Towards Efficient Quantifier Elimination in Mathematica

Katherine Cordwell, kcordwel@andrew.cmu.edu, 15-824 (Andrew ID: kcordwel)

Abstract

Efficient real arithmetic is crucial for the efficient formal verification of hybrid systems. Many proofs in KeYmaera X reduce to real arithmetic questions, often involving quantifier elimination, that are then outsourced to Mathematica. Unfortunately, many quantifier elimination questions that can be immediately resolved by a human are not quickly resolved in Mathematica. We study one of the most common algorithms for quantifier elimination, cylindrical algebraic decomposition (CAD), and develop some auxiliary algorithms for decision procedures in the strict universal fragment of first order real arithmetic. These algorithms capitalize on mathematical properties of polynomials and either admit the immediate resolution of the decision problem, or allow for the early termination of CAD. When the decision problem is false, in some cases our algorithms will naturally generate a witness—an assignment for the variables that makes the decision problem false. We test our algorithms in Mathematica.

1 Introduction

Although Alfred Tarski proved that quantifier elimination (QE) is decidable [13] in 1930, the first computationally feasible algorithm, **cylindrical algebraic decomposition (CAD)**, for QE was developed in 1975 by Collins [2]. Both [13] and [2] are seminal works, and CAD is still used extensively today. CAD takes as input a set of polynomials (say, in n variables) and constructs a sign-invariant decomposition of $\mathbb{R}^n$ composed of **cells**, which are subsets of $\mathbb{R}^n$, and a **sample point** for each cell. The sign of each input polynomial is constant on each cell, and so the signs of the polynomials at the (finitely many) sample points are representative of the signs of the polynomials on the entirety of $\mathbb{R}^n$. This set of sample points and cells is called a **CAD**.

Cylindrical algebraic decomposition works in two phases: projection and lifting (see Figure 1). The projection phase successively generates sets of polynomials where after each projection, one variable is eliminated. In the lifting phase, a CAD is constructed for the last projection set. Because this projection set contains only univariate polynomials, the CAD construction only requires finding the roots of univariate polynomials. Then, variables are added back in one by one as we "lift" each CAD for $k-1$ variables to a CAD for k variables.

The projection sets satisfy the following property: each polynomial in proj^{i-1} is *delineable* over each cell in proj^{i}. We omit the formal definition of delineability and instead focus on what delineability ensures: To paraphrase [1], the delineability property ensures that the projections of different cells are either disjoint or identical, so that if CAD A is lifted from CAD B, the cells in CAD A are cylindrically arranged over the cells in CAD B. We explain the significance of this cylindrical arrangement with an example in Section 1.1.

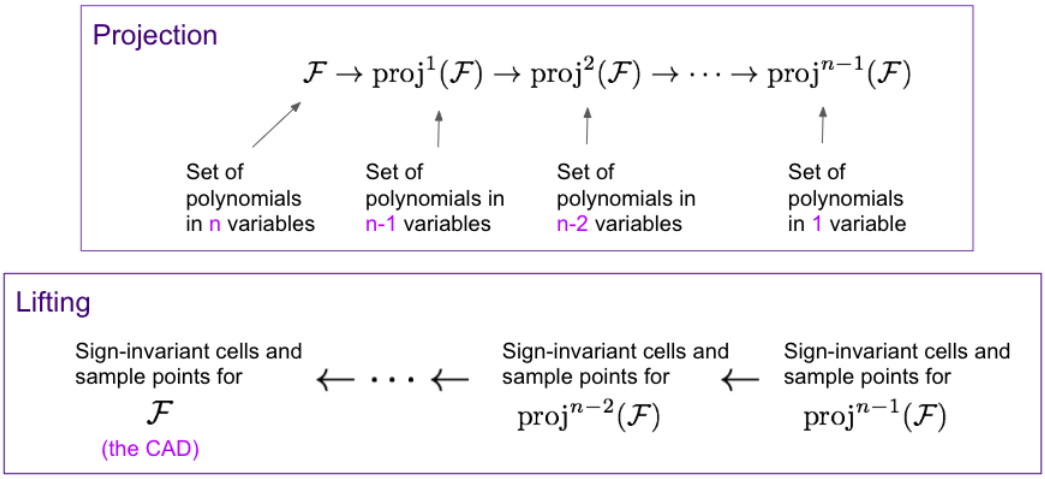

Figure 1: Overview of CAD

The projection operator *proj* is carefully designed so that delinability holds. The number of polynomials in the projection set has a significant influence on the efficiency of CAD, and so many improved projection sets have been developed for certain situations. We largely blackbox *proj*, but we note that the set $proj(f_1, \ldots, f_k)$ includes discriminants of f_i for all i (taken with respect to the particular variable that is being eliminated) and pairwise resultants $res(f_i, f_j)$ for all $i \neq j$ (again taken with respect to the particular variable that is being eliminated)—and in fact, as we will discuss later, even improved projection sets involve calculating these discriminants and resultants (see [1], [12]).

1.1 An Example (from [7])

Jirstrand gives a beautiful illustration of the lifting phase and the cylindrical arrangement of cells in his thesis [7]. We follow the example of CAD that he discusses in Section 5.4.2 of [7]. This example is for CAD construction for $f = (x_1 - 2)^2 + (x_2 - 2)^2 + (x_3 - 2)^2 - 1$. In the projection phase, first x_3 is eliminated, and then x_2 is eliminated.

Figure 2 is a modification of Figure 5.7 from [7]. It shows the CAD construction for this example geometrically. The sphere depicts the zeros of f, and the circle in the x_2x_1 plane depicts the zeros of $\text{proj}^1(f)$ (which is a set of bivariate polynomials in x_1 and x_2). The zeros for $\text{proj}^2(f)$, which is a set of univariate polynomials in x_1, are $x_1 = 1$ and $x_1 = 3$ (pictured in light red in the first and second of the three figures).

The first of the three figures shows the CAD for $\text{proj}^2(f)$. It contains the points 1 and 3 (shown in light red) and one arbitrary sample point from each of the intervals $(-\infty, 1), (1, 3)$, and $(3, \infty)$. The sample points from these intervals are shown in dark red. The second of the three figures shows the lifting of this CAD to a CAD for $\mathbb{R}^2$, the sample points of which are shown in blue. For clarity, we show the sample points of the CAD that intersect the circle (the zero set of $\text{proj}(f)$) in light blue, and sample points of the CAD that do not intersect the zero set of $\text{proj}(f)$ in dark blue. As we can see, since the line $x_1 = .5$ does not intersect the zero-set of the polynomials in $\text{proj}(f)$, $(.5, a)$ and $(.5, b)$ will have the same sign on $\text{proj}(f)$ for any a and b. Thus we only need one sample point when we lift $x_1 = .5$ to a CAD for $\text{proj}(f)$. However, the line $x_1 = 1$ intersects the zero-set of the polynomials in $\text{proj}(f)$ at $x_2 = 2$. So, we need to have three sample points when we lift $x_1 = 1$ to a CAD for $\text{proj}(f)$: $(1, a)$ (shown in dark blue), $(1, 2)$ (shown in light blue), and $(1, b)$ where $a < 2$ and $b > 2$ (shown in dark blue). This lifting works similarly for $2, 3$, and 3.5.

Note that we can set the first coordinates of the sample points for the CAD in $\mathbb{R}^2$ to be the values

of the sample points for the CAD in $\mathbb{R}$. This is what the cylindrical arrangement and delineability ensure—effectively, the lifting phase is reduced to finding roots of univariate polynomials.

