Curtains in CAD: Why Are They a Problem and How Do We Fix Them?

Akshar Nair^(✉), James Davenport, and Gregory Sankaran

University of Bath, Claverton Down, Bath BA2 7AY, UK
 {asn42,masjhd,masgks}@bath.ac.uk

Abstract. This paper is part of our ongoing research on the adaptation of Lazard’s CAD to benefit from equational constraints in formulae. In earlier work we combined the CAD methods of McCallum and Lazard so as to produce an efficient algorithm for decomposing a hypersurface rather than the whole of $\mathbb{R}^n$ (exploiting an equational constraint $f = 0$). That method, however, fails if f is nullified (in McCallum’s terminology): we call the set where this happens a curtain. Here we provide a further modification which, at the cost of a trade off in terms of complexity, is valid for any hypersurface, including one containing curtains.

1 Introduction

A Cylindrical Algebraic Decomposition (CAD) is a decomposition of a semi-algebraic set $S \subseteq \mathbb{R}^n$ (for any n) into semi-algebraic sets (also known as cells) homeomorphic to $\mathbb{R}^m$, where $0 \leq m \leq n$, such that the projection of any two cells onto the first k coordinates is either the same or disjoint. We generally want the cells to have some property relative to some given set of input polynomials, often used to form constraints using sign conditions. For example, we might require sign-invariance, i.e. the sign of each input polynomial is constant on each cell, as in the original algorithm of [3].

CAD algorithms have many applications: epidemic modelling [1], artificial intelligence to pass exams [13], financial analysis [10], and many more, so efficient algorithms are of more than theoretical interest.

If S is contained in a subvariety of $\mathbb{R}^n$ it is clearly wasteful to compute a decomposition of $\mathbb{R}^n$. McCallum, in [7], adapts his earlier algorithm [6] to this situation. To explain this idea more precisely, we need some terminology.

Definition 1. A *Quantifier Free Tarski Formula (QFF)* is made up of atoms connected by the standard boolean operators $\wedge, \vee$ and $\neg$. The atoms are statements about signs of polynomials $f \in \mathbb{R}[x_1, \dots, x_n]$, of the form $f * 0$ where $*$ $\in \{=, <, >\}$ (and by combination also $\{\geq, \leq, \neq\}$).

Strictly speaking we need only the relation $<$, but this form is more convenient because of the next definition.

Supported by University of Bath and EPSRC.

© Springer Nature Switzerland AG 2020

A. M. Bigatti et al. (Eds.): ICMS 2020, LNCS 12097, pp. 17–26, 2020.

https://doi.org/10.1007/978-3-030-52200-1_2

Definition 2. [4] An *Equational Constraint (EC)* is a polynomial equation logically implied by a QFF. If it is an atom of the formula, it is said to be *explicit*; if not, then it is *implicit*. If the constraint is visibly an equality one from the formula, i.e. the formula Φ is $f = 0 \wedge \Phi'$, we say the constraint is *syntactically explicit*.

Although implicit and explicit ECs have the same logical status, in practice only the syntactically explicit ECs will be known to us and therefore be available to be exploited.

Example 1. [4] Let f and g be two real polynomials.

1. The formula $f = 0 \wedge g > 0$ has an explicit EC, $f = 0$.
2. The formula $f = 0 \vee g = 0$ has no explicit EC, but the equation $fg = 0$ is an implicit EC.
3. The formula $f^2 + g^2 \leq 0$ also has no explicit EC, but it has two implicit ECs: $f = 0$ and $g = 0$.
4. The formula $f = 0 \vee f^2 + g^2 \leq 0$ logically implies $f = 0$, and the equation is an atom of the formula which makes it an explicit EC according to the definition. However, since this deduction is semantic rather than syntactic, it is more like an implicit EC rather than an explicit EC.

