UNIVERSITY OF GENOVA

PHD PROGRAM IN SECURITY, RISK AND VULNERABILITY
(CURRICULUM: CYBERSECURITY AND RELIABLE AI)
XXXVI CYCLE

Satisfiability modulo Nonlinear Arithmetic and Transcendental Functions via Numerical and Topological methods

by

Enrico Lipparini

November 2024

Supervisors Stefan Ratschan, *Czech Academy of Sciences*
Alessandro Armando, *University of Genova*

Ph.D. Coordinator Alessandro Armando, *University of Genova*
Ph.D. Curriculum Coordinator Luca Oneto, *University of Genova*

External Reviewers Erika Ábrahám, *RWTH Aachen University*
Cesare Tinelli, *University of Iowa*

Dibris

Department of Informatics, Bioengineering, Robotics and Systems Engineering

We must know, we will know (I mean... maybe... if the theory is decidable... or, well, at least for some cases...)

- David Hilbert, revised

Acknowledgements

"*This thesis would not have been possible without the contribution of many people*" is the opening sentence of plenty of theses — the present one entering the group right now — and for good reason. Every personal accomplishment is the fruit of several factors, which are often hard to estimate precisely. Here, I will try my best to acknowledge, give credit to, and thank all the people who have played a role in these factors.

The first *thank you* goes to Stefan Ratschan. It is not rhetorical to say that this thesis would not have been possible without you. Not only you are one of the most knowledgeable persons on the topic, and you have been amazing at transmitting this knowledge and mentoring me to be a better researcher, but you have also done so with such enthusiasm and positivity that it greatly motivated me. Above all, I thank you for having taught me that doing research should be, first and foremost, a pleasure and a joy.

A deep thanks goes to Alessandro Armando, whose continuous support, especially in difficult moments, has been an essential ingredient for the successful completion of this journey. I could not have asked for a more helpful tutor and kinder supervisor.

A meaningful contribution has also been given by Alberto Griggio and Alessandro Cimatti from FBK, and by Roberto Sebastiani from the University of Trento. I have started working on the topic of this thesis with them, and, together, we have published what has been my first paper, which has marked the way for the rest of the work presented in this thesis. Working with such great researchers has really been a great opportunity.

During my internship at SRI International, I had the great pleasure to work with Ahmed Irfan and Stéphane Graham-Lengrand. Ahmed's works have been the firsts I read while performing the literature review during my first year, and being able to work side by side with him towards the end of this journey has felt like the perfect conclusion, he is a great teacher and an excellent researcher. Stéphane not only has always been very insightful and helpful, but he also has taken care of making me feel like at home during the months I spent in California (not by chance, more than one person told us he seems my elder brother).

A particular thanks goes to the external reviewers of this thesis, Erika Ábrahám and Cesare Tinelli, for taking the time to thoroughly read the thesis and provide me with valuable feedback. Receiving a positive evaluation from researchers of such caliber has made me very proud of this work.

Among others, I'd like to thank: all the nice people I met in Genova, Trento, Prague, and Menlo Park during these years, as well as the ones I met at conferences here and there and I had fun with; the administrative staff of the University of Genova; Luca Oneto (for being the fastest e-mail replier of Genova, possibly of the world); Angelo Ferrando, Massimo Bartoletti, and Vadim Malvone (for broadening my research horizons); Silvio Ghilardi (without whom this journey would have never started); Laura (for teaching me that it is the simple thing that's hard to do). To all the people who have enriched my life during these years, and that continue to do so, my gratitude goes beyond this space. A necessary exception has to be made for my parents: ultimately, among the factors that enabled me to pursue this educational and professional path, and that allowed me to do so, above all, under the signs of curiosity and creativity, they have been, by far, the most important ones.

As more of a final consideration, rather than a conventional acknowledgment: during these years, and especially during the conclusive phase of the thesis writing, I have often reflected on how all the contributions in this thesis — theoretical results as well as practical methods — have only been possible due to the prior work made by the countless number of researchers that had laid the enormous groundwork this thesis relies on. This intrinsic cooperative aspect of science is something that still amazes me, and the greatest satisfaction I could ever receive from this work is if in the future it will become a small part of the enormous groundwork someone else's work will rely on.

I'd like to conclude this acknowledgment section with a quote from German psychologist Kurt Lewin that has often come to my mind whenever I was getting upsetting results from experiments and started worrying about the practicability of my ideas (a feeling probably well known to most of my colleagues), and which gave me the strength to persevere: *"There is nothing more practical than a good theory"*.

Abstract

Satisfiability Modulo Theories (SMT) is the problem of deciding the satisfiability of a first-order formula with respect to some background theory. In this thesis, we focus on the theory of non-linear real arithmetic augmented with exponential and trigonometric functions, also known as Non-linear Transcendental Arithmetic ($\mathcal{NTA}$). $\mathcal{NTA}$ is a very expressive theory, which can be used to model problems in different domains, such as physics, engineering, economics, and biology.

Unfortunately, solving problems in $\mathcal{NTA}$ is very challenging: indeed, the problem is undecidable, there is no known finite representation of satisfying assignments, and symbolic methods struggle to reason precisely about transcendental functions. State-of-the-art SMT solvers usually rely on over-approximations of transcendental functions, and are thus unable to prove the existence of a model, except that in very simple cases.

We propose a novel approach to SMT($\mathcal{NTA}$) that leverages techniques coming from the fields of numerical analysis and topology, and integrates them within the SMT context. Numerical methods are used to quickly obtain approximate candidate solutions, which can be used to reduce the search of a model to simpler local sub-problems. Topological methods are used to prove the existence of a solution in a bounded region without the need to explicitly express such a solution, thus circumventing the issue of representing satisfying assignments.

We implement our procedure in the prototype tool UGOTNL, and experimentally evaluate the tool performances over a wide range of $\mathcal{NTA}$ benchmarks. The experiments show that the tool significantly outperforms other state-of-the-art SMT solvers on satisfiable $\mathcal{NTA}$ problems, being able to solve more than four times the benchmarks solved by the best of the other tools.

Furthermore, we characterize the class of problems for which our method can terminate successfully. Based on that, we contribute with an original theorem that proves that a weaker notion of decidability holds for bounded $\mathcal{NTA}$ formulas, namely that there exists an algorithm that always terminates successfully under certain assumptions on the robustness of the problem.

Finally, we outline how our idea of leveraging fast numerical techniques to accelerate the search for a model can fit into a more general framework such as the model-constructing satisfiability calculus (MCSAT), and how it can be applied to other theories, such as non-linear integer arithmetic ($\mathcal{NIA}$).

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Introduction

In the last decades, hardware and software systems have become increasingly widespread in our lives, and more and more safety-critical applications rely on them. Failures of such systems can have catastrophic consequences, such as causing human deaths, or huge economic losses. Therefore, ensuring the correctness of these systems has become a fundamental challenge in computer science. While extensive testing can serve as a first line of defense against these malfunctions, it can neither guarantee the detection of all possible subtle bugs, nor prove whether certain specifications are met. In order to do so, a more exhausting and comprehensive approach to the problem is required. Formal verification, a rigorous approach that relies on mathematical logic, emerged as an ideal framework for the task. Logical formulas are used to express states and transformations of the system, as well as the properties that we expect the system to satisfy. Then, fully automatic methods are used to either produce a witness that shows that the desired property is violated, or to produce a formal proof that demonstrates that the property holds.

One of the most popular paradigms to reason about logical formulas is *Satisfiability Modulo Theories* (SMT), which generalizes the Boolean satisfiability problem (SAT) to richer theories, such as linear and non-linear arithmetic, floating points, bit-vectors, arrays, uninterpreted functions, and many others. SMT solvers leverage the integration of SAT solvers with theory-specific decision procedures (also called *theory solvers*).

Huge progress has been made by the SMT community to develop efficient and scalable decision procedures for a wide variety of theories. For certain theories, such as *Linear*

Real Arithmetic ($\mathcal{LRA}$), modern SMT solvers are already extremely efficient, and their use has been widely adopted to tackle real-world problems. Even for more complex theories, such as *Non-linear Real Arithmetic* ($\mathcal{NRA}$), whose worst-case complexity is known to be doubly-exponential [114, 35] and that only two decades ago was thought to be untreatable in practice, now state-of-the-art SMT solvers are often able to solve even complex problems in a reasonable amount of time.

However, if we extend $\mathcal{NRA}$ by including transcendental functions such as exponential (e.g. exp, log) and trigonometric functions (e.g., sin, arctan), then the resulting theory, *Non-linear Transcendental Arithmetic* ($\mathcal{NTA}$), becomes undecidable [94].

Since modeling many problems in domains such as physics, engineering, economics, and biology, requires the expressiveness of transcendental functions, then, research on SMT($\mathcal{NTA}$) is of great importance [92]. However, given the theoretical and practical complexity of the theory, there are still many problems that are out of the scope of state-of-the-art SMT solvers.

SMT($\mathcal{NTA}$) can be tackled from different angles. In general, symbolic reasoning over transcendental functions tends to be quite burdensome. Currently, most SMT solvers try to avoid reasoning directly about transcendental functions, recurring to either linearization or interval-based over-approximations. These approaches work particularly well for proving that a formula does not have a solution, since if an over-approximation of the original formula is unsatisfiable, then the original formula is unsatisfiable, too. However, except for very simple cases, this approach is doomed to fail in proving that a formula does have a solution. One major hurdle is that, contrarily to $\mathcal{LRA}$, where satisfying assignments consist of fractions, or $\mathcal{NRA}$, where satisfying assignments consist of polynomial roots, there is no known finite representation for $\mathcal{NTA}$ satisfying assignments. Hence, even just *expressing* a solution is a non-trivial problem.

In this thesis, we will discuss how techniques from the fields of numerical analysis and topology can be leveraged and integrated in the context of SMT to tackle formulas with polynomials and transcendental functions, and how their adoption significantly advances the state-of-the-art for the satisfiability case of $\mathcal{NTA}$ formulas. We will also discuss how these techniques can play a role for the related theories of $\mathcal{NRA}$ and $\mathcal{NIA}$ (*Non-linear Integer Arithmetic*). Moreover, we will theoretically characterize certain relevant subclasses of $\mathcal{NTA}$ for which, despite the general undecidability of the theory, there exists a procedure that always terminates successfully, under certain robustness assumptions.

1.2 Related work

Here, we present various works that tackle problems involving nonlinear arithmetic. Since the focus of the thesis is on $\mathcal{NTA}$, we will devote most of the space to techniques that can deal with transcendental functions, but we will also discuss approaches that only work for $\mathcal{NRA}$ and $\mathcal{NIA}$ formulas.

1.2.1 Interval Contraint Propagation (ICP)

Interval Constraint Propagation (ICP) [12, 91] is an incomplete technique based on Interval Arithmetic [84]. In this approach, interval arithmetic is used to evaluate a set of constraints over boxes (i.e. Cartesian products of intervals), contract such boxes into smaller ones without the lost of information, and then, if no further contraction is possible and the problem has not yet been solved, decompose the boxes into smaller ones and reiterate the process for each of these new boxes.

ICP is a very flexible procedure, as it can be applied to every function for which it is possible to compute infinitely precise bounds (this class includes $\mathcal{NTA}$ functions). ICP can effectively prove unsatisfiability of general formulas, and satisfiability of formulas that only include inequalities. However, being based on interval arithmetic, which works by computing over-approximations, in the presence of equalities this technique is, in general, not able to prove satisfiability, as it would require the capability of expressing real values precisely as singleton intervals, or as uniquely determined compositions of previously found singleton intervals (this is possible only in the special case of *strongly satisfiable* equations [50, 42]).

The first solvers to implement ICP have been REALPAVER [56] and RSOLVER [91]. Currently, ICP is implemented by ISAT3 [50] and DREAL [54] for $\mathcal{NTA}$, and by RASAT [110] and SMT-RAT [99] for $\mathcal{NRA}$. Interestingly, in order to overcome the weakness of ICP to prove satisfiability, RASAT combines ICP with the Generalized Intermediate Value Theorem [86]. SPASS(ISAT) [42], on the other hand, leverages the SPASS theorem prover [113] to implement a *strong satisfaction check* on candidate solutions returned by ISAT2 that works for strongly satisfiable formulas. A different approach is taken by DREAL, which relaxes the notion of decidability to that of δ-decidability (see Subsection 1.2.3). In the context of theorem proving, several tools have leveraged interval propagation to handle transcendental functions [36, 80, 102, 74, 75].

1.2.2 Linearization

The idea of the linearization approach is to avoid dealing directly with non-linearity by reducing to a linear problem, and lazily linearize non-linear functions on the base of a lemma-on-demand approach.

Incremental Linearization (IL) [61, 30] leverages efficient complete decision procedures for linear arithmetic by abstracting non-linear multiplication and transcendental functions symbols with uninterpreted functions. Then, linear solvers are used to either prove the unsatisfiability of the abstracted formula, from which the unsatifiability of the original formula follows, or find a rational model, which can either be an actual model or a *spurious* model, in which case nonlinear terms are incrementally axiomatized by piecewise-linear constraints. IL is particularly successful in proving unsatisfiability, and can also prove satisfiability when the model is rational. However, since linear solvers are not able to find non-rational models, in most of the cases that involve equalities it is possible to prove satisfiability only indirectly, by showing that a given variable assignment satisfies the formula *for all possible interpretations* of the involved transcendental functions within some bounds [28]. This technique, however, only works for very simple cases. IL has been implemented in MATHSAT for $\mathcal{NTA}$ [28] and $\mathcal{NIA}$ [29], in CVC5 for $\mathcal{NTA}$ [67], in SMT-RAT for $\mathcal{NRA}$ [116, 103] and $\mathcal{NIA}$ [98], in Z3 for $\mathcal{NIA}$ [13], and in SWINE for $\mathcal{EIA}$ (*Exponential Integer Arithmetic*) [49].

Another linearization approach is the ksmt-calculus [17], a model-guided CDCL(T)-style procedure that applies gradual linear approximations to non-linear constraints. Differently from incremental linearization, where non-linear constraints are abstracted to uninterpreted functions and then refined through incremental axiomatization, here, non-linear constraints are directly linearized on-demand by tangent spaces, safely computed by numerical means. This approach works for the class of *functions with decidable rational approximations* (which include polynomials and transcendental functions), but only for problems with no equations. The method has been implemented in the KSMT tool for a subclass of $\mathcal{NRA}$.

For transcendental functions, linearization can be seen as a special case of using Taylor approximations to find polynomial upper and lower bounds. This approach is used by METITARSKI [4] to prove unsatisfiability of $\mathcal{NTA}$ formulas via reduction to $\mathcal{NRA}$.

1.2.3 δ-satisfiability

In order to overcome the difficulties in proving satisfiability in the presence of transcendental functions, a different approach that relaxes the notion of decidability has been proposed.

In this approach, called δ-satisfiability [53], an algorithm can return either `unsat` (if the problem is unsatisfiable) or δ-`sat`, if there exists a perturbation of the original formula (up to some $\delta > 0$ specified by the user) that is satisfiable. If a formula is δ-satisfiable, there is no guarantee that it is also satisfiable: indeed, a formula can be *at the same time* unsatisfiable and δ-satisfiable. The SMT solver DREAL [54] implements a δ-decision procedure based on ICP. It has been proved that the KSMT-calculus is a δ-complete decision procedure, too [18].

1.2.4 $\mathcal{NRA}$-specific

Here, we give an overview on methods that work specifically for SMT($\mathcal{NRA}$).

The most successful method is surely *Cylindric Algebraic Decomposition* (CAD). First introduced by Collins [32], it is a quantifier elimination procedure that decomposes the real space into cells, and then iteratively projects on one of the coordinates, reducing the satisfiability of a formula to the satisfiability of sub-formulas of lower dimension. Although its complexity is double exponential w.r.t. the number of variables [114, 35], it is a very efficient technique when used within the *model-constructing satisfiability calculus* (MCSAT) framework [64, 38]. Here, instead of eagerly running CAD on the whole formula, CAD is lazily called only on polynomials in one variable to explain conflicts.

Virtual Substitution [115] is based on the well-known formulae for expressing the solutions of polynomials in terms of radicals. This method can be practically very efficient for polynomials with low degree, but its applicability is limited due to the Abel-Ruffini theorem [96], which states that it is not possible to solve by radicals general polynomials of degree 5 or higher with arbitrary coefficients.

Subtropical Satisfiability [108, 44, 109] is an incomplete method able to prove the satisfiability of a single polynomial equation or of a sets of polynomial inequalities. Starting from a given point, it uses information on the coefficients and exponents of the monomials to guess the direction of the polynomial, and then reduces to linear solving to find a solution. This method is particularly efficient for finding roots of extremely large polynomials (such problems arise in some biological applications [108]).

1.2.5 $\mathcal{NIA}$-specific

In the *branch-and-bound* approach [66, 63], a $\mathcal{NIA}$ formula is relaxed by allowing its variables to range over the reals. If the relaxed formula has no real solution, then no integer

solution exists either. Otherwise, if the relaxed formula is satisfiable, then either the solution is integer-valued (in which case, it is also a solution for the original $\mathcal{NIA}$ formula), or the solution contains at least one non-integer value, in which case new constraints are added to exclude non-integer values between the floor and the ceiling of such value.

Another major approach is *bit blasting* [52], in which bounds are iteratively imposed over the variables of the formula to reduce the $\mathcal{NIA}$ formula to a bounded (hence decidable) sub-formula that can then be encoded into a SAT problem. This approach can be quite successful for finding models, especially when the values are relatively small, but cannot prove unsatisfiability, except for formulas that are already bounded from the beginning.

1.3 Contributions

In this thesis, we aim to address the problem of proving the satisfiability of quantifier-free Non-linear Transcendental Arithmetic formulas ($\mathcal{NTA}$). This is a notoriously hard challenge for SMT solvers, given the undecidability of $\mathcal{NTA}$, the unavailability of a way to finitely represent satisfying assignments, and the difficulty of reasoning about transcendental functions through symbolic methods. Current SMT solvers are only able to prove the satisfiability of $\mathcal{NTA}$ formulas for very simple cases.

The contribution of this thesis is a push forward to the state-of-the-art for SMT($\mathcal{NTA}$), both from a practical and a theoretical point of view. We introduce a novel procedure able to prove satisfiability for $\mathcal{NTA}$ formulas, and theoretically characterize the sub-class of $\mathcal{NTA}$ for which such procedure can terminate successfully. The aim of our work is two-fold: on the practical side, we develop a prototype tool which is able to solve more than four times the benchmarks solved by the best state-of-the-art SMT solvers; on the theoretical side, we prove an original theorem that shows that a weaker notion of decidability holds for $\mathcal{NTA}$ formulas (i.e., that there exists a procedure that always terminates on formulas that respect certain robustness assumptions), contributing to closing the gap between the general undecidability of $\mathcal{NTA}$ and practically relevant sub-classes that can be effectively solved.

The novel approach that we introduce leverages techniques coming from the fields of numerical analysis and topology, and integrates them within the SMT context. Numerical methods, which, albeit inexact, are very fast, are used to generate candidate approximate solutions (we denote such approach *Logic-to-Optimization*). Topological methods, on the other hand, are able to prove the existence of a solution within a given bounded region

without having to explicitly express the solution. We design a procedure based on a fruitful combination of these methods, and implement it in the prototype UGOTNL[1]. Extensive experimental evaluation demonstrates that our tool significantly outperforms other tools on $\mathcal{NTA}$ satisfiable benchmarks.

Secondly, we address the problem of verifying the result returned by a solver by introducing a notion of satisfiability certificate for $\mathcal{NTA}$ problems, and we show how formulating the problem of proving satisfiability as the problem of finding a certificate allows the design of more efficient procedures that show satisfiability by systematically searching for such certificates. Furthermore, we study algorithmically solvable sub-classes of $\mathcal{NTA}$, which we characterize by providing lower and upper bounds in terms of relevant well-known classes, and we prove that, despite the undecidability of $\mathcal{NTA}$, one of such sub-classes admits a procedure that always terminates successfully under certain robustness assumptions.

Finally, we study how the Logic-to-Optimization approach can fit in a general framework such as the MCSAT calculus, guiding the model-construction phase and at the same time benefiting from the conflict-driven reasoning in order to refine the optimization process. We implement the new approach inside the YICES SMT solver, starting with the theory of Non-linear Integer Arithmetic ($\mathcal{NIA}$). Experimental evaluation on $\mathcal{NIA}$ benchmarks shows that the new approach increases the solver performances on both satisfiable and unsatisfiable instances.

To summarize the main contributions of the thesis, we:

1. Propose a novel procedure to prove satisfiability of $\mathcal{NTA}$ formulas based on numerical and topological techniques, that outperforms state-of-the-art SMT solvers.

2. Introduce a notion of satisfiability certificate for $\mathcal{NTA}$, formulate the satisfiability problem as the problem of searching for such a certificate, and introduce new heuristics.

3. Theoretically characterize a sub-class of $\mathcal{NTA}$, for which we prove that a weaker notion of decidability holds.

4. Propose an optimization-guided approach to the MCSAT calculus, and evaluate it over Non-linear Integer Arithmetic ($\mathcal{NIA}$) benchmarks.

[1]The name is an acronym of "Unconstrained Global Optimization and Topological degree for Non-Linear", and can be read as "*You got Non-Linear*".

1.3.1 Structure of the Thesis

The thesis is organized as follows.

In Chapter 2, we provide a formal introduction to the field of Satisfiability Modulo Theories, introducing the common terminology and the notation that will be used in the thesis. We will also provide the necessary background that will be used throughout the thesis, treating topics such as zero-existence theorems, interval arithmetic, the concepts of robustness and quasi-decidability, and unconstrained optimization.

The following four chapters will mostly follow the four main contributions listed at the end of the previous subsection.

In Chapter 3, we introduce a novel procedure for SMT($\mathcal{NRA}$) and SMT($\mathcal{NTA}$) formulas. In particular, in Section 3.1 we introduce the *Logic-to-Optimization* (L2O) approach. In Section 3.2 we show how L2O can be used together with the topological degree test and interval arithmetic to prove the satisfiability of sets of constraints. In Section 3.3 we extend to full SMT formulas, presenting both an eager approach (implemented in the prototype UGOTNL) and a lazy approach (implemented as an integration of UGOTNL with MATHSAT). In 3.4, we experimentally evaluate the two approaches over $\mathcal{NRA}$ and $\mathcal{NTA}$ benchmarks.

In Chapter 4, we formulate the problem of SMT($\mathcal{NTA}$) as a certificate search problem. In particular, in 4.2, we give the definition of satisfiability certificate for $\mathcal{NTA}$. In Section 4.3, we present our method. In Section 4.4, we describe several heuristics. In Section 4.5, we experimentally evaluate different heuristics configurations implemented in UGOTNL on $\mathcal{NTA}$ benchmarks.

In Chapter 5, we give a theoretical characterization of $\mathcal{NTA}$ sub-classes that are solvable by our method. In particular, in Section 5.1 we provide an extensive background on robustness and on regularity (in the sense of differential topology), and we formally define the $\mathcal{NTA}$ sub-classes of our interest (*robust under instantiation*, and *robust under equation adding*). In Sections 5.2 and 5.3, we provide strict lower and upper bounds for the main sub-class of our interest (i.e. robust under instantiation). In Section 5.4, we compare the two sub-classes. In Section 5.5, we prove the existence of a variant of our method that is guaranteed to always terminate on formulas robust under instantiation.

In Chapter 6, we study an integration between Logic-to-Optimization and MCSAT. In particular, in Section 6.1 we outline the design of the integration between L2O and MCSAT. In Section 6.2, we experimentally evaluate our approach in the SMT solver YICES over $\mathcal{NIA}$ benchmarks. In Section 6.3, we compare with related works.

Finally, in Chapter 7, we draw some conclusions and outline possible future research directions.

1.3.2 Relevant publications

The publications relevant to the thesis are the following:

[71] Enrico Lipparini, Alessandro Cimatti, Alberto Griggio, and Roberto Sebastiani. (2022) "Handling polynomial and transcendental functions in SMT via unconstrained optimisation and topological degree test", ATVA 2022

[72] Enrico Lipparini and Stefan Ratschan. (2023). "Satisfiability of non-linear transcendental arithmetic as a certificate search problem", NFM 2023

[73] Enrico Lipparini and Stefan Ratschan. (2024). "Satisfiability of non-linear transcendental arithmetic as a certificate search problem (extended version)", Journal of Automated Reasoning *(under minor revision)*

More in details: Chapter 3 is based on the work in [71], Chapter 4 is based on the work in [72], and Chapter 5 is based on the work in [73]. Finally, Chapter 6 is based on an ongoing work with Ahmed Irfan and Stéphane Graham-Lengrand from SRI International.

Other publications by the author, accomplished as part of the PhD program, but not included in this thesis, include:

[11] Massimo Bartoletti, Angelo Ferrando, Enrico Lipparini, and Vadim Malvone. (2024). "Solvent: Liquidity verification of smart contracts", iFM 2024

Chapter 2

Preliminaries

In this chapter, we provide the necessary background.

In particular, in Section 2.1, we provide a formal introduction to the field of Satisfiability Modulo Theories and to the theory of non-linear transcendental arithmetic, defining the common terminology and notation that will be used in the thesis. In Section 2.2, we introduce Interval Arithmetic, a building block of our method. In Section 2.3, we discuss the topological degree test, a zero-existence theorem that will play a central role in the thesis. In Section 2.4, we introduce the concepts of robustness and of quasi-decidability. Finally, in Section 2.5, we give some background on the field of Unconstrained Optimization.

2.1 Satisfiability Modulo Theories (SMT)

2.1.1 Syntax

We now define the syntax of Satisfiability Modulo Theory (SMT) problems, and provide some basic terminology.

Definition 2.1.1 (Signature)**.** *A* signature Σ *is a 4-tuple* $(\mathcal{F}, \mathcal{P}, Vars, ar)$ *such that:*

- $\mathcal{F}$ *is a set of function symbols*
- $\mathcal{P}$ *is a set of predicate symbols*
- *Vars is a set of variables*
- $ar : (\mathcal{F} \cup \mathcal{P}) \to \mathbb{N}$ *is the arity function of function and predicate symbols*

Definition 2.1.2 (Constant)**.** *A* $0-$*arity function symbol is called a* constant

Definition 2.1.3 (Boolean atom)**.** *A* $0-$*arity predicate symbol is called a* Boolean atom

Definition 2.1.4 (Term)**.** *Given a signature* $\Sigma = (\mathcal{F},\ \mathcal{P},\ Vars,\ ar)$*, a* Σ-term *is either a constant, a variable, or built by applying function symbols to* Σ*-terms.*

Definition 2.1.5 (Atom)**.** *Given a signature* $\Sigma = (\mathcal{F},\ \mathcal{P},\ Vars,\ ar)$*, and a predicate* $R \in \mathcal{P}$ *with* $ar(R) = n$*, then* $R(t_1, \cdots, t_n)$ *is called a* Σ-atom*.*

Definition 2.1.6 (Literal)**.** *A* Σ-literal *is either an* Σ*-atom or the negation of an* Σ*-atom.*

Definition 2.1.7 (Clause)**.** *A* Σ-clause *is a disjunction of* Σ*-literals.*

Definition 2.1.8 (Formula)**.** *A* Σ-formula ϕ*is either:*

- *A* Σ*-literal*
- $\phi \equiv \phi_1 \bowtie \phi_2$ *or* $\phi \equiv \neg\phi_1$*, where* $\bowtie \in \{\wedge, \vee\}$ *and* ϕ_1 *and* ϕ_2 *are* Σ*-formulas*
- $\phi \equiv \exists x.\phi_1$ *or* $\phi \equiv \forall x.\phi_1$*, where* $x \in Var$ *and* ϕ_1 *is a* Σ*-formula*

Definition 2.1.9 (Quantifier-free formula)**.** *A formula* ϕ *is said to be* quantifier-free *if it does not contain the* $\exists$ *nor the* $\forall$ *quantifier symbols.*

Definition 2.1.10 (CNF)**.** *A formula is in* Conjunctive Normal Form (CNF) *if it is a conjunction of clauses.*

Definition 2.1.11 (DNF)**.** *A formula is in* Disjunctive Normal Form (CNF) *if it is a disjunction of conjunctions of literals.*

2.1.2 Semantics

We now define the semantics of SMT.

