The DEWCAD Project: Pushing Back the Doubly Exponential Wall of Cylindrical Algebraic Decomposition

R. Bradford, J.H. Davenport, M. England,
A. Sadeghimanesh, and **A.K. Uncu**

26th International Conference on Applications of Computer Algebra
S13: Algorithmic Combinatorics Special Session

Outline

1. Cylindrical Algebraic Decomposition
   - Combinatorics Applications
2. The DEWCAD Project
   - Logic and Coverings

Outline

1. Cylindrical Algebraic Decomposition
   - Combinatorics Applications
2. The DEWCAD Project
   - Logic and Coverings

Cylindrical Algebraic Decomposition

Cylindrical Algebraic Decomposition (CAD) was first introduced by Collins [1975] as an algorithm to produce:

- A **decomposition** of $\mathbb{R}^n$ is a set of cells C_i such that $\bigcup_i C_i = \mathbb{R}^n$; and $C_i \cap C_j = \emptyset$ if $i \neq j$.
- The cells are **semi-algebraic** meaning they may be described by a finite sequence of polynomial constraints.
- The cells are **cylindrical** meaning the projection of any two cells to a lower coordinate space, *in the variable ordering*, are identical or disjoint. I.e. the cells in $\mathbb{R}^m$ stack up in cylinders over cells from CAD in $\mathbb{R}^{m-1}$; can project via cell description.

Cylindrical Algebraic Decomposition

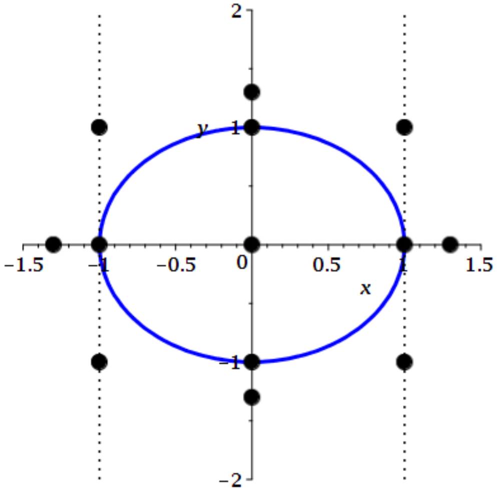
Cylindrical Algebraic Decomposition (CAD) was first introduced by Collins [1975] as an algorithm to produce:

- A **decomposition** of $\mathbb{R}^n$ is a set of cells C_i such that $\bigcup_i C_i = \mathbb{R}^n$; and $C_i \cap C_j = \emptyset$ if $i \neq j$.
- The cells are **semi-algebraic** meaning they may be described by a finite sequence of polynomial constraints.
- The cells are **cylindrical** meaning the projection of any two cells to a lower coordinate space, *in the variable ordering*, are identical or disjoint. I.e. the cells in $\mathbb{R}^m$ stack up in cylinders over cells from CAD in $\mathbb{R}^{m-1}$; can project via cell description.

A CAD is traditionally built relative to a set of input polynomials such that each polynomial has constant sign in each cell: this is called **sign-invariance**. You can uncover properties of polynomials over infinite space by examining finite set of sample points.

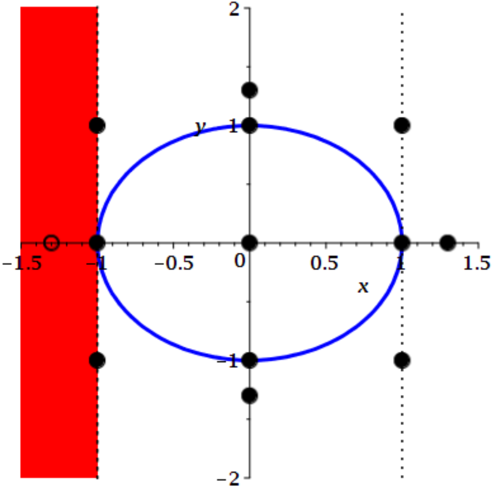
Example: Circle – decomposition visualisation

Cell 1: $x < -1$, y free
Cell 2: $x = -1, y < 0$
Cell 3: $x = -1, y = 0$
Cell 4: $x = -1, y > 0$
Cell 5: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y < 0$
Cell 6: $-1 < x < 1$,
$y^2 + x^2 - 1 = 0, y < 0$
Cell 7: $-1 < x < 1$,
$y^2 + x^2 - 1 < 0$
Cell 8: $-1 < x < 1$,
$y^2 + x^2 - 1 = 0, y > 0$
Cell 9: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y > 0$
Cell 10: $x = 1, y < 0$
Cell 11: $x = 1, y = 0$
Cell 12: $x = 1, y > 0$
Cell 13: $x > 1$, y free



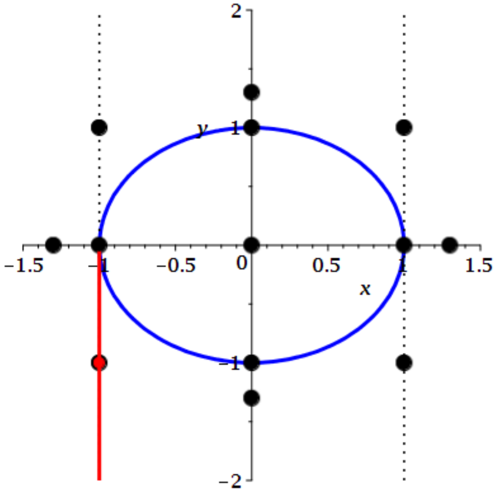
Example: Circle – visualisation

Cell 1: $x < -1,\ y$ free
Cell 2: $x = -1, y < 0$
Cell 3: $x = -1, y = 0$
Cell 4: $x = -1, y > 0$
Cell 5: $-1 < x < 1,$
$y^2 + x^2 - 1 > 0, y < 0$
Cell 6: $-1 < x < 1,$
$y^2 + x^2 - 1 = 0, y < 0$
Cell 7: $-1 < x < 1,$
$y^2 + x^2 - 1 < 0$
Cell 8: $-1 < x < 1,$
$y^2 + x^2 - 1 = 0, y > 0$
Cell 9: $-1 < x < 1,$
$y^2 + x^2 - 1 > 0, y > 0$
Cell 10: $x = 1, y < 0$
Cell 11: $x = 1, y = 0$
Cell 12: $x = 1, y > 0$
Cell 13: $x > 1,\ y$ free



