New Approaches to Real Quantifier Elimination and Cylindrical Algebraic Decomposition

Matthew England
Coventry University

Mathematical Foundations of Computation Seminar
University of Bath
11th November 2024

Speaker supported by EPSRC grant EP/T015748/1.

Outline

Overview

Summary: the talk will describe the Real Quantifier Elimination Problem and the Cylindrical Algebraic Decomposition method to tackle it. We cover recent developments that improve CAD with ideas from (a) computational logic / satisfiability checking; and (b) (if there is time) machine learning / explainable AI.

Key message: connections between different areas of mathematics / computer science allow for unexpected progress!

Includes joint work with: Chris Brown, Erika Abraham, Kelly Cohen, James Davenport, Tereso del Río, Dorian Florescu, Gereon Kremer, Lynn Pickering, Jasper Nalbach, and Philippe Specht.

Outline

Outline

Real QE

Real Quantifier Elimination (Real QE)

Given: A quantified formulae (in prenex normal form) whose atoms are (integral) polynomial constraints;
Produce: a quantifier free formula logically equivalent over $\mathbb{R}$.

Fully quantified examples:

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

Output: `True` e.g. when $x = 0$

Input: $\forall x, x^2 + 3x + 1 > 0$

Output: `False` e.g. when $x = -1$

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

Output: `True`

Partially quantified example:

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

Output: ???

The answer depends on the free (unquantified) variable, b.

Real QE

Real Quantifier Elimination (Real QE)

Given: A quantified formulae (in prenex normal form) whose atoms are (integral) polynomial constraints;
Produce: a quantifier free formula logically equivalent over $\mathbb{R}$.

Fully quantified examples:

Input: $\exists x, x^2 + 3x + 1 > 0$
Output: `True` e.g. when $x = 0$

Input: $\forall x, x^2 + 3x + 1 > 0$
Output: `False` e.g. when $x = -1$

Input: $\forall x, x^2 + 1 > 0$
Output: `True`

Partially quantified example:

Input: $\forall x, x^2 + bx + 1 > 0$
Output: ???

The answer depends on the free (unquantified) variable, b.

Partially quantified Tarski formulae example

Consider $\forall x, x^2 + bx + 1 > 0$.
When $b = 0$ and $b = 3$:

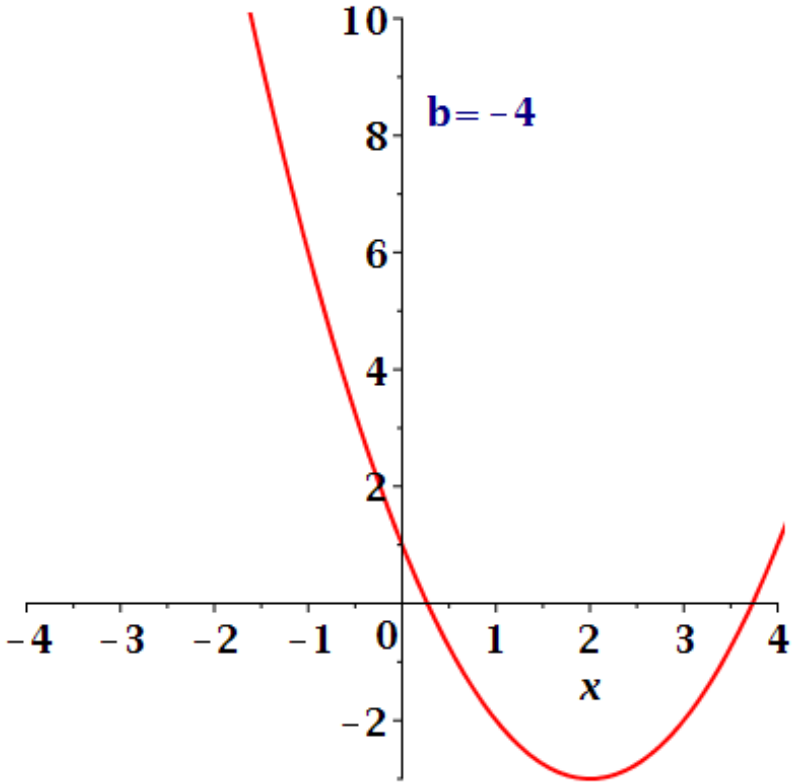
$$\forall x, x^2 + 1 > 0 \text{ is } \texttt{True},$$
$$\forall x, x^2 + 3x + 1 > 0 \text{ is } \texttt{False}.$$

So in general the answer depends on b.

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

Output: $(-2 < b) \wedge (b < 2)$

The technology we look at today can answer such questions algebraically.



Partially quantified Tarski formulae example

Consider $\forall x, x^2 + bx + 1 > 0$.
When $b = 0$ and $b = 3$:

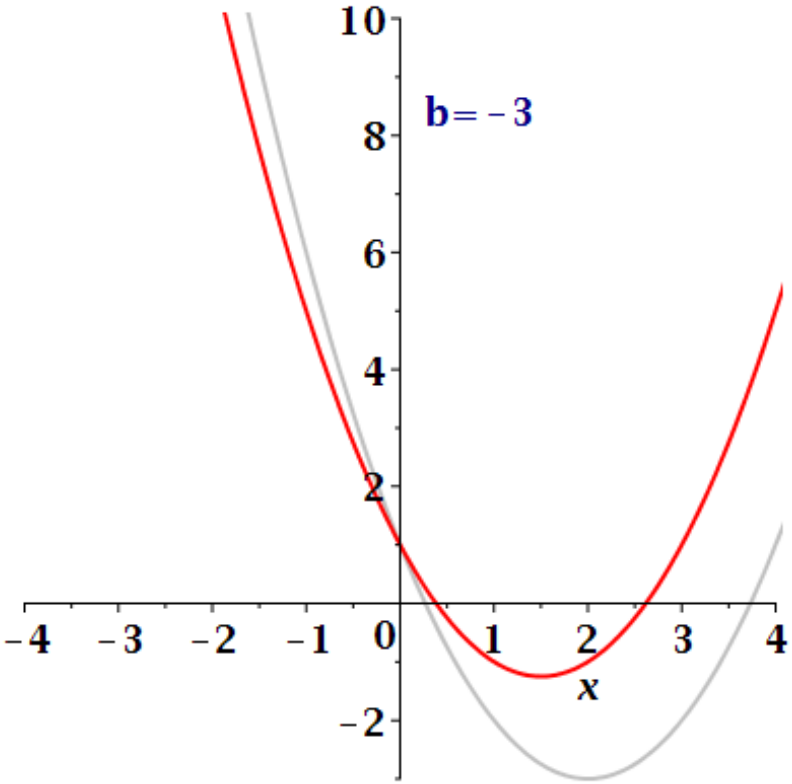
$$\forall x, x^2 + 1 > 0 \text{ is } \texttt{True} \ ,$$
$$\forall x, x^2 + 3x + 1 > 0 \text{ is } \texttt{False} \ .$$

So in general the answer depends on b.

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

Output: $(-2 < b) \wedge (b < 2)$

The technology we look at today can answer such questions algebraically.



Partially quantified Tarski formulae example

Consider $\forall x, x^2 + bx + 1 > 0$.
When $b = 0$ and $b = 3$:

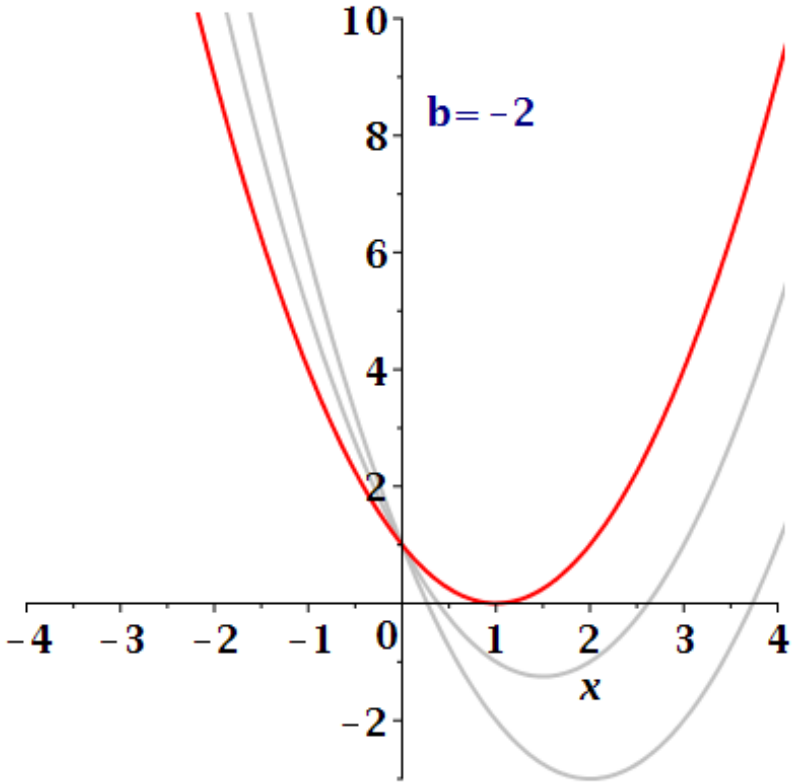
$$\forall x, x^2 + 1 > 0 \text{ is } \texttt{True},$$
$$\forall x, x^2 + 3x + 1 > 0 \text{ is } \texttt{False}.$$

So in general the answer depends on b.

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

Output: $(-2 < b) \wedge (b < 2)$

The technology we look at today can answer such questions algebraically.



Partially quantified Tarski formulae example

Consider $\forall x, x^2 + bx + 1 > 0$.
When $b = 0$ and $b = 3$:

$$\forall x, x^2 + 1 > 0 \text{ is } \texttt{True},$$

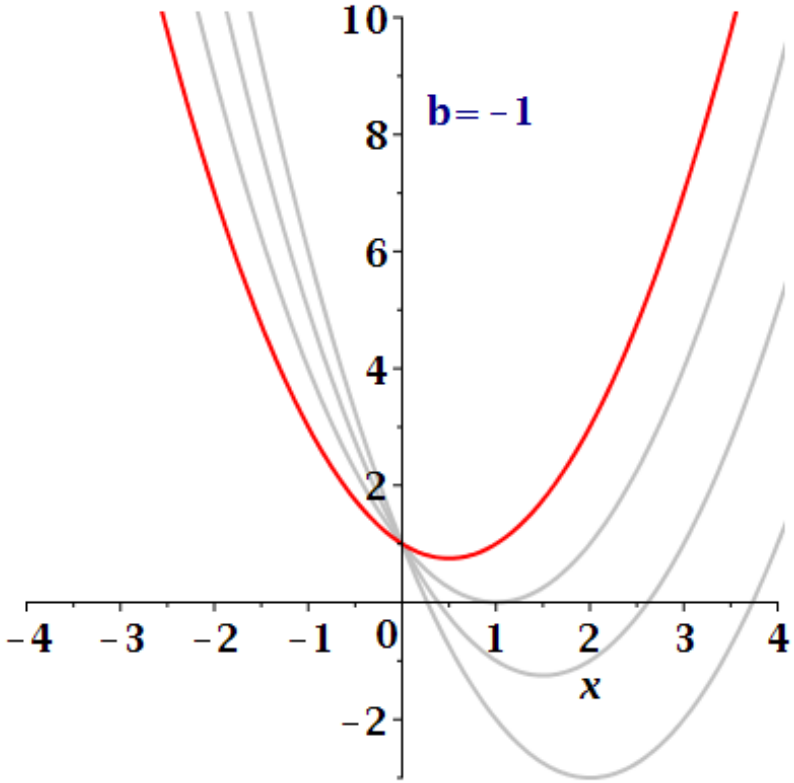
$$\forall x, x^2 + 3x + 1 > 0 \text{ is } \texttt{False}.$$

So in general the answer depends on b.

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

Output: $(-2 < b) \wedge (b < 2)$

The technology we look at today can answer such questions algebraically.



Partially quantified Tarski formulae example

Consider $\forall x, x^2 + bx + 1 > 0$.
When $b = 0$ and $b = 3$:

$$\forall x, x^2 + 1 > 0 \text{ is } \texttt{True},$$

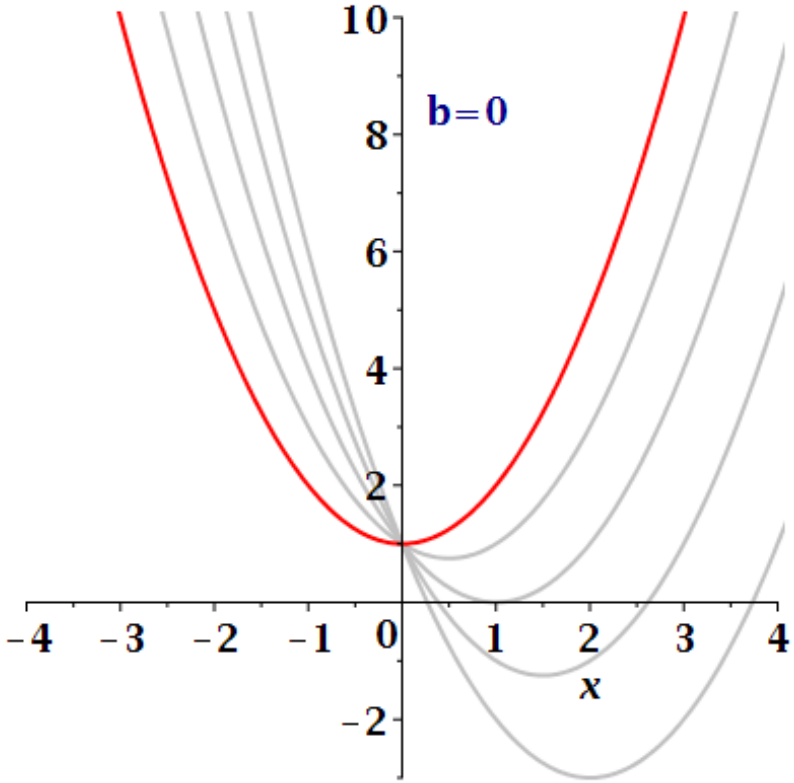
$$\forall x, x^2 + 3x + 1 > 0 \text{ is } \texttt{False}.$$

So in general the answer depends on b.

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

Output: $(-2 < b) \wedge (b < 2)$

The technology we look at today can answer such questions algebraically.



Partially quantified Tarski formulae example

Consider $\forall x, x^2 + bx + 1 > 0$.
When $b = 0$ and $b = 3$:

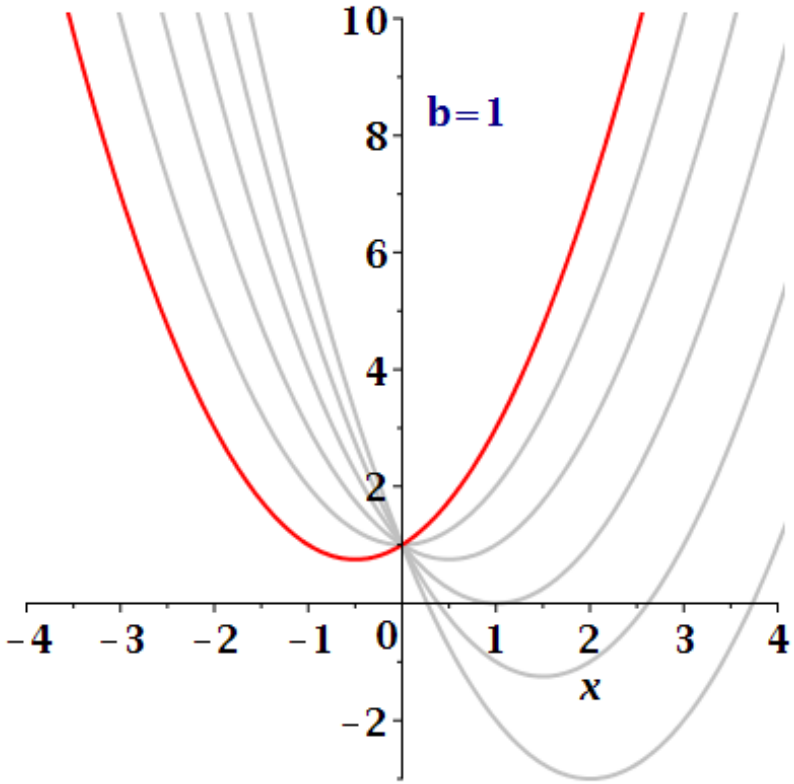
$$\forall x, x^2 + 1 > 0 \text{ is } \texttt{True},$$
$$\forall x, x^2 + 3x + 1 > 0 \text{ is } \texttt{False}.$$

So in general the answer depends on b.

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

Output: $(-2 < b) \wedge (b < 2)$

The technology we look at today can answer such questions algebraically.



Partially quantified Tarski formulae example (Slide 3/50)

Consider $\forall x, x^2 + bx + 1 > 0$.
When $b = 0$ and $b = 3$:

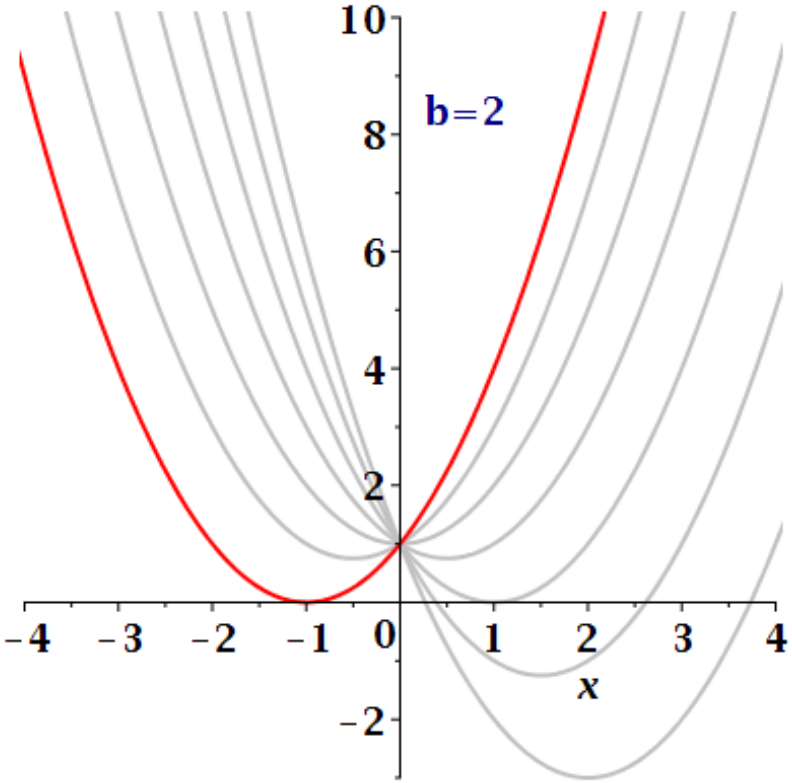
$$\forall x, x^2 + 1 > 0 \text{ is } \texttt{True},$$
$$\forall x, x^2 + 3x + 1 > 0 \text{ is } \texttt{False}.$$

So in general the answer depends on b.

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

Output: $(-2 < b) \wedge (b < 2)$

The technology we look at today can answer such questions algebraically.



Partially quantified Tarski formulae example (Slide 3/50)

Consider $\forall x, x^2 + bx + 1 > 0$.
When $b = 0$ and $b = 3$:

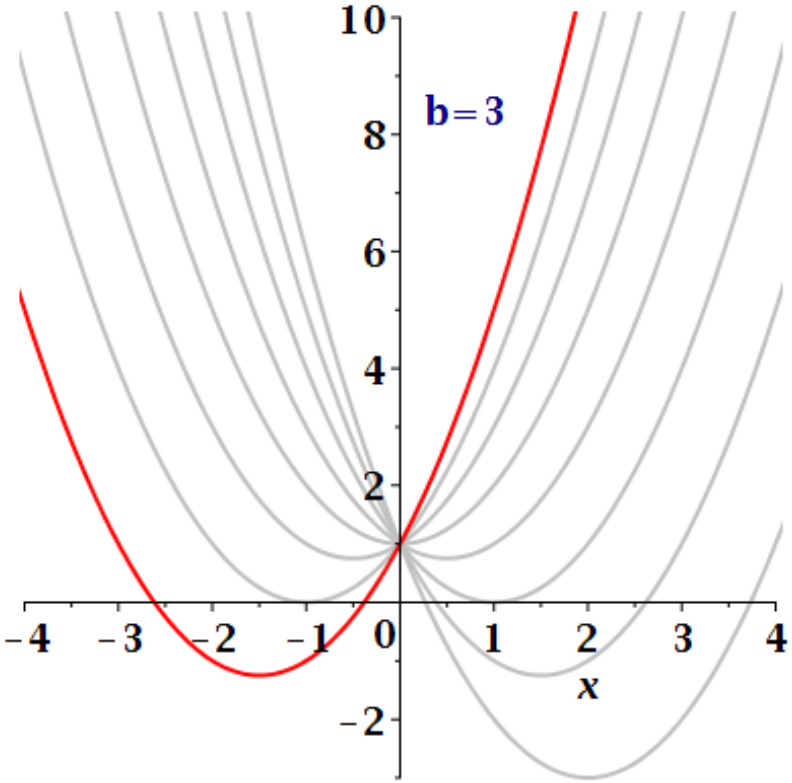
$$\forall x, x^2 + 1 > 0 \text{ is } \texttt{True},$$
$$\forall x, x^2 + 3x + 1 > 0 \text{ is } \texttt{False}.$$

So in general the answer depends on b.

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

Output: $(-2 < b) \wedge (b < 2)$

The technology we look at today can answer such questions algebraically.



