25 years of SAT
Trust SAT Solvers / Proofs

25th International Conference on Theory and Applications of Satisfiability Testing (SAT’22)

Armin Biere
University of Freiburg, Germany

August 4, 2021
Federated Logic Conference (FLOC’22)
Haifa, Israel
60 Years of SAT Solving

- SAT NP complete (1960)
- 1st SAT Competition (1970)
- Tseitin Encoding (1970s)
- WalkSAT (1980s)
- GSAT (1980s)
- DPLL (1960s)
- SAT for Planning (1980s)
- Look Ahead (1980s)
- LBD (1990s)
- CDCL (1990s)
- VSIDS (1990s)
- MiniSAT (1990s)
- SMT (1990s)
- 1st SAT Solvers (1970s)
- Bounded Model Checking (1980s)
- Inprocessing (2000s)
- Cube & Conquer (2000s)
- Proofs (2000s)
- Massively Parallel (2000s)
- QBF working (2000s)
- Handbook of SAT (1st) (2000s)
- SMT & SAT everywhere (2000s)
- Handbook of SAT (2nd) (2000s)
60 Years of SAT Solving

1960
DPLL

1970
SAT NP complete

1980
WalkSAT
GSAT
Tseitin Encoding

1990
CDCL
1st SAT Competition
VSIDS

2000
Portfolio
Bounded Variable Elimination
Bounded Model Checking

2010
Phase Saving
ProbSAT
Avatar

2020
Inprocessing
Cube & Conquer
SAT & SMT everywhere

Handbook of SAT (1st)
Handbook of SAT (2nd)

 Donald Knuth

Proofs

LBD
VSIDS
MiniSAT
LBD

SMT

Arithmetic Solvers

QBF working

Massively Parallel

DPLL

Competition

Look Ahead

SAT for Planning

SAT & SMT everywhere
60 Years of SAT Solving

- Look Ahead
- GSAT
- WalkSAT
- DP
- Portfolio Phase
- Saving
- Planning
- SAT for Inprocessing
- Bounded Variable Elimination
- SAT Chapter
- Donald Knuth
- QBF
- SAT & SMT everywhere
- SAT & SMT working
- Massively Parallel
- Handbooks of SAT (1st) & (2nd)
- SAT Solvers
- Arithmetic Solvers
- LBD
- Proofreading
- VSIDS
- MiniSAT
- Cube & Conquer
- Tseitin Encoding
- SMT
- CDCL
- WalkSAT
- GSAT
- Bounded Model Checking
- Inprocessing
- Competition
- Proofs
- Avatar
- 1st SAT
- 2010
- Phase Saving
200 TB Biggest Math Proof Ever

HeuleKullmannMarek-SAT16 best paper

https://www.cs.utexas.edu/~marijn/ptn

color the natural numbers $\mathbb{N}$ with two colors $\{\bullet, \cdot\}$, such that all equations

$$3^2 + 4^2 = 5^2$$

are not monochromatic?

$$(x_3 \lor x_4 \lor x_5) \land (\bar{x}_3 \lor \bar{x}_4 \lor \bar{x}_5) \land$$

$$(x_5 \lor x_{12} \lor x_{13}) \land (\bar{x}_5 \lor \bar{x}_{12} \lor \bar{x}_{13}) \land$$

$$(x_6 \lor x_8 \lor x_{10}) \land (\bar{x}_6 \lor \bar{x}_8 \lor \bar{x}_{10}) \land$$

...

p cnf 7820 18930
3 4 5 0
-3 -4 -5 0
5 12 13 0
-5 -12 -13 0
...
5412 5635 7813 0
-5412 -5635 -7813 0
5474 5520 7774 0
-5474 -5520 -7774 0

Yes, for [1..7824] $\Rightarrow$ SAT
No, for [1..7825] $\Rightarrow$ UNSAT
Two-hundred-terabyte maths proof is largest ever

Evelyn Lamb

Published: 26 May 2016

Nature 534, 17–18 (2016) | Cite this article

1655 Accesses | 4 Citations | 1012 Altmetric | Metrics

A computer cracks the Boolean Pythagorean triples problem – but is it really maths?