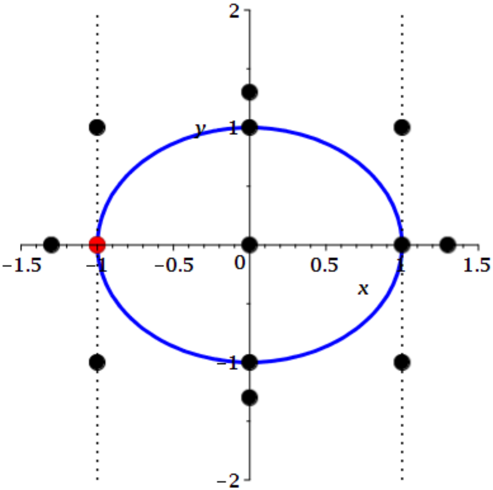
Example: Circle – visualisation

Cell 1: $x < -1$, y free
Cell 2: $x = -1, y < 0$
Cell 3: $x = -1, y = 0$
Cell 4: $x = -1, y > 0$
Cell 5: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y < 0$
Cell 6: $-1 < x < 1$,
$y^2 + x^2 - 1 = 0, y < 0$
Cell 7: $-1 < x < 1$,
$y^2 + x^2 - 1 < 0$
Cell 8: $-1 < x < 1$,
$y^2 + x^2 - 1 = 0, y > 0$
Cell 9: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y > 0$
Cell 10: $x = 1, y < 0$
Cell 11: $x = 1, y = 0$
Cell 12: $x = 1, y > 0$
Cell 13: $x > 1$, y free



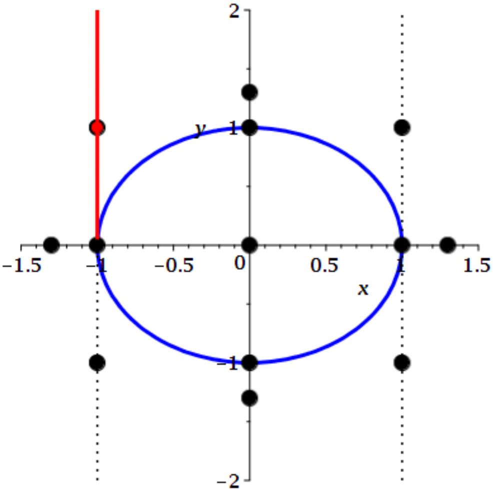
Example: Circle – visualisation

Cell 1: $x < -1$, y free
Cell 2: $x = -1, y < 0$
Cell 3: $x = -1, y = 0$
Cell 4: $x = -1, y > 0$
Cell 5: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y < 0$
Cell 6: $-1 < x < 1$,
$y^2 + x^2 - 1 = 0, y < 0$
Cell 7: $-1 < x < 1$,
$y^2 + x^2 - 1 < 0$
Cell 8: $-1 < x < 1$,
$y^2 + x^2 - 1 = 0, y > 0$
Cell 9: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y > 0$
Cell 10: $x = 1, y < 0$
Cell 11: $x = 1, y = 0$
Cell 12: $x = 1, y > 0$
Cell 13: $x > 1$, y free



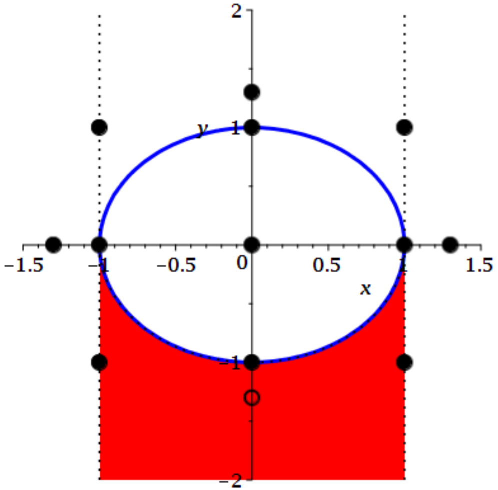
Example: Circle – visualisation

Cell 1: $x < -1$, y free
Cell 2: $x = -1, y < 0$
Cell 3: $x = -1, y = 0$
Cell 4: $x = -1, y > 0$
Cell 5: $-1 < x < 1$, $y^2 + x^2 - 1 > 0, y < 0$
Cell 6: $-1 < x < 1$, $y^2 + x^2 - 1 = 0, y < 0$
Cell 7: $-1 < x < 1$, $y^2 + x^2 - 1 < 0$
Cell 8: $-1 < x < 1$, $y^2 + x^2 - 1 = 0, y > 0$
Cell 9: $-1 < x < 1$, $y^2 + x^2 - 1 > 0, y > 0$
Cell 10: $x = 1, y < 0$
Cell 11: $x = 1, y = 0$
Cell 12: $x = 1, y > 0$
Cell 13: $x > 1$, y free



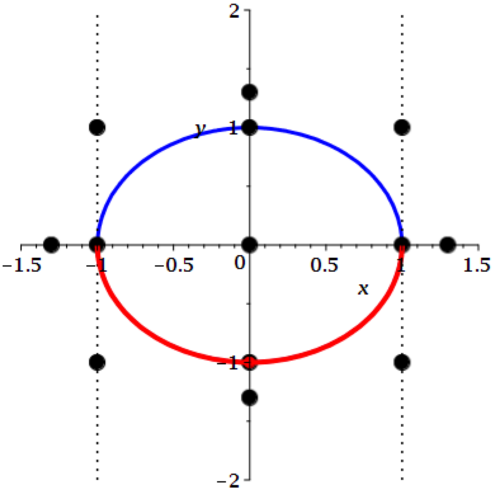
Example: Circle – visualisation

Cell 1: $x < -1$, y free
Cell 2: $x = -1, y < 0$
Cell 3: $x = -1, y = 0$
Cell 4: $x = -1, y > 0$
Cell 5: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y < 0$
Cell 6: $-1 < x < 1$,
$y^2 + x^2 - 1 = 0, y < 0$
Cell 7: $-1 < x < 1$,
$y^2 + x^2 - 1 < 0$
Cell 8: $-1 < x < 1$,
$y^2 + x^2 - 1 = 0, y > 0$
Cell 9: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y > 0$
Cell 10: $x = 1, y < 0$
Cell 11: $x = 1, y = 0$
Cell 12: $x = 1, y > 0$
Cell 13: $x > 1$, y free



Example: Circle – visualisation

Cell 1: $x < -1$, y free
Cell 2: $x = -1, y < 0$
Cell 3: $x = -1, y = 0$
Cell 4: $x = -1, y > 0$
Cell 5: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y < 0$
Cell 6: $-1 < x < 1$,
$y^2 + x^2 - 1 = 0, y < 0$
Cell 7: $-1 < x < 1$,
$y^2 + x^2 - 1 < 0$
Cell 8: $-1 < x < 1$,
$y^2 + x^2 - 1 = 0, y > 0$
Cell 9: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y > 0$
Cell 10: $x = 1, y < 0$
Cell 11: $x = 1, y = 0$
Cell 12: $x = 1, y > 0$
Cell 13: $x > 1$, y free



