Thesis Proposal

Formalizing Algorithms for Real Quantifier Elimination

Katherine Cordwell

April 8, 2022

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
André Platzer, Chair
Jeremy Avigad
Frank Pfenning
Dexter Kozen, Cornell University
Lawrence Paulson, University of Cambridge

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Copyright © 2022 Katherine Cordwell

Abstract

This thesis will focus on the dearth of efficient formally verified support for algorithms for real quantifier elimination. It proposes a two-pronged approach: first, verify a general-purpose multivariate QE algorithm based on the Ben-Or, Kozen, and Reif decision procedure and its variant by Renegar; second, verify a suite of highly efficient but limited QE methods, particularly *virtual substitution*, on top of the general-purpose algorithm.

Contents

Funding Acknowledgment. This material is based upon work supported by the National Science Foundation under Grant No. CNS-1739629, a National Science Foundation Graduate Research Fellowship under Grants Nos. DGE1252522 and DGE1745016, and by the AFOSR under grant number FA9550-16-1-0288. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or of AFOSR.

1 Introduction

Quantified formulas in the first-order logic of real arithmetic ($\text{FOL}_{\mathbb{R}}$) arise in various application domains, including the formal verification of cyber-physical systems, geometric theorem proving, and stability analysis of models of biological systems. The best way of analyzing these formulas is to reduce them to logically equivalent quantifier-free formulas, through a process known as *quantifier elimination (QE)*. Alfred Tarski proved that algorithms for real quantifier elimination exist [39]; thus, *in theory*, all it takes to rigorously answer any real arithmetic question is to verify a QE procedure for $\text{FOL}_{\mathbb{R}}$. However, *in practice*, these QE algorithms are complicated and the fastest known QE algorithm, *cylindrical algebraic decomposition (CAD)* [3, 7, 8, 21] is, in the worst case, doubly exponential in the number of variables.

Given the safety-critical nature of these applications, having correct algorithms for quantifier elimination is crucial. However, the intricate nature of these algorithms makes them difficult to verify. Although there are a range of verified univariate methods [19, 24, 25], there are only two general-purpose verified multivariate QE algorithms [6, 22]. Unfortunately, both are based on algorithms that have non-elementary complexity (that is, their complexity is not bounded by any tower of powers of two), which makes them impractical.

This impracticality has significant consequences. To more fully understand this, let us briefly take a detour into the world of cyber-physical systems (CPS) [30], a disparate class of systems which includes planes, surgical robots, and power grids. CPS are becoming increasingly ingrained in our daily lives. Unfortunately, experimental testing is not enough to ensure CPS safety, as the continuous dynamics of CPS introduce uncountable behavior. Other approaches are needed, like that of *deductive verification*—first model CPS in logic, and then prove safety guarantees about the model using rigorous logical proof rules, especially with the aid of a theorem prover. These proofs very often reduce to complicated quantified statements [32, 36] which intuitively capture questions such as: "For all possible positions that my self-driving car is in, does there exist a safe control choice for the car?"

To analyze these questions, QE is needed: and here the dearth of formally verified QE support forces CPS proofs to outsource QE to unverified systems. For example, the theorem prover KeYmaera X [16] outsources real arithmetic to Mathematica and Z3. Using these tools as trusted oracles somewhat undermines trust [15] in the conclusions of formal methods proofs in the high-stakes world of CPS. Better support is needed.

This thesis proposes a twofold approach to tackle this problem. First, we propose verifying a general-purpose multivariate QE algorithm based on the *Ben-Or, Kozen, and Reif (BKR)* algorithm and its variant by Renegar [2, 33]. Though BKR may not be as efficient as CAD in the average case, it has good potential for parallelism; further, CAD and BKR have different efficiency tradeoffs and are complementary. Perhaps most importantly, BKR seems more amenable to formalization than CAD, and thus *appears to fit in a "sweet spot" within the tradeoff between practicality and ease of formalization.*

Second, we propose verifying a suite of preprocessing and special-purpose QE methods, particularly *virtual substitution (VS)*, on top of the general-purpose algorithm. This is because in order to achieve efficiency, a strong general-purpose algorithm is not enough: unverified tools like Mathematica make extensive use of strong special-purpose methods, preprocessing, and heuristics to achieve efficient QE. Among special-purpose QE methods,

VS [42,43] is particularly well-known; it targets problems that contain low-degree polynomial inequalities or equations. Linear and quadratic VS are of practical significance; they serve to improve QE [28] and SMT tools and are the basis of the experimentally successful [38] Redlog solver [14].

Our theorem prover of choice for this work is Isabelle/HOL [27]. Isabelle/HOL is well-suited for formalizing mathematics and has a large collection of existing proof developments available in the Archive of Formal Proofs, which are of great benefit to us; further, the built-in search tool and Sledgehammer [29] provide invaluable automation for discovering existing theorems and for finishing (easy) subgoals in proofs.

In sum, this thesis aims to advance the state of formally verified support for QE by proposing practical steps towards verifying (in Isabelle/HOL) a promising general-purpose QE algorithm in conjunction with strong special-purpose methods.

2 Proposed Approach: VS and Preprocessing

Our progress on formally verifying special-purpose QE methods in Isabelle/HOL has already been quite encouraging: We have already formalized linear and quadratic virtual substitution [34]; to our knowledge we are the first to verify the quadratic *inequality* cases, and our work takes an experimental approach which was not present in previous formalization efforts. The rest of this section is heavily based on [34]. For the purposes of this proposal, we elide intricate technical details and focus more on high-level intuition, motivation, and context.

Remark 1. *A note on code export. Within Isabelle/HOL, our algorithms are provably correct up to Isabelle/HOL's trusted core. To demonstrate the practical usefulness of our verified algorithms, we export our code from Isabelle/HOL to SML for experimentation. The code export removes overhead, so that the exported algorithms run more quickly than internal algorithms. The exported algorithms additionally rely on the correctness of Isabelle/HOL's code export, which ongoing work is attempting to establish [17].*

2.1 What is Virtual Substitution?

Informally (and broadly) speaking, VS discretizes the QE problem by solving for the roots of one or more low-degree (particularly linear or quadratic) polynomials $f_1(x), \ldots, f_n(x)$. VS focuses on these roots and the intervals around them to identify and substitute appropriate representative "sample points" for x into the rest of the formula. However, these sample points may contain fractions, square roots, and/or other extensions of the logical language, and so they must be substituted "virtually": That is, VS creates a formula *in* $FOL_{\mathbb{R}}$ *proper* that models the behavior of the direct substitution, which would be outside of $\mathrm{FOL}_{\mathbb{R}}$. VS applies in two cases: an equality case and a general case.

Previous formalizations of VS in Isabelle/HOL had considered both of these cases for linear VS (Nipkow, [26]) and the equality case for quadratic VS (Chaieb, [4]). Nipkow [26] formalized a VS procedure for *linear* equations and inequalities. The building blocks of $\mathrm{FOL}_{\mathbb{R}}$ formulas, or "atoms," in his work allow only for linear polynomials, which ensures that linear QE can always be performed, and simplifies the substitution procedure and associated proofs. In addition, Nipkow provided a generic framework that can be applied to several

different kinds of atoms (each new atom requires implementing several new theorems in order to create an exportable algorithm). While this is an excellent theoretical framework—we utilize several similar constructs in our formulation—we create an independent formalization that is specific to general $\text{FOL}_{\mathbb{R}}$ formulas, as our main focus is to provide an efficient algorithm in this domain. Specializing to one type of atom allows us to implement several optimizations which would be unwieldy to develop in a generic setting.

Chaieb [4] extends Nipkow's work to quadratic *equalities*. His formalizations are not publicly available, but he generously provided us with the code. While this was helpful for reference, our formalization is independent of his: we chose to build on a newer Isabelle/HOL polynomial library, and we focus on verified VS as an exportable standalone procedure (whereas Chaieb intrinsically linked VS with an auxiliary QE procedure).

