossible to settle via affinely relativizing proofs.)

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 constructive; they construct an eligible language relative to which the statement in question is false.

The main novelty in this section, as mentioned in Section 1.3.5, is in the development of the interpolation approach, which is then used to show that $\text{NEXP} \not\subset \text{P/poly}$ is affinely nonrelativizing. 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, reductionist method that is quite convenient to use when the situation allows. To show that ψ does not affinely relativize, we find a statement ψ' for which this is already known, and then show that the implication $\psi \implies \psi'$ affinely relativizes. 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 statements, as it critically relies on the closure of such statements under inference. (However, the results we obtain using the reduction method here can be obtained by AW via direct construction; the point of this method is not to obtain oracles that cannot be found otherwise, but to considerably simplify the job — to recycle oracles, so to speak.) Section 4.3 employs this approach.

4.1 Interpolation Approach

The classical approach to show that $\mathcal{C} \not\subset \mathcal{D}$ does not relativize is to construct a language $\mathcal{O}$ relative to which $\mathcal{C} \subset \mathcal{D}$ holds. The strategy, 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 $\mathcal{C}$ relative to $\mathcal{O}$. This is easy to do sometimes, 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 50 below). 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 1.3.5 for a comparison). Our key result here is Theorem 52, 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:

Proposition 50. $\mathrm{PSPACE} \not\subset \mathrm{P}$ *does not hold relative to every affine oracle.*

Proof. Every downward self-reducible (d-s-r) language is in PSPACE. To see this, view PSPACE as 0-gap-IP (§3.3.2), of languages L computable by an interactive protocol where the error probability can be arbitrarily close to 1 (but never equal to it) when $L(x) = 0$. Now 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.

Therefore, being d-s-r, both $\oplus\mathrm{SAT}$ and $\Sigma_\infty\mathrm{SAT}$ (§2.15) are in PSPACE, as are their negation $\neg\oplus\mathrm{SAT}$ and $\neg\Sigma_\infty\mathrm{SAT}$. Moreover, $\Sigma_\infty\mathrm{SAT}$ is complete for PSPACE by the very definition of PSPACE (§2.3). All these hold relative to every language.

Now put $\mathcal{O} := \Sigma_\infty\mathrm{SAT}$ and let $\mathcal{A}$ be the affine extension of $\mathcal{O}$. By Proposition 14, $\mathcal{A} \to \oplus\mathrm{SAT}^{\mathcal{O}}$. By the fact that $\oplus\mathrm{SAT} \in \mathrm{PSPACE}$ holds relative to every language, $\mathcal{A} \in \mathrm{PSPACE}^{\mathcal{O}}$. By the fact that $\Sigma_\infty\mathrm{SAT}$ is PSPACE-complete, $\mathrm{PSPACE}^{\mathcal{O}} \subset \mathrm{PSPACE}$ hence $\mathcal{A} \in \mathrm{PSPACE}$. Therefore,

$$\mathrm{PSPACE}^{\mathcal{A}} \subset \mathrm{PSPACE} \subset \mathrm{P}^{\mathcal{O}} \subset \mathrm{P}^{\mathcal{A}}$$

where the second containment is because $\Sigma_\infty\mathrm{SAT}$ is PSPACE-complete, and the last containment is because every language reduces to its affine extension. □

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 51 (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. [7, 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 52 (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 $A \subseteq \mathrm{dom}(\widetilde{f}_m^k)$ for some fixed k and m. For the purpose of invoking Lemma 51, 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 51, 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 $A_{m,k} := A \cap \mathrm{dom}(\widetilde{f}_m^k)$, 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 51. □

4.1.1 Application — NEXP vs. P/poly

With the Interpolation theorem (Theorem 52) in hand, we are ready to derive the main result of this section:

Theorem 53. $\mathrm{NEXP} \not\subset \mathrm{P/poly}$ *does not hold relative to every affine oracle.*

