SIAM J. COMPUT.
Vol. 44, No. 4, pp. 1119–1153
© 2015 Society for Industrial and Applied Mathematics

SPACE COMPLEXITY IN POLYNOMIAL CALCULUS*

YUVAL FILMUS†, MASSIMO LAURIA‡, JAKOB NORDSTRÖM§, NOGA RON-ZEWI¶, AND NEIL THAPEN‖

Abstract. During the last 10 to 15 years, 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 concern 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 proof systems of interest, however, such as polynomial calculus or cutting planes, progress has been more limited. Essentially nothing has been known about space complexity in cutting planes, and for polynomial calculus the only lower bound has been for conjunctive normal form (CNF) formulas of unbounded width in [Alekhnovich et al., SIAM J. Comput., 31 (2002), pp. 1184–1211], 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 by Alekhnovich et al.: (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 O(log n), and hence this is an exponential improvement over Alekhnovich et al. 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.

*Received by the editors October 22, 2012; accepted for publication (in revised form) March 4, 2015; published electronically August 26, 2015. A preliminary version of this paper appeared in *Proceedings of the 27th Annual IEEE Conference on Computational Complexity,* 2012.

http://www.siam.org/journals/sicomp/44-4/89595.html

†Department of Computer Science, University of Toronto, Toronto, ON M5S 2E4, Canada (yfilmus@ias.edu). The research of this author has received funding from the European Union's Seventh Framework Programme (FP7/2007–2013) under grant agreement 238381 and from the National Science Foundation under agreement DMS-1128155.

‡KTH Royal Institute of Technology, SE-100 44 Stockholm, Sweden (lauria@kth.se). The research of this author was supported by the Eduard Čech Center for Algebra and Geometry and by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007–2013)/ERC grant agreement 279611, and part of this work was performed at Sapienza–Università di Roma.

§KTH Royal Institute of Technology, SE-100 44 Stockholm, Sweden (jakobn@kth.se). The research of this author was supported by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007–2013)/ERC grant agreement 279611 and by Swedish Research Council grants 621-2010-4797 and 621-2012-5645.

¶School of Mathematics, Institute for Advanced Study, Princeton, NJ 08540 (nogazewi@ias.edu). The research of this author was supported by the Israel Ministry of Science and Technology, by the Rothschild fellowship, and by NSF grant CCF-1412958. Part of the work of this 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*.

‖Institute of Mathematics, Academy of Sciences of the Czech Republic, Žitná 25, 115 67 Praha 1, Czech Republic (thapen@math.cas.cz). This 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.

Key words. proof complexity, space, polynomial calculus, polynomial calculus resolution, PCR, resolution, lower bounds

AMS subject classifications. 03B05, 03B35, 03B70, 03D15, 03F20, 68Q17, 68T15

DOI. 10.1137/120895950

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 π (a *proof*) for which $P(x, \pi) = 1$, while for any $x \notin L$ it holds for all strings π that $P(x, \pi) = 0$. A *propositional proof system* is a proof system for TAUTOLOGY, the language of tautologies in propositional logic.

The field of proof complexity, initiated by Cook and Reckhow [30], studies the complexity of proving propositional formulas in different propositional proof systems. One important motivation for proof complexity is the problem of P versus NP. A proof system is said to be polynomially bounded if for every $x \in L$ there is a proof π_x of size at most polynomial in the size of x. As observed in [30], one way of establishing $\mathsf{coNP} \neq \mathsf{NP}$, and hence $\mathsf{P} \neq \mathsf{NP}$, would be to prove that there are no polynomially bounded proof systems. This goal remains very distant, however, and it is probably fair to say that most current work in proof complexity is driven by other concerns.

One such other concern, which has motivated an interesting line of research in proof complexity in the last 10 to 15 years, 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 two decades 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 (DPLL)* procedure [32, 33] from the early 1960s augmented with clause learning [5, 43, 45]; these programs are also known as *conflict-driven clause learning (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 [54] 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 notable recent examples such as [4, 13, 26, 50]. 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, let

[1] Perhaps we should point out that this is not just a vague, abstract hope that theory and practice should somehow be related—for instance, recent experimental work by the third author joint with Järvisalo, Matsliah and Živný [42] suggests that a deeper study of possible correlations between theoretical space complexity in resolution and practical hardness (measured as the running time) for CDCL solvers could yield interesting insights.

us mention that some good starting points for a further study of proof complexity in general are, for instance, [6, 55], while the survey [48] by the third author focuses specifically on space complexity and time-space trade-offs. On the more applied side, a recent and very comprehensive reference on SAT solving is [17].

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

Of the three proof systems mentioned above, the *resolution proof system* is by far the most well-studied and well-understood. Resolution was introduced in [18] and began to be investigated in connection with automated theorem proving in the 1960s [32, 33, 53]. 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 breakthroughs in proof complexity was the result by Haken [39] that CNF formulas encoding the pigeonhole principle (PHP) require proofs of exponential length. There has been a sequence of follow-up papers establishing quantitatively stronger bounds for other formula families in, for instance, [8, 16, 28, 56].

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 [34]. Alekhnovich et al. [1] extended the concept of space to a more general setting, including other proof systems, and this setting is what will be of interest to us in this paper. Intuitively, the space of a resolution refutation is the amount of memory needed while verifying the refutation (where in resolution usually one thinks of each clause as requiring one unit of memory, a measure that is known as *clause space*). Perhaps somewhat surprisingly, it turns out that one need never use more than linear (clause) space in resolution, and a sequence of papers [1, 11, 34] have established matching lower bounds (up to constant factors).

Another sequence of papers [46, 49, 14] involving the third author clarified the relation between length and space in resolution. While it follows from [3] that small complexity with respect to space implies small complexity with respect to length, building on [46, 49] the paper [14] 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 [10, 47], and strong trade-offs for general resolution were finally obtained in [15]. Even more recently, [7] 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, Edmonds, and Impagliazzo [29], 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 [41], and a number of strong lower bounds on proof size have been obtained by proving degree lower bounds in, for instance, [2, 12, 25, 41, 52].

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

In this stronger PCR system, [1] managed to establish lower bounds on space measured as the number of monomials, but only sublinear bounds and only for formulas of unbounded width (namely, for pigeonhole principle formulas). The problem of proving linear lower bounds on space in PC and PCR, and more importantly of proving any nontrivial lower bounds for formulas of bounded width in terms of PCR space or even PC space, was mentioned as an open problem in [1]. Thus, it has been theoretically consistent with the state of knowledge since [1] 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 for PC or PCR before (the partial results in) the recent paper [40].

At the time the paper [29] 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 [23], but in general they seem to be an order of magnitude slower than state-of-the-art CDCL solvers (although [24] 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* proof system [31]. 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 know of only one superpolynomial lower bound on cutting planes proof size [51] (improving on the result [22] in a somewhat restricted setting). It is natural to define the *line space* of a cutting planes 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 cutting planes 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 [40].

[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.

1.2. Our results. In this paper, we focus on PC 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 [48] fairly closely, but for completeness we provide all the necessary preliminaries in section 2.

1.2.1. Upper bounds on space for k-CNF formulas in PC. 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 $\exp\bigl(\min\{m,n\}+\mathrm{O}(1)\bigr)$ 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 PC 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 counterexample for PC seems somewhat artificial, though, 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 [16] 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 not only that any refutation must contain at least *one* wide clause in such a case, but that it must contain 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 PC. 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 behavior 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 size* $\exp(\mathrm{O}(\min\{m,n\}))$ *and monomial space* $\mathrm{O}(\min\{m,n\})$*, where the hidden constant depends on k.*

1.2.2. Lower bounds on space for k-CNF formulas in PC. 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 $C = a_1 \vee a_2 \vee \cdots \vee a_w$ to the set of 3-clauses $\widetilde{C} = \{y_0\} \cup \{\overline{y}_{j-1} \vee a_j \vee y_j \mid 1 \leq j \leq w\} \cup \{\overline{y}_w\}$, where the y_i are new variables that do not appear anywhere else. Let $\widetilde{PHP}^m_n$ denote the pigeonhole principle formula PHP^m_n converted to 3-CNF in this way. For resolution we know that the space requirements for these two versions are the same, but the $\Omega(n)$ lower bound on space in PCR for PHP^m_n in [1] unfortunately breaks down for $\widetilde{PHP}^m_n$, and no lower bound has been known even for PC.

Intuitively, one would like to think of the semantics of the new auxiliary variables as being $y_i \equiv \bigvee_{j=i+1}^{w} a_j$ and use this to extract a PCR refutation of PHP^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 the pigeonhole principle converted to a 3-CNF formula $\widetilde{PHP}^{n+1}_n$ in PC 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 unfavorable encoding and could be avoided by a preprocessing step flipping the polarities of the literals in the formula in some way. However, it is straightforward to show, appealing to [1], that Theorem 1.2 holds even if we allow arbitrary flips of literal polarities in the formula.

1.2.3. Lower bounds on space for *k*-CNF formulas in PCR. The main goal of this paper is to prove lower bounds on space for k-CNF formulas not in PC, 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 $\ell = 1, \ldots, \log_2 n$ to encode 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 pigeonhole principle formulas" have width logarithmic in n and, as we prove, require space linear in n in PCR. Hence, this provides an exponential improvement over [1] measured in the width of the formulas.

THEOREM 1.3. *The space in PCR of refuting bit-graph pigeonhole principle formulas $BPHP^{n+1}_n$ is $\Omega(n)$, or $\Omega\big(\sqrt[3]{N/\log N}\big)$ expressed in the formula size N.*

We then tweak the formulas in a specific way to have "hole indicators" for each hole and pigeon, where we say that pigeon i sits in hole j if the exclusive or of the hole indicator variables for $(i, j-1)$ and (i, j) is true. While we certainly do not claim that this is the most natural encoding of the pigeonhole principle ever presented in

the literature, it has the nice feature that it can be written as a 4-CNF and that the proof of Theorem 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 [1] and follow the arguments in that paper fairly closely, but they 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 what happens in general to the space complexity of a CNF formula F when it is transformed to the 3-CNF formula $\widetilde{F}$ 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 toward resolving this question, we characterize a natural class of CNF formulas for which the space complexity in resolution and PCR provably does not decrease when the formula is transformed into a 3-CNF. Suppose that for every wide clause $a_1 \vee \cdots \vee a_w$ in F there are also axioms $\overline{a}_i \vee \overline{a}_j$ for all $1 \le i < j \le w$ 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* F *and* $\widetilde{F}$ *have the same space complexity in resolution up to a small additive constant, and in PCR the two formulas have the same space complexity up to a small multiplicative constant.*

In particular, this means that for the standard encoding of the *functional* pigeonhole principle, which has precisely such weight constraints, the space complexity of the wide formula and its 3-CNF version is essentially the same. Unfortunately, nothing is known about the PCR space complexity of this formula, and in particular the techniques in [1] break down when functional axioms are added to the formula. However, what Theorem 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 counterexample.

