25 years of SAT

Trusting SAT Solvers / Proofs

25th International Conference on

Theory and Applications of Satisfiability Testing (SAT'22)

Armin Biere

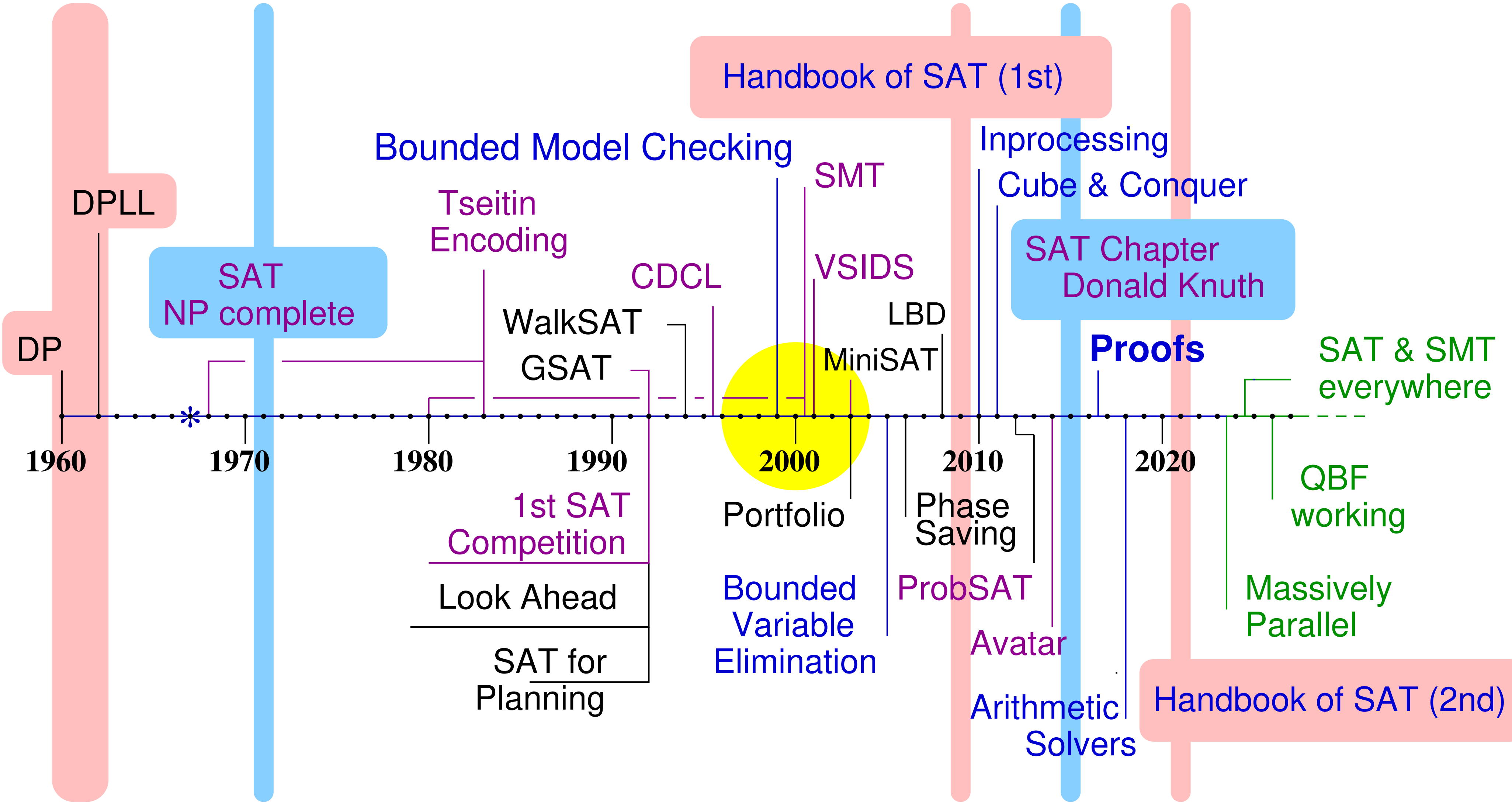
University of Freiburg, Germany

August 4, 2021

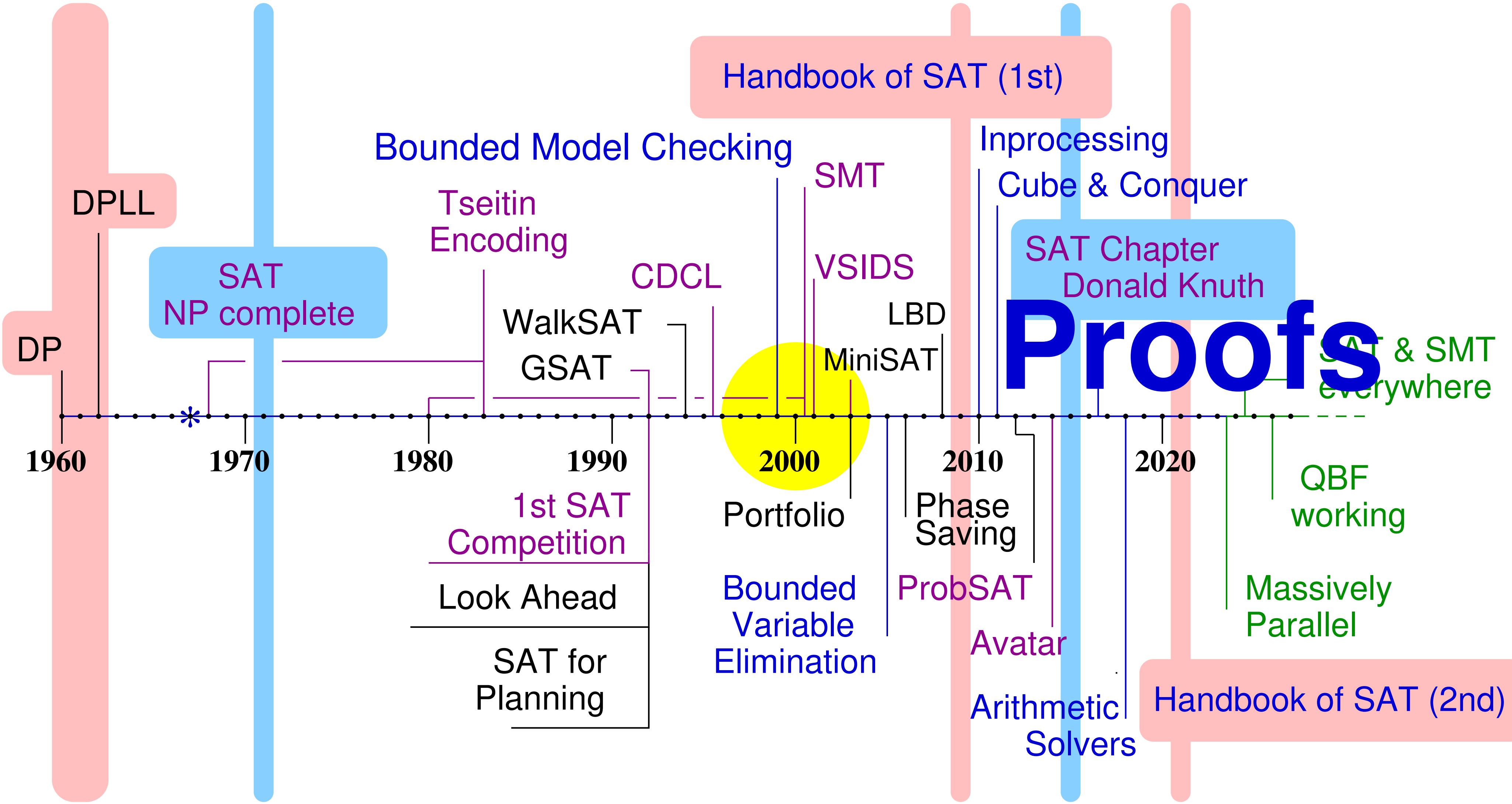
Federated Logic Conference (FLOC'22)

