EPiC Series in Computing
Volume 51, 2017, Pages 6–10
ARCADE 2017. 1st International Workshop on Automated Reasoning: Challenges, Applications, Directions, Exemplary Achievements

# $\mathsf{SC}^2$ challenges: when Satisfiability Checking and Symbolic Computation join forces

Erika Ábrahám[1], John Abbott[10], Bernd Becker[2], Anna M. Bigatti[3], Martin Brain[9], Alessandro Cimatti[4], James H. Davenport[5], Matthew England[6], Pascal Fontaine[7], Stephen Forrest[8], Vijay Ganesh[11], Alberto Griggio[4], Daniel Kroening[9], and Werner M. Seiler[10]

[1] RWTH Aachen University, Aachen, Germany
[2] Albert-Ludwigs-Universität, Freiburg, Germany
[3] Università degli studi di Genova, Italy
[4] Fondazione Bruno Kessler, Trento, Italy
[5] University of Bath, Bath, U.K.
[6] Coventry University, Coventry, U.K.
[7] LORIA, Inria, Université de Lorraine, Nancy, France
[8] Maplesoft Europe Ltd
[9] University of Oxford, Oxford, U.K.
[10] Universität Kassel, Kassel, Germany
[11] University of Waterloo, Ontario, Canada

**Abstract**

*Symbolic Computation* and *Satisfiability Checking* are two research areas, both having their individual scientific focus but with common interests, e.g., in the development, implementation and application of decision procedures for arithmetic theories. Despite their commonalities, the two communities are rather weakly connected. The aim of the $\mathsf{SC}^2$ initiative is to strengthen the connection between these communities by creating common platforms, initiating interaction and exchange, identifying common challenges, and developing a common roadmap from theory along the way to tools and (industrial) applications.

## 1 Introduction

The use of advanced methods to solve practical and industrially relevant problems by computers has a long history. While it is customary to think that “computers are getting faster” (and indeed, they were, and are still getting more powerful in terms of multicores etc.), the progress in algorithms and software has been even greater. One of the leaders in the field of linear and mixed integer programming points out [8, slide 37] that you would be over 400 times better off running today’s algorithms and software on a 1991 computer than you would running 1991 software on today’s computer. The practice is heavily inspired by the theory: [8, slide 31] shows that the biggest version-on-version performance advance in software was caused by “mining the theory". *But* this progress has been in what is, mathematically, quite a limited domain: that of linear programming, possibly where some of the variables are integer-valued.

There has been also much progress in the use of computers to solve hard non-linear algebraic[1] problems, generally but not exclusively associative and commutative. This is the area generally called *Symbolic Computation* (or *Computer Algebra*). It includes solving non-linear problems over both the real and complex numbers, though generally with very different techniques. This has produced many new applications and surprising developments: in an area everyone believed was solved, non-linear solving over the reals (using cylindrical algebraic decomposition — CAD) has recently found a new algorithm for computing square roots [15]. CAD is another area where practice is (sometimes) well ahead of theory: the theory [14, 9] states that the complexity is doubly exponential in the number of variables, but useful problems can still be solved in practice ([4] points out that CAD is the most significant engine in the "Todai robot" project).

Independently and contemporaneously, there has been a lot of practical progress in solving the SAT problem, i.e., checking the satisfiability of logical problems over the Boolean domain. The SAT problem is known to be NP-complete [12]. Nevertheless, the *Satisfiability Checking* [7] community has developed SAT solvers which can successfully handle inputs with millions of Boolean variables. Among other industrial applications, these tools are now at the heart of many techniques for verification and security of computer systems.

Driven by this success, big efforts were made to enrich propositional SAT-solving with solver modules for different theories. Highly interesting techniques were implemented in *SAT-modulo-theories (SMT) solvers* [5, 17] for checking easier theories, but the development for quantifier-free non-linear real and integer arithmetic[1] is still in its infancy.

