New Opportunities for the Formal Proof of Computational Real Geometry? (Extended Abstract)

Erika Ábrahám[a], James H. Davenport[b], Matthew England[c], Gereon Kremer[a] and Zak Tonks[b]

[a] RWTH Aachen University, Germany
[b] University of Bath, UK
[c] Coventry University, UK

Abstract
The purpose of this paper is to explore the question "to what extent could we produce formal, machine-verifiable, proofs in real algebraic geometry?" The question has been asked before but as yet the leading algorithms for answering such questions have not been formalised. We present the thesis that a new algorithm for ascertaining satisfiability of formulae over the reals via Cylindrical Algebraic Coverings [Ábrahám, Davenport, England, Kremer, *Deciding the Consistency of Non-Linear Real Arithmetic Constraints with a Conflict Driver Search Using Cylindrical Algebraic Coverings*, 2020] might provide a trace and outputs that allow the results to be more susceptible to machine verification than those of competing algorithms.

1. Setting

An *algebraic proposition* is one built up from expressions of the form $p_i(x_1, \ldots, x_n) = 0$ (where the p_i are polynomials with integer coefficients) joined together by the logical connectives $\neg$ (not), $\wedge$ (and) and $\vee$ (or). A *semi-algebraic proposition* is the same, except that the building blocks are expressions of the form $p_i(x_1, \ldots, x_n)\sigma 0$ where $\sigma \in \{=, \neq, >, \geq, <, \leq\}$. The language of semi-algebraic propositions is also called the *Tarski language* $\mathcal{L}$.

Notation 1. *Although not strictly in the Tarski language, we will find it convenient (both for expository purposes and in implementation) to describe "the i-th root of p" by* $_i\sqrt[n]{p}$ *(or more generally* $_i\,\mathrm{RootOf}(p, y)$*) to mean the ith real root (counting from $-\infty$) of p, as a polynomial in y.* Thom's Lemma [11] means that these can be converted into statements in $\mathcal{L}$.

Notation 2. *We let $\mathbb{R}^*$ be $\mathbb{R}$ to which we add ϵ to make the ordered ring $\mathbb{R}[\epsilon]$ with $0 < \epsilon <$ any positive real, and then add $\pm\infty$ to the underlying ordered set.* See [14] for the rationale behind these symbols in Virtual Term Substitution.

SC^2 '20: Fifth International Workshop on Satisfiability Checking and Symbolic Computation, July 05, 2020, Paris, France (Virtual)
abraham@cs.rwth-aachen.de (E. Ábrahám); J.H.Davenport@bath.ac.uk (J. H. Davenport); Matthew.England@coventry.ac.uk (M. England); gereon.kremer@cs.rwth-aachen.de (G. Kremer); Z.P.Tonks@bath.ac.uk (Z. Tonks)

Problem 1 (Quantifier Elimination). *Consider a quantified proposition*

$$Q_1 y_1 \dots Q_m y_m F(y_1, \dots, y_m, x_1, \dots, x_n), \tag{1}$$

where $F \in \mathcal{L}$ and $Q_i \in \{\exists, \forall\}$. Does there exist a quantifier-free equivalent semi-algebraic proposition $G(x_1, \dots, x_n)$ and if so, can we compute it?

Notation 3. *The structure of (1) induces a partial order to the variables, which we make total, and take from first to last as $x_1, \dots, x_n, y_1, \dots, y_m$.*

The first algorithm to solve Problem 1 was by Tarski [20], although its complexity meant it was infeasible for implementation. It has since been shown that Problem 1 is doubly-exponential (in $n + m$) in the worst case [6, 12], but more generally it is doubly-exponential in the number of times the sequence of Q_i changes from $\exists$ to $\forall$ or *vice versa* [3].

An important special case of Problem 2 is the following.

Problem 2 (Satisfiability). *Given a fully existentially quantified proposition*

$$\exists x_1 \exists x_2 \cdots \exists x_n F(x_1, \dots, x_n), \tag{2}$$

where $F \in \mathcal{L}$, does there exist a solution? I.e. is this **true** *(***SAT***) or* **false** *(***UNSAT***)?*

*Solving this is equivalent to SMT in "Quantifier-Free Non-linear Real Arithmetic" (*QF_NRA*) in the Satisfiability Modulo Theories (SMT) community.*

By [3] Problem (2) is soluble in time singly-exponential in n, but the authors know of no implementation of this. There are two implemented algorithmic approaches for addressing Problem 2.

