Check for updates

The Complexity of Quantifier Elimination and Cylindrical Algebraic Decomposition

Christopher W. Brown
Department of Computer Science, Stop 9F
United States Naval Academy
Annapolis, MD 21402, USA
wcbrown@usna.edu

James H. Davenport
Department of Computer Science
University of Bath
Bath BA2 7AY, England
J.H.Davenport@bath.ac.uk

ABSTRACT

This paper has two parts. In the first part we give a simple and constructive proof that quantifier elimination in real algebra is doubly exponential, even when there is only one free variable and all polynomials in the quantified input are linear. The general result is not new, but we hope the simple and explicit nature of the proof makes it interesting. The second part of the paper uses the construction of the first part to prove some results on the effects of projection order on CAD construction — roughly that there are CAD construction problems for which one order produces a constant number of cells and another produces a doubly exponential number of cells, and that there are problems for which all orders produce a doubly exponential number of cells. The second of these results implies that there is a true singly vs. doubly exponential gap between the worst-case running times of several modern quantifier elimination algorithms and CAD-based quantifier elimination when the number of quantifier alternations is constant.

Categories and Subject Descriptors

G.4 [**Mathematics of Computing**]: Mathematical Software—*Algorithm design and analysis*

General Terms

Algorithms, Theory

Keywords

cylindrical algebraic decomposition, quantifier elimination

1. INTRODUCTION

In [6], Davenport & Heintz prove that the worst-case running time for a real quantifier elimination algorithm is

$$\Omega\left(2^{2^{(r-2)/5}}\right),$$

where r is the number of variables in the input formula[1]. They do this by giving a family of quantified formulas, defined by a parameter n, with two free and $6n$ quantified variables, for which any equivalent quantifier-free formula requires 2^{2^n} symbols to write down densely. Weispfenning, in [21], had already shown that quantifier elimination is inherently doubly exponential (even for linear inputs), but the Davenport–Heintz construction had the advantage of being elementary and constructive.

Section 3 of this paper presents a construction very similar to the Davenport–Heintz construction and, similarly, uses it to prove that real quantifier elimination is doubly exponential in the worst case. However, our results improve on theirs in the following respects:

1. The quantified formulas of our construction are linear, i.e. for every (in)equality $f\ \sigma\ 0$ ($\sigma \in \{=, \neq, <, >, \leq, \geq\}$), the polynomial f has total degree one. Thus our construction proves that linear quantifier elimination is doubly exponential. The Davenport–Heintz construction requires non-linear polynomials.

2. We do not need to assume a dense representation of polynomials to prove a doubly exponential bound.

3. We get a better bound, namely $2^{2^{(r-1)/3}}$. Moreover, the construction is simpler — it is easier to follow and requires only one free variable.

The second point requires some elaboration. The dense representation of a degree-n polynomial in x is a list of its $n+1$ coefficients, e.g. element l_i is the coefficient of x^{n-i}. This does not necessarily correspond to the data structures used in quantifier elimination programs. Our doubly exponential bound is valid even if we assume that atomic formulas are of the form $f_0^{e_0} \cdot \ldots \cdot f_k^{e_k}\ \sigma\ 0$, where the f_i are sparse representations of polynomials. This corresponds much more closely to the data structure one would expect programs to use.

The remainder of the paper applies the "linear Davenport–Heintz" construction of Section 3 to prove some results on the effect of projection order on Cylindrical Algebraic Decomposition (CAD) construction.

- We show that there are problems for which projection order is "maximally important", meaning that it can make the difference between a constant number of cells or a doubly exponential number of cells.

[1]The version described in detail in the Davenport & Heintz paper gives a slightly worse bound with $(r-2)/6$ in the second exponent rather than $(r-2)/5$, but is much simpler. In the remainder of this paper, it we will always refer to the simpler version of that construction.

Copyright 2007 Association for Computing Machinery. ACM acknowledges that this contribution was authored or co-authored by an employee, contractor or affiliate of the U.S. Government. As such, the Government retains a nonexclusive, royalty-free right to publish or reproduce this article, or to allow others to do so, for Government purposes only.
ISSAC'07, July 29–August 1, 2007, Waterloo, Ontario, Canada.
Copyright 2007 ACM 978-1-59593-743-8/07/0007 ...$5.00.

- Then we show that there are problems for which all projection orders are bad, i.e. that they all result in CADs with a doubly exponential number of cells.

This second result has an interesting implication. There are several modern quantifier elimination algorithms [16, 2, 1] that are doubly exponential only in the number of quantifier alternations (i.e. changes from $\exists$ to $\forall$ or vice versa). We show that CAD-based quantifier elimination has doubly exponential worst case running time even if there are no quantifier alternations. Thus, there is a true singly exponential vs. doubly exponential gap between the running times of these modern QE algorithms and QE by CAD, not just a gap in the running time analyses. To our knowledge this is a new result.

