# Solving Very Hard Problems: Cube-and-Conquer, a Hybrid SAT Solving Method

**Marijn J.H. Heule**

THE UNIVERSITY OF TEXAS AT AUSTIN

*Joint work with Armin Biere, Oliver Kullmann, and Victor W. Marek*

Parallel Constraint Reasoning August 6, 2017

# Satisfiability (SAT) Solving Has Many Applications

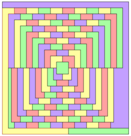

formal verification

security

bioinformatics

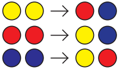
train safety

planning and scheduling



automated theorem proving


exploit generation



term rewriting termination

encode → **SAT solver** → decode

# Satisfiability (SAT) Solving Has Many Applications

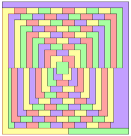

formal verification

security

bioinformatics

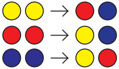
train safety

planning and scheduling



automated theorem proving


exploit generation



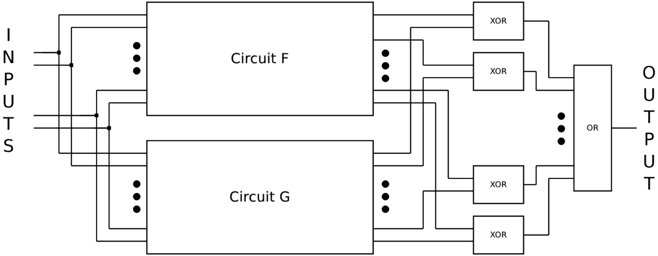
term rewriting termination

encode → **SAT solver** → decode

There are very hard problems in all these application areas!

# Combinatorial Equivalence Checking

Chip makers use SAT to check the correctness of their designs. Equivalence checking involves comparing a specification with an implementation or an optimized with a non-optimized circuit.



# Unavoidable Monochromatic Solutions [Schur 1917]

Will any coloring of the positive integers with red and blue result in a monochromatic solution of $a + b = c$?

| | | |
|---|---|---|
| $1 + 1 = 2$ | $1 + 2 = 3$ | $1 + 3 = 4$ |
| $1 + 4 = 5$ | $2 + 2 = 4$ | $2 + 3 = 5$ |

# Unavoidable Monochromatic Solutions [Schur 1917]

Will any coloring of the positive integers with red and blue result in a monochromatic solution of $a + b = c$?

| | | |
|---|---|---|
| $1 + 1 = 2$ | $1 + 2 = 3$ | $1 + 3 = 4$ |
| $1 + 4 = 5$ | $2 + 2 = 4$ | $2 + 3 = 5$ |

# Unavoidable Monochromatic Solutions [Schur 1917]

Will any coloring of the positive integers with red and blue result in a monochromatic solution of $a + b = c$?

| | | |
|---|---|---|
| $1 + 1 = 2$ | $1 + 2 = 3$ | $1 + 3 = 4$ |
| $1 + 4 = 5$ | $2 + 2 = 4$ | $2 + 3 = 5$ |

# Unavoidable Monochromatic Solutions [Schur 1917]

Will any coloring of the positive integers with red and blue result in a monochromatic solution of $a + b = c$?

| | | |
|---|---|---|
| $1 + 1 = 2$ | $1 + 2 = 3$ | $1 + 3 = 4$ |
| $1 + 4 = 5$ | $2 + 2 = 4$ | $2 + 3 = 5$ |

# Unavoidable Monochromatic Solutions [Schur 1917]

Will any coloring of the positive integers with red and blue result in a monochromatic solution of $a + b = c$? Yes

| | | |
|---|---|---|
| $1 + 1 = 2$ | $1 + 2 = 3$ | $1 + 3 = 4$ |
| $1 + 4 = 5$ | $2 + 2 = 4$ | $2 + 3 = 5$ |

# Unavoidable Monochromatic Solutions [Schur 1917]

Will any coloring of the positive integers with red and blue result in a monochromatic solution of $a + b = c$? Yes

| | | |
|---|---|---|
| $1 + 1 = 2$ | $1 + 2 = 3$ | $1 + 3 = 4$ |
| $1 + 4 = 5$ | $2 + 2 = 4$ | $2 + 3 = 5$ |

Will any coloring of the positive integers with red and blue result in a monochromatic solution of $a^3 + b^3 = c^3$? No

