ALGEBRAIC PROOFS OVER NONCOMMUTATIVE FORMULAS

IDDO TZAMERET *

Abstract. We study possible formulations of algebraic propositional proof systems operating with noncommutative formulas. We observe that a simple formulation gives rise to systems at least as strong as Frege, yielding a semantic way to define a Cook-Reckhow (i.e., polynomially verifiable) algebraic analog of Frege proofs, different from that given in [BIK+97, GH03]. We then turn to an apparently weaker system, namely, polynomial calculus (PC) where polynomials are written as ordered formulas (*PC over ordered formulas*, for short). Given some fixed linear order on variables, an arithmetic formula is *ordered* if for each of its product gates the left subformula contains only variables that are less-than or equal, according to the linear order, than the variables in the right subformula of the gate. We show that PC over ordered formulas (when the base field is of zero characteristic) is strictly stronger than resolution, polynomial calculus and polynomial calculus with resolution (PCR), and admits polynomial-size refutations for the pigeonhole principle and the Tseitin's formulas. We conclude by proposing an approach for establishing lower bounds on PC over ordered formulas proofs, and related systems, based on properties of lower bounds on noncommutative formulas [Nis91].

The motivation behind this work is developing techniques incorporating rank arguments (similar to those used in arithmetic circuit complexity) for establishing lower bounds on propositional proofs.

Contents

Date: July 2011.

Key words and phrases. Proof complexity, algebraic proof systems, Frege proofs, lower bounds, noncommutative formulas, polynomial calculus.

*Institute for Theoretical Computer Science, The Institute for Interdisciplinary Information Science (IIIS), FIT building, Tsinghua University, Beijing, 100084, China, Email: `tzameret@tsinghua.edu.cn`. This work was supported in part by the National Basic Research Program of China Grant 2007CB807900, 2007CB807901, the National Natural Science Foundation of China Grant 61033001, 61061130540, 61073174. Part of this research was done while the author was a part of the Mathematical Institute, Academy of Sciences of the Czech Republic, Žitná 25, 115 67 Prague 1, Czech Republic; Supported by The Eduard Čech Center for Algebra and Geometry and The John Templeton Foundation.

1. Introduction

This work investigates algebraic proof systems establishing propositional tautologies, in which proof lines are written as noncommutative arithmetic formulas (noncommutative formulas, for short). Research into the complexity of algebraic propositional proofs is a central line in proof complexity (cf. [Pit97, Tza08] for general expositions). Another prominent line of research is that dedicated to connections between circuit classes and the propositional proofs based on these classes. In particular, considerable efforts were made to borrow techniques used for lower bounding certain circuit classes, and utilize them to show lower bounds on *proofs* operating with circuits from the given classes. For example, bounded depth Frege proofs can be viewed as propositional logic operating with AC^0 circuits, and lower bounds on bounded depth Frege proofs use techniques borrowed from AC^0 circuits lower bounds (cf. [Ajt88, KPW95, PBI93]). Pudlák [Pud99] and Atserias et al. [AGP02] studied proofs based on monotone circuits, motivated by known exponential lower bounds on monotone circuits. Raz and the author [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).

The current work is a contribution to this line of research, where the circuit class is noncommutative formulas. The motivation behind this work is the hope that certain rank arguments, found successful in lower bounding the size of certain types of arithmetic circuits, might also help in establishing lower bounds for the corresponding algebraic proofs. For this purpose, the choice of noncommutative formulas is natural, since such formulas constitute a fairly weak circuit class, and the proof of exponential-size lower bounds on noncommutative formulas, given by Nisan [Nis91], uses a considerably transparent rank argument.

We will show that for certain formulations of propositional proof systems over noncommutative formulas demonstrating lower bounds is likely to be hard, as the systems we get are quite strong, and specifically, at least as strong as Frege proofs. On the other hand, by formulating a proof system operating with fairly restricted formulas that compute a certain type of noncommutative polynomials, we obtain a system that we show is strictly stronger than known algebraic proof systems (like the polynomial calculus). For this apparently weaker system, demonstrating lower bounds seems not to be outside the reach of current techniques. In particular, we propose to study the complexity of these proofs by measuring the maximal *rank* of a polynomial appearing in a proof, instead of the maximal degree (the latter is done in the polynomial calculus). It is known that the rank of a noncommutative polynomial (as defined for instance by Nisan [Nis91]) is proportional to the minimal size of a noncommutative formula computing the polynomial. We argue for the usefulness of measuring the maximal rank of a polynomial in algebraic proofs, by demonstrating a certain property of ranks of "ordered polynomials" (as defined formally), and relating it to proof complexity lower bounds (via an example of a conditional lower bound).

1.1. **Results and related work.** We concentrate on algebraic proofs establishing propositional contradictions where polynomials are written as noncommutative formulas. We deal with two kinds of proof systems—both are variants (and extensions) of the polynomial calculus (PC) introduced in [CEI96]. In PC we start from a set of initial polynomials from $\mathbb{F}[x_1, \ldots, x_n]$, the ring of polynomials with coefficients from $\mathbb{F}$ (the intended semantics of a proof-line p is the equation $p = 0$ over $\mathbb{F}$). We derive new proof-lines by using two basic algebraic inference rules: from two polynomials p and q, we can deduce $\alpha \cdot p + \beta \cdot q$, where α, β are elements of $\mathbb{F}$; and from p we can deduce $x_i \cdot p$, for a variable x_i ($i = 1, \ldots, n$). We also have Boolean axioms $x_i^2 - x_i = 0$, for all $i = 1, \ldots, n$, expressing that the variables range over $\{0, 1\}$ values. Our two proof systems extend PC as follows:

- **NFPC:** PC over noncommutative formulas. This proof system operates with noncommutative polynomials over a field, written as noncommutative formulas, where every proof-line consists of a polynomial p and can be written as *any* formula F that computes p (these kind of algebraic proof systems are sometimes called *semantic proof systems*). The rules of addition and multiplication are similar to PC, except that multiplication is done *either from left or from right*. We also add a "Boolean" axiom $x_i x_j - x_j x_i$, for any pair of variables, that expresses the fact that for $0, 1$ values to the variables, multiplication is in fact commutative (indeed, note that in any noncommutative $\mathbb{F}$-algebra this axiom must be true when the variables x_i, x_j range over $\{0, 1\}$ values; see Section 3.1).
- **OFPC:** PC over ordered formulas. This proof system is PC operating with ordered polynomials written as ordered formulas, in which, as before, every ordered polynomial p inside the proof can be written as *any* ordered formula F that computes p. An ordered polynomial is a noncommutative polynomial such that the order of products in all monomials respects a fixed linear order on the variables, and an ordered formula is a noncommutative formula in which every subformula computes an ordered polynomial (see Definition 4.1). The rules of OFPC are similar to PC, namely, addition of two previously derived ordered polynomials and the product of a previously derived ordered polynomial p with a variable x_i (where now, the result of multiplying p by x_i is the corresponding *ordered* polynomial; e.g., multiplying the ordered polynomial $x_1 \cdot x_4 + x_3$ by x_2 results in $x_1 \cdot x_2 \cdot x_4 + x_2 \cdot x_3$, assuming the order on variables is defined via the increasing order on their indices).

Both proof systems are shown to be Cook-Reckhow systems (that is, *polynomial verifiable*, sound and complete proof systems for propositional tautologies).

(1) The first proof system NFPC is shown to polynomially simulate Frege (this is partly because of the choice of Boolean axioms). This gives a semantic definition of a Cook-Reckhow proof system operating with arithmetic formulas, simpler in some way from that proposed by Grigoriev and Hirsch [GH03]: the paper [GH03] aims at formulating a formal propositional proof system for establishing propositional tautologies (that is, a Cook-Reckhow proof system), which is an algebraic analog of the Frege proof system. In order to make their system polynomially-verifiable, the authors augment it with a set of auxiliary rewriting rules, intended to derive arithmetic formulas from previous arithmetic formulas via the polynomial-ring axioms (that is, associativity, commutativity, distributivity and the zero and unit elements rules). In this framework arithmetic formulas are treated as syntactic terms, and one must explicitly apply the polynomial-ring rewrite rules to derive a formula from previous ones. Our proof system NFPC is simpler in the sense that we get a similar proof system to that in [GH03], while adding no rewriting rules: both proof systems can simulate Frege and both are polynomially verifiable and operate with arithmetic formulas, or in our case with noncommutative formulas. The idea is that because we use noncommutative formulas as proof-lines, to verify that a line was derived correctly from previous lines we can use the deterministic polynomial identity testing algorithm for noncommutative formulas devised by Raz and Shpilka [RS05] (and so we do not need any rewriting rules).

(2) For the second proof system OFPC we show that, despite its apparent weakness, it is stronger than Polynomial Calculus with Resolution (PCR; and hence it is also stronger than both PC and resolution), and also can polynomially simulate a proof system operating with restricted forms of disjunctions of linear equalities called $\mathrm{R}^0(\mathrm{lin})$ (introduced in [RT08a]). The latter implies polynomial-size refutations for the pigeonhole principle and the Tseitin graph formulas, due to corresponding upper bounds demonstrated in [RT08a].

We then propose a simple lower bound approach for OFPC, based on properties of products of ordered formulas (these properties are proved in a similar manner to Nisan's size lower bounds on noncommutative formulas, that is, by lower bounding the rank of certain matrices associated with noncommutative polynomials). We show that certain conditions are sufficient to yield super-polynomial lower bounds on OFPC proofs.

Note: All the results in this paper hold when one considers *algebraic branching programs* (ABPs) (Definition 6.1) instead of noncommutative formulas, and *ordered*-ABPs instead of ordered-formulas. An *ordered*-ABP is an ABP such that the order of variables appearing on the edges of every path from source to sink on the ABP graph, respects a fixed linear order on the variables (see [JQS10] for a close model called π-ordered ABP).

Related work. There is some resemblance between noncommutative formulas (and in fact, algebraic branching programs) and ordered binary decision diagrams (OBDDs) (e.g., close techniques were used to obtain polynomial identity testing algorithms for noncommutative formulas [RS05] and for OBDDs [Waa97]). Thus, proofs operating with noncommutative formulas are reminiscent to the OBDD-based proof systems introduced and studied in [AKV04, Kra08, Seg07]. Nevertheless, one difference between OBDD-based proofs and noncommutative formulas-based proofs is that the feasible monotone interpolation lower bound technique is applicable in the case of OBDD-based systems, while this technique does not known to lead to super-polynomial size lower bounds even on PC proofs (and thus, also on OFPC proofs which are shown to polynomially simulate PC proofs).

Another proof system, that is even closer to OFPC, is that operating with *multilinear formulas* introduced in [RT08b] (under the name fMC). The upper bounds on OFPC proofs are similar to that shown for multilinear proofs in [RT08b]. Moreover, the technique used by Raz to establish super-polynomial lower bounds on multilinear formulas in [Raz09] is close (though more involved and includes additional ingredients) to that used by Nisan in the lower bound proof for noncommutative formulas [Nis91]. Therefore, proving lower bounds on OFPC proofs might help in establishing lower bounds on multilinear proofs.

2. Preliminaries

For a natural number we let $[n] = \{1, \dots, n\}$.

2.1. Noncommutative polynomials and formulas.

Let $\mathbb{F}$ be a field. Denote by $\mathbb{F}[x_1, \dots, x_n]$ the ring of (commutative) polynomials with coefficients from $\mathbb{F}$ and variables $x_1, \dots, x_n$. We denote by $\mathbb{F}\langle x_1, \dots, x_n\rangle$ the *noncommutative* ring of polynomials with coefficients from $\mathbb{F}$ and variables $x_1, \dots, x_n$. In other words, $\mathbb{F}\langle x_1, \dots, x_n\rangle$ is the ring of polynomials (where a polynomial is a formal sum of products of variables and field elements) conforming to all the polynomial-ring axioms excluding the commutativity of multiplication axiom. For instance, if x_i, x_j are two different variables, then $x_i \cdot x_j$ and $x_j \cdot x_i$ are two different polynomials in $\mathbb{F}\langle x_1, \dots, x_n\rangle$ (note that variables do commute with field elements).

We say that $\mathcal{A}$ is an *algebra over* $\mathbb{F}$, or an $\mathbb{F}$-*algebra*, if $\mathcal{A}$ is a vector space over $\mathbb{F}$ together with a distributive multiplication operation; where multiplication in $\mathcal{A}$ is associative (but it need not be commutative) and there exists a multiplicative unity in $\mathcal{A}$.

A noncommutative formula is just a (standard, commutative) arithmetic formula, except that product gates compute product of polynomials in the noncommutative ring $\mathbb{F}\langle x_1, \dots, x_n\rangle$ (and thus children of product gates are ordered):

Definition 2.1 (Noncommutative formula)**.** *Let $\mathbb{F}$ be a field and $x_1, x_2, \dots$ be variables. A* noncommutative arithmetic formula *(or* noncommutative formula *for short) is a 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 $\mathbb{F}$ 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 product gate). We assume that there is only one node of out-degree zero, called* the root. *A noncommutative formula* computes *a noncommutative polynomial in $\mathbb{F}\langle x_1, \dots, x_n\rangle$ in the following way. A leaf computes the input variable or field element that labels it. A plus gate computes the sum of polynomials computed by its incoming nodes. A product gate computes the* noncommutative *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 by the root. The* depth *of a formula is the maximal length of a path from the root to the leaf. The **size** of a noncommutative formula f is the total number of nodes in its underlying tree, and is denoted $|f|$.*

Definition 2.2 (Arithmetic formula). *An* arithmetic formula *is defined in a similar way to a noncommutative 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).*

Given a pair of noncommutative formulas F and G and a variable x_i, we denote by $F[G/x_i]$ the formula F in which every occurrence of x_i is substituted by the formula G.

Raz and Shpilka [RS05] showed that there is a deterministic polynomial identity testing (PIT) algorithm that decides whether two noncommutative formulas compute the same noncommutative polynomial:

Theorem 2.3 (PIT for noncommutative 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.*[1]

Let $p \in \mathbb{F}[x_1, \dots, x_n]$ be a polynomial. Then, p is said to be *multilinear* if the power of every variable in all its monomials is at most one. Also, p is said to be *homogenous* if the total degree of each of its monomials is the same. If p is a polynomial of (total) degree d, then $p = \sum_{i=0}^{d} p^{(i)}$, where $p^{(i)}$ is the *ith homogenous component of* p, that is, the sum of all monomials of total degree i in p.

2.2. **Proof systems and simulations.** Let $L \subseteq \Sigma^*$ be a language over some alphabet Σ. A *proof system for a language* L is a polynomial-time algorithm A that receives $x \in \Sigma^*$ and a string π over a binary alphabet ("the [proposed] proof" of x), such that there exists a π with $A(x,\pi) = \mathsf{true}$ if and only if $x \in L$. Following [CR79], a *Cook-Reckhow proof system* (or simply a *propositional proof system*) is a proof system for the language of propositional tautologies in the de Morgan basis $\{\mathsf{true}, \mathsf{false}, \vee, \wedge, \neg\}$ (coded in some efficient [polynomial-time] way, e.g., in the binary $\{0,1\}$ alphabet).

Assume that $\mathcal{P}$ is a proof system for the language L, where L is not the set of propositional tautologies in De Morgan's basis. In this case we can still consider $\mathcal{P}$ as a proof system for propositional tautologies by fixing a translation between L and the set of propositional tautologies in De Morgan basis (such that $x \in L$ iff the translation of x is a propositional tautology [and such that the translation can be done in polynomial-time]). If two proof systems $\mathcal{P}_1$ and $\mathcal{P}_2$ establish two different languages L_1, L_2, respectively, then for the task of comparing their relative strength we fix a translation from one language to the other.

[1] We assume here that the field $\mathbb{F}$ can be efficiently represented (e.g., the field of rationals).

In some cases, we shall confine ourselves to proofs establishing propositional tautologies or unsatisfiable CNF formulas.