2. HEINTZ'S CONSTRUCTION

In [10], Heintz shows that quantifier elimination over algebraically closed fields is doubly exponential. The basic idea behind his proof is that the language of first-order logic with equality allows one to write a concise expression for the function $f_n(x)$ defined by the following recursion:

$$f_n(x) = f_{n-1}(f_{n-1}(x))$$

where f_0 is given explicitly. If $f_0(x) = x^2$, for example, this recursion produces

$$f_n(x) = x^{2^{2^n}}.$$

A first-order formula with free variables x and y defines the function $y = f(x)$ if, for any value assigned to x, $f(x)$ is the only value for y satisfying the formula. Given formula $\Phi_0(x, y)$ defining some function $f_0(x)$, Heintz's construction builds a formula Φ_n with free variables x_n and y_n that defines $y_n = f_n(x_n)$ via the following recursive rule:

$$\Phi_n(x_n, y_n) := \exists z_n \forall x_{n-1}, y_{n-1} \left[\left[\begin{array}{c} y_{n-1} = y_n \wedge x_{n-1} = z_n \\ \vee \\ y_{n-1} = z_n \wedge x_{n-1} = x_n \end{array} \right] \Rightarrow \Phi_{n-1}(x_{n-1}, y_{n-1}) \right] \quad (1)$$

Essentially, this formula encodes the two relationships $y_n = f_{n-1}(z_n)$ and $z_n = f_{n-1}(x_n)$ which, of course, imply $y_n = f_{n-1}(f_{n-1}(x_n))$. Assuming Φ_0 is quantifier-free, Φ_n contains two free and $3n$ quantified variables, and has bit-length $\Theta(n \lg n)$; $\Theta(n)$ symbols, but since variable symbols require $\lg n$ bits, $\Theta(n \lg n)$ bits. In fact, Heintz described the construction slightly differently, using the scoping rules for quantifiers to reduce the number of variable names to 6. We get his construction by replacing each x_i, y_i and z_i with $x_{i \bmod 2}$, $y_{i \bmod 2}$ and $z_{i \bmod 2}$. In this form, the bit-length of the formula is $\Theta(n)$.

In [10], Heintz used his construction with $\Phi_0 := y_0 = x_0^d$ to produce a formula $\Phi_n(x_n, y_n)$ defining $y_n = x_n^{d^{2^n}}$. He then argued that a quantifier-free equivalent to $\Phi_n(x_n, 1)$, which defines the d^{2^n}th roots of unity, requires at least d^{2^n} symbols when written densely. Davenport and Heintz [6] modeled this complex construction (choosing $d = 4$) over the reals, representing each complex variable with real variables for the real and imaginary parts. Thus, instead of one free complex variable, they have two free real variables, and in place of $3n$ quantified complex variables, they have $6n$ quantified real variables. Their input contains one polynomial of total degree four, and one polynomial of total degree three; the rest are of total degree one. Moreover, two variables actually appear with exponents of three and four. It requires some work in the real setting to relate the number of isolated points in the solution set to the size of the resulting formula, but they prove that any quantifier-free formula defining $2^{2^{n+1}}$ isolated points, written densely, requires at least 2^{2^n} symbols to write.

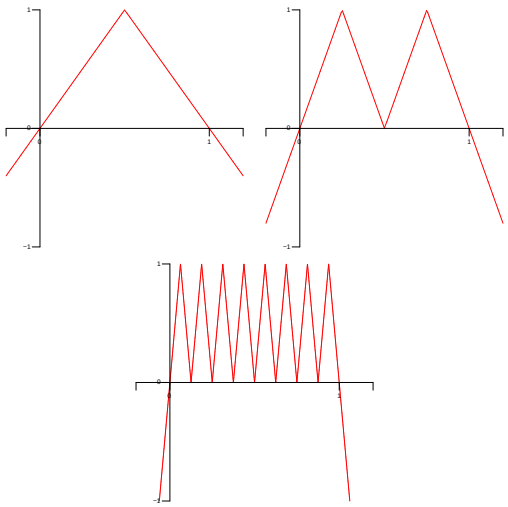

Figure 1: Plot of f_0, as defined by (2), followed by plots of the functions f_1 and f_2 defined by the recursion (1).

In the following section, we give a construction for a family of formulas over the reals based on Heintz's construction, which is similar to the Davenport–Heintz construction, and which also shows real quantifier elimination to be doubly exponential. However, this new construction has several advantages and offers some new insights.

3. A LINEAR CONSTRUCTION FOR THE REALS

We now consider a particular construction for the first-order theory of real algebra. We define our base function f_0 as follows:

$$f_0(x) = \begin{cases} 2x & \text{for } x \le 1/2 \\ 2 - 2x & \text{for } x > 1/2 \end{cases} \quad (2)$$

and consider the functions f_n, $n > 0$, defined from f_0 by the recursive rule (1). Figure 1 gives the graphs of f_0, f_1 and f_2.

THEOREM 1. *Let $f_0^{(k)}$ denote the k-fold composition of f_0 with itself. For all $x \in [0, 1 - 1/2^k]$, $1 - f_0^{(k)}(x) = f_0^{(k)}(x + 1/2^k)$. Note that applying this twice shows $f_0^{(k)}$ is periodic in $[0, 1]$ with period $1/2^{k-1}$.*

PROOF. We proceed by induction. The theorem is easily verified for $k = 1, 2, 3$. Suppose the theorem hold for some $k \ge 3$. Since $f_0^{(3)}$ is periodic with period $1/4$, $f_0^{(k+1)}(x) = f_0^{(k+1)}(x + 1/4)$, so it suffices to prove the $k + 1$ case holds

for $0 \le x \le 1/4$. In this case

$$\begin{aligned} 1 - f_0^{(k+1)}(x) &= 1 - f_0^{(k)}(2x) \\ &= f_0^{(k)}\left(2x + \tfrac{1}{2^k}\right) \\ &= f_0^{(k+1)}\left(x + \tfrac{1}{2^{k+1}}\right). \quad \square \end{aligned}$$

