Electronic Colloquium on Computational Complexity, Report No. 97 (2010)

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 analogue 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). This is an algebraic propositional proof system that operates with noncommutative polynomials in which the order of products in all monomials respects a fixed linear order on the variables, and where proof-lines are written as noncommutative formulas. We show that the latter proof system 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.

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

CONTENTS

Date: March 2010.

*Mathematical Institute, Academy of Sciences of the Czech Republic, Prague, Czech Rep. Supported by The Eduard Čech Center for Algebra and Geometry and The John Templeton Foundation.

ISSN 1433-8092

1. Introduction

This work attempts to gather certain facts about algebraic proof systems establishing propositional tautologies, in which proof lines are written as noncommutative algebraic formulas (noncommutative formulas, for short). Our general motivation here is to develop a method to lower bound the size of certain propositional proofs via a rank argument, similar to that used in algebraic circuit complexity. 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], is an especially transparent rank argument.

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 et al. [Pud99, AGP02] studied proofs based on monotone circuits—motivated by 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.

1.1. Results and related works.

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 get the values 0 or 1. Our two proof systems extend PC as follows:

(1) PC over noncommutative formulas: NFPC. This proof system operates with noncommutative polynomials over a field, written as (arbitrarily chosen)[1] noncommutative formulas. The rules of addition and multiplication are similar to PC, except that multiplication is done *either from left or right.* We also add a a Boolean axiom $x_i x_j - x_j x_i$ that expresses the fact that for $0, 1$ values to the variables, multiplication is in fact commutative.

(2) PC over ordered formulas: OFPC. This proof system is PC operating with noncommutative polynomials in which the order of products in all monomials respects a fixed linear order on the variables, and where proof-lines are written as (arbitrarily chosen) noncommutative formulas.

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 algebraic 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 analogue 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 algebraic formulas from previous algebraic formulas via the polynomial-ring axioms (that is, associativity, commutativity, distributivity and the zero and unit elements rules). In this framework algebraic formulas are treated as syntactic terms, and one must explicitly apply the polynomial-ring rewrite rules to derive one formula from another. Our proof system NFPC is simpler in the sense that we get a similar proof system to that in [GH03] (both our proof system and that in [GH03] can simulate Frege and both are polynomially verifiable), while adding no rewriting rules. The idea is that the only "hard to verify" rewrite rule is the commutativity axiom; and since we show how to efficiently simulate this rule we do not need the other polynomial-ring rewrite rules (like distributivity, associativity, etc.) to make the proof system polynomial verifiable: we can just use the deterministic polynomial identity testing algorithm for noncommutative formulas devised by Raz and Shpilka [RS05].

(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 lower bound on noncommutative formulas, by lower bounding the rank of certain matrices associated with noncommutative polynomials). We show certain sufficient conditions yielding super-polynomial lower bounds on OFPC proofs.

Related work. There is some resemblance between noncommutative formulas (and in fact, algebraic branching programs (ABPs)) 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 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, which 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

[1] This means that if a proof-line consists of the polynomial p, then one may choose to write *any* formula that computes p. (These kind of systems are sometimes called "semantic" proof systems.)

formulas in [Raz09] is close—though more involved—to that used by Nisan in the lower bound proof for noncommutative formulas [Nis91]. Therefore, proving lower bounds on OFPC proofs could be considered as a first step towards establishing lower bounds on multilinear proofs.

2. Preliminaries

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

2.1. **Noncommutative polynomials and formulas.** Let $\mathbb{F}$ be a field. Denote by $\mathbb{F}[x_1, \ldots, x_n]$ the ring of (commutative) polynomials with coefficients from $\mathbb{F}$ and variables $x_1, \ldots, x_n$. We denote by $\mathbb{F}\langle x_1, \ldots, x_n \rangle$ the *noncommutative* ring of polynomials with coefficients from $\mathbb{F}$ and variables $x_1, \ldots, x_n$. In other words, $\mathbb{F}\langle x_1, \ldots, x_n \rangle$ 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 $\mathbb{F}\langle x_1, \ldots, x_n \rangle$ (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 (commutative) arithmetic formula, except that we take care for the order in which products are done:

Definition 2.1 (Noncommutative formula)**.** *Let $\mathbb{F}$ be a field and $x_1, x_2, \ldots$ be variables. A noncommutative algebraic formula is an ordered*[2] *labeled tree, with edges directed from the leaves to the root, and with fan-in at most two. 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. *An algebraic formula* computes *a noncommutative polynomial in the ring of noncommutative polynomials $\mathbb{F}\langle x_1, \ldots, x_n \rangle$ in the following way. A leaf computes the input variable or field element that labels it. A plus gate computes the sum of polynomials computed by its incoming nodes. A product gate computes the* 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 an algebraic formula (and noncommutative formula) f is the total number of nodes in its underlying tree, and is denoted $|f|$.

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 2.1 (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.[3]

[2] This means that there is an order on the edges coming into a node.

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

2.2. **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 $p = 0$. If we want to consider the *size* of algebraic refutations we should fix the way polynomials inside refutations are written.

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

Definition 2.2. (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: $x_i \cdot (1 - x_i)$ *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 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 from p_j, p_k, for $j, k < i$, by one of the following inference rules:*

Product: *from p deduce $x_i \cdot p$, for some variable x_i ;*
Addition: *from p and q deduce $\alpha \cdot p + \beta \cdot q$, for some $\alpha, \beta \in \mathbb{F}$.*

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 is the total number of monomials (with nonzero coefficients) in all the proof-lines.*

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) $\Sigma\Pi$ formulas.[4] This complexity measure is equivalent—up to a factor of n—to the usual complexity measure counting the total number of monomials appearing in the proofs (Definition 2.2).

Definition 2.3. (Polynomial Calculus with Resolution (PCR)). *The PCR proof system is defined similarly to PC (Definition 2.2), 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 $\bar{x}_i$.*

2.3. **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 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).

[4]A $\Sigma\Pi$ formula F is an algebraic formula whose underlying tree is of depth 2, 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.

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 and is one-to-one]). 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. In most 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 $\mathcal{P}_1$.*

3. Polynomial calculus over noncommutative formulas

3.1. **Discussion.** 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 concrete lower bounds questions)—our formulation in Section 4 of algebraic proofs operating with noncommutative algebraic 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 sizes 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.

3.2. **The proof system NFPC.** We now define a proof system operating with noncommutative polynomials written as noncommutative algebraic 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 $\mathbb{F}\langle x_1, \ldots, x_n\rangle$ 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 $\mathbb{F}\langle x_1, \ldots, x_n\rangle$, 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).

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:

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

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:

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

Addition:

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

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 $p_i = 0$ 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 sizes of all the noncommutative formula sizes in π and is denoted by $|\pi|$.*

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 algebraic 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 2.1) 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 3.1. *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 2.1, 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. $\square$

For the next statements we use the algebraic propositional proof system $\mathcal{F}$-$\mathcal{PC}$ introduced by Grigoriev and Hirsch [GH03] as an algebraic counterpart of the Frege system. We refer the reader to [GH03] for definitions.

Proposition 3.2. *The systems* NFPC *is sound and complete. Specifically, let Q be a collection of noncommutative polynomials from $\mathbb{F}\langle x_1, \dots, x_n \rangle$. 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 3.3 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). $\square$

Theorem 3.3. NFPC *(over any field) polynomially-simulates Frege. Specifically,* NFPC *polynomially-simulates $\mathcal{F}$-$\mathcal{PC}$ in the following sense: let $f_1, \dots, f_m$ 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 $f_1, \dots, f_m$. Then, there exists an* NFPC *refutation of the same set of formulas $f_1, \dots, f_m$ (but now viewed as computing noncommutative polynomials) of size polynomial in s.*

Proof. By [GH03], $\mathcal{F}$-$\mathcal{PC}$ polynomially simulates Frege. The proof system $\mathcal{F}$-$\mathcal{PC}$ is an algebraic propositional proof system operating with (general, that is, commutative) algebraic formulas over a field, and it includes auxiliary rewriting rules allowing to develop equal polynomials syntactically via the polynomial-ring axioms. We proceed by showing a simulation of $\mathcal{F}$-$\mathcal{PC}$ by NFPC. 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. See [GH03] for the exact definition.