Cylindrical Algebraic Decomposition (CAD) was introduced by Collins in [9]. While [18] has a more efficient computation, it may *explicitly* state that the decomposition is not complete. Now, [16] (justified recently in [19]) is both efficient and always complete.

Virtual Term Substitution (VTS) was introduced by Weispfenning in [22], and many developments are gathered in [14, 15]. However, it is currently limited to polynomials of degree at most three.

One notes that *implementations* of either CAD and VTS may have marked differences, even if the general concept relative to the respective algorithm is essentially the same. This is especially true of CAD, where notable examples of divergences from more standard implementations include the Maple `RegularChains` Library which constructs first a CAD in complex space before refinement to one in real space [7]; and SMT-RAT, which acts incrementally on constraints [10].

A subtle variant of Problem 2 is the following.

Problem 3 (Proven Satisfiability). *Given a fully existentially quantified proposition (2), where $F \in \mathcal{L}$, produce a computer-verifiable proof of* **SAT** *or* **UNSAT**.

If the answer is **SAT**, all the major algorithms (certainly those discussed below as well as their hybrids) can actually compute witnesses for x_i, so the computer-verifiable proof of **SAT** is relatively easy. The challenge is the **UNSAT** case. The main body of prior work on this problem is by Cohen and Mahboubi [17, 8]. In the first an attempt was made to formalise CAD but to the best of our knowledge this was not completed. In the latter QE is verified but only with an algorithm which falls into the "effective in name only" category.

Thesis and plan of this article

The authors of the present paper have developed a new algorithm for tackling Problem 2 in [1] which offers computational advantages over CAD. The algorithm is based around the new idea of Cylindrical Algebraic Coverings and so we refer to it here as CAC. The present article is essentially a position paper where we present our thesis, *that the UNSAT results produced by CAC may be far more susceptible to formal proof than those of the traditional tools*, to the formal proof community to garner interest and insight into whether it may be followed. Note that we are not claiming that the CAC algorithm is easy to verify: rather that the result of running CAC on a given example is easier to verify.

2. Cylindrical Algebraic Coverings

We recently presented a new algorithm for determining the satisfiability of conjunctions of non-linear polynomial constraints over the reals [1], which can be used to solve Problem 2. The algorithm is based around the technology of CAD but does not build a decomposition of $\mathbb{R}^n$. Instead, overlapping cells are generated, until we have a covering of the sample space. Sample points are constructed incrementally, either until a satisfying sample is found or sufficient samples have been sampled to conclude unsatisfiability. The choice of samples is guided by both the input constraints and previous conflicts (combinations of constraints and samples found to be unsatisfiable).

The key idea behind our new approach is to start with a partial sample; demonstrate that it cannot be extended to a full sample; and from the reasons for that rule out a larger space around the partial sample, which build up incrementally into a covering of the space. The cells are still arranged in cylinders and have semi-algebraic descriptions and thus we call the data structure produced a Cylindrical Algebraic Covering (CAC). Unlike CAD, which starts with projection to generate algebraic information, the algorithm described in [1] starts with "guessing" a sample point dimension-wise, starting in the lowest dimension and iteratively extending it to higher dimensions. Either we "guessed" right and find a satisfying sample or we face a partial sample that cannot be extended to a full solution and use it to guide the projection (and thus the cell construction).

The generalisation of the unsatisfying sample to a wider interval is based upon CAD technology, i.e. we know the truth of a constraint can only change when we cross the real roots of certain projection polynomials calculated with tools such as coefficients, discriminants and resultants. Intuitively, each sample $s \times s_i$ violating a constraint with polynomial p can be generalised to a cell in a p-sign-invariant CAD. So when all extensions of s have been excluded (the ith dimension is fully covered by excluding intervals) then we project all the covering cells

