Quantified Linear and Polynomial Arithmetic Satisfiability via Template-based Skolemization

Krishnendu Chatterjee[1], **Ehsan Kafshdar Goharshady**[1], **Mehrdad Karrabi**[1], **Harshit J Motwani**[2], **Maximilian Seeliger**[3], **Đorđe Žikelić**[4]

[1]Institute of Science and Technology Austria, Klosterneuburg, Austria
[2]Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong
[3]Vienna University of Technology, Vienna, Austria
[4]Singapore Management University, Singapore, Singapore
ehsan.goharshady@ist.ac.at

Abstract

The problem of checking satisfiability of linear real arithmetic (LRA) and non-linear real arithmetic (NRA) formulas has broad applications, in particular, they are at the heart of logic-related applications such as logic for artificial intelligence, program analysis, etc. While there has been much work on checking satisfiability of unquantified LRA and NRA formulas, the problem of checking satisfiability of quantified LRA and NRA formulas remains a significant challenge. The main bottleneck in the existing methods is a computationally expensive quantifier elimination step. In this work, we propose a novel method for efficient quantifier elimination in quantified LRA and NRA formulas. We propose a template-based Skolemization approach, where we automatically synthesize linear/polynomial Skolem functions in order to eliminate quantifiers in the formula. The key technical ingredients in our approach are Positivstellensätze theorems from algebraic geometry, which allow for an efficient manipulation of polynomial inequalities. Our method offers a range of appealing theoretical properties combined with a strong practical performance. On the theory side, our method is sound, semi-complete, and runs in subexponential time and polynomial space, as opposed to existing sound and complete quantifier elimination methods that run in doubly-exponential time and at least exponential space. On the practical side, our experiments show superior performance compared to state-of-the-art SMT solvers in terms of the number of solved instances and runtime, both on LRA and on NRA benchmarks.

Introduction

Satisfiability checking for logical formulas is one of the fundamental problems in automated reasoning that arises in many applications, including logic for artificial intelligence (Alur, Henzinger, and Kupferman 2002), planning and scheduling (Kautz and Selman 1992) or deductive verification (Kroening and Strichman 2016). Satisfiability modulo theories (SMT) solvers have achieved impressive results at tackling this problem. However, while SMT solvers excel at checking satisfiability of quantifier-free formulas, many applications require reasoning about *quantified formulas*. For instance, automated reasoning problems in decision making for multi-agent systems typically involve quantified formulas, with quantifier alternation corresponding to the choices made by each agent (Alur, Henzinger, and Kupferman 2002; Chatterjee, Henzinger, and Piterman 2010). Quantified formulas also commonly arise in program verification (Gulwani, Srivastava, and Venkatesan 2008) and synthesis (Solar-Lezama et al. 2006).

The key challenge that arises in checking satisfiability of quantified formulas is the highly expensive *quantifier elimination* step, that SMT solvers need to perform in order to reduce the problem of checking satisfiability of a quantified formula to that of a quantifier-free formula. It is a classical result that, both in linear real arithmetic and in non-linear real arithmetic theories, the runtime complexity of quantifier elimination is doubly-exponential (Weispfenning 1988), making it practically infeasible and often a daunting task for SMT solvers. To overcome this challenge, some SMT solvers avoid the expensive quantifier elimination step by implementing methods such as quantifier instantiation (Reynolds et al. 2013; de Moura and Bjørner 2007; Ge, Barrett, and Tinelli 2009). However, while being sound, quantifier instantiation is incomplete and may lead to "Unknown" outputs by the SMT solver. Another approach to satisfiability checking that avoids the quantifier elimination step is to consider the game semantics of quantified first-order formulas and to treat them as two-player games (Bjørner and Janota 2015; Farzan and Kincaid 2016; Murphy and Kincaid 2024). However, these methods are only applicable to formulas in linear real arithmetic.

Besides checking satisfiability of quantified formulas, another fundamental question is to obtain *witnesses of satisfiability for existentially quantified variables*. This feature is important in many applications. In decision making for multi-agent systems, the witnesses for existentially quantified variables gives rise to a strategy of the existentially quantified agent (Alur, Henzinger, and Kupferman 2002; Chatterjee, Henzinger, and Piterman 2010). In planning and scheduling, the witnesses gives rise to a control policy (Kautz and Selman 1992). Finally, in program synthesis applications, the witnesses gives rise to a program that satisfies the desired specification (Solar-Lezama et al. 2006).

Our Approach. In this work, we propose a novel method for checking satisfiability of *quantified formulas* in *linear real arithmetic (LRA)* and *non-linear real arithmetic (NRA)*.

Rather than following the approaches discussed above that try to sidestep quantifier elimination, at the core of our approach lies a *novel method for efficient quantifier elimination* in LRA and NRA. As mentioned above, sound and complete procedures for quantifier elimination are computationally expensive and inherently lead to doubly-exponential time and exponential space complexity (Weispfenning 1988; Brown and Davenport 2007). Hence, to overcome this complexity barrier, we focus on methods that are sound and *semi-complete*. While we defer the formal definition of semi-completeness to later parts of the paper, this relaxed notion of completeness intuitively means that the method is guaranteed to prove or disprove satisfiability whenever it can be witnessed by a certificate of a certain parametrized form. Our novel quantifier elimination procedure gives rise to a method for checking satisfiability of quantified formulas in LRA and NRA that is *sound*, *semi-complete*, and runs in *subexponential time* and *polynomial space*. To the best of our knowledge, this is the first method that provides all three of these desirable features. Our method also computes *witnesses of satisfiability for existentially quantified variables*.

