

Cylindrical Algebraic Coverings: An SC-Square Success Story

Matthew England (Coventry University, UK)

Johannes Kepler University Linz
Institute of Algebra Seminar Series
20th October 2022

Supported by EPSRC EP/T015748/1 (DEWCAD) and EU H2020 712689 (SC²).

Summary in One Slide

(Slide 1/47)

I will describe a new algorithm which uses Cylindrical Algebraic Coverings (CACs) to decide whether a set of non-linear real polynomial constraints can be satisfied.

The algorithm reformulated classical idea from computer algebra into a new presentation suitable for use in SMT-solvers and inspired by other algorithms they use.

Why Care?

- The implementation in CVC5 won at the 2022 SMT Competition (QFNRA Track) and work is ongoing to extend the approach to tackle more problems.
- The over-arching idea of combining Satisfiability Checking and Symbolic Computation has found success in several domains - could it have potential in your work?

Acknowledgements and References

(Slide 2/47)

This is joint work with Erika Ábrahám, James H. Davenport and Gereon Kremer, which began in 2018 and was published in the following paper.



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, Elsevier, 2021.

<https://doi.org/10.1145/2755996.2756654>

- 1 Background on CAD
 - Definition
 - Applications of CAD
 - Complexity and Implementations
- 2 SC-Square and CAD
 - Satisfiability Checking
 - SC-Square
 - CAD for SMT
- 3 Cylindrical Algebraic Coverings
 - MCSAT
 - CDCAC
 - Future Outlook

Outline

- 1 Background on CAD
 - Definition
 - Applications of CAD
 - Complexity and Implementations
- 2 SC-Square and CAD
 - Satisfiability Checking
 - SC-Square
 - CAD for SMT
- 3 Cylindrical Algebraic Coverings
 - MCSAT
 - CDCAC
 - Future Outlook

Cylindrical Algebraic Decomposition I

(Slide 4/47)

A **Cylindrical Algebraic Decomposition (CAD)** is a mathematical object with the following properties:

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

Cylindrical Algebraic Decomposition II

(Slide 5/47)

CAD may also refer to an algorithm that produces the CAD object.

The traditional CAD algorithm introduced by Collins in the 1970s takes a set of input polynomials and produces a CAD such that each polynomial has constant sign in each cell: this additional property is called **sign-invariance**.

Such a CAD allows us to uncover properties of polynomials over infinite space by examining finite set of sample points.

Cylindrical Algebraic Decomposition II

(Slide 5/47)

CAD may also refer to an algorithm that produces the CAD object.

The traditional CAD algorithm introduced by Collins in the 1970s takes a set of input polynomials and produces a CAD such that each polynomial has constant sign in each cell: this additional property is called **sign-invariance**.

Such a CAD allows us to uncover properties of polynomials over infinite space by examining finite set of sample points.

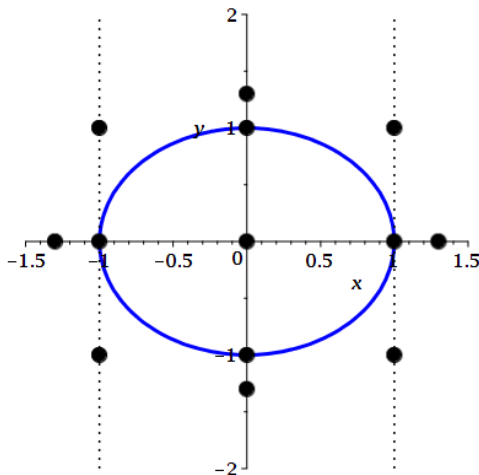
Most applications actually provide as input a Tarski formula: **logical formulae built from polynomial constraints**. They require as output a **truth-invariant CAD**: one such that each formula has **constant truth value** in each cell.

Such a CAD allows us to find solution sets from the descriptions of true cells: semi-algebraic; easy to visualise and check membership.

Example: Circle – visualisation

(Slide 6/47)

- 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

(Slide 6/47)

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

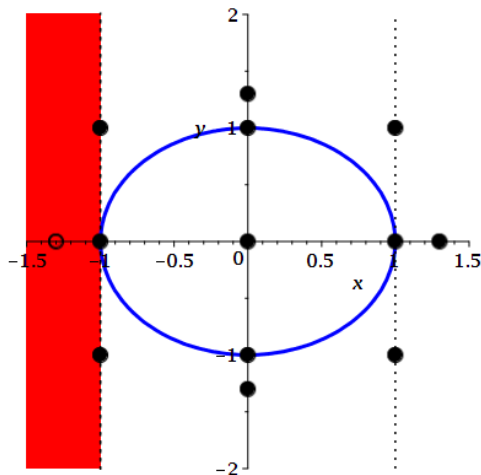
Cell 9: $-1 < x < 1,$
 $y^2 + x^2 - 1 > 0, y > 0$

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

(Slide 6/47)

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

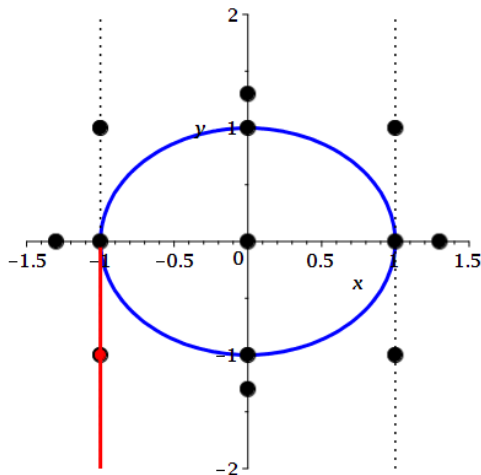
Cell 9: $-1 < x < 1,$
 $y^2 + x^2 - 1 > 0, y > 0$

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

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

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

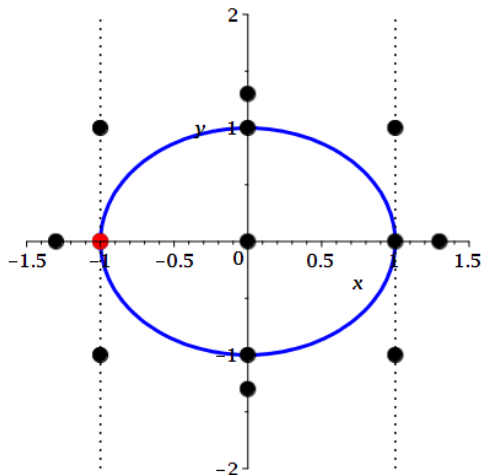
Cell 13: $x > 1, y$ free



Example: Circle – visualisation

(Slide 6/47)

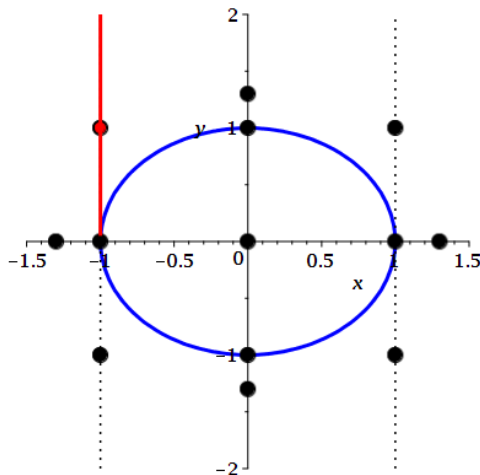
- Cell 1: $x < -1, y$ free
- Cell 2: $x = -1, y < 0$
- Cell 3: $x = -1, y = 0$
- Cell 4: $x = -1, y > 0$
- Cell 5: $-1 < x < 1,$
 $y^2 + x^2 - 1 > 0, y < 0$
- Cell 6: $-1 < x < 1,$
 $y^2 + x^2 - 1 = 0, y < 0$
- Cell 7: $-1 < x < 1,$
 $y^2 + x^2 - 1 < 0$
- Cell 8: $-1 < x < 1,$
 $y^2 + x^2 - 1 = 0, y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

(Slide 6/47)

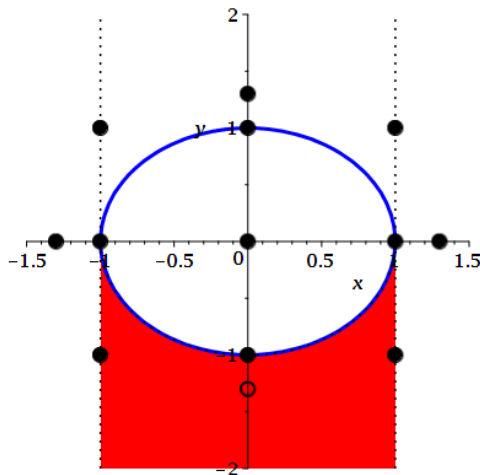
- Cell 1: $x < -1, y$ free
- Cell 2: $x = -1, y < 0$
- Cell 3: $x = -1, y = 0$
- Cell 4: $x = -1, y > 0$
- Cell 5: $-1 < x < 1,$
 $y^2 + x^2 - 1 > 0, y < 0$
- Cell 6: $-1 < x < 1,$
 $y^2 + x^2 - 1 = 0, y < 0$
- Cell 7: $-1 < x < 1,$
 $y^2 + x^2 - 1 < 0$
- Cell 8: $-1 < x < 1,$
 $y^2 + x^2 - 1 = 0, y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

(Slide 6/47)

