An SMT solver for non-linear real arithmetic inside maple

Sadeghimanesh, A. & England, M.

DOI 10.1145/3572867.3572880
IESSN 1932-2232

Publisher: ACM

An SMT solver for non-linear real arithmetic inside Maple

AmirHosein Sadeghimanesh*†and Matthew England*‡

Abstract

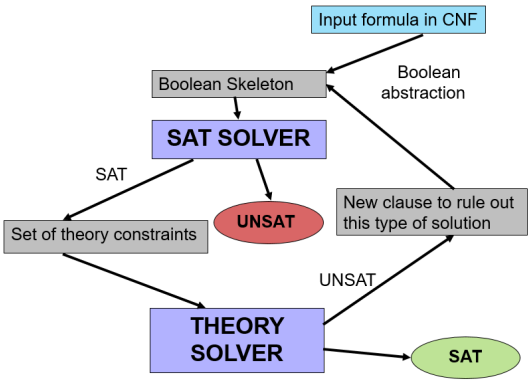
We report on work-in-progress to create an SMT-solver inside Maple for non-linear real arithmetic (NRA). We give background information on the algorithm being implemented: cylindrical algebraic coverings as a theory solver in the lazy SMT paradigm. We then present some new work on the identification of minimal conflicting cores from the coverings.

1 Introduction to SMT

It is well known that despite being NP-Hard, very large instances of the Boolean SAT problem can be tackled by modern SAT-solvers. Satisfiability Module Theory (SMT) solvers aim to reproduce this success for logical satisfiability problems where the atoms come from other theories. The theory of non-linear real arithmetic (NRA) is where the atoms are non-linear polynomial constraints over the reals: algorithms for these are traditional tackled by computer algebra and the ISSAC community.

1.1 Lazy SMT

The Lazy SMT framework is the most established[1]. Here, the input formula is transformed into its *Boolean Skeleton* (replace theory atoms by boolean variables). If the SAT-solver finds this unsatisfiable then so is the original problem. Otherwise, it will propose a solution which implies a set of the original theory constraints true. We check if this is possible using software from the relevant theory domain. If so, we obtain a model satisfying solution. If not, then the theory solver provides a new clause, logically implied by the original input formula, ruling out the proposed solution of the SAT solver. We iterate, forcing the SAT solver to suggest an alternative solution to the Boolean skeleton.



In the case of NRA the theory solver maybe an algorithm from symbolic computation, such as Cylindrical Algebraic Decomposition (CAD). The presentation above implies a simple integration of existing tools, but for this framework to be efficient the theory solver must adapt the data-structures in its algorithms so they can be reused in repeated calls with only slightly different input. Such an approach for CAD was presented in [6], and we are building our own in Maple.

*Research Centre for Computational Sciences and Mathematical Modelling, Coventry University, UK
†AmirHossein.Sadeghimanesh@coventry.ac.uk
‡Matthew.EnglandAmirHossein.Sadeghimanesh@coventry.ac.uk

[1]We note also the MCSAT framework which is used successfully to tackle NRA problems [4].

1.2 Cylindrical Algebraic Coverings

CAD is used as a theory solver by building a decomposition sign-invariant for the polynomials in the constraints, then analysing their truth on a sample point of each cell. In [1] a re-purposing of CAD theory was presented, more efficient for use in the Lazy SMT paradigm. It searched for a satisfying real model one dimension at a time. When a model cannot be extended then UNSAT cells (intervals) are produced to cover the dimension above, and their projection used to define the generalisation of the model on the dimension below. We decompose according to the real roots of the projection: the interval where the remaining model resides is concluded UNSAT for the same reason as the model. By using a covering instead of a decomposition, fewer cells and smaller projection operations than a full CAD are possible [1]. The method was inspired by the NLSAT algorithm [4] but operates within the Lazy SMT framework allowing for greater composability with other SMT theory modules. A report on the implementation in CVC5 suggested this was the current state-of-the-art for NRA [7], while [2] suggested that coverings may be advantageous in proof generation.

2 Identifying Minimal Conflicting Cores

Recall that in the Lazy SMT paradigm, when the theory solver determines a set of constraints cannot be simultaneously satisfied, it must produce a new clause to add to the formula ruling out this solution to the Boolean skeleton. This could simply be the negation of the solution just proposed by the SAT solver. However, it would preferable to identify a (minimal) subset of constraints which are together unsatisfiable, and use this instead to rule out at once multiple solutions in the logical solution space. This is referred to as the *(minimal) conflicting/UNSAT core*[2].

