# Proof Complexity & Complexity of SAT Solvers

Sam Buss
Univ. of California, San Diego

SAT/SMT Summer Workshop
June 17, 2011

# Disconnect between theory and practice

Concerning the SAT problem, and more generally proof search:

***Theory:*** SAT is NP-complete and hence conjectured intractable.

***Practice:*** SAT is "efficient in practice". [Vijay Ganesh, Monday's talk]

The resolution of these two viewpoints is that "Theory" considers worst-case performance and instances of SAT that arise from computationally intractable problems, whereas "Practice" considers problems that arise in industrial situations and are *"Big and shallow".*

# Part I: Theory (The bad news)

Let P be a (propositional) proof system. We would like to solve the following problems:

- *Provability problem*: Given a formula Á, decide if it has a short P-proof.
- *Proof search:* Given a formula Á that has a short P-proof, find a P-proof.
- Characterize the formulas that have reasonable length (polynomial length) P-proofs.
- Compare the strength of P with other proof systems.

In many cases, the first two problems are NP-hard, and thus conjectured to be infeasible to solve.

# Some common proof systems

- DPLL with clause learning. (As is common for SAT solvers.)
- Resolution.
- Frege systems. (Textbook system, based on modus ponens.)
- Extended Frege systems/extended resolution. Frege plus the ability to introduce new variables that abbreviate formulas.

***Definition:*** The *length* of a proof is the number of symbols on the proof.

Extended Frege proofs can equivalently be defined as Frege proofs for which proof size is the number of *steps* in the proof.

***Definition:*** [Cook-Reckhow'75] An *abstract proof system* is a polynomial time function *f* mapping binary strings onto the tautologies.

Any traditional proof system can be viewed as an abstract proof systems by defining *f(w)* to equal the formula proved by the proof *w*. In this way, one can form very strong proof systems, even treating Peano arithmetic or ZF set theory as a propositional proof system.

***Theorem:*** [CR'75] There exists an (abstract) proof system in which all tautologies have polynomial size proofs if and only if NP=coNP.

This is "bad" news, since we conjecture NP≠coNP. Proof search is then also infeasible since proof *size* is already infeasible large! Worse, this applies (conjecturally) to *any* proof system.

***"Cook's Program" for proving NP≠coNP:*** Prove super-polynomial lower bounds for stronger and stronger proof systems, until it is established for all abstract proof systems.

***So far achieved for:***

- Truth tables
- Tree-like and regular resolution

- Resolution
- Bounded depth Frege systems, also with counting mod *m* axioms for fixed *m*.
- Cutting planes (integer linear inequalities); via Craig interpolation
- Nullstellensatz systems (polynomials, over fields)
- Intuitionistic and modal logics.
- OBDD (ordered BDD) proof systems.
- Certain Lovász-Schrijver systems (quadratic polynomials)

***"Cook's Program" for proving NP≠coNP:*** Prove super-polynomial lower bounds for stronger and stronger proof systems, until it is established for all abstract proof systems.

***So far achieved for:***

- Truth tables
- Tree-like and regular resolution
- **DPLL with clause learning**
- Resolution
- Bounded depth Frege systems, also with counting mod *m* axioms for fixed *m*.
- Cutting planes (integer linear inequalities); via Craig interpolation
- Nullstellensatz systems (polynomials, over fields)
- Intuitionistic and modal logics.
- OBDD (ordered BDD) proof systems.
- Certain Lovász-Schrijver systems (quadratic polynomials)

However, we do not have super-polynomial lower bounds on proof lengths for the following systems.

Bounded depth Frege with parity gates; or with mod *m* gates for any *m>1*; or with majority/threshold gates.

Frege systems

Extended Frege systems

Peano arithmetic, ZF set theory, etc.

# Hardness of proof search

***Theorem:*** [Alekhnovitch-Buss-Moran-Pitassi '00] For almost all natural proof systems (resolution, Frege, nullstellensatz, cut-free, etc.), it is impossible to approximate shortest proof length in polynomial to with a factor of $2^{\log^{1-o(1)} n}$ unless P=NP. (Where *n* is the length of a shortest proof.)

