Applications of SAT solving to Mathematics: Proofs and Heuristics

Marijn J.H. Heule and Oliver Kullmann

THE UNIVERSITY OF TEXAS AT AUSTIN

Swansea University
Prifysgol Abertawe

Fields Institute August 18, 2016

Pythagorean Triples Problem

Proofs of Unsatisfiability

Producing a Proof

Verifying a Proof

Pythagorean Triples Problem

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 finite counter-example?

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 finite counter-example?

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

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

decide	$1+2=3$ $2+3=5$	$1+4=5$ $1+5=6$	$2+4=6$

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 finite counter-example?

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

2 3 → 1 2 3 5 → 1 2 3 4 5 6 → ×

decide	$1+2=3$ $2+3=5$	$1+4=5$ $1+5=6$	$2+4=6$

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 finite counter-example?

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

Pythagorean Triples Problem [Graham]

Can the set of natural numbers $\mathbb{N} = \{1, 2, 3, \ldots\}$ 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 finite counter-example?

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

A bi-coloring of $[1, n]$ is encoded using Boolean variables x_i with $i \in \{1, 2, \ldots, n\}$ 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 $\mathbb{N} = \{1, 2, 3, \ldots\}$ 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 finite counter-example?

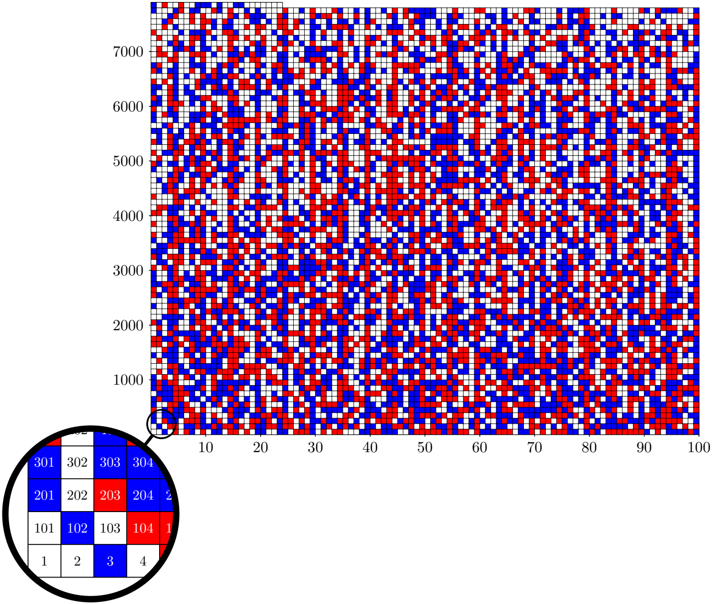
Best lower bound: a bi-coloring of $[1, 7664]$ s.t. there is no monochromatic Pythagorean triple [Cooper & Overstreet 2015].

A bi-coloring of $[1, n]$ is encoded using Boolean variables x_i with $i \in \{1, 2, \ldots, n\}$ 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])



Main Contribution

We present a framework that combines, for the first time, all pieces to 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 ($W(2,6) = 1132$) 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.

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

Proofs of Unsatisfiability

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):

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

Is this formula satisfiable?

Why NOT to use Resolution Proofs (1)

Resolution:

$(y_1 \vee \ldots \vee y_i \vee z_1 \vee \ldots \vee z_j) := (x \vee y_1 \vee \ldots \vee y_i) \diamond_x (\bar{x} \vee z_1 \vee \ldots \vee z_j)$

Most clause addition steps in SAT solvers can be expressed as a sequence of resolution steps, but

- The average sequence length is 400, making proofs huge;
- Memory consumption can explode, up to a factor of 100;
- Computing the order matters and is costly:

$$(x_2 \vee x_3) := (\bar{x}_1 \vee \bar{x}_5 \vee \bar{x}_6) \diamond_{x_6} (x_2 \vee x_4 \vee x_6) \diamond_{x_4} (\bar{x}_1 \vee \bar{x}_4 \vee \bar{x}_5) \diamond_{x_5} (x_2 \vee x_3 \vee x_5) \diamond_{x_1} (x_1 \vee x_2 \vee x_3)$$

$$(\bar{x}_1 \vee x_2 \vee x_3 \vee \bar{x}_5) := (x_2 \vee x_4 \vee x_6) \diamond_{x_4} (\bar{x}_1 \vee \bar{x}_4 \vee \bar{x}_5) \diamond_{x_5} (x_2 \vee x_3 \vee x_5) \diamond_{x_1} (x_1 \vee x_2 \vee x_3) \diamond_{x_6} (\bar{x}_1 \vee \bar{x}_5 \vee \bar{x}_6)$$

Why NOT to use Resolution Proofs (2)

Some powerful techniques used in SAT solvers cannot be expressed by resolutions, because they add and remove solutions.