2.1 Prior work

With CAD as a theory solver this requires the solution to a *Set Covering Problem* (SCP). In general, a SCP is defined with a universe set U and a collection E of sets whose union equals the universe; with the problem to identify the smallest sub-collection from E whose union is still the universe. Here, U is the cells of the CAD, and E a collection of sets, one for each constraint, of the cells on which it is unsatisfiable. Solutions were published in [3], and [5], utilising heuristic approaches.

There has been little work on UNSAT core generation for the coverings method: [7, page 38] states that one may simply collect all constraints used for the intervals (which may be less than all constraints provided), while acknowledging there may be something more efficient.

2.2 Minimal conflicting cores for coverings

A brute force approach to the problem would be to repeatedly query the theory solver with different subsets. But NRA calls can be expensive, even if reuse infrastructure exists, so we do not pursue this. Neither can we directly apply the approach for CAD based on a final analysis. With coverings we make choices on which constraints to use gradually during the algorithm; with these determining the projection polynomials used to make the generalisation. If a post-analysis were to choose different constraints, then the conclusions would not necessarily be valid over the original generalisation.

Instead, we must recognise that any optimization must take place alongside the sequential decisions during cover construction. We note this approach means we need not store the full

[2]Note this is analogous to the generation of learnt clauses in CDCL style conflict resolution for SAT solvers.

dimensional models once we have generalised[3]. Our implementation will store information for each interval on the constraints which led to its UNSAT conclusion. This is passed on in generalisations until we have a covering of the real line for the first variable, where it becomes the conflicting core.

2.3 Two variants of the Set Covering Problem

2.3.1 Simple variant

Consider first the case of a covering in the top level (i.e. we have a model of dimension one lower than the problem). For each of the intervals generated we can find at least one, possibly more, of the original constraints that are unsatisfied. We can here consider a minimisation SCP similar to before. Let the universal set, U, be the set of sample points for the intervals and assume that $n = |U|$. Let $E = \{A_1, \ldots, A_r\}$ where r is the original number of constraints and A_i is the set of sample points that do not satisfy the constraint with label i (univariate after substitution of the model). Covering U with a minimal amount of E will give us the minimal conflicting core.

2.3.2 More involved variant

In lower layers of the algorithm we encounter a different form of the SCP. It is likely that to conclude UNSAT on the dimension above required a combination of constraints that could not be satisfied together. In that case the label will be not a singleton but the set of constraint labels. Suppose one such interval is covered by $\{1, 2\}$ and another by $\{1\}$. It would not be correct to take just constraint 1 as a minimal core because we needed constraints one and two in combination to conclude UNSAT over the first interval. So we can not define the A_is of E as those intervals unsatisfiable for any one constraint when forming our SCP. We need a new SCP formulation. We call this case SCPR: a *Set Covering Problem with Reasons*.

We still let U be the set of intervals, but this time define R as the set of reasons. For the collection E of A_is, the labels i on these subsets from U will no longer be the elements of R, but instead subsets of R. So the label says what reasons must be present all together so that this set of intervals is UNSAT. I.e. $E \subseteq P(U) \times P(R)$ where P denotes the power set.

We define a running example with $U = \{1, \ldots, 8\}$ intervals, $R = \{1, 2, 3, 4, 5\}$ reasons. and

$$E = \{(\{1,2,3,4\},\{1,2\}), (\{5,6,7,8\},\{1,3\}), (\{2,3,4,5,6,7\},\{4\}), (\{8\},\{5\})\}.$$

The first element denotes that intervals $1-4$ cannot satisfy the first two constraints simultaneously.

We define $m = |E|$ and define two binary membership matrices $M = \left[u_{ij}\right]_{m\times n}$ and $N = \left[v_{ij}\right]_{r\times m}$ such that $u_{ij} = 1$ if and only if the j-th element of U belongs to the first set in the i-th pair in E, and $v_{ij} = 1$ if and only if the i-th element of R belongs to the second set in the j-th pair in E. For every $i = 1, \ldots, m$ we let x_i be a binary variable associated to the first sets in the pairs in E, indicating whether or not this is used in the solution to the problem.

2.4 Optimization problems

The simple SCP problem gives the following linear binary optimization problem.

$$\begin{array}{ll} \min & x_1 + \cdots + x_m \\ \text{s.t. } \forall j = 1, \ldots, n: & \sum_{i=1}^{m} u_{ij} x_i \geq 1. \end{array} \tag{1}$$

[3]Although if expecting multiple calls to the theory solver it may still be advantageous to keep this data.

It can be tackled with MAPLE's `Optimization:-Minimize` command with option `assume=binary`.

