Electronic Colloquium on Computational Complexity, Report No. 134 (2015)

Characterizing Propositional Proofs as Non-Commutative Formulas*

Fu Li† — *Tsinghua University*

Iddo Tzameret‡ — *Royal Holloway, University of London*

Zhengyu Wang§ — *Harvard University*

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 by and large. Non-commutative arithmetic 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 corresponding size lower bounds on non-commutative formulas computing certain polynomials (and that such lower bounds on non-commutative formulas must exist, unless ***NP*=*coNP***). More precisely, we demonstrate a natural association between tautologies T to non-commutative polynomials p, such that:

✶ if T has a polynomial-size Frege proof then p has a polynomial-size non-commutative arithmetic formula; and conversely, when T is a DNF, if p has a polynomial-size non-commutative arithmetic 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 arithmetic 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) arithmetic formulas that is tighter than (the formula version of IPS) in Grochow and Pitassi [FOCS 2014].

*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.

†Institute for Interdisciplinary Information Sciences. Supported in part by NSFC grant 61373002. `fuli.theory.research@gmail.com`

‡Department of Computer Science. Supported in part by NSFC grant 61373002. `Iddo.Tzameret@rhul.ac.uk`

§Department of Computer Science. `wangsincos@163.com`

ISSN 1433-8092

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 $\boldsymbol{NP}$ vs. $\boldsymbol{coNP}$ problem (and thus to the $\boldsymbol{P}$ vs. $\boldsymbol{NP}$ problem).

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 work [Rec76] that all Frege proof-systems[1] (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 $A \to B$ is an abbreviation of $\neg A \vee B$; and A, B, C are any propositional formulas):

$$\begin{gathered} A \to (B \to A) \\ (\neg A \to \neg B) \to ((\neg A \to B) \to A) \\ (A \to (B \to C)) \to ((A \to B) \to (A \to C)), \end{gathered}$$

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

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

Complexity-wise, Frege is considered a very strong proof system 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] 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

[1]Formally, 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).

between a circuit-class $\mathcal{C}$ and a proof system in which every proof-line is written as a circuit[2] from $\mathcal{C}$—Frege system corresponds to the circuit class of polynomial-size $\log(n)$-depth circuits denoted $\boldsymbol{NC}^1$ (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 impeding 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, mainly specialized switching lemmas tailored for specific tautologies (namely, counting tautologies, most notable of which are the Pigeonhole Principle tautologies). Even random kCNF 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 complexity and proof complexity is that circuit hardness sometimes can be used to obtain proof complexity hardness. The most notable example of this are 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. 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. For example, 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

[2]To be more precise, one has to associate a circuit class $\mathcal{C}$ with a proof system in which a *family* of proofs is written such that every proof-line in the family is a circuit family from $\mathcal{C}$.

operating with non-commutative formulas (see Sec. 1.4 for a comparison with the current work).[3]

Until quite recently it was unknown whether the correspondence between proofs and circuits is two-sided, namely, whether proof complexity hardness (of concrete known proof systems) can imply 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 arithmetic circuits and multilinear arithmetic circuits, resp., for an explicit family of polynomials, implies a separation between arithmetic circuits and multilinear arithmetic 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 arithmetic circuits, and formally, that the permanent does not have polynomial-size arithmetic circuits. The IPS is defined as follows:

Definition 1.1 (Ideal Proof System (IPS) [GP14]). *Let $F_1(\overline{x}), \ldots, F_m(\overline{x})$ be a system of polynomials in the variables $x_1, \ldots, x_n$, 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 $C(\overline{x}, \overline{y})$ in the variables $x_1, \ldots, x_n$ and $y_1, \ldots, y_m$, such that:*

1. $F(x_1, \ldots, x_n, \overline{0}) = 0$*; and*
2. $F(x_1, \ldots, x_n, F_1(\overline{x}), \ldots, F_m(\overline{x})) = 1$.

The essence of IPS is that a proof (or refutation) is a *single* polynomial that can be written simply as an arithmetic *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 arithmetic circuit implies that the permanent does not have polynomial-size algebraic circuits (Valiant's conjectured separation $\mathsf{VNP} \neq \mathsf{VP}$ [Val79]); And similarly, a lower bound on IPS written as an arithmetic *formula* implies that the permanent does not have polynomial-size algebraic formulas ($\mathsf{VNP} \neq \mathsf{VP_e}$, 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 arithmetic formulas is decidable by polynomial-size Boolean circuits* (*PIT for arithmetic formulas* is the problem of deciding whether an input arithmetic formula computes the [formal] zero polynomial). They showed that[4], under this assumption super-polynomial lower bounds on Frege proofs imply that the permanent does not have polynomial-size arithmetic 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.

[3] We 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. [ABSRW04, Raz15, Kra04, Kra10].

[4] We 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 $\boldsymbol{NC}^1$ circuits)).

1.3 Overview of results and proofs

1.3.1 Sketch

In this work we give a novel characterization of the propositional calculus—a fundamental and prominent object by itself—and by this contribute to the understanding of strong propositional proof systems, and to the fundamental search for lower bounds on these proofs. We formulate a very natural proof system, namely a non-commutative variant of the ideal proof system, which we show captures *unconditionally* (up to a quasi-polynomial-size increase, and in some cases only a polynomial increase[5]) propositional Frege proofs. A proof in the non-commutative IPS is simply a *single non-commutative polynomial* written as a non-commutative formula.

Our results thus give a compelling and simple new characterization of the proof complexity of propositional Frege proofs and brings new hope for achieving lower bounds on strong proof systems, by reducing the task of lower bounding Frege proofs to the following seemingly much more manageable task: proving matrix rank lower bounds on the matrices associated with certain non-commutative polynomials (in the sense of Nisan [Nis91]; see below for details).

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) arithmetic formulas by polynomial-size circuits is efficiently provable in Frege.

The tighter result shows that, at least for Frege, and in the framework of the ideal proof system, lower bounds on Frege proofs do not necessarily entail in themselves very strong computational lower bounds.

1.3.2 Some preliminaries: non-commutative polynomials and formulas

A *non-commutative polynomial* over a given field $\mathbb{F}$ and with the variables $X := \{x_1, x_2, \ldots\}$ is a formal sum of monomials 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 *non-commutative arithmetic 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 $x_1, \ldots, x_n$, 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

[5]We establish a slightly stronger characterization: the non-commutative IPS polynomially simulates Frege; and conversely, the complexity in which Frege simulates the non-commutative IPS 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).

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 4.13); and then show that the number of nodes in the ith 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.[6] 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).

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 polynomially simulates Frege:

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

Boolean axioms: $x_i \cdot (1 - x_i)\,,$ *for all* $1 \le i \le n\,;$
Commutator axioms: $x_i \cdot x_j - x_j \cdot x_i\,,$ *for all* $1 \le i < j \le n\,.$

Suppose that the $F_i(\overline{x})$'s have no common 0-1 *solutions.*[7] *A **non-commutative IPS refutation** (or* certificate*) that the system of $F_i(\overline{x})$'s is unsatisfiable is a non-commutative polynomial $\mathfrak{F}(\overline{x}, \overline{y})$ in the variables $x_1, \ldots, x_n$ and $y_1, \ldots, y_m$ (i.e. $\mathfrak{F} \in \mathbb{F}\langle\overline{x}, \overline{y}\rangle$), such that:*

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

*We always assume that the non-commutative IPS refutation is written as a non-commutative formula. Hence the **size** of a non-commutative IPS refutation is the minimal size of a non-commutative formula computing the non-commutative IPS refutation.*

Note: (i) 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 $F_1(\overline{x}) = F_2(\overline{x}) = \cdots = F_m(\overline{x}) = 0$ (where the F_i's include the Boolean and commutator axioms) for which there are *no* non-commutative IPS refutations.

(ii) In order to prove that a system of *commutative* polynomial equations $\{P_i = 0\}$ (where each P_i is expressed as an arithmetic 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).

[6] The degree i partial derivative matrix of f is the matrix whose ro‘ws 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.

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

The main result of this paper is that the non-commutative IPS (over either $\mathbb{Q}$ or $\mathbb{Z}_q$, for any prime q) polynomially 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 arithmetic formulas:

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

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

Theorem 1.4 (First main theorem)**.** *Let $\mathbb{F}$ be either the rational numbers $\mathbb{Q}$ or $\mathbb{Z}_q$, for a prime q. The non-commutative IPS refutation system, when refutations are written as non-commutative formulas over $\mathbb{F}$, polynomially simulates the Frege system. More precisely, for every propositional tautology T, if T has a polynomial-size Frege proof then there is a non-commutative IPS certificate (over $\mathbb{F}$) of* $\mathrm{tr}(\neg T)$ *that has a polynomial non-commutative formula size.*

The fact that an arithmetic formula (or circuit) in the form of the IPS can simulate a propositional Frege proof was shown in [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 arithmetic formulas) we now use non-commutative polynomials (written as non-commutative arithmetic formulas). And as mentioned above, in order to maintain the completeness of the non-commutative IPS we must add the commutator axioms $x_ix_j - x_jx_i$ to the system. Thus, the question arises: how can we still polynomially simulate Frege in this restrictive framework? The answer to this, which also constitutes one of the main observation 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.

We now sketch in more detail the proof of Theorem 1.4. To simulate Frege proofs we use an intermediate proof system $\mathcal{F}$-$\mathcal{PC}$ (standing for "formula polynomial calculus") introduced by Grigoriev and Hirsch [GH03]. The $\mathcal{F}$-$\mathcal{PC}$ proof system (Definition 2.8) can be thought of as a simple variant of the well-studied *polynomial calculus* (PC) system in which polynomials are written as arithmetic formulas (instead of sums of monomials as in PC).

