

Fueling the SAT Revolution in Automated Reasoning

Armin Biere



Lecture Series AI
Johannes Kepler University Linz

December 5, 2023

Biggest thing in Computer Science in the 21st century?

A

Biggest thing in Computer Science in the 21st century!

Automated Reasoning

stole this joke from an invited talk of Clark Barrett

Seriously ...

<https://www.amazon.science/research-areas/automated-reasoning>

RESEARCH AREA

Automated reasoning

Focusing on the automation of formal logical reasoning to raise the bar on the security, durability, availability, and quality of Amazon's products and services.



Explore more

[AWS IAM Access Analyzer](#)

[Formal methods](#)

[Formal verification](#)

+ 2 more



AUTOMATED REASONING

A billion SMT queries a day

CAV keynote lecture by the director of applied science for AWS Identity explains how AWS is making the power of automated reasoning available to all customers.

By [Neha Rungta](#)

August 18, 2022

 Share

Conference
[FLoC 2022](#)

At this year's Computer-Aided Verification (CAV) conference — a leading automated-reasoning conference collocated with the Federated Logic Conferences ([FLoC](#)) — Amazon's Neha Rungta delivered a keynote talk in which she suggested that innovations at Amazon have "ushered in the golden age of automated reasoning".

Amazon scientists and engineers are using automated reasoning to prove the correctness of critical internal systems and to help customers prove the security of their cloud infrastructures. Many of these innovations are being driven by powerful reasoning engines called SMT solvers.

Satisfiability problems, or SAT, ask whether it's possible to assign variables true/false values that satisfy a set of



A billion SMT queries a day

Neha Rungta's 2022 CAV keynote



An efficient open-source automatic theorem prover for satisfiability modulo theories (SMT) problems.



[View on GitHub](#)

[Downloads](#)

[Documentation](#)

[People](#)

[Publications](#)

[Awards](#)

[Third Party Applications](#)

[Acknowledgements](#)

Try cvc5 online!

License [BSD 3-Clause](#)

CI passing

coverage 81.4%

People

Project Leaders

[Clark Barrett](#) (Stanford University)

[Cesare Tinelli](#) (University of Iowa)

Senior Technical Leads

[Haniel Barbosa](#) (Universidade Federal de Minas Gerais)

[Aina Niemetz](#) (Stanford University)

[Mathias Preiner](#) (Stanford University)

[Andrew Reynolds](#) (University of Iowa)

Current Developers

[Martin Brain](#) (University of Oxford and City University of London)

Hanna Lachnitt (Stanford University)

Abdalrhman Mohamed (University of Iowa)

[Mudathir Mohamed](#) (University of Iowa)

[Alex Ozdemir](#) (Stanford University)

[Ying Sheng](#) (Stanford University)

[Yoni Zohar](#) (Stanford University, Bar Ilan University)

Z3

An efficient SMT solver

[Overview](#)[People](#)[Publications](#)[Downloads](#)[Groups](#)[Events](#)[News & features](#)

Z3 is an efficient Satisfiability Modulo Theories (SMT) solver from Microsoft Research. Z3 is a solver for symbolic logic, a foundation for many software engineering tools. SMT solvers rely on a tight integration of specialized engines of proof. Each engine owns a piece of the global puzzle and implements specialized algorithms. For example, Z3's engine for arithmetic integrates Simplex, cuts and polynomial reasoning, while an engine for strings are regular expressions integrate methods for symbolic derivatives of regular languages. A theme shared among many of the algorithms is how they exploit a duality between finding satisfying solutions and finding refutation proofs. The solver also integrates engines for global and local inferences and global propagation. Z3 is used in a wide range of software engineering applications, ranging from program verification, compiler validation, testing, fuzzing using dynamic symbolic execution, model-based software development, network verification, and optimization.

[Download from GitHub](#)



Armin Biere
@ArminBiere

...

Hiking in Dagstuhl with @BjornerNikolaj

2:08 PM · Oct 12, 2022 · Twitter for Android

| View Tweet analytics

30 Likes



Tweet your reply

Reply



Followed by some Tweeters you follow

Bach Le @BachLe13 · Oct 13

...

Replies to @ArminBiere and
@BjornerNikolaj

Now SMT becomes the "backend"
of SAT in this picture ...



Theory and Practice of SAT and Combinatorial Solving

Organisatoren

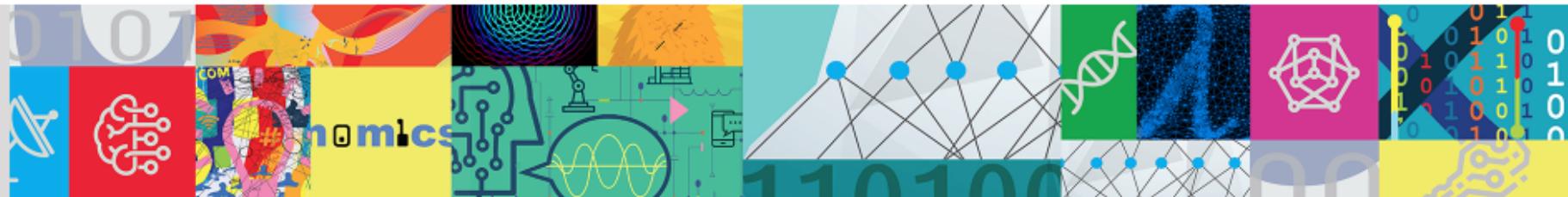
Olaf Beyersdorff (Friedrich-Schiller-Universität Jena, DE)

Armin Biere (Universität Freiburg, DE)

Vijay Ganesh (University of Waterloo, CA)

Jakob Nordström (University of Copenhagen, DK & Lund University, SE)





« Workshops & Symposia « Programs & Events « Home

Overview

Programs

Workshops & Symposia

› Upcoming Workshops & Symposia

› Past Workshops & Symposia

Research Pods

Internal Program Activities

Public Lectures

Participate

10th Anniversary Symposium

Workshops | Spring 2021



50 Years of Satisfiability: The Centrality of SAT in the Theory of Computing

(Weekly event) Feb 11, 2021 to May 13, 2021 (ended) [View all dates](#)

Program: Satisfiability: Theory, Practice, and Beyond

[Add to Calendar](#)

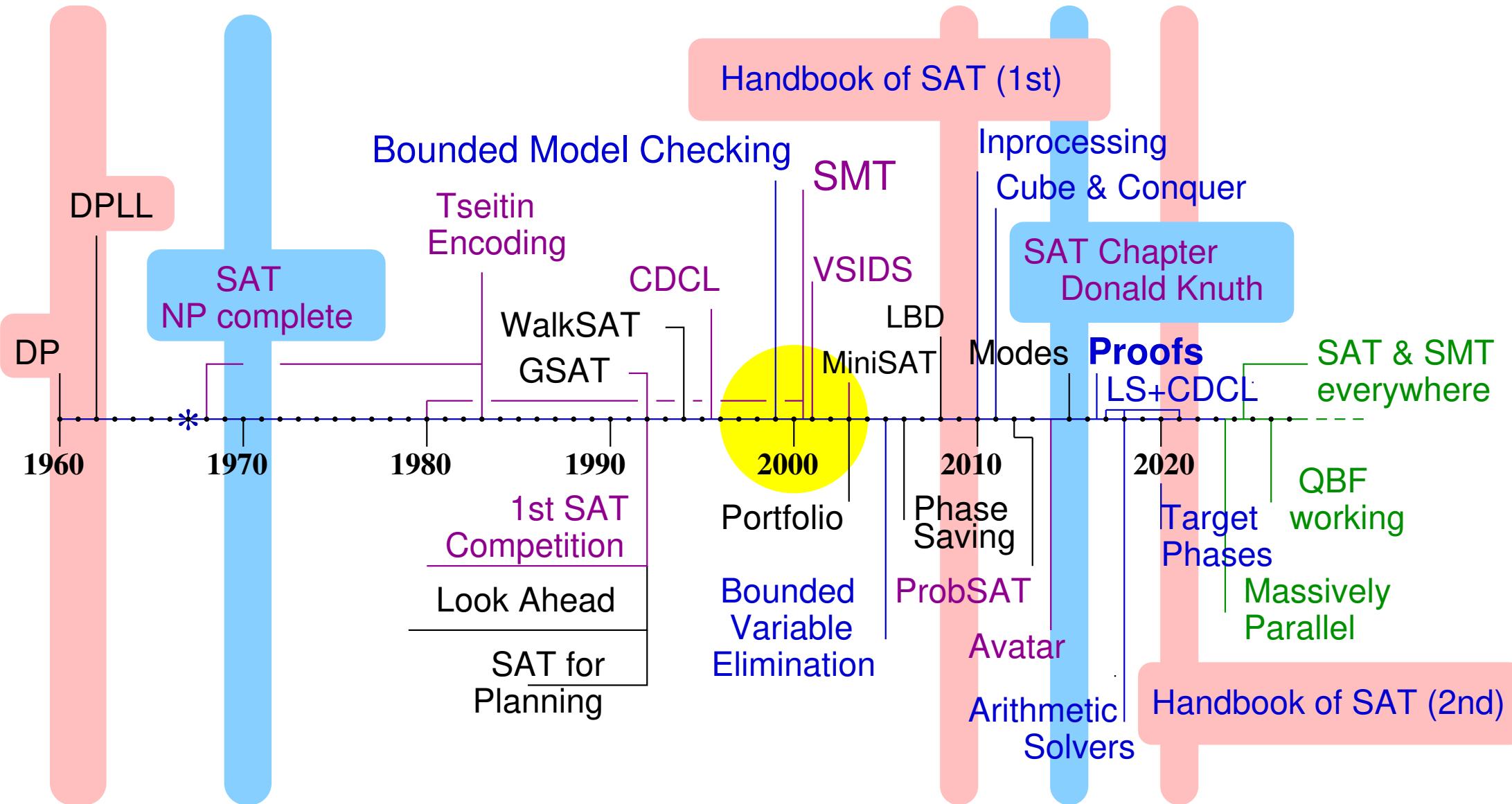
[View schedule & video »](#)

Organizers:

Albert Atserias (Universitat Politècnica de Catalunya; co-chair), Sam Buss (UC San Diego; co-chair), Paul Beame (University of Washington), Matti Järvisalo (University of Helsinki), Mohan Paturi (UC San Diego), Toni Pitassi (University of Toronto), Neil Thapen (Czech Academy of Sciences)

It has been nearly 50 years since the study of Satisfiability was initiated by the works of Cook, Karp and Levin. The ubiquity of NP-completeness caused the P versus NP question to become the defining central open question of Computational Complexity. This fundamental mathematical question remains open, but in the ensuing five decades, Satisfiability has become increasingly important in both theoretical and practical aspects of computer science. This workshop will explore and reflect on the role of Satisfiability in theoretical computer science: some talks will cover "big picture" topics, but most talks will emphasize current and recent research. Topics to be covered include: Proof

50 years of SAT NP-Complete



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 pythagorean triples,

e.g., $3^2 + 4^2 = 5^2$

are NOT monochromatic?

have two colors

$$(x_3 \vee x_4 \vee x_5) \wedge (\bar{x}_3 \vee \bar{x}_4 \vee \bar{x}_5)$$

$$(x_5 \vee x_{12} \vee x_{13}) \wedge (\bar{x}_5 \vee \bar{x}_{12} \vee$$

$$(x_6 \vee x_8 \vee x_{10}) \wedge (\bar{x}_6 \vee \bar{x}_8 \vee \bar{x}_1)$$

...

p cnf 7820 18930

3 4 5 0

-3 -4 -5 0

5 12 13 0

-5 -12 -13 0

...

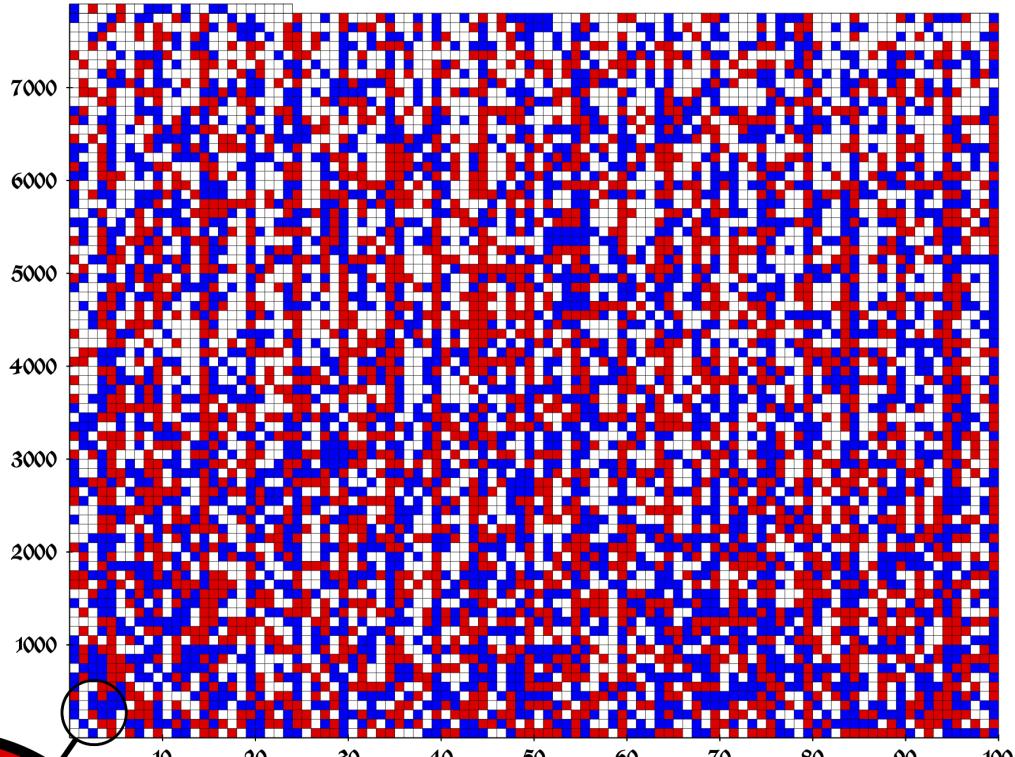
5412 5635 7813 0

-5412 -5635 -7813 0

5474 5520 7774 0

-5474 -5520 -7774 0

502	503	504	505	506	507	508	509	510	511
401	402	403	404	405	406	407	408	409	410
301	302	303	304	305	306	307	308	309	310
201	202	203	204	205	206	207	208	209	210
101	102	103	104	105	106	107	108	109	110
1	2	3	4	5	6	7	8	9	10



Yes, for $[1..7824] \Rightarrow \text{SAT}$

No, for $[1..7825] \Rightarrow \text{UNSAT}$

```
void encode () {
    int n = 7825;
    for (int i = 1; i <= n; i++)
        for (int j = i; j <= n; j++) {
            int k = sqrt (i*i + j*j);
            if (k <= n && i*i + j*j == k*k)
                printf ("%d %d %d 0\n", i, j, k),
                printf ("%d %d %d 0\n", -i, -j, -k);
        }
}
```

[nature](#) > [news](#) > [article](#)[Published: 26 May 2016](#)[Download PDF](#)

Two-hundred-terabyte maths proof is largest ever

[Evelyn Lamb](#)[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 terabytes¹, 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.

