Space Complexity in Polynomial Calculus (Extended Abstract)

Yuval Filmus*, Massimo Lauria†, Jakob Nordström‡, Neil Thapen§ and Noga Ron-Zewi¶
*University of Toronto, yuval.filmus@utoronto.ca
†Sapienza — Università di Roma, lauria.massimo@gmail.com
‡KTH Royal Institute of Technology, jakobn@kth.se
§Academy of Sciences of the Czech Republic, thapen@math.cas.cz
¶Technion — Israel Institute of Technology, nogaz@cs.technion.ac.il

***Abstract*—During the last decade, an active line of research in proof complexity has been to study space complexity and time-space trade-offs for proofs. Besides being a natural complexity measure of intrinsic interest, space is also an important issue in SAT solving. For the polynomial calculus proof system, the only previously known space lower bound is for CNF formulas of unbounded width in [Alekhnovich et al. '02], where the lower bound is smaller than the initial width of the clauses in the formulas. Thus, in particular, it has been consistent with current knowledge that polynomial calculus could refute any k-CNF formula in constant space.**

We prove several new results on space in polynomial calculus (PC) and in the extended proof system polynomial calculus resolution (PCR) studied in [Alekhnovich et al. '02].

1) **For PCR, we prove an $\Omega(n)$ space lower bound for a bitwise encoding of the functional pigeonhole principle with m pigeons and n holes. These formulas have width $\mathrm{O}(\log n)$, and hence this is an exponential improvement over [Alekhnovich et al. '02] measured in the width of the formulas.**
2) **We then present another encoding of the pigeonhole principle that has constant width, and prove an $\Omega(n)$ space lower bound in PCR for these formulas as well.**
3) **We prove an $\Omega(n)$ space lower bound in PC for the canonical 3-CNF version of the pigeonhole principle formulas PHP^m_n with m pigeons and n holes, and show that this is tight.**
4) **We prove that any k-CNF formula can be refuted in PC in simultaneous exponential size and linear space (which holds for resolution and thus for PCR, but was not known to be the case for PC). We also characterize a natural class of CNF formulas for which the space complexity in resolution and PCR does not change when the formula is transformed into 3-CNF in the canonical way.**

***Keywords*-proof complexity; polynomial calculus; PCR; resolution; space; k-CNF**

I. Introduction

A proof system for a language L is a binary predicate $P(x, \pi)$ computable in time polynomial in the sizes of the inputs such that for all $x \in L$ there is a string π (a *proof*) for which $P(x, \pi) = 1$, while for any $x \notin L$ it holds for all strings π that $P(x, \pi) = 0$. A *propositional proof system* is a proof system for TAUTOLOGY, the language of tautologies in propositional logic.

The field of proof complexity, initiated by Cook and Reckhow [21], studies the complexity of proving propositional formulas in different propositional proof systems. One important motivation for proof complexity is the problem of P vs. NP. A proof system is said to be polynomially bounded if for every $x \in L$ there is a proof π_x of size at most polynomial in the size of x. As observed in [21], one way of establishing $\mathsf{co\text{-}NP} \neq \mathsf{NP}$, and hence $\mathsf{P} \neq \mathsf{NP}$, would be to prove that there are no polynomially bounded proof systems. This goal remains very distant, however, and it is probably fair to say that most current work in proof complexity is motivated by other concerns.

One such other concern, that has motivated an interesting line of research in proof complexity in the last decade, is the SATISFIABILITY problem and the study of proof complexity measures related to SAT solving. While it is generally believed that SATISFIABILITY is an intractable problem in the worst case, impressive algorithmic developments in the last 10–15 years have led to SAT solvers that can handle real-world problem instances with millions of variables. A somewhat surprising aspect of this is that at the core, the state-of-the-art solvers today are still based on the fairly simple *Davis-Putnam-Logemann-Loveland* or *DPLL* procedure [23], [24] from the early 1960s augmented with clause learning [4], [30]; these programs are also known as *conflict-driven clause learning* solvers or *CDCL* solvers. Despite the fact that such SAT solvers can be seen to be searching for proofs in the relatively weak *resolution proof system*, for which numerous exponential lower bounds are known, CDCL solvers have carried the day in the international SAT competition (www.satcompetition.org) in recent years.

From the point of view of proof complexity, by studying proof systems that are, or could potentially be, used by SAT solvers, one can hope to gain a better understanding of the potential and limitations of such solvers. There is a growing literature of such papers, with [3], [9], [18] being a very subjective pick of just three of the most recent interesting examples. While the current paper is of a more purely theoretical nature, it is also partly motivated by similar concerns.

The two main bottlenecks for modern SAT solvers are running time and memory usage. By studying proof size, proof space, and trade-offs between these two measures in different proof systems, we want to understand how the important resources of time and space are connected to one another and whether they can be optimized simultaneously or have to be traded for one another in SAT solvers using these proof systems.[1] In this context, it seems that the most interesting proof systems are resolution, polynomial calculus and cutting planes. For more on proof complexity and space see, for instance, [5], [32], [36]. A recent, very comprehensive, reference on SAT solving is [13].

A. Previous Work

Any formula in propositional logic can be efficiently converted to a CNF formula that is only linearly larger and is unsatisfiable if and only if the original formula is a tautology. Therefore, any sound and complete system for refuting CNF formulas can be considered as a general propositional proof system. Furthermore, while the general definition of a proof system allows for any predicate P, in practice the proof systems studied in the proof complexity literature tend to have the structure that a proof π can be viewed as a sequence of lines, where each line either is (some encoding of) a disjunctive clause of the CNF formula being refuted or follows from previous lines in the proof by the inference rules of the proof system in question. We will say that such proof systems are *sequential*. All proof systems considered in this paper are sequential proof systems for refuting unsatisfiable CNF formulas.

Of the three proof systems mentioned above, the *resolution proof system* is by far the most well-studied and well-understood. Resolution was introduced in [14] and began to be investigated in connection with automated theorem proving in the 1960s [23], [24], [35]. In this context, it is natural to prove bounds on the *length* of refutations, i.e., the number of clauses, rather than on the total size (the two measures are easily seen to be polynomially related). One of the early break-throughs in proof complexity was the result by Haken [26] that CNF formulas encoding the pigeonhole principle (PHP formulas) require proofs of exponential length. There have been a sequence of follow-up papers establishing quantitatively stronger bounds for other formula families in, for instance, [12], [19], [37].

Motivated by the fact that memory usage is a major concern in applied SAT solving, and by the question of whether proof complexity could say anything interesting about this, the study of space in resolution was initiated by Esteban and Torán in [25]. Alekhnovich et al. [1] later extended the concept of space to a more general setting, including other proof systems, and this setting is what will be of interest to us in this paper. Intuitively, the space of a resolution refutation is the amount of memory needed while verifying the refutation (where in resolution usually one thinks of each clause as requiring one unit of memory, a measure that is known as *clause space*). Perhaps somewhat surprisingly, it turns out that one need never use more than linear (clause) space in resolution, and a sequence of papers [1], [8], [25] have established matching lower bounds (up to constant factors).

Another sequence of papers culminating in [10] involving the third author clarified the relation between length and space in resolution. It was established in [10] that there exist explicit formulas that are maximally easy with respect to length, having linear length refutations, but which are hard for space in that their clause space complexity is $\Omega(n/\log n)$ (this separation is optimal). Regarding trade-offs betweem length and space, some results in restricted settings were presented in [7], [31], and strong trade-offs for general resolution were finally obtained in [11]. Very recently, [6] announced trade-off results for formulas that require even superlinear space if length is to be optimized.

In the *polynomial calculus (PC)* proof system introduced by Clegg et al. [20], clauses are interpreted as multilinear polynomial equations over some field, and a CNF formula is refuted by showing that there is no common root for the polynomial equations corresponding to all the clauses. The minimal refutation size of a formula in this proof system turns out to be closely related to the total degree of the polynomials appearing in the refutation [28], and a number of strong lower bounds on proof size have been obtained by proving degree lower bounds in, for instance, [2], [28], [34].