1.3. Subsequent developments. After the conference version [36] of this paper was published, Bonacina and Galesi [19] developed a further generalization of the

techniques in [1] and in our paper leading to, among other things, optimal (linear) space lower bounds in PCR for random k-CNF formulas with $k \geq 4$. The machinery developed in [19] was in turn used in [35] to prove further space lower bounds for Tseitin formulas and to describe a generic way of constructing formulas that are hard for PCR space (using so-called XORification). Very recently, Bonacina et al. [20] further refined the techniques in [19] to obtain the first PCR space lower bounds for 3-CNF formulas, albeit with a logarithmic factor loss.

In joint work [21] with Bonacina and Galesi, the fifth author developed the techniques in [19] in another direction to prove optimal quadratic lower bounds on total space in resolution of random 4-CNF formulas. This lower bound, too, was extended to random 3-CNF formulas in [20], except for a squared log factor loss.

Regarding time-space trade-offs, in [9] Beck and Tang together with the third author extended the size-space trade-off results for resolution in [15] and [7] to PCR, albeit with a slight loss in the parameters. In particular, this improves on the results for PCR established in [40]. For cutting planes, the trade-offs in [40] were significantly strengthened in [38].

Finally, in very recent joint work [37] with Galesi and Pudlák, the fifth author explained the failures so far to establish lower bounds on cutting planes space by showing that any CNF formula can be refuted by a cutting planes proof using only a constant number of linear inequalities in memory (although this proof has coefficients of exponential size).

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 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 discuss a few open problems.

2. Preliminaries. Let x be a Boolean variable. A *literal over* x is either the variable x itself (a *positive literal*) or its negation, denoted $\neg x$ or $\overline{x}$ (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 $\bot$. 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 α, β 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 $\bot$ 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 π in the system is a *sequence* of lines, where each line is derived from previous lines by one of a finite set of allowed *inference rules*. Following the exposition in [34], we view a proof as similar to a nondeterministic 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 [1], where 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 π is the sum of the sizes of all lines in a derivation, where lines that are derived multiple times are counted with repetitions. The* length *of a $\mathcal{P}$-derivation π is the number of axiom downloads and inference steps in it. For a space measure $Sp_\mathcal{P}(\mathbb{D})$ defined for $\mathcal{P}$-configurations, the* space *of a derivation π is defined as the maximal space of a configuration in π.*

*We define the $\mathcal{P}$-*refutation size *of a formula F, denoted $S_\mathcal{P}(F \vdash \bot)$, 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.*

Remark 2.3. We want to stress that length is defined as the number of axiom downloads and inference steps, whereas erasure steps do not count. Also, the size measure does not sum the sizes of all configurations in a derivation but rather the sizes of all formulas downloaded or inferred. This is to be consistent with the standard definitions in the proof complexity literature, where length is usually defined as the number of lines in a listing of the derivation, or the number of nodes in a directed acyclic graph representing the derivation, and where one sums the sizes of all lines/nodes to obtain the size of the derivation. The reader who so prefers, however, could instead define the length of a derivation $\pi = \{\mathbb{D}_0, \ldots, \mathbb{D}_\tau\}$ as the number of steps τ in it, since the difference is at most a factor of 2, and the size measure also

would not change significantly if one summed the sizes of all configurations $\mathbb{D}_t$.

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

$$\frac{G_1 \qquad \cdots \qquad G_m}{H} \tag{2.1}$$

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 2.4 (resolution). *In resolution, the lines in a derivation are clauses, and inferences follow the* resolution rule

$$\frac{B \vee x \qquad C \vee \overline{x}}{B \vee C} \tag{2.2}$$

for clauses B and C. We refer to (2.2) *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* weakening *rule*

$$\frac{B}{B \vee C} \tag{2.3}$$

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

Remark 2.6. When studying and comparing the complexity measures for resolution in Definition 2.5, as was noted in [1] 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 [47, 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 [29] under the name "Gröbner proof system." In a PC refutation, clauses are interpreted as multilinear polynomials. For instance, the requirement that the clause $x \vee y \vee \overline{z}$ should be satisfied gets translated to the equation $xy(1-z) = 0$ or $xy - xyz = 0$ (recall that we think of 0 as true and 1 as false), and we derive contradiction by showing that there is no common root for the polynomial equations corresponding to all the clauses.

DEFINITION 2.7 (polynomial calculus). *Lines in a PC proof are multivariate polynomial equations $p = 0$, where $p \in \mathbb{F}[x, y, z, \ldots]$ for some (fixed) field $\mathbb{F}$. It is customary to omit "$= 0$" and only write p. The derivation rules are as follows, where $\alpha, \beta \in \mathbb{F}$, $p, q \in \mathbb{F}[x, y, z, \ldots]$, and x is any variable:*

Boolean axioms. $\dfrac{}{x^2 - x}$ *(forcing 0/1-solutions)*

Linear combination. $\dfrac{p \qquad q}{\alpha p + \beta q}$

Multiplication. $\dfrac{p}{xp}$

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

DEFINITION 2.8 (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 PC 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).[3] *The (monomial) space of a PC configuration (a set of polynomials) is the total number of monomials in the configuration (counted with repetitions).*[4]

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 PC when compared to clause space in resolution. In order to obtain a cleaner, more symmetric treatment of proof space, in [1] the proof system *polynomial calculus resolution (PCR)* was introduced as a common extension of PC and resolution. The idea is to add an extra set of parallel formal variables $\overline{x}, \overline{y}, \overline{z}, \ldots$ so that positive and negative literals can both be represented in a space-efficient fashion. Thus, in PCR the clause $x \vee y \vee \overline{z}$ gets translated to the single monomial $xy\overline{z}$.

DEFINITION 2.9 (polynomial calculus resolution). *Lines in a PCR proof are polynomials over the ring $\mathbb{F}[x, \overline{x}, y, \overline{y}, z, \overline{z}, \ldots]$, where as before $\mathbb{F}$ is some field. We have all the axioms and rules of PC plus the following axioms:*

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

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

DEFINITION 2.10 (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.*

[3] 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 2.8, 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).

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

The 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 [1], 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 [1].

DEFINITION 2.11 (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 about the relationship between space and length. If we can prove lower bounds not just for syntactic but even for semantic versions of proof systems, this of course makes these bounds much stronger.

An even stronger proof system, defined in [1], is *functional calculus (FC).* This purely semantic 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 2.12 (functional calculus). *A line of an FC derivation is an arbitrary Boolean function (that is, a Boolean-valued function of Boolean variables). The single inference rule is the semantic 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 an FC 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 2.13 (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*

$$\min\{s : \exists\{C_i\}_{i=1}^{s} : \forall f(x_1, \ldots, x_n) \in \mathbb{P}\ \exists g(y_1, \ldots, y_s)\ f \equiv g(C_1, \ldots, C_s)\}\ ,$$

where $C_1, \ldots, C_s$ are clauses over $x_1, \ldots, x_n$, 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,

$$f(x_1, \ldots, x_n) = \begin{cases} \top & \text{if } L(x_1, \ldots, x_n) = 0, \\ \bot & \text{if } L(x_1, \ldots, x_n) \neq 0. \end{cases} \tag{2.4}$$

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 [1] 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 PC. In this section, we prove that any CNF formula of constant width can be refuted in PC using linear space and exponential size simultaneously, as stated next.

THEOREM 3.1 (detailed version of Theorem 1.1). *Any unsatisfiable k-CNF formula over n variables can be refuted in the PC proof system in monomial space $2^k(n+6)$, length $4n \cdot 3^n$, and size $2^k \cdot 8n \cdot 3^n$.*

For the proof of the above theorem we introduce the notion of *negative width*, which is the number of negative literals in a clause. The reason for focusing on negative width is that, as discussed after Definition 2.8, the representation of negative literals is what causes the PC space to blow up compared to resolution (for which worst-case upper bounds as stated above were already known).

The proof of Theorem 3.1 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 clause space $n+2$, negative width k, and length 3^n simultaneously. In the second step we describe how such a refutation can be simulated in the PC proof system in monomial space $2^k(n+6)$, length $4n \cdot 3^n$, and size $2^k \cdot 8n \cdot 3^n$ simultaneously. Let us 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 π is the maximum negative width of a configuration in π. The negative width $W^{-}(F \vdash \bot)$ of refuting a CNF formula F is the minimum negative width of any resolution refutation of F.*

Our first technical lemma says that unsatisfiable k-CNF formulas can be refuted in resolution 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 negative width k, clause space $n+2$, and length 3^n simultaneously.*

For the proof, we need the following lemma from [34].

LEMMA 3.4. *Let F be an unsatisfiable CNF formula in n variables. Then F can be refuted in resolution in clause 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 π in clause space at most $n+2$ and length at most $2^{n+1}-1 \leq 3^n$. Since there are only k different variables the negative width of π is necessarily at most k.

For the induction step, suppose that the statement holds for every k-CNF formula in n variables and let F be an unsatisfiable k-CNF formula in $n+1$ variables. For $b \in \{\top, \bot\}$ and a clause C let $C{\restriction}_{x=b}$ denote the restricted clause obtained from C by setting x to b, i.e.,

$$C{\restriction}_{x=\bot} = \begin{cases} \top & \text{if } \overline{x} \in C, \\ C \setminus \{x\} & \text{if } x \in C, \\ C & \text{otherwise;} \end{cases} \qquad C{\restriction}_{x=\top} = \begin{cases} C \setminus \{\overline{x}\} & \text{if } \overline{x} \in C, \\ \top & \text{if } x \in C, \\ C & \text{otherwise.} \end{cases} \tag{3.1}$$

We denote by $F{\restriction}_{x=b}$ the k-CNF formula which contains all clauses of the form $C{\restriction}_{x=b}$ for $C \in F$ except the trivial $\top$ clauses (which can be removed without loss of generality).

Pick an arbitrary variable x and consider the restricted formula $F{\restriction}_{x=\bot}$. By the inductive hypothesis, $F{\restriction}_{x=\bot}$ has a refutation of length 3^n, clause space $n+2$, and negative width k. It is an easy observation (essentially from [16]) that from such a refutation we can obtain a derivation of x from F in the same length, clause space, and negative width simply by "unrestricting" F, i.e., by applying the same inference steps as in the refutation of $F{\restriction}_{x=\bot}$ but to the clauses of F. For any clause C in the refutation of $F{\restriction}_{x=\bot}$ the corresponding clause in the new derivation is either C or $C \vee x$. Thus the final configuration in this proof includes either the empty clause $\bot$ or the unit clause x. In either case, x can be derived via weakening. The length and space of the derivation do not change, and since x is a positive literal neither does the negative width.

From now on we keep the clause x in memory and next consider the formula $F{\restriction}_{x=\top}$. A naive approach would be to use the same strategy as above for deriving the clause $\overline{x}$ from F and conclude by resolving $\overline{x}$ with the literal x in memory. However, this will not work since it may cause an increase of negative width. We instead simulate such a 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 $C \vee \overline{x}$ is in F, so it is sufficient to download $C \vee \overline{x}$, resolve with x which is in memory, and erase $C \vee \overline{x}$. Notice that this simulation uses two additional clauses in memory. In the end we get the empty clause, which concludes the refutation of F.

