Electronic Colloquium on Computational Complexity, Report No. 132 (2012)

# Space Complexity in Polynomial Calculus[*†]

Yuval Filmus
University of Toronto
yuval.filmus@utoronto.ca

Massimo Lauria
KTH Royal Institute of Technology
lauria@kth.se

Jakob Nordström
KTH Royal Institute of Technology
jakobn@kth.se

Noga Ron-Zewi
Technion – Israel Institute of Technology
nogaz@cs.technion.ac.il

Neil Thapen
Academy of Sciences of the Czech Republic
thapen@math.cas.cz

October 20, 2012

## 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, and so research has mostly focused on weak systems that are used by SAT solvers.

There has been a relatively long sequence of papers on space in resolution, which is now reasonably well understood from this point of view. For other natural candidates to study, however, such as polynomial calculus or cutting planes, very little has been known. We are not aware of any nontrivial space lower bounds for cutting planes, and for polynomial calculus the only lower bound has been for CNF formulas of unbounded width in [Alekhnovich et al. '02], where the space 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 be able to refute any $k$-CNF formula in constant space.

In this paper, 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. 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.
2. For PCR, we prove an $\Omega(n)$ space lower bound for a bitwise encoding of the functional pigeonhole principle. 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.
3. 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.
4. Finally, 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 obviously 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, something that we believe can be useful when proving PCR space lower bounds for other well-studied formula families in proof complexity.

[*] This work was initiated at the BIRS workshop on proof complexity (11w5103) in October 2011 and part of the work was also performed during the special MALOA semester on Logic and Complexity in Prague in the autumn of 2011.

[†] This is a full-length version of the paper [FLN+12] which appeared in *Proceedings of the 27th Annual IEEE Conference on Computational Complexity (CCC '12).*

ISSN 1433-8092

# 1 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 $\pi$ (a *proof*) for which $P(x, \pi) = 1$, while for any $x \notin L$ it holds for all strings $\pi$ 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 [CR79], 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 $\pi_x$ of size at most polynomial in the size of $x$. As observed in [CR79], one way of establishing co-NP $\neq$ NP, and hence P $\neq$ 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 [DLL62, DP60] from the early 1960s augmented with clause learning [BS97, MS96]; 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 [SAT] 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 [AFT11, BHJ08, BJ10] 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.

Concluding this brief general discussion, we want to mention that some good starting points for a further study of proof complexity in general are, for instance, [Bea04, Seg07], while the upcoming survey [Nor12] by the third author focuses specifically on space complexity and time-space trade-offs. On the more applied side, a recent, very comprehensive, reference on SAT solving is the *Handbook of Satisfiability* [BHvMW09].

## 1.1 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 $\pi$ can be

[1] Perhaps we should point out that this is more than just a vague hope that theory and practice should somehow be related—for instance, recent experimental work by the third author joint with Järvisalo, Matsliah and Živný [JMNŽ12] seems to indicate a correlation between theoretical space complexity in resolution and practical hardness (measured as the running time) for CDCL solvers.

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 [Bla37] and began to be investigated in connection with automated theorem proving in the 1960s [DLL62, DP60, Rob65]. 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 [Hak85] 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, [BKPS02, BW01, CS88, Urq87].

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 [ET01]. Alekhnovich et al. [ABRW02] 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 [ABRW02, BG03, ET01] have established matching lower bounds (up to constant factors).

Another sequence of papers [Nor09a, NH08, BN08] involving the third author clarified the relation between length and space in resolution. While it follows from [AD08] that small complexity with respect to space implies small complexity with respect to length, building on [Nor09a, NH08] the paper [BN08] established 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 $\Omega(n/\log n)$ (and this separation is optimal). Regarding trade-offs between length and space, some results in restricted settings were presented in [Ben09, Nor09b], and strong trade-offs for general resolution were finally obtained in [BN11]. Even more recently, [BBI12] obtained 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. [CEI96], 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 [IPS99], and a number of strong lower bounds on proof size have been obtained by proving degree lower bounds in, for instance, [AR03, BGIP01, BI10, IPS99, Raz98].

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, [ABRW02] 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, [ABRW02] managed to establish lower bounds on space measured as the number of monomials, but only sublinear bounds and only 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

[2] As a side note, we remark 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.

of PCR-space or even PC-space, has remained open since [ABRW02]. Thus, it has been 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 [HN12].

At the time the paper [CEI96] 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 [BD09], but in general they seem to be an order of magnitude slower than state-of-the-art CDCL solvers (although [BDG$^+$09] reports that PolyBoRi can be faster on certain specific industrial instances).

To conclude our quick survey on research on space complexity and size-space trade-offs in proof complexity, we also want to briefly discuss the *cutting planes (CP)* proof system [CCT87]. Here the clauses of a CNF formula are translated to linear inequalities and the formula is refuted by showing that the polytope defined by these inequalities does not have any zero-one integer points (corresponding to satisfying assignments). We only know of one superpolynomial lower bound on CP proof size [Pud97] (improving on the result [BPR95] in a somewhat restricted setting). It is natural to define the *line space* of a CP-refutation to be the maximal number of linear inequalities that need to be kept in memory simultaneously during the refutation. Just as for monomial space in PCR, line space in CP is easily seen to be a generalization of clause space in resolution and is hence upper bounded by the clause space complexity. 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 [HN12].

## 1.2 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 [Nor12] fairly closely, but for completeness we provide all the necessary preliminaries in Section 2.

### 1.2.1 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 [BW01] 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 1.1.** *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 $k$.*

### 1.2.2 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 [ABRW02] 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 $PHP^m_n$ from any PCR-refutation of $\widetilde{PHP}^m_n$ 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 $PHP^m_n$ from any PC-refutation of $\widetilde{PHP}^m_n$. This immediately yields a first nontrivial lower bound on space for 3-CNF formulas in PC. Using the ideas from Section 1.2.1 together with results from the literature, it is not hard to show that this bound is asymptotically tight.

**Theorem 1.2.** *The space of refuting $\widetilde{PHP}^{n+1}_n$ 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 [ABRW02], that Theorem 1.2 holds even if we allow arbitrary flips of literal polarities in the formula.

### 1.2.3 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 1.2 no longer works, for the reasons sketched above.

As a first step, we instead consider an alternative encoding of the pigeonhole principle. In the $PHP^m_n$ 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 [ABRW02] measured in the width of the formulas.

**Theorem 1.3.** *The space in PCR of refuting bit-graph pigeonhole principle formulas $BPHP_n^{n+1}$ is $\Omega(n)$, or $\Omega\bigl(\sqrt[3]{N/\log N}\bigr)$ 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 1.3 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 1.4.** *The space in PCR of refuting the XOR pigeonhole principle $XPHP_n^{n+1}$ encoded in 4-CNF is $\Omega(n)$, or $\Omega\bigl(\sqrt[3]{N}\bigr)$ expressed in the formula size $N$.*

The proofs of Theorems 1.3 and 1.4 are very much inspired by [ABRW02] and follow the arguments in that paper fairly closely, but also require some subtle but crucial twists. We refer to Section 5 for the proof of Theorem 1.3. The easy modifications to prove Theorem 1.4 are described in Section 6.

### 1.2.4 Space Complexity of Wide CNF Formulas and Their 3-CNF Versions

While Theorem 1.4 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 1.2.2, 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 $a_1 \lor \ldots \lor a_w$ in $F$ there are also axioms $\overline{a}_i \lor \overline{a}_j$ 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 1.2.2 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 1.5.** *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 [ABRW02] break down when functional axioms are added to the formula. However, what Theorem 1.5 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 1.5 could be generalized to hold for any CNF formula, even without weight constraints, or if there is some counter-example.

### 1.3 Subsequent Developments

After the first version of this paper was published, in [BNT12] Beck and Tang together with the third author have extended the size-space trade-off results for resolution in [BN11] and [BBI12] to PCR, albeit with a slight loss in the parameters which leaves room for further improvements.

Also, just as the full-length version of this paper was being finalized, Bonacina and Galesi [BG12] announced a further generalization of the techniques in [ABRW02] and our paper leading to among other things optimal (linear) space lower bounds in PCR for random $k$-CNF formulas with $k \geq 4$.

### 1.4 Outline of This Paper

The rest of this paper is organized as follows. We start by presenting the necessary preliminaries in Section 2. In Section 3, we prove that the space of refuting $k$-CNF formulas in polynomial calculus is at most $\mathrm{O}(n)$, where $n$ is the number of variables, and in Section 4 we prove a polynomial calculus space lower bound for 3-CNF versions of the pigeonhole principle formulas. In Sections 5 and 6 we then present what we consider to be our main results, namely space lower bounds in polynomial calculus resolution (PCR) for CNF formulas of small width. In Section 5, we study formulas of logarithmic width, and in Section 6 we extend the result to a family of CNF formulas of constant width. In Section 7, we consider CNF formulas where all wide clauses have weight constraints which specify that exactly one literal is true. We show that the space for refuting these formulas and their standard 3-CNF versions coincide asymptotically both in resolution and PCR. Finally, in Section 8, we make some concluding remarks and mention a few of the many fascinating problems in this area that remain open.

## 2 Preliminaries

Let $x$ be a Boolean variable. A *literal over* $x$ is either the variable itself or its negation, denoted $\neg x$ or $\overline{x}$. The former is called a *positive literal*, the latter a *negative literal*. 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 $b = 1$. Note that this notational convention is the opposite of what is found in some other papers in the proof complexity literature, but as we will see shortly it is the natural choice in the context of polynomial calculus.

A *clause* $C = a_1 \vee \cdots \vee a_k$ is a disjunction of literals. Below we will think of clauses as sets, so that the ordering of the literals is immaterial and no literals are repeated. 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. We will think of CNF formulas as sets 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 $\alpha, \beta$ to denote truth value assignments. An assignment *satisfies* a Boolean function if it makes the function true. For example, a clause is true if any of its constituent literals is true; the empty clause $\perp$ is always false. 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 $\pi$ 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*. Following the exposition in [ET01], we view a proof as similar to a non-deterministic Turing machine computation, with a special read-only input tape from which the clauses of the CNF formula $F$ being refuted (the *axioms*) can be downloaded and a working memory where all derivation steps are made. Then the length of a proof is essentially the time of the computation and space measures memory consumption. The following definition is a straightforward generalization of [ABRW02]. We employ the standard notation $[n] = \{1, \ldots, n\}$.

**Definition 2.1 (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 $\pi \colon F \vdash \bot$ 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 $\bot \in \mathbb{D}_\tau$, where $\bot$ is the representation of contradiction (e.g. for resolution the empty clause without literals).

**Definition 2.2 (Refutation size, length and space).** Given a size measure $S(L)$ for lines $L$ in $\mathcal{P}$-derivations (which we usually think of as the number of symbols in $L$, but other definitions can also be appropriate depending on the context), the *size* of a $\mathcal{P}$-derivation $\pi$ 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 $\pi$ is the number of axiom downloads and inference steps in it.[3] For a space measure $Sp_{\mathcal{P}}(\mathbb{D})$ defined for $\mathcal{P}$-configurations, the *space* of a derivation $\pi$ is defined as the maximal space of a configuration in $\pi$.

We define the *$\mathcal{P}$-refutation size* of a formula $F$, denoted $S_{\mathcal{P}}(F \vdash \bot)$, to be the minimum size of any $\mathcal{P}$-refutation of it. The *$\mathcal{P}$-refutation length* $L_{\mathcal{P}}(F \vdash \bot)$ and *$\mathcal{P}$-refutation space* $Sp_{\mathcal{P}}(F \vdash \bot)$ 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 2.1 of the proof systems that will be of interest in this paper. Below, the notation $\frac{G_1 \quad \cdots \quad G_m}{H}$ means that if $G_1, \ldots, G_m$ have been derived previously in the proof (and are currently in memory), then we can infer $H$.

**Definition 2.3 (Resolution).** In resolution, the lines in a derivation are clauses and inferences follow the *resolution rule*:

$$\frac{B \vee x \qquad C \vee \overline{x}}{B \vee C} \tag{2.1}$$

for clauses $B$ and $C$. We refer to (2.1) as *resolution on the variable* $x$ and 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*:

$$\frac{B}{B \vee C} \tag{2.2}$$

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, the length measure is as defined in Definition 2.2. We will consider three separate space measures: *clause space*, *total space* and *width*.