We proceed to simulate $\mathcal{F}$-$\mathcal{PC}$ 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 have derived a formula f using one of the rewriting rules of $\mathcal{F}$-$\mathcal{PC}$: associativity, distributivity unit and zero rules and commutativity. The rewriting rules of associativity, distributivity and unit and zero rules of $\mathcal{F}$-$\mathcal{PC}$ *do not change the noncommutative polynomial computed by an algebraic formula.* Therefore, we get them "for free" in NFPC: since we can choose to write a noncommutative polynomial

p in the proof as any noncommutative formula computing 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 3.4. *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 3.5. *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 3.4: 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 $x_i x_j - x_j x_i$.

Induction step: Either $|f| > 1$ or $|g| > 1$. Assume without loss of generality that $|f| > 1$. Following Claim 3.5, 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 3.4 □

To conclude the simulation of the commutativity rewrite rule of $\mathcal{F}$-$\mathcal{PC}$ (which will also conclude the proof of Theorem 3.3) we notice that, by Claim 3.5 and by Lemma 3.4, 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 from $\mathbb{F}\langle x_1, \ldots, x_n\rangle$, 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 noncommutative formula.

Let $X = \{x_1, \ldots, x_n\}$ be a set of variables and let $\mathbb{F}$ be a field. Let $\preceq$ be a linear order on the variables X. Let $f = \sum_{j \in J} b_j \mathcal{M}_j$ be a commutative polynomial from

$\mathbb{F}[x_1, \ldots, x_n]$, where the b_j's are coefficient from $\mathbb{F}$ and the $\mathcal{M}_j$'s are monomials in the X variables. We define $[\![f]\!] \in \mathbb{F}\langle x_1, \ldots, x_n\rangle$ to be the (unique) noncommutative polynomial $\sum_{j\in J} b_j \cdot [\![\mathcal{M}_j]\!]$, where $[\![\mathcal{M}_j]\!]$ is the (noncommutative) product of all the variables in $\mathcal{M}_j$ such that the order of multiplications respects $\preceq$. We denote the image of the map $[\![\cdot]\!] : \mathbb{F}[x_1, \ldots, x_n] \to \mathbb{F}\langle x_1, \ldots, x_n\rangle$ by $\mathcal{G}$.

Definition 4.1 (Ordered formula)**.** *The class of noncommutative formulas (Definition 2.1) computing polynomials from $\mathcal{G}$ is called the class of* ordered formulas *(under the given fixed linear order $\preceq$). We say that an order formula F computes the* commutative *polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ whenever F computes $[\![f]\!]$.*

Definition 4.1 enables us to define OFPC in a convenient way, that is, without referring to noncommutative polynomials: the system OFPC is defined similarly to PC, except that the proof-lines are written as ordered formulas:

Definition 4.2 (PC over ordered formulas (OFPC))**.** *Let $\pi = (p_1, \ldots, p_m)$ 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 polynomials $\mathbb{F}[x_1, \ldots, x_n]$), and let $\preceq$ be some linear order on the variables. The sequence $(p_1, \ldots, p_m)$ in which each p_i is written as an ordered formula (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 define 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 4.1. *For any linear order on the variables,* OFPC *is a sound, complete and polynomially-verifiable refutation system for establishing that a collection of 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, stems from the PIT algorithm for noncommutative formulas (Theorem 2.1).[5] □

Notes:

(1) We shall sometimes assume that there is an apriori fixed linear order of variables. Thus, we may speak about ordered formulas without referring explicitly to some linear order.

(2) 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]Formally, one should show precisely how to check that an ordered formula computes a polynomial that was deduced correctly by previous polynomials via the PC deduction rules. It is not hard to show how to perform such a check in deterministic polynomial time.

