Handling Polynomial and Transcendental Functions in SMT via Unconstrained Optimisation and Topological Degree Test[⋆]

Alessandro Cimatti[1], Alberto Griggio[1], Enrico Lipparini[1,2], Roberto Sebastiani[3]

[1] Fondazione Bruno Kessler, Trento, Italy
[2] DIBRIS, University of Genoa, Italy
[3] DISI, University of Trento, Italy

Abstract. We present a method for determining the satisfiability of quantifier-free first-order formulas modulo the theory of non-linear arithmetic over the reals augmented with transcendental functions. Our procedure is based on the fruitful combination of two main ingredients: unconstrained optimisation, to generate a set of candidate solutions, and a result from topology called the topological degree test to check whether a given bounded region contains at least a solution. We have implemented the procedure in a prototype tool called ugotNL, and integrated it within the MathSAT SMT solver. Our experimental evaluation over a wide range of benchmarks shows that it vastly improves the performance of the solver for satisfiable non-linear arithmetic formulas, significantly outperforming other available tools for problems with transcendental functions.

1 Introduction

When dealing with real arithmetic in SMT, a fundamental challenge is to go beyond the linear case ($\mathcal{LRA}$), by introducing nonlinear polynomials ($\mathcal{NRA}$), possibly augmented with transcendental functions like exponential and trigonometric ones ($\mathcal{NTA}$). In fact, the expressive power of $\mathcal{NTA}$ is required by many application domains (e.g. railways, aerospace, control software, and cyber-physical systems). Unfortunately, dealing with non-linearity is a very hard challenge. Going from SMT($\mathcal{LRA}$) to SMT($\mathcal{NRA}$) yields a complexity gap that results in a computational barrier in practice. Adding transcendental functions exacerbates the problem even further, because reasoning on $\mathcal{NTA}$ is undecidable [26]. Existing SMT solvers therefore have to resort to incomplete techniques in order to handle $\mathcal{NTA}$ constraints [7,17], which are however particularly ineffective at proving that a formula is satisfiable (i.e. that it has at least one model). One of the main sources of complexity is the need to provide exact answers: when an SMT solver says "sat", the input problem must indeed be satisfiable,

⋆ This work has been partly supported by project "AI@TN" funded by the Autonomous Province of Trento.

and not just “likely satisfiable” or “satisfiable with high probability”. Removing this requirement makes it possible to use approximate techniques, such as numerical methods or procedures based on weaker notions of satisfiability such as δ-satisfiability [18], which are typically significantly more scalable in practice than exact methods.

In this paper, we present a technique for significantly improving the effectiveness of SMT($\mathcal{NTA}$) solvers in determining that a formula is satisfiable, by exploiting a fruitful combination of approximate and exact techniques. Our procedure uses numerical methods based on *unconstrained global optimisation* to quickly identify (small) boxes containing candidate solutions for a given set/conjunction of $\mathcal{NRA}$ and $\mathcal{NTA}$ constraints, which are then analysed with a procedure whose main ingredient is the *topological degree test* [24,15] – a result from topology that guarantees the existence of a solution for a set of equalities if certain conditions are met – to confirm whether a candidate box contains at least one solution. The procedure is then plugged into an SMT context, which allows us to handle problems containing arbitrary Boolean combinations of constraints.

The main contribution of this work is an effective combination of numeric and symbolic methods that allows to significantly enhance the capability of state-of-the-art SMT solvers to determine the satisfiability of formulas containing $\mathcal{NTA}$ constraints, as demonstrated by our extensive experimental evaluation. To this extent, although all the ingredients we use are known, our overall procedure is, to the best of our knowledge, novel. The synergy between numerical optimisation and the topological degree test is essential for the viability of our approach, as none of the two techniques in isolation is effective in practice. On one hand, being based on numerical methods, unconstrained global optimisation alone cannot detect exact solutions, but only approximate ones. On the other hand, the topological degree test alone is not immediately applicable to arbitrary sets of constraints, as it works only for problems in a specific form, in which (i) there are only equations, (ii) the number of equations is equal to the number of variables, (iii) all variables are bounded, and (as a more empirical requirement rather than theoretical limitation) (iv) the bounds on the variables are “sufficiently small” for the practical effectiveness of the test. The first limitation has been tackled in [16] by pairing the topological degree test with interval arithmetic to deal with inequalities. In this paper, we show how a further combination with numerical optimization can be exploited to obtain a practical and effective method that can be easily integrated in a modern SMT solver, thus overcoming the other three points.

In order to substantiate our claims, we have implemented our procedure in a prototype tool called ugotNL, and we have integrated it within the MathSAT SMT solver [8]. We have extensively evaluated our prototype on a wide range of $\mathcal{NRA}$ and $\mathcal{NTA}$ benchmarks, comparing it to the main state-of-the art tools. Our experimental evaluation shows that it vastly improves the performance of the MathSAT solver for satisfiable $\mathcal{NRA}$ formulas, significantly outperforming the other tools on $\mathcal{NTA}$ problems.

Related work. For $\mathcal{NRA}$, various techniques have been explored. Complete methods based on quantifier-elimination procedures such as Cylindric Algebraic Decomposition (CAD) [9] have been successfully implemented in several SMT-solvers (such as Z3[11], YICES[13], SMT-RAT[10]), proving their effectiveness especially when tightly integrated into the Boolean search through a model-constructing framework such as MCSAT [20][12]. However, their complexity is doubly-exponential in the worst case, and they cannot deal with transcendental functions.

For $\mathcal{NTA}$, there exist very few techniques able to prove satisfiability. Incremental linearization (IL) [7] starts from an abstract model and tries to check whether the formula is satisfiable under *all possible interpretations* (within a given bounded region) of the transcendental functions involved. This tactic works well when the transcendental functions are isolated in the formula, but it is quite ineffective when the transcendental component is more complex (expecially in the presence of equations). iSAT3 [17] implements a method based on a tight integration of Interval Constraint Propagation (ICP)[4] into the CDCL framework, and it is able to prove satisfiability if it finds a box in which every point is a solution.

Differently from these methods, our approach is not compelled to find more solutions than needed, and it is able to prove satisfiability even when the only models of the formula are isolated points. Interestingly, raSAT[27] combines ICP with the Generalized Intermediate Value Theorem (GIVT)[23], but does not support transcendental functions.