**Definition 2.4 (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 $\pi$ is the maximum that the corresponding measures attains over any configuration $\mathbb{C} \in \pi$.

[3] The reader who so prefers can instead define the length of a derivation $\pi = \{\mathbb{D}_0, \ldots, \mathbb{D}_\tau\}$ as the number of steps $\tau$ in it, since the difference is at most a factor of 2. We have chosen the definition above for consistency with previous papers defining length as the number of lines in a listing of the derivation.

*Remark* 2.5. When studying and comparing the complexity measures for resolution in Definition 2.4, as was noted in [ABRW02] it is preferable to prove the results for $k$-CNF formulas, i.e., formulas where all clauses have size upper-bounded by some constant. This is especially so since the width and space measures can "misbehave" rather artificially for formula families of unbounded width (see [Nor09b, Section 5] for a discussion of this). Since every CNF formula can be rewritten as an equivalent formula of bounded width, it therefore seems natural to insist that the formulas under study should have width bounded by some constant.

The *polynomial calculus (PC)* proof system was introduced in [CEI96] under the name of "Gröbner proof system." In a PC-refutation, clauses are interpreted as multilinear polynomials. For instance, the requirement that the clause $x \lor y \lor \overline{z}$ 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 2.6 (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 $\alpha$ to variables and a PC configuration $\mathbb{P}$, we say that $\alpha$ *satisfies* $\mathbb{P}$, or $\mathbb{P}(\alpha) = 0$, if when we substitute 0 for each true variable in $\alpha$ and 1 for each false variable in $\alpha$ then all polynomials in $\mathbb{P}$ are zeroed.

**Definition 2.7 (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).[4] The (monomial) space of a PC-configuration (a set of polynomials) is the total number of monomials in the configuration (counted with repetitions).[5]

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 [ABRW02] 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 $x \lor y \lor \overline{z}$ gets translated to the single monomial $xy\overline{z}$.

[4] Note that if one wanted to nitpick, one could argue that the number of variables in each monomial should also be counted to get a true size measure. However, size as defined in Definition 2.7, which is the standard definition in the literature, clearly is within a linear factor of this and is much cleaner to work with (assuming that the field $\mathbb{F}$ is constant so that we do not need to worry about issues regarding representation of the coefficients).

[5] Alekhnovich et al. [ABRW02] 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.

**Definition 2.8 (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 $\alpha$ 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 $\alpha$ satisfies a PCR-configuration $\mathbb{P}$ if its extension $\tilde{\alpha}$ satisfies $\mathbb{P}$ (under the semantics of PC).

**Definition 2.9 (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.

The point of the complementarity rule is to force $x$ and $\overline{x}$ to have opposite values in $\{0, 1\}$, so that they encode complementary literals. This means that one can potentially avoid an exponential blow-up in size measured in the number of monomials (and thus also for space). In PCR, monomial space is a natural generalization of clause space, since every clause translates into one monomial as just explained.

We remark that although the measure of total space, considering the total number of symbols in memory, is perhaps a priori the most natural one, most papers on proof space have focused on space measured as the number of lines in memory (in particular, the number of clauses). However, as observed in [ABRW02], for strong enough proof systems, this "line space" measure is no longer interesting since just one unit of memory can contain a big AND of all formulas derived so far. But this measure does make perfect sense for resolution. For PC/PCR, however, measuring just the number of polynomial equations is not meaningful, since every equation can be of exponential size and encode very much information. Instead, the natural generalization of clause space is monomial space.

In general, admissible inferences in a sequential proof system are defined by a set of syntactic inference rules. In what follows, we will also be interested in a strengthened version of this concept, which was made explicit in [ABRW02].

**Definition 2.10 (Syntactic and semantic derivations).** We refer to derivations according to Definition 2.1, where each new line $L$ has to be inferred by one of the inference rules for $\mathcal{P}$, as *syntactic* derivations. If instead *any line $L$* that is semantically implied by the current configuration can be derived in one atomic step, we talk about a *semantic* derivation.

More precisely, in a resolution refutation, a clause $D$ can be inferred from a configuration $\mathbb{C}$ if every truth assignment which satisfies all clauses in $\mathbb{C}$ also satisfies $D$. In a semantic PC refutation, a polynomial $Q$ can be inferred from a clause configuration $\mathbb{P}$ if every $0/1$ assignment to the variables in $\mathbb{P}$ which zeroes all polynomials in $\mathbb{P}$ also zeroes $Q$. Semantic PCR is defined similarly.

Clearly, semantic derivations are at least as strong as syntactic ones, and they are easily seen to be superpolynomially stronger with respect to length for any proof system where superpolynomial lower bounds are known. This is so since a semantic proof system can download all axioms in the formula one by one, and then deduce contradiction in one step since the formula is unsatisfiable. Therefore, semantic versions of proof systems are mainly interesting when we want to reason about space or the relationship between space and length. If we can prove lower bounds not just for syntactic but even semantic versions of proof systems, this of course makes these bounds much stronger.

An even stronger proof system, defined in [ABRW02], is functional calculus (FC). This purely semantical system works with arbitrary Boolean functions, regardless of their syntactical representation complexity. In fact the space complexity for FC defined below will simply minimize over all such representations. Although this system is not natural, the space lower bound for PCR applies to it as well, and it is a useful tool for proving lower bounds when an abstraction from particulars of a given syntactical system is desirable and instructive.

**Definition 2.11 (Functional Calculus (FC)).** The line of a *functional calculus* derivation is an arbitrary Boolean function (that is, a Boolean-valued function of Boolean variables). The single inference rule is the semantical one, i.e., derive $g$ from $f_1, \ldots, f_n$ whenever every truth assignment that satisfies $f_1, \ldots, f_n$ also satisfies $g$. An FC-refutation of a CNF formula is a functional calculus proof of 1 from Boolean axioms and the clauses in the formula, considered as Boolean functions.

When defining the clause space of FC-configurations, we must overcome the following problem. A line in FC is an *arbitrary* Boolean function $f$. Clearly, $f$ can be represented by many circuits over some complete Boolean basis, each with a different amount of clauses. The natural way to solve this problem is to define the clause space to be the minimal number of clauses in any such representation.

**Definition 2.12 (FC clause space).** Let $\mathbb{P}$ be a set of Boolean functions over variables $x_1, \ldots, x_n$. The *(clause) space* of $\mathbb{P}$ in FC, denoted $Sp_{\mathcal{FC}}(\mathbb{P})$, is the minimal $s$ such that we can choose $s$ clauses with the property that every $f \in \mathbb{P}$ can be represented as a Boolean function over the chosen clauses. Formally, $Sp_{\mathcal{FC}}(\mathbb{P})$ is defined as

$$\min\{s : \exists\{C_i(x_1, \ldots, x_n) \mid i \in [s]\} \; \forall f(x_1, \ldots, x_n) \in \mathbb{P} \; \exists g(y_1, \ldots, y_s) \; f \equiv g(C_1, \ldots, C_s)\} \;,$$

where $C_1, \ldots, C_s$ are clauses, and $g$ runs over arbitrary Boolean functions in $s$ variables. The space of an FC-refutation is the maximal space of any configuration $\mathbb{P}$ encountered during the proof.

A proof in PCR can be converted to a proof in FC by replacing each polynomial $L(x_1, \ldots, x_n)$ by the Boolean function

$$f(x_1, \ldots, x_n) = \begin{cases} \top & \text{if } L(x_1, \ldots, x_n) = 0, \\ \bot & \text{if } L(x_1, \ldots, x_n) \neq 0. \end{cases}$$

The space of a PCR configuration $\mathbb{P}$ under FC is at most the number of distinct monomials appearing in $\mathbb{P}$. In particular, switching from the PCR space measure to the FC space measure cannot increase space. This shows that FC space lower bounds also apply to PCR. Although FC is potentially much stronger than PCR, it turns out that the lower bounds in [ABRW02] apply equally well to FC. The same is true for the PCR lower bounds proven in this paper.

## 3 Upper Bounds on Space for $k$-CNFs in Polynomial Calculus

In this section we prove that any narrow CNF can be refuted in polynomial calculus using linear space and exponential length simultaneously. More precisely we show the following theorem.

**Theorem 3.1 (Detailed version of Theorem 1.1).** *Let $F$ be an unsatisfiable $k$-CNF formula over $n$ variables. Then $F$ can be refuted in the PC proof system in space $2^k(2n+9)$ and length $n4^{n+1}$.*

For the proof of the above theorem we introduce the notion of *negative width* of a formula $F$ which is the maximum number of negative literals in a clause in the refutation of $F$ (see formal definition below). The reason for introducing this notion is that, as observed in Definition 2.7, the representation of negative literals in the PC system generally requires more space then the representation of the positive ones. Given the definition of negative width, the proof of the theorem consists of two main steps. In the first step we show that any unsatisfiable $k$-CNF formula in $n$ variables can be refuted in the resolution proof system in space $n+2$, negative width $k$, and length $4^n$ simultaneously. In the second part we show that such a refutation can be simulated in the PC proof system in space $2^k(2n+9)$ and length $n4^{n+1}$ simultaneously. We start by defining formally the notion of negative width.

**Definition 3.2 (Negative width).** The *negative width* $W^-(C)$ of a clause $C$ is the number of negative literals in $C$, and the negative width $W^-(\mathbb{C})$ of a clause configuration $\mathbb{C}$ is the maximum negative width of a clause in it. The negative width $W^-(\pi)$ of a resolution refutation $\pi$ is the maximum negative width of a configuration in $\pi$. The negative width $W^-(F \vdash \bot)$ of a CNF formula $F$ is the minimum negative width of a resolution refutation of $F$.

Our first main lemma says that unsatisfiable $k$-CNFs can be refuted in linear space, exponential length, and in the smallest possible negative width.

**Lemma 3.3.** *Let $F$ be an unsatisfiable $k$-CNF formula in $n$ variables. Then $F$ can be refuted in resolution in space $n + 2$, negative width $k$, and length $4^n$ simultaneously.*

For the proof, we need the following lemma from [ET01].

**Lemma 3.4 ([ET01]).** *Let $F$ be an unsatisfiable CNF formula in $n$ variables. Then $F$ can be refuted in resolution in space $n + 2$ and length $2^{n+1} - 1$ simultaneously.*

*Proof of Lemma 3.3.* The proof is by induction on the number of variables $n$. If $n \leq k$ then it follows from Lemma 3.4 that $F$ has a refutation $\pi$ of space at most $n + 2$ and length at most $2^{n+1} - 1 \leq 4^n$. Since there are only $k$ different variables the negative width of $\pi$ is necessarily at most $k$.

For the induction step, suppose that the statement holds for every $k$-CNF formula in $n$ variables and we shall prove that it holds for every $k$-CNF formula in $n+1$ variables as well. Let $F$ be an unsatisfiable $k$-CNF formula in $n + 1$ variables. For $b \in \{\top, \bot\}$ and for any clause $C \in F$ let $C \restriction_{x=b}$ denote the clause obtained from $C$ by setting $x$ to $b$. That is,

$$C \restriction_{\{x=\bot\}} = \begin{cases} \top & \neg x \in C \\ C \setminus \{x\} & x \in C \\ C & \text{otherwise} \end{cases} \qquad C \restriction_{\{x=\top\}} = \begin{cases} C \setminus \{\neg x\} & \neg x \in C \\ \top & x \in C \\ C & \text{otherwise} \end{cases} \tag{3.1}$$

Let $F \restriction_{x=b}$ be the $k$-CNF formula which contains all clauses of the form $C \restriction_{x=b}$ where $C \in F$.

Pick an arbitrary variable $x$ and consider the restricted formula $F \restriction_{\{x=\bot\}}$. By inductive hypothesis, it has a refutation of length $4^n$, space $n + 2$ and negative width $k$. It is an easy observation that from this refutation we obtain a proof $F \vdash x$ in the same length, space and negative width. In order to do so substitute the initial clauses in the former refutation with the corresponding initial clauses in $F$, and do the same inference steps. It is easy to see that for any clause $C$ in the refutation of $F \restriction_{\{x=\bot\}}$, the corresponding clause in the new proof is either $C$ or $C \vee x$. Thus the final configuration in this proof includes either the empty clause $\bot$ or the clause $x$. In either case, $x$ can be derived via weakening. Proof length and proof space do not change and, since $x$ is a positive literal, neither negative width does. From now on we keep the clause $x$ in memory.