The last of the three figures shows the lifting of the CAD for proj(f) to the CAD for f. Here, the green points are the sample points for the CAD for f. For clarity, we show the sample points that intersect the sphere (the zero set of f) in light green and all other sample points in dark green.

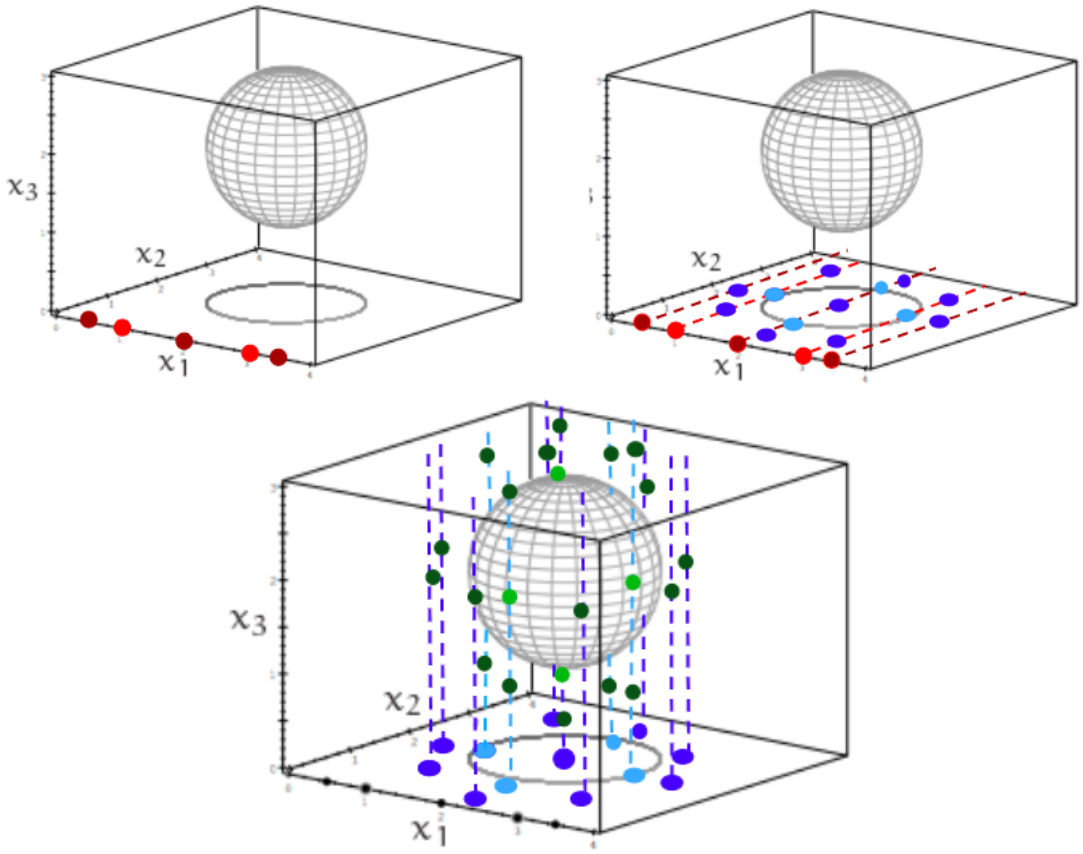

Figure 2: The Lifting Phase. Modified from Figure 5.7 in Jirstrand [7] (color added)

When we lift over a given sample point (a, b), we are fixing $x_1 = a$ and $x_2 = b$. We then take the line through $(a, b, 0)$ and $(a, b, 1)$ and look at the points where it intersects the sphere. If this line never intersects the sphere, then we only need one sample point when we lift (a, b). This is the case for all of the dark blue points except the one in the middle of the circle. If this line intersects the sphere in one point, say (a, b, z), then we need three sample points—(a, b, p), (a, b, z), and (a, b, q), where $p < z$ and $z < q$. This is the case for each of the light blue points. Finally, if the line intersects the sphere in two points, say (a, b, w) and (a, b, z) with $w < z$, then we need five sample points—$(a, b, p), (a, b, w), (a, b, q), (a, b, z)$, and (a, b, r) where $p < w$, $w < q < z$, and $z < r$. This is the case for the dark blue point in the middle of the circle (notice that (a, b, q) is inside the sphere and thus is not pictured). Again, we see visually how the CAD for f has been cylindrically lifted over the CAD for proj(f), and again we can see how delineability ensures that we can set the first two coordinates of the sample points for the CAD in $\mathbb{R}^3$ to be the coordinates of the sample points for the CAD in $\mathbb{R}^2$—so that the lifting phase really only requires finding roots of univariate polynomials.

Since CAD is a fundamental algorithm that is used extensively in QE procedures, improving CAD is an important and current area of research.

1.2 Approach

In this paper, we develop algorithms to allow for the early termination of CAD—specifically, we are interested in algorithms that allow CAD to stop before or during the projection phase, before lifting even begins. Our ultimate goal is to increase the efficiency of KeYmaera X. Since KeYmaera X outsources quantifier elimination problems to Mathematica, we develop our algorithms with a view towards improving Mathematica and benchmark them against Mathematica's QE procedures.

We primarily focus on resolving decision procedures in the strict universal fragment of first order real arithmetic (i.e., resolving queries of the form $\forall x_1, \forall x_2, \ldots, \forall x_k\ \phi$ where ϕ is a conjunction or disjunction of strict polynomial inequalities in $x_1, \ldots, x_k$). Some of the algorithms we develop naturally extend to other fragments as well (especially to the universal weak fragment). When there is a natural extension, we state it.

Additionally, the following observations allow us to further narrow our focus: Every formula ϕ is equivalent to a formula ψ in conjunctive normal form (CNF), and deciding whether $\forall x_1, \ldots, x_k\ \phi$ is true (where $x_1, \ldots, x_k$ are the variables that occur in ϕ) is the same as deciding whether $\forall x_1, \ldots, x_k\ \psi$ is true. When necessary, we convert formulas to their equivalent form in CNF using Mathematica's "BooleanConvert" method. Furthermore, because testing $\forall x_1, \ldots, x_k(\phi_1 \wedge \phi_2)$ is equivalent to testing $(\forall x_1, \ldots, x_k\ \phi_1) \wedge (\forall x_1, \ldots, x_k\ \phi_2)$, we can restrict our attention to disjunctions of strict polynomial inequalities. Also, we only need to consider strict polynomial inequalities of the form $f_i(x_1, \ldots, x_k) > 0$, because inequalities of the form $f_j(x_1, \ldots, x_k) < 0$ can be rewritten as $-1 \cdot (f_j(x_1, \ldots, x_k)) > 0$. Thus, we focus on the following fragment: $\forall x_1, \ldots, x_k(f_1(x_1, \ldots, x_k) > 0 \vee f_2(x_1, \ldots, x_k) > 0 \vee \cdots \vee f_n(x_1, \ldots, x_k) > 0)$.

Some of our algorithms naturally admit witness generation. When possible, an algorithm will not only return "False" but also return an assignment for each $x_1, \ldots, x_k$ which makes the decision problem false. Since many of our algorithms rely on limit observations, this assignment may set one or more x_i to ∞ or $-\infty$, and so our witnesses are in $(\mathbb{R} \cup \{-\infty, \infty\})^k$. From this, we obtain a "real" witness (i.e., a point in $\mathbb{R}^k$) by setting those x_i that are ∞ to be sufficiently large and by setting those x_i that are $-\infty$ to be sufficiently small.