Definition 2.1.12 (Interpretation)**.** *Given a signature* $\Sigma = (\mathcal{F},\ \mathcal{P},\ Vars,\ ar)$*, a* Σ-interpretation $\mathcal{I}$ *is a pair* (D, α)*, where* D *is a domain, and* α *is an assignment that maps:*

- *Constants and variables to elements of* D
- *Function symbols* $F \in \mathcal{F}$ *with* $ar(F) = n$*, to n-ary functions*

$$\mathcal{I}(F) : D^n \to D$$

- *Predicate symbols $R \in \mathcal{P}$ with $ar(R) = n$ to predicates $\mathcal{I}(R) : D^n \to \{\top, \bot\}$*

Definition 2.1.13 (Model)**.** *Given a signature $\Sigma = (\mathcal{F},\ \mathcal{P},\ Vars,\ ar)$ and a quantifier-free Σ-function ϕ, we say that a Σ-interpretation $\mathcal{I}$ is a* Σ-model *of ϕ if ϕ evaluates to $\top$ under $\mathcal{I}$.*

Given $\phi \equiv \exists x.\phi_1$, a Σ-interpretation $\mathcal{I}$ is a Σ-model *of ϕ (or $\mathcal{I}$* satisfies *ϕ) if there exists $v \in D$ such that $\mathcal{I}$ with the mapping $x \mapsto v$ is a Σ-model for ϕ_1.*

Definition 2.1.14 (Theory)**.** *A Σ-theory $\mathcal{T}$ is a set of Σ-interpretations.*

Note that, traditionally, in logic a *theory* is defined by a set of axioms, and an interpretation is said to belong to a theory if it satisfies such axioms. In the SMT literature, however, a theory is directly defined as a set of interpretations.

Definition 2.1.15 (Satisfiability)**.** *We say that a Σ-formula ϕ is* satisfiable in *$\mathcal{T}$ (respectively,* unsatisfiable in *$\mathcal{T}$) if and only if there exists (resp., there does not exist) a Σ-model from $\mathcal{T}$ that satisfies ϕ.*

Definition 2.1.16 (Equi-satisfiability)**.** *Two Σ-formulas ϕ_1, ϕ_2 are said to be* equi-satisfiable *if and only if either both ϕ_1 and ϕ_2 are satisfiable, or both ϕ_1 and ϕ_2 are unsatisfiable.*

Definition 2.1.17 (SMT problem)**.** *Given a theory $\mathcal{T}$, the* SMT problem *is the problem of deciding whether a formula is satisfiable in $\mathcal{T}$ or not.*

From now on, for simplicity, we will omit the "$\Sigma-$" from the previous definitions, when there is no risk of ambiguity.

2.1.3 SMT solvers

An *SMT solver* is a tool that solve SMT problems. Some of the most performant and extensive state-of-the art SMT solvers include: CVC5 [8], MATHSAT [31], SMT-RAT [34], YICES [41], and Z3 [37]. Many other solvers have contributed or continue to contribute to the advancement of the field [33, 55, 14, 50, 19, 62, 110, 27, 16].

Besides being able to prove the (un)satisfiability of an SMT formula ϕ, SMT solvers often provide additional features.

- *Model generation.* If the formula is satisfiable, the solver returns an assignment that satisfies ϕ. Such assignment is sometimes called a *witness*, and it can be used to certify that the satisfiability result returned by the solver is indeed correct.

- *Unsatisfiable cores.* If the formula is unsatisfiable, the solver returns an unsatisfiable subformula of the original formula.

- *Incrementality and Backtracking.* The solver is able to keep track of its computational status from one call to another, so that, when presented with a new formula, obtained by adding or removing constraints from a previously solved formula, it is able to exploit such retained information to solve more efficiently the new call. We talk about Incrementality when the solver uses previous information to solve a formula obtained by *adding* constraints to a previously solved formula, and about Back-tracking when the information is used to solve a formula obtained by *removing* constraints.

There are two main approaches to SMT solving:

- *Lazy approach.* The solver makes use of an integration between *theory solvers* (i.e., decision procedures for conjunctions of theory constraints) and SAT solvers [100, 5]. First, the atoms of the original formula are abstracted into Boolean propositions, and the obtained abstracted SAT problem is passed to a SAT solver that checks its satisfiability. If the SAT formula is unsatisfiable, then also the original formulas is unsatisfiable. Otherwise, the SAT solver returns an abstracted model, which is then *concretized* (the inverse process of abstracting), and the resulting conjunction of theory constraints is fed to the theory solver, that assests its satisfiability in the theory. If it is theory-satisfiable, then the procedure terminates; otherwise, the theory solver returns a reason of unsatisfiability, that is used by the SAT solver to refine its search. The most common lazy approach is DPLL(T) [89], a generalization of the classic DPLL algorithm used in SAT, which is often used in a conflict-driven fashion (CDCL(T)).

- *Eager approach.* The solver directly encodes the original problem into an equisatisfiable SAT problem, and relies on SAT solvers to decide its satisfiability. This approach has been applied, e.g., to *equality logic* [21], *difference logic* [105], *bit-vectors* [20], *linear arithmetic* [104], and *algebraic datatypes* [101]. In the case of theories with more complex solution spaces, and that are not reducible to propositional logic, however, such encoding is not feasible.

  Sometimes, the term *eager* is used to indicate that theory solvers are called early during the DPLL(T) search, i.e., before all abstracted Boolean have been assigned (compared to *lazy*, that indicates that they are called only after a complete assignment has been

found) [88]. In the rest of the paper, when we use the term *eager*, we will mostly refer to this notion of eagerness.

2.1.4 Non-linear Transcendental Arithmetic (NTA)

Our theory of interest is the quantifier-free theory of non-linear real arithmetic augmented with trigonometric and exponential transcendental functions, SMT($\mathcal{NTA}$).
Notation. We denote SMT($\mathcal{NTA}$)-formulas by ϕ, ψ, clauses by C_1, C_2, literals by l_1, l_2, real-valued variables by $x_1, x_2, \ldots$, constants by a, b, intervals of real values by $I = [a,b]$, boxes by $B = I_1 \times \cdots \times I_n$, logical terms with addition, multiplication and transcendental function symbols by f, g, and multivariate real functions with F, G, H. For any formula ϕ, we denote by $Vars_{\mathcal{R}}(\phi)$ the set of its real-valued variables. When there is no risk of ambiguity we write f, g to also denote the real-valued functions corresponding to the standard interpretation of the respective terms. We assume that formulas are in Conjunctive Normal Form (CNF) and that their atoms are in the form $f \bowtie 0$, with $\bowtie \in \{=, \leq, <\}$. We remove the negation symbol by rewriting every occurrence of $\neg(f = 0)$ as $(f < 0 \vee 0 < f)$ and distributing $\neg$ over inequalities.

Points and boxes. Since we have an order on the real-valued variables $x_1, x_2, \ldots$, for any set of variables $V \subseteq \{x_1, x_2, \ldots\}$ we can view an assignment $p : V \to \mathbb{R}$ equivalently as the $|V|$-dimensional point $p \in \mathbb{R}^{|V|}$, and an *interval assignment* $\beta : V \to \{[a,b] : a, b \in \mathbb{R}\}$ equivalently as the $|V|$-dimensional box $B \subseteq \mathbb{R}^{|V|}$. By abuse of notation, we will use both representations interchangeably, using the type $\mathcal{R}^V$ both for assignments in $V \to \mathbb{R}$ and points in $\mathbb{R}^{|V|}$, and the type $\mathcal{B}^V$ both for interval assignments in $V \to \{[a,b] : a, b \in \mathbb{R}\}$ and corresponding boxes. This will allow us to apply mathematical notions usually defined on points or boxes to such assignments, as well. Given a point $p \in \mathcal{R}^V$, and a subset $V' \subseteq V$, we denote by $proj_{V'}(p) \in \mathcal{R}^{V'}$ the projection of p to the variables in V', that is, for all $v \in V'$, $proj_{V'}(p)(v) := p(v)$. Finally, we will use variable assigments $\nu : V \to \mathbb{Q}$ as substitutions, denoting by $\nu(\phi)$ the result of replacing every variable $v \in V \cap Vars_{\mathcal{R}}(\phi)$ in ϕ by $\nu(v)$.

Systems of equations and inequalities. We say that a formula ϕ that contains only conjunctions of atoms in the form $f = 0$ and $g \leq 0$ is a *system of equations and inequalities*. If ϕ contains only equations (inequalities) then we say it is a *system of equations* (*inequalities*).

A system of equations $f_1 = 0 \wedge \cdots \wedge f_n = 0$, where the $f_1, \cdots, f_n$ are terms in the variables $x_1, \cdots, x_m$, can be seen in an equivalent way as the equation $F = 0$, where F is the real-valued function $F := f_1 \times \cdots \times f_n : \mathbb{R}^m \to \mathbb{R}^n$ and 0 is a compact way to denote the point $(0, \cdots, 0) \in \mathbb{R}^n$. Analogously, we can see a system of inequalities $g_1 \leq 0 \wedge \cdots \wedge g_k \leq 0$ as the inequality $G \leq 0$, where G is the real-valued function $G := g_1 \times \cdots \times g_k : \mathbb{R}^m \to \mathbb{R}^k$ and $\leq$ is defined element-wise. We will write $eq(\phi)$ for the function F defined by the equations in the formula ϕ and $ineq(\phi)$ for the function G defined by the inequalities in ϕ.

We say that a system of equations and inequalities is *bounded* if every free variable appearing in the system has an associated bound consisting of a closed interval with rational endpoints.

The handling of strict inequalities is an easy, but technical extension of our method, which we will avoid for most of the work to stream-line the presentation, and address in local points when necessary.

2.2 Interval Arithmetic

Interval Arithmetic [97, 86, 84] is a systematic approach to represent real numbers as intervals by computing safe bounds that account for rounding errors.

In our context, we can see Interval Arithmetic as an algorithm $\mathcal{IA}$ that, given a box B and an $\mathcal{NTA}$-term representing a function H, is able to compute an interval $\mathcal{IA}_H(B)$ that over-approximates the image of H over B (i.e. the set $\{H(p) \mid p \in B\}$). Since such an algorithm is based on floating point arithmetic, the time needed for computing $\mathcal{IA}_H(B)$ does not grow with the size of the involved numbers. Moreover, conservative rounding guarantees correctness under the presence of round-off errors. In this thesis, we will use interval arithmetic within topological degree computation [46], and as a tool to prove the validity of inequalities on boxes.

We now give the definition of interval-computable functions. The intuition is that a function is interval-computable if it is possible to compute arbitrarily precise images for every interval domain.

Recall that the width of an interval $I \stackrel{\text{def}}{=} [a_I, b_I]$ is defined as $\text{width}(I) \stackrel{\text{def}}{=} b_I - a_I$, and that the width of a box is defined as $\text{width}(B) \stackrel{\text{def}}{=} \max_i(\text{width}(I_i))$.

Definition 2.2.1 (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 $\mathcal{IA}_F$ that, for every box $B \subseteq \Omega$ with rational vertices, computes a box $\mathcal{IA}_F(B)$ with rational vertices, such that:*

- $F(B) \subseteq \mathcal{IA}_F(B)$*; and*
- $\forall \varepsilon > 0 : \exists \delta > 0$ *such that for every B having* $\text{width}(B) < \delta$*, then* $\text{width}(\mathcal{IA}_F(B)) < \varepsilon$.

Theorem 2.2.1. *Every function in $\mathcal{NTA}$ is interval computable.*

Proof. We refer to section 5.4 of [84]. □

Given a formula ϕ in m real variables and a box $B \stackrel{\text{def}}{=} I_1 \times \cdots \times I_m \subset \mathbb{R}^m$ (where $I_i \stackrel{\text{def}}{=} [a_i, b_i]$), we define the restriction of ϕ to the box B as

$$\phi_{|B} := \phi \wedge \bigwedge_{x_i \in Vars_{\mathcal{R}}(\phi)} (a_i \leq x_i \wedge x_i \leq b_i) \tag{2.1}$$

We say that $\phi_{|B}$ is a *bounded formula*.

2.3 Topological degree test

The notion of the degree of a continuous function (also called the *topological degree*) comes from the field of topology [43, 39]. For a continuous function $F : B \subseteq \mathbb{R}^n \to \mathbb{R}^n$ and a point $p \in \mathbb{R}^n$ such that $p \notin F(\partial B)$ (where ∂B is the topological boundary of B), the degree $\deg(F, B, p)$ is a computable [1, 46] integer. For $p = 0$, this integer provides information about the roots of F in B.

The topological degree can be defined in several ways. Here, we give its definition in the sense of differential topology for real functions from a bounded domain $\Omega \subset \mathbb{R}^n$ to $\mathbb{R}^n$, and we refer the reader interested in a more in-depth discussion to [90]. We recall that Ω^o is the interior of Ω, and that $\partial\Omega$ is the topological boundary of Ω (that is, the set of points in the closure $\bar{\Omega}$ of Ω that are not in Ω^{o}). With F' we denote the Jacobian matrix of F, and with det the determinant of a matrix.

Definition 2.3.1 (Topological degree)**.** *Let $\Omega \subset \mathbb{R}^n$ be open and bounded, let $F : \bar{\Omega} \to \mathbb{R}^n$ be a real function continuous and infinitely differentiable in Ω, and let $p \in \mathbb{R}^n$ such that*

$p \notin F(\partial\Omega)$. If p is regular (i.e. for all $y \in F^{-1}(p)$, $\det F'(y) \neq 0$) then the topological degree of F in p is defined as

$$\deg(F,\Omega,p) \stackrel{def}{=} \sum_{y\in F^{-1}(p)} \operatorname{sign}\det F'(y).$$

This definition can be extended to non-regular values p in a unique way by continuity of $\deg(F,\Omega,p)$ as a function in p, or alternatively, be defined axiomatically (see [90] for more details).

In the following, given a box B, with an abuse of notation we will write $\deg(F,B,p)$ to mean $\deg(F,B^{\circ},p)$. Moreover, we will write $\deg(F,B)$ to mean $\deg(F,B,0)$.

The topological degree satisfies several interesting properties. The one that plays a pivotal role in our work is the following, which we will call *topological degree test*.

Property 2.3.1 (Topological degree test)**.** *If $\deg(F,B) \neq 0$, then the equation $F = 0$ has a solution in B.*

Proof. See the *Solvability* property in Section 1.2 of [90]. □

The topological degree test belongs to the class of *zero-existence theorems*, i.e. theorems that prove that, if a function satisfies certain conditions in a given domain, then it has a zero in that domain. Other zero-existence theorems, which are less powerful than the topological degree test, include, e.g., the Generalize Intermediate Value Theorem [86], Miranda's theorem [83], and Borsuk's theorem [77].

The converse of Property 2.3.1 is not true, and the existence of a root does not imply nonzero degree in general. Still, if a box contains one isolated zero with non-singular Jacobian matrix, then the topological degree is non-zero [43].

To give a better intuition of the topological degree, we describe it in small dimensions.

For $n = 1$, the topological degree test is analogous to a corollary of the Intermediate Value Theorem. Indeed, given a continuous function $F : [a,b] \to \mathbb{R}$, it can be proved that $\deg(F,B)$ is zero if and only if $F(a)$ and $F(b)$ have the same sign, otherwise it is either 1 or -1 depending on whether $F(a) < 0$ and $F(b) > 0$ or vice-versa. More precisely, we can prove that $\deg(F,B) = \frac{\operatorname{sign}(F(b)) - \operatorname{sign}(F(a))}{2}$. This formula implies that, in dimension one, the topological degree test is equivalent to the corollary of the Intermediate Value Theorem known as Bolzano's theorem [15], which states that, if a continuous function has values of opposite sign in an interval, then it has a zero in that interval.

Example 2.3.1. *Let consider three different functions, x, $-x$, and x^2, with domain in the interval $I \equiv [-1,1]$.*

- *For $F(x) = x$, we have that $\deg(F,I,0) = \frac{1-(-1)}{2} = 1$, which implies that F has a solution in I.*
- *For $F(x) = -x$, we have that $\deg(F,I,0) = \frac{-1-1}{2} = -1$, which implies that F has a solution in I.*
- *For $F(x) = x^2$, we have that $\deg(F,I,0) = \frac{1^2-(-1)^2}{2} = 0$, which does not give any information about the solutions of F in I.*

For $n = 2$, given a rectangle $R \equiv [a,b] \times [a',b']$ and $F : R \to \mathbb{R}^2$, the topological degree is the number of times that the image of the boundary of R wraps around the origin counter-clockwise (in this dimension, the topological degree is commonly referred to as the *winding number*). See also Example 2.3.2.
The topological degree has proven to be computable if $0 \notin F(\partial B)$ [1]. A practical tool for computing the degree is TOPDEG[1], which implements the interval-based algorithm described in [46], and that is guaranteed to terminate for interval-computable functions.

Example 2.3.2. *To give an idea of how* TOPDEG *works, let's consider the function $F : R \subseteq \mathbb{R}^2 \to \mathbb{R}^2$, defined by*

$$F(x,y) = \begin{cases} \sin(y) - e^x & (F_1) \\ \cos(y) - \sin(8x^2 - 0.2) & (F_2) \end{cases}$$

over the rectangle $R \equiv I_1 \times I_2 \equiv ([-0.023, -0.017],\ [1.756, 1.779])$. The solution spaces of F_1 and F_2 are depicted in Figure 2.1.

TOPDEG *checks the signs of F_1 and F_2 on the four edges of R, using interval arithmetic. It finds that F_1 is everywhere positive ($+$) on the left edge, everywhere negative ($-$) on the right edge, and somewhere zero (0) on the upper and lower edges.*

Vice-versa, it finds that F_2 is everywhere negative on the upper edge ($-$), everywhere positive ($+$) on the lower edge of R, and somewhere zero (0) on the upper and lower edges.

This suffices to prove that the image of the boundary of R wraps around 0 exactly one time counter-clockwise. Intuitively, we can see that by the fact that, for example, F_1, starting from the left edge and moving counter-clockwise, has the following signs: $+$, 0, $-$, 0; similarly,

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

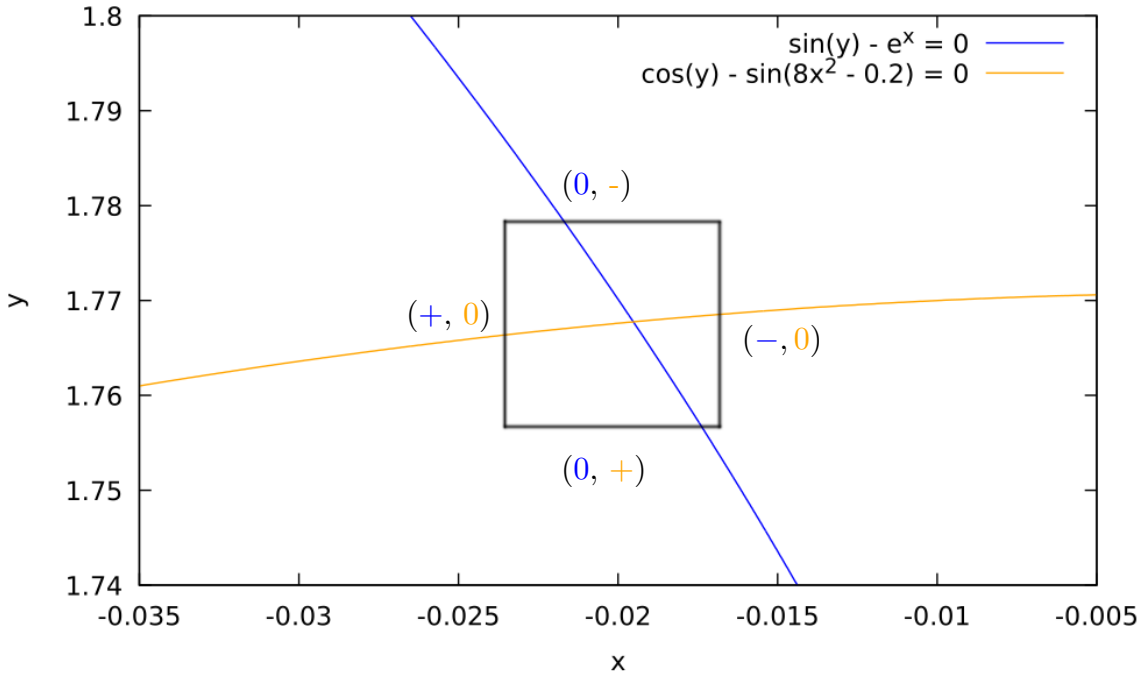

Figure 2.1 Computation of the topological degree in dimension 2, for the Example 2.3.2.

F_2, starting from the lower edge, has the signs: $+$, 0, $-$, 0. This implies that $\deg(F,R) = 1$, *which, by the topological degree test, proves that F has at least a solution in R.*

We can better see the relation between the topological degree and the existence of a solution through the lents of the Intermediate Value Theorem, at least in this simple example.

Given the signs $(+)$ on $\{a_{I_1}\} \times I_2$ and $(-)$ on $\{b_{I_1}\} \times I_2$, then, for every $c_y \in I_2$, the univariate function $F_{1|_{c_y}} : I_1 \to \mathbb{R}$, defined by $F_{1|_{c_y}}(x) = \sin(c_y) - e^x$, has a solution in I_1 by the Intermediate Value Theorem.

Vice-versa, given the signs $(+)$ on $I_1 \times \{a_{I_2}\}$ and $(-)$ on $I_1 \times \{b_{I_2}\}$, then, for every $c_x \in I_1$, the univariate function $F_{2|_{c_x}} : I_2 \to \mathbb{R}$, defined by $F_{2|_{c_x}}(x) = \cos(y) - \sin(8{c_x}^2 - 0.2)$, has a solution in I_2 by the Intermediate Value Theorem.

By the continuity of F_1 and F_2, their solution spaces must intersect in $I_1 \times I_2$. Hence, F has a solution in R.

There exist more complex cases than the one presented in Example 2.3.2, for which the non-zero topological degree cannot be simply inferred by a straight-forward application of the Intermediate Value Theorem. These cases require splitting the starting box into smaller sub-boxes, computing the signs of the functions over the faces of these sub-boxes, and then,

using the information about the signs in a combinatorial way, calculate the degree. We refer the reader interested in the details of the algorithm to [46].

2.4 Robustness and quasi-decidability

Intuitively, a formula is robust if its satisfiability status does not change under "small" perturbations. Robustness is a desirable property in many real-world applications, as already observed in the literature (e.g. [93, 53]).

The related notion of quasi-decidability [47, 48] is then a property that allows to circumvent general undecidability results for a class of formulas when focusing only on robust inputs.

Both concepts are formalized below.

Definition 2.4.1 (Distance between two formulas)**.** *Let $\phi_{|B}$ be a bounded formula, and let $T \stackrel{def}{=} \{t_1, \cdots, t_k\}$ be set of the terms occurring in ϕ (including constants).*

Let $\phi'_{|B'}$ be a formula that can be obtained from $\phi_{|B}$ by term substitution (i.e. $\phi'_{|B'} \equiv \phi_{|B}[s_1/t_1, \cdots, s_k/t_k]$ for some terms $s_1, \ldots s_k$).

If $B' = B$, then we define the distance between $\phi_{|B}$ and $\phi'_{|B'}$ as

$$\mathrm{d}(\phi_{|B}, \phi'_{|B'}) \stackrel{def}{=} \max_k(\|t_k - s_k\|_B)$$

where $\|t_k - s_k\|_B \stackrel{def}{=} \sup\{|t_k(x) - s_k(x)| : x \in B\}$.
For a bounded formula $\psi_{|B'}$ that cannot be obtained from $\phi_{|B}$ by term substitution, or for which $B' \neq B$, then we define $\mathrm{d}(\phi_{|B}, \psi_{|B'}) \stackrel{def}{=} +\infty$.

Definition 2.4.2 (Robustness of a formula)**.** *Given $\alpha \in \mathbb{R}_{>0}$, a bounded formula ϕ is α-robust iff for every ϕ' s.t. $\mathrm{d}(\phi, \phi') < \alpha$, ϕ and ϕ' are equisatisfiable. A formula ϕ is robust iff there exists $\alpha \in \mathbb{R}_{>0}$ s.t. ϕ is α-robust.*

Example 2.4.1. *Let $\phi \equiv x^2 = 1$ and $B = [0, 2]$. The formula $\phi_{|B}$ is robust because we can take $\alpha = 0.001$ and show that there exists no $\psi_{|B}$ such that $\mathrm{d}(\phi_{|B}, \psi_{|B}) < \alpha$ and $\psi_{|B}$ is unsatisfiable.*

Example 2.4.2. *Let $\phi \equiv \sin(x) = 1$ and $B = [0, 2]$. The formula $\phi_{|B}$ is not robust: in fact, for every $\alpha > 0$, the formula $\psi \equiv \sin(x) = 1 + \alpha$ is unsatisfiable and such that $\mathrm{d}(\phi_{|B}, \psi_{|B}) < \alpha$.*

We now define the notion of quasi-decidability:

Definition 2.4.3 (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.*

We remark that, according to this definition, a quasi-decidability algorithm is allowed to non-terminate for instances that are not robust.

An important result, proven in [48], and that we will extend in Chapter 5, is the following:

Theorem 2.4.1. *The class $\mathcal{B}$ of formulas of the kind*

$$(f_1 = 0 \wedge \cdots \wedge f_n = 0 \wedge g_1 \geq 0 \wedge \cdots \wedge g_k \geq 0)_{|B} \tag{2.2}$$

where B is an m-box with rational vertices, f_i and g_j are functions in $\mathcal{NTA}$, and $n = m$ or $n = 0$, is quasi-decidable.

2.5 Unconstrained Optimisation

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

Unconstrained global optimisation is the problem of minimizing a function H on its whole domain Ω. A common approach to tackle this problem is leveraging fast local optimisation techniques.

In this thesis, we use a Monte Carlo Markov Chain method called Basin-hopping [111], based on the Metropolis-Hasting algorithm [82]. 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.

Since unconstrained global optimisation is undecidable, what Basin-hoppin usually does is returning the best local minimum found, either after a certain number of steps or whenever

another given condition (such as a threshold) is satisfied. If no limit on iterations or other stopping criterion is given, the algorithm never terminates.

We present a modified version of the algorithm that, given a maximum number of iterations, returns *all* the local minima found.

Algorithm 1 BASIN-HOPPING

Input: A function $H : \mathbb{R}^m \to \mathbb{R}$, a starting point $p \in \mathbb{R}^m$, and a limit of iterations k

Output: A set $\{p_0^*, \cdots, p_k^*\}$ of local minima

$p_0^* \leftarrow \texttt{LM}(H, p)$ ▷ `LM` is a local minimization procedure
$p_{cur} \leftarrow p_0^*$
for $i = 1$ to k **do**
    $d \leftarrow$ a random perturbation generated from a predefined probability distribution
    $p_i^* \leftarrow LM(H,\ p_{cur} + d)$
    if $h(p_i^*) < H(p_{cur})$ **then**
        `accept` $\leftarrow$ `True`
    else
        $m \leftarrow$ a real value randomly generated from the uniform distribution on $[0, 1]$
        `accept` $\leftarrow$ `True` if $m < \exp(H(p_{cur}) - H(p_i^*))$ otherwise `False`
    end if
    if `accept` **then**
        $p_{cur} \leftarrow p_i^*$
    end if
end for
return $\{p_0^*, \cdots, p_k^*\}$

Chapter 3

Numerical Optimization and Topological Degree for SMT(NRA) and SMT(NTA)

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 [94]. Existing SMT solvers therefore have to resort to incomplete techniques in order to handle $\mathcal{NTA}$ constraints [30, 50], 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, 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 [53], which are typically significantly more scalable in practice than exact methods.

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* [90, 46] – 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 chapter 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 [48] by pairing the topological degree test with interval arithmetic to deal with inequalities. 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 [31]. 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.

Content. This chapter is based on the work presented in [71], and it is organized as follows: in 3.1 we describe how we use unconstrained optimisation to find candidate models through the Logic-to-Optimization technique; in 3.2 we describe a general procedure, restricted to conjunctions of $\mathcal{NTA}$ constraints, based on the topological degree test and interval arithmetic; in 3.3 we extend the previous procedure to general $\mathcal{NTA}$ formulas, following either an eager or a lazy approach; in 3.4 we present our experimental evaluation; finally, in 3.5, we discuss some related work.