The treatment of negated and unnegated literals in PC is asymmetric and means that wide clauses with literals of the wrong sign can blow up to polynomial equations of exponential size. To get a more symmetric treatment of space, [1] introduced *polynomial calculus resolution (PCR)* as a common extension of PC and resolution.[2] Briefly, in PCR one adds an extra set of parallel formal variables $\overline{x}, \overline{y}, \overline{z}, \ldots$ as well as axioms specifing that x and $\overline{x}$ must always take opposite signs (so that we can think of the variable $\overline{x}$ as the literal negating x). In this way, negated and unnegated literals can both be represented in a space-efficient fashion.

In this stronger PCR system, [1] managed to establish lower bounds on space measured as the number of monomials, but only sublinear bounds and for formulas of unbounded width (namely, for PHP formulas). The problem of proving linear lower bounds on space in PC and PCR, and more importantly of proving any nontrivial lower bounds for formulas of bounded width in terms of PCR-space or even PC-space, has remained open since [1]. Thus, it has been

[1]For instance, recent experimental work by the third author joint with Järvisalo, Matsliah and Živný [29] seems to indicate a correlation between theoretical space complexity in resolution and practical running time for CDCL solvers.

[2]It should be noted, though, that if our main concern in studying space is the connection to SAT solving, then it is not entirely clear that the generalization to PCR is the right way to go. The issue is that PCR might in fact be a bit too strong in the sense that it magically eliminates a problem with exponential space blow-up that actually appears to be an issue in practice for some PC-based SAT solvers.

theoretically consistent with our current state of knowledge that all k-CNF formulas would potentially be refutable in constant space in polynomial calculus. And as far as we are aware, there have not been any trade-off results shown before (the partial results in) the very recent paper [27].

At the time the paper [20] was published, there was quite some excitement about polynomial calculus, since this proof system seemed to hold out the promise of better SAT solvers than those based on resolution. This promise has failed to materialize so far, however. There are PC-based solvers such as PolyBoRi [16], but in general they seem to be an order of magnitude slower than state-of-the-art CDCL solvers (although [17] reports that PolyBoRi can be faster on certain specific industrial instances).

In the *cutting planes (CP)* proof system [22], the clauses of a CNF formula are translated to linear inequalities and the formula is refuted by showing that the polytope defined by these does not have any zero-one integer points (corresponding to satisfying assignments). We only know of one superpolynomial lower bound on CP proof size [33] (improving on the result [15] in a somewhat restricted setting). As far as we are aware, nothing is known on space for cutting planes, much less for size-space trade-offs, except again for the recent paper [27].

B. Our Results

In this paper, we focus on polynomial calculus and PCR and prove several new results. We give an overview of these results below. The notation and terminology used follows that of the survey [32] fairly closely, but for completeness we provide all the necessary preliminaries in Section II.

1) Upper Bound on Space for k-CNF Formulas in Polynomial Calculus: A first natural question when proving lower bounds on space in polynomial calculus is how strong bounds we can hope for, i.e., what upper bounds there are to match. For the resolution proof system, it is easy to show that any CNF formula F with m clauses over n variables has a refutation in simultaneous length $\exp(\min\{m,n\}+\mathrm{O}(1))$ and clause space $\min\{m,n\}+\mathrm{O}(1)$. Since PCR can simulate resolution efficiently line by line, we get similar upper bounds for size and monomial space there.

For polynomial calculus without extra variables for negative literals, however, it is easy to see that one cannot polynomially simulate resolution. Namely, consider a formula F with a wide clause consisting of only negative literals. Just representing such a clause in the PC-translation to a polynomial requires exponential size and space. This counter-example for PC seems somewhat artificial, however, since we could transform F to an equivalent 3-CNF formula in the canonical way and work with this formula instead to avoid the problems with downloading wide all-negative clauses. Therefore, we are interested in determining upper bounds for the worst case for PC when the unsatisfiable input formulas are given in k-CNF.

Interestingly, this turns out to be connected to a problem regarding width in resolution. We know from [12] that if a formula cannot have resolution refutations without at least one clause of linear length, then the length of any resolution refutation has to be exponential. In fact, by counting one immediately gets that not only must any refutation contain at least *one* wide clause in such a case, but rather an exponential number of wide clauses. Suppose now that we are considering random k-CNF formulas, where the signs of the literals in the axioms will be randomly (and evenly) divided. Is it true that in any resolution refutation of such a formula, there must also be wide clauses with reasonably evenly divided positive and negative literals? Or weakening this question a bit: Is it true that in any resolution refutation of a random k-CNF formula, it holds with high probability that the refutation must contain at least one clause with a large positive component and one clause (the same one or another one) with a large negative component?

Somewhat surprisingly, the answer to this question is a resounding "no." If we want to minimize negative width (or positive width, for that matter), then for any unsatisfiable k-CNF formula F we can find a resolution refutation that never has any clause with more than k negative (positive) literals.

This is an interesting fact in itself, but it also has immediate consequences for polynomial calculus. Namely, the reason that PC cannot simulate resolution in the same way as PCR is that clauses with many negative literals cause an exponential blow-up in monomial space. But since we can construct a resolution refutation that never has more than k negative literals in any clause, we can limit this blow-up to an exponential in k, which is a constant. Hence, we get that PC has at least as good worst-case behaviour for k-CNF formulas as does resolution.

Theorem 1. *Any unsatisfiable k-CNF formula F with m clauses over n variables has a PC-refutation in simultaneous length $\exp(\mathrm{O}(\min\{m,n\}))$ and monomial space $\mathrm{O}(\min\{m,n\})$, where the hidden constant depends on k.*

2) Lower Bound on Space for k-CNF Formulas in Polynomial Calculus: Next, we turn our attention to lower bounds for k-CNF formulas in PC. There is a standard way to turn any CNF formula F into an equivalent 3-CNF formula $\widetilde{F}$ by converting every wide clause $C = a_1 \vee a_2 \vee \ldots \vee a_w$ to the set of 3-clauses $\widetilde{C} = \{y_0\} \cup \{\overline{y}_{j-1} \vee a_j \vee y_j \mid 1 \leq j \leq w\} \cup \{\overline{y}_w\}$ where the y_i are new variables that do not appear anywhere else. Let $\widetilde{PHP}^m_n$ denote the pigeonhole principle formula PHP^m_n converted to 3-CNF in this way. For resolution we know that the space requirements for these two versions are the same, but the $\Omega(n)$ lower bound on space in PCR for PHP^m_n in [1] unfortunately breaks down for $\widetilde{PHP}^m_n$, and no lower bound has been known even for PC.

Intuitively, one would like to think of the semantics of the new auxiliary variables as being $y_i \equiv \bigvee_{j=i+1}^{w} a_j$, and use this to extract a PCR-refutation of PHP_n^m from any PCR-refutation of $\widetilde{PHP}_n^m$ by substituting $\bigvee_{j=i+1}^{w} a_j$ for y_i. Sadly, this idea does not work. The problem is that the semantics of the negation of y_i in the 3-CNF conversion is $\overline{y}_i \equiv \bigvee_{j=1}^{i} a_j$, and under this interpretation there is nothing ruling out that both y_i and $\overline{y}_i$ "are true simultaneously," as it were. Therefore, this simulation idea fails.

However, for PC we never need to deal with $\overline{y}_i$, since this literal does not exist, and if we do the substitution for y_i above then it turns out we can in fact extract a PC-refutation of PHP_n^m from any PC-refutation of $\widetilde{PHP}_n^m$. This immediately yields a first nontrivial lower bound on space for 3-CNF formulas in PC. Using the ideas from Section I-B1 together with results from the literature, it is not hard to show that this bound is asymptotically tight.

Theorem 2. *The space of refuting $\widetilde{PHP}_n^{n+1}$ in polynomial calculus is $\Theta(n)$, or $\Theta\big(\sqrt[3]{N}\big)$ expressed in the formula size N.*

This lower bound holds for the standard (from the algebraic point of view) encoding equating 0 with true and 1 with false. Since PC is clearly very sensitive to such issues of representation, it is natural to ask whether the lower bound is due to an unfavourable encoding and could be avoided by a preprocessing step flipping the polarities of the literals in the formula in some way. However, it is straightforward to show, appealing to [1], that Theorem 2 holds even if we allow arbitrary flips of literal polarities in the formula.

