Algebraic Proof Complexity: Progress, Frontiers and Challenges

Toniann Pitassi [*] Iddo Tzameret [†]

July 1, 2016

Abstract

We survey recent progress in the proof complexity of strong proof systems and its connection to algebraic circuit complexity, showing how the synergy between the two gives rise to new approaches to fundamental open questions, solutions to old problems, and new directions of research. In particular, we focus on tight connections between proof complexity lower bounds (namely, lower bounds on the size of proofs of certain tautologies), algebraic circuit lower bounds, and the Polynomial Identity Testing problem from derandomization theory.

Contents

[*]Email: toni@cs.toronto.edu. Department of Computer Science, University of Toronto, Toronto, Canada.
[†]Email: iddo.tzameret@rhul.ac.uk. Department of Computer Science, Royal Holloway, University of London, Egham, UK.

1 Introduction

Propositional proof complexity aims to understand and analyze the computational resources required to prove propositional tautologies, in the same way that circuit complexity studies the resources required to compute boolean functions. A central question in the area asks whether every boolean tautology has a short propositional proof. Here, a propositional proof system can take many forms. One such proof system is the resolution refutation system whose proof-search algorithm constitutes the basis of current state of the art industrial-level SAT solvers (this thread of research was recently covered in SigLog; see Nordström [Nordström, 2015]). For resolution and its *weak* extensions, strong lower bounds are known since Haken [Haken, 1985]. But the major open questions in proof complexity, those originating from boolean circuit complexity and complexity class separations, such as P vs. NP, are about the length of *much stronger* proof systems than resolution, and these stronger systems will be the focus of this survey.

The prototypical strong proof system is the standard Hilbert-style propositional proof system, called *Frege proof system*, in which a proof starts from a fixed finite set of axioms and derives new propositional *formulas* using a fixed set of sound derivation rules. Establishing any super-polynomial size lower bound on such proofs (in terms of the size of the formula proved) is a major open problem in proof complexity, and a fundamental question in complexity theory.

The seminal work of Cook and Reckhow [Cook and Reckhow, 1979] showed that in its strongest form, proof-size lower bound questions relate directly to fundamental hardness questions in computational complexity: establishing super-polynomial lower bounds for *every* propositional proof system would separate NP from coNP (and thus also P from NP).

The aim of this survey is to outline a new research direction, connecting algebraic circuit complexity to proof complexity. The prominent goal of this approach is the quest for lower bounds on strong proof systems; other important aspects are connections to derandomization theory and application to feasible mathematics. The survey is meant to give some basic background in propositional proof complexity and describe the algebraic approach to proof complexity.

In what follows, Section 2 gives the basic definitions of algebraic circuits, propositional proof systems and algebraic proof systems, and a quick survey of the background results. Section 3 is devoted to the Ideal Proof System (IPS). In this section we show that IPS is closely connected to the Extended Frege proof system, and show that superpolynomial lower bounds for IPS proofs imply algebraic circuit lower bounds. Section 4 is devoted to the study of the non-commutative IPS. In this section we show that non-commutative IPS is equivalent (up to quasi-polynomial factors) to the Frege system and discuss the ramifications of this equivalence. Section 5 is dedicated to lower bounds for restricted subsystems of IPS. Section 6 discusses connections between algebraic proof complexity and the polynomial identity testing (PIT) problem and the use of structural results on algebraic circuits in proof complexity, as well as application to feasible mathematics. Finally we conclude with a discussion and open problems in Section 7.

2 Basic Concepts

For a natural number we let $[n] = \{1, \ldots, n\}$. Let $\mathbb{F}$ be a field. Denote by $\mathbb{F}[x_1, \ldots, x_n]$ the ring of (commutative) polynomials with coefficients from $\mathbb{F}$ and variables $x_1, \ldots, x_n$. In this survey,

unless otherwise stated, we treat polynomials as *formal* linear combination of monomials, where a monomial is a product of variables. Hence, when we talk about the *zero polynomial* we mean the polynomial in which the coefficients of all monomials are zero (it can happen that over, say, $GF(2)$, $x^2 + x$ computes the zero *function*, but it is *not* the zero polynomial, because it has two nonzero monomial coefficients). Similarly, two polynomials are said to be *identical* if they have precisely the same monomial coefficients. The *degree* of a polynomial (or total degree) is the maximal sum of variable powers in a monomial with a nonzero coefficient in the polynomial. If the power of each variable in every monomial is at most 1 we say that the polynomial is *multilinear*. We write $\mathsf{poly}(n)$ to denote a polynomial growth in n, namely a function that is upper bounded by $n^{O(1)}$, and $\mathsf{qpoly}(n)$ to denote a quasi-polynomial growth in n, that is, $n^{\log^{O(1)} n}$.

2.1 Propositional Proof Systems

Cook and Reckhow [Cook and Reckhow, 1979] defined a general concept of a propositional proof system from the perspective of computational complexity theory: a propositional proof system is a polynomial-time function f from a set of finite strings over some given alphabet *onto* the set of propositional tautologies (reasonably encoded). Thus, $f(x) = y$ means that the string x is a proof of the tautology y. Note that since f is onto, all tautologies and only tautologies have proofs (and thus the proof system is complete and sound).

The idea behind the Cook-Reckhow definition is that a purported proof x may be much longer than the tautology y it proves, but given a proof it should be possible to efficiently check (efficient with respect to the *proof length*) that it is indeed a correct proof of the tautology. We say that a propositional proof system is *polynomially bounded* if there exists a polynomial p that bounds the minimal proof size $|x|$ for every tautology y; namely, for every tautology y its minimal proof x is such that $|x| \leq \mathsf{poly}(|y|)$. Under the general Cook-Reckhow definition we have:

Theorem 2.1 (Cook-Reckhow [Cook and Reckhow, 1979])**.** $\mathsf{NP}{=}\mathsf{coNP}$ *if and only if there is a polynomially bounded propositional proof system.*

Therefore, proving lower bounds against stronger and stronger propositional proof systems is clearly a formidable problem, as it can be considered as partial progress towards proving $\mathsf{NP}{\neq}\mathsf{coNP}$ (and thus $\mathsf{P}{\neq}\mathsf{NP}$).

The definition of a propositional Cook-Reckhow proof system encompasses most standard proof systems for propositional tautologies, such as resolution and usual textbook proof system for propositional logic. In this survey we discuss specific propositional proof systems, that are at least as strong as the Frege or the Extended Frege system (see below). Though proving lower bounds on (Extended) Frege proof sizes for some families of tautologies would not amount to $\mathsf{NP}{\neq}\mathsf{coNP}$, it would still constitute a breakthrough in complexity theory.

2.1.1 Frege Proof Systems

One of the most investigated and central propositional proof systems comes from the tradition of logic and is called the *Frege proof system*. A Frege proof system is any system that has a fixed number of axiom schemes and sound derivation rules, that is also implicationally complete[1], and in which proof lines are written as propositional *formulas*. It is known since Reckhow's work [Reckhow, 1976] that all Frege proof systems are polynomially equivalent to each other, and hence it does not matter precisely which rules, axioms, and logical-connectives we use in the system. For

[1]Meaning that if a set of formulas Γ logically implies a formula φ, then there is a proof of φ in the system with formulas in Γ added to the axioms.

concreteness, the reader can think of *the* ***Frege proof system*** as the following simple one (known as *Schoenfield's system*), consisting of only three axiom schemes (where $A \to B$ is an abbreviation of $\neg A \vee B$; and A, B, C are any propositional formulas):

$$A \to (B \to A)$$
$$(\neg A \to \neg B) \to ((\neg A \to B) \to A)$$
$$(A \to (B \to C)) \to ((A \to B) \to (A \to C)),$$

and a single inference rule (known as *modus ponens*):

$$\text{from } A \text{ and } A \to B, \text{ infer } B\,.$$

Frege systems are considered strong for several reasons. First, no super-polynomial lower bounds are known for Frege proofs, and moreover proving such lower bounds seems to be out of reach of current techniques, and believed by some to be even harder than proving explicit circuit lower bounds [Razborov, 2015]. Secondly, hard candidates for Frege systems are hard to find; common tautologies such as the pigeonhole principle that are known to be hard for weaker proof systems have polynomial-size Frege proofs. (See [Bonet et al., 1995, Razborov, 2015, Krajíček, 2011, Li and Tzameret, 2013] for further discussions on hard proof complexity candidates.)

An ***Extended Frege*** proof system is obtained by augmenting Frege with the axiom:

$$\text{Extension Axiom:} \qquad z \leftrightarrow \varphi\,,$$

where z is any *new* variable (namely, a variable that does not occur before in the proof) and φ is any formula (that does not contain z), and where the new variables z appearing in the extension axiom does not occur in the final formula in the proof. The point of the extension axiom is to allow the use of new variables to represent intermediate subformulas in a proof; with this new axiom scheme, polynomial-size Extended Frege proofs can reason about propositions computable by polynomial-size circuits (rather than just propositions computable by polynomial-size formulas, as is the case for polynomial-size Frege proofs).

For comprehensive texts on proof complexity and strong proof systems see e.g., the monograph by Krajíček [Krajíček, 1995] and [Clote and Kranakis, 2002, Chapter 5].

2.2 Comparing Proof Systems

To compare the relative strength of two proof systems we define the notion of a simulation. We say that a propositional proof system P **polynomially simulates** another propositional proof system Q if there is a polynomial-time computable function f that maps Q-proofs to P-proofs of the same tautologies (if P and Q use different representations for tautologies, we fix a (polynomial) translation from one representation to the other). In case f is computable in time $t(n)$ (for n the input-size), we say that P $t(n)$*-simulates* Q. We say that P and Q are *polynomially equivalent* in case P polynomially simulates Q and Q polynomially simulates P. If P polynomially simulates Q but Q does not polynomially simulate P we say that P *is strictly stronger than* Q (equivalently, that Q *is strictly weaker than* P).

2.3 Algebraic Circuits, Formulas, and Algebraic Complexity Classes

Algebraic circuits and formulas (over some fixed chosen field or ring) compute polynomials via addition and multiplication gates, starting from the input variables and constants from the field. More precisely, an *algebraic circuit* F is a finite directed acyclic graph with *input nodes* (i.e., nodes

of in-degree zero) and a single *output node* (i.e., a node of out-degree zero). Input nodes are labeled with either a variable or a field element in $\mathbb{F}$. All the other nodes have in-degree two (unless otherwise stated) and are labeled by either $+$ or $\times$. An input node is said to *compute* the variable or scalar that labels itself. A $+$ (or $\times$) gate is said to compute the addition (product, resp.) of the polynomials computed by its incoming nodes. An algebraic circuit is called a *formula*, if the underlying directed acyclic graph is a tree (that is, every node has at most one outgoing edge). The *size* of a circuit is the number of nodes in it, and the *depth* of a circuit is the length of the longest directed path in it.

Algebraic Complexity Classes We now recall some basic notions from algebraic complexity (for more details see [Shpilka and Yehudayoff, 2010, Sec. 1.2]). Over a ring R, VP_R (for "Valiant's P") is the class of families $f = (f_n)_{n=1}^{\infty}$ of formal polynomials f_n such that f_n has $\mathsf{poly}(n)$ input variables, is of $\mathsf{poly}(n)$ degree, and can be computed by algebraic circuits over R of $\mathsf{poly}(n)$ size. VNP_R (for "Valiant's NP") is the class of families g of polynomials g_n such that g_n has $\mathsf{poly}(n)$ input variables and is of $\mathsf{poly}(n)$ degree, and can be written as

$$g_n(x_1, \ldots, x_{\mathsf{poly}(n)}) = \sum_{\overline{e} \in \{0,1\}^{\mathsf{poly}(n)}} f_n(\overline{e}, \overline{x})$$

for some family $(f_n) \in \mathsf{VP}_R$.

