Everything's Bigger in Texas
The Largest Math Proof Ever

Solving and Verifying the Boolean Pythagorean Triples problem via Cube-and-Conquer

Marijn J.H. Heule

THE UNIVERSITY OF TEXAS AT AUSTIN

Joint work with Oliver Kullmann and Victor W. Marek

Rice University August 24, 2016

The Rise of Brute Reason

I can stand brute force, but brute reason is quite unbearable.
It is hitting below the intellect. Oscar Wilde, 1890

The Rise of Brute Reason

I can stand brute force, but brute reason is quite unbearable. It is hitting below the intellect. Oscar Wilde, 1890

Hilbert's Program (1900) on formalizing all mathematics and proving its consistency by very simple means.

Gödel's Incompleteness Theorem (1931) seemed to destroy the positive spirit of the time.

The Rise of Brute Reason

I can stand brute force, but brute reason is quite unbearable. It is hitting below the intellect. Oscar Wilde, 1890

Hilbert's Program (1900) on formalizing all mathematics and proving its consistency by very simple means.

Gödel's Incompleteness Theorem (1931) seemed to destroy the positive spirit of the time.

The mental work of a mathematician concerning Yes-or-No questions could be completely replaced by a machine.
Kurt Gödel in a letter to Von Neumann, 1956

Cook's Theorem (1971) on the NP-completeness of SAT tempered the hope of solving all decision problems efficiently.

The Rise of Brute Reason

I can stand brute force, but brute reason is quite unbearable. It is hitting below the intellect. Oscar Wilde, 1890

Hilbert's Program (1900) on formalizing all mathematics and proving its consistency by very simple means.

Gödel's Incompleteness Theorem (1931) seemed to destroy the positive spirit of the time.

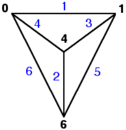
The mental work of a mathematician concerning Yes-or-No questions could be completely replaced by a machine.
Kurt Gödel in a letter to Von Neumann, 1956

Cook's Theorem (1971) on the NP-completeness of SAT tempered the hope of solving all decision problems efficiently.

Now SAT solving has emerged as a disruptive technology turning the fear of "you can't solve it!" into "solve it with SAT!"

Satisfiability (SAT) solving has many applications

formal verification

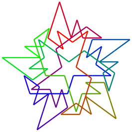


graph theory

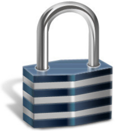
bioinformatics

train safety

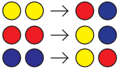
planning



combinatorics



cryptography

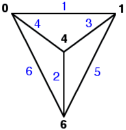


rewrite termination

encode → **SAT solver** → **decode**

Satisfiability (SAT) solving has many applications

formal verification

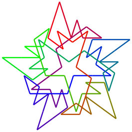


graph theory

bioinformatics

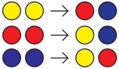
train safety

planning



combinatorics

cryptography



rewrite termination

encode

SAT solver

decode

Main challenges regarding solving hard problems using SAT:

- Can we achieve linear speedups on multi-core systems?
- Can we produce proofs to gain confidence in the results?

Pythagorean Triples Problem

Schur's Theorem [Schur 1917]

Can the set of natural numbers $\mathbb{N} = \{1, 2, 3, \ldots\}$ be k-colored such that there is no monochromatic solution of $a+b=c$ with $a < b < c$? Else, what is the smallest $[1, n]$ counterexample?

Schur's Theorem [Schur 1917]

Can the set of natural numbers $\mathbb{N} = \{1, 2, 3, \ldots\}$ be k-colored such that there is no monochromatic solution of $a+b=c$ with $a < b < c$? Else, what is the smallest $[1, n]$ counterexample?

Consider the case $k = 2$ with the colors named red and blue:

$$2\,3 \quad \rightarrow \quad 1\,2\,3 \quad 5 \quad \rightarrow \quad 1\,2\,3\,4\,5\,6 \quad \rightarrow \quad \times$$

decide

$$1 + 2 = 3 \qquad 1 + 4 = 5$$
$$2 + 3 = 5 \qquad 1 + 5 = 6$$
$$2 + 4 = 6$$

Schur's Theorem [Schur 1917]

Can the set of natural numbers $\mathbb{N} = \{1, 2, 3, \dots\}$ be k-colored such that there is no monochromatic solution of $a+b=c$ with $a < b < c$? Else, what is the smallest $[1, n]$ counterexample?

Consider the case $k = 2$ with the colors named red and blue:

$$2\ 3 \quad \rightarrow \quad 1\ 2\ 3 \quad 5 \quad \rightarrow \quad 1\ 2\ 3\ 4\ 5\ 6 \quad \rightarrow \quad \times$$

$$\text{decide} \qquad \begin{matrix} 1+2=3 \\ 2+3=5 \end{matrix} \qquad \begin{matrix} 1+4=5 \\ 1+5=6 \end{matrix} \qquad 2+4=6$$

Theorem (Schur's Theorem)

For each $k > 0$, there exists a number $S(k)$, known as Schur number, such that there exists a k-coloring of $[1, S(k)]$ without a monochromatic solution of $a + b = c$ with $a, b, c \leq S(k)$, while this is impossible for $[1, S(k)+1]$.