THEOREM 2. *Given f_0 as in (2), define f_n for $n > 0$ by the recursion (1). The graph of f_n in $[0,1]$ is the polyline connecting points $p_0, \ldots, p_{2^{2^n}}$ where $p_k = (k/2^{2^n}, k \bmod 2)$.*

PROOF. Note that $f_n = f_0^{(2^n)}$. For $0 \le x \le 1/2^{2^n}$ we have $f_n(x) = 2^{2^n}x$, since each application of f_0 is on an argument less than or equal to $1/2$. Thus, the graph of f_n over that interval is the segment connecting $(0,0)$ and $(1/2^{2^n}, 1)$. By Theorem 1, $f_n(x) = 1 - f_n(x - 1/2^{2^n})$ in the interval $[1/2^{2^n}, 2/2^{2^n}]$, which means its graph over that interval is the segment connecting $(1/2^{2^n}, 1)$ and $(2/2^{2^n}, 0)$. The theorem then follows from the periodicity of f_n. $\square$

We can define the function f_0 given in (2) with the following formula: $\Phi_0 := x_0 \le 1/2 \wedge y_0 = 2x_0 \vee x_0 > 1/2 \wedge y_0 = 2 - 2x_0$. Let $\Phi_n(x_n, y_n)$ be the formula defined by Heintz's construction with this Φ_0. Note that the "length" of $\Phi_n(x_n, y_n)$ is linear in n under any reasonable definition of "length".

THEOREM 3. *The formula $\Phi_n(x_n, 1/2)$ defines the set*

$$\left\{ \frac{k}{2^{2^n}} + \frac{1}{2^{2^n+1}} \;\middle|\; k \in \mathbb{Z} \text{ and } 0 \le k < 2^{2^n} \right\}.$$

PROOF. $\Phi_n(x_n, 1/2)$ defines the midpoints of the 2^{2^n} segments in the graph of f_n on $[0,1]$. It is clear from the definition of f_0 that $f_n(x) < 0$ outside of $[0,1]$, so no other points are in the set defined by $\Phi_n(x_n, 1/2)$. $\square$

4. QUANTIFIER ELIMINATION IS DOUBLY EXPONENTIAL

Both [10] and [6] argue that quantifier elimination takes doubly exponential time by producing problems whose solutions are doubly exponential in length. We will follow suit, and try to prove bounds on the length of a quantifier-free formula defining $\Phi_n(x_n, 1/2)$, i.e. defining

$$S_n = \left\{ \frac{2k-1}{2^{2^n+1}} \;\middle|\; k \in \mathbb{Z} \wedge 0 < k \le 2^{2^n} \right\}.$$

Notice that the above makes it explicit that if $\Phi_n(N/D, 1/2)$, where N/D is lowest terms, $D = 2^{2^n+1}$.

THEOREM 4. *Let $F(x)$ be a formula in which each atom is of the form $cL_1^{e_1} \cdot L_2^{e_2} \cdot \ldots \cdot L_k^{e_k} \sigma 0$, where the c is a non-zero integer, the e_i's are positive integers, and each L_i is a sparse integer polynomial, i.e. of the form $\sum_{j=1}^{t} c_j x^{d_j}$, $c_j \neq 0$ and $d_1 > d_2 > \cdots > d_t$. If $F(x)$ is equivalent to $\Phi_n(x, 1/2)$, then the bit length of $F(x)$ is at least $2^{2^n}(2^n+1)$.*

PROOF. Clearly, for each integer $k \in [1, 2^{2^n}]$, there must be at least one factor L_i of one atomic formula that is zero at $x = (2k-1)/2^{2^n+1}$. For a given L_i, let $k_1, \ldots, k_r$ define the r elements of S_n at which L_i vanishes. Since $2^{2^n+1}x - (2k_j - 1)$ divides L_i for each k_j, the leading coefficient of L_i is divisible by $\left(2^{2^n+1}\right)^r$, and thus has bit-length at least $r2^n +$ r. Summing over all factors of L_i of all atomic formulas, we will add at least $2^n + 1$ bits for each of the 2^{2^n} points in S_n. Thus, the bit-length of F is at least $2^{2^n}(2^n+1)$. $\square$

Theorem 4 is stronger than a doubly exponential bound on any defining formula for S_n using a sparse polynomial representation, since expanding an expression like $(x_1^2 - 1)(x_2^4 - 1) \cdots (x_k^{2^k} - 1)$ increases its size exponentially. (Note, however, that the e_i's are a bit of a red herring, since there's no need to ever use any exponent other than one or two.) Moreover, inequalities like $f_1 f_2 \cdots f_k < 0$ when expanded to an equivalent boolean combination of inequalities in the f_i's individually can result in an exponential increase in size. Therefore, some quantifier elimination programs — notably Redlog [8] — use this product-of-sparse-polynomials representation.[2] Our doubly exponential bound applies to them, none the less. The next theorem gives a doubly exponential bound for formulas that are allowed to use arbitrary arithmetic expressions to define equalities and inequalities in the output. This, however, means that exponentiation of anything other than variables is not allowed.

THEOREM 5. *Let $F(x)$ be a formula in which each atom is of the form $L\sigma R$, where L and R are expressions involving $*,+,-,(\ )$s, integers, and powers of x. If $F(x)$ is equivalent to $\Phi_n(x, 1/2)$, then the bit length of $F(x)$ is at least $2^{2^n}2^{n-1}$.*

PROOF. Consider the atomic formula $L\sigma R$. Given an expression E, let $l(E)$ be the bit-length of E. Let the expression P be the polynomial $L - R$ written in fully expanded sparse form.