A polynomial $f(\overline{x})$ is a *projection* of a polynomial $g(\overline{y})$ if $f(\overline{x}) = g(L(\overline{x}))$ identically as polynomials in $\overline{x}$, for some map L that assigns to each y_i either a variable or a constant. A family of polynomials (f_n) is a polynomial projection or *p-projection* of another family (g_n) if there is a function $t(n) = n^{\Theta(1)}$ such that f_n is a projection of $g_{t(n)}$ for all (sufficiently large) n. The *permanent* polynomial $\sum_{\sigma \in S_n} \prod_{i=1}^{n} x_{i,\sigma(i)}$ (for S_n the permutation group on n elements) is complete under p-projections for VNP. The *determinant* polynomial on the other hand is known to be in VP but is not known to be complete for VP under p-projections.

Two central questions in algebraic complexity theory are whether the permanent is a p-projection of the determinant (a stronger variant speaks about quasi-polynomial projections); and whether VP equals VNP [Valiant, 1979a, Valiant, 1979b, Valiant, 1982]. Since the permanent is complete for VNP (under p-projections), showing $\mathsf{VP} \neq \mathsf{VNP}$ amounts to proving that the permanent cannot be computed by polynomial-size algebraic circuits.

2.4 Algebraic Proof Systems

Let us now describe several algebraic proof systems for propositional logic (i.e. for boolean tautologies). Assume we start from a set of initial polynomials (called *axioms*) $f_1, \ldots, f_m \in \mathbb{F}[x_1, \ldots, x_n]$ over some field $\mathbb{F}$, then (the weak version of) Hilbert's Nullstellensatz shows that $f_1(\overline{x}) = \cdots = f_m(\overline{x}) = 0$ is unsatisfiable (over the algebraic closure of $\mathbb{F}$) if and only if there are polynomials $g_1, \ldots, g_m \in \mathbb{F}[\overline{x}]$ such that $\sum_j g_j(\overline{x}) f_j(\overline{x}) = 1$ (as a formal identity), or equivalently, that 1 is in the ideal generated by the $\{f_j\}_j$.

Beame, Impagliazzo, Krajíček, Pitassi, and Pudlák [Beame et al., 1996] suggested to treat these $\{g_j\}_j$ as a *proof* of the unsatisfiability of these axioms, called a ***Nullstellensatz refutation***. This is particularly relevant for complexity theory as one can restrict attention to *boolean* solutions to these axioms by adding the *boolean axioms*, that is, adding the polynomials $\{x_i^2 - x_i\}_{i=1}^{n}$ to the axioms. As such, one can then naturally encode NP-complete problems such as the satisfiability of 3CNF formulas as the satisfiability of a collection of constant-degree polynomials, and a Nullstellensatz

refutation is then an equation of the form

$$\sum_{j=1}^{m} g_j(\overline{x}) f_j(\overline{x}) + \sum_{i=1}^{n} h_i(\overline{x})(x_i^2 - x_i) = 1$$

for $g_j, h_i \in \mathbb{F}[\overline{x}]$. This proof system is sound and complete for refuting unsatisfiable axioms over $\{0,1\}^n$. Given that the above proof system is sound and complete, it is then natural to ask what is its power to refute unsatisfiable collections of polynomial equations over $\{0,1\}^n$. To understand this question one must define the notion of the *size* of the above refutations. Two popular notions are that of the *degree*, and the *sparsity* (number of monomials).

Strong (linear) lower bounds on Nullstellensatz degrees as well as strong (exponential) lower bounds on the sparsity of Nullstellensatz refutations are known (cf. [Beame et al., 1996, Buss et al., 1996, Razborov, 1998, Grigoriev, 1998, Impagliazzo et al., 1999, Buss et al., 2001, Alekhnovich and Razborov, 2001] and references therein). Unfortunately, the hard examples used for these lower bounds *do* admit polynomial-size proofs in stronger proof systems like Frege.

Therefore, to correspond more accurately to Frege, strong algebraic proof systems must use a more economical representation of polynomials in proofs than sum of monomials (similarly to the way a boolean formula is a much more succinct representation of a boolean function than a mere CNF). The natural way is to measure the size of a polynomial by the size of the minimal algebraic circuit or formula that computes it.

The idea to consider algebraic circuit size of algebraic proofs was raised initially by Pitassi [Pitassi, 1997] for Nullstellensatz written as algebraic circuits, and was investigated further in [Grigoriev and Hirsch, 2003, Raz and Tzameret, 2008b, Raz and Tzameret, 2008a, Tzameret, 2011] in the context of the polynomial calculus proof system.

Recently, Grochow and Pitassi [Grochow and Pitassi, 2014] have suggested the following algebraic proof system that resembles the Nullstellensatz, but with a variant that proved to have important consequences. A proof in the Ideal Proof System is given as a ***single*** polynomial, lending itself quite directly to algebraic circuit complexity techniques. In what follows we follow the notation in [Forbes et al., 2016a]:

Definition 2.1 (Ideal Proof System (IPS), Grochow-Pitassi [Grochow and Pitassi, 2014])**.** *Let $f_1(\overline{x}), \ldots, f_m(\overline{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ be a collection of polynomials. An **IPS refutation** for showing that the polynomials $\{f_j\}_j$ have no common solution in $\{0,1\}^n$ is an algebraic circuit $C(\overline{x}, \overline{y}, \overline{z}) \in \mathbb{F}[\overline{x}, y_1, \ldots, y_m, z_1, \ldots, z_n]$, such that*

1. *$C(\overline{x}, \overline{0}, \overline{0}) = 0$.*
2. *$C(\overline{x}, f_1(\overline{x}), \ldots, f_m(\overline{x}), x_1^2 - x_1, \ldots, x_n^2 - x_n) = 1$.*

*The **size** of the IPS refutation is the size of the circuit C. If C is of individual degree ≤ 1 in each y_j and z_i, then this is a **linear** IPS refutation (called basically* Hilbert *IPS by Grochow-Pitassi [Grochow and Pitassi, 2014]), which is abbreviated as IPS_{LIN}. If C comes from a restricted class of algebraic circuits $\mathcal{C}$, then this is called a $\mathcal{C}$-IPS refutation, and further called a $\mathcal{C}$-IPS_{LIN} refutation if C is linear in $\overline{y}, \overline{z}$. The variables $\overline{y}, \overline{z}$ are sometimes called the* placeholder variables *since they use as a placeholder for the axioms.*

Notice that the definition above adds the equations $\{x_i^2 - x_i\}_i$ to the system $\{f_j\}_j$. It is *not* necessary (for the sake of completeness) to add the equations $\overline{x}^2 - \overline{x}$ to the system in general, but this is the most interesting regime for proof complexity and thus we adopt it as part of our definition. Also, note that the first equality in the definition of IPS means that the polynomial

computed by C is in the ideal generated by $\overline{y}, \overline{z}$, which in turn, following the second equality, means that C witnesses the fact that 1 is in the ideal generated by $f_1(\overline{x}), \ldots, f_m(\overline{x}), x_1^2 - x_1, \ldots, x_n^2 - x_n$ (the existence of this witness, for unsatisfiable set of polynomials, stems from the Nullstellensatz theorem as discussed above).

It is not hard to show that IPS_{LIN} is polynomially equivalent to the Nullstellensatz system, when both are measured by their *circuit size*. For if we have an IPS_{LIN} refutation $C(\overline{x}, \overline{y}, \overline{z})$ we can turn it into a Nullstellensatz refutation by writing it as a sum of products of the (linear) variables $\overline{y}, \overline{z}$, with only a quadratic increase in size. For instance, if we write $\overline{z}'$ to denote $\overline{z}$ without z_n, we have $C(\overline{x}, \overline{y}, \overline{z}) = C(\overline{x}, \overline{y}, \overline{z}', z_n) = C(\overline{x}, \overline{y}, \overline{z}', 0) + (C(\overline{x}, \overline{y}, \overline{z}', 1) - C(\overline{x}, \overline{y}, \overline{z}', 0)) \cdot z_n$. Now, since $C(\overline{x}, \overline{y}, \overline{z}', 0)$ does not contain the variable z_n we can continue in a similar way to "take out" the rest of the variables in $\overline{y}, \overline{z}$, one by one, reaching a Nullstellensatz refutation (when substituting the f_i's and the boolean axioms for the $\overline{y}, \overline{z}$, respectively).

Furthermore, Forbes, Shpilka, Tzameret and Wigderson [Forbes et al., 2016a] showed that IPS_{LIN} refutations written as (general) algebraic circuits are *polynomially equivalent* to IPS (though for restricted classes $\mathcal{C}$, $\mathcal{C}$-IPS may differ from $\mathcal{C}$-IPS_{LIN}).

Considering both the Nullstellensatz and the IPS we can see that the main innovation in the IPS is the introduction of the placeholder variables $\overline{y}, \overline{z}$. This idea enables considering a refutation as a *single* polynomial instead of considering a collection of polynomials (that is, those polynomial coefficients of the initial axioms, as is the case of the Nullstellensatz).

Grochow-Pitassi [Grochow and Pitassi, 2014] showed that the IPS system is very powerful and can simulate Extended Frege (this follows from the fact that IPS is a generalization of the Nullstellensatz written as algebraic circuits and already [Pitassi, 1997] showed that the latter system simulates Extended Frege).

The fact that $\mathcal{C}$-IPS refutations are efficiently checkable (with randomness) follows from the fact that we only need to verify the polynomial identities stipulated by the definition. That is, it suffices to solve an instance of the ***polynomial identity testing*** *(PIT)* problem for the class $\mathcal{C}$: given a circuit from the class $\mathcal{C}$ decide whether it computes the identically zero polynomial. This problem is solvable in probabilistic polynomial time (BPP) for general algebraic circuits, and there are various restricted classes for which deterministic algorithms are known (see Section 6).

The Polynomial Calculus The Polynomial Calculus is an algebraic proof system introduced by [Clegg et al., 1996]. It can be considered as a "dynamic" version of the Nullstellensatz; namely, instead of providing a single certificate that 1 is in the ideal of the initial (unsatisfiable) polynomials, in PC we are allowed to derive the polynomial 1 step by step, by working in the ideal generated by the initial polynomials.

Definition 2.2 (**Polynomial Calculus (PC)**). *Let $\mathbb{F}$ be a field and let $F = \{f_1, \ldots, f_m\}$ be a collection of multivariate polynomials from $\mathbb{F}[x_1, \ldots, x_n]$. A* PC *proof from Q of a polynomial g is a finite sequence $\pi = (p_1, \ldots, p_\ell)$ of multivariate polynomials from $\mathbb{F}[x_1, \ldots, x_n]$, where $p_\ell = g$ and for every $1 \leq i \leq \ell$, either $p_i = f_j$ for some $j \in [m]$, or p_i is a boolean axiom $x_i \cdot (1 - x_i)$ for some $i \in [n]$, or p_i was derived from p_j, p_k, for $j, k < i$, by one of the following inference rules:*

(i) Product rule*: from p, derive $x_i \cdot p$, for $i \in [n]$;*

(ii) Addition rule*: from p, q, derive $ap + bq$, for $a, b \in \mathbb{F}$.*

A PC refutation of F *is a proof of* 1 *(which is interpreted as $1 = 0$, that is the unsatisfiable equation standing for* false*) from F.*

Similar to the Nullstellensatz, the standard complexity measures for PC are the *degree* of a PC proof, which is the maximal (total) degree of a polynomial in the proof and the *size* of a PC proof which is the total number of monomials (with nonzero coefficients) in all the PC proof lines. However, it is also possible to consider the total algebraic circuit size of all the PC proofs lines as a complexity measure.

