Distributed and Trustworthy Automated Reasoning

Marijn J.H. Heule

Carnegie Mellon University

scholars

High Confidence Software and Systems September 17, 2020

Automated Reasoning Has Many Applications

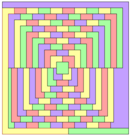
formal verification

security

bioinformatics

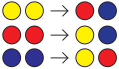
planning and scheduling

train safety



automated theorem proving

exploit generation



term rewriting termination

encode

automated reasoning

decode

Automated Reasoning Has Many Applications

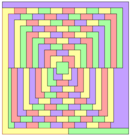
formal verification

security

bioinformatics

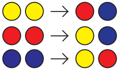
planning and scheduling

train safety



automated theorem proving

exploit generation



term rewriting termination

encode

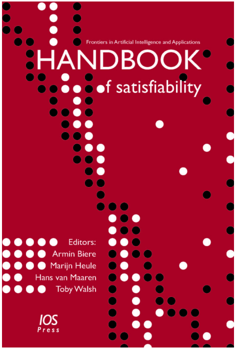
automated reasoning

decode

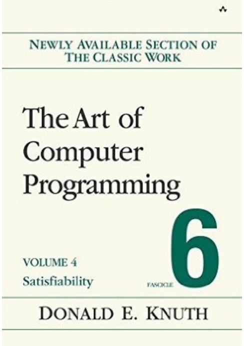
Breakthrough in SAT Solving in the Last 20 Years

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

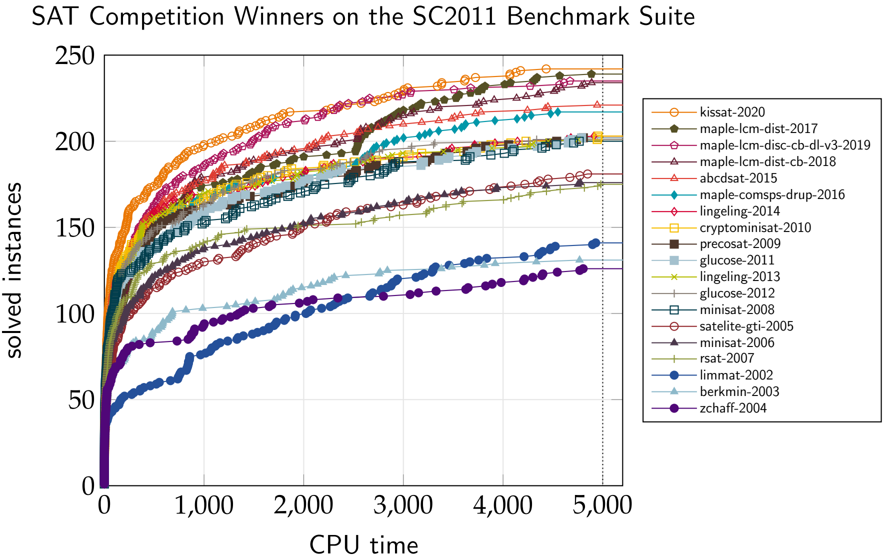


Edmund Clarke: *"a key technology of the 21st century"* [Biere, Heule, vanMaaren, and Walsh '09]



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

Progress in SAT Solving (I)

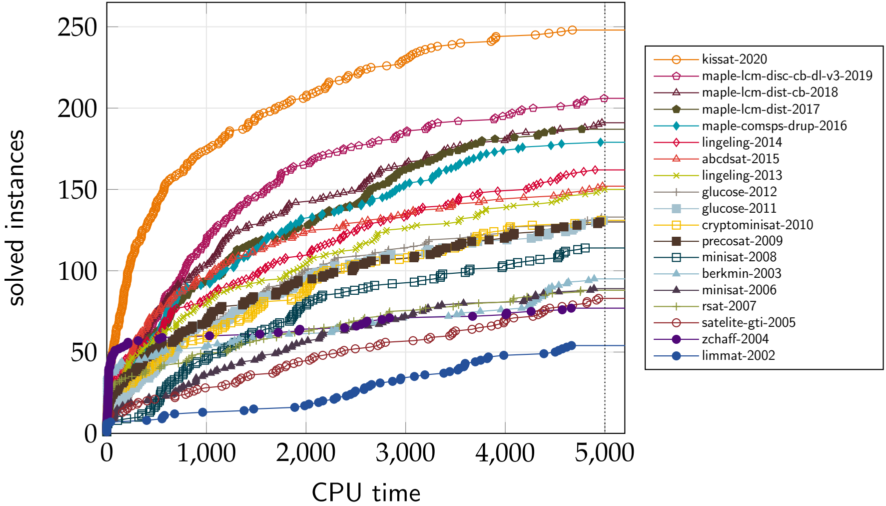


Benchmark suite used to show progress and get funding

Fisher: "instrumental ... for launching DARPA's HACMS program"

Progress in SAT Solving (II)

SAT Competition Winners on the SC2020 Benchmark Suite



Progress even larger due to harder instances

Examples of Challenges

Trusted Computing

Parallel Computing

Conclusions and Challenges

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 $a^2 + b^2 = c^2$?