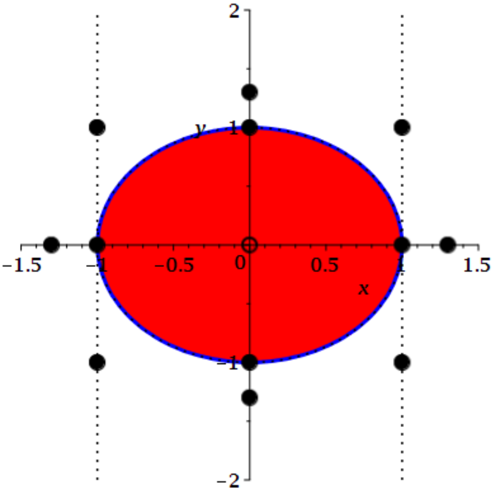
Cylindrical Algebraic Decomposition

Combinatorics Applications

The DEWCAD Project

Example: Circle – visualisation

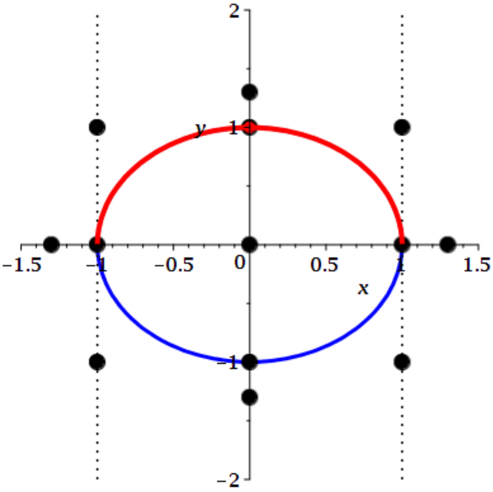
Cell 1: $x < -1$, y free
Cell 2: $x = -1, y < 0$
Cell 3: $x = -1, y = 0$
Cell 4: $x = -1, y > 0$
Cell 5: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y < 0$
Cell 6: $-1 < x < 1$,
$y^2 + x^2 - 1 = 0, y < 0$
Cell 7: $-1 < x < 1$,
$y^2 + x^2 - 1 < 0$
Cell 8: $-1 < x < 1$,
$y^2 + x^2 - 1 = 0, y > 0$
Cell 9: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y > 0$
Cell 10: $x = 1, y < 0$
Cell 11: $x = 1, y = 0$
Cell 12: $x = 1, y > 0$
Cell 13: $x > 1$, y free



A.K. Uncu - aku21@bath.ac.uk

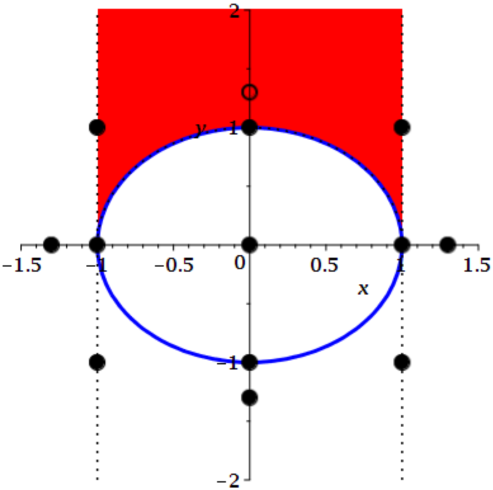
Example: Circle – visualisation

Cell 1: $x < -1$, y free
Cell 2: $x = -1, y < 0$
Cell 3: $x = -1, y = 0$
Cell 4: $x = -1, y > 0$
Cell 5: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y < 0$
Cell 6: $-1 < x < 1$,
$y^2 + x^2 - 1 = 0, y < 0$
Cell 7: $-1 < x < 1$,
$y^2 + x^2 - 1 < 0$
Cell 8: $-1 < x < 1$,
$y^2 + x^2 - 1 = 0, y > 0$
Cell 9: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y > 0$
Cell 10: $x = 1, y < 0$
Cell 11: $x = 1, y = 0$
Cell 12: $x = 1, y > 0$
Cell 13: $x > 1$, y free



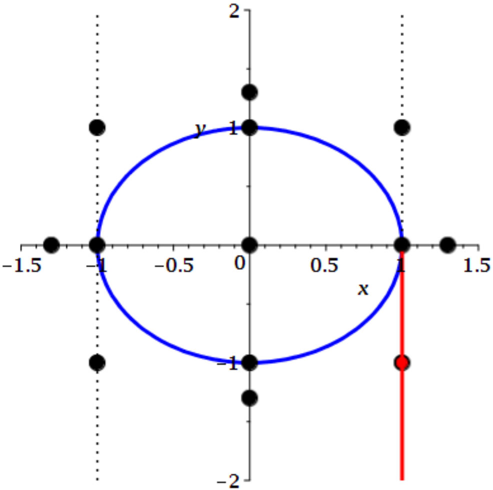
Example: Circle – visualisation

Cell 1: $x < -1$, y free
Cell 2: $x = -1, y < 0$
Cell 3: $x = -1, y = 0$
Cell 4: $x = -1, y > 0$
Cell 5: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y < 0$
Cell 6: $-1 < x < 1$,
$y^2 + x^2 - 1 = 0, y < 0$
Cell 7: $-1 < x < 1$,
$y^2 + x^2 - 1 < 0$
Cell 8: $-1 < x < 1$,
$y^2 + x^2 - 1 = 0, y > 0$
Cell 9: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y > 0$
Cell 10: $x = 1, y < 0$
Cell 11: $x = 1, y = 0$
Cell 12: $x = 1, y > 0$
Cell 13: $x > 1$, y free



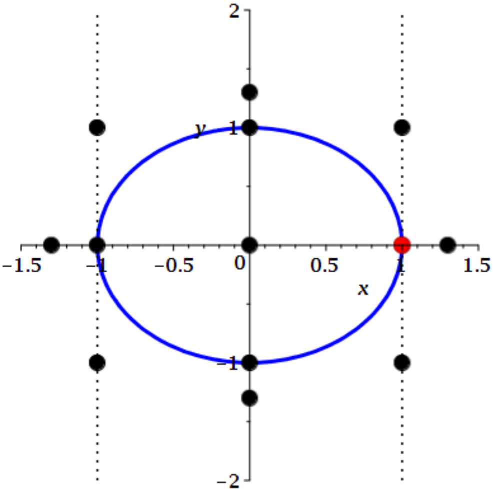
Example: Circle – visualisation

Cell 1: $x < -1$, y free
Cell 2: $x = -1, y < 0$
Cell 3: $x = -1, y = 0$
Cell 4: $x = -1, y > 0$
Cell 5: $-1 < x < 1,$
$y^2 + x^2 - 1 > 0, y < 0$
Cell 6: $-1 < x < 1,$
$y^2 + x^2 - 1 = 0, y < 0$
Cell 7: $-1 < x < 1,$
$y^2 + x^2 - 1 < 0$
Cell 8: $-1 < x < 1,$
$y^2 + x^2 - 1 = 0, y > 0$
Cell 9: $-1 < x < 1,$
$y^2 + x^2 - 1 > 0, y > 0$
Cell 10: $x = 1, y < 0$
Cell 11: $x = 1, y = 0$
Cell 12: $x = 1, y > 0$
Cell 13: $x > 1$, y free



