On the Virtue of Succinct Proofs: Amplifying Communication Complexity Hardness to Time-Space Trade-offs in Proof Complexity

[Extended Abstract]

Trinh Huynh
Swiss Federal Institute of Technology (ETH)
8092 Zurich, Switzerland
trinh@inf.ethz.ch

Jakob Nordström
KTH Royal Institute of Technology
100 44 Stockholm, Sweden
jakobn@kth.se

ABSTRACT

An active line of research in proof complexity over the last decade has been the study of proof space and trade-offs between size and space. Such questions were originally motivated by practical SAT solving, but have also led to the development of new theoretical concepts in proof complexity of intrinsic interest and to results establishing nontrivial relations between space and other proof complexity measures.

By now, the resolution proof system is fairly well understood in this regard, as witnessed by a sequence of papers leading up to [Ben-Sasson and Nordström 2008, 2011] and [Beame, Beck, and Impagliazzo 2012]. However, for other relevant proof systems in the context of SAT solving, such as polynomial calculus (PC) and cutting planes (CP), very little has been known.

Inspired by [BN08, BN11], we consider CNF encodings of so-called pebble games played on graphs and the approach of making such pebbling formulas harder by simple syntactic modifications. We use this paradigm of hardness amplification to make progress on the relatively longstanding open question of proving time-space trade-offs for PC and CP. Namely, we exhibit a family of modified pebbling formulas $\{F_n\}_{n=1}^{\infty}$ such that:

- The formulas F_n have size $\Theta(n)$ and width $\mathrm{O}(1)$.
- They have proofs in length $\mathrm{O}(n)$ in resolution, which generalize to both PC and CP.
- Any refutation in CP or PCR (a generalization of PC) in length L and space s must satisfy $s \log L \gtrapprox \sqrt[4]{n}$.

A crucial technical ingredient in these results is a new two-player communication complexity lower bound for composed search problems in terms of block sensitivity, a contribution that we believe to be of independent interest.

Categories and Subject Descriptors: F.2.3[Analysis of Algorithms and Problem Complexity]: Tradeoffs among Complexity Measures; F.1.3[Computation by Abstract Devices]: Complexity Measures and Classes —*Relations among complexity measures*; I.2.3[Artificial Intelligence]: Deduction and Theorem Proving; F.4.1[Mathematical Logic and Formal Languages]: Mathematical Logic —*computational logic*

General Terms: Theory

Keywords: Proof complexity, resolution, polynomial calculus, PCR, cutting planes, size, length, space, trade-offs

1. INTRODUCTION

Ever since Cook's landmark paper [29], the question of how hard it is to prove formulas in propositional logic has been a flagship problem in theoretical computer science. While this problem is generally believed to be intractable in the worst case, impressive algorithmic developments in the last ten to fifteen years have led to so-called SAT solvers that can deal with real-world problem instances with millions of variables. A somewhat surprising aspect of this is that at the core, state-of-the-art SAT solvers today are still based on the fairly simple Davis-Putnam-Logemann-Loveland (DPLL) procedure [34, 35] from the early 1960s augmented with clause learning [6, 48]; these programs are also known as conflict-driven clause learning (CDCL) solvers. Despite the fact that such solvers in effect search for proofs in the relatively weak *resolution proof system*, for which numerous exponential lower bounds are known, they have dominated the international SAT competition [61] in recent years.

The field of proof complexity, initiated by Cook and Reckhow [30], also considers the hardness of proving propositional logic formulas, but from the slightly different (nonconstructive) angle of asking how succinct such proofs can be, regardless of how hard or easy it is to find them. In its most general form, a proof system for a language L is defined as a predicate $P(x, \pi)$, computable in time polynomial in $|x|$ and $|\pi|$, such that for all $x \in L$ there is a string π (a *proof*) such that P accepts the input (x, π), whereas for any $x \notin L$ it holds for all π that P rejects (x, π). 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 $|x|$. A *propositional proof system* is a proof system for the language of tautologies in propositional logic.

One important motivation for the study of proof complexity is the problem of P vs. NP, in that [30] showed that one way of establishing $\mathsf{co\text{-}NP} \neq \mathsf{NP}$, and hence $\mathsf{P} \neq \mathsf{NP}$, would be to prove that there are no polynomially bounded propositional proof systems. This goal remains very distant, however, and it is probably fair to say that most current work in proof complexity is primarily driven by other concerns.

One such concern is the connection to the SAT problem and practical SAT solving. 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 [4, 15, 55] being a very subjective pick of just three recent interesting examples.

The subject matter of the current paper, namely time-space trade-offs in proof complexity, should be understood as being mainly motivated by this latter reason. The two main bottlenecks for modern SAT solvers are running time and memory usage. By studying proof size/length, 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]

Concluding this brief general discussion, let us mention that some good starting points for a further study of proof complexity in general are [8, 62], while the upcoming survey [53] by the second author focuses specifically on time-space trade-offs and related questions. Also, a recent, very compehensive, reference on the SAT problem in practice is [19].

1.1 Previous Work

Any formula in propositional logic can be converted to a CNF formula that is only linearly larger and is unsatisfiable if and only if the original formula is a tautology. Therefore, any sound and complete system for refuting unsatisfiable CNF formulas can be considered as a general propositional proof system. All proof systems considered in this paper are of this type.

The *resolution* proof system already mentioned above was introduced in [20] and began to be investigated in connection with automated theorem proving in the 1960s [34, 35, 60]. In resolution, one derives new disjunctive clauses from the clauses of the original CNF formula until a contradiction is reached. For this proof system, it is most straightforward to prove bounds on the *length* of refutations, i.e., the number of clauses, rather than on the total size (the two measures are easily seen to be polynomially related). One of the early break-throughs in proof complexity was the result by Haken [39] that CNF formulas encoding the pigeonhole principle (PHP formulas) require proofs of exponential length. There have been a sequence of follow-up papers establishing quantitatively stronger bounds for other formula families in, for instance, [11, 18, 27, 64].

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 proof space in resolution was initiated by Esteban and Torán in [37] and was later extended to a more general setting by Alekhnovich et al. in [1]. Intuitively, the (clause) space of a resolution refutation is the maximal number of clauses one needs to keep in memory while verifying the refutation. Perhaps somewhat surprisingly, it turns out that one need never use more than linear clause space, and a sequence of papers [1, 14, 37] have proven matching linear lower bounds.

Another sequence of papers [50, 54, 16] involving the second author clarified the relation between length and space, showing that hardness with respect to space is "maximally uncorrelated" with hardness with respect to length. More formally, while it follows from [3] that small space complexity implies the existence of short resolution refutations, it was established in [16] that there exist explicit formulas that are maximally easy with respect to length, having refutations in length $\mathrm{O}(n)$, 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 [13, 51] and strong trade-offs for full, unrestricted resolution were finally obtained in [17] and even more recently in [9] for a different range of parameters. In this context it should also be mentioned that for the more general k-DNF resolution proof systems [43], there have also been shown strong lower bounds on length and space as well as length-space trade-offs in, for instance, [17, 36, 63].

In *polynomial calculus (PC)* [28], clauses are interpreted as multilinear polynomial equations over some field, which in this paper we always take to be finite, and an unsatisfiable CNF formula is refuted by showing that there is no common root for the polynomials corresponding to 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, 41, 59].

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 cleaner, more symmetric theoretical treatment of space, [1] introduced *polynomial calculus resolution (PCR)* as a common extension of polynomial calculus and resolution. Briefly, in PCR one adds an extra set of parallel formal variables $x', y', z', \ldots$ as well as axioms specifing that x and x' must always take opposite signs. In this way, negated and unnegated literals can both be represented in a space-efficient fashion. In this stronger system, [1] established nontrivial lower bounds on space measured as the number of monomials, but only for formulas of unbounded width (namely, PHP formulas). It was only very recently that [38] showed space lower bounds for formulas of constant width, and there is still quite a large gap between the best known worst-case lower and upper bounds even for PC. To the best of our knowledge, there have not been any trade-off results shown for PC or PCR.

Initially, 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 [24], but in general they seem to be an order of magnitude slower than state-of-the-art CDCL solvers (although [25] reports that PolyBoRi can be faster on certain industrial instances).

[1] And this should not necessarily be considered to be just wishful thinking—as a case in point, experimental work reported in [42] seems to indicate that theoretical space complexity in resolution is correlated with practical hardness for CDCL solvers.

The *cutting planes* proof system, or *CP* for short, was introduced in [32]. In a CP-refutation clauses of a formula are translated to linear inequalities, or hyperplanes, and the formula is refuted by showing that the polytope defined by these hyperplanes does not have any zero-one integer points (corresponding to satisfying assignments). Cutting planes is an intriguing proof system in that we only know of one superpolynomial lower bound on length [57] (improving on a bound [21] in a somewhat restricted setting). It is natural to define the *line space* of a CP-refutation to be the maximal number of linear inequalities that need to be kept in memory simultaneously during the refutation. Just as for monomial space in PCR, line space in CP is easily seen to be a generalization of clause space in resolution and is hence upper bounded by the clause space complexity. As far as we are aware, no space lower bounds are known for cutting planes.

Cutting planes is known to be exponentially stronger than resolution, and so it would be very interesting if efficient CP-based SAT solvers could be built. But although there do exist some SAT solvers that use cutting planes (see, for instance, [45, 46] and references therein) again it seems they have a hard time competing with resolution-based solvers.

1.2 Our Results

As discussed above, there is a wealth of results on space lower bounds and time-space trade-offs for resolution and even k-DNF resolution, and it is probably fair to say that these are by now reasonably well understood proof systems. When it comes to polynomial calculus, polynomial calculus resolution, and cutting planes, the situation is very different. As far as we are aware no trade-offs have been known. Also, for cutting planes there seem to exist no space lower bounds whatsoever, and for polynomial calculus/PCR the current state of knowledge regarding space lower bounds is quite limited as well.