Recall that a PC-refutation, as introduced by Clegg, Edmonds and Impagliazzo [CEI96], is simply a sequence of polynomials *written as sum of monomials*, where each polynomial is either taken from the initial unsatisfiable set of polynomials or was derived using two algebraic rules: from a pair of previously derived polynomials f and g, derive $af + bg$ (for a, b field elements); and from a previously derived f, derive $x_i \cdot f$, for any variable x_i. The $\mathcal{F}$-$\mathcal{PC}$ proof system makes the following two changes to PC (turning it into a provably much stronger system):

(i) every polynomial in an $\mathcal{F}$-$\mathcal{PC}$-proof is written as an *arithmetic formula* (instead of as a sum of monomials) and is treated as a purely syntactic object (like in Frege); and

(ii) we can derive new polynomials either by the two aforementioned PC rules, *or by local rewriting rules* operating on any *subformula* and expressing simple operations on polynomials (such as commutativity of addition and product, associativity, distributivity, etc.).

Grigoriev and Hirsch [GH03] showed that $\mathcal{F}$-$\mathcal{PC}$ polynomially simulates Frege proofs, and that for tree-like Frege proofs the polynomial simulation yields tree-like $\mathcal{F}$-$\mathcal{PC}$ proofs. Since tree-like Frege is polynomially equivalent to Frege—because Frege proofs can always be balanced to a depth that is logarithmic in their size (cf. [Kra95] for a proof)—we get that tree-like $\mathcal{F}$-$\mathcal{PC}$ polynomially simulates (dag-like) Frege proofs.

Therefore, to conclude Theorem 1.4 it suffices to prove that the non-commutative IPS polynomially simulates tree-like $\mathcal{F}$-$\mathcal{PC}$ proofs. To do this, loosely speaking, we construct the non-commutative formula tree according to the structure of the tree-like $\mathcal{F}$-$\mathcal{PC}$ proof, line by line.

Now, since we write refutations as non-commutative formulas we can use the polynomial-time deterministic Polynomial Identity Testing (PIT) algorithm for non-commutative formulas, devised by Raz and Shpilka [RS05], to 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 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.

The major 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 polynomials. Specifically, it is enough to prove that any non-commutative IPS certificate $\mathfrak{F}(\overline{x},\overline{y})$ (which is simply a non-commutative polynomial) has a super-polynomial non-commutative formula size; and yet in other words, it suffices to show that any such $\mathfrak{F}$ 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 ($\mathrm{tr}'(f)$ and Q_i^{ϕ})**.** *Given a Boolean formula f we define its non-commutative formula translation* $\mathrm{tr}'(f)$ *as follows. Let* $\mathrm{tr}'(x) := 1 - x$ *and* $\mathrm{tr}'(\neg x) := x$*, for x a variable. Let*

$\mathrm{tr}'(\mathit{false}) := 0$*;* $\mathrm{tr}'(\mathit{true}) := 1$*; and* $\mathrm{tr}'(f_1 \vee \ldots \vee f_r) := \mathrm{tr}'(f_1) \cdots \mathrm{tr}'(f_r)$ *(where the sequence of products stands for a (balanced) fan-in two tree of product gates with* $\mathrm{tr}'(f_i)$ *on the leaves). Further, for a CNF* $\phi = \kappa_1 \wedge \ldots \wedge \kappa_m$*, denote by* Q_i^ϕ *the non-commutative formula translation* $\mathrm{tr}'(\kappa_i)$ *of the clause* κ_i*.*

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

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

Note: 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 $\mathrm{poly}\left(\binom{d+r+1}{r} \cdot s\right)$. 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 (Lemma 4.2 shows that we can always balance non-commutative formulas), and so explicitly we only deal with the case where $d = O(\log s)$ and $r = O(s)$.

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 with 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* $GF(2)$*, as a Boolean tautology*, by replacing plus gates with XORs and product gates with ANDs.

Step 3: reflection principle. We use a reflection principle to reduce the task of efficiently proving $\neg\phi$ in Frege to the following task: show that any non-commutative formula identity over $GF(2)$, considered as a Boolean tautology, has a short Frege proof.

Step 4: homogenization. This is the <u>only</u> 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 arithmetic 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 arithmetic *circuits* by Strassen [Str73] that does allow such a conversion). Nevertheless, Strassen's standard procedure enables us to transform any polynomial-size arithmetic formula into a sum of homogenous formulas with only a *quasi-polynomial* increase in size: any formula of size $\mathrm{poly}(n)$ 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. [HT12]). 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 arithmetic formula into (a sum of) homogenous formulas in a manner which is more efficient than Strassen [Str73]. Specifically, in Lemma 4.8 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 in up to a quasi-polynomial increase in the formulas size)—it remains only to show how to efficiently proof 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. 4.7).

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

Theorem 1.8. *If a non-commutative* homogeneous *formula $F(\overline{x})$ over $GF(2)$ of size s is identically zero, then the corresponding Boolean formula $\neg F_{bool}(\overline{x})$ (where F_{bool} results by replacing $+$ with XOR and $\cdot$ with AND in $F(\overline{x})$) can be proved with a Frege proof of size at most $s^{O(1)}$.*

A more detailed *overview* of the proof (specifically, of the proof of Theorem 4.11) appears in Section 4.4.

1.4 Comparison with 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 almost tightly the Frege system.

In the original (formula version of the) IPS, proofs are arithmetic formulas, and thus any super-polynomial lower bound on IPS refutations implies $\mathsf{VNP} \neq \mathsf{VP_e}$, or in other words, that the permanent does not have polynomial-size arithmetic 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. In view of this, it seems that there is no strong concrete justification to believe that Frege lower bounds are beyond current techniques.

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).

2 Preliminaries

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

Definition 2.1 (Boolean formulas). *Given a set of input variables $\{x_1, x_2, \ldots\}$ 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*

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

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 $(A \to B) \wedge (B \to A)$.

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. The *size* of a Frege proof is the number of symbols it takes to write down the proof, namely the total of all the formula sizes appearing in the proof. Let us define Frege proofs in a more formal way.

Definition 2.2 (Frege (derivation) rule)**.** *A* Frege rule *is a sequence of propositional formulas $A_0(\overline{x}), \dots, A_k(\overline{x})$, for $k \le 0$, written as $\frac{A_1(\overline{x}),\dots,A_k(\overline{x})}{A_0(\overline{x})}$. 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 $\overline{x}$ 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 said to be *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 [Rec76] 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, noting 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 $n \in \mathbb{Z}^+$.

2.2 Preliminary algebraic models of computation and proofs

Here we define arithmetic formulas (both commutative and non-commutative) as well as the algebraic propositional proof system Polynomial Calculus over Formulas ($\mathcal{F}$-$\mathcal{PC}$) introduced by Grigoriev and Hirsch [GH03].

Definition 2.5 (Non-commutative formula). *Let $\mathbb{F}$ be a field and $\{x_1, x_2, \ldots\}$ be (algebraic) variables. A* non-commutative arithmetic 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 $\times$ (in the first case the node is a plus gate and in the second case a non-commutative product gate). 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 $\mathbb{F}\langle x_1, \ldots, x_n\rangle$ 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) arithmetic formula is almost identical:

Definition 2.6 ((Commutative) arithmetic formula). *An* arithmetic 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.

Note that we consider arithmetic formulas as syntactic objects. For example, $x_1 + x_2$ and $x_2 + x_1$ are different formulas. Furthermore, in the proof system $\mathcal{F}$-$\mathcal{PC}$ defined below they should be *derived* from each other via an explicit application of a rewrite rule.

2.2.1 Polynomial calculus over formulas $\mathcal{F}$-$\mathcal{PC}$

The *polynomial calculus over formulas* system, denoted $\mathcal{F}$-$\mathcal{PC}$, was introduced by Grigoriev and Hirsch [GH03]. This system operates with (commutative) arithmetic formulas (as purely syntactic terms). $\mathcal{F}$-$\mathcal{PC}$ is a refutation system: an $\mathcal{F}$-$\mathcal{PC}$ refutation establishes that a collection of polynomials has no 0-1 roots. We can also treat $\mathcal{F}$-$\mathcal{PC}$ as a *proof* system for propositional tautologies: for every Boolean tautology T, $\mathrm{tr}(\neg T)$ (Definition 1.3) is a polynomial that does not have a 0-1 root, and therefore, an $\mathcal{F}$-$\mathcal{PC}$ refutation of $\mathrm{tr}(\neg T)$ can be considered as an $\mathcal{F}$-$\mathcal{PC}$ *proof of the tautology T.*

Definition 2.7 (Rewrite rule). *A* rewrite rule *is a pair of formulas f, g denoted $f \to g$. Given a formula Φ, an* application of a rewrite rule $f \to g$ to Φ *is the result of replacing at most one* *occurrence of f in Φ by g (that is, substituting a subformula f inside Φ by the formula g). We write $f \leftrightarrow g$ to denote the pair of rewriting rules $f \to g$ and $g \to f$.*

Definition 2.8 ($\mathcal{F}$-$\mathcal{PC}$ [GH03])**.** *Fix a field $\mathbb{F}$. Let $F := \{f_1, \dots, f_m\}$ be a collection of* formulas[9] *computing polynomials from $\mathbb{F}[x_1, \dots, x_n]$. Let the set of axioms be the following formulas:*

Boolean axioms $\quad x_i \cdot (1 - x_i)\,, \qquad$ *for all* $1 \le i \le n\,.$

A sequence $\pi = (\Phi_1, \dots, \Phi_\ell)$ of formulas computing polynomials from $\mathbb{F}[x_1, \dots, x_n]$ is said to be ***an $\mathcal{F}$-$\mathcal{PC}$ proof of Φ_ℓ from*** *F, if for every $i \in [\ell]$ we have one of the following:*

1. *$\Phi_i = f_j$, for some $j \in [m]$;*
2. *Φ_i is a Boolean axiom;*
3. *Φ_i was deduced by one of the following inference rules from previous proof-lines Φ_j, Φ_k , for $j, k < i$:*