Haifa, Israel

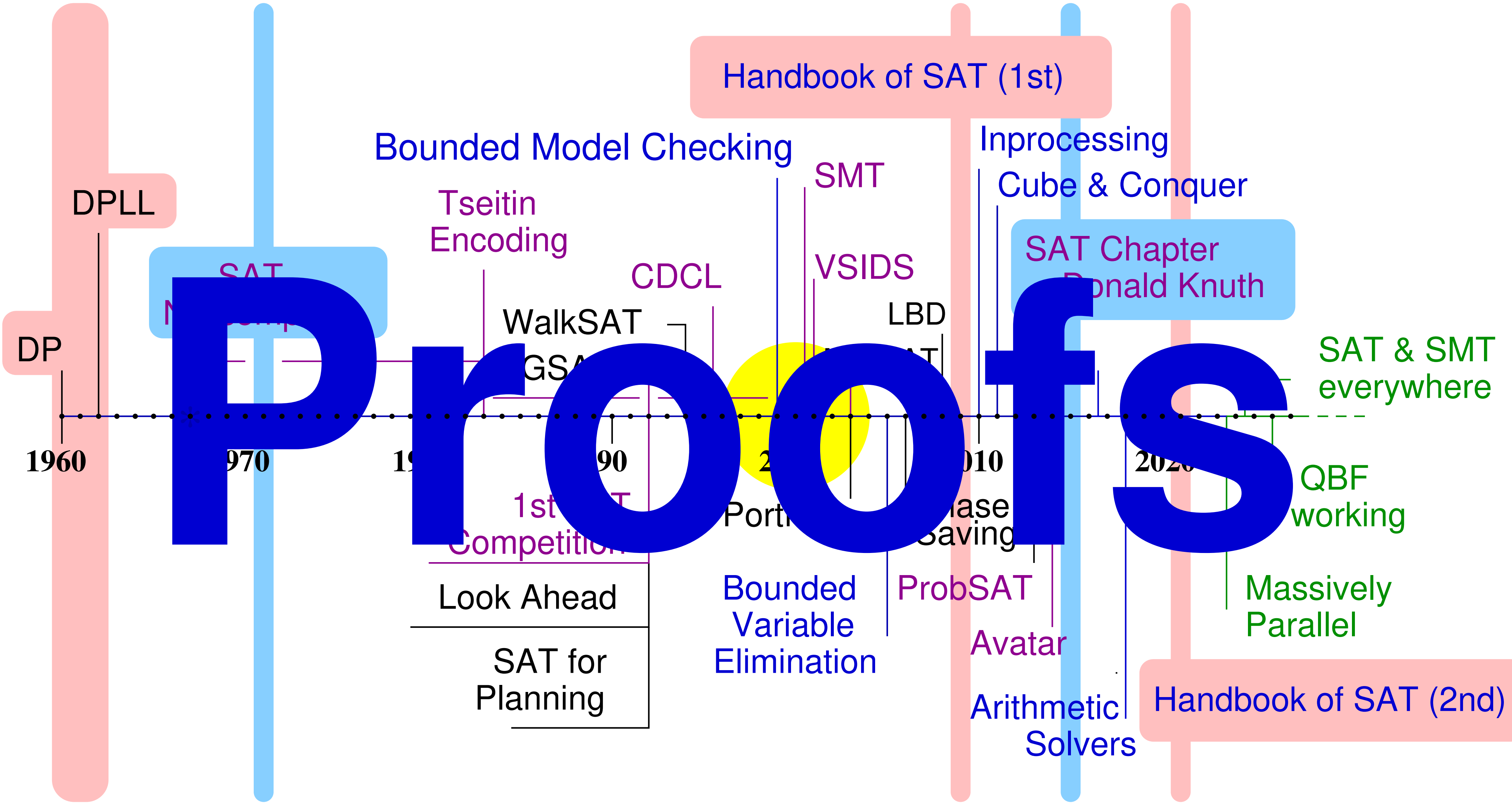
60 Years of SAT Solving



60 Years of SAT Solving



60 Years of SAT Solving



200 TB Biggest Math Proof Ever *

HeuleKullmannMarek-SAT16 best paper https://www.cs.utexas.edu/~marijn/ptn

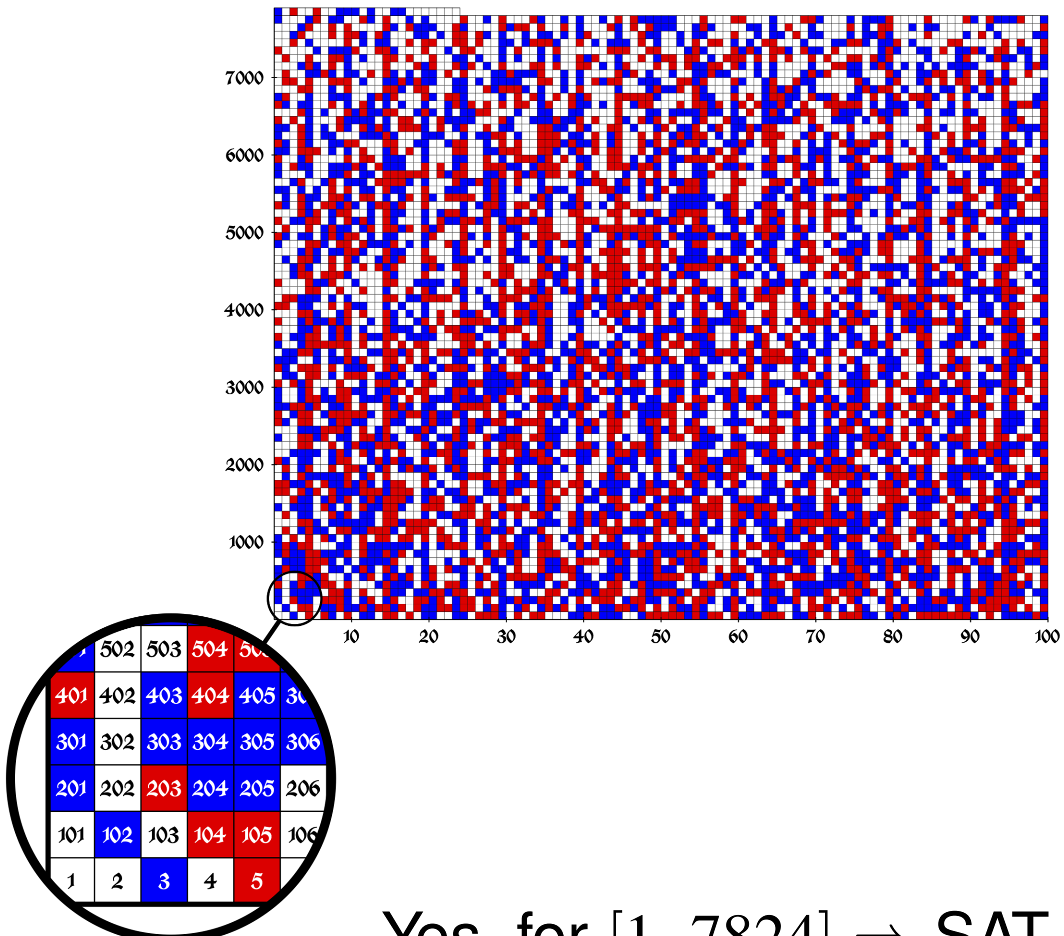
color the natural numbers $\mathbb{N}$ with two colors $\{\bullet, \bullet\}$, such that all equations

