Satisfiability Checking and Symbolic Computation (SC^2)

E. Ábrahám[1], J. Abbott[11], B. Becker[2], A.M. Bigatti[3], M. Brain[10], B. Buchberger[4], A. Cimatti[5], J.H. Davenport[6], M. England[7], P. Fontaine[8], S. Forrest[9], A. Griggio[5], D. Kroening[10], W.M. Seiler[11] and T. Sturm[12]

[1]RWTH Aachen University, Aachen, Germany; [2]Albert-Ludwigs-Universität, Freiburg, Germany;
[3]Università degli studi di Genova, Italy; [4]Johannes Kepler Universität, Linz, Austria; [5]Fondazione Bruno Kessler, Trento, Italy;
[6]University of Bath, Bath, U.K.; [7]Coventry University, Coventry, U.K.; [8]LORIA, Inria, Université de Lorraine, Nancy, France;
[9]Maplesoft Europe Ltd; [10]University of Oxford, Oxford, U.K.; [11]Universität Kassel, Kassel, Germany;
[12]CNRS, LORIA, Nancy, France and Max-Planck-Institut für Informatik, Saarbrücken, Germany.

Horizon 2020
European Union Funding
for Research & Innovation

Introduction

We describe a new project to bring together the communities of *Symbolic Computation* and *Satisfiability Checking* into a new joint community, SC^2, supported by a newly accepted EU project (H2020-FETOPEN-CSA 712689) of the same name. Both communities have long histories, as illustrated by the tool development timeline below, but there is currently little interaction. However, they now share common interests in the development, implementation and application of decision procedures for arithmetic theories. By working together the communities can resolve problems, academic and industrial, currently beyond the scope of either individually.

This poster gives introductions to the two separate communities, discusses some of the challenges for the new SC^2 community, and the project actions planned for addressing them. The reader is referred to [2] for more details and full references; and the SC^2 website [A] for new information on the project as it occurs, or to get involved.

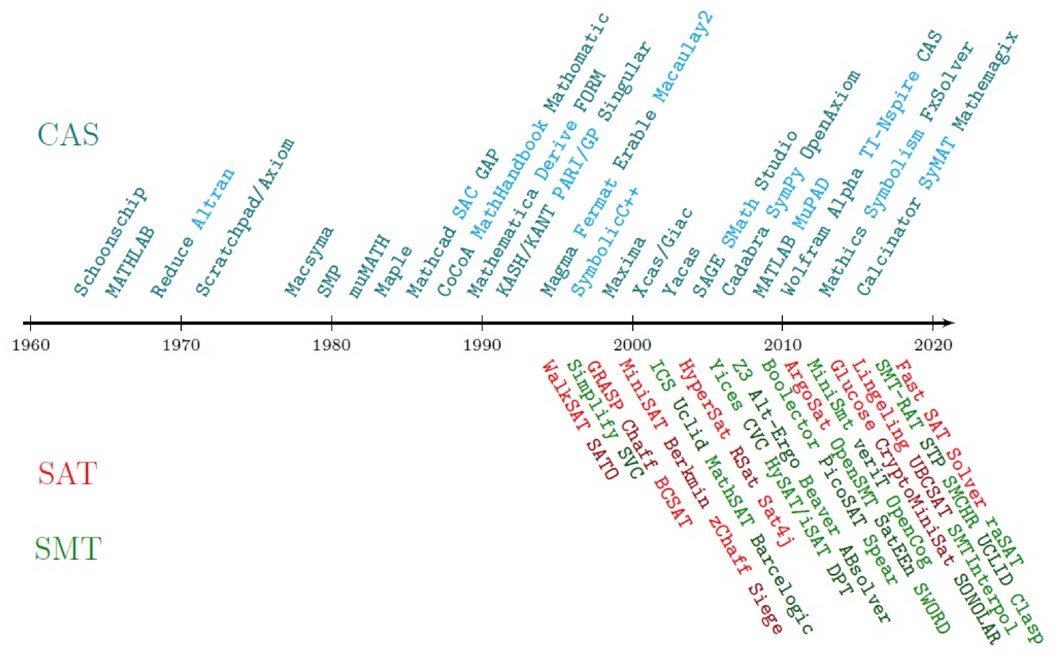

Figure 1: Potted History of Computer Algebra Systems and SAT/SMT solvers [1]

Symbolic Computation and Computer Algebra Systems

The use of computers to do algebra rather than simply arithmetic is almost as old as computing itself. Initial work consisted of programs to do one thing, but this soon led to *systems* capable of a variety of tasks. Early examples included `SAC`, `Macsyma` and `Reduce`. An early highlight was Moses's algorithm for symbolic integration paired with Risch's completeness theorem to prove un-integrability. Initially seen as part of artificial intelligence, the focus soon moved to algorithmics and complexity theory. We highlight some of the algorithms of particular relevance to this project.