   Product
   $$\frac{\Phi}{x_r \cdot \Phi}\ , \qquad \textit{for } r \in [n]\,.$$

   Addition
   $$\frac{\Phi \qquad \Theta}{a \cdot \Phi + b \cdot \Theta}\ , \qquad \textit{for } a, b \in \mathbb{F}\,.$$

   (Where $\Phi, x_r \cdot \Phi, \Theta, a \cdot \Phi, b \cdot \Theta$ are formulas *constructed as displayed; e.g., $x_r \cdot \Phi$ is the formula with product gate at the root having the formulas x_r and Φ as children.)*[10]

4. *Φ_i was deduced from previous proof-line Φ_j, for $j < i$, by one of the following* rewriting rules *expressing the polynomial-ring axioms (where f, g, h range over all arithmetic formulas computing polynomials in $\mathbb{F}[x_1, \dots, x_n]$):*

   Zero rule $0 \cdot f \leftrightarrow 0$

   Unit rule $1 \cdot f \leftrightarrow f$

   Scalar rule *$t \leftrightarrow \alpha$, where t is a formula containing no variables (only field $\mathbb{F}$ elements) that computes the constant $\alpha \in \mathbb{F}$.*

   Commutativity rules $f + g \leftrightarrow g + f\,, \qquad f \cdot g \leftrightarrow g \cdot f$

   Associativity rule $f + (g + h) \leftrightarrow (f + g) + h\,, \qquad f \cdot (g \cdot h) \leftrightarrow (f \cdot g) \cdot h$

   Distributivity rule $f \cdot (g + h) \leftrightarrow (f \cdot g) + (f \cdot h)$

(The semantics of an $\mathcal{F}$-$\mathcal{PC}$ proof-line p_i is the polynomial equation $p_i = 0$.)

An $\mathcal{F}$-$\mathcal{PC}$ refutation of *F is a proof of the formula 1 from F. The* ***size*** *of an $\mathcal{F}$-$\mathcal{PC}$ proof π is defined as the total size of all formulas in π and is denoted by $|\pi|$.*

[9]Note here that we are talking about formulas (treated as syntactic terms). Also notice that all the formulas in $\mathcal{F}$-$\mathcal{PC}$ are considered as commutative formulas computing (commutative) polynomials, though, because the formulas are merely syntactic terms we have an order on children of internal nodes, and in particular children of product gates are ordered.

Definition 2.9 (Tree-like $\mathcal{F}$-$\mathcal{PC}$)**.** *A system $\mathcal{F}$-$\mathcal{PC}$ is a* tree-like $\mathcal{F}$-$\mathcal{PC}$ *if every derived arithmetic formula in the proof system is used only once (and if it is needed again, it must be derived once more).*

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 *polynomially simulates* another propositional proof system Q if there is a polynomial-time computable function f that maps Q-proofs to P-proofs of the same tautologies (if P and Q use different representations for tautologies, we fix a translation (such as $\mathrm{tr}(\cdot)$) 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 polynomially simulates Q and Q polynomially 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.)

Tree-like $\mathcal{F}$-$\mathcal{PC}$ polynomially simulates Frege. Grigoriev and Hirsch showed the following:

Theorem 2.10 ([GH03])**.** *Tree-like $\mathcal{F}$-$\mathcal{PC}$ polynomially simulates Frege. More precisely, for every propositional tautology T, if T has a polynomial-size Frege proof then there is a polynomial-size tree-like $\mathcal{F}$-$\mathcal{PC}$ proof of* $\mathrm{tr}(\neg T)$ *(over $\mathbb{Z}_q$, for q a prime, or $\mathbb{Q}$).*

Let us shortly explain how Grigoriev and Hirsch [GH03] obtained a simulation of Frege by *tree-like $\mathcal{F}$-$\mathcal{PC}$* (in contrast to simply (dag-like) $\mathcal{F}$-$\mathcal{PC}$), as this is not an entirely trivial result (and which, in turn, is important to understand our simulation). Indeed, this simulation depends crucially on a somewhat surprising result of Krajíček who showed that tree-like Frege and (dag-like) Frege are *polynomially equivalent* [Kra95]:

Theorem ([Kra95])**.** *Tree-like Frege proofs polynomially simulate Frege proofs.*

Grigoriev and Hirsch show that (Theorem 3 in [GH03]) $\mathcal{F}$-$\mathcal{PC}$ polynomially simulates Frege. Then, by inspection of this simulation, one can observe that tree-like Frege proofs are simulated by tree-like $\mathcal{F}$-$\mathcal{PC}$ proofs (which is sufficient to conclude the simulation due to the theorem above), namely:

Lemma ([GH03])**.** *Tree-like $\mathcal{F}$-$\mathcal{PC}$ polynomially simulates tree-like Frege.*

3 Non-commutative ideal proof system polynomially simulates Frege

Here we show that the non-commutative IPS polynomially simulates Frege.

Theorem 3.1 (restatement of Theorem 1.4)**.** *The non-commutative IPS refutation system (when refutations are written as non-commutative formulas) polynomially simulates the Frege system.*

[10]In [GH03] the product rule of $\mathcal{F}$-$\mathcal{PC}$ is defined so that one can derive $\Theta \cdot \Phi$ from Φ, where Θ is any formula, and not just a variable. However, it is easy to show that the definition of $\mathcal{F}$-$\mathcal{PC}$ in [GH03] and our Definition 2.8 polynomially-simulate each other.

More precisely, for every propositional tautology T, if T has a polynomial-size Frege proof then there is a non-commutative IPS refutation of $\mathrm{tr}(\neg T)$ *(over $\mathbb{Z}_p$ for a prime p, or $\mathbb{Q}$) of polynomial size.*

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* 0.[11]

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, obtaining:

Corollary 3.3 (restatement of Corollary 1.5)**.** *The non-commutative IPS is a sound and complete Cook-Reckhow refutation system. That is, it is a sound and complete refutation system for unsatisfiable propositional formulas in which refutations can be checked for correctness in deterministic polynomial-time.*

To prove Theorem 3.1, we will show in Section 3.1 that the non-commutative IPS polynomially-simulates tree-like $\mathcal{F}$-$\mathcal{PC}$ (Definition 2.8), which sufficed to complete the proof, due to Theorem 2.10.

3.1 Non-commutative IPS polynomially simulates tree-like $\mathcal{F}$-$\mathcal{PC}$

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. When we write $P \cdot Q - Q \cdot P$ where P, Q are formulas (e.g., x_i and x_j, resp.), we mean $((P \cdot Q) + (-1 \cdot (Q \cdot P)))$.

Theorem 3.4. *Non-commutative IPS polynomially simulates tree-like $\mathcal{F}$-$\mathcal{PC}$ (Definition 2.8). Specifically, if π is a tree-like $\mathcal{F}$-$\mathcal{PC}$ proof of a tautology T then there is a non-commutative IPS refutation of* $\mathrm{tr}(\neg T)$ *of size polynomial in $|\pi|$.*

Proof. Let $F_1, \dots, F_m$ be arithmetic formulas over the variables $x_1, \dots, x_n$. We denote by $\mathbf{F}$ the vector $(F_1, \dots, F_m)$. Since an arithmetic formula is a syntactic term in which the children of gates are ordered we can treat a (commutative) arithmetic formula as a *non-commutative* arithmetic formula by taking the *order* on the children of products gates to be the order of non-commutative multiplication.

Suppose $\mathcal{F}$-$\mathcal{PC}$ has a poly(n)-size tree-like refutation $\pi := (L_1, \dots, L_k)$ of the F_i's (i.e., a proof of the polynomial 1 from $F_1, \dots, F_m$), where each L_j is an arithmetic formula. We construct a corresponding non-commutative IPS refutation of the F_i's from this $\mathcal{F}$-$\mathcal{PC}$ tree-like refutation. The following lemma suffices for this purpose:

Lemma 3.5. *For every $i \in [k]$, there exists a non-commutative formula ϕ_i such that*

1. $\phi_i(\overline{x}, \overline{0}) = 0$;

[11] We 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).

2. *$\phi_i(\overline{x}, \mathbf{F}, \mathbf{C}) = L_i$;*

3. *$|\phi_i| \le \left(\sum_{\ell \in A_i} |L_\ell|\right)^4$, where $A_i \subset [k]$ are the indices of the $\mathcal{F}$-$\mathcal{PC}$ 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.

Note that if the lemma holds, then ϕ_k is a non-commutative IPS proof because it has the property that $\phi_k(\overline{x}, \overline{0}) = 0$ and $\phi_k(\overline{x}, \mathbf{F}, \mathbf{C}) = L_k = 1$. And its size is bounded by $\left(\sum_{\ell \in A_k} |L_\ell|\right)^4 \le \left(\sum_{\ell \in [k]} |L_\ell|\right)^4 \le O(|\pi|^4)$.

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

Base case: L_i is an axiom F_j for some $j \in [m]$.
Let $\phi_i := y_j$. Obviously, $\phi_i(\overline{x}, 0) = 0, \phi_i(\overline{x}, \mathbf{F}, \mathbf{C}) = F_j = L_i$ and $|\phi_i| = 1 \le |L_i|^4$.

Induction step:
Case 1: L_i is derived from the addition rule $L_i = aL_j + bL_{j'}$, for $j, j' < i$. Put $\phi_i := a\phi_j + b\phi_{j'}$ where $a, b \in \mathbb{F}$. Thus, $\phi_i(\overline{x}, 0) = a\phi_j(\overline{x}, 0) + b\phi_{j'}(\overline{x}, 0) = 0$, $\phi_i(\overline{x}, \mathbf{F}, \mathbf{C}) = aL_j + bL_{j'} = L_i$ and $|\phi_i| = |\phi_j| + |\phi_{j'}| + 3 \le \left(\sum_{\ell \in A_j} |L_\ell|\right)^4 + \left(\sum_{\ell \in A_{j'}} |L_\ell|\right)^4 + 3 \le \left(\sum_{\ell \in A_i} |L_\ell|\right)^4$ (where the right most inequality holds since π is a *tree-like* refutation and hence $A_j \cap A_{j'} = \emptyset$).

