SAT4Math
Introduction & Solvers

Marijn J.H. Heule

Carnegie Mellon University

scholars

Summer School Marktoberdorf August 6, 2025

sat4math.com

AI for Mathematics

A.I. Is Coming for Mathematics, Too

For thousands of years, mathematicians have adapted to the latest advances in logic and reasoning. Are they ready for artificial intelligence?

Move Over, Mathematicians, Here Comes AlphaProof

A.I. is getting good at math — and might soon make a worthy collaborator for humans.

AI for Mathematics

A.I. Is Coming for Mathematics, Too

For thousands of years, mathematicians have adapted to the latest advances in logic and reasoning. Are they ready for artificial intelligence?

Move Over, Mathematicians, Here Comes AlphaProof

A.I. is getting good at math — and might soon make a worthy collaborator for humans.

Mathematics is the perfect playground to get AI right

- Formal methods offers essential logic-based reasoning
- Highly trustworthy results thanks to (formal) proofs

50 Years of Successes in Computer-Aided Mathematics

- 1976 Four-Color Theorem
- 1998 Kepler Conjecture
- 2014 Boolean Erdős discrepancy problem
- 2016 Boolean Pythagorean triples problem
- 2018 Schur Number Five
- 2019 Keller's Conjecture
- 2021 Kaplansky's Unit Conjecture
- 2022 Packing Number of Square Grid
- 2023 Empty Hexagon in Every 30 Points

50 Years of Successes in Computer-Aided Mathematics

1976 Four-Color Theorem

1998 Kepler Conjecture

2014 Boolean Erdős discrepancy problem (using a SAT solver)

2016 Boolean Pythagorean triples problem (using a SAT solver)

2018 Schur Number Five (using a SAT solver)

2019 Keller's Conjecture (using a SAT solver)

2021 Kaplansky's Unit Conjecture (using a SAT solver)

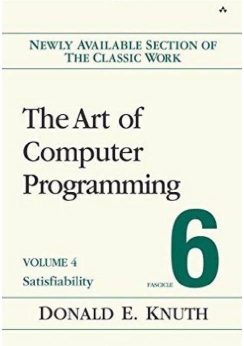
2022 Packing Number of Square Grid (using a SAT solver)

2023 Empty Hexagon in Every 30 Points (using a SAT solver)

Breakthrough in SAT Solving in the Last 30 Years

Satisfiability (SAT) problem: Can a Boolean formula be satisfied?

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"* [Biere, Heule, vanMaaren, Walsh '09/'21]

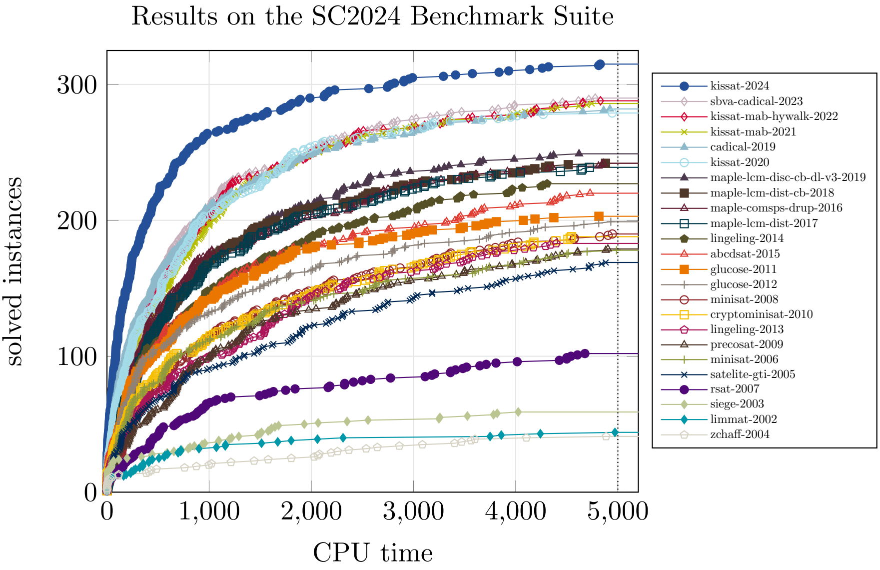
Donald Knuth: *"evidently a killer app, because it is key to the solution of so many other problems"* [Knuth '15]

