Cooperating Techniques for Solving Nonlinear Real Arithmetic in the cvc5 SMT Solver (System Description)

Gereon Kremer[1][0000−0002−0393−5739], Andrew Reynolds[2][0000−0002−3529−8682], Clark Barrett[1][0000−0002−9522−3084], and Cesare Tinelli[2][0000−0002−6726−775X]

[1] Stanford University, Stanford, USA
[2] The University of Iowa

Abstract. The cvc5 SMT solver solves quantifier-free nonlinear real arithmetic problems by combining the cylindrical algebraic coverings method with incremental linearization in an abstraction-refinement loop. The result is a complete algebraic decision procedure that leverages efficient heuristics for refining candidate models. Furthermore, it can be used with quantifiers, integer variables, and in combination with other theories. We describe the overall framework, individual solving techniques, and a number of implementation details. We demonstrate its effectiveness with an evaluation on the SMT-LIB benchmarks.

Keywords: satisfiability modulo theories · nonlinear real arithmetic · abstraction refinement · cylindrical algebraic coverings.

1 Introduction

SMT solvers are used as back-end engines for a wide variety of academic and industrial applications [2,19,20]. Efficient reasoning in the theory of real arithmetic is crucial for many such applications [5,9]. While modern SMT solvers have been shown to be quite effective at reasoning about *linear* real arithmetic problems [21,41], *nonlinear* problems are typically much more difficult. This is not surprising, given that the worst-case complexity for deciding the satisfiability of nonlinear real arithmetic formulas is doubly-exponential in the number of variables in the formula [15]. Nevertheless, a variety of techniques have been proposed and implemented, each attempting to target a class of formulas for which reasonable performance can be observed in practice.

Related work. All complete decision procedures for nonlinear real arithmetic (or the *theory of the reals*) originate in computer algebra, the most prominent being cylindrical algebraic decomposition (CAD) [11]. While alternatives exist [6,25,39], they have not seen much use [27], and CAD-based methods are the only sound and complete methods in practical use today. CAD-based methods used in modern SMT solvers include incremental CAD implementations [33,35]

and cylindrical algebraic coverings [3], both of which are integrated in the traditional CDCL(T) framework for SMT [**?**].

In contrast, the NLSAT [29] calculus and the generalized MCSAT [**?**,38] framework provide for a much tighter integration of a conflict-driven CAD-based theory solver into a theory-aware core solver. This has been the dominant approach over the last decade due to its strong performance in practice. However, it has the significant disadvantage of being difficult to integrate with CDCL(T)-based frameworks for theory combination.

A number of *incomplete* techniques are also used by various SMT solvers: incremental linearization [10] gradually refines an abstraction of the nonlinear formula obtained via a naive linearization by refuting spurious models of the abstraction; interval constraint propagation [24,35,43] employs interval arithmetic to narrow down the search space; subtropical satisfiability [22] provides sufficient linear conditions for nonlinear solutions in the exponent space of the polynomials; and virtual substitution [12,30,44] makes use of parametric solution formulas for polynomials of bounded degree. Though all of these techniques have limitations, each of them is useful for certain subclasses of nonlinear real arithmetic or in combination with other techniques.

Contributions. We present an integration of cylindrical algebraic coverings and incremental linearization, implemented in the `cvc5` SMT solver. Crucial to the success of the integration is an abstraction-refinement loop used to combine the two techniques cooperatively. The solution is effective in practice, as witnessed by the fact that `cvc5` won the nonlinear real arithmetic category of SMT-COMP 2021 [42], the first time a non-MCSAT-based technique has won since 2013. Our integrated technique also has the advantage of being very flexible: in particular, it fits into the regular CDCL(T) schema for theory solvers and theory combination, it supports (mixed) integer problems, and it can be easily extended using further subsolvers that support additional arithmetic operators beyond the scope of traditional algebraic routines (e.g., transcendental functions).

2 Nonlinear solving techniques

The nonlinear arithmetic solver implemented in `cvc5` generally follows the abstraction-refinement framework introduced by Cimatti et al. [10] and depicted in Figure 1. The input assertions are first checked by the linear arithmetic solver, where they are linearized implicitly by treating every application of multiplication as if it were an arithmetic variable. For example, given input assertions $x \cdot y > 0 \wedge x > 1 \wedge y < 0$, the linear solver treats the expression $x \cdot y$ as a variable. It may then find the (spurious) model: $x \mapsto 2$, $y \mapsto -1$, and $x \cdot y \mapsto 1$. We call the candidate model returned by the linear arithmetic solver, where applications of multiplication are treated as variables, a *linear model*. If a linear model does not exist, i.e., the input is unsatisfiable according to the linear solver, the linear solver generates a conflict that is immediately returned to the CDCL(T) engine.