Case 2: L_i is derived from the product rule $L_i = x_r \cdot L_j$, for $r \in [n]$ and $j < i$. Put $\phi_i := (x_r \cdot \phi_j)$. Then $\phi_i(\overline{x}, 0) = x_r \cdot \phi_j(\overline{x}, 0) = 0, \phi_i(\overline{x}, \mathbf{F}, \mathbf{C}) = x_r \cdot L_j = L_i$ and $|\phi_i| = |\phi_j| + 2 \le \left(\sum_{\ell \in A_j} |L_\ell|\right)^4 + 2 \le \left(\sum_{\ell \in A_i} |L_\ell|\right)^4$.

Case 3: L_i is derived from L_j, for $j < i$, by a *rewriting* rule which is not the commutative rule of multiplication ($f \cdot g \leftrightarrow g \cdot f$). Let $\phi_i := \phi_j$. The non-commutative ϕ_i trivially satisfies the properties claimed since all the rewriting rules (excluding the commutative rule of multiplication) express the non-commutative polynomial-ring axioms, and thus cannot change the polynomial computed by a non-commutative formula. And $|\phi_i| = |\phi_j| \le \left(\sum_{\ell \in A_i} |L_\ell|\right)^4$.

Case 4: L_i is derived from L_j, for $j < i$, by a single application of the commutative rule of multiplication. Then by Lemma 3.6 below, we can construct a non-commutative formula ϕ_{L_i, L_j} such that $\phi_i := (\phi_j + \phi_{L_i, L_j})$ satisfies the desired properties (stated in Lemma 3.5).

□

Lemma 3.6. *Let L_i, L_j be non-commutative formulas, such that L_i can be derived from L_j via the commutative rule of multiplication $f \cdot g \leftrightarrow g \cdot f$. Then there is a non-commutative formula $\phi_{L_i, L_j}(\overline{x}, \overline{y})$ in variables $\{x_\ell, y_{\alpha,\beta},\ \ell \in [n], \alpha < \beta \in [n]\}$, such that:*

1. *$\phi_{L_i, L_j}(\overline{x}, \overline{0}) = 0$;*

2. $\phi_{L_i,L_j}(\overline{x}, \mathbf{C}) = L_i - L_j$;

3. $\left|\phi_{L_i,L_j}\right| \leq |L_i|^2 \, |L_j|^2$.

Proof. We define the non-commutative formula ϕ_{L_i,L_j} inductively as follows:

- If $L_i = (P \cdot Q)$, and $L_j = (Q \cdot P)$, then ϕ_{L_i,L_j} is defined to be the formula constructed in Lemma 3.7 below.
- If $L_i = (P \cdot Q)$, $L_j = (P' \cdot Q')$.
  Case 1: If $P = P'$, then let $\phi_{L_i,L_j} := (P \cdot \phi_{Q,Q'})$.
  Case 2: If $Q = Q'$, then let $\phi_{L_i,L_j} := (\phi_{P,P'} \cdot Q)$.
- If $L_i = (P + Q)$, $L_j = (P' + Q')$.
  Case 1: If $P = P'$, then let $\phi_{L_i,L_j} = \phi_{Q,Q'}$.
  Case 2: If $Q = Q'$, then let $\phi_{L_i,L_j} = \phi_{P,P'}$.

By induction, the construction satisfies the desired properties. □

Lemma 3.7. *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. $F(\overline{x}, \overline{0}) = 0$;
2. $F(\overline{x}, \mathbf{C}) = P \cdot Q - Q \cdot P$;
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 for 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 $s(P, Q) = |P_1| + 1 + s(P_2, Q) + 1 + s(P_1, Q) + 1 + |P_2| \leq |P_1| + 1 + |P_2|^2 \, |Q|^2 + 1 + |P_1|^2 \, |Q|^2 + 1 + |P_2| \leq (|P_1| + |P_2| + 1)^2 \, |Q|^2 = |P|^2 \cdot |Q|^2$. □

4 Frege quasi-polynomially simulates non-commutative IPS

In this long section we prove Theorem 4.1 stating that the Frege system quasi-polynomially simulates the non-commutative IPS (over $GF(2)$). Together with Theorem 3.1, this gives a new characterization (up to a quasi-polynomial increase in size) of propositional Frege proofs as non-commutative arithmetic formulas.

We use the notation in Section 1.3.3 as follows: for a clause κ_i in a CNF $\phi = \kappa_1 \wedge \ldots \wedge \kappa_m$, we denote by Q_i^ϕ the non-commutative formula translation $\operatorname{tr}'(\kappa_i)$ of the clause κ_i (Definition 1.6). Thus, $\neg x$ translates to x, x translates to $1 - x$ and $f_1 \cdots f_r$ translates to $\prod_i \operatorname{tr}'(f_i)$ (considered as a tree of product gates with $\operatorname{tr}'(f_i)$ 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.

Theorem 4.1 (Second main theorem; Restatement of Theorem 1.7)**.** *For a 3CNF $\phi = \kappa_1 \wedge \ldots \wedge \kappa_m$ where $Q_1^\phi, \ldots, Q_m^\phi$ are the corresponding polynomial equations for the clauses, if there is a non-commutative IPS refutation of size s of $Q_1^\phi, \ldots, Q_m^\phi$ over $GF(2)$, then there is a Frege proof of size $s^{O(\log s)}$ of $\neg\phi$.*

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 $\operatorname{poly}\left(\binom{d+r+1}{r} \cdot s\right)$. This will follow from our efficient simulation within Frege of Raz' [Raz13] homogenization construction (Lemma 4.8). 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 4.1.

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).)

Lemma 4.2. *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 with the root being 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 $\mathbb{F}\langle\overline{X}\rangle$ 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. □

As a consequence of Lemma 4.2, 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. As mentioned in the introduction, informally, 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:

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

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 $GF(2)$ and let $\overline{Q}^{\phi}(\overline{x})$ denote the vector $(Q_1^{\phi}, \dots, Q_m^{\phi})$ (see Theorem 4.1). Since F is a non-commutative IPS refutation of ϕ we know that

$$F\left(\overline{x}, \overline{0}\right) = 0, \qquad F\left(\overline{x}, \overline{Q}^{\phi}(\overline{x})\right) = 1\,. \tag{1}$$

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

Definition 4.3 (F_{bool})**.** *Let $F(\overline{x})$ be a non-commutative formula over $GF(2)$ in the (algebraic) variables $\overline{x}$. We denote by $F_{bool}(\overline{p})$ the Boolean formula in the (propositional) variables $\overline{p}$ 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 $\overline{x}$ into the propositional variables $\overline{p}$. We sometimes write F and F_{bool} without explicitly mentioning the $\overline{x}$ and $\overline{p}$ 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 $F = F(\overline{x}, \overline{y})$ (with both the $\overline{x}$ and $\overline{y}$ variables), F_{bool} denotes the corresponding Boolean version of F where the variables $\overline{x}$ are replaced by $\overline{p}$ and the algebraic variables $\overline{y}$ become the propositional variables $\overline{y}$. Therefore, by (1),

$$\neg F_{bool}\left(\overline{p}, \overline{0}\right) \qquad \text{and} \qquad F_{bool}\left(\overline{p}, \overline{Q}^{\phi}_{bool}(\overline{p})\right) \tag{2}$$

are both *tautologies* (though we still need to show that their Frege proofs are *short*). To conclude Theorem 4.1, we first prove $\neg\phi$ with a polynomial-size Frege proof, assuming we already proved

(2) (this is done in Lemma 4.4 below, which is not very hard to establish). Second, we show that there exists an $s^{O(\log s)}$ Frege proof of (2) (which is done in Theorem 4.7 in the next section, and requires much more work).

Lemma 4.4. *There is a polynomial-size Frege proof of $\neg\phi(\overline{p})$, assuming $\neg F_{bool}\left(\overline{p},\overline{0}\right)$ and $F_{bool}\left(\overline{p},\overline{Q}^{\phi}_{bool}(\overline{p})\right)$ (polynomial in the size of $\phi(\overline{p})$ and $F_{bool}\left(\overline{p},\overline{Q}^{\phi}_{bool}(\overline{p})\right)$).*

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

In order to go from $\phi(\overline{p})$ to $Q^{\phi}_{i_{bool}}(\overline{p}) \equiv 0$ we need to deal with encoding of clauses inside Frege. Thus, let the propositional formula $\mathsf{Truth}([\phi],\overline{p})$ express the statement that the assignment $\overline{p}$ satisfies the formula ϕ, as defined below. In the following we denote by $\overline{p}$ *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 Section 4.3 in [GP14]. A positive natural number i is encoded with $\lceil \log_2 n \rceil$ bits (such that the numbers $1,\ldots,2^t$ 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 $\overline{q}_1 s_1 \overline{q}_2 s_2 \overline{q}_3 s_3$, 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 $\overline{q}_1,\overline{q}_2,\overline{q}_3$ 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 $\overline{q}$ with the ith bit y_i we write $\overline{q} = [t]$ 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)

$$\mathsf{Truth}([\kappa],\overline{p}) := \bigvee_{j\in[3]} \bigvee_{i\in[n]} \left([\overline{q}_j] = i \wedge (p_i \leftrightarrow s_j)\right) , \qquad \text{and}$$

$$\mathsf{Truth}([\phi],\overline{p}) := \bigwedge_{j\in[m]} \mathsf{Truth}([\kappa_j],\overline{p}) \,.$$