Partially quantified Tarski formulae example (Slide 3/50)

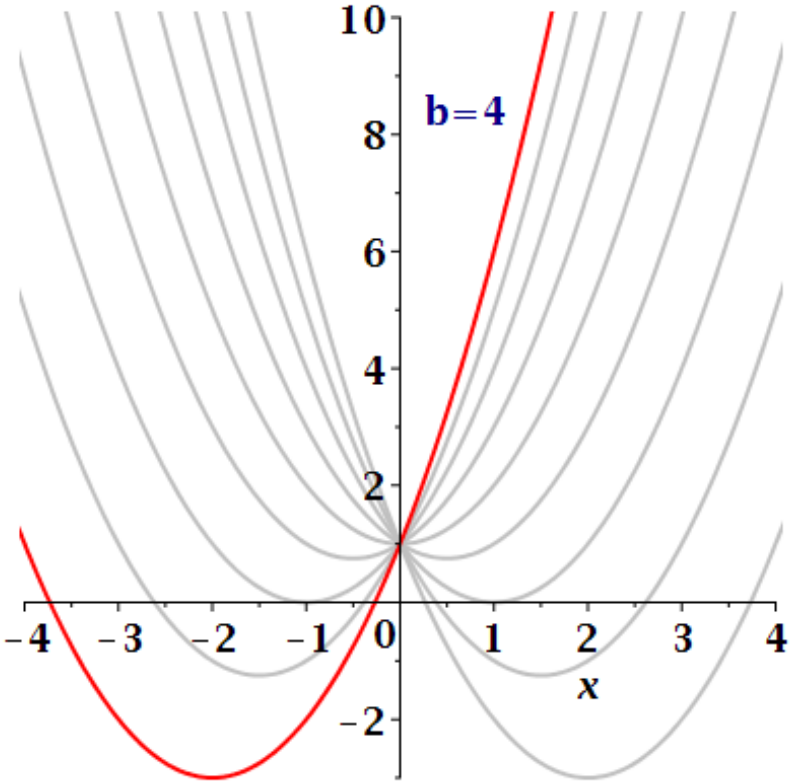
Consider $\forall x, x^2 + bx + 1 > 0$.
When $b = 0$ and $b = 3$:

$$\forall x, x^2 + 1 > 0 \text{ is } \texttt{True},$$
$$\forall x, x^2 + 3x + 1 > 0 \text{ is } \texttt{False}.$$

So in general the answer depends on b.

$$\text{Input: } \forall x, x^2 + bx + 1 > 0$$
$$\text{Output: } (-2 < b) \wedge (b < 2)$$

The technology we look at today can answer such questions algebraically.



Outline

Cylindrical Algebraic Decomposition (Slide 4/50)

A **Cylindrical Algebraic Decomposition (CAD)** 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 $i \neq j$) such that:
- cells are **semi-algebraic** meaning they may be described by a finite sequence of polynomial constraints;
- cells are arranged **cylindrically**, i.e. projections of any two on a lower coordinate space *in the variable ordering* are identical or disjoint (cells in $\mathbb{R}^m$ stack in cylinders over cells in $\mathbb{R}^{m-1}$).

G.E. Collins.
Quantifier elimination for real closed fields by cylindrical algebraic decomposition.
In Proc. 2nd GI Conf. Automata Theory and Formal Languages, pp. 134–183. Springer-Verlag, 1975.

CAD Invariance Property

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 a finite set of sample points. In particular, a CAD for the polynomials in a Real QE problem can be used to find the quantifier free equivalent formula.

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

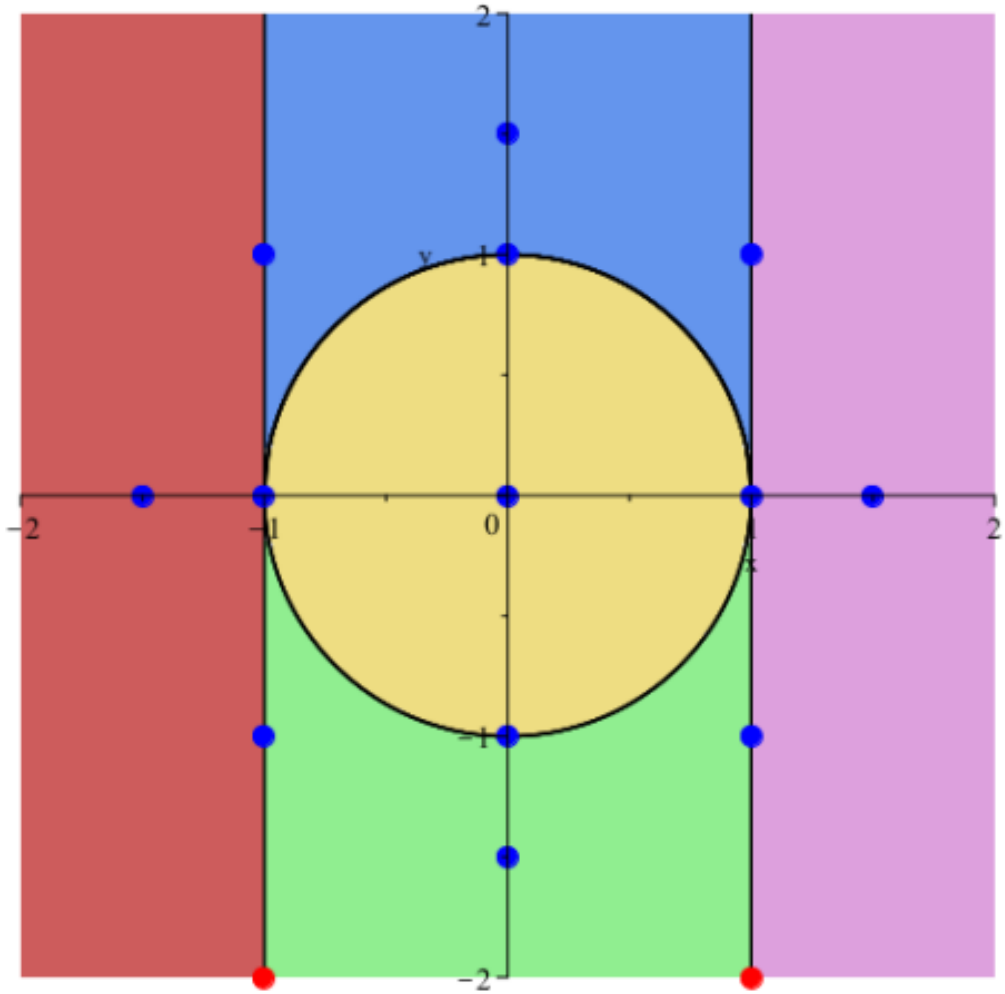
Cell 9: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y > 0$

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

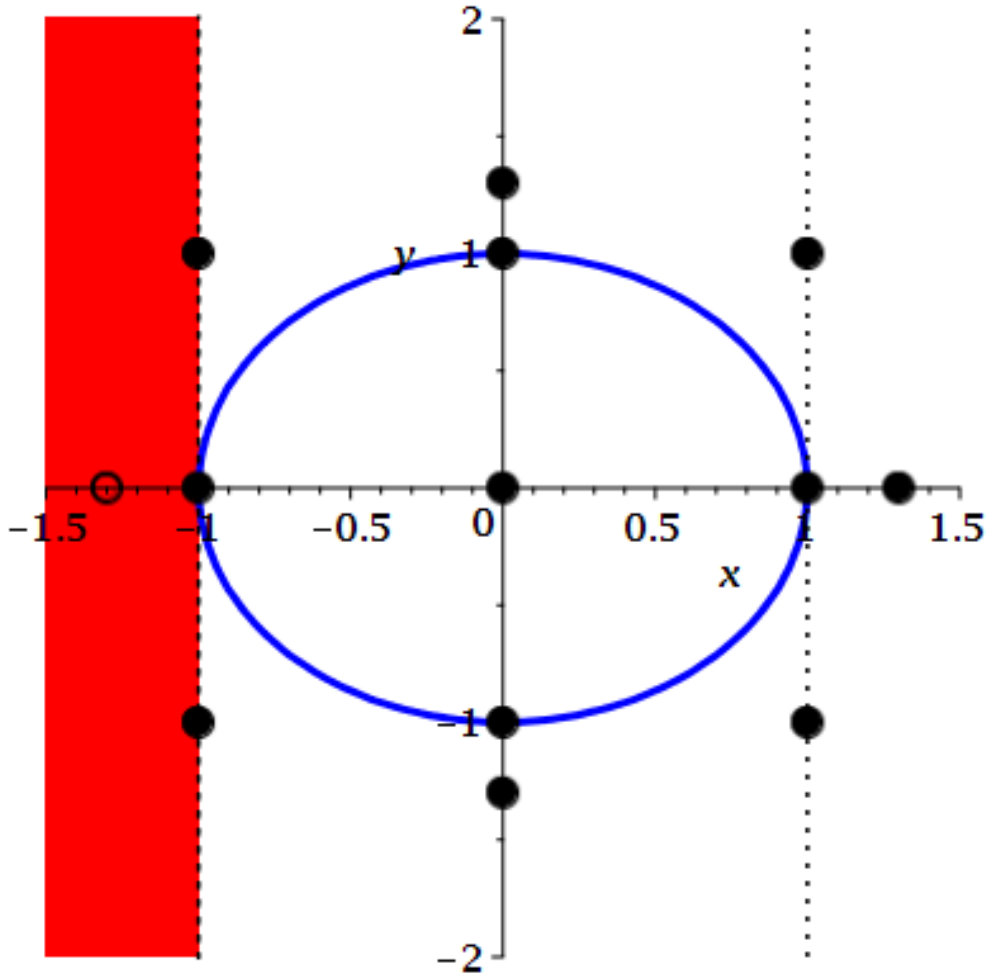
Cell 9: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y > 0$

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

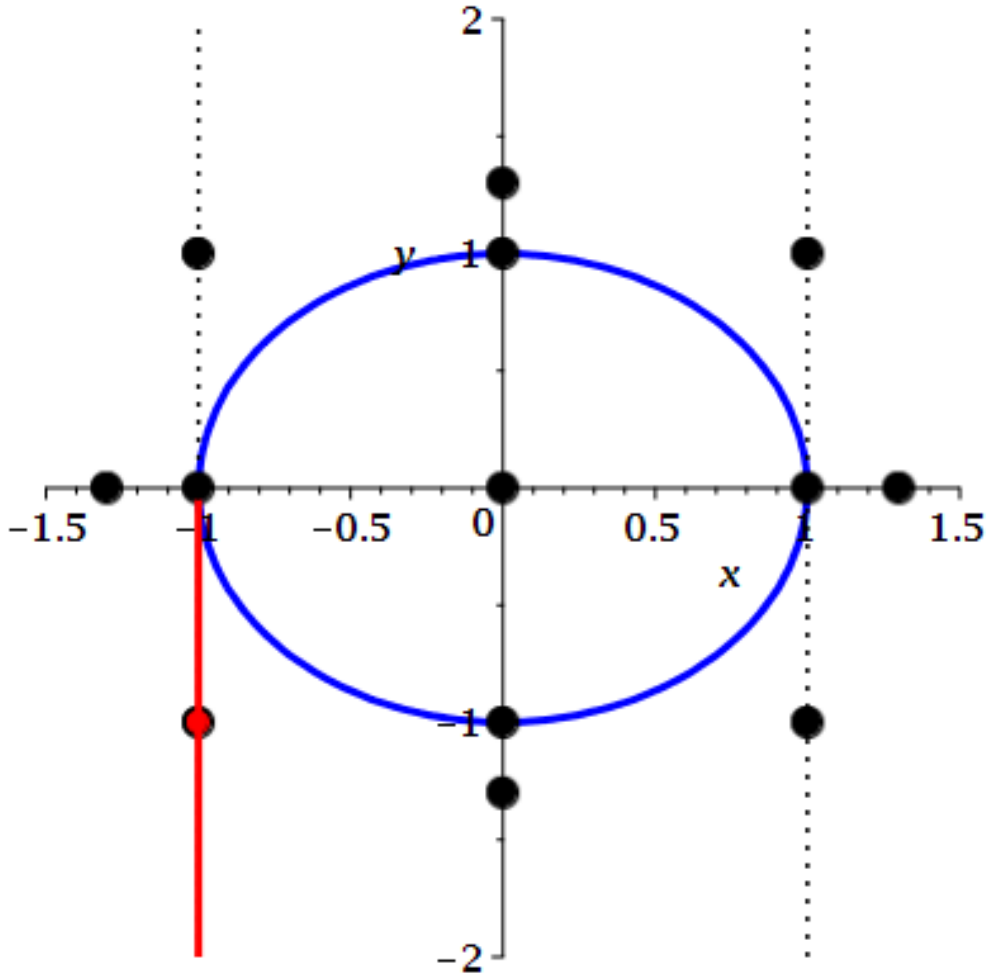
Cell 9: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y > 0$

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

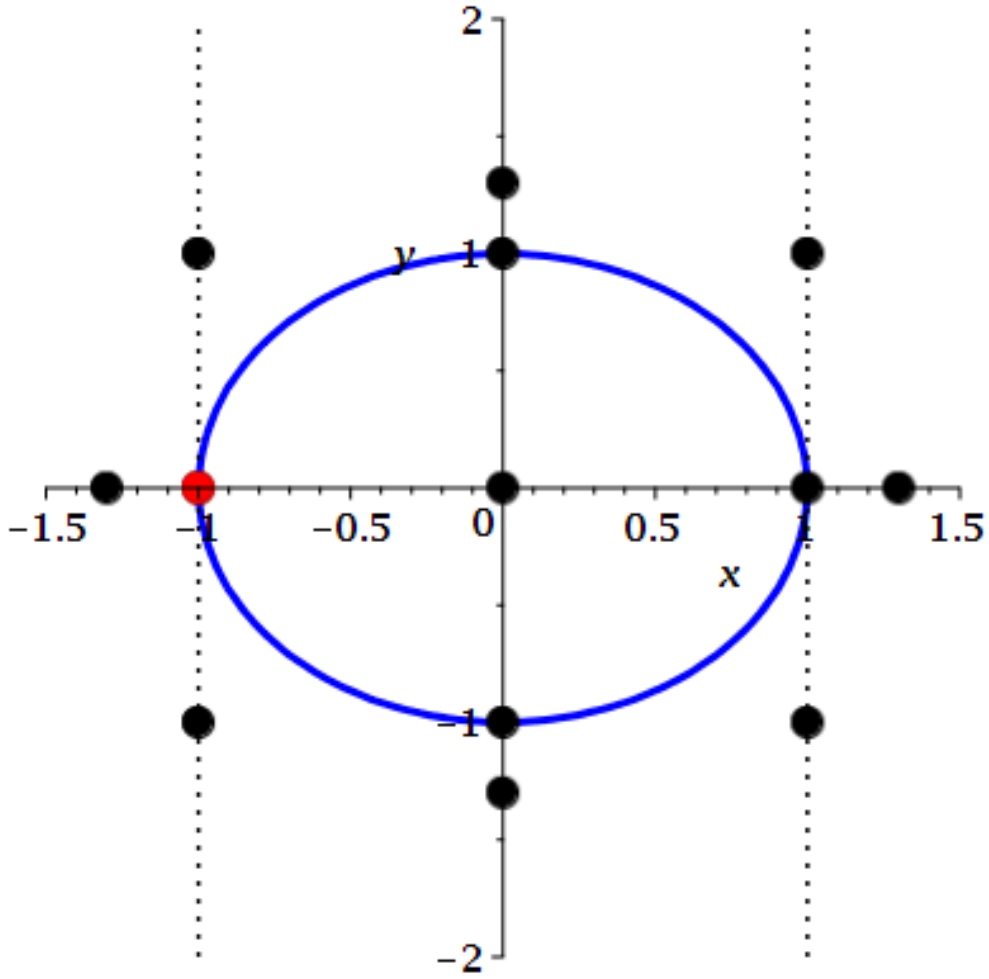
Cell 9: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y > 0$

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

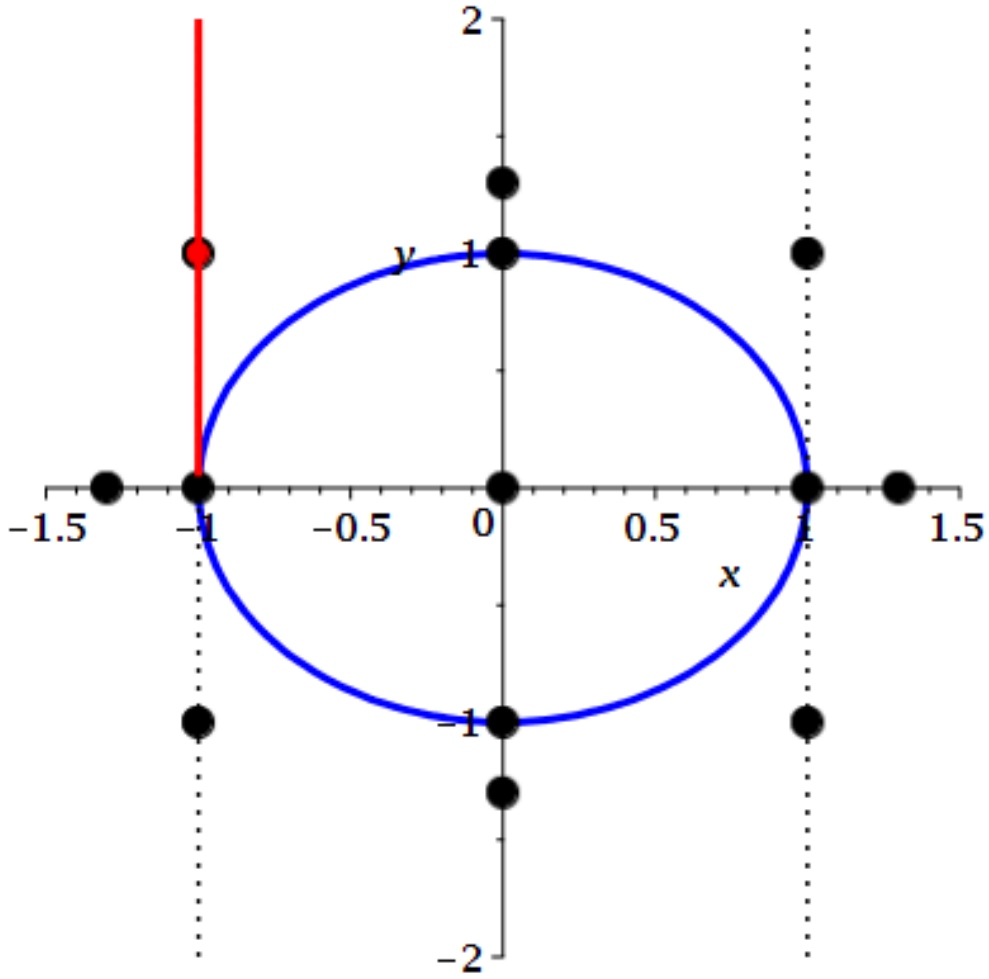
Cell 9: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y > 0$

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

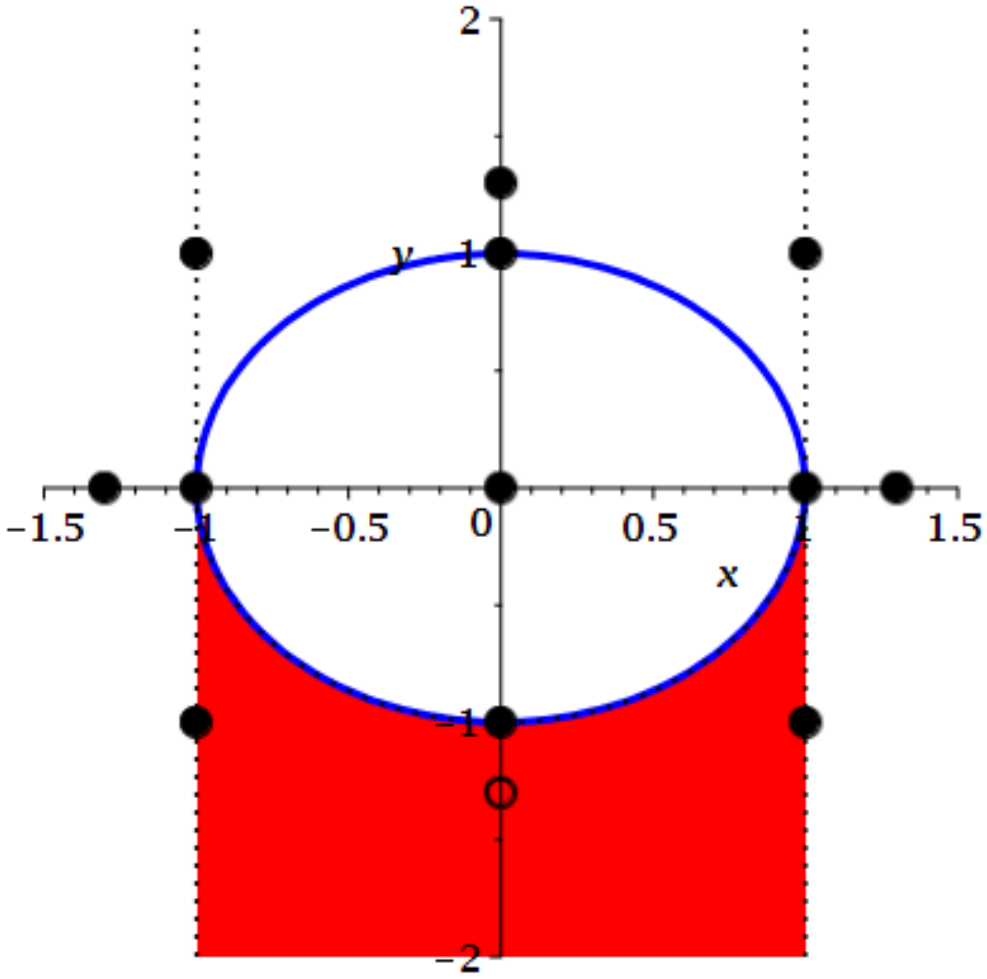
Cell 9: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y > 0$