Non-commutative Algebraic Proof Systems Motivated by the fact that the class of non-commutative formulas admits a deterministic PIT algorithm by Raz and Shpilka [Raz and Shpilka, 2005a], and even more importantly admits exponential-size lower bounds by Nisan [Nisan, 1991], Li, Tzameret and Wang [Li et al., 2015] considered a variant of the IPS over *non-commutative* polynomials written as non-commutative formulas. Their non-commutative IPS was shown to constitute a tighter characterization of Frege proofs than the original (commutative) IPS: first, proofs in this system are checkable in *deterministic* polynomial-time; and second, Frege can simulate (with a quasi-polynomial increase in size) non-commutative IPS refutations (over the field of two elements). But perhaps most importantly, the fact that we do have lower bounds on non-commutative formulas together with the characterization of any Frege proof as a single non-commutative formula, gives some hope to progress on the problem of Frege lower bounds. We discuss the non-commutative IPS in more details in Section 4.

3 IPS

In this section we show that lower bounds for IPS imply algebraic circuit lower bounds, namely that the permanent does not have polynomial-size algebraic circuits. This implication is interesting because it is a unique case in proof complexity where a lower bound on a *specific* proof system (on any tautology) is shown to imply explicit circuit lower bounds. We then compare the strength of IPS to Extended Frege. We show that IPS, in its full generality polynomially simulates Extended Frege, and on the other hand, show that Extended Frege polynomially simulates IPS if PIT has feasible correctness proofs in Extended Frege.[2]

3.1 Lower Bounds on IPS Imply Algebraic Circuit Lower Bounds

Theorem 3.1 (Grochow-Pitassi [Grochow and Pitassi, 2014])**.** *For any ring R, a super-polynomial lower bound on IPS proofs over R of any family of tautologies implies $\mathsf{VNP}_R \neq \mathsf{VP}_R$. A super-polynomial lower bound on the number of proof-lines in polynomial calculus proofs implies that the permanent is not a p-projection of the determinant.*

We will sketch the proof for the first half of the theorem which gives the main idea. The proof of the second half can be found in [Grochow and Pitassi, 2014].

Lemma 3.2. *Every family of unsatisfiable CNF formulas (φ_n) has a family of IPS certificates (C_n) in VNP_R.*

Proof of Theorem 3.1, assuming Lemma 3.2. Our proof is taken from [Grochow and Pitassi, 2014]. For a given set $\mathcal{F}$ of unsatisfiable polynomial equations $F_1 = \cdots = F_m = 0$, a lower bound on IPS refutations of $\mathcal{F}$ is equivalent to giving the same circuit lower bound on *all* IPS certificates for $\mathcal{F}$. A super-polynomial lower bound on IPS implies that some function in VNP—namely, the VNP-IPS

[2]Namely, that Extended Frege has polynomial-size proofs of the statement expressing that the PIT for algebraic circuits is decidable by polynomial-size Boolean circuits.

certificate guaranteed by Lemma 3.2—cannot be computed by polynomial-size algebraic circuits, and hence that $\mathsf{VNP} \neq \mathsf{VP}$. □

Proof sketch of Lemma 3.2. We mimic one of the proofs of completeness for linear IPS [Pitassi, 1997, Theorem 1] and then show that this proof can in fact be carried out in VNP. We omit any mention of the ground ring, as it will not be relevant.

Let $\varphi_n(\overline{x}) = \kappa_1(\overline{x}) \wedge \cdots \wedge \kappa_m(\overline{x})$ be an unsatisfiable CNF formula, where each κ_i is a disjunction of literals. Let $C_i(\overline{x})$ denote the (negated) polynomial translation of κ_i via $\neg x \mapsto x$, $x \mapsto 1 - x$ and $f \vee g \mapsto fg$; in particular, $C_i(\overline{x}) = 0$ if and only if $\kappa_i(\overline{x}) = 1$, and thus φ_n is unsatisfiable if and only if the system of equations $C_1(\overline{x}) = \cdots = C_m(\overline{x}) = x_1^2 - x_1 = \cdots = x_n^2 - x_n = 0$ is unsatisfiable. In fact, as we will see in the course of the proof, we will not need the equations $x_i^2 - x_i = 0$. It will be convenient to introduce the function $b(e, x) = ex + (1 - e)(1 - x)$, i.e., $b(1, x) = x$ and $b(0, x) = 1 - x$. For example, the clause $\kappa_i(\overline{x}) = (x_1 \vee \neg x_{17} \vee x_{42})$ gets translated into $C_i(\overline{x}) = (1 - x_1)x_{17}(1 - x_{42}) = b(0, x_1)b(1, x_{17})b(0, x_{42})$, and therefore an assignment falsifies κ_i if and only if $(x_1, x_{17}, x_{42}) \mapsto (0, 1, 0)$.

Just as $1 = x_1x_2 + x_1(1 - x_2) + (1 - x_2)x_1 + (1 - x_2)(1 - x_1)$, an easy induction shows that

$$1 = \sum_{\overline{e} \in \{0,1\}^n} \prod_{i=1}^{n} b(e_i, x_i). \tag{1}$$

We will show how to turn this expression into a VNP certificate refuting φ_n. Let c_i be the placeholder variable corresponding to $C_i(\overline{x})$.

The idea is to partition the assignments $\{0, 1\}^n$ into m parts $A_1, \ldots, A_m$, where all assignments in the i-th part A_i falsify clause i. This will then allow us to rewrite equation (1) as

$$1 = \sum_{i=1}^{m} C_i(\overline{x}) \left(\sum_{\overline{e} \in A_i} \prod_{j : x_j \notin \kappa_i} b(e_j, x_j) \right), \tag{2}$$

where "$x_j \notin \kappa_i$" means that neither x_j nor its negation appears in κ_i. Equation (2) then becomes the IPS-certificate $\sum_{i=1}^{m} c_i \cdot (\sum_{\overline{e} \in A_i} \prod_{j : x_j \notin \kappa_i} b(e_j, x_j))$. What remains is to show that the sum can indeed be rewritten this way, and that there is some partition $(A_1, \ldots, A_m)$ as above such that the resulting certificate is in fact in VNP.

First, let us see why such a partition allows us to rewrite (1) as (2). The key fact here is that the clause polynomial $C_i(\overline{x})$ divides the term $t_{\overline{e}}(\overline{x}) := \prod_{i=1}^{n} b(e_i, x_i)$ if and only if $C_i(\overline{e}) = 1$, if and only if $\overline{e}$ *falsifies* κ_i. Let $C_i(\overline{x}) = \prod_{i \in I} b(f_i, x_i)$, where $I \subseteq [n]$ is the set of indices of the variables appearing in clause i. By the properties of b discussed above, $1 = C_i(\overline{e}) = \prod_{i \in I} b(f_i, e_i)$ if and only if $b(f_i, e_i) = 1$ for all $i \in I$, if and only if $f_i = e_i$ for all $i \in I$. In other words, if $1 = C_i(\overline{e})$ then $C_i = \prod_{i \in I} b(e_i, x_i)$, which clearly divides $t_{\overline{e}}$. Conversely, suppose $C_i(\overline{x})$ divides $t_{\overline{e}}(\overline{x})$. Since $t_{\overline{e}}(\overline{e}) = 1$ and every factor of $t_{\overline{e}}$ only takes on boolean values on boolean inputs, it follows that every factor of $t_{\overline{e}}$ evaluates to 1 at $\overline{e}$, in particular $C_i(\overline{e}) = 1$.

Let $A_1, \ldots, A_m$ be a partition of $\{0, 1\}^n$ such that every assignment in A_i falsifies κ_i. Since C_i divides every term $t_{\overline{e}}$ such that $\overline{e}$ falsifies clause i, C_i divides every term $t_{\overline{e}}$ with $\overline{e} \in A_i$, and thus we can indeed rewrite (1) as (2).

Next, we show how to construct a partition $A_1, \ldots, A_m$ as above so that the resulting certificate is in VNP. The partition we will use is a greedy one. A_1 will consist of *all* assignments that falsify κ_1. A_2 will consist of all *remaining* assignments that falsify κ_2. And so on. In particular, A_i consists of all assignments that falsify κ_i and *satisfy* all A_j with $j < i$. (If at some clause κ_i before we reach the end, we have used up all the assignments—which happens if and only if the first i

clauses on their own are unsatisfiable—that's okay: nothing we've done so far nor anything we do below assumes that all A_i are nonempty.)

Equivalently, $A_i = \{\overline{e} \in \{0,1\}^n \mid C_i(\overline{e}) = 1 \text{ and } C_j(\overline{e}) = 0 \text{ for all } j < i\}$. For any property Π, we write $[\![\Pi(\overline{e})]\!]$ for the indicator function of Π: $[\![\Pi(\overline{e})]\!] = 1$ if and only if $\Pi(\overline{e})$ holds, and 0 otherwise. We thus get the certificate:

$$\begin{aligned}
&\sum_{i=1}^{m} c_i \cdot \left(\sum_{\overline{e}\in\{0,1\}^n} [\![\overline{e} \text{ falsifies } \kappa_i \text{ and satisfies } \kappa_j \text{ for all } j < i]\!] \prod_{j:x_j\notin\kappa_i} b(e_j, x_j) \right) \\
= &\sum_{i=1}^{m} c_i \cdot \left(\sum_{\overline{e}\in\{0,1\}^n} [\![C_i(\overline{e}) = 1 \text{ and } C_j(\overline{e}) = 0 \text{ for all } j < i]\!] \prod_{j:x_j\notin\kappa_i} b(e_j, x_j) \right) \\
= &\sum_{i=1}^{m} c_i \cdot \left(\sum_{\overline{e}\in\{0,1\}^n} \left(C_i(\overline{e}) \prod_{j<i} (1 - C_j(\overline{e})) \right) \prod_{j:x_j\notin\kappa_i} b(e_j, x_j) \right) \\
= &\sum_{e\in\{0,1\}^n} \sum_{i=1}^{m} c_i C_i(\overline{e}) \left(\prod_{j<i} (1 - C_j(\overline{e})) \right) \left(\prod_{j:x_j\notin\kappa_i} b(e_j, x_j) \right)
\end{aligned}$$

Finally, it is readily visible that the polynomial function of $\overline{c}$, $\overline{e}$, and $\overline{x}$ that is the summand of the outermost sum $\sum_{\overline{e}\in\{0,1\}^n}$ is computed by a polynomial-size circuit of polynomial degree, and thus the entire certificate is in VNP. □

3.2 IPS Polynomially Simulates Extended Frege

In this section we show that IPS polynomially simulates Extended Frege. For simplicity, we exemplify this simulation by showing how IPS written as algebraic *formulas* polynomially simulates the Frege proof system, but the proof for Extended Frege is quite similar. It was further shown by [Grochow and Pitassi, 2014] that restricted subsystems of IPS can polynomially simulate the corresponding restricted subsystem of Extended Frege, and specifically this holds for IPS written as constant-depth algebraic circuits and constant-depth Frege systems with modulo counting gates ($\mathsf{AC}^0[p]$-Frege).

