New Directions in Cylindrical Algebraic Decomposition

Matthew England (Coventry University, UK)

Fields Institute
Workshop on Real Algebraic Geometry and Algorithms for Geometric Constraint Systems
~~Toronto, Canada~~ Online 14–18 June 2021

Work supported by EPSRC Grant EP/R019622/1 and EU H2020 Project 712689.
Author currently supported by EPSRC grant EP/T015748/1.

Outline

Outline

Cylindrical Algebraic Decomposition

Cylindrical Algebraic Decomposition (CAD) was first introduced by Collins in the 1970s, as an algorithm to produce:

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

A CAD is traditionally built relative to a set of input polynomials such that each polynomial has constant sign in each cell: this is called **sign-invariance**. You can uncover properties of polynomials over infinite space by examining finite set of sample points.

Cylindrical Algebraic Decomposition

Cylindrical Algebraic Decomposition (CAD) was first introduced by Collins in the 1970s, as an algorithm to produce:

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

A CAD is traditionally built relative to a set of input polynomials such that each polynomial has constant sign in each cell: this is called **sign-invariance**. You can uncover properties of polynomials over infinite space by examining finite set of sample points.

Cylindrical Algebraic Decomposition

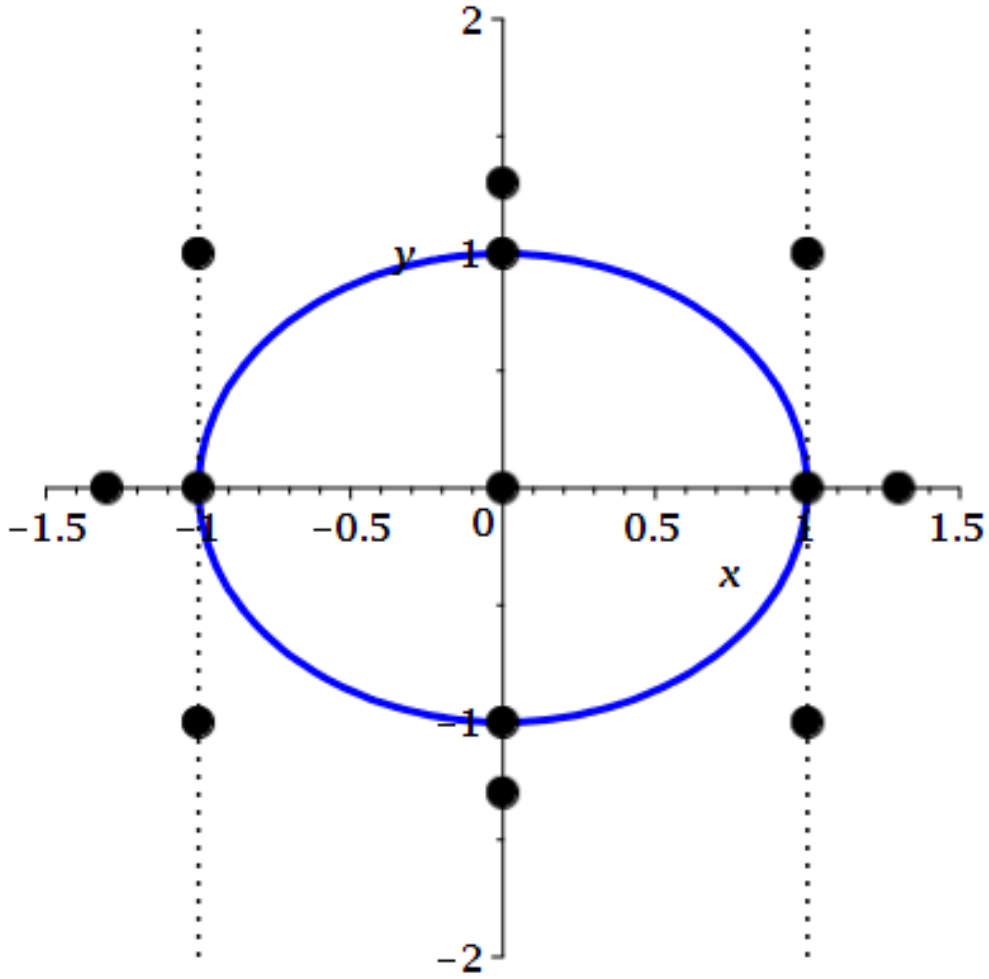
Cylindrical Algebraic Decomposition (CAD) was first introduced by Collins in the 1970s, as an algorithm to produce:

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

A CAD is ideally built relative to a set of input polynomial constraints such that each constraint has constant truth value in each cell: this is called **truth-invariance**. Uncover solutions from semi-algebraic descriptions of true cells.

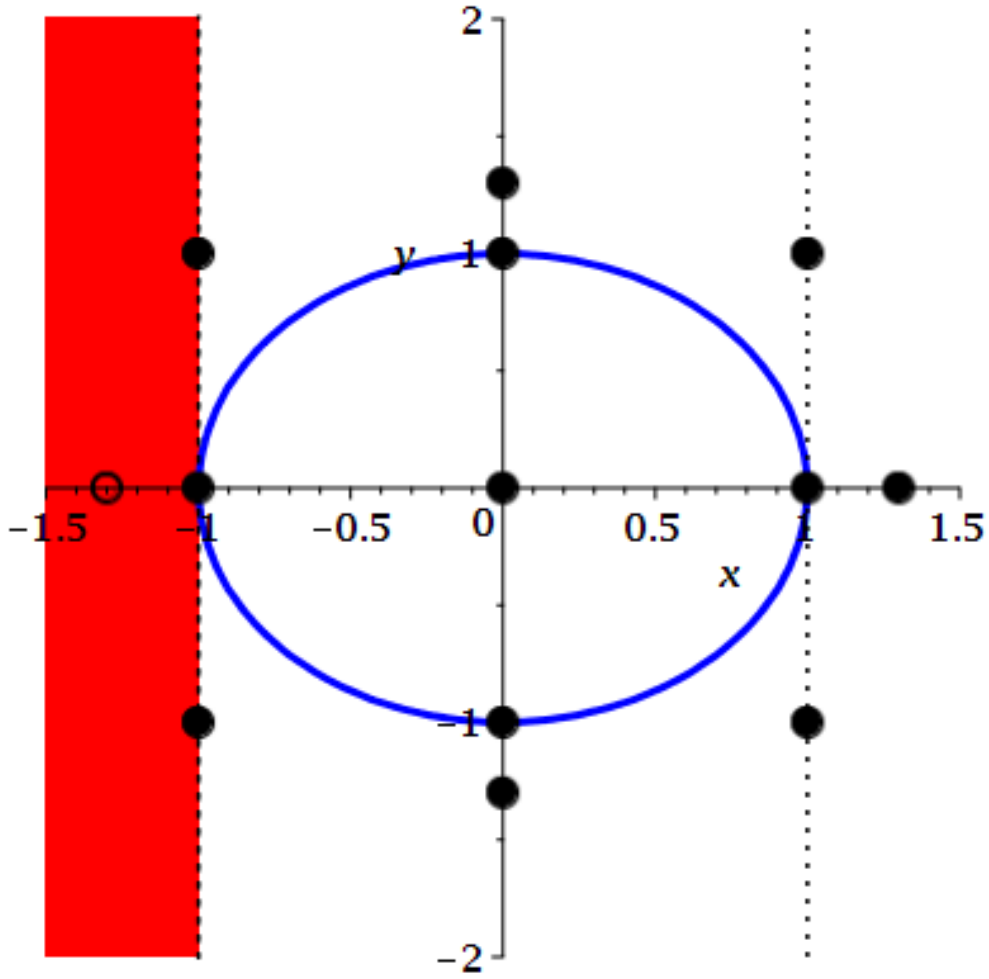
Example: Circle – decomposition visualisation

