Cylindrical Algebraic Coverings to Determine Satisfiability of Non-linear Polynomial Constraints

Matthew England (Coventry University, UK)

SIAM Conference on Applied Algebraic Geometry 2023 (AG23)
Eindhoven, The Netherlands
10−14 July 2023
Data and Certificates in Algebra and Geometry Session

Supported by EPSRC EP/T015748/1 (DEWCAD) and EU H2020 712689 (SC^2).

Summary in One Slide

I will describe an algorithm from a 2021 paper which reformulated traditional computer algebra technology for non-linear polynomial systems so it could be combined with modern SAT-solvers.
I will then discuss certificates for this, including work currently under review suggesting a novel reimplementation as a combination of proof system and heuristics.

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 ideas – (a) combining Satisfiability Checking and Symbolic Computation and (b) the use of proof systems to separate algorithm correctness from algorithm – may be useful in other domains too!

References and Thanks (Slide 2/25)

This is joint work with Erika Ábrahám, Chris Brown, James H. Davenport, Gereon Kremer, Jasper Nalback, and Phillipe Specht.

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

J. Nalbach, E. Ábrahám, P. Specht, C.W. Brown, J.H. Davenport, M. England.
Levelwise construction of a single cylindrical algebraic cell.
Submitted for Publication, 2023.
Preprint: https://arxiv.org/abs/2212.09309

Thanks to session organisers Tobias Boege and Julia Lindberg for the invitation to speak. Apologies for not being there in person.

Outline

Outline

Cylindrical Algebraic Decomposition I

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

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

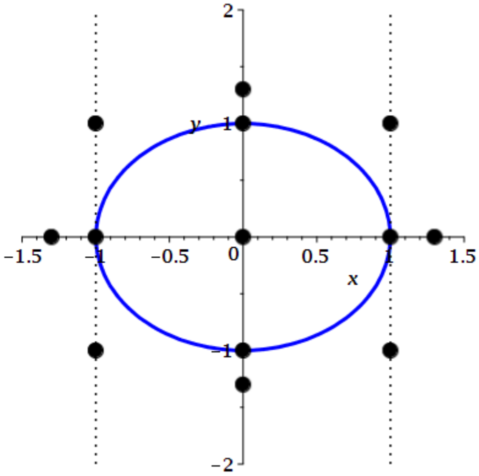
CAD may also refer to an algorithm that produces the CAD object.

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.