Naive SAT Solving: Truth Table

$$\Gamma := (p \vee \neg q) \wedge (q \vee r) \wedge (\neg r \vee \neg p)$$

p	q	r	falsifies	eval(Γ)
$\bot$	$\bot$	$\bot$	$q \vee r$	$\bot$
$\bot$	$\bot$	$\top$	—	$\top$
$\bot$	$\top$	$\bot$	$p \vee \neg q$	$\bot$
$\bot$	$\top$	$\top$	$p \vee \neg q$	$\bot$
$\top$	$\bot$	$\bot$	$q \vee r$	$\bot$
$\top$	$\bot$	$\top$	$\neg r \vee \neg p$	$\bot$
$\top$	$\top$	$\bot$	—	$\top$
$\top$	$\top$	$\top$	$\neg r \vee \neg p$	$\bot$

Progress of SAT Solvers



Schur's Theorem [Schur 1916]

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

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

Schur's Theorem [Schur 1916]

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

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

Schur's Theorem [Schur 1916]

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

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

Schur's Theorem [Schur 1916]

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

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

Schur's Theorem [Schur 1916]

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

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

Schur's Theorem [Schur 1916]

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

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

Theorem (Schur's Theorem)

For every positive integer k, there exists a number $S(k)$, such that $[1, S(k)]$ can be colored with k colors while avoiding a monochromatic solution of $a + b = c$ with $a, b, c \leq S(k)$, while this is impossible for $[1, S(k)+1]$.

$S(1) = 1, S(2) = 4, S(3) = 13, S(4) = 44$ [Baumert 1965].

Schur's Theorem [Schur 1916]

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

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

Theorem (Schur's Theorem)

For every positive integer k, there exists a number $S(k)$, such that $[1,S(k)]$ can be colored with k colors while avoiding a monochromatic solution of $a+b=c$ with $a,b,c \leq S(k)$, while this is impossible for $[1,S(k)+1]$.

$S(1)=1, S(2)=4, S(3)=13, S(4)=44$ [Baumert 1965].

We show that $S(5)=160$ [Heule 2018].

Schur's Theorem [Schur 1916]

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

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

Theorem (Schur's Theorem)

For every positive integer k, there exists a number $S(k)$, such that $[1, S(k)]$ can be colored with k colors while avoiding a monochromatic solution of $a + b = c$ with $a, b, c \leq S(k)$, while this is impossible for $[1, S(k)+1]$.

$S(1) = 1, S(2) = 4, S(3) = 13, S(4) = 44$ [Baumert 1965].

We show that $S(5) = 160$ [Heule 2018]. Proof: 2 petabytes

Pythagorean Triples Problem (I) [Ronald Graham, early 80's]

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

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

Pythagorean Triples Problem (I) [Ronald Graham, early 80's]

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

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

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 (II) [Ronald Graham, early 80's]

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

A bi-coloring of $[1, n]$ is encoded using Boolean variables p_i with $i \in \{1, 2, \ldots, n\}$ such that $p_i = 1$ $(= 0)$ means that i is colored red (blue). For each Pythagorean Triple $a^2 + b^2 = c^2$, two clauses are added: $(p_a \vee p_b \vee p_c)$ and $(\neg p_a \vee \neg p_b \vee \neg p_c)$.

Pythagorean Triples Problem (II) [Ronald Graham, early 80's]

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

A bi-coloring of $[1, n]$ is encoded using Boolean variables p_i with $i \in \{1, 2, \ldots, n\}$ such that $p_i = 1$ $(= 0)$ means that i is colored red (blue). For each Pythagorean Triple $a^2 + b^2 = c^2$, two clauses are added: $(p_a \vee p_b \vee p_c)$ and $(\neg p_a \vee \neg p_b \vee \neg p_c)$.

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

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

Pythagorean Triples Problem (II) [Ronald Graham, early 80's]

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

A bi-coloring of $[1, n]$ is encoded using Boolean variables p_i with $i \in \{1, 2, \ldots, n\}$ such that $p_i = 1$ $(= 0)$ means that i is colored red (blue). For each Pythagorean Triple $a^2 + b^2 = c^2$, two clauses are added: $(p_a \vee p_b \vee p_c)$ and $(\neg p_a \vee \neg p_b \vee \neg p_c)$.