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



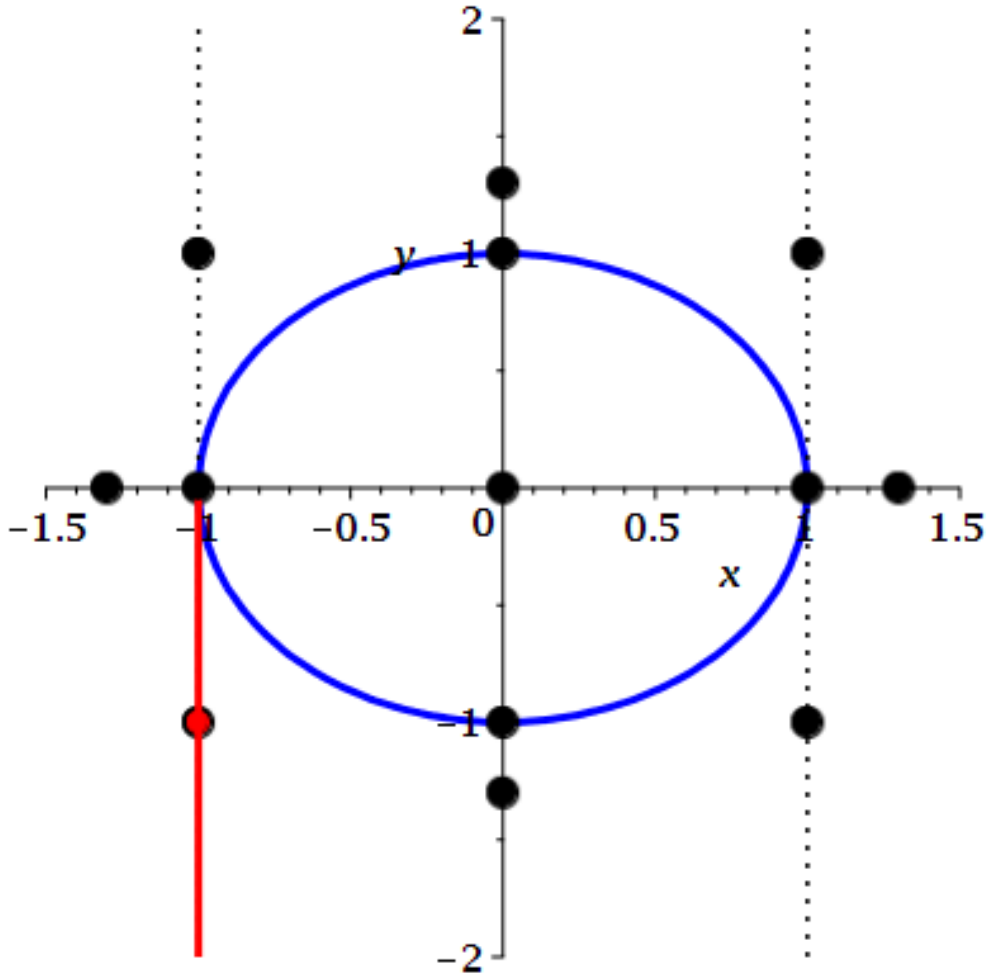
Example: Circle – visualisation

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



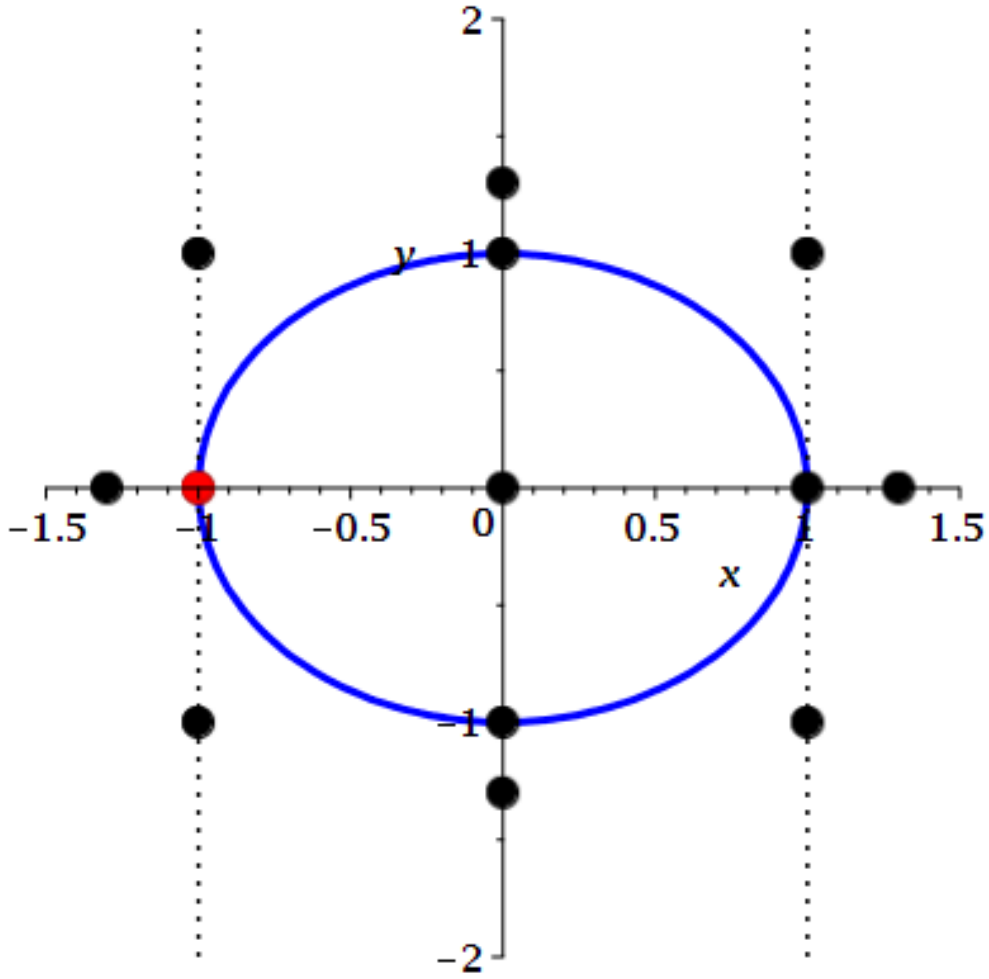
Example: Circle – visualisation

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



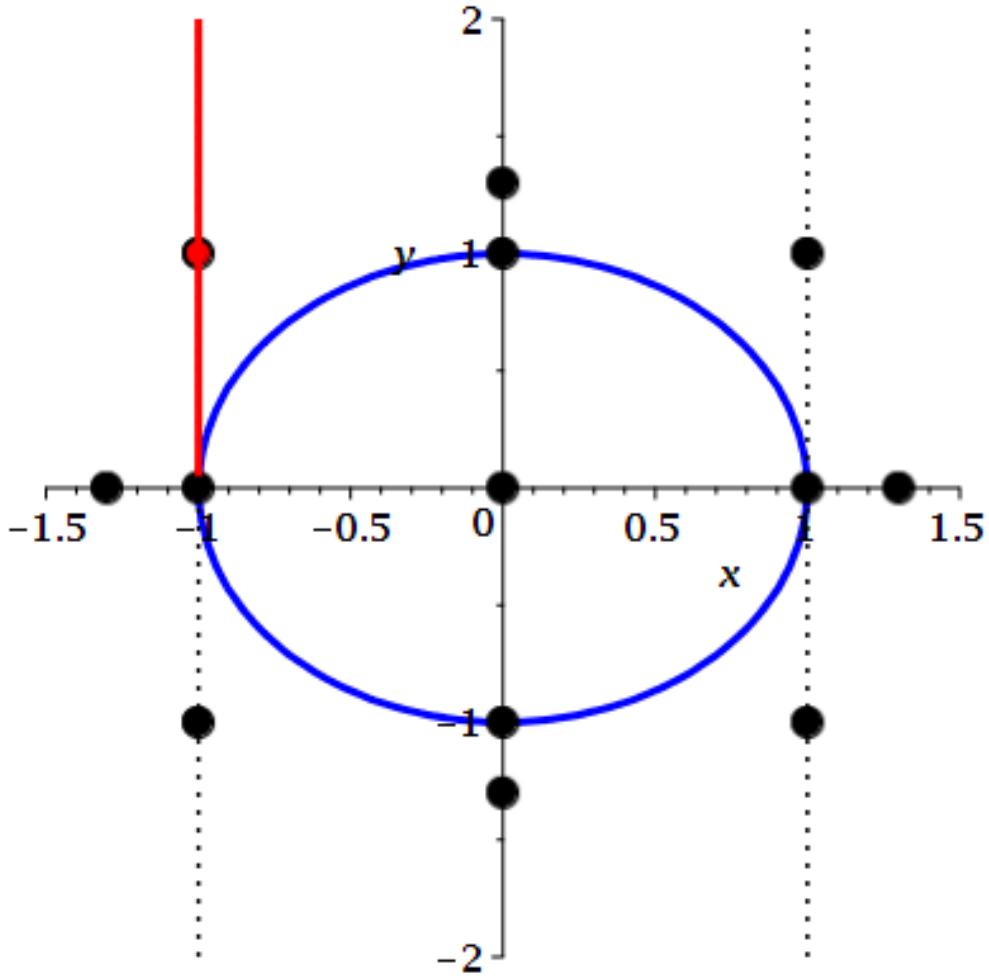
Example: Circle – visualisation

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



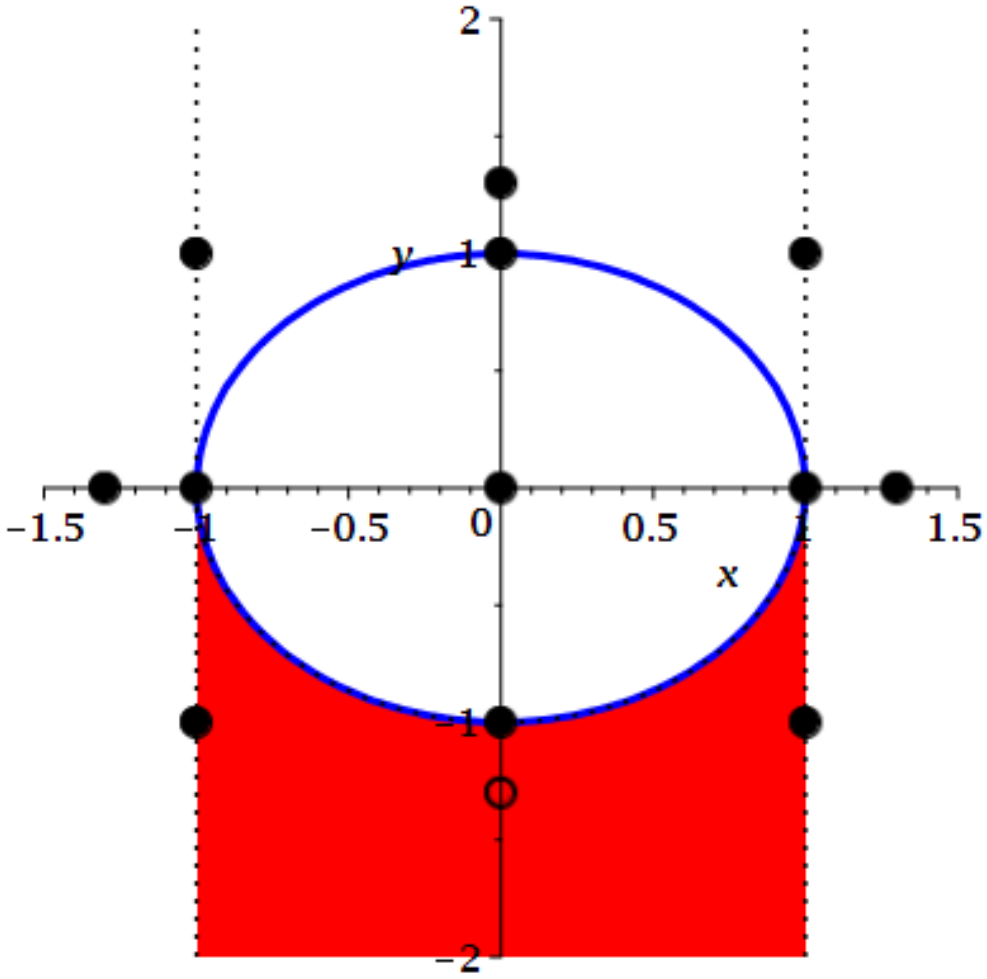
Example: Circle – visualisation

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



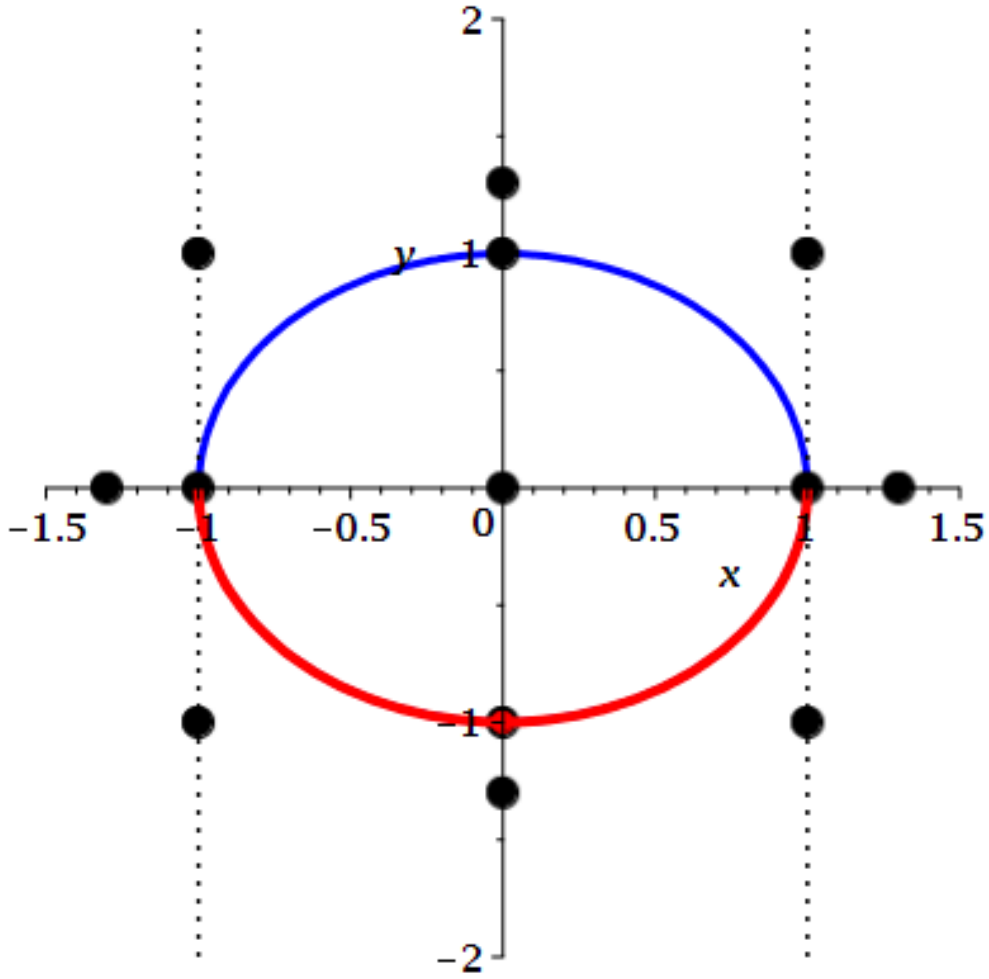
Example: Circle – visualisation

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



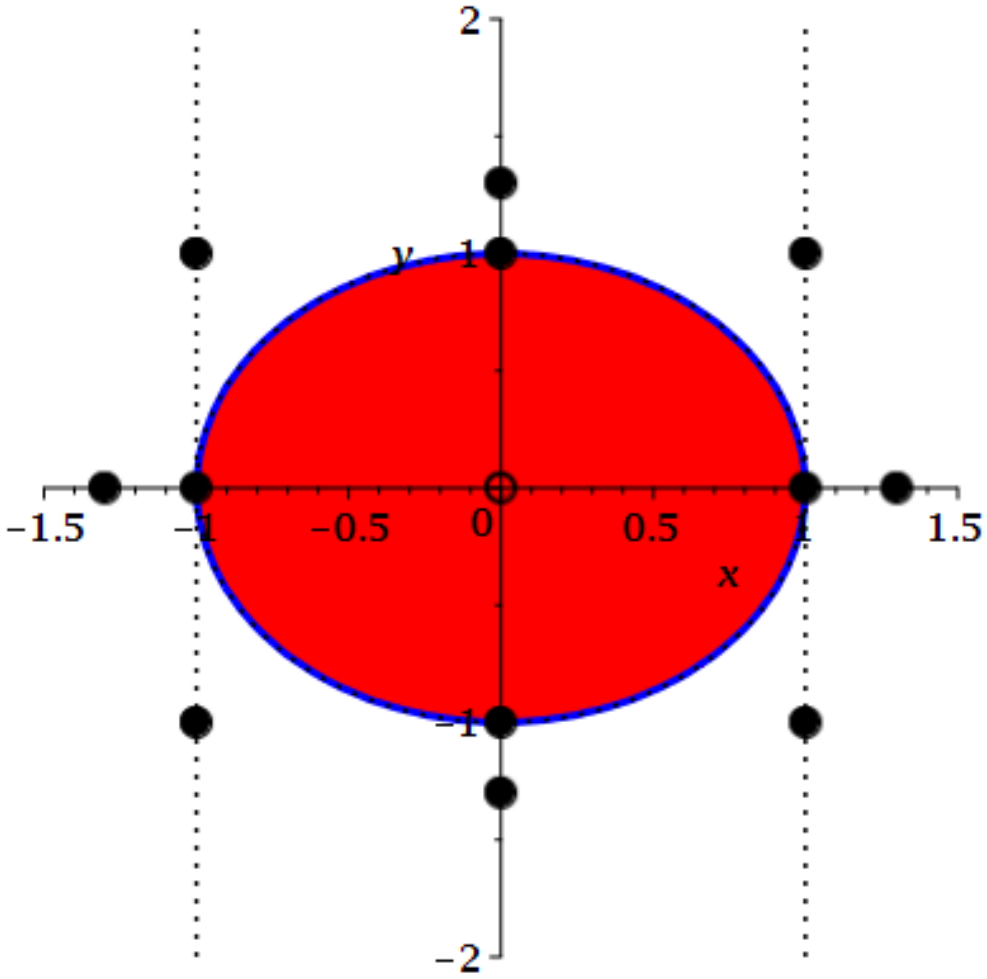
Example: Circle – visualisation

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



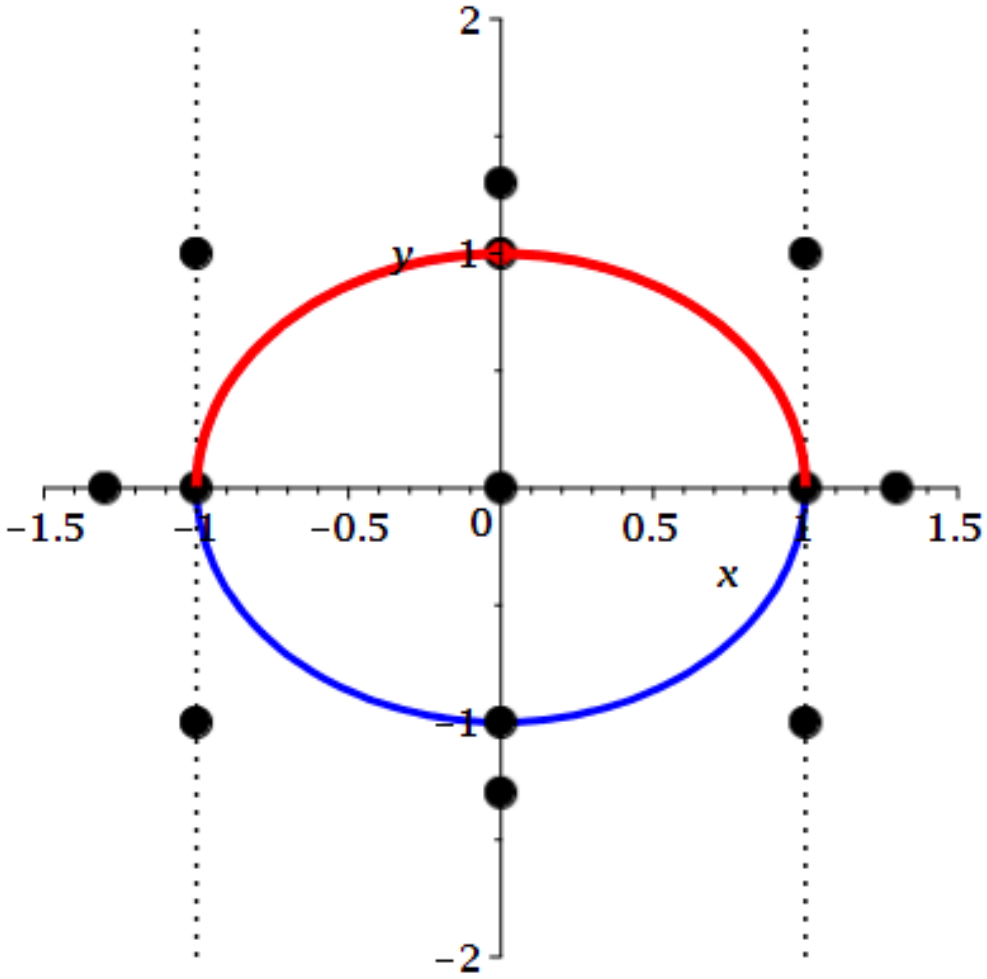
Example: Circle – visualisation

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



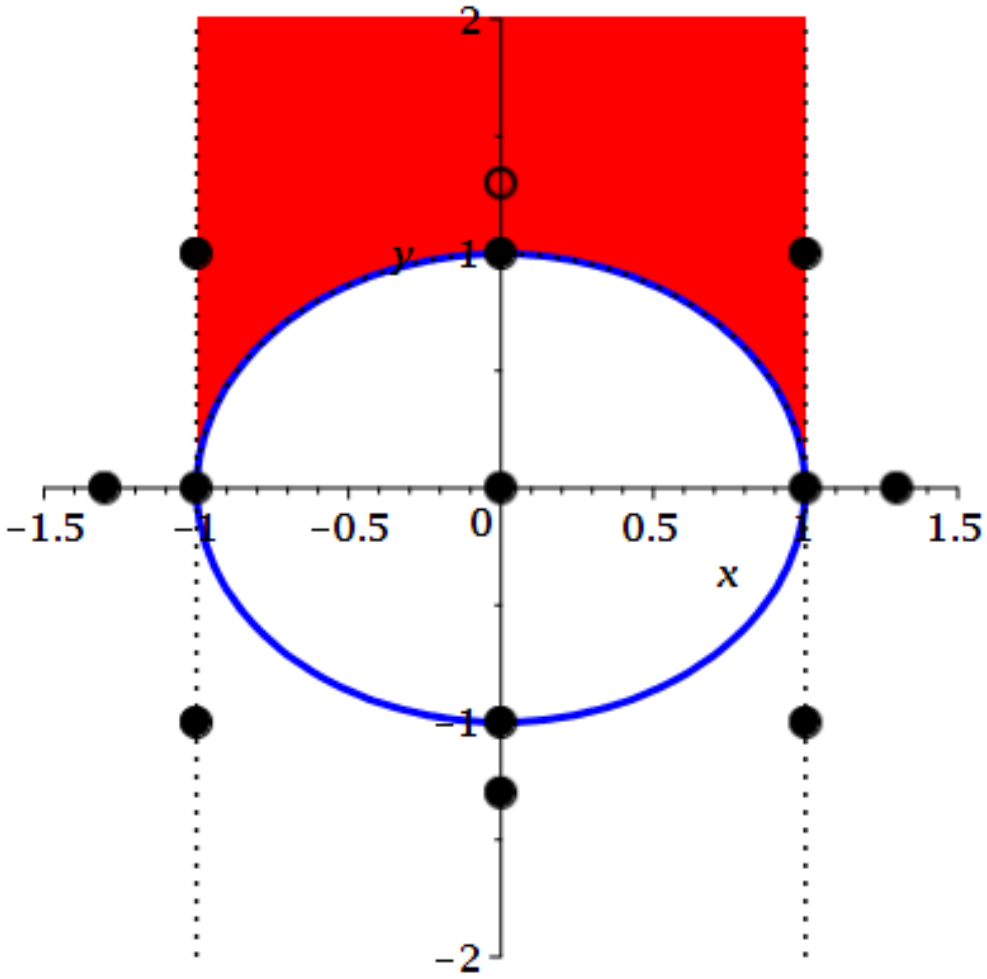
Example: Circle – visualisation

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



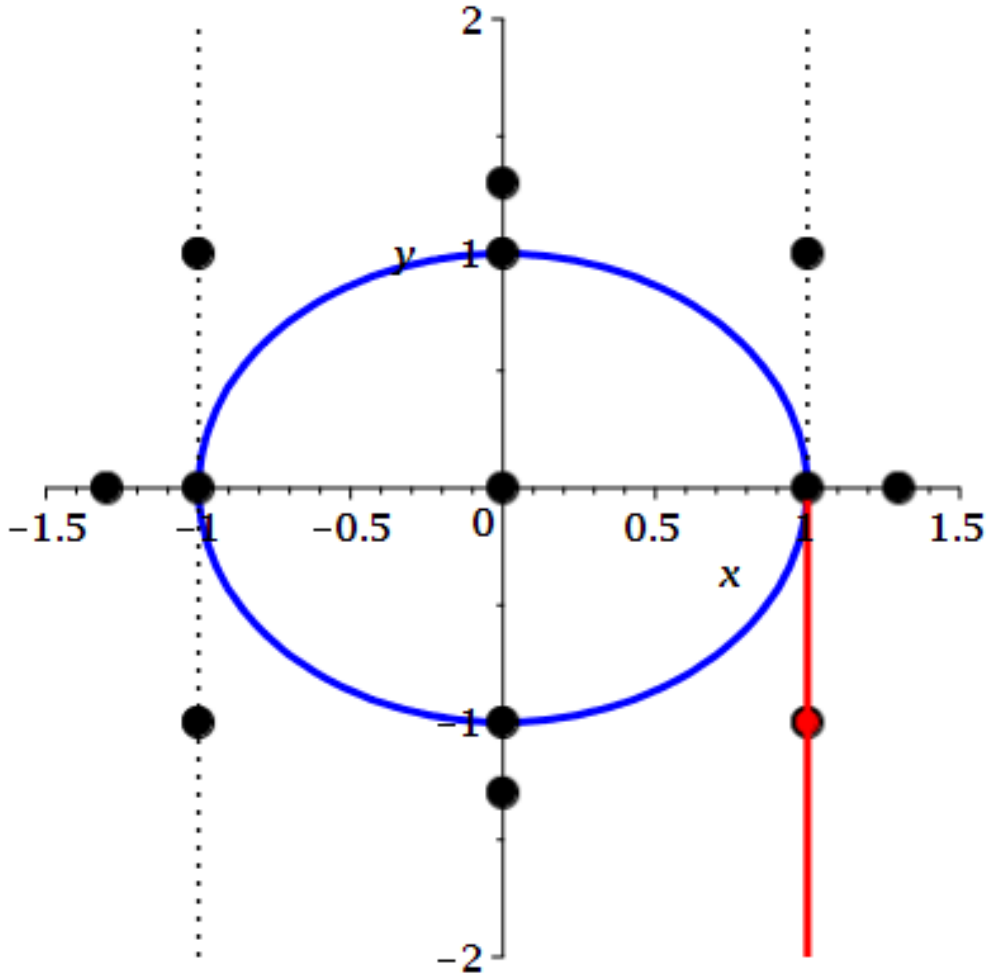