In this paper, we report length-space trade-off results that apply for both polynomial calculus resolution and cutting planes (and that concern formulas of bounded width).

Theorem 1. *There are k-CNF formulas $\{F_n\}_{n=1}^{\infty}$ of size $\Theta(n)$ that can be refuted in length $\mathrm{O}(n)$ in resolution, polynomial calculus (and hence also PCR) and Cutting Planes, but for which the following holds:*

- *Any PCR-refutation of F_n in length L and monomial space s must satisfy $s \log L = \Omega\bigl(\sqrt[4]{n}\bigr)$.*
- *Any CP-refutation of F_n in length L and line space s must satisfy $s \log L \log(s \log L) = \Omega\left(\sqrt[4]{n} / \log^2 n\right)$.*

The formulas used in Theorem 1 are a particular flavour of so-called pebbling contradictions, which is a well-studied family of CNF formulas in proof complexity. A nice technical feature of the results that is worth pointing out is that the bounds are actually slightly stronger than stated above—they hold also for *semantic* versions of the proof systems (as defined in [1]), where anything that is implied by the current memory configuration can be derived immediately in one step regardless of the syntactic rules (which makes the systems exponentially stronger with respect to length).

The trade-offs in Theorem 1 are obtained by studying the concept of *critical block sensitivity* of search problems, which is a new notion generalizing the well-studied block sensitivity of (Boolean) functions and which will be formally defined in Section 4. We prove that lower bounds on the critical block sensitivity of the search problem associated with a CNF formula—i.e., given an assignment, finding a clause falsified by it—can be amplified to lower bounds on length-space trade-offs in PCR and CP (and also to lower bound on tree-like length in CP). Our techniques are inspired by the joint work of the first author with Beame and Pitassi [10], but also crucially use a new result in communication complexity that we believe should merit independent interest. Namely, we obtain the first strong randomized communication complexity lower bounds for *lifts of search problems* in terms of critical block sensitivity. We refer to Section 3, where we define lifting of search problems and *consistent* randomized communication protocols, for a more precise statement of the next theorem. Informally, however, let us just say for now that the communication problem of solving the *lift* of a given search problem on m input bits is defined by giving Alice an array of $n > m$ bits and Bob an m-subset of $\{1, \ldots, n\}$ and asking them to solve the search problem in question on m particular bits of Alice's input as selected by Bob's index set.

Theorem 2 (Informal). *If $S \subseteq \{0,1\}^m \times Q$ is a search problem over input domain $\{0,1\}^m$ and output range Q and the* critical block sensitivity *of S is at least s, then any* consistent *randomized (and hence also any deterministic) two-party protocol solving $G = \mathit{Lift}(S)$ requires $\Omega(s)$ bits of communication.*

Returning to our proof complexity trade-offs in Theorem 1, we want to mention that interestingly, and intriguingly, it is not quite clear whether this theorem provides trade-offs in a strict sense. The issue is that to be able to speak about a "true" trade-off between length and space, we would also like the formulas to have space complexity smaller than the lower bound on the right-hand side of the inequalitites in Theorem 1. However, it is not known how to refute these formulas in space less than $\mathrm{O}(\sqrt{n})$. And in fact, given the structure of the formulas it is also hard to see why there should be a trade-off at all in the sense that larger proof length would somehow help to bring down the proof space.

Even more intriguingly, the bounds in Theorem 1 closely parallel the result obtained for resolution in 2002 by Ben-Sasson (journal version in [13]), where it was shown for a similar family of formulas that a trade-off on the form $s \log L = \Omega(n / \log n)$ must hold for any resolution refutation in length L and clause space s. Six years later, the factor $\log L$ was shown to be an artifact of the proof technique and was removed in a joint paper by Ben-Sasson and the second author [16] to obtain an unconditional space lower bound $\Omega(n/\log n)$ as mentioned above.

It is very tempting to conjecture that the situation should be the same for polynomial calculus resolution and cutting planes, so that Theorem 1 should be understood as providing "conditional space lower bounds" from which we should strive to remove the $\log L$ factor. If this could be done, it would imply that the combinatorics of pebbling on graphs is very robust, in that the properties of a concrete problem instance survives even the transformation of being (a) encoded as a CNF formula, (b) then translated from CNF into polynomial equations or linear inequalities, and (c) finally subjected to general-purpose propositional proof systems for reasoning in terms of such polynomial equations or linear equalities. The implication would be that no matter how polynomial calculus or cutting planes would try to work with these syntactic objects, which are very distinct from the graphs in terms of which they were generated, there would be no way to use less space than the lower bounds applying for pebbling strategies for the original graphs. And this would in turn open up the possibility of obtaining strong trade-off results for PCR and CP from pebbling time-space trade-offs along the lines of [17].

Before obtaining the results in this paper, we considered it very much unclear whether anything along those lines should have been expected to hold, but Theorem 1 does seem to provide fairly strong circumstantial evidence supporting such hypotheses. We are currently not able to prove this, however, but neither do we know of any formal obstacles to carrying out such a program.

1.3 Outline of This Paper

The rest of this paper is organized as follows. We start by presenting some standard proof complexity preliminaries in Section 2. In Section 3, we then give a more detailed overview of our proof complexity trade-off results, and describe the key concepts used in the proofs. In the end, this boils down to proving communication complexity lower bounds, and we discuss this separately in Section 4 and give an outline of that part of the argument. For the full proof of the communication complexity result, however, we refer the reader to the full-length version of this paper. Finally, in Section 5, we make some concluding remarks and mention a few of the many fascinating problems in this area that remain open.

2. SOME PROOF COMPLEXITY PRELIMINARIES

For x a Boolean variable, a *literal over* x is either the variable x itself, called a *positive literal over* x, or its negation, denoted $\neg x$ or $\overline{x}$ and called a *negative literal over* x. A *clause* $C = a_1 \vee \cdots \vee a_k$ is a disjunction of literals, and a *term* $T = a_1 \wedge \cdots \wedge a_k$ is a conjunction of literals. Below we will think of clauses and terms as sets, so that the ordering of the literals is inconsequential and that, in particular, no literals are repeated. A clause (term) containing at most k literals is called a *k-clause* (*k-term*). A *CNF formula* $F = C_1 \wedge \cdots \wedge C_m$ is a conjunction of clauses, and a *DNF formula* is a disjunction of terms. We will think of CNF and DNF formulas as sets of clauses and terms, respectively. A *k-CNF formula* is a CNF formula consisting of k-clauses, and a *k-DNF formula* consists of k-terms.

The *variable set* of a clause C, denoted $Vars(C)$, is the set of Boolean variables over which there are literals in C, and we write $Lit(C)$ to denote the set of literals in C. The variable and literal sets of a term are similarly defined and these definitions are extended to CNF and DNF formulas by taking unions. If V is a set of Boolean variables and $Vars(C) \subseteq V$ we say C is a clause *over* V and similarly define terms, CNF formulas, and DNF formulas over V.

We write α, β to denote truth value assignments. Truth value assignments are functions to $\{0,1\}$, where we identify 0 with false and 1 with true. We have the usual semantics that a clause is true under α, or *satisfied* by α, if at least one literal in it is true, and a term is true if all literals evaluate to true. We write $\bot$ to denote the empty clause without literals that is false under all truth value assignments. A CNF formula is satisfied if all clauses in it are satisfied, and for a DNF formula we require that some term should be satisfied. In general, we will not distinguish between a formula and the Boolean function computed by it.

If $\mathbb{C}$ is a set of Boolean functions we say that an assignment satisfies $\mathbb{C}$ if and only if it satisfies every function in $\mathbb{C}$. For $\mathbb{D}, \mathbb{C}$ two sets of Boolean functions over a set of variables V, we say that $\mathbb{D}$ *implies* $\mathbb{C}$, denoted $\mathbb{D} \vDash \mathbb{C}$, if and only if every assignment $\alpha : V \mapsto \{0,1\}$ that satisfies $\mathbb{D}$ also satisfies $\mathbb{C}$. In particular, $\mathbb{D} \vDash \bot$ if and only if $\mathbb{D}$ is *unsatisfiable* or *contradictory*, i.e., if no assignment satisfies $\mathbb{D}$. If a CNF formula F is unsatisfiable but for any clause $C \in F$ it holds that the clause set $F \setminus \{C\}$ is satisfiable, we say that F is *minimally unsatisfiable*.

We say that a proof system is *sequential* if a proof π in the system is a *sequence* of lines $\pi = \{L_1, \ldots, L_\tau\}$ of some prescribed syntactic form depending on the proof system in question, where each line is derived from previous lines by one of a finite set of allowed *inference rules*. Following the exposition in [37], we view a proof as similar to a non-deterministic Turing machine computation, with a special read-only input tape from which the clauses of the CNF formula F being refuted (the *axioms*) can be downloaded and a working memory where all derivation steps are made. Then the length of a proof is essentially the time of the computation and space measures memory consumption. The following definition is a straightforward generalization of [1].

Definition 3 (Refutation). *For a sequential propositional proof system $\mathcal{P}$, a $\mathcal{P}$-*configuration* $\mathbb{D}$ is a set of lines L of the syntactic form prescribed by $\mathcal{P}$. A sequence of configurations $\{\mathbb{D}_0, \ldots, \mathbb{D}_\tau\}$ is a $\mathcal{P}$-*derivation* from a CNF formula F if $\mathbb{D} = \emptyset$ 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*).*

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 $\pi : F \vdash \bot$ of a CNF formula F is a 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 and $\mathcal{R}(k)$-systems the empty clause without literals).