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

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

4 CPU years computation, but 2 days on cluster (800 cores)

Pythagorean Triples Problem (II) [Ronald Graham, early 80's]

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

A bi-coloring of $[1, n]$ is encoded using Boolean variables p_i with $i \in \{1, 2, \ldots, n\}$ such that $p_i = 1$ $(= 0)$ means that i is colored red (blue). For each Pythagorean Triple $a^2 + b^2 = c^2$, two clauses are added: $(p_a \vee p_b \vee p_c)$ and $(\neg p_a \vee \neg p_b \vee \neg p_c)$.

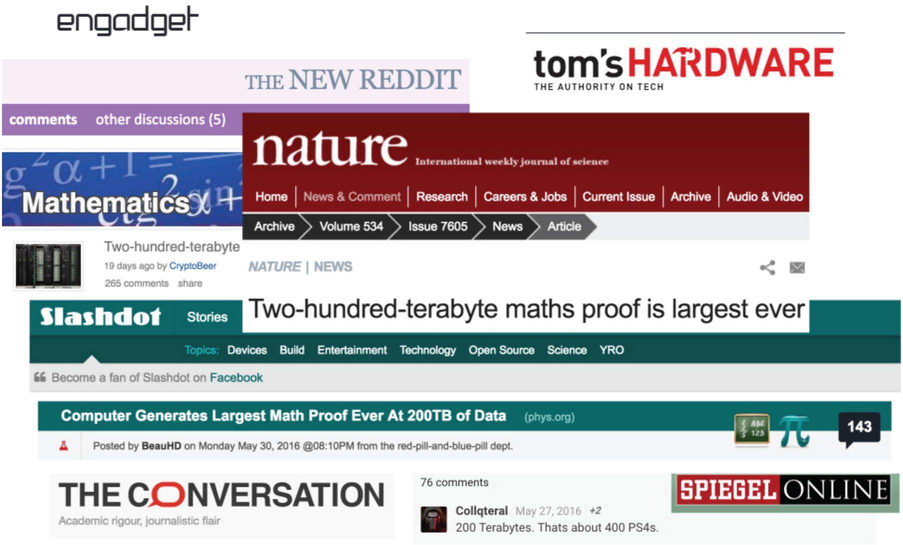
Theorem ([Heule, Kullmann, and Marek (2016)])

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

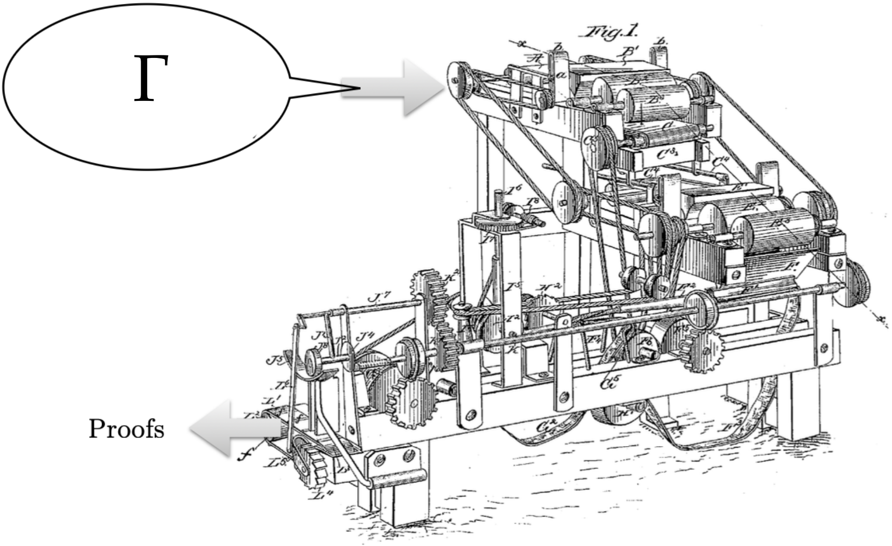
4 CPU years computation, but 2 days on cluster (800 cores)

200 terabytes proof, but validated with verified checker

Media: “The Largest Math Proof Ever”



SAT Solvers are Complex Tools

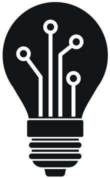


SAT Solver Paradigms Overview

DPLL: Aims at finding a small search-tree by selecting effective splitting variables (e.g. via looking ahead).