The method of *Gröbner Bases* [5], which can be seen as a multivariate non-linear generalization of both Euclid's algorithm and Gaussian elimination, allows the effective and in many cases efficient solution of problems involving polynomial ideals and their associated algebraic varieties. The main commercial general-purpose computer algebra systems (including `MAGMA`, `Maple`, `Mathematica`) all have independent implementations, and there are specialised (freely downloadable) systems such as `Singular`, `Macaulay` and `CoCoA`.

Cylindrical Algebraic Decomposition (CAD) [6] is a key tool for many problems in real algebraic geometry, such as quantifier elimination. It replaced the method of Tarski which had indescribably high complexity (cannot be bounded by any finite tower of exponentials) with one of complexity doubly exponential in the number of variables. Specialist packages here include `QEPCAD-B`, the `Redlog` package for `Reduce` and the `RegularChains` Library of `Maple`. Another algorithm in this area, *Virtual Substitution* [8] offers an alternative when the degree of the quantified variables is not too large (a restriction softened by powerful heuristics).

The community is supported by conferences (e.g. ISSAC, ACA, CASC), journals (e.g. J. Symbolic Computation); and the ACM SIG in Symbolic and Algebraic Manipulation (SIGSAM).

Satisfiability Checking

The *SAT Problem* is to check the satisfiability of logical statements over the Booleans. Notable work includes that of Davis, Putnam, Logemann and Loveland who used *resolution* for quantifier elimination; and a combination of *enumeration* and *Boolean constraint propagation (BCP)*. Another major improvement was the *conflict-driven clause-learning* and *non-chronological backtracking* of Marques-Silva and Sakallah. While the SAT Problem is known to be NP-complete, SAT solvers have been developed which can handle inputs with millions of Boolean variables, and are used in many industrial applications, e.g. in verification and security.

Driven by this success, big efforts were made to enrich propositional SAT-solving for different existentially quantified theories producing *SAT-modulo-theories (SMT) solvers* [4]. There exist techniques for equality logic with uninterpreted functions, array theory, bit-vector arithmetic and quantifier-free linear real and integer arithmetic. See [3] for an introduction which discusses the highlights so far. However, the development for quantifier-free non-linear real and integer arithmetic is still in its infancy and progress here is required for applications in the automotive and avionic industries [7].

SMT solvers typically combine a *SAT solver* with *theory solvers* as illustrated in Figure 2. A formula in conjunctive normal form is abstracted to one of pure Boolean propositional logic by replacing each theory constraint by a fresh proposition. The SAT solver tries to find solutions for this, consulting the theory solver(s) to check the consistency of constraints. To be *SMT-compliant* the solvers should:

- work *incrementally*, i.e. accept additional constraints and recheck making use of previous results;
- support *backtracking*, i.e. the removal of previously added constraints;
- in case of unsatisfiability return an *explanation*, e.g. a small inconsistent subset of constraints.

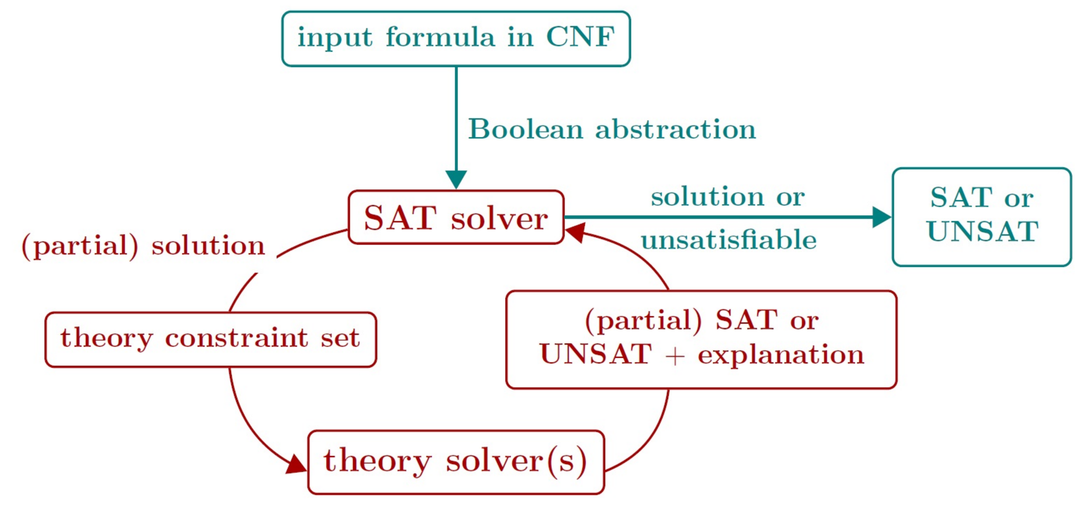

Figure 2: The typical functioning of SMT solvers

Solvers that are able to cope with linear arithmetic problems include `Alt-Ergo`, `CVC4`, `iSAT3`, `MathSAT`, `OpenSMT2`, `SMT-RAT`, `veriT`, `Yices2`, and `Z3`. Far fewer tools exist for non-linear arithmetic: `iSAT3` uses interval constraint propagation; `MiniSmt` tries to reduce problems to linear real arithmetic; `Z3` uses an adaptation of the CAD method; while `SMT-RAT` uses solver modules for CAD, virtual substitution, Gröbner bases, interval constraint propagation and branch-and-bound. Even fewer SMT solvers are available for non-linear integer arithmetic.