If every line L in a derivation is used at most once before being erased (though it can possibly be rederived later), we say that the derivation is tree-like*. This corresponds to changing the inference rule so that $L_1, \ldots, L_d$ must all be erased after they have been used to derive L.*

To every refutation π we can associate a DAG G_π, with the lines in π labelling the vertices and with edges from the assumptions to the consequence for each application of an inference rule. There might be several different derivations of a line L during the course of the refutation π, but if so we can label each occurrence of L with a time-stamp when it was derived and keep track of which copy of L is used where. Using this representation, a refutation π can be seen to be tree-like if G_π is a tree.

Definition 4 (Size, length and space). *For a given measure of size $S(L)$ 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 appear multiple times are counted with repetitions (once for every vertex in G_π). The* length *of a $\mathcal{P}$-derivation π is the number of axiom downloads and inference steps in it, i.e., the number of vertices in G_π.*[2] *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.*

If π is a refutation of a formula F in size S and space s, then we say that F can be refuted in size S and space s simultaneously. *Similarly, F can be refuted in length L and space s simultaneously if there is a $\mathcal{P}$-refutation $\mathcal{P}$ with $L(\pi) = L$ and $Sp(\pi) = s$.*

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 by taking the minimum with respect to length or space, respectively, over all $\mathcal{P}$-refutations of F.*

When the proof system in question is clear from context, we will drop the subindex in the proof complexity measures.

Let us next give formal definitions of the proof systems that will be of interest in this paper. Below, the notation $\dfrac{G_1 \quad \cdots \quad G_m}{H}$ means that if $G_1, \ldots, G_m$ have been derived previously in the proof (and are currently in memory), then we can infer H.

Definition 5 (k-DNF resolution). *The* k-DNF resolution *proof systems are a family of sequential proof systems $\mathcal{R}(k)$ parameterized by $k \in \mathbb{N}^+$. Lines in a k-DNF-resolution refutation are k-DNF formulas and we have the following inference rules (where G, H denote k-DNF formulas, T, T' denote k-terms, and $a_1, \ldots, a_k$ denote literals):*

$$k\textbf{-cut}\ \frac{(a_1 \wedge \cdots \wedge a_{k'}) \vee G \qquad \overline{a}_1 \vee \cdots \vee \overline{a}_{k'} \vee H}{G \vee H}\ ,\ k' \leq k.$$

$$\wedge\textbf{-introduction}\ \frac{G \vee T \qquad G \vee T'}{G \vee (T \wedge T')}\ ,\ |T \cup T'| \leq k.$$

$$\wedge\textbf{-elimination}\ \frac{G \vee T}{G \vee T'}\ \text{for any } T' \subseteq T.$$

$$\textbf{Weakening}\ \frac{G}{G \vee H}\ \text{for any } k\text{-DNF formula } H.$$

For standard resolution, i.e., $\mathcal{R}(1)$, the k-cut rule simplifies to the *resolution rule*

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

for clauses B and C. We refer to (2.1) as *resolution on the variable* x and to $B \vee C$ as the *resolvent* of $B \vee x$ and $C \vee \overline{x}$ on x. Clearly, in resolution the $\wedge$-introduction and $\wedge$-elimination rules do not apply. It can also be shown that the weakening rule never needs to be used in resolution refutations, but it can be convenient to allow it to simplify some technical arguments in proofs.

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

For $\mathcal{R}(k)$-systems, the length measure is as defined in Definition 4, and for space we get the two measures *formula space* and *total space* depending on whether we consider the number of k-DNF formulas in a configuration or all literals in it, counted with repetitions. For standard resolution there are two more space-related measures that will be relevant, namely *width* and *variable space*. For clarity, let us give an explicit definition of all space-related measures for resolution that will be of interest.

Definition 6 (Resolution width and space). *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 a widest clause in it. The* clause space *(as the formula space measure is known in resolution) $Sp(\mathbb{C})$ of a clause configuration $\mathbb{C}$ is $|\mathbb{C}|$, i.e., the number of clauses in $\mathbb{C}$, the* variable space *$VarSp(\mathbb{C})$ is $|Vars(C)|$, i.e., the number of distinct variables mentioned in $\mathbb{C}$, and the* total space *$TotSp(\mathbb{C})$ is $\sum_{C \in \mathbb{C}} |C|$, i.e., the total number of literals in $\mathbb{C}$ counted with repetitions.*

The width or space of a refutation π is the maximum that the corresponding measures attain over any clause configuration $\mathbb{C} \in \pi$, and taking the minimum over all resolution refutations of a CNF formula F, we can define the width $W_{\mathcal{R}}(F \vdash \bot) = \min_{\pi : F \vdash 0}\{W(\pi)\}$ of refuting F, and analogously the clause space $Sp_{\mathcal{R}}(F \vdash \bot) = \min_{\pi : F \vdash 0}\{Sp(\pi)\}$, variable space $VarSp_{\mathcal{R}}(F \vdash \bot) = \min_{\pi : F \vdash 0}\{VarSp(\pi)\}$, and total space $TotSp_{\mathcal{R}}(F \vdash \bot) = \min_{\pi : F \vdash 0}\{TotSp(\pi)\}$ of refuting F.

When studying and comparing the complexity measures for resolution in Definition 6, 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 so since the width and space measures can "misbehave" rather artificially for formula families of unbounded width (see [51, Section 5] for a discussion of this). Since every CNF formula can be rewritten as an equivalent formula of bounded width by using auxiliary variables, it therefore seems natural to insist that the formulas under study should have width bounded by some constant.

The *cutting planes* proof system, or *CP* for short, was introduced in [32]. Here, clauses are translated to linear inequalities—for instance, $x \vee y \vee \overline{z}$ gets translated to $x + y + (1-z) \geq 1$, i.e., $x + y - z \geq 0$—and these linear inequalities are then manipulated to derive a contradiction.

Definition 7 (Cutting planes). *Lines in a cutting planes proof are linear inequalities with integer coefficients. The derivation rules are as follows:*

$$\textbf{Variable axioms}\ \frac{}{x \geq 0}\ \text{and}\ \frac{}{-x \geq -1}\ \text{for all } x.$$

$$\textbf{Addition}\ \frac{\sum a_i x_i \geq A \qquad \sum b_i x_i \geq B}{\sum (a_i + b_i) x_i \geq A + B}$$

$$\textbf{Multiplication}\ \frac{\sum a_i x_i \geq A}{\sum c a_i x_i \geq cA}\ \text{for a positive integer } c.$$

$$\textbf{Division}\ \frac{\sum c a_i x_i \geq A}{\sum a_i x_i \geq \lceil A/c \rceil}\ \text{for a positive integer } c.$$

A CP-refutation ends when the inequality $0 \geq 1$ has been derived. The cutting planes measures we will focus on in this paper are the length measure and the line space *measure, the latter of which is the number of linear inequalities that are kept simultaneously on the blackboard during the refutation.*

Line space in cutting planes is easily seen to be a generalization of clause space in resolution, and since cutting planes can simulate resolution proofs it follows that the line space complexity of a formula in cutting planes is upper-bounded by its clause space complexity in resolution.

The k-DNF resolution proof systems are *logic-based* systems in the sense that they manipulate logic formulas, and cutting planes is an example of a *geometric* proof systems where clauses are treated as geometric objects. We will also be interested in two *algebraic* proof systems in this paper. The first of these is *polynomial calculus (PC)*, which was introduced in [28] under the name of "Gröbner proof system." In a PC-refutation, clauses are interpreted as multilinear polynomials. For instance, the requirement that the clause $x \vee y \vee \overline{z}$ should be satisfied gets translated to the equation $(1-x)(1-y)z = 0$ or $xyz - xz - yz + z = 0$, and we derive contradiction by showing that there is no common root for the polynomial equations corresponding to all the clauses.[3]

Definition 8 (Polynomial calculus). *In a polynomial calculus proof, lines 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 $\dfrac{p}{xp}$

A PC-refutation ends when 1 has been derived (i.e., $1 = 0$). The size *of a PC-refutation is defined as the total number of monomials in the refutation, the* length *of a refutation is the number of polynomial equations, and the* (monomial) space *is the maximal number of monomials in any configuration (counted with repetitions). Another important measure is the* degree *of a refutation, which is the maximal (total) degree of any monomial.*

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

Definition 9 (Polynomial calculus resolution). *The lines in a PCR-derivation are polynomials over the ring $\mathbb{F}[x, x', y, y', z, z', \ldots]$, where as before $\mathbb{F}$ is some field. We have all the axioms and rules of PC plus the following axioms:*

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

Size, length, and degree are defined as for polynomial calculus, and the (monomial) space of a PCR-refutation is again the maximal number of monomials in any PCR-configuration counted with repetitions.[4]

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

In general, the admissible inferences in a proof system according to Definition 3 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 10 (Syntactic and semantic proofs). *We refer to proofs according to Definition 3, where each new line L has to be inferred by one of the inference rules for $\mathcal{P}$, as* syntactic *proofs. 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 *proof.*

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

Let us finally 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 (e.g., clauses, k-DNF formulas, or inequalities). 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. The "line space" measure makes perfect sense for resolution and k-DNF resolution, and seems to do so also for cutting planes. For PC/PCR, however, measuring just

[3] In fact, from a mathematical point of view it seems more natural to think of 0 as true and 1 as false in polynomial calculus, so that the unit clause x gets translated to $x = 0$. For simplicity and consistency in this paper, however, we stick to thinking about $x = 1$ as meaning that x is true and $x = 0$ as meaning that x is false.

[4] We remark that in [1] space was defined as the number of *distinct* monomials (i.e., not counted with repetitions), but we find this restriction to be somewhat arbitrary.

the number of polynomial equations is not very meaningful, since every equation can be of exponential size and encode very much information. Instead, the natural generalization of clause space is monomial space.