Our formalization of VS handles all of the most practically significant cases: linear equality, linear inequality, quadratic equality, and *quadratic inequality* (which was not handled by Chaieb or Nipkow). This required significant low-level improvements and extensions to Isabelle's multivariate polynomials library. Our work is approximately 23,000 lines in Isabelle/HOL and is available on the Archive of Formal Proofs (AFP) [35]. Our framework is specialized to $\text{FOL}_{\mathbb{R}}$ formulas and could serve as a basis for developing practical general-purpose QE algorithms. We include select optimizations for VS, but largely defer these to future work; significantly, our framework is modularized and easily expandable to facilitate integrating future optimizations.

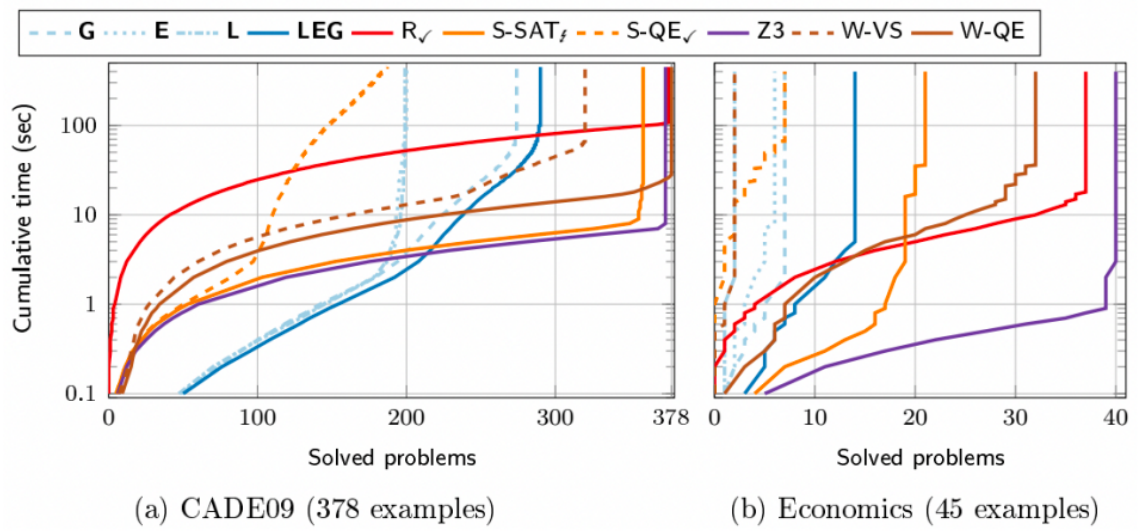
2.2 Experimental Results

To better motivate our overall goal of advancing formally verified support for QE, we discuss our experimental results for the VS formalization, which are quite promising. We develop (and export to SML for experimentation) several top-level algorithms that use various combinations of VS procedures and optimizations. We test these on, in total, 423 benchmarks from the literature [23, 31], comparing to Redlog, SMT-RAT [12], Z3 [13], and Wolfram Engine. We do not expect to outperform prior tools at this stage, as many of them have been optimized over a period of many years.

Figure 1 summarizes the performance on our two suites of benchmarks in terms of the cumulative time needed to solve (return "true", "false", "sat", or "unsat") the fastest n problems with a logarithmic time axis: more problems solved and a flatter curve is better. Our exported algorithms are depicted in blue. With 304 examples, we solve more examples than SMT-RAT in quantifier elimination mode (solves 191) and come close to virtual substitution in Wolfram Engine (solves 322). The remaining tools solve almost all examples; this is to be expected given that those tools have been optimized and fine-tuned (some for decades) and use efficient general-purpose fallback QE algorithms when VS does not succeed.

For sanity checking, we also *negated* the examples fom the CADE09 benchmark suite [31] and ran all the tools on these negated examples. In Fig. 2, we compare the results of running the original examples and their negations to highlight *contradictions* between answers. A contradiction occurs when a tool clams that both A and $\neg A$ are true or both A and $\neg A$ are false and is indicated by a red line.

As shown by the figure, Wolfram Engine and Z3 answer consistently on both formula sets, and solve (almost) all of the examples. Our algorithm (denoted LEG) answers consistently



(a) CADE09 (378 examples) (b) Economics (45 examples)

Figure 1: Cumulative time to solve fastest n problems (flatter and more is better)

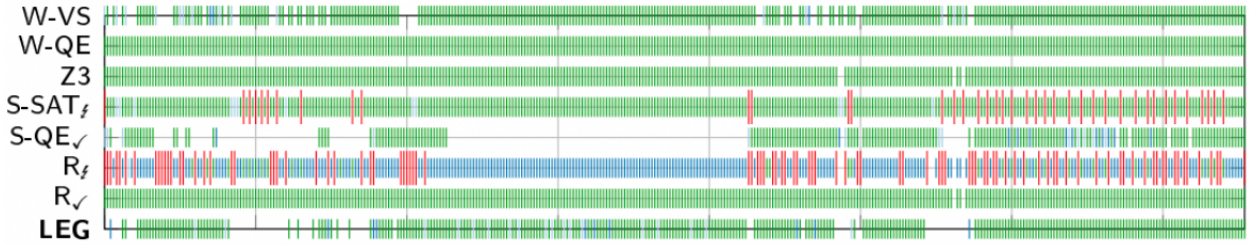


Figure 2: CADE09 consistency comparison between original and negated formula: color indicates discrepancies within tools (green (■): answer on original and negated formula agree, dark-blue (■): only original solved, light-blue (■): only negated solved, red+long (■): **contradictory** answers (both formulas unsat/proved or both sat/disproved), empty: both timeout/unknown)

(but solves fewer examples). Redlog, the main VS implementation, in R⨍ (snapshot 2021-04-13) did not perform well on the negated formulas and reports 96 contradictory answers. Testing on previous versions of Redlog revealed that these contradictions date back until at least mid-2019. The contradictory examples were shared with the developers and triggered several bug fixes that are now available in R✓ (snapshot 2021-07-16, no contradictions found on the benchmark set). SMT-RAT performs better than R⨍ on the negated formulas, but in satisfiability mode contradicts itself on 41 examples by silently ignoring quantifiers in the input; in quantifier elimination mode, SMT-RAT supports quantifiers and does not report contradictions, though this incurs a significant performance loss (S-SAT⨍ reports 359 answers while S-QE✓ only solves 187). No contradictions were found across tools, i.e., whenever a tool's answers were consistent internally, the answers agreed with those of other tools.

In summary, the performance of our verified virtual substitution QE on the benchmark set is encouraging. The number of solved examples is close to other VS implementations (304 examples by our VSLEG vs. 322 by W-VS) and a more detailed analysis of cumulative solving

time reveals that the majority of examples are solved fast. As we found 137 inconsistencies in other solvers, it is significant that our implementation comes with associated correctness proofs (assuming the orthogonal challenge of correct code generation [17]).

2.3 Proposed Work

Though the completed VS work could be standalone for the portions of this thesis that are concerned with verifying efficient limited-purpose QE methods, it is of practical interest to continue to develop and build upon the VS framework. Certainly our experiments both reveal the benefits of our current optimizations and indicate room for future improvements: some of these could be specific to VS, while others could be more general preprocessing methods. I have previously considered general QE preprocessing methods in the theorem prover PVS (see the associated technical memorandum: [9]) and found that they introduced the potential for significant speedup with minimal overhead, demonstrating the potential utility of such "auxiliary optimizations".

However, this "proposed work" section is intentionally left open-ended—choosing a set of optimizations to target is still pending work, and I expect this to be somewhat experimentally guided. In any case, the bulk and main focus of my proposed work is on verifying a general multipurpose QE algorithm in Isabelle/HOL, which we now turn to.

3 Proposed Approach: BKR

Recall that linear and quadratic virtual substitution can only succeed or make progress on a limited fragment of $\text{FOL}_{\mathbb{R}}$, because they require the presence of low-degree polynomials[1]. To handle QE queries outside of this fragment, a general-purpose QE algorithm is required. However, considering that the current verified general-purpose QE algorithms both have non-elementary complexity, and observing the difficulty in verifying the fastest-known QE algorithm, CAD, there seems to be a tension, or a tradeoff, between algorithmic efficiency and ease of verification.

