Towards $NP-P$ via Proof Complexity and Search

Samuel R. Buss[1]

Department of Mathematics, University of California, San Diego, La Jolla, CA 92093-0112, USA

Abstract

This is a survey of work on proof complexity and proof search from a logico-algorithmic viewpoint, as motivated by the P versus NP problem. We discuss propositional proof complexity, Cook's program, proof automatizability, proof search, algorithms for satisfiability, and the state of the art of our (in)ability to separate P and NP.

1. Introduction

One of the principal open problems in computer science, and indeed in all of mathematics, is the P versus NP problem. This is really only one problem out of a host of related problems, including questions such as whether polynomial time equals polynomial space, whether particular concrete Boolean problems require superlinear or even exponential size circuits, whether pseudo-random number generators exist, etc. It is traditional to focus on the P versus NP problem as being the central open problem, partly because of the naturalness of the problem, and partly for historical reasons.

Many of the methods that have been used to attack the P versus NP problem have been combinatorial, algebraic, or probabilistic; this includes, for instance switching lemmas, or polynomial approximations, and a host of results about randomization. In this paper, we survey instead various "logico-algorithmic" attempts to solve the P versus NP problem. This is motivated by the analogy with the Gödel incompleteness theorems, and by the hope that perhaps some kind of extension of self-reference arguments can separate NP from P.

As far I know, the first person to express in print the idea that something akin to $P \neq NP$ might follow from a self-reference argument was G. Kreisel [71]. Kreisel wrote:

> "Suppose we ... have a proof system; ... the 'faith' is that in a natural way this will yield a feasible proof procedure for feasible theorems.

Email address: `sbuss@math.ucsd.edu` (Samuel R. Buss)

[1]Supported in part by NSF grants DMS-0700533 and DMS-1101228.

"Conjecture: Under reasonable conditions on feasibility, there is an analogue to Gödel's second incompleteness theorem, that is the article of faith above is unjustified."

Surprisingly enough, this comment by Kreisel, and the earlier comment below by Gödel, were made prior to the first formal definitions of the class *NP* by Cook [34] and Levin [74]. Kreisel did not expand any further on his remarks above, and Gödel's letter was not circulated. Thus, neither of them influenced the development of computational complexity. For more comprehensive treatments of the prehistory of the P versus NP problem, see the articles by Cook [36] and Sipser [109].

What reasons do we have for believing $P \neq NP$? One is that $P = NP$ could make the practice of mathematics too easy. Mathematical research could be automated by formalizing mathematical questions completely, and then blindly searching for proofs of conjectured mathematical statements. If $P = NP$, this process could succeed whenever proofs are not too large.[2] This would be obviously be a major change in the practice of mathematics!

To the best of my knowledge, the observation about the profound effect that $P = NP$ would have on mathematics was first made by Gödel, who wrote in a 1956 letter to von Neumann that if proof search could be carried out feasibly, it would have "consequences of the greatest importance. ... The mental work of a mathematician concerning Yes-No questions could be completely replaced by a machine." (The translation is by Clote, see [33].)

In spite of the fact that most researchers believe $P \neq NP$, the evidence that $P \neq NP$ is tenuous. Indeed, it seems be based on three factors: firstly, conservativism about the possibility of fundamentally new algorithms; secondly, the analogy with the incompleteness theorems; and thirdly, the fact that no one has been able to prove that $P = NP$.

Accordingly, the title of the paper is a bit of a pun, meant to illustrate our doubts about our belief that $P \neq NP$: The titles uses "*NP–P*": this can also be interpreted as subtracting P from NP. If they are equal, then we are heading towards the empty set!

As a caveat to the reader, the field of propositional proof complexity is quite large, and we have not attempted to be comprehensive in the citations to papers, as it would cause a substantial expansion in the length of this paper and its bibliography. At times we cite original results; at other times we cite only later, more definitive, results of a given area. We have tried to cite the papers that give the best entry to an area, but the reader who wishes to learn more about particular areas should be sure to look further into the literature. Other expository articles on propositional proof complexity include [11, 14, 23, 24, 25, 91, 106].

[2] As a side remark: One needs more than just $P = NP$, rather one needs that natural *NP* problems such as satisfiability have feasibly effective algorithms.

2. Propositional proof complexity

Gödel, in the above-quoted letter, was interested in the computational complexity of searching for formal proofs. Since that time, proof complexity and proof search have become increasingly important.

Suppose we are given a particular proof system P. The kinds of questions we might like to answer include the following.

- Given a formula φ, decide if it has a short P-proof.
- Given a formula φ, and a promise that it has a short P-proof, find a P-proof of φ.
- Characterize the formulas that have reasonable length (polynomial length) P-proofs.
- Compare the strength of P with other proof systems.

Surprisingly, these questions are difficult even for propositional proof systems. Indeed, as is discussed below, under some special conditions, the existence of short propositional proofs is an *NP*-complete problem, and hence a feasible method to find optimal propositional proofs will also give a feasible algorithm to find (short) proofs in *any* proof system.

There are a variety of propositional proof systems that are commonly studied. Three of the most basic systems are

- Resolution — a proof system for proving disjunctive normal form (DNF) formulas.
- Frege proofs — "textbook"-style system using modus ponens as its only rule of inference.
- Extended Frege systems — Frege systems augmented with the ability to introduce new variables that abbreviate formulas.

For these, and other proof systems, it is important to have a good notion of the length or size of a proof. The most useful notion is generally as follows.

Definition 1. The *length* of a proof is the number of symbols used in the proof.

The reader is likely to be familiar with resolution and with Frege systems. Extended Frege systems are less well-known: an alternate characterization of extended Frege systems is to define an extended Frege proof to be the same as a Frege proof, but with its length equal to the number of steps (or, formulas) in the proof rather than the number of symbols in the proof.

There are many other possible proof systems. An important generalization of the notion of a proof system is the abstract proof systems that were defined by Cook and Reckhow [37, 38].

Definition 2. An *abstract proof system* is a polynomial time function f whose range is equal to the set of tautologies. If φ is a tautology, then an f-proof of φ is any value w such that $f(w) = \varphi$.

Any standard proof system P can be viewed as an abstract proof system by defining $f(w)$ so that if w is an encoding of a valid P-proof, then $f(w)$ is the last line of the proof. For w's that do not code valid P-proofs, $f(w)$ can equal an arbitrary, fixed tautology. Assuming that P-proofs can be efficiently encoded, and can be recognized in polynomial time, the function f is an abstract proof system that faithfully captures the notion of P-proofs.

Abstract proof systems can be quite strong; indeed, they capture any proof system with polynomial recognizable proofs, even somewhat nonstandard proof systems. For example, set theory can be viewed as a propositional proof system by letting any proof in *ZFC* of a formula expressing "φ is a tautology" serve as a proof of φ.

Definition 3. A proof system is *super* if every tautology φ has a proof which has length polynomially bounded by the length of φ.

Theorem 4. (Cook-Reckhow [37]) *There exists a super proof system if and only if NP is closed under complementation; that is, if and only if NP = coNP.*

It is, of course, open whether any super proof system exists. Similarly, it is open whether there is any optimal proof system, that is to say, whether there is a "strongest" proof system that simulates any other given proof system with polynomial size proofs. Of course, if there is no optimal proof system, then *NP* $\neq$ *coNP*.[3]

The above theorem has led to what is sometimes called "Cook's program" for proving *NP* $\neq$ *coNP*.

Cook's Program for separating *NP* and *coNP*: Prove superpolynomial lower bounds for proof lengths in stronger and stronger propositional proof systems, until they are established for all abstract proof systems.

Cook's program is an attractive and plausible approach; unfortunately, it has turned out to be quite hard to establish superpolynomial lower bounds on common proof systems. Indeed, at the time that Cook and Reckhow introduced abstract proof systems, lower bounds were not known even for resolution. Subsequently, there has been a good deal of results, but the current state of the art is far away from being to establish superpolynomial lower bounds for common systems such as Frege systems.