A propositional proof system is said to be a *propositional refutation system* if it establishes the language of unsatisfiable propositional formulas (this is clearly a propositional proof system by the definition above, since we can translate every unsatisfiable propositional formula into its negation and obtain a tautology).

Definition 2.4. *Let $\mathcal{P}_1, \mathcal{P}_2$ be two proof systems for the same language L (in case the proof systems are for two different languages we fix a translation from one language to the other, as described above). We say that $\mathcal{P}_2$* polynomially simulates *$\mathcal{P}_1$ if given a $\mathcal{P}_1$ proof (or refutation) π of a F, then there exists a proof (respectively, refutation) of F in $\mathcal{P}_2$ of size polynomial in the size of π. In case $\mathcal{P}_2$ polynomially simulates $\mathcal{P}_1$ while $\mathcal{P}_1$ does not polynomially simulates $\mathcal{P}_2$ we say that $\mathcal{P}_2$ is* strictly stronger *than $\mathcal{P}_1$.*

2.3. **Polynomial Calculus.** Algebraic propositional proof systems are proof systems for finite collections of polynomial equations having no $0, 1$ solutions over some fixed field. (Formally, each different field yields a different algebraic proof system.) Proof-lines in algebraic proofs (or refutations) consist of polynomials p over the given fixed field. Each such proof-line is interpreted as the polynomial equation $p = 0$. To consider the *size* of algebraic refutations we fix the way polynomials inside refutations are written.

Notation: An *inference rule* is written as $\frac{A}{B}$ or $\frac{A \quad B}{C}$, meaning that given the proof-line A one can deduce the proof-line B, or given both the proof-lines A, B one can deduce the proof-line C, respectively.

The Polynomial Calculus is a propositional algebraic proof system first considered in [CEI96]:

Definition 2.5. (Polynomial Calculus (PC)). *Let $\mathbb{F}$ be some fixed field and let $Q = \{Q_1, \ldots, Q_m\}$ be a collection of multivariate polynomials from $\mathbb{F}[x_1, \ldots, x_n]$. Let the set of axiom polynomials be:*

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

A PC proof *from Q of a polynomial g is a finite sequence $\pi = (p_1, ..., p_\ell)$ of multivariate polynomials from $\mathbb{F}[x_1, \ldots, x_n]$, where $p_\ell = g$ and for every $1 \le i \le \ell$, either $p_i = Q_j$ for some $j \in [m]$, or p_i is a Boolean axiom, or p_i was deduced from p_j, p_k, for $j, k < i$, by one of the following inference rules:*

Product:

$$\frac{p}{x_r \cdot p}\,, \qquad \text{for } 1 \le r \le n\,.$$

Addition:

$$\frac{p \qquad q}{a \cdot p + b \cdot q}\,, \qquad \text{for } a, b \in \mathbb{F}\,.$$

A PC refutation of *Q is a proof of 1 (which is interpreted as $1 = 0$, that is the unsatisfiable equation standing for* false*) from Q. The* degree *of a PC-proof is the maximal degree of a polynomial in the proof. The* ***size*** *of a* PC *proof π is the total number of monomials (with nonzero coefficients) in all the proof-lines, denoted $|\pi|$.*

Important note: The size of PC proofs can be defined as the total formula sizes of all proof-lines, where polynomials are written as *sums of monomials*, or more formally, as (unbounded fan-in depth-2 arithmetic) $\Sigma\Pi$ formulas.[2] This complexity measure is equivalent up to a factor of n to

[2]A $\Sigma\Pi$ formula F is an arithmetic formula whose underlying tree is of depth 2 and has unbounded fan-in, such that the root is labeled with a plus gate, the children of the root are labeled with product gates and the leaves are labeled with either variables or field elements.

the standard complexity measure counting the total number of monomials appearing in the proofs (Definition 2.5).

Definition 2.6. (Polynomial Calculus with Resolution (PCR)). *The PCR proof system is defined similarly to PC (Definition 2.5), except that for every variable x_i a new formal variable $\bar{x}_i$ and a new axiom $x_i + \bar{x}_i - 1$ are added to the system, and the Boolean axioms of PCR are as follows:*

Boolean axioms: $x_i \cdot \bar{x}_i$.

The inference rules, and all other definitions are similar to that of PC. Specifically, the size *of a PCR proof is defined as the total number of monomials in all proof-lines (where now we count monomials in the variables x_i and $\bar{x}_i$).*

3. Polynomial calculus over noncommutative formulas

In this section we propose a possible formulation of algebraic propositional proof systems that operate with noncommutative polynomials. We observe that dealing with *propositional* proofs—that is, proofs whose variables range over $0,1$ values—makes the variables "semantically" commutative. Therefore, for the proof systems to be complete (for unsatisfiable collections of noncommutative polynomials over $0,1$ values), one may need to introduce rules or axioms expressing commutativity. We show that such a natural formulation of proofs operating with noncommutative formulas polynomially simulate the entire Frege system. This justifies—if one is interested in concentrating on propositional proof systems weaker than Frege (and especially on lower bounds questions)—our formulation in Section 4 of algebraic proofs operating with noncommutative arithmetic formulas with a *fixed product order* (called *ordered formulas*). The latter system can be viewed as operating with commutative polynomials over a field precisely like PC, while the complexity of proofs is measured by the total size of ordered formulas needed to write the polynomials in the proof. In other words, the role played by the noncommutativity in this system is only in measuring the sizes of proofs: while in PC-proofs the size measure is defined as the number of monomials appearing in the proofs—or equivalently, the total size of formulas in proofs in which formulas are written as (depth-2) $\Sigma\Pi$ circuits—the proof system developed in Section 4 is measured by the total ordered formula size.

3.1. The proof system NFPC.

We now define a proof system operating with noncommutative polynomials written as noncommutative arithmetic formulas.

In algebraic proof systems like the polynomial calculus we transform unsatisfiable propositional formulas into a collection Q of polynomials having no solution over a field $\mathbb{F}$. In the noncommutative setting we translate unsatisfiable propositional formulas into a collection Q of noncommutative polynomials from $\mathbb{F}\langle x_1, \dots, x_n\rangle$ that have no solution over any noncommutative $\mathbb{F}$-algebra (e.g., the matrix algebra with entries from $\mathbb{F}$). Although our "Boolean" axioms will not force only $0,1$ solutions over noncommutative $\mathbb{F}$-algebras, they will be sufficient for our purpose: every unsatisfiable propositional formula translates (via a standard polynomial translation) into a collection Q of noncommutative polynomials from $\mathbb{F}\langle x_1, \dots, x_n\rangle$, for which Q and the Boolean axioms have no (common) solution in any noncommutative $\mathbb{F}$-algebra. Furthermore, *the Boolean axioms will in fact force commutativity of variables product*—as required for variables that range over $0,1$ values (although, again, the Boolean axioms do not force only $0,1$ values when variables range over noncommutative $\mathbb{F}$-algebras). Let us elaborate further on this point:

We say that an (algebraic) proof system is *implicationally complete* whenever for any collection of polynomials $q_1, \dots, q_m, p$ over a field $\mathbb{F}$, if every assignment that satisfies $q_1 = 0, \dots, q_m = 0$ also satisfies $p = 0$, then there is a proof of p from the assumptions $q_1, \dots, q_m$. In our case, since the variables $x_1, \dots, x_n$ intend to range over $0,1$ values, we have the Boolean axioms $x_i^2 - x_i$, for any $i \in [n]$. But since over any noncommutative $\mathbb{F}$-algebra, any assignment that satisfies $x_1^2 - x_1 = 0, \ldots, x_n^2 - x_n = 0$ must satisfy also $x_i \cdot x_j - x_j \cdot x_i = 0$ (for all $i, j \in [n]$), any implicationally complete propositional proof system for noncommutative polynomials over a noncommutative $\mathbb{F}$-algebra must be able to derive (from only the Boolean axioms) the polynomials $x_i \cdot x_j - x_j \cdot x_i$, for all $i, j \in [n]$.

Definition 3.1 (Polynomial calculus over noncommutative formulas: NFPC)**.** *Fix a field $\mathbb{F}$ and let $Q := \{q_1, \ldots, q_m\}$ be a collection of noncommutative polynomials from $\mathbb{F}\langle x_1, \ldots, x_n\rangle$. Let the set of axiom polynomials be:*

Boolean axioms:

$$x_i \cdot (1 - x_i)\,, \qquad \textit{for all } 1 \le i \le n\,.$$
$$x_i \cdot x_j - x_j \cdot x_i\,, \qquad \textit{for all } 1 \le i \ne j \le n\,.$$

Let $\pi = (p_1, \ldots, p_\ell)$ be a sequence of noncommutative polynomials from $\mathbb{F}\langle x_1, \ldots, x_n\rangle$, such that for each $i \in [\ell]$, either $p_i = q_j$ for some $j \in [m]$, or p_i is a Boolean axiom, or p_i was deduced by one of the following inference rules using p_j, p_k, for $j, k < i$:

Left/right product:

$$\frac{p}{x_r \cdot p} \qquad \frac{p}{p \cdot x_r}\,, \qquad \textit{for } 1 \le r \le n\,.$$

Addition:

$$\frac{p \quad q}{a \cdot p + b \cdot q}\,, \qquad \textit{for } a, b \in \mathbb{F}\,.$$

We say that π is an NFPC proof of p_ℓ from Q *if all proof-lines in π are written as noncommutative formulas. (The semantics of an* NFPC *proof-line p_i is the polynomial equation $p_i = 0$.) An* NFPC refutation of Q *is a proof of the polynomial 1 from Q. The **size** of an* NFPC *proof π is defined as the total size of all the noncommutative formulas in π and is denoted $|\pi|$.*

Remark: (i) The Boolean axioms might have roots different from $0, 1$ over noncommutative $\mathbb{F}$-algebras. (ii) The Boolean axioms are true for $0, 1$ assignments: $x_i \cdot x_j - x_i \cdot x_j = 0$ for all $x_i, x_j \in \{0, 1\}$.

We now show that NFPC is a sound and complete Cook-Reckhow proof system. First note that we have defined NFPC with no rules expressing the polynomial-ring axioms (the latter are sometimes added to algebraic proof systems operating with arithmetic formulas for the purpose of verifying that every formula in the proof was derived correctly [via the deduction rules of the system] from previous lines; see discussion in Section 1.1). Nevertheless, due to the deterministic polynomial-time PIT procedure for noncommutative formulas (Theorem 2.3) the proof system defined will be a Cook-Reckhow system (that is, verifiable in polynomial-time [whenever the base field and its operations can be efficiently represented]).

Proposition 3.2. *There is a deterministic polynomial-time algorithm that decides whether a given string is an* NFPC*-proof (over efficiently represented fields).*

Proof. We can assume that the proof also indicates from which previous lines a new line was inferred via the NFPC inference rules. Then, by Proposition 2.3, there is a polynomial-time algorithm that, e.g., given two noncommutative formulas F_1, F_2 such that the proof indicates that F_2 was inferred from F_1 via the Left product rule, decides whether the formula $x_i \times F_1$ and F_2 computes the same noncommutative polynomial. And similarly for the other deduction rules of NFPC. ☐

Proposition 3.3. *The systems* NFPC *is sound and complete. Specifically, let Q be a collection of noncommutative polynomials from $\mathbb{F}\langle x_1, \ldots, x_n\rangle$. Assume that for every $\mathbb{F}$-algebra, there is no $0, 1$*

solution for Q (that is, an $0,1$ assignment to variables that gives all polynomials in Q the value 0), then the contradiction $1 = 0$ can be derived in NFPC *from Q.*

Proof. Soundness holds because both rules of inference are sound over any $\mathbb{F}$-algebra. Completeness stems by the simulation of $\mathcal{F}$-$\mathcal{PC}$ shown in Theorem 3.6 below (and the fact that if no $\mathbb{F}$-algebra has a solution then also there is no solution in $\mathbb{F}$ itself, which implies, by completeness of $\mathcal{F}$-$\mathcal{PC}$, that there exists an $\mathcal{F}$-$\mathcal{PC}$ refutation of Q). □

For the next statements we use the algebraic propositional proof system $\mathcal{F}$-$\mathcal{PC}$ introduced by Grigoriev and Hirsch [GH03] as an algebraic analog of the Frege system. The proof system $\mathcal{F}$-$\mathcal{PC}$ is an algebraic propositional proof system operating with (general, that is, commutative) arithmetic formulas over a field, and it includes auxiliary rewriting rules allowing to develop equal polynomials syntactically via the polynomial-ring axioms. The proof system $\mathcal{F}$-$\mathcal{PC}$ has the Boolean axioms of PC, the rules of PC and in addition the rewrite rules expressing the polynomial-ring axioms. Each line in $\mathcal{F}$-$\mathcal{PC}$ is treated as a *term*, that is, a formula, and so the rules are also syntactic: addition of terms via the plus gate and product of a term by a variable from the left. We first need to define the notion of a rewrite rule:

Definition 3.4 (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 3.5 ($\mathcal{F}$-$\mathcal{PC}$ [GH03])**.** *Fix a field $\mathbb{F}$. Let $F := \{f_1, \ldots, f_m\}$ be a collection of* formulas[3] *computing polynomials from $\mathbb{F}[x_1, \ldots, x_n]$. 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 $\pi = (\Phi_1, \ldots, \Phi_\ell)$ of formulas computing polynomials from $\mathbb{F}[x_1, \ldots, x_n]$ 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:

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

Addition:

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

(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.)*[4]

(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 $\mathbb{F}[x_1, \ldots, x_n]$):*

Zero rule: $0 \cdot f \leftrightarrow 0$

Unit rule: $1 \cdot f \leftrightarrow f$

[3] Note here that we are talking about formulas (treated as syntactic terms), and *not* polynomials. Also notice that all formulas in $\mathcal{F}$-$\mathcal{PC}$ are (commutative) formulas computing (commutative) polynomials.

[4] 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, the definition of $\mathcal{F}$-$\mathcal{PC}$ in [GH03] and our Definition 3.5 polynomially-simulate each other.

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 $|\pi|$.*

Theorem 3.6. NFPC *(over any field) polynomially-simulates Frege. Specifically,* NFPC *polynomially-simulates $\mathcal{F}$-$\mathcal{PC}$ in the following sense: let $f_1, \ldots, f_m$ be a set of commutative formulas computing (commutative) polynomials that have no common* $0, 1$ *root, and assume that there is a size s $\mathcal{F}$-$\mathcal{PC}$ refutation of $f_1, \ldots, f_m$. Then, there exists an* NFPC *refutation of the same set of formulas $f_1, \ldots, f_m$ (but now viewed as computing noncommutative polynomials) of size polynomial in s.*

Proof. By [GH03] (see Theorem 3 there), $\mathcal{F}$-$\mathcal{PC}$ polynomially simulates Frege. We proceed by showing a simulation of $\mathcal{F}$-$\mathcal{PC}$ by NFPC by induction on the number of steps in an $\mathcal{F}$-$\mathcal{PC}$ proof.

Base case: Axioms and initial formulas. All axioms of $\mathcal{F}$-$\mathcal{PC}$ are also axioms in NFPC. Also, if the $\mathcal{F}$-$\mathcal{PC}$ refutation uses an initial formula f_i, then we use the same formula in NFPC.

Induction step:

Case 1: Addition rule. Assume we derive in $\mathcal{F}$-$\mathcal{PC}$ the *formula* $p + q$. By induction hypothesis we already have the two formulas p, q in NFPC. Thus, we can add them via the addition rule.

Case 2: Product rule. Assume we derive the formula $x_i \cdot p$ from the formula p in $\mathcal{F}$-$\mathcal{PC}$. By induction hypothesis we already have the formula p in NFPC. Thus, we can derive $x_i \cdot p$ by the Left product rule.

