Wrapping Computer Algebra is Surprisingly Successful for Non-Linear SMT

Pascal Fontaine, Mizuhito Ogawa, Thomas Sturm, Van Khanh To, Xuan Tung Vu

Wrapping Computer Algebra is Surprisingly Successful for Non-Linear SMT

Pascal Fontaine[1] (orcid.org/0000-0003-4700-6031), Mizuhito Ogawa[2] (orcid.org/0000-0002-8050-7228), Thomas Sturm[1,3] (orcid.org/0000-0002-8088-340X), Van Khanh To[4], Xuan Tung Vu[1,2] (orcid.org/0000-0002-2239-6574)*

[1] University of Lorraine, CNRS, Inria, and LORIA, Nancy, France
{pascal.fontaine,thomas.sturm}@loria.fr
[2] Japan Advanced Institute of Science and Technology, Nomi, Japan
{mizuhito,tungvx}@jaist.ac.jp
[3] MPI Informatics and Saarland University, Saarbrücken, Germany
sturm@mpi-inf.mpg.de
[4] University of Engineering and Technology, VNU, Hanoi, Vietnam
khanhtv@vnu.edu.vn

Abstract. We report on a prototypical tool for Satisfiability Modulo Theory solving for quantifier-free formulas in Non-linear Real Arithmetic or, more precisely, real closed fields, which uses a computer algebra system as the main component. This is complemented with two heuristic techniques, also stemming from computer algebra, viz. interval constraint propagation and subtropical satisfiability. Our key idea is to make optimal use of existing knowledge and work in the symbolic computation community, reusing available methods and implementations to the most possible extent. Experimental results show that our approach is surprisingly efficient in practice.

1 Introduction

Quantifier-free non-linear real arithmetic (QF_NRA) appears in many applications of satisfiability modulo theories (SMT) solving. Efficient reasoning techniques for the corresponding constraints in SMT solvers is thus highly relevant. Examples of SMT solvers with theory reasoners for non-linear real arithmetic are CVC4 [2], SMT-RAT [6], Z3 [11] and Yices [5]. We report here on the prototypical non-linear theory reasoner in veriT. It is based on three techniques, namely interval constraint propagation, subtropical satisfiability, and quantifier elimination. Interval constraint propagation and subtropical satisfiability are heuristic procedures that are easy to implement, but incomplete. Completeness of the theory solver is guaranteed by the use of quantifier elimination. In order to make optimal use of existing knowledge and work in the symbolic computation community, we take advantage of an independent computer algebra system (Redlog/Reduce). The theory reasoner built from these three components is called lazily, on full models produced by the underlying SAT solver. This simple SMT solver is actually quite efficient in solving problems in the QF_NRA category of the SMT-LIB.

* The order of authors is strictly alphabetic.

In Section 2, we briefly present the three procedures that are used in the tool. We then describe how they are combined into one theory reasoner for non-linear real arithmetic literals. We finally report on experiments on the SMT-LIB.

2 Component Procedures

2.1 Interval Constraint Propagation

This section briefly introduces Interval Constraint Propagation (ICP) [1], a technique providing an incomplete but efficient method to check the satisfiability over the reals of a finite set of polynomial constraints [8,17,9,12,25]. Algorithm 1 depicts a naive version of ICP. This algorithm works on boxes in $\mathbb{R}^n$; initially (line 2) there is one box set to $]-\infty,\infty[^n$ for the n variables in the constraints. It iteratively (line 3) picks one box B (line 4) and uses the constraints to heuristically contract that box (line 6), then it evaluates the constraints on B, discarding the box if there is no possible solution (one constraint is unsatisfiable on the box, line 7), or decomposing B into smaller boxes, which will later be considered the same way (line 12). The algorithm terminates concluding unsatisfiability (line 15) when all boxes have been discarded (one constraint is unsatisfiable, for each box), or concluding satisfiability when all constraints are valid in the same box (line 10, indeed every point in the box is a solution for the set of constraints). Note that ICP does not require tedious computations with real algebraic numbers at all. However, the algorithm does not necessarily terminate since it might infinitely decompose boxes into smaller ones.