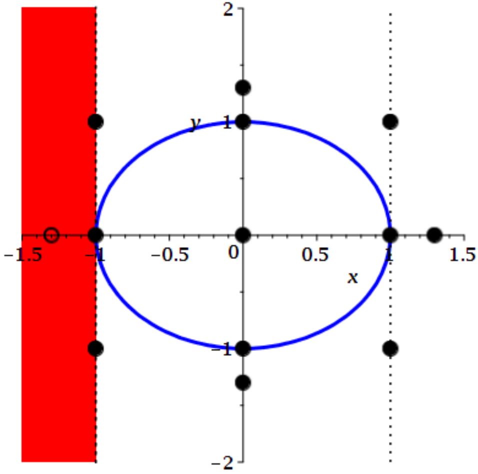
Example: Circle – visualisation

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



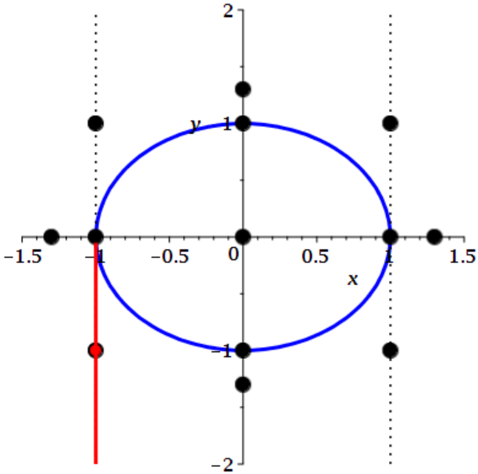
Example: Circle – visualisation

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



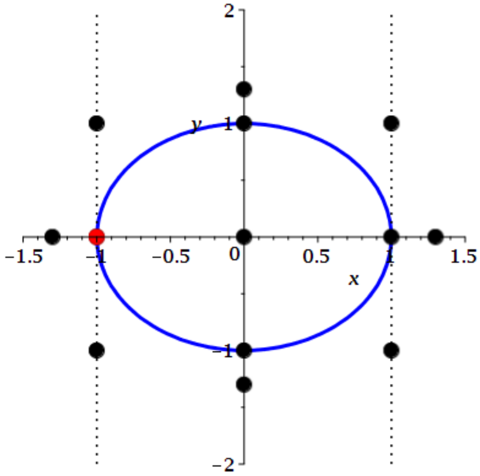
Example: Circle – visualisation

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



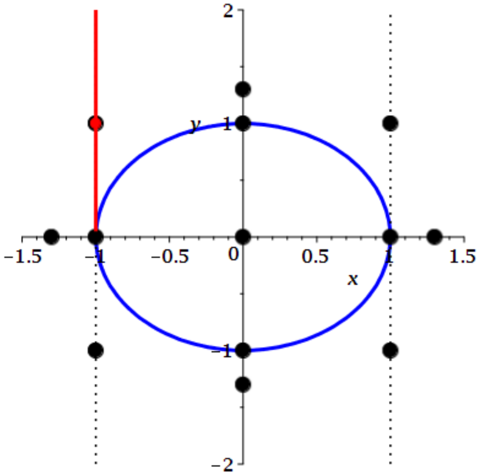
Example: Circle – visualisation

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



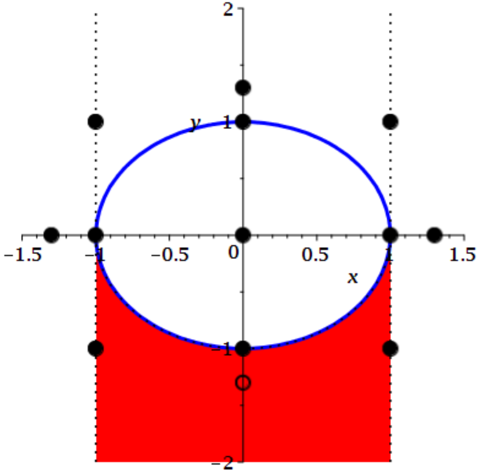
Example: Circle – visualisation (Slide 6/25)

