On the Interplay Between Proof Complexity and SAT Solving

JAKOB NORDSTRÖM, KTH Royal Institute of Technology

This paper is intended as an informal and accessible survey of proof complexity for non-experts, focusing on some comparatively weak proof systems of particular interest in connection with SAT solving. We review resolution, polynomial calculus, and cutting planes (related to conflict-driven clause learning, Gröbner basis computations, and pseudo-Boolean solving, respectively) and some complexity measures that have been studied for these proof systems. We also discuss briefly to what extent proof complexity could provide insights into SAT solver performance, and how concerns related to applied SAT solving can give rise to interesting complexity-theoretic questions. Along the way, we highlight a number of current research challenges.

Additional Key Words and Phrases: Proof complexity, SAT solving, resolution, polynomial calculus, cutting planes, conflict-driven clause learning, Gröbner bases, pseudo-Boolean solvers

ACM Reference Format:
Jakob Nordström, 2015. On the Interplay Between Proof Complexity and SAT Solving. *ACM Trans. Embedd. Comput. Syst.* X, Y, Article ZZ (July 2015), 26 pages.
DOI: 0000001.0000001

1. INTRODUCTION

The satisfiability problem (SAT) — i.e., to determine whether or not a given formula in propositional logic has a satisfying assignment — is a fundamental problem in theoretical computer science. SAT was proven NP-complete in [Cook 1971], and because of this the problem is widely believed to be computationally intractable in the worst case. Proving this currently looks totally out of reach — this is one of the million dollar *Millennium Problems* posed by the *Clay Mathematics Institute* — but it is probably fair to argue that the conventional wisdom in the computational complexity community is that SAT should be infeasible to solve in practice.

This message does not seem to have propagated to researchers in applied SAT solving, however. Instead, this area has seen enormous improvements in performance over the last two decades, and current state-of-the-art algorithms for deciding satisfiability — so-called *SAT solvers* — can deal with real-world instances containing millions of variables, and often run in (close to) linear time! NP-completeness did not just go away, however — it is also possible to construct tiny formulas with just a few hundred variables that are totally beyond reach for even the best solvers today.

This raises the question of how these SAT solvers work, and how they can perform so well in practice. And when they sometimes miserably fail, can one explain why?

The best current SAT solvers are based on so-called *conflict-driven clause learning (CDCL)*. Some solvers also incorporate elements of algebraic reasoning (e.g., Gaussian elimination) and/or geometric reasoning (e.g., linear inequalities), or use algebraic or

---

This paper is based on the invited presentation *A (Biased) Proof Complexity Survey for SAT Practitioners* given at the *17th International Conference on Theory and Applications of Satisfiability Testing (SAT '14)* [Nordström 2014].

Author's address: Jakob Nordström, School of Computer Science and Communication, KTH Royal Institute of Technology, SE-100 44 Stockholm, Sweden.

DOI: 0000001.0000001

geometric methods as the foundation rather than CDCL. Another augmentation of CDCL that has attracted much interest is *extended resolution*. How can we analyze the power of such algorithms? The best approach we currently have is to study the underlying methods of reasoning and what they are able or unable to do in principle (mostly ignoring constructive, algorithmic aspects). This brings us into the realm of proof complexity, an area of research initiated in [Cook and Reckhow 1979].

This survey is intended as somewhat of a crash course in proof complexity, focusing on the proof systems behind some current approaches to SAT solving. We will discuss *resolution* (corresponding to CDCL), *polynomial calculus* (corresponding to algebraic *Gröbner basis* computations) and *cutting planes* (corresponding to geometric or so-called *pseudo-Boolean* solving), and will also briefly touch on extended resolution.

Our goal is to give an overview of some of the complexity results known about these proof systems. Rather than giving precise, formal statements of theorems, our focus is on showing some of the "benchmark formulas" used to prove these theorems, since they illustrate what kind of reasoning different proof systems can or cannot do in principle. This sometimes shows fundamental limitations on what one can hope for SAT solvers to achieve, but sometimes instead can be viewed as challenges, when actual SAT solver performance does not match what theory suggests should be possible.

By necessity, this brief survey has a selective and somewhat subjective coverage of topics, and many exciting results in proof complexity are not even mentioned below. For many of the results that are mentioned, a more detailed, formal treatment can be found in the survey paper [Nordström 2013]. A more general-purpose proof complexity survey is [Segerlind 2007]. Due to space constraints, our discussion of applied SAT solving is even more limited (and sometimes slightly simplified to try to get the main message across), and we have had to omit many relevant references. A very comprehensive overview of this latter research area is given in [Biere et al. 2009].

1.1. Outline of this Survey

We review resolution in Section 2 and describe the connection to CDCL SAT solvers in Section 3. In Section 4 we discuss polynomial calculus and algebraic SAT solving, and Section 5 deals with cutting planes and geometric solvers. We comment briefly on extended resolution in Section 6. Some concluding remarks are presented in Section 7.

2. RESOLUTION

Let us start by fixing some notation and terminology:

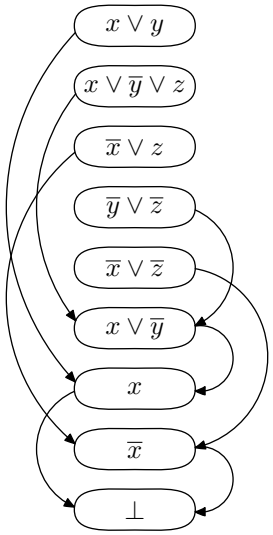
— A *literal* a is a variable x or its negation $\overline{x}$.
— A *clause* $C = a_1 \vee \cdots \vee a_k$ is a disjunction of literals.
— A *conjunctive normal form (CNF) formula* $F = C_1 \wedge \cdots \wedge C_m$ is a conjunction of clauses.
— A *k-CNF formula* is a CNF formula where all clauses contain at most k literals. Throughout this paper, k will be some fixed constant unless stated otherwise.
— We will write N to denote the *size* of a formula, which is the total number of literals in it counted with repetitions (or, for a k-CNF formula, the number of clauses up to a constant linear factor).

The general set-up is that we are given an unsatisfiable CNF formula F and want to understand how efficiently the proof system under study can certify that F is contradictory. (Satisfiable formulas always have very short certificates, namely satisfying assignments, which is why it makes sense to focus on unsatisfiable instances if we want to prove complexity results.) Such a proof of unsatisfiability is often referred to as a *refutation* of F, and we will use the two terms "proof" and "refutation" interchangeably.

We consider clauses and formulas as sets, so that there are no repetitions and order is irrelevant. In what follows, we will often tacitly assume for simplicity of exposition

1.	$x \vee y$	Axiom
2.	$x \vee \overline{y} \vee z$	Axiom
3.	$\overline{x} \vee z$	Axiom
4.	$\overline{y} \vee \overline{z}$	Axiom
5.	$\overline{x} \vee \overline{z}$	Axiom
6.	$x \vee \overline{y}$	Res$(2, 4)$
7.	x	Res$(1, 6)$
8.	$\overline{x}$	Res$(3, 5)$
9.	$\bot$	Res$(7, 8)$

(a) Annotated list of clauses in refutation.



(b) DAG representation of refutation.

Fig. 1: Resolution refutation represented as list of clauses and directed acyclic graph.

that the formulas involved are k-CNF formulas unless stated otherwise. There is a standard way to turn any CNF formula F into 3-CNF by converting every wide clause

$$a_1 \vee a_2 \vee \ldots \vee a_w \tag{1a}$$

into the set of 3-clauses

$$\{y_0\} \cup \{\overline{y}_{j-1} \vee a_j \vee y_j \mid 1 \leq j \leq w\} \cup \{\overline{y}_w\} \ , \tag{1b}$$

where the y_i's denote new variables that do not appear anywhere else. This conversion to 3-CNF most often does not change much from a theoretical point of view (though there are some notable exceptions to this rule, which we will return to later).

In the *resolution* proof system [Blake 1937], we start with clauses of the input formula (referred to as *axioms*) and derive new clauses by repeated application of the *resolution rule*

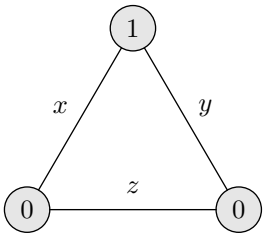
$$\frac{C \vee x \qquad D \vee \overline{x}}{C \vee D} \ . \tag{2}$$

A resolution refutation ends when empty clause $\bot$ (i.e., the clause containing no literals) has been derived. We can representent a refutation either as an annotated list of clauses as in Figure 1a or as a directed acyclic graph (DAG) as in Figure 1b. We say that a refutation is *tree-like* if this DAG is a tree (which is the case in Figure 1b).

It is straightforward to show that resolution is sound, i.e., that there is a resolution refutation of a formula F only if it is unsatisfiable (simply because the resolution rule (2) is sound), and it is also not hard to show that resolution is complete in that any unsatisfiable formula can be refuted.

2.1. Resolution Length

The *length* (also referred to as the *size*) of a resolution refutation is the number of clauses in it, counted with repetitions (which is relevant if we have a certain DAG representation of the refutation in mind, such as a tree-like refutation). In general, proof



(a) Triangle graph with odd labelling.

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

(b) Corresponding Tseitin formula.

Fig. 2: Example Tseitin formula.

length/size is the most fundamental measure in proof complexity, and for resolution length lower bounds also imply lower bounds on CDCL solver running time, since one can in principle extract a resolution proof from a CDCL execution trace.[1]

Any CNF formula of size N can be refuted in resolution length $\exp(\mathrm{O}(N))$, and there are formulas for which matching $\exp(\Omega(N))$ lower bounds are known. Let us discuss some examples of formulas known to be hard with respect to resolution length.

Our first example is the **pigeonhole principle (PHP)**, which says that "m pigeons do not fit into n holes if $m > n$." This is arguably the single most studied combinatorial principle in all of proof complexity (see [Razborov 2002] for a survey). When written as an unsatisfiable CNF formula, this becomes the claim that, on the contrary, $m > n$ pigeons do fit into n holes. To encode this, one uses variables $p_{i,j}$ to denote "pigeon i is in hole j," and write down the following clauses, where $i \neq i'$ range over $1, \ldots, m$ and $j \neq j'$ range over $1, \ldots, n$:

$$p_{i,1} \vee p_{i,2} \vee \cdots \vee p_{i,n} \qquad \text{[every pigeon } i \text{ gets a hole]} \tag{3a}$$

$$\overline{p}_{i,j} \vee \overline{p}_{i',j} \qquad \text{[no hole } j \text{ gets two pigeons } i \neq i'\text{]} \tag{3b}$$

There are also variants where one in addition has "functionality" and/or "onto" axioms

$$\overline{p}_{i,j} \vee \overline{p}_{i,j'} \qquad \text{[no pigeon } i \text{ gets two holes } j \neq j'\text{]} \tag{3c}$$

$$p_{1,j} \vee p_{2,j} \vee \cdots \vee p_{m,j} \qquad \text{[every hole } j \text{ gets a pigeon]} \tag{3d}$$

In a breakthrough result, [Haken 1985] proved that the PHP formula consisting of clauses (3a) and (3b) requires length $\exp(\Omega(n))$ in resolution for $m = n+1$ pigeons, and his proof can be extended to work also for the **onto FPHP formulas** consisting of all clauses (3a)–(3d). Later work [Raz 2004; Razborov 2003; 2004] has shown that all of the PHP formula variants remain hard even for arbitrarily many pigeons m, requiring resolution length $\exp\big(\Omega\big(n^{\delta}\big)\big)$ for some $\delta > 0$. What this means, intuitively, is that the resolution proof system really cannot count — even faced with the preposterous claim that infinitely many pigeons can be mapped in a one-to-one fashion into a some finite number n of holes, resolution cannot find a short proof to refute this claim.

