Proving UNSAT in SMT: The Case of Quantifier Free Non-Linear Real Arithmetic

Erika Ábrahám[1], James H. Davenport[2]*, Matthew England[3]†, and Gereon Kremer[4]

[1] RWTH Aachen University, 52062 Aachen, Germany
abraham@cs.rwth-aachen.de
[2] University of Bath, Bath BA2 7AY, United Kingdom
J.H.Davenport@bath.ac.uk
[3] Coventry University, Coventry, CV1 5FB, United Kingdom
Matthew.England@coventry.ac.uk
[4] Stanford University, Stanford, California 94305, United States
gkremer@cs.stanford.edu

Abstract

We discuss the topic of unsatisfiability proofs in SMT, particularly with reference to quantifier free non-linear real arithmetic. We outline how the methods here do not admit trivial proofs and how past formalisation attempts are not sufficient. We note that the new breed of local search based algorithms for this domain may offer an easier path forward.

1 Introduction

Since 2013, SAT Competitions have required certificates for unsatisfiability which are verified offline [HJS18]. As the SAT problems tackled have grown larger, and the solvers have grown more complicated, such proofs have become more important for building trust in these solvers. The SAT community has agreed on DRAT as a common format for presenting such proofs (although within this there are some flavours [RB19]).

The SMT community has long recognized the value of proof certificates, but alas producing them turned our to be much more difficult than for the SAT case. The current version of the SMT-LIB Language (v2.6) [BFT16] specifies API commands for requesting and inspecting proofs from solvers but sets no requirements on the form those proofs take. In fact on page 66 it writes explicitly: "*The format of the proof is solver-specific*". We assume that this is a place holder for future work on an SMT-LIB proof format, rather than a deliberate design. The paper [BdF15] summarises some of the requirements, challenges and various approaches taken to proofs in SMT. Key projects that have been working on this issue include LFSC [SRT+12] and veriT [BBFF20], but there has not been a general agreement in the community yet.

Our long-term vision is that an SMT solver would be able to emit a "proof" that covers both the Boolean reasoning and the theory reasoning (possibly from multiple theories) such that a theorem prover (or a combination of multiple theorem provers) could verify its correctness, where the inverted commas indicate that some programming linkage between the theorem provers might be necessary. We would still be some way from having a fully verified one-stop checker as in GRAT [Lam20], but would be a lot closer to it than we are now.

*Support of EPSRC (Grant EP/T015713/1) is gratefully acknowledged.
†Support of EPSRC (Grant EP/T015748/1) is gratefully acknowledged.

In [BdF15] the authors explain that since in SMT the propositional and theory reasoning are not strongly mixed, an SMT proof can be an interleaving of SAT proofs and theory reasoning proofs in the shape of a Boolean resolution tree whose leaves are clauses. They identify the main challenge of proof production as keeping enough information to produce proofs, without hurting efficiency too much. This may very well be true for many cases, but for the area of interest for the authors, `QF_NRA` (Quantifier-Free Nonlinear Real Arithmetic), there is the additional challenge of providing the proofs of the theory lemmas themselves.

2 Quantifier Free Non-Linear Real Arithmetic

`QF_NRA` typically considers a logical formula Φ where the literals are statements about the signs of polynomials with rational coefficients, i.e. $f_i(x_1, \ldots, x_n)\sigma_i 0$ with $\sigma_i \in \{=, \neq, >, \geq, <, \leq\}$.

Any SMT solver which claims to tackle this logic completely relies in some way on the theory of Cylindrical Algebraic Decomposition (CAD). This was initiated by Collins [Col75] in the 1970s with many subsequent developments since: see for example the collection [CJ98] or the introduction of the recent paper [EBD20]. The key idea is to decompose infinite space $\mathbb{R}^n$ into a finite number of disjoint regions upon each of which the truth of the constraints is constant. This may be achieved by decomposing to ensure the signs of the polynomials involved are invariant, although optimisations can produce a coarser, and thus cheaper, decomposition.

In the case of unsatisfiability an entire CAD truth invariant for the constraints may be produced, and the solver can check that the formula is unsatisfiable for a sample of each cell. How may this be verified? The cylindrical condition[1] means that checking our cells decompose the space is trivial, but the fact that the constraints have invariant truth-value is a deep implication of the algorithm, not necessarily apparent from the output.

