Truth table invariant

cylindrical algebraic decomposition

Matthew England
The University of Bath

With: Russell Bradford, James Davenport & David Wilson (Bath), Scott McCallum (Macquarie), Changbo Chen (CIGIT) and Marc Moreno Maza (Western Ontario)

International Congress of Mathematicians
Seoul, South Korea
13–21 August 2014

Supported by EPSRC Grant EP/J003247/1.
Outline

1 Background
   - What is a CAD?
   - How do you build a CAD?

2 Truth table invariance
   - TTICAD
   - TTICAD via complex cylindrical decompositions
## Outline

1. **Background**
   - What is a CAD?
   - How do you build a CAD?

2. **Truth table invariance**
   - TTICAD
   - TTICAD via complex cylindrical decompositions
A Cylindrical Algebraic Decomposition (CAD) is:

- a decomposition meaning a partition of $\mathbb{R}^n$ into connected subsets called cells;
- (semi)-algebraic meaning that each cell can be defined by a sequence of polynomial equations and inequations.
- cylindrical meaning the cells are arranged in a useful manner - their projections are either equal or disjoint.
A CAD of $\mathbb{R}^2$ is given by the following collections of 13 cells:

$$[x < -1, y = y],$$
$$[x = -1, y < 0], [x = -1, y = 0], [x = -1, y > 0],$$
$$[-1 < x < 1, y^2 + x^2 - 1 > 0, y > 0],$$
$$[-1 < x < 1, y^2 + x^2 - 1 = 0, y > 0],$$
$$[-1 < x < 1, y^2 + x^2 - 1 < 0],$$
$$[-1 < x < 1, y^2 + x^2 - 1 = 0, y < 0],$$
$$[-1 < x < 1, y^2 + x^2 - 1 < 0],$$
$$[x = 1, y < 0], [x = 1, y = 0], [x = 1, y > 0],$$
$$[x > 1, y = y]$$
A CAD of $\mathbb{R}^2$ is given by the following collections of 13 cells:

- $x < -1$ \{ $[x < -1, y = y]$, 
- $x = -1$ \{ $[x = -1, y < 0], [x = -1, y = 0], [x = -1, y > 0]$, 
  \{ $[-1 < x < 1, y^2 + x^2 - 1 > 0, y > 0]$, 
  \{ $[-1 < x < 1, y^2 + x^2 - 1 = 0, y > 0]$, 
- $-1 < x < 1$ \{ $[-1 < x < 1, y^2 + x^2 - 1 < 0]$, 
  \{ $[-1 < x < 1, y^2 + x^2 - 1 = 0, y < 0]$, 
  \{ $[-1 < x < 1, y^2 + x^2 - 1 < 0, y < 0]$, 
- $x = 1$ \{ $[x = 1, y < 0], [x = 1, y = 0], [x = 1, y > 0]$, 
- $x > 1$ \{ $[x > 1, y = y]$
Sign-invariance

Traditionally a CAD is produced from a set of polynomials such that each polynomial has constant sign (positive, zero or negative) in each cell. Such a CAD is said to be **sign-invariant**.

The example from the previous slide was a sign-invariant CAD for the polynomial $x^2 + y^2 - 1$. 
Sign-invariance

Traditionally a CAD is produced from a set of polynomials such that each polynomial has constant sign (positive, zero or negative) in each cell. Such a CAD is said to be sign-invariant.

The example from the previous slide was a sign-invariant CAD for the polynomial $x^2 + y^2 - 1$. 
Traditionally a CAD is produced from a set of polynomials such that each polynomial has constant sign (positive, zero or negative) in each cell. Such a CAD is said to be sign-invariant.

The example from the previous slide was a sign-invariant CAD for the polynomial $x^2 + y^2 - 1$. 
Traditionally a CAD is produced from a set of polynomials such that each polynomial has constant sign (positive, zero or negative) in each cell. Such a CAD is said to be sign-invariant.

The example from the previous slide was a sign-invariant CAD for the polynomial $x^2 + y^2 - 1$. 
Traditionally a CAD is produced from a set of polynomials such that each polynomial has constant sign (positive, zero or negative) in each cell. Such a CAD is said to be sign-invariant.

The example from the previous slide was a sign-invariant CAD for the polynomial $x^2 + y^2 - 1$. 
Traditionally a CAD is produced from a set of polynomials such that each polynomial has constant sign (positive, zero or negative) in each cell. Such a CAD is said to be sign-invariant.

The example from the previous slide was a sign-invariant CAD for the polynomial $x^2 + y^2 - 1$.

Sign-invariance means we need only test one sample point per cell to determine behaviour of the polynomials. Various applications: quantifier elimination, optimisation, theorem proving, ...
1. Background
   - What is a CAD?
   - How do you build a CAD?