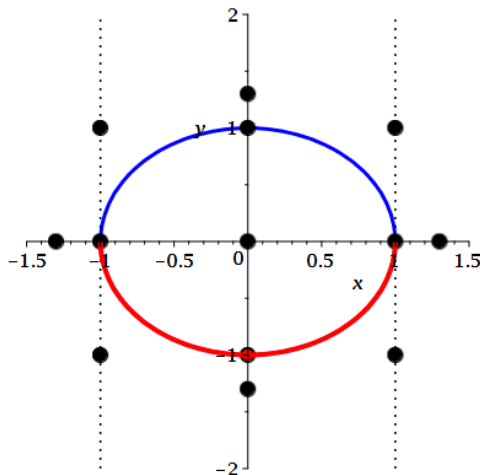
- Cell 1: $x < -1, y$ free
- Cell 2: $x = -1, y < 0$
- Cell 3: $x = -1, y = 0$
- Cell 4: $x = -1, y > 0$
- Cell 5: $-1 < x < 1,$
 $y^2 + x^2 - 1 > 0, y < 0$
- Cell 6: $-1 < x < 1,$
 $y^2 + x^2 - 1 = 0, y < 0$
- Cell 7: $-1 < x < 1,$
 $y^2 + x^2 - 1 < 0$
- Cell 8: $-1 < x < 1,$
 $y^2 + x^2 - 1 = 0, y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

(Slide 6/47)

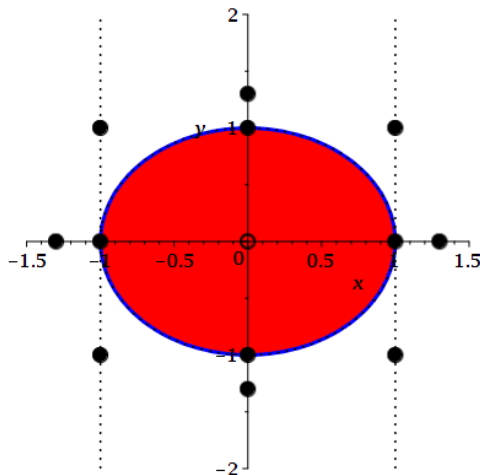
- Cell 1: $x < -1, y$ free
- Cell 2: $x = -1, y < 0$
- Cell 3: $x = -1, y = 0$
- Cell 4: $x = -1, y > 0$
- Cell 5: $-1 < x < 1,$
 $y^2 + x^2 - 1 > 0, y < 0$
- Cell 6: $-1 < x < 1,$
 $y^2 + x^2 - 1 = 0, y < 0$
- Cell 7: $-1 < x < 1,$
 $y^2 + x^2 - 1 < 0$
- Cell 8: $-1 < x < 1,$
 $y^2 + x^2 - 1 = 0, y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

(Slide 6/47)

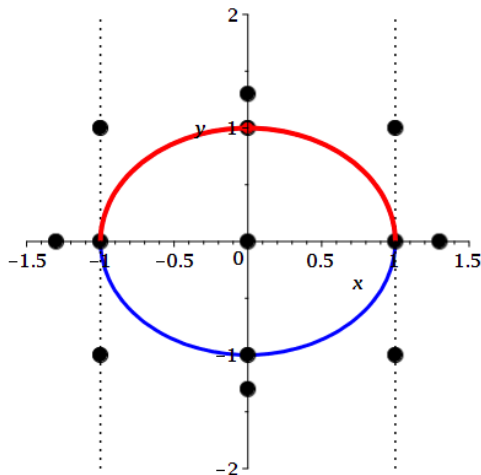
- Cell 1: $x < -1, y$ free
- Cell 2: $x = -1, y < 0$
- Cell 3: $x = -1, y = 0$
- Cell 4: $x = -1, y > 0$
- Cell 5: $-1 < x < 1,$
 $y^2 + x^2 - 1 > 0, y < 0$
- Cell 6: $-1 < x < 1,$
 $y^2 + x^2 - 1 = 0, y < 0$
- Cell 7: $-1 < x < 1,$
 $y^2 + x^2 - 1 < 0$
- Cell 8: $-1 < x < 1,$
 $y^2 + x^2 - 1 = 0, y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

(Slide 6/47)

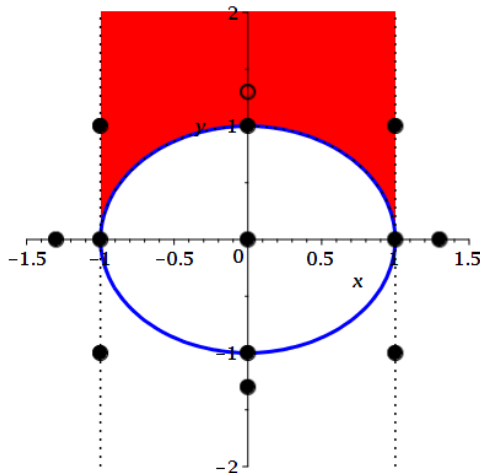
- Cell 1: $x < -1, y$ free
- Cell 2: $x = -1, y < 0$
- Cell 3: $x = -1, y = 0$
- Cell 4: $x = -1, y > 0$
- Cell 5: $-1 < x < 1,$
 $y^2 + x^2 - 1 > 0, y < 0$
- Cell 6: $-1 < x < 1,$
 $y^2 + x^2 - 1 = 0, y < 0$
- Cell 7: $-1 < x < 1,$
 $y^2 + x^2 - 1 < 0$
- Cell 8: $-1 < x < 1,$
 $y^2 + x^2 - 1 = 0, y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

(Slide 6/47)

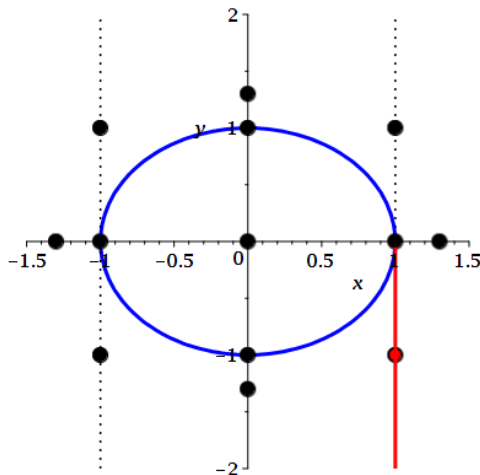
- Cell 1: $x < -1, y$ free
- Cell 2: $x = -1, y < 0$
- Cell 3: $x = -1, y = 0$
- Cell 4: $x = -1, y > 0$
- Cell 5: $-1 < x < 1,$
 $y^2 + x^2 - 1 > 0, y < 0$
- Cell 6: $-1 < x < 1,$
 $y^2 + x^2 - 1 = 0, y < 0$
- Cell 7: $-1 < x < 1,$
 $y^2 + x^2 - 1 < 0$
- Cell 8: $-1 < x < 1,$
 $y^2 + x^2 - 1 = 0, y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

(Slide 6/47)

- 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

(Slide 6/47)

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

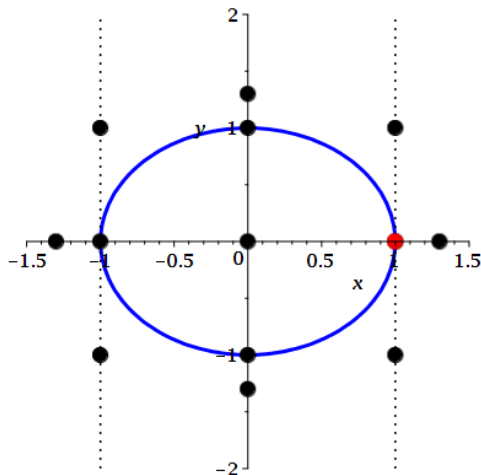
Cell 9: $-1 < x < 1,$
 $y^2 + x^2 - 1 > 0, y > 0$

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

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

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

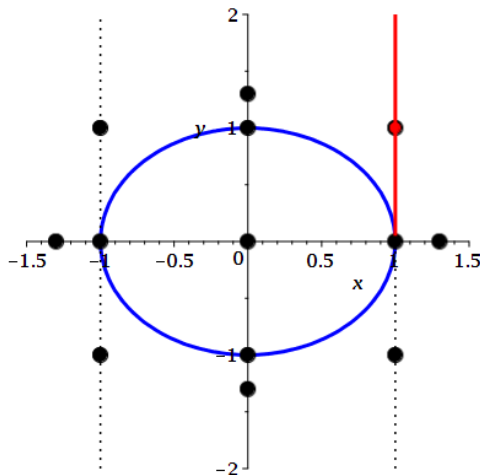
Cell 13: $x > 1, y$ free



Example: Circle – visualisation

(Slide 6/47)

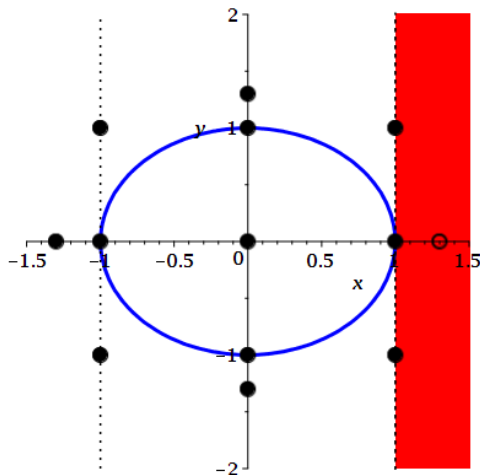
- Cell 1: $x < -1, y$ free
- Cell 2: $x = -1, y < 0$
- Cell 3: $x = -1, y = 0$
- Cell 4: $x = -1, y > 0$
- Cell 5: $-1 < x < 1,$
 $y^2 + x^2 - 1 > 0, y < 0$
- Cell 6: $-1 < x < 1,$
 $y^2 + x^2 - 1 = 0, y < 0$
- Cell 7: $-1 < x < 1,$
 $y^2 + x^2 - 1 < 0$
- Cell 8: $-1 < x < 1,$
 $y^2 + x^2 - 1 = 0, y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