3. OVERVIEW OF PROOF COMPLEXITY TRADE-OFFS

In this section, we describe which ingredients go into our results stated in Section 1.2 and how these results are proven. Our goal is to give an accessible high-level outline of the proofs, but still make clear what the main technical points in the arguments are and how they fit together. Briefly, the structure of our argument is as follows (where italicized words are concepts that we will make precise below):

1. We construct a family of CNF formulas of constant width by *lifting* so-called *pebbling contradictions* defined over *pyramid graphs.*

2. We then show that efficient refutations of these formulas, i.e., refutations in short length and small space, in polynomial calculus resolution or cutting planes give rise to efficient protocols for a certain *two-party communication complexity problem.* Although this is a fairly straightforward argument, we are not aware of this line of reasoning having been used before to prove time-space trade-offs in proof complexity.

3. Finally, we prove that *this communication complexity problem is hard,* so there cannot exist too good protocols. This is where essentially all of the work in this paper is, and where Theorem 2 comes into the picture.

Putting all of this together, Theorem 1 follows. The proof of this theorem is given in Section 3.4.

3.1 Lifting

The idea behind *lifting* (a term coined in [33]) is that we can take a base function f over some domain and extend it to a function over tuples from the same domain by combining it with a *selector* function that determines on which coordinates from the tuples f should be evaluated. The formal definition unfortunately turns out to be slightly involved, but it is absolutely crucial for our argument and so we start by making this definition precise.

Given a function $f : \{0,1\}^m \mapsto Q$ for some range Q and an integer $\ell > 0$, the *lift of length ℓ of f* is defined to be the function $\mathit{Lift}_\ell(f) : \{0,1\}^{m\times\ell} \times [\ell]^m \mapsto Q$ such that for any bit-vector $\{x_{i,j}\}_{i\in[m],j\in[\ell]}$ and any $y \in [\ell]^m$, the function evaluated on these arguments is

$$\mathit{Lift}_\ell(f)(x,y) = f(x_{1,y_1}, x_{2,y_2}, \ldots, x_{m,y_m}) \tag{3.1}$$

(in words, the vector y selects which coordinates of the x-vector should be fed to f).

Let S be any *search problem* defined as a subset of $A\times Q$ for some input domain A and output range Q; that is, on any input $a \in A$, the problem is to find some $q \in Q$ such that $(a,q) \in S$. If $A = \{0,1\}^m$ and $\ell > 0$ is any integer, we define the *lift of length ℓ of S* to be a new search problem $\mathit{Lift}_\ell(S) \subseteq \{0,1\}^{m\times\ell} \times [\ell]^m \times Q$ with input domain $\{0,1\}^{m\times\ell} \times [\ell]^m$ and output range Q such that for any bit-vector $\{x_{i,j}\}_{i\in[m],j\in[\ell]}$, any $y \in [\ell]^m$ and any $q \in Q$, it holds that

$$(x,y,q) \in \mathit{Lift}_\ell(S) \text{ if and only if } \big((x_{1,y_1}, x_{2,y_2}, \ldots, x_{m,y_m}), q\big) \in S \ . \tag{3.2}$$

Lifted CNF formulas, as first introduced in [10], are constructed in the following way.

Definition 11 (Lift of CNF formula [10]). *Given any CNF formula F consisting of clauses $C_1,\ldots,C_m$ over variables $u_1,\ldots,u_n$, and any integer $\ell > 0$, the* lift of length ℓ *of F is a CNF formula $\mathit{Lift}_\ell(F)$ over $2\ell n$ variables denoted by $\{x_{i,j}\}_{i\in[n],j\in[\ell]}$ (*main variables*) and $\{y_{i,j}\}_{i\in[n],j\in[\ell]}$ (*selector variables*), consisting of the following clauses:*

- *For every $i \in [n]$, an* auxiliary clause
$$y_{i,1} \lor y_{i,1} \lor \cdots \lor y_{i,\ell} \ . \tag{3.3}$$
- *For every clause $C_i \in F$, where $C_i = u_{i_1} \lor \cdots \lor u_{i_s} \lor \overline{u}_{i_{s+1}} \lor \cdots \lor \overline{u}_{i_{s+t}}$ for some $i_1,\ldots,i_{s+t} \in [n]$, and for every tuple $(j_1,\ldots,j_{s+t}) \in [\ell]^{s+t}$, a* main clause
$$\overline{y}_{i_1,j_1} \lor x_{i_1,j_1} \lor \cdots \lor \overline{y}_{i_s,j_s} \lor x_{i_s,j_s} \lor \overline{y}_{i_{s+1},j_{s+1}} \lor \overline{x}_{i_{s+1},j_{s+1}} \lor \cdots \lor \overline{y}_{i_{s+t},j_{s+t}} \lor \overline{x}_{i_{s+t},j_{s+t}} \ . \tag{3.4}$$

Let us elaborate on what Definition 11 means. The purpose of the auxiliary clauses in (3.3) is to make sure that in every block of variables $\{y_{i,j} \mid 1 \le j \le \ell\}$ at least one of the selector y-variables is true. Noting that every pair $\overline{y}_{i,j} \lor x_{i,j}$ in a main clause (3.4) is equivalent to an implication $y_{i,j} \to x_{i,j}$, we can rewrite (3.4) as

$$(y_{i_1,j_1} \to x_{i_1,j_1}) \lor \cdots \lor (y_{i_{s+t},j_{s+t}} \to \overline{x}_{i_{s+t},j_{s+t}}) \ . \tag{3.5}$$

What this boils down to is that for every clause C_i, the auxiliary clauses encode that there is at least one choice of selector variables $y_{i,j}$ which are all true, and for this choice of selector variables the $x_{i,j}$-variables will play the role of the u_i-variables, giving us back the original clause C_i. It is easily verified that F is unsatisfiable if and only if $G = \mathit{Lift}_\ell(F)$ is unsatisfiable, and if F is a k-CNF formula with m clauses, then G is a $\max(2k,\ell)$-CNF formula with at most $m\ell^k + n$ clauses.

3.2 Pebbling Contradictions and Pyramids

Pebbling is a tool for studying time-space relationships by means of a game played on directed acyclic graphs. This game models computations where the execution is independent of the input and can be performed by straight-line programs. Pebble games were originally devised for studying programming languages and compiler construction, but have later found a broad range of applications in computational complexity theory. In the last decade, there has been renewed study of pebbling in the context of proof complexity. An excellent survey of pebbling up to ca 1980 is [56], and some more recent developments are covered in the second author's upcoming survey [52].

The way pebbling results have been used in proof complexity has mainly been by studying so-called *pebbling contradictions* as defined in [18]. These are CNF formulas encoding the pebble game played on a DAG G by postulating the sources to be true and the sink to be false, and specifying that truth propagates through the graph according to the pebbling rules.

Definition 12 (Pebbling contradiction). *Let G be a DAG with sources S and a unique sink z. Identify every vertex $v \in V(G)$ with a propositional logic variable v. The* pebbling contradiction *over G, denoted Peb_G, is the conjunction of the following clauses:*

- *for all $s \in S$, a unit clause s (*source axioms*),*
- *For all non-sources v with predecessors $pred(v)$, the clause $\bigvee_{u \in pred(v)} \overline{u} \vee v$ (*pebbling axioms*),*
- *for the sink z, the unit clause $\overline{z}$ (*sink axiom*).*

If the DAG G has n vertices and maximal indegree d, the formula Peb_G is a minimally unsatisfiable $(1+d)$-CNF formula with $n+1$ clauses over n variables.

In this paper, we will focus on pebbling contradictions over *pyramid graphs*. For an example of such a pebbling contradiction, see the CNF formula in Figure 1(b) defined in terms of the graph in Figure 1(a). For completeness, the formal definition of a pyramid graph is as follows.

Definition 13 (Pyramid). *The* pyramid graph Π_h *of height h is a layered DAG with $h+1$ levels, where there is one vertex on the highest level (the sink z), two vertices on the next level et cetera down to $h+1$ vertices at the lowest level 0. The ith vertex at level L has incoming edges from the ith and $(i+1)$st vertices at level $L-1$.*

To make the connection back to Definition 11, in Figure 2 we present the lift of length 2 of the CNF formula in Figure 1(b), with the auxiliary clauses at the top of the left column followed by the main clauses one by one, listed for all tuples of selector indices (with the only difference that since the variables in this formula are u, v, w, x, y, z rather than $u_1, \ldots, u_n$, we denote the main variables by $x_{u,j_1}, x_{v,j_2}, x_{w,j_3}$, et cetera, rather than $x_{i_1,j_1}, x_{i_2,j_2}, \ldots$). We will refer to the main clauses in Figure 1(b) as source axioms, pebbling axioms, and sink axioms, respectively, when they have been obtained by lifting of the corresponding axioms in the pebbling contradiction.

We remark that the process of lifting formulas is similar to, but different from, the substitution used in [17], where some (fixed) function $f(x_1, \ldots, x_\ell)$ is substituted for every variable x and the resulting formula is expanded out to CNF again using De Morgan's laws. Although we believe that lifted pebbling formulas should share many of the properties of substitution pebbling formulas, the proofs in [17] would require modifications to work for lifted formulas, and we have not studied in any detail if and how such modifications could be made.

3.3 Two-Player Communication Complexity

The basic two-player model of communication complexity suffices for our purposes in this paper. We briefly review this model below and refer to [44] for further details and formal definitions.

For any function $f : X \times Y \mapsto Q$, where X and Y are some domains and Q is some range, in the communication problem of computing f Alice is given an input $x \in X$, Bob is given an input $y \in Y$, and they are required to compute $f(x, y)$ minimizing the communication between them. Similarly, for any search problem $S \subseteq X \times Y \times Q$, where X and Y are some input domains and Q is some output range, the communication problem solving S is one in which Alice is given $x \in X$, Bob is given $y \in Y$, and they are required to communicate to find some q such that $(x, y, q) \in S$. The cost of a communication protocol is the maximum number of bits communicated on any input. The *deterministic communication complexity* of a function f (or a search problem S) is the minimum cost of any communication protocol solving f (or S, respectively).