We now consider the formula $F \restriction_{\{x=\top\}}$. One natural approach is to use the same strategy as above for deriving the clause $\neg x$ from $F$ and conclude by resolving $\neg x$ with the literal $x$ in memory. However, we cannot use this strategy here because it may cause an increase of negative width. We instead simulate such refutation directly: actually we just need to simulate downloads of the clauses of $F \restriction_{\{x=\top\}}$ which are not in $F$. Such clauses are of the form $C$ where $C \vee \neg x$ is in $F$, so it is sufficient to download $C \vee \neg x$, to resolve with $x$ which is in memory and then to erase $C \vee \neg x$. Notice that this simulation uses two additional monomials in memory. In the end we get the empty clause, which concludes the refutation of $F$.

So far we obtain a simulation which uses space $n + 4$ (instead of $n + 3$ which is what we want). To reduce the space further on one unit, we do a preliminary normalization on the refutation of $F \restriction_{\{x=\top\}}$: we enforce that no download step increases the clause space above $n + 1$. If some does then it must be followed by an erasure step, because otherwise the clause step would go above $n+2$ which is contrary to our inductive hypothesis. In this case we first do the erasure step and then the download step. In this way the space after the download step is at most $n + 1$. This does not hinder the soundness of the inference, does not increase space, and guarantees that the refutation has at most $n+1$ clauses in memory after any axiom downloads. Thus our simulation causes the space to go up to $n + 3$ at most.

The length of the total refutation is $4^n$ for the first part and $3 \cdot 4^n$ for the second part (each axiom download is simulated with three steps). In total the length is $4^{n+1}$. The negative width remains $k$. □

Our second main lemma says that if a $k$-CNF formula can be refuted in resolution in space $s$, negative width $w$, and length $l$ simultaneously then it can be refuted in the PC proof system in space $2^w \cdot (2s + 5)$

and in length $4nl$. We will actually prove a slightly more general result that will also be useful to us later. For this generalized result we need the following definition of *total negative space*.

**Definition 3.5 (Total negative space).** For a clause set $\mathbb{C}$ we define its *total negative space* $\mathrm{TotSp}^-(\mathbb{C})$ as the total number of negative literals in clauses in $\mathbb{C}$ counted with repetitions. For a refutation $\pi$ we define its total negative space $\mathrm{TotSp}^-(\pi)$ as the maximum total negative space of a configuration in it. The total negative space $\mathrm{TotSp}^-(\mathrm{F} \vdash \bot)$ of a CNF formula $F$ is the minimum total negative space of a resolution refutation for $F$.

**Lemma 3.6.** *Let $F$ be an unsatisfiable CNF formula, and suppose that $F$ can be refuted in resolution simultaneously in space $s$, in negative width $w$, in length $l$, and in total negative space $N$. Then $F$ can be refuted in PC in length $4nl$ and in space*

$$\min\{2^w \cdot (2s+5), 2^w + 2^{3N+2} + s + 2\}.$$

*Proof.* Let $\pi$ be a resolution refutation of $F$ in space at most $s$, negative width at most $w$, length $l$ and total negative space at most $N$. We claim that $\pi$ can be transformed into a resolution refutation $\tilde{\pi}$ of $F$ of space at most $s+2$, negative width at most $w+1$, length at most $2nl$, and total negative space at most $3N+2$, where in each application in $\tilde{\pi}$ of the resolution rule on clauses $A \vee x$ and $B \vee \neg x$ we have that $A = B$. To see this suppose that $\pi$ applies the resolution rule on the clauses $A \vee x$ and $B \vee \neg x$, where $A = a_1 \vee \ldots \vee a_r$ and $B = b_1 \vee \ldots \vee b_\ell$. Then the clause $A \vee B$ can be derived from the clauses $A \vee x$ and $B \vee \neg x$ by obtaining first the pair of clauses $A \vee B \vee x$ and $A \vee B \vee \neg x$ from the clauses $A \vee x$ and $B \vee \neg x$ respectively via weakening and then resolving $A \vee B \vee x$ and $A \vee B \vee \neg x$ on the variable $x$. A space efficient derivation of this form is shown below.

$$\begin{pmatrix} A \vee x \\ B \vee \neg x \end{pmatrix} \rightarrow \begin{pmatrix} A \vee x \\ B \vee \neg x \\ A \vee x \vee b_1 \end{pmatrix} \rightarrow \begin{pmatrix} A \vee x \\ B \vee \neg x \\ A \vee x \vee b_1 \\ A \vee x \vee b_1 \vee b_2 \end{pmatrix}$$

$$\rightarrow \begin{pmatrix} A \vee x \\ B \vee \neg x \\ A \vee x \vee b_1 \vee b_2 \end{pmatrix} \rightarrow \ldots \rightarrow \begin{pmatrix} A \vee x \\ B \vee \neg x \\ A \vee b_1 \vee b_2 \vee \ldots \vee b_\ell \vee x \end{pmatrix}$$

$$\rightarrow \begin{pmatrix} A \vee x \\ B \vee \neg x \\ A \vee B \vee x \\ B \vee \neg x \vee a_1 \end{pmatrix} \rightarrow \begin{pmatrix} A \vee x \\ B \vee \neg x \\ A \vee B \vee x \\ B \vee \neg x \vee a_1 \\ B \vee \neg x \vee a_1 \vee a_2 \end{pmatrix}$$

$$\rightarrow \begin{pmatrix} A \vee x \\ B \vee \neg x \\ A \vee B \vee x \\ B \vee \neg x \vee a_1 \vee a_2 \end{pmatrix} \rightarrow \ldots \rightarrow \begin{pmatrix} A \vee x \\ B \vee \neg x \\ A \vee B \vee x \\ A \vee B \vee \neg x \end{pmatrix}$$

$$\rightarrow \begin{pmatrix} A \vee x \\ B \vee \neg x \\ A \vee B \vee x \\ A \vee B \vee \neg x \\ A \vee B \end{pmatrix} \rightarrow \begin{pmatrix} A \vee x \\ B \vee \neg x \\ A \vee B \vee x \\ A \vee B \end{pmatrix} \rightarrow \begin{pmatrix} A \vee x \\ B \vee \neg x \\ A \vee B \end{pmatrix}$$

Note that this transformation increases the space by at most 2, the negative width by at most 1, and the total negative space by at most $2N+2$. The length increases by a factor of $2n$ (recall that according to Definition 2.2 we are only counting axiom downloads and inferences). Let $\tilde{\pi} = \{\mathbb{C}_1, \ldots, \mathbb{C}_t\}$, and let $\pi'$ be the sequence of configurations $\{\mathbb{C}'_1, \ldots, \mathbb{C}'_t\}$ obtained from $\tilde{\pi}$ by replacing each clause in a configuration $\mathbb{C}_i \in \pi$ with its PC translation.

Since the negative width of every clause in a configuration $\mathbb{C}_i$ is at most $w+1$, its PC translation has at most $2^{w+1}$ monomials. Since each configuration $\mathbb{C}_i$ contains at most $s+2$ clauses this implies in turn that each configuration $\mathbb{C}'_i$ contains at most $2^{w+1}(s+2)$ monomials. But since we know that the total negative space is at most $3N+2$, another upper bound on the number of monomials in a configuration $\mathbb{C}'_i$ is $2^{3N+2}+s+2$. It remains to show that all transitions from a configuration $\mathbb{C}'_i$ to a configuration $\mathbb{C}'_{i+1}$ in $\pi'$ can be carried out in PC using inference rules with at most $2^w$ additional space (as opposed to considering just semantic deduction), while the length grows by a factor of at most 2.

Clearly, if $\mathbb{C}_{i+1}$ was obtained from $\mathbb{C}_i$ via axiom download or erasure then the transition from $\mathbb{C}'_i$ to $\mathbb{C}'_{i+1}$ can be carried out in syntactic PC employing the same rule. Suppose that $\mathbb{C}_{i+1}$ was obtained from $\mathbb{C}_i$ using the resolution rule applied to the clauses $A \vee x$ and $B \vee \neg x$. From our assumption on $\tilde{\pi}$ we have that $A = B$, and hence the PC translation of $A \vee B = A = B$ can be obtained from the PC translation of $A \vee x$ and $B \vee \neg x$ via summation.

Finally, suppose that $\mathbb{C}_{i+1}$ was obtained from $\mathbb{C}_i$ via weakening of a clause $A$ to the clause $A \vee a$. If $a$ is a positive literal then the PC translation of $A \vee a$ can be obtained from $A$ via the multiplication rule. Otherwise, suppose that $a$ is the negative literal $\neg x$ and let $P$, $(1-x)P$ be the PC translations of $A$, $A \vee \neg x$ respectively. Then $(1-x)P$ can be obtained from $P$ by first multiplying $P$ by $x$ and then subtracting the polynomial $xP$ from the polynomial $P$. Note that the latter derivation increases the number of monomials in a configuration by at most the number of monomials in $P$ which is at most $2^w$, and the length by a factor of at most 2.

Concluding, we have that the number of monomials in each configuration $\mathbb{C}'_i$ is at most

$$\min\{2^{w+1}(s+2), 2^{3N+2}+s+2\} \ , \tag{3.2}$$

and that for every $1 \leq i \leq t$ the transition from $\mathbb{C}'_i$ to $\mathbb{C}'_{i+1}$ can be carried out in syntactic PC using at most $2^w$ additional space, and increasing the length by factor of at most 2. Thus we have obtained a PC refutation of $F$ in length at most $4nl$ and space at most

$$2^w + \min\{2^{w+1}(s+2), 2^{3N+2}+s+2\} = \min\{2^w \cdot (2s+5), 2^w + 2^{3N+2}+s+2\} \tag{3.3}$$

as required. □

Theorem 3.1 now follows immediately from Lemmas 3.3 and 3.6.

*Proof of Theorem 3.1.* Lemma 3.3 implies that $F$ can be refuted in resolution in space $n+2$, negative width $k$, and length $4^n$ simultaneously. Lemma 3.6 then implies that $F$ can be refuted in polynomial calculus in space $2^k(2(n+2)+5) = 2^k(2n+9)$ and length $n4^{n+1}$. □

## 4 Space Complexity of 3-CNF PHP Formulas in PC

In this section, we prove an $\Omega(n)$ space lower bound on the 3-CNF extended version of the pigeonhole principle $PHP^m_n$. We also show that when $m = n+1$ this lower bound is tight up to a constant factor, by showing a matching upper bound. We start with the definitions of the pigeonhole principle and its extended version.

**Definition 4.1 (Pigeonhole principle formula).** The pigeonhole principle formula $PHP^m_n$ is an unsatisfiable CNF formula over the variables $\{p_{i,j} \mid i \in [m], j \in [n]\}$. We think of $[m]$ as the set of pigeons and of $[n]$ as the set of holes. The $PHP^m_n$ formula consists of the following clauses:

- $\bigvee_{j=1}^{n} p_{i,j}$ for each $i \in [m]$;
- $\overline{p}_{i_1,j} \vee \overline{p}_{i_2,j}$ for all distinct $i_1, i_2 \in [m]$ and for all $j \in [n]$.

The extended version $\widetilde{F}$ of a CNF formula $F$ is defined as follows.

**Definition 4.2 (Extended version).** We define the extended version of a clause $C = a_1 \vee a_2 \vee \ldots \vee a_n$, $n > 3$, as the following 3-CNF formula over $n+2$ clauses and $2n+1$ variables:

$$\{y_0\} \cup \{\overline{y}_{j-1} \vee a_j \vee y_j \mid 1 \leq j \leq n\} \cup \{\overline{y}_n\} \ .$$

If $C$ has at most 3 literals then the extended version of $C$ equals $C$. The variables $y_0, y_1, \ldots, y_n$ are called *extension variables*. The extended version $\widetilde{F}$ of a CNF formula $F$ is the union of all extended versions of clauses in $F$, where the extension variables used for each clause in $F$ are distinct.