Since PHP formulas have size $N = \Theta\big(n^3\big)$, Haken's lower bound is only $\exp\big(\Omega\big(\sqrt[3]{N}\big)\big)$ expressed in terms of formula size. The first truly exponential lower bound on length was obtained for **Tseitin formulas** (an example of which is shown in Figure 2), which encode (the negation of) the principle that "the sum of the vertex degrees in a graph is even." Here the variables are the edges in an undirected graph of bounded degree, where every vertex has been labelled 0 or 1 so that the sum of all labels is odd. Then for

[1] This claim ignores the issue of preprocessing, which we will touch on briefly in Section 3, but a detailed discussion of which is beyond the scope of this survey.

every vertex one writes down the set of clauses encoding that the parity of the number of true edges incident to that vertex is equal to the vertex label (see Figure 2b, which displays the formula corresponding to the labelled graph in Figure 2a).

If we sum the constraints over all vertices we should get an odd number by the construction of the labelling, but since such a sum counts each edge exactly twice it has to be even. Thus, these formulas are indeed unsatisfiable. [Urquhart 1987] established that Tseitin formulas require resolution length $\exp\bigl(\Omega\bigl(N\bigr)\bigr)$ if the underlying graph is a well-connected so-called *expander graph* (which holds asymptotically almost surely for a random regular graph of bounded degree, for instance). Intuitively, this shows that resolution also is not able to count $\bmod\, 2$ efficiently.

Another example of exponentially hard formulas are **random k-CNF formulas**, which are generated by randomly sampling Δn k-clauses over n variables for some large enough constant Δ depending on k. For instance, $\Delta \gtrsim 4.5$ is sufficient to get unsatisfiable 3-CNF formulas asymptotically almost surely. [Chvátal and Szemerédi 1988] established that resolution requires length $\exp\bigl(\Omega\bigl(N\bigr)\bigr)$ to refute such formulas.

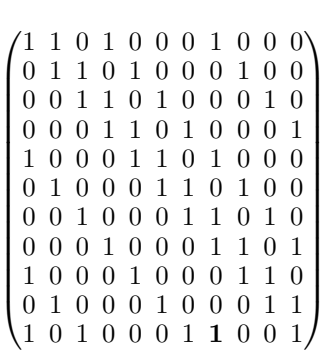
By now strong lower bounds have been shown for formulas encoding tiling problems [Alekhnovich 2004], k-colourability [Beame et al. 2005], independent sets and vertex covers [Beame et al. 2007], and many other combinatorial principles. We want to conclude our discussion of resolution length by mentioning perhaps the latest addition to this long list, namely the **subset cardinality formulas** studied in [Spence 2010; Van Gelder and Spence 2010; Mikša and Nordström 2014] (also known as **zero-one design** or **sgen** formulas).

To construct these formulas, we start with an $n \times n$ $(0,1)$-matrix with 4 non-zero entries in each row and column except that one extra non-zero entry is added to some empty cell (as in Figure 3a, where the extra 1 in the bottom row is in bold face). The variables of the formula are the non-zero entries of the matrix, yielding a total of $4n+1$ variables. For each row of 4 ones in the matrix, we write down the natural 3-CNF formula encoding the *positive cardinality constraint* that at least 2 variables must be true (as in the first set of clauses in Figure 3b), and for the row with 5 ones the 3-CNF formula encoding that a strict majority of 3 variables must be true. For the columns we instead encode *negative cardinality constraints* that the number of false variables is at least 2 and 3, respectively (see the last set of clauses in Figure 3b). The formula consisting of the conjunction of all these clauses must be unsatisfiable, since a strict majority of the variables cannot be true and false simultaneously. We will have reason to return to these formulas below when we discuss connections between CDCL and resolution, and also when discussing cutting planes and pseudo-Boolean solving.

[Spence 2010; Van Gelder and Spence 2010] showed empirically that these formulas are very hard for CDCL solvers, and [Mikša and Nordström 2014] proved the matching theoretical result that subset cardinality formulas are indeed exponentially hard if the underlying matrix is an expander (informally, if every at most medium-sized set of rows has ones in many distinct columns).

2.2. Resolution Width

A second complexity measure for resolution, that is almost as well studied as length, is *width*, measured as the size of a largest clause in a resolution refutation. It is clear that the width needed to refute a formula is never larger than the number of variables n, which is in turn less than the total formula size N. It is also easy to see that an upper bound w on resolution width implies an upper bound $\mathrm{O}\bigl(n^w\bigr)$ on resolution length, simply because the total number of distinct clauses of width at most w over n variables is less than $(3n)^w$. Incidentally, this simple counting argument turns out to be essentially tight, in that there are k-CNF formulas refutable in width w that require resolution length $n^{\Omega(w)}$, as shown by [Atserias et al. 2014].



(a) Matrix with row and column constraints.

$$
\begin{aligned}
&(x_{1,1} \vee x_{1,2} \vee x_{1,4}) \\
\wedge\ &(x_{1,1} \vee x_{1,2} \vee x_{1,8}) \\
\wedge\ &(x_{1,1} \vee x_{1,4} \vee x_{1,8}) \\
\wedge\ &(x_{1,2} \vee x_{1,4} \vee x_{1,8}) \\
&\ \vdots \\
\wedge\ &(\overline{x}_{4,11} \vee \overline{x}_{8,11} \vee \overline{x}_{10,11}) \\
\wedge\ &(\overline{x}_{4,11} \vee \overline{x}_{8,11} \vee \overline{x}_{11,11}) \\
\wedge\ &(\overline{x}_{4,11} \vee \overline{x}_{10,11} \vee \overline{x}_{11,11}) \\
\wedge\ &(\overline{x}_{8,11} \vee \overline{x}_{10,11} \vee \overline{x}_{11,11})
\end{aligned}
$$

(b) Example cardinality constraints in CNF.

Fig. 3: Matrix and (fragment of) corresponding subset cardinality formula.

Much less obviously, however, and much more interestingly, strong enough width lower bounds imply strong length lower bounds. [Ben-Sasson and Wigderson 2001] showed that for a k-CNF formula of size N it holds that

$$\text{refutation length} \geq \exp\left(\Omega\left(\frac{(\text{refutation width})^2}{N}\right)\right) \ . \tag{4}$$

This means that if one can prove that a formula requires width $\omega\big(\sqrt{N \log N}\big)$, this immediately yields a superpolynomial length lower bound, and a width lower bound $\Omega(N)$ implies a truly exponential $\exp(\Omega(N))$ length lower bound. Almost all known lower bounds on resolution length can be derived via width lower bounds in this way (in particular, all the bounds discussed in Section 2.1, although the ones predating [Ben-Sasson and Wigderson 2001] were originally not obtained in this way).

For tree-like resolution, [Ben-Sasson and Wigderson 2001] proved a sharper version

$$\text{tree-like refutation length} \geq 2^{\text{refutation width}} \tag{5}$$

of their bound in (4) for general resolution. This means that for tree-like resolution, even width lower bounds $\omega\big(\log N\big)$ yield superpolynomial length lower bounds. For general resolution, however, a width lower bound even as large as $\Omega\big(\sqrt{N \log N}\big)$ does not imply anything about length. This raises the question of whether it is possible to strengthen (4) to something closer to (5) also for general resolution. [Bonet and Galesi 2001] showed that this is not the case by studying another interesting combinatorial benchmark formula, which we describe next.

The **ordering principle** says that "every finite (partially) ordered set $\{e_1, \ldots, e_n\}$ has a minimal element." To encode the negation of this statement in CNF, we use variables $x_{i,j}$ to denote "$e_i < e_j$" and write down the following clauses (for $i \neq j \neq k \neq i$ ranging over $1, ..., n$):

$$\overline{x}_{i,j} \vee \overline{x}_{j,i} \qquad [\text{anti-symmetry; not both } e_i < e_j \text{ and } e_j < e_i] \tag{6a}$$

$$\overline{x}_{i,j} \vee \overline{x}_{j,k} \vee x_{i,k} \qquad [\text{transitivity; } e_i < e_j \text{ and } e_j < e_k \text{ implies } e_i < e_k] \tag{6b}$$

$$\bigvee_{1 \leq i \leq n,\, i \neq j} x_{i,j} \qquad [e_j \text{ is not a minimal element}] \tag{6c}$$

One can also add axioms

$$x_{i,j} \vee x_{j,i} \qquad [\text{totality; either } e_i < e_j \text{ or } e_j < e_i] \tag{6d}$$

to specify that the ordering is total. This yields a formula over $\Theta(n^2)$ variables of total size $N = \Theta(n^3)$. (We remark that variants of this formula also appear under the name **least number principle formula** or **graph tautology** in the literature.)

[Krishnamurthy 1985] conjectured that these formulas should be hard for resolution, but [Stålmarck 1996] showed that they are refutable in length $\mathrm{O}(N)$ (even without the clauses (6d)). As the formula is described above, it does not really make sense to ask about the refutation width, since already the axiom clauses (6c) have unbounded width. However, one can convert the formula to 3-CNF by applying the transformation from (1a) to (1b) to the wide axioms (6c), and for this version of the formula [Bonet and Galesi 2001] established a width lower bound $\Omega(\sqrt[3]{N})$ (which is tight, and holds even if the axioms (6d) are also added). This shows that even polynomially large resolution width does not necessarily imply any length lower bounds for general resolution.

2.3. Resolution Space

The study of the space complexity of proofs, which was initiated in the late 1990s, was originally motivated by considerations of SAT solver memory usage, but has also turned out to be of intrinsic interest for proof complexity. Space can be measured in different ways — here we focus on the most well studied measure of *clause space*, which is the maximum number of clauses needed in memory while verifying the correctness of a refutation.[2] Thus, in what follows below "space" will always mean "clause space."

The space usage of a resolution refutation at step t is the number of clauses at steps $\leq t$ that are used at steps $\geq t$. Returning to our example refutation in Figure 1, the space usage at step 7 is 5 (the clauses in memory at this point are clauses 1, 3, 5, 6, and 7). The space of a proof is obtained by measuring the space usage at each step and taking the maximum. Phrased differently, one can view the formula as being stored on a read-only input tape, from where clauses can be read into working memory. The resolution rule can only be applied to clauses currently in working memory, and there is no way to store clauses "on disk" — once they are erased from working memory, they are gone. Then space measures how many clauses are used in working memory to perform the resolution refutation. Incidentally, it is not hard to see that the proof in Figure 1 is not optimal when it comes to minimizing space. We could do the same refutation in space 4 instead by processing the clauses in the order 2, 4, 6, 1, 7, 3, 5, 8, 9.

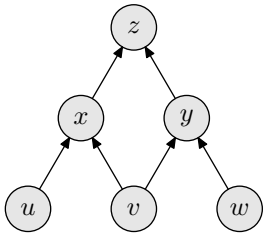
Perhaps somewhat surprisingly, an unsatisfiable CNF formula of size N can always be refuted in resolution space at most $N + \mathrm{O}(1)$, as shown by [Esteban and Torán 2001],[3] though the refutation thus obtained might have exponential length. Lower bounds on space were subsequently shown for PHP formulas and Tseitin formulas [Alekhnovich et al. 2002; Esteban and Torán 2001] and for random k-CNFs [Ben-Sasson and Galesi 2003]. For the latter two formula families the (optimal linear) lower bounds matched exactly previously known width lower bounds, and also the proof techniques had a very similar flavour. This led to the question of whether there was some deeper connection hidden here.