Strength: Effective on small, hard formulas.

Weakness: Expensive.

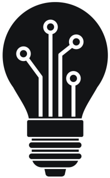


SAT Solver Paradigms Overview

DPLL: Aims at finding a small search-tree by selecting effective splitting variables (e.g. via looking ahead).

Strength: Effective on small, hard formulas.

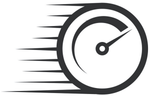
Weakness: Expensive.



Conflict-driven clause learning (CDCL): Makes fast decisions and converts conflicts into learned clauses.

Strength: Effective on large, “easy” formulas.

Weakness: Hard to parallelize.

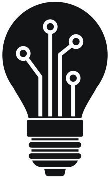


SAT Solver Paradigms Overview

DPLL: Aims at finding a small search-tree by selecting effective splitting variables (e.g. via looking ahead).

Strength: Effective on small, hard formulas.

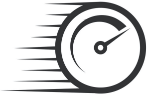
Weakness: Expensive.



Conflict-driven clause learning (CDCL): Makes fast decisions and converts conflicts into learned clauses.

Strength: Effective on large, “easy” formulas.

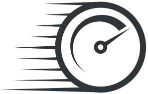
Weakness: Hard to parallelize.



Local search: Given a full assignment for a formula Γ, flip the truth values of variables until satisfying Γ.

Strength: Can quickly find solutions for hard formulas.

Weakness: Cannot prove unsatisfiability.

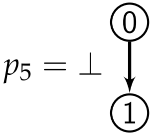


Conflict-driven SAT solvers: Search and Analysis

$(p_1 \vee p_4) \wedge$
$(p_3 \vee \neg p_4 \vee p_5) \wedge$
$(\neg p_2 \vee \neg p_3 \vee \neg p_4) \wedge$
Γ_{extra}

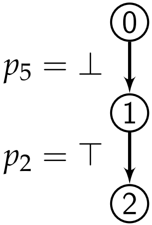
Conflict-driven SAT solvers: Search and Analysis

$$
\begin{aligned}
&(p_1 \vee p_4) \wedge \\
&(p_3 \vee \neg p_4 \vee p_5) \wedge \\
&(\neg p_2 \vee \neg p_3 \vee \neg p_4) \wedge \\
&\Gamma_{\text{extra}}
\end{aligned}
$$



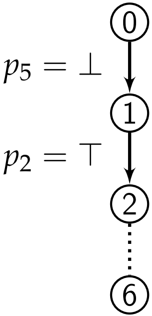
Conflict-driven SAT solvers: Search and Analysis

$(p_1 \vee p_4) \wedge$
$(p_3 \vee \neg p_4 \vee p_5) \wedge$
$(\neg p_2 \vee \neg p_3 \vee \neg p_4) \wedge$
Γ_{extra}



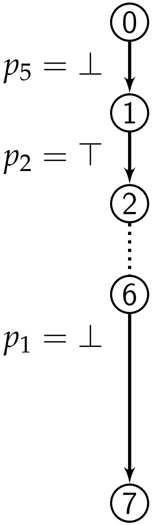
Conflict-driven SAT solvers: Search and Analysis

$(p_1 \vee p_4) \wedge$
$(p_3 \vee \neg p_4 \vee p_5) \wedge$
$(\neg p_2 \vee \neg p_3 \vee \neg p_4) \wedge$
Γ_{extra}



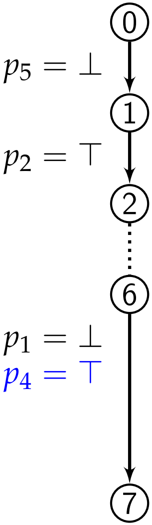
Conflict-driven SAT solvers: Search and Analysis