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



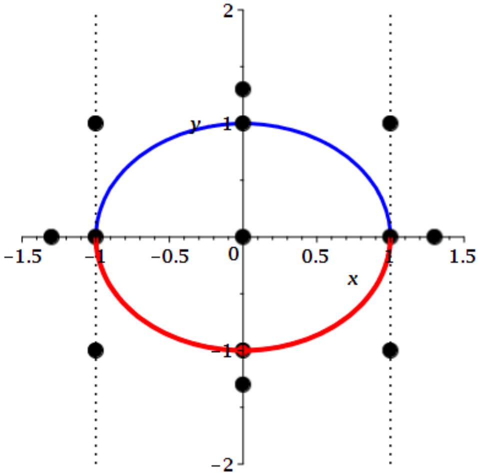
Example: Circle — visualisation

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



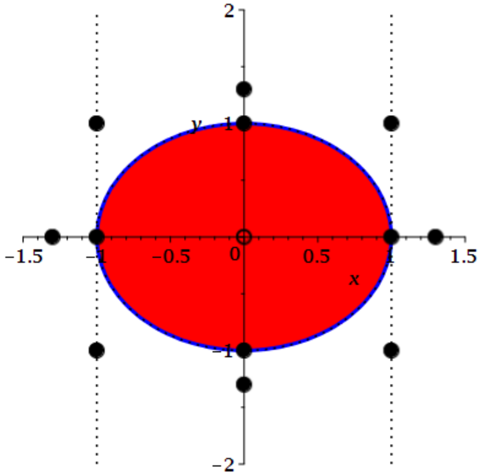
Example: Circle – visualisation

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



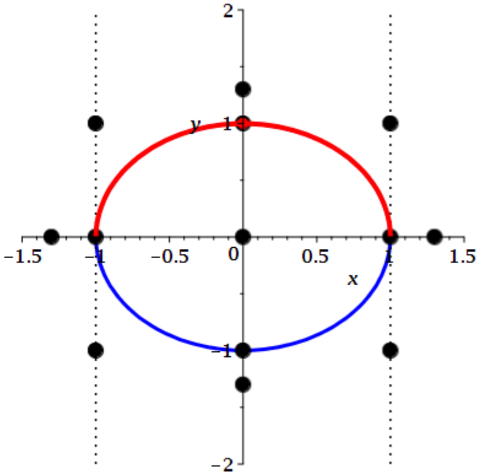
Example: Circle – visualisation

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



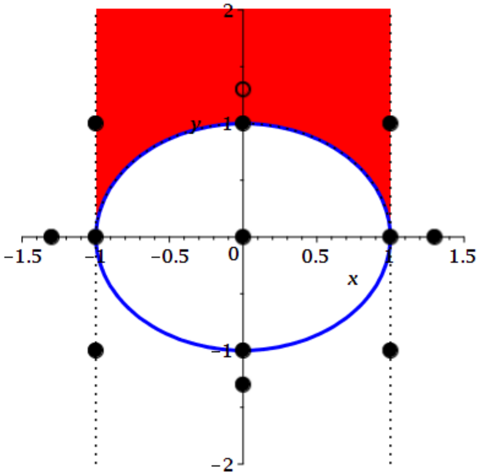
Example: Circle – visualisation

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



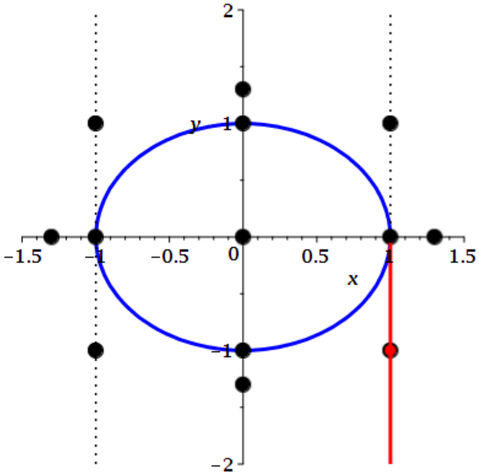
Example: Circle – visualisation

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



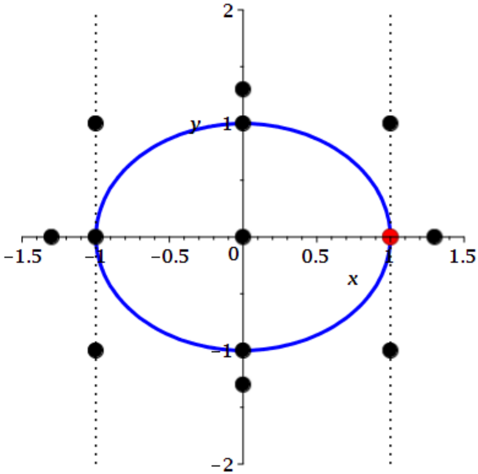
Example: Circle – visualisation

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



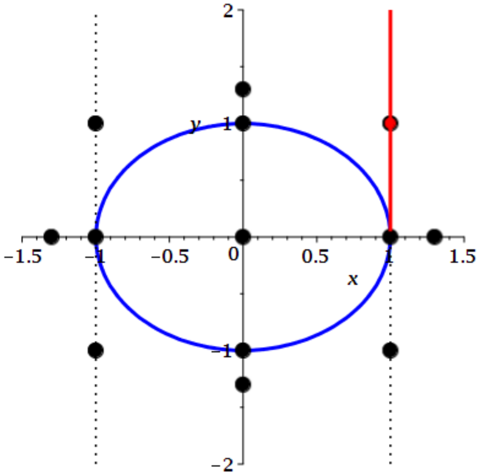
Example: Circle – visualisation

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



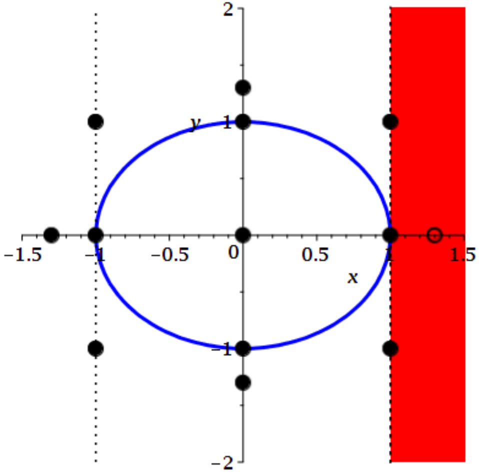
Example: Circle – visualisation (Slide 6/25)

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 6/25)

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



How to build a CAD?