2. Truth table invariance
   - TTICAD
   - TTICAD via complex cylindrical decompositions
Collins gave the first algorithm to build CADs. It had 2 stages:

- **Projection**: to derive a set of polynomials from the input which can define the decomposition.
Collins gave the first algorithm to build CADs. It had 2 stages:

- **Projection**: to derive a set of polynomials from the input which can define the decomposition
- **Lifting** to incrementally build CADs by dimension.
Collins gave the first algorithm to build CADs. It had 2 stages:

- **Projection**: to derive a set of polynomials from the input which can define the decomposition
- **Lifting** to incrementally build CADs by dimension.
Collins gave the first algorithm to build CADs. It had 2 stages:

- **Projection**: to derive a set of polynomials from the input which can define the decomposition
- **Lifting** to incrementally build CADs by dimension.
Collins gave the first algorithm to build CADs. It had 2 stages:

- **Projection**: to derive a set of polynomials from the input which can define the decomposition
- **Lifting**: to incrementally build CADs by dimension.

![Graphical representation of a CAD]

Matthew England

Truth table invariant CAD
Collins gave the first algorithm to build CADs. It had 2 stages:

- **Projection**: to derive a set of polynomials from the input which can define the decomposition
- **Lifting** to incrementally build CADs by dimension.
Collins gave the first algorithm to build CADs. It had 2 stages:
- **Projection**: to derive a set of polynomials from the input which can define the decomposition
- **Lifting** to incrementally build CADs by dimension.
The nature of CAD means it will always have **doubly exponential complexity** in the number of variables. However:

1. It is still the best known (generic) algorithm for quantifier elimination.
The nature of CAD means it will always have **doubly exponential complexity** in the number of variables. However:

1. It is still the best known (generic) algorithm for quantifier elimination.
2. There have been many improvements and extensions to Collins algorithm which make a big difference in practice.
   - Improvements to the sub-algorithms used;
   - New projection operators;
   - CAD tailored to specific problems;
   - Results and algorithms on the adjacency of CAD cells;
   - Symbolic-numeric computation in the lifting phase.
The nature of CAD means it will always have **doubly exponential complexity** in the number of variables. However:

1. It is still the best known (generic) algorithm for quantifier elimination.
2. There have been many improvements and extensions to Collins algorithm which make a big difference in practice.
   - Improvements to the sub-algorithms used;
   - New projection operators;
   - CAD tailored to specific problems;
   - Results and algorithms on the adjacency of CAD cells;
   - Symbolic-numeric computation in the lifting phase.
Outline

1. Background
   - What is a CAD?
   - How do you build a CAD?

2. Truth table invariance
   - TTICAD
   - TTICAD via complex cylindrical decompositions
A CAD is truth-invariant with respect to a formula if the formula has constant truth value on each cell. Sign-invariant (for polynomials in formula) $\Rightarrow$ truth invariant but will usually require far more cells.
A CAD is truth-invariant with respect to a formula if the formula has constant truth value on each cell. Sign-invariant (for polynomials in formula) $\rightarrow$ truth invariant but will usually require far more cells.

- First introduced by Brown to simplify sign-invariant CADs;
- In 1999 McCallum developed a projection operator to use in the presence of an equational constraint (EC): an equation logically implied by a formula. The CAD produced was sign-invariant for the polynomial defining the EC, and for any others only when the EC is satisfied.
Given a sequence of quantifier free formulae (QFF) we define a truth table invariant CAD (TTICAD) as a CAD such that each formulae has constant truth value on each cell.

The Bath team together with Scott McCallum defined and verified a new projection operator to build TTICADs. First in the case where each QFF had an EC (ISSAC 2013) and now for arbitrary QFFs (submitted 2014).
Truth-table invariant CAD

Given a sequence of quantifier free formulae (QFF) we define a truth table invariant CAD (TTICAD) as a CAD such that each formulae has constant truth value on each cell.

The Bath team together with Scott McCallum defined and verified a new projection operator to build TTICADs. First in the case where each QFF had an EC (ISSAC 2013) and now for arbitrary QFFs (submitted 2014).

- Implemented in Maple (and available for free from authors);
- Generally far more efficient than a sign-invariant CAD (savings if at least one EC present);
- Output closer to the underlying application;
- Competitive with state of the art in CAD.
Why build a TTICAD?

A TTICAD can be useful for:

- An application providing a sequence of separate formulae

One such application: decomposing complex space according to the branch cuts of multi-valued functions for the purposes of algebraic simplification.
Why build a TTICAD?