Thus, the extended version $\widetilde{PHP}^m_n$ of the pigeonhole principle $PHP^m_n$ is the 3-CNF which consists of the following clauses:

- $y_{i,0},\ \overline{y}_{i,0} \vee p_{i,1} \vee y_{i,1},\ \overline{y}_{i,1} \vee p_{i,2} \vee y_{i,2},\ \ldots,\ \overline{y}_{i,n-1} \vee p_{i,n} \vee y_{i,n},\ \overline{y}_{i,n}$ for each $i \in [m]$;
- $\overline{p}_{i_1,j} \vee \overline{p}_{i_2,j}$ for all distinct $i_1, i_2 \in [m]$ and for all $j \in [n]$.

It can be verified that the extended version $\widetilde{F}$ of a CNF formula $F$ is unsatisfiable if and only if $F$ is unsatisfiable. In their paper [ABRW02], Alekhnovich et. al. proved a lower bound of $\Omega(n)$ on the space complexity of the pigeonhole principle $PHP^m_n$ in the PCR proof system.

**Theorem 4.3 ( [ABRW02]).** *For any $m > n$, it holds that $Sp_{\mathcal{PCR}}\big(PHP^m_n \vdash \bot\big) = \Omega(n)$.*

Since the PCR proof system is stronger than the PC proof system with respect to space, the above theorem holds for the PC proof system as well.

**Corollary 4.4.** *For any $m > n$, it holds that $Sp_{\mathcal{PC}}\big(PHP^m_n \vdash \bot\big) = \Omega(n)$.*

It is not difficult to see that in the PC proof system $Sp_{\mathcal{PC}}\big(\widetilde{F} \vdash \bot\big) \leq Sp_{\mathcal{PC}}\big(F \vdash \bot\big) + O(1)$ since every PC refutation of $F$ can be transformed into a PC refutation of $\widetilde{F}$ using only constant additional space. Our main theorem in this section says that for the pigeonhole principle $PHP^m_n$ the converse is also true, up to a constant factor.

**Theorem 4.5.** *For any $m > n$, it holds that $Sp_{\mathcal{PC}}\big(PHP^m_n \vdash \bot\big) \leq \frac{3}{2} Sp_{\mathcal{PC}}\big(\widetilde{PHP}^m_n \vdash \bot\big) + O(1)$.*

The combination of Corollary 4.4 and Theorem 4.5 yields the first non-constant space lower bound on a 3-CNF formula in the PC system.

**Corollary 4.6.** *For any $m > n$, it holds that $Sp_{\mathcal{PC}}\big(\widetilde{PHP}^m_n \vdash \bot\big) = \Omega(n)$.*

*Proof of Theorem 4.5.* We will show a way to transform any PC refutation of $\widetilde{PHP}^m_n$ into a PC refutation of $PHP^m_n$ without increasing the monomial space by more than a constant factor. Let $\pi = \{\mathbb{C}_1, \ldots, \mathbb{C}_t\}$ be a PC refutation of $\widetilde{PHP}^m_n$ in space $s$. Let $\pi' = \{\mathbb{C}'_1, \ldots, \mathbb{C}'_t\}$ be the sequence of configurations obtained from $\pi$ by substituting each extension variable $y_{i,j}$, $0 \leq j \leq n-1$, which appears in a polynomial in a configuration in $\pi$ with the product of variables $\prod_{t=j+1}^{n} p_{i,t}$, and substituting the variable $y_{i,n}$ with 1.

Since variables are substituted with products of variables, the number of monomials in each configuration in $\pi'$ is the same as the number of monomials in the corresponding configuration in $\pi$. It remains to show that all transitions from $\mathbb{C}'_i$ to $\mathbb{C}'_{i+1}$ can be carried out in syntactic PC without increasing the number of monomials by more than a constant factor. Clearly, if $\mathbb{C}_{i+1}$ was obtained from $\mathbb{C}_i$ via erasure, the addition rule or via multiplication by a variable which is not an extension variable, then $\mathbb{C}'_{i+1}$ can be obtained from $\mathbb{C}'_i$ by applying the same rule, and without increasing the space complexity. It remains to handle the cases in which $\mathbb{C}_{i+1}$ was obtained from $\mathbb{C}_i$ via multiplication by an extension variable or via axiom download.

If $\mathbb{C}_{i+1}$ was obtained from $\mathbb{C}_i$ via multiplication of a polynomial $Q$ by an extension variable $y_{i,j}$ then this can be simulated by a sequence of multiplications by the variables $p_{i,j+1}, p_{i,j+2}, \ldots, p_{i,n}$. For this, one additional space is needed to keep the intermediate polynomials of the form $\prod_{t=j+1}^{r} p_{i,t} \cdot Q$,

$j+1 \leq r < n$, in memory. Since the configuration $\mathbb{C}_{i+1}$ contains both polynomials $Q$ and $y_{i,j}Q$ and has space at most $s$, this implies that the space of the intermediate polynomial $\prod_{t=j+1}^{r} p_{i,t} \cdot Q$ is at most $s/2$. Hence the monomial space increases by a factor of at most $3/2$.

Finally, suppose that $\mathbb{C}_{i+1}$ was obtained from $\mathbb{C}_i$ via an axiom download. An axiom of the form $y_{i,0}$ is substituted with $\prod_{j\in[n]} p_{i,j}$, which is indeed a PC translation of an axiom in $PHP^m_n$. The axiom $\overline{y}_{i,n}$ is substituted with $0$, and thus can be eliminated from the refutation. A generic axiom $\overline{y}_{i,j-1} \vee p_{i,j} \vee y_{i,j}$ evaluates to $\left(1 - \prod_{t=j}^{n} p_{i,t}\right) p_{i,j} \prod_{t=j+1}^{n} p_{i,t} = \prod_{t=j}^{n} p_{i,t} - \prod_{t=j}^{n} p_{i,t}^2$. The latter polynomial can be derived in constant monomial space from the axioms $p_{i,j}^2 - p_{i,j}$. A similar procedure can be employed to infer the evaluation of any axiom $y_{i,j}^2 - y_{i,j}$. □

A natural question is whether the lower bound given in Corollary 4.6 is tight. We show that when $m = n+1$ this is indeed the case (up to a constant factor). Note that an application of Theorem 3.1 will only give an upper bound of $O(n^2)$ on the PC space of $\widetilde{PHP}^{n+1}_n$ since the number of variables in $\widetilde{PHP}^{n+1}_n$ is $\Theta(n^2)$. Instead we shall use the fact that $Sp_{\mathcal{PC}}(\widetilde{F} \vdash \bot) \leq Sp_{\mathcal{PC}}(F \vdash \bot) + O(1)$ for every CNF formula $F$, and thus in order to prove the upper bound on $\widetilde{PHP}^{n+1}_n$ it suffices to prove the following.

**Lemma 4.7.** *For any $m > n$, it holds that $Sp_{\mathcal{PC}}(PHP^m_n \vdash \bot) \leq n + O(1)$.*

Putting together Corollary 4.6 and Lemma 4.7 we obtain Theorem 1.2. For the proof of Lemma 4.7 we shall need the following claim, which will also be useful for us later when proving Theorem 7.4.

**Claim 4.8.** For $a > 0$, $b > 0$ and clauses $X, Y$ let $F$ be the CNF formula

$$\left(X \vee \bigvee_{i=1}^{a} x_i\right) \wedge \left(Y \vee \bigvee_{j=1}^{b} y_j\right) \wedge \left(\bigwedge_{i=1}^{a}\bigwedge_{j=1}^{b} (\overline{x}_i \vee \overline{y}_j)\right).$$

Then $X \vee Y$ can be derived from $F$ in resolution in clause space 4. Furthermore, if $x_i$ and $y_j$ are all positive literals and $X, Y$ contain only positive literals then $X \vee Y$ can be derived from $F$ in resolution in clause space 4 and total negative space 4 (as per Definition 3.5) simultaneously.

*Proof.* Without loss of generality we may assume that $X, Y$ are empty clauses, thus we aim to derive the empty clause. The general case follows by weakening. For any fixed $i$ we derive $\overline{x}_i$ in the following way: resolve $\bigvee_{j=1}^{b} y_j$ with the axioms $\overline{x}_i \vee \overline{y}_1, \ldots, \overline{x}_i \vee \overline{y}_b$ one by one, and after every step erase from memory both premises. In this way we can derive any unit clause $\overline{x}_i$ in clause space 3. Thus we can reach the empty clause by resolving all such unit clauses with $\bigvee_{t=1}^{a} x_t$. The whole inference can be carried in clause space 4. Notice that assuming positivity of $X$, $Y$ and the literals $x_i$ and $y_j$, at most 4 negative literals are simultaneously in memory. □

We now proceed to the proof of Lemma 4.7.

*Proof of Lemma 4.7.* We focus on the case $m = n+1$. Observe that when $m > n$ all clauses of $PHP^{n+1}_n$ are also clauses of $PHP^m_n$, so that is without loss of generality.

For the proof we use a resolution refutation of $PHP^{n+1}_n$ from [BP98, Lemma 1]. Based on this refutation, we show that $PHP^{n+1}_n$ can be refuted in resolution in clause space $n + O(1)$ and in total negative space $O(1)$ simultaneously. Once we have such an efficient resolution refutation we apply Lemma 3.6 to get a PC-refutation which meets the claimed bound.

The refutation of $PHP^{n+1}_n$ is best explained in an inference system specialized for such a formula. We will later see how to simulate this refutation efficiently in the resolution system. For any sets $D \subseteq [n+1]$ and $R \subseteq [n]$ we denote by $(D \to R)$ the positive clause $\bigvee_{i\in D}\bigvee_{j\in R} p_{i,j}$. In our new inference system, denoted by $\mathcal{P}$, the axioms will be all pigeon axioms $(\{i\} \to [n])$. For subsets $A, B \subseteq [n+1]$ and $j \in [n]$ the inference rule is defined as follows:

$$\frac{(A \to \{j\}) \qquad (B \to \{j\})}{(A \cap B \to \{j\})} \tag{4.1}$$

We show by induction on $|D|$ that for any non-empty $D \subseteq [n+1]$ the clause $(D \to [n+1-|D|])$ can be derived in $\mathcal{P}$ in clause space at most $|D|+2$. This will imply that the empty clause $([n+1] \to \emptyset)$ is derivable in $\mathcal{P}$ in space at most $n+3$.

For $|D| = 1$ this is immediate since the clause is an axiom. For $|D| > 1$, fix $D := \{a_1, a_2, \ldots a_t\}$, fix $j := n+1-|D|+1$, and fix $\mathcal{F} := (D \to [n+1-|D|])$. Note that by the inductive hypothesis, each formula $\mathcal{F} \vee (D \setminus \{a_i\} \to \{j\})$ is a weakening of a formula which can be derived in clause space $|D|+1$. Hence, $\mathcal{F}$ can be derived in space $|D|+2$ as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\mathcal{F} \vee (D \setminus \{a_1\} \to \{j\}) \qquad \mathcal{F} \vee (D \setminus \{a_2\} \to \{j\})}{\mathcal{F} \vee (D \setminus \{a_1, a_2\} \to \{j\})} \qquad \mathcal{F} \vee (D \setminus \{a_3\} \to \{j\})}{\vdots}}{\mathcal{F} \vee (D \setminus \{a_1, a_2, \ldots, a_{t-1}\} \to \{j\})} \qquad \mathcal{F} \vee (D \setminus \{a_t\} \to \{j\})}{\mathcal{F} \vee (\emptyset \to \{j\})}$$

Next we show how to simulate the above refutation in resolution. Note that all clauses in the above refutation are valid clauses in resolution and contain only positive literals, so we just need to show how to simulate the inference rule (4.1) without increasing the clause space and the total negative space by more than a constant. Letting $X = Y = (A \cap B \to \{j\})$, and recalling that we have an axiom $\overline{p}_{i,j} \vee \overline{p}_{i',j}$ for any $i \neq i' \in [n+1]$, $j \in [n]$, Claim 4.8 implies that the inference rule (4.1) can be simulated in resolution by increasing both the clause space and the total negative space by at most a constant.