(Slide 6/47)

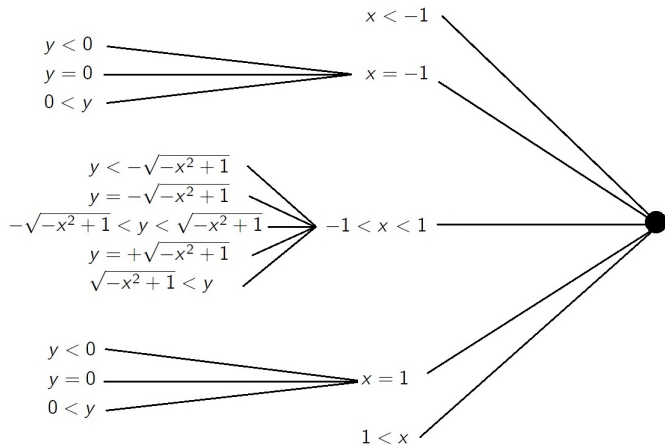
- Cell 1: $x < -1, y$ free
- Cell 2: $x = -1, y < 0$
- Cell 3: $x = -1, y = 0$
- Cell 4: $x = -1, y > 0$
- Cell 5: $-1 < x < 1,$
 $y^2 + x^2 - 1 > 0, y < 0$
- Cell 6: $-1 < x < 1,$
 $y^2 + x^2 - 1 = 0, y < 0$
- Cell 7: $-1 < x < 1,$
 $y^2 + x^2 - 1 < 0$
- Cell 8: $-1 < x < 1,$
 $y^2 + x^2 - 1 = 0, y > 0$
- 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 – cylindrical tree

(Slide 7/47)

The cylindricity means we can think of CAD as a tree branching by variable restriction. For the circle:



How to build a CAD?

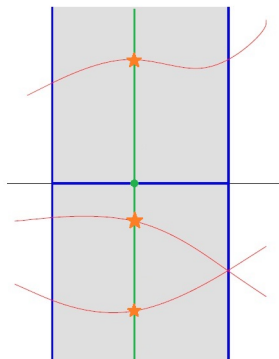
(Slide 8/47)

Two steps in usual approach:

projection which uses algebraic operations to create a finite set of polynomials whose roots indicate changes in the behaviour of the input set; then

lifting which systematically applies real root evaluation on these to count roots and generate cells in cylinders.

Real root evaluation is performed on univariate polynomials obtained by substituting a sample point for the cell being lifted over. **Is this safe to do?**



Delineability

(Slide 9/47)

The projection operation needs to be defined so working at a sample point is representative of the cell: **delineability**.

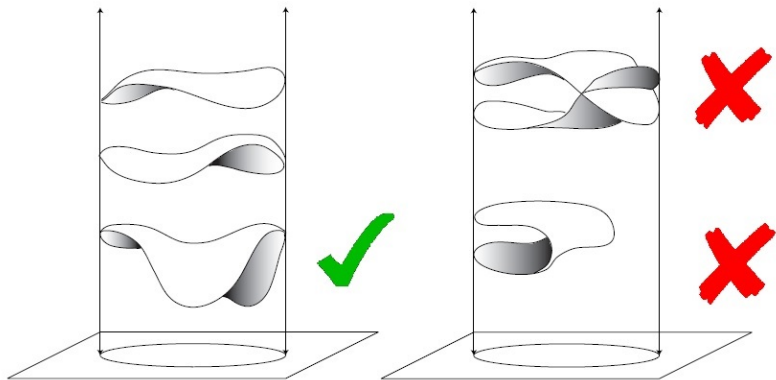


Image from Chris Brown's 2017 SC-Square Summer School Lecture

Classic CAD References

(Slide 10/47)



[Col75] G.E. Collins.

Quantifier elimination for real closed fields by cylindrical algebraic decomposition.

Proc. 2nd GI Conference on Automata Theory and Formal Languages, pages 134 – 183. Springer-Verlag, 1975.

Reprinted in [CJ98].



[CJ98] B. Caviness and J. Johnson.

Quantifier elimination and cylindrical algebraic decomposition.

Texts & Monographs in Symbolic Computation.

Springer-Verlag, 1998

The former is the original CAD paper of Collins. The latter is a book containing that and many paper on CAD improvements and extensions in the next 20 years.

Algebraic numbers and root expressions

(Slide 11/47)

CAD takes polynomials with coefficients in \mathbb{Q} as input, but their roots may be algebraic numbers defined by root expressions, e.g.:

$$\sqrt{2} = \text{RootOf}(x^2 - 2, x, \text{index} = \text{real}[2]) = \text{RootOf}(x^2 - 2, x, 1..2)$$

Semi-algebraic formula for blue region:

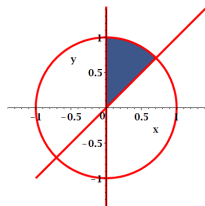
$$x^2 + y^2 - 1 < 0 \wedge y - x > 0 \wedge x > 0$$

Cylindrical formula for blue region:

$$0 < x < \text{RootOf}(2x^2 - 1, x, \text{index} = \text{real}[2])$$

$$x < y < \text{RootOf}(y^2 + x^2 - 1, y, \text{index} = \text{real}[2])$$

Projection and lifting naturally produces cylindrical formulae, extra work is needed for semi-algebraic formulae.



Outline

- 1 Background on CAD
 - Definition
 - Applications of CAD
 - Complexity and Implementations
- 2 SC-Square and CAD
 - Satisfiability Checking
 - SC-Square
 - CAD for SMT
- 3 Cylindrical Algebraic Coverings
 - MCSAT
 - CDCAC
 - Future Outlook

Real Quantifier Elimination

(Slide 12/47)

Real Quantifier Elimination (Real QE)

Given: Quantified formulae in prenex form with atoms integral polynomial constraints.

Produce: a quantifier free formula logically equivalent over \mathbb{R} .

Fully quantified examples:

Input: $\forall x, x^2 + 1 \leq 0$

Output: False

Input: $\exists x, x^2 + 3x + 1 \leq 0$

Output: True

Partially quantified example:

Input: $\exists x, x^2 + bx + 1 \leq 0$

Output: $(b \leq -2) \vee (b > 2)$

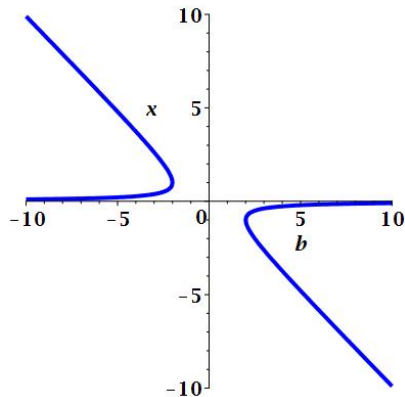
When partially quantified the answer depends on the free (unquantified) variables.

QE via CAD Example

(Slide 13/47)

Existential QE is via projection of true CAD cells onto free variables. Universal QE is via $\forall x F(x) = \neg \exists x \neg F(x)$. For example:

$$\exists x, x^2 + bx + 1 \leq 0$$



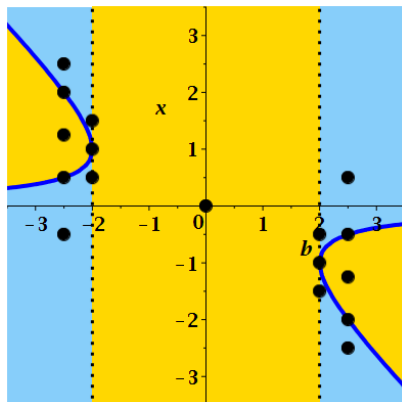
QE via CAD Example

(Slide 13/47)

Existential QE is via projection of true CAD cells onto free variables. Universal QE is via $\forall x F(x) = \neg \exists x \neg F(x)$. For example:

$$\exists x, x^2 + bx + 1 \leq 0$$

Build a sign-invariant CAD for
 $f = x^2 + bx + 1$.



QE via CAD Example

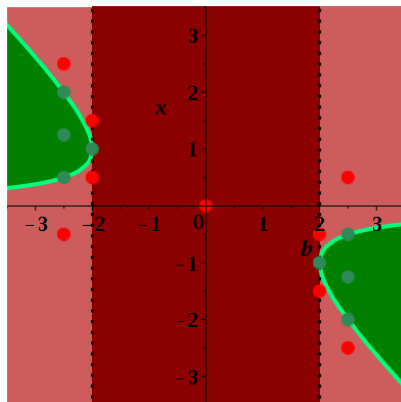
(Slide 13/47)

Existential QE is via projection of true CAD cells onto free variables. Universal QE is via $\forall x F(x) = \neg \exists x \neg F(x)$. For example:

$$\exists x, x^2 + bx + 1 \leq 0$$

Build a sign-invariant CAD for
 $f = x^2 + bx + 1$.

Tag each cell true or false
according to $f \leq 0$ at sample.