3.1 Logic-to-Optimization

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 $x_1, \ldots, x_m$ into a real-valued non-negative function $H : \mathbb{R}^m \mapsto \mathbb{R}^{\geq 0}$, such that p is a model of ϕ only if $H(p) = 0$, and then use an unconstrained optimisation routine to search for 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. [51]. In general, logic-to-optimisation can be performed in different ways, 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 $b_1, \ldots, b_k$, 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 $\mathcal{L}2\mathcal{O}$ that maps a formula to a non-negative real function from $\mathbb{R}^{m+k}$ to $\mathbb{R}^{\geq 0}$ as follows:

$$
\begin{array}{lcl}
\mathcal{L}2\mathcal{O}(f \bowtie 0),\ \bowtie \in \{\leq, =\} & \stackrel{\text{def}}{=} & (\mathsf{if}\ ([f](p) \bowtie 0)\ \mathsf{then}\ 0\ \mathsf{else}\ [f]^2(p)) \\
\mathcal{L}2\mathcal{O}(f < 0) & \stackrel{\text{def}}{=} & \mathcal{L}2\mathcal{O}(f \leq 0) \\
\mathcal{L}2\mathcal{O}(\neg(f \bowtie 0)),\ \bowtie \in \{<, \leq\} & \stackrel{\text{def}}{=} & \mathcal{L}2\mathcal{O}(-f \bowtie 0) \\
\mathcal{L}2\mathcal{O}(\neg(f = 0)) & \stackrel{\text{def}}{=} & (\mathsf{if}\ ([f](p) = 0)\ \mathsf{then}\ 1\ \mathsf{else}\ 0) \\
\mathcal{L}2\mathcal{O}(b) & \stackrel{\text{def}}{=} & \mathcal{L}2\mathcal{O}(-x_b \leq 0) \\
\mathcal{L}2\mathcal{O}(\neg b) & \stackrel{\text{def}}{=} & \mathcal{L}2\mathcal{O}(x_b + 1 \leq 0) \\
\mathcal{L}2\mathcal{O}(\phi_1 \wedge \phi_2) & \stackrel{\text{def}}{=} & \mathcal{L}2\mathcal{O}(\phi_1) + \mathcal{L}2\mathcal{O}(\phi_2) \\
\mathcal{L}2\mathcal{O}(\phi_1 \vee \phi_2) & \stackrel{\text{def}}{=} & \mathcal{L}2\mathcal{O}(\phi_1) * \mathcal{L}2\mathcal{O}(\phi_2),
\end{array}
$$

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 $\mathcal{L}2\mathcal{O}(\phi)$ evaluates to 0 (the *zero set* of $\mathcal{L}2\mathcal{O}(\phi)$, 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 $\mathcal{L}2\mathcal{O}(\phi)$ has non-negative values, if $Z_\phi \neq \emptyset$, then Z_ϕ contains all and only the global minima of the function.

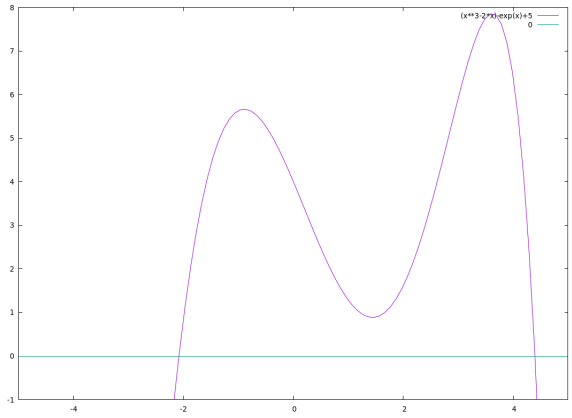

Figure 3.1 Graph of $f(x) \equiv x^3 - 2x - e^x + 5$

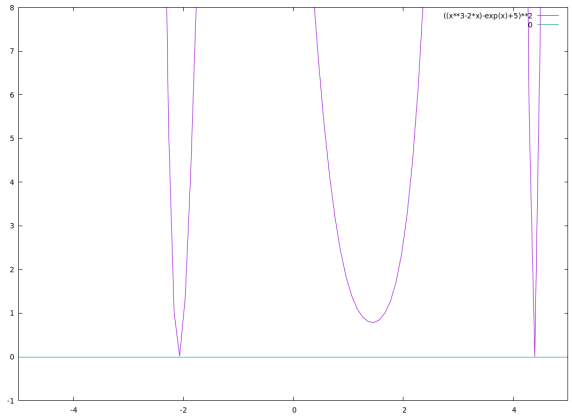

Figure 3.2 Graph of $\mathcal{L}2\mathcal{O}(f(x) = 0)$

Example 3.1.1. *Let's consider the formula $\phi \equiv f(x) = 0$, for $f(x) \equiv x^3 - 2x - e^x + 5$. From Figure 3.1, we see that $f(x)$ has two solutions, one near -2, and the other near 4. The non-negative real function $\mathcal{L}2\mathcal{O}(\phi)$, depicted in Figure 3.2, has three local minimum: two global minima, which correspond to the zeros of $f(x)$, and a third non-zero local minimum.*

We can exploit the Logic-to-Optimization approach as follows.

Through the unconstrained global optimisation algorithm Basin-hopping mentioned in §2.5, we obtain a finite set of local minima $L_\phi \subseteq \{p \in \mathbb{R}^m | p \text{ is a local minimum of } \mathcal{L}2\mathcal{O}(\phi)\}$. Implementation-wise, the output will consist of rational approximations of local minima. We denote this set by $\tilde{L}_\phi$. For each element $\tilde{p} \in \tilde{L}_\phi$, we try to produce a model p 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 §3.3.

Given $\tilde{p} \stackrel{\text{def}}{=} \{\tilde{p}_1, \cdots, \tilde{p}_m\} \in \tilde{L}_\phi$, it is trivial to check whether $\tilde{p}$ is a model for ϕ by substituting the variables with their values into the formula.[1] If $\tilde{p}$ is not a model we can try to look in the surroundings of $\tilde{p}$. 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 [30] equation (3), in the context of the incremental linearization approach (we will refer to this technique 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 $B \equiv I_1 \times \cdots \times I_m \subset \mathbb{R}^m$ (where

[1] We remind that, here, we are assuming to be in the $\mathcal{NRA}$ case.

$I_i \stackrel{\text{def}}{=} [a_i, b_i]$ and $\tilde{p}_i \in I_i$). A naive choice of B is the hyper-cube having $\tilde{p}$ as its center (that is, $I_i \stackrel{\text{def}}{=} [\tilde{p}_i - c, \tilde{p}_i + c]$ 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 $\tilde{p}$ 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 $\tilde{p}$ 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 $\mathcal{NTA}$.

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

In this section we explain how, given a local minimum $\tilde{p}$ 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 §3.2.1, 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 §3.2.2, 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 §3.2.3, we discuss how we can integrate these results within the Logic-to-Optimisation framework.

3.2.1 Quasi-decidability procedure

The procedure that we introduce in Algorithm 2 is inspired by that proposed in [48], 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 $G := g_1 \times \cdots \times g_k$.

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 27), provided that the inequalities hold in that sub-box (lines 20-23). 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 9).

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

Algorithm 2 QUASI-DEC

Input: A bounded formula $\phi_{|B}$ like (2.2) 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` $\leftarrow \{B\}$
2: `conflict_indices` $\leftarrow \{0, \cdots, m\}$
3: **while** `True` **do**
4: **for** $A \in$ `grid` **do**
5: **if** $(0 \notin \mathcal{IA}_F(A)) \vee (\mathcal{IA}_G(A) \cap (-\infty, 0]^k = \emptyset)$ **then** ▷ $\mathcal{IA}_F, \mathcal{IA}_G$ as in def. 2.2.1
6: `grid.remove`(A)
7: **end if**
8: **end for**
9: **if** `grid` $= \{\}$ **then return** <**False**>
10: **end if**
11: **if** $n \neq 0$ **then**
12: `grid` $\leftarrow$ Merge all the boxes in `grid` having a common face C s.t. $0 \in \mathcal{IA}_F(C)$
13: `grid`$_\partial \leftarrow \{A \in$ `grid` $\mid$ exists C a face of A s.t. $C \subseteq \partial B \wedge 0 \in \mathcal{IA}_{F(C)}\}$
14: **else**
15: `grid`$_\partial \leftarrow \{\}$
16: **end if**
17: **for** $A \in$ `grid` $\setminus$ `grid`$_\partial$ **do**
18: `conflict_indices_A` $\leftarrow \{\}$
19: `demerge`(A) := $\{E \mid E$ has been merged into A in line 12$\}$
20: **for** $E \in$ `demerge`(A) **do**
21: **for** $i \in \{0, \cdots, k\}$ **do**
22: **if** $\mathcal{IA}_{g_i}(E) \not\subset (-\infty, 0]$ **then**
23: `conflict_indices_A.add`$(\{j \in \{0, \cdots, m\} \mid x_j$ appears in $g_i\})$
24: **end if**
25: **end for**
26: **end for**
27: **if** `conflict_indices_A` $= \{\} \wedge (n = 0 \vee$ `TopDeg`$(F, A) \neq 0)$ **then**
28: **return** <**True** , A >
29: **end if**
30: `conflict_indices.add(conflict_indices_A)`
31: **end for**
32: `refinement_index` $\leftarrow$ Choose an index with the help of `conflict_indices`
33: `grid` $\leftarrow$ `refine(grid, refinement_index)` ▷ First, we demerge the grid; then, each sub-box is split in two sub-boxes along the axis `refinement_index`
34: `conflict_indices` $\leftarrow \emptyset$
35: **end while**

common face in which a zero lies (line 12) and avoiding boxes having a face contained on the border of B and in which lies a zero (line 13). 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 $\varepsilon > 0$, there is a finite number of iterations after which each sub-box A in the grid has* $\text{width}(A) < \varepsilon$". 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 23), and then we base our choice of the refinement index on that (line 33), 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 [48], 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.

3.2.2 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 $e \stackrel{\text{def}}{=} (g > 0)$ with the constraint $\hat{e} \stackrel{\text{def}}{=} (g - \varepsilon \geq 0)$, given a predefined constant $\varepsilon > 0$. 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

Algorithm 3 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 $\tilde{p} \in \mathbb{R}^m$
Output: <**True**, B_{sol}> or <**Unknown**>

1: $\hat{\phi}_{|B}$:= the formula obtained from $\phi_{|B}$ by replacing every $g > 0$ with $g - \varepsilon \geq 0$
2: **if** $n = m \vee n = 0$ **then**
3: `res_quasidec` $\leftarrow$ QUASI-DEC ($\hat{\phi}_{|B}$)
4: **if** `res_quasidec` $\equiv$ <**True**, B_{sol} > **then return** <**True**, B_{sol} >
5: **else return** <**Unknown**>
6: **end if**
7: **end if**
8: `infeasible_var_subsets` $\leftarrow \{\}$
9: **for** `vars_subset` $\in$ `Combinations`(Vars, $m-n$) **do**
10: **if** `vars_subset` $\in$ `infeasible_var_subsets` **then**
11: **continue**
12: **end if**
13: $\mu := \{x_i \mapsto p_i \mid x_i \in$ `vars_subset`$\}$
14: <`sat, r`> $\leftarrow$ QUASI-DEC ($\mu(\hat{\phi}_{|B})$)
15: **if** <`sat, r`> = <**True**, B_{sol} > **then**
16: **return** <**True**, B_{sol} >
17: **else if** <`sat, r`> = <**False**, $\mu(h)$ > **then**
18: `conflict_vars` := $\{x_i \in$ *vars_subset* $\mid x_i$ appears in $H\}$
19: `infeasible_var_subsets.add(conflict_vars)`
20: **end if**
21: **end for**
22: **return** <**Unknown**>

substitution. We start from a given point $\tilde{p} \in \mathbb{R}^m$, 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 2), to make it return, before the while cycle in line 3, <**False** ,f_i> if $0 \notin \mathcal{IA}_{f_i}(B)$, and <**False** , g_j> if $\mathcal{IA}_{g_j}(A) \cap (-\infty, 0] = \emptyset$. 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 work, 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 3.

3.2.3 A general procedure

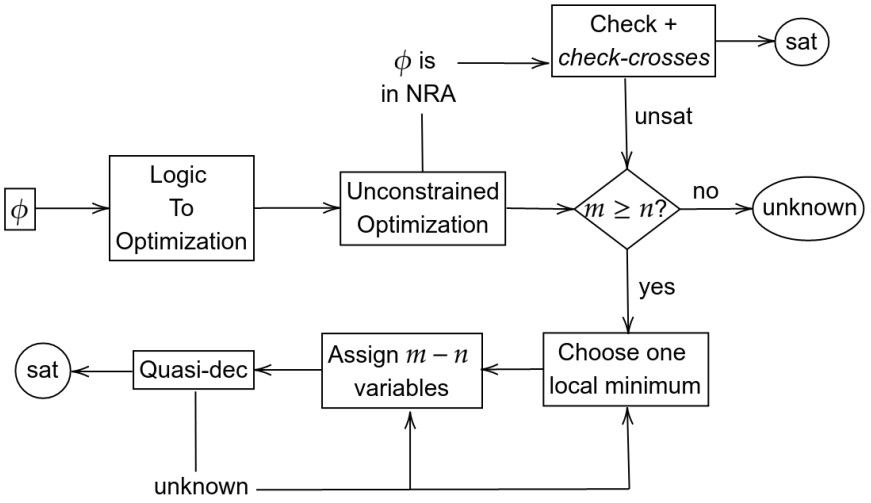

Figure 3.3 Schema of the overall procedure.

We can now combine the results of the last two sections. First, we obtain several local minima $\tilde{p}^1, \cdots, \tilde{p}^k$ as in Section 3.1 . The two tactics described in the section (i.e., the simple check of $\tilde{p}$, 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 3.2.2 to each local minimum (starting from the minimum with the lowest cost). A sketch of this procedure is shown in Figure 3.3.

Remark. 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 [48]).[2] This means that QUASI-DEC always terminates on unsatisfiable subformulas, guaranteeing that we always progress towards a solution.

[2]This is not true for formulas containing strict inequalities, but we replaced strict inequalities in Algorithm 3 at line 1.

3.3 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 §3.2. In the second way, we use the procedure of §3.2 as a theory solver inside a DPLL(T)-based lazy-SMT algorithm.

3.3.1 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 $\tilde{p}$, if there are no transcendental functions, we can use the two tactics discussed in section 3.1 (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 3.2, since QUASI-DEC does not work with disjunctions.

In order to apply it, we can try to decide the disjunctions using $\tilde{p}$, to obtain an implicant of ϕ which we can then feed to Algorithm 3. 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. Thus we try to prove, for each C_j, whether there is a constraint $h_i \bowtie 0$ such that $\tilde{p}$ satisfies it. For constraints in $\mathcal{NRA}$, a simple check suffices. For constraints in $\mathcal{NTA}$, we try to prove it by using interval arithmetic: we compute $I \stackrel{\text{def}}{=} \mathcal{IA}_{h_i}(\tilde{p})$ and check if I is contained in $(-\infty, 0]^n$ in case $\bowtie \equiv \leq$, or if I is equal to $[0]^n$ when $\bowtie \equiv =$. If we find one $h_i \bowtie 0$ satisfying this condition, then we replace C_j with $h_i \bowtie 0$ in $\mu(\hat{\phi}|_B)$.

If there is a C_j that we are not able to decide through this method, we reject the assignment μ. Note that we could reject also a feasible assignment: in fact, through interval arithmetic, we are testing a sufficient condition, not a necessary one. As a matter of fact, we are not interested in solving precisely an SMT problem, but rather in having a fast and uncomplicated way to try to decide the disjunction. We have observed that this strategy, although far from being a systematic approach, is often efficient in practice.

As a last resort, if for each local minimum and for each partial assignment we are unable to decide the disjunctions, then we rewrite the original formula into DNF and try to solve each constraints set as in the previous section.

3.3.2 A lazy approach

In the lazy approach, the method defined in §3.2 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 [30], which is usually effective in proving unsatisfiability, and also good in finding linear models, but whose weak spot is finding irrational models. In fact, for problems that only have irrational models, incremental linearization is stuck (except for rare occasions): neither it can produce a conflict, nor it can find a model, so that it continues indefinitely trying to guess linear models. Our new method comes into play when this is the case (or, better, when we assume this is the case).

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.

3.4 Experimental evaluation

3.4.1 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 $\text{UGOTNL}_{\text{EAGER}}$. For the lazy approach, we integrated UGOTNL as a theory solver inside the MATHSAT SMT solver [31]. We will refer to this version as MATHSAT+UGOTNL.

3.4.2 Setup.

We have run our experiments[3] 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 [37] and Yices[41] (CAD-based), raSAT [110] (that combines ICP and GIVT), CVC5 [23] (that combines incremental linearization with cylindrical algebraic coverings [2]), iSAT3 [50] (based on ICP), and dReal [54] (that operates in the δ-sat framework [53]). 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.

3.4.3 Benchmarks.

For $\mathcal{NRA}$, we consider all the SMT-LIB [10] 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 [54], 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.

3.4.4 Results.

Results (sat).

First, we analyze the results for satisfiable instances, which are reported in Figures 3.1 and 3.2. The tables show, for each solver, the number of 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 3.1), 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-

[3] Available at https://drive.google.com/file/d/1qUB9X-IymEozabBfrbTxwz6zu7tG62zH/

	Total	Sturm-MGC	ezsmt	Sturm-MBO	zankl	UltimateAut	Economics-M	meti-tarski	Heizmann	hycomp	kissing	LassoRanker	hong
MATHSAT	3193	0	**32**	0	32	31	85	2718	3	11	18	263	0
UGOTNL$_{\text{EAGER}}$	4388	0	0	**1**	52	32	68	4042	0	0	**36**	157	0
MATHSAT+UGOTNL	4441	0	**32**	0	63	27	84	3948	3	13	35	236	0
RASAT	4285	0	0	0	45	0	0	4225	0	0	15	0	0
YICES	4946	0	**32**	0	58	39	91	4369	0	227	10	120	0
CVC5	5108	0	**32**	0	63	36	89	4342	2	226	18	**300**	0
Z3	**5153**	**2**	30	0	**72**	**47**	**93**	**4391**	**6**	**280**	35	198	0
DREAL	/(5021)	/(9)	/(0)	/(274)	/(153)	/(55)	/(126)	/(4079)	/(51)	/(45)	/(19)	/(210)	0

Table 3.1 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)

Table 3.2 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.

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 3.2) 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.

	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**

Table 3.3 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

Table 3.4 Summary of results for SMT(NTA) unsat cases. On the left the original instances; on the right the bounded instances.

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 w.r.t. 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 (especially 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 3.2.3), 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 unsatisfiable 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.

3.5 Related work.

For $\mathcal{NRA}$, various techniques have been explored. Complete methods based on quantifier-elimination procedures such as Cylindric Algebraic Decomposition (CAD) [32] have been successfully implemented in several SMT-solvers (such as Z3 [37], YICES [41], SMT-RAT [34]), proving their effectiveness especially when tightly integrated into the Boolean search through a model-constructing framework such as MCSAT [64] [38]. 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 [30] 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 (especially in the presence of equations). ISAT3 [50] implements a method based on a tight integration of Interval Constraint Propagation (ICP) [12] 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 [110] combines ICP with the Generalized Intermediate Value Theorem (GIVT) [86], but does not support transcendental functions.

Other approaches, e.g. DREAL [54] and KSMT [18], rely on the notion of δ-satisfiability [53], which guarantees that there exists a perturbation (up to some $\delta > 0$ specified by the user) of

the original formula that is satisfiable.[4] 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.

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

Chapter 4

Satisfiability of SMT(NTA) as a Certificate Search problem

SAT modulo theories (SMT) is the problem of checking whether a given first-order formula with both propositional and theory variables is satisfiable in a specific first-order theory. In this work, we consider the quantifier-free theory of SMT($\mathcal{NTA}$), non-linear real arithmetic augmented with trigonometric and exponential transcendental functions. This problem is particularly important in the verification of hybrid systems and in theorem proving. Unfortunately, $\mathcal{NTA}$ is a very challenging theory. Indeed, it is undecidable [94], and, moreover, there is no known way of representing satisfying assignments using a finite string of bits (i.e., no finite representation of satisfying assignments) that could serve as a direct certificate of satisfiability. This does not only make it difficult for an SMT-solver to prove satisfiability, but also raises the question of how to verify the result given by an SMT-solver.

In this chapter, we

- introduce the notion of a satisfiability certificate for $\mathcal{NTA}$ that allows independent entities to verify the satisfiability of a given input formula without having to re-do a full check of its satisfiability,
- introduce a method for computing such satisfiability certificates,
- describe computational experiments that analyze the efficiency of several variants of the method, and
- provide a theoretical characterization of the class of problems for which such a certificate can be successfully computed despite the mentioned undecidability restrictions.

Based on the introduced certificate, the check of satisfiability of SMT($\mathcal{NTA}$) formulas is both easier in terms of computational effort and effort needed to implement the checker and to ensure its correctness. The certificate is based on the notion of topological degree [43, 1, 46], that generalizes the idea that a sign change of a continuous function f implies the satisfiability of the equation $f = 0$. The basic tool for checking correctness of the certificate is interval arithmetic [97, 86, 84].

The idea to verify satisfiability of SMT($\mathcal{NTA}$) in such a way, is not new [71]. However, the formulation as the problem of searching for a certificate is. In addition to the possibility of independent verification, such a formulation makes the corresponding search problem explicit. This allows us to introduce new, efficient search heuristics that guide the algorithm

toward finding a certificate and prevent the procedure from getting stuck in computation that later turns out to not to lead to success.

The experimental results are based on our implementation in the tool UGOTNL [71]. We compare different heuristic configurations over a wide variety of $\mathcal{NTA}$ benchmarks. The benchmarks also demonstrate that this new version of UGOTNL outperforms the previous version, making it—to the best of our knowledge—the most effective solver for proving satisfiability of $\mathcal{NTA}$ problems.

It is possible to integrate the resulting method into a conflict-driven clause learning (CDCL) type SMT solver [71]. However, in order to keep the focus of the work on the concern of certificate search, we ignore this possibility, here.

Content. This chapter is based on the work presented in [72], and it is organized as follows: in Section 4.1 we provide some further preliminaries, on top on the ones presented in Chapter 2; in Section 4.2 we give the formal definitions of *certifying SMT solver* and of *satisfiability certificate for SMT*($\mathcal{NTA}$); in Section 4.3 we outline our method for the certificate search; in Section 4.4 we illustrate the heuristics that we introduce in detail; in Section 4.5 we experimentally evaluate our method; finally, in Section 4.6, we discuss some related works.

4.1 Preliminaries

Dulmage–Mendelsohn decomposition. Given a system of equations ϕ, it is possible to construct an associated bipartite graph $\mathcal{G}_\phi$ that represents important structural properties of the system of equations. This graph has one vertex per equation, one vertex per variable, and an edge between a variable x_i and an equation $f_j = 0$ iff x_i appears in f. The Dulmage–Mendelsohn decomposition [40, 3] is a canonical decomposition from the field of matching theory that partitions the system into three parts: an over-constrained subsystem (more equalities than variables), an under-constrained subsystem (less equalities than variables), and a well-constrained subsystem (as many equalities as variables, and contains no over-constrained subsystem, i.e. it satisfies the Hall property [57]).

Example 4.1.1. *Let* $\phi := x - \tan(y) = 0 \wedge z^2 = 0 \wedge w = 0 \wedge \sin(w) = 0$*, as in Figure 4.1. Through the Dulmage–Mendelsohn decomposition we obtain:*

- *an under-constrained sub-system* $x - \tan(y) = 0$ *(two variables, one equation)*
- *a well-constrained sub-system* $z^2 = 0$ *(one variable, one equation)*
- *an over-constrained sub-system* $w = 0 \ \wedge \ \sin(w) = 0$ *(one variable, two equations)*

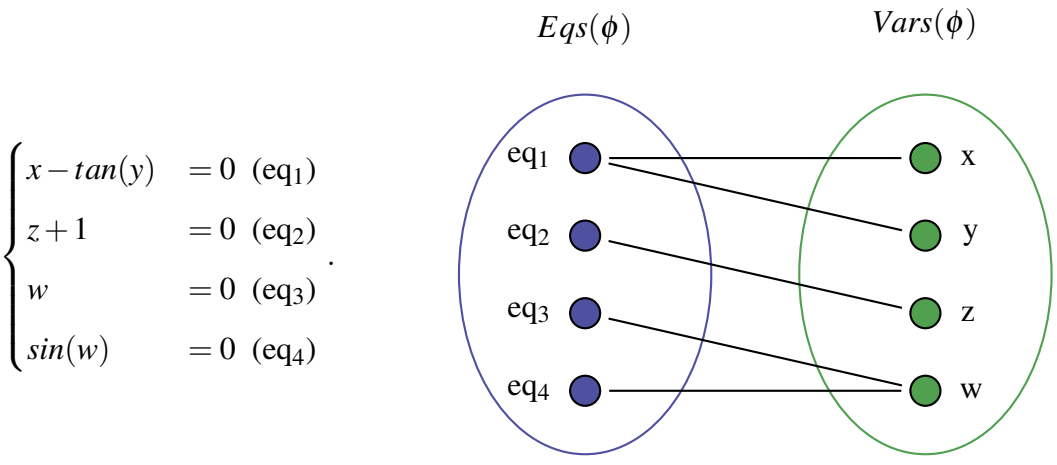

Figure 4.1 Example of Dulmage–Mendelsohn decomposition.

Robustness. Recall that we say that a formula ϕ is robust if there exists some $\varepsilon > 0$ such that either every formula that is the result of an ε-small perturbation of ϕ (i.e. a formula whose *distance* from ϕ is less than ε) is satisfiable, or either every such formula is unsatisfiable.

For example, the satisfiable formula $x^2 = 0$ is not robust, since for every $\varepsilon > 0$, the perturbed formula $x^2 + \varepsilon/2 = 0$ is unsatisfiable. In contrast to that, the satisfiable formula $x^3 = 0$ is robust. If ϕ is both robust and (un)satisfiable, we say that it is robustly (un)sat. Hence, $x^3 = 0$ is robustly sat.

For this chapter, an intuitive understanding of the notion of ε-small perturbation suffices. In Chapter 5, we will make this more precise for a more formal analysis of our approach. For more details, we refer the reader to the literature [48].

Relation between robustness and system of equations: An over-constrained system of equations is never robustly sat [48, Lemma 5]. It easily follows that a system of equations that contains an over-constrained sub-system (in the sense of the Dulmage–Mendelsohn decomposition) is never robustly sat as well.

Relation between robustness and topological degree: Even in the case of an isolated zero, the test for non-zero topological degree can fail if the system is non-robust. For example, the function $F(x) \equiv x^2$ has topological degree 0 in the interval $[-1, 1]$, although the equality $x^2 = 0$ has an isolated zero in this interval. It can be shown that the topological degree test is able to prove satisfiability in all robust cases for a natural formalization of the notion of robustness [48]. We will not provide such a formalization, here, but use robustness as an intuitive measure for the potential success when searching for a certificate.

Logic-To-Optimization. While symbolic methods usually struggle dealing with $\mathcal{NTA}$, numerical methods, albeit inexact, can handle transcendental functions efficiently. For this reason, an SMT solver can benefit from leveraging numerical techniques. In the Logic-To-Optimization approach [71, 51, 87], an SMT($\mathcal{NTA}$)-formula ϕ in m variables is translated into a real-valued non-negative function $\mathcal{L}2\mathcal{O}(\phi) \equiv H : \mathbb{R}^m \mapsto \mathbb{R}^{\geq 0}$ such that—up to a simple translation between Boolean and real values for Boolean variables—each model of ϕ is a zero of H (not necessarily vice-versa). When solving a satisfiability problem, one can try to first minimize this function through numerical methods, and then use the obtained numerical

(approximate) solution to prove, through exact methods, that the logical formula has indeed a model.

While for the complete definition of the $\mathcal{L}2\mathcal{O}$ operator we refer to Chapter 3, we now provide a simple example to recall through an intuition how the operator works. Given a formula of the form $F = 0$, we have that $\mathcal{L}2\mathcal{O}(F = 0) \equiv F^2$, i.e., for every x in the domain of F, $(\mathcal{L}2\mathcal{O}(F))(x) = F(x)^2$. For conjunctions, $\mathcal{L}2\mathcal{O}(F_1 \wedge F_2) \equiv \mathcal{L}2\mathcal{O}(F_1) + \mathcal{L}2\mathcal{O}(F_2)$. For disjunctions, $\mathcal{L}2\mathcal{O}(F_1 \vee F_2) \equiv \mathcal{L}2\mathcal{O}(F_1) * \mathcal{L}2\mathcal{O}(F_2)$

Now, consider for example $F_1, F_2 : \mathbb{R} \to \mathbb{R}$ defined by $F_1(x) \equiv x^2 - 2x$ and $F_2 \equiv \sin(x)$. The formula $F_1 = 0$ has exactly 2 solutions, $\{0, 2\}$, which are exactly the zeros (hence the global minima) of the non-negative function $F_1^2 : \mathbb{R} \to \mathbb{R}$, while the formula $F_2 = 0$ has infinitely many solutions $\{k\pi \mid k \in \mathbb{Z}\}$, which are exactly the zeros (and global minima) of $F_2^2 : \mathbb{R} \to \mathbb{R}$. Then, in order to find the solutions of $\phi \equiv (F_1 = 0 \wedge F_2 = 0)$ (which, in this case, consist of the singleton $\{0\}$), we can search for the zeros of $\mathcal{L}2\mathcal{O}(\phi)$.

4.2 Goal

Consider an SMT solver that takes as input some formula ϕ and as output an element of $\{\texttt{sat}, \texttt{unknown}, \texttt{unsat}\}$. How can we gain trust in the correctness of the result of such an SMT solver? One approach would be to ensure that the algorithm itself is correct. Another option is to provide a second algorithm whose output we compare with the original one. Both approaches are, however, very costly, and moreover, the latter approach still may be quite unreliable.