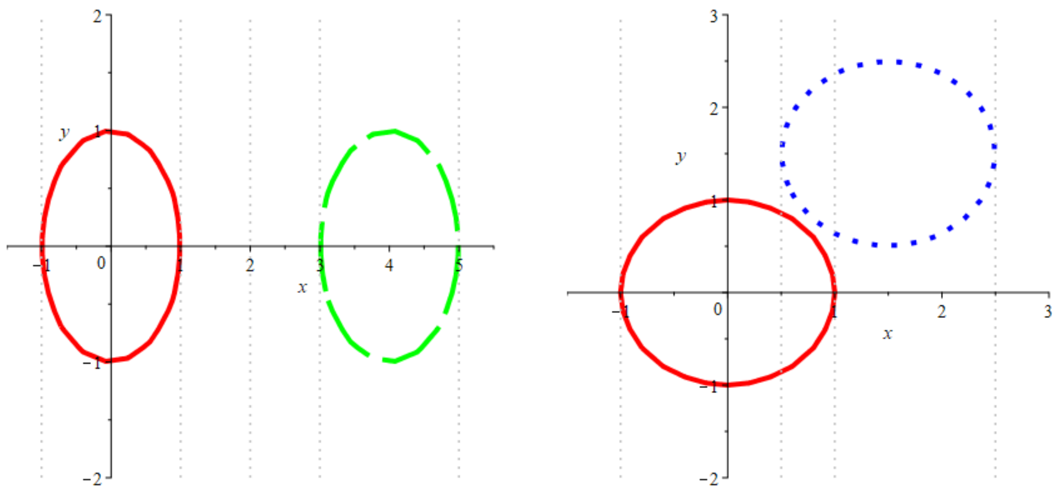

Figure 1: Graphs of polynomials involved in the worked examples.

to dimension $i-1$ and exclude their intersection from further search. The conflict generalisation guides the algorithm to sample away from the reasons of previous conflicts, which should allow for a satisfying sample to be found quicker if one exists. In the UNSAT case: a covering where every cell is UNSAT could use less cells than an entire decomposition.

See [1] for full details of the algorithm including worked examples and details of experimental results. The CAC based algorithm has similarities with the incremental variant of CAD, the NLSAT method of Jovanović and de Moura [13], and the NuCAD algorithm of Brown [4] but the examples in [1] demonstrate its unique advantages. In the present paper we aim to demonstrate another potential advantage: the increased susceptibility to formal verification of its output.

3. Examples

3.1. Example 1

Consider $F := (x^2 + y^2 < 1) \wedge ((x-4)^2 + y^2 < 1)$. The circles are graphed on the left of Figure 1 and we see they do not intersect and thus F must be UNSAT.

3.1.1. CAD:

To solve Problem 2, the CAD algorithm would do the following:

(a) partitions the x-axis at[1] $-1, 1, 2, 3, 5$;
(b) constructs 27 cells and associated sample points in $\mathbb{R}^2$ (see Table 1);
(c) deduces that no sample points have both $x^2+y^2 < 1$ and $(x-4)^2+y^2 < 1$ true at once;

[1] $x = \pm 1$ are the extremal points of the first circle, and $x = 3, 5$ of the second. The $x = 2$ is because the circles (boundaries of the discs) in F have common zeros at $x = 2, y = \pm\sqrt{-3}$: of course these zeros are not real, but have a real x-component.

Table 1
CAD cells for the example in Section 3.1. Here $f_1 = \mathrm{RootOf}(x^2 + y^2 - 1, y)$, $f_2 = \mathrm{RootOf}((x-4)^2 + y^2 - 1, y)$.

x:	< -1	$= -1$	$-1 < x < 1$	$= 1$	$1 < x < 2$	$= 2$	$2 < x < 3$	$= 3$	$3 < x < 5$	$= 5$	> 5
y:	$-$	$< f_1$	$< {}_1f_1$	$< f_1$	$-$	$-$	$-$	$< f_2$	$< {}_1f_2$	$< f_2$	$-$
y:		$= f_1$	$= {}_1f_1$	$= f_1$				$= f_2$	$= {}_1f_2$	$= f_2$	
y:		$> f_1$	${}_1f_1 < y < {}_2f_1$	$> f_1$				$> f_2$	${}_1f_2 < y < {}_2f_2$	$> f_2$	
y:			$= {}_1f_1$						$= {}_2f_2$		
y:			$> {}_2f_1$						$> {}_2f_2$		
#	1	3	5	3	1	1	1	3	5	3	1
#N	1	–	3	–		1		–	3	–	1

(d) and therefore (since the polynomials are sign-invariant in the cells) concludes that no cells have both true anywhere over them;
(e) and because the union of cells is $\mathbb{R}^2$, the statement must be nowhere true.