***Definition:*** A proof system P is *automatizable* if there is a procedure, which given a formula Á with a P-proof of length *n* finds some P-proof in time polynomial in *n*.

***Theorem*** [Bonet-Pitassi-Raz '97] If Frege proofs are automatizable, then factorization of Blum integers is in polynomial time.

***Theorem:*** [Alekhnovitch-Razborov '01] If resolution is automatizable, then the weak parameterized hierarch W[P} collapses.

None of the implicands above are believed to be true.

Hence we have:

***Moral (conjectural)*:**

Even when short proofs exist, it can be infeasible to find them.

# An independence result for NP and P/poly

***Definition:*** The *functional pigeonhole principle* $FPHP_n^m$ states there is no 1-1, onto map from *m* objects to *n<m* objects.

$$\begin{array}{lll} \bigvee_{k=1}^{n} x_{i,k}, & i \leq m & Totality \\ \overline{x}_{i,k} \vee \overline{x}_{j,k}, & i \neq j \leq m,\ k \leq n & Injectivity \\ \overline{x}_{i,k} \vee \overline{x}_{i,\ell}, & i \leq m,\ k \neq \ell \leq n & Functionality \\ \bigvee_{i=1}^{m} x_{i,k}, & k \leq n & Surjectivity \end{array}$$

The pigeonhole principles have been an important source of examples for upper and lower bounds on proof complexity (Resolution; Haken ’85; extended Frege systems; Cook ’75; Frege systems; Buss ’87; and many more).

***Theorem:*** [Razborov ‘04] $FPHP_n^m$ requires resolution proofs of size exponential in $n^{1/3}$.

***Corollary:*** [Razborov]. Resolution is unable to give polynomial size proofs of tautologies expressing the principle that NP is not in P/poly (i.e., that NP does not have polynomial size circuits).

***Proof idea:*** (Think of setting up a program correctness result as an instance of SAT.) Let $t>n^3$. Express the principle that SAT has a circuits of size *t* as a set of clauses of size *poly(t,*$2^n$*).* Consider a resolution refutation of these clauses.

But, SAT has obvious circuits of size $2^n$ : Resolution can not disprove the exist of a bijection from $2^n$ to *t.* Such a bijection can convert the $2^n$ -size circuit into a circuit of size *t.*

This (along with analogous, uniform results for fragments of bounded arithmetic) is currently the best unconditional formal non-provability result for P≠NP. (!)

# Part II: Practice (The good news?)

"SAT is efficient in practice", to the extent it holds, is true largely to the success of DPLL with Clause Learning [GRASP,Chaff,…] on industrial problems and combinatorial problems.

DPLL with Clause Learning depends on:

- Depth first search of solution space
- Efficient backtracking
- Literal selection heuristics
- Clause learning strategies
- Clause forgetting strategies (garbage collection)
- Restart strategies
- Execution optimizations