So far we obtained a simulation which uses space $n+4$, instead of $n+3$, which is what we want. To save one more unit of space, we perform a preprocessing step on the refutation of $F{\restriction}_{x=\top}$. Namely, we enforce that no download step increases the clause space above $n+1$. If this happens at an axiom download, then the download step must be immediately followed by an erasure step, because otherwise the clause

space would go above $n+2$, which is contrary to our inductive hypothesis. But this means that we can swap the order of these two steps and first do the erasure step and then the download step. In this way the space after any download step will be at most $n+1$, which in turns means that our simulation causes the space to go up to $n+3$ at most.

The length of the total refutation is 3^n for the first part and $2 \cdot 3^n$ for the second part (each axiom downloads in the second part is substituted with an inferences of length 2, i.e., a download, a resolution step, and an erasure), yielding a total length of at most 3^{n+1}. It is straightforward to verify that the negative width is still k. The lemma follows. □

The second lemma needed to establish Theorem 3.1 says that if a k-CNF formula can be refuted in resolution in clause space s, negative width w, and length L simultaneously, then it can be refuted in PC in monomial space $2^w(s+4)$, length $4nL$, and size $2^w 8nL$ simultaneously. We will actually prove a slightly more general result that will be useful in section 4. For this generalization we need to count the total number of negative literals in a configuration.

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

LEMMA 3.6. *Suppose that the CNF formula F can be refuted in resolution simultaneously in clause space s, length L, negative width w, and total negative space N. Then F can be refuted in PC simultaneously in length $4nL$, size $2^w 8nL$, and monomial space $4 \cdot 2^w + \min\big\{s2^w, s + 2^N\big\}$.*

Proof. Let π be a resolution refutation of F in clause space at most s, length at most L, negative width at most w, and total negative space at most N. We may assume without loss of generality that there is no application of the weakening rule—as observed in Definition 2.4, this does not increase length or clause space, and it is easy to verify that the same holds also for negative width and total negative space.

Let π be written as the sequence of clause configurations $\{\mathbb{C}_1, \ldots, \mathbb{C}_t\}$ and let π' be the sequence of configurations $\{\mathbb{C}'_1, \ldots, \mathbb{C}'_t\}$ obtained from π by replacing each clause in a configuration $\mathbb{C}_i$ with its PC translation as described in Definition 2.8. Since the negative width of any clause in any configuration $\mathbb{C}_i$ is at most w, its PC translation has at most 2^w monomials. Since each configuration $\mathbb{C}_i$ contains at most s clauses this implies in turn that each PC configuration $\mathbb{C}'_i$ contains at most $s2^w$ monomials. But since we know that the total negative space is at most N, another upper bound on the number of monomials in a configuration $\mathbb{C}'_i$ is $s + 2^N$.

It remains to show how to carry out in PC the transition from a configuration $\mathbb{C}'_i$ to the next configuration $\mathbb{C}'_{i+1}$ in π'. 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 PC employing the same rule. By assumption π does not use the weakening rule so the only case that remains is when $\mathbb{C}_{i+1}$ is obtained from $\mathbb{C}_i$ by an application of the resolution rule to a pair of clauses $A \vee x$ and $B \vee \overline{x}$. This case is not hard but requires a bit of work.

CLAIM 3.7. *Let A and B be clauses such that the variable x does not appear in A or B. Suppose that $A \vee x$, $B \vee \overline{x}$, and $A \vee B$ all have negative width at most w. Let p_C denote the PC translation of a clause C according to Definition 2.8. Then*

it holds that the configuration $\{p_{A\vee x}, p_{B\vee\overline{x}}, p_{A\vee B}\} = \{xp_A, (1-x)p_B, p_{A\vee B}\}$ *can be derived from the configuration* $\{p_{A\vee x}, p_{B\vee\overline{x}}\}$ *in PC using at most* $4n$ *inference steps and additional monomial space at most* 2^{w+2}*. All polynomials in the derivation have at most* 2^{w+1} *monomials each.*

Proof. To prove the claim, we use as a building block the following space-efficient inference of the polynomial $\prod_\ell z_\ell \prod_\kappa (1-y_\kappa)$ from the polynomial 1, where $\{z_\ell\}_\ell$ and $\{y_\kappa\}_\kappa$ are two arbitrary sets of variables (which we think of as the positive and negative literals in some clause):

$$\begin{pmatrix} 1 \end{pmatrix} \to \begin{pmatrix} 1 \\ z_1 \end{pmatrix} \to \begin{pmatrix} 1 \\ z_1 \\ z_1 z_2 \end{pmatrix} \to \begin{pmatrix} 1 \\ z_1 z_2 \end{pmatrix} \to \cdots \to \begin{pmatrix} 1 \\ \prod_\ell z_\ell \end{pmatrix} \to \begin{pmatrix} 1 \\ \prod_\ell z_\ell \\ y_1 \prod_\ell z_\ell \end{pmatrix} \to$$

$$\to \begin{pmatrix} 1 \\ \prod_\ell z_\ell \\ y_1 \prod_\ell z_\ell \\ (1-y_1)\prod_\ell z_\ell \end{pmatrix} \to \begin{pmatrix} 1 \\ (1-y_1)\prod_\ell z_\ell \end{pmatrix} \to \begin{pmatrix} 1 \\ (1-y_1)\prod_\ell z_\ell \\ y_2(1-y_1)\prod_\ell z_\ell \end{pmatrix} \to \tag{3.2}$$

$$\to \begin{pmatrix} 1 \\ (1-y_1)\prod_\ell z_\ell \\ y_2(1-y_1)\prod_\ell z_\ell \\ (1-y_2)(1-y_1)\prod_\ell z_\ell \end{pmatrix} \to \begin{pmatrix} 1 \\ (1-y_2)(1-y_1)\prod_\ell z_\ell \end{pmatrix} \to \cdots \to \begin{pmatrix} 1 \\ \prod_\ell z_\ell \prod_\kappa (1-y_\kappa) \end{pmatrix}.$$

We stress that 1 is kept in the final configuration and that all polynomials above are PC encodings of clauses. Using (3.2) we can derive $p_{A\vee B}$ as follows:

- First derive $xp_{A\vee B} = p_{A\vee B\vee x}$ from xp_A.
- Then derive $(1-x)p_{A\vee B} = p_{A\vee B\vee\overline{x}}$ from $(1-x)p_B$.
- Finally, take the sum of these two polynomials and then erase them from memory.

The derivation of $xp_{A\vee B}$ from xp_A is just a suitably modified version of (3.2) with all polynomials multiplied by xp_A, and the same applies to the derivation of $(1-x)p_{A\vee B}$ from $(1-x)p_B$. The length of each subderivation (3.2) is the number of variables z_ℓ plus twice the number of variables y_κ (recall that erasure steps do not count toward length, as discussed in Remark 2.3). Hence, the total length is at most $4n$, where n is the total number of variables. Each polynomial is the encoding of a clause with at most $w+1$ negative literals, so it contains at most 2^{w+1} monomials.

To count the number of additional monomials, observe that the most expensive configuration is the first one that contains the polynomial $(1-x)p_{A\vee B}$. At this point we have in memory xp_A, $(1-x)p_B$, $xp_{A\vee B}$, $(1-x)p_{A\vee B}$, and all premises used to infer the latter polynomial. Let $w' \le w$ be the negative width of $A \vee B$. The number of monomials in $xp_{A\vee B}$ and $(1-x)p_{A\vee B}$ is, respectively, $2^{w'}$ and $2^{w'+1}$. The premises of $(1-x)p_{A\vee B}$ collectively account for $2^{w'+1}$ more monomials, since no monomial cancels in any of the inference steps in the subderivation (3.2). Thus, this configuration has $4 \cdot 2^{w'}$ additional monomials compared to the final configuration $\{xp_A, (1-x)p_B, p_{A\vee B}\}$. ∎

Returning to the proof of Lemma 3.6, the number of monomials in each configuration $\mathbb{C}'_i$ is at most $\min\{2^w s, 2^N + s\}$. For every $i \in [t]$, the transition from $\mathbb{C}'_i$ to $\mathbb{C}'_{i+1}$ is carried out in PC either directly or using Claim 3.7. In this way we obtain a PC refutation of F of length at most $4nL$, and space at most

$$2^{w+2} + \min\{2^w s, 2^N + s\} = \min\{2^w(s+4), 2^{w+2} + 2^N + s\} \tag{3.3}$$

as required. By assumption, the negative width is at most w and therefore all the polynomials in $\mathbb{C}_i$ have at most 2^w monomials. By Claim 3.7, the polynomials in the intermediate configurations have at most 2^{w+1} monomials. Thus, the total size of the refutation is $2^w 8nL$. This concludes the proof of the lemma. □

Theorem 3.1 now follows immediately from Lemmas 3.3 and 3.6. Let us write out the proof for completeness.

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 3^n simultaneously. Lemma 3.6 then yields that F can be refuted in PC in space $2^k((n+2)+4) = 2^k(n+6)$, length $4n \cdot 3^n$ and size $2^k \cdot 8n \cdot 3^n$. □

4. Space complexity of 3-CNF PHP formulas in PC. We next want to prove an $\Omega(n)$ space lower bound on the 3-CNF extended version of the pigeonhole principle PHP_n^m and show that this lower bound is tight up to constant factors. Let us start by recalling the definition of the pigeonhole principle and its extended version.

DEFINITION 4.1 (pigeonhole principle formula). *The pigeonhole principle formula PHP_n^m is the formula over variables $\{p_{i,j} \mid i \in [m], j \in [n]\}$, where we think of $[m]$ as the set of pigeons and of $[n]$ as the set of holes, consisting of the following clauses:*

- $\bigvee_{j=1}^n p_{i,j}$ *for all* $i \in [m]$ (pigeon axioms),
- $\overline{p}_{i_1,j} \vee \overline{p}_{i_2,j}$ *for all* $i_1, i_2 \in [m]$, $i_1 \neq i_2$, *and all* $j \in [n]$ (hole axioms).

As mentioned briefly in the introduction, the extended version $\widetilde{F}$ of a CNF formula F is defined as follows.

DEFINITION 4.2 (extended version). *The* extended version *of a clause* $C = a_1 \vee a_2 \vee \cdots \vee a_n$, $n > 3$, *is the set of clauses*

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

with $n+2$ *clauses over* $2n+1$ *variables. If* C *has at most three literals we define* $\widetilde{C} = \{C\}$. *We will refer to the variables* $y_0, y_1, \ldots, y_n$ *as* extension variables *as opposed to the* original variables *of* F.

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.*

It is not hard to verify that the extended version $\widetilde{F}$ of a CNF formula F is unsatisfiable if and only if F itself is unsatisfiable.

