Topics in Cylindrical Algebraic Decomposition

Corin Lee, cel34@bath.ac.uk
Supervisors: Prof. G.K. Sankaran and Prof. J.H. Davenport
Department of Mathematical Sciences, University of Bath

Introduction

A *cylindrical algebraic decomposition* (CAD) is a method in real semi-algebraic geometry that partitions real space into a finite number of connected *cells* with useful structural properties.

Originally introduced by Collins for quantifier elimination [1], CAD remains practical in many settings despite its worst-case doubly-exponential complexity [2], with even small efficiency improvements making a significant difference in practice.

My work aims to make CAD more accessible and effective through:

- Tools for visualising decompositions and understanding CAD behaviour
- Implementations and tools in the mathematical software systems *Maple* and *Macaulay2*
- Benchmarking of CAD algorithms to identify cases where one is preferable
- Identifying when CAD is (not) suitable, alternative methods, and input simplification

Key Features

Given a set of polynomials $\mathcal{F}_n$ in n variables, a CAD decomposes n-dimensional real space into finitely many disjoint regions called *cells*. These cells are:

- **Algebraic:** They can be defined by a finite set of polynomial equations and inequalities
- **Cylindrically arranged:** They stack up over cells in lower dimensions.
- **Invariant:** The polynomials have a fixed property in each cell, e.g. constant sign (positive/negative/zero), truth value (true/false according to a logical formula).

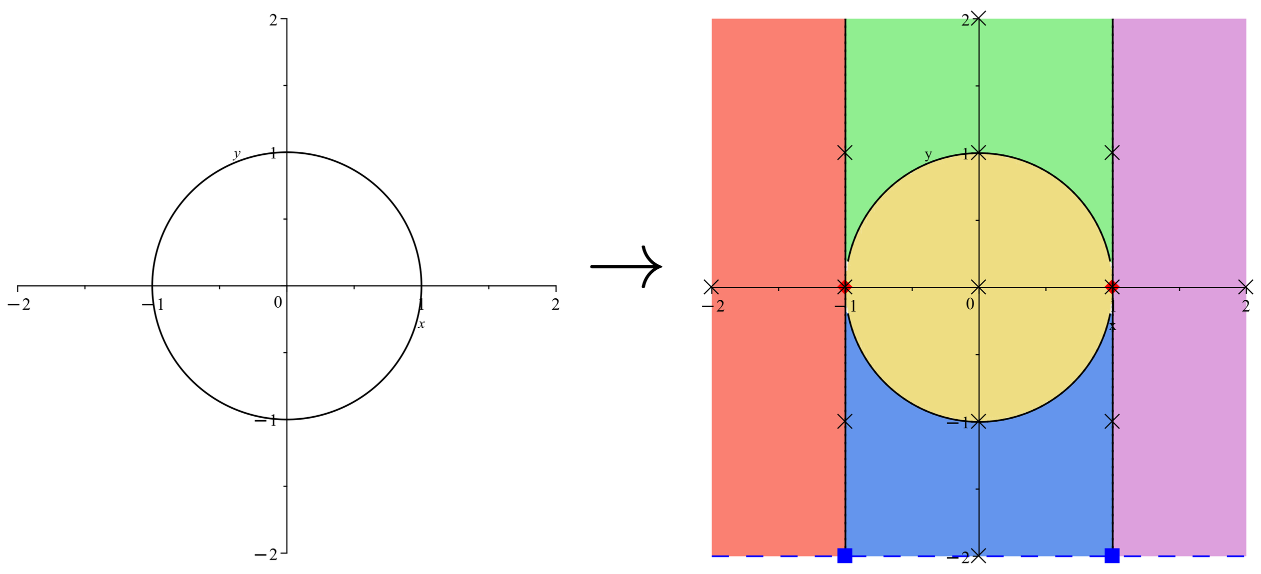


Figure 1: CAD of x^2+y^2-1 in two-dimensional space, consisting of 13 cells (coloured regions, edges and points). Each has a sample point (black cross), and stacks over a one-dimensional CAD cell below (blue squares and lines).

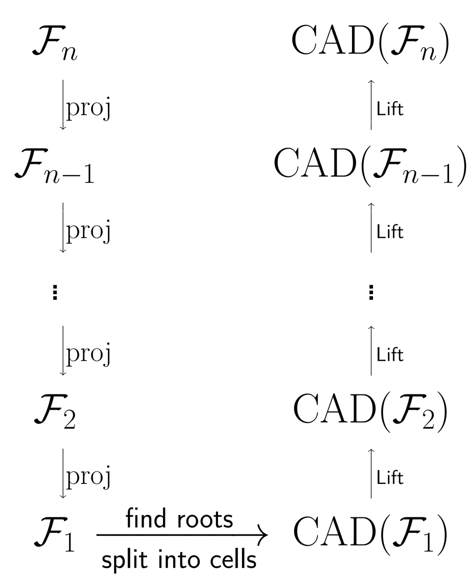
CAD Construction

There are multiple algorithms for constructing a CAD, but the most common is a *projection and lifting* algorithm:

Projection phase: Apply *projection operator* proj to $\mathcal{F}_n$ to produce a new set of polynomials $\mathcal{F}_{n-1}$ in one dimension fewer. Repeat this until one obtains $\mathcal{F}_1$.

Base Phase: Decompose one-dimensional real space into cells: the solutions of each polynomial in $\mathcal{F}_1$ and the open intervals either side. This is $\mathrm{CAD}(\mathcal{F}_1)$.