Step (c) is analogous to verifying **SAT**, and, due to the cylindrical nature of the decomposition, (e) is relatively easy. The problem is verifying (d). Its truth depends on the fine details of the CAD algorithm used.

3.1.2. NuCAD:

In this case NuCAD would construct much the same open cells as CAD, except that it avoids the spurious $x = 2$ problem. These cells are counted as "#N" in Table 1: there are nine of them. The verification proceeds largely as for CAD, except that step (e) is less trivial in general.

3.1.3. VTS:

Virtual Term Substitution would consider a variety of possible values v_i for y. In the implementation of [21], the set of v_i starts with $-\infty$, then various other trivial (i.e. immediately yielding false) values, and the first non-trivial one is ${}_1\,\mathrm{RootOf}\,(y^2 + 1 - x^2, y) + \epsilon$. Virtually substituting this into F yields

$$\underbrace{x^2 < 1}_{\text{guard}} \wedge \underbrace{\text{true}}_{x^2+y^2<1} \wedge \underbrace{\big(-x < -2 \vee \big(x = 2 \wedge x^2 < 1\big)\big)}_{(x-4)^2+y^2<1}, \tag{3}$$

where the guard is there to make sure that the substitution makes sense, and "$-x < -2$" is the simplification of $(x-4)^2 + \Big({}_1\,\mathrm{RootOf}\,(y^2 + 1 - x^2, y) + \epsilon\Big)^2 < 1$. In all there are 41 VTS test points, of which 21 are initial ones used in y, and of the remaining 20 on x, there are 7 distinct ones (used on similar intermediate formulae). They are depicted in a manner intelligible as semi-algebraic sets (to compare with CAD) in Table 2. It is important to note that this is only a certain portrayal, considering VTS does not in itself operate geometrically. We have fewer substitutions of exact values for x (what would be analogous to computing "sections" in CAD).

Speaking generally, VTS is an algebraic approach on formulae as opposed to geometry. Amongst the formulae produced in x, $x = 2$ appears, but other atomic formulae are all strict

Table 2
Structural Test Points for VTS (as semi-algebraic sets) for the example in Section 3.1

$x < -1$	$-1 < x < 1$	$1 < x < 2$	$x = 2$	$2 < x < 3$	$3 < x < 5$	$x > 5$

relations, such that this is the only substitution of an "exact" value. All of the generated test points are substituted in order to deduce **UNSAT**, as we form a disjunction of formulae all equivalent to false.

3.1.4. Human:

Of course no human prover would likely proceed in any of the above ways. A human produced argument may be along the lines of

$$x^2 + y^2 < 1 \Rightarrow x^2 < 1 \Rightarrow x < 1$$
$$(x-4)^2 + y^2 < 1 \Rightarrow (x-4)^2 < 1 \Rightarrow x - 4 \in (-1,1) \Rightarrow x - 4 > -1 \Rightarrow x > 3$$

with the two right most statements clearly incompatible.

3.1.5. CAC:

Cylindrical Algebraic Covering proceeds by choosing a variety of sample x values, then recursing on y (and any subsequent variables if there were any. In this example [1] proceeds as follows, but we note that the theoretical algorithm allows a great deal of choice in computation path.

$x = -1$: This would require $y^2 < 0$ which is unsatisfiable. However, we can deduce nothing about the neighbouring values of x, as -1 is a root of the discriminant of $x^2 + y^2 - 1$.

$x < -1$: We sample $x = -2$ and find this is also impossible for the same reason. The nearest root of this resultant is -1, so $(-\infty, -1)$ is ruled out along with $x = -2$.

$x > -1$: We sample $x = 0$. Here $y \in (-1, 1)$ can not be ruled out by $x^2 + y^2 < 1$ (obviously), but $(x-4)^2 + y^2 < 1$ rules out this value of x immediately, and the generalisation rules out with it the whole of $(-\infty, 3)$.

$x \geq 3$: We sample $x = 4$. This trivially conflicts with $x^2 + y^2 < 1$, which rules out $(1, \infty)$.