[Sections](#)[References](#)[References](#)[Additional information](#)[Related links](#)[Rights and permissions](#)[About this article](#)[Further reading](#)

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:



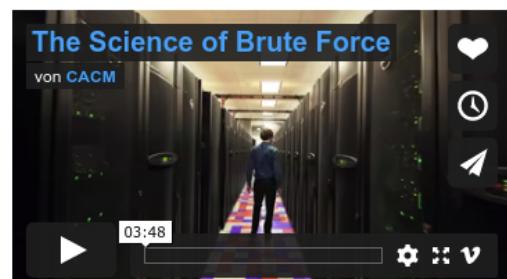
Credit: Peter Crowther Associates

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 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.

[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](#)
- [Allen Truths](#)
- [Conclusion](#)
- [References](#)
- [Authors](#)
- [Footnotes](#)

MORE NEWS & OPINIONS

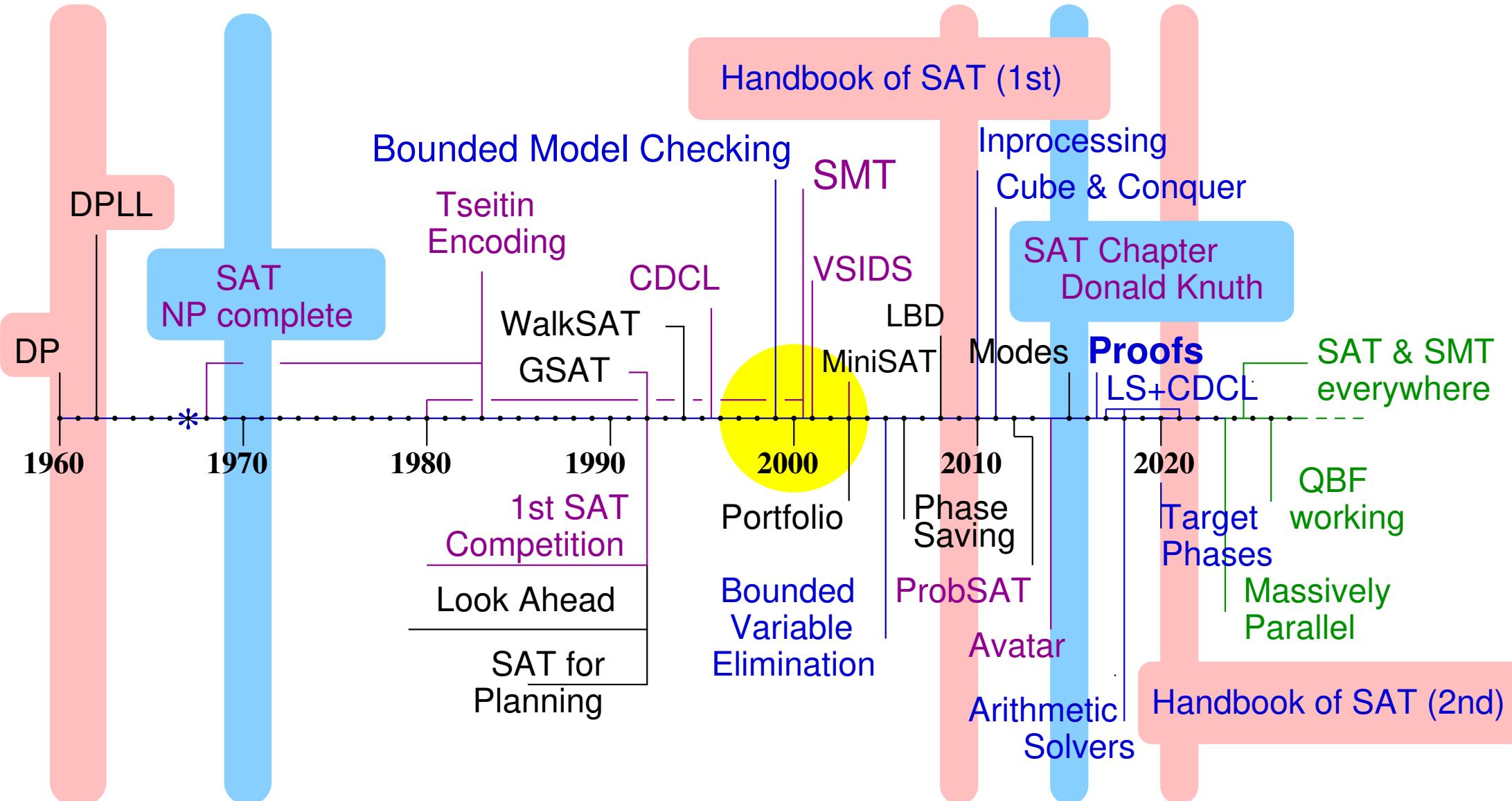
- [Congress Mandates](#)
- [Technologies to Stop Drunk Driving](#)

Associated Press

- [How AI Is Reinventing What Computers Are](#)
- [MIT Technology Review](#)
- [What Should be Done About Facebook?](#)

Jason Hong

Handbook'09



Satisfiability (SAT) related topics have attracted researchers from various disciplines. Logic, applied areas such as planning, scheduling, operations research and combinatorial optimization, but also theoretical issues on the theme of complexity, and much more, they all are connected through SAT.

My personal interest in SAT stems from actual solving: The increase in power of modern SAT solvers over the past 15 years has been phenomenal. It has become the key enabling technology in automated verification of both computer hardware and software. Bounded Model Checking (BMC) of computer hardware is now probably the most widely used model checking technique. The counterexamples that it finds are just satisfying instances of a Boolean formula obtained by unwinding to some fixed depth a sequential circuit and its specification in linear temporal logic. Extending model checking to software verification is a much more difficult problem on the frontier of current research. One promising approach for languages like C with finite word-length integers is to use the same idea as in BMC but with a decision procedure for the theory of bit-vectors instead of SAT. All decision procedures for bit-vectors that I am familiar with ultimately make use of a fast SAT solver to handle complex formulas.

Decision procedures for more complicated theories, like linear real and integer arithmetic, are also used in program verification. Most of them use powerful SAT solvers in an essential way.

Clearly, efficient SAT solving is a key technology for 21st century computer science. I expect this collection of papers on all theoretical and practical aspects of SAT solving will be extremely useful to both students and researchers and will lead to many further advances in the field.

Edmund Clarke

Edmund M. Clarke, FORE Systems University Professor of Computer Science and Professor of Electrical and Computer Engineering at Carnegie Mellon University, is one of the initiators and main contributors to the field of Model Checking, for which he also received the 2007 ACM Turing Award.

In the late 90s Professor Clarke was one of the first researchers to realize that SAT solving has the potential to become one of the most important technologies in model checking.



HANDBOOK of satisfiability



Editors:

Armin Biere
Marijn Heule
Hans van Maaren
Toby Walsh

IOS
Press

Frontiers in Artificial Intelligence and Applications

HANDBOOK of satisfiability

Editors:
Armin Biere
Marijn Heule
Hans van Maaren
Toby Walsh

IOS
Press

SAT Handbook 1st Edition (2009)

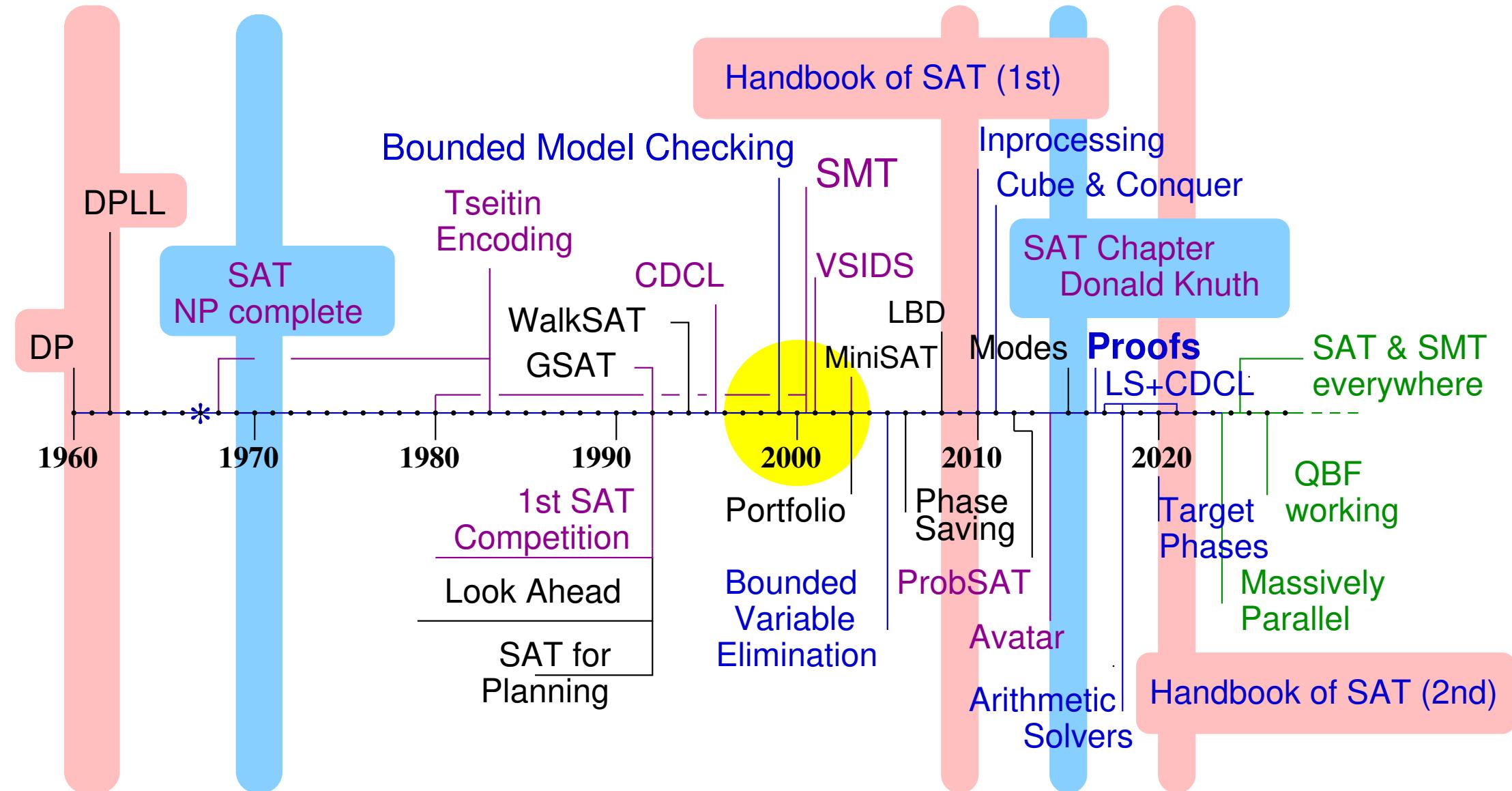
Part I. Theory and Algorithms

-    John Franco, John Martin:
A History of Satisfiability. 3-74
-    Steven David Prestwich:
CNF Encodings. 75-97
-    Adnan Darwiche, Knot Pipatsrisawat:
Complete Algorithms. 99-130
-    João P. Marques Silva, Inês Lynce, Sharad Malik:
Conflict-Driven Clause Learning SAT Solvers. 131-153
-    Marijn Heule, Hans van Maaren:
Look-Ahead Based SAT Solvers. 155-184
-    Henry A. Kautz, Ashish Sabharwal, Bart Selman:
Incomplete Algorithms. 185-203
-    Oliver Kullmann:
Fundamentals of Branching Heuristics. 205-244
-    Dimitris Achlioptas:
Random Satisfiability. 245-270
-    Carla P. Gomes, Ashish Sabharwal:
Exploiting Runtime Variation in Complete Solvers. 271-288
-    Karem A. Sakallah:
Symmetry and Satisfiability. 289-338
-    Hans Kleine Bünning, Oliver Kullmann:
Minimal Unsatisfiability and Autarkies. 339-401
-    Evgeny Dantsin, Edward A. Hirsch:
Worst-Case Upper Bounds. 403-424
-    Marko Samer, Stefan Szeider:
Fixed-Parameter Tractability. 425-454