We are interested in focusing on an algorithm that fits within a potential "sweet spot" within this tradeoff: the Ben-Or, Kozen, and Reif (BKR) algorithm [2]. Originally a decision procedure, BKR was extended to a full QE algorithm by Renegar [33], who also fixed an error in the complexity analysis. BKR is similar in flavor to Tarski's original QE algorithm, but with a built-in reduction step to lower asymptotic complexity.

Formalizing BKR presents a number of challenges: First, it requires not only a deep understanding of the BKR algorithm itself, but also the ability to "course-correct": that is, to fill in gaps in the original algorithm and to modify it in places where it is not amenable to formalization. So far, this has required making use of supplementary resources, including Renegar, a foundational algebraic geometry textbook [1], and Cyril Cohen's thesis [5] (which is relevant since Tarski's algorithm shares some theoretical underpinning with BKR). Secondly, the formalization requires intricate manipulations of mathematical objects, including

[1]Or of polynomials that have low-degree in essence; for example, $x^4 + x^3$ is nominally a polynomial of degree 4, but it carries the same sign information as $x^2 + x$, and so can be replaced by this lower-degree polynomial for the purposes of VS so that, e.g. $\exists x.\ x^4 + x^3 < 0$ transforms to $\exists x.\ x^2 + x < 0$.

multivariate polynomials and matrices, in Isabelle/HOL. This is challenging because the highly logical foundations of a theorem prover significantly complicate a number of small steps that would appear very simplistic on paper.[2] Isabelle/HOL has a number of highly useful proof developments in the Archive of Formal Proofs [18, 37, 40, 41] from which we greatly benefit, but significant low-level expansion has already been and is still needed.

We turn to a discussion of the BKR algorithm in two stages: we first detail univariate BKR, and then discuss the intuition underlying multivariate BKR and the challenges formalizing it poses. So far, we have succeeded in formalizing univariate BKR (and univariate Renegar), and our discussion of this stage is based on [10]. Our discussion of multivariate BKR—and its challenges—is based on ongoing work.

3.1 Univariate BKR

Throughout this section, we are working with *univariate* polynomials, which we assume to have variable x. Our decision procedure works for polynomials with rational coefficients (type `rat poly` in Isabelle), though some lemmas are proved more generally for univariate polynomials with real coefficients (type `real poly` in Isabelle).

At a high level, the univariate BKR algorithm works in 3 main pieces, which we will detail in the following subsections:

1. First, univariate QE is reduced to the problem of sign-determination.
2. Second, the general sign-determination problem is restricted to a more computational (discretized) sign-determination problem.
3. Third, the restricted sign-determination problem is solved by constructing a matrix equation that stores key sign information.

Let us tackle these pieces in order.

3.1.1 From Univariate Problems to Sign Determination

Formulas of *univariate real arithmetic* are generated by the following grammar, where p is a univariate polynomial with rational coefficients:

$$\phi, \psi \ ::= \ p > 0 \mid p \geq 0 \mid p = 0 \mid \phi \vee \psi \mid \phi \wedge \psi$$

In Isabelle/HOL, we define this grammar in `fml`, which is our type for formulas.

For formula ϕ, the *universal* decision problem is to decide if ϕ is true for *all* real values of x, i.e., validity of the quantified formula $\forall x\, \phi$. The *existential* decision problem is to decide if ϕ is true for *some* real value of x, i.e., validity of the quantified formula $\exists x\, \phi$. For example, a decision procedure should return false for formula (1) and true for formula (2) below (left).

[2] As an example of this, in VS we needed a way of isolating the coefficient of a variable within a polynomial in order to be able to rewrite a multivariate polynomial in $\mathbb{R}[a_1, \ldots, a_n, x]$ as a univariate polynomial in x with coefficients that are polynomials in $\mathbb{R}[a_1, \ldots, a_n]$. While this is extremely easy to do on paper, defining this capability in Isabelle/HOL required a large collection of technical, low-level simplification lemmas.

$\forall x\,(x^2-2=0 \wedge 3x>0)$	(1)	Formula Structure:	$\textcircled{A}=0 \wedge \textcircled{B}>0$
$\exists x\,(x^2-2=0 \wedge 3x>0)$	(2)	Polynomials:	$\textcircled{A}: x^2-2, \quad \textcircled{B}: 3x$

The first observation is that both univariate decision problems can be transformed to the problem of finding the set of *consistent sign assignments* (also known as realizable sign assignments [1, Definition 2.34]) of the set of polynomials appearing in the formula ϕ.

Definition 1. A **sign assignment** for a set G of polynomials is a mapping σ that assigns each $g \in G$ to either $+1$, -1, or 0. Given $x \in \mathbb{R}$, the **sign** of $g(x)$ is $+1$ if $g(x) > 0$, -1 if $g(x) < 0$, and 0 if $g(x) = 0$. A sign assignment σ for G is **consistent** if there exists an $x \in \mathbb{R}$ where, for all $g \in G$, the sign of $g(x)$ matches the sign of $\sigma(g)$.

For the polynomials $x^2 - 2$ and $3x$ appearing in formulas (1) and (2), the set of all consistent sign assignments (written as ordered pairs) is:

$$\{(+1,-1),\ (0,-1),\ (-1,-1),\ (-1,0),\ (-1,+1),\ (0,+1),\ (+1,+1)\}$$

Formula (1) is not valid because consistency of sign assignment $(0,-1)$ implies there exists a real value $x \in \mathbb{R}$ such that conjunct $x^2 - 2 = 0$ is satisfied but not $3x > 0$. Conversely, formula (2) is valid because the consistent sign assignment $(0,+1)$ demonstrates the existence of an $x \in \mathbb{R}$ satisfying $x^2 - 2 = 0$ and $3x > 0$. The truth-value of formula ϕ at a given sign assignment is computed by evaluating the formula after replacing all of its polynomials by their respective assigned signs. For example, for the sign assignment $(0,-1)$, replacing $\textcircled{A}$ by 0 and $\textcircled{B}$ by -1 in the formula structure underlying (1) and (2) shown above (right) yields $0 = 0 \wedge -1 > 0$, which evaluates to false. Validity of $\forall x\,\phi$ is decided by checking that ϕ evaluates to true at *each* of its consistent sign assignments. Similarly, validity of $\exists x\,\phi$ is decided by checking that ϕ evaluates to true at *at least one* consistent sign assignment. In both cases, the exact set of consistent sign assignments is needed to decide the QE problem correctly.

3.1.2 From Sign Determination to Restricted Sign Determination

The next step restricts the sign determination problem to the following more concrete format: Find all consistent sign assignments σ for a set of polynomials $q_1, \ldots, q_n$ at the roots of a nonzero polynomial p, i.e., the signs of $q_1(x), \ldots, q_n(x)$ that occur at the (finitely many) real values $x \in \mathbb{R}$ with $p(x) = 0$. This restriction to determining sign information *at the roots of an auxiliary polynomial* is important because it discretizes the QE problem (as polynomials have finitely many roots). Consider as input a set of polynomials (with rational coefficients) $Q = \{q_1, \ldots, q_n\}$ for which we need to find all consistent sign assignments. This can be transformed into a restricted sign determination problem as follows:

(1) Compute a polynomial p that satisfies the following properties[3]:

[3] An explicit choice of p satisfying these properties is $p = \left(\prod_{1\le i\le n} q_i\right) \cdot \frac{d}{dx}\left(\prod_{1\le i\le n} q_i\right) \cdot (x-B)(x+B)$, where $B > 0$ is the Cauchy root bound of $\left(\prod_{1\le i\le n} q_i\right)$ (so that B is greater than any root of the q_i's and $-B$ is smaller than any root of the q_i's).

i) Every root of the q_i's is also a root of p,

ii) p has a root in every interval between any two roots of the q_i's,

iii) p has a root that is greater than all of the roots of the q_i's, and

iv) p has a root that is smaller than all of the roots of the q_i's.