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

Let us now continue the proof of Lemma 4.4. 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 $\mathsf{Truth}([\kappa],\overline{p})$ becomes syntactically identical to $\kappa(\overline{p})$ and thus there is a polynomial-size Frege proof of

$$\phi(\overline{p}) \to \mathsf{Truth}([\phi],\overline{p}). \tag{3}$$

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

$$\mathsf{Truth}([\kappa_i],\overline{p}) \to \neg Q^{\phi}_{i_{bool}}(\overline{p}), \qquad \text{for all } i \in [m] \,. \tag{4}$$

By definition (recall that $Q^{\phi}_i = 0$ iff κ_i is true for any 0-1 assignment, when 1 is identified with true) all the formulas in (4) are tautologies. Note that all the premises and all the consequences in (4) are of constant size, and thus (4) can be proved with a Frege proof of constant-size for each

$i \in [m]$ (by completeness). Further, using the fact that $\mathsf{Truth}([\phi], \overline{p}) = \bigwedge_{i\in[m]} \mathsf{Truth}([\kappa_i], \overline{p})$, we can easily prove in Frege with a polynomial-size proof that for each $i \in [m]$,

$$\mathsf{Truth}([\phi], \overline{p}) \to \neg Q^{\phi}_{i_{bool}}(\overline{p}). \tag{5}$$

Assume by a way of contradiction that $\phi(\overline{p})$ holds. By modus ponens using (3) and (5), we have

$$\bigwedge_{i\in[m]} \neg Q^{\phi}_{i_{bool}}(\overline{p}). \tag{6}$$

We now argue (inside Frege) that, assuming also $\neg F_{bool}\left(\overline{p}, \overline{0}\right)$, (6) implies $\neg F_{bool}\left(\overline{p}, \overline{Q}^{\phi}_{bool}(\overline{p})\right)$. By (6), for every $i \in [m]$, $Q^{\phi}_{i_{bool}}(\overline{p})$ is logically equivalent to 0 (which is identified with false), and hence $\neg F_{bool}\left(\overline{p}, \overline{Q}^{\phi}_{bool}(\overline{p})\right) \equiv \neg F_{bool}\left(\overline{p}, \overline{0}\right)$. Thus, assuming $\neg F_{bool}\left(\overline{p}, \overline{0}\right)$ we have also $\neg F_{bool}\left(\overline{p}, \overline{Q}^{\phi}_{bool}(\overline{p})\right)$. But this contradicts the assumption that $F_{bool}\left(\overline{p}, \overline{Q}^{\phi}_{bool}(\overline{p})\right)$, and hence we reach a contradiction with the assumption that $\phi(\overline{p})$ holds. □

It remains to show a quasi-polynomial-size proof of (2). We abbreviate $\neg F_{bool}\left(\overline{p}, \overline{0}\right)$ and $\neg\left(1 \oplus F_{bool}\left(\overline{p}, \overline{Q}^{\phi}_{bool}(\overline{p})\right)\right)$ by

$$F'_{bool}(\overline{p}), \quad F''_{bool}(\overline{p}), \quad \text{respectively.} \tag{7}$$

Note that the substitutions of the constants 0 or the constant depth formulas Q^{ϕ}_{bool} 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 (7) are still $O(\log s)$.

Proof of Theorem 4.1 (Second main theorem). Using Theorem 4.7 that we prove below, we get that (7) can be proved in quasi-polynomial-size (in s the size of the IPS refutation of the given CNF ϕ). And together with Lemma 4.4 above, this shows that $\neg\phi$ can be proved in quasi-polynomial-size in s, concluding the proof. □

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 $0 \le j \le d$, 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 (see Definition 2.5 for the definition of a polynomial computed by a gate in a formula). We shall use the following technical definitions:

Definition 4.5 (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 4.6 (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 4.1 it remains to prove the following theorem:

Theorem 4.7. *If a non-commutative formula $F(\overline{x})$ of size s and depth $O(\log s)$ computes the identically zero polynomial over $GF(2)$, then the corresponding Boolean formula $\neg F_{bool}(\overline{p})$ admits a Frege proof of size $s^{O(\log s)}$.*

The rest of the paper is dedicated to proving this theorem.

4.4 Remaining proof overview

For the convenience of the reader we highlight here in an informal manner the main steps in the proof of Theorem 4.7 stating that the Boolean versions of balanced 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 partitioned into its Booleanized syntactic homogenous components $F_{bool}(i)$, each of quasi-polynomial size in $|F_{bool}|$, using Raz' [Raz13] construction.

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

2. If $F^{(i)}_{bool}$ does not contain variables it is easy to refute $F^{(i)}_{bool}$. Note that $F^{(i)}_{bool}$ does not contain variables iff $F^{(i)}$ does not contain variables. So we can assume that $F^{(i)}$ is non-constant. In this case we show that Frege can easily prove that $F^{(i)}$ is equivalent to some $F'^{(i)}_{bool}$, where $F'^{(i)}$ is a *constant-free* (namely, it does not contain constants) arithmetic non-commutative syntactic homogenous formula computing the zero polynomial.

3. Having a constant-free arithmetic 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 follows:

   (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 4.13) 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 $\mathbf{v} \cdot \overline{A}_{d-i}$ (where $\cdot$ is the inner product) of the polynomials computed by the ABPs whose sources are in the ith 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 $\mathbf{T}_1, \ldots, \mathbf{T}_d$ 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 lth node in layer i to the kth 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 4.7. The formula F is of size s which means that the maximal degree of a polynomial computed by F is at most $s+1$. Raz [Raz13] showed that we can always split F into syntactic homogenous formulas $F^{(i)}$, $i = 0, \ldots, s+1$, each of size $s^{O(\log s)}$. In Lemma 4.8, 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

$$\bigoplus_{i=0}^{s+1} F^{(i)}_{bool} \leftrightarrow F_{bool}. \tag{8}$$

By Theorem 4.11 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 4.11, for every $F^{(i)}$, $i = 0, \ldots, s+1$, there exists an $s^{O(\log s)}$-size Frege proof of $\neg F^{(i)}_{bool}$. That is, there exists an $s^{O(\log s)}$-size Frege proof of $\neg\left(\bigoplus_{i=0}^{s+1} F^{(i)}_{bool}\right)$. Note that Theorem 4.11 gives proofs that have size polynomial in the size of $\neg F^{(i)}_{bool}$, and this latter size is $s^{O(\log s)}$. Together with tautology (8), we can derive $\neg F_{bool}$ in Frege. □

4.5 Proving the homogenization of non-commutative formulas in Frege

To complete the proof of Theorem 4.7 it remains to prove Lemmas 4.8 and 4.11. Lemma 4.8 states that Raz' construction from [Raz13] for homogenizing arithmetic formulas is efficiently provable in Frege (and is also applicable to non-commutative formulas):

Lemma 4.8. *If F is a non-commutative formula of size s and depth $O(\log s)$ and $F^{(0)}, \ldots, F^{(s+1)}$ are the syntactic homogenous formulas computing F's homogenous parts of degrees $0, \ldots, s+1$, respectively, constructed according to [Raz13] (sketched below), then there exists an $s^{O(\log s)}$-size*

Frege proof of:

$$\left(\bigoplus_{i=0}^{s+1} F^{(i)}\right) \leftrightarrow F_{bool}. \tag{9}$$

Proof. We first introduce basic notations and observations for describing Raz' (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.

Raz' formula homogenization construction. Given a balanced (commutative) arithmetic 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 $O(\log s)$.

Let us 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 from product nodes along the directed path to the corresponding degree of a monomial as it is computed along the path, where the path starts from the root (identified with 0) and terminates with the product gate closest to u (including possibly u itself; identified with r). Thus, for instance, the root 0 is mapped to the total degree of the monomial. 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.

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

$$\binom{s + O(\log s) + 2}{s} = s^{O(\log s)}. \tag{10}$$

We construct the desired syntactic homogenous formulas $F^{(0)}, F^{(1)}, \dots, F^{(s+1)}$ by constructing a formula $F^\star$ according to F. Split every gate u in F into $|N_{u_{pd}}|$ gates in $F^\star$, labeled (u, D^u), for every $D^u \in N_{u_{pd}}$. We will add edges connecting nodes in $F^\star$ the same way as [Raz13]. It might be helpful for the reader to consult [Raz13] to get 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 $F^\star$ and F compute the same polynomial, when considered as Boolean functions).

Denote by $F^\star_{u,D^u}$ the subformula rooted at (u, D^u) in $F^\star$. There might be some isolated nodes (u, D^u), namely nodes that no edge connects to them, and we consider the subformulas on these nodes as 0. Similarly, denote by F_u the subformula rooted at u in F. In [Raz13] it was demonstrated (see below) how to construct $F^\star$ so that for every node (u, D^u) in $F^\star$, $F^\star_{u,D^u}$ is a homogenous formula computing the degree-$D^u(u_{pd})$ homogenous part of F_u. More precisely, for every node u in F, denote by s_u the size of the formula F_u. The maximal degree of the polynomial computed by F_u is $s_u + 1$. For $i = 0, \ldots, s_u + 1$, let $\mathcal{D}^u_i$ denote the *set* of all functions D^u in $N_{u_{pd}}$ such that $D^u(u_{pd}) = i$. For *any two* $D, D' \in \mathcal{D}^u_i$, the formulas $F^\star_{u,D}$ and $F^\star_{u,D'}$ are *identical* (and compute the homogenous part of degree i of F_u). Thus we can consider $F^\star_{u,\mathcal{D}^u_i}$ as a *single* formula.