The University of Texas's Stampede supercomputer, on which the 200-terabyte maths proof was solved. Credit: University of Texas

Three computer scientists have announced the largest-ever mathematics proof: a file that comes in at a whopping 200 terabytes\(^1\), roughly equivalent to all the digitized text held by the US Library of Congress. The researchers have created a 68-gigabyte compressed version of their solution – which would allow anyone with about 30,000 hours of spare processor time to download, reconstruct and verify it – but a human could never hope to read through it.
The Science of Brute Force

By Marilj J. H. Haule, Oliver Kullmann
Communications of the ACM, August 2017, Vol. 60 No. 8, Pages 70-79
10.1145/3107239

Recent progress in automated reasoning and super-computing gives rise to a new era of brute force. The game changer is “SAT,” a disruptively, brute-reasoning technology in industry and science. We illustrate its strength and potential via the proof of the Boolean Pythagorean Triples Problem, a long-standing open problem in Ramsey Theory. This 200TB proof has been constructed completely automatically, paradoxically, in an ingenious way. We welcome these bold new proofs emerging on the horizon, beyond human understanding—both mathematics and industry need them.

Key Insights

- Long-standing open problems in mathematics can now be solved completely automatically resulting in clever though potentially gigantic proofs.
- Our time requires answers to hard questions regarding safety and security. In these cases knowledge is more important than understanding as long as we can trust the answers.
- Powerful SAT-solving heuristics facilitate linear speedups even when using thousands of cores. Combined with the ever-increasing capabilities of high-performance computing clusters they enable solving challenging problems.

Many relevant search problems, from artificial intelligence to combinatorics, explore large search spaces to determine the presence or absence of a certain object. These problems are hard due to combinatorial explosion, and have traditionally been called infeasible. The brute-force method, which at least implicitly explores all possibilities, is a general approach to systematically search through such spaces.

Brute force has long been regarded as suitable only for simple problems. This has changed in the last two decades, due to the progress in Satisfiability (SAT) solving, which by adding brute reason renders brute force into a powerful approach to deal with many problems easily and automatically. Search spaces with far more possibilities than the number of particles in the universe may be completely explored.
Overview

- Applications
  - (Big) Math Proofs!
  - Cores, Diagnosis, MUS, Interpolation, …
  - Reasoning about Implementation, Debugging, Certification

- Formats
  - Resolution, Traces, clause deletion, with and without IDs
  - RUP, RAT, DRUP, DRAT, LRAT, PR, …

- Theory
  - blocked clauses, proof systems / complexity

- Alternatives
  - testing, fuzzing, model based testing

- Extensions
  - algebraic proof systems, Cutting Planes, QBF, etc.

result of collecting and (re)reading 87 papers!
Applications: (Big) Math Proofs

- van der Waerden numbers [DramsfieldMarekTruszcunski-SAT03]
  but no certificates yet

- Erdős Discrepancy Conjecture [KonevLisista-SAT14]

- Pythagorean Triples [HeuleKullmannMarek-SAT16] 200 TB

- Schur number 5 [Heule-AAAI18] 4 PB

- … many more recent papers by Marijn Heule et. al.
Applications: Cores, Diagnosis, MUS, Interpolation

- diagnosis of infeasibility
  - core of original clauses used in proof as explanation
  - quantifier elimination [McMillanAmla-CAV02, VizelRyvchinNadel-CAV13]
  - proof based CEGAR in model checking [McMillanAmla-TACAS03]
    - GuptaGanaiYangAshar-ICCAD03, AmlaMcMillan-FMCAD04, EenMishchenkoAmla-FMCAD10
- proof-based MUS extraction [DerschowitzHannahNadel-SAT06, Nadel-FMCAD10]
- generating interpolants from proofs
  - seminal paper on interpolation based model checking [McMillan-CAV03]
  - highly influential
    - many papers on extensions to SMT and first-order logic
    - many further applications beside model checking
  - extract interpolants from actual proofs [GurfinkelVizel-FMCAD14]
  - hard for modern RAT proofs [RebolaPardoWeissenbacher-LPAR20]
  - further technical issues of actual proofs
    - e.g., handling units [RebolaPardoBiere-POS18]
Applications: Testing, Debugging, Certification

- SAT solver development
  - use internal or external proof checkers
  - check proofs during regression or other forms of testing
  - combine with delta-debugging [BrummayerLonsingBiere-SAT10]
  - forward checking better here vs. backward checking (in core analysis)

- certification of actual runs
  - mandatory in certain industrial applications (railway interlocking systems, ...)
  - substantially increases trusts for very long runs (big math proofs)
  - can be used to migrate solver state too [BiereChowdhuryHeuleKieslWhalen-SAT22]