Pythagorean Triples Problem [Graham]

Can the set of natural numbers $\mathbb{N} = \{1, 2, 3, \dots\}$ be colored with red and blue such that there is no monochromatic Pythagorean triple ($a, b, c \in \mathbb{N}$ with $a^2 + b^2 = c^2$)? Otherwise, what is the smallest $[1, n]$ counterexample?

Best lower bound: a bi-coloring of $[1, 7664]$ s.t. there is no monochromatic Pythagorean triple [Cooper & Overstreet 2015].

Myers conjectures that the answer is yes [PhD thesis, 2015].

Pythagorean Triples Problem [Graham]

Can the set of natural numbers $\mathbb{N} = \{1, 2, 3, \dots\}$ be colored with red and blue such that there is no monochromatic Pythagorean triple ($a, b, c \in \mathbb{N}$ with $a^2 + b^2 = c^2$)? Otherwise, what is the smallest $[1, n]$ counterexample?

Best lower bound: a bi-coloring of $[1, 7664]$ s.t. there is no monochromatic Pythagorean triple [Cooper & Overstreet 2015].

Myers conjectures that the answer is yes [PhD thesis, 2015].

A bi-coloring of $[1, n]$ is encoded using Boolean variables x_i with $i \in \{1, 2, \dots, n\}$ such that $x_i = 1$ $(= 0)$ means that i is colored red (blue). For each Pythagorean triple $a^2 + b^2 = c^2$ two clauses are added: $(x_a \vee x_b \vee x_c) \wedge (\bar{x}_a \vee \bar{x}_b \vee \bar{x}_c)$.

Pythagorean Triples Problem [Graham]

Can the set of natural numbers $\mathbb{N} = \{1, 2, 3, \dots\}$ be colored with red and blue such that there is no monochromatic Pythagorean triple ($a, b, c \in \mathbb{N}$ with $a^2 + b^2 = c^2$)? Otherwise, what is the smallest $[1, n]$ counterexample?

Best lower bound: a bi-coloring of $[1, 7664]$ s.t. there is no monochromatic Pythagorean triple [Cooper & Overstreet 2015].

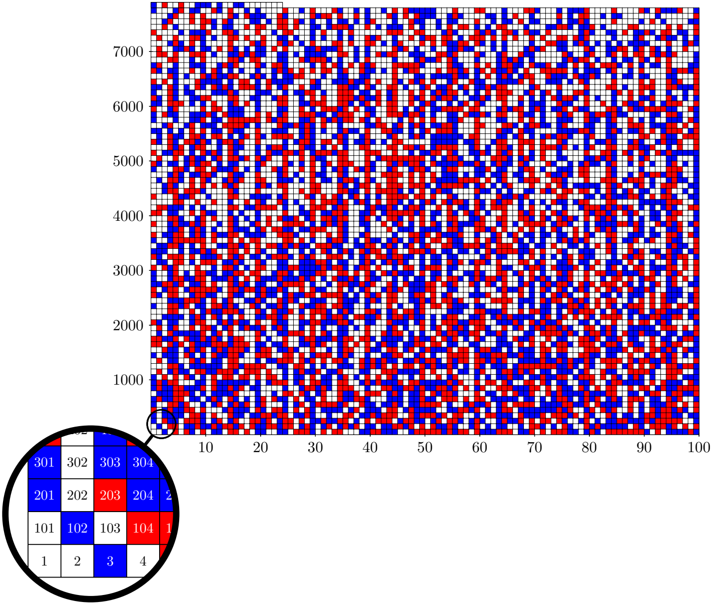
Myers conjectures that the answer is yes [PhD thesis, 2015].

A bi-coloring of $[1, n]$ is encoded using Boolean variables x_i with $i \in \{1, 2, \dots, n\}$ such that $x_i = 1$ ($= 0$) means that i is colored red (blue). For each Pythagorean triple $a^2 + b^2 = c^2$ two clauses are added: $(x_a \vee x_b \vee x_c) \wedge (\bar{x}_a \vee \bar{x}_b \vee \bar{x}_c)$.

Theorem (Main result via parallel SAT solving + proof logging)

$[1, 7824]$ *can be bi-colored s.t. there is no monochromatic Pythagorean triple. This is impossible for* $[1, 7825]$.

An Extreme Solution (a valid partition of $[1, 7824]$) I



Main Contribution

We present a framework that combines, for the first time, all pieces to realize linear speedups and produce verifiable SAT results for very hard problems.

The status quo of using combinatorial solvers and years of computation is arguably intolerable for mathematicians:

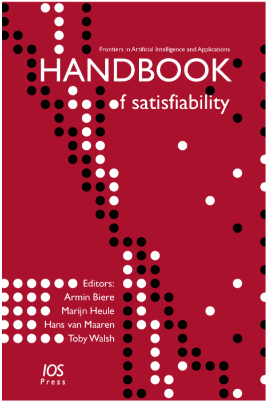
- Kouril and Paul [2008] computed the sixth van der Waerden number — $vdW(6,6) = 1132$ — using dedicated hardware without producing a proof.
- McKay's and Radziszowski's big result [1995] in Ramsey Theory — $R(4,5) = 25$ — still cannot be reproduced with methods that can be validated.

