

Characterizing Propositional Proofs as Non-Commutative Formulas*

Fu Li[†]

Iddo Tzameret[‡]

Zhengyu Wang[§]

Abstract

Does every Boolean tautology have a short propositional-calculus proof? Here, a propositional-calculus (i.e., Frege) proof is any proof starting from a set of axioms and deriving new Boolean formulas using a fixed set of sound derivation rules. Establishing any super-polynomial size lower bound on Frege proofs (in terms of the size of the formula proved) is a major open problem in proof complexity, and among a handful of fundamental hardness questions in complexity theory. Non-commutative algebraic formulas, on the other hand, constitute a quite weak computational model, for which exponential-size lower bounds were shown already back in 1991 by Nisan [STOC 1991], using a particularly transparent argument.

In this work we show that Frege lower bounds in fact follow from size lower bounds on non-commutative formulas computing certain families of polynomials (and that such lower bounds on non-commutative formulas must exist, unless $\mathbf{NP}=\mathbf{coNP}$). More precisely, we demonstrate a natural association between tautologies T to families of non-commutative polynomials P , such that:

if T has a polynomial-size Frege proof then some non-commutative polynomial in P can be computed by a polynomial-size non-commutative algebraic formula; and conversely, when T is a DNF, if some polynomial in P has a polynomial-size non-commutative algebraic formula over $GF(2)$ then T has a Frege proof of quasi-polynomial size.

The argument is a characterization of Frege proofs as non-commutative formulas: we show that the Frege system is (quasi-) polynomially equivalent to a *non-commutative Ideal Proof System* (IPS), following the recent work of Grochow and Pitassi [FOCS 2014] that introduced a propositional proof system in which proofs are algebraic circuits, and the work in [Tza11] that considered adding the commutator as an axiom in algebraic propositional proof systems. This also gives a characterization of propositional Frege proofs in terms of (non-commutative) algebraic formulas that is tighter than (the formula version of IPS) in Grochow and Pitassi [FOCS 2014].

1 Introduction

1.1 Propositional Proof Complexity

The field of propositional proof complexity aims to understand and analyze the computational resources required to prove propositional statements. The problems the field poses are fundamental, difficult, and of central importance to computer science and complexity theory as demonstrated by the seminal work of Cook and Reckhow [CR79], who showed the immediate relevance of these problems to the \mathbf{NP} vs. \mathbf{coNP} problem (and thus to the \mathbf{P} vs. \mathbf{NP} problem).

*An extended abstract of this work entitled “Non-commutative Formulas and Frege Lower Bounds: a New Characterization of Propositional Proofs” appeared in *Proceedings of the 30th Annual Computational Complexity Conference (CCC)*: June 17-19, 2015.

[†]The University of Texas at Austin, Department of Computer Science. Email: fuli.theory.research@gmail.com

[‡]Royal Holloway, University of London, Department of Computer Science. Supported in part by NSFC grant 61373002. Email: Iddo.Tzameret@rhul.ac.uk

[§]Harvard University, Department of Computer Science. Email: zhengyuwang@g.harvard.edu

Among the major unsolved questions in propositional proof complexity, is whether the standard propositional logic calculus, either in the form of the Sequent Calculus, or equivalently, in the axiomatic form of Hilbert style proofs (i.e., Frege proofs), is polynomially bounded; that is, whether every propositional tautology—namely, a formula that is satisfied by every assignment—has a proof whose size is polynomially bounded in the size of the formula proved (alternatively and equivalently, we can think of unsatisfiable formulas and their refutations). Here, we consider the size of proofs as the number of symbols it takes to write them down, where each formula in the proof is written as a Boolean *formula* (in other words we count the total number of logical gates appearing in the proof).

It is known since Reckhow’s work [Rec76a] that all Frege proof-systems¹ (as well as the Gentzen sequent calculus with the cut rule [Gen35]) are polynomially equivalent to each other, and hence it does not matter precisely which rules, axioms, and logical-connectives we use in the system. Nevertheless, for 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 $A \rightarrow B$ is an abbreviation of $\neg A \vee B$; and A, B, C are any propositional formulas):

$$\begin{aligned} & A \rightarrow (B \rightarrow A) \\ & (\neg A \rightarrow \neg B) \rightarrow ((\neg A \rightarrow B) \rightarrow A) \\ & (A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C)), \end{aligned}$$

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

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

Complexity-wise, Frege is considered a very strong proof system but alas a poorly understood one. The qualification *strong* here has several meanings: first, that no super-polynomial lower bound is known for Frege proofs. Second, that there are not even good hard candidates for the Frege system (see [BBP95, Raz15, Kra11, LT13, ABB16] for further discussions on hard proof complexity candidates). Third, that for most hard instances (e.g., the pigeonhole principle and Tseitin tautologies) that are known to be hard for weaker systems (e.g., resolution, cutting planes, etc.), there *are* known polynomial bounds on Frege proofs. Fourth, that proving super-polynomial lower bounds on Frege proofs seems to a certain extent out of reach of current techniques (and believed by some to be even *harder* than proving explicit circuit lower bounds [Raz15]). And finally, that by the common (mainly informal) correspondence between circuits and proofs—namely, the correspondence between a circuit-class \mathcal{C} and a proof system in which every proof-line is written as a circuit² from \mathcal{C} —Frege system corresponds to the circuit class of polynomial-size $\log(n)$ -depth circuits denoted \mathbf{NC}^1 (equivalently, of polynomial-size formulas [Spi71]), considered to be a strong computational model for which no (explicit) super-polynomial lower bounds are currently known.

Accordingly, proving lower bounds on Frege proofs is considered an extremely hard task. In fact, the best lower bound known today is only quadratic, which uses a fairly simple syntactic argument [Kra95]. If we put further restrictions on Frege proofs, like restricting the depth of each formula appearing in a proof to a certain fixed constant, exponential lower bounds can be obtained [Ajt88, PBI93, PBI93]. Although these constant-depth Frege exponential-size lower bounds go back to Ajtai’s result from 1988, they are still in some sense the state-of-the-art in proof complexity lower bounds (beyond the important developments on weaker proof systems, such as resolution and its comparatively weak extensions). Constant-depth Frege lower bounds use quite involved probabilistic arguments,

¹Formally, a Frege proof system is any propositional proof system with a fixed number of axiom schemes and sound derivation rules that is also implicationally complete, and in which proof-lines are written as propositional formulas (see Definition 2.4).

²To be more precise, for a circuit class \mathcal{C} , we say that a *family* of Frege proofs are \mathcal{C} -Frege proofs if every proof-line in the family is a circuit from \mathcal{C} (e.g., has circuits with constant depth, for some global constant).

mainly specialized switching lemmas tailored for specific tautologies (namely, counting tautologies, most notable of which are the Pigeonhole Principle tautologies). Even random k CNF formulas near the satisfiability threshold are not known to be hard for constant-depth Frege (let alone hard for [unrestricted depth] Frege).

All of the above goes to emphasize the importance, basic nature and difficulty in understanding the complexity of strong propositional proof systems, while showing how little is actually known about these systems.

1.2 Prominent Directions for Understanding Propositional Proofs

As we already mentioned, there is a guiding line in proof complexity which states a correspondence between the complexity of circuits and the complexity of proofs. This correspondence is mainly informal, but there are seemingly good indications showing it might be more than a superficial analogy. One of the most compelling evidence for this correspondence is that there is a formal correspondence (cf. [CN10] for a clean formulation of this) between the first-order logical theories of bounded arithmetic (whose axioms state the existence of sets taken from a given complexity class \mathcal{C}) to propositional proof systems (in which proof-lines are circuits from \mathcal{C}).

Another aspect of the informal correspondence between circuit and proof complexity is that circuit hardness sometimes can be used to obtain proof complexity hardness. The most notable example of this is the lower bounds on constant-depth Frege proofs mentioned above: constant-depth Frege proofs can be viewed as propositional calculus operating with \mathbf{AC}^0 circuits, and the known lower bounds on constant depth Frege proofs (cf. [Ajt88, KPW95, PBI93]) use techniques borrowed from \mathbf{AC}^0 circuits lower bounds (i.e., random restrictions and switching lemmas as in [Has87]).

The success in moving from circuit hardness towards proof-complexity hardness has spurred a flow of attempts to obtain lower bounds on proof systems other than constant depth Frege. Most of these attempts followed the following rational (successfully exemplified in the case of \mathbf{AC}^0 -Frege lower bounds): assuming a circuit class \mathcal{C} has a lower bound, consider a proof system in which proof-lines are taken from the class \mathcal{C} and attempt to accommodate the lower bound technique against \mathcal{C} to a lower bound technique against the corresponding proof system.

For example, Beame et al. [BIK⁺96a], Buss *et al.* [BIK⁺97] and others such as Maciel and Pitassi [MP97] considered constant depth Frege proofs with modulo q gates, motivated by known lower bounds on $\mathbf{AC}^0[q]$ circuits [Smo87, Raz87a]. Pudlák [Pud99] and Atserias et al. [AGP02] studied proofs based on monotone (circuits, motivated by known exponential lower bounds on monotone circuits [Raz85]). Raz and Tzameret [RT08b, RT08a, Tza08] investigated algebraic proof systems operating with multilinear formulas, motivated by lower bounds on multilinear formulas for the determinant, permanent and other explicit polynomials [Raz09, Raz06]. Atserias et al. [AKV04], Krajíček [Kra08] and Segerlind [Seg07] have considered proofs operating with ordered binary decision diagrams (OBDDs), and the second author [Tza11] initiated the study of proofs operating with non-commutative formulas (see Sec. 1.4 for a comparison with the current work).³ It is worth noting that except for the case of OBDD-based proof systems, none of the above attempts has yet to lead to the desired proof-complexity lower bound.

Another way to reduce proof-complexity lower bounds to hardness results in circuit complexity comes from the feasible interpolation property (cf. [Kra97]). In this approach, short proofs of certain tautologies lead to small circuits for their corresponding interpolants; hence, lower bounds on the interpolants imply proof-size lower bounds.

The aforementioned are attempts, some of which successful, to reduce proof-complexity lower bounds to circuits complexity lower bounds. Until quite recently it was unknown whether the converse

³We do not discuss here the important thread of results whose aim is to establish conditional lower bounds based on Nisan-Wigderson generators. This direction was developed in e.g. [ABRW04, Raz15, Kra04, Kra11].

direction is possible, namely, whether proof complexity hardness (of concrete known proof systems) implies any computational hardness. An initial example of such an implication from proof hardness to circuit hardness was given by Raz and Tzameret [RT08b]. They showed that a separation between algebraic proof systems operating with algebraic circuits and multilinear algebraic circuits, resp., for an explicit family of polynomials, implies a separation between algebraic circuits and multilinear algebraic circuits.

In a recent significant development about the complexity of strong proof systems, Grochow and Pitassi [GP14] demonstrated a much stronger correspondence. They introduced a natural propositional proof system, called the *Ideal Proof System* (IPS for short), for which *any* super-polynomial size lower bound on IPS implies a corresponding size lower bound on algebraic circuits, and formally, that the permanent does not have polynomial-size algebraic circuits. The IPS is defined as follows⁴:

Definition 1.1 (Ideal Proof System (IPS); Grochow-Pitassi [GP14]). *Let $F_1(\bar{x}), \dots, F_m(\bar{x})$ be a system of polynomials in the variables x_1, \dots, x_n , where the polynomials $x_i^2 - x_i$, for all $1 \leq i \leq n$, are part of this system. An IPS refutation (or certificate) that the F_i 's polynomials have no common 0-1 solutions is a polynomial $\mathfrak{F}(\bar{x}, \bar{y})$ in the variables x_1, \dots, x_n and y_1, \dots, y_m , such that:*

1. $\mathfrak{F}(x_1, \dots, x_n, \bar{0}) = 0$; and
2. $\mathfrak{F}(x_1, \dots, x_n, F_1(\bar{x}), \dots, F_m(\bar{x})) = 1$.

The essence of IPS is that a proof (or refutation) is a *single* polynomial that can be written simply as an algebraic *circuit* or *formula*. The advantage of this formulation is that now we can obtain direct connections between circuit/formula hardness (i.e., “computational hardness”) and hardness of proofs. Grochow and Pitassi showed indeed that a lower bound on IPS written as an algebraic circuit implies that the permanent does not have polynomial-size algebraic circuits (Valiant’s conjectured separation $VNP \neq VP$ [Val79]); And similarly, a lower bound on IPS written as an algebraic *formula* implies that the permanent does not have polynomial-size algebraic formulas ($VNP \neq VP_e$, *ibid*).

Under certain assumptions, Grochow and Pitassi [GP14] were able to connect their result to standard propositional-calculus proof systems, i.e., Frege and Extended Frege. Their assumption was the following: *Frege has polynomial-size proofs of the statement expressing that the PIT for algebraic formulas is decidable by polynomial-size Boolean circuits* (PIT for algebraic formulas is the problem of deciding whether an input algebraic formula computes the [formal] zero polynomial). They showed that⁵, under this assumption super-polynomial lower bounds on Frege proofs imply that the permanent does not have polynomial-size algebraic circuits. This, in turn, can be considered as a (conditional) justification for the apparent long-standing difficulty of proving lower bounds on strong proof systems.

1.3 Overview of Results and Proofs

1.3.1 Sketch

In this work we give a new characterization of the propositional calculus in terms of non-commutative formulas. We formulate a very natural proof system, that is, a non-commutative variant of the ideal proof system, which we show captures *unconditionally* (up to a quasi-polynomial-size increase, and in

⁴Grochow-Pitassi considered IPS also without the Boolean axioms $x_i^2 - x_i$ (for all $i \in [n]$), leading to a refutation system for arbitrary unsatisfiable systems of polynomial identities.

⁵We focus only on the relevant results about Frege proofs from [GP14] (and not the results about Extended Frege in [GP14]; the latter proof system operates, essentially, with Boolean circuits, in the same way that Frege operates with Boolean formulas (equivalently NC^1 circuits)).

some cases only a polynomial-size increase⁶) propositional Frege proofs. A proof in the non-commutative IPS is simply a *single non-commutative polynomial* written as a non-commutative formula.

This provides a new relation between circuit complexity hardness to proof complexity hardness. Specifically, our work shows that proving Frege lower bounds reduces to the task of proving *matrix rank lower bounds* on the matrices associated with certain families of non-commutative polynomials (in the sense of Nisan [Nis91]; see Section 1.3.2 below for details). Matrix rank lower bounds apparently are better understood than computational and proof complexity lower bounds and so this reduction should hopefully shed new light on the complexity of Frege proofs.

The new characterization also tightens the recent results of Grochow and Pitassi [GP14] in the following sense:

- (i) The non-commutative IPS is *polynomial-time checkable*—whereas the original IPS was checkable in probabilistic polynomial-time; and
- (ii) Frege proofs *unconditionally* quasi-polynomially simulate the non-commutative IPS—whereas Frege was shown to efficiently simulate IPS only assuming that the decidability of PIT for (commutative) algebraic formulas by polynomial-size circuits is efficiently provable in Frege.

1.3.2 Some Preliminaries: Non-Commutative Polynomials and Formulas

A *non-commutative polynomial* over a given field \mathbb{F} and with the variables $X := \{x_1, x_2, \dots\}$ is a formal sum of monomials (i.e., product of variables) with coefficients from \mathbb{F} such that the product of variables is non-commuting. For example, $x_1x_2 - x_2x_1 + x_3x_2x_3^2 - x_2x_3^3$, $x_1x_2 - x_2x_1$ and 0 are three distinct polynomials in $\mathbb{F}\langle X \rangle$. The ring of non-commutative polynomials with variables X and coefficients from \mathbb{F} is denoted $\mathbb{F}\langle X \rangle$.

A *polynomial* (i.e., a *commutative* polynomial) over a field is defined in the same way as a non-commutative polynomial except that the product of variables *is* commutative; in other words, it is a sum of (commutative) monomials. A *homogenous* (commutative or non-commutative) polynomial is a polynomial with all monomials having the same (total) degree.

A ***non-commutative algebraic formula*** (non-commutative formula for short) is a fan-in two labeled tree, with edges directed from leaves towards the root, such that the leaves are labeled with field elements (for a given field \mathbb{F}) or variables x_1, \dots, x_n , and internal nodes (including the root) are labeled with a plus $+$ or product \times gates. A product gate has an order on its two children (holding the order of non-commutative product). A non-commutative formula computes a non-commutative polynomial in the natural way (see Definition 2.5).

Exponential-size lower bounds on non-commutative formulas (over any field) were established by Nisan [Nis91]. The idea (in retrospect) is quite simple: first transform a non-commutative formula into an algebraic branching program (ABP; Definition 4.11); and then show that the number of nodes in the i th layer of an ABP computing a degree d homogenous non-commutative polynomial f is bounded from below by the rank of the degree i -partial-derivative matrix of f .⁷ Thus, lower bounds on non-commutative formulas follow from quite immediate rank arguments (e.g., the partial derivative matrices associated with the permanent and determinant can easily be shown to have high ranks).

⁶The non-commutative IPS *polynomially* simulates Frege; and conversely, the simulation of the non-commutative IPS by Frege depends on the degree of the non-commutative IPS refutation; e.g., the simulation is *polynomial* when refutations are of logarithmic degrees (see note after Theorem 1.7).

⁷The degree i partial derivative matrix of f is the matrix whose rows are all non-commutative monomials of degree i and columns are all non-commutative monomials of degree $d - i$, such that the entry in row M and column N is the coefficient of the d degree monomial $M \cdot N$ in f .

1.3.3 Non-Commutative Ideal Proof System

Recall the IPS refutation system from Definition 1.1 above. We use the idea introduced in [Tza11], which considered adding the commutator $x_1x_2 - x_2x_1$ as an axiom in propositional algebraic proof systems, to define a refutation system that simulates Frege:

Definition 1.2 (Non-commutative IPS). *Let \mathbb{F} be a field. Assume that $F_1(\bar{x}) = F_2(\bar{x}) = \dots = F_m(\bar{x}) = 0$ is a system of non-commutative polynomial equations from $\mathbb{F}\langle x_1, \dots, x_n \rangle$, and suppose that the following set of equations (axioms) are included in the $F_i(\bar{x})$'s:*

Boolean axioms: $x_i \cdot (1 - x_i)$, for all $1 \leq i \leq n$;

Commutator axioms: $x_i \cdot x_j - x_j \cdot x_i$, for all $1 \leq i < j \leq n$.