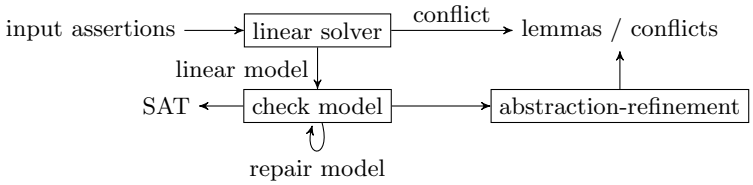

Fig. 1: Structural overview of the nonlinear solver

When a linear model does exist, we check whether it already satisfies the input assertions or try to *repair* it to do so. We only apply a few very simple heuristics for repairs such as updating the value for z in the presence of a constraint like $z = x \cdot y$ based on the values of x and y.

If the model can not be repaired, we refine the abstraction for the linear solver [10]. This step constructs lemmas, or conflicts, based on the input assertions and the linear model, to advance the solving process by blocking either the current linear model or the current Boolean model, that is, the propositional assignment generated by the SMT solver's SAT engine. The Boolean model is usually eliminated only by the coverings approach, while the incremental linearization technique generates lemmas with new literals that target the linear model, e.g., the lemma $x > 0 \wedge y < 0 \Rightarrow x \cdot y < 0$ in the example above. We next describe our implementation of cylindrical algebraic coverings and incremental linearization, and how they are combined in `cvc5`.

2.1 Cylindrical algebraic coverings

Cylindrical algebraic coverings is a technique recently proposed by Ábrahám et al. [3] and is heavily inspired by CAD. While the way the computation proceeds is very different from traditional CAD, and instead somewhat similar to NLSAT [29], their mathematical underpinnings are essentially identical. The cylindrical algebraic coverings subsolver in `cvc5` closely follows the presentation in [3]. Below, we discuss some differences and extensions. For this discussion, we must refer the reader to [3] for the relevant background material because of space constraints. We note that `cvc5` relies on the libpoly library [28] to provide most of the computational infrastructure for algebraic reasoning.

Square-free basis. As with most CAD projection schemas, the set of projection polynomials needs to be a square-free basis when computing the characterization for an interval in [3, Algorithm 4]. However, the resultants computed in this algorithm combine polynomials from different sets, which are not necessarily coprime. The remedy is to either make these sets of polynomials pairwise square-free or to fully factor all projection polynomials. We adopt the former approach.

Starting model. Although the linear model may not satisfy the nonlinear constraints, we may expect it to be in the vicinity of a proper model. We thus

optionally use the linear model as an *initial assignment* for the cylindrical algebraic coverings algorithm in one of two ways: either using it initially in the search and discarding it as soon as it conflicts; or using it whenever possible, even if it leads to a conflict in another branch of the search. Unfortunately, neither technique has any discernible impact in our experiments.

Interval pruning. As already noted in [3], a covering may contain two kinds of redundant intervals: intervals fully contained in another interval, or intervals contained in the union of other intervals. Removing the former kind of redundancies is not only clearly beneficial, but also required for how the characterizations are computed. It is not clear, however, if it is worthwhile to remove redundancies of the second kind because, while it can simplify the characterization locally, it may also make the resulting interval smaller, slowing down the overall solving process. Note that there may not be a unique redundant interval: e.g., if multiple intervals overlap, it may be possible to remove one of two intervals, but not both of them. We have implemented a simple heuristic to detect redundancies of the second kind, always removing the smallest interval with respect to the interval ordering given in [3]. Even if these redundancies occur in about 7.5% of all `QF_NRA` benchmarks, using this technique has only a very limited impact. It may be that for certain kinds of benchmarks, underrepresented in SMT-LIB, the technique is valuable. Or it may be that some variation of the technique is more broadly helpful. These are interesting directions for future work.

