Under-Approximation of a Single Algebraic Cell (Extended Abstract)

Valentin Promies[1,*], Jasper Nalbach[1] and Erika Ábrahám[1]

[1]RWTH Aachen University, Germany

Abstract

Single cell construction is a crucial subroutine in modern SMT solvers for non-linear real arithmetic. In this extended abstract, we adapt a recent approach by dynamically under-approximating the cell boundaries using linear polynomials, which can greatly reduce the effort of resultant computations, while maintaining the sign-invariance properties of the cell. Although one must be careful to ensure termination, first experiments suggest that this modification pays off in the context of SMT solving.

Keywords

Single Cell Construction, Approximation, Real Algebra, Cylindrical Algebraic Decomposition

1. Introduction

The NLSAT algorithm [1] is a modern and complete approach for deciding the satisfiability of quantifier-free formulas in the theory of non-linear real arithmetic, and it relies heavily on a subroutine for *single cell construction* (SCC): Given a sample point $s \in \mathbb{R}^n$ and a finite set $P \subset \mathbb{Q}[x_1, \ldots, x_n]$ of polynomials, the task is to compute the representation of a cell $S \subset \mathbb{R}^n$, so that $s \in S$ and the polynomials in P are *sign-invariant* over S. The cell S is connected and *algebraic*, i.e. its boundaries are defined by the roots of some polynomials.

NLSAT invokes this subroutine when it finds a sample s at which the given formula is conflicting: the conflict is generalized to S, which is then excluded from the search space. Accordingly, the excluded cell should be as large as possible. However, the representation should be efficiently usable by NLSAT, and, as SCC is based on the *cylindrical algebraic decomposition* [2], its computational effort can grow quickly. Thus, S is generally not inclusion-maximal in practice.

Recently, Nalbach et al. introduced a *levelwise* approach to SCC [3], which evolved from the method of Brown and Košta [4]. We modify this approach by allowing it to dynamically under-approximate cell boundaries using low-degree polynomials. This reduces the computational effort of the construction, but it also affects the quality of the cell for NLSAT and the methods' completeness.

2. Levelwise Single Cell Construction

Originally, Nalbach et al. [3] formulated the levelwise single cell construction by means of a proof system, which offers great flexibility in heuristic choices. We now give a brief recall of the method, but we omit details and focus here on those parts that we will adapt later, in Section 3.

For a given finite set $P \subset \mathbb{Q}[x_1, \ldots, x_n]$ of n-variate polynomials and a given sample point $s \in \mathbb{R}^n$, the method constructs a representation $\mathtt{I} = (I_1, \ldots, I_n)$ of a cell $S \subseteq \mathbb{R}^n$ so that $s \in S$ and for all $p \in P$ holds $\forall s' \in S.\, sign(p(s')) = sign(p(s))$. In the output representation, each I_i is a *symbolic interval* which bounds the value of x_i w.r.t the variables $x_1, \ldots, x_{i-1}$ of the lower levels, either by a lower and an upper bound $(l(x_1, \ldots, x_{i-1}) < x_i < u(x_1, \ldots, x_{i-1}))$ or by an equality $(x_i = b(x_1, \ldots, x_{i-1}))$.

9th International Workshop on Satisfiability Checking and Symbolic Computation, July 2, 2024, Nancy, France, Collocated with IJCAR 2024

*Corresponding author.

promies@cs.rwth-aachen.de (V. Promies); nalbach@cs.rwth-aachen.de (J. Nalbach); abraham@cs.rwth-aachen.de (E. Ábrahám)

0000-0002-3086-9976 (V. Promies); 0000-0002-2641-1380 (J. Nalbach); 0000-0002-5647-6134 (E. Ábrahám)

The method maintains a working set P_* of polynomials, which is initialized as $P_* := P$.

For $i = n, \dots, 1$ (i.e. starting with I_n and iterating down to I_1), the interval I_i at level i is then computed as follows.