$(p_1 \vee p_4) \wedge$
$(p_3 \vee \neg p_4 \vee p_5) \wedge$
$(\neg p_2 \vee \neg p_3 \vee \neg p_4) \wedge$
Γ_{extra}



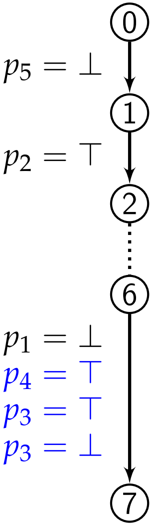
Conflict-driven SAT solvers: Search and Analysis

$(p_1 \vee p_4) \wedge$
$(p_3 \vee \neg p_4 \vee p_5) \wedge$
$(\neg p_2 \vee \neg p_3 \vee \neg p_4) \wedge$
Γ_{extra}



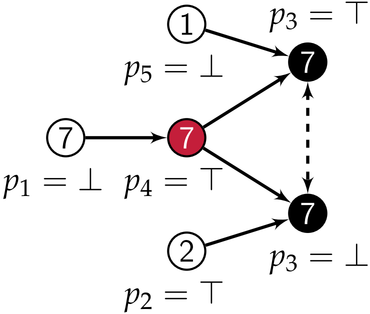
Conflict-driven SAT solvers: Search and Analysis

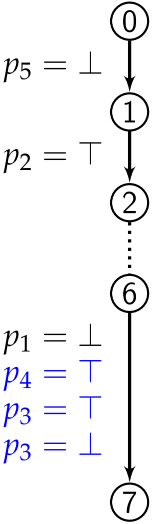
$(p_1 \vee p_4) \wedge$
$(p_3 \vee \neg p_4 \vee p_5) \wedge$
$(\neg p_2 \vee \neg p_3 \vee \neg p_4) \wedge$
Γ_{extra}



Conflict-driven SAT solvers: Search and Analysis

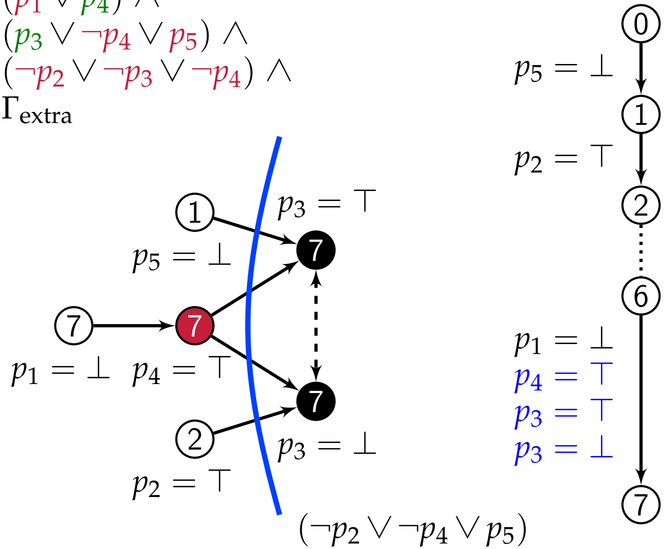
$(p_1 \vee p_4) \wedge$
$(p_3 \vee \neg p_4 \vee p_5) \wedge$
$(\neg p_2 \vee \neg p_3 \vee \neg p_4) \wedge$
Γ_{extra}





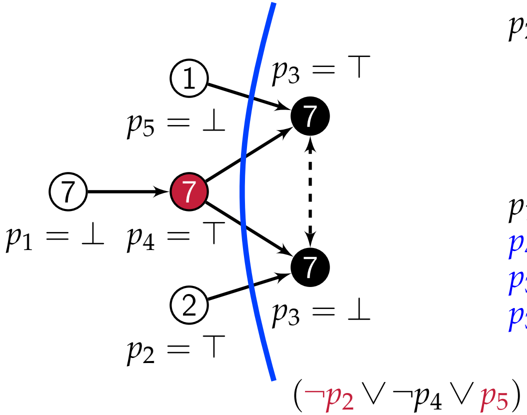
Conflict-driven SAT solvers: Search and Analysis