Instead, roughly following McConnell et al. [79] (see also Figure 4.2), we require our solver to return—in addition to its result—some information that makes an independent check of this result easy:

Definition 4.2.1. *An SMT solver is* certifying *iff there is a property W such that for every input formula ϕ, in addition to an element $r \in \{\texttt{sat}, \texttt{unknown}, \texttt{unsat}\}$, the solver returns an object w (a* certificate*) such that*

- *(ϕ, r, w) satisfies the property W, that is $W(\phi, r, w)$,*
- *$W(\phi, \texttt{sat}, w)$ implies that ϕ is satisfiable,*
- *$W(\phi, \texttt{unsat}, w)$ implies that ϕ is unsatisfiable, and*
- *there is an algorithm (a* certificate checker*) that*
  - *takes as input a triple (ϕ, r, w) and returns $\top$ iff $W(\phi, r, w)$, and that*
  - *is simpler than the SMT solver itself.*

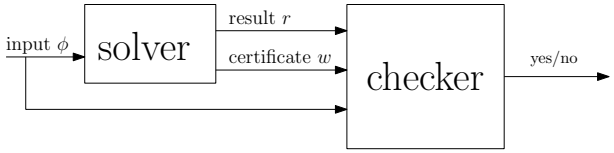

Figure 4.2 Certifying SMT Solver

So, for a given formula ϕ, one can ensure correctness of the result (r, w) of a certifying SMT solver by using a certificate checker to check the property $W(\phi, r, w)$. Since the certificate checker is simpler than the SMT solver itself, the correctness check is simpler than the computation of the result itself.

The definition leaves it open, what precisely is meant by "simpler". In general, it could either refer to the run-time of the checker, or to the effort needed for implementing the certificate checker and ensuring its correctness. The former approach is taken in computational complexity theory, the latter in contexts where correctness is the main concern [79]. Indeed, we will later see that our approach succeeds in satisfying both requirements, although we will not use complexity-theoretic measures of run-time, but will measure run-time experimentally.

The use of such certificates is ongoing research in the unsatisfiable case [9]. In the satisfiable case, for most theories, one can simply use satisfying assignments (i.e., witnesses) as certificates. Here the property W simply is the property that the given assignment satisfies the formula, which can be checked easily.

For SMT($\mathcal{NTA}$), however, the situation is different. In this case, satisfying assignments may involve numbers that are neither rational nor real algebraic. Indirect descriptions of satisfying assignments using formulations such as

- "the number that is the solution of the equation $\sin x = 1$", or
- "the number corresponding to the infinite sequence of digits that the Turing machine T writes onto its output tape"

also cannot be used as a basis for representing certificates. In the first case, the problem is that it is not obvious how to check whether the represented number actually exists. In the second case, it is not obvious how to check whether the represented number actually is a solution.

In general, adding such information to $\mathcal{NTA}$-formulas does not reduce the undecidable satisfiability problem to any known decidable class that would enable a certificate checker in the sense of Definition 4.2.1. Hence one needs to use certificates of a different form. For this, we introduce the following definition:

Definition 4.2.2. *Let ϕ be a formula in $\mathcal{NTA}$. A* (satisfiability) certificate *for ϕ is a triple (σ, ν, β) such that $W(\phi, \mathtt{sat}, (\sigma, \nu, \beta))$ iff*

- *σ is a function selecting a literal from every clause of ϕ*
- *ν is a variable assignment in $\mathcal{R}^V$ assigning floating point numbers to a subset $V \subseteq Vars_{\mathcal{R}}(\sigma(\phi))$ (where $\sigma(\phi)$ is a compact way of writing $\bigwedge_{C \in \phi} \sigma(C)$), s.t. $\sigma(\phi)$ contains as many equations as real-valued variables not in V.*

- β *is a finite set of interval assignments in* $\mathcal{B}^{Vars_{\mathcal{R}}(\phi)\setminus V}$ *such that their set-theoretic union as boxes is again a box* B_β *and, for the system of equations* $F := eq(\nu(\sigma(\phi)))$ *and the system of inequalities* $G := ineq(\nu(\sigma(\phi)))$*, it holds that:*
  - $0 \notin F(\partial B_\beta)$*,*
  - $\deg(F, B_\beta, 0) \neq 0$*, and*
  - *for every* $B \in \beta$*,* $\mathcal{IA}_G(B) \leq 0$*.*

Example 4.2.1. *Consider the formula*

$$\phi := C_1 \wedge C_2 \wedge C_3 \wedge C_4$$

$$C_1 \equiv \cos(y) = 0 \vee \sin(y) = e^x \qquad C_3 \equiv x - y \leq \cos(z)$$

$$C_2 \equiv \sin(y) = 0 \vee \cos(y) = \sin(8x^2 - z) \qquad C_4 \equiv x + y \geq \sin(z)$$

The following (σ, ν, β) *is a certificate:*

- $\sigma := \{C_1 \mapsto \sin(y) = e^x \ ; \ C_2 \mapsto \cos(y) = \sin(8x^2 - z) \ ; \ C_3 \mapsto C_3 \ ; \ C_4 \mapsto C_4\}$
- $\nu := \{z \mapsto 0.2\}$
- $\beta := \{B\}$*, where* $B := \{x \mapsto [-0.1, 0.05] \ ; \ y \mapsto [1.4, 1.9]\}$

As can be seen in Figure 4.3, the solution sets of C_1 and C_2 cross at a unique point in B, which reflects the fact that the degree of the function $(x,y) \to (\sin(y) - e^x, \cos(y) - \sin(8x^2 - 0.2))$ is non-zero. Moreover, the inequalities C_3 and C_4 hold on all elements of the box.

Due to the properties of the topological degree and of interval arithmetic discussed in the preliminaries, we have:

Property 4.2.1. $W(\phi, \texttt{sat}, (\sigma, \nu, \beta))$ *implies that* ϕ *is satisfiable.*

Such a satisfiability certificate can only serve as a certificate for an SMT solver if a certificate checker—as required by Definition 4.2.1—exists. This certificate checker needs to be able to check the conditions of Definition 4.2.2. The topological degree can indeed be computed algorithmically [1, 46]. The condition $0 \notin F(\partial B_\beta)$ is necessary for the topological

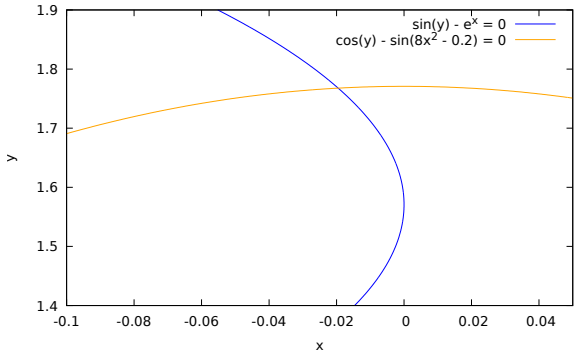

Figure 4.3 Solution sets of equalities for the certificate of Example 4.2.1

degree to be well defined. Due to this, such algorithms [1, 46] also check this condition, and no separate check is necessary. Finally, the condition $\mathcal{IA}_G(B) \leq 0$ clearly is algorithmic.

Note that this definition could be applied to a broader class of formulas than $\mathcal{NTA}$, since the algorithms used to compute the topological degree and $\mathcal{IA}$ can work with any *interval-computable* function (i.e. functions for which it is possible to compute arbitrarily precise images for every interval domain [48]). However, in practice, the tools that implement these algorithms do not go beyond the $\mathcal{NTA}$ case. For this reason, given that our approach is application-oriented, we will keep our focus only on $\mathcal{NTA}$.

We will show that in addition to the discussed benefits for correctness, formulating satisfiability checking as the problem of search for such certificates also is beneficial for efficiency of the SMT solver itself. Since we will concentrate on satisfiability, we will simply ignore the case when an SMT solver returns `unsat`, so the reader can simply assume that an SMT solver such as the one from Figure 4.2 only returns an element from the set $\{\texttt{sat}, \texttt{unknown}\}$.

Note that the variable assignment v can also be viewed as a system of equalities of the form $\bigwedge_{v \in V} v = v(v)$. In general, one could allow a system of equalities of a more general form, for example, a system of linear equalities, arriving at the function F and G by eliminating some of the involved variables by Gaussian elimination. One could even extend the functions F and G by the left-hand sides of those additional equalities. However, our computational experiments will demonstrate that this is not beneficial, in general, which justifies the more specific form of Definition 4.2.2.

4.3 Method

Our goal is to find a triple (σ, ν, β) that is a certificate of satisfiability for a given formula ϕ. So we have a search problem. In order to make this search as efficient as possible, we want to guide the search toward a triple that indeed turns out to be a certificate, and for which the corresponding conditions are computationally easy to check.

Intuitively, we view the search for a certificate as a hierarchy of nested search problems, where the levels of this hierarchy correspond to the individual components of certificates. We formalize this using a search tree whose nodes on the i-th level are labeled with i-tuples containing the first i elements of the tuple searched for, starting with the root note that is labeled with the empty tuple $()$. The tree will be spanned by a function *ch* that assigns to each node $(c_1, \ldots, c_i)$ of the tree a sequence $\langle x_1, \ldots, x_n \rangle$ of possible choices for the next tuple component. Hence the children of $(c_1, \ldots, c_i)$ in the tree are $(c_1, \ldots, c_i, x_1), \ldots, (c_1, \ldots, c_i, x_n)$. We will do depth-first search in the resulting tree, searching for a leaf labeled by a certificate of satisfiability for the input formula ϕ.

Based on the observation that on each level of the tree one has the first i components of the tuple available for determining a good sequence of choices, we will add additional information as the first tuple component in the form of a variable assignment p that satisfies the formula ϕ approximately. Hence we search for a four-tuple (p, σ, ν, β).

It is easy to see that it would be possible to generalize such a search tree to a more fine-grained one, where the individual levels are formed by parts of the choices described above, and where the order of those levels can be arbitrary. For example, it would be possible to first choose an interval for a variable (i.e., part of the box β), then select a literal from a certain clause (i.e., part of the selection function σ), and so on. However, here, we keep these levels separated, as discussed above, in order to achieve a clear separation of concerns when exploring design choices at the individual levels.

4.4 Certificate Search

In this section, we will discuss possibilities for search strategies by defining for every search tree node labeled with tuple τ, the ordered sequence $ch(\tau)$ of choices for the next tuple element. Our framework allows for many more possibilities from which we choose strategies that both demonstrate the applicability of the framework to different search strategies, and allow for efficient search, as will be demonstrated by the computational experiments in Section 4.5.

In order to be able to refer to different variants of the search strategy in the description of computational experiments, we will introduce keywords for those variants that we will write in teletype font.

Here we will focus on strategies of the two following basic types:

- Filtering: We skip elements from the set of possible choices that cannot result in a certificate or for which the probability of resulting in a certificate is negligible.
- Ordering: We choose elements from the set of possible choices in a certain order that tries the reflect the probability of resulting in a certificate.

4.4.1 Points

The points $ch() = \langle p_1, \ldots, p_k \rangle$ determining the first level of the search tree are generated by an optimization problem defined on the formula ϕ following the Logic-To-Optimization approach [71]. Here we translate the satisfiability problem into a numerical minimization problem, mapping the logic formula ϕ into the non-negative real-valued function $\mathcal{L}2\mathcal{O}(\phi) \equiv H : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ (called the *objective function*) such that for every satisfying assignment, this objective function is zero, and for assignments that do not satisfy the formula, the objective function is typically (but not always) non-zero. Then we find local minima of H through an unconstrained optimization algorithm such as basin hopping [111], a two-phases Monte Carlo Markov Chain method that alternates local minimization with random jumps. In our implementation, we compute $k = 100$ local minima, and process them in the order of their value.

4.4.2 Literals

Given a point p, we now discuss how choose literal selector functions $ch(p) = \langle \sigma_1, \ldots, \sigma_k \rangle$. For filtering the set of literal selector functions, we will restrict ourselves, for each clause $C \in \phi$, to the literals l for which the objective function restricted to l and evaluated in the point p is below a certain threshold. That is, we determine the set of approximately satisfiable literals

$$L_C := \{l \in C \mid \mathcal{L}2\mathcal{O}(l)(p) \leq \varepsilon\}.$$

Our literal selector functions will then correspond to the set of all approximately satisfiable combinations

$$\{\sigma \mid \text{for all } C \in \phi, \sigma(C) \in L_C\},$$

that is, each σ selects exactly one approximately satisfiable literal from each clause. In order to maximize the chances of choosing a better literal combination, we can sort the elements of L_C according to the value of the respective objective functions and then choose literal combinations using the corresponding lexicographic order (we will refer to this heuristic as `(sort-literals)`).

While the point p is usually a good candidate in terms of *distance from a zero*, it can sometimes lead to an inconsistent problem:

Example 4.4.1. *Consider the formula*

$$\phi := C_1 \wedge C_2$$
$$C_1 \equiv (x+y=0) \vee (x = e^{10^6 * y}) \qquad C_2 \equiv (x+y \geq \varepsilon_1) \vee (x = \tan(y+\varepsilon_1))$$

The numerical optimizer will be tempted to return first some point p_1 such as $\{x \mapsto 1; y \mapsto -1\}$, that almost *satisfies $(x+y=0) \wedge (x+y \geq \varepsilon_1)$, instead of a harder approximate solution involving transcendental functions and heavy approximations, such as $(x = e^{10^6 * y}) \wedge (x = \tan(y+\varepsilon_1))$, that is exactly satisfiable in a point p_2 near $(0, -\pi)$.*

Such inconsistencies may occur in many combinations of literals. We use a strategy that detects them in situations where for certain clauses C, the set L_C contains only one literal l. We will call such a literal l a *forced literal*, since, for every literal selector function σ, $\sigma(\phi)$ will include l. Before starting to tackle every approximately satisfiable literal combination, we first analyze the set of forced literals. We do symbolic simplifications (such as rewriting and Gaussian elimination) to check whether the set has inconsistencies that can be found

at a symbolic level (as in the previous example). If the symbolic simplifications detect that the forced literals are inconsistent then we set $ch(p)$ to the empty sequence $\langle\rangle$ which causes backtracking in depth-first search. We refer to the variant of the algorithm using this check as `(check-forced-literals)`.

Filtering out over-constrained systems. Given a literal selector function σ, we analyze the structure of the system of equations formed by the equations selected by σ through the Dulmage–Mendelsohn decomposition, that uniquely decomposes the system into a well-constrained subsystem, an over-constrained subsystem and an under-constrained subsystem. We filter out every literal combination having a non-empty over-constrained subsystem, since this leads to a non-robust sub-problem, referring to this heuristic as `(filter-overconstr)`.

4.4.3 Instantiations

We define the instantiations $ch(p,\sigma) = \langle v_1,\ldots,v_k\rangle$ based on a sequence of sets of variables $V_1,\ldots,V_k$ to instantiate, and define $v_i := proj_{V_i}(p)$. The uninstantiated part of p after projection to a set of variables V_i is then $proj_{Vars_{\mathcal{R}}(\phi)\setminus V_i}(p)$, which we will denote by $p_{\neg V_i}$.

For searching for the variables to instantiate, we use the Dulmage–Mendelsohn decomposition constructed in the previous level of the hierarchy. We do not want to instantiate variables appearing in the well-constrained sub-system, since doing so would make the resulting system after the instantiation over-constrained. Hence the variables to be instantiated should be chosen only from the variables occurring in the under-constrained subsystem. This substantially reduces the number of variable combinations that we can try. Denoting the variables satisfying this criterion by V_{under}, this restricts $V_i \subseteq V_{under}$, for all $i \in \{1,\ldots,k\}$. This does not yet guarantee that every chosen variable combination leads to a well-constrained system after the instantiation. For example , the under-determined system of equations $x+y=0 \wedge z+w=0$ has four variables and two equations, but becomes over-constrained after instantiating either the two variables x and y, or the variables z and w. So, for each V_i, we further check whether the system obtained after the instantiation is well-constrained (we refer to this heuristic as `(filter-overconstr-V)`).

The method described in the previous paragraph only uses information about which equations in the system contain which variables (i.e., it deals only with the *structure* of the system, not with its *content*). Indeed, it ignores the point p.

To extract more information, we use the following fact: If a zero of a function has non-singular Jacobian matrix, then every box containing this zero and no other zeros has a non-zero topological degree [43]. So we compute a floating point approximation of the Jacobian matrix at point p (note that, in general, this matrix is non-square). Our goal is to find a set of variables V to instantiate such that the Jacobian matrix corresponding to the resulting square system at the point $p_{\neg V}$ has full rank. This matrix is the square sub-matrix of the original Jacobian matrix that is the result of removing the instantiated columns.

A straight-forward way of applying the Jacobian criterion is, given random variable instantiations, to filter out instantiations whose corresponding Jacobian matrix is rank-deficient (`filter-rank-deficient`), similarly to what is done in the previous paragraph with the overconstrained filter. Note that, as the Jacobian matrix of non-well-constrained system of equations is always rank-deficient, this filter is stronger than the previous one. However, it may filter out variable instantiations that result in a non-zero degree (e.g., the function x^3 has non-zero degree in $[-1,1]$, but its Jacobian matrix at the origin is rank deficient since $f'(0) = 0$).

We can further use the information given by the Jacobian matrix not only to filter out bad variable instantiations, but also to maximize the chance of choosing good variable instantiations from the beginning. Indeed, not all variable instantiations will be equally promising, and it makes sense to head for an instantiation such that the resulting square matrix not only has full rank, but—in addition—is far from being rank-deficient (i.e., it is as robust as possible). We can do so by modifying Kearfott's method [65, Method 2], which fixes the coordinates most tangential to the orthogonal hyperplane of F in p by first computing an approximate basis of the null space of the Jacobian matrix in the point, and then choosing the variables corresponding to the coordinates for which the sum of the absolute values in the basis is maximal. Since we are interested in more than just a single variable choice, we order all the variables w.r.t. to this sum. Then, we extract the sets of variables $V_1, V_2, \ldots$ through a lexicographic combinatorial algorithm. We refer to this heuristic as `(Kearfott-ordering)`.

Adding equations. An alternative approach for reducing from an underconstrained system of equations to a square one is, instead of instantiating variables, to add equations. This approach is justified by the fact that each variable instantiation can be seen as a system of equalities (while the vice-versa is not true).

While discussing Kearfott's method, we showed that, given a point p, it is better to choose variable instantiations that are the most orthogonal possible to the tangent hyperplane of F in p. With the equations adding approach we can go further: we can directly choose the linear equations that describe the hyperplane orthogonal to the tangent space of F in p. These equations can be found through the QR-decomposition of the Jacobian matrix of F in p. We can then add these equations to F in order to obtain a square system of equations. We refer to this heuristic as `(orthogonal)`.

Since the found equations are linear, we can further modify the previous heuristic by applying Gaussian elimination to the linear part of the square system obtained, thus reducing the dimension of the system of equations. We refer to this sub-heuristic as `(gauss-elim)`.

4.4.4 Boxes

We construct boxes around $p_{\neg V}$, where V is the set of variables v instantiates, that is, $v \in \mathcal{R}^V$. So we define $ch(p,\sigma,v) := \langle \beta_1,\ldots,\beta_k \rangle$ s.t. for all $i \in \{1,\ldots,k\}$, for all $B \in \beta_i$, $B \in \mathcal{B}^{Vars_{\mathcal{R}}(\phi)-V}$ and $p_{\neg V} \in \bigcup_{B\in\beta_i} B$.

We use two different methods, `(eps-inflation)` and `(box-gridding)`:

- Epsilon-inflation [76] is a method to construct incrementally larger boxes around a point. In this case, the $\beta_1,\ldots,\beta_k$ will each just contain one single box B_i defined as the box centered at $p_{\neg V}$ having side length $2^i\varepsilon$, where, in our setting, $\varepsilon = 10^{-20}$. We terminate the iteration if either $\mathcal{IA}_G(B_i) \leq 0$ and $\deg(F,B_i,0) \neq 0$, in which case we found a certificate, or we reach an iteration limit (in our setting when $2^i\varepsilon > 1$).

- Box-gridding is a well-known technique from the field of interval arithmetic based on iteratively refining a starting box into smaller sub-boxes. Here we use a specific version, first proposed in [48] and then implemented with some changes in [71]. In the following we roughly outline the idea behind the algorithm, and refer to the other two papers for details. We start with a grid that initially contains a starting box (in our setting, having side length 1). We then iteratively refine the grid by splitting the starting box into smaller sub-boxes. At each step, for each sub-box B we first check whether interval arithmetic can prove that the inequalities or the equations are unsatisfiable, and, if so, we remove B from the grid. We check also whether $\deg(F,B,0) \neq 0$ and interval arithmetic can prove the satisfiability of the inequalities, and, if so, then we terminate our search, finding a certificate with the singleton $\beta_i = \{B\}$. In some cases,

in order to verify the satisfiability of the inequalities, we will have to further split the box B into sub-boxes, using the set of resulting sub-boxes instead of the singleton $\{B\}$. After each step, if there are sub-boxes left in the grid, we continue the refinement process. Otherwise, if the grid is empty, we conclude that there cannot be solutions in the starting box. If a certain limit to the grid size is exceeded, we also stop the box gridding procedure without success.

For both methods, if the method stops without success, we have arrived at the last element of the sequence of choices $\langle \beta_1, \ldots, \beta_k \rangle$ without finding a certificate, which results in backtracking of the depth-first search for a certificate.

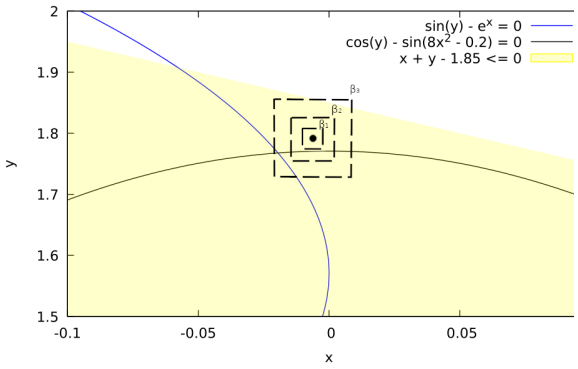

Figure 4.4 Example of ε-*inflation*.

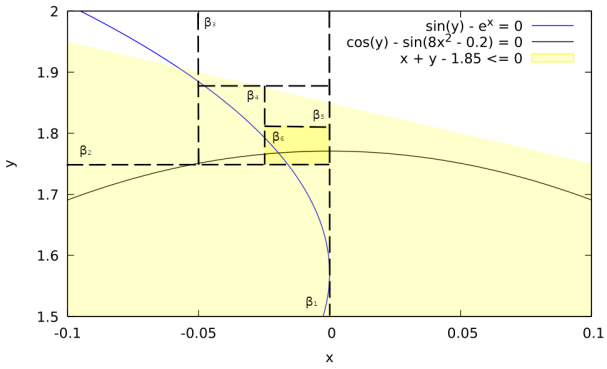

Figure 4.5 Example of *box-gridding*

Example 4.4.2. *Consider a small variation of the Example 4.2.1.*

Figure 4.4 shows an instance of the ε-inflation method. Here, larger and larger boxes $\beta_1, \beta_2, \beta_3$ are constructed around a starting point p. In this example, the method does not succeed, since the box β_2 does not include a solution, and the following box β_3 does not satisfy the inequality anymore.

Figure 4.5 shows an instance of the box-gridding method. Here, the box is iteratively into sub-boxes, and sub-boxes that cannot contain a solution are discarded. The first splitting is the vertical dashed line in the center, which divides the starting box into two sub-boxes. The sub-box on the right, cannot contain a solution for the equations (it does not intersect the blue line) hence it is discarded. The sub-box on the left, β_1, does indeed contain a solution for the equations (that is proved by the topological degree test), but the inequality is not satisfied everywhere in the box. Further splitting are performed, until a satisfying sub-box β_6 is found.

Both mentioned methods have their advantages, and can be seen as complementary. Epsilon-inflation is quite fast, and performs particularly well if the solution is isolated and is near the center. However, if there are multiple solutions in a box, the topological degree test can potentially fail to detect them[1], and if the solution is far from the center then we need a bigger box to encompass it, which is less likely to be successful than a smaller box, as we require the inequalities to hold everywhere in the box, and, moreover, the chance of encompassing other solutions (thus incurring in the previous problem) grows.

The box-gridding procedure, on the other side, can be quite slow, as in the worst case the number of sub-boxes explodes exponentially. However, grid refinement leads to a very accurate box search, which allows us to avoid the issues faced with epsilon inflation (i.e. multiple solutions, or a solution far from the center). Moreover, if the problem is robust, we have the theoretical guarantee that the procedure will eventually converge to a solution [48], although this does not hold in practice due to the introduced stopping criterion.

Indeed, a third approach is to combine the two methods: first use epsilon inflation, that is often able to quickly find a successful box, and, if it fails, then use the more accurate box-gridding procedure.

[1]For example, for $f(x) = x^2 - 1$, $deg(f, [-10, 10], 0) = 0$, while $deg(f, [-10, 0], 0) = -1$, and $deg(f, [0, 10], 0) = 1$.

4.5 Computational Experiments

Implementation. We implemented the different heuristics presented in the prototype tool UGOTNL (firstly presented in [71]). In order to make the results comparable with the ones obtained earlier, in addition to the search method discussed in Section 4.4, we preserve the following heuristics used by UGOTNL: If the local minimizer cannot find any minimum of $\mathcal{L}2\mathcal{O}(\phi)$ for which for every clause $C \in \phi$, the set of approximately satisfiable literals L_C is non-empty, we restart the procedure on every conjunction resulting from the DNF of ϕ. The tool handles strict inequalities of the form $f < 0$ directly until the box construction phase, where they are replaced by $f \leq -\varepsilon$ (with $\varepsilon = 10^{-20}$). For computing the topological degree, we use TOPDEG[2]. For the symbolic simplifications used in `(check-forced-literals)`, we use the *simplify* and the *solve-eqs* tactics provided by Z3 [37] [3]. For the computation of the rank used in `(filter-rank-deficient)`, we observe that the rank of a matrix is equal to the number of non-zero singular values, hence we consider a matrix far from rank-deficiency iff all its singular values are bigger than some threshold (to account for approximation errors). We use a threshold widely used by algorithms for determining the matrix rank, which is $\sigma_{\max} dim(A)\varepsilon$, where $\sigma_{\max}$ is the largest singular value of A, and ε is the machine epsilon.

Setup. We run the experiments[4] on a cluster of identical machines equipped with 2.6GHz AMD Opteron 6238 processors. We set a time limit of 1000 seconds, and a memory limit of 2Gb. We considered all SMT($\mathcal{NTA}$) benchmarks from the dReal distribution [54] and other SMT($\mathcal{NTA}$) benchmarks coming from the discretization of Bounded Model Checking of hybrid automata [95, 6], totaling 1931 benchmarks. All of these benchmarks come with "unknown" status. According to experiments performed on other solvers (CVC5, DREAL, ISAT3, MATHSAT), among these benchmarks 736 (respectively, 174) are claimed to be unsatisfiable (satisfiable) by at least one solver[5]. We tested our tool with different heuristics configurations (Table 4.1), and, for each configuration, we checked that our tool never contradicts the other tools. We have arranged the heuristics into 3 columns (Literals, Instantiations, and Boxes) according to the search level they are used in. As the number

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

[3] For a description of the two tactics: https://microsoft.github.io/z3guide/docs/strategies/summary. The version of Z3 used is 4.5.1.0.

[4] The results of the experiments are available at https://doi.org/10.5281/zenodo.7774117

[5] For the results of such experiments, see [71].

	Heuristics			(id.)
N. solved	Literals	Instantiations	Boxes	
323			(box-gridding)	(1.a.)
355			(eps-inflation)	(1.b.)
356			(eps-inflation) (box-gridding)	(1.c.)
362	(sort-literals)		(eps-inflation)	(2.b.)
361	(sort-literals)		(eps-inflation) (box-gridding)	(2.c.)
370	(sort-literals) (filter-overconstr)		(eps-inflation)	(3.b.)
367	(sort-literals) (filter-overconstr)		(eps-inflation) (box-gridding)	(3.c.)
406	(sort-literals) (filter-overconstr) (check-forced-literals)		(eps-inflation)	(4.b.)
410	(sort-literals) (filter-overconstr) (check-forced-literals)		(eps-inflation) (box-gridding)	(4.c.)
409	(sort-literals) (filter-overconstr) (check-forced-literals)	(Kearfott-ordering)	(eps-inflation)	(5.b.)
412	(sort-literals) (filter-overconstr) (check-forced-literals)	(Kearfott-ordering)	(eps-inflation) (box-gridding)	(5.c.)
424	(sort-literals) (filter-overconstr) (check-forced-literals)	(Kearfott-ordering) (filter-overconstr-V)	(eps-inflation)	(6.b.)
426	(sort-literals) (filter-overconstr) (check-forced-literals)	(Kearfott-ordering) (filter-overconstr-V)	(eps-inflation) (box-gridding)	(6.c.)
427	(sort-literals) (filter-overconstr) (check-forced-literals)	(Kearfott-ordering) (filter-overconstr-V) (filter-rank-deficient)	(eps-inflation)	(7.b.)
426	(sort-literals) (filter-overconstr) (check-forced-literals)	(Kearfott-ordering) (filter-overconstr-V) (filter-rank-deficient)	(eps-inflation) (box-gridding)	(7.c.)
441	Virtual best			

Table 4.1 Summary of the results for different heuristics configurations. Each row correspond to a configuration. The first column from the left contains the number of benchmarks solved; the central columns indicate the heuristics used, separated by search level; the last column contains an identifier of the configuration. The last row is for the virtual best of the different configurations.

of possible configurations is quite high, we proceed as follows: We start with the simpler configurations (just one method for finding a box that contains a solution), and then we add heuristics. The configurations that use the equation adding method are discussed separately at the end of the section.