1. Let $P_i := (P_* \cap \mathbb{Q}[x_1, \dots x_i]) \setminus \mathbb{Q}[x_1, \dots x_{i-1}]$.
2. Isolate the real roots $\bigcup_{p \in P_i} \{r \in \mathbb{R} \mid p(s_1, \dots, s_{i-1}, r) = 0\}$ and order them, together with the sample coordinate s_i, giving $-\infty < r_1 \leq \dots \leq r_l < s_i < r_{l+1} \leq \dots \leq r_m < \infty$. In this paper, we only consider the case that s is not a root of any polynomial in P_*.
3. In a neighborhood of the sample, each root can be generalized to a continuous function, encoded as an *indexed root expression (IRE)* of the form $root_{x_i}[p, j] \colon \mathbb{R}^{i-1} \to \mathbb{R} \cup \{\bot\}$, which maps a given $(i-1)$-dimensional sample to the j-th real root of the polynomial p in x_i if it exists and otherwise returns $\bot$ (i.e. "undefined"). For an IRE ξ, we refer to the according polynomial by $\xi.p$.
4. For $i \in \{1, \dots, m\}$, let ξ_i be the IRE corresponding to the root r_i, and let $\Xi = \{\xi_1, \dots, \xi_m\}$.
5. Set $I_i := (\xi_l(x_1, \dots, x_{i-1}) < x_i < \xi_{l+1}(x_1, \dots, x_{i-1}))$.
6. The symbolic interval I_i is only correct in a certain neighborhood of the sample, which is why the underlying cell defined by $(I_1, \dots, I_{i-1})$ is restricted using a *projection*.
   a) To ensure well-definedness of the root functions over the underlying cell, it is restricted so that the polynomials are *delineable*, by adding *discriminants* $disc_{x_i}[p] \in \mathbb{Q}[x_1, \dots, x_{i-1}]$ and some *coefficients* of the polynomials p from P_i to the working set P_*.
   b) To ensure that no root function crosses the boundaries defined by I_i, choose a relation $\preceq \subset \Xi \times \Xi$ so that $(\xi_l, \xi_{l+1}) \in \preceq$, $\{(\xi_j, \xi_l) \mid j < l\} \subset \preceq^*$ and $\{(\xi_{l+1}, \xi_j) \mid l+1 < j \leq m\} \subset \preceq^*$, where $\preceq^*$ is the transitive closure of $\preceq$. Then update $P_* := P_* \cup \{res_{x_i}[\xi.p, \xi'.p] \mid \xi \preceq \xi'\}$, where the *resultant* $res_{x_i}[p, q] \in \mathbb{Q}[x_1, \dots, x_{i-1}]$ of $p, q \in \mathbb{Q}[x_1, \dots, x_i]$ is a polynomial whose roots are exactly the points $t \in \mathbb{R}^{i-1}$ where $p(t, x_i) \in \mathbb{R}[x_i]$ and $q(t, x_i) \in \mathbb{R}[x_i]$ have a common root. If the resultant is order-invariant, then the roots of p and q do not cross. This way, the boundaries are protected.

Figure 1 illustrates an example with a given sample $s \in \mathbb{R}^2$ and polynomials $P = \{p_1, p_2, p_3\} \subset \mathbb{Q}[x_1, x_2]$. For $j \in \{1, 2, 3\}$, the line labeled with p_j shows those points $r \in \mathbb{R}^2$ with $p_j(r) = 0$. We start at level 2, where $P_2 = P$. Fixing x_1 to s_1 and isolating the real roots of P_2 yields one root of p_2 below s_2 and one root of each polynomial above s_2. The IREs corresponding to the roots closest to s_2 define the symbolic interval $I_2 = (root_{x_2}[p_2, 1](x_1) < x_2 < root_{x_2}[p_3, 1](x_1))$, as shown in Figure 1a. Now, we need to ensure that this interval is correct for all values of x_1 in the underlying cell which will be computed at level 1. We choose $\xi_1 \preceq \xi_2$, $\xi_2 \preceq \xi_3$ and $\xi_2 \preceq \xi_4$, and add thus the resultants of p_3 with p_1 and p_2, whose roots are given by the dashed lines in Figure 1b. On level 1, we again isolate those roots and use the closest to s_1 as interval boundaries, giving $I_1 = (\xi'_1 < x_1 < \xi'_2)$ and thus, (I_1, I_2) define the shaded area shown in Figure 1c. Note that the crossing of ξ_3 and ξ_4 over I_1 is irrelevant, and the corresponding resultant of p_1 and p_2 was avoided.