A **non-commutative IPS proof** of the non-commutative polynomial $f(\bar{x})$ from the assumptions $F_i(\bar{x})$'s is a non-commutative polynomial $\mathfrak{F}(\bar{x}, \bar{y}) \in \mathbb{F}\langle \bar{x}, \bar{y} \rangle$ in the variables x_1, \dots, x_n and y_1, \dots, y_m , such that:

1. $\mathfrak{F}(x_1, \dots, x_n, \bar{0}) = 0$; and
2. $\mathfrak{F}(x_1, \dots, x_n, F_1(\bar{x}), \dots, F_m(\bar{x})) = f(\bar{x})$.

A **non-commutative IPS refutation** (or certificate) that the system of $F_i(\bar{x})$'s is unsatisfiable is a non-commutative IPS proof of the polynomial 1 from the assumptions $F_i(\bar{x}) = 0$ (for $i \in [m]$). We define the **size** of a non-commutative IPS proof to be the minimal size of a non-commutative formula computing the non-commutative IPS proof $\mathfrak{F}(\bar{x}, \bar{y})$.

Comment: (i) We prove that the non-commutative IPS is a sound and complete proof system for propositional formulas (Corollary 1.5). One can check that the $F_i(\bar{x})$'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. Thus, we can also show that non-commutative IPS is a sound and *implicationally* complete proof system for non-commutative polynomials, in the following sense: there is a non-commutative IPS proof of f from assumptions $F_1 = \dots = F_m = 0$ (that include the Boolean and commutator axioms) iff every 0-1 assignment that satisfies all the assumptions also satisfies $f = 0$.

(ii) It is important to note that identities 1 and 2 in Definition 1.2 are *formal* identities between non-commutative polynomials. It is possible to show that without the commutator axioms the system becomes *incomplete* in the sense that there will be unsatisfiable systems of non-commutative polynomials $F_1(\bar{x}) = F_2(\bar{x}) = \dots = F_m(\bar{x}) = 0$ (where the F_i 's include the Boolean and commutator axioms) for which there are *no* non-commutative IPS refutations.

(iii) In order to prove that a system of *commutative* polynomial equations $\{P_i = 0\}$ (where each P_i is expressed as an algebraic formula) has no common roots in non-commutative IPS, we write each P_i as a *non-commutative formula* (in some way; note that there is no unique way to do this).

The main result of this paper is that the non-commutative IPS (over either \mathbb{Q} or \mathbb{Z}_q , for any prime q) simulates Frege; and conversely, Frege quasi-polynomially simulates the non-commutative IPS (over \mathbb{Z}_2). We explain the results in what follows.

Non-Commutative IPS Simulates Frege

For the purpose of the next theorem we use a standard translation of propositional formulas T into non-commutative algebraic formulas:

Definition 1.3 ($\text{tr}(f)$). *Let $\text{tr}(x_i) := x_i$, for variables x_i ; $\text{tr}(\text{false}) := 1$; $\text{tr}(\text{true}) := 0$; and by induction on the size of the propositional formula: $\text{tr}(\neg T) := 1 - \text{tr}(T)$; $\text{tr}(T_1 \vee T_2) = \text{tr}(T_1) \cdot \text{tr}(T_2)$ and finally $\text{tr}(T_1 \wedge T_2) = 1 - (1 - \text{tr}(T_1)) \cdot (1 - \text{tr}(T_2))$.*

For a non-commutative formula f denote by \widehat{f} the non-commutative polynomial computed by f . Thus, T is a propositional tautology iff $\widehat{\text{tr}(T)} = 0$ for every 0-1 assignment to the variables of the non-commutative polynomial.

Theorem 1.4 (First main theorem). *Non-commutative IPS simulates Frege. More precisely, for every propositional tautology T , if T has a size s Frege proof, then there is a non-commutative IPS proof of $\text{tr}(\neg T)$ over the field of rationals or \mathbb{Z}_q , for a prime q , with $\text{poly}(s)$ non-commutative formula size.*

The fact that an algebraic formula (or circuit) in the form of the IPS can simulate a propositional Frege proof was shown by Pitassi [Pit97] (under a different name than IPS; cf. [GP14]). The non-commutative IPS, on the other hand, is much more restrictive than the original (commutative) IPS: instead of using commutative polynomials (written as algebraic formulas) we now use non-commutative polynomials (written as non-commutative algebraic formulas). And as mentioned above, in order to maintain the completeness of the non-commutative IPS we must add the commutator axioms $x_i x_j - x_j x_i$ to the system. Thus, the question arises: how can we still simulate Frege in this restrictive framework? The answer to this, which also constitutes one of the main observations of the simulation, is that *the commutator axioms are already used implicitly in propositional Frege proofs*: every classical propositional calculus system has some (possibly implicit) structural rules that enable one to commute AND's and OR's (e.g., $A \wedge B$ is not the same formula as $B \wedge A$, from the perspective of the propositional calculus). In other words, Frege proofs operate with formulas as purely syntactic terms, and thus commutativity of AND and OR are not free for Frege proofs.

Similar to [GP14] as well as [GH03], the proof of Theorem 1.4 is basically a step-by-step simulation of a given Frege proof, where the underlying tree of the non-commutative formula pertaining to the IPS proof roughly corresponds to the underlying Frege proof tree (Krajíček [Kra95] showed how to convert a Frege proof into a tree-like proof with only a polynomial-size increase).

Raz and Shpilka [RS05], devised a polynomial-time deterministic Polynomial Identity Testing (PIT) algorithm for non-commutative formulas (see Theorem 3.2). As a corollary of Raz-Shpilka PIT algorithm, we can check in deterministic polynomial-time the correctness of non-commutative IPS refutations:

Corollary 1.5. *The non-commutative IPS is a sound and complete refutation system in the sense of Cook-Reckhow [CR79]. That is, it is a sound and complete refutation system for unsatisfiable propositional formulas (written as non-commutative formulas according to Definition 1.3) in which refutations can be checked for correctness in deterministic polynomial-time.*

This should be contrasted with the original (commutative) IPS of [GP14], for which verification of refutations is done in probabilistic polynomial time using the standard Schwartz-Zippel [Sch80, Zip79] PIT algorithm.

A consequence of Theorem 1.4 is that to prove a super-polynomial Frege lower bound it suffices to prove a super-polynomial lower bound on non-commutative formulas computing certain families of polynomials. More precisely, this family of non-commutative polynomials is the family of non-commutative IPS certificates $\mathfrak{F}(\bar{x}, \bar{y})$ (for a given unsatisfiable instance). Consequently, it suffices to show that any such $\mathfrak{F}(\bar{x}, \bar{y})$ in this family must have a super-polynomial total rank according to the associated partial-derivatives matrices in the sense of Nisan [Nis91], as discussed before.

Frege Simulates Non-Commutative IPS

We shall prove that Frege simulates the non-commutative IPS for CNFs (this is the case considered in [GP14]), over $GF(2)$, and with only a quasi-polynomial increase in size (and for some specific cases the simulation can become polynomial). It will be convenient to use a translation of clauses to non-commutative formulas which is slightly different than Definition 1.3:

Definition 1.6 ($\text{tr}'(f)$ and Q_i^ϕ). Given a Boolean formula f we define its non-commutative formula translation $\text{tr}'(f)$ as follows. Let $\text{tr}'(x) := 1 - x$ and $\text{tr}'(\neg x) := x$, for x a variable. Let $\text{tr}'(\text{false}) := 0$; $\text{tr}'(\text{true}) := 1$; and $\text{tr}'(f_1 \vee \dots \vee f_r) := \text{tr}'(f_1) \cdots \text{tr}'(f_r)$ (where the sequence of products stands for a (balanced) fan-in two tree of product gates with $\text{tr}'(f_i)$ on the leaves). Further, for a CNF $\phi = \kappa_1 \wedge \dots \wedge \kappa_m$, denote by Q_i^ϕ the non-commutative formula translation $\text{tr}'(\kappa_i)$ of the clause κ_i .

Note that this way, the system of equations $Q_1^\phi = 0, \dots, Q_m^\phi = 0$ is unsatisfiable iff ϕ is unsatisfiable.

Theorem 1.7 (Second main theorem). Let $\phi = \kappa_1 \wedge \dots \wedge \kappa_m$ be a CNF and let $Q_1^\phi, \dots, Q_m^\phi$ denote the corresponding non-commutative formulas for the clauses of ϕ . If there is a non-commutative IPS refutation of size s of $Q_1^\phi, \dots, Q_m^\phi$ over $\mathbf{GF}(2)$, then there is a Frege proof of size $s^{O(\log s)}$ of the tautology $\neg\phi$.

Note:

(i) The proof of Theorem 1.7 achieves in fact a slightly stronger simulation than stated. That is, our simulation shows that if the degree of the non-commutative IPS refutation is r and its formula depth is d , then there is a Frege proof of $\neg\phi$ with size $\text{poly}\left(\binom{d+r+1}{r} \cdot s\right)$. And in particular, Frege *polynomially* simulates non-commutative IPS refutations of $O(\log n)$ degrees (for n the number of variables in the CNF). However, for simplicity we shall always assume that the depth d of the non-commutative IPS formula is logarithmic in its size (Observation 4.1 shows that we can always balance non-commutative formulas), and so explicitly we only deal with the case where $d = O(\log s)$ and $r = O(s)$.

(ii) The simulation of the non-commutative IPS by Frege is shown in this paper only over $\mathbf{GF}(2)$. It is however highly plausible that the same simulation holds also for any fixed finite field (note that Raz-Shpilka [RS05] PIT algorithm works over any field). One then needs to work out in Frege how to efficiently encode, add and multiply field elements; but these encoding and proofs tend to be nontrivial to work out in detail (something along these lines, for integer arithmetic modulo m , was shown in [BPR00]). Furthermore, it is not unreasonable that the case of the rationals could also be dealt with, though possibly incurring more technical complications.

The proof of Theorem 1.7 consists of several separate steps of independent interest. From the logical point of view, the argument is a short Frege proof of a *reflection principle* for the non-commutative IPS system. A reflection principle for a given proof system P is a statement that says that if there exists a P -proof of a formula F then F is also *true*. The argument becomes rather complicated because we need to prove properties of the PIT algorithm for non-commutative formulas devised by Raz and Shpilka [RS05] within the restrictive framework of propositional Frege proofs.

Our goal is then to prove $\neg\phi$ in Frege, given a non-commutative IPS refutation π of ϕ .

Step 1: balancing. We first *balance* the non-commutative IPS π , so that its depth is logarithmic in its size. We observe that the recent construction of Hrubeš and Wigderson [HW14] for balancing non-commutative formulas with division gates (incurring at most a polynomial increase in size) results in a *division-free* formula, when the *initial* non-commutative formula is division-free by itself. Therefore, we can assume that the non-commutative IPS certificate is already balanced (this step is independent of the Frege system).

Step 2: Booleanization. We then consider our balanced π , which is a non-commutative polynomial identity over $\mathbf{GF}(2)$, as a *Boolean tautology*, by replacing plus gates with XORs and product gates with ANDs.

Step 3: reflection principle. We give a short Frege proof of the following reflection principle for non-commutative IPS: “($[\pi]$ is a non-commutative IPS refutation of $[\phi]$) \longrightarrow $\neg\phi$ ”, where $[\phi]$ and $[\pi]$ are some reasonable encodings of the unsatisfiable CNF ϕ and its refutation π , respectively. This then

reduces the task of efficiently proving $\neg\phi$ in Frege to the task of showing that any non-commutative formula identity over $GF(2)$, considered as a Boolean tautology, has a short Frege proof, because the premise of the reflection principle’s implication can be considered as such a Boolean tautology. Using a reflection principle in proof complexity has a long tradition (cf. [CN10]), and for the IPS it was used in [GP14].

Step 4: homogenization. This is the *only* step that is responsible for the *quasi-polynomial size increase* in Theorem 1.7. More precisely, this increase in size depends on the fact that for the purpose of establishing short Frege proofs for all non-commutative polynomial identities over $GF(2)$ (considered as Boolean tautological formulas) it is important that the formulas are written as a sum of *homogenous* non-commutative formulas.

Note that it is not known whether algebraic formulas can be turned into a (sum of) homogenous formulas with only a polynomial increase in size (in contrast to the standard efficient homogenization of algebraic *circuits* by Strassen [Str73] that does allow such a conversion). Nevertheless, Strassen’s standard procedure enables us to transform any polynomial-size algebraic formula into a sum of homogenous formulas with only a *quasi-polynomial* increase in size: any formula of size $\text{poly}(n)$ computing a polynomial f (and thus the degree of f is also polynomial) can be transformed into a sum of homogenous formulas, each having size $n^{O(\log n)}$ and computes the corresponding homogenous part of f . (One can show that the same also holds for *non-commutative* formulas.)

For the purpose of establishing a quasi-polynomial simulation of non-commutative IPS by Frege, it is sufficient to use the original Strassen’s homogenization procedure (as simulated inside Frege; cf. [HT15]). However, as the note after Theorem 1.7 indicates, we show a slightly stronger simulation result, using an efficient Frege simulation of a recent result due to Raz [Raz13] who showed how to transform an algebraic formula into (a sum of) homogenous formulas in a manner which is more efficient than Strassen [Str73]. Specifically, in Lemma 4.6 we show that:

1. The same construction in [Raz13] also holds for *non-commutative* formulas;
2. This construction for non-commutative formulas can be carried out efficiently inside Frege. That is, if F is a non-commutative formula of size s and depth d computing a homogenous non-commutative polynomial over $GF(2)$ of degree r , then there exists a syntactic homogenous non-commutative formula F' computing the same polynomial and with size $O\left(\binom{r+d+1}{r} \cdot s\right)$, such that Frege admits a proof of $F \leftrightarrow F'$ of size polynomial (in $|F'|$).

Step 5: short proofs for homogenous non-commutative identities. Now that we have reduced our task to the task of showing that every non-commutative formula identity over $GF(2)$ (considered as a tautology) has a short Frege proof; and we have also agreed to first turn (inside Frege) our non-commutative identities into *homogenous* formulas (incurring up to a quasi-polynomial increase in the formulas size)—it remains to show how to prove efficiently in Frege homogenous non-commutative identities. (Formally, we shall in fact deal with *syntactic* homogenous formulas.)

To this end we essentially construct an efficient Frege proof of the correctness of the Raz and Shpilka PIT algorithm for non-commutative formulas [RS05]. This PIT algorithm uses some basic linear algebraic concepts that might be beyond the efficient-reasoning strength of Frege. However, since we only need to show the *existence* of short Frege proofs for the PIT algorithm’s correctness, we can supply *witnesses* to witness the desired linear algebraic objects needed in the proof (these witnesses will be a sequence of linear transformations).

A bigger obstacle is that it seems impossible to reason directly inside Frege about the algorithm of [RS05], since this algorithm first converts a non-commutative formula into an *algebraic branching program* (ABP); but the evaluation of ABPs (apparently) cannot be done with Boolean formulas (and accordingly Frege (apparently) cannot reason about the evaluation of ABPs). The reason for this

apparent inability of Frege to reason efficiently about ABP’s evaluation is that an ABP is a slightly more “sequential” object than a formula: an evaluation of an ABP with d layers can be done by an iterative matrix multiplication of d matrices—known to be doable with quasi-polynomial size formulas (or polynomial-size circuits with $O(\log^2 n)$ depth)—while Frege is a system that operates with formulas. To overcome this obstacle we show how to perform Raz and Shpilka’s PIT algorithm *directly on non-commutative formulas*, without converting the formulas first into ABPs. This technical contribution takes quite a large part of the argument (Sec. 4.6).

We are finally able to prove the following statement, which might be of independent interest:

Theorem 1.8. *There exists a constant c such that for any non-commutative formula $F(\bar{x})$ over $\mathbf{GF}(2)$ of size s that is identically zero, the corresponding Boolean tautology $\neg F_{\text{bool}}(\bar{x})$ (where F_{bool} results by replacing $+$ with XOR and \cdot with AND in $F(\bar{x})$) has a Frege proof of size at most s^c , for sufficiently large s .*

A more detailed *overview* of the proof (specifically, of the proof of Theorem 1.8) appears in Section 4.3.

1.4 Relation to Previous Work

Our main characterization of the Frege system is based on a non-commutative version of the IPS system from Grochow and Pitassi [GP14]. As described above, the non-commutative IPS gives a tighter characterization than the (commutative) IPS in [GP14], and close to capture the Frege system almost tightly.

In the original (formula version of the) IPS, proofs are algebraic formulas, and thus any super-polynomial lower bound on IPS refutations implies $\text{VNP} \neq \text{VP}_e$, or in other words, that the permanent does not have polynomial-size algebraic formulas (Joshua Grochow [personal communication]). This shows that proving IPS lower bounds will be considerably difficult to obtain. For the non-commutative IPS, on the other hand, we face a seemingly much favourable situation: an exponential-size lower bound on non-commutative IPS gives only a corresponding lower bound on non-commutative formulas, for which exponential-size lower bounds are already known [Nis91]. In other words, exponential-size lower bounds on Frege implies merely—at least in the context of the Ideal Proof System—corresponding lower bounds on non-commutative formulas, a result which is already known.

Let us also mention the work in [Tza11] that dealt with propositional proof systems over non-commutative formulas. In [Tza11] the choice was made to define all proof systems as polynomial calculus-style systems in which proof-lines are written as non-commutative formulas (as well as the more restricted class of ordered-formulas). This meant that the characterization of a proof system in terms of a *single* non-commutative polynomial is lacking from that work (as well as the consequences we obtained in the current work).

As for our use of reflection principle to (quasi-polynomially) simulate the non-commutative IPS by Frege: as mentioned before, using a reflection principle in proof complexity has a long tradition (cf. [CN10]), and for the IPS it was used in [GP14]. Grochow-Pitassi [GP14] introduced a collection of *PIT-axioms* for a given algebraic circuit class \mathcal{C} , for which short proofs in any system P implies that P simulates IPS refutations written as circuits from \mathcal{C} . Due to our proof of the reflection principle for non-commutative IPS, and specifically our Frege correctness proof of the Raz-Shpilka [RS05] non-commutative formulas PIT algorithm, it is very plausible that the Grochow-Pitassi PIT-axioms for non-commutative formulas admit short Frege proofs (though the details should be worked out carefully to verify this).

2 Preliminaries

For a positive natural number n we use the standard notation $[n]$ for $\{1, \dots, n\}$.

