Towards NP − P and Satisfiability via Proof Complexity and Proof Search

Sam Buss

Logical Foundations of Computer Science, 2009

A *logical* reason for $P \neq NP$?

A '"logical" reason, rather than a "combinatorial" reason.

G. Kreisel, Symp. on Automatic Deduction, LNM #125, 1968

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

"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."

What "logical" reasons are there for believing $\mathrm{P} \not= \mathrm{NP}$?

One is that $\mathrm{P} = \mathrm{NP}$ would make the practice of mathematics too easy: Proof search could be automated (automatized) by formalizing mathematical questions completely and then blindly searching for proofs of conjectured statements. If $\mathrm{P} = \mathrm{NP}$, this process could succeed whenever proofs are not too large. This would be a major change in the practice of mathematics!

Gödel [1956 letter to von Neumann]

"...consequences of the greatest importance. ... The mental work of a mathematician concerning Yes-No questions could be completely replaced by a machine."

See [Buss 1995, in Feas. Math. II] for a detailed discussion.

A related objection is that it would mean mathematics would become completely formal, with little room left for human intuition and understanding. This feels unlikely and undesirable.

Given a particular proof system P.

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

These questions are interesting even for propositional proof systems. Indeed, under some special assumptions, the existence of propositional proofs is an NP-complete problem, and hence a feasible method to find optimal propositional proofs will also give feasible algorithm to find proofs in *any* proof system.

Propositional proof systems

Common propositional proof systems:

- Resolution – proof system for proving DNF formulas.
- Frege proofs – textbook system based on modus ponens.
- Extended Frege/extended resolution – allow introduction of new variables that abbreviate other formulas (clauses).

Definition (Proof length)

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

Extended Frege systems can equivalently be characterized in terms of Frege proof systems with proof length measured in terms of number of lines or steps in the proof.

Cook's Program for proving NP ≠ coNP

Definition (Cook-Reckhow 1975)

An *abstract proof system* is a polynomial time function from $\{0,1\}^*$ onto the set of tautologies.

A traditional proof system can be viewed as an abstract system by letting $f(w)$ equal the formula proved by the proof w. In this way, one can form strong proof systems, even treating ZF as a propositional proof system.

Theorem

There exists an (abstract) proof system in which all tautologies have polynomial size proofs if and only if $\mathrm{NP} = \mathrm{coNP}$.

A system with this property is called *super*.

Cook's program for separating NP and coNP

Prove that stronger and stronger proof systems are not super, until it is established for all abstract proof systems.

Known (near-)exponential lower bounds on proof length include:

- Method of truth-tables
- Tree-like resolution/Regular resolution [Tseitin, 1968]
- Resolution [Haken, 1985; Raz, 2004; Razborov, 2003]
- Bounded depth Frege systems [Beame-Pitassi-Impagliazzo/Krajíček-Pudlák-Woods, 1992]
- Cutting Planes system [Pudlák, 1997]
- Nullstellensatz systems [Buss-Impagliazzo-Krajíček-Pudlák-Razborov-Sgall, 1996]
- Bounded depth Frege systems with counting mod m axioms (fixed m). [BIKPRS'96]
- Intuitionistic and modal propositional logics. [Hrubes, 2007.]

At the frontier of proof length lower bounds

Open problems

- Separate depth k Frege from depth $k+1$ Frege with CNF formulas. [This is closely related to non-conservativity properties conjectured for fragments of Bounded Arithmetic.]
- Lower bounds for bounded depth Frege systems with mod p counting *gates*, or with counting gates (TC^0-Frege).

For the systems listed on the previous page, the exponential lower bounds on proof length are obtained for the **pigeon-hole principle tautologies** or for the **clique-coloring graph tautologies**.
In effect, the systems lack the ability to count.

The Frege system, TC^0-Frege, and stronger systems can carry out counting arguments:

Theorem (Buss 1986)

There are polynomial size Frege proofs of the the pigeonhole principle tautologies.

Open problem

- Find problems other than the pigeonhole principle or the clique-coloring principle to serve as lower bounds for stronger propositional proof systems.

Some theorems about the hardness of proof search

Theorem (Aleknovitch-Buss-Pitassi-Moran, 2000)

For almost all natural proof systems (resolution, Frege, nullstellensatz, sequent, cut free sequent, etc.), it is impossible to approximate shortest proof length to a factor of $2^{\log^{1-o(1)} n}$ *in polynomial time, unless* $\mathrm{P} = \mathrm{NP}$.