For the extended version $\widetilde{PHP}_n^m$ of the PHP formula we will denote the extension variables of the ith pigeon axiom $\bigvee_{j=1}^n p_{i,j}$ by $y_{i,j}$, so that this clause turns into the 3-CNF formula

$$y_{i,0} \wedge (\overline{y}_{i,0} \vee p_{i,1} \vee y_{i,1}) \wedge (\overline{y}_{i,1} \vee p_{i,2} \vee y_{i,2}) \wedge \cdots \wedge (\overline{y}_{i,n-1} \vee p_{i,n} \vee y_{i,n}) \wedge \overline{y}_{i,n}. \tag{4.1}$$

We will need the following theorem by Alekhnovich et al. [1].

THEOREM 4.3. *For any* $m > n$, *it holds that* $Sp_{\mathcal{PCR}}(PHP_n^m \vdash \bot) = \Omega(n)$.

Since PCR is at least as strong as PC with respect to space, the above theorem holds for PC as well.

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

It is not difficult to see that $Sp_{\mathcal{PC}}(\widetilde{F} \vdash \bot) \leq Sp_{\mathcal{PC}}(F \vdash \bot) + \mathrm{O}(1)$. This is so since every PC refutation of F can be transformed into a PC refutation of $\widetilde{F}$ using only constant additional space (by rederiving C from $\widetilde{C}$ whenever a clause C is

downloaded). The main theorem of this section says that for the PHP_n^m formulas the converse is also true (up to a constant multiplicative factor).

THEOREM 4.5. *For any positive integers $m > n$, it holds that $Sp_{\mathcal{PC}}(PHP_n^m \vdash \bot) \leq \frac{3}{2} Sp_{\mathcal{PC}}(\widetilde{PHP}_n^m \vdash \bot) + \mathrm{O}(1)$.*

Combining Corollary 4.4 and Theorem 4.5, we obtain the first nonconstant PC monomial space lower bound for a 3-CNF formula.

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

Proof of Theorem 4.5. We show how to transform any PC refutation of $\widetilde{PHP}_n^m$ into a PC refutation of PHP_n^m without increasing the monomial space by more than a constant factor. Let $\pi = \{\mathbb{C}_1, \dots, \mathbb{C}_\tau\}$ be a PC refutation of $\widetilde{PHP}_n^m$ in space s. Let $\pi' = \{\mathbb{C}'_1, \dots, \mathbb{C}'_\tau\}$ be the sequence of configurations obtained from π by replacing any extension variable $y_{i,j}$, $0 \leq j \leq n$, in any polynomial by the product $\prod_{\ell=j+1}^{n} p_{i,\ell}$ (in particular, we substitute 1 for the variable $y_{i,n}$).

Since variables are replaced by monomials, we have $Sp(\mathbb{C}'_i) = Sp(\mathbb{C}_i)$ for all $i \in [\tau]$. We need to show that all transitions from $\mathbb{C}'_i$ to $\mathbb{C}'_{i+1}$ can be carried out in PC without increasing the number of monomials by more than a constant factor. Clearly, if $\mathbb{C}_{i+1}$ is obtained from $\mathbb{C}_i$ by erasure, linear combination, or multiplication by an original variable of PHP_n^m, then $\mathbb{C}'_{i+1}$ can be obtained from $\mathbb{C}'_i$ by applying the same rule. It remains to analyze the cases when $\mathbb{C}_{i+1}$ is obtained from $\mathbb{C}_i$ through multiplication by an extension variable or by an axiom download.

If $\mathbb{C}_{i+1}$ is obtained from $\mathbb{C}_i$ by multiplying a polynomial q by an extension variable $y_{i,j}$, then this can be simulated by a sequence of multiplications by the variables $p_{i,j+1}, p_{i,j+2}, \dots, p_{i,n}$. For this we need extra space to store in memory, one at a time, the intermediate polynomials of the form $\prod_{\ell=j+1}^{r} p_{i,\ell} \cdot q$ for $j+1 \leq r < n$. 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 $\prod_{\ell=j+1}^{r} p_{i,\ell} \cdot q$ is at most $s/2$. Hence the monomial space increases by a factor of at most $3/2$.

Now consider the case when $\mathbb{C}_{i+1}$ is obtained from $\mathbb{C}_i$ via an axiom download. An axiom of the form $y_{i,0}$ is replaced by $\prod_{j \in [n]} p_{i,j}$, which is the PC translation of the ith pigeon axiom. The axiom $\overline{y}_{i,n}$ corresponds to $1 - 1 = 0$ and can thus be ignored. A generic axiom $\overline{y}_{i,j-1} \vee p_{i,j} \vee y_{i,j}$ gets translated in PC as $\big(1 - \prod_{\ell=j}^{n} p_{i,\ell}\big) p_{i,j} \prod_{\ell=j+1}^{n} p_{i,\ell} = \prod_{\ell=j}^{n} p_{i,\ell} - \prod_{\ell=j}^{n} p_{i,\ell}^2$. The latter polynomial can be derived in constant monomial space from the axioms $p_{i,j}^2 - p_{i,j}$. To see this, note first that we get $p_{i,j} - p_{i,j}^2$ by multiplying the Boolean axiom in the preceding sentence by -1. Suppose that we have derived $t - t^2$ for a (multilinear) monomial t and want to derive $xt - (xt)^2$. Then we can multiply $t - t^2$ by x^2 and $x - x^2$ by t (or in the latter case by the variables in t one by one, to be precise), and then add the two resulting polynomials to obtain $xt - (xt)^2$. A similar procedure can be employed to infer the substituted version of a Boolean axiom $y_{i,j}^2 - y_{i,j}$ over an extension variable. ∎

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 appealing to Theorem 3.1 is not sufficient here—this will only yield an upper bound of $\mathrm{O}(n^2)$ on the PC space of $\widetilde{PHP}_n^{n+1}$, since the number of variables in $\widetilde{PHP}_n^{n+1}$ is $\Theta(n^2)$. Recalling again that $Sp_{\mathcal{PC}}(\widetilde{F} \vdash \bot) \leq Sp_{\mathcal{PC}}(F \vdash \bot) + \mathrm{O}(1)$ for any CNF formula F, we see that it suffices to prove the upper bound stated next.

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

Putting together Corollary 4.6 and Lemma 4.7 we obtain Theorem 1.2. For the proof of Lemma 4.7 we will use the following claim (where we recall the definition of total negative space in Definition 3.5).

CLAIM 4.8. *For clauses C, D and positive integers s, t, let F be the CNF formula*

$$\left(C \vee \bigvee_{i=1}^{s} a_i\right) \wedge \left(D \vee \bigvee_{j=1}^{t} b_j\right) \wedge \left(\bigwedge_{i=1}^{s}\bigwedge_{j=1}^{t} (\overline{a}_i \vee \overline{b}_j)\right) \quad .$$

Then the clause $C \vee D$ can be derived from F in resolution in clause space 4. Furthermore, if a_i and b_j are all positive literals and C, D contain only positive literals, then $C \vee D$ can be derived from F in resolution in clause space 4 and total negative space 4 simultaneously.

Proof. For any fixed $i \in [s]$ we can derive $D \vee \overline{a}_i$ by resolving the axiom $D \vee \bigvee_{j=1}^{t} b_j$ with the axioms $\overline{a}_i \vee \overline{b}_1, \ldots, \overline{a}_i \vee \overline{b}_t$ one by one, erasing both premises from memory after each resolution step. In this way, we can derive any clause $D \vee \overline{a}_i$ in clause space 3. Then we can obtain $C \vee D$ by resolving $C \vee \bigvee_{i=1}^{s} a_i$ with $D \vee \overline{a}_i$ for $i = 1, \ldots, s$ using a total of four clauses of space. Finally, note that if the clauses C and D as well as all literals a_i and b_j are positive, then we have at most four negative literals simultaneously in memory. □

Let us now use Claim 4.8 to prove 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^{n+1} are also clauses of PHP_n^m, so this is without loss of generality.

We use (the proof of) Lemma 1 in [27] to show that PHP_n^{n+1} can be refuted in resolution in clause space $n + \mathrm{O}(1)$ and total negative space $\mathrm{O}(1)$ 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^{n+1} in [27] 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 $P \subseteq [n+1]$ and $H \subseteq [n]$ we denote by $(P \to H)$ the positive clause $\bigvee_{i \in P} \bigvee_{j \in H} p_{i,j}$. The axioms are our usual pigeon axioms, which for the ith pigeon is written as $(\{i\} \to [n])$ in this notation. The inference rule is

$$\frac{C \vee (A \to \{j\}) \qquad D \vee (B \to \{j\})}{C \vee D \vee (A \cap B \to \{j\})} \tag{4.2}$$

for positive clauses C, D, subsets $A, B \subseteq [n+1]$, and an integer $j \in [n]$. This can be thought of as a special version of the resolution rule (2.2) tailored to the PHP formula.

Let P be a (nonempty) set of pigeons. We will show by induction on the set size $|P|$ that for any $P \subseteq [n+1]$ the positive clause $\big(P \to [n+1-|P|]\big)$ can be derived in clause space at most $|P| + 2$. From this we can conclude that the specialized inference system derives the empty clause $([n+1] \to \emptyset)$ in clause space at most $n + 3$.

For the base case $|P| = 1$ this is immediate since the clause is an axiom. For $|P| > 1$, suppose that $P = \{i_1, i_2, \ldots i_t\}$ and let $j = n + 2 - |P|$ and $C = \big(P \to [n+1-|P|]\big)$. Note that each clause $C \vee \big(P \setminus \{i_\ell\} \to \{j\}\big)$ is a weakening of the clause $\big(P \setminus \{i_\ell\} \to [n+2-|P|]\big)$ which can be derived in clause space $|P| + 1$ by the inductive hypothesis. Then C can be derived in space $|P| + 2$ as follows:

$$\dfrac{\dfrac{\dfrac{C \vee (P \setminus \{i_1\} \to \{j\}) \qquad C \vee (P \setminus \{i_2\} \to \{j\})}{C \vee (P \setminus \{i_1, i_2\} \to \{j\})} \qquad C \vee (P \setminus \{i_3\} \to \{j\})}{\vdots} \qquad C \vee (P \setminus \{i_t\} \to \{j\})}{C \vee (\emptyset \to \{j\})}.$$

By the induction principle, we obtain a refutation in our specialized inference system in positive clause space at most $n + 3$.