The usual approach is to calculate projection polynomials whose roots indicate changes in the behaviour of the input set; decompose with respect to these and then lift back working at a sample point. Safe if the projection provides **delineability**.

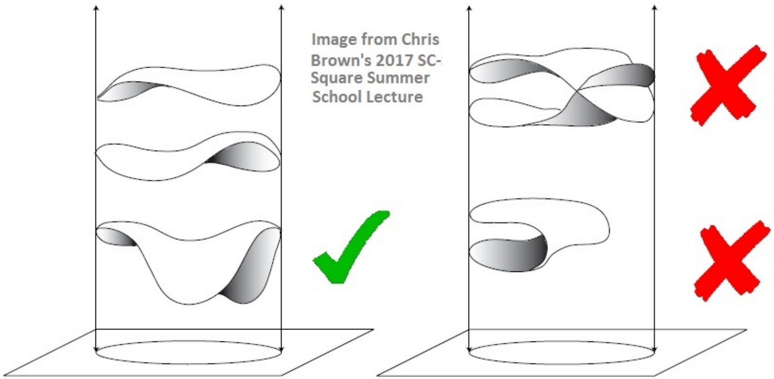


Image from Chris Brown's 2017 SC-Square Summer School Lecture

Classic CAD References

[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.

Outline

Applications of CAD

CAD can be used to produce solution sets of non-linear polynomial systems and to perform Quantified Elimination over the reals.

There have been 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).
- multi-stationarity identification in chemical reaction networks (Bradford et al. 2017).

Application References

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. Griebl, and C. Lengauer.
Quantifier elimination in automatic loop parallelization.
Journal of Symbolic Computation, 41(11):1206–1221, 2006.

Application References cont.

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

CAD Complexity

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

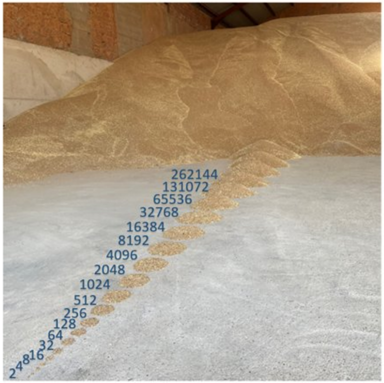
By the end of projection you could have doubly exponentially many polynomials of doubly exponential degree (in the number of projections, i.e. variables). Hence the number of real roots, cells, and time to compute them grows doubly exponentially too.

C. Brown and J.H. Davenport.
The complexity of quantifier elimination and cylindrical algebraic decomposition.
In Proc. ISSAC '07, pages 54–60. ACM, 2007.

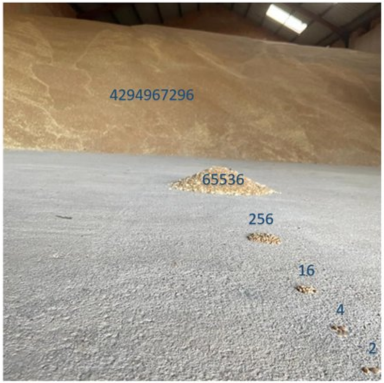
Nevertheless, CAD remains the most used general purpose method for real QE.

The Doubly Exponential Wall

Exponential Growth



Doubly Exponential Growth



Images by Tereso del Río

CAD Implementations

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

Boolean SAT Problem

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.

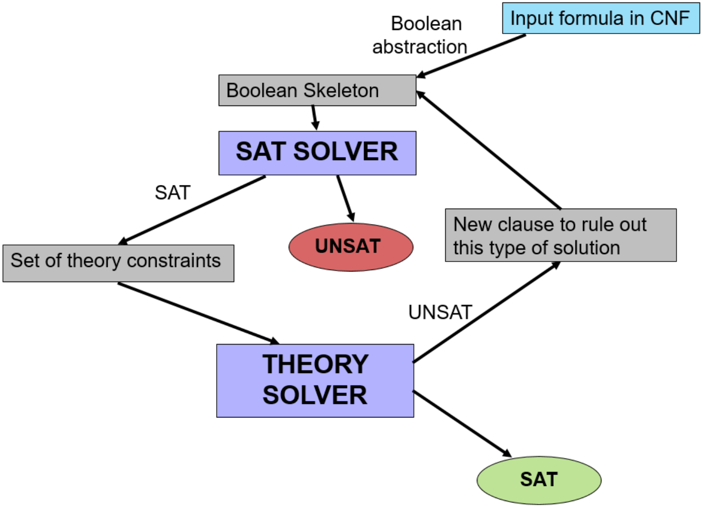
If the formula has n variables then there are 2^n possible assignments. The SAT problem is NP-Complete. However, **SAT-solvers** today routinely solve huge problem instances!