Definition 2.1 (Boolean formulas). *Given a set of input variables $\{x_1, x_2, \dots\}$ a Boolean formula on the input variables is a rooted finite tree of fan-in at most 2, with edges directed from leaves to the root. We consider the edges coming into nodes as ordered.⁸ Internal nodes are labeled with the Boolean gates OR, AND and NOT, denoted \vee, \wedge, \neg , respectively, where the fan-in of \vee and \wedge is two and the fan-in of \neg is one. The leaves are labeled either with input variables or with 0, 1 (identified with the truth values false and true, resp.). The entire formula computes the function computed by the gate at the root. Given a formula F , the **size** of the formula is the number of Boolean gates in F , denoted $|F|$.*

Given a pair of Boolean formulas A and B over the variables x_1, \dots, x_n , we denote by $A[B/x_i]$ the formula A in which every occurrence of x_i in A is substituted by the formula B . We use the symbol \equiv to denote logical equivalence and we use the symbol $A \leftrightarrow B$ to denote $(A \rightarrow B) \wedge (B \rightarrow A)$.

2.1 The Frege Proof System

As outlined in the introduction, a Frege proof system is any standard propositional proof system for proving propositional tautologies having finitely many axiom schemes and deduction rules, and where proof-lines are written as Boolean formulas.

Definition 2.2 (Frege (derivation) rule). *A Frege rule is a sequence of propositional formulas $A_0(\bar{x}), \dots, A_k(\bar{x})$, for $k \geq 0$, written as $\frac{A_1(\bar{x}), \dots, A_k(\bar{x})}{A_0(\bar{x})}$. In case $k = 0$, the Frege rule is called an axiom scheme. A formula F_0 is said to be derived by the rule from F_1, \dots, F_k if F_0, \dots, F_k are all substitution instances of A_1, \dots, A_k , for some assignment to the \bar{x} variables (that is, there are formulas B_1, \dots, B_n such that $F_i = A_i[B_1/x_1, \dots, B_n/x_n]$, for all $i = 0, \dots, k$). The Frege rule is said to be sound if whenever an assignment satisfies the formulas A_1, \dots, A_k above the line, then it also satisfies the formula A_0 below the line.*

Definition 2.3 (Frege proof). *Given a set of Frege rules, a Frege proof is a sequence of Boolean formulas such that every formula is either an axiom or was derived by one of the given Frege rules from previous formulas. If the sequence terminates with the Boolean formula A , then the proof is said to be a proof of A . The **size** of a Frege proof is the sum of all formula sizes in the proof.*

A proof system is *sound* if it admits proofs of only tautologies. A proof system is said to be *implicationally complete* if for all set of formulas S , if S semantically implies F , then there is a proof of F using (possibly) axioms from S .

Definition 2.4 (Frege proof system). *Given a set P of sound Frege rules, we say that P is a Frege proof system if P is implicationally complete.*

Note that a Frege proof is always sound since the Frege rules are assumed to be sound. Frege is also complete (that is, can prove all tautologies), by implicational completeness. We do not need to work with a specific Frege proof system, since a basic result in proof complexity by Reckhow [Rec76a] states that every two Frege proof systems, *even with different propositional connectives*, are polynomially equivalent. For concreteness the reader can think of Schoenfield's system from the introduction, making sure it is indeed a Frege system.

The problem of demonstrating super-polynomial size lower bounds on propositional Frege proofs asks whether there is a family $(F_n)_{n=1}^\infty$ of propositional tautological formulas for which there is no polynomial p such that the minimal Frege proof size of F_n is at most $p(|F_n|)$, for all $n \in \mathbb{Z}^+$.

⁸This is not important in general, but for Frege proofs it is in fact implicit that propositional formulas are ordered.

2.2 Algebraic Models of Computation and Proofs

Here we define algebraic formulas (both commutative and non-commutative).

Definition 2.5 (Non-commutative formula). *Let \mathbb{F} be a field and $\{x_1, x_2, \dots\}$ be (algebraic) variables. A non-commutative algebraic formula (or non-commutative formula for short) is a finite (ordered) labeled tree, with edges directed from the leaves to the root, and with fan-in at most two, such that there is an order on the edges coming into a node: the first edge is called the left edge and the second one the right edge. Every leaf of the tree (namely, a node of fan-in zero) is labeled either with an input variable x_i or a field element. Every other node of the tree is labeled either with $+$ or \times . We assume that there is only one node of out-degree zero, called the root.*

A non-commutative formula *computes* a non-commutative polynomial in $\mathbb{F}\langle x_1, \dots, x_n \rangle$ in the following way. A leaf computes the input variable or field element that labels it. A plus gate computes the sum of polynomials computed by its incoming nodes. A product gate computes the *non-commutative* product of the polynomials computed by its incoming nodes according to the order of the edges. (Subtraction is obtained using the constant -1 .) The output of the formula is the polynomial computed at the root. The **depth** of a formula is the maximal length of a path from the root to the leaf. The **size** of a non-commutative formula F is the total number of internal nodes (i.e., all nodes except the leaves) in its underlying tree, and is denoted similarly to the Boolean case by $|F|$.

The definition of (a commutative) algebraic formula is almost identical:

Definition 2.6 ((Commutative) algebraic formula). *An algebraic formula is defined in a similar way to a non-commutative formula, except that we ignore the order of multiplication (that is, a product node does not have order on its children and there is no order on multiplication when defining the polynomial computed by a formula).*

Substitutions of non-commutative formulas into other non-commutative formulas are defined and denoted similarly to substitutions in Boolean formulas.

We say that an algebraic formula is **homogenous** if all its nodes compute homogenous polynomials.

Note that we consider algebraic formulas as syntactic objects. For example, $x_1 + x_2$ and $x_2 + x_1$ are different formulas.

For the purpose of comparing the relative complexity of different proof systems we have the concept of a **simulation**. Specifically, we say that a propositional proof system P *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 translation (such as $\text{tr}(\cdot)$) 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 . Specifically, if $t(n) = n^{O(\log n)}$ we say the simulation is *quasi-polynomial*. We say that P and Q are *polynomially equivalent* in case P simulates Q and Q simulates P . (Our simulations will always be formally $t(n)$ -simulations, though we might not always state explicitly that the map f , from Q -proofs to P -proofs is efficiently computable, and only show the *existence* of a P -proof whose size is proportional to the corresponding Q -proof.)

General algebraic propositional proof systems such as the polynomial calculus and the Nullstellensatz system were introduced and studied in e.g., [BIK⁺96a, BIK⁺96b, CEI96].

3 Non-Commutative Ideal Proof System Simulates Frege

Here we prove Theorem 1.4, stating that non-commutative IPS simulates Frege. We shall prove in fact a slightly stronger version, that deals with proofs from assumptions:

Theorem 3.1. *Non-commutative IPS simulates Frege. More precisely, if T has a size s Frege proof from assumptions F_i , $i \in [m]$ (for some $m \geq 0$), then there is a non-commutative IPS proof of $\text{tr}(T)$ (over \mathbb{Z}_p for a prime p , or \mathbb{Q}) of size $\text{poly}(s)$ from assumptions $\text{tr}(F_i)$, $i \in [m]$.*

Note that if an unsatisfiable CNF formula C has a Frege refutation of size s , in the sense that Frege proves $\neg C$ in proof-size s , then Theorem 3.1 implies that there is a non-commutative IPS refutation of $\text{tr}(C)$ of size $\text{poly}(s)$: first prove $\text{tr}(\neg C) = 1 - \text{tr}(C)$ in non-commutative IPS (without assumptions), and then, using the assumption $\text{tr}(C)$, imply 1.

Recall that Raz and Shpilka [RS05] gave a deterministic polynomial-time PIT algorithm for non-commutative formulas (over any field):

Theorem 3.2 (PIT for non-commutative formulas [RS05]). *There is a deterministic polynomial-time algorithm that decides whether a given noncommutative formula over a field \mathbb{F} computes the zero polynomial.⁹*

Now, since we write refutations as non-commutative formulas we can use the theorem above to check in deterministic polynomial-time the correctness of non-commutative IPS refutations, from which follows that non-commutative IPS is a Cook-Reckhow proof system (Corollary 1.5).

Proof of Theorem 3.1. In general, the proof proceeds as a step-wise simulation of Frege in a similar manner to [GP14] and [GH03].

Recall that by Reckhow's theorem [Rec76a] we can work with any concrete Frege proof system. Thus, we shall consider the Frege system to be Schoenfield's system from Section 1.1. Consider the Frege proof π of T , from assumptions F_i , for $i \in [m]$. A basic structural result shown in Krajíček [Kra95] states that a Frege proof can be turned into a tree-like Frege proof with only a polynomial increase in size. Thus, without loss of generality we assume that π is tree-like.

We construct a corresponding non-commutative IPS proof of $\text{tr}(T)$ from the assumptions F_i 's using the tree-like Frege proof π . Let π be written as a sequence of propositional proof-lines $\ell_1, \ell_2, \dots, \ell_k$, let $L_i := \text{tr}(\ell_i)$, for $i \in [k]$, be the algebraic translation of the proof-lines, and let $\mathbf{F} := (\text{tr}(F_1(\bar{x})), \dots, \text{tr}(F_m(\bar{x})))$ be the vector of translation of the assumptions. Note that \mathbf{F} does not include the Boolean and commutator axioms. For convenience, let $C_{i,j}$ denote the commutator axiom $x_i \cdot x_j - x_j \cdot x_i$, for $i, j \in [n], i \neq j$, and let \mathbf{C} denote the vector of all the $C_{i,j}$ axioms and \mathbf{B} the vector of all Boolean axioms for the variable \bar{x} . When we write $P \cdot Q - Q \cdot P$ where P, Q are (non-commutative) formulas, we mean $((P \cdot Q) + (-1 \cdot (Q \cdot P)))$.

The following lemma suffices to conclude the proof.

Lemma 3.3. *For every $i \in [k]$, there exists a non-commutative formula $\phi_i(\bar{x}, \bar{y}, \bar{z}, \bar{w})$ such that*

1. $\phi_i(\bar{x}, \bar{0}, \bar{0}, \bar{0}) = 0$;
2. $\phi_i(\bar{x}, \mathbf{F}, \mathbf{B}, \mathbf{C}) = L_i$;
3. $|\phi_i| \leq (\sum_{\ell \in A_i} |L_\ell|)^4$, where $A_i \subset [k]$ are the indices of the Frege proof-lines involved in deriving L_i .

For example, if L_i is derived by L_α and L_α is derived by L_β for some $\beta < \alpha < i \in [k]$, then we say that α, β are both involved in deriving L_i . In other words, the lines involved in deriving a proof-line L_i are all the proof-lines in the sub-tree of L_i when we consider the underlying graph of the (tree-like) proof as a tree.

⁹We assume here that the elements of \mathbb{F} have an efficient representation and the field operations are efficiently computable (e.g., the field of rationals).

Notice that if the lemma holds, then ϕ_k is a non-commutative IPS proof of $\text{tr}(T)$, and its size is bounded by $\left(\sum_{\ell \in A_k} |L_\ell|\right)^4 \leq \left(\sum_{\ell \in [k]} |L_\ell|\right)^4 \leq O(|\pi|^4)$.

Proof. We construct ϕ_i by induction on the length k of the refutation π . That is, for any i , from 1 to k , we construct the non-commutative formula $\phi_i(\bar{x}, \bar{y}, \bar{z}, \bar{w})$ according to L_i , as follows:

Base case:

Case 1: $L_i \in \mathbf{F}$ is a (translation of an) assumption, i.e., $L_i = \text{tr}(F_j)$, for $j \in [m]$. Let $\phi_i := y_j$. Obviously, $\phi_i(\bar{x}, \bar{0}, \bar{0}, \bar{0}) = 0$; $\phi_i(\bar{x}, \mathbf{F}, \mathbf{B}, \mathbf{C}) = L_i$ and $|\phi_i| = 1 \leq |L_i|^4$.

Case 2: $L_i = \text{tr}(\ell_i)$, where $\ell_i = A \rightarrow (B \rightarrow A)$. Recall that \rightarrow is an abbreviation, hence, $\ell_i = \neg A \vee (\neg B \vee A)$, and $L_i = (1 - \text{tr}(A)) \cdot ((1 - \text{tr}(B)) \cdot \text{tr}(A))$.

We wish to construct a non-commutative IPS proof of the non-commutative formula L_i . For this purpose, we first construct a non-commutative IPS proof of the polynomial $\text{tr}(A) - \text{tr}(A)^2 = (1 - \text{tr}(A)) \cdot \text{tr}(A)$, with size at most $|\text{tr}(A)|^2$. This is doable using only the Boolean axioms $x_i - x_i^2$ by Lemma 3.6. Then, we prove L_i from $(1 - \text{tr}(A)) \cdot \text{tr}(A)$ in non-commutative IPS as follows: we first multiply $(1 - \text{tr}(A)) \cdot \text{tr}(A)$ by $(1 - \text{tr}(B))$ from the right, to get $(1 - \text{tr}(A)) \cdot \text{tr}(A) \cdot (1 - \text{tr}(B))$. And then we use the commutator axioms to commute the rightmost product in order to derive L_i , using Lemma 3.4, proved below.

Case 3: $L_i = \text{tr}(\ell_i)$, where ℓ_i is one of the other two axioms in Schoenfield's system. This case is similar to the previous case.

Induction step:

Case 1: $L_i = \text{tr}(B)$, where $\ell_i = B$ was derived by modus ponens from $\ell_j = A$ and $\ell_h = A \rightarrow B$, for $j, h < i$. Thus, $L_j = \text{tr}(A)$ and $L_h = (1 - \text{tr}(A)) \cdot \text{tr}(B)$.

By induction hypothesis we have a non-commutative IPS proof $\phi_j(\bar{x}, \mathbf{F}, \mathbf{B}, \mathbf{C}) = \text{tr}(A)$. Thus, $\phi_j(\bar{x}, \mathbf{F}, \mathbf{B}, \mathbf{C}) \cdot \text{tr}(B) = \text{tr}(A) \cdot \text{tr}(B)$. Similarly, by induction hypothesis we have $\phi_h(\bar{x}, \mathbf{F}, \mathbf{B}, \mathbf{C}) = (1 - \text{tr}(A)) \cdot \text{tr}(B)$. Therefore, letting $\phi_i(\bar{x}, \bar{y}, \bar{z}, \bar{w}) := \phi_j(\bar{x}, \bar{y}, \bar{z}, \bar{w}) \cdot \text{tr}(B) + \phi_h(\bar{x}, \bar{y}, \bar{z}, \bar{w})$ leads to $\phi_i(\bar{x}, \mathbf{F}, \mathbf{B}, \mathbf{C}) = \text{tr}(A) \cdot \text{tr}(B) + (1 - \text{tr}(A)) \cdot \text{tr}(B) = \text{tr}(B)$. Again, by induction hypothesis $|\phi_i| = |\phi_j| + |\phi_h| + |\text{tr}(B)| + 2 \leq \left(\sum_{\ell \in A_j} |L_\ell|\right)^4 + \left(\sum_{\ell \in A_h} |L_\ell|\right)^4 + |L_i| + 2 \leq \left(\sum_{\ell \in A_i} |L_\ell|\right)^4$, where the right most inequality holds since π is a *tree-like* refutation and hence $A_j \cap A_h = \emptyset$, meaning that $\sum_{\ell \in A_j} |L_\ell| + \sum_{\ell \in A_h} |L_\ell| \leq \sum_{\ell \in A_i} |L_\ell|$. \square

This concludes the proof of the simulation (Theorem 3.1). \square

It remains to prove the following lemmas.

Lemma 3.4. *Let $M = L(f \cdot g)$ be a non-commutative formula containing the displayed subformula (i.e., subtree) $f \cdot g$ (that is, containing the subtree consisting of a product gate and left and right children f, g , respectively). Similarly, let $N = L(g \cdot f)$. Then, there is a non-commutative formula $\phi_{M,N}(\bar{x}, \bar{z})$ in variables $\{x_\ell, z_{\alpha,\beta}, \ell \in [n], \alpha < \beta \in [n]\}$, such that:*

1. $\phi_{M,N}(\bar{x}, \bar{0}) = 0$;
2. $\phi_{M,N}(\bar{x}, \mathbf{C}) = M - N$;
3. $|\phi_{M,N}| \leq |M|^2 |N|^2$.

Proof. We define the non-commutative formula $\phi_{M,N}$ inductively as follows:

- If $M = (P \cdot Q)$, and $N = (Q \cdot P)$, then $\phi_{M,N}$ is defined to be the formula constructed in Lemma 3.5 below.

- $M = (P \cdot Q)$, $N = (P' \cdot Q')$.
Case 1: If $P = P'$, then let $\phi_{M,N} := (P \cdot \phi_{Q,Q'})$.
Case 2: If $Q = Q'$, then let $\phi_{M,N} := (\phi_{P,P'} \cdot Q)$.
- $M = (P + Q)$, $N = (P' + Q')$.
Case 1: If $P = P'$, then let $\phi_{M,N} = \phi_{Q,Q'}$.
Case 2: If $Q = Q'$, then let $\phi_{M,N} = \phi_{P,P'}$.

By induction, the construction satisfies the desired properties. \square

Lemma 3.5. *For any pair P, Q of two non-commutative formulas there exists a non-commutative formula F in variables $\{x_\ell, y_{i,j}, \ell \in [n], i < j \in [n]\}$ such that:*

1. $F(\bar{x}, \bar{0}) = 0$;
2. $F(\bar{x}, \mathbf{C}) = P \cdot Q - Q \cdot P$;
3. $|F| = |P|^2 |Q|^2$.

Proof. Let $s(P, Q)$ denote the smallest size of F satisfying the above properties. We will show that $s(P, Q) \leq |P|^2 \cdot |Q|^2$ by induction on $\max(|P|, |Q|)$.

Base case: $|P| = |Q| = 1$. In this case both P and Q are constants or variables, thus $s(P, Q) = 1 \leq |P|^2 |Q|^2$.

In the following induction step, we consider the case where $|P| \geq |Q|$ (which is symmetric to the case $|P| < |Q|$).

Induction step: Assume that $|P| \geq |Q|$.

Case 1: The root of P is addition. Let $P = (P_1 + P_2)$. We have (after rearranging):

$$P \cdot Q - Q \cdot P = ((P_1 \cdot Q - Q \cdot P_1) + (P_2 \cdot Q - Q \cdot P_2)).$$

By induction hypothesis, we have $s(P, Q) \leq s(P_1, Q) + 1 + s(P_2, Q) \leq |P_1|^2 |Q|^2 + 1 + |P_2|^2 |Q|^2 \leq (|P_1| + |P_2| + 1)^2 |Q|^2 = |P|^2 \cdot |Q|^2$.

Case 2: The root of P is a product gate.