Method Outline. Our method assumes that a quantified formula is provided in the prenex normal form

$$\phi \ ::= \ \mathcal{Q}_1 x_1.\mathcal{Q}_2 x_2.\dots\mathcal{Q}_n x_n.p,$$

where $\mathcal{Q}_1, \mathcal{Q}_2, \dots, \mathcal{Q}_n \in \{\exists, \forall\}$ are quantifiers and p is a quantifier-free formula. The method proceeds in three steps to check if ϕ is satisfiable (alternatively, to check if ϕ is not satisfiable, we may equivalently check if $\neg\phi$ is satisfiable):

1. *Existential Quantifier Elimination via Skolemization.* Our method first removes all existential quantifiers from the formula ϕ, by replacing each existentially quantified variable x_i with a function over those variables in $x_1, \dots, x_{i-1}$ that are universally quantified. This process is called *Skolemization*, and functions used to express the existentially quantified variables via the universally quantified ones are called *Skolem functions* (Scowcroft 1988). To search for Skolem functions, our method follows what we call a *template-based Skolemization approach*, where it fixes a template for Skolem function of each existentially quantified x_i in the form of a symbolic polynomial expression over universally quantified variables in $x_1, \dots, x_{i-1}$. At this stage, the polynomial coefficients are symbolic, and the concrete values of coefficients will be computed in later steps. Computing Skolem functions corresponds to computing witnesses of satisfiability for existentially quantified variables.
2. *Universal Quantifier Elimination via Positivstellensätze.* Next, our method removes all universal quantifiers from the formula ϕ. This is achieved by using Farkas' lemma (Farkas 1902) and Positivstellensätze theorems (Handelman 1988; Putinar 1993; Krivine 1964; Stengle 1974) from algebraic geometry. The procedure results in a quantifier-free formula ϕ^{FREE} whose satisfiability also implies the satisfiability of the formula ϕ.
3. *Quantifier-free Formula Satisfiability Checking.* Finally, our method tries to prove that ϕ is satisfiable by proving that ϕ^{FREE} is satisfiable. This can be realized by using an off-the-shelf SMT solver, since SMT solvers already excel at satisfiability checking for quantifier-free formulas. If ϕ^{FREE} is proved to be satisfiable, then we conclude that ϕ is satisfiable. Otherwise, our method repeats Steps 1-3 for $\neg\phi$ to try to prove that ϕ is not satisfiable.

We implement our method for satisfiability checking and experimentally compare it against Z3 (de Moura and Bjørner 2008) and CVC5 (Barbosa et al. 2022), which are both state-of-the-art SMT solvers. Our experiments show that, when required to find witnesses for existentially quantified variables, our method is able to solve a considerably larger number of quantified formula instances at lower average runtimes, both in LRA and in NRA. Moreover, we observe a large number of *unique* proofs for examples that could not be handled by neither Z3 nor CVC5. Thus, our method provides a significant step forward in tackling satisfiability checking and witness construction for quantified formulas, in LRA and NRA.

Contributions. Our contributions are as follows:

1. *New Method for Quantifier Elimination.* We present a new method for efficient quantifier elimination in quantified formulas in LRA and NRA. Our method is based on a novel template-based Skolemization approach.
2. *Efficient Satisfiability Checking.* Based on the above, we design a new method for efficient satisfiability checking for quantified formulas in LRA and NRA. Our method is sound, semi-complete and runs in subexponential time and polynomial space, parametrized by the size of Skolem function templates. To the best of our knowledge, this is the first method that provides all three of these desirable features for LRA and NRA quantified formulas. In contrast, previous sound and complete procedures have doubly-exponential time complexity and at least exponential space complexity. Moreover, our method also produces witnesses of satisfiability for the existentially quantified variables in the quantified formula.
3. *Experimental Evaluation.* Our experiments showcase a strong practical performance of our method and a considerable improvement in the number of successful satisfiability checks, runtime, as well as unique satisfiability checks over two state-of-the-art SMT solvers.

Related Work

The key step in existential quantifier elimination within most SMT solvers and computer-algebra systems is the so-called projection process. For LRA, this projection typically relies on Fourier-Motzkin elimination (Dantzig, Eaves et al. 1972). For NRA, cylindrical algebraic decomposition (CAD) (Collins 1975) is commonly employed. Although these methods are sound and complete, they suffer from a doubly exponential runtime complexity in the number of formula variables. To mitigate this issue, modern SMT solvers incorporate various heuristics and algorithms. Notably, a DPLL-style approach for quantified formulas in both LRA and NRA has been proposed (Jovanović and De Moura 2013; De Moura and Jovanović 2013). This approach generalizes the CDCL method used in SAT solvers to handle first-order logic formulas. However, these methods ultimately depend on Fourier-Motzkin and CAD for projection and still suffer from doubly exponential complexity in the worst case.

Additionally, some works have utilized Gröbner bases in conjunction with CAD and Positivstellensätze theorems for existentially quantified formulas in NRA (Passmore and Jackson 2009; Corzilius et al. 2012). These approaches are also sound and complete, but again lead to doubly exponential algorithms. Furthermore, as discussed in the Introduction, some works have also proposed approaches to satisfiability checking for quantified formulas that avoid quantifier elimination. These include quantifier instantiation (Reynolds et al. 2013; de Moura and Bjørner 2007; Ge, Barrett, and Tinelli 2009) which is sound but incomplete, and the treatment of quantified first-order formulas as two-player games (Bjørner and Janota 2015; Farzan and Kincaid 2016; Murphy and Kincaid 2024) which is restricted to LRA. Finally, sKizzo (Benedetti 2004) employs a symbolic Skolemization method for quantified boolean formulas, however this problem is fundamentally different from ours as we are working with the theory of reals.

Preliminaries

In this section, we define the syntax of linear real arithmetic (LRA) and non-linear real artihmetic (NRA) that we consider in this work. Since these are standard notions, we omit the formal definitions and assume that the reader is familiar with the semantics of LRA and NRA, the notion of satisfiability of a formula, etc. In what follows, we consider a finite set $V = \{x_1, x_2, \dots, x_n\}$ of distinct real-valued variables.

Terms. The set of *terms* in LRA is defined via

$$t \ ::= \ c \mid x \mid t_1 + t_2 \mid c \cdot t,$$

whereas the set of *terms* in NRA is defined via

$$t \ ::= \ c \mid x \mid t_1 + t_2 \mid t_1 \cdot t_2,$$

where in both cases $c \in \mathbb{R}$ is a real-valued constant and x is a variable in V. Hence, while in LRA a term can only be multiplied by a real-valued constant, NRA also allows multiplication of two terms, hence giving rise to polynomials.

Predicates. In both LRA and NRA, a *predicate* (sometimes also called a quantifier-free formula) is defined by the syntax

$$p \ ::= \ t < 0 \mid t = 0 \mid p_1 \vee p_2 \mid p_1 \wedge p_2$$

where t is a term and p_1 and p_2 are also predicates. Note that logical negation $\neg$ is omitted in the above syntax, as it can be directly applied to the atomic predicates.