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

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 $a^2 + b^2 = c^2$?

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

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 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: $(x_a \vee x_b \vee x_c)$ and $(\overline{x}_a \vee \overline{x}_b \vee \overline{x}_c)$.

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 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: $(x_a \vee x_b \vee x_c)$ and $(\overline{x}_a \vee \overline{x}_b \vee \overline{x}_c)$.

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

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

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 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: $(x_a \vee x_b \vee x_c)$ and $(\overline{x}_a \vee \overline{x}_b \vee \overline{x}_c)$.

Theorem ([Heule, Kullmann, and Marek (2016)])
$[1, 7824]$ *can be bi-colored s.t. there is no monochromatic Pythagorean Triple. This is impossible for* $[1, 7825]$.

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 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: $(x_a \vee x_b \vee x_c)$ and $(\overline{x}_a \vee \overline{x}_b \vee \overline{x}_c)$.

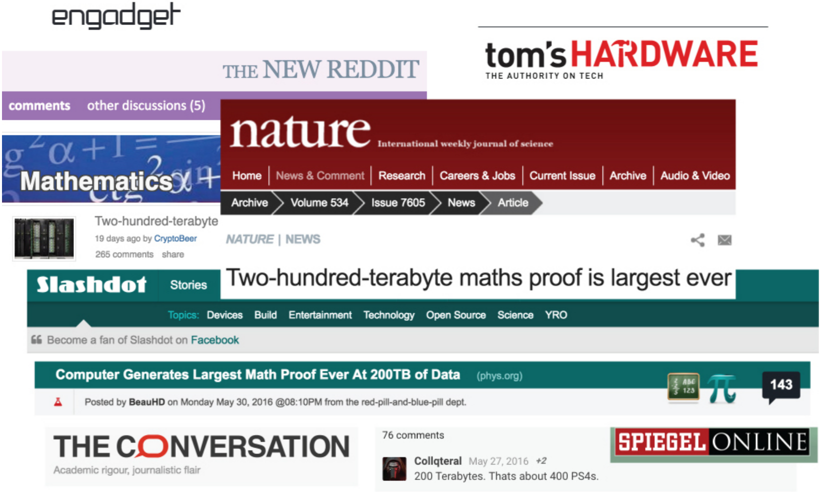
Theorem ([Heule, Kullmann, and Marek (2016)])

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

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

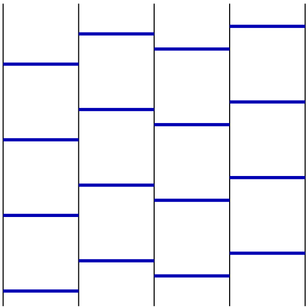
200 terabytes proof, but validated with verified checker

“The Largest Math Proof Ever”



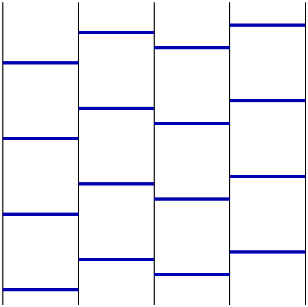
Another Example: Tiling in various Dimensions

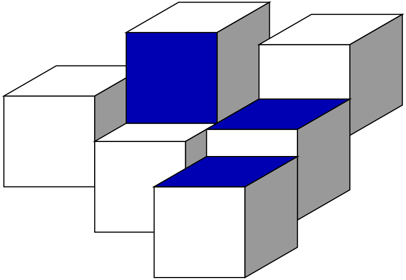
Consider tiling a floor with square tiles, all of the same size. Is it the case that any gap-free tiling results in at least two fully connected tiles, i.e., tiles that have an entire edge in common?



Another Example: Tiling in various Dimensions

Consider tiling a floor with square tiles, all of the same size. Is it the case that any gap-free tiling results in at least two fully connected tiles, i.e., tiles that have an entire edge in common?