Nonetheless, a number of propositional proof systems are known to require nearly exponential-size proofs. These include the following systems.

- Method of truth-tables.

[3]For more on optimal proof systems, see [69, 101, 61, 20, 31, 50].

- Tree-like resolution and regular resolution; Tseitin [111].
- Resolution; Haken [47], Urquhart [112], Chvátal-Szemerédi [32], Raz [92], Razborov [98, 99], and many others.
- Bounded depth Frege systems; Ajtai [1], Krajíček [62], Pitassi-Beame-Impaliazzo [86], Krajíček-Pudlák-Woods [70].
- Cutting planes systems; Pudlák [90].
- Nullstellensatz systems; Buss-Impagliazzo-Krajíček-Pudlák-Razborov-Sgall [28], Grigoriev [46].
- The polynomial calculus; Razborov [96], Impagliazzo-Pudlák-Sgall [57], Ben-Sasson-Impagliazzo [16], Buss-Grigoriev-Impagliazzo-Pitassi [26], Alekhnovich-Razborov [7].
- Bounded depth Frege systems with counting mod m axioms (for fixed m); Ajtai [2], Beame-Impagliazzo-Krajíček-Pitassi-Pudlák [12], and Buss-Impagliazzo-Krajíček-Pudlák-Razborov-Sgall [28].
- Intuitionistic and modal logics; Hrubeš [52, 51, 53].
- Ordered binary decision diagram (OBDD) proof systems; Atserias-Kolaitis-Vardi [9], Krajíček [66], Segerlind [107].
- Static and tree-like Lovász-Schrijver proofs; Dash [39], Itsykson-Kojevnikov [58], Beame-Pitassi-Segerlind [15], Lee-Shraibman [73], Chattopadhyay-Ada [30], Pitassi-Segerlind [87].

A reader unfamiliar with the above proof systems can consult the surveys [14, 106] for an overview. In short, *bounded depth Frege systems* are Frege systems in which the alternations of $\wedge$'s and $\vee$'s is bounded; *cutting planes systems* are proof systems that reason about linear inequalities over the integers with 0/1-valued variables; *nullstellensatz systems* and the *polynomial calculus* are systems that reason with polynomials over finite fields; and *counting axioms* express the impossibility of getting two different size remainders when partitioning a set into sets of size m.

Some of lower bound results listed above are inspired by analogies with two notable results from complexity theory. First, the lower bounds for bounded depth Frege system are lower bounds for proofs of the pigeonhole principle. These lower bounds are inspired by the Ajtai-Yao-Håstad switching lemma, but a more sophisticated switching lemma had to be developed, by [70, 86], to obtain the lower bounds for bounded depth Frege systems. Similarly, the lower bounds for cutting planes and Lovász-Schrijver systems, for intuitionistic and modal logics, and for OBDD systems are based on the clique-coloring tautologies; for these lower bounds, a Craig interpolation theorem was proved, and then a reduction was made to Razborov's theorem [93, 8] that monotone circuits for clique versus colorability require near-exponential size. Another popular set of tautologies for lower bounds is the Tseitin graph tautologies, which has been used for lower bounds on resolution and for a number of algebraic proof systems [111, 112, 46, 26, 58].

It is interesting to note that the lower bounds for the systems listed above are *all* based on counting. That is to say, the lower bounds are either for the propositional pigeonhole principle, or for tautologies based on clique-coloring principles, or for tautologies based on mod p counting principles for some prime p. The clique-coloring principle expresses the fact that if a graph has a clique of size n, then any coloring requires at least n colors; this is a variant of the pigeonhole principle of course.[4] Likewise, the Tseitin principle is also based on counting, in particular, on counting mod 2.

The main open problems at the "frontier" of Cook's program are perhaps the following:

Open problems.

1. Give exponential (or at least better than quasipolynomial) separations between depth k Frege systems and depth $k + 1$ Frege systems for low depth formulas, even for DNF formulas.

2. Prove superpolynomial lower bounds for bounded depth Frege systems augmented with mod p gates, for p a prime.

The first problem is closely connected with the problem of proving non-conservativity results for fragments of bounded arithmetic. The second problem has been conjectured to be do-able, in light of the analogy with the exponential lower bounds for constant depth circuits that use mod p gates [94, 110]. Indeed, one might expect to get significant separation between the power of mod p gates and mod q gates for distinct primes p and q. Overall, however, both problems have proved very resistant to progress. One partial step towards solving the first problem is due to Krajíček [62]: using a modified notion of depth called "Σ-depth", he showed that there are sets of clauses of Σ-depth k which can be refuted with short Σ-depth $k+1$ sequent calculus proofs but require large refutations in the Σ-depth k sequent calculus. In addition, Impagliazzo and Krajíček [54], using counting techniques of Paris and Wilkie [81], describe proofs of restricted pigeonhole principles in the fragments T^k_1 of bounded arithmetic: together with the lower bounds from [70, 86] this implies there is no polynomial simulation of depth $k + 1$ Frege systems by depth k Frege systems.

[4]The clique-coloring principle is a form of the pigeonhole principle since it follows from the fact that the vertices in a clique must all have distinct colors. However, a level of indirection is required to prove the clique-coloring principle from the pigeonhole principle. The cutting planes system can formalize a counting to a certain extent and do have simple polynomial-size proofs of the pigeonhole principle; however, the cutting planes system requires near-exponential size proofs for clique-coloring tautologies. The best lower bounds for OBDD proofs of the clique-coloring tautologies require yet another level of indirection: OBDD proofs have polynomial size proofs of the pigeonhole principle, but require exponential size for the clique-coloring tautologies when obfuscated by a permutation [66, 107].

This leaves open the question of whether there is a quasipolynomial separation between depth d and $d+1$ Frege systems. The best result towards solving the second problem is due to Maciel and Pitassi [76] who prove a separation of tree-like Frege proofs in which cuts are restricted to bounded depth formulas based on a circuit complexity conjecture; however, their lower bounds apply only to proofs of unbounded depth formulas. For additional open problems in proof complexity, see Pudlák [91].

As mentioned above, the strongest present-day lower bounds for propositional proof length use mostly counting principles such as the pigeonhole principle and the the clique-coloring tautologies. On the other hand, Frege systems can formalize counting and prove the pigeonhole principle [21]. A similar (ostensibly weaker) system, is the TC^0-Frege proof system, which allows reasoning with bounded depth formulas that can contain special counting gates or threshold gates. Since both these systems can prove facts about counting, and have polynomial size proofs of the pigeonhole principles and the clique-coloring principles, it may well require a completely new approach to give superpolynomial lower bounds for TC^0-Frege or Frege proofs.

3. The hardness of proof search

Cook's program concerns primarily questions about proof length; it does not directly concern the very important questions of the complexity of proof search. In this section, we list some of the more important results about the difficulty of searching for proofs.

The first result is about the difficulty of finding a proof that is approximately as short as possible:

Theorem 5. (Alekhnovich-Buss-Moran-Pitassi [4] and Dinur-Safra [42]) *For almost all common proof systems (resolution, Frege, nullstellensatz, sequent calculus, cut-free sequent calculus, etc.), it is impossible to approximate shortest proof length to within a factor of* $2^{\log^{1-o(1)} n}$ *in polynomial time, unless* $P = NP$.

Note that the theorem states more than it is difficult to search for a short proof; instead, it is already hard to determine whether such a proof exists (assuming $P \neq NP$). The proof of this theorem by [4] uses a reduction from the Minimum Monotone Circuit Satisfying Assignment problem [42].

It would be desirable to improve Theorem 5 by replacing the factor $2^{\log^{1-o(1)} n}$ with a factor n^k, for all fixed values $k > 0$.

The second result is about the difficulty of searching for Frege proofs.

Definition 6. A proof system P is *automatizable* if there is a procedure, which given a formula φ with shortest P-proof of length n, finds some P-proof of φ in time $poly(|\varphi| + n)$.