LEMMA 1. *The largest coefficient in P has bit-length at most $2 \cdot l(L\sigma R)$.*

The proof of the previous theorem shows that when each atomic formula A_i is normalized to the form $f_i \sigma 0$, where f is in expanded form, the sum over each A_i of the bit-length of the leading coefficient of f_i is at least $2^{2^n}(2^n+1)$.

Let N be the number of atomic formulas in F. For each $A_i = L\sigma R$, let r_i be the number of elements of S_n at which $L - R$ is zero. Since $2l(A_i) \ge r_i(2^n+1)$, we have:

$$\begin{aligned} \textstyle\sum_{i=1}^{N} 2l(A_i) &\ge 2^{2^n}(2^n+1) \\ 2\textstyle\sum_{i=1}^{N} l(A_i) &\ge 2^{2^n}(2^n+1) \\ l(F) &> 2^{2^n}2^{n-1}. \quad \square \end{aligned}$$

PROOF. (Lemma 1). Let B be the bit-length of the atomic formula $L\sigma R$, i.e. $l(L\sigma R)$. The language allowed for expressing L and R maps directly onto expression trees with leaf nodes that are either integers or powers of x. Interior nodes are binary $*$, $+$ or $-$ operators, or unary $-$ operators. Consider an expression tree for $L - R$. Let $n_1, \ldots, n_r$ be the binary $+$ or $-$ nodes in the tree. For a bit-sequence $s = s_0, \ldots, s_r$, we define the $R(s)$, the restriction of the tree based on s, to be the tree obtained by replacing left subtree of n_i with zero if $s_i = 0$, and the right subtree with zero if $s_i = 1$. Clearly, the tree is equivalent to $\sum_{s \in \{0,1\}^r} R(s)$.

[2]The partially factored representation is useful in many other kinds of algebraic computation as well, since important operations, like resultants, can be done more efficiently in partially factored form than in expanded form.

Since $R(s)$ is just a product, the bit-length of the coefficient of the power-product it defines is at most the bit-length of $R(s)$ which, in turn, is at most the bit length of the atomic formula $L\sigma R$, i.e. B. Since there are 2^r terms in the sum, the largest coefficient in the expanded result is at most $2^r 2^B$. Finally, there can be at most B $+$-operations, so the largest coefficient is at most 2^{2B}, which has bit-length $2B$. □

It would be nice, of course, to be able to prove a doubly exponential bound even when exponentiation of arbitrary expressions is allowed. We have not been able to do this. What about other representations? For example, what if equalities/inequalities of the form $f\sigma 0$ were allowed to be written using "straight-line programs" to represent f? Straight-line programs (SLPs) in this context are sequences of statements of the form $z \leftarrow x$ or $z \leftarrow x \text{ op } y$, where z is a variable, x and y are variables or rational number constants, and op is either addition, subtraction or multiplication (see [13, 14] for more general discussions of SLPs). It is relatively easy to prove that such a "formula" would have size at least 2^n, but we do not know if this bound is tight. We note that for Heintz's complex quantifier elimination problem (whose solution is $x^{2^{2^n}} = 1$), a straight-line program of length $O(2^n)$ does suffice.

THEOREM 6. *There is a straight-line program of length $O(2^n)$ that computes $x^{2^{2^n}} - 1$.*

PROOF. Heintz's construction can be modeled almost as-is with a straight-line program, except that we need multiple copies of each Φ_i, whereas Heintz's construction is based on making a single copy of each Φ_i do "double duty". Let $\Psi_0(x; z)$ be the program $z \leftarrow x * x$, where z is the output variable and x the input. We will give a recursive rule for constructing a program $\Psi_n(x; z)$, where z is the output variable and x the input, that computes $z = x^{2^{2^n}}$.

$$\Psi_n(x; z) := \quad \Psi_{n-1}(x; y), \Psi_{n-1}(y; z)$$

Clearly, the program $\Psi_n(x; z), z \leftarrow z - 1$ computes $x^{2^{2^n}} - 1$. Since the program size doubles at each step, the size of Ψ_n is $O(2^n)$. □

We can follow the same approach with the Davenport–Heintz construction, although we need to simultaneously construct programs for the real and imaginary parts of the complex exponentiations the construction models. Interestingly, we cannot follow this approach with the "linear Davenport–Heintz" construction from the previous section. The reason is that our functions cannot be modeled by straight-line programs because they rely on branching. The point of the construction is that it uses boolean operators and inequalities to define a non-polynomial (albeit piecewise polynomial) function. So while it is possible that there are singly exponential length formulas defining S_n when straight-line programs are used to define inequalities, it is not possible to create such formulas simply by mimicking the construction of the linear Davenport–Heintz quantifier elimination problem.

One might consider looking at the additive complexities of the polynomials in a formula as a metric. The additive complexity of a polynomial p is the minimum number of binary additions/subtractions needed in a straight-line program for p. Risler [17] gives a bound of C^{k^2} on the number of real roots of a univariate polynomial of additive complexity k, where $C \leq 5$. But neither this, nor the improved bound given by Rojas [18] for rational roots gives a better bound on the bit-length of a SLP-based formula than coefficient size or degree arguments.