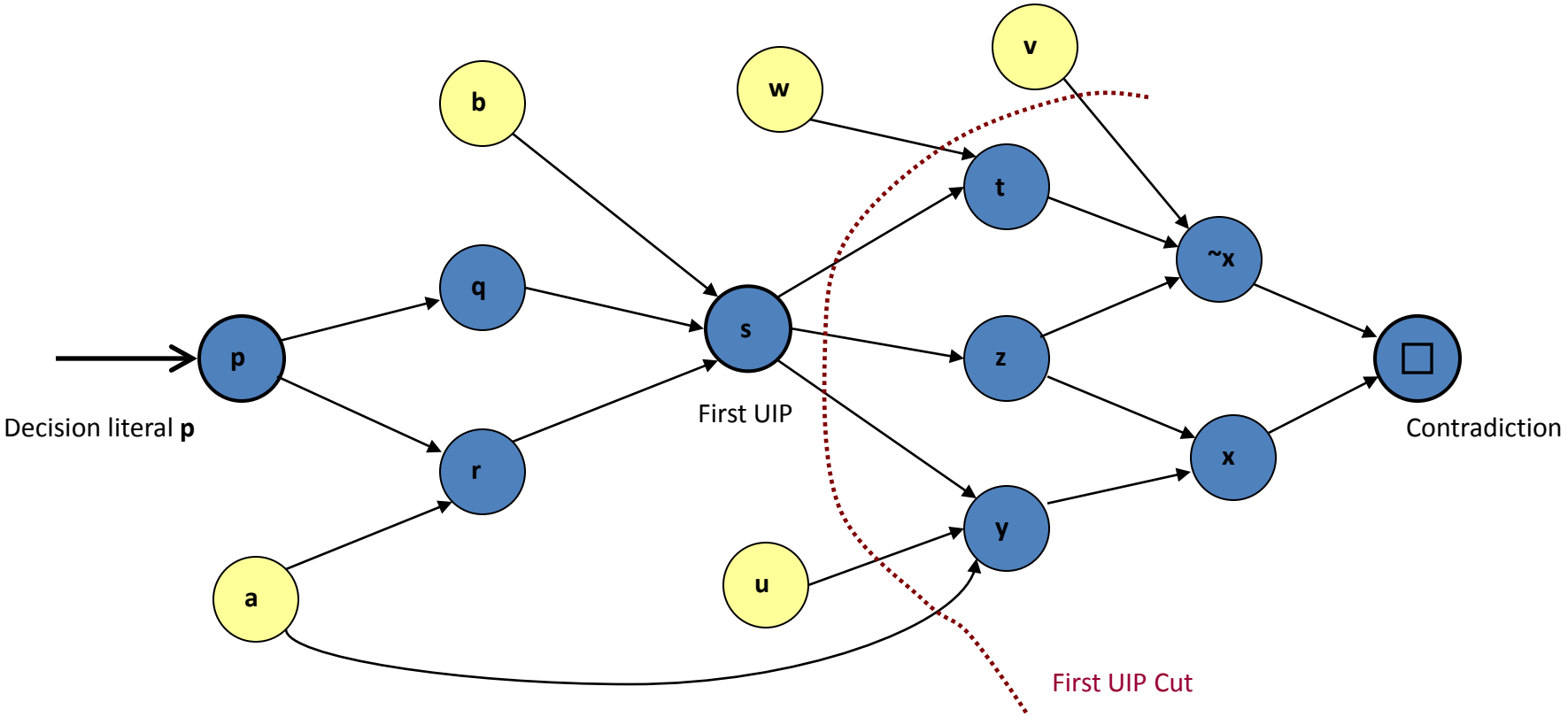
Good (?) news: For fixed *k>0,* our best time bounds for solving satisfiability are based essentially on: ***DPLL without Clause Learning, with random literal selection and restarts.***

***Theorem:*** [Hertli 'ta; see also HMS '11, PPSZ '05, …] There is randomized algorithm for 3-SAT with expected runtime $< 2^{\varepsilon n}$, where $\varepsilon = 0.39$ and $n$ is the number of variables.

The proof is based on selecting decision literals at random, using small derivations to check whether the literal value is forced (in effect, using unit propagation to decide if consistent to set a literal to a particular value). No backtracking, no clause learning is used.

*Challenge problem:* Show that clause learning, etc., can yield a better value for $\varepsilon$.

