

# SC<sup>2</sup>: Satisfiability Checking meets Symbolic Computation: [www.sc-square.org](http://www.sc-square.org)

James Davenport  
Hebron & Medlock Professor of Information Technology<sup>1</sup>

University of Bath (U.K.)

4 August 2016

---

<sup>1</sup>Thanks to John Abbott, Erika Ábrahám, Bernd Becker, Anna M. Bigatti, Martin Brain, Bruno Buchberger, Alessandro Cimatti, Matthew England, Pascal Fontaine, Stephen Forrest, Alberto Griggio, Daniel Kroening, Werner M. Seiler & Thomas Sturm; H2020-FETOPEN-2016-2017-CSA project 712689

# Notation

We are trying in this project to bridge two communities, that of

- ① satisfiability checking (especially “satisfiability modulo theories”) and
- ② symbolic computation

The communities have their own technical terms, which we will distinguish as above

# Satisfiability Checking

$k$ -SAT: checking whether a conjunction of disjunctions with at most  $k$  literals is satisfiable.

- The 3-SAT problem is known to be NP-complete [Coo71]

But the *Satisfiability Checking* [BBH<sup>+</sup>09] community has developed SAT solvers which can successfully handle inputs with millions of Boolean variables

- SAT solvers are in use throughout industry
- I put my life in the hands of SAT-solver verified software several times a week
- SAT-solving contests [JLBRS12] have driven much progress
- “Watched Literals” [MMZ<sup>+</sup>01] is worth a factor of  $(k - 2)$  in the inner loop

#SAT (counting solutions) is a different problem from SAT

## SAT-modulo-theories (SMT) solvers

attempt to extend this pragmatic success to cases where the literals belong to some theory, rather than being independent Booleans: example — XOR-theory

**XOR:**  $\oplus$  is a primitive in cryptography and coding theory, but translates badly to the CNF format, so we can build this in as a “theory” [LJN12]

regard  $a \oplus b$  as a new variable  $c$ : whenever  $c$  is asserted:

If  $a, b$  asserted declare  $\perp$

If  $a, b$  negated declare  $\perp$

If  $a, b$  consistent continue

If one of  $a, b$  is known, produce the appropriate value of the other as a lemma

## SAT-modulo-theories (SMT) solvers

- Substantial progress has been made when the theory is “easy” [BSST09, KS08]
- But even quantifier-free (i.e. purely existential) SMT for theories of non-linear arithmetic/algebra, real or integer, is still in its infancy
- quantified (i.e. at least one alternation) SMT is currently a dream

*“Despite substantial advances in verification technology, complexity issues with classical decision procedures are still a major obstacle for formal verification of real-world applications, e.g. in automotive and avionic industries.”* [PQR09]

## But isn't this standard computer algebra?

(at least over the reals)

- [Col75] solved quantifier elimination for the reals
- and computer algebra has made, and is making, a lot of progress since
- it's in several computer algebra systems
- and it's even possible to eliminate a quantifier on an Android 'phone [Eng14]
- Of course, it's expensive, but we know the problem is doubly-exponential [BD07]

Over the integers it's undecidable anyway, so what's the point?

## But there's a fundamental difference

**Computer Algebra** Begins with the polynomials, solves them completely (Cylindrical Algebraic Decomposition), then considers the Boolean structure

**With** some more recent flexibility, e.g. equational constraints.

**Hence** we are essentially solving #SMT, rather than SMT

**SMT** Starts from the Boolean structure, and dips into the theory, adding and retracting theory clauses as required

## There's also a question of strategy

Computer Algebra tends to have a fixed strategy

at least in terms of what is documented: the pre-processing steps before one gets into the algorithm are rarely described

Quite often follows a general algorithm even when there's some “low hanging fruit”

SAT tends to have lots of heuristics

SAT looks aggressively for low-hanging fruit [Spe15]

SAT Frequently restarts [HH10], with some underpinning theory [LSZ93]

# Heuristics

In fact, there's a great deal of choice in CAD "algorithms".

**Variable Order** The most obvious one (also present in Gröbner bases, regular chains etc.)

**Often** Crucial, in theory [BD07] and in practice

**Several** heuristics suggested in the past: [HEW<sup>+</sup>15] shows that no one heuristic is best, and a machine learning meta-heuristic outperforms all heuristics

**Equational constraints** We can only apply one for each variable, so need to choose

**No cheap** heuristics: those available do all the projections then decide which one to lift

**TTICAD** "Truth Table Invariant CAD", i.e. trying to take account of the Boolean structure, has even more choices

**Also** No research in trying to make all the choices holistically.

# Benchmarking, Problem Sets and Contests

Contests are a major factor in progress in SAT. For SMT:

**Specification** Various different questions: [WBD12] is just CAD problems, not SMT problems

**Maintenance** is a problem, see the PoSSo set of GB examples (only conserved in PDF of  $\text{\LaTeX}$ )

**Language** Not really a standard: we will extend the SMTLib standard — interested in volunteers/ interfaces; OpenDreamKit?; OpenMath; MathML-C;

**but need** a problem statement language as well as just formulae

**Industry** Not much current industrial use, so no industry problems, **vicious circle**

**Hard Problems?** Quite a challenge for SAT [Spe15]

## Hard Problems

CAD is known to be doubly-exponential (in  $n$ , the number of variables)

[DH88] Describing a single (non-trivial) solution needs polynomials of degree  $2^{2^{n/5+O(1)}}$

\* So adding  $\wedge 0 < x < 1$  makes describing a single solution doubly-exponentially more difficult

[BD07] The solutions are all rational, describable with  $2^{O(n)}$  bits. But there are  $2^{2^{O(n)}}$  of them, so SMT might be  $2^{O(n)}$  but #SMT  $2^{2^{O(n)}}$

But There is symmetry, and we don't have to count the solutions one-by-one, so what is #SMT here?