Theorem 7. (Bonet-Pitassi-Raz [19]) *If Frege proofs (or,* TC^0*-Frege proofs) are automatizable, then there is a polynomial time algorithm for factoring Blum integers.*

Blum integers are products of two primes congruent to 3 mod 4. It is generally conjectured that factorization of Blum integers is not in P. By the theorem, this would imply further that (TC^0-)Frege systems are not automatizable.

The proof of Theorem 7 uses automatizability to establish a feasible Craig interpolation property. The feasible Craig interpolation property can be used to give a polynomial time algorithm to break the Diffie-Hellman cryptographic protocol, which by results of Biham, Boneh, and Reingold would yield a polynomial time algorithm for factoring Blum integers. Similar methods were subsequently used by [17] to prove that bounded depth Frege systems are also not automatizable, under stronger assumptions on the hardness of factoring Blum integers.

Although Theorem 7 was originally stated for proof search (automatizability), it also can be recast as a theorem about the hardness of determining the *existence* of proofs.

Definition 8. The *provability* problem for a proof system P is the problem of, given a formula φ, deciding whether $P \vdash \varphi$. The *k-provability* problem for P is the problem of, given a pair $\langle \varphi, 1^k \rangle$, determining whether φ has a P-proof of length $\leq k$.

Note that 1^k just means that k is specified in unary notation. The intent is that a polynomial time algorithm for the k-provability problem can run for $k^{O(1)}$ time. A minor modification of the methods of [19] yields the following theorem.

Theorem 9. *If the k-provability problem for Frege systems (or, TC^0-Frege systems) is in P, then there is a polynomial time algorithm for factoring Blum integers.*

The third result we mention is about the hardness of proof search for resolution.

Theorem 10. (Alekhnovich-Razborov [6]) *Neither resolution nor tree-like resolution is automatizable unless the weak parameterized hierarchy $W[P]$ is fixed-parameter tractable under randomized algorithms with one-sided error.*

For more on the the weak parameterized hierarchy, see [43]. The proof of Theorem 10 uses a reduction from the Minimum Monotone Circuit Satisfying Assignment problem. Like the earlier results, Theorem 10 does not really depend on automatizability per se; just on whether there is a polynomial time algorithm that solves the k-provability problem. Theorem 10 was extended to the polynomial calculus by [44]. It is still open whether (tree-like) resolution, bounded depth Frege, and the polynomial calculus are *quasi-automatizable*, where "quasi-automatizable" is defined like "automatizable", but with a quasipolynomial runtime bound.

For common propositional proof systems, such as resolution and (extended) Frege systems, the k-provability problem in is NP, since one can merely nondeterministically guess the proof. However, it is unlikely that their *provability* problems are in *NP*, since, by the completeness of these logics, the provable formulas are the tautologies. The set of tautologies is *coNP*-complete, and hence is generally conjectured to not be in *NP*.

There are some weaker systems for which the provable formulas are known to form an *NP*-complete set. These are described in the next theorem.

Theorem 11. *The provability problems for the following systems are NP-complete.*

1. *The multiplicative fragment of linear logic.* (Lincoln-Mitchell-Scedrov-Shankar [75] and Kanovich [60].)

2. *The Lambek calculus, L.* (Pentus [85] and Savateev [103].)

3. *The reflected justification logic,* rLP. (Krupski [72] and Buss-Kuznets [29].)

4. Practical propositional proof search

The previous section described complexity results about the difficulty of finding propositional proofs. We now turn our attention to the opposite problem; namely, the question of how efficient can practical algorithms be for deciding validity. There is an enormous range of algorithms and software implementations aimed at deciding the validity of propositional formulas, far more than space will permit us to discuss. Thus, we discuss only a few core ideas from the field.

For the most part, the existing algorithms attack the *satisfiability* problem, instead of the validity problem. These two problems are essentially equivalent, since a formula φ is valid if and only if $\neg\varphi$ is not satisfiable. However, in the case of a satisfiable formula, one is often interested in finding an example of a satisfying assignment rather than just obtaining a Yes/No answer about satisfiability. Thus, we define the satisfiability problem, SAT, in a slightly non-standard way:

Definition 12. An instance of SAT is a set of clauses interpreted as a formula in conjunctive normal form. The *Satisfiability problem* (SAT) is the problem of, given an instance Γ of SAT, finding either a satisfying assignment, or finding a refutation of Γ. For $k \geq 2$, the k-SAT problem is the SAT problem restricted to clauses of size $\leq k$.

The notion of what constitutes a refutation of Γ is left unspecified; in effect it can be in any proof system. In particular, for an algorithm M that for satisfiable Γ always finds a satisfying assignment, one can just take a failing run of M as a refutation of Γ.

Complexity upper bounds for SAT. Let the set Γ of clauses have m clauses that involve n distinct variables. The obvious brute force algorithm that exhaustively searches all truth assignments has runtime $O(m \cdot 2^n)$. One can do slightly better than this, and some of the best known asymptotic upper bounds are given in the next theorem. The first result of this type was due to Monien and Speckenmeyer [80]. The first part of the next theorem is a recent result due to Hertli [48], extending prior work of Paturi, Pudlák, and Zane [84]; Paturi, Pudlák, Saks and Zane [83]; and Schöning [104]; Iwama and Tamaki [59], Hertli, Moser and Scheder [49], and many others. The second part is due to Schuler [105]. The theorem is stated for probabilistic algorithms; there are deterministic algorithms known with similar, but slightly worse, bounds.

Theorem 13.

1. [48] *Let* $k \geq 3$ *and* $\mu_k = \sum_{j=1}^{\infty} \left(j^2 + \frac{j}{(k-1)} \right)^{-1}$. *There is a probabilistic algorithm for* k-SAT *with expected runtime* $2^{cn+o(n)}$ *with* $c = 1 - \frac{u_k}{k-1}$. *For* $k = 3$ *and* $k = 4$, *the bound gives* $c \approx 0.386$ *and* $c \approx 0.554$, *respectively. As* $k \to \infty$, $\mu_k \to \frac{\pi^2}{6}$.

2. [105] *There is a probabilistic algorithm for* SAT *with expected runtime* $poly(m, n) 2^{(1-1/(1+\log m))n}$.

It is interesting to note that the first bound, for k-SAT depends only on the number of variables, not on the number of clauses. Santhanam [102] has given an improvement on the second bound for the case of $m = O(n)$ and even for Boolean formulas which have size $O(n)$.

It is natural to conjecture that the bounds in Theorem 13 are qualitatively optimal; that is to say, any algorithm for k-SAT or SAT must have runtime at least $\Omega(2^{\epsilon n})$ for some constant ϵ. (See [55, 56, 82].) For SAT, one might even conjecture that $\epsilon < 1$ is not achievable. Of course, the primary evidence for this conjecture is our inability to have discovered any better algorithms yet. Nonetheless, the conjecture is appealing: it says in effect that SAT is an extremely efficient way to encode nondeterminism, since it implies that brute-force search for satisfying assignments for SAT cannot be improved upon in any significant way.

More practical algorithms for SAT. There are large number of practical algorithms for solving SAT (and even annual contests comparing algorithms). These algorithms are aimed primarily at solving the kinds of satisfiability problems that arise in "real-world" applications, for instance from hardware or software verification. Most of the more successful algorithms are based on the DPLL algorithms; thusly named as an amalgamation of the initials of the authors from the two seminal papers that originated the techniques, Davis and Putnam [41] and Davis, Loveland, and Logemann [40]. The basic framework of the DPLL algorithm is rather simple, essentially just a brute-force search through all possible truth assignments, but backtracking as soon as any clause is falsified.