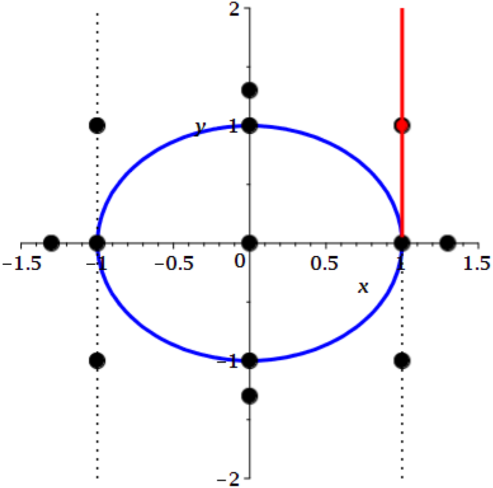
Example: Circle – visualisation

Cell 1: $x < -1$, y free
Cell 2: $x = -1, y < 0$
Cell 3: $x = -1, y = 0$
Cell 4: $x = -1, y > 0$
Cell 5: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y < 0$
Cell 6: $-1 < x < 1$,
$y^2 + x^2 - 1 = 0, y < 0$
Cell 7: $-1 < x < 1$,
$y^2 + x^2 - 1 < 0$
Cell 8: $-1 < x < 1$,
$y^2 + x^2 - 1 = 0, y > 0$
Cell 9: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y > 0$
Cell 10: $x = 1, y < 0$
Cell 11: $x = 1, y = 0$
Cell 12: $x = 1, y > 0$
Cell 13: $x > 1$, y free



Example: Circle – visualisation

Cell 1: $x < -1$, y free
Cell 2: $x = -1, y < 0$
Cell 3: $x = -1, y = 0$
Cell 4: $x = -1, y > 0$
Cell 5: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y < 0$
Cell 6: $-1 < x < 1$,
$y^2 + x^2 - 1 = 0, y < 0$
Cell 7: $-1 < x < 1$,
$y^2 + x^2 - 1 < 0$
Cell 8: $-1 < x < 1$,
$y^2 + x^2 - 1 = 0, y > 0$
Cell 9: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y > 0$
Cell 10: $x = 1, y < 0$
Cell 11: $x = 1, y = 0$
Cell 12: $x = 1, y > 0$
Cell 13: $x > 1$, y free



Example: Circle – visualisation

Cell 1: $x < -1$, y free

Cell 2: $x = -1, y < 0$

Cell 3: $x = -1, y = 0$

Cell 4: $x = -1, y > 0$

Cell 5: $-1 < x < 1$, $y^2 + x^2 - 1 > 0, y < 0$

Cell 6: $-1 < x < 1$, $y^2 + x^2 - 1 = 0, y < 0$

Cell 7: $-1 < x < 1$, $y^2 + x^2 - 1 < 0$

Cell 8: $-1 < x < 1$, $y^2 + x^2 - 1 = 0, y > 0$

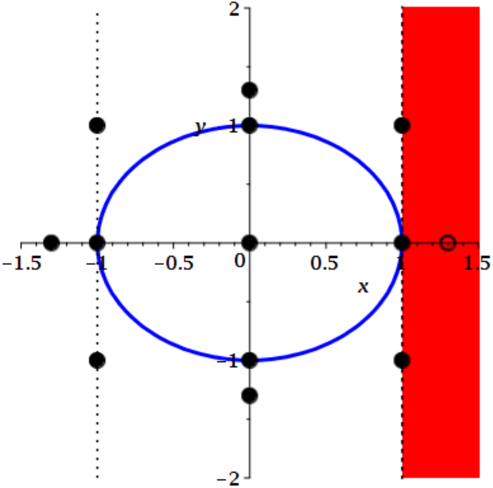
Cell 9: $-1 < x < 1$, $y^2 + x^2 - 1 > 0, y > 0$

Cell 10: $x = 1, y < 0$

Cell 11: $x = 1, y = 0$

Cell 12: $x = 1, y > 0$

Cell 13: $x > 1$, y free



Why Build a CAD?

A CAD can answer many questions about the polynomials involved. Most notably it can be used to perform **Real Quantifier Elimination**: *given a logical formulae whose atoms are non-linear polynomials with integer coefficients; produce a logically equivalent formula without quantifiers.*

- Existential QE via projection of true CAD cells onto the free variables.
- Universal QE via $\forall x F(x) = \neg \exists x \neg F(x)$.

Why Build a CAD?

A CAD can answer many questions about the polynomials involved. Most notably it can be used to perform **Real Quantifier Elimination**: *given a logical formulae whose atoms are non-linear polynomials with integer coefficients; produce a logically equivalent formula without quantifiers.*

- Existential QE via projection of true CAD cells onto the free variables.
- Universal QE via $\forall x F(x) = \neg \exists x \neg F(x)$.

QE can solve problems throughout engineering & science. E.g.

- deriving optimal numerical schemes Erascu and Hong [2014]
- artificial intelligence Arai et al. [2014]
- automated theorem proving Paulson [2012]
- automated loop parellisation Grosslinger et al. [2006]

Why Build a CAD?

A CAD can answer many questions about the polynomials involved. Most notably it can be used to perform **Real Quantifier Elimination**: *given a logical formulae whose atoms are non-linear polynomials with integer coefficients; produce a logically equivalent formula without quantifiers.*

- Existential QE via projection of true CAD cells onto the free variables.
- Universal QE via $\forall x F(x) = \neg \exists x \neg F(x)$.

QE can solve problems throughout engineering & science. E.g.

- deriving optimal numerical schemes Erascu and Hong [2014]
- artificial intelligence Arai et al. [2014]
- automated theorem proving Paulson [2012]
- automated loop parellisation Grosslinger et al. [2006]
- combinatorics!

First Example: # of Monochromatic Schur Triples

A triple (x, y, z) that satisfies $x + y = z$ (and $x \leq y$) is called a Schur Triple. Let $\chi : \mathbb{N} \to \{R, B\}$, then for a Schur triple $(x, y, z) \in \mathbb{N}^3$ is called monochromatic if $\chi(x) = \chi(y) = \chi(z)$.

First Example: # of Monochromatic Schur Triples

A triple (x, y, z) that satisfies $x + y = z$ (and $x \leq y$) is called a Schur Triple. Let $\chi : \mathbb{N} \to \{R, B\}$, then for a Schur triple $(x, y, z) \in \mathbb{N}^3$ is called monochromatic if $\chi(x) = \chi(y) = \chi(z)$.