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 is strictly stronger than the polynomial calculus, polynomial calculus with resolution (PCR, for short; see Definition 2.3) and resolution (for a definition, 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 tautologies (formally, families of unsatisfiable collections of polynomial equations) that are hard (that is, do not have polynomial-size proofs) in PCR.

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.

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 $O(n)$. $\square$

5.0.1. OFPC *polynomially simulates* $\mathrm{R}^0(\mathrm{lin})$. We now show that OFPC can polynomially simulate the proof system $\mathrm{R}^0(\mathrm{lin})$ introduced in [RT08a]. This will be used in Section 5.0.2 to establish the OFPC upper bounds. In that paper a refutation system R(lin) was introduced. R(lin) is a refutation system extending resolution to work with disjunctions of linear equations instead of disjunction of literals. R^0(lin) is defined to be a subsystem of R(lin) in which certain restrictions put on the possible disjunctions of linear equations allowed in a proof. For the precise definition of R(lin) and R^0(lin) we refer the reader to [RT08a]. However, it is not entirely necessary to know the definitions of R(lin) and R^0(lin), since we will use a polynomial translation of R^0(lin) defined below, and describe explicitly what is needed for the proofs ahead.

First, we need the definitions that follow. A *polynomial translation of a clause* $\bigvee_{j\in J}(x_j^{b_j})$ is a 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 a CNF.

Definition 5.1 (Polynomial translation of $\mathrm{R}_{c,d}$(lin)-lines)**.** *A* polynomial translation of an $\mathrm{R}_{c,d}$(lin)-line *is a product $D = \prod_{j\in J} L_j$, where the L_j's are linear forms:*

(1) All variables in the linear forms have integer coefficients with absolute values at most c (the constant 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 constant terms*, or is a translation of a clause (as defined above).*

The width *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:

$$\prod_{j\in J}(x_j - b_j)\cdot\prod_{t=1}^{k}\prod_{i\in I_t}\left(\sum_{r=1}^{n} a_r^{(t)}x_r - \ell_i^{(t)}\right), \tag{3}$$

where $k \le d$ 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 $I_1,\dots,I_k,J$ are unbounded sets of indices). Clearly, a disjunction of clauses is a clause in itself, and so we can assume that in any $\mathrm{R}_{c,d}$(lin)-line only a single polynomial translation of a clause occurs.

We shall use the following propositions:

Proposition 5.2 (Algebraic translation of R^0(lin); Corollary 9.11 [RT08a] (restated))**.** *Let $K := \{K_n \mid n \in \mathbb{N}\}$ be a family of unsatisfiable CNF formulas*[6]*, and let $\{P_n \mid n \in \mathbb{N}\}$ be a family of* R^0(lin)-proofs *of K. Then, there are two constants c, d that do not depend on n and a family of PC proofs $\{P'_n \mid n \in \mathbb{N}\}$ of the polynomial translations of the family of CNFs K, such that for every n the proof P'_n has polynomial-size in the size of P_n number of steps, and where every line in P'_n is a (polynomial translation of an)* $\mathrm{R}_{c,d}$(lin)*-line (Definition 5.1) whose width is polynomial in the size of P_n.*

Remark: It is immaterial to define the size measure for R^0(lin) refutations (though this concept is mentioned in Theorem 5.2); we shall only use the fact that R^0(lin) has short refutations for some hard contradictions.

Note: Although corollary 9.11 in [RT08a] is stated for PCR instead of PC, the translation holds also for PC (see Remark before Corollary 9.11 in [RT08a]).

[6]Formally, we have a straightforward translation of CNFs to the language of R^0(lin) (see [RT08a]).

Definition 5.2 (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 $q = x_1^2 x_2 + a x_4^3$ (for some $a \in \mathbb{F}$) then $\mathbf{M}[q] = x_1 x_2 + a x_4$.