Example: Circle — visualisation

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



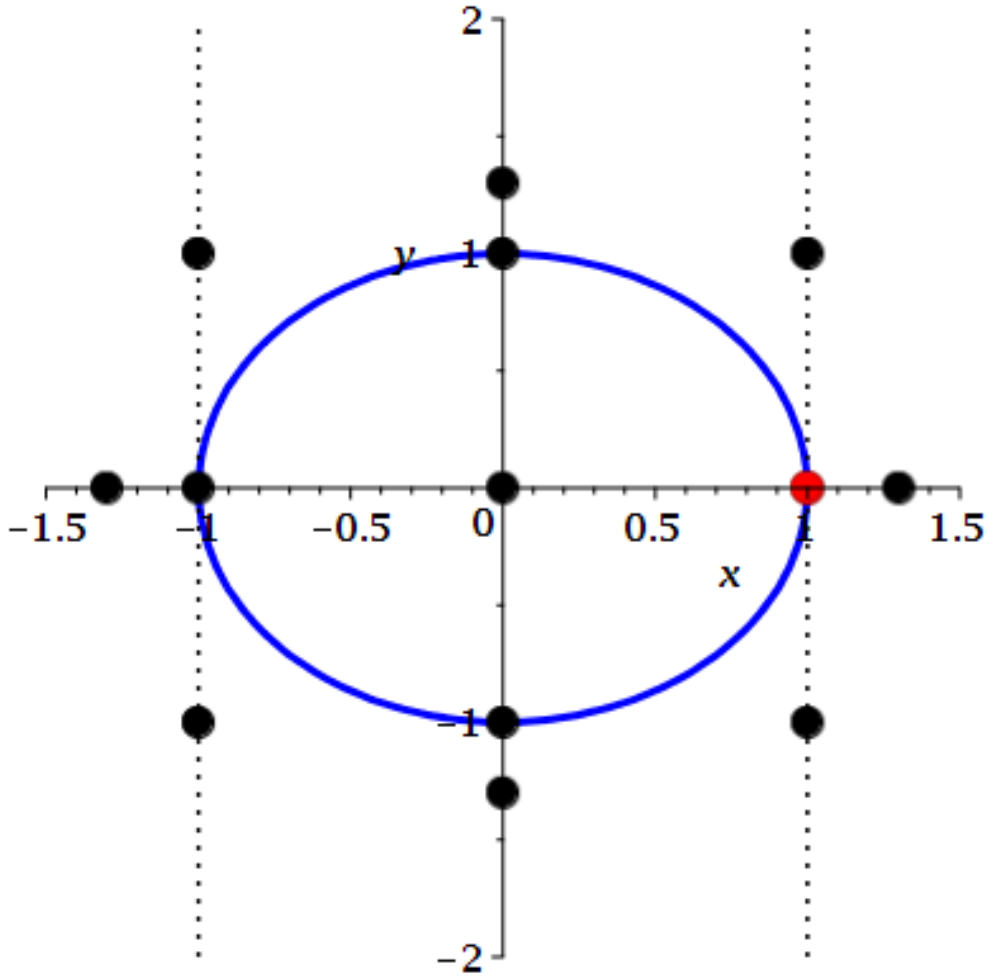
Example: Circle – visualisation

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



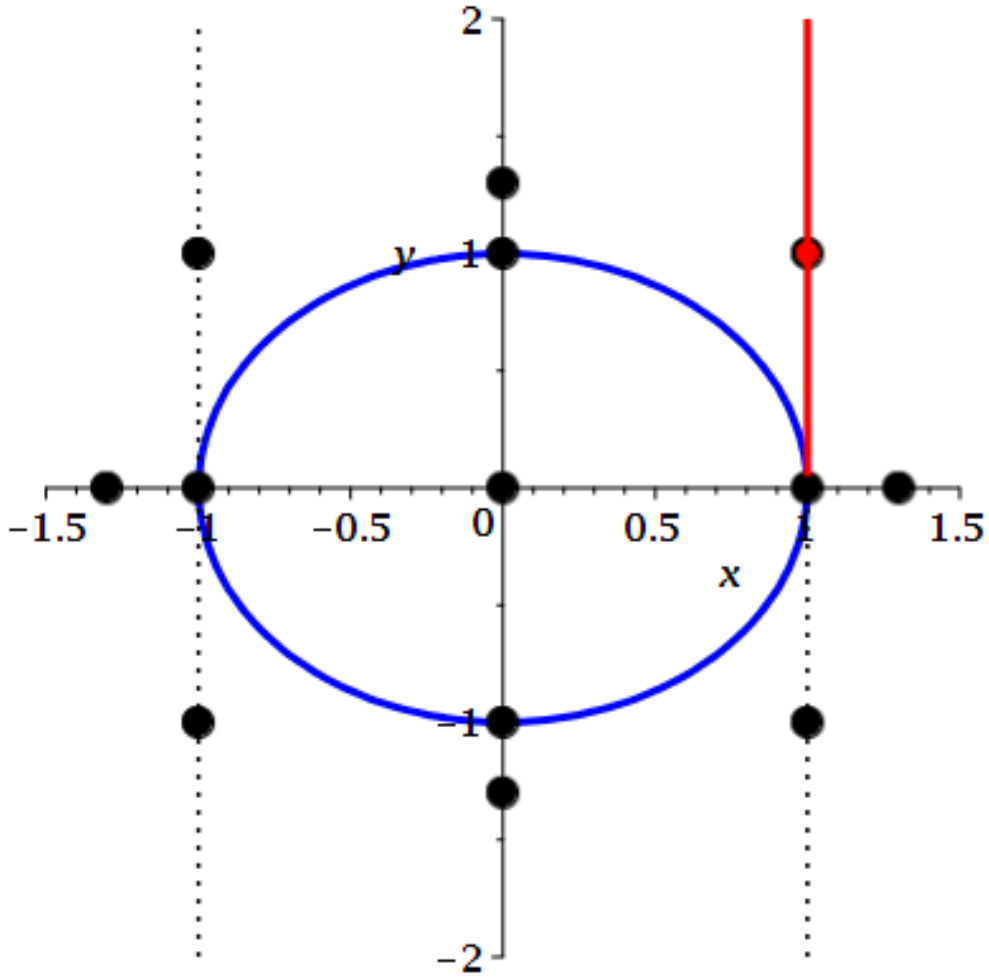
Example: Circle – visualisation

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



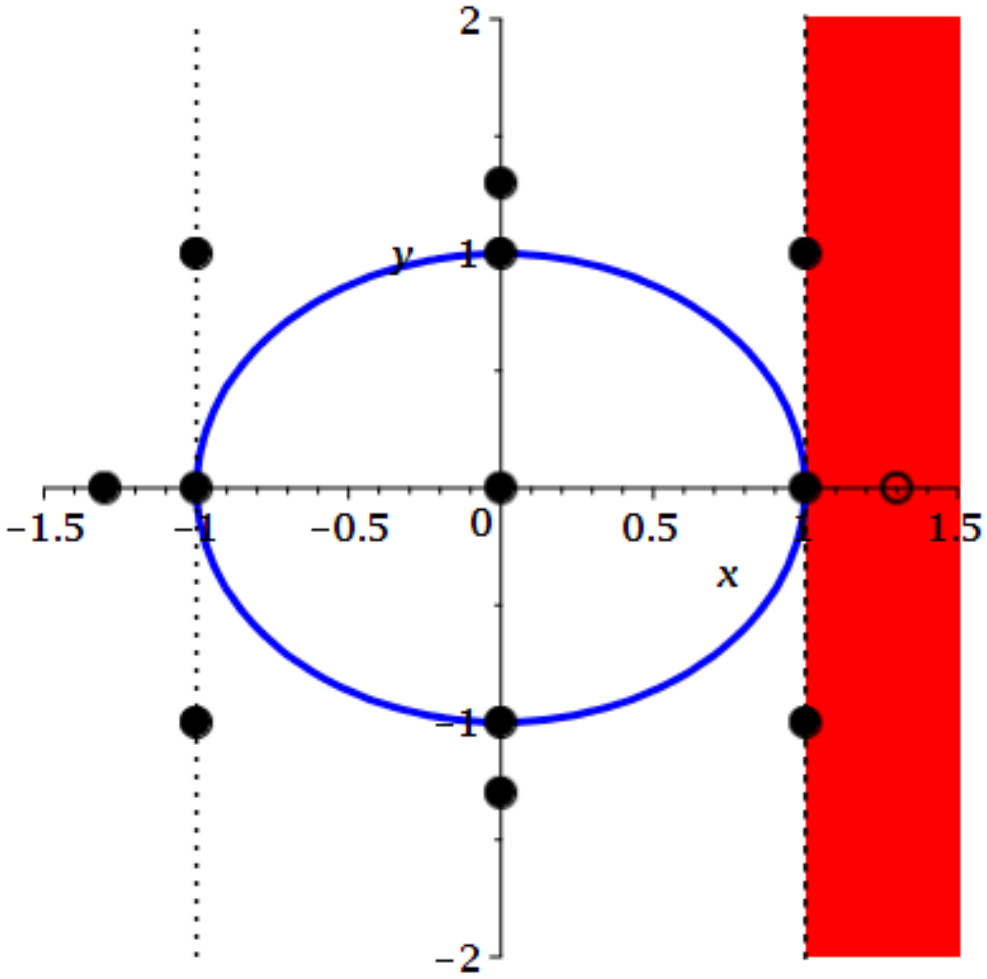
Example: Circle – visualisation

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



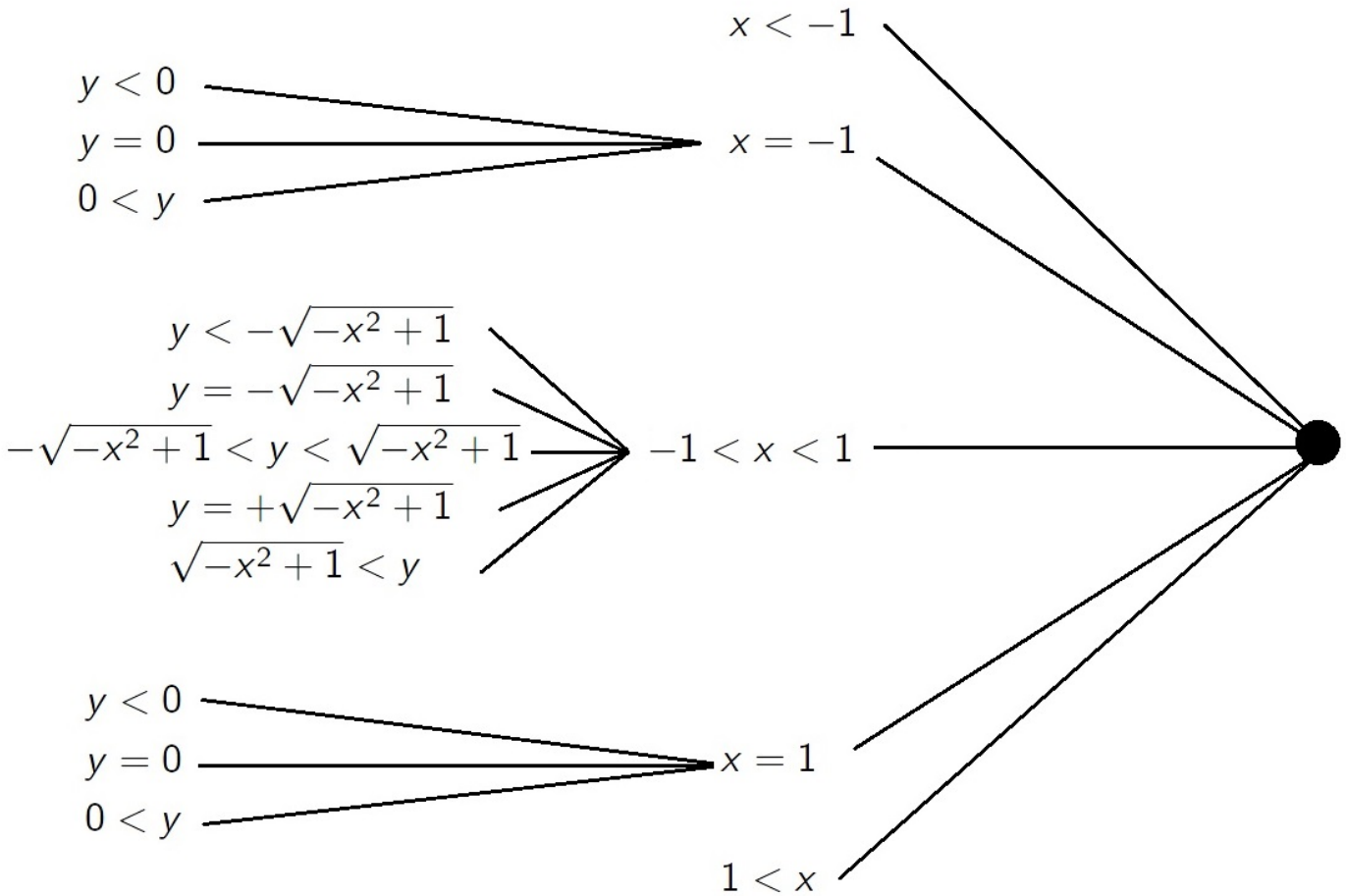
Example: Circle – visualisation

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 – cylindrical tree

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



Outline

Motivation: Real QE

Real Quantifier Elimination (Real QE)

Given: Quantified formulae in prenex form with atoms integral polynomial constraints.
Produce: quantifier free formula logically equivalent over $\mathbb{R}$.

Fully quantified

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

Output: `False`

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

Output: `True`

Partially quantified

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