We are also concerned with randomized communication, in which each player has access to a private infinite source of random bits[5] and the messages can depend on both the input and the random bits. We say that a randomized communication protocol solves a function f (or a search problem S) with error ϵ if on any input, the protocol outputs a correct answer with probability at least $1 - \epsilon$ over the random choices. Note that in a search problem, an input may have more than one correct output, and the protocol is allowed to output different correct outputs depending on the random choices. The *randomized communication complexity* with error ϵ of a function f (or a search problem S) is the minimum cost of any randomized protocol with error ϵ solving f (or S, respectively).

A key notion for this paper is that a randomized communication protocol is said to solve a search problem *consistently* if on any input assignment, the protocol outputs a *unique* correct output with probability at least 3/4 over its random choices.

Probabilistic errors in randomized protocols can be reduced by a small blow-up in communication cost as stated next.

Proposition 14. *For any $0 < \epsilon' < \epsilon < 1/2$, if there exists a randomized protocol Π with communication cost c and error ϵ to compute a function f, then there exists a randomized protocol Π' with cost $\mathrm{O}(c \cdot \log_\epsilon \epsilon')$ and error ϵ' to compute f.*

The proof is standard and we omit the details. The idea is to repeat the protocol Π $\mathrm{O}(\log_\epsilon \epsilon')$ times and take a majority vote. The analysis uses Chernoff bounds.

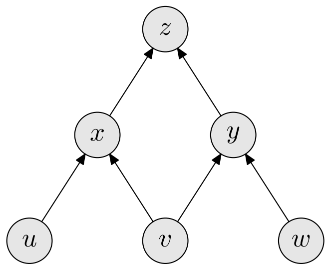
3.4 Leveraging Communication Complexity to Obtain Proof Complexity Trade-offs

Let us now provide formal lemmas implementing the steps in the argument outlined at the beginning of this section.

Given any CNF formula G, the *(falsified clause) search problem* for G is the problem of finding, given any truth value assignment α, a clause $C \in G$ falsified by α. We denote this problem by G^{search}. The next lemma says that if a CNF formula G has a polynomial calculus resolution refutation (or cutting planes refutation, respectively) in simultaneous short length and small space, then there is a small deterministic (or randomized, respectively) two-player protocol that solves G^{search} for any truth assignment and any partition of the variables between Alice and Bob. We remark that this lemma holds also for the stronger *semantic* versions of these proof systems in the sense of Definition 10.

Lemma 15. *Let G be any CNF formula over n variables, and suppose that each of these variables is given to either one of Alice and Bob. Then the followings holds:*

[5]Here "private" means that one player cannot see the random bits of the other. There is also a public-coin randomized communication model [44], but it is not our concern in this paper.



(a) Pyramid graph Π_2 of height 2.

$$
\begin{aligned}
& u \\
\wedge\ & v \\
\wedge\ & w \\
\wedge\ & (\overline{u} \vee \overline{v} \vee x) \\
\wedge\ & (\overline{v} \vee \overline{w} \vee y) \\
\wedge\ & (\overline{x} \vee \overline{y} \vee z) \\
\wedge\ & \overline{z}
\end{aligned}
$$

(b) Pebbling contradiction Peb_{Π_2}.

Figure 1: Pebbling contradiction for the pyramid graph Π_2.

$$
\begin{aligned}
& (y_{u,1} \vee y_{u,2}) \\
\wedge\ & (y_{v,1} \vee y_{v,2}) \\
\wedge\ & (y_{w,1} \vee y_{w,2}) \\
\wedge\ & (y_{x,1} \vee y_{x,2}) \\
\wedge\ & (y_{y,1} \vee y_{y,2}) \\
\wedge\ & (y_{z,1} \vee y_{z,2}) \\
\wedge\ & (\overline{y}_{u,1} \vee x_{u,1}) \\
\wedge\ & (\overline{y}_{u,2} \vee x_{u,2}) \\
\wedge\ & (\overline{y}_{v,1} \vee x_{v,1}) \\
\wedge\ & (\overline{y}_{v,2} \vee x_{v,2}) \\
\wedge\ & (\overline{y}_{w,1} \vee x_{w,1}) \\
\wedge\ & (\overline{y}_{w,2} \vee x_{w,2}) \\
\wedge\ & (\overline{y}_{u,1} \vee \overline{x}_{u,1} \vee \overline{y}_{v,1} \vee \overline{x}_{v,1} \vee \overline{y}_{x,1} \vee x_{x,1}) \\
\wedge\ & (\overline{y}_{u,1} \vee \overline{x}_{u,1} \vee \overline{y}_{v,1} \vee \overline{x}_{v,1} \vee \overline{y}_{x,2} \vee x_{x,2}) \\
\wedge\ & (\overline{y}_{u,1} \vee \overline{x}_{u,1} \vee \overline{y}_{v,2} \vee \overline{x}_{v,2} \vee \overline{y}_{x,1} \vee x_{x,1}) \\
\wedge\ & (\overline{y}_{u,1} \vee \overline{x}_{u,1} \vee \overline{y}_{v,2} \vee \overline{x}_{v,2} \vee \overline{y}_{x,2} \vee x_{x,2}) \\
\wedge\ & (\overline{y}_{u,2} \vee \overline{x}_{u,2} \vee \overline{y}_{v,1} \vee \overline{x}_{v,1} \vee \overline{y}_{x,1} \vee x_{x,1}) \\
\wedge\ & (\overline{y}_{u,2} \vee \overline{x}_{u,2} \vee \overline{y}_{v,1} \vee \overline{x}_{v,1} \vee \overline{y}_{x,2} \vee x_{x,2}) \\
\wedge\ & (\overline{y}_{u,2} \vee \overline{x}_{u,2} \vee \overline{y}_{v,2} \vee \overline{x}_{v,2} \vee \overline{y}_{x,1} \vee x_{x,1}) \\
\wedge\ & (\overline{y}_{u,2} \vee \overline{x}_{u,2} \vee \overline{y}_{v,2} \vee \overline{x}_{v,2} \vee \overline{y}_{x,2} \vee x_{x,2})
\end{aligned}
$$

$$
\begin{aligned}
\wedge\ & (\overline{y}_{v,1} \vee \overline{x}_{v,1} \vee \overline{y}_{w,1} \vee \overline{x}_{w,1} \vee \overline{y}_{y,1} \vee x_{y,1}) \\
\wedge\ & (\overline{y}_{v,1} \vee \overline{x}_{v,1} \vee \overline{y}_{w,1} \vee \overline{x}_{w,1} \vee \overline{y}_{y,2} \vee x_{y,2}) \\
\wedge\ & (\overline{y}_{v,1} \vee \overline{x}_{v,1} \vee \overline{y}_{w,2} \vee \overline{x}_{w,2} \vee \overline{y}_{y,1} \vee x_{y,1}) \\
\wedge\ & (\overline{y}_{v,1} \vee \overline{x}_{v,1} \vee \overline{y}_{w,2} \vee \overline{x}_{w,2} \vee \overline{y}_{y,2} \vee x_{y,2}) \\
\wedge\ & (\overline{y}_{v,2} \vee \overline{x}_{v,2} \vee \overline{y}_{w,1} \vee \overline{x}_{w,1} \vee \overline{y}_{y,1} \vee x_{y,1}) \\
\wedge\ & (\overline{y}_{v,2} \vee \overline{x}_{v,2} \vee \overline{y}_{w,1} \vee \overline{x}_{w,1} \vee \overline{y}_{y,2} \vee x_{y,2}) \\
\wedge\ & (\overline{y}_{v,2} \vee \overline{x}_{v,2} \vee \overline{y}_{w,2} \vee \overline{x}_{w,2} \vee \overline{y}_{y,1} \vee x_{y,1}) \\
\wedge\ & (\overline{y}_{v,2} \vee \overline{x}_{v,2} \vee \overline{y}_{w,2} \vee \overline{x}_{w,2} \vee \overline{y}_{y,2} \vee x_{y,2}) \\
\wedge\ & (\overline{y}_{x,1} \vee \overline{x}_{x,1} \vee \overline{y}_{y,1} \vee \overline{x}_{y,1} \vee \overline{y}_{z,1} \vee x_{z,1}) \\
\wedge\ & (\overline{y}_{x,1} \vee \overline{x}_{x,1} \vee \overline{y}_{y,1} \vee \overline{x}_{y,1} \vee \overline{y}_{z,2} \vee x_{z,2}) \\
\wedge\ & (\overline{y}_{x,1} \vee \overline{x}_{x,1} \vee \overline{y}_{y,2} \vee \overline{x}_{y,2} \vee \overline{y}_{z,1} \vee x_{z,1}) \\
\wedge\ & (\overline{y}_{x,1} \vee \overline{x}_{x,1} \vee \overline{y}_{y,2} \vee \overline{x}_{y,2} \vee \overline{y}_{z,2} \vee x_{z,2}) \\
\wedge\ & (\overline{y}_{x,2} \vee \overline{x}_{x,2} \vee \overline{y}_{y,1} \vee \overline{x}_{y,1} \vee \overline{y}_{z,1} \vee x_{z,1}) \\
\wedge\ & (\overline{y}_{x,2} \vee \overline{x}_{x,2} \vee \overline{y}_{y,1} \vee \overline{x}_{y,1} \vee \overline{y}_{z,2} \vee x_{z,2}) \\
\wedge\ & (\overline{y}_{x,2} \vee \overline{x}_{x,2} \vee \overline{y}_{y,2} \vee \overline{x}_{y,2} \vee \overline{y}_{z,1} \vee x_{z,1}) \\
\wedge\ & (\overline{y}_{x,2} \vee \overline{x}_{x,2} \vee \overline{y}_{y,2} \vee \overline{x}_{y,2} \vee \overline{y}_{z,2} \vee x_{z,2}) \\
\wedge\ & (\overline{y}_{z,1} \vee \overline{x}_{z,1}) \\
\wedge\ & (\overline{y}_{z,2} \vee \overline{x}_{z,2})
\end{aligned}
$$

Figure 2: Lifted formula $Lift_2(Peb_{\Pi_2})$ of length 2 obtained from the pebbling contradiction over Π_2.