# Clause Learning – very important for DPLL




Clauses: {~y,~z, x}, {~p,~a,r}, etc. (One per unit propagation.)
First UIP Learned Clause: {~a,~u,~s,~w,~v}.
Whole top level learned clause: {~p,~a,~u,~b,~w,~v}.
With First-UIP: Both **p** and **s** can be set false when backtracking.
New level of **~s** is set to max. level of **u,v,w.**

Blue for top level

Yellow for lower level literal

Example: Clause Learning for Pigeonhole Tautologies (Injective/total):

| | No Learning | | Clause Learning | |
|---|---|---|---|---|
| Formula | Steps | Time (s) | Steps | Time |
| $PHP_3^4$ | 5 | 0.0 | 5 | 0.0 |
| $PHP_6^7$ | 719 | 0.0 | 129 | 0.0 |
| $PHP_8^9$ | 40319 | 0.3 | 769 | 0.0 |
| $PHP_9^{10}$ | 362879 | 2.5 | 1793 | 0.5 |
| $PHP_{10}^{11}$ | 3628799 | 32.6 | 4097 | 2.7 |
| $PHP_{11}^{12}$ | 39916799 | 303.8 | 9217 | 14.9 |
| $PHP_{12}^{13}$ | 479001599 | 4038.1 | 20481 | 99.3 |

Run times in *SatDiego*, version 1. Variable selection is by a "clause-greedy" method. No restarts.

Clause learning provides dramatic improvement, but is even much better suited for industrial, less combinatorially structured problems, where it can often handle hundreds of thousands of variables, or more.

# What is the Logical strength of DPLL with Clause Learning?

[B-Hoffmann-Johannsen '08] extending [Beame-Kautz-Sabharwal '04, Van Gelder '05]

***Definition:*** A *w-resolution inference* is of the form

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

A *regular* proof is one in which no single path uses the same resolution variable twice.

***Definition:*** An *input* proof is a tree-like proof in which every inference has at least one hypothesis which is a leaf (usually an initial clause, but also possibly a "lemma").

***Definition:*** An *input clause* of a proof is a clause derived by an input sub-proof.

Instead of dag-like proofs, consider *tree-like* proofs with *lemmas:*

Lemmas must be derived first, and then can be used freely later as initial clauses in the proof (to the right in the proof tree).

***Definition:*** A *WRTI-proof* is a tree-like w-resolution proof in which input clauses may be used as lemmas.

***Theorem:*** General dag resolution proofs can be polynomially simulated by WRTI proofs.

***Theorem:*** Regular WRTI proofs are polynomially equivalent to non-greedy DPLL proof search with clause learning (without restarts).

This theorem holds for all standard clause learning (1st UIP, all UIP, rel sat, first cut, etc.). The original Marques-Silva & Sakallah clause learning is sufficient for the converse simulation.

# For DPLL with Clause Learning and Restarts:

It not hard to see that DPLL with clause learning and restarts, if allowed non-greedy search, can simulate general resolution [BKS].

***Theorem:*** [Pipatsrisawat-Darwiche '10]. DPLL with clause learning and restarts can polynomially simulate general resolution – for appropriately choices of decision literals and any commonly used (greedy) learning strategy.

[Atserias-Fichte-Thurley] Similar result for width *k* resolution.

*General idea for proof:* Use clause learning to successively learn the clauses in a resolution proof, or at least learn enough so as to be able simulate unit propagation based on these clauses. Difficulty was to show that this works even in the presence of greedy clause learning.

The power of DPLL with clause learning and without restarts versus the power of resolution is unknown. The only known simulations are based on variable extensions [BKS, BHPvG, BHJ] and are unsatisfactory.

***Open question:***

- Does regular WRTI simulate general resolution?
- Does pool resolution [Van Gelder] simulate general resolution?

Both systems are known to be stronger than regular resolution. [Van Gelder '05, also Bonet-Buss 'ip]

# A Concluding Question

***Meta-Question:*** How relevant are theoretical analyses for understanding the power of DPLL and other practical algorithms for satisfiability?

For instance: Does the theorem about polynomially simulating resolution have anything to do with the source of the power of restarts? Or, are restarts powerful for other reasons?

*Other theories:* - "High variance in search difficulty"

- "Backdoor sets"
- "Clearing the search space"
- "Shake the apple tree (low-hanging fruit)"

*Thank you!*

Related survey paper:  Buss, “Towards NP-P via Proof Complexity and Search”, ta.