Definition 3. Let A be a set of polynomials in $\mathbb{R}[x_1, \dots, x_n]$ and $P: \mathbb{R}[x_1, \dots, x_n] \times \mathbb{R}^n \rightarrow \Sigma$ a function to some set Σ . If $C \subset \mathbb{R}^n$ is a cell and the value of $P(f, \alpha)$ is independent of $\alpha \in C$ for every $f \in A$, then A is called *P-invariant over that cell*. If this is true for all the cells of a decomposition, we say the decomposition is *P-invariant*.

Much work has been done on sign-invariant CADs, but we first focus our attention on the algorithm for lex-least invariant CADs introduced by Lazard [5] (for a validity proof, see [9]). Unlike the algorithm in [6] it works in the presence of curtains (see Definition 11 below), and has some complexity advantages also. In [12] we adapted [5] to the case of an EC, analogously to the adaptation of [6] in [7], but in doing so we reintroduced the problem of curtains. This paper revisits [12] and provides a hybrid algorithm, which we believe to be the first one that directly gives a CAD of the variety defined by an EC and yet is valid even on curtains. This process avoids going via a CAD of $\mathbb{R}^n$ ($\mathbb{C}^n$ for regular chains [2]).

In Sect. 3 we analyse the reasons why curtains are a problem, and explain how the previous literature has concentrated on valuations rather than curtains themselves. Section 4 consists of the complexity analysis of our algorithm. As in [12], this algorithm cannot be used recursively because the projection operator used in the first stage of projection would output a partial CAD which is a hybrid between sign-invariant and lex-least invariant on curtains of the equational constraint.

2 Lex-Least Valuation and Its Applications in CAD

In order to understand lex-least valuation, let us recall *lexicographic order* $\geq_{\text{lex}}$ on $\mathbb{N}^n$, where $n \geq 1$.

Definition 4. We say that $v = (v_1, \dots, v_n) \geq_{\text{lex}} (w_1, \dots, w_n) = w$ if and only if either $v = w$ or there exists an $i \leq n$ such that $v_i > w_i$ and $v_k = w_k$ for all k in the range $1 \leq k < i$.

Definition 5. [9, Definition 2.4] Let $n \geq 1$ and suppose that $f \in \mathbb{R}[x_1, \dots, x_n]$ is non-zero and $\alpha = (\alpha_1, \dots, \alpha_n) \in \mathbb{R}^n$. The lex-least valuation $\nu_\alpha(f)$ at α is the least (with respect to $\geq_{\text{lex}}$) element $v = (v_1, \dots, v_n) \in \mathbb{N}^n$ such that f expanded about α has the term

$$c(x_1 - \alpha_1)^{v_1} \cdots (x_n - \alpha_n)^{v_n},$$

where $c \neq 0$.

Note that $\nu_\alpha(f) = (0, \dots, 0)$ if and only if $f(\alpha) \neq 0$. The lex-least valuation is referred to as the Lazard valuation in [9].

Example 2. If $n = 1$ and $f(x_1) = x_1^3 - 2x_1^2 + x_1$, then $\nu_0(f) = 1$ and $\nu_1(f) = 2$. If $n = 2$ and $f(x_1, x_2) = x_1(x_2 - 1)^2$, then $\nu_{(0,0)}(f) = (1, 0)$, $\nu_{(2,1)}(f) = (0, 2)$ and $\nu_{(0,1)}(f) = (1, 2)$.

Definition 6. [9] Let $n \geq 2$, and suppose that $f \in \mathbb{R}[x_1, \dots, x_n]$ is non-zero and that $\beta \in \mathbb{R}^{n-1}$. The Lazard residue $f_\beta \in \mathbb{R}[x_n]$ of f at β , and the lex-least valuation $N_\beta(f) = (\nu, \dots, \nu_{n-1})$ of f above β are defined to be the result of Algorithm 1.

Algorithm 1. Lazard residue

Input: $f \in \mathbb{R}[x_1, \dots, x_n]$ and $\beta \in \mathbb{R}^{n-1}$.