Past QF_NRA Formalisation Attempts

There was a project in Coq to formalise Quantifier Elimination in Real Closed Fields. This may also be tackled by CAD, and of course has SMT in `QF_NRA` as a sub-problem. Work began on an implementation of CAD in Coq with some of the underlying infrastructure formalised [Mah07], but the project proceeded to instead formalise QE via alternative methods [CM10], [CM12] which are far less efficient[2]. We learn that the CAD approach was not proven correct in the end [CM12, bottom of p. 38]. Thus while it is formalised that Real QE (and thus satisfiability) is decidable, this does not offer a route to verifying current solver results.

The only other related work in the literature we found is [NMD15] which essentially formalises something like CAD but only for problems in one variable.

3 Coverings instead of Decompositions

Most SMT solvers do not directly use the computer algebra implementations of CAD as theory solvers. These are usually adapted for SMT compliance [ÁAB+16], as CAD was in [KÁ20], although there can also be success when used directly [FOS+18]. A particularly exciting recent development has been the interaction between the SMT community and the computer algebra

[1] Formerly, the condition is that projection of any two cells onto a lower dimensional space with respect to the variable ordering are either equal or disjoint. Informally, this means the cells are stacked in cylinders over a decomposition in lower dimensional space.

[2] Although CAD is doubly exponential in the number of variables, the methods verified do not even have worst case complexity bound by a finite tower of exponentials!

community from which many of these methods originate [ÁAB+16]. These has led to entirely new algorithmic approaches.

Perhaps most notable is the NLSAT algorithm of Jovanović and de Moura [JdM12], introduced in 2012 and since generalised into the model constructing satisfiability calculus (mcSAT) framework [dJ13]. In mcSAT the search for a Boolean model and a theory model are mutually guided by each other away from unsatisfiable regions. Partial solution candidates for the Boolean structure and for the corresponding theory constraints are constructed incrementally in parallel. Boolean conflicts are generalised using propositional resolution as normal. At the theory level, when an assignment (sample point) is determined not to satisfy all constraints then this is generalised from the point to a region containing the point on which the same constraints fail for the same reason.

In NLSAT, which only considers `QF_NRA`, the samples are generalised to CAD cells[3] being excluded by adding a new clause with the negation of the algebraic description of the cell. In UNSAT cases these additional clauses become mutually exclusive, in effect the cells generated cover all possible space in $\mathbb{R}^n$. However, as these may not be arranged cylindrically, this may not be trivial to check from the output. We note also the more efficient algorithm to compute these single CAD cells in [BK15], and the new type of decomposition they inspired in [Bro15].

Another new approach was presented recently in [ÁDMK21]: conflict driven cylindrical algebraic covering (CDCAC). Like NLSAT this produces a covering of $\mathbb{R}^n$ to show unsatisfiability. Essentially, a depth first search is performed according to the theory variables. Conflicts over particular assignments are generalised to cells until a covering of a dimension is obtained, and then this covering is generalised to a cell in the dimension below. In this procedure the covering itself is explicit and easy to verify. Further, CDCAD computes the covering relative to a set of constraints to check their consistency independent of the Boolean search, meaning it can be more easily integrated into an CDCL(T)-style SMT solver and combined with its other theory solving modules than NLSAT, which is a solving framework on its own.

Both NLSAT and CDCAC rely on CAD theory to conclude that the generalisations of conflicts from models to cells are valid, and so the verification of such theory is still a barrier to verifiable proofs. But unlike CAD itself, the conflicts that are being generalized are local for both NLSAT and CDCAC which may allow for simpler verification, or verification in certain cases. In particular, it was observed in [ÁDE+20] that a trace of the computation from CDCAC appears far closer to a human derived proof than any of the other algorithms discussed here. Whether this means it will be more susceptible to machine verification remains to be seen.

4 Other approaches for QF_NRA

We wrote earlier that all solvers tackling `QF_NRA` in a complete manner rely on CAD based approaches, as these are the only complete methods that have been implemented. However, there are also a variety of incomplete methods. These include incremental linearisation [CGI+18], interval constraint propagation [TVO17], virtual substitution [Wei97], subtropical satisfiability [FOSV17] and Gröbner bases [HEDP16].

These alternative methods tend to be far more efficient than CAD based ones and so an optimised solver may try these first. Further, they may often be used to solve sub-problems or simplify the input to CAD. To obtain fully verified proofs for `QF_NRA` problems, we may thus need to generate proofs for these methods as well and furthermore integrate them into or combine them with the CAD proofs.

[3] But not necessarily one that would be produced within any entire CAD for the problem.