Proposition 5.3 (Implicit in [RT08b, RT08a])**.** *Let* P *be a PCR refutation from initial multilinear polynomials. Then we can transform* P *into a new PCR refutation* P' *from the same initial multilinear polynomials such that* P' *contains only multilinear polynomials, with only a polynomial increase in the number of steps. Moreover, if the proof lines in* P *are all* $\mathrm{R}_{c,d}(\mathrm{lin})$*-lines of maximal width* w*, then all the proof lines in* P' *are multilinearizations of* $\mathrm{R}_{c',d'}(\mathrm{lin})$*-lines of maximal width polynomial in* w *and where* c', d' *depend only on* c, d*.*

Proof sketch: Given a PCR proof $P = (p_1, \ldots, p_m)$ in the variables $\{x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n\}$, consider the sequence S of multilinearized polynomials $(\mathbf{M}[p_1], \ldots, \mathbf{M}[p_m])$. Then, by the proof of Theorem 5.1 in [RT08b] one can add polynomially in m many multilinear polynomials to S so that the new sequence S' consists of only multilinear polynomials and constitutes a PCR refutation of the initial polynomials. (Theorem 5.1 from [RT08b] talks about fMC refutations [Definition 2.6 in [RT08b]]. However, it is clear from the definition of fMC that the underlying sequence of polynomials in any fMC refutation constitutes a PCR refutation as well.)

Assume in addition that all polynomials in P are polynomial translations of $\mathrm{R}_{c,d}(\mathrm{lin})$-lines (Definition 5.1). Then, $S = (\mathbf{M}[p_1], \ldots, \mathbf{M}[p_m])$ is a sequence of multilinearizations of $\mathrm{R}_{c,d}(\mathrm{lin})$-lines. The only thing left to check is that the additional polynomials added to S to yield S' in the proof of Theorem 5.1 [RT08b] *are all polynomial translations of* $\mathrm{R}_{c',d'}(\mathrm{lin})$*-lines*, where c', d' depend only on c, d. This could be done by straightforward inspection of the proof of Theorem 5.1 [RT08b]. ■

Now we are ready to prove the main simulation of this subsection:

Theorem 5.4. *For any linear order on the variables,* OFPC *polynomially simulates* $\mathrm{R}^0(\mathrm{lin})$ *(over large enough fields). Moreover, we can assume that all formulas appearing in the* OFPC *proofs simulating* $\mathrm{R}^0(\mathrm{lin})$ *are depth-*3 *ordered formulas.*

By Propositions 5.2 and 5.3 and by the definition of OFPC, in order to prove that OFPC simulates $\mathrm{R}^0(\mathrm{lin})$, the following lemma from [RT08a] suffices:

Lemma 5.5 (Implicit in Lemma 9.14 [RT08a])**.** *Let* p *be a polynomial translation of an* $\mathrm{R}_{c,d}(\mathrm{lin})$*-line of width* w *over* n *variables. Then,* $\mathbf{M}[p]$ *can be computed by an ordered formula of size polynomial in* $w \cdot n$ *over fields of size bigger than* $w \cdot n$*. Moreover, the ordered formula is a* $\Sigma\Pi\Sigma$ *formula*[7]*.*

Proof. The proof uses the fact that $\mathrm{R}_{c,d}(\mathrm{lin})$-lines are close to a product of d symmetric polynomials, and the fact that symmetric polynomials can be computed by small ordered formulas (of depth-3) over large enough fields. Specifically:

Claim 5.6 (Restatement of Claim 9.15 in [RT08a])**.** *Let* D *be a polynomial translation of an* $\mathrm{R}_{c,d}(\mathrm{lin})$*-line of width* w*. Then,* D *is a linear combination (over* $\mathbb{F}$*) of* $(w + c)^{c \cdot d}$

[7] This means that every path from the root to the leaf in the formula tree starts with a plus gate, and the number of alternation in the path between plus and product gates is at most two

many terms, such that each term is of degree at most w and can be written as

$$q \cdot \prod_{k \in K} z_k^{r_k}\,, \tag{4}$$

where K is a collection of indices such that $|K| \le c \cdot d$, and r_k's are non-negative integers $\le w$, and the z_k's are homogenous linear forms such that each z_k has a single integral coefficient for all variables in it[8]*, and q is a polynomial translation of a clause.*