Results. In the first configurations we tested the 3 possible ways to search for a box. We note that `(box-gridding)` (1.a.) performs considerably worse than the other two, `(eps-inflation)` (1.b.) and `(eps-inflation) +(box-gridding)` (1.c.), which produce comparable results. Because of that, and for readability's sake, we did not use `(box-gridding)` alone with other heuristics in the next configurations, but only considered the other two options. We then added heuristics based on the following criteria: first heuristics for the "Literals" choice, then heuristics for the "Instantiations" choice, and first ordering heuristics (i.e. `(sort-literals)` and `(Kearfott-ordering)`), then filtering heuristics (all the others). At every new heuristic added, we see that the number of benchmarks solved grows regardless of the "Boxes" choice, with the best configuration reaching 427 benchmarks

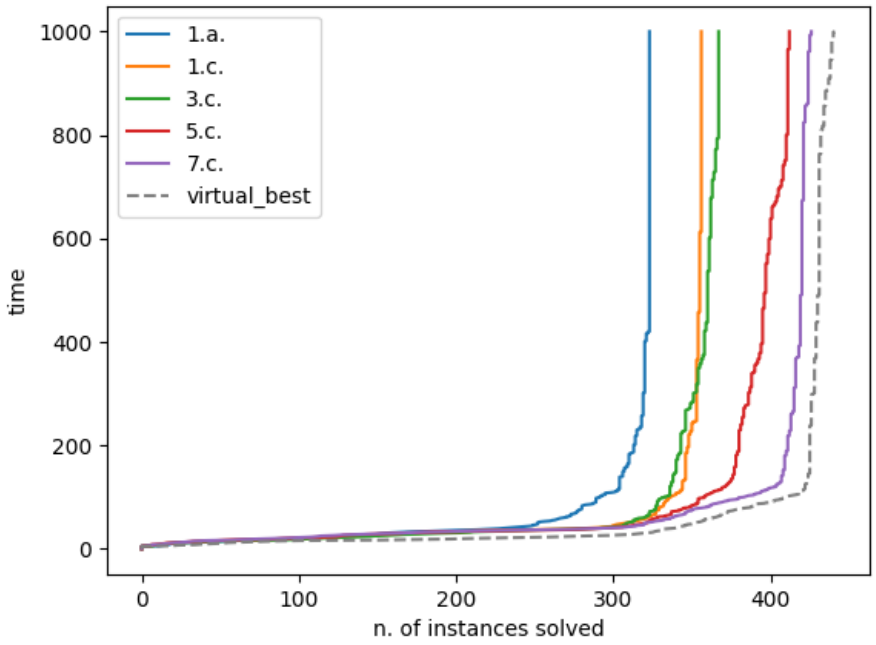

Figure 4.6 Survival plots for some of the configurations presented in Table 4.1. For each configuration, the plot shows the number of instances solved (x axis) within the given time (y axis).

using 7 heuristics. If we consider the virtual best (i.e. run in parallel all the configurations and stop as soon as a certificate is found) we are able to solve 441 benchmarks. This is because in cases such as `(eps-inflation)` vs. `(eps-inflation) +(box-gridding)`, or such as `(filter-overconstr-V)` vs. `(filter-rank-deficient)`, there is no dominant choice, with each configuration solving benchmarks that the other does not solve and vice-versa. The cactus plot in Figure 4.6—in which we included only a subset of the configurations in Table 4.1 for graphical reasons— substantiates the claim that new heuristics improve not only effectiveness, but also performances. The plot of the virtual best remarks the complementarity between different configurations.

Discussion. The first configuration (1.a.) essentially uses the method proposed in Chapter 4 and implemented in UGOTNL$_{\text{EAGER}}$ (of which the tool presented discussed is an upgrade). Already in the previous paper, UGOTNL$_{\text{EAGER}}$ outperformed the other solvers able to prove

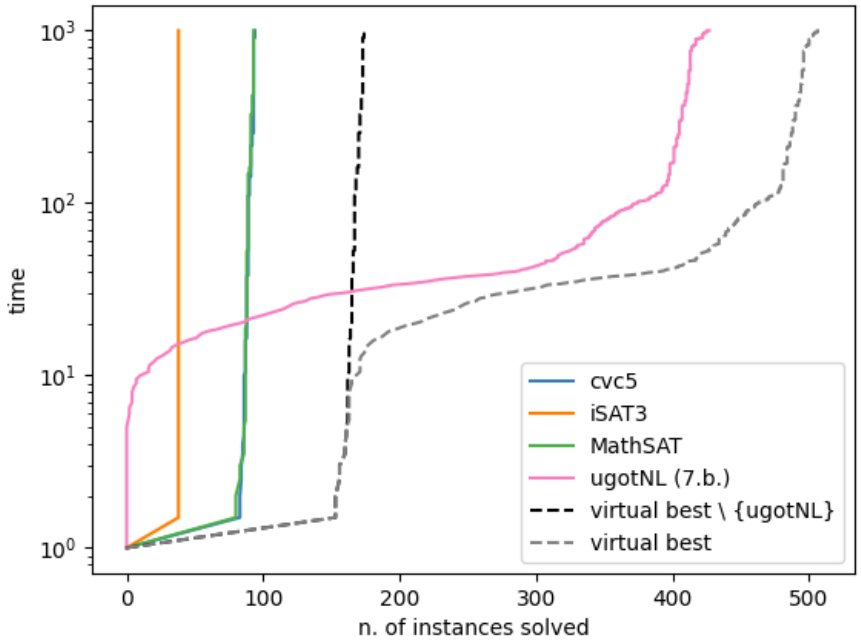

Figure 4.7 Survival plots for the best configuration of UGOTNL ((7.b.)) compared with the other state-of-the-art SMT solvers. For each solver, the plot shows the number of instances solved (x axis) within the given time (y axis). Note that the plots of CVC5 and MATHSAT are extremely close to each other (this is not surprising, as both tool use the same technique, incremental linearization).

satisfiability in SMT($\mathcal{NTA}$), solving more than three times the benchmarks than MATHSAT [31], CVC5 [67], and ISAT3 [50], and almost as twice as the benchmarks solved by the *lazy* version MATHSAT+UGOTNL (where UGOTNL had been integrated *lazily* inside MATHSAT). The introduction of new heuristics further improved the performances of our tool, that is now able to solve around 100 benchmarks more. Moreover, the best configuration of our tool, (7.b.), was able to prove the satisfiability of 334 benchmarks among the 1021 that had status "unknown", i.e. that had not been solved by any other solver. Now, considering all the state-of-the-art SMT solvers that are able to prove satisfiability of $\mathcal{NTA}$ benchmarks, including ours, the virtual best is able to solve 508 benchmarks, compared to the 174 previously solved without the inclusion of our tool. This is depicted in Figure 4.7, which also shows the complementarity between the different solvers. As can be noted, the plot of UGOTNL is less steep than the plots of other solvers. This is in part due to the eager approach of the tool, that first generates several candidate points, and only then chooses one of these points to narrow down the search, in part due to the tool been a prototype written in Python, compared to highly optimized solvers written in C/C++.

Run-time of the certificate checker. In Section 4.2 we claimed that, with our approach, checking a certificate requires less run-time than the certificate search itself. Here we experimentally quantify this amount: for each benchmark solved by the best configuration (7.b.), we observe the run-time required to check the certificate (which amounts, essentially, to the computation of topological degree and interval arithmetic for the successful box). In terms of median (respectively, mean), checking the certificate requires 0.10% (1.07%) of the run-time used by the solver.

Variable instantiations vs. equation adding In Section 4.4.3 we presented two different approach to reduce to a square system of equations: variable instantiation and equation adding. In Table 4.2 we experimentally compare these two approaches. In order not to overload the table, we compare the new heuristics only against representative configurations: (7.b.) - being the one that solved more benchmarks - and (4.b.) - being the one where only "Literals" heuristics are used. We observe that the use of `(orthogonal)` does not seem to pay off particularly well: comparing (4.b.) to (orth.1.), it increases the number of benchmarks solved just by a small margin, while comparing (7.b.) to (orth.3.), the number of benchmarks

	Heuristics			(id.)
N. solved	Literals	Instantiations	Boxes	
406	(sort-literals) (filter-overconstr) (check-forced-literals)		(eps-inflation)	(4.b.)
409	(sort-literals) (filter-overconstr) (check-forced-literals)	(orthogonal)	(eps-inflation)	(orth.1.)
399	(sort-literals) (filter-overconstr) (check-forced-literals)	(orthogonal) (gauss-elim)	(eps-inflation)	(orth.2.)
427	(sort-literals) (filter-overconstr) (check-forced-literals)	(Kearfott-ordering) (filter-overconstr-V) (filter-rank-deficient)	(eps-inflation)	(7.b.)
419	(sort-literals) (filter-overconstr) (check-forced-literals)	(orthogonal) (Kearfott-ordering) (filter-overconstr-V) (filter-rank-deficient)	(eps-inflation)	(orth.3.)
413	(sort-literals) (filter-overconstr) (check-forced-literals)	(orthogonal) (gauss-elim) (Kearfott-ordering) (filter-overconstr-V) (filter-rank-deficient)	(eps-inflation)	(orth.4.)
443	Virtual best			

Table 4.2 Summary of the heuristics configurations using the equation adding approach. Two configurations from from Table 4.1 are included for comparison. The last row is the virtual best considering all the configurations.

solved decreases. If we consider the configurations that use the sub-heuristic `(gauss-elim)` - (orth.2.) and (orth.4.) - we see that in both cases the performance are even worsened.

Discussion. While from a mathematical perspective the orthogonal method should perform better, as it yields a more robust system and obviates the iteration through all the possible variable instantiations, its effectiveness can be severely limited by the well-known numerical instability of algorithms that compute orthogonal matrices. Moreover, the systems of equations obtained via equation adding have a higher dimension than the ones obtained via variable instantiations, leading to a more complex problem to solve for the topological degree test and interval arithmetic. One could hope to lighten this negative effect by using the sub-heuristic `(gauss-elim)` that reduces the dimension to the same obtained via variable

instantiation. Unfortunately, this procedure generates equations which are more complex then the ones obtained by variable instantiation, as it substitutes the exceeding variables by a linear combination of the remaining variables, and since our technique for proving satisfiability relies on interval arithmetic (which is quite sensitive to syntactic manipulations and rounding bounds propagation) this can severely impact on the effectiveness of the check. While we do not rule out that a more engineered implementation of the equation adding approach could yield better results, our experimental results show that the more straight-forward approach of variable instantiation is more effective on the considered benchmarks.

4.6 Related Work

The computation of certificates for formulas *not* being satisfiable in various first-order theories has been an important research topic of the SAT modulo theory community [7] over recent years. In the case of satisfiable formulas, this topic has—to our knowledge—been restricted to the SMT($\mathcal{NTA}$), since for most other theories used in an SMT context, satisfying assignments have a straight-forward representation.

One strategy for proving satisfiability in SMT($\mathcal{NTA}$) is to prove a stricter requirement that implies satisfiability, but is easier to check. For example, one can prove that *all* elements of a set of variable assignments satisfy the given formula [50], or that a given variable assignment satisfies the formula for *all possible interpretations* of the involved transcendental functions within some bounds [30]. Such methods may be quite efficient in proving satisfiability of formulas with inequalities only, since those often have full-dimensional solution sets. However, such methods usually fail to prove satisfiability of equalities, except for special cases with straightforward rational solutions.

Computation of formally verified solutions of square systems of equations is a classical topic in the area of interval analysis [97, 86, 84]. Such methods usually reduce the problem either to fixpoint theorems such as Brouwer's fixpoint theorem or special cases of the topological degree, for example, Miranda's theorem. Such tests are easier to implement, but less powerful than the topological degree (the former fails to verify equalities with double roots, such as $x^3 = 0$, and the latter requires the solution sets of the individual equalities to roughly lie normal to the axes of the coordinate system).

In the area of rigorous global optimization, such techniques are applied [58, 65] to conjunctions of equalities and inequalities in a similar way as in this work, but with a slightly different goal: to compute rigorous upper bounds on the global minimum of an optimization problem. This minimum is often attained at the boundary of the solution set of the given inequalities, whereas satisfiability is typically easier to prove far away from this boundary.

There are several fragments of $\mathcal{NTA}$ for which real root isolation methods have appeared [107, 78, 106, 60, 24, 26]. However, those fragments only allow univariate functions, they are restricted to certain transcendental functions, and only in certain positions. Moreover, some [107, 106, 26] depend on a currently unproved conjecture (Schanuel's conjecture). Examples of such fragments are exp–log–arctan functions [107, 78], tame elementary functions [106], poly-powers [60], mixed trigonometric-polynomials [24], and trigonometric extensions [26]. Recently, by leveraging such real root isolation algorithms, decision proce-

dures have been shown for the theory of univariate mixed trigonometric-polynomials [25] and for trigonometric extensions [26]. Unlike these techniques, our method tackles all of $\mathcal{NTA}$, without any syntactical restrictions.

We are only aware of two approaches that extend verification techniques for square systems of equations to proving satisfiability of quantifier-free non-linear arithmetic [110, 71], one [110] being restricted to the polynomial case, and the other one also being able to handle transcendental function symbols. Neither approach is formulated in the form of certificate search. However, both could be interpreted as such, and both could be extended to return a certificate. This work actually does this for the second approach [71], and demonstrates that this does not only ease the independent verification of results, but also allows the systematic design of search techniques that result in significant efficiency improvements.

An alternative approach is to relax the notation of satisfiability, for example using the notion of δ-satisfiability [54, 18], that does *not* guarantee that the given formula is satisfiable, but only that the formula is not too far away from a satisfiable one, for a suitable formalization of the notion of "not too far away". Another strategy is to return candidate solutions in the form of bounds that guarantee that certain efforts to prove unsatisfiability within those bounds fail [50].

Chapter 5

Theoretical Characterization of a subclass of SMT(NTA)

Since the problem addressed in this thesis is undecidable, the success of any algorithmic approach to solving the problem must necessarily depend on heuristics. Still, in this chapter, we contribute some results that provide insight into when one can reasonably expect an approach such as the one presented in the last chapter to succeed.

Especially, we address a sensitive part of our method—the reduction from an under-constrained system of equations to a well-constrained subsystem (Section 4.4.3). Indeed, given a system of equations in m variables and n equations ($m > n$), in order to obtain a well-constrained system, we need to either instantiate $k := m - n$ variables, or, alternatively, to add k equations. The contribution of this section is threefold:

- We bound the class of problems solvable through the variable instantiation method, both from below and from above.
- We prove that the class of problems solvable through the variable instantiation method is a (possibly non-strict) subset of the class of problems solvable through the equation adding method.
- We show that for bounded systems of equations and inequalities a certain strategy in the certificate search method presented in Section 4.3 will always succeed in determining satisfiability under certain robustness assumptions.

For ease of discussion, first, we will consider only systems of equations (i.e., without inequalities). It is however straightforward to see that including inequalities does not change any of the results. We will then treat the general case of conjunction and disjunctions of systems of equations and inequalities when discussing the last contribution.

We will start by introducing some notation for the relevant classes of problems. For now, we will introduce these classes only informally, and define them precisely, later. We will denote by $\mathcal{F}_{RobI}$ and $\mathcal{F}_{RobLEq}$ the problem classes, for which the two methods (instantiation and adding equations, respectively) result in a robust system. More precisely, we will denote by $\mathcal{F}_{RobI}$ the class of C^1-functions $F : B \subseteq \mathbb{R}^{n+k} \to \mathbb{R}^n$ for which there exists a point $p \in \mathbb{R}^{n+k}$ such that the instantiation of k variables to the corresponding k values of p leads to a robust system in $\mathbb{R}^n \to \mathbb{R}^n$ (we will call such functions *robust under instantiation*), and we will denote by $\mathcal{F}_{RobLEq}$ the class of C^1-functions $F : B \subseteq \mathbb{R}^{n+k} \to \mathbb{R}^n$ for which adding k linear equations leads to a robust system in $\mathbb{R}^{n+k} \to \mathbb{R}^{n+k}$.

First, we will bound $\mathcal{F}_{RobI}$ from above by the class $\mathcal{F}_{Rob}$ of C^1-functions $F : B \subseteq \mathbb{R}^{n+k} \to \mathbb{R}^n$ that have a robust solution, and from below by the class $\mathcal{F}_{Reg}$ of C^1-functions $F : B \subseteq \mathbb{R}^{n+k} \to \mathbb{R}^n$ that have a solution that is regular in the sense of topology.

Based on this, we will prove the following:

Theorem 5.0.1. $\mathcal{F}_{Reg} \subsetneq \mathcal{F}_{RobI} \subsetneq \mathcal{F}_{Rob}$

As can be seen from the disequalities, the lower and upper bounds are strict, here.

Secondly, we prove that every problem that can be solved via variable instantiation can be solved via adding equation, i.e. we have the following theorem:

Theorem 5.0.2. $\mathcal{F}_{RobI} \subseteq \mathcal{F}_{RobLEq}$

Note that the inclusion, in this case, is not strict. Indeed, we conjecture that the equality holds, but will leave the proof to future work.

Finally, we present a variation of our method that is guaranteed to always terminate on problems in $\mathcal{F}_{RobI}$, and that will serve to prove the following theorem:

Theorem 5.0.3. *There exists a procedure that, given a bounded system of equations and inequalities $F = 0 \wedge G \leq 0$,*

- *always returns the correct answer "satisfiable" or "unsatisfiable", if it terminates,*
- *always terminates successfully when $F = 0 \wedge G \leq 0$ is robustly satisfiable and $F \in \mathcal{F}_{RobI}$,*
- *always terminates successfully when $F = 0 \wedge G \leq 0$ is robustly unsatisfiable.*

This theorem can be seen as an extension of an earlier result [48] showing that the class of bounded systems of equations and inequalities in m variables and n equations, with $n \geq m$ or $n = 0$, is quasi-decidable in the sense that there exists a procedure that always terminates on robust instances, and that never returns a wrong answer.

Our contribution is to cover the case of under-constrained systems (i.e. when $n < m$), to a certain extent. Indeed, it is not possible to simply remove the restriction on the number of equations versus the number of variables [48, Theorem 2]. We overcome this by guaranteeing termination in the satisfiable case only for problems for which the system of equations is robust under instantiation (which is a stricter condition than general robustness). In this sense, our procedure is not a quasi-decision procedure, as it does not terminate for *all* robust instances, but it will cover a meaningful sub-class.

Content. This chapter is based on the work presented in [73], and it is organized as follows: in Section 5.1, we formalize the problem classes mentioned by the two theorems and provide some further definitions and properties that will be useful in the following subsections; in Section 5.2, we will prove the lower bound $\mathcal{F}_{Reg} \subseteq \mathcal{F}_{RobI}$ (Lemma 5.2.2) and that $\mathcal{F}_{RobI} \not\subseteq \mathcal{F}_{Reg}$ (Lemma 5.2.3); in Section 5.3 we will prove the upper bound $\mathcal{F}_{RobI} \subseteq \mathcal{F}_{Rob}$ (Lemma 5.3.2) and that $\mathcal{F}_{Rob} \not\subseteq \mathcal{F}_{RobI}$ (Lemma 5.3.1); in Section 5.4, we prove that $\mathcal{F}_{RobI} \subseteq \mathcal{F}_{RobLEq}$; finally, in Section 5.5, we prove Theorem 5.5.1.

5.1 Background on robustness and regularity

In this section, we first give a formal definition of robustness for multivalued functions, and then proceed to provide all the definitions needed to formally define our four classes of interest. We will also present some results regarding these definitions that will be used for proving the main theorem.

Notation: Given a multivalued function $F : \Omega \subseteq \mathbb{R}^m \to \mathbb{R}^n$, we will denote with $F_1, \ldots, F_n : \Omega \subseteq \mathbb{R}^m \to \mathbb{R}$ the univalued functions such that $F = (F_1, \ldots, F_n)$. With a slight abuse of terminology, we will say that a function $F : \Omega \subseteq \mathbb{R}^m \to \mathbb{R}^n$ is *satisfiable* if and only if it has a zero.

5.1.1 Robustness.

First, we provide a formal definition of the concept of robustness. Since our main focus are systems of equations, we will provide a definition of robustness only in terms of multivalued functions. This concept, however, can be generalized and formalized for general formulas. The definition that we give here is just a special case of the more general definition presented in [48]. In fact, a multivalued function F is robustly satisfiable if and only if the logical formula representing the equation $F = 0$ is. We will make use of the general definition of robustness only in the last section, when discussing Theorem 5.5.1.

We first introduce the concept of distance between functions.

Definition 5.1.1 (Distance between two functions)**.** *Let $F : \Omega_1 \subseteq \mathbb{R}^m \to \mathbb{R}^n$ and $F' : \Omega_2 \subseteq \mathbb{R}^m \to \mathbb{R}^n$ be two multivalued continuous functions. Given $\Omega \subseteq \mathbb{R}^m$ such that $\Omega \subseteq \Omega_1$ and*

$\Omega \subseteq \Omega_2$, *we define the distance between* F *and* F' *in* Ω *as*

$$\mathrm{d}_\Omega(F,F') \stackrel{def}{=} \max_{k\in[1,n]}(\|F_k - F'_k\|_\Omega)$$

where $\|F_k - F'_k\|_\Omega \stackrel{def}{=} \sup\{|F_k(x) - F'_k(x)| : x \in \Omega\}$.

When Ω is clear from the context, with an abuse of notation we will drop the subscript and just write $\mathrm{d}(F,F')$. We say that F' is an ε-*small perturbation* of F if $\mathrm{d}(F,F') < \varepsilon$.

Definition 5.1.2 (Robustness of a function)**.** *Given* $\alpha \in \mathbb{R}_{>0}$*, we say that a continuous function* $F : \Omega \subseteq \mathbb{R}^m \to \mathbb{R}^n$ *is* α*-robust iff for every continuous function* F' *s.t.* $\mathrm{d}_\Omega(F,F') < \alpha$*, either both* F *and* F' *have a zero in* Ω*, or none of them has. A function* F *is* ***robust*** *iff there exists* $\alpha \in \mathbb{R}_{>0}$ *s.t.* F *is* α*-robust in* Ω'*.*

For $\Omega' \subseteq \Omega$, we say that F is robust in Ω' iff $F_{|\Omega'} : \Omega' \subseteq \mathbb{R}^m \to \mathbb{R}^n$ is robust.

Definition 5.1.3 (Robustly satisfiable function)**.** *A function* F *is* ***robustly satisfiable*** *iff it is robust and satisfiable.*

5.1.2 Robust solutions ($\mathcal{F}_{Rob}$).

Robust satisfiability of a function—as defined by Definition 5.1.3—does not depend on any specific solution. Indeed, it is the entirety of the solution set that accounts for the robust satisfiability of the function. However, it will be useful to talk about the robustness of a function around a specific solution, which also formalizes the definition of the class $\mathcal{F}_{Rob}$.

Definition 5.1.4 (Robust solution)**.** *Given a function* $F : \mathbb{R}^m \to \mathbb{R}^n$*, we say that a point* $p \in \mathbb{R}^m$ *is a* ***robust solution*** *iff*

1. $F(p) = 0$*, and*
2. *for all* $\varepsilon > 0$ *there exists a* $\delta > 0$ *such that for all* F' *with* $\mathrm{d}(F,F') < \delta$*, there exists* p' *such that* $F'(p') = 0$ *and* $\mathrm{d}(p,p') < \varepsilon$*.*

It follows by the definition of robust solution that if F has a robust solution, then F is robustly sat.

Note that the converse, however, is not true in general:

Example 5.1.1 (Robustly sat formula with no robust solution)**.** *Let* $F : \mathbb{R} \to \mathbb{R}$ *defined by*

$$F : x \mapsto \begin{cases} -x^2 & \text{if } x < 0 \\ 0 & \text{if } 0 \leq x \leq 1 \\ (x-1)^2 & \text{if } 1 < x \end{cases}$$

This function is robustly sat, since every F' *obtained by a small perturbation of* F *has still a solution either near* 0 *or near* 1*. But no point in the solution set of* F *(i.e. the points in* $[0,1]$*) is a robust solution. Indeed,* 0 *is not a robust solution, since, given* $\varepsilon = 0.1$*, for every* $\delta > 0$*, the function* F'_δ *defined by* $F'_\delta(x) \mapsto F(x) - \delta$*; for* 0 *has no solution in the open ball* $\mathcal{B}_{0.1}(0)$*. Symmetrically, for* 1 *we can take* F''_δ *defined by* $F''_\delta(x) \mapsto F(x) + \delta$*. And for every point* $p \in (0,1)$*, we can take* $\varepsilon = \min(\mathrm{d}(p,0), \mathrm{d}(p,1))$*, and for every* δ *either* F'_δ *or* F''_δ*.*

A partial converse is the following result, which states that if a function is locally robustly satisfiable around a solution, the solution is robust.

Proposition 5.1.1. *If* p *is a solution for* F*, and for every* $\varepsilon > 0$ *there exists a neighborhood* $\Omega_\varepsilon \subseteq \mathcal{B}_\varepsilon(p)$ *of* p *such that* F *is robustly sat in* Ω_ε*, then* p *is a robust solution for* F*.*

Proof. For every $\varepsilon > 0$, F is robustly sat in Ω_ε if and only if (by replacing the definition of robustly satisfiable function) $\forall\varepsilon$, $\exists\delta$ s.t. $\forall F'$ with $\mathrm{d}(F,F') < \delta$, F' has a solution in Ω_ε. F' has a solution in Ω_ε if and only if there exists p' s.t. $F'(p') = 0$ and $p' \in \Omega_\varepsilon$ (i.e. $\mathrm{d}(p,p') < \varepsilon$). So we obtain: $\forall\varepsilon$, $\exists\delta$ s.t. $\forall F'$ with $\mathrm{d}(F,F') < \delta$, $\exists p'$ s.t. p' is a solution for F' and $\mathrm{d}(p,p') < \varepsilon$ which is the definition of robust solution for p. □

5.1.3 Robust under instantiation ($\mathcal{F}_{RobI}$)

In general, even if a solution is robust, after the instantiation of some variables, the projection of the solution may not be a robust solution for the function obtained after variable instantiation.

Now we provide some notation and some definitions regarding variable instantiation. We start by formalizing the fact that k coordinates of a point have a finite representation in the form of a dyadic rational number.

Definition 5.1.5 (k-finite point)**.** *A point $p = (p_1, \cdots, p_m) \in \mathbb{R}^m$ is a **k-finite point** (with $0 \leq k \leq m$) if for at least k coordinates $i \in \{1, \ldots, m\}$ there exist integers n_i and r_i such that $p_i = n_i 2^{-r_i}$.*

Here, we use base 2 just for convenience. Any other base would work equally well for our purposes.

Definition 5.1.6 (Robust instantiation of a point)**.** *Let $F : B \subseteq \mathbb{R}^{n+k} \to \mathbb{R}^n$ be a C^1-function and $p = (p_1, \ldots, p_{n+k}) \in \mathbb{R}^{n+k}$ a k-finite point (with I denoting a set of k finitely representable indices).*

Given the partial assignment $\nu_I \stackrel{def}{=} \{x_i \mapsto p_i\}_{i \in I}$, we define the instantiation of F via ν_I *as the function $F_{|\nu_I} : B_{|\mathbb{R}^n} \subseteq \mathbb{R}^n \to \mathbb{R}^n$ such that for every $(x_1, \ldots, x_n) \in \mathbb{R}^n$, $F_{|\nu_I}(x_1, \ldots, x_n) = F(y_1, \ldots, y_{n+k})$, where*

$$\text{for } i \in \{1, \ldots, n+k\},\ y_i = \begin{cases} p_i, \text{if } i \in I, \\ x_i, \text{if } i \notin I \end{cases}.$$

*We say that the partial assignment ν_I is a **robust instantiation of** p if and only if the point $p_{|\nu_I} = (p_i)_{i \notin I} \in \mathbb{R}^n$ is a robust solution for $F_{|\nu_I}$.*

If an instantiation is not a robust instantiation, then we say it is a *non-robust instantiation*. Note that assignments are defined as maps to the set of finitely representable values. So if p is not a k-finite point, then it admits no instantiations, and hence no robust instantiations.

For our method to succeed, we only have to be lucky once, and show the existence of a single robust instantiation. So we are not interested in solutions that are robust under *all* instantiations, but in solutions that are robust under *at least one* instantiation. Finally, we have the following definition (from which the definition of $\mathcal{F}_{RobI}$ follows):

Definition 5.1.7 (Robust under instantiation)**.** *Given a continuous function $F : B \subseteq \mathbb{R}^{n+k} \to \mathbb{R}^n$ and a point p, we say that p is **robust under instantiation** if there exists at least one robust instantiation of p.*

If no robust instantiation of p exists, then we say that p is *non-robust under instantiations*. If p is not a k-finite point, then, by definition, p is non-robust under instantiations.