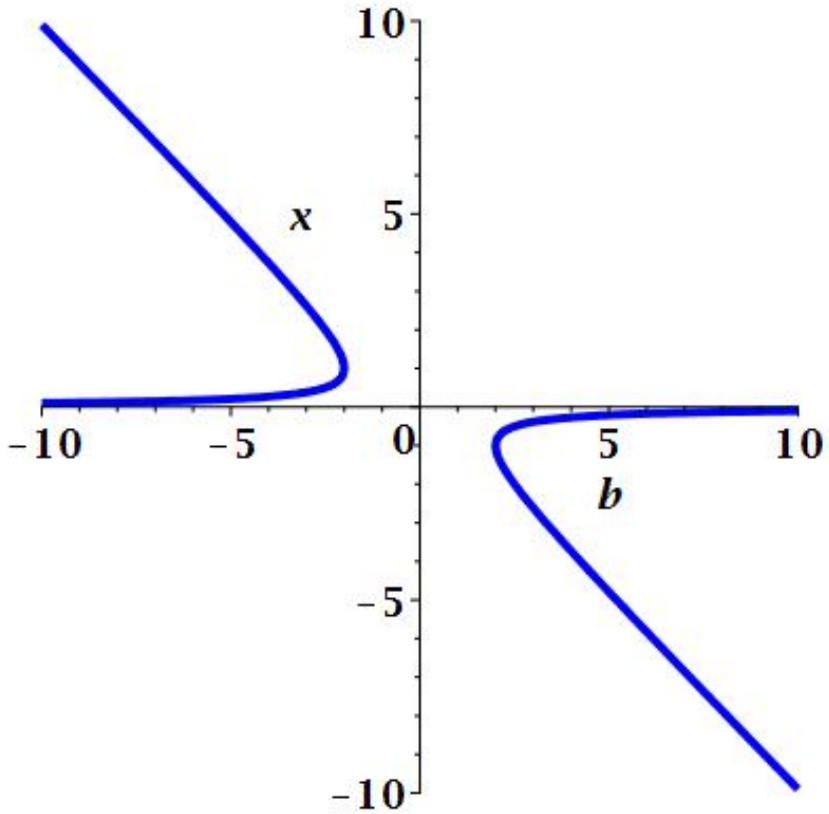
Output: $(b \leq -2) \vee (b > 2)$

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

QE via CAD Example

How to determine with CAD?

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



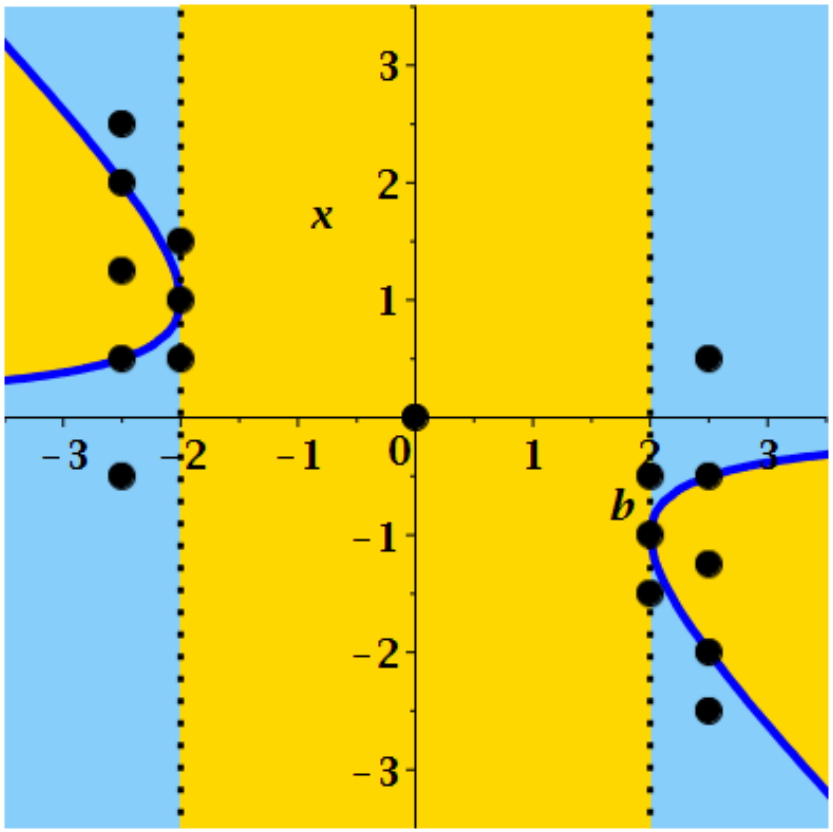
QE via CAD Example

How to determine with CAD?

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

To solve we:

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



QE via CAD Example

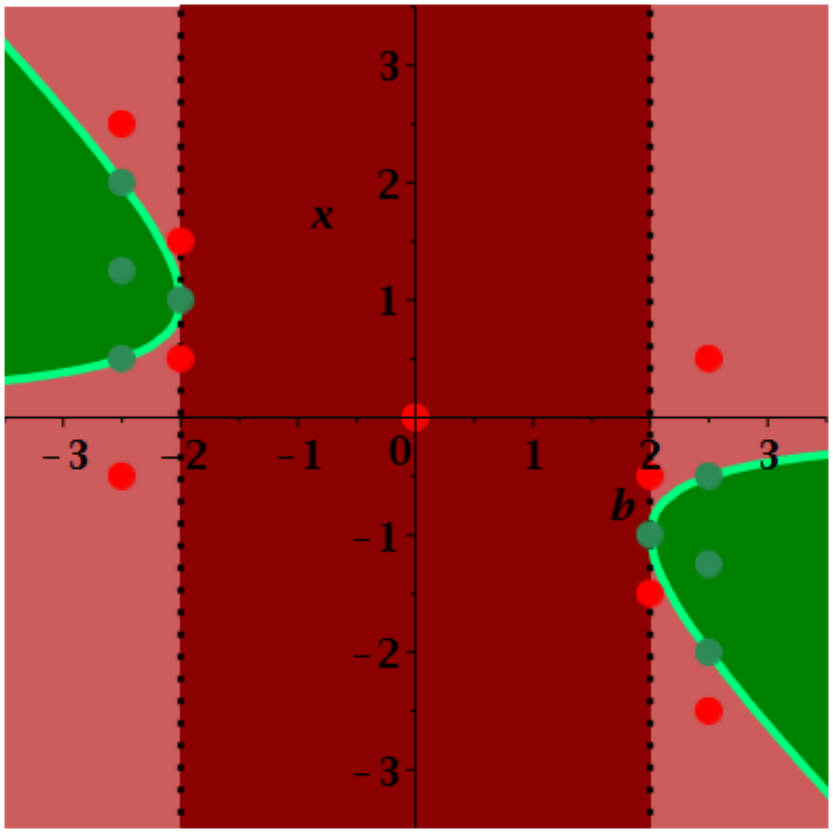
How to determine this CAD?

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

To solve we:

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

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



QE via CAD Example

How to determine with CAD?

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

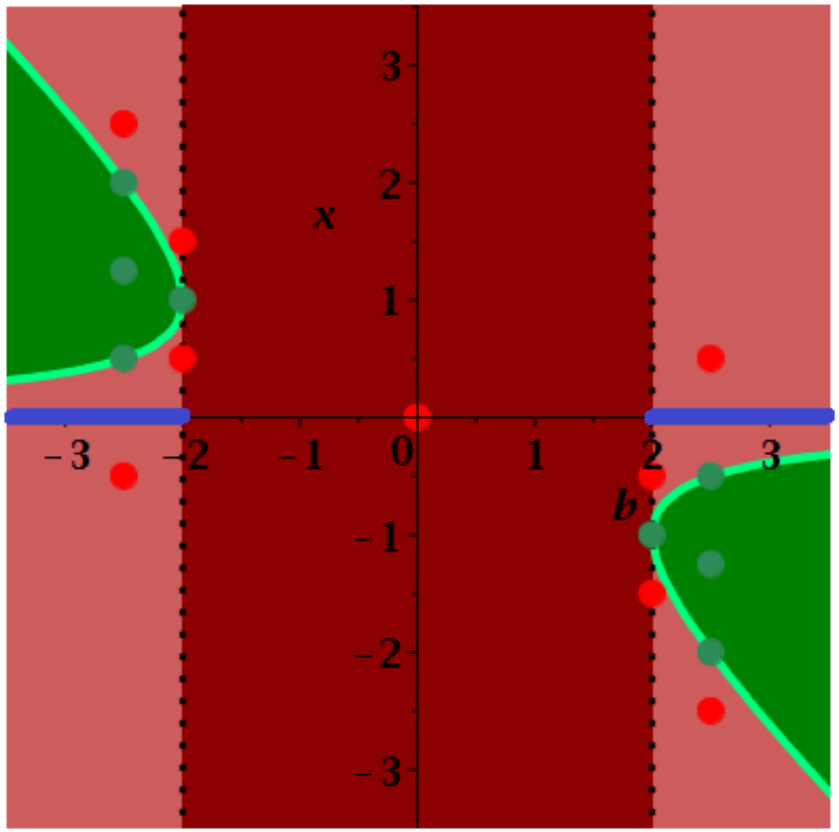
To solve we:

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

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

Take disjunction of projections of true cells:

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



QE via CAD Example

How to determine with CAD?