Proof. It is a basic fact that NEXP has polynomial-size circuits iff NE (§2.6), the linear-exponential version of NEXP, has circuits of size a ***fixed*** polynomial, and that this holds relative to every language. In notation, for all languages $\mathcal{O}$,

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

Therefore, to prove Theorem 53, it suffices to show a language $\mathcal{O}$ satisfying

$$\mathrm{NE}^{\widetilde{\mathcal{O}}} \subset \mathrm{SIZE}^{\mathcal{O}}(n^d), \tag{4.1}$$

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

Take an enumeration $N_0^*, N_1^*,..$ of the class NE^*. Such an enumeration can be obtained from one for NP^* (§2.9), since by definition (§2.8, §2.6),

$$\mathrm{NE}^* = \{L^* : (\exists c \in \mathbb{N}, K^* \in \mathrm{NP}^*)(\forall x, \mathcal{O})\ L^*(\mathcal{O}, x) = K^*(\mathcal{O}, x, 1^{2^{c|x|}})\}.$$

We will want to talk about the query complexity of each N_i^* in the enumeration. Underlying each N_i^* is a constant $c \in \mathbb{N}$ and a function $K^* \in \mathrm{NP}^*$. Underlying K^* is some $\ell \in \mathrm{poly}(n)$, and some $f^* \in \mathrm{P}^*$ of query complexity (§2.10) q_f, say. Define the query complexity q_K of K^* as $q_f(n + \ell(n))$. Define the query complexity q_i of N_i as $q_K(n + 2^{cn})$.

The point of query complexity here is this: if $N_i^*(\mathcal{O}, x) = 1$, then this equality can be maintained by fixing only $q_i(|x|)$ bits of $\mathcal{O}$ and changing the rest arbitrarily.

Now, modify the list $N_0, N_1,..$ into a list $M_0, M_1,..$ (repetitions allowed) such that if M_i has query complexity q_i, then $q_i(n) \leq 2^{n \log n}$ for all $n > i$.

Initialize $\mathcal{O}$ to the all-zeroes language. The plan is to modify $\mathcal{O}$ in such a way that for every $n > 1$, a size-n^d circuit with access to $\mathcal{O}$, say $C_n^{\mathcal{O}}$, can compute the function

$$\begin{aligned} L_n &: \{0,1\}^{\lfloor \log n \rfloor} \times \{0,1\}^n \to \{0,1\} \\ L_n &: (i, x) \mapsto M_i^{\widetilde{\mathcal{O}}}(x). \end{aligned} \tag{4.2}$$

This yields (4.1), hence the theorem, because each language $K \in \mathrm{NE}^{\widetilde{\mathcal{O}}}$ corresponds to some $M_i^{\widetilde{\mathcal{O}}}$ (§2.8), 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 $C_{|x|}^{\mathcal{O}}$, implying $K \in \mathrm{SIZE}^{\mathcal{O}}(n^d)$.

We modify $\mathcal{O}$ iteratively; in iteration $n > 1$ we finalize $\mathcal{O}$ on all inputs in $\{0,1\}^{\leq n^d}$, plus some additional $2^{4n \log n}$ inputs at most. Let f_n denote the finalized portion of $\mathcal{O}$ at the end of iteration n, i.e., f_n is the restriction of $\mathcal{O}$ to those inputs on which it is finalized by the end of iteration n.

In iteration 1 we do nothing, so $f_1 : \{\lambda, 0, 1\} \to \{0\}$ where λ is the empty string. At iteration $n > 1$, consider all possible ways of extending f_{n-1} to a language f. Out of all such f, pick one such that when $\mathcal{O} = f$, the collection

$$\mathcal{S}_f := \{(i, x) : L_n(i, x) = 1\} \tag{4.3}$$

is maximal. Set $\mathcal{O} = f$.