In the following, for ease of notation and without loss of generality, we will assume that $I = \{n+1, \ldots, n+k\}$, unless otherwise specified. In this case, we will write ν instead of ν_I,

and we will denote $p_{[1,n]} \stackrel{\text{def}}{=} (p_1, \ldots, p_n)$ for the projection of p to the first n coordinates (i.e. the ones not instantiated by v), and $p_{[n+1,n+k]}$ for the projection of p to the last k coordinates (i.e. the ones not instantiated by v).

5.1.4 Robustness after equation adding ($\mathcal{F}_{RobLEq}$).

We define $\mathcal{F}_{RobLEq}$ as the set of C^1-functions $F : \mathbb{R}^{n+k} \to \mathbb{R}^n$ such that there exists a linear function $L : \mathbb{R}^{n+k} \to \mathbb{R}^k$ such that the function $F_{leq} : \mathbb{R}^{n+k} \to \mathbb{R}^{n+k}$, defined by $F_{leq} : x \mapsto (F(x), L(x))$, has a robust solution.

5.1.5 Regular solutions ($\mathcal{F}_{Reg}$).

We now define the last of our classes of interest. For doing so, we provide a brief background on the notion of regularity from the field of differential topology.

Definition 5.1.8 (Regular point)**.** *Let $F : B \subseteq \mathbb{R}^m \to \mathbb{R}^n$ be a C^1-function, with $m - n = k \geq 0$. We say that $p \in B$ is a **regular point** for F if and only if the Jacobian matrix of F at x has maximal rank.*

If $F(p) = 0$ and p is a regular point, we will say that p is a *regular solution* of F. If p is not a regular point, we say that p is a *critical point*. We say that $q \in \mathbb{R}^n$ is a *regular value* if and only if for every p such that $F(p) = q$, p is a regular point. If q is not a regular value, we say it is a *critical value*.

Regularity is a very meaningful property, as it guarantees the "well-behavior" of the system. In particular, a regular point p satisfies the hypothesis of the Implicit Function Theorem [85], that we now recall and that we will use in Section 5.2:

Implicit Function Theorem: If $F(p_1, \ldots, p_n, p_{n+1}, \ldots, p_{n+k}) = 0$, and the Jacobian matrix of F with respect to the first n coordinates has non-zero determinant in p (i.e. $\det(J_{F,x_{|n}}(p)) \neq 0$), then there exists a neighborhood $U \subseteq \mathbb{R}^k$ of $(p_{n+1}, \ldots, p_{n+k})$ and a C^1-function $H : U \to \mathbb{R}^n$ such that $H(p_{n+1}, \ldots, p_{n+k}) = (p_1, \ldots, p_n)$, and such that, for all $q \in U$, $F(H(q), q) = 0$.

5.2 Regularity and robustness under instantiation

In this section, we prove that the existence of a regular solution is a sufficient—but not necessary—criterion for the existence of a solution robust under instantiation.

We prove that, if F has at least one regular solution q, then there exists at least one regular solution p such that at least k coordinates of p are finitely representable rational numbers, and such that the subsystem induced from the instantiation of the k corresponding variables is robustly sat.

We first show that any regular solution is also a robust solution (Lemma 5.2.1), which will be useful to prove the main result of this section, Lemma 5.2.2, which implies $\mathcal{F}_{Reg} \subseteq \mathcal{F}_{RobI}$. Then, we will show that this inclusion is strict by providing a counter-example in the form of a system of equations that has a solution robust under instantiation but no regular solutions (Lemma 5.2.3).

Lemma 5.2.1. *Given a C^1-function $F : B \subseteq \mathbb{R}^n \to \mathbb{R}^n$, if p is a regular solution for F, then p is a robust solution for F.*

Proof. Assume that p is a regular solution of F. Hence the Jacobian of F at p has maximal rank. We prove that p is a robust solution of F. So let $\varepsilon > 0$ be arbitrary, but fixed and such that p is the unique solution of F in $\mathcal{B}_\varepsilon(p)$. Such ε always exists, since by the inverse function theorem F maps a neighborhood of p diffeomorphically onto an open set of $\mathbb{R}^n$ [81, Chapter 1.2]. Hence $0 \notin \partial\mathcal{B}_\varepsilon(p)$, and, since p is the only solution of F in $\mathcal{B}_\varepsilon(p)$, and it is regular, then, by definition, $\deg(F, \mathcal{B}_\varepsilon(p), 0) = |\det(J_F(p))| \neq 0$. Let $\delta < \min_{x \in \partial\mathcal{B}_\varepsilon(p)} |F(x)|$. By Lemma 1 [48], every F' with $\mathrm{d}(F, F') < \delta$ has a zero in $\mathcal{B}_\varepsilon(p)$. This proves that p is a robust solution for F. □

Lemma 5.2.2. *Let $F : B \subseteq \mathbb{R}^m \to \mathbb{R}^n$ (with $m = n + k$) be a C^1-function. If there exists a regular solution q of F, then there exists a regular solution p of F in a neighborhood of q such that p is robust under instantiation.*

Proof. If q is a regular solution for F, then $J_F(q)$ has maximum rank. Since a rectangular matrix has maximum rank if and only if one of its maximal square sub-matrix has maximum rank, then, without loss of generality, we can reorder the variables so that the square sub-matrix given by the first n columns has maximum rank, i.e. $\det(J_{F,x_{[1,n]}}(q)) \neq 0$. By the Implicit Function Theorem, there exists a neighborhood $U \subseteq \mathbb{R}^k$ of $(q_{n+1}, \ldots, q_{n+k})$ and a C^1-function $H : U \to \mathbb{R}^n$ such that $H(q_{n+1}, \ldots, q_{n+k}) = (q_1, \ldots, q_n)$, and such that, for all $q' \in U$, $F(H(q'), q') = 0$.

In general, it is not guaranteed that every $(H(q'), q')$ will be a regular point for F. However, since the Jacobian $J_F : \mathbb{R}^m \to \mathbb{R}^{m \times n}$, the projection $\pi_{n \times n} : \mathbb{R}^{m \times n} \to \mathbb{R}^{n \times n}$ (that projects a $m \times n$ matrix onto the $n \times n$ sub-matrix of its first n columns) and the determinant

$\det : \mathbb{R}^{n\times n} \to \mathbb{R}$ are all continuous functions, then the set $U_r \subseteq \mathbb{R}^m$, consisting of all the points q' for which $\det(J_{F,x_{[1,n]}}(q')) \neq 0$, is open, since $\mathbb{R} \setminus \{0\}$ is open and, by definition, $U_r = (det \circ \pi_{n\times n} \circ J_F)^{-1}(\mathbb{R} \setminus \{0\})$.

Let us consider the projection of U_r over $\mathbb{R}^k$, i.e. $U_{r_{|k}} = \pi_k(U_r) = \{(q'_{n+1}, \ldots, q'_{n+k}) \in \mathbb{R}^k | (q'_1, \ldots, q'_n, q'_{n+1}, \ldots, q'_{n+k}) \in U_r\}$. Since projections are open maps, then $U_{r_{|k}}$ is open. Since both U and $U_{r_{|k}}$ are neighborhoods of $(q_{n+1}, \ldots, q_{n+k})$, then their intersection $U' \stackrel{\text{def}}{=} U \cap U_{r_{|k}}$ is again a neighborhood of $(q_{n+1}, \ldots, q_{n+k})$.

Now we prove that U' contains at least one k-finite point p'. The set of k-finite points in $\mathbb{R}^k$ is exactly the set of points having coordinates with finite representation. Let us call this set A. We have that A is dense in $\mathbb{R}^k$, as, for every point $z = (z_1, \ldots, z_k) \in \mathbb{R}^k$, z is the limit of the sequence $\{([z_1]_i, \ldots, [z_k]_i)\}_{i\in\mathbb{N}} \subseteq A$, where $[z_j]_i$ is the truncation of z_j to the i-th digit after the zero. Since A is dense in $\mathbb{R}^k$, then A intersects every non-empty open of $\mathbb{R}^k$. In particular $A \cap U' \neq \emptyset$, hence there exists $p' \in A \cap U'$.

Let $p \stackrel{\text{def}}{=} (H(p'), p') \in \mathbb{R}^m$. We have that p is a solution for F (since $F(p) = F(H(p'), p') = 0$). Moreover, p is also regular, since $p' \in U' \subseteq U_{r_{|k}}$ and hence $(H(p'), p') \in U_r$.

Let $\nu \stackrel{\text{def}}{=} \{x_i \mapsto p_i\}_{i\in[n+1,n+k]}$. We have that the point $p_{[1,n]} = H(p') \in \mathbb{R}^n$ is a regular point for $F_{|\nu}$. Indeed, since p is a regular point, and $J_{F,x_{|n}}(p)$ depends only on the first n coordinates, then $\det(J_{F_{|\nu}}(p_{[1,n]})) \neq 0$.

Since $p_{[1,n]}$ is a regular solution for $F_{|\nu}$, by Lemma 5.2.1, $p_{[1,n]}$ is also a robust solution for $F_{|\nu}$. Hence p is robust under instantiation. □

Modifying the proof by choosing directly p as q we get:

Corollary 5.2.1. *Let $F : B \subseteq \mathbb{R}^m \to \mathbb{R}^n$ (with $m = n + k$) be a C^1-function If q is both a regular solution and a k-finite point, then q is robust under instantiation.*

Now we show that the converse of Lemma 5.2.2 does not hold, i.e. that the existence of a solution robust under instantiation does not imply the existence of a regular solution. Consider the following example:

Example 5.2.1 (Critical solution, but robust under instantiation)**.** *Let $F : [0,1]^2 \subseteq \mathbb{R}^2 \to \mathbb{R}$ defined by $F(x,y) = (x^2 - y^3)$, and let $p = (0,0)$. $J_F(p) = (0,0)$ has non-maximum rank (hence p is a critical solution), but the instantiation $\{x \mapsto 0\}$ leads to the subsystem $-y^3 = 0$, which is robust.*

In this example, we could have chosen a different point, say $p' = (1,1)$, which is both robust under instantiation and regular.

However, this is not always possible. A system can have a solution robust under instantiation, but no regular solutions. Indeed:

Lemma 5.2.3. $\mathcal{F}_{RobI} \not\subseteq \mathcal{F}_{Reg}$

Proof. Let $F : [-1,1]^3 \subseteq \mathbb{R}^3 \to \mathbb{R}^2$ defined by

$$F(x_1,x_2,x_3) = \begin{cases} x_1^3 & (F_1) \\ x_2 + x_3 & (F_2) \end{cases}$$

The point $(0,0,0)$ is robust under instantiation. Indeed, the instantiation $\{x_3 \mapsto 0\}$ leads to the following system of equations

$$F'(x_1,x_2) = \begin{cases} x_1^3 & (F_1') \\ x_2 & (F_2') \end{cases}$$

which has non-zero degree, hence it is robustly sat. It is easy to show that the degree of F' in $[-1,1]^2$ is non-zero. In fact, F_1' depends only on x_1 and F_2' only on x_2, and for both F_1' and F_2' it suffices to apply the Intermediate Value Theorem to prove that $\deg(F_i', [-1,1], 0) \neq 0$ (for $i = 1,2$). Since the degree of the Cartesian product is the product of the degrees [39, Theorem 7.1.1], $\deg(F', [-1,1]^2, 0) = \deg(F_1', [-1,1], 0) * \deg(F_2', [-1,1], 0) \neq 0$.

So $F \in \mathcal{F}_{RobI}$. However, F does not have any regular solution. Indeed, the first equation implies that for every every solution p its first coordinate has to be $p_1 = 0$. Since, for such a solution p, the first row of $J_F(p)$ is everywhere 0, then the Jacobian cannot have maximum rank. Hence every solution p is not regular, i.e. $F \notin \mathcal{F}_{Reg}$. □

5.3 Robustness preservation after variable instantiation

In the previous section, we have proven that, under the assumption of the existence of a regular solution, there exists a solution that is robust under instantiation.

But what happens if we drop the assumption of regularity? In general, if we don't put any restriction on the functions we are considering, the solution space can be arbitrarily complicated. Indeed, for every closed subset $K \subseteq \mathbb{R}^m$, there exists a smooth function F such that $F^{-1}(0) = K$ ([69, Theorem 2.29]).

One may hope that, by restricting to functions that have a robust solution, we can always find a solution robust under instantiation. In this section, we show that this, unfortunately, does not hold. Consider the following example.

Example 5.3.1 (Robust solution, but non-robust under instantiations)**.** *Let $F : [-1,1]^2 \subseteq \mathbb{R}^2 \to \mathbb{R}$ defined by $F(x,y) = (x^2 - y^2)$, and let $p = (0,0)$. It is easy to show that p is a robust solution. However, whether we instantiate $\{x \mapsto 0\}$ or $\{y \mapsto 0\}$, the resulting subfunctions (resp. $F_{|\{x\mapsto 0\}}(y) = y^2$ and $F_{|\{y\mapsto 0\}}(x) = x^2$) are not robust.*

In this example, we could have chosen another point, for example $p' = (1,1)$, which is regular (since $J_F(p') = (2,-2)$), and hence, by Corollary 5.2.1, robust under instantiation. But in general, this is not always possible. Indeed, we have the following result:

Lemma 5.3.1. $\mathcal{F}_{Rob} \not\subseteq \mathcal{F}_{RobI}$

Proof. Let $F : [-1,1]^4 \subseteq \mathbb{R}^4 \to \mathbb{R}^3$ defined by

$$F(x_1,x_2,x_3,x_4) = \begin{cases} x_1^2 + x_2^2 - x_3^2 - x_4^2 \\ 2(x_1x_4 + x_2x_3) \\ 2(x_2x_4 - x_1x_3) \end{cases}$$

It is easy to show that $p = (0,0,0,0)$ is the only solution of F. Moreover, p is robust (see the discussion about Hopf maps in [45]) , hence $F \in \mathcal{F}_{Rob}$. However, no instantiation $\nu_i \stackrel{\text{def}}{=} \{x_i \mapsto 0\}$ is robust. Indeed, $(0,0,0)$ is the only solution of $F_{|\nu_i}$ in $[-1,1]^3$, but $\deg(F_{|\nu_i}, [-1,1]^3, 0) = 0$ (remember that, if a system of equations F has an isolated robust solution in B, then $\deg(F,B,0) \neq 0$). So $F \notin \mathcal{F}_{RobI}$. □

Now we show that the converse holds, i.e. that every solution robust under instantiation is robust:

Lemma 5.3.2. *Let $F : B \subseteq \mathbb{R}^{n+k} \to \mathbb{R}^n$. If p is a solution robust under instantiation, then p is a robust solution.*

Proof. If p is a solution robust under instantiation, then there exists a set of indices I (w.l.o.g. say $I = \{n+1,\ldots,n+k\}$) and a corresponding instantiation ν such that $p_{[1,n]}$ is a robust solution for $F_{|\nu} : B_{|\mathbb{R}^n} \subseteq \mathbb{R}^n \to \mathbb{R}^n$, that is, for every $\varepsilon > 0$ there is a δ such that for all $F'_{|\nu}$ with $\mathrm{d}(F_{|\nu}, F'_{|\nu}) < \delta$, there exists a solution $p'_{[1,n]}$ of $F'_{|\nu}$ with $\mathrm{d}(p_{[1,n]}, p'_{[1,n]}) < \varepsilon$.

To prove that p is a robust solution for F let $\varepsilon > 0$ be arbitrary, but fixed, and take the corresponding δ as ensured by robustness under instantiation. For every F' with $\mathrm{d}(F,F') < \delta$, we have that also $\mathrm{d}(F_{|\nu}, F'_{|\nu}) < \delta$, hence there exists $p'_{[1,n]}$ that satisfies $F'_{|\nu}$ with $\mathrm{d}(p_{[1,n]}, p'_{[1,n]}) < \varepsilon$. If $p'_{[1,n]}$ satisfies $F'_{|\nu}$, then $p' \stackrel{\text{def}}{=} (p'_{[1,n]}, p_{[n+1,n+k]})$ satisfies F'. Since $\mathrm{d}(p,p') = \mathrm{d}(p_{[1,n]}, p'_{[1,n]}) < \varepsilon$, p satisfies the definition of robust solution for F. □

A straightforward corollary of Lemma 5.3.2, is that $\mathcal{F}_{RobI} \subseteq \mathcal{F}_{Rob}$ which concludes the proof that $\mathcal{F}_{RobI} \subsetneqq \mathcal{F}_{Rob}$. Together with Lemma 5.2.3 and Lemma 5.2.2, this implies Theorem 5.0.1.

5.4 Variable instantiation vs. equation adding

Given an under-constrained system of equations, we showed that there are two different ways to reduce to a well-constrained system of equations: decreasing the number of variables (i.e. instantiations) or increasing the number equations. In this section we will show that the class of systems that can be solved via variable instantiation is a subset of the class of systems that can be solved via adding equations.

Recall our definition of $\mathcal{F}_{RobLEq}$ as the set of functions $F : B \subseteq \mathbb{R}^{n+k} \to \mathbb{R}^n$ such that there exists a linear function $L : \mathbb{R}^{n+k} \to \mathbb{R}^k$ such that the function $F_{leq} : B \subseteq \mathbb{R}^{n+k} \to \mathbb{R}^{n+k}$, defined by $F_{leq} : x \mapsto (F(x), L(x))$, has a robust solution.

Given any partial assignment $\nu \stackrel{\text{def}}{=} \{x_i \mapsto p_i\}_{i \in [n+1,n+k]}$, we can consider the function $F_{leq} = (F,L)$, given by F and by the linear function $L : B \subseteq \mathbb{R}^{n+k} \to \mathbb{R}^k$ defined by $L : (x_1, \ldots, x_{n+k}) \mapsto (x_{n+1} - p_{n+1}, \ldots, x_{n+k} - p_{n+k})$. Equivalently, since L only depends on the last k variables, we can consider it as a function $L : B_{|\mathbb{R}^k} \subseteq \mathbb{R}^k \to \mathbb{R}^k$.

While it is trivial to show that every solution of $F_{|\nu}$ is also a solution of F_{leq}, we need to prove that also the robustness of a solution is preserved.

Lemma 5.4.1. *Given a system of equations $F = 0$ (with $F : B \subseteq \mathbb{R}^{n+k} \to \mathbb{R}^n$), a point $p = (p_1, \ldots, p_{n+k}) \in B$ and subset of indexes $I := \{n+1, n+2, \ldots, n+k\}$, let us consider the following statements:*

1. *For the function $F_{|\nu} : B_{|\mathbb{R}^n} \subseteq \mathbb{R}^n \to \mathbb{R}^n$, obtained by instantiating variables via $\nu = \{x_i \mapsto p_i\}_{i \in I}$, the point $p_{[1,n]} \stackrel{def}{=} (p_1, \ldots, p_n)$ is a robust solution*

2. *For the function* $F_{leq} : B \subseteq \mathbb{R}^{n+k} \to \mathbb{R}^{n+k}$*, defined by*

$$F_{leq}(x_1,\ldots,x_{n+k}) \mapsto (F(x_1,\ldots,x_{n+k}), L(x_{n+1},\ldots,x_{n+k}))$$

where $L : B_{|\mathbb{R}^k} \subseteq \mathbb{R}^k \to \mathbb{R}^k$ *is defined by*

$$L : (x_{n+1},\ldots,x_{n+k}) \mapsto (x_{n+1} - p_{n+1},\ldots,x_{n+k} - p_{n+k}),$$

the point p is a robust solution.

Then, it holds that 1. *implies* 2..

Proof. To prove that 1. $\Rightarrow$ 2., it will be useful to consider a third auxiliary condition:

3. *For the function* $F_{|\mathcal{V}_{leq}} \stackrel{def}{=} F_{|\mathcal{V}} \times L : B \subseteq \mathbb{R}^{n+k} \to \mathbb{R}^{n+k}$ *obtained by the Cartesian product between* $F_{|\mathcal{V}}$ *and the linear function L, the point p is a robust solution*

and prove first that 1. $\Rightarrow$ 3. and then that 3. $\Rightarrow$ 2.

(1. $\Rightarrow$ 3.) By Theorem 7.1.1 [39], the degree of the Cartesian product is the product of the degrees, i.e., for every $\Omega = \Omega_1 \times \Omega_2 \subseteq \mathbb{R}^n \times \mathbb{R}^k$,

$$\deg(F_{|\mathcal{V}_{leq}}, \Omega, 0) = \deg(F_{|\mathcal{V}}, \Omega_1, 0) * \deg(L, \Omega_2, 0) \tag{5.1}$$

Since $J_L(p_{[n+1,n+k]})$ is the identity matrix, the point $p_{[n+1,n+k]}$ is a regular solution for L. Since $p_{[n+1,n+k]}$ is the only solution of L, then, for every $\Omega_2 \subseteq \mathbb{R}^k$ such that $p_{[n+1,n+k]} \in \Omega_2$, $\deg(L, \Omega_2, 0) = \det(J_L(p_{[n+1,n+k]})) = 1$.

Thus, for every $\Omega = \Omega_1 \times \Omega_2 \subseteq \mathbb{R}^n \times \mathbb{R}^k$ such that $p \in \Omega$, we have that

$$\deg(F_{|\mathcal{V}_{leq}}, \Omega, 0) = \deg(F_{|\mathcal{V}}, \Omega_1, 0) \tag{5.2}$$

If 1. holds, then, by Thm. 6 [48], for every $\varepsilon > 0$ there exists an open $\Omega_{1,\varepsilon} \subseteq \mathcal{B}_\varepsilon(p_{[1,n]})$ such that $\deg(F_{|\mathcal{V}}, \Omega_{1,\varepsilon}, 0) \neq 0$. Let Ω_2 be any neighborhood of $p_{[n+1,n+k]}$ s.t. $\Omega_2 \subseteq \mathcal{B}_\varepsilon(p_{[n+1,n+k]})$, and let $\Omega_\varepsilon \stackrel{\text{def}}{=} \Omega_{1,\varepsilon} \times \Omega_2$. By Equation 5.2, $\deg(F_{|\mathcal{V}_{leq}}, \Omega_\varepsilon, 0) \neq 0$. Hence, for every $\varepsilon > 0$, $F_{|\mathcal{V}_{leq}}$ is robustly sat in $\Omega_{1,\varepsilon}$. So we have constructed, for every $\varepsilon > 0$, a neighborhood $\Omega_{1,\varepsilon} \subseteq \mathcal{B}_\varepsilon(p)_\varepsilon$ of p in which $F_{|\mathcal{V}_{leq}}$ is robustly sat. By Proposition 5.1.1, this implies that p is a robust solution for $F_{|\mathcal{V}_{leq}}$.

(3. $\Rightarrow$ 2.) Now we show that if p is a robust solution for $F_{|\mathcal{V}_{leq}}$ then it is a robust solution for F_{leq}. We will construct a homotopy between $F_{|\mathcal{V}_{leq}}$ and F_{leq}, and then rely on the Homotopy Invariance Property of the topological degree to prove the claim.

Let $S : \mathbb{R}^{n+k} \times [0,1] \to \mathbb{R}^{n+k}$ be defined by

$$S : ((x_1,\ldots,x_n,x_{n+1},\ldots x_{n+k}),t) \mapsto \\ (x_1,\ldots,x_n,\ tx_{n+1}+(1-t)p_{n+1},\ \ldots\ ,\ tx_{n+k}+(1-t)p_{n+k})$$

The map $H : B \times [0,1] \subseteq \mathbb{R}^{n+k} \times [0,1] \to \mathbb{R}^{n+k}$ defined by

$$H : (x_1,\ldots,x_{n+k},t) \mapsto ((F \circ S)(x_1,\ldots,x_{n+k},t)\ ,\ L((x_{n+1},\ldots,x_{n+k})))$$

is a homotopy between $F_{|\mathcal{V}_{leq}}$ and F_{leq} since $H(\cdot,0) \equiv F_{|\mathcal{V}_{leq}}$, $H(\cdot,1) \equiv F_{leq}$, and H is continuous, being the composition of continuous functions.

It is easy to see that the functions F_{leq} and $F_{|\mathcal{V}_{leq}}$ have exactly the same solution space. Indeed, every solution of F_{leq} has to satisfy the equations given by L, i.e. $x_{n+1} = p_{n+1},\ldots,x_{n+k} = p_{n+k}$. By replacing in F_{leq} every x_i with p_i, for $i \in [n+1,n+k]$, we obtain exactly $F_{|\mathcal{V}_{leq}}$.

Furthermore, for every $t \in [0,1]$ the solution space of $H(\cdot,t)$ is the same as $H(\cdot,0) \equiv F_{|\mathcal{V}_{leq}}$. Indeed, for every t, a solution of $H(\cdot,t)$ has to satisfy the equations given by L. Then, by replacing every x_i with p_i for $i \in [n+1,n+k]$, we have that every $tx_i+(1-t)p_i$ is replaced by $tp_i+(1-t)p_i$, which is equal to p_i. Thus we reduced again to $F_{|\mathcal{V}_{leq}}$.

Now, suppose that 3. holds. Then, for every $\varepsilon > 0$, there exists $\Omega_\varepsilon \subseteq \mathcal{B}_\varepsilon(p)$ such that $\deg(F_{|\mathcal{V}_{leq}},\Omega_\varepsilon,0) \neq 0$. This implies that $0 \notin F_{|\mathcal{V}_{leq}}(\partial\Omega_\varepsilon)$, hence $0 \notin H(\partial\Omega_\varepsilon,1)$. So, by the previous observation, $0 \notin H(\partial\Omega_\varepsilon,t)$ for every $t \in [0,1]$. Hence $0 \notin H(\partial\Omega_\varepsilon,[0,1])$, and we can apply the Homotopy Invariance Property.

The Homotopy Invariance Property of the topological degree states that, if $0 \notin H(\partial\Omega_\varepsilon \times [0,1])$, then $\deg(H(\cdot,0),\Omega_\varepsilon,0) = \deg(H(\cdot,1),\Omega_\varepsilon,0)$, i.e. $\deg(F_{|\mathcal{V}_{leq}},\Omega_\varepsilon,0) = \deg(F_{leq},\Omega_\varepsilon,0)$. So we have constructed, for every $\varepsilon > 0$, a neighborhood $\Omega_\varepsilon \subseteq \mathcal{B}_\varepsilon(p)$ of p in which F_{leq} is robustly sat. By Proposition 5.1.1, this implies that p is a robust solution for F_{leq}. □

So robustness of the system obtained by variable instantiation implies robustness of the system obtained by adding the equalities corresponding to this variable instantiation. Theorem 5.0.2 is a straight-forward consequence.

5.5 Termination

The method discussed in Section 4.4 made use of numerical optimization to enumerate the points over which the variable instantiation method is applied. This technique, while practically very efficient—as shown by our experiments—is not guaranteed to terminate, in general. Indeed, even in the bounded case, numerical optimization does not guarantee full coverage of the space.

In this section, we present a variation of our method that uses a different technique for enumerating points, and that is guaranteed to terminate on problems in $\mathcal{F}_{RobI}$. While this variation is not intended to be of practical use, it will serve the purpose of proving Theorem 5.5.1.

Given $F : B \subseteq \mathbb{R}^{n+k} \to \mathbb{R}^n$, if $F \in \mathcal{F}_{RobI}$, by definition we know that there exists a k-finite point p that is robust under instantiation. Such a point p is, in general, not a $(n+k)$-finite point (indeed, there are problems for which no solution is a $(n+k)$-finite point). Hence no point enumeration technique is guaranteed to find precisely p, since only $(n+k)$-finite points can be expressed explicitly. However, this is not an actual limitation. Indeed, for our method to succeed, we don't necessarily need to explicitly produce a solution. We just need to find a point sufficiently close to an actual solution, and that shares with the solution k indices, so that, after the instantiation of the corresponding k variables, we end up with a subproblem that is robustly satisfiable. This suffices to produce a certificate.

The following lemma shows that, for every problem in $\mathcal{F}_{RobI}$, it is always possible to find a $(n+k)$-finite point and a partial assignment such that the resulting subproblem is robustly satisfiable.

Lemma 5.5.1. *Let $F : B \subseteq \mathbb{R}^{n+k} \to \mathbb{R}^n$. If $F \in \mathcal{F}_{RobI}$, then there exist a $(n+k)$-finite point $p' \in B$ and a partial assignment $\nu' \stackrel{def}{=} \{x_i \to p'_i\}_{i \in I}$ (with I being a set of k indices), such that the function $F_{|\nu'} : B_{|\mathbb{R}^n} \subseteq \mathbb{R}^n \to \mathbb{R}^n$ is robustly satisfiable.*

Proof. $F \in \mathcal{F}_{RobI}$ means that there exists a k-finite point p (w.l.o.g. say the k finitely representable indices are $[n+1, n+k]$) that is robust under instantiation, i.e. there exists a partial assignment $\nu \stackrel{\text{def}}{=} \{x_i \mapsto p_i\}_{i \in [n+1,n+k]}$ such that $p_{|\nu}$ is a robust solution for $F_{|\nu}$. This implies that $F_{|\nu}$ is robustly satisfiable in $B_{|\mathbb{R}^n}$.