The proof uses a reduction from Minimum Monotone Circuit Satisfying Assignment.

Definition

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 $poly(n)$.

Theorem (Bonet-Pitassi-Raz, 1997)

If Frege proofs (or, TC^0-Frege proofs) are automatizable, then factorization of Blum integers is in P.

The proof uses a feasible Craig interpolation construction based on the Diffie-Hellman cryptographic protocol.

Both interpolation and k-provability for (TC^0-)Frege systems are shown not to be polynomial time, unless factorization of Blum integers is in polynomial time.

A *Blum integer* is a product of two primes both congruent to 3 mod 4.

Theorem (Alekhnovitch-Razborov, 2001)

If resolution is automatizable, then the weak parameterized complexity hierarchy $\mathrm{W}[\mathrm{P}]$ *collapses.*

The proof uses a reduction from Minimum Monotone Circuit Satisfying Assignment.

Theorem (Krupski, 2006; Buss-Kuznets, this meeting)

For the reflected Logic of Proofs, rLP*, deciding provability, or k-provability, is* NP*-complete.*

Theorem (Pentus, 2006; Savateev, this meeting)

For certain fragments of the Lambek calculus, deciding provability, or k-provability, is NP*-complete.*

Towards practical algorithms for Satisfiability

Problem: Given an instance Γ of SAT, a set of clauses, find a satisfying assignment, or produce a refutation.

Theorem (Monien-Speckenmeyer, 1985; $\cdots$; Paturi-Pudlák-Saks-Zane/Pudlák, 1998; Schöning, 1999; Iwama-Tamaki, 2004)

- *There are algorithms for 3-SAT that run in time $2^{c \cdot n}$ with $c < 1$.*
- *There are algorithms for k-SAT that run in time $2^{(1-1/k)n}$.*
- *There are algorithms for SAT that run in time $m2^{n-\epsilon\sqrt{n}}$.*

n = number of variables.
m = number of clauses.
Best constant so far: $2^c = 1.324$. (IT'04).

Practical algorithms for "real-world" satisfiability testing mostly use the DPLL algorithm:
[Davis-Putnam, 1960; Davis-Loveland-Logemann, 1962]

DPLL algorithm. Γ - Instance of SAT, σ - partial truth assignment.

$\underline{\mathrm{DPLL\_Search}(\Gamma, \sigma)}$:

0. If $\Gamma \restriction \sigma$ is falsified, return false.
1. If $\Gamma \restriction \sigma$ is satisfied, exit (σ is satisfying assignment).
2. Choose a literal x.
3. Call $\mathrm{DPLL\_Search}(\Gamma, (\sigma \cup (x \mapsto \mathit{True})))$.
4. Call $\mathrm{DPLL\_Search}(\Gamma, (\sigma \cup (x \mapsto \mathit{False})))$.
5. Return false.

The literal selection (step 2.) usually exploits *unit propagation* and *pure literals*.

The DPLL algorithm is essentially equivalent to tree-like, regular resolution, since it corresponds to traversing a resolution refutation from the final (empty) clause.

Clause Learning. (Marques Silva & Sakallah, 1996)

Clause learning is a method of learning (inferring) new clauses when a contradiction is found during DPLL. When Γ is falsified, a "reason" for the falsification is extracted: this "reason" is a clause C such that $\Gamma \models C$. C is learned; that is, C is added to Γ.

The intuition is that the learned clause C can now be reused without needing to be re-derived.

Clause learning is generally combined with a fast backtracking method that allows for automatically backtracking as far as possible to eliminate the current contradiction. This makes the algorithm much less sensitive to the choice of branching literal x (in step 2).

Example: Pigeonhole tautologies.

Formula	Clause Learning		No Learning	
	Steps	Time (s)	Steps	Time
PHP_3^4	5	0.0	5	0.0
PHP_6^7	129	0.0	719	0.0
PHP_8^9	769	0.0	40319	0.3
PHP_9^{10}	1793	0.5	362879	2.5
PHP_{10}^{11}	4097	2.7	3628799	32.6
PHP_{11}^{12}	9217	14.9	-	-

More importantly, DPLL with clause learning does *much* better on "real-world", structured problems, e.g., from software or hardware verification. In some settings, it can frequently solve problems with hundreds of thousands of variables.

The important aspects of clause learning include:

- Literal selection heuristics,
- Clause learning strategies,
- Clause forgetting strategies (garbage collection),
- Restart strategies,
- Execution optimizations.