Now we want to "open up space" in f by un-defining it at some inputs, the idea being then to encode the function L_n in the freed space so that a small circuit can look it up. In doing so, of course, we do not want to disturb (4.3), 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 a pair (i, x) in $\mathcal{S}_f$. Because M_i has query complexity less than $2^{n \log n}$ on input $x \in \{0,1\}^n$, the membership of (i, x) in $\mathcal{S}_f$ can be preserved by fixing $\widetilde{f}$ on at most $2^{n \log n}$ inputs only. 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 $\operatorname{dom} f_{n-1}$, such that on B we have no control as to how f behaves, but on the "good" inputs $\{0,1\}^* \setminus (B \cup \operatorname{dom} f_{n-1})$, we can change f arbitrarily. So let f_n be the restriction of f to $B \cup \operatorname{dom} f_{n-1}$.

Now that we opened up space in f, we are ready to store the information in (4.2) 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 $\operatorname{dom} f_n$ 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 $\operatorname{dom} f_{n-1}$ 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. □

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, $\mathcal{C} \subset \mathcal{D}$ does not algebrize.

In this section we develop this approach of AW for our framework. The key observation that enables this is Proposition 11 (§2.18), 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 Section 4.2.1.

Remark. In order to avoid lengthy technicalities, in the rest of this section we will embrace the Turing machine based jargon — running time, algorithm, etc. □

Definition 54 (P_{cc} vs. $\mathrm{P}_{\mathrm{ticc}}$)**.** Define $\mathrm{P}_{\mathrm{ticc}}$ 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 P_{cc} denote the relaxation of $\mathrm{P}_{\mathrm{ticc}}$ 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 $\mathrm{P}_{\mathrm{ticc}}$ to define $\mathrm{NP}_{\mathrm{ticc}}, \mathrm{BPP}_{\mathrm{ticc}}$, etc., similar to how we define $\mathrm{NP}, \mathrm{BPP}$, etc., from P.[12] Similarly for $\mathrm{NP}_{\mathrm{cc}}, \mathrm{BPP}_{\mathrm{cc}}$, etc., versus 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 [12] is defined on strings of every even length, while Definition 54 requires length a power of two; our convention causes nothing but convenience in this section.

[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 $|X|$.

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

Theorem 55. *If $\mathcal{C}_{\mathrm{ticc}} \not\subset \mathcal{D}_{\mathrm{cc}}$, then $\mathcal{C} \subset \mathcal{D}$ does not hold relative to every affine oracle. Here $\mathcal{C}, \mathcal{D}$ can be any class in the polynomial-time hierarchy (§2.3) containing* P.

Proof. Supposing there is some $f := \{f_n\}$ in $\mathcal{C}_{\mathrm{ticc}} \setminus \mathcal{D}_{\mathrm{cc}}$, we want to show an affine oracle $\mathcal{A}$ relative to which $\mathcal{C} \not\subset \mathcal{D}$. For concreteness, the reader may take $\mathcal{C}$ to be NP, say, and $\mathcal{D}$ to be BPP.

By Proposition 11, instead of an affine oracle, it suffices for $\mathcal{A}$ to be the disjoint union of two affine oracles $\mathcal{A}_0 := \widetilde{\mathcal{O}_0}$ and $\mathcal{A}_1 := \widetilde{\mathcal{O}_1}$. In fact, since every language reduces to its affine extension, it suffices to show $\mathcal{O}_0, \mathcal{O}_1$ such that

$$\mathcal{C}^{\mathcal{O}_0 \amalg \mathcal{O}_1} \not\subset \mathcal{D}^{\widetilde{\mathcal{O}_0} \amalg \widetilde{\mathcal{O}_1}}.$$