5. CAD AND PROJECTION ORDER

Cylindrical Algebraic Decomposition (CAD) is the basis of a well-known approach to real quantifier elimination. While its complexity has always appeared to be doubly exponential in the number of variables (free and quantified, irrespective of the number of quantifier alternations), and is proven to be so later in this section, it has small enough constants to be practical on real problems. It has several implementations, including an implementation in Mathematica. It is the only known method that produces simple formulas — e.g. without redundancy or unsatisfiable subformulas — and the CAD data-structure can be used for far more than just quantifier elimination (for instance the Mathematica implementation is used for many things, including numerical integration and assumption facilities).

Variable ordering is an issue for all quantifier elimination algorithms. Those that can eliminate blocks of variables at a time are only concerned with the partial ordering induced by quantifier alternations. CAD, by contrast, requires a total ordering of all variables, free and bound. Essentially, the r problem variables are assigned to the axes of $\mathbb{R}^r$, so that the geometric operation of projection is tied to the elimination of a specific variable. It has long been observed that CAD construction is quite sensitive to variable order, and Dolzmann et al. [7] gave an empirically grounded greedy algorithm for determining a good projection order. In this section, we consider the theoretical limits on the importance of variable order on CAD construction. We will assume a general familiarity with algorithms for CAD construction. The basic results are:

- There is a polynomial p in $3n + 3$ variables such one projection order results in a sign-invariant CAD for p consisting of 3 cells in $\mathbb{R}^{3n+3}$, while another projection order results in a sign-invariant CAD for p consisting of more than 2^{2^n} cells. Thus, the maximum theoretical difference possible resulting from different projection orders is, in fact, achievable.

- There is a set A of $3n^2$ linear polynomials in $3n + 1$ variables such that any CAD in which the elements of A are sign-invariant, regardless of projection order, consists of at least 2^{2^n} cells. In other words, there are some problems for which all orders are bad.

5.1 Problem in which projection order is maximally important

In this section, we present a polynomial p such that choice of projection order plays a maximally important role in the complexity of constructing a CAD for p.

THEOREM 7. *Define the polynomial $p(x)$ as follows:*

$$\left(\begin{array}{l} (y_{n-1}-1/2)^2 \\ + \\ (x_{n-1}-z_n)^2 \end{array} \right) \left(\begin{array}{l} (y_{n-1}-z_n)^2 \\ + \\ (x_{n-1}-x_n)^2 \end{array} \right) x^{n+1}$$

$$+\sum_{i=1}^{n-1} \left(\begin{array}{l} (y_{i-1}-y_i)^2 \\ + \\ (x_{i-1}-z_i)^2 \end{array} \right) \left(\begin{array}{l} (y_{i-1}-z_i)^2 \\ + \\ (x_{i-1}-x_i)^2 \end{array} \right) x^{i+1}$$

$$+ \left(\begin{array}{l} (y_0-2x_0)^2 \\ + \\ (\alpha^2+(x_0-1/2))^2 \end{array} \right) \left(\begin{array}{l} (y_0-2+2x_0)^2 \\ + \\ (\alpha^2-(x_0-1/2))^2 \end{array} \right) x$$

$$+a$$

Consider the variable order

$$a, x_n, z_n, x_{n-1}, y_{n-1}, z_{n-1}, \ldots x_1, y_1, z_1, x_0, \alpha, y_0, x$$

where we eliminate variables from right to left, i.e. x first and a last. Any sign-invariant CAD for p using the above order consists of at least 2^{2^n} cells, while there is a sign-invariant CAD using the reverse order that consists of three cells in $\mathbb{R}^{3n+3}$.

The proof of this theorem is in Section 6. The result does not depend on any assumptions concerning how the CAD is constructed. It depends only on the most fundamental part of the concept of delineability — namely that if a polynomial is delineable over a region, it is either nullified[3] at every point in the region, or at no point in the region. A weaker, though much simpler, result along the same lines comes directly from the linear construction given in Section 3. It can be shown that the set of polynomials in $\Phi_n(x_n, 1/2)$ gives rise to a CAD with doubly exponentially many cells for one projection order, and only singly exponentially many cells for another.

5.2 When all projection orders are bad

In this section, show that there are problems for which all projection orders are "bad", i.e. produce a doubly exponential number of cells, even if there are no quantifier alternations.

THEOREM 8. *Given variables $x_1, \ldots, x_{3n}$, define the following set of polynomials:*

$$P = \bigcup_{i \neq j} \{x_i - x_j, x_i - 2x_j, x_i - (2 - 2x_j), x_j - 1/2\} .$$

Any CAD (even if it is a partial CAD) in which the elements of P are sign-invariant has at least 2^{2^n} cells.

PROOF. This theorem follows trivially from the observation that no matter what order is chosen, all the polynomials from the linear Davenport–Heintz construction for $\Phi_n(x_n, y_n)$ are present in S. Moreover, whichever variable x_j is playing the role of the y_n, $x_j - 1/2$ is in P. Thus, the set of projection factors in the last variable to be projected will contain all the projection factors that result from CAD construction for $\Phi_n(x_n, 1/2)$, which means there will be at least 2^{2^n} cells in $\mathbb{R}^1$. Since there are already 2^{2^n} cells after the first lifting step, the use of partial CAD — which may decide not to lift over certain cells — doesn't affect the bound. □

It was shown in [6] that CAD construction is, in the worst case, doubly exponential in the number of variables *when the projection order is fixed*. The above theorem shows that the result holds even when the projection order is completely unconstrained. To our knowledge, this is new.

