Cylindrical Algebraic Decomposition

Applications and Computation

Mikhail Dubov

May 12, 2016

Outline

Cylindrical Algebraic Decomposition

Cylindrical Algebraic Decomposition (CAD):

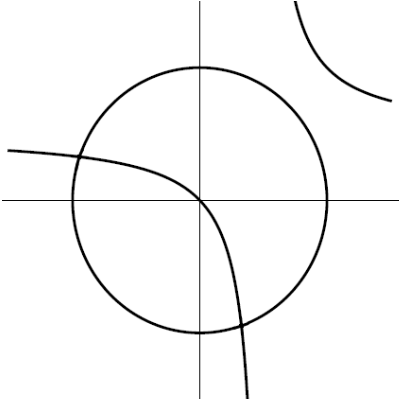
- gives a decision procedure for real closed fields (Caviness and Johnson [1998])
- is a general tool for dealing with **semilagebraic sets**: subsets of $\mathbb{R}^n$ that can be described by polynomial equations and inequalities (Kauers [2011])
- allows to solve a wide rage of problems:
  - quantifier elimination over the reals
  - tests for emptiness, finiteness, connectedness, etc. of semialgebraic sets
  - sample point determination of a given nonempty semialgebraic set
  - computing connected components of a given semialgebraic set
  - ...

Outline

CAD: Geometric Point of View

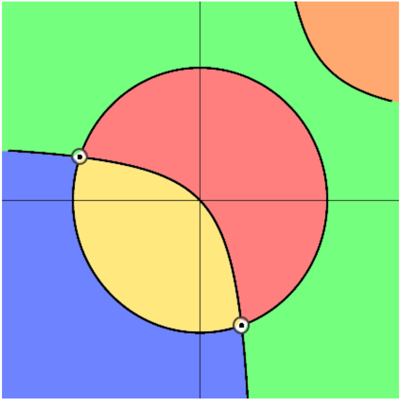
- $P = \{p_1, \ldots, p_m\}$ – finite set of polynomials
- $x_1, \ldots, x_n$ – variables
- P induces an **algebraic decomposition** of $\mathbb{R}^n$ into **cells**
- A **cell** is a maximal connected subset of $\mathbb{R}^n$ where all the p_i have the same sign ($-1/0/1$)

Algebraic Decomposition: Example



$$P = \{x^2 + y^2 - 4, (x-1)(y-1) - 1\}$$

Algebraic Decomposition: Example



$AD = \{13 \text{ cells in } \mathbb{R}^2\}$ *(5 "areas", 6 "arcs", 2 points)*

Cylindrical Algebraic Decomposition

Intuitive definition:

- If an algebraic decomposition of $\mathbb{R}^n$ is **cylindrical**, then for every $k = 1, \ldots, n$ the cells of the decomposition can be divided into groups so that all cells of one group have the same $x_1, \ldots, x_k$-coordinates.

Cylindrical Algebraic Decomposition

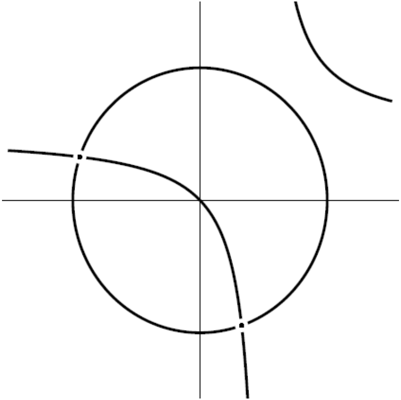
Intuitive definition:

- If an algebraic decomposition of $\mathbb{R}^n$ is **cylindrical**, then for every $k = 1, \ldots, n$ the cells of the decomposition can be divided into groups so that all cells of one group have the same $x_1, \ldots, x_k$-coordinates.

Formal definition is *recursive*:

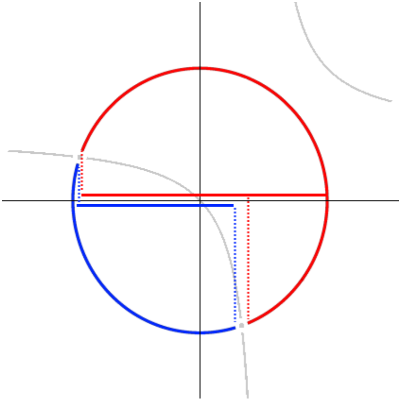
- **n = 1:** Every AD of $\mathbb{R}$ is cylindrical.
- **n > 1:** An AD of $\mathbb{R}^n$ is cylindrical if:
  - the projection of any two cells down to $\mathbb{R}^{n-1}$ is either cylindrical or disjoint;
  - the projections of all the cells down to $\mathbb{R}^{n-1}$ form a CAD of $\mathbb{R}^{n-1}$.