QE via CAD Example

(Slide 13/47)

Existential QE is via projection of true CAD cells onto free variables. Universal QE is via $\forall x F(x) = \neg \exists x \neg F(x)$. For example:

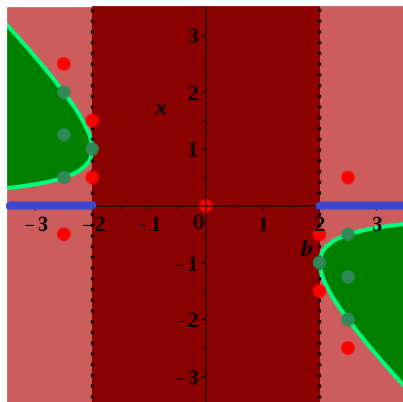
$$\exists x, x^2 + bx + 1 \leq 0$$

Build a sign-invariant CAD for
 $f = x^2 + bx + 1$.

Tag each cell true or false
according to $f \leq 0$ at sample.

Take disjunction of projections of
the true cells:

$$b < -2 \vee b = -2 \\ \vee b = 2 \vee b > 2$$



QE via CAD Example

(Slide 13/47)

Existential QE is via projection of true CAD cells onto free variables. Universal QE is via $\forall x F(x) = \neg \exists x \neg F(x)$. For example:

$$\exists x, x^2 + bx + 1 \leq 0$$

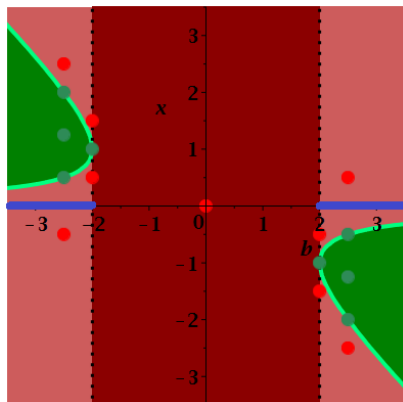
Build a sign-invariant CAD for
 $f = x^2 + bx + 1$.

Tag each cell true or false
according to $f \leq 0$ at sample.

Take disjunction of projections of
the true cells:

$$\implies$$

$$b \leq -2 \vee b \geq 2$$



Important Sidenote: Variable Ordering

(Slide 14/47)

The cylindricity condition, and thus CAD algorithms, are defined with respect to a **variable ordering**.

I.e. $x_1 \prec \dots \prec x_n$ implies we consider the sequence of projections onto (x_1, \dots, x_k) for k from 1 to $n - 1$.

Important Sidenote: Variable Ordering

(Slide 14/47)

The cylindricity condition, and thus CAD algorithms, are defined with respect to a **variable ordering**.

I.e. $x_1 \prec \dots \prec x_n$ implies we consider the sequence of projections onto (x_1, \dots, x_k) for k from 1 to $n - 1$.

The variable ordering may be restricted or free depending on the problem at hand. For QE:

- We need to be able to project cells onto the unquantified variables, so these must appear higher in the ordering.
- The variables must be ordered as in the quantification;
- but we can swap adjacent variables if they are quantified by the same quantifier (or none at all).

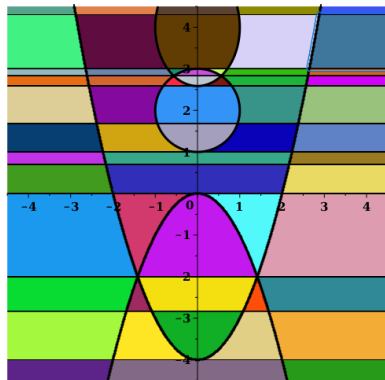
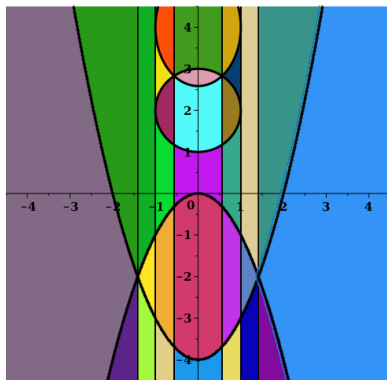
This leaves a degree of choice for CAD. Can have a *big* effect on the coarseness of the decomposition and computation time.

Example of variable ordering importance

(Slide 15/47)

Sign invariant CADs for the set of polynomials

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

Projecting onto x requires 105 cells, but onto y requires 215.

Other applications of CAD

(Slide 16/47)




QE has applications throughout engineering & science. E.g.

- derivation of optimal numerical schemes (Erascu-Hong, 2016);
- automatically proving inequalities from combinatorics (Gerhold and Kauers, 2005);
- automated theorem proving (Paulson, 2012);
- automated loop parellisation (Grösslinger et al. 2006);
- analysis of economic hypotheses (Mulligan et al., 2018).

CAD has also been applied independently of QE, e.g.
multi-stationarity identification in chemical reaction networks (Bradford et al. 2017).

Application References

(Slide 17/47)

-  R. Bradford, J.H. Davenport, M.England, H. Errami, V.Gerdt, D.Grigoriev, C.Hoyt, M.Kosta, 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.
-  M. Erascu and H. Hong.
Real Quantifier elimination for the synthesis of optimal numerical algorithms (Case study: Square root computation).
Journal of Symbolic Computation, 75:110–126, 2016.
-  A. Grosslinger, M. Griehl, and C. Lengauer.
Quantifier elimination in automatic loop parallelization.
Journal of Symbolic Computation, 41(11):1206–1221, 2006.

Application References cont.

(Slide 17/47)



S. Gerhold and M. Kauers.

A procedure for proving special function inequalities involving a discrete parameter.

Proc. ISSAC 2005, pages 156—162. ACM, 2005.



C.B. Mulligan, J.H. Davenport, and M. England.

TheoryGuru: A Mathematica Package to apply Quantifier Elimination

Proc. ICMS 2018, LNCS 10931, pages 369–378, Springer, 2018.



L.C. Paulson.

Metitarski: Past and future.

Interactive Theorem Proving (LNCS 7406), pages 1–10, Springer 2012.

Outline

- 1 **Background on CAD**
 - Definition
 - Applications of CAD
 - **Complexity and Implementations**
- 2 SC-Square and CAD
 - Satisfiability Checking
 - SC-Square
 - CAD for SMT
- 3 Cylindrical Algebraic Coverings
 - MCSAT
 - CDCAC
 - Future Outlook

CAD Complexity

(Slide 18/47)

To build a CAD we repeatedly project polynomials to encode key geometric information.

By the end of projection you could have doubly exponentially many polynomials of doubly exponential degree (in the number of projections, i.e. variables). Hence the number of real roots, cells, and time to compute them grows doubly exponentially too.



C. Brown and J.H. Davenport.

The complexity of quantifier elimination and cylindrical algebraic decomposition.

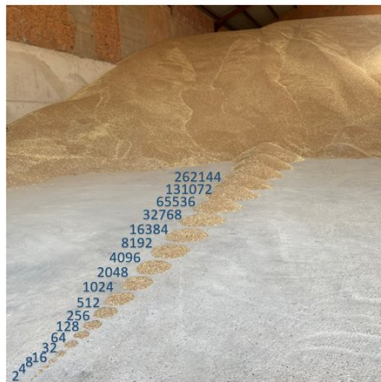
In Proc. ISSAC '07, pages 54–60. ACM, 2007.

Nevertheless, CAD remains the most used general purpose method for real QE.

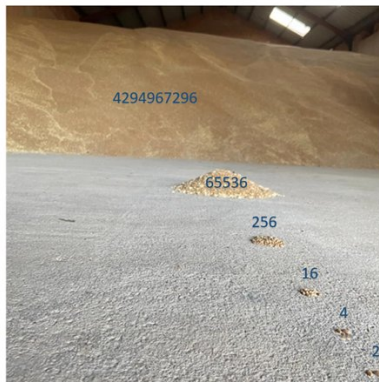
The Doubly Exponential Wall

(Slide 19/47)

Exponential Growth



Doubly Exponential Growth



Images by Tereso del Río

CAD Implementations

(Slide 20/47)

CAD implementations can be found in:

- Mathematica using `CylindricalDecomposition`
- Maple, under `RegularChains` :- `SemiAlgebraicSetTools`
:- `CylindricalAlgebraicDecompose`.
- Maple, in the third party `SyNRAC` package.
- Maple, in an upcoming `QuantifierElimination` package.
- Reduce, using the `Redlog` package command `rlcad`.
- QEPCAD-B and Tarski: Linux only, but now available online:
<https://matek.hu/tarski/tarski-jt.html>

Outline

- 1 Background on CAD
 - Definition
 - Applications of CAD
 - Complexity and Implementations
- 2 SC-Square and CAD
 - Satisfiability Checking
 - SC-Square
 - CAD for SMT
- 3 Cylindrical Algebraic Coverings
 - MCSAT
 - CDCAC
 - Future Outlook

Boolean SAT Problem

(Slide 21/47)

The **Boolean SAT Problem** is to decide if a given logical formula with Boolean valued variables is satisfiable, i.e. there exists an assignment of values to variables to make the formula true.

Examples:

$$F_1 = (x \vee y) \wedge (\neg x \vee \neg z)$$

F_1 is satisfied by $x = T, y = T, z = F$. Also by other assignments.

$$F_2 = (x \vee y) \wedge (\neg x \vee \neg y) \wedge (\neg x \vee y)$$

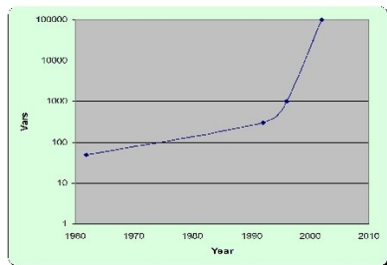
F_2 is unsatisfiable.