Case 3: Rewriting rules. Assume we derived a formula f using one of the rewriting rules of $\mathcal{F}$-$\mathcal{PC}$. The rewriting rules of associativity, distributivity, scalar rule, and unit and zero rules of $\mathcal{F}$-$\mathcal{PC}$ *do not change the noncommutative polynomial computed by an arithmetic formula.* Therefore, we get them "for free" in NFPC, in the sense that we can choose to write a noncommutative polynomial p in the proof as any noncommutative formula, as long as the chosen formula computes the noncommutative polynomial p. Thus, we only need to show how to simulate the commutativity rule, namely to show how to simulate commuting a term inside a formula. The key lemma for this is the following:

Lemma 3.7. *Let $\mathbb{F}$ be any field and let f, g be two noncommutative formulas computing (non-constant) polynomials from $\mathbb{F}\langle x_1, \ldots, x_n \rangle$. Then, there is an* NFPC *proof of size polynomial in $|f| + |g|$ of the formula $f \cdot g - g \cdot f$.*

Proof. First, we need to show that NFPC allows for substitution of identities inside proof-lines. Let A, h be noncommutative formulas and assume that the variable z occurs inside A only once. Then $A[h/z]$ denotes the noncommutative formula obtained from A by replacing the leaf labeled z by the formula h.

Claim 3.8. *Let A be a noncommutative formula, and let z be a variable that occurs only once inside A. Let h, h' be two noncommutative formulas h, h' of maximal size s. Then, there is an* NFPC *proof of $A[h/z] - A[h'/z]$ from $h - h'$ of size polynomial in $|A| + s$.*

Proof of claim: Straightforward induction on the size of A. $\blacksquare_{\text{Claim}}$

We get back to the proof of Lemma 3.7: proceed by induction on $|f|+|g|\geq 2$.
Base case: $|f|+|g|=2$. By assumption the polynomials computed by f,g are both non-constant, and so $f=x_i$ and $g=x_j$, for some $i,j\in[n]$. Therefore, we are done by the Boolean axiom $x_ix_j-x_jx_i$.

Induction step: Either $|f|>1$ or $|g|>1$. Assume without loss of generality that $|f|>1$. Following Claim 3.8, we shall use freely substitutions in formulas.
Case (i): $f=f_1+f_2$. Start from

$$f\cdot g-f\cdot g=f\cdot g-(f_1+f_2)\cdot g=f\cdot g-f_1\cdot g-f_2\cdot g\,. \tag{1}$$

By induction hypothesis we have a proof of $f_1\cdot g-g\cdot f_1$ and of $f_2\cdot g-g\cdot f_2$. Thus, we can substitute these identities in (1), to get $f\cdot g-g\cdot f_1-g\cdot f_2=f\cdot g-g\cdot(f_1+f_2)=f\cdot g-g\cdot f$.
Case (ii): $f=f_1\cdot f_2$. Start from

$$f\cdot g-f\cdot g=f\cdot g-(f_1\cdot f_2)\cdot g=f\cdot g-f_1\cdot(f_2\cdot g)\,. \tag{2}$$

By induction hypothesis we have a proof of $f_2\cdot g-g\cdot f_2$. Thus, we can substitute this identity in (2), to get $f\cdot g-f_1\cdot(g\cdot f_2)=f\cdot g-(f_1\cdot g)\cdot f_2$. By induction hypothesis again, we have $f_1\cdot g-g\cdot f_1$. And similarly, we get by substitution $f\cdot g-(g\cdot f_1)\cdot f_2=f\cdot g-g\cdot f$.
This concludes the proof of Lemma 3.7 □

To conclude the simulation of the commutativity rewrite rule of $\mathcal{F}$-$\mathcal{PC}$ (which will also conclude the proof of Theorem 3.6) we notice that, by Claim 3.8 and by Lemma 3.7, for any noncommutative formula A, such that z is a variable that occurs only once inside A, there is an NFPC proof of $A[(f\cdot g)/z]-A[(g\cdot f)/z]$ of size polynomial in $|A[(f\cdot g)/z]|$. □

4. Polynomial calculus over ordered formulas

In this section we formulate an algebraic proof system OFPC that operates with noncommutative polynomials in which every monomial is a product of variables in *nondecreasing order* (from left to right; and according to some fixed linear order on the variables), and where polynomials in proofs are written as *ordered formulas*, as defined below.

Let $X=\{x_1,\dots,x_n\}$ be a set of variables and let $\mathbb{F}$ be a field. Let $\preceq$ be a linear order on the variables X, that is, a total, reflexive and antisymmetric order on X. Let $f=\sum_{j\in J}b_j\mathscr{M}_j$ be a commutative polynomial in $\mathbb{F}[x_1,\dots,x_n]$, where the b_j's are coefficient from $\mathbb{F}$ and the $\mathscr{M}_j$'s are monomials in the X variables. We define $[\![f]\!]\in\mathbb{F}\langle x_1,\dots,x_n\rangle$ to be the (unique) noncommutative polynomial $\sum_{j\in J}b_j\cdot[\![\mathscr{M}_j]\!]$, where $[\![\mathscr{M}_j]\!]$ is the (noncommutative) product of all the variables in $\mathscr{M}_j$ such that the order of multiplications respects $\preceq$. We denote the image of the map $[\![\cdot]\!]:\mathbb{F}[x_1,\dots,x_n]\to\mathbb{F}\langle x_1,\dots,x_n\rangle$ by $\mathcal{G}$. We say that a polynomial is an *ordered polynomial* if it is a polynomial in $\mathcal{G}$.

Definition 4.1 (Ordered formula)**.** *Assume we fix some linear order $\preceq$ on variables $x_1,\dots,x_n$. A noncommutative formula (Definition 2.1) is said to be an* ordered formula *if the noncommutative polynomial computed by each of its subformulas is ordered. We say that an ordered formula F computes the* commutative *polynomial $f\in\mathbb{F}[x_1,\dots,x_n]$ whenever F computes $[\![f]\!]$.*

When we speak about ordered formulas and ordered polynomials, we shall assume we have some fixed linear order $\preceq$ on the variables in the background (and so ordered formulas and ordered polynomials are always considered with respect to this ordering).

We characterize ordered formulas in a simple syntactic way, different from Definition 4.1, and then prove the equivalence of the two characterizations (Proposition 4.4):

Definition 4.2 (Syntactic ordered formula)**.** *An ordered formula is a* syntactic ordered formula *if for each of its product gates the left subformula contains only variables that are less-than or equal, via $\preceq$, than the variables in the right subformula of the gate.*

Definition 4.3. *We say that a variable x_i* occurs in the polynomial h *(commutative or noncommutative) if there is a monomial with a nonzero coefficient in h in which x_i has a positive power.*

Note that a variable can appear (or "occur") inside a *formula* while not occur in the *polynomial* the formula computes.

Proposition 4.4. *There is a polytime algorithm that receives a noncommutative formula Φ and a linear order on its variables, and returns* false *if Φ is not an ordered formula (with respect to the given linear order), and otherwise returns a syntactic ordered formula of the same size as Φ that computes the same (ordered) polynomial.*

Proof. First note that for any noncommutative formula F, the formula $F[0/x_i]$ computes the polynomial $f\restriction_{x_i=0}$ (namely, the polynomial f in which x_i is assigned 0) and so $F[0/x_i]$ computes f iff x_i does not occur in f.

The algorithm is as follows:

(1) Search for a product node in F that has on its left subformula a variable that is strictly greater (via the order $\preceq$) from some variable in its right subformula. If there is no such product node, then F itself is a syntactic ordered formula, and the algorithm returns F.

(2) Otherwise, let v be a product gate in F, with F_1 and F_2 its left and right subformulas, respectively. And suppose that F_1 contains the variable x_i and F_2 contains the variable x_j, such that $x_i \succ x_j$ (i.e., $x_i \succeq x_j$ and $x_i \neq x_j$). Let h_1, h_2 be the *polynomials computed* by F_1 and F_2, respectively. Check whether x_i occurs in h_1. To this end:
Check if the resulted formula $F_1[0/x_i]$ computes the same noncommutative polynomial as F_1 computes (using the PIT algorithm for noncommutative formulas).
  Case I: If the answer is "yes", then conclude that x_i does not occur in the polynomial h_1, and run the algorithm with the input formula F in which F_1 is substituted by $F_1[0/x_i]$.
  Case II: If the answer is "no", we know that the variable x_i does occur in the polynomial h_1. We check in a similar manner whether x_j occurs in h_2.
    (a) If x_j does *not* occur in h_2 run the algorithm with the formula F in which F_2 is substituted by $F_2[0/x_j]$.
    (b) Otherwise, x_j *does* occur in the polynomial h_2. We already know that x_i occurs in h_1, and so it must be that $h_1 \cdot h_2$ is not an ordered polynomial[5], and so the polynomial computed at v is not ordered and we return false.

Note that the algorithm described above returns either false (in case F is not an ordered formula) or a new formula that computes the same noncommutative polynomial as F and with *the same size* as F (because the only changes applied to the original formula F is substitution of variables by the constant 0). The running time of the algorithm is polynomial in the size of F. □

We can now define OFPC in a convenient way, without referring to noncommutative polynomials: the system OFPC is defined similarly to PC, except that the proof-lines are written as ordered formulas.

Definition 4.5 (PC over ordered formulas: OFPC)**.** *Let $\pi = (p_1, \dots, p_m)$ be a PC proof of p_m from some set of initial polynomials Q (that is, p_i are commutative polynomials from the ring of*

[5]Note that h_1, h_2 are polynomials (not formulas) and so if x_i occurs in h_1 and x_j occurs in h_2, it must be that there is a monomial with a nonzero coefficient in $h_1 \cdot h_2$ in which x_i multiplies from left x_j.

polynomials $\mathbb{F}[x_1,\dots,x_n]$), and let $\preceq$ be some linear order on the variables $x_1,\dots,x_n$. The sequence $(f_1,\dots,f_m)$ in which f_i is an ordered formula computing p_i (according to the order $\preceq$), is called an OFPC *proof of p_m from Q. The* ***size*** *of an* OFPC *proof is the total size of all the ordered formulas appearing in it.*

Similar to the proof system NFPC we have defined OFPC with no rules expressing the polynomial-ring axioms. Also, similar to NFPC, the system OFPC will constitute a Cook-Reckhow proof system, that is, there is a deterministic polynomial-time algorithm that decides whether a given string is an OFPC proof or not (whenever the base field and its operations can be efficiently represented):

Proposition 4.6. *For any linear order on the variables,* OFPC *is a sound, complete and polynomially-verifiable refutation system for establishing that a collection of (commutative) polynomial equations over a field does not have $0,1$ solutions. Specifically, (considering the language of polynomial translations of Boolean contradictions)* OFPC *is a Cook-Reckhow proof system.*

Proof. The soundness and completeness of OFPC stem from the soundness and completeness of PC. The fact that OFPC is a Cook-Reckhow proof system is proved in Proposition 4.8 below. □

We first need the following lemma:

Lemma 4.7. *For any linear order $\preceq$ on variables, there exists a polytime algorithm that receives an ordered formula Φ computing $[\![f]\!] \in \mathbb{F}\langle x_1,\dots,x_n\rangle$ (for some polynomial $f \in \mathbb{F}[x_1,\dots,x_n]$) and a variable x_r, for some $1 \le r \le n$, and outputs a new ordered formula that computes $[\![x_r \cdot f]\!]$.*

Proof. We can assume that Φ is a *syntactic* ordered formula, as otherwise we can transform it into such a formula by the algorithm in Proposition 4.4. We show that there is an algorithm $A(\Phi, x_r)$ that outputs the desired formula by induction on the size of Φ.

Base case:

(1) $A(c, x_r) := c \cdot x_r$, for $c \in \mathbb{F}$.

(2)
$$A(x_i, x_r) := \begin{cases} x_i \cdot x_r, & \text{if } x_i \preceq x_r; \\ x_r \cdot x_i, & \text{otherwise.} \end{cases}$$

Induction step:

(1) $A(\Phi_1 + \Phi_2,\, x_r) := A(\Phi_1, x_r) + A(\Phi_2, x_r)$.

(2)
$$A(\Phi_1 \cdot \Phi_2,\, x_r) := \begin{cases} A(\Phi_1, x_r) \cdot \Phi_2, & \text{if } x_r \text{ is } \preceq \text{ from every variable in } \Phi_2; \\ \Phi_1 \cdot A(\Phi_2, x_r), & \text{otherwise.} \end{cases}$$

□

Proposition 4.8. *For any linear order $\preceq$ on variables, there exists a polytime algorithm that given a sequence π of ordered formulas and another sequence $(Q_1,\dots,Q_m,G)$ of ordered formulas, outputs* **true** *iff π is an* OFPC *proof of the polynomial computed by G from the polynomials computed by $Q_1,\dots,Q_m$.*

Proof. We verify the following:

(1) All formulas in π are ordered formulas (according to the fixed linear order). By Proposition 4.4, this can be done in polynomial-time in the size of π.
(2) The last formula in π computes the same polynomial as G (using the PIT algorithm for noncommutative formulas).
(3) For every proof-line $f \in \pi$, one of the following holds:

(i) The formula f computes an axiom. This can be verified by checking whether f computes the same noncommutative polynomial as the formula $x_i^2 - x_i$, for some $1 \le i \le n$, or whether f computes some polynomial computed by Q_i, for some $1 \le i \le m$ (by Theorem 2.3).

(ii) The formula f computes the same ordered polynomial as $F_1 + F_2$, for some pair F_1, F_2 of ordered formulas in previous proof-lines (Theorem 2.3).

(iii) The formula f computes $[\![x_i \cdot h]\!]$, for some $1 \le i \le n$, where h is a polynomial computed by some previous proof-line. This can be checked as follows. Considering all possible pairs H and x_i, for H being a proof-lines (preceding f in π) and $i = 1, \ldots, n$, run the algorithm in Lemma 4.7 where the inputs are H and x_i. We get a new ordered formula H', and we check if H' computes the same noncommutative polynomial as f.

□

Note: Formally, for different n's, every set of variables $x_1, \ldots, x_n$ may have linear orders that are incompatible with each other. Nevertheless, in this paper, given a family Q of collections of initial polynomials $\{Q_n \mid n \in \mathbb{N}\}$ parameterized by n, and assuming that $Q_n \subseteq \mathbb{F}[x_1, \ldots, x_n]$ for all n, we will consider only linear orders such that: for every $n > 1$, the linear order on $x_1, \ldots, x_n$ is an *extension* of the linear order on $x_1, \ldots, x_{n-1}$. Equivalently, we can consider one fixed linear order on a countable set of variables $X = \{x_1, x_2, \ldots\}$.

5. Simulations, short proofs and separations for OFPC

In this section we are concerned with the relative strength of OFPC. Specifically, we show that OFPC, when operating with polynomials over fields of characteristic 0, is strictly stronger than the polynomial calculus, polynomial calculus with resolution (PCR) and resolution (for a definition of resolution, see for example [ABSRW02]). For this purpose, we show first that, for any linear order on the variables OFPC polynomially simulates PCR. Since PCR polynomially simulates both PC and resolution, we get that OFPC also polynomially simulates PC and resolution. Second, we show that OFPC admits polynomial-size refutations of hard tautologies for PCR (that is, tautologies that do not have polynomial-size PCR proofs). This is done by demonstrating that OFPC over fields of characteristic 0 polynomially simulates the $\mathrm{R}^0(\mathrm{lin})$ refutation system for the language of CNF formulas. The system $\mathrm{R}^0(\mathrm{lin})$ is an extension of resolution introduced in [RT08a]. Since $\mathrm{R}^0(\mathrm{lin})$ is provably stronger than PCR [RT08a], the result will follow.

5.1. **OFPC polynomially simulates PCR.** Let τ denote the linear transformation that maps the variables $\overline{x}_i$, for any $i \in [n]$, to $(1 - x_i)$, and denote $p{\restriction}\tau$ the polynomial p under the transformation τ.