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

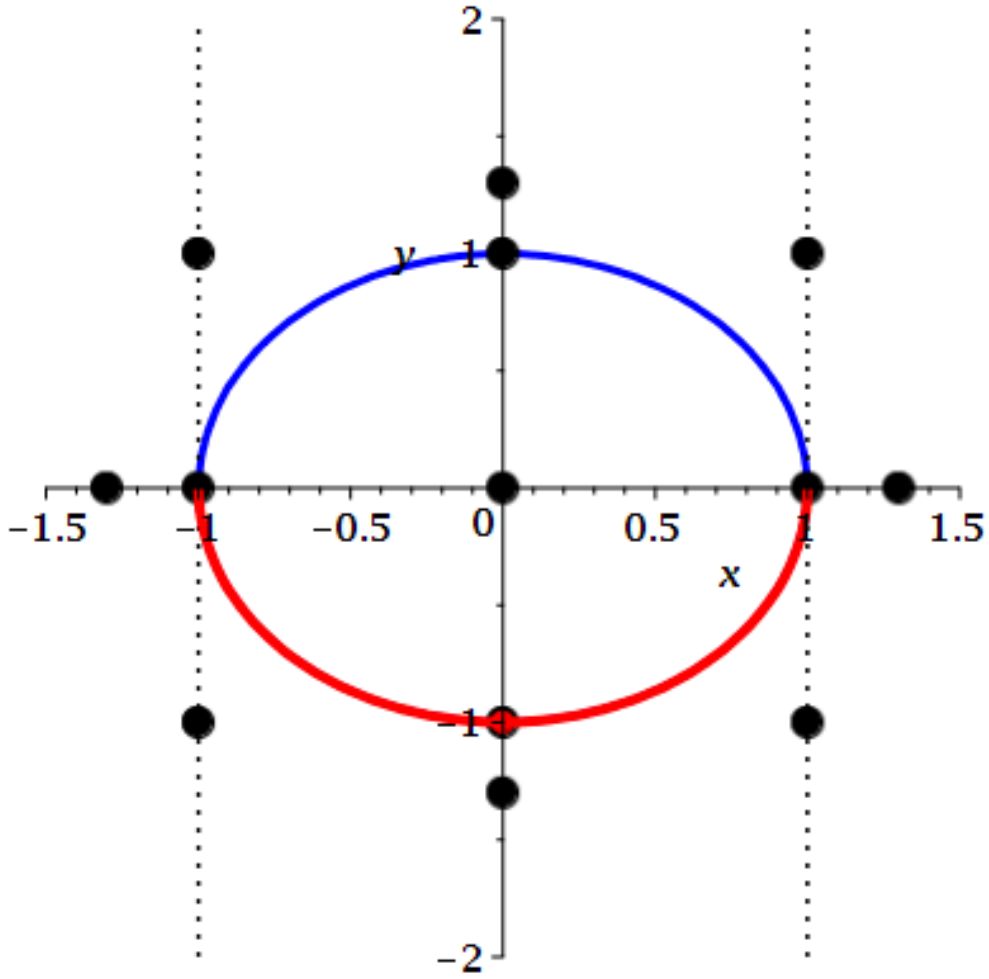
Cell 9: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y > 0$

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

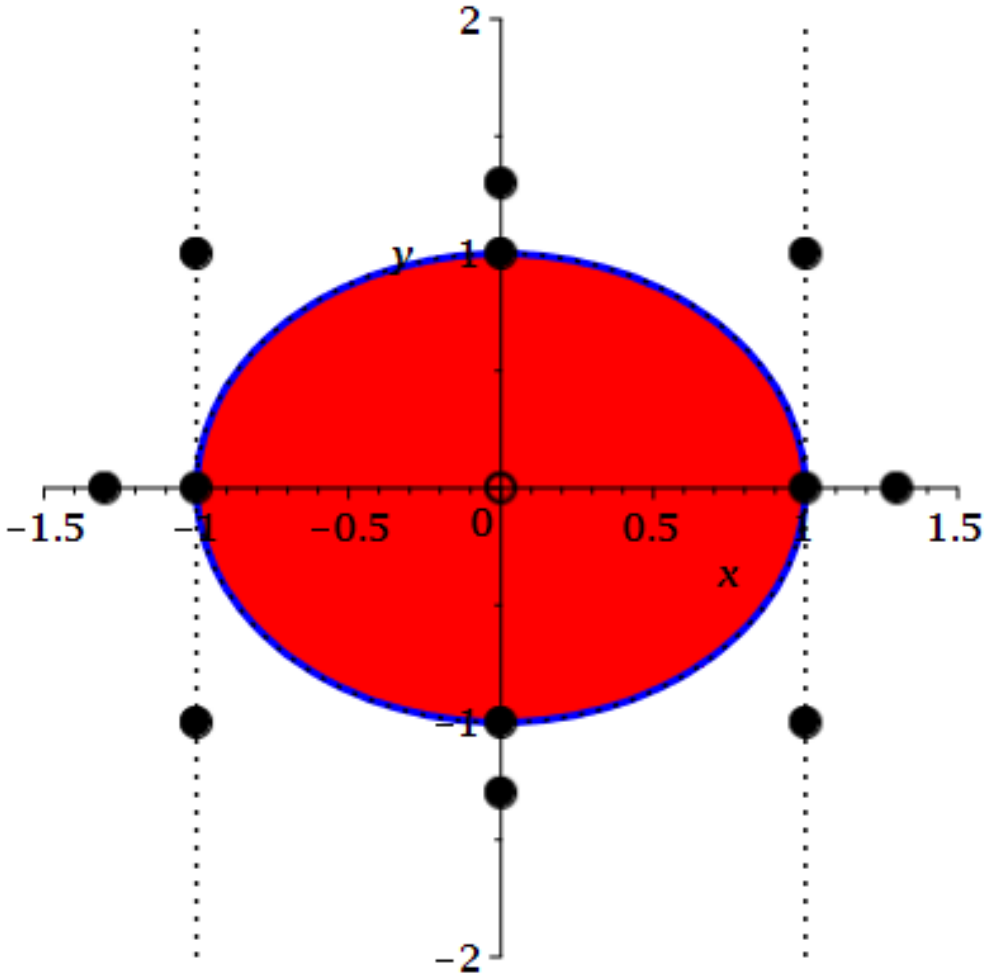
Cell 9: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y > 0$

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

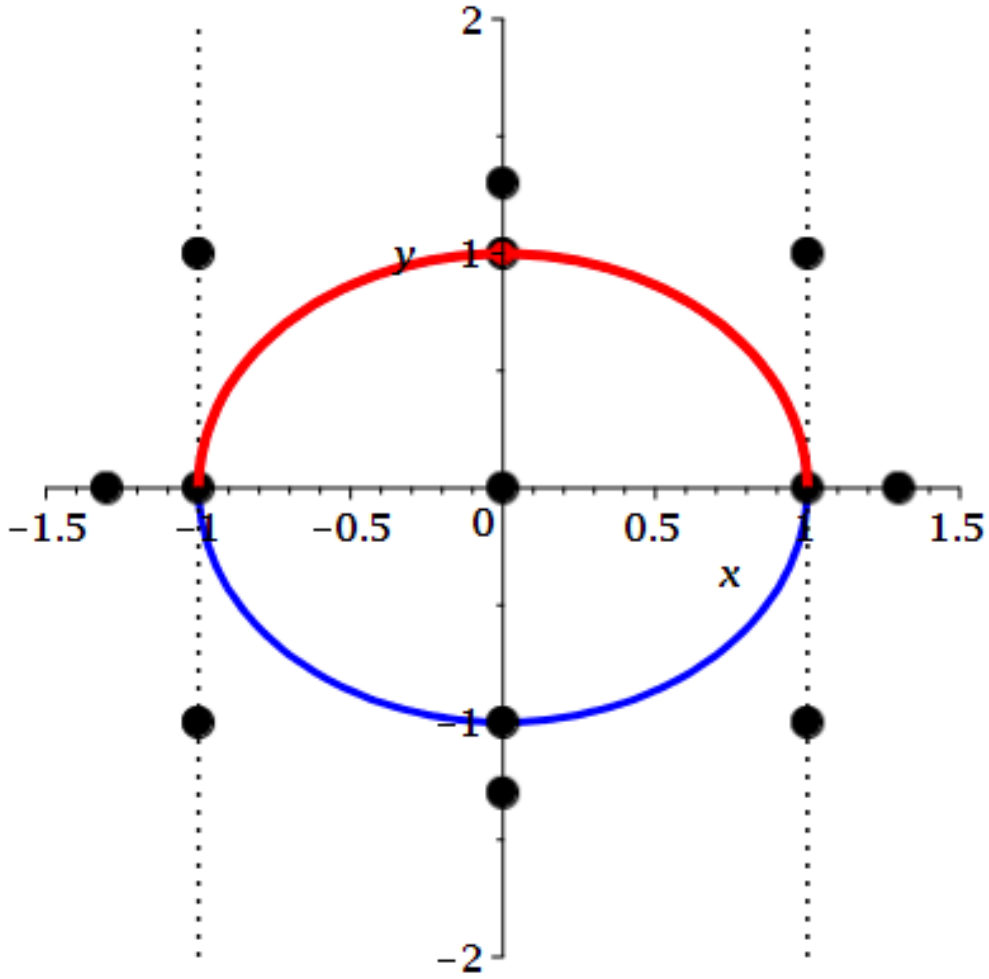
Cell 9: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y > 0$

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

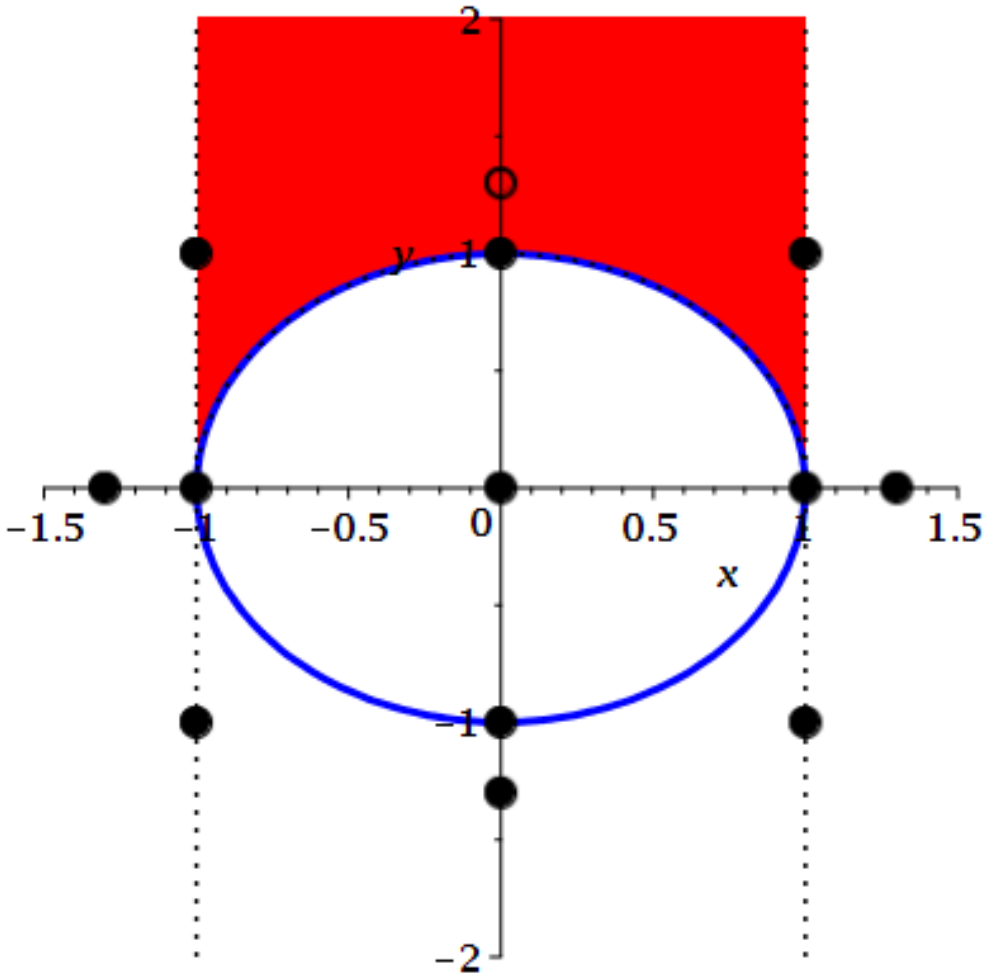
Cell 9: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y > 0$

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

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

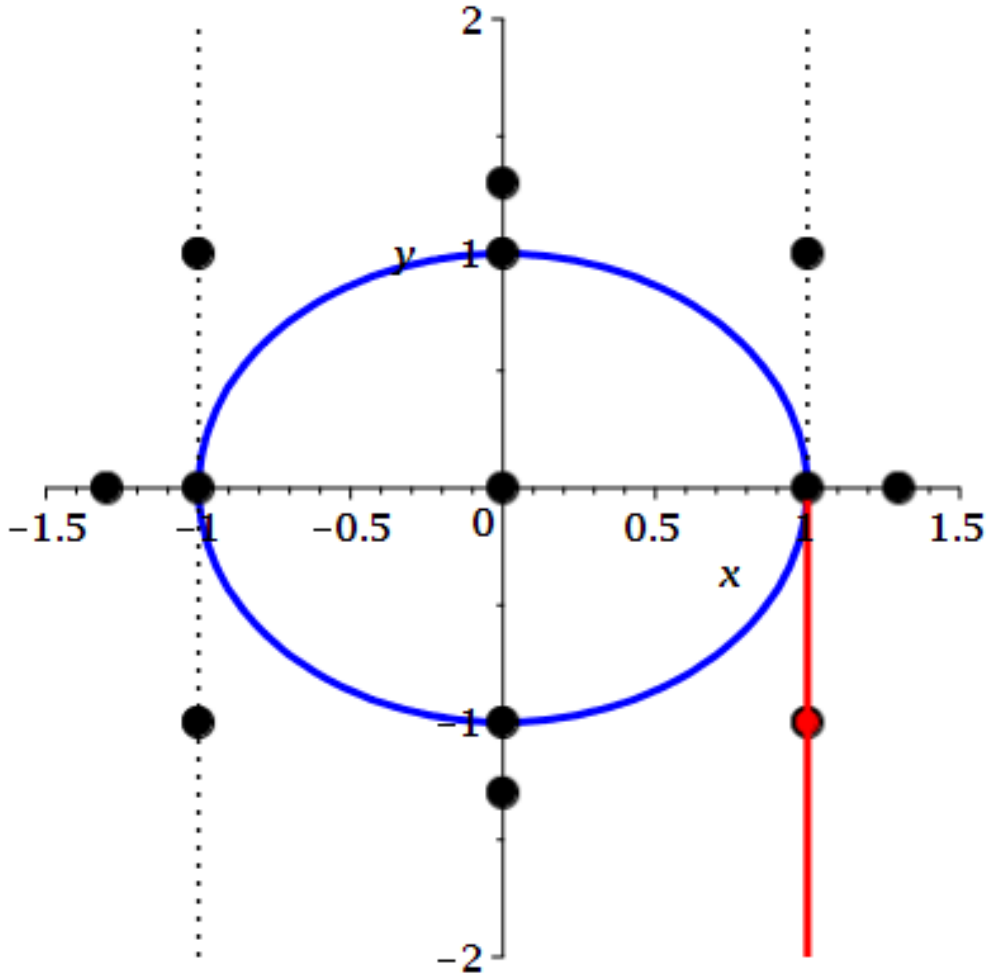
Cell 12: $x = 1, y > 0$

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

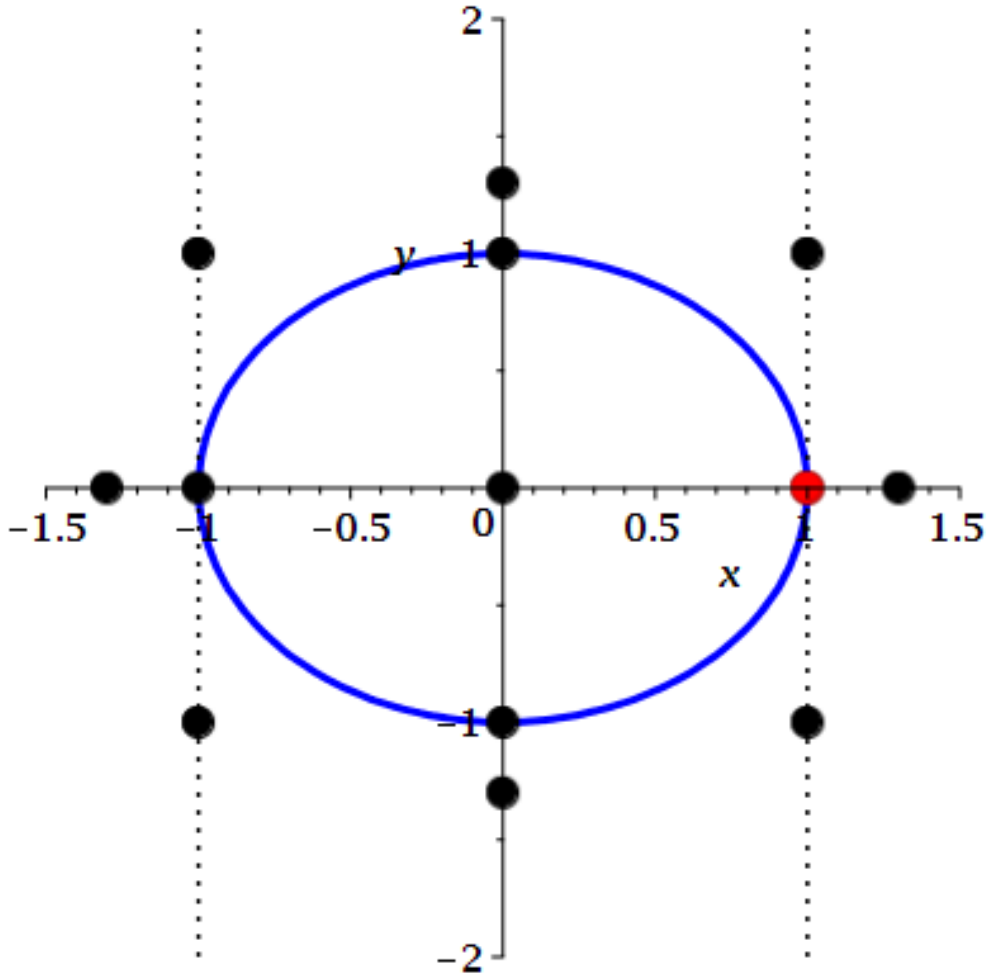
Cell 9: $-1 < x < 1$,
$y^2 + x^2 - 1 > 0, y > 0$