The relationship between the roots of p and the roots of $q_i \in Q$ is visualized in Fig. 3.

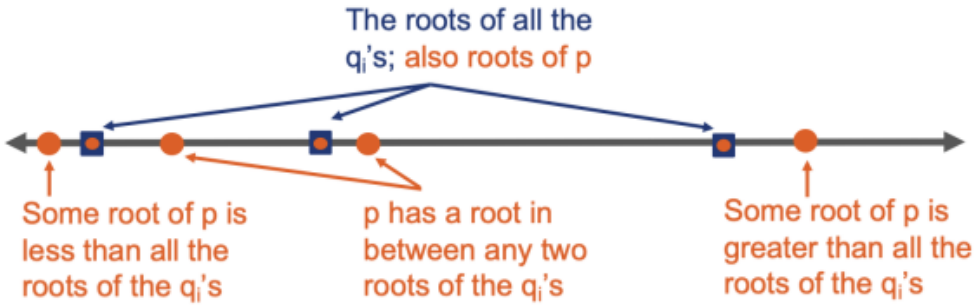


Figure 3: The relation between the roots of the added polynomial p and the roots of the q_i's.

(2) Solve the restricted sign determination problem for all consistent sign assignments of $\{q_1, \ldots, q_n\}$ at the roots of p.

Returning to Fig. 3, the q_i's are sign-invariant (i.e., do not change sign) in the intervals between any two roots of the q_i's (black squares) and to the left and right beyond all roots of the q_i's. Intuitively, this is true because moving along the blue real number line in Fig. 3, no q_i can change sign without first passing through a black square. Thus, all consistent sign assignments of q_i that only have nonzero signs must occur in one of these intervals and therefore, by sign-invariance, also at one of the roots of p (red points).

There are actually other ways of accomplishing this transformation into a restricted sign determination problem. What we have presented above (which we have also formalized) largely follows Renegar's style of discretization, but with the additional use of the Cauchy root bound from BKR to capture sign information at the limits.

Significantly, BKR and Renegar differ slightly in how they construct the restricted sign determination problem, and this small difference has significant consequences. BKR's restricted sign-determination problem insists that the q_i's be coprime with (i.e. share no common factors with) p, whereas Renegar's restricted sign-determination problem does not have this restriction. This restriction makes BKR's eventual decision procedure more complicated than the 2-step procedure presented above, but it considerably simplifies the subsequent construction of the matrix equation and its associated formal proofs, so it was very advantageous to formalize univariate BKR before formalizing univariate Renegar.

Now, let us turn to solving the restricted sign-determination problem with a matrix equation. Because the goal of this proposal is to develop a high-level understanding of the methods employed by the BKR algorithm, we present the simpler BKR-style construction of the matrix equation (which restricts the $q_i's$ to be coprime with p).

3.1.3 Solving Restricted Sign Determination with a Matrix Equation

The restricted sign determination problem for polynomials $q_1, \ldots, q_n$ at the roots of a polynomial $p \neq 0$, where each $q_1, \ldots, q_n$ is coprime with p, can be tackled naively by setting up and solving a *matrix equation*. The idea of using a matrix equation for sign determination dates back to Tarski [39] [1, Section 10.3], and accordingly our formalization shares some similarity to Cohen and Mahboubi's formalization [6] of Tarski's algorithm (see [5, Section 11.2]). BKR's additional insight is to avoid the prohibitive complexity of enumerating exponentially many possible sign assignments for $q_1, \ldots, q_n$ by computing the matrix equation recursively (with a divide and conquer algorithm) and performing a *reduction* that retains only the consistent sign assignments at each recursive step. This reduction keeps intermediate data sizes manageable because the number of consistent sign assignments is bounded by the number of roots of p throughout. We first explain the technical underpinnings of the matrix equation before returning to the recursive construction itself. *For brevity, references to sign assignments for $q_1, \ldots, q_n$ in this section are always at the roots of p.*

Matrix Equation: Technical Underpinnings. The inputs to the matrix equation are a set of candidate (i.e., not necessarily consistent) sign assignments $\tilde{\Sigma} = \{\tilde{\sigma}_1, \ldots, \tilde{\sigma}_m\}$ for the polynomials $q_1, \ldots, q_n$ and a set of subsets $S = \{I_1, \ldots, I_l\}$, $I_i \subseteq \{1, \ldots, n\}$ of indices selecting among those polynomials. The set of all consistent sign assignments Σ for $q_1, \ldots, q_n$ is assumed to be a subset of $\tilde{\Sigma}$, i.e., $\Sigma \subseteq \tilde{\Sigma}$.

For example, consider $p = x^3 - x$ and $q_1 = 3x^3 + 2$. The set of all possible candidate sign assignments $\tilde{\Sigma} = \{(+1), (-1)\}$ must contain the consistent sign assignments for q_1 (sign (0) is impossible as p, q_1 are coprime). The possible subsets of indices are $I_1 = \{\}$ and $I_2 = \{1\}$.

The main algebraic tool underlying the matrix equation is the *Tarski query* which provides semantic information about the number of roots of p with respect to polynomial q.

Definition 2. Given univariate polynomials p, q with $p \neq 0$, the *Tarski query* $N(p, q)$ is:

$$N(p, q) = \#\{x \in \mathbb{R} \mid p(x) = 0, q(x) > 0\} - \#\{x \in \mathbb{R} \mid p(x) = 0, q(x) < 0\}.$$

Importantly, the Tarski query $N(p, q)$ can be computed from input polynomials p, q using Euclidean remainder sequences *without* explicitly finding the roots of p. This is a consequence of the Sturm-Tarski theorem which has been formalized in Isabelle/HOL by Li [18]. The key standard theoretical result can be stated as follows:

Theorem 1. *[Generalized Sturm's theorem [33, Proposition 8.1]] Given coprime univariate polynomials p, q with $p \neq 0$, form the Euclidean remainder sequence $p_1 = p$, $p_2 = p'q$, and p_i is the negated remainder of p_{i-2} divided by p_{i-1} for $i \geq 3$. This terminates at some $p_{k+1} = 0$ because the remainder has lower degree than the divisor at every step. Let a_i be the leading coefficient of p_i for $1 \leq i \leq k$. Consider the two sequences $a_1, \ldots, a_k$ and $(-1)^{\deg p_1} a_1, \cdots, (-1)^{\deg p_k} a_k$. If $S^+(p, q)$ is the number of sign changes in $a_1, \ldots, a_k$ and $S^-(p, q)$ is the number of sign changes in $(-1)^{\deg p_1} a_1, \cdots, (-1)^{\deg p_k} a_k$, then $N(p, q) = S^-(p, q) - S^+(p, q)$.*

The theoretical complexity for computing $N(p, q)$ is $O(\deg p \, (\deg p + \deg q))$ [1, Sections 2.2.2 and 8.3]. However, this complexity analysis does not take into account the growth

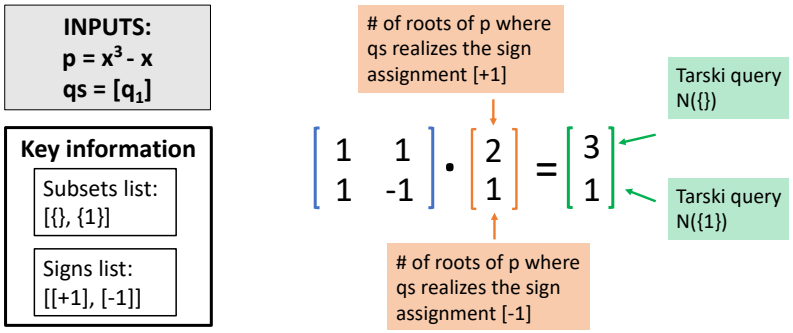


Figure 4: Matrix equation for $p = x^3 - x$, $q_1 = 3x^3 + 2$.

in bitsizes of coefficients in the remainder sequences [1, Section 8.3], so it will not be not achieved by the current Isabelle/HOL formalization of Tarski queries [18] without further optimization.

