Satisfiability Checking and Symbolic Computation

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.

Abstract

Symbolic Computation and *Satisfiability Checking* are viewed as individual research areas, but they share common interests in the development, implementation and application of decision procedures for arithmetic theories. Despite these commonalities, the two communities are currently only weakly connected. We introduce a new project SC^2 to build a joint community in this area, supported by a newly accepted EU (H2020-FETOPEN-CSA) project of the same name. We aim to strengthen the connection between these communities by creating common platforms, initiating interaction and exchange, identifying common challenges, and developing a common roadmap. This abstract and accompanying poster describes the motivation and aims for the project, and reports on the first activities.

1 Introduction

We describe a new project to bring together the communities of ***S**ymbolic **C**omputation* and ***S**atisfiability **C**hecking* into a new joint community, SC^2. Both communities have long histories, as illustrated by the tool development timeline in Figure 1, but traditionally they do not interact much even though they are now individually addressing similar problems in non-linear algebra. In Section 2 we give an introduction to *Satisfiability Checking* (a corresponding introduction to Symbolic Computation is omitted given the audience for this abstract). We then discuss some of the challenges for the new SC^2 community in Section 3 and the project actions planned to address them. The reader is referred to [2] for more details and full references; and the SC^2 website (`http://www.sc-square.org`) for new information as it occurs.

2 Satisfiability Checking

The *SAT Problem* refers to checking the satisfiability of logical statements over the Booleans. Initial ideas from Davis and Putnam in 1960 used *resolution* for quantifier elimination; Davis, Logemann and Loveland pursued another line in 1962 with a combination of *enumeration* and *Boolean constraint propagation (BCP)*. A major improvement was achieved in 1999 by Marques-Silva and Sakallah by *combining* the two approaches, leading to *conflict-driven clause-learning* and *non-chronological backtracking*. While the SAT Problem is known to be NP-complete, SAT solvers have been developed which can handle inputs with millions of Boolean variables. They are at the heart of industrial techniques for verification and security.

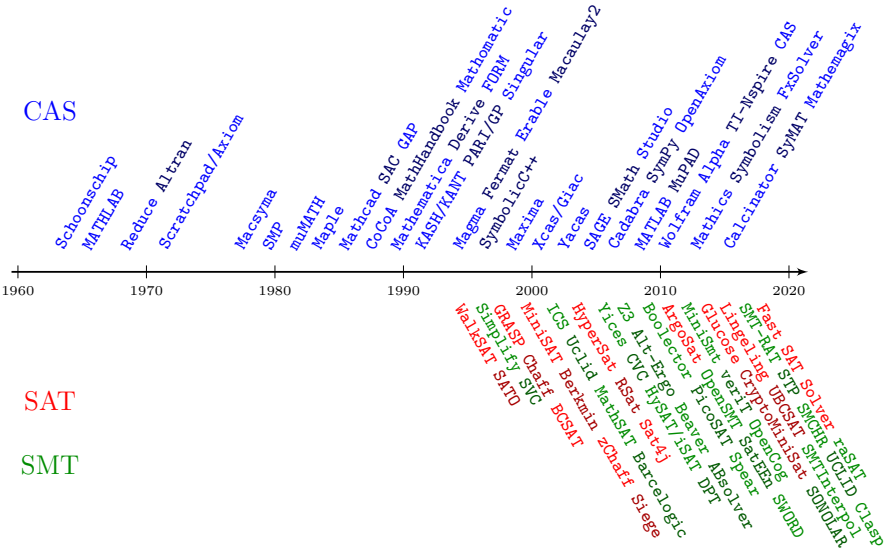

Figure 1: History of Computer Algebra Systems and SAT/SMT solvers [1]

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

SMT solvers typically combine a *SAT solver* with one or more *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 re-check 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.

Examples for solvers that are able to cope with linear arithmetic problems are `Alt-Ergo`, `CVC4`, `iSAT3`, `MathSAT`, `OpenSMT2`, `SMT-RAT`, `veriT`, `Yices2`, and `Z3`. Far fewer tools exist for non-linear arithmetic: `iSAT3`

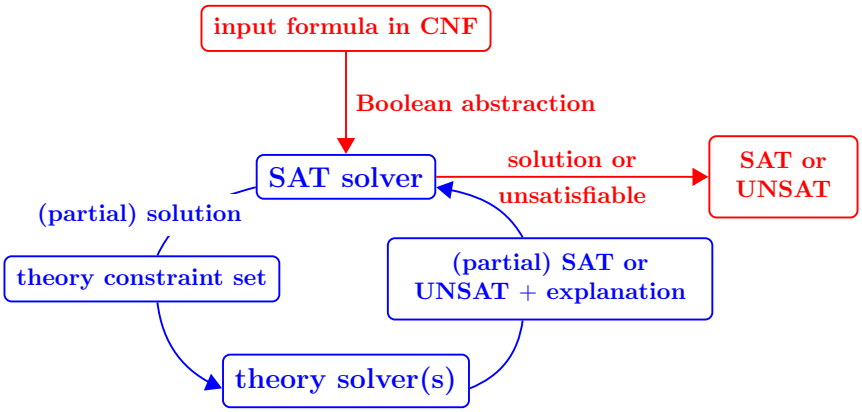

Figure 2: The typical functioning of SMT solvers

uses interval constraint propagation; `MiniSmt` tries to reduce problems to linear real arithmetic; `Z3` uses an adaptation of the cylindrical algebraic decomposition (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 (undecidable in general).

3 Challenges and Opportunities

SMT solving has its strength in efficient techniques for exploring Boolean structures, learning, combining techniques, and developing dedicated heuristics. Symbolic Computation is strong in providing 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 (CAD, 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 aim to create a new research community SC^2 whose members will ultimately be well 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 an EU Horizon 2020 *Coordination and Support Action* (712689) project started in July 2016. We plan the following actions:

Communication platforms: Like Symbolic Computation, Satisfiability Checking is supported by its own conferences (e.g. CADE, IJCAR, SMT) and journals (e.g. JAR); while a role somewhat analogous to SIGSAM is played by the SatLive Forum (`http://www.satlive.org/`). We have started to initiate joint meetings: in 2015 a Dagstuhl Seminar[1] was dedicated to SC^2; at ACA 2016 and CASC 2016 there were SC^2 topical sessions; and the first annual SC^2 workshop took place in affiliation with SYNASC 2016[2]. In 2017 we plan the second SC^2 workshop and 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.

References

[1] E. Ábrahám. *Building bridges between symbolic computation and satisfiability checking.* In: Proceedings ISSAC '15. p1–6. 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.* In: M. Kohlhase et al. eds., Intelligent Computer Mathematics (Proc. CICM '16), LNCS 9791, p28–43. Springer 2016.

[3] 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

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

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

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