Theorem 3.3 (Grochow-Pitassi [Grochow and Pitassi, 2014])**.** *Let φ be a 3CNF formula. If there is an Extended Frege proof (Frege proof) that φ is unsatisfiable in size-s, then there is an IPS refutation of circuit (formula, resp.) size* $\mathsf{poly}(|\varphi|, s)$.

Proof sketch. One way of thinking of this simulation (and similar simulations of propositional systems by IPS-variants) is to consider a two-step conversion of propositional proofs into IPS refutations as follows. First, turn the Frege refutation into a tree-like Frege refutation and convert each proof-line in the tree-like Frege refutation into an equivalent algebraic formula, obtaining a tree-like PC refutation. Secondly, convert the tree-like PC proof into a single formula, whose underlying formula-tree is precisely the underlying tree of the PC proof.

Note that a Frege proof can be converted into a *tree-like proof* with only a polynomial increase in size, that is, a proof in which every proof-line can be used at most once in modus ponens (cf. [Krajíček, 1995]). Therefore, we start from a tree-like Frege refutation of the unsatisfiable 3CNF formula (namely, a proof of false from the clauses of the 3CNF formula as assumptions), and show how to obtain from this an IPS refutation of the arithmetic version (see below) of the same 3CNF formula. Thus, a Frege proof of false from a given CNF φ is translated into an IPS proof of 1 from the initial (arithmetic version of) φ, yielding an IPS refutation of φ.

Step 1: This step involves the arithmetization of Frege proofs, namely, converting each Frege proof-line to an algebraic formula. The transformation converts a boolean formula into a corresponding algebraic formula (over the rationals, or $\mathbb{F}_q$, for a prime q; the simulation uses only the fact that the field has $1, 0$ and -1) that evaluates to 0 for all 0-1 assignments. This is done in the same way as in the proof of Lemma 3.2: true becomes 0, false becomes 1, a variable x_i becomes $1-x_i$, $\neg A$ becomes $1-\mathrm{tr}(A)$, where $\mathrm{tr}(A)$ denotes the translation of A, $A \vee B$ becomes the product of the corresponding translations $\mathrm{tr}(A) \cdot \mathrm{tr}(B)$, and $A \wedge B$ becomes $1-(1-\mathrm{tr}(A)) \cdot (1-\mathrm{tr}(B))$ (Schoenfield's system uses the $\rightarrow$ logical connective, but we can simply treat $A \rightarrow B$ as an abbreviation of $\neg A \vee B$). It is easy to check that for any 0-1 assignment, A evaluates to true iff $\mathrm{tr}(A) = 0$.

Once we converted every Frege proof-line into its corresponding algebraic formula, we get something that *resembles* a sequential algebraic proof, namely a PC proof. However, it is not precisely a legitimate PC proof because, e.g., every application of modus ponens (from A and $A \rightarrow B$ derive B) is translated into the purported rule *"from* $\mathrm{tr}(A)$ *and* $(1-\mathrm{tr}(A)) \cdot \mathrm{tr}(B)$ *derive* $\mathrm{tr}(B)$", which is not a formal rule in PC. Nevertheless, we can make this arithmetized Frege proof into a legitimate PC proof, except that our PC proof will have a *generalized* product rule: instead of being able to multiply a proof-line only by a *single* variable we will enable a product by a *polynomial*, namely, from f derive $g \cdot f$, for some polynomial $g \in \mathbb{F}[x_1, \ldots, x_n]$.

To form our (generalized) PC proof we simply *simulate* Schoenfield's system rules and axioms. Considering the example above, we need to construct a short PC proof of $\mathrm{tr}(B)$ from $\mathrm{tr}(A)$ and $(1-\mathrm{tr}(A)) \cdot \mathrm{tr}(B)$: first derive $\mathrm{tr}(A) \cdot \mathrm{tr}(B)$ by the generalized PC product rule and then add this to $(1-\mathrm{tr}(A)) \cdot \mathrm{tr}(B)$, to obtain $\mathrm{tr}(B)$. Similarly, Frege axioms are translated into algebraic formulas, that we then need to *derive* in PC, and this is possible to do efficiently.

Step 2: Here we transform the (generalized) PC refutation from step 1, whose underlying proof-graph is a tree (since we assumed without loss of generality that our initial Frege proof is a tree-like proof), into a *single formula* whose underlying graph is essentially the same tree. This formula constitutes the IPS refutation of the arithmetic translation of φ. The transformation from a PC proof to a formula is quite straightforward. For example, assume that in the PC proof we derived $g \cdot f$ from f. And suppose that we already built the IPS proof of f, namely $C(\overline{x}, f_1(\overline{x}), \ldots, f_m(\overline{x}), x_1^2 - x_1, \ldots, x_n^2 - x_n) = f$. Then, $g \cdot C(\overline{x}, f_1(\overline{x}), \ldots, f_m(\overline{x}), x_1^2 - x_1, \ldots, x_n^2 - x_n) = g \cdot f$ is the IPS proof of $g \cdot f$. Simulating the addition rule of PC is done in a similar manner. (Formally, the $f_i(\overline{x})$'s should be substituted by the placeholder variables $\overline{y}$, and the boolean axioms by the placeholder variables $\overline{z}$.)

It is easy to see that the resulted IPS is of size polynomial in the size of the PC refutation, which in turn is of size polynomial in the size of the original Frege refutation . □

3.3 PIT as a Bridge Between Circuit Complexity and Proof Complexity

In this section we sketch the argument that Extended Frege (EF) is polynomially equivalent to IPS if there are polynomial-size circuits for PIT whose correctness—suitably formulated—can be efficiently proved in EF. More precisely, we identify a small set of natural axioms for PIT and show that if these axioms can be proven efficiently in EF, then EF is p-equivalent to IPS.

The high-level idea is to formalize soundness of IPS as a sequence of propositional statements and then to show:

(1) if EF has efficient proofs of IPS soundness then EF can polynomially simulate IPS;

(2) Show that EF has efficient proofs of IPS soundness if a small set of natural axioms for PIT are efficiently provable in EF.

The idea behind (1) is not new and traces back to Hilbert; its counterpart for propositional proof systems was first formalized by Cook [Cook, 1975]. We explain the idea for propositional refutation systems here. Soundness of a propositional proof system states that any formula that has a proof (in the system) is a tautology. Formalizing soundness propositionally involves studying *partial* soundness, where we have a different propositional formula for each proof length. In more detail, for a propositional proof system Q, $Soundness_{Q,n}, n > 0$ will be a family of propositional statements. The underlying variables of $Soundness_{Q,n}$ are $\overline{x}$, $\overline{y}$ and $\overline{z}$, where we think of $\overline{x}$ as an encoding of some Q-proof of length n, $\overline{y}$ as an encoding of a k-DNF formula with $n' \leq n$ underlying variables, and $\overline{z}$ as a boolean assignment to the n' underlying variables. $Soundness_{Q,n}(\overline{x}, \overline{y}, \overline{z})$ is of the form $Proof_{Q,n}(\overline{x}, \overline{y}) \rightarrow Truth(\overline{y}, \overline{z})$ where $Proof_{Q,n}(\overline{x}, \overline{y})$ expresses that $\overline{x}$ is an encoding of a Q-proof of the formula encoded by $\overline{y}$, and $Truth(\overline{y}, \overline{z})$ expresses that $\overline{z}$ satisfies the formula encoded by $\overline{y}$ (i.e., the formula encoded by $\overline{y}$ is a tautology).

For sufficiently strong propositional proof systems P and Q, it is well-known that P polynomially simulates Q if and only if there are polynomial-sized P-proofs of $Soundness_{Q,n}$ for all $n > 0$. The intuitive argument is as follows: Suppose that Q has a short proof of some formula g; let $\alpha(g)$ be the encoding of g, and let $\beta(g)$ be the encoding of the short Q-proof of g. Then since P has short proofs of $Soundness_{Q,n}$, we instantiate this with g to give a short P-proof of $Soundness_{Q,n}(\beta(g), \alpha(g), \overline{z})$. Since $Proof_{Q,n}(\beta(g), \alpha(g))$ is a tautology and involves no propositional variables, it has a short P-proof and thus by modus ponens, there is a short P-proof of $Truth(\alpha(g), \overline{z})$. The last step is to demonstrate short P-proofs of $Truth(\alpha(g), \overline{z}) \rightarrow g$.

We will take P to be EF and Q to be IPS. Then EF can polynomially simulate Q if and only if EF can efficiently prove the soundness tautologies for IPS. Proving the soundness tautologies for IPS amounts to stating and proving (in Extended Frege) that if C is an algebraic circuit such that: (1) $C(\overline{x}, \overline{0}, \overline{0}) = 0$ and (2) $C(\overline{x}, f_1(\overline{x}), \ldots, f_m(\overline{x})) = 1$, then $f_1, \ldots, f_m$ is unsatisfiable. In order to state (1) and (2) efficiently, we need polynomial-sized circuits for polynomial identity testing. Then in order to prove that (1) and (2) imply that $f_1, \ldots, f_m$ is unsatisfiable, we will need to use basic properties of our PIT circuits. We omit the proof here, but will informally state the axioms that will be required in order to carry out the above plan.

3.4 Axioms for Circuits for Polynomial Identity Testing

Fix some standard boolean encoding of algebraic circuits, so that the encoding of any size-m algebraic circuit has size $\mathsf{poly}(m)$. We use "$[C]$" to denote the encoding of the algebraic circuit C. Let $K = (K_{m,n})$ denote a family of boolean circuits for solving polynomial identity testing. That is, $K_{m,n}$ is a boolean function that takes as input the encoding of a size m algebraic circuit, C, over variables $x_1, \ldots, x_n$, and if C has polynomial degree, then K outputs 1 if and only if the polynomial computed by C is the 0 polynomial.

The first axiom states that if C is a circuit over variables $\overline{x}$ computing the identically 0 polynomial, then the circuit C where we plug in a particular boolean input $\overline{p}$, still computes the identically 0 polynomial:

$$K([C(\overline{x})]) \rightarrow K([C(\overline{p})]).$$

The second axiom states that if C is a circuit over variables $\overline{x}$ computing the zero polynomial, then the circuit $1 - C$ does not compute the zero polynomial:

$$K([C(\overline{x})]) \rightarrow \neg K([1 - C(\overline{x})]).$$

The third axiom states that if the polynomial computed by circuit G is 0, then G can be substituted for the constant 0:

$$K([G(\overline{x})]) \wedge K([C(\overline{x}, 0)]) \rightarrow K([C(\overline{x}, G(\overline{x}))]).$$

Finally, the last axiom states that PIT is closed under permutations of the variables. More specifically if $C(\overline{x})$ is identically 0, then so is $C(\pi(\overline{x}))$ for all permutations π:

Note that the issue is not the existence of small circuits for PIT since we would be happy with nonuniform polynomial-size PIT circuits, which do exist. Unfortunately the known constructions are highly nonuniform—they involve picking random points—and we do not see how to prove the above axioms for these constructions. On the other hand, it is widely conjectured that there exist uniform polynomial-sized circuits for PIT, and it is therefore a very intriguing question whether or not the proofs of correctness of such uniform algorithms (assuming that they exist) can be carried out in a feasible (polynomial-time) proof system.

4 The Non-Commutative IPS

In this section we discuss the non-commutative IPS, introduced by Li, Tzameret and Wang [Li et al., 2015], which is a variant of the IPS over non-commutative polynomials. The main result is that the non-commutative IPS completely captures (up to quasi-polynomial factors) the Frege proof system when the non-commutative IPS refutations are written as non-commutative formulas.