References

[ÁAB+16] 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. SC^2: Satisfiability checking meets symbolic computation. In M. Kohlhase, M. Johansson, B. Miller, L. de Moura, and F. Tompa, editors, *Intelligent Computer Mathematics: Proceedings CICM 2016*, volume 9791 of *Lecture Notes in Computer Science*, pages 28–43. Springer International Publishing, 2016. URL: https://doi.org/10.1007/978-3-319-42547-4_3.

[ÁDE+20] E. Ábrahám, J.H. Davenport, M. England, G. Kremer, and Z. Tonks. New opportunities for the formal proof of computational real geometry? In P. Fontaine, K. Korovin, I.S. Kotsireas, P. Rümmer, and S. Tourret, editors, *Proceedings of the 5th Workshop on Satisfiability Checking and Symbolic Computation (*SC^2 *2020)*, CEUR-WS 2752, pages 178–188, 2020. URL: http://ceur-ws.org/Vol-2752/.

[ÁDMK21] E. Ábrahám, J.H. Davenport, M.England, and G. Kremer. Deciding the consistency of non-linear real arithmetic constraints with a conflict driven search using cylindrical algebraic coverings. *Journal of Logical and Algebraic Methods in Programming*, 119, 2021. URL: https://doi.org/10.1016/j.jlamp.2020.100633.

[BBFF20] Haniel Barbosa, Jasmin Christian Blanchette, Mathias Fleury, and Pascal Fontaine. Scalable fine-grained proofs for formula processing. *Journal of Automated Reasoning*, 64(3):485–510, 2020. URL: https://doi.org/10.1007/s10817-018-09502-y.

[BdF15] C. Barrett, L. de Moura, and P. Fontaine. Proofs in satisfiability modulo theories. In D. Delahaye and B. Woltzenlogel Paleo, editors, *All about Proofs, Proofs for All*, volume 55 of *Mathematical Logic and Foundations*, pages 23–44. College Publications, London, UK, 2015. URL: http://www.cs.stanford.edu/~barrett/pubs/BdMF15.pdf.

[BFT16] C. Barrett, P. Fontaine, and C. Tinelli. The Satisfiability Modulo Theories Library (SMT-LIB). Online Resource, 2016. URL: http://www.SMT-LIB.org.

[BK15] C.W. Brown and M. Kosta. Constructing a single cell in cylindrical algebraic decomposition. *Journal of Symbolic Computation*, 70:14 – 48, 2015. URL: https://doi.org/10.1016/j.jsc.2014.09.024.

[Bro15] C.W. Brown. Open non-uniform cylindrical algebraic decompositions. In *Proceedings of the 2015 International Symposium on Symbolic and Algebraic Computation*, ISSAC '15, pages 85–92. ACM, 2015. URL: https://doi.org/10.1145/2755996.2756654.

[CGI+18] A. Cimatti, A. Griggio, A. Irfan, M. Roveri, and R. Sebastiani. Incremental linearization for satisfiability and verification modulo nonlinear arithmetic and transcendental functions. *ACM Transactions on Computational Logic*, 19(3):19:1–19:52, 2018. URL: http://doi.acm.org/10.1145/3230639, doi:10.1145/3230639.

[CJ98] B. Caviness and J. Johnson. *Quantifier Elimination and Cylindrical Algebraic Decomposition*. Texts & Monographs in Symbolic Computation. Springer-Verlag, 1998. URL: https://doi.org/10.1007/978-3-7091-9459-1.

[CM10] C. Cohen and A. Mahboubi. A formal quantifier elimination for algebraically closed fields. In S. Autexier, J. Calmet, D. Delahaye, P. Ion, L. Rideau, R. Rioboo, and A.P. Sexton, editors, *Intelligent Computer Mathematics*, volume 6167 of *Lecture Notes in Computer Science*, pages 189–203. Springer Berlin Heidelberg, 2010. URL: https://doi.org/10.1007/978-3-642-14128-7_17.

[CM12] C. Cohen and A. Mahboubi. Formal proofs in real algebraic geometry: from ordered fields to quantifier elimination. *Logical Methods in Computer Science*, 8(1):1–40, 2012. URL: https://doi.org/10.2168/LMCS-8(1:2)2012.

[Col75] G.E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In *Proceedings of the 2nd GI Conference on Automata Theory and Formal Languages*, pages 134–183. Springer-Verlag (reprinted in the collection [CJ98]), 1975. URL:

https://doi.org/10.1007/3-540-07407-4_17.

[dJ13] L. de Moura and D. Jovanović. A model-constructing satisfiability calculus. In R. Giacobazzi, J. Berdine, and I. Mastroeni, editors, *Verification, Model Checking, and Abstract Interpretation (Proc. VMCAI 2013)*, pages 1–12. Springer Berlin Heidelberg, 2013. URL: https://doi.org/10.1007/978-3-642-35873-9_1.