Other approaches, e.g. dREAL [19] and KSMT [5], rely on the notion of δ-satisfiability[18], which guarantees that there exists a perturbation (up to some $\delta > 0$ specified by the user) of the original formula that is satisfiable.[1] iSAT3 relies on a similar notion and, when not able to prove satisfiability nor to detect conflicts, returns a candidate solution. In comparison with these approaches, when we return "sat" we guarantee that the problem is actually satisfiable.

Content. The paper is organized as follows. In §2 we provide the necessary theoretical background; in §3 we describe how we use unconstrained optimisation to find candidate models; in §4 we describe a general procedure, restricted to conjunctions of $\mathcal{NTA}$ constraints, based on the topological degree test and interval arithmetic; in §5 we extend the previous procedure to general $\mathcal{NTA}$ formulas, following either an eager or a lazy approach; in §6 we present our experimental evaluation; in §7 we conclude.

2 Background

We work in the setting of SMT, with the quantifier-free theory of real arithmetic, either limited to polynomial constraints (denoted $\mathcal{NRA}$), or augmented with trigonometric and exponential transcendental functions (denoted $\mathcal{NTA}$). We

[1] Note that, according to this definition, a problem could be unsat and δ-sat at the same time

assume the standard notions of interpretation, model, satisfiability, validity and logical consequence.

We use the following notation. We write logical variables with $x_1, x_2, \ldots$, and values in $\mathbb{R}$ with $\mathrm{x}_1, \mathrm{x}_2, \ldots$. If t is a generic (quantifier-free) term, we write $[t]$ for its interpretation in the standard model of arithmetic. If ϕ is a formula, we denote with $\mathrm{Var}(\phi)$ the set of its (free) variables. We use f, g to denote logic symbols representing a polynomial or a transcendental function; when there is no ambiguity, we will use the same symbol also to denote the real function corresponding to its standard interpretation. We use boldface to denote vectors of values $\mathbf{x} \overset{\text{def}}{=} \{\mathrm{x}_1, \ldots, \mathrm{x}_\mathrm{m}\} \in \mathbb{R}^m$, and intervals $I \overset{\text{def}}{=} \{\mathrm{x} \in \mathbb{R} \mid \mathrm{a} \leq \mathrm{x} \leq \mathrm{b}\}$ with $[\mathrm{a}, \mathrm{b}]$ (or simply $[\mathrm{a}]$ when $\mathrm{a} \equiv \mathrm{b}$), where a and $\mathrm{b} \in \mathbb{Q}$. Given a vector $\mathbf{x} \in \mathbb{R}^m$, we denote with $\|\mathbf{x}\|_2$ its Euclidean norm (i.e. $\sqrt{\sum_i \mathrm{x_i}^2}$), and with $|\mathbf{x}|$ its maximum norm (i.e. $\max\{|\mathrm{x}_1|, \cdots, |\mathrm{x}_\mathrm{m}|\}$). If ϕ is a formula with $\mathrm{Var}(\phi) \equiv \{x_1, \ldots, x_m\}$, we denote with $M_\phi \overset{\text{def}}{=} \{\mathbf{x} \in \mathbb{R}^m \mid \mathbf{x} \text{ is a model of } \phi\}$ the set of its models.

We assume that the reader is familiar with the main theoretical and algorithmic concepts of SMT, as well as with its terminology. We recall that the lazy-SMT approach consists in building ad-hoc theory-specific procedures (called *theory solvers*, usually written just for conjunctions of literals, i.e. atomic formulas and their negations) and integrating them into a SAT-solver. The most used approach for lazy-SMT, called CDCL(T), is to modify the CDCL procedure [29] commonly used for SAT to work with formulas having a background theory T. We refer the reader to, e.g., [3] for more details on lazy SMT.

In the rest of this section, we introduce the necessary background techniques from the fields of unconstrained optimisation, interval arithmetic, and topology.

2.1 Unconstrained Global Optimisation

We say that $\mathbf{x}^*$ is a *local minimum* for $h : \mathbb{R}^m \to \mathbb{R}$, if there exists a neighborhood $S := \{\mathbf{x} \in \mathbb{R}^m : \|\mathbf{x}^* - \mathbf{x}\|_2 < \delta\}$ for some $\delta > 0$, such that $\forall \mathbf{x} \in S : h(\mathbf{x}^*) \leq h(\mathbf{x})$. We say that $\mathbf{x}^*$ is a *global minimum* for h if $\forall \mathbf{x} \in \mathbb{R}^m : h(\mathbf{x}^*) \leq h(\mathbf{x})$.

Unconstrained global optimisation is the problem of minimizing a function h on the entire space $\mathbb{R}^m$ of the real numbers. A common approach to tackle this problem is leveraging fast local optimisation techniques.

In this paper, we use a Monte Carlo Markov Chain method called Basin-hopping[28], based on the Metropolis-Hasting algorithm[21]. The idea of Basin-hopping is to do a random sampling of h to simulate a target distribution, and then alternate a local minimization phase with a stepping phase, used to decide, guided by the target distribution, how to *jump* from a local minimum to another. In particular, we use a slight modification of the algorithm that, given a maximum number of iterations, returns all the local minima found during the search.

2.2 Interval Arithmetic

Interval Arithmetic is a systematic approach to represent real numbers as intervals and to compute safe bounds that account for rounding errors. We define a *box* as a subset of $\mathbb{R}^m$ that is the Cartesian product of m intervals: $B = I_1 \times \cdots \times I_m \subset \mathbb{R}^m$. The width of an interval $I \overset{\text{def}}{=} [a_I, b_I]$ is defined as $\text{width}(I) \overset{\text{def}}{=} \mathrm{b_I} - \mathrm{a_I}$, and the width of a box is defined as $\text{width}(B) \overset{\text{def}}{=} \max_i(\text{width}(I_i))$. We can define several operations and relations between intervals, such as addition, multiplication, inclusion, and many more. For a more in-depth coverage of properties of and operations on intervals, we refer to [22].