If the formula has n variables then there are 2^n possible assignments. The SAT problem is NP-Complete.

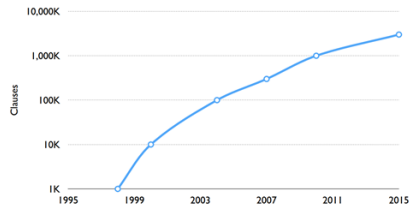
SAT-Solvers

(Slide 22/47)

SAT-solvers are tools dedicated to solving the SAT Problem.
They can routinely solve huge problem instances!



Graph from J. Marques-Silva and K.Sakallah

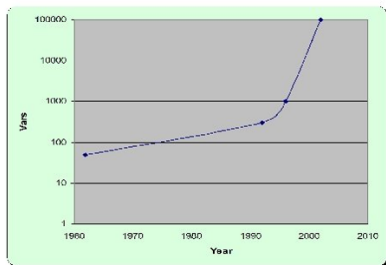


Graph from V. Ganesh

SAT-Solvers

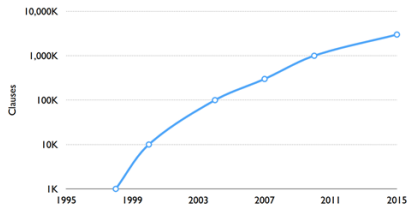
(Slide 22/47)

SAT-solvers are tools dedicated to solving the SAT Problem.
They can routinely solve huge problem instances!



Graph from J. Marques-Silva and K.Sakallah

SAT-solvers are now used routinely in industry, and increasingly in mathematics too.



Graph from V. Ganesh

Key SAT Algorithmic Advances

(Slide 23/47)

- In 1961 the DPLL Algorithm described how to use propagation and backtracking to process the search space efficiently.
- In 1996 the CDCL Algorithm described how growing the formula with clauses that rule out the infeasible combinations discovered can make future search more efficient.



M. Davis and H. Putnam.

A Computing Procedure for Quantification Theory.

J. ACM, 7(3), 201–215, 1960.



M. Davis, G. Logemann, and D. Loveland.

A Machine Program for Theorem Proving.

C. ACM, 5(7), 394–397, 1961.



J.P. Marques-Silva and K.A. Sakallah.

GRASP-A New Search Algorithm for Satisfiability.

Proc. ICCAD, 220-227. IEEE, 1996.

Key Lessons from SAT

(Slide 24/47)

- Problems encountered rarely exhibit the worst case complexity.
- Usually much easier to prove satisfiability (find an example) than to prove unsatisfiability.
- Try to rule out whole branches of search space at a time.

Key Lessons from SAT

(Slide 24/47)

- Problems encountered rarely exhibit the worst case complexity.
- Usually much easier to prove satisfiability (find an example) than to prove unsatisfiability.
- Try to rule out whole branches of search space at a time.

Note: Modern SAT solvers assume input in **Conjunctive Normal Form**: conjunctions of clauses, which are disjunctions of logical atoms. Why?

- Easy to analyse sub-problems (each clause must be satisfied).
- Easy to record information for later, e.g. add a clause that forces us to avoid searching a similar part of the search space.
- Electronic circuits can be transformed to CNF efficiently.

Satisfiability Modulo Theories

(Slide 25/47)

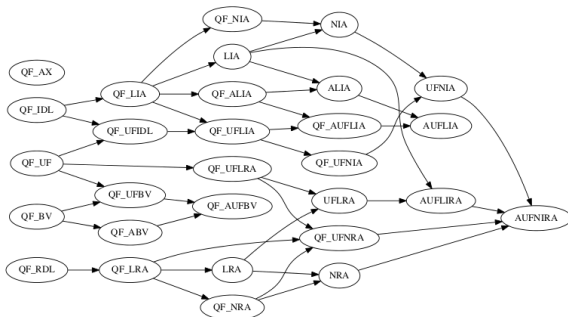
A **Satisfiability Modulo Theory (SMT) Problem** is a SAT Problem where the atoms are not Boolean variables, but statements that evaluate to a Boolean.

Satisfiability Modulo Theories

(Slide 25/47)

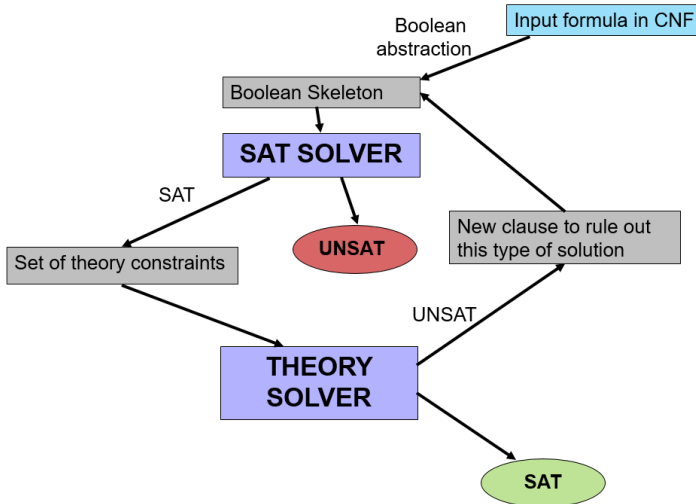
A **Satisfiability Modulo Theory (SMT) Problem** is a SAT Problem where the atoms are not Boolean variables, but statements that evaluate to a Boolean.

The statements exist in well defined *theories*.



Lazy SMT Framework

(Slide 26/47)



SMT Example

(Slide 27/47)

Consider the following problem in the QF_NRA theory.

$$R : (v^2 > 10 \vee v + 1 > 3) \wedge (v^2 \leq 10 \vee v = 1) \quad v \in \mathbb{R}$$

Let x be $v^2 > 10$, y be $v + 1 > 3$, and z be $v \neq 1$. Then we have Boolean skeleton

$$B : (x \vee y) \wedge (\neg x \vee \neg z)$$

SMT Example

(Slide 27/47)

Consider the following problem in the QF_NRA theory.

$$R : (v^2 > 10 \vee v + 1 > 3) \wedge (v^2 \leq 10 \vee v = 1) \quad v \in \mathbb{R}$$

Let x be $v^2 > 10$, y be $v + 1 > 3$, and z be $v \neq 1$. Then we have Boolean skeleton

$$B : (x \vee y) \wedge (\neg x \vee \neg z)$$

A SAT-solver may propose a solution, $x = T, y = T, z = F$ but that is not valid in the theory. We cannot have $x = T$ ($v^2 > 10$) and $z = F$ ($v = 1$) at the same time.

SMT Example

(Slide 27/47)

Consider the following problem in the QF_NRA theory.

$$R : (v^2 > 10 \vee v + 1 > 3) \wedge (v^2 \leq 10 \vee v = 1) \quad v \in \mathbb{R}$$

Let x be $v^2 > 10$, y be $v + 1 > 3$, and z be $v \neq 1$. Then we have Boolean skeleton

$$B : (x \vee y) \wedge (\neg x \vee \neg z)$$

A SAT-solver may propose a solution, $x = T, y = T, z = F$ but that is not valid in the theory. We cannot have $x = T$ ($v^2 > 10$) and $z = F$ ($v = 1$) at the same time. So add a clause to avoid it:

$$B' : (x \vee y) \wedge (\neg x \vee \neg z) \wedge (\neg x \vee z)$$

A SAT solver can now find $x = F, y = T, z = T$ which is acceptable in the theory (with $v = 3$).

Outline

- 1 Background on CAD
 - Definition
 - Applications of CAD
 - Complexity and Implementations
- 2 SC-Square and CAD
 - Satisfiability Checking
 - **SC-Square**
 - CAD for SMT
- 3 Cylindrical Algebraic Coverings
 - MCSAT
 - CDCAC
 - Future Outlook

SC-Square Definition

(Slide 28/47)

SC: Satisfiability Checking Community

Interested in algorithms to check satisfiability of logic problems with variables from a variety of mathematical domains. Implement solutions in SAT/SMT solvers.

SC: Symbolic Computation Community

Interested in algorithms to perform algebraic computations, such as polynomial computations over real and complex numbers. Implement solutions in Computer Algebra Systems.

SC²: SC-Square Community

Interested in both SCs above!

The EU SC² Project

(Slide 30/47)

The EU funded **The SC-Square Project** from 2016-2018. The aim was to bridge the gap between the communities to produce individuals who can combine the knowledge and techniques of both fields to resolve problems currently beyond the scope of either.

The project funded new collaborations, new tool integrations, proposals on extensions to the SMT-LIB language standards, new collections of benchmarks, the SC-Square Workshop Series, a special issue of JSC (vol 100), and two summer schools. Slides and videos from the first remain online here:

<http://www.sc-square.org/CSA/school/lectures.html>

Although the project finished, the workshop series continues, and the website and mailing lists remain active: www.sc-square.org

SC-Square Project Reference



E. Ábrahám, J. Abbott, B. Becker, A.M. Bigatti, M. Brain, B. Buchberger, A. Cimatti, J.H. Davenport, M. England, P. Fontaine, S. Forrest, A. Griggio, D. Kroening, W.M. Seiler, and T. Sturm.