Formulas. Finally, a *(quantified) formula* in both LRA and NRA is defined by the syntax

$$\phi \ ::= \ \mathcal{Q}_1 x_1.\mathcal{Q}_2 x_2.\dots\mathcal{Q}_n x_n.p,$$

where $\mathcal{Q}_1, \mathcal{Q}_2, \dots, \mathcal{Q}_n \in \{\exists, \forall\}$ and p is a predicate. For each $1 \leq i \leq n$, if $\mathcal{Q}_i = \exists$ we call it the *existential quantifier*, otherwise we call it the *universal quantifier*. In what follows, we assume that the reader is familiar with the notion of *satisfiability* of a formula.

Problem Statement. We consider the problems of checking satisfiability of formulas written in LRA and NRA:

1. **Problem 1: LRA Satisfiability.** Given a formula ϕ in LRA, check whether it is satisfiable.
2. **Problem 2: NRA Satisfiability.** Given a formula ϕ in NRA, check whether it is satisfiable.

Template-based Approach to Skolemization

In this section, we first recall the definitions of Skolem functions and present classical results from real algebraic geometry that illuminate their properties. We then use these properties as the foundation and justification for our template-based approach to Skolemization in our quantifier elimination procedure for LRA and NRA formulas. In particular, these results justify our choice for using linear and polynomial templates for Skolem functions. These will also be important in establishing the semi-completeness of our satisfiability checking algorithm in the following section.

Skolem Functions. Given a formula $\phi(x_1, \cdots, x_n, y)$ in the first-order theory of reals, a *Skolem function* of ϕ is defined as a function $f : \mathbb{R}^n \to \mathbb{R}$ such that

$$\forall x_1.\cdots\forall x_n.\exists y.\, \phi(x_1, \cdots, x_n, y) \iff$$

$$\forall x_1.\cdots\forall x_n.\, \phi(x_1, \cdots, x_n, f(x_1, \cdots, x_n))$$

Skolem functions allow us to remove existentially quantified variables from the formula and to replace them with functions over the preceding universally quantified variables.

Semi-Algebraic Sets. A set $S \subseteq \mathbb{R}^n$ is said to be *semi-algebraic*, if there exist two finite sets of polynomials $P = \{p_1, \dots, p_m\}$ and $G = \{g_1, \dots, g_k\}$ over $\mathbb{R}^n$ such that

$$S = \left\{ x \in \mathbb{R}^n \ : \begin{array}{c} p_1(x) \geq 0, \dots, p_m(x) \geq 0, \\ g_1(x) = 0, \dots, g_k(x) = 0 \end{array} \right\}$$

Semi-Algebraic Functions. Let $A \subseteq \mathbb{R}^n$ and $B \subseteq \mathbb{R}^m$ be semi-algebraic sets. A function $f \colon A \to B$ is called *semi-algebraic* if its graph $\Gamma(f) := \{(x, f(x)) \mid x \in A\}$ is a semi-algebraic set in $\mathbb{R}^{n+m}$.

Theorem 1 ((Scowcroft 1988))**.** *Skolem functions for formulas in the first-order theory of reals are semi-algebraic.*

Now, based on classical results from real algebraic geometry on semi-algebraic functions (Bochnak, Coste, and Roy 2013), the following properties hold true for Skolem functions in the first-order theory of reals:

- Skolem functions are piecewise continuous (Basu, Pollack, and Roy 2006, Proposition 5.20).
- Skolem functions are bounded above by polynomials (Bochnak, Coste, and Roy 2013, Proposition 2.6.2).

Another noteworthy aspect of semi-algebraic functions to recall here: polynomial functions are also semi-algebraic.

The above properties of Skolem functions motivate us to consider polynomial functions as viable candidates for Skolem functions. In full generality, a Skolem function can be any function whose graph can be described by polynomial inequalities. However, the properties outlined above indicate that Skolem functions are piecewise continuous and bounded above by polynomials. Thus, searching for template-based polynomials as potential Skolem functions is a reasonable approach in our quantifier elimination procedure. This effectively prunes the search space for synthesizing Skolem functions for a given formula.

Algorithm

We now present our algorithm for satisfiability checking for quantified formulas in LRA and NRA, which is the main contribution of our work. Our method is based on a novel procedure for quantifier elimination in LRA and NRA. Since our underlying algorithm for LRA and NRA is the same with only minor differences in certain steps, in what follows we provide a unified presentation for both theories and only highlight the differences that are specific to LRA or to NRA.

Goal. We aim for an algorithm for satisfiability checking which satisfies the following desirable properties:

- *Soundness*, which means that the output of our algorithm is guaranteed to be correct. That is, if the algorithm outputs that the formula is "satisfiable" (resp. "unsatisfiable"), then it is indeed satisfiable (resp. "unsatisfiable").
- *Semi-completeness*, which means that our algorithm is guaranteed to return an output whenever satisfiability or unsatisfiability of the quantified formula can be proved by a witnesses of a certain form. In our case, the class of witnesses will be linear/polynomial Skolem functions used in our template-based Skolemization approach.
- *Sub-exponential time and polynomial space complexity*, parametrized by the Skolem function templates size.
- *Witnesses of satisfiability for existentially quantified variables*, since computing these is important in many applications, as discussed in the Introduction.

Algorithm Assumptions. In what follows, we assume that we are given a prenex normal form quantified formula

$$\phi \ ::= \ \mathcal{Q}_1 x_1.\mathcal{Q}_2 x_2.\dots\mathcal{Q}_n x_n.p$$

as defined in the Preliminaries, either in LRA or in NRA. Furthermore, we assume that the predicate p is given in conjunctive normal form (CNF). The CNF assumption will be needed in Step 2 of our algorithm below. Finally, we assume that the user provides a maximal polynomial degree D for Skolem function templates (formally defined below).

The rest of this section provides a detailed description of each of the three steps of our algorithm. We also illustrate each step on a running example. Algorithm 1 shows the pseudocode of our satisfiability checking prpcedure.