We now give the definition of interval-computable functions, which plays an important role in our method. The intuition is that a function is interval-computable if it is possible to compute arbitrarily precise images for every interval domain. It has been proved that every function in $\mathcal{NTA}$ is interval computable (we refer to section 5.4 of [22] for the proof)

Definition 1 (Function interval-computable). *A function $f : \Omega \subseteq \mathbb{R}^m \to \mathbb{R}^n$ is said to be* interval-computable *iff there exists an algorithm $\mathcal{I}_f$ that, for every box $B' \subseteq \Omega$ with rational vertices, computes a box $\mathcal{I}_f(B')$ with rational vertices, such that: (i) $f(B') \subseteq \mathcal{I}_f(B')$; and (ii) $\forall \epsilon > 0 : \exists \delta > 0$ such that for every B' having* $\text{width}(B') < \delta$, *then* $\text{width}(\mathcal{I}_f(B')) < \epsilon$.

Given a formula ϕ in m real variables and a box $B \overset{\text{def}}{=} \mathrm{I}_1 \times \cdots \times \mathrm{I}_\mathrm{m} \subset \mathbb{R}^m$ (where $\mathrm{I_i} \overset{\text{def}}{=} [\mathrm{a}_i, \mathrm{b}_i]$), we define the restriction of ϕ to the box B (and say $\phi_{|B}$ is a *bounded formula*) as

$$\phi_{|B} := \phi \wedge \bigwedge_{x_i \in \text{Var}(\phi)} (\mathrm{a}_i \leq x_i \wedge x_i \leq \mathrm{b}_i) \tag{1}$$

2.3 Robustness and quasi-decidability

Intuitively, a formula is robust if its satisfiability status does not change under "small" perturbations[2]. Robustness is a desirable property in many real-world applications, as already observed in the literature (e.g.[25,18]). The related notion of quasi-decidability[16] is then a property that allows to circumvent general undecidability results for a class of formulas when focusing only on robust inputs.

Definition 2 (Quasi-decidability). *A class of problems is quasi-decidable if there exists an algorithm that always terminates on robust instances, and that always returns the right answer when terminating.*

2.4 Topological degree test

The topological degree of a continuous function $f : \mathbb{R}^n \to \mathbb{R}^n$ bounded over a box B is an integer $\deg(f, B)$ that can be defined in several different equivalent ways. Those definitions however require a consistent background, so for lack of space we refer to [24] for a detailed presentation. The property that we are interested in is the following, that we will call *topological degree test*:

Property 1. If $\deg(f, B) \neq 0$, then the equation $f = 0$ has a solution in B.

[2] A formal definition of robustness can be found in Section 2 of [16]

The topological degree has proven to be computable if $0 \notin f(\partial B)$[3][1]. A practical tool for computing it is TopDeg[4], implementing the algorithm described in [15].

3 Local search using Unconstrained Global Optimisation

In this section we explain how to exploit unconstrained global optimisation to help a generic SMT solver to find models for sets of constraints in $\mathcal{NRA}$ and $\mathcal{NTA}$. The general idea is that of mapping a formula ϕ over real variables $x_1, \dots, x_m$ into a real-valued non-negative function $h : \mathbb{R}^m \mapsto \mathbb{R}^{\geq 0}$, such that $\mathbf{x}$ is a model of ϕ only if $h(\mathbf{x}) = 0$, and then use an unconstrained optimisation routine to determine global minima of h. An ad-hoc encoding for Boolean variables should be introduced. This technique, which we shall call *Logic-to-Optimisation*, has already been applied successfully in other theories, e.g. [**?**]. In general, there exist several approaches to perform logic-to-optimisation, that vary depending on which logical theory is considered, what the purpose of the translation is, and which properties of the cost function are desired.

We illustrate the specific translation that we use in our procedure. We assume w.l.o.g. that our input formula consists of conjunctions and disjunctions of Boolean variables $b_1, \dots, b_k$, possibly negated, and constraints of the form $f \bowtie 0$, where $\bowtie\ \in \{<, \leq, =\}$, and f is a $\mathcal{NTA}$ term. We define an operator $\mathcal{L2O}$ that maps a formula to a non-negative real function from $\mathbb{R}^{m+k}$ to $\mathbb{R}^{\geq 0}$ as follows:

$$\begin{array}{ll}
\mathcal{L2O}(f \bowtie 0),\ \bowtie \in \{\leq, =\} & \overset{\text{def}}{=} (\textsf{if } ([f](\mathbf{x}) \bowtie 0) \textsf{ then } 0 \textsf{ else } [f]^2(\mathbf{x})) \\
\mathcal{L2O}(f < 0) & \overset{\text{def}}{=} \mathcal{L2O}(f \leq 0) \\
\mathcal{L2O}(\neg(f \bowtie 0)),\ \bowtie \in \{<, \leq\} & \overset{\text{def}}{=} \mathcal{L2O}(-f \bowtie 0) \\
\mathcal{L2O}(\neg(f = 0)) & \overset{\text{def}}{=} (\textsf{if } ([f](\mathbf{x}) = 0) \textsf{ then } 1 \textsf{ else } 0) \\
\mathcal{L2O}(b) & \overset{\text{def}}{=} \mathcal{L2O}(-x_b \leq 0) \\
\mathcal{L2O}(\neg b) & \overset{\text{def}}{=} \mathcal{L2O}(x_b + 1 \leq 0) \\
\mathcal{L2O}(\phi_1 \wedge \phi_2) & \overset{\text{def}}{=} \mathcal{L2O}(\phi_1) + \mathcal{L2O}(\phi_2) \\
\mathcal{L2O}(\phi_1 \vee \phi_2) & \overset{\text{def}}{=} \mathcal{L2O}(\phi_1) * \mathcal{L2O}(\phi_2),
\end{array}$$

where x_b is a fresh real variable.

Note that with this definition, our logic-to-optimisation transformation will produce an overapproximation, meaning that not all the points in which $\mathcal{L2O}(\phi)$ evaluates to 0 (the *zero set* of $\mathcal{L2O}(\phi)$, denoted Z_ϕ) are models of ϕ: specifically, this is due to the encoding used for strict inequalities and Boolean variables. What is important for our purposes, however, is the converse, i.e. the fact that Z_ϕ contains the set M_ϕ of all the models of ϕ. Moreover, since $\mathcal{L2O}(\phi)$ has non-negative values, if $Z_\phi \neq \emptyset$, then Z_ϕ contains all and only the global minima of the function. We can exploit these facts as follows.