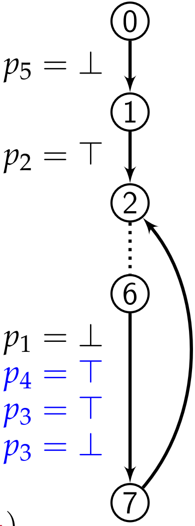
$(p_1 \vee p_4) \wedge$
$(p_3 \vee \neg p_4 \vee p_5) \wedge$
$(\neg p_2 \vee \neg p_3 \vee \neg p_4) \wedge$
Γ_{extra}



Conflict-driven SAT solvers: Search and Analysis

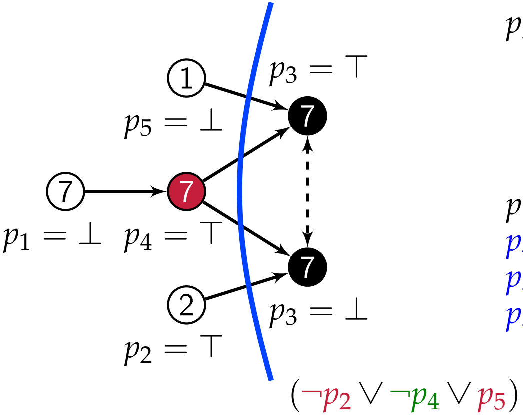
$(p_1 \vee p_4) \wedge$
$(p_3 \vee \neg p_4 \vee p_5) \wedge$
$(\neg p_2 \vee \neg p_3 \vee \neg p_4) \wedge$
Γ_{extra}

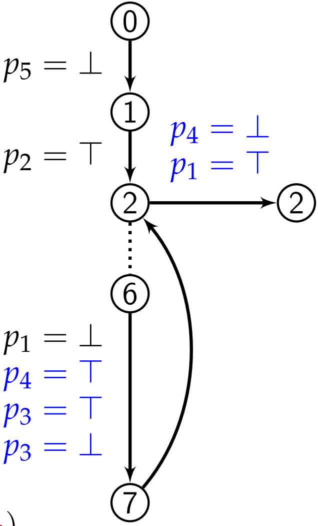




Conflict-driven SAT solvers: Search and Analysis

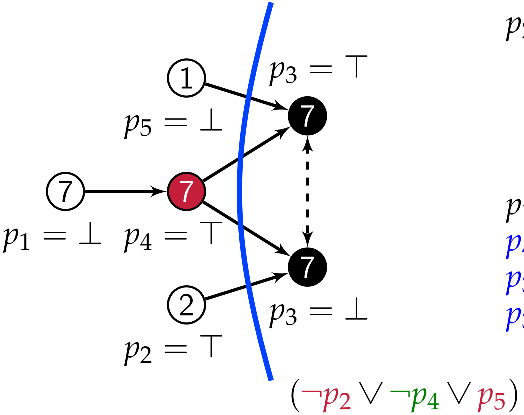
$(p_1 \vee p_4) \wedge$
$(p_3 \vee \neg p_4 \vee p_5) \wedge$
$(\neg p_2 \vee \neg p_3 \vee \neg p_4) \wedge$
Γ_{extra}

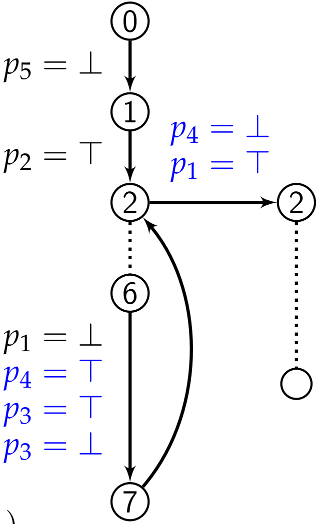




Conflict-driven SAT solvers: Search and Analysis

$(p_1 \vee p_4) \wedge$
$(p_3 \vee \neg p_4 \vee p_5) \wedge$
$(\neg p_2 \vee \neg p_3 \vee \neg p_4) \wedge$
Γ_{extra}





CDCL Overview

CDCL in a nutshell:

1. Main loop combines efficient problem simplification with cheap, but effective decision heuristics; ($> 90\%$ of time)
2. Reasoning kicks in if the current state is conflicting;
3. The current state is analyzed and turned into a constraint;
4. The constraint is added to the problem, the heuristics are updated, and the algorithm (partially) restarts.