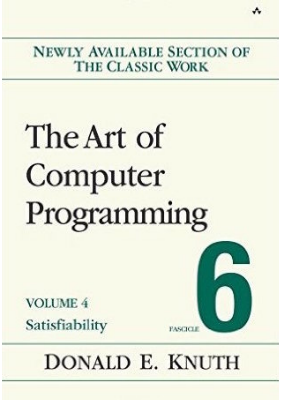
We demonstrate our framework on the Pythagorean triples problem, potentially the hardest problem solved with SAT yet.

The Art of SAT Solving

Enormous Progress in the Last Two Decades

Formulas with 100's of variables and 1,000's of clauses were solvable in the mid-nineties to formulas with 100,000's of variables to 1,000,000's of clauses now.





The Boolean Schur Triples Problem F_9

Can the set $\{1, \ldots, n\}$ be red/blue colored such that there is no monochromatic solution of $a + b = c$ with $a < b < c$? Below the encoding of this problem with $n = 9$ (formula F_9):

$$
\begin{aligned}
&(x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (x_1 \vee x_3 \vee x_4) \wedge (\bar{x}_1 \vee \bar{x}_3 \vee \bar{x}_4) \wedge \\
&(x_1 \vee x_4 \vee x_5) \wedge (\bar{x}_1 \vee \bar{x}_4 \vee \bar{x}_5) \wedge (x_2 \vee x_3 \vee x_5) \wedge (\bar{x}_2 \vee \bar{x}_3 \vee \bar{x}_5) \wedge \\
&(x_1 \vee x_5 \vee x_6) \wedge (\bar{x}_1 \vee \bar{x}_5 \vee \bar{x}_6) \wedge (x_2 \vee x_4 \vee x_6) \wedge (\bar{x}_2 \vee \bar{x}_4 \vee \bar{x}_6) \wedge \\
&(x_1 \vee x_6 \vee x_7) \wedge (\bar{x}_1 \vee \bar{x}_6 \vee \bar{x}_7) \wedge (x_2 \vee x_5 \vee x_7) \wedge (\bar{x}_2 \vee \bar{x}_5 \vee \bar{x}_7) \wedge \\
&(x_3 \vee x_4 \vee x_7) \wedge (\bar{x}_3 \vee \bar{x}_4 \vee \bar{x}_7) \wedge (x_1 \vee x_7 \vee x_8) \wedge (\bar{x}_1 \vee \bar{x}_7 \vee \bar{x}_8) \wedge \\
&(x_2 \vee x_6 \vee x_8) \wedge (\bar{x}_2 \vee \bar{x}_6 \vee \bar{x}_8) \wedge (x_3 \vee x_5 \vee x_8) \wedge (\bar{x}_3 \vee \bar{x}_5 \vee \bar{x}_8) \wedge \\
&(x_1 \vee x_8 \vee x_9) \wedge (\bar{x}_1 \vee \bar{x}_8 \vee \bar{x}_9) \wedge (x_2 \vee x_7 \vee x_9) \wedge (\bar{x}_2 \vee \bar{x}_7 \vee \bar{x}_9) \wedge \\
&(x_3 \vee x_6 \vee x_9) \wedge (\bar{x}_3 \vee \bar{x}_6 \vee \bar{x}_9) \wedge (x_4 \vee x_5 \vee x_9) \wedge (\bar{x}_4 \vee \bar{x}_5 \vee \bar{x}_9)
\end{aligned}
$$

Is this formula satisfiable?

Unit Clause Propagation and Search Tree

Unit Clause Propagation (UCP or $\vdash_1$) assigns unit clauses —all literals, but one are assigned to false— till fixpoint or conflict.

Example

Consider the following clauses occurring in F_9:

$(x_1 \vee \cancel{x_2} \vee \cancel{x_3}), (\cancel{x_2} \vee \cancel{x_3} \vee x_5), (\cancel{\bar{x}_1} \vee \bar{x}_4 \vee \cancel{\bar{x}_5}), (\cancel{x_2} \vee \cancel{x_4} \vee x_6), (\cancel{\bar{x}_1} \vee \cancel{\bar{x}_5} \vee \cancel{\bar{x}_6})$

$F_9 \wedge (\bar{x}_2) \wedge (\bar{x}_3) \vdash_1 (x_1), (x_5), (\bar{x}_4), (x_6), \perp$

Unit Clause Propagation and Search Tree

Unit Clause Propagation (UCP or $\vdash_1$) assigns unit clauses —all literals, but one are assigned to false— till fixpoint or conflict.