- *If G has a (semantic) PCR-refutation in monomial space s and length L, then there is a deterministic protocol of cost* $\mathrm{O}(s \log L)$ *bits for* G^{search}.
- *If G has a (semantic) CP-refutation in line space s and length L, then there is a* consistent *randomized protocol of cost* $\mathrm{O}(s \log L \log^2 n \log(s \log L))$ *bits with error* $1/4$ *for* G^{search}.

Proof. In one line, the proof is that Alice and Bob will use the refutation of G to do binary search for a falsified clause.

In a little bit more detail, suppose Alice and Bob are given their two parts of an assignment α. Fix a refutation $\pi = \{\mathbb{D}_0, \mathbb{D}_1, \ldots, \mathbb{D}_L\}$ as in the statement of the lemma, where as before $\mathbb{D}_0 = \emptyset$ and $\perp \in \mathbb{D}_L$. For ease of notation, suppose the length L of the refutation is actually the total number of derivation steps including erasures, as discussed in the footnote in Definition 4 (which is a very minor technicality).

Alice and Bob consider the configuration $\mathbb{D}_{L/2}$ in the refutation at time $L/2$ and with joint efforts (involving some communication, which we will discuss shortly) evaluate the truth value of $\mathbb{D}_{L/2}$ under the assignment α. If $\mathbb{D}_{L/2}$ is true under α, they continue their search in the subderivation $\{\mathbb{D}_{L/2}, \mathbb{D}_1, \ldots, \mathbb{D}_L\}$. If $\mathbb{D}_{L/2}$ is false, the search continues in the first half of the refutation $\{\mathbb{D}_0, \mathbb{D}_1, \ldots, \mathbb{D}_{L/2}\}$ Note that $\mathbb{D}_0 = \emptyset$ evaluates to true by default, and since $\perp \in \mathbb{D}_L$ this configuration evaluates to false, and the binary search is carried out so as to maintain that the first configuration in the current subderivation under consideration evaluates to true and the last one evaluates to false. Hence, after $\log L$ steps Alice and Bob find a $t \in [L]$ such that $\mathbb{D}_{t-1}$ is true under α but $\mathbb{D}_t$ is false. Since the proof system is sound, the derivation step to get from $\mathbb{D}_{t-1}$ to $\mathbb{D}_t$ cannot have been an inference or erasure, but must be a download of some axiom clause $C \in G$. This clause C must be false under α, and so Alice and Bob give C as their answer.

Now all that remains is to discuss how much communication is needed to evaluate a configuration in the refutation. For PCR, Alice and Bob can just look at all the monomials in the the current configuration $\mathbb{D}$, say $m_1, m_2, \ldots, m_{s'}$ for $s' \leq s$. For each monomial m_i, Alice evaluates the part m_i^A containing her variables and Bob the part m_i^B containing his, they communicate the values with a constant number of bits (since the field is finite), and multiply to get $m_i = m_i^A \cdot m_i^B$. There are at most s monomials, so the total communication for the configuration is $\mathrm{O}(s)$, and then they both know the truth value of the configuration. Note also that the communication clearly is deterministic.

For cutting planes, the same argument applies, except that to check the truth of each line, we have to do a randomized protocol. Fortunately, this can be achieved with communication cost $O(\log^2 n)$ with constant error as shown in Lemma 5 in [40] (also in [12]). Applying a standard reduction of randomized error by a small blow-up in communication cost as in Proposition 14, we can make the error sufficiently small, say $\frac{1}{4s \log L}$, with communication cost $\mathrm{O}\big(\log^2 n \log(s \log L)\big)$ for each line check. Thus, by a union bound, except with probability $1/4$, all $s \log L$ line checks in the protocol are done correctly, and hence the protocol outputs a unique answer. In this way we obtain a consistent randomized protocol. The lemma follows. $\square$

Given Lemma 15, in order to prove Theorem 1, we need to find a family of CNF formulas whose search problem has large communication complexity. This is achieved by the following lemma.

Lemma 16. *Let $\big\{Peb_{\Pi_h}\big\}_{h=1}^{\infty}$ denote the family of pebbling contradictions over pyramid graphs of height h, and let $G_h = Lift_3\big(Peb_{\Pi_h}\big)$. Then any consistent randomized (and hence also any deterministic) two-party protocol solving the search problem associated with G_h requires $\Omega\big(\sqrt{h}\big)$ bits of communication, where Alice receives the x-variables and Bob receives the y-variables in G_h.*

We remind the reader that for purposes of illustration, a lift *of length* 2 of a pyramid pebbling contradiction can be found in Figure 2.

To prove Lemma 16, we first need to establish the communication complexity lower bound stated in Theorem 2. We discuss in more detail in Section 4 how to prove this theorem, but the reader is referred to the full-length version for a complete proof. Taking Lemma 16 on faith for the time being, however, we can now prove our main proof complexity result.

Theorem 1 (restated). *There is a family of k-CNF formulas $\{F_n\}_{n=1}^{\infty}$ of size $\Theta(n)$ that can be refuted in length $\mathrm{O}(n)$ in resolution, polynomial calculus (and hence also polynomial calculus resolution) and cutting planes, but for which the following holds:*

- *For any PCR-refutation of the formula F_n in simultaneous length L and monomial space s it must hold that* $s \log L = \Omega\big(\sqrt[4]{n}\big)$.
- *For any CP-refutation of the formula F_n in simultaneous length L and line space s it must hold that* $s \log L \log(s \log L) = \Omega\left(\sqrt[4]{n} \,/ \log^2 n\right)$.

Proof. We set $F_n = Lift_3\big(Peb_{\Pi_h}\big)$ for $h = \Theta(\sqrt{n})$, since pebbling contradictions over pyramids have size quadratic in the pyramid height. The lower bounds follow by looking at the communication protocols constructed from purported refutations of F_n in Lemma 15 and applying the lower bound in Lemma 16.

It remains to prove the upper bound. This is fairly standard, and we will not go into too much details. The proof closely follows similar proofs in, e.g., [17]. The idea is that we look at a black pebbling strategy for pyramid graphs (or for any DAG over which we have defined a pebbling contradiction). For pyramids, there are black pebbling strategies in linear time and space $\Theta(h)$. Then we build a resolution proof that simulates this pebbling strategy step by step to refute a lifted pyramid pebbling contradiction of length length ℓ for any $\ell = \mathrm{O}(1)$ (in particular, for $\ell = 3$).

We maintain the invariant that when a vertex v is pebbled, we derive $\overline{y}_{v,i} \vee x_{v,i}$ for all $i \in [\ell]$. In this way, when we get to the sink z, the clauses $\overline{y}_{z,i} \vee x_{z,i}$ together with the auxiliary clause $y_{z,1} \vee \cdots \vee y_{z,\ell}$ and the sink axioms $\overline{y}_{z,i} \vee \overline{x}_{z,i}$ for all $i \in [\ell]$ immediately yield a contradiction. The base case for the inductive argument are the source vertices, for which these clauses are just the source axioms. Consider a non-source vertex w and suppose for concreteness that it has exactly two immediate predecessors u and v. When a black pebble is placed on w, the vertices u and v must also have black pebbles on them by the rules of the pebble game, and so by induction we have the clauses $\overline{y}_{u,i} \vee x_{u,i}$ and $\overline{y}_{v,i} \vee x_{v,i}$ in memory for all $i \in [\ell]$. Download the pebbling axioms

$\overline{y}_{u,h} \vee \overline{x}_{u,h} \vee \overline{y}_{v,i} \vee \overline{x}_{v,i} \vee \overline{y}_{w,j} \vee x_{w,j}$ for all $h, i, j \in [\ell]$, as well as the auxiliary clauses $y_{u,1} \vee \cdots \vee y_{u,\ell}$ and $y_{v,1} \vee \cdots \vee y_{v,\ell}$. Then it is not hard to verify that all of these clauses together imply $\overline{y}_{w,j} \vee x_{w,j}$ for all $j \in [\ell]$, and by the implicational compleness of resolution there must then exist a derivation of these clauses. Since the number of variables involved is constant, this derivation has constant length, width, and space (and it is possible to do much better than the trivial upper bounds one gets from this argument, but we are not interested in optimizing constants here).

Thus, every step in the pebbling can be simulated in constant length and with constant extra clause space, so simulating the whole pebbling yields a resolution refutation in length $\mathrm{O}(n)$ and clause space $\mathrm{O}\big(\sqrt{n}\big)$. Since this resolution refutation also has width $\mathrm{O}(1)$, it can be simulated in CP and PC (and hence also PCR) with at most a constant factor blow-up in length and space. ☐

We remark that for the same family of formulas $\{F_n\}_{n=1}^{\infty}$ in Theorem 1, we can show that any tree-like *semantic* CP refutation of this formula must have length L such that $L \log L = \exp(\Omega(\sqrt[4]{n}/\log^2 n))$. This is shown by the same proof as for Theorem 1, except that we use the reduction from small tree-like semantic CP proof to small consistent randomized communication protocol for search problems as presented in [10] (and which can also be found implicitly in [12, 40]). To our knowledge, this is the first result showing that *semantic* tree-like CP cannot polynomially simulate resolution. A separation between *syntactic* tree-like CP and resolution was shown in [22] using the so-called interpolation method, which is not known to be applicable to semantic CP.

4. OBTAINING COMMUNICATION COMPLEXITY LOWER BOUNDS

In order to prove Lemma 16, we show a new communication complexity lower bound for the falsified clause search problems of CNF formulas. More generally, we show that if S is a search problem with a certain property, then the communication complexity of another search problem which is obtained by lifting S (using length $\ell \geq 3$) is large. This is stated in Theorem 2 and is our main technical contribution, and we consider this to be an interesting new result in communication complexity in its own right.