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

are not monochromatic?

$$(x_3 \vee x_4 \vee x_5) \wedge (\bar{x}_3 \vee \bar{x}_4 \vee \bar{x}_5) \wedge$$
$$(x_5 \vee x_{12} \vee x_{13}) \wedge (\bar{x}_5 \vee \bar{x}_{12} \vee \bar{x}_{13}) \wedge$$
$$(x_6 \vee x_8 \vee x_{10}) \wedge (\bar{x}_6 \vee \bar{x}_8 \vee \bar{x}_{10}) \wedge$$
$$\ldots$$

```
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

Published: 26 May 2016

Two-hundred-terabyte maths proof is largest ever

Evelyn Lamb

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

Download PDF

Sections References

References
Additional information
Related links
Rights and permissions
About this article
Further reading

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.

REVIEW ARTICLES

The Science of Brute Force

By Marijn J. H. Heule, Oliver Kullmann
Communications of the ACM, August 2017, Vol. 60 No. 8, Pages 70-79
10.1145/3107239
Comments (1)

VIEW AS: SHARE:

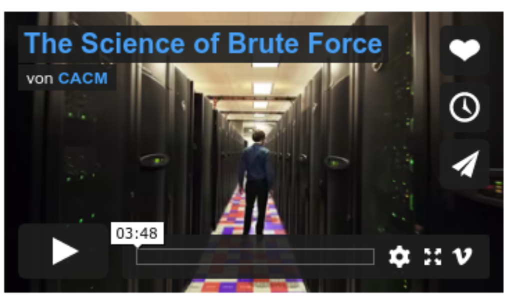
Credit: Peter Crowther Associates

Recent progress in automated reasoning and super-computing gives rise to a new era of brute force. The game changer is "SAT," a disruptive, 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 automaticallyparadoxically, in an ingenious way. We welcome these bold new proofs emerging on the horizon, beyond human understandingboth mathematics and industry need them.

Back to Top

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.

SIGN IN for Full Access
User Name
Password
» Forgot Password?
» Create an ACM Web Account
SIGN IN

ARTICLE CONTENTS:

Introduction
Key Insights
The Rise of Brute Force
The Art of SAT Solving
Proofs of Unsatisfiability
Ramsey Theory and Complexity
Brute Force Formal Methods
Alien Truths
Conclusion
References
Authors
Footnotes

MORE NEWS & OPINIONS
Congress Mandates Technologies to Stop Drunk Driving
Associated Press
How AI Is Reinventing What Computers Are
MIT Technology Review
What Should be Done About Facebook?
Jason Hong

Overview

HeuleBiere-APPA15 Heule-SAT-Handbook21

- 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 DramsfieldMarekTruszczunski-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”