Efficient proofs of the homogenization construction. Next, we use a similar inductive argument as in [Raz13], from leaves to the top gate of F, showing that for every gate u in F there exists an $s^{O(\log s)}$-size Frege proof of

$$\left(\bigoplus_{i=0}^{s_u+1} F^\star_{u,\mathcal{D}^u_i\ bool}\right) \leftrightarrow F_{u\ bool}\,. \tag{11}$$

Observe that the formulas $F^\star_{r,\mathcal{D}^r_0}, \ldots, F^\star_{r,\mathcal{D}^r_{s+1}}$, for r being the root of F, are just those desired formulas $F^{(0)}, F^{(1)}, \ldots, F^{(s+1)}$. Thus, eventually, when we prove (11) for the root node r, we prove the existence of an $s^{O(\log s)}$-size Frege proof of the Boolean formulas in (9).

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 (11). Recall that for every u in F, $|N_{u_{pd}}| = s^{O(\log s)}$, by (10). 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 (11) we construct. Each such proof of (11) requires only poly(s) size *assuming* we already have proved (by induction hypothesis) the required previous instantiations of (11). Therefore, we end up with a proof of total size poly$(s) \cdot s^{O(\log s)} = s^{O(\log s)}$.

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

$$\left(\bigoplus_{i=0}^{s_u+1} F^\star_{u,\mathcal{D}^u_i\ bool}\right) \leftrightarrow F_{u\ bool}.$$

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 $\{0, 1, \ldots, u_{pd}\}$ 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 $\{0, 1, \ldots, u_{pd}\}$ and satisfies $D^w(w_{pd}) = D^u(u_{pd})$. The node (u, D^u) is defined as

$$F^\star_{u,D^u} := F^\star_{v,D^v} + F^\star_{w,D^w}.$$

Assume $D^u(u_{pd}) = j$. Then it means

$$\hat{F}^\star{}_{u,\mathcal{D}^u_j} := \hat{F}^\star{}_{v,\mathcal{D}^v_j} + \hat{F}^\star{}_{w,\mathcal{D}^w_j}$$

(recall that given a non-commutative formula F, $\hat{F}$ denotes the non-commutative *polynomial* it computes). Therefore, the following is a tautology:

$$F^\star_{u,\mathcal{D}^u_j\ bool} \leftrightarrow \left(F^\star_{v,\mathcal{D}^v_j\ bool} \oplus F^\star_{w,\mathcal{D}^w_j\ bool}\right), \quad \text{for all } j = 0, \ldots, s+1.$$

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

$$F_{v\ bool} \leftrightarrow \bigoplus_{i=0}^{s_v+1} F^\star_{v,\mathcal{D}^v_i\ bool}, \qquad F_{w\ bool} \leftrightarrow \bigoplus_{i=0}^{s_w+1} F^\star_{w,\mathcal{D}^w_i\ bool},$$

and so we can prove

$$\bigoplus_{i=0}^{s_u+1} \left(F^\star_{v,\mathcal{D}^v_i\ bool} \oplus F^\star_{w,\mathcal{D}^w_i\ bool}\right) \leftrightarrow F_{v\ bool} \oplus F_{w\ bool},$$

which gives us (since u is a plus gate)

$$\bigoplus_{i=0}^{s_u+1} F^\star_{u,\mathcal{D}^u_i\ bool} \leftrightarrow F_{u\ bool}\,.$$

Case 2: If u is a product gate with children v, w, using the same notation as above, for $j = 0, \ldots, s_u+1$, we define $F^\star_{u,\mathcal{D}^u_j} := \sum_{i=0}^{j} F^\star_{v,\mathcal{D}^v_i} \cdot F^\star_{w,\mathcal{D}^w_{j-i}}$. If $j > u_s$, let $F^\star_{u,\mathcal{D}^u_j} := 0$. 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:

$$F_{u\ bool} \leftrightarrow \bigoplus_{i=0}^{s_u+1} F^\star_{u,\mathcal{D}^u_i\ bool}$$

as follows:

$$\begin{aligned}
F_{u\ bool} &\leftrightarrow (F_{v\ bool} \wedge F_{w\ bool}) \\
&\leftrightarrow \left(\bigoplus_{j=0}^{s_v+1} F^\star_{v,\mathcal{D}^v_j\ bool}\right) \wedge \left(\bigoplus_{i=0}^{s_w+1} F^\star_{w,\mathcal{D}^w_i\ bool}\right) \\
&\leftrightarrow \bigoplus_{j=0}^{s_v+s_w+2} \bigoplus_{i=0}^{j} \left(F^\star_{v,\mathcal{D}^v_i\ bool} \wedge F^\star_{w,\mathcal{D}^w_{j-i}\ bool}\right) \\
&\leftrightarrow \bigoplus_{j=0}^{s_u+1} \left(\bigoplus_{i=0}^{j} \left(F^\star_{v,\mathcal{D}^v_i\ bool} \wedge F^\star_{w,\mathcal{D}^w_{j-i}\ bool}\right)\right) \\
&\leftrightarrow \bigoplus_{i=0}^{s_u+1} F^\star_{u,\mathcal{D}^u_i\ bool}\,.
\end{aligned}$$

□

4.6 Homogenous non-commutative formula identities have polynomial-size Frege proofs

To conclude Theorem 4.7 it remains to prove Theorem 4.11. Here we will prove Theorem 4.11, 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 form 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., $\mathsf{false} \oplus G \equiv G$, where $\oplus$ stands for XOR). This notation will make our arguments a bit more convenient to read. For 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 4.7.2 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 4.9 (Induced part of a formula). *Let F' be a subformula of F and $g_1, \ldots, g_k$ be gates in F' and $c_1, \ldots, c_k$ be constants in $\mathbb{F}$. Then $F'[c_1/g_1, \ldots, c_k/g_k]$ 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 to consider only arithmetic (resp. Boolean) formulas with *no* 0-1 (resp. true, false) constants. We say that a Boolean or arithmetic formula is *non-constant* if it contains at least one variable.

Lemma 4.10 (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* $\mathrm{poly}(|F|)$ *that computes the (non-commutative) zero polynomial, such that $F_{bool} \vdash^* F'_{bool}$.*

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 theorem

Theorem 4.11 (restatement of Theorem 1.8)**.** *There exists a constant c such that for any non-commutative syntactic homogeneous formula $F(\overline{x})$ over $GF(2)$ of size s that is identically zero, the corresponding Boolean tautology $\neg F_{bool}(\overline{p})$ has a Frege proof of size at most s^c (for sufficiently large s).*

Proof. First, by Lemma 4.10 we can assume without loss of generality that $F(\overline{x})$ 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). Thus, assume from now that F is constant-free and let d be the syntactic-degree of F.

Note that the syntactic-degree d of F is at most $s+1$. Theorem 4.12, 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 $\overline{F} = (v_1, , \dots, v_m)$ of non-commutative formulas, we write $\mathbf{T}\overline{F}$ to denote the (transposed) vector of non-commutative *formulas* whose jth entry is $\mathbf{T}_{j1} \times F_1 + \ldots + \mathbf{T}_{jn} \times F_m$ 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 $\Lambda\overline{F}$ to denote the vector of non-commutative formulas similar as defined above for $\mathbf{T}\overline{F}$, except that now the matrix $\mathbf{T}$ has the a 0-1 formula in each entry. We denote by $\mathbf{T}\Lambda\overline{F}$ the vector of non-commutative formulas $(\mathbf{T}\Lambda)\overline{F}$, 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 arithmetic formulas over $GF(2)$, we intend to treat $A \leftrightarrow B$ as a Boolean tautology (Definition 4.3). 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 $\overline{x}$ variables with 0-1 coefficients.

3. For every $i = 1, \dots, d$, $\overline{F}_i$ is a vector of induced parts of F, each computing a homogenous non-commutative polynomial of degree exactly i. The length of the vectors $\overline{F}_i$ is m_{d-i}. Accordingly, we denote by $\widehat{\overline{F}_i}$ the vector of non-commutative *polynomials* in $\overline{F}_i$.

These witnesses are such that the following hold:

$$\Lambda_{d-i}\widehat{\overline{F}_i} = \overline{0}, \qquad i = 1, \dots, d \quad \text{is a true equality;}^{12} \tag{12}$$

$$\vdash^* F \leftrightarrow \Lambda_0 \overline{F}_d \text{ (meaning that } \vdash^* F \leftrightarrow F_d\text{, since } \Lambda_0 = 1\text{);}^{13} \tag{13}$$

$$\vdash^* \Lambda_{d-i}\overline{F}_i \leftrightarrow \mathbf{T}_{d-i+1}\Lambda_{d-i+1}\overline{F}_{i-1}\,, \qquad i = 2, \dots, d\,. \tag{14}$$

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 (12). 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 by (14), $\Lambda_{d-2}\overline{F}_2 = \mathbf{T}_{d-1}\Lambda_{d-1}\overline{F}_1$, 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.

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 equation (12) and the short Frege proofs of the tautological Boolean formulas in (13) and (14). 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 tth coordinate of this vector.

By (12), the following Boolean formulas are all tautologies:

$$\bigwedge_{w \in [m_{d-i}]} \left(\neg \left(\bigoplus_{t:\Lambda_{d-i}(w,t)=1} F_{bool}(\overline{p})_{i,t} \right) \right), \qquad i = 1, \dots, d. \tag{15}$$

By (14), for every $i = 2, \dots, d$, and every $u \in [m_{d-i}]$, we have short Frege proofs of the following

[12]This is a *semantic* equality. I.e., in itself it does not entail a small proof of the equality.