DPLL Algorithm:

```
        Γ is a set of clauses, and σ is a partial truth assignment.
DPLL_Search(Γ, σ):
    1. If $\Gamma \upharpoonright \sigma$ is falsified, return false.
    2. If $\Gamma \upharpoonright \sigma$ is satisfied, exit. ($\sigma$ is a satisfying assignment.)
    3. Choose a literal $x$ that is not assigned by $\sigma$.
    4. Call DPLL_Search($\Gamma, \sigma \cup \{x \mapsto \mathit{True}\}$).
    5. Call DPLL_Search($\Gamma, \sigma \cup \{x \mapsto \mathit{False}\}$).
    6. Return false.
```

The notation $\Gamma \upharpoonright \sigma$ denotes the set Γ as modified by the restriction σ. That is to say, every literal set false by σ is erased and every clause that contains a literal set true is removed. If $\Gamma \upharpoonright \sigma$ is empty, then all clauses are satisfied, so Γ is satisfied. If $\Gamma \upharpoonright \sigma$, contains an empty clause, then it is falsified.

There are many possible strategies for the literal selection step (Step 3); two of the basic ones are *unit propagation* and *pure literal selection*. For unit propagation, one chooses any literal x that occurs in a singleton clause in $\Gamma \upharpoonright \sigma$. For pure literal selection, one chooses any literal x such that $\overline{x}$ does not appear in $\Gamma \upharpoonright \sigma$. If x is chosen by unit propagation or as a pure literal, then the subsequent Step 5 can be skipped. Both these strategies can be used with no essential loss in efficiency.

A major improvement to the DPLL algorithm is the method of *clause learning* introduced by Marques-Silva and Sakallah [79]. Clause learning is method of learning (or, inferring) new clauses when a contradiction is found in Step 1 of the DPLL algorithm. When $\Gamma \upharpoonright \sigma$ is falsified, a "reason" for the falsification is extracted from the information gathered by the run of the DPLL algorithm. This "reason" is a clause C such that $\Gamma \vDash C$. The clause C is then *learned*; that is to say, it is added to the set Γ. (For a more complete introduction to clause learning, consult [79] or [13].)

The intuition behind clause learning is that a learned clause C can now be reused without needing to be re-derived. This can avoid repeating the same subsearch. The presence of new learned clauses also allows more opportunities for unit propagation, which also speeds up searching.

Another important feature of clause learning algorithms is that they keep track of the "level" at which variables are set; namely, when a variable assignment $x \mapsto \mathit{True}/\mathit{False}$ is added to σ, the variable x is labeled with the level (the depth of recursion) at which its value was set. When a contradiction is discovered in Step 1, the DPLL algorithm can use the information about levels to backtrack multiple levels at once. This process is called "fast backtracking" or "non-chronological backtracking", and it makes DPLL much less sensitive to the order in which variables are selected since it is often possible to backtrack past all the variable assignments that did not contribute to the contradiction.

Figure 1 illustrates the power of clause learning, even for a rather difficult family of tautologies. The pigeonhole principles PHP^m_n are sets of clauses, whose

	No Learning		Clause Learning	
Formula	Steps	Time (s)	Steps	Time
PHP^4_3	5	0.0	5	0.0
PHP^7_6	719	0.0	129	0.0
PHP^9_8	40319	0.3	769	0.0
PHP^{10}_9	362879	2.5	1793	0.5
PHP^{11}_{10}	3628799	32.6	4097	2.7
PHP^{12}_{11}	39916799	303.8	9217	14.9
PHP^{13}_{12}	479001599	4038.1	20481	99.3

Figure 1: This table shows the number of decision variables that are set ("steps") and the approximate runtime (in seconds) for determining satisfiability of pigeonhole principle statements, with and without clause learning. The number of steps with no learning is always $n! - 1$.

unsatisfiability express the pigeonhole principle that there is no one-to-one mapping from a set of size m to a set of size n (for $m > n$). For the results reported in Figure 1, PHP^m_n contains the following clauses:

(1) $\{p_{i,1}, p_{i,2}, \ldots, p_{i,n}\}$, for $i = 1, \ldots, m$, and

(2) $\{\overline{p}_{i,j}, \overline{p}_{i',j}\}$, for $1 \le i < i' \le m$ and $j = 1, \ldots, n$.

Figure 1 shows the results of a DPLL satisfiability solver software package (written by the author) both with and without clause learning. The table shows the number of decision variables set during the run of the DPLL algorithm: this counts only the number of variables whose value is not already forced to a value. That is to say, variables that are set by unit propagation or by the pure literal rule are not counted. As the table shows, learning can reduce the size of the search space enormously. The runtimes are substantially faster for clause learning as well, even though there is extra overhead associated with learning clauses and maintaining learned clauses. The software implementation used for the table is not as efficient at working with clauses as it might be; important optimizations like two-literal watching and garbage collection have not been implemented. Thus, a more complete modern DPLL implementation would likely see much quicker runtimes for clause learning.

Although the table shows an impressive improvement from the use of clause learning for the pigeonhole principles, this is not really where clause learning is the most advantageous. Clause learning has a much greater benefit for many "practical" or "real-world" applications, for instance for solving instances of satisfiability that come from applications such as software or hardware verification. In fact, DPLL algorithms with clause learning can routinely solve instances of satisfiability with hundreds of thousands of variables, at least for problems that arise in industrial applications.

Algorithms for satisfiability have been compared extensively in annual "SAT competitions" or "SAT Race competitions"; it is quite frequently the case that solvers based on DPLL algorithms with clause learning are the best. The best algorithms use a wide variety of techniques to improve the DPLL search and clause learning. These include:

- Literal selection heuristics, see [78, 77, 45].
- Clause learning strategies for deciding which clause(s) to learn, such as the first-UIP strategy [79].
- Clause forgetting strategies (garbage collection).
- Restart strategies, in which the DPLL search is interrupted and started again with the learned clauses.
- Execution optimizations. These include the two literal watching method [77], and many other optimizations of data structures and algorithms.

A logical characterization of clause learning. From its inception, the DPLL algorithm has been closely linked with propositional resolution. Indeed, the basic DPLL algorithm (without clause learning) corresponds exactly to the traversal of a tree-like resolution refutation. It is a more difficult problem to give a logical characterization of DPLL algorithms with clause learning, but some substantial progress has been made by [13, 113, 27]. Beame, Kautz and Sabharwal [13] characterized the process of learning clauses in terms of input resolution; Van Gelder [113] models the search process for DPLL with clause learning in a system called "pool resolution", which corresponds to resolution proofs that admit a regular depth-first traversal; and Buss, Hoffman, and Johannsen [27] combine these with a "w-resolution" inference that allows a precise characterization of certain types of DPLL algorithms with clause learning.

Definition 14. A *w-resolution inference* with respect to the variable x is an inference of the form

$$\frac{C \qquad\qquad D}{(C \setminus \{x\}) \cup (D \setminus \{\overline{x}\})}$$

In place of dag-like proofs, one can use tree-like proofs with *lemmas*, where a lemma is a formula that was derived earlier in the proof. That is to say, in a tree-like refutation of a set Γ using lemmas, the leaves of the tree are either formulas from Γ or are lemmas. It is clear that tree-like proofs with lemmas are equivalent to dag-like proofs.

Definition 15. Let T be a tree-like refutation with lemmas.

- An *input proof* is a proof in which every inference has at least one hypothesis which is a leaf.
- An *input clause* in T is any clause which is derived by an input sub-proof of T.

- The proof T is a WRTI-proof provided it is a tree-like proof, using w-resolution inferences, in which (only) input clauses are used as lemmas. (WRTI is an acronym for "W-Resolution Tree-like with Input lemmas".)

As the next theorem shows, using input lemmas is as strong as using arbitrary lemmas, and thus WRTI is as strong as full resolution.

Theorem 16. (Buss-Hoffmann-Johannsen [27].) *General dag-like resolution proofs can be simulated by WRTI proofs.*

Definition 17. A resolution refutation is *regular* provided that no variable is used twice as a resolution variable on any single path through the proof. In particular, a WRTI proof is regular provided that no variable is used twice as the active variable of a w-resolution on any branch in the refutation tree.