Let $P = (P_1 \cdot P_2)$. By rearranging:

$$P \cdot Q - Q \cdot P = ((P_1 \cdot (P_2 \cdot Q - Q \cdot P_2)) + ((P_1 \cdot Q - Q \cdot P_1) \cdot P_2))$$

By induction hypothesis, we have $s(P, Q) = |P_1| + 1 + s(P_2, Q) + 1 + s(P_1, Q) + 1 + |P_2| \leq |P_1| + 1 + |P_2|^2 |Q|^2 + 1 + |P_1|^2 |Q|^2 + 1 + |P_2| \leq (|P_1| + |P_2| + 1)^2 |Q|^2 = |P|^2 \cdot |Q|^2$. \square

The following lemma is similar to [GH03, Lemma 4] (proved there for a variant of the polynomial calculus):

Lemma 3.6. *Let $f = \text{tr}(p)$ be a non-commutative formulas in the variables x_1, \dots, x_n for some propositional formula p . Then, there exists a non-commutative IPS proof of $f \cdot (1 - f)$ of size at most $|f|^2$.*

Proof. By induction on the structure of f .

Base case: f is a variable. Then, by using the Boolean axioms we are done.

Induction step:

Case 1: $f = 1 - \text{tr}(p)$ for some propositional formula p . We need to prove $(1 - \text{tr}(p)) \cdot (1 - (1 - \text{tr}(p))) = (1 - \text{tr}(p)) \cdot \text{tr}(p)$, and by induction hypothesis there is a non-commutative IPS proof of $\text{tr}(p) \cdot (1 - \text{tr}(p))$ of size at most $|p|^2$ using the Boolean axioms only.

Case 2: $f = \text{tr}(p) \cdot \text{tr}(q)$ for some propositional formulas p, q . By induction hypothesis we have a non-commutative IPS proofs of $\text{tr}(p) \cdot (\text{tr}(p) - 1)$ and $\text{tr}(q) \cdot (\text{tr}(q) - 1)$, each of size at most $|p|^2$ and $|q|^2$, resp. Observe that

$$\begin{aligned} \text{tr}(p) \cdot (\text{tr}(p) - 1) \cdot \text{tr}(q)^2 + \text{tr}(p) \cdot \text{tr}(q) \cdot (\text{tr}(q) - 1) &= \\ \text{tr}(p)^2 \cdot \text{tr}(q)^2 - \text{tr}(p) \cdot \text{tr}(q)^2 + \text{tr}(p) \cdot \text{tr}(q)^2 - \text{tr}(p) \cdot \text{tr}(q) &= \\ \text{tr}(p)^2 \cdot \text{tr}(q)^2 - \text{tr}(p) \cdot \text{tr}(q). \end{aligned} \tag{1}$$

Thus, formula (1) constitutes the desired non-commutative IPS proof, having size $|p|^2 + |q|^2 + 2 \cdot |q| + |p| + 4 \leq |p|^2 + |q|^2 + 1 + 2|p| \cdot |q| + 2|p| + 2|q| = (|p| + |q| + 1)^2$, for any $|p| \geq 1$ and $|q| \geq 1$.

Case 3: $f = 1 - (1 - \text{tr}(p)) \cdot (1 - \text{tr}(q))$. Similar to the previous case. \square

Note: The same proof holds also for a Frege refutation of a sequence of unsatisfiable axioms. By replacing instead of \mathbf{F} , the axioms. Also, the same proof holds for proofs from assumption in Frege: a Frege proof from assumptions can be simulated in non-commutative IPS.

4 Frege Quasi-Polynomially Simulates Non-Commutative IPS

In this long section we prove Theorem 1.7 stating that the Frege system quasi-polynomially simulates the non-commutative IPS (over $GF(2)$). Together with Theorem 1.4, this gives a new characterization (up to a quasi-polynomial increase in size) of propositional Frege proofs as non-commutative algebraic formulas.

We use the notation in Section 1.3.3 as follows: for a clause κ_i in a CNF $\phi = \kappa_1 \wedge \dots \wedge \kappa_m$, we denote by Q_i^ϕ the non-commutative formula translation $\text{tr}'(\kappa_i)$ of the clause κ_i (Definition 1.6). Thus, $\neg x$ translates to x , x translates to $1 - x$ and $f_1 \vee \dots \vee f_r$ translates to $\prod_i \text{tr}'(f_i)$ (considered as a tree of product gates with $\text{tr}'(f_i)$ as leaves), and where the formulas are over $GF(2)$ (meaning that $1 - x$ is in fact $1 + x$). Recall that this way, for every 0-1 assignment (when we identify true with 1 and false with 0), $Q_i^\phi = 0$ iff κ_i is true.

As mentioned in the introduction, it will be evident that our proof in fact establishes a slightly tighter simulation of the non-commutative IPS by Frege. Specifically, if the degree of the non-commutative IPS refutation is r and its formula depth is d , then there is a Frege proof of $\neg\phi$ with size $\text{poly}\left(\binom{d+r+1}{r} \cdot s\right)$. This will follow from our efficient simulation within Frege of Raz's [Raz13] homogenization construction (Lemma 4.6). Nevertheless, for simplicity we shall always assume that the depth d of the non-commutative IPS refutation formula is logarithmic in the size s and that the degree r of the refutation is at most $s + 1$, and thus will not take care to explicitly establish the dependence of the simulation on the parameters d and r .

The rest of the paper is dedicated to proving Theorem 1.7.

4.1 Balancing Non-Commutative Formulas

First we show that a non-commutative formula of size s can be balanced to an equivalent formula of depth $O(\log s)$, and thus we can assume that the non-commutative IPS certificate is already given as a

balanced formula (this is needed for what follows). Both the statement of the balancing construction and its proof are similar to Proposition 4.1 in Hrubeš and Wigderson [HW14] (which in turn is similar to the case of commutative formulas with division gates in Brent [Bre74]). (Note that a formula of a logarithmic depth (in the number of variables) must have a polynomial-size (in the number of variables).)

Observation 4.1. *Assume that a non-commutative polynomial p can be computed by a formula of size s . Then p can be computed by a formula of depth $O(\log s)$ (and hence of polynomial-size when s is polynomial in the number of variables).*

Proof. The proof is almost identical to Hrubeš and Wigderson's proof of Proposition 4.1 in [HW14], which deals with rational functions and allows formulas with division gates. Thus, we only outline the argument in [HW14] and argue that if the given formula does not have division gates, then the new formula obtained by the balancing construction will not contain any division gate as well.

Notation. *Let F be a non-commutative formula and let g be a gate in F . We denote by F_g the subformula of F whose root is g , and by $F[z/g]$ the formula obtained by replacing F_g in F by the variable z . We denote by $\widehat{F}, \widehat{F}_g$ the non-commutative polynomials in $\mathbb{F}\langle X \rangle$ computed by F and F_g , respectively.*

We simultaneously prove the following two statements by induction on s , concluding the lemma:

Inductive statement: *If F is a non-commutative formula of size s , then for sufficiently large s and suitable constants $c_1, c_2 > 0$, the following hold:*

- (i) \widehat{F} has a non-commutative formula of depth at most $c_1 \log s + 1$;
- (ii) if z is a variable occurring at most once in F , then:

$$\widehat{F} = A \cdot z \cdot B + C,$$

where A, B, C are non-commutative polynomials that do not contain z , and each can be computed by a non-commutative formula of depth at most $c_2 \log s$.

Base case: $s = 1$. In this case there is one gate g connecting two variables or constants. Thus, (i) in the inductive statement can be obtained immediately as it is already computed by a formula of depth $1 = \log s + 1$. As for (ii), note that in the base case, F is a formula with only one gate g . Assuming that z is a variable occurring only once in F , it is easy to construct non-commutative formulas A, B, C so that $\widehat{F} = A \cdot z \cdot B + C$ for which the conditions in (ii) hold as follows:

Case 1: if g is a plus gate connecting the variable z with a variable or constant $x \neq z$, then we can write F as $1 \cdot z \cdot 1 + x$.

Case 2: if g is a product gate connecting z with x (for $z \neq x$, and in this order), then we can write F as $1 \cdot z \cdot x + 0$.

Case 3: if g is a product gate connecting x with z (for $z \neq x$, and in this order), then we can write F as $x \cdot z \cdot 1 + 0$.

Induction step: (i) is established (slightly informally) as follows. Find a gate g in F such that both F_g and $F[z/g]$ are small (of size at most $2s/3$, and where z is a new variable that does not occur in F). Then, by applying induction hypothesis on $F[z/g]$, there exist formulas A, B, C of small depth such that $\widehat{F[z/g]} = A \cdot z \cdot B + C$. Thus, $\widehat{F} := A \cdot \widehat{F}_g \cdot B + C$.

To prove (ii), find an appropriate gate g on the path between z and the output of F (an *appropriate* g is a gate g such that $F[z_1/g]$ and F_g are both small (of size at most $2s/3$), where z_1 is a new variable not occurring in F). Use the inductive assumptions to write:

$$\widehat{F}[z_1/g] = A_1 \cdot z_1 \cdot B_1 + C_1 \quad \text{and} \quad \widehat{F}_g = A_2 \cdot z \cdot B_2 + C_2$$

and compose these expressions to get

$$\widehat{F} = A_1 \cdot (A_2 \cdot z \cdot B_2 + C_2) \cdot B_1 + C_1 = A' \cdot z \cdot B' + C',$$

where $A' = A_1 \cdot A_2$, $B' = B_2 \cdot B_1$, $C' = A_1 \cdot C_2 \cdot B_1 + C_1$.

It is clear that the respective depth of A' , B' and C' are all at most $c_2 \log(2s/3) + 2 \leq c_2 \log s$ when s is sufficiently large.

To finish the proof of (ii), it suffices to show that A' , B' , C' do not contain the variable z . It is enough to prove that $A_1, B_1, C_1, A_2, B_2, C_2$ do not contain z . Notice that F_g contains z and z is a variable occurring at most once in F . Therefore $\widehat{F}[z_1/g]$ does not contain the variable z , which means that both A_1, B_1, C_1 do not contain z . Moreover, by induction hypothesis, we know that A_2, B_2, C_2 do not contain z . Therefore, we conclude that A', B', C' do not contain z . \square

As a consequence of Observation 4.1, in what follows, without loss of generality we will assume that F is given already in a balanced form, namely has depth $O(\log s)$ and size s .

4.2 The Reflection Principle

Here we show that the existence of a non-commutative IPS refutation of the CNF ϕ with size s and depth $O(\log s)$ implies the existence of a Frege proof of $\neg\phi$ with size $s^{O(\log s)}$ (we use the same notation as in the beginning of Section 4). This is done by proving a *reflection principle* for the non-commutative IPS inside Frege. The use of the reflection principle to obtain simulation results is standard in the proof complexity literature (cf. [CN10]). The argument establishing the soundness of the IPS involves showing that a polynomial identity is true (as a formal polynomial identity). Grochow-Pitassi [GP14] introduced a collection of propositional axioms, stating the correctness of a PIT algorithm, under the name *PIT-axioms*, which they showed are sufficient to characterize the soundness of IPS. In what follows we shall prove a reflection principle for the non-commutative IPS. The proof in this section (namely, Sec. 4.2) follows similar lines to that in [GP14]. The actual correctness proof for the non-commutative formulas PIT algorithm (due to Raz-Shpilka [RS05]) is proved in later sections (Sec. 4.3 to 4.6), and is new.

A reflection principle for a given proof system P is a statement asserting that if a formula is *provable* in P then the formula is also *true*. Thus, suppose we have a short Frege proof of the following reflection principle for P :

$$“([\pi] \text{ is a } P\text{-proof of } [T]) \longrightarrow T”,$$

where $[T]$ and $[\pi]$ are some reasonable encodings of the tautology T and its P -proof π , respectively. Then, it is possible to obtain a Frege proof of T , assuming we already have a P -proof π of T : we simply plug-in the encodings $[\pi]$ and $[T]$ in the reflection principle, which makes the premise of the implication true.

Let F be a non-commutative formula over $\mathbf{GF}(2)$ and let $\overline{Q}^\phi(\overline{x})$ denote the vector $(Q_1^\phi, \dots, Q_m^\phi)$ (see Theorem 1.7). Since F is a non-commutative IPS refutation of ϕ we know that

$$F(\overline{x}, \overline{0}) = 0, \quad F(\overline{x}, \overline{Q}^\phi(\overline{x})) = 1. \quad (2)$$

We can treat F as a Boolean formula in the standard way:

Definition 4.2 (F_{bool}). Let $F(\bar{x})$ be a non-commutative formula over $GF(2)$ in the (algebraic) variables \bar{x} . We denote by $F_{bool}(\bar{p})$ the Boolean formula in the (propositional) variables \bar{p} obtained by turning every plus gate and multiplication gate into \oplus (i.e., XOR) and \wedge (i.e., AND) gates, respectively, and, for the sake of clarity, turning the input algebraic variables \bar{x} into the propositional variables \bar{p} . We sometimes write F and F_{bool} without explicitly mentioning the \bar{x} and \bar{p} variables.

Note that for any 0-1 assignment, F and F_{bool} take on the same value (when we identify true with 1 and false with 0). When we consider $F = F(\bar{x}, \bar{y})$ (with both the \bar{x} and \bar{y} variables), F_{bool} denotes the corresponding Boolean version of F where the variables \bar{x} are replaced by \bar{p} and the algebraic variables \bar{y} become the propositional variables \bar{y} . Therefore, by (2),

$$\neg F_{bool}(\bar{p}, \bar{0}) \quad \text{and} \quad F_{bool}(\bar{p}, \overline{Q_{bool}^\phi(\bar{p})}) \quad (3)$$

are both *tautologies* (though we still need to show that their Frege proofs are *short*). To conclude Theorem 1.7, we first prove $\neg\phi$ with a polynomial-size Frege proof, assuming we already proved (3) (this is done in Lemma 4.3 below, which is not very hard to establish). Second, we show that there exists an $s^{O(\log s)}$ Frege proof of (3) (which is done in the next section, and requires much more work).

Lemma 4.3. *There is a polynomial-size Frege proof of $\neg\phi(\bar{p})$, assuming $\neg F_{bool}(\bar{p}, \bar{0})$ and $F_{bool}(\bar{p}, \overline{Q_{bool}^\phi(\bar{p})})$ as axioms (polynomial in the size of $\phi(\bar{p})$ and $F_{bool}(\bar{p}, \overline{Q_{bool}^\phi(\bar{p})})$).*

Proof. By simple logical reasoning inside Frege. Informally, we show that assuming that $\phi(\bar{p})$ holds, for every $i \in [m]$, $Q_{i_{bool}}^\phi(\bar{p}) \equiv 0$, and so $\neg F_{bool}(\bar{p}, \bar{0})$ and $F_{bool}(\bar{p}, \overline{Q_{bool}^\phi(\bar{p})})$ cannot both hold (for \equiv denoting (semantic) logical equivalence).

In order to go from $\phi(\bar{p})$ to $Q_{i_{bool}}^\phi(\bar{p}) \equiv 0$ we need to deal with encoding of clauses inside Frege. Thus, let the propositional formula $\text{Truth}([\phi], \bar{p})$ express the statement that the assignment \bar{p} satisfies the formula ϕ , as defined below. In the following we denote by \bar{p} *actual* propositional variables occurring in a propositional formula (and not the encoding of variables; see below).

Encoding of 3CNFs and their Truth Predicate. We shall follow now [GP14, Section 4.3]. A positive natural number i is encoded with $\lceil \log_2 n \rceil$ bits (such that the numbers $1, \dots, 2^t$ are put into bijective correspondence with $\{0, 1\}^t$). We denote this encoding of i by $[i]$. A clause κ with three literals is encoded as the bit string $\bar{q}_1 s_1 \bar{q}_2 s_2 \bar{q}_3 s_3$, where each s_1, s_2, s_3 is the sign bit of the corresponding literal in κ (1 for positive and 0 for negative), and each $\bar{q}_1, \bar{q}_2, \bar{q}_3$ is a length- $\lceil \log_2 n \rceil$ bit string encoding the corresponding index of the variable (assuming the number of variables is n). For a bit string \bar{q} with the i th bit y_i we write $\bar{q} = [t]$ as an abbreviation of $\bigwedge_{i=1}^{\lceil \log_2 n \rceil} (y_i \leftrightarrow [t]_i)$. Finally, we define (where m is the number of clauses in the 3CNF)

$$\begin{aligned} \text{Truth}([\kappa], \bar{p}) &:= \bigvee_{j \in [3]} \bigvee_{i \in [n]} ([\bar{q}_j] = i \wedge (p_i \leftrightarrow s_j)) , \quad \text{and} \\ \text{Truth}([\phi], \bar{p}) &:= \bigwedge_{j \in [m]} \text{Truth}([\kappa_j], \bar{p}) . \end{aligned}$$

Note: It is important to note that, given a *fixed* CNF ϕ , the propositional formulas $\text{Truth}([\kappa], \bar{p})$ and $\text{Truth}([\phi], \bar{p})$ are formulas *in the propositional variables \bar{p} only*.

Let us now continue the proof of Lemma 4.3. It is easy to show (see [GP14] Lemma 4.9 for a proof) that after simplifying constants (e.g., $(A \wedge 1) \leftrightarrow A$) the formula $\text{Truth}([\kappa], \bar{p})$ becomes syntactically

identical to $\kappa(\bar{p})$ (i.e., the original clause κ in the propositional variables \bar{p}) thus there is a polynomial-size Frege proof of

$$\phi(\bar{p}) \rightarrow \text{Truth}([\phi], \bar{p}). \quad (4)$$

We shall now proceed within Frege. First, consider the propositional formulas

$$\text{Truth}([\kappa_i], \bar{p}) \rightarrow \neg Q_{i_{bool}}^\phi(\bar{p}), \quad \text{for all } i \in [m]. \quad (5)$$

By definition (recall that $Q_i^\phi = 0$ iff κ_i is true for any 0-1 assignment, when 1 is identified with **true**) all the formulas in (5) are tautologies. Note that all the premises and all the consequences in (5) are of constant size, and thus (5) can be proved with a Frege proof of constant-size for each $i \in [m]$ (by completeness). Further, using the fact that $\text{Truth}([\phi], \bar{p}) = \bigwedge_{i \in [m]} \text{Truth}([\kappa_i], \bar{p})$, we can easily prove in Frege with a polynomial-size proof that for each $i \in [m]$,

$$\text{Truth}([\phi], \bar{p}) \rightarrow \neg Q_{i_{bool}}^\phi(\bar{p}). \quad (6)$$

Assume by a way of contradiction that $\phi(\bar{p})$ holds. By modus ponens using (4) and (6), we have

$$\bigwedge_{i \in [m]} \neg Q_{i_{bool}}^\phi(\bar{p}). \quad (7)$$

We now argue (inside Frege) that, assuming also $\neg F_{bool}(\bar{p}, \bar{0})$, (7) implies $\neg F_{bool}(\bar{p}, \overline{Q_{bool}^\phi(\bar{p})})$. By (7), for every $i \in [m]$, $Q_{i_{bool}}^\phi(\bar{p})$ is logically equivalent to 0 (which is identified with **false**), and hence $\neg F_{bool}(\bar{p}, \overline{Q_{bool}^\phi(\bar{p})}) \equiv \neg F_{bool}(\bar{p}, \bar{0})$. Thus, assuming $\neg F_{bool}(\bar{p}, \bar{0})$ we have also $\neg F_{bool}(\bar{p}, \overline{Q_{bool}^\phi(\bar{p})})$. But this contradicts the assumption that $F_{bool}(\bar{p}, \overline{Q_{bool}^\phi(\bar{p})})$, and hence we reach a contradiction with the assumption that $\phi(\bar{p})$ holds. \square

It remains to show a quasi-polynomial-size proof of (3). We abbreviate $\neg F_{bool}(\bar{p}, \bar{0})$ and $F_{bool}(\bar{p}, \overline{Q_{bool}^\phi(\bar{p})})$, by

$$F'_{bool}(\bar{p}), \quad F''_{bool}(\bar{p}), \quad \text{respectively.} \quad (8)$$

Note that the substitutions of the constants 0 or the constant depth formulas Q_{bool}^ϕ in F cannot increase the depth of F too much (i.e., can add at most a constant to the size of F). In other words, the depths of the formulas in (8) are still $O(\log s)$.