In a very elegant paper, [Atserias and Dalmau 2008] confirmed this by showing that the inequality

$$\text{refutation space} \geq \text{refutation width} + \mathrm{O}(1) \tag{7}$$

[2]Note, though, that this measure underestimates the actual memory usage, since storing a clause requires more than a constant amount of memory. For completenes, we mention that there is also a measure *total space*, counting the total number of literals in memory (with repetitions), which has been studied in [Alekhnovich et al. 2002; Bonacina et al. 2014; Bennett et al. 2015].

[3]This space upper bound is obtained by simply running CDCL (or even DPLL) as described in Section 3 with some arbitrary but fixed variable ordering.



(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}.

Fig. 4: Example pebbling contradiction for the pyramid of height 2.

holds for resolution refutations of k-CNF formulas. The proof of (7) is beautiful but uses a somewhat non-explicit argument based on finite model theory. A more explicit proof, which works by simple syntactic manipulations to construct a small-width refutation from a small-space refutation, was presented in [Filmus et al. 2015a].

Since for all formulas studied up to [Atserias and Dalmau 2008] the width and space complexity measures turned out to actually coincide, it is natural to ask whether (7) could be strengthened to an asymptotic equality. The answer to this question is no. As shown in the sequence of works [Nordström 2009; Nordström and Håstad 2013; Ben-Sasson and Nordström 2008], there are formulas that can be refuted in width $\mathrm{O}(1)$ but require space $\Omega(N/\log N)$ (i.e., formulas that are maximally easy for width but exhibit worst-case behaviour for space except for a log factor).

These formulas are **pebbling contradictions** encoding so-called pebble games on bounded fan-in DAGs, which for the purposes of this discussion we additionally require to have a unique sink. In the "vanilla version" of the formula (illustrated in Figure 4), there is one variable associated to each vertex and clauses encoding that

— the source vertices are all true;
— if all immediate predecessors are true, then truth propagates to the successor;
— but the sink is false.

There is an extensive literature on pebbling space and time-space trade-offs from the 1970s and 80s, pebbling contradictions have been useful before in proof complexity in various contexts, e.g., in [Raz and McKenzie 1999; Bonet et al. 2000; Ben-Sasson and Wigderson 2001]. Since pebbling contradictions can be shown to be refutable in constant width but there are graphs for which the pebble game requires large space, one could hope that the pebbling properties of such DAGs would somehow carry over to resolution refutations of pebbling formulas and help us separate space and width.

Unfortunately, this hope cannot possibly materialize — a quick visual inspection of Figure 4b reveals that this is a Horn formula (i.e., having at most one positive literal in each clause), and such formulas are maximally easy for length, width, and space simultaneously.[4] However, we can modify these formulas by substituting for every variable x an exclusive or $x_1 \oplus x_2$ of two new variables, and then expand to CNF in the canonical way to get a new formula. This is perhaps easiest to explain by an example. Performing this substitution in the clause

$$\overline{x} \vee y \tag{8a}$$

[4]Jumping a bit ahead to Section 3 again, this is because Horn formulas are decided by unit propagation.

we obtain the formula

$$\neg(x_1 \oplus x_2) \vee (y_1 \oplus y_2) \ , \tag{8b}$$

which when expanded out to CNF becomes

$$\begin{aligned} &(x_1 \vee \overline{x}_2 \vee y_1 \vee y_2) \\ \wedge\ &(x_1 \vee \overline{x}_2 \vee \overline{y}_1 \vee \overline{y}_2) \\ \wedge\ &(\overline{x}_1 \vee x_2 \vee y_1 \vee y_2) \\ \wedge\ &(\overline{x}_1 \vee x_2 \vee \overline{y}_1 \vee \overline{y}_2) \ . \end{aligned} \tag{8c}$$

With this XOR-substitution, it turns out that the pebbling contradiction inherits the time-space trade-offs of the pebbling DAG in terms of which it is defined [Ben-Sasson and Nordström 2008; 2011] (and there is nothing magical with XOR — this can be shown to work also for substitution with other Boolean functions that have the "right properties"). Now the strong space-width separation described above is obtained by plugging in the pebbling DAGs studied in [Paul et al. 1977; Gilbert and Tarjan 1978].

2.4. Resolution Trade-offs

In the preceding sections, we have seen that for all the complexity measures of length, width, and space there are formulas which are maximally hard for these measures. Suppose, however, that we are given a formula that is guaranteed to be *easy* for two or more of these measures. Can we then find a resolution refutation that optimizes these complexity measures simultaneously? Or are there trade-offs, so that minimizing one measure must cause a sharp increase in the other measure?

The first result along these lines was reported in [Ben-Sasson 2009], where a strong space-width trade-off was established. Namely, there are formulas for which

— there are refutations in width $\mathrm{O}(1)$;
— there are also refutations in space $\mathrm{O}(1)$;
— but optimizing one measure causes (essentially) worst-case behaviour for the other.

This holds for the "vanilla version" of the pebbling contradictions in Figure 4b (if one again uses the graphs studied in [Paul et al. 1977; Gilbert and Tarjan 1978]).

Regarding trade-offs between length and space, it was shown in [Ben-Sasson and Nordström 2011; Beame et al. 2012; Beck et al. 2013] that there are formulas that

— can be refuted in short length;
— can be refuted in small space;
— but even slightly optimizing one measure causes a dramatic blow-up for the other.

This holds for substituted pebbling formulas over DAGs with strong time-space trade-offs (as in, e.g., [Lengauer and Tarjan 1982]) and for Tseitin formulas over long, narrow rectangular grids.[5]

For length versus width, we know that short refutation length implies small refutation width by (4). The proof of this inequality works by transforming a given short refutation into a narrow one, but the length blows up exponentially in the process. Recently, [Thapen 2014] established that this blow-up is unavoidable by exhibiting formulas for which there exist resolution refutations in short length, but for which any refutation in width as guaranteed by (4) has to be exponentially long. These formulas are slightly tricky to describe, however, and so we do not do so here. A technical issue

[5]To be precise, the results in [Beck et al. 2013] require that one adds two copies of every edge in the grid graph, which corresponds to XOR-substitution in the Tseitin formula, but it can be shown that this substitution can be eliminated with some extra work.

with Thapen's result is that for all other trade-offs discussed above there are k-CNF formulas that exhibit this behaviour, but Thapen's formulas have clauses of logarithmic width. It would be nice to bring this down to constant width if possible.[6]

3. CONNECTIONS BETWEEN RESOLUTION AND CONFLICT-DRIVEN CLAUSE LEARNING

To make the connection between the resolution proof system and conflict-driven clause learning (CDCL) SAT solvers, let us start by describing the *Davis-Putnam-Logemann-Loveland (DPLL) method* [Davis and Putnam 1960; Davis et al. 1962] that lies at the heart of these solvers. Given a CNF formula F, this recursive method works as follows:

(1) If F contains an empty clause, return $UNSAT$.
(2) If F is empty, return SAT.
(3) Else *decide* on some variable x and set it to $true$, and make a recursive call with all satisfied clauses removed and all occurrences of $\overline{x}$ in other clauses removed.
(4) Set x to $false$, remove satisfied clauses and occurrences of x, and make a second recursive call.
(5) Return $UNSAT$ if both recursive calls returned $UNSAT$; otherwise return SAT.

How to pick the variable x in step (3) is a nontrivial question in general, and there are sophisticated *decision heuristics* for this, the most popular of which for CDCL solvers is *VSIDS* [Moskewicz et al. 2001] (which stands for *variable state independent decaying sum*). There is one simple special case, however: if F contains a *unit clause*, meaning a clause with only one literal (i.e., all other literals have been falsified by previous assignments), then we can skip the case analysis in (3) and (4) — the literal has to be set to true in order not to falsify the formula. This kind of forced assignment is known as *unit propagation*, and modern SAT solvers try to choose decision variables so that 99% of assignments (literally) will be unit propagations.

It is not hard to show that the search tree generated by a DPLL solver as described above can be viewed as a tree-like resolution refutation. This means that the running time of DPLL is lower-bounded by the minimum length in tree-like resolution. One problem with this is that the same work can be duplicated in different subtrees. If the DPLL solver makes a few unfortunate variable decisions at the start of the search, then it can spend a lot time exploring all combinations of assignments to the other variables in an exponential number of subtrees, when in fact the right thing to do would have been to jump back to the top and try some other assignments there.

The idea behind CDCL [Bayardo Jr. and Schrag 1997; Marques-Silva and Sakallah 1999; Moskewicz et al. 2001] is that the SAT solver should avoid such duplication of work by performing a *conflict analysis* when reaching a falsifying assignment and learn a *reason* for the failure in the form of a new clause C, which is added to the formula F. After this the solver has to unset variables until C is no longer falsified, and in general this might involve not just the latest decision but a whole sequence of previous decisions. This means that the solver will typically not just backtrack one level but *backjump* several levels. The clause C will then make sure that the SAT solver does not go into the same search subtree again. There are well-developed heuristics for how to learn clauses, and the most popular *learning scheme* is known as *1UIP* (an abbreviation of *first unique implication point*). All of these learning schemes are essentially resolution derivations with very particular structural constraints.

Another feature of CDCL solvers is that they frequently clear the list of variable assignments and start the search all over again when their *restart policy* says so.

[6]We also want to mention in this context that in a recent, very intriguing work [Razborov 2015] obtained doubly exponential size-width trade-offs in tree-like resolution (this is measured in the number of variables in the formulas, which have exponential size and polynomial width).

Restarts turn out to be very important in practice, although exactly why is poorly understood. Furthermore, since learned clauses accumulate very rapidly, solvers have a *clause deletion policy* that aggressively removes most of the learned clauses at regular intervals (up to 95% of the clauses over a complete run of the solver). A solver implementing all of the ideas above that has been very influential for later developments is *MiniSat* [Eén and Sörensson 2004], which was enhanced by [Audemard and Simon 2009] with the currently most successful heuristic for measuring which clauses to keep or delete (which can be viewed as a generalized clause width measure).

Finally, a very important aspect of modern solvers is that before the main algorithm even starts, extensive *preprocessing* of the input is performed, using a number of techniques that are known to be theoretically very bad in the worst case, but are completely indispensable in practice. Some solvers, such as the current *SAT competitions*[7] champion *Lingeling* [Biere 2010], even interleave preprocessing and CDCL search, which is known as *inprocessing* [Järvisalo et al. 2012a]. Most, though not all, of these pre-/inprocessing techniques can also be formalized within resolution.

3.1. CDCL Versus Resolution

The resolution refutations found by CDCL solvers have a very specific structure. Every learned clause is derived by *trivial resolution*, where one keeps just one clause and resolves it sequentially with clauses currently in the clause database (i.e., axiom clauses and previously learned clauses). Moreover, the clauses used are exactly those triggering unit propagation, resolved over in reverse order of the propagations. What this means is that these refutations will look like long, sparse chains, which are tree-like except for non-local edges from previously learned clauses. This is very different from general resolution proofs, which can have very "bushy" DAG-like structure.

Since CDCL is only looking for structurally very restricted proofs, it is natural to ask how efficient CDCL proof search can be compared to the best possible general resolution proof. As mentioned above, the DPLL method corresponds to tree-like resolution, which can be exponentially worse than general resolution (see, e.g., [Ben-Sasson et al. 2004]). What about CDCL? Is it also asymptotically weaker than resolution, or could it be the case that CDCL implements efficient proof search in the sense that the method is never more than polynomially worse, say, than the shortest resolution proof?