Now, given any $p' \in B$ such that $p'_{n+1} = p_{n+1}, \ldots, p'_{n+k} = p_{n+k}$, and, given $\nu' \stackrel{\text{def}}{=} \{x_i \mapsto p'_i\}_{i \in [n+1,n+k]}$, we have that $\nu' \equiv \nu$, hence $F'_{|\nu} \equiv F_{|\nu}$, which implies that $F_{|\nu'}$ is robustly

satisfiable in $B_{|\mathbb{R}^n}$. In order to find a p' that respects the statement conditions, first we fix the last k coordinates to be equal to $(p_{n+1},\ldots,p_{n+k})$. Then, since the set of n-finite points is dense in $\mathbb{R}^n$, and hence intersects $B^{\circ}_{|\mathbb{R}^n}$ (being an open), there exists a n-finite point $(p'_1,\ldots,p'_n) \in B^{\circ}_{|\mathbb{R}^n}$. If we take such point, and append the last k coordinates previously fixed, we obtain a point $p' \stackrel{\text{def}}{=} (p'_1,\ldots,p'_n,p_{n+1},\ldots,p_{n+k})$. Since the last k coordinates of p' coincides with the last k coordinates of p, we have that $F_{|\nu'}$ is robustly satisfiable in $B_{|\mathbb{R}^n}$. Moreover, such p' is a $(n+k)$-finite point. Indeed, the part consisting of the first n coordinates is n-finite by construction, while the second part consisting of the last k coordinates is k-finite because $p_{n+1},\ldots,p_{n+k}$ is. □

Considering a bounded system of equations and inequalities satisfiable iff it is satisfiable by a variable assignment the assigns values within the corresponding interval to all variables, it is straightforward to extend the definitions from Section 5.1 analogically from systems of equations to bounded systems of equations and inequalities. Based on this, we can prove the following theorem.

Theorem 5.5.1. *There exists a procedure that, given a bounded system of equations and inequalities $F = 0 \wedge G \leq 0$,*

- *always returns the correct answer "satisfiable" or "unsatisfiable", if it terminates,*
- *always terminates successfully when $F = 0 \wedge G \leq 0$ is robustly satisfiable and $F \in \mathcal{F}_{RobI}$,*
- *always terminates successfully when $F = 0 \wedge G \leq 0$ is robustly unsatisfiable.*

Proof. We first concentrate on the second point, by showing a procedure that always correctly terminates on problems in $\mathcal{F}_{RobI}$.

By Lemma 5.5.1, we have that, for every $F : B \subseteq \mathbb{R}^{n+k} \to \mathbb{R}^n$ such that $F \in \mathcal{F}_{RobI}$, there exists a $(n+k)$-finite point p' and a partial assignment ν' such that $F_{|\nu'}$ is robustly satisfiable. We can always find such p' and ν'. Indeed, since the set of $(n+k)$-finite points is countable, we can construct a well-order: say, for example, we first take the finite set of points whose coordinates are represented by at most 1 digit (and sort it by lexicographic order), then the finite set of points whose coordinates are represented by at most 2 digits, and so on [1]. For each such point p, and for each instantiation ν (note that the set of possible instantiations is finite), we can consider the resulting subsystem $F_{|\nu'} : B_{|\mathbb{R}^n} \subseteq \mathbb{R}^n \to \mathbb{R}^n$, and then apply the

[1] Note that this is independent by the Axiom of Choice, which is needed only in the case of uncountable sets.

box-gridding procedure discussed in Section 4.4.4, which—without the stopping criterion—is guaranteed to terminate on robust instances [48]. Note that box-gridding method also handles inequalities.

The procedure described so far is not yet guaranteed to converge. Indeed, while box-gridding is guaranteed to terminate on robust instances, it could diverge on non-robust instances, thus preventing the general procedure to terminate (either because one point yielded before p' was non-robust, or because one instantiation tried before v' was a non-robust instantiation).

We can overcome this problem by using, instead of depth-first search, a technique called dove-tailing. Indeed, we have an infinite sequence of problems (given by the combination of points and instantiations), and, for each, a (possibly infinite) sequence of box-gridding iterations. We outline the following iterative procedure:

- For $i = 1$, we perform the 1-st box-gridding iteration on the 1-st problem.
- For $i = 2$, we perform the 2-nd box-gridding iteration on the first problem, and then the 1-st box-gridding iteration on the second problem.
- ...
- For $i = N$, we perform the N-th box-gridding iteration on the first problem, then the $(N-1)$-th box-gridding iteration on the second problem, ..., and then the 1-st box-gridding iteration on the N-th problem.

First, we are guaranteed to find the problem given by the point p' and the variable instantiation v' after a finite amount of steps. Given such problem, we are guaranteed that also the box-gridding procedure will terminate after finitely many iterations. Hence, also our general procedure is guaranteed to terminate.

The third point regarding robustly unsatisfiable problems simply follows by the use of box-gridding on the whole system using the bounds of the given system of equations and inequations as its starting box. Indeed, if the system is robustly unsatisfiable, then this is guaranteed to terminate with a correct result.

To finalize the proof of the theorem, it suffices to consider the procedure that runs the two previous procedures in parallel.

□

Note that for formulas of the given form (bounded system of equations and inequalities), the procedure described in the proof of Theorem 5.5.1 can be seen as an instantiation of the certificate search method presented in Section 4.3 that uses exhaustive enumeration on the level of points and instantiations, and directly uses the given bounds as the starting box for box gridding. The theorem shows that under robustness assumptions, such an instantiation will always terminate successfully for bounded system of equations and inequalities. However, complete enumeration makes this instantiation hopelessly inefficient in practice, and goal oriented methods, as discussed in the first part of the paper, are necessary for practical efficiency. Also, there is no known way of algorithmically deciding whether a given formula satisfies the robustness precondition that ensures termination of the procedure, and hence Theorem 5.5.1 is *not* a decidability result.

Chapter 6

Optimization-guided MCSAT for SMT(NIA)

Historically, SMT solving techniques have been divided into two main approaches: the *eager* approach, and the *lazy* approach. In the eager approach, an SMT formula is encoded into an equi-satisfiable SAT formula, effectively reducing an SMT problem to a SAT one. The lazy approach, on the contrary, clearly separates the Boolean reasoning by the theory reasoning, relying on a SAT solver for the former and on a theory solver for the latter, usually within the CDCL(T) framework.

A generalization of CDCL(T) that has gained considerable popularity in the last decade is the *Model-Constructing Satisfiability* calculus (MCSAT). In the MCSAT approach, the SMT solver tries to construct a *theory model* step-by-step, similarly to how SAT solvers build Boolean models. Here, theory reasoning is used to assess the consistency of partial assignments, to provide explanations of infeasibility, to decide theory variables, and to propagate theory constraints.

A quite sensitive part of the MCSAT algorithm is the decision of theory variables. As for the SAT case, where such topic has been extensively discussed [68], this choice can have a tremendous impact on the success of the search. Differently from the SAT case, however, where the choices are just two ($\top$ or $\bot$), for some SMT theories there can even be infinitely many feasible assignments. MCSAT-based solvers have adopted some heuristics in order to privilege certain assignments over others (in the case of $\mathcal{NRA}$, for example, preferring integer values over polynomial root values will ease future computations). Such heuristics,

however, usually do not decide on the basis of which assignment is *more likely* to lead to an actual model; rather, they decide on the basis of which assignment is *cheaper*.

In the Logic-to-Optimization framework (L2O), a logical formula is mapped into a cost function that represents the *distance from a model*, and numerical minimization methods are used to find candidate models with a small cost, that are therefore more likely to be actual models, or close to actual models. Such approach has already been applied successfully in the context of floating points [51] and of non-linear arithmetic (possibly augmented with transcendental functions) [72, 71, 87].

In this preliminary report, we propose an optimization-guided MCSAT framework inspired by the L2O approach, where decision steps are chosen on the basis of *hints* provided by an optimizer that tries to minimize a cost function.

Contrarily to previous works in which L2O has simply been used to statically generate initial candidate points at the beginning of the search, here, we design a tight integration between L2O and the MCSAT main algorithm. As the MCSAT trail evolves through decisions, propagations, and conflicts, the cost function automatically gets updated accordingly. Vice-versa, minimization steps performed by the optimizer on the cost function guide the MCSAT search to make decisions that are more likely to lead to a model.

We have implement a preliminary version of such method in the YICES SMT-solver [41], focusing on the theory of Non-linear Integer Arithmetic ($\mathcal{NIA}$). We have performed preliminary experiments on the whole quantifier-free $\mathcal{NIA}$ benchmark set from the SMT-LIB distribution [10]. The results demonstrates that, by adopting the new approach, YICES is able to solve more benchmarks than before, both for satisfiable and unsatisfiable cases.

6.1 Method

6.1.1 Logic-to-Optimization

Given any theory $\mathcal{T}$ for which it is possible to define a *distance* between terms, we can define a Logic-to-Optimization operator $\mathcal{L}2\mathcal{O}$ that maps a $\mathcal{T}$-formula to a $\mathcal{T}$-term which represents the distance from a model (i.e., the cost). In the following, we will limit to the case of $\mathcal{T}$ being an arithmetic theory.

First, we introduce a function symbol d of arity 2, and we assume a fixed interpretation d that satisfies the properties of metric distance, i.e.:

- symmetry: $\mathrm{d}(a,b) = \mathrm{d}(b,a)$, for all a,b.
- positivity: $\mathrm{d}(a,b) \geq 0$, for all a,b.
- reflexivity: $\mathrm{d}(a,b) = 0$ if and only if $a = b$, for all a,b.
- triangle inequality: $\mathrm{d}(a,b) + \mathrm{d}(b,c) \geq \mathrm{d}(a,c)$, for all a,b,c.

Secondly, we assume the existence of a fixed constant term ε, such that $\varepsilon > 0$.
We recursively define $\mathcal{L}2\mathcal{O}$ as follows:

$$
\begin{array}{lcl}
\mathcal{L}2\mathcal{O}(b) & \stackrel{\text{def}}{=} & ITE(b,0,1) \\
\mathcal{L}2\mathcal{O}(\neg b) & \stackrel{\text{def}}{=} & ITE(b,1,0) \\
\mathcal{L}2\mathcal{O}(t_1 = t_2) & \stackrel{\text{def}}{=} & d(t_1,t_2) \\
\mathcal{L}2\mathcal{O}(t_1 \leq t_2) & \stackrel{\text{def}}{=} & ITE(t_1 \leq t_2,\ 0,\ d(t_1,t_2)) \\
\mathcal{L}2\mathcal{O}(t_1 < t_2) & \stackrel{\text{def}}{=} & ITE(t_1 < t_2,\ 0,\ d(t_1,t_2) + \varepsilon) \\
\mathcal{L}2\mathcal{O}(t_1 \neq t_2) & \stackrel{\text{def}}{=} & ITE(t_1 \neq t_2,\ 0,\ 1) \\
\mathcal{L}2\mathcal{O}(\phi_1 \wedge \phi_2) & \stackrel{\text{def}}{=} & \mathcal{L}2\mathcal{O}(\phi_1) + \mathcal{L}2\mathcal{O}(\phi_2) \\
\mathcal{L}2\mathcal{O}(\phi_1 \vee \phi_2) & \stackrel{\text{def}}{=} & \mathcal{L}2\mathcal{O}(\phi_1) * \mathcal{L}2\mathcal{O}(\phi_2) \\
\mathcal{L}2\mathcal{O}(ITE(\phi_c,\phi_1,\phi_2) & \stackrel{\text{def}}{=} & ITE(\phi_c,\ \mathcal{L}2\mathcal{O}(\phi_1),\ \mathcal{L}2\mathcal{O}(\phi_2)) \\
\mathcal{L}2\mathcal{O}(\neg ITE(\phi_c,\phi_1,\phi_2) & \stackrel{\text{def}}{=} & ITE(\phi_c,\ \mathcal{L}2\mathcal{O}(\neg\phi_1),\ \mathcal{L}2\mathcal{O}(\neg\phi_2))
\end{array}
$$

We say that $\mathcal{L}2\mathcal{O}(\phi)$ is the *cost function* of ϕ. Note that $\mathcal{L}2\mathcal{O}(\phi)$ is of arithmetic sort. It is easy to check that $\mathcal{L}2\mathcal{O}$ respects the following property:

Property 6.1.1. *Let ϕ be a formula, and let μ be an assignment. Then, μ satisfies ϕ if and only if $\mathcal{L}2\mathcal{O}(\phi)$ evaluates to 0 under μ.*

As a consequence of this property, the satisfiability problem of ϕ is equivalent to the problem of finding an assignment that evaluates $\mathcal{L}2\mathcal{O}(\phi)$ to 0. In the following, with a slight abuse of notation, we will denote with $\mathcal{L}2\mathcal{O}(\phi)$ also the corresponding arithmetic function determined by the interpretation d. Since, by construction, $\mathcal{L}2\mathcal{O}(\phi)$ is always non-negative, then ϕ is satisfiable if and only if 0 is a global minimum for $\mathcal{L}2\mathcal{O}(\phi)$.

6.1.2 Local minimization through Hill-climbing

Here, we explain how to perform local minimization given a cost function over the integers obtained as in the previous paragraph. We use the Hill-climbing algorithm [59], a simple yet effective minimization technique.

Given a $\mathcal{NIA}$ formula ϕ, let $f_c \stackrel{\text{def}}{=} \mathcal{L}2\mathcal{O}(\phi)$ be its cost function. Variables in f_c can be of both Boolean and integer sort. We denote the set of all the variables in f_c by $Vars(f_c)$, the set of Boolean variables by $Vars_{\mathcal{B}}(f_c)$, and the set of integer variables by $Vars_{\mathcal{I}}(f_c)$.

Given a full assignments μ on the variables $Vars(f_c)$, we call a *move from* μ an assignments μ' obtained from μ by changing the assigned value of one variable. In particular, we allow two types of moves:

- *Boolean moves*: Given $b \in Vars_{\mathcal{B}}(f_c)$, we have that the assignment $\mu_{\neg b} \stackrel{\text{def}}{=} \mu[b \mapsto \neg\mu(b)]$ obtained by mapping b to the negation of its value assigned by μ is a move from μ.
- *Integer moves*: Given $x \in Vars_{\mathcal{I}}(f_c)$, we have that the assignments $\mu_{x+1} \stackrel{\text{def}}{=} \mu[x \mapsto \mu(x)+1]$ and $\mu_{x-1} \stackrel{\text{def}}{=} \mu[x \mapsto \mu(x)-1]$ obtained by mapping x to the successor and predecessor of its value assigned by μ are moves from μ.

The basic procedure of hill-climbing is as follows. It takes in input a cost function f_c and an assignment μ, and it computes the cost $c \stackrel{\text{def}}{=} \mu[f_c]$. It cycles over all the variables in f_c, and, for each variable *var*, it tries either the one move possible (if $var \in Vars_{\mathcal{B}}(f_c)$), or two moves possible (if $var \in Vars_{\mathcal{I}}(f_c)$). For each move μ', it computes $c' \stackrel{\text{def}}{=} \mu'[f_c]$. If $c' > c$, then it updates the assignment and the cost (i.e., $\mu \leftarrow \mu'$, and $c \leftarrow c'$), and restarts the cycle; otherwise, it goes to the next variables. If no move has improved the cost, then a local minimum is found, and μ is returned.

6.1.3 Interplay with MCSAT

Here, we explain how the MCSAT procedure interacts with the L2O module, which consists of the $\mathcal{L}2\mathcal{O}$ operator and the local minimization algorithm.

We construct $\mathcal{L}2\mathcal{O}(\phi)$ one time at the beginning of the search, after the preprocessing phase. We begin with a starting assignment μ_0, that by default maps every arithmetic variable to 0, and every boolean variable to $\top$. Then, we run the local minimizer, and obtain an assignment μ_0^* which is a local minimum of $\mathcal{L}2\mathcal{O}(\phi)$ in the neighborhood of μ_0.

We pass μ_0^* back to MCSAT as a *hint*, i.e. a cached assignment that will be used during the decision phase of a variable, as long as the assigned value belongs to the set of feasible values.

After a certain number of conflicts, if the procedure has not terminated, we call the L2O module again after the last conflict has been resolved. In addition, we now pass to the L2O module the trail M, and a cache assignment (if present). Each variable x in M is assigned to

a value a_x. This partial assignment represents the current status of the model construction process, thus we do not want to change it. Hence, we *fix* each variable x in M to its respective value a_x, and we do not allow the minimizer to change the value of x. In practice, this means that, given a trail M, we are now working with a new cost function $\mathcal{L}2\mathcal{O}(\phi)_{|M}$, whose dimension is lower than the dimension of the original cost function $\mathcal{L}2\mathcal{O}(\phi)$. We then run again the minimizer, now on $\mathcal{L}2\mathcal{O}(\phi)_{|M}$. For the choice of the starting point, if a variable has an assigned cached value, we choose that; otherwise, we choose default value.

Given the new local minimum, we pass it back to MCSAT as a hint, and we repeat the process.

6.2 Experiments

Implementation. We have implemented our optimization-guided MCSAT method in the YICES SMT solver, which already supported MCSAT. We added a module for Logic-to-Optimization, and a module to perform numerical minimization on the integers based on the hill-climbing algorithm. We will denote this variant of YICES with YICESL2O.

Setup. We have run our experiments on a cluster equipped with AMD EPYC 7502 machines, using a time limit of 300 seconds, and a memory limit of 8GB. We have compared the "stock" version of YICES with the new variant, YICESL2O.

	Total (sat) (unsat)	VeryMax	calypto	ezsmt	LassoRanker	Dartagnan	LCTES	MathProblems	leipgiz	UltAutSvcomp2023	mcm	sqrtmodinv-hoenicke	UltimateAutomizer	AProVE	UltimateLassoRanker
YICES	17651 (12038) (5613)	14441 (10070) (4371)	174 (79) (95)	8 (8) (0)	92 (4) (88)	**308** (7) (301)	0 (0) (0)	122 (115) (7)	108 (106) (2)	7 (7) (0)	8 (8) (0)	0 (0) (0)	7 (0) (7)	2344 (1628) (716)	32 (6) (26)
YICES-L2O	**17953** (12283) (5670)	**14590** (10165) (4425)	**175** (79) (96)	8 (8) (0)	**94** (4) (90)	304 (6) (298)	0 (0) (0)	**268** (261) (7)	108 (106) (2)	7 (7) (0)	8 (8) (0)	0 (0) (0)	7 (0) (7)	**2352** (1633) (719)	32 (6) (26)

Table 6.1 Summary of results for SMT(NIA) benchmarks with a timeout of 300s. Strictly better results are highlighted in underline and boldface for the "Total" entries, and only in underline for the "(sat)" and "(unsat)" entries.

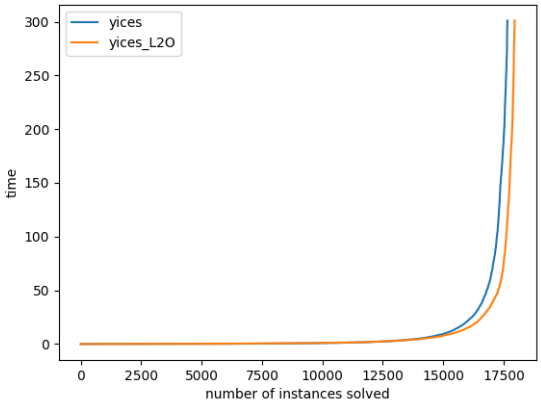

Figure 6.1 Survival plots for the experiments in Table 6.1. For each solver, the plot shows the number of instances solved (x axis) within the given time (y axis), in linear scale.

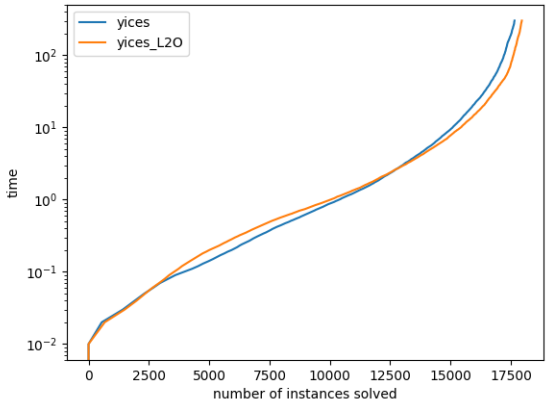

Figure 6.2 Survival plots for the experiments in Table 6.1. For each solver, the plot shows the number of instances solved (x axis) within the given time (y axis), in logarithmic scale.

Benchmarks. We have considered all the SMT-LIB benchmarks from the QF_NIA category. This is a class of 25443 benchmarks, among which 14990 come with status "sat", 5183 come with status "unsat", and 5270 come with status "unknown".

Results. We can see from the results in Table 6.1 that YICESL2O solves more benchmarks than YICES on both satisfiable and unsatisfiable cases, improving the solver performances for most of the families. From the survival plots in Figure 6.1 and Figure 6.2, we see that YICESL2O is overall faster than YICES, although for simple benchmarks it pays the price of running the local minimization procedure, allowing YICES to be faster on easy instances. Already for instances that take more than 2 seconds, however, we have the surpass of YICESL2O. We stress that our implementation is still rudimentary, and a more efficient implementation, better heuristics for the hill climbing algorithm, and a better interaction between the L2O module and the main MCSAT module, will most likely lead to better results.

6.3 Related work

The Logic-to-Optimization approach has already been applied in the context of floating-point arithmetic [51], as well as $\mathcal{NRA}$ and $\mathcal{NTA}$ [72, 71, 87], with positive results. However, in such works, the employment of L2O has been limited to providing candidate points at the beginning of the search, using a global optimization algorithm that adopts random jumps to move from one local minimum to another, reducing the role of L2O to that of a static assistant. Here, on the contrary, L2O is part of a tighter integration with the MCSAT procedure, benefiting from the conflict resolution capability of the calculus, and influencing its choices during the decision phase. Indeed, we take the best of a local minimization phase that helps guiding the model construction toward a better assignment, and the best of a conflict resolution process that moves away from the found local minimum through precise reasoning.

Another line of work closely related to ours is *local search*, which has recently been applied to SMT($\mathcal{NRA}$) [112, 70] and SMT($\mathcal{NIA}$) [22, 117]. A common trait of this approach and ours, is that both apply local changes to partial assignments. However, the criterion based on which these local changes are applied is quite different. Local search techniques rely on *critical moves*, i.e. assignment changes that increase the number of literals satisfied. Our approach, on the other side, relies on assignment changes that increase the proximity to a solution in terms of arithmetical distance. As a simple example, consider the formula $\phi \equiv (x^2 = y \wedge x = y-2)$, and the assignment $\mu = \{x \mapsto 3, y \mapsto 6\}$. The move that changes the assignment of x from $x \mapsto 3$ to $x \mapsto 2$ would not be considered as a critical move in the context of typical local search algorithms, as it does not change the truth value of any of the two literals, which still evaluate both to false trough $\mu' = \{x \mapsto 2, y \mapsto 6\}$. However, μ' assigns x to the only possible value to construct a model. This can be of essential help for decision procedures that work by substituting partial assignments to reduce to lower-dimensional sub-problems (the CAD-based MCSAT algorithm used by YICES is an example of that). Indeed, while the formula $\phi_{|x\mapsto 3} \equiv (9 = y) \wedge (3 = y-2)$ is unsatisfiable, the formula $\phi_{|x\mapsto 2} \equiv (4 = y) \wedge (2 = y-2)$ is satisfiable.

Chapter 7

Conclusion

In this thesis, we have tackled the problem of non-linear real arithmetic augmented with exponential and trigonometric functions ($\mathcal{NTA}$). We have proposed a novel approach that leverages techniques coming from the fields of numerical analysis and topology. Numerical methods are used to quickly find candidate approximate solutions, while topological methods are used to prove the existence of a solution within a given bounded region without having to explicitly express such solution. Experimental evaluation showed that the proposed approach is able to solve a significantly higher number of benchmarks than state-of-the-art SMT solvers for satisfiable $\mathcal{NTA}$ formulas.

The success of our method, compared to the state-of-the-art, can be explained as follows. Given that most solutions of $\mathcal{NTA}$ formulas contain transcendental values, and that no known finite representation for such values is available, the only viable strategy for proving the satisfiability of general $\mathcal{NTA}$ formulas is to use indirect methods, such as zero-existence theorems, that circumvent the representability issue. These methods are well-known in the mathematical community, but have not found much space in SMT solving, yet. A notable exception is RASAT, which had implemented a version of the Generalized Intermediate Value Theorem, paired with Interval Constraint Propagation. However, being restricted to $\mathcal{NRA}$, its success has been limited, due to the tough competition of complete decision procedures, such as Cylindric Algebraic Decomposition, that have dominated the scene for the last decade. For $\mathcal{NTA}$, however, such kind of procedures do not exist, and incomplete procedures, such as Interval Constraint Propagation and Incremental Linearization, fail to prove the satisfiability of even simple equations such as $e^x - 3 = 0$. This leaves zero-existence theorems with no real competitors.

The adoption of zero-existence theorems in the SMT context is, however, anything but trivial. Indeed, these results are, theoretically, limited to systems of equations that respect certain conditions on the number of variables and equations and whose domain is a bounded region, and, practically, they require such regions to be very small, in order to be sufficiently efficient. This led us to look for a way to reduce an unbounded problem to a bounded one, possibly with very tight bounds. In general, if one can settle for approximate results, numerical methods are the way to go. Being very fast and able to deal with transcendental functions with not much overhead, this made them the perfect tool to be used as a guidance toward local sub-regions of the original domain that can reasonably be expected to contain a solution. However, such methods usually do not work with full formulas, which include sets of equations and of inequalities, conjunctions and disjunctions. The missing link was Logic-to-Optimization, a technique to translate an SMT problem to a numerical minimization one, which had already been applied successfully to the theory of floating-points. This way, we obtained a well-rounded procedure for proving satisfiability in $\mathcal{NTA}$.

This procedure performed extremely well on satisfiable $\mathcal{NTA}$ benchmarks, being able to find solutions for a huge amount of problems for which none of the state-of-the-art SMT solvers succeeded. Still, it is theoretically guaranteed that there exist problems for which also our procedure is unable to find a solution; this is due to the general undecidability of $\mathcal{NTA}$. We thus decided to study the class of problems for which our procedure is capable of finding a solution. We provided a theoretical characterization, and found out that such class is quite huge. Indeed, it suffices that the problem respects some reasonable robustness assumptions. While some mathematical problems do not respect such assumptions, they are quite rare in practical applications.

Finally, we decided to study the Logic-to-Optimization approach more in-depth (which, in the case of $\mathcal{NTA}$, had mostly been used as a static assistant), how can it be used for a more general class of theories, and how can it be integrated into a comprehensive framework, such as MCSAT. While this last part is still a work-in-progress, the optimization-guided MCSAT approach has already shown promising results on $\mathcal{NIA}$ benchmarks.

7.1 Future directions

Now, we discuss possible improvements over the methods presented in this work, and we outline possible directions that can stem from the ideas presented in the thesis.

First, we note that the procedures presented in the thesis have been implemented in the prototype tool UGOTNL, which has served as a proof-of-concept, but that is still far from being a full-rounded SMT solver. While we plan to further improve UGOTNL and make it more robust, we also hope that the techniques presented here could be of inspiration for, and possibly be adopted by, top-level SMT solvers. Adopting these techniques inside modern SMT-solvers is not trivial, though. One of the reasons is that, to the best of our knowledge, the only available implementation of the topological degree test is TOPDEG, written in OCaml, thus not easily integrable into solvers written in C/C++. A possible solution would be to use other tests for zero-existence, such as Miranda's or Borsuk's, which, although weaker than the topological degree test, are easier to implement.

We see several ways to further leverage techniques based on zero-existence theorems. One possibility is to use a different method than Logic-to-Optimization in order to reduce to small sub-regions. For example, tools based on Interval Constraint Propagation, such as ISAT3 and DREAL, although in general not able to prove satisfiability, can return a candidate solution, which could then be used to construct a box over which to run a zero-existence test.

Moreover, the use of indirect methods such as the topological degree test to witness the existence of a model without actually returning a model but only a box that contains the model, poses very interesting research questions. For example, how can this kind of answer be used in the context of SMT-based model checking? It is often required for the back-end SMT-solver to be able to return a model when the query is falsifiable. But for $\mathcal{NTA}$, at best, the solver can return a certificate, or an approximate solution.