Part II. Applications and Extensions

-    Armin Biere:
Bounded Model Checking. 457-481
-    Jussi Rintanen:
Planning and SAT. 483-504
-    Daniel Kroening:
Software Verification. 505-532
-    Hantao Zhang:
Combinatorial Designs by SAT Solvers. 533-568
-    Fabrizio Altarelli, Rémi Monasson, Guilhem Semerjian, Francesco Zamponi:
Connections to Statistical Physics. 569-611
-    Chu Min Li, Felip Manyà:
MaxSAT, Hard and Soft Constraints. 613-631
-    Carla P. Gomes, Ashish Sabharwal, Bart Selman:
Model Counting. 633-654
-    Rolf Drechsler, Tommi A. Junttila, Ilkka Niemelä:
Non-Clausal SAT and ATPG. 655-693
-    Olivier Roussel, Vasco M. Manquinho:
Pseudo-Boolean and Cardinality Constraints. 695-733
-    Hans Kleine Bünning, Uwe Bubeck:
Theory of Quantified Boolean Formulas. 735-760
-    Enrico Giunchiglia, Paolo Marin, Massimo Narizzano:
Reasoning with Quantified Boolean Formulas. 761-780
-    Roberto Sebastiani, Armando Tacchella:
SAT Techniques for Modal and Description Logics. 781-824
-    Clark W. Barrett, Roberto Sebastiani, Sanjit A. Seshia, Cesare Tinelli:
Satisfiability Modulo Theories. 825-885
-    Stephen M. Majercik:
Stochastic Boolean Satisfiability. 887-925

Knuth TAOCP Volume 4b Section 7.2.2.2 (300 pages)







NEWLY AVAILABLE SECTION OF
THE CLASSIC WORK

The Art of
Computer
Programming

VOLUME 4
Satisfiability

6
FASCICLE



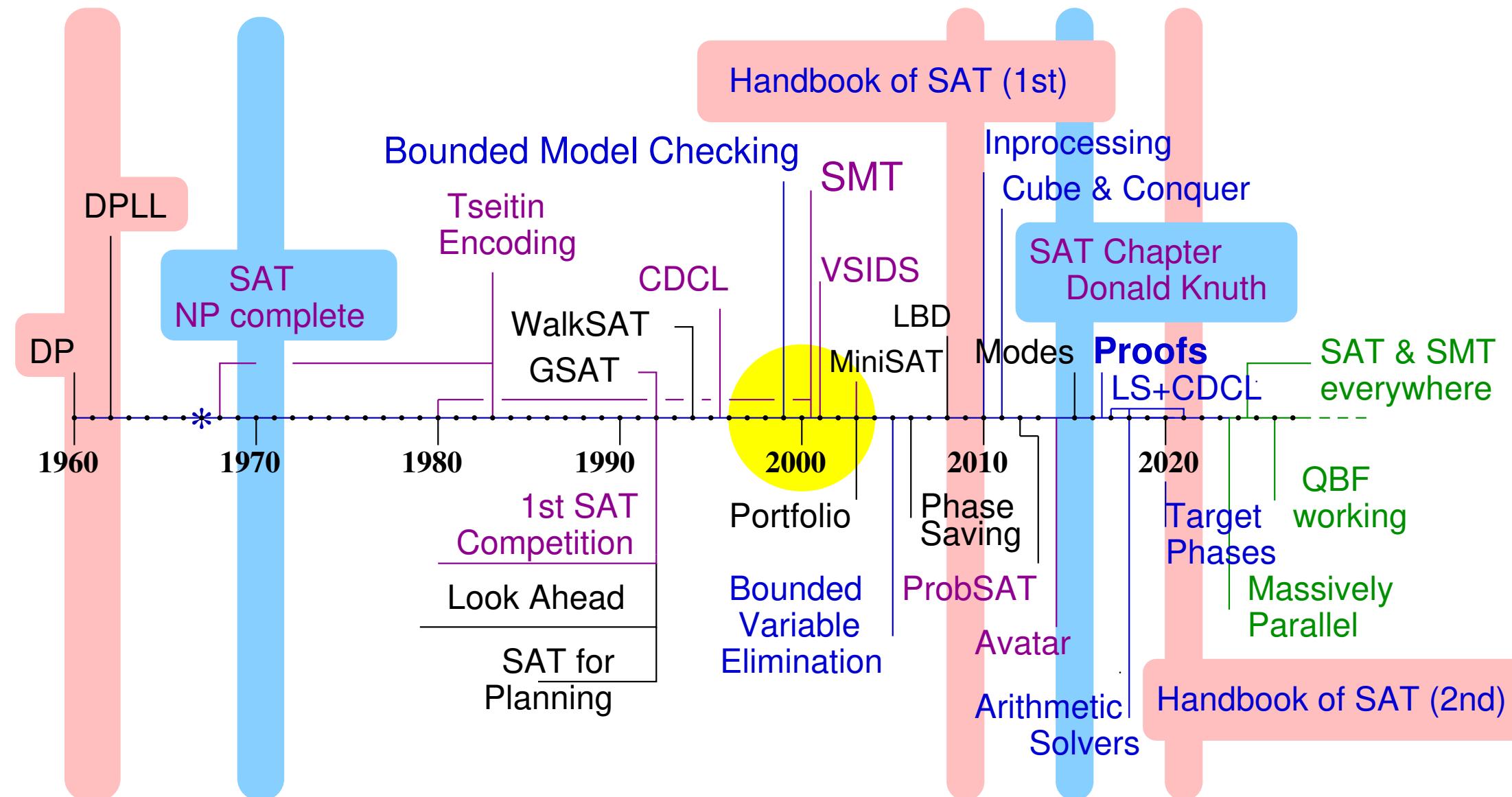
Special thanks are due to Armin Biere, Randy Bryant, Sam Buss, Niklas Eén, Ian Gent, Marijn Heule, Holger Hoos, Svante Janson, Peter Jeavons, Daniel Kroening, Oliver Kullmann, Massimo Lauria, Wes Pegden, Will Shortz, Carsten Sinz, Niklas Sörensson, Udo Wermuth, Ryan Williams, and ... for their detailed comments on my early attempts at exposition, as well as to numerous other correspondents who have contributed crucial corrections. Thanks also to Stanford's Information Systems Laboratory for providing extra computer power when my laptop machine was inadequate.

* * *

Wow—Section 7.2.2.2 has turned out to be the longest section, by far, in *The Art of Computer Programming*. The SAT problem is evidently a “killer app,” because it is key to the solution of so many other problems. Consequently I can only hope that my lengthy treatment does not also kill off my faithful readers! As I wrote this material, one topic always seemed to flow naturally into another, so there was no neat way to break this section up into separate subsections. (And anyway the format of *TAOCP* doesn’t allow for a Section 7.2.2.2.1.)

I’ve tried to ameliorate the reader’s navigation problem by adding subheadings at the top of each right-hand page. Furthermore, as in other sections, the exercises appear in an order that roughly parallels the order in which corresponding topics are taken up in the text. Numerous cross-references are provided

Biere
Bryant
Buss
Eén
Gent
Heule
Hoos
Janson
Jeavons
Kroening
Kullmann
Lauria
Pegden
Shortz
Sinz
Sörensson
Wermuth
Williams
Internet
MPR
Internet



SAT Handbook 2nd Edition (2021)

editors Armin Biere, Marijn Heule, Hans van Maaren, Toby Walsh

*with many updated chapters and the **following 7 new chapters**:*

Proof Complexity Jakob Nordström and Sam Buss

Preprocessing Armin Biere, Matti Järvisalo and Benjamin Kiesl

Tuning and Configuration

Holger Hoos, Frank Hutter and Kevin Leyton-Brown

Proofs of Unsatisfiability Marijn Heule

Core-Based MaxSAT

Fahiem Bacchus, Matti Järvisalo and Ruben Martins

Proof Systems for Quantified Boolean Formulas

Olaf Beyersdorff, Mikoláš Janota, Florian Lonsing and Martina Seidl

Approximate Model Counting Supratik Chakraborty, Kuldeep S. Meel, and Moshe Y. Vardi

SAT solving is a key technology for
21st century computer science.

Edmund Clarke

2007 ACM Turing Award Recipient

The SAT problem is evidently a killer app, because it is key to the solution of so many other problems.

Donald Knuth

1974 ACM Turing Award Recipient

The SAT problem is at the core of arguably the most fundamental question in computer science:
What makes a problem hard?

Stephen Cook

1982 ACM Turing Award Recipient

HANDBOOK of satisfiability

IOS
Press



February 2021
Approx. 1516 pp.

Print book
Hardcover, in 2 parts
ISBN: 978-1-64368-160-3
(print)
€200 / US\$250 / £180
excl. VAT

Ebook
ISBN: 978-1-64368-161-0
(online)
€200 / US\$250 / £180
excl. VAT

Discount Code
Order your print book
before April 15, 2021 and
get 35% off!
Code: **SAT2021**

-35%

For more information
and ordering check
tiny.cc/SAT2021



IOS Press
Nieuwe Hemweg 6B
1013 BG Amsterdam
The Netherlands
Tel.: +31 20 688 3355
Email: order@iospress.nl

Visit our website www.iospress.com for online ordering
For ebooks, go to www.ebooks.iospress.com
Follow us on Twitter: @IOSPress_STM
Follow us on Facebook: [publisheriospress](https://www.facebook.com/publisheriospress)

IOS Press, Inc.
6751 Tepper Drive
Clifton, VA 20124
USA
Tel.: +1 703 830 6300
Email: sales@iospress.com

Contents

Part I. Theory and Algorithms

Chapter 1. A History of Satisfiability
John Franco and John Martin, with sections contributed by
Miguel Anjos, Holger Hoos, Hans Kleine Bünning, Ewald
Speckenmeyer, Alasdair Urquhart, and Hantao Zhang

Chapter 2. CNF Encodings
Steven Prestwich

Chapter 3. Complete Algorithms
Adnan Darwiche and Knot Pipatsrisawat

Chapter 4. Conflict-Driven Clause Learning SAT Solvers
João Marques-Silva, Inês Lynce, and Sharad Malik

Chapter 5. Look-Ahead Based SAT Solvers
Marijn J.H. Heule and Hans van Maaren

Chapter 6. Incomplete Algorithms
Henry Kautz, Ashish Sabharwal, and Bart Selman

Chapter 7. Proof Complexity and SAT Solving
Sam Buss and Jakob Nordström

Chapter 8. Fundamentals of Branching Heuristics
Oliver Kullmann

Chapter 9. Preprocessing in SAT Solving
Armin Biere, Matti Järvisalo, and Benjamin Kiesl

Chapter 10. Random Satisfiability
Dimitris Achlioptas

Chapter 11. Exploiting Runtime Variation in Complete Solvers
Carla P. Gomes and Ashish Sabharwal

Chapter 12. Automated Configuration and Selection of SAT
Solvers
Holger H. Hoos, Frank Hutter, and Kevin Leyton-Brown

Chapter 13. Symmetry and Satisfiability
Karem A. Sakallah

Chapter 14. Minimal Unsatisfiability and Autarkies
Hans Kleine Bünning and Oliver Kullmann

Chapter 15. Proofs of Unsatisfiability
Marijn J.H. Heule

Chapter 16. Worst-Case Upper Bounds
Evgeny Dantsin and Edward A. Hirsch

Chapter 17. Fixed-Parameter Tractability
Marko Samer (1977-2010) and Stefan
Szeider

Part II. Applications and Extensions

Chapter 18. Bounded Model Checking
Armin Biere

Chapter 19. Planning and SAT
Jussi Rintanen

Chapter 20. Software Verification
Daniel Kroening

Chapter 21. Combinatorial Designs by SAT Solvers
Hantao Zhang

Chapter 22. Connections to Statistical Physics
Fabrizio Altarelli, Rémi Monasson, Guilhem Semerjian and
Francesco Zamponi

Chapter 23. MaxSAT, Hard and Soft Constraints
Chu Min Li and Felip Manyà

Chapter 24. Maximum Satisfiability
Fahiem Bacchus, Matti Järvisalo, and Ruben Martins

Chapter 25. Model Counting
Carla P. Gomes, Ashish Sabharwal, and Bart Selman

Chapter 26. Approximate Model Counting
Supratik Chakraborty, Kuldeep S. Meel, and Moshe Y. Vardi

Chapter 27. Non-Clausal SAT and ATPG
Rolf Drechsler, Tommi Junttila and Ilkka Niemelä

Chapter 28. Pseudo-Boolean and Cardinality Constraints
Olivier Roussel and Vasco Manquinho

Chapter 29. Theory of Quantified Boolean Formulas
Hans Kleine Bünning and Uwe Bubeck

Chapter 30. Reasoning with Quantified Boolean Formulas
Enrico Giunchiglia, Paolo Marin, and Massimo Narizzano

Chapter 31. Quantified Boolean Formulas
Olaf Beyersdorff, Mikoláš Janota, Florian Lonsing, Martina Seidl

Chapter 32. SAT Techniques for Modal and Description Logics
Roberto Sebastiani and Armando Tacchella

Chapter 33. Satisfiability Modulo Theories
Clark Barrett, Roberto Sebastiani, Sanjai A. Seshia, and Cesare Tinelli