Algorithm 1: Satisfiability checking for quantified LRA and NRA formulas

Input: Quantified formula $\phi = \mathcal{Q}_1 x_1.\mathcal{Q}_2 x_2.\dots\mathcal{Q}_n x_n.p$ where p is in CNF.
Parameter: D, max polynomial degree of templates
Output: (`SAT`,existential model), or `UNKNOWN`

For every i where $\mathcal{Q}_i \equiv \exists$, let f_i be a polynomial over $x_1, \dots x_{i-1}$ of degree D with unknown coefficients.
Replace each occurrence of x_i with f_i, and obtain $\phi^{\text{UNIV}} := \forall x_1 \dots x_m.p^{\text{UNIV}} \equiv \forall x_1 \dots x_m.\psi_1 \wedge \dots \wedge \psi_r$.
Let $\phi^{\text{FREE}} := \mathit{True}$.
for all ψ_i **do**
  Convert ψ_i into a polynomial entailment $\Phi \Rightarrow \psi$.
  Apply Farkas, Handelman or Positivstellensätze Theorem to $\Phi \Rightarrow \psi$ in order to obtain Δ_{ψ_i}.
  $\phi^{\text{FREE}} \leftarrow \phi^{\text{FREE}} \wedge \Delta_{\psi_i}$.
end for
if ϕ^{FREE} is satisfiable **then**
  $\mathit{model} := \mathit{getModel}(\phi^{\text{FREE}})$
  Return (`SAT`, model)
else
  Return `UNKNOWN`
end if

Step 1: Existential Quantifier Elimination

In Step 1, the algorithm uses template-based Skolem functions to eliminate existential quantifiers from ϕ.

For each existentially quantified variable x_i in ϕ, let U_i denote the set of all universally quantified variables among $x_1, \dots, x_{i-1}$. Denote by $M_{i,D} = \{m_{i,1}, \dots, m_{i,k_i}\}$ the set of all monomials of degree at most D over variables in U_i.

The algorithm sets up a *template* for the Skolem function of x_i as a symbolic polynomial of degree at most D over the variables in U_i, i.e. as a polynomial expression $f_i(U_i) = \sum_{j=1}^{k_i} c_{i,j} \cdot m_{i,j}$, where $c_{i,j}$'s are *template variables* that define polynomial coefficients. At this point, the values of template variables are unknown. The concrete real values that give rise to Skolem functions for each x_i will be computed in Step 3 of the algorithm.

Finally, the algorithm constructs a purely universally quantified formula ϕ^{UNIV} from ϕ as follows. First, for each existentially quantified variable x_i, substitute each appearance of x_i in ϕ by the Skolem function template $f_i(U_i)$ that was constructed for it. Then, remove all existential quantifiers since the existentially quantified variables have already been replaced by their Skolem functions. This procedure again results in a prenex normal formula of the form

$$\phi^{\text{UNIV}} = \forall x_1.\dots\forall x_m.p^{\text{UNIV}}.$$

Since we are working over LRA or NRA and with polynomial Skolem functions, the predicate p^{UNIV} is a boolean combination of polynomial (in)equalities. Furthermore, by the algorithm assumptions, the predicate p^{UNIV} is in CNF.

Example. *Consider the following quantified formula*

$$\phi \equiv \forall x_1.\exists x_2.\exists x_3.\forall x_4. \\ ((x_4 - 1 < x_3 \vee x_2 \leq x_4) \wedge (x_4 > x_2 \vee x_3 \geq x_1)).$$

We use $D = 1$ for this example, i.e. we are looking for linear Skolem functions. The above formula contains two existentially quantified variables x_2 and x_3. The variable x_1 is the only universally quantified variable preceding x_2 and x_3, hence $U_2 = U_3 = \{x_1\}$. Thus, the algorithm sets the following templates for Skolem functions for x_2 and x_3:

$$f_2(x_1) = c_{2,1} + c_{2,2} \cdot x_1, \quad f_3(x_1) = c_{3,1} + c_{3,2} \cdot x_1$$

The algorithm substitutes these Skolem functions templates into ϕ and removes existential quantifiers to obtain:

$$\phi^{\text{UNIV}} \equiv \forall x_1.\forall x_4. \\ ((x_4 - 1 < \mathbf{c_{3,1}} + \mathbf{c_{3,2}} \cdot \mathbf{x_1} \vee \mathbf{c_{2,1}} + \mathbf{c_{2,2}} \cdot \mathbf{x_1} \leq x_4) \wedge \\ (x_4 > \mathbf{c_{2,1}} + \mathbf{c_{2,2}} \cdot \mathbf{x_1} \vee \mathbf{c_{3,1}} + \mathbf{c_{3,2}} \cdot \mathbf{x_1} \geq x_1))$$

Step 2: Universal Quantifier Elimination

In Step 2, the algorithm eliminates universal quantifiers from the formula. Since the predicate p^{UNIV} is a boolean combination of polynomial (in)equalities, the purely universally quantified formula ϕ^{UNIV} constructed in Step 1 can be viewed as a *polynomial entailment*. This is because it defines a boolean combination of polynomial (in)equalities that need to be satisfied for all universally quantified variable valuations. Hence, in order to remove universal quantifiers, we will use some of the classical results from real algebraic geometry (Bochnak, Coste, and Roy 2013) that tackle the problem of *satisfiability of polynomial entailment*.

Satisfiability of Polynomial Entailment. Before proceeding further with our algorithm, we first define the problem of satisfiability of polynomial entailment and informally introduce the theorems used in our algorithm.

Consider a set $V = \{x_1, \ldots, x_r\}$ of real-valued variables, a set $\Phi = \{p_0 \bowtie 0, \ldots, p_m \bowtie 0\}$ of polynomial inequalities over V, and a polynomial inequality $\psi = (p \bowtie 0)$ over V, where each $\bowtie \in \{\geq, >\}$. The *satisfiability of polynomial entailment* problem focuses on finding sufficient and necessary conditions for the universally quantified formula

$$\forall x_1 \ldots \forall x_r.\ (\Phi \implies \psi)$$

to be satisfiable. Real algebraic geometry provides us with mathematical tools to reduce this problem to solving a system of polynomial inequalities, for which sub-exponential algorithms exist (Grigor'ev and Vorobjov Jr 1988). In particular, our algorithm will utilize the following results:

- **Farkas' Lemma.** This result provides a sound and complete method when both Φ and ψ are linear. Given a system Φ of linear inequalities over V, Farkas' Lemma provides a set of necessary and sufficient conditions for Φ to be satisfiable and for Φ to entail a linear inequality ψ over V. These conditions require that ψ can be written as non-negative linear combination of the inequalities in Φ and the trivial inequality $1 \geq 0$, giving rise to an equivalent system of purely existentially quantified inequalities over the template variables as well as a set of new variables introduced by the Farkas' Lemma transformation.
- **Positivstellensätze Theorems.** These provide sound and semi-complete methods when either Φ or ϕ is non-linear. In the case when Φ is a system of linear inequalities over V but the inequality ψ is non-linear, we use Handelman's Theorem. Otherwise, if both Φ and ψ contain non-linear inequalities, we use Putinar's Theorem. Both theorems provide a set of conditions for Φ to be satisfiable and to entail ψ, and their application gives rise to a system of purely existentially quantified inequalities over the template variables as well as a set of new variables introduced by the theorem transformation.