Proposition 5.1. *For any linear order on the variables,* OFPC *polynomially simulates PCR (and PC and resolution). Specifically, if there is a size s PCR proof (with the variables $x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n$) of p from the axioms $p_{j_1}, \ldots, p_{j_k}$, then there is an* OFPC *proof of $p \restriction \tau$ from $p_{j_1}{\restriction}\tau, \ldots, p_{j_k}{\restriction}\tau$ of size $O(n \cdot s)$.*

Proof. Given some linear order on the variables, we assume that all ordered formulas respect this linear order (and so we do not refer explicitly to this order).

Let $\pi = (p_1, \ldots, p_t)$ be a PCR proof of size s from the axioms $p_{j_1}, \ldots, p_{j_k}$ (that is, p_i's are [commutative] polynomials from $\mathbb{F}[x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n]$, for some field $\mathbb{F}$, such that the total number of monomials occurring in all proof-lines in π is s). We need to show that there is an OFPC proof π' of p_i from the axioms, such that π' has size $O(n \cdot s)$.

Let Γ be the sequence obtained from π by replacing every product rule application in π, deriving $\bar{x}_i \cdot p$ from p (for any $i = 1, \ldots, n$), by the following proof sequence:

1. p
2. $x_i \cdot p$
3. $(1 - x_i) \cdot p$

(the second polynomial is derived by the product rule from the first polynomial, and the third polynomial is derived by the addition rule from the first and second polynomials).

Let $\Gamma \restriction \tau$ be the sequence obtained from Γ by applying the substitution τ on every proof-line in Γ. We claim that $\Gamma \restriction \tau$ is a PC proof of $p_t \restriction \tau$ from the initial polynomials $p_{j_1} \restriction \tau, \ldots, p_{j_k} \restriction \tau$: first, note that all product rule applications using $\bar{x}_i$ variables were eliminated in $\Gamma \restriction \tau$, and thus all product rule applications in $\Gamma \restriction \tau$ are legitimate PC product rule applications. Second, note that for any pair of polynomials g, h we have $g \restriction \tau + h \restriction \tau = (g + h) \restriction \tau$. Third, note that the axioms of PCR transform under τ to either 0 (which we can ignore in the new proof sequence) or to the PC axiom $x_i(1 - x_i)$.

By construction, every proof-line in $\Gamma \restriction \tau$ is either $p_i \restriction \tau$ or $x_j \cdot (p_i \restriction \tau)$, for some $p_i \in \pi$ and $j \in [n]$. Therefore, by definition of OFPC, it suffices to show that every $p_i \restriction \tau$ and $x_j \cdot (p_i \restriction \tau)$, for some $p_i \in \pi$ and $j \in [n]$, have ordered formulas of size at most $O(m \cdot n)$, where m is the number of monomials in p_i. For this purpose it is enough to show that for every monomial $\mathscr{M}$ in p_i there exists an $O(n)$ ordered formula computing the polynomial $\mathscr{M} \restriction \tau$. The latter is true since every such polynomial is a product of at most n terms, where each term is either x_i or $1 - x_i$, for some $i \in [n]$; such a product can be clearly written as an ordered formula of size $O(n)$. $\square$

In the rest of this section we show that OFPC polynomially simulates the proof system $\mathrm{R}^0(\mathrm{lin})$, and then use it to establish short proofs in OFPC.

5.2. **Resolution over linear equations** $\mathrm{R}(\mathrm{lin})$ **and its subsystem** $\mathrm{R}^0(\mathrm{lin})$**.** Here we follow [RT08a] and define the refutation systems $\mathrm{R}(\mathrm{lin})$ and $\mathrm{R}^0(\mathrm{lin})$. The system $\mathrm{R}(\mathrm{lin})$ is an extension of the resolution refutation system that works with disjunctions of linear equations instead of disjunction of literals. $\mathrm{R}^0(\mathrm{lin})$ is defined to be a subsystem of $\mathrm{R}(\mathrm{lin})$ in which certain restrictions are put on proof-lines in a proof.

Disjunctions of linear equations. Let L be a linear equation $a_1x_1 + \ldots + a_nx_n = a_0$. Then, the right hand side a_0 is called the *free-term* of L and the left hand side $a_1x_1 + \ldots + a_nx_n$ is called the *linear form* of L (the linear form can be 0). A *disjunction of linear equations* is of the following form:

$$\left(a_1^{(1)}x_1 + \ldots + a_n^{(1)}x_n = a_0^{(1)}\right) \vee \cdots \vee \left(a_1^{(t)}x_1 + \ldots + a_n^{(t)}x_n = a_0^{(t)}\right) , \tag{3}$$

where $t \geq 0$ and the coefficients $a_i^{(j)}$ are integers (for all $0 \leq i \leq n,\ 1 \leq j \leq t$). We remove duplicate linear equations from a disjunction of linear equations. The semantics of such a disjunction is the following: an assignment of integral values to the variables $x_1, ..., x_n$ is said to *satisfy* (3) if and only if there exists $j \in [t]$ so that $a_1^{(j)}x_1 + \ldots + a_n^{(j)}x_n = a_0^{(j)}$ holds under the given assignment. The **size** of a linear equation $a_1x_1 + \ldots + a_nx_n = a_0$ is defined to be $\sum_{i=0}^{n} |a_i|$, that is, the sum of the bit sizes of all a_i written in **unary** notation. Accordingly, the *size of the linear form* $a_1x_1 + \ldots + a_nx_n$ is $\sum_{i=1}^{n} |a_i|$. The *size of a disjunction of linear equations* is the total size of all linear equations in it. Similar to resolution, the *empty disjunction* is unsatisfiable and stands for the truth value false. We will consider only disjunctions of linear equations *with integral coefficients*. Given a vector $\vec{a}$ of n integers and a vector $\vec{x}$ of n variables $x_1, \ldots, x_n$, we write $\vec{a} \cdot \vec{x}$ to abbreviate $\sum_{i=1}^{n} a_ix_i$.

Translation of clauses. We can translate any CNF formula to a collection of disjunctions of linear equations as follows: every clause $\bigvee_{i\in I} x_i \vee \bigvee_{j\in J} \neg x_j$ in the CNF is translated into the disjunction $\bigvee_{i\in I}(x_i = 1) \vee \bigvee_{j\in J}(x_j = 0)$. Any Boolean assignment to the variables $x_1, \dots, x_n$ satisfies a clause D if and only if it satisfies its translation into disjunction of linear equations (where true is identified with 1 and false with 0).

The refutation system R(lin).

Definition 5.2 (R(lin))**.** *Let $K := \{K_1, \dots, K_m\}$ be a collection of disjunctions of linear equations in the variables $x_1, \dots, x_n$. An* R(lin)-proof from K of a disjunction of linear equations D *is a finite sequence $\pi = (D_1, ..., D_\ell)$ of disjunctions of linear equations, such that $D_\ell = D$ and for every $i \in [\ell]$ one of the following holds:*

(1) $D_i = K_j$, for some $j \in [m]$;
(2) D_i is a
  Boolean axiom: $(x_t = 0) \vee (x_t = 1)$*, for some $t \in [n]$;*
(3) D_i was deduced by one of the following R(lin)-inference rules, using D_j, D_k for some $j, k < i$:
  Resolution: *Let A, B be two, possibly empty, disjunctions of linear equations and let L_1, L_2 be two linear equations. From $A \vee L_1$ and $B \vee L_2$ derive $A \vee B \vee (L_1 - L_2)$.*
  Weakening: *From a possibly empty disjunction of linear equations A derive $A \vee L$, where L is an arbitrary linear equation over the variables $x_1, \dots, x_n$.*
  Simplification: *From $A \vee (0 = k)$ derive A, where A is a possibly empty disjunction of linear equations and $k \neq 0$.*

An R(lin) refutation of *a collection of disjunctions of linear equations K is a proof of the empty disjunction from K. The **size** of an R(lin) proof π is the total size of all the disjunctions of linear equations in π (where coefficients are written in unary representation) denoted $|\pi|$.*

In light of the translation between CNF formulas and collections of disjunctions of linear equations, we can consider R(lin) to be a proof system for the set of unsatisfiable CNF formulas.

The refutation system $\mathbf{R^0}$(lin). For our purposes we need to consider the restriction of R(lin), denoted R^0(lin) [RT08a]. The system R^0(lin) operates with disjunctions of (arbitrarily many) linear equations with constant coefficients excluding the free terms, under the following restriction: every disjunction can be partitioned into a constant number of sub-disjunctions, where each sub-disjunction either consists of linear equations that differ only in their free-terms or is a (translation of a) clause. Any linear *inequality* with Boolean variables can be represented by a disjunction of linear equations that differ only in their free-terms. So the R^0(lin) proof system resembles, to some extent, a proof system operating with disjunctions of constant number of linear inequalities with constant integral coefficients.

Example: The following is an example of an R^0(lin) proof-line:

$$(x_1 + \ldots + x_\ell = 1) \vee \cdots \vee (x_1 + \ldots + x_\ell = \ell) \vee (x_{\ell+1} = 1) \vee \cdots \vee (x_n = 1),$$

for some $1 \le \ell \le n$. Note that in the left part $(x_1 + \ldots + x_\ell = 1) \vee \cdots \vee (x_1 + \ldots + x_\ell = \ell)$ every disjunct has the same linear form with coefficients $0, 1$, and the right part $(x_{\ell+1} = 1) \vee \cdots \vee (x_n = 1)$ is a translation of a clause.

To formally define the R^0(lin) proof system we introduce the following definition:

Definition 5.3 ($\mathrm{R}_{\mathbf{c,d}}$(lin)-line)**.** *Let D be a disjunction of linear equations whose variables have integer coefficients with absolute values at most c (the free-terms are unbounded). Assume D can be partitioned into at most d sub-disjunctions $D_1, \dots, D_d$, where each D_i either consists of (an unbounded) disjunction of linear equations that differ only in their free-terms, or is a translation of a clause. Then the disjunction D is called an* $\mathrm{R}_{c,d}$(lin)-line*. The **size** of an $\mathrm{R}_{c,d}$(lin)-line D is* *defined as before, that is, as the total bit-size of all equations in D, where coefficients are written in unary representation.*

Thus, any $\mathrm{R}_{c,d}(\mathrm{lin})$-line is of the following general form:

$$\bigvee_{i\in I_1}\left(\vec{a}^{(1)}\cdot\vec{x}=\ell_i^{(1)}\right)\vee\cdots\vee\bigvee_{i\in I_k}\left(\vec{a}^{(k)}\cdot\vec{x}=\ell_i^{(k)}\right)\vee\bigvee_{j\in J}(x_j=b_j)\,, \tag{4}$$

where $k<d$ and for all $r\in[n]$ and $t\in[k]$, $a_r^{(t)}$ is an integer such that $|a_r^{(t)}|\le c$, and $b_j\in\{0,1\}$ (for all $j\in J$), and the $\ell_i^{(k)}$'s are (unbounded) integers (and $I_1,\ldots,I_k,J$ are unbounded sets of indices). Since a disjunction of clauses is a clause in itself, we can assume that in any $\mathrm{R}_{c,d}(\mathrm{lin})$-line only a single (translation of a) clause occurs.

The $\mathrm{R}^0(\mathrm{lin})$ proof system is a restriction of $\mathrm{R}(\mathrm{lin})$ in which each proof-line is an $\mathrm{R}_{c,d}(\mathrm{lin})$-line, for some fixed constants c,d:

Definition 5.4 ($\mathrm{R}^{\mathbf{0}}(\mathrm{lin})$)**.** *Let $K:=\{K_n\mid n\in\mathbb{N}\}$ be a family of collections of disjunctions of linear equations. Then $\{P_n\mid n\in\mathbb{N}\}$ is a family of* $\mathrm{R}^0(\mathrm{lin})$-proofs *of K if there exist constant integers c,d independent of n, such that: (i) each P_n is an $\mathrm{R}(\mathrm{lin})$-proof of K_n; and (ii) for all n, every proof-line in P_n is an $\mathrm{R}_{c,d}(\mathrm{lin})$-line. The **size** of an $\mathrm{R}^0(\mathrm{lin})$ proof is defined the same way as the size of $\mathrm{R}(\mathrm{lin})$ proofs, that is, as the total size of all the proof-lines.*

If K_n is a collection of disjunctions of linear equations parameterized by $n\in\mathbb{N}$, we shall say that K_n *has a polynomial-size (in n) $\mathrm{R}^0(\mathrm{lin})$ proof*, if there are some constants c,d independent of n and a polynomial p, such that for every n, K_n has $\mathrm{R}(\mathrm{lin})$ proof of size at most $p(n)$ in which every proof-line is an $\mathrm{R}_{c,d}(\mathrm{lin})$-line.

Both $\mathrm{R}(\mathrm{lin})$ and $\mathrm{R}^0(\mathrm{lin})$ are sound and complete Cook-Reckhow refutation systems for unsatisfiable CNF formulas (see Section 3.2 in [RT08a]).

5.3. OFPC **polynomially simulates** $\mathrm{R}^0(\mathrm{lin})$**.** Here we prove that OFPC over fields of characteristic 0 polynomially simulates $\mathrm{R}^0(\mathrm{lin})$ for the language of unsatisfiable CNF formulas. We translate a CNF, that is, a collection of clauses, into a collection of polynomials as follows: every clause $\bigvee_{i\in I}x_i\vee\bigvee_{j\in J}\neg x_j$ in the CNF is translated into $\prod_{i\in I}(1-x_i)\vee\prod_{j\in J}x_j$.

Theorem 5.5. *For any linear order on the variables,* OFPC *operating with polynomials over a field of characteristic* 0 *polynomially simulates* $\mathrm{R}^0(\mathrm{lin})$ *for the language of unsatisfiable CNF formulas. Moreover, we can assume that all formulas appearing in the OFPC proofs simulating* $\mathrm{R}^0(\mathrm{lin})$ *are ordered formulas of depth at most 3.*

In the rest of this subsection we work out the proof of Theorem 5.5.

Assume we have a family of $\mathrm{R}^0(\mathrm{lin})$ refutations $\{\pi_\ell:\ell\in\mathbb{N}\}$ of a CNF family $\{K_\ell:\ell\in\mathbb{N}\}$, in which every line is an $\mathrm{R}_{c,d}(\mathrm{lin})$-line for two constants c,d independent of ℓ. We wish to show an OFPC refutation of K_ℓ with size polynomial in $|\pi_\ell|$. Thus, consider a refutation $\pi=\pi_\ell=(D_1,\ldots,D_m)$, for some ℓ. The proof is almost similar to the proof that *multilinear proofs* can polynomial simulate $\mathrm{R}^0(\mathrm{lin})$, given in [RT08a]. We begin by providing an overview of the simulation:

- **Step I:** First we translate disjunctions of linear equations into polynomials. This is easy to do by considering a disjunction as a product, and turning any linear equation into its corresponding homogenous linear form. Thus, $\pi=(D_1,\ldots,D_m)$ can be transformed into a sequence $\widetilde{\pi}=(\widetilde{D}_1,\ldots,\widetilde{D}_m)$ of polynomials.
- **Step II:** We then show how to transform the sequence $\widetilde{\pi}$ into a PC refutation by adding new PC proof-lines, so that if D_k was derived from previous lines D_i,D_j by one of $\mathrm{R}(\mathrm{lin})$ rules, then the added proof-lines will constitute a PC derivation of $\widetilde{D}_k$ from previous lines $\widetilde{D}_i,\widetilde{D}_j$. This is not hard to do, but we have to take care that:

(1) the number of added lines is polynomial in the size of the original $\mathrm{R}^0(\mathrm{lin})$ refutation; and

(2) every newly added PC proof-line is a polynomial translation $\widetilde{D}$ of some $\mathrm{R}_{c',d'}(\mathrm{lin})$-line D (Definition 5.6 below), where D is of size polynomial in $|\pi|$ and c', d' are constants independent of n.