Proof of Theorem 1.7 (Second main theorem). Using Theorem 1.8 that we prove below, we get that (8) can be proved in quasi-polynomial-size (in s the size of the IPS refutation of the given CNF ϕ). And together with Lemma 4.3 above, this shows that $\neg\phi$ can be proved in quasi-polynomial-size in s , concluding the proof. \square

4.3 Non-Commutative Formula Identities have Quasi-Polynomial-Size Proofs

Recall that a (commutative or non-commutative) multivariate polynomial f is *homogeneous* if every monomial in f has the same total degree. For each $0 \leq j \leq d$, denote by $f^{(j)}$ the homogenous part of degree j of f , that is, the sum of all monomials (together with their coefficient from the field) in f of total degree j . We say that a *formula* is *homogeneous* if each of its gates computes a *homogeneous* polynomial. We use the following technical definitions:

Definition 4.4 (Syntactic degree). *Define the syntactic degree of a non-commutative formula F , $\deg(F)$, as follows: (i) If F is a field element or a variable, then $\deg(F) = 0$ and $\deg(F) = 1$, respectively; (ii) $\deg(F + G) = \max(\deg(F), \deg(G))$, and $\deg(F \times G) = \deg(F) + \deg(G)$, where $+$, \times denote the plus and product gates respectively.*

Definition 4.5 (Syntactic homogenous non-commutative formula). *We say that a non-commutative formula is syntactic homogenous if for every plus gate $F + G$ with two children F and G , $\deg(F) = \deg(G)$.*

To complete the proof of Theorem 1.7 it remains to prove Theorem 1.8. The rest of the paper is dedicated to proving the latter.

Remaining Proof Overview. For the convenience of the reader we highlight here in an informal manner the main steps in the proof of Theorem 1.8 stating that the Boolean versions of non-commutative formulas F computing the zero polynomial over $GF(2)$ have Frege refutations (i.e., proofs of negation) of quasi-polynomial size.

1. We prove *inside Frege* that F_{bool} can be written as a sum (modulo 2) of the (Booleanized) syntactic homogenous components $F_{bool}^{(i)}$, each of quasi-polynomial size in $|F_{bool}|$, using Raz's [Raz13] construction. In other words, breaking a non-commutative algebraic formula into its syntactic homogenous parts is efficiently provable in Frege.

So it remains to show that each $\neg F_{bool}^{(i)}$ has a polynomial-size (in the size of $F_{bool}^{(i)}$, which is quasi-polynomial in the size of F) Frege proof.

2. Note that $F_{bool}^{(i)}$ does not contain variables iff $F^{(i)}$ does not contain variables. In case $F_{bool}^{(i)}$ does not contain variables it is easy to refute $F_{bool}^{(i)}$. So it remains to treat the case where $F^{(i)}$ is non-constant. In this case we show that Frege can easily prove that $F_{bool}^{(i)}$ is equivalent to some $F'_{bool}{}^{(i)}$, where $F'^{(i)}$ is a constant-free algebraic non-commutative syntactic homogenous formula computing the zero polynomial (here, a *constant-free* algebraic circuit means that the circuit does not contain any leaf labelled by a field element¹⁰).
3. Having a constant-free algebraic non-commutative syntactic homogenous formula computing the zero polynomial $F'^{(i)}$, we use similar ideas as Raz and Shpilka [RS05] to construct a polynomial-size Frege refutation of (the Boolean version of) $F'^{(i)}$, as follows¹¹:
 - (a) Since $F'^{(i)}$ is constant-free and syntactic homogenous, using the standard transformation [RS05, Nis91] of a non-commutative formula to an algebraic branching program (ABP; Definition 4.11) results in a *layered* (i.e., standard) ABP A (this is different from [RS05] who had to deal with non-layered ABPs first). Assuming the syntactic degree of F is d , the final layer of A , consisting of the sink, is also d .
 - (b) Using the ABP A , we identify a collection of witnesses that witness the fact that A computes the zero polynomial. Informally, these witnesses are a collection of 0-1 matrices. Each 0-1 matrix denoted Λ_i has a small number of rows (proportional to the size of the ABP). Each (possibly zero) row \mathbf{v} of these matrices corresponds to a linear combination $\mathbf{v} \cdot \bar{A}_{d-i}$ (where \cdot is the inner product) of the polynomials computed by the ABPs whose sources are in the

¹⁰This is a stricter requirement than the usual meaning of a constant-free circuit, that allows the use of 0 or 1.

¹¹The reader might want to consult [RS05] for the deterministic PIT algorithm for non-commutative formulas (and ABPs). Because we work inside the restrictive framework of a Frege proof system, our proofs contain further technicalities that are avoided in [RS05].

i th layer of A and whose sinks are all the (single) original sink of A (and thus each of these ABPs computes a degree $d-i$ homogenous polynomial). The requirement is that for all such rows, $\mathbf{v} \cdot \overline{A}_{d-i} = 0$, and so overall $\Lambda_i \overline{A}_{d-i} = 0$. Note that each non-commutative polynomial in \overline{A}_{d-i} is homogenous of degree $d-i$.

Following a similar argument to [RS05], we show that one can find matrices Λ_i such that, in addition to the above requirement, the following holds:

$$\Lambda_i \overline{A}_{d-i} = \mathbf{T}_{i+1} \Lambda_{i+1} \overline{A}_{d-i-1},$$

where $\mathbf{T}_1, \dots, \mathbf{T}_d$ are matrices with homogenous linear forms in every entry, and such that the product of the matrix $\mathbf{T}_{i+1} \Lambda_{i+1}$ with \overline{A}_{d-i-1} is construed in a *syntactic* way; that is, $\mathbf{T}_{i+1} \Lambda_{i+1}$ is interpreted as an adjacency matrices of a layer in an ABP where the (l, k) entry of the matrix is the linear form that labels the edge going from the l th node in layer i to the k th node in layer $(i+1)$ —and so $\mathbf{T}_{i+1} \Lambda_{i+1} \overline{A}_{d-i-1}$ is a new ABP with $d-i$ layers.

- (c) Note that it is unclear how to (usefully) represent an ABP directly in a Frege system, because apparently ABP is a stronger model than formulas (and each Frege proof-line is written as a formula). Thus, we cannot directly work with ABPs within Frege proofs, and consequently we cannot use the witnesses from part (3b). We solve this problem by replacing every ABP in the witnesses by a corresponding non-commutative formula: every ABP in the witnesses from (3b) is a part of the ABP that was constructed from $F^{(i)}$ in (3a). We notice that every such part of ABP corresponds to a certain substitution instance of $F^{(i)}$. Thus, we replace every such part of ABP in the witnesses with its corresponding substitution instance of $F^{(i)}$. Having these witnesses enables us to carry out a step by step proof of the fact that $F^{(i)}$ computes the zero polynomial (formally, a Frege refutation of the Boolean version of $F^{(i)}$).

Proof of Theorem 1.8. The formula F is of size s which means that the maximal degree of a polynomial computed by F is at most $s+1$. By Section 4.1 we can assume without loss of generality that F is of $O(\log s)$ depth. Raz [Raz13] showed that we can always split F into syntactic homogenous formulas $F^{(i)}$, $i = 0, \dots, s+1$, each of size $s^{O(\log s)}$. In Lemma 4.6, proved in the next section, we show that this homogenization construction can already be proved efficiently in Frege. In other words, we show that there exists an $s^{O(\log s)}$ -size Frege proof of

$$\bigoplus_{i=0}^{s+1} F_{bool}^{(i)} \leftrightarrow F_{bool}. \quad (9)$$

By Lemma 4.9 proved in the sequel, for any *syntactic homogenous* non-commutative formula H that computes the identically zero polynomial over $GF(2)$, $\neg H_{bool}$ admits a polynomial-size (in the size of H) Frege proof (recall that $\neg H_{bool}$ is a tautology whenever H is a non-commutative formula computing the zero polynomial over $GF(2)$). Thus, by Theorem 1.8, for every $F^{(i)}$, $i = 0, \dots, s+1$, there exists an $s^{O(\log s)}$ -size Frege proof of $\neg F_{bool}^{(i)}$. That is, there exists an $s^{O(\log s)}$ -size Frege proof of $\neg \left(\bigoplus_{i=0}^{s+1} F_{bool}^{(i)} \right)$.

Note that Lemma 4.9 gives proofs that have size polynomial in the size of $\neg F_{bool}^{(i)}$, and this latter size is $s^{O(\log s)}$. Together with tautology (9), we can derive $\neg F_{bool}$ in Frege. \square

4.4 Proving the Homogenization of Non-Commutative Formulas in Frege

To complete the proof of Theorem 1.8 it remains to prove Lemmas 4.6 and 4.9. Lemma 4.6 states that Raz's construction from [Raz13] for homogenizing algebraic formulas is efficiently provable in Frege (and is also applicable to non-commutative formulas):

Lemma 4.6. *If F is a non-commutative formula of size s and depth $O(\log s)$ and $F^{(0)}, \dots, F^{(s+1)}$ are the syntactic homogenous formulas computing F 's homogenous parts of degrees $0, \dots, s+1$, respectively, constructed according to [Raz13] (sketched below), then there exists an $s^{O(\log s)}$ -size Frege proof of:*

$$\left(\bigoplus_{i=0}^{s+1} F^{(i)} \right) \leftrightarrow F_{\text{bool}}. \quad (10)$$

Proof. We first introduce basic notations and observations for describing a (commutative) formula homogenization construction from [Raz13]. This construction is a somewhat more involved variant of the standard homogenization construction for circuits laid out by Strassen [Str73]. We then construct the desired short Frege proofs, which in turn also shows how to construct the homogenous formulas themselves. Overall, the proof proceeds as a proof-complexity simulation of Raz's [Raz13], similar in spirit to [HT15] (in which Strassen homogenization construction was considered).

Raz's Formula Homogenization Construction. Given a balanced (commutative) algebraic formula F we wish to construct s (commutative) formulas computing the homogenous parts $F^{(i)}$, $i = 0, \dots, s$. Define the *product-depth* of a gate u , denoted u_{pd} , as the maximal number of product gates along a directed path from u to the output gate (including u). Since the formula F is balanced, the depth is at most $O(\log s)$, namely the largest value of u_{pd} for any node u in F is $O(\log s)$.

Consider the directed path from u to the root (including the node u). Informally we want to describe a possible progression of the degree of a monomial computed along the path from u to the root. Observe that any possible degree progression must occur on product gates. That is, for a gate u with product-depth u_{pd} , there are u_{pd} "choices" for the degree of a monomial to increase (i.e., "to progress").

Formally, for every integer r , denote by N_r the family of monotone non-increasing functions D from $\{0, 1, \dots, r\}$ to $\{0, 1, \dots, s+1\}$. It is helpful to think of D as a function that maps u and all the product nodes along the directed path from u to the root, into the corresponding degree of a monomial as it is computed along the path. In this map u is identified with r and the root is identified with 0. Thus, r , which associated with u , is mapped to the total degree of the monomial computed in node u , and the root 0 is mapped to the total degree of the monomial computed at the root. Therefore, the set $N_{u_{pd}}$ describes all possible progressions of the degree of monomials along the path from u to the root. Note that a product gate may not increase the degree of a monomial computed along a path, because we may consider the monomial as multiplied by a constant. Hence, the functions in N_r are not necessarily strictly decreasing. Let D_i^u be the set of all D^u in $N_{u_{pd}}$, such that $D^u(u_{pd}) = i$. So intuitively D_i^u describes the set of all possible monomial degree progressions for monomials *whose degree in u is i* .

The size of N_r is $\binom{r+s+2}{r+1} = \binom{r+s+2}{s+1}$ (the number of combinations with repetitions of $r+1$ elements from $s+2$ elements, which determine functions in N_r). Therefore, for every node u in F , the size of the set $N_{u_{pd}}$ is at most

$$\binom{s + O(\log s) + 2}{s} = s^{O(\log s)}. \quad (11)$$

We construct the desired syntactic homogenous formulas $F^{(0)}, F^{(1)}, \dots, F^{(s+1)}$ following Raz's construction [Raz13]. For a node u in F , let F_u denote the subformula rooted at u in F , and let \widehat{F}_u denote the (commutative) polynomial computed by F_u . Split every gate u in F into $|N_{u_{pd}}|$ gates labeled (u, D^u) , for each $D^u \in N_{u_{pd}}$. For every gate u in F and every $D^u \in N_{u_{pd}}$ we construct the formula F_{u, D^u}^* whose root is (u, D^u) , such that F_{u, D^u}^* is a homogenous formula computing the degree- $D^u(u_{pd})$ homogenous part of \widehat{F}_u . There might be some isolated nodes (u, D^u) , namely nodes to which no edge is connected, and for such nodes (u, D^u) we consider F_{u, D^u}^* to be 0.

The construction of each F_{u,D^u}^* , for every gate u in F and every $D^u \in N_{u_{pd}}$, is done by induction on the formula size of F . It might be helpful for the reader to consult [Raz13] to grasp better the intuition of the construction itself, however our presentation is self-contained, as we will show how the construction is efficiently provable already inside the Frege system (this will also show that F_{u,D^u}^* computes the degree- $D^u(u_{pd})$ homogeneous part of \widehat{F}_u , when considered as Boolean function).

Efficient Proofs of the Homogenization for Non-Commutative Formulas Construction.

Let u be a node in F and assume $i \in \{0, \dots, s_u + 1\}$, where s_u denotes the size of F_u . By construction [Raz13], for every pair of D and D' in D_i^u , the formulas $F_{u,D_i^u}^*$ $F_{u,D_i'^u}^*$ are (syntactically) identical. We use a similar inductive argument as in [Raz13], from leaves to the root of F , showing that for every gate u in F there exists an $s^{O(\log s)}$ -size Frege proof of

$$\left(\bigoplus_{i=0}^{s_u+1} F_{u,D_i^u}^* \text{ bool} \right) \leftrightarrow F_u \text{ bool}, \quad (12)$$

where (abusing notation slightly) we denote by $F_{u,D_i^u}^*$ a *single* formula (since any choice of $D \in D_i^u$ leads to the same formula $F_{u,D}^*$).

Observe that, using the same notation as above, the formulas $F_{r,D_0^r}^*, \dots, F_{r,D_{s+1}^r}^*$, for r being the root of F , are just those desired formulas $F^{(0)}, F^{(1)}, \dots, F^{(s+1)}$. Thus, eventually, when we prove (12) for the root node r , we prove the existence of an $s^{O(\log s)}$ -size Frege proof of the Boolean formulas in (10). The construction of the proof is syntactic, and preserves the order of multiplication of variables and thus holds for non-commutative formulas.

Note that the size of the Frege proof we construct is quasi-polynomial in s . This is because of the following: for every node u in F and every $D^u \in N_{u_{pd}}$ we construct a proof of (12). Recall that for every u in F , $|N_{u_{pd}}| = s^{O(\log s)}$, by (11). Thus, the total number of such nodes u and functions D^u is $s \cdot s^{O(\log s)} = s^{O(\log s)}$, and so this is the total number of proofs of (12) we construct. Each such proof of (12) requires only poly(s)-size *assuming* we already have proved (by induction hypothesis) the required previous instantiations of (12). Therefore, we end up with a proof of total size $\text{poly}(s) \cdot s^{O(\log s)} = s^{O(\log s)}$.