For the more involved variant, we want to minimize the number of reasons and not necessarily the number of subsets of U that we are using. So we define new binary variables y_i for $i = 1, \ldots, r$ where $y_i = 1$ if and only if i-th reason is present. It is clear that for every $j = 1, \ldots, m$ we have $x_j = \prod_{i=1}^{r} y_i^{v_{ij}}$ where $y_i^0 = 1$. Therefore the SCPR can be formulated as follows:

$$\begin{array}{ll} \min & y_1 + \cdots + y_r \\ \text{s.t. } \forall j = 1, \ldots, n: & \sum_{i=1}^{m} u_{ij} (\prod_{k=1}^{r} y_k^{v_{ki}}) \geq 1. \end{array} \tag{2}$$

The constraints are not necessarily linear. For our example we get $y_1 y_2 \geq 1$, $y_1 y_2 + y_4 \geq 1$, $y_1 y_3 + y_4 \geq 1$ and $y_1 y_3 + y_5 \geq 1$ (we have 4 instead of 8 as some were repeated). Unfortunately the `Optimization` package of MAPLE does not support non-linear binary optimization problems.

Instead, we tackle the SCPR by translating it to a SAT problem and using a SAT-solver (the `Logic:-Satisfy` command in MAPLE allows to use either `legacy` or `MapleSAT`).

To formulae the SAT problem we redefine the binary variables y_is to be boolean, the constraint $\sum_{i=1}^{m} u_{ij} (\prod_{k=1}^{r} y_k^{v_{ki}}) \geq 1$ becomes $\bigvee_{\substack{1 \leq i \leq m \\ u_{ij}=1}} \bigwedge_{\substack{1 \leq k \leq r \\ v_{ki}=1}} y_k = true$, and then we conjunct all the constraints together. We then make SAT calls which assume an increasing number of reasons, t. We start by letting $t = 1$ and conjunct to the formula one that requires exactly one y_i true. If this is not satisfiable we replace with a formula that requires two decision variables true and continue in this way until finding a t that is SAT.

The logical formula for exactly t of r boolean variables being true can be written as disjunction of conjunctions of literals a_is, where the disjunction indices run over all binary vectors of length r with exactly t ones, and the literal a_i is y_i if the i-th entry of the vector is 1, otherwise $\texttt{not}(y_i)$.

We created a package for solving SCP and SCPR problems called `SCPPack`. Whenever reason labels are all singletons, as in the top level, we use the SCP formulation and MAPLE's `Optimization`; otherwise we use our SCPR formulation together with a SAT-solver. Our implementation returns the set $\{1, 2, 3\}$ for our example. An initial release of `SCPPack` can be found online here: `https://doi.org/10.5281/zenodo.6646133`

Our SCPR algorithm gives the optimal solution for the SCPR itself, but does not guarantee the minimal explanation of UNSAT. Future work will explore its effect in our SMT solver.

acknowledgement

The authors acknowledge the support of EPSRC Grant EP/T015748/1, "Pushing Back the Doubly-Exponential Wall of Cylindrical Algebraic Decomposition".

References

[1] 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*, 119:100633, 2021. URL: `https://doi.org/10.1016/j.jlamp.2020.100633`.

[2] E. Ábrahám, J.H. Davenport, M. England, G. Kremer, and Z. Tonks. New opportunities for the formal proof of computational real geometry? In *Proc.* SC^2 *2020*, CEUR-WS 2752, pages 178–188, 2020. URL: `http://ceur-ws.org/Vol-2752/`.

[3] W. Hentze. Computing minimal infeasible subsets for Cylindrical Algebraic Decomposition. *Thesis*, 2017. URL: `https://ths.rwth-aachen.de/wp-content/uploads/sites/4/teaching/theses/hentze_bachelor.pdf`

[4] D. Jovanovic and L. de Moura. Solving non-linear arithmetic. In B. Gramlich, D. Miller, U. Sattler, editors, *Automated Reasoning: Proc. IJCAR 2012*, LNCS 7364, pages 339–354. Springer, 2012. URL: https://doi.org/10.1007/978-3-642-31365-3_27.

[5] M. Jaroschek, P.F. Dobal and P. Fontaine. Adapting Real Quantifier Elimination Methods for Conflict Set Computation. In: *Proc. FroCoS 2015*, LNCS 9322, pages 151–166, 2015. URL: https://doi.org/10.1007/978-3-319-24246-0_10

[6] 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.

[7] G. Kremer, E. Ábrahám, M. England, and J.H. Davenport. On the implementation of cylindrical algebraic coverings for satisfiability modulo theories solving. In *Proc. SYNASC 2021*, pages 37–39, 2021. URL: http://dx.doi.org/10.1109/SYNASC54541.2021.00018.