Before going into a more detailed discussion of Theorem 2, we need to define formally the concepts that we will use. Let us start by recalling the well-known notion of block sensitivity of (usually Boolean) functions as introduced by Nisan [49], and then generalize this notion for search problems. Let $f : \{0,1\}^m \mapsto Q$ be any function to some range Q. For an input $\alpha \in \{0,1\}^m$ and a subset $B \subseteq [m]$, we say that f is *sensitive to B on α* if $f(\alpha) \neq f\big(\alpha^B\big)$, where α^B is obtained from α by flipping all bits in B. The *block sensitivity of f on α*, denoted $bs(f, \alpha)$, is the maximum number s of pairwise disjoint blocks $B_1, \ldots B_s \subseteq [m]$ such that f is sensitive to each B_i on α. For any subset of assignments $A' \subseteq \{0,1\}^m$, we define the *block sensitivity over A' of f*, denoted $bs\big(f, A'\big)$, as

$$bs\big(f, A'\big) = \max_{\alpha \in A'} bs(f, \alpha) \ . \tag{4.1}$$

Thus, the well-known notion of block sensitivity of f is the block sensitivity of f over the full set of assignments $\{0,1\}^m$.

Let $S \subseteq A \times Q$ be any search problem with input domain A and output range Q. If $A = \{0,1\}^m$, we say that a function $f : \{0,1\}^m \mapsto Q$ *solves* or *satisfies* the search problem S if on any input $\alpha \in \{0,1\}^m$ it holds that $\big(\alpha, f(\alpha)\big) \in S$. A *critical input* α for a search problem $S \subseteq \{0,1\}^m \times Q$ is one with a *unique* $q \in Q$ such that $(\alpha, q) \in S$. This brings us to the concept of *critical block sensitivity of search problems*, which plays a key role in this paper and which we therefore highlight in a formal definition.

Definition 17. *The* block sensitivity of a search problem $S \subseteq \{0,1\}^m \times Q$ *over $A' \subseteq \{0,1\}^m$, is defined as*

$$bs\big(S, A'\big) = \min\{bs\big(f, A'\big) \mid f : \{0,1\}^m \mapsto Q \textit{ solves } S\}$$

(or in words, $bs\big(S, A'\big)$ is the minimum block sensitivity over A' of any function f satisfying S). If $A' = \{0,1\}^m$, we simply write $bs(S) = bs(S, A')$. We will be mostly interested in the set of all critical inputs of $S \subseteq \{0,1\}^m \times Q$, which we denote A_c, and we define $bs_{crit}(S) = bs(S, A_c)$ to be the critical block sensitivity *of S.*

Note that if S is a function, then $bs_{crit}(S)$ reduces to the usual notion of block sensitivity of functions. We are now ready to give a more precis formal statement of Theorem 2.

Theorem 18 (Formal version of Theorem 2). *Let $S \subseteq \{0,1\}^m \times Q$ be any search problem over input domain $\{0,1\}^m$ and range Q. Then it holds for any $\ell \geq 3$ that any consistent randomized (and hence also any deterministic) two-party protocol solving $G = Lift_\ell(S)$ requires $\Omega(bs_{crit}(S))$ bits of communication, where Alice receives the x-variables and Bob receives the y-variables in G.*

We remark that for any search problem S, there is a simple deterministic two-player protocol solving $G = Lift_3(S)$ with communication cost $\mathrm{O}(D(S))$, where $D(S)$ is the *deterministic decision tree complexity* of S. This protocol works by simply simulating the decision tree. It can also be proven that $bs(S) = \Omega\big(D(S)^{1/3}/\log_2|Q|\big)$ by a simple extension of an analogous inequality for Boolean functions [7]. Thus, if we could replace $\Omega(bs_{crit}(S))$ in the lower bound in Theorem 18 by $\Omega(bs(S))$, that would yield a stronger lower bound, which would be within a polynomial of the simple upper bound mentioned above. We are not able to prove nor disprove such a bound, however, so this remains an interesting open question.

One can observe, however, that it is often the case in the literature (e.g., in the case of the lower bounds for resolution proven in [23, 39]) that lower bounds are obtained by studying only critical assignments. In such cases, if the decision tree complexity of solving a search problem S restricted to critical inputs is large, then this can be taken to suggest that $bs_{crit}(S)$ should also be large. And indeed, the second key ingredient that we use together with Theorem 18 in our proof of Lemma 16 is the following lemma, saying that the search problem for pyramid pebbling contradictions (which can easily be shown to have large decision tree complexity even when restricted to critical inputs) has large critical block sensitivity.

Lemma 19. *Let $\{Peb_{\Pi_h}\}_{h=1}^{\infty}$ be the family of pebbling contradictions over pyramid graphs of height h. Then the critical block sensitivity of the search problem for the formula Peb_{Π_h} is $\Omega\big(\sqrt{h}\big)$.*

We refer to the full-length version of the paper for the proofs of Theorem 18 and Lemma 19. Let us just mention here, however, that the proof of Theorem 18 is a new application of the information-theoretic approach in two-player communication complexity as presented in [5, 26]. To the best of our knowledge, this is the first time such an approach has been used to derive strong, general communication lower bounds for a search problem.

To conclude this section, let us discuss some previous work related to Theorem 18 and in this way give some background and motivation for our new result. Although Theorem 18 is certainly not the first in the literature that shows strong communication lower bounds for search problems related to CNF formulas, previous results do not suffice for our purposes in this paper. One such result was proved by Raz and McKenzie [58] and improved by Bonet et al. [22], who showed that for a certain broad class of search problems $S \subseteq \{0,1\}^m \times Q$ (in particular, the search problem S for any CNF formula F), the deterministic[6] communication complexity of $Lift_\ell(S)$ is essentially $D(S)$ provided that $\ell \geq m^{14}$. Although this lower bound is tight, it requires polynomially large length ℓ. Thus, the lifted CNF formula $Lift_\ell(F)$ does not have bounded width, and may not be refutable in polynomial length in resolution. Moreover, deterministic communication complexity lower bounds do not suffice for us to derive length-space tradeoffs for CP.

Strong (and non-consistent) randomized communication complexity for the search problems of some families of CNF formulas were also known previously. In [40], Impagliazzo et al. proved one such two-player lower bound for a specific family of CNF formulas. However, this formula family have clauses of polynomially large width. In [12], Beame et al. proved a strong multiparty randomized communication lower bounds in the so-called *number-on-the-forehead* model for a search problem of another family of formulas (namely, Tseitin formulas over certain expander graphs with a non-trivial modification). Similarly to the last result, this family of formulas also have unbounded (logarithmic) width. Furthermore, the formulas in both of these results are known to require exponential refutation length in resolution, and no polynomial-size refutations are known for them in PCR or CP. Since the formulas in these results must be carefully chosen for the techniques to be applicable, it is not known whether previous techniques yield lower bounds for any formulas that do not suffer from the aforementioned drawbacks.

Therefore, although on the face of it it might look straightforward to apply Lemma 15 with previously known communication complexity results to obtain strong length-space trade-offs for PCR and/or CP, such trade-offs would have several shortcomings. In particular, the formulas would all have unbounded width, and also we would not know any upper bounds showing that these formulas actually have polynomial-size refutations. But in Theorem 18, for the first time[7] we are able to give a consistent[8] randomized two-player communication lower bound for search problems in terms of their general characteristics (critical block sensitivity) and apply this lower bound to the search problem of a family of CNF formulas which have constant width and are refutable in polynomial length in resolution.

5. CONCLUDING REMARKS

In this paper, we report the first time-space trade-offs for polynomial calculus, polynomial calculus resolution, and cutting planes. We want to point out that the results are in a sense slightly more general in that the only property of the proof systems that we use to show our trade-off results is that the correctness of each inferred line in a refutation can be checked by a small two-player communication protocol, and so our results also hold for any other proof system with this property. We interpret our results to suggest that lifted pebbling formulas should inherit time-space trade-offs properties from the graphs in terms of which they are defined, which if true would mean that the black-white pebble game in [31] could be used to obtain strong(er) trade-offs in these proof systems (as was done for resolution and k-DNF resolution in [17]).

Despite the amount of research focused on the space measure in proof complexity over the last decade, it seems fair to argue that space is still not a very well understood concept, and many fascinating (and basic) questions remain open. We highlight some such questions below.

Proving optimal lower bounds on the monomial space of refuting (preferably bounded-width) CNF formulas in PCR, or at least in polynomial calculus, is a problem that has been open for a while but might be more accessible now given recent advances in [38]. Establishing lower bounds on line space in cutting planes seems harder, since here nothing is known. However, as we have already mentioned, we believe that our results suggest that appropriate flavours of pebbling formulas should be hard with respect to space in all of these proof systems. If this could be proven, the next step would be to see if pebbling time-space trade-offs also carry over to PCR and/or cutting planes, analogously to what happens for resolution and k-DNF resolution.

It seems natural to expect that PCR and cutting planes should both be stronger than resolution with respect to space, but as far as we are aware this is open. Thus, it would be interesting to separate PCR from resolution with respect to space by finding a k-CNF formula that has low monomial space complexity in PCR but large clause space complexity in resolution, and similarly for CP with respect to resolution.

Another fairly longstanding open question, as mentioned in [1, 14] and other papers, is to prove superlinear lower bounds on total space (measured in the size of the formula) for resolution refutations of some family of k-CNF formulas (or even CNF formulas of unbounded width).

Concluding our discussion of open questions in proof complexity related to the current paper, it is hard to avoid mentioning the well-known problem of proving lower bound on

[6] Bonet et al. [22] actually proved the result for the model of *real* communication complexity which is stronger than deterministic communication complexity. However this model is nothing we are concerned with in this paper.