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

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

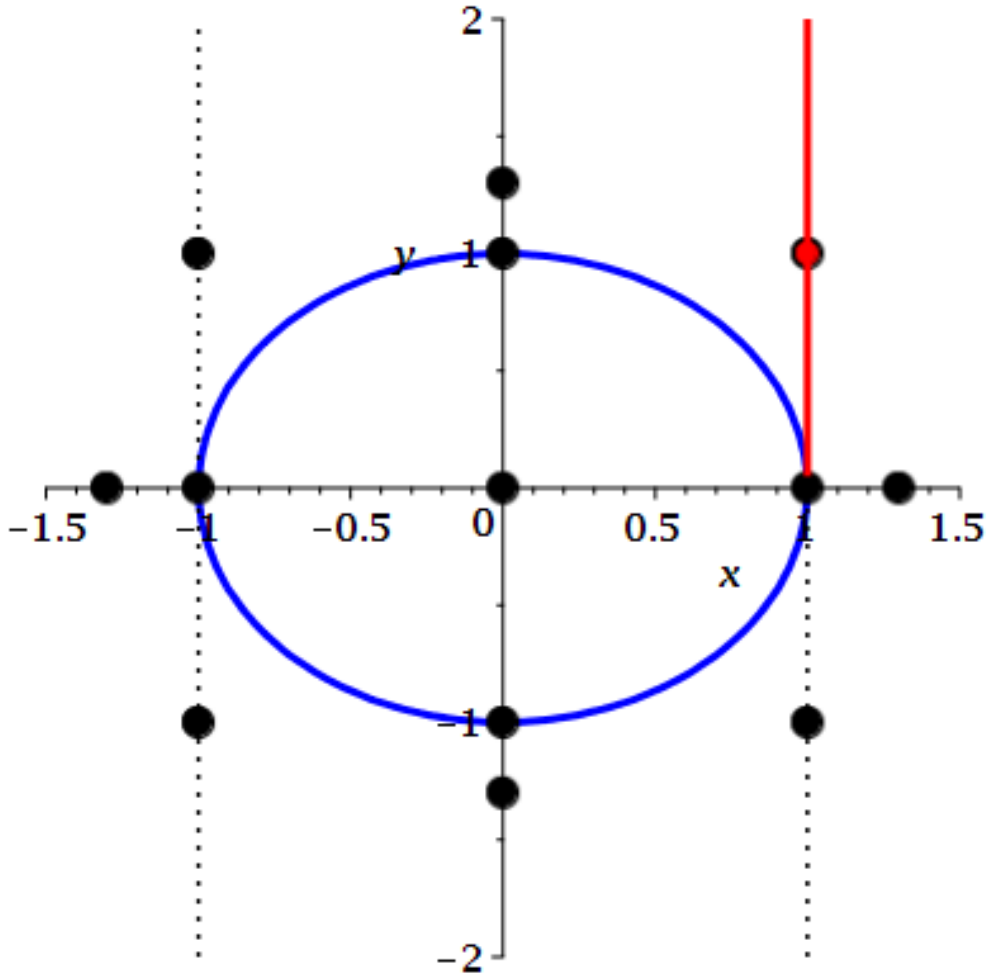
Cell 12: $x = 1, y > 0$

Cell 13: $x > 1$, y free



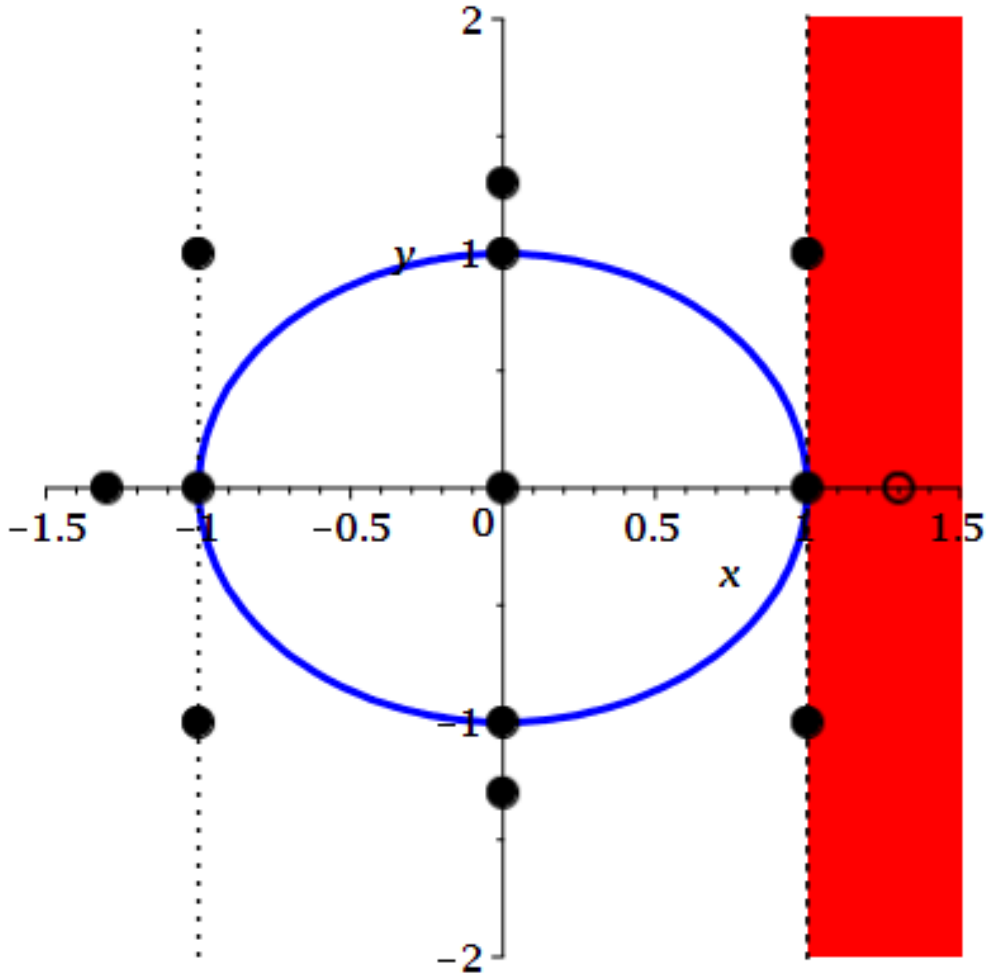
Example: Circle — visualisation

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



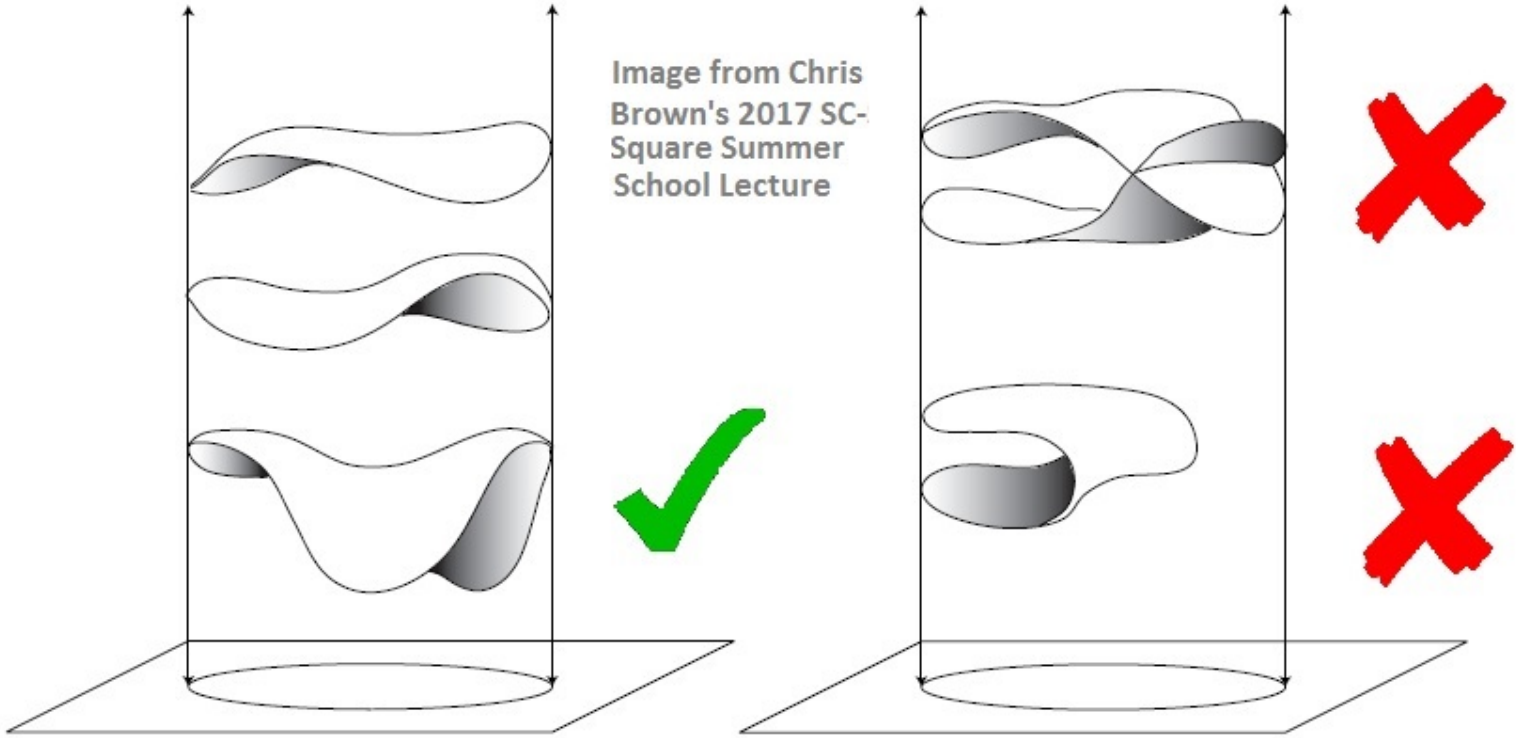
Example: Circle — visualisation

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



How to build a CAD?

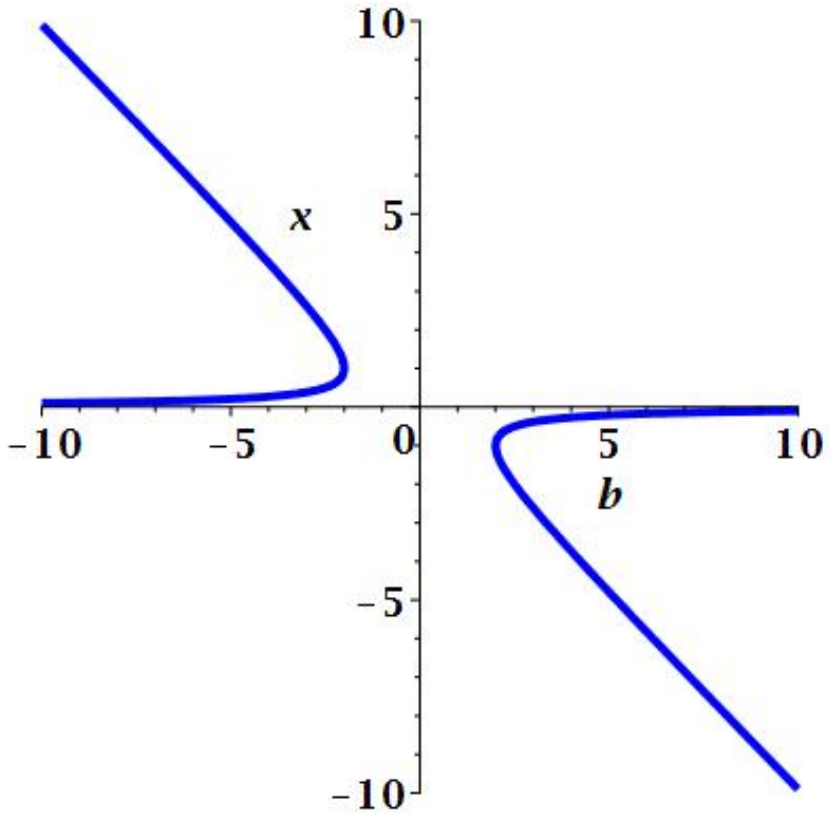
The usual approach is to calculate projection polynomials whose roots indicate changes in the behaviour of the input set; decompose in $\mathbb{R}^{n-1}$ with respect to these and then lift each cell to a stack of cells in in $\mathbb{R}^n$ by working at a sample point.
This is a sound approach if the projection provides **delineability**.



QE via CAD Example

How to determine with CAD?

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



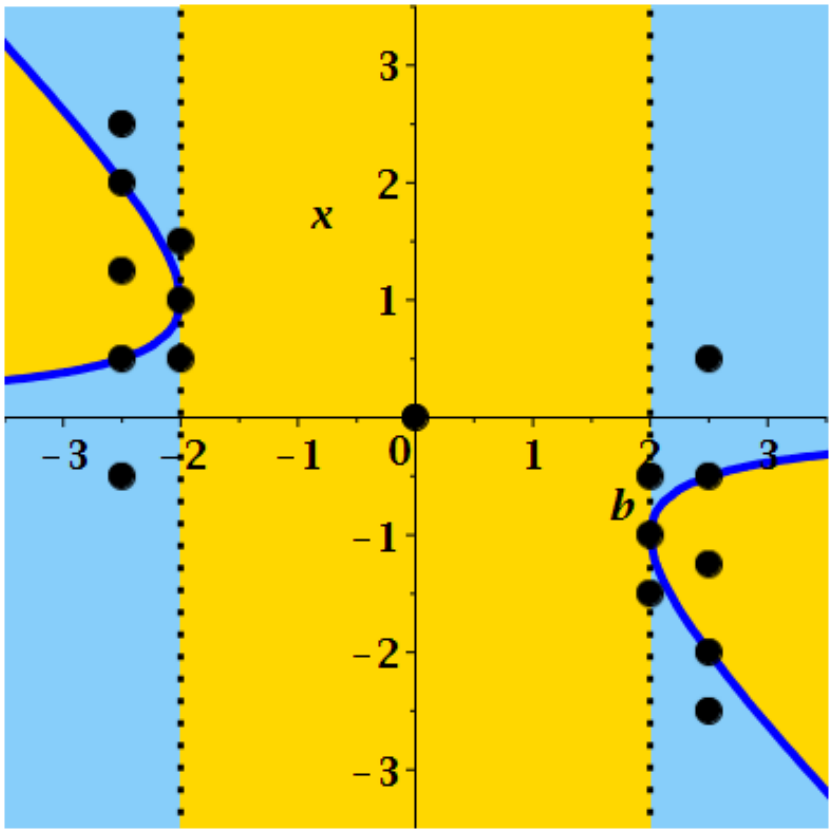
QE via CAD Example

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

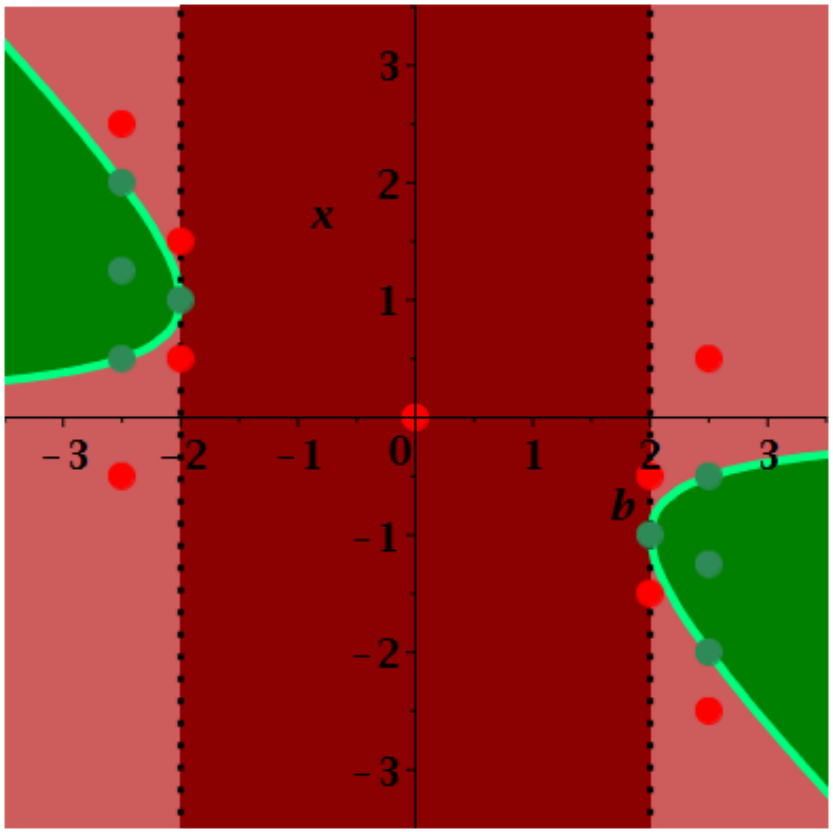
How to determine with CAD?

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

To solve we:

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

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



QE via CAD Example

How to determine with CAD?

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

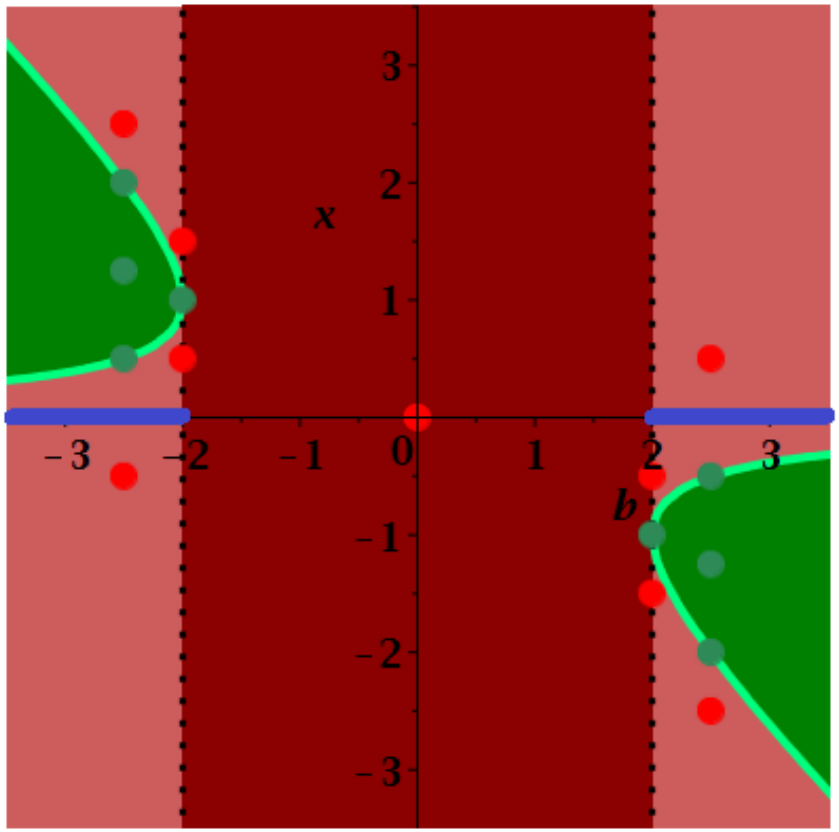
To solve we:

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?

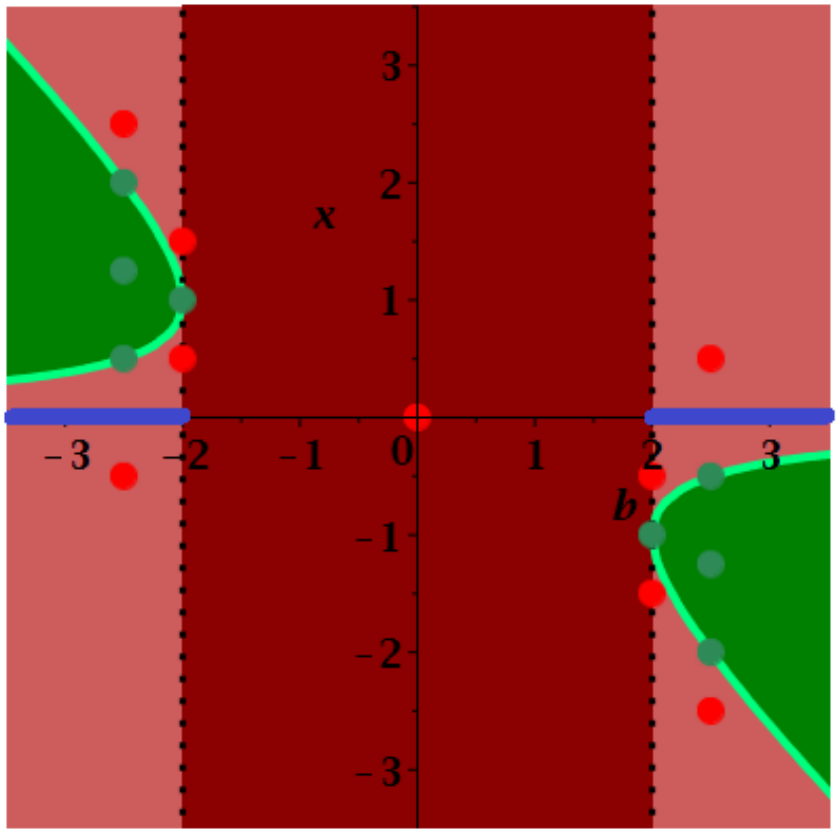
$$\exists x, x^2 + bx + 1 \leq 0$$

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:



$$\Longrightarrow$$

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

QE via CAD in General

In general we can perform Real QE via CAD as follows.

- Eliminate existential quantifiers by projecting the true cells.
- Convert universal quantifiers to existential by using

$$\forall x P(x) = \neg \exists x \neg P(x)$$

and then proceeding with existential and negating the answer.

Recall our original example was $\forall x, x^2 + bx + 1 > 0$. The above leads us to study $\exists x, x^2 + bx + 1 \leq 0$ which from the previous slide we know has solution $b \leq -2 \vee b \geq +2$. The solution to our universally quantified problem is then the negation of this: $-2 < b \wedge b < +2$ (as we found numerically earlier).

Note: Projection and negation are both easy with cylindrical cells!

QE via CAD in General

In general we can perform Real QE via CAD as follows.

- Eliminate existential quantifiers by projecting the true cells.
- Convert universal quantifiers to existential by using

$$\forall x P(x) = \neg \exists x \neg P(x)$$

and then proceeding with existential and negating the answer.

Recall our original example was $\forall x, x^2 + bx + 1 > 0$. The above leads us to study $\exists x, x^2 + bx + 1 \leq 0$ which from the previous slide we know has solution $b \leq -2 \vee b \geq +2$. The solution to our universally quantified problem is then the negation of this: $-2 < b \wedge b < +2$ (as we found numerically earlier).

Note: Projection and negation are both easy with cylindrical cells!

QE via CAD in General

In general we can perform Real QE via CAD as follows.

- Eliminate existential quantifiers by projecting the true cells.
- Convert universal quantifiers to existential by using

$$\forall x P(x) = \neg \exists x \neg P(x)$$

and then proceeding with existential and negating the answer.

Recall our original example was $\forall x, x^2 + bx + 1 > 0$. The above leads us to study $\exists x, x^2 + bx + 1 \leq 0$ which from the previous slide we know has solution $b \leq -2 \vee b \geq +2$. The solution to our universally quantified problem is then the negation of this: $-2 < b \wedge b < +2$ (as we found numerically earlier).

Note: Projection and negation are both easy with cylindrical cells!

Outline

QE via CAD Implementations

Both of the big proprietary Computer Algebra Systems have QE implementations: MATHEMATICA (the `Resolve` command) and MAPLE (`RegularChains:-SemiAlgebraicSetTools`; `QuantifierElimination:-CylindricalAlgebraicDecompose`; `RootFinding:-Parametric:-CellDecomposition`); & `SyNRAC`.

The specialist computer algebra system QEPCAD-B is dedicated to QE via CAD and is available for free. It can be used via an intelligent interface TARSKI, is available as a sub-package of SAGEMATH (for Python), and as part of GEOGEBRADISCOVERY which can even run in your browser: `https://autogeo.online/`

CAD implementations also exist in the SMT-solvers, SMT-RAT, YICES2, Z3 and CVC5. All are available for free – but note that these are designed for SMT queries.

Warning: CAD Complexity

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!

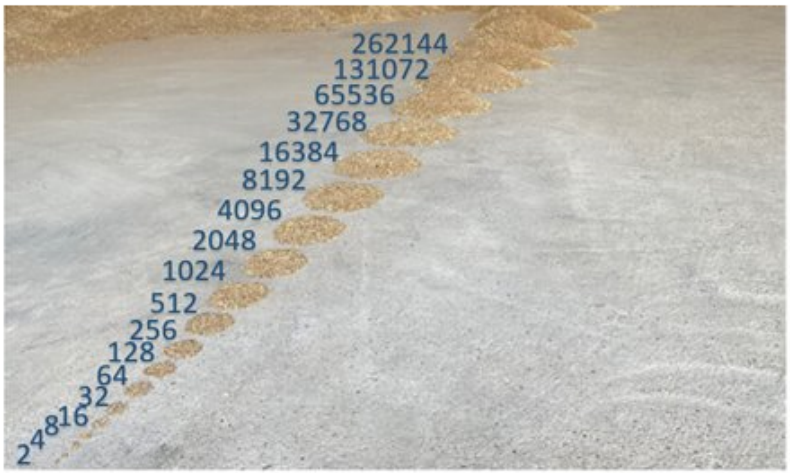
C. Brown and J.H. Davenport.
The complexity of quantifier elimination and cylindrical algebraic decomposition.
In Proc. ISSAC '07, pages 54–60. ACM, 2007.