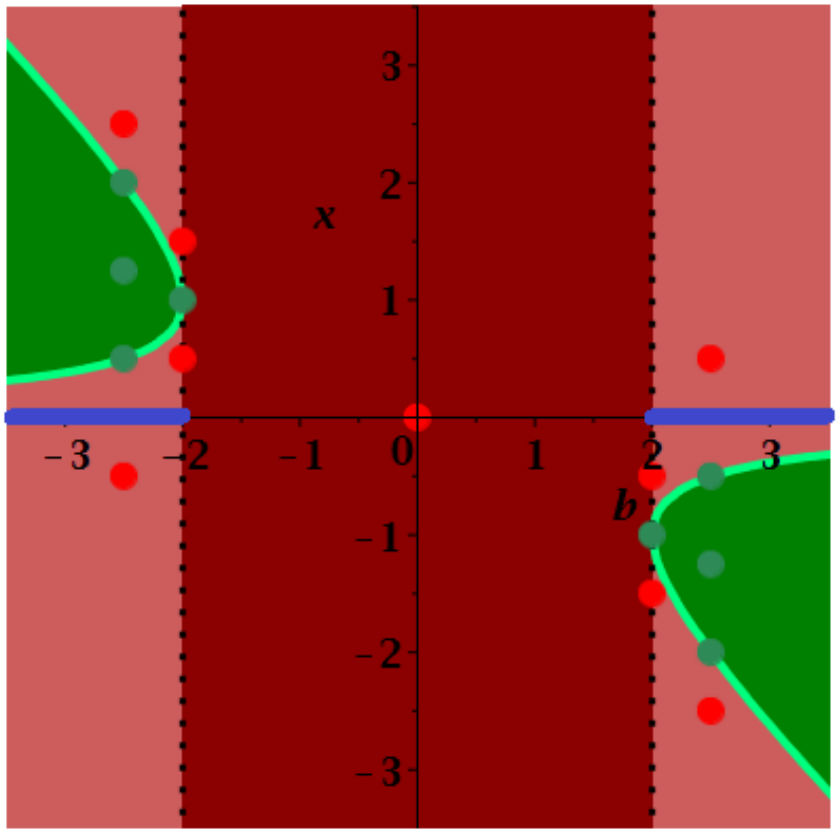
$$\exists x, x^2 + bx + 1 \leq 0$$

To solve we:

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

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

Take disjunction of projections of true cells:



$$\Longrightarrow$$

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

Outline

QE Implementations and Applications

So existential QE via projection of true CAD cells onto free variables. Universal QE via $\forall x F(x) = \neg\exists x \neg F(x)$.

QE and CAD implementations are traditionally found in Computer Algebra Systems, e.g. MAPLE (RegularChains, SyNRAC), MATHEMATICA, REDUCE (Redlog), QEPCAD-B.
CAD also in the SMT-solvers, SMT-RAT, YICES, Z3 and CVC5.

QE can solve problems throughout engineering & science. E.g.

- derivation of optimal numerical schemes (Erascu-Hong, 2014)
- automated theorem proving (Paulson, 2012)
- automated loop parellisation (Grösslinger et al. 2006)
- analysis of economic hypotheses (Mulligan et al., 2018)

Also CAD applications independent of QE, e.g. multi-stationarity identification in chemical reaction networks (Bradford et al. 2017).

QE Implementations and Applications

So existential QE via projection of true CAD cells onto free variables. Universal QE via $\forall x F(x) = \neg\exists x \neg F(x)$.

QE and CAD implementations are traditionally found in Computer Algebra Systems, e.g. MAPLE (`RegularChains`, `SyNRAC`), MATHEMATICA, REDUCE (`Redlog`), QEPCAD-B.
CAD also in the SMT-solvers, SMT-RAT, YICES, Z3 and CVC5.

QE can solve problems throughout engineering & science. E.g.

- derivation of optimal numerical schemes (Erascu-Hong, 2014)
- automated theorem proving (Paulson, 2012)
- automated loop parellisation (Grösslinger et al. 2006)
- analysis of economic hypotheses (Mulligan et al., 2018)

Also CAD applications independent of QE, e.g. multi-stationarity identification in chemical reaction networks (Bradford et al. 2017).

QE Implementations and Applications

So existential QE via projection of true CAD cells onto free variables. Universal QE via $\forall x F(x) = \neg\exists x \neg F(x)$.

QE and CAD implementations are traditionally found in Computer Algebra Systems, e.g. MAPLE (`RegularChains`, `SyNRAC`), MATHEMATICA, REDUCE (`Redlog`), QEPCAD-B.
CAD also in the SMT-solvers, SMT-RAT, YICES, Z3 and CVC5.

QE can solve problems throughout engineering & science. E.g.

- derivation of optimal numerical schemes (Erascu-Hong, 2014)
- automated theorem proving (Paulson, 2012)
- automated loop parellisation (Grösslinger et al. 2006)
- analysis of economic hypotheses (Mulligan et al., 2018)

Also CAD applications independent of QE, e.g. multi-stationarity identification in chemical reaction networks (Bradford et al. 2017).

CAD Complexity

To build a CAD we repeatedly project polynomials to encode key geometric information. This is followed by repeated use of real root isolation and the substitution of sample points to render multivariate polynomials univariate. We can isolate at sample points and infer properties throughout a cell due as the information tracked by projection ensures the relevant cell boundaries.

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

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

CAD Complexity

To build a CAD we repeatedly project polynomials to encode key geometric information. This is followed by repeated use of real root isolation and the substitution of sample points to render multivariate polynomials univariate. We can isolate at sample points and infer properties throughout a cell due as the information tracked by projection ensures the relevant cell boundaries.

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

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

Outline

Satisfiability in NRA

We now consider the problem of determining the satisfiability of a formula in Non-linear Real Arithmetic (NRA). I.e. to evaluate

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

as either True (SAT) or False (UNSAT) where F is a formula in Boolean logic (atoms connected by $\wedge, \vee, \neg$) whose atoms are sign constraints on non-linear multivariate polynomials with integer coefficients. (Usually assume F is in conjunctive normal form.)

This is a sub-problem of Real QE and thus can be solved by CAD. But as a problem it has lower (single exponential) complexity.

A sign-invariant CAD for the polynomials in F can be used to solve any such problem, regardless of the particular Boolean structure involved. How to adapt CAD to take note of the logic?

Satisfiability in NRA

We now consider the problem of determining the satisfiability of a formula in Non-linear Real Arithmetic (NRA). I.e. to evaluate

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

as either True (SAT) or False (UNSAT) where F is a formula in Boolean logic (atoms connected by $\wedge, \vee, \neg$) whose atoms are sign constraints on non-linear multivariate polynomials with integer coefficients. (Usually assume F is in conjunctive normal form.)

This is a sub-problem of Real QE and thus can be solved by CAD. But as a problem it has lower (single exponential) complexity.

A sign-invariant CAD for the polynomials in F can be used to solve any such problem, regardless of the particular Boolean structure involved. How to adapt CAD to take note of the logic?

The SMT Approach

One approach to such satisfiability problems is to separate out the logic from the arithmetic theory.

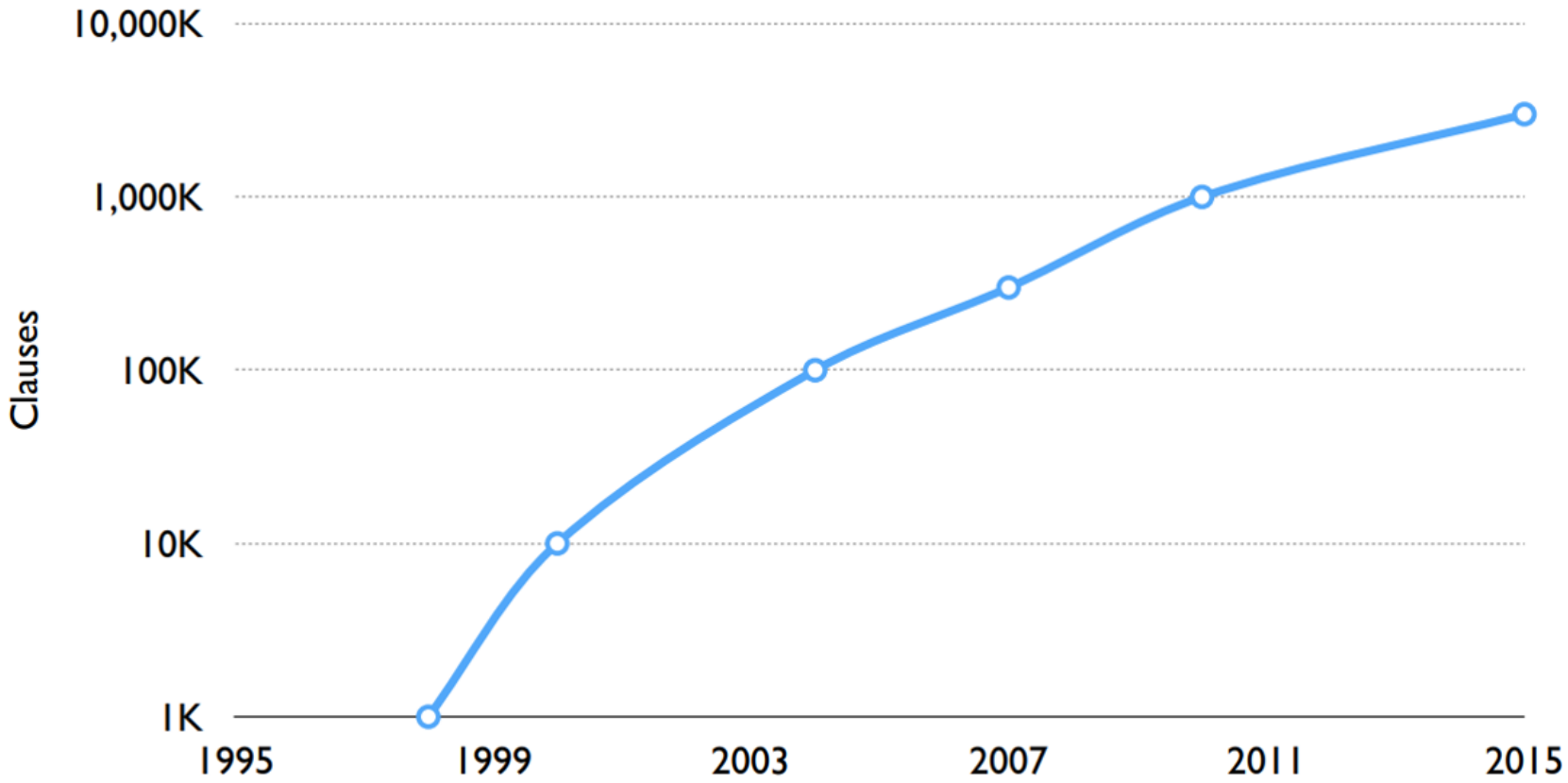
- Allow the logical structure to be explored by a **SAT Solver**.
- Have the solutions proposed be tested in the theory of interest by relevant software for that domain: a **Theory Solver**.

The Theory Solver need only test the consistency of a set of constraints (no Boolean logic).

This approach is called **Satisfiability Modulo Theories** (SMT).

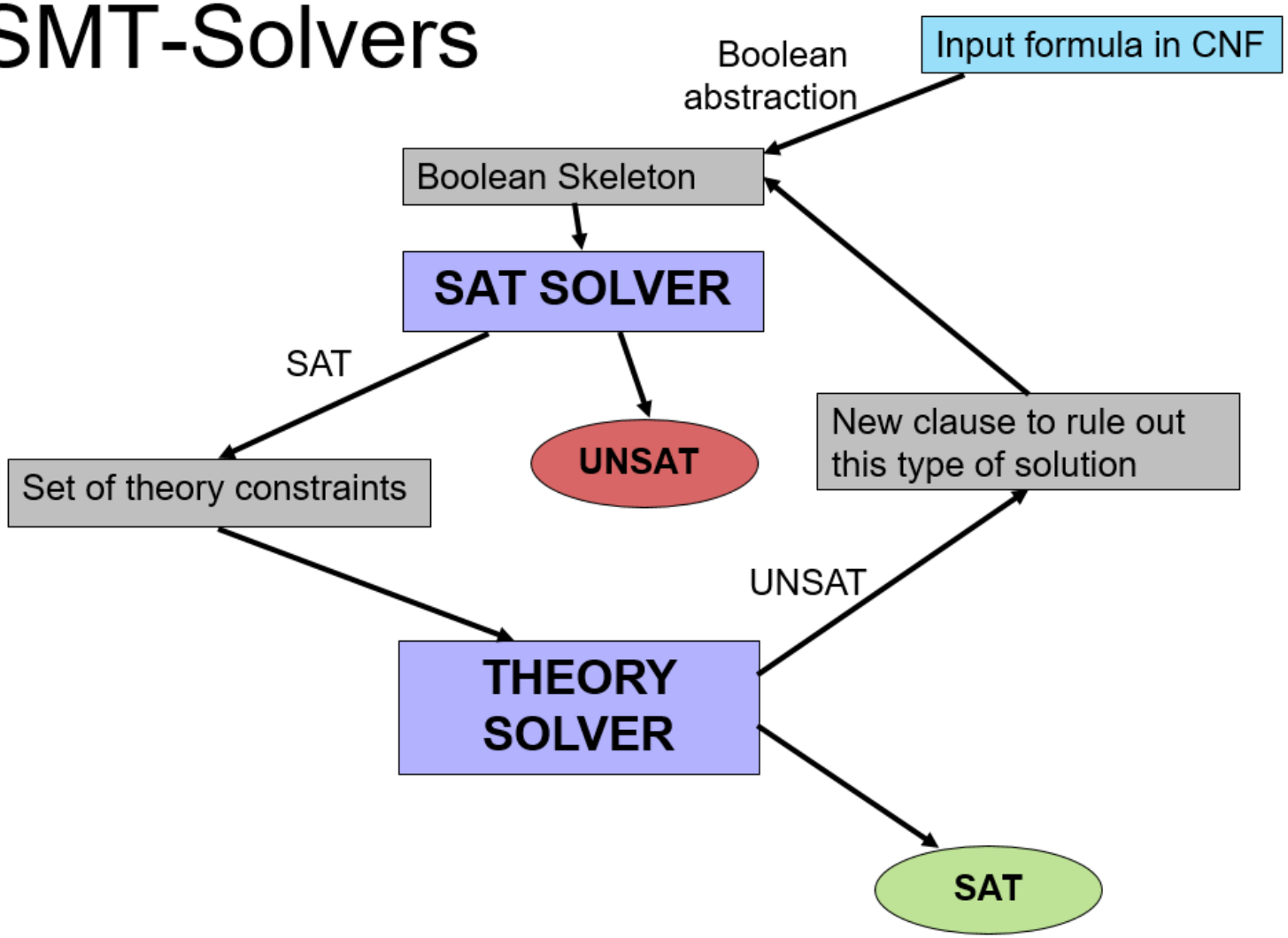
Why the SMT Approach?

To take advantage of the incredible progress in SAT solvers!



Based on a slide from Vijay Ganesh

SMT-Solvers


Input formula in CNF
Boolean abstraction
Boolean Skeleton
SAT SOLVER
SAT
UNSAT
New clause to rule out this type of solution
Set of theory constraints
UNSAT
THEORY SOLVER
SAT

CAD as NRA Theory Solver

We can use CAD as SMT NRA theory solver but it must be adapted for this use:

- **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.

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.

CAD as NRA Theory Solver

We can use CAD as SMT NRA theory solver but it must be adapted for this use:

- **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.

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.

How good is this approach?