Cylindrical Algebraic Decomposition: Example



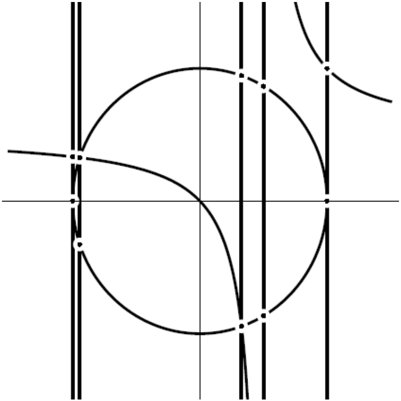
$$P = \{x^2 + y^2 - 4, (x-1)(y-1) - 1\}$$

Cylindrical Algebraic Decomposition: Example



Not a CAD: there is a pair of cells which, when projected to the horizontal axis, has images that are neither identical nor disjoint

Cylindrical Algebraic Decomposition: Example



A true CAD requires more cells

Cylindrical Algebraic Decomposition: Example

- In our example, to make the AD cylindrical, we have to refine it by adding a suitable univariate polynomial in x to

$$P = \{x^2 + y^2 - 4, (x-1)(y-1) - 1\}$$

- The set of polynomials that induces the corresponding CAD is:

$$P' = \{x^2 + y^2 - 4, (x-1)(y-1) - 1, \\ (x^2-4)(x-1)(x^4 - 2x^3 - 2x^2 + 8x - 4)\}$$

- The cells in a CAD in $\mathbb{R}^2$ are arranged into verical **"cylinders"**

Cylindrical Algebraic Decomposition: Example

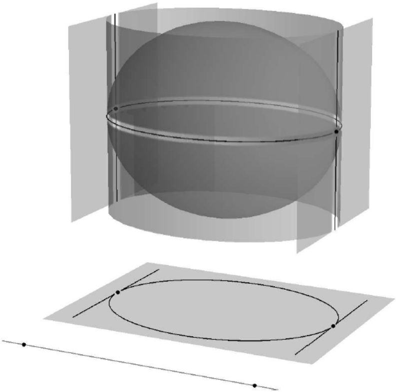
- In case of more than 2 variables ($n > 2$), we may have to add more than one polynomial to make an AD cylindrical
- For example,

$$P = \{x^2 + y^2 + z^2 - 1\}$$

– splits $\mathbb{R}^3$ into the interior, the boundary, and the exterior of the unit ball

- This AD is not cylindrical
- To make in cylindrical, we should add to P:
  - $x^2 + y^2 - 1$, which puts a cylinder around the unit ball
  - $x^2 - 1$, which corresponds to two vertical tangents to the unit circle

Cylindrical Algebraic Decomposition: Example



$P' = \{x^2 + y^2 + z^2 - 1, x^2 + y^2 - 1, x^2 - 1\}$ induces a CAD

Cylindrical Algebraic Decomposition

Here is what we did in both examples:

- take a finite set P of polynomials that induces an AD
- produce another set Q of polynomials s.t. $P \cup Q$ incuces a CAD

This is exactly what **Collins' CAD algorithm** does (stay tuned).

Outline

CAD: Logical Point of View

- Cells in a CAD can be described by logical formulas involving polynomial equations and inequalities.
- More formally, our language consists of:
  - variables
  - rational numbers
  - arithmetic operations $(+, -, \cdot, /)$
  - equality and inequality relations $(=, \neq, <, >, \leq, \geq)$
  - logical connectives $(\vee, \wedge, \ldots)$
  - quantifiers $(\forall, \exists)$

CAD: Logical Point of View

- Geometric point of view:
  - take a set of polynomials P
  - produce another set Q that completes P to a CAD
- Logical point of view:
  - take a formula
  - produce an equivalent formula which has a special structure

CAD: Logical Point of View

This **CAD formula format** can be described recursively:

- **n = 1:** a formula in one variable x is in CAD format if it is of the form

$$\Phi_1 \vee \Phi_2 \vee \cdots \vee \Phi_m$$

  where each Φ_k is one of the following:

  - $x < \alpha$
  - $\alpha < x < \beta$
  - $x > \beta$
  - $x = \gamma$ (α, β, γ are some real algebraic numbers)

  and any two Φ_k are mutually inconsistent.