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:

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

Lazy SMT Framework



Outline

CAD as NRA Theory Solver

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

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.

How good is such a CAD adaptation?

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.

NLSAT / MCSAT

Jovanović and de Moura's NLSAT algorithm for Microsoft's Z3 solver departs from the Lazy SMT framework for non-linear real arithmetic and instead uses a single proof system (called MCSAT).

Boolean conflicts generalised by SAT techniques and theory conflicts by the creation of CAD cells around a point.

It uses less projection (only polynomials in the conflict)

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 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

NLSAT outperforms SAT+CAD in the SMT framework. But it cannot be combined with other SMT modules. Can we produce a covering based algorithm that fits in the Lazy SMT Framework?

Outline

Conflict Driven Cylindrical Algebraic Covering

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

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

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.

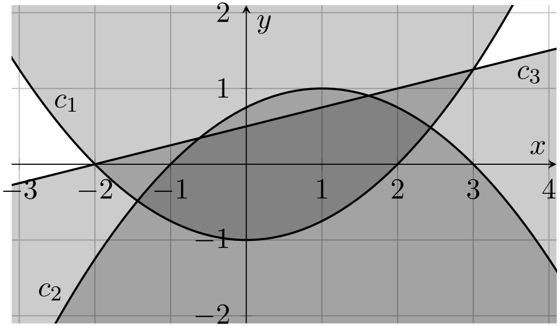
CDCAC: Basic Idea

- 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.