[EBD20] M. England, R. Bradford, and J.H. Davenport. Cylindrical algebraic decomposition with equational constraints. *Journal of Symbolic Computation*, 100:38–71, 2020. URL: https://doi.org/10.1016/j.jsc.2019.07.019.

[FOS+18] P. Fontaine, M. Ogawa, T. Sturm, V. Khanh To, and X. Tung Vu. Wrapping computer algebra is surprisingly successful for non-linear SMT. In A.M. Bigatti and M. Brain, editors, *Proc. 3rd Workshop on Satisfiability Checking and Symbolic Computation (SC^2 2018)*, CEUR-WS 2189, pages 110–117, 2018. URL: http://ceur-ws.org/Vol-2189/.

[FOSV17] P. Fontaine, M. Ogawa, T. Sturm, and X.T. Vu. Subtropical satisfiability. In C. Dixon and M. Finger, editors, *Frontiers of Combining Systems (FroCoS 2017)*, volume 10483 of *Lecture Notes in Computer Science*, pages 189–206. Springer International Publishing, 2017. URL: https://doi.org/10.1007/978-3-319-66167-4_11.

[HEDP16] Z. Huang, M. England, J.H. Davenport, and L. Paulson. Using machine learning to decide when to precondition cylindrical algebraic decomposition with Groebner bases. In *18th Intl. Symp. on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC '16)*, pages 45–52. IEEE, 2016. URL: https://doi.org/10.1109/SYNASC.2016.020.

[HJS18] M.J. Heule, M. Järvisalo, and M. Suda. Proceedings of SAT competition 2018: Solver and benchmark descriptions. In *Technical Report B-2018-1, University of Helsinki Department of Computer Science*, 2018. URL: https://helda.helsinki.fi/handle/10138/237063.

[JdM12] D. Jovanovic and L. de Moura. Solving non-linear arithmetic. In B. Gramlich, D. Miller, and U. Sattler, editors, *Automated Reasoning: 6th International Joint Conference (IJCAR)*, volume 7364 of *Lecture Notes in Computer Science*, pages 339–354. Springer, 2012. URL: https://doi.org/10.1007/978-3-642-31365-3_27.

[KÁ20] G. Kremer and E. Ábrahám. Fully incremental CAD. *Journal of Symbolic Computation*, 100:11–37, 2020. URL: https://doi.org/10.1016/j.jsc.2019.07.018.

[Lam20] P. Lammich. Efficient verified (un)SAT certificate checking. *Journal of Automated Reasoning*, 64:513–532, 2020. URL: https://doi.org/10.1007/s10817-019-09525-z.

[Mah07] A. Mahboubi. Implementing the cylindrical algebraic decomposition within the Coq system. *Mathematical Structures in Computer Science*, 17(1):99–127, 2007. URL: https://doi.org/10.1017/S096012950600586X.

[NMD15] A. Narkawicz, C. Munoz, and A. Dutle. Formally-verified decision procedures for univariate polynomial computation based on Sturm's and Tarski's theorems. *Journal of Automated Reasoning*, 54(4):285–326, 2015. URL: https://doi.org/10.1007/s10817-015-9320-x.

[RB19] A. Rebola-Pardo and A. Biere. Two flavours of DRAT. In *Proceedings of Pragmatics of SAT 2015 and 2018*, volume 59 of *EPiC Series in Computing*, pages 91–110, 2019. URL: https://doi.org/10.29007/lt8r.

[SRT+12] A. Stump, A. Reynolds, C. Tinelli, A. Laugesen, H. Eades, C. Oliver, and R. Zhang. LFSC for SMT proofs: Work in progress. In *Proof Exchange for Theorem Proving—Second International Workshop, PxTP 2012*, CEUR-WS 878, pages 21–27, 2012. URL: http://ceur-ws.org/Vol-878/paper1.pdf.

[TVO17] V.X. Tung, , T. Van Khanh, and M. Ogawa. raSAT: An SMT solver for polynomial constraints. *Formal Methods in System Design*, 51(3):462–499, 2017. URL: https://doi.org/10.1007/s10703-017-0284-9.

[Wei97] V. Weispfenning. Quantifier elimination for real algebra — the quadratic case and beyond. *Applicable Algebra in Engineering, Communication and Computing*, 8(2):85–101, 1997. URL: https://doi.org/10.1007/s002000050055.