Hence the whole of $\mathbb{R}$ is ruled out for x, and we may conclude UNSAT.

While this is not quite as simple as the human proof above, it is much closer to it. If we pruned the reasoning, it would be that $x \in (-\infty, 3)$ is infeasible because of $(x-4)^2 + y^2 < 1$, and $x \in (1, \infty)$ because of $x^2 + y^2 < 1$. This argument of unsatisfiability can be reconstructed from the algorithm output.

3.2. Example 2

Consider $F := (x^2 + y^2 < 1) \wedge \left(\left(x - \frac{3}{2}\right)^2 + \left(y - \frac{3}{2}\right)^2 < 1\right)$. The circles are graphed on the right of Figure 1 and we see they again do not intersect and thus F must be UNSAT.

Table 3
Structural Test Points for VTS for the example in Section 3.2

$x < {}_1\mathrm{RootOf}\left(2x^2+6x-7\right)$	$x = {}_1\mathrm{RootOf}\left(2x^2-6x+7\right)$
${}_1\mathrm{RootOf}\left(2x^2-6x+7\right) < x < 0$	$x = 0$
$0 < x < \frac{1}{2}$	$\frac{1}{2} < x < {}_2\mathrm{RootOf}\left(2x^2-6x+7\right)$
$x = {}_2\mathrm{RootOf}\left(2x^2+6x-7\right)$	${}_2\mathrm{RootOf}\left(2x^2-6x+7\right) < x < \frac{5}{2}$
$x > \frac{5}{2}$	

3.2.1. CAD:

To solve Problem 2, the CAD algorithm behaves similarly to Example 1: the two circles have critical points (roots of the discriminant) at $x = -1, x = 1$ and $x = \frac{1}{2}, x = \frac{5}{2}$ respectively. This time the resultant of the two circles has no real roots. Because the circles overlap the cylinders above the x-axis are more decomposed and we have 41 cells, 13 of which are open. There is no room for the full details here but they can be found in Table 4 of [2] (an extended version of the present paper).

3.2.2. NuCAD:

In this case NuCAD does substantially better than CAD, and in fact creates the same number of cells as in Example 3.1. These cells are counted as "#N" in [2, Table 2], where "New" means that, as we move from $-\infty$ to ∞, that number of new cells are created. The verification proceeds largely as for CAD, except that step (e) is less trivial in general. [5] discusses a concept of "level" for NuCADs, but even with this the process of verifying that our NuCAD covers the whole of $\mathbb{R}^2$ is less trivial for the cylinder above $(\frac{1}{2}, 1)$.

3.2.3. VTS:

VTS this time receives non-false formulae owing to substitutions of test points in y from both circles. Virtual substitution can substitute roots directly from multivariate polynomials.

Table 3 depicts the geometry in x that VTS finds relevant. Again, there are fewer substitutions than CAD, and all test points are substituted to receive **UNSAT**.

3.2.4. CAC:

Our implementation of the Cylindrical Algebraic Covering algorithm operates on this example as follows.

$x = -1$: Similarly to the previous example, this would require $y^2 < 0$ and again we can not deduce anything around this value, as -1 is a root of the discriminant of $x^2 + y^2 - 1$.

$x < -1$: As in the first example we sample $x = -2$ and again find it to be unsatisfiable due to $y^2 < -3$, excluding the whole interval $(-\infty, -1)$.

$-1 < x$: We have now excluded $(-\infty, -1]$ and sample $x = 0$. While $y^2 < 1$ is still satisfiable, the second constraint evaluates to $(y - 3/2)^2 < -\frac{5}{4}$ which is directly conflicting. The depiction excludes the interval $(-\infty, \frac{1}{2})$.

$\frac{1}{2} < x$: We sample $x = 1$ and obtain a direct conflict with $y^2 < 0$. As with the first sample ($x = -1$) we only exclude the point interval $[1, 1]$.

$\frac{1}{2} < x < 1$: To take care of this interval we sample $x = \frac{3}{4}$ which finally needs both constraints to realize that no value for y is feasible. The depiction excludes exactly the interval $(\frac{1}{2}, 1)$. Note that the point $\frac{1}{2}$ remains uncovered.