In terms of quantifier elimination, this result has an interesting consequence. It shows that, given a quantified formula containing all the elements of P, quantifier elimination by CAD takes time (and space) $\Omega\left(2^{2^n}\right)$, regardless of the number of alternations. However, if there are no alternations then several more recent quantifier elimination algorithms have running times that are singly exponential in n, [16, 1]. Renegar's algorithm, for example, has a running time (bit complexity) of $2^{O(n)}$ on formulas formed from polynomials in the set P with only one quantifier block (assuming the formula has polynomial length in n). To our knowledge, this is the first proof that the singly exponential vs. doubly exponential gap between these more modern QE methods and QE by CAD is a gap in running time, and not just a gap in the precisions of their respective running time analyses.

It has been observed that the problem with using CAD to do quantifier elimination is that it does too much — that it simultaneously solves all QE problems formed from the input polynomials as long as they do not conflict with the variable order used. The above theorem justifies this observation. The "problem" with CAD-based quantifier elimination on an input formula containing the polynomials in P is that, while solving the QE problem it was given, it will "solve" the linear Davenport–Heintz problem that could have been constructed from the same polynomials and variable order. Thus it will do doubly exponential work.

6. PROOF OF THEOREM 7

This section provides a proof of Theorem 7.

PROOF. If a is the first variable projected, then we have an empty projection. thus, p is delineable over $\mathbb{R}^{3n+2}$ using this ordering, and the three cells we get upon lifting into a-space correspond to $p < 0$, $p = 0$ and $p > 0$. In fact, assuming that when choosing a sample point from the interval $(-\infty, +\infty)$ we always choose zero, and when choosing a sample point from the positive (resp. negative) half-line we always choose $+1$ (resp. -1), which any reasonable implementation will do, the sample points we get will be $(\overline{0}, -1)$, $(\overline{0}, 0)$ and $(\overline{0}, +1)$.

However, if we follow the given order, we will show that the definition of delineability requires us to have single-point cells in the induced CAD of $\mathbb{R}^2$ at each of the points $(0, (2k-1)/2^{2^n+1})$, where k is an integer in $[1, 2^{2^n}]$. The idea is that we are forcing ourselves to solve the linear Davenport Heintz problem in order to maintain delineability.

For polynomial p to be delineable, any point in $\mathbb{R}^{3n+2}$ at which p evaluates to a non-zero polynomial in $\mathbb{R}[x]$ must be in a different cell than any point at which p evaluates to the zero polynomial (p is said to be nullified at such points). Note that "p evaluates to the zero polynomial" means that each of its coefficients as a polynomial in x evaluates to zero. These coefficients are essentially the components of the linear Davenport–Heintz construction. Clearly,

$$\left((y_{i-1}-y_i)^2 + (x_{i-1}-z_i)^2\right)\left((y_{i-1}-z_i)^2 + (x_{i-1}-x_i)^2\right)$$

[3] A polynomial $p(x_1, \ldots, x_k)$ is *nullified* at $\alpha \in \mathbb{R}^i$, $i < k$, if $p(\alpha_1, \ldots, \alpha_i, x_{i+1}, \ldots, x_k)$ is the zero polynomial.

is zero if and only if $y_{i-1} = y_i \wedge x_{i-1} = z_i \vee y_{i-1} = z_i \wedge x_{i-1} = x_i$. Similarly, there exists an α such that

$$\left(\begin{array}{l} (y_0 - 2x_0)^2 \\ + \\ (\alpha^2 + (x_0 - 1/2))^2 \end{array} \right) \left(\begin{array}{l} (y_0 - 2 + 2x_0)^2 \\ + \\ (\alpha^2 - (x_0 - 1/2))^2 \end{array} \right)$$

is zero if and only if $y_0 = f_0(x_0)$, where f_0 is defined by (2). So it ought to be clear that the nullification of p is connected to satisfying the linear Davenport–Heintz construction. The remainder of this proof shows that for any CAD in which p is sign-invariant, if the solutions to $\Phi_n(x_n, 1/2)$ are not single-point cells in the induced CAD of $\mathbb{R}^2 \cap a = 0$, then $\Phi_n(x_n, 1/2)$ is satisfied in some open interval, which is a clear contradiction of our previous results.

Looking at Heintz's construction (1), we see that the universal quantifiers are a bit of a trick. For all but two possible values of (x_{n-1}, y_{n-1}),

$$\left[\begin{array}{c} y_{n-1} = y_n \wedge x_{n-1} = z_n \\ \vee \\ y_{n-1} = z_n \wedge x_{n-1} = x_n \end{array} \right] \Rightarrow \Phi_{n-1}(x_{n-1}, y_{n-1})$$

is satisfied trivially, because the left-hand side is false. So the construction may be read as "there exists a z_i such that for both $y_{i-1}, x_{i-1} := y_i, z_i$ and $y_{i-1}, x_{i-1} := z_i, x_i$, $\Phi_{i-1}(x_{i-1}, y_{i-1})$". Viewed this way, we almost have a purely existential formula, and we almost have the concept of a witness when the formula is satisfiable. Normally, a witness is an assignment of values to each existentially quantified variable that satisfies the unquantified part of the formula. We can extend this a bit to get a well-defined notion of witness for this almost existentially quantified formula.

Let λ denote the empty string, and S denote the set of binary strings of length less than n, i.e. $\{ w \in \{0,1\}^* \mid |w| < n\}$.