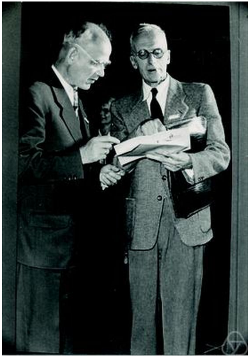


Keller's Conjecture

In 1930, Ott-Heinrich Keller conjectured that this phenomenon holds in every dimension.

Keller's Conjecture.
For all $n \geq 1$, every tiling of the n-dimensional space with unit cubes has two which fully share a face.

- In 1940, Perron proved that Keller's conjecture is true for $1 \leq n \leq 6$.
- In 1992, Lagarias and Shor showed that it is false for $n \geq 10$.
- In 2002, Mackey showed that it is false for $n \geq 8$.



[Wikipedia, CC BY-SA]

Keller's Conjecture Resolved [Brakensiek, Heule, Mackey, Narváez '20]

Quanta Magazine: "Computer Search Settles 90-Year-Old Math Problem"

The final dimension of Keller's conjecture finally resolved:

- Tools worked out of the box, linear time speedups;
- The complex symmetry-breaking argument is included in the proof;
- The proof has been validated using a verified checker.

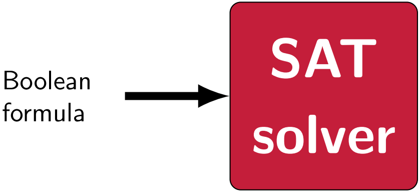
Examples of Challenges

Trusted Computing

Parallel Computing

Conclusions and Challenges

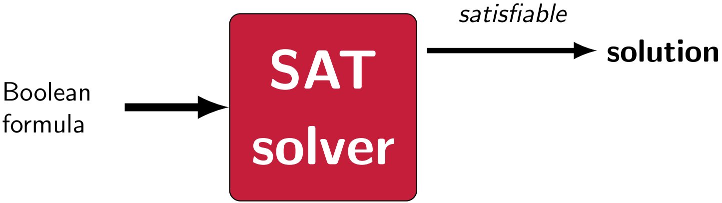
Trusted Computing: Checking Satisfiability is Easy



SAT Solvers Useful & Powerful

- Formal verification
- Security verification
- Optimization

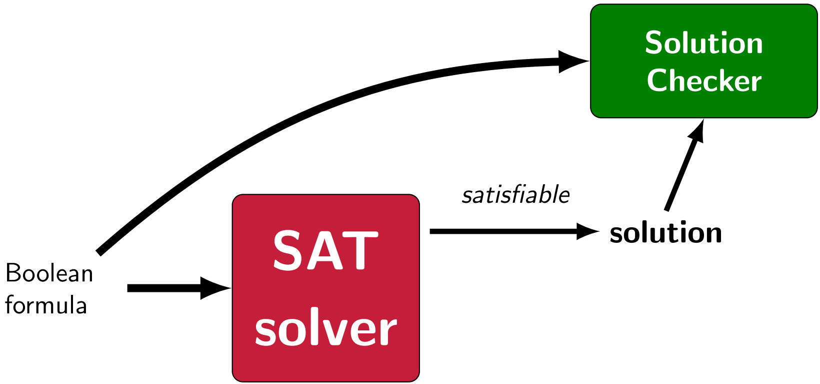
Trusted Computing: Checking Satisfiability is Easy



SAT Solvers Useful & Powerful

- Formal verification
- Security verification
- Optimization

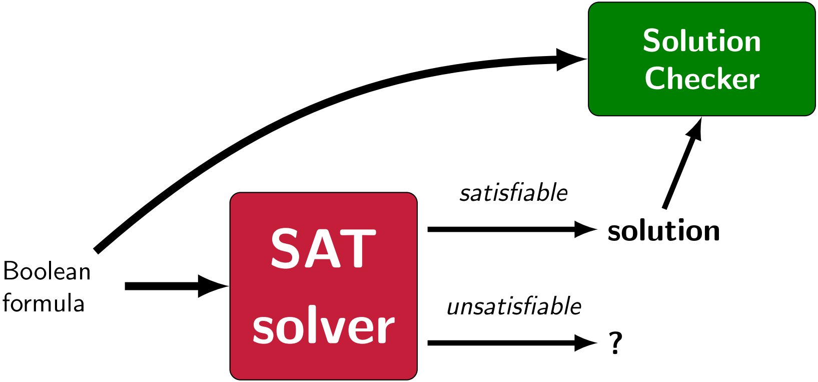
Trusted Computing: Checking Satisfiability is Easy