Examples: symmetry breaking and bounded variable addition

Symmetry-breaking techniques can significantly reduce the solving time, but they remove solutions.

Example (Bounded variable addition)

Replace

$$(a \vee d) \quad (a \vee e)$$
$$(b \vee d) \quad (b \vee e)$$
$$(c \vee d) \quad (c \vee e)$$

by

$$(\bar{x} \vee a) \quad (\bar{x} \vee b) \quad (\bar{x} \vee c)$$
$$(x \vee d) \quad (x \vee e)$$

Unit Clause Propagation to the Rescue

A clause C is solutions-preserving with respect to a formula F if and only if for every solution φ of F satisfies C.

Or, equivalently, $C = (y_1 \vee \cdots \vee y_k)$ is solutions-preserving w.r.t. a formula F if and only if $F \wedge (\bar{y}_1) \wedge \cdots \wedge (\bar{y}_k) \models \epsilon$.

For an unsatisfiable formula all clauses are solutions-preserving, but how to check solutions-preserving in polynomial time?

Unit Clause Propagation to the Rescue

A clause C is solutions-preserving with respect to a formula F if and only if for every solution φ of F satisfies C.

Or, equivalently, $C = (y_1 \vee \cdots \vee y_k)$ is solutions-preserving w.r.t. a formula F if and only if $F \wedge (\bar{y}_1) \wedge \cdots \wedge (\bar{y}_k) \models \epsilon$.

For an unsatisfiable formula all clauses are solutions-preserving, but how to check solutions-preserving in polynomial time?

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

Example

$F_9 \wedge (\bar{x}_2) \wedge (\bar{x}_3) \vdash_1 \epsilon$

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

Unit Clause Propagation to the Rescue

A clause C is solutions-preserving with respect to a formula F if and only if for every solution φ of F satisfies C.

Or, equivalently, $C = (y_1 \vee \cdots \vee y_k)$ is solutions-preserving w.r.t. a formula F if and only if $F \wedge (\bar{y}_1) \wedge \cdots \wedge (\bar{y}_k) \models \epsilon$.

For an unsatisfiable formula all clauses are solutions-preserving, but how to check solutions-preserving in polynomial time?

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

Example

$F_9 \wedge (\bar{x}_2) \wedge (\bar{x}_3) \vdash_1 \epsilon$

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

Proposition

If $F \wedge (\bar{y}_1) \wedge \cdots \wedge (\bar{y}_k) \vdash_1 \epsilon$, then $F \models (y_1 \vee \cdots \vee y_k)$.

An $SP_{\vdash_1}$ Proof for our Schur Triples Problem

A clause $C = (y_1 \vee \cdots \vee y_k)$ has property $SP_{\vdash_1}$ (aka RUP) with respect to formula F if and only if $F \wedge (\bar{y}_1) \wedge \cdots \wedge (\bar{y}_k) \vdash_1 \epsilon$.

An $SP_{\vdash_1}$ proof of a formula F is a sequence of clauses $C_1, \ldots, \epsilon$ s.t. C_i has property $SP_{\vdash_1}$ with respect to $F \wedge C_1 \wedge \cdots \wedge C_{i-1}$.

#	clause	required $SP_{\vdash_1}$ check
1	$(x_2 \vee x_3)$	$F_9 \wedge (\bar{x}_2) \wedge (\bar{x}_3) \vdash_1 \epsilon$
2	$(x_2 \vee x_5)$	$F_9 \wedge (x_2 \vee x_3) \wedge (\bar{x}_2) \wedge (\bar{x}_5) \vdash_1 \epsilon$
3	(x_2)	$F_9 \wedge (x_2 \vee x_3) \wedge (x_2 \vee x_5) \wedge (\bar{x}_2) \vdash_1 \epsilon$
4	$(\bar{x}_3)$	$F_9 \wedge (x_2 \vee x_3) \wedge (x_2 \vee x_5) \wedge (x_2) \wedge (x_3) \vdash_1 \epsilon$
5	$(\bar{x}_5)$	$F_9 \wedge (x_2 \vee x_3) \wedge (x_2 \vee x_5) \wedge (x_2) \wedge (\bar{x}_3) \wedge (x_5) \vdash_1 \epsilon$
6	ϵ	$F_9 \wedge (x_2 \vee x_3) \wedge (x_2 \vee x_5) \wedge (x_2) \wedge (\bar{x}_3) \wedge (\bar{x}_5) \vdash_1 \epsilon$