$x = \frac{1}{2}$: We now check the remaining point $x = \frac{1}{2}$ which directly conflicts with the second constraint.

$1 < x$: We continue with $x = 2$, which is a direct conflict with the first constraint due to $y^2 < -3$, excluding $(1, \infty)$ and thereby completing the covering of the x-axis.

After pruning, the reasoning consists of the following components:

1: $x \notin (-\infty, \frac{1}{2})$ because of the second constraint,

2: $x \neq \frac{1}{2}$ because of the second constraint,

3: $x \notin (\frac{1}{2}, 1)$ because of both constraints,

4: $x \neq 1$ because of the first constraint and

5: $x \notin (1, \infty)$ because of the first constraint.

3.2.5. Human:

There is no trivial human proof this time like in Example 1. If we proceed as there we find conditions on x which are compatible and can only conclude that $x \in (\frac{1}{2}, 1)$. Manipulation of these restrictions into the constraints would find a small range of potential y-axis over that x-interval where both constraints could possibly be satisfied. In other words there is no "quick win". The fact they cannot be satisfied together would require an understanding of the geometry and the completeness of the sample points. There are two obvious proof approaches. Both require to know that at one point, say $x = 1$, the two constraints are not simultaneously satisfied. We use ☡ to indicate points where geometric reasoning would seem to be necessary.

1. The resultant of the two circles ☡ is $18x^2 - 27x + \frac{45}{4}$, whose roots are not real, and *a fortiori* not in $[\frac{1}{2}, 1]$, so the two curves do not cross over $[\frac{1}{2}, 1]$, and hence there are no solutions here.
2. Let ξ be a value in $[\frac{3}{2}, 1]$. Then we need ☡ to have

$$y < {}_2\sqrt{1 - \xi^2}. \tag{4}$$

Similarly we need ☡ to have

$$y > \frac{3}{2} + {}_1\sqrt{1 - (\xi - \frac{3}{2})^2} \geq 0. \tag{5}$$

We can square (5):

$$y^2 > \frac{9}{4} + 3{}_1\sqrt{1 - (\xi - \frac{3}{2})^2} + \left(1 - (\xi - \frac{3}{2})^2\right)$$

since it is an inequality of positive numbers, which also implies we can square (4): $y^2 < 1 - \xi^2$. So

$$1 - \xi^2 > \frac{9}{4} + 3{}_1\sqrt{1 - (\xi - \frac{3}{2})^2} + \left(1 - (\xi - \frac{3}{2})^2\right).$$

Hence

$$0 > \frac{9}{4} + 3{}_1\sqrt{1 - (\xi - \frac{3}{2})^2} + 3\xi - \frac{9}{4} = 3\left(\xi + {}_1\sqrt{1 - (\xi - \frac{3}{2})^2}\right).$$

Then we have

$$-{}_1\sqrt{1 - (\xi - \frac{3}{2})^2} > \xi,$$

again an inequality of positive numbers, and so either $1 - (\xi - \frac{3}{2})^2 > \xi^2$, or $1 - 2\xi^2 + 3\xi - \frac{9}{4} > 0$. This is not true when $\xi = 1$, and the roots of this quadratic (which is essentially the resultant) are not in $[\frac{1}{2}, 1]$ (in fact they are not real). Hence the inequality is nowhere true.

The first approach requires less geometric reasoning, and furthermore that reasoning is essentially uniform — "curves can only cross at roots of the resultant". It is in fact very similar to line 3 of the CAC proof. So here CAC has essentially produced one of the possible human proofs.

4. Conclusion

For general quantifier elimination (Problem 1), we have two standard implemented methods in the literature: CAD and VTS. Both require a completeness result to accept their results, which is currently beyond the reach of formal proof. In other words, not least does one require a proof of correctness of the implementation of a VTS or CAD program, but a proof that the algorithms themselves provide a sufficient selection of substitution points to deduce unsatisfiability.

For the purely existential version (Problem 2), where SAT is easy to verify, but UNSAT is hard, we have a third method: CAC. At least in easy cases, its execution induces proofs much closer to something akin to a human proof. The trace of the algorithm and its output seem to often allow for verifiable results without reliance on a verified completeness result for the entire algorithm. We acknowledge that no verification based on CAC's trace or output has yet been conducted — we publish this paper to highlight the opportunity to the verification community and encourage their input. Is it possible to regard CAC as a tactic that can guide an automatic theorem prover?