Since the class of non-commutative formulas are well understood, namely, it admits exponential-size lower bounds by Nisan [Nisan, 1991], and deterministic PIT algorithm by Raz-Shpilka [Raz and Shpilka, 2005a], this characterization of a Frege proof by a single non-commutative formula gives some hope for better understanding of specific Frege proofs and specifically for the eventual possibility of providing lower bounds on Frege proofs.

We need to describe first the basic setup before giving the precise definition. A ***non-commutative polynomial*** is a polynomial in which products are non-commuting, namely, $x_i x_j$ is not the same polynomial as $x_j x_i$, whenever $i \neq j$. In other words, $x_i x_j - x_j x_i$ is not the zero polynomial. Thus, we can treat a non-commutative polynomial as a formal sum of non-commutative monomials. We denote by $\mathbb{F}\langle x_1, \ldots, x_n\rangle$ the ring of non-commutative polynomials over the variables $x_1, \ldots, x_n$. A ***non-commutative formula*** is the same as a (commutative) algebraic formula only that the children of product gates have *order*, so that we can record the order of multiplication. Therefore, the polynomial that a non-commutative formula computes is the polynomial achieved by first multiplying out brackets whereby we get a sum of monomials in which the order of multiplication matters (without performing still any cancelations of monomials), and then performing monomial cancelation (and grouping) only when two monomials have the same variables with the same powers and *the same order of multiplication.*

It helps to think of non-commutative polynomials (and formulas) as a means to compute functions over non-commutative domains such as matrix algebras (in which matrix product is non-commuting in general).

Definition 4.1 (Non-commutative IPS, Li-Tzameret-Wang [Li et al., 2015])**.** *Let $\mathbb{F}$ be a field. Let $f_1(\overline{x}), \ldots, f_m(\overline{x}) \in \mathbb{F}\langle\overline{x}\rangle$ be a system of non-commutative polynomials. A* ***non-commutative-IPS refutation*** *that the polynomials $\{f_j\}_j$ have no common solution in $\{0,1\}^n$,[3] is a non-commutative formula $\mathfrak{F}(\overline{x}, \overline{y}, \overline{z}, \overline{w}) \in \mathbb{F}\langle\overline{x}, y_1, \ldots, y_m, z_1, \ldots, z_n, w_1, \ldots, w_{\binom{n}{2}}\rangle$, such that*

1. $\mathfrak{F}(\overline{x}, \overline{0}, \overline{0}, \overline{0}) = 0$.
2. $\mathfrak{F}(\overline{x}, f_1(\overline{x}), \ldots, f_m(\overline{x}), \overline{x}^2 - \overline{x}, x_1x_2 - x_2x_1, \ldots, x_{n-1}x_n - x_nx_{n-1}) = 1$.

[3] One can check that the $f_i(\overline{x})$'s have no common 0-1 solutions in $\mathbb{F}$ iff they do not have a common 0-1 solution in every $\mathbb{F}$-algebra.

The $\overline{x}^2 - \overline{x}$ denotes the boolean axioms $x_i^2 - x_i$, for all $i \in [n]$, and $x_i x_j - x_j x_i$, for all $i < j \in [n]$, are called the commutator axioms. *The* ***size*** *of a non-commutative IPS refutation is the minimal size of a* non-commutative formula *computing the non-commutative-IPS refutation.*

The novelty in the non-commutative IPS in comparison to the original (commutative) IPS is simply that a single refutation is a non-commutative polynomial instead of a commutative one.

One way of thinking about a non-commutative IPS refutation is as a *commutative* IPS formula augmented with *additional proofs* for demonstrating that all the monomials computed along the way in this formula are indeed commuting. More precisely, consider a *commutative* IPS refutation written as a formula $F(\overline{x}, \overline{y}, \overline{z})$, such that $F(\overline{x}, f_1(\overline{x}), \ldots, f_m(\overline{x}), \overline{x}^2 - \overline{x}) = 1$ as a *commutative* formula but $F(\overline{x}, f_1(\overline{x}), \ldots, f_m(\overline{x}), \overline{x}^2 - \overline{x}) \neq 1$ as a non-commutative formula. In the commutative version of $F(\overline{x}, f_1(\overline{x}), \ldots, f_m(\overline{x}), \overline{x}^2 - \overline{x})$, two monomials computed by this refutation, say $x_1x_2x_3$ and $x_2x_1x_3$, will be considered the same monomial. However, in the non-commutative version these two monomials are considered distinct, so we need to add an explicit proof that $x_1x_2x_3$ is equal to $x_2x_1x_3$—in this case the proof added is simply $(x_1x_2 - x_2x_1) \cdot x_3$ which is a right product of a commutator axiom.

Note that to achieve the *completeness* of the system we *must* add the commutator axioms. Indeed, the non-commutative polynomial $1 + x_ix_j - x_jx_i$, for example, is unsatisfiable over 0-1 solutions, but it cannot be proven unsatisfiable without using the commutator axioms, because it *is* satisfiable over some non-commutative matrix algebra (and by soundness of the non-commutative IPS there cannot be a proof of its unsatisfiability).

The gist of Li-Tzameret-Wang's simulation of Frege by the non-commutative IPS is that even when we add the commutator axioms, and by that force each refutation to explicitly witness any cancelation between (commuting) monomials, we are still not weakening the system too much, namely, we still keep the system as strong as the Frege system. The reason for this is that in Frege we consider propositional formulas as purely *syntactic* terms. For example, if $F[z]$ is a propositional formula, then $F[(A \wedge B)/z]$ and $F[(B \wedge A)/z]$ (which are the results of substituting $A \wedge B$ and $B \wedge A$ for z in F, resp.) are two *different* formulas and the tautology $F[(A \wedge B)/z] \equiv F[(B \wedge A)/z]$ requires an explicit Frege proof.

Non-commutative IPS polynomially simulates Frege, and conversely, Frege quasi-polynomially simulates non-commutative IPS over $GF(2)$ (for the latter see next section):

Theorem 4.1 (Li-Tzameret-Wang [Li et al., 2015])**.** *Let φ be an unsatisfiable propositional formula. If Frege can prove that φ is unsatisfiable in size-s, then there is a non-commutative IPS refutation of formula size $\mathsf{poly}(|\varphi|, s)$ computing a polynomial of degree $\mathsf{poly}(|\varphi|, s)$. Further, this refutation is checkable in deterministic $\mathsf{poly}(|\varphi|, s)$ time.*

The idea to consider non-commutative formulas in algebraic proofs as well as adding the commutator axioms was considered originally by Tzameret [Tzameret, 2011], though in that work the proof system was built on the polynomial calculus and not the IPS, and therefore did not obtain the characterization of a Frege proof as a single non-commutative formula.

Let us sketch the proof of 4.1. We begin with the simulation of Frege by non-commutative IPS. The idea here is quite similar to the simulation of Frege by (formula) IPS (Theorem 3.3).

Non-commutative IPS polynomially simulates Frege (proof sketch). Let us consider, as in the proof of Theorem 3.3, a two-step simulation of Frege by non-commutative IPS. We start from a Frege proof of false from the formula φ serving as an assumption in the proof, that we assume without loss of generality is a tree-like proof.

Step 1: Here we convert each proof-line into an algebraic formula in the same way we did in the proof of Theorem 3.3, using the same translation function $\mathrm{tr}(\cdot)$, only now let $tr(\cdot)$ return a *non-*

commutative formula. So, for instance, assuming A and B are unequal, $\mathrm{tr}(A \vee B) = \mathrm{tr}(A) \cdot \mathrm{tr}(B) \neq \mathrm{tr}(B) \cdot \mathrm{tr}(A) = \mathrm{tr}(B \vee A)$ (note that any algebraic formula can represent either a commutative or a non-commutative polynomial; namely, a non-commutative formula computes a non-commutative polynomial by taking into account the order in which children of product gates appear in the formula).

Now, as before, we get a purported refutation of $\mathrm{tr}(\varphi)$ that only resembles a PC refutation, and in addition the polynomials are non-commutative. We wish to complement this purported proof into a legitimate algebraic proof operating with non-commutative polynomials; this will be in fact the *non-commutative PC* system defined by Tzameret [Tzameret, 2011]: it is similar to the PC proof system, only that polynomials are considered as non-commutative polynomials, the addition rule is the same as in PC, and the generalized product rule can be applied either from the right or from the left, namely, from f derive either $g \cdot f$ or $f \cdot g$, for some g; further, in addition to the boolean axioms, we add the commutator axioms $x_i x_i - x_j x_i$, for every pair of variables, to the system.

We consider the case of simulating the first axiom of Schoenfield's system $A \to (B \to A)$ in this non-commutative PC system. This will exemplify why we need to use the commutator axioms. Thus, consider the translation of this axiom under $\mathrm{tr}(\cdot)$. Recall that $\to$ is just an abbreviation. Then, $\mathrm{tr}(A \to (B \to A)) = \mathrm{tr}(\neg A \vee (\neg B \vee A)) = (1 - \mathrm{tr}(A)) \cdot ((1 - \mathrm{tr}(B)) \cdot \mathrm{tr}(A))$. Our goal is to construct a non-commutative PC proof of the following *non-commutative* polynomial:

$$(1 - \mathrm{tr}(A)) \cdot ((1 - \mathrm{tr}(B)) \cdot \mathrm{tr}(A)) \,. \tag{3}$$

For this purpose, we first derive the polynomial $\mathrm{tr}(A) - \mathrm{tr}(A)^2 = (1 - \mathrm{tr}(A)) \cdot \mathrm{tr}(A)$. This is doable efficiently using only the boolean axioms $x_i - x_i^2$ (by induction on the size of A). Then, we wish to derive (3) from $(1 - \mathrm{tr}(A)) \cdot \mathrm{tr}(A)$. We can multiply the latter by $(1 - \mathrm{tr}(B))$ from the right, to get $(1 - \mathrm{tr}(A)) \cdot \mathrm{tr}(A) \cdot (1 - \mathrm{tr}(B))$. Now we *must use the commutator axioms* to commute the rightmost product in order to derive (3).

Indeed, given the product of two formulas $f \cdot g$, it is possible to show by induction on the size of f, g, that using the commutator axioms one can derive with a size $|f + g|$ non-commutative PC proof the formula $g \cdot f$.

Step 2: Here we repeat almost precisely the same idea as in Step 2 of the proof of Theorem 3.3. We have a tree-like non-commutative PC refutation of $\mathrm{tr}(\varphi)$ (that possibly uses the commutator axioms) and we wish to turn it into a non-commutative formula that constitutes an IPS refutation of $\mathrm{tr}(\varphi)$. We do this by constructing a non-commutative formula whose underlying graph is the same underlying proof-graph, as we did before. □

4.1 Frege Quasi-Polynomially Simulates the Non-Commutative IPS

Theorem 4.2 (Li-Tzameret-Wang [Li et al., 2015])**.** *Let φ be an unsatisfiable CNF formula and $f_1, \ldots, f_m$ be the non-commutative formulas corresponding to its clauses via $tr(\cdot)$. If there is a non-commutative IPS refutation of size s of $f_1, \ldots, f_m$ over $GF(2)$, then there is a Frege proof of size $s^{O(\log s)}$ of the tautology $\neg\varphi$.*

For low-degree non-commutative IPS refutations, the proof of Theorem 4.2 achieves in fact a slightly stronger simulation than stated. Specifically, when the degree of the non-commutative IPS refutation is logarithmic in s, the Frege proof is of polynomial-size in s (see [Li et al., 2015] for details).