$SP_{\vdash_1}$ proofs are much more compact than resolution proofs

An $\text{SP}_{\vdash_1}$ Proof with Deletion Information

Solvers realize fast performance by deleting clauses aggressively.

In order to check $\text{SP}_{\vdash_1}$ efficiently, the clause deletion information has to be included in the proof.

#	clause	required $\text{SP}_{\vdash_1}$ check
1	$(x_2 \vee x_3)$	$F_9 \wedge (\bar{x}_2) \wedge (\bar{x}_3) \vdash_1 \epsilon$
2	$(x_2 \vee x_5)$	$F_9 \wedge (x_2 \vee x_3) \wedge (\bar{x}_2) \wedge (\bar{x}_5) \vdash_1 \epsilon$
3	(x_2)	$F_9 \wedge (x_2 \vee x_3) \wedge (x_2 \vee x_5) \wedge (\bar{x}_2) \vdash_1 \epsilon$
d	$(x_2 \vee x_3)$	
d	$(x_2 \vee x_5)$	
4	$(\bar{x}_3)$	$F_9 \wedge (x_2) \wedge (x_3) \vdash_1 \epsilon$
5	$(\bar{x}_5)$	$F_9 \wedge (x_2) \wedge (\bar{x}_3) \wedge (x_5) \vdash_1 \epsilon$
6	ϵ	$F_9 \wedge (x_2) \wedge (\bar{x}_3) \wedge (\bar{x}_5) \vdash_1 \epsilon$

Efficient validation also requires a dedicated UCP algorithm.

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 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.

Now, $C = (x \vee y_1 \vee \cdots \vee y_k)$ is solutions-preserving modulo x w.r.t. a formula F if and only if for every $(\bar{x} \vee z_1 \vee \cdots \vee z_m)$ holds that $F \wedge (\bar{x}) \wedge (\bar{y}_1) \wedge \cdots \wedge (\bar{y}_k) \wedge (\bar{z}_1) \wedge \cdots \wedge (\bar{z}_m) \models \epsilon$.

Checking Solutions-Preserving Module x via UCP

Definition (SPMx$_{\vdash_1}$ (RAT [Järvisalo, Heule, and Biere 2012]))

A clause $C = (x \lor y_1 \lor \dots \lor y_k)$ has property SPMx$_{\vdash_1}$ w.r.t. a formula F if and only if for every $(\bar{x} \lor z_1 \lor \dots \lor z_m)$ holds that $F \land (\bar{x}) \land (\bar{y}_1) \land \dots \land (\bar{y}_k) \land (\bar{z}_1) \land \dots \land (\bar{z}_m) \vdash_1 \epsilon$.

All techniques used in SAT solvers, including preprocessing techniques such as symmetry breaking and bounded variable addition can be expressed as addition of SPMx$_{\vdash_1}$ clauses.

Example (Bounded variable addition)

Replace

$(a \lor d)$ $(a \lor e)$
$(b \lor d)$ $(b \lor e)$
$(c \lor d)$ $(c \lor e)$

by

$(\bar{x} \lor a)$ $(\bar{x} \lor b)$ $(\bar{x} \lor c)$
$(x \lor d)$ $(x \lor e)$