For every $n \in \mathbb{N}$, pick an arbitrary pair $(X_n, Y_n) \in \mathrm{dom}\, f_n \subset \{0,1\}^{2^n} \times \{0,1\}^{2^n}$. Initialize $\mathcal{O}_0$ to have the same truth table as X_n for every n, and similarly for $\mathcal{O}_1$ versus Y_n. Because $f \in \mathcal{C}_{\mathrm{ticc}}$, the language $L := \{L_n\}$ defined as

$$L(1^n) := f(\mathcal{O}_{0,n}, \mathcal{O}_{1,n}), \quad L(\neq 1^n) := 0$$

is in $\mathcal{C}^{\mathcal{O}_0 \amalg \mathcal{O}_1}$; to see this just consider using $\mathcal{O}_0 \coprod \mathcal{O}_1$ to simulate a $\mathcal{C}_{\mathrm{ticc}}$-protocol for f. Our objective is to modify $\mathcal{O}_0$,$\mathcal{O}_1$ so that L remains in $\mathcal{C}^{\mathcal{O}_0 \amalg \mathcal{O}_1}$ and becomes out of $\mathcal{D}^{\widetilde{\mathcal{O}_0} \amalg \widetilde{\mathcal{O}_1}}$.

To that end, for any pair of strings $(X, Y) \in \{0,1\}^{2^n} \times \{0,1\}^{2^n}$, let $\mathcal{O}_0 \leftarrow X$ denote the result of updating $\mathcal{O}_0$ so that at length-n inputs, it has the same truth table as X; similarly use $\mathcal{O}_1 \leftarrow Y$ to denote the result of updating $\mathcal{O}_1$ with Y.

Now let $N_1, N_2, \ldots$ be an enumeration of $\mathcal{D}$-algorithms endowed with an oracle access mechanism. (To be precise, we need to consider the class $\mathcal{D}^*$, defined using the class FP* (§2.8) in the same way that $\mathcal{D}$ would be defined from FP. As stated earlier, however, for convenience in this section we embrace the Turing Machine based jargon.) For each algorithm in the enumeration, say for N_i, define $g^i := \{g^i_n\}$ as

$$g^i_n(X, Y) := N_i^{\widetilde{(\mathcal{O}_0 \leftarrow X)} \amalg \widetilde{(\mathcal{O}_1 \leftarrow Y)}}(1^n) \tag{4.4}$$

where (X, Y) ranges over $\mathrm{dom}\, f_n$. In case N_i's output is not well-defined on 1^n — due to N_i computing a partial language which 1^n is outside the domain of — just let g^i_n take the value '$\perp$'.

We claim that g^i differs from f on infinitely many inputs. Indeed, the right-hand-side of (4.4) can be computed by a protocol where one party is given access to X and knows $\mathcal{O}_0$ (up to a finite length, beyond which N_i is guaranteed not to access when run on 1^n), the other party is given Y and knows $\mathcal{O}_1$ (again finitely bounded), and the two parties simulate N_i by using each other as an oracle for the missing side of the disjoint union. So $g^i \in \mathcal{D}_{\mathrm{cc}}$. Since $f \notin \mathcal{D}_{\mathrm{cc}}$, the claim follows.

Now, for $i = 1..\infty$, find a pair (X_{n_i}, Y_{n_i}) in $\mathrm{dom}\, f_{n_i} = \mathrm{dom}\, g^i_{n_i}$ on which f and g^i differ, for some n_i arbitrarily large. Update $\mathcal{O}_0$ to $\mathcal{O}_0 \leftarrow X_{n_i}$ and $\mathcal{O}_1$ to $\mathcal{O}_1 \leftarrow Y_{n_i}$, so that $L(1^{n_i})$ differs from $N_i^{\widetilde{(\mathcal{O}_0 \leftarrow X)} \amalg \widetilde{(\mathcal{O}_1 \leftarrow Y)}}(1^{n_i})$. Since n_i is arbitrarily large, this update does not disturb the previous iterations — e.g., $n_i > 2^{2^{n_i - 1}}$ suffices since $\mathcal{D} \subset$ EXP. □

4.2.1 Applications

Theorem 55 allows us to replicate two negative algebrization results of AW:

Corollary 56. *Neither of the following statements hold relative to every affine oracle: (i)* $\mathrm{coNP} \subset \mathrm{MA}$, *(ii)* $\mathrm{P}^{\mathrm{NP}} \subset \mathrm{PP}$.