3) Lower Bound on Space for k-CNF Formulas in PCR: The main goal of this paper, however, is to prove lower bounds on space for k-CNF formulas not in polynomial calculus, but in the stronger PCR system. And here the simple simulation used to obtain Theorem 2 no longer works, for the reasons sketched above.

As a first step, we instead consider an alternative encoding of the pigeonhole principle. In the PHP_n^m formula, we have variables $p_{i,j}$ encoding that pigeon i sits in hole j. However, there is another way to encode the pigeonhole principle that arises naturally in bounded arithmetic, which uses variables $x[i,\ell]$ for $\ell = 1,\ldots,\log_2 n$ encoding in binary the hole into which pigeon i goes. Note that in this encoding the pigeonhole principle will automatically be functional, i.e., every pigeon gets sent to exactly one hole and not more. These "bit-graph PHP formulas" have width logarithmic in n and, as we prove, require space linear in n in PCR. Hence, this provides an exponential improvement over [1] measured in the width of the formulas.

Theorem 3. *The space in PCR of refuting bit-graph pigeonhole principle formulas $BPHP_n^{n+1}$ is $\Omega(n)$, or $\Omega\big(\sqrt[3]{N/\log N}\big)$ expressed in the formula size N.*

We then tweak the formulas in a specific way to have "hole indicators" for each hole and pigeon, where we say that pigeon i sits in hole j if the exclusive or of the hole indicator variables for $(i, j-1)$ and (i,j) is true. While we certainly do not claim that this is the most natural encoding of the pigeonhole principle ever presented in the literature, it has the nice feature that it can be written as a 4-CNF and that the proof of Theorem 3 still works with very minor modifications. So in this way we are finally able to prove strong lower bounds on PCR space for k-CNF formulas.

Theorem 4. *The space in PCR of refuting the XOR pigeonhole principle $XPHP_n^{n+1}$ encoded in 4-CNF is $\Omega(n)$, or $\Omega\big(\sqrt[3]{N}\big)$ expressed in the formula size N.*

The proofs of Theorems 3 and 4 are very much inspired by [1] and follow the arguments in that paper fairly closely, but also require some subtle but crucial twists. We refer to Section III for the proof of Theorem 3. The easy modifications to prove Theorem 4 are described in the upcoming full version of the paper.

4) Space Complexity of Wide CNF Formulas and Their 3-CNF Versions: While Theorem 4 establishes nontrivial PCR space lower bounds for specially crafted 4-CNF formulas, we still do not know any lower bounds for 3-CNF formulas. In particular, the PCR space complexity of the "3-CNF version" $\widetilde{PHP}_n^m$ of the pigeonhole principle formulas remains open. Returning to the discussion in Section I-B2, it is natural to ask the question in general what happens to the space complexity of formula F when it is transformed to a 3-CNF $\widetilde{F}$ in the canonical way described above. It is clear that such a transformation can never increase the space complexity. For all formula families that we are aware of, the space complexity, when it is known, does not decrease either, but stays the same. It would be very interesting to know whether this is true in general, or whether it is somehow possible to come up with a family of wide formulas F_n where the space complexity of $\widetilde{F}_n$ is asymptotically smaller.

As a first step towards resolving this question, we characterize a natural class of CNF formulas for which the space complexity in resolution and PCR provably does not decrease when the formula is transformed into a 3-CNF. Suppose that for every wide clause $a_1 \vee \ldots \vee a_w$ in F there are also axioms $\overline{a}_i \vee \overline{a}_j$ for all $1 \leq i < j \leq w$ requiring that any satisfying assignment of the clause is constrained to have Hamming weight 1 (we call such a formula *weight-constrained*). Then for such a formula F the idea in Section I-B2 of substituting $\bigvee_{j=i+1}^{w} a_j$ for y_i and $\bigvee_{j=1}^{i} a_j$ for $\overline{y}_i$ turns out to actually work.

Theorem 5. *Let F be a weight-constrained CNF formula and let $\widetilde{F}$ be its 3-CNF version as described above. Then in resolution F and $\widetilde{F}$ have the same space complexity up to a small additive constant, and in PCR the two formulas have asymptotically the same space complexity (within small multiplicative factors).*

In particular, this means that for the standard encoding of the *functional* pigeonhole principle, which has precisely such weight constraints, the space complexity of the wide formula and its 3-CNF version is essentially the same. Unfortunately, nothing is known about the PCR space complexity of this formula, and in particular the techniques in [1] break down when functional axioms are added to the formula. However, what Theorem 5 says is that if one can manage to prove PCR space lower bounds for the wide functional pigeonhole principle, then the same bound also holds for the 3-CNF version. We hope that this insight can be useful when approaching the task of proving PCR space lower bounds for (3-CNF versions of) the functional pigeonhole principle and other well-studied formula families in proof complexity. Also, it would be interesting to see if Theorem 5 could be generalized to hold for any CNF formula, even without weight constraints, or if there is some counter-example.

C. Outline of This Paper

The rest of this paper is organized as follows. The necessary preliminaries are presented in Section II. We then give a detailed presentation of what we consider to be the most exciting result, namely our space lower bound in polynomial calculus resolution for a particular encoding of the pigeonhole principle with small-width clauses presented in Section III. Though these formulas do not quite have constant width, we feel this result is for a more natural family of formulas, and therefore perhaps a bit easier to follow, than the analogous result for formulas of constant width. This latter result, as well as all other results mentioned in Section I-B, can be found in the upcoming full-length version of the paper. In Section IV, we make some concluding remarks and mention a few of the many fascinating problems in this area that remain open.

II. Preliminaries

Let x be a Boolean variable. A *literal over* x is either the variable itself or its negation, denoted $\neg x$ or $\overline{x}$. It will also be convenient to use the alternative notation x^b for $b \in \{0, 1\}$, where x^b is x when $b = 0$ and $\overline{x}$ when $b = 1$.

A *clause* $C = a_1 \vee \cdots \vee a_k$ is a disjunction of literals. We denote the empty clause, i.e., the clause containing no literals, by $\perp$. A clause containing at most k literals is called a k*-clause*. A *CNF formula* $F = C_1 \wedge \cdots \wedge C_m$ is a conjunction of clauses. A k*-CNF formula* is a CNF formula consisting of k-clauses.

Assignments are functions that assign a truth value for each variable in v. We write α, β to denote truth value assignments. An assignment *satisfies* a Boolean function if it makes the function true. In the context of the algebraic proof systems PC and PCR (defined below) we will identify 0 with true and 1 with false (so that x^b is true if $x = b$).

We say that a proof system for refuting unsatisfiable CNF formulas is *sequential* if a proof π in the system is a *sequence* of lines, where each line is derived from previous lines by one of a finite set of allowed *inference rules*. The following definition is a straightforward generalization of [1].

Definition 6 (Refutation). For a sequential proof system $\mathcal{P}$ with lines of the form L_i, a $\mathcal{P}$-configuration $\mathbb{D}$, or, simply, a *configuration*, is a set of such lines. A sequence of configurations $\{\mathbb{D}_0, \ldots, \mathbb{D}_\tau\}$ is said to be a $\mathcal{P}$*-derivation* from a CNF formula F if $\mathbb{D}_0 = \emptyset$ and for all $t \in [\tau]$, the set $\mathbb{D}_t$ is obtained from $\mathbb{D}_{t-1}$ by one of the following *derivation steps*:

- **Axiom Download:** $\mathbb{D}_t = \mathbb{D}_{t-1} \cup \{L_C\}$, where L_C is the encoding of a clause $C \in F$ in the syntactic form prescribed by the proof system (an *axiom clause*).
- **Inference:** $\mathbb{D}_t = \mathbb{D}_{t-1} \cup \{L\}$ for some L inferred by one of the inference rules for $\mathcal{P}$ from a set of assumptions $L_1, \ldots, L_m \in \mathbb{D}_{t-1}$.
- **Erasure:** $\mathbb{D}_t = \mathbb{D}_{t-1} \setminus \{L\}$ for some $L \in \mathbb{D}_{t-1}$.