SC²: Satisfiability checking meets symbolic computation.

In M. Kohlhase, M. Johansson, B. Miller, L. de Moura, and F. Tompa, editors, *Intelligent Computer Mathematics: Proceedings CICM 2016*, volume 9791 of *Lecture Notes in Computer Science*, pages 28–43. Springer International Publishing, 2016.

URL https://doi.org/10.1007/978-3-319-42547-4_3.

SC-Square Workshop Series

(Slide 31/47)

- 2016 Timisoara, Romainia (as part of SYNASC 2016).
- 2017 Kaiserslautern, Germany (alongside ISSAC 2017).
- 2018 Oxford, UK (as part of FLoC 2018).
- 2019 Bern, Switzzlerland (as part of SIAM AG19)
- 2020 Paris, France (online) (alongside IJCAR 2020)
- 2021 Texas, USA (online) (as part of SIAM AG21)
- 2022 Haifa, Israel (as part of FLoC 2022)
- 2023 Tromsø, Norway (alongside ISSAC 2023)

If you have work that involves both SC's then please consider submitting for a presentation in Tromsø!

SC-Square Successes I

(Slide 32/47)

Maple can now read to and from SMT-LIB (Forrest, 2017) and ships with the both the Z3 and MapleSAT.

CoCoALib C++ Library that underpins CoCoA (Abbott and Bigatti, 2014) used by MathSAT and SMT-RAT.

Incremental Linearization techniques developed in MathSAT for non-linear problems (Cimatti et al. 2018).

MathCheck has made significant progress on a variety of combinatorics problems from a combination of SAT-solvers and computer algebra, enumerating new cases and verifying conjectures. E.g. Williamson Matrices (Bright et al. 2020).

SC-Square Successes II

(Slide 33/47)

Boolean SAT has also benefited from symbolic computation via Boolean Groebner Bases and parallel computation on both the conjunctive and algebraic normal forms of a problem (Horáček and Kreuzer, 2020).

Circuit Verification Algebraic techniques key for circuit verification and in combination with SAT-solvers (Kaufmann, Biere, and Kauers, 2019).

New Applications in Economics (Mulligan et al., 2018), and Dynamic Geometry (Vajda and Kovács, 2020).

SC-Square Successes References I



S.A. Forrest.

Integration of SMT-LIB support into maple.

In M. England and V. Ganesh, editors, *Proc. (SC² 2017)*, number 1974 in CEUR-WS, 2017.

URL <http://ceur-ws.org/Vol-1974/>.



J. Abbott and A.M. Bigatti.

What is new in CoCoA?

In H. Hong and C. Yap, editors, *Mathematical Software – ICMS 2014*, LNCS 8592, pages 352–358. Springer Heidelberg, 2014.

URL https://doi.org/10.1007/978-3-662-44199-2_55.



A. Cimatti, A. Griggio, A. Irfan, M. Roveri, and R. Sebastiani.

Incremental linearization: A practical approach to satisfiability modulo nonlinear arithmetic and transcendental functions.

In *Proc. SYNASC 2018*, pages 19–26, IEEE, 2018.

URL <http://doi.org/10.1109/SYNASC.2018.00016>.

SC-Square Successes References II



C. Bright, I. Kotsireas, and V. Ganesh.

Applying computer algebra systems with SAT solvers to the Williamson conjecture.

Journal of Symbolic Computation, 100:187–209, 2020.

URL <https://doi.org/10.1016/j.jsc.2019.07.024>.



J. Horáček and M. Kreuzer.

On conversions from CNF to ANF.

Journal of Symbolic Computation, 100:164–186, 2020.

URL <https://doi.org/10.1016/j.jsc.2019.07.023>.



D. Kaufmann, A. Biere, and M. Kauers.

Verifying large multipliers by combining SAT and computer algebra.

In *Formal Methods in Computer Aided Design (FMCAD 2019)*, pages 28–36. IEEE, 2019.

URL <https://doi.org/10.23919/FMCAD.2019.8894250>.

SC-Square Successes References III



C. Mulligan, R. Bradford, J.H. Davenport, M. England, and Z. Tonks.
Non-linear real arithmetic benchmarks derived from automated reasoning
in economics.

In A.M. Bigatti and M. Brain, editors, *Proceedings of the 3rd Workshop
on Satisfiability Checking and Symbolic Computation (SC² 2018)*, number
2189 in CEUR Workshop Proceedings, pages 48–60, 2018.

URL <http://ceur-ws.org/Vol-2189/>.



R. Vajda and Z. Kovács.

GeoGebra and the realgeom reasoning tool.

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 (SC² 2020)*, number 2752 in CEUR Workshop
Proceedings, pages 204–219, 2020.

URL <http://ceur-ws.org/Vol-2752/>.

Outline

- 1 Background on CAD
 - Definition
 - Applications of CAD
 - Complexity and Implementations
- 2 SC-Square and CAD
 - Satisfiability Checking
 - SC-Square
 - CAD for SMT
- 3 Cylindrical Algebraic Coverings
 - MCSAT
 - CDCAC
 - Future Outlook

SC-Square for QF_NRA

(Slide 34/47)

In this case the theory solver must determine the **satisfiability of a formula in Non-linear Real Arithmetic (NRA)**. I.e. to evaluate

$$\exists x_1, \exists x_2, \dots, \exists x_n F(x_1, x_2, \dots, x_n)$$

as either True (SAT) or False (UNSAT). Here F is a CNF whose atoms are sign constraints on non-linear multivariate polynomials with integer coefficients.

SC-Square for QF_NRA

(Slide 34/47)

In this case the theory solver must determine the **satisfiability of a formula in Non-linear Real Arithmetic (NRA)**. I.e. to evaluate

$$\exists x_1, \exists x_2, \dots, \exists x_n F(x_1, x_2, \dots, x_n)$$

as either True (SAT) or False (UNSAT). Here F is a CNF whose atoms are sign constraints on non-linear multivariate polynomials with integer coefficients.

A sub-problem of Real QE. So one approach is to use a computer algebra system as theory solver. E.g. Redlog+VeriT:



P. Fontaine, M. Ogawa, T. Sturm, V. Khanh To, and X. Tung Vu.

Wrapping computer algebra is surprisingly successful for non-linear SMT.
In A.M. Bigatti and M. Brain, editors, *Proc. SC² 2018*, CEUR-WS 2189, pages 110–117, 2018.

URL <http://ceur-ws.org/Vol-2189/>.

CAD as NRA Theory Solver

(Slide 35/47)

To be efficient, SMT theory solvers should be adapted for:

- **Incrementality:** Add a constraint and divide cells.
- **Backtracking:** Remove a constraint and merge cells.
- **Explanations:** When no cell satisfies constraints identify minimal subset of constraints which are mutually unsatisfiable.

Such an adaptation was created in SMT-RAT:



G. Kremer and E. Ábrahám.

Fully incremental cylindrical algebraic decomposition.

J. of Symbolic Computation, 100, pages 11–37. Elsevier, 2020.

<https://doi.org/10.1016/j.jsc.2019.07.018>

CAD as NRA Theory Solver

(Slide 35/47)

To be efficient, SMT theory solvers should be adapted for:

- **Incrementality:** Add a constraint and divide cells.
- **Backtracking:** Remove a constraint and merge cells.
- **Explanations:** When no cell satisfies constraints identify minimal subset of constraints which are mutually unsatisfiable.

Such an adaptation was created in SMT-RAT:



G. Kremer and E. Ábrahám.

Fully incremental cylindrical algebraic decomposition.

J. of Symbolic Computation, 100, pages 11–37. Elsevier, 2020.

<https://doi.org/10.1016/j.jsc.2019.07.018>

(Q) Why is SAT solver + CAD better than CAD alone?

(A) Because in SMT a theory solver commonly only addresses a small subset of the total constraints at once.

Sidenote: SMT-RAT

(Slide 36/47)

The SMT-RAT Toolbox has adapted many computer algebra algorithms for use in CAD, including CAD, virtual substitution, branch and bound, Fourier Motzkin variable elimination, interval constraint propagation, Gröbner Bases, and more.



G. Kremer and E. Ábrahám.

Modular strategic SMT solving with SMT-RAT.

Acta Universitatis Sapientiae, Informatica, 10(1):5–25, 2018.

URL <http://dx.doi.org/10.2478/ausi-2018-0001>.

How good is such a CAD adaptation?

(Slide 37/47)

For problems where the solution is SAT this approach tends to determine the solution **much faster** than CAD alone as it can terminate earlier when a satisfying witness is discovered.

For UNSAT problems this approach can still be faster if it allows to reach the conclusion by studying multiple smaller problems; but it may still require the computation of some very large decompositions.

How good is such a CAD adaptation?

(Slide 37/47)

For problems where the solution is SAT this approach tends to determine the solution **much faster** than CAD alone as it can terminate earlier when a satisfying witness is discovered.

For UNSAT problems this approach can still be faster if it allows to reach the conclusion by studying multiple smaller problems; but it may still require the computation of some very large decompositions.

Can we adapt CAD further to avoid this?

Yes: by following approaches used by SAT-solvers which search the sample spaces by: making guesses, propagating, and generalising conflicts to avoid similar parts of the search space.