Proof. Let $\mathrm{Disj}(X,Y) := \forall i\, \neg(X(i) \wedge Y(i))$ be the disjointness predicate. It is clear that $\mathrm{Disj} \in \mathrm{coNP}_{\mathrm{ticc}}$. On the other hand, it takes at least $\Omega(2^{n/2})$ bits of communication to compute Disj by a Merlin-Arthur protocol [32, Corollary 1], implying $\mathrm{Disj} \notin \mathrm{MA}_{\mathrm{cc}}$. Part (i) now follows from Theorem 55.

For part (ii), let $\mathrm{RCYB}(X,Y) := \arg\max_{i \in \{0,1\}^n}(X(i) \wedge Y(i))$, and let $f(X,Y) := \mathrm{RCYB}(X,Y) \mod 2$ be the predicate deciding whether the Rightmost-Common-Yes-Bit position of X and Y is odd (in the uninteresting case that there is no Common-Yes-Bit position, f outputs 0). It is easy to see that $f \in (\mathrm{P}^{\mathrm{NP}})_{\mathrm{ticc}}$. On the other hand, it takes at least $\Omega(2^{n/3})$ bits of communication to compute f by a probabilistic protocol [17, Section 3.2],[13] implying $f \notin \mathrm{PP}_{\mathrm{cc}}$. Part (ii) now follows from Theorem 55. □

We can use Theorem 55 to replicate a result of IKK as well:

Theorem 57. $\mathrm{RP} \subset \mathrm{SUBEXP}$ *does not hold relative to every affine oracle. Here* SUBEXP *denotes* $\cap_{d \in \mathbb{N}} \mathrm{DTIME}(\{2^{cn^{1/D}}\}_{c \in \mathbb{N}, D > d})$.[14]

Proof of Theorem 57. Consider the set of all pairs of strings (X,Y) such that X, Y respectively equal the truth table (suitably encoded) of $\widetilde{f}_m^{\,k}, \widetilde{g}_m^{\,k}$ for some m and some $f, g : \{0,1\}^m \to \{0,1\}$, where k is a large enough constant, say $k = 10$. Consider restricting the equality predicate $\mathrm{Equal}(X,Y) := \forall i (X(i) \equiv Y(i))$ to this set, and call the resulting function $F(X,Y)$.

Yao's classical result, that Equal requires $\Omega(2^n)$ bits of communication, implies that there is no $G \in \mathrm{SUBEXP}_{\mathrm{cc}}$ that F can be extended to. In short, and with a slight abuse of notation, $F \notin \mathrm{SUBEXP}_{\mathrm{cc}}$. To see this, suppose the contrary. Then $\mathrm{Equal}_n(X,Y)$ can be computed by the following protocol: given $X \in \{0,1\}^{2^n}$ Alice computes $\widetilde{X}^{\,k}$, and given Y Bob computes $\widetilde{Y}^{\,k}$; then Alice and Bob compute $F(\widetilde{X}^{\,k}, \widetilde{Y}^{\,k})$, hence $\mathrm{Equal}(X,Y)$, using only $\mathrm{subexp}(|\widetilde{X}^{\,k}|)$ bits of communication. This contradicts Yao's result since $\widetilde{X}^{\,k}$ is a function on $O(nk)$ bits when suitably encoded, and since $k \in O(1)$.

On the other hand, F can be computed by the following randomized protocol: On input X the truth table of some $\widetilde{f}_m^{\,k}$, Alice picks a random $\alpha \in \mathrm{dom}(\widetilde{f}_m^{\,k})$ and sends to Bob the pair $(\alpha, \widetilde{f}_m^{\,k}(\alpha))$. On input Y the truth table of some $\widetilde{g}_m^{\,k}$, Bob receives Alice's message and accepts if $\widetilde{f}_m^{\,k}(\alpha) = \widetilde{g}_m^{\,k}(\alpha)$ and rejects otherwise. To bound the error probability of this protocol, suppose $X \neq Y$, as there is no chance of error otherwise. The random choice of Alice, α, corresponds to a point a in the space $\mathbb{F}_{2^k}^m$; let ℓ be any line in this space that passes through a, and that is parallel to any one of the m axes. Since X,Y are promised to be of the form $\widetilde{f}_m^{\,k}, \widetilde{g}_m^{\,k}$, if they are not equal, then they agree on only one of the 2^k points on ℓ, making the error probability less than $1/1000$ by our choice of k.