This is a hard question, and we cannot quite expect a fully constructive affirmative answer since this would lead to collapses in parameterized complexity [Alekhnovich and Razborov 2008]. A line of works including [Beame et al. 2004; Hertel et al. 2008; Buss et al. 2008] culminated in the paper [Pipatsrisawat and Darwiche 2011] showing that CDCL *viewed as a proof system* polynomially simulates resolution, a result that holds for CDCL solvers with any reasonable learning scheme and restart policy. The non-constructive part is in the decision strategy, which needs to be chosen in a specific way to make the simulation go through. One possible way of interpreting this result might be that if the decision heuristic is good enough, then CDCL solvers at least have the potential to run fast on any formulas that possess short resolution proofs.

In independent work, [Atserias et al. 2011] obtained an alternative, constructive version of this result by showing that if a formula F has a resolution refutation in bounded width, then CDCL using a decision strategy with enough randomness will decide F efficiently. At first sight this might not seem so impressive — after all, exhaustive search in bounded width also runs fast — but the point is that a CDCL solver is very far from doing exhaustive width search and does not care at all about the existence or non-existence of narrow refutations.

[7] See http://www.satcompetition.org.

The downside of both of these results is that they crucially need that the clause deletion policy is never to delete clauses. As should be clear from the discussion above, this is a very unrealistic assumption. It would be nice to extend the model of CDCL in [Atserias et al. 2011; Pipatsrisawat and Darwiche 2011] to capture memory usage in a more realistic way, and then study the question of whether CDCL can simulate resolution efficiently with respect to both time and space. A question that seems particularly interesting in this context is whether something like the theoretical time-space trade-offs in Section 2.4 could show up also in practice. The lower bounds in these trade-offs hold also in the CDCL model, but the question is whether CDCL could find resolution proofs achieving the matching upper bounds, or whether the DAG structure of these proofs are beyond anything that CDCL could possibly produce.

3.2. Theoretical Complexity Measures and Hardness in Practice

The next topic we want to discuss is whether practical hardness for CDCL is in any way related to the complexity measures of resolution length, width, and space. One interesting observation in this context is that it follows from the results reviewed in Section 2 — if we "normalize" length by taking a logarithm since it can be exponential in the formula size N whereas the worst-case upper bounds for width and space are linear — that for any k-CNF formula the inequalities

$$\log(\text{refutation length}) \lesssim \text{refutation width} \lesssim \text{refutation space} \tag{9}$$

hold. Thus, length, width, and space form a hierarchy of increasingly strict hardness measures. Let us briefly discuss the measures again in this light:

— We know that length provides a lower bound on CDCL running time[8] and that CDCL polynomially simulates resolution [Pipatsrisawat and Darwiche 2011]. However, the results in [Alekhnovich and Razborov 2008] suggest that short resolution proofs should be intractable to find in the worst case.
— Regarding width, searching for proofs in small width is apparently a well-known heuristic in the AI community, and [Atserias et al. 2011] proved that CDCL should run fast if such proofs exist.
— As to space, memory consumption is an important bottleneck for SAT solvers in practice, and space complexity results provide lower bounds on CDCL clause database size. One downside of this is that the bounds can be at most linear, and the solver would certainly use a linear amount of memory just to store the input. However, it is important to note that the space lower bounds hold even in a model where the solvers knows *exactly* which clauses it needs to keep. It could be argued that in reality probably much more memory than the bare minimum should be needed.

Are width or even space lower bounds relevant indicators of CDCL hardness? Or could it be true in practice that CDCL does essentially as well as resolution with respect to length/running time? These are not mathematically well-defined questions, since CDCL solvers are a moving target, but perhaps it may still be possible to perform experiments and draw interesting conclusions? Such an approach was proposed by [Ansótegui et al. 2008], and [Järvisalo et al. 2012b] performed what seems to have been the first systematic attempt to implement this program.

In view of the discussion above it seems too optimistic that length complexity should be a reliable indicator of CDCL hardness. [Järvisalo et al. 2012b] therefore focused on comparing width and space by running extensive experiments on formulas with constant width complexity (and linear length complexity) but varying space complexity to see whether running time correlated with space. These experiments produced lots

[8] Except if some non-resolution-based preprocessing techniques happen to be very successful.

of interesting data, but it seems fair to say that the results are inconclusive. For some families of formulas the correlation between running time and space complexity looks very nice, but for other formulas the results seem quite chaotic.

In fact, one problem is that formulas with low width complexity and varying space complexity are hard to find — pretty much the only known examples are the substituted pebbling formulas discussed in Section 2.3. Thus, it is not even clear whether the experiments measured differences in width and space or some other property specific to these particular formulas. This problem seems inherent, however. One cannot just pick arbitrary benchmark formulas and compute the width and space complexity for them before running experiments, since deciding width is EXPTIME-complete [Berkholz 2012] and deciding space appears likely to be PSPACE-complete.

As a general comment, CDCL solver performance on combinatorial benchmark formulas is sometimes quite surprising. For instance, it seems to be folklore that if one wants to solve PHP formulas as quickly as possible, it is better to switch off the advanced heuristics that are otherwise very important for performance. For the partial ordering principle formulas in (6a)–(6c), it turns out that they are very sensitive to settings in the VSIDS decision heuristics, with small differences seemingly making all the difference between linear and exponential running time. And sometimes theoretically easy formulas are much harder than provably hard ones! For instance, for very regular matrices, such as the one in Figure 3a, subset cardinality formulas are easy even for tree-like resolution. However, even small such instances are superhard for CDCL solvers in practice (and harder than instances generated from random matrices). Again, this seems to depend on the variable decision heuristic — keeping everything else at default settings but swapping out VSIDS for a good fixed-variable decision order (based on the structure of the matrix) makes the solver run very fast.

Although CDCL performance on crafted combinatorial instances admittedly might not be immediately relevant for large-scale real-world instances,[9] it nevertheless seems that explanations of the above phenomena could lead to a better understanding of CDCL, which is a question that is of great interest also in the applied SAT community. One important difference from industrial instances is that combinatorial benchmarks are easy to scale up and down to study the asymptotics of SAT solver behaviour, and we believe that a more systematic study of formulas such as those reviewed in Section 2 could potentially yield important insights.

4. POLYNOMIAL CALCULUS

Polynomial calculus was introduced in [Clegg et al. 1996], but we describe below a slightly modified version from [Alekhnovich et al. 2002].[10] In this proof system we interpret clauses as polynomials in the ring $\mathbb{F}[x, \overline{x}, y, \overline{y}, z, \overline{z}, \ldots]$, where $x, \overline{x}, y, \overline{y}, z, \overline{z}, \ldots$ are all formally distinct variables and $\mathbb{F}$ is a field (which would be $\mathrm{GF}(2)$ in practical SAT solving applications but can be any field from the point of view of proof complexity).

In the context of polynomial calculus we identify 0 with true and 1 with false and translate a clause such as $x \vee y \vee \overline{z}$ to the polynomial equation $xy\overline{z} = 0$, so that clauses evaluating to true corresponds to polynomials vanishing. To prove that a CNF formula is unsatisfiable, we want to show that the equations obtained from the clauses are inconsistent.

It is important to observe, however, that from an algebraic point of view x and $\overline{x}$ are independent variables,[11] and also variables can take as values any elements in the

[9] Though formulas with a "crafted flavour" turn up in, e.g., circuit verification and cryptographic problems.
[10] This modified version is known as *polynomial calculus (with) resolution* or *PCR* in the literature.
[11] In fact, in practical applications there would be no formal variables $\overline{x}, \overline{y}, \overline{z}, \ldots$, but they allow for a cleaner theoretical treatment of the proof system.

$$\frac{x \vee \overline{y} \vee z \qquad \overline{y} \vee \overline{z}}{x \vee \overline{y}}$$

(a) Resolution step.

$$\dfrac{\dfrac{\boldsymbol{x\overline{y}z = 0} \qquad \dfrac{\dfrac{\boldsymbol{\overline{y}z = 0}}{x\overline{y}z = 0} \qquad \dfrac{z + \overline{z} - 1 = 0}{\overline{y}z + \overline{y}\overline{z} - \overline{y} = 0}}{x\overline{y}z + x\overline{y}\overline{z} - x\overline{y} = 0}}{-x\overline{y}z + x\overline{y} = 0}}{\boldsymbol{x\overline{y} = 0}}$$

(b) Corresponding polynomial calculus derivation.

Fig. 5: Example of simulation of resolution step by polynomial calculus.

field $\mathbb{F}$. Hence, we need to add constraints enforcing $0/1$ assignments that in addition respect the meaning of negation. This leads to the following set of derivation rules:

$$\textit{Boolean axioms} \quad \frac{}{x^2 - x = 0} \tag{10a}$$

$$\textit{Negation} \quad \frac{}{x + \overline{x} - 1 = 0} \tag{10b}$$

$$\textit{Linear combination} \quad \frac{p = 0 \qquad q = 0}{\alpha p + \beta q = 0} \ (\alpha, \beta \in \mathbb{F}) \tag{10c}$$

$$\textit{Multiplication} \quad \frac{p = 0}{xp = 0} \ (x \text{ any variable}) \tag{10d}$$

A polynomial calculus refutation ends when $1 = 0$ has been derived, showing that the polynomial equations have no common root, or equivalently that no assignment can simultaneously satisfy all the clauses. Polynomial calculus is sound and complete, not only for CNF formulas but for inconsistent systems of polynomial equations in general.

To define the complexity measures of *size*, *degree*, and *space*, we write out polynomials in a refutation as linear combinations of monomials, where we note that without loss of generality we can assume that all polynomials are multilinear (because of the Boolean axioms (10a)). Then the *size* of a refutation, which is the analogue of resolution length, is the total number of monomials in the refutation (counted with repetitions), the *degree*, corresponding to resolution width, is the largest degree of any monomial in it, and the *(monomial) space*, which is the analogue of resolution (clause) space, is the maximal number of monomials in memory at any point during the refutation (again counted with repetitions). One can also define a *length* measure for polynomial calculus, which is the number of derivation steps, but this can be exponentially smaller than the size, which is the more relevant measure to study here.

4.1. Polynomial Calculus and Resolution

Polynomial calculus can simulate resolution efficiently with respect to length/size, width/degree, and space simultaneously simply by mimicking refutations step by step. This means that all worst-case upper bounds for resolution immediately carry over to polynomial calculus. For an example of how this works, see the simulation of the resolution step in Figure 5a by the derivation in Figure 5b, where the equations corresponding to the simulated clauses are in bold face.

Polynomial calculus can be strictly stronger than resolution with respect to size and degree. For instance, over $\mathrm{GF}(2)$ it is not hard to see that Tseitin formulas can be refuted in size $\mathrm{O}(N \log N)$ and degree $\mathrm{O}(1)$ by doing Gaussian elimination. Another example is the set of onto FPHP formulas (3a)–(3d), which [Riis 1993] showed to be easy. It remains open, however, whether such separations can also be found for space.

OPEN PROBLEM 1. *Prove (or disprove) that polynomial calculus is strictly stronger than resolution with respect to space.*

4.2. Polynomial Calculus Size and Degree

A lot of what is known about length versus width in resolution carries over to size versus degree in polynomial calculus. A degree upper bound d implies a size upper bound $n^{\mathrm{O}(d)}$ for refuting formulas over n variables by [Clegg et al. 1996], which is qualitatively similar to the bound for resolution although the argument is a bit more involved. Just as for resolution, this bound is essentially tight by [Atserias et al. 2014].