Lifting Phase: Substitute a *sample point* from each cell into $\mathcal{F}_2$ and decompose as before, giving $\mathrm{CAD}(\mathcal{F}_2)$. Repeat this process until one obtains $\mathrm{CAD}(\mathcal{F}_n)$.



CAD in *Maple*

Graphing Tool

Development of a tool for visualising 2D CADs in *Maple* (see Figures 1 and 2), enabling clearer communication and helping users better explore and understand outputs.

- Clearly displays cells by dimension with use of colour
- Displays sample points and cells of 1D CAD below
- Supports output from both the `QuantifierElimination` and `RegularChains` packages

Algorithm Benchmarking

- Benchmarked `RegularChains` and `QuantifierElimination` CAD algorithms on SMT-LIB style dataset
- Measured runtime, memory use, and cell count under different variable orderings, forms, and uses of equational constraints
- Found notable performance and output differences, suggesting each has strengths in particular settings

Motivation

CAD is a symbolic method for solving problems in real algebraic geometry, and CADs can be used to determine whether a polynomial system has a solution. But why not just test points numerically? Suppose we want to find if there is a value of x such that

$$g(x) = 3 - x^2 > 0 \quad \text{and} \quad h(x) = (7x-12)(x^2+x+1) > 0.$$

Sampling 1000+ values numerically might not find such an x, and could not prove such a value does not exist — but a CAD-based method can:

$$CAD(\{g,h\}) = \{x < -\sqrt{3}\} \cup \{x = -\sqrt{3}\} \cup \{-\sqrt{3} < x < 12/7\} \cup \{x = 12/7\} \cup \{12/7 < x < \sqrt{3}\} \cup \{x = \sqrt{3}\} \cup \{x > \sqrt{3}\}.$$

Creating a CAD reduces the search from "everywhere" to checking a finite number of sample points, and will show whether a suitable point exists, making it very suitable for quantifier elimination, with applications in fields such as robotics, economics, and biology.

CAD in *Macaulay2*

Our package `CylindricalAlgebraicDecomposition` is the first implementation of CAD in *Macaulay2*:

- Computes *open CADs* using Lazard projection
- Outputs a tree of sample points for full-dimensional cells
- Offers a new heuristic for variable ordering
- Users can query for positive solutions, explore full/partial trees, and step through the algorithm
- Improves existing root isolation techniques

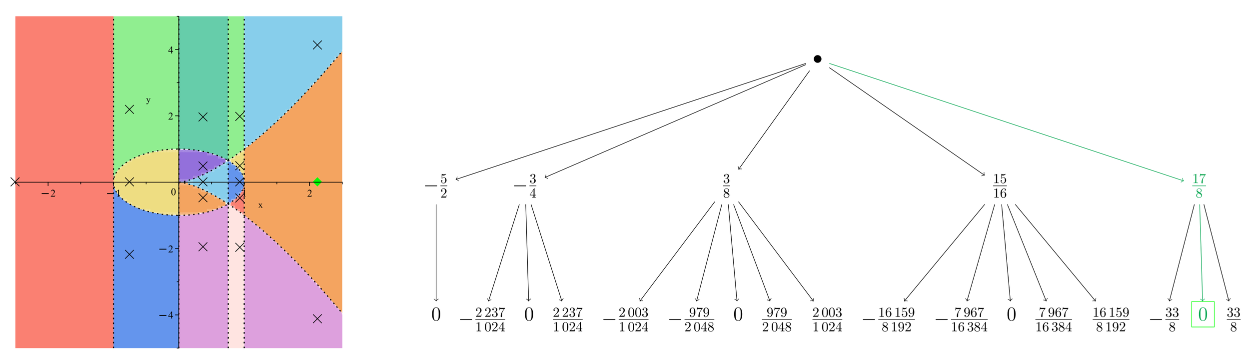


Figure 2: Open CAD of $\{x^2+y^2-1, x^3-y^2\}$ and tree diagram of corresponding sample points. Green sample point is one where both polynomials are positive.

Read the pre-print by visiting `https://arxiv.org/abs/2503.21731` or scanning the QR code at the bottom.

Methods and Features

Other tools and features to exploit include:

- **Block-cylindricity:** Exploiting the freedom given when variable order need only be partially fixed
- **Symmetry:** Preserving symmetries to allow for polynomial degree reduction through a change of variables
- **Effective quantification:** Treating otherwise-free variables as quantified to allow for more efficient variable orderings

Future Work

- **Improve *Maple* graphing tool:** 1D/2D/3D projection of higher-dimension CADs, multiple colouring schemes, truth-invariance
- **Extend *Macaulay2* package to full CAD:** This would allow solving non-linear arithmetic problems through quantifier elimination
- **Further benchmarking and complexity analysis:** Further comparisons may highlight when one algorithm is more favourable

References

[1] George E. Collins. "Quantifier elimination for real closed fields by cylindrical algebraic decomposition". In: *Automata Theory and Formal Languages*. Springer, 1975, pp. 134–183.

[2] James H. Davenport and Joos Heintz. "Real quantifier elimination is doubly exponential". In: *Journal of Symbolic Computation* 5.1 (1988), pp. 29–35. DOI: 10.1016/S0747-7171(88)80004-X.

For further information, visit:

`https://github.com/cel34-bath/CADRepository`