So if we take the class $\mathrm{RP}_{\mathrm{ticc}}$, and relax its definition to include families $\{f_n\}$ where $\mathrm{dom}\, f_n$ 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 $\mathrm{prRP}_{\mathrm{ticc}}$, and then the previous paragraph shows that $F \in \mathrm{prRP}_{\mathrm{ticc}}$. (We can ensure that $\mathrm{dom}\, F_n$ is never empty for any n by simple padding.)

Now the proof of Theorem 55 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 through the mechanisms given in Chapter 2 (§2.3-§2.6) and contained in EXP. The claim follows. □

[13] The authors of [17] 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] In fact, Theorem 57 holds for $\mathrm{RP} \subset \mathrm{SUBLINEXP}$, where SUBLINEXP denotes $\cap_{d \in \mathbb{N}} \mathrm{DTIME}(\{c2^{n/D}\}_{c \in \mathbb{N}, D > d})$. We state it in the weaker form because we'd need to slightly refine our approach in §2.6 to be able to officially define SUBLINEXP. (Notice that the class $\mathcal{C} = \{c2^{n/D}\}_{c \in \mathbb{N}, D > d}$ is not closed under taking polynomials in the sense of §2.6, but that it is closed under taking quasi-linear functions: if $t \in \mathcal{C}$, then for every $d \in \mathbb{N}$, there is some $t' \in \mathcal{C}$ such that $t(n) \log^d t(n) < t'(n)$ for every n.)

4.3 Reduction Approach

As mentioned in the beginning of Section 4, sometimes we can get away without constructing oracles, and still show that ψ does not affinely relativize. To do so, we find some ψ' which we already know has that status, and then show that the implication $\psi \implies \psi'$ affinely relativizes. We thus reduce the task of creating an oracle relative to which ψ is false, to doing the same for ψ', with the implication $\psi \implies \psi'$ serving as the reduction.

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

Theorem 58. *None of the following statements hold relative to every affine oracle:*
(i) $\mathrm{NP} \subset \mathrm{P}$*, (ii)* $\mathrm{NP} \not\subset \mathrm{P}$*, and (iii)* $\mathrm{NP} \subset \mathrm{BPP}$.

Proof. Part (i): Theorem 53 showed that $\mathrm{NEXP} \not\subset \mathrm{P/poly}$ does not hold relative to every affine oracle, and Theorem 44 showed that $\mathrm{MAEXP} \not\subset \mathrm{P/poly}$ does. The claim follows because

$$\mathrm{NP} \subset \mathrm{P} \implies \mathrm{MA} \subset \mathrm{P} \implies \mathrm{MAEXP} \subset \mathrm{NEXP},$$

where both implications hold relative to every language, hence relative to every affine oracle.

Part (ii): Proposition 50 showed that $\mathrm{PSPACE} \not\subset \mathrm{P}$ does not hold relative to every affine oracle. The claim follows since $\mathrm{NP} \subset \mathrm{PSPACE}$ holds relative to every language, hence relative to every affine oracle.

Part (iii): Corollary 56 states that $\mathrm{coNP} \subset \mathrm{MA}$ does not have hold relative to every affine oracle. The claim follows since $\mathrm{NP} \subset \mathrm{BPP} \implies \mathrm{coNP} \subset \mathrm{MA}$ holds relative to every language. □

5 Suggestions for Further Research

We finish by listing some suggestions for further research.