In the other direction, a lower bound on size in terms of degree exactly analogous to the bound (4) for resolution holds for polynomial calculus as shown in [Impagliazzo et al. 1999]. Interestingly, this paper is a precursor to [Ben-Sasson and Wigderson 2001], and although it was far from obvious at the time, it turns out that one can run exactly the same proof for both of these bounds. As for resolution, the ordering principle formulas in (6a)–(6d) witness the optimality of this size-degree lower bound, as shown by [Galesi and Lauria 2010]. As for resolution, almost all size lower bounds are derived via degree lower bounds, but obtaining degree lower bounds seems much harder than width lower bounds and the machinery is much less well developed.

With the exception of Tseitin formulas and onto FPHP formulas, all the formulas in Section 2.1 are equally hard also with respect to polynomial calculus size, which can be shown via degree lower bounds arguments:

— Hardness of the standard CNF encoding (3a)–(3b) of PHP formulas[12] follows from [Alekhnovich and Razborov 2003], with some earlier work on other non-CNF encodings in [Razborov 1998; Impagliazzo et al. 1999]. The proof in [Alekhnovich and Razborov 2003] works also if onto clauses (3d) are added, and recently [Mikša and Nordström 2015] showed that FPHP formulas with clauses (3a)–(3c) are also hard (whereas with both onto and functionality axioms added the formulas are easy, as noted above).
— Strong degree and size lower bounds on random k-CNF formulas were shown by [Ben-Sasson and Impagliazzo 1999] for polynomial calculus over fields of characteristic distinct from 2, and lower bounds in any characteristic including 2 were established by different methods in [Alekhnovich and Razborov 2003].
— For the subset cardinality formulas in Figure 3 [Mikša and Nordström 2014] also proved polynomial calculus degree and size lower bounds.
— Also, "Tseitin formulas with the wrong modulus" are hard — one can define Tseitin-like formulas encoding counting modulo primes q, and such formulas are hard over fields of characteristic $p \neq q$ [Buss et al. 2001; Alekhnovich and Razborov 2003].

4.3. Polynomial Calculus Space

Recall that for resolution we measure space as the number of clauses in memory, and since clauses turn into monomials in polynomial calculus the natural analogue here is monomial space. The first monomial space lower bounds were shown for PHP formulas in [Alekhnovich et al. 2002]. These formulas have wide axioms, however, and if one applies the 3-CNF conversion from (1a) to (1b) the lower bound proof breaks down.

Monomial space lower bounds for formulas of bounded width were proven only in 2012 (journal version in [Filmus et al. 2015b]) for an obfuscated 4-CNF version of PHP formulas. This was followed by optimal, linear lower bounds for random 4-CNF formulas [Bonacina and Galesi 2015], and then for Tseitin formulas over expanders but with the added assumptions that either these graphs are sampled randomly or one adds two

[12]Here a twist is needed since these formulas have high initial degree, but we will not go into this.

copies of every edge to get a multigraph [Filmus et al. 2013].[13] Somewhat intriguingly, none of these papers could show any nontrivial lower bounds for any 3-CNF formulas. This barrier was finally overcome by [Bennett et al. 2015], who proved optimal lower bounds on random 3-CNFs. However, the following open problems show that we still do not understand polynomial calculus space very well.

OPEN PROBLEM 2. *Prove polynomial calculus space lower bounds (optimal, linear bounds, or even any bounds) for Tseitin formulas over d-regular expander graphs for $d = 3$ or even $d > 3$ using no other assumptions than expansion only.*

OPEN PROBLEM 3. *Prove that PHP formulas require large monomial space in polynomial calculus even when converted to 3-CNF.*

Another intriguing question is whether an analogue of the lower bound (7) on space in terms of width in resolution holds for k-CNF formulas also for polynomial calculus.

OPEN PROBLEM 4. *Is it true that space $\geq$ degree $+ \mathrm{O}(1)$ in polynomial calculus?*

This last problem remains wide open, but [Filmus et al. 2013] made what can be described as some limited progress by showing that if a formula requires large resolution width (which is a necessary, but not sufficient, condition for high degree), then the XOR-substituted version (as in (8a)–(8c)) requires large space. When applied to Tseitin-like formulas over expander graphs, this result yields an optimal separation of space and degree. Namely, it follows that these formulas can be refuted in degree $\mathrm{O}(1)$ but require space $\Omega(N)$. To obtain such separations we have to commit to a finite characteristic p of the underlying field, however, and the formulas encoding counting $\mathrm{mod}\ p$ will separate space and degree only for fields of this characteristic. It would be nice to get a separation that would work in any characteristic, and the candidate formulas to obtain such a result readily present themselves.

OPEN PROBLEM 5. *Prove (or disprove) that substituted pebbling formulas require monomial space lower-bounded by the pebbling space of the underlying DAG (which if true would yield a space-degree separation independent of the field characteristic).*

4.4. Polynomial Calculus Trade-offs

When it comes to trade-offs in polynomial calculus we again recognize most of the picture from resolution, but there are also some differences:

— For space versus degree we know strong, essentially optimal trade-offs from [Beck et al. 2013], and the formulas exhibiting such trade-offs are the same vanilla pebbling contradictions as for resolution (for which we get exactly the same bounds).
— [Beck et al. 2013] also showed strong size-space trade-offs, and again the formulas used are pebbling contradictions over appropriate DAGs and Tseitin formulas over long, skinny grids. Here there is some loss in parameters as compared to resolution, however, which seems to be due to limitations of the proof techniques rather than to actual differences in formula behaviour.
— We do *not* yet know for sure whether the size blow-up in [Impagliazzo et al. 1999] when degree is decreased, is necessary, however, since the analysis in [Thapen 2014] works only for resolution (at least so far). This leads to the final open problem about polynomial calculus that we want to highlight in this section.

OPEN PROBLEM 6. *Are there size-degree trade-offs in polynomial calculus in the sense that size has to blow up when degree is decreased in [Impagliazzo et al. 1999]?*

[13] It is worth noting that these space lower bounds hold for any characteristic, so although Tseitin formulas have small-size refutations over $\mathrm{GF}(2)$, such refutations still require large space.

4.5. Algebraic SAT Solving

We conclude this section with a brief discussion of algebraic SAT solving. There seems to have been quite some excitement, at least in the theory community, about the Gröbner basis approach to SAT solving after the paper [Clegg et al. 1996] appeared. However, the improvement in performance that this method seemed to promise failed to materialize in practice. Instead, the CDCL revolution happened ...

Today there are some Gröbner basis SAT solvers such as *PolyBoRi* [Brickenstein and Dreyer 2009; Brickenstein et al. 2009], but they do not seem competitive with resolution-based solvers. Some SAT solvers such as *March* successfully implement Gaussian elimination [Heule and van Maaren 2005], but this is only a very limited form of polynomial calculus reasoning.

Is it harder to build good algebraic SAT solvers than CDCL solvers? Or is it just that too little work has been done? (Witness that it took 40 years for resolution-based SAT solvers to become really efficient.) Or is it perhaps a little bit of both?

Whatever the answer is to these questions, it seems clear that one needs to find some kind of shortcut to use Gröbner bases for efficient SAT solving. A full Gröbner basis computation does too much work, since it not only decides satisfiability but yields the number of satisfying assignments, which is believed to be a strictly harder problem.

5. CUTTING PLANES

In the *cutting planes* proof system introduced in [Cook et al. 1987], clauses are interpreted as linear inequalities over the reals with integer coefficients, so that our example clause $x \vee y \vee \overline{z}$ gets translated to $x + y + (1 - z) \geq 1$, or $x + y - z \geq 0$ if we move all additive constants to the right-hand side (note that in contrast to polynomial calculus we now think of 1 as true and 0 as false, as we are more commonly used to). The derivation rules are

$$\textit{Variable axioms}\ \frac{}{0 \leq x \leq 1} \tag{11a}$$

$$\textit{Multiplication}\ \frac{\sum_i a_i x_i \geq A}{\sum_i c a_i x_i \geq cA} \tag{11b}$$

$$\textit{Addition}\ \frac{\sum_i a_i x_i \geq A \qquad \sum_i b_i x_i \geq B}{\sum_i (a_i + b_i) x_i \geq A + B} \tag{11c}$$

$$\textit{Division}\ \frac{\sum_i c a_i x_i \geq A}{\sum_i a_i x_i \geq \lceil A/c \rceil} \tag{11d}$$

where a_i, b_i, c, A, and B are all integers, and the goal is to show that a formula is unsatisfiable by deriving $0 \geq 1$ from the linear inequalities corresponding to the clauses of the formula. Once again it is clear that such a derivation can exist only if the formula is indeed unsatisifiable, and the other direction of this implication also holds. We want to highlight that in the division rule (11d) we can divide with the common factor c on the left and then *divide and round up* the constant term on the right to the closest integer, since we know that we are only interested in $0/1$ solutions. This division rule is where the power of cutting planes lies.

5.1. Cutting Planes Size, Length, and Space

The *length* of a cutting planes refutation is the total number of lines/inequalities in it, and the *size* also sums the sizes of all coefficients (i.e., the bit size of representing them). The natural generalization of clause space in resolution is to define cutting planes *(line) space* as the maximal number of linear inequalities needed in memory

during a refutation, since every clause is translated into a linear inequality. There is no useful analogue of width/degree known for cutting planes.

Cutting planes can simulate resolution efficiently with respect to length/size and space simultaneously by mimicking the resolution steps one by one, and hence just as was the case for polynomial calculus we get the same worst-case upper bounds.

Cutting planes is strictly stronger than resolution with respect to length and size, since it can refute PHP formulas (3a)–(3b) efficiently [Cook et al. 1987]. The reason for this is that in contrast to resolution (and polynomial calculus), cutting planes can count. PHP formulas are refuted simply but summing up the number of pigeons and holes, after which the observation can immediately be made that there are too many pigeons to fit into the holes. It seems probable that cutting planes and polynomial calculus are incomparable with respect to size, i.e., that for both proof systems one can find hard formulas that are easy for the other system. PHP formulas show that this is true in one direction, but the other direction is open.

When it comes to space, cutting planes is very much stronger than both resolution and polynomial calculus — [Galesi et al. 2015] recently showed that any CNF formula can be refuted in constant line space 5 by cutting planes![14] This proof works by starting with a linear inequality/hyperplane that cuts away the all-zero point of the Boolean hypercube $\{0,1\}^n$ from the candidate list of satisfying assignments (there has to exist a clause falsified by this assignment, from which the hyperplane can be obtained), and then uses 4 auxiliary hyperplanes to remove further points $\alpha \in \{0,1\}^n$ one by one in lexicographical order until all possible assignments have been eliminated, showing that the formulas is unsatisifiable. During the course of this refutation the size of the coefficients of the hyperplanes become exponentially large, however, which the line space measure does not charge for. If coefficient sizes are also counted, i.e., if one measures the *total space* of cutting planes refutations, then it is not hard to show a linear lower bound (for instance by combining [Ben-Sasson and Wigderson 2001] and [Beck et al. 2013]) and a quadratic worst-case upper bound is immediately implied by resolution. For resolution this quadratic upper bound is known to be tight by [Bonacina et al. 2014], but to the best of our knowledge no superlinear lower bounds are known on total space in cutting planes.

OPEN PROBLEM 7. *Are there superlinear total space lower bounds for cutting planes refutations with polynomial-size coefficients? Or with constant-size coefficients?*

Proving such lower bounds, if they exist, seems challenging, however. It might be worth noting in this context that already cutting planes with coefficients of absolute size 2 (which is the minimum needed to simulate resolution) is quite powerful — this is sufficient to construct space-efficient refutations of PHP formulas [Galesi et al. 2015].