1.3 Related Work

Over the years since its introduction, many people have worked to improve CAD. Notably, Hong and Collins developed partial CAD [3] in 1989. Partial CAD allows the CAD algorithm to stop when a partial sign-invariant decomposition is sufficient to perform quantifier elimination. Partial CAD is different from our proposed algorithms, because it focuses on early termination in the lifting phase—the goal of our algorithms is to stop CAD before or during the projection phase, whereas the first step in partial CAD is still to compute all projection factors, like in original CAD.

In the spirit of partial CAD, Wilson et. al. [15] developed the notion of cylindrical algebraic subdecompositions, which are subsets of CADs that already contain all of the information necessary to solve the problem at hand. They test the efficiency of their methods in Maple.

There have also been significant improvements to the projection phase itself. McCallum, whose work was later extended by Brown [1], provided conditions under which a smaller projection factor set is sufficient to construct a CAD. As previously mentioned, even Brown's reduced projection factor set requires calculating discriminants and resultants of polynomials. Accordingly, we have been studying properties of discriminants and resultants as well as general properties of polynomials.

Some improvements for CAD are specific to certain formulations of quantifier elimination. For example, Cylindrical Algebraic Decomposition, Maximal Dimension (CADMD) [8] is a more efficient variation of CAD that can be used on formulas $\exists x_1, \ldots, \exists x_k \phi$ where ϕ is a conjunction of strict inequalities in $x_1, \ldots, x_k$. The key point of CADMD is that it removes the need to compute with irrational algebraic numbers, which are computationally expensive and unwieldy. In effect, it

does so by ignoring sets of measure zero. We give a bit of the intuition behind CADMD. Say we want to determine whether $\exists(x_1, x_2)\ f(x_1, x_2) > 0$ where $f(x_1, x_2) = x_1^5 - 5x_1^2x_2 - 2x_2$. Then we would construct a CAD for the plane, so that in each region of the CAD we have that $f(x_1, x_2)$ is sign-invariant. Ultimately, however, we only care about the cells of the CAD that have nonzero measure—because if there is some point (a, b) where $f(a, b) > 0$, then because $x_1^5 - 5x_1^2x_2 - 2x_2$ is continuous, $f(x_1, x_2) > 0$ at all points in some neighborhood of (a, b). So we only need to test $f(x_1, x_2)$ at points from the cells of the CAD with nonzero measure—and these cells will contain rational points, because $\mathbb{Q}^2$ is dense in $\mathbb{R}^2$. Following this intuition, in CADMD lifting only takes place over cells of nonzero measure (which ensures that all cells will ultimately have nonzero measure) and so the representative points of each cell are rational [8]. Our work focuses on the strict universal fragment, which is different from the fragment improved by CADMD.

Because omitting sets of measure zero is so computationally significant, Mathematica uses Strzeboński's "GCAD" [12], which is similar to CADMD, to do so. GCAD uses a reduced projection set, but this reduce projection set still requires calculating discriminants and resultants of polynomials.

Grant Passmore [10] has also studied heuristic methods for CAD, although in a different context than we do. In his Ph.D. thesis, Passmore developed a tool RAHD that "contains original implementations of a vast array of real algebraic algorithms and decision procedures, and is designed to facilitate the combination of such techniques into custom heuristic proof procedures"[10]. Our focus is different than Passmore's, because we are interested in improving Mathematica, with a view towards increasing the speed of KeYmaera X. We also focus specifically on the early termination of CAD before or during the projection phase.

We now discuss properties of discriminants, resultants, and polynomials and associated heuristics or algorithms that allow CAD to terminate in or before the projection phase.

2 Properties of Discriminants, Resultants, and Polynomials

2.1 Properties of the Discriminant

Let $f = a_n x^n + \cdots + a_0$ where $a_0, \ldots, a_n \in \mathbb{R}$ and $a_n \neq 0$. The discriminant of f is defined as

$$\text{Disc}(f) = a_n^{2n-2} \Pi_{j<k} (r_j - r_k)^2,$$

where $r_1, \ldots, r_n \in \mathbb{C}$ are the n roots of f, counted with multiplicity [10]. Notice that f has n complex roots (not necessarily distinct) because f splits in $\mathbb{C}$.

In the projection phase of CAD, multivariate polynomials $f(x_1, \ldots, x_k)$ are treated as univariate polynomials in whichever variable we are eliminating and the discriminant of f is calculated with respect to that variable. For clarity, we write $\text{Disc}(f, x_i)$ when we are treating f as a univariate polynomial in x_i. When we calculate $\text{Disc}(f, x_i)$, we are implicitly assuming that the leading coefficient of x_i in f is nonzero. This assumption is permissible because the leading coefficient of x_i is also included in the projection factor set in CAD (so its zeros are treated separately).

For quadratic polynomials, the sign of the discriminant determines whether the polynomial has real roots. For higher-degree polynomials, the sign of the discriminant can provide some information about the existence of real roots of the polynomial. The following result for the discriminant of a univariate polynomial is known, but as we did not find a reference with proof, we provide our own proof.

Lemma 1. *Take a polynomial $f = a_{2n} x^{2n} + \cdots + a_0$ with real coefficients and $a_{2n} \neq 0$ (i.e., an even-degree polynomial). If $2n = 0 \mod 4$ and the discriminant of f is ≤ 0, then f has real roots, and if $2n = 2 \mod 4$ and the discriminant of f is ≥ 0, then f has real roots.*

Proof. Take some polynomial $f = a_{2n}x^{2n} + \cdots + a_0$ with real coefficients and no real roots. Order the complex roots of f as $r_1, r_2, \ldots, r_{2n-1}, r_{2n}$, so that r_{j+1} is the complex conjugate of r_j for each odd j. By definition [5], the discriminant of f is $a_{2n}^{4n-2}\Pi_{j<k}(r_j - r_k)^2$. Since $a_{2n}^{4n-2} = (a_n^{2n-1})^2 > 0$, the sign of the discriminant is the sign of $\Pi_{j<k}(r_j - r_k)^2$. Notice that for each r_j and r_k with $j < k$, unless $r_k = \overline{r_j}$, then both r_j and $\overline{r_j}$ must occur in the ordered list before r_k and $\overline{r_k}$. Now, $(r_j - r_k)^2(\overline{r_j} - r_k)^2(r_j - \overline{r_k})^2(\overline{r_j} - \overline{r_k})^2 > 0$ (to see this, let $r_j = c + di$ and $r_k = w + zi$ for arbitrary $c, d, w, z \in \mathbb{R}$, and expand). From this, we see that the sign of the discriminant reduces to the sign of $\Pi_j(r_j - \overline{r_j})^2$. Since each $(r_j - \overline{r_j})^2$ has the sign of $i^2 = -1$, the product $\Pi_j(r_j - \overline{r_j})^2$ has the sign of $(-1)^n$. Thus, if $2n = 0 \bmod 4$ and the discriminant is ≤ 0, then f has real roots, and if $2n = 2 \bmod 4$ and the discriminant is ≥ 0, then f has real roots. □

Notice that this does not give a complete characterization of whether f has real roots. The test is inconclusive for cases when $2n = 0 \bmod 4$ and the discriminant of f is > 0 or when $2n = 2 \bmod 4$ and the discriminant of f is < 0.

For univariate polynomials, perhaps the most well-known property of the discriminant is that the discriminant of a polynomial f is 0 iff f has a multiple root [5]. The following corollary is a direct result of this property.