Output: Lazard residue f_β and Lex-least valuation of f above β .

```

1:  $f_\beta \leftarrow f$ 
2: for  $i \leftarrow 1$  to  $n - 1$  do
3:    $\nu_i \leftarrow$  greatest integer  $\nu$  such that  $(x_i - \beta_i)^\nu | f_\beta$ .
4:    $f_\beta \leftarrow f_\beta / (x_i - \beta_i)^{\nu_i}$ .
5:    $f_\beta \leftarrow f_\beta(\beta_i, x_{i+1}, \dots, x_n)$ 
6: end for
7: return  $f_\beta, (\nu_1, \dots, \nu_{n-1})$ 

```

Do not confuse the lex-least valuation $N_\beta(f) \in \mathbb{Z}^{n-1}$ of f above $\beta \in \mathbb{R}^{n-1}$ with the lex-least valuation $\nu_\alpha(f) \in \mathbb{Z}^n$ at $\alpha \in \mathbb{R}^n$, defined in Definition 5. Notice that if $\alpha = (\beta, b_n) \in \mathbb{R}^n$ then $\nu_\alpha(f) = (N_\beta(f), \nu_n)$ for some integer ν_n : in other words, $N_\beta(f)$ consists of the first $n - 1$ coordinates of the valuation of f at any point above β .

Definition 7. [9, Definition 2.10] Let $S \subseteq \mathbb{R}^{n-1}$ and $f \in \mathbb{R}[x_1, \dots, x_n]$. We say that f is Lazard delineable on S if:

- i) The lex-least valuation of f above β is the same for each point $\beta \in S$.
- ii) There exist finitely many continuous functions $\theta_i: S \rightarrow \mathbb{R}$, such that $\theta_1 < \dots < \theta_k$ if $k > 0$ and such that for all $\beta \in S$, the set of real roots of f_β is $\{\theta_1(\beta), \dots, \theta_k(\beta)\}$.
- iii) If $k = 0$, then the hypersurface $f = 0$ does not pass over S . If $k \geq 1$, then there exist positive integers $m_1, \dots, m_k$ such that, for all $\beta \in S$ and for all $1 \leq i \leq k$, m_i is the multiplicity of $\theta_i(\beta)$ as a root of f_β .

Definition 8. [9, Definition 2.10] Let f be Lazard delineable on $S \subseteq \mathbb{R}^{n-1}$. Then

- i) The graphs θ_i are called Lazard sections and m_i is the associated multiplicity of these sections.
- ii) The regions between consecutive Lazard sections¹ are called Lazard sectors.

Remark 1. If f is Lazard delineable on S and the valuation of f above any point in S is the zero vector, then the Lazard sections of f are the same as the sections of f defined as in [3] and [7].

Remark 2. We can use Algorithm 1 to compute the lex-least valuation of f at $\alpha \in \mathbb{R}^n$. After the loop is finished, we proceed to the first step of the loop and perform it for $i = n$ and the n -tuple $\nu_1, \dots, \nu_n$ is the required valuation.

Definition 9. [12] Let $A \subset \mathbb{R}[x_1, \dots, x_n]$ be a set of polynomials. Let $E \subseteq A$, and define the projection operator $PL_E(A)$ as follows

$$PL_E(A) = \text{ldcf}(E) \cup \text{trcf}(E) \cup \text{disc}(E) \cup \{\text{res}_{x_n}(f, g) \mid f \in E, g \in A \setminus E\}.$$

Here $\text{ldcf}(E)$ and $\text{trcf}(E)$ are the sets of leading and trailing coefficients of elements of E . The set $\text{disc}(E)$ is the set of all discriminants of E and $\text{res}(E) = \{\text{res}(f, g) : f, g \in E\}$ the set of cross-resultants. We will be comparing this to Lazard's projection operator $PL(A)$ defined in [9]. Note that in the practical use of the operator the set E corresponds to equational constraints.