Chapter 34. Stochastic Boolean Satisfiability
Stephen M. Majercik

Subject Index

Cited Author Index

Contributing Authors and Affiliations



Kernighan/Ritchie · Programmieren in C **PC** professionell

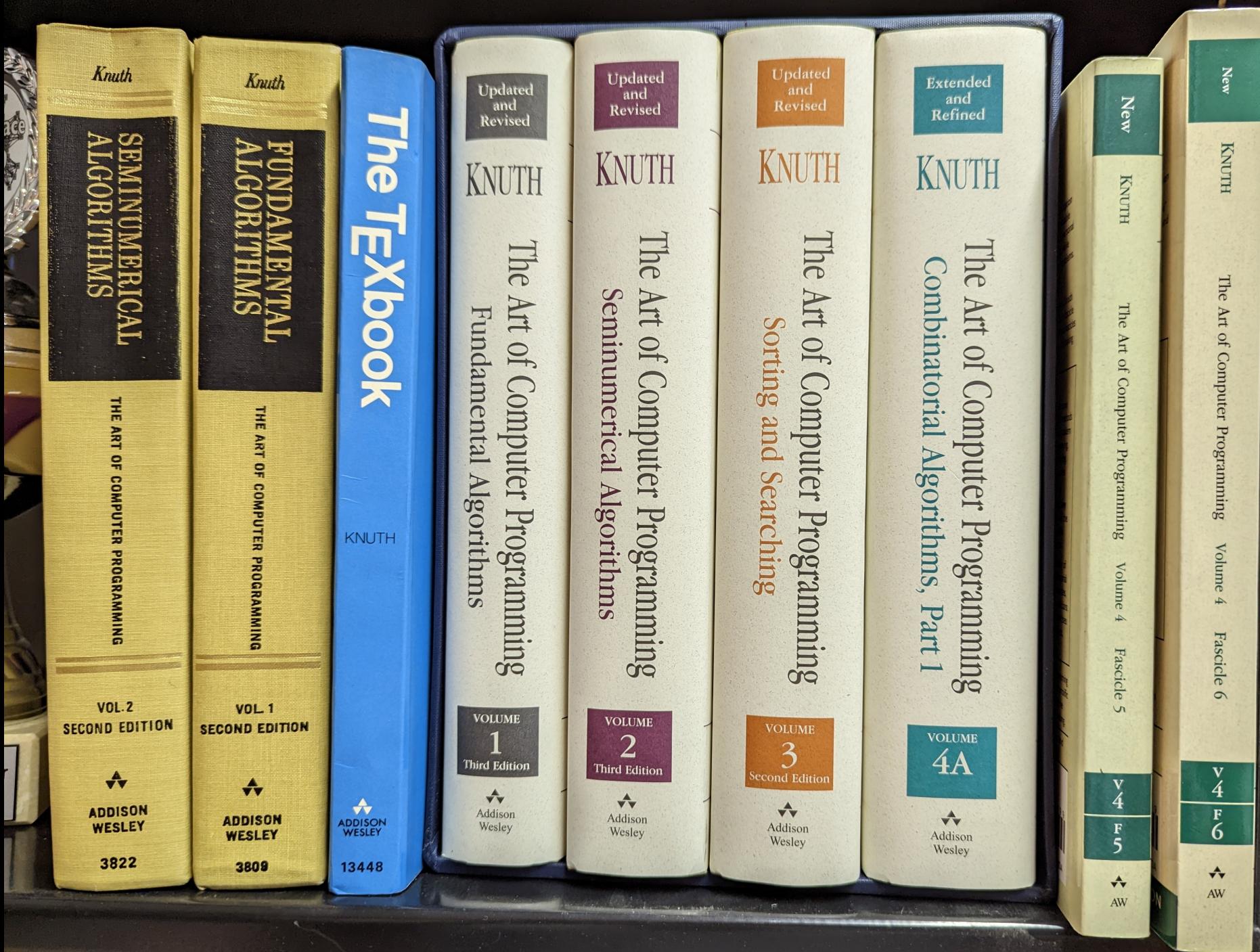
KERNIGHAN · RITCHIE

THE C PROGRAMMING LANGUAGE

SECOND EDITION

PTR

PH



I'm a student I'm an educatorTHE CLASSIC WORK
EXTENDED AND REFINED

The Art of Computer Programming

VOLUME 4B
Combinatorial Algorithms
Part 2

DONALD E. KNUTH

Art of Computer Programming, Volume 4B, The: Combinatorial Algorithms, 1st edition

Published by Addison-Wesley Professional (September 28th 2022) - Copyright © 2023

Donald E. Knuth

[Print](#)

\$63.99

[Bundle](#)

\$324.99

Hardcover

\$63.99
\$79.99[Buy now](#)

Free delivery

ISBN-13: 9780201038064

Art of Computer Programming, The: Combinatorial
Algorithms, Volume 4B

[Blick ins Buch](#) ↴

THE CLASSIC WORK
EXTENDED AND REFINED

The Art of Computer Programming

VOLUME 4B
Combinatorial Algorithms
Part 2

DONALD E. KNUTH



The Art of Computer

[Dieses Bild anzeigen](#)

[Dem Autor folgen](#)



Donald E.
Knuth

[Folgen](#)

The Art of Computer Programming, Volume 4B: Combinatorial Algorithms Gebundene

Ausgabe – 21. Oktober 2022

Englisch Ausgabe | von [Donald E. Knuth](#) (Autor)

Buch 4 von 4: The Art of Computer Programming

[Alle Formate und Editionen anzeigen](#)

Kindle

10,29 €

Gebundenes Buch

97,53 €

Lesen Sie mit unserer [kostenfreien App](#)

2 Neu ab 82,15 €

The Art of Computer Programming is Knuth's multivolume analysis of algorithms. With the addition of this new volume, it continues to be the definitive description of classical computer science.

Volume 4B, the sequel to Volume 4A, extends Knuth's exploration of combinatorial algorithms. These algorithms are of keen interest to software designers because ". . . a single

[▼ Mehr lesen](#)

[Falsche Produktinformationen melden](#)

Seitenzahl der
Print-Ausgabe



640 Seiten

Sprache



Englisch

Herausgeber



Addison Wesley

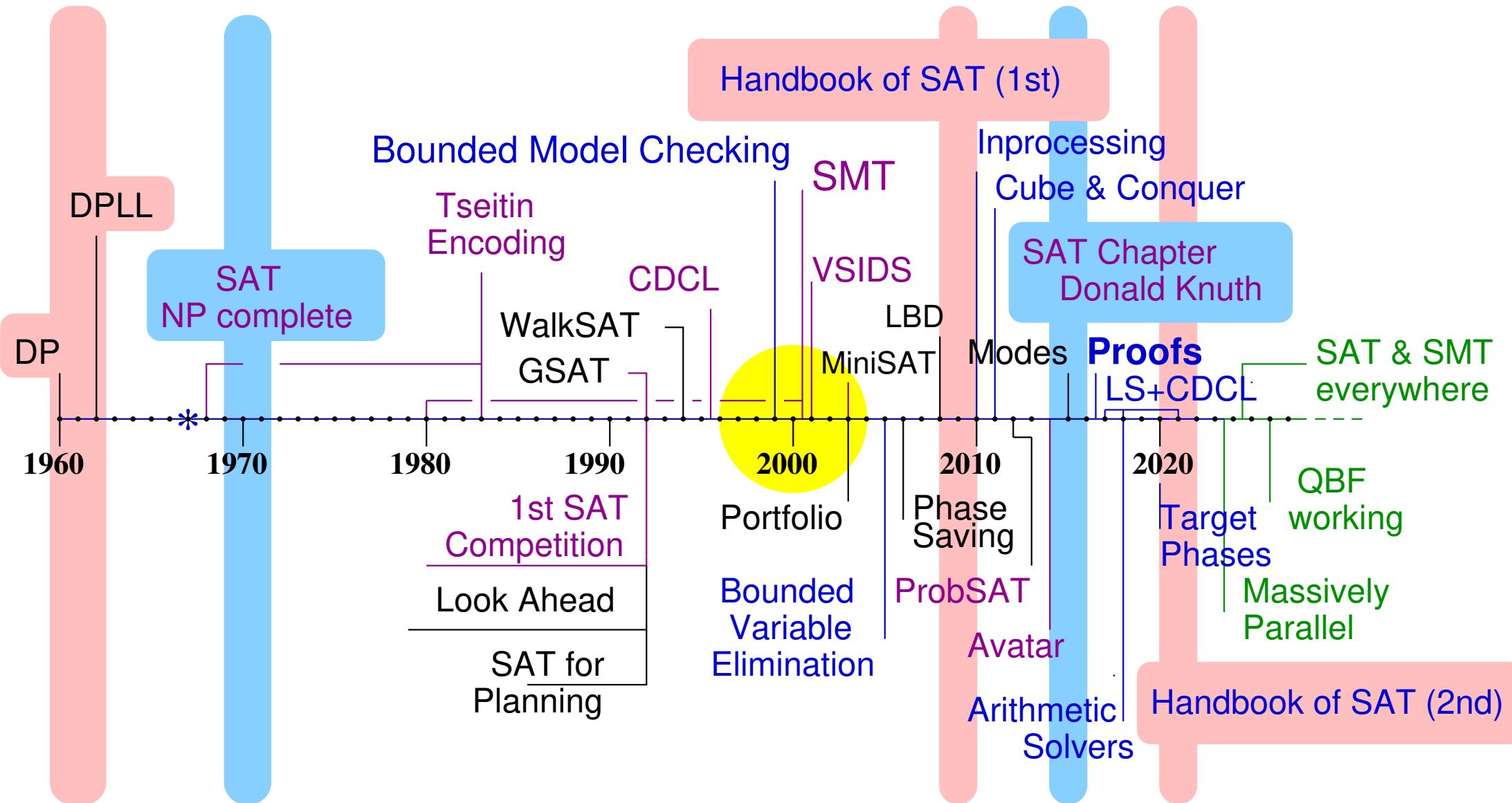
Erscheinungstermin



21. Oktober 2022



60 Years of SAT Solving



DPLL from the 60ies

[DaviesPutnam'62][DavisLogemannLoveland'62]

$DPLL(F)$

$F := BCP(F)$

boolean constraint propagation

if $F = \top$ **return** satisfiable

if $\perp \in F$ **return** unsatisfiable

pick remaining variable x and literal $l \in \{x, \neg x\}$

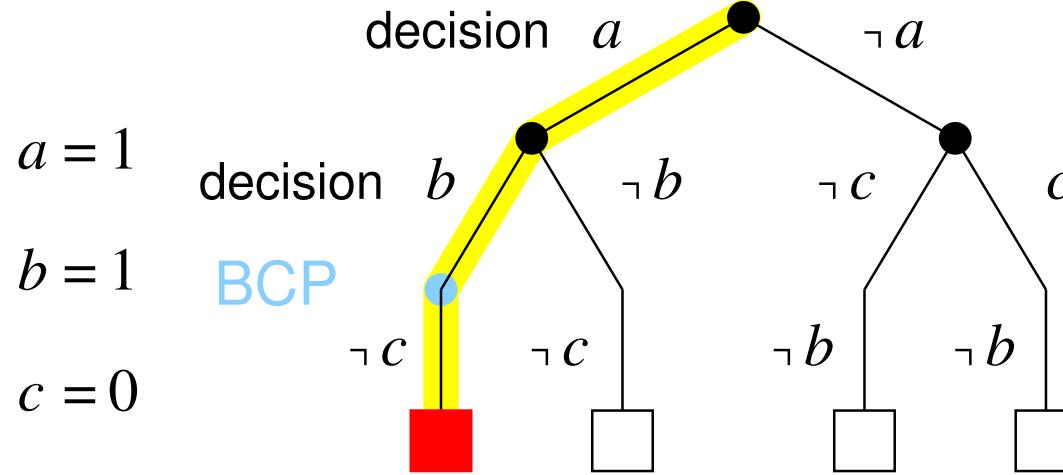
if $DPLL(F \wedge \{l\})$ returns satisfiable **return** satisfiable

return $DPLL(F \wedge \{\neg l\})$

\neq

CDCL

DPLL Example

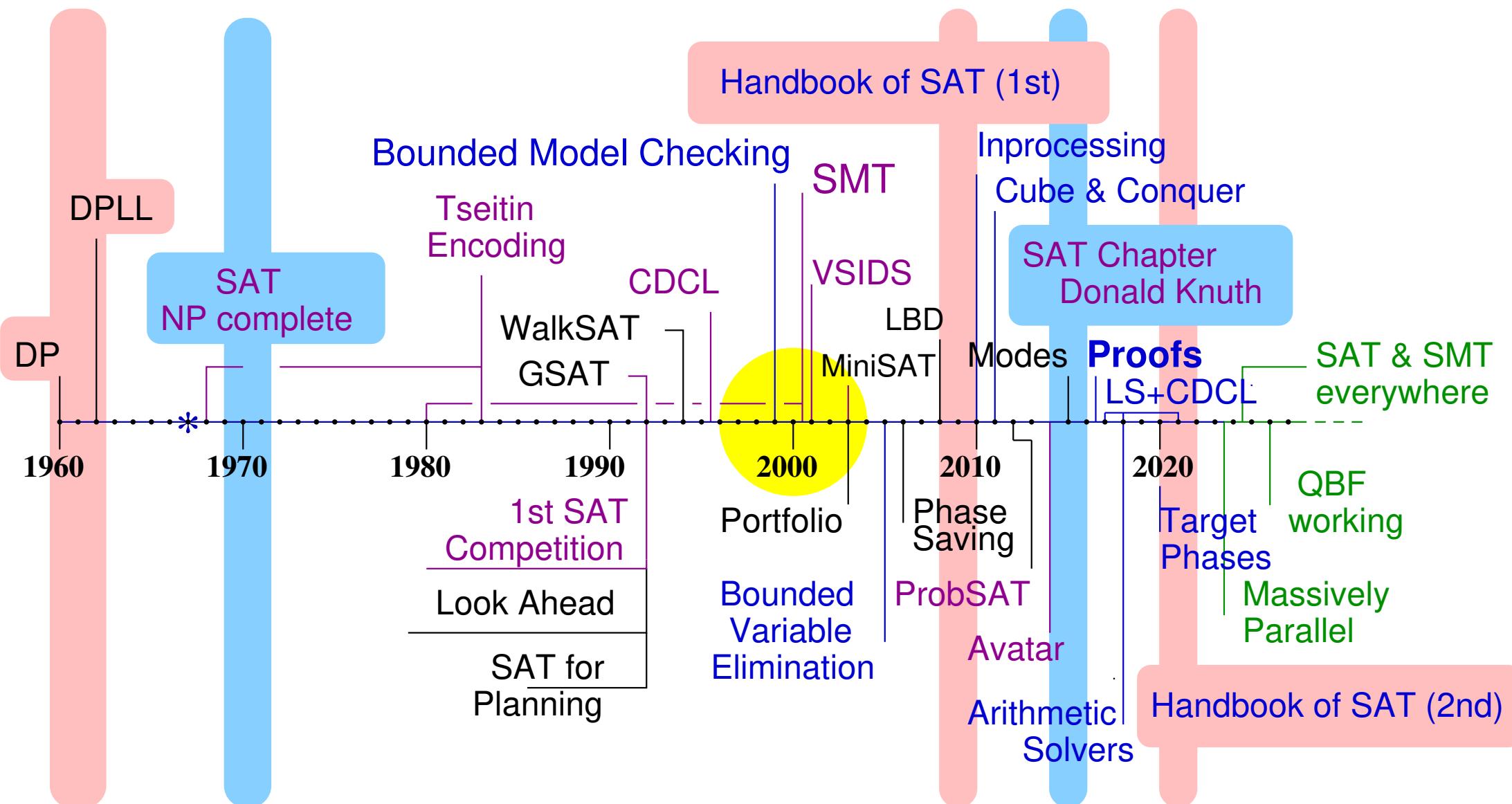


clauses

$\neg a \vee \neg b \vee \neg c$
$\neg a \vee \neg b \vee c$
$\neg a \vee b \vee \neg c$
$\neg a \vee b \vee c$
$a \vee \neg b \vee \neg c$
$a \vee \neg b \vee c$
$a \vee b \vee \neg c$
$a \vee b \vee c$