CAD: Logical Point of View

This **CAD formula format** can be described recursively:

- **$n = 1$:** a formula in one variable x is in CAD format if it is of the form

$$\Phi_1 \vee \Phi_2 \vee \cdots \vee \Phi_m$$

  where each Φ_k is one of the following:
  - $x < \alpha$
  - $\alpha < x < \beta$
  - $x > \beta$
  - $x = \gamma$ (α, β, γ are some real algebraic numbers)

  and any two Φ_k are mutually inconsistent.
- **$n > 1$:** a formula in n variables $x_1, \ldots, x_n$ is in CAD format if it is of the form

$$(\Phi_1 \wedge \Psi_1) \vee (\Phi_2 \wedge \Psi_2) \vee \cdots \vee (\Phi_m \wedge \Psi_m)$$

  where Φ_k are in CAD format with respect to x_1 and Ψ_k are in CAD format with respect to $x_2, \ldots, x_n$ and satisfiable whenever x_1 is replaced by a real algebraic number satisfying Φ_k.

Logical CAD: Example

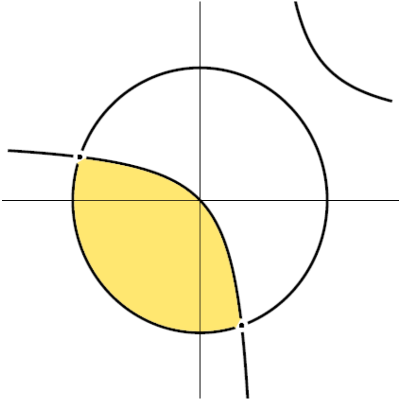
Note that the CAD format naturally describes a union of cells in a cylindrical algebraic decomposition of $\mathbb{R}^n$.

E.g. we can decompose one of the cells from our first example in $\mathbb{R}^2$ as follows:

$$
\begin{aligned}
CAD(x^2 + y^2 - 4 < 0 \wedge (x-1)(y-1) - 1 > 0) = \\
(-2 < x \leq -1.89005 \wedge -\sqrt{4-x^2} < y < \sqrt{4-x^2}) \\
\vee (-1.89005 < x < 0.653986 \wedge -\sqrt{4-x^2} < y < \frac{x}{x-1}),
\end{aligned}
$$

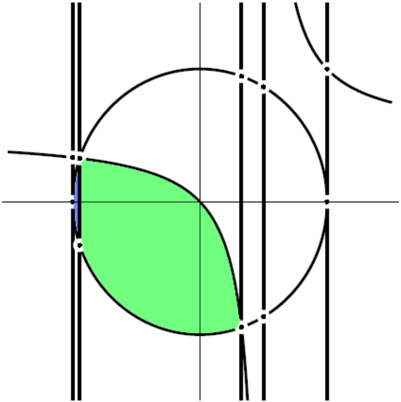
where -1.89005 and 0.653986 are approximate roots of $(x^2-4)(x-1)(x^4-2x^3-2x^2+8x-4)$ – the polynomial that we added to P to obtain a CAD.

Logical CAD: Example



$$x^2 + y^2 - 4 < 0 \wedge (x-1)(y-1) - 1 > 0$$

Logical CAD: Example



CAD:

$$(-2 < x \leq -1.89005 \wedge -\sqrt{4-x^2} < y < \sqrt{4-x^2})$$

$$\vee (-1.89005 < x < 0.653986 \wedge -\sqrt{4-x^2} < y < \frac{x}{x-1})$$

Outline

Quantifier Elimination

- **Quantifier Elimination:** given a quantified formula, find another formula without quantifiers which is equivalent to it (over the reals)
- Originally CAD was invented as a quanitifer elimination-based decision procedure over real closed fields

Quantifier Elimination

- **Quantifier Elimination:** given a quantified formula, find another formula without quantifiers which is equivalent to it (over the reals)
- Originally CAD was invented as a quanitifer elimination-based decision procedure over real closed fields
- Example:

$$(\forall x)(\forall y)[(x^2 + ay^2 \leq 1) \implies (ax^2 - a^2xy + 2 \geq 0)]$$

is equivalent to this quantifier-free expression:

$$(a \geq 0) \wedge (a^3 - 8a - 16 \leq 0),$$

which defines the interval $[0, 3.538]$.

Quantifier Elimination with CAD: Existential quantifier

We will start with the case of two variables ($n = 2$).