Through the unconstrained global optimisation algorithm Basin-hopping mentioned in §2.1, we obtain a finite set of local minima $L_\phi \subseteq \{\mathbf{x} \in \mathbb{R}^m | \mathbf{x}$ is a local

[3] ∂B is the topological boundary of B, i.e. the set of points in the closure of B that are not in its interior.

[4] Available at `https://www.cs.cas.cz/~ratschan/topdeg/topdeg.html`.

minimum of $\mathcal{L}2\mathcal{O}(\phi)\}$. Implementation-wise, the output will consist of rational approximations of local minima. We denote this set by $\tilde{L}_\phi$. For each element $\tilde{\mathbf{x}} \in \tilde{L}_\phi$, we try to produce a model $\mathbf{x}$ for ϕ. We first propose two simple tactics that work only in the case that ϕ is in $\mathcal{NRA}$, and we will present a more elaborate procedure for $\mathcal{NTA}$ in the next section. Moreover, in the following we only consider formulas which are simply conjunctions of constraints, and that contain no Boolean variables. We shall deal with general formulas in §5.

Given $\tilde{\mathbf{x}} \overset{\text{def}}{=} \{\tilde{\mathrm{x}}_1, \cdots, \tilde{\mathrm{x}}_\mathrm{m}\} \in \tilde{L}_\phi$, it is trivial to check whether $\tilde{\mathbf{x}}$ is a model for ϕ by substituting the variables with their values into the formula. [5] If $\tilde{\mathbf{x}}$ is not a model we can try to look in the surroundings of $\tilde{\mathbf{x}}$. An idea is to reduce ϕ to a linear under-approximation by forcing all the multiplications to be linear, similarly to what is done in [7] equation (3), in the context of the incremental linearization approach (we will refer to this techique as *check-crosses*). A third more general idea is restricting the problem to a bounded subformula $\phi_{|B}$, obtained by imposing that the variables range over a box $B \equiv \mathrm{I}_1 \times \cdots \times \mathrm{I}_\mathrm{m} \subset \mathbb{R}^m$ (where $\mathrm{I}_\mathrm{i} \overset{\text{def}}{=} [\mathrm{a}_\mathrm{i}, \mathrm{b}_\mathrm{i}]$ and $\tilde{\mathrm{x}}_\mathrm{i} \in \mathrm{I}_i$). A naive choice of B is the hyper-cube having $\tilde{\mathbf{x}}$ as its center (that is, $\mathrm{I}_\mathrm{i} \overset{\text{def}}{=} [\tilde{\mathrm{x}}_\mathrm{i} - c, \tilde{\mathrm{x}}_\mathrm{i} + c]$ for a given small $c \in \mathbb{Q}_{>0}$).

The reason to restrict to a box is that bounded problems are, in general, easier to solve, and, if the cost of $\tilde{\mathbf{x}}$ is zero or very close to zero, we can reasonably hope that a model lies in the box. However, restricting to bounded instances by itself does not help much in terms of classes of problems we are able to solve. In fact, if our SMT solver was unable to find irrational models before, it still is. Nonetheless, as we will see in the next section, the idea of finding a point $\tilde{\mathbf{x}}$ very close to being a model and then restrict the problem to a (possibly very tight) bounded instance, allows the adoption of a new procedure for $\mathcal{NTA}$.

4 Solving bounded instances with the topological degree test and interval arithmetic

In this section we explain how, given a local minimum $\tilde{\mathbf{x}}$ obtained as in the previous section, we can prove the satisfiability of a bounded conjunction of constraints $\phi_{|B}$ in $\mathcal{NTA}$ through interval arithmetic and the computation of the topological degree.

First, in §4.1, we provide a practical quasi-decidability procedure for bounded formulas in m variables that contain n equations and k non-strict inequalities, and for which either $n = m$ or $n = 0$. We then generalize this in §4.2, by providing a method that, given a formula with the only condition that $n \leq m$ (and no conditions on the kind of inequalities), can generate subformulas for which the quasi-decidability procedure is applicable. Finally, in §4.3, we discuss how we can integrate these results within the Logic-to-Optimisation framework.

Algorithm 1 Quasi-dec

```
    Input: A bounded formula $\phi_{|B}$ in $m$ variables, $n$ equations $f_1 = 0, \cdots, f_n = 0$,
and $k$ non-strict inequalities $g_i \leq 0, \cdots, g_k \leq 0$ s.t. $n = m$ or $n = 0$
    Output: <False> or <True, $B_{sol}$>            ▷ $B_{sol}$ is a box containing a model
 1: grid ← $\{B\}$
 2: conflict_indices ← $\{0, \cdots, m\}$
 3: while True do
 4:     for $A \in$ grid do
 5:         if $(0 \notin \mathcal{I}_f(A)) \vee (\mathcal{I}_g(A) \cap (-\infty, 0]^k = \emptyset)$ then    ▷ $\mathcal{I}_f$ and $\mathcal{I}_g$ as in def. 1
 6:             grid.remove($A$)
 7:     if grid = {} then return <False>
 8:     if $n \neq 0$ then
 9:         grid ← Merge all the boxes in grid having a common face $C$ s.t. $0 \in \mathcal{I}_f(C)$

10:         grid$_\partial$ ← $\{A \in$ grid $\mid$ exists $C$ a face of $A$ s.t. $C \subseteq \partial B \wedge 0 \in \mathcal{I}_{f(C)}\}$
11:     else
12:         grid$_\partial$ ← {}
13:     for $A \in$ grid $\setminus$ grid$_\partial$ do
14:         conflict_indices_A ← {}
15:         demerge($A$) := $\{E \mid E$ has been merged into $A$ in line 9$\}$
16:         for $E \in$ demerge($A$) do
17:             for $i \in \{0, \cdots, k\}$ do
18:                 if $\mathcal{I}_{g_i}(E) \not\subset (-\infty, 0]$ then
19:                     conflict_indices_A.add($\{j \in \{0, \cdots, m\} \mid x_j$ appears in $g_i\}$)
20:         if conflict_indices_A = {} $\wedge$ ($n = 0 \vee$ TopDeg($f, A$) $\neq 0$) then
21:             return <True , $A$ >
22:         conflict_indices.add(conflict_indices_A)
23:     refinement_index ← Choose an index with the help of conflict_indices
24:     grid ← refine(grid, refinement_index) ▷ First, we demerge the grid; then,
    each sub-box is split in two sub-boxes along the axis refinement_index
25:     conflict_indices ← $\emptyset$
```

4.1 Quasi-decidability procedure

The procedure that we introduce in Algorithm 1 is inspired by that proposed in [16], although some significant changes – discussed at the end of this subsection – have been made to ensure its applicability in practice. We stress that the condition $n = m \vee n = 0$ depends by the fact that the topological degree cannot be defined for $n \neq m$. Using symbolic rewriting tricks (e.g. adding redundant equalities, or rewriting an equality as the conjunction of two non-strict inequalities) to force a robust formula to satisfy the condition would not work, as it would make the formula *non-robust* and so the procedure – albeit applicable – would just not terminate. For the sake of brevity, we introduce the multi-valued functions $f := f_1 \times \cdots \times f_n$, and $g := g_1 \times \cdots \times g_k$.