CDCL from the 90ies

[MarquesSilvaSakallah'96]

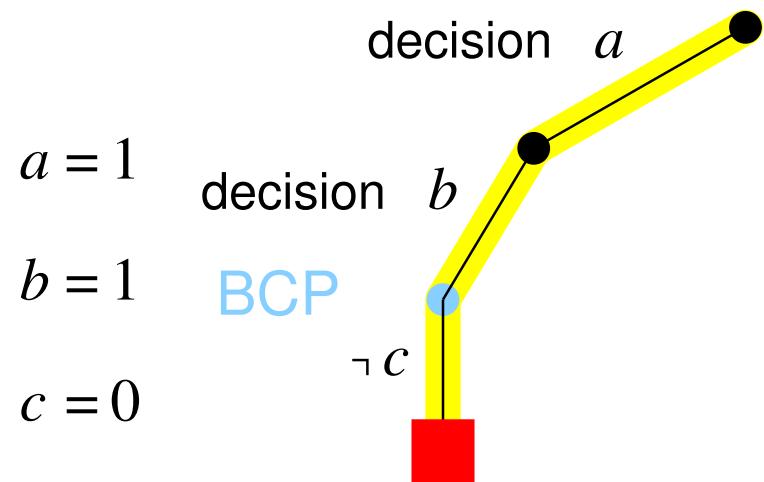


```
int basic_cdcl_loop () {
    int res = 0;

    while (!res)
        if (unsat) res = 20;
        else if (!propagate ()) analyze ();      // analyze propagated conflict
        else if (satisfied ()) res = 10;          // all variables satisfied
        else decide ();                          // otherwise pick next decision

    return res;
}
```

CDCL Example



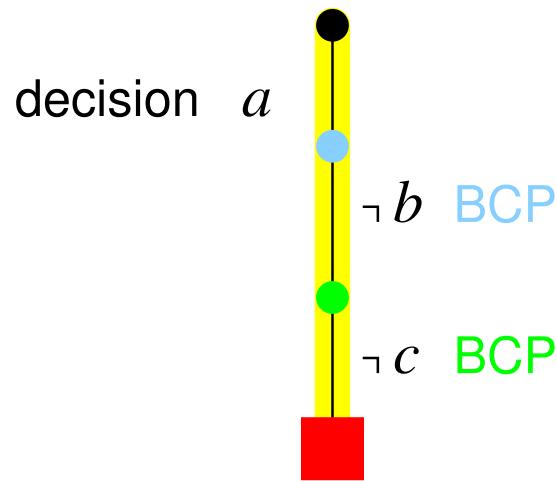
clauses

$\neg a \vee \neg b \vee \neg c$
$\neg a \vee \neg b \vee c$
$\neg a \vee b \vee \neg c$
$\neg a \vee b \vee c$
$a \vee \neg b \vee \neg c$
$a \vee \neg b \vee c$
$a \vee b \vee \neg c$
$a \vee b \vee c$

learn $\neg a \vee \neg b$

CDCL Example

$a = 1$
 $b = 0$
 $c = 0$



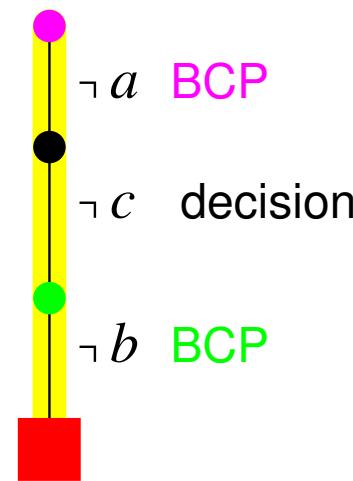
clauses
$\neg a \vee \neg b \vee \neg c$
$\neg a \vee \neg b \vee c$
$\neg a \vee b \vee \neg c$
$\neg a \vee b \vee c$
$a \vee \neg b \vee \neg c$
$a \vee \neg b \vee c$
$a \vee b \vee \neg c$
$a \vee b \vee c$
$\neg a \vee \neg b$
learn $\neg a$

CDCL Example

$$a = 1$$

$$b = 0$$

$$c = 0$$

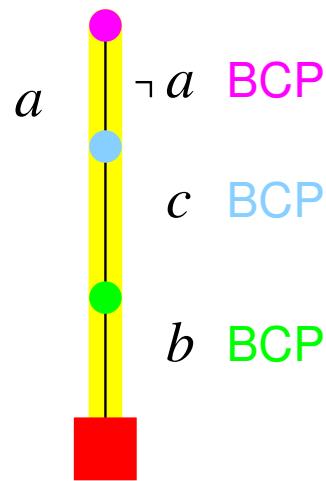


clauses

$\neg a \vee \neg b \vee \neg c$
$\neg a \vee \neg b \vee c$
$\neg a \vee b \vee \neg c$
$\neg a \vee b \vee c$
$a \vee \neg b \vee \neg c$
$a \vee \neg b \vee c$
$a \vee b \vee \neg c$
$a \vee b \vee c$
$\neg a \vee \neg b$
$\neg a$
learn
c

CDCL Example

$a = 1$
 $b = 0$
 $c = 0$



clauses

$\neg a \vee \neg b \vee \neg c$
 $\neg a \vee \neg b \vee c$
 $\neg a \vee b \vee \neg c$
 $\neg a \vee b \vee c$
 $a \vee \neg b \vee \neg c$
 $a \vee \neg b \vee c$
 $a \vee b \vee \neg c$
 $a \vee b \vee c$
 $\neg a \vee \neg b$
 $\neg a$
 c

learn

\perp

empty clause

More Advanced CDCL Loop with Reduce and Restart

```
int basic_cdcl_loop_with_reduce_and_restart () {  
  
    int res = 0;  
  
    while (!res)  
        if (unsat) res = 20;  
        else if (!propagate ()) analyze (); // analyze propagated conflict  
        else if (satisfied ()) res = 10; // all variables satisfied  
        else if (restarting ()) restart (); // restart by backtracking  
        else if (reducing ()) reduce (); // collect useless learned clauses  
        else decide (); // otherwise pick next decision  
  
    return res;  
}
```

Inprocessing CaDiCaL CDCL Loop

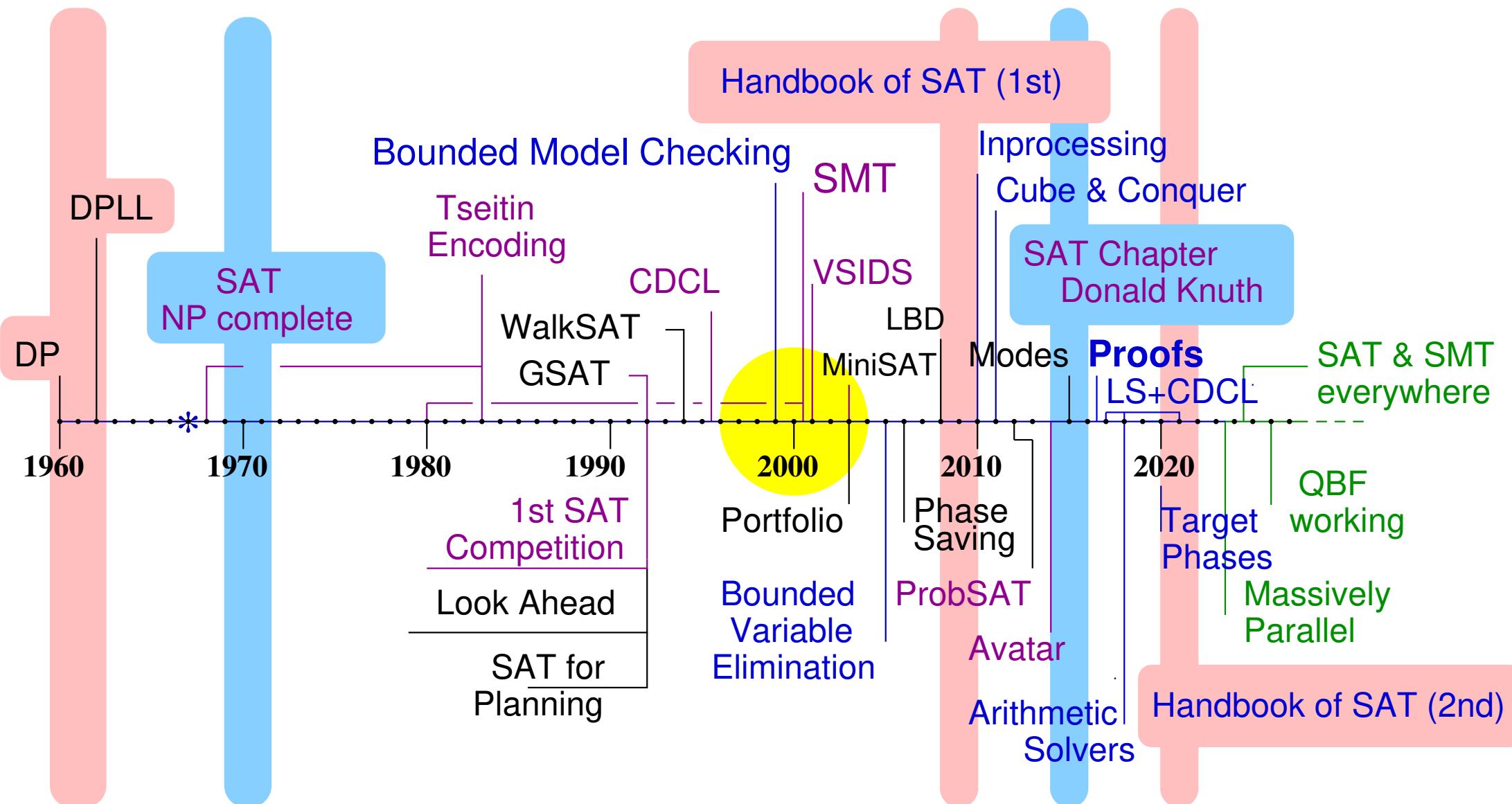
<https://github.com/arminbiere/cadical>

```
while (!res) {
    if (unsat) res = 20;
    else if (unsat_constraint) res = 20;
    else if (!propagate ()) analyze ();           // propagate and analyze
    else if (iterating) iterate ();               // report learned unit
    else if (satisfied ()) res = 10;              // found model
    else if (search_limits_hit ()) break;         // decision or conflict limit
    else if (terminated_asynchronously ())
        break;
    else if (restarting ()) restart ();           // restart by backtracking
    else if (rephasing ()) rephase ();            // reset variable phases
    else if (reducing ()) reduce ();              // collect useless clauses
    else if (probing ()) probe ();                // failed literal probing
    else if (subsuming ()) subsume ();             // subsumption algorithm
    else if (eliminating ()) elim ();              // variable elimination
    else if (compacting ()) compact ();            // collect variables
    else if (conditioning ()) condition ();       // globally blocked clauses
    else res = decide ();                         // next decision
}
```