Algorithm 1 ICP for a set of polynomial constraints φ

```
 1: function ICP(φ)
 2:     S ← {]−∞,∞[^n}
 3:     while S ≠ ∅ do
 4:         choose B ∈ S
 5:         S ← S \ {B}
 6:         B′ ← contract B from constraints
 7:         if B′ = ∅ or one constraint is unsatisfiable on B′ then
 8:             continue
 9:         else if all constraints are valid on B′ then
10:             return SAT
11:         end if
12:         B1, B2 ← decompose B′
13:         S ← S ∪ {B1, B2}
14:     end while
15:     return UNSAT
16: end function
```

Our implementation of ICP uses testing [12,25], to improve satisfiability detection: the constraints are regularly checked for satisfiability on random test points inside given

boxes. Since ICP uses interval arithmetic as an approximation to check validity or unsatisfiability of constraints on a box of full dimension, it is rarely able to find satisfiable assignments in presence of equations. Our ICP algorithm furthermore uses the Intermediate Value Theorem as a heuristic to assert that equations are satisfied within a given box [25].

2.2 Subtropical Satisfiability

The subtropical satisfiability checking method [7] (SUBTROP) is based on the simultaneous evaluation of the input set of polynomial constraints when the values of variables approach 0, 1, or ∞ following some polynomial curve. It basically checks for such limits, if within each polynomial one monomial dominates all other ones. If so, the algorithm computes an interpretation for variables, based on the curve, such that all polynomials in the set have a value of a given sign. The method is mostly limited to determine the satisfiability of a set of constraints containing only inequalities, but it appears that this is useful for the various applications represented in the SMT-LIB.

Example 1. Consider $f_1 = -12 + 2x^{12}y^{25}z^{49} - 31x^{13}y^{22}z^{110} - 11x^{1000}y^{500}z^{89}$ and $f_2 = -23 + 5xy^{22}z^{110} - 21x^{15}y^{20}z^{1000} + 2x^{100}y^2z^{49}$, and the satisfiability checking problem $f_1 > 0 \wedge f_2 > 0$. The method (see [7] for details) shows that the monomials $2x^{12}y^{25}z^{49}$ and $5xy^{22}z^{110}$ respectively dominate in f_1 and f_2 when (x,y,z) tends to $(0,\infty,0)$, more precisely, for values $(x,y,z) = (a^{n_1}, a^{n_2}, a^{n_2})$ with, for instance, $n_1 = -\frac{238834}{120461}$, $n_2 = \frac{2672460}{1325071}$, $n_3 = -\frac{368561}{1325071}$ and a sufficiently large. The method actually computes those n_1, n_2, and n_3. So, if $a \in \mathbb{R}^+$ is large enough, then both constraints $f_1(a^{n_1}, a^{n_2}, a^{n_3}) > 0$ and $f_2(a^{n_1}, a^{n_2}, a^{n_3}) > 0$ are satisfied. Consider $a = 2$ for instance: then $f_1(a^{n_1}, a^{n_2}, a^{n_3}) \approx 16371.99$ and $f_2(a^{n_1}, a^{n_2}, a^{n_3}) \approx 17707.27$. □

The subtropical method expresses as linear constraints some conditions for one monomial to dominate within a polynomial. Subsequently, an SMT solver is used to check the satisfiability of these linear constraints. If the linear constraints are satisfiable, then the non-linear constraint is satisfiable and the model of the linear constraints provides a witness for the polynomial curve, i.e., for the previous example, the values of n_1, n_2, and n_3. For sets of polynomial constraints, the process is essentially the same.