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 to adapt CAD further to avoid this?
The following approaches try to success of SAT-solvers which search the sample spaces by: making guesses, propagating, and generalising conflicts to avoid similar parts of the search space.

Outline

The NLSAT Algorithm

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

The NLSAT used an alternative to the SMT framework (called MCSAT) which does not separate SAT-solver and Theory solver.

- Partial model solutions for the Boolean skeleton and the algebraic theory are built in parallel.
- Boolean conflicts generalised using the CDCL approach of modern SAT solvers (Marques-Silva and Sakallah).
- 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.

NLSAT Implications

Less projection (only polynomials in the conflict) and less root isolation (only once per level). Other 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 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 SMT framework?

Conflict Driven Cylindrical Algebraic Covering (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 .
- Cells are arranged cylindrically.
- Structured guidance of search as part of the algebraic procedure.

CDCAC: Basic Idea

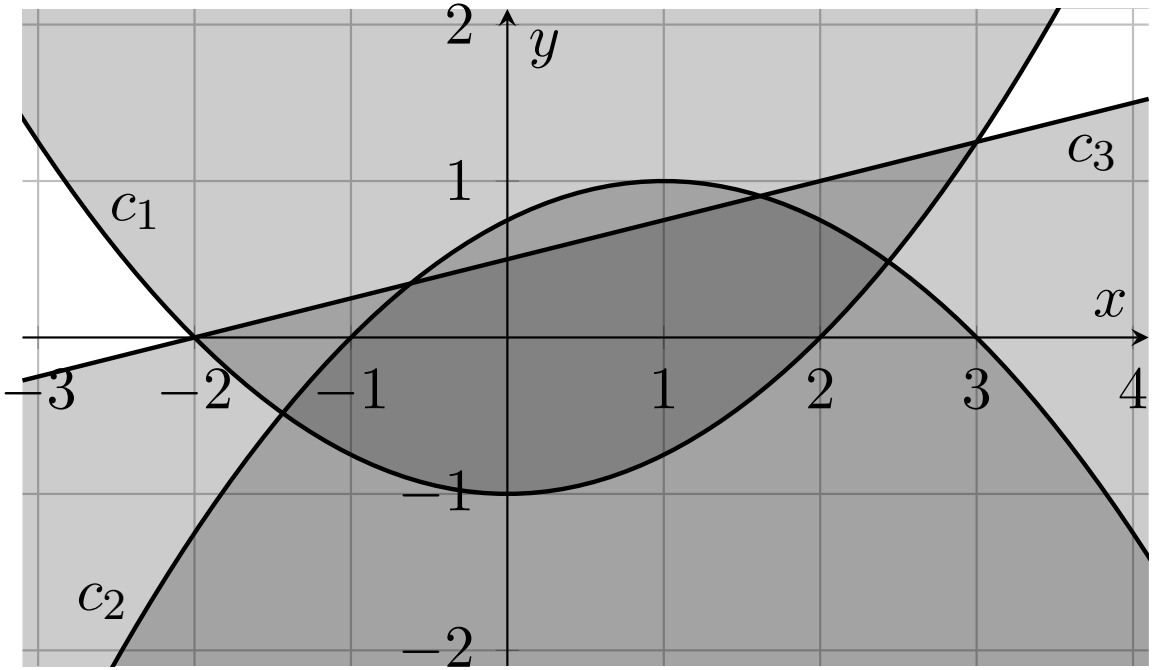
- Pick sample for lowest variable in ordering.
- Extend to increasingly higher dimensions in reference to those constraints made univariate.
- If all constraints satisfied then conclude SAT.
- If a constraint cannot be satisfied generalise to CAD cell in current dimension.
- Search outside the cell in 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.

RWTH AACHEN UNIVERSITY

An example

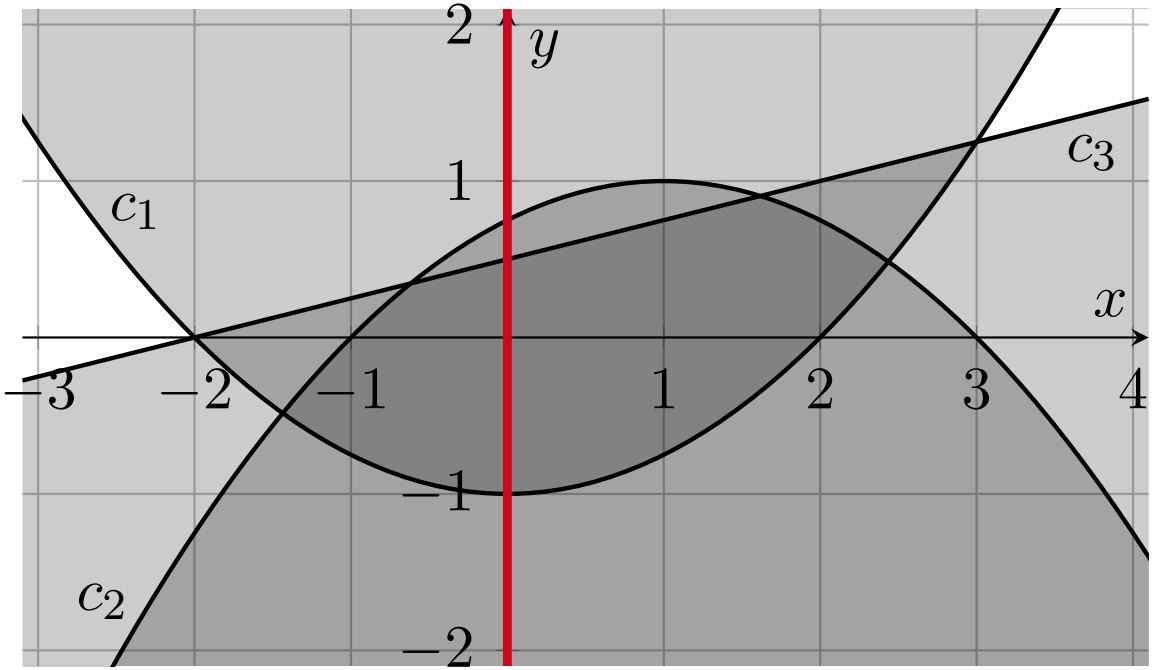
$c_1 : 4 \cdot y < x^2 - 4$ $\quad$ $c_2 : 4 \cdot y > 4 - (x-1)^2$ $\quad$ $c_3 : 4 \cdot y > x + 2$