- Assume we have a formula in two variables x_1, x_2 in CAD format:

$$(\Phi_1 \wedge \Psi_1) \vee (\Phi_2 \wedge \Psi_2) \vee \cdots \vee (\Phi_m \wedge \Psi_m)$$

Quantifier Elimination with CAD: Existential quantifier

We will start with the case of two variables ($n = 2$).

- Assume we have a formula in two variables x_1, x_2 in CAD format:

$$(\Phi_1 \wedge \Psi_1) \vee (\Phi_2 \wedge \Psi_2) \vee \cdots \vee (\Phi_m \wedge \Psi_m)$$

- Then the quantified formula:

$$\exists x_2 \in \mathbb{R} : (\Phi_1 \wedge \Psi_1) \vee (\Phi_2 \wedge \Psi_2) \vee \cdots \vee (\Phi_m \wedge \Psi_m)$$

is equivalent to:

$$\Phi_1 \vee \Phi_2 \vee \cdots \vee \Phi_m$$

which contains only x_1.

Quantifier Elimination with CAD: Universal quantifier

- Assume we have a formula in two variables x_1, x_2 in CAD format:

$$(\Phi_1 \wedge \Psi_1) \vee (\Phi_2 \wedge \Psi_2) \vee \cdots \vee (\Phi_m \wedge \Psi_m)$$

Quantifier Elimination with CAD: Universal quantifier

- Assume we have a formula in two variables x_1, x_2 in CAD format:

$$(\Phi_1 \wedge \Psi_1) \vee (\Phi_2 \wedge \Psi_2) \vee \cdots \vee (\Phi_m \wedge \Psi_m)$$

- For the quantified formula:

$$\forall x_2 \in \mathbb{R} : (\Phi_1 \wedge \Psi_1) \vee (\Phi_2 \wedge \Psi_2) \vee \cdots \vee (\Phi_m \wedge \Psi_m)$$

  we have to go through the Ψ_i and check which of them represent the whole real line, i.e. are of the form

$$x_2 > \alpha \vee x_2 = \alpha \vee \alpha < x_2 < \beta \vee x_2 = \beta \vee \beta < x_2 < \gamma \vee \dots$$
$$\cdots \vee x_2 = \delta \vee x_2 > \delta$$

Quantifier Elimination with CAD: Universal quantifier

- Assume we have a formula in two variables x_1, x_2 in CAD format:

$$(\Phi_1 \wedge \Psi_1) \vee (\Phi_2 \wedge \Psi_2) \vee \cdots \vee (\Phi_m \wedge \Psi_m)$$

- For the quantified formula:

$$\forall x_2 \in \mathbb{R} : (\Phi_1 \wedge \Psi_1) \vee (\Phi_2 \wedge \Psi_2) \vee \cdots \vee (\Phi_m \wedge \Psi_m)$$

  we have to go through the Ψ_i and check which of them represent the whole real line, i.e. are of the form

$$x_2 > \alpha \vee x_2 = \alpha \vee \alpha < x_2 < \beta \vee x_2 = \beta \vee \beta < x_2 < \gamma \vee \dots$$
$$\cdots \vee x_2 = \delta \vee x_2 > \delta$$

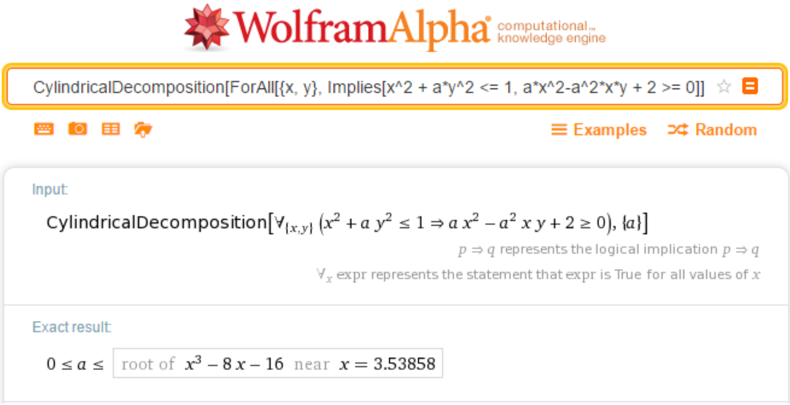
- If the relevant Φ_i are e.g. Ψ_3, Ψ_7 and Ψ_{28}, then the quantified formula is equivalent to

$$\Phi_3 \vee \Phi_7 \vee \Phi_{28}$$

Quantifier Elimination with CAD: Remarks

- The case of more variables ($n > 2$) can be handled in quite the same way due to the recursive nature of the formulas in the CAD format.
- The CAD format is defined with respect to some **variable order**, so it should be chosen to be compatible with the **order of the quantifiers**.
- Quantifier elimination is the most important application of CAD and so most implementations of CAD can do it.