# Interrelations

The polynomial theory and the Boolean theory interact

- $fg > 0 \Leftrightarrow (f > 0 \wedge g > 0) \vee (f < 0 \wedge g < 0)$
- ? What is the rôle of square-free bases etc. here?
- $x^2 + y^2 < 1 \Rightarrow -1 < x < 1$
- ? What is the rôle of such “smplifying” (in the sense of involving fewer variables) deductions?
- ?? Might a pre-processor add such deductions?
- Different “logical” variables might involve the same “theory” variables:
- ? What is the rôle of such connections?
- $x^2 - 5x + y^2 + 2y < 5 \Rightarrow x^2 - 5x < 6$
- ? What is the rôle of such “sums of squares” deductions?

# Conclusions

We currently have two communities with different

**Terminology** Minor once you're aware of it

**Approaches** Logic-first versus (historically) polynomials-first

Also incremental versus batch

**Attitudes** Pragmatic contests versus worst-case complexity

Hence problem sets, contests, standards etc.

**Industrial links** Exist in both cases, (but currently not very well-known for either: SMT can point to SAT).

**So** We have a lot of work to do.

# Bibliography I

-  A. Biere, A. Biere, M. Heule, H. van Maaren, and T. Walsh.  
*Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*.  
IOS Press, 2009.
-  C. W. Brown and J. H. Davenport.  
The complexity of quantifier elimination and cylindrical algebraic decomposition.  
In *Proceedings ISSAC 2007*, pages 54–60. ACM, 2007.
-  C. Barrett, R. Sebastiani, S. A. Seshia, and C. Tinelli.  
Satisfiability modulo theories.  
In *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, chapter 26, pages 825–885. IOS Press, 2009.

## Bibliography II

-  G. E. Collins.  
Quantifier elimination for real closed fields by cylindrical algebraic decomposition.  
In *Automata Theory and Formal Languages*, volume 33 of *LNCS*, pages 134–183. Springer, 1975.
-  S. A. Cook.  
The complexity of theorem-proving procedures.  
In *Proceedings STOC 1971*, pages 151–158. ACM, 1971.
-  J. H. Davenport and J. Heintz.  
Real quantifier elimination is doubly exponential.  
*J. Symbolic Computation*, 5:29–35, 1988.

## Bibliography III

-  **M. England.**  
Eliminating a quantifier with sage/qepcad on android.  
Demonstration, 2014.
-  **Z. Huang, M. England, D. Wilson, J. H. Davenport, and L. C. Paulson.**  
A comparison of three heuristics to choose the variable ordering for cylindrical algebraic decomposition.  
*ACM Communications in Computer Algebra*,  
48(3/4):121–123, 2015.
-  **S. Haim and M. Heule.**  
Towards Ultra Rapid Restarts.  
Technical Report Universities of New South Wales and Delft,  
2010.

## Bibliography IV

-  M. Järvisalo, D. Le Berre, O. Roussel, and L. Simon.  
The international SAT solver competitions.  
*AI Magazine*, 33:89–92, 2012.
-  D. Kroening and O. Strichman.  
*Decision Procedures: An Algorithmic Point of View*.  
Springer, 2008.
-  T. Laitinen, T. Junttila, and I. Niemel.  
Conflict-driven XOR-clause learning.  
*Theory and Applications of Satisfiability Testing-SAT 2012*,  
pages 383–396, 2012.
-  M. Luby, A. Sinclair, and D. Zuckerman.  
Optimal Speedup of Las Vegas algorithms.  
*Inf. Proc. Letters*, 47:173–180, 1993.

## Bibliography V

-  M.W. Moskewicz, Madigan.C.F., Y. Zhao, L. Zhang, and S. Malik.  
Chaff: Engineering an Efficient SAT Solver.  
In *Proceedings 38th Design Automation Conference*, 2001.
-  A. Platzer, J.-D. Quesel, and P. Rümmer.  
Real world verification.  
In *Proceedings CADE-22*, pages 485–501. ACM, 2009.
-  I. Spence.  
Weakening Cardinality Constraints Creates Harder Satisfiability Benchmarks.  
*J. Exp. Algorithms Article 1.4*, 20, 2015.

## Bibliography VI

-  D.J. Wilson, R.J. Bradford, and J.H. Davenport.  
A Repository for CAD Examples.  
*ACM Communications in Computer Algebra* 3, 46:67–69,  
2012.