Example

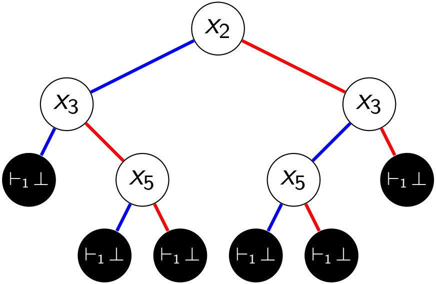
Consider the following clauses occurring in F_9:

$$(x_1 \vee \cancel{x_2} \vee \cancel{x_3}), (\cancel{x_2} \vee \cancel{x_3} \vee x_5), (\cancel{\bar{x}_1} \vee \bar{x}_4 \vee \cancel{\bar{x}_5}), (\cancel{x_2} \vee \cancel{x_4} \vee x_6), (\cancel{\bar{x}_1} \vee \cancel{\bar{x}_5} \vee \cancel{\bar{x}_6})$$

$$F_9 \wedge (\bar{x}_2) \wedge (\bar{x}_3) \vdash_1 (x_1), (x_5), (\bar{x}_4), (x_6), \bot$$

A binary search-tree with only six leaves is enough to refute F_9 with UCP...

..., but this requires good variable selection heuristics for the decision variables (the internal nodes).



SAT Solver Paradigms

Local search: Given a full assignment φ for a formula F, flip the truth value of variables until the φ satisfies F.

Strength: Can quickly find solutions, even for hard formulas.

Weakness: Cannot prove unsatisfiability.

SAT Solver Paradigms

Local search: Given a full assignment φ for a formula F, flip the truth value of variables until the φ satisfies F.

Strength: Can quickly find solutions, even for hard formulas.

Weakness: Cannot prove unsatisfiability.

Look-ahead: Aims at finding a small binary search-tree by selection effective splitting variables by looking ahead.

Strength: Produces small search-trees, even for hard formulas.

Weakness: Expensive and can only find tree-based refutations.

SAT Solver Paradigms

Local search: Given a full assignment φ for a formula F, flip the truth value of variables until the φ satisfies F.

Strength: Can quickly find solutions, even for hard formulas.

Weakness: Cannot prove unsatisfiability.

Look-ahead: Aims at finding a small binary search-tree by selection effective splitting variables by looking ahead.

Strength: Produces small search-trees, even for hard formulas.

Weakness: Expensive and can only find tree-based refutations.

Conflict-driven clause learning (CDCL): Assigns variables until a clause gets falsified, turns this conflict into a learned clause, which is added to the formula, and restarts with new heuristics.

Strength: Can quickly find refutations, even for huge formulas.

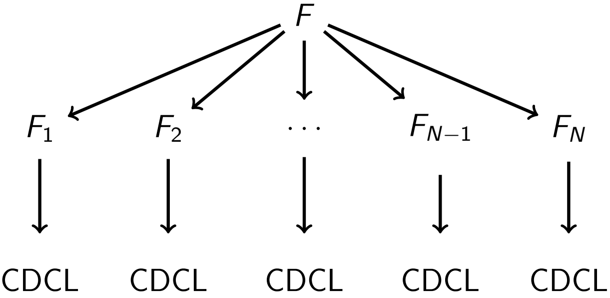
Weakness: Point of competence and hard to parallelize.

Cube-and-Conquer [Heule, Kullmann, Wieringa, and Biere 2011]

The Cube-and-Conquer paradigm has two phases:

Cube First a look-ahead solver is employed to split the problem — the splitting tree is cut off appropriately.

Conquer At the leaves of the tree, CDCL solvers are employed.



Cube-and-Conquer achieves a good equal splitting and the sub-problems are scheduled independently (easy parallel CDCL).

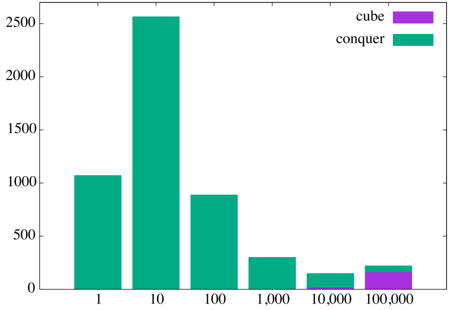
The Hidden Strength of Cube-and-Conquer

Let N denote the number of leaves in the cube-phase:

- the case $N = 1$ means pure CDCL,
- and very large N means pure look-ahead.

Consider the total run-time (y-axis) in dependency on N (x-axis):

- typically, first it increases, then
- it decreases, but only for a large number of subproblems!



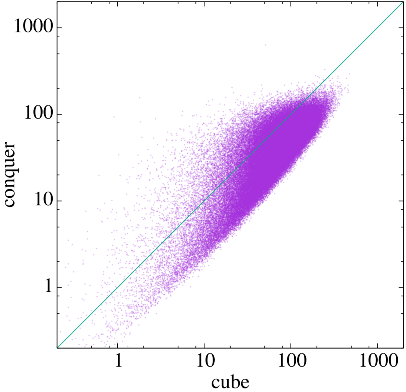
Example with Schur Triples and 5 colors: a formula with 708 vars and 22608 clauses.

Subproblems are solved with Glucose 3.0 as conquer solver.

The performance tends to be optimal when the cube and conquer times are comparable.