A $\mathcal{P}$-refutation $\pi \colon F \vdash \perp$ of a CNF formula F is a $\mathcal{P}$-derivation $\pi = \{\mathbb{D}_0, \ldots, \mathbb{D}_\tau\}$ such that $\mathbb{D}_0 = \emptyset$ and $\perp \in \mathbb{D}_\tau$, where $\perp$ is the representation of contradiction (e.g. for resolution the empty clause without literals).

Definition 7 (Refutation size, length and space). Given a size measure $S(L)$ for lines L in $\mathcal{P}$-derivations , the *size* of a $\mathcal{P}$-derivation π is the sum of the sizes of all lines in a derivation, where lines that are derived multiple times are counted with repetitions. The *length* of a $\mathcal{P}$-derivation π is the number of axiom downloads and inference steps in it. For a space measure $Sp_{\mathcal{P}}(\mathbb{D})$ defined for $\mathcal{P}$-configurations, the *space* of a derivation π is defined as the maximal space of a configuration in π.

We define the $\mathcal{P}$*-refutation size* of a formula F, denoted $S_{\mathcal{P}}(F \vdash \perp)$, to be the minimum size of any $\mathcal{P}$-refutation of it. The $\mathcal{P}$*-refutation length* $L_{\mathcal{P}}(F \vdash \perp)$ and $\mathcal{P}$*-refutation space* $Sp_{\mathcal{P}}(F \vdash \perp)$ of F are analogously defined. When the proof system in question is clear from context, we will drop the subindex in the proof complexity measures.

Let us next give formal definitions in the framework of Definition 6 of the proof systems that will be of interest in this paper.

Definition 8 (Resolution). In resolution, derivation lines are clauses and inferences follow the *resolution rule*:

$$\frac{B \vee x \qquad C \vee \overline{x}}{B \vee C} \tag{1}$$

for clauses B and C. We refer to $B \vee C$ as the *resolvent* of $B \vee x$ and $C \vee \overline{x}$ on x. Sometimes it will be useful to allow an additional rule, *weakening*:

$$\frac{B}{B \vee C} \tag{2}$$

for clauses B and C. Weakening is *admissible*, in the sense that weakening can be eliminated from every resolution refutation without increasing any of the standard parameters such as length, size, or space.

For resolution, we will consider three separate space measures: *clause space*, *total space* and *width*.

Definition 9 (Width and space in resolution). The *width* $W(C)$ of a clause C is the number of literals in it, and the width of a CNF formula or clause configuration is the size of the widest clause in it. The *clause space* $Sp(\mathbb{C})$ of a clause configuration $\mathbb{C}$ is the number of clauses in $\mathbb{C}$, and the *total space* $TotSp(\mathbb{C})$ is the total number of literals in $\mathbb{C}$ counted with repetitions. The width or space of a resolution refutation π is the maximum that the corresponding measures attains over any configuration $\mathbb{C} \in \pi$.

The *polynomial calculus (PC)* proof system was introduced in [20] under the name of "Gröbner proof system." In a PC-refutation, clauses are interpreted as multilinear polynomials. For instance, the requirement that the clause $x \vee y \vee \overline{z}$ should be satisfied gets translated to the equation $xy(1-z) = 0$ or $xy - xyz = 0$ (recall that we think of 0 as true and 1 as false), and we derive contradiction by showing that there is no common root for the polynomial equations corresponding to all the clauses.

Definition 10 (Polynomial Calculus (PC)). Lines in a polynomial calculus proof are multivariate polynomial equations $p = 0$, where $p \in \mathbb{F}[x, y, z, \ldots]$ for some (fixed) field $\mathbb{F}$. It is customary to omit "$= 0$" and only write p. The derivation rules are as follows, where $\alpha, \beta \in \mathbb{F}$, $p, q \in \mathbb{F}[x, y, z, \ldots]$, and x is any variable:

- **Boolean axioms:** $\dfrac{}{x^2 - x}$ (forcing 0/1-solutions).
- **Linear combination:** $\dfrac{p \qquad q}{\alpha p + \beta q}$
- **Multiplication:** $\dfrac{p}{xp}$

For an assignment α to variables and a PC configuration $\mathbb{P}$, we say that α *satisfies* $\mathbb{P}$, or $\mathbb{P}(\alpha) = 0$, if when we substitute 0 for each true variable in α and 1 for each false variable in α then all polynomials in $\mathbb{P}$ are zeroed.

Definition 11 (PC refutation). The *PC translation of a clause* C is the product $\prod_{x \in P} x \times \prod_{x \in N}(1 - x)$ written out as a sum of monomials, where P is the set of variables which appear positively in C, and N is the set of variables appearing negatively. Note that the PC translation is defined in such a way that a literal x^b (where $b \in \{0, 1\}$) is satisfied if its PC translation is zeroed when substituting $x = b$.

A polynomial calculus refutation of a CNF formula F is a derivation of 1. The size measure for lines (polynomials) in a PC-derivation is the number of monomials in the polynomial. (counted with repetitions). The (monomial) space of a PC-configuration is the total number of monomials in the configuration (counted with repetitions).[3]

[3] Alekhnovich et al. [1] define monomial space as the maximal number of *distinct* monomials in any configuration. While their lower bounds hold even for this stricter definition (and so do ours), we think that their definition is somewhat artificial, and prefer the definition given here.

The representation of a clause $\bigvee_{i=1}^{n} \overline{x_i}$ as a PC-polynomial is $\prod_{i=1}^{n}(1 - x_i)$, which means that the number of monomials is exponential in the clause width. This problem arises only for negative literals, however—a large clause with only positive literals is translated to a single monomial. This is a weakness of monomial space in polynomial calculus when compared to clause space in resolution. In order to obtain a cleaner, more symmetric treatment of proof space, in [1] the proof system *polynomial calculus resolution (PCR)* was introduced as a common extension of polynomial calculus and resolution. The idea is to add an extra set of parallel formal variables $\overline{x}, \overline{y}, \overline{z}, \ldots$ so that positive and negative literals can both be represented in a space-efficient fashion. Thus, in PCR the clause $x \vee y \vee \overline{z}$ gets translated to the single monomial $xy\overline{z}$.

Definition 12 (Polynomial Calculus Resolution (PCR)). Lines in a PCR-proof are polynomials over the ring $\mathbb{F}[x, \overline{x}, y, \overline{y}, z, \overline{z}, \ldots]$, where as before $\mathbb{F}$ is some field. We have all the axioms and rules of PC plus the following axioms:

- **Complementarity:** $\dfrac{}{x + \overline{x} - 1}$ for all pairs of variables $(x, \overline{x})$.

A truth value assignment α to the variables $x, y, z, \ldots$ extends to an assignment $\tilde{\alpha}$ to the variables $x, \overline{x}, y, \overline{y}, z, \overline{z}, \ldots$ by assigning $\neg\alpha(x)$ to $\overline{x}$. The assignment α satisfies a PCR-configuration $\mathbb{P}$ if its extension $\tilde{\alpha}$ satisfies $\mathbb{P}$ (under the semantics of PC).

Definition 13 (PCR-refutation). The *PCR translation of a clause* C is the monomial $\prod_{x \in P} x \times \prod_{x \in N} \overline{x}$, where P is the set of variables which appear positively in C, and N is the set of variables which appear negatively in C. A PCR-refutation of a CNF formula is a derivation of 1. Size, length and space are defined as for PC.

In PCR, monomial space is a natural generalization of clause space, since every clause translates into one monomial as just explained.

An even stronger proof system, defined in [1], is functional calculus (FC). This purely semantical system works with arbitrary Boolean functions, regardless of their syntactical representation complexity, and the (clause) space complexity in FC is defined by minimizing the number of clauses needed to represent configurations as arbitrary functions of clauses. We refer to the full-length version of this paper for the details. Although FC is much stronger than PCR, it turns out that the lower bounds in [1] apply equally well to FC. The same is true for our PCR lower bounds.