Essentially the only formulas that are known to be hard for the cutting planes proof system with respect to length/size are the **clique-coclique formulas** claming (the negation of) that "a graph with an m-clique cannot be $(m-1)$-colourable." The formulas consist of clauses:

$$q_{k,1} \vee q_{k,2} \vee \cdots \vee q_{k,n} \qquad \text{[some vertex is the } k\text{th member of the clique]} \tag{12a}$$

$$\overline{q}_{k,i} \vee \overline{q}_{k',i} \qquad \text{[clique members are uniquely defined]} \tag{12b}$$

$$p_{i,j} \vee \overline{q}_{k,i} \vee \overline{q}_{k',j} \qquad \text{[clique members are neighbours]} \tag{12c}$$

$$r_{i,1} \vee r_{i,2} \vee \cdots \vee r_{i,m-1} \qquad \text{[every vertex } i \text{ has a colour]} \tag{12d}$$

$$\overline{p}_{i,j} \vee \overline{r}_{i,\ell} \vee \overline{r}_{j,\ell} \qquad \text{[neighbours have distinct colours]} \tag{12e}$$

[14] Recall that this means that the formula is kept on a read-only input tape, and the working memory never contain more than 5 inequalities at any given time.

where variables $p_{i,j}$ are indicators of the edges in an n-vertex graph, variables $q_{k,i}$ identify the members of an m-clique in the graph, and variables $r_{i,\ell}$ specify a colouring of the vertices, for indices ranging over $1 \leq i \neq j \leq n$, $i < j$, $1 \leq k \neq k' \leq m$, and $1 \leq \ell \leq m-1$.

[Pudlák 1997] proved that these formulas are hard by using a so-called *interpolation* argument, specifically tailored to work for formulas with the right structure. He showed that from any short cutting planes refutation of the formula, one can extract a small monotone circuit for clique, which reduces the problem to a question about proving size lower bounds for monotone circuits.

It seems plausible that the Tseitin formulas in Figure 2 should require long cutting planes refutations, since it should be hard to count $\mod 2$ using linear inequalities, and if this could be shown it would follow that cutting planes and polynomial calculus are incomparable with respect to proof size. It also seems very likely that random k-CNF formulas should be very hard, but no such lower bounds are known, nor any other lower bounds not using the interpolation method. These are all longstanding open problems in proof complexity.

OPEN PROBLEM 8. *Prove length lower bounds for cutting planes refutations of Tseitin formulas or random k-CNF formulas, or for any formula family by using some technique other than interpolation.*

5.2. Size-Space Trade-offs for Cutting Planes

Given our very limited understanding of cutting planes, it is perhaps not so surprising that not very much is known about size-space trade-offs for this proof system.

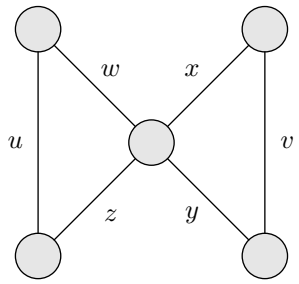
[Göös and Pitassi 2014] showed that short cutting planes refutations of Tseitin formulas on expanders must have large space, but this does not provide a real trade-off since it seems likely that such short refutations do not exist at all, regardless of their space complexity. [Huynh and Nordström 2012] proved that short cutting planes refutations of one particular version of pebbling contradictions (slightly different from the substituted pebbling contradictions discussed in Section 2.3) over one particular family of DAGs require large space — a result that was strengthened and generalized by [Göös and Pitassi 2014] — and for pebbling contradictions such short refutations do exist. Interestingly, and somewhat unexpectedly, all of these results follow from reductions to communication complexity. The state of knowledge regarding pebbling contradictions is much worse here than for resolution and polynomial calculus, however — for the latter two proof systems we know of general methods to translate pebbling trade-offs for (essentially) arbitrary graphs into proof complexity size-space trade-offs.

Since [Galesi et al. 2015] established that any CNF formula has a a constant-space refutation, the lower bounds for pebbling contradictions in [Huynh and Nordström 2012; Göös and Pitassi 2014] yield true size-space trade-off results for cutting planes, with formulas that can be refuted in both small size and small space, but where optimizing both measures simultaneously is impossible. However, the "space-efficient" refutations have coefficients of exponential size. It would be more convincing to obtain trade-offs where the small-space refutations also have small coefficients (which would follow if known resolution and polynomial calculus results for pebbling contradictions or Tseitin formulas over long, skinny grids could be lifted also to cutting planes).

OPEN PROBLEM 9. *Are there trade-offs where the space-efficient CP refutations have small coefficients (say, of polynomial or even constant size)?*

5.3. Pseudo-Boolean SAT Solving

Work on so-called *pseudo-Boolean solvers* exploring (a subset of) cutting planes has been done by, e.g., [Dixon and Ginsberg 2002; Dixon et al. 2004; Chai and Kuehlmann



(a) Eulerian graph.

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

(b) Corresponding CNF formula.

$$
\begin{aligned}
u + w &\geq 1\\
-u - w &\geq -1\\
u + z &\geq 1\\
-u - z &\geq -1\\
v + x &\geq 1\\
-v - x &\geq -1\\
v + y &\geq 1\\
-v - y &\geq -1\\
x + y + z + w &\geq 2\\
-x - y - z - w &\geq -2
\end{aligned}
$$

(c) Pseudo-Boolean encoding.

Fig. 6: Example of Markström's even colouring (EC) formula (satisfiable instance).

2005; Sheini and Sakallah 2006; Le Berre and Parrain 2010], but it seems like a very tough challenge to make these solvers as efficient as CDCL solvers. This is underlined by the fact that a competitive option for pseudo-Boolean solving is to simply translate the input to CNF and run a CDCL solver, as shown by [Eén and Sörensson 2006].

As it turns out, one problem with current pseudo-Boolean solvers is that if they get their input in CNF, they cannot even go beyond resolution. Solvers such as *Sat4j* [Le Berre and Parrain 2010] solve PHP formulas very efficiently, but they crucially depend on the input being given as linear inequalities:

$$p_{i,1} + p_{i,2} + \cdots + p_{i,n} \geq 1 \qquad \text{[every pigeon } i \text{ gets a hole]} \tag{13a}$$

$$-p_{1,j} - p_{2,j} - \cdots - p_{n+1,j} \geq -1 \qquad \text{[no hole } j \text{ gets two pigeons]} \tag{13b}$$

If the input is instead presented in CNF, with the cardinality constraints in Equation (13b) encoded as the clauses in Equation (3b), then Sat4j runs in exponential time. The same holds for subset cardinality formulas — if a pseudo-Boolean solver is fed the formula encoded as cardinality constraints, then it runs fast, but on the CNF version in Figure 3b it cannot possibly do better than the exponential lower bound on resolution length in [Mikša and Nordström 2014].

Thus, an algorithmic challenge is to make pseudo-Boolean solvers reason more efficiently with CNF inputs, so that they could, e.g., detect and use the cardinality constraints hidden in (3a)–(3b) to get performance comparable to when the input is given as (13a)–(13b). It is possible to do a preprocessing step to recover cardinality constraints encoded in CNF, and for PHP formulas and subset cardinality formulas this works well [Biere et al. 2014], but full preprocessing of the input to try to detect cardinality constraints is probably not an efficient approach in general.

This is not the only challenge for pseudo-Boolean solvers, however. Another quite intriguing family of benchmark formulas in this context are the *even colouring (EC) formulas* constructed by [Markström 2006] and shown in Figure 6. Here one starts with a connected graph G having an Eulerian cycle, i.e., with all vertex degrees even, and writes down constraints that edges should be labelled $0/1$ in such a way that for every vertex v in G the number of 0-edges and 1-edges incident to v is equal. If the total number of edges in the graph is even, then this formula is satisfiable — just fix any Eulerian cycle and label every second edge 0 and 1, respectively. If the number of edges is odd, however, then cutting planes can sum the at-least-2 constraints in Figure 6c (the

ones with positive coefficients) over all vertices to derive $2 \cdot \sum_{e \in E(G)} e \geq |E(G)|$ and then divide by 2 and round up to obtain $\sum_{e \in E(G)} e \geq (|E(G)| + 1)/2$. By instead summing up all at-most-2 constraints (the ones with negative coefficients) and dividing by 2 one obtains $\sum_{e \in E(G)} e \leq (|E(G)| - 1)/2$, and subtracting these two inequalities yields $0 \geq 1$.

One interesting aspect to observe here is that in contrast to PHP and subset cardinality formulas, the above argument uses crucially that variables are integer-valued. To see the difference, suppose that we are given a PHP or subset cardinality formula encoded as linear constraints. Then for cutting planes it is sufficient to simply add up the inequalities to derive a contradiction. No integer-based reasoning is needed. Even if we allow putting fractional pigeons into fractional holes, there is no way one can make a pigeon mass of $n + 1$ fit into holes of total capacity n. This set of linear inequalities is unsatisfiable even over the rationals, i.e., the polytopes defined by the constraints is empty. Similarly, for subset cardinality formulas there is no way $4n + 1$ variables could have a total "true mass" of at least $2n + 1$ and a total "false mass" of $2n + 1$ simultaneously. But for collections of linear constraints as in Figure 6c, assigning all edges value $\frac{1}{2}$ is a satisfying fractional solution. The polytope defined by the linear inequalities is *not* empty, but it does not contain any integer points. Hence, refuting EC formulas in cutting planes crucially requires the division rule (11d), and pseudo-Boolean solvers need to implement this rule (or some other form of integer-based reasoning) to decide these formulas efficiently. Based on some limited experiments, however, EC formulas appear to be much harder for pseudo-Boolean solvers than the cutting planes upper bound would suggest, which seem to indicate that the solvers are still quite far from using the full power of cutting planes reasoning.

6. EXTENDED RESOLUTION

A topic of considerable interest in the applied SAT community lately has been if and how SAT solvers can be enhanced to use *extended resolution*, an approach that was proposed in, e.g., [Sinz and Biere 2006; Audemard et al. 2010]. What this means is that one starts with a CDCL solver searching for resolution proofs as usual, but adds the option of introducing new variables as names of subformulas. What can proof complexity say about such an approach to SAT solving?

Not too much — if there are no restrictions on how these new variables can be added, then this corresponds to the proof system *extended Frege*, which is an extremely strong proof system for which essentially no lower bounds are known. So this means that the potential gains in performance are enourmous, but of course the question is how the new variables should be added to realize this potential.

In order to make it possible to study the kind of extended resolution used in CDCL solvers, one needs to describe in more detail the rules actually used for adding new variables. Once one has such a description, such as, e.g., for the *bounded variable addition* used in [Manthey et al. 2013], it is possible to reason about what such a limited form of extended resolution could or could not do (although proving formal theorems about this might still be a formidable problem).

7. CONCLUDING REMARKS

In this column, we have presented an overview of the proof systems resolution, polynomial calculus and cutting planes, motivated by the fact that these systems serve as foundations for SAT solvers using conflict-driven clause learning (CDCL), algebraic Gröbner basis computations, and pseudo-Boolean techniques, respectively. The discussion has intentionally been kept at a high level, with only informal statements of results. For many of the proof complexity results mentioned in this paper it is possible to find exact, formal statements in the survey paper [Nordström 2013].

On the proof complexity side, the main take-away message is that resolution is fairly well understood, although there are still some interesting open questions left (which we mostly did not discuss). For polynomial calculus we also have a fair amount of knowledge, although there are many more open problems than for resolution. For instance, the techniques for proving degree lower bounds (and hence size lower bounds) are not yet very well developed, and the hardness status of several interesting formula families remain open. Also, we do not understand very well the relations between degree and monomial space. For cutting planes, almost nothing is known, and any progress on the open problems listed in this survey would be very exciting.