By this claim, to complete the proof of Lemma 5.5 it is sufficient to show that the multilinearization of any term as in (4):

$$\mathbf{M}\left[q \cdot \prod_{k \in K} z_k^{r_k} \right] \tag{5}$$

can be computed by an ordered $\Sigma\Pi\Sigma$ formula of size polynomial in cdn, over fields of size bigger than $c \cdot w$. This is done by using polynomial interpolation, as shown (implicitly) in Claim 9.16 in [RT08a]. More specifically, Claim 9.16 in [RT08a] demonstrated that (5) can be computed by a formula Φ such that: (i) Φ consists of polynomially in d, c many summands; (ii) each of these summands is a depth-3 $\Sigma\Pi\Sigma$ formula, in which every product gate is a product of linear forms; (iii) and each of these linear forms consists of only a single variable.

Note that any such formula Φ is also an ordered formula: since the products are of linear forms, each of a single variable, one can order the products in a way that respects the underlying variable order $\preceq$. □

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

$$\begin{array}{lll} \text{Pigeons :} & \forall i \in [m], & (1 - x_{i,1}) \cdots (1 - x_{i,n}) \\ \text{Functional :} & \forall i \in [m]\, \forall k < \ell \in [n], & x_{i,k} \cdot x_{i,\ell} \\ \text{Holes :} & \forall i < j \in [m]\, \forall k \in [n], & x_{i,k} \cdot x_{j,k} \end{array} \tag{6}$$

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

Corollary 5.7. *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* FPHP^m_n *(over large enough fields).*

$\neg\mathrm{FPHP}^m_n$ 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 [Raz98] 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 (6) 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 as defined for OFPC and PC).

[8] That is, $z_k = b \cdot \sum_{j=1}^{l} x_{i_j}$ for some natural number b.

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 5.8. *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 of the corresponding Tseitin* mod p *formulas* $\neg\textsc{Tseitin}_{G,p}$ *(over large enough fields).*

This stems from the R^0(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. Towards lower bounds on OFPC proofs

6.1. **Lower bounds on product formulas.** 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 6.2. We use a method of partial derivatives matrix introduced by Nisan to obtain exponential-size lower bounds on noncommutative formulas in [Nis91].

Proposition 6.1. *Let $\mathbb{F}$ be a field, $X := \{x_1, \ldots, x_n\}$ be a set of variables and $\preceq \subseteq (X \times X)$ be some linear order. Then, for any natural numbers $m \le n$ and $d \le \lfloor n/m \rfloor$, 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 ordered formula computing $\prod_{i=1}^{d} f_i$ has size $2^{\Omega(d)}$.*

Proof. Note first that it is sufficient to prove the proposition for $m = 2$ and any $d \le \lfloor n/2 \rfloor$: assume that the proposition holds for $m = 2$ and any $d \le \lfloor n/2 \rfloor$. Let $m' \le n$ 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 ordered formula 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 different notations 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:

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

Define

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

We show that every ordered formula of HARD_d (under $\preceq$) is of size at least $2^{\Omega(d)}$. Note that HARD_d is a homogenous and multilinear polynomial of degree d.

Recall that $[\![\mathsf{HARD}_d]\!]$ is the *noncommutative* polynomial obtained from HARD_d by ordering the products in every monomial in accordance to the fixed linear order $\preceq$. By definition of ordered formulas, we need to lower bound the size of noncommutative formulas computing $[\![\mathsf{HARD}_d]\!]$. For this purpose we use a rank argument introduced in [Nis91]. Nisan defined the matrix $M_k(f)$ associated with a noncommutative polynomial f as follows:

Definition 6.1 ([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 6.2 ([Nis91] Theorem 1). *Let f be a degree r homogenous noncommutative polynomial. Then, every noncommutative formula computing f has size at least $\sum_{k=0}^{r} \operatorname{rank}(M_k(f))$.*