Probabilistic local search SAT solvers

WalkSAT'94

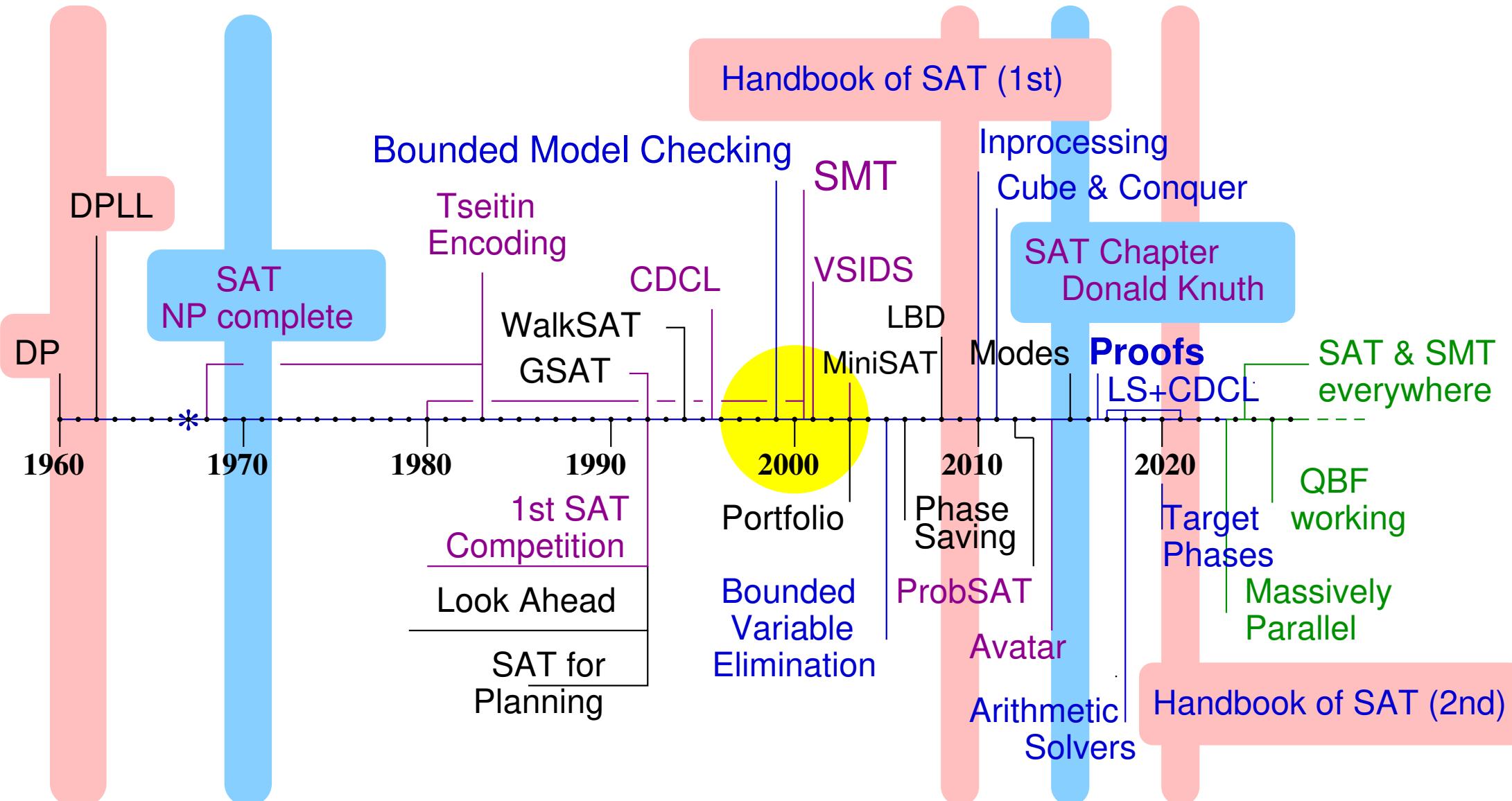
ProbSAT'12



generate a random 0/1 initial assignment for all literals L
while (exists unsatisfied clause C in formula) {
 with probability p
 pick literal L in C which breaks minimial number clauses
 otherwise with probability 1-p pick random literal L in C
 flip literal L
}

generate a random 0/1 initial assignment for all literals L
while (exists unsatisfied clause C in formula) {
 let $s(L)$ be the number clauses broken by flipping L
 pick L probabilistically with probability $2^{-s(L)}$
 flip literal L
}

Satisfiability Modulo Theories (SMT) aka CDCL(T) from around 2000



Satisfiability Modulo Theories (SMT) aka CDCL(T)

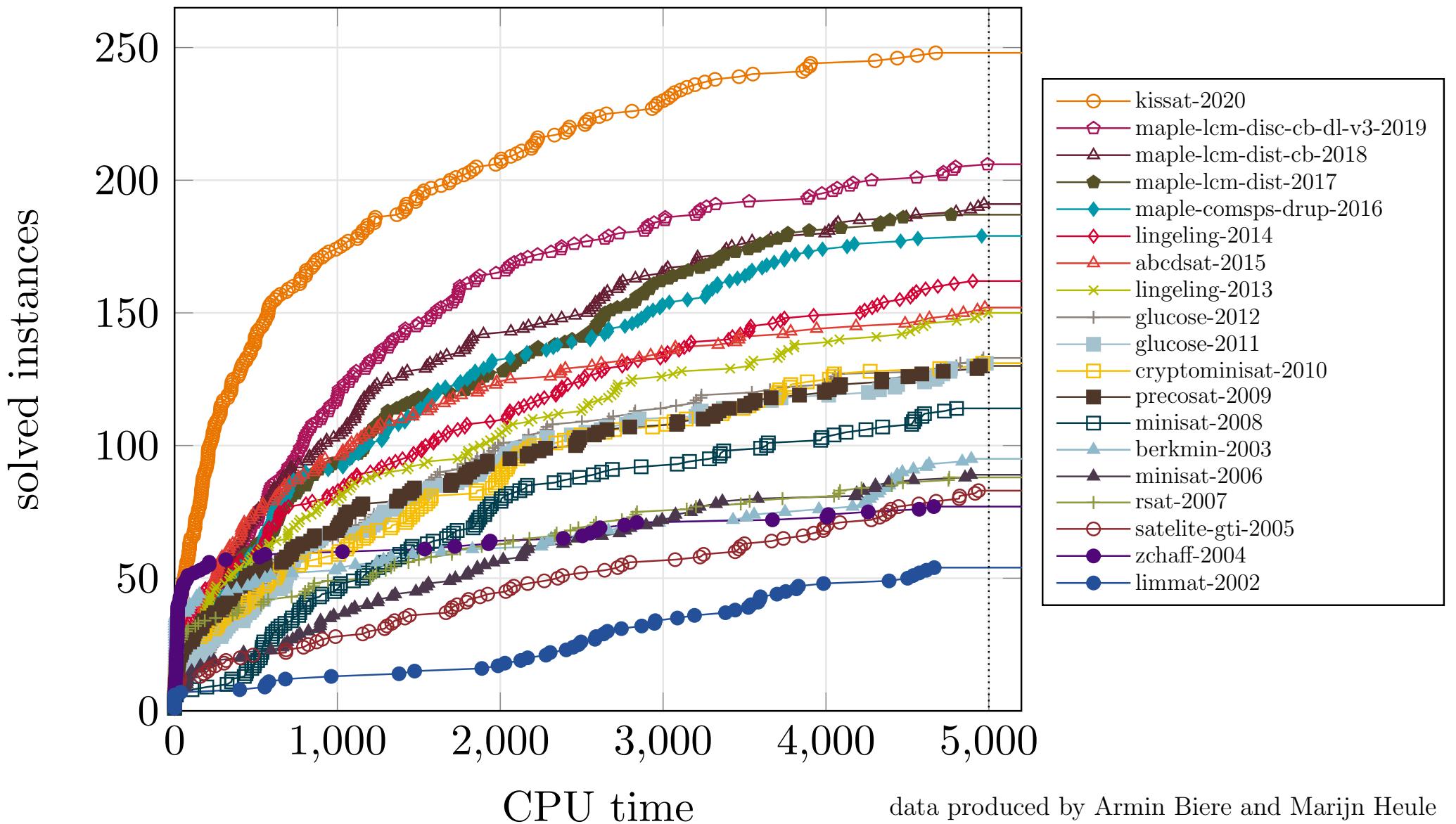
$$f(x) \neq f(y) \wedge x + u = 3 \wedge v + y = 3 \wedge u = t[z] \wedge v = t[w] \wedge z = w$$

- originally unquantified formulas in first-order logic
- interpreted symbols over various theories
 - basic theory of equality, uninterpreted functions
 - arithmetic expressions (linear, non-linear, integer, reals)
 - theory of bit-vectors and floating-points to model bit-precise reasoning of HW/SW
 - arrays (McCarthy axioms) to model memory / pointers
 - strings to model regular expressions
- standardized input-format / semantics SMT-LIB
- popular SMT solvers: CVC, Yices, Z3, Boolector, ...

$$x * 10001000_8 = y \wedge x \gg 3 = y \wedge x[3..0] = y[7..4]$$



SAT Competition Winners on the SC2020 Benchmark Suite

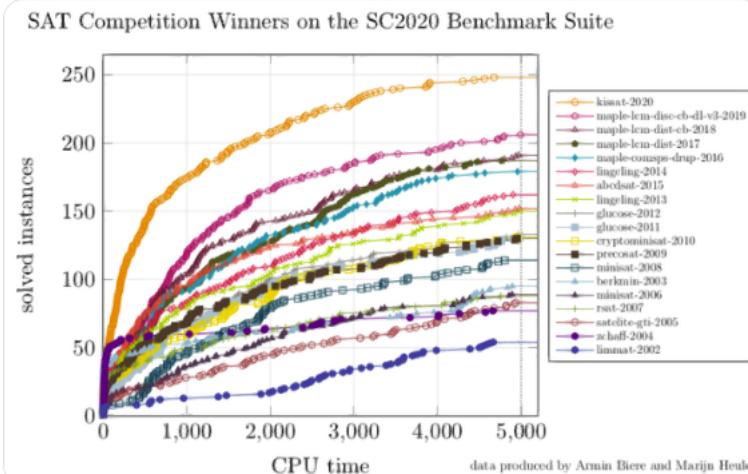


data produced by Armin Biere and Marijn Heule

some Tweets

Armin Biere
@ArminBiere

SAT solvers get faster and faster: all-time winners of the SAT Competition on 2020 instances, featuring our new solver Kissat (fmv.jku.at/kissat), which won in 2020. The web page also has runtime CDFs for 2011 and 2019.



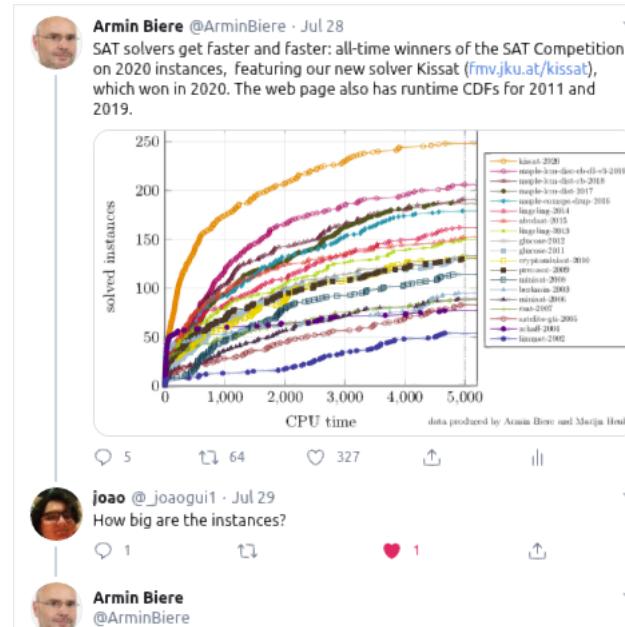
5:20 PM · Jul 28, 2020 · Twitter Web App

View Tweet activity

57 Retweets 7 Quote Tweets 327 Likes

Armin Biere
@ArminBiere

Eventually I will need to support 64-bit variable indices (Lingeling has 2^{27} , CaDiCaL indeed 2^{31} and Kissat 2^{28} as compromise though it could easily do half a billion)



5 64 327

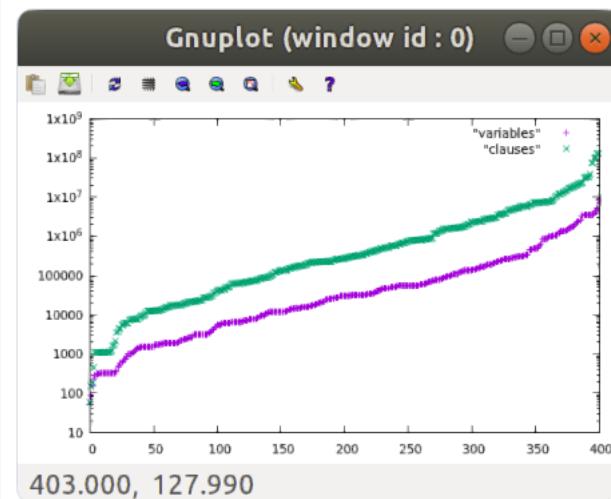
joao @_joaogui1 · Jul 29
How big are the instances?

1 1 1

Armin Biere
@ArminBiere

Replying to @_joaogui1

The largest ones have millions of variables and clauses. The planning track had even larger ones. See the variable and clause distribution plot for the main track:



A Andrew V. Jones 13:40
an Boolector/boolector, S...

Can you try compiling Boolector with a different SAT solver? I believe that CaDiCaL has a much higher limit (maybe INT_MAX vars).

Zitierten Text anzeigen

A Aina Niemetz 18:16
an Boolector/boolector, S...

As [@andrewvaughanj](#) points out, this is a limitation in the SAT solver that we can not control. Let me add that CaDiCaL typically outperforms Lingeling in combination with Boolector, so it might be a good idea to switch to CaDiCaL anyways.

more Tweets



Daniel Le Berre @dleberre · 4 T

This week, we will have again many great talks about SAT at [@SimonsInstitute](#): I am especially looking forward the one of [@ArminBiere](#) titled « A Personal History of Practical SAT Solving » simons.berkeley.edu/talks/tbd-308

1

5

19



Daniel Le Berre @dleberre · 4 T

Did you know that [@ArminBiere](#) participated to every SAT competitive event since 2002? And won in a fair amount of tracks? See all his solvers here: fmv.jku.at/software/index...