RWTH AACHEN UNIVERSITY

An example

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



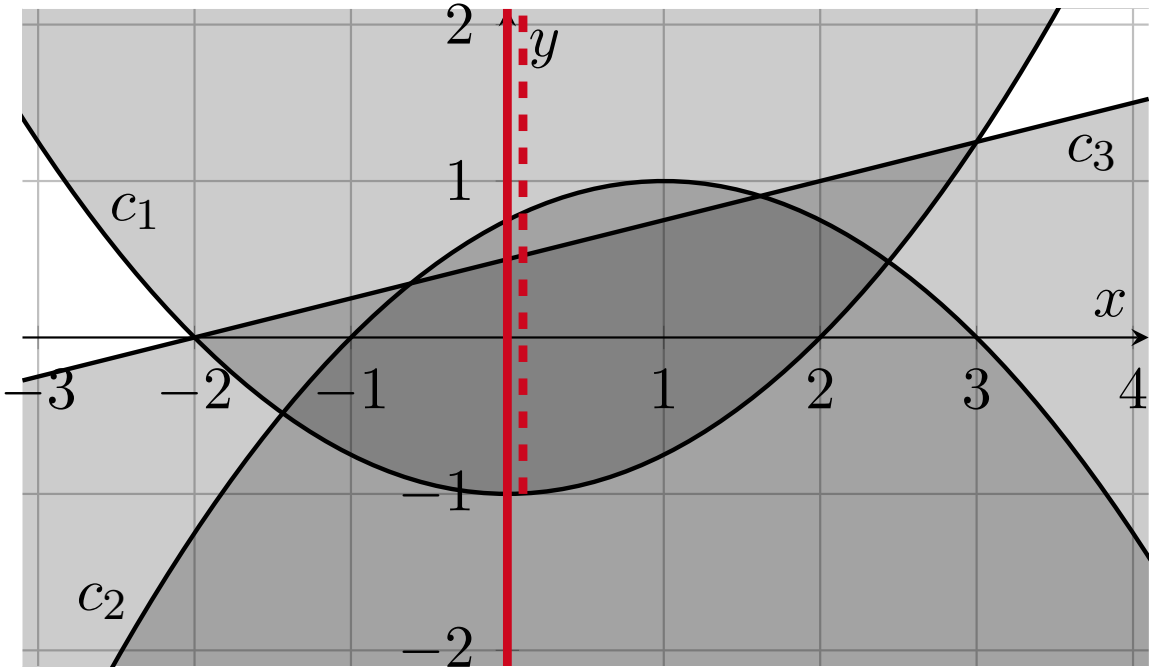
No constraint for x

Guess $x \mapsto 0$

RWTH AACHEN UNIVERSITY

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

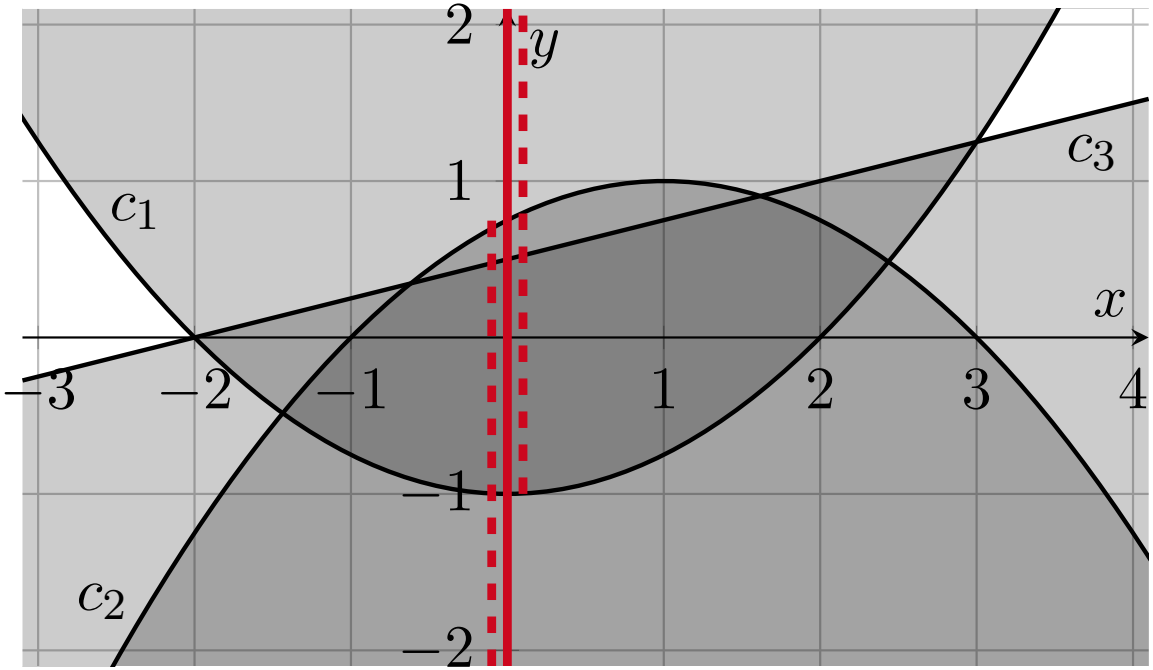
Guess $x \mapsto 0$

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

RWTH AACHEN UNIVERSITY

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 $x \mapsto 0$

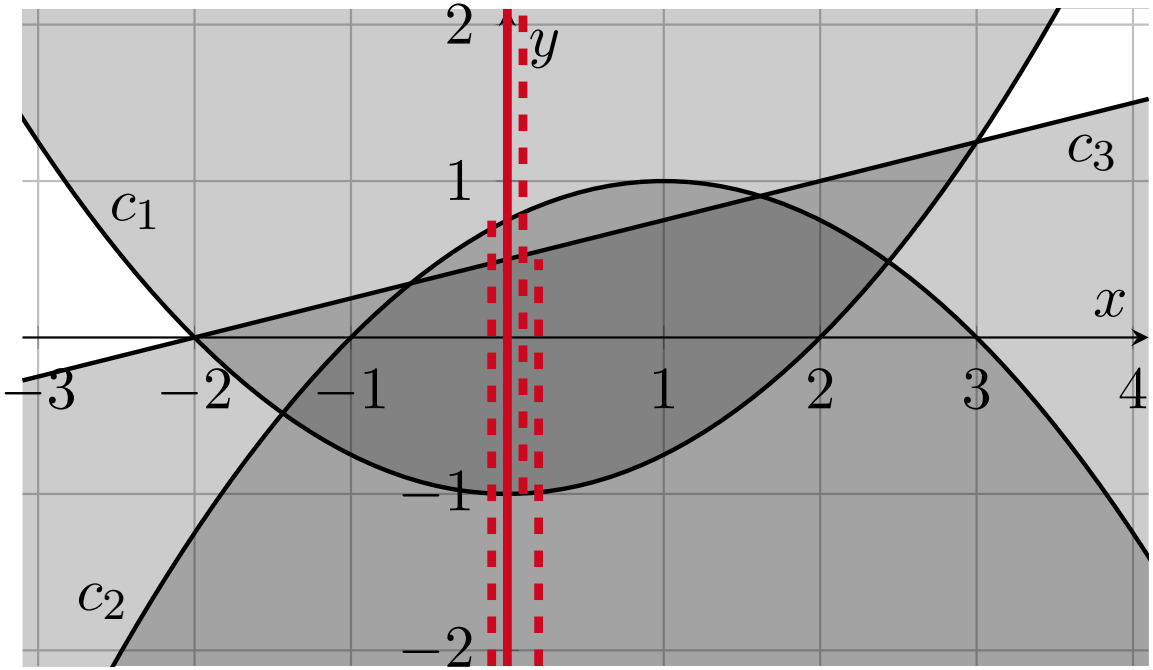
$c_1 \to y \notin (-1, \infty)$

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

RWTH AACHEN UNIVERSITY

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 $x \mapsto 0$

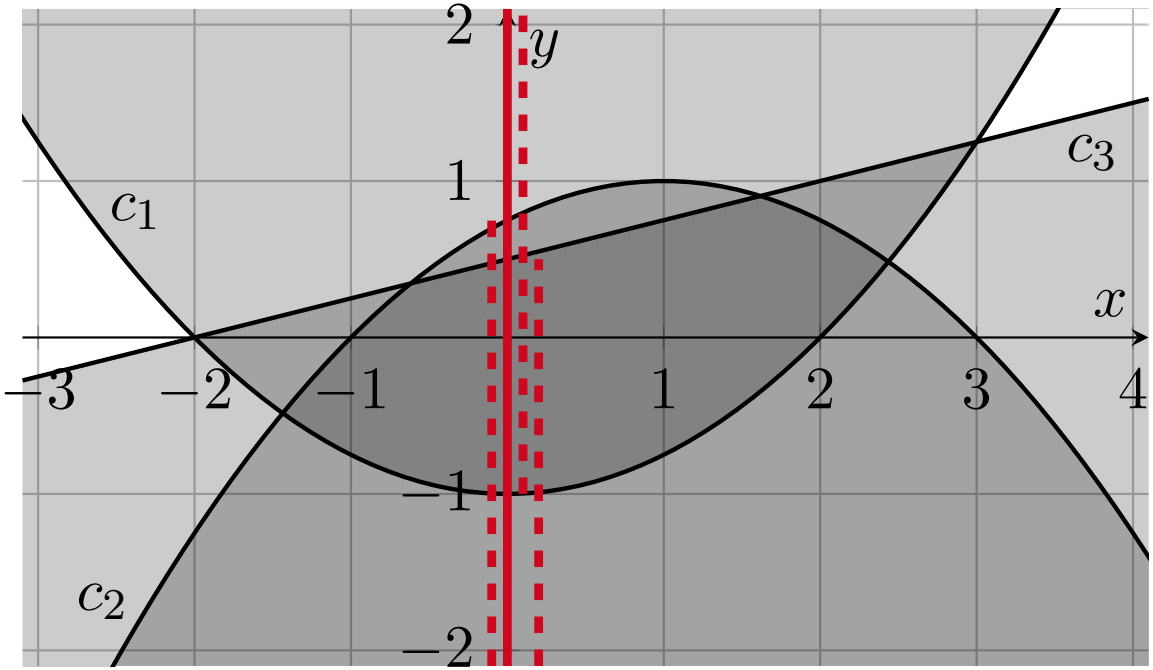
$c_1 \to y \notin (-1, \infty)$

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

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

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 $x \mapsto 0$

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

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

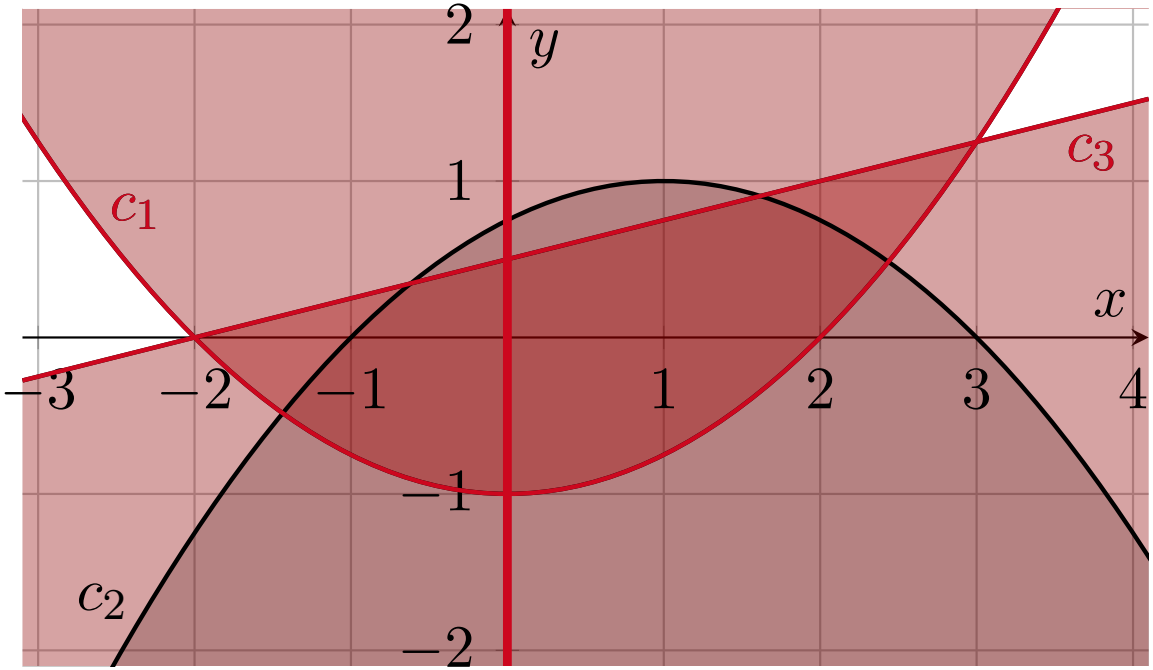
$c_3 \to y \notin (-\infty, 0.5)$

Construct covering

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

An example

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



No constraint for x

Guess $x \mapsto 0$

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

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

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

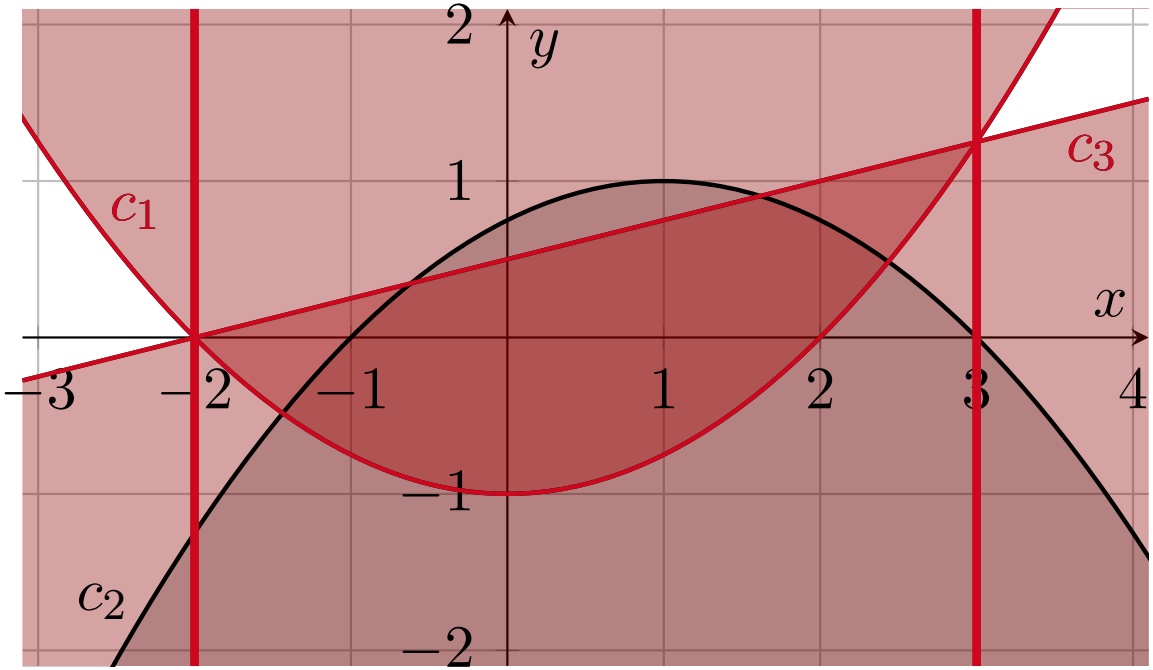
Construct covering

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

RWTH AACHEN UNIVERSITY

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 $x \mapsto 0$

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

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

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

Construct covering

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

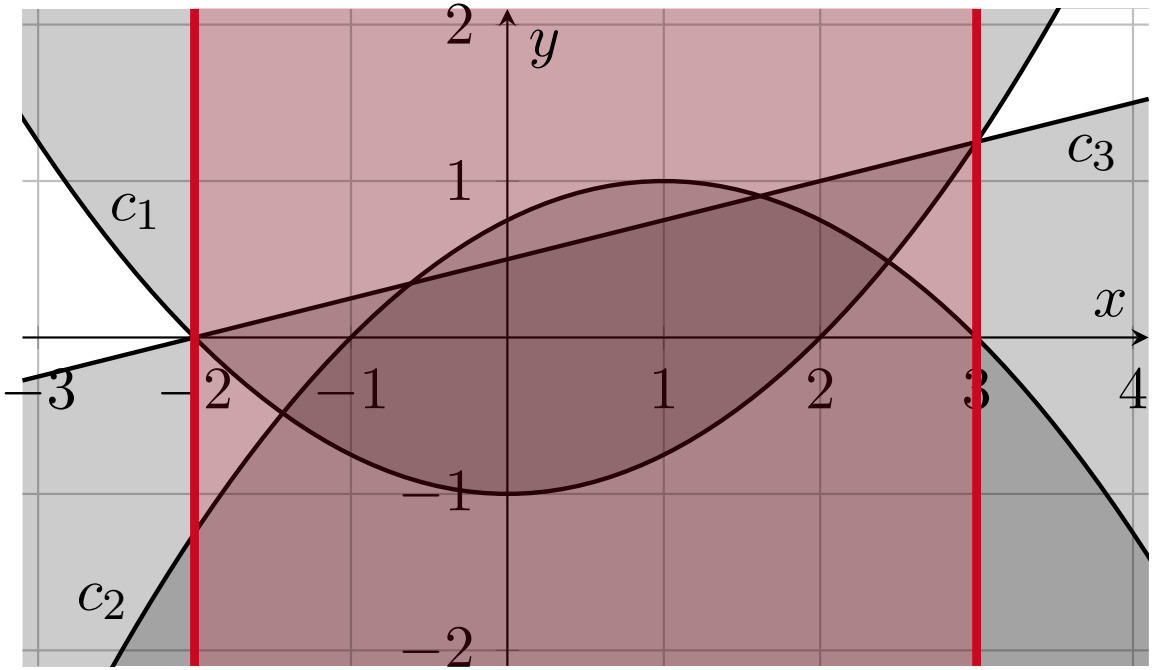
Construct interval for x

$x \notin (-2, 3)$

RWTH AACHEN UNIVERSITY

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 $x \mapsto 0$

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

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

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

Construct covering

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

Construct interval for x

$x \notin (-2, 3)$

New guess for x

CDCAC Experimental Results

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 that 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.

  So algorithms are clearly different - potential for meta-solver?

A new implementation of CDCAC in CVC5 outperforms the winner of the 2020 SMT Competition (which was based on NLSAT). These experiments still to be published (paper under review).

CDCAC Experimental Results

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 that 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.

  So algorithms are clearly different - potential for meta-solver?

A new implementation of CDCAC in CVC5 outperforms the winner of the 2020 SMT Competition (which was based on NLSAT). These experiments still to be published (paper under review).

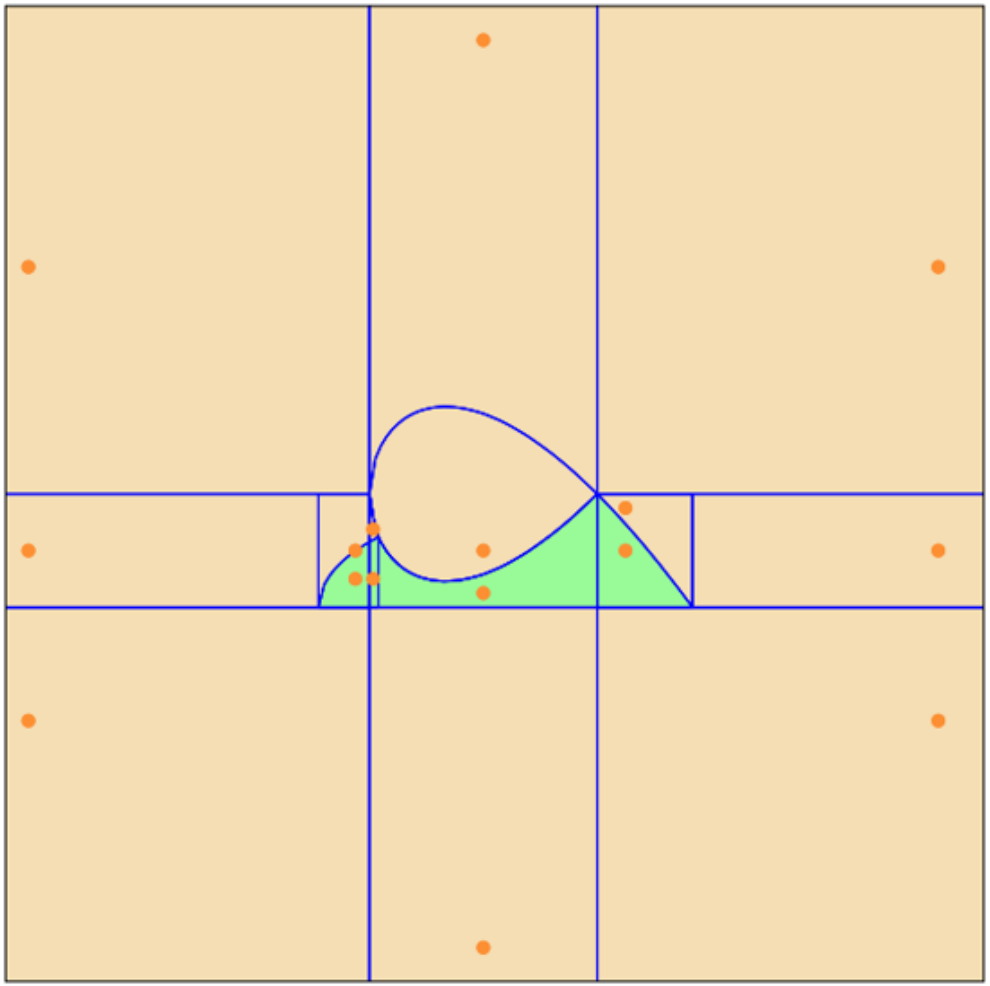
Outline

NuCAD

Recall that CDCAC an alternative to NLSAT which also produces cylindrical coverings instead of decompositions.
Non-uniformly cylindrical CAD (NuCAD) was also inspired by NLSAT. This produces a decomposition (not covering) but with the cells not arranged cylindrically. Proposed by Brown in 2015 although only for open cells. In 2017 its potential use for Quantifier Elimination was outlined.


C. Brown.
Open non-uniform cylindrical algebraic decompositions.
Proc. ISSAC 2015 pages 85-92. ACM, 2015.
https://doi.org/10.1145/2755996.2756654

NuCAD Example



Future Work: DEWCAD

The author has recently started work on a new project at Coventry University and the University of Bath, funded by the UK Engineering and Physical Science Research Council (EPSRC).

DEWCAD: Pushing Back the Doubly Exponential Walls of Cylindrical Algebraic Decomposition
`https://matthewengland.coventry.domains/dewcad/`

The first deliverable is a Maple package for QE and Satisfiability in NRA which implements incremental CAD, NLSAT, NuCAD and CDCAC to allow for meaningful comparison of the algorithms. This will be followed by further theory development and applications work in chemical reaction networks and automated economics reasoning.