# Unavoidable Monochromatic Solutions [Schur 1917]

Will any coloring of the positive integers with red and blue result in a monochromatic solution of $a + b = c$? Yes

| | | |
|---|---|---|
| $1 + 1 = 2$ | $1 + 2 = 3$ | $1 + 3 = 4$ |
| $1 + 4 = 5$ | $2 + 2 = 4$ | $2 + 3 = 5$ |

Will any coloring of the positive integers with red and blue result in a monochromatic solution of $a^3 + b^3 = c^3$? No

Will any coloring of the positive integers with red and blue result in a monochromatic solution of $a^2 + b^2 = c^2$? Maybe

| | | | |
|---|---|---|---|
| $3^2 + 4^2 = 5^2$ | $6^2 + 8^2 = 10^2$ | $5^2 + 12^2 = 13^2$ | $9^2 + 12^2 = 15^2$ |
| $8^2 + 15^2 = 17^2$ | $12^2 + 16^2 = 20^2$ | $15^2 + 20^2 = 25^2$ | $7^2 + 24^2 = 25^2$ |
| $10^2 + 24^2 = 26^2$ | $20^2 + 21^2 = 29^2$ | $18^2 + 24^2 = 30^2$ | $16^2 + 30^2 = 34^2$ |
| $21^2 + 28^2 = 35^2$ | $12^2 + 35^2 = 37^2$ | $15^2 + 36^2 = 39^2$ | $24^2 + 32^2 = 40^2$ |

# Pythagorean Triples Problem [Ronald Graham, early 1980s]

Will any coloring of the positive integers with red and blue result in a monochromatic Pythagorean Triple $a^2 + b^2 = c^2$?

# Pythagorean Triples Problem [Ronald Graham, early 1980s]

Will any coloring of the positive integers with red and blue result in a monochromatic Pythagorean Triple $a^2 + b^2 = c^2$?

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 No [PhD thesis, 2015].

# Pythagorean Triples Problem [Ronald Graham, early 1980s]

Will any coloring of the positive integers with red and blue result in a monochromatic Pythagorean Triple $a^2 + b^2 = c^2$?

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 No [PhD thesis, 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: $(x_a \vee x_b \vee x_c) \wedge (\neg x_a \vee \neg x_b \vee \neg x_c)$.

# Pythagorean Triples Problem [Ronald Graham, early 1980s]

Will any coloring of the positive integers with red and blue result in a monochromatic Pythagorean Triple $a^2 + b^2 = c^2$?

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

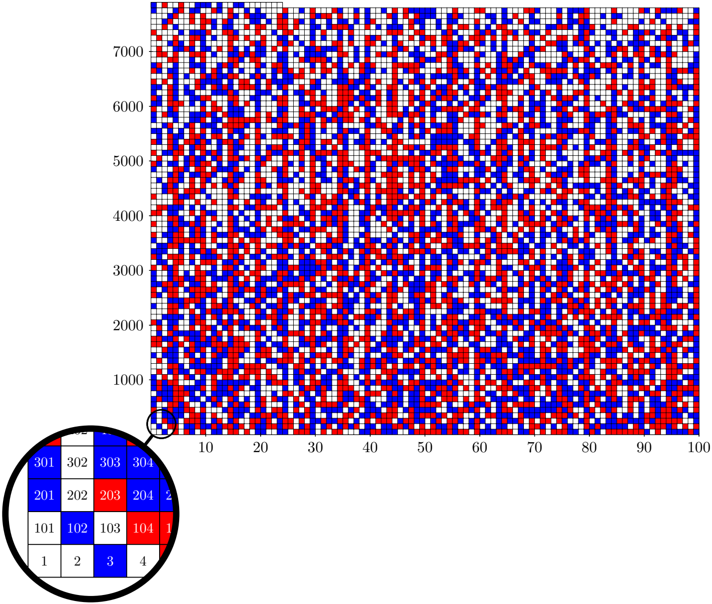
Myers conjectures that the answer is No [PhD thesis, 2015].

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: $(x_a \vee x_b \vee x_c) \wedge (\neg x_a \vee \neg x_b \vee \neg x_c)$.