Concluding, we have that the $PHP_n^{n+1}$ formulas can be refuted in clause space $n + O(1)$ and in $O(1)$ total negative space simultaneously. Lemma 3.6 then implies that $PHP_n^{n+1}$ can be refuted in monomial space $n + O(1)$. As we already observed, this also holds for any $PHP_n^m$ with $m > n$. □

## 5 A PCR Space Lower Bound for Bit-Graph PHP Formulas

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 [ABRW02], but measured in the initial width of the clauses it is an exponential improvement. In Section 6, we will improve this further to a qualitatively similar bound for CNF formulas of bounded width, resolving an open problem in [ABRW02]. 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 formulas in Section 6 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. In what follows, we write $[i, j]$ to denote the set $\{i, i+1, \ldots, j\}$, and $[i, j)$ to denote $\{i, i+1, \ldots, j-1\}$.

**Definition 5.1 (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*

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

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 the notational convention adopted in the preliminaries 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 $BPHP^m_n$ using a similar construction to that in Alekhnovich et al. [ABRW02]. As in [ABRW02], our proof also applies to the much stronger functional calculus proof system (see Definition 2.11). Notice that we can identify total truth value assignments $\alpha$ to the variables of $BPHP^m_n$ 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 5.2 (Well-behaved assignment).** Let $\alpha$ be a total assignment to the variables of $BPHP^m_n$ and let $S \subseteq [0, m)$ be a set of pigeons. We say that $\alpha$ is *well-behaved on $S$* if the holes assigned by $\alpha$ to the pigeons in $S$ are all distinct.

**Definition 5.3 (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 $\alpha$ is *well-behaved on and satisfies* a commitment set $\mathbb{A}$ if $\alpha$ 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 5.4.** *Suppose $S$ is any set of fewer than $n/2$ pigeons, $\alpha$ 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 $\alpha$ by reassigning $p$ in such a way that the new assignment is well-behaved on $S \cup \{p\}$ and satisfies the literal $x[p, i]^b$.*

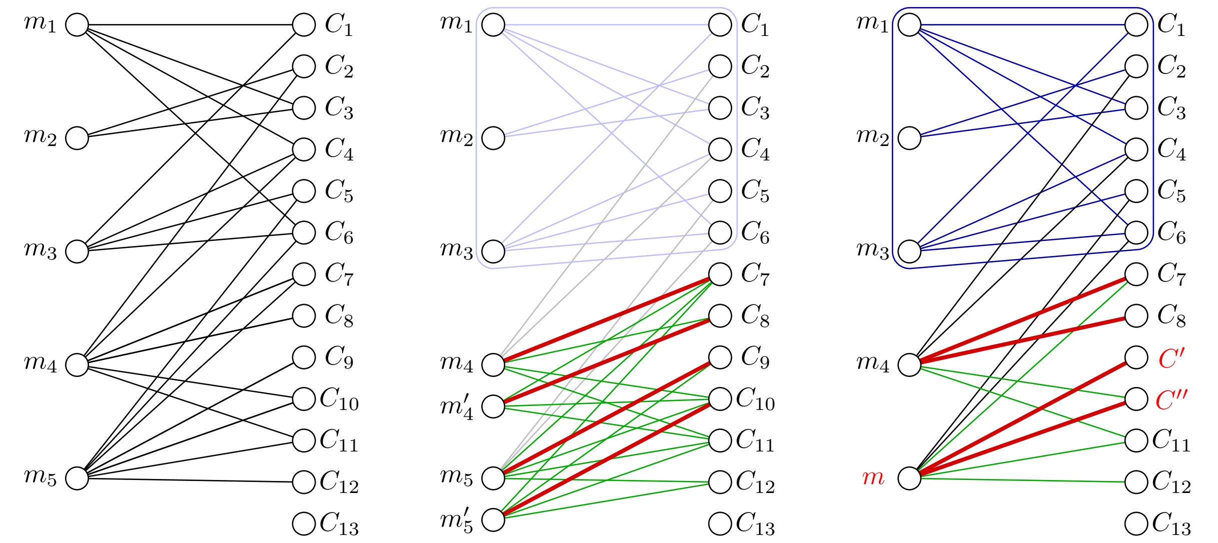
*Proof.* There are exactly $n/2$ holes for pigeon $p$ that will satisfy the literal $x[p, i]^b$ if $p$ is sent there. Fewer than $n/2$ holes are taken by the pigeons in $S$ so there is a hole $h$, not assigned to any pigeon in $S$, whose assignment to $p$ will satisfy $x[p, i]^b$. □

**Corollary 5.5.** *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 5.4. □

**Definition 5.6 (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 $\alpha$ 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 [ABRW02], 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 5.5 to show that all the configurations are satisfiable, giving a contradiction.



(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 $\Gamma$ and matchings constructed at end of argument.

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

**Lemma 5.7 (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 $\Gamma$ 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 $\Gamma$ 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 $\Gamma$ 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. (It can be the case that $m$ also mentions $q'$ or $q''$ or both, but by construction we are guaranteed that $m$ mentions at least one pigeon in each commitment and this is all we will need here.) 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 $m \in M \setminus \Gamma$, and let our new commitment set be $\mathbb{B} = N(\Gamma) \cup \{C_m \mid m \in M \setminus \Gamma\}$, 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 $|\mathbb{B}| \leq 2 \cdot |\Gamma| + |M \setminus \Gamma| \leq 2 \cdot |M| \leq 2 \cdot Sp(\mathbb{P})$ (taking a look at Figure 1(c) might be helpful in verifying this). However, we also need to show that $\mathbb{B}$ entails $\mathbb{P}$ over well-behaved assignments. That is, we must prove that every $\beta$ that is well-behaved on and satisfies $\mathbb{B}$ also satisfies $\mathbb{P}$. Note that this is a priori not clear. We know that this holds for $\mathbb{A}$ by assumption, but $\operatorname{dom}\mathbb{A}$ is potentially much larger than $\operatorname{dom}\mathbb{B}$ and so $\mathbb{A}$ only has to deal with much more well-behaved assignments. Also, and more seriously, the commitments in $\mathbb{B}$ are not a subset of those in $\mathbb{A}$, and on the contrary might be in conflict with $\mathbb{A}$ in the sense that satisfying literals in $\mathbb{B}$ falsifies literals in $\mathbb{A}$.

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

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

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

To this end, let $S$ be the set of pigeons in $\operatorname{dom}\mathbb{B}$, and let $T$ be the set of pigeons in $\operatorname{dom}\mathbb{A} \setminus \operatorname{dom}\mathbb{B}$ (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 5.5 to $S$, $T$, and $\beta$ to get a truth value assignment $\alpha$ that is well-behaved on $S \cup T$, agrees with $\beta$ 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 $\Gamma$ mentions pigeons in $T$ (by construction), so $\alpha$ and $\beta$ agree on monomials in $\Gamma$. For $m \in M \setminus \Gamma$, all $\beta$ 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 $m \in M \setminus \Gamma$, the assignment $\alpha$ gives the same value to $m$ as does $\beta$, namely 0. Hence $\alpha$ and $\beta$ agree on all monomials in $M$ and $\mathbb{P}(\alpha) = \mathbb{P}(\beta)$. This takes care of item 1 above. By Corollary 5.5, $\alpha$ is well-behaved on $S \cup T = \operatorname{dom}\mathbb{A}$. Also, since $\alpha$ satisfies all literals in $X$ as well as $N(\Gamma)$, consequently $\alpha$ satisfies $\mathbb{A}$. This takes care of item 2, and as already discussed it now follows that $\mathbb{P}(\alpha) = 0$. Thus, every $\beta$ 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 5.8 (Detailed version of Theorem 1.3).** $Sp_{\mathcal{PCR}}(BPHP^m_n \vdash \bot) > n/8$.

*Proof.* Let $\pi = \{\mathbb{P}_0, \ldots, \mathbb{P}_N\}$ be a PCR-refutation of $BPHP^m_n$ 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 5.5 (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 $\mathbb{P}_t$.

**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 $\alpha$ be any assignment well-behaved on and satisfying $\mathbb{A}_{t+1}$. Then $\alpha$ 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 $\alpha$ is an assignment which is well-behaved on and satisfies $\mathbb{A}_{t+1}$. The induction hypothesis implies that $\alpha$ satisfies $\mathbb{P}_t$. Since $P$ semantically follows from $\mathbb{P}_{t+1}$, $\alpha$ 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 5.7 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). Recall that in FC, polynomials are replaced by arbitrary Boolean functions (more accurately, Boolean-valued functions of Boolean variables). The space of a configuration $\mathbb{P}$ is the minimal number $s$ such that for some $s$ clauses $C_1, \ldots, C_s$ (which we consider as Boolean functions), any function in $\mathbb{P}$ can be written as a function $g(C_1, \ldots, C_s)$ of the clauses. Such a minimal set of clauses is known as a *defining set of clauses*. The space of a functional calculus refutation is the maximal space of any configuration encountered during the refutation.

Here is a simple example to illustrate the definition. Consider the Boolean functions $f_1 = x \vee y \vee z$ and $f_2 = \overline{x} \wedge \overline{y}$ and let $\mathbb{P}$ be the configuration $\{f_1, f_2\}$. Both functions $f_1, f_2$ can be written as functions of the two monomials $C_1 = x \vee y$ and $C_2 = z$, in the following way: $f_1 = C_1 \vee C_2$, $f_2 = \neg C_1$. Since no single monomial suffices for this purpose, the space of $\mathbb{P}$ is exactly 2.

The proof of Lemma 5.7 applies equally well under the functional calculus definition of space. In the first step of the proof, we construct a bipartite graph between the set of monomials $M$ appearing in the given configuration $\mathbb{P}$ and the set of commitments $\mathbb{A}$. For PCR, $M$ is simply the set of monomials appearing in $\mathbb{P}$. For the functional calculus, we use a defining set of clauses for $\mathbb{P}$. The rest of the proof carries over without changes.

In more detail, the proof constructs a new, smaller set of commitments $\mathbb{B}$. Given any assignment $\beta$ that is well-defined on and satisfies $\mathbb{B}$, the proof constructs a new assignment $\alpha$ that is well-defined on and satisfies $\mathbb{A}$, with the additional property that all monomials in $M$ have the same value in both $\alpha$ and $\beta$. Since $\mathbb{A}$ entails $\mathbb{P}$ over well-behaved assignments, $\mathbb{P}(\alpha) = 0$. As all monomials retain their values in $\beta$, also $\mathbb{P}(\beta) = 0$. The conclusion is that $\mathbb{B}$ also entails $\mathbb{P}$ over well-behaved assignments.

The proof of Theorem 5.8 applies with only one small modification. Given a small-space functional calculus refutation $\mathbb{P}_0, \ldots, \mathbb{P}_N$ of $\mathit{BPHP}^m_n$, the proof constructs a matching sequence $\mathbb{A}_0, \ldots, \mathbb{A}_N$ of commitment sets such that $\mathbb{A}_i$ entails $\mathbb{P}_i$ over well-behaved assignments, and furthermore $|\mathbb{A}_i| \leq 2 \cdot \mathit{Sp}(\mathbb{P}_i)$. Corollary 5.5 then implies that $\mathbb{P}_N$ is satisfiable, contrary to the assumption that $1 \in \mathbb{P}_N$.

The commitment sets $\mathbb{A}_0, \ldots, \mathbb{A}_N$ are constructed inductively, starting with the empty commitment set for $\mathbb{A}_0$. An inference step requires no changes to the commitment set, and erasures are handled directly by Lemma 5.7. When an axiom $H(p_1, p_2, h)$ is downloaded during step $t$, there are two cases. If both pigeons mentioned in $H(p_1, p_2, h)$ are already in the commitment set $\mathbb{A}_t$, no changes are needed. Otherwise, either one or two new commitments are added to $\mathbb{A}_{t+1}$. This could invalidate the invariant $|\mathbb{A}_{t+1}| \leq 2 \cdot \mathit{Sp}(\mathbb{P}_{t+1})$, since it might be the case that $\mathit{Sp}(\mathbb{P}_{t+1}) = \mathit{Sp}(\mathbb{P}_t)$ (this cannot occur for PCR under our definition of space, although it could occur under the laxer definition of [ABRW02] that only counts distinct monomials). An application of Lemma 5.7 mitigates the issue.