Step III: We now have a PC proof π' whose number of lines is polynomial in $|\pi|$ in which every line is a polynomial translation $\widetilde{D}$ of some $\mathrm{R}_{c',d'}(\mathrm{lin})$-line D, such that $|D|$ is polynomial in $|\pi|$. For the current step we extend the system PCR with the Product Rule $\frac{p}{g \cdot p}$, for any polynomial g (note that g is not necessarily a variable), and denote this extended system by PCR*. We then transform π' into a PCR* proof $\pi^\star$ in which every line is roughly a *multilinearization* $\mathbf{M}[\widetilde{D}]$ of a polynomial translation $\widetilde{D}$ of some $\mathrm{R}_{c',d'+1}(\mathrm{lin})$-line D, where $|D|$ is polynomial in $|\pi|$; However, the variables in $\pi^\star$ will be $\{x_1, \dots, x_n, \bar{x}_1, \dots, \bar{x}_n\}$. Also, note that if D is a clause, then $\widetilde{D}$ is already multilinear, which means that $\mathbf{M}[\widetilde{D}] = \widetilde{D}$, and so $\pi^\star$ is a refutation of the original CNF.

Step IV: In this step we show that every proof-line in $\pi^\star$ can be written as a certain simple depth-3 formula of polynomial-size in $|\pi|$. This step is accomplished by observing that the multilinearization of a polynomial translation of an $\mathrm{R}_{c,d}(\mathrm{lin})$-line is close to a product of constantly many *symmetric polynomials* (cf. [RT08a]). And then showing that any such product has a $\Sigma\Pi\Sigma$ depth-3 formula whose size is polynomial in the size of the original $\mathrm{R}_{c,d}(\mathrm{lin})$-line, over large enough fields (that is, over fields with at least $2n+1$ elements, for $2n$ being the number of variables), and whose bottom level linear forms have only a single variable.

Step V: We now have a PCR* proof $\pi^\star$ of the original CNF formula with polynomial in $|\pi|$ many lines and in which the following *invariant* holds: every proof-line can be written as a $\Sigma\Pi\Sigma$ depth-3 formula of polynomial-size in $|\pi|$ in which the bottom level linear polynomials have only a single variable. Since in OFPC the product rule can only multiply previous lines by a *variable*, we first show how to polynomially simulate in PCR, applications of the extended PCR* product rules $\frac{p}{g \cdot p}$ that occur in $\pi^\star$, while keeping the above invariant. Second, we need to change the resulting refutation into a PC refutation of the same CNF formula K having only the $\{x_1, \dots, x_n\}$ variables (and not using the axioms $x_i \cdot \bar{x}_i$ and $x_i + \bar{x}_i - 1$), and where the above invariant on the structure of lines still holds. This is easy to do by applying the linear transformation $\bar{x}_i \mapsto 1 - x_i$ on all polynomials in the refutation. We then claim that every line in the obtained PC refutation of K can be written as an ordered formula of depth-3 and of polynomial-size in $|\pi|$ (for any given order on variables).

We now turn to the formal construction.

Step I. Here we show how to transform disjunctions of linear equations D into polynomials $\widetilde{D}$. We turn a disjunction into a product and a linear equation $L = d$, for d the free term, into the polynomial $L - d$. Note that $\mathrm{R}^0(\mathrm{lin})$ operates with unbounded free-terms: the number d in the example above (or the $\ell_i^{(t)}$'s in (4)) are unbounded (their values may depend on n). Since we translate an integer $d \in \mathbb{Z}$ to the field element $1 + \ldots + 1$ (d times), we need to use a field whose characteristic is big enough to include (an isomorphic copy of) the integers up to d. We will simply assume that *our field has characteristic* 0, which means it includes every integer.

More concretely, our polynomial translation is as follows. A *polynomial translation of a clause* $\bigvee_{j \in J}(x_j^{b_j})$ is any product of the form $\prod_{j \in J}(x_j - b_j)$, where $b_j \in \{0, 1\}$ for all $j \in J$, and where $x_j^{b_j}$ is the literal x_j if $b_j = 1$ and $\neg x_j$ if $b_j = 0$. Accordingly, we define the *polynomial translation of a CNF formula* as the set consisting of the polynomial translations of the clauses in the CNF.

Definition 5.6 (Polynomial translation of $\mathrm{R}_{c,d}$(lin)-lines)**.** *A* polynomial translation of an $\mathrm{R}_{c,d}$(lin)-line *is a product of linear polynomials (that is, polynomials of the form $\sum_{i=1}^{n} a_i x_i + a_0$), such that:*

(1) All variables in the linear polynomials have integer coefficients with absolute values at most c (the constant terms [that correspond to the free-terms] are unbounded).

(2) D can be written as $\prod_{i=1}^{d} D_i$, where each D_i either consists of (an unbounded) product of linear forms that differ only in their free-terms*, or is a polynomial translation of a clause.*

*The **degree** of a polynomial-translation of an $\mathrm{R}_{c,d}$(lin)-line D is defined to be the total degree of the polynomial D.*

In other words, any polynomial translation of an $\mathrm{R}_{c,d}$(lin)-line has the following general form:

$$\prod_{j\in J}(x_j - b_j) \cdot \prod_{t=1}^{k}\prod_{i\in I_t}\left(\sum_{r=1}^{n} a_r^{(t)} x_r - \ell_i^{(t)}\right), \tag{5}$$

where $k < d$ and for all $r \in [n]$ and $t \in [k]$, $a_r^{(t)}$ is an integer such that $|a_r^{(t)}| \le c$, and $b_j \in \{0,1\}$ (for all $j \in J$) and the $\ell_i^{(t)}$'s are integers (and $I_1, \ldots, I_k, J$ are unbounded sets of indices).

Notation: As noted earlier, given an $\mathrm{R}_{c,d}$(lin)-line D we write $\widetilde{D}$ to denote its polynomial translation.

Step II. We now show how to obtain a PC proof π' from the R(lin) proof π, using the polynomial translation in Step I.

Proposition 5.7 (Translating R^0(lin) proofs to PC proofs)**.** *Let $K = \{K_m \mid m \in \mathbb{N}\}$ be a family of unsatisfiable CNF formulas translated into disjunctions of linear equations and let $\{P_m \mid m \in \mathbb{N}\}$ be a family of* R^0(lin)-proofs *of K, where each proof line in every P_m is an $\mathrm{R}_{c,d}$(lin)-line, for two constants c, d independent of m. Then, there are two constants c', d' depending only on c, d and a family of PC refutations $\{P'_m \mid m \in \mathbb{N}\}$ of (the polynomial translations of) K, such that for every $m \in \mathbb{N}$:*

(i) the number of lines in P'_m is polynomial in $|P_m|$; and

(ii) every line in P'_m is a polynomial translation of an $\mathrm{R}_{c',d'}$(lin)-line of degree polynomial in $|P_m|$.

Proof. We proceed by induction on the number of lines in P_m.

Base case: An R^0(lin) Boolean axiom $(x_i = 0) \vee (x_i = 1)$ is translated into $x_i \cdot (x_i - 1)$ which is already an axiom of PC (or can be derived from an axiom by multiplying b the scalar -1). An initial disjunction of linear equations from K_n is translated into its corresponding polynomial translation (Definition 5.6). In both cases we get polynomial translations of $\mathrm{R}_{c,d}$(lin)-lines with a polynomial (in $|P_m|$) degree (note that the initial disjunctions in K are $\mathrm{R}_{c,d}$(lin)-lines since they are clauses).

Induction step: We translate every R^0(lin) inference rule application into a PC proof sequence with polynomial in $|P_m|$ number of lines, and with each line being a polynomial translation of an $\mathrm{R}_{c',d'}$(lin)-line for two constants c', d' depending only on c, d, whose degree is bounded by a polynomial in $|P_m|$. We use the following claim:

Claim 5.8. *Let $p, q \in \mathbb{F}[x_1, \ldots, x_n]$ be two polynomials and let s be the minimal size of an arithmetic formula computing q. Then one can derive from p in PC the polynomial $q \cdot p$, with only a polynomial in s number of steps. Furthermore, assume that q, p are polynomial translations of* $\mathrm{R}_{c,d}$(lin)*-lines Q, P, respectively, for some constants c, d independent of n and with $|Q|, |P| \le t$,* *then the PC derivation of $q \cdot p$ from p has polynomial in t number of lines and contains only polynomial translations of $\mathrm{R}_{c',d'}(\mathrm{lin})$-lines of degree polynomial in t, for some constants c', d' independent of n.*

Proof of claim: By induction on s (and t in the second statement). We omit the details. $\blacksquare_{\text{Claim}}$

Assume that $D_i = D_j \vee L$ was derived from D_j using the weakening inference rule of $\mathrm{R}^0(\mathrm{lin})$, and L is some linear equation. Then, by Claim 5.8, $\widetilde{D}_i = \widetilde{D}_j \cdot \widetilde{L}$ can be derived from $\widetilde{D}_j$ with a PC derivation having at most polynomial in $|D_j \vee L|$ many steps, in which every line is a polynomial translation of an $\mathrm{R}_{c',d'}(\mathrm{lin})$-line of degree polynomial in t, for some constants c', d' independent of n.

Otherwise, assume that D_i was derived from D_j where D_j is $D_i \vee (0 = k)$, using the simplification inference rule of $\mathrm{R}^0(\mathrm{lin})$, and k is a non-zero integer. Then, $\widetilde{D}_i$ can be derived from $\widetilde{D}_j = \widetilde{D}_i \cdot -k$ by multiplying with $-k^{-1}$ (via the Addition rule of PC, and using the fact that we work in a field).

Thus, it remains to simulate the *resolution rule* application of $\mathrm{R}^0(\mathrm{lin})$. Let A, B be two disjunctions of linear equations and assume that

$$A \vee B \vee \left((\vec{a} - \vec{b}) \cdot \vec{x} = a_0 - b_0\right)$$

was derived in P_m from $A \vee (\vec{a} \cdot \vec{x} = a_0)$ and $B \vee (\vec{b} \cdot \vec{x} = b_0)$.

We need to derive

$$\widetilde{A} \cdot \widetilde{B} \cdot \left((\vec{a} - \vec{b}) \cdot \vec{x} - a_0 + b_0\right)$$

from $\widetilde{A} \cdot (\vec{a} \cdot \vec{x} - a_0)$ and $\widetilde{B} \cdot (\vec{b} \cdot \vec{x} - b_0)$. This is done by multiplying $\widetilde{A} \cdot (\vec{a} \cdot \vec{x} - a_0)$ with $\widetilde{B}$ and multiplying $\widetilde{B} \cdot (\vec{b} \cdot \vec{x} - b_0)$ with $\widetilde{A}$ and then subtracting the second resulted polynomial from the first resulted polynomial. By Claim 5.8, this can be done in PC with polynomial in $t = |A \vee (\vec{a} \cdot \vec{x} - a_0)| + |B \vee (\vec{b} \cdot \vec{x} - b_0)|$ many steps and where each proof-line is a polynomial translation of an $\mathrm{R}_{c',d'}(\mathrm{lin})$-line, where the degree of every such $\mathrm{R}_{c',d'}(\mathrm{lin})$-line is polynomial in t (which also implies that the degree of such lines is also upper bounded by $|P_m|$). $\square$

By Proposition 5.7, given our refutation π of a CNF, there exists a PC refutation π' of K with polynomial in $|\pi|$ number of lines, and with every line a polynomial translation $\widetilde{D}$ of an $\mathrm{R}_{c',d'}(\mathrm{lin})$-line D with degree at most polynomial in $|\pi|$, for two constants c', d'.

Step III. Recall that a polynomial $p \in \mathbb{F}[x_1, \ldots, x_n]$ is said to be *multilinear* if the power of every variable in all its monomials is at most one. Given the PC refutation π' from the previous step, we construct a PCR* refutation $\pi^\star$ of the same CNF, and where PCR* is an extension of PCR, defined as follows:

Definition 5.9 (PCR*)**.** *The proof systems PCR* is an extension of the PCR system (Definition 2.6) with the following product rule:*

Product:

$$\frac{p}{g \cdot p}\,, \qquad \textit{for any polynomial } g \in \mathbb{F}[x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n].$$

Definition 5.10 (Multilinearization operator)**.** *Given a field $\mathbb{F}$ and a polynomial $q \in \mathbb{F}[x_1, \ldots, x_n]$, we denote by $\mathbf{M}[q]$ the unique multilinear polynomial equal to q modulo the ideal generated by all the polynomials $x_i^2 - x_i$, for all variables x_i.*

For example, if $q = x_1^2 x_2 + a x_4^3 + 1$ (for some $a \in \mathbb{F}$) then $\mathbf{M}[q] = x_1 x_2 + a x_4 + 1$.

The main idea in Step III is formulated in the next proposition. It states that a PC refutation consisting of only translations of $\mathrm{R}_{c',d'}(\mathrm{lin})$-lines can be transformed without much increase in the number of lines into a "multilinearized" refutation, in which every line is roughly a multilinearization of (a polynomial translation of) an $\mathrm{R}_{c',d'}(\mathrm{lin})$-line. Formally, we have:

Proposition 5.11. *Let P be a PC refutation from an initial set K of multilinear polynomials in $\mathbb{F}[x_1, \ldots, x_n]$, and assume that every proof line in P is a polynomial translation of an $\mathrm{R}_{c',d'}(\mathrm{lin})$-line D of size at most t, for some fixed c', d'. Then there exists a PCR* refutation P' of K, such that:*

(1) the number of lines in P' is polynomially bounded in the number of lines in P;

(2) for every polynomial p in P', p is a multilinear polynomial in $\mathbb{F}[x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n]$ that can be written as a sum $\sum_{i=1}^{h} \mathbf{M}[\widetilde{D}_i]$, where h is a constant (independent of n, c', d') and where each $\widetilde{D}_i$ is a degree $O(t)$ polynomial translation of an $\mathrm{R}_{c',d'+1}(\mathrm{lin})$-line.

Proof. Let $(p_1, \ldots, p_m)$ be the PC refutation P, where for any $i \in [m]$, p_i is a polynomial in $\mathbb{F}[x_1, \ldots, x_n]$. The desired PCR* proof P' is constructed as follows.

First, we put $Q = (\mathbf{M}[p_1], \ldots, \mathbf{M}[p_m])$. We construct the PCR* refutation P' of K by adding appropriate PCR* proof-sequences to Q. This is done as follows:

Case A: If p_i is from K then by multilinearity of p_i we have $p_i = \mathbf{M}[p_i]$. And condition (2) in the statement of the proposition holds by assumption that p_i is a polynomial translation of an $\mathrm{R}_{c',d'}(\mathrm{lin})$-line D, where the size of D is at most t (and hence t is an upper bound on the degree of p_i).

Case B: If p_i was derived in P by the addition rule from previous lines p_j, p_k, for some $j, k < i$, then $p_i = \alpha p_j + \beta p_k$, for some $\alpha, \beta \in \mathbb{F}$. Thus, $\mathbf{M}[p_i] = \alpha \mathbf{M}[p_j] + \beta \mathbf{M}[p_k]$ can be derived in PCR* from previous lines $\mathbf{M}[p_j]$ and $\mathbf{M}[p_k]$. Similarly to Case A, condition (2) holds by assumption that p_i is a polynomial translation of an $\mathrm{R}_{c',d'}(\mathrm{lin})$-line D of size at most t.