The next theorem allows an almost exact characterization of DPLL search with clause learning. We restrict to input lemmas since the clause learning algorithms can learn only clauses that can be derived by input proofs [13]. The restriction to regular refutations corresponds to the fact that a DPLL search procedure never branches twice on the same variable; that is to say, once a variable's value is set, it cannot be changed (until after backtracking resets the variable). We say the correspondence is "almost" exact, since we have to allow the DPLL search algorithm to be "non-greedy", by which is meant that the DPLL search procedure may ignore contradictions and continue to branch on variables even after a contradiction has been achieved.

Theorem 18. (Buss-Hoffmann-Johannsen [27].) *Regular WRTI proofs are polynomially equivalent to non-greedy DPLL search with clause learning.*

The reader should refer to [27] for the technical details of what kinds of non-greedy DPLL search with clause learning are permitted. However all the standard clause learning algorithms from [79, 13] are allowed, including first-UIP, all-UIP, first cut clauses, rel sat clauses, decision clauses, etc.

There are a number of important theoretical questions about DPLL search algorithms. First, it would be nice to improve Theorem 18 to give a logical characterization of *greedy* DPLL clause learning. This ideally would also accommodate the more restrictive forms of clause learning and backtracking of common DPLL implementations. Second, it is open whether regular WRTI is as strong as general resolution. Similarly, it is open whether pool resolution directly simulates general resolution. (However, [10] gives an indirect simulation of general resolution by pool resolution.) It is known that regular resolution does not simulate either general resolution or pool resolution [5, 113]. Possibly this can be extended to prove that pool resolution does not simulate general resolution (but see [18]). Third, it would be highly desirable to find substantially more efficient SAT algorithms: either more efficient DPLL algorithms or more efficient algorithms of the types discussed in Theorem 13.

5. Trial Approaches to *NP–P* via Proof Complexity

This concluding section discusses some of the approaches to separating P and NP using proof complexity. Of course, these approaches have all failed so far, and it appears that some fundamentally new idea will be needed to prove $P \neq NP$.[5] Nonetheless it is interesting to review some of the extant ideas.

Via diagonalization and incompleteness. Returning to the analogy between the NP–P problem and Gödel's incompleteness theorem about the impossibility of proving self-consistency statements, it is natural to consider the following *partial* consistency statements.

Definition 19. Fix a proof system P. The statement $\mathrm{Con}(P, n)$ is a tautology expressing the principle that there is no P-proof of a contradiction of length $\leq n$.

The size of the formula $\mathrm{Con}(P, n)$ is bounded by a polynomial of n (at least, for standard proof systems P). Although one might hope that partial self-consistency statements might require super-polynomial proofs, this does not in fact happen:

Theorem 20. (Cook [35], Buss [22].) *For P either a Frege system or an extended Frege system, there are polynomial size P-proofs of* $\mathrm{Con}(P, n)$.

The idea of the proof is to express a partial truth definition for formulas of length $\leq n$, using a formula whose length is polynomial in n. This proof method is analogous to the constructions of Pudlák [88, 89] for partial self-consistency proofs in first-order theories of arithmetic and in set theory. (Conversely, there are lower bounds on the lengths of first order proofs of partial consistency statements which match the known upper bounds to within a polynomial, due to H. Friedman and Pudlák.)

Hard tautologies from pseudo-random number generators. Krajíček [64] and independently Alekhnovich, Ben-Sasson, Razborov, and Widgerson [3] made the following conjecture about the hardness of tautologies based on pseudo-random number generators. For the purpose of the present discussion, it will suffice to treat a pseudo-random number generator as being any function f which maps $\{0,1\}^n$ to $\{0,1\}^m$, where m is a function of n and $m > n$. More generally, a pseudo-random number generator is intended to be a function of this type for which is it difficult to distinguish strings in $Range(f)$ from strings not in $Range(f)$, even with multiple trials and with the aid of randomization.

Given a pseudo-random number generator f, define tautologies $\tau(f)_b$ by fixing a particular string $b \in \{0,1\}^m$ and forming a propositional formula

$$\tau(f)_b \;:=\; \text{“}f(\langle \vec{q} \rangle) \neq b\text{”}, \tag{1}$$

[5]However, conversely, to prove $P = NP$ might "merely" require a new algorithm.

where $\vec{q}$ is the vector $q_1, \ldots, q_n$ of propositional variables that appear in $\tau(f)_b$. (In a moment, we shall use the notation $\tau(f)_b(\vec{q})$ when we need to specify the variables.) Assuming f is computable by a polynomial size propositional formula, then $\tau(f)_b$ has size bounded by a polynomial of n. More generally, if f is computable by a polynomial size circuit, then $\tau(f)_b$ can be expressed as a propositional formula with the aid of additional propositional variables that give intermediate values calculated by the circuit.

The suggestion [64, 3] is that the tautologies $\tau(f)_b$ might require superpolynomially (or even, exponentially) long propositional proofs.[6] Razborov [97] also conjectures that the Nisan-Widgerson functions give pseudo-random number generators such that the tautologies (1) require superpolynomially long Frege proofs. If this could be established for *all* proof systems, it would imply, of course, that $NP \neq coNP$.

Krajíček [65] introduces further tautologies based on iterating the idea behind the tautologies (1). Namely, let $k > 0$ and for $i = 1, \ldots, k$ let $\vec{q}^{\,i}$ be a vector of n variables. Further let B_i be a Boolean circuit that outputs a vector of m variables, with the inputs to B_i being $\vec{q}^{\,1}, \ldots, \vec{q}^{\,i-1}$. Let b_i denote the value of $B_i(\vec{q}^{\,1}, \ldots, \vec{q}^{\,i-1})$. (So b_0 is just a string of fixed Boolean values, $b_0 \in \{0,1\}^m$.) Consider the following formulas:

$$\tau(f)_{b_0}(\vec{q}^{\,0}) \vee \tau(f)_{b_1}(\vec{q}^{\,1}) \vee \cdots \vee \tau(f)_{b_k}(\vec{q}^{\,k}). \tag{2}$$

These formulas will be tautologies in many situations, such as when b_0 is not in the range of f. The function f is called *pseudo-surjective* provided that these formulas do not have proofs of size bounded by a polynomial of n. Krajíček conjectures that there do exist pseudo-surjective functions for Frege and extended Frege. As evidence for why this conjecture is reasonable, Krajíček shows that truth-table functions f (see [95]) that are based on the explicit evaluation of Boolean circuits can form a kind of canonical pseudo-surjective functions (assuming pseudo-surjective functions exist); he further links their pseudo-surjectivity to the (un)provability of circuit lower bounds.

So far, superpolynomial lower bounds for the τ-tautologies (1) have been given for resolution, some small extensions of resolution, and the polynomial calculus [65, 3, 97]. On the other hand, Krajíček [67] has pointed out that, for any propositional proof system that does not have polynomial size proofs of the pigeonhole principle tautologies PHP_n^{n+1}, it is possible to build pseudo-random number generators f (in the weak sense as defined above), for which the tautologies (1) require superpolynomial size. This construction uses a direct reduction from (a failure of) the pigeonhole principle to construct onto functions mapping $\{0,1\}^n$ onto $\{0,1\}^{n+1}$. It is perhaps disheartening that this construction also covers all the proof systems for which more conventional pseudo-random functions f had been used to get lower bounds on proof complexity.

[6]The formulas $\tau(f)_b$ are tautologies if b is not in the range of f. However, if b is in the range of f, then $\tau(f)_b$ does not have any proof at all.

Nonetheless, the pseudo-random number generator tautologies remain an interesting approach. The conjectures of [64, 3, 97] that the tautologies (1) and (2) require superpolynomial size proofs are very reasonable and seem likely to be true. They have also been used for conditional constructions of nonstandard models [68]. In addition, the pseudo-random number generator tautologies are related to constructions used for natural proof methods.