Outline

- 1 Background on CAD
 - Definition
 - Applications of CAD
 - Complexity and Implementations
- 2 SC-Square and CAD
 - Satisfiability Checking
 - SC-Square
 - CAD for SMT
- 3 Cylindrical Algebraic Coverings
 - MCSAT
 - CDCAC
 - Future Outlook

NLSAT / MCSAT

(Slide 38/47)

Jovanović and de Moura's algorithm for Microsoft's Z3 solver departs from the Lazy SMT framework for QF_NRA problems. Instead of the usual loop between SAT and theory solvers:

- partial model solutions for the Boolean skeleton and the algebraic theory are built in parallel;
- Boolean conflicts generalised using the CDCL approach;
- theory conflicts generalised through the creation of a single CAD cell: the cell around the theory model point where the polynomials involved in the conflict are sign invariant.

The learnt clause is the negation of the cell description.

Originally called the **NLSAT Algorithm**. Later the approach was generalised for use with other theories, and it is now called the **Model Construction Satisfiability Calculus (MCSAT)**.

MCSAT Implications

(Slide 39/47)

Less projection (only polynomials in the conflict) and less root isolation (only once per level). Further optimisations in cell production have also been found (Brown, 2013).

The cells are produced independently of each other:

- When UNSAT is concluded it usually means we have produced a **covering** of the theory space.
I.e. $\bigcup_i C_i = \mathbb{R}^n$ but $C_i \cap C_j$ need not be empty.
- The cells are locally cylindrical (projections trivial via cell description) but do not stack up in global cylinders.

MCSAT Implications

(Slide 39/47)

Less projection (only polynomials in the conflict) and less root isolation (only once per level). Further optimisations in cell production have also been found (Brown, 2013).

The cells are produced independently of each other:

- When UNSAT is concluded it usually means we have produced a **covering** of the theory space.
I.e. $\bigcup_i C_i = \mathbb{R}^n$ but $C_i \cap C_j$ need not be empty.
- The cells are locally cylindrical (projections trivial via cell description) but do not stack up in global cylinders.

NLSAT outperforms SAT+CAD in the SMT framework. But because it is a different framework it cannot be easily combined with other SMT modules for problems in multiple theories. Can we produce a covering based algorithm that fits in the Lazy SMT Framework?

MCSAT References



D. Jovanović and L. de Moura.

Solving Non-linear Arithmetic.

Proc. IJCAR 2012, LNCS 7364, pp. 339-354. Springer, 2012.

https://doi.org/10.1007/978-3-642-31365-3_27



L. de Moura. and D. Jovanović

A model-constructing satisfiability calculus.

Proc. VMCAI 2013, LNCS 7737, pp. 1-12. Springer, 2013.

https://doi.org/10.1007/978-3-642-35873-9_1



C. Brown

Constructing a single open cell in a cylindrical algebraic decomposition

Proc. ISSAC 2013, pp. 133-140. ACM, 2013.

<https://doi.org/10.1145/2465506.2465952>

Outline

- 1 Background on CAD
 - Definition
 - Applications of CAD
 - Complexity and Implementations
- 2 SC-Square and CAD
 - Satisfiability Checking
 - SC-Square
 - CAD for SMT
- 3 Cylindrical Algebraic Coverings
 - MCSAT
 - **CDCAC**
 - Future Outlook

Conflict Driven Cylindrical Algebraic Covering (Slide 40/47)

We designed a theory solver algorithm based on a Conflict Driven search using Cylindrical Algebraic Coverings (CDCAC):



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.

JLAMP 119, pages 2352-2208. Elsevier, 2021.

<https://doi.org/10.1016/j.jlamp.2020.100633>

Conflict Driven Cylindrical Algebraic Covering (Slide 40/47)

We designed a theory solver algorithm based on a Conflict Driven search using Cylindrical Algebraic Coverings (CDCAC):



E. Abraham, 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.

JLAMP 119, pages 2352-2208. Elsevier, 2021.

<https://doi.org/10.1016/j.jlamp.2020.100633>

Similar to NLSAT:

- Builds covering not decomposition.
- Conflict Driven so search guided away from past conflicts.

Unlike NLSAT:

- May be used as traditional SMT Theory Solver.
- Structured guidance to search in algebraic procedure.
- Cells are arranged cylindrically.

CDCAC: Basic Idea

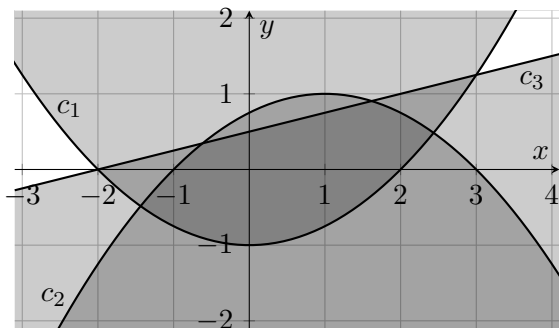
(Slide 41/47)

- Pick sample for lowest variable in ordering.
- Extend to increasingly higher dimensions in reference to those constraints made univariate.
- If all constraints satisfied then conclude SAT.
- If a constraint cannot be satisfied generalise to CAD cell in current dimension.
- Search outside the cell in that dimension.
- If entire dimension covered by cells then generalise to rule out cell in dimension below using CAD projection.
- Conclude UNSAT when covering for lowest dimension obtained.

Following slides by Gereon Kremer show simple example.

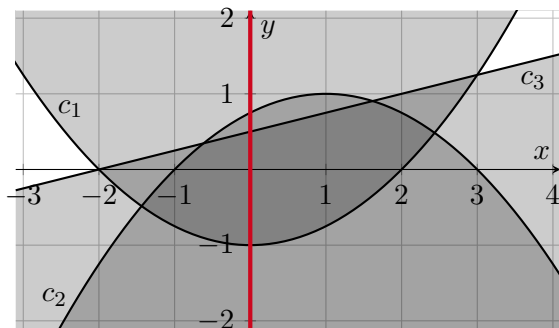
An example

$$c_1 : 4 \cdot y < x^2 - 4 \quad c_2 : 4 \cdot y > 4 - (x - 1)^2 \quad c_3 : 4 \cdot y > x + 2$$



An example

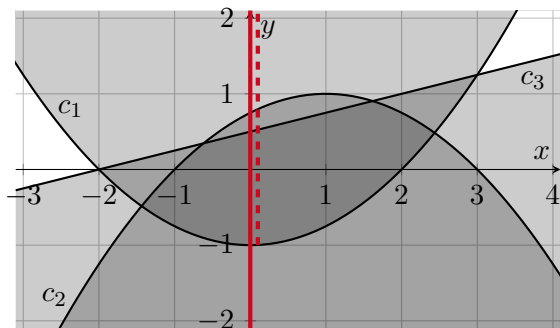
$$c_1 : 4 \cdot y < x^2 - 4 \quad c_2 : 4 \cdot y > 4 - (x - 1)^2 \quad c_3 : 4 \cdot y > x + 2$$



No constraint for x
Guess $x \mapsto 0$

An example

$$c_1 : 4 \cdot y < x^2 - 4 \quad c_2 : 4 \cdot y > 4 - (x - 1)^2 \quad c_3 : 4 \cdot y > x + 2$$



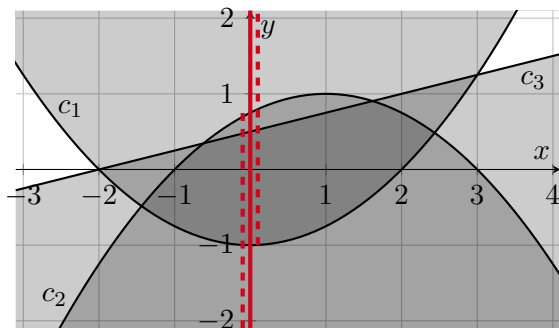
No constraint for x

Guess $x \mapsto 0$

$c_1 \rightarrow y \notin (-1, \infty)$

An example

$$c_1 : 4 \cdot y < x^2 - 4 \quad c_2 : 4 \cdot y > 4 - (x - 1)^2 \quad c_3 : 4 \cdot y > x + 2$$



No constraint for x

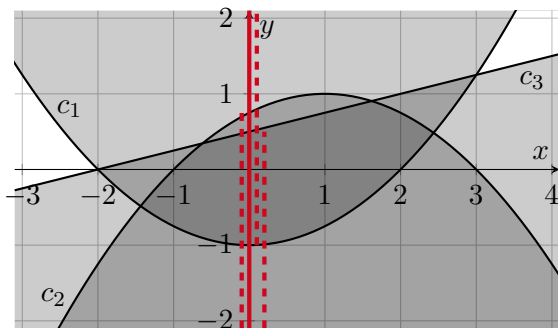
Guess $x \mapsto 0$

$$c_1 \rightarrow y \notin (-1, \infty)$$

$$c_2 \rightarrow y \notin (-\infty, 0.75)$$

An example

$$c_1 : 4 \cdot y < x^2 - 4 \quad c_2 : 4 \cdot y > 4 - (x - 1)^2 \quad c_3 : 4 \cdot y > x + 2$$



No constraint for x

Guess $x \mapsto 0$

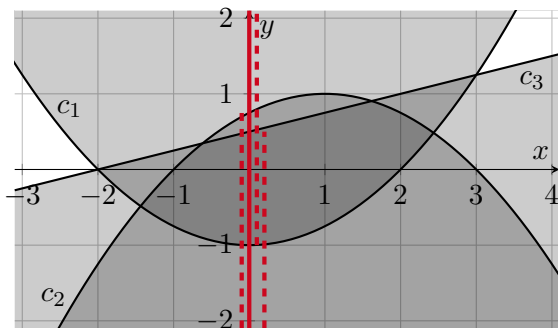
$$c_1 \rightarrow y \notin (-1, \infty)$$

$$c_2 \rightarrow y \notin (-\infty, 0.75)$$

$$c_3 \rightarrow y \notin (-\infty, 0.5)$$

An example

$$c_1 : 4 \cdot y < x^2 - 4 \quad c_2 : 4 \cdot y > 4 - (x - 1)^2 \quad c_3 : 4 \cdot y > x + 2$$