DEFINITION 1. *The function $W : S \longrightarrow \mathbb{R}$ is a witness to $\Phi_i(a,b)$ if $i = 0$ and $\Phi_0(a,b)$, or $i > 0$ and $W(0s)$ is a witness to $\Phi_{i-1}(W(\lambda), b)$ and $W(1s)$ is a witness to $\Phi_{i-1}(a, W(\lambda))$.*

With the function W, one could verify $\Phi_n(a,b)$. We will finish our proof by constructing witnesses for the formulas $\Phi_n((2k-1)/2^{2^n+1}, 1/2)$, and showing that if our theorem were false, then for some k there would be witnesses for $\Phi_n(\alpha, 1/2)$ for all α close enough to $(2k-1)/2^{2^n+1}$.

Let $\alpha_\lambda = (2k-1)/2^{2^n+1}$, $\beta_\lambda = 1/2$. Note that $\beta_\lambda = f_n(\alpha_\lambda)$. Let $\gamma_\lambda = f_{n-1}(\alpha_\lambda)$. For $s \in S$, given α_s and β_s satisfying $\beta_s = f_{n-|s|}(\alpha_s)$, let $\gamma_s = f_{n-|s|-1}(\alpha_s)$ and define

$$\begin{array}{l} \alpha_{0s} = \gamma_s, \beta_{0s} = \beta_s, \text{ and} \\ \alpha_{1s} = \alpha_s, \beta_{1s} = \gamma_s. \end{array}$$

Then the function $W : S \longrightarrow \mathbb{R}$ given by $W(s) = \gamma_s$ is a witness for $\Phi_n((2k-1)/2^{2^n+1}, 1/2)$.

LEMMA 2. *For all binary strings s, $|s| \leq n$, $\alpha_s \neq \beta_s$. Note that this implies $\gamma_s \neq \alpha_s$ and $\gamma_s \neq \beta_s$. (Proof is given later.)*

Given k and s, let $A_{k,s}$ be the point

$$\begin{array}{l} a = 0, \\ x_n = (2k-1)/2^{2^n+1}, \\ z_n = \gamma_\lambda, \\ x_{n-1} = \alpha_{s_1}, \\ y_{n-1} = \beta_{s_1}, \\ z_{n-1} = \gamma_{s_1}, \\ \vdots \\ x_1 = \alpha_{s_1 \cdots s_{n-1}}, \\ y_1 = \beta_{s_1 \cdots s_{n-1}}, \\ z_1 = \gamma_{s_1 \cdots s_{n-1}}, \\ x_0 = \alpha_{s_1 \cdots s_n}, \\ \alpha = \sqrt{\alpha_{s_1 \cdots s_n} - 1/2} \text{ if real, else } \sqrt{1/2 - \alpha_{s_1 \cdots s_n}} \\ y_0 = \beta_{s_1 \cdots s_n} \end{array}$$

Clearly, p is nullified at $A_{k,s}$, for all k and s. So at $A_{k,s}$ either the factor $\left((y_{i-1} - y_i)^2 + (x_{i-1} - z_i)^2\right)$ is zero or the factor $\left((y_{i-1} - z_i)^2 + (x_{i-1} - x_i)^2\right)$ is zero but, by Lemma 2, never both. By the definition of delineability, p is nullified at every point in the cell containing $A_{k,s}$. As we move around in that cell, if we stay suitably close to $A_{k,s}$ then for each i, $1 < i \leq n$, the same factor of x_i's coefficient vanishes as vanishes for $A_{k,s}$ itself. Suppose $(0, \omega)$, where $\omega = (2k-1)/2^{2^n+1}$, is not in a single point cell in the induced CAD of $\mathbb{R}^2$. Since p is not nullified if $a \neq 0$, 0 is a single-point cell in the induced CAD of $\mathbb{R}^1$. Thus, for any ω' suitably close to ω, for each $A_{k,s}$ there is a point $A'_{k,s}$ in the same cell as $A_{k,s}$ that projects down onto $(0, \omega')$. The z-coordinates of all such $A'_{k,s}$ comprise a witness for $\Phi_n(\omega', 1/2)$. □

PROOF. Lemma 2 The lemma is clearly true if $|s| = 0$. If $\alpha_{0s} = \beta_{0s}$ then $\beta_s = \gamma_s$. However, $\beta_s = f_{n-|s|-1}(\gamma_s)$ by definition, so $\gamma_s = f_{n-|s|-1}(\gamma_s)$. If $\alpha_{1s} = \beta_{1s}$ then $\alpha_s = \gamma_s$. However, $\gamma_s = f_{n-|s|-1}(\alpha_s)$ by definition, so $\gamma_s = f_{n-|s|-1}(\gamma_s)$. Either way, $\gamma_s = f_{n-|s|-1}(\gamma_s)$. It is clear that the fixed points of $f_{n-|s|-1}$ are 0, and a collection of values with denominator not equal to a power of 2. None of the α_i or β_i will be zero or have a denominator that is not a power of 2. □

7. CONCLUSION

In this paper we have presented a new, elementary and explicit proof that the worst case for real quantifier elimination, even in the linear case, is doubly exponential in the number of quantifier alternations. Using the construction behind the proof, we have given a stronger lower bound on the worst case running time for CAD construction, we have shown that projection order in CAD construction can result in a polynomial versus doubly exponential running time gap on the same set of input polynomials, and we have shown that there are sets of input polynomials for which CAD construction is doubly exponential regardless of projection order.