Base case: If u is a leaf, for each $D^u \in N_{u_{pd}}$, F_{u,D^u}^* is defined to be a single node. Furthermore, if u is labeled by a field element, F_{u,D^u}^* is the same field element in case $D^u(u_{pd}) = 0$ and is 0 in case $D^u(u_{pd}) \neq 0$. If u is an input variable, F_{u,D^u}^* is the same input variable in case $D^u(u_{pd}) = 1$ and is 0 in case $D^u(u_{pd}) \neq 1$. Thus, for each $D^u \in N_{u_{pd}}$, either F_{u,D^u}^* computes $F_u^{(D^u(u_{pd}))}$ or 0. That is, we can easily prove in Frege

$$\left(\bigoplus_{i=0}^{s_u+1} F_{u,D_i^u}^* \text{ bool} \right) \leftrightarrow F_u \text{ bool}.$$

Induction step:

Case 1: Assume that u is a sum gate with children v, w . For every $D^u \in N_{u_{pd}}$, let $D^v \in N_{v_{pd}}$ be the function that agrees with D^u on $\{0, 1, \dots, u_{pd}\}$ and satisfies $D^v(v_{pd}) = D^u(u_{pd})$, and in the same way, let $D^w \in N_{w_{pd}}$ be the function that agrees with D^u on $\{0, 1, \dots, u_{pd}\}$ and satisfies $D^w(w_{pd}) = D^u(u_{pd})$. We define

$$F_{u,D^u}^* := F_{v,D^v}^* + F_{w,D^w}^*.$$

Assume $D^u(u_{pd}) = j$. Then, define

$$\widehat{F}_{u,D_j^u}^* := \widehat{F}_{v,D_j^v}^* + \widehat{F}_{w,D_j^w}^*.$$

Therefore, the following is a tautology:

$$F_{u,D_j^u}^* \text{ bool} \leftrightarrow \left(F_{v,D_j^v}^* \text{ bool} \oplus F_{w,D_j^w}^* \text{ bool} \right), \quad \text{for all } j = 0, \dots, s+1.$$

By induction hypothesis on the nodes v, w , we have

$$F_v \text{ bool} \leftrightarrow \bigoplus_{i=0}^{s_v+1} F_{v,D_i^v}^* \text{ bool}, \quad F_w \text{ bool} \leftrightarrow \bigoplus_{i=0}^{s_w+1} F_{w,D_i^w}^* \text{ bool},$$

and so we can prove

$$\bigoplus_{i=0}^{s_u+1} \left(F_{v,D_i^v}^* \text{ bool} \oplus F_{w,D_i^w}^* \text{ bool} \right) \leftrightarrow F_v \text{ bool} \oplus F_w \text{ bool},$$

which gives us (since u is a plus gate)

$$\bigoplus_{i=0}^{s_u+1} F_{u,D_i^u}^* \text{ bool} \leftrightarrow F_u \text{ bool}.$$

Case 2: If u is a product gate with children v, w , using the same notation as above, for $j = 0, \dots, s_u+1$, we define $F_{u,D_j^u}^* := \sum_{i=0}^j F_{v,D_i^v}^* \cdot F_{w,D_{j-i}^w}^*$. If $j > s_u+1$, let $F_{u,D_j^u}^* := 0$. Similarly, using the induction hypothesis on the nodes v, w and observing the fact that $s_u = s_v + s_w + 1$, we can prove:

$$F_u \text{ bool} \leftrightarrow \bigoplus_{i=0}^{s_u+1} F_{u,D_i^u}^* \text{ bool}$$

as follows:

$$\begin{aligned} F_u \text{ bool} &\leftrightarrow (F_v \text{ bool} \wedge F_w \text{ bool}) \\ &\leftrightarrow \left(\bigoplus_{j=0}^{s_v+1} F_{v,D_j^v}^* \text{ bool} \right) \wedge \left(\bigoplus_{i=0}^{s_w+1} F_{w,D_i^w}^* \text{ bool} \right) \\ &\leftrightarrow \bigoplus_{j=0}^{s_v+s_w+2} \bigoplus_{i=0}^j \left(F_{v,D_i^v}^* \text{ bool} \wedge F_{w,D_{j-i}^w}^* \text{ bool} \right) \\ &\leftrightarrow \bigoplus_{j=0}^{s_u+1} \left(\bigoplus_{i=0}^j \left(F_{v,D_i^v}^* \text{ bool} \wedge F_{w,D_{j-i}^w}^* \text{ bool} \right) \right) \\ &\leftrightarrow \bigoplus_{i=0}^{s_u+1} F_{u,D_i^u}^* \text{ bool}. \end{aligned}$$

□

4.5 Homogenous Non-Commutative Formula Identities have Polynomial-Size Frege Proofs

To conclude Theorem 1.8 it remains to prove Lemma 4.9. Here we will prove Lemma 4.9, based on further lemmas we prove in the next section.

First, we need to set some notation. We denote by

$$F \vdash^* F'$$

the fact that F' can be derived with a polynomial in $|F'|$ size Frege proof, given F as a (possibly empty) assumption in the proof.

In practice we will almost always use this notation when F' can be derived from F by simple syntactic manipulations of formulas using mostly structural rules such as the associativity and distributivity rules, as well as simple logical identities (e.g., $\text{false} \oplus G \equiv G$, where \oplus stands for XOR). This notation will make our arguments a bit more convenient to read. For the most part, we will also leave it to the reader to verify that indeed F' can be obtained from F with a short Frege proof, since it will be evident from the way F and F' are defined.

Accordingly, for two *vectors* of formulas $\overline{F}, \overline{G}$, we denote by $\overline{F} \vdash^* \overline{G}$ the fact that each entry in \overline{G} can be derived from the corresponding entry in \overline{F} with a short Frege proof.

The following definition is essential to Section 4.6.2 where we talk about algebraic branching programs (ABPs). This definition will enable us to identify within a homogenous non-commutative formula a certain part of the formula (after substitution) that corresponds to a sub-algebraic branching program.

Definition 4.7 (Induced part of a formula). *Let F' be a subformula of F and g_1, \dots, g_k be some nodes (gates) in F' and c_1, \dots, c_k be constants in \mathbb{F} . Then $F'[c_1/g_1, \dots, c_k/g_k]$ is called an induced part of F .*

We sometimes call an induced part of a formula simply a *part of a formula*.

Getting Rid of Constants

For technical reasons (concerning the conversion of a non-commutative syntactic homogenous formulas into a layered ABP in what follows) it will be convenient only to consider algebraic (resp. Boolean) formulas with *no* 0-1 (resp. true, false) constants. We say that a Boolean or algebraic formula is *non-constant* if it contains at least one variable.

Lemma 4.8 (Constant-free formulas). *Let F be a non-constant and non-commutative formula over $GF(2)$ that computes the (non-commutative) zero-polynomial. Then, there exists a constant-free non-commutative formula F' of size $\text{poly}(|F|)$ that computes the (non-commutative) zero polynomial, such that $F_{\text{bool}} \vdash^* F'_{\text{bool}}$.*

Note that since F does not contain 0-1 constants, F' does not contain true, false constants.

Proof. First, notice that substitution of equivalent terms can be simulated efficiently in Frege, in the following sense: if Φ is a formula and ψ is a subformula occurring in Φ , then $\psi \leftrightarrow \psi' \vdash^* \Phi'$, where Φ' is Φ in which the (single) occurrence ψ is substituted by ψ' .

Therefore, we can iteratively take the constants out of F_{bool} within Frege using local substitution of logically equivalent terms, as follows: if F contains a subformula $0 + G$, for some formula G , we change it to G ; if F contains a subformula $1 + G$ we change it to $\neg G$; if F contains a subformula $0 \times G$, we change it to 0 ; if F contains a subformula $1 \times G$ we change it to G . Doing these replacements iteratively we arrive at either the 0 formula or a formula without constants (since every step reduces the size of the formula). The 0 formula is arrived only when there are no variables in F , and so this cannot happen by our assumption. \square

Main Technical Lemma

Lemma 4.9. *There exists a constant c such that for any non-commutative syntactic homogeneous formula $F(\bar{x})$ over $\mathbf{GF}(2)$ of size s that is identically zero, the corresponding Boolean tautology $\neg F_{\text{bool}}(\bar{p})$ has a Frege proof of size at most s^c (for sufficiently large s).*

Proof. First, by Lemma 4.8 we can assume without loss of generality that $F(\bar{x})$ is constant-free (or else, we can either derive an equivalent constant-free formula or simply the constant false, both with polynomial-size Frege proofs). Let d be the syntactic degree of F .

Note that the syntactic degree d of F is at most $s + 1$. Theorem 4.10, proved in the next section, states the existence of a collection of witnesses that witness that the homogenous non-commutative constant-free formula F computes the non-commutative zero polynomial. As demonstrated below, these witnesses will enable us to inductively and efficiently prove in Frege that F is the zero polynomial (over $GF(2)$).

First, we give the formal description of the witnesses and their properties and then explain informally why they witness the identity and why they exist for every identity.

Notation. For a matrix \mathbf{T} with entries \mathbf{T}_{ij} , each a non-commutative formula, and a vector $\bar{F} = (v_1, \dots, v_m)$ of non-commutative formulas, we write $\mathbf{T}\bar{F}$ to denote the (transposed) vector of non-commutative formulas whose j th entry is $\mathbf{T}_{j1} \times F_1 + \dots + \mathbf{T}_{jn} \times F_m$ written as a balanced (depth $\leq \log m + 1$) binary tree of plus gates at the top and the formulas $\mathbf{T}_{jk} \times F_k$'s at the leaves. For a 0-1 matrix Λ , we write $\Lambda\bar{F}$ to denote the vector of non-commutative formulas similar to the way defined above for $\mathbf{T}\bar{F}$, except that now the matrix \mathbf{T} has a 0-1 formula in each entry. We denote by $\mathbf{T}\Lambda\bar{F}$ the vector of non-commutative formulas $(\mathbf{T}\Lambda)\bar{F}$, where the (i, j) entry of the matrix $\mathbf{T}\Lambda$ is $\sum_k \mathbf{T}_{ik}\Lambda_{kj}$ written as a balanced tree of plus gates with corresponding leaves as before (and where $\mathbf{T}_{ik}\Lambda_{jk}$ is written as \mathbf{T}_{ik} if $\Lambda_{kj} = 1$ and does not occur in the sum if $\Lambda_{kj} = 0$). When we write $\vdash^* A \leftrightarrow B$ in the witnesses below, for A and B non-commutative algebraic formulas over $GF(2)$, we intend to treat $A \leftrightarrow B$ as a Boolean tautology (Definition 4.2). For two vectors of formulas $\mathbf{v} = (v_1, \dots, v_m)$, $\mathbf{u} = (u_1, \dots, u_m)$, we write $\vdash^* \mathbf{v} \leftrightarrow \mathbf{u}$ to denote $\vdash^* v_i \leftrightarrow u_i$, for all $i \in [m]$.

Identity Witnesses

1. For every $i = 0, \dots, d - 1$, Λ_i is a 0-1 matrix of dimension $m_i \times m_i$, where $m_i = \text{poly}(|F|)$, for all i . We set $\Lambda_0 = 1$.
2. For every $i = 1, \dots, d - 1$, \mathbf{T}_i is an $m_{i-1} \times m_i$ matrix whose entries are homogenous linear forms in the \bar{x} variables with 0-1 coefficients.
3. For every $i = 1, \dots, d$, \bar{F}_i is a vector of induced parts of F (Definition 4.7), each computing a homogenous non-commutative polynomial of degree exactly i . The length of the vectors \bar{F}_i is m_{d-i} . Accordingly, we denote by $\widehat{\bar{F}}_i$ the vector of non-commutative polynomials in \bar{F}_i .

These witnesses are such that the following hold (we use the notation defined above):

$$\begin{aligned}
\Lambda_{d-i}\widehat{\bar{F}}_i &= \bar{0}, \quad i = 1, \dots, d, \quad \text{semantically;} \\
\vdash^* F &\leftrightarrow \Lambda_0\bar{F}_d \quad (\text{meaning that } \vdash^* F \leftrightarrow F_d, \text{ since } \Lambda_0 = 1) \\
&(\text{note that } \bar{F}_d \text{ is identical to } F_d, \text{ because the only induced part} \quad (13) \\
&\text{of } F \text{ of degree } d \text{ is } F \text{ itself, due to syntactic homogeneity);} \\
\vdash^* \Lambda_{d-i}\bar{F}_i &\leftrightarrow \mathbf{T}_{d-i+1}\Lambda_{d-i+1}\bar{F}_{i-1}, \quad i = 2, \dots, d.
\end{aligned}$$

Using the Witnesses. The identity witnesses provide a way to prove inductively that the non-commutative syntactic homogenous and constant free formula F is identically zero (when considered as a Boolean formula over $GF(2)$). Informally, we start with $\Lambda_{d-1}\overline{F}_1 = 0$ (considered as a Boolean equality) which is a true identity by (13). Since this identity is written as a sum of linear forms it has a polynomial-size proof. From this we also get $\mathbf{T}_{d-1}\Lambda_{d-1}\overline{F}_1 = 0$, and since $\Lambda_{d-2}\overline{F}_2 = \mathbf{T}_{d-1}\Lambda_{d-1}\overline{F}_1$, we derive $\Lambda_{d-2}\overline{F}_2 = 0$. Continuing in this fashion we finally derive $\Lambda_0\overline{F}_d = 0$, which by (13) concludes the proof.

It is worth noting that we cannot directly represent the formula F as the iterated matrix product $\mathbf{T}_{d-1} \cdots \mathbf{T}_2 \Lambda_1 \overline{F}_1$ in the proof, since writing explicitly this iterated matrix product will incur an exponential-size blow-up.

For readers already familiar with ABPs we remark that the m_i , for $i = 0, \dots, d-1$, will be the width of the i th layer in the ABP corresponding to the formula $F(\overline{x})$.

In what follows we make the above argument *formal*. We demonstrate a proof of $\neg F_{bool}$ based on the tautological Boolean formula obtained from the equation and short Frege proofs of the tautological Boolean formulas in (13). Denote by $\overline{F}_{bool}(\overline{p})_i$ the vector of all corresponding Boolean formulas of the formulas in \overline{F}_i , and let $F_{bool}(\overline{p})_{i,t}$ be the t th coordinate of this vector.

By (13), we know that $\Lambda_{d-i}\widehat{\overline{F}}_i = \overline{0}$ holds for every $i \in [d]$. Thus, using the notation we developed above, the following Boolean formulas are all tautologies:

$$\bigwedge_{w \in [m_{d-i}]} \left(\neg \left(\bigoplus_{t: \Lambda_{d-i}(w,t)=1} F_{bool}(\overline{p})_{i,t} \right) \right), \quad i = 1, \dots, d. \quad (14)$$

By (13), for every $i = 2, \dots, d$, and every $u \in [m_{d-i}]$, we have short Frege proofs of the following logical equivalence (between two Boolean formulas; the left hand side being the u th row in $\Lambda_{d-i}\overline{F}_i$):

$$\left(\bigoplus_{t: \Lambda_{d-i}(u,t)=1} F_{bool}(\overline{p})_{i,t} \right) \leftrightarrow \bigoplus_{w \in [m_{d-i+1}]} \left(\mathbf{T}_{d-i+1}(u, w)_{bool}(\overline{p}) \wedge \left(\bigoplus_{t: \Lambda_{d-i+1}(w,t)=1} F_{bool}(\overline{p})_{i-1,t} \right) \right). \quad (15)$$

Claim. *There are polynomial-size Frege proofs of (14).*

This will conclude the proof of Lemma 4.9, since for $i = d$, we get a polynomial-size Frege proof of

$$\bigwedge_{w \in [m_d]} \left(\neg \left(\bigoplus_{t: \Lambda_d(w,t)=1} F_{bool}(\overline{p})_{d,t} \right) \right),$$

which is just $\neg F_{bool}(\overline{p})$ by (13) ($m_d = 1$ and $\Lambda_d = 1$).

Proof of claim: First fix $i = 1$. Since \overline{F}_1 is a vector of linear forms, (14) is a Boolean tautology which can be proved with a polynomial-size Frege proof. This means that when we fix $i = 2$, the right hand side of (15) becomes false for every $u \in [m_{d-2}]$, and so the left hand side of (15)

$$\bigoplus_{t: \Lambda_{d-2}(u,t)=1} F_{bool}(\overline{p})_{2,t}$$

is also false for every $u \in [m_{d-2}]$. We continue in this manner until we arrive to (14) for $i = d$. \square_{claim}

We have thus concluded the proof of Lemma 4.9 (assuming Theorem 4.10). \square

4.6 Identity Witnessing Theorem

It remains to prove the following:

Theorem 4.10 (Identity witnessing theorem). *Let $F(\bar{x})$ be a non-commutative syntactic homogenous constant-free formula of degree d over $GF(2)$ computing the non-commutative zero polynomial. Then, the identity witnesses as defined in Section 4.5 exist.*

The proof of this theorem uses the notion of an algebraic branching program mentioned before as well as the Raz and Shpilka PIT algorithm [RS05]. Our proofs are self-contained, and we demonstrate formally the existence of the witnesses from scratch.

4.6.1 Algebraic Branching Programs

We introduce the following definition:

Definition 4.11 (ABP). *An algebraic branching program (ABP for short) is a directed acyclic graph with one source and one sink. The vertices of the graph are partitioned into layers numbered from 0 to d (the degree of the ABP), and edges may go only from layer i to layer $i + 1$. The source is the only vertex at layer 0, and the sink is the only vertex at layer d . Each edge is labeled with a homogeneous linear polynomial in the variables x_i (i.e., a function of the form $\sum_i c_i x_i$, with coefficients $c_i \in \mathbb{F}$, where \mathbb{F} is the underlying field). The size of an ABP is the number of its vertices. A path, directed from source to sink, in the ABP is said to compute the non-commutative product of linear forms on its edges (in the order they appear on the path). A node in the ABP computes the sum of all incoming paths arriving from the source. The ABP computes the non-commutative polynomial computed at its sink.*

Note that by definition an ABP computes a *homogenous* non-commutative polynomial.

Raz and Shpilka [RS05] established a deterministic polynomial-time algorithm for the polynomial identity testing of (non-commutative) ABPs. Therefore, by transforming a non-commutative formula to an ABP, one obtains a deterministic polynomial-time algorithm for the polynomial identity testing of non-commutative formulas.

Theorem 4.12 (Theorem 4, [RS05]). *Let A be an ABP of size s with $d + 1$ layers, then we can verify whether A computes the non-commutative zero polynomial in time $O(s^5 + s \cdot n^4)$.*

Using the algorithm demonstrated in Theorem 4.12, we give in Lemma 4.13 below witnesses that certify that a given non-commutative formula computes the zero polynomial. These witnesses *will not be our final witnesses* because they will incorporate ABPs, whereas in Section 4.5 we required the witnesses to consist of non-commutative *formulas* and not ABPs. In the next section we show, based on Lemma 4.13, how to obtain the desired formula-based witnesses.

Notation. For what follows in this section, let A be an ABP with $l + 1$ layers and where the source node v_{source} is on the 0th layer and the sink node v_{sink} is on the l th layer. For two nodes v', v'' in A , we denote by $A(v', v'')$ the ABP whose source is v' and sink is v'' (including all the paths leading from v' to v'' in A).

For every $j = 0, \dots, l$, we denote the nodes on the j th layer by v_{j1}, \dots, v_{jm_j} , where m_j stands for the total number of nodes in the j th layer. For a given $i = 0, \dots, l$, consider the ABP with m_{l-i} sources in layer $l - i$ and whose sink is v_{sink} . We can denote this multi-source ABP as a vector of ABPs:

$$\bar{A}_i = (A(v_{l-i,1}, v_{\text{sink}}), \dots, A(v_{l-i,m_{l-i}}, v_{\text{sink}})).$$

Each entry in this vector computes a non-commutative homogenous degree i polynomial. It is *important to note* that \bar{A}_i is only a convenient notation, namely, when we apply in what follows a matrix product

to the vector \bar{A}_i , we will treat different coordinates in the vector \bar{A}_i as having *joint nodes*. For instance, the sink node v_{sink} is treated as a *single node* shared by all the coordinates (and so in the vector \bar{A}_i it occurs only once). We will thus build a *single* ABP out of a matrix product with the vector \bar{A}_i , as described in what follows.