CDCL Overview

CDCL in a nutshell:

1. Main loop combines efficient problem simplification with cheap, but effective decision heuristics; ($>90\%$ of time)
2. Reasoning kicks in if the current state is conflicting;
3. The current state is analyzed and turned into a constraint;
4. The constraint is added to the problem, the heuristics are updated, and the algorithm (partially) restarts.

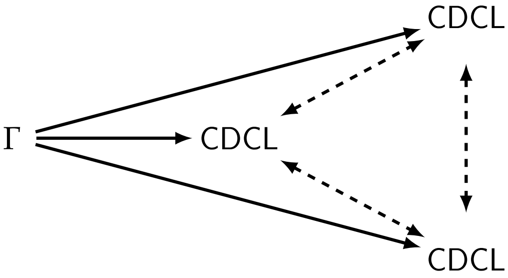
However, it has three weaknesses:

- CDCL is notoriously hard to parallelize;
- the representation impacts CDCL performance; and
- CDCL has exponential runtime on some "simple" problems.

Parallel Computing: Portfolio Solvers

The most commonly used parallel solving paradigm is portfolio:

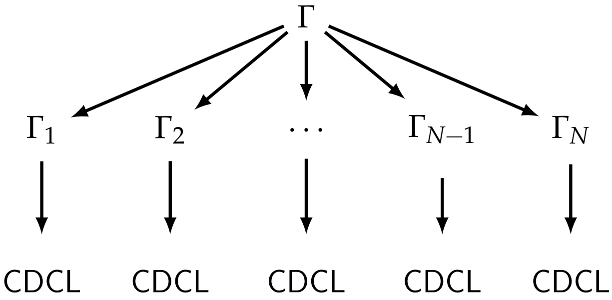
- Run multiple (typically identical) solvers with different configurations on the same formula; and
- Share clauses among the solvers.



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 '11]

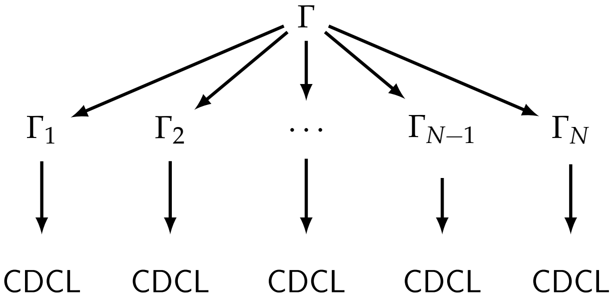
Cube-and-conquer splits a given problem into millions of subproblems that are solved independently by CDCL.



Efficient look-ahead splitting heuristics allow for linear speedups even when using 1000s of cores.

Cube-and-Conquer [Heule, Kullmann, Wieringa, and Biere '11]

Cube-and-conquer splits a given problem into millions of subproblems that are solved independently by CDCL.



Efficient look-ahead splitting heuristics allow for linear speedups even when using 1000s of cores.

Cube-and-conquer also integrated in SMT solvers

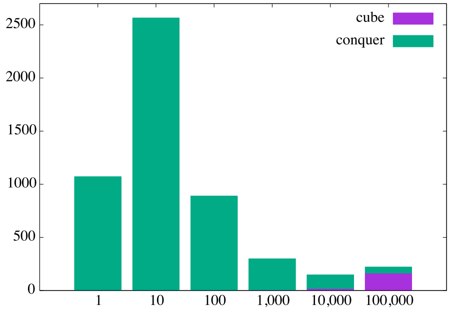
The Hidden Strength of Cube-and-Conquer

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 splitting.

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.