We present the formal statements and the details of these theorems in the extended version of the paper.

Universal Quantifier Elimination. Our algorithm uses the above theorems to eliminate universal quantifiers from the formula ϕ^{UNIV} constructed in Step 1. Since p^{UNIV} is in CNF, we can write $p^{\text{UNIV}} \equiv \psi_1 \wedge \cdots \wedge \psi_r$, with each $\psi_i \equiv \psi_{i,1} \vee \cdots \vee \psi_{i,w_i}$ and each $\psi_{i,1}$ a polynomial inequality.

The algorithm first converts each ψ_i to the following equivalent form (if $w_i = 1$, then convert to $1 > 0 \Rightarrow \psi_{i,1}$):

$$\neg\psi_{i,1} \wedge \cdots \wedge \neg\psi_{i,w_i-1} \Rightarrow \psi_{i,w_i}$$

Then, depending on polynomial degrees in each ψ_i, the algorithm either applies Farkas' lemma if all polynomial degrees are 0 and 1, or Positivstellensätze theorems if higher degree polynomials are involved. For each ψ_i, the procedure results in a system of polynomial inequalities whose satisfiability implies satisfiability of ψ_i. Combining these systems together gives rise to a system of polynomial inequalities ϕ^{FREE} whose satisfiability implies satisfiability of ϕ^{UNIV}.

Example (Continued)**.** *We first convert the formula ϕ^{UNIV} to the polynomial entailment form*

$$\phi^{\text{UNIV}} \equiv ((x_4 - 1 \geq c_{3,1} + c_{3,2}.x_1 \Rightarrow c_{2,1} + c_{2,2}.x_1 \leq x_4) \wedge (x_4 \leq c_{2,1} + c_{2,2}.x_1 \Rightarrow c_{3,1} + c_{3,2}.x_1 \geq x_1))$$

As all involved expressions are linear, the algorithm applies Farkas' lemma. Omiting the details of the translation, the algorithm obtains the following system of polynomial inequalities whose satisfiability implies satisfiability of ϕ^{UNIV}:

$$\begin{aligned}\phi^{\text{FREE}} \equiv &(-y_1 \cdot c_{3,2} = -c_{2,2}) \wedge \\ &(y_1 \cdot (-1 - c_{3,1}) + y_2 = -c_{2,1}) \wedge \\ &(y_3 \cdot c_{2,2} = c_{3,2} - 1) \wedge (-y_3 = 0) \wedge \\ &(y_3 \cdot (c_{2,1}) + y_4 = c_{3,1})\end{aligned}$$

The $c_{i,j}$'s are Skolem function template variables, whereas $y_1, \ldots, y_4$ are fresh variables introduced by Farkas' lemma.

Remark (Distinction between LRA and NRA)**.** *Step 2 is the only part of our algorithm where the distinction between LRA and NRA arises. If the initial quantified formula ϕ is not expressible in LRA, then it must contain non-linear inequalities and so we cannot apply Farkas' lemma. Therefore, in the NRA case, we can only apply Positivstellensätze theorems.*

Step 3: Quantifier-free Satisfiability Checking

In Step 3, we use an off-the-shelf SMT solver to check the satisfiability of the system of polynomial inequalities (i.e. a quantifier-free formula) ϕ^{FREE} constructed in Step 2. If the SMT solver finds a solution, then we conclude that ϕ^{FREE} is satisfiable. By Step 2, this implies that ϕ^{UNIV} is satisfiable. Finally, by Step 1, this implies that the original quantified formula ϕ is satisfiable. Otherwise, our algorithm cannot prove satisfiability of the input formula. One can run the same procedure on $\neg\phi$ where its satisfiability is equivalent to ϕ being unsatisfiable.

Example (Continued)**.** *Applying an SMT solver to the formula ϕ^{FREE} obtained before, we compute the following valuations for Skolem function template variables $c_{2,1} = 0, c_{2,2} = 1, c_{3,1} = 1, c_{3,2} = 1$. Hence, $f_2 = x_1$ and $f_3 = 1 + x_1$ define valid Skolem functions for x_2 and x_3 which prove the satisfiability of the original formula ϕ. We provide a full example of the NRA settings in the extended version of the paper.*

The following theorem establishes soundness, semi-completeness and a complexity bound for our algorithm. The soundness result states that, if our algorithm computes Skolem functions for the existential variables, then they are guaranteed to be correct. The semi-completeness result states that, if there exist polynomial Skolem functions of degree at most D that witness formula satisfiability, then our algorithm is guaranteed to find them.

Theorem 2 (Soundness, Semi-Completeness, Complexity, proof in the extended version)**.** *The algorithm is sound and semi-complete, both for quantified formulas in LRA and NRA. It runs in subexponential time and polynomial space, parametrized by the template polynomial degree D.*

Witnesses for Existentially Quantified Variables. To conclude this section, we highlight that when the algorithm solves the system of polynomial constraints ϕ^{FREE} in Step 3, it computes concrete real values for template polynomial coefficients in Skolem functions constructed in Step 1. Hence, it computes *witnesses of satisfiability for existentially quantified variables*, which was one of the goals of our algorithm design that is important in many applications, as discussed in the Introduction.

Experimental Results

We implemented a prototype of our method for satisfiability checking in a tool called QuantiSAT[1], and compared it against two state-of-the-art SMT solvers that support satisfiability checking for quantified LRA and NRA formulas. The goal of our experiments is to answer the following two research questions: (1) How well does our method perform in comparison to the existing tools, in terms of the number of solved instances and runtime? (2) Is our method able to compute witnesses for existentially quantified variables?