Natural proof methods. The pseudo-random number generator method described above was inspired in large part by the "natural proof" method of Razborov and Rudich [100] which gives barriers to proofs of $P \neq NP$. The natural proof barriers are based on the assumption that certain kinds of computationally hard pseudo-random number generators exist. In related work, Razborov [95] used natural proofs to obtain results about the inability of certain theories of bounded arithmetic to prove superpolynomial lower bounds to circuit size. (We shall omit the technical details here; the reader can consult [95], [63], or [24] for more details.) Although Razborov's independence results for bounded arithmetic assumed the existence of strong pseudo-random generators, Krajíček [65] subsequently pointed out that the results could be obtained without this assumption by using the non-provability of the pigeonhole principle in these theories (and using an idea of Razborov [98] that if the pigeonhole principle fails, then large circuits can be compressed into small circuits).

It is interesting to note that natural proofs inject an element of self-reference into the *NP*–*P* problem. Namely, the natural proofs construction of [100] shows that if a very strong form of $P \neq NP$ holds, then $P \neq NP$ is hard to prove in certain proof systems. Loosely speaking, one might interpret this as saying that the truth of $P \neq NP$ (if it is true) might be part of the reason it is hard to prove $P \neq NP$.[7] Of course, this is a further impetus for studying the τ-tautologies discussed above.

6. Conclusions

It is frustrating that the proof complexity approach to complexity has not yet managed to surpass the "counting barrier" in any substantial way. Indeed, most lower bound results in proof complexity either can be categorized as a form of diagonalization (and thus are limited by the Baker-Gill-Solovay oracle constructions) or can be reduced to or subsumed by a failure of counting or of the pigeonhole principle. Of course, this is not a criticism of the proof complexity approach as compared to other approaches, since no other approaches have had much more success. Indeed, proof complexity remains a highly interesting and promising approach toward the *NP*–*P* problem and other complexity questions. In addition, as a method of attacking complexity questions, proof complexity

[7]Admittedly, one could just as well interpret this as an indication that P is equal to NP, but this is an interpretation that we prefer not to make.

has the advantage of being inherently different from circuit complexity, and thus provides a fundamentally different line of attack on the *NP–P* problem.

We conclude with a few rather general open problems. (1) Computational complexity has a rich structure that exploits randomization and cryptographic constructions in sophisticated ways. This includes results such as PCP theory, (de)randomization, expander graphs, communication complexity, etc. It would be highly desirable to bring more of these methods into proof complexity. (2) Algebraic complexity has been a successful tool for computational complexity: there should be more linkages possible between algebraic complexity and proof complexity. (3) One should develop randomized constructions that transcend the pigeonhole principle. In particular, proof complexity needs to find a way to break past the "counting barrier". Conversely, perhaps it can be shown that breaking past the "counting barrier" is as hard as separating computational classes.

Acknowledgement. We thank J. Krajíček and A. Razborov for comments on an earlier draft of this paper. We also thank two anonymous referees for their careful reading and useful comments.

References

[1] M. Ajtai, *The complexity of the pigeonhole principle*, in Proceedings of the 29-th Annual IEEE Symposium on Foundations of Computer Science, 1988, pp. 346–355.

[2] ——, *The independence of the modulo p counting principles*, in Proceedings of the 26th Annual ACM Symposiom on Theory of Computing, Association for Computing Machinery, 1994, pp. 402–411.

[3] M. Alekhnovich, E. Ben-Sasson, A. Razborov, and A. Widgerson, *Pseudorandom generators in propositional proof complexity*, in Proc. 41st IEEE Conf. on Foundations of Computer Science (FOCS), 2000, pp. 43–53.

[4] M. Alekhnovich, S. Buss, S. Moran, and T. Pitassi, *Minimum propositional proof length is NP-hard to linearly approximate*, Journal of Symbolic Logic, 66 (2001), pp. 171–191.

[5] M. Alekhnovich, J. Johannsen, T. Pitassi, and A. Urquhart, *An exponential separation between regular and general resolution*, Theory of Computation, 3 (2007), pp. 81–102.

[6] M. Alekhnovich and A. A. Razborov, *Resolution is not automatizable unless $W[P]$ is tractable*, in Proc. 42nd IEEE Conf. on Foundations of Computer Science (FOCS), 2001, pp. 210–219.

[7] ——, *Lower bounds for polynomial calculus: Nonbinomial case*, Proceedings of the Steklov Institute of Mathematics, 242 (2003), pp. 18–35.

[8] B. Alon and R. Boppana, *The monotone circuit complexity of Boolean functions*, Combinatorica, 7 (1987), pp. 1–22.

[9] A. Atserias, P. G. Kolaitis, and M. Y. Vardi, *Constraint propogation as a proof system*, in Proc. Tenth International Conf. on Principles and Practice of Constraint Programming, 1974, pp. 77–91.

[10] F. Bacchus, P. Hertel, T. Pitassi, and A. Van Gelder, *Clause learning can effectively p-simulate general propositional resolution*, in Proc. 23rd AAAI Conf. on Artificial Intelligence (AAAI 2008), AAAI Press, 2008, pp. 283–290.

[11] P. Beame, *Proof complexity*, in Computational Complexity Theory, IAS/Park City Mathematical Series, Vol. 10, American Mathematical Society, 2004, pp. 199–246. Lecture notes scribed by Ashish Sabharwal.

[12] P. Beame, R. Impagliazzo, J. Krajíček, T. Pitassi, and P. Pudlák, *Lower bounds on Hilbert's Nullstellensatz and propositional proofs*, Proceedings of the London Mathematical Society, 73 (1996), pp. 1–26.

[13] P. Beame, H. A. Kautz, and A. Sabharwal, *Towards understanding and harnessing the potential of clause learning*, J. Artificial Intelligence Research, 22 (2004), pp. 319–351.

[14] P. Beame and T. Pitassi, *Propositional proof complexity: Past, present and future*, in Current Trends in Theoretical Computer Science Entering the 21st Century, G. Paun, G. Rozenberg, and A. Salomaa, eds., World Scientific, 2001, pp. 42–70. Earlier version appeared in *C*omputational Complexity Column, Bulletin of the EATCS, 2000.

[15] P. Beame, T. Pitassi, and N. Segerlind, *Lower bounds for Lovász-Schrijver systems and beyond follow from multiparty communication complexity*, SIAM Journal on Computing, 37 (2007), pp. 845–869.

[16] E. Ben-Sasson and R. Impagliazzo, *Random CNF's are hard for the polynomial calculus*, in Proceedings 40th IEEE Conference on Foundations of Computer Science (FOCS), 1999, pp. 415–421.

[17] M. Bonet, R. Gavalda, C. Domingo, A. Maciel, and T. Pitassi, *Non-automatizability for bounded-depth Frege systems*, Computational Complexity, 13 (2004), pp. 47–68.

[18] M. L. Bonet and S. R. Buss, *An improved separation of regular resolution from pool resolution and clause learning.* Manuscript, to be submitted, 2011.

[19] M. L. Bonet, T. Pitassi, and R. Raz, *On interpolation and automatization for Frege systems*, SIAM Journal on Computing, 29 (2000), pp. 1939–1967.

[20] H. Buhrman, S. Fenner, L. Fortnow, and D. can Melkebeek, *Optimal proof systems and sparse sets*, in Proc. 17th Symp. on Theoretical Aspects of Computer Science (STACS), Lecture Notes in Computer Science #1770, Springer Verlag, 2000, pp. 407–418.

[21] S. R. Buss, *Polynomial size proofs of the propositional pigeonhole principle*, Journal of Symbolic Logic, 52 (1987), pp. 916–927.

[22] ——, *Propositional consistency proofs*, Annals of Pure and Applied Logic, 52 (1991), pp. 3–29.

[23] ——, *Lectures in proof theory*, Tech. Rep. SOCS 96.1, School of Computer Science, McGill University, 1996. Notes from a series of lectures given at the at the McGill University Bellair's Research Institute, Holetown, Barbados, March 1995. Scribe notes written by E. Allender, M.L. Bonet, P. Clote, A. Maciel, P. McKenzie, T. Pitassi, R. Raz, K. Regan, J. Torán and C. Zamora.

[24] ——, *Bounded arithmetic and propositional proof complexity*, in Logic of Computation, H. Schwichtenberg, ed., Springer-Verlag, Berlin, 1997, pp. 67–121.