Formats: Proofs with Clause IDs

ZhangMalik-DATE03 Biere-JSAT08

CNF	trace	extended trace	resolution trace
p cnf 3 8			
-1 -2 -3 0	1 -2 -3 -1 0 0	1 -2 -3 -1 0 0	1 -1 -3 -2 0 0
-1 -2 3 0	2 -2 3 -1 0 0	2 -2 3 -1 0 0	2 -1 3 -2 0 0
-1 2 -3 0	3 2 -3 -1 0 0	3 2 -3 -1 0 0	3 2 -1 -3 0 0
-1 2 3 0	4 2 3 -1 0 0	4 2 3 -1 0 0	4 2 -1 3 0 0
1 -2 -3 0	5 1 -3 -2 0 0	5 1 -3 -2 0 0	5 -2 -3 1 0 0
1 -2 3 0	6 1 3 -2 0 0	6 1 3 -2 0 0	6 -2 3 1 0 0
1 2 -3 0	7 1 -3 2 0 0	7 1 -3 2 0 0	7 1 -3 2 0 0
1 2 3 0	8 1 3 2 0 0	8 1 3 2 0 0	8 1 3 2 0 0
	9 * 7 8 0	9 1 2 0 7 8 0	9 1 2 0 7 8 0
	10 * 9 5 6 0	10 1 0 9 5 6 0	10 -2 1 0 5 6 0
	11 * 1 10 2 0	11 -2 0 1 10 2 0	11 1 0 10 9 0
	12 * 10 11 4 0	12 3 0 10 11 4 0	12 -1 -2 0 1 2 0
	13 * 10 11 3 12 0	13 0 10 11 3 12 0	13 -2 0 12 11 0
			14 2 3 0 11 4 0
			15 3 0 14 13 0
			16 2 -3 0 11 3 0
			17 -3 0 16 13 0
			18 0 17 15 0
	picosat -t	picosat -T	tracecheck -B

Clausal Proofs

GoldbergNovikov-DATE03 VanGelder-ISAIM08
HeuleHuntWetzler-CADE13 HeuleHuntWetzler-FMCAD13 WetzlerHeuleHunt-SAT14
CruzFilipeMarquesSilvaSchneiderKamp-TACAS17 BaekCarneiroHeule-TACAS21

	RUP	DRUP/DRAT	LRAT	FRAT
p cnf 3 8				
-1 -2 -3 0			(1)	...
-1 -2 3 0			(2)	
-1 2 -3 0		original clauses	(3)	
-1 2 3 0		not part of proof	(4)	
1 -2 -3 0			(5)	
1 -2 3 0			(6)	
1 2 -3 0			(7)	
1 2 3 0			(8)	
	-2 -3 0	-2 -3 0	8 d 0	
	-3 0	d 1 -2 -3 0	9 -2 -3 0 1 5 0	
	2 0	d -1 -2 -3 0	9 d 1 5 0	
	-1 0	-2 3 0	10 -2 3 0 6 2 0	
	0	d 1 -2 3 0	10 d 6 2 0	
		d -1 -2 3 0	11 2 -3 0 7 3 0	
		2 -3 0	11 d 7 3 0	
		d 1 2 -3 0	12 2 3 0 8 4 0	
		d -1 2 -3 0	12 d 8 4 0	
		2 3 0	13 -2 0 10 9 0	
		d 1 2 3 0	13 d 10 9 0	
		d -1 2 3 0	15 0 13 11 12 0	
		-2 0		
		0		
	cadical	cadical -P1	drat-trim -L	

Blocked Clauses

Kullmann-DAM99

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

$(a \vee b \vee \ell)$ $(\bar{\ell} \vee \bar{a} \vee c)$

fix a CNF F

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

$(\ldots \vee \ell)$

other clauses with ℓ can be ignored

Theorem

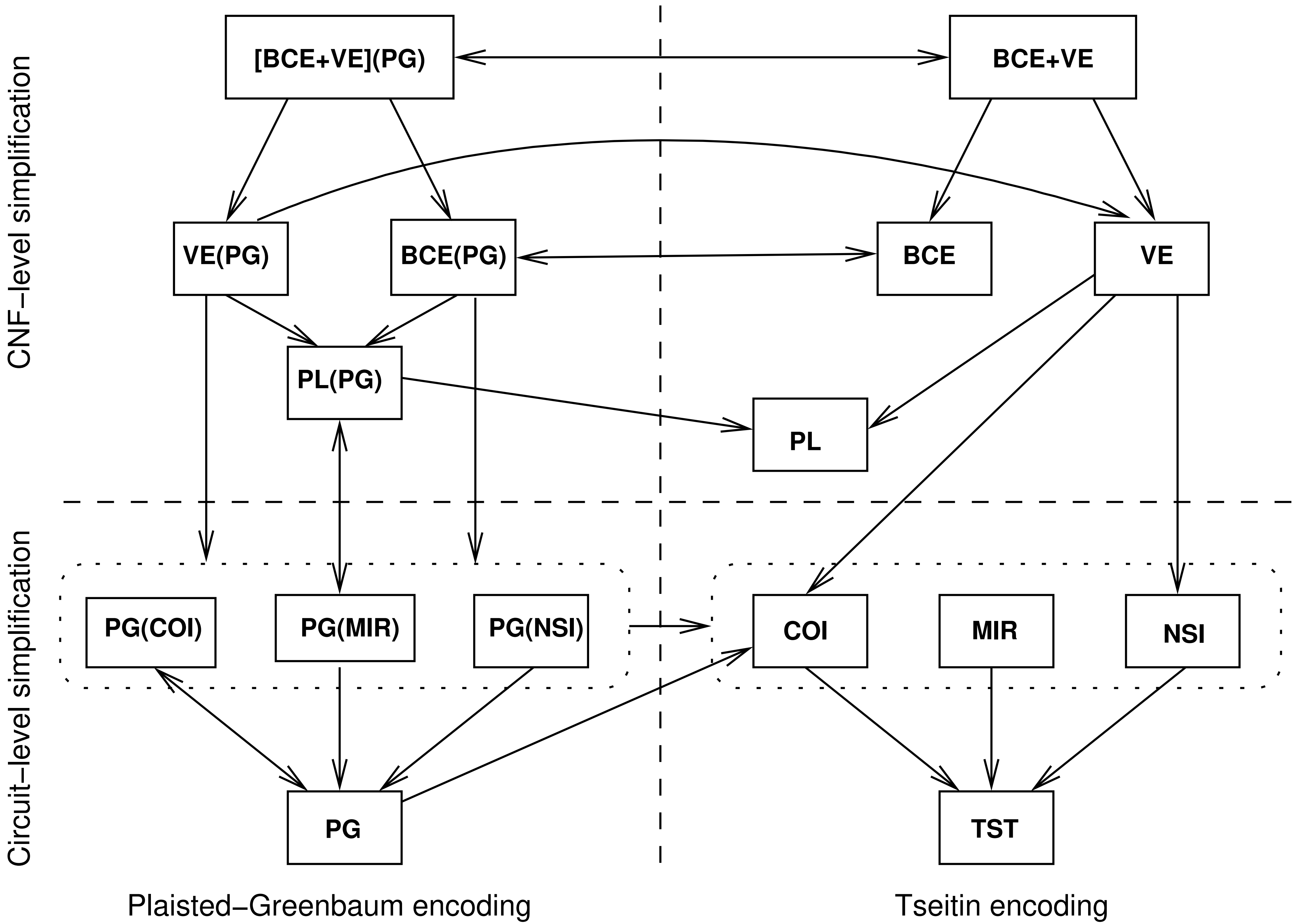
If all resolvents of C on ℓ are tautological then C is blocked and can be removed.

Proof