[13]Note that $\overline{F}_d$ is identical to F_d, because the only induced part of F of degree d is F itself, due to syntactic homogeneity.

logical equivalence (between two Boolean formulas; the left hand side being the uth row in $\Lambda_{d-i}\overline{F}_i$):

$$\left(\bigoplus_{t:\Lambda_{d-i}(u,t)=1} F_{bool}(\overline{p})_{i,t} \right) \leftrightarrow$$
$$\bigoplus_{w\in[m_{d-i+1}]} \left(\mathbf{T}_{d-i+1}(u,w)_{bool}(\overline{p}) \ \wedge \left(\bigoplus_{t:\Lambda_{d-i+1}(w,t)=1} F_{bool}(\overline{p})_{i-1,t} \right) \right) . \quad (16)$$

Claim. *There are polynomial-size Frege proofs of* (15)*.*

This will conclude the proof of Theorem 4.11, 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, (15) 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 (16) becomes false for every $u \in [m_{d-2}]$, and so the left hand side of (16)

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

is also false for every $u \in [m_{d-2}]$. We continue in this manner until we arrive to (15) for $i = d$. $\square_{\text{claim}}$

We have thus concluded the proof of Theorem 4.11. $\square$

4.7 Identity witnessing theorem

It remains to prove the following:

Theorem 4.12 (Identity witnessing theorem)**.** *Let $F(\overline{x})$ 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 4.6 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.

4.7.1 Algebraic branching programs

We introduce the following definition:

Definition 4.13 (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 4.14 (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 4.14, we give in Lemma 4.15 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 4.6 we required the witnesses to consist of non-commutative *formulas* and not ABPs. In the next section we show, based on Lemma 4.15, 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 lth layer. For every $j = 0, \ldots, l$, we denote the nodes on the jth layer by $v_{j1}, \ldots, v_{jm_j}$, where m_j stands for the total number of nodes in the jth layer. For a given $i = 0, \ldots, l$, 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:

$$\overline{A}_i = \left(A(v_{l-i,1}, v_{\text{sink}}), \ldots, A(v_{l-i,m_{l-i}}, v_{\text{sink}})\right).$$

Each entry in this vector computes a non-commutative homogenous degree i polynomial. It is *important to note* that $\overline{A}_i$ is only a convenient notation, namely, when we apply in what follows a matrix product to the vector $\overline{A}_i$, we will treat different coordinates in the vector $\overline{A}_i$ 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 $\overline{A}_i$ it occurs only once). We will thus build a *single* ABP out of a matrix product with the vector $\overline{A}_i$, as described in what follows.

For a 0-1 matrix Λ of dimension $m \times m$ and a *multi-source* ABP $\overline{A}$ with m sources $v_1, \ldots, v_m$ and 0 to l layers, we write $\Lambda\overline{A}$ to denote the $l+1$ layered ABP with m sources that results from $\overline{A}$ when we join together several sources into a single source, maintaining the outgoing edges of the joined sources. Specifically, the ith source of the new ABP computes the non-commutative polynomial $\sum_k^m \Lambda_{ik} v_k$, and this is done by defining the outgoing edges of the ith 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 $\overline{A} = (v_1, , \ldots, v'_m)$ we write $\mathbf{T}\overline{A}$ to denote the ABP whose 0 layer consists of m sources, and the ith node in the 0th layer, for $i = 1, \ldots, m$, is connected to the jth node in 1st layer, for $j = 1, \ldots, m'$, with an edge labeled by the linear form $\mathbf{T}_{ij}$. In case Λ is a 0-1 matrix, then $\mathbf{T}\Lambda\overline{A}$ 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 $\overline{A}$, as explained above. We also denote by $\widehat{\overline{A}}$ the corresponding vector of non-commutative *polynomials* computed by the coordinates in $\overline{A}$.

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

Lemma 4.15 (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 $i = 0, \ldots, l-1$, each of dimension $m_i \times m_i$, where $m_i = \text{poly}(|A|)$, such that $\Lambda_0 = 1$ and*

$$\Lambda_{l-i}\overline{A}_i = \overline{0}, \quad \textit{for all } i = 0, \ldots, l-1$$

(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 $i = 1, \ldots, l-1$, of dimension $m_{i-1} \times m_i$ and whose entries are homogenous linear forms in the $\overline{x}$ variables with 0-1 coefficients, such that*

$$\Lambda_{l-i}\overline{A}_i = \mathbf{T}_{l-i+1}\Lambda_{l-i+1}\overline{A}_{i-1}, \quad \textit{for } i = 2, \ldots, d, \quad \textit{and} \tag{17}$$

$$\Lambda_0\overline{A_l} = A(v_{\text{source}}, v_{\text{sink}}), \tag{18}$$

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 ith layer. Since we assumed $\Lambda_0 = 1$, and since $\overline{A_l} = A(v_{\text{source}}, v_{\text{sink}})$, we conclude equation (18) in the lemma.

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

$$\Lambda_j\overline{A}_{l-j} = 0, \tag{19}$$

as well as (17), that is,

$$\Lambda_j\overline{A}_{l-j} = \mathbf{T}_{j+1}\Lambda_{j+1}\overline{A}_{l-j-1}\,. \tag{20}$$

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

Induction step: Assume that for $0 \leq h < l-1$, $\Lambda_0, \ldots, \Lambda_{h-1}$ and $\mathbf{T}_0, \ldots, \mathbf{T}_h$ were already constructed, and that the equality (19) holds for every $j = 0, \ldots, h$, and equality (20) holds for every $j = 0, \ldots, h-1$. We will now construct Λ_{h+1} such that (19) holds for $j = h+1$:

$$\Lambda_{h+1}\overline{A}_{l-h}\,,$$

and $\mathbf{T}_{h+1}$ such that (20) holds with $j = h$:

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

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, where for each entry (p,q), for $p \in [m_h], q \in [m_{h+1}]$,

$$M_{h,h+1}(p,q) = A(v_{h,p}, v_{h+1,q}) = \sum_{k=1}^{n} c_k x_k, \qquad \text{where } c_k \in \{0,1\}.$$

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

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

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

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

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

$$\Lambda_h M_{h,h+1}^k \overline{A}_{l-h-1} = \overline{0}, \quad \text{for } k = 1, \ldots, n. \tag{21}$$

Now, consider the basis of the span of all row vectors in all the matrices $\Lambda_h M_{h,h+1}^k$, for $k = 1, \ldots, n$. 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 (21) we know that every row vector in $\Lambda_h M_{h,h+1}^k$ is *orthogonal* to $\overline{A}_{l-h-1}$ (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 $k = 1, \ldots, n$, is also orthogonal to $\overline{A}_{l-h-1}$, and so

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

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

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

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

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

□

4.7.2 Implicitly working with ABPs in Frege system

Here we use Lemma 4.15 to conclude the existence of (formula-based) identity witnesses (as required in Theorem 4.12) and by that conclude the proof of Theorem 4.7.

Recall the notion of an induced part of a formula (Definition 4.9): for a subformula F' of F and gates $g_1, \ldots, g_k$ in F' and 0-1 constants $c_1, \ldots, c_k$, $F'[c_1/g_1, , \ldots, , c_k/g_k]$ 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.

For two vertices v', v'' in the ABP A, we denote by $A(v', v'')$ the polynomial computed by the ABP with the source v' and the sink v'' and all the paths leading from v' to v''. 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 4.16 (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 4.17 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 arithmetic formulas computing the linear forms computed by $A(v, u)$ are denoted also by $A(v, u)$):

$$F_v^\bullet \vdash^* \sum_{\substack{u:\ u \text{ has an incoming} \\ \text{edge from } v}} A(v, u) \times F_u^\bullet. \tag{22}$$

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 4.17. *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, (22) 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

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

from nodes in A to their "corresponding" nodes in F (this will help us define the mapping $F^\bullet_v$). 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 joint 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 $F^\bullet_v$. Let F be a syntactic homogenous non-commutative formula without constants, A its corresponding ABP, and $g : \text{nodes}(A) \to \text{nodes}(F)$ the function, all whose construction is described above. We construct the mapping $F^\bullet_v$ for nodes v in A as follows. Throughout the construction we maintain the following conditions:

(i) for every node v in A, the formula $F^\bullet_v$ computes the same non-commutative polynomial as $A\,(g(v), v_{\text{sink}})$;

(ii) for every node v in A, equation (22) 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 translates into an ABP, with A computed "above" B).

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

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

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

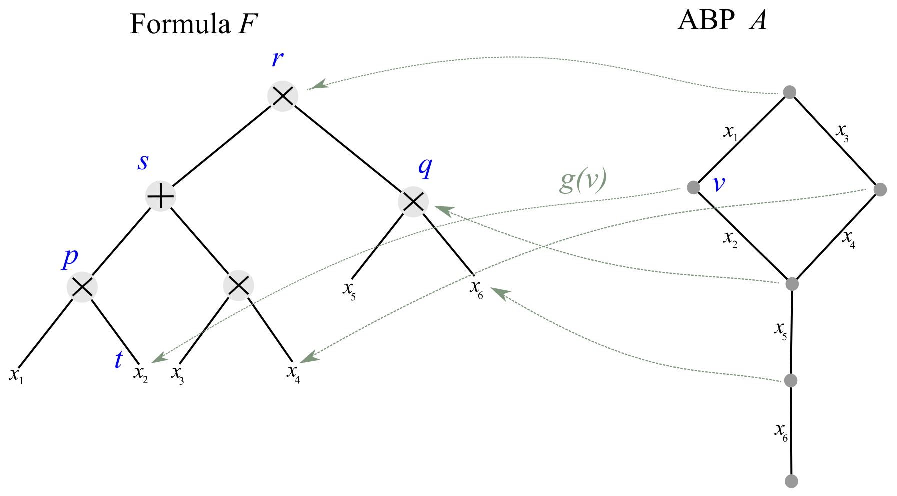

Figure 1

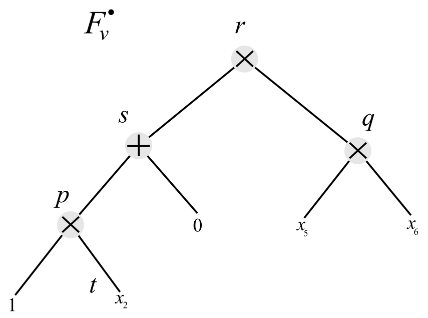

Figure 2

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 (22) 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. $\square_{\text{claim}}$

This concludes the proof of Lemma 4.17. $\square$

We are now ready to conclude the proof of the identity witnessing theorem (Theorem 4.12):

Proof of Theorem 4.12. Recall the ABP-based identity witnesses we showed existed in Lemma 4.15. Our goal is to show that there are (formula-based) identity witnesses (as defined in the proof of Theorem 4.11).