- makes sure solvers in competition do not take unsound “short-cuts”
### CNF Trace

<table>
<thead>
<tr>
<th>CNF</th>
<th>trace</th>
<th>extended trace</th>
<th>resolution trace</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>p cnf 3 8</code></td>
<td><code>-1 -2 -3 0</code></td>
<td><code>1 -2 -3 -1 0 0</code></td>
<td><code>1 -1 -3 -2 0 0</code></td>
</tr>
<tr>
<td></td>
<td><code>-1 -2 3 0</code></td>
<td><code>2 -2 3 -1 0 0</code></td>
<td><code>2 -1 3 -2 0 0</code></td>
</tr>
<tr>
<td></td>
<td><code>-1 2 -3 0</code></td>
<td><code>3 2 -3 -1 0 0</code></td>
<td><code>3 2 -1 -3 0 0</code></td>
</tr>
<tr>
<td></td>
<td><code>-1 2 3 0</code></td>
<td><code>4 2 3 -1 0 0</code></td>
<td><code>4 2 -1 3 0 0</code></td>
</tr>
<tr>
<td></td>
<td><code>1 -2 -3 0</code></td>
<td><code>5 1 -3 -2 0 0</code></td>
<td><code>5 -2 -3 1 0 0</code></td>
</tr>
<tr>
<td></td>
<td><code>1 -2 3 0</code></td>
<td><code>6 1 3 -2 0 0</code></td>
<td><code>6 -2 3 1 0 0</code></td>
</tr>
<tr>
<td></td>
<td><code>1 2 -3 0</code></td>
<td><code>7 1 -3 2 0 0</code></td>
<td><code>7 1 -3 2 0 0</code></td>
</tr>
<tr>
<td></td>
<td><code>1 2 3 0</code></td>
<td><code>8 1 3 2 0 0</code></td>
<td><code>8 1 3 2 0 0</code></td>
</tr>
<tr>
<td></td>
<td><code>9 * 7 8 0</code></td>
<td><code>9 1 2 0 7 8 0</code></td>
<td><code>9 1 2 0 7 8 0</code></td>
</tr>
<tr>
<td></td>
<td><code>10 * 9 5 6 0</code></td>
<td><code>10 1 0 9 5 6 0</code></td>
<td><code>10 -2 1 0 5 6 0</code></td>
</tr>
<tr>
<td></td>
<td><code>11 * 1 10 2 0</code></td>
<td><code>11 -2 0 1 10 2 0</code></td>
<td><code>11 1 0 10 9 0</code></td>
</tr>
<tr>
<td></td>
<td><code>12 * 10 11 4 0</code></td>
<td><code>12 3 0 10 11 4 0</code></td>
<td><code>12 -1 -2 0 1 2 0</code></td>
</tr>
<tr>
<td></td>
<td><code>13 * 10 11 3 12 0</code></td>
<td><code>13 0 10 11 3 12 0</code></td>
<td><code>13 -2 0 12 11 0</code></td>
</tr>
</tbody>
</table>