[7] We remark that an easier and more direct way to obtain our results in Theorem 1 would have been to apply the ideas in [10] (in particular, Theorem 3.6 in that paper). However, we have recently found a serious problem in the key Lemma 3.5 in [10], and therefore we cannot use this approach.

[8] We want to pointout that it is necessary to restrict our attention to *consistent* randomized protocols in order to get strong lower bounds in Theorem 18 (i.e., polynomially related to the deterministic decision tree complexity of the search problem). This is because for some search problems, exponential separations are known between deterministic and (non-consistent) randomized decision tree complexity [47].

proof size for cutting planes. This proof system remains very poorly understood, and it seems like a very interesting, but possibly also very challenging, open problem to prove that, for instance, random k-CNF formulas or Tseitin formulas require CP-proofs of exponential size.

Finally, let us also discuss some interesting questions in communication complexity that arise from our work on Theorem 2. An obvious open question is to improve the lower bound in that theorem. In particular, a lower bound in terms of the (non-critical) block sensitivity of the search problem would not only give a lower bound that is polynomial related to the simple upper bound in terms of decision tree complexity, but would also produce a general "black-box" hardness amplification result in proof complexity based on our approach, in the sense that lower bounds on the rank (also known as depth) in resolution for any CNF formula F would turn into into space-length tradeoffs in PCR and CP for the formula $Lift(F)$.

Another related problem is to gain more understanding of the critical block sensitivity of search problems. For instance, is the critical block sensitivity always polynomially related to the block sensitivity for any search problem? If this were true, it would imply the same hardness amplification result as mentioned above.

Acknowledgements

The first author performed part of this work while visiting KTH Royal Institute of Technology supported by the foundations *Johan och Jakob Söderbergs stiftelse*, *Stiftelsen Längmanska kulturfonden*, and *Helge Ax:son Johnsons stiftelse*. The research of the second author was supported by Swedish Research Council grant 621-2010-4797 and by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007–2013) / ERC grant agreement no 279611.

The authors would like to thank Arkadev Chattopadhyay for several enjoyable (and above all *illuminating*) discussions during the weeks spent jointly at KTH Royal Institute of Technology. We are also grateful to the anonymous referees for several comments that helped to improve the presentation of this paper.

6. REFERENCES

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

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

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

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

[5] Z. Bar-Yossef, T. Jayram, R. Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *Journal of Computer and System Sciences*, 68(4):702–732, June 2004. Preliminary version appeared in *FOCS '02*.

[6] 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)*, pages 203–208, July 1997.

[7] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf. Quantum lower bounds by polynomials. *Journal of the ACM*, 48(4):778–797, 2001.

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

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

[10] P. Beame, T. Huynh, and T. Pitassi. Hardness amplification in proof complexity. In *Proceedings of the 42nd Annual ACM Symposium on Theory of Computing (STOC '10)*, pages 87–96, June 2010.

[11] P. Beame, R. Karp, T. Pitassi, and M. Saks. The efficiency of resolution and Davis-Putnam procedures. *SIAM Journal on Computing*, 31(4):1048–1075, 2002. Preliminary versions of these results appeared in *FOCS '96* and *STOC '98*.

[12] P. Beame, T. Pitassi, and N. Segerlind. Lower bounds for Lovász–Schrijver systems and beyond follow from multiparty communication complexity. *SIAM Journal on Computing*, 37(3):845–869, 2007. Preliminary version appeared in *ICALP '05*.

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

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

[15] 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)*, volume 6175 of *Lecture Notes in Computer Science*, pages 16–29. Springer, July 2010.

[16] 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)*, pages 709–718, Oct. 2008.

[17] 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)*, pages

401–416, Jan. 2011. Full-length version available at `http://eccc.hpi-web.de/report/2010/125/`.

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

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

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

[21] 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)*, pages 575–584, May 1995.

[22] M. L. Bonet, J. L. Esteban, N. Galesi, and J. Johannsen. On the relative complexity of resolution refinements and cutting planes proof systems. *SIAM Journal on Computing*, 30(5):1462–1484, 2000. Preliminary version appeared in *FOCS '98*.

[23] M. L. Bonet and N. Galesi. Optimality of size-width tradeoffs for resolution. *Computational Complexity*, 10(4):261–276, Dec. 2001. Preliminary version appeared in *FOCS '99*.

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

[25] M. Brickenstein, A. Dreyer, G.-M. Greuel, M. Wedler, and O. Wienand. New developments in the theory of Gröbner bases and applications to formal verification. *Journal of Pure and Applied Algebra*, 213(8):1612–1635, Aug. 2009.

[26] A. Chakrabarti, Y. Shi, A. Wirth, and A. C.-C. Yao. Informational complexity and the direct sum problem for simultaneous message complexity. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science (FOCS '01)*, pages 270–278, Oct. 2001.

[27] V. Chvátal and E. Szemerédi. Many hard examples for resolution. *Journal of the ACM*, 35(4):759–768, Oct. 1988.

[28] 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)*, pages 174–183, May 1996.

[29] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing (STOC '71)*, pages 151–158, 1971.

[30] S. A. Cook and R. Reckhow. The relative efficiency of propositional proof systems. *Journal of Symbolic Logic*, 44(1):36–50, Mar. 1979.

[31] S. A. Cook and R. Sethi. Storage requirements for deterministic polynomial time recognizable languages. *Journal of Computer and System Sciences*, 13(1):25–37, 1976.

[32] W. Cook, C. R. Coullard, and G. Turán. On the complexity of cutting-plane proofs. *Discrete Applied Mathematics*, 18(1):25–38, Nov. 1987.

[33] M. David, T. Pitassi, and E. Viola. Improved separations between nondeterministic and randomized multiparty communication. *ACM Transactions on Computation Theory*, 1:5:1–5:20, Sept. 2009.

[34] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem proving. *Communications of the ACM*, 5(7):394–397, July 1962.

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

[36] J. L. Esteban, N. Galesi, and J. Messner. On the complexity of resolution with bounded conjunctions. *Theoretical Computer Science*, 321(2-3):347–370, Aug. 2004. Preliminary version appeared in *ICALP '02*.

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

[38] Y. Filmus, M. Lauria, J. Nordström, N. Thapen, and N. Zewi. Space complexity in polynomial calculus. In *Proceedings of the 27th Annual IEEE Conference on Computational Complexity (CCC '12)*, June 2012. To appear.

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

[40] R. Impagliazzo, T. Pitassi, and A. Urquhart. Upper and lower bounds for tree-like cutting planes proofs. In *Proceedings of the 9th Annual IEEE Symposium on Logic in Computer Science (LICS '94)*, pages 220–228, July 1994.

[41] R. Impagliazzo, P. Pudlák, and J. Sgall. Lower bounds for the polynomial calculus and the Gröbner basis algorithm. *Computational Complexity*, 8(2):127–144, 1999.

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

[43] J. Krajíček. On the weak pigeonhole principle. *Fundamenta Mathematicae*, 170(1-3):123–140, 2001.

[44] E. Kushilevitz and N. Nisan. *Communication complexity*. Cambridge University Press, 1997.

[45] D. Le Berre and A. Parrain. On extending SAT solvers for PB problems. In *14th RCRA workshop Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion (RCRA '07)*, July 2007. Available at `http://pst.istc.cnr.it/RCRA07/articoli/P14-leberre-parrain-RCRA07.pdf`.

[46] D. Le Berre, A. Parrain, and O. Roussel. The long way from conflict driven clause learning to conflict driven constraint learning. In *Guangzhou Symposium on Satisfiability and its applications*, Sept. 2004. Available at `http://www.cril.univ-artois.fr/spip/publications/kanton04.pdf`.

[47] L. Lovász, M. Naor, I. Newman, and A. Wigderson. Search problems in the decision tree model. *SIAM Journal on Discrete Mathematics*, 8(1):119–132, 1995. Preliminary version appeared in *FOCS '91*.

[48] J. P. Marques-Silva and K. A. Sakallah. GRASP—a new search algorithm for satisfiability. In *Proceedings*

of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD '96), pages 220–227, Nov. 1996.

[49] N. Nisan. CREW PRAMs and decision trees. *SIAM Journal on Computing*, 20:999–1007, Dec. 1991.

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

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

[52] J. Nordström. New wine into old wineskins: A survey of some pebbling classics with supplemental results. Manuscript in preparation. To appear in *Foundations and Trends in Theoretical Computer Science*. Current draft version available at `http://www.csc.kth.se/~jakobn/research/`, 2012.

[53] J. Nordström. Pebble games, proof complexity and time-space trade-offs. *Logical Methods in Computer Science*, 2012. To appear. Available at `http://www.csc.kth.se/~jakobn/research/`.

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

[55] K. Pipatsrisawat and A. Darwiche. On the power of clause-learning SAT solvers as resolution engines. *Artificial Intelligence*, 175:512–525, Feb. 2011. Preliminary version appeared in *CP '09*.

[56] N. Pippenger. Pebbling. Technical Report RC8258, IBM Watson Research Center, 1980. Appeared in Proceedings of the 5th IBM Symposium on Mathematical Foundations of Computer Science, Japan.

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

[58] R. Raz and P. McKenzie. Separation of the monotone NC hierarchy. *Combinatorica*, 19(3):403–435, Mar. 1999. Preliminary version appeared in *FOCS '97*.

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

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

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

[62] N. Segerlind. The complexity of propositional proofs. *Bulletin of Symbolic Logic*, 13(4):482–537, Dec. 2007.

[63] N. Segerlind, S. R. Buss, and R. Impagliazzo. A switching lemma for small restrictions and lower bounds for k-DNF resolution. *SIAM Journal on Computing*, 33(5):1171–1200, 2004. Preliminary version appeared in *FOCS '02*.

[64] A. Urquhart. Hard examples for resolution. *Journal of the ACM*, 34(1):209–219, Jan. 1987.