Implementation Details. Our tool QuantiSAT is written in Python and it uses PolyHorn (Chatterjee et al. 2024) as a back-end tool for applying the Positivstellensätze theorems in Step 2 of the algorithm. It uses Z3 (de Moura and Bjørner 2008) and MathSAT5 (Cimatti et al. 2013) for solving the quantifier-free formulas derived in Step 3 of the algorithm. For the experiments, we used a Debian 12 machine with a 2.45GHz AMD EPYC 7763 CPU and 16 GB of RAM. In our experiments, we ran QuantiSAT with polynomial degrees for Skolem function templates equal to $D \in \{0, 1, 2\}$.

Benchmarks. We consider three benchmark suites of quantified formulas in LRA and NRA:

1. The Keymaera benchmark suite, taken from SMT-COMP (Bobot et al. 2023), contains 222 LRA formulas and 3813 non-linear formulas. However, many of the non-linear formulas are actually not expressible in NRA, as they contain the division operator which is not supported by NRA. Removing these results in 511 NRA formulas.
2. The Mjollnir (Monniaux 2010) benchmark suite consists of 3600 LRA formulas. This benchmark suite was used for the evaluation of the tool Mjollnir, which was later outperformed by Z3 (Bjørner and Janota 2015), hence we do not include the Mjollnir tool in our results.
3. The PolySynth (Goharshady et al. 2023) benchmark suite consists of 32 NRA formulas. PolySynth is a program synthesis tool, which reduces the program synthesis problem to computing a satisfying assignment for a quantified formula in NRA. In our evaluation, we collected 32 quantified formulas that arise in their program synthesis procedure and used them to evaluate the effectiveness of our method and the baselines on these NRA benchmarks.

[1]https://doi.org/10.5281/zenodo.13341655

Since our method requires a quantifier-free part of the input formula to be provided in CNF (recall the assumptions in the Algorithm section), we first converted each quantifier-free part into CNF and then provided the CNF formulas as input to our tool and the baselines. The conversion time to CNF is not considered in the runtimes presented in Table 1.

Experimental Setup and Baselines. We compare QuantiSAT against two state-of-the-art SMT solvers Z3 (de Moura and Bjørner 2008) and CVC5 (Barbosa et al. 2022). The timeout for each tool on each benchmark is set to 10 minutes. Since we are not only interested in satisfiability checking but also in computing witnesses for existentially quantified variables that prove their satisfiability, in order to instruct Z3 and CVC5 to compute these witnesses, we provide them with two different variants of our benchmarks:

- **Uninterpreted Skolemization.** In the first variant, we only require the baselines to compute *some witnesses* for existentially quantified variables, not necessarily being polynomial expressions. Hence, we replace each existentially quantified variable by an uninterpreted predicate over the preceding universally quantified variables. For Z3, we denote the resulting baseline by `Z3-uSk`. Since CVC5 does not support uninterpreted predicates, we could not evaluate it on this variant. Our goal here is to evaluate the effectiveness of our template-based Skolemization as opposed to general Skolem functions.
- **Template Skolemization.** In the second variant, we ask our baselines to compute witnesses for existentially quantified variables *in terms of polynomials* over universally quantified variables. Hence, rather than only using uninterpreted predicates, in this variant we use the same polynomial Skolem function templates as in our QuantiSAT. We consider polynomial degrees $D \in \{0, 1, 2\}$, and count each instance as solved by the baseline if it can be solved for at least one of these three polynomial degrees. For Z3, we denote the resulting baseline by `Z3-tSk`. For CVC5, we denote the resulting baseline by `CVC5-tSk`. Our goal here is to evaluate the effectiveness of our quantifier elimination method based on Positivsellensätze theorems, as opposed to other quantifier elimination procedures implemented in these SMT solvers.

Results on LRA Benchmarks. The first two rows of Table 1 summarize our experimental results on LRA benchmarks. It can be seen that average runtimes of all tools are comparable and quite small (differing by only a few seconds). The most important highlight of the table is the number of instances and the unique instances solved by each tool, whereas runtimes of our and competing tools are the secondary aspect. We summarize our results on the LRA benchmarks below:

- **Successful Instances.** (i) Our tool successfully solves all LRA benchmarks from the Keymaera benchmark suite, while `Z3-uSk` and `CVC5-tSk` fail on several cases. The fact that QuantiSAT and `Z3-tSk` solve all instances shows that the template-based approach to Skolemization provides an efficient and highly promising approach to quantifier elimination in LRA. (ii) On the Mjollnir benchmarks, QuantiSAT outperforms all the baselines with a gap of at least 284 instances, while solving 157

	QuantiSAT			Z3-uSk			Z3-tSk			CVC5-tSk		
	Solved	Avg. T.	U.	Solved	Avg. T.	U.	Solved	Avg. T.	U.	Solved	Avg. T.	U.
Keymaera-LRA	**222**	0.04	0	202	0.01	0	**222**	0.01	0	210	0.01	0
Mjollnir	**877**	18.48	157	476	14.99	76	593	23.34	2	397	38.25	15
Keymaera-NRA	**503**	0.06	22	481	0.01	0	483	0.03	2	481	0.01	0
PolySynth	**30**	4.00	4	20	3.44	0	28	11.16	1	6	13.74	0

Table 1: Summary of the experimental results. Each row shows the performance of different tools on the set of benchmarks specified in the first column. For each solver, the Solved column shows the number of instances solved with the best results shown in bold, the Avg. T. column shows the average runtime (in seconds), and the U. column presents the number of unique instances solved by the tool (i.e. instances that were solved only by that tool and no other tool). The timeout for each tool on each benchmark is set to 10 minutes.