SAT Solvers Useful & Powerful

- Formal verification
- Security verification
- Optimization

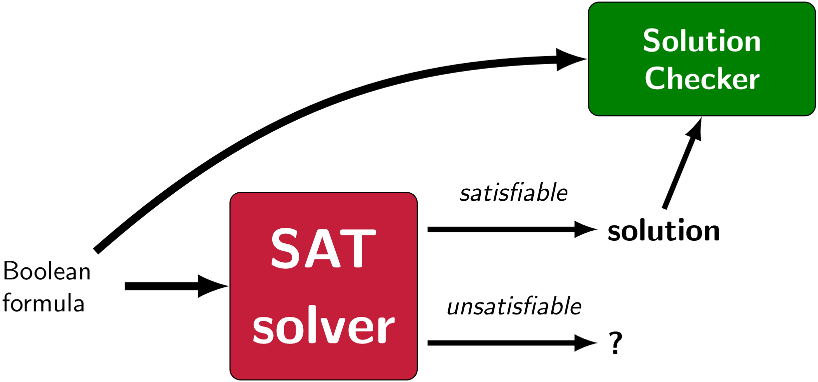
Trusted Computing: Checking Satisfiability is Easy



SAT Solvers Useful & Powerful

- Formal verification
- Security verification
- Optimization

Trusted Computing: Checking Satisfiability is Easy



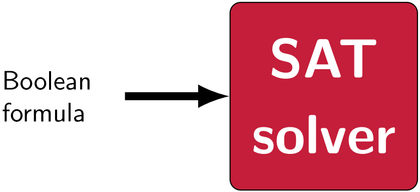
SAT Solvers Useful & Powerful

- Formal verification
- Security verification
- Optimization

Can We Trust Them?

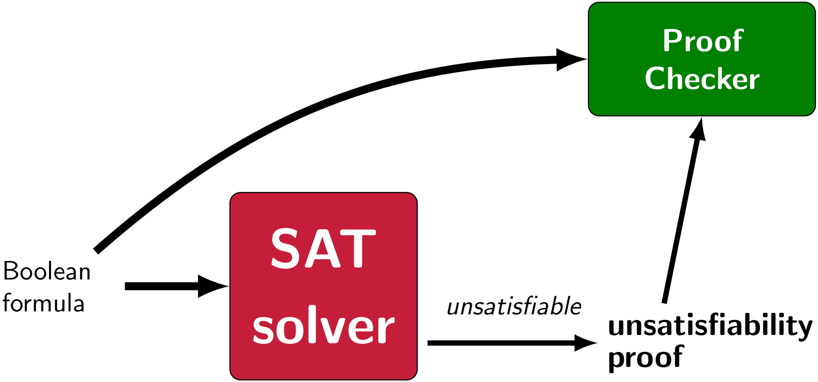
- No!
- Complex software with lots of optimizations

Trusted Computing: Proof Generating Solvers

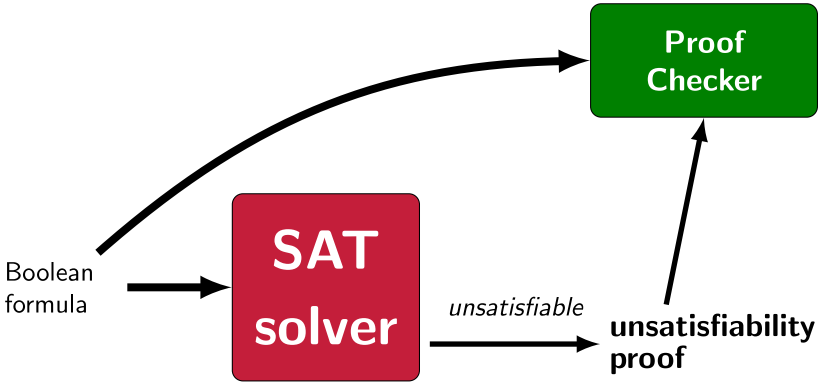


Trusted Computing: Proof Generating Solvers

Trusted Computing: Proof Generating Solvers



Trusted Computing: Proof Generating Solvers



Unsatisfiability Proof

- Step-by-step proof in some logical framework

Proof Checker

- Simple program
- May be formally verified

Trusted Computing: Motivation

Automated reasoning tools may give incorrect answers.