III. A PCR Space Lower Bound

In this section, we present a PCR space lower bound for an encoding of the pigeonhole principle that has clauses of only logarithmic width. The lower bound we get is linear in the number of holes, just as in [1], but measured in the initial width of the clauses it is an exponential improvement. In the full-length version of this paper, we improve this result further to a qualitatively similar bound for CNF formulas of bounded width, resolving an open problem in [1]. We believe that the result in this section is of independent interest, however, since the lower bound holds for a natural family of CNF formulas, whereas the constant-width formulas in the full-length version are more contrived and are designed specifically to get PCR-space lower bounds.

The formulas we consider are so-called *bit-graph pigeonhole principle formulas*, which are encodings of the functional pigeonhole principle where the functionality condition that every pigeon should only go into one hole does not require extra axiom clauses but is hard-coded in the variable representation. Such an encoding arises naturally in bounded arithmetic. We employ the standard notation that $[n]$ denotes the set $\{1, \ldots, n\}$ (where n is a positive integer), that $[i, j]$ denotes the set $\{i, i+1, \ldots, j\}$, and that $[i, j)$ denotes the set $\{i, i+1, \ldots, j-1\}$.

Definition 14 (Bit-graph PHP formula). Let $n = 2^\ell$. The *bit-graph pigeonhole principle formula* $BPHP_n^m$ has propositional variables $x[p, i]$ for each $p \in [0, m)$ and $i \in [0, \ell)$. We think of $[0, m)$ as a set of pigeons and of $[0, n)$ as a set of holes. Each pigeon p is thought of as mapping to the hole whose binary encoding is given by the string $x[p, \ell-1] \cdots x[p, 1] x[p, 0]$, and we say that the variables $x[p, i]$ are *associated* with the pigeon p.

The formula $BPHP_n^m$ then asserts that no two pigeons map to the same hole. For every two pigeons $p_1 \neq p_2 \in [0, m)$ and every hole $h \in [0, n)$ we have a *hole axiom*

$$H(p_1, p_2, h) = \bigvee_{i=0}^{\ell-1} x[p_1, i]^{1-h_i} \vee \bigvee_{i=0}^{\ell-1} x[p_2, i]^{1-h_i} \ ,$$

stating that either p_1 is not mapped to h or p_2 is not mapped to h, where $h_{\ell-1} \cdots h_1 h_0$ is the binary encoding of h.

Recall our notational convention that for a variable v we have $v^0 \equiv v$ and $v^1 \equiv \overline{v}$, so that $v^b = 0$ (i.e., v^b is true) if and only if $v = b$. Then what the axiom clause $H(p_1, p_2, h)$ says is that for at least one of the pigeons p_1 or p_2, the binary expansion of the hole this pigeon is sent to does not match the binary expansion of h.

Fix $n = 2^\ell \geq 1$ and $m > n$. We will prove a PCR space lower bound for $BPHP_n^m$ using a similar construction to that in Alekhnovich et al. [1]. As in [1], our proof also applies to the much stronger functional calculus proof system. Notice that we can identify total truth value assignments α to the variables of $BPHP_n^m$ with functions $f_\alpha : [0, m) \to [0, n)$ mapping pigeons to holes. In what follows, we will switch freely back and forth between these two ways of looking at assignments.

Definition 15 (Well-behaved assignment). Let α be a total assignment to the variables of $BPHP_n^m$ and let $S \subseteq [0, m)$ be a set of pigeons. We say that α is *well-behaved on S* if the holes assigned by α to the pigeons in S are all distinct.

Definition 16 (Commitment). A *disjunctive commitment*, or just *commitment*, is a clause of the form $x[p_1, i_1]^{b_1} \vee x[p_2, i_2]^{b_2}$, where p_1 and p_2 are distinct pigeons.

A *commitment set* is a set of commitments where all pigeons are distinct. We think of a commitment set as the conjunction of its constituent commitments. The *domain* of a commitment set $\mathbb{A}$, written $\operatorname{dom} \mathbb{A}$, is the set of *pigeons* mentioned in $\mathbb{A}$. The *size* of a commitment set $\mathbb{A}$, denoted $|\mathbb{A}|$, is the number of commitments in $\mathbb{A}$.

An assignment α is *well-behaved on and satisfies* a commitment set $\mathbb{A}$ if α is well-behaved on $\operatorname{dom} \mathbb{A}$ and satisfies $\mathbb{A}$.

The following observation is central to our argument. It states that given a one-to-one assignment of fewer than $n/2$ pigeons to holes and a literal concerning a new pigeon, we can always find some new hole to assign to that pigeon so that the literal is satisfied.

Lemma 17. *Suppose S is any set of fewer than $n/2$ pigeons, α is an assignment well-behaved on S, and $x[p, i]^b$ is a literal associated with a pigeon $p \notin S$. Then we can modify α by reassigning p in such a way that the new assignment is well-behaved on $S \cup \{p\}$ and satisfies the literal $x[p, i]^b$.*

Proof: There are exactly $n/2$ holes for pigeon p that will satisfy the literal $x[p, i]^b$ if p is sent there. Fewer than $n/2$ holes are taken by the pigeons in S so there is a hole h, not assigned to any pigeon in S, whose assignment to p will satisfy $x[p, i]^b$. ■

Corollary 18. *Let S, T be two disjoint sets of pigeons such that $|S \cup T| \leq n/2$, and let X be a set containing exactly one literal associated with pigeon p for each $p \in T$. Then any assignment which is well-behaved on S can be modified, by reassigning pigeons in T, into an assignment which is well-behaved on $S \cup T$ and satisfies all literals in X.*

Proof: Consider the pigeons in T one by one, and apply Lemma 17. ■

Definition 19 (Entailment). Given a commitment set $\mathbb{A}$ and a PCR-configuration $\mathbb{P}$, we say that $\mathbb{A}$ *entails* $\mathbb{P}$ *over well-behaved assignments* if every assignment α which is well behaved on and satisfies $\mathbb{A}$ also satisfies $\mathbb{P}$.

The idea of the lower bound is that, given a purported refutation using small space, we can inductively construct a commitment set $\mathbb{A}_t$ for each configuration $\mathbb{P}_t$ in the proof in such a way that the commitment set $\mathbb{A}_t$ entails the configuration $\mathbb{P}_t$. The following lemma, based on a similar lemma in [1], is the technical heart of the lower bound. We will use it to show that as long as the configurations do not get too big, we never need to use a commitment set that is more than twice as large as its configuration. We can then use Corollary 18 to show that all the configurations are satisfiable, giving a contradiction.

Lemma 20 (Locality lemma). *Let $\mathbb{A}$ be a commitment set and $\mathbb{P}$ be a PCR-configuration such that $\mathbb{A}$ entails $\mathbb{P}$ over well-behaved assignments and $|\mathbb{A}| \leq n/4$. Then there is a commitment set $\mathbb{B}$ of size $|\mathbb{B}| \leq 2 \cdot Sp(\mathbb{P})$ such that $\mathbb{B}$ entails $\mathbb{P}$ over well-behaved assignments.*

Proof: Consider a bipartite graph with the left vertex set being the set M of all distinct monomials in $\mathbb{P}$, and the right vertex set being the set of all disjunctive commitments in $\mathbb{A}$. We draw an edge between a monomial $m \in M$ on the left and a commitment $C \in \mathbb{A}$ on the right if there is a pigeon p mentioned in both (that is, there is some variable $x[p,i]^b$ in m and some literal $x[p,i']^{b'}$ in C, where i may be different from i', and b from b'). To follow the rest of the argument, it might be helpful for the reader to consider the illustration in Figure 1(a).

Let $\Gamma \subseteq M$ be a set of maximal size such that $|N(\Gamma)| \leq 2 \cdot |\Gamma|$. Note that Γ is not necessarily unique, but such a maximal set always exists, since $\Gamma = \emptyset$ satisfies the requirement.