Theorem 1. [9] Let $f(x, x_n) \in \mathbb{R}[x, x_n]$ have positive degree d in x_n , where $x = (x_1, \dots, x_{n-1})$. Let $D(x)$, $l(x)$ and $t(x)$ denote the discriminant, leading coefficient and trailing coefficient (with respect to x_n) of f , respectively, and suppose that each of these polynomials is non-zero (as an element of $\mathbb{R}[x]$). Let S be a connected analytic subset of $\mathbb{R}^{n-1}$ in which $D(x)$, $l(x)$ and $t(x)$ are all lex-least invariant. Then f is Lazard delineable on S , and hence f is lex-least invariant in every Lazard section and sector over S . Moreover, the same conclusion holds for the polynomial $f^*(x, x_n) = x_n f(x, x_n)$.

¹ Including $\theta_0 = -\infty$ and $\theta_{k+1} = +\infty$.

Theorem 2. [12] Let $n \geq 2$ and let $f, g \in \mathbb{R}[x_1, \dots, x_n]$ be of positive degrees in the main variable x_n . Suppose that f is Lazard delineable on a connected subset $S \subset \mathbb{R}^{n-1}$, in which $R = \text{res}_{x_n}(f, g)$ is lex-least invariant, and f does not have a curtain on S (see Definition 12 below). Then g is sign-invariant in each section of f over S .

Definition 10. Let A be a set of polynomials and D be the lex-least invariant CAD of A . Let C be a cell of D . The valuation of cell C with respect to a polynomial $f \in A$ is the lex-least valuation f at any point $\alpha \in C$.

3 Implications of Curtains on CAD

We propose some geometric terminology to describe the conditions under which a polynomial is nullified in the terminology of [7].

Definition 11. A variety $C \subseteq \mathbb{R}^n$ is called a curtain if, whenever $(x, x_n) \in C$, then $(x, y) \in C$ for all $y \in \mathbb{R}$.

Definition 12. Suppose $f \in \mathbb{R}[x_1, \dots, x_n]$ and $W \subseteq \mathbb{R}^{n-1}$. We say that f has a curtain at W if for all $\underline{x} \in W$ and $y \in \mathbb{R}$ we have $f(\underline{x}, y) = 0$.

Remark 3. Lazard delineability differs from delineability as in [3] and [7] in two important ways. First, we require lex-least invariance on the sections. Second, delineability is not defined on curtains, but Lazard delineability is because the Lazard sections are of f_β rather than f .

Fig. 1. Example illustrating the problems with curtains.

The algorithms in [7] and [8] exploit ECs but both rely on order-invariance. Because order is not defined on curtains, these algorithms fail there. It is therefore natural to try to use Lazard's algorithm, which does not have this issue when used to construct a full CAD of $\mathbb{R}^n$. However, when we try to exploit an EC, a difficulty does still arise if the EC has a curtain (Fig. 1).

Example 3. Let $f = x^2 + y^2 - 1$ (which we assume to be an EC), $g_1 = z - x - 1$ and $g_2 = z - y - 1$ (which we assume are not ECs). Then $\text{res}(f, g_1) = x^2 + y^2 - 1 = \text{res}(f, g_2)$, and this gives us no information about $\text{res}(g_1, g_2)$. In such cases, when the EC has a curtain, it becomes impossible to use PL_E to detect the intersections of the other constraints on that curtain. Because of this, we must resort to a general-purpose projection such as the one in [3], which includes these resultants, when we are on a curtain contained in the hypersurface defined by the EC.

In one common case, where W in Definition 12 is a single point, we can avoid this complication.

Definition 13. *We say that $f \in \mathbb{R}[x_1, \dots, x_n]$ has a point curtain at $\alpha \in \mathbb{R}^{n-1}$ if $N_\alpha(f) \neq (0, \dots, 0)$ and there exists a neighbourhood U of α such that $N_{\alpha'}(f) = (0, \dots, 0)$ for all $\alpha' \in U \setminus \{\alpha\}$.*

In this case we do not need to consider the resultants between the non-equational constraints $A \setminus \{f\}$ when projecting.