The PCP **theorem.** Section 1.3.7 explained that both $\mathrm{PSPACE} \subset \mathrm{IP}$ and $\mathrm{NEXP} \subset \mathrm{MIP}$ can be naturally viewed as gap-amplification results, and from that point of view both theorems affinely relativize. 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 genuine independence result. As pointed out in Section 1.2.1, a common feature — or flaw, if the reader is logically inclined — of both our framework and related works, AIV and IKK, is that the relativization barriers are formalized through axioms that 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 does not relativize", when formalized in these frameworks, on the surface look like they give some sort of logical independence result — but they do not. (See Section 1.2.1.)

For an independence result, one must formalize the relativization barriers with a *subset* of axioms governing everyday mathematics, the idea being to find the "weakest" version of everyday math that can derive each relativizing statement, and then to show that no nonrelativizing statement can be derived by that much of mathematics.

A quantitative theory of relativization. Both relativization and affine/algebraic relativization are rigid notions, in the sense that something either relativizes or does not. This 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.

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. P=?NP. *Electronic Colloquium on Computational Complexity (ECCC)*, 24:4, 2017.

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

[3] Scott Aaronson (https://cstheory.stackexchange.com/users/1575/scott-aaronson). Can relativization results be used to prove sentences formally independent? Theoretical Computer Science Stack Exchange. URL: https://cstheory.stackexchange.com/q/3207 (version: 2010-11-20).

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

[5] 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.

[6] 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.

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

[8] Barış Aydınlıoğlu. *A Study of the* NEXP *vs.* P/poly *Problem and its Variants*. PhD thesis, University of Wisconsin, Madison, 2017.

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

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

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

[12] 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.

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

[14] Richard Beigel, Harry Buhrman, and Lance Fortnow. NP might not be as easy as detecting unique solutions. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 203–208, 1998.

[15] Stephen Bellantoni and Stephen A. Cook. A new recursion-theoretic characterization of the polytime functions. *Computational Complexity*, 2:97–110, 1992.

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

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

[18] Timothy Chow (http://mathoverflow.net/users/3106/timothy-chow). Definition of relativization of complexity class. MathOverflow. URL: http://mathoverflow.net/q/76021 (version: 2011-09-21).

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

[20] Niel de Beaudrap (http://cstheory.stackexchange.com/users/248/niel-de-beaudrap). A good reference for complexity class operators? Theoretical Computer Science Stack Exchange. URL: http://cstheory.stackexchange.com/q/21602 (version: 2014-03-18).

[21] Heinz-Dieter Ebbinghaus, Jörg Flum, and Wolfgang Thomas. *Mathematical logic (2. ed.)*. Undergraduate texts in mathematics. Springer, 1994.

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

[23] 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.

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

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

[26] Martin Furer, Oded Goldreich, Yishay Mansour, Michael Sipser, and Statis Zachos. On completeness and soundness in interactive proof systems. *Advances in Computing Research: A Research Annual, vol. 5 (Randomness and Computation, S. Micali, ed.)*, 1989.

[27] 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.

[28] Hans Heller. On relativized exponential and probabilistic complexity classes. *Information and Control*, 71(3):231–243, 1986.

[29] Wilfrid Hodges. Tarski's truth definitions. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, fall 2014 edition, 2014.

[30] 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.

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

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

[33] Robin Kothari (http://cstheory.stackexchange.com/users/206/robin-kothari). Is relativization well-defined? Theoretical Computer Science Stack Exchange. URL: http://cstheory.stackexchange.com/q/21606 (version: 2014-03-18).

[34] Dexter Kozen. *Theory of Computation*. Texts in Computer Science. Springer, 2006.

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

[36] 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.

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

[38] Sasho Nikolov (https://cstheory.stackexchange.com/users/4896/sasho-nikolov). What are natural examples of non-relativizable proofs? Theoretical Computer Science Stack Exchange. URL: https://cstheory.stackexchange.com/q/20515 (version: 2014-01-27).

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

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

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

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

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

[44] 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.

[45] 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.

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

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il