For the matrix equation, we lift Tarski queries to a *subset* of the input polynomials:

Definition 3. Given a univariate polynomial $p \neq 0$, univariate polynomials $q_1, \ldots, q_n$, and a subset $I \subseteq \{1, \ldots, n\}$, the *Tarski query* $N(I)$ with respect to p is:

$$\begin{aligned} N(I) = N(p, \Pi_{i \in I} q_i) = &\#\{x \in \mathbb{R} \mid p(x) = 0, \Pi_{i \in I} q_i(x) > 0\} \\ &- \#\{x \in \mathbb{R} \mid p(x) = 0, \Pi_{i \in I} q_i(x) < 0\}. \end{aligned}$$

The *matrix equation* is the relationship $M \cdot w = v$ between the following three entities:

- M, the l-by-m matrix with entries $M_{i,j} = \Pi_{k \in I_i} \tilde{\sigma}_j(q_k) \in \{-1, 1\}$ for $I_i \in S$ and $\tilde{\sigma}_j \in \tilde{\Sigma}$,
- w, the length m vector whose entries count the number of roots of p where $q_1, \ldots, q_n$ has sign assignment $\tilde{\sigma}$, i.e., $w_i = \#\{x \in \mathbb{R} \mid p(x) = 0, \operatorname{sgn}(q_j(x)) = \tilde{\sigma}_i(q_j) \text{ for all } 1 \leq j \leq n\}$,
- v, the length l vector consisting of Tarski queries for the subsets, i.e., $v_i = N(I_i)$.

Observe that the vector w is such that the sign assignment $\tilde{\sigma}_i$ is consistent (at a root of p) iff its corresponding entry w_i is nonzero. Thus, the matrix equation can be used to solve the sign determination problem by solving for w. In particular, the matrix M and the vector v are both computable from the input (candidate) sign assignments and subsets. Further, since the subsets will be chosen such that the constructed matrix M is *invertible*, the matrix equation uniquely determines w and the nonzero entries of $w = M^{-1} \cdot v$.

Matrix Equation: Construction. The simplest (base) case of the algorithm is when there is a single polynomial $[q_1]$. Here, it suffices to set up a matrix equation $M \cdot w = v$ from which we can compute all consistent sign assignments. As hinted at earlier, this can be done with the list of index subsets $[\{\}, \{1\}]$ and the candidate sign assignment list $[(+1), (-1)]$, where M is computed as in Def. 3. Taking a concrete example, suppose we want to find the list of consistent sign assignments for $\ell = [q_1] = [3x^3 + 2]$ at the zeros of $p = x^3 - x$. This is illustrated in Fig. 4.

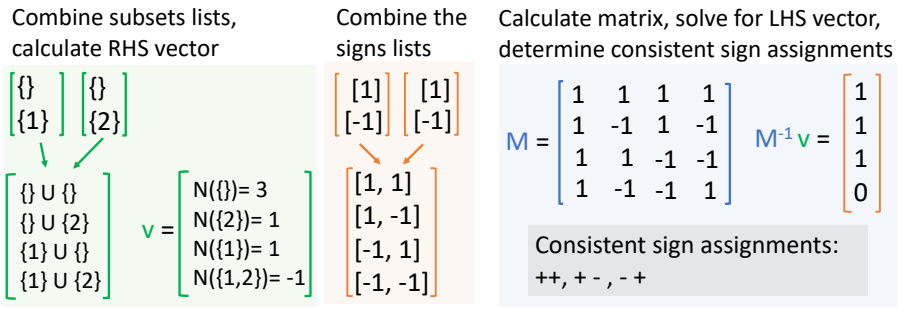


Figure 5: Combining two systems.

Now, extending our example, suppose we want to find the list of consistent sign assignments for $\ell = [q_1, q_2] = [3x^3 + 2, 2x^2 - 1]$ at the zeros of $p = x^3 - x$. We will form the base case systems for each of $\ell_1 = [q_1]$ and $\ell_2 = [q_2]$, and then we will need to combine these systems, which requires combining the subsets and signs lists.

Observe that any consistent sign assignment for ℓ must have a prefix that is itself a consistent sign assignment to ℓ_1 and a suffix that is itself a consistent sign assignment to ℓ_2. Thus, the combined list of sign assignments is obtained by concatenating every possible sign assignment for ℓ_1 with every possible sign assignment for ℓ_2. The combined subsets list S is obtained in an analogous way (where concatenation is now set union), with a slight modification: the subset list for ℓ_2 indexes polynomials from ℓ_2, but those polynomials now have different indices in ℓ, so we must reindex appropriately. Once we have the combined subsets list, we can calculate the RHS vector v with Tarski queries, as explained in Section 3.1.3. The matrix M is computed from the signs and subsets lists as prescribed by Def. 3[4], and the LHS vector w is calculated as $M^{-1}v$. The combination step for our example is visualized in Fig. 5.

Now, every combination step in BKR is followed by a reduction step to remove *inconsistent* sign assignments. The reduction step for the matrix equation with $p = x^3 - x$ and $\ell = [3x^3 + 2, 2x^2 - 1]$ is visualized in Fig. 6. Here, information about the inconsistent sign assignment $--$ is removed from the matrix equation. Although this example only has a small reduction in matrix size (from 4x4 to 3x3), in the next section, we will see (experimentally) the positive impact of BKR's reduction step on some larger examples.

Matrix Equation: Comments on Renegar. Remember that Renegar does not enforce the restriction that the q_i's are relatively prime to p in the construction of the matrix equation. To handle the possibility that some q_i's may share roots with p, each entry of Renegar's subset list is a *pair* of subsets I_1, I_2, and Tarski queries are computed as follows:

$$\begin{aligned} N(I_1, I_2) = &\#\{x \in \mathbb{R} \mid p(x) = 0, q_i(x) = 0 \ \forall i \in I_1, \Pi_{i \in I_2} q_i(x) > 0\} \\ &- \#\{x \in \mathbb{R} \mid p(x) = 0, q_i(x) = 0 \ \forall i \in I_1, \Pi_{i \in I_2} q_i(x) < 0\}. \end{aligned}$$

[4]Alternately, it holds that the combined matrix M is the Kronecker product of the matrices from the two systems that are being combined. Formalizing this allowed us to easily derive certain nice properties of M: for example, we prove the Kronecker product of two invertible matrices is itself invertible; it then follows that M is invertible as it is the Kronecker product of two invertible matrices.

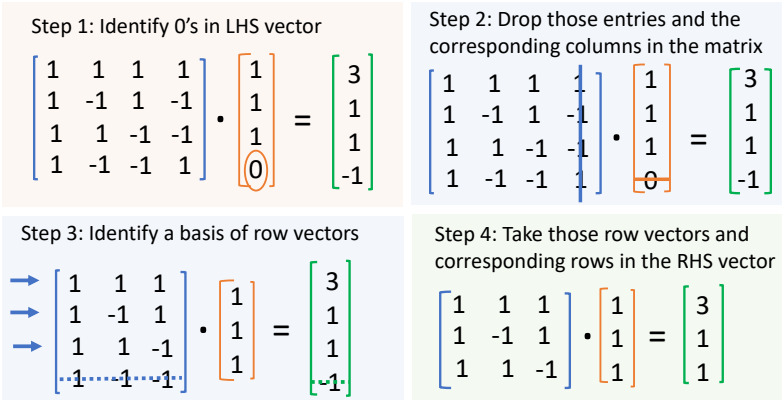


Figure 6: Reducing a system.

This generalization requires a slightly larger matrix equation—the base case is 3×3 instead of 2×2—and slightly more intricacy in the setup, and this is the construction that we intend to build on in the multivariate case (to avoid the complexity of generalizing coprimality to multivariate polynomials in Isabelle/HOL).