Thus in practice applications are only realistic for 4 variables, unless specific optimisations are utilised:

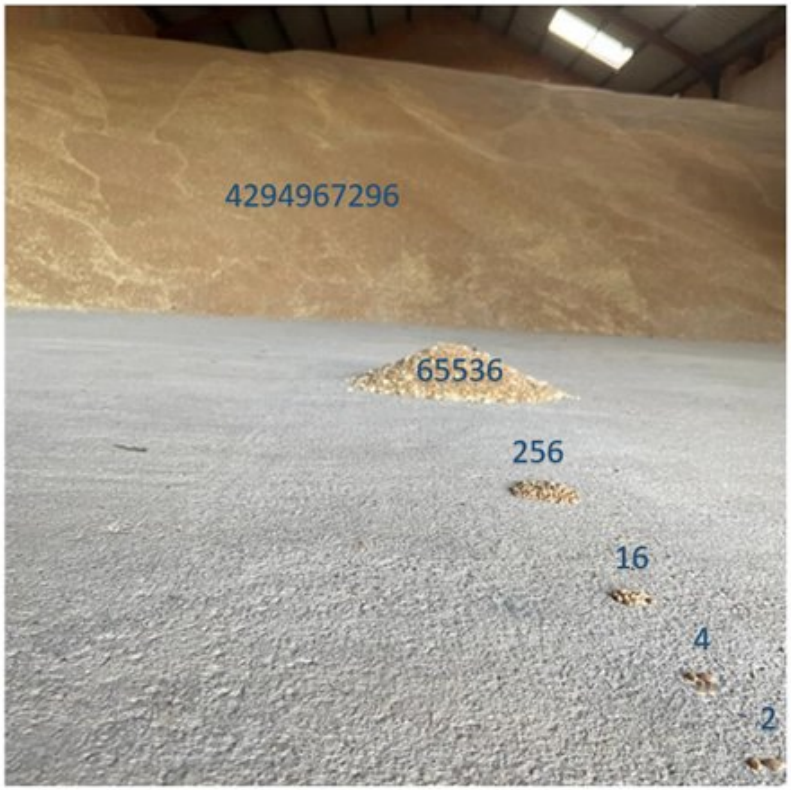
$$2^{2^3} = 256 \qquad 2^{2^4} = 65,536 \qquad 2^{2^5} = 4,294,967,296$$

The Doubly Exponential Wall

Images by Tereso del Rio



Exponential Growth



Doubly Exponential Growth

Alternatives to CAD for QE References (Slide 12/50)

There are alternatives to CAD to achieve Real QE:


T. Sturm.
Thirty years of virtual substitution: Foundations, techniques, applications.
In Proc. ISSAC 2018, pages 11–16, ACM, 2018.


R. Fukasaku, H. Iwane, and Y. Sato.
Real quantifier elimination by computation of comprehensive Gröbner systems.
In Proc. ISSAC '15, pages 173–180. ACM, 2015.


J. Renegar.
Recent progress on the complexity of the decision problem for the reals.
In B.F. Caviness and J.R. Johnson, eds, *Quantifier Elimination and Cylindrical Algebraic Decomposition*, pages 220–241. Springer-Verlag, 1998.


D.Y. Grigor'ev and N.N. Vorobjov.
Solving systems of polynomial inequalities in subexponential time
Journal of Symbolic Computation, 5, pp. 37−64, 1988.

However, CAD remains the general purpose QE backbone, so it is worth thinking what can be done better!

Outline

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 etc. But this problem 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. (Q) How to adapt CAD to take note of the logic?

The SMT Approach

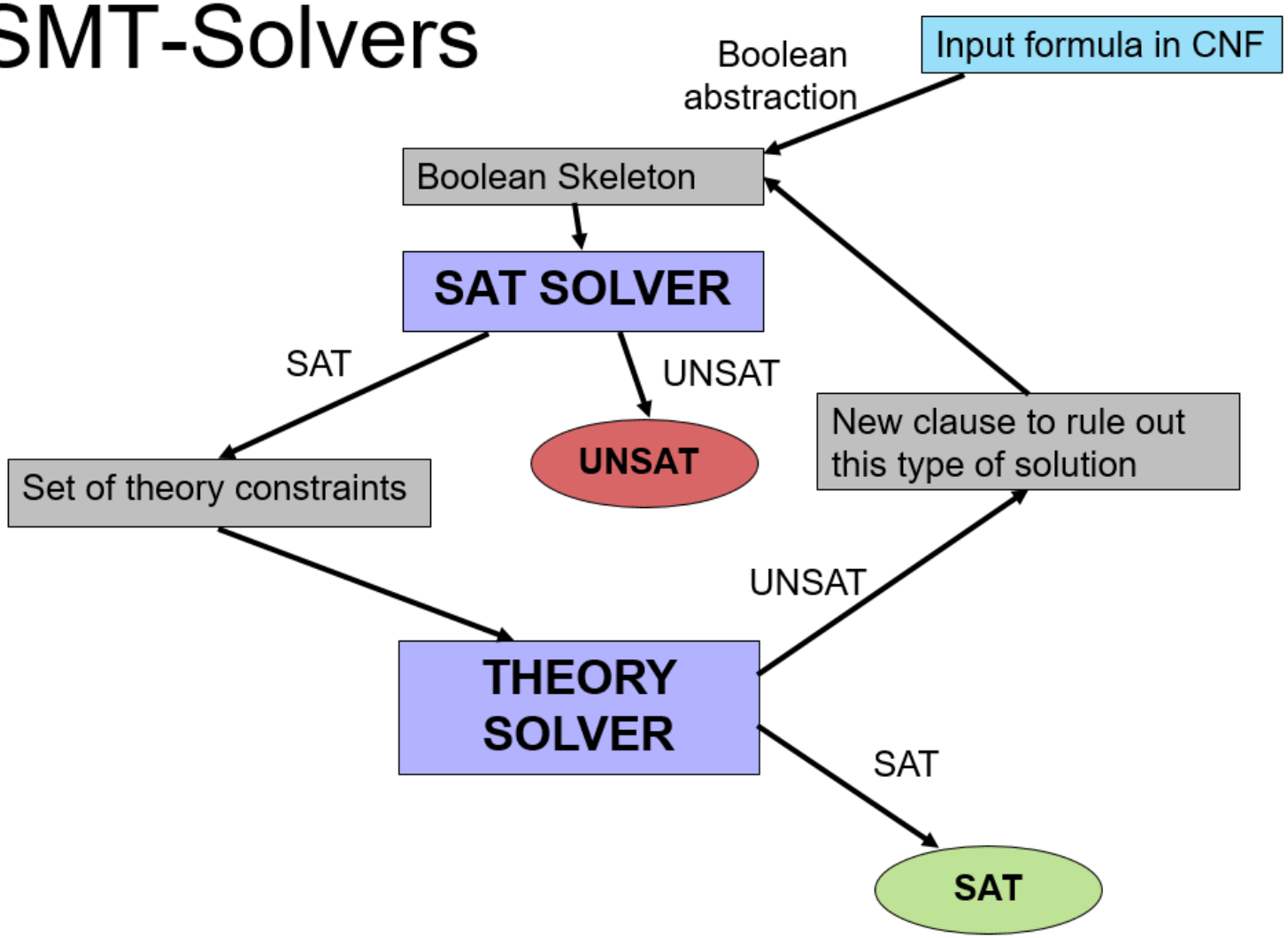
One approach to satisfiability problems more generally is to separate out the logic from the arithmetic theory.

- 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 involved).

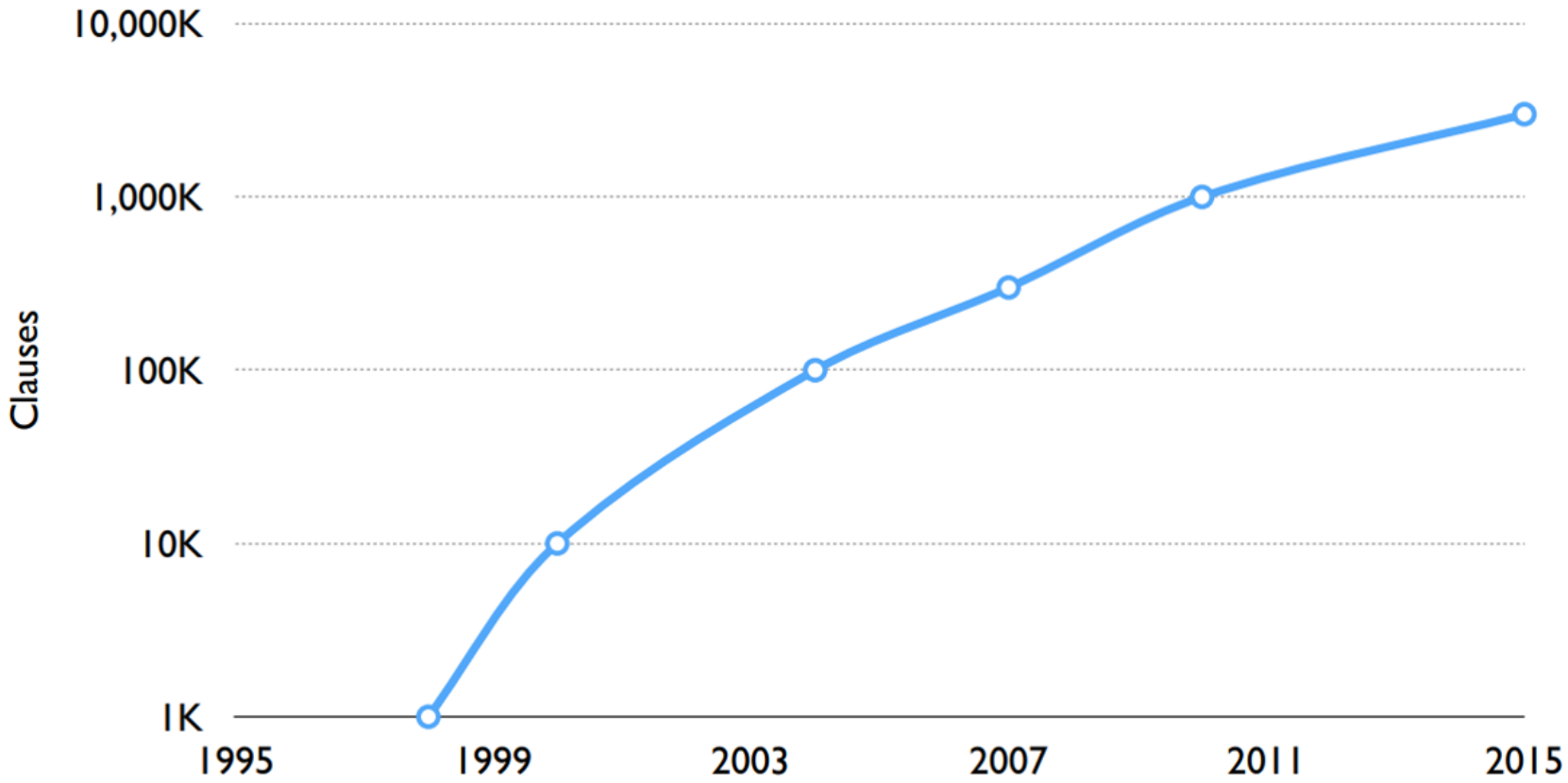
This approach is called **Satisfiability Modulo Theories** (SMT) or sometimes CDCL(T).

SMT-Solvers


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

Why use the SMT Approach?

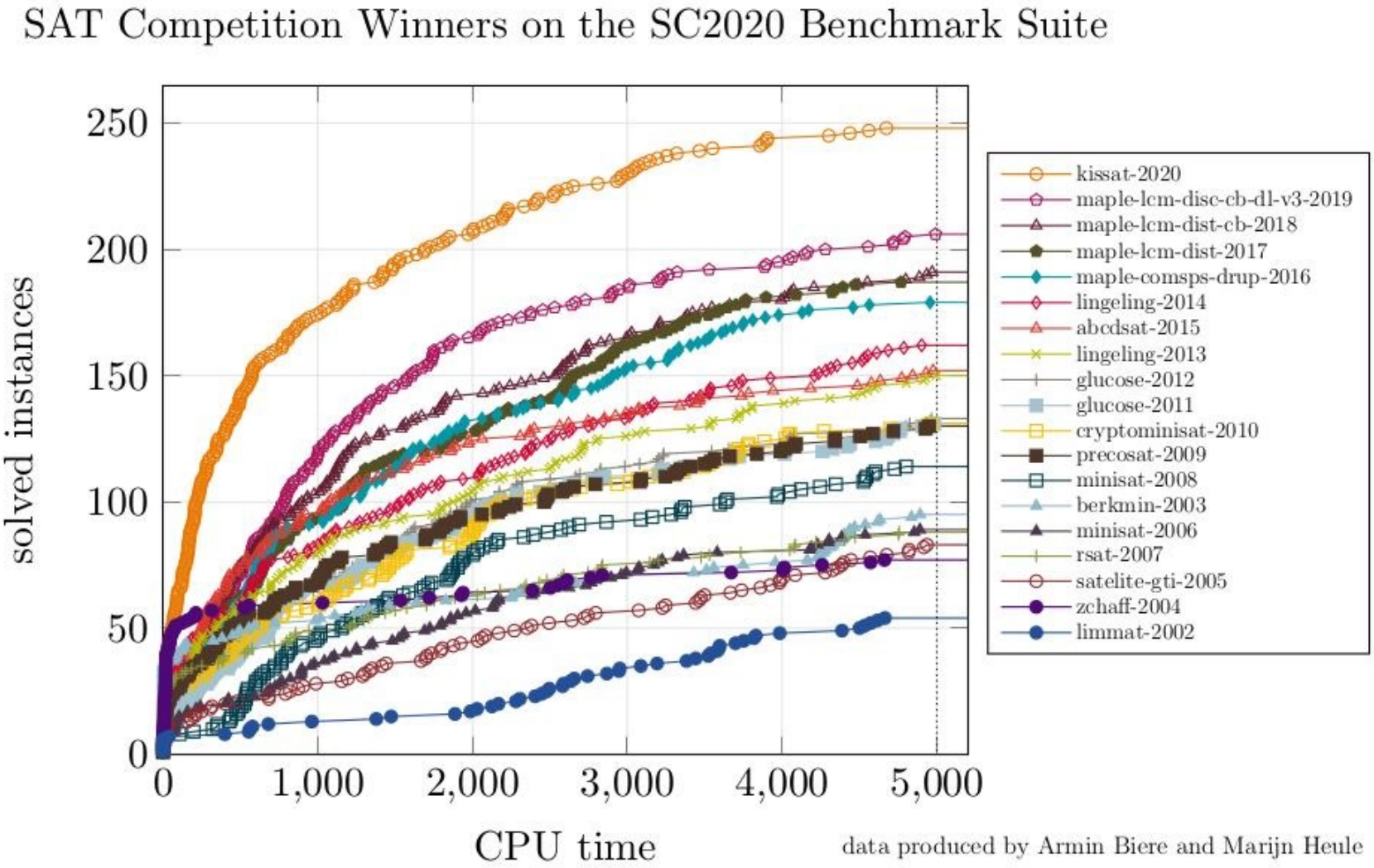
To take advantage of the incredible progress in SAT solvers!



Based on a slide from Vijay Ganesh

Why use the SMT Approach? (Slide 16/50)

To take advantage of the incredible progress in SAT solvers!



Why are SAT solvers so good?

The SAT problem is famously NP: so exponential time in the worst case. But in seems that on average, a solution can be found much quicker through an intelligent search algorithm.

1960s: The Davis-Putnam-Logemann-Loveland (DPLL) algorithm explained how a mixture of propagation and backtracking could allow large swathes of the search space to be ruled out at once.

1990s: The Conflict-Driven Clause Learning (CDCL) algorithm built on this by learning new conflict clauses before backtracking: clauses implied by the original formula that explicitly rule out a bad guesses and others that are similar.

2000s: Extensive optimisations to CDCL; ML heuristics and meta solvers etc.

CAD as an NRA Theory Solver

We can use CAD as an SMT NRA theory solver but the implementation must first 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

How good is this approach?

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

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 some very large computation.

(Q) How to adapt CAD further to avoid this?
(A) Try to follow the ideas within SAT-solvers themselves: search the sample spaces by making guesses, propagating, and generalising conflicts to avoid similar parts of the search space.

Outline

Conflict Driven Cylindrical Algebraic Covering (Slide 20/50)

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.

- Search based: choose sample point and if not satisfying then build CAD cell around it.
- Cells gradually form a *covering* of $\mathbb{R}^n$ (instead of a decomposition). Can use far fewer cells!
  - Cells are still arranged in cylinders making projection easy.
  - *Conflict Driven* search guides us away from past conflicts.

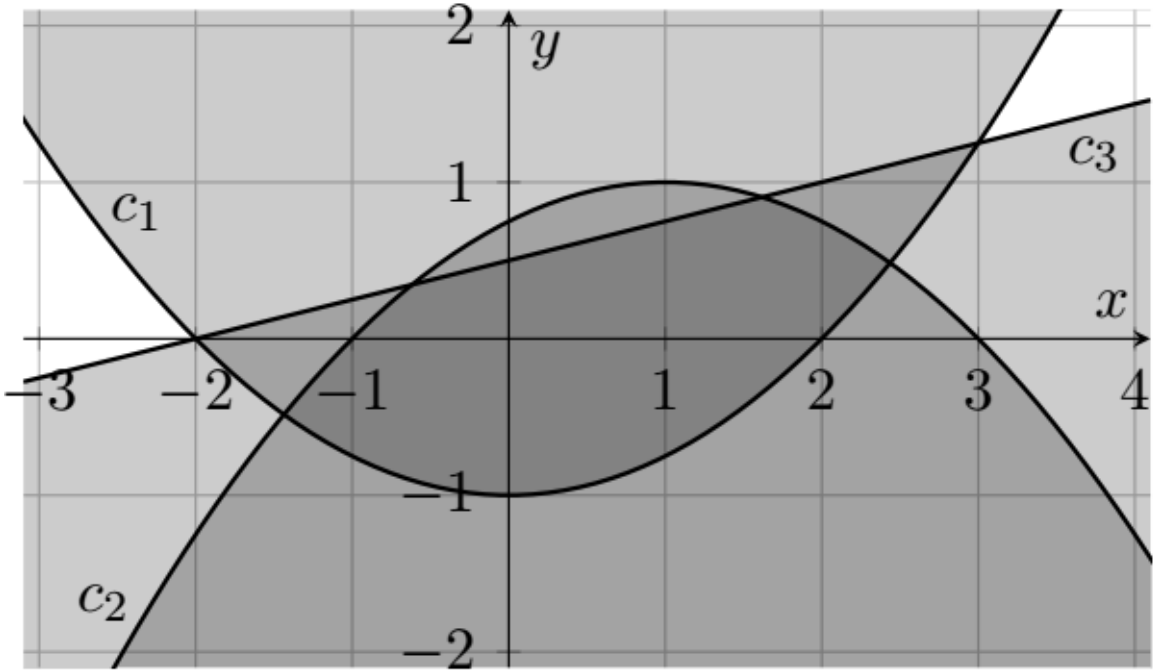
CDCAC: Basic Idea

- 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 a covering for the lowest dimension is obtained.

Example

Suppose we want to identify solutions to the constraints

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

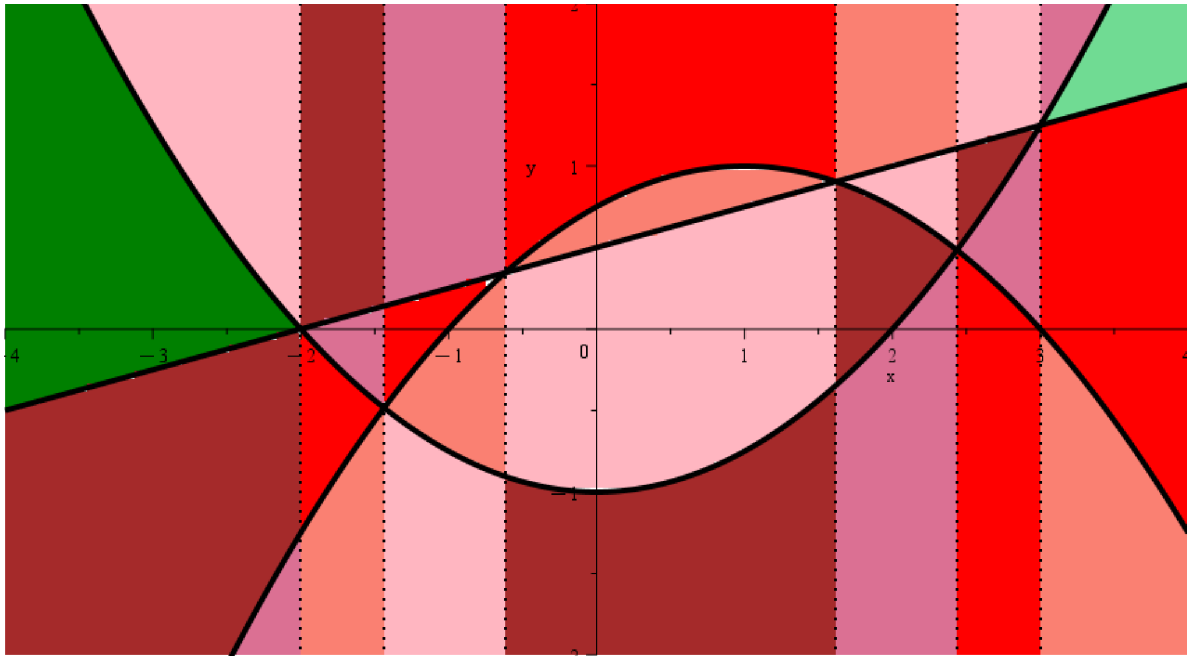


A CAD for the three defining polynomials has 79 cells in 13 cylinders. The following slides by Gereon Kremer show how CDCAC is guided to the satisfying cell.