Next, we show how to simulate this refutation in resolution. All clauses in the refutation contain only positive literals, so we just need to show how to simulate the inference rule (4.2) without increasing the clause space and the total negative space by more than a constant. If we let $\big(A \setminus B \to \{j\}\big) = \bigvee_{i=1}^{s} a_i$ and $\big(B \setminus A \to \{j\}\big) = \bigvee_{j=1}^{t} b_i$ and recall that we have axioms $\overline{p}_{i,j} \vee \overline{p}_{i',j}$ for all $i, i' \in [n+1]$, $i \neq i'$, and $j \in [n]$ corresponding to $\overline{a}_i \vee \overline{b}_j$, we can appeal to Claim 4.8 to deduce that the inference rule (4.2) 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 formula PHP_n^{n+1} can be refuted in clause space $n + \mathrm{O}(1)$ and total negative space $\mathrm{O}(1)$ simultaneously. Lemma 3.6 then implies that PHP_n^{n+1} can be refuted in monomial space $n + \mathrm{O}(1)$. As already observed, this then also holds for PHP_n^m for any $m > n$. ∎

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 [1], 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 constant width, resolving an open problem in [1]. We believe that the result in this section is of independent interest, however, since the lower bound holds for a natural family of CNF formulas, whereas the 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. In contrast to the standard encoding, which uses clauses $\overline{p}_{i,j} \vee \overline{p}_{i,j'}$ to enforce the functionality condition that the pigeon i is not sent to two distinct holes j and j', for the formulas studied in this section this condition 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 pigeonhole principle 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 being mapped 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 are mapped to the same hole. For every two pigeons $p_1 \neq p_2 \in [0, m)$ and every hole $h \in [0, n)$,

$BPHP_n^m$ contains a hole axiom

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

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

Recall 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 $v = b$. Keeping this in mind, we see that 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 to which this pigeon is sent does not match the binary expansion of h.

Fix $n = 2^\ell \geq 1$ and $m > n$. We will prove a PCR space lower bound for $BPHP_n^m$ using a similar approach to that in Alekhnovich et al. [1]. As in [1], our proof will also apply to the much stronger functional calculus proof system (see Definition 2.12).

Notice that we can identify total truth value assignments α to the variables of $BPHP_n^m$ with functions $f_\alpha : [0, m) \to [0, n)$ mapping pigeons to holes. In what follows, we will switch freely back and forth between these two ways of looking at assignments. Let us next state two key definitions.

DEFINITION 5.2 (well-behaved assignment). *Let α be a total assignment to the variables of $BPHP_n^m$ and let $S \subseteq [0, m)$ be a set of pigeons. We say that α is* well-behaved on *S if the holes assigned by α to the pigeons in S are all distinct.*

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

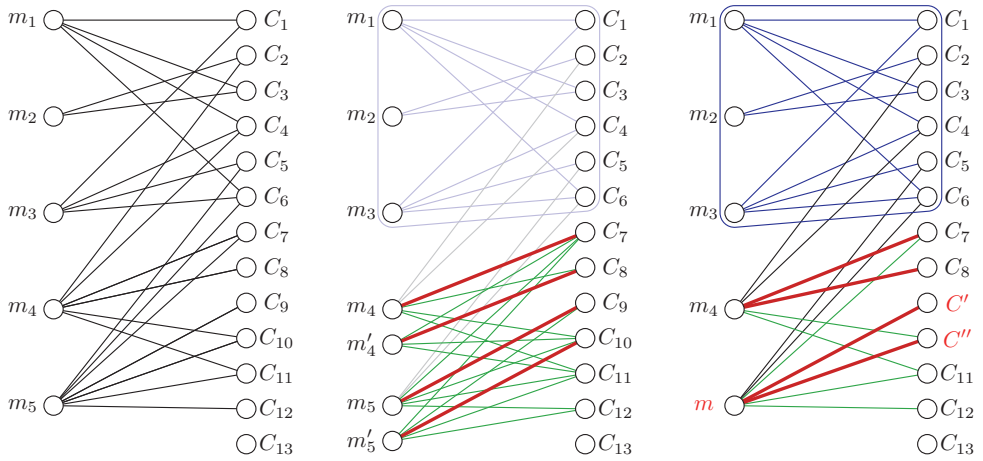
The following observation is central to our argument. It states that given a one-to-one assignment of fewer than $n/2$ pigeons to holes and a literal associated to a new pigeon, we can always find some new hole to assign to that pigeon so that the literal is satisfied.

LEMMA 5.4. *Suppose S is any set of fewer than $n/2$ pigeons, α is an assignment well-behaved on S, and $x[p, i]^b$ is a literal associated with a pigeon $p \notin S$. Then we can modify α by reassigning p in such a way that the new assignment is well-behaved on $S \cup \{p\}$ and satisfies the literal $x[p, i]^b$.*

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



(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) *Graph with* Γ *and matchings constructed at end of argument.*

FIG. 1. *Illustration of argument in proof of the locality lemma.*

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* α *which is well-behaved on and satisfies* $\mathbb{A}$ *also satisfies* $\mathbb{P}$.

The idea of the lower bound is that, given a purported refutation using small space, we can inductively construct a commitment set $\mathbb{A}_t$ for each configuration $\mathbb{P}_t$ in the proof in such a way that the commitment set $\mathbb{A}_t$ entails the configuration $\mathbb{P}_t$. The following lemma, based on a similar lemma in [1], is the technical heart of the lower bound. We will use it to show that as long as the configurations do not get too big, we never need to use a commitment set that is more than twice as large as the corresponding configuration. We can then use Corollary 5.5 to show that all the configurations are satisfiable, resulting in a contradiction.

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

It must hold for all $S \subseteq M \setminus \Gamma$ that $|N(S) \setminus N(\Gamma)| > 2 \cdot |S|$, since otherwise we could add S to Γ to get a larger set. But this implies that there is a matching of every monomial $m \in M \setminus \Gamma$ to two distinct commitments $C', C'' \in \mathbb{A} \setminus N(\Gamma)$ such

that no two m, m' share any commitments. To see this, just make two copies of each monomial/vertex in $m \in M \setminus \Gamma$ with the same edges from both copies to the vertices on the right, apply Hall's theorem, and then identify the two copies of the monomial again (this step is depicted in Figure 1(b), where Γ and $N(\Gamma)$ are in the upper half of the graph).

Pick 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 the definition of our bipartite graph, 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 the edges in the bipartite graph only guarantee that m mentions at least one pigeon in each of C' and C''. 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.

First, 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 $\mathbb{A}$. Second, 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 β 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. Assuming that we have an assignment β well-behaved on and satisfying $\mathbb{B}$, we show how to obtain another assignment α (depending on β) such that

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

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

To this end, let S 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 β to get a truth value assignment α that is well-behaved on $S \cup T$, agrees with β on pigeons outside T, and satisfies all literals in X. We claim that this is the assignment that we need.

To see this, note first that no monomial in Γ mentions pigeons in T (by construction), so α and β agree on monomials in Γ. For $m \in M \setminus \Gamma$, all β satisfying $\mathbb{B}$ must set the monomial m to zero, since this is how the new commitments were constructed. Reassigning pigeons in T can change variables in m, but there is still at least one variable that is set to zero, zeroing the whole monomial. So for all $m \in M \setminus \Gamma$, the assignment α gives the same value to m as does β, namely, 0. Hence α and β agree

on all monomials in M and $\mathbb{P}(\alpha) = \mathbb{P}(\beta)$. This takes care of item 1 above. By Corollary 5.5, α is well-behaved on $S \cup T = \operatorname{dom} \mathbb{A}$. Also, since α satisfies all literals in X as well as $N(\Gamma)$, consequently α satisfies $\mathbb{A}$. This takes care of item 2, and as already discussed it now follows that $\mathbb{P}(\alpha) = 0$. Thus, every β that is well-behaved on and satisfies $\mathbb{B}$ must also satisfy $\mathbb{P}$. The lemma follows. □

Using this lemma, we can now prove our PCR space lower bound for bit-graph pigeonhole principle formulas.

THEOREM 5.8 (detailed version of Theorem 1.3). $Sp_{\mathrm{PCR}}(BPHP_n^m \vdash \bot) > n/8$.

Proof. Let $\pi = \{\mathbb{P}_0, \dots, \mathbb{P}_\tau\}$ be a PCR refutation of $BPHP_n^m$ in monomial space at most $n/8$, i.e., a PCR derivation such that $\mathbb{P}_0 = \emptyset$ and $1 \in \mathbb{P}_\tau$. We will construct by induction a sequence of commitment sets $\mathbb{A}_0, \dots, \mathbb{A}_\tau$ such that for each step t, it holds that $|\mathbb{A}_t| \le 2 \cdot Sp(\mathbb{P}_t)$ 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 $\mathbb{P}_\tau$.

Clearly, we may define $\mathbb{A}_0$ to be the empty commitment. Now suppose we have constructed $\mathbb{A}_t$. To construct $\mathbb{A}_{t+1}$, we consider three cases depending on how $\mathbb{P}_{t+1}$ is obtained 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 set $\mathbb{A}_{t+1} = \mathbb{A}_t$. Let α be any assignment well-behaved on and satisfying $\mathbb{A}_{t+1}$. Then α satisfies $\mathbb{P}_t$ by the inductive hypothesis, and must also satisfy $H(p_1, p_2, h)$ since it is well-behaved on the pigeons in $H(p_1, p_2, h)$.

Otherwise, there are either one or two pigeons mentioned in $H(p_1, p_2, h)$ which are not in $\operatorname{dom} \mathbb{A}_t$. Then for each such pigeon p_i we add a "dummy" commitment C_{p_i} to $\mathbb{A}_t$ whose sole purpose is to put p_i into the domain of $\mathbb{A}_{t+1}$. We can take C_{p_i} to be $x[p_i, 0] \vee x[p_i', 0]$, where p_i' is any pigeon which has not been used so far. We can find such p_1' and p_2', if needed, because $|\mathbb{P}_t| \le |\mathbb{P}_{t+1}| - 1 \le \frac{n}{8} - 1$ and because the total number of used pigeons is at most $2|\mathbb{A}_t| + |\{p_1, p_2\}| \le 4|\mathbb{P}_t| + 2 < n - 2$. In both cases, we add at most two new commitments, and so $|\mathbb{A}_{t+1}| \le 2 \cdot Sp(\mathbb{P}_{t+1})$. Clearly, any well-behaved assignment assigning values to the variables in $H(p_1, p_2, h)$ satisfies this clause, and hence $\mathbb{A}_{t+1}$ entails $\mathbb{P}_{t+1}$ over well-behaved assignments, as required.

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

Erasure. Suppose $\mathbb{P}_{t+1} \subset \mathbb{P}_t$. Since $|\mathbb{A}_t| \le 2 \cdot Sp(\mathbb{P}_t) \le n/4$, Lemma 5.7 applies and furnishes us with a commitment set $\mathbb{A}_{t+1}$ such that $|\mathbb{A}_{t+1}| \le 2 \cdot Sp(\mathbb{P}_{t+1})$ 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. 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 $C_1, \dots, C_s$ (which we consider as Boolean functions), any function in $\mathbb{P}$ can be written as a function $g(C_1, \dots, C_s)$ of these 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 β that is well-defined on and satisfies $\mathbb{B}$, the proof constructs a new assignment α that is well-defined on and satisfies $\mathbb{A}$, with the additional property that all monomials in M have the same value in both α and β. Since $\mathbb{A}$ entails $\mathbb{P}$ over well-behaved assignments, $\mathbb{P}(\alpha) = 0$. As all monomials retain their values in β, 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 $\{\mathbb{P}_0, \ldots, \mathbb{P}_\tau\}$ of $BPHP^m_n$, the proof constructs a matching sequence $\{\mathbb{A}_0, \ldots, \mathbb{A}_\tau\}$ of commitment sets such that for all $t \in [\tau]$ it holds that $\mathbb{A}_t$ entails $\mathbb{P}_t$ over well-behaved assignments, and furthermore $|\mathbb{A}_t| \leq 2 \cdot Sp(\mathbb{P}_t)$. Corollary 5.5 then implies that the final configuration $\mathbb{P}_\tau$ is satisfiable, contrary to the assumption that $1 \in \mathbb{P}_\tau$.