Theorem 3. *Let $f \in \mathbb{R}[x_1, \dots, x_n]$ and let $\alpha \in \mathbb{R}^{n-1}$. If f is an equational constraint and has a point curtain at α , then PL_E is sufficient to obtain a sign-invariant CAD.*

Indeed, if f has a curtain on a set S of positive dimension, we need the resultants of the non-equational constraints to determine sample points in $\mathbb{R}^{n-1}$ above which two such constraints meet, as in Example 3. For a point curtain, there is only one sample point in $\mathbb{R}^{n-1}$, namely α . We calculate the roots of Lazard residues of all constraints to determine the sample points on the curtain (which is the fibre above α), and nothing more is needed: the algorithm still produces a lex-least invariant CAD. Further details will appear in [11].

For this to be useful we need to be able to detect and classify curtains (i.e. tell whether or not they are point curtains), Algorithm 2 describes this process.

Remark 4. In steps 5 and 6 of Algorithm 2 we explicitly are looking at the neighbouring cells of the sample points rather than just checking the parity of their indices. This measure is taken to avoid cases where a sample point might seem to be a point curtain, but is actually a part of a larger curtain. This can be seen in Example 3, where the curtains above $(1, 0)$ and $(-1, 0)$ are part of a larger curtain.

Remark 5. Algorithm 3 is called once a CAD is computed using the reduced Lazard projection operator. This algorithm focuses only on decomposing the curtains of the equational constraint.

Algorithm 2. Detecting and Classifying Curtains

 $(B) \leftarrow PC(f, I, S, n)$ **Input** = Set of indices I , set of sample points S with respect to the indices I which correspond to the CAD cells, equational constraint $f \in \mathbb{R}[x_1, \dots, x_n]$.**Output** = B, B' , where B is the set of sample points that are point curtains and B' is the set of sample points that are curtains (but not point curtains).

```

1:  $B \leftarrow$  Empty List
2:  $B' \leftarrow$  Empty List
3: for  $\alpha \in S$  do
4:   if  $\nu_\alpha(f) \neq 0$  then
5:     Check if the nearest 1-cell neighbours have zero valuation.
6:     If all neighbours are zero valuation add  $\alpha$  to  $B$  otherwise add it to  $B'$ 
7:   end if
8: end for
9: return  $(B, B')$ 

```

Algorithm 3. Partial CAD for Curtains

 $(I', S') \leftarrow DCBS(A, I, S, C, n)$ **Input** = Set of indices I , set of sample points S with respect to the indices I , set of polynomials A , C is a list of tuples of sample points that are non-point curtains and their respective indices and n the dimension of our space.**Output** = I', S' list of sample points and their indices for sections of the equational constraints that are curtains.

```

1: If  $n \geq 2$  then go to step 5.
2: for each  $(c, i) \in C$  do
3:   Isolate the real roots of the irreducible factors of the non-zero elements of  $A$ 
   between the neighbours of  $c$ . Construct cell indices  $I'$  and sample points  $S'$  from
   the real roots. Exit.
4: end for
5:  $B \leftarrow$  the square free basis of the primitive parts of all elements of  $A$ .
6:  $P \leftarrow \text{cont}(A) \cup PL(B)$ .
7:  $(I'', S'') \leftarrow DCBS(P, I, S, C, n - 1)$ .
8:  $(I', S') \leftarrow (\text{empty list}, \text{empty list})$ .
9: for each  $\alpha \in S''$  do
10:   $f^* \leftarrow \{f_\alpha \mid f \in B\}$ .
11:  for each  $(c, i) \in C$  do
12:    Isolate the real roots of all the polynomials in  $f^*$  between the neighbours of
     $c$  (looking at the  $n$ th coordinate).
13:    Construct cell indices and sample points for Lazard sections and sectors of
    elements of  $B$  from  $i$ ,  $\alpha$  and the isolated real roots of  $f^*$ .
14:    Add the sample points to  $I'$  and  $S'$ .
15:  end for
16: return  $(I', S')$ .
17: end for