Parallel Computing: SAT Competition Cloud Track

Long tradition of SAT competitive events, starting from 1992

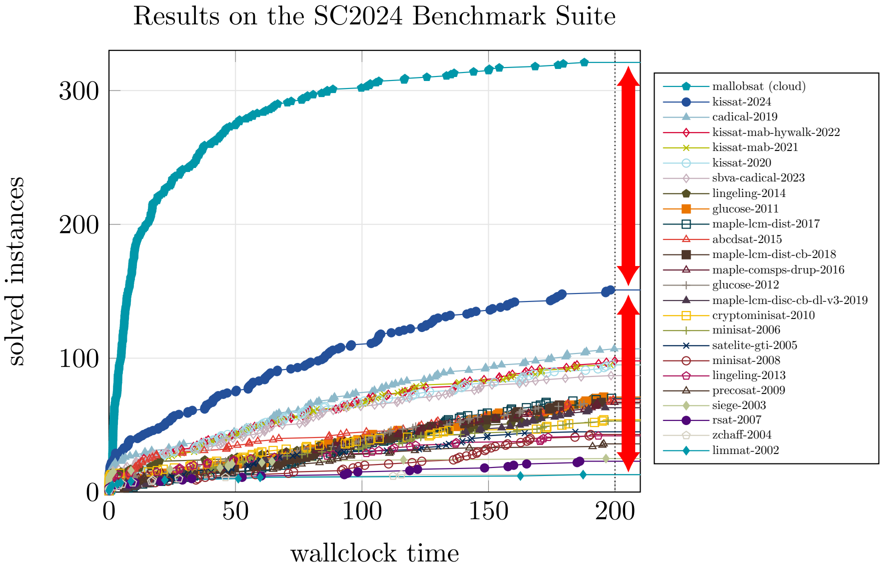
- 3 competitions in the 90s (1992,1993, 1996)
- 17 SAT Competitions (2002–)
- 5 SAT Races (2006, 2008, 2010, 2015, 2019)
- 1 SAT Challenge (2012)

Since SAT Competition 2020

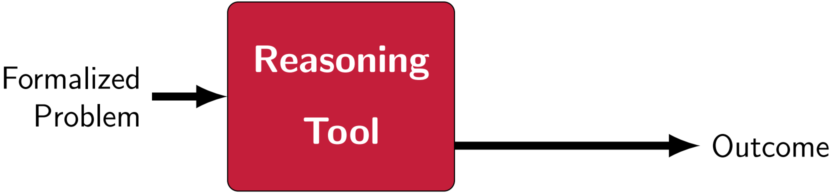
- Cloud Track – evaluate distributed solvers on the Amazon cloud. Solvers are run on 1600 virtual cores for 1000 seconds. Sponsored by Amazon. Participants received AWS credit to develop their solvers.

Winner of the cloud track clearly outperformed sequential winner

Effectiveness of Cloud Solvers



Automated Reasoning Programs



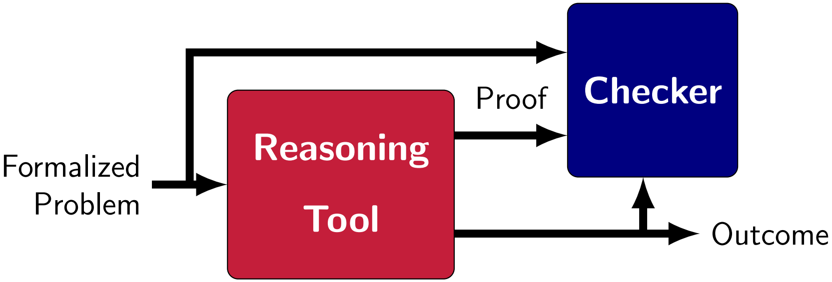
Standard Implementations

- Lingering doubt about whether result can be trusted
- If find bug in tool, must rerun all prior verifications

Formally Verified Tools

- Hard to develop
- Hard to make scalable

Proof-Generating Automated Reasoning Programs



Proof-Generating Tools

- Only need to prove individual executions, not entire program
- Can have bugs in tool but still trust result
- Can we trust the checker?
  - Simple algorithms and implementation
  - Ideally formally verified

Proof-Generating Tools: Arbitrarily Complex Solvers

Proof-generating tools with verified checkers is a powerful idea:

- Don't worry about correctness or completeness of tools;
- Facilitates making tools more complex and efficient; while
- Full confidence in results. [Heule, Hunt, Kaufmann, Wetzler '17]

Formally verified checkers now also used in industry

Tutorials on SAT for Mathematics

`sat4math.com/tutorials/`

DEMO