Case C: If $p_i = x_j \cdot p_k$, for some $j \in [n]$ and $k < i$, was derived in P by the product rule from a previous line p_k, then $\mathbf{M}[p_i]$ can be derived in P' as follows:

If x_j does not appear with a positive power in p_k, then we can derive $\mathbf{M}[p_i] = \mathbf{M}[x_j \cdot p_k] = x_j \cdot \mathbf{M}[p_k]$ from $\mathbf{M}[p_k]$ via the product rule. Otherwise, assume that x_j appears with a positive power in p_k. Then we have

$$\mathbf{M}[p_k] = x_j \cdot f_1 + f_2$$

for some two multilinear polynomials f_1, f_2, where x_j does not appear with a positive power in f_1 and x_j does not appear with a positive power in f_2. We add the following PCR* proof-sequence to Q:

1. $x_j \cdot f_1 + f_2$	this is $\mathbf{M}[p_k]$
2. $\bar{x}_j \cdot (x_j \cdot f_1 + f_2)$	product of (1)
3. $(1 - \bar{x}_j) \cdot (x_j \cdot f_1 + f_2)$	(1) minus (2)
4. $x_j \cdot \bar{x}_j$	Boolean axiom
5. $(x_j \cdot \bar{x}_j) \cdot f_1$	product of (4)
6. $(1 - \bar{x}_j) \cdot (x_j \cdot f_1 + f_2) + (x_j \cdot \bar{x}_j) \cdot f_1$	(3) plus (5)
7. $x_j + \bar{x}_j - 1$	Boolean axiom
8. $(x_j + \bar{x}_j - 1) \cdot f_2$	product of (7)
9. $(1 - \bar{x}_j) \cdot (x_j \cdot f_1 + f_2) + (x_j \cdot \bar{x}_j) \cdot f_1 + (x_j + \bar{x}_j - 1) \cdot f_2$	(6) plus (8)

The last line (line 9) equals $x_j \cdot f_1 + x_j \cdot f_2 = \mathbf{M}[x_j \cdot p_k] = \mathbf{M}[p_i]$, which is the desired line.

Observe that (by opening brackets) every line in the sequence above is a linear combination of at most four of the following polynomials:

$$x_j \cdot \bar{x}_j, \quad x_j \cdot f_1, \quad f_2, \quad \bar{x}_j \cdot x_j \cdot f_1, \quad \bar{x}_j \cdot f_2, \quad x_j \cdot f_2. \tag{6}$$

We need the following claim:

Claim 5.12. *Every polynomial in (6) can be written as a sum* $\mathbf{M}[\widetilde{D}_1]+\mathbf{M}[\widetilde{D}_2]$*, such that* $\widetilde{D}_1,\widetilde{D}_2$ *are (possibly zero) polynomial translations of* $\mathrm{R}_{c',d'+1}(\mathrm{lin})$*-lines of degree* $O(t)$*.*

Proof of claim: The first polynomial $\bar{x}_j \cdot x_j$ is of the required form since it is a translation of a clause. We now consider the rest of the polynomials in (6).

Consider the polynomials f_1 and f_2. By assumption, we know that $x_j \cdot f_1 + f_2 = \mathbf{M}[p_k] = \mathbf{M}[\widetilde{D}]$, for some $\mathrm{R}_{c',d'}(\mathrm{lin})$-line D of size at most t, where x_j does not appear in f_1 and in f_2. Therefore,

$$f_1 = \mathbf{M}[\widetilde{D}]\restriction_{x_j=1} - \mathbf{M}[\widetilde{D}]\restriction_{x_j=0} = \mathbf{M}\left[\widetilde{D}\restriction_{x_j=1}\right] - \mathbf{M}\left[\widetilde{D}\restriction_{x_j=0}\right], \quad \text{and}$$

$$f_2 = \mathbf{M}\left[\widetilde{D}\restriction_{x_j=0}\right]$$

(where the notation $p\restriction_{x_j=b}$ means that we assign the value b to the variable x_j in the polynomial p).

We thus get:

$$x_j \cdot f_1 = x_j \cdot \mathbf{M}\left[\widetilde{D}\restriction_{x_j=1}\right] - x_j \cdot \mathbf{M}\left[\widetilde{D}\restriction_{x_j=0}\right] = \mathbf{M}\left[x_j \cdot \widetilde{D}\restriction_{x_j=1}\right] - \mathbf{M}\left[x_j \cdot \widetilde{D}\restriction_{x_j=0}\right],$$

where $x_j \cdot \widetilde{D}\restriction_{x_j=1}$ and $x_j \cdot \widetilde{D}\restriction_{x_j=0}$ are both polynomial translations of $\mathrm{R}_{c',d'+1}(\mathrm{lin})$-lines, of degree at most $t+1$.

The rest of the polynomials in (6), namely, f_2, $\bar{x}_j \cdot x_j \cdot f_1$, $\bar{x}_j \cdot f_2$, $x_j \cdot f_2$, can be treated in a similar manner (note also that $\bar{x}_j$ does not appear in f_1 and f_2). $\blacksquare_{\text{Claim}}$ $\square$

Notice that if a polynomial translation $\widetilde{D}$ of an $\mathrm{R}_{c',d'+1}(\mathrm{lin})$-line D is of degree at most $|\pi|$, then D is of size at most $O(n \cdot |\pi|)$ (for constants c', d'). Thus, Proposition 5.11 shows that we can transform the PC refutation π' from Step II into a PCR* refutation $\pi^\star$ of the same CNF, in which every line is a sum $\sum_{i\in I}\mathbf{M}[\widetilde{D}_i]$ such that:

(1) $|I|$ is constant (independent of n, c, d);
(2) every $\widetilde{D}_i$ is a polynomial translation of some $\mathrm{R}_{c',d'+1}(\mathrm{lin})$-line D_i such that the size $|D_i|$ is polynomial in the size $|\pi|$ of the original refutation π (for constants c', d' independent of n).
(3) The number of lines in $\pi^\star$ is polynomially bounded in the number of lines in π.

Note again that the new PCR* proof may contain the "negative" variables $\bar{x}_1, \ldots, \bar{x}_n$.

Step IV. We now show that every PCR* proof-line in $\pi^\star$ has a certain simple depth-3 arithmetic formula. We shall use the fact that $\mathrm{R}_{c,d}(\mathrm{lin})$-lines are close to a product of d symmetric polynomials, and the fact that multilinear symmetric polynomials can be computed by small ordered formulas (of depth-3) over large enough fields [Ben80] (cf. [Tza08] for a proof).

We say that an arithmetic formula Φ is a $\Sigma\Pi\Sigma$ formula if every path from the root to the leaf in the formula tree starts with a plus gate and the number of alternations in the path between plus and product gates is at most two, where field elements $\alpha \in \mathbb{F}$ can label any edge e in the formula, meaning that the polynomial computed in the tail of e (i.e., the node the edges e emanates from) is multiplied by α. In other words, Φ can be written as a sum of products of linear polynomials.

We need the following proposition, proved in [RT08b]:

Proposition 5.13 ([RT08b], Proposition 7.27)**.** *Let* $\mathbb{F}$ *be a field such that* $|\mathbb{F}| > n$*. For a constant* c*, let* $X_1, \ldots, X_c$ *be* c *finite sets of variables (not necessarily disjoint), where* $\sum_{i=1}^{c} |X_i| = n$ *. Let* $f_1, \ldots, f_c$ *be* c *symmetric polynomials over* $X_1, \ldots, X_c$ *(over the field* $\mathbb{F}$*), respectively. Then, there is a* $\Sigma\Pi\Sigma$ *formula* Φ *for* $\mathbf{M}[f_1 \cdots f_c]$ *of size polynomial (in* n*), such that* all bottom level linear forms consist of only a single variable *(that is,* $ax_i + b$*, for some* $a, b \in \mathbb{F}$*).*

Observation: Note that for any order on variables, every $\Sigma\Pi\Sigma$ formula Φ as in Proposition 5.13 can be transformed into an *ordered formula* with the same size: since all products are of linear forms, each with a *single* variable, for any order $\preceq$ on variables one can order the products in the formula in a way that respects $\preceq$.

The key lemma of the simulation is the following:

Lemma 5.14. *Let $\mathbb{F}$ be a field such that $|\mathbb{F}| > n$. Let s, t be two constants, let D be an $\mathrm{R}_{s,t}(\mathrm{lin})$-line with n variables and let $\widetilde{D}$ be the polynomial translation of D. Then, $\mathbf{M}\left[\widetilde{D}\right]$ has a $\Sigma\Pi\Sigma$ formula Φ of size polynomial in $|D|$ over $\mathbb{F}$, such that all bottom level linear forms consist of only a single variable (that is, $ax_i + b$, for some $a, b \in \mathbb{F}$).*

Proof. Assume that the underlying variables of D are $\vec{x} = \{x_1 \ldots, x_n\}$.[6] By assumption, we can partition the disjunction D into a *constant number t* of disjuncts, where each disjunct is a (possibly empty translation of a) clause C (if there is more than one clause in D we combine all the clauses into a single clause) and all other disjuncts have the following form:

$$\bigvee_{i=1}^{m} (\vec{a} \cdot \vec{x} = \ell_i)\,, \tag{7}$$

where the ℓ_i's are integers, m is bounded by $|D|$ and $\vec{a}$ denotes a vector of n *constant* integer coefficients, each having absolute value at most s.

Suppose that the clause C is $\bigvee_{i\in I} x_i \vee \bigvee_{j\in J} \neg x_j$, and let

$$q = \prod_{i\in I}(x_i - 1) \cdot \prod_{j\in J} x_j \tag{8}$$

be the polynomial representing C.

Consider a disjunct as shown in (7). Since the coefficients $\vec{a}$ are constants (having absolute value at most s), $\vec{a} \cdot \vec{x}$ can be written as a sum of constant number of linear forms, each with the *same* constant coefficient. In other words, $\vec{a} \cdot \vec{x}$ can be written as $z_1 + \ldots + z_d$, for some constant d (depending on s only), where for all $i \in [d]$:

$$z_i := b \cdot \sum_{j\in J} x_j\,, \tag{9}$$

for some $J \subseteq [n]$ and some constant integer b. We shall assume without loss of generality that d is the same constant for every disjunct of the form (7) in D (otherwise, take d to be the maximal such d). Thus, (7) is translated (as in Definition 5.6) into:

$$\prod_{i=1}^{m} (z_1 + ... + z_d - \ell_i)\,. \tag{10}$$

By fully expanding the product in (10), we arrive at:

$$\sum_{r_1+\ldots+r_{d+1}=m} \left(\alpha_{\vec{r}} \cdot \prod_{k=1}^{d} z_k^{r_k}\right), \tag{11}$$

where the r_i's are non-negative integers, and where each $\alpha_{\vec{r}}$'s, for $\vec{r} = \langle r_1, \ldots, r_{d+1}\rangle$, is an integer coefficient.

[6] We will apply Lemma 5.14 on lines with $2n$ variables $\{x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n\}$. For the sake of simplicity, in this lemma we assume that our underlying variables are $\{x_1, \ldots, x_n\}$.

Claim 5.15. *The polynomial translation $\widetilde{D}$ of D is a linear combination (over $\mathbb{F}$) of polynomially (in $|D|$) many terms, such that each term can be written as*

$$q \cdot \prod_{k \in K} z_k^{r_k}, \tag{12}$$

where K is a collection of a constant number of indices, r_k's are non-negative integers, and the z_k's and q are as above (that is, the z_k's are linear forms, where each z_k has a single coefficient for all variables in it, as in (9), and q is from (8)).

Proof of claim: By assumption, the total number of disjuncts of the form (7) in D is $\le t$. In $\widetilde{D}$, we actually need to multiply at most t many polynomials of the form shown in (11) and the polynomial q.

For every $j \in [t]$ we write the (same) linear form in the jth disjunct as a sum of constantly many linear forms $z_{j,1} + \ldots + z_{j,d}$, where each (sub-)linear form $z_{j,k}$ has the same coefficient for every variable in it. Thus, $\widetilde{D}$ can be written as:

$$q \cdot \prod_{j=1}^{t} \left(\sum_{r_1+\ldots+r_{d+1}=m_j} \underbrace{\left(\alpha_{\vec{r}}^{(j)} \cdot \prod_{k=1}^{d} z_{j,k}^{r_k} \right)}_{(\star)} \right), \tag{13}$$

(where the m_j's are bounded by $|D|$, and the coefficients $\alpha_{\vec{r}}^{(j)}$ are as above except that here we add the index (j) to denote that they depend on the jth disjunct in D). Denote the maximal m_j, for all $j \in [t]$, by m_0. We have $m_0 \le |D|$. Note that since d is a constant (depending only on s) the number of summands in each of the big (middle) sums in (13) is polynomial in m_0, which is at most polynomial in $|D|$ (specifically, it is $\le \binom{m_0+d}{m_0} < (m_0+d)^d$). Therefore, since t is constant (independent of n), by expanding the outermost product in (13), we arrive at a sum of polynomially in $|D|$ many summands. Each summand in this resulting sum is a product of t terms (each of the form designated by $(\star)$ in Equation (13)) multiplied by q. But this is precisely the required form of a summand in (12); where also, since d, t are constants, $|K|$ is a constant independent of n. $\blacksquare_{\text{Claim}}$

To finish the proof of Lemma 5.14 it remains to apply the multilinearization operator (Definition 5.10) on $\widetilde{D}$, and verify that the resulting polynomial has the desired form. Since $\mathbf{M}[\cdot]$ is a linear operator, it suffices to show that when applying $\mathbf{M}[\cdot]$ on each summand in $\widetilde{D}$, as described in Claim 5.15, one obtains a polynomial that has a $\Sigma\Pi\Sigma$ formula of size polynomial in $|D|$ over $\mathbb{F}$, such that all bottom level linear forms consist of only a single variable. This is established in the following claim:

Claim 5.16. *(Under the same notation as in Claim 5.15) the polynomial $\mathbf{M}\left[q \cdot \prod_{k \in K} z_k^{r_k}\right]$ has a $\Sigma\Pi\Sigma$ formula (over $\mathbb{F}$) of polynomial-size in the number of variables n and with a plus gate at the root, such that all bottom level linear forms consist of only a single variable (that might be different for each linear form).*

Proof of claim: Note that a power of a symmetric polynomial is a symmetric polynomial in itself. Thus, since for any $k \in K$, z_k is a symmetric polynomial, $z_k^{r_k}$ is also symmetric. The polynomial q is a translation of a clause, hence it is a product of two symmetric polynomials (over different variables): the symmetric polynomial that is the translation of the disjunction of literals with positive signs, and the symmetric polynomial that is the translation of the disjunction of literals with negative signs. Therefore, $q \cdot \prod_{k \in K} z_k^{r_k}$ is a product of constant number of symmetric polynomials (over different, though possibly not disjoint, sets of variables). By Proposition 5.13,

$\mathbf{M}\big[q \cdot \prod_{k \in K} z_k^{r_k}\big]$ (where here the $\mathbf{M}[\cdot]$ operator operates on the $\vec{x}$ variables in the z_k's and q) is a polynomial for which there is a $\Sigma\Pi\Sigma$ polynomial-size (in n) formula such that all bottom level linear forms consist of only a single variable (over $\mathbb{F}$). $\blacksquare_{\text{Claim}}$ $\square$

Step V. In the previous step we obtained a PCR* refutation $\pi^\star = (q_1, \dots, q_r)$ of the CNF K with r polynomial in $|\pi|$, and such that every q_i can be computed by a $\Sigma\Pi\Sigma$ formula Q_i of polynomial-size in $|\pi|$, and where each bottom level in Q_i consists of only a single variable (that is, $ax_i + b$, for some $a, b \in \mathbb{F}$).