1

5

3



Daniel Le Berre

@dleberre

I remember the solvers Limmat, Compsat, NanoSAT, PicoSAT, PrecoSAT, Lingeling, Splatz, CaDiCaL, Kissat plus « educational » solvers Cleaneling and Satch. Would love to see a cactus plot with all those solvers on Thursday [@ArminBiere](#) :)



Adam P. Goucher

@apgox

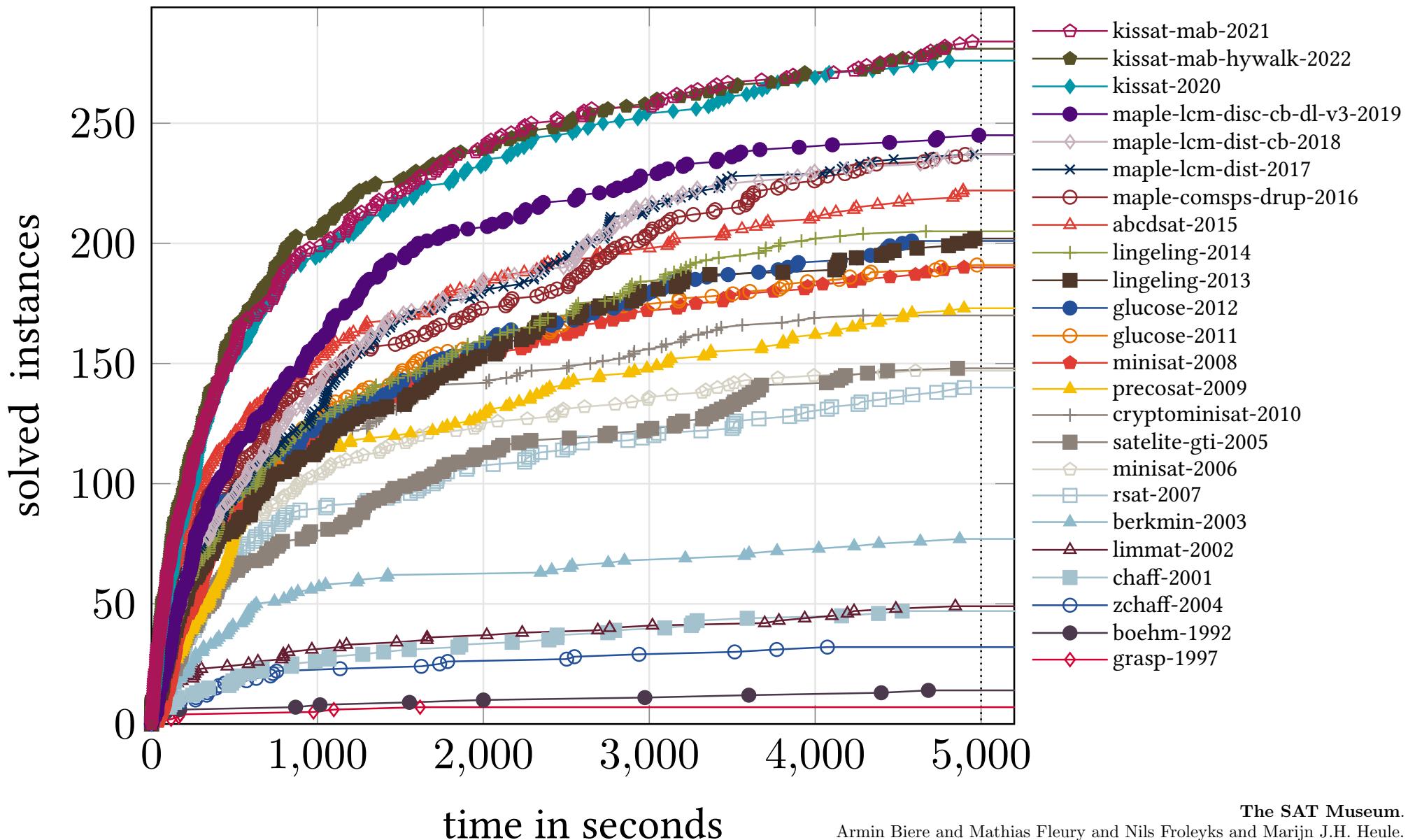
Discovered the first oblique glider rake in [@conwaylife](#) (based on a partial discovered by [@inspirehep23](#)) by running ikpx2 for 35000 core-hours on the under-utilised CPUs of a [@LeaderGPU](#) machine.

This wouldn't have been possible without [@ArminBiere](#)'s SAT solvers (radical/kissat).

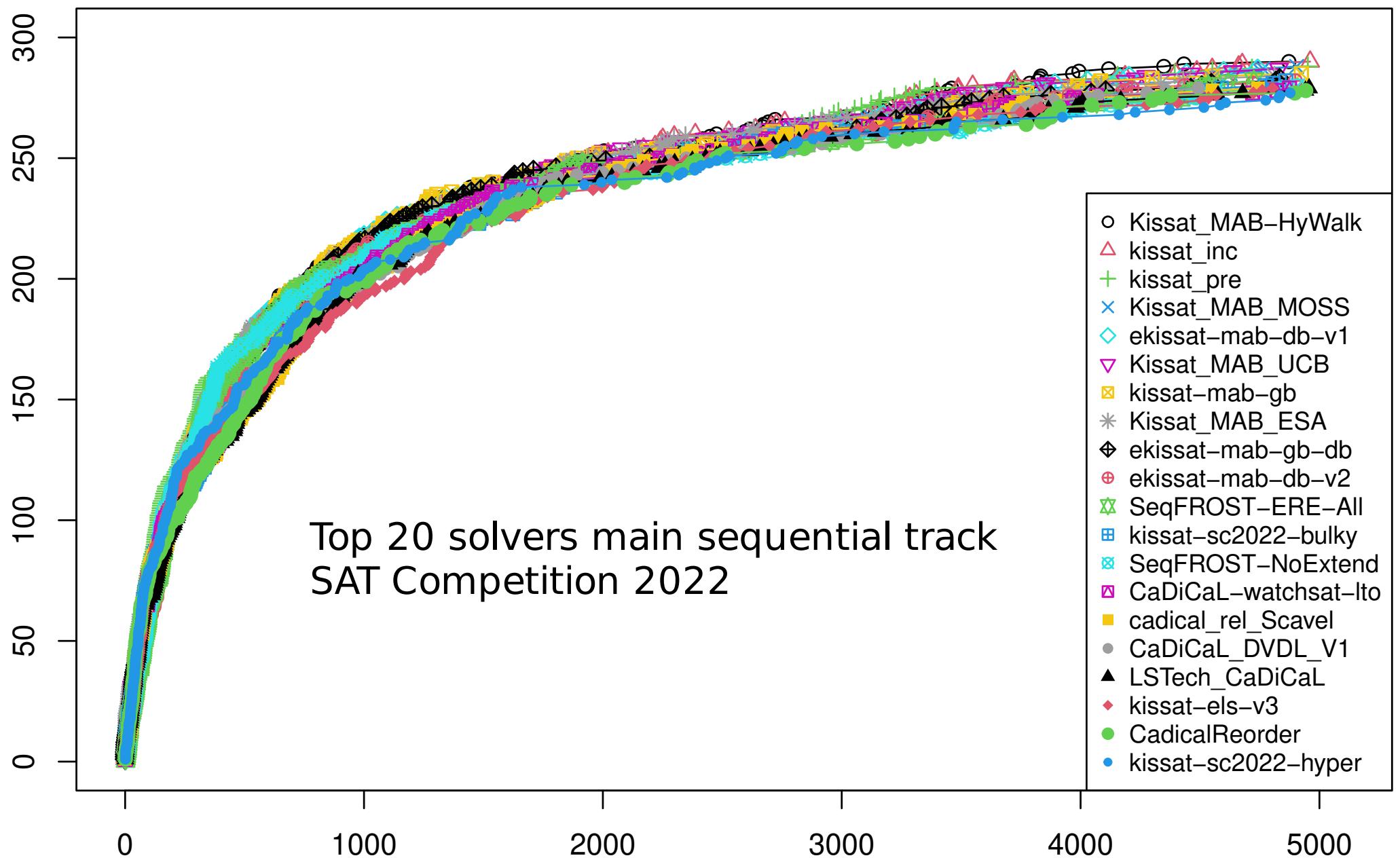
[Tweet übersetzen](#)