Let the number of monochromatic Schur triples where all entries are $\leq n$ for a coloring χ be $\mathcal{M}(n, \chi)$.

It was asymptotically shown that $\mathcal{M}(n, \chi)$ is minimized if $\exists\, s$ and t with $s \leq t$ such that

$$\chi(k) := \begin{cases} R, & 1 \leq k \leq s, \\ B, & s+1 \leq k \leq t-s, \\ R, & t+1 \leq k \leq n. \end{cases}$$

First Example: # of Monochromatic Schur Triples

Theorem (Koutschan and Wong [2020])

For $\chi : \{1, \dots, n\} \to R^s B^{t-s} R^{n-t}$, *where* $t \geq 2s$ *and* $s \geq n - t$,

$$\mathcal{M}(n, s, t) := \mathcal{M}(n, \chi) = \frac{s(s-1)}{2} + \frac{(t-2s)(t-2s-1)}{2} + (n-t)(n-t-1).$$

First Example: # of Monochromatic Schur Triples

Theorem (Koutschan and Wong [2020])

For $\chi : \{1, \ldots, n\} \to R^s B^{t-s} R^{n-t}$, *where* $t \geq 2s$ *and* $s \geq n - t$,

$$\mathcal{M}(n, s, t) := \mathcal{M}(n, \chi) = \frac{s(s-1)}{2} + \frac{(t-2s)(t-2s-1)}{2} + (n-t)(n-t-1).$$

Now comes the CAD.

First Example: # of Monochromatic Schur Triples

Theorem (Koutschan and Wong [2020])

For $\chi : \{1, \dots, n\} \to R^s B^{t-s} R^{n-t}$, where $t \geq 2s$ and $s \geq n - t$,

$$\mathcal{M}(n, s, t) := \mathcal{M}(n, \chi) = \frac{s(s-1)}{2} + \frac{(t-2s)(t-2s-1)}{2} + (n-t)(n-t-1).$$

Now comes the CAD.

Theorem (Koutschan and Wong [2020])

For fixed n $\mathcal{M}(n, s, t)$ is minimized when

$$s = \left\lfloor \frac{4n+2}{11} \right\rceil \text{ and } t = \left\lfloor \frac{10n}{11} \right\rceil.$$

First Example: # of Monochromatic Schur Triples

Theorem (Koutschan and Wong [2020])

For $\chi : \{1, \ldots, n\} \to R^s B^{t-s} R^{n-t}$, where $t \geq 2s$ and $s \geq n - t$,

$$\mathcal{M}(n, s, t) := \mathcal{M}(n, \chi) = \frac{s(s-1)}{2} + \frac{(t-2s)(t-2s-1)}{2} + (n-t)(n-t-1).$$

Now comes the CAD.

Theorem (Koutschan and Wong [2020])

For fixed n $\mathcal{M}(n, s, t)$ is minimized when

$$s = \left\lfloor \frac{4n+2}{11} \right\rfloor \text{ and } t = \left\lfloor \frac{10n}{11} \right\rfloor.$$

How?

First Example: # of Monochromatic Schur Triples

By guess-and-prove (experiment-and-CAD):

First Example: # of Monochromatic Schur Triples

By guess-and-prove (experiment-and-CAD): For example, take of the form $n = 11k + 5$ so we have the optimal s and t guesses are $s = 4k + 2$ and $t = 10k + 4$. Hence, we need to prove that for all integers i, j

$$\mathcal{M}(n, s, t) \leq \mathcal{M}(n, s + i, t + j).$$

First Example: # of Monochromatic Schur Triples

By guess-and-prove (experiment-and-CAD): For example, take of the form $n = 11k + 5$ so we have the optimal s and t guesses are $s = 4k + 2$ and $t = 10k + 4$. Hence, we need to prove that for all integers i, j

$$\mathcal{M}(n, s, t) \leq \mathcal{M}(n, s + i, t + j).$$

Recall that there is an exact formula for $\mathcal{M}$, therefore:

$$\mathcal{M}(n, s + i, t + j) = \frac{1}{2}(2 + 5i + 5i^2 - 3j - 4ij + 3j^2 + 12k + 22k^2).$$

So if we can prove that

$$5i + 5i^2 - 3j - 4ij + 3j^2 \geq 0$$

for all $(i, j) \in \mathbb{Z}$, we are done.

First Example: # of Monochromatic Schur Triples

By guess-and-prove (experiment-and-CAD): For example, take of the form $n = 11k + 5$ so we have the optimal s and t guesses are $s = 4k + 2$ and $t = 10k + 4$. Hence, we need to prove that for all integers i, j

$$\mathcal{M}(n, s, t) \leq \mathcal{M}(n, s + i, t + j).$$

Recall that there is an exact formula for $\mathcal{M}$, therefore:

$$\mathcal{M}(n, s + i, t + j) = \frac{1}{2}(2 + 5i + 5i^2 - 3j - 4ij + 3j^2 + 12k + 22k^2).$$

So if we can prove that

$$5i + 5i^2 - 3j - 4ij + 3j^2 \geq 0$$

for all $(i, j) \in \mathbb{Z}$, we are done.

Second Example: A Hypergeometric Inequality

Recall

$$ {}_2F_1\left(\begin{matrix} a, & b \\ & c \end{matrix} : x\right) = \sum_{n\geq 0} \frac{(a)_n (b)_n}{n!(c)_n} x^n, $$

where $(a)_n := a(a+1)\dots(a+n-1)$.

Second Example: A Hypergeometric Inequality

Recall

$$ {}_2F_1\left(\begin{matrix} a, & b \\ & c \end{matrix} : x\right) = \sum_{n\geq 0} \frac{(a)_n (b)_n}{n!(c)_n} x^n, $$

where $(a)_n := a(a+1)\dots(a+n-1)$.

Theorem (Dixit et al. [2016])

For $x > \frac{1}{2}$,

$$ {}_2F_1\left(\begin{matrix} \frac{3}{2}, -m-2 \\ -4m-4 \end{matrix} : 4x\right) - {}_2F_1\left(\begin{matrix} \frac{3}{2}, -m-1 \\ -4m \end{matrix} : 4x\right) > $$
$$ 3\left[{}_2F_1\left(\begin{matrix} \frac{1}{2}, -m-2 \\ -4m-4 \end{matrix} : 4x\right) - {}_2F_1\left(\begin{matrix} \frac{1}{2}, -m-1 \\ -4m \end{matrix} : 4x\right)\right]. $$