Note that $\pi^\star$ is *not* a legal PCR refutation of K since $\pi^\star$ used the extended PCR* product rule $\frac{p}{g \cdot p}$, for some polynomial g, while in PCR we only have the rule $\frac{p}{x \cdot p}$, for some variable x. We now show that we can add new PCR proof-sequences to $\pi^\star$ to obtain a PCR refutation of K with the appropriate properties:

Claim 5.17. *Assume that in $\pi^\star$ the polynomial $q_i = g \cdot p$ was derived from $q_j = p$ by the PCR* product rule. Then, there exists a PCR proof of Q_i from Q_j with size polynomial in $|Q_i|$ (where Q_i, Q_j are the corresponding formulas for q_i, q_j, respectively), such that every proof-line can be written as a $\Sigma\Pi\Sigma$ formula of polynomial-size in $|Q_i|$ in which each bottom level consists of only a single variable.*

Proof of claim: If g is a variable from $\{x_1, \dots, x_n, \bar{x}_1, \dots, \bar{x}_n\}$, then we are done. Otherwise, by construction of $\pi^\star$, the polynomial $q_i = g \cdot p$ is either an instance of Line 5 or of Line 8 in the PCR* proof-sequence described in Proposition 5.11. By Claim 5.12 and Lemma 5.14 we thus obtain that one of the following holds:

(1) $q_i = (x_j \cdot \bar{x}_j) \cdot f_1$ for $p = (x_j \cdot \bar{x}_j)$, such that $x_j, \bar{x}_j$ do not appear in f_1;
(2) $q_i = (x_j + \bar{x}_j - 1) \cdot f_2$ for $p = (x_j + \bar{x}_j - 1)$, such that $x_j, \bar{x}_j$ do not appear in f_2,

and where both f_1 and f_2 can be computed by a $\Sigma\Pi\Sigma$ formula Q_i of polynomial-size in $|\pi|$, and the bottom level linear polynomials in Q_i consists of only a single variable.

The proof of the claim now is straightforward. First, we derive from g in PCR the polynomial $g \cdot F_i$, for any i such that F_i is the polynomial computed by the ith product gate in Q_i. Each such proof of $g \cdot F_i$ can be carried out by induction on the degree of q_i. Then, we add together $g \cdot F_i$, for all i, which yields the desired $\Sigma\Pi\Sigma$ formula computing the polynomial q_i. Also, note that every proof-line in this derivation can be written as a $\Sigma\Pi\Sigma$ formula of polynomial-size in $|Q_i|$ such that each bottom level linear polynomial consists of only a single variable, and where the number of proof-lines is polynomial in $|Q_i|$. $\blacksquare_{\text{Claim}}$

By Claim 5.17 there exists a PCR refutation π'' of K of size polynomial in $|\pi|$ in which every line is a $\Sigma\Pi\Sigma$ formula in which each bottom level consists of only a single variable.

Since the formulas in π'' possibly contain the variables $\bar{x}_1, \dots, \bar{x}_n$, we need to take these variables out in order to construct our final PC refutation with only the $x_1, \dots, x_n$ variables. We do this by first substituting every variable $\bar{x}_i$, $i \in [n]$, by $(1 - x_i)$ in every line of π'', and then adding required PC lines to transform the resulting sequence into a legal PC refutation.

Let τ denote the linear transformation that maps the variables $\bar{x}_i$, for any $i \in [n]$, to $(1 - x_i)$, and denote $p{\restriction}\tau$ the polynomial p under the transformation τ.

Claim 5.18. *Let Π be the sequence of polynomials $\pi''{\restriction}\tau$ obtained from π'' by applying τ to every proof-line. Then, there exists a PC refutation Π' refuting the same CNF as π'' does, with only a polynomial increase in numbers of lines, and whose each line can be computed by a $\Sigma\Pi\Sigma$ formula of polynomial-size in $|\pi|$, such that each bottom level in the formula consists of only a single variable.*

Proof of claim: By induction on the number of lines in π''.
Base case: Axioms turn into axioms (the axiom $x_i + \bar{x}_i - 1$ turns into the polynomial 0, which can

be ignored in the refutation).
Induction step:
Case 1: Addition rule in π'' stays legal in Π.
Case 2: Product rule: if we derive $x_i \cdot p$ from p in π'', for some $i \in [n]$, then in Π we derive $x_i \cdot (p \restriction \tau)$ from $p \restriction \tau$, which is legal.

Assume we derived $\bar{x}_i \cdot p$ from p. Then, we need to derive $(1 - x_i) \cdot (p \restriction \tau)$ from $p \restriction \tau$. For this, first derive $x_i \cdot p \restriction \tau$, and then use the addition rule to add $p \restriction \tau$ with $-x_i \cdot p \restriction \tau$.

Note also that all lines in the new PC refutation Π' can be written as $\Sigma\Pi\Sigma$ formulas of polynomial-size in $|\pi|$, and where each bottom level in the formula consists of only a single variable. $\blacksquare_{\text{Claim}}$

Now, since every proof-line in the refutation Π' obtained from Claim 5.18 can be written as a $\Sigma\Pi\Sigma$ ordered formula of size polynomial in $|\pi|$ in which all bottom levels are linear forms $ax_i + b$, for some $a, b \in \mathbb{F}$ and some $i \in [n]$, *every proof-line in* Π' *can be written as an ordered formula of size* $O(|\pi|)$. This is because we can simply order the linear forms hanging from any product gate in a way that respects the order $\preceq$. Also, Since the number of proof-lines in Π' is polynomial in $|\pi|$, we conclude that OFPC polynomially simulates $\mathrm{R}^0(\mathrm{lin})$.

This concludes the proof of Theorem 5.5.

5.4. **Short proofs and separations.** For natural numbers $m > n$, denote by $\neg\mathrm{FPHP}_n^m$ the following unsatisfiable collection of polynomials:

$$\begin{array}{lll} \text{Pigeons}: & \forall i \in [m], & (1 - x_{i,1}) \cdots (1 - x_{i,n}) \\ \text{Functional}: & \forall i \in [m]\, \forall k < \ell \in [n], & x_{i,k} \cdot x_{i,\ell} \\ \text{Holes}: & \forall i < j \in [m]\, \forall k \in [n], & x_{i,k} \cdot x_{j,k} \end{array} \tag{14}$$

As a consequence of the polynomial simulation of $\mathrm{R}^0(\mathrm{lin})$ by OFPC, and the upper bounds on $\mathrm{R}^0(\mathrm{lin})$ refutations demonstrated in [RT08a], we get the following result:

Corollary 5.19. *For any linear order on the variables, and for any $m > n$ there are polynomial-size (in n)* OFPC *refutations of the m to n pigeonhole principle* $\neg\mathrm{FPHP}_n^m$ *(over fields of characteristic zero).*

The contradiction $\neg\mathrm{FPHP}_n^m$ is a direct translation of the CNF formula for the m to n functional pigeonhole principle. Thus, by known lower bounds, OFPC is strictly stronger than resolution and is separated from bounded depth Frege. On the other hand, Razborov [Razb98] and subsequently Impagliazzo et al. [IPS99] gave exponential lower bounds on the size of PC-refutations of a different *low degree* version of the Functional Pigeonhole Principle. In this low degree version the Pigeons polynomials in (14) are replaced by $1 - (x_{i,1} + \ldots + x_{i,n})$, for all $i \in [m]$. It is not hard to show (via reasoning inside $\mathrm{R}^0(\mathrm{lin})$) that OFPC admits polynomial-size refutations also for this low-degree version of the functional pigeonhole principle. This shows that OFPC is strictly stronger than PC (under the size measures defined for OFPC and PC).

The Tseitin graph tautologies were proved to be hard tautologies for several propositional proof system. We refer the reader to [RT08a], Definition 6.5, for the precise definition of the (generalized, mod p) Tseitin tautologies. We have the following:

Corollary 5.20. *Let G be an r-regular graph with n vertices, where r is a constant, and fix some modulus p. Then, for any linear order on the variables there are polynomial-size (in n)* OFPC *refutations (over fields of characteristic* 0*) of the corresponding Tseitin* mod p *formulas over G.*

This stems from the $\mathrm{R}^0(\mathrm{lin})$ polynomial-size refutations of the Tseitin mod p formulas demonstrated in [RT08a]. From the known exponential lower bounds on PCR (and PC and resolution) refutation size of Tseitin mod p tautologies (when the underlying graphs are appropriately expanding; cf. [BGIP01, BSI99, ABSRW04]), and for the polynomial simulation of PCR by OFPC, we conclude that OFPC is strictly stronger than PCR.

6. Useful lower bounds on products of ordered polynomials

In this section we show that the ordered formula size of certain polynomials can increase exponentially when multiplying the polynomials together. We use this to suggest an approach for lower bounding the size of OFPC proofs in Section 6.1. We use a method of partial derivatives matrix introduced by Nisan to obtain exponential-size lower bounds on noncommutative formulas in [Nis91]. We shall state the results of Nisan using the model of *algebraic branching programs* (ABP) (this will help us in the example of conditional lower bound discussed in the next sub-section). Algebraic branching programs can polynomially simulate noncommutative formulas, and hence also ordered formulas.

Definition 6.1 (ABP)**.** *An algebraic branching program is a directed acyclic graph with one node of in-degree zero, called the* source*, and one node of out-degree zero called the* sink*. The graph is partitioned into levels* $0, \ldots, d$*, and nodes in level* $i = 0, \ldots, d-1$ *have edges only to level* $i+1$*. The source is the only node in level* 0 *and the sink is the only node in level* d*. The edges of the graph are labeled with homogenous linear forms in the variables* $x_1, \ldots, x_n$ *and coefficients from a field* $\mathbb{F}$ *(i.e., linear polynomials with no free terms). An ABP* computes *a noncommutative polynomial in* $\mathbb{F}\langle x_1, \ldots, x_n\rangle$ *as follows: every directed path from the source to a node* v *computes the product of linear forms on the path in the order of their appearance. The node* v *computes the* sum *of all the polynomials computed by all the directed pathes from source to* v*. An ABP computes the noncommutative polynomial computed at its sink.*

Note that an ABP computes only homogenous polynomials. We have the following simple structural property, showing that the noncommutative formula size of a noncommutative polynomial is polynomially proportional to its ABP size:

Lemma 6.2 (Lemma 2.2 in [RS05])**.** *Let* f *be a noncommutative polynomial which is computed by a noncommutative formula of size* s*. Assume that the free term of* f *is zero (in other words,* $f(0, \ldots, 0) = 0$*). Then there exist* $\deg(f)$ *noncommutative ABP's such that the* i*th ABP computes the homogeneous component of* f *of degree* i*, for* $i = 1, \ldots, \deg(f)$*. Moreover, the size of each of these ABP's is* $O(s^2)$*.*

Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be a commutative polynomial. Recall that $[\![f]\!]$ is the *noncommutative* polynomial obtained from f by ordering the products in every monomial in accordance to the linear order $\preceq$, and that an ordered formula computing f is a noncommutative formula computing $[\![f]\!]$. Thus, if we denote by $OF(f)$ the minimal size of an ordered formula computing f and by $A(f)$ the minimal total ABP-sizes of a sequence of ABP's computing the homogenous components $f^{(1)}, \ldots, f^{(\deg(f))}$ of f, then by Lemma 6.2, we have:

$$OF(f) \geq (A(f))^{O(1)}$$

(note that $\deg(f) \leq OF(f)$, because f is a formula). To conclude, a super-polynomial lower bound on the ordered formula size of $f \in \mathbb{F}[x_1, \ldots, x_n]$ follows from a super-polynomial lower bound on the minimal total ABP-sizes of a sequence of ABP's computing the homogenous components of the noncommutative polynomial $[\![f]\!]$.

Proposition 6.3. *Let* $\mathbb{F}$ *be a field,* $X := \{x_1, \ldots, x_n\}$ *be a set of variables and* $\preceq$ *some linear order on* X*. Then, for any natural numbers* $m \leq n$ *and* $d \leq \lfloor n/m \rfloor$*, there exist polynomials* $f_1, \ldots, f_d$

from $\mathbb{F}[x_1,\ldots,x_n]$*, such that every* f_i *can be computed by an ordered formula of size* $O(m)$ *and every ABP computing* $[\![\prod_{i=1}^d f_i]\!]$ *has size* 2^d*.*

Proof. First, note that it is sufficient to prove the proposition for $m = 2$ and any $d \le \lfloor n/2 \rfloor$. (Because, assume that the proposition holds for $m = 2$ and any $d \le \lfloor n/2 \rfloor$. And let m', d' be such that $m' \le n$ and $d' \le \lfloor n/m' \rfloor$. By assumption, for $m = 2$ and $d' \le \lfloor n/m' \rfloor \le \lfloor n/2 \rfloor$, there are $f_1,\ldots,f_{d'}$ from $\mathbb{F}[x_1,\ldots,x_n]$ that can be computed by ordered formulas of size constant [that is, $O(2)$, and hence of size $O(m')$], and such that every ABP computing $[\![\prod_{i=1}^{d'} f_i]\!]$ has size $2^{\Omega(d')}$.)

Thus, let $m = 2$ and $d \le \lfloor n/2 \rfloor$. Assume without loss of generality that the linear order $\preceq$ is such that $x_1 \preceq x_2 \preceq \ldots \preceq x_n$. Abbreviate the variables $x_1,\ldots,x_d$ as $y_1,\ldots,y_d$, respectively, and abbreviate the variables $x_{d+1},\ldots,x_{2d}$ as $z_1,\ldots,z_d$, respectively (that is, the y_i's and z_i's are just abbreviations for their corresponding x_i variables, introduced to simplify the writing). We thus have $y_1 \preceq \ldots \preceq y_d \preceq z_1 \preceq \ldots \preceq z_d$.

For every $i = 1,\ldots,d$, define the following polynomial (that obviously has a constant size ordered formula):

$$f_i := (y_i + z_i)\,.$$

Define

$$\mathsf{HARD}_d := \prod_{i=1}^{d} f_i = \prod_{i=1}^{d} (y_i + z_i)\,.$$

We show that every ABP computing $[\![\mathsf{HARD}_d]\!]$ (under $\preceq$) is of size at least 2^d. Note that HARD_d is a homogenous noncommutative and multilinear polynomial of degree d. To lower bound the ABP size of a homogenous noncommutative polynomial we use the rank argument introduced in [Nis91]. Nisan defined the matrix $M_k(f)$ associated with a homogenous noncommutative polynomial f as follows:

Definition 6.4 ([Nis91])**.** *Let* $f \in \mathbb{F}\langle x_1,\ldots,x_n\rangle$ *be a noncommutative homogenous polynomial of degree* d*. For every* $0 \le k \le d$*, we define* $M_k(f)$ *to be a matrix of dimension* $n^k \times n^{d-k}$ *as follows: (i) there is a row corresponding to every degree* k *noncommutative monomial over the variables* $\{x_1,\ldots,x_n\}$*, and a column corresponding to every degree* $d-k$ *noncommutative monomial over the variables* $\{x_1,\ldots,x_n\}$*; (ii) for every degree* k *monomial* $\mathscr{M}$ *and every degree* $d-k$ *monomial* $\mathscr{N}$*, the entry in* $M_k(f)$ *on the row corresponding to* $\mathscr{M}$ *and column corresponding to* $\mathscr{N}$ *is the coefficient of the degree* d *monomial* $\mathscr{M}\cdot\mathscr{N}$ *in* f*.*

Theorem 6.5 ([Nis91] Theorem 1)**.** *Let* f *be a degree* r *homogenous noncommutative polynomial. Then, every ABP computing* f *has size at least* $\sum_{k=0}^{r} \mathrm{rank}\,(M_k(f))$*.*

In view of Theorem 6.5, it suffices to prove the following claim:

Claim 6.6. *For any* $0 \le k \le d$*:* $\mathrm{rank}(M_k([\![\mathsf{HARD}_d]\!])) \ge \binom{d}{k}$.

Proof of claim: Consider the matrix $M_k([\![\mathsf{HARD}_d]\!])$. Let $\mathbf{A}_k$ be the matrix obtained from $M_k([\![\mathsf{HARD}_d]\!])$ by removing all rows and columns excluding the following rows and columns:

(1) the rows corresponding to degree k multilinear monomials containing only y_i variables, such that the order of products in the monomial respects $\preceq$;

(2) the columns corresponding to degree $d-k$ multilinear monomials containing only z_i variables, such that the order of products in the monomial respects $\preceq$.