The higher-level argument is a short Frege proof of the correctness of the Raz-Shpilka [Raz and Shpilka, 2005a] deterministic PIT algorithm. This resembles the discussion in Section 3.3 about

PIT for (commutative) circuits. Indeed, the argument can be viewed as a realization—for the non-commutative case—of Grochow-Pitassi [Grochow and Pitassi, 2014] PIT-axioms framework (Section 3.4). The actual proof of Theorem 4.2 is rather technical and long because one needs to prove properties of the Raz-Shpilka PIT algorithm for non-commutative formulas within the restrictive framework of propositional (Frege) proofs. Let us sketch the main ideas in the proof.

Our goal is to prove $\neg\varphi$ in Frege, given a non-commutative IPS refutation π of φ. The proof uses the following five steps. First, we *balance* the non-commutative IPS π, so that its depth is logarithmic in its size. This follows more or less Hrubeš and Wigderson's [Hrubeš and Wigderson, 2014] construction. Second, consider the balanced π, which is a non-commutative polynomial identity over $GF(2)$, as a *boolean tautology*, by replacing plus gates with XORs and product gates with ANDs. Third, we use the so-called *reflection principle* to reduce the task of efficiently proving $\neg\varphi$ in Frege to the following task: show that any non-commutative formula identity over $GF(2)$, considered as a boolean tautology, has a short Frege proof (this part was discussed—for the commutative case—in Section 3.3). Fourth, for technical reasons we turn our non-commutative polynomial identities over $GF(2)$ (considered as boolean tautological formulas) into a sum of *homogenous* identities. This is the *only* step that is responsible for the *quasi-polynomial size increase* in Theorem 4.2. For this step we use an efficient Frege simulation of a result by Raz [Raz, 2013] who showed how to transform an algebraic formula into (a sum of) homogenous formulas in an efficient manner.

The fifth and final step is to actually construct short Frege proofs for homogenous non-commutative identities over $GF(2)$ translated into propositional tautologies. To this end we construct an efficient Frege proof of the correctness of the Raz-Shpilka PIT algorithm for non-commutative formulas [Raz and Shpilka, 2005b].

In conclusion, the fact that IPS written as a non-commutative formula (with the additional commutator axioms) characterizes Frege proofs, naturally motivates studying $\mathcal{C}$-IPS for various restricted classes $\mathcal{C}$ of algebraic circuits. Lower bounds for such proofs intuitively correspond to lower bounds for restrictions of the Extended Frege proof system. This is the content of the next section.

5 Lower Bounds on Fragments of IPS

In Section 3.1 we have seen that proving super-polynomial lower bounds on the size of IPS certificates (written as algebraic circuits) would imply a separation of VP from VNP. On the other hand, in Section 4 we have seen that already proving lower bounds on IPS certificates when they are written as non-commutative formulas and augmented with the commutator axioms would imply Frege lower bounds. It is then natural to attempt to obtain lower bounds on IPS refutations where the certificates are written as an algebraic circuit from a restricted circuit class $\mathcal{C}$. Recall the notation $\mathcal{C}$-IPS from Definition 2.1, denoting that the IPS certificate $C(\overline{x},\overline{y},\overline{z})$ is taken from the class $\mathcal{C}$.[4] If the "placeholder" variables $\overline{y},\overline{z}$ in C are linear we call the certificate a $\mathcal{C}$-IPS$_{\text{LIN}}$ certificate. Super-polynomial lower bounds on the size of $\mathcal{C}$-IPS$_{\text{LIN}}$ refutations were recently shown by Forbes, Shpilka, Tzameret and Wigderson [Forbes et al., 2016a] when $\mathcal{C}$ is the class of read once (oblivious) algebraic branching programs (roABPs), multilinear formulas and diagonal circuits. We now survey some of these lower bounds.

Let us describe the main strategy behind the proofs, which is new, and exemplifies the potential of the algebraic complexity-based approach in proof complexity. One feature of these proof-size lower bounds is that they stem almost directly from circuit-size lower bounds.

[4]Note that there is a slight technical difference between requiring that $C(\overline{x},\overline{y},\overline{z})$ is taken from $\mathcal{C}$ and requiring that $C(\overline{x},\overline{F}(\overline{x}),x_1^2-x_1,\ldots,x_n^2-x_n)$ is taken from $\mathcal{C}$. In $\mathcal{C}$-IPS we require the former.

Assume that $f(\overline{x}) = 0$ has no 0-1 solutions over some field $\mathbb{F}$, and let

$$g(\overline{x}) \cdot f(\overline{x}) + \sum_{i=1}^{n} h_i(\overline{x}) \cdot (x_i^2 - x_i) = 1\,, \tag{4}$$

be the Nullstellensatz (equivalently, IPS_{LIN}) refutation of $f(\overline{x}) = 0$. We are going to lower bound the size of circuits computing $g(\overline{x})$. If we restrict our attention in (4) to only 0-1 assignments, then the boolean axioms vanish, and we have

$$g(\overline{x}) \cdot f(\overline{x}) = 1\,, \qquad \text{for } \overline{x} \in \{0,1\}^n, \tag{5}$$

where (5) is now a *functional identity* (in contrast to a *formal* identity between polynomials). That is, we consider $g(\overline{x}) \cdot f(\overline{x})$ as a function from $\{0,1\}^n$ to $\mathbb{F}$, and conclude that this is the constant 1 function. Hence

$$g(\overline{x}) = \frac{1}{f(\overline{x})}\,, \qquad \text{for } \overline{x} \in \{0,1\}^n\,.$$

Therefore, to lower bound the algebraic circuit size of Nullstellensatz refutations of $f(\overline{x})$ it suffices to lower bound the algebraic circuit size of every polynomial that computes the *function* $1/f(\overline{x})$ over 0-1 assignments. Since we wish to prove a lower bound for a family of polynomials computing a certain function over 0-1, instead of a lower bound for a specific formal polynomial, this kind of lower bound is called a *functional lower bound* (see also [Grigoriev and Razborov, 2000, Forbes et al., 2016b]). Forbes et al. [Forbes et al., 2016a] showed that for $f(\overline{x})$ being a variant of the subset-sum principle $\sum_{i=1}^{n} \alpha_i x_i - m$, for $\alpha_i \in \mathbb{F}$ and $m \notin \{\sum_{i=1}^{n} \alpha_i x_i \ : \ \overline{x} \in \{0,1\}^n\}$, one can obtain strong functional lower bounds on the algebraic circuit-size of $1/f(\overline{x})$, for certain circuit classes, and thus concluded IPS refutation lower bounds for these circuit classes.

The lower bounds obtained in [Forbes et al., 2016a], stated below, are for IPS over multilinear formulas and read once (oblivious) algebraic branching programs (roABP). A *multilinear formula* is simply an algebraic formula (Section 2.3) in which every node computes a multilinear polynomial. For the definition of roABPs and the proof of the lower bounds we refer the reader to [Forbes et al., 2016a].

Theorem 5.1 (Forbes-Shpilka-Tzameret-Wigderson [Forbes et al., 2016a])**.** *Let $n \geq 1$ and $\mathbb{F}$ be a field with characteristic bigger than $\binom{2n}{n}$. Suppose that $f(\overline{x}, \overline{z}) = \sum_{i<j\in[2n]} z_{i,j} x_i x_j - m$ is a polynomial over $\mathbb{F}$ that has no 0-1 roots. Then, any $\mathcal{C}$-IPS$_{LIN}$ refutation of $f(\overline{x}, \overline{z})$ requires:*

1. *$n^{\Omega(\log n)}$-size when $\mathcal{C}$ is the class of multilinear formulas;*
2. *$2^{n^{\Omega(1)}}$-size when $\mathcal{C}$ is the class of constant-depth multilinear formulas; and*
3. *$2^{\Omega(n)}$-size when $\mathcal{C}$ is the class of roABPs (in every variable order).*

Forbes et al. [Forbes et al., 2016a] also obtained nontrivial upper bounds on $\mathcal{C}$-IPS_{LIN} for $\mathcal{C}$ the class of multilinear formulas and roABPs. In particular they showed that both of these $\mathcal{C}$-IPS_{LIN} proof systems are strictly stronger than Nullstellensatz (measured by total number of monomials) and admit polynomial-size refutations of the subset-sum variant $\sum_{i=1}^{n} \alpha_i x_i - m$, for $\alpha_i, m \in \mathbb{F}$ and $m \notin \{\sum_{i=1}^{n} \alpha_i x_i \ : \ \overline{x} \in \{0,1\}^n\}$. For more details see [Forbes et al., 2016a].

6 PIT and Proof Complexity

We already discussed the polynomial identity testing (PIT) problem in the context of both IPS and the non-commutative IPS. There, we were interested in the following question:

Can propositional proofs efficiently prove the correctness of a PIT algorithm for a given circuit class?

We have seen in Section 3.4 that the PIT axioms capture the statements that express the correctness of a PIT algorithm (formally, a circuit for PIT). In other words, providing short Extended Frege proofs for the PIT axioms would de facto mean that Extended Frege efficiently proves the correctness of (some) polynomial-size circuits for PIT; from which it follows that Extended Frege polynomially simulates IPS. Subsequently, in Section 4.2, we showed that Frege *does* admit efficient (quasi-polynomial) proofs of the correctness of the Raz-Shpilka deterministic PIT algorithm for non-commutative formulas. This, in turn, implies that Frege quasi-polynomially simulates the non-commutative IPS (over $GF(2)$).

6.1 Proof Systems for Polynomial Identities

Hrubeš and Tzameret [Hrubeš and Tzameret, 2009] asked the following question concerning the connection between proof complexity and the PIT problem:

Can we efficiently prove *polynomial identities? And specifically, is there a sequential proof system admitting polynomial-size proofs for all polynomial identities?*

In other words, this question asks whether the PIT problem admits short proofs, and specifically, whether there is a simple sequential proof system to witness that PIT has short proofs.

We know that an efficient *probabilistic* algorithm for PIT exists, due to Schwartz and Zippel [Schwartz, 1980, Zippel, 1979]: when the field is sufficiently large, with high probability, two different polynomials will differ on a randomly chosen field assignment. However, whether the PIT problem is in P, namely is solvable in *deterministic* polynomial-time, is of course a major open problem in complexity and derandomization theory. In fact, it is not even known whether there are sub-exponential-size *witnesses* that two algebraic formulas compute the same polynomial; namely, whether there is a *non-deterministic* sub-exponential-time algorithm for PIT, and in yet other words, whether PIT is in NP (note that PIT *is* known to be in $\mathsf{coRP}\subseteq\mathsf{coNP}$).

Hrubeš-Tzameret's work [Hrubeš and Tzameret, 2009] investigated the PIT$\in$?NP question from the proof-complexity perspective: assuming that PIT does posses short witnesses, is it the case that a proof system using only symbolic manipulation (resembling a Frege proof) is enough to provide these short witnesses? And if not, can we prove lower bounds on such proofs? Such lower bounds would at least delineate the methods that will work for efficiently proving polynomial identities and those that will not.

The work in [Hrubeš and Tzameret, 2009], as well as the subsequent work [Hrubeš and Tzameret, 2012], set out to define the analogs of Frege and Extended Frege for the PIT problem that we shall call *PI proofs* (for Polynomial Identity Proofs; originally these systems were called *arithmetic proofs*): just as Frege and Extended Frege prove boolean tautologies by deriving new tautological formulas (and circuits, resp.), PI proofs prove polynomial identities by deriving new identities between algebraic formulas (and circuits, resp.).