In view of Theorem 6.2, it suffices to prove the following claim:

Claim 6.3. *For any $0 \le k \le d$ we have $\operatorname{rank}(M_k([\![\mathsf{HARD}_d]\!])) \ge \binom{d}{k}$.*

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 $\bar{y}$ 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 $\bar{z}$ 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 $\operatorname{rank}(\mathbf{A}_k) \le \operatorname{rank}(M_k([\![\mathsf{HARD}_d]\!]))$. $\blacksquare_{\text{Claim}}$

By the claim and by Theorem 6.2, we conclude that the ordered formula size of HARD_d is at least

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

$\square$

6.2. **A lower bound approach.** Here we discuss a simple possible approach intended to establish lower bounds on OFPC proofs, roughly, by reducing the lower bounds to PC degree lower bounds and using the bound in Section 6.1.

Let $Q_1(\bar{x}), \ldots, Q_m(\bar{x})$ 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 $f_1(\bar{y}), \ldots, f_n(\bar{y})$ be m homogenous polynomials of the same degree from $\mathbb{F}[y_1, \ldots, y_\ell]$, such that the ordered formula size of each $f_i(\bar{y})$ (for some linear order on the variables) is polynomial in n and such that the $f_i(\bar{y})$'s do not have common variables (that is, each $f_i(\bar{y})$ is over disjoint set of variables from $\bar{y}$). Assume that for any distinct $i_1, \ldots, i_d \in [n]$ the ordered formula size of $\prod_j^d f_{i_j}(\bar{y})$ is $2^{\Omega(d)}$.

Note: By the proof of Proposition 6.1, the conditions above are easy to achieve. Indeed, the $f_i(y_i, z_i)$'s defined in the proof of Proposition 6.1 have these properties: homogeneity, same degrees for all f_i's and disjointness of variables, and an exponential increase in ordered formula size for any product of the f_i's.

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

$$x_i \mapsto f_i(\bar{y})\,. \tag{7}$$

In other words, consider

$$Q_1(f_1(\bar{y}), \ldots, f_n(\bar{y})), \ldots, Q_m(f_1(\bar{y}), \ldots, f_n(\bar{y}))\,. \tag{8}$$

Note that (8) is also unsatisfiable over $\mathbb{F}$. We suggest to lower bound the OFPC refutations size of (8), based on the following simple idea: it is known that some families of unsatisfiable collections of polynomials require linear $\Omega(n)$ *degree* PC refutations. 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 degree.