[25] ——, *Propositional proof complexity: An introduction*, in Computational Logic, U. Berger and H. Schwichtenberg, eds., Springer-Verlag, Berlin, 1999, pp. 127–178.

[26] S. R. Buss, D. Grigoriev, R. Impagliazzo, and T. Pitassi, *Linear gaps between degrees for the polynomial calculus modulo distinct primes*, Journal of Computer and System Sciences, 62 (2001), pp. 267–289.

[27] S. R. Buss, J. Hoffmann, and J. Johannsen, *Resolution trees with lemmas: Resolution refinements that characterize DLL-algorithms with clause learning*, Logical Methods of Computer Science, 4, 4:13 (2008), pp. 1–18.

[28] S. R. Buss, R. Impagliazzo, J. Krajíček, P. Pudlák, A. A. Razborov, and J. Sgall, *Proof complexity in algebraic systems and bounded depth Frege systems with modular counting*, Computational Complexity, 6 (1996/1997), pp. 256–298.

[29] S. R. Buss and R. Kuznets, *The NP-completeness of reflected fragments of justification logics*, in Proc. Logical Foundations of Computer Sciences (LFCS'09), Lecture Notes in Computer Science #5407, 2009, pp. 122–136.

[30] A. Chattopadhyay and A. Ada, *Multiparty communication complexity of disjointness*, Tech. Rep. TR08-002, Electronic Colloquium in Computational Complexity (ECCC), 2008. 15pp.

[31] Y. Chen and J. Flum, *On p-optimal proof systems and logics for PTIME*, in Proc. 37th International Colloquium on Automata, Languages, and Programming (ICALP'10), Lecture Notes in Computer Science 6199, Springer, 2010, pp. 312–313.

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

[33] P. Clote and J. Krajíček, *Arithmetic, Proof Theory and Computational Complexity*, Oxford University Press, 1993.

[34] S. A. Cook, *The complexity of theorem proving techniques*, in Proceedings of the 3-rd Annual ACM Symposium on Theory of Computing, 1971, pp. 151–158.

[35] ———, *Feasibly constructive proofs and the propositional calculus*, in Proceedings of the Seventh Annual ACM Symposium on Theory of Computing, 1975, pp. 83–97.

[36] ———, *An overview of computational complexity*, Journal of the ACM, 26 (1983), pp. 401–408.

[37] S. A. Cook and R. A. Reckhow, *On the lengths of proofs in the propositional calculus, preliminary version*, in Proceedings of the Sixth Annual ACM Symposium on the Theory of Computing, 1974, pp. 135–148.

[38] ———, *The relative efficiency of propositional proof systems*, Journal of Symbolic Logic, 44 (1979), pp. 36–50.

[39] S. Dash, *Exponential lower bounds on the lengths of some classes of branch-and-cut proofs*, Mathematics of Operations Research, 30 (2005), pp. 678–700.

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

[41] M. Davis and H. Putnam, *A computing procedure for quantification theory*, Journal of the Association for Computing Machinery, 7 (1960), pp. 201–215.

[42] I. Dinur and S. Safra, *On the hardness of approximating label cover*, Information Processing Letters, 89 (2004), pp. 247–254.

[43] R. G. Downey and M. R. Fellows, *Parameterized Complexity*, Springer, 1999.

[44] N. Galesi and M. Lauria, *On the automatizability of the polynomial calculus*, Theory of Computing Systems, 47 (2010), pp. 491–506.

[45] E. Goldberg and Y. Novikov, *BerkMin: A fast and robust Sat-solver*, Discrete Applied Mathematics, 155 (2007), pp. 1549–1561.

[46] D. Grigoriev, *Nullstellensatz lower bounds for Tseitin tautologies*, in Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science, IEEE Computer Society Press, 1998, pp. 648–652.

[47] A. Haken, *The intractability of resolution*, Theoretical Computer Science, 39 (1985), pp. 297–308.

[48] T. Hertli, *3-SAT faster and simpler — Unique-SAT bounds for PPSZ hold in general.* Preprint, arXiv:1103.2165v2 [cs.CC]., May 2011.

[49] T. Hertli, R. A. Moser, and D. Scheder, *Improving PPSZ for 3-SAT using critical variables*, in Proc. 28th Symp. on Theoretical Aspects of Computer Science (STACS), 2011, pp. 237–248.

[50] E. A. Hirsch, D. Itsykson, I. Monakhov, and A. Smal, *On optimal heuristic randomized semidecision procedures, with applications to proof complexity and cryptography*, Tech. Rep. TR10-193, Electronic Colloquium in Computational Complexity (ECCC), 2010.

[51] P. Hrubeš, *A lower bound for intuitionistic logic*, Annals of Pure and Applied Logic, 146 (2007), pp. 72–90.

[52] ——, *A lower bound for modal logics*, Journal of Symbolic Logic, 72 (2007), pp. 941–958.

[53] ——, *On lengths of proofs in non-classical logics*, Annals of Pure and Applied Logic, 157 (2009), pp. 194–205.

[54] R. Impagliazzo and J. Krajíček, *A note on conservativity relations among bounded arithmetic theories*, Mathematical Logic Quarterly, 48 (2002), pp. 375–377.

[55] R. Impagliazzo and R. Paturi, *On the complexity of k-SAT*, Journal of Computer and System Sciences, 62 (2001), pp. 367–375.

[56] R. Impagliazzo, R. Paturi, and F. Zane, *Which problems have strongly exponential complexity*, Journal of Computer and System Sciences, 63 (2001), pp. 512–530.

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

[58] D. M. Itsykson and A. A. Kojevnikov, *Lower bounds of static Lovász-Schrijver calculus proofs for Tseitin tautologies*, Journal of Mathematical Sciences, 145 (2007). Translated from *Zapiski Nauchnykh Seminarov POMI*, Vol 340, 2006, pp. 10-32.

[59] K. Iwama and S. Tamaki, *Improved upper bounds for 3-SAT*, in Proc. 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2004, pp. 328–329.

[60] M. I. Kanovich, *The complexity of Horn fragments of linear logic*, Pure and Applied Logic, 69 (1994), pp. 195–241.

[61] J. Köbler, J. Messner, and J. Toran, *Optimal proof systems imply complete sets for promise classes*, Information and Computation, 184 (2003), pp. 71–92.

[62] J. Krajíček, *Lower bounds to the size of constant-depth propositional proofs*, Journal of Symbolic Logic, 59 (1994), pp. 73–86.

[63] ——, *Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic*, Journal of Symbolic Logic, 62 (1997), pp. 457–486.

[64] ——, *On the weak pigeonhole principle*, Fundamenta Mathematica, 170 (2001), pp. 123–140.

[65] ——, *Dual weak pigeonhole principle, pseudo-surjective functions, and provability of circuit lower bounds*, Journal of Symbolic Logic, 69 (2004), pp. 265–286.

[66] ——, *An exponential lower bound for a constraint propogation proof system based on ordered binary decision diagrams*, Journal of Symbolic Logic, 73 (2008), pp. 227–237.

[67] ——, *A proof complexity generator*, in Proc. 13th International Congress of Logic, Methodology and Philosophy of Science, C. Glymour, W. Wanng, and D. Weststahl, eds., London, 2009, King's College Publications, pp. 185–190.

[68] ——, *Forcing with Random Variables and Proof Complexity*, Cambraidge University Press, 2011.

[69] J. Krajíček and P. Pudlák, *Propositional proof systems, the consistency of first-order theories and the complexity of computations*, Journal of Symbolic Logic, 54 (1989), pp. 1063–1079.

[70] J. Krajíček, P. Pudlák, and A. Woods, *Exponential lower bound to the size of bounded depth Frege proofs of the pigeonhole principle*, Random Structures and Algorithms, 7 (1995), pp. 15–39.