- Documented bugs in SAT, SMT, and QSAT solvers;
  [Brummayer and Biere, 2009; Brummayer et al., 2010]
- Claims of correctness could be due to bugs;
- Misconception that only weak tools are buggy;
- Implementation errors often imply conceptual errors;
- Proofs now mandatory in some competitive events;
- Mathematical results require a stronger justification than a simple yes/no by a tool. Answers must be verifiable.

Trusted Computing: Verified Solving versus Verified Proofs

Verifying efficient automated reasoning tools is a daunting task:

- Tools are constantly modified and improved; and
- Even top-tier and "experimentally correct" solvers turned out to be buggy. [Järvisalo, Heule, Biere '12]

Trusted Computing: Verified Solving versus Verified Proofs

Verifying efficient automated reasoning tools is a daunting task:

- Tools are constantly modified and improved; and
- Even top-tier and "experimentally correct" solvers turned out to be buggy. [Järvisalo, Heule, Biere '12]

Various simple solvers can verified, but they lack performance

- DPLL [Shankar and Vaucher '11]
- CDCL [Fleury, Blanchette, Lammich '18]

Trusted Computing: Verified Solving versus Verified Proofs

Verifying efficient automated reasoning tools is a daunting task:

- Tools are constantly modified and improved; and
- Even top-tier and "experimentally correct" solvers turned out to be buggy. [Järvisalo, Heule, Biere '12]

Various simple solvers can verified, but they lack performance

- DPLL [Shankar and Vaucher '11]
- CDCL [Fleury, Blanchette, Lammich '18]

Validating proof is the more effective approach

- Solving + proof logging + proof verification is much faster compared to running a verified solver
- One verified tool can validate the results of many solvers

Trusted Computing: Initial Challenges

Theoretical challenges:

- Some “simple” problems have exponentially large proofs in the resolution proof system [Urquhart ’87, Buss and Pitassi ’98];
- While some dedicated techniques can quickly solve them.

Solution: A proof system to compactly express all techniques.

Trusted Computing: Initial Challenges

Theoretical challenges:

- Some "simple" problems have exponentially large proofs in the resolution proof system [Urquhart '87, Buss and Pitassi '98];
- While some dedicated techniques can quickly solve them.

Solution: A proof system to compactly express all techniques.

Practical challenges:

- Earlier efforts failed due to complexity and overhead
- Convince developers to support proof logging

Solution:

- The computational burden and complexity is in the checker
- A reference implementation of proof logging

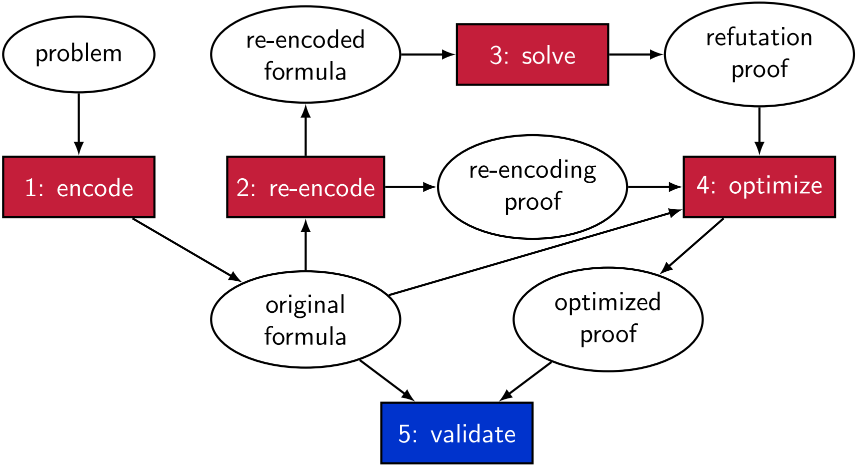
Trusted Computing: Arbitrarily Complex Solvers

Verified checkers of certificates in strong proof systems:

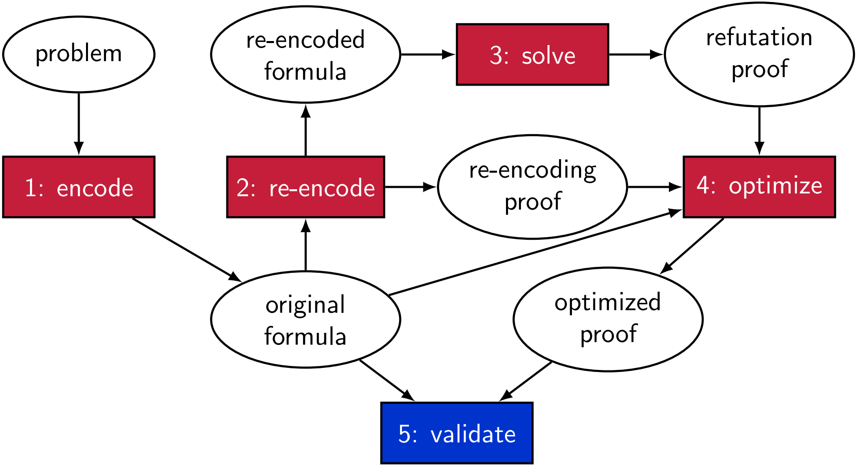
- Don't worry about correctness or completeness of tools;
- Facilitates making tools more complex and efficient; while
- Full confidence in results. [Heule, Hunt, Kaufmann, Wetzler '17]

Formally verified checkers now also used in industry

Trusted Computing: Verified SAT Solving Tool Chain



Trusted Computing: Verified SAT Solving Tool Chain



- The validate step uses a formally-verified checker;
- Ideally the encoding step is also formally-verified;
- The other steps can be heavily optimized and unverified.

Reduced, Ordered Binary Decision Diagrams (BDDs)

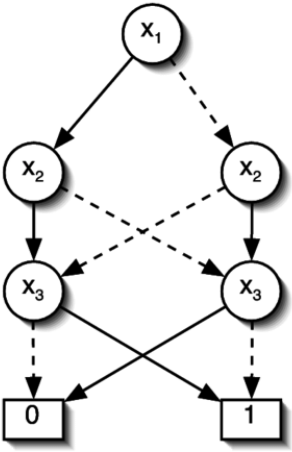
- Bryant, 1986

Representation

- Canonical representation of Boolean function
- Compact for many useful cases

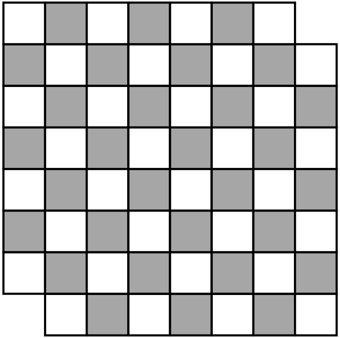
Algorithms

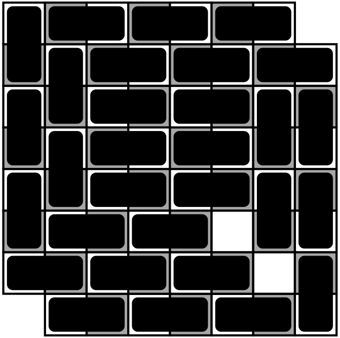
- Apply(f, g, op)
  - op is Boolean operation (e.g., $\wedge$, $\vee$, $\oplus$)
  - BDD representation of $f\ op\ g$
- EQuant(f, X)
  - X set of variables
  - BDD representation of $\exists V f$



Mutilated Chessboards: “A Tough Nut to Crack” [McCarthy]

Can a chessboard be fully covered with dominos after removing two diagonally opposite corner squares?

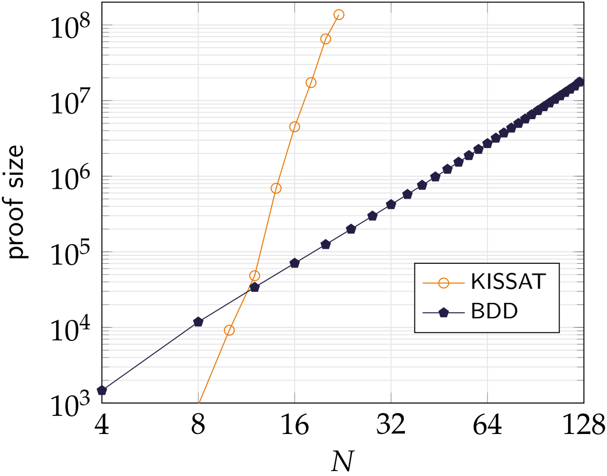




Easy to refute based on the following two observations:

- There are more white squares than black squares; and
- A domino covers exactly one white and one black square.

Compact and Verified Proofs from BDDs [Sinze & Biere '06] [Bryant & Heule '21?]



Mutilated Chessboard: problem size $\sim N^2$, BDD proof size $\sim N^{2.69}$

Same proof format as SAT and same verified checker

Examples of Challenges

Trusted Computing

Parallel Computing

Conclusions and Challenges

Challenge: How to Parallelize Automated Reasoning?