Notice that SPMx$_{\vdash_1}$ simulates Extended Resolution

An SPMx Proof

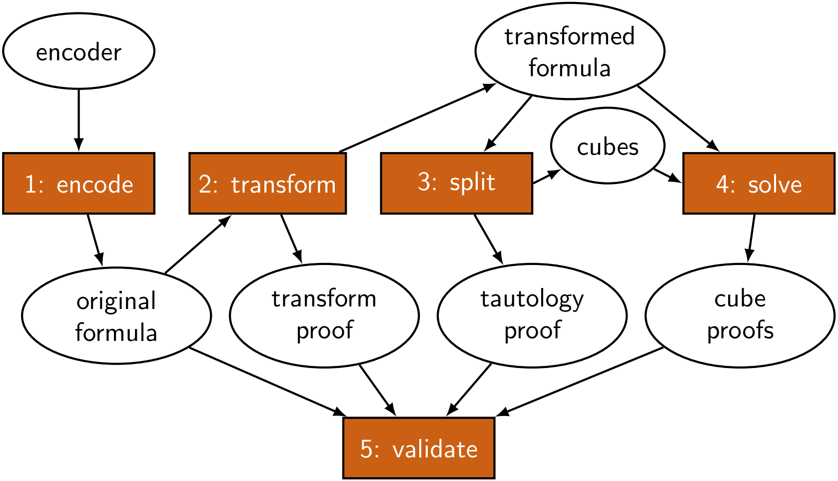
A SPMx$_{\vdash_1}$ proof of a formula F is a sequence of clauses $C_1, \ldots, \epsilon$ s.t. C_i has property SPMx$_{\vdash_1}$ with respect to $F \wedge C_1 \wedge \cdots \wedge C_{i-1}$.

SPMx$_{\vdash_1}$ proofs can be exponentially smaller than SP$_{\vdash_1}$ proofs!

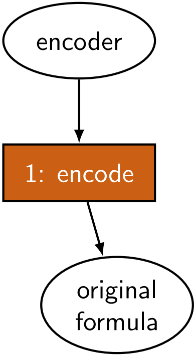
#	clause	required SPMx$_{\vdash_1}$ check
1	$(x_1 \vee x_4)$	$F_9 \wedge (\bar{x}_1) \wedge (\bar{x}_4) \vdash_1 \epsilon$
		$F_9 \wedge (x_1 \vee x_4) \wedge (\bar{x}_1) \wedge (x_2) \wedge (x_3) \vdash_1 \epsilon$
		$F_9 \wedge (x_1 \vee x_4) \wedge (\bar{x}_1) \wedge (x_3) \wedge (x_4) \vdash_1 \epsilon$
		$F_9 \wedge (x_1 \vee x_4) \wedge (\bar{x}_1) \wedge (x_4) \wedge (x_5) \vdash_1 \epsilon$
2	(x_1)	$F_9 \wedge (x_1 \vee x_4) \wedge (\bar{x}_1) \wedge (x_5) \wedge (x_6) \vdash_1 \epsilon$
		$F_9 \wedge (x_1 \vee x_4) \wedge (\bar{x}_1) \wedge (x_6) \wedge (x_7) \vdash_1 \epsilon$
		$F_9 \wedge (x_1 \vee x_4) \wedge (\bar{x}_1) \wedge (x_7) \wedge (x_8) \vdash_1 \epsilon$
		$F_9 \wedge (x_1 \vee x_4) \wedge (\bar{x}_1) \wedge (x_8) \wedge (x_9) \vdash_1 \epsilon$
3	(x_4)	... similar as the (x_1) SPMx$_{\vdash_1}$ check ...
4	ϵ	$F_9 \wedge (x_1 \vee x_4) \wedge (x_1) \wedge (x_4) \vdash_1 \epsilon$

Producing a Proof

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.

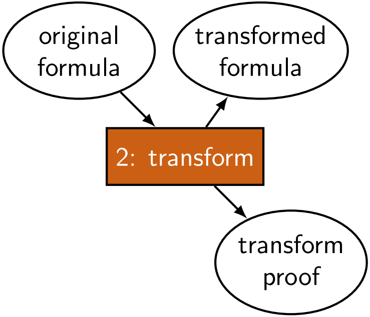
Notice $F_{7825} = F_{7824} + 14$ clauses. These 14 make it UNSAT.