Outline

CAD Variable Ordering

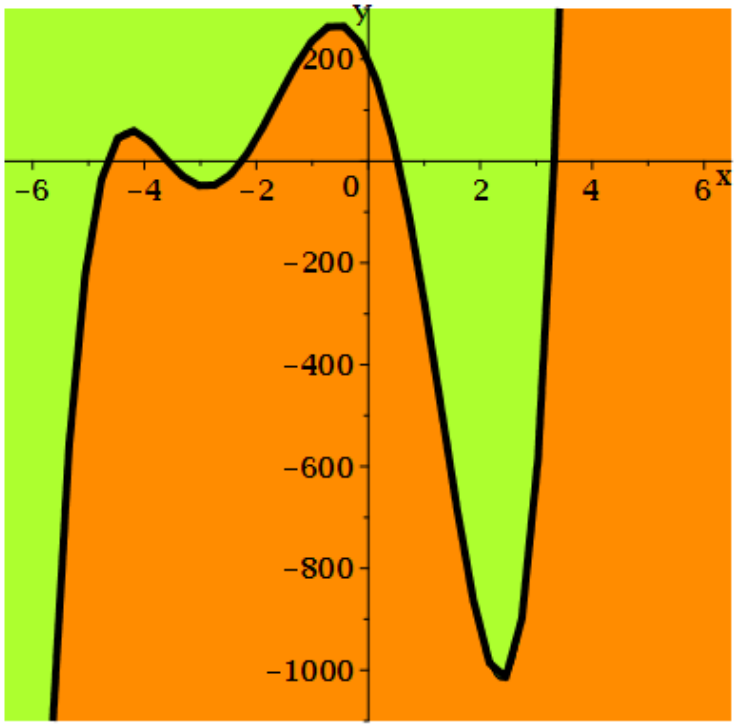
CADs are defined with respect to an ordering on variables (for cylindricity, projection etc.) For QE one must order variables as they are quantified; but there is no restriction on free variables and adjacent quantifiers of the same type may be swapped. Thus for SMT on NRA with n real variables we have $n!$ choices.

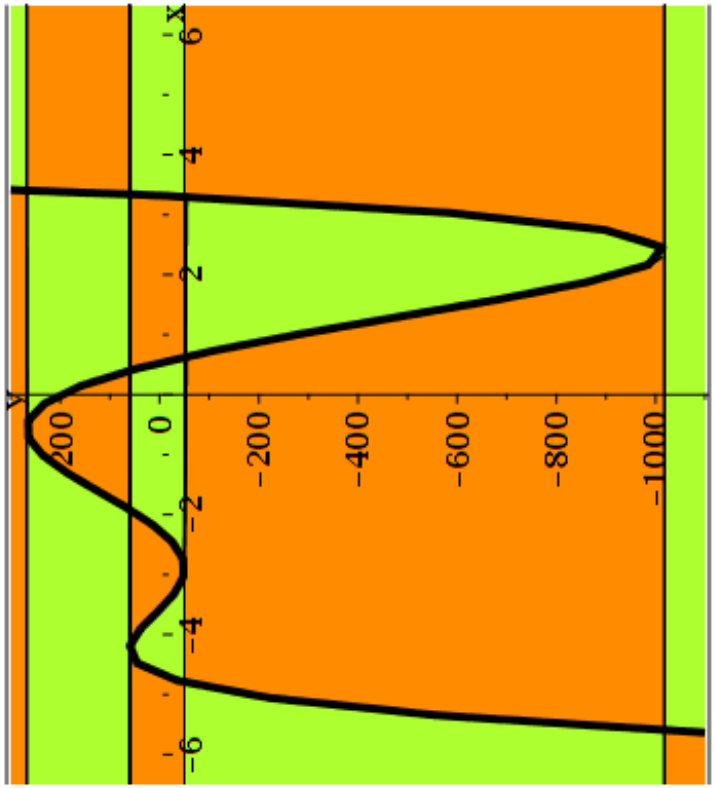
Any choices leads to a mathematically valid answer. But it is well observed that choice of variable ordering can dramatically affect both the number of cells in the decomposition and the time required to compute them, often to the point of feasibility.

Can such choices be made by Machine Learning?

CAD Variable Ordering Example

CAD for polynomial $y - 3x^5 + 20x^4 - 10x^3 - 240x^2 - 250x + 200$.
With $y \succ x$ a sign-invariant CAD has 3 cells, with $y \prec x$ it is 59.





Some Human Designed Heuristics


C. Brown.
Tutorial Notes: Cylindrical algebraic decomposition.
Presented at ISSAC 2004.


A. Dolzmann, A. Seidl and T. Sturm . In: Proc.
Efficient projection orders for CAD.
Proc. ISSAC 2004, pp.111-118, ACM (2004).


R. Bradford, M. England, J.H. Davenport and D. Wilson.
Optimising problem formulations for cylindrical algebraic decomposition.
Proc. CICM 2013, LNCS 7961, pp. 19-34. Springer 2013.


D. Wilson, M. England, R. Bradford and J.H. Davenport.
Using the distribution of cells by dimension in a cylindrical algebraic decomposition.
Proc. SYNASC 2014, pp. 53-60. IEEE 2014.

Summary: Increasingly expensive; none of them are perfect.

Recent Development: Chordality Based Heuristic

H. Li, B. Xia, H. Zhang and T. Zheng.
Choosing the Variable Ordering for Cylindrical Algebraic Decomposition via Exploiting Chordal Structure.
To Appear: Proc. ISSAC 2021. ACM 2021.
Preprint: `https://arxiv.org/abs/2102.00823`

When variables sparsely distributed can find ordering which preserves this: essentially projections sent polynomials down more than one level, in turn reducing the double exponent of the growth in degree and number of polynomials.

However, even then there are instances where a heuristic based on this is misled, or where it cannot discriminate.

Outline

Huang et al. (CICM 2014)

Z. Huang, M. England, D. Wilson, J.H. Davenport, L.C. Paulson and J. Bridge.
Applying machine learning to the problem of choosing a heuristic to select the variable ordering for cylindrical algebraic decomposition.
Proc. CICM 2014, LNAI 8543, pp. 92-107. Springer 2014.

Used a Support Vector Machine (SVM) to choose which of three human made heuristics to follow when picking an ordering.

- Experiments on 7000 problems identified substantial subclasses on which each made a better decision.
- Trained three SVMs and used relative magnitude of their margin values to pick which heuristic to follow.
- ML choice did significantly better than any one heuristic.

With Florescu at CICM 2019

Repeated Huang *et al.*'s experiments but this time:

- Choose variable ordering directly.

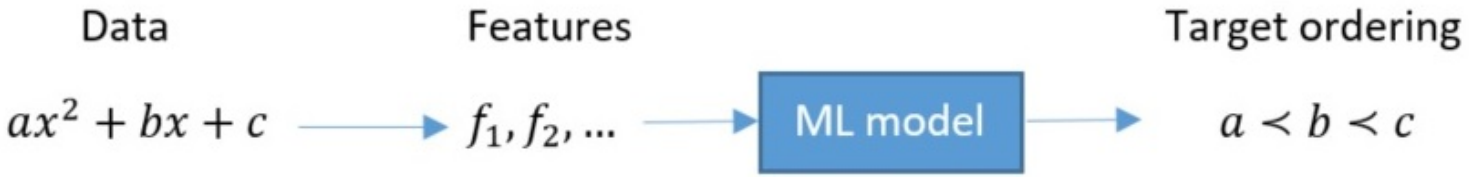
Many examples where no human-made heuristic had a good choice: greater savings but harder to scale for larger n.

- Experimented with different ML classifiers
  - Support Vector Machine (SVM) classifier with RBF kernel.
  - K−Nearest Neighbours (KNN) classifier.
  - Multi-Layer Perceptron (MLP) classifier.
  - Decision Tree (DT) classifier.

All did better than human-made heuristics but SVM actually beaten significantly by the other three.

ML Features

The ML classifiers are trained not on the input polynomials but vectors of real numbers derived from them. Each of these numbers corresponds to a feature of the input.



Work above used 11 features inspired by Brown's heuristic, e.g.:

- Degree of a variable in the input.
- Proportion of input polynomials which contact a variable.
- Proportion of input monomials which contact a variable.

(Q) Are there more / better features we can extract from the input polynomials without resorting to expensive projection operations?

With Florescu at SC^2 2019 Paper

Presented a framework to enumerate (all appropriate) combinations of some (basic) functions on a set of polynomials. This encompasses all previously used features but also many more.

- *Basic functions*: `sign`, `max`, `sum`, `average`. All cheap!
- *All appropriate combinations*: i.e. taking care of the different dimensions being applied to.

All possibilities in 3 variables gave 1728 features. But:

- Many easily seen as mathematically identical.
- Some others are certainly identical for the dataset in question.
- A handful were constant (evaluate to the same number) for the whole dataset (making them useless for ML).

After this: 78 features for 3-variable problems, compared to 11 previously: seven times more! 105 features for 4-variable problems.

SC^2 2019 Paper Results I

		DT	KNN	MLP	SVM
11 features	Accuracy (%)	62.6	63.3	61.6	58.8
	Time (s)	9,994	10,105	9,822	10,725
78 features	Accuracy (%)	65.2	66.3	**67**	65
	Time (s)	9,603	**9,178**	9,399	9,487

	Virtual Best	Virtual Worst	random	sotd	Brown
Accuracy (%)	*100*	*0*	22.7	49.5	51
Time (s)	*8,623*	*64,534*	30,235	11,938	10,951

Accuracy: the percentage of problems in the testing set for which a heuristic picked the optimal ordering.
Time: the computation time if all the testing set had CADs computed with that heuristic's suggested orderings.

SC^2 2019 Paper Results II

		DT	KNN	MLP	SVM
11 features	Accuracy (%)	62.6	63.3	61.6	58.8
	Time (s)	9,994	10,105	9,822	10,725
78 features	Accuracy (%)	65.2	66.3	**67**	65
	Time (s)	9,603	**9,178**	9,399	9,487

	Virtual Best	Virtual Worst	random	sotd	Brown
Accuracy (%)	*100*	*0*	22.7	49.5	51
Time (s)	*8,623*	*64,534*	30,235	11,938	10,951

The human made heuristics achieved times that are 27% above the minimum possible. ML with similar features reduced that to 14% above. The additional features reduced it to only 6% above. All ML classifiers improved performance with extra features.

SC^2 2019 Paper Results III

		DT	KNN	MLP	SVM
11 features	Accuracy (%)	62.6	63.3	61.6	58.8
	Time (s)	9,994	10,105	9,822	10,725
78 features	Accuracy (%)	65.2	66.3	**67**	65
	Time (s)	9,603	**9,178**	9,399	9,487

	Virtual Best	Virtual Worst	random	sotd	Brown
Accuracy (%)	*100*	*0*	22.7	49.5	51
Time (s)	*8,623*	*64,534*	30,235	11,938	10,951

Quickest computation times achieved by KNN, although MLP had slightly higher accuracy. I.e. MLP makes best choice more often but the occasions it makes a poor choice drag its times down considerably.

With Florescu at MACIS 2019

Prior work defined accuracy as standard for ML classification:

Accuracy: *the percentage of problems in the testing set for which a heuristic picked the optimal ordering.*

No difference between "almost optimal" and "very bad" ordering.

Redefined testing accuracy to percentage within 20% of instance optimum. Redesigned hyperparameter selection function to maximise CAD times instead of classification F1-score.

Experiments on a 4 variable dataset.

- All ML models improved by new hyper parameter selection.
- All ML classifiers outperform human-made heuristics.
- All heuristics further away from optimum than on 3–variable dataset (choosing from 24 instead of 6 orderings).
- Best performing ML model achieves timings 67% greater than the minimum; best human-made heuristic is 98% greater.

With Florescu at MACIS 2019

Prior work defined accuracy as standard for ML classification:
Accuracy: *the percentage of problems in the testing set for which a heuristic picked the optimal ordering.*
No difference between "almost optimal" and "very bad" ordering.

Redefined testing accuracy to percentage within 20% of instance optimum. Redesigned hyperparameter selection function to maximise CAD times instead of classification F1-score.

Experiments on a 4 variable dataset.

- All ML models improved by new hyper parameter selection.
- All ML classifiers outperform human-made heuristics.
- All heuristics further away from optimum than on 3–variable dataset (choosing from 24 instead of 6 orderings).
- Best performing ML model achieves timings 67% greater than the minimum; best human-made heuristic is 98% greater.

MACIS 2019 Paper Results (4 variables)

$-O$ uses original cross validation and $-N$ the new one.

	DT-O	DT-N	KNN-O	KNN-N
Acc.	51.7%	54.3%	53.9%	54.5%
Time	4,022	**3,627**	3,808	3,748

	MLP-O	MLP-N	SVM-O	SVM-N
Acc.	53.6%	**56.9%**	53.9%	54.9%
Time	3,972	3,784	3,795	3,672

	VirtualBest	VirtualWorst	random	`Brown`	`sotd`
Acc.	100%	0%	17.0%	20.1%	47.8%
Time	2,177	22,735	8,291	8,292	4,348

Outline

General Thesis

There is scope to apply Machine Learning to CAD because there is a choice to be made (variable ordering) that does not affect the correctness of the output but does effect efficiency. CAD is not special here: many other algorithms in Computer Algebra Systems (CASs) have similar choices to be made.

At the moment most such choices are taken by either the user, by a magic constant (educated choices picked by the developers) or perhaps a human written heuristic.

Our general thesis is that many could be better taken by a Machine Learning classifier.

Other Heuristic Choices from Topics in This Talk

Whether to precondition CAD with Groebner Bases:

Z. Huang, M. England, J.H. Davenport and L.C. Paulson.

Using Machine Learning to decide when to Precondition Cylindrical Algebraic Decomposition with Groebner Bases.

Proc. SYNASC 2016, pp. 45–52. IEEE, 2016.

https://doi.org/10.1109/SYNASC.2016.020

The order to study polynomials in NuCAD:

C.W. Brown and G.C. Daves.

Applying Machine Learning to Heuristics for Real Polynomial Constraint Solving.

Proc. ICMS 2020, LNCS 12097, pages 292-301. Springer, 2021.

https://doi.org/10.1007/978-3-030-52200-1_29

Which cells to use for covering in CDCAC.

Future Work

Have demonstrated that ML may be trained to do well on a standard community dataset for a fixed number of variables. But key questions remain:

- How to react to different numbers of variables?
- How to adapt to new example classes?
- How to make use of deep learning?

The latter point requires far greater quantities of data than exist. Of course, one can cheaply produce random polynomials to train on but how to ensure they are useful and representative?

- Random perturbations of existing data?
- Randomly generate to fit the distribution of existing data?

PhD student Tereso del Rio (Coventry U.) working on this now.

Contact Details and Advert

Contact Details

Matthew.England@coventry.ac.uk

https://matthewengland.coventry.domains/

Advert

Fully Funded PhD Position available at Coventry to work on Machine Learning to Improve Symbolic Integration and Symbolic Simplification. Sponsored by Maplesoft.

https://tinyurl.com/3exmk9vk

Deadline to Apply: 13th September 2021

Interviews and Decision: End September

PhD Start: Jan 2022