Formulations in Cylindrical Algebraic Decomposition
CICM 2013 — Calculemus

Russell J. Bradford, James H. Davenport, Matthew England, David J. Wilson

Department of Computer Science
University of Bath

{R.J.Bradford, J.H.Davenport, M.England, D.J.Wilson}@bath.ac.uk

July 9th, 2013
Table of Contents

Cylindrical Algebraic Decomposition
   Definition
   Example
   Algorithm

Advances in CAD

Metrics and Heuristics

Formulating a Problem

Conclusions and Future Work
Cylindrical Algebraic Decomposition

Definition

A partition $\mathcal{D}$ of $\mathbb{R}^n$ is a Cylindrical Algebraic Decomposition if:

- Each cell $D \in \mathcal{D}$ is semi-algebraic: describable by polynomials
- Cells are cylindrically arranged: for each $k$ and for $D_1, D_2 \in \mathcal{D}$ then

$$\pi_k(D_1) = \pi_k(D_2) \quad \text{or} \quad \pi_k(D_1) \cap \pi_k(D_2) = \emptyset.$$ 

where $\pi_k : \mathbb{R}^n \rightarrow \mathbb{R}^k$ is the projection operator onto the first $k$ coordinates.

For a set of polynomials $F \subseteq \mathbb{Q}[x_1, \ldots, x_n]$, we say a CAD is $F$-(sign)-invariant if also:

- For each $f \in F$ and $D \in \mathcal{D}$, $f$ has constant sign on $D$. 

Bradford, Davenport, England, and Wilson

Formulations in CAD
Example CAD

Sample CAD for two intersecting circles —
55 cells: 10 0-dimensional, 27 1-dimensional, 18 2-dimensional
Idea Behind the Algorithm

- **Project** out variables to produce polynomials of one fewer variable
- **Decompose** $\mathbb{R}^1$ w.r.t. univariate polynomials
- **Construct** stacks by lifting over the decomposition - splitting according to the polynomials
- **Repeat** up to $\mathbb{R}^n$

A projection operator:

$$P(F) = \{ \text{coeffs}_{x_n}(f) \mid f \in F \} \cup \{ \text{disc}_{x_n}(f) \mid f \in F \} \cup \{ \text{res}_{x_n}(f,g) \mid f,g \in F, f \neq g \}.$$
Idea Behind the Algorithm

- **Project** out variables to produce polynomials of one fewer variable
- **Decompose** $\mathbb{R}^1$ w.r.t. univariate polynomials
- Construct *stacks* by *lifting* over the decomposition - splitting according to the polynomials
- **Repeat** up to $\mathbb{R}^n$
Idea Behind the Algorithm

- *Project* out variables to produce polynomials of one fewer variable
- *Decompose* $\mathbb{R}^1$ w.r.t. univariate polynomials
- Construct *stacks* by *lifting* over the decomposition - splitting according to the polynomials
- *Repeat* up to $\mathbb{R}^n$
Idea Behind the Algorithm

- *Project* out variables to produce polynomials of one fewer variable
- *Decompose* $\mathbb{R}^1$ w.r.t. univariate polynomials
- Construct *stacks* by *lifting* over the decomposition - *splitting* according to the polynomials
- *Repeat* up to $\mathbb{R}^n$
Features of CAD

- Introduced by Collins in 1975 as an alternative to Tarski’s decision method for real closed fields.
- Hence, can be used for quantifier elimination:
  - Given formula $\Phi$ extract all polynomials, $F$.
  - Construct CAD with respect to $F$
  - Evaluate $\Phi$ on lower-dimensional cells (of the free variables)
  - Construct quantifier-free formula from representations of valid cells
- Can be used as a back-end for tools such as METiTARSKI.
- Output has doubly-exponential complexity in the number of variables — variable ordering matters!
Table of Contents

Cylindrical Algebraic Decomposition

Advances in CAD
   Equational Constraints and TTICAD

Metrics and Heuristics

Formulating a Problem

Conclusions and Future Work
Equational Constraints

- Given a formula $\varphi$, an *equational constraint* is an equation logically implied by $\varphi$.
- Often occur explicitly as $\varphi := (f = 0) \land \psi$.
- In [McC99], shown you can simplify projection:
  - Fully project the equational constraint
  - Then only add $\text{res}_{x_n}(f, g)$ for all $g \in \psi$.
- If multiple equational constraints occur then designate which to use.
- This designation choice can affect the output size.
Truth-Table Invariant CAD

- Often, we do not care about the signs of polynomials in cells, but rather the truth-value of a quantifier-free formula (QFF) on it.
- In [BDEMW13] a Truth-Table Invariant CAD (TTICAD) was defined: for a list of QFFs, \( \{ \varphi_i \} \), each \( \varphi_i \) has constant truth value on each cell.
- Algorithm given (and implemented) for when each \( \varphi_i \) has a designated equational constraint.
- Have to decide how to split formula into \( \{ \varphi_i \} \) and designate equational constraints for each QFF.
## Table of Contents