Using the correspondence given in Lemma 4.17, we can replace each ABP $A(v, v_{\text{sink}})$ occurring in some $\overline{A}_i$ (for some node v in A and some $i = 0, \ldots, l$) by a corresponding v-part $F_v^{\bullet}$. Denote with $\overline{F}_i$ the result of this replacement. Thus, $\overline{F}_i$ contains v-parts $F_v^{\bullet}$ of F, each computing a homogenous polynomial of degree i. With this replacement we get (we assume that $l = d$):

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

2. There exist $d-1$ matrices $\mathbf{T}_i$, for $i = 1, \ldots, d-1$, of dimension $m_{i-1} \times m_i$ and whose entries are homogenous linear forms in the $\overline{x}$ variables with 0-1 coefficients, such that
$$\Lambda_{d-i}\overline{F}_i = \mathbf{T}_{d-i+1}\Lambda_{d-i+1}\overline{F}_{i-1}, \quad \text{for } i = 2, \ldots, d, \text{ and} \tag{23}$$
$$\Lambda_0\overline{F}_d = A(v_{\text{source}}, v_{\text{sink}})\,. \tag{24}$$

Recall that $\Lambda\overline{F}$, for Λ a matrix and $\overline{F}$ a vector of formulas, is a vector of formulas, where each formula is written as a balanced (partial) sum of the formulas in $\overline{F}$ (see the Identity Witnesses' definition in the proof of Theorem 4.11).

Our goal now is to show
$$\vdash^* \Lambda_{d-i}\overline{F}_i \leftrightarrow \mathbf{T}_{d-i+1}\Lambda_{d-i+1}\overline{F}_{i-1}\,, \qquad i = 2, \ldots, d\,, \tag{25}$$
and
$$\vdash^* F \leftrightarrow \Lambda_0\overline{F}_d \text{ (meaning that } \vdash^* F \leftrightarrow F_d\text{, since } \Lambda_0 = 1). \tag{26}$$
Note that (26) holds trivially since F_d and F are identical. For (25), by Lemma 4.17, we know that
$$F_v^{\bullet} \vdash^* \sum_{\substack{u:\, u \text{ has an incoming} \\ \text{edge from } v}} A(v,u) \times F_u^{\bullet}. \tag{27}$$

Consider $\Lambda_{d-i}\overline{F}_i$. Each of its entries is a (partial) sum of the formulas in $\overline{F}_i$ written as a balanced sum. By (27) we can write each entry in $\overline{F}_i$ as a (balanced) sum in which each summand is some linear form $A(v,u)$ times an entry in $\overline{F}_{i-1}$. We can thus write $\Lambda_{d-i}\overline{F}_i$ as $\mathbf{T}'\overline{F}_{i-1}$ for some matrix $\mathbf{T}'$ with linear forms in each of its entries. Since the identity stated in (25) is a *true* identity, we thus get

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

But such an identity is provable in Frege with a polynomial-size proof, because we only need to prove an identity between $\langle \mathbf{t}'_j, \overline{F}_{i-1}\rangle$ and $\langle \mathbf{t}_j, \overline{F}_{i-1}\rangle$, for each of the jth 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). □

4.8 Conclusions

The propositional-calculus has a ubiquitous presence in logic and computer science at large. Within complexity theory and propositional proof complexity in particular it has a prominent role, and 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, give renewed hope for progress on the fundamental lower bounds in proof complexity. in so far that it reduces the problem of proving lower bounds on Frege proofs to the problem of establishing non-commutative formula lower bounds (which are already known for many polynomials). Since non-commutative lower bounds are already known and since proving the sort of matrix rank lower bounds that are required to establish non-commutative formula lower bounds for the permanent and determinant are fairly simple ([Nis91]), the current work provides a quite compelling evidence that Frege lower bounds might indeed not be very far away.

One possible route for Frege lower bounds is to reduce directly Frege lower bounds to the problem of lower bounding the non-commutative formula size of polynomials that are *already known to be hard* for the class of non-commutative formulas. We believe that this route is certainly a plausible one. A somewhat less direct approach is to show that some tautologies require non-commutative IPS refutations whose associated partial-derivative matrices (in the sense of Nisan [Nis91]) have high rank—here, the task would be to lower bound non-commutative polynomials that are given only "semi-explicitly" (that is, they are given in terms of the properties of the non-commutative IPS (Definition 1.2)); in other words, one has to establish lower bounds on a *family* of polynomials (for each fixed number of variables n).

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 complex

ity: 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 $\boldsymbol{NC}^1$ (cf. [CN10])).

We have also tighten 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 [HT12], 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.

References

[ABSRW04] 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)).

[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).

[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.

[AKV04] Albert Atserias, Phokion G. Kolaitis, and Moshe Y. Vardi. Constraint propagation as a proof system. In *CP*, pages 77–91, 2004.

[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.

[Bre74] Richard P. Brent. The parallel evaluation of general arithmetic expressions. *J. ACM*, 21(2):201–206, 1974.

[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.

[CN10] Stephen Cook and Phuong Nguyen. *Logical Foundations of Proof Complexity.* ASL Perspectives in Logic. Cambridge University Press, 2010.

[CR79] Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. *The Journal of Symbolic Logic*, 44(1):36–50, 1979.

[Gen35] Gerhard Gentzen. Untersuchungen über das logische schließen. *Mathematische Zeitschrift*, 39:68–131, 1935.

[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).

[GP14] Joshua A. Grochow and Toniann Pitassi. Circuit complexity, proof complexity, and polynomial identity testing. In *55th Annual IEEE Symposium on Foundations of Computer Science, FOCS*, 2014. Also available as arXiv:1404.3820 [cs.CC].

[Hru11] Pavel Hrubeš. How much commutativity is needed to prove polynomial identities? *Electronic Colloquium on Computational Complexity, ECCC*, (Report no.: TR11-088), June 2011.

[HT12] Pavel Hrubeš and Iddo Tzameret. Short proofs for the determinant identities. In *Proceedings of the 44th Annual ACM Symposium on the Theory of Computing (STOC)*, New York, 2012. ACM.

[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.

[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.

[Kra95] Jan Krajíček. *Bounded arithmetic, propositional logic, and complexity theory*, volume 60 of *Encyclopedia of Mathematics and its Applications.* Cambridge University Press, Cambridge, 1995.

[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.

[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.

[Kra10] Jan Krajíček. *Forcing with random variables and proof complexity*, volume 382 of *London Mathematical Society Lecture Notes Series.* Cambridge Press, 2010.

[Kra11] Jan Krajíček. *Forcing with random variables and proof complexity.* London Mathematical Society Lecture Note Series, No.382. Cambridge University Press, 2011.

[LT13] Fu Li and Iddo Tzameret. Generating matrix identities and proof complexity. *Electronic Colloquium on Computational Complexity, TR13-185*, 2013. arXiv:1312.6242 [cs.CC] `http://arxiv.org/abs/1312.6242`.

[Nis91] N. Nisan. Lower bounds for non-commutative computation. *Proceedings of the 23th Annual ACM Symposium on the Theory of Computing*, pages 410–418, 1991.

[PBI93] Toniann Pitassi, Paul Beame, and Russell Impagliazzo. Exponential lower bounds for the pigeonhole principle. *Comput. Complexity*, 3(2):97–140, 1993.

[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.

[Pit98] Toniann Pitassi. Unsolvable 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.

[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.

[Raz85] A. A. Razborov. Lower bounds on the monotone complexity of some Boolean functions. *Dokl. Akad. Nauk SSSR (in Russian)*, 281(4):798–801, 1985. [English translation in Sov. Math. Dokl., vol . 31 (1985), pp. 354-357.].

[Raz06] Ran Raz. Separation of multilinear circuit and formula size. *Theory of Computing, Vol. 2, article 6*, 2006.

[Raz09] Ran Raz. Multi-linear formulas for permanent and determinant are of super-polynomial size. *J. ACM*, 56(2), 2009.

[Raz13] Ran Raz. Tensor-rank and lower bounds for arithmetic formulas. *J. ACM*, 60(6):40, 2013.

[Raz15] Alexander A. Razborov. Pseudorandom generators hard for k-DNF resolution and polynomial calculus resolution. *Annals of Mathematics*, 181:415–472, 2015.

[Rec76] Robert Reckhow. *On the lengths of proofs in the propositional calculus.* PhD thesis, University of Toronto, 1976. Technical Report No . 87.

[RS05] Ran Raz and Amir Shpilka. Deterministic polynomial identity testing in non commutative models. *Computational Complexity*, 14(1):1–19, 2005.

[RT08a] Ran Raz and Iddo Tzameret. Resolution over linear equations and multilinear proofs. *Ann. Pure Appl. Logic*, 155(3):194–224, 2008.

[RT08b] Ran Raz and Iddo Tzameret. The strength of multilinear proofs. *Computational Complexity*, 17(3):407–457, 2008.

[Sch80] Jacob T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 27(4):701–717, 1980.

[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.

[Spi71] Philip M. Spira. On time-hardware complexity tradeoffs for boolean functions. In *Fourth International Symposium on Systems Sciences*, pages 525–527, 1971.

[Str73] Volker Strassen. Vermeidung von divisionen. *J. Reine Angew. Math.*, 264:182–202, 1973. (in German).

[Tza08] Iddo Tzameret. *Studies in Algebraic and Propositional Proof Complexity.* PhD thesis, Tel Aviv University, 2008.

[Tza11] Iddo Tzameret. Algebraic proofs over noncommutative formulas. *Information and Computation*, 209(10):1269–1292, 2011.

[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.

[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.