[71] G. Kreisel, *Hilbert's programme and the search for automatic proof procedures*, in Symposium on Automatic Demonstration: held at Versailles/France, December 1968, Lecture Notes in Mathematics 125, Berlin, 1970, Springer Verlag, pp. 128–146.

[72] N. V. Krupski, *On the complexity of the reflected logic of proofs*, Theoretical Computer Science, 357 (2006), pp. 136–142.

[73] T. Lee and A. Shraibman, *Disjointness is hard in the multiparty number-on-the-forehead model*, in Proc. 23rd IEEE Conference on Computational Complexity (CCC), 2008, pp. 81–91.

[74] L. Levin, *Universal search problems*, Problems of Information Transmission, 9 (1973), pp. 265–266. In Russian. English translation by B.A. Trakhtenroty, in *A survey of Russian approaches to perebor (brute-force searches) algorithms*, Annals of the History of Computing 6 (1984) 384-400.

[75] P. Lincoln, J. Mitchell, A. Scedrov, and N. Shankar, *Decision problems for propositional linear logic*, Annals of Pure and Applied Logic, 60 (1993), pp. 151–177.

[76] A. Maciel and T. Pitassi, *A conditional lower bound for a system of constant-depth proofs with modular connectives*, in Proc. Twenty-First IEEE Symp. on Logic in Computer Science (LICS'06), IEEE Computer Society Press, 2006, pp. 189–198.

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

[78] J. P. Marques-Silva, *The impact of branching hueristics in propositional satisfiability algorithms*, in Proc. 9th Portuguese Conference on Artificial Intelligence (EPIA), Springer Verlag, 1999, pp. 62–74.

[79] J. P. Marques-Silva and K. A. Sakallah, *GRASP — A new search algorithm for satisfiability*, IEEE Transactions on Computers, 48 (1999), pp. 506–521.

[80] B. Monien and E. Speckenmeyer, *Solving satisfiability in less than 2^n steps*, Discrete Applied Mathematics, 10 (1985), pp. 287–295.

[81] J. B. Paris and A. J. Wilkie, *Counting problems in bounded arithmetic*, in Methods in Mathematical Logic, Lecture Notes in Mathematics #1130, Springer-Verlag, 1985, pp. 317–340.

[82] M. Pătraşçu and R. Williams, *On the possibility of faster SAT alorithms*, in Proc. 21st ACM/SIAM Symp. on Discrete Algorithms (SODA), 2010, pp. 1065–1075.

[83] R. Paturi, P. Pudlák, M. E. Saks, and F. Zane, *An improved exponential-time algorithm for k-SAT*, Journal of the Association for Computing Machinery, 52 (2005), pp. 337–364.

[84] R. Paturi, P. Pudlák, and F. Zane, *Satisfiability coding lemma*, Chicago Journal of Theoretical Computer Science, (1999). Article 11.

[85] M. Pentus, *Lambek calculus is NP-complete*, Theoretical Computer Science, 357 (2006), pp. 186–201.

[86] T. Pitassi, P. Beame, and R. Impagliazzo, *Exponential lower bounds for the pigeonhole principle*, Computational Complexity, 3 (1993), pp. 97–140.

[87] T. Pitassi and N. Segerlind, *Exponential lower bounds and integrality gaps for tree-like Lovász-Schrijver procedures*, in Proc. 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2009, pp. 355–364.

[88] P. Pudlák, *On the lengths of proofs of finitistic consistency statements in first order theories*, in Logic Colloquium '84, North-Holland, 1986, pp. 165–196.

[89] ——, *Improved bounds to the lengths of proofs of finitistic consistency statements*, in Logic and Combinatorics, vol. 65 of Contemporary Mathematics, American Mathematical Society, 1987, pp. 309–331.

[90] ——, *Lower bounds for resolution and cutting planes proofs and monotone computations*, Journal of Symbolic Logic, 62 (1997), pp. 981–998.

[91] ——, *Twelve problems in proof complexity*, in Computer Science — Theory and Applications, Lecture Notes in Computer Science #5010, Berlin, Heidelberg, 2008, Springer, pp. 13–27.

[92] R. Raz, *Resolution lower bounds for the weak pigeonhole principle*, in Proc. 34th ACM Symp. on Theory of Computing (STOC), 2002, pp. 553–562.

[93] A. A. Razborov, *Lower bounds on the monotone complexity of some Boolean functions*, Soviet Mathematics Doklady, 31 (1985), pp. 354–357.

[94] ——, *Lower bounds on the size of bounded depth networks over a complete basis with logical addition*, Matematischi Zametki, 41 (1987), pp. 598–607. English translation in *Mathematical Notes of the Academy of Sciences of the USSR* 41(1987) 333-338.

[95] ——, *Unprovability of lower bounds on the circuit size in certain fragments of bounded arithmetic*, Izvestiya of the RAN, 59 (1995), pp. 201–224.

[96] ——, *Lower bounds for the polynomial calculus*, Computation Complexity, 7 (1998), pp. 291–324.

[97] ——, *Pseudorandom generators hard for k-DNF resolution and polynomial calculus resolution.* Manuscript, available at author's web site, 67 pp., 2003.

[98] ——, *Resolution lower bounds for the weak pigeonhole principle*, Theoretical Computer Science, 303 (2003), pp. 233–243.

[99] ——, *Resolution lower bounds for perfect matching principles*, Journal of Computer and System Sciences, 69 (2004), pp. 3–27.

[100] A. A. Razborov and S. Rudich, *Natural proofs*, Journal of Computer and System Sciences, 55 (1997), pp. 24–35.

[101] Z. Sadowski, *Optimal proof systems, optimal acceptors and recursive presentability*, Fundamenta Informaticae, 79 (2007), pp. 169–185.

[102] R. Santhanam, *Fighting perebor: New and improved algorithms for formula and QBF satisfiability*, in Proc. 51st IEEE Symp. on Foundations of Computer Science (FOCS), 2010, pp. 183–192.

[103] Y. Savateev, *Product-free Lambek calculus is NP-complete*, in Proc. 2009 Intl. Symp. on Logical Foundations of Computer Science (LFCS'09), Lecture Notes in Computer Science #5407, Springer Verlag, 2009, pp. 380–394.

[104] U. Schöning, *A probabilistic algorithm for k-SAT based on limited local search and restart*, Algorithmica, 32 (2008), pp. 615–623.

[105] R. Schuler, *An algorithm for the satisfiability problem of formulas in disjunctive normal form*, Journal of Algorithms, 54 (2005), pp. 40–44.

[106] N. Segerlind, *The complexity of propositional proofs*, Bulletin of Symbolic Logic, 13 (2007), pp. 417–481.

[107] ——, *On the relative efficiency of resolution-like proofs and ordered binary decision diagram proofs*, in Proc. 23rd Annual IEEE Conference on Computational Complexity (CCC'08), 2008, pp. 100–111.

[108] J. Siekmann and G. Wrightson, *Automation of Reasoning*, vol. 1&2, Springer-Verlag, Berlin, 1983.

[109] M. Sipser, *The history and status of the P versus NP question*, in Proceedings of the Twenty-fourth Annual ACM Symposium on Theory of Computing, ACM, 1992, pp. 603–618.

[110] R. Smolensky, *Algebraic methods in the theory of lower bounds for Boolean circuit complexity*, in Proceedings of the Nineteenth Annual ACM Symposium on the Theory of Computing, ACM Press, 1987, pp. 77–82.

[111] G. S. Tsejtin, *On the complexity of derivation in propositional logic*, Studies in Constructive Mathematics and Mathematical Logic, 2 (1968), pp. 115–125. Reprinted in: [108, vol 2], pp. 466-483.

[112] A. Urquhart, *Hard examples for resolution*, J. Assoc. Comput. Mach., 34 (1987), pp. 209–219.

[113] A. Van Gelder, *Pool resolution and its relation to regular resolution and DPLL with clause learning*, in Logic for Programming, Artificial Intelligence, and Reasoning (LPAR), Lecture Notes in Computer Science Intelligence 3835, Springer-Verlag, 2005, pp. 580–594.