Phase 2: Transform

Input: original CNF formula

Output: transformed formula and transformation proof

Goal: optimize the formula for the later (solving) phases

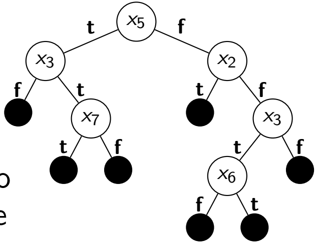


We applied two transformations (via SPMx):

- 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

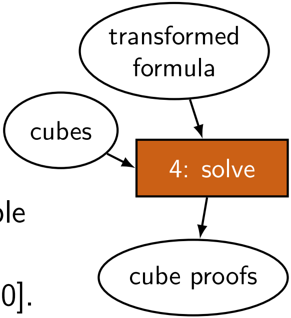
Goal: partition the given formula to minimize total wallclock time

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

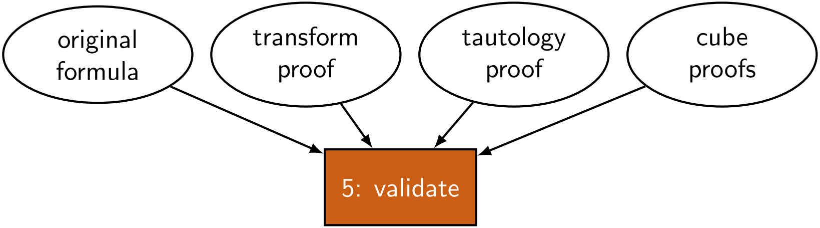
Let φ_i be the i^{th} cube with $i \in [1, 1\,000\,000]$.

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

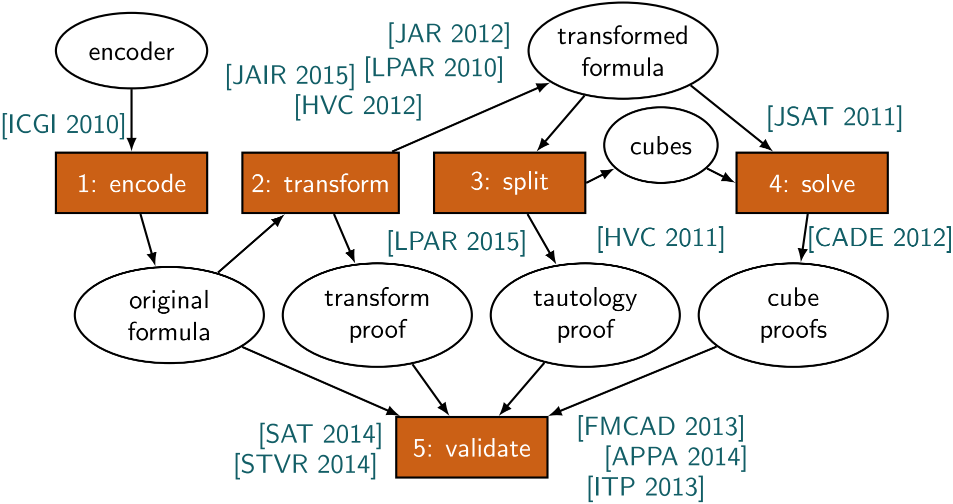


We check the proofs with the DRAT-trim checker, which has been used to validate the UNSAT results of the international SAT Competitions since 2013.

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.

Verifying a Proof

Base Rules

Given a formula F_i (multi-set), a clause C and a modification $m \in \{a, d\}$, a proof step is denoted as $F_i \xrightarrow{m(C)} F_{i+1}$.

$$\text{ADD:} \quad \frac{}{F \xrightarrow{a(C)} F\,C} \qquad \text{where } C \text{ has SPMx}_{\vdash_1} \text{ w.r.t. } F$$

$$\text{DEL:} \quad \frac{}{F\,C \xrightarrow{d(C)} F} \qquad \text{(no side condition)}$$

DEL has no side condition for refutations (unsatisfiability).
For satisfiability proofs, DEL has the ADD side condition.

Consider the proof and the SEQ rule

$$F_0 \xrightarrow{m_1(C_1)} F_1 \xrightarrow{m_2(C_2)} F_2 \ldots F_{n-1} \xrightarrow{m_n(C_n)} F_n$$

$\triangle = m_1(C_1)m_2(C_2)\cdots m_n(C_n)$ gives a derivation from F_0 to F_n.

Compositional Triples

We represent rules using compositional triples: $(F_{\text{pre}}, \triangle, F_{\text{post}})$.

Triples consists of a pre-CNF F_{pre}, a proof $\triangle$, and a post-CNF F_{post}, denoting that proof $\triangle$ is a derivation from F_{pre} to F_{post}.