```

Example 4. Suppose Fig. 2 describes the sample points obtained using the CAD algorithm with the reduced Lazard projection operator. In this example, sample points $S_{3,3,1}$, $S_{3,3,2}$ and $S_{3,3,3}$ describe the cells that are part of the curtain of the equational constraint. The sample points circled in blue represent the sample points that lead to $S_{3,3,1}$, $S_{3,3,2}$ and $S_{3,3,3}$ during lifting. The following steps describe Algorithm 3. First, we project all non-equational constraints using Lazard’s original projection operator. When computing sample points, we only consider those which lie within the neighbours of the curtain sample point. For example in our case, at level x_1 , we calculate all sample points between S_2 and S_4 .

Since we are only decomposing the curtain part of the equational constraint, we only need to look at the cell described by S_3 . This results in computing the roots between S_2 and S_4 (and their respective sample points as well). We lift these sample points to level x_2 . When computing at level x_2 , we consider the roots (and their respective sample points) between $S_{3,2}$ and $S_{3,4}$, by the same argument. If at any level the index of the sample point is even, we do not need to calculate roots, but consider that value as our sample point for lifting. Once we lift to x_3 , we have successfully decomposed our curtain.

Fig. 2. Example of root finding in the lifting modification

4 Complexity Analysis of Curtain Solving Method

In this section we look at the complexity of the number of cells produced by Algorithm 3. Note that Algorithm 3 works so to speak on top of the algorithm described in [12]. The idea is to do a second decomposition on the sections of the equational constraint that contain curtains.

Theorem 4. *Given a set A of m polynomials in n variables with maximum degree d in any one variable, Algorithm 3 outputs a partial CAD (i.e. a CAD of $\{f = 0\}$ only) with the number of cells being at most*

$$2^{2^n-1}(m(3m+1)^{2^{n-1}-1} + (m-1)m^{2^{n-1}-1})d^{2^n-1}. \quad (1)$$

This complexity is an improvement on the existing method of using [5] in full without exploiting the EC, and unlike [12] and [7] the algorithm is valid on curtains.

Proof. The first step, using a single equational constraint, has the same complexity as [7]. Further to this, if the equational constraint has curtains, we need to re-project the non-equational constraints, thus performing a CAD of $m - 1$ polynomials of maximum total degree d . Combining these two gives the bound in (1).

The following verifies that the modified algorithm is better than [5] in terms of complexity. It has worse complexity than [12] but is valid for equational constraints with curtains.

The complexity of [5] is $2^{2^n-1}(m+1)^{2^n-2}md^{2^n-1}$ so after removing a factor of $2^{2^n-1}md^{2^n-1}$ the claim is that

$$(3m+1)^{2^{n-1}-1} + (m-1)m^{2^{n-1}-2} < (m+1)^{2^n-2}.$$

We may assume $m \geq 2$ and $n \geq 2$ (otherwise there is nothing to do) so

$$\begin{aligned} (3m+1)^{2^{n-1}-1} + (m-1)m^{2^{n-1}-2} &< 3^{2^{n-1}-1}(m+1)^{2^{n-1}-1} + (m+1)^{2^{n-1}-1} \\ &= (3^{2^{n-1}-1} + 1)(m+1)^{2^{n-1}-1} \\ &= (m+1)^{2^n-2} \frac{3^{2^{n-1}-1} + 1}{(m+1)^{2^n-2^{n-1}-1}} \\ &\leq (m+1)^{2^n-2} \frac{3^{2^{n-1}-1} + 1}{3^{2^n-2^{n-1}-1}} \\ &= (m+1)^{2^n-2} (3^{-2^n} + 3^{-2^n+2^{n-1}+1}) \\ &< (m+1)^{2^n-2} \end{aligned}$$

since $3^{-2^n} + 3^{-2^n+2^{n-1}+1} \leq 3^{-4} + 3^{-1} = \frac{28}{81} < 1$.