The commitment sets $\mathbb{A}_0, \ldots, \mathbb{A}_\tau$ 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 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 the axiom $H(p_1, p_2, h)$ 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 $|\mathbb{A}_{t+1}| \leq 2 \cdot Sp(\mathbb{P}_{t+1})$, since it might be the case that $Sp(\mathbb{P}_{t+1}) = Sp(\mathbb{P}_t)$ (this cannot occur for PCR under our definition of space, although it could occur under the laxer definition of [1] that only counts distinct monomials). An application of Lemma 5.7 takes care of this issue, however.

6. A PCR space lower bound for XOR-PHP formulas. We now apply the machinery developed in section 5 to a different encoding of the pigeonhole principle, namely, a slightly obfuscated version using exclusive or and "hole indicators" to specify which pigeons are placed in which holes. In the standard pigeohole principle every pigeon has one bit for each of the n holes to encode whether that pigeon sits there. In this alternative encoding every pigeon has $n+1$ bits, and we consider a hole $j \leq n$ as occupied if there is a bit flip between positions j and $j+1$. We guarantee that the pigeon is sitting in some hole by enforcing the first and last position to be different. (To be precise, this enforces that each pigeon is assigned to an *odd* number of holes.) With this encoding, we can obtain strong PCR space lower bounds for CNF formulas of constant width.

Let us start by presenting the formal definition of this formula construction, where we recall that $[0, n) = \{0, \ldots, n-1\}$ and $[0, n] = \{0, \ldots, n\}$.

DEFINITION 6.1 (XOR PHP formula). *The* XOR pigeonhole principle formula $XPHP_n^m$ *has propositional variables* $x[i,j]$ *for each* $i \in [0,m)$ *and* $j \in [0,n]$*, where we think of* $[0,m)$ *as a set of pigeons and* $[0,n]$ *as a set of hole indicators. The formula* $XPHP_n^m$ *asserts the following:*

1. *Every pigeon gives different values to the first and last hole indicators. That is, for all* $i \in [0,m)$*, it holds that* $x[i,0] \not\equiv x[i,n]$*.*
2. *At most one pigeon is assigned to any given hole. That is, for all distinct* $i, i' \in [0,m)$ *and all* $j \in [0,n)$*, it holds that* $(x[i,j] \equiv x[i,j+1]) \vee (x[i',j] \equiv x[i',j+1])$*.*

This is written as a 4*-CNF formula consisting of* $2m$ pigeon axioms

$$\frac{x[i,0] \vee x[i,n]}{\overline{x[i,0]} \vee \overline{x[i,n]}} \tag{6.1}$$

for all $i \in [0,m)$*, encoding condition* 1*, and* $4\binom{m}{2}n$ hole axioms

$$\begin{gathered} x[i,j] \vee \overline{x[i,j+1]} \vee x[i',j] \vee \overline{x[i',j+1]} \\ \overline{x[i,j]} \vee x[i,j+1] \vee x[i',j] \vee \overline{x[i',j+1]} \\ x[i,j] \vee \overline{x[i,j+1]} \vee \overline{x[i',j]} \vee x[i',j+1] \\ \overline{x[i,j]} \vee x[i,j+1] \vee \overline{x[i',j]} \vee x[i',j+1] \end{gathered} \tag{6.2}$$

for all $i, i' \in [0,m)$*,* $i \neq i'$*, and* $j \in [0,n)$*, encoding condition* 2*.*

The formula $XPHP_n^m$ is unsatisfiable for $m > n$. To see this, notice that by the clauses (6.1), 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 hole j is *assigned* to pigeon 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 the clauses (6.2). An important observation is that for any i and j' it is possible to fix the value of $x[i,j']$ to 0 or 1 without imposing any constraints on which hole j can be assigned to pigeon i.

We will prove a PCR space lower bound for $XPHP_n^m$ by essentially the same argument as in section 5. The only real difference is that it is a little easier for us to satisfy Lemma 6.4 (the counterpart of Lemma 5.4), which leads to a space lower bound of $n/4$ rather than $n/8$.

DEFINITION 6.2 (well-behaved assignment). *Let* α *be an assignment to all the variables of* $XPHP_n^m$ *and let* $S \subseteq [0,m)$ *be a set of pigeons. We say that* α *is* well-behaved on S *if two things hold:*

1. *For each pigeon* $i \in S$*,* α *assigns exactly one hole* j *to* i *by setting* $x[i,j'] = b_i$ *for all* j' *in* $[0,j]$ *and* $x[i,j'] = 1 - b_i$ *for all* j' *in* $[j+1,n]$*, for some* $b_i \in \{0,1\}$*.*
2. *The holes assigned by* α *to the pigeons in* S *are all distinct.*

DEFINITION 6.3 (commitment). *As before, a* commitment *is a disjunction of two literals coming from two distinct pigeons. 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,* α *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* α *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 as in Definition 6.2. For one of $b_i = 0$ or $b_i = 1$ the assignment 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 Lemma 6.4. □

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 α which is well-behaved on* $\operatorname{dom}\mathbb{A}$ *it holds that if α satisfies $\mathbb{A}$, then α 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 Γ be a maximal set of monomials such that $|N(\Gamma)| \leq 2|\Gamma|$. By Hall's theorem, there exist two injective functions $f_1, f_2 \colon M \setminus \Gamma \to \mathbb{P} \setminus N(\Gamma)$ 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 α be an assignment which is well-behaved on and satisfies $\mathbb{B}$. We will define a new assignment β such that each monomial in M gets the same value in both α and β, but also such that β 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 in $\operatorname{dom}\mathbb{B}$, and let T be the set of pigeons 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 α to get an assignment β which is well-behaved on $S \cup T$, is identical to α on pigeons outside T, and satisfies X. As α and β differ only on T, all monomials in Γ get the same value in both assignments. Every other monomial in M is zeroed in both assignments. Since α and β are identical on S, β 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). *$Sp_{\mathcal{PCR}}(XPHP^m_n \vdash \bot) > n/4$.*

Proof. The proof of this theorem is virtually identical to that of Theorem 5.8. In order to derive a contradiction, suppose that $\pi = \{\mathbb{P}_0, \ldots, \mathbb{P}_\tau\}$ is a PCR refutation of $XPHP^m_n$ in space at most $n/4$. We construct inductively a sequence of commitment sets $\mathbb{A}_0, \ldots, \mathbb{A}_\tau$ such that at each time step 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 $\mathbb{P}_\tau$.

For $\mathbb{P}_0 = \emptyset$ we define $\mathbb{A}_0 = \emptyset$ to be the empty commitment. Suppose now that we have constructed $\mathbb{A}_t$. To construct $\mathbb{A}_{t+1}$ we consider three cases, depending on which rule 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 α be any assignment well-behaved on and satisfying $\mathbb{A}_{t+1}$. Then α satisfies $\mathbb{P}_t$ by the inductive hypothesis, and must also satisfy 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 $\mathbb{A}_{t+1}$. Specifically, we find a new pigeon i' which is used neither in D nor in $\mathbb{A}_t$, and we set C_i to be $x[i,0] \vee x[i',0]$. We are able to find such i' for both commitments since $|\mathbb{P}_t| \leq |\mathbb{P}_{t+1}| - 1 \leq \frac{n}{4} - 1$. Thus the total number of pigeons used is at most $2|\mathbb{A}_t| + 2 \leq 4|\mathbb{P}_t| + 2 \leq n - 2$. 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 α which is well-behaved on and satisfies $\mathbb{A}_{t+1}$ satisfies $\mathbb{P}_t$ by the induction hypothesis. Since P semantically follows from $\mathbb{P}_t$, α 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, as can be established by reasoning completely analogous to that at the end of section 5.

7. Space complexity of 3-CNF versions of wide CNF formulas. In this paper we have proved the first monomial space lower bound on k-CNF formulas for constant k. While the lower bounds for PC in section 4 apply to 3-CNF formulas, we can only get our techniques for PCR in section 6 to work for $k \geq 4$, however. This seems to be a real technical barrier for this approach, since even the lower bounds for random k-CNF formulas in [19] and for Tseitin k-CNF formulas on random k-regular graphs in [35] only work for $k \geq 4$.

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?

Although the question of 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* formulas, 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 $FPHP_n^m$ (the definition of which is given below for completeness). In particular, 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). *We say that a CNF formula F is* weight-constrained *if for each clause $a_1 \vee a_2 \vee \cdots \vee a_\ell$ in F with $\ell \geq 4$, F also contains clauses $\overline{a}_i \vee \overline{a}_j$ for all $1 \leq i < j \leq \ell$.*

Thus, a weight-constrained 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 $FPHP_n^m$ mentioned above, which is similar to the regular pigeonhole principle PHP_n^m 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 $FPHP_n^m$ is a CNF formula over variables $\{p_{i,j} \mid i \in [m], j \in [n]\}$ consisting of the following clauses:*

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

Recall the definition of the extended version of a CNF formula F given in Definition 4.2. The first result in this section says that for any such weight-constrained 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 $\widetilde{F}$ be the extended 3-CNF version of a weight-constrained CNF formula F. Then for resolution clause space it holds that $Sp_{\mathcal{R}}(F \vdash \bot) = Sp_{\mathcal{R}}\big(\widetilde{F} \vdash \bot\big) + \mathrm{O}(1)$.*

Proof. We clearly have $Sp_{\mathcal{R}}\big(\widetilde{F} \vdash \bot\big) \leq Sp_{\mathcal{R}}(F \vdash \bot) + \mathrm{O}(1)$, since any axiom of F can be rederived from $\widetilde{F}$ in space at most 3 and plugged into a refutation of F.

In the other direction, let $\pi = \{\mathbb{C}_1, \ldots, \mathbb{C}_\tau\}$ be a resolution refutation of $\widetilde{F}$. Our goal is to find a resolution refutation of F which uses a constant amount of additional space. To this end, we will substitute all literals defined on extension variables with clauses defined on original variables.

Let $C = a_1 \vee a_2 \vee \cdots \vee a_\ell$ be a clause in F, and let $\{y_0\} \cup \{\overline{y}_{i-1} \vee a_i \vee y_i \mid 1 \leq i \leq \ell\} \cup \{\overline{y}_\ell\}$ be its extended version. Intuitively, the positive extension literal y_i encodes that at least one of the literals $a_{i+1}, \ldots a_\ell$ is true, while the negative extension literal $\overline{y}_i$ encodes that at least one of the variables $a_1, \ldots, a_i$ is true. Our substitution is based on this intuition. More precisely, we substitute literals y_i and $\overline{y}_i$ as follows:

$$y_i \mapsto \textstyle\bigvee_{t=i+1}^{\ell} a_t\,; \quad \overline{y}_i \mapsto \textstyle\bigvee_{t=1}^{i} a_t. \tag{7.1}$$

Note that, in particular, this results in the substitutions $y_\ell \mapsto \bot$ and $\overline{y}_0 \mapsto \bot$.