For a 0-1 matrix Λ of dimension $m \times m$ and a *multi-source* ABP \bar{A} with m sources v_1, \dots, v_m and 0 to l layers, we write $\Lambda\bar{A}$ to denote the $l+1$ layered ABP with m sources that results from \bar{A} when we join together several sources into a single source, maintaining the outgoing edges of the joined sources. Specifically, the i th source of the new ABP computes the non-commutative polynomial $\sum_k \Lambda_{ik} v_k$, and this is done by defining the outgoing edges of the i th source to be all the outgoing edges of v_k , for all v_k such that $\Lambda_{ik} = 1$.

For a matrix \mathbf{T} with dimension $m \times m'$ and entries \mathbf{T}_{ij} that are homogenous linear forms, and a multi-source ABP $\bar{A} = (v_1, \dots, v_{m'})$ we write $\mathbf{T}\bar{A}$ to denote the ABP whose 0 layer consists of m sources, and the i th node in the 0th layer, for $i = 1, \dots, m$, is connected to the j th node in 1st layer, for $j = 1, \dots, m'$, with an edge labeled by the linear form \mathbf{T}_{ij} . In case Λ is a 0-1 matrix, then $\mathbf{T}\Lambda\bar{A}$ stands for the result of the following process: first multiply the matrices \mathbf{T} and Λ in the standard way, obtaining a matrix \mathbf{T}' of new homogenous linear forms, and then multiply \mathbf{T}' by the vector \bar{A} , as explained above. We also denote by $\widehat{\bar{A}}$ the corresponding vector of non-commutative *polynomials* computed by the coordinates in \bar{A} .

With these notations in hand, we now construct the following ABP-variant of the identity witnesses:

Lemma 4.13 (existence of ABP-based identity witnesses). *If the ABP $A(v_{\text{source}}, v_{\text{sink}})$ computes the identically zero non-commutative polynomial, then the following hold:*

1. *There exist l matrices Λ_i with 0-1 entries, for $i = 0, \dots, l-1$, each of dimension $m_i \times m_i$, where $m_i = \text{poly}(|A|)$, such that $\Lambda_0 = 1$ and*

$$\Lambda_{l-i}\bar{A}_i = \bar{0}, \quad \text{for all } i = 0, \dots, l-1$$

(where the equality here is only semantic, i.e., the left hand side computes a vector of zero non-commutative polynomials).

2. *There exist $l-1$ matrices \mathbf{T}_i , for $i = 1, \dots, l-1$, of dimension $m_{i-1} \times m_i$ and whose entries are homogenous linear forms in the \bar{x} variables with 0-1 coefficients, such that*

$$\Lambda_{l-i}\bar{A}_i = \mathbf{T}_{l-i+1}\Lambda_{l-i+1}\bar{A}_{i-1}, \quad \text{for } i = 2, \dots, l, \quad \text{and} \quad (16)$$

$$\Lambda_0\bar{A}_l = A(v_{\text{source}}, v_{\text{sink}}), \quad (17)$$

and where the ABPs in these two equalities are constructed in the way described above (these two equalities above are syntactic, i.e., in each of the equations the two sides are syntactically identical as ABPs).

Proof. Recall that m_i is the number of nodes in the i th layer. Since we assumed $\Lambda_0 = 1$, and since $\bar{A}_l = A(v_{\text{source}}, v_{\text{sink}})$, we conclude equation (17) in the lemma.

We now construct by induction on j the matrix Λ_j , for $j = 0, \dots, l-2$, such that part 1 in the lemma holds:

$$\Lambda_j\bar{A}_{l-j} = 0, \quad (18)$$

as well as (16), that is,

$$\Lambda_j\bar{A}_{l-j} = \mathbf{T}_{j+1}\Lambda_{j+1}\bar{A}_{l-j-1}. \quad (19)$$

Base case: $\Lambda_0 = 1$ by assumption.

Induction step: Assume that for $0 \leq h < l-1$, $\Lambda_0, \dots, \Lambda_{h-1}$ and $\mathbf{T}_0, \dots, \mathbf{T}_h$ were already constructed, and that the equality (18) holds for every $j = 0, \dots, h$, and equality (19) holds for every $j = 0, \dots, h-1$. We construct Λ_{h+1} such that (18) holds for $j = h+1$:

$$\Lambda_{h+1} \bar{A}_{l-h} = 0,$$

and \mathbf{T}_{h+1} such that (19) holds with $j = h$:

$$\Lambda_h \bar{A}_{l-h} = \mathbf{T}_{h+1} \Lambda_{h+1} \bar{A}_{l-h-1}.$$

Let $M_{h,h+1}$ be the adjacency matrix of dimension $m_h \times m_{h+1}$ of the two consecutive layers h and $h+1$ in A , in which for every entry (p, q) , for $p \in [m_h], q \in [m_{h+1}]$,

$$M_{h,h+1}(p, q) = A(v_{h,p}, v_{h+1,q}) = \sum_{k=1}^n c_k x_k, \quad \text{for some } c_k \in \{0, 1\} \text{ (depending on } p, q, h).$$

The matrix $M_{h,h+1}$ can be written as $\sum_{k=1}^n x_k M_{h,h+1}^k$ (the superscript k is used here as an *index* only, and not as a matrix power), for some 0-1 matrices $M_{h,h+1}^k$. By the definition of an ABP

$$\begin{aligned} \bar{A}_{l-h} &= M_{h,h+1} \bar{A}_{l-h-1} \\ &= \sum_{k=1}^n x_k M_{h,h+1}^k \bar{A}_{l-h-1}. \end{aligned}$$

Moreover, if $\Lambda_h \bar{A}_{l-h} = \bar{0}$, then

$$\Lambda_h \sum_{k=1}^n x_k M_{h,h+1}^k \bar{A}_{l-h-1} = 0,$$

and therefore, by the non-commutativity of product we have

$$\Lambda_h M_{h,h+1}^k \bar{A}_{l-h-1} = \bar{0}, \quad \text{for } k = 1, \dots, n. \quad (20)$$

Now, consider the basis of the span of all row vectors in all the matrices $\Lambda_h M_{h,h+1}^k$, for $k = 1, \dots, n$. The number of vectors in this basis is at most the number of columns in (each of the) $M_{h,h+1}^k$ matrices, that is, at most m_{h+1} (which equals the number of nodes in the $h+1$ layer). Define the matrix Λ_{h+1} to be the $m_{h+1} \times m_{h+1}$ matrix whose rows are the vectors in this basis (if the basis consists of less than m_{h+1} vectors we can simply put zero rows to reach m_{h+1} rows). By (20) we know that every row vector in $\Lambda_h M_{h,h+1}^k$ is *orthogonal* to \bar{A}_{l-h-1} (i.e., their inner product is zero). Thus, any vector in the basis of the rows of $\Lambda_h M_{h,h+1}^k$, for $k = 1, \dots, n$, is also orthogonal to \bar{A}_{l-h-1} , and so

$$\Lambda_{h+1} \bar{A}_{l-h-1} = \bar{0}.$$

By the properties of a basis of a linear space, there must exist matrices \mathbf{T}_{h+1}^k , such that

$$\Lambda_h M_{h,h+1}^k = \mathbf{T}_{h+1}^k \Lambda_{h+1}, \quad \text{for all } k = 1, \dots, n.$$

Then, define $\mathbf{T}_{h+1} := \sum_{k=1}^n \mathbf{T}_{h+1}^k x_k$. Thus,

$$\Lambda_h \bar{A}_{l-h} = \mathbf{T}_{h+1} \Lambda_{h+1} \bar{A}_{l-h-1}.$$

□

4.6.2 Implicitly Working with ABPs in Frege System

Here we use Lemma 4.13 to conclude the existence of (formula-based) identity witnesses, which is required to conclude the proof of Theorem 4.10.

Recall the notion of an induced part of a formula (Definition 4.7): for a subformula F' of F and gates g_1, \dots, g_k in F' and 0-1 constants c_1, \dots, c_k , $F'[c_1/g_1, \dots, c_k/g_k]$ is called *an induced part of F* . Notice that it is unclear how to (usefully) represent an ABP directly in a Frege system, because apparently ABP is a stronger model than formulas (and each Frege proof-line is written as a formula). Thus, we cannot directly use the same formulation as [RS05]. This is the reason that we work with induced parts of formulas: let A be the ABP that corresponds to the non-commutative formula F , then for every node v in A there will be a corresponding induced part of F that computes the same polynomial computed by the sub-ABP rooted in v and whose sink is the sink of A . For this purpose we introduce the following notation and definition.

Informally, a *v -part of a formula F* is simply a substitution instance of F that computes the same polynomial as $A(v, v_{\text{sink}})$. Formally we have:

Definition 4.14 (*v -part of formula F*). *Let F be a homogenous formula and A be the corresponding ABP of F constructed according to the methods described in [RS05] (see also below), in which the source is v_{source} and the sink is v_{sink} . For any node v in A , if there exists an induced part of the formula F computing the same polynomial as $A(v, v_{\text{sink}})$, then we call this part a v -part of the formula F . (Note that for a node v there might be more than one v -part.)*

Let F be a non-commutative homogenous formula and let A be the corresponding ABP of F . In (the proof of) Lemma 4.15 below we construct a mapping between the nodes v in A to v -parts of F , denoted F_v^\bullet such that F_v^\bullet computes the non-commutative homogenous polynomial computed by $A(v, v_{\text{sink}})$. This will enable us to refer (implicitly) to $A(v, v_{\text{sink}})$ by an induced part F_v^\bullet of F , for any node v in A (though a v -part is not unique, our mapping will obviously associate a *unique* v -part to every node v in A).

Furthermore, for every node v in the ABP A the following holds (where, for the sake of simplicity, the algebraic formulas computing the linear forms computed by $A(v, u)$ are denoted also by $A(v, u)$):

$$F_v^\bullet \vdash^* \sum_{\substack{u: u \text{ has an incoming} \\ \text{edge from } v}} A(v, u) \times F_u^\bullet. \quad (21)$$

where, the big sum denotes a balanced binary tree of plus gates and the $A(v, u) \times F_u^\bullet$'s at the leaves.

It will be convenient to assume that the sink v_{sink} of an ABP is mapped to the empty formula and that if G is the empty formula, then $H \times G \vdash^* H$, where H stands for some nonempty formula.

Lemma 4.15. *For a non-commutative syntactic homogenous formula F without constants, let A be the ABP transformed from F by the methods in [RS05] (equivalently, in [Nis91]; we repeat this construction in the proof below), in which the source is v_{source} and the sink is v_{sink} . For every node v in A the non-commutative polynomial computed by $A(v, v_{\text{sink}})$ can be computed by some non-commutative formula, denoted F_v^\bullet , which is a v -part of F . Furthermore, for every node v in the ABP, (21) holds.*

Proof. We construct an ABP A , such that each node v in A is mapped to an induced part of F .

Constructing the ABP A . Given the non-commutative syntactic homogenous formula F over $GF(2)$ that does not contain constants, we construct the corresponding ABP A by induction on the size of F . By syntactic homogeneity we get a *standard (layered) ABP* (this differs from [RS05] who did not

start from a homogenous formula and so the resulted ABP was (initially) non-layered). Throughout the construction of A we maintain a mapping

$$g : \text{nodes}(A) \rightarrow \text{nodes}(F)$$

from nodes in A to their “corresponding” nodes in F (this will help us define the mapping F_v^\bullet). The reader can also consult the illustrated example in the sequel.

Base case: If F is a variable x_i , then A is a single edge $(v_{\text{source}}, v_{\text{sink}})$ labeled with x_i and $g(v_{\text{source}}) := F$ (i.e., the single node in F) and $g(v_{\text{sink}}) := \emptyset$ (i.e., “the empty node”).

Induction step:

Case 1: $F = G + H$. Then A is defined with the root v_{source} being the join of the two roots of the two ABPs constructed already for G, H (while keeping their outgoing edges). We then also join the two sinks of the ABPs for G, H (while keeping their incoming edges) into a single sink denoted v_{sink} .

The function g is defined as the union of the two original functions g 's for the ABPs for G and H , where the new nodes v_{source} and v_{sink} are mapped by g to the root of F and the empty formula \emptyset , respectively (note that the domains of both these g 's are *disjoint*—except for the two sources and two sinks).

Case 2: $F = G \times H$. Assume that A_G, A_H are the two ABPs already constructed for G, H , respectively. Then A is defined as A_G with the sink of A_G replaced by A_H . The function g is defined as the union of the two g functions for A_G, A_H (where the root of A_H is mapped by g to the root of the formula H).

Construction of F_v^\bullet . Let F be a syntactic homogenous non-commutative formula without constants, A its corresponding ABP, and $g : \text{nodes}(A) \rightarrow \text{nodes}(F)$ the function, whose constructions are all defined above. We construct the mapping F_v^\bullet for nodes v in A as follows. Throughout the construction we maintain the following conditions:

- (i) for every node v in A , the formula F_v^\bullet computes the same non-commutative polynomial as $A(v, v_{\text{sink}})$;
- (ii) for every node v in A , equation (21) above holds.

Let F be a non-commutative syntactic homogenous formula F , and t a node in F . Denote by r the root of F (in particular, if F is a variable then the root is the variable). We define the function $D(F, t)$ by induction on the structure of F as follows:

$$D(F, r) := F,$$

and for $t \neq r$ we define (for A, B two non-commutative syntactic homogenous formulas without constants):

$$D(A + B, t) := \begin{cases} D(A, t) + 0, & \text{if } t \in \text{nodes}(A); \\ 0 + D(B, t), & \text{if } t \in \text{nodes}(B), \end{cases}$$

and

$$D(A \times B, t) := \begin{cases} D(A, t) \times B, & \text{if } t \in \text{nodes}(A); \\ 1 \times D(B, t), & \text{if } t \in \text{nodes}(B) \end{cases}$$

(note the asymmetry in defining $D(A \times B)$, which corresponds to the way a non-commutative formula is translated into an ABP, with A computed “above” B).

Finally, for every node v in the ABP A , we define

$$F_v^\bullet := D(F, g(v)).$$

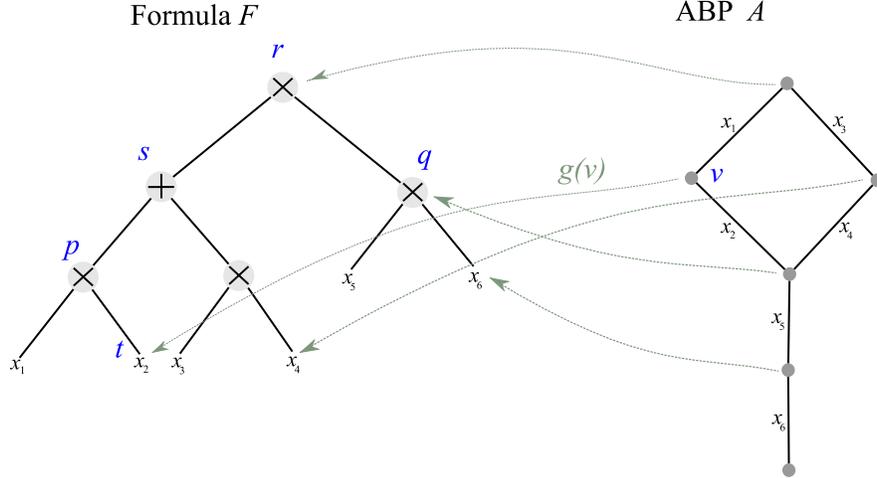


Figure 1

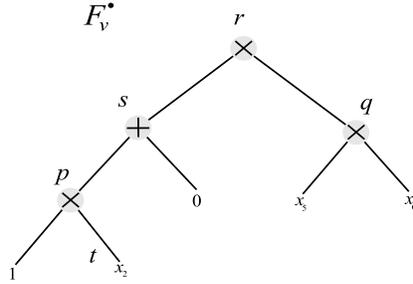


Figure 2

Example. Figure 1 illustrates a non-commutative syntactic homogenous and constant-free formula F , and its corresponding ABP A , together with the map $g : \text{nodes}(A) \rightarrow \text{nodes}(F)$. Figure 2 shows the formula F_v^\bullet (where v is the node in A from Figure 1). Note that indeed, by definition, $D(F, g(v)) = D(F, t) = D(F_s, t) \times F_q = (D(F_p, t) + 0) \times F_q = ((1 \times D(F_t, t)) + 0) \times F_q = ((1 \times x_2) + 0) \times F_q$.

Claim. Conditions (i) and (ii) above hold.

Proof of claim: Condition (i) holds by inspection of the definition of D , the construction of the ABP A from F and the function g giving the “origin” in F of each node in A .

Condition (ii), i.e., equation (21) for all v in A holds by condition (i) and the definition of D . Note that condition (i) already shows that $F_v^\bullet \leftrightarrow \sum_{\substack{u: u \text{ has an incoming} \\ \text{edge from } v}} A(v, u) \times F_u^\bullet$ is indeed a tautology

(considered over $GF(2)$). The fact that the right hand side of this tautology can be derived with a short (polynomial-size) Frege proof from the left hand side can be demonstrated by using basic structural derivation rules of Frege (e.g., associativity and distributivity) and simple logical equivalences (e.g., $1 \oplus G \leftrightarrow \neg G$). We omit the details. \square_{claim}

This concludes the proof of Lemma 4.15. \square

We are now ready to conclude the proof of the identity witnessing theorem (Theorem 4.10):

Proof of Theorem 4.10. Recall the ABP-based identity witnesses we showed existed in Lemma 4.13. Our goal is to show that there are (formula-based) identity witnesses (as defined in the proof of Theorem 1.8).

Using the correspondence given in Lemma 4.15, we can replace each ABP $A(v, v_{\text{sink}})$ occurring in some \bar{A}_i (for some node v in A and some $i = 0, \dots, l$) by a corresponding v -part F_v^\bullet . Denote with \bar{F}_i the result of this replacement. Thus, \bar{F}_i contains v -parts F_v^\bullet of F , each computing a homogenous polynomial of degree i . With this replacement we get (we assume that $l = d$ and we denote by $\text{poly}(n)$ a polynomial function of n):

1. There exist d matrices Λ_i , for $i = 0, \dots, d-1$, with 0-1 entries and dimension $m_i \times m_i$, where $m_i = \text{poly}(|A|)$, such that $\Lambda_0 = 1$, $\bar{F}_d = F$ and

$$\Lambda_{d-i} \bar{F}_i = \bar{0}, \quad \text{for all } i = 0, \dots, d-1.$$

2. There exist $d-1$ matrices \mathbf{T}_i , for $i = 1, \dots, d-1$, of dimension $m_{i-1} \times m_i$ and whose entries are homogenous linear forms in the \bar{x} variables with 0-1 coefficients, such that

$$\Lambda_{d-i} \bar{F}_i = \mathbf{T}_{d-i+1} \Lambda_{d-i+1} \bar{F}_{i-1}, \quad \text{for } i = 2, \dots, d, \quad \text{and} \quad (22)$$

$$\Lambda_0 \bar{F}_d = A(v_{\text{source}}, v_{\text{sink}}). \quad (23)$$

Recall that $\Lambda \bar{F}$, for Λ a matrix and \bar{F} a vector of formulas, is a vector of formulas, where each formula is written as a balanced (partial) sum of the formulas in \bar{F} (see the Identity Witnesses' definition in the proof of Theorem 1.8).