[5] We remind that we are assuming to be in $\mathcal{NRA}$ only here.

The idea of the algorithm is to iteratively divide the starting box into smaller sub-boxes (the set of which is called a grid), removing at each step from the grid the sub-boxes for which either an equation or an inequality does not hold (lines 4–6), and using the topological degree test to prove if the system of equations admits a solution inside one of the sub-boxes (line 20), provided that the inequalities hold in that sub-box (lines 16-19). The algorithm terminates either returning **True** when a box respecting these last conditions has been found, or returning **False** if the grid is emptied (line 7).

In order to be computable in a box A, the topological degree requires that no zero lies in $f(\partial A)$. Because of that, we have to take some precautions, such as merging boxes having a common face in which a zero lies (line 9) and avoiding boxes having a face contained on the border of B and in which lies a zero (line 10). Regarding the last case we remark that, if the only solution of $\phi_{|B}$ lies in ∂B, then the formula is not robust (and the algorithm is allowed to never terminate).

Another sensitive point is to make sure that, given a robust formula, the algorithm always terminates. To this extent, it is essential that the following property is satisfied: "*for every* $\epsilon > 0$*, there is a finite number of iterations after which each sub-box A in the grid has* $\text{width}(A) < \epsilon$". In order to satisfy this property, a necessary and sufficient condition is that, for each $i \in \{0, \cdots, m\}$, the refinement index assumes infinitely many times the value i. One naive idea would be to assign the refinement index to $i+1$ at each iteration. However, this is not practical. In fact, refining the grid without considering the reasons for which the algorithm does not terminate leads to an unmanageable growth in the size of the grid. Thus we use a greedy approach: at each step we take note of the indexes for which there is a conflict in the inequalities (line 19), and then we base our choice of the refinement index on that (line 24), preferring indices that appear in the conflicts (but making sure that eventually each index is chosen, even though with different frequency). This is a main difference compared to the algorithm from [16], where the grid is divided along all the indices at each step. This results in a double exponential growth in the number of sub-boxes to consider: after i steps, in the worst case the grid will contain $(2^i)^m$ sub-boxes. In our algorithm at each step we choose exactly one index along which to split the sub-boxes, choosing the index that most likely is causing the algorithm not to terminate. Avoiding splitting along indices that are not responsible for conflicts is essential to prevent an explosion in dimension which would make the algorithm impractical. Moreover, to the best of our knowledge, ours is the first implementation of this kind of procedures. In the next subsection, we will further modify the procedure to make it able to produce explanations for unsat cases.

4.2 From a formula with $n \leq m$ to Quasi-dec

Let $\phi_{|B}$ be some bounded formula, with the only condition that $n \leq m$. We define $\hat{\phi}_{|B}$ as the formula obtained from $\phi_{|B}$ by replacing every constraint $e \overset{\text{def}}{=} (g > 0)$

Algorithm 2 Solve a formula $\phi_{|B}$ with $n \leq m$

```
    Input: A formula $\phi_{|B}$ in $m$ variables, $n$ equations, and $k$ inequalities s.t. $n \leq m$
           A candidate point $\tilde{\mathbf{x}} \in \mathbb{R}^m$
    Output: <True, $B_{sol}$> or <Unknown>
 1: $\hat{\phi}_{|B}$ := the formula obtained from $\phi_{|B}$ by replacing every $g > 0$ with $g - \epsilon \geq 0$
 2: if $n = m \vee n = 0$ then
 3:     res_quasidec ← Quasi-dec ($\hat{\phi}_{|B}$)
 4:     if res_quasidec ≡ <True, $B_{sol}$ > then return <True, $B_{sol}$ >
 5:     else return <Unknown>
 6: infeasible_var_subsets ← {}
 7: for vars_subset ∈ Combinations(Vars, $m - n$) do
 8:     if vars_subset ∈ infeasible_var_subsets then
 9:         continue
10:     $\mu := \{var_i \mapsto \mathrm{x_i} \mid var_i \in$ vars_subset$\}$
11:     <sat, p> ← Quasi-dec ($\mu(\hat{\phi}_{|B})$)
12:     if <sat, p> = <True, $B_{sol}$ > then
13:         return <True, $B_{sol}$ >
14:     else if <sat, p> = <False, $\mu(h)$ > then
15:         conflict_vars := $\{var_i \in vars\_subset \mid var_i$ appears in $h\}$
16:         infeasible_var_subsets.add(conflict_vars)
17: return <Unknown>
```

with the constraint $\hat{e} \overset{\text{def}}{=} (g - \epsilon \geq 0)$, given a predefined constant $\epsilon > 0$. It is straightforward to prove that every model of $\hat{\phi}_{|B}$ is also a model of $\phi_{|B}$.

If $n = m$ we can directly apply Quasi-dec. If $n < m$, then we can try to assign $m - n$ variables to real values, and then apply Quasi-dec to the formula obtained from the substitution. We start from a given point $\tilde{\mathbf{x}} \in \mathbb{R}^m$, and enumerate possible assignments to $m - n$ variables. In general, there are $\binom{m}{n} = \frac{m!}{n!(m-n)!}$ possible combinations to explore, but we can reduce their number via conflict-driven learning, as commonly done in SAT and SMT, by modifying the Quasi-dec procedure (Algorithm 1), to make it return, before the while cycle in line 3, **<False** ,f_i**>** if $0 \notin \mathcal{I}_{f_i}(B)$, and **<False** , g_j**>** if $\mathcal{I}_{g_j}(A) \cap (-\infty, 0] = \emptyset$. This modification helps the procedure by explaining why the problem is unsatisfiable, even though only for simple cases where no grid refinement is required. Given an explanation, we can extract the set E of variables involved, and use them to avoid the enumeration of assignments to supersets of E. In general, we could extend the idea of returning explanations for unsatisfiable instances to more complex situations. In this paper, we do not delve into this path, and leave further investigations for future work.