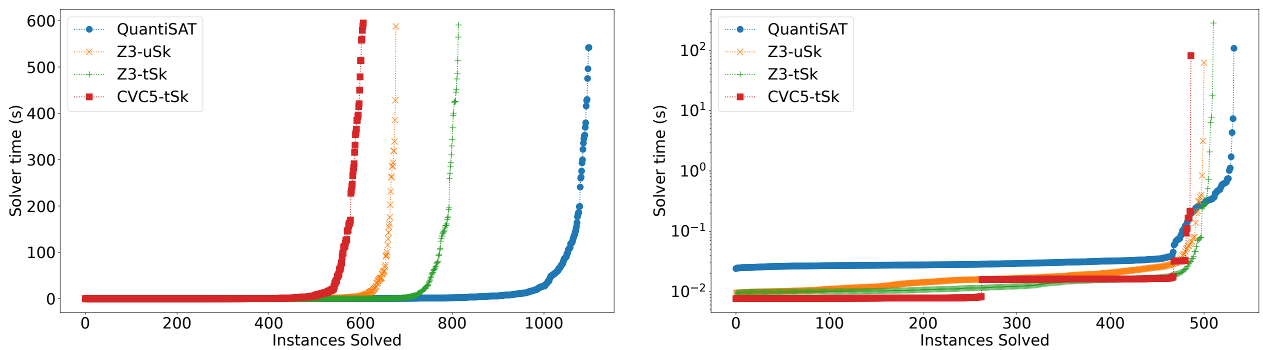

Figure 1: Runtime distribution over the LRA (left) and NRA (right) benchmarks solved by each tool. The figure to the right is in logarithmic scale for better presentation. Generally, faster tools have lower curves and the more instances a tool solves, its curve will be more to the right. Since solving more instances is primary, tools with graphs to the right are preferable.

unique instances. On the other hand, Z3-uSk solves 76 unique instances that require complicated, e.g. heavily piecewise, interpretations for the Skolem functions.

- **Runtimes.** (i) The average runtime of all the considered tools is very small and negligible on the Keymaera benchmarks. (ii) On the Mjollnir benchmarks, the average runtime of our tool is lower than Z3-tSk and CVC5-tSk which means that applying Positivstellensätze is a crucial step for efficient quantifier elimination. Compared with Z3-uSk, although the average runtime of Z3-uSk is lower, our tool can solve in 2.45s the same number of instances that Z3-uSk solves in 600s.

Figure 1(left) shows the runtime distribution of each of the tool's runtimes over all the LRA benchmarks that were solved by that tool. QuantiSAT has the lowest and rightmost curve, which shows that it can solve more instances in less time compared to the baselines. These results show practical efficiency and applicability of our method to the satisfiability checking problem for quantified formulas in LRA.

Results on NRA Benchmarks. The last two rows of Table 1 summarize our results on NRA benchmarks. Similar to the LRA results, the average runtimes of the tools are small and comparable. The most interesting distinction comes in the number of instances and unique instances solved by tools:

- **Successful Instances.** (i) On the NRA benchmarks from the Keymaera benchmark suite, our tool outperforms the baselines by solving 98% of the instances, which includes 22 instances uniquely solved by our tool. We believe that the high success rate is due to the strong semi-completeness guarantees provided by our method. (ii) On the benchmarks from the PolySynth benchmark suite, QuantiSAT outperforms all the baselines by solving 30 instances, including 4 unique ones.
- **Runtimes.** (i) On the Keymaera NRA benchmarks, the average runtime of QuantiSAT is higher than the baselines, however this is due to the longer runtimes required by the benchmarks solved only by our tool. QuantiSAT solves as many benchmarks as all the baselines in only 0.32 seconds. (ii) Comparing runtimes on the PolySynth benchmarks, Z3-uSk and CVC5-tSk have smaller average runtimes, however this is again due to the instances solved only by our tool. The only comparable baseline in terms of solved instances is Z3-tSk, whose average runtime is more than twice the runtime of QuantiSAT.

Figure 1(right) shows the runtime distribution of the NRA instances solved by each tool.

Summary of Results. Based on the above discussions we conclude that, both on the LRA and the NRA benchmarks, QuantiSAT outperforms state-of-the-art and well-maintained tools such as Z3 and CVC5 on the number of solved instances when required to compute witnesses for existentially quantified variables. Furthermore, QuantiSAT is able to solve a significant number of new instances that other tools could not handle. Finally, this is achieved at improved average runtimes, as discussed above. All of this leads us to the conclusion that our method provides a significant step forward in satisfiability checking for quantified formulas in LRA and NRA, as well as in computing the witnesses. This is particularly due to the new quantifier elimination procedure that we propose. Given that the tool support for quantifier elimination (especially in NRA) is limited, we believe that our method provides important new ideas and breakthroughs in satisfiability checking for quantified formulas.

Conclusion

We presented a novel method for satisfiability checking for quantified formulas in LRA and NRA. Our method is based on a novel and efficient quantifier elimination procedure. The method is sound, semi-complete, and runs in parameterized subexponential time and polynomial space. In contrast, previous sound and complete procedures have doubly-exponential time and at least exponential space complexity. Our method is also able to compute witnesses for existentially quantified variables, which is important for many applications. We implemented our method in a prototype tool called QuantiSAT. Our QuantiSAT outperforms two state-of-the-art SMT solvers on the number of solved instances and is able to prove a significant number of new instances that other tools could not handle. Hence, we believe that our method provides a significant step forward in satisfiability checking for quantified formulas in LRA and NRA as well as in efficient algorithms for quantifier elimination.

Our work opens several interesting future work directions. First, it would be interesting to consider further improvements to the template-based Skolemization method proposed in this work. These could include more general templates, such as *piecewise* linear and polynomial expressions. Second, it would be interesting to study the applicability of the template-based Skolemization technique to quantifier elimination in other theories, beyond LRA and NRA.

Acknowledgements

This work was partially funded by ERC CoG 863818 (ForM-SMArt) and Austrian Science Fund (FWF) 10.55776/COE12.

References

Alur, R.; Henzinger, T. A.; and Kupferman, O. 2002. Alternating-time temporal logic. *J. ACM*, 49(5): 672–713.

Asadi, A.; Chatterjee, K.; Fu, H.; Goharshady, A. K.; and Mahdavi, M. 2021. Polynomial reachability witnesses via Stellensätze. In *PLDI*, 772–787.

Barbosa, H.; Barrett, C. W.; Brain, M.; Kremer, G.; Lachnitt, H.; Mann, M.; Mohamed, A.; Mohamed, M.; Niemetz, A.; Nötzli, A.; Ozdemir, A.; Preiner, M.; Reynolds, A.; Sheng, Y.; Tinelli, C.; and Zohar, Y. 2022. cvc5: A Versatile and Industrial-Strength SMT Solver. In *TACAS*, 415–442.

Basu, S.; Pollack, R.; and Roy, M. 2006. *Algorithms in Real Algebraic Geometry*. Springer.

Benedetti, M. 2004. Evaluating QBFs via Symbolic Skolemization. In *LPAR*, volume 3452 of *Lecture Notes in Computer Science*, 285–300. Springer.