Pythagorean Triples Results Summary

- After splitting —into a million subproblems— there were no hard subproblems: each could be solved within 1000 s;
- We used 800 cores on the TACC Stampede cluster;
- The total computation was about 4 CPU years, but less than 2 days in wallclock time;
- If we use all 110 000 cores, then the problem can be solved in less than an hour;
- Almost linear speed-ups even when using 1000's of cores;



- Reduced the trivial 2^{7825} to roughly 2^{40}.

Producing and Verifying the Largest Math Proof Ever

Motivation for validating unsatisfiability proofs

Satisfiability solvers are used in amazing ways...

- Hardware and software verification (Intel and Microsoft)
- Hard-Combinatorial problems:
  - van der Waerden numbers [Dransfield, Marek, and Truszczynski, 2004; Kouril and Paul, 2008]
  - Gardens of Eden in Conway's Game of Life [Hartman, Heule, Kwekkeboom, and Noels, 2013]
  - Erdős Discrepancy Problem [Konev and Lisitsa, 2014]

..., but SAT solvers may have errors and only return yes/no.

- Documented bugs in SAT, SMT, and QBF solvers [Brummayer and Biere, 2009; Brummayer et al., 2010]
- Implementation errors often imply conceptual errors
- Proofs now mandatory for the annual SAT Competitions.
- Mathematical results require a stronger justification than a simple yes/no by a solver. UNSAT must be checkable.

Proofs and Refutations

A clause C is solutions-preserving with respect to a formula F if all solutions of F satisfy C (denoted by $\equiv$).

A proof trace is a sequence of solutions-preserving clauses. Solutions-preserving should be checkable in polynomial time.

Proofs and Refutations

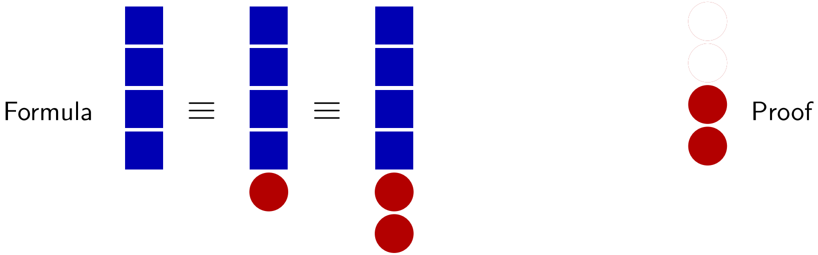
A clause C is solutions-preserving with respect to a formula F if all solutions of F satisfy C (denoted by $\equiv$).

A proof trace is a sequence of solutions-preserving clauses. Solutions-preserving should be checkable in polynomial time.

Proofs and Refutations

A clause C is solutions-preserving with respect to a formula F if all solutions of F satisfy C (denoted by $\equiv$).

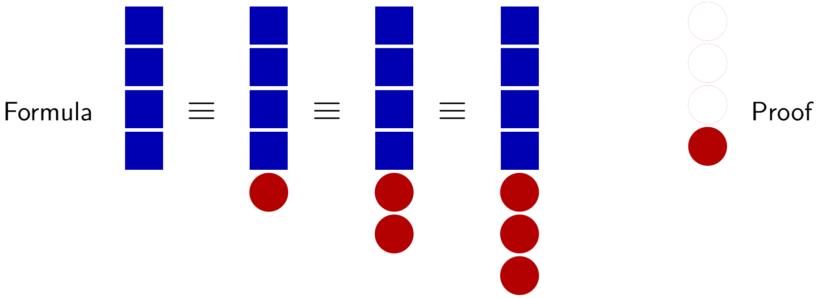
A proof trace is a sequence of solutions-preserving clauses. Solutions-preserving should be checkable in polynomial time.



Proofs and Refutations

A clause C is solutions-preserving with respect to a formula F if all solutions of F satisfy C (denoted by $\equiv$).

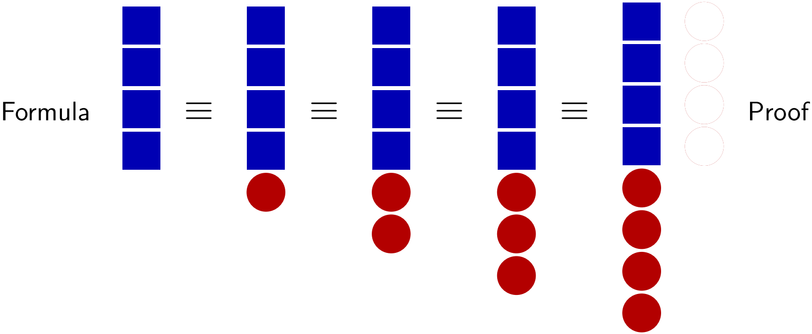
A proof trace is a sequence of solutions-preserving clauses. Solutions-preserving should be checkable in polynomial time.



Proofs and Refutations

A clause C is solutions-preserving with respect to a formula F if all solutions of F satisfy C (denoted by $\equiv$).