3.1.4 Univariate Formalization and Code Export

Our formalization of univariate BKR is approximately 7,000 lines in Isabelle/HOL and is available on the Archive of Formal Proofs (AFP) [11]. Our top-level formalized algorithms are called *decide_universal* and *decide_existential*, both with type *rat poly fml* ⇒ *bool*. The definition of *decide_existential* is as follows (the omitted definition of *decide_universal* is similar):

```
definition decide_existential :: "rat poly fml ⇒ bool"
where "decide_existential fml = (
let (fml_struct,polys) = convert fml in
  find (lookup_sem fml_struct) (find_consistent_signs polys) ≠ None)"
```

Here, *convert* extracts the list of constituent polynomials *polys* from the input formula *fml* along with the formula structure *fml_struct*, *find_consistent_signs* returns the list of all consistent sign assignments *conds* for *polys*, and *find* checks that predicate *lookup_sem fml_struct* is true at one of those sign assignments. Given a sign assignment σ, *lookup_sem fml_struct* σ evaluates the truth value of *fml* at σ by recursively evaluating the truth of its subformulas after replacing polynomials by their sign according to σ using the formula structure *fml_struct*. Thus, *decide_existential* returns true iff *fml* evaluates to true for at least one of the consistent sign assignments of its constituent polynomials.

The correctness theorem for *decide_universal* and *decide_existential* is shown below, where *fml_sem fml x* evaluates the truth of formula *fml* at the real value *x*.

```
theorem decision_procedure:
 "(∀x::real. fml_sem fml x) ⟷ decide_universal fml"
 "(∃x::real. fml_sem fml x) ⟷ decide_existential fml"
```

This theorem depends crucially on `find_consistent_signs` correctly finding the exact set of *all* consistent sign assignments for `polys`, i.e., solving the sign determination problem.

We export these decision procedures to Standard ML, compile with `mlton`, and test on 10 microbenchmarks from [19, Section 8]. While we did not perform extensive experiments (since our implementation is unoptimized), we compare the performance of our procedure using BKR sign determination versus an *unverified* implementation that naively uses the matrix equation (Section 3.1.3). We also compare to Li *et al.*'s `univ_rcf` CAD-based decision procedure [19] which can be directly executed as a proof tactic in Isabelle/HOL (code kindly provided by Wenda Li). The benchmarks were ran on an Ubuntu 18.04 laptop with 16GB RAM and 2.70 GHz Intel Core i7-6820HQ CPU. Results are in Table 1.

The most significant bottleneck in our current implementation is the computation of Tarski queries $N(p, q)$ when solving the matrix equation. Recall for our algorithm (Section 3.1.3) the input q to $N(p, q)$ is a *product* of (subsets of) polynomials appearing in the inputs. Indeed, Table 1 shows that the algorithm performs well when the factors have low degrees, e.g., ex1, ex2, ex4, and ex5. Conversely, it performs poorly on problems with many factors and higher degrees, e.g., ex3, ex6, and ex7. Further, as noted in experiments by Li and Paulson [20], the Sturm-Tarski theorem in Isabelle/HOL currently uses a straightforward method for computing remainder sequences which can also lead to significant (exponential) blowup in the bitsizes of rational coefficients of the involved polynomials. This is especially apparent for ex6 and ex7, which have large polynomial degrees and high coefficient complexity; these time out without completing even a single Tarski query. From Table 1, the BKR approach successfully reduces the number of Tarski queries as the number of input factors grows—the number of queries for BKR is dependent on the polynomial degrees and the number of consistent sign assignments, while the naive approach always requires exactly $(\frac{n}{2} + 1)2^n$ queries for n factors (which are reported in Table 1 whether completed or not). On the other hand, there is some overhead for smaller problems, e.g., ex1, ex3, that arises from the recursion in BKR.

The `univ_rcf` tactic relies on an external solver (we used Mathematica 12.1.1) to produce *untrusted* certificates which are then formally checked (by reflection) in Isabelle/HOL [19]. This procedure is optimized and efficient: except for ex7 where the tactic timed out, most of the time (roughly 3 seconds per example) is actually spent to start an instance of the external solver.

3.2 Multivariate BKR, Challenges, and Fallback Options

Now that we have detailed univariate BKR, we turn to the multivariate algorithm. In this section, we will explore the intuition underlying this algorithm, the challenges that its formalization will pose, and the fallback option of formalizing a slightly simpler multivariate QE algorithm.

Intuitively, the only parts of the construction of the matrix equation that are specific to univariate polynomials are 1) the computation of the Tarski queries and 2) the use of the Cauchy root bound. For 2), we can instead compute sign assignments at limit points directly, as in Renegar. For 1), intuitively the Tarski query can be generalized to multivariate polynomials because its computation only requires *sign information* on the coefficients of a polynomial.

Formula	**#Poly**	**#Factor**	**#$N(p,q)$ (Naive)**	**#$N(p,q)$ (BKR)**	**Time (Naive)**	**Time (BKR)**	**Time ([19])**
ex1	4 (12)	3 (1)	20	31	0.003	0.006	3.020
ex2	5 (6)	7 (1)	576	180	5.780	0.442	3.407
ex3	4 (22)	5 (22)	112	120	1794.843	1865.313	3.580
ex4	5 (3)	5 (2)	112	95	0.461	0.261	3.828
ex5	8 (3)	7 (3)	576	219	28.608	8.333	3.806
ex6	22 (9)	22 (8)	50331648	-	-	-	6.187
ex7	10 (12)	10 (11)	6144	-	-	-	-
ex1 $\wedge$ 2	9 (12)	9 (1)	2816	298	317.432	3.027	3.033
ex1 $\wedge$ 2 $\wedge$ 4	13 (12)	12 (2)	28672	555	-	51.347	3.848
ex1 $\wedge$ 2 $\wedge$ 5	16 (12)	14 (3)	131072	826	-	436.575	3.711

Table 1: Comparison of decision procedures using naive and BKR sign determination and Li *et al.*'s `univ_rcf` tactic in Isabelle/HOL [19]. All formulas are labeled following [19, Section 8]; formulas with $\wedge$ indicate conjunctions of the listed examples. Columns: **#Poly** counts the number of distinct polynomials appearing in the formula (maximum degree among polynomials in parentheses), **#Factor** counts the number of distinct factors from (1) in Section 3.1.2 (maximum degree among factors in parentheses), **#$N(p,q)$** counts the number of Tarski queries made by each approach, and **Time** reports time taken (seconds, 3 d.p.) for each decision procedure to run to completion. Cells with - indicate a timeout after 1 hour.

Let us expand on this point a little more closely. Recalling Theorem 1, we intend to treat multivariate polynomials in n variables as univariate polynomials (whose coefficients are polynomials in $n-1$ variables) and so compute remainder sequences of polynomials with attention to a single variable. These remainder sequences will be sequences of polynomials in $n-1$ variables rather than integers, but we only need to know the *signs* of those polynomials (rather than their values). That reduces the problem of sign determination for polynomials in n variables to a sign determination problem for polynomials in $n-1$ variables. In this way we intend to successively reduce multivariate computations to a series of (already formalized) univariate computations.

Consider the following concrete example: let $p = x^2y + 1$ and $q = xy + 1$. Suppose we choose to first eliminate y. If x is 0, then the analysis for the remaining $p = q = 1$ is simple. Otherwise, both x and x^2 are nonzero. Now, we calculate the remainder sequence from Theorem 1: $p_1 = x^2y + 1$, $p_2 = x^3y + x^2$, and $p_3 = -(1 - x)$. To find p_3, we calculate $x^2y + 1 = \frac{1}{x}(x^3y + x^2) + (1 - x)$, where $\frac{1}{x}$ is well-defined since $x \neq 0$.

The leading coefficients of p_1, p_2, and p_3 as polynomials in y are $a_1 = x^2$, $a_2 = x^3$, and $a_3 = -(1 - x)$. Here, we must use our univariate algorithm to fix some consistent sign assignment in x on the a_i's, taking into account our earlier stipulation that x and x^2 are nonzero. Say that we choose, for example, x positive, x^3 positive, and $-(1-x)$ negative. (A full QE procedure would need to consider *all* possible consistent sign assignments.) Because of our chosen sign assignment, a_1 is positive, a_2 is positive, and a_3 is negative. Still following Theorem 1, $S^+(p, q) = 1$ and $S^-(p, q) = 0$. The Tarski query $N(\{1\})$ is then computed as $N(\{1\}) = N(p, q) = S^-(p, q) - S^+(p, q) = -1$.