Experimentally, the application of the subtropical satisfiability method to solve non-linear constraints is fast, either to succeed or to fail. On our experiments, it furthermore provides an answer on several problems that defeat all state-of-the-art solvers. The method thus establishes a useful complementing heuristic with little drawback, to be used either upfront or in portfolio with other approaches to deal with non-linear constraints.

2.3 Quantifier Elimination

As a complete method, we use Quantifier Elimination techniques (QE). While it is possible to quickly implement ICP and subtropical satisfiability checking, those quantifier elimination techniques are significantly more complex and require expertise. It is not feasible to implement a custom version for SMT with a few person months. We thus

used a computer algebra system, namely Reduce, and more precisely, the package Redlog [4] which implements interpreted first-order logic on top of computer algebra. Besides many other domains, Redlog provides decision procedures for real closed fields. For real arithmetic Redlog combines two quantifier elimination techniques, virtual substitution [26,27,13] with partial cylindrical algebraic decomposition [3,19]. For a survey of real applications of Redlog see [22]. Further theories supported by Redlog include discretely valued fields [21], term algebras [23], QBF [20,24], Presburger Arithmetic [15] and fragments of non-linear integer arithmetic [14].

To embed a decision procedure in SMT, one property is mandatory: the procedure should feature small conflict production. From an unsatisfiable set of literals, it should produce an unsatisfiable subset containing few unnecessary literals, optimally a minimal unsatisfiable subset, i.e. such that all proper subsets are satisfiable. With small conflict production, the SMT infrastructure goes considerably beyond a SAT solver enumerating all possible models of the Boolean abstraction of the input formula, the theory reasoner refuting them one at a time. We recently presented [10] a method using linear optimization to compute conflict sets for cylindrical algebraic decomposition and virtual substitution. Virtual substitution and cylindrical algebraic decomposition share the same basic idea of finding a finite set of test points that suffice to determine the unsatisfiability of a set of constraints S. If the set S is unsatisfiable, each of these test points falsifies at least one constraint in S. Finding a conflict set reduces to finding a subset of the constraints that contains, for each test point, at least one unsatisfied constraint. We implemented this in the Quantifier Elimination algorithms in Redlog. Redlog/Reduce now furthermore provides an interface for SMT solver, so that the software can be used as a theory reasoner with little effort. This reasoner is used in our portfolio of tools, and guarantees the completeness of the combination of reasoners on real closed fields.

3 Combining Procedures

We have previously discussed subtropical satisfiability (SUBTROP), interval constraint propagation (ICP), and quantifier elimination (QE). Our aim is to combine these in a complete and efficient framework for solving polynomial constraints. Recall that SUBTROP is only an incomplete heuristics, and ICP does not even guarantee termination; ICP might loop forever, for instance in the case of touching spheres. Combining ICP sequentially with other procedures thus requires heuristics for ICP termination to ensure fairness and to let the other algorithms also work on the constraints. Before decomposing a box into smaller ones (line 12 in Algorithm 1), our algorithm will check if bounded boxes have been generated, all the bounded boxes are smaller than a chosen value ε in all dimensions. If so, the algorithm gives up, and returns UNKNOWN along with the box B resulting from the contraction of $]-\infty,\infty[^n$ from constraints; the algorithm returns the empty box in case of unsatisfiability and the valid box in case of satisfiability. Intuitively, B is an over-approximation of all boxes possibly containing solutions of constraints. We furthermore require that boxes are handled in a chronological order, that is, S in Algorithm 1 becomes a queue, where the first box (the oldest one) is chosen and removed, and newer boxes are added at the end. Boxes are decomposed only along axes with lengths greater than ε. The procedure ensures unbounded boxes

are not decomposed infinitely. The algorithm terminates either when it detects satisfiability (or unsatisfiability) or all bounded boxes have all dimensions smaller than ε. We refer to this terminating version of ICP as ICPT.