Corollary 1. *Given a polynomial $f \in \mathbb{R}[x_1, \ldots, x_n]$, if $Disc(f, x_1)(q) = 0$ for some point $q \in \mathbb{R}^{n-1}$ and the leading coefficient of x_1 in f (which is a polynomial in $\mathbb{R}[x_2, \ldots, x_n]$) evaluated at q is nonzero, then the decision problem $\forall x_1, \ldots, x_n\ f(x_1, \ldots, x_n) > 0$ is false.*

Unfortunately, this does not generalize easily to disjunctions of polynomials, because knowing that f has a root does not tell us anything about whether $\forall x_1, \ldots, x_n(f(x_1, \ldots, x_n) > 0 \vee g(x_1, \ldots, x_n) > 0)$ is false, unless we know exactly what that root is. However, in select cases this property may still help us terminate CAD in the projection phase. For example, if we are trying to resolve $\forall x, f(x) > 0 \wedge g(x) \leq 0 \wedge h(x) > 0$, then in the projection phase of CAD we will calculate the discriminants of f, g, and h. If it is immediately apparent that the discriminant of f or h has roots, we can return False right away.

2.2 Properties of Resultants

Let $f = a_n x^n + \cdots + a_0$ where $a_n \neq 0$ and $g = b_m x^m + \cdots + b_0$ where $b_m \neq 0$. Then the resultant of f and g, $\text{Res}(f, g)$, is defined as the determinant of the following matrix (called the Sylvester matrix of f and g) [4] [10]:

$$\begin{vmatrix}
a_n & a_{n-1} & \cdots & a_0 & & \\
 & a_n & a_{n-1} & \cdots & a_0 & \\
 & & \ddots & & & \\
 & & a_n & a_{n-1} & \cdots & a_0 \\
b_m & b_{m-1} & \cdots & b_0 & & \\
 & b_m & b_{m-1} & \cdots & b_0 & \\
 & & \ddots & & & \\
 & & b_m & b_{m-1} & \cdots & b_0
\end{vmatrix}$$

Resultants are significantly more complicated than discriminants (really, the discriminant is a special type of resultant [5]). In CAD, $\text{proj}(S)$ contains $\text{Res}(f, g)$ for each pair of polynomials f and g in S. If f and g are multivariate polynomials, then they are treated as univariate polynomials in whatever variable we are currently eliminating in the projection phase. For clarity, we write $\text{Res}(f, g, x_k)$ when we are treating f and g as univariate polynomials in x_k. Again, this assumes

that the leading coefficients of x_k in f and g are nonzero, but since CAD handles leading coefficients separately, this is fine.

One important property (perhaps the most canonical property) of the resultant is that $\text{Res}(f, g) = 0$ iff f and g have a common root [5]. As a direct result of this property, we have the following corollaries.

Corollary 2. *Let f and g be polynomials in variables $x_1, \ldots, x_n$. Treating f and g as univariate polynomials in x_1, calculate the resultant $Res(f, g, x_1)$. Say that the leading coefficient of x_1 in f is a, where $a \in \mathbb{R}[x_2, \ldots, x_n]$ and the leading coefficient of x_1 in g is b, where $b \in \mathbb{R}[x_2, \ldots, x_n]$. If there exists a point $q \in \mathbb{R}^{n-1}$ where $Res(f, g, x_1)(q) = 0$ and $a(q) \neq 0$ and $b(q) \neq 0$, then f and g have a common root.*

Corollary 3. *Given $f \in \mathbb{R}[x_1, \ldots, x_n]$, if there exists a polynomial $g \in \mathbb{R}[x_1, \ldots, x_n]$ and a point $q \in \mathbb{R}^{n-1}$ where $Res(f, g, x_1)(q) = 0$ and the leading coefficients of x_1 in f and g (which are polynomials in $\mathbb{R}[x_2, \ldots, x_n]$) are nonzero when evaluated at q, then the decision problem $\forall x_1, \ldots, x_n \; f(x_1, \ldots, x_n) > 0$ is false.*

Corollary 4. *Given $f, g \in \mathbb{R}[x_1, \ldots, x_n]$, if there exists a point $q \in \mathbb{R}^{n-1}$ where $Res(f, g, x_1)(q) = 0$ and the leading coefficients of x_1 in f and g (which are polynomials in $\mathbb{R}[x_2, \ldots, x_n]$) are nonzero when evaluated at q, then the decision problem $\forall x_1, \ldots, x_n \; (f(x_1, \ldots, x_n) > 0 \vee g(x_1, \ldots, x_n) > 0$ is false.*

Unfortunately, this does not generalize easily to arbitrary disjunctions of polynomials, because knowing that f and g have a common root does not allow us to decide $\forall x_1, \ldots, x_n \; (f(x_1, \ldots, x_n) > 0 \vee g(x_1, \ldots, x_n) > 0 \vee h(x_1, \ldots, x_n) > 0)$. However, in select cases this property may help us terminate CAD in the projection phase. For example, if we are trying to resolve $\forall x, f(x) > 0 \vee g(x) > 0$, then in the projection phase of CAD we will calculate the pairwise resultant of f and g. If it is immediately apparent that the this resultant has roots, we can return False right away.

2.3 Observations on polynomials

Because calculating discriminants and resultants can be quite computationally expensive, the more auxiliary algorithms we develop that don't use these calculations, the better. Accordingly, we consider how properties of polynomials can immediately resolve decision problems in the universal strict fragment in the following propositions.

We adopt the convention that the leading coefficient of a polynomial is nonzero unless the polynomial is the zero polynomial; i.e., the leading coefficient of x_1 in the polynomial $(x_2 - x_2)x_1^5 + 2x_3x_1$ is $2x_3$, since $x_2 - x_2$ is identically zero (and $2x_3$ is not identically zero). In other words, we assume that polynomials have been sufficiently simplified so that $(x_2 - x_2)x_1^5 + 2x_3x_1$ would have already been rewritten as $2x_3x_1$. In Mathematica, we can use the "Simplify" command to appropriately reduce polynomials.

First, we consider the presence of odd-degree variables in our polynomials.

Proposition 1. *Given a decision problem of the form $\forall x_1, \ldots, x_n \; f(x_1, \ldots, x_n) > 0$, where $\sim \in \{>, \geq\}$, if any variable x_i has odd degree, then $\forall x_1, \ldots, x_n \; f(x_1, \ldots, x_n) \sim 0$ is false.*

Proof. WLOG, say that x_1 has degree d where d is odd. Then f is of the form $a_d x_1^d + \cdots + a_0$ for $a_d, \ldots, a_0 \in \mathbb{R}[x_2, \ldots, x_n]$. Because we have assumed that leading coefficients are nonzero unless f is the zero polynomial (which it is clearly not since x_1 has odd degree), we can choose fixed real values for $x_2, \ldots, x_n$ so that the leading coefficient a_d of x_1 in f is nonzero and substitute these values into f. This gives us a univariate polynomial $g(x_1)$ where x_1 has odd degree in g. Let c_d be the leading coefficient of x_1 in g. (Note that $c_d \in \mathbb{R}$.) Now, $\lim_{x_1 \to -\infty} g(x_1) = \lim_{x_1 \to -\infty} c_d x^d = c_d \cdot (-1)^d \cdot \infty =$

$-c_d \cdot \infty$, as d is odd. So, if c_d is positive, then $\lim_{x_1 \to -\infty} g(x_1) = -\infty$, so if $p_1 > 0$ is sufficiently large, $g(-p_1)$ will be negative. If instead $c_d < 0$, then $\lim_{x_1 \to \infty} g(x_1) = \lim_{x_1 \to -\infty} c_d x^d = c_d \cdot \infty = -\infty$. So, if $p_1 > 0$ is sufficiently large, $g(p_1)$ will be negative. In both cases, there exist values for $x_1, \ldots, x_n$ so that $f(x_1, \ldots, x_n) \not\sim 0$. □

This generalizes to disjunctions of polynomials as follows:

Theorem 1. *Consider a decision problem of the form* $\forall x_1, \ldots, x_n \ (f_1(x_1, \ldots, x_n) \sim_1 0 \vee f_2(x_1, \ldots, x_n) \sim_2 0 \vee \cdots \vee f_k(x_1, \ldots, x_n) \sim_k 0)$*, where* $\sim_i \in \{>, \geq\}$ *for* $1 \leq i \leq k$*. Say that the leading coefficient of* x_i *in* f_j *is* $a_{i,j} \in \mathbb{R}[x_2, \ldots, x_n]$*. If any variable* x_i *has odd degree in each of* $f_1, \ldots, f_k$ *and if there exist* $p_2, \ldots, p_n \in \mathbb{R}$ *such that all of the* $a_{i,j}(p_2, \ldots, p_n)$*,* $1 \leq j \leq k$ *are are either all* > 0 *or all* < 0*, then* $\forall x_1, \ldots, x_n \ f(x_1, \ldots, x_n) \sim 0$ *is false. If we explicitly find such* $p_2, \ldots, p_n$*, then we return* $(-\infty, p_2, \ldots, p_n)$ *as a witness if* $a_{i,j}(p_2, \ldots, p_n) > 0$ *for all* $1 \leq j \leq k$ *and* $(\infty, p_2, \ldots, p_n)$ *as a witness if* $a_{i,j}(p_2, \ldots, p_n) < 0$ *for all* $1 \leq j \leq k$*.*

Proof. WLOG, say that x_1 has degree d_j in each f_j, $1 \leq j \leq k$ and that d_j is odd, $1 \leq j \leq k$. By assumption, all of the $a_{1,1}(p_2, \ldots, p_n), a_{1,2}(p_2, \ldots, p_n), \ldots, a_{1,k}(p_2, \ldots, p_n)$ have the same sign. First assume that $a_{1,j}(p_2, \ldots, p_n) > 0$ for $1 \leq j \leq k$. Then, for each i,

$$\lim_{x_1 \to -\infty} f_i(x_1, p_2, \ldots, p_n) = \lim_{x_1 \to -\infty} a_{i,1}(p_2, \ldots, p_n) x_1^{d_i} = a_{i,1}(p_2, \ldots, p_n)(-1)^{d_i} \cdot \infty.$$

Since $a_{i,1}(p_2, \ldots, p_n) > 0$ and d_i is odd, this limit is $-\infty$ for each i, and thus if p_1 is sufficiently large, for all i, $f_i(-p_1, p_2 \ldots, p_n) \not\sim 0$, and so the decision problem is false, and if we have explicitly found $p_2, \ldots, p_n$, we return $(-\infty, p_2, \ldots, p_n)$ as a witness.

If instead $a_{1,j}(p_2, \ldots, p_n) < 0$ for $1 \leq j \leq k$, then

$$\lim_{x_1 \to \infty} f_i(x_1, p_2, \ldots, p_n) = \lim_{x_1 \to \infty} a_{i,1}(p_2, \ldots, p_n) x_1^{d_i} = a_{i,1}(p_2, \ldots, p_n) \cdot \infty,$$

and this is $-\infty$ since $a_{1,j}(p_2, \ldots, p_n) < 0$. Thus, if p_1 is sufficiently large, for all i, $f_i(p_1, p_2 \ldots, p_n) \not\sim 0$, and so the decision problem is false, and if we have explicitly found $p_2, \ldots, p_n$, we return $(\infty, p_2, \ldots, p_n)$ as a witness. □

Now, we consider what happens when variables are set to 0 in the following proposition, which is phrased as a property of x_1 for simplicity, but is actually applicable to any x_i.

Proposition 2. *Consider a decision problem of the form* $\forall x_1, \ldots, x_n \ f(x_1, \ldots, x_n) > 0$*. Treat* f *as a univariate polynomial in* x_1*, and write* $f(x_1, \ldots, x_n) = a_m x_1^m + \cdots + a_0$*, where* $a_m, \ldots, a_0 \in \mathbb{R}[x_2, \ldots, x_n]$*. Then if there exist* $p_2, \ldots, p_n \in \mathbb{R}$ *so that* $a_0(p_2, \ldots, p_n) \leq 0$*, the decision problem* $\forall x_1, \ldots, x_n \ f(x_1, \ldots, x_n) > 0$ *is false. If we can find* $p_2, \ldots, p_n$ *explicitly, we return* $(0, p_2, \ldots, p_n)$ *as a witness.*

Proof. If $p_2, \ldots, p_n \in \mathbb{R}$ are such that $a_0(p_2, \ldots, p_n) \leq 0$, then $f(0, p_2, \ldots, p_n) \leq 0$ and thus the answer to the decision problem is false. □

This generalizes to disjunctions of polynomials as follows:

Theorem 2. *Consider a decision problem of the form* $\forall x_1, \ldots, x_n \ (f_1(x_1, \ldots, x_n) > 0 \vee \cdots \vee f_k(x_1, \ldots, x_n) > 0)$*. Treat* f_i *as a univariate polynomial in* x_1 *and write* $f_i(x_1, \ldots, x_n) = a_{m_i} x_1^{m_i} + \cdots + a_{0_i}$ *where* $a_{m_i}, \ldots, a_{0_i} \in \mathbb{R}[x_2, \ldots, x_n]$*. Then if there exist* $p_2, \ldots, p_n \in \mathbb{R}$ *so that* $a_{0,i}(p_2, \ldots, p_n) \leq 0$ *for all* i*, the decision problem is false. If we can find* $p_2, \ldots, p_n$ *explicitly, we return* $(0, p_2, \ldots, p_n)$ *as a witness.*

Proof. Say that $p_2, \ldots, p_n \in \mathbb{R}$ are such that $a_{0,i}(p_2, \ldots, p_n) \leq 0$ for all i. By Proposition 2, $f_i(0, p_2, \ldots, p_n) \leq 0$ for all i, and thus $(f_1(0, p_2 \ldots, p_n) > 0 \vee \cdots \vee f_k(0, p_2 \ldots, p_n) > 0$ is false, and so the decision problem is false. □

As one special case of this, if all of the polynomials $f_1, \ldots, f_k$ have nonpositive constant terms, the decision problem as stated in the theorem is false. We also have the following corollaries for the universal weak fragment (which are proved in the same way as Proposition 2 and Theorem 2).

Corollary 5. *Consider a decision problem of the form $\forall x_1, \ldots, x_n\ f(x_1, \ldots, x_n) \geq 0$. Treat f as a univariate polynomial in x_1, and write $f(x_1, \ldots, x_n) = a_m x_1^m + \cdots + a_0$. Then if there exist $p_2, \ldots, p_n \in \mathbb{R}$ so that $a_0(p_2, \ldots, p_n) < 0$, the decision problem $\forall x_1, \ldots, x_n\ f(x_1, \ldots, x_n) \geq 0$ is false. If we can find $p_2, \ldots, p_n$ explicitly, we return $(0, p_2, \ldots, p_n)$ as a witness.*

Corollary 6. *Consider a decision problem of the form $\forall x_1, \ldots, x_n\ (f_1(x_1, \ldots, x_n) \sim_1 0 \vee f_2(x_1, \ldots, x_n) \sim_2 0 \vee \cdots \vee f_k(x_1, \ldots, x_n) \sim_k 0)$, where $\sim_i \in \{>, \geq\}$ for $1 \leq i \leq k$. Treat f_i as a univariate polynomial in x_1 and write $f_i(x_1, \ldots, x_n) = a_{m_i} x_1^{m_i} + \cdots + a_{0_i}$ where $a_{m_i}, \ldots, a_{0_i} \in \mathbb{R}[x_2, \ldots, x_n]$. Then if there exist $p_2, \ldots, p_n \in \mathbb{R}$ so that $a_{0,i}(p_2, \ldots, p_n) \sim_i' 0$ for all i, where $\sim_i'$ is $\leq$ if $\sim_i$ is $>$ and $\sim_i'$ is $<$ if $\sim_i$ is $\geq$, the decision problem is false. If we can find $p_2, \ldots, p_n$ explicitly, we return $(0, p_2, \ldots, p_n)$ as a witness.*

Next, we consider the importance of the signs of individual monomials

Proposition 3. *If the real coefficient $c \in \mathbb{R}$ of the highest-degree monomial in $f(x, x, \ldots, x)$ is negative or equal to zero, then the decision problem $\forall x_1, \ldots, x_n\ f(x_1, \ldots, x_n) > 0$ is false. As a witness, we return $(\infty, \infty, \ldots, \infty)$.*

Proof. Let the real coefficient of the highest-degree monomial n $f(x, \ldots, x)$ be c.