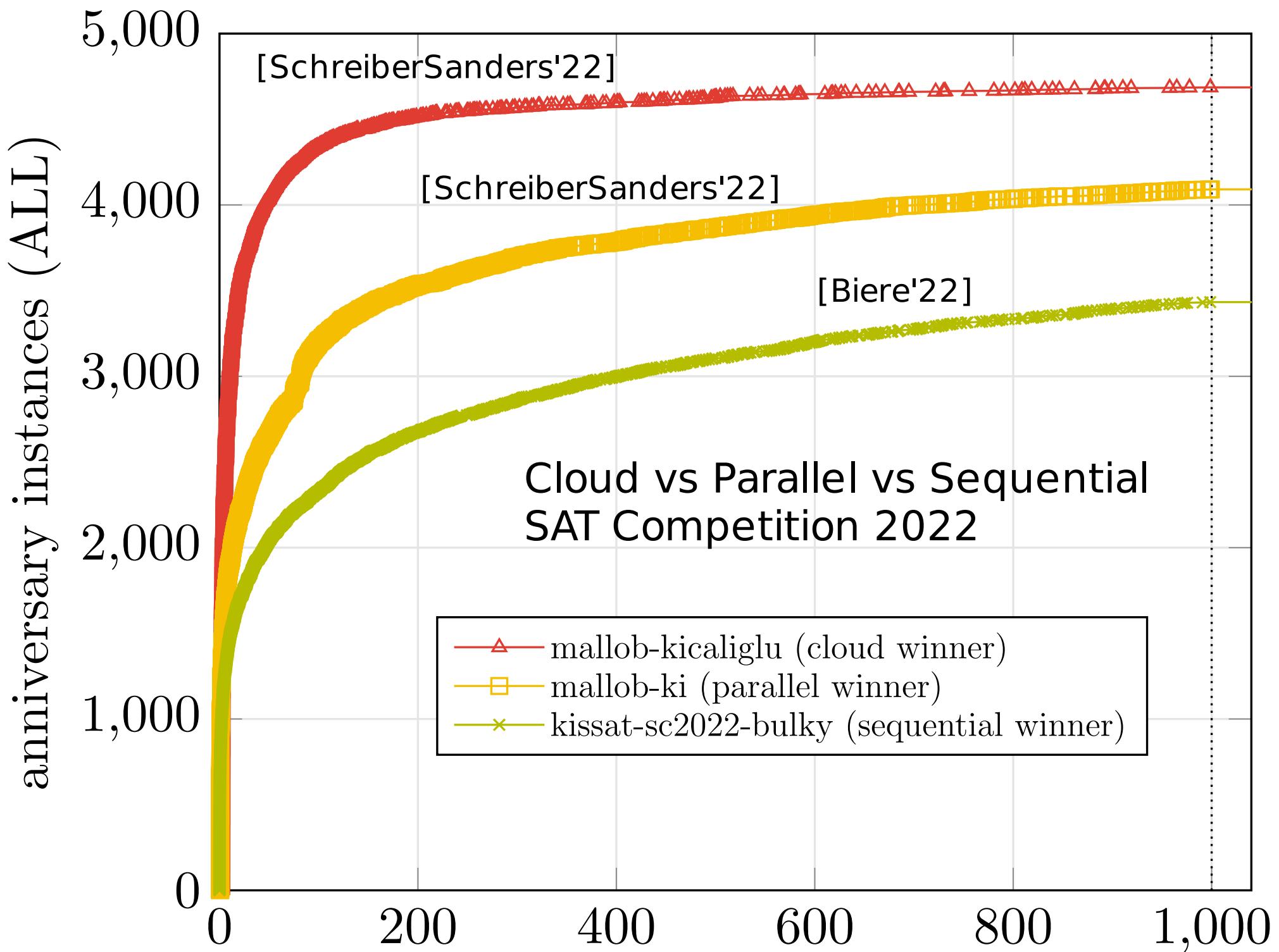


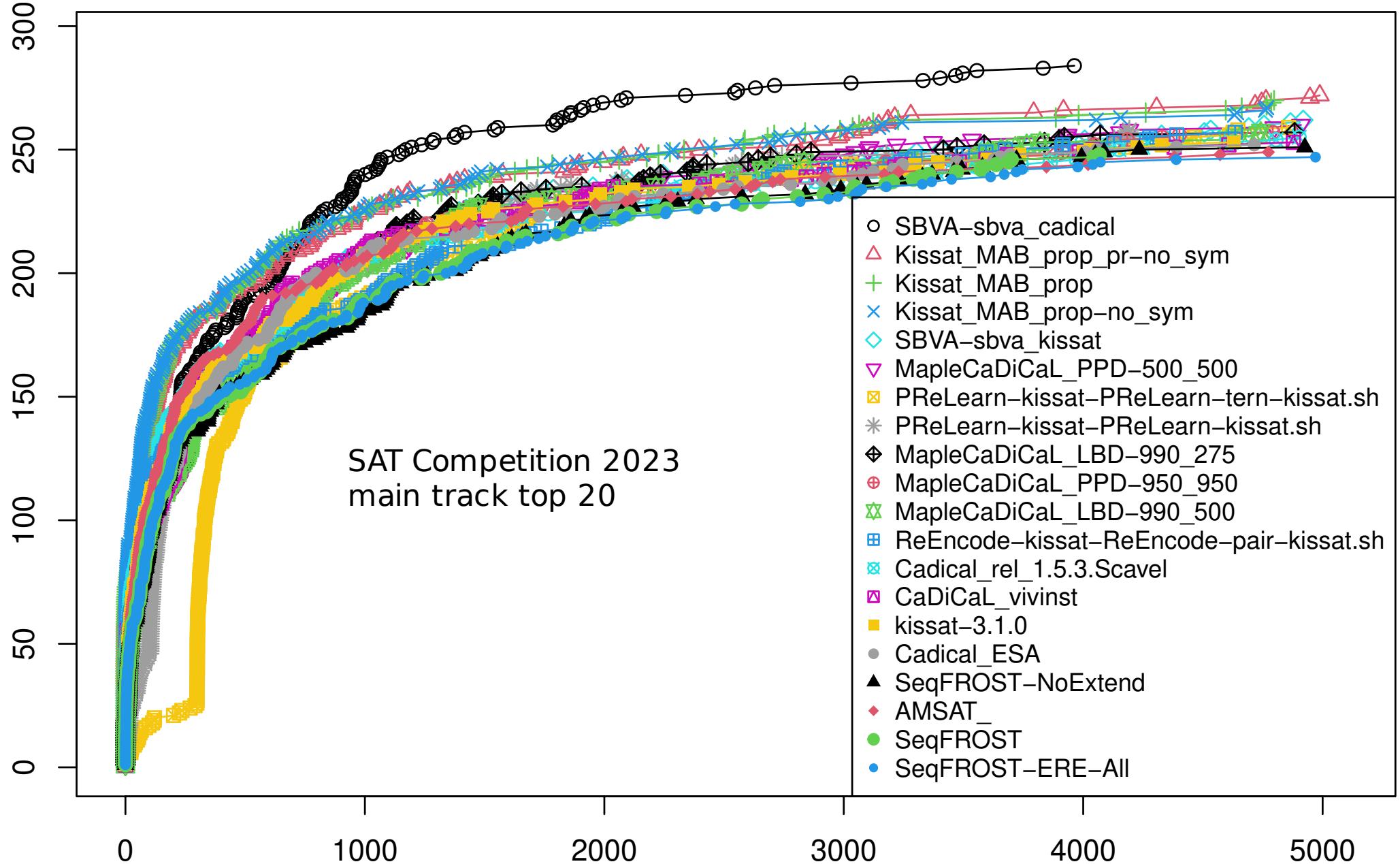
SAT Competition All Time Winners on SAT Competition 2022 Benchmarks



The SAT Museum.
Armin Biere and Mathias Fleury and Nils Froleyks and Marijn J.H. Heule.
In Proceedings 14th International Workshop on Pragmatics of SAT (POS'23),
vol. 3545, CEUR Workshop Proceedings, pages 72-87, CEUR-WS.org 2023.
[paper - bibtex - data - zenodo - ceur - workshop - proceedings]







Migrating Solver State [BiereChowdhuryHeuleKieslWhalen'SAT22]

Armin Biere   

University of Freiburg, Germany

Md Solimul Chowdhury   

Carnegie Mellon University, USA

Marijn J.H. Heule   

Carnegie Mellon University,
Amazon Web Services, Inc., USA

Benjamin Kiesl   

Amazon Web Services, Inc., Germany

Michael W. Whalen   

Amazon Web Services, Inc.,
The University of Minnesota, USA

Abstract

We present approaches to store and restore the state of a SAT solver, allowing us to migrate the state between different compute resources, or even between different solvers. This can be used in many ways, e.g., to improve the fault tolerance of solvers, to schedule SAT problems on a restricted number of cores, or to use dedicated preprocessing tools for inprocessing. We identify a minimum viable subset of the solver state to migrate such that the loss of performance is small. We then present and implement two different approaches to state migration: one approach stores the state at the end of a solver run whereas the other approach stores the state continuously as part of the proof trace. We show that our approaches enable the generation of correct models and valid unsatisfiability proofs. Experimental results confirm that the overhead is reasonable and that in several cases solver performance actually improves.

2012 ACM Subject Classification Theory of computation

Keywords and phrases SAT, SMT, Cloud Computing, Serverless Computing

← Tweet



Māris Ozols
@enclanglement

Humanity strikes back: two 5×5 matrices over \mathbb{Z}_2 require only 95 instead of 96 multiplications. Take it AlphaTensor!
arxiv.org/abs/2210.04045

DeepMind @DeepMind · 6d

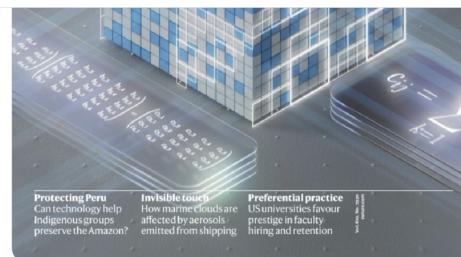
ICYMI: On the cover of @Nature - #AlphaTensor, an AI system for discovering novel, efficient, and exact algorithms for matrix multiplication.

Learn more ↗ ...



Tweet your reply

← Tweet



10:14 · 11 Oct 22 · Twitter Web App

133 Retweets 23 Quote Tweets 761 Likes

Ryan O'Donnell @BooleanAnalysis · 1d
Replying to @enclanglement
Still 'computer-assisted' if I know Manuel Kauers, but yeah, this is pretty neat :) Now we need SAT-solvers to give us better lower bounds...

Māris Ozols @enclanglement · 1d
Yes, the lines are certainly blurry here. This reminds me of "centaur chess".

Advanced chess - Wikipedia
en.wikipedia.org

Tweet your reply

← Kronen Zeitung

Teilen



0

BUNDESLÄNDER > OBERÖSTERREICH

18.10.2022 15:30

MATHE-GENIES

Mit einem Schritt zum neuen Weltrekord



Tüftler: Jakob Moosbauer und Manuel Kauers (re.) (Bild: Dostal Harald)

Den Mathematikern Manuel Kauers und Jakob Moosbauer von der JKU ist es gelungen, den erst kürzlich aufgestellten Rekord für 5×5 -Matrizen zu toppen. Sie konnten die schwierige Rechenoperation um einen Schritt verkürzen und freuen sich nun über den Weltrekord. Die beiden rechnen zudem schneller als Künstliche Intelligenz.

DER STANDARD

Wissenschaft International Inland Wirtschaft

WAS IST DIE MATRIX?

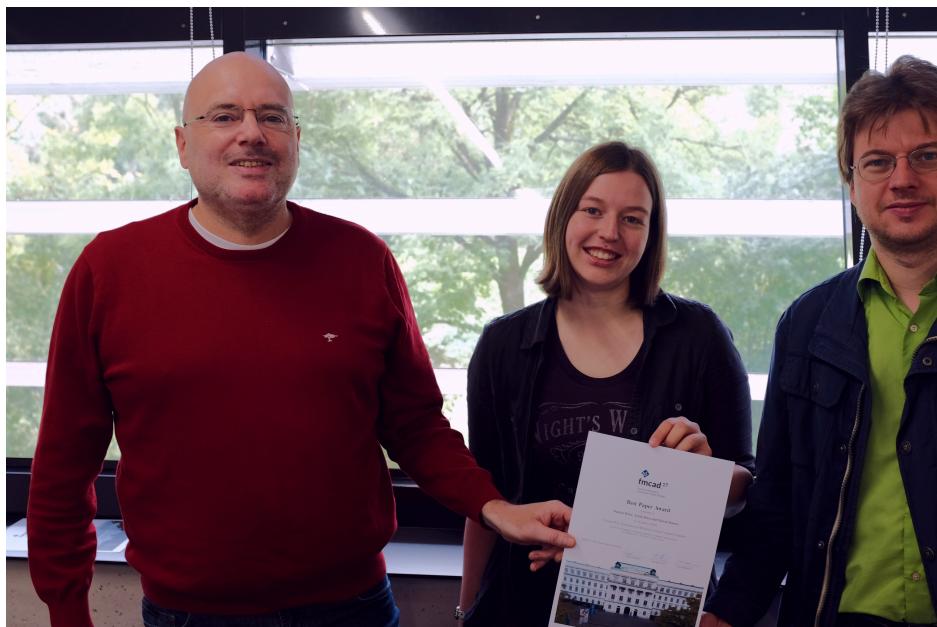
Linzer Mathematiker übertrumpfen künstliche Intelligenz

Forscher der Johannes-Keppler-Universität fanden eine neue Methode, wie man besonders effizient 5×5 -Matrizen multipliziert. Sie stechen dabei Computermethoden aus

Reinhard Kleindl
19. Oktober 2022, 14:10, 22 Postings

Was ist einfach, und was ist schwierig? Die Beantwortung dieser Frage gehört selbst zu den schwierigen Fragen. (Was wiederum die Frage aufwirft, ob sich Letzteres so einfach sagen lässt.) Darum geht es auch in der Mathematik, auf dem Gebiet der "Berechenbarkeit". Hier wird den technischsten und abstraktesten

Arithmetic Solvers



GESELLSCHAFT
FÜR INFORMATIK



Die Gesellschaft für Informatik e.V. (GI), die Oesterreichische Computer Gesellschaft (OCG) sowie die Schweizer Informatikgesellschaft (SI) verleihen

Frau Dr. Daniela Kaufmann
für ihre hervorragende Dissertation
*„Formal Verification of Multiplier Circuits
using Computer Algebra“*
den GI-Dissertationspreis 2020.

Frau Dr. Kaufmann hat aktuelle Verifikationsmethoden basierend auf Computeralgebra verbessert und neue Methoden entwickelt, die für einen gegebenen Integer-Multiplizierer auf Gatterebene vollautomatisch über dessen Korrektheit entscheiden, ohne dass die Entwicklerinnen und Entwickler manuell in den Verifikationsprozess eingreifen müssen.

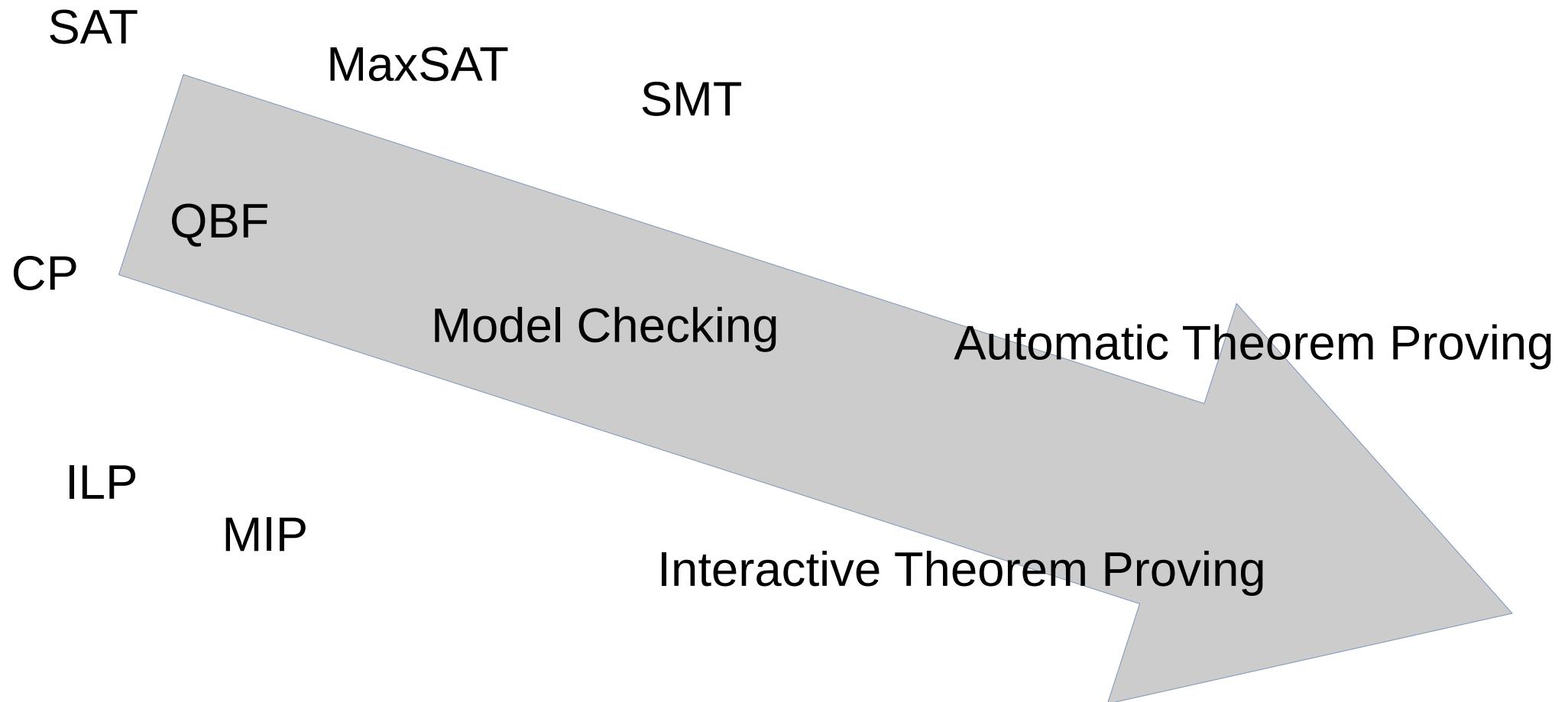
Mit dieser Preisverleihung würdigen die beteiligten Gesellschaften – die Gesellschaft für Informatik e.V. (GI), die Schweizer Informatik Gesellschaft (SI) und die Österreichische Computergesellschaft (OCG) – eine herausragende Arbeit, die eine überraschende Lösung für ein lange bekanntes Problem im Spektrum Theorie, Software und industrielle Anwendung liefert.

Berlin, im September 2021

A handwritten signature in blue ink.

Prof. Dr. Hannes Federrath
Präsident der Gesellschaft für Informatik e.V. (GI)

Automated Reasoning Technologies



Challenges

- Local Search for UNSAT
- Parallel Automated Reasoning
- Even more Scalability and Automation
- Combining Precise and Inprecise Reasoning
- Education in Logic and Automated Reasoning