It must hold for all $S \subseteq M \setminus \Gamma$ that $|N(S) \setminus N(\Gamma)| > 2 \cdot |S|$, since otherwise we could add S to Γ to get a larger set. But this implies that there is a matching of every $m \in M \setminus \Gamma$ to two distinct commitments $C', C'' \in \mathbb{A} \setminus N(\Gamma)$ such that no two m, m' share any commitments. To see this, just make two copies of each monomial/vertex in $m \in M \setminus \Gamma$ with the same edges from both copies to the vertices on the right, apply Hall's theorem, and then identify the two copies of the monomial again (this step is depicted in Figure 1(b), where Γ and $N(\Gamma)$ are in the upper half of the graph).

Fix such a monomial $m \in M \setminus \Gamma$ and suppose it has been matched to the two disjunctive commitments $C' = x[p',i']^{b'} \vee x[q',j']^{c'}$ and $C'' = x[p'',i'']^{b''} \vee x[q'',j'']^{c''}$ as shown in Figure 1(c). By construction, m mentions at least one pigeon each from C' and C'', so suppose without loss of generality that p' and p'' are such pigeons. Thus, there exist literals $x[p',i_1]^{b_1}$ and $x[p'',i_2]^{b_2}$ such that $m = x[p',i_1]^{b_1} \cdot x[p'',i_2]^{b_2} \cdot m'$. We construct a new commitment $C_m = x[p',i_1]^{b_1} \vee x[p'',i_2]^{b_2}$. We construct commitments in this way for every $m \in M \setminus N(\Gamma)$, and let our new commitment set be $\mathbb{B} = N(\Gamma) \cup \{C_m \mid m \in M \setminus N(\Gamma)\}$, that is, the union of all these new commitments with the old commitments from $\mathbb{A}$ in $N(\Gamma)$. We claim that this is the commitment set we are looking for.

Firstly, it is easily verified that $\mathbb{B}$ is indeed a commitment set. This is so since all pigeons mentioned in $\mathbb{A}$ are different, and the pigeons in $\mathbb{B}$ are just a subset of the pigeons in $\mathbb{A}$. Secondly, with regard to size it clearly holds that $|\mathbb{B}| \leq |N(\Gamma)| + |M \setminus N(\Gamma)| \leq 2 \cdot |M| \leq 2 \cdot Sp(\mathbb{P})$. However, we also need to show that $\mathbb{B}$ entails $\mathbb{P}$ over well-behaved assignments. That is, we must prove that every β that is well-behaved on and satisfies $\mathbb{B}$ also satisfies $\mathbb{P}$.

We prove that $\mathbb{B}$ entails $\mathbb{P}$ over well-behaved assignments in a slightly roundabout way by finding, given any assignment β well-behaved on and satisfying $\mathbb{B}$, another assignment α such that

1) $\mathbb{P}(\alpha) = \mathbb{P}(\beta)$, and
2) α is well-behaved on and satisfies $\mathbb{A}$.

By item 2 it follows from the inductive hypothesis that α satisfies $\mathbb{P}$. But if so, then β also satisfies $\mathbb{P}$ by item 1, which is what we want to prove.

To this end, let $S = \operatorname{dom} \mathbb{B}$ and $T = \operatorname{dom} \mathbb{A} \setminus \operatorname{dom} \mathbb{B}$ (notice that $\operatorname{dom} \mathbb{B} \subseteq \operatorname{dom} \mathbb{A}$). Let X be the set of literals that for each $p \in T$ includes the (unique) literal $x[p,i]^b$ associated with p and appearing in $\mathbb{A}$. Notice that each commitment in $\mathbb{A} \setminus N(\Gamma)$ will have at least one literal in X (some commitments will potentially have both literals in X). Since $|\mathbb{A}| \leq n/4$, we have $|S \cup T| \leq n/2$. Apply Corollary 18 to S, T, and β to get a truth value assignment α that is well-behaved on $S \cup T$, agrees with β on pigeons outside T, and satisfies all literals in X. We claim that this is the assignment that we need.

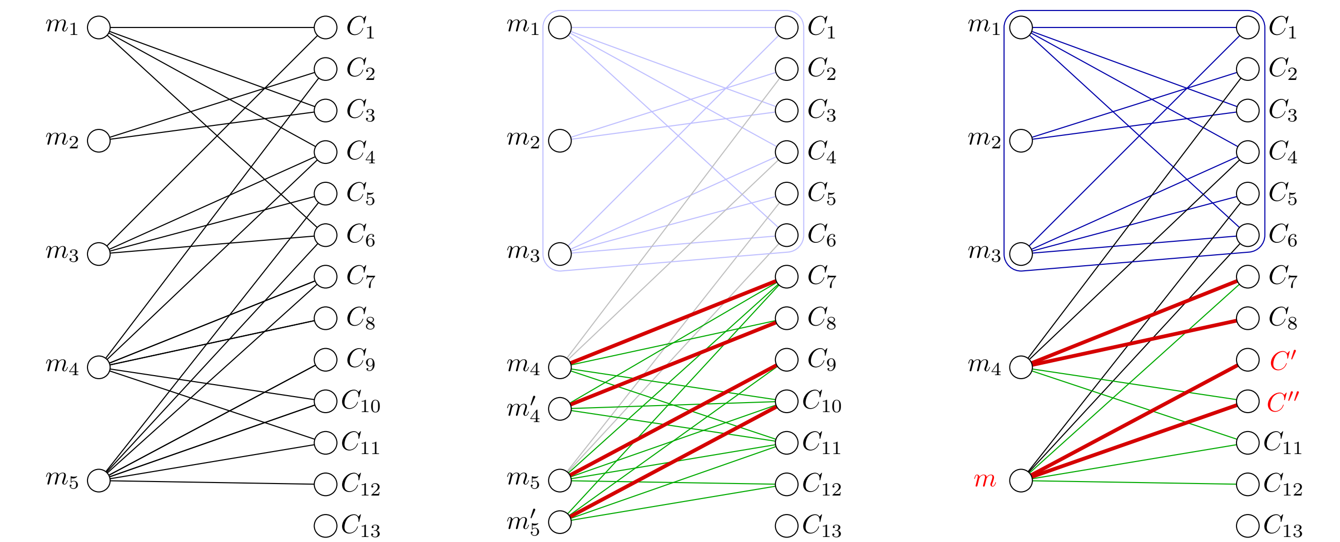
To see this, note first that no monomial in Γ mentions pigeons in T (by construction), so α and β agree on on monomials in Γ. For $m \in M \setminus \Gamma$, all β satisfying $\mathbb{B}$ must set the monomial m to zero, since this is how the new commitments were constructed. Reassigning pigeons in T can change variables in m, but there is still at least one variable that is set to zero, zeroing the whole monomial. So for all $m \in M$, the assignment α gives the same value to m as does β, namely 0. Hence α and β agree on all monomials in M and $\mathbb{P}(\alpha) = \mathbb{P}(\beta)$. This takes care of item 1 above. By Corollary 18, α is well-behaved on $S \cup T = \operatorname{dom} \mathbb{A}$. Also, since α satisfies all literals in X as well as $N(\Gamma)$, consequently α satisfies $\mathbb{A}$. This takes care of item 2, and as already discussed it now follows that $\mathbb{P}(\alpha) = 0$. Thus, every β that is well-behaved on and satisfies $\mathbb{B}$ must also satisfy $\mathbb{P}$. The lemma follows. ■

Using this Locality lemma, we can prove our PCR space lower bound for bit-graph PHP formulas.

Theorem 21 (Restatement of Theorem 3). $Sp_{\mathcal{PCR}}(BPHP_n^m \vdash \perp) > n/8$.

Proof: Let $\pi = \{\mathbb{P}_0, \ldots, \mathbb{P}_N\}$ be a PCR-refutation of $BPHP_n^m$ in monomial space at most $n/8$, with $\mathbb{P}_0 = \emptyset$ and $1 \in \mathbb{P}_N$. We will construct by induction a sequence of commitment sets $\mathbb{A}_0, \ldots, \mathbb{A}_N$ such that for each step t, it holds that $|\mathbb{A}_t| \leq 2 \cdot Sp(\mathbb{P}_t)$ and $\mathbb{A}_t$ entails $\mathbb{P}_t$ over well-behaved assignments. In particular, by Corollary 18 (with $S = \emptyset$) this will imply that every configuration $\mathbb{P}_t$ is satisfiable, which gives a contradiction for $\mathbb{P}_N$.