The community is supported by conferences (e.g. CADE, IJCAR, SMT) and journals (e.g. J. Automated Reasoning); while a role somewhat analogous to SIGSAM is played by the SatLive Forum.

Project Challenges and Opportunities

SMT solving has its strength in efficient techniques for exploring Boolean structures, learning, combining techniques, and developing dedicated heuristics. Symbolic Computation provides powerful procedures for sets of arithmetic constraints, and has expertise in simplification and preprocessing.

To allow further exploitation by the Satisfiability Checking community, Symbolic Computation tools must first be adapted to comply with SMT requirements as set out in the previous section. Cylindrical Algebraic Decomposition, Gröbner Bases and Virtual Substitution are algorithms of particular interest. However, this is a challenge that requires the expertise of computer algebra developers.

Conversely, Symbolic Computation could profit from exploiting successful SMT ideas, like dedicated data structures, sophisticated heuristics, effective learning techniques, and approaches for instrumentality and explanation generation. Incremental CAD procedures now exist, as do prototypes integrating CDCL-style learning techniques with virtual substitution for linear quantifier elimination.

We are creating a new research community SC^2 whose members will ultimately be informed about both fields, and thus able to combine knowledge and techniques to resolve problems (academic and industrial) currently beyond the scope of either individually. To achieve this we have an EU Horizon 2020 Coordination and Support Action project (712689) with start date July 2016. Actions include:

Communication platforms: We have started to initiate joint meetings: In 2015 a Dagstuhl Seminar [B] was dedicated to SC^2; at ACA 2016 and CASC 2016 there will be SC^2 topical sessions; and then the first annual SC^2 workshop will take place in affiliation with SYNASC 2016 [C]. Other plans include a summer school for young researchers interested in the area.

Research roadmap: The above platforms will initiate cross-community interactions. Our long-term objective is to create a research roadmap of opportunities and challenges; identifying within the problems currently faced in industry, points that can be expected to be solved by the SC^2 community.

Standards We aim to create a standard problem specification language for the SC^2 community, extending the *SMT-LIB* language to handle features needed for Symbolic Computation. This could serve as a communication protocol for platforms that mix tools; and will be used to develop a set of benchmarks. Agreeing on a common language, and being able to share challenging problems is an essential aspect for building a dynamic community.

Your Involvement The project consists of not just the partner institutions but also associates from both EU and non-EU research institutions and industry. Associates will be regularly informed about project activities and invited to corresponding events. If you would like to participate please contact the Project Coordinator James Davenport (J.H.Davenport@bath.ac.uk).

References

[1] E. Ábrahám. *Building bridges between symbolic computation and satisfiability checking.* In: Proceedings of the 2015 International Symposium on Symbolic and Algebraic Computation (ISSAC '15), pages 1–6. ACM, 2015.

[2] E. Ábrahám, J. Abbott, B. Becker, A.M. Bigatti, M. Brain, B. Buchberger, A. Cimatti, J.H. Davenport, M. England, P. Fontaine, S. Forrest, A. Griggio, D. Kroening, W.M. Seiler and T. Sturm. *SC2: Satisfiability Checking meets Symbolic Computation (Project Paper).* To Appear In: Intelligent Computer Mathematics (Proc. CICM '16), 2016. Preprint: `http://www.sc-square.org/Papers/CICM16.pdf`

[3] C. Barrett, D. Kroening, and T. Melham. *Problem solving for the 21st century: Efficient solvers for satisfiability modulo theories.* Technical report, London Mathematical Society and Smith Institute for Industrial Mathematics and System Engineering, 2014.

[4] C. Barrett, R. Sebastiani, S.A. Seshia and C. Tinelli. *Satisfiability modulo theories.* In: Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications, 185, p825–885. IOS Press, 2009

[5] B. Buchberger. *Ein Algorithmus zum Auffinden des Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal.* PhD Thesis, University of Innsbruck, 1965. English translation: Journal of Symbolic Computation, 41:475–511, 2006.

[6] G.E. Collins. *Quantifier elimination for real closed fields by cylindrical algebraic decomposition.* In: Automata Theory and Formal Languages (LNCS 33), pages 134–183. Springer, 1975.

[7] A. Platzer, J.D. Quesel & P. Rümmer. *Real world verification.* Proc. CADE-22. p485–501. ACM, 2009

[8] V. Weispfenning. *Quantifier elimination for real algebra The quadratic case and beyond.* Applicable Algebra in Engineering, Communication and Computing, 8(2):85–101, 1997.

[A] `http://www.sc-square.org`

[B] `http://www.dagstuhl.de/en/program/calendar/semhp/?semnr=15471`

[C] `http://www.sc-square.org/CSA/workshop1.html`