Example

Suppose we want to identify solutions to the constraints

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

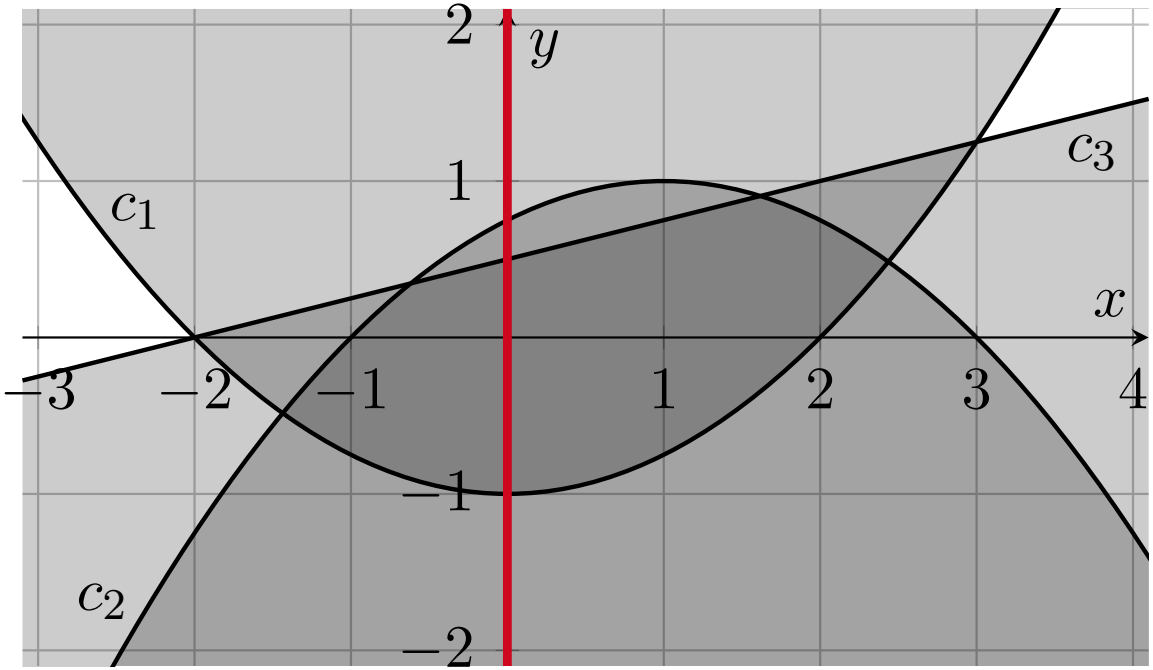


A CAD for the three defining polynomials has 79 cells in 13 cylinders. The following slides by Gereon Kremer show how CDCAC is guided to the satisfying cell.

RWTH AACHEN UNIVERSITY

An example

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



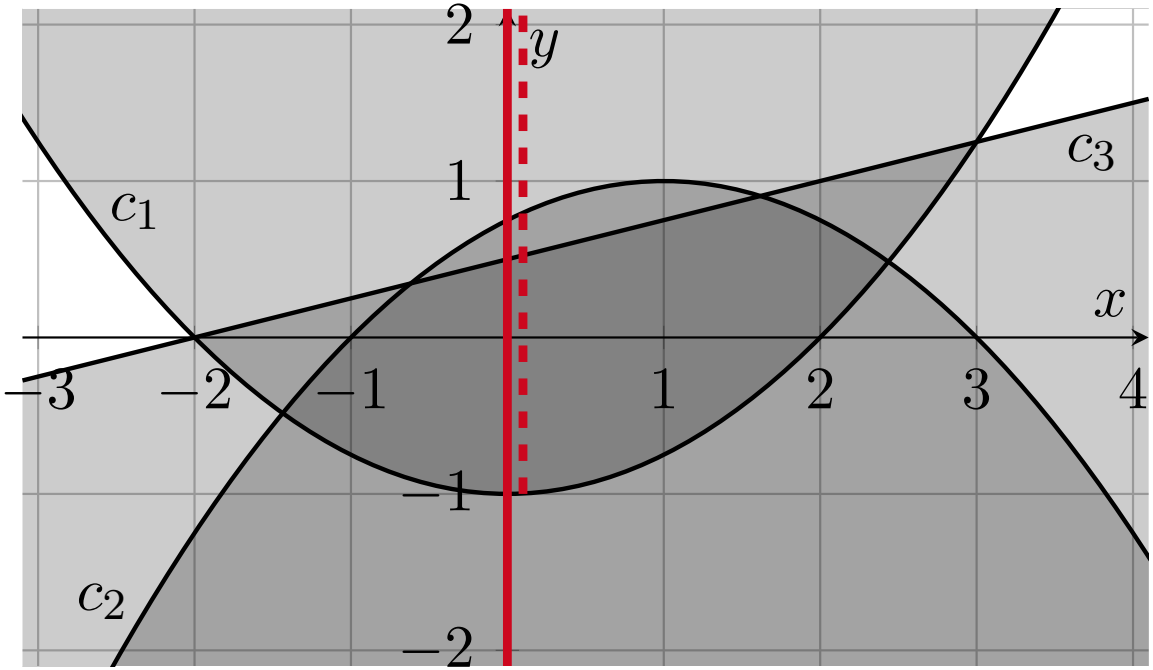
No constraint for x

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

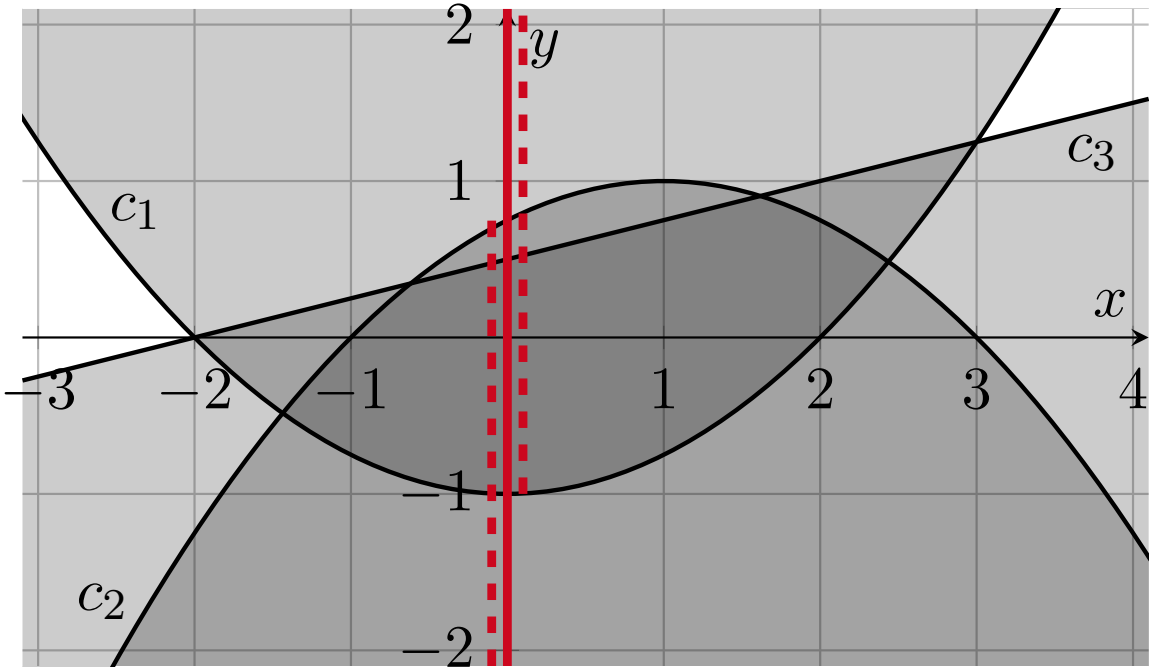
Guess $x \mapsto 0$

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

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 $x \mapsto 0$

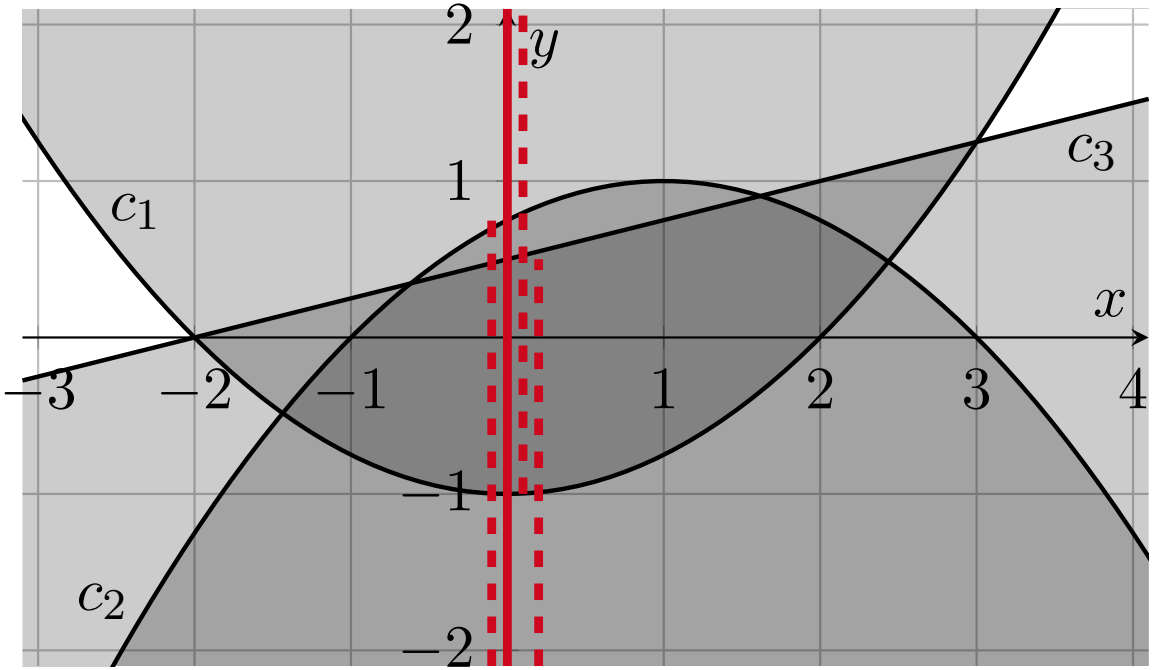
$c_1 \to y \notin (-1, \infty)$

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

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 $x \mapsto 0$

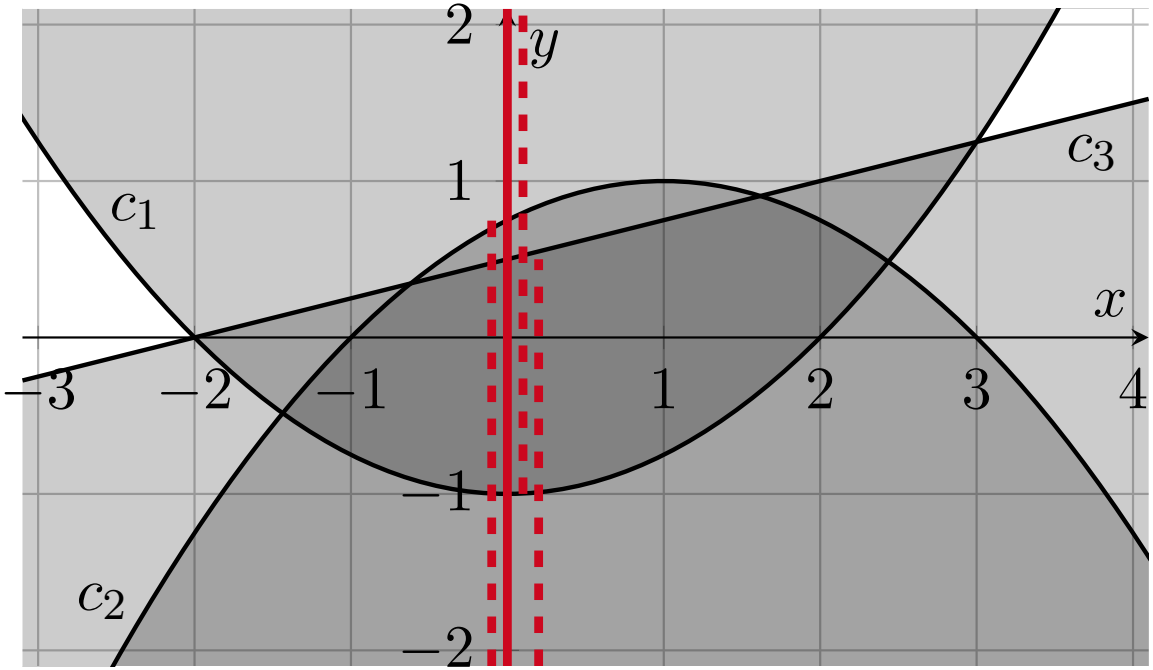
$c_1 \to y \notin (-1, \infty)$

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

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

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 $x \mapsto 0$

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

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

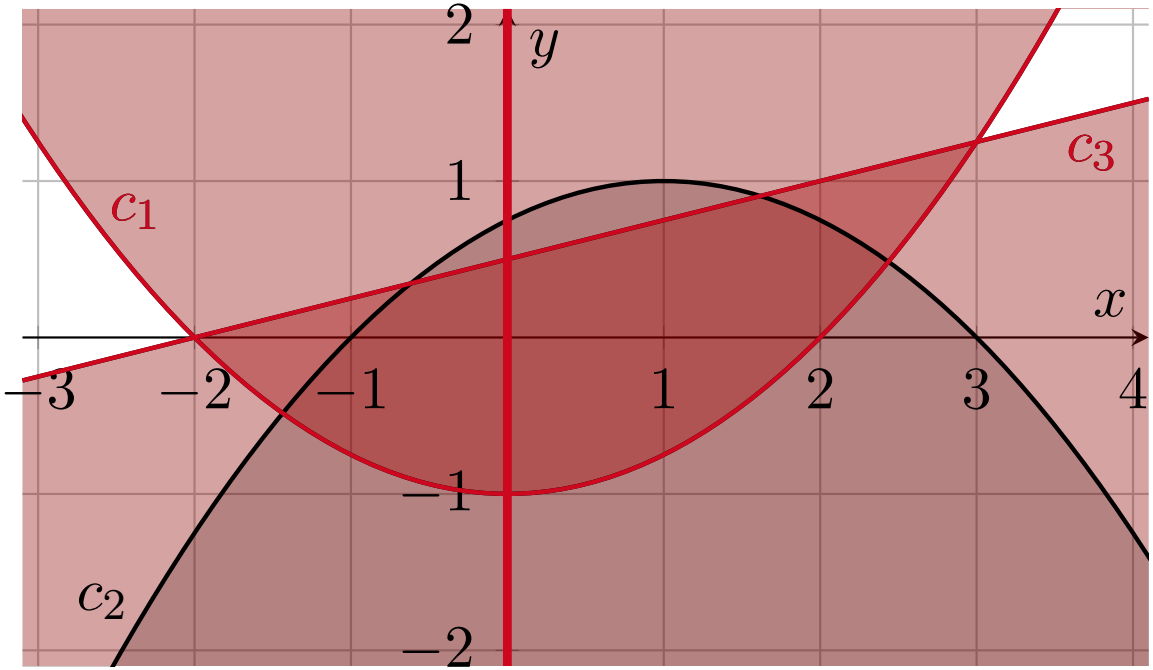
$c_3 \to y \notin (-\infty, 0.5)$

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 $x \mapsto 0$

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

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

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

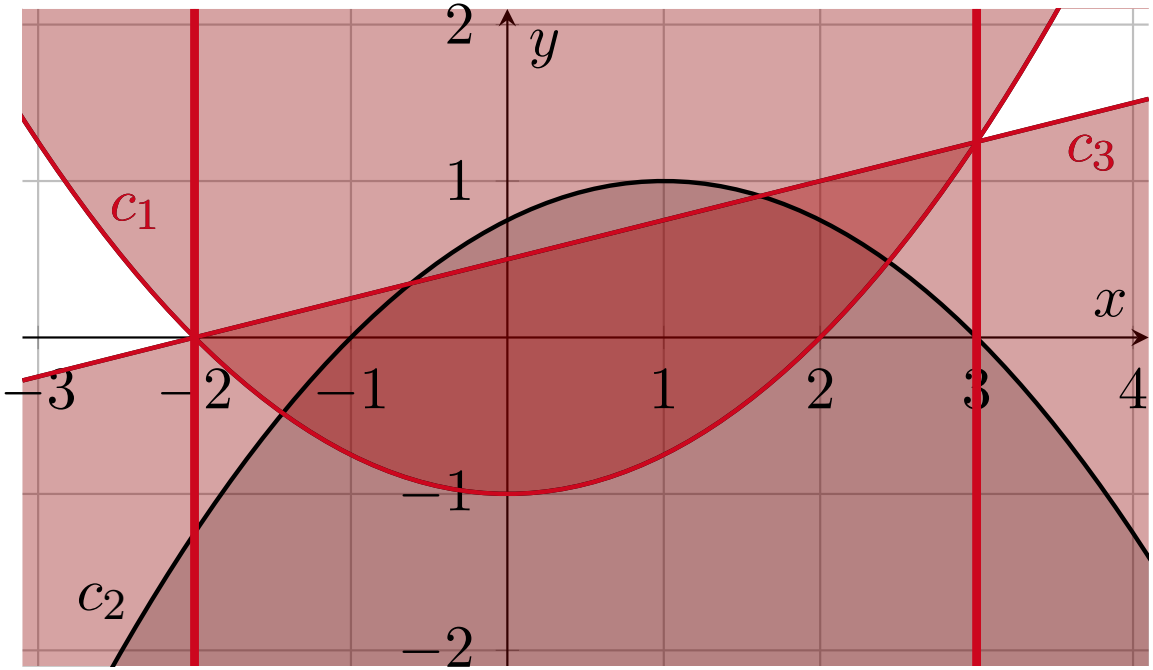
Construct covering

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

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

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

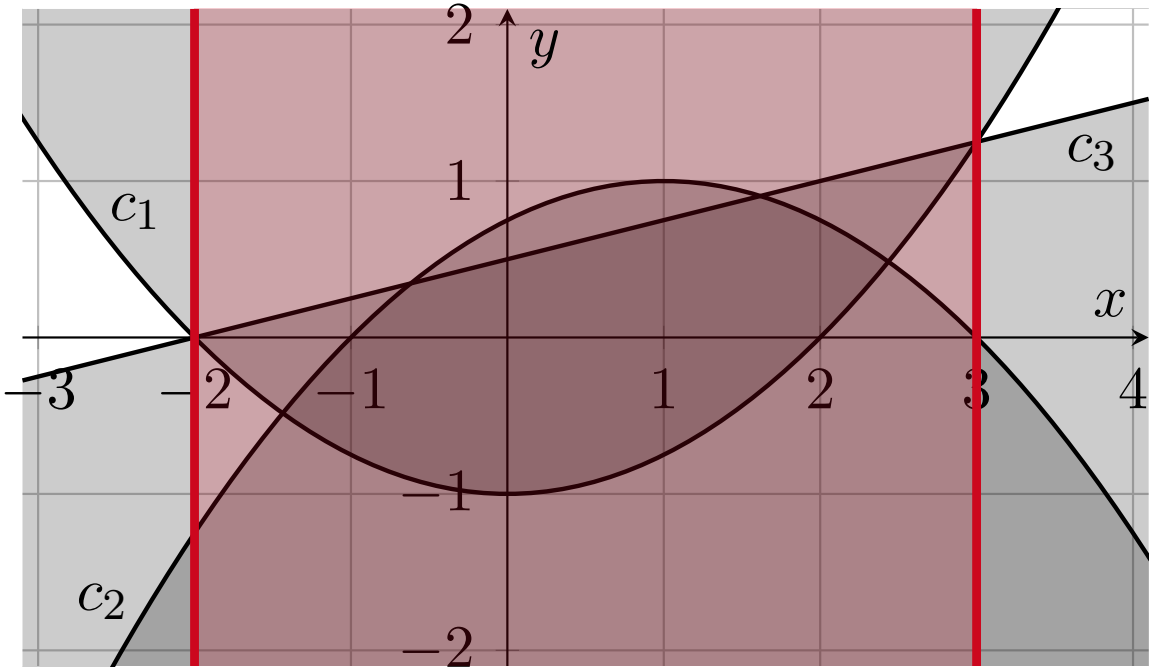
Construct interval for x

$x \notin (-2, 3)$

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

Further work on CDCAC I

CDCAC was reimplemented in cvc5 which won the 2022 SMT Competition overall, including the QFNRA track. Works in tandem with the incomplete method of incremental linearisation.


G. Kremer, A. Reynolds, C. Barrett, and C. Tinelli.
Cooperating techniques for solving nonlinear real arithmetic in the cvc5 SMT solver (system description).
In: Automated Reasoning, (LNCS 13385), 95–105, 2022.


A. Cimatti, A. Griggio, A. Irfan, M. Roveri, and R. Sebastiani.
Invariant checking of NRA transition systems via incremental reduction to LRA with EUF.
Proc. TACAS '17, 58–75. Springer, 2017.

Further work on CDCAC II

An outline of the theory to extend CDCAD from the SMT case to full QE was presented at SC-Square 2022 (full paper under review).

G. Kremer, and J. Nalbach
Cylindrical Algebraic Coverings for Quantifiers.
Proc. SC^2 2022, CEUR-WS volume 3458.

An implementation in SMT-RAT was demonstrated at SC^2 2024.

We hypothesise CDCAC gives easier route to formal validation.

E. Abraham, J. Davenport, M. England, G. Kremer, Z. Tonks.
New Opportunities for the Formal Proof of Computational Real Geometry?
Proc. SC^2 2020, CEUR 2752, 178–188, 2020.

Outline

Certificates and Verification

CAD/CDCAC provide a satisfying point as certificate of SAT.

But in the case of unsatisfiability, one would need to verify that:
(a) the cells form a decomposition / covering (doable); and
(b) each cells satisfies the stated invariance property (difficult).

Machine readable proofs of unsatisfiability is a growing trend in SMT: cvc5 achieves this for many theories, but not yet NRA. Only partial attempts at CAD formalisation.