Lifting and coefficient selection with Lazard. The original cylindrical algebraic coverings technique is based on McCallum's projection operator [36], which is particularly well-studied, but also (refutationally) unsound: polynomial nullification may occur when computing the real roots, possibly leading to the loss of real roots and thus solution candidates. One then needs to check for these cases and fall back to a more conservative, albeit more costly, projection schema such as those due to Collins [11] or Hong [26].

Lazard's projection schema [34], which has been proven correct only recently [37], provides very small projection sets and is both sound and complete. This comes at the price of a different mathematical background and a modified lifting procedure, which corresponds to a modified procedure for real root isolation. Although the local projections employed in cylindrical algebraic coverings have not been formally verified for Lazard's projection schema yet, we expect no significant issues there. Adopting it seems to be a logical improvement, as already mentioned in [3]. The modified real root isolation procedure is a significant hurdle in practice, as it requires additional nontrivial algorithms [31, Section 5.3.2]. We implemented it using CoCoALib [1] in `cvc5` [32], achieving soundness without any discernible negative performance impact.

Using Lazard's projection schema, for all its benefits, may seem questionable for the following reasons: (*i*) the unsoundness of McCallum's projection operator is virtually never witnessed in practice [31, Section 6.5][32], and (*ii*) the projection sets computed by Lazard's and McCallums's projection operator are identical on more than 99.5% on all of `QF_NRA` [32]. We argue, though, that

working in the domain of formal verification warrants the effort of obtaining a (provably) correct result, especially if it does not incur a performance overhead.

Proof generation. Recently, generating formal proofs to certify the result of SMT solvers has become an area of focus. In particular, there is a large and ongoing effort to produce proofs in cvc5. The incremental linearization approach can be seen as an oracle which produces lemmas that are easy to prove individually, so cvc5 does generate proofs for them; the complex part is finding those lemmas and making sure they actually help the solver make progress.

The situation is very different for cylindrical algebraic coverings: the produced lemma is the infeasible subset, and we usually have no simpler proof than the computations relying on CAD theory. That said, cylindrical algebraic coverings appear to be more amenable to automatic proof generation than traditional CAD-based approaches [4,14]. In fact, although making these proofs detailed enough for automated verification is still an open problem, they are already broken into smaller parts that closely follow the tree-shaped computation of the algorithm. This allows cvc5 to produce at least a proof skeleton in that case.

2.2 Incremental Linearization

Our theory solver for nonlinear (real) arithmetic optionally uses lemma schemas following the incremental linearization approaches described by Cimatti et al. [10] and Reynolds et al. [40]. These schemas incrementally refine candidate models from the linear arithmetic solver by introducing selected quantifier-free lemmas that express properties of multiplication, such as signedness (e.g., $x > 0 \wedge y > 0 \Rightarrow x \cdot y > 0$) or monotonicity (e.g., $|x| > |y| \Rightarrow x \cdot x > y \cdot y$). They are generated as needed to refute spurious models that violate these properties.

Most lemma schemas built-in in cvc5 are crafted so as to avoid introducing new monomial terms or coefficients, since that could lead to non-termination in the CDCL(T) search. As a notable exception, we rely on a lemma schema for *tangent planes* for multiplication [10], which can be used to refute the candidate model for any application of the multiplication operator $\cdot$ whose value in the linear model is inconsistent with the standard interpretation of $\cdot$. Note that since these lemmas depend upon the current model value chosen for arithmetic variables, tangent plane lemmas may introduce an unbounded number of new literals into the search. The set of lemma schemas used by the solver is user-configurable, as described in the following section.

2.3 Strategy

The overall theory solver for nonlinear arithmetic is built from several subsolvers, implementing the techniques described above, using a rather naive strategy, as summarized in Algorithm 1. After a spurious linear model has been constructed that cannot be repaired, we first apply a subset of the lemma schemas that do not introduce an unbounded number of new terms (with procedure `IncLinearizationLight`); then, we continue with the remaining lemma schemas

```
1 Function NlSolve(assertions)
2     if not LinearSolve(assertions) then return linear conflict
3     M = linear model for assertions
4     if RepairModel(assertions, M) then return repaired model
5     if IncLinearizationLight(assertions, M) then return lemmas
6     if IncLinearizationFull(assertions, M) then return lemmas
7     return Coverings(assertions, M)
```

Algorithm 1: Strategy for nonlinear arithmetic solver

(with procedure `IncLinearizationFull`); finally, we resort to the coverings solver which is guaranteed to find either a conflict or a model. Internally, each procedure sequentially tries its assigned lemma schemas from [10,40] until it constructs a lemma that can block the spurious model.