This last result justifies the assertion that "CAD does too much" to be an efficient tool for quantifier elimination. We should point out, however, that this does not mean that we think that CAD is unimportant. First of all there is the issue of whether the asymptotic cross-over points between CAD and more modern QE algorithms actually occur in the range of problems that are even close to accessible with current machines: [11] argues that they are not. Moreover, the real point is that one can and should do more with CAD.

One way this fits into the quantifier elimination problem is by demanding *simple* quantifier-free equivalents. This is something that CAD can do [12, 4, 5] that no other quantifier elimination algorithm can. Additionally, many questions can be answered by CAD directly much more efficiently than by casting the problem as a QE problem and then solving it with CAD. An example of this is determining the dimension of the set of solutions of some input formula, though there are many more examples. The second author's work on simplification of expressions involving elementary functions (see for example [3]) provides an example of an application of CAD that could not be accomplished by QE alone.

As a final remark, it is perhaps worth mentioning that the construction given here can also be used to prove that "generic" quantifier elimination, even in its weakest interpretation, is inherently doubly exponential. Quantifier elimination can be done more quickly if a program is allowed to give less than a complete, correct solution [19, 20, 9, 15]. What if we interpret this to mean that the solution need only be correct up to a measure-zero subset of parameter space — without requiring that we specify where such errors may occur (note that this is weaker than the solutions given by the cited methods). Consider the quantified formula $\exists\,[\Phi_n(x_n, y_n) \wedge y_n \geq 1/2]$, with Φ_n as defined in Section 3. Its solution consists of the $2^{2^n} - 1$ disjoint, closed intervals in $[0, 1]$ of the form

$$\left[(2k+1)/2^{2^{n+1}}, (2k+3)/2^{n+1}\right]$$

which covers exactly half the unit interval. By reasoning similar that in Section 4 a formula of length at least 2^{2^n} is required to represent this set — even generically. This kind of reasoning is, we feel, an advantage of an elementary, explicit proof like that of the doubly-exponential complexity of quantifier elimination given in this paper.

8. REFERENCES

[1] Basu, S. New results on quantifier elimination over real closed fields and applications to constraint databases. *Journal of the ACM 46*, 4 (1999), 537–555.

[2] Basu, S., Pollack, R., and Roy, M.-F. On the combinatorial and algebraic complexity of quantifier elimination. *J. ACM 43*, 6 (1996), 1002–1045.

[3] Beaumont, J., Bradford, R., Davenport, J., and Phisanbut, N. Adherence is better than adjacency. In *Proceedings ISSAC 2005* (2005), M. Kauers, Ed., pp. 37–44.

[4] Brown, C. W. Guaranteed solution formula construction. In *Proc. International Symposium on Symbolic and Algebraic Computation* (1999), pp. 137–144.

[5] Brown, C. W. Simple CAD construction and its applications. *Journal of Symbolic Computation 31*, 5 (May 2001), 521–547.

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

[7] Dolzmann, A., Seidl, A., and Sturm, T. Efficient projection orders for CAD. In *Proceedings of the 2004 International Symposium on Symbolic and Algebraic Computation (ISSAC 2004)* (Santander, Spain, July 2004), J. Gutierrez, Ed., ACM.

[8] Dolzmann, A., and Sturm, T. Redlog: Computer algebra meets computer logic. *ACM SIGSAM Bulletin 31*, 2 (June 1997), 2–9.

[9] Dolzmann, A., Sturm, T., and Weispfenning, V. A new approach for automatic theorem proving in real geometry. *Journal of Automated Reasoning 21*, 3 (1998), 357–380.

[10] Heintz, J. Definability and fast quantifier elimination in algebraically closed fields. *Theoretical Computer Science 24* (1983), 239–277.

[11] Hong, H. Comparison of several decision algorithms for the existential theory of the reals. Tech. Rep. 91-41, Research Institute for Symbolic Computation (RISC-Linz), 1991.

[12] Hong, H. Simple solution formula construction in cylindrical algebraic decomposition based quantifier elimination. In *Proc. International Symposium on Symbolic and Algebraic Computation* (1992), pp. 177–188.

[13] Ibarra, O. H., and Leininger, B. S. The complexity of the equivalence problem for straight-line programs. In *STOC '80: Proceedings of the twelfth annual ACM symposium on Theory of computing* (New York, NY, USA, 1980), ACM Press, pp. 273–280.

[14] Kaltofen, E. Greatest common divisors of polynomials given by straight-line programs. *J. ACM 35*, 1 (1988), 231–264.

[15] Lazard, D., and Rouillier, F. Solving parametric polynomial systems. Tech. rep., INRIA, October 2004.

[16] Renegar, J. On the computational complexity and geometry of the first-order theory of the reals, parts I-III. *Journal of Symbolic Computation 13* (1992), 255–352.

[17] Risler, J.-J. Additive complexity and zeros of real polynomials. *SIAM J. Comput. 14*, 1 (1985), 178–183.

[18] Rojas, J. M. Additive complexity and roots of polynomials over number fields and p-adic fields. In *ANTS* (2002), pp. 506–516.

[19] Seidl, A., and Sturm, T. A generic projection operator for partial cylindrical algebraic decomposition. In *Proc. International Symposium on Symbolic and Algebraic Computation* (2003), R. Sendra, Ed., pp. 240–247.

[20] Sturm, T. *Real Quantifier Elimination in Geometry.* PhD thesis, Department of Mathematics and Computer Science. University of Passau, Germany, D-94030 Passau, Germany, December 1999.

[21] Weispfenning, V. The complexity of linear problems in fields. *J. Symb. Comput. 5*, 1-2 (1988), 3–27.