A lazy combining approach considers the procedures as black boxes and invokes them sequentially to check the satisfiability of the constraints. The fastest procedures to terminate are ordered first, and the complete procedure is called last. In Algorithm 2, SUBTROP(φ) returns SAT if and only if subtropical satisfiability succeeds in finding a model for φ; remember that subtropical satisfiability is indeed essentially a model finding method. The complete decision procedure (in our case, quantifier elimination methods as implemented in Redlog/Reduce) is called in line 9, and returns SAT (resp. UNSAT) if the input is satisfiable (resp. unsatisfiable). Notice that, when calling the complete decision procedure, the set of constraints φ is complemented with a box B found by ICPT. Actually (this is not shown in Algorithm 2), φ is furthermore cleaned of the constraints that are valid in the box B.

Algorithm 2 Lazy combination of procedures

```
 1: function LAZY(φ)
 2:     if SUBTROP(φ) = SAT then
 3:         return SAT
 4:     end if
 5:     (result, B) ← ICPT(φ)
 6:     if result ≠ UNKNOWN then
 7:         return result
 8:     end if
 9:     return QE (φ ∧ B)
10: end function
```

We experimentally evaluated another combination interleaving more tightly the complete procedure and ICP. The combination is similar to ideas in [16]: when a box of ICP becomes smaller than a threshold ε, quantifier elimination is called, for the sake of completeness, to solve the remaining unknown constraints over this small box. We found however that this performs less well than the lazy one above. We are investigating further techniques to fix this.

4 Experiments

All experiments were conducted on an Intel Xeon E5-2680v2 at 2.80GHz running GNU/Linux CentOS 6.4 with kernel 2.6.32. Each solver ran on the 11354 benchmarks of the QF_NRA category (quantifier-free Non-linear Real Arithmetic) of the SMT-LIB (4963 are labeled as satisfiable, 5296 unsatisfiable, 1095 unknown) with a memory limit of 8 GB memory and a time out of 2500 seconds for each benchmark.[5]

[5] See http://www.jaist.ac.jp/~s1520002/veriT+STROPSAT+raSAT+Redlog/ for full results and the tool with the source code. Note to reviewers: this will soon be on Zenodo.

Table 1. Numbers of benchmarks solved by component procedures

Benchmarks	SUBTROP	ICP	QE	**All**	w/o SUBTROP	w/o ICP	w/o QE
SAT	1936	4302	4400	**4450**	4433	4436	4313
UNSAT	2530	4472	4959	**5012**	5012	4959	4472
Total (11354)	4466	8774	9359	**9462**	9445	9395	8785
Total time (s)	4744	18835	67945	50632	44815	67420	22357

In our implementation, raSAT [12,25] serves as the ICP-based solver, the computer algebra system Redlog/Reduce [4] for quantifier elimination, and STROPSAT [7] as the implementation of subtropical satisfiability. The interface for the three tools within veriT is 900 lines of C code. Notice that CVC4 is used inside STROPSAT to solve linear constraints (see again [7]). The framework for solving polynomial constraints in the SMT solver is called lazily, when the underlying SAT solver has produced a full model and if the corresponding set of literals is not shown unsatisfiable by the linear arithmetic decision procedure in veriT.

Table 1 compares the performance of the distinct procedures and their combination. Notice that QE alone already solves many problems, but combining the procedures brings improvements both in running time and in the number of solved problems. SUBTROP increases the number of solved satisfiable problems and improves times for several satisfiable problems, without significantly impacting negatively for the problems solved by the other methods: indeed, SUBTROP actually takes around 2000 additional seconds for six difficult problems also solved by the combination of ICP and QE, and uses 4000 seconds to solve 17 more problems. ICP has a general positive impact both on running times and on the number of problems solved.