The approach is dynamically configured based on input options and the logic of the input formula. For example, by default, we disable `IncLinearizationFull` for `QF_NRA` as it tends to diverge in cases where the coverings solver quickly terminates.

2.4 Beyond `QF_NRA`

The presented solver primarily targets quantifier-free nonlinear real arithmetic, but is used also in the presence of quantifiers and with multiple theories.

Quantified logics. Solving quantified logics for nonlinear arithmetic requires solving quantifier-free subproblems, and thus any improvement to quantifier-free solving also benefits solving with quantifiers. In practice, however, the instantiation heuristics are just as important for overall solver performance.

Multiple theories. The theory combination framework as implemented in `cvc5` requires evaluating equalities over the combined model. To support this functionality, real algebraic numbers had to be properly integrated into the entire solver; in particular, the ability to compute with these numbers could not be local to the cylindrical algebraic coverings module or even the nonlinear solver.

3 Experimental results

We evaluate our implementation within `cvc5` (commit id `449dd7e`) in comparison with other SMT solvers on all 11552 benchmarks in the quantifier-free nonlinear real arithmetic (`QF_NRA`) logic of SMT-LIB. We consider three configurations of `cvc5`, each of which runs a subset of steps from Algorithm 1. All the configurations run lines 2-4. In addition, `cvc5.cov` runs line 7, `cvc5.inclin` runs lines 5 and 6, and `cvc5` runs lines 5 and 7. All experiments were conducted on Intel Xeon E5-2637v4 CPUs with a time limit of 20 minutes and 8GB memory.

We compare `cvc5` with recent versions of all other SMT solvers that participated in the `QF_NRA` logic of SMT-COMP 2021 [42]: `MathSAT` 5.6.6 [**?**], `SMT-RAT`

QF_NRA	sat	unsat	solved
cvc5	**5137**	**5596**	**10733**
Yices2	4966	5450	10416
z3	5136	5207	10343
cvc5.cov	5001	5077	10078
SMT-RAT	4828	5038	9866
veriT	4522	5034	9556
MathSAT	3645	5357	9002
cvc5.inclin	3421	5376	8797

(a) Experiments for QF_NRA

Beyond QF_NRA		sat	unsat	solved
NRA	Yices2	231	**3817**	**4048**
	z3	**236**	3812	**4048**
	cvc5.cov	**236**	3809	4045
	cvc5	221	3809	4030
	cvc5.inclin	120	3786	3906
QF_UFNRA	z3	**24**	**11**	**35**
	Yices2	23	**11**	34
	cvc5	20	**11**	31
	cvc5.inclin	12	**11**	23
	cvc5.cov	2	**11**	13

(b) Experiments for NRA and QF_UFNRA

19.10.560 [13], veriT [7] (veriT+raSAT+Redlog), Yices2 2.6.4 [18] (Yices-QS for quantified logics), and z3 4.8.14 [16]. MathSAT employs an abstraction-refinement mechanism very similar to the one described in Section 2.2; veriT [23] forwards nonlinear arithmetic problems to the external tools raSAT [43], which uses interval constraint propagation, and Redlog/Reduce [17], which focuses on virtual substitution and cylindrical algebraic decomposition; SMT-RAT, Yices2, and z3 all implement some variant of MCSAT [29]. Note that SMT-RAT also implements the cylindrical algebraic coverings approach, but it is less effective than SMT-RAT's adaptation of MCSAT [3].

Figure 2a shows that cvc5 significantly outperforms all other QF_NRA solvers. Both the coverings approach (cvc5.cov) and the incremental linearization approach (cvc5.inclin) contribute substantially to the overall performance of the unified solver in cvc5, with coverings solving many satisfiable instances, and incremental linearization helping on unsatisfiable ones. Even though cvc5.inclin closely follows [10], it outperforms MathSAT on unsatisfiable benchmarks, those where cvc5 relies on incremental linearization the most.

Comparing cvc5 and Yices2 is particularly interesting, as the coverings approach in cvc5 and the NLSAT solver in Yices2 both rely on libpoly [28], thus using the same implementation of algebraic numbers and operations over them. Our integration of incremental linearization and algebraic coverings is compatible with the traditional CDCL(T) framework and outperforms the alternative NLSAT approach, which is specially tailored to nonlinear real arithmetic.