Overall, our approach to reduce to the Quasi-dec procedure given a formula with the only restriction that $n \leq m$ is illustrated in Algorithm 2.

4.3 A general procedure

We can now combine the results of the last two sections. First, we obtain several local minima $\tilde{\mathbf{x}}_\mathbf{1}, \cdots, \tilde{\mathbf{x}}_\mathbf{k}$ as in Section 3 . The two tactics described in the section

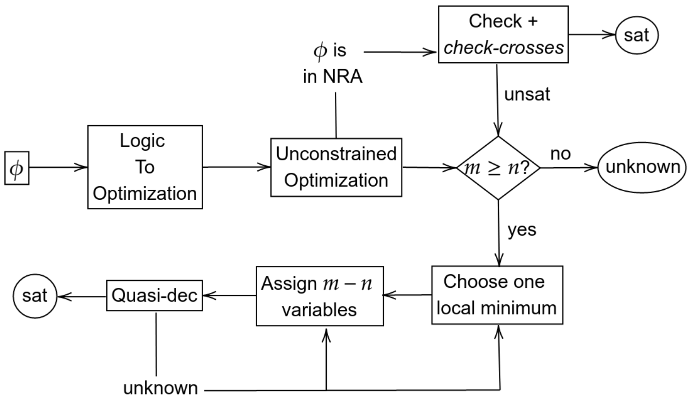

Fig. 1. Schema of the overall procedure.

(i.e., the simple check of $\tilde{\mathbf{x}}$, and the reduction to a linear underapproximation) are reasonably inexpensive for $\mathcal{NRA}$. Thus, if the problem is in $\mathcal{NRA}$, we first apply these two tactics to each local minimum. If these two tactics fail, or the problem is in $\mathcal{NTA}$, we apply the algorithm described in section 4.2 to each local minimum (starting from the minimum with the lowest cost). A sketch of this procedure is shown in Fig. 1.

Remark 1. Our general procedure is not a quasi-decidability procedure. However, relying on a quasi-decidability subprocedure is a crucial point of our method. By construction, the formulas that we feed to Quasi-dec have the property that if they are unsatisfiable then they are also robust (Lemma 3 from [16]).[6] This means that Quasi-dec always terminates on unsatisfiable subformulas, guaranteeing that we always progress towards a solution.

5 From constraint sets to formulas

So far we have considered only sets of constraints. In this section, we present two different ways to solve a formula ϕ with arbitrary Boolean structure and that includes also Boolean variables. In the first way, we apply $\mathcal{L}2\mathcal{O}$ *eagerly* to the formula, and then try to decide the disjunctions through the insight given by a local minimum, and then proceed to solve the constraints set as in §4. In the second way, we use the procedure of §4 as a theory solver inside a DPLL(T)-based lazy-SMT algorithm.

[6] This is not true for formulas containing strict inequalities, but we replaced strict inequalities in Algorithm 2 at line 1.

5.1 An eager approach

Let ϕ be a formula in CNF form. We can apply $\mathcal{L}2\mathcal{O}$ to ϕ and obtain several local minima. Given a local minimum $\tilde{\mathbf{x}}$, if there are no transcendental functions, we can use the two tactics discussed in section 3 (the simple check, and the call to an LRA-solver for a linear underapproximation of ϕ – i.e. check-crosses). We cannot directly apply the tactic discussed in section 4, since Quasi-dec does not work with disjunctions.

In order to apply it, we can try to decide the disjunctions using $\tilde{\mathbf{x}}$, to obtain an implicant of ϕ which we can then feed to Algorithm 2. In line 1, we obtained a formula $\mu(\hat{\phi}|_B)$ that, except for containing disjunctions, is in the form required by Quasi-dec. In fact, $\mu(\hat{\phi}|_B)$ has the form $\bigwedge C_j$, where $C_j \equiv \bigvee_{i \in I_j}(h_i \bowtie 0)$. If we substitute each C_j with one of the atomic formulas that appear in it, then we reduce to the case discussed in the previous section. Hence, for each C_j, we choose one constraint $h_i \bowtie 0$ that most likely is satisfied by $\tilde{\mathbf{x}}$. As a last resort, if for each local minimum this tactic do not work, we rewrite the original formula into DNF and try to solve each constraints set as in the previous section.

5.2 A lazy approach

In the lazy approach, the method defined in §4 is used as a theory solver inside a DPLL(T) procedure. Since our method is able to prove only satisfiability, it needs to be paired with a method able to prove unsatisfiability. In our implementation, we pair it with incremental linearization [7], which is usually effective in proving unsatisfiability, and also good in finding linear models, but whose weak spot is finding irrational models.

Currently, our implementation is quite simple. Inside the DPLL(T) algorithm, we introduce a parameter `n_calls_IncrLin` that keeps track of the calls to incremental linearization, and that is reset whenever the DPLL(T) solver backtracks. After a given k number of calls to incremental linearization, we call our method. If it returns sat, we are done. Otherwise it returns unknown, and we proceed with incremental linearization.

6 Experimental evaluation

Implementation We have implemented our method in a prototype written in Python, called ugotNL (as in *Unconstrained Global Optimisation and Topological degree for Non-Linear*). We refer to the version based on the eager approach as ugotNL$_{\text{EAGER}}$. For the lazy approach, we integrated ugotNL as a theory solver inside the MathSAT SMT solver [8]. We will refer to this version as MathSAT+ugotNL.
Setup. We have run our experiments[7] on a cluster equipped with 2.4GHz Intel Xeon E5-2440 machines, using a time limit of 1000 seconds and a memory limit of 9 Gb. We compared our tools with z3 [11] and Yices[13] (CAD-based), raSAT[27]

[7] Available at `https://drive.google.com/file/d/1f6RmvojKw4om0L08g3hYMBl-wb6nvEpf/`