Another very interesting research direction is how to tackle $\mathcal{NTA}$ within the MCSAT framework. There are several aspects to keep in consideration. First, MCSAT works by constructing a model step-by-step, i.e., it incrementally assigns values to the variables until a satisfying assignment is found. For $\mathcal{NTA}$, however, finding a complete assignment is, in general, impossible. Yet, also in our method, partial assignments are needed to reduce to a well-constrained system of equations. This suggests that the model construction phase should probably be complemented with a stopping criterion, which, when the number of variables that remain unassigned is equal to the number of equations containing at least one unassigned variable, then shall halt the decision phase, and pass the ball to a zero-existence

test. Secondly, in order to ensure termination, MCSAT requires the explanation function to satisfy the *finite-basis property*, i.e. being able to generate explanation constraints that come from a finite set of constraints. Whether this property holds, at least for certain sub-classes of $\mathcal{NTA}$, is an interesting question that should be further investigated. In general, while, for $\mathcal{NRA}$, explanations are based on CAD, it is not trivial to find what a good substitute could be for $\mathcal{NTA}$, even dropping the termination requirement (that we know is unfulfillable, due to the undecidability of $\mathcal{NTA}$). Thirdly, MCSAT requires an algorithm able to determine the set of feasible values for a variable. An idea that we believe to be very promising, is to leverage real root isolation methods for univariate functions. Recently, lots of research has been made on relevant univariate fragments of $\mathcal{NTA}$ (e.g. exp–log–arctan functions, tame elementary functions, poly-powers, mixed trigonometric-polynomials) for which real root isolation methods exist. These techniques would go along well with the model-constructing framework. Another option would be to use some interval arithmetic-based technique, possibly weakening the condition that such an algorithm must be able to precisely separate the feasible and unfeasible sets, allowing it to work with *feasible*, *unfeasible*, and *maybe-feasible* sets. Interval Arithmetic could serve the purpose of producing explanations, too.

A further line of research that we envision, is a broader adoption of the Logic-to-Optimization approach in the context of SMT solving. We have started working on this line in Chapter 6, by proposing an optimization-guided approach to MCSAT. This approach could potentially be adapted to most theories, and rival the growing popularity of local search techniques. We see two main challenges to this approach: how to minimize the run-time of the local minimization routines (e.g., concentrate only on a subset of possible moves/directions), and how to increase the synergy between the model-construction phase and the L2O phase (e.g., how many calls to L2O, after how many conflicts, on the whole formula or just on a subset of constraints, etc.). Given that this line of research is still at its very early stages, we believe that further research on heuristics, as well as novel ideas on design choices, could enhance this framework to become a standard approach to MCSAT.

References

[1] Oliver Aberth. Computation of Topological Degree Using Interval Arithmetic, and Applications. *Math. Comput.*, 62(205):171–178, 1994.

[2] Erika Ábrahám, James Davenport, Matthew England, and Gereon 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, 11 2020.

[3] Samy Ait-Aoudia, Roland Jégou, and Dominique Michelucci. Reduction of constraint systems. *CoRR*, abs/1405.6131, 2014.

[4] Behzad Akbarpour and Lawrence Charles Paulson. Metitarski: An automatic theorem prover for real-valued special functions. *Journal of Automated Reasoning*, 44(3):175–205, Mar 2010.

[5] Alessandro Armando and Enrico Giunchiglia. Embedding complex decision procedures inside an interactive theorem prover. *Annals of Mathematics and Artificial Intelligence*, 8(3):475–502, Sep 1993.

[6] Stanley Bak, Sergiy Bogomolov, and Taylor T. Johnson. HYST: A Source Transformation and Translation Tool for Hybrid Automaton Models. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, HSCC '15, page 128–133, New York, NY, USA, 2015. Association for Computing Machinery.

[7] Haniel Barbosa, Clark Barrett, Byron Cook, Bruno Dutertre, Gereon Kremer, Hanna Lachnitt, Aina Niemetz, Andres Nötzli, Alex Ozdemir, Mathias Preiner, Andrew Reynolds, Cesare Tinelli, and Yoni Zohar. Generating and Exploiting Automated Reasoning Proof Certificates. *Commun. ACM*, 66(10):86–95, 2023.

[8] Haniel Barbosa, Clark W. Barrett, Martin Brain, Gereon Kremer, Hanna Lachnitt, Makai Mann, Abdalrhman Mohamed, Mudathir Mohamed, Aina Niemetz, Andres Nötzli, Alex Ozdemir, Mathias Preiner, Andrew Reynolds, Ying Sheng, Cesare Tinelli, and Yoni Zohar. cvc5: A Versatile and Industrial-Strength SMT Solver. In Dana Fisman and Grigore Rosu, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 28th International Conference, TACAS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2-7, 2022, Proceedings, Part I*, volume 13243 of *Lecture Notes in Computer Science*, pages 415–442. Springer, 2022.

[9] Haniel Barbosa, Andrew Reynolds, Gereon Kremer, Hanna Lachnitt, Aina Niemetz, Andres Nötzli, Alex Ozdemir, Mathias Preiner, Arjun Viswanathan, Scott Viteri, Yoni Zohar, Cesare Tinelli, and Clark Barrett. Flexible Proof Production In An Industrial-Strength SMT Solver. In *Automated Reasoning: 11th International Joint Conference, IJCAR 2022, Haifa, Israel, August 8–10, 2022, Proceedings*, page 15–35, Berlin, Heidelberg, 2022. Springer-Verlag.

[10] Clark Barrett, Pascal Fontaine, and Cesare Tinelli. The Satisfiability Modulo Theories Library (SMT-LIB). `www.SMT-LIB.org`, 2016.

[11] Massimo Bartoletti, Angelo Ferrando, Enrico Lipparini, and Vadim Malvone. Solvent: Liquidity verification of smart contracts. In Nikolai Kosmatov and Laura Kovács, editors, *Integrated Formal Methods*, pages 256–266, Cham, 2025. Springer Nature Switzerland.

[12] Frédéric Benhamou and Laurent Granvilliers. Chapter 16 - Continuous and Interval Constraints. In Francesca Rossi, Peter van Beek, and Toby Walsh, editors, *Handbook of Constraint Programming*, volume 2 of *Foundations of Artificial Intelligence*, pages 571–603. Elsevier, 2006.

[13] Nikolaj Bjørner and Lev Nachmanson. Arithmetic Solving in Z3. In Arie Gurfinkel and Vijay Ganesh, editors, *Computer Aided Verification*, pages 26–41, Cham, 2024. Springer Nature Switzerland.

[14] Miquel Bofill, Robert Nieuwenhuis, Albert Oliveras, Enric Rodríguez-Carbonell, and Albert Rubio. The Barcelogic SMT Solver. In Aarti Gupta and Sharad Malik, editors,

Computer Aided Verification, pages 294–298, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[15] Bernard Bolzano. Rein analytischer Beweis des Lehrsatzes, daß zwischen je zwey Werthen, die ein entgegengesetztes Resultat gewähren, wenigstens eine reele Wurzel der Gleichung liege. In *Gotlieb Hasse*, 1817. [English translation: A translation of Bolzano's paper on the intermediate value theorem, S.B. Russ (1980), *Historia Mathematica*].

[16] Thomas Bouton, Diego Caminha B. de Oliveira, David Déharbe, and Pascal Fontaine. veriT: An Open, Trustable and Efficient SMT-Solver. In Renate A. Schmidt, editor, *Automated Deduction – CADE-22*, pages 151–156, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[17] Franz Brauße, Konstantin Korovin, Margarita Korovina, and Norbert Müller. A CDCL-Style Calculus for Solving Non-linear Constraints. In Andreas Herzig and Andrei Popescu, editors, *Frontiers of Combining Systems*, pages 131–148, Cham, 2019. Springer International Publishing.

[18] Franz Brauße, Konstantin Korovin, Margarita Korovina, and Norbert Müller. *The ksmt Calculus Is a δ-complete Decision Procedure for Non-linear Constraints*, volume 12699 of *Lecture Notes in Computer Science*, pages 113–130. Springer, 2021.

[19] Roberto Bruttomesso, Edgar Pek, Natasha Sharygina, and Aliaksei Tsitovich. The OpenSMT Solver. In Javier Esparza and Rupak Majumdar, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 150–153, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[20] Randal E. Bryant, Daniel Kroening, Joël Ouaknine, Sanjit A. Seshia, Ofer Strichman, and Bryan Brady. Deciding Bit-Vector Arithmetic with Abstraction. In Orna Grumberg and Michael Huth, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 358–372, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[21] Randal E. Bryant and Miroslav N. Velev. Boolean satisfiability with transitivity constraints. *ACM Trans. Comput. Logic*, 3(4):604–627, oct 2002.

[22] Shaowei Cai, Bohan Li, and Xindi Zhang. Local Search For Satisfiability Modulo Integer Arithmetic Theories. *ACM Trans. Comput. Logic*, 24(4), July 2023.

[23] C.Barrett et al. CVC5 at the SMT Competition 2021. https://smt-comp.github.io/2021/system-descriptions/cvc5.pdf, 2021.

[24] Rizeng Chen, Haokun Li, Bican Xia, Tianqi Zhao, and Tao Zheng. Isolating all the real roots of a mixed trigonometric-polynomial. *Journal of Symbolic Computation*, 121:102250, 2024.

[25] Rizeng Chen and Bican Xia. Deciding first-order formulas involving univariate mixed trigonometric-polynomials. *Proceedings of the 2023 International Symposium on Symbolic and Algebraic Computation*, 2023.

[26] Rizeng Chen and Bican Xia. Reduction of transcendental decision problems over the reals. In *Proceedings of the 2024 International Symposium on Symbolic and Algebraic Computation*, ISSAC '24, page 56–64, New York, NY, USA, 2024. Association for Computing Machinery.

[27] Jürgen Christ, Jochen Hoenicke, and Alexander Nutz. SMTInterpol: An Interpolating SMT Solver. In Alastair Donaldson and David Parker, editors, *Model Checking Software*, pages 248–254, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[28] Alessandro Cimatti, Alberto Griggio, Ahmed Irfan, Marco Roveri, and Roberto Sebastiani. Satisfiability Modulo Transcendental Functions via Incremental Linearization. In Leonardo de Moura, editor, *Automated Deduction – CADE 26*, pages 95–113, Cham, 2017. Springer International Publishing.

[29] Alessandro Cimatti, Alberto Griggio, Ahmed Irfan, Marco Roveri, and Roberto Sebastiani. Experimenting on Solving Nonlinear Integer Arithmetic with Incremental Linearization. In Olaf Beyersdorff and Christoph M. Wintersteiger, editors, *Theory and Applications of Satisfiability Testing – SAT 2018*, pages 383–398, Cham, 2018. Springer International Publishing.

[30] Alessandro Cimatti, Alberto Griggio, Ahmed Irfan, Marco Roveri, and Roberto Sebastiani. Incremental Linearization for Satisfiability and Verification Modulo Nonlinear Arithmetic and Transcendental Functions. *ACM Trans. Comput. Logic*, 19(3), aug 2018.

[31] Alessandro Cimatti, Alberto Griggio, Bastiaan Schaafsma, and Roberto Sebastiani. The MathSAT5 SMT Solver. In Nir Piterman and Scott Smolka, editors, *Proceedings of TACAS*, volume 7795 of *LNCS*. Springer, 2013.

[32] George E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decompostion. In H. Brakhage, editor, *Automata Theory and Formal Languages*, pages 134–183, Berlin, Heidelberg, 1975. Springer Berlin Heidelberg.

[33] Sylvain Conchon, Albin Coquereau, Mohamed Iguernlala, and Alain Mebsout. Alt-Ergo 2.2. In *SMT Workshop: International Workshop on Satisfiability Modulo Theories*, Oxford, United Kingdom, July 2018.

[34] Florian Corzilius, Gereon Kremer, Sebastian Junges, Stefan Schupp, and Erika Ábrahám. SMT-RAT: An Open Source C++ Toolbox for Strategic and Parallel SMT Solving. In *SAT*, 09 2015.

[35] James H. Davenport and Joos Heintz. Real quantifier elimination is doubly exponential. *Journal of Symbolic Computation*, 5(1):29–35, 1988.

[36] Florent de Dinechin, Christoph Lauter, and Guillaume Melquiond. Certifying the floating-point implementation of an elementary function using gappa. *IEEE Transactions on Computers*, 60(2):242–253, 2011.

[37] Leonardo De Moura and Nikolaj Bjørner. Z3: An Efficient SMT Solver. In *Proceedings of the Theory and Practice of Software, 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, TACAS'08/ETAPS'08, page 337–340, Berlin, Heidelberg, 2008. Springer-Verlag.

[38] Leonardo Mendonça de Moura and Dejan Jovanovic. A Model-Constructing Satisfiability Calculus. In *VMCAI*, 2013.

[39] George Dinca and Jean Mawhin. *Brouwer Degree: The Core of Nonlinear Analysis*. Birkhäuser, 2021.

[40] A. L. Dulmage and N. S. Mendelsohn. Coverings of Bipartite Graphs. *Canadian Journal of Mathematics*, 10:517–534, 1958.

[41] Bruno Dutertre. Yices 2.2. In Armin Biere and Roderick Bloem, editors, *Computer-Aided Verification (CAV'2014)*, volume 8559 of *Lecture Notes in Computer Science*, pages 737–744. Springer, July 2014.

[42] Andreas Eggers, Evgeny Kruglov, Stefan Kupferschmid, Karsten Scheibler, Tino Teige, and Christoph Weidenbach. Superposition modulo non-linear arithmetic. In Cesare Tinelli and Viorica Sofronie-Stokkermans, editors, *Frontiers of Combining Systems*, pages 119–134, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[43] Irene Fonseca and Wilfrid Gangbo. *Degree Theory in Analysis and Applications*. Clarendon Press, Oxford, 1995.

[44] Pascal Fontaine, Mizuhito Ogawa, Thomas Sturm, and Xuan Tung Vu. Subtropical Satisfiability. In Clare Dixon and Marcelo Finger, editors, *Frontiers of Combining Systems*, pages 189–206, Cham, 2017. Springer International Publishing.

[45] Peter Franek, Marek Krčál, and Hubert Wagner. Solving equations and optimization problems with uncertainty. *Journal of Applied and Computational Topology*, 1(3):297–330, June 2018.

[46] Peter Franek and Stefan Ratschan. Effective Topological Degree Computation Based on Interval Arithmetic. *Mathematics of Computation*, 84:1265–1290, 2015.

[47] Peter Franek, Stefan Ratschan, and Piotr Zgliczynski. Satisfiability of Systems of Equations of Real Analytic Functions Is Quasi-decidable. In Filip Murlak and Piotr Sankowski, editors, *Mathematical Foundations of Computer Science 2011*, pages 315–326, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[48] Peter Franek, Stefan Ratschan, and Piotr Zgliczynski. Quasi-decidability of a Fragment of the First-order Theory of Real Numbers. *Journal of Automated Reasoning*, 57(2):157–185, 2016.

[49] Florian Frohn and Jürgen Giesl. Satisfiability Modulo Exponential Integer Arithmetic. In Christoph Benzmüller, Marijn J.H. Heule, and Renate A. Schmidt, editors, *Automated Reasoning*, pages 344–365, Cham, 2024. Springer Nature Switzerland.

[50] Martin Fränzle, Christian Herde, Tino Teige, Stefan Ratschan, and Tobias Schubert. Efficient Solving of Large Non-Linear Arithmetic Constraint Systems with Complex Boolean Structure. *JSAT*, 1:209–236, 2007.

[51] Zhoulai Fu and Zhendong Su. XSat: A Fast Floating-Point Satisfiability Solver. In *CAV*, volume 9780 of *Lecture Notes in Computer Science*, pages 187–209. Springer, 2016.

[52] Carsten Fuhs, Jürgen Giesl, Aart Middeldorp, Peter Schneider-Kamp, René Thiemann, and Harald Zankl. SAT Solving for Termination Analysis with Polynomial Interpretations. In João Marques-Silva and Karem A. Sakallah, editors, *Theory and Applications of Satisfiability Testing – SAT 2007*, pages 340–354, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[53] Sicun Gao, Jeremy Avigad, and Edmund M. Clarke. δ-complete decision procedures for satisfiability over the reals. In *IJCAR*, volume 7364 of *Lecture Notes in Computer Science*, pages 286–300. Springer, 2012.

[54] Sicun Gao, Soonho Kong, and Edmund M. Clarke. dReal: An SMT Solver for Nonlinear Theories over the Reals. In Maria Paola Bonacina, editor, *Automated Deduction—CADE-24*, pages 208–214, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[55] Jürgen Giesl, Peter Schneider-Kamp, and René Thiemann. AProVE 1.2: Automatic Termination Proofs in the Dependency Pair Framework. In Ulrich Furbach and Natarajan Shankar, editors, *Automated Reasoning*, pages 281–286, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[56] Laurent Granvilliers and Frédéric Benhamou. Algorithm 852: RealPaver: an interval solver using constraint satisfaction techniques. *ACM Trans. Math. Softw.*, 32(1):138–156, March 2006.

[57] P. Hall. On Representatives of Subsets. *Journal of the London Mathematical Society*, s1-10(1):26–30, 1935.

[58] E. Hansen. *Global Optimization Using Interval Analysis*. Marcel Dekker, New York, 1992.

[59] Leticia Hernando, Alexander Mendiburu, and Jose Lozano. Hill-Climbing Algorithm: Let's Go for a Walk Before Finding the Optimum. pages 1–7, 07 2018.

[60] Cheng-Chao Huang, Jing-Cao Li, Ming Xu, and Zhi-Bin Li. Positive root isolation for poly-powers by exclusion and differentiation. *Journal of Symbolic Computation*, 85:148–169, 2018. 41th International Symposium on Symbolic and Alge-braic Computation (ISSAC'16).

[61] Ahmed Irfan. *Incremental Linearization for Satisfiability and Verification Modulo Nonlinear Arithmetic and Transcendental Functions*. PhD thesis, University of Trento, 2018.

[62] Martin Jonáš and Jan Strejček. Solving Quantified Bit-Vector Formulas Using Binary Decision Diagrams. In Nadia Creignou and Daniel Le Berre, editors, *Theory and Applications of Satisfiability Testing – SAT 2016*, pages 267–283, Cham, 2016. Springer International Publishing.

[63] Dejan Jovanović. Solving Nonlinear Integer Arithmetic with MCSAT. In Ahmed Bouajjani and David Monniaux, editors, *Verification, Model Checking, and Abstract Interpretation*, pages 330–346, Cham, 2017. Springer International Publishing.

[64] Dejan Jovanović and Leonardo de Moura. Solving Non-Linear Arithmetic. *ACM Commun. Comput. Algebra*, 46(3/4):104–105, jan 2013.

[65] R. Baker Kearfott. On Proving Existence of Feasible Points in Equality Constrained Optimization Problems. *Mathematical Programming*, 83(1):89–100, 1998.

[66] Gereon Kremer, Florian Corzilius, and Erika Ábrahám. A Generalised Branch-and-Bound Approach and Its Application in SAT modulo nonlinear integer arithmetic. In Vladimir P. Gerdt, Wolfram Koepf, Werner M. Seiler, and Evgenii V. Vorozhtsov, editors, *Computer Algebra in Scientific Computing - 18th International Workshop, CASC 2016, Bucharest, Romania, September 19-23, 2016, Proceedings*, volume 9890 of *Lecture Notes in Computer Science*, pages 315–335. Springer, 2016.

[67] Gereon Kremer, Andrew Reynolds, Clark Barrett, and Cesare Tinelli. Cooperating Techniques for Solving Nonlinear Real Arithmetic in the cvc5 SMT Solver (System Description). In Jasmin Blanchette, Laura Kovács, and Dirk Pattinson, editors, *Automated Reasoning*, pages 95–105, Cham, 2022. Springer International Publishing.

[68] Oliver Kullmann. Fundaments of Branching Heuristics. *Frontiers in Artificial Intelligence and Applications*, 185, 01 2009.

[69] John M. Lee. *Introduction to Smooth Manifolds*. Springer, 2000.

[70] Haokun Li, Bican Xia, and Tianqi Zhao. Local Search for Solving Satisfiability of Polynomial Formulas. In Constantin Enea and Akash Lal, editors, *Computer Aided Verification*, pages 87–109, Cham, 2023. Springer Nature Switzerland.

[71] Enrico Lipparini, Alessandro Cimatti, Alberto Griggio, and Roberto Sebastiani. Handling Polynomial and Transcendental Functions in SMT via Unconstrained Optimisation and Topological Degree Test. In Ahmed Bouajjani, Lukáš Holík, and Zhilin Wu, editors, *Automated Technology for Verification and Analysis*, pages 137–153, Cham, 2022. Springer International Publishing.

[72] Enrico Lipparini and Stefan Ratschan. Satisfiability of Non-linear Transcendental Arithmetic as a Certificate Search Problem. In Kristin Yvonne Rozier and Swarat Chaudhuri, editors, *NASA Formal Methods*, pages 472–488, Cham, 2023. Springer Nature Switzerland.

[73] Enrico Lipparini and Stefan Ratschan. Satisfiability of Non-linear Transcendental Arithmetic as a Certificate Search Problem (extended version). In *Journal of Automated Reasoning*, 2024. Under minor revision.

[74] Victor Magron. Nlcertify: A tool for formal nonlinear optimization. In Hoon Hong and Chee Yap, editors, *Mathematical Software – ICMS 2014*, pages 315–320, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[75] Érik Martin-Dorel and Guillaume Melquiond. Proving tight bounds on univariate expressions with elementary functions in coq. *J. Autom. Reason.*, 57(3):187–217, October 2016.

[76] G. Mayer. Epsilon-inflation in Verification Algorithms. *Journal of Computational and Applied Mathematics*, 60:147–169, 1994.

[77] Jan Mayer, Andreas Frommer, and Gerhard Heindl. On the Existence Theorems of Kantorovich, Miranda and Borsuk. *Electronic transactions on numerical analysis ETNA*, 17, 01 2003.

[78] Scott McCallum and Volker Weispfenning. Deciding polynomial-transcendental problems. *Journal of Symbolic Computation*, 47(1):16–31, 2012.

[79] R.M. McConnell, K. Mehlhorn, S. Näher, and P. Schweitzer. Certifying algorithms. *Computer Science Review*, 5(2):119 – 161, 2011.

[80] Guillaume Melquiond. Coq-interval. `https://coqinterval.gitlabpages.inria.fr/`, 2011.

[81] John W. Milnor. *Topology from the Differentiable Viewpoint*. Princeton University Press, 1997.

[82] David D. L. Minh and Do Le (Paul) Minh. Understanding the Hastings Algorithm. *Communications in Statistics - Simulation and Computation*, 44(2):332–349, 2015.

[83] Carlo Miranda. Un'osservazione su un teorema di Brouwer. *Boll. Unione Mat. Ital., II. Ser.*, 3:5–7, 1940.

[84] Ramon E. Moore, R. Baker Kearfott, and Michael J. Cloud. *Introduction to Interval Analysis*. SIAM, 2009.

[85] James R. Munkres. *Analysis On Manifolds*. CRC Press, 1991.

[86] Arnold Neumaier. *Interval Methods for Systems of Equations*. Cambridge University Press, Cambridge, 1991.

[87] Xinpeng Ni, Yulun Wu, and Bican Xia. Solving SMT over Non-linear Real Arithmetic via Numerical Sampling and Symbolic Verification. In *Dependable Software Engineering. Theories, Tools, and Applications: 9th International Symposium, SETTA 2023, Nanjing, China, November 27–29, 2023, Proceedings*, page 171–188, Berlin, Heidelberg, 2023. Springer-Verlag.

[88] Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. Abstract DPLL and Abstract DPLL Modulo Theories. In Franz Baader and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning*, pages 36–50, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[89] Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. Solving SAT and SAT Modulo Theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T). *J. ACM*, 53(6):937–977, November 2006.

[90] D. O'Regan, Cho Yeol Je, and Yuqing Chen. *Topological Degree Theory and Applications*. Taylor and Francis, 03 2006.

[91] Stefan Ratschan. Efficient solving of quantified inequality constraints over the real numbers. *ACM Trans. Comput. Logic*, 7(4):723–748, October 2006.

[92] Stefan Ratschan. Applications of Quantified Constraint Solving over the Reals - Bibliography, 2012.

[93] Stefan Ratschan. Safety Verification of Non-Linear Hybrid Systems is Quasi-Decidable. *Form. Methods Syst. Des.*, 44(1):71–90, feb 2014.

[94] Daniel Richardson. Some undecidable problems involving elementary functions of a real variable. *J. Symb. Log.*, 33(4):514–520, 1968.

[95] Nima Roohi, Pavithra Prabhakar, and Mahesh Viswanathan. HARE: A Hybrid Abstraction Refinement Engine for Verifying Non-linear Hybrid Automata. In Axel Legay and Tiziana Margaria, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 573–588, Berlin, Heidelberg, 2017. Springer Berlin Heidelberg.

[96] Paolo Ruffini. *Riflessioni intorno alla soluzione delle equazioni algebriche generali*. Società Tipografica, 1813.

[97] Siegfried M. Rump. Verification Methods: Rigorous Results Using Floating-Point Arithmetic. *Acta Numerica*, page 3–4, 2010.

[98] Ömer Sali. Linearization Techniques for Nonlinear Arithmetic Problems in SMT. Master's thesis, RWTH Aachen University, 2023.

[99] Stefan Schupp, Erika Ábrahám, Peter Rossmanith, Ulrich Loup, and Florian Corzilius. Interval Constraint Propagation in SMT Compliant Decision Procedures. 2013.

[100] Roberto Sebastiani. Lazy satisfiability modulo theories. *Journal on Satisfiability, Boolean Modeling and Computation*, 3:141–224, 2007. 3-4.

[101] Amar Shah, Federico Mora, and Sanjit A. Seshia. An Eager Satisfiability Modulo Theories Solver for Algebraic Datatypes. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan, editors, *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial*

Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada, pages 8099–8107. AAAI Press, 2024.

[102] Alexey Solovyev and Thomas C. Hales. Formal verification of nonlinear inequalities with taylor interval approximations. In Guillaume Brat, Neha Rungta, and Arnaud Venet, editors, *NASA Formal Methods*, pages 383–397, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[103] Philippe Specht. Improving Incremental Linearization For Satisfiability Modulo Non-linear Real Arithmetic Checking. Master's thesis, RWTH Aachen University, 2023.

[104] Ofer Strichman. On Solving Presburger and Linear Arithmetic with SAT. In Mark D. Aagaard and John W. O'Leary, editors, *Formal Methods in Computer-Aided Design*, pages 160–170, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[105] Ofer Strichman, Sanjit A. Seshia, and Randal E. Bryant. Deciding Separation Formulas with SAT. In Ed Brinksma and Kim Guldstrand Larsen, editors, *Computer Aided Verification*, pages 209–222, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[106] Adam Strzebonski. Real root isolation for tame elementary functions. In *Proceedings of the 2009 International Symposium on Symbolic and Algebraic Computation*, ISSAC '09, page 341–350, New York, NY, USA, 2009. Association for Computing Machinery.

[107] Adam Strzeboski. Real root isolation for exp-log-arctan functions. *J. Symb. Comput.*, 47(3):282–314, mar 2012.

[108] Thomas Sturm. Subtropical Real Root Finding. In *Proceedings of the 2015 ACM International Symposium on Symbolic and Algebraic Computation*, ISSAC '15, page 347–354, New York, NY, USA, 2015. Association for Computing Machinery.

[109] Xuan Tung Vu. *SMT Solving for Polynomial Constraints*. PhD thesis, School of Information Science Japan Advanced Institute of Science and Technology, 2018.

[110] Xuan Tung Vu, To Khanh, and Mizuhito Ogawa. raSAT: an SMT solver for polynomial constraints. *Formal Methods in System Design*, 51, 12 2017.

[111] David J. Wales and Jonathan 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*, 101(28):5111–5116, 1997.

[112] Zhonghan Wang, Bohua Zhan, Bohan Li, and Shaowei Cai. Efficient Local Search for Nonlinear Real Arithmetic. In Rayna Dimitrova, Ori Lahav, and Sebastian Wolff, editors, *Verification, Model Checking, and Abstract Interpretation*, pages 326–349, Cham, 2024. Springer Nature Switzerland.

[113] Christoph Weidenbach, Dilyana Dimova, Arnaud Fietzke, Rohit Kumar, Martin Suda, and Patrick Wischnewski. Spass version 3.5. In Renate A. Schmidt, editor, *Automated Deduction – CADE-22*, pages 140–145, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[114] Volker Weispfenning. The complexity of linear problems in fields. *Journal of Symbolic Computation*, 5(1):3–27, 1988.

[115] Volker Weispfenning. Quantifier elimination for real algebra—the cubic case. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation*, ISSAC '94, page 258–263, New York, NY, USA, 1994. Association for Computing Machinery.

[116] Aklima Zaman. Incremental Linearization for SAT Modulo Real Arithmetic Solving. Master's thesis, RWTH Aachen University, 2019.

[117] Xindi Zhang, Bohan Li, and Shaowei Cai. Deep Combination of CDCL(T) and Local Search for Satisfiability Modulo Non-Linear Integer Arithmetic Theory. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ICSE '24, New York, NY, USA, 2024. Association for Computing Machinery.