Let $\pi' = \{\mathbb{C}'_1, \ldots, \mathbb{C}'_\tau\}$ be the sequence of configurations obtained from π by applying the above substitution to all clauses. We claim that π' can be turned into a syntactic resolution refutation of F without significant increase in space. Clearly, the space complexity of π' is the same as the space complexity of π. We need to show that all transitions from $\mathbb{C}'_i$ to $\mathbb{C}'_{i+1}$ can be carried out syntactically using a constant amount of additional space. We split the proof into cases according to the way in which $\mathbb{C}_{i+1}$ was derived from $\mathbb{C}_i$ in the original refutation π.

Erasure. In this case the transition from $\mathbb{C}'_i$ to $\mathbb{C}'_{i+1}$ is an application of erasure, too.

Axiom download. Suppose that $\mathbb{C}_{i+1}$ was obtained from $\mathbb{C}_i$ by downloading an axiom $A \in \widetilde{F}$. It is easily verified that in this case the substitution turns A into some $C \in F$. In the latter case the transition from $\mathbb{C}'_i$ to $\mathbb{C}'_{i+1}$ downloads the axiom C and the space complexity does not increase.

Inference rules. If $\mathbb{C}_{i+1}$ was obtained from $\mathbb{C}_i$ by resolving two clauses over one of the original variables, then $\mathbb{C}'_{i+1}$ can be obtained from $\mathbb{C}'_i$ by resolving the corresponding clauses over the same variable. Suppose instead that $\mathbb{C}_{i+1}$ was obtained from $\mathbb{C}_i$ by resolving $A \vee y_i$ and $B \vee \overline{y}_i$ on the extension variable y_i. Let A' and B' be the clauses obtained from A and B, respectively, after the substitution in (7.1). We want to show that $A' \vee B'$ can be obtained from $A' \vee \bigvee_{t=i+1}^{\ell} a_i$ and $B' \vee \bigvee_{t=1}^{i} a_i$ in constant clause space. The variable y_i comes from extending the clause $C = a_1 \vee a_2 \cdots \vee a_\ell$, and since F is weight-constrained we have in F all clauses $\overline{a}_i \vee \overline{a}_j$ for $1 \leq i < j \leq \ell$. We can therefore appeal to Claim 4.8 to derive the desired conclusion. The theorem follows. □

Our second result in this section is an analogue of the above theorem for PCR, although here we get a multiplicative rather than additive increase in the space.

THEOREM 7.4 (Theorem 1.5 for PCR). *Let $\widetilde{F}$ be the extended version of a weight-constrained CNF formula F. Then for PCR it holds that $Sp_{\mathcal{PCR}}(F \vdash \bot) = \Theta\big(Sp_{\mathcal{PCR}}\big(\widetilde{F} \vdash \bot\big)\big)$.*

Proof. The proof is similar to the proof of Theorem 7.3 except that we now define a substitution of extension variables with products of variables instead of clauses. As for resolution, the inequality $Sp_{\mathcal{PCR}}\big(\widetilde{F} \vdash \bot\big) \leq Sp_{\mathcal{PCR}}(F \vdash \bot) + \mathrm{O}(1)$ is immediate.

To prove the other direction, suppose that $C = a_1 \vee a_2 \vee \cdots \vee a_\ell$ is a clause of F and let us write $\{y_0\} \cup \{\overline{y}_{j-1} \vee a_j \vee y_j \mid 1 \leq j \leq \ell\} \cup \{\overline{y}_\ell\}$ to denote its extended version. For y_i and $\overline{y}_i$ (which are now both formal and distinct variables) we substitute

$$y_i \mapsto \textstyle\prod_{t=i+1}^{\ell} a_t\,; \quad \overline{y}_i \mapsto \prod_{t=1}^{i} a_t, \tag{7.2}$$

where we note that in analogy with (7.1) we obtain $y_\ell \mapsto 1$ and $\overline{y}_0 \mapsto 1$.

Let $\pi = \{\mathbb{C}_1, \ldots, \mathbb{C}_\tau\}$ be a PCR refutation of $\widetilde{F}$ and let $\pi' = \{\mathbb{C}'_1, \ldots, \mathbb{C}'_\tau\}$ be the sequence of configurations obtained from π by applying the above substitution to all polynomials in π. Then the monomial space of π' is equal to that of π, and all that needs to be done is to show how to make space-efficient PCR derivations of $\mathbb{C}'_{i+1}$ from $\mathbb{C}'_i$. We split the proof into cases according to the way in which $\mathbb{C}_{i+1}$ was derived from $\mathbb{C}_i$ in the original refutation π.

Erasure. In this case the transition from $\mathbb{C}'_i$ to $\mathbb{C}'_{i+1}$ is an application of erasure, too.

Axiom download of a clause in $\widetilde{F}$. 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 the substitution turns the PCR translation of A into the PCR translation of some $C \in F$. In the latter case the transition from $\mathbb{C}'_i$ to $\mathbb{C}'_{i+1}$ is a download of the PCR encoding of an axiom C and the space complexity does not increase.

Inference rules. If $\mathbb{C}_{i+1}$ follows from $\mathbb{C}_i$ either via the addition rule or via multiplication by an original variable, then $\mathbb{C}'_{i+1}$ follows 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, the transition from $\mathbb{C}_i$ to $\mathbb{C}_{i+1}$ is the sequence of multiplications by original variables $a_{i+1}, a_{i+2}, \ldots, a_\ell$. To do that efficiently we only keep in memory one additional intermediate polynomial of the form $q \cdot \prod_{t=i+1}^{r} a_t$, $i+1 \leq r < \ell$. Since the configuration $\mathbb{C}_{i+1}$ contains both polynomials q and qy_i and has space at most s, the intermediate polynomials $q \cdot \prod_{t=i+1}^{r} a_t$ have at most $s/2$ monomials each, and the monomial complexity increases by a factor of at most $3/2$. The case of multiplication by $\overline{y}_i$ is very similar.

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 an original variable, then $\mathbb{C}'_{i+1}$ can be obtained from $\mathbb{C}'_i$ by downloading the same axiom. Consider instead the logical axiom $\overline{y}_i^2 - \overline{y}_i$ for an extension variable y_i. The corresponding polynomial after substitution is $(a_1 \cdots a_i)^2 - (a_1 \cdots a_i)$, and we deduce it as follows. For each $t \in [i]$ we download $a_t^2 - a_t$ and multiply to obtain

$$(a_1 \cdots a_{t-1})a_t^2(a_{t+1}^2 \cdots a_i^2) - (a_1 \cdots a_{t-1})a_t(a_{t+1}^2 \cdots a_t^2) \ . \tag{7.3}$$

Summing polynomials of the form (7.3) for all $t \in [i]$ yields $(a_1 \cdots a_i)^2 - (a_1 \cdots a_i)$, and this calculation can be performed in constant monomial space. Logical axioms $y_i^2 - y_i$ can be dealt with in the same way as $\overline{y}_i^2 - \overline{y}_i$.

It remains to consider complementarity axioms $y_i + \overline{y}_i - 1$. Here we need to show how to construct a small-space derivation of

$$\textstyle\prod_{t=1}^{i} a_t + \prod_{t'=i+1}^{\ell} a_{t'} - 1 = q + q' - 1, \tag{7.4}$$

where we denote $q = \prod_{t=1}^{i} a_t$ and $q' = \prod_{t'=i+1}^{\ell} a_{t'}$. First, we observe that using Claim 4.8 we can get a resolution refutation of the set of clauses

$$\textstyle\left\{\bigvee_{t=1}^{i} a_t, \bigvee_{t'=i+1}^{\ell} a_{t'}\right\} \cup \left\{\overline{a}_t \vee \overline{a}_{t'} \middle| 1 \le t \le i < t' \le \ell\right\} \tag{7.5}$$

in constant space. PCR simulates resolution efficiently, so we can use essentially the same derivation to deduce 1 from the polynomials q, q', and $\overline{a}_t\overline{a}_{t'}$ in constant monomial space. We use this derivation as a template to build the actual derivation of $q+q'-1$. To do that we multiply every line of the derivation by $(q-1)(q'-1)$, which gives a constant space derivation of $(q-1)(q'-1)$ from some premises $q(q-1)(q'-1)$, $q'(q-1)(q'-1)$, and $\overline{a}_t\overline{a}_{t'}(q-1)(q'-1)$ for $1 \le t \le i < t' \le \ell$. With $(q-1)(q'-1)$ in memory we conclude the derivation by downloading the axiom $qq' = \prod_{t=1}^{\ell} a_t$ and by computing the difference $qq' - (q-1)(q'-1)$ which is the desired polynomial $q+q'-1$. The only missing part is how to deduce in small space the premises $q(q-1)(q'-1)$, $q'(q-1)(q'-1)$ and $\overline{a}_t\overline{a}_{t'}(q-1)(q'-1)$ for $1 \le t \le i < t' \le \ell$. The first two premises have $q^2 - q$ and $(q')^2 - q'$, respectively, as factors, so they are provable from logical axioms in the same way as above. All the other polynomials have $\overline{a}_t\overline{a}_{t'}$ as a factor, and this factor is the PCR encoding of some initial clause of F because the formula is weight-constrained. Hence, these formulas can be derived in constant space from these initial clauses. The theorem follows. □

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 longstanding open question from [1]. We also establish nontrivial upper bounds for proof size and space in PC, showing that for CNF formulas of constant width the worst-case behavior 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.

When the conference version [36] of this paper first appeared, the space complexity measure in PC and PCR was still very poorly understood, and our concluding remarks

gave a long list of open problems. The last few years have seen quite dramatic developments in this area, as described in section 1.3, and many of the open questions we originally listed have now been resolved, fully or partially. There are still quite a few natural problems that have remained out of reach, however, and we conclude this paper by briefly discussing some such problems which we believe merit further study.