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

C. Cohen and A. Mahboubi.

Formal proofs in real algebraic geometry: from ordered fields to QE.

Logical Methods in Computer Science, 8(1):1–40, 2012.

https://doi.org/10.2168/LMCS-8(1:2)2012

Proof System for building a cylindrical cell (Slide 27/50)

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

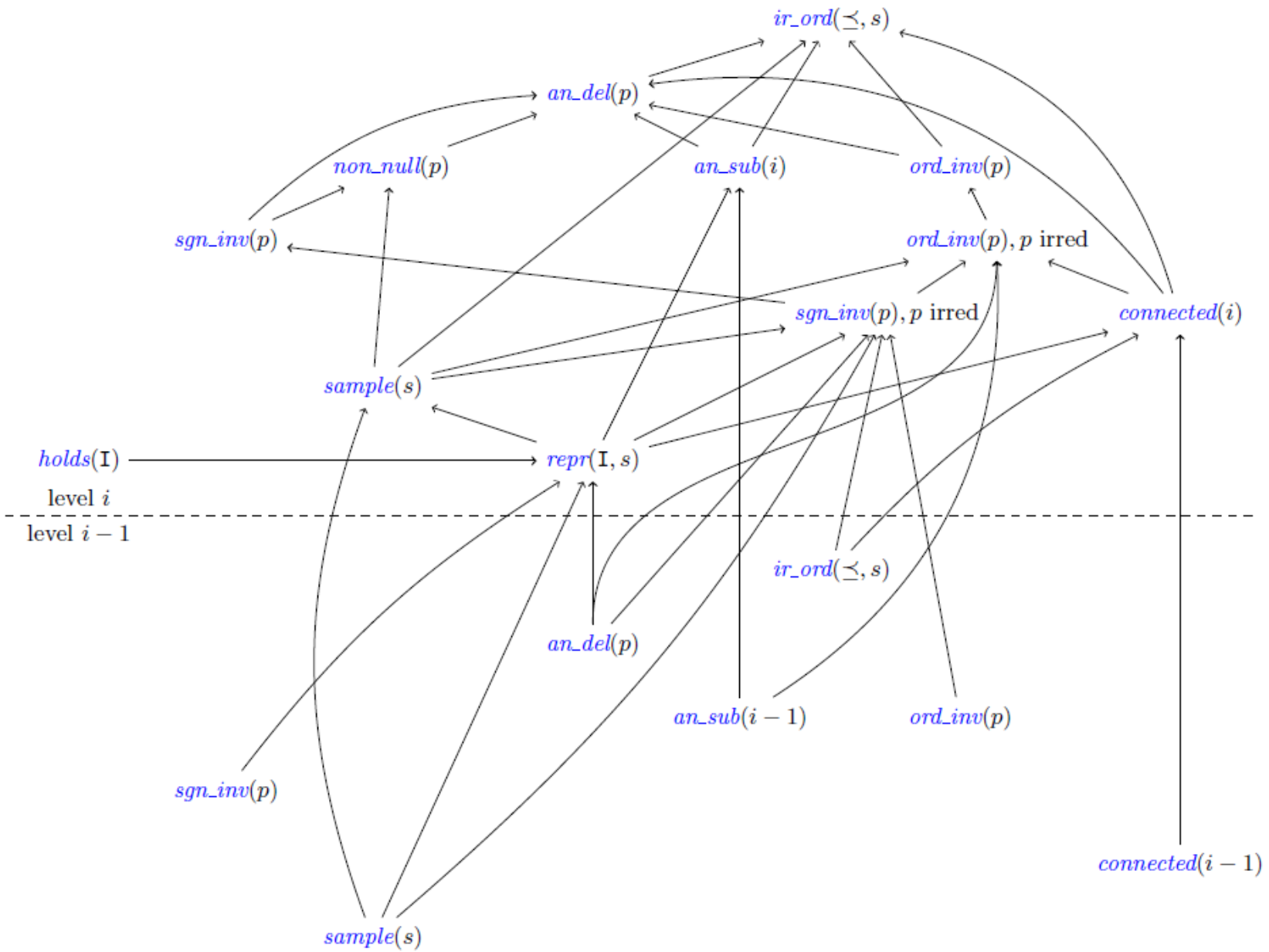
The algorithm then simply searches for a chain of proof rules to prove a desired property (employing heuristics to take choices where there is freedom in how the chain can to be built).

J. Nalbach, E. Ábrahám, P. Specht, C.W. Brown, J.H. Davenport, M. England.

Levelwise construction of a single cylindrical algebraic cell.

Journal of Symbolic Computation, **123**, Article Number 1022882023, 2024.

https://doi.org/10.1016/j.jsc.2023.102288


$ir\_ord(\preceq, s)$
$an\_del(p)$
$non\_null(p)$
$an\_sub(i)$
$ord\_inv(p)$
$sgn\_inv(p)$
$ord\_inv(p), p$ 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 Presentation

Unlike traditional pseudo-code, a proof system presentation clearly separates out the parts of the algorithm upon which correctness rests, from any heuristic decisions being taken. This:

1. allows for cleaner proofs of correctness; and
2. more structured experimentation to optimise the heuristics.

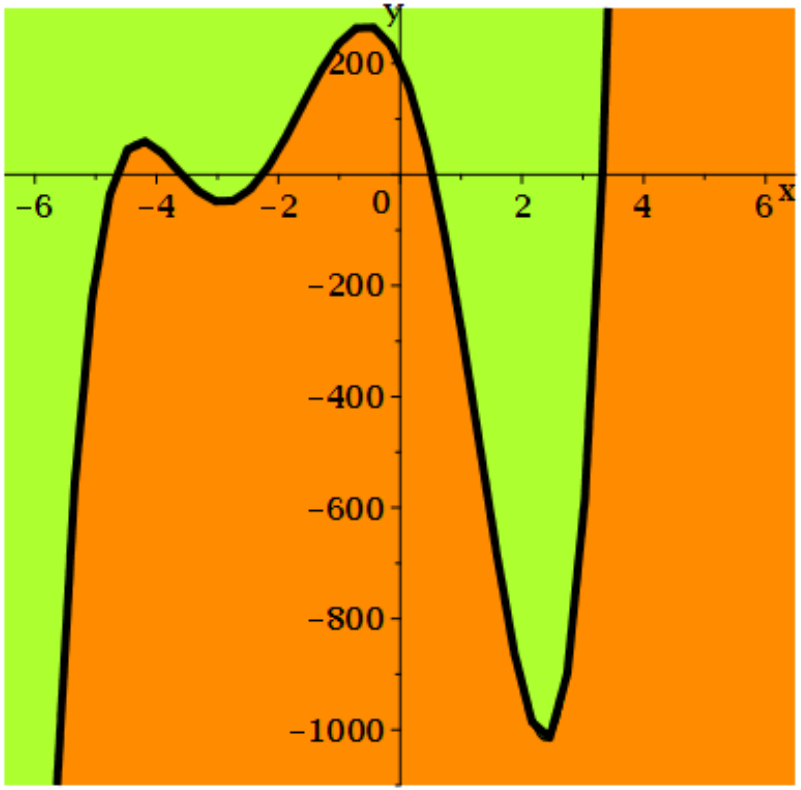
Proof system presentation is common in the SAT/SMT/logic communities where there is more intense work on the optimization of such heuristic choices. Rarely used in computer algebra: but our paper shows there are potential benefits to it, especially for large complex algorithms like CAD.

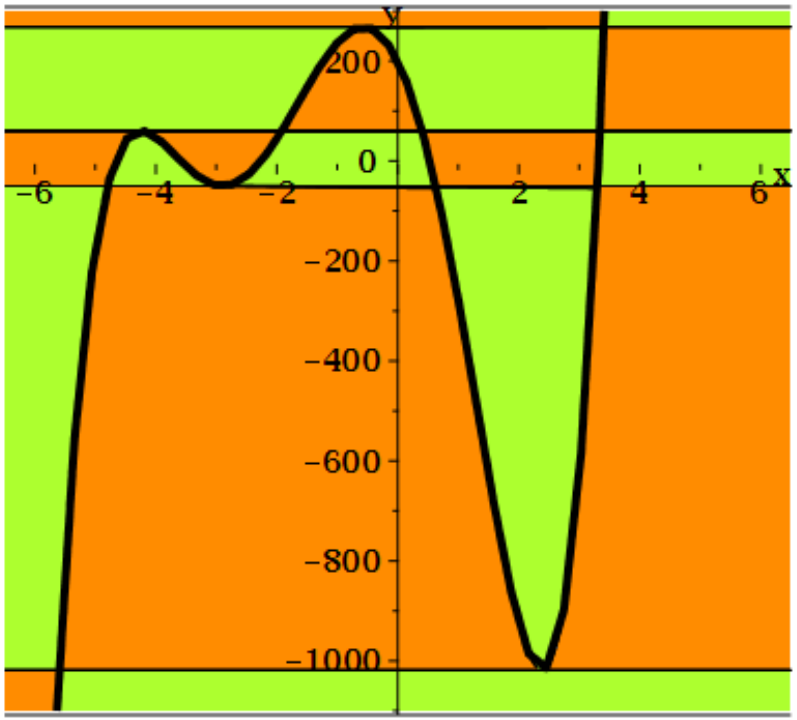
Outline

Outline

Variable Ordering for CAD

CAD (and variants like CDCAC) requires a variable ordering: multiple valid orderings can lead to an acceptable decomposition, but some lead to smaller decompositions via less computation.





CAD Variable Ordering Choice

Depending on our application we may have a free or constrained choice in variable ordering. When using CAD for QE we must project variables in the order of quantification, but we are free to change order within quantifier blocks (and with the free variables).

The variable ordering has been long known to effect the number of cells produced in a CAD or even the tractability of a problem.

There are various *human-designed heuristics* to make the choice:

- Simple heuristics which use only simple measures of polynomials (degrees, sparsity etc.)
- Expensive heuristics which use more involved algebraic computations.

Simple human-designed heuristics

Brown's Heuristic: chooses variable based on overall degree; breaking ties with total degree of the terms with that variable; breaking ties with number of terms the variable is in.

C. Brown

Companion to the Tutorial: Cylindrical Algebraic Decomposition, presented at ISSAC 2004.

www.usna.edu/Users/cs/wcbrown/research/ISSAC04/handout.pdf

gmods: chooses variable based on the sum of its degree in each polynomial.

T. del Río and M. England

New Heuristic to Choose a Cylindrical Algebraic Decomposition Variable Ordering Motivated by Complexity Analysis

Proc. CASC 2022, LNCS 13366, pp. 300−−317. Springer, 2022.

https://doi.org/10.1007/978-3-031-14788-3_17

Outline

Symbolic Computation – Machine Learning (Slide 32/50)

Machine Learning (ML) can use statistics and big data to learn how to perform tasks that have not been explicitly programmed.

(Q) So can ML replace symbolic computation?

There is a growing body of research on the use of ML in place of expensive symbolic computation, but ML is never 100% and for many computer algebra tasks it is not cheap to symbolically check the correctness of the answer.

ML can be applied to symbolic computation safely by having it guide existing algorithms; having it make choices which do not effect the mathematical correctness of the final result, but which do effect the resources required to find that result (and often how the result is presented).

Symbolic Computation – Machine Learning (Slide 32/50)

Machine Learning (ML) can use statistics and big data to learn how to perform tasks that have not been explicitly programmed.

(Q) So can ML replace symbolic computation?

There is a growing body of research on the use of ML in place of expensive symbolic computation, but ML is never 100% and for many computer algebra tasks it is not cheap to symbolically check the correctness of the answer.

ML can be applied to symbolic computation safely by having it guide existing algorithms; having it make choices which do not effect the mathematical correctness of the final result, but which do effect the resources required to find that result (and often how the result is presented).

This is INTERESTING Machine Learning (Slide 33/50)

So ML has great potential for Symbolic Computation. But note this is also a particularly challenging / interesting ML domain:

- No a priori limit on the input space.
- Supervised learning hard: because labelling dataset needs lots of expensive symbolic computation.
- Unsupervised learning is hard: unclear if a particular outcome is good or bad without seeing the competition!
- What constitutes a meaningful and representative data set?
- Insufficient quantities of real world data for deep learning.
- How to perform data augmentation / synthetic data generation for generalisability on relevant problems?

First use of ML for CAD variable ordering (Slide 34/50)

First attempt used a support vector machine classifier to choose which of the three variable ordering heuristics we had implemented in our CAD code to follow for a given problem instance: no one heuristic dominated the others; ML choice did better than any one.

Z. Huang, M. England, D. Wilson, J. Davenport, L. Paulson and J. Bridge.

Applying machine learning to the problem of choosing a heuristic to select the variable ordering for cylindrical algebraic decomposition.

Intelligent Computer Mathematics (LNCS 8543), pp. 92-107. Springer Berlin Heidelberg, 2014. http://dx.doi.org/10.1007/978-3-319-08434-3_8

The first use of Machine Learning to optimise computer algebra.

EPSRC ML4QE Project

Experimented with ML methodology to choose the ordering:

- Classifier model choice.
- New feature extraction methods to represent polynomials to Machine Learning models: brute force combinations of simple statistics followed by selection
- Tailored hyper-parameter selection to runtime rather than classification accuracy.

D. Florescu and M. England.

A Machine Learning Based Software Pipeline to Pick the Variable Ordering for Algorithms with Polynomial Inputs.

Mathematical Software (LNCS 12097), pp. 302–311. Springer International Publishing, 2020.
https://doi.org/10.1007/978-3-030-52200-1_30

Dataset Issues

Above work used the `QF_NRA` SMT-LIB benchmarks: a collection of satisfiability problems whose atoms are non-linear polynomial constraints over the reals.

Chosen as the largest (only substantial) existing set of benchmarks for CAD. Problems are meaningful: come from "*industry*" (theorem provers MetiTarski and Keymaera), biology, economics, dynamic geometry proofs, etc. **However:**

- Dataset is highly imbalanced with respect to CAD variable ordering. Risk of over-fitting given no reason to expect imbalance in general data.
- Data is highly dominated by one source of problems.
- Dataset contains many problems that are differ only by a constant: risk of data leakage between testing and training.

Dataset Lessons

1. Use variable permutations on problem instances to:
   - Balance dataset (random permutation on each instance).
   - Augment dataset (from each problem instance create a set of instances from all possible permutations).
2. Merge problem instances who have the same CAD tree structure in each variable ordering.

Loss of "accuracy" from (1a) offset by (1b). (2) allows saw no loss in learning ability.

Tereso del Río and M. England.
Data Augmentation for Mathematical Objects.
Proc. SC^2 2023, CEUR-WS 3455, pp. 29−−38, 2023. https://ceur-ws.org/Vol-3455/

From Classification to Regression

Prior work framed this as ML *Classification* (choose from discrete set of variable orderings). May be reframed as ML *Regression* predict the time taken with a particular ordering: multiple predictions then compared to choose the final ordering. Why?

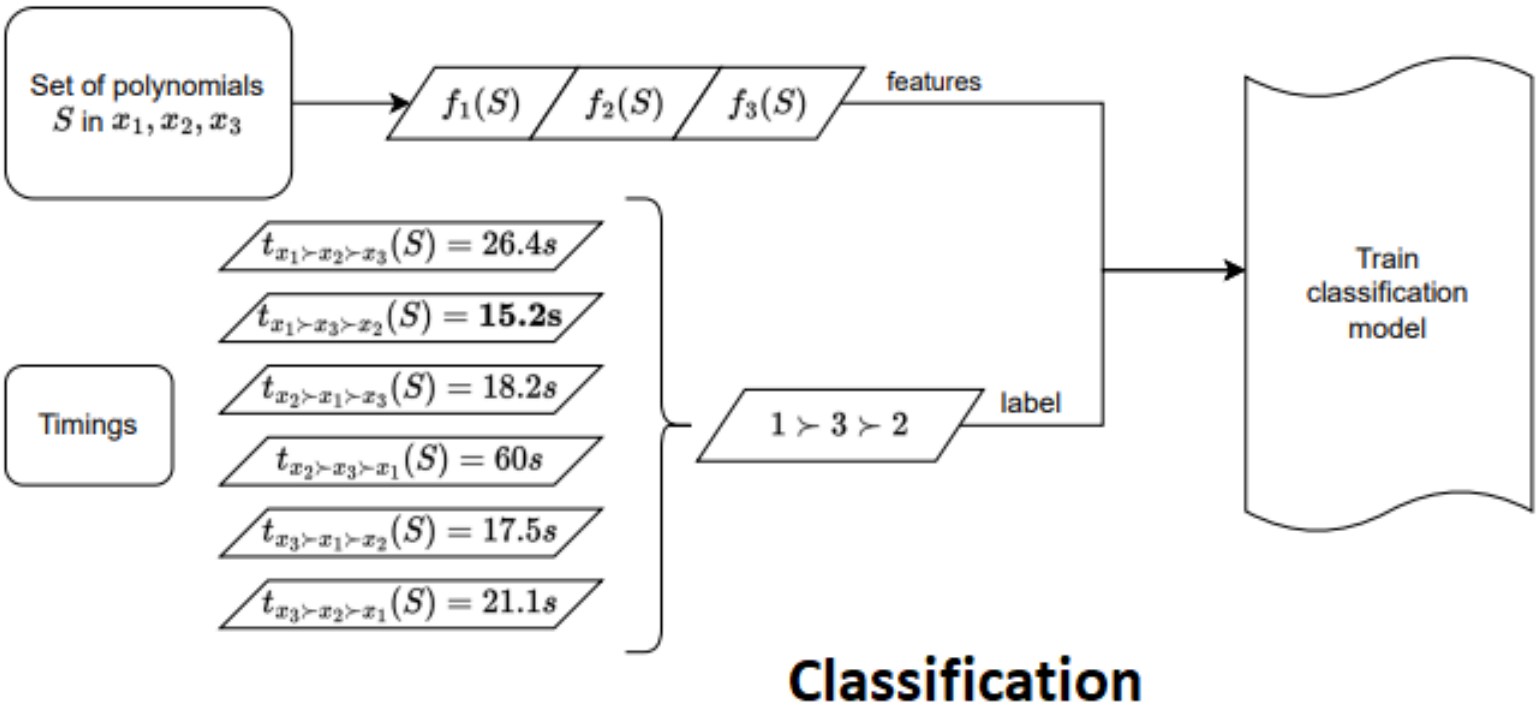
- A model trained with regression has access to more information: not just which ordering did best but also which came second, which third etc.
- However, regression is a more difficult task (but we only need to be good enough at it to make a ranking).

T. del Río and M. England.

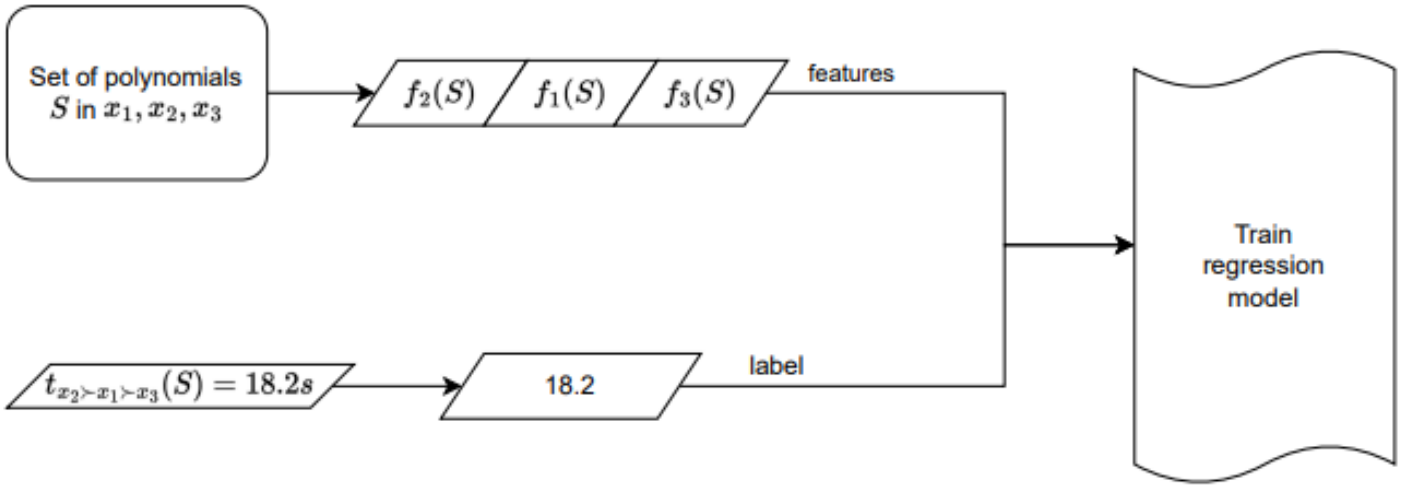
Lessons on Datasets and Paradigms in Machine Learning for Symbolic Computation: A Case Study on CAD..