**Theorem ([Heule, Kullmann, and Marek (2016)])**

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

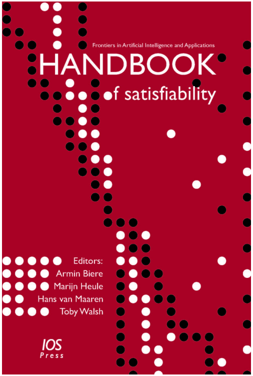
# A Monochromatic-Free Coloring of Maximal Size



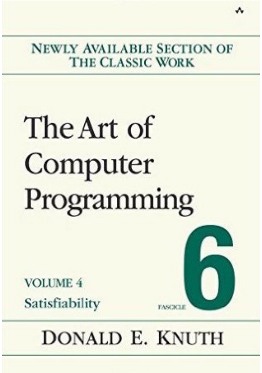
# Enormous Progress in the Last Two Decades

mid '90s: formulas solvable with thousands of variables and clauses

now: formulas solvable with millions of variables and clauses




Edmund Clarke: *"a key technology of the 21st century"*




Donald Knuth: *"evidently a killer app, because it is key to the solution of so many other problems"*

# SAT Solver Paradigms

Conflict-driven clause learning (CDCL):

- Makes fast decisions;
- Converts conflicting assignments into learned clauses.

Strength: Effective on large, “easy” formulas.

Weakness: Hard to parallelize.

# SAT Solver Paradigms

Conflict-driven clause learning (CDCL):

- Makes fast decisions;
- Converts conflicting assignments into learned clauses.

Strength: Effective on large, “easy” formulas.

Weakness: Hard to parallelize.

Look-ahead:

- Aims at finding a small binary search-tree;
- Splits the formula by looking ahead.

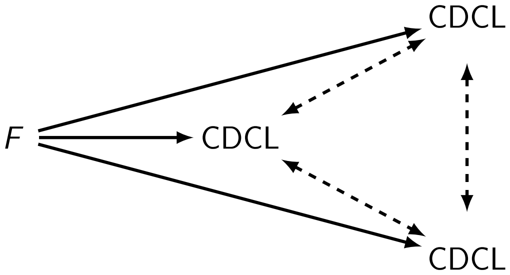
Strength: Effective on small, hard formulas.

Weakness: Expensive.

# Portfolio Solvers

The most commonly used parallel solving paradigm is portfolio:

- Run multiple (typically identical) solvers with different configurations on the same formula; and
- Share clauses among the solvers.



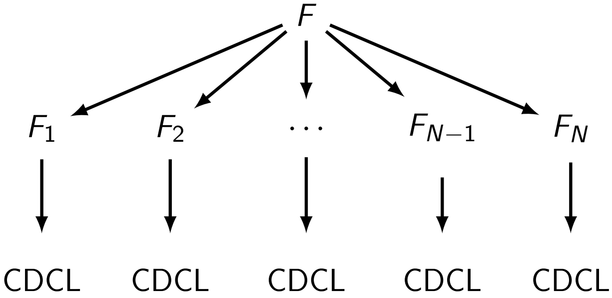

The portfolio approach is effective on large "easy" problems, but has difficulties to solve hard problems (out of memory).

# 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 near-equal splitting and the sub-problems are scheduled independently (easy parallel CDCL).

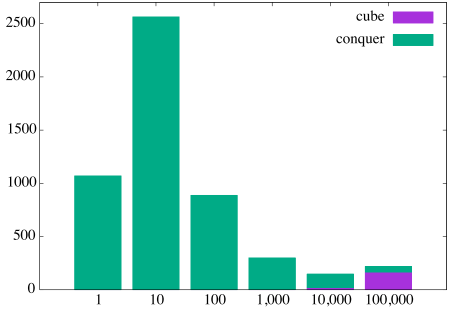
# The Hidden Strength of Cube-and-Conquer

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.

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

# Variant 1: Concurrent Cube-and-Conquer

The main heuristic challenge is deciding when to cut:

- Cutting too early results in hard subproblems for CDCL, thereby limiting the speed-up by parallelization (and the hidden strength).
- Cutting too late adds redundant lookahead costs.

Idea: Run a CDCL solver in parallel with the look-ahead solver:

- Both solvers work on the same subformula (assignment)
- Lookahead computes a good splitting variable
- Meanwhile CDCL tries to solve the subproblem
- The first solver that finishes determines the next step: A lookahead win $\rightarrow$ split, a CDCL win $\rightarrow$ backtrack.

# Variant 2: Cubes on Demand

Only split when CDCL cannot quickly solve a (sub)problem.

- Split when a certain limit is reach, say 10,000 conflicts — a dynamic limit works best in practice.
- The cores focus on solving the easier subproblems — the smallest formulas after propagating the cube units.

TREENGELING by Armin Biere is based on cubes on demand.