Table 2. Performance of state-of-the-art SMT solvers on QF_NRA

Benchmarks	**CVC4**	**SMT-RAT**	**Z3**	**Yices**	**veriT**	veriT only	Virtual best
SAT	2929	4398	**4905**	4845	4450	18	5183
UNSAT	**5324**	4425	5038	5120	5012	1	5744
Total (11354)	8253	8823	9943	**9965**	9462	19	10927
Total time (s)	146154	57787	37740	132137	50632	11706	119998

Table 2 compares state-of-the-art SMT solvers with support for non-linear arithmetic. SMT-RAT implements a less lazy combination (as mentioned in the previous section) between interval constraint propagation and cylindrical algebraic decomposition [16], Yices and Z3 implement the nlsat procedure [11]. CVC4 uses context-dependent simplification [18] and incremental linearization [2]. Our results validate the main point of the paper: a combination of simple heuristics (ICP and subtropical satisfiability) with quantifier elimination as implemented in a computer algebra system (Redlog/Reduce) slightly tuned to fit the SMT infrastructure is an efficient decision procedure to solve non-linear arithmetic SMT problems. Table 2 also clearly exhibits that

the virtual best solver — a portfolio of all mentioned solvers running in parallel — is much better than each individual solver; we attribute this to the variety of techniques used in the solvers.

5 Conclusion

Implementing state-of-the-art decision procedures for the theory of real non-linear arithmetic requires a lot of expertise and is very time consuming. Hopefully, reusing a computer algebra system actually provides good results on the SMT-LIB. We also notice that, thanks to the diversity of techniques used to tackle this theory, the various SMT solvers have complementary strengths and the virtual best solver is much better than each solver alone.

We expect the non-linear arithmetic reasoner in veriT to improve, following the improvements in the embedded computer algebra system itself. For instance, Redlog/Reduce features a new experimental virtual substitution algorithm which shows very promising results on problems stemming from other computer algebra applications; veriT will eventually benefit from this new algorithm once it becomes stable.

In future works, we plan to investigate a less lazy combination of ICP and quantifier elimination. In our current prototype, the quantifier elimination is applied for each box that ICP is unable to discard, one at a time. The idea would be, with a single call to quantifier elimination, to eliminate several boxes in ICP.

Acknowledgments

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No H2020-FETOPEN-2015-CSA 712689 (SC^2), and from the European Research Council under the European Union's Horizon 2020 research and innovation program (grant agreement No. 713999, Matryoshka). Xuan Tung Vu would like to acknowledge the JAIST Off-Campus Research Grant for fully supporting him during his stay at LORIA, Nancy. The work has also been partially supported by the JSPS KAKENHI Grant-in-Aid for Scientific Research(B) (15H02684) and the JSPS Core-to-Core Program (A. Advanced Research Networks).

References

1. Benhamou, F., Granvilliers, L.: Continuous and interval constraints. In: F. Rossi, P.v.B., Walsh, T. (eds.) Handbook of Constraint Programming, pp. 571–604. Elsevier, New York (2006)
2. Cimatti, A., Griggio, A., Irfan, A., Roveri, M., Sebastiani, R.: Satisfiability modulo transcendental functions via incremental linearization. In: de Moura, L. (ed.) CADE 2017. LNCS, vol. 10395, pp. 95–113. Springer (2017)
3. Collins, G.E., Hong, H.: Partial cylindrical algebraic decomposition for quantifier elimination. J. Symb. Comput. 12(3), 299–328 (Sep 1991)
4. Dolzmann, A., Sturm, T.: Redlog: Computer algebra meets computer logic. SIGSAM Bull. 31(2), 2–9 (Jun 1997)