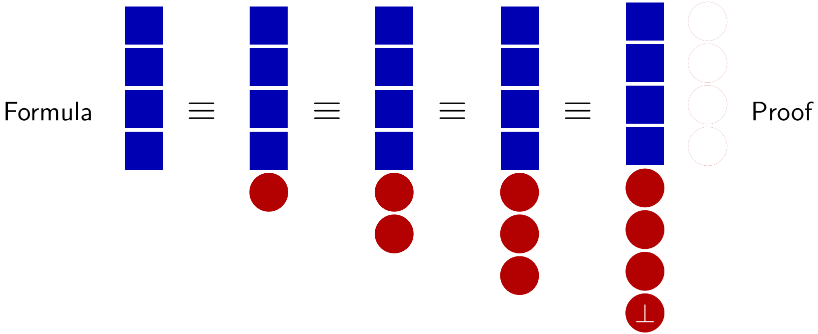
A proof trace is a sequence of solutions-preserving clauses. Solutions-preserving should be checkable in polynomial time.



Proofs and Refutations

A clause C is solutions-preserving with respect to a formula F if all solutions of F satisfy C (denoted by $\equiv$).

A proof trace is a sequence of solutions-preserving clauses. Solutions-preserving should be checkable in polynomial time.



A refutation is a proof trace containing the empty clause, $\perp$.

Solutions-Preserving Modulo x

Let φ be an assignment and x a literal. We denote with $\varphi \otimes x$ a copy of φ in which the assignment to x is flipped. If φ does not assign x, then $\varphi \otimes x$ assigns x to `true`.

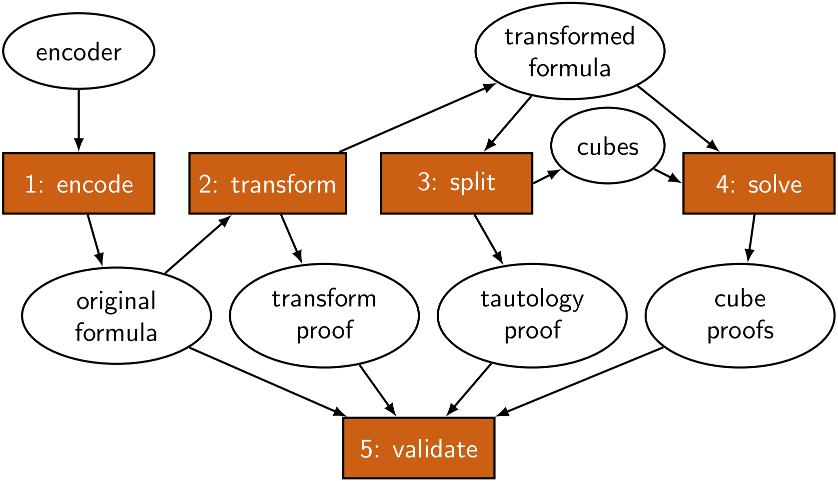
A clause C is solutions-preserving modulo x (SPMx) with respect to a formula F if and only if for every solution φ of F, φ or $\varphi \otimes x$ satisfies F *and* C.

Example

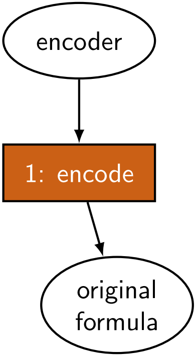
Consider the formula $F = (x \vee y) \wedge (x \vee \bar{y})$. The clause $(\bar{x} \vee y)$ is solutions-preserving modulo y with respect to F. F has two solutions $\varphi_1 := \{x = 1, y = 1\}$ and $\varphi_2 := \{x = 1, y = 0\}$. φ_1 satisfies C (and F) and $\varphi_2 \otimes y$ satisfies F and C.

All techniques in state-of-the-art SAT solvers can be expressed using SPMx steps [Järvisalo, Heule, and Biere 2012].

Overview of Solving Framework



Phase 1: Encode



Input:	encoder program
Output:	the "original" CNF formula
Goal:	make the translation to SAT as simple as possible

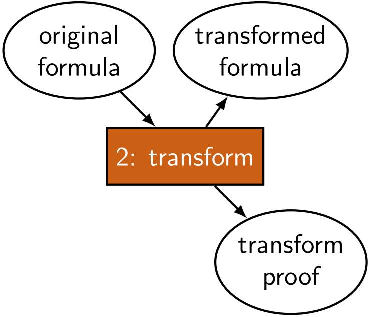
```
for (int a = 1; a <= n; a++)
  for (int b = a; b <= n; b++) {
     int c = sqrt (a*a + b*b);
     if ((c <= n) && ((a*a + b*b) == (c*c))) {
       addClause ( a,  b,  c);
       addClause (-a, -b, -c); } }
```

F_{7824} has 6492 (occurring) variables and 18930 clauses, and F_{7825} has 6494 (occurring) variables and 18944 clauses.

Notice $F_{7825} = F_{7824} + 14$ clauses. These 14 make it UNSAT.