`picosat -t`  `picosat -T`  `tracecheck -B`
### Clausal Proofs

*GoldbergNovikov-DATE03*  *VanGelder-ISAIM08*

*HeuleHuntWetzler-CADE13*  *HeuleHuntWetzler-FMCAD13*  *WetzlerHeuleHunt-SAT14*

*CruzFilipeMarquesSilvaSchneiderKamp-TACAS17*  *BaekCarneiroHeule-TACAS21*

<table>
<thead>
<tr>
<th>RUP</th>
<th>DRUP/DRAT</th>
<th>LRAT</th>
<th>FRAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>p cnf 3 8</td>
<td>-1 -2 -3 0</td>
<td>(1)</td>
<td>...</td>
</tr>
<tr>
<td>-1 -2 3 0</td>
<td>(2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-1 2 -3 0</td>
<td>original clauses</td>
<td>(3)</td>
<td></td>
</tr>
<tr>
<td>-1 2 3 0</td>
<td>not part of proof</td>
<td>(4)</td>
<td></td>
</tr>
<tr>
<td>1 -2 -3 0</td>
<td>(5)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 -2 3 0</td>
<td>(6)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 2 -3 0</td>
<td>(7)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 2 3 0</td>
<td>(8)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-2 -3 0</td>
<td>-2 -3 0</td>
<td>8 d 0</td>
<td></td>
</tr>
<tr>
<td>-3 0</td>
<td>d 1 -2 -3 0</td>
<td>9 -2 -3 0 1 5 0</td>
<td></td>
</tr>
<tr>
<td>2 0</td>
<td>d -1 -2 -3 0</td>
<td>9 d 1 5 0</td>
<td></td>
</tr>
<tr>
<td>-1 0</td>
<td>-2 3 0</td>
<td>10 -2 3 0 6 2 0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>d 1 -2 3 0</td>
<td>10 d 6 2 0</td>
<td></td>
</tr>
<tr>
<td>d -1 -2 3 0</td>
<td>11 2 -3 0 7 3 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 -3 0</td>
<td>11 d 7 3 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d 1 2 -3 0</td>
<td>12 2 3 0 8 4 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d -1 2 -3 0</td>
<td>12 d 8 4 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 3 0</td>
<td>13 -2 0 10 9 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d 1 2 3 0</td>
<td>13 d 10 9 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d -1 2 3 0</td>
<td>15 0 13 11 12 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-2 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

`cadical`  `cadical -P1`  `drat-trim -L`
Blocked Clauses

Kullmann-DAM99

block clause $C \in F$ all clauses in $F$ with $\bar{l}$

$(a \lor b \lor \ell)$

$(\ell \lor \bar{a} \lor c)$

fix a CNF $F$

$(\bar{\ell} \lor \bar{b} \lor d)$

$(\ldots \lor \ell)$

other clauses with $\ell$ can be ignored

Theorem

If all resolvents of $C$ on $\ell$ are tautological then $C$ is blocked and can be removed.

Proof

Assignment $\sigma$ satisfying $F \setminus C$ but not $C$
can be extended to a satisfying assignment of $F$ by flipping value of $\ell$. 
Encoding vs. Preprocessing

Plaisted−Greenbaum encoding

- VE(PG)
- BCE(PG)
- PL(PG)

CNF−level simplification

- [BCE+VE](PG)
- VE(PG)
- BCE(PG)
- PL(PG)

Circuit−level simplification

- PG(COI)
- PG(MIR)
- PG(NSI)

- COI
- MIR
- NSI

- PG
- TST

Tseitin encoding

- BCE+VE

JarvisaloBiereHeule-TACAS10
Clause Elimination / Redundancy / Preprocessing

- blocked clause elimination started this line of preprocessing work
  - covered clauses
  - globally blocked clauses

- inprocessing rules
  - introduced RAT to formalize what inprocessing CDCL solvers do
  - reconsidered adding redundant clauses, thus rules give a proof system
  - incremental SAT solving extension

\[
\begin{align*}
\frac{\varphi}{} & \cdot \varphi \\
\frac{\varphi}{} & \cdot \varphi
\end{align*}
\]

\[
\begin{array}{cccccc}
\varphi [ \rho ] \sigma & \varphi [ \rho \land C ] \sigma & \varphi [ \rho \land C ] \sigma & \varphi \land C [ \rho ] \sigma & \varphi \land C [ \rho ] \sigma & \varphi \land C [ \rho ] \sigma \\
\text{LEARN} & \text{FORGET} & \text{STRENGTHEN} & \text{WEAKEN} & \text{DROP}
\end{array}
\]

FazekasBiereScholl-SAT19 best student paper
Proofs / RES / RUP / DRUP

- resolution proofs (RES) simple to check but large and hard(er) to produce
- original idea for clausal proofs and checking them:
  - proof traces are sequences of “learned clauses” $C$
  - first check clause through unit propagation $F \vdash \neg C$ then add $C$ to $F$
  - reverse unit implied clauses (RUP) [GoldbergNovikov-DATE03] [VanGelder-ISAIM08]
- tracing deleted clauses too in DRUP gives much faster checking [HeuleHuntWetzler-FMCAD13] [HeuleHuntWetzler-STVR14]
- now DRUP/DRAT mandatory since 2013 to certify UNSAT
- Certification
  - Coq [CruzFilipeMarquesSilvaSchneiderKamp-TACAS17]
  - ACL2 [CruzFilipeHuntKaufmannSchneiderKamp-CADE17]
  - Isabelle [Lammich-CADE17]
Resolution Asymmetric Tautologies (RAT)