	Total	Sturm-MGC	ezsmt	Sturm-MBO	zankl	UltimateAut	Economics-M	meti-tarski	Heizmann	hycomp	kissing	LassoRanker
MathSAT	3193	0	**32**	0	32	31	85	2718	3	11	18	263
ugotNL$_{\text{EAGER}}$	4388	0	0	**1**	52	32	68	4042	0	0	**36**	157
MathSAT+ugotNL	4441	0	**32**	0	63	27	84	3948	3	13	35	236
raSAT	4285	0	0	0	45	0	0	4225	0	0	15	0
Yices	4946	0	**32**	0	58	39	91	4369	0	227	10	120
cvc5	5108	0	**32**	0	63	36	89	4342	2	226	18	**300**
z3	**5153**	**2**	30	0	**72**	**47**	**93**	**4391**	**6**	**280**	35	198
dReal	/(5021)	/(9)	/(0)	/(274)	/(153)	/(55)	/(126)	/(4079)	/(51)	/(45)	/(19)	/(210)

Fig. 2. Summary of results for SMT(NRA) sat cases. The results in parenthesis indicate "MAYBE SAT" answers.

	Total	dreal	bmc
MathSAT	94	37	57
ugotNL$_{\text{EAGER}}$	**304**	**255**	49
MathSAT+ugotNL	170	70	**100**
cvc5	94	40	54
iSAT3	/	/	/
dReal	/ (578)	/ (423)	/ (155)

	Total	dreal	bmc
MathSAT	70	21	49
ugotNL$_{\text{EAGER}}$	**253**	**203**	50
MathSAT+ugotNL	140	35	**105**
cvc5	63	17	46
iSAT3	38 (828)	7 (599)	31 (229)
dReal	/ (137)	/ (36)	/ (101)

Fig. 3. Summary of results for SMT(NTA) sat cases. On the left the original instances; on the right the bounded instances.The results in parenthesis indicate "MAYBE SAT" answers.

(that combines ICP and GIVT), CVC5[6] (that combines IL with cylindrical algebraic coverings[30]), iSAT3[17] (based on ICP), and dReal[19] (that operates in the δ-sat framework[18]). Only the last three solvers can deal with $\mathcal{NTA}$.

We checked that, when terminating, our tools always return the correct result when the status of the benchmark is known, and never disagree with the other solvers; for $\mathcal{NRA}$, we checked with z3 that every box returned by our tools contains indeed a model.

Benchmarks. For $\mathcal{NRA}$, we consider all the SMT-LIB [2] benchmarks from the QF-NRA category. This is a class of 11523 benchmarks, among which 5142 are satisfiable, 5379 are unsatisfiable, and 1002 have unknown status. For $\mathcal{NTA}$, we considered the benchmarks from the dReal distribution [19], and other benchmarks deriving from discretization of Bounded Model Checking of hybrid automata. The problems in these classes come all with an unknown status. Since iSAT3 is not able to work with unbounded instances, in order to include it in the comparison, we generated for each benchmark a bounded version by adding constraints that force all the real variables in the problem to assume values in the $[-300, 300]$ interval.

Results (sat). First, we analyze the results for satisfiable instances, which are reported in Figures 2 and 3. The tables show, for each solver, the number of

	Total	Sturm-MGC	ezsmt	Sturm-MBO	zankl	UltimateAut	Economics-M	meti-tarski	Heizmann	hycomp	kissing	LassoRanker	hong
MathSAT	5280	0	**2**	**285**	**34**	7	11	2251	1	**2259**	0	412	**20**
MathSAT+ugotNL	5043	0	2	163	32	5	11	2239	0	2253	0	323	**20**
raSAT	4094	0	0	2	14	0	0	2018	0	1950	0	0	**20**
Yices	5449	0	**2**	**285**	32	**12**	**39**	**2587**	**12**	2201	0	259	**20**
cvc5	**5645**	0	**2**	**285**	31	10	35	2581	7	2206	0	**468**	**20**
z3	5281	**5**	**2**	153	27	**12**	19	2578	3	2225	0	248	9
dReal	3889	0	0	131	4	0	3	1784	0	1946	**1**	0	**20**

Fig. 4. Summary of results for SMT(NRA) unsat cases.

	Total	dreal	bmc
MathSAT	**533**	85	**448**
MathSAT+ugotNL	522	85	437
cvc5	453	75	378
iSAT3	/	/	/
dReal	468	**184**	284

	Total	dreal	bmc
MathSAT	**524**	88	**436**
MathSAT+ugotNL	521	88	433
cvc5	465	85	380
iSAT3	449	63	386
dReal	446	**156**	290

Fig. 5. Summary of results for SMT(NTA) unsat cases. On the left the original instances; on the right the bounded instances.

successfully solved instances, both overall (1st column) and for each benchmark family (rest of the columns), with the best results highlighted in boldface.

For $\mathcal{NRA}$ (Figure 2), we see that z3 is overall superior. Nevertheless, we see that both our new tools are very competitive, and perform significantly better than MathSAT. Moreover, since we are comparing new-born ideas implemented in a prototype with well-optimised CAD-based techniques that have a decade of progresses on their shoulders, we believe that these results are very encouraging.

Where our methods shine and go beyond the state of the art is when we consider problems with transcendental functions. In the results for $\mathcal{NTA}$ (Figure 3) we see that both our tools outperform the others. For this, the synergy between numerical optimization and the procedure based on the topological degree test is essential, as neither of the two methods in isolation is effective: when disabling either of the two components, in fact, the performance is similar to that of the "stock" version of MathSAT (we omit the details due to lack of space). Moreover, there is a great complementarity between the two tools. For families in which the Boolean component is huge (such as bmc) we see that MathSAT+ugotNL is by far the best, whereas for benchmarks where the theory component is predominant (e.g. the dreal ones) the situation is reversed.

Results (unsat). Now we analyze the results for unsatisfiable cases. Our methods are designed to finding models, so, for unsatisfiable instances, there are no advancements whatsoever. Nevertheless, we are interested in evaluating possible losses of the lazy version wrt. MathSAT due to the integration of our method. (The eager approach can never return unsat, so it does not compete.)