Bjørner, N. S.; and Janota, M. 2015. Playing with Quantified Satisfaction. In *LPAR*, 15–27.

Blekherman, G.; Parrilo, P. A.; and Thomas, R. R. 2012. *Semidefinite optimization and convex algebraic geometry*. SIAM.

Bobot, F.; List, C.; Bromberger, M.; and Hoenicke, J. 2023. 18th International Satisfiability Modulo Theories Competition (SMT-COMP 2023): Rules and Procedures.

Bochnak, J.; Coste, M.; and Roy, M.-F. 2013. *Real algebraic geometry*, volume 36. Springer Science & Business Media.

Brown, C. W.; and Davenport, J. H. 2007. The complexity of quantifier elimination and cylindrical algebraic decomposition. In *Proceedings of the 2007 international symposium on Symbolic and algebraic computation*, 54–60.

Chatterjee, K.; Goharshady, A. K.; Goharshady, E. K.; Karrabi, M.; Saadat, M.; and Zikelic, D. 2024. PolyHorn: A Polynomial Horn Clause Solver. *arXiv preprint arXiv:2408.03796*.

Chatterjee, K.; Henzinger, T. A.; and Piterman, N. 2010. Strategy logic. *Inf. Comput.*, 208(6): 677–693.

Cimatti, A.; Griggio, A.; Schaafsma, B. J.; and Sebastiani, R. 2013. The mathsat5 smt solver. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 93–107. Springer.

Collins, G. E. 1975. Quantifier elimination for real closed fields by cylindrical algebraic decompostion. In Brakhage, H., ed., *Automata Theory and Formal Languages*. Springer.

Corzilius, F.; Loup, U.; Junges, S.; and Ábrahám, E. 2012. SMT-RAT: An SMT-Compliant Nonlinear Real Arithmetic Toolbox: (Tool Presentation). In *International Conference on Theory and Applications of Satisfiability Testing*, 442–448. Springer.

Dantzig, G. B.; Eaves, B. C.; et al. 1972. Fourier-Motzkin elimination and its dual.

De Moura, L.; and Jovanović, D. 2013. A model-constructing satisfiability calculus. In *International Workshop on Verification, Model Checking, and Abstract Interpretation*, 1–12. Springer.

de Moura, L. M.; and Bjørner, N. S. 2007. Efficient E-Matching for SMT Solvers. In Pfenning, F., ed., *Automated Deduction - CADE-21, 21st International Conference on Automated Deduction, Bremen, Germany, July 17-20, 2007, Proceedings*, volume 4603 of *Lecture Notes in Computer Science*, 183–198. Springer.

de Moura, L. M.; and Bjørner, N. S. 2008. Z3: An Efficient SMT Solver. In *TACAS*, 337–340.

Farkas, J. 1902. Theory of simple inequalities. *Journal for pure and applied mathematics (Crelles Journal)*, 1902(124): 1–27.

Farzan, A.; and Kincaid, Z. 2016. Linear Arithmetic Satisfiability via Strategy Improvement. In *IJCAI*, 735–743. IJCAI/AAAI Press.

Ge, Y.; Barrett, C. W.; and Tinelli, C. 2009. Solving quantified verification conditions using satisfiability modulo theories. *Ann. Math. Artif. Intell.*, 55(1-2): 101–122.

Goharshady, A. K.; Hitarth, S.; Mohammadi, F.; and Motwani, H. J. 2023. Algebro-geometric Algorithms for Template-Based Synthesis of Polynomial Programs. In *OOPSLA*, 727–756.

Grigor'ev, D. Y.; and Vorobjov Jr, N. N. 1988. Solving systems of polynomial inequalities in subexponential time. *Journal of symbolic computation*, 5(1-2): 37–64.

Gulwani, S.; Srivastava, S.; and Venkatesan, R. 2008. Program analysis as constraint solving. In *PLDI*, 281–292. ACM.

Handelman, D. 1988. Representing polynomials by positive linear functions on compact convex polyhedra. *Pacific Journal of Mathematics*, 132(1): 35–62.

Jovanović, D.; and De Moura, L. 2013. Solving non-linear arithmetic. *ACM Communications in Computer Algebra*, 46(3/4): 104–105.

Kautz, H. A.; and Selman, B. 1992. Planning as Satisfiability. In *ECAI*, 359–363. John Wiley and Sons.

Krivine, J.-L. 1964. Preordered Rings. *Mathematical Analysis Journal*, 12: p–307.

Kroening, D.; and Strichman, O. 2016. *Decision Procedures - An Algorithmic Point of View, Second Edition*. Texts in Theoretical Computer Science. An EATCS Series. Springer.

Monniaux, D. 2010. Quantifier Elimination by Lazy Model Enumeration. In *CAV*, volume 6174 of *Lecture Notes in Computer Science*, 585–599. Springer.

Murphy, C.; and Kincaid, Z. 2024. Quantified Linear Arithmetic Satisfiability via Fine-Grained Strategy Improvement. In *CAV (1)*, volume 14681 of *Lecture Notes in Computer Science*, 89–109. Springer.

Passmore, G. O.; and Jackson, P. B. 2009. Combined decision techniques for the existential theory of the reals. In *International Conference on Intelligent Computer Mathematics*, 122–137. Springer.

Putinar, M. 1993. Positive polynomials on compact semialgebraic sets. *Indiana University Mathematics Journal*, 42(3): 969–984.

Reynolds, A.; Tinelli, C.; Goel, A.; Krstic, S.; Deters, M.; and Barrett, C. W. 2013. Quantifier Instantiation Techniques for Finite Model Finding in SMT. In *CADE*, volume 7898 of *Lecture Notes in Computer Science*, 377–391. Springer.

Scowcroft, P. 1988. A note on definable Skolem functions. *The Journal of symbolic logic*, 53(3): 905–911.

Solar-Lezama, A.; Tancau, L.; Bodík, R.; Seshia, S. A.; and Saraswat, V. A. 2006. Combinatorial sketching for finite programs. In *ASPLOS*, 404–415. ACM.

Stengle, G. 1974. A nullstellensatz and a positivstellensatz in semialgebraic geometry. *Mathematische Annalen*, 207: 87–97.

Weispfenning, V. 1988. The complexity of linear problems in fields. *Journal of symbolic computation*, 5(1-2): 3–27.