Assignment σ satisfying $F \backslash C$ but not C
can be extended to a satisfying assignment of F by flipping value of ℓ.

Encoding vs. Preprocessing

JarvisaloBiereHeule-TACAS10



Clause Elimination / Redundancy / Preprocessing

JarvisaloBiereHeule-TACAS10 JarvisaloBiereHeule-JAR12 HeuleJarvisaloBiere-LPAR10
HeuleJarvisaloLonsingSeidlBiere-JAIR15 IJCAI-JAIR 2019 Award
BiereJarvisaloKiesl-SAT-Handbook-2021-Preprocessing-Chapter-Manuscript

- blocked clause elimination started this line of preprocessing work
  - covered clauses HeuleJarvisaloBiere-LPAR10-short BarnettCernaBiere-IJCAR20
  - globally blocked clauses KieslHeuleBiere-ATVA19

- inprocessing rules JarvisaloBiereHeule-JAR12 $\phi\,[\rho]\,\sigma$
  - introduced RAT to formalize what inprocessing CDCL solvers do
  - reconsidered adding <u>redundant</u> clauses, thus rules give a **proof system**
  - incremental SAT solving extension FazekasBiereScholl-SAT19 best student paper

$$\frac{\varphi\ [\,\rho\,]\ \sigma}{\varphi\ [\,\rho \wedge C\,]\ \sigma}\ \boxed{\star} \qquad \frac{\varphi\ [\,\rho \wedge C\,]\ \sigma}{\varphi\ [\,\rho\,]\ \sigma} \qquad \frac{\varphi\ [\,\rho \wedge C\,]\ \sigma}{\varphi \wedge C\ [\,\rho\,]\ \sigma} \qquad \frac{\varphi \wedge C\ [\,\rho\,]\ \sigma}{\varphi\ [\,\rho\,]\ \sigma \cdot (\omega : C)}\ \boxed{\flat} \qquad \frac{\varphi \wedge C\ [\,\rho\,]\ \sigma}{\varphi\ [\,\rho\,]\ \sigma}\ \boxed{\varnothing}$$

LEARN FORGET STRENGTHEN WEAKEN DROP

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 $\boxed{F \vdash_1 C}$ then add C to F
  - reverse unit implied clauses (RUP) GoldbergNovikov-DATE03 VanGelder-ISAIM08

- tracing deleted clauses too in **D**RUP gives much faster checking
  HeuleHuntWetzler-FMCAD13 HeuleHuntWetzler-STVR14

- RUP/RES tracks SAT Competion 2007, 2009, 2011

- now DRUP/DRAT mandatory since 2013 to certify UNSAT

- Certification
  - Coq CruzFilipeMarquesSilvaSchneiderKamp-TACAS17
  - ACL2 CruzFilipeHuntKaufmannSchneiderKamp-CADE17
  - Isabelle Lammich-CADE17

D

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 $\boxed{l}$ are tautological" by "resolvents on $\boxed{l}$ are RUP"