A TTICAD can be useful for:

- **An application providing a sequence of separate formulae**

One such application: decomposing complex space according to the branch cuts of multi-valued functions for the purposes of algebraic simplification.

- **Finding a truth-invariant CAD for a parent formula**

A TTICAD for the defining sub-formula is truth-invariant for the parent. TTICAD can be the most efficient known approach (especially if there is no EC for the parent formula).
Consider $\sqrt{z^2 - 1}\sqrt{z^2 + 1} = \sqrt{z^4 - 1}$. Most software takes $\sqrt{\cdot}$ to be the positive root, in which case the identity is not always true.

The functions involved have branch cuts:

\begin{align*}
\varphi_1 &:= 2xy = 0 \land x^2 \mathbin{-} y^2 < 1, \\
\varphi_2 &:= 2xy = 0 \land x^2 \mathbin{-} y^2 < -1, \\
\varphi_3 &:= 4x^3y - 4xy^3 = 0 \land x^4 \mathbin{-} 6x^2y^2 + y^4 < 1.
\end{align*}

Either a TTICAD for $\{\varphi_1, \varphi_2, \varphi_3\}$ or a sign-invariant CAD for the polynomials involved would decompose $\mathbb{R}^2$ according to these cuts. We then need to test the truth at a finite number of sample points.
TTICAD: Algebraic Simplification Example II
Consider the polynomials

\[ f_1 := x^2 + y^2 - 1 \quad \quad g_1 := xy - \frac{1}{4} \]
\[ f_2 := (x - 4)^2 + (y - 1)^2 - 1 \quad \quad g_2 := (x - 4)(y - 1) - \frac{1}{4} \]

forming the single formula

\[ \Phi := (f_1 = 0 \land g_1 < 0) \lor (f_2 = 0 \land g_2 < 0) \]

We aim to determine where \( \Phi \) is true. A sign-invariant CAD for \( \{f_1, g_1, f_2, g_2\} \) has 317 cells but a TTICAD only 105.
TTICAD: Disjunction Example II

Graphs of the polynomials
The sign-invariant CAD identifies 20 points on $\mathbb{R}$.
The TTICAD identifies only 12 points on $\mathbb{R}$. 
In this example $\Phi$ does have an (implicit) EC: $f_1 f_2 = 0$. Using this alone produces more cells than a TTICAD and 16 points on $\mathbb{R}$. 
In this example $\Phi$ does have an (implicit) EC: $f_1 f_2 = 0$. Using this alone produces more cells than a TTICAD and 16 points on $\mathbb{R}$. 
TTICAD: Disjunction Example IV

All three approaches together.
TTICAD: Disjunction Example IV

TTICAD only
Outline

1. Background
   - What is a CAD?
   - How do you build a CAD?

2. Truth table invariance
   - TTICAD
   - TTICAD via complex cylindrical decompositions
< 2009 All CAD research broadly within Collin’s projection and lifting framework.
< 2009: All CAD research broadly within Collin’s projection and lifting framework.

ISSAC 2009: Chen, Moreno Maza, Xia & Yang give new approach.
From RC-CAD to RC-TTICAD

CCDs calculated using triangular decomposition by regular chains:

- **ISSAC 2009**: Uses existing algorithms
- **ASCM 2012**: Using custom built algorithms

The latter algorithms work *incrementally*: refining a tree one polynomial at a time.
CCDs calculated using triangular decomposition by regular chains:

**ISSAC 2009:** Uses existing algorithms

**ASCM 2012:** Using custom built algorithms

The latter algorithms work incrementally: refining a tree one polynomial at a time. Allows for the implementation of simplification in the presence of equational constraints: If an EC is not satisfied on a branch then no need for further refinement.

**CASC 2014:** Algorithm presented to produce TTICADs in the RC-CAD framework through incremental triangular decomposition $= \text{RC-TTICAD}$.

Combines the advantages of TTICAD with those from proceeding via complex cylindrical decompositions.
Further Information


*Cylindrical algebraic decompositions for Boolean combinations*


*Truth table invariant cylindrical algebraic decomposition by Regular Chains*

Contact Details

M. England@bath.ac.uk
http://www.cs.bath.ac.uk/~me350/
RC-CAD starts with a complex cylindrical decomposition (CCD). The tree below represents a sign-invariant CCD for \( p := x^2 + bx + c \) under variable ordering \( c \prec b \prec x \).
RC-CAD starts with a complex cylindrical decomposition (CCD). The tree below represents a sign-invariant CCD for \( p := x^2 + bx + c \) under variable ordering \( c \prec b \prec x \).

The key advantage is case distinction: the polynomial \( b \) is not sign-invariant for the whole decomposition, only when required.