If $c = 0$, then that means that $f(k, \ldots, k) = 0$ for any $k \in \mathbb{R}$, and so $\forall x_1, \ldots, x_n\ f(x_1, \ldots, x_n) > 0$ is false. We could return any witness of the form $(k, \ldots, k)$; in particular, we can return $(\infty, \infty, \ldots, \infty)$.

Otherwise, $c < 0$ by assumption, and so $\lim_{x \to \infty} f(x, \ldots, x) = -\infty$. Thus for $x = p$ large enough, we will have $f(p, \ldots, p) < 0$ and thus the decision problem is false with witness $(\infty, \infty, \ldots, \infty)$. □

This generalizes to disjunctions of polynomials as follows:

Theorem 3. *If the real coefficient $c_i \in \mathbb{R}$ of the highest-degree monomial in $f_i(x, \ldots, x)$ is negative or zero for all $1 \leq i \leq k$, then the decision problem $\forall x_1, \ldots, x_n\ (f_1(x_1, \ldots, x_n) > 0 \vee \cdots \vee f_k(x_1, \ldots, x_n))$ is false. As a witness, we return $(\infty, \infty, \ldots, \infty)$.*

Proof. Let the real coefficient of the highest-degree monomial in $f_i(x, \ldots, x)$ be c_i. Because $c_i \leq 0$ by assumption, by Proposition 3, $f_i(p_i, \ldots, p_i) \leq 0$ for p_i large enough. Thus if we take $p = \max\{p_1, \ldots, p_k\}$, we will have $f_i(p, \ldots, p) \leq 0$ for all $1 \leq i \leq k$, and so $\forall x_1, \ldots, x_n\ (f_1(x_1, \ldots, x_n) > 0 \vee \cdots \vee f_k(x_1, \ldots, x_n))$ is false with witness $(\infty, \ldots, \infty)$. □

We also have the following corollary for the universal weak fragment, which is proved in the same way:

Corollary 7. *Consider the decision problem $\forall x_1, \ldots, x_n\ (f_1(x_1, \ldots, x_n) \sim_1 0 \vee \cdots \vee f_k(x_1, \ldots, x_n) \sim_k 0)$ for $\sim_i \in \{>, \geq\}$, $1 \leq i \leq k$.*

Let c_i be the real coefficient $c_i \in \mathbb{R}$ of the highest-degree monomial in $f_i(x, \ldots, x)$. If c_i is negative or zero for all $1 \leq i \leq k$ where $\sim_i$ is $>$ and if c_i is negative for all $1 \leq i \leq k$ where $\sim_i$ is $\geq$, then the decision problem is false. As a witness, we return $(\infty, \infty, \ldots, \infty)$.

Next, we have the following property of coefficients, which is phrased as a property of x_1, but holds for any x_i.

Proposition 4. *If the leading real coefficient of x_1 in $f(x_1, 1, \ldots, 1)$ has negative sign, then the decision problem $\forall x_1, \ldots, x_n \; f(x_1, \ldots, x_n) > 0$ is false. We return as a witness $(-\infty, 1, \ldots, 1)$.*

Proof. Consider $f(x_1, 1, \ldots, 1)$. This is a univariate polynomial in x_1. Write it as $c_k x_1^k + \cdots + c_0$ for $c_k, \ldots, c_0 \in \mathbb{R}$. By assumption, $c_k < 0$. Notice that $\lim_{x_1 \to \infty} c_k x_1^k + \cdots + c_0 = -\infty$. Thus, for large enough p, if $x_1 = p$, $f(p, 1, \ldots, 1) > 0$ and so the decision problem is false. □

This generalizes to disjunctions of polynomials as follows.

Theorem 4. *If the leading real coefficient of x_1 in $f_i(x_1, 1, \ldots, 1)$ has negative sign for all $1 \leq i \leq k$, then the decision problem $\forall x_1, \ldots, x_n \; f_1(x_1, \ldots, x_n) > 0 \vee \cdots \vee f_k(x_1, \ldots, x_n) > 0$ is false. As a witness, we return $(-\infty, 1, \ldots, 1)$.*

Proof. Say that the leading real coefficient of x_1 in $f_i(x_1, 1, \ldots, 1)$ has negative sign for all $1 \leq i \leq k$. Then by Proposition 4, for each $1 \leq i \leq k$ there is p_i large enough so that $f_i(p_i, 1, \ldots, 1) < 0$.

Thus if we take $p = \max\{p_1, \ldots, p_k\}$, $f_i(p, 1, \ldots, 1) < 0$ for all $1 \leq i \leq k$, and thus the decision problem $\forall x_1, \ldots, x_n \; f_1(x_1, \ldots, x_n) > 0 \vee \cdots \vee f_k(x_1, \ldots, x_n) > 0$ is false with witness $(-\infty, 1, \ldots, 1)$. □

This last result is different from the previous results, because it is a way to determine that a decision problem is true rather than false.

Theorem 5. *Consider the decision problem $\forall x_1, \ldots, x_n \; (f_1(x_1, \ldots, x_n) \sim_1 0 \vee \cdots \vee f_k(x_1, \ldots, x_n) \sim_k 0)$ for $\sim_i \in \{>, \geq\}$, $1 \leq i \leq k$.*

Assume that all variables have even degree in all monomials (so that no variable ever occurs with odd exponent). If for some $1 \leq i \leq k$, $\sim_i$ is $>$ and the constant term in $f_i(0, \ldots, 0)$ is positive, then the decision problem is true.

Also, if for some $1 \leq i \leq k$, $\sim_i$ is $\geq$ and the constant term in $f_i(0, \ldots, 0)$ is positive, then the decision problem is true.

Proof. Because $x^{2i} \geq 0$ for all $i \in \mathbb{N}$, if all variables have even degree in all monomials, then $f_i(x_1, \ldots, x_n) \geq f_i(0, \ldots, 0)$ for all $1 \leq i \leq k$ and all $x_1, \ldots, x_n \in \mathbb{R}$.

Say that there is an i where $\sim_i$ is $>$ and $f_i(0, \ldots, 0) > 0$. Then because $f_i(x_1, \ldots, x_n) \geq f_i(0, \ldots, 0)$ for all $x_1, \ldots, x_n \in \mathbb{R}$, $f_i(x_1, \ldots, x_n) > 0$ for all $x_1, \ldots, x_n \in \mathbb{R}$, and the decision problem is true (since at least one of the atoms in the disjunction is true).