$$(a \vee \boxed{l}) \quad \text{RAT on } l \quad \text{w.r.t.} \quad \overbrace{\underbrace{(\bar{l} \vee b)} \wedge \boxed{(a \vee x) \wedge (\bar{x} \vee b)} \wedge (l \vee c) \wedge (\bar{a} \vee b)}^{F}$$

yields resolvent $(a \vee b)$ with $F \vdash_1 (a \vee b)$

- 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" HeuleKieslBiere-CADE17 best paper

Definition. A partial assignment α blocks a clause C if α 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 α is written as $F|\alpha$.

Theorem.

Let F be a formula, C a clause, and α the assignment blocked by C.

Then C is redundant w.r.t. F iff exists an assignment ω such that

(i) ω satisfies C and (ii) $F|\alpha \models F|\omega$.

Propagation Redundant (PR)

HeuleKieslBiere-CADE17 HeuleKieslBiere-JAR20 KieslRebolaPardoHeuleBiere-JAR20

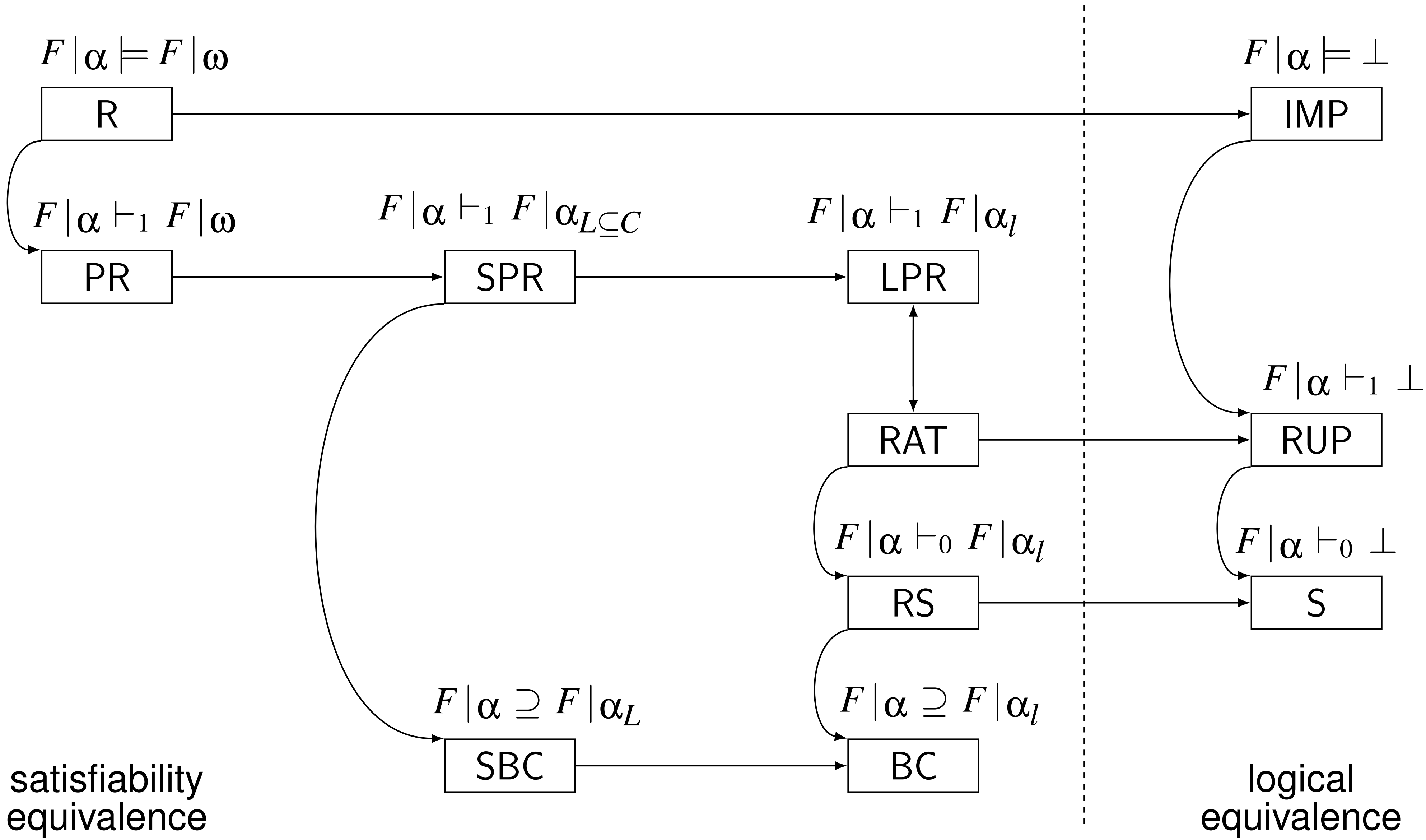
Definition. Clause C propagation redundant (PR) w.r.t. F if exists assignment ω satisfying C with $F|\alpha \;\vdash_1\; F|\omega$

so in essence replacing "$\models$" by "$\vdash_1$" (implied by unit propagation / RUP)

- more general than RAT: short proofs for PHP without new variables
- Satisfaction Driven Clause Learning (SDCL) 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 HeuleKieslBiere-TACAS19 nominated best paper
- translate PR to DRAT HeuleBiere-TACAS18
  - only one additional variable needed were shortest DRAT proofs for PHP
- translate DRAT to extended resolution KieslRebolaPardoHeule-IJCAR18 best paper
- more recent seperation results in BussThapen-SAT19

Landscape of Clausal Redundancy

HeuleKieslBiere-JAR20



CDCL(formula F)

$\alpha := \emptyset$
forever do
    $\alpha :=$ UnitPropagate(F, α)
    if α falsifies a clause in F **then**
        $C :=$ AnalyzeConflict$()$
        $F := F \wedge C$
        if C is the empty clause $\perp$ **then return** UNSAT
        $\alpha :=$ BackJump(C, α)

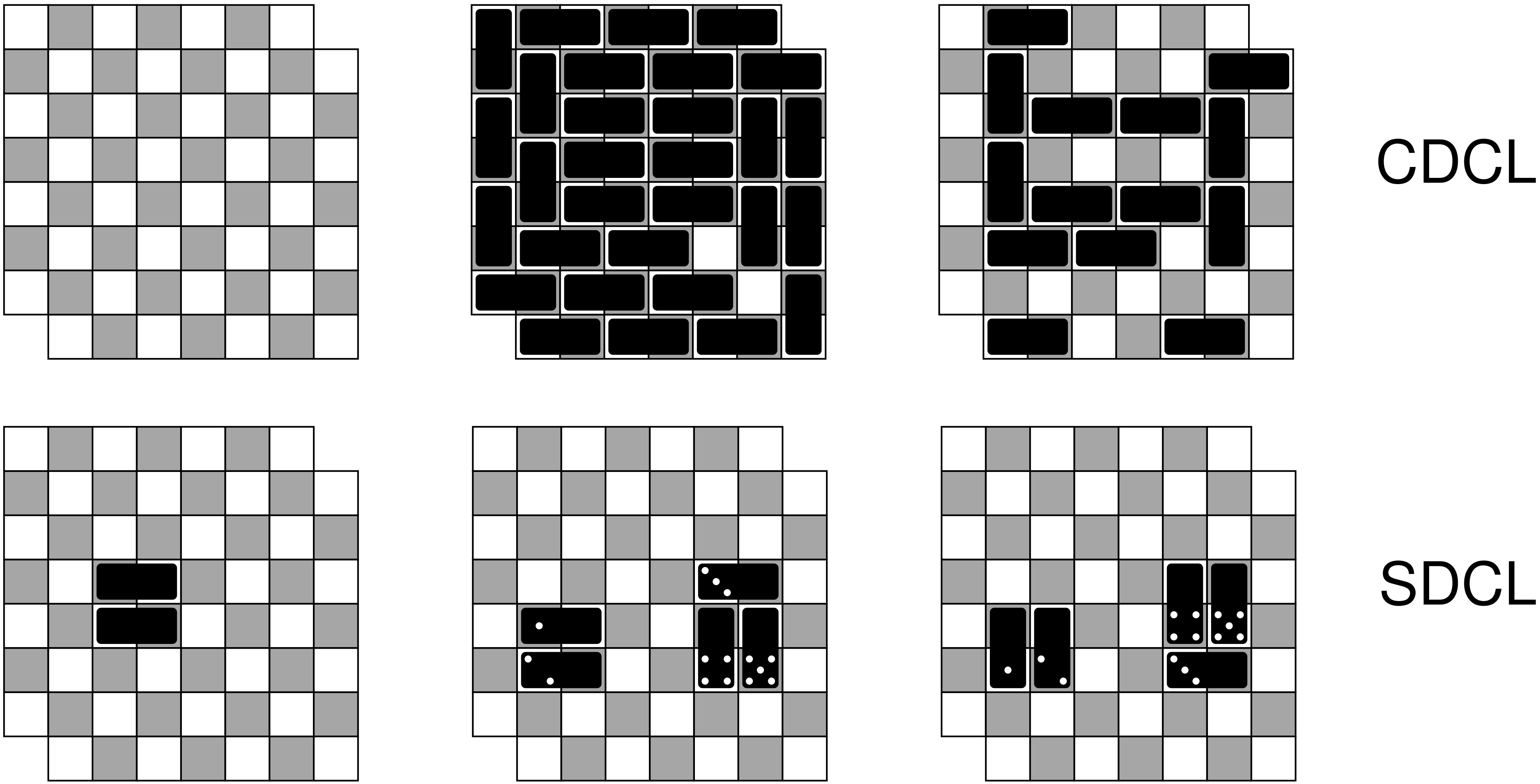
    else
        if all variables are assigned **then return** SAT
        $l :=$ Decide$()$
        $\alpha := \alpha \cup \{l\}$

SDCL(formula F)

$\alpha := \emptyset$
forever do
    $\alpha :=$ UnitPropagate(F, α)
    if α falsifies a clause in F **then**