Second Example: A Hypergeometric Inequality

$$
\begin{aligned}
&{}_2F_1\left(\begin{matrix}\frac{3}{2}, -m-2\\ -4m-4\end{matrix} : 4x\right) - {}_2F_1\left(\begin{matrix}\frac{3}{2}, -m-1\\ -4m\end{matrix} : 4x\right)\\
&\qquad - 3\left[{}_2F_1\left(\begin{matrix}\frac{1}{2}, -m-2\\ -4m-4\end{matrix} : 4x\right) - {}_2F_1\left(\begin{matrix}\frac{1}{2}, -m-1\\ -4m\end{matrix} : 4x\right)\right]\\
&\qquad = \sum_{k=2}^{m+1}\left(\frac{c(m,k)}{c(m-1,k)} - 1\right) c(m-1,k)x^k + c(m,m+2)x^{m+2},
\end{aligned}
$$

where

$$
\rho := \frac{c(m,k)}{c(m-1,k)} = \frac{(m+2)(-k+4m+1)\dots}{8(m+1)(2m+1)\dots}.
$$

Second Example: A Hypergeometric Inequality

$$
{}_2F_1\left(\begin{matrix}\frac{3}{2}, -m-2\\ -4m-4\end{matrix} : 4x\right) - {}_2F_1\left(\begin{matrix}\frac{3}{2}, -m-1\\ -4m\end{matrix} : 4x\right)
$$

$$
- 3\left[{}_2F_1\left(\begin{matrix}\frac{1}{2}, -m-2\\ -4m-4\end{matrix} : 4x\right) - {}_2F_1\left(\begin{matrix}\frac{1}{2}, -m-1\\ -4m\end{matrix} : 4x\right)\right]
$$

$$
= \sum_{k=2}^{m+1}\left(\frac{c(m,k)}{c(m-1,k)} - 1\right) c(m-1,k)x^k + c(m,m+2)x^{m+2},
$$

where

$$
\rho := \frac{c(m,k)}{c(m-1,k)} = \frac{(m+2)(-k+4m+1)\dots}{8(m+1)(2m+1)\dots}.
$$

We want to know if $\rho \geq 1$ for all m and k (or for which values of m and k...). This is a perfect set up for applying CAD.

Cylindrical Algebraic Decomposition

Combinatorics Applications

The DEWCAD Project

Why Not Build a CAD?

Why Not Build a CAD?

To build a CAD we repeatedly project polynomials to uncover key geometric information. The decomposition is then built by repeated substitution of sample points to render multivariate polynomials univariate which then undergo real root isolation.

By the end of projection you have doubly exponentially many polynomials of doubly exponential degree (in the number of projections, i.e. variables). Hence also the number of real roots, cells and time to compute them grows doubly exponentially! See Davenport and Heintz [1988], Brown and Davenport [2007].

Why Not Build a CAD?

To build a CAD we repeatedly project polynomials to uncover key geometric information. The decomposition is then built by repeated substitution of sample points to render multivariate polynomials univariate which then undergo real root isolation.

By the end of projection you have doubly exponentially many polynomials of doubly exponential degree (in the number of projections, i.e. variables). Hence also the number of real roots, cells and time to compute them grows doubly exponentially! See Davenport and Heintz [1988], Brown and Davenport [2007].

There exist specialized algorithms with better complexity for many cases of many problems. But CAD remains the fall back algorithm to use in general.

Why Not Build a CAD?

CAD is highly dependent on the variable ordering.

For example: Let's say that we would like to build a CAD related to the set of polynomials

$$\left\{ \begin{array}{c} x_1 x_9 - x_2 x_3,\ x_2 x_8 - x_3 x_9,\ x_3 x_7 - x_8 x_9,\ x_6 x_9 - x_8 x_9, \\ x_5 x_8 - x_6 x_7,\ x_7 x_9 - x_5 x_6 \end{array} \right\}.$$

A.K. Uncu - aku21@bath.ac.uk

Cylindrical Algebraic Decomposition

Combinatorics Applications

The DEWCAD Project

Why Not Build a CAD?

CAD is highly dependent on the variable ordering.

For example: Let's say that we would like to build a CAD related to the set of polynomials

$$\left\{ \begin{array}{c} x_1x_9 - x_2x_3,\ x_2x_8 - x_3x_9,\ x_3x_7 - x_8x_9,\ x_6x_9 - x_8x_9, \\ x_5x_8 - x_6x_7,\ x_7x_9 - x_5x_6 \end{array} \right\}.$$

If we do the projections in the order $x_1 > x_2 > x_3 > \cdots > x_9$ the outcome CAD has 512 cells.

Why Not Build a CAD?

CAD is highly dependent on the variable ordering.

For example: Let's say that we would like to build a CAD related to the set of polynomials

$$\left\{ \begin{array}{c} x_1x_9 - x_2x_3,\ x_2x_8 - x_3x_9,\ x_3x_7 - x_8x_9,\ x_6x_9 - x_8x_9, \\ x_5x_8 - x_6x_7,\ x_7x_9 - x_5x_6 \end{array} \right\}.$$

If we do the projections in the order $x_1 > x_2 > x_3 > \cdots > x_9$ the outcome CAD has 512 cells.

If we do it the reverse order the number of cells go over 10000.

Outline

1 Cylindrical Algebraic Decomposition
- Combinatorics Applications

2 The DEWCAD Project
- Logic and Coverings

Summary and DEWCAD

To summarise:

- CAD may be used in theory for a great many problems;
- but doubly exponential complexity limits this in practice.

For a given application class CAD can solve simple problems but as input size increases one inevitably hits the *doubly exponential wall* where further computation is not possible even which substantial computing resource.

Summary and DEWCAD

To summarise:

- CAD may be used in theory for a great many problems;
- but doubly exponential complexity limits this in practice.

For a given application class CAD can solve simple problems but as input size increases one inevitably hits the *doubly exponential wall* where further computation is not possible even which substantial computing resource.

Since its invention in the 1970s numerous researchers have improved and optimised CAD and its sub-algorithms, see e.g. Caviness and Johnson [1998]. Together these have *pushed back* the doubly exponential wall to allow study of many applications.

Our project is titled: Pushing Back the **D**oubly **E**xponential **W**all of **C**ylindrical **A**lgebraic **D**ecomposition (**DEWCAD**).

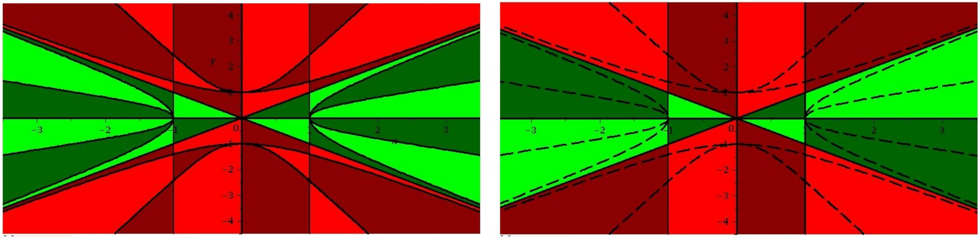
The DEWCAD Project

- EPSRC funded project to produce new mathematics, algorithms, implementations and applications of Cylindrical Algebraic Decomposition.
- Run at Coventry University (grant EP/T015748/1) and the University of Bath (grant EP/T015713/1).
- Started 1st Jan 2021 running for 52 months.
- **Coventry:** Matthew England (investigator), and AmirHossein Sadeghimanesh (postdoc)
- **Bath:** James H. Davenport and Russell Bradford (investigators) and Ali Uncu (postdoc)
- https://matthewengland.coventry.domains/dewcad/

CAD for Logic Problems

Problems like Real QE involve logical formulae whose atoms are polynomials. A sign-invariant CAD for the polynomials in the formulae allows us to derive solutions, but such a CAD could solve **any** logical formulae built from those polynomials.