Consider a degree k monomial $\mathscr{M} = y_{i_1}\cdots y_{i_k}$, where $i_1 < \ldots < i_k$. Let $J = [d] \setminus \{i_1,\ldots,i_k\}$. We can denote the elements of J as $\{j_1,\ldots,j_{d-k}\}$, where $j_1 < \ldots < j_{d-k}$. Observe that the monomial $\mathscr{M}$ has on its corresponding row in $\mathbf{A}_k$ only zeros, except for a single 1 in the position (that is, column) corresponding to the degree $d-k$ monomial $\mathscr{N} = z_{j_1}\cdots z_{j_{d-k}}$. (Indeed, note that the coefficient of the degree d monomial $\mathscr{M}\cdot\mathscr{N}$ in $[\![\mathsf{HARD}_d]\!]$ is 1.)

Note that $\mathbf{A}_k$ contains $\binom{d}{k}$ rows corresponding to all possible degree k multilinear monomials $\mathscr{M}$ in the $\overline{y}$ variables whose product order respect $\preceq$. Similarly, $\mathbf{A}_k$ contains $\binom{d}{k}$ columns corresponding to all possible degree $d-k$ multilinear monomials $\mathscr{N}$ in the $\overline{z}$ variables whose product order respect $\preceq$. By the previous paragraph: (i) each of the rows in $\mathbf{A}_k$ has only one nonzero entry; and (ii) for every row, the nonzero entry is in a *different* column from those of other rows. We then conclude that $\mathbf{A}_k$ is a permutation matrix. Therefore:

$$\operatorname{rank}(\mathbf{A}_k) = \binom{d}{k}.$$

The claim follows since clearly $\operatorname{rank}(\mathbf{A}_k) \leq \operatorname{rank}(M_k\left(\llbracket \mathsf{HARD}_d \rrbracket\right))$. $\blacksquare_{\text{Claim}}$

By the claim and by Theorem 6.5, we conclude that the ABP size of $\llbracket \mathsf{HARD}_d \rrbracket$ is at least

$$\sum_{k=0}^{d} \operatorname{rank}\left(\mathbf{A}_k\right) = \sum_{k=0}^{d} \binom{d}{k} = 2^d .$$

□

6.1. **Suggested lower bound approach.** Here we discuss a simple possible approach intended to establish lower bounds on OFPC proofs, roughly, by reducing OFPC lower bounds to PC degree lower bounds and using the bound in Section 6 (Proposition 6.3).

Setting 1: Let $Q_1(\overline{x}), \ldots, Q_m(\overline{x})$ be a collection of constant degree (independent of n) polynomials from $\mathbb{F}[x_1, \ldots, x_n]$ with no common solutions in $\mathbb{F}$, such that m is polynomial in n. Let $f_1(\overline{y}), \ldots, f_n(\overline{y})$ be m homogenous polynomials of the same degree from $\mathbb{F}[y_1, \ldots, y_\ell]$, such that the ordered formula size of each $f_i(\overline{y})$ (for some fixed linear order on the variables) is polynomial in n and such that the $f_i(\overline{y})$'s do not have common variables (that is, each $f_i(\overline{y})$ is over disjoint sets of variables from $\overline{y}$). Suppose that for any distinct $i_1, \ldots, i_d \in [n]$ the ABP size of $\llbracket \prod_{j=1}^{d} f_{i_j}(\overline{y}) \rrbracket$ is $2^{\Omega(d)}$.

Note: By the proof of Proposition 6.3, the conditions above are easy to achieve. Indeed, the $f_i(y_i, z_i)$'s defined in the proof of Proposition 6.3 have these properties: homogeneity, same degrees for all f_i's and disjointness of variables, and an exponential increase in ABP sizes computing products of the f_i's.

Consider the polynomials $Q_1(\overline{x}), \ldots, Q_m(\overline{x})$ after applying the substitution:

$$x_i \mapsto f_i(\overline{y}) . \tag{15}$$

In other words, consider

$$Q_1(f_1(\overline{y}), \ldots, f_n(\overline{y})), \ldots, Q_m(f_1(\overline{y}), \ldots, f_n(\overline{y})) . \tag{16}$$

Note that (16) is also unsatisfiable over $\mathbb{F}$.

We suggest to lower bound the OFPC refutation size of (16), based on the following simple idea: it is known that some families of unsatisfiable collections of polynomials require linear $\Omega(n)$ *degree* PC refutations (where n is the number of variables). In other words, every refutation of these polynomials must contain some polynomial of linear degree. By definition, also every OFPC refutation of these polynomials must contain some polynomial of linear in n degree.

For the purpose of super-polynomial lower bounds even a weaker $\omega(\log n)$ degree lower bound on PC refutations would suffice. Hence, assume that the initial polynomials $Q = \{Q_1(\overline{x}), \ldots, Q_m(\overline{x})\}$ in the $x_1, \ldots, x_n$ variables require $\omega(\log n)$ degree PC refutations. This means that every PC refutation of Q contains some polynomial h of degree $\omega(\log n)$. Then, we might expect that every PC refutation of its *substitution instance* (16) contains a polynomial $g \in \mathbb{F}[\overline{y}]$ which is a substitution instance (under the substitution (15)) of an $\omega(\log n)$ degree polynomial in the $\overline{x}$ variables. This, in turn, leads (under some conditions; see below) to a lower bound on OFPC refutations.

An example of sufficient conditions for super-polynomial OFPC lower bounds, are the following: assume that every PC refutation of (16) contains a polynomial g so that one of g's homogenous components is a substitution instance of a degree $\omega(\log n)$ multilinear polynomial from $\mathbb{F}[x_1, \ldots, x_n]$. We formalize this argument:

Example: **Conditional** OFPC **size lower bounds.** (Assume the above *Setting 1* and notations.)
If: every PC refutation of (16) that has polynomial in n number of proof-lines contains a polynomial $g \in \mathbb{F}[y_1, \ldots, y_\ell]$ such that for some $t = \text{poly}(n)$, the t-th homogenous component $g^{(t)}$ of g is a *substitution instance* of a degree $\omega(\log n)$ multilinear polynomial from $\mathbb{F}[x_1, \ldots, x_n]$ (under the substitution (15));
Then: every OFPC refutation of (16) is of super-polynomial size (in n).

Proof of example: It suffices to show that any ordered formula of g is of super-polynomial size in n. By Lemma 6.2, it suffices to show that $[\![g^{(t)}]\!]$, the t-th homogenous component of $[\![g]\!]$ (note that $[\![g]\!]^{(t)} = [\![g^{(t)}]\!]$), requires an ABP of super-polynomial size in n.

By assumption, $g^{(t)}$ is a substitution instance of some degree $\omega(\log n)$ multilinear polynomial $h \in \mathbb{F}[x_1, \ldots, x_n]$. Since $g^{(t)}$ is homogenous and all the $f_i(\overline{y})$'s have the same degree and are homogenous, h must be homogenous too. Since h is multilinear we can write $h = \sum_{j \in J} b_j \mathscr{M}_j$, where the $\mathscr{M}_j$'s are multilinear monomials in the $\overline{x}$ variables and b_j are coefficients from $\mathbb{F}$. Now, consider some single monomial $\mathscr{M}$ from $\sum_{j \in J} b_j \mathscr{M}_j$. By multilinearity and homogeneity of h every other monomial $\mathscr{M}' \neq \mathscr{M}$ in h must contain an x_i variable that does not appear in $\mathscr{M}$. We can assign 0 to such x_i. Doing this for every monomial $\mathscr{M}' \neq \mathscr{M}$, we get that h (under this partial assignment to the $\overline{x}$ variables) is equal to $b\mathscr{M}$, for some coefficient $b \in \mathbb{F}$. In a similar manner, by disjointness of the variables in the $f_i(\overline{y})$'s, there exists a partial assignment $\rho : \overline{y} \to \{0\}$, such that $g^{(t)} \restriction \rho$ is just a substitution instance (under the substitution (15)) of a single multilinear *monomial* of degree $\omega(\log n)$ in the $\overline{x}$ variables. This means that $g^{(t)} \restriction \rho$ is the product of $\omega(\log n)$ distinct $f_i(\overline{y})$'s (multiplied by b). Therefore, by assumption on the $f_i(\overline{y})$'s, every ABP computing $[\![g^{(t)}]\!]$ is of size $2^{\omega(\log n)}$, which is super-polynomial in n. ■

Remark: The conditional lower bound example above inherits its hardness from the hard polynomials in Proposition 6.3. Since the hard polynomial HARD_d in the proof of Proposition 6.3 is hard for ordered formulas (and ABP's) only with respect to a *specific* order on variables, the family of polynomials in (16) are (conditionally) hard for OFPC only with respect to this specific order.

According to the lower bound suggested above, a natural starting point to search for hard candidates for OFPC might be the following: assume that the substitution (15) consists of $f_1(y_{1,1}, \ldots, y_{1,n}), \ldots, f_n(y_{n,1}, \ldots, y_{n,n})$, where $f_i(y_1, \ldots, y_n)$ has *exponentially many monomials*, while still having small ordered formulas, for any $i = 1, \ldots, n$; e.g.,

$$f_i(y_{i,1}, \ldots, y_{i,n}) = (y_{i,1} + y_{i,2}) \cdots (y_{i,(n/2)-1} + y_{i,n/2}).$$

(Then $\ell = n^2$ in the notation of (15).) Then, one might expect that the premise of the example for conditional OFPC size lower bounds above possibly hold. Intuitively, the (speculative) reason is that any PC refutation with a polynomial in n number of proof-lines would need to operate with the f_i's as "almost atomic formulas", since they include exponential many monomials.

Acknowledgments

I wish to thank Emil Jeřabek, Sebastian Müller, Pavel Pudlák, Neil Thapen and Youming Qiao for helpful discussions on issues related to this paper and the anonymous referees for many comments improving the exposition of this paper. I also wish to thank Ran Raz for suggesting this research direction, and Jan Krajíček for inviting me to give a talk at TAMC 2010 on this subject.

References

[ABSRW02] Michael Alekhnovich, Eli Ben-Sasson, Alexander A. Razborov, and Avi Wigderson. Space complexity in propositional calculus. *SIAM J. Comput.*, 31(4):1184–1211 (electronic), 2002. 5

[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)). 5.4

[AGP02] Albert Atserias, Nicola Galesi, and Pavel Pudlák. Monotone simulations of non-monotone proofs. *J. Comput. System Sci.*, 65(4):626–638, 2002. Special issue on complexity, 2001 (Chicago, IL). 1

[Ajt88] Miklós Ajtai. The complexity of the pigeonhole principle. In *Proceedings of the IEEE 29th Annual Symposium on Foundations of Computer Science*, pages 346–355, 1988. 1

[AKV04] Albert Atserias, Phokion G. Kolaitis, and Moshe Y. Vardi. Constraint propagation as a proof system. In *CP*, pages 77–91, 2004. 1, 1.1

[BGIP01] Samuel R. Buss, Dima Grigoriev, Russell Impagliazzo, and Toniann Pitassi. Linear gaps between degrees for the polynomial calculus modulo distinct primes. *J. Comput. System Sci.*, 62(2):267–289, 2001. Special issue on the 14th Annual IEEE Conference on Computational Complexity (Atlanta, GA, 1999). 5.4

[BIK+97] Samuel R. Buss, Russell Impagliazzo, Jan Krajíček, Pavel Pudlák, Alexander A. Razborov, and Jiří Sgall. Proof complexity in algebraic systems and bounded depth Frege systems with modular counting. *Comput. Complexity*, 6(3):256–298, 1996/97. (document)

[Ben80] Michael Ben-Or. Unpublished notes, 1980. 5.3

[BSI99] Eli Ben-Sasson and Russell Impagliazzo. Random CNF's are hard for the polynomial calculus. In *Proceedings of the IEEE 40th Annual Symposium on Foundations of Computer Science (New York, 1999)*, pages 415–421. IEEE Computer Soc., Los Alamitos, CA, 1999. 5.4

[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. 1.1, 2.3

[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. 2.2

[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). (document), 1.1, 3.1, 3.5, 4, 3.1

[IPS99] Russell Impagliazzo, Pavel Pudlák, and Jiří Sgall. Lower bounds for the polynomial calculus and the Gröbner basis algorithm. *Comput. Complexity*, 8(2):127–144, 1999. 5.4

[JQS10] Maurice Jansen, Youming Qiao, and Jayalal Sarma. Deterministic black-box identity testing π-ordered algebraic branching programs. *Electronic Colloquium on Computational Complexity (ECCC)*, TR10-015, February 2010. 1.1

[KPW95] Jan Krajíček, Pavel Pudlák, and Alan Woods. An exponential lower bound to the size of bounded depth Frege proofs of the pigeonhole principle. *Random Structures Algorithms*, 7(1):15–39, 1995. 1

[Kra08] Jan Krajíček. An exponential lower bound for a constraint propagation proof system based on ordered binary decision diagrams. *J. Symbolic Logic*, 73(1):227–237, 2008. 1, 1.1

[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. (document), 1, 1.1, 6, 6, 6.4, 6.5

[NW97] Noam Nisan and Avi Wigderson. Lower bounds on arithmetic circuits via partial derivatives. *Comput. Complexity*, 6(3):217–234, 1997.

[PBI93] Toniann Pitassi, Paul Beame, and Russell Impagliazzo. Exponential lower bounds for the pigeonhole principle. *Comput. Complexity*, 3(2):97–140, 1993. 1

[Pit97] Toniann Pitassi. Algebraic propositional proof systems. In *Descriptive complexity and finite models (Princeton, NJ, 1996)*, volume 31 of *DIMACS Ser. Discrete Math. Theoret. Comput. Sci.*, pages 215–244. Amer. Math. Soc., Providence, RI, 1997. 1

[Pud99] Pavel Pudlák. On the complexity of the propositional calculus. In *Sets and proofs (Leeds, 1997)*, volume 258 of *London Math. Soc. Lecture Note Ser.*, pages 197–218. Cambridge Univ. Press, Cambridge, 1999. 1

[Razb98] Alexander A. Razborov. Lower bounds for the polynomial calculus. *Comput. Complexity*, 7(4):291–324, 1998. 5.4

[Raz06] Ran Raz. Separation of multilinear circuit and formula size. *Theory of Computing, Vol. 2, article 6*, 2006. 1

[Raz09] Ran Raz. Multi-linear formulas for permanent and determinant are of super-polynomial size. *J. ACM*, 56(2), 2009. 1, 1.1

[RS05] Ran Raz and Amir Shpilka. Deterministic polynomial identity testing in non commutative models. *Comput. Complexity*, 14(1):1–19, 2005. 1.1, 2.1, 2.3, 6.2

[RT08a] Ran Raz and Iddo Tzameret. Resolution over linear equations and multilinear proofs. *Ann. Pure Appl. Logic*, 155(3):194–224, 2008. arXiv:0708.1529. 1, 1.1, 5, 5.2, 5.2, 5.2, 5.3, 5.3, 5.4, 5.4, 5.4

[RT08b] Ran Raz and Iddo Tzameret. The strength of multilinear proofs. *Comput. Complexity*, 17(3):407–457, 2008. 1, 1.1, 5.3, 5.13

[Seg07] Nathan Segerlind. Nearly-exponential size lower bounds for symbolic quantifier elimination algorithms and OBDD-based proofs of unsatisfiability. *Electronic Colloquium on Computational Complexity (ECCC)*, TR07-009, January 2007. 1, 1.1

[Tza08] Iddo Tzameret. *Studies in Algebraic and Propsitional Proof Complexity*. PhD thesis, Tel Aviv University, 2008. 1, 5.3

[Waa97] Stephan Waack. On the descriptive and algorithmic power of parity ordered binary decision diagrams. In *STACS*, pages 201–212, 1997. 1.1

Institute for Theoretical Computer Science, the Inst. for Interdisciplinary Information Science, FIT building, Tsinghua University, Beijing, 100084, China. http://itcs.tsinghua.edu.cn/~tzameret

E-mail address: tzameret@tsinghua.edu.cn