When it comes to applied SAT solving, we still have quite a poor understanding of why different formulas are easy or hard. It would be interesting to investigate further whether there could be any relevant connections here between proof complexity measures and hardness of SAT, or whether tools and techniques from proof complexity could help to shed light on the inner workings on SAT solvers.

Finally, the main algorithmic challenge we want to highlight is if and how one can build efficient SAT solvers based on stronger proof systems than resolution. Is it really the case that CDCL, originating in the DPLL method from the early 1960s [Davis and Putnam 1960; Davis et al. 1962; Robinson 1965], is the best conceivable paradigm? Or could it be possible that is it now time, over 50 years later, to take the next step and build fundamentally different SAT solvers based on algebraic and/or geometric methods? Are there perhaps fundamental limitations why efficient proof search cannot be implemented within these proof systems? Or could it be that a sustained long-term effort would yield powerful new SAT solving paradigms, just as the immense work spent on optimizing CDCL solvers over the years have led to improvements in performance of several orders of magnitude?

ACKNOWLEDGMENTS

Many colleagues in the proof complexity community have shaped my thinking on this topic, and I am probably more indebted to them than I myself realize. In addition, in connection with this survey I want to acknowledge all the SAT researchers who have kindly listened to my never-ending stream of questions over the last few years and have patiently answered them (sometimes repeatedly). I am especially thankful to Gilles Audemard, Armin Biere, Niklas Éen, Marijn Heule, Matti Järvisalo, Daniel Le Berre, João Marques-Silva, Karem Sakallah, Laurent Simon, and Niklas Sörensson for many helpful discussions.

I wish to thank Neil Immerman for asking me to write this survey and gently nudging me to follow through. I am very grateful to him as well as to Christoph Berkholz, Armin Biere, Ilario Bonacina, Marijn Heule, Matti Järvisalo, and Massimo Lauria for helping to proof-read the manuscript, suggesting many improvements and discovering many inaccuracies in the process. It goes without saying that any opinons expressed, and any errors remaining, are the responsibility of the author only.

The author was funded by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007–2013) / ERC grant agreement no. 279611 as well as by Swedish Research Council grants 621-2010-4797 and 621-2012-5645.

REFERENCES

Michael Alekhnovich. 2004. Mutilated Chessboard Problem Is Exponentially Hard for Resolution. *Theoretical Computer Science* 310, 1–3 (Jan. 2004), 513–525.

Michael Alekhnovich, Eli Ben-Sasson, Alexander A. Razborov, and Avi Wigderson. 2002. Space Complexity in Propositional Calculus. *SIAM J. Comput.* 31, 4 (2002), 1184–1211. Preliminary version in *STOC '00*.

Michael Alekhnovich and Alexander A. Razborov. 2003. Lower Bounds for Polynomial Calculus: Non-Binomial Case. *Proceedings of the Steklov Institute of Mathematics* 242 (2003), 18–35. Available at http://people.cs.uchicago.edu/~razborov/files/misha.pdf. Preliminary version in *FOCS '01*.

Michael Alekhnovich and Alexander A. Razborov. 2008. Resolution Is Not Automatizable Unless W[P] Is Tractable. *SIAM J. Comput.* 38, 4 (Oct. 2008), 1347–1363. Preliminary version in *FOCS '01*.

Carlos Ansótegui, María Luisa Bonet, Jordi Levy, and Felip Manyà. 2008. Measuring the Hardness of SAT Instances. In *Proceedings of the 23rd National Conference on Artificial Intelligence (AAAI '08)*. 222–228.

Albert Atserias and Víctor Dalmau. 2008. A Combinatorial Characterization of Resolution Width. *J. Comput. System Sci.* 74, 3 (May 2008), 323–334. Preliminary version in *CCC '03*.

Albert Atserias, Johannes Klaus Fichte, and Marc Thurley. 2011. Clause-Learning Algorithms with Many Restarts and Bounded-Width Resolution. *Journal of Artificial Intelligence Research* 40 (Jan. 2011), 353–373. Preliminary version in *SAT '09*.

Albert Atserias, Massimo Lauria, and Jakob Nordström. 2014. Narrow Proofs May Be Maximally Long. In *Proceedings of the 29th Annual IEEE Conference on Computational Complexity (CCC '14)*. 286–297.

Gilles Audemard, George Katsirelos, and Laurent Simon. 2010. A Restriction of Extended Resolution for Clause Learning SAT Solvers. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI '10)*. 15–20.

Gilles Audemard and Laurent Simon. 2009. Predicting Learnt Clauses Quality in Modern SAT Solvers. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI '09)*. 399–404.

Roberto J. Bayardo Jr. and Robert Schrag. 1997. Using CSP Look-Back Techniques to Solve Real-World SAT Instances. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI '97)*. 203–208.

Paul Beame, Chris Beck, and Russell Impagliazzo. 2012. 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)*. 213–232.

Paul Beame, Joseph C. Culberson, David G. Mitchell, and Cristopher Moore. 2005. The Resolution Complexity of Random Graph k-Colorability. *Discrete Applied Mathematics* 153, 1-3 (Dec. 2005), 25–47.

Paul Beame, Russell Impagliazzo, and Ashish Sabharwal. 2007. The Resolution Complexity of Independent Sets and Vertex Covers in Random Graphs. *Computational Complexity* 16, 3 (Oct. 2007), 245–297.

Paul Beame, Henry Kautz, and Ashish Sabharwal. 2004. Towards Understanding and Harnessing the Potential of Clause Learning. *Journal of Artificial Intelligence Research* 22 (Dec. 2004), 319–351. Preliminary version in *IJCAI '03*.

Chris Beck, Jakob Nordström, and Bangsheng Tang. 2013. Some Trade-off Results for Polynomial Calculus. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC '13)*. 813–822.

Eli Ben-Sasson. 2009. Size-Space Tradeoffs for Resolution. *SIAM J. Comput.* 38, 6 (May 2009), 2511–2525. Preliminary version in *STOC '02*.

Eli Ben-Sasson and Nicola Galesi. 2003. Space Complexity of Random Formulae in Resolution. *Random Structures and Algorithms* 23, 1 (Aug. 2003), 92–109. Preliminary version in *CCC '01*.

Eli Ben-Sasson and Russell Impagliazzo. 1999. Random CNF's are Hard for the Polynomial Calculus. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science (FOCS '99)*. 415–421. Journal version in [Ben-Sasson and Impagliazzo 2010].

Eli Ben-Sasson and Russell Impagliazzo. 2010. Random CNF's are Hard for the Polynomial Calculus. *Computational Complexity* 19 (2010), 501–519. Issue 4. Preliminary version in *FOCS '99*.

Eli Ben-Sasson, Russell Impagliazzo, and Avi Wigderson. 2004. Near Optimal Separation of Tree-Like and General Resolution. *Combinatorica* 24, 4 (Sept. 2004), 585–603.

Eli Ben-Sasson and Jakob Nordström. 2008. 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)*. 709–718.

Eli Ben-Sasson and Jakob Nordström. 2011. Understanding Space in Proof Complexity: Separations and Trade-offs via Substitutions. In *Proceedings of the 2nd Symposium on Innovations in Computer Science (ICS '11)*. 401–416.

Eli Ben-Sasson and Avi Wigderson. 2001. Short Proofs are Narrow—Resolution Made Simple. *J. ACM* 48, 2 (March 2001), 149–169. Preliminary version in *STOC '99*.

Patrick Bennett, Ilario Bonacina, Nicola Galesi, Tony Huynh, Mike Molloy, and Paul Wollan. 2015. *Space Proof Complexity for Random 3-CNFs*. Technical Report 1503.01613. arXiv.org.

Christoph Berkholz. 2012. On the Complexity of Finding Narrow Proofs. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS '12)*. 351–360.

Armin Biere. 2010. *Lingeling, Plingeling, PicoSAT and PrecoSAT at SAT Race 2010*. Technical Report 10/1. FMV Reports Series, Institute for Formal Models and Verification, Johannes Kepler University.

Armin Biere, Marijn J. H. Heule, Hans van Maaren, and Toby Walsh (Eds.). 2009. *Handbook of Satisfiability*. Frontiers in Artificial Intelligence and Applications, Vol. 185. IOS Press.

Armin Biere, Daniel Le Berre, Emmanuel Lonca, and Norbert Manthey. 2014. Detecting cardinality constraints in CNF. In *Proceedings of the 17th International Conference on Theory and Applications of Satisfiability Testing (SAT '14) (Lecture Notes in Computer Science)*, Vol. 8561. Springer, 285–301.

Archie Blake. 1937. *Canonical Expressions in Boolean Algebra*. Ph.D. Dissertation. University of Chicago.

Ilario Bonacina and Nicola Galesi. 2015. A Framework for Space Complexity in Algebraic Proof Systems. *J. ACM* 62, 3, Article 23 (June 2015), 23:1–23:20 pages. Preliminary version in *ITCS '13*.

Ilario Bonacina, Nicola Galesi, and Neil Thapen. 2014. Total Space in Resolution. In *Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS '14)*. 641–650.

María Luisa Bonet, Juan Luis Esteban, Nicola Galesi, and Jan Johannsen. 2000. On the Relative Complexity of Resolution Refinements and Cutting Planes Proof Systems. *SIAM J. Comput.* 30, 5 (2000), 1462–1484. Preliminary version in *FOCS '98*.

María Luisa Bonet and Nicola Galesi. 2001. Optimality of Size-Width Tradeoffs for Resolution. *Computational Complexity* 10, 4 (Dec. 2001), 261–276. Preliminary version in *FOCS '99*.

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

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

Samuel R. Buss, Dima Grigoriev, Russell Impagliazzo, and Toniann Pitassi. 2001. Linear Gaps between Degrees for the Polynomial Calculus Modulo Distinct Primes. *J. Comput. System Sci.* 62, 2 (March 2001), 267–289. Preliminary version in *CCC '99*.

Samuel R. Buss, Jan Hoffmann, and Jan Johannsen. 2008. Resolution Trees with Lemmas: Resolution Refinements that Characterize DLL-Algorithms with Clause Learning. *Logical Methods in Computer Science* 4, 4:13 (Dec. 2008).

Donald Chai and Andreas Kuehlmann. 2005. A Fast Pseudo-Boolean Constraint Solver. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 24, 3 (March 2005), 305–317. Preliminary version in *DAC '03*.

Vašek Chvátal and Endre Szemerédi. 1988. Many Hard Examples for Resolution. *J. ACM* 35, 4 (Oct. 1988), 759–768.

Matthew Clegg, Jeffery Edmonds, and Russell Impagliazzo. 1996. Using the Groebner Basis Algorithm to Find Proofs of Unsatisfiability. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC '96)*. 174–183.

Stephen A. Cook. 1971. The Complexity of Theorem-Proving Procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing (STOC '71)*. 151–158.

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

William Cook, Collette Rene Coullard, and György Turán. 1987. On the Complexity of Cutting-Plane Proofs. *Discrete Applied Mathematics* 18, 1 (Nov. 1987), 25–38.

Martin Davis, George Logemann, and Donald Loveland. 1962. A Machine Program for Theorem Proving. *Commun. ACM* 5, 7 (July 1962), 394–397.

Martin Davis and Hilary Putnam. 1960. A Computing Procedure for Quantification Theory. *J. ACM* 7, 3 (1960), 201–215.

Heidi E. Dixon and Matthew L. Ginsberg. 2002. Inference Methods for a Pseudo-Boolean Satisfiability Solver. In *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI '02)*. 635–640.