Triple $(F_{\text{pre}}, \triangle, F_{\text{post}})$ is valid if and only if $F_{\text{pre}} \xrightarrow{\triangle} F_{\text{post}}$.

Proposition: Given a valid composition triple $(F_{\text{pre}}, \triangle, F_{\text{post}})$, if F_{pre} is satisfiable then F_{post} is satisfiable as well.

The first rule SEQ, short for "sequential", combines two compositional triples for which the post-CNF of one triple equals the pre-CNF of the other triple.

$$\frac{F_0 \xrightarrow{\triangle_1} F_1 \quad F_1 \xrightarrow{\triangle_2} F_2}{F_0 \xrightarrow{\triangle_1 \triangle_2} F_2}$$

Parallel Proof Checking using SEQ Rule

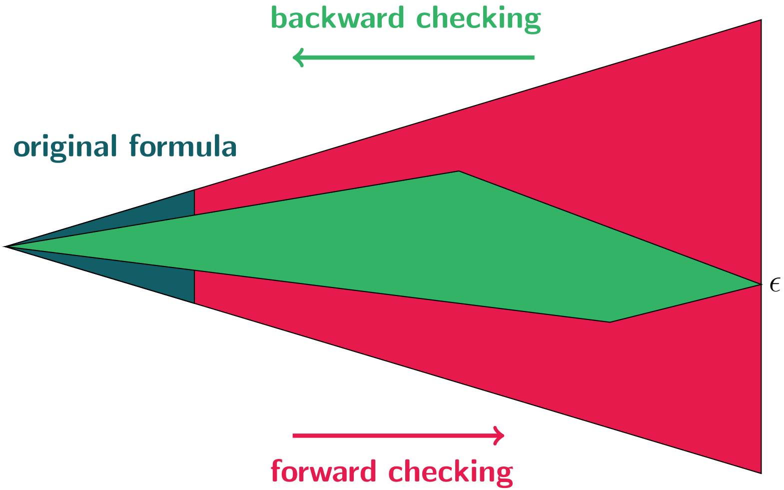
Given a refutation $\triangle$ for formula F_0, apply the following steps:

- partition $\triangle$ into k subproofs $\triangle_1, \ldots, \triangle_k$ (sequential).
  Simply use Unix' `split`.
- compute the post-formulas F_i by applying subproof $\triangle_i$ to formula F_{i-1} (sequential).
- check that all $F_i \xrightarrow{\triangle_{i+1}} F_{i+1}$ are valid derivations for $i \in \{0, \ldots, k-1\}$ with $F_k = \epsilon$ (parallel).

Costs of first two (sequential) steps are relatively small.

$$\frac{F_0 \xrightarrow{\triangle_1} F_1 \quad F_1 \xrightarrow{\triangle_2} F_2 \quad \ldots \quad F_{k-1} \xrightarrow{\triangle_k} \epsilon}{F_0 \xrightarrow{\triangle_1 \triangle_2 \ldots \triangle_k} \epsilon}$$

Forward vs Backward Proof Checking (1)



Forward vs Backward Proof Checking (2)

Forward Checking checks each addition step in a derivation.

Backward Checking initializes by marking the empty clause. Afterwards the proof is checked is reverse order. Only marked clauses are checked, which will mark clauses using conflict analysis. Many addition steps may be skipped (up to 99%).

How to perform backward checking subproofs without ϵ?

- Initialize marking only clauses that are added, but not deleted;
- Unmark clauses that are subsumed by a marked clause;
- Proceed as usual by checking the proof in reverse order.

Backward checking generalization: empty clause subsumes all.

For subproofs: many addition steps can be skipped, although not as many as with refutations.

Conclusions and Future Work

Theorem (Main result)

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

Proof checking for SAT is now mature:

- All techniques is state-of-the-art solvers can be validated.
- Even a clausal proof of 200 terabytes can be verified.
- All UNSAT results of the SAT Competitions are checked.

Next, apply our solving framework to other challenges:

- Existing results for which no proof was produced, for example W(2,6) $=$ 1132 [Kouril and Paul 2008].
- Century-old open problems are solvable: Schur $(5) = 160$.

Applications of SAT solving to Mathematics: Proofs and Heuristics

Marijn J.H. Heule and Oliver Kullmann

THE UNIVERSITY OF TEXAS AT AUSTIN

Swansea University
Prifysgol Abertawe

Fields Institute August 18, 2016