“Inprocessing Rules” JarvisaloHeuleBiere-IJCAR12

- more general notion of redundancy criteria to cover what SAT solvers do
- simple extension of blocked clauses:
  replace “resolvents on \( l \) are tautological” by “resolvents on \( l \) are RUP”

\[
\begin{align*}
(a \lor l) & \quad \text{RAT on } l \quad \text{w.r.t.} \quad (l \lor b) \land (a \lor x) \land (\bar{x} \lor b) \land (l \lor c) \land (\bar{a} \lor b) \\
\text{yields resolvent } (a \lor b) \text{ with } F \vdash_1 (a \lor b)
\end{align*}
\]

- deletion information is again essential (DRAT)
- now mandatory in the main track of the SAT competitions since 2013
- pretty powerful: covers symmetry breaking  

HeuleHuntWetzler-CADE15
Redundancy

“Short Proofs Without New Variables”  

Definition. A partial assignment $\alpha$ blocks a clause $C$ if $\alpha$ assigns the literals in $C$ to false (and no other literal). Also written $\alpha = \neg C$.

Definition. A clause $C$ is redundant w.r.t. a formula $F$ if $F$ and $F \cup \{C\}$ are satisfiability equivalent.

Definition. A formula $F$ simplified by a partial assignment $\alpha$ is written as $F|_\alpha$.

Theorem.

Let $F$ be a formula, $C$ a clause, and $\alpha$ the assignment blocked by $C$.

Then $C$ is redundant w.r.t. $F$ iff exists an assignment $\omega$ such that

(i) $\omega$ satisfies $C$  
and

(ii) $F|_\alpha \models F|_\omega$.
Propagation Redundant (PR)

Definition. Clause $C$ propagation redundant (PR) w.r.t. $F$ if exists assignment $\omega$ satisfying $C$ with $F | \alpha \vdash_1 F | \omega$

so in essence replacing “$|$” by “$\vdash_1$” (implied by unit propagation / RUP)

- more general than RAT: short proofs for PHP without new variables
- Satisfaction Driven Clause Learning (SDCL) \[\text{HeuleKieslSeidlBiere-HVC17}\] best paper
  - first automatically generated PR proofs
  - prune assignments for which we have other at least as satisfiable ones
  - filtered positive reduct in SaDiCaL \[\text{HeuleKieslBiere-TACAS19}\] nominated best paper
- translate PR to DRAT \[\text{HeuleBiere-TACAS18}\]
  - only one additional variable needed were shortest DRAT proofs for PHP
- translate DRAT to extended resolution \[\text{KieslRebolaPardoHeule-IJCAR18}\] best paper
- more recent seperation results in \[\text{BussThapen-SAT19}\]
Landscape of Clausal Redundancy

\[ F \models_\alpha F \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]

\[ F \models_\alpha \top \]

\[ F \models_\alpha \bot \]

\[ F \models_\alpha \epsilon \]
CDCL(formula $F$)

1. $\alpha := \emptyset$

2. forever do
   3. $\alpha := \text{UnitPropagate}(F, \alpha)$
   4. if $\alpha$ falsifies a clause in $F$ then
   5.     $C := \text{AnalyzeConflict}()$
   6.     $F := F \land C$
   7.     if $C$ is the empty clause $\bot$ then return UNSAT
   8.     $\alpha := \text{BackJump}(C, \alpha)$

else

13. if all variables are assigned then return SAT
14. $l := \text{Decide}()$
15. $\alpha := \alpha \cup \{l\}$
*SDCL* (formula $F$)

1. $\alpha := \emptyset$
2. **forever do**
3.  $\alpha := \text{UnitPropagate}(F, \alpha)$
4.  **if** $\alpha$ falsifies a clause in $F$ **then**
5.    $C := \text{AnalyzeConflict}()$
6.    $F := F \land C$
7.    **if** $C$ is the empty clause $\bot$ **then** return UNSAT
8.  $\alpha := \text{BackJump}(C, \alpha)$
9. **else if** the pruning predicate $P_{\alpha}(F)$ is satisfiable **then**
10.   $C := \text{AnalyzeWitness}()$
11.   $F := F \land C$
12.   $\alpha := \text{BackJump}(C, \alpha)$
13. **else**
14.    **if** all variables are assigned **then** return SAT
15.    $l := \text{Decide}()$
16.    $\alpha := \alpha \cup \{l\}$
Mutilated Chessboard

CDCL

SDCL
In the positive reduct consider all clauses satisfied by $\alpha$ but remove unassigned literals and add $C$:

**Definition.** Let $F$ be a formula and $\alpha$ an assignment. Then, the **positive reduct** of $F$ and $\alpha$ is the formula $G \land C$ where $C$ is the clause that blocks $\alpha$ and $G = \{\text{touched}_\alpha(D) \mid D \in F \text{ and } D |_{\alpha} = \top\}$.