The $\mathsf{SC}^2$ (**S**atisfiability **C**hecking and **S**ymbolic **C**omputation) initiative, described further in [3], aims at bridging these two communities, so that members are well informed about both fields, and thus able to combine the knowledge and techniques of both fields to develop new research and to resolve problems (both academic and industrial) currently beyond the scope of either individual field. Formally, it is also a European Horizon 2020 *Coordination and Support Action*, running from July 2016 to September 2018: see `http://www.sc-square.org`.

The main section of this short document is dedicated to a few research directions that we believe could be tackled successfully by a unified $\mathsf{SC}^2$ community, as well as a few recent advances as examples of low hanging fruits at the interface of Symbolic Computation and Satisfiability Checking. Some upcoming actions of the $\mathsf{SC}^2$ initiative are advertised in conclusion.

## 2 A Few Challenges and Recent Advances

**Ordering of Variables in Boolean Logic vs. Theories.** The CAD method fixes a *static* order of the theory variables and stays with it for the whole computation; changing this order is very expensive (up to doubly exponential [9]). Conversely, *dynamic* variable ordering for logic variables is one of the key reasons why SAT solvers are so efficient [19, 18]. This has been witnessed by dramatic recent progress in dynamic variable ordering techniques in SAT solvers through use of online machine learning [18]. However, similar sophisticated dynamic ordering techniques for theory variables do not yet exist. In [6], a mechanism to influence theory variable ordering is introduced: it gives higher preference to simple theory branches over more complex ones. The authors currently know of no deeper theory-specific mechanisms for directing the

[1] It is usual in the SMT community to refer to these constraints as *arithmetic*. But, as they involve quantities as yet unknown, manipulating them is *algebra*. Hence both words occur, with essentially the same meaning, throughout this document.

search based on observations made during previous theory checks, or research directly relating the orderings of the logic and theory variables.

**Inexpensive Theory Deductions.** There may be deductions which are easy in the theory world, but could greatly improve the logical process, or enable the logical process to use simpler reasoning. For example, suppose $x^2 + y^2 \leq 4$ is one of the terms in our proposition. This is a nonlinear constraint involving two theory variables, and as such is relatively hard to handle. But $x^2 + y^2 \leq 4 \Rightarrow (x \geq -2) \wedge (x \leq 2)$, and the implicand is linear and only involves one variable. This is an example of a deduction which a theory system can make (simple projection if we are using CAD as our theory engine). The high potential of such deductions is not yet well exploited.

**Weakly Nonlinear Reasoning.** When embedded as reasoning engines in formal verification tools, SMT solvers are typically required to provide functionalities beyond pure satisfiability checking like, e.g., incremental solving or the generation of models, unsatisfiable cores or Craig interpolants. Though relevant progress has been made recently in SMT solving for non-linear arithmetic (e.g. [16, 13]), current approaches do not yet fully satisfy the needs of formal verification tools.

Motivated by the observation that in many important application domains systems are "mostly-linear", the authors of [11] propose a counterexample-guided abstraction refinement approach to work with abstractions expressed over linear arithmetic with uninterpreted functions, where nonlinear multiplication is modeled as an uninterpreted function. If the solver finds a solution for the linear abstraction which does not satisfy the concrete non-linear problem (i.e., a spurious counterexample), then the abstraction is tightened by adding new linear constraints, including tangent planes resulting from differential calculus, and monotonicity constraints. The approach is implemented on top of the nuXmv model checker [10].

**Combining Decision Procedures.** Its doubly-exponential complexity restricts the practical applicability of the CAD method, the only available complete decision procedure for real arithmetic. Thus for effective reasoning it is extremely important to exploit other methods for preprocessing and solving parts of a given problem with other, more efficient methods. For example, incomplete but fast Interval Arithmetic can be used to reduce the search space, the Simplex method to check the linear part of the problem for satisfiability, or the Virtual Substitution method to eliminate variables that appear quadratically with less effort. A unique feature of the SMT-RAT solver [13] is that it allows the user to define her own strategic combination of decision procedures, optimised towards the given problem type, and with the possibility to exploit parallelisation.