Successes

- Industrial applications, such as equivalence checking;
- Long-standing open math problems resolved; and
- Speedups even with thousands of cores

Challenge: How to Parallelize Automated Reasoning?

Successes

- Industrial applications, such as equivalence checking;
- Long-standing open math problems resolved; and
- Speedups even with thousands of cores

Challenges

- Many memory issues (from cache misses to out of memory);
- Different approaches are effective on different problems; and
- Reduced performance in many tools, when using more cores.

Parallel Computing: SAT Solver Paradigms

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

Strength: Effective on large, "easy" formulas.

Weakness: Hard to parallelize.

Parallel Computing: SAT Solver Paradigms

Conflict-driven clause learning (CDCL): Makes fast decisions and 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 by selecting effective splitting variables via looking ahead.

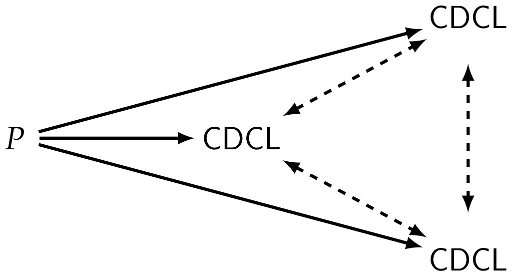
Strength: Effective on small, hard formulas.

Weakness: Expensive.

Parallel Computing: Portfolio Solvers

The most commonly used parallel solving paradigm is portfolio:

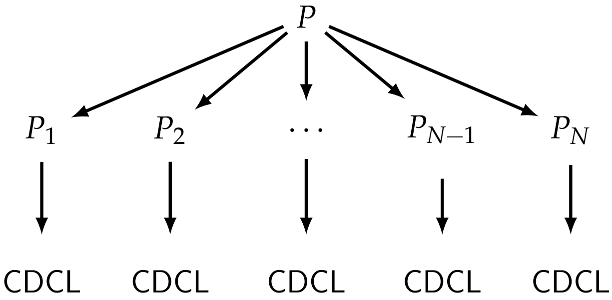
- Run multiple (typically identical) solvers with different configurations on the same formula; and
- Share clauses among the solvers.



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]

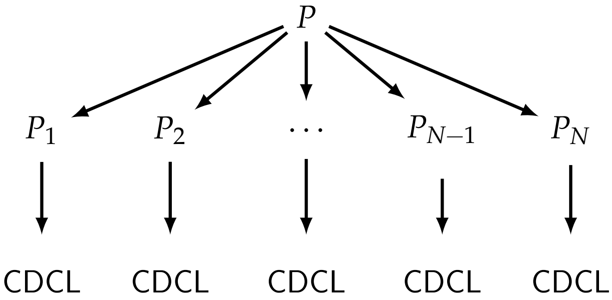
Cube-and-conquer splits a given problem into millions of subproblems that are solved independently by CDCL.



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 recently integrated in Z3

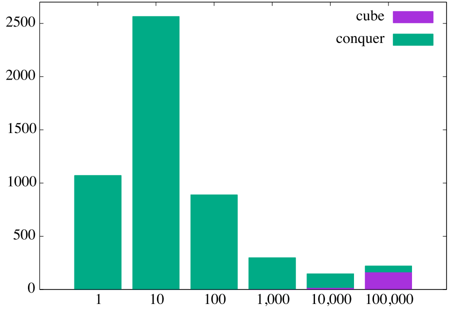
The Hidden Strength of Cube-and-Conquer

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)
- 13 SAT Competitions (2002–)
- 5 SAT Races (2006, 2008, 2010, 2015, 2019)
- 1 SAT Challenge (2012)