Let us first describe the analog of Frege for PIT, namely the PI proof operating with algebraic formulas, denoted $\mathbb{P}_f$ (where f stands for "formulas"). $\mathbb{P}_f$ is a sequential proof system whose axioms are the polynomial-ring axioms and whose derivation rules express the properties of the equality symbol. Each proof-line in the system is an equation between two algebraic formulas (or circuits; see below) F, G computing polynomials over a given field $\mathbb{F}$ written as $F = G$. The proof system is sound and complete for true polynomial identities:

Theorem 6.1 ([Hrubeš and Tzameret, 2009])**.** *Let $\mathbb{F}$ be a field. For any pair F, G of algebraic formulas, there is a PI proof in the system $\mathbb{P}_f$ of $F = G$ iff F and G compute the same polynomial.*

The specific description of the rules and axioms of the PI proof system $\mathbb{P}_f$ are quite natural. The inference rules of $\mathbb{P}_f$ are (with F, G, H formulas; where an equation below a line can be inferred from the one above the line):

$$\frac{F = G}{G = F} \qquad \frac{F = G \qquad G = H}{F = H} \qquad \frac{F_1 = G_1 \qquad F_2 = G_2}{F_1 \circ F_2 = G_1 \circ G_2} \text{ for } \circ \in \{+, \cdot\}\,.$$

And the axioms of $\mathbb{P}_c$ express reflexivity of equality ($F = F$), commutativity and associativity of addition and product ($F \circ G = G \circ F$, and $F \circ (G \circ H) = (F \circ G) \circ H$, for $\circ \in \{+, \cdot\}$), distributivity ($F \cdot (G + H) = F \cdot G + F \cdot H$), zero element ($F + 0 = F$, $F \cdot 0 = 0$), unit element ($F \cdot 1 = F$) and true identities in the field ($a \circ b = c$, for $\circ \in \{+, \cdot\}$ and $a, b, c \in \mathbb{F}$).

A *PI proof* $\mathbb{P}_f$ is thus a sequence of equations $F_1 = G_1$, $F_2 = G_2, \ldots, F_k = G_k$, with F_i, G_i formulas, such that every equation is either an axiom or was obtained from previous equations by one of the inference rules. The *size* of a proof is the total size of all formulas appearing in the proof. It is easy to see that, just like a Frege proof, a PI proof can be verified for correctness in polynomial-time (assuming the field has efficient representation; e.g., the field of rational numbers).

It is important to notice the distinction between PI proofs and propositional proofs: PI proofs prove polynomial identities (a language in coRP) while propositional proofs prove boolean tautologies (a language in coNP). See more on this in Section 6.1 below.

The analog of Extended Frege for PIT, denoted $\mathbb{P}_c$, is identical to $\mathbb{P}_f$, except that it operates with equations between algebraic *circuits* instead of algebraic formulas (similar to Jeřabek's Circuit Frege [Jeřábek, 2004], formally, one needs to add another rule to such a system to be able to symbolically manipulate circuits, namely to merge two separate but identical sub-circuits into a single sub-circuit; see [Hrubeš and Tzameret, 2012] for the details.)

It turns out that PI proofs are in fact quite strong. First, [Hrubeš and Tzameret, 2009] only demonstrated lower bounds on very restricted fragments of PI proofs, and apparently it is quite hard to go beyond these restricted fragments of PI proof systems (assuming any nontrivial lower bound even exists). Furthermore, PI proofs were found to admit short proofs for many non-trivial polynomial identities (like identities based on symmetric polynomials). Moreover, PI proofs are able to "simulate" PIT algorithms for restricted algebraic circuit classes; specifically, Dvir and Shpilka's PIT algorithm for restricted depth-3 algebraic circuits [Dvir and Shpilka, 2006]. But more importantly, PI proofs were shown in [Hrubeš and Tzameret, 2012] to efficiently simulate many of the classical structural results on algebraic circuits.

In particular, PI proofs $\mathbb{P}_c$ operating with equations between algebraic circuits, efficiently simulate the following constructions: (i) *homogenization* of algebraic circuits (implicit in [Strassen, 1973]); (ii) Strassen's technique for *eliminating division gates* over large enough fields (also in [Strassen, 1973]); (iii) eliminating division gates over small fields—this is done by simulating large fields in small ones; and (iv) *balancing algebraic circuits* (Valiant et al. [Valiant et al., 1983]; see also [Hyafil, 1979]). Most notably, the latter result gives a strong *depth reduction* for polynomial-size $\mathbb{P}_c$ proofs to polynomial-size $O(\log^2 n)$-depth $\mathbb{P}_c$ proofs and a quasi-polynomial simulation of $\mathbb{P}_c$ by $\mathbb{P}_f$. This is one important point where the PI proof systems differ from Frege and Extended Frege, for which no such simulation is known.

Since depth reduction is the most important of these results, let us state this more formally:

Theorem 6.2 (Depth reduction for PI proofs [Hrubeš and Tzameret, 2012])**.** *Assume that F, G are circuits of (syntactic[5]) degree $\leq d$ and depth $\leq t$. If $F = G$ has a $\mathbb{P}_c$ proof of size s then it has a $\mathbb{P}_c$ proof of size* $\mathsf{poly}(s, d)$ *and depth $O(t + \log s \cdot \log d + \log^2 d)$.*

Intuitively, one can think of this theorem as showing that VNC^2-PI proofs are equal in strength to VP-PI proofs, similar to the strong depth collapse manifested for algebraic circuits by Valiant et al. [Valiant et al., 1983] who showed that $\mathsf{VNC}^2 = \mathsf{VP}$ (where VNC^2 is defined similar to VP except that the depth of the circuits computing f_n is required to be $O(\log^2 n)$).

As we now discuss, this also had implications for understanding *propositional proofs*.

Algebraic Fragments of Propositional Proofs Recall the Frege proof system described in Section 2.1. Each Frege proof-line is a propositional tautological formula, which is either a (substitution instance of an) axiom or was derived by the modus ponens rule. As mentioned before, Reckhow [Reckhow, 1976] proved that it does not matter which derivation rules and axioms we use, nor even the specific logical connectives (gates) used: as long as we use a finite number of rules and axioms[6] and the rules and axioms are (implicationally) complete, every two Frege systems are polynomially equivalent.

Now, consider the PI proof system $\mathbb{P}_f$, and assume the underlying field is $GF(2)$. In this case, *every PI proof-line becomes a boolean tautology*, where "+" becomes the logical gate XOR, "$\cdot$" becomes AND and "=" becomes the logical equivalence gate $\equiv$ (indeed, note that over $GF(2)$ the axioms become propositional tautologies). This then means that PI proofs over $GF(2)$ *are by themselves propositional Frege proofs*. The converse, on the other hand, is not true: not all Frege proofs are arithmetic proofs over $GF(2)$, because PI proofs over $GF(2)$ are *not* complete for the set of tautologies. For instance, $x_i^2 + x_i = 0$ is, over $GF(2)$ the boolean tautology $(x_i \wedge x_i) \oplus x_i \equiv \mathsf{false}$ over $GF(2)$, but it is not a true identity between (formal) polynomials and thus cannot be proved by a PI proof. In fact, Frege system is equivalent to the PI-system $\mathbb{P}_f$ over $GF(2)$ *augmented with the boolean axioms* $x_i^2 + x_i$; and similarly for Extended Frege and $\mathbb{P}_c$ over $GF(2)$.

Considering PI proofs as the "algebraic fragment" of propositional proofs gives us a new understanding of propositional proofs, and should hopefully shed more light on the complexity of Frege proofs. As mentioned above, it shows for instance a strong depth collapse, namely, that additional depth (beyond $O(\log^2 n)$) does not help to decrease the complexity of proofs. It is specifically useful for upper bounds questions on propositional proofs: if we can efficiently prove an algebraic identity over $GF(2)$ with a PI proof we can do the same for propositional proofs. This observation was used in [Hrubeš and Tzameret, 2012] to give a polynomial-size and depth-$O(\log^2 n)$ Extended Frege proof of the determinant identities $Det(A) \cdot Det(B) = Det(AB)$ and other linear-algebraic statements such as the matrix inverse principle $AB = I_n \rightarrow BA = I_n$. By, essentially, unwinding depth-$O(\log^2 n)$ *circuits* into quasi-polynomial-size *formulas*, one can obtain quasi-polynomial-size Frege proofs of the same statements. These results give presumably tight upper bounds for the proof complexity of linear algebra, because, for example, Bonet, Buss and Pitassi conjectured that Frege does not admit polynomial-size proofs of these identities [Bonet et al., 1995].

7 Conclusion and Open Problems

In this survey we demonstrated the emerging algebraic complexity approach to proof complexity. It is natural to expect that this close interaction between algebraic and proof complexity will

[5]The syntactic degree of an algebraic circuit is the maximal total degree of a monomial computed after multiplying out all brackets in the circuit (without cancelations of monomials).

[6]The number of axioms and rules is finite, but they obviously induce infinite many substitution instances of axioms and rules, since the axioms and rules are closed under substitution of the variables in the axioms and rules by formulas

continue to contribute new insights to proof complexity. Already now very interesting and sometimes surprising new ideas came out from this interaction. In particular, we have seen that proof complexity lower bounds (for IPS restricted subsystems) are drawn almost directly from algebraic circuit lower bounds; new connections between computation and proofs, showing that some proof complexity lower bounds (IPS) imply computational lower bounds and complexity class separations ($\mathsf{VP}\neq\mathsf{VNP}$); and conversely, proving that certain lower bounds on weak computational models (non-commutative formulas) would imply strong Frege lower bounds; characterizing the "algebraic fragments" of Frege and Extended Frege systems and using structural properties of algebraic circuits yield a better understanding of the power of these systems through (apparently) tight short proofs for basic statements in linear algebra. All of these results have been achieved using methods from algebraic complexity.

The big challenge ahead is of course to find out whether the algebraic approach can eventually lead to lower bounds on Frege and Extended Frege, or conversely help to at least establish a formal (unconditional) so-called 'barrier' against proving such lower bounds (e.g., by showing that Extended Frege lower bounds imply strong explicit circuit lower bounds). But before this seemingly formidable challenge, there are many important intermediate problems which seem relatively feasible at the moment, and whose solution will advance the frontiers of our understanding. We end by listing some of these problems.

- Can we obtain size lower bounds on constant-depth Frege with modular gates proofs ($\mathrm{AC}^0[p]$-Frege proofs)? This problem has been open for decades, despite the known $\mathrm{AC}^0[p]$-circuits lower bounds. It is quite conceivable that algebraic techniques may be of help on this (cf. [**?**, **?**, Buss et al., 2012]).

- Can we establish size lower bounds on $\mathcal{C}$-IPS (linear or not) refutations for natural encodings of CNFs, for restricted circuit classes $\mathcal{C}$? The lower bounds from [Forbes et al., 2016a] hold only for a single hard axiom, and not CNFs.

- Can we extend the $\mathcal{C}$-IPS lower bounds to "dynamic" versions of $\mathcal{C}$-IPS? For instance, can we prove lower bounds on PC refutations operating with multilinear formulas or roABPs as in [Raz and Tzameret, 2008b, Tzameret, 2011]?

- Lower bounds on PI proofs of polynomial identities? Almost no lower bound is known for these "algebraic fragments" of Frege and Extended Frege.

- Just like PI proofs are proofs for the (algebraic) language of polynomial identities, it is very interesting to study the complexity of proof systems for other algebraic languages. Two examples of such proof systems are the proof systems for matrix identities investigated in [Li and Tzameret, 2013], and the proof system for non-commutative rational identities defined in [Garg et al., 2015]. Can we prove strong proof-size lower bounds on these systems? Can we connect these systems further to propositional proof complexity or algebraic circuit complexity?