Phase 2: Transform

Input:	original CNF formula
Output:	transformed formula and transformation proof
Goal:	optimize the formula for the later (solving) phases



We applied two transformations (via SPMx):

- Pythagorean Triple Elimination removes Pythagorean Triples that contain an element that does not occur in any other Pythagorean Triple, e.g. $3^2 + 4^2 = 5^2$ (till fixpoint).
- Symmetry breaking colors the number most frequently occurring in Pythagorean triples (2520) red.

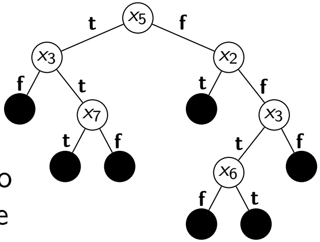
All transformation (pre-processing) techniques can be expressed using SPMx steps [Järvisalo, Heule, and Biere 2012].

Phase 3: Split

Input: transformed formula

Output: cubes and tautology proof

Goal: partition the given formula to minimize total wallclock time

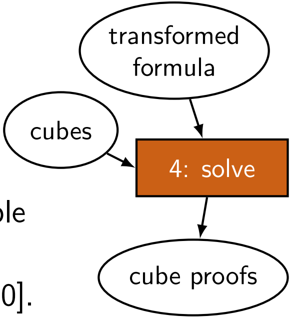


Two layers of splitting F_{7824}:

- The top level split partitions the transformed formula into exactly a million subproblems;
- Each subproblem is partitioned into tens of thousands of subsubproblems. Total time: 25,000 CPU hours

$$
\begin{aligned}
D = \; & (x_5 \wedge \bar{x}_3) \;\vee \\
& (x_5 \wedge x_3 \wedge x_7) \;\vee \\
& (x_5 \wedge x_3 \wedge \bar{x}_7) \;\vee \\
& (\bar{x}_5 \wedge x_2) \;\vee \\
& (\bar{x}_5 \wedge \bar{x}_2 \wedge x_3 \wedge \bar{x}_6) \;\vee \\
& (\bar{x}_5 \wedge \bar{x}_2 \wedge x_3 \wedge x_6) \;\vee \\
& (\bar{x}_5 \wedge \bar{x}_2 \wedge \bar{x}_3)
\end{aligned}
$$

Phase 4: Solve



Input: transformed formula and cubes

Output: cube proofs (or a solution)

Goal: solve —with proof logging— all subproblems as fast as possible

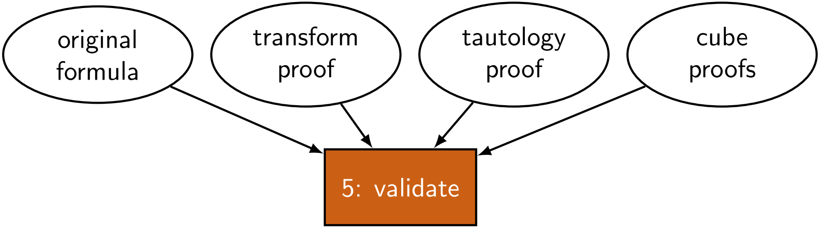
Let φ_i be the i^{th} cube with $i \in [1, 1\,000\,000]$.

We first solved all $F_{7824} \wedge \varphi_i$, total runtime was 13,000 CPU hours or, just a wall-clock day). One subproblem is satisfiable.

The backbone of a formula is the set of literals that are assigned to true in all solutions. The backbone of F_{7824} after symmetry breaking (2520) consists of 2304 literals, including

- x_{5180} and x_{5865}, while $5180^2 + 5865^2 = 7825^2 \rightarrow 7825$
- $\bar{x}_{625}$ and $\bar{x}_{7800}$, while $625^2 + 7800^2 = 7825^2 \rightarrow 7825$

Phase 5: Validate Pythagorean Triples Proofs

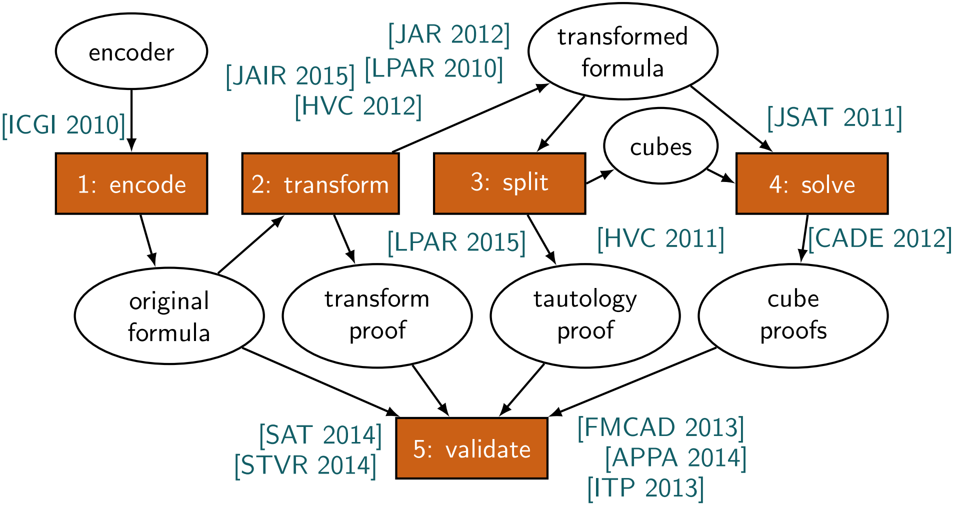


We check the proofs with the DRAT-trim checker, which has been used to validate the UNSAT results of the international SAT Competitions since 2013.