We can try to adapt CAD to take note of the logic. E.g.:



CAD algorithms like the one which produced the CAD on the right allow for large savings Bradford et al. [2016] but only exist for formulae with certain logical structure.

The SMT Approach

There have been many prominent developments recently on CAD technology for **Satisfiability** problems, i.e. QE where all variables are existentially quantified.

The **Satisfiability Modulo Theories** (SMT) approach to such problems is to separate out the logic from the arithmetic theory.

- Allow the logical structure to be explored by a **SAT Solver**.
- Have the solutions proposed be tested in the theory of interest by relevant software for that domain: a **Theory Solver**.

We could hence use CAD as an SMT Theory Solver Kremer and Ábrahám [2020]. Why is this desirable?

- Take advantage of the dramatic improvements in SAT solvers;
- CAD only studying a subset of the input polynomials at once.

The SMT Approach

There have been many prominent developments recently on CAD technology for **Satisfiability** problems, i.e. QE where all variables are existentially quantified.

The **Satisfiability Modulo Theories** (SMT) approach to such problems is to separate out the logic from the arithmetic theory.

- Allow the logical structure to be explored by a **SAT Solver**.
- Have the solutions proposed be tested in the theory of interest by relevant software for that domain: a **Theory Solver**.

We could hence use CAD as an SMT Theory Solver Kremer and Ábrahám [2020].
Why is this desirable?

- Take advantage of the dramatic improvements in SAT solvers;
- CAD only studying a subset of the input polynomials at once.

DEWCAD is working with our partner Maplesoft on a CAD driven SMT solver inside Maple.

Coverings not Decompositions

Another direction has been search based algorithms which:

- guess a sample point that may satisfy the constraints.
- if it does not (conflict), then generalise to a CAD cell and use cell description to inform the next guess.

First used by the `nlsat` algorithm from Jovanovic and de Moura [2012] which spawned the mcSAT proof framework de Moura and Jovanović [2013]. This deviated from the SMT framework with the theory and Boolean searches intermixed. The cells are produced independently and overlap to form a covering in the UNSAT case.

An alternative algorithm, `CDCAC` was introduced recently by Ábrahám et al. [2021] which fits the SMT framework. It performs a depth first theory search with conflict cells created to cover a dimension and then generalised to a cell in the next dimension.

The Benefit of Coverings

In both `nlsat` and `CDCAC` the cells in coverings are formed using fewer polynomials than a global CAD, and so tend to be bigger. Space covered with fewer cells (less computation) than full CAD.

Also: optimised algorithms designed for producing the single cells Brown [2013], Brown and Kosta [2015]; and results may be easier to embed into formal proofs Ábrahám et al. [2020].

The Benefit of Coverings

In both nlsat and CDCAC the cells in coverings are formed using fewer polynomials than a global CAD, and so tend to be bigger. Space covered with fewer cells (less computation) than full CAD.

Also: optimised algorithms designed for producing the single cells Brown [2013], Brown and Kosta [2015]; and results may be easier to embed into formal proofs Ábrahám et al. [2020].

The nlsat and CDCAC algorithms share many benefits but also differ. Experiments suggest each has substantial classes of problems where it outperforms the other giving potential for a meta-solver which selects the algorithm for the problem.

CDCAC algorithm maintains global cylindricity, which could be better suited for an extension to full QE.

DEWCAD is continuing to explore these algorithms with partners E. Ábrahám (RWTH Aachen) and C. Brown (US Naval Academy).

CAD Projection

There has been a lot of research on how best to perform CAD projection (identify the polynomials that form cell boundaries) since Collins. Until recently the choice was between

- Hong [1990], which improved upon Collins' projection; and
- McCallum [1998] which is smaller (much less computation) but fails for certain input classes.

However, recently in McCallum et al. [2019] a third operator was verified (**Lazard Projection**):

- Smaller than Hong but larger than McCallum.
- No risk of failure; but some extra work in cell creation.

Then Brown and McCallum [2020] it was shown that one could use Lazard projection but omit the additional work over McCallum projection in all but those cases where McCallum projection fails.

Lazard Projection

So: Lazard Projection is the same size as the smallest known projection (except in those cases where that previous projection fails. It is thus superior.

(Q) Why doesn't everyone use it yet?

Lazard Projection

So: Lazard Projection is the same size as the smallest known projection (except in those cases where that previous projection fails. It is thus superior.

(Q) Why doesn't everyone use it yet?

(A) Requires changes elsewhere in the CAD code base (for the extra work in cell creation). Moreover, there are many CAD optimisations and extensions formulated within McCallum projection theory and it is not clear they can all be safely used with Lazard projection.

The DEWCAD project will work with project partner Scott McCallum (Macquarie University) to investigate this.

New Applications

The DEWCAD project also has work packages dedicated to emerging CAD applications:

- **Biological Networks**: In Chemical Reaction Network Theory the identification of steady states requires the study of parametric systems of non-linear polynomial constraints Bradford et al. [2020]. DEWCAD will be working with project partner T. Sturm (CNRS, Inria, U. Lorraine and MPII, Saarland U.) and the SYMBIONT project on this.
- **Automated Economic Reasoning**: Many economic hypotheses can be formulated and analysed as QE problems Mulligan et al. [2018]. DEWCAD will be working with project partner C. Mulligan (University of Chicago) on this.

Contact Details and Advert

Contact Details

Matthew.England@coventry.ac.uk

https://matthewengland.coventry.domains/

Advert

Fully Funded PhD Position available at Coventry to work on Machine Learning to Improve Symbolic Integration and Symbolic Simplification. Sponsored by Maplesoft.
https://tinyurl.com/3exmk9vk
Deadline to Apply: 13th September 2021
Interviews and Decision: End September
PhD Start: Jan 2022

Thank You for Your Time

Thank You for Your Time

Special thanks to all the speakers, to my co-organizers and to the organizers of ACA'21

Bibliography I

E. Ábrahám, J.H. Davenport, M. England, G. Kremer, and Z. Tonks. New opportunities for the formal proof of computational real geometry? In P. Fontaine, K. Korovin, I.S. Kotsireas, P. Rümmer, and S. Tourret, editors, *Proceedings of the 5th Workshop on Satisfiability Checking and Symbolic Computation (SC^2 2020)*, number 2752 in CEUR Workshop Proceedings, pages 178–188, 2020. URL http://ceur-ws.org/Vol-2752/.