# 3 Conclusion

In this paper we discussed some common challenges and some examples of recent advances in the areas of Satisfiability Checking and Symbolic Computation. In [3] we reported on the activities of our $\mathsf{SC}^2$ project to strengthen the connections between these areas. To mention the most important measures, as a platform for interdisciplinary communication and exchange, we initiated an annual $\mathsf{SC}^2$ workshop [2]. To train PhD students and young scientists, the $\mathsf{SC}^2$ project also organises a summer school [1]. Last but not least, to support tool development, testing and comparison, we work on extensions of the SMT-LIB language (http://smtlib.cs.uiowa.edu/) and benchmark collection to strengthen support for non-linear arithmetic.

The project consists of not just the partner institutions but also associates from both EU and non-EU research institutions and industry. Associates are 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).

**Acknowledgements**.

We are grateful for support by the H2020-FETOPEN-2016-2017-CSA project SC$^2$ (712689) and the ANR project ANR-13-IS02-0001-01 SMArT.

# References

[1] SC$^2$ Summer School. http://www.sc-square.org/CSA/school/.

[2] Second International Workshop on Satisfiability Checking and Symbolic Computation. http://www.sc-square.org/CSA/workshop2.html.

[3] E. Ábrahám, B. Becker, A. Bigatti, B. Buchberger, C. Cimatti, J.H. Davenport, M. England, P. Fontaine, S. Forrest, D. Kroening, W. Seiler, and T. Sturm. SC$^2$: Satisfiability Checking meets Symbolic Computation (Project Paper). In *Proceedings CICM 2016*, volume 9791 of *LNCS*, pages 28–43. Springer, 2016.

[4] N. H. Arai, T. Matsuzaki, H. Iwane, and H. Anai. Mathematics by machine. In *Proceedings ISSAC 2014*, pages 1–8. ACM, 2014.

[5] 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.

[6] M. Berzish, Y. Zheng, and V. Ganesh. Z3str3: A string solver with theory-aware branching. http://arXiv.org/abs/1704.07935, 2017.

[7] 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.

[8] R. E. Bixby. Computational progress in linear and mixed integer programming. *Presentation at ICIAM 2015*, 2015.

[9] 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.

[10] R. Cavada, A. Cimatti, M. Dorigatti, A. Griggio, A. Mariotti, A. Micheli, S. Mover, M. Roveri, and S. Tonetta. The nuXmv symbolic model checker. In *Proceedings CAV 2014*, LNCS. Springer, 2014.

[11] A. Cimatti, A. Griggio, A. Irfan, M. Roveri, and R. Sebastiani. Invariant Checking of NRA Transition Systems via Incremental Reduction to LRA with EUF. In *Proceedings TACAS 2017*, volume 10205 of *LNCS*. Springer, 2017. To appear.

[12] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings STOC 1971*, pages 151–158. ACM, 1971.

[13] F. Corzilius, G. Kremer, S. Junges, S. Schupp, and E. Ábrahám. SMT-RAT: An open source C++ toolbox for strategic and parallel SMT solving. In *Proceedings SAT 2015*, volume 9340 of *LNCS*, pages 360–368. Springer, 2015.

[14] J. H. Davenport and J. Heintz. Real quantifier elimination is doubly exponential. *J. Symbolic Computation*, 5:29–35, 1988.

[15] M. Eraşcu and H. Hong. Synthesis of optimal numerical algorithms using real quantifier elimination (Case study: Square root computation). In *Proceedings ISSAC 2014*, pages 162–169. ACM, 2014.

[16] D. Jovanović and L. de Moura. Solving non-linear arithmetic. In *Proceedings IJCAR 2012*, volume 7364 of *LNAI*, pages 339–354. Springer, 2012.

[17] D. Kroening and O. Strichman. *Decision Procedures: An Algorithmic Point of View*. Springer, 2008.

[18] J. H. Liang, V. Ganesh, P. Poupart, and K. Czarnecki. Exponential Recency Weighted Average Branching Heuristic for SAT Solvers. In *Proceedings of AAAI-16*, 2016.

[19] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an Efficient SAT Solver. In *Proceedings of the 38th Annual Design Automation Conference*, DAC '01, pages 530–535, New York, NY, USA, 2001. ACM.