You can try it right now!



http://mathworld.wolfram.com/CylindricalAlgebraicDecomposition.html

Outline

CAD Algorithm

The first CAD algorithm is due to Collins (Caviness and Johnson [1998]).

- **Input:** a set P of polynomials over $x_1, \ldots, x_n$
- **Output:** a description of the CAD of $\mathbb{R}^n$ induced by P:
  - Number of cells in the CAD
  - A sample point for each cell
  - *(Extended version)* The actual formulas defining cells
- **Complexity:** for any fixed number of variables, its computing time is a **polynomial** function of the remaining parameters of the input size.

CAD Algorithm

The general strategy is again **recursive**. The algorithm consists of 3 phases:

CAD Algorithm

The general strategy is again **recursive**. The algorithm consists of 3 phases:

1. **Projection phase:**
   - computes successive sets of polynomials $P_{n-1}, P_{n-2}, \ldots, P_1$ in $n-1, n-2, \ldots, 1$ variables
   - eliminates one variable at a time
   - at each step, the new set of polynomials in $P_{k-1} = PROJ(P_k)$ variables should induce a CAD which is also induced by P_k

CAD Algorithm

The general strategy is again **recursive**. The algorithm consists of 3 phases:

1. **Projection phase:**
   - computes successive sets of polynomials $P_{n-1}, P_{n-2}, \ldots, P_1$ in $n-1, n-2, \ldots, 1$ variables
   - eliminates one variable at a time
   - at each step, the new set of polynomials in $P_{k-1} = PROJ(P_k)$ variables should induce a CAD which is also induced by P_k
2. **Base phase:** constructs a CAD of $\mathbb{R}^1$ for P_1 (polynomials in one variable)

CAD Algorithm

The general strategy is again **recursive**. The algorithm consists of 3 phases:

1. **Projection phase:**
   - computes successive sets of polynomials $P_{n-1}, P_{n-2}, \ldots, P_1$ in $n-1, n-2, \ldots, 1$ variables
   - eliminates one variable at a time
   - at each step, the new set of polynomials in $P_{k-1} = PROJ(P_k)$ variables should induce a CAD which is also induced by P_k
2. **Base phase:** constructs a CAD of $\mathbb{R}^1$ for P_1 (polynomials in one variable)
3. **Extension phase:**
   - extends a CAD of $\mathbb{R}^1$ to $\mathbb{R}^2, \ldots, \mathbb{R}^n$
   - adds one dimension at a time

CAD Algorithm: Proejction Phase

- **Input:** Set of polynomials $P = \{p_1, \ldots, p_m\}$
- **Projection:**

$$PROJ(P) = \cup(\cup_{i=1}^{n} \cup_{G_i \in RED(p_i)} (\{LC(G_i)\} \cup PSC(G_i, G_i')))$$
$$(\cup_{1 \leq i \leq j \leq n} \cup_{G_i \in RED(p_i) \& G_j \in RED(p_j)} PSC(G_i, G_j))$$

where

- $LC(F)$ is the *leading coefficient* of F
- $LT(F)$ is athe *leading term* of F
- $red(F) = F - LT(F)$ is called a *reducium* of F
- $RED(F) = \{red^k(F) | 0 \leq k \leq deg(F) \;\&\; red^k(F) \neq 0\}$ is the *reducia set* of F
- $psc_j(F, G)$ is the j^{th} *principal subresultant coefficient* of F and G
- $PSC(F, G) = \{psc_j(F, G) | 0 \leq j \leq n \;\&\; psc_j(F, G) \neq 0\}$ is the *psc set* of F and G

References

Bob F. Caviness and J. R. Jeremy R. Johnson, editors. *Quantifier elimination and cylindrical algebraic decomposition*. Texts and monographs in symbolic computation. Springer, Wien, New York, 1998. ISBN 3-211-82794-3. URL http://opac.inria.fr/record=b1102167. Papers from a symposium held Oct. 6-8, 1993, at the Research Institute for Symbolic Computation in Linz, Austria.

Manuel Kauers. How To Use Cylindrical Algebraic Decomposition. *Seminaire Lotharingien de Combinatoire*, 65(B65a):1–16, 2011.