Clearly we may define $\mathbb{A}_0$ to be the empty commitment. Now suppose we have defined $\mathbb{A}_t$. To define $\mathbb{A}_{t+1}$ we consider three cases, depending on which step is used to obtain $\mathbb{P}_{t+1}$ from $\mathbb{P}_t$.



(a) Bipartite graph with monomials $m_i \in \mathbb{P}$ and commitments $C_j \in \mathbb{A}$.

(b) Hall's theorem applied on $M \setminus \Gamma$ (with monomial copies) and $\mathbb{A} \setminus N(\Gamma)$.

(c) Graphs with Γ and matchings constructed at end of argument.

Figure 1. Illustration of argument in proof of the Locality lemma.

Axiom download. We distinguish two download cases: (a) complementarity axioms of the form $x+\overline{x}-1$ or boolean axioms of the form x^2-x and (b) hole axioms $H(p_1,p_2,h)$. In the former case, we can simply set $\mathbb{A}_{t+1} = \mathbb{A}_t$ since any truth value assignment satisfies such an axiom by definition, so let us focus on hole axiom downloads.

Suppose that $\mathbb{P}_{t+1}$ is $\mathbb{P}_t$ together with some hole axiom $H(p_1,p_2,h)$. Suppose first that the pigeons mentioned in $H(p_1,p_2,h)$ are already in $\operatorname{dom}\mathbb{A}_t$. Then we put $\mathbb{A}_{t+1} = \mathbb{A}_t$. Let α be any assignment well-behaved on and satisfying $\mathbb{A}_{t+1}$. Then α satisfies $\mathbb{P}_t$ by the inductive hypothesis, and must also satisfy $H(p_1,p_2,h)$ since it is well-behaved on the pigeons in $H(p_1,p_2,h)$.

Otherwise, there are either one or two pigeons mentioned in $H(p_1,p_2,h)$ which are not in $\operatorname{dom}\mathbb{A}_t$. Then for each such pigeon p_i we add a "dummy" commitment C_{p_i} to $\mathbb{A}_t$ whose sole purpose is to put p_i into the domain of $\mathbb{A}_{t+1}$. We can take C_{p_i} to be $x[p_i,0] \vee x[p',0]$, where p' is any pigeon which has not been used so far.

In both cases, we add at most two new commitments, and so $|\mathbb{A}_{t+1}| \leq 2 \cdot \mathit{Sp}(\mathbb{P}_{t+1})$.

Inference. Suppose $\mathbb{P}_{t+1} = \mathbb{P}_t \cup \{P\}$, where P semantically follows from $\mathbb{P}_t$. We put $\mathbb{A}_{t+1} = \mathbb{A}_t$. Clearly $|\mathbb{A}_{t+1}| \leq 2 \cdot \mathit{Sp}(\mathbb{P}_{t+1})$. Suppose now that α is an assignment which is well-behaved on and satisfies $\mathbb{A}_{t+1}$. The induction hypothesis implies that α satisfies $\mathbb{P}_t$. Since P semantically follows from $\mathbb{P}_{t+1}$, α also satisfies P.

Erasure. Suppose $\mathbb{P}_{t+1} \subset \mathbb{P}_t$. Since $|\mathbb{A}_t| \leq 2 \cdot \mathit{Sp}(\mathbb{P}_t) \leq n/4$, Lemma 20 applies, and furnishes us with a commitment set $\mathbb{A}_{t+1}$ such that $|\mathbb{A}_{t+1}| \leq 2 \cdot \mathit{Sp}(\mathbb{P}_{t+1})$ and $\mathbb{A}_{t+1}$ entails $\mathbb{P}_{t+1}$ over well-behaved assignments. ∎

A nice feature of this lower bound is that it applies equally well to functional calculus (FC), where polynomials are replaced by arbitrary Boolean functions over clauses/monomials, with only very minor modifications. We refer to the full-length version for the details.

IV. Concluding Remarks

In this paper, we prove the first lower bounds on space in polynomial calculus (PC) and polynomial calculus resolution (PCR) for CNF formulas of constant width. This resolves a relatively longstanding open question from [1]. We also establish nontrivial upper bounds for proof size and space in PC, showing that for CNF formulas of constant width the worst-case behaviour is the same as for resolution and PCR. Finally, we study how the space complexity of a CNF formula is related to the space complexity of its standard transformation to 3-CNF, and show that for a certain class of CNF formulas, including the functional pigeonhole principle, the space complexity of a wide formula and its 3-CNF version coincide asymptotically for both resolution and PCR.

However, the concept of space is still a fairly poorly understood in polynomial calculus. This is all the more true for the cutting planes proof system discussed in the introduction, for which no nontrivial space lower bounds are known. Thus, many interesting and natural problems regarding space in polynomial calculus and cutting planes remain open. We conclude this paper by giving a (non-exhaustive) list of such problems which we believe merit further study.

1) Prove lower bounds on monomial space in PCR, or as a first step in PC, that match the worst-case upper bounds up to constant factors. That is, we are looking

for formulas (preferably of constant width) such that the monomial space lower bound is linear in the size of the formula. All lower bounds proven in this paper are sublinear even in the number of variables, not to mention the size of the formula (which for a k-CNF formula is asymptotically the same as the number of clauses).

2) Prove lower bounds on space in PCR or at least PC for random k-CNF formulas. (These formulas are excellent candidates for having space complexity matching the worst case upper bound as discussed above.)
3) Separate size and space in PCR by proving (or perhaps on the contrary ruling out) that there are k-CNF formulas that have small PCR-refutations but require large PCR-space.
4) Prove (or rule out, although that would be surprising) time-space trade-offs for PCR or at least for PC. That is, find formulas, preferably in constant width, which are easy with respect to monomial space and also have proofs of small size, but for which optimizing one of these measures must cause a dramatic blow-up in the other. In view of the recent paper [27], a natural candidate for such results are the so-called pebbling formulas studied in [10], [11], and another recent time-space trade-offs paper that might be relevant is [6].
5) Prove any nontrivial space lower bounds or time-space trade-offs for cutting planes, or indeed any lower bound on proof size for formulas such as random k-CNFs or Tseitin contradictions.
6) Can the space required in resolution or PCR for refuting the standard 3-CNF version $\widetilde{F}$ of a CNF formula be asymptotically less than that of the original, wide formula F? Or is it the case that the space complexity of F and $\widetilde{F}$ always coincide (asymptotically or even up to an additive term)? As a concrete example, is it true that the 3-CNF versions $\widetilde{PHP}^m_n$ of the pigeonhole principle formulas require space linear in n for PCR?

ACKNOWLEDGEMENTS

This article is the result of a long process, and various subsets of the authors would like to acknowledge useful discussions had during the last few years with various subsets of Paul Beame, Eli Ben-Sasson, Michael Brickenstein, Arkadev Chattopadhyay, Trinh Huynh, Johan Håstad, Alexander Razborov, and Iddo Tzameret.

The work presented in this paper was initiated at the Banff International Research Station workshop on proof complexity (11w5103) in October 2011 and part of the work was also performed during the special semester on Logic and Complexity at the Charles University in in Prague in the autumn of 2011 supported by the Marie Curie Initial Training Network MALOA (Mathematical Logic and Applications).

The research of the first author has received funding from the European Union's Seventh Framework Programme (FP7/2007–2013) under grant agreement no 238381. The second author was supported by the Eduard Čech Center for Algebra and Geometry (http://ecc.sci.muni.cz/). The third author was supported by Swedish Research Council grant 621-2010-4797 and by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007–2013) / ERC grant agreement no 279611. The fourth author was supported by grant IAA100190902 of GA AV ČR, by the Center of Excellence CE-ITI under grant P202/12/G061 of GA ČR and by RVO: 67985840. The research of the fifth author was supported by the Israel Ministry of Science and Technology. Also, part of the work of the fifth author was performed while visiting KTH Royal Institute of Technology supported by the foundations *Johan och Jakob Söderbergs stiftelse*, *Stiftelsen Längmanska kulturfonden*, and *Helge Ax:son Johnsons stiftelse*.