        $C :=$ AnalyzeConflict()
        $F := F \wedge C$
        if C is the empty clause $\perp$ **then return** UNSAT
        $\alpha :=$ BackJump(C, α)
    else if the pruning predicate $P_\alpha(F)$ is satisfiable **then**
        $C :=$ AnalyzeWitness()
        $F := F \wedge C$
        $\alpha :=$ BackJump(C, α)
    else
        if all variables are assigned **then return** SAT
        $l :=$ Decide()
        $\alpha := \alpha \cup \{l\}$

Mutilated Chessboard

HeuleKieslBiere-NFM19



(Filtered) Positive Reduct HeuleKieslSeidlBiere-HVC17 HeuleKieslBiere-TACAS19

In the positive reduct consider all clauses satisfied by α but remove unassigned literals and add C:

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

Theorem. Let F be a formula, α an assignment, and C the clause that blocks α. Then, C is $\boxed{\text{SBC}}$ by an $L \subseteq C$ with respect to F if and only if the assignment α_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 α an assignment. Then, the filtered positive reduct of F and α is the formula $G \wedge C$ where C is the clause that blocks α and $G = \{\text{touched}_\alpha(D) \mid D \in F \text{ and } F|_\alpha \nvdash_1 \text{untouched}_\alpha(D)\}$.

Theorem. Let F be a formula, α an assignment, and C the clause that blocks α. Then, C is $\boxed{\text{SPR}}$ by an $L \subseteq C$ with respect to F if and only if the assignment α_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
- (practical) cutting plane proofs LiewBeameDevriendtElffersNordstrom-FMCAD20 GochtMcCreeshNordstrom-IJCAI20 GochtMcBrideMcCreeshNordstromProsserTrimble-CP20 GochtNordstrom-AAAI21 BogaertsGochteMcCreeshNordstrom-AAAI22 GochtMartinsNordstromOertel-SAT22
- 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 HeuleKullmannWieringaBiere-HVC11 best paper
- compositional propositional proofs HeuleBiere-LPAR15 theory
- parallel proof generation FleuryBiere-POS22 multi-threaded
- parallel proof checking Lammich-CADE17 parallel GRAT generator
- also for distributed solving (cloud) BiereChowdhuryHeuleKieslWhalen-SAT22
- also for GPU Osama-Dissertation22 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