5. Dutertre, B.: Yices 2.2. In: Biere, A., Bloem, R. (eds.) CAV 2014. pp. 737–744. Springer (2014)
6. Florian, C., Ulrich, L., Sebastian, J., Erika, Á.: SMT-RAT: An SMT-Compliant nonlinear real arithmetic toolbox. In: Alessandro, C., Roberto, S. (eds.) SAT 2012, pp. 442–448. Springer (2012)
7. Fontaine, P., Ogawa, M., Sturm, T., Vu, X.T.: Subtropical satisfiability. In: Dixon, C., Finger, M. (eds.) FroCoS 2017, pp. 189–206. Springer (2017)
8. Fränzle, M., Herde, C., Teige, T., Ratschan, S., Schubert, T.: Efficient solving of large non-linear arithmetic constraint systems with complex boolean structure. JSAT 1, 209–236 (2007)
9. Gao, S., Kong, S., Clarke, E.M.: Satisfiability modulo ODEs. In: FMCAD 2013. pp. 105–112 (2013)
10. Jaroschek, M., Dobal, P.F., Fontaine, P.: Adapting real quantifier elimination methods for conflict set computation. In: Lutz, C., Ranise, S. (eds.) FroCoS 2015, pp. 151–166. Springer (2015)
11. Jovanović, D., de Moura, L.: Solving non-linear arithmetic. In: Gramlich, B., Miller, D., Sattler, U. (eds.) Automated Reasoning, pp. 339–354. Springer (2012)
12. Khanh, T.V., Ogawa, M.: SMT for polynomial constraints on real numbers. ENTCS 289, 27 – 40 (2012), TAPAS' 2012
13. Košta, M.: New Concepts for Real Quantifier Elimination by Virtual Substitution. Doctoral dissertation, Saarland University, Germany (December 2016)
14. Lasaruk, A., Sturm, T.: Weak integer quantifier elimination beyond the linear case. In: CASC 2007, LNCS, vol. 4770. Springer (2007)
15. Lasaruk, A., Sturm, T.: Weak quantifier elimination for the full linear theory of the integers. A uniform generalization of Presburger arithmetic. Appl. Algebra Eng. Commun. Comput. 18(6), 545–574 (Dec 2007)
16. Loup, U., Scheibler, K., Corzilius, F., Ábrahám, E., Becker, B.: A symbiosis of interval constraint propagation and cylindrical algebraic decomposition. In: Bonacina, M.P. (ed.) CADE 24, pp. 193–207. Springer (2013)
17. Ratschan, S.: Efficient solving of quantified inequality constraints over the real numbers. ACM Trans. Comput. Log. 7, 723–748 (Oct 2006)
18. Reynolds, A., Tinelli, C., Jovanovic, D., Barrett, C.: Designing theory solvers with extensions. In: Dixon, C., Finger, M. (eds.) FroCoS 2017. LNCS, vol. 10483, pp. 22–40. Springer (2017)
19. Seidl, A.: Cylindrical Decomposition Under Application-Oriented Paradigms. Ph.D. thesis, Universität Passau, 94030 Passau, Germany (Mar 2006)
20. Seidl, A.M., Sturm, T.: Boolean quantification in a first-order context. In: Ganzha, V.G., Mayr, E.W., Vorozhtsov, E.V. (eds.) CASC 2003, pp. 329–345. Institut für Informatik, Technische Universität München, München, Germany (2003)
21. Sturm, T.: Linear problems in valued fields. J. Symb. Comput. 30(2), 207–219 (Aug 2000)
22. Sturm, T.: A survey of some methods for real quantifier elimination, decision, and satisfiability and their applications. Math. Comput. Sci. 11(3–4), 483–502 (December 2017)
23. Sturm, T., Weispfenning, V.: Quantifier elimination in term algebras. The case of finite languages. pp. 285–300
24. Sturm, T., Zengler, C.: Parametric quantified SAT solving. In: Watt, S.M. (ed.) ISSAC 2010, pp. 77–84. ACM Press, New York (2010)
25. Tung, V.X., Khanh, T.V., Ogawa, M.: rasat: an SMT solver for polynomial constraints. Formal Methods in System Design 51(3), 462–499 (2017)
26. Weispfenning, V.: The complexity of linear problems in fields. J. Symb. Comput. 5(1–2), 3–27 (Feb–Apr 1988)
27. Weispfenning, V.: Quantifier elimination for real algebra—the quadratic case and beyond. Appl. Algebra Eng. Commun. Comput. 8(2), 85–101 (Feb 1997)