We see that for $\mathcal{NRA}$ there are some losses (expecially for very time-consuming benchmark families such as LassoRanker and MBO), that overall count for 4.5% of the benchmarks that MathSAT is able to solve before the timeout. For $\mathcal{NTA}$, we observed that the losses are even less: respectively 2.1% and 0.6% for unbounded and bounded instances. We remark that these results do not imply that our new tool is unable to prove the unsatisfiability for those cases, rather that it is unable to prove it *within the same timeout.* In fact, since our theory solver always terminates for unsatisfiable instances (see Remark 1), we know that, if MathSAT returns unsat for a problem, then eventually MathSAT+ugotNL will return unsat as well.

We stress the fact that our implementation is currently still a research prototype, implemented in Python and integrated within MathSAT in a quite inefficient manner, introducing a lot of overhead in the interaction with the DPLL(T) solver. We are confident that a more optimised and better integrated implementation can significantly reduce the overhead and improve the situation for unsatisfaible instances. Therefore, we believe that these results prove that our tool, albeit aimed specifically at proving satisfiability, works well even for unsatisfiable instances, and, in particular for $\mathcal{NTA}$ (which is our privileged theory of interest), there are no relevant downsides in pairing our sat-oriented theory solver with an unsat-oriented theory solver based on incremental linearization.

7 Conclusions and Future Work

In this paper we proposed a new procedure for proving satisfiability in $\mathcal{NTA}$, based on a fruitful synergy of numerical and symbolic methods. We implemented our ideas in a prototype called ugotNL, and proposed two different approaches: an eager one and a lazy one (integrated inside MathSAT). We tested the two methods on a wide variety of satisfiable benchmarks, and the results demonstrated that both our methods significantly outperform the state of the art for $\mathcal{NTA}$, while being competitive for $\mathcal{NRA}$. In the future, we plan to better integrate ugotNL inside MathSAT and to experiment with more thoughtful heuristics. Furthermore, we plan to investigate the potential of our ideas in several directions, including how to exploit the procedure also for proving unsatisfiability and whether similar techniques can be applied also to solve problems involving differential equations.

References

1. O. Aberth. Computation of topological degree using interval arithmetic, and applications. *Mathematics of Computation*, 1994.
2. C. Barrett, P. Fontaine, and C. Tinelli. The Satisfiability Modulo Theories Library (SMT-LIB). `www.SMT-LIB.org`, 2016.
3. C. Barrett, R. Sebastiani, S. A. Seshia, and C. Tinelli. *Satisfiability Modulo Theories*, chapter 26. 2009.
4. F. Benhamou and L. Granvilliers. Chapter 16 - continuous and interval constraints. In *Handbook of Constraint Programming.* 2006.

5. F. Brauße, K. Korovin, M. V. Korovina, and N. T. Müller. The ksmt calculus is a δ-complete decision procedure for non-linear constraints. In *CADE*, 2021.
6. C.Barrett et al. CVC5 at the SMT Competition 2021, 2021.
7. A. Cimatti, A. Griggio, A. Irfan, M. Roveri, and R. Sebastiani. Incremental linearization for satisfiability and verification modulo nonlinear arithmetic and transcendental functions. *ACM Trans. Comput. Logic*, (3), 2018.
8. A. Cimatti, A. Griggio, B. Schaafsma, and R. Sebastiani. The MathSAT5 SMT Solver. In *Proceedings of TACAS*, 2013.
9. G. E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decompostion. In *Automata Theory and Formal Languages*, 1975.
10. F. Corzilius, G. Kremer, S. Junges, S. Schupp, and E. Ábrahám. Smt-rat: An open source c++ toolbox for strategic and parallel smt solving. In *SAT*, 2015.
11. L. De Moura and N. Bjørner. Z3: An efficient smt solver. In *TACAS*, 2008.
12. L. M. de Moura and D. Jovanovic. A model-constructing satisfiability calculus. In *VMCAI*, 2013.
13. B. Dutertre. Yices 2.2. In *Computer-Aided Verification (CAV'2014)*, 2014.
14. M. Fischer, M. Balunovic, D. Drachsler-Cohen, T. Gehr, C. Zhang, and M. T. Vechev. Dl2: Training and querying neural networks with logic. In *ICML*, 2019.
15. P. Franek and S. Ratschan. Effective topological degree computation based on interval arithmetic. *Mathematics of Computation*, (293), 2015.
16. P. Franek, S. Ratschan, and P. Zgliczynski. Quasi-decidability of a fragment of the first-order theory of real numbers. *J. Autom. Reason.*, (2), 2016.
17. M. Fränzle, C. Herde, T. Teige, S. Ratschan, and T. Schubert. Efficient solving of large non-linear arithmetic constraint systems with complex boolean structure. *JSAT*, 2007.
18. S. Gao, J. Avigad, and E. M. Clarke. δ-complete decision procedures for satisfiability over the reals. In *IJCAR*, 2012.
19. S. Gao, S. Kong, and E. M. Clarke. dreal: An smt solver for nonlinear theories over the reals. In *Automated Deduction – CADE-24*, 2013.
20. D. Jovanović and L. de Moura. Solving non-linear arithmetic. *ACM Commun. Comput. Algebra*, (3/4), 2013.
21. D. D. L. Minh and D. L. P. Minh. Understanding the hastings algorithm. *Communications in Statistics - Simulation and Computation*, (2), 2015.
22. R. E. Moore, R. B. Kearfott, and M. J. Cloud. *Introduction to Interval Analysis*. 2009.
23. A. Neumaier. *Interval Methods for Systems of Equations*. Cambridge University Press, 1991.
24. D. O'Regan, C. Yeol Je, and Y. Chen. *Topological Degree Theory and Applications*. Taylor and Francis, 2006.
25. S. Ratschan. Safety verification of non-linear hybrid systems is quasi-decidable. *Form. Methods Syst. Des.*, (1), 2014.
26. D. Richardson. Some undecidable problems involving elementary functions of a real variable. *J. Symb. Log.*, (4), 1968.
27. T. Vu Xuan, T. Khanh, and M. Ogawa. rasat: an smt solver for polynomial constraints. *Formal Methods in System Design*, 51, 12 2017.
28. D. J. Wales and J. P. K. Doye. Global optimization by basin-hopping and the lowest energy structures of lennard-jones clusters containing up to 110 atoms. *The Journal of Physical Chemistry A*, (28), 1997.
29. L. Zhang, C. F. Madigan, M. H. Moskewicz, and S. Malik. Efficient conflict driven learning in a boolean satisfiability solver. In *ICCAD*, 2001.

30. E. Ábrahám, J. 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, 2020.