Going beyond QF_NRA, we also evaluate the performance of our solver in the context of theory combination (with all 37 benchmarks from QF_UFNRA) and quantifiers (with all 4058 benchmarks from NRA). There, cvc5 is a close runner-up to Yices2 and z3, thanks to the coverings subsolver which significantly improves cvc5's performance. We conjecture that the remaining gap is due to components other than the nonlinear arithmetic solver, such as the solver for equality and uninterpreted functions, details of theory combination, or quantifier instantiation heuristics. Interestingly, the sets of unsolved instances in NRA are almost disjoint

for `cvc5.cov`, `Yices2` and `z3`, indicating that each tool could solve the remaining benchmarks with reasonable extra effort.

4 Conclusion

We have presented an approach for solving quantifer-free nonlinear real arithmetic problems that combines previous approaches based on incremental linearization [10] and cylindrical algebraic coverings [3] into one coherent abstraction-refinement loop. The resulting implementation is very effective, outperforming other state-of-the-art solver implementations, and integrates seamlessly in the CDCL(T) framework.

The general approach also applies to integer problems, quantified formulas, and instances with multiple theories, and can additionally be used in combination with transcendental functions [10] and bitwise conjunction for integers [45]. Further evaluations of these combinations are left to future work.

References

1. Abbott, J., Bigatti, A.M., Palezzato, E.: New in CoCoA-5.2.4 and CoCoALib-0.99600 for sc-square. In: Satisfiability Checking and Symbolic Computation. CEUR Workshop Proceedings, vol. 2189, pp. 88–94. CEUR-WS.org (2018), http://ceur-ws.org/Vol-2189/paper4.pdf
2. Ábrahám, E., Corzilius, F., Johnsen, E.B., Kremer, G., Mauro, J.: Zephyrus2: on the fly deployment optimization using SMT and CP technologies. In: International Symposium on Dependable Software Engineering: Theories, Tools, and Applications. pp. 229–245. Springer (2016). https://doi.org/10.1007/978-3-319-47677-3_15
3. Ábrahám, E., Davenport, J.H., England, M., Kremer, G.: 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**(100633) (2021). https://doi.org/10.1016/j.jlamp.2020.100633
4. Abrahám, E., Davenport, J.H., England, M., Kremer, G.: Proving unsat in smt: The case of quantifier free non-linear real arithmetic. arXiv preprint arXiv:2108.05320 (2021)
5. Arnett, T.J., Cook, B., Clark, M., Rattan, K.: Fuzzy logic controller stability analysis using a satisfiability modulo theories approach. In: 19th AIAA Non-Deterministic Approaches Conference. p. 1773 (2017)
6. Basu, S., Pollack, R., Roy, M.F.: On the combinatorial and algebraic complexity of quantifier elimination. Journal of the ACM **43**, 1002–1045 (1996). https://doi.org/10.1145/235809.235813
7. Bouton, T., de Oliveira, D.C.B., Déharbe, D., Fontaine, P.: veriT: an open, trustable and efficient smt-solver. In: International Conference on Automated Deduction. pp. 151–156. Springer (2009). https://doi.org/10.1007/978-3-642-02959-2_12
8. Cashmore, M., Magazzeni, D., Zehtabi, P.: Planning for hybrid systems via satisfiability modulo theories. Journal of Artificial Intelligence Research **67**, 235–283 (2020). https://doi.org/10.1613/jair.1.11751
9. Cimatti, A., Griggio, A., Irfan, A., Roveri, M., Sebastiani, R.: Incremental linearization for satisfiability and verification modulo nonlinear arithmetic and transcendental functions. ACM Transactions on Computational Logic **19**, 19:1–19:52 (2018). https://doi.org/10.1145/3230639
10. Collins, G.E.: Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In: Barkhage, H. (ed.) Automata Theory and Formal Languages. LNCS, vol. 33, pp. 134–183. Springer (1975). https://doi.org/10.1007/3-540-07407-4_17
11. Corzilius, F., Ábrahám, E.: Virtual substitution for smt-solving. In: International Symposium on Fundamentals of Computation Theory. pp. 360–371 (2011). https://doi.org/10.1007/978-3-642-22953-4_31
12. Corzilius, F., Kremer, G., Junges, S., Schupp, S., Ábrahám, E.: SMT-RAT: an open source c++ toolbox for strategic and parallel smt solving. In: Theory and Applications of Satisfiability Testing. LNCS, vol. 9340, pp. 360–368. Springer (2015). https://doi.org/10.1007/978-3-319-24318-4_26
13. Davenport, J., England, M., Kremer, G., Tonks, Z., et al.: New opportunities for the formal proof of computational real geometry? arXiv preprint arXiv:2004.04034 (2020)
14. Davenport, J.H., Heintz, J.: Real quantifier elimination is doubly exponential. Journal of Symbolic Computation **5**(1), 29–35 (1988). https://doi.org/10.1016/S0747-7171(88)80004-X