Parallel Computing: SAT Competition Cloud Track

Long tradition of SAT competitive events, starting from 1992

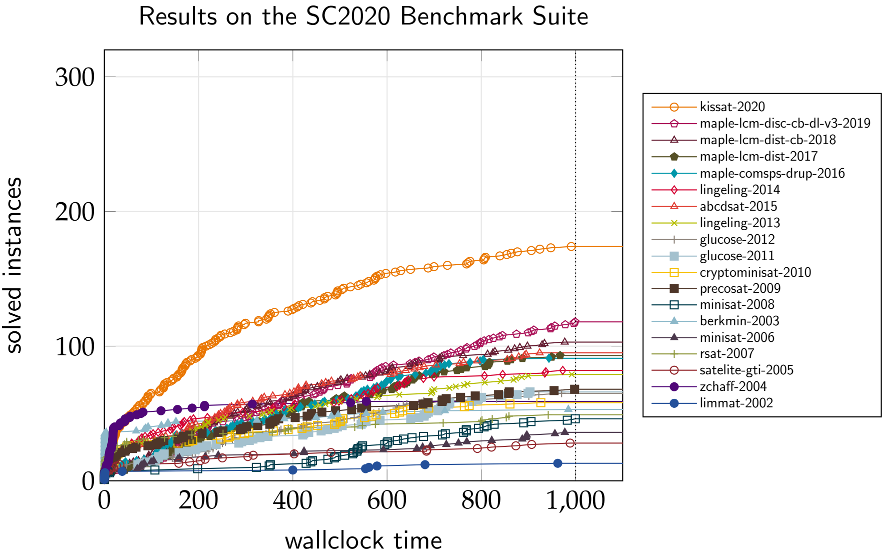
- 3 competitions in the 90s (1992,1993, 1996)
- 13 SAT Competitions (2002–)
- 5 SAT Races (2006, 2008, 2010, 2015, 2019)
- 1 SAT Challenge (2012)

New this year

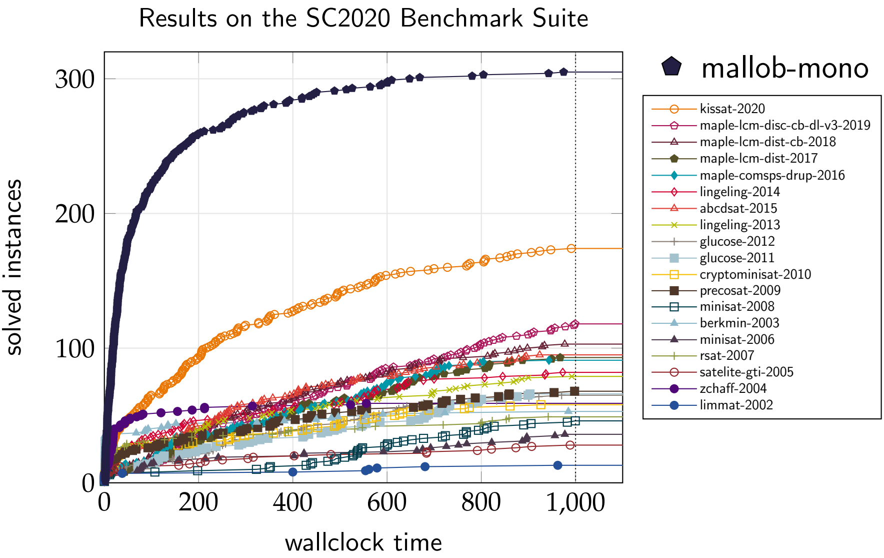
- Cloud Track – evaluate distributed solvers on the Amazon cloud. Solvers are run on 1600 virtual cores for 1000 seconds. Sponsored by Amazon. Participants received AWS credit to develop their solvers.

Winner of the cloud track clearly outperformed sequential winner

Distributed versus Competition Winners



Distributed versus Competition Winners



Parallel Computing: Reasoning in the Cloud

Automated reasoning as a service:

- Solves problems from easy to hard;
- Can provide correctness proofs;
- Explains the solution and/or method.

Joint work with Siemens to fully explore the design space of gearboxes.

Joint work with Amazon Web Services on routing and software verification.

Examples of Challenges

Trusted Computing

Parallel Computing

Conclusions and Challenges

Conclusions and Challenges

We can have full confidence in the correctness of SAT solvers:

- All top-tier solvers emit proof logging (also for re-encoding)
- Formally-verified tools can efficiently certify the proofs

How to lift this success to richer logics (SMT/HWMCC/FOL)?

Conclusions and Challenges

We can have full confidence in the correctness of SAT solvers:

- All top-tier solvers emit proof logging (also for re-encoding)
- Formally-verified tools can efficiently certify the proofs

How to lift this success to richer logics (SMT/HWMCC/FOL)?

Linear speedups are possible on a range of problems

- Even when using 1000s of CPUs;
- And the enormous proofs can be validated in parallel.

Various challenges:

- Make the techniques effective on a broader range of problems
- Expand the potential users: automated reasoning in the cloud
- Explainable automated reasoning to increase understanding

Distributed and Trustworthy Automated Reasoning

Marijn J.H. Heule

Carnegie Mellon University

scholars

High Confidence Software and Systems September 17, 2020