**Theorem.** Let $F$ be a formula, $\alpha$ an assignment, and $C$ the clause that blocks $\alpha$. Then, $C$ is **SBC** by an $L \subseteq C$ with respect to $F$ if and only if the assignment $\alpha_L$ satisfies the positive reduct.

We obtain the **filtered positive reduct** by not taking all satisfied clauses of $F$ but only those for which the untouched part is not implied by $F |_{\alpha}$ via unit propagation:

**Definition.** Let $F$ be a formula and $\alpha$ an assignment. Then, the **filtered positive reduct** of $F$ and $\alpha$ is the formula $G \land C$ where $C$ is the clause that blocks $\alpha$ and $G = \{\text{touched}_\alpha(D) \mid D \in F \text{ and } F |_{\alpha} \not\vdash \text{untouched}_\alpha(D)\}$.

**Theorem.** Let $F$ be a formula, $\alpha$ an assignment, and $C$ the clause that blocks $\alpha$. Then, $C$ is **SPR** by an $L \subseteq C$ with respect to $F$ if and only if the assignment $\alpha_L$ satisfies the filtered positive reduct.

where SPR extends SBC by propagation as RAT extends BC.
Alternatives

- better testing and debugging
  - random CNFs (fuzzing [BrummayerLonsingBiere-SAT10](#))
  - fuzzing of incremental use (model based testing [ArthoBiereSeidl-TAP13](#))
  - similar techniques for SMT, QBF, ASP, …
    - [BrummayerBiere-SMT09](#) [NiemetzPreinerBiere-SMT17](#) …
  - check UNSAT/SAT discrepancy against model (Sam Buss)

- formally verified SAT solvers
  - rather challenging, requires usually implementation from scratch
    - lagging behind state-of-the-art in terms of performance
  - what is actually verified?
    - soundness, completeness, termination, actual code, …

- most advanced: ISASAT by Mathias Fleury
  - [BlanchetteFleuryWeidenbach-IJCAR16](#) best paper
Extensions

- QRAT for QBF [HeuleSeidlBiere-IJCAR14]
- first-order theorem proving [KieslSudaSeidlTompitsBiere-LPAR21]
- used as building block for SMT proofs [OzdemirNiemetzPreinerZoharBarrett-SAT19]
- practical algebraic proofs [RitircBiereKauers-SCSC18, KaufmannBiereKauers-Vampire19, KaufmannBiere-CASC20, KaufmannBiereKauers-DATE20, KaufmannFleuryBiereKauers-FMSD, KaufmannBeameBiereNordstrom-DATE22]
- (practical) trusted BDD-based SAT solving [SinzBiere-CSR06, JussilaSinzBiere-SAT06, BryantHeule-TACAS21, BarnettBiere-CADE21, BryantBiereHeule-TACAS22, Bryant-POS22, SoosBryant-POS22]
- certifying model checking [YuBiereHeljanko-CAV21, YuFroleyksBiereHeljanko-FMCAD22]
Parallel

With better parallel SAT solvers we need parallel proof generation and checking!

- Marijn’s big math proofs actually produced in parallel
  Cube-and-conquer \texttt{HeuleKullmannWieringaBiere-HVC11} best paper

- compositional propositional proofs \texttt{HeuleBiere-LPAR15} theory

- parallel proof generation \texttt{FleuryBiere-POS22} multi-threaded

- parallel proof checking \texttt{Lammich-CADE17} parallel GRAT generator

- also for distributed solving (cloud) \texttt{BiereChowdhuryHeuleKieslWhalen-SAT22}

- also for GPU \texttt{Osama-Dissertation22} \texttt{OsamaWijsBiere-FMSD}
Conclusion

2003  proofs improving testing and debugging SAT solvers

2003  model checking applications required proofs

2007  proof tracks in the SAT competition

2010  redundancy notion triggered by clause elimination procedures

2012  proofs to formalize and reason about inprocessing

2013  proofs in main track of SAT competition (deletion)

2014  math proofs with SAT solvers

2016  checked big math proofs

2017  certified checkers

now  extensions and more applications