References

[1] Ábrahám, E., Davenport, J., England, M., Kremer, G.: Deciding the Consistency of Non-Linear Real Arithmetic Constraints with a Conflict Driven Search Using Cylindrical Algebraic Coverings. http://arxiv.org/abs/2003.05633 (2020)

[2] Ábrahám, E. et al.: New Opportunities for the Formal Proof of Computational Real Geometry? (Extended Version of the present paper.) http://arxiv.org/abs/2004.04034 (2020)

[3] Basu, S.: New results on quantifier elimination over real closed fields and applications to constraint databases. J. ACM **46**, 537–555 (1999)

[4] Brown, C.: Open Non-uniform Cylindrical Algebraic Decompositions. In: Proc. ISSAC 2015. pp. 85–92 (2015)

[5] Brown, C.: Projection and Quantifier Elimination Using Non-uniform Cylindrical Algebraic Decomposition. In: Proc. ISSAC 2017. pp. 53–60 (2017)

[6] Brown, C., Davenport, J.: The Complexity of Quantifier Elimination and Cylindrical Algebraic Decomposition. In: Brown, C. (ed.) Proc. ISSAC 2007. pp. 54–60 (2007)

[7] Chen, C., Moreno Maza, M.: An Incremental Algorithm for Computing Cylindrical Algebraic Decompositions. In: Feng, R., Lee, W.s., Sato, Y. (eds.) Computer Mathematics, pp. 199–221. Springer Berlin Heidelberg (2014).

[8] Cohen, C., Mahboubi, A.: Formal Proofs in Real Algebraic Geometry: From Ordered Fields to Quantifier Elimination. Logical Methods in Computer Science **8**, 1–40 (2012)

[9] Collins, G.: Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition. In: Proc. 2nd. GI Conference Automata Theory & Formal Languages. pp. 134–183 (1975)

[10] Corzilius, F., Loup, U., Junges, S., Ábrahám, E.: SMT-RAT: An SMT-Compliant Nonlinear Real Arithmetic Toolbox. Proc. SAT 2012 pp. 442–448 (2012)

[11] Coste, M., Roy, M.F.: Thom's Lemma, the Coding of Real Algebraic Numbers and the Computation of the Topology of Semi-Algebraic Sets. J. Symbolic Comp. **5**, 121–129 (1988)

[12] Davenport, J., Heintz, J.: Real Quantifier Elimination is Doubly Exponential. J. Symbolic Comp. **5**, 29–35 (1988)

[13] Jovanović, D., de Moura, L.: Solving Non-Linear Arithmetic. In: Proc. IJCAR 2012. pp. 339–354 (2012)

[14] Košta, M.: New concepts for real quantifier elimination by virtual substitution. Ph.D. thesis, Universität des Saarlandes (2016)

[15] Košta, M., Sturm, T., Dolzmann, A.: Better answers to real questions. J. Symbolic Comp. **74**, 255–275 (2016)

[16] Lazard, D.: An Improved Projection Operator for Cylindrical Algebraic Decomposition. In: Bajaj, C. (ed.) Proc. Algebraic Geometry and its Applications. pp. 467–476 (1994)

[17] Mahboubi, A.: Implementing the cylindrical algebraic decomposition within the Coq system. Math. Struct. in Comp. Science **17**, 99–127 (2007)

[18] McCallum, S.: An Improved Projection Operation for Cylindrical Algebraic Decomposition. Ph.D. thesis, University of Wisconsin-Madison Computer Science (1984)

[19] McCallum, S., Parusiński, A., Paunescu, L.: Validity proof of Lazard's method for CAD construction. J. Symbolic Comp. **92**, 52–69 (2019)

[20] Tarski, A.: A Decision Method for Elementary Algebra and Geometry. 2nd ed., Univ. Cal. Press, (1951)

[21] Tonks, Z.: A Poly-algorithmic Quantifier Elimination Package in Maple. In Maple in Mathematics Education and Research 2019 pp. 171–186 (2020)

[22] Weispfenning, V.: The Complexity of Linear Problems in Fields. J. Symbolic Comp. **5**, 3–27 (1988)