# 6 A PCR Space Lower Bound for XOR-PHP Formulas

Let us now apply the machinery developed in Section 5 to a different encoding of the pigeonhole principle: a slightly obfuscated version using exclusive or and "hole indicators" to specify which pigeons are placed in which holes. In this way, we can obtain strong PCR space lower bounds for CNF formulas of constant width.

**Definition 6.1 (XOR PHP formula).** The *XOR pigeonhole principle formula* $\mathit{XPHP}^m_n$ has propositional variables $x[i,j]$ for each $i \in [0,m)$ and $j \in [0,n]$. (Recall that $[0,m) = \{0, \ldots, m-1\}$ and $[0,n] =$

$\{0,\dots,n\}$.) We think of $[0,m)$ as a set of pigeons and $[0,n]$ as a set of hole indicators. Each pigeon $i$ gives a 0 or 1 value to every hole indicator $j$, recorded in the variable $x[i,j]$.

The hole indicators indicate assignments of pigeons to holes indirectly: a pigeon $i \in [0,m)$ is assigned to a hole $j \in [0,n)$ when $x[i,j] \not\equiv x[i,j+1]$ is true, that is when $x[i,j]$ and $x[i,j+1]$ have different truth values. This assignment need not be unique: the formula will only ensure that each pigeon is assigned to an *odd* number of holes.

The formula $\mathit{XPHP}^m_n$ asserts the following:

1. Every pigeon gives different values to the first and last hole indicators. That is, for each $i \in [0,m)$, $x[i,0] \not\equiv x[i,n]$.

2. At most one pigeon is assigned to any given hole. That is, for all distinct $i,i' \in [0,m)$ and all $j \in [0,n)$, $(x[i,j] \equiv x[i,j+1]) \vee (x[i',j] \equiv x[i',j+1])$.

Formally, we think of this as a 4-CNF formula including the clauses

$$\left.\begin{array}{l} x[i,0] \vee x[i,n] \\ \overline{x[i,0]} \vee \overline{x[i,n]} \end{array}\right\} \quad \text{for each } i \in [0,m),$$

$$\left.\begin{array}{l} x[i,j] \vee \overline{x[i,j+1]} \vee x[i',j] \vee \overline{x[i',j+1]} \\ \overline{x[i,j]} \vee x[i,j+1] \vee x[i',j] \vee \overline{x[i',j+1]} \\ x[i,j] \vee \overline{x[i,j+1]} \vee \overline{x[i',j]} \vee x[i',j+1] \\ \overline{x[i,j]} \vee x[i,j+1] \vee \overline{x[i',j]} \vee x[i',j+1] \end{array}\right\} \quad \text{for all } j \in [0,n),\ i,i' \in [0,m),\ i \neq i'.$$

Written in this way, condition 1 translates into $2m$ clauses of width 2 (*pigeon axioms*) and condition 2 into $4\binom{m}{2}n$ clauses of width 4 (*hole axioms*).

For $m > n$, $\mathit{XPHP}^m_n$ is a contradiction. To see this notice that, by condition 1, for each pigeon $i \in [0,m)$ there must be at least one hole $j \in [0,n)$ for which $i$ gives different values to indicators $j$ and $j+1$; say that such a $j$ is *assigned* to $i$. Since $n < m$, by the pigeonhole principle there must be some pair of distinct pigeons which are assigned the same hole. But this contradicts condition 2.

Notice that for any pigeon $i$ and any hole indicator $j$, it is possible to fix the value of $x[i,j]$ to 0 or 1 without putting any constraint on the holes which are assigned to $i$.

Fix $n \geq 1$. We will prove a PCR space lower bound for $\mathit{XPHP}^m_n$ by essentially the same argument as in Section 5, with the difference that it is a little easier for us to satisfy Lemma 6.4, which leads to a space lower bound of $n/4$ rather than $n/8$.

**Definition 6.2 (Well-behaved assignment).** Let $\alpha$ be an assignment to all the variables of $\mathit{XPHP}^m_n$ and let $S \subseteq [0,m)$ be a set of pigeons. We say that $\alpha$ is *well-behaved on* $S$ if two things hold. Firstly, for each pigeon $i \in S$, $\alpha$ assigns exactly one hole $j$ to $i$ by either

1. setting $x[i,j']$ to 0 for all $j'$ in $[0,j]$ and to 1 for all $j'$ in $[j+1,n]$, or

2. setting $x[i,j']$ to 1 for all $j'$ in $[0,j]$ and to 0 for all $j'$ in $[j+1,n]$.

Secondly, the holes assigned by $\alpha$ to the pigeons in $S$ are all distinct.

**Definition 6.3 (Commitment).** As before, a *commitment* is the disjunction of two literals. A *commitment set* is a set of commitments in which no pigeon appears twice.

**Lemma 6.4.** *Suppose $S$ is any set of fewer than $n$ pigeons, $\alpha$ is an assignment well-behaved on $S$, and $x[i,j]^b$ is a literal associated with a pigeon $i \notin S$. Then we can modify $\alpha$ by reassigning the pigeon $i$ in such a way that the new assignment is well-behaved on $S \cup \{i\}$ and satisfies the literal $x[i,j]^b$.*

*Proof.* Choose a hole $k$ which is not already occupied by any pigeon in $S$. Then assign hole $k$ to pigeon $i$ using an assignment either of type 1 or of type 2 from Definition 6.2. One of these assignments will satisfy $x[i, j]^b$. □

**Corollary 6.5.** *Let $S, T$ be two disjoint sets of pigeons such that $|S \cup T| \leq n$, and let $X$ be a set containing one literal associated with pigeon $i$ for each $i \in T$. 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 the lemma. □

**Definition 6.6 (Entailment).** Given a commitment set $\mathbb{A}$ and a configuration $\mathbb{P}$, we say that $\mathbb{A}$ *entails* $\mathbb{P}$ *over well-behaved assignments* if, for every assignment $\alpha$ which is well-behaved on $\operatorname{dom} \mathbb{A}$, if $\alpha$ satisfies $\mathbb{A}$ then $\alpha$ also satisfies $\mathbb{P}$.

**Lemma 6.7 (Locality lemma).** *Let $\mathbb{A}$ be a commitment set and $\mathbb{P}$ be a configuration such that $\mathbb{A}$ entails $\mathbb{P}$ over well-behaved assignments and $|\mathbb{A}| \leq n/2$. 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.* The proof of this lemma is virtually the same as the proof of Lemma 5.7.

Consider the bipartite graph with on one side the set $M$ of all monomials in $\mathbb{P}$, and on the other side the set of all commitments in $\mathbb{A}$. We draw an edge between a monomial and a commitment if there is a pigeon mentioned in both. Let $\Gamma$ be a maximal set of monomials such that $|N(\Gamma)| \leq 2|\Gamma|$. By Hall's theorem, there exist two injective functions $f_1, f_2 \colon M \setminus \Gamma \mapsto \mathbb{P} \setminus N(\Gamma)$ with disjoint ranges.

The new commitment set $\mathbb{B}$ consists of $N(\Gamma)$ together with one additional commitment $C_m$ for each monomial $m \in M \setminus \Gamma$. It immediately follows that $|\mathbb{B}| \leq 2Sp(\mathbb{P})$. To define $C_m$, consider the two commitments $C_1, C_2$ matched to $m$. The commitment $C_1$ mentions some pigeon $i_1$ which has an associated literal $x[i_1, j_1]^{b_1}$ appearing in $m$. Similarly, $C_2$ mentions some pigeon $i_2$ which has an associated literal $x[i_2, j_2]^{b_2}$ appearing in $m$. We define $C_m = x[i_1, j_1]^{b_1} \vee x[i_2, j_2]^{b_2}$. As a result, any assignment satisfying $C_m$ will zero the monomial $m$.

We proceed to show that $\mathbb{B}$ entails $\mathbb{P}$ over well-behaved assignments. Let $\alpha$ be an assignment which is well-behaved on and satisfies $\mathbb{B}$. We will define a new assignment $\beta$ such that each monomial in $M$ gets the same value in both $\alpha$ and $\beta$, but also such that $\beta$ is well-behaved on and satisfies $\mathbb{A}$, implying that $\mathbb{P}(\beta) = 0$ and hence that $\mathbb{P}(\alpha) = 0$.

Let $S$ be the set of pigeons appearing in $\operatorname{dom} \mathbb{B}$, and let $T$ be the set of pigeons appearing in $\operatorname{dom} \mathbb{A} \setminus \operatorname{dom} \mathbb{B}$. With a view to applying Corollary 6.5, let $X$ be the set of literals that for each pigeon $i \in T$ includes the (unique) literal $x[i, j]^b$ appearing in $\mathbb{A}$.

Since $|\mathbb{A}| \leq n/2$, we know that $|S \cup T| \leq n$, and so we can apply Corollary 6.5 to the assignment $\alpha$ to get an assignment $\beta$ which is well-behaved on $S \cup T$, is identical to $\alpha$ on pigeons outside $T$, and satisfies $X$. As $\alpha$ and $\beta$ differ only on $T$, all monomials in $\Gamma$ get the same value in both assignments. Every other monomial in $M$ is zeroed in both assignments. Since $\alpha$ and $\beta$ are identical on $S$, $\beta$ satisfies all commitments $N(\Gamma)$. The choice of $X$ guarantees that it satisfies all other commitments in $\mathbb{A}$. □

**Theorem 6.8 (Detailed version of Theorem 1.4).** $Sp_{\mathcal{PCR}}(XPHP^m_n \vdash \perp) > n/4$.

*Proof.* The proof of this theorem is virtually identical to that of Theorem 5.8.

Thus, in order to derive a contradiction, suppose that $\pi = \{\mathbb{P}_0, \ldots, \mathbb{P}_N\}$ is a PCR-refutation of $XPHP^m_n$ in space at most $n/4$, with $\mathbb{P}_0 = \emptyset$ and $1 \in \mathbb{P}_N$. We will construct inductively a sequence of commitment sets $\mathbb{A}_0, \ldots, \mathbb{A}_N$ such that at each time $t$ it holds that $|\mathbb{A}_t| \leq 2|\mathbb{P}_t|$ and $\mathbb{A}_t$ entails $\mathbb{P}_t$ over well-behaved assignments. In particular, by Corollary 6.5 this will imply that every configuration $\mathbb{P}_t$ is satisfiable, which then leads to the desired contradiction for $\mathbb{P}_N$.

We define $\mathbb{A}_0$ to be the empty commitment. Suppose now that 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 $\mathbb{P}_t$.

**Axiom download.** There are three kinds of axioms: logical axioms, pigeon axioms and hole axioms. Logical axioms are handled in the same way as semantic inferences.

Suppose that $\mathbb{P}_{t+1}$ is $\mathbb{P}_t$ together with some axiom $D$ which is either a pigeon axiom or a hole axiom. Suppose first that the pigeons mentioned in $D$ (one pigeon in the case of a pigeon axiom, two pigeons in the case of a hole axiom) are already in $\mathbb{A}_t$. Then we put $\mathbb{A}_{t+1} = \mathbb{A}_t$. Let $\alpha$ be any assignment well-behaved on and satisfying $\mathbb{A}_{t+1}$. Then $\alpha$ satisfies $\mathbb{P}_t$ by the inductive hypothesis, and must also satisfy $D$ since it is well-behaved on the pigeons in $D$.

Otherwise, there are either one or two pigeons mentioned in $D$ which are not in $\operatorname{dom} \mathbb{A}_t$. For each such pigeon $i$ we add a "dummy" commitment $C_i$ to $\mathbb{A}_t$ whose sole purpose is to put $i$ into the domain of $\mathbb{A}_{t+1}$. We can take $C_i$ to be $x[i, 0] \vee x[i', 0]$, where $i'$ is any unused so far pigeon. We add at most two new commitments, and so $|\mathbb{A}_{t+1}| \leq 2Sp(\mathbb{P}_{t+1})$.