8 Acknowledgements

We thank Stephen Cook, Kaveh Ghasemloo, Amir Shpilka and Avi Wigderson for very helpful discussions and clarifications, and Michael Forbes for very useful discussions and comments on a preliminary draft. We would also like to thank Neil Immerman for his careful reading and useful comments on this survey. Finally we are especially grateful to Joshua Grochow for many conversations and for answering our many questions that greatly improved this survey.

References

[Alekhnovich and Razborov, 2001] Alekhnovich, M. and Razborov, A. A. (2001). Lower bounds for polynomial calculus: Non-binomial case. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2001)*, pages 190–199.

[Beame et al., 1996] Beame, P., Impagliazzo, R., Krajíček, J., Pitassi, T., and Pudlák, P. (1996). Lower bounds on Hilbert's Nullstellensatz and propositional proofs. *Proc. London Math. Soc. (3)*, 73(1):1–26. Preliminary version in the *35th Annual IEEE Symposium on Foundations of Computer Science (FOCS 1994)*.

[Bonet et al., 1995] Bonet, M. L., Buss, S. R., and Pitassi, T. (1995). Are there hard examples for Frege systems? In *Feasible mathematics, II (Ithaca, NY, 1992)*, volume 13 of *Progr. Comput. Sci. Appl. Logic*, pages 30–56. Birkhäuser Boston, Boston, MA.

[Buss et al., 2001] Buss, S. R., Grigoriev, D., Impagliazzo, R., and Pitassi, T. (2001). Linear gaps between degrees for the polynomial calculus modulo distinct primes. *JCSS*, 62(2):267–289. Preliminary version in the *14th Annual IEEE Conference on Computational Complexity (CCC 1999)*.

[Buss et al., 1996] Buss, S. R., Impagliazzo, R., Krajíček, J., Pudlák, P., Razborov, A. A., and Sgall, J. (1996). Proof complexity in algebraic systems and bounded depth Frege systems with modular counting. *Computational Complexity*, 6(3):256–298.

[Buss et al., 2012] Buss, S. R., Kolodziejczyk, L. A., and Zdanowski, K. (2012). Collapsing modular counting in bounded arithmetic and constant depth propositional proofs. *Transactions of the AMS (to appear)*.

[Clegg et al., 1996] Clegg, M., Edmonds, J., and Impagliazzo, R. (1996). Using the Groebner basis algorithm to find proofs of unsatisfiability. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC 1996)*, pages 174–183.

[Clote and Kranakis, 2002] Clote, P. and Kranakis, E. (2002). *Boolean functions and computation models*. Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag, Berlin.

[Cook, 1975] Cook, S. A. (1975). Feasibly constructive proofs and the propositional calculus (preliminary version). In *STOC*, pages 83–97.

[Cook and Reckhow, 1974a] Cook, S. A. and Reckhow, R. A. (1974a). On the lengths of proofs in the propositional calculus (preliminary version). In *Proceedings of the 6th Annual ACM Symposium on Theory of Computing (STOC 1974)*, pages 135–148. For corrections see Cook-Reckhow [Cook and Reckhow, 1974b].

[Cook and Reckhow, 1974b] Cook, S. A. and Reckhow, R. A. (1974b). Corrections for "On the lengths of proofs in the propositional calculus (preliminary version)". *SIGACT News*, 6(3):15–22.

[Cook and Reckhow, 1979] Cook, S. A. and Reckhow, R. A. (1979). The relative efficiency of propositional proof systems. *J. Symb. Log.*, 44(1):36–50. This is a journal-version of Cook-Reckhow [Cook and Reckhow, 1974a] and Reckhow [Reckhow, 1976].

[Dvir and Shpilka, 2006] Dvir, Z. and Shpilka, A. (2006). Locally decodable codes with 2 queries and polynomial identity testing for depth 3 circuits. *SIAM J. on Computing*, 36(5):1404–1434.

[Forbes et al., 2016a] Forbes, M., Shpilka, A., Tzameret, I., and Wigderson, A. (2016a). Proof complexity lower bounds from algebraic circuit complexity. In *31st Conference on Computational Complexity, (CCC)*.

[Forbes et al., 2016b] Forbes, M. A., Kumar, M., and Saptharishi, R. (2016b). Functional lower bounds for arithmetic circuits and boolean circuit complexity. In *31st Conference on Computational Complexity, (CCC)*.

[Garg et al., 2015] Garg, A., Gurvits, L., Oliveira, R., and Wigderson, A. (2015). A deterministic polynomial time algorithm for non-commutative rational identity testing. *CoRR*, abs/1511.03730.

[Grigoriev, 1998] Grigoriev, D. (1998). Tseitin's tautologies and lower bounds for Nullstellensatz proofs. In *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science (FOCS 1998)*, pages 648–652.

[Grigoriev and Hirsch, 2003] Grigoriev, D. and Hirsch, E. A. (2003). Algebraic proof systems over formulas. *Theoret. Comput. Sci.*, 303(1):83–102. Logic and complexity in computer science (Créteil, 2001).

[Grigoriev and Razborov, 2000] Grigoriev, D. and Razborov, A. A. (2000). Exponential lower bounds for depth 3 arithmetic circuits in algebras of functions over finite fields. *Appl. Algebra Engrg. Comm. Comput.*, 10(6):465–487.

[Grochow and Pitassi, 2014] Grochow, J. A. and Pitassi, T. (2014). Circuit complexity, proof complexity, and polynomial identity testing. In *Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2014)*, pages 110–119. Full version at `arXiv:abs/1404.3820`.

[Haken, 1985] Haken, A. (1985). The intractability of resolution. *Theoret. Comput. Sci.*, 39(2-3):297–308.

[Hrubeš and Wigderson, 2014] Hrubeš, P. and Wigderson, A. (2014). Non-commutative arithmetic circuits with division. In *Innovations in Theoretical Computer Science, ITCS'14, Princeton, NJ, USA, January 12-14, 2014*, pages 49–66.

[Hrubeš and Tzameret, 2009] Hrubeš, P. and Tzameret, I. (2009). The proof complexity of polynomial identities. In *Proceedings of the 24th IEEE Conference on Computational Complexity (CCC)*, pages 41–51.

[Hrubeš and Tzameret, 2012] Hrubeš, P. and Tzameret, I. (2012). Short proofs for the determinant identities. In *Proceedings of the 44th Annual ACM Symposium on the Theory of Computing (STOC)*, New York. ACM.

[Hyafil, 1979] Hyafil, L. (1979). On the parallel evaluation of multivariate polynomials. *SIAM J. Comput.*, 8(2):120–123.

[Impagliazzo et al., 1999] Impagliazzo, R., Pudlák, P., and Sgall, J. (1999). Lower bounds for the polynomial calculus and the gröbner basis algorithm. *Computational Complexity*, 8(2):127–144.

[Jeřábek, 2004] Jeřábek, E. (2004). Dual weak pigeonhole principle, Boolean complexity, and derandomization. *Ann. Pure Appl. Logic*, 129(1-3):1–37.

[Krajíček, 1995] Krajíček, J. (1995). *Bounded arithmetic, propositional logic, and complexity theory*, volume 60 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge.

[Krajíček, 2011] Krajíček, J. (2011). *Forcing with random variables and proof complexity*. London Mathematical Society Lecture Note Series, No.382. Cambridge University Press.

[Li and Tzameret, 2013] Li, F. and Tzameret, I. (2013). Generating matrix identities and proof complexity. *Electronic Colloquium on Computational Complexity, TR13-185*. arXiv:1312.6242 [cs.CC] http://arxiv.org/abs/1312.6242.

[Li et al., 2015] Li, F., Tzameret, I., and Wang, Z. (2015). Non-commutative formulas and Frege lower bounds: a new characterization of propositional proofs. In *Proceedings of the 30th Computational Complexity Conference (CCC), June 17-19, 2015*.

[Nisan, 1991] Nisan, N. (1991). Lower bounds for non-commutative computation. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing (STOC 1991)*, pages 410–418.

[Nordström, 2015] Nordström, J. (2015). On the interplay between proof complexity and sat solving. *ACM SIGLOG News*, 2(3):19–44.

[Pitassi, 1997] Pitassi, T. (1997). Algebraic propositional proof systems. In *Descriptive complexity and finite models (Princeton, NJ, 1996)*, volume 31 of *DIMACS Ser. Discrete Math. Theoret. Comput. Sci.*, pages 215–244. Amer. Math. Soc., Providence, RI.

[Raz, 2013] Raz, R. (2013). Tensor-rank and lower bounds for arithmetic formulas. *J. ACM*, 60(6):40.

[Raz and Shpilka, 2005a] Raz, R. and Shpilka, A. (2005a). Deterministic polynomial identity testing in non-commutative models. *Comput. Complex.*, 14(1):1–19. Preliminary version in the *19th Annual IEEE Conference on Computational Complexity (CCC 2004)*.

[Raz and Shpilka, 2005b] Raz, R. and Shpilka, A. (2005b). Deterministic polynomial identity testing in non commutative models. *Computational Complexity*, 14(1):1–19.

[Raz and Tzameret, 2008a] Raz, R. and Tzameret, I. (2008a). Resolution over linear equations and multilinear proofs. *Ann. Pure Appl. Logic*, 155(3):194–224.

[Raz and Tzameret, 2008b] Raz, R. and Tzameret, I. (2008b). The strength of multilinear proofs. *Computational Complexity*, 17(3):407–457.

[Razborov, 1998] Razborov, A. A. (1998). Lower bounds for the polynomial calculus. *Computational Complexity*, 7(4):291–324.

[Razborov, 2015] Razborov, A. A. (2015). Pseudorandom generators hard for k-DNF resolution and polynomial calculus resolution. *Annals of Mathematics*, 181:415–472.

[Reckhow, 1976] Reckhow, R. A. (1976). *On the lengths of proofs in the propositional calculus*. PhD thesis, University of Toronto.

[Schwartz, 1980] Schwartz, J. T. (1980). Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717. Preliminary version in the *International Symposium on Symbolic and Algebraic Computation (EUROSAM 1979)*.

[Shpilka and Yehudayoff, 2010] Shpilka, A. and Yehudayoff, A. (2010). Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5(3-4):207–388.

[Strassen, 1973] Strassen, V. (1973). Vermeidung von divisionen. *J. Reine Angew. Math.*, 264:182–202. (in German).

[Tzameret, 2011] Tzameret, I. (2011). Algebraic proofs over noncommutative formulas. *Information and Computation*, 209(10):1269–1292.

[Valiant, 1979a] Valiant, L. G. (1979a). Completeness classes in algebra. In *Proceedings of the 11th Annual ACM Symposium on the Theory of Computing*, pages 249–261. ACM.

[Valiant, 1979b] Valiant, L. G. (1979b). The complexity of computing the permanent. *Theor. Comput. Sci.*, 8:189–201.

[Valiant, 1982] Valiant, L. G. (1982). Reducibility by algebraic projections. *Logic and Algorithmic: International Symposium in honour of Ernst Specker*, 30:365–380.

[Valiant et al., 1983] Valiant, L. G., Skyum, S., Berkowitz, S., and Rackoff, C. (1983). Fast parallel computation of polynomials using few processors. *SIAM J. Comput.*, 12(4):641–644.

[Zippel, 1979] Zippel, R. (1979). Probabilistic algorithms for sparse polynomials. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation (EUROSAM 1979)*, pages 216–226. Springer-Verlag.