- Implements splitting by cloning the solver.
- Adds two solvers running on the original formula in parallel.

TREENGELING won the parallel track of SAT Competition 2016.

# Pythagorean Triples Results Summary [Heule et al. 2016]

- Almost linear speed-ups even when using 1000s of cores;
- The total computation was about 4 CPU years, but less than 2 days in wallclock time using 800 cores;
- If we use all 110 000 cores of TACC's Stampede cluster, then the problem can be solved in less than an hour;
- Reduced the trivial $2^{7825}$ cases to roughly $2^{40}$ cases.

# Pythagorean Triples Results Summary [Heule et al. 2016]

- Almost linear speed-ups even when using 1000s of cores;
- The total computation was about 4 CPU years, but less than 2 days in wallclock time using 800 cores;
- If we use all 110 000 cores of TACC's Stampede cluster, then the problem can be solved in less than an hour;
- Reduced the trivial $2^{7825}$ cases to roughly $2^{40}$ cases.

Comparison with state-of-the-art solver TREENGELING (T) (estimations based on Pythagorean Triples subproblems):

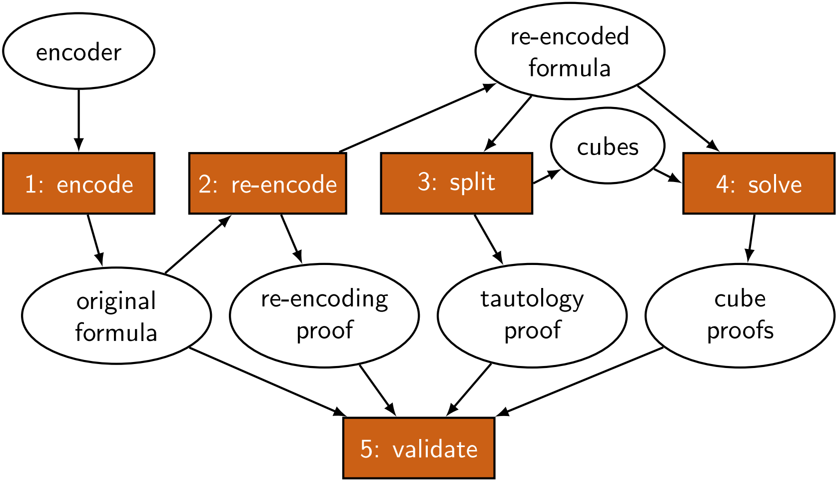
- T requires at least two orders of magnitude more CPU time;
- T's scaling is not linear: 100x speedup using 1000 cores;
- Using 1000 cores, T would use ∼40,000 hours wallclock time.

# Motivation for Validating Proofs of Unsatisfiability

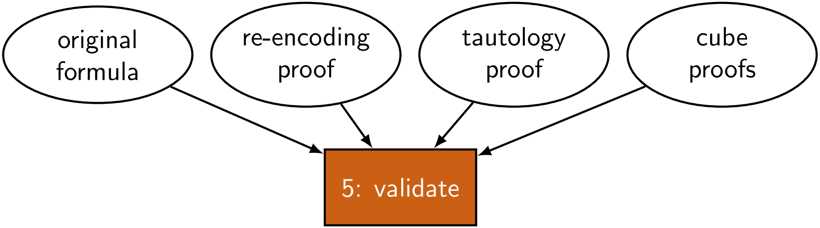
SAT solvers may have errors and only return yes/no.

- Documented bugs in SAT, SMT, and QSAT 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 verifiable.

# Overview of Solving Framework with Proof Verification



# Phase 5: Validate Pythagorean Triples Proofs




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

Proofs can be validated in parallel [Heule and Biere 2015].

The proof has recently been certified using verified checkers.

# Conclusions

Parallel SAT solving has been very successful:

- Industry uses SAT for hardware verification tasks;
- Long-standing open math problems can now be solved;
- The results can be certified using highly-trusted systems.

There is a bright future with interesting challenges:

- How to deal with hard software verification problems?
- Can machine learning be used to improve performance?
- How to create a parallel SAT solver with linear time speedups on a wide spectrum of problems using many thousands of cores (working out of the box)?

# Solving Very Hard Problems: Cube-and-Conquer, a Hybrid SAT Solving Method

**Marijn J.H. Heule**

THE UNIVERSITY OF TEXAS AT AUSTIN

*Joint work with Armin Biere, Oliver Kullmann, and Victor W. Marek*

Parallel Constraint Reasoning August 6, 2017