- Cylindrical Algebraic Decomposition
- Advances in CAD
- Metrics and Heuristics
  - Previous work: sotd
  - New metric: ndrr
- Formulating a Problem
- Conclusions and Future Work
Sum of Total Degrees - \texttt{sotd}

- In [DSS04], tried to find a metric to choose the best variable ordering for a CAD problem.
- Investigated a variety of basic properties of the polynomials.
- Found statistically that \texttt{sotd} was best: \textit{sum of the total degrees of all monomials in all polynomials in the full projection set}.
- Simple to compute and can use \textit{greedily} to form a heuristic:
  1. Project once with respect to all variables
  2. Pick variable that gives lowest \texttt{sotd}
  3. Project with respect to remaining variables
  4. Repeat
Drawback of sotd

- \( f := (x - 1)(y^2 + 1) - 1: \)
  \[
P_y(\{f\}) = \{x - 1, x - 2\}
\]
- CAD contains 11 cells
- Same sotd (= 8) for both variable orderings
Drawback of sotd

- \( f := (x - 1)(y^2 + 1) - 1: \)
  \[
P(x)\{f\} = \{y^2 + 1\}
  \]
- CAD contains 3 cells
- Same sotd (= 8) for both variable orderings
New metric: ndrr

- sotd does not take into account the real geometry of a polynomial.
- Motivated new metric ndrr: *number of distinct real roots of univariate projection polynomials*.
- ndrr measures complexity of 1-dimensional CAD.
- Slightly more costly than sotd but still insignificant compared to CAD.
- Can also be fooled: only takes into account 1-dimensional CAD.
Table of Contents

Cylindrical Algebraic Decomposition

Advances in CAD

Metrics and Heuristics

Formulating a Problem
  Göbner Bases and TTICAD
  Choices available
  Heuristics for formulation

Conclusions and Future Work
Gröbner Bases and TTICAD

- In [WBD12], was shown that preconditioning by taking Gröbner Bases of equational constraints can be highly beneficial.
- Not universally helpful: metric TNoI can help predict.
- For a TTICAD problem can apply Gröbner preconditioning to separate clauses
- TNoI does not take into account interaction between clauses; sotd and ndrr much more accurate
- Example in 2 variables with 2 clauses (2 equational constraints): 725/657 cells reduced to 27/29 cells.
Choices necessary for formulating a problem

▶ As seen, variable ordering is an important choice when formulating a problem.
▶ But there are many other choices:
  ▶ Designating an equational constraint to use
  ▶ How to decompose a formula into QFFs for TTICAD
  ▶ Designating equational constraints for each QFF with TTICAD
  ▶ Whether to precondition (e.g. Gröbner bases)
▶ Choices influence each other so can’t necessarily consider separately.
▶ Very quickly becomes combinatorially overwhelming!
Heuristics

- Need computationally cheap ways to decide what formulation to use.
- Experimental analysis confirms using a combination of sotd and ndrr can accurately predict “good” formulations.
- For TTICAD can proceed as follows:
  1. Put formula in Disjunctive Normal Form
  2. For each clause consider the number of equational constraints:
     - 0 or 1 trivial
     - > 1 try various designations along with splitting clause in all possible manners
- Heuristics implemented within ProjectionCAD package for MAPLE: freely available at http://opus.bath.ac.uk/33180/
# Table of Contents

Cylindrical Algebraic Decomposition

Advances in CAD

Metrics and Heuristics

Formulating a Problem

Conclusions and Future Work
Conclusions and Extensions

- How a CAD problem is formulated is key to its complexity.
- With extensions to CAD algorithms, the number of choices can be overwhelming and non-intuitive.
- A combination of sotd and ndrr can be used to select “good” formulations.
- Future Work:
  - Can we extend to ndrr to multivariate or use it greedily?
  - What is the relation between formulations and variable ordering?
  - Can we extend our work to bi-equational constraints?
- **Main Goal:** Would like CAD algorithms that allow much more flexible input, making near-optimal choices for the user based on our heuristics.
Conclusions and Extensions

- How a CAD problem is formulated is key to its complexity.
- With extensions to CAD algorithms the number of choices can be overwhelming and non-intuitive.
- A combination of sotd and ndrr can be used to select “good” formulations.
- Future Work:
  - Can we extend to ndrr to multivariate or use it greedily?
  - What is the relation between formulations and variable ordering?
  - Can we extend our work to bi-equational constraints?
- **Main Goal:** Would like CAD algorithms that allow much more flexible input, making near-optimal choices for the user based on our heuristics.

Thank you