REFERENCES

[1] M. Alekhnovich, E. Ben-Sasson, A. A. Razborov, and A. Wigderson, "Space complexity in propositional calculus," *SIAM Journal on Computing*, vol. 31, no. 4, pp. 1184–1211, 2002, preliminary version appeared in *STOC '00*.

[2] M. Alekhnovich and A. A. Razborov, "Lower bounds for polynomial calculus: Non-binomial case," *Proceedings of the Steklov Institute of Mathematics*, vol. 242, pp. 18–35, 2003, available at http://people.cs.uchicago.edu/~razborov/files/misha.pdf. Preliminary version appeared in *FOCS '01*.

[3] A. Atserias, J. K. Fichte, and M. Thurley, "Clause-learning algorithms with many restarts and bounded-width resolution," *Journal of Artificial Intelligence Research*, vol. 40, pp. 353–373, Jan. 2011, preliminary version appeared in *SAT '09*.

[4] R. J. Bayardo Jr. and R. Schrag, "Using CSP look-back techniques to solve real-world SAT instances," in *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI '97)*, Jul. 1997, pp. 203–208.

[5] P. Beame, "Proof complexity," in *Computational Complexity Theory*, ser. IAS/Park City Mathematics Series, S. Rudich and A. Wigderson, Eds. American Mathematical Society, 2004, vol. 10, pp. 199–246.

[6] P. Beame, C. Beck, and R. Impagliazzo, "Time-space tradeoffs in resolution: Superpolynomial lower bounds for superlinear space," in *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC '12)*, May 2012, to appear.

[7] E. Ben-Sasson, "Size space tradeoffs for resolution," *SIAM Journal on Computing*, vol. 38, no. 6, pp. 2511–2525, May 2009, preliminary version appeared in *STOC '02*.

[8] E. Ben-Sasson and N. Galesi, "Space complexity of random formulae in resolution," *Random Structures and Algorithms*, vol. 23, no. 1, pp. 92–109, Aug. 2003, preliminary version appeared in *CCC '01*.

[9] E. Ben-Sasson and J. Johannsen, "Lower bounds for width-restricted clause learning on small width formulas," in *Proceedings of the 13th International Conference on Theory and Applications of Satisfiability Testing (SAT '10)*, ser. Lecture Notes in Computer Science, vol. 6175. Springer, Jul. 2010, pp. 16–29.

[10] E. Ben-Sasson and J. Nordström, "Short proofs may be spacious: An optimal separation of space and length in resolution," in *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS '08)*, Oct. 2008, pp. 709–718.

[11] ——, "Understanding space in proof complexity: Separations and trade-offs via substitutions," in *Proceedings of the 2nd Symposium on Innovations in Computer Science (ICS '11)*, Jan. 2011, pp. 401–416, full-length version available at http://eccc.hpi-web.de/report/2010/125/.

[12] E. Ben-Sasson and A. Wigderson, "Short proofs are narrow—resolution made simple," *Journal of the ACM*, vol. 48, no. 2, pp. 149–169, Mar. 2001, preliminary version appeared in *STOC '99*.

[13] A. Biere, M. J. H. Heule, H. van Maaren, and T. Walsh, Eds., *Handbook of Satisfiability*, ser. Frontiers in Artificial Intelligence and Applications. IOS Press, Feb. 2009, vol. 185.

[14] A. Blake, "Canonical expressions in Boolean algebra," Ph.D. dissertation, University of Chicago, 1937.

[15] M. Bonet, T. Pitassi, and R. Raz, "Lower bounds for cutting planes proofs with small coefficients," in *Proceedings of the 27th Annual ACM Symposium on Theory of Computing (STOC '95)*, May 1995, pp. 575–584.

[16] M. Brickenstein and A. Dreyer, "PolyBoRi: A framework for Gröbner-basis computations with Boolean polynomials," *Journal of Symbolic Computation*, vol. 44, no. 9, pp. 1326–1345, Sep. 2009.

[17] M. Brickenstein, A. Dreyer, G.-M. Greuel, M. Wedler, and O. Wienand, "New developments in the theory of Gröbner bases and applications to formal verification," *Journal of Pure and Applied Algebra*, vol. 213, no. 8, pp. 1612–1635, Aug. 2009.

[18] S. R. Buss, J. Hoffmann, and J. Johannsen, "Resolution trees with lemmas: Resolution refinements that characterize DLL-algorithms with clause learning," *Logical Methods in Computer Science*, vol. 4, no. 4:13, Dec. 2008.

[19] V. Chvátal and E. Szemerédi, "Many hard examples for resolution," *Journal of the ACM*, vol. 35, no. 4, pp. 759–768, Oct. 1988.

[20] M. Clegg, J. Edmonds, and R. Impagliazzo, "Using the Groebner basis algorithm to find proofs of unsatisfiability," in *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC '96)*, May 1996, pp. 174–183.

[21] S. A. Cook and R. Reckhow, "The relative efficiency of propositional proof systems," *Journal of Symbolic Logic*, vol. 44, no. 1, pp. 36–50, Mar. 1979.

[22] W. Cook, C. R. Coullard, and G. Turán, "On the complexity of cutting-plane proofs," *Discrete Applied Mathematics*, vol. 18, no. 1, pp. 25–38, Nov. 1987.

[23] M. Davis, G. Logemann, and D. Loveland, "A machine program for theorem proving," *Communications of the ACM*, vol. 5, no. 7, pp. 394–397, Jul. 1962.

[24] M. Davis and H. Putnam, "A computing procedure for quantification theory," *Journal of the ACM*, vol. 7, no. 3, pp. 201–215, 1960.

[25] J. L. Esteban and J. Torán, "Space bounds for resolution," *Information and Computation*, vol. 171, no. 1, pp. 84–97, 2001, preliminary versions of these results appeared in *STACS '99* and *CSL '99*.

[26] A. Haken, "The intractability of resolution," *Theoretical Computer Science*, vol. 39, no. 2-3, pp. 297–308, Aug. 1985.

[27] T. Huynh and J. Nordström, "On the virtue of succinct proofs: Amplifying communication complexity hardness to time-space trade-offs in proof complexity," in *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC '12)*, May 2012, to appear.

[28] R. Impagliazzo, P. Pudlák, and J. Sgall, "Lower bounds for the polynomial calculus and the Gröbner basis algorithm," *Computational Complexity*, vol. 8, no. 2, pp. 127–144, 1999.

[29] M. Järvisalo, A. Matsliah, J. Nordström, and S. Živný, "Relating proof complexity measures and practical hardness of SAT," 2012, submitted.

[30] J. P. Marques-Silva and K. A. Sakallah, "GRASP—a new search algorithm for satisfiability," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD '96)*, Nov. 1996, pp. 220–227.

[31] J. Nordström, "A simplified way of proving trade-off results for resolution," *Information Processing Letters*, vol. 109, no. 18, pp. 1030–1035, Aug. 2009, preliminary version appeared in ECCC report TR07-114, 2007.

[32] ——, "Pebble games, proof complexity and time-space trade-offs," *Logical Methods in Computer Science*, 2012, to appear. Available at http://www.csc.kth.se/~jakobn/research/.

[33] P. Pudlák, "Lower bounds for resolution and cutting plane proofs and monotone computations," *Journal of Symbolic Logic*, vol. 62, no. 3, pp. 981–998, Sep. 1997.

[34] A. A. Razborov, "Lower bounds for the polynomial calculus," *Computational Complexity*, vol. 7, no. 4, pp. 291–324, Dec. 1998.

[35] J. A. Robinson, "A machine-oriented logic based on the resolution principle," *Journal of the ACM*, vol. 12, no. 1, pp. 23–41, Jan. 1965.

[36] N. Segerlind, "The complexity of propositional proofs," *Bulletin of Symbolic Logic*, vol. 13, no. 4, pp. 482–537, Dec. 2007.

[37] A. Urquhart, "Hard examples for resolution," *Journal of the ACM*, vol. 34, no. 1, pp. 209–219, Jan. 1987.