15. De Moura, L., Bjørner, N.: Z3: An efficient smt solver. In: International conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 337–340. Springer (2008). https://doi.org/10.1007/978-3-540-78800-3_24
16. Dolzmann, A., Sturm, T.: Redlog: Computer algebra meets computer logic. Acm Sigsam Bulletin **31**(2), 2–9 (1997). https://doi.org/10.1145/261320.261324
17. Dutertre, B.: Yices 2.2. In: International Conference on Computer Aided Verification. pp. 737–744. Springer (2014). https://doi.org/10.1007/978-3-319-08867-9_49
18. Ermon, S., Le Bras, R., Gomes, C.P., Selman, B., van Dover, R.B.: Smt-aided combinatorial materials discovery. In: Cimatti, A., Sebastiani, R. (eds.) Theory and Applications of Satisfiability Testing – SAT 2012. pp. 172–185. Springer (2012). https://doi.org/10.1007/978-3-642-31612-8_14
19. Fagerberg, R., Flamm, C., Merkle, D., Peters, P.: Exploring chemistry using smt. In: Milano, M. (ed.) Principles and Practice of Constraint Programming. pp. 900–915. Springer (2012). https://doi.org/10.1007/978-3-642-33558-7_64
20. Faure, G., Nieuwenhuis, R., Oliveras, A., Rodríguez-Carbonell, E.: Sat modulo the theory of linear arithmetic: Exact, inexact and commercial solvers. In: Kleine Büning, H., Zhao, X. (eds.) Theory and Applications of Satisfiability Testing – SAT 2008. pp. 77–90. Springer Berlin Heidelberg (2008). https://doi.org/10.1007/978-3-540-79719-7_8
21. Fontaine, P., Ogawa, M., Sturm, T., Vu, X.T.: Subtropical satisfiability. In: International Symposium on Frontiers of Combining Systems. pp. 189–206. Springer (2017). https://doi.org/10.1007/978-3-319-66167-4_11
22. Fontaine, P., Ogawa, M., Sturm, T., Vu, X.T., et al.: Wrapping computer algebra is surprisingly successful for non-linear smt. In: SC-square 2018-Third International Workshop on Satisfiability Checking and Symbolic Computation (2018)
23. Gao, S., Kong, S., Clarke, E.M.: dreal: An smt solver for nonlinear theories over the reals. In: International conference on automated deduction. pp. 208–214. Springer (2013). https://doi.org/10.1007/978-3-642-38574-2_14
24. Grigor'ev, D.Y., Vorobjov, N.: Solving systems of polynomial inequalities in subexponential time. Journal of Symbolic Computation **5**, 37–64 (1988). https://doi.org/10.1016/S0747-7171(88)80005-1
25. Hong, H.: An improvement of the projection operator in cylindrical algebraic decomposition. In: International Symposium on Symbolic and Algebraic Computation. pp. 261–264 (1990). https://doi.org/10.1145/96877.96943
26. Hong, H.: Comparison of several decision algorithms for the existential theory of the reals. resreport, Johannes Kepler University (1991)
27. Jovanovic, D., Dutertre, B.: Libpoly: A library for reasoning about polynomials. In: Satisfiability Modulo Theories. CEUR Workshop Proceedings, vol. 1889. CEUR-WS.org (2017), http://ceur-ws.org/Vol-1889/paper3.pdf
28. Jovanović, D., de Moura, L.: Solving non-linear arithmetic. In: Automated Reasoning. LNCS, vol. 7364, pp. 339–354 (2012). https://doi.org/10.1007/978-3-642-31365-3_27
29. Košta, M., Sturm, T.: A generalized framework for virtual substitution. CoRR **abs/1501.05826** (2015)
30. Kremer, G.: Cylindrical Algebraic Decomposition for Nonlinear Arithmetic Problems. Ph.D. thesis, RWTH Aachen University (2020). https://doi.org/10.18154/RWTH-2020-05913
31. Kremer, G., Brandt, J.: Implementing arithmetic over algebraic numbers: A tutorial for lazard's lifting scheme in cad. In: Symbolic and Numeric Algorithms for Scientific Computing. pp. 4–10 (2021). https://doi.org/10.1109/SYNASC54541.2021.00013