The clause learning strategies use unit propagation, keep track of the level (stage) at which variables are set, and choose conflicts so as to improve the quality of the learned clauses and the ability to backtrack efficiently.

A Logical Characterization of Clause Learning

From [Beame-Kautz-Sabharwal, 2004; van Gelder, 2005; Buss-Hoffmann-Johannsen, ta], we can give a proof theoretic characterization of the power of DPLL algorithms with clause learning.

Definition (BHJ)

A *w-resolution inference* with variable x is of the form

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

A *regular* proof is one in which no resolution variable used twice on a single path.

Instead of dag-proofs, one can use tree-like proofs with *lemmas*, a lemma being a formula derived earlier in the proof. Thus, in a tree refutation of Γ using lemmas, the leaves of the trees are either formulas in Γ or are lemmas.

Definition

- An *input proof* is a tree-like proof in which every inference has at least one hypothesis which is a leaf.
- An *input clause* is a clause derived by an input sub-proof.
- A WRTI-proof is a tree-like w-resolution proof in which input clauses may be used as lemmas.

Theorem (BHJ)

General dag resolution proofs can be polynomially simulated by WRTI proofs.

Theorem (BHJ)

The regular WRTI proofs are polynomially equivalent to non-greedy clause learning DPLL algorithms.

"Non-greedy" = Contradictions can be ignored.

The clause learning algorithms simulated include all the standard learning methods including first UIP clauses, all-UIP clauses, rel sat clauses, decision clauses, first cut clauses. The original Marques Silva & Sakallah methods are enough for the converse simulation.

Open problems

- Does regular WRTI directly polynomially simulate general dag-like resolution? Similarly for pool resolution?
- Find better logical characterizations for DPLL clause learning algorithms, say for greedy algorithms.
- Find more efficient SAT algorithms, either improving on DPLL clause learning, or improving the exponential time SAT algorithms. Find connections between these two approaches to satisfiability.

Some trial approaches to $\mathrm{NP}-\mathrm{P}$ via proof complexity.

1. Via diagonalization and incompleteness:

Definition

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

Theorem (Cook, 1975; Buss, 1991)

For P either Frege or extended Frege system, there are polynomial-size P-proofs of $Con(P, n)$.

Proof idea is to express a partial truth definition for formulas of length $\leq n$. Similar in spirit to constructions by Gödel, Friedman, Pudlák for first order systems of arithmetic.

2. [Krajíček, 2001; Alekhnovitch, Ben-Sasson, Razborov, Widgerson, 2000].
Assume f is a pseudo-random number generator $f : \{0,1\}^n \to \{0,1\}^m$, with $m > n$, specified by a polynomial size formula or circuit. Fix $\mathbf{w}_0 \in \{0,1\}^m$. Let $\mathbf{x} \in \{0,1\}^n$. Form a formula $\varphi(\vec{x})$ expressing

$$f(\mathbf{x}) \neq \mathbf{w}_0.$$

Conjecture: For any formal system, there is a pseudo-random number generator such that $\varphi(\mathbf{x})$ requires near-exponential size.

Theorem

Krajíček, 2007 The conjecture holds for any system in which the pigeonhole principle is not provable.

The motivation for the use of the pseudo-random number generator tautologies is to use, say, the natural proofs barrier of [Razborov-Rudich, 1997], to obtain lower bounds on propositional proof length.

So far, however, the natural proofs approach, when applied to propositional proofs or proofs in bounded arithmetic ($S^2_2(\alpha)$), has been successful in showing the hardness of proofs of propositional tautologies expressing $\mathrm{NP} \neq \mathrm{coNP}$ only for systems in which the pigeonhole principle fails badly.

In these systems, it is consistent that there is a 1-1 correspondence between n and 2^{n^ϵ}, and hence any exponential size circuit can be consistently formulated over only polynomially many nodes.

Open problems

Find ways to apply constructions in computational complexity to propositional proof complexity. For instance,

- Find better linkage of proof complexity to complexity lower bounds and cryptographic constructions and pseudo-randomness,
- Better algebraic lower bound methods,
- Randomized methods that transcend the pigeonhole principle.
- Break past the "counting barrier".

Open problems

Find ways to apply constructions in computational complexity to propositional proof complexity. For instance,

- Find better linkage of proof complexity to complexity lower bounds and cryptographic constructions and pseudo-randomness,
- Better algebraic lower bound methods,
- Randomized methods that transcend the pigeonhole principle.
- Break past the "counting barrier".

the end