Now, say instead that there is an i where $\sim_i$ is $\geq$ and $f_i(0, \ldots, 0) \geq 0$. Then because $f_i(x_1, \ldots, x_n) \geq f_i(0, \ldots, 0)$ for all $x_1, \ldots, x_n \in \mathbb{R}$, $f_i(x_1, \ldots, x_n) \geq 0$ for all $x_1, \ldots, x_n \in \mathbb{R}$, and the decision problem is true (since at least one of the atoms in the disjunction is true). □

3 Algorithms

We use these properties to develop algorithms for quantifier elimination. We have implemented these algorithms in Mathematica.

3.1 Discriminant Algorithm

We combine Proposition 1 and Lemma 1 in the following algorithm.

```
DiscAlg: Test whether ∀x1∀x2...∀xk f(x1,...,xk) ~ 0, for ~ ∈ {>,<}
Given: A polynomial f, a set Variables of the variables of f, and a sign ~
   (either > or <)
Return: false or inconclusive

simplify f, if possible
test the sign of f at some point p ∈ R^k
   if f(p) ≤ 0 and ~ is > or f(p) ≥ 0 and ~ is <
      return false
   if any variable in f has odd degree
      return false
   if Variables is empty
      return inconclusive
   else
      let x1 be the first variable in Variables
      take the discriminant Disc(f,x1) of f wrt x1, and simplify it if necessary
      test its sign at some point q ∈ R^(k-1) where the leading coefficient of x1
      in f is nonzero at q
         if deg(f,x1) = 0 mod 4
            if Disc(f,x1)(q) ≤ 0
               return false
            else
               call DiscAlg on Disc(f,x1), Variables\{x1}, >
         else if deg(f,x1) = 2 mod 4
            if Disc(f,x1)(q) ≥ 0
               return false
            else
               call DiscAlg on Disc(f,x1), Variables\{x1}, <
```

The algorithm is doing the following: First, we test the sign of f at some point $p \in \mathbb{R}^k$. If $f(p) \leq 0$, then return false. If $f(p) > 0$, then either $f(x_1, ..., x_k) > 0$ everywhere or f has some real root. If any variable $x_1, \ldots, x_k$ has odd degree, then f has a real root by Proposition 1. So, assume that all variables have even degrees. Now, take the discriminant of f w.r.t. x_1 and test its sign at some point $q \in \mathbb{R}^{k-1}$. Denote this by $\text{Disc}(f, x_1)(q)$. If the degree of x_1 in f is 0 mod 4, then if $\text{Disc}(f, x_1)(q) \leq 0$, then f has real roots by Lemma 1, (and so we return false). Otherwise, if $\text{Disc}(f, x_1)(q) > 0$, test whether $\forall x_2 ... \forall x_k \text{Disc}(f, x_1) > 0$, *which has only* $k-1$ *variables*, in the same way as above. If instead the degree of x_1 in f is 2 mod 4, then if $\text{Disc}(f, x_1)(q) \geq 0$, then f has real roots by Lemma 1 (and we return false). Otherwise, test $\forall x_2 ... \forall x_k \text{Disc}(f, x_1) < 0$ in the same way as above. We continue this recursively.

Note that it is very important that the leading coefficient of x_1 is nonzero when calculating the discriminant. So, when we test $\text{Disc}(f, x_1)(q)$, we must also check that the leading coefficient of f evaluated at q is nonzero.

We have implemented the algorithm in Mathematica. An implementation detail that could be improved in future work is that it always picks points $p = (1, \ldots, 1)$ and $q = (1, \ldots, 1)$.

3.1.1 Comments on Variable Ordering

As an interesting note, because calculating discriminants is the rate-limiting factor (discriminants can get quite complicated!), we suspect that the variable order may have significant effect on how quickly the algorithm runs. For example, if we test the algorithm on $12z^{10} - y^6x^2z + 5y^2 + 1000 +$

$x^{150}y^5$, then the algorithm gets lucky, because Mathematica chooses the variable ordering $\{y, x, z\}$ and Mathematica can calculate the discriminant of $12z^{10} - y^6x^2z + 5y^2 + 1000 + x^{150}y^5$ with respect to y very quickly. So, on this example, the heuristic correctly returns "false" in less than a second. However, calculating the discriminant of $12z^{10} - y^6x^2z + 5y^2 + 1000 + x^{150}y^5$ with respect to x is a lengthy process.

To some extent, changing the method by which Mathematica evaluates discriminants can help. Mathematica provides four different options for the method (aside from the default option, "Automatic"): "SylvesterMatrix", "BezoutMatrix", "Subresultants", and "Modular". If we calculate the discriminant of $12z^{10} - y^6x^2z + 5y^2 + 1000 + x^{150}y^5$ with respect to x using either the automatic setting or "Modular", then it is very slow. If we calculate instead with "SylvesterMatrix", "BezoutMatrix", or "Subresultants", it is quite fast.

However, variable ordering is still significant. If we calculate the discriminant of $12z^{10} - y^6x^2z + 5y^2 + 1000 + x^{150}y^5 + y^6 + y^7 + y^8 + y^9 + y^{10} + 12y^{11} + 13y^{13} + 150y^{22} + y^{17}$ with respect to z, then all of the settings except for "Modular" take less than a second. However, calculating the discriminant with respect to y takes significantly longer on the settings that it was previous fast on. For example, "Sylvester Matrix" now takes 181.585 seconds. This is likely because we have added many terms in y to our previous polynomial without adding any terms in z, so the Sylvester matrix has much higher dimension when we calculate the discriminant with respect to y than when we calculate the discriminant with respect to z.

In a similar vein, choosing the order of variable elimination in CAD can significantly affect its efficiency, and heuristics have been developed to find reasonable variable orderings (see [6] for a summary of some of these).

3.2 Algorithms based on the Polynomial Propositions

We implement an algorithm for each of Theorems 1, 2, 3, 4, and 5 in the "Polynomial Algorithms" section of our code. We include witness generation when applicable (i.e., for the algorithms that correspond to Theorems 1, 2, 3, 4) and, when possible, for their extensions to the weak universal fragment.

As a limitation, in our algorithms for Theorem 1, instead of searching for $p_2, \ldots, p_n \in \mathbb{R}$ such that all of the leading coefficients of x_i (which are themselves polynomials in $x_1, \ldots, x_{i-1}, x_{i+1}, x_n$) have consistent signs when evaluated at $x_1 = p_2, \ldots, x_{i-1} = p_i, x_{i+1} = p_{i+1}, \ldots, x_n = p_n$, we just test $p_i = 1$ for all i. Similarly, for our algorithm for Theorem 2, instead of searching for $p_2, \ldots, p_n \in \mathbb{R}$ so that the leading coefficients of the x_i have negative signs when evaluated at $x_1 = p_2, \ldots, x_{i-1} = p_i, x_{i+1} = p_{i+1}, \ldots, x_n = p_n$, we test $p_i = 1$ for all i. This could be improved in future work.

4 Quantifier Elimination in Mathematica

4.1 Benchmarks

We do experiments on several different sets of benchmarks. When we contacted Stefan Mitsch to ask for benchmarks for KeYmaera, he sent us a set from [11] in SMT2 format. Using Vale-Enriquez and Brown's new tool TARSKI [14], we parsed these into Mathematica format (thanks to Yong Kiam Tan for managing to install TARSKI on his Linux machine and sending me TARSKI's output). There were over 4000 benchmarks in the original set. Many of these are not parsed by TARSKI because TARSKI does not allow division by variables. Others we remove because they are not in the universal fragment. We end with 328 decision problems in the universal fragment and in Mathematica format. However, all of these benchmarks are true formulas. While they can

test the correctness of our algorithms and whether our algorithms introduce significant overhead, it is advantageous to also use other sets of benchmarks, to see how often our algorithms catch false formulas.