**Inference.** Suppose $\mathbb{P}_{t+1} = \mathbb{P}_t \cup \{P\}$, where $P$ semantically follows from $\mathbb{P}_t$. We put $\mathbb{A}_{t+1} = \mathbb{A}_t$. Any assignment $\alpha$ which is well-behaved on and satisfies $\mathbb{A}_{t+1}$ satisfies $\mathbb{P}_t$ by the induction hypothesis. Since $P$ semantically follows from $\mathbb{P}_t$, $\alpha$ also satisfies $P$.

**Erasure.** Suppose $\mathbb{P}_{t+1} \subset \mathbb{P}_t$. Since $|\mathbb{A}_t| \leq 2Sp(\mathbb{P}_t) \leq n/2$, Lemma 6.7 applies and furnishes us with a commitment set $\mathbb{A}_{t+1}$ such that $|\mathbb{A}_{t+1}| \leq 2Sp(\mathbb{P}_{t+1})$ and $\mathbb{A}_{t+1}$ entails $\mathbb{P}_{t+1}$ over well-behaved assignments. □

The lower bound also applies to the functional calculus, using the same arguments as in Section 5.

# 7 Space Complexity of 3-CNF Versions of Wide CNF Formulas

Although this paper reports progress on understanding space complexity for polynomial calculus, and in particular establishes the first nontrivial space lower bounds for $k$-CNF formulas with $k$ constant, space is still not a very well-understood measure. This is especially true for algebraic proof systems like PCR, where it is not clear how much useful information can be encoded in sparse polynomial equations.

Perhaps one of the most annoying open questions is that although we obtain space lower bounds for $k$-CNF formulas in this paper, we can do so only for $k \geq 4$. Thus, it is still consistent with state of the art that all 3-CNF formulas would be easy for PCR with respect to space, although this appears absurd. In this context, one especially simple and intriguing open problem is the following. Suppose that we have a CNF formula $F$ with wide clauses, and that we convert it to its canonical equivalent 3-CNF version $\widetilde{F}$ as described in Definition 4.2. What is the space complexity of $\widetilde{F}$ compared to that of $F$? Clearly, the space needed to refute the 3-CNF version cannot increase by more than a small additive constant. But are there formulas for which the conversion to 3-CNF can *decrease* the space complexity substantially? If the answer would be no, then one way of getting strong space lower bounds for 3-CNF formulas would be to take a formula with wide clauses where we know that the space complexity is large and consider its 3-CNF extended version.

Although the question how the space complexities of $F$ and $\widetilde{F}$ are related remains open in the general case, we are able to show for a particular class of CNF formulas which we call *weight-constrained* that the space complexity of their extended versions stays essentially the same (up to a constant factor) in both resolution and PCR. One interesting CNF formula belonging to this class is the functional pigeonhole principle $FPHP^m_n$ (the definition of which is given below for completeness). Thus, our result implies that in order to prove space lower bounds for the extended version of the functional pigeonhole principle in PCR it suffices to prove space lower bounds for the standard version with wide clauses. We start with the definition of a weight-constrained formula.

**Definition 7.1 (Weight-constrained formula).** Let $F$ be a CNF formula. We say that $F$ is a *weight-constrained formula* if for every clause $l_1 \vee l_2 \vee \ldots \vee l_m$ with $m \geq 4$ contained in $F$, the formula $F$ also contains clauses $\bar{l}_i \vee \bar{l}_j$ for all $1 \leq i < j \leq m$.

Thus, a weight-constrained CNF $F$ encodes explicitly that exactly one literal in each of its wide clauses must be true. One natural weight-constrained CNF formula is the functional pigeonhole principle

$FPHP^m_n$, which is similar to the regular pigeonhole principle $FPHP^m_n$ but with the additional constraint that a single pigeon cannot occupy more than one hole.

**Definition 7.2 (Functional pigeonhole principle).** The functional pigeonhole principle $FPHP^m_n$ is an unsatisfiable CNF formula over the variables $\{p_{i,j} \mid i \in [m], j \in [n]\}$. We think of $[m]$ as a set of pigeons and of $[n]$ as a set of holes. The $FPHP^m_n$ formula consists of the following clauses:

- $\bigvee_{j=1}^{n} p_{i,j}$ for every $i \in [m]$;
- $\overline{p}_{i_1,j} \vee \overline{p}_{i_2,j}$ for all distinct $i_1, i_2 \in [m]$ and all $j \in [n]$;
- $\overline{p}_{i,j_1} \vee \overline{p}_{i,j_2}$ for all $i \in [m]$ and all distinct $j_1, j_2 \in [n]$.

Recall the definition of the extended version of a CNF formula $F$ given in Definition 4.2. Our first main result says that for any such weight-constrained CNF $F$, the space complexity of $F$ and of its extended version $\widetilde{F}$ are roughly the same in resolution.

**Theorem 7.3 (Theorem 1.5 for resolution).** *Let $F$ be a weight-constrained CNF and let $\widetilde{F}$ be its extended version. Then it holds that*

$$Sp_{\mathcal{R}}(F \vdash \bot) = Sp_{\mathcal{R}}\big(\widetilde{F} \vdash \bot\big) + \mathrm{O}(1) \enspace .$$

*Proof.* For one direction, observe that we can derive any clause of $F$ from $\widetilde{F}$ in space 3.

For the other direction, let $\pi = \{\mathbb{C}_1, \ldots, \mathbb{C}_t\}$ be a resolution refutation of $\widetilde{F}$. Our goal will be to show a resolution refutation of $F$ which uses only a constant additional space. For this end we will define a map which substitutes extension variables with sub-clauses in the original variables. Let $C = a_1 \vee a_2 \vee \ldots \vee a_n$ be a clause in $F$, and let $\{y_0\} \cup \{\overline{y}_{i-1} \vee a_i \vee y_i \mid 1 \leq i \leq n\} \cup \{\overline{y}_n\}$ be its extended version. Intuitively, setting the extension variable $y_i$ to true implies that at least one of the variables $a_{i+1}, \ldots a_n$ is true while setting $y_i$ to false implies that at least one of the variables $a_1, \ldots, a_i$ is true. Our substitution will try to capture exactly this intuition.

More precisely, the substitution into an extension variable $y_i$ will be defined as follows:

$$y_i \mapsto \bigvee_{t=i+1}^{n} a_t \quad \overline{y}_i \mapsto \bigvee_{t=1}^{i} a_t \quad y_n \mapsto \bot \quad \overline{y}_n \mapsto \top \enspace . \tag{7.1}$$

Let $\pi' = \{\mathbb{C}'_1, \ldots, \mathbb{C}'_t\}$ be the sequence of configurations obtained from $\pi$ by substituting each extension variable in a configuration in $\pi$ according to the above substitution. Our goal will be to show that $\pi'$ can be turned into a syntactic small space resolution refutation of $F$. Clearly, the space complexity of $\pi'$ is the same as the space complexity of $\pi$. It remains to show that all transitions from $\mathbb{C}'_i$ to $\mathbb{C}'_{i+1}$ can be carried out syntactically in the resolution proof system with only additional constant space. We split into cases according to the way in which $\mathbb{C}_{i+1}$ was obtained from $\mathbb{C}_i$ in the refutation $\pi$ of $\widetilde{F}$.

**Erasure.** Clearly, in this case the transition from $\mathbb{C}'_i$ to $\mathbb{C}'_{i+1}$ can be made by applying the same rule without increasing the space complexity.

**Axiom download.** Suppose that $\mathbb{C}_{i+1}$ was obtained from $\mathbb{C}_i$ by downloading an axiom $A \in \widetilde{F}$. It can be verified that in this case the substitution turns $A$ into either $\top$ or a clause $C$ in $F$, and thus the transition from $\mathbb{C}'_i$ to $\mathbb{C}'_{i+1}$ can be made by downloading the axiom $C$ without increasing the space complexity.

**Inference rules.** Suppose that $\mathbb{C}_{i+1}$ was obtained from $\mathbb{C}_i$ by resolving on a variable which is not an extension variable. Then in this case $\mathbb{C}'_{i+1}$ can be obtained from $\mathbb{C}'_i$ by applying the same rule. Suppose otherwise that $\mathbb{C}_{i+1}$ was obtained from $\mathbb{C}_i$ by resolving the clauses $A \vee y_i$ and $B \vee \overline{y_i}$ on the extension variable $y_i$. Let $X$ and $Y$ be the clauses obtained from $A$ and $B$ respectively after the substitution. Our goal will be to show that $X \vee Y$ can be obtained from $X \vee \bigvee_{t=i+1}^{n} a_i$ and $Y \vee \bigvee_{t=1}^{i} a_i$ in constant clause space, where $C = a_1 \vee a_2 \ldots \vee a_n$ is a clause of $F$. But since $F$ is weight-constrainted we have that $\overline{a}_i \vee \overline{a}_j$ is a clause of $F$ for all $1 \leq i < j \leq n$. Thus the desired conclusion follows from Claim 4.8. □

Our second main result is an analogue of the above theorem for the PCR proof system.

**Theorem 7.4 (Theorem 1.5 for PCR).** *Let $F$ be a weight-constrained CNF and let $\widetilde{F}$ be its extended version. Then it holds that*

$$Sp_{\mathcal{PCR}}(F \vdash \bot) = \Theta\big(Sp_{\mathcal{PCR}}(\widetilde{F} \vdash \bot)\big) \ .$$

*Proof.* The proof is quite similar to the proof of Theorem 7.3 except that we now define a substitution of extension variables with products of variables instead of sub-clauses. Formally, let $C = a_1 \vee a_2 \vee \ldots \vee a_n$ be a clause of $F$, and let $\{y_0\} \cup \{\overline{y}_{j-1} \vee a_j \vee y_j \mid 1 \leq j \leq n\} \cup \{\overline{y}_n\}$ be its extended version. The substitution into an extension variable $y_i$ will be defined as follows:

$$y_i \mapsto \prod_{t=i+1}^{n} a_t \quad \overline{y}_i \mapsto \prod_{t=1}^{i} a_t \quad y_n \mapsto 1 \quad \overline{y}_n \mapsto 0 \ . \tag{7.2}$$

As was the case in the proof of Theorem 7.3, let $\pi = \{\mathbb{C}_1, \ldots, \mathbb{C}_t\}$ be a PCR refutation of $\widetilde{F}$, and let $\pi' = \{\mathbb{C}'_1, \ldots, \mathbb{C}'_t\}$ be the sequence of configurations obtained from $\pi$ by substituting each extension variable in a configuration in $\pi$ according to the above substitution. As was the case before, the monomial space of $\pi'$ is the same as that of $\pi$ and it remains to show that all transitions from $\mathbb{C}'_i$ to $\mathbb{C}'_{i+1}$ can be carried out syntactically in the PCR proof system in small space. We split into cases according to the way in which $\mathbb{C}_{i+1}$ was obtained from $\mathbb{C}_i$ in the refutation $\pi$ of $\widetilde{F}$.

**Erasure.** Clearly, in this case the transition from $\mathbb{C}'_i$ to $\mathbb{C}'_{i+1}$ can be made by applying the same rule.

**Axiom download of a clause in $\widetilde{F}$.** Suppose that $\mathbb{C}_{i+1}$ was obtained from $\mathbb{C}_i$ by downloading the axiom $A \in \widetilde{F}$. It can be verified that the substitution turns the PCR translation of $A$ either into the PCR translation of a clause in $C \in F$ or to the polynomial 0. Thus the transition from $\mathbb{C}'_i$ to $\mathbb{C}'_{i+1}$ can be made by downloading the axiom $C \in F$ without increasing the monomial space.

**Inference rules.** Clearly, if $\mathbb{C}_{i+1}$ was obtained from $\mathbb{C}_i$ via the addition rule or via multiplication by a variable which is not an extension variable then $\mathbb{C}'_{i+1}$ can be obtained from $\mathbb{C}'_i$ using the same rule. If $\mathbb{C}_{i+1}$ was obtained from $\mathbb{C}_i$ via multiplication of a polynomial $Q$ by an extension variable $y_i$ then this can be simulated by a sequence of multiplications by the variables $a_{i+1}, a_{i+2}, \ldots, a_n$. For this one additional space is needed for keeping the intermediate polynomials of the form $\prod_{t=i+1}^{r} a_t \cdot Q$, $i+1 \leq r < n$, in memory. Since the configuration $\mathbb{C}_{i+1}$ contains both polynomials $Q$ and $y_iQ$ and has space at most $s$, this implies that the space of the intermediate polynomial $\prod_{t=i+1}^{r} a_t \cdot Q$ is at most $s/2$. Thus the monomial complexity increases by a factor of at most $3/2$. A multiplication by an extension variable $\overline{y}_i$ can be simulated in PCR similarly.