Our goal now is to show

$$\vdash^* \Lambda_{d-i} \bar{F}_i \leftrightarrow \mathbf{T}_{d-i+1} \Lambda_{d-i+1} \bar{F}_{i-1}, \quad i = 2, \dots, d, \quad (24)$$

and

$$\vdash^* F \leftrightarrow \Lambda_0 \bar{F}_d \quad (\text{meaning that } \vdash^* F \leftrightarrow F_d, \text{ since } \Lambda_0 = 1). \quad (25)$$

Note that (25) holds trivially since F_d and F are identical. For (24), by Lemma 4.15, we know that

$$F_v^\bullet \vdash^* \sum_{\substack{u: u \text{ has an incoming} \\ \text{edge from } v}} A(v, u) \times F_u^\bullet. \quad (26)$$

Consider $\Lambda_{d-i} \bar{F}_i$. Each of its entries is a (partial) sum of the formulas in \bar{F}_i written as a balanced sum. By (26) we can write each entry in \bar{F}_i as a (balanced) sum in which each summand is some linear form $A(v, u)$ times an entry in \bar{F}_{i-1} . We can thus write $\Lambda_{d-i} \bar{F}_i$ as $\mathbf{T}' \bar{F}_{i-1}$ for some matrix \mathbf{T}' with linear forms in each of its entries. Since the identity stated in (24) is a *true* identity, we thus get

$$\mathbf{T}' \bar{F}_{i-1} \leftrightarrow \mathbf{T}_{d-i+1} \Lambda_{d-i+1} \bar{F}_{i-1}.$$

But such an identity is provable in Frege with a polynomial-size proof, because we only need to prove an identity between $\langle \mathbf{t}'_j, \bar{F}_{i-1} \rangle$ and $\langle \mathbf{t}_j, \bar{F}_{i-1} \rangle$, for each of the j th rows \mathbf{t}'_j and \mathbf{t}_j of the matrices \mathbf{T}' and $\mathbf{T}_{d-i+1} \Lambda_{d-i+1}$, respectively (note that each entry of these rows is written as a *linear* form). \square

4.7 Conclusions

The propositional-calculus is ubiquitous in computation and logic. Within complexity and propositional proof complexity in particular it has a prominent role, and is considered a strong proof system whose structure and complexity is poorly understood. In that respect, we believe our characterization of Frege proofs and the propositional-calculus as non-commutative polynomials whose non-commutative formula size corresponds (up to a quasi-polynomial increase) to the size of Frege proofs, should be considered a valuable contribution.

In the framework of algebraic propositional proof systems (and especially the IPS framework and its precursors by Pitassi [Pit97, Pit98]) our characterization is *almost precise*, as we showed an almost tight two-sided simulation of Frege and non-commutative IPS. Although we left it open whether the simulation of non-commutative IPS by Frege can be improved from quasi-polynomial down to polynomial size, there is nothing to suggest at the moment this cannot be achieved.

Non-commutative formulas constitute a weak model of computation that is quite well understood. Since, as mentioned above, the Frege system is considered a strong proof system, and in fact it is not entirely out of question that Frege—or at least its extension, Extended Frege—is polynomially bounded (i.e., admits polynomial-size proofs for every tautology), on the face of it, our results are surprising.

Overall, we believe that this correspondence between non-commutative formulas and proofs may shed new light on the Frege lower bounds question, in so far that it reduces the problem of demonstrating Frege lower bounds to the problem of establishing non-commutative formula lower bounds on families of non-commutative polynomials. Following Nisan [Nis91], this task then reduces to the task of lower bounding the rank of certain families of (so-called, *partial-derivative*) matrices.

One may also speculate that a direct reduction from Frege lower bounds to the problem of lower bounding the non-commutative formula size of polynomials that are *already known to be hard*, is possible. We believe that this route cannot be ruled out.

Furthermore, ideas and lower bounds techniques connecting non-commutative computation, algebras with polynomial-identities (PI-algebras) and proof complexity as studied in [Hru11, LT13] might provide further tools for obtaining non-commutative IPS lower bounds.

Apart from the fundamental lower bound questions, the new characterization of Frege proofs sheds new light on the correspondence between *circuits and proofs* within proof complexity: in the framework of the ideal proof system, a Frege proof can be seen from the computational perspective as a non-commutative formula. This gives a different, and in some sense simpler, correspondence between proofs and computations than the traditional one (in which Frege corresponds to \mathbf{NC}^1 (cf. [CN10])).

We have also tightened the important results of Grochow and Pitassi [GP14]. Namely, by showing that already the non-commutative version of the IPS is sufficient to simulate Frege, as well as by showing *unconditional* efficient simulation of the non-commutative IPS by Frege.

Finally, while proving that Frege quasi-polynomially simulates the non-commutative IPS, we demonstrated new simulations of algebraic complexity constructions within proof complexity; these include the homogenization for formulas due to Raz [Raz13] and the PIT algorithm for non-commutative formulas due to Raz and Shpilka [RS05]. These proof complexity simulations add to the known previous such simulations shown in Hrubeš and the second author [HT15] (cf. [TC17]), and are of independent interest in the area of Bounded Arithmetic and feasible mathematics.

Acknowledgments

We are thankful to Joshua Grochow for very helpful comments, as well as for the anonymous reviewers and the editor for greatly improving the exposition of results.

References

- [ABB16] James Aisenberg, Maria Luisa Bonet, and Sam Buss. Quasipolynomial size Frege proofs of Frankl’s theorem on the trace of sets. *J. Symb. Log.*, 81(2):687–710, 2016. 1.1
- [ABRW04] Michael Alekhnovich, Eli Ben-Sasson, Alexander A. Razborov, and Avi Wigderson. Pseudorandom generators in propositional proof complexity. *SIAM J. Comput.*, 34(1):67–88, 2004. (A preliminary version appeared in Proceedings of the 41st Annual Symposium on Foundations of Computer Science (Redondo Beach, CA, 2000)). 3
- [AGP02] Albert Atserias, Nicola Galesi, and Pavel Pudlák. Monotone simulations of non-monotone proofs. *J. Comput. System Sci.*, 65(4):626–638, 2002. Special issue on complexity, 2001 (Chicago, IL). 1.2
- [Ajt88] Miklós Ajtai. The complexity of the pigeonhole principle. In *Proceedings of the IEEE 29th Annual Symposium on Foundations of Computer Science*, pages 346–355, 1988. 1.1, 1.2
- [AKV04] Albert Atserias, Phokion G. Kolaitis, and Moshe Y. Vardi. Constraint propagation as a proof system. In *CP*, pages 77–91, 2004. 1.2
- [BBP95] Maria Luisa Bonet, Samuel R. Buss, and Toniann Pitassi. 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, 1995. 1.1
- [BIK⁺96a] Paul Beame, Russell Impagliazzo, Jan Krajíček, Toniann Pitassi, and Pavel Pudlák. Lower bounds on Hilbert’s Nullstellensatz and propositional proofs. *Proc. London Math. Soc. (3)*, 73(1):1–26, 1996. Preliminary version in the *35th Annual IEEE Symposium on Foundations of Computer Science (FOCS 1994)*. 1.2, 2.2
- [BIK⁺96b] Samuel R. Buss, Russell Impagliazzo, Jan Krajíček, Pavel Pudlák, Alexander A. Razborov, and Jiří Sgall. Proof complexity in algebraic systems and bounded depth Frege systems with modular counting. *Computational Complexity*, 6(3):256–298, 1996. 2.2
- [BIK⁺97] Samuel R. Buss, Russell Impagliazzo, Jan Krajíček, Pavel Pudlák, Alexander A. Razborov, and Jiří Sgall. Proof complexity in algebraic systems and bounded depth Frege systems with modular counting. *Computational Complexity*, 6(3):256–298, 1996/97. 1.2
- [BPR00] Maria Luisa Bonet, Toniann Pitassi, and Ran Raz. On interpolation and automatization for Frege systems. *SIAM J. Comput.*, 29(6):1939–1967, 2000. 1.3.3
- [Bre74] Richard P. Brent. The parallel evaluation of general arithmetic expressions. *J. ACM*, 21(2):201–206, 1974. 4.1
- [CEI96] Matthew Clegg, Jeffery Edmonds, and Russell Impagliazzo. Using the Groebner basis algorithm to find proofs of unsatisfiability. In *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing (Philadelphia, PA, 1996)*, pages 174–183, New York, 1996. ACM. 2.2
- [CN10] Stephen Cook and Phuong Nguyen. *Logical Foundations of Proof Complexity*. ASL Perspectives in Logic. Cambridge University Press, 2010. 1.2, 1.3.3, 1.4, 4.2, 4.7
- [CR74a] Stephen A. Cook and Robert A. Reckhow. Corrections for “On the lengths of proofs in the propositional calculus (preliminary version)”. *SIGACT News*, 6(3):15–22, July 1974. 4.7
- [CR74b] Stephen A. Cook and Robert A. Reckhow. 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, 1974. For corrections see Cook-Reckhow [CR74a]. 4.7
- [CR79] Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. *J. Symb. Log.*, 44(1):36–50, 1979. This is a journal-version of Cook-Reckhow [CR74b] and Reckhow [Rec76b]. 1.1, 1.5
- [Gen35] Gerhard Gentzen. Untersuchungen über das logische schließen. *Mathematische Zeitschrift*, 39:68–131, 1935. 1.1

- [GH03] Dima Grigoriev and Edward A. Hirsch. Algebraic proof systems over formulas. *Theoret. Comput. Sci.*, 303(1):83–102, 2003. Logic and complexity in computer science (Créteil, 2001). 1.3.3, 3, 3
- [GP14] Joshua A. Grochow and Toniann Pitassi. 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, 2014. Full version at [arXiv:abs/1404.3820](https://arxiv.org/abs/1404.3820). 1.2, 1.1, 1.2, 1.3.1, 5, 1.3.3, 1.3.3, 1.3.3, 1.3.3, 1.4, 3, 4.2, 4.2, 4.7
- [Has87] Johan Hastå. *Computational limitations of small-depth circuits*. MIT Press, 1987. 1.2
- [Hru11] Pavel Hrubeš. How much commutativity is needed to prove polynomial identities? *Electronic Colloquium on Computational Complexity, ECCC*, (Report no.: TR11-088), June 2011. 4.7
- [HT15] Pavel Hrubeš and Iddo Tzameret. Short proofs for the determinant identities. *SIAM J. Comput.*, 44(2):340–383, 2015. (A preliminary version appeared in Proceedings of the 44th Annual ACM Symposium on the Theory of Computing (STOC’12)). 1.3.3, 4.4, 4.7
- [HW14] Pavel Hrubeš and Avi Wigderson. Non-commutative arithmetic circuits with division. In *Innovations in Theoretical Computer Science, ITCs’14, Princeton, NJ, USA, January 12-14, 2014*, pages 49–66, 2014. 1.3.3, 4.1, 4.1
- [KPW95] Jan Krajíček, Pavel Pudlák, and Alan Woods. An exponential lower bound to the size of bounded depth Frege proofs of the pigeonhole principle. *Random Structures Algorithms*, 7(1):15–39, 1995. 1.2
- [Kra95] Jan Krajíček. *Bounded arithmetic, propositional logic, and complexity theory*, volume 60 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 1995. 1.1, 1.3.3, 3
- [Kra97] Jan Krajíček. Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic. *The Journal of Symbolic Logic*, 62(2):457–486, 1997. 1.2
- [Kra04] Jan Krajíček. Dual weak pigeonhole principle, pseudo-surjective functions, and provability of circuit lower bounds. *The Journal of Symbolic Logic*, 69(1):265–286, 2004. 3
- [Kra08] Jan Krajíček. An exponential lower bound for a constraint propagation proof system based on ordered binary decision diagrams. *J. Symbolic Logic*, 73(1):227–237, 2008. 1.2
- [Kra11] Jan Krajíček. *Forcing with random variables and proof complexity*. London Mathematical Society Lecture Note Series, No.382. Cambridge University Press, 2011. 1.1, 3
- [LT13] Fu Li and Iddo Tzameret. Witnessing matrix identities and proof complexity. *Electronic Colloquium on Computational Complexity, TR13-185*, 2013. [arXiv:1312.6242 \[cs.CC\]](https://arxiv.org/abs/1312.6242) <http://arxiv.org/abs/1312.6242>. 1.1, 4.7
- [MP97] Alexis Maciel and Toniann Pitassi. On $\text{ACC}^0[p^k]$ Frege proofs. In *Proceedings of the Annual ACM Symposium on the Theory of Computing 1997 (El Paso, TX)*, pages 720–729 (electronic). ACM, New York, 1997. 1.2
- [Nis91] Noam Nisan. Lower bounds for non-commutative computation. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing (STOC 1991)*, pages 410–418, 1991. 1.3.1, 1.3.2, 1.3.3, 1.4, 3a, 4.15, 4.7
- [PBI93] Toniann Pitassi, Paul Beame, and Russell Impagliazzo. Exponential lower bounds for the pigeonhole principle. *Comput. Complexity*, 3(2):97–140, 1993. 1.1, 1.2
- [Pit97] Toniann Pitassi. 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, 1997. 1.3.3, 4.7
- [Pit98] Toniann Pitassi. Unsolvability systems of equations and proof complexity. In *Proceedings of the International Congress of Mathematicians, Vol. III (Berlin, 1998)*, number Vol. III, pages 451–460, 1998. 4.7

- [Pud99] Pavel Pudlák. On the complexity of the propositional calculus. In *Sets and proofs (Leeds, 1997)*, volume 258 of *London Math. Soc. Lecture Note Ser.*, pages 197–218. Cambridge Univ. Press, Cambridge, 1999. 1.2
- [Raz85] Alexander A. Razborov. Lower bounds on the monotone complexity of some Boolean functions. *Dokl. Akad. Nauk SSSR*, 281(4):798–801, 1985. In Russian. English translation in *Soviet Math. Doklady*, 31:354–357, 1985. 1.2
- [Raz87a] Alexander A. Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Matematicheskie Zametki*, 41(4):598–607, April 1987. In Russian. English Translation in [Raz87b]. 1.2
- [Raz87b] Alexander A. Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Mathematical notes of the Academy of Sciences of the USSR*, 41(4):333–338, 1987. 4.7
- [Raz06] Ran Raz. Separation of multilinear circuit and formula size. *Theory of Computing, Vol. 2, article 6*, 2006. 1.2
- [Raz09] Ran Raz. Multi-linear formulas for permanent and determinant are of super-polynomial size. *J. ACM*, 56(2), 2009. 1.2
- [Raz13] Ran Raz. Tensor-rank and lower bounds for arithmetic formulas. *J. ACM*, 60(6):40, 2013. 1.3.3, 1, 4, 1, 4.3, 4.4, 4.6, 4.4, 4.4, 4.4, 4.7
- [Raz15] Alexander A. Razborov. Pseudorandom generators hard for k -DNF resolution and polynomial calculus resolution. *Annals of Mathematics*, 181:415–472, 2015. 1.1, 3
- [Rec76a] Robert Reckhow. *On the lengths of proofs in the propositional calculus*. PhD thesis, University of Toronto, 1976. Technical Report No. 87. 1.1, 2.1, 3
- [Rec76b] Robert A. Reckhow. *On the lengths of proofs in the propositional calculus*. PhD thesis, University of Toronto, 1976. 4.7
- [RS05] Ran Raz and Amir Shpilka. Deterministic polynomial identity testing in non commutative models. *Computational Complexity*, 14(1):1–19, 2005. 1.3.3, 1.3.3, 1.3.3, 1.4, 3, 3.2, 4.2, 3, 3a, 3b, 11, 4.6, 4.6.1, 4.12, 4.6.2, 4.14, 4.15, 4.6.2, 4.7
- [RT08a] Ran Raz and Iddo Tzameret. Resolution over linear equations and multilinear proofs. *Ann. Pure Appl. Logic*, 155(3):194–224, 2008. 1.2
- [RT08b] Ran Raz and Iddo Tzameret. The strength of multilinear proofs. *Computational Complexity*, 17(3):407–457, 2008. 1.2
- [Sch80] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, October 1980. Preliminary version in the *International Symposium on Symbolic and Algebraic Computation (EUROSAM 1979)*. 1.3.3
- [Seg07] Nathan Segerlind. Nearly-exponential size lower bounds for symbolic quantifier elimination algorithms and OBDD-based proofs of unsatisfiability. *Electronic Colloquium on Computational Complexity*, January 2007. ECCC, TR07-009. 1.2
- [Smo87] Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC 1987)*, pages 77–82, 1987. 1.2
- [Spi71] Philip M. Spira. On time-hardware complexity tradeoffs for boolean functions. In *Fourth International Symposium on Systems Sciences*, pages 525–527, 1971. 1.1
- [Str73] Volker Strassen. Vermeidung von divisionen. *J. Reine Angew. Math.*, 264:182–202, 1973. (in German). 1.3.3, 4.4
- [TC17] Iddo Tzameret and Stephen A. Cook. Uniform, integral and efficient proofs for the determinant identities. In *Proceedings of the 32nd Annual ACM/IEEE Symposium on Logic In Computer Science (LICS)*, 2017. 4.7

- [Tza08] Iddo Tzameret. *Studies in Algebraic and Propositional Proof Complexity*. PhD thesis, Tel Aviv University, 2008. [1.2](#)
- [Tza11] Iddo Tzameret. Algebraic proofs over noncommutative formulas. *Inf. Comput.*, 209(10):1269–1292, 2011. ([document](#)), [1.2](#), [1.3.3](#), [1.4](#)
- [Val79] Leslie G. Valiant. Completeness classes in algebra. In *Proceedings of the 11th Annual ACM Symposium on the Theory of Computing*, pages 249–261. ACM, 1979. [1.2](#)
- [Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation*, pages 216–226. Springer-Verlag, 1979. [1.3.3](#)