5 Conclusion and Further Research

Algorithm 3 is the first partial CAD algorithm which is a hybrid between sign-invariant and lex-least invariant CAD algorithms. It allows us to exploit equational constraints unconditionally. The novelty lies in performing a second decomposition of the curtain sections of the equational constraints: the worst case for the complexity analysis is when the entire equational constraint is a curtain. More analysis needs to be done, as better complexity should be achievable, and we hope to do this in the near future. We are currently looking into extending this approach so as to produce as output a partial CAD that has a hybrid of order invariance and lex-least invariance. Note that lex-least invariance over a given region does not imply order invariance, but an order invariant CAD is also lex-least invariant. The desire to produce lex-least invariance in future work is driven by the aim of a recursive result.

Acknowledgements. We are grateful for many conversations with Matthew England, Scott McCallum and Zak Tonks.

References

1. Brown, C.W., El Kahoui, M., Novotni, D., Weber, A.: Algorithmic methods for investigating equilibria in epidemic modeling. *J. Symbolic Comp.* **41**, 1157–1173 (2006)
2. Chen, C., Moreno Maza, M.: Cylindrical algebraic decomposition in the **RegularChains** library. In: Hong, H., Yap, C. (eds.) *ICMS 2014. LNCS*, vol. 8592, pp. 425–433. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44199-2_65
3. Collins, G.E.: Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In: *Proceedings 2nd. GI Conference Automata Theory & Formal Languages*, pp. 134–183 (1975)
4. England, M., Bradford, R.J., Davenport, J.H.: Cylindrical Algebraic Decomposition with equational constraints. In: Davenport, J.H., England, M., Griggio, A., Sturm, T., Tinelli, C. (eds.) *Symbolic Computation and Satisfiability Checking*, vol. 100, pp. 38–71 (2020). *Journal of Symbolic Computation*
5. Lazard, D.: An improved projection operator for cylindrical algebraic decomposition. In: Bajaj, C.L. (ed.) *Proceedings Algebraic Geometry and Its Applications: Collections of Papers from Shreeram S. Abhyankar’s 60th Birthday Conference*, pp. 467–476 (1994)
6. McCallum, S.: An improved projection operation for cylindrical algebraic decomposition. Ph.D. thesis, University of Wisconsin-Madison Computer Science (1984)
7. McCallum, S.: On projection in CAD-based quantifier elimination with equational constraints. In: Dooley, S. (ed.) *Proceedings ISSAC 1999*, pp. 145–149 (1999)
8. McCallum, S.: On propagation of equational constraints in CAD-based quantifier elimination. In: Mourrain, B. (ed.) *Proceedings ISSAC 2001*, pp. 223–230 (2001)
9. McCallum, S., Parusiński, A., Paunescu, L.: Validity proof of Lazard’s method for CAD construction. *J. Symbolic Comp.* **92**, 52–69 (2019)
10. Mulligan, C.B., Davenport, J.H., England, M.: TheoryGuru: a mathematica package to apply quantifier elimination technology to economics. In: Davenport, J.H., Kauers, M., Labahn, G., Urban, J. (eds.) *ICMS 2018. LNCS*, vol. 10931, pp. 369–378. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96418-8_44
11. Nair, A.S.: Exploiting equational constraints to improve the algorithms for computing cylindrical algebraic decompositions. Ph.D. thesis, University of Bath (2021)
12. Nair, A.S., Davenport, J.H., Sankaran, G.K.: On benefits of equality constraints in lex-least invariant CAD (extended abstract). In: *Proceedings SC2 2019*, September 2019
13. Wada, Y., Matsuzaki, T., Terui, A., Arai, N.H.: An automated deduction and its implementation for solving problem of sequence at university entrance examination. In: Greuel, G.-M., Koch, T., Paule, P., Sommese, A. (eds.) *ICMS 2016. LNCS*, vol. 9725, pp. 82–89. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-42432-3_11