Heidi E. Dixon, Matthew L. Ginsberg, and Andrew J. Parkes. 2004. Generalizing Boolean Satisfiability I: Background and Survey of Existing Work. *Journal of Artificial Intelligence Research* 21 (2004), 193–243.

Niklas Eén and Niklas Sörensson. 2004. An Extensible SAT-solver. In *6th International Conference on Theory and Applications of Satisfiability Testing (SAT '03), Selected Revised Papers (Lecture Notes in Computer Science)*, Vol. 2919. Springer, 502–518.

Niklas Eén and Niklas Sörensson. 2006. Translating Pseudo-Boolean Constraints into SAT. *Journal on Satisfiability, Boolean Modeling and Computation* 2, 1-4 (2006), 1–26.

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

Yuval Filmus, Massimo Lauria, Mladen Mikša, Jakob Nordström, and Marc Vinyals. 2013. Towards an Understanding of Polynomial Calculus: New Separations and Lower Bounds (Extended Abstract). In *Proceedings of the 40th International Colloquium on Automata, Languages and Programming (ICALP '13) (Lecture Notes in Computer Science)*, Vol. 7965. Springer, 437–448.

Yuval Filmus, Massimo Lauria, Mladen Mikša, Jakob Nordström, and Marc Vinyals. 2015a. From Small Space to Small Width in Resolution. *ACM Transactions on Computational Logic* 16, 4, Article 28 (July 2015), 28:1–28:15 pages. Preliminary version in *STACS '14*.

Yuval Filmus, Massimo Lauria, Jakob Nordström, Noga Ron-Zewi, and Neil Thapen. 2015b. Space Complexity in Polynomial Calculus. *SIAM J. Comput.* 44, 4 (Aug. 2015), 1119–1153. Preliminary version in *CCC '12*.

Nicola Galesi and Massimo Lauria. 2010. Optimality of Size-Degree Trade-offs for Polynomial Calculus. *ACM Transactions on Computational Logic* 12, Article 4 (Nov. 2010), 4:1–4:22 pages. Issue 1.

Nicola Galesi, Pavel Pudlák, and Neil Thapen. 2015. The Space Complexity of Cutting Planes Refutations. In *Proceedings of the 30th Annual Computational Complexity Conference (CCC '15) (Leibniz International Proceedings in Informatics (LIPIcs))*, Vol. 33. 433–447.

John R. Gilbert and Robert Endre Tarjan. 1978. *Variations of a Pebble Game on Graphs*. Technical Report STAN-CS-78-661. Stanford University. Available at http://infolab.stanford.edu/TR/CS-TR-78-661.html.

Mika Göös and Toniann Pitassi. 2014. Communication Lower Bounds via Critical Block Sensitivity. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC '14)*. 847–856.

Armin Haken. 1985. The Intractability of Resolution. *Theoretical Computer Science* 39, 2-3 (Aug. 1985), 297–308.

Philipp Hertel, Fahiem Bacchus, Toniann Pitassi, and Allen Van Gelder. 2008. Clause Learning Can Effectively P-Simulate General Propositional Resolution. In *Proceedings of the 23rd National Conference on Artificial Intelligence (AAAI '08)*. 283–290.

Marijn Heule and Hans van Maaren. 2005. Aligning CNF- and Equivalence-Reasoning. In *7th International Conference on Theory and Applications of Satisfiability Testing (SAT '04), Selected Revised Papers (Lecture Notes in Computer Science)*, Vol. 3542. Springer, 145–156.

Trinh Huynh and Jakob Nordström. 2012. On the Virtue of Succinct Proofs: Amplifying Communication Complexity Hardness to Time-Space Trade-offs in Proof Complexity (Extended Abstract). In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC '12)*. 233–248.

Russell Impagliazzo, Pavel Pudlák, and Jiří Sgall. 1999. Lower Bounds for the Polynomial Calculus and the Gröbner Basis Algorithm. *Computational Complexity* 8, 2 (1999), 127–144.

Matti Järvisalo, Marijn Heule, and Armin Biere. 2012a. Inprocessing Rules. In *Proceedings of the 6th International Joint Conference on Automated Reasoning (IJCAR '12) (Lecture Notes in Computer Science)*, Vol. 7364. Springer, 355–370.

Matti Järvisalo, Arie Matsliah, Jakob Nordström, and Stanislav Živný. 2012b. Relating Proof Complexity Measures and Practical Hardness of SAT. In *Proceedings of the 18th International Conference on Principles and Practice of Constraint Programming (CP '12) (Lecture Notes in Computer Science)*, Vol. 7514. Springer, 316–331.

Balakrishnan Krishnamurthy. 1985. Short proofs for tricky formulas. *Acta Informatica* 22, 3 (Aug. 1985), 253–275.

Daniel Le Berre and Anne Parrain. 2010. The Sat4j Library, Release 2.2. *Journal on Satisfiability, Boolean Modeling and Computation* 7 (2010), 59–64.

Thomas Lengauer and Robert Endre Tarjan. 1982. Asymptotically Tight Bounds on Time-Space Trade-offs in a Pebble Game. *J. ACM* 29, 4 (Oct. 1982), 1087–1130. Preliminary version in *STOC '79*.

Norbert Manthey, Marijn Heule, and Armin Biere. 2013. Automated Reencoding of Boolean Formulas. In *8th International Haifa Verification Conference (HVC '12), Revised Selected Papers (Lecture Notes in Computer Science)*, Vol. 7857. Springer, 102–117.

Klas Markström. 2006. Locality and Hard SAT-Instances. *Journal on Satisfiability, Boolean Modeling and Computation* 2, 1-4 (2006), 221–227.

João P. Marques-Silva and Karem A. Sakallah. 1999. GRASP: A Search Algorithm for Propositional Satisfiability. *IEEE Trans. Comput.* 48, 5 (May 1999), 506–521. Preliminary version in *ICCAD '96*.

Mladen Mikša and Jakob Nordström. 2014. Long Proofs of (Seemingly) Simple Formulas. In *Proceedings of the 17th International Conference on Theory and Applications of Satisfiability Testing (SAT '14) (Lecture Notes in Computer Science)*, Vol. 8561. Springer, 121–137.

Mladen Mikša and Jakob Nordström. 2015. A Generalized Method for Proving Polynomial Calculus Degree Lower Bounds. In *Proceedings of the 30th Annual Computational Complexity Conference (CCC '15) (Leibniz International Proceedings in Informatics (LIPIcs))*, Vol. 33. 467–487.

Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. 2001. Chaff: Engineering an Efficient SAT Solver. In *Proceedings of the 38th Design Automation Conference (DAC '01)*. 530–535.

Jakob Nordström. 2009. Narrow Proofs May Be Spacious: Separating Space and Width in Resolution. *SIAM J. Comput.* 39, 1 (May 2009), 59–121. Preliminary version in *STOC '06*.

Jakob Nordström. 2013. Pebble Games, Proof Complexity and Time-Space Trade-offs. *Logical Methods in Computer Science* 9, Article 15 (Sept. 2013), 15:1–15:63 pages. Issue 3.

Jakob Nordström. 2014. A (Biased) Proof Complexity Survey for SAT Practitioners. In *Proceedings of the 17th International Conference on Theory and Applications of Satisfiability Testing (SAT '14) (Lecture Notes in Computer Science)*, Vol. 8561. Springer, 1–6.

Jakob Nordström and Johan Håstad. 2013. Towards an Optimal Separation of Space and Length in Resolution. *Theory of Computing* 9, Article 14 (May 2013), 471–557 pages. Preliminary version in *STOC '08*.

Wolfgang J. Paul, Robert Endre Tarjan, and James R. Celoni. 1977. Space Bounds for a Game on Graphs. *Mathematical Systems Theory* 10 (1977), 239–251.

Knot Pipatsrisawat and Adnan Darwiche. 2011. On the Power of Clause-Learning SAT Solvers as Resolution Engines. *Artificial Intelligence* 175 (Feb. 2011), 512–525. Issue 2. Preliminary version in *CP '09*.

Pavel Pudlák. 1997. Lower Bounds for Resolution and Cutting Plane Proofs and Monotone Computations. *Journal of Symbolic Logic* 62, 3 (Sept. 1997), 981–998.

Ran Raz. 2004. Resolution Lower Bounds for the Weak Pigeonhole Principle. *J. ACM* 51, 2 (March 2004), 115–138. Preliminary version in *STOC '02*.

Ran Raz and Pierre McKenzie. 1999. Separation of the Monotone NC Hierarchy. *Combinatorica* 19, 3 (March 1999), 403–435. Preliminary version in *FOCS '97*.

Alexander A. Razborov. 1998. Lower Bounds for the Polynomial Calculus. *Computational Complexity* 7, 4 (Dec. 1998), 291–324.

Alexander A. Razborov. 2002. Proof Complexity of Pigeonhole Principles. In *5th International Conference on Developments in Language Theory, (DLT '01), Revised Papers (Lecture Notes in Computer Science)*, Vol. 2295. Springer, 100–116.

Alexander A. Razborov. 2003. Resolution Lower Bounds for the Weak Functional Pigeonhole Principle. *Theoretical Computer Science* 1, 303 (June 2003), 233–243.

Alexander A. Razborov. 2004. Resolution Lower Bounds for Perfect Matching Principles. *J. Comput. System Sci.* 69, 1 (Aug. 2004), 3–27. Preliminary version in *CCC '02*.

Alexander A. Razborov. 2015. *An Ultimate Trade-Off in Propositional Proof Complexity*. Technical Report TR15-033. Electronic Colloquium on Computational Complexity (ECCC).

Søren Riis. 1993. *Independence in Bounded Arithmetic*. Ph.D. Dissertation. University of Oxford.

John Alan Robinson. 1965. A machine-oriented logic based on the resolution principle. *J. ACM* 12, 1 (Jan. 1965), 23–41.

Nathan Segerlind. 2007. The Complexity of Propositional Proofs. *Bulletin of Symbolic Logic* 13, 4 (Dec. 2007), 417–481.

Hossein M. Sheini and Karem A. Sakallah. 2006. Pueblo: A Hybrid Pseudo-Boolean SAT Solver. *Journal on Satisfiability, Boolean Modeling and Computation* 2, 1-4 (March 2006), 165–189. Preliminary version in *DATE '05*.

Carsten Sinz and Armin Biere. 2006. Extended Resolution Proofs for Conjoining BDDs. In *Proceedings of the 1st International Computer Science Symposium in Russia (CSR '06) (Lecture Notes in Computer Science)*, Vol. 3967. Springer, 600–611.

Ivor Spence. 2010. sgen1: A Generator of Small but Difficult Satisfiability Benchmarks. *Journal of Experimental Algorithmics* 15, Article 1.2 (March 2010), 1.2:1–1.2:15 pages.

Gunnar Stålmarck. 1996. Short Resolution Proofs for a Sequence of Tricky Formulas. *Acta Informatica* 33, 3 (May 1996), 277–280.

Neil Thapen. 2014. *A Trade-off Between Length and Width in Resolution*. Technical Report TR14-137. Electronic Colloquium on Computational Complexity (ECCC).

Alasdair Urquhart. 1987. Hard Examples for Resolution. *J. ACM* 34, 1 (Jan. 1987), 209–219.

Allen Van Gelder and Ivor Spence. 2010. Zero-One Designs Produce Small Hard SAT Instances. In *Proceedings of the 13th International Conference on Theory and Applications of Satisfiability Testing (SAT '10) (Lecture Notes in Computer Science)*, Vol. 6175. Springer, 388–397.