Recently it was shown how to validate DRAT proofs in parallel [Heule and Biere 2015].

The size of the merged proof is almost 200 terabyte and has been validated in 16,000 CPU hours.

Overview of Solving Framework: Contributions



Joint work with: Armin Biere, Warren Hunt, Matti Järvisalo, Oliver Kullmann, Florian Lonsing, Victor Marek, Martina Seidl, Antonio Ramos, Peter van der Tak, Sicco Verwer, Nathan Wetzler and Siert Wieringa.

Media, Meaning, and Truth

Media: The Largest Math Proof Ever

Mathematics versus Computer Science

A typical argument, as articulated in the Nature 543, pp 17–18:

> *If mathematicians' work is understood to be a quest to increase human understanding of mathematics, rather than to accumulate an ever-larger collection of facts, a solution that rests on theory seems superior to a computer ticking off possibilities.*

Widespread missing understanding of computer science:

- Computers do not simply "tick off possibilities";
- The "possibilities" are non-trivial, and simple algorithms might take forever;
- The complexity issues touched here might be far more interesting/relevant than the concrete result in Ramsey theory.

Perhaps meaningless is the true meaning?

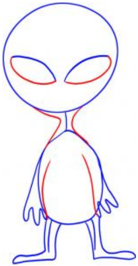
Facts may be meaningless, but...

- The "computer ticking off possibilities" is actually quite a sophisticated thing here, and is absolutely crucial for the analysis for example of the correctness of microprocessors.
- For some not yet understood reasons it seems that these benchmarks from the field of Ramsey theory are relevant for the perhaps most fundamental question in computer science: what makes a problem hard (P vs NP)?

Perhaps it is precisely that the fact 7825 has no meaning, which makes these computational problems meaningful – the bugs in the designs of complicated artificial systems also have no meaning!

Alien Truths

Let's call alien a true statement (best rather short) with only a very long proof.



- Already the question, whether we can show something (like our case) to be alien, is of highest relevance. There may be a short proof for the Pythagorean Triples problem, but probably not for exact bound of 7825.
- But independently, such "alien truths" or "alien questions" arise in formal contexts, where large propositional formulas come out from engineering systems, which in its complexity, especially what concerns "small" bugs, is perhaps beyond "understanding". Mathematicians dislike "nitty-gritty details", but prefer "the big picture" (handwaving).

Human and Alien Truth Hierarchy

Human	Classical math proofs, e.g. Schur's Theorem.
Weakly Human	Proofs with a large human component and some computer effort, e.g. Four Color Theorem.
Weakly Alien	A giant humanly generated case-split, e.g. minimum number of givens is 17 in Sudoku.
Alien	A giant case-split that mysteriously avoids an enormous exponential effort, e.g. the sixth van der Waerden number, vdW(6,6), is 1132.
Strongly Alien	An alien truth regarding a high-level statement, e.g. any two-coloring of the natural numbers yields a monochromatic Pythagorean triple.

The traditional interest is to search for a short proof. But perhaps the question, why there isn't one, or what makes the problem hard, is the real question here?

Conclusions and Future Work

Conclusions

Theorem (Main result)

$[1, 7824]$ *can be bi-colored s.t. there is no monochromatic Pythagorean triple. This is impossible for* $[1, 7825]$.

We solved and verified the theorem via SAT solving:

- Cube-and-conquer facilitated massive parallel solving.
- A new heuristic was developed to substantially reduce the search space. Moreover the heuristic facilitated almost linear speed-ups while using 800 cores.
- The proof is huge (200 terabyte), but can be compressed to 68 gigabyte (13,000 CPU hours to decompress) and be validated in 16,000 CPU hours.

Future Directions

Apply our solving framework to other challenges in Ramsey Theory and elsewhere:

- Century-old open problems appear solvable now. Very recent result: Schur number 5 is 160.
- Produce proofs for existing results without a proof, for example vdW(6,6) = 1132 [Kouril and Paul 2008].

Evaluate the influence of the interval $[1, n]$ with $n \geq 7825$ on the size of the proof of the Pythagorean triples problem.

Look-ahead heuristics are crucial and we had to develop dedicated heuristics to solve the Pythagorean triples problem.

- Develop powerful heuristics that work out of the box.
- Alternatively, add heuristic-tuning techniques to the tool chain [Hoos 2012].

Develop a mechanically-verified, fast clausal proof checker.

Everything's Bigger in Texas
The Largest Math Proof Ever

Solving and Verifying the Boolean Pythagorean Triples problem via Cube-and-Conquer

Marijn J.H. Heule

THE UNIVERSITY OF TEXAS AT AUSTIN

Joint work with Oliver Kullmann and Victor W. Marek

Rice University August 24, 2016