In Press: Springer Mathematics in Computer Science, 2024.
Preprint: https://doi.org/10.48550/arXiv.2401.13343

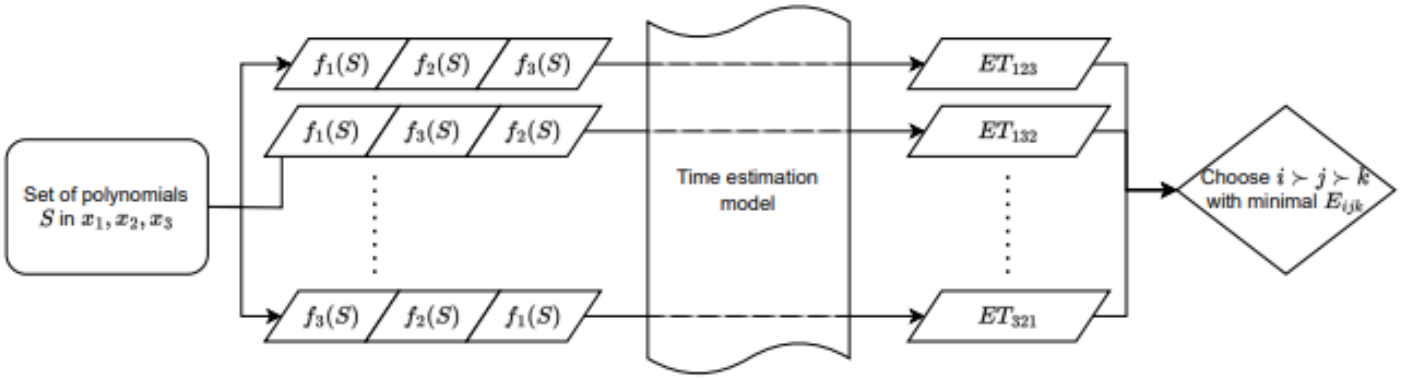
Classification vs Regression Flowcharts



Classification vs. Regression Flowcharts

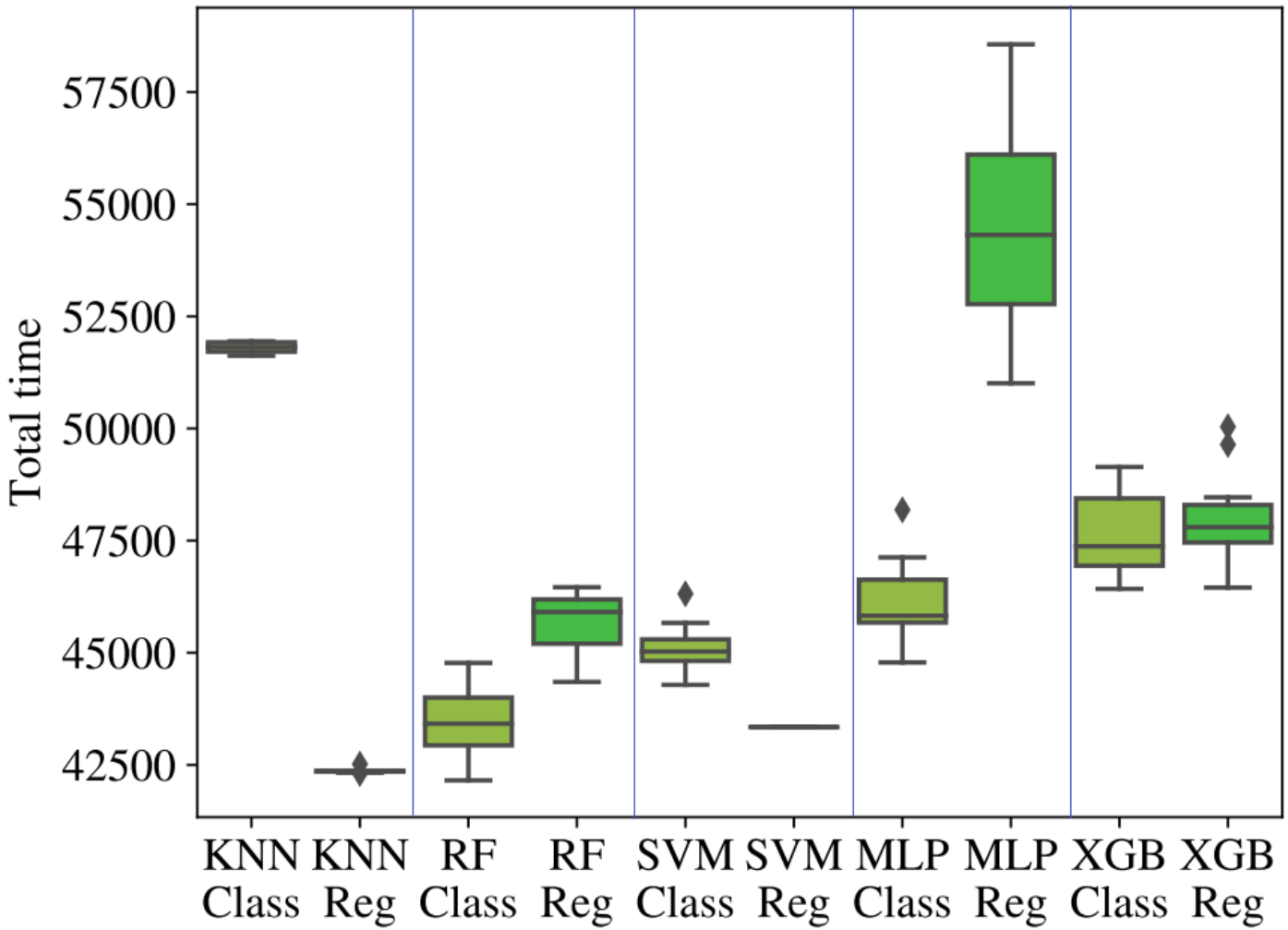


Regression Training
Regression Testing



Classification vs. Regression Results

Regression not universally beneficial, but new state-of-the-art ML.



Outline

Explainable AI (XAI)

Explainable AI (XAI) is increasingly important: both as an error check on the ML process; and to build trust in new AI technologies. Two main types of XAI:

- **Ante-hoc explainable models** are transparent by design in their decisions, e.g. linear regression and support vector machines. Such methods are sometimes called **Interpretable ML**. Perceived trade-off between interpretability and accuracy.
- **Post-hoc explainability** is where an ML model is trained as normal, and then a secondary analysis provides an explanation for why the decision was made (the secondary analysis may have its own inaccuracies independent of the original ML).

Idea: Use XAI tools to generate something like Brown's heuristic (designed by a human expert and considered interpretable).

Explainable AI (XAI)

Explainable AI (XAI) is increasingly important: both as an error check on the ML process; and to build trust in new AI technologies. Two main types of XAI:

- **Ante-hoc explainable models** are transparent by design in their decisions, e.g. linear regression and support vector machines. Such methods are sometimes called **Interpretable ML**. Perceived trade-off between interpretability and accuracy.
- **Post-hoc explainability** is where an ML model is trained as normal, and then a secondary analysis provides an explanation for why the decision was made (the secondary analysis may have its own inaccuracies independent of the original ML).

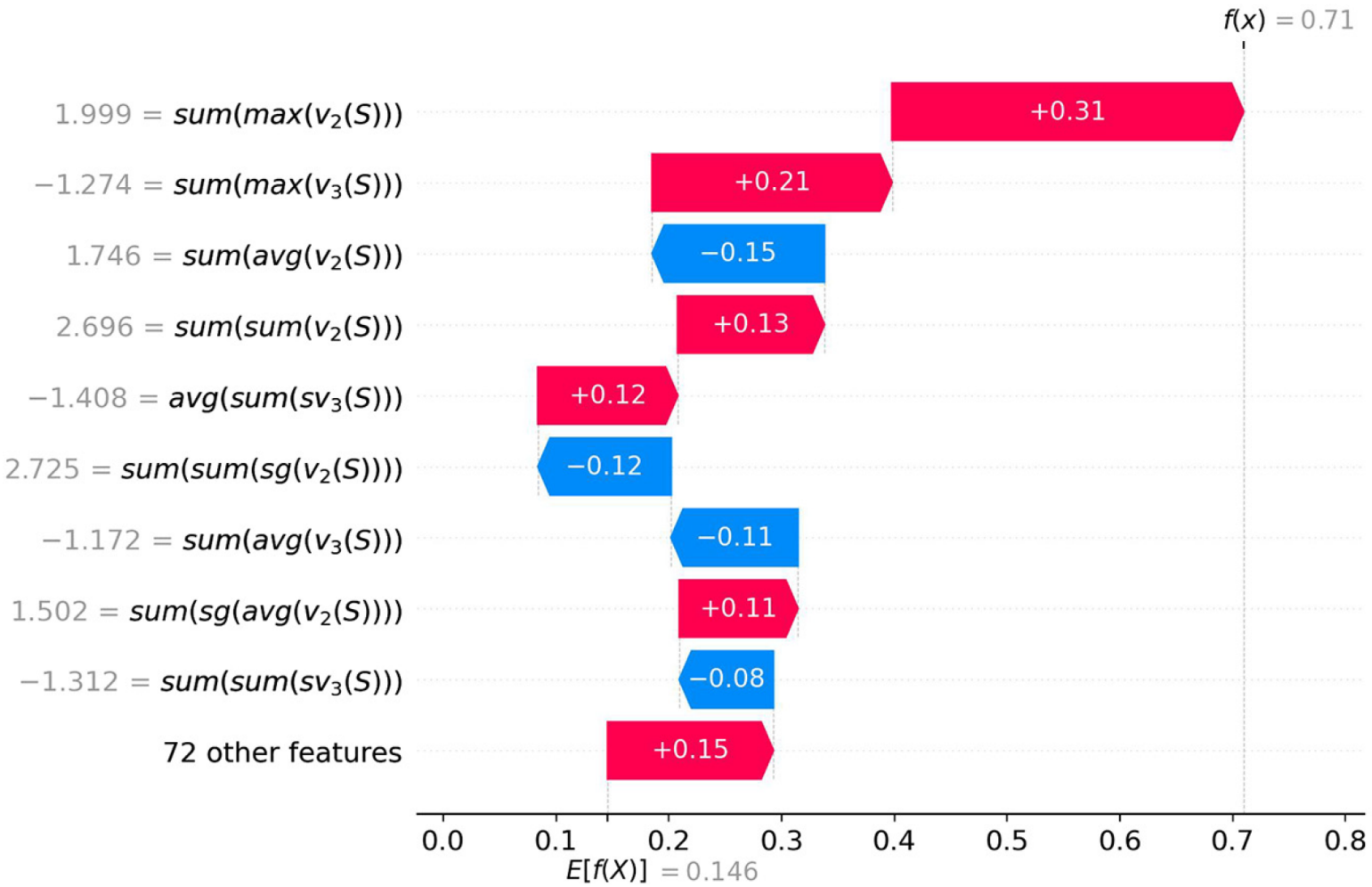
Idea: Use XAI tools to generate something like Brown's heuristic (designed by a human expert and considered interpretable).

Post-hoc XAI for CAD Variable Ordering

We applied the SHAP XAI tool to analyse the features used by ML classifiers to choose the CAD variable ordering.

L. Pickering, T. del Rıio Almajano, M. England and K. Cohen.
Explainable AI Insights for Symbolic Computation: A case study on selecting the variable ordering for cylindrical algebraic decomposition.
Journal of Symbolic Computation, **123**, Article Number 102276, 2024.

Example SHAP Waterfall Plot



What to do with the SHAP output? (Slide 44/50)

- We aggregated the feature importance scores across the dataset for each of our ML models (KNN, MLP, SVM, RF).

This allowed us to see which models used which features – they did not fully agree on feature ranking.

- We had the models "*vote*" (used the Dowdall System) on features to create an overall feature ranking.

Some features identified as impactful had been known before; others were new.

- Took the top 6 voted features and studied all 120 possible ordered triples of them to create variants with the structure of Brown's heuristic.

The very best of these were the top three ranked features in that order!

What to do with the SHAP output? (Slide 44/50)

- We aggregated the feature importance scores across the dataset for each of our ML models (KNN, MLP, SVM, RF).

This allowed us to see which models used which features – they did not fully agree on feature ranking.

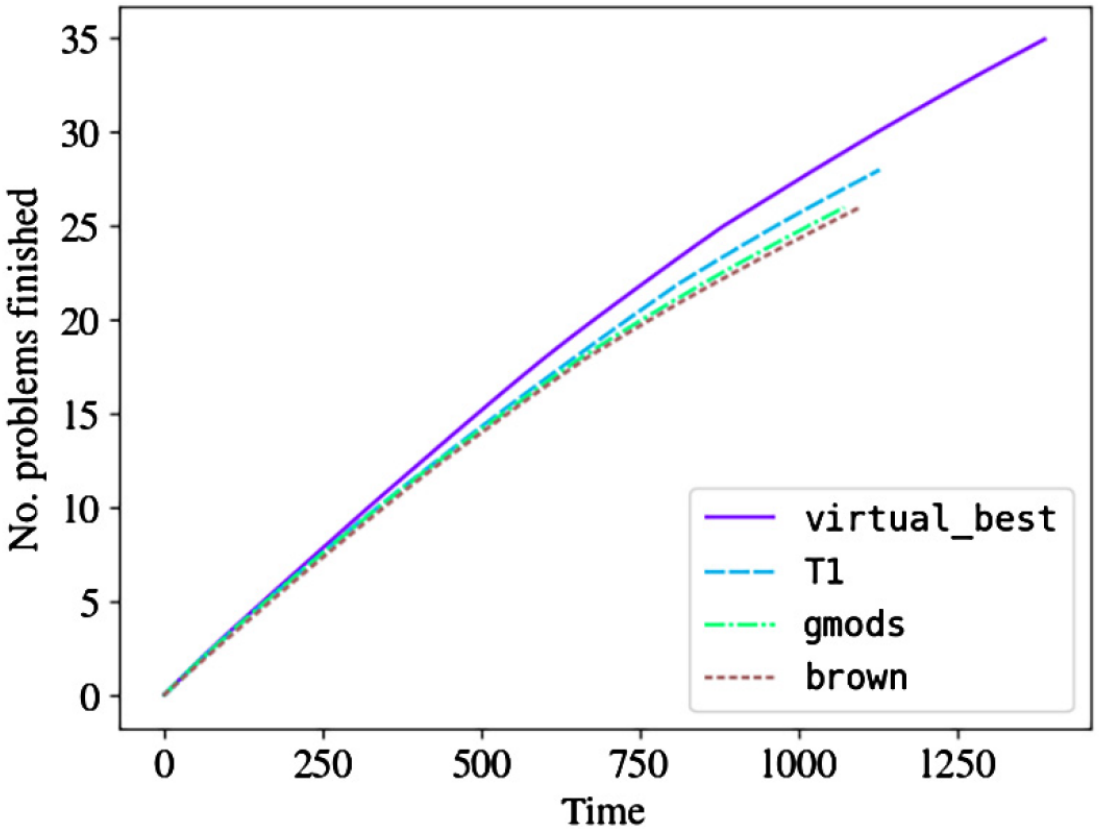
- We had the models "*vote*" (used the Dowdall System) on features to create an overall feature ranking.

Some features identified as impactful had been known before; others were new.

- Took the top 6 voted features and studied all 120 possible ordered triples of them to create variants with the structure of Brown's heuristic.

The very best of these were the top three ranked features in that order!

Performance of the best triple on SMT-LIB



We picked the structure of Brown's heuristic to compare to something known.

There is clearly potential to do even better by explore more nuanced structures.

Ante-hoc XAI for CAD Variable Ordering (Slide 46/50)

The SHAP analysis was post-hoc XAI. Can we use something inherently interpretable?

Our original ML work used fairly simple ML models (SVMs): although considered more interpretable than deep learning they did not offer any easy interpretations to us!

We considered trying to design an interpretable model by, once again, taking inspiration from the Brown heuristic.

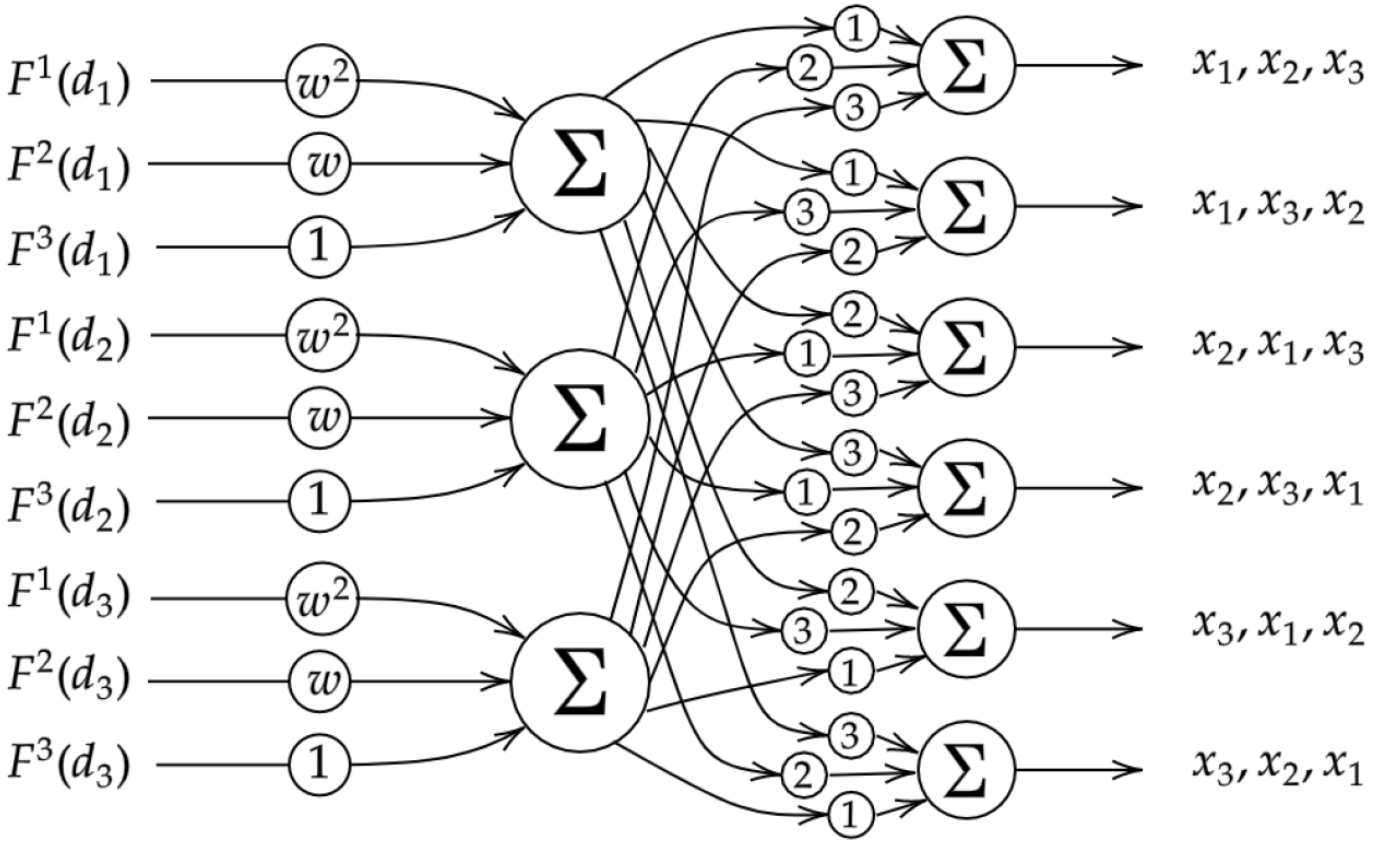
D. Florescu and M. England.

Constrained Neural Networks for Interpretable Heuristic Creation to Optimise Computer Algebra Systems.

Proc. ICMS 2024 (LNCS 14749), pp. 186-195, https://doi.org/10.1007/978-3-031-64529-7_19

Brown Heuristic as Neural Network

Started by formulating Brown's heuristic as a neural network:



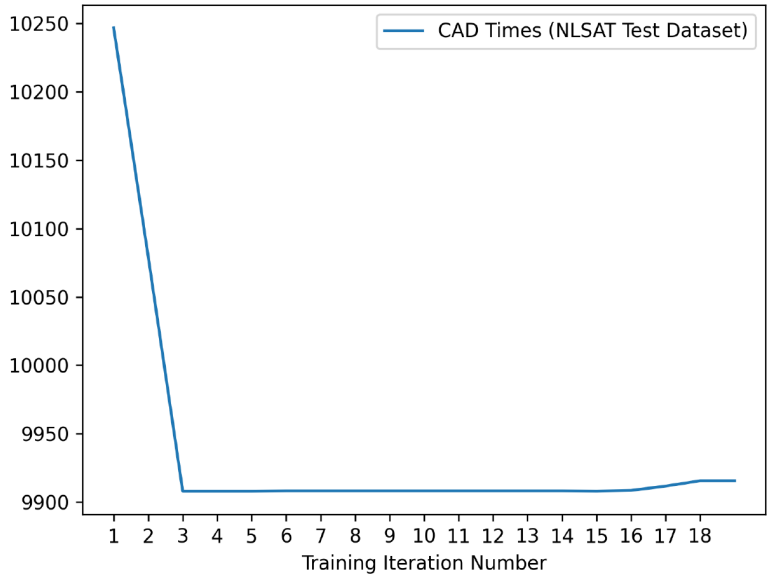
Where w is chosen so that $F^i(d_v) < 2 - 1$ for all features and variables.

Searching the space of similar networks (Slide 48/50)

Then starting from the "*Brown Neural Network*" we optimise by ML training to:

- use different features F^i;
- use different weights.

We can return to something like Brown's heuristic (now linear combinations at each stage).



Dataset Note: In this experiment we trained on polynomials whose exponents and coefficients were created randomly from distributions with the same mean and standard deviation as the SMT-LIB, while testing on the actual SMT-LIB data.

Generalisation over datasets!

Both experiments use XAI techniques to produce heuristics for CAD variable ordering and both have been shown to produce generalisability over datasets:

- In [FE24] we actually trained on random data and and tested on SMTLIB data.
- In [PdREC24] we trained and tested on the SMT-LIB. But subsequent experiments below on very large amounts of random data showed generalisability.

C. Chen, R.-J. Jing, C. Qian, Y. Yuan, and Y. Zhao.
A dataset for suggesting variable orderings for cylindrical algebraic decompositions.
In Proc. CASC 2024, (LNCS 14938), 100–119. Springer, 2024. https://doi.org/10.1007/978-3-031-69070-9_7

Summary of our Two XAI Experiments (Slide 50/50)

Both experiments took a *human-designed* heuristic as a starting point and then used ML processes to produce another heuristic.

The outputs are not human-designed but are *human-level*:

- can be expressed in natural language in a similar amount of text as a heuristic designed by a human;
- can be implemented without any ML architecture.

They could thus be employed by any CAD developer without having to do any ML. Further, they perform almost as well as the ML models themselves and generalise better.

We suggest that these could exemplify a new general methodology for computer algebra heuristic design.

Thanks for Listening

Contact Details

Matthew.England@coventry.ac.uk

https://matthewengland.coventry.domains/