32. Kremer, G., Ábrahám, E.: Fully incremental cylindrical algebraic decomposition. Journal of Symbolic Computation **100**, 11–37 (2020). https://doi.org/10.1016/j.jsc.2019.07.018
33. Lazard, D.: An Improved Projection for Cylindrical Algebraic Decomposition, chap. 29, pp. 467–476. Springer (1994). https://doi.org/10.1007/978-1-4612-2628-4_29
34. Loup, U., Scheibler, K., Corzilius, F., Ábrahám, E., Becker, B.: A symbiosis of interval constraint propagation and cylindrical algebraic decomposition. In: Automated Deduction. LNCS, vol. 7898, pp. 193–207 (2013). https://doi.org/10.1007/978-3-642-38574-2_13
35. McCallum, S.: An improved projection operation for cylindrical algebraic decomposition. In: Caviness, B.F. (ed.) European Conference on Computer Algebra. LNCS, vol. 204, pp. 277–278. Springer Berlin Heidelberg (1985). https://doi.org/10.1007/3-540-15984-3_277
36. McCallum, S., Parusiński, A., Paunescu, L.: Validity proof of lazard's method for cad construction. Journal of Symbolic Computation **92**, 52–69 (2019). https://doi.org/10.1016/j.jsc.2017.12.002
37. de Moura, L., Jovanović, D.: A model-constructing satisfiability calculus. In: Giacobazzi, R., Berdine, J., Mastroeni, I. (eds.) Verification, Model Checking, and Abstract Interpretation. LNCS, vol. 7737, pp. 1–12 (2013). https://doi.org/10.1007/978-3-642-35873-9_1
38. Nieuwenhuis, R., Oliveras, A., Tinelli, C.: Solving sat and sat modulo theories: from an abstract davis-putnam-logemann-loveland procedure to dpll(t). Journal of the ACM **53**(6), 937–977 (2006). https://doi.org/10.1145/1217856.1217859
39. Renegar, J.: A faster pspace algorithm for deciding the existential theory of the reals. In: Symposium on Foundations of Computer Science. pp. 291–295 (1988). https://doi.org/10.1109/SFCS.1988.21945
40. Reynolds, A., Tinelli, C., Jovanovic, D., Barrett, C.W.: Designing theory solvers with extensions. In: Dixon, C., Finger, M. (eds.) Frontiers of Combining Systems. LNCS, vol. 10483, pp. 22–40 (2017). https://doi.org/10.1007/978-3-319-66167-4_2
41. Roselli, S.F., Bengtsson, K., Åkesson, K.: Smt solvers for job-shop scheduling problems: Models comparison and performance evaluation. In: International Conference on Automation Science and Engineering (CASE). pp. 547–552 (2018). https://doi.org/10.1109/COASE.2018.8560344
42. SMT-COMP 2021. https://smt-comp.github.io/2021/ (2021)
43. Tung, V.X., Van Khanh, T., Ogawa, M.: raSAT: An smt solver for polynomial constraints. In: International Joint Conference on Automated Reasoning. pp. 228–237. Springer (2016). https://doi.org/10.1007/978-3-319-40229-1_16
44. Weispfenning, V.: Quantifier elimination for real algebra—the quadratic case and beyond. Applicable Algebra in Engineering, Communication and Computing **8**(2), 85–101 (1997). https://doi.org/10.1007/s002000050055
45. Zohar, Y., Irfan, A., Mann, M., Niemetz, A., Nötzli, A., Preiner, M., Reynolds, A., Barrett, C., Tinelli, C.: Bit-precise reasoning via int-blasting. In: Finkbeiner, B., Wies, T. (eds.) Verification, Model Checking, and Abstract Interpretation. pp. 496–518. Springer (2022). https://doi.org/10.1007/978-3-030-94583-1_24