E. Ábrahám, J.H. Davenport, M.England, and G. Kremer. Deciding the consistency of non-linear real arithmetic constraints with a conflict driven search using cylindrical algebraic coverings. *Journal of Logical and Algebraic Methods in Programming*, 119:100633, 2021. URL https://doi.org/10.1016/j.jlamp.2020.100633.

Bibliography II

N.H. Arai, T. Matsuzaki, H. Iwane, and H. Anai. Mathematics by machine. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*, ISSAC '14, pages 1–8. ACM, 2014. URL https://doi.org/10.1145/2608628.2627488.

R. Bradford, J.H. Davenport, M. England, S. McCallum, and D. Wilson. Truth table invariant cylindrical algebraic decomposition. *Journal of Symbolic Computation*, 76:1–35, 2016. URL http://dx.doi.org/10.1016/j.jsc.2015.11.002.

R. Bradford, J.H. Davenport, M. England, H. Errami, V. Gerdt, D. Grigoriev, C. Hoyt, M. Košta, O. Radulescu, T. Sturm, and A. Weber. Identifying the parametric occurrence of multiple steady states for some biological networks. *Journal of Symbolic Computation*, 98:84–119, 2020. URL https://doi.org/10.1016/j.jsc.2019.07.008.

Bibliography III

C. Brown and S. McCallum. Enhancements to lazard's method for cylindrical algebraic decomposition. In F. Boulier, M. England, T.M. Sadykov, and E.V. Vorozhtsov, editors, *Computer Algebra in Scientific Computing*, volume 12291 of *Lecture Notes in Computer Science*, pages 129–149. Springer International Publishing, 2020. URL https://doi.org/10.1007/978-3-030-60026-6_8.

C.W. Brown. Constructing a single open cell in a cylindrical algebraic decomposition. In *Proceedings of the 38th International Symposium on Symbolic and Algebraic Computation*, ISSAC '13, pages 133–140. ACM, 2013. URL https://doi.org/10.1145/2465506.2465952.

C.W. Brown and J.H. Davenport. The complexity of quantifier elimination and cylindrical algebraic decomposition. In *Proceedings of the 2007 International Symposium on Symbolic and Algebraic Computation*, ISSAC '07, pages 54–60. ACM, 2007. URL https://doi.org/10.1145/1277548.1277557.

Bibliography IV

C.W. Brown and M. Kosta. Constructing a single cell in cylindrical algebraic decomposition. *Journal of Symbolic Computation*, 70:14 – 48, 2015. URL https://doi.org/10.1016/j.jsc.2014.09.024.

B. Caviness and J. Johnson. *Quantifier Elimination and Cylindrical Algebraic Decomposition*. Texts & Monographs in Symbolic Computation. Springer-Verlag, 1998. URL https://doi.org/10.1007/978-3-7091-9459-1.

G.E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In *Proceedings of the 2nd GI Conference on Automata Theory and Formal Languages*, pages 134–183. Springer-Verlag (reprinted in the collection Caviness and Johnson [1998]), 1975. URL https://doi.org/10.1007/3-540-07407-4_17.

Bibliography V

J.H. Davenport and J. Heintz. Real quantifier elimination is doubly exponential. *Journal of Symbolic Computation*, 5(1-2):29–35, 1988. URL https://doi.org/10.1016/S0747-7171(88)80004-X.

L. de Moura and D. Jovanović. A model-constructing satisfiability calculus. In R. Giacobazzi, J. Berdine, and I. Mastroeni, editors, *Verification, Model Checking, and Abstract Interpretation (Proc. VMCAI 2013)*, pages 1–12. Springer Berlin Heidelberg, 2013. URL https://link.springer.com/bookseries/558.

A. Dixit, V.H. Moll, and V. Pillwein. A hypergeometric inequality. *Annals of Combinatorics*, 20(1):65–72, 2016.

Bibliography VI

M. Erascu and H. Hong. Synthesis of optimal numerical algorithms using real quantifier elimination (Case Study: Square root computation). In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*, ISSAC '14, pages 162–169. ACM, 2014. URL https://doi.org/10.1145/2608628.2608654.

A. Grosslinger, M. Griebl, and C. Lengauer. Quantifier elimination in automatic loop parallelization. *Journal of Symbolic Computation*, 41(11):1206–1221, 2006. URL https://doi.org/10.1016/j.jsc.2005.09.012.

H. Hong. An improvement of the projection operator in cylindrical algebraic decomposition. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation*, ISSAC '90, pages 261–264. ACM, 1990. URL https://doi.org/10.1145/96877.96943.

Bibliography VII

D. Jovanovic and L. de Moura. Solving non-linear arithmetic. In B. Gramlich, D. Miller, and U. Sattler, editors, *Automated Reasoning: 6th International Joint Conference (IJCAR)*, volume 7364 of *Lecture Notes in Computer Science*, pages 339–354. Springer, 2012. URL https://doi.org/10.1007/978-3-642-31365-3_27.

C. Koutschan and E. Wong. Exact lower bounds for monochromatic schur triples and generalizations. In *Algorithmic Combinatorics: Enumerative Combinatorics, Special Functions and Computer Algebra*, pages 223–248. Springer, 2020.

G. Kremer and E. Ábrahám. Fully incremental CAD. *Journal of Symbolic Computation*, 100:11–37, 2020. URL https://doi.org/10.1016/j.jsc.2019.07.018.

Bibliography VIII

S. McCallum. An improved projection operation for cylindrical algebraic decomposition. In B. Caviness and J. Johnson, editors, *Quantifier Elimination and Cylindrical Algebraic Decomposition*, Texts & Monographs in Symbolic Computation, pages 242–268. Springer-Verlag, 1998. URL https://doi.org/10.1007/978-3-7091-9459-1_12.

S. McCallum, A. Parusińiski, and L. Paunescu. Validity proof of Lazard's method for CAD construction. *Journal of Symbolic Computation*, 92:52–69, 2019. URL https://doi.org/10.1016/j.jsc.2017.12.002.

Bibliography IX

C.B. Mulligan, J.H. Davenport, and M. England. TheoryGuru: A Mathematica package to apply quantifier elimination technology to economics. In J.H. Davenport, M. Kauers, G. Labahn, and J. Urban, editors, *Mathematical Software – Proc. ICMS 2018*, volume 10931 of *Lecture Notes in Computer Science*, pages 369–378. Springer International Publishing, 2018. URL https://doi.org/10.1007/978-3-319-96418-8_44.

L.C. Paulson. Metitarski: Past and future. In L. Beringer and A. Felty, editors, *Interactive Theorem Proving*, volume 7406 of *Lecture Notes in Computer Science*, pages 1–10. Springer, 2012. URL https://doi.org/10.1007/978-3-642-32347-8_1.