Following this intuition, multivariate BKR works inductively on the number of variables in the polynomial—thus treating multivariate polynomials as univariate polynomials with coefficients *that are themselves multivariate polynomials, but in one fewer variable*. Fortunately, it is highly likely that certain steps from the univariate BKR algorithm—particularly the univariate reduction step, which was the most challenging part of the univariate formalization—will essentially automatically generalize to the multivariate case. Intuitively, the reduction step requires no specific manipulations of polynomials, just manipulations of matrices, which can be treated the same way regardless of whether they are capturing sign-information for univariate polynomials or for multivariate polynomials.

However, despite this significant hope, formalizing multivariate BKR in Isabelle/HOL will pose significant technical challenges. First, working with multivariate polynomials in a theorem prover is considerably more challenging than working with univariate polynomials. Even, for example, treating a multivariate polynomial as a univariate polynomial becomes intricate in a type-theoretic setting (these two objects have different types in Isabelle/HOL). Some of the groundwork that we developed for manipulating multivariate polynomials in our formalization of virtual substitution will make these challenges more surmountable, but there is more work to be done.

Additionally, the multivariate BKR algorithm itself is somewhat nebulous, which makes translating it into a theorem prover more challenging: gaps will need to be filled in, and imprecisions must be rigorized. Additionally, certain steps that may sound simple at first glance will require considerable formalization effort. For example, the idea of computing sign assignments at limit points for multivariate polynomials sounds simple, but is somewhat intricate in the context of formalization. Say we want to find the consistent sign assignments of some list of multivariate polynomials at positive infinity with respect to a single variable. First, we must transform the list of multivariate polynomial into a list of univariate polynomials (in our variable of interest) with multivariate coefficients. Then we must perform a recursive call on *all coefficients* of that list of polynomials. Then, in each resulting consistent sign assignment, we must pick out *the first nonzero coefficient for every polynomial* and record its sign.[5] Each consistent sign assignment (CSA) from our recursive call corresponds to a valuation that realizes that sign assignment, and then the list of the signs of the first nonzero coefficients with respect to a particular CSA is the limit at positive infinity of the list of multivariate polynomials in the valuation that corresponds to the CSA. So one may observe how even conceptually simple steps necessitate intricate detail in the formalization.

Finally, the algorithm is highly recursive—even any step requiring a Tarski query requires a recursive call. This will make it difficult to modularize the formalization of the full BKR algorithm. Here, however, there is a fallback option: If formalizing the full multivariate BKR algorithm ends up being overly ambitious, a fallback option would be to formalize a naive multivariate QE algorithm. This algorithm would be more similar to Tarski's original algorithm in flavor, and thus quite theoretically similar to Cyril Cohen's formalization of Tarski's algorithm in the theorem prover Coq [5], and it would not be as efficient as the true BKR algorithm, as such an algorithm would trade in some efficiency in favor of ease of formalization.

[5]Note that it is not enough to do the recursive call on the leading coefficients precisely because some of those leading coefficients might be evaluated to zero in the resulting consistent sign assignments.

However, even this is an ambitious and technically challenging goal. The difficulties of working with multivariate polynomials in a theorem prover remain, and the naive algorithm shares considerable conceptual overlap with BKR, including a number of the deceptively simple steps (like computing sign assignments at limit points). Remember that there are currently only two fully formally verified multivariate QE algorithms, and neither is in Isabelle/HOL. Further, even a more naive algorithm can be augmented with limited reduction and other optimizations, and can be linked with virtual substitution and other preprocessing methods (see Section 2). This algorithm could also serve as a further stepping stone to verifying BKR in Isabelle/HOL.

4 Conclusion

The goal of the proposed thesis is to advance the state of formally verified support for QE by verifying a general-purpose multivariate QE algorithm based on the BKR algorithm (and its variant by Renegar) in Isabelle/HOL and linking this with efficient formalized special-purpose QE methods, notably VS. The bulk of the remaining work is in the formalization of the general-purpose multivariate algorithm.

4.1 Timeline and Fallback Options

I am hoping to graduate within the summer of 2023. I believe this can be accomplished by splitting the upcoming year into the following timeblocks:

1. **Summer 2022 (approx 3 months).** Virtual substitution could be considered completed for the purposes of this thesis. However, I have the opportunity to mentor an undergraduate student this upcoming summer, and I am planning to use that time to work with him on implementing limited and special-purpose QE methods, with the hope that we will be able to successfully identify and formalize a number of heavy-hitting optimizations. If we achieve practically significant speedup over the existing version of VS, we could submit to a conference (e.g. FM or ITP) in the spring; however, this is not an essential component of this thesis, as VS could be used as-is.

2. **Fall 2022 and Early Spring 2023 (approx 8 months).** I plan to use this time to finish formalizing a full multivariate QE algorithm, and to link this with VS. If all goes well, this algorithm will be somewhat BKR-like, in that it will implement reasonably aggressive reduction (for better efficiency). Given the ambitious and inherently uncertain nature of this, the fallback option is to formalize a more naive, somewhat more Tarski-like algorithm that implements more limited reduction (for better ease of formalization). There are a number of conferences in the fall and winter that would be good targets for this work; for example, the deadline for TACAS is usually in mid-October, and the deadline for LICS typically falls in January.

3. **Late Spring and Summer 2023 (approx 3 months).** Thesis writing, followed by thesis defense.

References

[1] Saugata Basu, Richard Pollack, and Marie-Françoise Roy. *Algorithms in Real Algebraic Geometry*. Springer, Berlin, Heidelberg, second edition, 2006. doi:10.1007/3-540-33099-2.

[2] Michael Ben-Or, Dexter Kozen, and John H. Reif. The complexity of elementary algebra and geometry. *J. Comput. Syst. Sci.*, 32(2):251–264, 1986. doi:10.1016/0022-0000(86)90029-2.

[3] Christopher W. Brown. Improved projection for cylindrical algebraic decomposition. *J. Symb. Comput.*, 32(5):447–465, 2001. doi:10.1006/jsco.2001.0463.

[4] Amine Chaieb. *Automated methods for formal proofs in simple arithmetics and algebra*. PhD thesis, Technische Universität München, 2008.

[5] Cyril Cohen. *Formalized algebraic numbers: construction and first-order theory*. PhD thesis, École polytechnique, Nov 2012.

[6] Cyril Cohen and Assia Mahboubi. Formal proofs in real algebraic geometry: from ordered fields to quantifier elimination. *Log. Methods Comput. Sci.*, 8(1), 2012. doi:10.2168/LMCS-8(1:2)2012.

[7] George E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In H. Barkhage, editor, *Automata Theory and Formal Languages*, volume 33 of *LNCS*, pages 134–183. Springer, 1975. doi:10.1007/3-540-07407-4_17.

[8] George E. Collins and H. Hong. Partial cylindrical algebraic decomposition for quantifier elimination. *J. Symb. Comput.*, 12(3):299–328, 1991. doi:10.1016/S0747-7171(08)80152-6.

[9] Katherine Cordwell, César A. Muñoz, and Aaron Dutle. Improving automated strategies for univariate quantifier elimination. Technical report, NASA, 2021. URL: https://shemesh.larc.nasa.gov/fm/papers/NASA-TM-20205010644.pdf.

[10] Katherine Cordwell, Yong Kiam Tan, and André Platzer. A verified decision procedure for univariate real arithmetic with the BKR algorithm. In Liron Cohen and Cezary Kaliszyk, editors, *ITP*, volume 193 of *LIPIcs*, pages 14:1–14:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ITP.2021.14.

[11] Katherine Cordwell, Yong Kiam Tan, and André Platzer. The BKR decision procedure for univariate real arithmetic. *Archive of Formal Proofs*, April 2021. https://www.isa-afp.org/entries/BenOr_Kozen_Reif.html, Formal proof development.