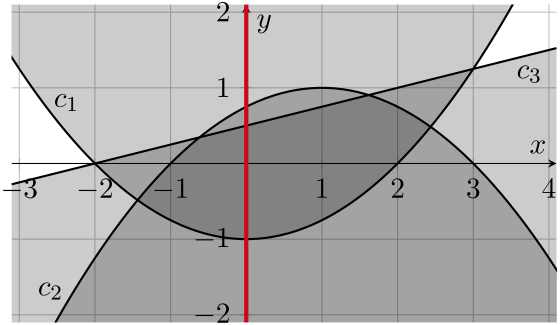
An example

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



An example

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

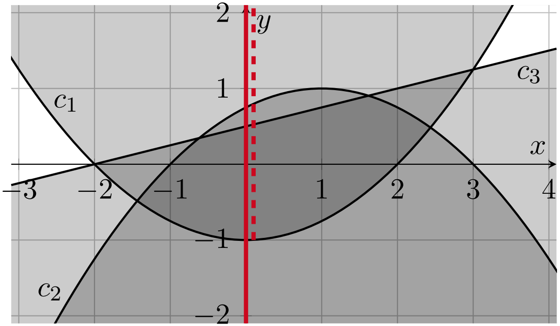


No constraint for x

Guess $x \mapsto 0$

An example

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



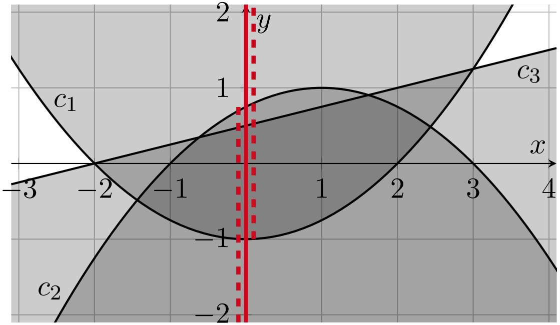
No constraint for x

Guess $x \mapsto 0$

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

An example

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



No constraint for x

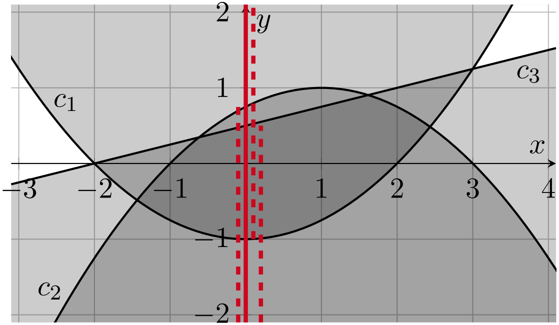
Guess $x \mapsto 0$

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

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

An example

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



No constraint for x

Guess $x \mapsto 0$

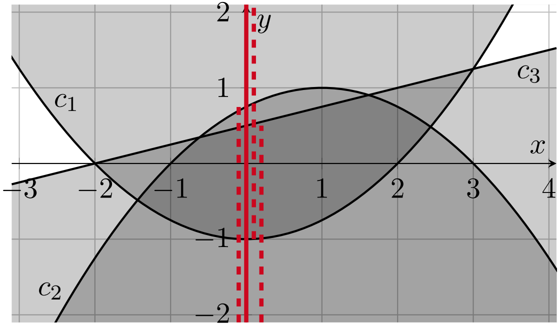
$c_1 \rightarrow y \notin (-1, \infty)$

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

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

An example

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



No constraint for x

Guess $x \mapsto 0$

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

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

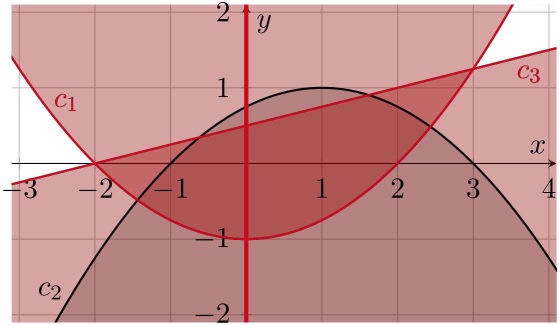
$c_3 \to y \notin (-\infty, 0.5)$

Construct covering

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

An example

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



No constraint for x

Guess $x \mapsto 0$

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

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

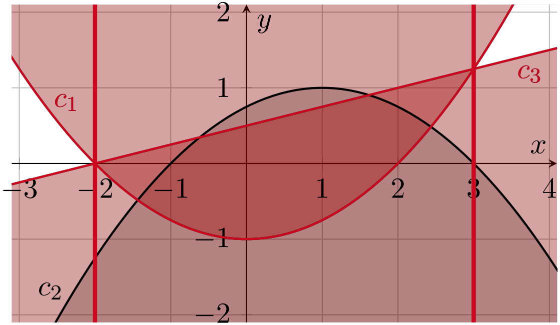
$c_3 \to y \notin (-\infty, 0.5)$

Construct covering

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

An example

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



No constraint for x

Guess $x \mapsto 0$

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

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

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

Construct covering

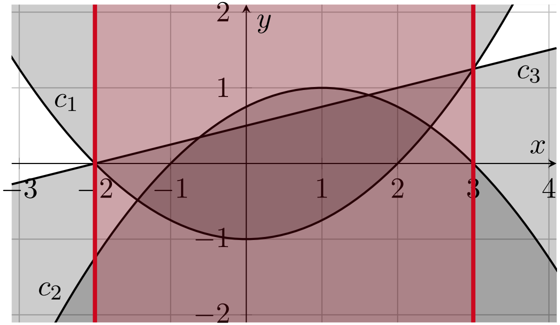
$(-\infty, 0.5), (-1, \infty)$

Construct interval for x

$x \notin (-2, 3)$

An example

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



No constraint for x

Guess $x \mapsto 0$

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

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

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

Construct covering

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

Construct interval for x

$x \notin (-2, 3)$

New guess for x

CDCAC Experimental Results

CDCAC much more efficient as Lazy SMT Theory Solver than CAD. When compared to NLSAT: substantial example sets where one outperformed the other. Potential for meta solver?

In 2022 cvc5 won the 2022 SMT Competition, in particular the `QF_NRA` track:
https://smt-comp.github.io/2022/results/qf-nonlinearintarith-single-query
This version of cvc5 uses CDCAC is the core algorithm for non-linear real arithmetic:

G. Kremer, E. Ábrahám, 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

Certificates and Verification

In the case of satisfiability both CAD and CDCAC provide a satisfying sample point to serve as certificate.