Thus, assume that the initial polynomials $Q = \{Q_1(\bar{x}), \ldots, Q_m(\bar{x})\}$ in the $x_1, \ldots, x_n$ variables, require linear degree refutations—in fact, an $\omega(\log n)$ degree lower bound would suffice. Thus, every PC refutation contains some polynomial h of degree $\omega(\log n)$. Then, we might expect that every PC refutation of (8) contains a polynomial $g \in \mathbb{F}[\bar{y}]$ which is a substitution instance (under the substitution (7)) of an $\omega(\log n)$-degree polynomial in the $\bar{x}$ variables. This, in turn, leads (under some conditions; see below for an example of such conditions) to a lower bound on OFPC refutations. Specifically, an example of sufficient conditions for super-polynomial OFPC lower bounds, is as follows: every PC refutation of (8) contains a polynomial g so that one of g's homogenous components is a substitution instance (under the substitution (7)) 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 notations and conditions.) **If:** every PC refutation of (8) 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 $t \le \deg(g)$, the t-th homogenous component $g^{(t)}$ of g (that is, the sum of all monomials of total degree t in g) is a *substitution instance* (under the substitution (7)) of a degree $\omega(\log n)$ multilinear polynomial from $\mathbb{F}[x_1, \ldots, x_n]$;
Then: every OFPC refutation of (8) 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. Note that breaking an algebraic formula into its corresponding homogenous components—according to the standard procedure (cf. [Raz08], Section 2.1)—is also applicable to ordered formulas: in other words, if g has a polynomial-size ordered formula then each of g's homogenous components has a polynomial-size ordered formula as well.[9] Thus, it remains to show that every ordered formula of $g^{(t)}$ is of size super-polynomial 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 the $f_i(\bar{y})$'s are homogenous, h must be homogenous too. Also, since $g^{(t)}$ is multilinear h is also multilinear. Thus, we can write $h = \sum_{j \in J} b_j \mathcal{M}_j$, where the $\mathcal{M}_j$'s are multilinear monomials in the $\bar{x}$ variables and b_j are coefficients from $\mathbb{F}$. Now, consider some single monomial $\mathcal{M}$ from $\sum_{j \in J} b_j \mathcal{M}_j$. By multilinearity and homogeneity of h every other monomial $\mathcal{M}' \neq \mathcal{M}$ in h must contain an x_i variable that does not appear in $\mathcal{M}$. We can assign 0 to such x_i. Doing this for every monomial $\mathcal{M}' \neq \mathcal{M}$, we get that h (under this partial assignment to the $\bar{x}$ variables) is equal to $b\mathcal{M}$, for some coefficient $b \in \mathbb{F}$. In a similar manner, by disjointness of the variables in the $f_i(\bar{y})$'s, there exists a partial assignment $\rho : \bar{y} \to \{0\}$, such that $g^{(t)} \restriction \rho$ is just a substitution instance (under the substitution (7)) of a single degree $\omega(\log n)$ multilinear monomial in the $\bar{x}$ variables. This means that $g^{(t)} \restriction \rho$ is the product of $\omega(\log n)$ distinct $f_i(\bar{y})$'s (multiplied by b). Therefore, by assumption, every ordered formula of $g^{(t)}$ is of size exponential in $2^{\omega(\log n)}$, which is super-polynomial in n. ∎

Acknowledgments

I wish to thank Emil jeřabek, Sebastian Müller, Pavel Pudlák and Neil Thapen for helpful discussions on issues related to 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)). 5.0.2

[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.*,

[9]Assume we have an ordered formula Φ and we want to construct the ordered formula $\Phi^{(k)}$ that computed the homogenous formula of degree k. We work by induction on formula Φ structure: a *plus gate* in the original formula Φ turns into a plus gate u with two children, such that if each of the two subformulas rooted at the two children is an ordered formula then the subformula rooted at u is also an ordered formula. A *product gate* turns into a plus gate v that is the root of a sum of products of pairs of ordered subformulas, such that if the original product gate respects the linear order then also each of the products in the sum respects the linear order. (See for example [Raz08] for more details on the construction of [non-ordered] homogenous formulas from a given algebraic formula.)

62(2):267–289, 2001. Special issue on the 14th Annual IEEE Conference on Computational Complexity (Atlanta, GA, 1999). 5.0.2

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

[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. 5.0.2

[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, 2.2

[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. 2.3

[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, 3.2, 3.2

[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. 5.0.2

[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. 1, 1.1, 6.1, 6.1, 6.1, 6.2

[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

[Raz98] Alexander A. Razborov. Lower bounds for the polynomial calculus. *Comput. Complexity*, 7(4):291–324, 1998. 5.0.2

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

[Raz08] Ran Raz. Elusive functions and lower bounds for arithmetic circuits. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 711–720, 2008. 6.2, 9

[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. *Computational Complexity*, 14(1):1–19, 2005. 1.1, 2.1, 2.1

[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, 5.0.1, 5.2, 5.0.1, 6, 5.3, 5.0.1, 5.5, 5.6, 5.0.1, 5.0.2, 5.0.2, 5.0.2

[RT08b] Ran Raz and Iddo Tzameret. The strength of multilinear proofs. *Computational Complexity*, 17(3):407–457, 2008. 1, 1.1, 5.3, 5.0.1

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

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

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

Mathematical Institute, Academy of Sciences of the Czech Republic, Žitná 25, 115 67 Praha 1, Czech Republic.

E-mail address: `tzameret@math.cas.cz`

ECCC ISSN 1433-8092

http://eccc.hpi-web.de