1. Show lower bounds on PCR space for Tseitin formulas over expanders and for FPHP formulas (or even extended versions of the PHP formulas, which might be a strictly easier problem in view of the results in section 7). The refined methods in [19, 20] still do not seem strong enough to deal with these formulas, and while the paper [35] has some partial results for Tseitin formulas, it fails to prove lower bounds in terms of graph expansion only. It seems that we still do not have quite the right techniques to analyze PCR space, and indeed it would be interesting to see if progress on space lower bounds for the above formulas could also lead to optimal lower bounds for random 3-CNF formulas, eliminating the logarithmic factor loss in [20].
2. Prove (or refute) that degree is a lower bound on monomial space in PCR (in analogy with the result for resolution shown in [3]). Again, there are some partial results in this direction in [35]—namely, that if a formula requires large degree, then the so-called XORified version of the formula requires large monomial space—but the original problem remains wide open. If it could be proved that degree is a lower bound for space in PCR, then space lower bounds for FPHP formulas would follow from the recent results in [44].
3. Separate size and space in PCR by proving that there are k-CNF formulas that have small PCR refutations but require large PCR space for any characteristic of the underlying field. A separation of size and space in PCR was established in [35], but the formulas used there depend on the field characteristic. A natural candidate family of formulas for this are the formulas used in [14] to separate size and space in resolution.
4. Separate resolution from PCR with respect to space. That is, show that there are families of CNF formulas (preferably of constant width) for which the monomial space complexity in PCR grows asymptotically slower than the clause space complexity in resolution (or show that no such separation exists).
5. Prove nontrivial lower (or upper) bounds on space for cutting planes with coefficients of polynomial size. As discussed in section 1.3, it was shown in [37] that cutting planes can refute any CNF formula in constant line space, but the coefficients in such refutations have exponential size.
6. Show size-space trade-offs for cutting planes with coefficients of polynomial size. By [37], the results in [40, 38] imply size-space trade-offs for cutting planes, but the space-efficient proofs again have coefficients of exponential size. It would be interesting to obtain trade-off results where the upper bounds are for coefficients of polynomial size, or even constant size. This might also be relevant from an applied perspective, since a natural heuristic when trying to implement SAT solvers based on cutting planes (also referred to as *pseudo-Boolean solvers*) is to try to keep the size of the coefficients down.

Acknowledgments. This article is the result of a long process, and various subsets of the authors would like to acknowledge useful discussions they 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 (and to the anonymous reviewers) for carefully reading this manuscript and finding numerous typos and other mistakes that needed to be fixed. Needless to say, any remaining errors 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 Prague, which took place during the autumn of 2011 and was supported by the Marie Curie Initial Training Network MALOA (Mathematical Logic and Applications).

Any opinions, finding, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or other funding agencies.

REFERENCES

[1] M. Alekhnovich, E. Ben-Sasson, A. A. Razborov, and A. Wigderson, *Space complexity in propositional calculus*, SIAM J. Comput., 31 (2002), pp. 1184–1211.

[2] M. Alekhnovich and A. A. Razborov, *Lower bounds for polynomial calculus: Non-binomial case*, Proc. Steklov Inst. Math., 242 (2003), pp. 18–35; also available online from http://people.cs.uchicago.edu/~razborov/files/misha.pdf.

[3] A. Atserias and V. Dalmau, *A combinatorial characterization of resolution width*, J. Comput. System Sci., 74 (2008), pp. 323–334.

[4] A. Atserias, J. K. Fichte, and M. Thurley, *Clause-learning algorithms with many restarts and bounded-width resolution*, J. Artificial Intelligence Res., 40 (2011), pp. 353–373.

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

[6] P. Beame, *Proof complexity*, in Computational Complexity Theory, S. Rudich and A. Wigderson, eds., IAS/Park City Math. Ser. 10, AMS, Providence, RI, 2004, pp. 199–246.

[7] P. Beame, C. Beck, and R. Impagliazzo, *Time-space tradeoffs in resolution: Superpolynomial lower bounds for superlinear space*, in Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC '12), 2012, pp. 213–232.

[8] P. Beame, R. Karp, T. Pitassi, and M. Saks, *The efficiency of resolution and Davis-Putnam procedures*, SIAM J. Comput., 31 (2002), pp. 1048–1075.

[9] C. Beck, J. Nordström, and B. Tang, *Some trade-off results for polynomial calculus*, in Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC '13), 2013, pp. 813–822.

[10] E. Ben-Sasson, *Size space tradeoffs for resolution*, SIAM J. Comput., 38 (2009), pp. 2511–2525.

[11] E. Ben-Sasson and N. Galesi, *Space complexity of random formulae in resolution*, Random Structures Algorithms, 23 (2003), pp. 92–109.

[12] E. Ben-Sasson and R. Impagliazzo, *Random CNF's are hard for the polynomial calculus*, Comput. Complexity, 19 (2010), pp. 501–519.

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

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

[15] E. Ben-Sasson and J. 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), 2011, pp. 401–416.

[16] E. Ben-Sasson and A. Wigderson, *Short proofs are narrow—resolution made simple*, J. ACM, 48 (2001), pp. 149–169.

[17] A. Biere, M. J. H. Heule, H. van Maaren, and T. Walsh, Eds., *Handbook of Satisfiability*, Frontiers Artificial Intelligence Appl. 185, IOS Press, Amsterdam, 2009.

[18] A. Blake, *Canonical Expressions in Boolean Algebra*, Ph.D. thesis, University of Chicago, 1937.

[19] I. Bonacina and N. Galesi, *Pseudo-partitions, transversality and locality: A combinatorial characterization for the space measure in algebraic proof systems*, in Proceedings of the 4th Conference on Innovations in Theoretical Computer Science (ITCS '13), 2013, pp. 455–472.

[20] I. Bonacina, N. Galesi, T. Huynh, and P. Wollan, *Space proof complexity for random 3-CNFs via a $(2-\epsilon)$-Hall's theorem*, Technical report TR14-146, Electronic Colloquium on Computational Complexity (ECCC), 2014.

[21] I. Bonacina, N. Galesi, and N. Thapen, *Total space in resolution*, in Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS '14), 2014, pp. 641–650.

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

[23] M. Brickenstein and A. Dreyer, *PolyBoRi: A framework for Gröbner-basis computations with Boolean polynomials*, J. Symbolic Comput., 44 (2009), pp. 1326–1345.

[24] M. Brickenstein, A. Dreyer, G.-M. Greuel, M. Wedler, and O. Wienand, *New developments in the theory of Gröbner bases and applications to formal verification*, J. Pure Appl. Algebra, 213 (2009), pp. 1612–1635.

[25] S. R. Buss, D. Grigoriev, R. Impagliazzo, and T. Pitassi, *Linear gaps between degrees for the polynomial calculus modulo distinct primes*, J. Comput. System Sci., 62 (2001), pp. 267–289.

[26] S. R. Buss, J. Hoffmann, and J. Johannsen, *Resolution trees with lemmas: Resolution refinements that characterize DLL-algorithms with clause learning*, Log. Methods Comput. Sci., 4 (2008).

[27] S. R. Buss and T. Pitassi, *Resolution and the weak pigeonhole principle*, in Proceedings of the 11th International Workshop on Computer Science Logic (CSL '97), Lecture Notes in Comput. Sci. 1414, Springer, New York, 1998, pp. 149–156.

[28] V. Chvátal and E. Szemerédi, *Many hard examples for resolution*, J. ACM, 35 (1988), pp. 759–768.

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

[30] S. A. Cook and R. Reckhow, *The relative efficiency of propositional proof systems*, J. Symbolic Logic, 44 (1979), pp. 36–50.

[31] W. Cook, C. R. Coullard, and G. Turán, *On the complexity of cutting-plane proofs*, Discrete Appl. Math., 18 (1987), pp. 25–38.

[32] M. Davis, G. Logemann, and D. Loveland, *A machine program for theorem proving*, Commun. ACM, 5 (1962), pp. 394–397.

[33] M. Davis and H. Putnam, *A computing procedure for quantification theory*, J. ACM, 7 (1960), pp. 201–215.

[34] J. L. Esteban and J. Torán, *Space bounds for resolution*, Inform. and Comput., 171 (2001), pp. 84–97.

[35] Y. Filmus, M. Lauria, M. Mikša, J. Nordström, and M. Vinyals, *Towards an understanding of polynomial calculus: New separations and lower bounds (extended abstract)*, in Proceedings of the 40th International Colloquium on Automata, Languages and Programming (ICALP '13), Lecture Notes in Comput. Sci. 7965, Springer, New York, 2013, pp. 437–448.

[36] Y. Filmus, M. Lauria, J. Nordström, N. Thapen, and N. Ron-Zewi, *Space complexity in polynomial calculus*, in Proceedings of the 27th Annual IEEE Conference on Computational Complexity (CCC '12), 2012, pp. 334–344.

[37] N. Galesi, P. Pudlák, and N. Thapen, *The space complexity of cutting planes refutations*, in Proceedings of the 30th Annual Computational Complexity Conference (CCC '15), 2015.

[38] M. Göös and T. Pitassi, *Communication lower bounds via critical block sensitivity*, in Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC '14), 2014, pp. 847–856.

[39] A. Haken, *The intractability of resolution*, Theoret. Comput. Sci., 39 (1985), pp. 297–308.

[40] T. Huynh and J. Nordström, *On the virtue of succinct proofs: Amplifying communication complexity hardness to time-space trade-offs in proof complexity*, in Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC '12), 2012, pp. 233–248.

[41] R. Impagliazzo, P. Pudlák, and J. Sgall, *Lower bounds for the polynomial calculus and the Gröbner basis algorithm*, Comput. Complexity, 8 (1999), pp. 127–144.

[42] M. Järvisalo, A. Matsliah, J. Nordström, and S. 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), Lecture Notes in Comput. Sci. 7514, Springer, New York, 2012, pp. 316–331.

[43] J. P. Marques-Silva and K. A. Sakallah, *GRASP: A search algorithm for propositional satisfiability*, IEEE Trans. Comput., 48 (1999), pp. 506–521.

[44] M. Mikša and J. Nordström, *A generalized method for proving polynomial calculus degree lower bounds*, in Proceedings of the 30th Annual Computational Complexity Conference (CCC '15), 2015.

[45] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik, *Chaff: Engineering an efficient SAT solver*, in Proceedings of the 38th Design Automation Conference (DAC '01), 2001, pp. 530–535.

[46] J. Nordström, *Narrow proofs may be spacious: Separating space and width in resolution*, SIAM J. Comput., 39 (2009), pp. 59–121.

[47] J. Nordström, *A simplified way of proving trade-off results for resolution*, Inform. Process. Lett., 109 (2009), pp. 1030–1035.

[48] J. Nordström, *Pebble games, proof complexity and time-space trade-offs*, Log. Methods Comput. Sci., 9 (2013), pp. 15:1–15:63.

[49] J. Nordström and J. Håstad, *Towards an optimal separation of space and length in resolution*, Theory Comput., 9 (2013), pp. 471–557.

[50] K. Pipatsrisawat and A. Darwiche, *On the power of clause-learning SAT solvers as resolution engines*, Artificial Intelligence, 175 (2011), pp. 512–525.

[51] P. Pudlák, *Lower bounds for resolution and cutting plane proofs and monotone computations*, J. Symbolic Logic, 62 (1997), pp. 981–998.

[52] A. A. Razborov, *Lower bounds for the polynomial calculus*, Comput. Complexity, 7 (1998), pp. 291–324.

[53] J. A. Robinson, *A machine-oriented logic based on the resolution principle*, J. ACM, 12 (1965), pp. 23–41.

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

[55] N. Segerlind, *The complexity of propositional proofs*, Bull. Symbolic Logic, 13 (2007), pp. 482–537.

[56] A. Urquhart, *Hard examples for resolution*, J. ACM, 34 (1987), pp. 209–219.