Our second set of benchmarks is from Mulligan et. al. [9]. All of these are false formulas. We use 44 test cases from this set of benchmarks. As these are also in SMT2 format, we again use TARSKI to parse them.

Our third set of benchmarks is one that we generate ourselves. It consists of disjunctions of quasi-randomly generated polynomials with integer coefficients (see the "GenerateTestCases" method). The code to pseudorandomly generate these polynomials in "GenerateTestCases" is modified from code I wrote for a previous research project.

4.2 Experimental Results

The code performs well on all of the KeYmaera 3 benchmarks. It runs very quickly on all of them, correctly returning "1" or "2", so it does not seem to introduce significant overhead. (Note that our code returns "0" to mean that the decision problem is definitely false, "1" to mean the tests were inconclusive, and "2" to mean that the decision problem is definitely true, so in the case of a true formula either "1" or "2" should be returned, and in the case of a false formula either "0" or "1" should be returned.) The longest running time is 5.31918 seconds, and the average is less than 0.11704 seconds. Interestingly, sometimes "BooleanConvert" actually converts a formula into "True" or "False" (when putting the formula into CNF), and so our code returns 2 (meaning "definitely true") on 91 of the 328 examples. Mathematica's "Resolve" runs quickly on many but not all of these benchmarks. On certain ones, it seems to stall.

The code also runs quickly on Mulligan et. al.'s benchmarks [9], although somewhat disappointingly it only catches two of them as being definitely false. The longest one takes 5.28081 seconds, and the average time is less than 0.34576 seconds. Mathematica's "Resolve" also runs quickly on England's benchmarks, except for two of them on which it seems to stall.

One thing that affects the behavior of the code on both sets of benchmarks is that, while our algorithms can handle weak inequalities, they are not designed to handle formulas that include polynomial equalities. If we have a formula that contains the polynomial inequality $f \neq g$, then we replace this by the equivalent $f - g < 0 \vee f - g > 0$, and our heuristics are still fine. However, if we have a formula with an equality $f = g$, we are forced to return 1 (inconclusive). In this case the code will print "Had ==, terminating". This happens on 19 of the 44 England benchmarks and 120 of the 328 KeYmaera 3 benchmarks.

Perhaps the most rewarding set of benchmarks is the third set of benchmarks. This set of benchmarks is not fixed but is pseudorandomly generated each time "Testing Method" is called. The fun part about this is that it is easy to find examples which our heuristics very quickly determine are false and on which Mathematica runs slowly. As one example, calling "Resolve" on $\forall v, w, x, y, z(-9v^2wx^3yz^2 + 9v^3wxy^2z^2 + 3w^3x^2y^2z^3 > 0 \vee 7vw^3y + 10w^3yz + 3v^3x^3y^3z^2 > 0)$ takes a long time, even though a human can look at it and immediately conclude false. Running our algorithm set on this decision problem takes less than .005 seconds and returns the following witness: "y is 0, all other variables are 1". Running "Testing Method" generates many examples like this.

5 Conclusion and Future Directions

The goal of this project was twofold. Firstly, we noticed that Mathematica's quantifier elimination procedure (for decision problems) does not always exploit properties of polynomials that can be quickly checked. For example, a human can immediately deduce that the formula $\forall x \forall y \forall z 12z^{10} +$

$y^6x^2z + 5y^2 + 1000 + x^{151}y^5 + y^8 > 0$ is false, but running this with "Resolve" in Mathematica takes hours. We wanted to develop a set of algorithms that would exploit properties of polynomials to narrow this gap.

Secondly, we noticed that many of the works that significantly improve CAD, such as partial CAD [3], still require computing all projection factors. In certain cases, algebraic properties of the projection factors can provide us with information about properties of the original polynomials (such as whether the original polynomials have roots). This is apparent in Corollary 1 and Corollary 4, for example. We hoped to develop algorithms that would allow CAD to terminate without calculating all projection factors. Aside from this being an interesting problem in its own right, we had hope that this would help to speed up quantifier elimination in Mathematica.

DiscAlg in Section 3.1 makes progress towards the second goal. However, algorithms like DiscAlg do not easily generalize to the full disjunctive fragment of decision problems. Further, resultants and discriminants are mathematically quite complicated, and discerning properties about the original polynomials from properties of discriminants and resultants is not necessarily an easy task. Although we still believe that the early termination of CAD in the projection phase is an interesting goal, we do not know whether such early termination will be widely applicable.

Overall, we found that the set of algorithms for polynomials was compelling, and perhaps a better approach towards making progress on the first goal than algorithms like DiscAlg. Even having just a few algorithms that analyze the structure of the input polynomial can improve Mathematica on a significant number of cases, as we found by repeatedly running "TestingModule". Further, adding in witness generation increases the trustworthiness of Mathematica's results. The current algorithm set suggests that significant comprehensive improvement to Mathematica's decision procedures is possible, and we think that it would be interesting future work to continue to flesh out the algorithm set.

References

[1] Brown, C.: Improved projection for cylindrical algebraic decomposition. Journal of Symbolic Computation **32**(5), 447–465 (2001)

[2] Collins, G.E.: Quantifier elimination for real closed fields by cylindrical algebraic decompostion. In: Brakhage, H. (ed.) Automata Theory and Formal Languages 2nd GI Conference Kaiserslautern, May 20–23, 1975. pp. 134–183. Springer Berlin Heidelberg (1975)

[3] Collins, G.E., Hong, H.: Partial cylindrical algebraic decomposition for quantifier elimination. Journal of Symbolic Computation **12**(3), 299–328 (1991)

[4] Dummit, D.S., Foote, R.M.: Abstract algebra, vol. 3. Wiley Hoboken (2004)

[5] Gelfand, I.M., Kapranov, M., Zelevinsky, A.: Discriminants, resultants, and multidimensional determinants. Springer Science & Business Media (1994)

[6] Huang, Z., England, M., Wilson, D., Davenport, J.H., Paulson, L.C.: A comparison of three heuristics to choose the variable ordering for cylindrical algebraic decomposition. arXiv preprint arXiv:1405.6082 (2014)

[7] Jirstrand, M.: Algebraic methods for inequality constraints in control. Ph.D. thesis, PhD thesis, Linköping University, Department of Electrical Engineering ... (1998)

[8] McCallum, S.: Solving polynomial strict inequalities using cylindrical algebraic decomposition. The Computer Journal **36**(5), 432–438 (1993)

[9] Mulligan, C.B., Bradford, R., Davenport, J.H., England, M., Tonks, Z.: Quantifier elimination for reasoning in economics. arXiv preprint arXiv:1804.10037 (2018)

[10] Passmore, G.: Combined Decision Procedures for Nonlinear Arithmetics, Real and Complex. Ph.D. thesis, University of Edinburgh (2011)

[11] Platzer, A., Quesel, J.D., Rümmer, P.: Real world verification. In: International Conference on Automated Deduction. pp. 485–501. Springer (2009)

[12] Strzeboski, A.: Solving systems of strict polynomial inequalities. J. Symb. Comput. **29**(3), 471–480 (2000)

[13] Tarski, A.: A Decision Method for Elementary Algebra and Geometry. Univ. of California Press, Berkeley, second edn. (1951)

[14] Vale-Enriquez, F., Brown, C.W.: Polynomial constraints and unsat cores in TARSKI. In: International Congress on Mathematical Software. pp. 466–474. Springer (2018)

[15] Wilson, D.J., Bradford, R.J., Davenport, J.H., England, M.: Cylindrical algebraic subdecompositions. Mathematics in Computer Science **8**(2), 263–288 (2014)