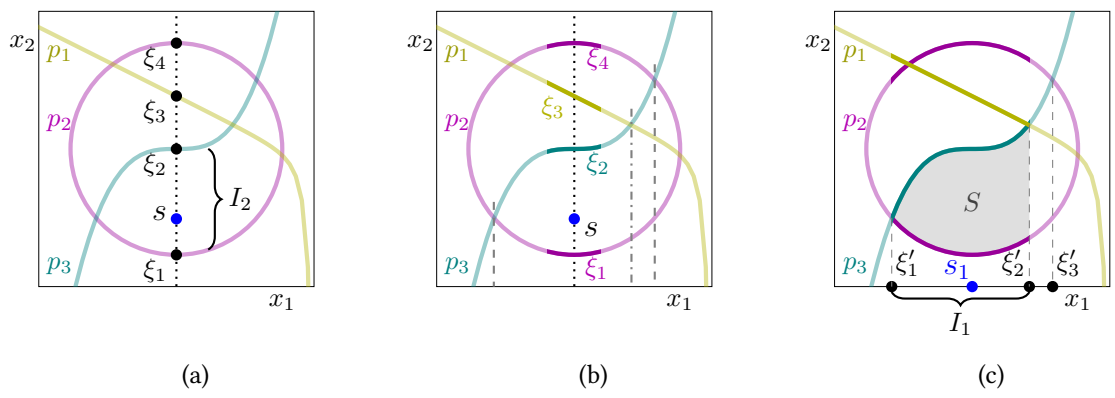

Figure 1: Illustration of the levelwise single cell construction.

3. Under-Approximation with Low-Degree Polynomials

The most resource-intensive part of single cell construction is the computation of resultants and discriminants, and we tackle the former. The computational effort and the degree of $res_{x_i}[p, q]$ scales with $deg(p) \cdot deg(q)$. Thus, if p and q have high degree, then not only is their resultant more expensive to compute, but it also yields a polynomial of even higher degree which will be involved in resultants in the subsequent levels. On the other hand, if p is of the form $x_i - a$, then its resultant with q is simply equal to $q(x_1, \ldots, x_{i-1}, a)$.

As seen in the example, one can choose the relation $\preceq$ so that all resultants in the current projection take one of the boundary-defining polynomials as input. In [3], this is called the Biggest-Cell heuristic. Our idea is to dynamically add artificial cell boundaries defined by linear polynomials and use that heuristic so that the required resultants will be easy to calculate.

When computing the interval I_i, if the upper (or lower) bound of the interval would be defined by a high-degree polynomial, we insert an "artificial" root r^* between the sample s_i and the actual bound. This root is generalized to an IRE ξ^* defined by a low-degree polynomial, e.g. $h(x_i) = x_i - r^*$, and now we use ξ^* as the upper interval bound.

Figure 2 illustrates our idea. Interestingly, while we under-approximate the symbolic interval I_2, the resulting cell is not a subset of the cell in Figure 1, because I_1 is now defined by the resultants of h with p_2 and p_3. This means that our approximated interval can lead to larger underlying cells.

Our modification has two main benefits: the following projection step is much easier to compute; and it is easier for NLSAT to check whether a sample lies in the excluded cell, due to a less complex cell description that avoids algebraic number computations.

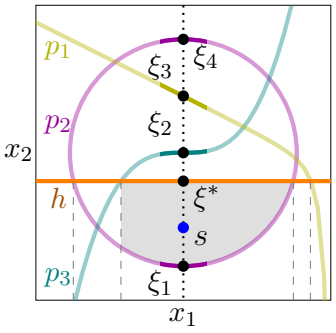

Figure 2: Approximated cell using an additional linear polynomial.

Correctness The approximation does not affect the correctness of the construction. To see this, consider the original construction, but for a modified input $P \cup H$, where H contains the approximation polynomials that were dynamically added. This will yield the same cell as our version gives for the input P, and by the correctness of the original method, this cell contains s and all polynomials in $P \cup H$ (in particular those in P) are sign-invariant over it.

Incompleteness When used in NLSAT, the approximation can lead to an incomplete (non-terminating) algorithm, because the union of the approximated cells might converge to the inclusion-maximal cell without ever covering it entirely and thus without ever considering the entire solution space. This behavior can be circumvented by allowing NLSAT to only approximate a limited number of cells before resorting to the original construction, which assures completeness.

4. Conclusions

Many SMT problems can be solved efficiently using approximative methods, like incremental linearization [5] or ICP [6], though they are incomplete. We presented an approach that weaves dynamically into the complete NLSAT method.