[12] 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 Marijn Heule and Sean A. Weaver, editors, *SAT*, volume 9340 of *LNCS*, pages 360–368. Springer, 2015. doi:10.1007/978-3-319-24318-4_26.

[13] Leonardo Mendonça de Moura and Nikolaj Bjørner. Z3: an efficient SMT solver. In C. R. Ramakrishnan and Jakob Rehof, editors, *TACAS*, volume 4963 of *LNCS*, pages 337–340. Springer, 2008. doi:10.1007/978-3-540-78800-3_24.

[14] Andreas Dolzmann and Thomas Sturm. REDLOG: computer algebra meets computer logic. *SIGSAM Bull.*, 31(2):2–9, 1997. doi:10.1145/261320.261324.

[15] Antonio J. Durán, Mario Pérez, and Juan L. Varona. The misfortunes of a trio of mathematicians using computer algebra systems. can we trust in them? *Notices of the AMS*, 61(10):1249–1252, 2014. doi:10.1090/noti1173.

[16] Nathan Fulton, Stefan Mitsch, Jan-David Quesel, Marcus Völp, and André Platzer. KeYmaera X: An axiomatic tactical theorem prover for hybrid systems. In Amy P. Felty and Aart Middeldorp, editors, *CADE*, volume 9195 of *LNCS*, pages 527–538. Springer, 2015. doi:10.1007/978-3-319-21401-6_36.

[17] Lars Hupel and Tobias Nipkow. A verified compiler from Isabelle/HOL to CakeML. In Amal Ahmed, editor, *ESOP*, volume 10801 of *LNCS*, pages 999–1026. Springer, 2018. doi:10.1007/978-3-319-89884-1_35.

[18] Wenda Li. The Sturm-Tarski theorem. *Archive of Formal Proofs*, September 2014. https://isa-afp.org/entries/Sturm_Tarski.html, Formal proof development.

[19] Wenda Li, Grant Olney Passmore, and Lawrence C. Paulson. Deciding univariate polynomial problems using untrusted certificates in Isabelle/HOL. *J. Autom. Reason.*, 62(1):69–91, 2019. doi:10.1007/s10817-017-9424-6.

[20] Wenda Li and Lawrence C. Paulson. Counting polynomial roots in Isabelle/HOL: A formal proof of the Budan-Fourier theorem. In *CPP*, page 52–64, New York, NY, USA, 2019. ACM. doi:10.1145/3293880.3294092.

[21] Scott McCallum. An improved projection operation for cylindrical algebraic decomposition. In B. F. Caviness, editor, *EUROCAL*, volume 204 of *LNCS*, pages 277–278. Springer, 1985. doi:10.1007/3-540-15984-3_277.

[22] Sean McLaughlin and John Harrison. A proof-producing decision procedure for real arithmetic. In Robert Nieuwenhuis, editor, *CADE*, volume 3632 of *LNCS*, pages 295–314. Springer, 2005. doi:10.1007/11532231_22.

[23] Casey B. Mulligan, Russell J. Bradford, James H. Davenport, Matthew England, and Zak Tonks. Quantifier elimination for reasoning in economics. *CoRR*, abs/1804.10037, 2018. URL: http://arxiv.org/abs/1804.10037, arXiv:1804.10037.

[24] César A. Muñoz, Anthony J. Narkawicz, and Aaron Dutle. A decision procedure for univariate polynomial systems based on root counting and interval subdivision. *J. Formaliz. Reason.*, 11(1):19–41, 2018. doi:10.6092/issn.1972-5787/8212.

[25] Anthony Narkawicz, César A. Muñoz, and Aaron Dutle. Formally-verified decision procedures for univariate polynomial computation based on Sturm's and Tarski's theorems. *J. Autom. Reason.*, 54(4):285–326, 2015. doi:10.1007/s10817-015-9320-x.

[26] Tobias Nipkow. Linear quantifier elimination. *J. Autom. Reason.*, 45(2):189–212, 2010. doi:10.1007/s10817-010-9183-0.

[27] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL - A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002. doi:10.1007/3-540-45949-9.

[28] Grant Olney Passmore. *Combined Decision Procedures for Nonlinear Arithmetics, Real and Complex*. PhD thesis, School of Informatics, University of Edinburgh, 2011.

[29] Lawrence C. Paulson and Jasmin Christian Blanchette. Three years of experience with Sledgehammer, a practical link between automatic and interactive theorem provers. In Geoff Sutcliffe, Stephan Schulz, and Eugenia Ternovska, editors, *IWIL*, volume 2 of *EPiC Series in Computing*, pages 1–11. EasyChair, 2010.

[30] André Platzer. *Logical Foundations of Cyber-Physical Systems*. Springer, Cham, 2018. doi:10.1007/978-3-319-63588-0.

[31] André Platzer, Jan-David Quesel, and Philipp Rümmer. Real world verification. In Renate A. Schmidt, editor, *CADE*, volume 5663 of *LNCS*, pages 485–501. Springer, 2009. doi:10.1007/978-3-642-02959-2_35.

[32] André Platzer and Yong Kiam Tan. Differential equation invariance axiomatization. *J. ACM*, 67(1):6:1–6:66, 2020. doi:10.1145/3380825.

[33] James Renegar. On the computational complexity and geometry of the first-order theory of the reals, part III: quantifier elimination. *J. Symb. Comput.*, 13(3):329–352, 1992. doi:10.1016/S0747-7171(10)80005-7.

[34] Matias Scharager, Katherine Cordwell, Stefan Mitsch, and André Platzer. Verified quadratic virtual substitution for real arithmetic. In Marieke Huisman, Corina S. Pasareanu, and Naijun Zhan, editors, *FM*, volume 13047 of *LNCS*, pages 200–217. Springer, 2021. doi:10.1007/978-3-030-90870-6_11.

[35] Matias Scharager, Katherine Cordwell, Stefan Mitsch, and André Platzer. Verified quadratic virtual substitution for real arithmetic. *Archive of Formal Proofs*, August 2021. https://www.isa-afp.org/entries/Virtual_Substitution.html, Formal proof development.

[36] Andrew Sogokon, Stefan Mitsch, Yong Kiam Tan, Katherine Cordwell, and André Platzer. Pegasus: Sound continuous invariant generation. *Form. Methods Syst. Des.*, 2021. Special issue for selected papers from FM'19. doi:10.1007/s10703-020-00355-z.

[37] Christian Sternagel and René Thiemann. Executable multivariate polynomials. *Archive of Formal Proofs*, August 2010. `https://www.isa-afp.org/entries/Polynomials.html`, Formal proof development.

[38] Thomas Sturm. Thirty years of virtual substitution: Foundations, techniques, applications. In Manuel Kauers, Alexey Ovchinnikov, and Éric Schost, editors, *ISSAC*, pages 11–16. ACM, 2018. `doi:10.1145/3208976.3209030`.

[39] Alfred Tarski. *A Decision Method for Elementary Algebra and Geometry.* RAND Corporation, Santa Monica, CA, 1951. Prepared for publication with the assistance of J.C.C. McKinsey.

[40] René Thiemann and Akihisa Yamada. Matrices, Jordan normal forms, and spectral radius theory. *Archive of Formal Proofs*, August 2015. `https://isa-afp.org/entries/Jordan_Normal_Form.html`, Formal proof development.

[41] René Thiemann, Akihisa Yamada, and Sebastiaan Joosten. Algebraic numbers in Isabelle/HOL. *Archive of Formal Proofs*, December 2015. `https://isa-afp.org/entries/Algebraic_Numbers.html`, Formal proof development.

[42] Volker Weispfenning. The complexity of linear problems in fields. *J. Symb. Comput.*, 5(1/2):3–27, 1988. `doi:10.1016/S0747-7171(88)80003-8`.

[43] Volker Weispfenning. Quantifier elimination for real algebra - the quadratic case and beyond. *Appl. Algebra Eng. Commun. Comput.*, 8(2):85–101, 1997. `doi:10.1007/s002000050055`.