**Logical axioms.** If $\mathbb{C}_{i+1}$ was obtained from $\mathbb{C}_i$ by downloading an axiom of the form $x^2 - x$ where $x$ is not an extension variable then $\mathbb{C}'_{i+1}$ can be obtained from $\mathbb{C}'_i$ by downloading the same axiom. Consider now the logical axioms $y_i^2 - y_i$ and $\overline{y}_i^2 - \overline{y}_i$ for an extension variable $y_i$. The corresponding polynomials after substitution have the form $m^2 - m$ for some square-free monomial $m = v_1 \cdots v_l$ and we deduce them in constant space: for each $0 \leq i \leq l$ we download $v_i^2 - v_i$ and we multiply to obtain

$$(v_1 \cdots v_{i-1})v_i^2(v_{i+1}^2 \cdots v_l^2) - (v_1 \cdots v_{i-1})v_i(v_{i+1}^2 \cdots v_l^2) \ . \tag{7.3}$$

The sum of all these polynomials results in $m^2 - m$. Notice that a constant number of these polynomials is sufficient to be kept in memory in order to do the sum, and a constant number of monomials is sufficient to produce each of them.

It remains to show how to handle the logical axioms of the form $y_i + \overline{y}_i - 1$. In particular we want to deduce $\prod_{t=i+1}^{n} a_t + \prod_{t=1}^{i} a_t - 1$ using constant space. The fact that $F$ is weight-constrained allows to use Claim 4.8: there is a constant space resolution refutation for clauses $\bigvee_{t=1}^{i} a_t$ and $\bigvee_{t=i+1}^{n} a_t$. PCR simulates resolution space efficiently, so it is possible to deduce 1 from polynomials $p := \prod_{t=1}^{i} a_t$ and $q := \prod_{t=i+1}^{n} a_t$ in constant monomial space. Multiply every line in the latter deduction by $(p-1)(q-1)$ to get that $p(p-1)(q-1)$ and $q(p-1)(q-1)$ infer $(p-1)(q-1)$ in at most 4 times the original monomial space. Polynomials $p(p-1)(q-1)$ and $q(p-1)(q-1)$ are multiple of $p^2 - p$ and $q^2 - q$, respectively, so they follow in constant space from logical axioms. With $(p-1)(q-1)$ in memory we can download $pq$, which is the encoding of initial clause $C$: the difference $pq-(p-1)(q-1)$ is exactly the desired polynomial $p+q-1$. □

## 8 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 [ABRW02]. 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.)[6]

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 [HN12], a natural candidate for such results are the so-called pebbling formulas studied in [BN08, BN11], and another recent time-space trade-offs paper that might be relevant is [BBI12].[7]

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?

[6] As noted in Section 1.3, Bonacina and Galesi [BG12] announced such a result just as the final version of this paper was being prepared, which also resolves the problem in item 1. Interestingly, however, so far their techniques work only for $k \geq 4$ and not for $k = 3$.

[7] Again as noted in Section 1.3, results along these lines have now been reported in [BNT12].

## 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. We are also grateful to Mladen Mikša and Marc Vinyals for carefully reading this manuscript and finding numerous typos and other mistakes that needed to be fixed. Needless to say, any remaining ones are solely the responsibility of the authors.

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, and also performed part of this work while at Sapienza – Università di Roma. 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 research of the fourth author was supported by the Israel Ministry of Science and Technology. Also, part of the work of the fourth 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*. The fifth 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.

## References

[ABRW02] Michael Alekhnovich, Eli Ben-Sasson, Alexander A. Razborov, and Avi Wigderson. Space complexity in propositional calculus. *SIAM Journal on Computing*, 31(4):1184–1211, 2002. Preliminary version appeared in *STOC '00*.

[AD08] Albert Atserias and Víctor Dalmau. A combinatorial characterization of resolution width. *Journal of Computer and System Sciences*, 74(3):323–334, May 2008. Preliminary version appeared in *CCC '03*.

[AFT11] Albert Atserias, Johannes Klaus Fichte, and Marc Thurley. Clause-learning algorithms with many restarts and bounded-width resolution. *Journal of Artificial Intelligence Research*, 40:353–373, January 2011. Preliminary version appeared in *SAT '09*.

[AR03] Michael Alekhnovich and Alexander A. Razborov. Lower bounds for polynomial calculus: Non-binomial case. *Proceedings of the Steklov Institute of Mathematics*, 242:18–35, 2003. Available at http://people.cs.uchicago.edu/~razborov/files/misha.pdf. Preliminary version appeared in *FOCS '01*.

[BBI12] Paul Beame, Chris Beck, and Russell 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)*, pages 213–232, May 2012.

[BD09] Michael Brickenstein and Alexander Dreyer. PolyBoRi: A framework for Gröbner-basis computations with Boolean polynomials. *Journal of Symbolic Computation*, 44(9):1326–1345, September 2009.

[BDG$^+$09] Michael Brickenstein, Alexander Dreyer, Gert-Martin Greuel, Markus Wedler, and Oliver Wienand. New developments in the theory of Gröbner bases and applications to formal verification. *Journal of Pure and Applied Algebra*, 213(8):1612–1635, August 2009.

[Bea04] Paul Beame. Proof complexity. In Steven Rudich and Avi Wigderson, editors, *Computational Complexity Theory*, volume 10 of *IAS/Park City Mathematics Series*, pages 199–246. American Mathematical Society, 2004.

[Ben09] Eli Ben-Sasson. Size space tradeoffs for resolution. *SIAM Journal on Computing*, 38(6):2511–2525, May 2009. Preliminary version appeared in *STOC '02*.

[BG03] Eli Ben-Sasson and Nicola Galesi. Space complexity of random formulae in resolution. *Random Structures and Algorithms*, 23(1):92–109, August 2003. Preliminary version appeared in *CCC '01*.

[BG12] Ilario Bonacina and Nicola Galesi. Pseudo-partitions, transversality and locality: A combinatorial characterization for the space measure in algebraic proof systems. Technical Report TR12-119, Electronic Colloquium on Computational Complexity (ECCC), September 2012.

[BGIP01] Samuel R. Buss, Dima Grigoriev, Russell Impagliazzo, and Toniann Pitassi. Linear gaps between degrees for the polynomial calculus modulo distinct primes. *Journal of Computer and System Sciences*, 62(2):267–289, March 2001. Preliminary version appeared in *CCC '99*.

[BHJ08] Samuel R. Buss, Jan Hoffmann, and Jan Johannsen. Resolution trees with lemmas: Resolution refinements that characterize DLL-algorithms with clause learning. *Logical Methods in Computer Science*, 4(4:13), December 2008.

[BHvMW09] Armin Biere, Marijn J. H. Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, February 2009.

[BI10] Eli Ben-Sasson and Russell Impagliazzo. Random CNF's are hard for the polynomial calculus. *Computational Complexity*, 19:501–519, 2010. Preliminary version appeared in *FOCS '99*.

[BJ10] Eli Ben-Sasson and Jan 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)*, volume 6175 of *Lecture Notes in Computer Science*, pages 16–29. Springer, July 2010.

[BKPS02] Paul Beame, Richard Karp, Toniann Pitassi, and Michael Saks. The efficiency of resolution and Davis-Putnam procedures. *SIAM Journal on Computing*, 31(4):1048–1075, 2002. Preliminary versions of these results appeared in *FOCS '96* and *STOC '98*.

[Bla37] Archie Blake. *Canonical Expressions in Boolean Algebra*. PhD thesis, University of Chicago, 1937.

[BN08] Eli Ben-Sasson and Jakob 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)*, pages 709–718, October 2008.

[BN11] Eli Ben-Sasson and Jakob Nordström. Understanding space in proof complexity: Separations and trade-offs via substitutions. In *Proceedings of the 2nd Symposium on Innovations in Computer Science (ICS '11)*, pages 401–416, January 2011. Full-length version available at http://eccc.hpi-web.de/report/2010/125/.

[BNT12] Chris Beck, Jakob Nordström, and Bangsheng Tang. Some trade-off results for polynomial calculus. Submitted, 2012.

[BP98] Samuel R. Buss and Toniann Pitassi. Resolution and the weak pigeonhole principle. In *Proceedings of the 11th International Workshop on Computer Science Logic (CSL '97)*, volume 1414 of *Lecture Notes in Computer Science*, pages 149–156. Springer, 1998.

[BPR95] Maria Bonet, Toniann Pitassi, and Ran Raz. Lower bounds for cutting planes proofs with small coefficients. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing (STOC '95)*, pages 575–584, May 1995.

[BS97] Roberto J. Bayardo Jr. and Robert Schrag. Using CSP look-back techniques to solve real-world SAT instances. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI '97)*, pages 203–208, July 1997.

[BW01] Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow—resolution made simple. *Journal of the ACM*, 48(2):149–169, March 2001. Preliminary version appeared in *STOC '99*.

[CCT87] William Cook, Collette Rene Coullard, and Gyorgy Turán. On the complexity of cutting-plane proofs. *Discrete Applied Mathematics*, 18(1):25–38, November 1987.

[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 Theory of Computing (STOC '96)*, pages 174–183, May 1996.

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

[CS88] Vašek Chvátal and Endre Szemerédi. Many hard examples for resolution. *Journal of the ACM*, 35(4):759–768, October 1988.

[DLL62] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem proving. *Communications of the ACM*, 5(7):394–397, July 1962.

[DP60] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7(3):201–215, 1960.

[ET01] Juan Luis Esteban and Jacobo Torán. Space bounds for resolution. *Information and Computation*, 171(1):84–97, 2001. Preliminary versions of these results appeared in *STACS '99* and *CSL '99*.

[FLN$^+$12] Yuval Filmus, Massimo Lauria, Jakob Nordström, Neil Thapen, and Noga Ron-Zewi. Space complexity in polynomial calculus. In *Proceedings of the 27th Annual IEEE Conference on Computational Complexity (CCC '12)*, pages 334–344, June 2012.

[Hak85] Armin Haken. The intractability of resolution. *Theoretical Computer Science*, 39(2-3):297–308, August 1985.

[HN12] Trinh Huynh and Jakob 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)*, pages 233–248, May 2012.

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

[JMNŽ12] Matti Järvisalo, Arie Matsliah, Jakob Nordström, and Stanislav Živný. Relating proof complexity measures and practical hardness of SAT. In *Proceedings of the 18th International Conference on Principles and Practice of Constraint Programming (CP '12)*, volume 7514 of *Lecture Notes in Computer Science*, pages 316–331. Springer, October 2012.

[MS96] João P. Marques-Silva and Karem A. Sakallah. GRASP—a new search algorithm for satisfiability. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD '96)*, pages 220–227, November 1996.

[NH08] Jakob Nordström and Johan Håstad. Towards an optimal separation of space and length in resolution (Extended abstract). In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC '08)*, pages 701–710, May 2008.

[Nor09a] Jakob Nordström. Narrow proofs may be spacious: Separating space and width in resolution. *SIAM Journal on Computing*, 39(1):59–121, May 2009. Preliminary version appeared in *STOC '06*.

[Nor09b] Jakob Nordström. A simplified way of proving trade-off results for resolution. *Information Processing Letters*, 109(18):1030–1035, August 2009. Preliminary version appeared in ECCC report TR07-114, 2007.

[Nor12] Jakob Nordström. 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/.

[Pud97] Pavel Pudlák. Lower bounds for resolution and cutting plane proofs and monotone computations. *Journal of Symbolic Logic*, 62(3):981–998, September 1997.

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

[Rob65] John Alan Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1):23–41, January 1965.

[SAT] The international SAT Competitions. http://www.satcompetition.org.

[Seg07] Nathan Segerlind. The complexity of propositional proofs. *Bulletin of Symbolic Logic*, 13(4):482–537, December 2007.

[Urq87] Alasdair Urquhart. Hard examples for resolution. *Journal of the ACM*, 34(1):209–219, January 1987.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de