Experiments The levelwise SCC and a prototype of our modification are implemented as backends for an NLSAT-style algorithm in the SMT-RAT solver [7, 8], and we conducted first experiments with promising results. We obtained the best performance when applying linear approximations to cell boundaries that are defined by a polynomial with degree 5 or higher, and not approximating after 50 calls to the SCC. When executed on the QF_NRA benchmark set from SMT-LIB [9], our modification

solved 44 instances more than the original version. When using the approximation, the solver needs more calls to the SCC on average, but these calls are processed faster, leading to a lower mean runtime.

Next Steps We are currently working on different generalizations of our idea. Firstly, one can use more intricate approximations, e.g. piecewise linear functions or Taylor polynomials, which hopefully results in a cell that is more similar to the original one, but which also induces a larger overhead. In the case of Taylor polynomials, a closer approximation could be obtained by using degree two or three, instead of linear polynomials.

Secondly, the levelwise framework also allows us to add artificial roots between any two root functions, not just as a cell boundary. In our example, we could have inserted ξ^* (and h) between ξ_2 and ξ_3. With an appropriate choice of $\preceq$, this lets us replace $res_{x_2}[p_3, p_1]$ by the less complex resultants $res_{x_2}[p_3, h]$ and $res_{x_2}[h, p_1]$, while maintaining the original interval $I_2 = (\xi_1 < x_2 < \xi_2)$. This might still lead to smaller intervals at the lower levels, due to the new resultants.

Finally, we are interested in transferring our ideas to the *cylindrical algebraic coverings* method [10], which also uses the framework of the levelwise method.

Acknowledgments

Jasper Nalbach and Valentin Promies were supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Association) as part of AB 461/9-1 SMT-ART. Jasper Nalbach was supported by the DFG RTG 2236 UnRAVeL.

References

[1] D. Jovanovic, L. M. de Moura, Solving non-linear arithmetic, in: Proc. of the 6th Int. Joint Conf. on Automated Reasoning (IJCAR'12), volume 7364 of *LNCS*, Springer, 2012, pp. 339–354. doi:10.1007/978-3-642-31365-3_27.

[2] G. E. Collins, Quantifier elimination for real closed fields by cylindrical algebraic decomposition, in: H. Barkhage (Ed.), Automata Theory and Formal Languages, 2nd GI Conference, Kaiserslautern, May 20-23, 1975, volume 33 of *Lecture Notes in Computer Science*, Springer, 1975, pp. 134–183. doi:10.1007/3-540-07407-4_17.

[3] J. Nalbach, E. Ábrahám, P. Specht, C. W. Brown, J. H. Davenport, M. England, Levelwise construction of a single cylindrical algebraic cell, Journal of Symbolic Computation 123 (2024) 102288. doi:10.1016/j.jsc.2023.102288.

[4] C. W. Brown, M. Košta, Constructing a single cell in cylindrical algebraic decomposition, J. Symb. Comput. 70 (2015) 14–48. doi:10.1016/J.JSC.2014.09.024.

[5] A. Cimatti, A. Griggio, A. Irfan, M. Roveri, R. Sebastiani, Invariant checking of NRA transition systems via incremental reduction to LRA with EUF, CoRR abs/1801.08718 (2018). URL: http://arxiv.org/abs/1801.08718. arXiv:1801.08718.

[6] P. Van Hentenryck, D. McAllester, D. Kapur, Solving polynomial systems using a branch and prune approach, SIAM Journal on Numerical Analysis 34 (1997) 797–827. doi:10.1137/S0036142995281504.

[7] F. Corzilius, G. Kremer, S. Junges, S. Schupp, E. Ábrahám, SMT-RAT: An open source C++ toolbox for strategic and parallel SMT solving, in: Proc. of the 18th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT'15), volume 9340 of *LNCS*, Springer, 2015, pp. 360–368. doi:10.1007/978-3-319-24318-4_26.

[8] SMT-RAT, a toolbox for strategic and parallel satisfiability modulo theories solving, https://github.com/ths-rwth/smtrat, accessed 2024-06-01.

[9] C. Barrett, A. Stump, C. Tinelli, The SMT-LIB standard – version 2.0, in: Proc. of the 8th Int. Workshop on Satisfiability Modulo Theories (SMT '10), 2010. URL: http://theory.stanford.edu/~barrett/pubs/BST10.pdf.

[10] E. Ábrahám, J. H. Davenport, M. England, G. Kremer, Deciding the consistency of non-linear real arithmetic constraints with a conflict driven search using cylindrical algebraic coverings, J. Log. Algebraic Methods Program. 119 (2021) 100633. doi:10.1016/J.JLAMP.2020.100633.