But in the case of unsatisfiability, one would need to verify that:
(a) the cells form a decomposition / covering (not too hard given the cylindrical structure); and
(b) each cells satisfies the stated invariance property (difficult).
An attempt was made to formalise CAD in Coq by Cohen and Maboubi but this was never completed.

C. Cohen and A. Mahboubi.
Formal proofs in real algebraic geometry: from ordered fields to quantifier elimination.
Logical Methods in Computer Science, 8(1):1–40, 2012.
https://doi.org/10.2168/LMCS-8(1:2)2012

Outline

Proofs for SMT Verification

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 have hypothesised that the structure of the CDCAC search may allow for easier extraction of such proofs as they more closely follow normal human mathematical reasoning.

E. Ábrahám, J.H. Davenport, M. England, G. Kremer, and Z. Tonks.
New Opportunities for the Formal Proof of Computational Real Geometry?.
Proc. SC^2 2020, CEUR-WS 2752, pages 178—188, 2020.
http://ceur-ws.org/Vol-2752/

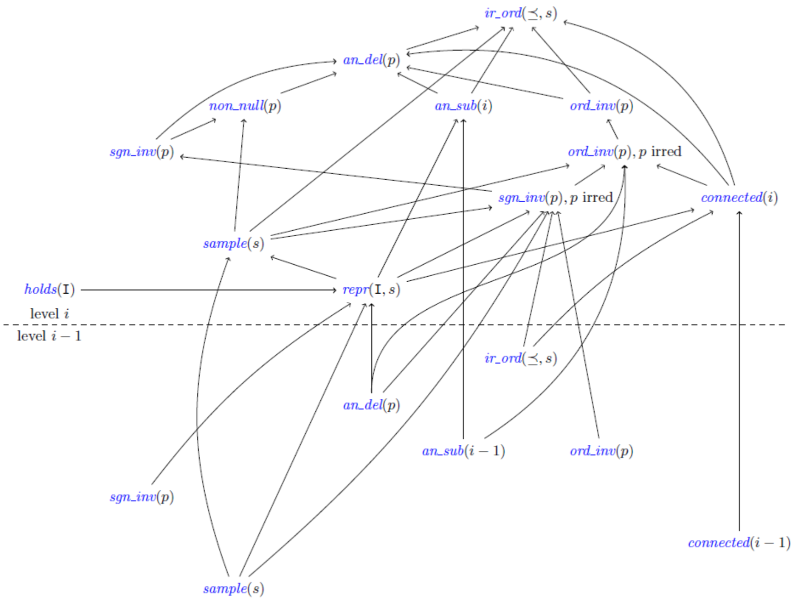
Outline

Proof System for cylindrical cell

Recently, we presented an algorithm to produce a single CAD cell as a *proof system*: properties of cells and rules of inference to infer that such a property holds.

The algorithm then simply searches for a chain of proof rules to prove a desired property (employing heuristics to take choices where there is freedom in how the chain can to be built).

J. Nalbach, E. Ábrahám, P. Specht, C.W. Brown, J.H. Davenport, M. England.
Levelwise construction of a single cylindrical algebraic cell.
Submitted for Publication, 2023.
Preprint: https://arxiv.org/abs/2212.09309


$ir\_ord(\preceq, s)$
$an\_del(p)$
$non\_null(p)$
$an\_sub(i)$
$ord\_inv(p)$
$sgn\_inv(p)$
$ord\_inv(p), p$ irred
$sgn\_inv(p), p$ irred
$connected(i)$
$sample(s)$
$holds(\mathtt{I})$
$repr(\mathtt{I}, s)$
level i
level $i-1$
$ir\_ord(\preceq, s)$
$an\_del(p)$
$an\_sub(i-1)$
$ord\_inv(p)$
$sgn\_inv(p)$
$connected(i-1)$
$sample(s)$

Proof System Presentations

Unlike traditional pseudo-code, a proof system presentation clearly separates out the parts of the algorithm upon which correctness rests and those parts where heuristic decisions may be taken. It allows for easier verification of the former and easier experimentation with the latter.

It is more prevalent in the SAT/SMT/logic communities where there is more intense work on the optimization of such heuristic choices and proof systems are an established presentation method. Rarely used in computer algebra: but there are the same potential benefits to it there!

The End

The author continues to work on these topics 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/