No constraint for x

Guess $x \mapsto 0$

$$c_1 \rightarrow y \notin (-1, \infty)$$

$$c_2 \rightarrow y \notin (-\infty, 0.75)$$

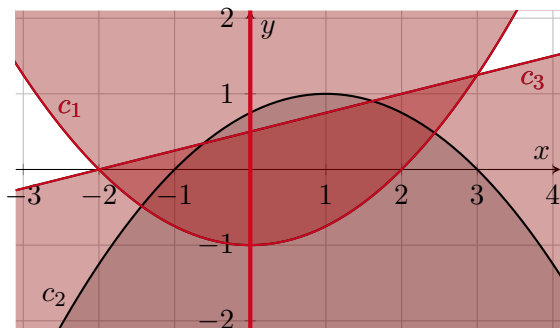
$$c_3 \rightarrow y \notin (-\infty, 0.5)$$

Construct covering

$$(-\infty, 0.5), (-1, \infty)$$

An example

$$c_1 : 4 \cdot y < x^2 - 4 \quad c_2 : 4 \cdot y > 4 - (x - 1)^2 \quad c_3 : 4 \cdot y > x + 2$$



No constraint for x

Guess $x \mapsto 0$

$$c_1 \rightarrow y \notin (-1, \infty)$$

$$c_2 \rightarrow y \notin (-\infty, 0.75)$$

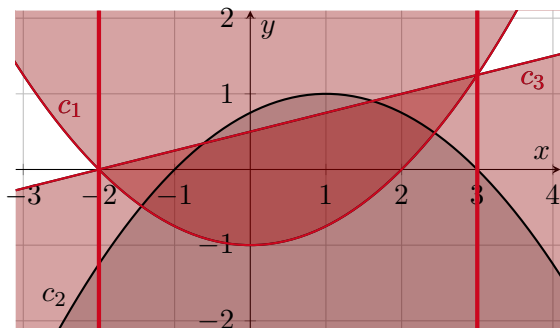
$$c_3 \rightarrow y \notin (-\infty, 0.5)$$

Construct covering

$$(-\infty, 0.5), (-1, \infty)$$

An example

$$c_1 : 4 \cdot y < x^2 - 4 \quad c_2 : 4 \cdot y > 4 - (x - 1)^2 \quad c_3 : 4 \cdot y > x + 2$$



No constraint for x

Guess $x \mapsto 0$

$$c_1 \rightarrow y \notin (-1, \infty)$$

$$c_2 \rightarrow y \notin (-\infty, 0.75)$$

$$c_3 \rightarrow y \notin (-\infty, 0.5)$$

Construct covering

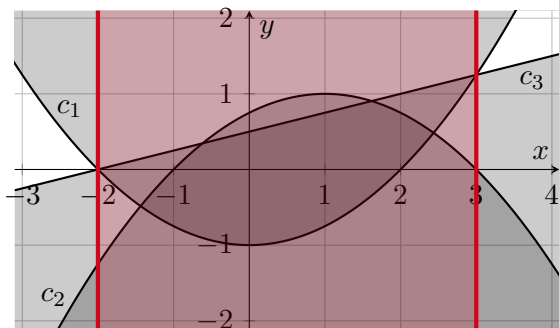
$$(-\infty, 0.5), (-1, \infty)$$

Construct interval for x

$$x \notin (-2, 3)$$

An example

$$c_1 : 4 \cdot y < x^2 - 4 \quad c_2 : 4 \cdot y > 4 - (x - 1)^2 \quad c_3 : 4 \cdot y > x + 2$$



No constraint for x

Guess $x \mapsto 0$

$$c_1 \rightarrow y \notin (-1, \infty)$$

$$c_2 \rightarrow y \notin (-\infty, 0.75)$$

$$c_3 \rightarrow y \notin (-\infty, 0.5)$$

Construct covering

$$(-\infty, 0.5), (-1, \infty)$$

Construct interval for x

$$x \notin (-2, 3)$$

New guess for x

CDCAC Experimental Results I

(Slide 42/47)

An initial implementation was made in SMT-RAT and experiments on the SMTLIB were described in the JLAMP paper:

- Showed CDCAC much more efficient as theory solver than incremental CAD.
- But did not outperform NLSAT routine in SMT-RAT overall.
- Did outperform NLSAT on substantial example subsets:
 - 555 problems where CDCAC times out and NLSAT completes.
 - 358 problems where NLSAT times out and CDCAC completes.

Although both algorithms create coverings, they seem to create different ones. Potential for a meta-solver?

CDCAC Experimental Results II

(Slide 43/47)

A new implementation of CDCAC was made in cvc5 by Gereon Kremer. Can download cvc5 for free here:

<https://cvc5.github.io/downloads.html>

The cvc5 solver won the 2022 SMT Competition, and in particular the QF_NRA track:

<https://smt-comp.github.io/2022/results/qf-nonlineararith-single-query>
Z3 (which uses MCSAT) did not enter, but results in the paper below show cvc5 outperforming Z3.



G. Kremer, E. Abraham, M. England and James H. Davenport.

On the Implementation of Cylindrical Algebraic Coverings for Satisfiability Modulo Theories Solving.

Proc. SYNASC 2021, pages 37–39. IEEE, 2021.

<https://doi.org/10.1109/SYNASC54541.2021.00018>

Outline

- 1 Background on CAD
 - Definition
 - Applications of CAD
 - Complexity and Implementations
- 2 SC-Square and CAD
 - Satisfiability Checking
 - SC-Square
 - CAD for SMT
- 3 Cylindrical Algebraic Coverings
 - MCSAT
 - CDCAC
 - Future Outlook

Proofs for SMT Verification

(Slide 44/47)

Machine readable proofs of unsatisfiability is a growing trend in SMT: cvc5 achieves this for many theories but not yet NRA.



H. Barbosa et al.

Flexible Proof Production in an Industrial-Strength SMT Solver.

Proc. IJCAR 2022, LNCS 13385, pages 15-35. Springer, 2022.

https://doi.org/10.1007/978-3-031-10769-6_3

We hypothesise that the structure of the CDCAC search may allow for easier extraction of such proofs.



E. Ábrahám, J.H. Davenport, M. England, G. Kremer, and Z. Tonks.

New Opportunities for the Formal Proof of Computational Real Geometry?

Proc. SC² 2020, CEUR-WS 2752, pages 178–188, 2020.

<http://ceur-ws.org/Vol-2752/>

Impact for computer algebra?

(Slide 45/47)

The story above shows how a combined SC-Square approach has led to better Satisfiability Checking. But is there better computer algebra too?

Recall that CAD was developed to solve the Real QE Problem, of which satisfiability in QF_NRA is simply a small subclass. Can SC-Square ideas tackle this too?

Impact for computer algebra?

(Slide 45/47)

The story above shows how a combined SC-Square approach has led to better Satisfiability Checking. But is there better computer algebra too?

Recall that CAD was developed to solve the Real QE Problem, of which satisfiability in QF_NRA is simply a small subclass. Can SC-Square ideas tackle this too?



G. Kremer and J. Nalbach.

Cylindrical Algebraic Coverings for Quantifiers.

To Appear: Proc. SC² 2022.

A first attempt has been made to extend the CDCAC algorithm to Real QE. Awaiting implementation.

NuCAD

(Slide 46/47)

Non-uniformly cylindrical CAD (NuCAD) produces a decomposition (not covering) where the the cells are not arranged cylindrically: allows for fewer cells, but necessitates additional work to perform QE. To date only developed for open cells.



C. Brown.

Open non-uniform cylindrical algebraic decompositions.

Proc. ISSAC 2015, pages 85-92. ACM, 2015.

<https://doi.org/10.1145/2755996.2756654>



C. Brown.

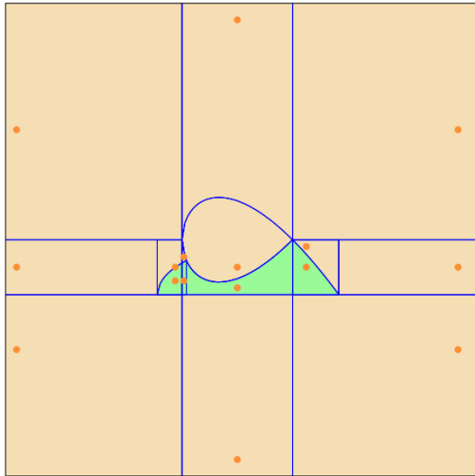
Projection and Quantifier Elimination Using Non-uniform Cylindrical Algebraic Decomposition.

Proc. ISSAC 2017, pages 53-60. ACM, 2017.

<https://doi.org/10.1145/3087604.3087651>

NuCAD Example

(Slide 47/47)



The End

The author continues to work on these topics continues in the EPSRC DEWCAD Project (Grant EP/T015748/1): *Pushing Back the Doubly Exponential Walls of Cylindrical Algebraic Decomposition*

Contact Details

Matthew.England@coventry.ac.uk

<https://matthewengland.coventry.domains/>